DEPARTAMENT D' INFORMÀTICA

ALGUNAS EXTENSIONES DEL MÉTODO DE VECTORES COMUNES DISCRIMINANTES PARA LA CLASIFICACIÓN DE IMÁGENES.

KATERINE DIAZ-CHITO

UNIVERSITAT DE VALÈNCIA
Servei de Publicacions
2011

Aquesta Tesi Doctoral va ser presentada a València el dia 22 de setembre de 2011 davant un tribunal format per:

- Dra. María del Carmen García Mateo
- Dr. Jesús Martínez del Rincón
- Dr. Luis Baumela Molina
- Dr. Sergio A. Velastin
- Dr. Wladimiro Díaz Villanueva

Va ser dirigida per:
Dr. Francesc J. Ferri

I.S.B.N.: 978-84-370-8518-0
Edita: Universitat de València
Servei de Publicacions
C/ Arts Gràfiques, 13 baix
46010 València
Spain
Telèfon:(0034)963864115

El Dr. **Francesc J. Ferri**, Catedrático de Universidad y miembro del Departament d' Informàtica de l' Escola Tècnica Superior d' Enginyeria de la Universitat de València

CERTIFICA que la memoria titulada "**Algunas Extensiones del Método de Vectores Comunes Discriminantes para la Clasificación de Imágenes**", ha sido realizada, bajo su dirección, por Doña Katerine Diaz Chito, para optar al grado de Doctor en Informática y Matemática Computacional.

Y para que así conste a efectos legales, firma el presente certificado

Fdo. Francesc J. Ferri
Director

Valencia, Septiembre de 2011

Algunas Extensiones del Método de Vectores Comunes Discriminantes para la Clasificación de Imágenes

Tesis Doctoral

Katerine Diaz Chito

Director: Francesc J. Ferri

Valencia, Septiembre de 2011
Departament d'Informàtica

Para mis padres Rolando y Cenide,
para mi hermano Fernando
para mi hermana Kelly

Una de las características que nos define
como humanos es nuestra capacidad de aprendizaje,
así como el hecho de poder disfrutar de ello

Resumen

Los métodos basados en subespacios son una herramienta muy utilizada en aplicaciones de visión por computador. Aquí se presentan y validan algunos algoritmos que hemos propuesto en este campo de investigación. El primer algoritmo está relacionado con una extensión del método de vectores comunes discriminantes con kernel, que hemos llamado el método de vectores comunes discriminantes extendidos con kernel (RKDCV, *Rough Discriminative Common Vector with Kernels*). RKDCV reinterpreta el espacio nulo de la matriz de dispersión intra-clase del conjunto de entrenamiento para obtener las características discriminantes. Este método también se puede ver como una versión no lineal de los vectores comunes discriminantes extendidos.

Dentro de los métodos basados en subespacios existen diferentes tipos de entrenamiento. Uno de los más populares, pero no por ello uno de los más eficientes, es el aprendizaje por lotes. En este tipo de aprendizaje, todas las muestras del conjunto de entrenamiento tienen que estar disponibles desde el inicio. De este modo, cuando nuevas muestras se ponen a disposición del algoritmo, el sistema tiene que ser reentrenado de nuevo desde cero. Una alternativa a este tipo de entrenamiento es el aprendizaje incremental. En los últimos años, y debido a la necesidad de poder actualizar los sistemas de clasificación, muchos de los métodos basados en subespacios, que utilizan el aprendizaje por lotes, han sido modificados

para un aprendizaje incremental. Aquí se proponen diferentes algoritmos incrementales del método de vectores comunes discriminantes.

El método DCV ha sido propuesto originalmente de dos formas diferentes, pero equivalentes. En este documento se formula un algoritmo incremental para cada uno de ellos. Además, se muestra una versión incremental propia para calcular los vectores comunes discriminantes. El primer algoritmo se basa en la eigendescomposición y lo hemos abreviado con las siglas IDCV-EVD del inglés *Incremental Discriminative Common Vectors by Eigendecomposition*. El segundo algoritmo emplea subespacios diferencia y la ortogonalización de Gram-Schmidt incremental (IDCV-GSO). El tercer algoritmo esta basado en subespacios diferencia y ortogonalización (IDCV-O).

Cuando la dimensión del espacio de las muestras es menor o igual a la diferencia entre el tamaño del conjunto de entrenamiento y el número de clases, la dimensión del subespacio discriminante en el método DCV es pequeña. Esto ocasiona que los vectores comunes discriminantes no sean representativos. En este caso, nosotros utilizamos el método de vectores comunes discriminantes extendidos (RDCV, *Rough Discriminative Common Vector*). RDCV relaja las bases del método DCV a partir de ampliar el subespacio discriminante. Aquí se presenta una versión incremental del método RDCV que permite actualizar la base de conocimiento desde un modelo precalculado.

Los algoritmos propuestos se validan empíricamente para diferentes casos de estudio, a partir de clasificar bases de datos que presentan características complejas.

Abstract

Subspace-based methods are a widely used tool in computer vision applications. This document introduces and validates several algorithms that we have proposed in this field of research. The first algorithm is an extension of the discriminative common vector method with kernels, called the Rough Discriminative Common Vector Method with Kernels (RKDCV). RKDCV reinterprets the null space of the within-class scatter matrix of the training set for calculating the discriminative features. This method can also be understood as a nonlinear version of the rough discriminative common vectors.

Among the subspace-based methods, there are different types of training. One of the most popular, although not the most efficient one, is the batch learning. In this type of learning all the samples of the training set must be available at the start of training. If new training samples are available a posterior, the system must be retrained from scratch. An alternative to this type of training is the incremental learning. In the last years, due to the need of updating the classification systems, most of the subspace-based methodologies that were using batch learning, have been modified for an incremental learning. Here we propose different incremental algorithms for discriminative common vector method.

The DCV method was originally proposed of two different but equivalent ways. In this document an incremental

algorithm for each of them is formulated. In addition, we show our own incremental version for calculating the discriminative common vectors. The first algorithm, as well as the method original, is based on the eigendecomposition and we have denominated it with the acronym IDCV-EVD (Incremental Discriminative Common Vectors by Eigendecomposition). The second algorithm uses difference subspaces and Incremental Gram-Schmidt orthogonalization (IDCV-GSO). The third algorithm is based on difference subspaces and orthogonalization (IDCV-O).

When the dimension of the sample space is smaller than or equal to the difference between the training set size and the number of classes, the dimension of the discriminant subspace for the DCV method is small. This produces discriminative common vectors that are not representative. In this case, we use the rough discriminative common vector method (RDCV). RDCV relaxes the bases of DCV method, expanding the size of the discriminant subspace. We present an incremental version of the RDCV method, which allows updating the knowledge base from a precalculated model.

The proposed algorithms in this document are validated empirically for different case studies, by classifying databases that present complex characteristics.

Agradecimientos

*Q*uiero agradecer de corazón a mi director de tesis el Dr. Francesc J. Ferri, gracias por su confianza, apoyo y (sobre todo) paciencia, sin su ayuda esto no hubiera sido posible.

También quiero extender este agradecimiento al Dr. Wladimiro Díaz Villanueva que continuamente me a ofrecido su ayuda.

Gracias a los colegas de café, Adrian, Carlos, Pedro, Ana ... así llueva, haga frío o calor, siempre hay tiempo para un café.

Gracias a Fulvio por su paciencia.

Gracias a mi familia por saber entenderme y apoyarme en todo momento, mil gracias.

Asimismo quiero agradecer al departamento de Informática de la Universidad de Valencia. Al Ministerios de Ciencia e Innovación del Gobierno de España por concederme una beca FPI, asociada al proyecto de investigación Aplicaciones del reconocimiento de formas a la automatización del control de calidad y otros procesos industriales (ARFAI), REF: DPI2006-15542-C04-04. Esta tesis también asido parcialmente soportada por FEDER y el Gobierno de España, desde los proyectos TIN2009-14205-C04-03, ACOMP/2010/287 y Consolider Ingenio 2010 CSD2007-00018.

Índice general

1. Lista de abreviaturas **XIX**

2. Notación Matemática **XXI**

3. Introducción **1**

3.1. Introducción general 1

3.2. Objetivos 6

3.3. Estructura de la Tesis 8

4. Métodos de clasificación basados en subespacios lineales **11**

4.1. Estado del arte 12

4.2. Análisis de componentes principales 14

4.3. Análisis discriminante lineal 17

4.4. Vectores comunes discriminantes 20

4.4.1. DCV a partir de eigendescomposición . 22

4.4.2. DCV a partir de ortogonalización 26

4.5. Vectores comunes discriminantes extendidos . . 28

4.6. Aplicabilidad 32

5. Métodos de clasificación basados en subespacios no lineales **35**

5.1. Conceptos fundamentales 36

5.2. Estado del arte 37

5.3. Análisis de componentes principales con kernel 39

5.4. Vectores comunes discriminantes con kernel . . 41

Contribuciones **45**

6. Vectores comunes discriminantes extendidos con kernel **47**

6.1. Desarrollo teórico 48

6.2. Experimentos 51

6.2.1. Experimentos con pocas muestras . . . 53

6.2.2. Experimentos con gran escala 56

6.3. Discusión 58

7. Vectores comunes discriminantes mediante eigendescomposición incremental **59**

7.1. Preliminares 60

7.1.1. Antecedentes y estado del arte 60

7.1.2. Planteamiento incremental de DCV mediante eigendescomposición 62

7.2. DCV mediante eigendescomposición incremental 64

7.2.1. Desarrollo teórico 64

7.2.2. Coste computacional 69

7.3. Casos particulares del método IDCV-EVD . . . 69

7.3.1. Una nueva muestra por clase 69

7.3.2. Nuevas clases en el conjunto de actualización 71

7.4. Evaluación empírica 72
7.4.1. Aprendizaje incremental con nuevas muestras 73
7.4.2. Nuevas clases 80
7.5. Discusión 83

8. Vectores comunes discriminantes incrementales mediante ortogonalización 85
8.1. IDCV mediante ortogonalización de Gram-Schmidt incremental 85
8.1.1. IDCV-GSO para nuevas muestras de entrenamiento 86
Complejidad del método 89
Validación empírica 89
8.1.2. IDCV-GSO que incorpora nuevas clases al conjunto de entrenamiento 91
Complejidad del método 92
Validación empírica 92
8.2. IDCV mediante subespacios diferencia y ortogonalización 94
8.2.1. Complejidad del método 96
8.2.2. Validación IDCV-O 97
8.3. Discusión 99

9. Vectores comunes discriminantes extendidos de forma incremental 101
9.1. Planteamiento 102
9.2. Desarrollo teórico 103
9.2.1. Antecedentes 103

9.2.2. RDCV incremental 104
Una muestra por clase 109
Nuevas clases 110
9.3. Evaluación empírica 111
9.4. Discusión 114

10. Conclusiones **117**
10.1. Publicaciones 120
10.2. Proyectos 121

Conclusion **123**

A. Conceptos **127**
A.1. Conceptos Fundamentales 127
A.2. Propiedades de los valores y vectores propios . 129
A.3. Subespacios 130
A.4. Eigendescomposición desde una matriz más pequeña 131
A.5. Criterio de Fisher 132
A.6. Funciones kernel 133

B. Descomposición de la matriz de dispersión intra-clase **135**

C. Teoremas **139**

D. Bases de datos **147**
D.1. Bases de datos de rostros 147
D.2. Bases de datos de objetos 150
D.3. Base de datos de dígitos escritos a mano 152

E. Resultados numéricos **153**
E.1. Validación IDCV-EVD 153
E.2. Validación IDCV-GSO 159
E.3. Validación IDCV-O 164
E.4. Validación IRDCV 169

Capítulo 1

Lista de abreviaturas

CPU	*Central Processing Unit*
CV	*Common Vectors*
DCV	*Discriminative Common Vectors*
DCV-EVD	*Discriminative common vectors from eigendecomposition*
DCV-GSO	*Discriminative common vectors from Gram-Schmidt orthonormalization*
EVD	*EigenValue Decomposition*
LDA	*Linear discriminant analysis*
GSO	*Gram-Schmidt Orthonormalization*
IDCV	*Incremental Discriminative Common Vectors*
IDCV-EVD	*Incremental discriminative common vectors from eigendecomposition*
IDCV-GSO	*Incremental discriminative common vectors from Gram-Schmidt orthonormalization*
IDCV-O	*Incremental discriminative common vectors from orthonormalization*
IRDCV	*Incremental Rough Discriminative Common Vectors*
K-NN	*K Nearest Neighbors*
KDCV	*Discriminative Common Vector with Kernels*
KPCA	*Kernel Principal Component Analysis*
RKDCV	*Rough Discriminative Common Vector with Kernels*
NFLD	*Null space based Fisher's linear discriminant*
PCA	*Principal Components Analysis*
RCV	*Rough Common Vectors*

RDCV	*Rough Discriminative Common Vectors*
seg.	segundos

Capítulo 2

Notación Matemática

Para lograr una comprensión adecuada del trabajo, aquí se define la mayor parte de la nomenclatura utilizada a lo largo de la tesis. Los vectores se denotan con letras minúsculas romanas tales como x, y todos son considerados vectores columna de dimensión d si no se especifica lo contrario. Las matrices se indican con letras mayúsculas romanas como X y están conformadas por vectores columna.

Los conjuntos de datos se denotan por letras mayúsculas de estilo boldface como $\mathbf{X}$. Las letras mayúsculas en estilo itálica, como S, indican matrices de dispersión. Las letras caligráficas mayúsculas representan subespacios. A lo largo de este documento se trata de mantener la tipografía descrita anteriormente.

En los siguientes capítulos, $\mathbf{X} \in \mathbb{R}^{d \times M}$ representa un conjunto de entrenamiento compuesto por M muestras, donde cada muestra es un vector columna $\mathrm{x}_j^i \in \mathbb{R}^d$ de dimensión d. El subíndice i hace referencia a las muestras dentro de la clase j, $j = 1, \ldots, c$. El numero de clases en el conjunto de entrenamiento es c, y el vector media de la clase j es x_j.

$\mathbf{X}$	conjunto de entrenamiento inicial, $\mathbf{X} \in \mathbb{R}^{d \times M}$
$\mathbf{Y}$	conjunto de entrenamiento incremental, $\mathbf{Y} \in \mathbb{R}^{d \times N}$

$\mathbf{Z}$	conjunto de entrenamiento concatenado, $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}] \in \mathbb{R}^{d\times(M+N)}$
W	matriz de proyección final
d	dimensión
c, $j = 1, \ldots, c$	número de clases en el conjunto de entrenamiento
m_j	número de muestras de la clase j en $\mathbf{X}$
n_j	número de muestras de la clase j en $\mathbf{Y}$
$M = \sum_{j=1}^{c} m_j$	número total de muestras en $\mathbf{X}$
$N = \sum_{j=1}^{c} n_j$	número total de muestras en $\mathbf{Y}$
$\mathrm{x}_j^i,\ \mathrm{y}_j^i,\ \mathrm{z}_j^i$	vectores que representan la i-ésima muestra de la clase j
$\mathrm{x}_j,\ \mathrm{y}_j,\ \mathrm{z}_j$	vectores media de la clase j en $\mathbf{X}$, $\mathbf{Y}$ y $\mathbf{Z}$, respectivamente
$\mathrm{x}_{cv}^j,\ \mathrm{z}_{cv}^j$	vector común de la clase j en $\mathbf{X}$ y $\mathbf{Z}$, respectivamente
$\mathrm{x}_{cv},\ \mathrm{z}_{cv}$	vector media de los vectores comunes x_{cv}^j y z_{cv}^j, respectivamente
S_{cv}	matriz de dispersión de los vectores comunes
$\mathrm{x}_{dcv}^j,\ \mathrm{z}_{dcv}^j$	vector común discriminante de la clase j en $\mathbf{X}$ y $\mathbf{Z}$, respectivamente
$\mathrm{x}_{rcv}^j,\ \mathrm{z}_{rcv}^j$	vector común extendido la clase j en $\mathbf{X}$ y $\mathbf{Z}$, respectivamente
$\mathrm{x}_{rcv},\ \mathrm{z}_{rcv}$	vector media de los vectores comunes extendidos x_{rcv}^j y z_{rcv}^j, respectivamente
S_{rcv}	matriz de dispersión de los vectores comunes extendidos
$\mathrm{x}_{rdcv}^j,\ \mathrm{z}_{rdcv}^j$	vector común discriminante extendido de la clase j en $\mathbf{X}$ y $\mathbf{Z}$, respectivamente
$\mathcal{R}(\cdot)$	espacio rango de $(\cdot)$
$\mathcal{N}(\cdot)$	espacio nulo de $(\cdot)$
$\tilde{\mathcal{R}}(\cdot)$	espacio rango extendido de $(\cdot)$
$\tilde{\mathcal{N}}(\cdot)$	espacio nulo de reducido de $(\cdot)$
r	dimensión del espacio rango
$n = (d - r)$	dimensión del espacio nulo
$\mathcal{B}$	espacio diferencia
$\mathcal{B}^{\perp}$	complemento ortogonal de $\mathcal{B}$ (espacio indiferencia)
$\mathcal{B}_{cv}$	subespacio diferencia de los vectores comunes
$\mathcal{B}_{rcv}$	subespacio diferencia de los vectores comunes extendidos

Φ	mapping
ϕ	función de mapeo
K	matriz kernel
K_c	matriz kernel centrada
K^{test}	matriz kernel del conjunto de test
K_c^{test}	matriz kernel del conjunto de test centrada
$<,>$	producto escalar
$\mid \mid$	determinante
$\| \|$	norma
$span\{\ \}$	generador de un subespacio
$\oplus$	suma directa de subespacios
$O(\cdot)$	notación asintótica para costes

Capítulo 3

Introducción

Resumen — En el área del reconocimiento y clasificación de patrones existen múltiples métodos como formas de aprender. Este capítulo ofrece una visión general de los métodos basados en subespacios en el contexto de clasificación de patrones. En él, también se expondrán los objetivos principales de la tesis y su justificación. La parte final del capítulo muestra la estructura que sigue este documento.

Contenido

3.1. Introducción general 1
3.2. Objetivos 6
3.3. Estructura de la Tesis 8

3.1. Introducción general

El proceso de reconocer cosas, como un sonido, una imagen, un amigo o un sabor, etc. . . es algo que hacemos de forma inconsciente. Este proceso es una de las funciones mentales más importantes en seres humanos y animales en el que se adquieren nuevas habilidades, destrezas, conocimientos o conductas, como resultado del aprendizaje.

En los sistemas de visión por computador el aprendizaje automático (del inglés *machine learning*) es de gran utilidad, debido a su gran potencial para contribuir en el desarrollo de algoritmos que sirven para mejorar el desempeño de los sistemas de visión, los cuales tienen un alto nivel de complejidad [98].

El reconocimiento automático, la descripción, la clasificación, y el agrupamiento de patrones son aspectos importantes en la ingeniería y ciencias, como biología, psicología, medicina, marketing, visión por computador e inteligencia artificial, entre otros. El reconocimiento de patrones es el estudio de cómo las máquinas pueden observar el ambiente, aprender a distinguir patrones de interés de su trasfondo, y producir salidas o tomar decisiones razonables acerca de las categorías de los patrones [59].

En el área del reconocimiento de patrones y análisis de imágenes, varias técnicas han sido propuestas en los últimos años motivadas por el número creciente de aplicaciones en el mundo real. La tabla 3.1 muestra algunas aplicaciones.

En el proceso de construir un sistema de reconocimiento se pueden distinguir dos etapas: entrenamiento (aprendizaje) y clasificación (pruebas). Cada etapa se compone de funciones diferentes, como son la adquisición de datos, el preprocesamiento, la presentación de datos, el aprendizaje, la toma de decisiones. La figura 3.1 muestra un modelo estándar para el reconocimiento estadístico de patrones.

Las funciones del módulo de preprocesamiento son segmentar el patrón de interés del trasfondo, quitar ruido, normalizar el patrón, entre otras operaciones que contribuyan a dar una representación compacta del patrón. En el entrenamiento, la función del módulo de extracción/selección de características consiste en encontrar las características apropiadas para representar el patrón de entrada. El camino de información retroactiva permite optimizar el preprocesamiento y las estrategias de extracción/selección de características.

Tabla 3.1: Ejemplos de aplicaciones del reconocimiento de patrones.

Dominio del problema	Aplicación	Patrón de entrada	Clases de patrón
Bioinformática	análisis de secuencias	DNA/secuencias de proteínas	conocer el tipo de genes/patrones
Minería de datos	buscar patrones significativos	puntos en un espacio multidimensional	obtener grupos compactos y bien separados
Clasificación de documentos	buscar en Internet	texto del documento	categoría semánticas
Análisis de imágenes de documentos	caracteres alfanuméricos, palabras	imagen del documento	caracteres alfanuméricos, palabras
Automatización Industrial	inspección de circuitos impresos	intensidad o rango de la imagen	producto defectuoso /no defectuoso
Recuperación de bases de datos de multimedia	búsqueda en internet	vídeo clip	genero de los vídeos
Reconocimiento biométrico	identificación de personas	rostros, iris	autorización de usuarios para accesos de control
Reconocimiento de la voz	exploración del directorio telefónico sin la asistencia del operador	voz	palabras habladas

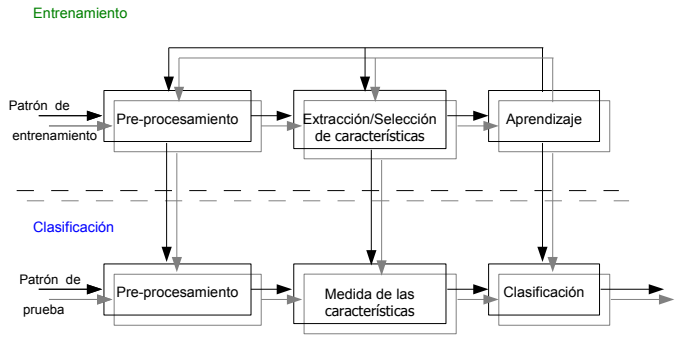


Figura 3.1: Modelo estándar para el reconocimiento estadístico de patrones.

La función del módulo de clasificación es asignar el patrón de entrada a uno de los patrones de entrenamiento a partir de las características medidas.

Los métodos de clasificación más conocidos en reconocimiento de patrones están relacionados con coincidencia de patrones, clasificación estadística, coincidencia sintáctica o estructural, y redes neurales. Estos métodos no son necesariamente independientes y algunas veces el mismo método de reconocimiento de patrones posee interpretaciones diferentes. Una breve descripción y algunas características de estos métodos se presentan en la tabla 3.2.

Tabla 3.2: Métodos de clasificación en reconocimiento de patrones.

Métodos de clasificación basados en	**Representación**	**Función de reconocimiento**	**Criterio típico**
Coincidencia de patrones	muestras, píxeles, curvas	correlación, medidas de distancia	error de clasificación
Estadística	características	función discriminante	error de clasificación
Coincidencia sintáctica o estructural	primitiva	reglas, gramática	error de aceptación
Redes neuronales	muestras, píxeles, curvas	función neurona	error cuadrático medio

Los métodos estadísticos, basados en subespacios, se han convertido en una herramienta estándar en la comunidad de visión por computador para realizar diversos tipos de aprendizaje a partir de información visual y para tareas de reconocimiento/clasificación.

Esta tesis está enfocada en algunos métodos estadísticos también llamados métodos basados en subespacios. En un subespacio, determinado espacio de representación, un patrón se representa con una agrupación de d características visto como un punto en un espacio d-dimensional. La idea principal de estos métodos es obtener las características de un nuevo subespacio obtenido a partir del anterior, donde los elementos

que pertenecen a cada clase se puedan diferenciar de forma más clara que en el espacio original del patrón. La efectividad del espacio de representación (conjunto de características) se determina por la separabilidad de los patrones de diferentes clases [39].

Desde el punto de vista de la visión por computador, el aprendizaje automático puede ofrecer métodos efectivos para adquirir modelos visuales, adaptar parámetros de la representación, transformar señales a símbolos, construir sistemas de procesamiento de imágenes entrenables, centrar la atención en el objeto fijado, etcétera [98].

En pocas palabras, con el aprendizaje automático se desarrollan técnicas que permitan a las computadoras aprender. Se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción. En muchas ocasiones, el campo de actuación del aprendizaje automático se solapa con el de la estadística, ya que las dos disciplinas se basan en el análisis de datos [46].

Existen distintas técnicas que permiten a las computadoras aprender. Una de las técnicas más comunes es el aprendizaje por lotes o *batch learning*, donde el proceso de aprendizaje en tales sistemas consiste en procesar repetidas veces el conjunto de entrenamiento como un todo, hasta obtener un modelo final que describe con la mayor exactitud posible al mayor número de ejemplos de entrenamiento. Para ello, es necesario aceptar como válidos tres supuestos:

1. todos los ejemplos necesarios para describir el dominio del problema están disponibles antes del proceso de aprendizaje,
2. todos los ejemplos de entrenamiento pueden ser cargados en memoria,
3. tras procesar debidamente el conjunto de entrenamiento, el aprendizaje puede darse por finalizado.

A pesar de su enorme campo de aplicación, las técnicas de aprendizaje por lotes no son convenientes en un gran número de problemas donde los datos proceden de entornos dinámicos y/o van siendo adquiridos a lo largo del tiempo. Por ejemplo, en un sistema de seguridad basado en la clasificación de rostros, cuando nuevas imágenes se ponen a disposición del sistema la memoria requerida y el coste computacional son muy altos si el sistema es reentrenado, cada vez, desde cero a partir del aprendizaje por lotes. Una alternativa a este problema es realizar un aprendizaje incremental.

Los métodos de aprendizaje incremental presentan ciertas características que los hacen muy apropiados para trabajar con grandes cantidades de datos, como son, su capacidad de incorporar nuevas experiencias a la base de conocimiento, y su capacidad de evolucionar esta base de conocimiento desde una estructura sencilla hacia otra más compleja.

El aprendizaje incremental se ha convertido en los últimos años en un tema importante en visión por computador, ya que los entornos están cambiando continuamente y, prácticamente, el supuesto de disponer de un conjunto de entrenamiento completo no se da por adelantado [67].

Un sistema con aprendizaje incremental normalmente se caracteriza porque puede aportar nueva información al sistema sin tener explícitamente acceso a los viejos datos. Esto motiva técnicas para actualizar de forma incremental los métodos basados en subespacios cuando más datos se ponen a disposición del sistema.

3.2. Objetivos

En esta tesis se desarrollan diferentes algoritmos de clasificación basados en subespacios que presentan distintos enfoques. Si bien, el método KDCV presenta en algunas aplicaciones características de clasificación aceptables, en otras

aplicaciones donde el grado de dificultad es mayor su criterio discriminante no es suficiente. Por esto, el primer objetivo de esta tesis es desarrollar un método de clasificación basado en una extensión del método de vectores comunes discriminantes con kernel (KDCV) que permita un mayor grado de generalización.

El método propuesto toma el nombre de vectores comunes discriminantes extendidos con kernel y se abrevia con las siglas RKDCV, del inglés *Rough Discriminative Common Vector with Kernels*. Este método es una generalización del método KDCV, que también puede verse como la versión no lineal del método de vectores comunes discriminantes extendidos.

En los últimos años, los sistemas de clasificación están cambiando continuamente y procesando cada vez más información, lo cual ha dado gran fuerza a los métodos con aprendizaje incremental. El segundo objetivo de la tesis se relaciona con este aspecto, y busca proponer diferentes algoritmos incrementales de algunos métodos basados en subespacios.

Desde estudios anteriores se observa que el método de vectores comunes discriminantes (DCV) presenta características muy apropiadas para tareas de clasificación de datos que presentan alta dimensión, y motivados por estas características se desarrollan algoritmos incrementales de este método.

Como último objetivo concreto, y debido a algunas limitaciones del método DCV respecto al tamaño y a la dimensión del espacio original de las muestras, se presenta un algoritmo incremental del método de vectores comunes discriminantes extendidos.

En síntesis, los objetivos son

- Desarrollar una versión extendida del método de vectores comunes discriminantes con kernel.
- Desarrollar diferentes algoritmos incrementales del método DCV, que incorporen nuevas muestras o nuevas clases a la base de conocimiento.

- Desarrollar un algoritmo incremental del método de vectores comunes discriminantes extendidos.
- Implementar los algoritmos desarrollados.
- Evaluar los algoritmos respecto a tareas de clasificación de imágenes que presentan diferentes características.

De este modo, los objetivos principales son implementar, evaluar y comparar los diferentes algoritmos de clasificación, basados en subespacios, que se proponen y se presentan en este documento.

3.3. Estructura de la Tesis

La estructura de este documento se define por capítulos. El capítulo 3 presenta una introducción general sobre la tesis, así como los objetivos y la estructura de la misma.

El capítulo 4 describe los principales métodos basados en subespacios lineales que son relevantes en nuestro caso de estudio, como son el análisis de componentes principales, el análisis discriminante lineal, el método de vectores comunes discriminantes, y el método de vectores comunes discriminantes extendidos.

El capítulo 5 explica algunos métodos basados en subespacios no lineales como el análisis de componentes principales con kernel y el método de vectores comunes discriminantes con kernel.

El capítulo 6 presenta la primera aportación relacionada con una extensión del método de vectores comunes discriminantes con kernel. Este capítulo muestra la validación empírica de la extensión propuesta para diferentes casos de estudio.

La primera aportación en cuanto a algoritmos incrementales se presenta en el capítulo 7. Aquí se expone el concepto y las bases matemáticas del algoritmo incremental de vectores

comunes discriminantes propuesto a partir de descomposición de valores y vectores propios. Este capítulo también muestra los resultados y el análisis de validar el algoritmo incremental en cuanto a la tasa de acierto y al tiempo de CPU, cuando nuevas muestras y clases se incorporan al conjunto de entrenamiento. El capítulo 7 además presenta la metodología principal de la validación empírica de los algoritmos incrementales.

El capítulo 8 expone, valida y analiza dos algoritmos incrementales del método de vectores comunes discriminantes, pero ahora a partir de subespacios diferencia y ortogonalización. Una versión incremental del método de vectores comunes discriminantes extendidos se describe y valida en el capítulo 9.

Este documento finaliza con el capítulo 10 que presenta las principales conclusiones a partir de los resultados obtenidos, y muestra las lineas de investigación futuras. Este capítulo también contiene las publicaciones a las que ha dado lugar el trabajo realizado en este documento.

Por último están los apéndices, que incluyen varios conceptos fundamentales, teoremas, la descripción de las bases de datos, y algunos resultados numéricos.

Capítulo 4

Métodos de clasificación basados en subespacios lineales

Resumen — Los métodos de clasificación basados en subespacios lineales emplean proyecciones ortogonales que transforman el espacio original de las muestras en un subespacio, normalmente de menor dimensión, donde se espera que las muestras sean separables linealmente. El modelo de transformación está dado por $\mathrm{T} = \mathrm{W}^T\mathbf{X}$, donde T es la matriz de datos en el espacio transformado de menor dimensión, $\mathbf{X}$ es la matriz de datos en el espacio original y W^T es la matriz de transformación. Este capítulo presenta el estado del arte de los métodos basados en subespacios lineales, y describe los métodos lineales que son relevantes en este documento.

Contenido

4.1. Estado del arte 12
4.2. Análisis de componentes principales . . . 14
4.3. Análisis discriminante lineal 17
4.4. Vectores comunes discriminantes 20
4.5. Vectores comunes discriminantes extendidos 28
4.6. Aplicabilidad 32

4.1. Estado del arte

Uno de los métodos lineales no supervisado más popular en reconocimiento de patrones, que se emplea para reducir la dimensión del espacio de entrada, es el Análisis de componentes principales (PCA, *Principal Components Analysis*) también llamado Transformada de Hotelling [14]. Su origen suele asociarse a la publicación de un artículo de K. Pearson en 1901 [65]. Sin embargo, el nombre de componentes principales y su primer desarrollo teórico no aparece hasta 1933, en un artículo de Hotelling [55]. En 1946 Kari Karhunen [64] y en 1955 Michel Loéve [79], le dieron el nombre de trasformada de Karhunen-Loéve o trasformada discreta de Karhunen-Loéve. Múltiples modificaciones y extensiones de este método se han propuesto, como son el análisis de componentes principales complejo (*complex-PCA*) [123], el PCA de dos dimensiones (*two-dimensional PCA*) [124], entre muchos otros.

Si bien el origen del PCA es no supervisado, en los últimos años se han formulado algunas versiones supervisadas que son una generalización del PCA original. La idea principal de estos métodos es buscar un subespacio donde la dependencia entre predictores y las variables de respuesta sea máxima [8, 9].

Para describir la relación lineal entre dos conjuntos de variables, Hotelling introduce en el año 1936 en [56] el análisis de correlación canónica (CCA, *Canonical Correlation Analysis*). El CCA es un método de aprendizaje no supervisado que busca una transformación lineal ortogonal que maximiza la correlación de las variables.

Entre los métodos lineales supervisados clásicos en reconocimiento de patrones, para la extracción/selección de características, está el análisis discriminante lineal (LDA, *Linear Discriminant Analysis*) también conocido como el discriminante lineal de Fisher (FLD, *Fisher's Linear Discriminant*). El LDA busca maximizar la distancia entre clases mientras

minimiza la distancia entre las muestras que pertenecen a la misma clase. Este método se desarrolló en el año 1936 por el biólogo y estadístico británico Ronald Aylmer Fisher [35]. Su campo de aplicación es amplio y se utiliza en tareas de clasificación de rostros [11, 73], gestos [114], música [3], iris [32], etcétera.

A lo largo de los años se han realizado múltiples modificaciones del método LDA, como en [89, 74, 106, 61, 117]. A partir del LDA también se han generado otros métodos como el análisis discriminante regularizado (RDA, *Regularized Discriminant Analysis*) [38], el criterio de márgenes máximo (MMC, *Maximal Margin Criterion*) [48], el análisis discriminante exponencial [128], entre varios.

Otro método lineal de extracción/selección de características es el análisis de componentes independientes (ICA, *Independent Component Analysis*). ICA busca una trasformación lineal ortogonal que minimiza la dependencia estadística entre los patrones. Este método es más apropiado cuando los patrones tienen una distribución no gaussiana. Su origen se sitúa aproximadamente en el año 1986 y se atribuye a Herault y Jutten [25].

En el año 1999 tuvo origen el método lineal supervisado de vectores comunes (CV, *Common Vector*). Este método fue propuesto por Gulmezoglu et al. para problemas de reconocimiento de palabras aisladas [44]. CV extrae del conjunto de entrenamiento las propiedades comunes de cada clase, eliminando las diferencias entre las muestras de una misma clase. CV se utiliza si la dimensión del espacio de muestras es mayor o igual al número de muestras de entrenamiento por clase. En reconocimiento de patrones también se emplea para el reconocimiento de rostros como en [53, 33].

En el año 2004 Cevikalp et al. proponen el método de los vectores comunes discriminantes (DCV, *Discriminative Common Vectors*) para el reconocimiento de rostros [22, 21]. DCV es un método lineal supervisado basado en el criterio discri-

minante lineal de Fisher modificado [§A.5]. DCV se utiliza cuando la dimensión del espacio de muestras es mayor a la diferencia entre el número de muestras de entrenamiento total y el número de clases. Además de usarse en tareas de reconocimiento de rostros, se emplea en tareas de reconocimiento de expresiones faciales [113], siluetas [119], etcétera [47]. En los últimos años varias modificaciones del método de vectores comunes discriminantes se han propuesto, como en [34, 29, 133, 71, 26].

Tamura et al. proponen en el año 2007 una extensión del método CV, llamada vectores comunes extendidos (RCV, *Rough Common Vector*), para resolver algunos problemas de los métodos CV y LDA [105]. El método RCV reinterpreta el espacio nulo de la matriz de dispersión intra-clase para obtener las características discriminantes.

Recientemente, técnicas de aprendizaje de subespacios multilineales se han expuesto. La reducción de la dimensión de los datos multidimensionales se realiza a partir de representar los datos como tensores. La investigación de estos métodos ha avanzado en pocos años, desde la exploración heurística para investigación sistemática [80] a algoritmos supervisados y no supervisados [121, 49].

4.2. Análisis de componentes principales

El análisis de componentes principales (PCA), es un método estadístico no supervisado usado en tareas de reducción de la dimensionalidad, compresión, extracción de características y visualización, que preserva al máximo la variabilidad de las muestras en el espacio original [55, 62, 15].

Una alta correlación de los vectores de entrada implica información redundante. El método PCA disminuye la redundancia mediante la incorrelación de dichos vectores. Por

lo tanto, los vectores de entrada de dimensión elevada y alta correlación pueden ser eficientemente representados en un espacio de dimensión inferior cuyos vectores no están correlados [62].

El método PCA pretende analizar la dispersión de las muestras, detectando las variables que son responsables de dicha dispersión y facilita el estudio de las relaciones que existen entre las muestras.

El PCA puede ser definido como una proyección ortogonal de los datos en un subespacio lineal de menor dimensión donde la varianza de los datos proyectados es máxima [55, 15]. El proceso de proyección ortogonal se muestra intuitivamente en la figura 4.1.

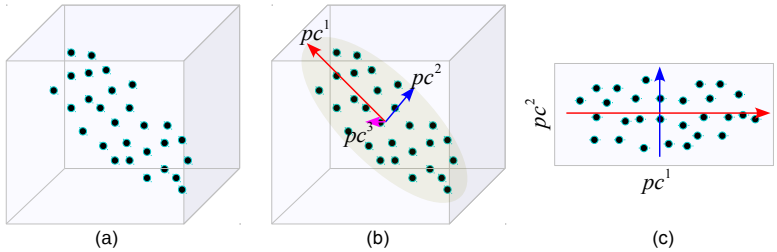


Figura 4.1: PCA busca un subespacio de menor dimensión donde el conjunto de entrenamiento tiene la máxima varianza. (a) conjunto de entrenamiento. (b) componentes principales ordenadas por varianza. (c) proyección del conjunto de entrenamiento utilizando las primeras componentes principales.

Las nuevas variables son combinaciones lineales de las anteriores y se construyen según el orden de importancia en cuanto a la varianza total de las muestras. Los nuevos vectores de características x_{pca}^k se definen mediante la transformación lineal:

$$\mathrm{x}_{pca}^k = \mathrm{W}_{pca}^T \mathrm{x}^k \qquad k = 1, \ldots, M \tag{4.1}$$

$\mathrm{W}_{pca} \in \mathbb{R}^{d \times r}$ es una matriz con columnas ortonormales, donde $r \leq d$.

La matriz de dispersión total, $S_T \in \mathbb{R}^{d \times d}$, está definida como

$$S_T = \sum_{i=1}^{M} (\mathrm{x}^i - \mathrm{x}_t)(\mathrm{x}^i - \mathrm{x}_t)^T \tag{4.2}$$

donde x_t es el vector media de todas las muestras en $\mathbf{X}$.

El PCA busca maximizar la dispersión total de las muestras proyectadas, de manera que

$$\begin{aligned} \mathrm{W}_{pca} &= \underset{\|\mathrm{w}\|=1}{argmax} \, |\mathrm{W}^T S_T \mathrm{W}| \\ &= [\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_r] \end{aligned} \tag{4.3}$$

El teorema 1, del apéndice C, muestra que el criterio dado en (4.3) es máximo para un valor r dado si los vectores columna que conforman W_{pca} son los primeros r vectores propios asociados a los valores propios de S_T organizados de forma descendente. Si los datos del conjunto de entrenamiento son linealmente independientes, se cumple que $r = (M - 1)$.

En el PCA el entrenamiento se da por finalizado una vez se obtienen las componentes principales. Con respecto a una nueva muestra centrada x^{test}, ésta se proyecta al subespacio transformado de la forma:

$$\mathrm{x}_{pca}^{test} = \mathrm{W}_{pca}^T \, \mathrm{x}^{test}$$

A partir de las muestras proyectadas, las distancias mínimas entre las componentes principales del conjunto de entrenamiento y de prueba se utilizan normalmente para la clasificación.

En reconocimiento de imágenes, la dimensión de las muestras suele ser grande, lo cual implica que el tamaño de la matriz de dispersión total S_T es también grande $(d \times d)$. En consecuencia, la descomposición en valores y vectores propios es costosa.

Si la dimensión del espacio original es mayor al número de muestras de entrenamiento, $d > M$, los vectores propios vinculados a los valores propios no nulos de cualquier matriz de dispersión se pueden calcular a partir de una matriz de menor tamaño [86]. Los detalles de este procedimiento se muestran en el apéndice A.4 (página 131).

4.3. Análisis discriminante lineal

El análisis discriminante lineal (LDA, *linear discriminant analysis*), también conocido como el discriminante lineal de Fisher (FLD, *Fisher's Linear Discriminant*), es un método de clasificación supervisado clásico en reconocimiento de patrones, desarrollado en el año 1936 por el biólogo y estadístico británico Ronald Aylmer Fisher [35].

El LDA computa una transformación lineal que minimiza la dispersión dentro de cada clase del conjunto de entrenamiento mientras maximiza la dispersión entre las clases. La figura 4.2 muestra intuitivamente la idea principal del método LDA.

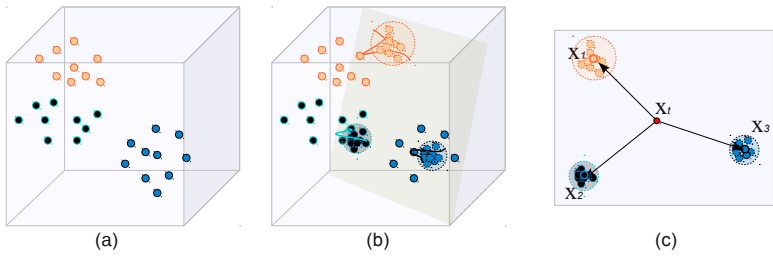


Figura 4.2: El LDA busca minimizar la distancia entre muestras que pertenecen a la misma clase mientras maximiza la distancia entre clases. (a) Conjunto de entrenamiento con 3 clases. (b) Proyección del conjunto de entrenamiento al subespacio de dimensión $\mathbb{R}^{d\times(c-1)}$, (c) donde la dispersión dentro de cada clase es mínima y la distancia entres las medias de cada clase, x_j, y la media global, x_t, es máxima.

El objetivo del LDA es encontrar una matriz de proyección, W_{lda}, que maximice:

$$\begin{aligned} \mathrm{W}_{lda} &= \underset{\|\mathrm{w}\|=1}{argmax} \frac{|\mathrm{W}^T S_B \mathrm{W}|}{|\mathrm{W}^T S_W \mathrm{W}|} \\ &= [\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_r] \end{aligned} \tag{4.4}$$

donde, S_B y S_W son las matrices de dispersión ínter e intra clase, respectivamente, dadas por

$$S_B = \sum_{j=1}^{c} (\mathrm{x}_j - \mathrm{x}_t)(\mathrm{x}_j - \mathrm{x}_t)^T \tag{4.5}$$

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{m_j} (\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T \tag{4.6}$$

x_j es el vector media de la clase j, y x_t es el vector media de todas muestras.

La función que se maximiza en la ecuación (4.4) se conoce como el criterio de Fisher. Si S_W es no singular, el criterio de Fisher es máximo cuando los vectores de la matriz de transformación W_{lda} son los vectores propios de $(S_W^{-1} S_B)$ [30, 15, 39].

Sin embargo, en aplicaciones relacionadas con el procesamiento de imágenes normalmente el tamaño del conjunto de entrenamiento es menor que la dimensión de las muestras, y como consecuencia la matriz S_W es singular, es decir, no tiene inversa. Este problema se conoce como **el problema del tamaño de muestras pequeño** (*small sample size problem*) [39].

En la práctica, el PCA [§4.2] se utiliza comúnmente como un paso de preprocesamiento para llevar a cabo el LDA [11]. Primero se emplea el PCA para reducir la dimensión del conjunto de entrenamiento, y de este modo la matriz de dispersión intra-clase es no singular en el espacio transformado. Luego, se aplica el método LDA. Esta estrategia es conocida como PCA+LDA [104]. En algunos problemas de clasificación

en procesamiento de imágenes, un uso adecuado de una técnica de clasificación lineal como PCA+LDA puede aprovechar la alta dimensionalidad del espacio de partida, para encontrar una proyección en la que el problema es linealmente resoluble. Un ejemplo de este hecho se produce en la clasificación de género a partir de la apariencia [10].

En este caso concreto, W_{lda} se define como

$$\mathrm{W}_{lda} = \underset{\|\mathrm{w}\|=1}{argmax} \frac{|\mathrm{W}^T \mathrm{W}_{pca}^T S_B \mathrm{W}_{pca} \mathrm{W}|}{|\mathrm{W}^T \mathrm{W}_{pca}^T S_W \mathrm{W}_{pca} \mathrm{W}|} \tag{4.7}$$

En los últimos años, muchos métodos se han propuesto para solucionar el problema de la dimensión en el LDA. Algunos de ellos son el LDA basado en el espacio nulo [23], PCA+NULL [57], regularizado (RLDA) [63, 38], directo (DLDA) [127], basado en QR/GSVD (LDA/QR-GSVD) [92], dual-space LDA [112], ortogonal (OLDA) [60], etc. Varios de estos métodos se describen en [60, 61, 128]. Entre los métodos a destacar por su importancia para el presente trabajo está el LDA basado en el espacio nulo.

En el contexto de este documento, el espacio nulo de cualquier matriz de dispersión S se define como

$$\mathcal{N}(S) = span\{v_i \,|\, Sv_i = 0, \quad v_i \in \mathbb{R}^d, \quad i = 1, \ldots, n\} \tag{4.8}$$

donde $n = (d - r)$ es la dimensión del espacio nulo, y r es el rango de la matriz de dispersión. v_i son los vectores propios asociados a los valores propios nulos de S [23].

Chen et al. proponen maximizar la distancia entre clases en el espacio nulo de la matriz de dispersión intra-clase, $\mathcal{N}(S_W)$, para solucionar el problema de la dimensión en LDA. De forma breve, la solución propuesta por Chen et al. [23] proyecta un conjunto de entrenamiento reducido al espacio generado por los vectores propios asociados a los valores propios nulos de S_W. Una vez que las muestras se transforman, el PCA se emplea para maximizar la dispersión entre las clases, y obtener así los vectores de proyección.

Huang et al. [57] proponen un método para mejorar la eficiencia del LDA basado en el espacio nulo. Este método primero reduce la dimensión del conjunto de entrenamiento a partir del PCA, y después remueve el espacio nulo de la matriz de dispersión total de las muestras transformadas. Esta aproximación se conoce como PCA+NULL.

4.4. Vectores comunes discriminantes

El método de vectores comunes discriminantes (**DCV**, *Discriminative Common Vectors*) fue propuesto por Cevikalp et al. en [22] para problemas de reconocimiento de rostros, donde la dimensión de los datos de entrada es mayor a la diferencia entre el número de muestras de entrenamiento total y el número de clases. Este método es equivalente al método LDA basado en el espacio nulo [23], con la excepción que el método DCV explota el espacio original del conjunto de entrenamiento.

DCV busca una transformación lineal que maximiza la separación entre clases y minimiza las diferencias dentro de cada clase a partir del criterio de Fisher basado en el espacio nulo [20, 11, 13, 61] [§A.5],

$$\begin{aligned} \mathrm{W}_{dcv} &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_B \mathrm{W}| \\ &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_T \mathrm{W}| \end{aligned} \tag{4.9}$$

Este criterio cuantifica la discriminación entre diferentes clases de un conjunto de datos. Las propiedades comunes de cada clase se extraen eliminando las diferencias entre las muestras de una misma clase.

En síntesis, el método DCV busca proyectar las muestras de entrenamiento en un subespacio donde todas las muestras de una misma clase tengan una única proyección (un vector común por clase [44, 22]). Y además, que estas proyecciones

tengan la máxima dispersión entre ellas en un subespacio de menor dimensión, consiguiendo un vector común discriminante para cada clase. La figura 4.3 muestra de forma intuitiva el concepto del método DCV.

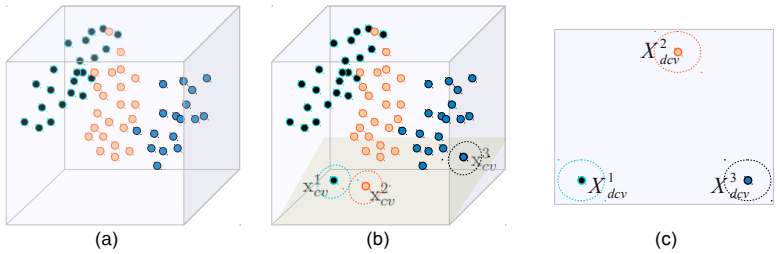


Figura 4.3: Método DCV. (a) Conjunto de entrenamiento. (b) El conjunto de entrenamiento se proyecta en un subespacio donde todas la muestras de una misma clase se colapsan, y se obtiene un único vector común para cada clase. Luego, (c) se busca un subespacio donde los vectores comunes tengan la máxima dispersión, obteniendo los vectores comunes discriminantes x^j_{dcv}.

La clasificación se realiza a partir de una medida de distancia, normalmente la euclídea, entre los vectores comunes discriminantes y el conjunto de test proyectado en el subespacio discriminante. La figura 4.4 muestra intuitivamente la clasificación.

El método DCV ha sido propuesto por los autores de dos formas distintas y equivalentes para obtener la matriz de proyección final. La primera de ellas se basa en la descomposición en valores y vectores propios de las matrices de dispersión, **DCV-EVD**. La segunda aproximación implica menos coste computacional, y se apoya en métodos basados en subespacios y en la ortogonalización de Gram-Schmidt, **DCV-GSO**. Las secciones 4.4.1 y 4.4.2 sintetizan la idea principal del método DCV a partir de eigendescomposición y a partir de ortogonalización, respectivamente.

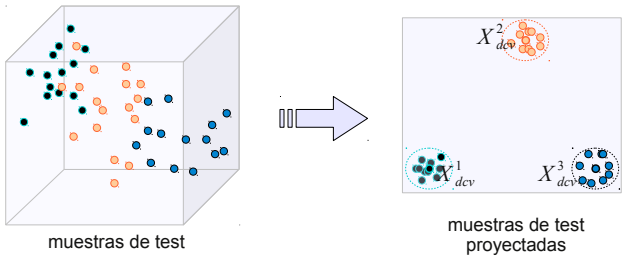


Figura 4.4: Clasificación de las muestras de test. Las muestras de test se proyectan en el subespacio donde están definidos los vectores comunes discriminantes, y luego una medida de distancia entre los datos de test proyectados y los vectores comunes discriminantes se emplea.

4.4.1. DCV a partir de eigendescomposición

El primer paso del método DCV es eliminar la variabilidad dentro de cada clase del conjunto de entrenamiento, obteniendo un único vector común por clase. Esto se puede conseguir proyectando el conjunto de entrenamiento $\mathbf{X} \in \mathbb{R}^{d \times M}$, en el espacio generado por los vectores propios asociados a los valores propios nulos de su matriz de dispersión intra-clase.

Sea S_W la matriz de dispersión intra-clase del conjunto de entrenamiento

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{m_j} (\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T \tag{4.10}$$

Sean $\mathrm{U} \in \mathbb{R}^{d \times r}$ y $\overline{\mathrm{U}} \in \mathbb{R}^{d \times n}$ matrices formadas por los vectores propios, u_i, asociados a los valores propios no nulos y nulos de S_W, donde r es el rango de S_W y $n = (d - r)$.

$$\mathrm{U} = [\mathrm{u}_1 \ \ldots \ \mathrm{u}_r], \qquad \overline{\mathrm{U}} = [\mathrm{u}_{(r+1)} \ \ldots \ \mathrm{u}_d]$$

Los vectores que conforman U y $\overline{\mathrm{U}}$ generan los subespacios rango y nulo de la matriz de dispersión intra-clase, $\mathcal{R}(S_W)$ y $\mathcal{N}(S_W)$, respectivamente.

Formalmente,

$$\mathcal{R}(S_W) = span\{\mathrm{u}_k | \, S_W \mathrm{u}_k \neq 0, \quad i = 1, \ldots, r\} \tag{4.11}$$
$$\mathcal{N}(S_W) = span\{\mathrm{u}_k | \, S_W \mathrm{u}_k = 0, \quad i = r+1, \ldots, d\} \tag{4.12}$$

donde $\mathcal{R}(S_W)$ y $\mathcal{N}(S_W)$ son subespacios complementarios ortogonales.

Las figuras 4.5(a-b) exponen intuitivamente el conjunto de entrenamiento y su proyección en los subespacios rango y nulo de la matriz de dispersión intra-clase.

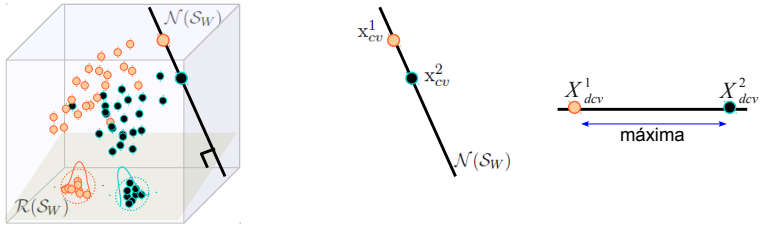


Figura 4.5: Método DCV. (a) Conjunto de entrenamiento proyectado en el espacio rango, $\mathcal{R}(S_W)$, y nulo, $\mathcal{N}(S_W)$. (b) Vectores comunes. (c) Vectores comunes discriminantes.

Si $\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T$ es el operador de proyección ortogonal al subespacio $\mathcal{N}(S_W)$, el vector común de la j-ésima clase del conjunto de entrenamiento se define como [22, 20]:

$$\mathrm{x}_{cv}^j = \overline{\mathrm{U}}\,\overline{\mathrm{U}}^T \mathrm{x}_j^i \tag{4.13}$$

donde x_{cv}^j es independiente del índice i. Para más detalles ver el teorema 2, del apéndice C.

Sin embargo, encontrar el conjunto de vectores propios, $\overline{\mathrm{U}}$, asociados a los valores propios nulos de S_W es una tarea computacionalmente costosa, ya que la dimensión de este espacio puede ser muy grande, y calcular proyecciones requeriría multiplicar matrices bastante grandes. En la práctica, es mucho más conveniente el uso de U que el de $\overline{\mathrm{U}}$ por que normalmente, en reconocimiento de imágenes, $r \ll n$.

Dado que $\mathcal{R}(S_W)$ y $\mathcal{N}(S_W)$ son subespacios complementarios, su suma directa es todo $\mathbb{R}^d = \mathcal{R}(S_W) \oplus \mathcal{N}(S_W)$. Por lo tanto, cada muestra $\mathrm{x}_j^i \in \mathbb{R}^d$ admite una única descomposición de la forma:

$$\mathrm{x}_j^i = \mathrm{U}\mathrm{U}^T\mathrm{x}_j^i + \overline{\mathrm{U}}\,\overline{\mathrm{U}}^T\mathrm{x}_j^i \tag{4.14}$$

donde $\mathrm{U}\mathrm{U}^T$ y $\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T$ son operadores de proyección ortogonales en $\mathcal{R}(S_W)$ y $\mathcal{N}(S_W)$, respectivamente.

De este modo, el vector común de la j-ésima clase del conjunto de entrenamiento se puede escribir como el residuo de x_j^i respecto a su proyección en $\mathrm{U}\mathrm{U}^T$.

$$\mathrm{x}_{cv}^j = \overline{\mathrm{U}}\ \overline{\mathrm{U}}^T\mathrm{x}_j^i = \mathrm{x}_j^i - \mathrm{U}\mathrm{U}^T\mathrm{x}_j^i \tag{4.15}$$

donde un único vector común se obtiene para todas las muestras de una misma clase ya que x_{cv}^j es independiente del índice i, y en consecuencia la matriz de dispersión intra-clase es cero en el subespacio definido por los x_{cv}^j.

El método DCV maximiza la dispersión de los vectores comunes, a partir del criterio de la ecuación (4.16) [§A.5] [22].

$$\begin{aligned} \mathrm{W}_{dcv} &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_B \mathrm{W}| = \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_T \mathrm{W}| \\ &= \underset{\|\mathrm{w}\|=1}{argmax} |\mathrm{W}^T S_{cv} \mathrm{W}| \\ &= [\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_r] \end{aligned} \tag{4.16}$$

S_{cv} es la matriz de dispersión de los vectores comunes,

$$S_{cv} = \sum_{j=1}^{c} (\mathrm{x}_{cv}^j - \mathrm{x}_{cv})(\mathrm{x}_{cv}^j - \mathrm{x}_{cv})^T \tag{4.17}$$

donde x_{cv} es el vector media de todos los x_{cv}^j.

De este modo, los vectores propios asociados a los valores propios no nulos de S_{cv} conforman la matriz de proyección final W_{dcv} que maximiza el criterio de la ecuación (4.16). Si

todos los vectores comunes son linealmente independientes el rango de S_{cv} es $(c-1)$, y por tanto el número de vectores propios que forman W_{dcv} es como máximo $r = (c-1)$. Es decir, $\mathrm{W} \in \mathbb{R}^{d\times(c-1)}$.

Así el vector comun discriminante, x^j_{dcv}, de la clase j se obtiene a partir de proyectar cualquier muestra, x^i_j, en el subespacio discriminante[†]. Como ya se ha dicho, x^j_{dcv} es independiente del índice i de la muestra (ver teorema 2) [22].

$$\mathrm{x}^j_{dcv} = \mathrm{W}^T_{dcv}\mathrm{x}^i_j \tag{4.18}$$

De la misma manera se puede proyectar cualquier muestra de test.

Normalmente cuando se trabaja con imágenes la dimensión de las muestras es grande. Ésto implica que el tamaño de las matrices de dispersión, involucradas en el cómputo de los vectores comunes discriminantes, es muy grande, y en consecuencia el coste computacional de realizar la eigendescomposición es elevado. Si la dimensión del espacio original es mayor al número de muestras de entrenamiento, $d > M$, los vectores propios vinculados a los valores propios no nulos de cualquier matriz de dispersión se pueden calcular a partir de una matriz de menor tamaño [§A.4] [86].

Después de varias mejoras [22, 20], el coste computacional asociado a este procedimiento para obtener los vectores comunes discriminantes, está dominado por $O(\ell(M^3) + dM^2)$ operaciones, vinculadas a la descomposición en valores y vectores propios de la matriz de dispersión intra-clase. ℓ es el número de iteraciones requeridas para que el algoritmo de eigendescomposición converja.

La figura 4.6 muestra cómo calcular los vectores comunes discriminantes de un conjunto de entrenamiento $\mathbf{X}$. En este

[†]A lo largo del documento x^j_{cv} es el vector comun definido en el espacio original de las muestras y x^j_{dcv} es el vector comun discriminante en el espacio reducido

proceso se distinguen dos fases. La primera calcula los vectores comunes. La segunda fase determina los vectores comunes discriminantes.

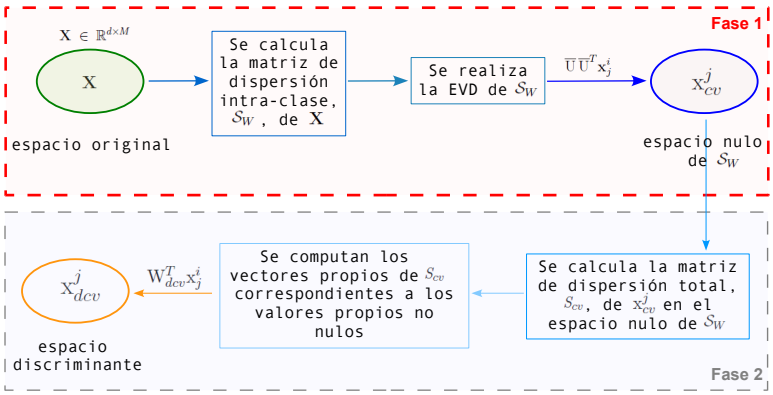


Figura 4.6: Proceso para obtener los DCV. $\mathbf{X}$ es el conjunto de entrenamiento. $\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T$ es el operador de proyección a $\mathcal{N}(S_W)$. x_{cv}^j son los vectores comunes. W_{dcv} es la matriz de proyección formada por los vectores propios asociados a los valores propios no nulos de la matriz de dispersión total, S_{cv}, de los vectores comunes. x_{dcv}^j es el vector común discriminante de la clase j.

4.4.2. DCV a partir de ortogonalización

Una forma alternativa de calcular los vectores comunes discriminantes, que implica menos coste computacional, es a partir de subespacios diferencia y la ortogonalización de Gram-Schmidt (GSO).

El espacio original de un conjunto de datos $\mathbf{X} \in \mathbb{R}^{d \times M}$ con c clases y m_j muestras en la clase j, también puede dividirse en dos subespacios complementarios, nombrados en [44] como subespacios diferencia e indiferencia. Sea $\mathcal{B}$ el subespacio diferencia de $\mathbf{X}$,

$$\mathcal{B} = span\{\mathrm{x}_1^2 - \mathrm{x}_1^1,\ \mathrm{x}_1^3 - \mathrm{x}_1^1,\ \ldots\ , \mathrm{x}_c^{n_c} - \mathrm{x}_c^1\}$$

con x_j^1 como vectores sustraendo y $j = 1, \ldots, c$.

Desde el teorema 6, el subespacio diferencia $\mathcal{B}$ y el subespacio rango de la matriz de dispersión intra-clase $\mathcal{R}(S_W)$, son el mismo subespacio. Luego, los operadores de proyección ortogonal son los mismos para el subespacio diferencia $\mathcal{B}$ y para el subespacio rango $\mathcal{R}(S_W)$ [44, 45].

Una base ortonormal, $\Theta \in \mathbb{R}^{d\times r}$, del subespacio diferencia $\mathcal{B}$ se puede conseguir a partir de la GSO. Básicamente, el método de ortonormalización de Gram-Schmidt consiste en lo siguiente: Dado un subespacio del que se conoce una base ortonormal y un vector que no está en el subespacio, se proyecta éste sobre el ortogonal de aquél a fin de conseguir una base ortonormal para un subespacio de dimensión una unidad mayor. Procediendo recursivamente desde un vector dado se puede conseguir una base ortonormal de todo el espacio. El algoritmo 1 muestra de forma general la ortogonalización de Gram-Schmidt para obtener una base ortonormal $Q = [q_1,\ q_2, \ldots, q_r] \in \mathbb{R}^{d\times r}$ del subespacio generado por los vectores $A = [a_1,\ a_2, \ldots, a_M] \in \mathbb{R}^{d\times M}$. En este algoritmo $\|a_1\| > 0$.

Algoritmo 1 *Ortogonalización de Gram-Schmidt.*

Entrada: $A \in \mathbb{R}^{d\times M}$.
Salida: $Q \in \mathbb{R}^{d\times r}$.

1. Normalizar el primer vector columna de A de modo que $q_1 = \frac{a_1}{\|a_1\|}$ y $l = 1$.
2. Desde $i \leftarrow 2$ hasta M
   a) $v = a_i - \sum_{k=1}^{l}(q_k^T\, a_i)\ q_k$
   b) si $\|v\| > 0$ entonces
      $l = l + 1$.
      $q_l = v/\|v\|$.
3. Una base ortonormal del subespacio generado por los vectores en A es $Q = [q_1,\ q_2, \ldots, q_r] \in \mathbb{R}^{d\times r}$, donde $r = l$.

Una vez se obtiene Θ, el operador de proyección ortogonal a $\mathcal{B}$ es $\Theta\Theta^T$. De esta manera, cualquier muestra de la clase j

se puede proyectar en $\mathcal{B}$ para obtener los vectores comunes, mediante la correspondiente sustitución

$$\mathrm{x}_{cv}^{j} = \mathrm{x}_{j}^{i} - \Theta\Theta^{T}\mathrm{x}_{j}^{i} \tag{4.19}$$

Los vectores comunes de la ecuación (4.19) son idénticos a los x_{cv}^{j} obtenidos a partir de eigendescomposición, ya que $\mathcal{B}$ y $\mathcal{R}(S_W)$ son el mismo subespacio. Por lo tanto, la matriz de proyección a $\mathcal{B}$ proyecta al subespacio $\mathcal{R}(S_W)$ (teorema 6), aunque las bases U y Θ son en general diferentes. El teorema 5 del apéndice C muestra que, los x_{cv}^{j} no dependen de la selección del vector sustraendo.

Para obtener la matriz de proyección final W_{dcv}, sólo se necesita una base ortonormal del subespacio rango de S_{cv}. Así que, W_{dcv} se calcula de forma eficiente a partir de una base ortonormal del subespacio diferencia, $\mathcal{B}_{cv}$, de los vectores comunes x_{cv}^{j},

$$\mathcal{B}_{cv} = span\{(\mathrm{x}_{cv}^{2} - \mathrm{x}_{cv}^{1}), \ldots, (\mathrm{x}_{cv}^{c} - \mathrm{x}_{cv}^{1})\} \tag{4.20}$$

ya que $\mathcal{B}_{cv}$ y el subespacio rango de S_{cv}, son el mismo subespacio [22]. Si los vectores comunes son linealmente independientes $\mathrm{W}_{dcv} \in \mathbb{R}^{d\times(c-1)}$.

El coste computacional de obtener los vectores comunes discriminantes mediante subespacios diferencia y la ortogonalización de Gram-Schmidt está dominado por $O(dM^2)$ operaciones [22].

La figura 4.7 presenta como obtener los vectores comunes discriminantes a partir de subespacios diferencia y ortogonalización de Gram-Schmidt.

4.5. Vectores comunes discriminantes extendidos

El método de los vectores comunes extendidos (traducción libre de *Rough Common Vector*, **RCV**) fue propuesto por Ta-

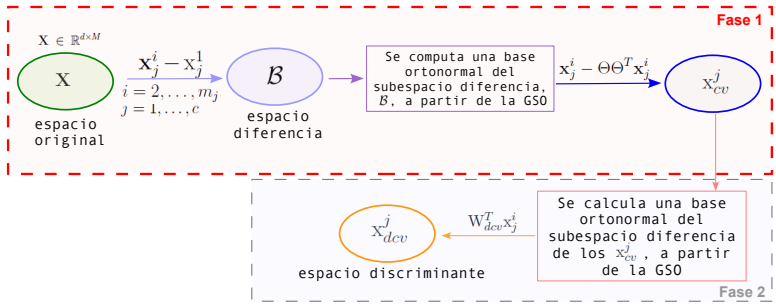


Figura 4.7: Método DCV a partir de subespacios diferencia y GSO. $\mathbf{X}$ es el conjunto de entrenamiento. $\mathcal{B}$ es el subespacio diferencia de $\mathbf{X}$. $\Theta\Theta^T$ es el operador de proyección ortogonal a $\mathcal{B}$. x_{cv}^j son los vectores comunes. W_{dcv} es una base ortonormal del subespacio diferencia de los x_{cv}^j, que contiene los vectores de proyección al subespacio discriminante. x_{dcv}^j es el vector común discriminante de la clase j.

mura et al. en [105], para tareas de reconocimiento de rostros. Ellos proponen una forma más flexible y robusta de calcular los vectores comunes para la clasificación.

La idea fundamental del método RCV se basa en extender el subespacio nulo de la matriz de dispersión intra-clase, S_W, a partir de considerar los vectores propios de S_W asociados a valores propios pequeños como si fueran nulos. De este modo el espacio de características se divide en dos subespacios. Un subespacio con los vectores propios asociados a los valores propios de mayor valor de la matriz de dispersión intra-clase, S_W, y un subespacio con los vectores propios asociados a los valores propios de menor valor (incluidos los nulos).

RCV reinterpreta el espacio nulo de la matriz de dispersión intra-clase para obtener las características discriminantes. Como consecuencia, el método RCV reduce el subespacio rango $\mathcal{R}(S_W)$, mientras amplia el subespacio nulo original $\mathcal{N}(S_W)$ a partir de los vectores asociados a los valores propios normalizados no nulos, organizados de mayor a menor, que no

satisfacen la condición:

$$(1-\alpha) \geq \sum_{i=1}^{k} \lambda_i, \qquad k \leq r \qquad \sum_{i=1}^{r} \lambda_i = 1. \tag{4.21}$$

donde α es una constante positiva, $0 \leq \alpha < 1$. El valor de la constante indica la cantidad de varianza que se añade al nuevo espacio nulo de S_W. En el caso particular $\alpha = 0$ se obtiene el método DCV original.

Sean U y $\overline{\mathrm{U}}$ el conjunto de vectores propios asociados a los valores propios no nulos y nulos de S_W, respectivamente. El nuevo conjunto de vectores asociados a los respectivos subespacios complementarios a partir de α es:

$$\mathrm{U}_\alpha = [\mathrm{u}_1 \ldots \mathrm{u}_k] \tag{4.22}$$
$$\overline{\mathrm{U}}_\alpha = [\mathrm{u}_{(k+1)} \ldots \mathrm{u}_r \, \mathrm{u}_{(r+1)} \ldots \mathrm{u}_d] \tag{4.23}$$

donde los vectores que conforman U_α y $\overline{\mathrm{U}}_\alpha$ generan el nuevo subespacio rango reducido $\tilde{\mathcal{R}}(S_W)$ y el nuevo subespacio nulo extendido $\tilde{\mathcal{N}}(S_W)$.

$$\tilde{\mathcal{R}}(S_W) = span\{\mathrm{u}_1, \ldots, \mathrm{u}_k\} \tag{4.24}$$
$$\tilde{\mathcal{N}}(S_W) = span\{\mathrm{u}_{(k+1)}, \ldots \mathrm{u}_r, \mathrm{u}_{(r+1)}, \ldots, \mathrm{u}_d\} \tag{4.25}$$

$\mathrm{U}_\alpha \mathrm{U}_\alpha^T$ y $\overline{\mathrm{U}}_\alpha \overline{\mathrm{U}}_\alpha^T$ son los operadores de proyección al espacio rango reducido y al espacio nulo extendido de S_W, respectivamente. De este modo, el vector común extendido de la clase j está dado por:

$$\mathrm{x}_{rcv}^j = \overline{\mathrm{U}}_\alpha \overline{\mathrm{U}}_\alpha^T \mathrm{x}_j^i = \mathrm{x}_j^i - \mathrm{U}_\alpha \mathrm{U}_\alpha^T \mathrm{x}_j^i \tag{4.26}$$

Una diferencia (muy) importante respecto al método DCV es que el cálculo del vector común extendido no es independiente de i. Una elección razonable que se utiliza en el artículo original es remplazar el vector x_j^i por el vector media de la clase j en $\mathbf{X}$ [105].

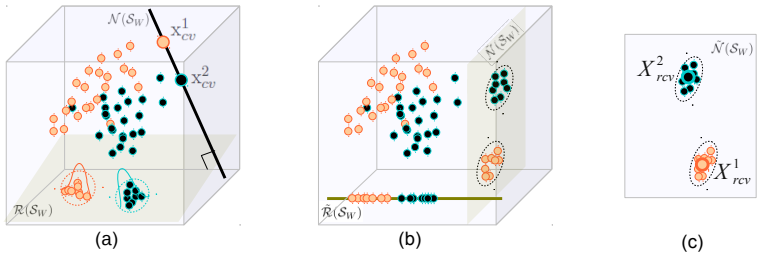


Figura 4.8: Método RCV. (a) Subespacios rango y nulo de S_W. (b) Subespacio rango reducido, $\tilde{\mathcal{R}}(S_W)$, y subespacio nulo extendido, $\tilde{\mathcal{N}}(S_W)$, de S_W. (c) Vectores comunes extendidos, x^j_{rcv}, calculados a partir de la media de cada clase.

La figura 4.8 muestra gráficamente la idea de los vectores comunes extendidos para un conjunto de entrenamiento proyectado en el espacio nulo extendido, a partir de $\overline{\mathrm{U}}_\alpha \overline{\mathrm{U}}_\alpha^T$.

Para obtener los vectores comunes discriminantes extendidos, x^j_{rdcv}, se calcula la matriz de proyección final, W_{rdcv}, a partir de maximizar la dispersión de los x^j_{rcv}. Por lo tanto, se utiliza el análisis de componentes principales [§4.2].

La matriz de dispersión de los vectores comunes extendidos se define como

$$S_{rcv} = \sum_{j=1}^{c} (\mathrm{x}^j_{rcv} - \mathrm{x}_{rcv})(\mathrm{x}^j_{rcv} - \mathrm{x}_{rcv})^T \tag{4.27}$$

donde x_{rcv} es el vector media de los x^j_{rcv}.

Finalmente, los vectores propios asociados a los valores propios no nulos de S_{rcv} son los vectores que forman la matriz de proyección final W_{rdcv}. De esta manera, los vectores comunes discriminantes extendidos se definen como

$$\mathrm{x}^j_{rdcv} = \mathrm{W}^T_{rdcv} \mathrm{x}_j \tag{4.28}$$

donde x_j es el vector media de la clase j.

El número de operaciones involucradas para obtener los x^j_{rdcv} es $O(\ell(M^3) + dM^2 + dMr_\alpha)$ del cálculo de S_W y de

su EVD. ℓ es el número de iteraciones requeridas para que el algoritmo de eigendescomposición converja y r_α es el rango de $\tilde{\mathcal{R}}(S_W)$. La segunda parte del método RDCV, a partir de subespacios diferencia y ortogonalización, conlleva $O(dc^2)$ operaciones.

4.6. Aplicabilidad

Los métodos PCA, LDA, DCV y RDCV, vistos en este capítulo, se basan en proyecciones ortonormales. De forma especifica, el PCA encuentra un subespacio donde las muestras de entrenamiento preservan la máxima variabilidad. En procesamiento de imágenes el método PCA se utiliza comúnmente como una técnica de reducción de la dimensionalidad.

El LDA es un método de clasificación supervisado que busca minimizar la distancia entre muestras de una misma clase, mientras maximiza la distancia entre muestras de diferentes clases. Como se ha dicho antes, debido al problema del tamaño de muestras pequeño, distintos algoritmos del método LDA han sido propuestos.

En cuanto al método DCV, éste se utiliza en tareas donde la dimensionalidad del espacio original de las muestras es mayor a la diferencia entre el conjunto de entrenamiento y el número de clases. DCV busca un subespacio donde todas las muestras de una misma clase tienen una única proyección, mientras la distancia entre estas proyecciones es máxima.

Las versiones de DCV, desde EVD y GSO, presentan la misma tasa de acierto ya que los dos algoritmos son semejantes en cuanto a ésta. La diferencia entre estos algoritmos es el coste computacional. Mientras la tendencia del coste computacional de DCV-GSO es cuadrática, en DCV-EVD la tendencia es cubica. Sin embargo, múltiples software matemáticos (como Matlab y OpenCV, entre otros) han desarrollado funciones eficientes para calcular la EVD.

En aquellos casos donde el tamaño del conjunto de entrenamiento es mayor o igual la dimensión del espacio original, el método DCV no proporciona un subespacio discriminante adecuado para la clasificación de las muestras. En estos casos el método RDCV se puede utilizar. RDCV relaja las bases del método DCV-EVD, para obtener un subespacio de proyección lo suficientemente discriminante.

Capítulo 5

Métodos de clasificación basados en subespacios no lineales

Resumen — Los métodos basados en subespacios no lineales son una generalización de los métodos basados en subespacios lineales. Este tipo de métodos se utilizan cuando las muestras del conjunto de entrenamiento tienen una distribución compleja y los métodos lineales no logran una adecuada separación para la clasificación. Este capítulo presenta el concepto fundamental y el estado del arte de los métodos basados en subespacios no lineales. Además describe algunos métodos basados en subespacios no lineales que se utilizan más adelante, como son el análisis de componentes principales con kernel y el método de los vectores comunes discriminantes con kernel.

Contenido

5.1. Conceptos fundamentales 36
5.2. Estado del arte 37
5.3. Análisis de componentes principales con kernel 39
5.4. Vectores comunes discriminantes con kernel 41

5.1. Conceptos fundamentales

La idea fundamental de los métodos basados en subespacios no lineales es proyectar las muestras de entrada en un espacio de características de mayor dimensión, $\mathcal{F}$, donde se espera que las muestras se separen de forma lineal o el problema presente una complejidad mucho menor. La proyección se realiza a partir de un mapping no lineal $\Phi = [\phi(\mathrm{x}_1)\ \phi(\mathrm{x}_2)\ \phi(\mathrm{x}_3)\ \ldots\ \phi(\mathrm{x}_i)]$ que proyecta las muestras desde $\mathbb{R}^d$ a $\mathcal{F}$.

$$\Phi : \mathbb{R}^d \longrightarrow \mathcal{F}$$

La figura 5.1 ilustra el concepto de proyectar un conjunto de entrenamiento, con distribución compleja, a un espacio de mayor dimensión donde las muestras se separan de forma lineal.

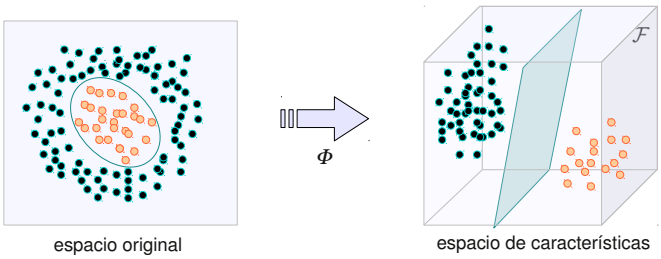


Figura 5.1: Los métodos basados en subespacios no lineales proyectan el conjunto de entrenamiento a un espacio de características de mayor dimensión, $\mathcal{F}$, utilizando un mapping no lineal Φ. En $\mathcal{F}$ se espera que las muestras se separen de forma lineal.

La proyección del conjunto de entrenamiento en $\mathcal{F}$ se realiza implícitamente a partir del producto punto entre cada par de muestras. Sea $\mathbf{X} \in \mathbb{R}^{d \times M}$ el conjunto de entrenamiento y $\mathcal{F}$ el espacio de características. La función que retorna el producto punto entre las muestras en el espacio $\mathcal{F}$ se conoce como función kernel.

Un kernel es una función k definida en $\mathbb{R}^d \times \mathbb{R}^d$, que para todo x_i, $\mathrm{x}_j \in \mathbf{X}$

$$k(\mathrm{x}_i, \mathrm{x}_j) = \langle \phi(\mathrm{x}_i), \phi(\mathrm{x}_j) \rangle$$

donde $\langle \cdot, \cdot \rangle$ denota el producto punto y $\phi(\ )$ es la función de mapeo que proyecta las muestras desde $\mathbb{R}^d$ a $\mathcal{F}$. La matriz que contiene los productos punto entre todos los pares de muestras determina en el espacio de características la posición relativa entre las muestras [12]. Esta matriz es siempre simétrica y semidefinida positiva,

$$\mathrm{K}_{ij} = k(\mathrm{x}_i, \mathrm{x}_j)$$

y se llama matriz kernel.

La solución de los métodos no lineales son funciones lineales en el espacio de características, de modo que

$$f(\mathrm{x}) = \mathrm{W}^T \phi(\mathrm{x}),$$

Si nos restringimos al caso en que W se pueden expresar como una combinación lineal de las muestras de entrenamiento, $\sum_{i=1}^{M} \alpha_i \phi(\mathrm{x}_i)$, se obtiene que

$$f(\mathrm{x}) = \sum_{i=1}^{M} \alpha_i^T k(\mathrm{x}_i, \mathrm{x})$$

Por lo tanto, la función de mapeo no lineal y las muestras proyectadas en el espacio de características no se usan explícitamente, lo cual hace los métodos no lineales computacionalmente factibles.

5.2. Estado del arte

En 1963, Vapnik y Lerner desarrollaron las máquinas de vectores soporte (SVM, *Support Vector Machines*) [111]. El

algoritmo original fue un hiper plano óptimo, considerado como un método supervisado que pertenecen a la familia de los clasificadores lineales [110]. Sin embargo en 1992 Bernhard Boser, Isabelle Guyon y Vapnik [16] utilizaron esta técnica para formular clasificadores no lineales aplicando el *kernel trick* (originalmente propuesto por Aizerman et al. [1]) para maximizar los márgenes de los hiperplanos. El algoritmo resultante es formalmente similar, excepto que cada producto punto se reemplaza por una función kernel no lineal.

Con las SVM surgieron generalizaciones/extensiones de los métodos lineales. Uno de los primeros métodos extendidos fue el PCA en el año 1996 por Schölkopf et al. [97], llamado Análisis de componentes principales con kernel (KPCA, *Kernel Principal Component Analysis*). KPCA utiliza una función de mapeo no lineal para proyectar los datos de entrada a un espacio de características de mayor dimensión donde se aplica PCA [§4.2].

En 1999 Mika et al. [95] propusieron la versión kernel del FLD, llamada Análisis discriminante de Fisher con Kernel (KFDA, *Kernel Fisher Discriminant Analysis*). Muchos algoritmos del método KFDA se propusieron posteriormente [93, 85, 84, 122, 120, 132, 75]

El análisis de correlación canónica con kernel (KCCA, *Kernel Canonical Correlation Analysis*) tuvo su origen en el año 2001 en [2]. KCCA es un método no lineal que determina las dependencias estadísticas entre dos conjuntos de variables aleatorias [52].

Francis R. Bach y Michael I. Jordan presentaron en el año 2002 la versión kernel de ICA, denominada como Análisis de componentes independientes con kernel (KICA, *Kernel Independent Component Analysis*) [6].

Cevikalp et al. proponen en el año 2006 el método de los Vectores comunes discriminantes con kernel (KDCV, *Discriminative Common Vector method with Kernels*) [20], que es la versión no lineal de DCV.

Posteriormente, en el año 2007 Cevikalp et al. publicaron en [19] la versión kernel de CV, llamada Vectores comunes con kernel (KCV, *The Kernel Common Vector method*).

Los métodos anteriores se formulan en términos de los productos punto de las muestras proyectadas en el espacio de características, donde la función kernel se emplea para computar estos productos punto. El apéndice A.6 muestra las funciones kernel más utilizadas en reconocimiento de patrones.

5.3. Análisis de componentes principales con kernel

El método de análisis de componentes principales con kernel (*Kernel Principal Component Analysis*, **KPCA**) fue propuesto por Schölkopf et al. en [97]. Este método utiliza una función de mapeo no lineal para proyectar los datos de entrada a un espacio de características de mayor dimensión donde se aplica PCA.

Como se ha dicho antes, el PCA encuentra las direcciones de proyección que maximizan la dispersión total del subespacio [§4.2]. Lo cual equivale a encontrar los valores propios mayores que cero y los vectores propios asociados de la matriz de dispersión total.

El método KPCA proyecta cada vector x presente en el espacio original de entrada $\mathbb{R}^d$, a un espacio de características de mayor dimensión $\mathcal{F}$, a partir de una función de mapeo no lineal Φ.

$$\Phi : \mathbb{R}^d \longrightarrow \mathcal{F} \tag{5.1}$$

Asumiendo por el momento que los datos en el espacio de características están centrados, la matriz de dispersión total se define como

$$S_T^{\Phi} = \frac{1}{M} \sum_{j=1}^{M} \phi(\mathrm{x}_j)\phi(\mathrm{x}_j)^T$$

Por lo tanto el problema de valores y vectores propios en el espacio $\mathcal{F}$ está dado por

$$\lambda \mathrm{w} = S_T^{\Phi} \mathrm{w} = \frac{1}{M} \sum_{j=1}^{M} \langle \phi(\mathrm{x}_j), \mathrm{w} \rangle \ \phi(\mathrm{x}_j)$$

como $\langle \ , \ \rangle$ denota el producto punto, todas las soluciones de w con $\lambda \neq 0$ están en el espacio generado por $\phi(\mathrm{x}_1)\ \phi(\mathrm{x}_2)\ \ldots\ \phi(\mathrm{x}_M)$. Consecuentemente, $\lambda \mathrm{w} = S_T^{\Phi} \mathrm{w}$ es equivalente a

$$\lambda \langle \phi(\mathrm{x}_k), \mathrm{w} \rangle = \langle \phi(\mathrm{x}_k), S_T^{\phi} \mathrm{w} \rangle \qquad \forall \quad k = 1, \ldots, M \tag{5.2}$$

donde existen coeficientes α_i que

$$\mathrm{w} = \sum_{i=1}^{M} \alpha_i \phi(\mathrm{x}_i) \tag{5.3}$$

Combinando las ecuaciones (5.2) y (5.3) se obtiene

$$\lambda M \sum_{i=1}^{M} \alpha_i \langle \phi(\mathrm{x}_k), \phi(\mathrm{x}_i) \rangle = \sum_{i=1}^{M} \alpha_i \left\langle \phi(\mathrm{x}_k), \sum_{j=1}^{M} \phi(\mathrm{x}_j) \langle \phi(\mathrm{x}_j), \phi(\mathrm{x}_i) \rangle \right\rangle \tag{5.4}$$

para todo $k = 1, \ldots, M$. De este modo el problema de valores y vectores propios de la ecuación (5.4) solo involucra los productos punto de las muestras de entrenamiento en el espacio de características, y Φ no se computa explícitamente. En su lugar, se calcula la matriz de kernel $\mathrm{K} \in \mathbb{R}^{M \times M}$.

$$\mathrm{K}_{ij} = \langle \phi(\mathrm{x}_i), \phi(\mathrm{x}_j) \rangle \tag{5.5}$$

Luego, la ecuación (5.4) se puede expresar como

$$\lambda M \sum_{i=1}^{M} \alpha_i \mathrm{K}_{ki} = \sum_{i=1}^{M} \alpha_i \sum_{j=1}^{M} \mathrm{K}_{kj} \mathrm{K}_{ji}$$

De esta forma, el problema del KPCA se define por

$$\lambda M \mathrm{K} \mathrm{w} = \mathrm{K}^2 \mathrm{w} \tag{5.6}$$

donde w denota el vector columna con entradas $\alpha_1, \ldots, \alpha_M$. Para encontrar la solución de (5.6), se resuelve el siguiente problema de valores y vectores propios para valores propios no nulos.

$$M \lambda \mathrm{w} = \mathrm{K} \mathrm{w} \tag{5.7}$$

De este modo, el PCA en $\mathcal{F}$ es equivalente a resolver el problema de valores y vectores propios de la ecuación (5.7). Antes de aplicar el PCA en el espacio de características las muestras de entrenamiento son centradas [96].

La matriz de kernel centrada, K_c, se computa como el producto punto de la matriz $(\mathbf{X} - 1_M\mathbf{X})$ consigo misma

$$\begin{aligned} \mathrm{K}_c &= (\mathbf{X} - 1_M\mathbf{X})(\mathbf{X} - 1_M\mathbf{X})^T \\ &= K - 1_M K - K1_M + 1_M K 1_M \end{aligned} \tag{5.8}$$

con $1_M = (1/M)_{(M\times M)}$ [12].

Finalmente, la dimensión del problema se reduce obteniendo los r primeros vectores propios asociados a los r primeros valores propios no nulos de K_c. Las componentes principales no lineales del conjunto de entrenamiento se definen como

$$\mathrm{X}_{kpca} = \mathrm{W}_{kpca}^T \mathrm{K}_c, \tag{5.9}$$

donde los vectores columna que conforman W_{kpca} son los vectores propios asociados a los valores propios $\lambda > 0$ de K_c.

Las componentes principales no lineales de test, $\mathbf{X}^{test} \in \mathbb{R}^{d\times p}$, se definen como

$$\mathrm{X}_{kpca}^{test} = \mathrm{W}_{kpca}^T \mathrm{K}_c^{test}$$

donde,

$$\begin{aligned} \mathrm{K}_c^{test} &= \mathrm{K}^{test} - 1'_M\mathrm{K} - \mathrm{K}^{test}1_M + 1'_M\mathrm{K}1_M \end{aligned} \tag{5.10}$$

con $\mathrm{K}_{ij}^{test} = \langle \Phi(\mathrm{x}_i), \Phi(\mathrm{x}_j^{test})\rangle$ y $1'_M$ como una matriz de tamaño $(p \times M)$ con todo sus elementos iguales a $(1/M)$.

5.4. Vectores comunes discriminantes con kernel

En algunos casos, los vectores comunes discriminantes (DCV [§4.4]) no son separables en el espacio original. En tales casos, el conjunto de entrenamiento puede ser proyectado a un espacio de mayor dimensión donde los nuevos vectores comunes discriminantes son distintos unos de otros.

El método **KDCV** (*Discriminative Common Vector with Kernels* [20]) utiliza una función de mapeo no lineal para proyectar las muestras del espacio de entrada original, $\mathbb{R}^d$, a un espacio de características, $\mathcal{F}$, de mayor dimensión donde se aplica el método DCV [§4.4], y se espera que las muestras sean separables linealmente.

KDCV maximiza el criterio de Fisher basado en el espacio nulo [§A.5], definido como:

$$\begin{aligned} \mathrm{W}_{kdcv} &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_B \mathrm{W}| \\ &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_T \mathrm{W}| \end{aligned}$$

Para un conjunto de entrenamiento $\mathbf{X}$ con M muestras, c clases y m_j muestras en la clase j. La transformación al subespacio $\mathcal{F}$ se define como $\Phi = [\phi(\mathrm{x}_1^1)\ \phi(\mathrm{x}_1^2)\ \ldots \phi(\mathrm{x}_1^{m_1})\ \phi(\mathrm{x}_2^1)\ \ldots \phi(\mathrm{x}_c^{m_c})]$.

Las matrices de dispersión intra-clase S_W^Φ, inter-clase S_B^Φ y total S_T^Φ en el espacio transformado, $\mathcal{F}$, son:

$$\begin{aligned} S_W^\Phi &= \sum_{j=1}^{c} \sum_{i=1}^{m_j} (\phi(\mathrm{x}_j^i) - \mathrm{x}_j^\Phi)(\phi(\mathrm{x}_j^i) - \mathrm{x}_j^\Phi)^T \\ &= (\Phi - \Phi\mathrm{G})(\Phi - \Phi\mathrm{G})^T \qquad (5.11) \\ S_B^\Phi &= \sum_{j=1}^{c} m_j (\mathrm{x}_j^\Phi - \mathrm{x}_t^\Phi)(\mathrm{x}_j^\Phi - \mathrm{x}_t^\Phi)^T \\ &= (\Phi\mathrm{H} - \Phi\mathrm{L})(\Phi\mathrm{H} - \Phi\mathrm{L})^T \qquad (5.12) \\ S_T^\Phi &= \sum_{j=1}^{c} \sum_{i=1}^{m_j} (\phi(\mathrm{x}_j^i) - \mathrm{x}_t^\Phi)(\phi(\mathrm{x}_j^i) - \mathrm{x}_t^\Phi)^T \\ &= (\Phi - \Phi 1_M)(\Phi - \Phi 1_M)^T = S_W^\Phi + S_B^\Phi \qquad (5.13) \end{aligned}$$

donde, x_j^i es la $i-$ésima muestra de la clase j. En el espacio de características $\mathcal{F}$, el vector media de la clase j y la media de todas las muestras se representan por x_j^Φ y x_t^Φ, respectivamente.

$\mathrm{G} = diag[\mathrm{G}_1, \ \ldots\ , \mathrm{G}_c] \in \mathbb{R}^{M\times M}$ y $\mathrm{H} = diag[\mu_1, \ldots, \mu_c] \in \mathbb{R}^{M\times c}$ son matrices con submatrices en su diagonal. Cada $G_j \in \mathbb{R}^{m_j \times m_j}$ es una matriz con todos sus elementos iguales a $(1/m_j)$, y cada $\mu_j \in \mathbb{R}^{m_j}$ es un vector con todos sus elementos iguales a $(1/\sqrt{m_j})$.

L es un matriz formada por c vectores columna de tamaño $(M \times 1)$, donde cada vector tiene todos sus elementos iguales a $\sqrt{m_j}/M$. 1_M es una matriz de tamaño $(M \times M)$ con todos sus elementos iguales a $(1/M)$.

Como se ha dicho antes, el método lineal DCV [§4.4], proyecta el conjunto de entrenamiento en el espacio nulo de la matriz de dispersión intra-clase $\mathcal{N}(S_W)$ y luego aplica el PCA [§4.2]. Otra opción es, proyectar primero el conjunto de entrenamiento en el espacio rango de la matriz de dispersión total $\mathcal{R}(S_T)$ a partir del PCA, y luego en el espacio transformado encontrar el subespacio nulo [20], como lo muestra el teorema 7.

Así como en el caso lineal, el método KDCV puede primero proyectar el conjunto de entrenamiento en $\mathcal{N}(S_W^\Phi)$ y luego aplicar PCA. O puede, primero aplicar PCA para proyectar el conjunto de entrenamiento en $\mathcal{R}(S_T^\Phi)$, y luego encontrar una base ortonormal para el nuevo espacio nulo de la matriz de dispersión intra-clase en el espacio transformado.

Sin embargo, el primer enfoque no es posible ya que necesita explícitamente la función Φ. Por consiguiente, el segundo enfoque es más apropiado para calcular los vectores comunes discriminantes no lineales, Por lo tanto, el conjunto de entrenamiento se proyecta en $\mathcal{R}(S_T^\Phi)$ aplicando el kernel PCA [97] [§5.3].

Con las muestras transformadas se calcula el espacio nulo de la nueva matriz de dispersión intra-clase, obteniendo los vectores comunes discriminantes no lineales que representan cada clase. Las propiedades matemáticas del método DCV [§4.4] se extienden a KDCV, difiriendo en el mapeo de las muestras.

De forma más específica, una vez se calcula la matriz de kernel $\mathrm{K} \in \mathbb{R}^{M\times M}$, ésta se centra

$$\mathrm{K}_c = \mathrm{K} - 1_M\mathrm{K} - \mathrm{K}1_M + 1_M\mathrm{K}1_M,$$

para después obtener los vectores propios, U, asociados a los valores propios no nulos, Λ, de K_c. Formalmente, $\mathrm{K}_c = \mathrm{U}\Lambda\mathrm{U}^T$.

La matriz que proyecta el conjunto de entrenamiento en $\mathcal{R}(S_T^\Phi)$ es $(\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2}$. Por tanto, en el espacio reducido las nuevas matrices de dispersión, total e intra-clase, están definidas por

$$\begin{aligned}
\tilde{S_T}^{\Phi} &= ((\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2})^T S_T^\Phi (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2} \\
&= \Lambda^{-1/2}\mathrm{U}^T\mathrm{U}\Lambda\mathrm{U}^T\mathrm{U}\Lambda\mathrm{U}^T\mathrm{U}\Lambda^{-1/2} = \Lambda \qquad (5.14)\\
\tilde{S_W}^{\Phi} &= ((\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2})^T S_W^\Phi (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2} \\
&= \Lambda^{-1/2}\mathrm{U}^T\tilde{\mathrm{K}}_w\tilde{\mathrm{K}}_w^T\mathrm{U}\Lambda^{-1/2} \qquad (5.15)
\end{aligned}$$

donde $\tilde{\mathrm{K}}_w = \mathrm{K} - \mathrm{KG} - 1_M\mathrm{K} + 1_M\mathrm{KG} = (\mathrm{K} - 1_M\mathrm{K})(\mathrm{I} - \mathrm{G})$.

Los vectores propios, V, asociados a los valores propios nulos de $\tilde{S_W}^{\Phi}$, satisfacen

$$\mathrm{V}^T\tilde{S_W}^{\Phi}\mathrm{V} = 0,$$

la matriz de proyección final se define como

$$\mathrm{W}_{kdcv} = (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2}\mathrm{V} \qquad (5.16)$$

donde W_{kdcv} tiene como máximo $(c-1)$ vectores de proyección.

Si el espacio nulo de $(\mathrm{V}^T\tilde{S_B}^{\Phi}\mathrm{V})$ es no vacío, éste se remueve y las direcciones de proyección se rotan de tal manera que las nuevas matrices

de dispersión total e inter-clase son diagonales.

$$\mathrm{V}^T \tilde{S_B}^{\Phi} \mathrm{V} = \mathrm{V}^T \tilde{S_T}^{\Phi} \mathrm{V} = \mathrm{V}^T \Lambda \mathrm{V} = \mathrm{Y} \tilde{\Lambda} \mathrm{Y}^T \tag{5.17}$$

En este caso, la matriz de proyección final es

$$\mathrm{W}_{kdcv} = (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2}\mathrm{VY} \tag{5.18}$$

Los vectores comunes discriminantes no lineales del conjunto de entrenamiento son por tanto:

$$\mathrm{x}_{kdcv} = \mathrm{W}_{kdcv}^T(\Phi - \Phi 1_M) = (\mathrm{U}\Lambda^{-1/2}\mathrm{VY})^T \mathrm{K}_c \tag{5.19}$$

La clasificación de las muestras de test, $\mathbf{X}^{test} \in \mathbb{R}^{d \times p}$, se realiza a partir de calcular la distancia mínima entre los x_{kdcv} y las muestras de test proyectadas x_{kdcv}^{test}.

$$\mathrm{x}_{kdcv}^{test} = (\mathrm{U}\Lambda^{-1/2}\mathrm{VY})^T \left(\mathrm{K}^{test} - \mathrm{K}1'_M - 1_M\mathrm{K}^{test} + 1_M\mathrm{K}1'_M\right) \tag{5.20}$$

donde K^{test} es la matriz kernel del conjunto de test y $1'_M$ es una matriz de tamaño $(M \times p)$ con todos sus elementos iguales a $(1/M)$.

Contribuciones

En algunos casos, los métodos DCV y KDCV no proporcionan una buena discriminación, ya sea por la limitación de que el número de muestras de entrenamiento es mayor a la dimensión del espacio original o porque el conjunto de entrenamiento presenta una distribución compleja. En algunos de estos casos, el subespacio lineal generado por el método RCV tampoco es suficiente para una adecuada clasificación.

Por esto, aquí se propone una versión no lineal del método de vectores comunes discriminantes extendidos que también puede verse como una generalización de los métodos de vectores comunes discriminantes con kernel propuestos hasta la fecha.

Independientemente de los resultados en clasificación y del poder de generalización, los métodos anteriores se basan en un aprendizaje por lotes o de tipo *batch*. El aprendizaje por lotes consiste en procesar una o repetidas veces el conjunto de entrenamiento como un todo hasta obtener un modelo final que representa a dicho conjunto.

A pesar de su enorme campo de aplicación, las técnicas de aprendizaje por lotes, no pueden ser aplicadas en un gran número de problemas donde los datos proceden de entornos dinámicos y/o van siendo adquiridos a lo largo del tiempo. Tampoco es conveniente su aplicación cuando el consumo de recursos es exagerado, ya sea de memoria o de tiempo de cómputo, al procesar todas las muestras a la vez.

Por ejemplo, un sistema de seguridad basado en la clasificación de rostros, se entrena con imágenes de rostros del personal. Si el aprendizaje del sistema es por lotes implica que, cuando hay nuevo personal en la empresa el sistema tiene que ser entrenado de nuevo desde cero (con las imágenes del personal actual e imágenes de los nuevos empleados). Esto conlleva que la memoria requerida y el coste computacional sean cada vez más altos en el entrenamiento.

Una alternativa a este problema es realizar un aprendizaje incremental. El aprendizaje incremental se basa en correcciones progresivas

aplicadas sobre un modelo previamente aprendido. Estas correcciones se calculan a partir de pocos datos y, por tanto, con costes temporales y espaciales reducidos.

Los métodos de aprendizaje incremental presentan características que los hacen muy apropiados para trabajar con grandes cantidades de datos, como son, su capacidad de incorporar nuevas experiencias a la base de conocimiento, y su capacidad de evolucionar esta base de conocimiento desde una estructura sencilla hacia otra más compleja.

Por esto, se proponen en este trabajo algunas versiones incrementales del método de los vectores comunes discriminantes y del método de los vectores comunes discriminantes extendidos. Estas extensiones, permiten incorporar al conjunto de entrenamiento nuevas muestras a las clases existentes, así como nuevas clases al conjunto de entrenamiento.

Las principales contribuciones de la tesis doctoral se presentan en los capítulos 6, 7, 8 y 9 que se estructuran como sigue. El capítulo 6 describe el método de los vectores comunes discriminantes extendidos con kernel (RKDCV [31]). Los capítulos 7 y 8 explican cómo obtener los vectores comunes discriminantes de forma incremental (IDCV) a partir de EVD y ortogonalización, respectivamente. El capítulo 9 presenta el método de vectores comunes discriminantes extendidos a partir de aprendizaje incremental (IRDCV). Los capítulos citados, además de exponer los métodos propuestos también presentan la validación empírica de los mismos. Por último, el capítulo 10 muestra las conclusiones principales obtenidas a partir del trabajo realizado, las publicaciones generadas a raíz de éste, y las lineas de investigación futuras.

Capítulo 6

Vectores comunes discriminantes extendidos con kernel

Resumen — Tanto para DCV como para KDCV, las suposiciones subyacentes para el uso de los vectores comunes garantizan siempre un error cero sobre el conjunto de entrenamiento. Sin embargo, no hay garantías sobre la probabilidad de error. Recientemente, se ha planteado una extensión del método DCV, llamada *Rough Discriminative Common Vector* (RDCV), donde se reinterpretan las bases del método DCV y se relajan, en cierta medida, las suposiciones. Con ello se consiguen dos cosas. Por un lado, desaparece la limitación (del método lineal) de que no puede haber más muestras que dimensiones en el conjunto de entrenamiento. Y por otro lado, se consigue una mejor capacidad de generalización del método en un abanico más amplio de problemas. En este capítulo, se propone la versión kernel del método RDCV, o dicho de otra forma, una extensión del método KDCV que conserva la misma línea en la que se ha propuesto el método lineal RDCV. Se estudia la capacidad de generalización del método respecto a los métodos basados en vectores comunes, frente a un determinado rango de problemas (con cierta variabilidad). Además se observa el comportamiento de este método frente a diferentes tamaños del conjunto de entrenamiento.

Contenido

6.1. Desarrollo teórico 48
6.2. Experimentos 51
6.3. Discusión 58

6.1. Desarrollo teórico

El método de vectores comunes discriminantes extendidos con kernel (*Rough Discriminative Common Vector with Kernels*, RKDCV) se puede ver como una versión no lineal del método RDCV [§4.5], o como una extensión del método KDCV [§5.4].

Si $\mathbf{X} \in \mathbb{R}^{d\times M}$ es el conjunto de entrenamiento centrado, la matriz kernel asociada a $\mathbf{X}$ es $\mathrm{K}_{ij} = \langle \phi(\mathrm{x}_i), \phi(\mathrm{x}_j)\rangle$. El apéndice A.6 muestra algunas de las funciones kernel más representativas.

Una vez la matriz kernel se calcula, ésta se centra en el espacio de características como

$$\begin{aligned} \mathrm{K}_c &= (\mathbf{X} - 1_M\mathbf{X})(\mathbf{X} - 1_M\mathbf{X})^T \\ &= K - 1_M K - K 1_M + 1_M K 1_M \end{aligned}$$

1_M es una matriz de tamaño $(M \times M)$ con todos sus elementos iguales a $(1/M)$.

La descomposición en valores y vectores propios de K_c, es

$$\mathrm{K}_c = \mathrm{U}\Lambda\mathrm{U}^T$$

Las matrices de dispersión total e intra-clase en el espacio reducido se han definido en las ecuaciones (5.14) y (5.15), de la sección 5.4, como $\tilde{S_T}^{\Phi}$ y $\tilde{S_W}^{\Phi}$, respectivamente.

Sea $\mathrm{V}\Delta\mathrm{V}^T$ la descomposición en valores y vectores propios de $\tilde{S_W}^{\Phi}$. El método RKDCV reduce el espacio rango original de $\tilde{S_W}^{\Phi}$ a los vectores propios asociados a los valores propios normalizados no nulos que satisfacen la condición:

$$(1-\alpha) \geq \sum_{i=1}^{k} \delta_i, \qquad i = 1, 2, \ldots, k, \ldots, r. \qquad \sum_{i=1}^{r} \delta_i = 1. \tag{6.1}$$

El valor de α indica la cantidad de varianza que se añade al espacio nulo de $\tilde{S_W}^{\Phi}$, donde $0 < \alpha < 1$. Los valores propios no nulos se han normalizado a partir de dividir cada valor propio por la suma de todos ellos.

El nuevo conjunto de vectores propios asociados a los valores propios no nulos y nulos de $\tilde{S_W}^{\Phi}$, a partir de α, es:

$$\begin{aligned} \mathrm{V}_\alpha &= [\mathrm{v}_1,\ \mathrm{v}_2,\ \ldots,\ \mathrm{v}_k] \\ \overline{\mathrm{V}}_\alpha &= [\mathrm{v}_{(k+1)},\ \mathrm{v}_{(k+2)},\ \ldots,\ \mathrm{v}_r,\ \mathrm{v}_{(r+1)},\ \ldots, \mathrm{v}_M] \end{aligned}$$

Los vectores que conforman V_α y $\overline{\mathrm{V}}_\alpha$, son un conjunto generador del espacio rango reducido y el espacio nulo extendido de $\tilde{S_W}^{\Phi}$, respectivamente. De este modo, $\overline{\mathrm{V}}_\alpha^T\, \tilde{S_W}^{\Phi}\, \overline{\mathrm{V}}_\alpha \approx 0$.

La matriz de proyección final es por tanto

$$\mathrm{W}_{rkdcv} = (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2}\overline{\mathrm{V}}_\alpha \tag{6.2}$$

Si el espacio nulo de $(\overline{\mathrm{V}}_\alpha^T \ \tilde{S}_B^{\ \Phi} \ \overline{\mathrm{V}}_\alpha)$ es no vacío las direcciones de proyección son rotadas para eliminar éste espacio nulo, y conseguir así que las nuevas matrices de dispersión total y inter-clase sean diagonales.

$$\overline{\mathrm{V}}_\alpha^T \tilde{S}_B^{\ \Phi} \overline{\mathrm{V}}_\alpha = \overline{\mathrm{V}}_\alpha^T \tilde{S}_T^{\ \Phi} \overline{\mathrm{V}}_\alpha = \overline{\mathrm{V}}_\alpha^T \Lambda \overline{\mathrm{V}}_\alpha = \mathrm{Y}\tilde{\Lambda}\,\mathrm{L}^T \tag{6.3}$$

La matriz L contiene los vectores propios asociados a los valores propios no nulos de $\left(\overline{\mathrm{V}}_\alpha^T \tilde{S}_B^{\ \Phi} \overline{\mathrm{V}}_\alpha\right)$. En este caso, la matriz de proyección final es

$$\mathrm{W}_{rkdcv} = (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2}\overline{\mathrm{V}}_\alpha\,\mathrm{L} \tag{6.4}$$

Los vectores comunes discriminantes extendidos no lineales del conjunto de entrenamiento se definen como

$$\mathrm{x}_{rkdcv} = (\mathrm{U}\Lambda^{-1/2}\overline{\mathrm{V}}_\alpha\,\mathrm{L})^T\mathrm{K}_c \tag{6.5}$$

si el espacio nulo de $\left(\overline{\mathrm{V}}_\alpha^T \ \tilde{S}_B^{\ \Phi} \ \overline{\mathrm{V}}_\alpha\right)$ es vacío, la matriz L es también vacía. En este documento, [ ] define una matriz vacía.

La figura 6.1 muestra un diagrama en bloques del método RKDCV, para un conjunto de entrenamiento $\mathbf{X} \in \mathbb{R}^{d\times M}$.

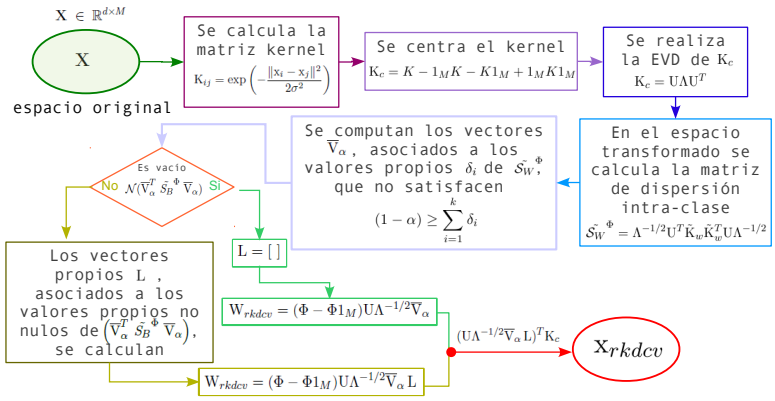


Figura 6.1: Proceso para obtener los vectores comunes discriminantes extendidos no lineales del conjunto de entrenamiento $\mathbf{X}$.

Los pasos para aplicar el método RKDCV a un conjunto de entrenamiento centrado $\mathbf{X} \in \mathbb{R}^{d\times M}$, se muestran en el algoritmo 2. Los

parámetros de entrada son el conjunto de entrenamiento y la varianza añadida al espacio nulo de $\tilde{S_W}^{\Phi}$. La matriz de proyección final, los vectores comunes discriminantes extendidos no lineales y la matriz kernel de entrenamiento, son los parámetros de salida.

Algoritmo 2 *Método RKDCV paso a paso.*

Entrada: $\mathbf{X} \in \mathbb{R}^{d \times M}$, α.
Salida: W_{rkdcv}, x_{rkdcv}, K.

1. Computar la matriz kernel del conjunto de entrenamiento. Normalmente es un kernel gaussiano [§A.6].

$$\mathrm{K}_{ij} = \exp\left(-\frac{\|\mathrm{x}_i - \mathrm{x}_j\|^2}{2\sigma^2}\right)$$

2. Centrar el kernel. $\mathrm{K}_c = K - 1_M K - K 1_M + 1_M K 1_M$, $1_M = (1/M)_{(M \times M)}$.
3. Descomponer K_c en términos de sus vectores y valores propios no nulos, $\mathrm{K}_c = \mathrm{U}\Lambda\mathrm{U}^T$.
4. Calcular la matriz de dispersión intra-clase en el espacio generado por los vectores U.

$$\tilde{S_W}^{\Phi} = \Lambda^{-1/2}\mathrm{U}^T\tilde{\mathrm{K}}_w\tilde{\mathrm{K}}_w^T\mathrm{U}\Lambda^{-1/2}$$

   donde $\tilde{\mathrm{K}}_w = (\mathrm{K} - 1_M\mathrm{K})(\mathrm{I} - \mathrm{G})$. I es la matriz identidad de tamaño $(M \times M)$ y $\mathrm{G} = diag[\mathrm{G}_1, \ \ldots \ , \mathrm{G}_c]$ es una matriz con submatrices $G_j \in \mathbb{R}^{m_j \times m_j}$ en su diagonal. Cada matriz G_j tiene todos sus elementos iguales a $(1/m_j)$.
5. Realizar la descomposición en valores y vectores propios de $\tilde{S_W}^{\Phi} = \mathrm{V}\Delta\mathrm{V}^T$.
6. Obtener los vectores propios $\overline{\mathrm{V}}_\alpha$, asociados a los valores propios Δ que no satisfacen

$$(1-\alpha) \geq \sum_{i=1}^{k} \delta_i, \qquad i = 1, 2, \ldots, k, \ldots, r. \qquad \sum_{i=1}^{r} \delta_i = 1.$$

   $\overline{\mathrm{V}}_\alpha = [\mathrm{v}_{(k+1)}, \ \mathrm{v}_{(k+2)}, \ \ldots, \ \mathrm{v}_r, \ \mathrm{v}_{(r+1)}, \ \ldots, \mathrm{v}_M]$.
7. Si el espacio nulo de $\left(\overline{\mathrm{V}}_\alpha^T \tilde{S_B}^{\Phi} \overline{\mathrm{V}}_\alpha\right)$ es no vacío, L es el conjunto de vectores propios asociados a los valores propios no nulos, en caso contrario L = [ ].

8. Calcular la matriz de proyección final como: $\mathrm{W}_{rkdcv} = (\Phi - \Phi 1_M)\mathrm{U}\Lambda^{-1/2}\overline{\mathrm{V}}_\alpha \mathrm{L}$.
9. Los vectores comunes discriminantes extendidos no lineales, son
$$\mathrm{x}_{rkdcv} = (\mathrm{U}\Lambda^{-1/2}\overline{\mathrm{V}}_\alpha\,\mathrm{L})^T\mathrm{K}_c$$
10. La clasificación de nuevas muestras, $\mathbf{X}^{test} \in \mathbb{R}^{d\times p}$, se realiza a partir de la distancia mínima entre los x_{rkdcv} y $\mathrm{x}^{test}_{rkdcv}$.

$$\mathrm{x}^{test}_{rkdcv} = (\mathrm{U}\Lambda^{-1/2}\overline{\mathrm{V}}_\alpha\,\mathrm{L})^T\,(\mathrm{K}^{test} - \mathrm{K}1'_M - 1_M\mathrm{K}^{test} + 1_M\mathrm{K}1'_M)$$

donde $\mathrm{K}^{test}_{ij} = \langle\phi(\mathrm{x}_i), \phi(\mathrm{x}^{test}_j)\rangle$ y $1'_M = (1/M)_{(M\times p)}$.

6.2. Experimentos

Estos experimentos se realizan con el objetivo de observar la tasa de acierto del método RKDCV en función de la varianza añadida al subespacio de proyección final y al número de muestras de entrenamiento. Además, se compara RKDCV con el método lineal no extendido DCV, así como con el método no lineal ni extendido KDCV y con la versión lineal RCV, para deducir si tiene o no mayor capacidad de generalización al clasificar las bases de datos empleadas. Las pruebas se han diseñado siguiendo el tipo de experimentación realizada en los trabajos originales donde se proponen los métodos [22], [105] y [20].

Las bases de datos elegidas para la validación del método cubren diversos dominios, y contienen imágenes en niveles de gris. Con el objetivo de estudiar el comportamiento de los métodos en situaciones más difíciles, se ha considerado la adición de diferentes tipos de ruido a algunas de las bases de datos. En concreto, se ha añadido ruido gaussiano de varianza 0.02, 0.04, 0.06 y 0.08, a las bases de datos AR Face [83] y ALL. AR y ALL con ruido añadido se les a dado el nombre de AR NG y ALL NG, respectivamente, y contiene imágenes de rostros de diferentes sujetos. ALL NG resulta de la selección aleatoria de 10 muestras sin oclusiones por clase de las bases de rostros AR Face [83], ORL [94], Yale [40] y Umist [115], obteniendo 95 clases. Cada clase de ALL NG tiene 50 muestras, de las cuales 10 no tienen ruido añadido y 40 presentan ruido gaussiano añadido (generadas a partir de las 10 imágenes originales de cada sujeto). Las características del ruido añadido son idénticas a las de AR NG. La sección D.1, del apéndice D, describe con detalle las características de AR NG y ALL NG.

Entre las bases de datos utilizadas para validar el método RKDCV también se encuentra una base de datos de imágenes de objetos llamada

Coil-40/30 [§D.2], que es un subconjunto de Coil-100 [87]. Además, hay una base de datos de dígitos de 0 a 9 escritos a mano llamada Mnist [72] [§D.3], donde el 10 % de las etiquetas de las muestras de entrenamiento se permutaron, sin favorecer a ninguna clase, para que el problema de clasificación presente más complejidad.

En todas las bases de datos, excepto en ALL NG, se cumple el caso del tamaño de muestras pequeño, que en nuestro caso se define como $d > (M - c)$. La tabla 6.1 muestra las características principales de las cuatro bases de datos utilizadas en la validación del método RKDCV.

Tabla 6.1: Características de las bases de datos usadas en la validación del método RKDCV. c es el número de clases, tamaño/c es el número de muestras por clase en toda la base de datos. m_j es el número de muestras por clase del conjunto de entrenamiento.

nombre	**tamaño imagen**	c	**tamaño**/c	m_j	**tipo**
AR NG	40×40	20	70	[7, 9, 14, 23, 35, 47]	rostros, expresión, iluminación & ruido gaussiano
Coil-40/30	40×40	40	30	[3, 4, 6, 10, 15, 20]	objetos, pose
Mnist	32×32	10	100	[10, 30, 20, 33, 50, 67]	dígitos de 0-9, intercambio de etiquetas
ALL NG	40×40	95	50	[5, 6, 10, 17, 25, 33]	rostros, expresión, iluminación, pose & ruido gaussiano

Los experimentos se realizan para varios tamaños relativos del conjunto de entrenamiento con respecto al número total de muestras disponibles, para estudiar la dependencia del método en relación a este parámetro. En particular, 1/10, 1/8, 1/5, 1/3, 1/2 y 2/3 del total de muestras disponibles se usan en el entrenamiento, y los experimentos se repiten 10, 8, 5, 3, 2 y 3 veces, respectivamente. El número de muestras de entrenamiento por clase en las diferentes iteraciones y para las diferentes bases de datos se muestra en la tabla 6.1. Los resultados de esta validación cruzada repetida en cuanto a la tasa de acierto son los respectivos valores promedio.

En la práctica, es interesante conocer el comportamiento de los métodos con un número reducido de muestras de entrenamiento ya que, obviamente, todos los métodos se aproximan al óptimo cuando este tamaño crece. Por lo tanto, en las siguientes subsecciones se muestran con más detalle los resultados para un valor representativo del tamaño del conjunto de entrenamiento.

La medida de distancia empleada entre las características obtenidas del conjunto de entrenamiento y las características del conjunto de prueba es la distancia euclídea. El valor del parámetro α, que añade varianza al subespacio de proyección, se incrementa en intervalos de 0.05 desde 0 a 0.5.

El kernel de los métodos basados en subespacios no lineales es de tipo gaussiano, por que es el normalmente utilizado en tareas de clasificación de patrones, como en [19], [20], [95] y [97].

Diferentes valores del parámetro de proximidad del kernel, σ, se han utilizado. En las bases de datos de rostros el intervalo de valores de σ está entre 20 y 500, y un buen resultado se obtiene cuando $\sigma = 100$. En la base de datos de objetos el intervalo de valores considerado es de 20 a 1400 y un resultado aceptable se ha obtenido con $\sigma = 1000$. El rango de valores considerado en la base de datos de dígitos es de 10 a 500, seleccionando $\sigma = 20$ como apropiado.

6.2.1. Experimentos con pocas muestras

La evaluación del método RKDCV en el caso del tamaño de muestras pequeño, $d > (M - c)$, emplea las bases de datos AR NG, Coil-40/30 y Mnist. La figura 6.2 muestra en cada fila algunas imágenes de estas bases de datos.

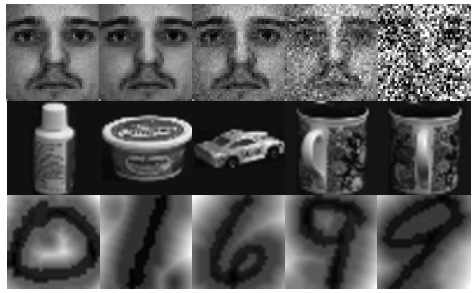
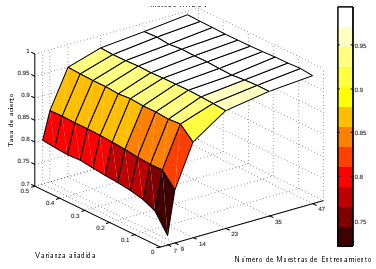
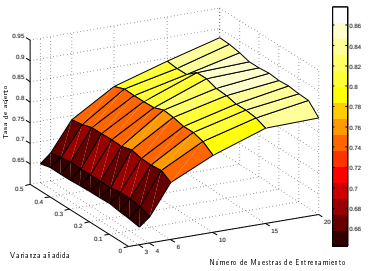


Figura 6.2: Bases de datos utilizadas en la validación del método RKDCV, para el caso del tamaño de muestras pequeño. Fila 1, ejemplo de la degradación de las imágenes con ruido gaussiano de 0, 0.02, 0.04, 0.06 y 0.08, en la base de rostros AR NG. Filas 2 y 3, algunas imágenes de las base de datos Coil-40/30 y Mnist, respectivamente.

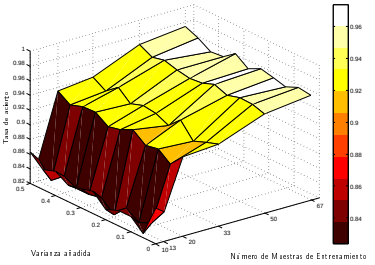
Para examinar en detalle el método RKDCV, se muestra en la figura 6.3 la tasa de acierto en función de la varianza añadida al subespacio de proyección final y en función del número de muestras de entrenamiento. En estos experimentos, el tamaño del conjunto de entrenamiento nunca supera la dimensión del espacio original. Se observa que, evidentemente y como era de esperar, a mayor número de muestras de entrenamiento tanto KDCV ($\alpha = 0$) como RKDCV ($\alpha > 0$) mejoran. Pero para el mismo número de muestras de entrenamiento el método RKDCV puede obtener mejores resultados al incrementar la varianza que se añade al subespacio de proyección final, como en el caso de AR NG y Coil-40/30.



(a) AR NG



(b) Coil-40/30

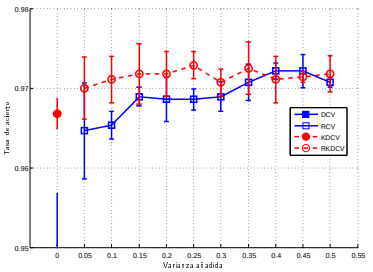


(c) Mnist

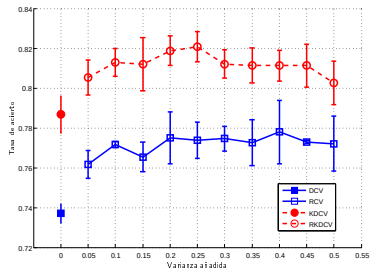
Figura 6.3: Tasa de acierto del método RKDCV en función del número de muestras de entrenamiento y la varianza añadida al subespacio de proyección.

Para un número representativo de 23, 10 y 50 muestras de entrenamiento por clase, para las bases de datos AR NG, Coil-40/30 y Mnist, respectivamente, la figura 6.4 muestra la tasa de acierto del método RKDCV al incrementar la varianza añadida al subespacio de proyección respecto a los métodos DCV, RCV y KDCV. Se observa que el método RKDCV tie-

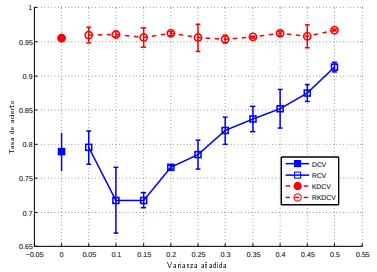
ne mayor poder de discriminación que los métodos DCV, RCV y KDCV. Estas diferencias son más evidentes en la base de datos Coil-40/30 ya que al aumentar la varianza del subespacio de proyección se puede ver claramente la superioridad del método propuesto.



(a) AR NG



(b) Coil-40/30



(c) Mnist

Figura 6.4: Tasa de acierto en función de la varianza añadida al subespacio de proyección de los métodos RCV, DCV, KDCV y RKDCV, para un determinado número de muestras de entrenamiento por clase de (a) 23 para AR NG donde la tasa de acierto del método DCV no es visible para poder observar mejor las diferencias presentes entre los métodos RKDCV y KDCV, (b) 10 para Coil-40/30 y (c) 50 para Mnist.

En la base de datos AR NG es de destacar que el método RKDCV presenta mejores resultados cuando la varianza añadida es de 0.25. Para valores de varianza mayores o iguales a 0.3 la tasa de acierto del método RKDCV es muy parecida a la del método lineal RCV. Para poder observar mejor las diferencias presentes entre los métodos RKDCV y KDCV, la tasa de acierto del método DCV no es visible en la figura 6.4a.

Respecto a la base de datos de dígitos Mnist, figura 6.4c, la tasa de acierto de los métodos lineales DCV y RCV no supera ni iguala a la de los métodos kernel.

Los resultados de la experimentación sugieren que incrementar la varianza del subespacio de proyección en el método RKDCV es eficiente siempre y cuando el número de muestras de entrenamiento no sea muy elevado, ya que el método KDCV ($\alpha = 0$) presenta un subespacio de proyección lo suficientemente discriminante cuando el tamaño del conjunto de entrenamiento es grande, lo cual no permite conseguir tasas de acierto mayores.

6.2.2. Experimentos con gran escala

En este caso se utiliza la base de imágenes de rostros ALL NG, que presenta cambios de expresión, iluminación, pose y ruido. La figura 6.5 expone algunas muestras de ALL NG.

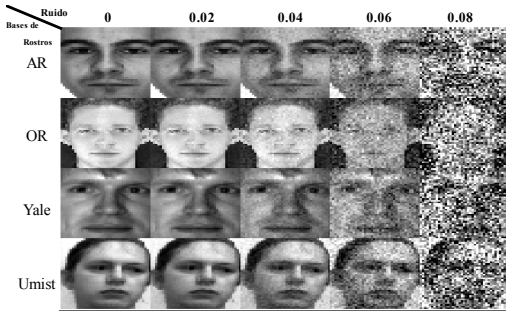


Figura 6.5: Ejemplo de la degradación de las imágenes en la base de datos ALL NG, que a su vez está formada por AR, ORL, Yale y Umist.

Con esta base de datos, significativamente más grande que las empleadas en la sección anterior, se estudia el comportamiento del método RKDCV respecto a RCV y KDCV, a medida que el tamaño del conjunto de entrenamiento es cada vez mayor. Concretamente, para 17 o más muestras por clase en el conjunto de entrenamiento la dimensión del espacio es menor al número total de muestras de entrenamiento.

La figura 6.6 presenta la tasa de acierto en función de la varianza añadida al subespacio de proyección de ALL NG, y en función del tamaño del conjunto de entrenamiento. Se observa que, a mayor número de

muestras de entrenamiento tanto KDCV ($\alpha = 0$) como RKDCV ($\alpha > 0$) obtienen mejores resultados. La tasa de acierto de RKDCV en relación con la de KDCV, presenta cambios más significativos cuando el tamaño del conjunto de entrenamiento es menor a la dimensión del espacio de las muestras. Para valores menores a 17 muestras por clase el método RKDCV consigue un 11,68 %, 9,95 % y 6,64 % más de aciertos. Si bien, para tamaños del conjunto de entrenamiento más grandes, también se obtienen mejoras.

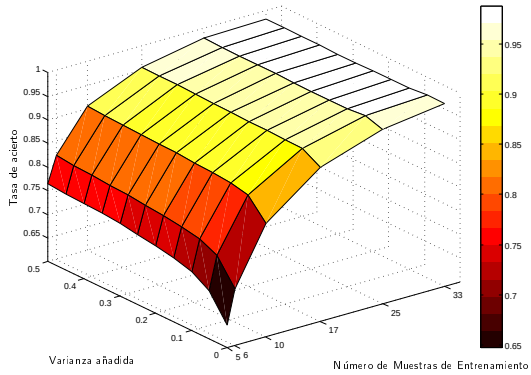


Figura 6.6: Tasa de acierto en función del número de muestras de entrenamiento y de la varianza añadida al subespacio de proyección de ALL NG, para los métodos KDCV ($\alpha = 0$) y RKDCV ($\alpha > 0$).

Por lo tanto, se pueden obtener tasas de acierto mayores a partir de incrementar la variabilidad que se añade al subespacio de proyección manteniendo un número de muestras de entrenamiento pequeño. Obviamente, otra opción que permite tasas de acierto altas es aumentar las muestras de entrenamiento, pero esto implica utilizar más recursos computacionales.

La figura 6.7 representa el comportamiento de los métodos RDCV, KDCV y RKDCV en función de la varianza añadida al subespacio de proyección, para aproximadamente el 33 % de las muestras totales en el conjunto de entrenamiento (17 muestras por clase). En este caso el método DCV no se aplica porque la dimensión del espacio original es menor al número de muestras de entrenamiento total. Desde esta figura, se observa que el método extendido RKDCV presenta mejores resultados que RDCV y KDCV. En concreto, cuando α =0.5 la tasa de acierto es de 0.96 respecto a 0.93 en RCV y 0.94 en KDCV.

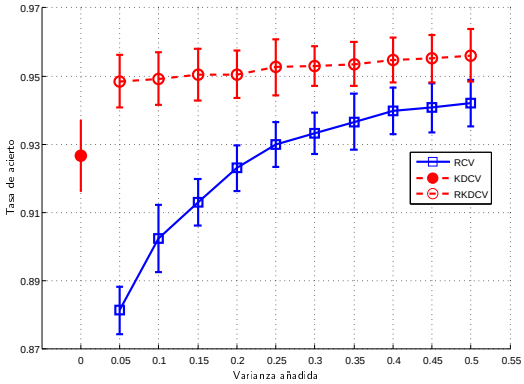


Figura 6.7: Tasa de acierto en función de la varianza añadida al subespacio de proyección final de ALL NG para los métodos RCV, KDCV y RKDCV, con 17 muestras de entrenamiento por clase.

6.3. Discusión

A partir de la experimentación se observa que el método RKDCV presenta propiedades discriminantes apropiadas para la clasificación de las bases de datos utilizadas, independiente de la relación entre la dimensión del espacio original del las muestras y el tamaño del conjunto de entrenamiento.

Si bien los métodos RCV y KDCV presentan buenas propiedades discriminantes a mayor número de muestras de entrenamiento. El método RKDCV presenta las mismas o mejores propiedades discriminantes utilizando un menor número de muestras de entrenamiento. Esto implica que las matrices involucradas para obtener el subespacio discriminante manejan menos cantidad de datos, y en consecuencia el esfuerzo computacional en el método RKDCV es menor.

Capítulo 7

Vectores comunes discriminantes mediante eigendescomposición incremental

Resumen — En muchas de las aplicaciones de clasificación donde se utilizan métodos basados en subespacios se necesita una realimentación informativa que resulta computacionalmente costosa si el sistema es reentrenado desde cero, cuando nuevas muestras de entrenamiento están disponibles. En el presente capítulo se propone una versión incremental del método de los vectores comunes discriminantes mediante la descomposición en valores y vectores propios respecto a nuevas muestras y nuevas clases de entrenamiento, para datos de muy alta dimensión.

Contenido

7.1. Preliminares 60
7.2. DCV mediante eigendescomposición incremental 64
7.3. Casos particulares del método IDCV-EVD 69
7.4. Evaluación empírica 72
7.5. Discusión 83

7.1. Preliminares

En el aprendizaje por lotes, todas las muestras se usan simultáneamente para computar el subespacio de proyección. En el aprendizaje incremental el subespacio existente se actualiza a partir de nuevas observaciones, por lo cual no es necesario disponer de todas las muestras de entrenamiento a la vez, reduciendo los requerimientos de almacenamiento y haciendo los grandes problemas computacionalmente factibles. El coste computacional relacionado con los métodos que emplean un aprendizaje incremental es eficiente en sentido relativo, no en sentido absoluto, ya que el entrenamiento incremental se realiza desde un modelo inicial precalculado, que conlleva un coste computacional previo.

Varios métodos de clasificación/representación basados en subespacios emplean alguna matriz de dispersión (estas matrices son semidefinidas positivas) para obtener la matriz de proyección. Normalmente, estas matrices se descomponen en términos de valores y vectores propios a partir de eigendescomposición o de descomposición en valores singulares.

Es lógico pensar que para realizar un aprendizaje incremental de estos métodos es necesario un desarrollo incremental de la eigendescomposición (o SVD). En general, éste es un problema clásico del análisis numérico y del álgebra computacional que se trata más adelante.

7.1.1. Antecedentes y estado del arte

Las primeras propuestas de un desarrollo incremental para métodos basados en subespacios se presentaron para la descomposición en valores singulares [41, 18, 17]. Estas aproximaciones permiten adicionar o quitar solo una muestra al conjunto de entrenamiento en el modelo previamente entrenado.

En 1982, Murakami et al. [86] plantearon la primera versión incremental para calcular las componentes principales de un conjunto de imágenes. La actualización en esta versión se realiza a partir de la adición de una sola muestra, y además, supone que la media de las muestras no cambia, por lo cual no es actualizada en el algoritmo.

Luego, Chandrasekeran et al. proponen un algoritmo incremental que se basa en la actualización de SVD [81, **?**]. Aunque la descomposición en valores singulares incremental fue abordada también anteriormente en [18, 43].

Todos estos métodos asumen que la media de las imágenes de entrada es siempre cero. Esto no es cierto en general, y esta suposición puede degradar los resultados de la clasificación [50].

Los primeros en incorporar la actualización de la media en algoritmos incrementales basados en subespacios fueron Hall et. al. en el año 1998 [50]. Pero no fue hasta el año 2000, donde los mismos autores propusieron en [51] un algoritmo incremental que actualiza la media y permite adicionar/quitar más de una muestra en el reentrenamiento.

Liu y Chen presentan en [78] un algoritmo incremental del análisis de componentes principales, aplicado al campo del vídeo, que proporciona pesos a la información del conjunto de entrenamiento. En este contexto, varios métodos con derivaciones diferentes se han propuesto [126, 116].

Franco et al. sugieren una aproximación para asociar múltiples espacios propios que puede verse como un PCA incremental, ya que adiciona información de nuevas muestras de forma sucesiva [37, 126]. En [100], Skocaj y Leonardis proponen un aprendizaje incremental robusto de subespacios propios para la detección de *outliers*.

El FLD también ha sido extendido de muchas maneras a una forma incremental, y para diferentes aplicaciones. Por ejemplo, desde redes neuronales para la extracción de características y la proyección de datos multivariantes [82], a EVD para la clasificación de cadenas de datos [91]. Mediante subespacios complementarios [117] para tareas de reconocimiento de rostros a sistemas de clasificación de patrones en linea [90]. También a partir de descomposición QR [125] y SVD [129], entre otros.

Pang et al. proponen el análisis discriminante lineal incremental de dos formas, una secuencial (*Sequential ILDA*) y otra por trozos (*Chunk ILDA*) [91]. En la forma *Chunk*, el espacio discriminante se actualiza con múltiples datos. Teóricamente, el algoritmo *Sequential ILDA* es un caso especial del *Chunk ILDA*.

Lin et al. exponen en [103] un modelo incremental discriminante para el seguimiento de objetos sometidos a grandes cambios de pose e iluminación. Kim et al. [70] proponen un método incremental para el aprendizaje de subespacios ortogonales, donde el algoritmo actualiza las componentes principales de la correlación de las clases, y luego calcula las componentes ortogonales de las componentes principales actualizadas.

Recientemente, Kim et al. presentan un algoritmo incremental del análisis discriminante lineal aplicado al reconocimiento de objetos. Ellos aplican el concepto de aproximación del conjunto generador suficiente (*sufficient spanning set*), para actualizar la matriz de dispersión interclases, la matriz de dispersión total y la matriz de proyección al espacio discriminante [67, 68].

Xinxiao et al. formulan en [118] un algoritmo incremental del análisis discriminante de correlación canónica [66], donde el modelo discriminante se actualiza de forma incremental para capturar cambios de apariencia humana, y facilitar la tarea de reconocimiento en entornos dinámicos.

Koç et al. proponen en [69] un algoritmo más rápido para el método original de vectores comunes. Este algoritmo realiza la clasificación con respecto a valores escalares, mientras que el método CV convencional usa vectores con dimensión igual a la del espacio de características.

Algunas aproximaciones incrementales de los métodos anteriores presentan desventajas en su precisión en comparación con los métodos con aprendizaje por lotes. Este efecto se amplifica debido a que los métodos incrementales se suelen aplicar repetidamente dando lugar a un aprendizaje continuo [51, 125].

7.1.2. Planteamiento incremental de DCV mediante eigendescomposición

El método DCV original, propuesto por Cevikalp et al. en [21, 22], procesa el conjunto de entrenamiento como un único conjunto. De esta manera, cuando nuevas muestras se incorporan al conjunto de entrenamiento, el proceso de aprendizaje se inicia desde cero.

Por lo tanto, para calcular los vectores comunes discriminantes de un conjunto de entrenamiento, primero se obtiene la matriz de dispersión intra-clase. Luego se descompone la matriz de dispersión en términos de sus vectores y valores propios, para obtener los vectores comunes. Después, la matriz de proyección que maximiza la dispersión de los vectores comunes se calcula.

De lo anterior se observa que el método DCV se compone principalmente de dos fases, como se muestra en las figuras 4.6 y 4.7. La primera fase consiste en computar el vector común de cada clase. La segunda radica en obtener la matriz de proyección final a partir de los vectores comunes.

De la sección 4.4 sabemos que la primera fase conlleva un coste computacional mucho mayor que la segunda fase. Ya que en la primera parte el tamaño de las matrices involucradas depende de la dimensionalidad y del número de muestras total de entrenamiento. En cambio, la segunda fase es independiente del número de muestras de entrenamiento, y sólo depende del numero de clases. Por lo tanto, la segunda fase es un problema mucho más pequeño computacionalmente y el verdadero cuello de botella está en la primera fase.

Por consiguiente, se plantea un desarrollo incremental de la primera fase del método DCV que busca actualizar las matrices de proyección a partir de un modelo precalculado, cada vez que hay nuevas muestras disponibles para el entrenamiento, y poder así reducir el coste computacional.

La figura 7.1 muestra de forma general la estructura del aprendizaje por lotes e incremental del método DCV. **X** es el conjunto de entrenamiento inicial que generan los vectores comunes discriminantes x^j_{dcv}.

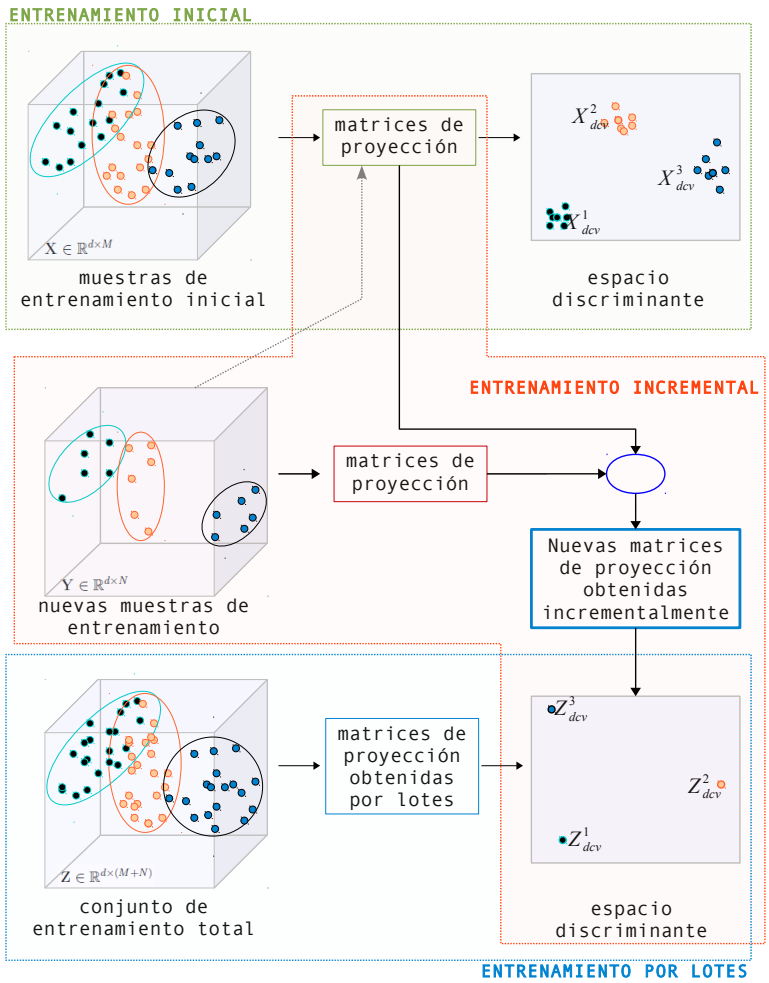


Figura 7.1: Esquema general del aprendizaje por lotes e incremental en el método DCV. **X** es el conjunto de entrenamiento inicial. **Y** es el nuevo conjunto para la actualización. **Z** es el conjunto de entrenamiento total formado por **X** e **Y**. z^j_{dcv} son los vectores comunes discriminantes resultantes del aprendizaje por lotes y del aprendizaje incremental.

$\mathbf{Y} \in \mathbb{R}^{d \times N}$ es el conjunto de entrenamiento que actualiza de forma incremental la información de $\mathbf{X}$. El conjunto de entrenamiento concatenado se define como $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}] \in \mathbb{R}^{d \times (M+N)}$ y genera z_{dcv}^{j}. Los vectores comunes discriminantes z_{dcv}^{j}, deberían ser idénticos a los obtenidos a partir de actualizar la información de $\mathbf{X}$ con $\mathbf{Y}$. El tamaño de los nuevos datos, N, puede ser variable aunque se supone que $M \gg N$. También se tratara separadamente el caso en el que los nuevos datos $\mathbf{Y}$ contengan nuevas clases o no.

En cuanto a la segunda fase, dado que el subespacio rango y el subespacio diferencia de los vectores comunes son subespacios equivalentes [22], la matriz de proyección final se puede obtener a partir de los vectores propios asociados a los valores propios no ceros de la matriz de dispersión de los vectores comunes o, a partir de una base ortonormal de el subespacio diferencia de los z_{cv}^{j}, tal y como se ha visto en la sección 4.4. Aunque, el proceso de ortogonalización sólo necesita $O(dc^2)$ operaciones y la eigendescomposición emplea $O(\ell(c^3) + dc^2)$, la matriz de proyección final, W_{dcv}, se calculará preferentemente a partir de ortogonalización.

7.2. DCV mediante eigendescomposición incremental

Esta sección presenta una versión incremental del método DCV original mediante eigendescomposición, donde las matrices de proyección se pueden actualizar a partir de la información resultante del entrenamiento anterior y de las nuevas muestras disponibles para el entrenamiento. De este modo, se incorpora nueva información a la base de conocimiento a partir de las nuevas muestras.

7.2.1. Desarrollo teórico

Para obtener los vectores comunes discriminantes del conjunto de entrenamiento total, $\mathbf{Z}$, a partir del método DCV-EVD, se tienen que computar los vectores y valores propios de la matriz de dispersión intra-clase S_Z^w, para conseguir primero los vectores comunes [§4.4.1].

Para evitar la EVD explícita de S_Z^w, se realiza una descomposición más simple. Esta descomposición está en términos de las matrices de dispersión intra-clase de $\mathbf{X}$ y de $\mathbf{Y}$. Además, se introduce una matriz S que relaciona las medias de los datos de entrenamiento iniciales, x_j, y las medias de los nuevos datos, y_j.

En particular, la matriz S_Z^w se puede escribir como

$$\begin{aligned} S_{\mathbf{Z}}^w &= \sum_{j=1}^{c}\sum_{i=1}^{o_j}(\mathrm{z}_j^i - \mathrm{z}_j)(\mathrm{z}_j^i - \mathrm{z}_j)^T \\ &= \sum_{j=1}^{c}\left[\sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{z}_j)(\mathrm{x}_j^i - \mathrm{z}_j)^T + \sum_{i=1}^{n_j}(\mathrm{y}_j^i - \mathrm{z}_j)(\mathrm{y}_j^i - \mathrm{z}_j)^T\right] \\ &= S_X^w + S_Y^w + S \end{aligned} \tag{7.1}$$

donde $\mathrm{z}_j = \left(\frac{m_j}{(m_j+nj)}\,\mathrm{x}_j + \frac{n_j}{(m_j+nj)}\,\mathrm{y}_j\right)$, x_j e y_j son las medias de la clase j en $\mathbf{Z}$, $\mathbf{X}$ e $\mathbf{Y}$, respectivamente. $M = \sum_{j=1}^{c} m_j$ es el número de muestras en $\mathbf{X}$ y, $N = \sum_{j=1}^{c} n_j$ es el número de muestras en $\mathbf{Y}$. $o_j = (m_j + n_j)$ es el número de muestras de la clase j en $\mathbf{Z}$.

La matriz S se define como $S = \mathrm{J}\mathrm{J}^T$, donde J es una matriz de tamaño $(d \times c)$ cuyas columnas son proporcionales a la diferencia entre los vectores media de cada clase de $\mathbf{X}$ y de $\mathbf{Y}$, y se define a su vez como

$$\mathrm{J} = [\mathrm{J}_1 \ldots \mathrm{J}_c] \qquad \text{con} \qquad \mathrm{J}_j = \sqrt{\frac{m_j n_j}{(m_j + n_j)}}\,(\mathrm{x}_j - \mathrm{y}_j) \tag{7.2}$$

para todo $j = 1, \ldots, c$.

Los pasos intermedios para obtener la descomposición de S_Z^w en términos de S_X^w, S_Y^w y S, se muestran detalladamente en el apéndice B. La ecuación (7.1) puede verse como una generalización de la descomposición propuesta por Hall et. al. para el método PCA [50, 51].

La finalidad de descomponer la matriz de dispersión intra-clase de $\mathbf{Z}$, es obtener la EVD, $S_Z^w = \mathrm{U}'\Lambda'\mathrm{U}'^T$, en función de los valores y vectores propios de $S_X^w = \mathrm{U}\Lambda\mathrm{U}^T$ (modelo precalculado) y de los de $S_Y^w = V\Delta V^T$ (información incremental). De este modo, los nuevos vectores comunes se pueden obtener a partir de los vectores y valores propios no nulos de S_Z^w.

Asociados a los tres términos de la ecuación (7.1), se tienen tres subespacios: dos subespacios correspondientes a los rangos de los datos de $\mathbf{X}$ e $\mathbf{Y}$, y un tercer subespacio asociado a las diferencias de sus vectores medias. El subespacio de rango asociado a los datos $\mathbf{Z}$, es la suma de los tres subespacios anteriores. Por lo tanto,

$$\mathcal{R}(S_Z^w) = \mathcal{R}(S_X^w) + \mathcal{R}(S_Y^w) + span\{\mathrm{J}\} \tag{7.3}$$

Si la suma de la ecuación (7.3) es una suma directa, una base ortonormal del subespacio $\mathcal{R}(S_Z^w)$ se obtiene de la concatenación de las bases ortonormales de los subespacios $\mathcal{R}(S_X^w)$, $\mathcal{R}(S_Y^w)$ y $span\{\mathrm{J}\}$, como lo muestra el teorema 3 del apéndice C. En el caso más general, puede existir algún tipo de dependencia lineal entre el modelo precalculado,

las nuevas muestras y la relación entre las medias de las clases, como se muestra en la figura 7.2.

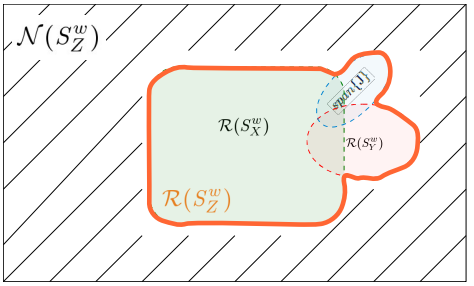


Figura 7.2: Subespacio rango de la matriz de dispersión S_Z^w, conformado por la suma de los subespacios rango de S_X^w y S_Y^w, y del subespacio generado por las diferencias de los vectores media de cada clase en $\mathbf{X}$ e $\mathbf{Y}$.

Por lo tanto, para obtener una base ortonormal de $\mathcal{R}(S_Z^w)$ en función de U, se debe encontrar una base ortonormal del subespacio $\mathcal{R}(S_Y^w)$ y $span\{\mathrm{J}\}$ que sea ortogonal a U. Una forma de hacer esto es proyectar V y J en el subespacio nulo de S_X^w, y conseguir así que la intersección entre el subespacio $\mathcal{R}(S_X^w)$ y los nuevos subespacios sea nula. Esto se puede conseguir con el operador de proyección ortogonal $\overline{\mathrm{U}}\ \overline{\mathrm{U}}^T$ o su equivalente $(\mathrm{I} - \mathrm{U}\mathrm{U}^T)$.

De esta forma, el subespacio generado por $span\{\overline{\mathrm{U}}\ \overline{\mathrm{U}}^T[\mathrm{V}\ \mathrm{J}]\}$ es ortogonal a $\mathcal{R}(S_X^w)$, y juntos conforman el subespacio de rango asociado a S_Z^w.

$$\mathcal{R}(S_Z^w) = \mathcal{R}(S_X^w) \oplus span\{\overline{\mathrm{U}}\ \overline{\mathrm{U}}^T[\mathrm{V}\ \mathrm{J}]\} \tag{7.4}$$

En este punto solo resta obtener una base de estos subespacios proyectados, que se puede calcular mediante ortogonalización. En particular

$$v = orth\left(\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T\ [\mathrm{V}\ \mathrm{J}]\right) \tag{7.5}$$

Finalmente, la base ortonormal de $\mathcal{R}(S_Z^w)$ se construye a partir de la base ortonormal del subespacio rango de S_X^w, aumentada con un conjunto de vectores ortonormales, v, al espacio nulo de S_X^w que se obtienen a partir de V y J. De manera que [U v] es una base ortonormal de $\mathcal{R}(S_Z^w)$.

Si el nuevo conjunto de muestras no es linealmente independiente respecto al anterior, los vectores columna que son ceros en $(\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T[\mathrm{V}\ \mathrm{J}])$

se eliminan después de la ortogonalización. Por el contrario, si el nuevo conjunto de entrenamiento es linealmente independiente respecto al anterior y a sí mismo, se cumplirá que $v \in \mathbb{R}^{d \times N}$. En general $[\mathrm{U}\ v] \in \mathbb{R}^{d \times r_z}$, donde r_z es la dimensión del espacio rango de la matriz de dispersión intra-clase del conjunto de entrenamiento $\mathbf{Z}$.

Por tanto, y como en el caso del aprendizaje por lotes, la dimensión del espacio rango aumenta a medida que hay nuevas muestras linealmente independientes para el entrenamiento, mientras la dimensión del espacio nulo disminuye.

La diferencia entre la base U' (de la EVD de S_Z^w) y la base $[\mathrm{U}\ v]$ (conseguida de forma incremental) es sólo una rotación R.

$$\mathrm{U}' = [\mathrm{U}\ v]\,\mathrm{R}$$

La matriz de rotación se puede relacionar con la descomposición en vectores y valores propios de la matriz S_Z^w, que podemos reescribir como:

$$S_Z^w = S_X^w + S_Y^w + \mathrm{J}\mathrm{J}^T = [\mathrm{U}\ v]\,\mathrm{R}\,\Lambda'\,\mathrm{R}^T\,[\mathrm{U}\ v]^T \tag{7.6}$$

donde $[\mathrm{U}\ v]^T$ es una inversa izquierda de $[\mathrm{U}\ v]$. Multiplicando ambos lados de la ecuación (7.6) por la izquierda por $[\mathrm{U}\ v]^T$, y por la derecha por $[\mathrm{U}\ v]$, se obtiene

$$[\mathrm{U}\ v]^T\,(S_X^w + S_Y^w + \mathrm{J}\mathrm{J}^T)\,[\mathrm{U}\ v] = \mathrm{R}\,\Lambda'\,\mathrm{R}^T \tag{7.7}$$

Por lo tanto, la matriz de rotación R y la matriz diagonal de valores propios Λ', se pueden calcular a partir de la diagonalización de la matriz de la ecuación (7.7), de una manera similar a la propuesta por Hall et. al. en [50, 51] para el PCA incremental.

En el caso del método DCV no es necesario calcular la rotación R, ya que el objetivo es encontrar el operador de proyección, $\overline{\mathrm{U}'}\,\overline{\mathrm{U}'}^T$, al espacio nulo de S_Z^w, que se obtiene implícitamente a partir del operador de proyección, $\mathrm{U}'\,\mathrm{U}'^T$, al espacio rango de S_Z^w. De la forma

$$\begin{aligned} \mathrm{U}'\mathrm{U}'^T &= [\mathrm{U}\ v]\,\cancelto{\mathrm{I}}{\mathrm{R}\mathrm{R}^T}\,[\mathrm{U}\ v]^T \\ &= [\mathrm{U}\ v]\,[\mathrm{U}\ v]^T \\ &= \mathrm{U}\mathrm{U}^T + vv^T \end{aligned} \tag{7.8}$$

donde

$$\overline{\mathrm{U}'}\,\overline{\mathrm{U}'}^T = (\mathrm{I} - \mathrm{U}'\mathrm{U}'^T) = (\mathrm{I} - \mathrm{U}\mathrm{U}^T - vv^T) \tag{7.9}$$

De este modo, sólo es necesario computar v para obtener el operador de proyección al espacio nulo de S_Z^w, y no es necesario resolver el eigenproblema de la ecuación (7.7).

Además, a partir de v también se pueden calcular directamente los nuevos vectores comunes como una actualización de los anteriores de la siguiente manera:

$$\begin{aligned} \mathrm{z}_{cv}^{j} &= \overline{\mathrm{U}'}\,\overline{\mathrm{U}'}^{T}\,\mathrm{z}_{j}^{i} = (\mathrm{I} - \mathrm{U}'\mathrm{U}'^{T})\,\mathrm{z}_{j}^{i} \\ &= (\mathrm{I} - \mathrm{U}\mathrm{U}^{T} - vv^{T})\,\mathrm{z}_{j}^{i} \\ &= \mathrm{x}_{cv}^{j} - vv^{T}\mathrm{x}_{j}^{i} \end{aligned} \tag{7.10}$$

El desarrollo paso a paso del método IDCV-EVD para incorporar nuevas muestras a las clases del conjunto de entrenamiento se muestra en el algoritmo 3.

Algoritmo 3 *Método IDCV-EVD que añade información a las clases del conjunto de entrenamiento* $\mathbf{X}$.

Entrada: $\mathrm{U} \in \mathbb{R}^{d \times r_x}$, $\mathbf{Y} \in \mathbb{R}^{d \times N}$, x_j, x_{cv}^{j}.
Salida: W_{dcv}, z_{dcv}^{j}, $[\mathrm{U}\ v]$, z_j.

1. Se calculan los vectores propios, V, asociados a los valores propios no nulos de la matriz de dispersión intra-clase S_Y^w.

2. La matriz $\mathrm{J} = [\mathrm{J}_1\ \ldots\ \mathrm{J}_c]$ se construye a partir de

$$\mathrm{J}_j = \sqrt{\frac{m_j n_j}{(m_j + n_j)}}\,(\mathrm{x}_j - \mathrm{y}_j)$$

3. Se computa v como: $v = orth\left(\left[\mathrm{I} - \mathrm{U}\mathrm{U}^{T}\right]\ [\mathrm{V}\ \mathrm{J}]\right)$.
4. La media de cada clase en $\mathbf{Z}$ es: $\mathrm{z}_j = \frac{m_j}{(m_j+n_j)}\,\mathrm{x}_j + \frac{n_j}{(m_k+n_j)}\,\mathrm{y}_j$.
5. $\mathrm{U}'\mathrm{U}'^{T} = [\mathrm{U}\ v]\,[\mathrm{U}\ v]^{T} = \mathrm{U}\mathrm{U}^{T} + vv^{T}$ es el nuevo operador de proyección al espacio rango de S_Z^w.
6. $\mathrm{z}_{cv}^{j} = \mathrm{x}_{cv}^{j} - vv^{T}\mathrm{x}_{j}^{i}$.
7. Luego, la matriz de proyección final se obtiene a partir de una base ortonormal del subespacio diferencia de los z_{cv}^{j}.

$$\mathrm{W}_{dcv} = orth([\mathrm{z}_{cv}^{2} - \mathrm{z}_{cv}^{1}\ \ldots\ \mathrm{z}_{cv}^{c} - \mathrm{z}_{cv}^{1}])$$

8. Los DCVs del conjunto de entrenamiento $\mathbf{Z}$ son,

$$\mathrm{z}_{dcv}^{j} = \mathrm{W}_{dcv}^{T}\mathrm{z}_{j}^{i}$$

En el próxima iteración: $\mathrm{U} = [\mathrm{U}\ v]$, $\mathrm{x}_j = \mathrm{z}_j$, $\mathrm{x}_{cv}^{j} = \mathrm{z}_{cv}^{j}$.

Los parámetros de entrada del algoritmo 3 son la matriz de vectores propios asociada a los valores propios no nulos de S_X^w, el nuevo conjunto de entrenamiento $\mathbf{Y}$, los vectores media de cada clase en $\mathbf{X}$ y los vectores comunes x_{cv}^j. Los parámetros de salida son la matriz de proyección final W, los vectores comunes discriminantes de $\mathbf{Z}$, la nueva matriz [U v] y los vectores media de cada clase actualizados.

7.2.2. Coste computacional

En el caso del aprendizaje por lotes, el coste computacional del método DCV-EVD para el conjunto de datos $\mathbf{Z} \in \mathbb{R}^{d\times(M+N)}$, está dominado por el cálculo y la EVD de S_Z^w, que implica $O(\ell((M+N)^3)+d(M+N)^2)$ operaciones.

Al incorporar nuevas muestras al conjunto de entrenamiento, el coste computacional del algoritmo IDCV-EVD está dominado por el cálculo de v, que conlleva $O(\ell(N^3) + dNr_x)$ operaciones, resultantes de la descomposición en valores y vectores propios de S_Y^w y de la ortogonalización de $\left(\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T\,[\mathrm{V\ J}]\right)$. r_x es el rango de S_X^w, y es como máximo igual a $(M-c)$ si todas las muestras en $\mathbf{X}$ son linealmente independientes.

En el contexto práctico de aplicación del método, el número de muestras de entrenamiento utilizadas en la actualización es significativamente menor al número de muestras empeladas en el modelo inicial. Dicho de otra forma $M >> N$. Por lo tanto, $(M+N)^3 >> N^3$ y $(M+N)^2 >> Nr_x$.

7.3. Casos particulares del método IDCV-EVD

El método IDCV-EVD propuesto en la sección anterior permite incorporar nuevas muestras a las clases del conjunto de entrenamiento $\mathbf{X}$.

En aquellos casos donde sólo una muestra por clase, o nuevas clases, se integran al conjunto de entrenamiento, el planteamiento del método IDCV-EVD cambia ligeramente. Las secciones 7.3.1 y 7.3.2 especifican con detalle el método IDCV-EVD para estos dos casos.

7.3.1. Una nueva muestra por clase

En procesos (en línea) donde las muestras para el entrenamiento están disponibles de una en una, $n_j = 1$ para todo $j = 1,\ldots,c$, la

matriz de dispersión intra-clase del nuevo conjunto de entrenamiento no se puede calcular, porque se necesitan como mínimo dos muestras por clase para su cómputo. En consecuencia, la descomposición de la matriz de dispersión intra-clase del conjunto de entrenamiento $\mathbf{Z}$ difiere de la presentada en la ecuación (7.1). En este caso, la matriz de dispersión intra-clase S_Z^w se puede descomponer como

$$\begin{aligned} S_Z^w &= \sum_{j=1}^{c}\sum_{i=1}^{o}(\mathrm{z}_j^i - \mathrm{z}_j)(\mathrm{z}_j^i - \mathrm{z}_j)^T \\ &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{z}_j)(\mathrm{x}_j^i - \mathrm{z}_j)^T + \sum_{j=1}^{c}(\mathrm{y}_j^1 - \mathrm{z}_j)(\mathrm{y}_j^1 - \mathrm{z}_j)^T \\ &= S_X^w + S \end{aligned} \tag{7.11}$$

donde $\mathrm{z}_j = \frac{1}{o_j}\sum_{i=1}^{o_j}\mathrm{z}_j^i = \left(\frac{m_j}{(m_j+1)}\,\mathrm{x}_j + \frac{1}{(m_j+1)}\,\mathrm{y}_j^1\right)$ y $o_j = m_j + 1$ son el vector media y el número de muestras en cada clase de $\mathbf{Z}$, respectivamente. x_j es el vector media de la clase j en $\mathbf{X}$. $S = \mathrm{J}\mathrm{J}^T$, con $\mathrm{J} = [\mathrm{J}_1\ \ldots\ \mathrm{J}_c]$ y $\mathrm{J}_j = \sqrt{\frac{m_j}{(m_j+1)}}\,(\mathrm{x}_j - \mathrm{y}_j^1)$. El apéndice B muestra de forma detallada cómo obtener la ecuación (7.11).

En este caso, el espacio rango de S_Z^w es el resultado de sumar el subespacio $\mathcal{R}(S_X^w)$ y el subespacio generado por J. Por lo tanto, una base ortonormal de $\mathcal{R}(S_Z^w)$ puede estar formada por la base ortonormal U (del entrenamiento anterior) y una base ortonormal v del subespacio generado por J que sea ortogonal a U.

Como en la sección anterior, v se puede obtener a partir de calcular una base ortonormal del subespacio que se obtiene al proyectar J en el espacio nulo de S_X^w, garantizando así que los vectores que conforman v son ortogonales a U y ortonormales entre si. Es decir,

$$v = orth(\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T\,\mathrm{J}) = orth(\mathrm{J} - \mathrm{U}\mathrm{U}^T\,\mathrm{J}) \tag{7.12}$$

Una vez se obtiene v, el operador de proyección $\overline{\mathrm{U}'}\,\overline{\mathrm{U}'}^T$ al espacio nulo de S_Z^w se calcula implícitamente, y con él los nuevos vectores comunes.

$$\overline{\mathrm{U}'}\,\overline{\mathrm{U}'}^T = (\mathrm{I} - \mathrm{U}'\mathrm{U}'^T) = (\mathrm{I} - \mathrm{U}\mathrm{U}^T - vv^T)$$

En este caso, los puntos 1-4 del algoritmo 3 se sustituyen por:

1. (no es necesario)
2. Se calcula $\mathrm{J} = [\mathrm{J}_1 \ldots \mathrm{J}_c]$, con $\mathrm{J}_j = \sqrt{\frac{m_j}{(m_j+1)}}\,(\mathrm{x}_j - \mathrm{y}_j^1)$.
3. Se computa v como: $v = orth(\left[\mathrm{I} - \mathrm{U}\mathrm{U}^T\right]\,\mathrm{J})$.
4. La media de cada clase en $\mathbf{Z}$ es: $\mathrm{z}_j = \frac{m_j}{(m_j+1)}\,\mathrm{x}_j + \frac{1}{(m_j+1)}\,\mathrm{y}_j^1$.

La segunda fase del algoritmo en la que se calculan los vectores comunes discriminantes no sufre cambio alguno con respecto al algoritmo 3.

Aquí se ha considerado una muestra por clase, pero también podría aplicarse el método cuando una sola muestra que pertenece a una clase arbitraria se adiciona.

El coste computacional del método IDCV-EVD cuando $n_j = 1$, está dominado por el cálculo de v, donde $O(dNr_x)$ operaciones son necesarias. $N = c$ y r_x es como máximo $(M - c)$.

7.3.2. Nuevas clases en el conjunto de actualización

Es posible que en algunas aplicaciones sea conveniente añadir información sobre nuevas clases que no estaban presentes al inicio del entrenamiento. En este caso, no se busca actualizar la información de las clases del conjunto de entrenamiento, sino que se busca incorporar nuevas clases a la base de conocimiento. Por esto, se propone una versión incremental del método DCV-EVD que permite adicionar nuevas clases al conjunto de entrenamiento sin reentrenar el sistema desde cero.

En aquellos casos donde las muestras presentes en $\mathbf{Y}$ no pertenecen a ninguna de las clases en $\mathbf{X}$, la descomposición de la matriz S_Z^w se puede escribir en función de las matrices de dispersión intra-clase de $\mathbf{X}$ e $\mathbf{Y}$ exclusivamente.

$$\begin{aligned} S_Z^w &= \sum_{j=1}^{c_z}\sum_{i=1}^{o}(\mathrm{z}_j^i - \mathrm{z}_j)(\mathrm{z}_j^i - \mathrm{z}_j)^T \\ &= \sum_{j=1}^{c_x}\sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T + \sum_{j=1}^{c_y}\sum_{i=1}^{n_j}(\mathrm{y}_j^i - \mathrm{y}_j)(\mathrm{y}_j^i - \mathrm{y}_j)^T \\ &= S_X^w + S_Y^w \end{aligned} \tag{7.13}$$

donde, c_x, c_y y $c_z = c_x + c_y$ son el número de clases que tienen los conjuntos de entrenamiento $\mathbf{X}$, $\mathbf{Y}$ y $\mathbf{Z}$, respectivamente.

Si el nuevo conjunto de entrenamiento es linealmente independiente respecto a $\mathbf{X}$, el nuevo operador de proyección al espacio rango de S_Z^w está definido por $[\mathrm{U}\ \mathrm{V}]\ [\mathrm{U}\ \mathrm{V}]^T$. $\mathrm{U} \in \mathbb{R}^{d\times r_x}$ y $\mathrm{V} \in \mathbb{R}^{d\times r_y}$ son las matrices de vectores propios asociadas a los valores propios no nulos de S_X^w y S_Y^w, respectivamente. r_x y r_y representan la dimensión del espacio rango de S_X^w y S_Y^w. Pero al igual que en los casos anteriores, no podemos garantizar que $\mathbf{Y}$ sea linealmente independiente respecto a $\mathbf{X}$.

Para eliminar cualquier dependencia lineal del nuevo conjunto de entrenamiento respecto a $\mathbf{X}$, se proyecta V en el espacio nulo de S_X^w de

la misma manera que en los casos anteriores. De este modo, $[\mathrm{V} - \mathrm{U}\mathrm{U}^T\mathrm{V}]$ es ortogonal a U, pero los vectores que conforman $[\mathrm{V} - \mathrm{U}\mathrm{U}^T\mathrm{V}]$ no son ortonormales entre si y es necesario ortogonalizar. Resultando

$$v = orth\left(\left[\mathrm{V} - \mathrm{U}\mathrm{U}^T\mathrm{V}\right]\right) \tag{7.14}$$

A partir de v, el nuevo operador de proyección a $\mathcal{R}(S_Z^w)$ se define como $\mathrm{U}'\mathrm{U}'^T = [\mathrm{U}\ v]\ [\mathrm{U}\ v]^T$.

Los nuevos vectores comunes, la matriz de proyección final y los vectores comunes discriminantes se calculan como se ha explicado anteriormente. Ahora el número de clases presentes en el conjunto de entrenamiento $\mathbf{Z}$ es igual a la suma del número de clases presentes en $\mathbf{X}$ e $\mathbf{Y}$, donde $c_z = c_x + c_y$. Por lo tanto, el tamaño de la matriz de proyección final cambia de $(c_x - 1)$ (del entrenamiento inicial) a $(c_x + c_y - 1)$ (por el nuevo conjunto de entrenamiento), es decir $\mathrm{W} \in \mathbb{R}^{d \times (c_x + c_y - 1)}$, si los vectores comunes discriminantes son linealmente independientes.

El algoritmo 3, de la sección 7.2.1, adaptado a este caso modifica los puntos 2-4 por:

2. (no es necesario)
3. v se define como: $v = orth([\mathrm{I} - \mathrm{U}\mathrm{U}^T]\,\mathrm{V})$.
4. La media de cada clase en $\mathbf{Z}$ es: $\mathrm{z}_j = \mathrm{x}_j$ para $j = 1, \ldots, c_x$, y $\mathrm{z}_j = \mathrm{y}_j$ para $j = c_x + 1, \ldots, (c_x + c_y)$.
   En el resto del algoritmo el recorrido del índice j es ahora $j = 1, \ldots, c_z$.

El método DCV-EVD con aprendizaje por lotes realiza $O(\ell((M+N)^3) + d(M+N)^2)$ operaciones para obtener el operador de proyección $\mathrm{U}'\mathrm{U}'^T$, mientras que el algoritmo incremental sólo realiza $O(\ell(N^3) + d\,r_x r_y)$ operaciones a partir de U y del nuevo conjunto de entrenamiento $\mathbf{Y}$. r_x y r_y representan el rango de las matrices de dispersión S_X^w y S_Y^w, respectivamente. En nuestro caso $M \gg N$, y por tanto, $(M+N)^3 >> N^3$ en la EVD y $(M+N)^2 >> r_x r_y$ para el cómputo de v. El máximo valor de r_x y r_y se presenta cuando las muestras en $\mathbf{X}$ y $\mathbf{Y}$ son linealmente independientes, es decir $r_x = (M - c_x)$ y $r_y = (N - c_y)$.

7.4. Evaluación empírica

La validación empírica del método IDCV-EVD consiste en comparar la tasa de acierto y sobre todo el tiempo de entrenamiento respecto al método original DCV-EVD, al clasificar diferentes bases de datos.

La clasificación se realiza a partir de la distancia euclídea entre el 1-vecino más próximo de las características obtenidas del conjunto de entrenamiento y las características del conjunto de prueba, como es habitual en trabajos anteriores [21, 22].

Todos los experimentos se han realizado 10 veces. Los resultados que se presentan corresponden al valor medio y a sus desviaciones estándar. El conjunto de entrenamiento y el conjunto de prueba son seleccionados aleatoriamente cada vez para eliminar cualquier tipo de dependencia en la clasificación.

Los algoritmos incrementales se han implementado en Matlab y se ejecutan en un equipo con procesador Dual Core AMD Opteron(tm) 270 con 1000,00 MHz de velocidad, 7,8 GB de RAM y Linux 2.6.25.20-0.5-default x86_64 como sistema operativo.

Las secciones 7.4.1 y 7.4.2 presentan las bases de datos empleadas en la validación empírica del método IDCV-EVD así como los resultados obtenidos al incorporar nuevas muestras y nuevas clases al conjunto de entrenamiento, respectivamente.

El esquema de la figura 7.3 muestra un resumen de los experimentos realizados para validar el método IDCV-EVD con respecto al método DCV-EVD original. El conjunto de entrenamiento se actualiza a partir de incorporar más de una muestra por clase, una muestra por clase, y nuevas clases.

7.4.1. Aprendizaje incremental con nuevas muestras

La experimentación del método IDCV-EVD para añadir información, ha requerido la consideración de bases de datos con un número de muestras por clase alto y con una variabilidad apreciable, para que el aprendizaje incremental sobre ellas tenga sentido. Se emplean cuatro bases de datos de características diferentes cuyos detalles se resumen en la figura 7.4.

La primera base de datos es de rostros, Cmu-Pie [107], y presenta cambios de pose e iluminación. La segunda, Coil-20 [88], es de objetos y presenta cambios de pose, de 0 a 355 grados con incrementos de 5 grados. Finalmente, las dos últimas bases de datos, Mnist [72] y Usps [58], son de dígitos de 0 a 9 escritos a mano. Para más detalles ver el apéndice D.

Las muestras de cada base de datos se han dividido en tres grupos distintos. El primer grupo es para el entrenamiento inicial que dará lugar a un primer modelo del problema. El segundo es para validar los algoritmos (prueba), midiendo la tasa de acierto. El tercero es para el aprendi-

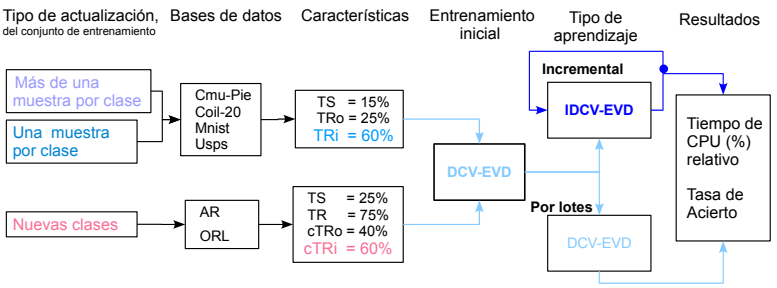


Figura 7.3: Experimentos realizados para validar el método incremental IDCV-EVD respecto al método original DCV-EVD con aprendizaje por lotes. Los casos de estudio son tres. El primero añade más de una muestra por clase al conjunto de entrenamiento. El segundo incorpora una muestra por clase. Aquí, TS, TRo y TRi, son el número de muestras por clase usadas en el test, en el aprendizaje inicial (para el método DCV-EVD), y en el aprendizaje incremental, respectivamente. El último caso incorpora nuevas clases al conjunto de entrenamiento, donde TR es el número de muestras por clase usadas para el entrenamiento. cTRo es el número de clases del entrenamiento inicial, y cTRi es el número de clases que se añaden de forma incremental. Todos los experimentos se realizaron 10 veces.

zaje incremental. Estos grupos están conformados aproximadamente por el 15 %, 25 % y 60 % de las muestras de cada clase, respectivamente.

Para cada base de datos se realizan cinco aprendizajes incrementales, y cada uno de ellos se realiza 10 veces. Las diferencias entre los cinco aprendizajes es el tamaño del conjunto de entrenamiento incremental. Por ejemplo, en Cmu-Pie, Coil-20 y Usps el tamaño del conjunto de entrenamiento incremental para cada uno de los cinco entrenamientos es de $n_j = 1,\ 2,\ 3,\ 4$ y 5 muestras por clase, respectivamente. En Mnist, n_j toma los valores de 1, 4, 7, 10 y 13 muestras por clase. De este modo, múltiples tamaños del conjunto de entrenamiento incremental son empleados, conservando siempre que el tamaño del conjunto de entrenamiento inicial sea mayor al de actualización. En cada una de las 10 iteraciones las muestras de entrenamiento (inicial e incremental) y de prueba se permutan al azar para eliminar dependencias en el orden en que los datos se ponen a disposición del algoritmo.

El entrenamiento inicial se realiza con el algoritmo base DCV-EVD, a partir de m_j muestras por clase. Luego, para adicionar más información

nombre	tamaño imagen	c	tamaño/c	m_j	n_j	tipo
Cmu-Pie	120×160	68	56	[8 . . . 40]	[1,2,3,4,5]	rostros, pose & iluminación
Coil-20	128×128	20	72	[11 . . . 51]	[1,2,3,4,5]	objetos, pose
Mnist	32×32	10	100	[15 . . . 71]	[1,4,7,10,13]	dígitos de 0-9
Usps	16×16	10	25	[7 . . . 22]	[1,2,3,4,5]	dígitos de 0-9

Figura 7.4: Algunas imágenes y detalles de las bases de datos usadas en la validación de los métodos incrementales, para incorporar nuevas muestras al conjunto de entrenamiento. c es el número de clases, tamaño/c es el número de muestras por clase en toda la base de datos. m_j y n_j son el número de muestras por clase para el entrenamiento por lotes e incremental, respectivamente.

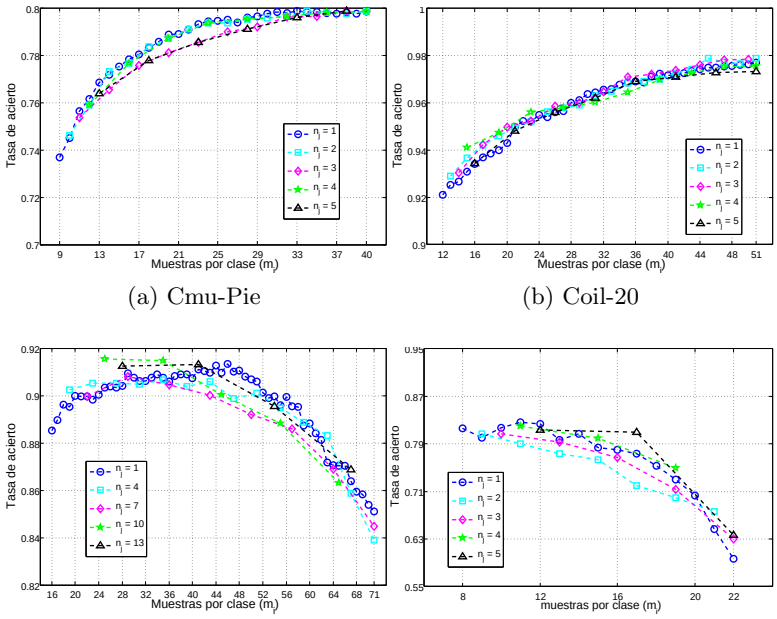
al conjunto de entrenamiento, se incorporan n_j nuevas muestras por cada clase. Los valores de m_j y n_j en los experimentos se muestran en la figura 7.4.

El método original DCV-EVD con el que se compara el algoritmo incremental propuesto, siempre se entrena desde cero con las mismas $(M + N)$ muestras. $M = \sum_{j=1}^{c} m_j$, $N = \sum_{j=1}^{c} n_j$ y $j = 1, \ldots, c$, donde c es el número de clases.

El algoritmo incremental IDCV-EVD se ejecuta por primera vez a partir de los datos obtenidos del entrenamiento inicial con las primeras M muestras. Una vez se realiza el primer entrenamiento con el algoritmo IDCV-EVD, los siguientes entrenamientos incrementales utilizan los datos obtenidos del algoritmo incremental inmediatamente anterior. De esta forma, los valores de m_j están en el rango del 15 % al 75 % de las muestras disponibles en cada base de datos, mientras que el valor de n_j es fijo.

En ambos casos (IDCV-EVD y DCV-EVD) los vectores comunes discriminantes se calculan usando la media de cada clase y la matriz de proyección final, ambas actualizadas.

La figura 7.5 muestra las tasas de acierto del algoritmo incremental cuando se aumenta el número de muestras de entrenamiento, al clasificar las base de datos Cmu-Pie, Coil-20, Mnist y Usps, para diferentes tamaños del conjunto de entrenamiento incremental. Esta figura también muestra implícitamente las tasas de acierto del método con aprendizaje por lotes, ya que ambos presentan exactamente las mismas tasas de acierto. En Cmu-Pie, Coil-20 y Usps los valores empleados de n_j son $[1, 2, 3, 4, 5]$. En Mnist estos valores están más espaciados para poder observar mejor el tiempo de CPU relativo del algoritmo incremental respecto al algoritmo por lotes.



(a) Cmu-Pie (b) Coil-20

(c) Mnist (d) Usps

Figura 7.5: Tasa de acierto en función del conjunto de entrenamiento acumulado de los métodos DCV-EVD e IDCV-EVD, para diferentes valores de n_j.

La variabilidad de la tasa de acierto, que se observa en la figura 7.5 para los diferentes valores de n_j, se debe a la aleatoriedad del conjunto de entrenamiento en cada iteración y no es en absoluto significativa.

Para las bases de datos Cmu-Pie y Coil-20 (figuras 7.5a y 7.5b, respectivamente) la tasa de acierto aumenta a medida que hay nuevas muestras disponibles en el entrenamiento, lo cual era de esperar. En cambio, en la bases de datos Mnist y Usps, figuras 7.5c y 7.5d respectivamente, la tasa de acierto aumenta hasta cierto punto, y después decrece a medida que hay nuevas muestras en el entrenamiento.

Este comportamiento se debe a un problema intrínseco del método DCV, ya que la dimensión del espacio original no es significativamente mayor a la diferencia entre el número de muestras de entrenamiento y el número de clases. Ya que, el método DCV se puede aplicar cuando la dimensión del espacio original es mucho mayor al número de muestras de entrenamiento menos el número de clases [§4.4]. Cuando el tamaño del conjunto de entrenamiento es cercano a la dimensión del espacio original, la dimensión del espacio nulo es pequeña y por tanto este espacio no es lo suficientemente discriminante.

En la figura 7.6 se muestra la relación del número de muestras de entrenamiento total menos el número de clases sobre la dimensión del espacio original, $(M - c)/d$. Esta gráfica está en función del número de muestras de entrenamiento por clase m_j, donde $M = \sum_{j=1}^{c} m_j$. Se observa que, en las bases de datos Usps y Mnist el tamaño de $(M - c)$ en cada iteración está cada vez más cerca a la dimensión del espacio original de las muestras. Mientras que esta relación es pequeña en las bases de datos Cmu-Pie y Coil-20.

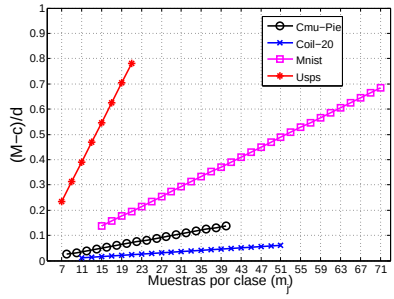


Figura 7.6: Relación de la diferencia entre el tamaño del conjunto de entrenamiento y el número de clases sobre la dimensión del espacio original, $(M - c)/d$, en función del número de muestras de entrenamiento por clase m_j.

La tabla 7.1 muestra la relación entre la dimensionalidad y el valor máximo de la diferencia entre el número de muestras y el número de clases de entrenamiento (en nuestra experimentación), para las bases de datos Cmu-Pie, Coil-20, Mnist y Usps.

Tabla 7.1: Relación entre la dimensión del espacio original d, y el tamaño máximo del conjunto de entrenamiento menos el número de clases $(M - c)$, $M = \sum_{j=1}^{c} m_j$ y $j = 1, \ldots, c$, para las bases de datos Cmu-Pie, Coil-20, Mnist y Usps.

nombre	m_j	c	d		$(M - c)$
Cmu-Pie	40	68	19,200	$\gg$	2,652
Coil-20	51	20	16,384	$\gg$	1,000
Mnist	71	10	1,024	$>$	700
Usps	22	10	256	$>$	210

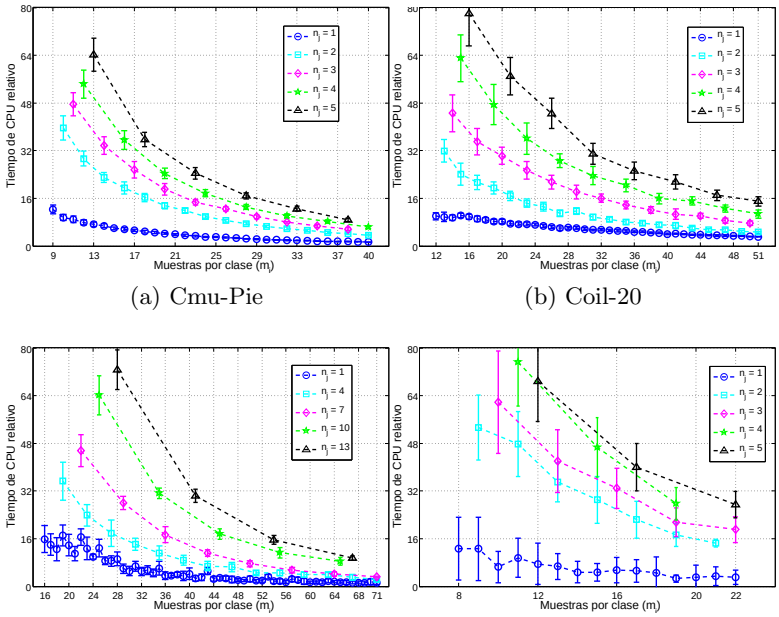
Desde las tasas de acierto podemos decir que, al incorporar nuevas muestras de entrenamiento a partir de las matrices generadas en el aprendizaje incremental anterior (si existe), las matrices de proyección no se degeneran en cada reentrenamiento con respecto al entrenamiento por lotes correspondiente.

En relación al coste computacional, la figura 7.7 presenta el tiempo de CPU relativo[‡], con su respectiva desviación estándar, del algoritmo incremental IDCV-EVD respecto al algoritmo con aprendizaje por lotes DCV-EVD, en cada reentrenamiento. Se observa que el algoritmo incremental es menos costoso que el algoritmo con aprendizaje por lotes. A mayor número de muestras de entrenamiento, más grande es la diferencia entre el coste computacional del algoritmo DCV-EVD e IDCV-EVD. El tiempo de CPU relativo decrece con M para un valor fijo de N. Esta tendencia observada es aproximadamente de $O(1/M)$, lo que se corresponde con los resultados asintóticos para valores fijos de N y d.

A medida que el número de muestras de entrenamiento total es mayor al número de muestras añadidas incrementalmente, las diferencias en el coste computacional son más significativas, hasta alcanzar una reducción en el coste computacional del 98,6 % 96,9 % 98,8 % y 97,41 %, en las bases de datos Cmu-Pie, Coil-20, Mnist y Usps, respectivamente.

Esta reducción en el coste computacional es significativa. Por ejemplo, en los casos extremos donde el número de muestras de entrenamiento es mínimo y máximo, el coste del algoritmo con aprendizaje por lotes, en la base de datos Cmu-Pie, es de 11,75 y 201,73 segundos (3,36 min), respectivamente, mientras que el algoritmo con aprendizaje incremental emplea tan sólo 1,45 y 2,75 segundos, respectivamente.

[‡] El tiempo de CPU relativo es el cociente entre el tiempo del algoritmo incremental y el tiempo del algoritmo por lotes.



(a) Cmu-Pie (b) Coil-20

(c) Mnist (d) Usps

Figura 7.7: Tiempo de CPU relativo del método incremental IDCV-EVD respecto al método original DCV-EVD, al aumentar el número de muestras de entrenamiento para clasificar las bases de datos.

En la base de datos Usps, figura 7.7d, la desviación estándar del tiempo de CPU relativo presenta mayores variaciones debido a que el tiempo computacional del entrenamiento se ve afectado por la suma de pequeñas operaciones, que no son relevantes cuando el tamaño del conjunto de entrenamiento es considerable.

Cuando $M >> N$ varios hechos interesantes se presentan a partir de los resultados obtenidos. En primer lugar, para valores de N no cercanos a M, el algoritmo incremental reduce drásticamente el coste computacional respecto al cómputo necesario para el caso del algoritmo con aprendizaje por lotes. En segundo lugar, el coste computacional del algoritmo incremental es directamente proporcional al tamaño del conjunto incremental. Cuando N es pequeño, $N = 1$, el coste asociado al algoritmo incremental también lo es, y esta disminución compensa incluso teniendo en cuenta que se necesita un número mayor de actualizaciones.

Las tablas E.1, E.2, E.3 y E.4, en el apéndice E, presentan de forma numérica los resultados de la validación del método IDCV-EVD, para las bases de datos Cmu-Pie, Coil-20, Mnist y Usps, respectivamente.

7.4.2. Nuevas clases

Para validar el método incremental IDCV-EVD, cuando nuevas clases se incorporan al conjunto de entrenamiento, se han considerado bases de datos con un número alto de clases para poder tener una adición incremental de éstas con suficiente recorrido. Las bases de datos utilizadas son AR [83] y ORL [94], y contienen imágenes de rostros que presentan cambios de expresión e iluminación. Algunas características de las bases de datos, y algunos parámetros del entrenamiento incremental y por lotes, se muestran en la figura 7.8. c es el número de clases total, tamaño/c es el número de muestras por clase. TR y TS son el número de muestras por clase en el entrenamiento y en el test, respectivamente. El número de clases en el entrenamiento por lotes es c_x, y en el entrenamiento incremental es c_y. También se muestra el tipo de variabilidad presente en las base de datos.

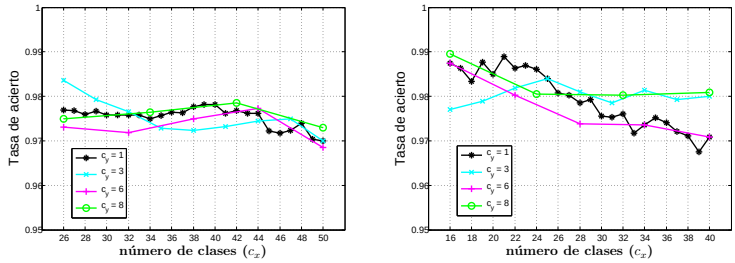
nombre	**tamaño imagen**	c	**tamaño**/c	**TR**	**TS**	c_x	c_y	**tipo**
AR	222×299	50	14	10	4	[26, 50]	[1, 3, 6, 8]	expresión, iluminación & anteojos
ORL	90×108	40	10	7	3	[16, 40]	[1, 3, 6, 8]	expresión, iluminación & anteojos

Figura 7.8: Características de las bases de datos utilizadas en la experimentación de los métodos incrementales para incorporar nuevas clases la conjunto de entrenamiento. **TR** y **TS** son el número de muestras, por clase, empleadas en el entrenamiento y en el test, respectivamente. c_x es el número de clases usas por el entrenamiento con aprendizaje por lotes. c_y es el número de clases añadidas de forma incremental en cada iteración al conjunto de entrenamiento.

En cada base de datos, aproximadamente el 75 % de las muestras se usa para entrenar el sistema, y el 25 % restante es de prueba para medir la tasa de acierto. El 40 % de las clases disponibles es para el entrenamiento inicial, y el 60 % restante es para actualizar el conjunto de entrenamiento.

Para poder observar el comportamiento del método incremental respecto a la tasa de clasificación y al tiempo de CPU, múltiples tamaños del conjunto de actualización, en el rango de 1 a 8, se emplean en la validación. Aquí se muestran y analizan los resultados obtenidos para $c_y = [1, 3, 6, 8]$, en ambas bases de datos.

En la figura 7.9 se muestra la tasa de acierto del método DCV-EVD en función del número de clases en el conjunto de entrenamiento, y para diferentes tamaños del conjunto incremental. La tasa de acierto de algoritmo incremental IDCV-EVD es igual a la del algoritmo con aprendizaje por lotes, y se pueden hacer las mismas consideraciones que en los experimentos de la sección anterior. Los pequeños cambios en la tasa de acierto respecto a los diferentes valores de c_y ocurren porque la distribución del conjunto de entrenamiento es aleatoria para cada c_y.



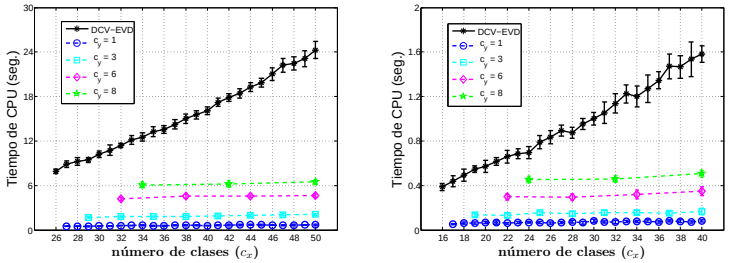
(a) AR (b) ORL

Figura 7.9: Tasa de acierto en función del número de clases en el conjunto de entrenamiento para el método DCV, e implícitamente la del método IDCV, al clasificar las bases de datos.

Al aumentar el número de clases en el conjunto de entrenamiento se observa que la tasa de acierto no sufre diferencias relevantes. Sin embargo, esto no sucede en algunos métodos incrementales que al añadir nuevas clases en el conjunto de entrenamiento la tasa de acierto va en descenso como se muestra en [130, 129].

El tiempo de CPU en segundos del proceso de entrenamiento, en los métodos DCV-EVD e IDCV-EVD, se presenta en la figura 7.10. En esta figura se observa que el tiempo de CPU del método con aprendizaje por

lotes e incremental tiene un comportamiento lineal. El tiempo de CPU del método incremental es casi constante para un mismo valor de c_y, mientras que el tiempo del método con aprendizaje por lotes aumenta en función del número de clases en el conjunto de entrenamiento.



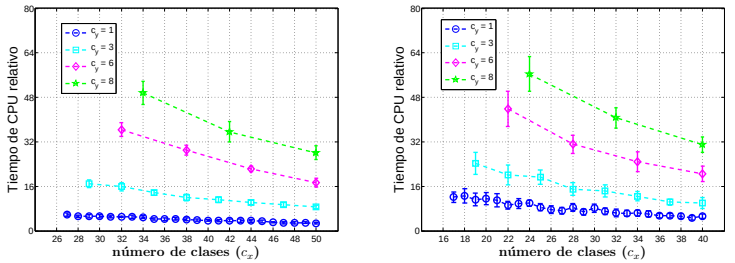
(a) AR (b) ORL

Figura 7.10: Tiempo de CPU en segundos de la fase de entrenamiento en los métodos DCV-EVD e IDCV-EVD, al incrementar el número de clases en el conjunto entrenamiento para clasificar las bases de datos.

El comportamiento lineal del tiempo de CPU en el método IDCV-EVD se mantiene aunque el número de clases incorporadas de forma incremental sea elevado, y lo único que cambia es el valor constante del tiempo para cada c_y. Este valor constante depende en parte del número de muestras en el entrenamiento incremental, y sobre todo de la dimensión del espacio original.

Para finalizar, el tiempo de CPU relativo del método IDCV-EVD con respecto al método DCV-EVD se muestra en la figura 7.11, donde se observa claramente que a menor número de muestras de entrenamiento en el aprendizaje incremental, mayor es la diferencia entre el tiempo computacional del algoritmo incremental y el algoritmo por lotes. Para valores no tan pequeños de c_y, el ahorro en el coste computacional sigue siendo apreciable.

En este caso la tendencia de $O(1/M)$ en el tiempo de CPU relativo es menos acusada, ya que en ambas bases de datos la dimensión del espacio original es elevada. Si nuevas clases se incorporaran a la base de conocimiento, o si la dimensión del espacio original fuera menor, la tendencia de $(1/M)$ seria más visible, como se ha visto en la sección 7.4.1.



(a) AR (b) ORL

Figura 7.11: Tiempo de CPU relativo del método IDCV-EVD respecto al método original DCV-EVD, al incrementar el número de clases en el conjunto entrenamiento para clasificar las bases de datos.

7.5. Discusión

A partir de la validación empírica del método IDCV-EVD se observa que, tanto en el caso de nuevas muestras para las clases existentes como en el caso de nuevas clases, el algoritmo incremental no presenta desventajas respecto a la tasa de acierto en cuanto al algoritmo con aprendizaje por lotes. Por lo tanto, aunque múltiples iteraciones se realicen las aproximaciones realizadas en el algoritmo incremental no afectan el poder discriminante del método original.

También se observa que en cada reentrenamiento el algoritmo incremental necesita menor tiempo computacional que el algoritmo con aprendizaje por lotes. Además, cuanto más grande es la diferencia entre el conjunto de entrenamiento precalculado (inicial) y el tamaño del conjunto de entrenamiento incremental, la diferencia entre los costes computacionales es mucho mayor. Cuando el tamaño del conjunto incremental es pequeño, $N = 1$ o $c_y = 1$, el coste asociado al algoritmo incremental también lo es, y esta disminución compensa incluso teniendo en cuenta que se necesita un número mayor de actualizaciones.

Si bien el tamaño del conjunto de entrenamiento afecta el coste computacional de los algoritmos, otro factor importante es la dimensión del espacio de entrada. Si la dimensión es considerablemente mayor al tamaño del conjunto de entrenamiento se observa que la tendencia del tiempo de CPU relativo del algoritmo incremental decrece con M para un valor fijo de N. Esta tendencia observada coincide con la teóricamente esperada y es aproximadamente de $O(1/M)$.

Capítulo 8

Vectores comunes discriminantes incrementales mediante ortogonalización

Resumen — Este capítulo presenta dos versiones del método IDCV que usan subespacios diferencia y ortogonalización. La primera de ellas es una extensión incremental del método original DCV-GSO. La segunda aproximación es un método propio que combina subespacios diferencias y ortogonalización.

Contenido

8.1. IDCV mediante ortogonalización de Gram-Schmidt incremental 85
8.2. IDCV mediante subespacios diferencia y ortogonalización 94
8.3. Discusión 99

8.1. IDCV mediante ortogonalización de Gram-Schmidt incremental

Igual que en el capítulo anterior, se estudian tres formas diferentes de incorporar la información al conjunto de entrenamiento, pero ahora a partir de una versión incremental del método DCV-GSO. El algoritmo del

método IDCV-GSO extiende el subespacio diferencia de las muestras de entrenamiento inicial con la nueva información del conjunto incremental.

8.1.1. IDCV-GSO para nuevas muestras de entrenamiento

El subespacio diferencia del conjunto de datos $\mathbf{Z}$ está dado por

$$\mathcal{B}_Z = span\{\mathrm{z}_1^2 - \mathrm{z}_1^1,\, \mathrm{z}_1^3 - \mathrm{z}_1^1, \ldots,\, \mathrm{z}_2^2 - \mathrm{z}_2^1,\, \mathrm{z}_2^3 - \mathrm{z}_2^1, \ldots,\, \mathrm{z}_c^{(m_c+n_c)} - \mathrm{z}_c^1\},$$

donde el primer vector de cada clase, z_j^1, se elige como vector sustraendo (si bien, la elección de éste es arbitraria). De la misma manera, los subespacios diferencia de $\mathbf{X}$ e $\mathbf{Y}$, se definen como

$$\begin{aligned} \mathcal{B}_X &= span\{\mathrm{x}_1^2 - \mathrm{x}_1^1,\, \mathrm{x}_1^3 - \mathrm{x}_1^1, \ldots,\, \mathrm{x}_2^2 - \mathrm{x}_2^1,\, \mathrm{x}_2^3 - \mathrm{x}_2^1, \ldots,\, \mathrm{x}_c^{m_c} - \mathrm{x}_c^1\} \; y \\ \mathcal{B}_Y &= span\{\mathrm{y}_1^2 - \mathrm{y}_1^1,\, \mathrm{y}_1^3 - \mathrm{y}_1^1, \ldots,\, \mathrm{y}_2^2 - \mathrm{y}_2^1,\, \mathrm{y}_2^3 - \mathrm{y}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{y}_c^1\}, \end{aligned}$$

respectivamente.

En el caso incremental donde las muestras presentes en $\mathbf{X}$ y en $\mathbf{Y}$ pertenecen a las mismas clases, los vectores diferencia de $\mathbf{Y}$ se calculan respecto a $\mathbf{X}$. Aquí se han seleccionado como vectores sustraendo los mismos vectores x_j^1 empleados en el entrenamiento anterior, donde se obtuvo el subespacio diferencia de $\mathbf{X}$. Pero el vector sustraendo en la clase j puede ser cualquier muestra x_j^i, como se muestra en el teorema 8. Por lo tanto, definimos el subespacio diferencia de $\mathbf{Y}$ respecto a los vectores sustraendo x_j^1 como

$$\mathcal{B}_{Y_x} = span\{\mathrm{y}_1^1 - \mathrm{x}_1^1,\, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1,\, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1\} \quad (8.1)$$

Nótese que la cardinalidad del conjunto generador de $\mathcal{B}_{Y_x}$ es mayor en c unidades que el de $\mathcal{B}_Y$ y por lo tanto su dimensionalidad es también potencialmente mayor.

De este modo, el subespacio diferencia del conjunto de entrenamiento total, $\mathbf{Z}$, se puede escribir como la suma del subespacio diferencia de $\mathbf{X}$ y el subespacio diferencia $\mathcal{B}_{Y_x}$.

$$\mathcal{B}_Z = \mathcal{B}_X + \mathcal{B}_{Y_x}, \quad (8.2)$$

Si las muestras son linealmente independientes la suma de los subespacios es una suma directa, $\mathcal{B}_Z = \mathcal{B}_X \oplus \mathcal{B}_{Y_x}$, como se muestra en el teorema 3 del apéndice C.

La finalidad de expresar el subespacio $\mathcal{B}_Z$ en términos de $\mathcal{B}_X$ y $\mathcal{B}_{Y_x}$, es poder encontrar una base ortonormal, Θ', de $\mathcal{B}_Z$ en función de una base ortonormal $\Theta \in \mathbb{R}^{d \times r_x}$ de $\mathcal{B}_X$ (calculada en el entrenamiento anterior, donde r_x es el rango de $\mathcal{B}_X$) y una base O de $\mathcal{B}_{Y_x}$.

A partir de las propiedades de suma de subespacios, una base ortonormal Θ' del subespacio diferencia $\mathcal{B}_Z$ se puede obtener desde la base ortonormal Θ ampliada con una base ortonormal de $\mathcal{B}_{Y_x}$ (que a su vez sea ortogonal a Θ). La figura 8.1 muestra intuitivamente los subespacios diferencia e indiferencia de un conjunto de entrenamiento $\mathbf{Z}$, donde $\mathbf{Z}$ está formado por $\mathbf{X}$ e $\mathbf{Y}$.

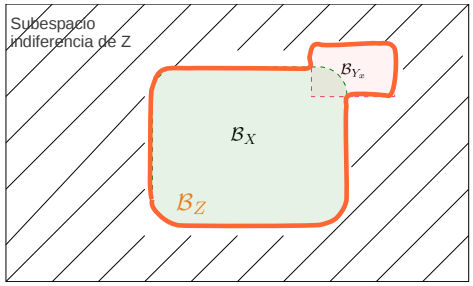


Figura 8.1: Subespacio diferencia ($\mathcal{B}_Z$) del conjunto de entrenamiento $\mathbf{Z}$, formado por la suma de los subespacios diferencia de $\mathbf{X}$ y de $\mathbf{Y}$ respecto a los mismos vectores sustraendo x_j^1.

La base ortonormal O, de $\mathcal{B}_{Y_x}$, se consigue mediante una ortogonalización de Gram-Schmidt a partir de la base ortonormal Θ. La ortogonalización de Gram-Schmidt se puede ver como un algoritmo incremental, ya que el proceso de ortogonalización puede incorporar nuevas muestras al conjunto de entrenamiento a partir de una base ortonormal precalculada sin empezar desde cero. De este modo se comprueba la dependencia lineal de las nuevas muestras de entrenamiento respecto a las muestras iniciales, garantizando que todos los vectores que conforman $\Theta' = [\Theta\ \mathrm{O}]$ son linealmente independientes.

El algoritmo 4 muestra de forma general la ortogonalización de Gram-Schmidt incremental, donde $Q = [q_1, q_2, \ldots, q_r] \in \mathbb{R}^{d\times r}$ es la base ortonormal precalculada (inicial), $B = [b_1, b_2, \ldots, b_N] \in \mathbb{R}^{d\times N}$ es una matriz de vectores columna que representa la nueva información y, $Q' \in \mathbb{R}^{d\times l}$ es una base ortonormal del subespacio generado por los vectores de Q y B.

Desde la ortogonalización de Gram-Schmidt incremental nuevos vectores son añadidos a Θ obteniendo una base ortonormal del subespacio diferencia $\mathcal{B}_Z$, y en consecuencia el operador de proyección $\Theta'\Theta'^T$ a este subespacio. De este modo, los z_{cv}^j se obtienen a partir de cualquier muestra z_j^i de la clase j y de la base incremental $\Theta' = [\Theta\ \mathrm{O}]$.

$$\mathrm{z}_{cv}^j = \mathrm{z}_j^i - \Theta'\Theta'^T\mathrm{z}_j^i = (\mathrm{I} - \Theta\Theta^T - \mathrm{O}\mathrm{O}^T)\mathrm{z}_j^i$$

o de forma equivalente como $\mathrm{z}_{cv}^j = \mathrm{x}_{cv}^j - \mathrm{O}\mathrm{O}^T\mathrm{x}_j^i$.

Algoritmo 4 *Ortogonalización de Gram-Schmidt incremental.*

Entrada: $Q \in \mathbb{R}^{d \times r}$, $B \in \mathbb{R}^{d \times N}$.
Salida: $Q' \in \mathbb{R}^{d \times l}$.

1. Inicializar $l = r$.
2. Desde $i \leftarrow 1$ hasta N

   a) $v = b_i - \sum_{k=1}^{l} (q_k^T \, b_i) \; q_k$

   b) si $\|v\| > 0$ entonces

   $l = l + 1$.

   $q_l = v / \|v\|$.
3. La base ortonormal calculada de forma incremental es $Q' = [q_1, \, q_2, \ldots, q_r, \; q_{(r+1)}, \, q_{(r+2)}, \ldots, q_l]$.
   En el siguiente entrenamiento: $Q = Q'$ y $r = l$.

Las principales operaciones para conseguir de forma incremental una base ortonormal del subespacio diferencia de $\mathbf{Z}$, los vectores comunes discriminantes y la matriz de proyección final, para incorporar una o más de una muestra por clase al conjunto de entrenamiento se muestran en el algoritmo 5. Los parámetros de entrada del algoritmo son la base ortonormal del subespacio $\mathcal{B}_X$, el nuevo conjunto de entrenamiento, los vectores sustraendo empleados en la generación de $\mathcal{B}_X$, y los vectores comunes x_{cv}^j.

Algoritmo 5 *Método IDCV-GSO que incorpora nuevas muestras al conjunto de entrenamiento* $\mathbf{X}$.

Entrada: $\Theta \in \mathbb{R}^{d \times r_x}$, $\mathbf{Y} \in \mathbb{R}^{d \times N}$, x_j^1, x_{cv}^j.
Salida: W_{dcv}, z_{dcv}^j, Θ'.

1. Obtener el subespacio diferencia de $\mathbf{Y}$ respecto a $\mathbf{X}$ $\mathcal{B}_{Y_x} = span\{\mathrm{y}_1^1 - \mathrm{x}_1^1, \, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1, \, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1\}$.
2. Calcular una base ortonormal O de $\mathcal{B}_{Y_x}$, a partir de GSO incremental desde la base ortonormal Θ de $\mathcal{B}_X$.
3. Obtener Θ' como $\Theta' = [\Theta \; \mathrm{O}]$.
4. Calcular los nuevos vectores comunes, $\mathrm{z}_{cv}^j = \mathrm{x}_{cv}^j - \mathrm{O}\mathrm{O}^T \mathrm{x}_j^i$.
5. Computar el subespacio diferencia de los nuevos vectores comunes, $\mathcal{B}_{cv} = span\{\mathrm{z}_{cv}^2 - \mathrm{z}_{cv}^1 \; \ldots \; \mathrm{z}_{cv}^c - \mathrm{z}_{cv}^1\}$.
6. Obtener una base ortonormal W_{dcv} de $\mathcal{B}_{cv}$. Esta base es la matriz de proyección final.
7. Los vectores comunes discriminantes son $\mathrm{z}_{dcv}^j = \mathrm{W}_{dcv}^T \mathrm{z}_j^i$.
   En el siguiente entrenamiento: $\Theta = \Theta'$, y $\mathrm{x}_{cv}^j = \mathrm{z}_{cv}^j$.

El planteamiento del método IDCV-GSO para añadir al conjunto de entrenamiento una muestra por clase no es diferente al propuesto para incorporar más de una muestra por clase, ya que el subespacio diferencia de $\mathbf{Y}$ respecto al subespacio diferencia de $\mathbf{X}$ puede ser calculado, sin modificar el algoritmo.

Complejidad del método

La complejidad del método DCV-GSO, para encontrar una base ortonormal del subespacio diferencia de $\mathbf{Z}$ (y al subespacio rango de S_Z^w), viene dada por $O(d(M+N-c)^2)$ operaciones.

El coste computacional del algoritmo IDCV-GSO está dominado por la GSO incremental, que logra reducir el coste a tan sólo $O(dNr_x)$ operaciones, donde r_x es el rango del subespacio $\mathcal{B}_X$. Si el conjunto de entrenamiento $\mathbf{X}$ es linealmente independiente, entonces $r_x = (M-c)$.

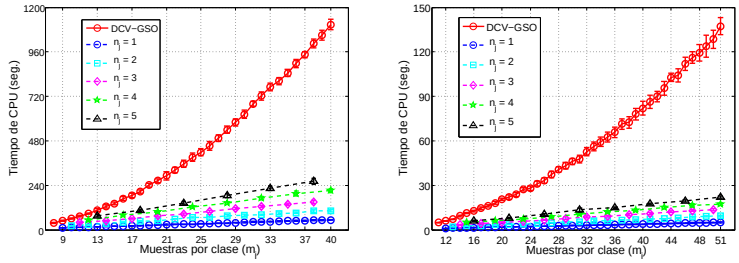
Validación empírica

La validación del método IDCV-GSO respecto al método original DCV-GSO utiliza las bases de datos Cmu-Pie y Coil-20, y la misma metodología de la sección 7.4 (que valida el método IDCV-EVD). El método IDCV-GSO también se ha validado con las bases de datos Mnist y Usps (de la sección 7.4), pero como ya se ha visto estas bases de datos están muy cerca del límite entre la dimensionalidad y la diferencia entre el número de muestras en el conjunto de entrenamiento y el número de clases. Por lo tanto, los resultados obtenidos para Mnist y Usps no son representativos, estos resultados se muestran en las tablas E.9 y E.10.

La tasa de acierto del algoritmo incremental propuesto, IDCV-GSO, es igual a la tasa de acierto del método original DCV-GSO, e igual a la del método DCV-EVD. Por esto, los resultados obtenidos respecto a la tasa de acierto no se muestran en esta sección, ya que se analizaron en la sección 7.4.1 para los métodos basados en EVD.

En cuanto al coste de ejecutar los algoritmos basados en subespacios diferencia y ortogonalización, la figura 8.2 muestra el tiempo de CPU en segundos de los métodos DCV-GSO y IDCV-GSO, para diferentes tamaños del conjunto de entrenamiento inicial e incremental. Se observa que el coste del algoritmo con aprendizaje por lotes presenta una tendencia cuadrática, mientras que el coste del método incremental IDCV-GSO presenta un comportamiento lineal.

En los experimentos, el único valor que cambia es el número de muestras incorporadas de forma incremental. Parámetros como la dimensiona-

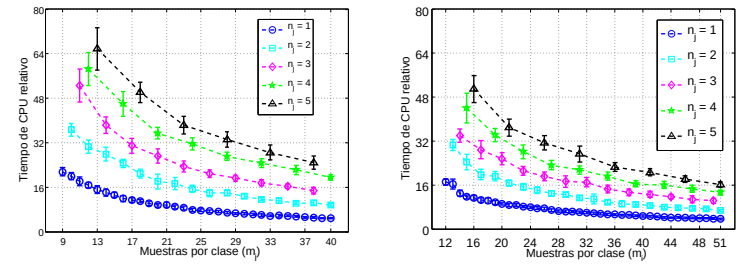


(a) Cmu-Pie (b) Coil-20

Figura 8.2: Tiempo de CPU en segundos que emplea el método DCV-GSO y IDCV-GSO, para diferentes tamaños del conjunto de entrenamiento al clasificar las bases de datos.

lidad y el número muestras de entrenamiento inicial son constantes. Por lo tanto, podemos decir que el coste del algoritmo incremental está dominado por la dimensión del espacio original, y en menor medida por el tamaño del conjunto de entrenamiento, puesto que las diferencias en el tiempo de cómputo para diferentes valores de n_j son lineales. Lo cual confirma el coste teórico de $O(dNr_x)$ operaciones, donde N es el tamaño del conjunto de actualización, r_x es la dimensión del espacio rango del conjunto de entrenamiento anterior y d es la dimensión del espacio original que es mucho mayor que N y r_x.

La figura 8.3 presenta el tiempo de CPU relativo del método incremental IDCV-GSO respecto al método original DCV-GSO.



(a) Cmu-Pie (b) Coil-20

Figura 8.3: Tiempo de CPU relativo del método incremental IDCV-GSO respecto al método original DCV-GSO, al aumentar el número de muestras de entrenamiento.

En la figura 8.3 se observa que, a mayor número de muestras de entrenamiento mayor es la diferencia entre el coste del algoritmo incremental respecto al coste del algoritmo con aprendizaje por lotes. El decremento en el coste computacional es proporcional a la diferencia entre el número de muestras en el conjunto precalculado y el número de muestras en el conjunto incremental.

Los resultados numéricos de validar el método incremental IDCV-GSO respecto nuevas muestras en el conjunto de entrenamiento se muestran en el apéndice E en las tablas E.7 y E.8, para las bases de datos Cmu-Pie y Coil, respectivamente.

8.1.2. IDCV-GSO que incorpora nuevas clases al conjunto de entrenamiento

Cuando el nuevo conjunto de entrenamiento, $\mathbf{Y}$, proporciona información respecto a nuevas clases su subespacio diferencia se define respecto a si mismo, y no respecto a $\mathbf{X}$ como en el caso anterior. El subespacio diferencia de $\mathbf{Y}$ tiene las mismas propiedades matemáticas del subespacio diferencia de $\mathbf{X}$. Y dado que, el vector común de una clase es independiente de la selección del vector sustraendo (ver detalles en el teorema 5), entonces cualquier muestra de la clase j en $\mathbf{Y}$ puede ser el vector sustraendo.

En este caso, el vector sustraendo que se utiliza para calcular el subespacio diferencia de $\mathbf{Y}$ es la primera muestra de cada clase presente en $\mathbf{Y}$.

$$\mathcal{B}_Y = span\{\mathrm{y}_1^2 - \mathrm{y}_1^1, \mathrm{y}_1^3 - \mathrm{y}_1^1, \ldots, \mathrm{y}_2^2 - \mathrm{y}_2^1, \mathrm{y}_2^3 - \mathrm{y}_2^1, \ldots, \mathrm{y}_{c_y}^{n_{c_y}} - \mathrm{y}_{c_y}^1\} \quad (8.3)$$

c_y es el número de clases en $\mathbf{Y}$.

Una vez se tiene el subespacio diferencia de $\mathbf{Y}$ respecto a si mismo, la base ortonormal Θ' del subespacio diferencia $\mathcal{B}_Z = \mathcal{B}_X + \mathcal{B}_Y$ se obtiene del mismo modo que en el caso anterior.

Por lo tanto, la base ortonormal O de $\mathcal{B}_Y$ se calcula a partir de la GSO incremental desde la base ortonormal Θ, obteniendo $\Theta' = [\Theta\ \mathrm{O}]$.

El algoritmo del método IDCV-GSO que permite incorporar nuevas clases al conjunto de entrenamiento difiere del algoritmo 5 en el punto 1, que se remplaza por:

1. Obtener el subespacio diferencia del conjunto de entrenamiento $\mathbf{Y}$ respecto a si mismo, como

   $$\mathcal{B}_Y = span\{y_1^2 - y_1^1,\ y_1^3 - y_1^1, \ldots,\ y_2^2 - y_2^1,\ y_2^3 - y_2^1, \ldots,\ y_c^{n_c} - y_c^1\}$$

   donde el vector sustraendo en $\mathbf{Z}$ es igual a $z_j^1 = x_j^1$ para $j = 1, \ldots, c_x$, y $z_j^1 = y_j^1$ para $j = c_x + 1,\ \ldots, (c_x + c_y)$.
   En el siguiente entrenamiento $x_j^1 = z_j^1$.

Complejidad del método

Al aumentar las clases presentes en el conjunto de entrenamiento $\mathbf{X}$ a partir del método IDCV-GSO, el número de operaciones involucradas en el algoritmo incremental es de $O(d(N - c_y)r_x)$. Donde r_x es como mucho igual a $(M - c_x)$, si todas las muestras de $\mathbf{X}$ son linealmente independientes.

Respecto al método DCV-GSO, el número de operaciones involucradas para calcular el operador de proyección $\Theta'\Theta'^T$ cuando las clases presentes en $\mathbf{X}$ son diferentes de las que hay en $\mathbf{Y}$, es de $O(d(M - c_x + N - c_y)^2)$. Es evidente que $(M - c_x + N - c_y)^2 >> (N - c_y)r_x$.

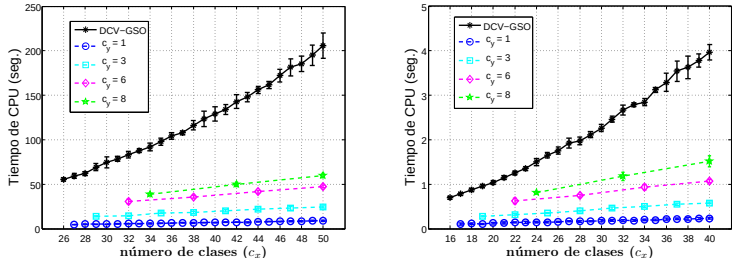
Validación empírica

El método IDCV-GSO para incorporar nuevas clases al conjunto de entrenamiento se valida a partir de la metodología descrita en la sección 7.4.2, y a partir de clasificar dos bases de datos de imágenes de rostros. Las características principales del proceso de entrenamiento y de las bases de datos se muestran en la figura 7.8.

Los métodos basados en GSO son idénticos, en cuanto a la tasa de acierto, a los métodos basados en EVD. Por esto, en este apartado no se presentan los resultados de la tasa de acierto en función del número de clases, ya que estos resultados son básicamente los mismo de la figura 7.9 en la sección 7.4.2.

El tiempo de CPU en segundos de los métodos DCV-GSO e IDCV-GSO, en la fase de entrenamiento, cuando la información de nuevas clases se añade a la base de conocimiento se muestra en la figura 8.4. Se observa que el tiempo de CPU del método original tiene un comportamiento cuadrático, como en el caso de nuevas muestras. En cuanto al método incremental, la tendencia del tiempo de CPU es lineal y está en función del número de clases que se incorporan de forma incremental a la base

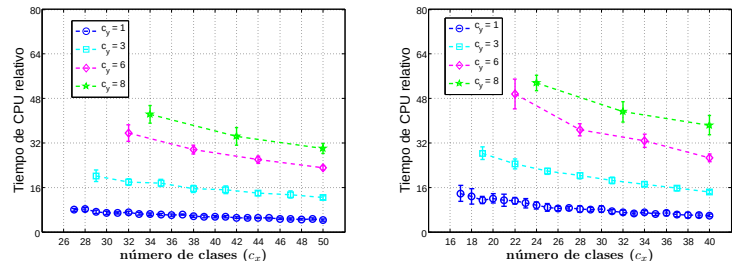
de conocimiento, e implícitamente del tamaño total del conjunto de actualización. También se observa que para un número pequeño de nuevas clases el tiempo de CPU, necesario en la fase de entrenamiento, es ínfimo respecto al de añadir un número más elevado de clases.



(a) AR (b) ORL

Figura 8.4: Tiempo de CPU en segundos de los métodos DCV-GSO e IDCV-GSO, en la fase de entrenamiento, al incrementar el número de clases.

La figura 8.5 muestra el tiempo de CPU relativo del método IDCV-GSO respecto al del método DCV-GSO, cuando nuevas clases se incorporan al conjunto de entrenamiento, en las bases de datos AR y ORL.



(a) AR (b) ORL

Figura 8.5: Tiempo de CPU relativo del método IDCV-GSO respecto al método original DCV-GSO, al incrementar el número de clases en el conjunto de entrenamiento.

Se puede concluir que actualizar la base de conocimiento con nuevas clases a partir del algoritmo incremental IDCV-GSO, conlleva reducir el coste computacional de la fase de entrenamiento como mínimo un 53 % y 57 %, para las bases de datos AR y ORL, respectivamente.

A medida que la diferencia entre el tamaño del conjunto de entrenamiento inicial y el de actualización es mayor, el algoritmo incremental tiene un coste computacional significativamente menor con respecto al del algoritmo con aprendizaje por lotes. El algoritmo incremental llega a costar tan solo un 8 % respecto al 100 % del algoritmo con aprendizaje por lotes.

Cuando una clase se incorpora de forma incremental a la base de conocimiento se observa que, el coste computacional del algoritmo incremental es mucho menor respecto al algoritmo con aprendizaje por lotes y respecto al algoritmo incremental para un número más alto de clases. Aunque más entrenamientos son necesarios cuando $c_y = 1$, la reducción en el coste computacional es considerable. Por ejemplo, en la base de datos AR, para valores de c_y iguales a 3, 6 y 8 hay como mínimo una reducción en el coste computacional del 8 %, 19 % y 26 %, respectivamente y del 92 % respecto al método DCV-GSO. En la base de datos ORL cuando $c_y = 1$ hay como mínimo una reducción del 8 %, 20 %, 32 % y 86 % para valores de c_y iguales a 3, 6, 8, y al método DCV-GSO, respectivamente.

8.2. IDCV mediante subespacios diferencia y ortogonalización

El método IDCV planteado mediante subespacios diferencia y ortogonalización, IDCV-O, se puede ver como una simplificación de los métodos tratados en las secciones 7.2 y 8.1. Aquí el nuevo operador de proyección para calcular los vectores comunes z_{cv}^j, se compone de la base ortonormal del subespacio empleado en el entrenamiento anterior, que podemos llamar U independientemente de que haya sido calculado por un método u otro (desde $\mathcal{R}(S_X^w)$ o $\mathcal{B}_X$), y de un nuevo conjunto de vectores (columna) ortonormales que llamamos v.

El nuevo conjunto de vectores que forman v, además de ser ortonormales entre si, tienen que ser ortogonales a la base U calculada en el entrenamiento previo.

Como en la sección anterior, el subespacio diferencia del conjunto de entrenamiento $\mathbf{Y}$ (que contiene una o más de una muestra por clase) se

define respecto al conjunto de entrenamiento $\mathbf{X}$, como:

$$\mathcal{B}_{Y_x} = span\{\mathrm{y}_1^1 - \mathrm{x}_1^1,\, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1,\, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1\}$$

En este método los vectores que generan el subespacio $\mathcal{B}_{Y_x}$ se proyectan en el subespacio complementario ortogonal que genera U, consiguiendo de esta forma que los vectores proyectados sean ortogonales a U.

$$\begin{aligned}\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T\,[\mathrm{y}_1^1 - \mathrm{x}_1^1,\, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1,\, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1] = \\ (\mathrm{I} - \mathrm{U}\mathrm{U}^T)\,[\mathrm{y}_1^1 - \mathrm{x}_1^1,\, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1,\, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1]\end{aligned} \tag{8.4}$$

El resultado de la ecuación (8.4) es un conjunto de vectores ortogonales a U, aunque estos vectores no son necesariamente ortogonales entre ellos. Por este motivo es necesario ortonormalizarlos, y para ello se aplica algún algoritmo de ortonormalización como es la descomposición QR, la transformación de Householder, la rotación de Givens o el método de ortogonalización de Gram-Schmidt, entre otros. Consiguiendo a la vez que se eliminen dependencias lineales entre el nuevo conjunto de entrenamiento y el inicial.

De ese modo, v se obtiene como:

$$v = orth\left((\mathrm{I} - \mathrm{U}\mathrm{U}^T)\,[\mathrm{y}_1^1 - \mathrm{x}_1^1,\, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1,\, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1]\right) \tag{8.5}$$

Los nuevos vectores comunes del conjunto de entrenamiento $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}]$, están dados, como en el capítulo 7, por:

$$\mathrm{z}_{cv}^j = (\mathrm{I} - [\mathrm{U}\ v][\mathrm{U}\ v]^T)\,\mathrm{z}_j^i = \mathrm{x}_{cv}^j - vv^T\,\mathrm{x}_j^i$$

El operador de proyección $[\mathrm{U}\ v][\mathrm{U}\ v]^T$, independientemente de cómo se calcule U, genera al mismo subespacio $\mathcal{R}(S_Z^w)$ (o $\mathcal{B}_Z$).

Si el conjunto de entrenamiento $\mathbf{Y}$ contiene c_y clases diferentes de las que tiene $\mathbf{X}$, el subespacio diferencia de $\mathbf{Y}$ se calcula respecto a si mismo como

$$\mathcal{B}_Y = span\{\mathrm{y}_1^2 - \mathrm{y}_1^1,\, \mathrm{y}_1^3 - \mathrm{y}_1^1, \ldots, \mathrm{y}_2^2 - \mathrm{y}_2^1,\, \mathrm{y}_2^3 - \mathrm{y}_2^1, \ldots,\, \mathrm{y}_{c_y}^{n_{c_y}} - \mathrm{y}_{c_y}^1\},$$

y el método propuesto se aplica de la misma forma que se ha explicado antes. Pero ahora el número de clases en el conjunto de entrenamiento $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}]$ es $(c_x + c_y)$, donde $j = 1, \ldots, c_x, c_x + 1, \ldots, c_x + c_y$.

El algoritmo 6 resume los pasos para aplicar el método IDCV mediante subespacios diferencia y ortogonalización.

Algoritmo 6 *Método IDCV mediante subespacios diferencia y ortogonalización.*

Entrada: $\mathrm{U} \in \mathbb{R}^{d \times r_x}$, $\mathbf{Y} \in \mathbb{R}^{d \times N}$, x_j^1, x_{cv}^j.
Salida: W_{dcv}, z_{dcv}^j, $[\mathrm{U}\ v]$.

1. Si $\mathbf{Y}$ tiene las mismas clases de $\mathbf{X}$, calcular el subespacio diferencia de $\mathbf{Y}$ respecto a $\mathbf{X}$.

$$\mathcal{B}_{Y_x} = span\{\mathrm{y}_1^1 - \mathrm{x}_1^1, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1\}$$

2. Si las clases de $\mathbf{Y}$ son diferentes a las clases de $\mathbf{X}$, se calcula el subespacio diferencia de $\mathbf{Y}$ respecto a si mismo.

$$\mathcal{B}_Y = span\{\mathrm{y}_1^2 - \mathrm{y}_1^1, \mathrm{y}_1^3 - \mathrm{y}_1^1, \ldots, \mathrm{y}_2^2 - \mathrm{y}_2^1, \mathrm{y}_2^3 - \mathrm{y}_2^1, \ldots, \mathrm{y}_{c_y}^{n_{c_y}} - \mathrm{y}_{c_y}^1\}$$

3. Se proyectan los vectores que generan $\mathcal{B}_{Y_x}/\mathcal{B}_Y$ en el subespacio complementario ortogonal que genera U. Luego, se obtiene una base ortonormal, v, de esta proyección a partir de un método de ortogonalización.
4. Los nuevos vectores comunes son $\mathrm{z}_{cv}^j = \mathrm{x}_{cv}^j - vv^T\,\mathrm{x}_j^i$.
5. Encontrar una base ortonormal, W_{dcv}, al subespacio diferencia de los vectores comunes z_{cv}^j.
6. Finalmente el DCV de la clase j se calcula como $\mathrm{z}_{dcv}^j = \mathrm{W}_{dcv}^T \mathrm{z}_j^i$.
   En el siguiente entrenamiento: $\mathrm{U} = [\mathrm{U}\ v]$, $\mathrm{x}_j^1 = \mathrm{z}_j^1$ y $\mathrm{x}_{cv}^j = \mathrm{z}_{cv}^j$.

8.2.1. Complejidad del método

El coste computacional asociado al método IDCV-O para actualizar la información de las clases en $\mathbf{X}$ es de $O(dNr_x)$ operaciones, donde r_x representa el rango del subespacio $\mathcal{R}(S_X^w)$. Cuando el conjunto de entrenamiento $\mathbf{Y}$ contiene información de c_y nuevas clases, el número de operaciones involucradas es de $O(d(N - c_y)r_x)$.

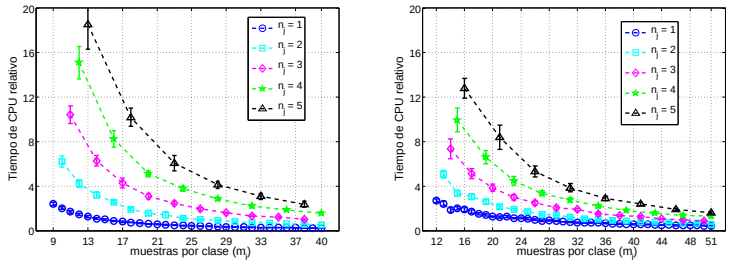
Aunque el método IDCV-O obtiene el operador de proyección al subespacio $\mathcal{R}(S_Z^w)$ (e implícitamente a $\mathcal{N}(S_Z^w)$) de forma diferente a la de IDCV-GSO, estos dos métodos presentan el mismo coste computacional, como se observa en la sección 8.1. El número de operaciones involucradas en el calculo de v es semejante al de realizar la GSO incremental en IDCV-GSO.

8.2.2. Validación IDCV-O

A continuación se valida empíricamente el método incremental IDCV-O respecto al método con aprendizaje por lotes IDCV-GSO. Se ha elegido como método de comparación a DCV-GSO porque los dos métodos están basados en subespacios diferencia, y la ortogonalización en el método IDCV-O se realiza a partir de la ortogonalización de Gram-Schmidt. En general la ortogonalización de Gram-Schmidt no es la mejor alternativa para realizar la ortogonalización necesaria en el método IDCV-O, ya que como es bien sabido este método presenta problemas de estabilidad numérica [42].

Para validar el método IDCV-O cuando nuevas muestras se incorporan a la base de conocimiento, se emplean las bases de datos Cmu-Pie y Coil-20, descritas en la sección 7.4 (y con más detalle en el apéndice D). Para el caso de nuevas clases, las bases de datos utilizadas son AR y ORL, también descritas en el apéndice D. En ambos casos la experimentación es idéntica a la descrita en la sección 8.1, para el método IDCV-GSO. La tasa de acierto del método IDCV-O es igual a la del método IDCV-GSO, y por tanto a la de DCV-EVD, y por este motivo no se muestra.

La figura 8.6 muestra el tiempo de CPU relativo del método IDCV-O con respecto al método DCV-GSO, para diferentes tamaños del conjunto de actualización.



(a) Cmu-Pie (b) Coil-20

Figura 8.6: Tiempo de CPU relativo del método IDCV-O respecto al método DCV-GSO, al clasificar las bases de datos: Cmu-Pie y Coil-20, para diferentes tamaños del conjunto de actualización como son $n_j = [1, 2, 3, 4, 5]$.

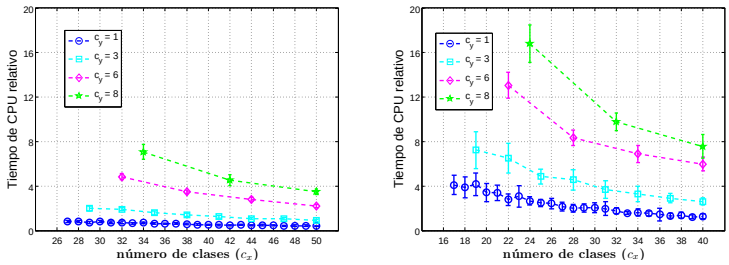
Al comparar estos resultados con los de la figura 8.3, se observa que el coste computacional de IDCV-O es significativamente menor al del algoritmo DCV-GSO, básicamente son 1/3 de los de DCV-GSO.

En la base de datos Cmu-Pie, el coste computacional se reduce como mínimo un 80 %, y un 88 % en la base de datos Coil-20. Además el coste relativo de IDCV-O es asintóticamente decreciente en función del número de muestras de entrenamiento total con una tendencia similar a la observada anteriormente de $O(1/M)$, y respalda por el resultado teórico.

Aunque teóricamente el coste asintótico de los algoritmos IDCV-O e IDCV-GSO es el mismo, de forma experimental se observa (figuras 8.3 y 8.6) que IDCV-O emplea menos tiempo computacional respecto a IDCV-GSO. Esto se puede explicar parcialmente, por el hecho que el número de operaciones involucradas en los algoritmos son una aproximación del número real de operaciones realizadas.

En el apéndice E, en las tablas E.13, E.14, E.15 y E.16, se muestran numéricamente los resultados de la validación empírica del método IDCV-O respecto al método con aprendizaje por lotes DCV-GSO, para las bases de datos Cmu-Pie, Coil-20, Mnist y Usps, respectivamente.

En cuanto al tiempo de CPU relativo de IDCV-O respecto a DCV-GSO cuando el conjunto de entrenamiento se actualiza con nuevas clases, en la figura 8.7 se observa que para valores pequeños de c_y (1-3), el coste del algoritmo incremental es mucho menor que el del algoritmo DCV-GSO, e incluso después de múltiples actualizaciones el coste del algoritmo incremental es casi constante en relación al del aprendizaje por lotes.



(a) AR (b) ORL

Figura 8.7: Tiempo de CPU relativo del método IDCV-O respecto al método DCV-GSO, al incrementar el número de clases en el conjunto entrenamiento.

La reducción en el coste computacional en IDCV-O es como mínimo del 92 % y del 84 % para las bases de datos AR y ORL, respectivamente. Este valor es cada vez mayor cuando la diferencia entre el tamaño del conjunto de entrenamiento inicial y el de actualización es más grande. También, se observa que si existiera un número mayor de clases para la

actualización, el tiempo de CPU relativo tendería a un límite constante relacionado con el tamaño del conjunto de actualización.

En la base de datos AR, figura 8.7a, el coste computacional presenta un mayor decremento al de la base de datos ORL (sin dejar de ser esta reducción importante), porque la relación entre la dimensión del espacio original y el tamaño del conjunto de entrenamiento es menor en AR que en ORL, como se muestra en la figura 8.8.

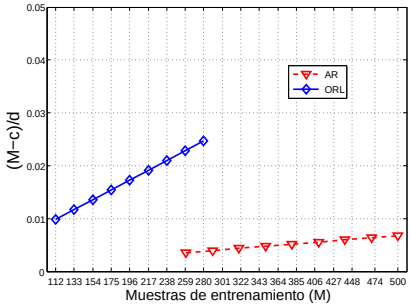


Figura 8.8: Relación de la diferencia entre el tamaño del conjunto de entrenamiento y el número de clases sobre la dimensión del espacio original, $(M - c)/d$, en función del número de muestras de entrenamiento M.

Al comparar los resultados del algoritmo IDCV-O (figura 8.7) con los de IDCV-GSO (figura 8.5), se observa que el coste de IDCV-O está menos acusado respecto a la dimensionalidad y al tamaño del conjunto de entrenamiento que IDCV-GSO, ya que los valores de los parámetros son los mismo en ambos algoritmos.

Las tablas E.17 y E.18, muestran de forma numérica los resultados experimentales del método IDCV-O con respecto a DCV-GSO, cuando nuevas clases se incorporan al conjunto de entrenamiento.

8.3. Discusión

Los métodos incrementales propuestos respecto al método de DCV presentan las mismas características en cuanto a la tasa de acierto. De este modo, e igual que el método IDCV-EVD, las aproximaciones realizadas para obtener las matrices de proyección de forma incremental al subespacio discriminante no afectan la capacidad discriminante del método.

Como ya se ha dicho antes, el tiempo de CPU del método original DCV-GSO tiene un comportamiento cuadrático, que es más o menos evidente en relación al número de muestras en el conjunto de entrenamiento y a la dimensión del espacio original.

En cuanto al método incremental IDCV-GSO, la tendencia del tiempo de CPU es lineal y está en función del número de muestras (clases) que se incorporan de forma incremental a la base de conocimiento, e implícitamente del tamaño total del conjunto de actualización. Además, para un número pequeño de muestras o clases incorporadas de forma incremental el tiempo de CPU de la fase de entrenamiento, es significativamente menor respecto al método DCV-GSO con aprendizaje por lotes.

Aunque teóricamente (asintóticamente) el coste computacional de los métodos IDCV-GSO y IDCV-O a partir de la ortogonalización de Gram-Schmidt es el mismo, desde la validación empírica se observa que el método IDCV-O emplea menos tiempo computacional respecto al método IDCV-GSO. Esto se explica parcialmente porque el número de operaciones involucradas en los algoritmos es una aproximación del número real de operaciones realizadas.

Además se observa en ambos métodos, IDCV-GSO y IDCV-O, que la tendencia del tiempo de CPU relativo decrece en función del tamaño del conjunto de entrenamiento precalculado. Cuando el tamaño del conjunto de entrenamiento precalculado es mucho mayor al tamaño del conjunto de actualización, desde la experimentación se observa que el tiempo de CPU relativo tiende a un límite constante.

Capítulo 9

Vectores comunes discriminantes extendidos de forma incremental

Resumen — Como ya se ha dicho, cuando la dimensión del espacio original de las muestras de entrenamiento es cercana al número de muestras, el método DCV no proporciona información suficiente para una adecuada clasificación. Tamura et al. propusieron en [105] el método de vectores comunes extendidos, que proporciona una solución a las limitaciones del método DCV. Sin embargo, el aprendizaje del método es por lotes, lo cual ocasiona que el coste computacional y la memoria requerida sean cada vez mayores cuando nuevas muestras se incorporan a la base de conocimiento. Para dar una solución a este problema, en este capítulo se propone un aprendizaje incremental del método RDCV que llamamos IRDCV del inglés *Incremental Discriminative Rough Common Vector*.

Contenido

9.1. Planteamiento 102
9.2. Desarrollo teórico 103
9.3. Evaluación empírica 111
9.4. Discusión 114

9.1. Planteamiento

Como en los capítulos anteriores, sea $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}] \in \mathbb{R}^{d\times(M+N)}$ el conjunto de entrenamiento total (formado por los conjuntos de entrenamiento inicial $\mathbf{X} \in \mathbb{R}^{d\times M}$ y el de actualización $\mathbf{Y} \in \mathbb{R}^{d\times N}$) en el espacio $d-$dimensional. En este espacio vectorial existen dos subespacios complementarios y ortogonales entre si definidos como el subespacio rango y nulo de la matriz de dispersión intra-clase de $\mathbf{Z}$, representados en la figura 9.1 por $\mathcal{R}(S_Z^w)$ y $\mathcal{N}(S_Z^w)$, respectivamente.

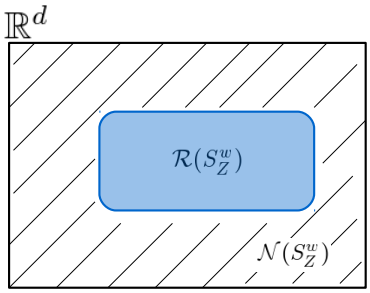


Figura 9.1: Espacio $d-$dimensional, formado por los subespacios rango $\mathcal{R}(S_Z^w)$ y nulo $\mathcal{N}(S_Z^w)$ de la matriz de dispersión intra-clase S_Z^w. Estos dos subespacios son complementarios y ortogonales.

Como ya se ha dicho, cuando el número de muestras de entrenamiento es cercano a la dimensión del espacio, y las muestras son linealmente independientes, la dimensión del espacio nulo es pequeña, y este subespacio puede no ser apropiado para la clasificación. En este caso el método RDCV aumenta la dimensión del subespacio $\mathcal{N}(S_Z^w)$, y en consecuencia se reduce la dimensión del espacio rango $\mathcal{R}(S_Z^w)$ [§4.5]. El subespacio nulo extendido y el subespacio rango reducido de S_Z^w se representan en la figura 9.2 como $\tilde{\mathcal{R}}(S_Z^w)$ y $\tilde{\mathcal{N}}(S_Z^w)$, respectivamente. De este modo el nuevo subespacio nulo extendido puede proporcionar mayor capacidad de discriminación que el subespacio nulo original.

El método incremental que se propone tiene como objetivo obtener implícitamente el operador de proyección al subespacio nulo extendido $\tilde{\mathcal{N}}(S_Z^w)$ a partir del operador de proyección al subespacio rango reducido $\tilde{\mathcal{R}}(S_Z^w)$, ya que son subespacios complementarios. Para ello se utiliza el operador de proyección del subespacio $\tilde{\mathcal{R}}(S_X^w)$ y la nueva información que proporciona el conjunto de entrenamiento $\mathbf{Y}$, sin la necesidad de entrenar el sistema a partir del número total de muestras de entrenamiento.

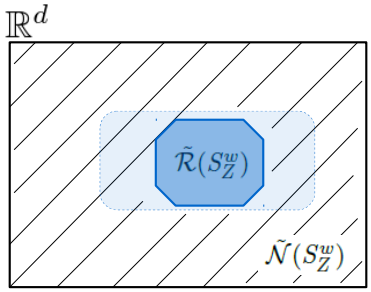


Figura 9.2: Subespacio nulo extendido y subespacio rango reducido de S_Z^w.

9.2. Desarrollo teórico

9.2.1. Antecedentes

Obtener los vectores comunes extendidos discriminantes del conjunto de entrenamiento $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}] \in \mathbb{R}^{d\times(M+N)}$, implica calcular la EVD de la matriz de dispersión intra-clase $S_Z^w = \mathrm{U}'\Lambda'\mathrm{U}'^T$, para obtener los vectores propios asociados a los valores propios normalizados no nulos λ_i' que satisfacen la condición

$$(1-\alpha) \geq \sum_{i=1}^{k} \lambda_i', \tag{9.1}$$

donde α es la constante que establece, según el valor de los valores propios λ', cuánto se expande el subespacio nulo de la matriz de dispersión intra-clase del conjunto de entrenamiento. Los valores propios están organizados de forma decreciente, $\lambda_1' > \lambda_2' > \cdots > \lambda_k' > \cdots > \lambda_{r_z}'$, y su suma es igual a $\sum_{i=1}^{r_z} \lambda_i' = 1$ y $k \leq r_z$. r_z es el rango de S_Z^w.

Como se ha visto en la sección 4.5, el conjunto de vectores propios asociados a los valores propios λ_i' que satisfacen la condición descrita en la ecuación (9.1), forman implícitamente el operador de proyección, $\overline{\mathrm{U}'}_\alpha \overline{\mathrm{U}'}_\alpha{}^T$, al espacio nulo extendido de S_Z^w. Los vectores comunes extendidos se definen como

$$\mathrm{z}_{rcv}^j = \overline{\mathrm{U}'}_\alpha \overline{\mathrm{U}'}_\alpha{}^T \mathrm{z}_j^i = \mathrm{z}_j^i - \mathrm{U}'_\alpha \mathrm{U}'_\alpha{}^T \mathrm{z}_j^i,$$

para $\mathrm{U}'_\alpha = [\mathrm{u}'_1 \ \ldots\ \mathrm{u}'_k]$, $\overline{\mathrm{U}'}_\alpha = [\mathrm{u}_{(k+1)} \ \ldots\ \mathrm{u}_{r_z}\ \mathrm{u}_{(r_z+1)} \ \ldots\ \mathrm{u}_d]$ y z_j^i un vector arbitrario de $\mathbf{Z}$.

El espacio rango reducido $\tilde{\mathcal{R}}(S_Z^w)$, de la matriz de dispersión intra-clase de $\mathbf{Z}$, se genera a partir de los vectores que conforman U'_α. Y los vectores de $\overline{\mathrm{U}'}_\alpha$ generan el espacio nulo extendido $\tilde{\mathcal{N}}(S_Z^w)$ de S_Z^w.

Luego, la matriz de proyección final, W_{rdcv}, se calcula a partir de aplicar un PCA [§4.2] a los vectores comunes extendidos z^j_{rcv}, o desde una base ortonormal del subespacio diferencia de los z^j_{rcv}.

$$\mathrm{z}^j_{rdcv} = \mathrm{W}^T_{rdcv}\mathrm{z}^i_j$$

El número de operaciones involucradas en el cálculo de los z^j_{rcv} es aproximadamente de $O(\ell((M+N)^3) + d(M+N)^2 + d(M+N)r_{z_\alpha})$ del cálculo de S^w_Z y de su EVD. r_{z_α} es el rango del subespacio rango reducido $\tilde{\mathcal{R}}(S^w_Z)$. La segunda parte del método RDCV, a partir de subespacios diferencia y ortogonalización, conlleva $O(dc^2)$ operaciones.

De lo anterior observamos que la segunda fase implica un coste computacional significativamente menor en cuanto a la primera fase, y así como se ha hecho para el método DCV, en las siguientes secciones se plantea y valida un algoritmo incremental del método RDCV respecto a la primera fase.

9.2.2. RDCV incremental

Se propone un desarrollo incremental del método RDCV. Sea $\mathbf{Y} \in \mathbb{R}^{d\times N}$ el conjunto de entrenamiento para actualizar la información de $\mathbf{X} \in \mathbb{R}^{d\times M}$. Se supone en principio que $\mathbf{X}$ y $\mathbf{Y}$ presentan las mismas clases.

Como se ha visto en el capítulo 7 [§7.2.1], la descomposición en valores y vectores propios de la matriz de dispersión intra-clase, S^w_Z, de $\mathbf{Z}$ se puede expresar en función de los valores y vectores propios de las matrices de dispersión intra-clase S^w_X y S^w_Y. La descomposición de S^w_X procede del entrenamiento preliminar y es más costosa respecto a la descomposición de S^w_Y, ya que $N \ll M$. La matriz S^w_Z se puede escribir en función de S^w_X, S^w_Y y de un término $S = \mathrm{J}\mathrm{J}^T$ que relaciona las medias de los datos de entrenamiento iniciales, x_j, y las medias de los nuevos datos, y_j, como se muestra en el apéndice B.

$$S^w_Z = S^w_X + S^w_Y + \mathrm{J}\mathrm{J}^T \tag{9.2}$$

En esta expresión, $\mathrm{J} \in \mathbb{R}^{d\times c}$ es una matriz cuyas columnas están en función de la diferencia de los vectores media de $\mathbf{X}$ y de $\mathbf{Y}$, como en la ecuación 7.2. J relaciona los centros de cada clase de $\mathbf{X}$ con los de $\mathbf{Y}$, ya que estos no están necesariamente en los subespacios propios de $\mathbf{X}$ o de $\mathbf{Y}$.

Sea U_α y Λ_α un conjunto reducido de vectores y valores propios, respectivamente, asociados a la matriz de dispersión S^w_X. La descomposición en valores y vectores propios de S^w_Y está dada por $\mathrm{V}\Delta\mathrm{V}^T$. La

correspondiente aproximación de la ecuación (9.2) en términos de valores y vectores propios se escribe como:

$$\mathrm{U}'\Lambda'\mathrm{U}'^{T} \approx \mathrm{U}_\alpha\Lambda_\alpha\mathrm{U}_\alpha^T + \mathrm{V}\Delta\mathrm{V}^T + \mathrm{J}\mathrm{J}^T \tag{9.3}$$

Con $\tilde{\mathcal{R}}(S_X^w)$ como el espacio rango reducido de $\mathbf{X}$ (generado por U_α en el entrenamiento anterior), y $\mathcal{R}(S_Y^w)$ como el espacio rango del nuevo conjunto de entrenamiento $\mathbf{Y}$. Es posible aproximar el subespacio rango de la matriz de dispersión S_Z^w en función de los subespacios $\tilde{\mathcal{R}}(S_X^w)$, $\mathcal{R}(S_Y^w)$ y $span\{\mathrm{J}\}$, como

$$\mathcal{R}(S_Z^w) \approx \tilde{\mathcal{R}}(S_X^w) + \mathcal{R}(S_Y^w) + span\{\mathrm{J}\} \tag{9.4}$$

Desde las propiedades de los subespacios, una base ortonormal del subespacio rango reducido $\tilde{\mathcal{R}}(S_Z^w)$ se puede obtener en función de una base ortonormal del subespacio $\tilde{\mathcal{R}}(S_X^w)$, como es U_α, y una base ortonormal v (ver el teorema 3). v debe ser una base ortonormal del subespacio generado por V y J, y a la vez ser ortogonal a U_α, para que la intersección de los subespacios sea nula.

De este modo, una base ortonormal U'_α de $\tilde{\mathcal{R}}(S_Z^w)$ se puede obtener en función de U_α, v y en función de una matriz de rotación R_α. La matriz R_α reorganiza los ejes de proyección y permite reducir el subespacio resultante.

$$\mathrm{U}'_\alpha = [\mathrm{U}_\alpha\ v]\,\mathrm{R}_\alpha \tag{9.5}$$

La descomposición en valores y vectores propios de la matriz de dispersión S_Z^w, ecuación (9.3), se reescribe como:

$$[\mathrm{U}_\alpha\ v]\,\mathrm{R}\ \Lambda'\ \mathrm{R}^T[\mathrm{U}_\alpha\ v]^T = \mathrm{U}_\alpha\Lambda_\alpha\mathrm{U}_\alpha^T + \mathrm{V}\Delta\mathrm{V}^T + \mathrm{J}\mathrm{J}^T \tag{9.6}$$

La base ortonormal v, que tiene que ser ortogonal a U_α y generar el subespacio definido por [V J], se puede calcular a partir de proyectar [V J] en el espacio nulo extendido de S_X^w. V es el conjunto de vectores propios asociados a los valores propios no nulos de S_Y^w. Luego los vectores que conforman v se calculan a partir de encontrar una base ortonormal de este conjunto proyectado,

$$v = orth\left(\overline{\mathrm{U}}_\alpha\overline{\mathrm{U}}_\alpha^T\,[\mathrm{V}\ \mathrm{J}]\right) \tag{9.7}$$

donde $\overline{\mathrm{U}}_\alpha\overline{\mathrm{U}}_\alpha^T\,[\mathrm{V}\ \mathrm{J}] = [\mathrm{V}\ \mathrm{J}] - \mathrm{U}_\alpha\mathrm{U}_\alpha^T\,[\mathrm{V}\ \mathrm{J}]$.

Una vez que se calcula la base v, la matriz de rotación se puede obtener multiplicando ambos lados la ecuación (9.6) por la izquierda por $[\mathrm{U}_\alpha\ v]^T$, y por la derecha por $[\mathrm{U}_\alpha\ v]$. Y usando el hecho de que $[\mathrm{U}_\alpha\ v]^T$ es una inversa izquierda de $[\mathrm{U}_\alpha\ v]$, se obtiene

$$
\begin{aligned}
\mathrm{R} \quad & \Lambda' \, \mathrm{R}^T = [\mathrm{U}_\alpha \; v]^T (S_X^w + S_Y^w + \mathrm{JJ}^T)[\mathrm{U}_\alpha \; v] \\
= \quad & \begin{bmatrix} \mathrm{U}_\alpha^T S_X^w \mathrm{U}_\alpha & \mathrm{U}_\alpha^T S_X^w v \\ v^T S_X^w \mathrm{U}_\alpha & v^T S_X^w v \end{bmatrix} + \begin{bmatrix} \mathrm{U}_\alpha^T S_Y^w \mathrm{U}_\alpha & \mathrm{U}_\alpha^T S_Y^w v \\ v^T S_Y^w \mathrm{U}_\alpha & v^T S_Y^w v \end{bmatrix} + \begin{bmatrix} \mathrm{U}_\alpha^T \mathrm{JJ}^T \mathrm{U}_\alpha & \mathrm{U}_\alpha^T \mathrm{JJ}^T v \\ v^T \mathrm{JJ}^T \mathrm{U}_\alpha & v^T \mathrm{JJ}^T v \end{bmatrix} \\
= \quad & \begin{bmatrix} \Lambda_\alpha & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathrm{U}_\alpha^T \mathrm{V} \Delta \mathrm{V}^T \mathrm{U}_\alpha & \mathrm{U}_\alpha^T \mathrm{V} \Delta \mathrm{V}^T v \\ v^T \mathrm{V} \Delta \mathrm{V}^T \mathrm{U}_\alpha & v^T \mathrm{V} \Delta \mathrm{V}^T v \end{bmatrix} + \begin{bmatrix} \mathrm{U}_\alpha^T \mathrm{JJ}^T \mathrm{U}_\alpha & \mathrm{U}_\alpha^T \mathrm{JJ}^T v \\ v^T \mathrm{JJ}^T \mathrm{U}_\alpha & v^T \mathrm{JJ}^T v \end{bmatrix} \qquad (9.8)
\end{aligned}
$$

Con lo cual se deduce que es posible calcular R y Λ' a partir de diagonalizar la parte derecha de la ecuación (9.8), de una manera similar a la propuesta por Hall et. al. en [50, 51] para el PCA incremental.

La matriz de rotación reducida R_α, de la ecuación 9.5, se obtiene desde los k vectores propios de R asociados a los valores propios λ'_i que satisfacen

$$(1 - \alpha') \geq \sum_{i=1}^{k} \lambda'_i \qquad (9.9)$$

donde α' es el valor de la variabilidad a añadir al subespacio nulo calculado de forma incremental. El valor de α' depende de la suma de los valores propios del entrenamiento anterior, $A_o = \sum_{i=1}^{k} \lambda_i$, y de la suma de los valores propios calculados de forma incremental, $A' = \sum_{i=1}^{(k+N)} \lambda'_i$. La variabilidad total del entrenamiento inicial es igual a $(1-\alpha)A_o + \alpha A_o$, y la variabilidad suprimida es igual a $\alpha/(1-\alpha)\, A_o$.

La variabilidad total en el conjunto de entrenamiento incremental está dada por la suma de los valores propios obtenidos en el entrenamiento incremental más una corrección debida a la variabilidad suprimida en el entrenamiento anterior. De este modo

$$
\begin{aligned}
(1 - \alpha') \quad &= \quad (1 - \alpha)(A' + \alpha/(1 - \alpha)A_o) \\
&= \quad (1 - \alpha)A' + \alpha A_o \\
&= \quad (1 - \alpha) \sum_{i=1}^{(k+N)} \lambda'_i + \alpha \sum_{i=1}^{k} \lambda_i \qquad (9.10)
\end{aligned}
$$

La figura 9.3 muestra de forma intuitiva lo que ocurre al limitar la variabilidad de los conjuntos de entrenamiento total, inicial e incremental, a partir de los valores propios.

Los nuevos vectores U'_α son una combinación de los viejos U_α aumentada con un conjunto de vectores ortonormales v (que están en el espacio nulo extendido de S_X^w), que se reorganizan a partir de la matriz de rotación R_α. De forma que los nuevos vectores propios se obtienen como $\mathrm{U}'_\alpha = [\mathrm{U}_\alpha \; v]\, \mathrm{R}_\alpha$.

De este modo, los nuevos vectores comunes extendidos z^j_{rcv} son

$$\mathrm{z}^j_{rcv} = \mathrm{z}^i_j - \mathrm{U}'_\alpha \mathrm{U}'^{\,T}_\alpha \mathrm{z}^i_j$$

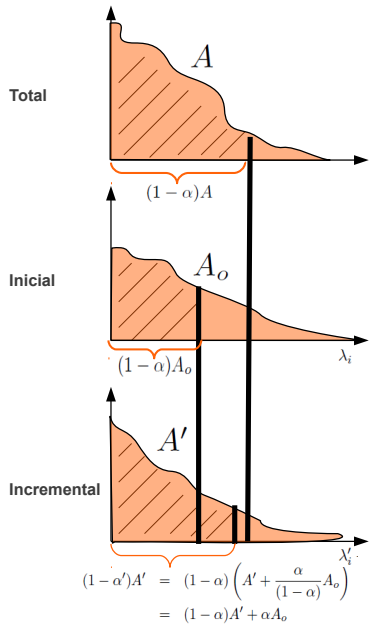


Figura 9.3: Variabilidad de los conjuntos de entrenamiento total, inicial e incremental, limitada a partir del los valores propios. Donde, A representa la suma de los valores propios de todo el conjunto de entrenamiento. $A_o = \sum_{i=1}^{k} \lambda_i$ es la suma de los valores propios del entrenamiento anterior. $A' = \sum_{i=1}^{(k+N)} \lambda'_i$ es la suma de los nuevos valores propios calculados de forma incremental. α es el valor de la variabilidad que se añade al subespacio nulo original. α' es el valor de la variabilidad que se añade al subespacio nulo calculada de forma incremental.

El algoritmo 7 muestra los pasos para aplicar el método IRDCV, donde los argumentos de entrada son la variabilidad α que se añade al subespacio nulo, los vectores propios U_α, la matriz diagonal de valores propios asociados Λ_α y las medias de cada clase del entrenamiento anterior, y por supuesto el nuevo conjunto de entrenamiento. Como argumentos de salida están la matriz de proyección final W, los vectores comunes discriminantes extendidos z^j_{rdcv}, la nueva matriz diagonal de valores propios Λ'_α y los vectores asociados U'_α, así como los vectores media actualizados.

Algoritmo 7 *Método IRDCV que actualiza la información del conjunto de entrenamiento* $\mathbf{X}$.

Entrada: $\mathrm{U}_\alpha \in \mathbb{R}^{d \times r_{x_\alpha}}$, $\Lambda_\alpha \in \mathbb{R}^{r_{x_\alpha} \times r_{x_\alpha}}$, α, x_j, $\mathbf{Y}$.
Salida: W_{rdcv}, z^j_{rdcv}, U'_α, ${\Lambda'}_\alpha$, z_j.

1. Se calcula la matriz de dispersión intra-clase de $\mathbf{Y}$, y se realiza la EVD, $S^w_Y = \mathrm{V}\Delta\mathrm{V}^T$.
2. Se obtiene la matriz que relaciona los vectores media de $\mathbf{X}$ e $\mathbf{Y}$.
   $\mathrm{J} = \left[\sqrt{\frac{m_1 n_1}{(m_1+n_1)}}(\mathrm{x}_1 - \mathrm{y}_1) \ \ldots \ \sqrt{\frac{m_j n_j}{(m_j+n_j)}}(\mathrm{x}_j - \mathrm{y}_j)\right], \qquad j = 1, \ldots, c.$
3. Se calcula la base ortonormal $v = orth\left([\mathrm{V}\ v] - \mathrm{U}_\alpha \mathrm{U}_\alpha^T [\mathrm{V}\ v]\right)$.
4. La matriz de rotación R y los nuevos valores propios Λ', se consiguen a partir de solucionar el eigenproblema de la ecuación (9.8).
5. Se obtiene R_α desde los vectores de R asociados a los valores propios λ'_i, que satisfacen

   $$(1 - \alpha') \geq \sum_{i=1}^{k} \lambda'_i$$

   donde $(1-\alpha') = (1-\alpha)\sum_{i=1}^{(k+N)} \lambda'_i + \alpha \sum_{i=1}^{k} \lambda_i$.
6. Los nuevos vectores propios U'_α y valores propios ${\Lambda'}_\alpha$, están dados por

   $$\begin{aligned} \mathrm{U}'_\alpha &= [\mathrm{U}_\alpha \ \ v]\ \mathrm{R}_\alpha \\ {\Lambda'}_\alpha &= diag(\lambda'_1 > \lambda'_2 > \ldots > \lambda'_k) \end{aligned}$$

7. Los nuevos vectores comunes extendidos son: $\mathrm{z}^j_{rcv} = \mathrm{z}^i_j - \mathrm{U}'_\alpha {\mathrm{U}'_\alpha}^T \mathrm{z}^i_j$.
8. Luego, la matriz de proyección final se calcula como $\mathrm{W}_{rdcv} = orth(\mathcal{B}_{rcv})$, donde $\mathcal{B}_{rcv} = span\{\mathrm{z}^2_{rcv} - \mathrm{z}^1_{rcv} \ \ldots \ \mathrm{z}^c_{rcv} - \mathrm{z}^1_{rcv}\}$.
9. Finalmente, los nuevos RDCV están dados por

   $$\mathrm{z}^j_{rdcv} = \mathrm{W}^T_{rdcv} \mathrm{z}^i_j$$

   En un nuevo entrenamiento se repiten los anteriores pasos. Pero ahora, los valores y vectores propios iniciales están dados por ${\Lambda'}_\alpha$ y U'_α, respectivamente. El vector media y el número de muestras total en cada clase son $\left(\frac{m_j}{(m_j+n_j)}\mathrm{x}_j + \frac{n_j}{(m_j+n_j)}\mathrm{y}_j\right)$ y $(m_j + n_j)$, respectivamente.

El diagrama de la figura 9.4 resume el desarrollo de la primera fase del método IRDCV.

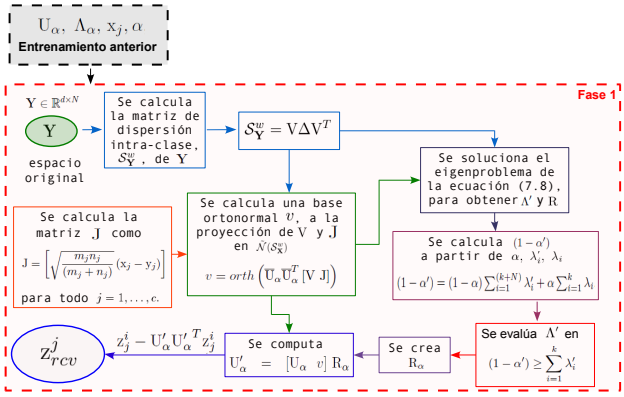


Figura 9.4: Primera fase del método IRDCV, cuando nuevas muestras se incorporan a la base de conocimiento.

En el caso del aprendizaje por lotes el coste computacional del método RDCV para el conjunto de datos $\mathbf{Z}$, está dominado por $O(\ell((M+N)^3) + d(M+N)^2 + dr_{z_\alpha}(M+N))$ operaciones.

El coste computacional del algoritmo IRDCV depende del cálculo y de la EVD de la ecuación (9.8), lo cual implica $O(\ell(N^3) + \ell((r_{x_\alpha} + r_y)^3) + dN^2 + dr_{x_\alpha}(r_{z_{\alpha'}} + r_y))$ operaciones. r_{x_α}, r_y, $r_{z_{\alpha'}}$ representan la dimensión de los subespacios $\tilde{\mathcal{R}}(S_X^w)$, $\mathcal{R}(S_Y^w)$ y $\tilde{\mathcal{R}}(S_Z^w)$, respectivamente. La dimensión de los subespacios involucrados en el algoritmo incremental es menor al número de muestras del entrenamiento total, donde $N << M$ y $(r_{x_\alpha} + r_y) < (M+N)$.

Una muestra por clase

Cuando el nuevo conjunto de entrenamiento solo dispone de una muestra por clase, el método IRDCV sufre pequeñas modificaciones. En este caso, y como se ha mencionado en capítulos anteriores, la matriz de dispersión intra-clase S_Y^w no se puede calcular, y por lo tanto la matriz de dispersión intra-clase del conjunto de entrenamiento $\mathbf{Z}$ se redefine como (ver apéndice B):

$$S_Z^w = S_X^w + JJ^T$$

donde $J = [J_1\ \ldots\ J_c]$ y $J_j = \sqrt{\frac{m_j}{(m_j+1)}}\,(x_j - y_j^1)$. Con x_j como el vector media de la clase j en $\mathbf{X}$.

Una aproximación del subespacio rango de S_Z^w en función del subespacio $\tilde{\mathcal{R}}(S_X^w)$ y el subespacio generado por J está dada por

$$\mathcal{R}(S_Z^w) \approx \tilde{\mathcal{R}}(S_X^w) + span\{\mathrm{J}\}$$

Como en el caso anterior, la base ortonormal del subespacio $\tilde{\mathcal{R}}(S_Z^w)$ se puede escribir en función de U_α (del entrenamiento anterior), de v que es una base ortonormal de la proyección de J en el espacio nulo extendido de S_X^w, y en función de una matriz de rotación R_α.

$$\mathrm{U}'_\alpha = [\mathrm{U}_\alpha\ v]\,\mathrm{R}_\alpha$$

donde $v = orth(\mathrm{J} - \mathrm{U}_\alpha \mathrm{U}_\alpha^T\,\mathrm{J})$.

La matriz de rotación R_α se obtiene a partir de realizar la descomposición en valores y vectores propios de

$$\begin{aligned}
\mathrm{R}\,\Lambda'\,\mathrm{R}^T &= [\mathrm{U}_\alpha\ v]^T(S_X^w + \mathrm{J}\mathrm{J}^T)[\mathrm{U}_\alpha\ v] \\
&= \begin{bmatrix} \mathrm{U}_\alpha^T S_X^w \mathrm{U}_\alpha & \mathrm{U}_\alpha^T S_X^w v \\ v^T S_X^w \mathrm{U}_\alpha & v^T S_X^w v \end{bmatrix} + \begin{bmatrix} \mathrm{U}_\alpha^T \mathrm{J}\mathrm{J}^T \mathrm{U}_\alpha & \mathrm{U}_\alpha^T \mathrm{J}\mathrm{J}^T v \\ v^T \mathrm{J}\mathrm{J}^T \mathrm{U}_\alpha & v^T \mathrm{J}\mathrm{J}^T v \end{bmatrix} \\
&= \begin{bmatrix} \Lambda_\alpha & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathrm{U}_\alpha^T \mathrm{J}\mathrm{J}^T \mathrm{U}_\alpha & \mathrm{U}_\alpha^T \mathrm{J}\mathrm{J}^T v \\ v^T \mathrm{J}\mathrm{J}^T \mathrm{U}_\alpha & v^T \mathrm{J}\mathrm{J}^T v \end{bmatrix} \qquad (9.11)
\end{aligned}$$

Los vectores propios asociados a los valores propios λ'_i que satisfacen la condición descrita en la ecuación (9.9), son los vectores columna que forman la matriz de rotación R_α.

El coste computacional asociado al método IRDCV cuando la información de una nueva muestra por clase se incorpora al conjunto de entrenamiento es de $O(\ell((r_{x_\alpha}+c)^3)+dr_{x_\alpha}c+dr_{z_{\alpha'}}(r_{x_\alpha}+c))$ operaciones.

Nuevas clases

El método IRDCV también se ha propuesto para incorporar nuevas clases al conjunto de entrenamiento donde el planteamiento es similar al descrito anteriormente. Pero ahora, la matriz de dispersión intra-clase de $\mathbf{Z}$ se descompone como $S_Z^w = S_X^w + S_Y^w$. Y el subespacio rango de $\mathbf{Z}$ se define como $\mathcal{R}(S_Z^w) \approx \tilde{\mathcal{R}}(S_X^w) + \mathcal{R}(S_Y^w)$.

Cuando nuevas clases se incorporan a la base de conocimiento se asumen las mismas suposiciones generales que en los casos anteriores. Un planteamiento más exhaustivo del método IRDCV para nuevas clases se presenta y valida en el artículo *Empirical evaluation of class-updating incremental discriminative common vector based face recognition* citado como [27].

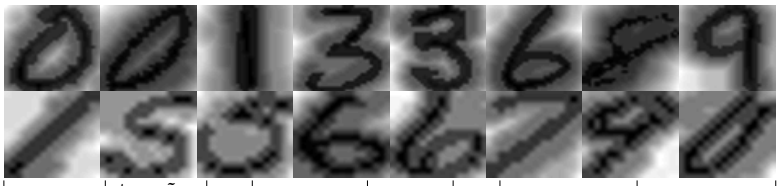
9.3. Evaluación empírica

Como se ha visto en el capítulo 7, el método DCV no presenta buenos resultados sobre las bases de datos Mnist y Usps, porque la diferencia entre la dimensión del espacio original y el número de muestras de entrenamiento es pequeña. La validación del algoritmo IRDCV se realiza a partir de medir la tasa de acierto y el tiempo de CPU respecto al método RDCV original, al clasificar ambas bases de datos.

El valor del parámetro que añade variabilidad al espacio nulo de la matriz de dispersión intra-clase se incrementa en intervalos de 0.05 desde 0.05 a 0.3. Puesto que en este caso se trata de una aproximación, no se utilizan valores mayores de 0.3 porque el método pierde su capacidad de discriminación al distorsionar excesivamente el espacio nulo.

Los vectores comunes discriminantes extendidos se calculan a partir de la media de cada clase y de la matriz de proyección, ambas actualizadas. La distancia euclídea y el primer vecino más próximo son las medidas utilizadas para comparar el conjunto de entrenamiento con el de prueba.

Aquí se sigue el mismo protocolo de evaluación que se explicó en el capítulo 7, y se recuerda solo lo que consideramos más importante. La figura 9.5 da a conocer imágenes de las bases de datos Mnist y Usps, así como sus características principales y parámetros relevantes del proceso de entrenamiento. El apéndice D especifica con más detalle las características de estas bases de datos.

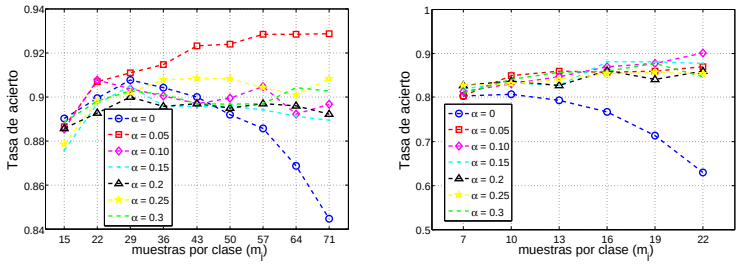


nombre	**tamaño imagen**	c	**tamaño**/c	m_j	n_j	α	**tipo**
Mnist	32×32	10	100	[15, 71]	7	[0.05:0.05:0.3]	dígitos de 0-9
Usps	16×16	10	25	[7, 22]	3	[0.05:0.05:0.3]	dígitos de 0-9

Figura 9.5: Algunas imágenes y características de las bases de datos Mnist y Usps. Además de algunos parámetros relevantes en la validación del método IRDCV. c es el número de clases, tamaño/c es el número de muestras por clase en toda la base de datos. m_j y n_j son el número de muestras por clase para el entrenamiento por lotes e incremental, respectivamente. α es el valor de la variabilidad añadida al subespacio de proyección.

El conjunto de entrenamiento disponibles para la actualización se ha dividido en varios tamaños obteniendo múltiples resultados que se muestran en el apéndice E.4. A nivel practico, aquí se presentan los resultados para un valor representativo del tamaño del conjunto de actualización, como es $n_j = 7$ para Mnist y $n_j = 3$ para Usps.

La tasa de acierto del método RDCV al incorporar en cada entrenamiento nuevas muestras, y para valores de varianza añadida al subespacio de proyección en el rango de [0-0.3] con incrementos de 0.05, se muestra en la figura 9.6. Cuando $\alpha = 0$, se aplica el método DCV original.



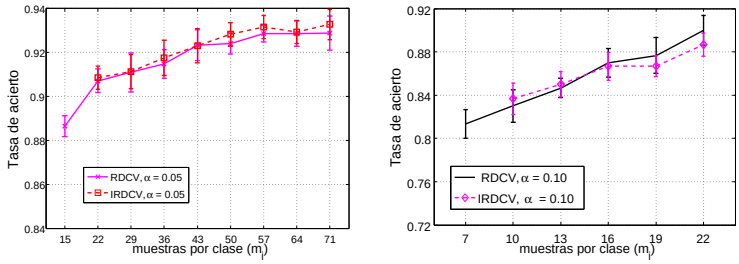
(a) Mnist (b) Usps

Figura 9.6: Tasa de acierto en función del conjunto de entrenamiento acumulado para los métodos DCV ($\alpha = 0$) y RDCV ($\alpha > 0$), al clasificar las bases de datos Mnist y Usps.

En ambas bases de datos la tasa acierto del método DCV ($\alpha = 0$) se deteriora a mayor número de muestras de entrenamiento. Esto sucede porque la diferencia entre la dimensionalidad del espacio original y el número de muestras de entrenamiento es cada vez más pequeña. Estos detalles ya se discutieron en la sección 7.4.1.

Cuando $\alpha = 0.05$, se observa que la tasa de acierto en Mnist (figura 9.6a) mejora a mayor número de muestras de entrenamiento. Respecto a la base de datos Usps, la tasa de acierto para los diferentes valores de $\alpha > 0$ no presenta diferencias apreciables, sin embargo ésta es mayor a la del método DCV ($\alpha = 0$).

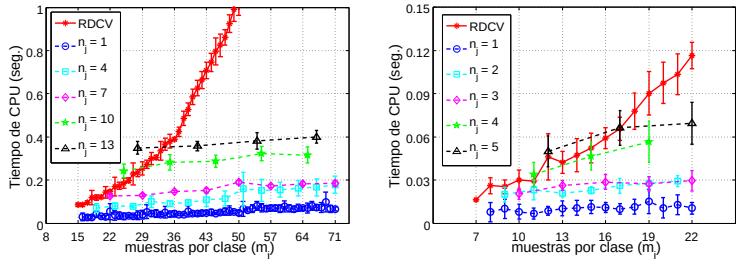
Las tasas de acierto de RDCV e IRDCV al clasificar las bases de datos Mnist y Usps, con $n_j = 7$ y $\alpha = 0.5$ en Mnist y, $n_j = 3$ y $\alpha = 0.10$ en Usps, se muestran en la figura 9.7. n_j es el número de muestras por clase en el conjunto de actualización. En este caso, la tasa de acierto del método incremental presenta pequeñas variaciones en cuanto al método RDCV (con aprendizaje por lotes) pero estas pequeñas diferencias no son relevantes, y corresponden a aproximaciones numéricas de la implementación.



(a) Mnist (b) Usps

Figura 9.7: Tasa de acierto en función del conjunto de entrenamiento acumulado para los métodos RDCV e IRDCV, al clasificar Mnist y Usps. Con $\alpha = 0.05$ y $\alpha = 0.10$ como los valores de variabilidad añadida al subespacio de proyección y, $n_j = 7$ y $n_j = 3$ como el tamaño del conjunto de actualización por clase en Mnist y Usps, respectivamente.

El tiempo de CPU en segundos de los métodos extendidos, para $n_j = [1, 4, 7, 10, 13]$ en Mnist (α =0.05) y $n_j = [1, 2, 3, 4, 5]$ en Usps (α =0.10), se muestra en la figura 9.8. Se observa que el tiempo computacional del algoritmo incremental es lineal, mientras que el del método RDCV presenta una tendencia cuadrática (que es más evidente en Mnist, figura 9.8a). Las variaciones que presenta el tiempo computacional en Usps se deben a que el rango en el que se mueven los tiempos es pequeño.

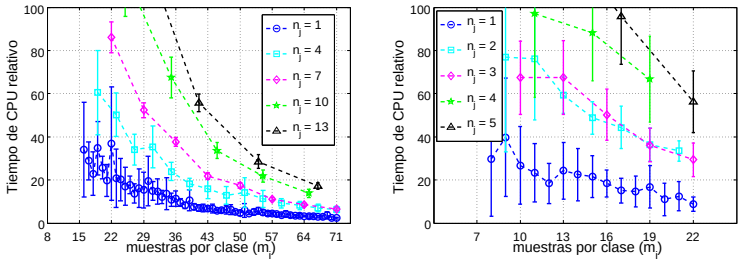


(a) Mnist (b) Usps

Figura 9.8: Tiempo de CPU en segundos, en la fase de entrenamiento, de los métodos RDCV e IRDCV en las bases de datos Mnist y Usps, para $\alpha = 0.05$ y $\alpha = 0.10$ y, $n_j = [1, 4, 7, 10, 13]$ y $n_j = [1, 2, 3, 4, 5]$, respectivamente.

El hecho de que el algoritmo incremental se comporta casi de forma constante respecto al método con aprendizaje por lotes, sugiere que para valores constantes de la dimensionalidad del espacio original y del tamaño del conjunto de actualización, el número de muestras de entrenamiento total domina con respecto a los rangos de los subespacios involucrados.

El tiempo de CPU relativo del algoritmo incremental IRDCV respecto al método con aprendizaje por lotes se presenta en la figura 9.9, para ambas bases de datos. Se observa que el coste relativo del algoritmo incremental se reduce según crece el tamaño del conjunto de entrenamiento precalculado. La tendencia de $O(1/M)$, como era de esperan desde la figura 9.8b, está más atenuada en la base de datos Usps con respecto a Mnist.



(a) Mnist (b) Usps

Figura 9.9: Tiempo de CPU relativo del método IRDCV respecto al método RDCV, al clasificar las bases de Mnist y Usps, con $\alpha = 0.05$ y $\alpha = 0.10$, y $n_j = [1, 4, 7, 10, 13]$ y $n_j = [1, 2, 3, 4, 5]$, respectivamente.

Los resultados númericos de esta sección se muestran en forma de tablas en el apéndice E.4.

9.4. Discusión

A partir de validar el algoritmo incremental IRDCV se observa que para valores de α pequeños las aproximaciones realizadas en el desarrollo teórico no afectan de forma drástica la tasa de clasificación.

Respecto a la corrección de la variabilidad añadida de forma incremental en el algoritmo IRDCV podemos decir que, ésta permite conservar el criterio discriminante del algoritmo original.

Si bien la tasa de acierto no sufre cambios importantes, no sucede lo mismo en cuanto al tiempo computacional del algoritmo incremental. A mayor número de muestras de entrenamiento en el conjunto precalculado, mayor es la diferencia del coste computacional del algoritmo incremental respecto al algoritmo por lotes. La tendencia del tiempo de CPU relativo es la misma que se ha observado en los capítulos anteriores respecto al tamaño del conjunto de entrenamiento.

Capítulo 10

Conclusiones

A lo largo de la tesis se han planteado, desarrollado e implementado diferentes algoritmos de clasificación basados en subespacios que presentan distintos enfoques. Además, estos algoritmos se han evaluado usando bases de datos que presentan cierta complejidad.

El primer algoritmo de clasificación propuesto se basa en una extensión del método de vectores comunes discriminantes con kernel, y lo hemos denominado el método de vectores comunes discriminantes extendidos con kernel (RKDCV, *Rough discriminative common vector with Kernels*). Este método es una generalización del método KDCV original, que también puede considerarse como la versión no lineal del método de vectores comunes extendido.

La evaluación empírica de RKDCV se ha llevado a cabo a partir de clasificar cuatro base de datos. Dos, de las cuatro bases de datos son de rostros y presentan cambios de pose, iluminación, expresión y ruido añadido. Las otras dos bases de datos son, una de objetos (con cambios de pose) y otra de dígitos escritos a mano (con cambio de etiquetas). Las simulaciones realizadas fueron de dos tipos. Una respecto al número de muestras de entrenamiento. Y otra respecto a la varianza añadida al subespacio de proyección.

A partir de la validación empírica se ha observado que RKDCV presenta propiedades discriminantes apropiadas para la clasificación de las bases de datos utilizadas, respecto a los métodos DCV, RCV y KDCV. Además que, RKDCV conserva el poder discriminante independientemente de que el tamaño del conjunto de entrenamiento sea menor, igual o mayor que la dimensión de espacio original de las muestras.

Si bien los métodos RCV y KDCV presentan buenas propiedades discriminantes a mayor número de muestras de entrenamiento, el método RKDCV presenta propiedades discriminantes comparables o incluso me-

jores respecto a los métodos RCV y KDCV, utilizando un número menor de muestras de entrenamiento. El tener menos muestras en el conjunto de entrenamiento implica menos esfuerzo computacional, ya que el tamaño del las matrices involucradas para obtener el subespacio discriminante es pequeño.

En cuanto al trabajo futuro en esta linea, se estudiaran las dimensiones involucradas en el espacio nulo extendido del método RKDCV para observar el comportamiento en función de la dimensión de este espacio. Además, se experimentará con otras medidas de distancia entre las características discriminantes de entrenamiento y las de prueba, para observar cómo este parámetro afecta la tasa de clasificación del método.

En esta tesis también se han propuesto algunos algoritmos incrementales motivados por el gran número de aplicaciones que necesitan sistemas de clasificación reentrenables. El propósito de los algoritmos incrementales es que la capacidad discriminante de los métodos se conserve o mejore cuando nueva información se pone a disposición del sistema. Los algoritmos incrementales que se han presentado permiten incorporar a la base de conocimiento nueva información respecto a las clases existentes, así como información de nuevas clases.

Dado que el método DCV presenta características muy apropiadas para tareas de clasificación de datos con alta dimensión, tres algoritmos incrementales diferentes de este método se han propuesto. El primer algoritmo emplea una eigendescomposición incremental (IDCV-EVD). El segundo subespacios diferencia y la ortogonalización de Gram-Schmidt incremental (IDCV-GSO). El tercer algoritmo se basa en subespacios diferencia y ortogonalización (IDCV-O).

La evaluación de los algoritmos incrementales se ha realizado respecto al método original DCV, en relación a la tasa de acierto y, por supuesto, al tiempo de entrenamiento, al añadir nuevas muestras a las clases existentes en el conjunto de entrenamiento, así como al añadir nuevas clases. Para ello se han considerado diferentes bases de datos que abarcan un rango amplio de características como son, bases de datos de imágenes de rostros con diferentes expresiones, poses e iluminación, bases de datos de objetos con poses diferentes, y bases de datos de dígitos escritos a mano.

Desde la validación empírica de los algoritmos incrementales, para nuevas muestras y nuevas clases, se observa que la tasa de clasificación de los algoritmos incrementales es idéntica a la del método original. De este modo, los algoritmos incrementales no degeneran las matrices de proyección en cada reentrenamiento.

Respecto al coste computacional, los algoritmos incrementales presentan un ahorro significativo en relación al coste computacional de los

métodos con aprendizaje por lotes. En cada reentrenamiento el algoritmo incremental necesita menor tiempo computacional que el algoritmo con aprendizaje por lotes. Además, cuanto más grande es la diferencia entre el conjunto de entrenamiento precalculado (inicial) y el tamaño del conjunto de entrenamiento incremental, la diferencia entre los costes computacionales es mucho mayor. Cuando el tamaño del conjunto incremental es pequeño, $N = 1$ o $c_y = 1$, el coste computacional asociado al algoritmo incremental es también pequeño, y esta disminución compensa incluso teniendo en cuenta que se necesita un número mayor de actualizaciones.

El coste computacional de los algoritmos incrementales, además de estar en función del tamaño del conjunto de entrenamiento está en función de la dimensionalidad del espacio original. Si la dimensión es considerablemente mayor al tamaño del conjunto de entrenamiento se observa que la tendencia del tiempo de CPU relativo del algoritmo incremental decrece con M para un valor fijo de N. Esta tendencia observada es de $O(1/M)$.

Desde la validación empírica de los algoritmos IDCV-GSO y IDCV-O (a partir de GSO), se ha observado que aunque teóricamente el coste computacional de los algoritmos es parecido, el método IDCV-O emplea menos tiempo computacional respecto al método IDCV-GSO. Esto se explica parcialmente porque el número teórico de operaciones involucradas en los algoritmos son una aproximación del número real de operaciones.

Por último, se ha presentado un algoritmo incremental del método de vectores comunes discriminantes extendidos, debido a algunas limitaciones del método DCV respecto al tamaño y a la dimensionalidad de espacio original de las muestras.

En el caso del método extendido, la tasa de acierto presenta algunas pequeñas variaciones respecto al algoritmo con aprendizaje por lotes. Estas diferencias se deben a que el algoritmo incremental dispone de solo una parte de la información anterior en cada nuevo reentrenamiento. Para solucionar en parte este problema se ha propuesto una manera de calcular de forma incremental la varianza añadida al subespacio de proyección. Obteniendo que los pequeños cambios que hay entre la tasa de acierto del algoritmo incremental y el algoritmo con aprendizaje por lotes no son relevantes.

En cuanto al coste computacional del algoritmo incremental IRDCV, la tendencia del ahorro computacional es similar a la que presentan los métodos incrementales anteriores. A mayor número de muestras de entrenamiento en el conjunto precalculado, mayor es la diferencia del coste computacional del algoritmo incremental respecto al algoritmo por lotes. La tendencia del tiempo de CPU relativo es de $O(1/M)$ en función del tamaño del conjunto de entrenamiento.

De este modo, los objetivos principales de la tesis que son implementar, evaluar y comparar los diferentes algoritmos de clasificación, basados en subespacios, que se proponen y presentan en este documento, se dan por concluidos.

Como trabajo futuro en la linea de los algoritmos incrementales, se pretende encontrar una relación que sea eficiente en términos de tasa de clasificación y coste computacional, con respecto a la dimensión del espacio de entrada y el número de muestras de entrenamiento, sin llegar a perder la capacidad discriminante de los métodos originales. En cuanto al método IDCV-O, ya que es un método propio, se estudiará el comportamiento del coste computacional respecto a otros métodos de ortogonalización con mayor estabilidad numérica y menor coste computacional. Además, se revisará el estado del arte de los algoritmos incrementales de los métodos kernel, y se intentará desarrollar un algoritmo incremental eficiente del método RKDCV.

A continuación se presentan algunas publicaciones que se han generado a partir del trabajo realizado, así como los proyectos en los que se a participado.

10.1. Publicaciones

1. **K. Diaz-Chito**, F.J. Ferri, W. Díaz-Villanueva. An Empirical Evaluation of Common Vector Based Classification Methods and Some Extensions. SSSPR 08. Structural, Syntactic and Statistical Pattern Recognition. Lect. Not in Comp Sci. Vol 5342, pp. 977-985, 2008. CORE A.
2. F.J. Ferri and **K. Diaz-Chito** and W. Díaz-Villanueva. Using Subspace-based Learning Methods for Medical Drug Design and Characterization. IEEE Intl. Conf on Systems, Man and Cybernetics, SMC 2008. pp.2111-2115, 2008. CORE B.
3. **K. Diaz-Chito**, F.J. Ferri and W. Díaz-Villanueva. Extracción de Características Mediante Vectores Comunes Discriminantes Extendidos con Kernels. Revista Iberoamericana de Inteligencia Artificial, Vol 13, No. 43, pp.1-15, 2009. Journal.
4. **K. Diaz-Chito**, F.J. Ferri. Empirical Evaluation of Class-Updating Incremental Discriminative Common Vector based Face Recognition. II Workshop de Reconocimiento de Formas y Análisis de Imágenes (AERFAI), CEDI2010. pp.125-132, 2010.
5. **K. Diaz-Chito**, F.J. Ferri and W. Díaz-Villanueva. Image Recognition through Incremental Discriminative Common Vectors.

Advanced Concepts for Intelligent Vision Systems - 12th International Conference, ACIVS 2010. pp. 304-311, 2010. CORE B.

6. F.J. Ferri and **K. Diaz-Chito** and W. Díaz Villanueva. Efficient Dimensionality Reduction on Undersampled Problems through Incremental Discriminative Common Vectors. The 10th IEEE International Conference on Data Mining Workshops, ICDM Workshops. pp. 1159-1166, 2010. CORE A.
7. **K. Diaz-Chito**, F.J. Ferri and W. Díaz-Villanueva. Null Space based Image Recognition using Incremental Eigendecomposition. Pattern Recognition and Image Analysis, 5th Iberian Conference, IbPRIA 2011. pp 313-320, 2011. CORE C.

10.2. Proyectos

Esta tesis doctoral Ha sido financiada por el Ministerio de Ciencia e Innovación del Gobierno de España, asociada al proyecto de investigación *Aplicaciones del reconocimiento de formas a la automatización del control de calidad y otros procesos industriales (ARFAI).*

Titulo : Aplicaciones del reconocimiento de formas a la automatización del control de calidad y otros procesos industriales (ARFAI)

IP: Francesc J. Ferri Rabasa

Inv: W. Díaz-Villanueva, J.V. Albert, V. Cerverón, R. Ferrís

Beca: Katerine Diaz-Chito

Tec. Inv:Manuel Yago, Carlos Esteve, Adrian Perez-Suay

REF: DPI2006-15542-C04-04

Fecha de inicio del proyecto: 1/10/2006

Fecha de finalización del proyecto: 30/9/2009

Además se ha participado en otros proyectos de investigación financiados por FEDER y el gobierno de España, como son

Titulo : Multimodal interaction in pattern recognition and computer vision (MIPRCV)

IP Subproyecto UVEG: Francesc J. Ferri Rabasa (IP: Enrique Vidal Ruíz, UPV)

Inv: W. Díaz-Villanueva, J.V. Albert, V. Cerverón, M. Arevalillo

Beca: Adrián Pérez-Suay

Asociados: Katerine Diaz-Chito

REF: CSD2007-00018

Fecha de inicio del proyecto: 1/10/2007

Fecha de finalización del proyecto: 29/11/2012

Titulo : Técnicas interactivas y adaptativas para sistemas automáticos de reconocimiento, aprendizaje y percepción (TIASA)

IP: Francesc J. Ferri Rabasa

Inv: W. Díaz-Villanueva, J.V. Albert, V. Cerverón, Katerine Díaz-Chito, Adrián Pérez-Suay, M. Arevalillo

REF: TIN2009-14205-C04-03

Fecha de inicio del proyecto: 1/1/2010

Fecha de finalización del proyecto: 31/12/2012

Conclusion

Throughout the thesis we have proposed, developed and implemented different algorithms for classification based on subspaces from different perspectives. Moreover, these algorithms have been evaluated on databases with certain complexity.

The first proposed classification algorithm is based on an extension of discriminative common vector method with kernel, called rough discriminative common vector with Kernels (RKDCV). This method is a generalization of the original KDCV method, which can also be seen as the nonlinear version of the rough common vectors method.

The empirical evaluation of RKDCV has been carried out by classifying four different databases. Two of the four databases are composed of face images and presented changes on the pose, lighting, expression and amount of added noise. The other two databases are, one of objects (with pose changes) and other of handwritten digits (with change of labels). The performed simulations were of two types: regarding the number of training samples, and about the variability added to the projection subspace.

From the empirical validation of the RKDCV method, we observed that RKDCV presents, in comparison with the DCV, RCV and KDCV methods, discriminant properties suitable for the classification of the databases used. Besides that, the RKDCV method preserves the discriminative property regardless of the fact that the training set size is smaller or bigger than the dimension of the original sample space.

Even though both the RCV and KDCV methods show good discriminant properties which increase with the number of training samples, the RKDCV method presents comparable or even better discriminant properties than the RCV and KDCV methods, using fewer training samples. Having less samples in the training set implies a lower computational cost, since the size of the matrices involved to obtain the discriminant subspace is small.

As future work in this line, we will study the dimensions involved in the extended nullspace of the RKDCV method in order to observe its behavior depending on the size of this space. In addition, we will perform experiments with other distance measurements between the discriminat feature of training and test, in order to observe how this parameter affects the accuracy of the method.

In the thesis we have also proposed several incremental algorithms motivated by the large number of applications that require retrainable classification systems. The purpose of incremental algorithms is that the discriminant properties of methods are maintained or improved when new information becomes available to the system. Incremental algorithms which have been proposed can incorporate new information to the knowledge base regarding both existing and new classes.

Since the DCV method has suitable features for classification tasks of high dimensional data, three different incremental algorithms based on this methodology have been proposed. The first algorithm uses an incremental eigendecomposition (IDCV-EVD). The second one uses difference subspace and incremental Gram-Schmidt orthogonalization (IDCV-GSO). Finally, the third algorithm is based on difference subspace and orthogonalization (IDCV-O).

The evaluation of the incremental algorithms were done by against the DCV original method, computing the accuracy and, of course, the training time by adding new samples to existing classes, and adding new classes to the training set. Different datasets that cover a wide range of situations have been considered such as facial image datasets with different expressions, poses and lighting, databases of objects with different poses, and handwritten digits databases. In the light of the empirical validation of the incremental algorithms, for new samples and new classes, we can conclude that the accuracy of the incremental algorithms is identical to the original method. Thus, the incremental algorithms do not degenerate the projection matrices at each retraining.

Regarding the computational cost, incremental algorithms show significant savings of relative computational cost in comparison with batch learning methods. At each retraining, the incremental algorithm needs less computational time that the equivalent batch learning algorithm. In addition, the larger the difference between the size of the precalculated training set (initial) and the size of the incremental training set, the bigger the difference between the computational cost. When the incremental set size is small, $N = 1$ or $c_y = 1$, the computational cost associated with the incremental algorithm is also small, and this decrease compensates the fact that a large number of updates are needed.

The computational cost of the incremental algorithms is a function of both the training set size and the original space dimensionality. If

the dimension is considerably bigger than the training set size, it can be observed that the relative CPU time of the incremental algorithm decreases with M for a fixed value of N. This decrement is $O(1/M)$.

From the experimental validation of the algorithms IDCV-GSO and IDCV-O, we noted that, although the theoretical complexities of the algorithms are similar, the IDCV-O method uses less computational time compared to IDCV-GSO method. This is partly explained because the theoretical number of operations involved in the algorithms is an approximation of the real number of operations performed.

Finally, we have presented an incremental algorithm of the rough discriminative common vector method, that mitigates some limitations of the DCV method regarding the size and dimension of original space of the samples.

The accuracy of the IRDCV present few small variations with regard to the batch learning algorithm. These differences are due to the fact the incremental algorithm has available only a part of the information at each new retraining. In order to partially solve this problem we have proposed a new way to calculate the variance added to the projection subspace. As conclusion, we have observed that small differences between the accuracies of the incremental algorithm and of the batch learning algorithm are not relevant.

Regarding the computational cost of IRDCV, the computational savings trend is similar to the one of the previous incremental methods. The more the number of train samples in the precalculated set, the bigger the difference between the computational cost of the incremental algorithm and the batch algorithm. The trend of relative CPU time is $O(1/M)$ as a function of the training set size.

Therefore, the main objectives of the thesis, which were to implement, evaluate and compare the different classification algorithms based on subspace proposed in this thesis, are considered as completely fulfilled.

As future work on the incremental algorithms line of research, we aim to find an efficient relationship in terms of accuracy and computational cost, according to the input space dimension and the number of training samples, and without losing the discriminative properties of original methods. About the IDCV-O method, we plan to study its behavior in terms of computational cost in comparison with other orthogonalization strategies that have shown, in principle, a better numerical stability and lower computational cost. In addition, we review the state of the art of the incremental algorithms of the kernel methods, and we try to develop an efficient incremental algorithm of the RKDCV method.

Apéndice A

Conceptos

Este capítulo define varios conceptos de estadística, álgebra lineal y representación de datos.

A.1. Conceptos Fundamentales

Distancia euclídea

Una parte fundamental relacionada con la presentación de datos es elegir una medida de distancia adecuada entre dos puntos. En un espacio d-dimensional cada punto a_1 queda definido por un vector d-dimensional, la distancia euclídea entre dos puntos p_1 y p_2 en un espacio $\mathbb{R}^d$ es:

$$d_{euclidea} = \sqrt{\sum_{i=1}^{d}(p_{1_i} - p_{2_i})^2}$$

que puede considerarse como la extensión del teorema de Pitágoras a un espacio d-dimensional.

Espacio de características

En reconocimiento de patrones, un espacio de características es un espacio donde cada muestra p se representa como un punto en un espacio d-dimensional. Esta dimensión está determinada por el número de características que describen la muestra. Objetos similares se encontrarán cercanos en el espacio de características, donde es posible realizar algún tipo de clasificación o agrupamiento de los mismos.

Subespacio diferencia

Dado un conjunto de datos linealmente independientes $\mathbf{X} \in \mathbb{R}^{d \times M}$ con c clases y m_j muestras en la clase j, el espacio de características puede dividirse en c subespacios diferencia e indiferencia.

El subespacio diferencia de cada clase $\mathcal{B}_j$ es generado por $(m_j - 1)$ vectores diferencia dados por $\mathrm{x}_j^i - \mathrm{x}_j^1,\ i = 2, \ldots, m_j,\ j = 1, 2, \ldots .c$. La primera muestra de cada clase es tomada como vector sustraendo y su elección es arbitraria. La suma de los c subespacios diferencia, $\mathcal{B}_j$, conforman el subespacio diferencia, $\mathcal{B}_X \subseteq \mathbb{R}^{d \times (M-c)}$, asociado a $\mathbf{X}$ [44].

Método de los k-vecinos mas próximos

El método de los k-vecinos mas próximos (K-NN) [36] es un clasificador supervisado que emplea alguna medida de distancia (normalmente es la distancia euclídea) entre la muestra de test y k muestras del conjunto de entrenamiento, en un espacio de características.

El método estima el valor de la probabilidad a posteriori de que un elemento x pertenezca a la clase j. El elemento x se asigna a la clase j si ésta es la clase más frecuente entre las k muestras de entrenamiento más cercanas.

Validación cruzada con $k-$bloques

La validación cruzada con $k-$bloques ($k-$*fold cross validation*) es uno, entre varios, de los métodos más populares que se utiliza para validar un modelo generado a partir de un conjunto de datos o muestra. En este método, el conjunto de datos original se divide en k submuestras (folds) de (aproximadamente) igual tamaño. La submuestra i, se utiliza para validar los datos (conjunto de prueba), y las restantes $(k-1)$ submuestras, son empleadas en el entrenamiento [101]. El proceso de validación se repite k veces (folds) con cada una de las k submuestras, usando una para validar los datos. Los k resultados se promedian para obtener un único resultado.

A.2. Propiedades de los valores y vectores propios

En diversos campos de la ingeniería y las matemáticas surge el problema de calcular los valores escalares λ, y los vectores $v \neq 0$, que satisfacen

$$\mathrm{A}v = \lambda v, \tag{A.1}$$

para la matriz cuadrada A.

Algunos de estos campos de aplicación son:
- Ecuaciones diferenciales
- Estabilidad de sistemas lineales
- Diagonalización de matrices, ...

El problema planteado por la ecuación (A.1) tiene solución si se reescribe como:

$$(\mathrm{A} - \lambda I)v = 0$$

De este modo, el problema se transforma en el ya conocido sistema lineal homogéneo $Bv = 0$, el cual tiene solución única $v = 0$ cuando $|B| \neq 0$, que es el caso que no nos interesa.

El número λ se dice valor propio de A si y sólo si

$$|\mathrm{A} - \lambda I| = 0$$

que es la ecuación característica de la matriz A.

Si λ es un valor propio de A, y si v es el vector no nulo tal que $\mathrm{A}v = \lambda v$, entonces v se dice vector propio de A correspondiente al valor propio λ.

El conjunto de vectores propios v que satisfacen la ecuación (A.1) forman un subespacio de A llamado **espacio propio** correspondiente a los valores propios λ [102].

Algunas propiedades de los valores y vectores propios son:

1. n vectores propios distintos correspondientes a n valores propios distintos de multiplicidad uno, constituyen una base del espacio vectorial para la matriz A.
2. Si λ_k son los valores propios de A, λ_k^n son los valores propios de A^n. Además, A y A^n tienen los mismos vectores propios. En particular, si $\lambda \neq 0$ es valor propio de A, entonces $1/\lambda$ es valor propio de A^{-1}.
3. El producto de los n valores propios de una matriz es igual a su determinante.
$$|\mathrm{A}| = \prod_{i=1}^{n} \lambda_i.$$

4. La suma de los n valores propios de una matriz es igual a su traza.

$$tr(\mathrm{A}) = \sum_{i=1}^{n} \lambda_i.$$

5. Dos matrices semejantes tienen la misma ecuación característica y consecuentemente los mismos valores propios con igual multiplicidad.
6. Una matriz triangular tiene como valores propios los elementos de la diagonal principal.
7. Una matriz simétrica de coeficientes reales posee valores propios reales, y los vectores propios asociados a valores propios distintos son ortogonales.

A.3. Subespacios

Operaciones entre subespacios

Sean $\mathcal{S}_1$ y $\mathcal{S}_2$ dos subespacios vectoriales de $\mathcal{S}$, se definen las siguientes operaciones:

Unión: $\mathcal{S}_1 \cup \mathcal{S}_2 = [x \in \mathcal{S} / x \in \mathcal{S}_1 \vee x \in \mathcal{S}_2]$
en la gran mayoría de los casos la unión de dos subespacios no es un subespacio de $\mathcal{S}$, porque no se cumple con la ley de composición interna. Sí pertenece de forma segura la unión a $\mathcal{S}$ en los casos en que $\mathcal{S}_1$ este contenido en $\mathcal{S}_2$ o viceversa.

Intersección: $\mathcal{S}_1 \cap \mathcal{S}_2 = [x \in \mathcal{S} / X \in \mathcal{S}_1 \wedge x \in \mathcal{S}_2]$
la intersección de un número cualquiera de subespacios vectoriales de un espacio vectorial $\mathcal{S}$ es, a su vez, un subespacio vectorial de $\mathcal{S}$.

Suma: $\mathcal{S}_1 + \mathcal{S}_2 = [x \in \mathcal{S} / x = (x_1 + x_2) \wedge x_1 \in \mathcal{S}_1 \wedge x_2 \in \mathcal{S}_2]$
la suma de dos subespacios es un subespacio de $\mathcal{S}$.

Suma directa: si la intersección entre $\mathcal{S}_1$ y $\mathcal{S}_2$ es el subespacio trivial (es decir, el vector nulo), entonces a la suma se la llama **suma directa**. Es decir que si $\mathcal{S}_1 \cap \mathcal{S}_2 = \{0\} \Rightarrow \mathcal{S}_1 \oplus \mathcal{S}_2$.

Subespacios ortogonales

Dos subespacios vectoriales $\mathcal{S}_1$ y $\mathcal{S}_2$ de un espacio vectorial $\mathcal{S}$ se dicen ortogonales cuando todos los vectores de uno de ellos son ortogonales a

todos los vectores del otro:

$$x \cdot y = 0 \text{ para cualesquiera } x \in \mathcal{S}_1,\ y \in \mathcal{S}_2$$

Subespacio complementario ortogonal

Dado un espacio vectorial $\mathcal{S}$ y un subconjunto $\mathcal{S}_1 \subset \mathcal{S}$, se llama subespacio ortogonal de $\mathcal{S}_1$ a:

$$\mathcal{S}_1^{\perp} = \{x \in \mathcal{S} | x \cdot v = 0 \text{ para todo } v \in \mathcal{S}_1\},$$

Se cumple que:

1. $\mathcal{S}_1^{\perp}$ es un espacio vectorial.
2. $\mathcal{S}_1 \subset \mathcal{S} \ \Rightarrow\ \mathcal{S}_1^{\perp} \subset \mathcal{S}^{\perp}$.
3. Si $\mathcal{S}_1$ es un subespacio vectorial, $\mathcal{S}_1$ y $\mathcal{S}_1^{\perp}$ son complementarios.

A.4. Eigendescomposición desde una matriz más pequeña

En el área del reconocimiento de imágenes la dimensionalidad de las imágenes es mucho mayor que el número de muestras disponibles para el entrenamiento. Este hecho conlleva que las matrices de dispersión estén definidas en un espacio de dimensión muy grande $\mathbb{R}^{d \times d}$, lo cual implica que su eigendescomposición (**EVD**) sea computacionalmente costosa. Una solución a este problema se muestra en [86].

Sea $\mathcal{S} = \mathrm{A}\mathrm{A}^T \in \mathbb{R}^{d \times d}$ alguna matriz de dispersión calculada a partir del conjunto de entrenamiento $\mathbf{X} \in \mathbb{R}^{d \times M}$, con M muestras de dimensión d, $d > M$ y $\mathrm{A} \in \mathbb{R}^{d \times M}$. En los casos que nos interesan siempre se podrá descomponer $\mathcal{S}$ como $(\mathrm{A}\mathrm{A}^T)$, puesto que la matriz $\mathcal{S}$ es semidefinida positiva.

La solución propuesta por Murakami y Kumar en [86] considera los vectores propios de una matriz de menor tamaño, como es $(\mathrm{A}^T\mathrm{A}) \in \mathbb{R}^{M \times M}$. Un vector $v_k \in \mathbb{R}^M$ no nulo, es un vector propio de $(\mathrm{A}^T\mathrm{A})$ asociado al valor propio $\lambda_k \neq 0 \in \mathbb{R}$ si:

$$\mathrm{A}^T\mathrm{A}v_k = \lambda_k v_k \tag{A.2}$$

Multiplicando ambos lados de la ecuación (A.2) por A, se obtiene

$$\mathrm{A}\mathrm{A}^T\mathrm{A}v_k = \lambda_k \mathrm{A}v_k$$

de donde, se deduce que $(\mathrm{A}v_k)$ es el vector propio u_k de $(\mathrm{A}\mathrm{A}^T)$ asociado a λ_k.

De este modo, los vectores propios u_k de $(\mathrm{A}\mathrm{A}^T)$ se obtienen a partir de los vectores propios v_k de $(\mathrm{A}^T\mathrm{A})$. Es decir,

$$u_k = \mathrm{A}v_k, \quad k = 1, 2, \ldots, M. \tag{A.3}$$

donde los valores propios de $(\mathrm{A}^T\mathrm{A})$ son los mismos para $(\mathrm{A}\mathrm{A}^T)$.

El coste computacional mediante este procedimiento es de $O(\ell(M^3)+ dM^2)$. Mientras que el coste computacional del método directo es de $O(\ell(d^3))$. ℓ es el número de iteraciones requeridas para que el algoritmo de eigendescomposición converja.

A.5. Criterio de Fisher

Una de las medidas más utilizadas en reconocimiento de patrones para cuantificar la discriminación entre diferentes clases de un conjunto de datos es el criterio de Fisher, definido como:

$$\mathrm{W} = \underset{\|\mathrm{w}\|=1}{argmax} \frac{|\mathrm{W}^T S_B \mathrm{W}|}{|\mathrm{W}^T S_W \mathrm{W}|} \tag{A.4}$$

$S_B \in \mathbb{R}^{d\times d}$ y $S_W \in \mathbb{R}^{d\times d}$ son las matrices de dispersión inter e intra clase del conjunto de entrenamiento, respectivamente. W es la matriz de transformación que maximiza (A.4).

El criterio de Fisher es máximo cuando los vectores de la matriz de transformación, W, son los vectores propios de $(S_W^{-1} S_B)$ [30, 15, 39, 12].

En tareas de reconocimiento de patrones, este criterio no puede aplicarse directamente en casos donde la dimensión del espacio de muestras es mayor al número de muestras en el conjunto de entrenamiento. Este problema se conoce como el problema del tamaño de muestras pequeño, *small sample size problem* [39].

Muchos métodos se han propuesto para resolver este problema, algunos de ellos están en [108, 54, 77, 76, 24, 11, 131, 23, 60, 61]. Liu et al. [77, 76] modifican el criterio de fisher convencional por:

$$J(\mathrm{W}) = \frac{|W^T S_B W|}{|W^T S_T W|} \tag{A.5}$$

nombrado como el criterio de Fisher modificado. Este criterio modificado emplea como divisor la matriz de dispersión total, $S_T = S_B + S_W$, en lugar de S_W [77, 76].

Chen et al. [23] formulan una solución donde (A.5) es máximo, a la vez que S_W es minimizada mientras S_B es maximizada. Ellos primero

proyectan el conjunto de entrenamiento en el espacio nulo $\mathcal{N}(S_W)$. Luego, los vectores propios correspondientes a los valores propios no nulos de la matriz de dispersión intra-clase, en el espacio transformado, son seleccionados como los vectores de proyección que conforman W.

De lo anterior, el criterio en (A.5) puede ser reescrito como:

$$\begin{aligned} \mathrm{W} &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_B \mathrm{W}| \\ &= \underset{|\mathrm{W}^T S_W \mathrm{W}|=0}{argmax} |\mathrm{W}^T S_T \mathrm{W}| \end{aligned} \tag{A.6}$$

El criterio dado en (A.6), es conocido como el criterio de Fisher basado en el espacio nulo [23, 13, 61].

A.6. Funciones kernel

Una función real definida sobre el producto cartesiano del espacio de entrada $\mathbb{R}^d$, que a cada pareja de elementos del espacio de entrada le hace corresponder el producto escalar en $\mathcal{F}$ de sus respectivas imágenes por la función ϕ, se llama función kernel.

$$k(\mathrm{x}_i, \mathrm{x}_j) = \langle \phi(\mathrm{x}_i), \phi(\mathrm{x}_j) \rangle$$

Son kernels todas aquellas funciones $k(\mathrm{x}_i, \mathrm{x}_j)$ que verifican el teorema de Mercer [99], es decir, para las cuales

$$\int_{\mathrm{x}_i, \mathrm{x}_j} k(\mathrm{x}_i, \mathrm{x}_j) g(\mathrm{x}_i) g(\mathrm{x}_j) d\mathrm{x}_i d\mathrm{x}_j > 0$$

para toda función $g(\cdot)$ de cuadrado integrable.

Entre las funciones kernel más comunes están:

Lineal	$k(\mathrm{x}_i, \mathrm{x}_j) = \langle \mathrm{x}_i, \mathrm{x}_j \rangle$
Polinomial	$k(\mathrm{x}_i, \mathrm{x}_j) = \langle \mathrm{x}_i, \mathrm{x}_j \rangle^{\eta}$
Gaussiano	$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\|\mathrm{x}_i - \mathrm{x}_j\|^2}{2\sigma^2}\right)$
Sigmoide	$k(\mathrm{x}_i, \mathrm{x}_j) = \tanh(\tau \langle \mathrm{x}_i, \mathrm{x}_j \rangle + \vartheta)$

donde η, σ, , τ y ϑ son especificadas a priori por el usuario.

Apéndice B

Descomposición de la matriz de dispersión intra-clase

Sean $\mathbf{X} \in \mathbb{R}^{d\times M}$ e $\mathbf{Y} \in \mathbb{R}^{d\times N}$ conjuntos de entrenamiento con las mismas clases. Donde $M = \sum_{j=1}^{c} m_j$, $N = \sum_{j=1}^{c} n_j$, x_j es la media de la clase j en $\mathbf{X}$ e, y_j es la media de la clase j en $\mathbf{Y}$.

Las matrices de dispersión intra-clase de $\mathbf{X}$ e $\mathbf{Y}$ están dadas por

$$\begin{aligned} S_X^w &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T \\ S_Y^w &= \sum_{j=1}^{c}\sum_{i=1}^{n_j}(\mathrm{y}_j^i - \mathrm{y}_j)(\mathrm{y}_j^i - \mathrm{y}_j)^T \end{aligned}$$

Sí $\mathbf{Z}$ es el conjunto de entrenamiento compuesto por $\mathbf{X}$ e $\mathbf{Y}$. La matriz de dispersión intra-clase de $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}]$, se define como

$$S_Z^w = \sum_{j=1}^{c}\sum_{i=1}^{(m_j+n_j)}(\mathrm{z}_j^i - \mathrm{z}_j)(\mathrm{z}_j^i - \mathrm{z}_j)^T \tag{B.1}$$

donde $\mathrm{z}_j = \left(\frac{m_j}{(mj+nj)}\mathrm{x}_j + \frac{n_j}{(mj+nj)}\mathrm{y}_j\right)$ es la media de la clase j en $\mathbf{Z}$.

La ecuación (B.1) se puede descomponer como:

$$S_Z^w = \underbrace{\sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{z}_j)(\mathrm{x}_j^i - \mathrm{z}_j)^T}_{①} + \underbrace{\sum_{j=1}^{c}\sum_{i=1}^{n_j}(\mathrm{y}_j^i - \mathrm{z}_j)(\mathrm{y}_j^i - \mathrm{z}_j)^T}_{②}$$

Donde ① y ②, se simplifican como

$$
\begin{aligned}
①.\ \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{z}_j)^T &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i-\mathrm{x}_j+\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{x}_j+\mathrm{x}_j-\mathrm{z}_j)^T \\
&= \sum_{j=1}^{c}\sum_{i=1}^{m_j}[\underbrace{(\mathrm{x}_j^i-\mathrm{x}_j)(\mathrm{x}_j^i-\mathrm{x}_j)^T}_{} \\
&\quad \underbrace{+(\mathrm{x}_j^i-\mathrm{x}_j)(\mathrm{x}_j-\mathrm{z}_j)^T+(\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{x}_j)^T}_{} \\
&\quad \underbrace{+(\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j-\mathrm{z}_j)^T}_{}] \\
&= S_X^w+0+\sum_{j=1}^{c}\frac{m_j n_j^2}{(m_j+n_j)^2}(\mathrm{x}_j-\mathrm{y}_j)(\mathrm{x}_j-\mathrm{y}_j)^T \\
②.\ \sum_{j=1}^{c}\sum_{i=1}^{n_j}(\mathrm{y}_j^i-\mathrm{z}_j)(\mathrm{y}_j^i-\mathrm{z}_j)^T &= \sum_{j=1}^{c}\sum_{i=1}^{n_j}(\mathrm{y}_j^i-\mathrm{y}_j+\mathrm{y}_j-\mathrm{z}_j)(\mathrm{y}_j^i-\mathrm{y}_j+\mathrm{y}_j-\mathrm{z}_j)^T \\
&= \sum_{j=1}^{c}\sum_{i=1}^{n_j}[\underbrace{(\mathrm{y}_j^i-\mathrm{y}_j)(\mathrm{y}_j^i-\mathrm{y}_j)^T}_{} \\
&\quad \underbrace{+(\mathrm{y}_j^i-\mathrm{y}_j)(\mathrm{y}_j-\mathrm{z}_j)^T+(\mathrm{y}_j-\mathrm{z}_j)(\mathrm{y}_j^i-\mathrm{y}_j)^T}_{} \\
&\quad \underbrace{+(\mathrm{y}_j-\mathrm{z}_j)(\mathrm{y}_j-\mathrm{z}_j)^T}_{}] \\
&= S_Y^w+0+\sum_{j=1}^{c}\frac{m_j^2 n_j}{(m_j+n_j)^2}(\mathrm{y}_j-\mathrm{x}_j)(\mathrm{y}_j-\mathrm{x}_j)^T
\end{aligned}
$$

De este modo, S_Z^w se redefine

$$
\begin{aligned}
S_Z^w &= \underbrace{S_X^w}+\underbrace{S_Y^w} \\
&\quad \underbrace{+\sum_{j=1}^{c}\frac{m_j n_j^2}{(m_j+n_j)^2}(\mathrm{x}_j-\mathrm{y}_j)(\mathrm{x}_j-\mathrm{y}_j)^T+\sum_{j=1}^{c}\frac{m_j^2 n_j}{(m_j+n_j)^2}(\mathrm{y}_j-\mathrm{x}_j)(\mathrm{y}_j-\mathrm{x}_j)^T}_{} \\
&= S_X^w+S_Y^w+\sum_{j=1}^{c}\frac{m_j\, n_j}{(m_j+n_j)}(\mathrm{x}_j-\mathrm{y}_j)(\mathrm{x}_j-\mathrm{y}_j)^T \qquad \text{(B.2)}
\end{aligned}
$$

Con $S=\mathrm{J}\mathrm{J}^T$, $\mathrm{J}=[\mathrm{J}_1\ \mathrm{J}_2\ldots\mathrm{J}_c]$ y $\mathrm{J}_j=\sqrt{\frac{m_j n_j}{(m_j+n_j)}}(\mathrm{x}_j-\mathrm{y}_j)$ para todo $j=1,\ldots,c$.

$$S_Z^w=S_X^w+S_Y^w+S$$

Por tanto, S_Z^w se define en términos de las matrices de dispersión intra-clase S_X^w y S_Y^w, además de un término S que relaciona las medias del conjunto de entrenamiento anterior y el de actualización.

Si $\mathbf{Y}\in\mathbb{R}^{d\times N}$ contiene una muestra nueva de cada clase j, la matriz de dispersión intra-clase de $\mathbf{Z}=[\mathbf{X}\ \mathbf{Y}]$ se puede escribir como:

$$S_Z^w=\underbrace{\sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{z}_j)^T}_{③}+\underbrace{\sum_{j=1}^{c}(\mathrm{y}_j^1-\mathrm{z}_j)(\mathrm{y}_j^1-\mathrm{z}_j)^T}_{④}$$

Simplificando

$$
\begin{aligned}
③.\ \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{z}_j)^T &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i-\mathrm{x}_j+\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{x}_j+\mathrm{x}_j-\mathrm{z}_j)^T \\
&= \sum_{j=1}^{c}\sum_{i=1}^{m_j}[\underbrace{(\mathrm{x}_j^i-\mathrm{x}_j)(\mathrm{x}_j^i-\mathrm{x}_j)^T}_{} \\
&\quad +\underbrace{(\mathrm{x}_j^i-\mathrm{x}_j)(\mathrm{x}_j-\mathrm{z}_j)^T+(\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j^i-\mathrm{x}_j)^T}_{} \\
&\quad +\underbrace{(\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j-\mathrm{z}_j)^T}_{}] \\
&\quad = S_X^w+0+\sum_{j=1}^{c}(\mathrm{x}_j-\mathrm{z}_j)(\mathrm{x}_j-\mathrm{z}_j)^T
\end{aligned}
$$

El vector media de cada clase en $\mathbf{Z}$ se define por:

$$\mathrm{z}_j=\frac{m_j}{(m_j+1)}\,\mathrm{x}_j+\frac{1}{(m_j+1)}\,\mathrm{y}_j^1$$

donde $(\mathrm{x}_j-\mathrm{z}_j)$ y $(\mathrm{y}_j^1-\mathrm{z}_j)$, son

$$
\begin{aligned}
\mathrm{x}_j-\mathrm{z}_j &= \mathrm{x}_j-\left(\frac{m_j}{(m_j+1)}\,\mathrm{x}_j+\frac{1}{(m_j+1)}\,\mathrm{y}_j^1\right) \\
&= \frac{1}{(m_j+1)}\,(\mathrm{x}_j-\mathrm{y}_j^1) \\
\mathrm{y}_j^1-\mathrm{z}_j &= \mathrm{y}_j^1-\left(\frac{m_j}{(m_j+1)}\,\mathrm{x}_j+\frac{1}{(m_j+1)}\,\mathrm{y}_j^1\right) \\
&= \frac{m_j}{(m_j+1)}\,(\mathrm{y}_j^1-\mathrm{x}_j)
\end{aligned}
$$

De este modo, S_Z^w se reescribe como

$$
\begin{aligned}
S_Z^w &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}(\mathrm{x}_j^i-\mathrm{x}_j)(\mathrm{x}_j^i-\mathrm{x}_j)^T \\
&\quad +\sum_{j=1}^{c}\frac{m_j}{(m_j+1)^2}\,(\mathrm{x}_j-\mathrm{y}_j^1)(\mathrm{x}_j-\mathrm{y}_j^1)^T+\sum_{j=1}^{c}\frac{m_j^2}{(m_j+1)^2}\,(\mathrm{y}_j^1-\mathrm{x}_j)(\mathrm{y}_j^1-\mathrm{x}_j)^T \\
&= S_X^w+\sum_{j=1}^{c}\frac{m_j}{(m_j+1)}\,(\mathrm{x}_j-\mathrm{y}_j^1)(\mathrm{x}_j-\mathrm{y}_j^1)^T \qquad \text{(B.3)} \\
&= S_X^w+\mathrm{J}\mathrm{J}^T
\end{aligned}
$$

donde $\mathrm{J}=[\mathrm{J}_1\ \ldots\ \mathrm{J}_c]$ para todo $\mathrm{J}_j=\sqrt{\frac{m_j}{(m_j+1)}}\,(\mathrm{x}_j-\mathrm{y}_j^1)$.

Si las clases de $\mathbf{Y}\in\mathbb{R}^{d\times N}$ difieren de las de $\mathbf{X}$. El número de clases total en $\mathbf{Z}$ está dado por $c_z=c_x+c_y$. Y el número de muestras total es $o=\sum_{j=1}^{c_x}m_j+\sum_{j=1}^{c_y}n_j$.

En este caso la matriz $S_{\mathbf{Z}}^w$ se define como

$$\begin{aligned} S_Z^w &= \sum_{j=1}^{c_z}\sum_{i=1}^{o}(\mathrm{z}_j^i - \mathrm{z}_j)(\mathrm{z}_j^i - \mathrm{z}_j)^T \\ &= \sum_{j=1}^{c_x}\sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T + \sum_{j=1}^{c_y}\sum_{i=1}^{n_j}(\mathrm{y}_j^i - \mathrm{y}_j)(\mathrm{y}_j^i - \mathrm{y}_j)^T \\ &= S_X^w + S_Y^w \end{aligned}$$

En síntesis, la matriz de dispersión intra-clase del conjunto de entrenamiento $\mathbf{Z} = [\mathbf{X}\ \mathbf{Y}]$ se ha descompuesto de tres formas diferentes según las características de $\mathbf{Y}$.

1. $\mathbf{X}$ e $\mathbf{Y}$ tienen las mismas clases

$$\begin{aligned} S_Z^w &= S_X^w + S_Y^w + \sum_{j=1}^{c}\frac{m_j\, n_j}{(m_j + n_j)}(\mathrm{x}_j - \mathrm{y}_j)(\mathrm{x}_j - \mathrm{y}_j)^T \\ &= S_Z^w = S_X^w + S_Y^w + \mathrm{J}\mathrm{J}^T \end{aligned}$$

   donde $\mathrm{J} = [\mathrm{J}_1\ \mathrm{J}_2 \ldots \mathrm{J}_c]$ y $\mathrm{J}_j = \sqrt{\frac{m_j n_j}{(m_j+n_j)}}(\mathrm{x}_j - \mathrm{y}_j)\ \forall\ j = 1, \ldots, c.$

2. $\mathbf{Y}$ tiene una muestra por clase

$$\begin{aligned} S_Z^w &= S_X^w + \sum_{j=1}^{c}\frac{m_j}{(m_j+1)}\,(\mathrm{x}_j - \mathrm{y}_j^1)(\mathrm{x}_j - \mathrm{y}_j^1)^T \\ &= S_X^w + \mathrm{J}\mathrm{J}^T \end{aligned}$$

   donde $\mathrm{J} = [\mathrm{J}_1\ \ldots\ \mathrm{J}_c]$ y $\mathrm{J}_j = \sqrt{\frac{m_j}{(m_j+1)}}\,(\mathrm{x}_j - \mathrm{y}_j^1)\ \forall\ j = 1, \ldots, c.$

3. $\mathbf{X}$ e $\mathbf{Y}$ tienen clases diferentes

$$S_Z^w = S_X^w + S_Y^w$$

Apéndice C

Teoremas

En este apéndice se presentan algunos teoremas empleados a lo largo del documento.

El teorema 1 muestra que los vectores propios asociados a los valores propios más grandes de la matriz de dispersión total, son los vectores de proyección óptimos para calcular las componentes principales del conjunto de entrenamiento.

El teorema 2 muestra que, la proyección de cualquier muestra del conjunto de entrenamiento de la clase j en el espacio nulo de su matriz de dispersión intra-clase genera un único vector.

El teorema 3 demuestra que, si $\mathcal{S}_1 + \mathcal{S}_2$ es un subespacio suma directa de $\mathcal{S}$ entonces, una base de la suma directa es la unión de las bases de $\mathcal{S}_1$ y $\mathcal{S}_2$.

El teorema 4 afirma que, la intersección de subespacios definidos por matrices de dispersión semidefinidas positivas es igual al subespacio definido por la suma de las matrices de dispersión.

El teorema 5 prueba que, los vectores comunes calculados a partir de subespacios diferencia son independientes de la selección del vector sustraendo.

El teorema 6 muestra que, el subespacio nulo de la matriz de dispersión intra-clase y el subespacio indiferencia son el mismo subespacio. A si mismo, el subespacio rango de la matriz de dispersión intra-clase es idéntico al subespacio diferencia del mismo conjunto de entrenamiento.

El teorema 7 muestra que las matrices de proyección a los subespacios $\mathcal{N}(S_T)$ y $\mathcal{R}(S_W)$ conmutan.

Finalmente, el teorema 8 muestra que el subespacio diferencia de $\mathbf{Y}$ respecto a $\mathbf{X}$ se puede calcular a partir de cualquier muestra de $\mathbf{X}$ que pertenezca a la misma clase en $\mathbf{Y}$.

Teorema 1 *El conjunto de vectores que maximizan el criterio:*

$$\mathrm{W}_{pca} = \underset{\|\mathrm{w}\|=1}{argmax} |\mathrm{W}^T S_T \mathrm{W}| \tag{C.1}$$

son los vectores propios asociados a los valores propios no nulos de la matriz de dispersión total, S_T, del conjunto de entrenamiento ([4], pág. 109).

Prueba. Para un vector w, el problema consiste en maximizar ($\mathrm{w}^T S_T \mathrm{w}$), donde $\mathrm{w}^T\mathrm{w} = 1$. El método habitual para maximizar una función sujeta a restricciones es el método de los multiplicadores de Lagrange. Derivando el Lagrangiano correspondiente a la ecuación (C.1) respecto a w

$$L(\mathrm{w}, \lambda) = \mathrm{w}^T S_T \mathrm{w} - \lambda(\mathrm{w}^T\mathrm{w} - 1)$$

e igualando la derivada a 0, tenemos

$$\begin{aligned} \frac{\partial L(\mathrm{w}, \lambda)}{\partial \mathrm{w}} &= 2S_T\mathrm{w} - 2\lambda\mathrm{w} = 0 \\ (S_T - \lambda I)\mathrm{w} &= 0 \end{aligned} \tag{C.2}$$

donde (C.2) es un sistema lineal de ecuaciones.

Para que el sistema tenga una solución distinta de 0 la matriz $(S_T - \lambda I)$ tiene que ser singular. Esto implica que el determinante debe ser igual a cero:

$$|S_T - \lambda I| = 0$$

De las propiedades de los valores y vectores propios (apéndice A.2), λ es un valor propio de S_T.

Desde la ecuación (C.2)

$$S_T\mathrm{w} = \lambda I\mathrm{w}$$

tal que, $\mathrm{w}^T S_T\mathrm{w} = \mathrm{w}^T\lambda I\mathrm{w} = \lambda\mathrm{w}^T\mathrm{w}$.

Usando el hecho que $\mathrm{w}^T\mathrm{w} = 1$

$$\mathrm{w}^T S_T\mathrm{w} = \lambda \tag{C.3}$$

se obtiene que la dispersión es máxima cuando w es el vector propio asociado al valor propio más grande de S_T.

Finalmente $\mathrm{W}_{pca} = [\mathrm{w}_1\, \mathrm{w}_2 \ldots \mathrm{w}_r] \in \mathbb{R}^{d\times r}$, son los vectores propios asociados a los valores propios no nulos (organizados en orden decreciente) de S_T ([4], pág. 109).

Teorema 2 *Sea la matriz $\overline{U} \in \mathbb{R}^{d\times n}$, una matriz compuesta por n vectores ortonormales que generan el espacio nulo de la matriz de dispersión intra-clase, S_W, de un conjunto de entrenamiento $\mathbf{X} \in \mathbb{R}^{d\times M}$.*

$M = \sum_{j=1}^{c} m_j$ es el número de muestras de entrenamiento y c es el número de clases.

La proyección de cualquier muestra, x_j^i, de la clase j en el espacio nulo de S_W, genera un único vector x_{cv}^j,

$$\mathrm{x}_{cv}^j = \overline{\mathrm{U}}\,\overline{\mathrm{U}}^T \mathrm{x}_j^i, \qquad i = 1, \ldots, m_j$$

Prueba. Por definición, un vector $u_k \in \mathbb{R}^d$ está en el espacio nulo de S_W si:

$$S_W u_k = 0$$

para todo $k = r+1, \ldots, d$. Con r como el rango de S_W, y $n = d - r$ como la dimensión del espacio nulo de S_W.

Sea x_j el vector media de la clase j, $D_j \in \mathbb{R}^{m_j \times m_j}$ es una matriz con todos sus elementos iguales a m_j^{-1}, y $\mathbf{X}_j \in \mathbb{R}^{d \times m_j}$ es la matriz con todas las muestras x_j^i de la clase j.
La matriz de dispersión intra-clase de $\mathbf{X}$ se define como

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{m_j} (\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T = \sum_{j=1}^{c} (\mathbf{X}_j - \mathbf{X}_j D_j)(\mathbf{X}_j - \mathbf{X}_j D_j)^T$$

Multiplicando ambos lados de $S_W u_k = 0$ por u_k^T, se obtiene

$$\sum_{j=1}^{c} u_k^T \mathbf{X}_j (I_j - D_j)(I_j - D_j)^T \mathbf{X}_j^T u_k = \sum_{j=1}^{c} \|(I_j - D_j)\mathbf{X}_j^T u_k\|^2 = 0$$

donde $I_j \in \mathbb{R}^{m_j \times m_j}$ es la matriz identidad, y $\|\cdot\|$ denota la norma euclídea.
Entonces $(I_j - D_j)\mathbf{X}_j^T u_k = 0$,

$$\begin{aligned} \mathbf{X}_j^T u_k &= D_j \mathbf{X}_j^T u_k \\ \mathrm{x}_j^{iT} u_k &= \mathrm{x}_j^T u_k \end{aligned} \tag{C.4}$$

La proyección de x_j^i en el espacio nulo de S_W proporciona el vector común x_{cv}^j

$$\mathrm{x}_{cv}^j = \sum_{k=r+1}^{d} \langle \mathrm{x}_j^i, u_k \rangle u_k$$

De la ecuación (C.4)

$$\mathrm{x}_{cv}^j = \sum_{k=r+1}^{d} \langle \mathrm{x}_j, u_k \rangle u_k$$

Por tanto el vector común x_{cv}^j es independiente del subíndice i [44].

Teorema 3 *Sea $\mathcal{S}_1, \ldots, \mathcal{S}_n$ una colección finita de subespacios de un espacio $\mathcal{S}$ con bases $B_1, \ldots, B_n$, respectivamente. Entonces las siguientes condiciones son equivalentes:*

1. $\mathcal{S} = \mathcal{S}_1 \oplus \ldots \oplus \mathcal{S}_n$.
2. $\bigcup_{i=1}^{n} B_i$ *es una base de* $\mathcal{S}$ *y* $\mathcal{S}_i \bigcap \mathcal{S}_j = \{0\}$ *para cada* $i \neq j$.

Prueba.1 $\Rightarrow$ 2 : Puesto que $\mathcal{S}_i \bigcap (\sum_{j \neq i} \mathcal{S}_j) = \{0\}$, entonces $\mathcal{S}_i \cap \mathcal{S}_j = \{0\}$ para cada $i \neq j$, y en consecuencia, $B_i \cap B_j = 0$ para cada $i \neq j$. Puesto que $\mathcal{S} = \mathcal{S}_1 + \ldots + \mathcal{S}_n$ y B_i es una base para $\mathcal{S}_i$, entonces es claro que $\cup_{i=1}^{n} B_i$ es un sistema de generadores para $\mathcal{S}$.

Sea $p_1, \ldots, p_k$ un subconjunto finito de $\cup_{i=1}^{n} B_i$ y $a_i, \ldots, a_k$ escalares tales que $a_1 p_1 + \ldots + a_k p_k = 0$, entonces adicionando sumandos nulos mediante coeficientes nulos se puede escribir esta ecuación, sin pérdida de generalidad, en grupos de sumandos de tal forma que los primeros sumandos sean de $\mathcal{S}_1$, los siguientes sean de $\mathcal{S}_2$, etc, y los últimos sean de $\mathcal{S}_n$. Pero de acuerdo con la hipótesis 1, el vector nulo solo es representable mediante sumandos nulos, por lo tanto, cada grupo de sumandos es nulo y, puesto que en cada grupo aparecen combinaciones lineales de vectores linealmente independientes, entonces todos los coeficientes a_i son nulos.

2 $\Rightarrow$ 1) : Puesto que $\cup_{i=1}^{n} B_i$ es una base de $\mathcal{S}$, entonces

$$\mathcal{S} = span\{\cap_{i=1}^{n} B_i\} = \mathcal{S}_1 + \ldots + \mathcal{S}_n$$

Si ahora $\{0\} = s_1 + \ldots + s_n$, con $s_i \in \mathcal{S}_i$, entonces se puede representar cada s_i como combinación lineal de los vectores de B_i de tal forma que se obtiene una combinación lineal de vectores de $\cup_{i=1}^{n} B_i$; teniendo en cuenta que $B_i \cap B_j = 0$ para cada $i \neq j$, entonces en esta última representación no hay sumandos repetidos, lo cual, combinado con la hipótesis de que $\cup_{i=1}^{n} B_i$ es una base para $\mathcal{S}$ se obtiene que todos los sumandos en tal representación son nulos, es decir, cada s_i es nulo. Esto completa la prueba de 1 [7].

Teorema 4 *Sean $S_1, \ldots, S_c$ matrices de dispersión semidefinidas positivas. Luego,*

$$\mathcal{N}(S_1 + \ldots + S_c) = \bigcap_{j=1}^{c} \mathcal{N}(S_j)$$

Prueba. Un vector $u_k \in \mathbb{R}^d$ está en el espacio nulo de $(S_1 + \ldots + S_c)$ si

$$(S_1 + \ldots + S_c)u_k = 0$$

Donde $u_k^T(S_1 + \ldots + S_c)u_k = u_k^T S_1 u_k + \ldots + u_k^T S_c u_k = 0$, y $u_k \in \mathcal{N}(S_1 + \ldots + S_c)$ si y solo si $u_k^T S_j u_k = 0$, para todo $j = 1, \ldots, c$. O de forma equivalente, $u_k \in \bigcap_{j=1}^{c} \mathcal{N}(S_j)$ [22].

Teorema 5 *El vector común* x_{cv}^{j}*, es independiente de la selección del vector sustraendo en* $\mathcal{B}$*.*

$$\mathcal{B} = span\{\mathrm{x}_1^2 - \mathrm{x}_1^1 \ \ldots \ \mathrm{x}_j^i - \mathrm{x}_j^1 \ \ldots \ \mathrm{x}_c^{m_c} - \mathrm{x}_c^1\}$$

x_j^1 *puede ser cualquier elemento que pertenece a la clase* j*.*

Prueba. Sea $\mathrm{b}_j^i \in \mathcal{B}$, donde $\mathrm{b}_j^i = \mathrm{x}_j^i - \mathrm{x}_j^1$, y $\{\theta_1, \theta_2, \ldots, \theta_{(M-c)}\}$ es una base de $\mathcal{B}$. b_j^i se expresa en la base como

$$\begin{aligned} \mathrm{b}_j^i &= \langle \mathrm{b}_j^i, \theta_1\rangle\theta_1 + \langle \mathrm{b}_j^i, \theta_2\rangle\theta_2 + \cdots + \langle \mathrm{b}_j^i, \theta_{(M-c)}\rangle\theta_{(M-c)} \\ \mathrm{x}_j^i - \mathrm{x}_j^1 &= \langle \mathrm{x}_j^i - \mathrm{x}_j^1, \theta_1\rangle\theta_1 + \cdots + \langle \mathrm{x}_j^i - \mathrm{x}_j^1, \theta_{(M-c)}\rangle\theta_{(M-c)} \end{aligned}$$

De este modo

$$\begin{aligned} \mathrm{x}_{cv}^{j} &= \mathrm{x}_j^i - [\langle \mathrm{x}_j^i, \theta_1\rangle\theta_1 + \cdots + \langle \mathrm{x}_j^i, \theta_{(M-c)}\rangle\theta_{(M-c)}] \\ &= \mathrm{x}_j^1 - [\langle \mathrm{x}_j^1, \theta_1\rangle\theta_1 + \cdots + \langle \mathrm{x}_j^1, \theta_{(M-c)}\rangle\theta_{(M-c)}] \end{aligned} \qquad \text{(C.5)}$$

Es claro que x_j^i puede ser cualquier elemento de la clase j, donde $i = 1, \ldots, m_j$. A si mismo, x_{cv}^{j} es independiente de la selección del vector sustraendo [45].

Teorema 6 *Sean* $\mathcal{R}(S_W)$ *y* $\mathcal{N}(S_W)$ *el espacio rango y nulo de la matriz de dispersión* S_W*, respectivamente.*

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{m_j} (\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T$$

$\mathcal{B}^{\perp}$ *es el complemento ortonormal del subespacio diferencia* $\mathcal{B}$*. Luego,*

$$\begin{aligned} \mathcal{N}(S_W) &= \mathcal{B}^{\perp} \\ \mathcal{R}(S_W) &= \mathcal{B} \end{aligned}$$

Prueba. La matriz de dispersión intra-clase S_W se puede escribir como:

$$S_W = [\mathrm{a}_1^1 \ \mathrm{a}_1^2 \ \cdots \ \mathrm{a}_c^{m_c}][\mathrm{a}_1^1 \ \mathrm{a}_1^2 \ \cdots \ \mathrm{a}_c^{m_c}]^T = \mathrm{A}\mathrm{A}^T$$

donde $\mathrm{A} = [\mathrm{a}_1^1 \ \mathrm{a}_1^2 \ \cdots \ \mathrm{a}_c^{m_c}]$, y $\mathrm{a}_j^i = \mathrm{x}_j^i - \mathrm{x}_j$ para todo $i = 1, \ldots, m_j$, $j = 1, \ldots, c$. x_j representa el vector media de la clase j.
Sea u_k un vector propio asociado a un valor propio nulo de S_W,

$$S_W \mathrm{u}_k = 0, \qquad o \qquad \mathrm{A}\mathrm{A}^T \mathrm{u}_k = 0, \qquad \forall k = r+1, \ldots, d.$$

Multiplicando la ecuación anterior por u_k^T, se obtiene que $\mathrm{u}_k^T \mathrm{A}\mathrm{A}^T \mathrm{u}_k = 0$. Con $\|\mathrm{A}^T \mathrm{u}_k\|$ igual a 0.

Por lo tanto:

$$\begin{aligned} \mathrm{a}_j^{iT}\mathrm{u}_k &= 0 \\ (\mathrm{x}_j^i - \mathrm{x}_j)^T\mathrm{u}_k &= 0 \end{aligned}$$

donde $(\mathrm{x}_j^i - \mathrm{x}_j) \in \mathcal{B}$. Y entonces $\mathrm{u}_k \perp \mathcal{B}$, lo cual es equivalente a $\mathrm{u}_k \in \mathcal{B}^\perp$.

El conjunto de vectores propios, u_k, asociados a los valores propios nulos de S_W son una base del espacio nulo $\mathcal{N}(S_W)$, luego

$$\mathcal{N}(S_W) \subset \mathcal{B}^\perp \tag{C.6}$$

A si $\mathcal{N}(S_W)$ es un subconjunto de $\mathcal{B}^\perp$.

Si $\theta_k \in \mathcal{B}^\perp$, consecuentemente $\theta_k \perp \mathcal{B}$. De esta forma

$$\begin{aligned} < \mathrm{x}_j^i - \mathrm{x}_j^1, \theta_k > &= 0 \\ < \mathrm{x}_j^i - \mathrm{x}_j, \theta_k > &= 0 \end{aligned}$$

para todo $k = (M - c) + 1, \ldots, d$.

Entonces

$$\begin{aligned} S_W\theta_k &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}[(\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T\theta_k] \\ &= \sum_{j=1}^{c}\sum_{i=1}^{m_j}[(\mathrm{x}_j^i - \mathrm{x}_j) < \mathrm{x}_j^i - \mathrm{x}_j, \theta_k >] \\ &= 0 \end{aligned}$$

tal que $\theta_k \in \mathcal{N}(S_W)$, y

$$\mathcal{B}^\perp \subset \mathcal{N}(S_W) \tag{C.7}$$

$\mathcal{B}^\perp$ es un subconjunto de $\mathcal{N}(S_W)$.

Desde las ecuaciones (C.6) y (C.7), se concluye que

$$\mathcal{N}(S_W) = \mathcal{B}^\perp \tag{C.8}$$

La matriz S_W se puede escribir como:

$$S_W = \sum_{j=1}^{c} S_j$$

donde cada $S_j = \sum_{i=1}^{m_j}(\mathrm{x}_j^i - \mathrm{x}_j)(\mathrm{x}_j^i - \mathrm{x}_j)^T$, es una matriz de dispersión semidefinida positiva.

Desde el teorema 4, y dado que $\mathcal{R}(S_W)$ y $\mathcal{N}(S_W)$ son subespacios complementarios

$$\begin{aligned} \mathcal{R}(S_W) &= \mathcal{R}(S_1 + \ldots + S_c) \\ &= (\mathcal{N}(S_1 + \ldots + S_c))^\perp = \left(\bigcap_{j=1}^{c}\mathcal{N}(S_j)\right)^\perp \\ &= (\mathcal{N}(S_1))^\perp + \ldots + (\mathcal{N}(S_c))^\perp \end{aligned}$$

De la ecuación (C.8)

$$\mathcal{N}(S_1) + \ldots + \mathcal{N}(S_c) = \mathcal{B}_1^{\perp} + \ldots + \mathcal{B}_c^{\perp}$$

donde $\mathcal{B} = (\mathcal{B}^{\perp})^{\perp}$

$$\mathcal{R}(S_1 + \ldots + S_c) = \mathcal{B}_1 + \ldots + \mathcal{B}_c$$

Finalmente

$$\begin{aligned} \mathcal{N}(S_W) &= \mathcal{B}^{\perp} \\ \mathcal{R}(S_W) &= \mathcal{B} \end{aligned}$$

Para mas detalles ver [44, 45].

Teorema 7 *Sean* $\mathrm{P}^{(1)}$ *y* $\mathrm{P}^{(2)}$ *matrices de proyección a los subespacios* $\mathcal{N}(S_T)$ *y* $\mathcal{R}(S_W)$*, respectivamente. Luego,* $\mathrm{P}^{(1)}$ *y* $\mathrm{P}^{(2)}$ *conmutan*

$$\mathrm{P}^{(1)}\mathrm{P}^{(2)} = \mathrm{P}^{(2)}\mathrm{P}^{(1)}$$

Prueba. Dado que $S_T = S_B + S_W$. El espacio nulo de S_T se puede definir como

$$\begin{aligned} \mathcal{N}(S_T) &= \mathcal{N}(S_B + S_W) \\ &= \mathcal{N}(S_B) \cap \mathcal{N}(S_W) \end{aligned}$$

y en particular, $\mathcal{N}(S_T) \subset \mathcal{N}(S_W)$.

Desde lo anterior y con el hecho que $\mathcal{N}(S_W) \perp \mathcal{R}(S_W)$, se muestra que

$$\mathcal{N}(S_T) \perp \mathcal{R}(S_W)$$

Si $\mathrm{P}^{(1)}$ y $\mathrm{P}^{(2)}$ son las matrices de proyección a los subespacios $\mathcal{N}(S_T)$ y $\mathcal{R}(S_W)$, respectivamente. Luego, $(\mathrm{I}-\mathrm{P}^{(1)})(\mathrm{I}-\mathrm{P}^{(2)}) = 0$ y $(\mathrm{I}-\mathrm{P}^{(2)})(\mathrm{I}-\mathrm{P}^{(1)}) = 0$.

$$\begin{aligned} (\mathrm{I} - \mathrm{P}^{(1)})(\mathrm{I} - \mathrm{P}^{(2)}) &= (\mathrm{I} - \mathrm{P}^{(2)})(\mathrm{I} - \mathrm{P}^{(1)}) = 0 \\ \mathrm{I} - \mathrm{P}^{(1)} - \mathrm{P}^{(2)} - \mathrm{P}^{(1)}\mathrm{P}^{(2)} &= \mathrm{I} - \mathrm{P}^{(2)} - \mathrm{P}^{(1)} - \mathrm{P}^{(2)}\mathrm{P}^{(1)} \end{aligned}$$

lo cual implica $\mathrm{P}^{(1)}\mathrm{P}^{(2)} = \mathrm{P}^{(2)}\mathrm{P}^{(1)}$ [44].

Teorema 8 *El vector sustraendo de la clase j para calcular el subespacio diferencia de* $\mathbf{Y}$ *respecto a* $\mathbf{X}$ *puede ser cualquier elemento de la clase j en* $\mathbf{X}$*. Donde el conjunto de datos* $\mathbf{X}$ *e* $\mathbf{Y}$ *representan las mimas clases.*

$$\mathcal{B}_{Y_x} = span\{\mathrm{y}_1^1 - \mathrm{x}_1^i, \mathrm{y}_1^2 - \mathrm{x}_1^i, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^i, \mathrm{y}_2^2 - \mathrm{x}_2^i, \ldots, \mathrm{y}_c^{n_c} - \mathrm{x}_c^i\}$$

Prueba. Sea $\mathcal{B}_{Y_x}$ el subespacio diferencia de $\mathbf{Y}$ respecto a $\mathbf{X}$, definido como

$$\begin{aligned}\mathcal{B}_{Y_x} &= span\{\mathrm{y}_1^1 - \mathrm{x}_1^1,\, \mathrm{y}_1^2 - \mathrm{x}_1^1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2^1,\, \mathrm{y}_2^2 - \mathrm{x}_2^1, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c^1\} \quad o, \\ &= span\{\mathrm{y}_1^1 - \mathrm{x}_1,\, \mathrm{y}_1^2 - \mathrm{x}_1, \ldots, \mathrm{y}_2^1 - \mathrm{x}_2,\, \mathrm{y}_2^2 - \mathrm{x}_2, \ldots,\, \mathrm{y}_c^{n_c} - \mathrm{x}_c\}\end{aligned}$$

Una base ortonormal υ de $\mathcal{B}_{Y_x}$, que sea ortogonal a la base U de $\mathcal{B}_X$, se obtiene a partir de una base ortonormal de la proyección de $\mathcal{B}_{Y_x}$ en el subespacio indiferencia de $\mathbf{X}$.

$$\upsilon = orth\left(\overline{\mathrm{U}}\,\overline{\mathrm{U}}^T\ \mathcal{B}_{Y_x}\right) = orth\left(\left(\mathrm{I} - \mathrm{U}\mathrm{U}^T\right)\mathcal{B}_{Y_x}\right)$$

Donde,

$$\begin{aligned}(\mathrm{y}_j^i - \mathrm{x}_j^1) - \mathrm{U}\mathrm{U}^T(\mathrm{y}_j^i - \mathrm{x}_j^1) &= \mathrm{y}_j^i - \mathrm{U}\mathrm{U}^T\mathrm{y}_j^i - \mathrm{x}_{cv}^j \\ (\mathrm{y}_j^i - \mathrm{x}_j) - \mathrm{U}\mathrm{U}^T(\mathrm{y}_j^i - \mathrm{x}_j) &= \mathrm{y}_j^i - \mathrm{U}\mathrm{U}^T\mathrm{y}_j^i - \mathrm{x}_{cv}^j\end{aligned}$$

con $i = 1, \ldots, n_j$ y $j = 1, \ldots, c$.

Ya que la proyección de los vectores diferencia de $\mathbf{Y}$ es independiente de la selección del vector sustraendo en $\mathcal{B}_X$, podemos definir el subespacio diferencia $\mathcal{B}_{Y_x}$ a partir de usar como vector sustraendo de la clase j cualquier muestra de la misma clase en $\mathbf{X}$ [28].

Apéndice D

Bases de datos

Aquí se describen las bases de datos empleadas en la validación empírica de los métodos propuestos. Todas la bases de datos son de imágenes y están públicamente disponibles. Las bases de datos se dividen en tres grupos principales que son de rostros [§D.1], de objetos [§D.2] y de dígitos escritos a mano [§D.3]. Todas las imágenes están normalizadas a niveles de gris en el rango de [0 1]. Además, las bases de datos de rostros se han alineado respecto al centro de los ojos y de la boca.

D.1. Bases de datos de rostros

Cmu-Pie

Cmu-Pie es un subconjunto de la base de datos de rostros CMU-PIE (Pose, Illumination, and Expression [107]). Este base de datos tiene imágenes de 68 personas, y de cada persona hay 56 imágenes con expresión neutra. De las 56 imágenes por persona, 13 presentan cambios de pose desde la derecha a la izquierda. Las 43 imágenes restantes son imágenes frontales que tienen condiciones de luz diferentes. Las imágenes de tamaño original 486×640 han sido escaladas a una resolución de 120×160. La figura D.1 muestra las imágenes normalizadas y centradas de un sujeto en Cmu-Pie.

Figura D.1: Muestras de una clase de la base de datos Cmu-Pie.

ORL

La base de datos ORL es un conjunto de imágenes tomadas entre 1992 y 1994 en los Laboratorios AT&T de la Universidad de Cambridge, UK. [94]. En esta base de datos hay imágenes de 40 personas diferentes, de cada persona hay 10 imágenes que presentan cambios de expresión facial (ojos abiertos/cerrados, sonriente/serio), pose y algunos detalles faciales (con gafas/sin gafas). Todas las imágenes se tomaron con un fondo homogéneo, en posición frontal y algunas con ciertas variaciones en el ángulo de la cara. El tamaño de cada imagen es de 92×112 píxeles. En la siguiente figura se muestra una imagen de cada uno de los individuos de la base de datos ORL.

Figura D.2: Una imagen de cada uno de los individuos de la base de datos ORL.

AR

La base de datos de rostros AR Face [83] fue creado por Aleix Martínez y Roberto Benavente en el Centro de Visión por Computador (CVC) de la UAB. Ésta contiene más de 4.000 imágenes que correspon-

den a los rostros de 126 personas. Los rostros están en posición frontal y presentan diferentes expresiones faciales, condiciones de iluminación y oclusiones (gafas de sol y bufanda). Las imágenes fueron tomas en dos sesiones, separadas por dos semanas (14 días) de tiempo. Las mismas imágenes se tomaron en las dos sesiones. En nuestra experimentación usamos 700 imágenes de 50 individuos, donde cada individuo tiene 14 imágenes. Concretamente, estas 14 imágenes corresponden a las imágenes (a)–(g) y (n)–(t) de la base de datos original. La resolución de cada imagen es de 222×299. La figura D.3 muestra una imagen de algunos de los 50 individuos.

Figura D.3: Algunas imágenes de la base de datos AR.

AR NG

AR NG es una base de datos de imágenes de rostros en niveles de gris que se originó al seleccionar 20 clases de la base de datos AR Face [83]. De cada clase se escogieron 14 muestras sin oclusiones que se normalizaron y se les añadió ruido gaussiano con valores de varianza de 0, 0.02, 0.04, 0.06 y 0.08. Las imágenes con ruido gaussiano se generan a partir de las 14 imágenes originales de cada sujeto, obteniendo 70 muestras por cada una de las 20 clases (individuos). Por lo tanto, 14 imágenes son originales y 56 (14 x 4) imágenes tienen ruido gaussiano. Las imágenes de tamaño inicial 768×576 fueron reducidas a un tamaño de 40×40. La figura D.4 enseña la degradación de las muestras en AR NG.

ALL NG

La base ALL NG resulta de la elección aleatoria de 10 muestras sin oclusiones por clase de las bases de rostros AR Face [83], ORL [94], Yale [40] y Umist [115], obteniendo 95 clases. Cada clase de ALL NG tiene 50 muestras, de las cuales 10 no presentan ruido añadido y 40 presentan

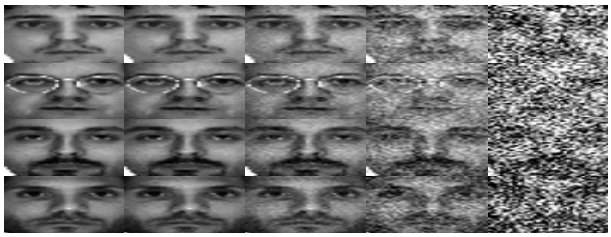


Figura D.4: Ejemplo de la degradación de las imágenes en la base de rostros AR NG, para algunos sujetos. El ruido añadido es de tipo gaussiano con valores de varianza de 0, 0.02, 0.04, 0.06 y 0.08, respectivamente.

ruido gaussiano añadido (generadas a partir de las 10 imágenes originales normalizadas de cada sujeto). Las características del ruido añadido son idénticas a las de AR NG. La resolución de las imágenes es de 40×40 y presentan cambios de expresión, iluminación, pose y ruido.

D.2. Bases de datos de objetos

Las bases de datos de objetos utilizadas en nuestra experimentación se generan a partir de Coil-100 [87]. La base de datos Coil-100 (*Columbia Object Image Library*) contiene imágenes de 100 objetos de apariencia y geometría diferentes. Las imágenes de los objetos fueron tomadas a intervalos de 5 grados desde 0 a 355, por lo cual hay 72 poses por objeto. Todas las imágenes tienen un tamaño de 128×128.

Coil-40/30

Coil-40/30 es una base de datos de objetos en niveles de gris que procede de Coil-100 [87], y está constituida por 40 clases seleccionadas al azar entre las 100 clases de la base de datos original. Por cada clase hay 30 imágenes seleccionadas aleatoriamente entre las 72 originales. Las imágenes de tamaño inicial 128x128 son reducidas a un tamaño de 40x40. La figura D.5 muestra un objeto por clase de la base de datos Coil-40/30.

Figura D.5: Clases de la base de datos Coil-40/30.

Coil-20

La base Coil-20 posee imágenes en niveles de gris de 20 objetos. Las imágenes de los objetos fueron tomadas a intervalos de 5 grados desde 0 a 355, por lo cual hay 72 poses por objeto. Los objetos en Coil-20 son los que presentan una geometría más compleja dentro de la base de datos Coil-100 [88]. La figura D.6 muestra un objeto por clase de Coil-20.

Figura D.6: Clases de la base de datos Coil-20

D.3. Base de datos de dígitos escritos a mano

Mnist

Mnist [72] es una base de datos de dígitos de 0 a 9 escritos a mano, que es un subconjunto de la base de datos Nist32 (disponible en prtools [109]). Las imágenes originales son binarias y se han convertido a niveles de gris a partir de la transformación de distancia continua descrita en [5]. En cada una de las 10 clases de Mnist hay 100 muestras que han sido seleccionadas al azar desde las imágenes de Nist32. Cada imagen es de tamaño 32×32. Algunas de las imágenes de la base de datos se muestran en la figura D.7.

Figura D.7: Imágenes de la base de datos Mnist.

Usps

Usps es un subconjunto de la base de datos descrita en [58]. Usps esta formada por imágenes de dígitos de 0 a 9 escritos a mano, con una resolución de 16×16 píxeles. En cada clase de Usps hay 25 imágenes del dígito. Las imágenes se han modificado con transformación de distancia continua descrita en [5]. La figura D.8 presenta muestras de Usps.

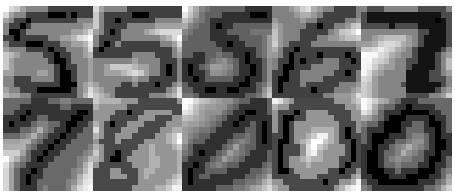


Figura D.8: Imágenes de la base de datos Usps.

Apéndice E

Resultados numéricos

Se presentan los resultados numéricos de la validación de los algoritmos IDCV-EVD/GSO/O y IRDCV, respecto a nuevas muestras y nuevas clases, según sea el caso.

En las siguientes tablas, y a menos que se diga lo contrario, $(1-e)$ es la tasa de acierto del algoritmo con aprendizaje por lotes, y es idéntica a la del algoritmo incremental. CPU(seg.) es el tiempo que emplea el método con aprendizaje por lotes en el entrenamiento. (CPU %) es el coste del método incremental respecto al algoritmo por lotes. El modelo inicial se obtiene a partir del algoritmo base original.

En las tablas que muestran los resultados de los algoritmos incrementales respecto a nuevas muestras de entrenamiento, m_j y n_j son el número de muestras de entrenamiento en el aprendizaje por lotes e incremental, respectivamente. Cuando las tablas muestran los resultados de los algoritmos incrementales respecto a nuevas clases de entrenamiento, c_x y c_y son el número de clases en el entrenamiento por lotes e incremental, respectivamente.

E.1. Validación IDCV-EVD

Las tablas E.1, E.2, E.3 y E.4 muestran los resultados de la validación del método IDCV-EVD respecto a DCV-EVD, en el caso de nuevas muestras de entrenamiento, para las bases de datos Cmu-Pie, Coil-20, Mnist y Usps, respectivamente. Las tablas E.5 y E.6 muestran los resultados de la validación del método IDCV-EVD cuando nuevas clases se incorporan a la base de conocimiento de AR y ORL, respectivamente.

Tabla E.1: Aprendizaje incremental en la base de datos Cmu-Pie, para muestras.

Cmu-Pie, IDCV-EVD											
m_j	$n_j = 1$			$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	$(1-e)$	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
8	0.72	9.87		9.53		9.11		9.36		9.50	
9	0.74	11.75	12.33								
10	0.75	14.74	9.58	14.07	39.62						
11	0.76	16.30	9.04			15.54	47.59				
12	0.76	19.23	7.92	18.64	29.35			17.93	54.36		
13	0.77	21.18	7.27							20.17	64.13
14	0.77	23.92	6.79	23.78	22.95	23.23	33.77				
15	0.78	27.40	5.95								
16	0.78	29.67	5.62	28.79	19.48			28.81	35.62		
17	0.78	31.64	5.31			31.95	25.56				
18	0.78	35.30	4.83	35.82	16.29					35.70	35.81
19	0.79	39.25	4.48								
20	0.79	43.21	4.22	44.50	13.50	43.31	19.09	43.31	24.36		
21	0.79	49.13	3.92								
22	0.79	54.03	3.60	51.71	11.90						
23	0.79	58.96	3.37			57.59	14.69			57.89	24.44
24	0.79	62.90	3.11	61.97	9.88			63.27	17.69		
25	0.79	68.11	3.01								
26	0.80	75.78	2.81	75.17	8.55	72.51	12.38				
27	0.79	81.06	2.64								
28	0.80	89.95	2.42	87.97	7.53			86.74	13.11	86.69	16.85
29	0.80	96.16	2.33			94.80	9.79				
30	0.80	104.52	2.16	102.69	6.56						
31	0.80	112.73	2.06								
32	0.80	121.57	1.97	117.79	5.88	117.94	7.96	116.95	10.15		
33	0.80	126.66	1.94							125.88	12.49
34	0.80	136.02	1.85	135.00	5.21						
35	0.80	146.20	1.66			143.39	6.73				
36	0.80	161.15	1.53	156.80	4.52			155.33	8.18		
37	0.80	167.05	1.55								
38	0.80	179.27	1.48	182.49	4.19	175.25	5.69			180.78	8.81
39	0.80	192.39	1.43								
40	0.80	201.73	1.36	208.25	3.65			207.14	6.32		

Tabla E.2: Aprendizaje incremental de la base de datos Coil-20, para muestras.

Coil-20, IDCV-EVD											
m_j	$n_j = 1$			$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	$(1-e)$	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
11	0.91	1.94		1.81		1.76		2.07		1.89	
12	0.92	2.14	10.01								
13	0.92	2.37	9.77	2.16	31.93						
14	0.92	2.40	9.52			2.45	44.53				
15	0.93	2.88	10.29	2.76	24.04			2.84	63.00		
16	0.93	3.01	9.79							2.83	78.09
17	0.93	3.31	9.11	3.31	21.32	3.32	34.96				
18	0.93	3.39	8.77								
19	0.94	3.75	8.17	3.69	19.52			3.85	47.43		
20	0.94	3.99	8.24			4.01	30.26				
21	0.94	4.37	7.51	4.28	16.82					4.20	57.04
22	0.95	4.77	7.33								
23	0.95	5.01	7.30	4.99	14.36	4.88	25.38	5.19	36.09		
24	0.95	5.29	7.16								
25	0.95	5.71	6.69	5.54	13.17						
26	0.95	6.14	6.37			5.93	21.48			6.08	44.48
27	0.95	6.57	6.00	6.70	11.00			6.62	28.56		
28	0.96	6.86	6.11								
29	0.96	7.04	6.08	6.99	11.68	6.95	18.15				
30	0.96	7.59	5.71								
31	0.96	8.24	5.36	7.99	9.742			8.42	23.58	8.16	30.98
32	0.96	8.32	5.38			8.27	16.06				
33	0.96	8.68	5.29	8.92	8.94						
34	0.96	9.32	5.16								
35	0.96	9.96	4.99	10.25	8.05	9.86	13.75	9.73	20.37		
36	0.96	10.36	4.75							10.44	25.27
37	0.96	10.73	4.73	10.87	7.70						
38	0.97	11.38	4.44			11.28	12.05				
39	0.97	11.71	4.39	11.75	7.13			12.45	15.94		
40	0.97	12.56	4.05								
41	0.97	13.19	4.19	12.98	6.84	13.29	10.68			12.79	21.48
42	0.97	13.87	3.94								
43	0.97	14.47	3.83	13.96	5.98			13.69	15.14		
44	0.97	15.36	3.71			14.66	10.12				
45	0.97	15.78	3.62	15.38	5.44						
46	0.98	16.63	3.55							16.17	17.00
47	0.98	17.34	3.55	17.03	5.40	17.08	8.54	16.96	12.58		
48	0.98	17.74	3.46								
49	0.98	18.90	3.35	18.68	4.86						
50	0.98	19.62	3.16			19.13	7.61				
51	0.98	20.43	3.11	20.04	4.65			19.74	10.79	20.52	14.96

Tabla E.3: Aprendizaje incremental de la base de datos Mnist, para muestras.

Mnist, IDCV-EVD											
m_j	$n_j = 1$			$n_j = 4$, CPU		$n_j = 7$, CPU		$n_j = 10$, CPU		$n_j = 13$, CPU	
	$(1-e)$	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
15	0.88	0.10		0.10		0.09	0	0.11		0.11	
16	0.88	0.10	15.84								
17	0.88	0.11	13.97								
18	0.89	0.12	12.50								
19	0.89	0.12	17.06	0.14	35.29						
20	0.90	0.14	13.80								
21	0.89	0.15	11.01								
22	0.89	0.17	16.52			0.16	45.47				
23	0.89	0.19	12.73	0.19	23.84						
24	0.90	0.20	9.95								
25	0.90	0.22	12.82					0.23	64.11		
26	0.90	0.24	8.66								
27	0.90	0.27	8.49	0.28	17.78						
28	0.90	0.29	9.16							0.31	72.74
29	0.90	0.31	5.92			0.29	27.95				
30	0.90	0.35	5.16								
31	0.90	0.35	6.71	0.35	14.15						
32	0.90	0.37	5.04								
33	0.90	0.42	5.48								
34	0.90	0.46	4.55								
35	0.90	0.50	6.09	0.50	11.26			0.47	31.23		
36	0.90	0.51	3.55			0.48	17.36				
37	0.90	0.52	3.65								
38	0.90	0.57	3.97								
39	0.90	0.69	3.47	0.66	8.70						
40	0.90	0.68	4.24								
41	0.91	0.75	2.64							0.75	30.37
42	0.91	0.78	3.08								
43	0.90	0.84	5.14	0.87	6.81	0.78	11.23				
44	0.91	0.90	2.42								
45	0.90	0.94	2.79					0.92	17.59		
46	0.91	0.99	2.67								
47	0.91	1.09	2.32	1.04	6.47						
48	0.91	1.12	2.16								
49	0.90	1.18	1.98								
50	0.90	1.28	2.46			1.22	7.61				
51	0.90	1.33	1.86	1.31	4.56						
52	0.90	1.48	1.88								
53	0.90	1.49	3.25								
54	0.90	1.63	1.71							1.50	15.60
55	0.90	1.62	1.73	1.61	4.47			1.62	11.32		
56	0.90	1.71	1.46								
57	0.90	1.75	2.42			1.75	5.39				
58	0.90	1.85	2.16								
59	0.90	2.00	1.69	1.88	3.89						
60	0.88	2.01	1.37								
61	0.88	2.12	1.58								
62	0.88	2.22	1.40								
63	0.87	2.35	1.79	2.29	3.95						
64	0.87	2.35	1.49			2.44	4.13				
65	0.87	2.52	1.41					2.39	8.46		
66	0.87	2.62	1.30								
67	0.86	2.68	1.26	2.80	3.01					2.56	9.53
68	0.85	2.76	1.28								
69	0.85	2.86	1.20								
70	0.85	3.00	1.50								
71	0.85	3.25	1.24	3.20	2.38	3.09	3.14				

Tabla E.4: Aprendizaje incremental de la base de datos Usps, para muestras.

Usps, IDCV-EVD											
m_j	$n_j = 1$			$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	$(1-e)$	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
7	0.77	0.03		0.02		0.02		0.02		0.02	
8	0.82	0.03	12.67								
9	0.80	0.03	12.60	0.02	53.33						
10	0.82	0.03	6.49			0.03	61.78				
11	0.83	0.04	9.59	0.03	47.72			0.03	75.36		
12	0.82	0.04	7.54							0.04	68.83
13	0.80	0.04	6.66	0.04	34.97	0.05	42.11				
14	0.81	0.05	4.80								
15	0.78	0.06	4.72	0.05	29.08			0.05	46.71		
16	0.78	0.07	5.44			0.06	32.89				
17	0.77	0.07	5.20	0.07	22.41					0.08	39.99
18	0.75	0.08	4.59								
19	0.73	0.09	2.59	0.08	17.28	0.10	21.52	0.09	27.73		
20	0.70	0.10	3.10								
21	0.65	0.11	3.43	0.09	14.49						
22	0.60	0.13	3.09			0.12	19.04			0.13	27.46

Tabla E.5: Aprendizaje incremental de la base de datos AR, con nuevas clases.

AR, IDCV-EVD												
c_x	$c_y = 1$			$c_y = 3$			$c_y = 6$			$c_y = 8$		
	$(1-e)$	CPU (seg.)	CPU %	$(1-e)$	CPU (seg.)	CPU %	$(1-e)$	CPU (seg.)	CPU %	$(1-e)$	CPU (seg.)	CPU %
26	0.98	7.93		0.98	8.17		0.97	8.58		0.97	7.76	
27	0.98	8.87	5.80									
28	0.98	9.27	5.26									
29	0.98	9.45	5.25	0.98	9.97	16.90						
30	0.98	10.20	5.28									
31	0.98	10.75	5.06									
32	0.98	11.40	5.04	0.98	11.27	15.90	0.97	11.55	36.35			
33	0.98	12.11	5.03									
34	0.97	12.53	4.88							0.98	12.23	49.67
35	0.98	13.24	4.19	0.97	13.22	13.69						
36	0.98	13.58	4.25									
37	0.98	14.24	4.27									
38	0.98	15.02	4.16	0.97	15.28	11.91	0.97	15.62	29.04			
39	0.98	15.54	3.92									
40	0.98	16.07	3.68									
41	0.98	17.20	3.73	0.97	17.11	11.11						
42	0.98	17.85	3.72							0.98	17.44	35.62
43	0.98	18.43	3.64									
44	0.98	19.26	3.63	0.97	19.46	10.23	0.98	20.52	22.34			
45	0.97	19.85	3.37									
46	0.97	21.02	3.14									
47	0.97	22.23	2.88	0.97	21.62	9.40						
48	0.97	22.44	2.95									
49	0.97	23.09	2.92									
50	0.97	24.26	2.76	0.97	24.27	8.61	0.97	26.78	17.29	0.97	23.05	28.08

Tabla E.6: Aprendizaje incremental de la base de datos ORL, con nuevas clases.

ORL, IDCV-EVD												
c_x	$c_y = 1$			$c_y = 3$			$c_y = 6$			$c_y = 8$		
	$(1-e)$	CPU (seg.)	CPU %	$(1-e)$	CPU (seg.)	CPU %	$(1-e)$	CPU (seg.)	CPU %	$(1-e)$	CPU (seg.)	CPU %
16	0.99	0.39		0.98	0.40		0.99	0.40		0.99	0.39	
17	0.99	0.44	12.15									
18	0.98	0.49	12.61									
19	0.99	0.55	11.27	0.98	0.55	24.23						
20	0.98	0.57	11.51									
21	0.99	0.62	11.01									
22	0.99	0.66	9.16	0.98	0.65	20.05	0.98	0.68	43.89			
23	0.99	0.69	9.82									
24	0.99	0.70	9.93							0.98	0.80	56.41
25	0.98	0.79	8.48	0.98	0.80	19.28						
26	0.98	0.83	7.61									
27	0.98	0.89	7.30									
28	0.98	0.87	8.41	0.98	0.96	15.03	0.97	0.95	31.14			
29	0.98	0.95	6.76									
30	0.98	1.00	8.12									
31	0.98	1.05	7.00	0.98	1.07	14.41						
32	0.98	1.14	6.47							0.98	1.12	40.60
33	0.97	1.22	6.31									
34	0.97	1.20	6.34	0.98	1.27	12.47	0.97	1.29	24.84			
35	0.98	1.27	6.13									
36	0.97	1.35	5.46									
37	0.97	1.47	5.51	0.98	1.46	10.31						
38	0.97	1.47	5.31									
39	0.97	1.53	4.67									
40	0.97	1.58	5.18	0.98	1.66	9.98	0.97	1.70	20.47	0.98	1.63	31.01

E.2. Validación IDCV-GSO

Las tablas E.7, E.8, E.9 y E.10 muestran los resultados de la validación del método IDCV-GSO respecto DCV-GSO, para nuevas muestras de entrenamiento en las bases de datos Cmu-Pie, Coil-20, Mnist y Usps, respectivamente. Las tablas E.11 y E.12 muestran los resultados de la validación respecto a nuevas clases de entrenamiento, en las bases de datos AR y ORL, respectivamente. La tasa de acierto del los métodos basados en GSO es idéntica a la tasa de acierto de los método basados en EVD.

Tabla E.7: Aprendizaje incremental en la base de datos Cmu-Pie, para muestras, a partir de IDCV-GSO.

Cmu-Pie, IDCV-GSO										
m_j	$n_j = 1$		$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
8	38.16		38.01		37.47		36.01		38.64	
9	48.32	21.55								
10	60.00	19.95	60.31	36.68						
11	73.78	18.05			74.78	52.61				
12	87.95	16.66	90.94	30.55			88.91	58.45		
13	106.24	15.18							113.16	65.83
14	126.45	14.12	123.19	27.88	128.43	38.29				
15	142.53	13.20								
16	168.49	11.89	160.88	24.56			168.40	45.99		
17	185.40	11.41			194.05	30.93				
18	205.84	11.03	218.22	20.85					211.45	50.14
19	240.74	10.21								
20	264.88	9.55	274.20	18.03	278.44	27.27	273.36	35.43		
21	291.15	9.66								
22	322.17	8.95	341.55	17.32						
23	356.14	8.59			366.01	23.38			377.14	38.31
24	391.97	7.84	426.93	15.43			389.07	31.62		
25	417.76	7.67								
26	453.78	7.37	489.73	13.89	473.87	20.87				
27	494.04	7.16								
28	540.37	6.83	533.57	14.06			534.47	27.13	555.68	33.18
29	579.40	6.54			594.86	19.24				
30	623.60	6.41	616.42	12.77						
31	679.91	6.18								
32	722.12	5.98	700.70	11.66	718.93	17.62	694.26	24.60		
33	769.57	5.73							785.20	28.46
34	802.62	5.75	793.26	11.22						
35	846.47	5.67			842.24	16.38				
36	898.31	5.50	895.86	10.27			881.70	22.20		
37	944.87	5.22								
38	1006.02	5.11	970.63	10.44	1017.69	14.70			1053.09	24.87
39	1049.23	4.87								
40	1107.53	4.79	1061.63	9.70			1084.93	19.47		

Tabla E.8: Aprendizaje incremental en la base de datos Coil-20, para muestras, a partir de IDCV-GSO.

Coil-20, IDCV-GSO										
m_j	$n_j = 1$		$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
11	4.86		4.98		5.27		5.03		5.00	
12	5.99	17.06								
13	7.07	16.40	7.05	30.52						
14	9.30	13.02			9.94	34.02				
15	11.34	11.71	10.94	24.40			11.16	43.98		
16	12.67	11.30							12.02	50.95
17	14.52	10.53	14.62	19.84	14.67	28.84				
18	16.16	10.33								
19	18.19	9.67	17.58	19.19			17.89	34.17		
20	20.63	9.13			19.83	25.53				
21	22.11	8.72	22.16	16.66					21.37	36.96
22	23.81	8.71								
23	27.22	8.11	26.78	15.25	27.42	21.08	26.38	28.13		
24	28.01	7.93								
25	31.13	7.62	31.08	14.10						
26	33.37	7.42			34.38	19.19			33.27	31.37
27	37.28	6.89	36.82	12.88			37.75	23.24		
28	40.65	6.54								
29	43.23	6.41	42.48	12.48	44.71	17.30				
30	46.05	6.28								
31	47.60	6.15	47.87	11.36			49.12	21.48	49.38	27.27
32	52.67	5.85			51.29	17.03				
33	56.19	5.63	57.36	10.89						
34	59.36	5.46								
35	62.53	5.33	63.57	9.81	63.72	14.53	63.56	19.21		
36	65.79	5.25							66.06	22.57
37	71.29	5.05	72.63	9.18						
38	72.43	5.03			77.23	13.39				
39	78.00	4.90	77.24	8.97			79.22	16.41		
40	81.98	4.74								
41	86.28	4.64	86.55	8.49	87.88	12.61			84.82	20.61
42	89.97	4.41								
43	95.55	4.30	95.35	7.88			93.72	15.98		
44	102.80	4.14			104.57	11.66				
45	104.08	4.17	102.79	7.74						
46	111.93	4.01							107.45	18.08
47	116.12	4.01	115.75	7.44	118.90	10.80	113.12	14.59		
48	119.38	3.94								
49	123.80	3.89	123.94	7.23						
50	128.63	3.79			132.30	10.23				
51	137.30	3.62	139.58	6.66			130.92	13.28	136.08	16.19

Tabla E.9: Aprendizaje incremental en la base de datos Mnist, para muestras, a partir de IDCV-GSO.

Mnist, IDCV-GSO										
m_j	$n_j = 1$		$n_j = 4$, CPU		$n_j = 7$, CPU		$n_j = 10$, CPU		$n_j = 13$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
15	0.15		0.16		0.15		0.16		0.16	
16	0.17	17.82								
17	0.19	14.88								
18	0.22	14.65								
19	0.25	13.66	0.25	42.50						
20	0.28	12.74								
21	0.31	11.68								
22	0.34	11.11			0.35	59.08				
23	0.37	11.66	0.38	34.42						
24	0.41	11.41								
25	0.45	9.42					0.49	63.45		
26	0.49	9.25								
27	0.53	9.15	0.52	29.87						
28	0.57	8.68							0.58	74.42
29	0.62	8.17			0.63	44.86				
30	0.67	7.74								
31	0.73	7.39	0.70	25.80						
32	0.75	7.54								
33	0.80	6.94								
34	0.86	6.73								
35	0.94	6.27	0.89	22.97			0.92	52.12		
36	0.97	6.45			0.99	36.94				
37	1.05	6.12								
38	1.07	6.37								
39	1.17	5.79	1.13	20.28						
40	1.26	5.86								
41	1.30	5.89							1.31	55.68
42	1.33	5.50								
43	1.44	5.39	1.41	18.19	1.42	30.69				
44	1.49	5.43								
45	1.54	5.20					1.55	40.44		
46	1.60	5.03								
47	1.70	4.72	1.70	16.71						
48	1.78	4.77								
49	1.81	4.71								
50	1.98	4.73			1.90	26.95				
51	2.05	4.31	2.03	15.26						
52	2.09	4.36								
53	2.20	4.66								
54	2.36	4.32							2.27	43.00
55	2.37	4.05	2.36	14.25			2.42	32.87		
56	2.49	3.93								
57	2.65	3.83			2.52	23.92				
58	2.65	3.77								
59	2.77	3.91	2.74	13.00						
60	2.88	3.75								
61	2.97	3.78								
62	3.05	3.55								
63	3.13	3.70	3.08	12.33						
64	3.21	3.44			3.18	21.83				
65	3.40	3.24					3.43	28.78		
66	3.42	3.33								
67	3.45	3.34	3.60	11.37					3.54	36.13
68	3.52	3.31								
69	3.72	3.14								
70	3.77	3.21								
71	3.95	3.15	3.97	11.04	3.96	19.19				

Tabla E.10: Aprendizaje incremental en la base de datos Usps, para muestras, a partir de IDCV-GSO.

Usps, IDCV-GSO										
m_j	$n_j = 1$		$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
7	0.01		0.01		0.01		0.02		0.01	
8	0.02	33.46								
9	0.02	29.00	0.02	42.86						
10	0.03	30.04			0.03	62.10				
11	0.03	23.82	0.04	37.32			0.04	65.80		
12	0.04	18.77							0.05	72.08
13	0.05	18.39	0.05	29.33	0.05	48.76				
14	0.06	16.95								
15	0.07	13.10	0.07	26.67			0.07	49.92		
16	0.08	13.95			0.08	36.58				
17	0.09	13.22	0.09	23.48					0.09	54.22
18	0.10	11.63								
19	0.11	11.48	0.11	22.43	0.11	38.76	0.11	40.49		
20	0.13	10.61								
21	0.14	10.21	0.14	20.74						
22	0.15	10.69			0.15	26.37			0.16	38.91

Tabla E.11: Aprendizaje incremental de la base de datos AR, para nuevas clases, a partir de IDCV-GSO.

AR, IDCV-GSO								
c_x	$c_y = 1$		$c_y = 3$		$c_y = 6$		$c_y = 8$	
	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %
26	55.41		55.64		57.16		55.85	
27	59.28	7.98						
28	62.16	8.14						
29	69.28	7.17	69.56	20.13				
30	74.44	6.85						
31	78.39	6.81						
32	83.24	6.93	81.99	17.89	86.38	35.57		
33	87.86	6.51						
34	91.67	6.37					90.75	42.35
35	97.85	6.29	100.81	17.55				
36	104.07	6.05						
37	107.78	6.21						
38	116.18	5.65	117.02	15.45	121.32	29.55		
39	123.67	5.52						
40	128.83	5.42						
41	133.81	5.49	134.05	15.08				
42	142.89	5.15					144.93	34.38
43	147.91	4.96						
44	155.94	5.08	157.42	13.94	160.03	25.95		
45	161.59	4.95						
46	172.05	4.72						
47	181.44	4.62	170.31	13.46				
48	185.39	4.52						
49	195.09	4.67						
50	205.79	4.35	197.78	12.28	206.41	23.04	198.26	30.04

Tabla E.12: Aprendizaje incremental de la base de datos ORL, para nuevas clases, a partir de IDCV-GSO.

ORL, IDCV-GSO								
c_x	$c_y = 1$		$c_y = 3$		$c_y = 6$		$c_y = 8$	
	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %
16	0.69		0.69		0.69		0.74	
17	0.78	13.78						
18	0.87	12.79						
19	0.96	11.43	0.99	28.27				
20	1.04	12.03						
21	1.15	11.33						
22	1.25	11.14	1.31	24.49	1.25	49.67		
23	1.35	10.37						
24	1.50	9.51					1.52	53.49
25	1.65	8.87	1.63	21.79				
26	1.75	8.48						
27	1.92	8.52						
28	1.97	8.25	1.97	20.26	2.02	36.76		
29	2.12	8.09						
30	2.26	8.13						
31	2.46	7.37	2.47	18.53				
32	2.67	7.06					2.72	43.22
33	2.79	6.63						
34	2.84	7.03	2.90	17.20	2.84	32.77		
35	3.12	6.35						
36	3.29	6.73						
37	3.54	6.28	3.49	15.65				
38	3.63	6.05						
39	3.78	6.06						
40	3.97	5.88	4.04	14.34	4.02	26.55	3.95	38.37

E.3. Validación IDCV-O

Las tablas E.13, E.14, E.15 y E.16 muestran los resultados de validar el método IDCV-O respecto a DCV-GSO, para nuevas muestras de entrenamiento en las bases de datos Cmu-Pie, Coil-20, Mnist y Usps, respectivamente. Los resultados respecto a nuevas clases están en las tablas E.17 y E.18, para las bases de datos AR y ORL, respectivamente.

Tabla E.13: Aprendizaje incremental en la base de datos Cmu-Pie, cuando muestras se incorporan a la base de conocimiento a partir de IDCV-O. En este caso el algoritmo base de IDCV-O es DCV-GSO.

Cmu-Pie, IDCV-O										
m_j	$n_j = 1$		$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
8	38.16		38.00		37.47		36.00		38.63	
9	48.32	2.41								
10	60.00	2.01	60.31	6.20						
11	73.77	1.69			74.77	10.43				
12	87.94	1.46	90.94	4.22			88.91	15.10		
13	106.23	1.27							113.16	18.54
14	126.44	1.07	123.18	3.20	128.43	6.28				
15	142.53	0.99								
16	168.49	0.85	160.87	2.56			168.40	8.25		
17	185.40	0.79			194.05	4.26				
18	205.83	0.73	218.21	1.92					211.45	10.18
19	240.74	0.65								
20	264.87	0.60	274.19	1.58	278.43	3.08	273.35	5.10		
21	291.15	0.57								
22	322.17	0.52	341.55	1.42						
23	356.14	0.48			366.01	2.46			377.13	6.07
24	391.97	0.45	426.93	1.11			389.06	3.78		
25	417.75	0.43								
26	453.77	0.41	489.73	0.99	473.86	1.94				
27	494.03	0.38								
28	540.37	0.35	533.57	0.92			534.46	2.84	555.68	4.12
29	579.39	0.34			594.85	1.58				
30	623.60	0.32	616.42	0.82						
31	679.90	0.30								
32	722.11	0.29	700.69	0.74	718.93	1.33	694.25	2.21		
33	769.56	0.28							785.20	3.06
34	802.61	0.27	793.25	0.67						
35	846.46	0.26			842.24	1.22				
36	898.30	0.25	895.86	0.60			881.70	1.84		
37	944.86	0.25								
38	1006.02	0.23	970.63	0.58	1017.69	0.99			1053.09	2.35
39	1049.23	0.23								
40	1107.52	0.22	1061.62	0.53			1084.93	1.55		

Tabla E.14: Aprendizaje incremental en la base de datos Coil-20, cuando muestras se incorporan a la base de conocimiento a partir de IDCV-O. En este caso el algoritmo base de IDCV-O es DCV-GSO.

Coil-20, IDCV-O										
m_j	$n_j = 1$		$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
11	4.86		4.98		5.27		5.03		5.00	
12	5.99	2.68								
13	7.07	2.42	7.05	5.06						
14	9.30	1.86			9.94	7.36				
15	11.34	2.00	10.94	3.37			11.16	9.94		
16	12.67	1.89							12.02	12.78
17	14.52	1.72	14.62	3.06	14.67	5.11				
18	16.16	1.50								
19	18.19	1.42	17.58	2.61			17.89	6.60		
20	20.63	1.27			19.83	3.84				
21	22.11	1.22	22.16	2.15					21.37	8.37
22	23.81	1.26								
23	27.22	1.10	26.78	1.88	27.42	3.01	26.38	4.50		
24	28.01	1.13								
25	31.13	1.06	31.08	1.72						
26	33.37	0.99			34.38	2.50			33.27	5.32
27	37.28	0.88	36.82	1.46			37.75	3.36		
28	40.65	0.91								
29	43.23	0.87	42.48	1.40	44.71	2.03				
30	46.05	0.79								
31	47.60	0.79	47.87	1.22			49.12	2.75	49.38	3.85
32	52.67	0.73			51.29	1.89				
33	56.19	0.70	57.36	1.09						
34	59.36	0.72								
35	62.53	0.70	63.57	1.00	63.72	1.52	63.56	2.21		
36	65.79	0.68							66.06	2.91
37	71.29	0.62	72.63	0.93						
38	72.43	0.62			77.23	1.34				
39	78.00	0.58	77.24	0.87			79.22	1.79		
40	81.98	0.56								
41	86.28	0.57	86.55	0.82	87.88	1.25			84.82	2.40
42	89.97	0.55								
43	95.55	0.52	95.35	0.76			93.72	1.57		
44	102.80	0.49			104.57	1.05				
45	104.08	0.50	102.79	0.72						
46	111.93	0.47							107.45	1.90
47	116.12	0.48	115.75	0.66	118.90	0.96	113.12	1.38		
48	119.38	0.47								
49	123.80	0.47	123.94	0.66						
50	128.63	0.44			132.30	0.87				
51	137.30	0.42	139.58	0.59			130.92	1.30	136.08	1.62

Tabla E.15: Aprendizaje incremental en la base de datos Mnist, cuando muestras se incorporan a la base de conocimiento a partir de IDCV-O. En este caso el algoritmo base de IDCV-O es DCV-GSO.

Mnist, IDCV-O										
m_j	$n_j = 1$		$n_j = 4$, CPU		$n_j = 7$, CPU		$n_j = 10$, CPU		$n_j = 13$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
15	0.15		0.16		0.15		0.16		0.16	
16	0.17	5.31								
17	0.19	4.10								
18	0.22	4.36								
19	0.25	4.96	0.25	11.95						
20	0.28	3.99								
21	0.31	3.45								
22	0.34	2.85			0.35	15.32				
23	0.37	4.62	0.38	7.42						
24	0.41	3.77								
25	0.45	3.46					0.49	22.33		
26	0.49	2.59								
27	0.53	2.38	0.52	5.92						
28	0.57	3.27							0.58	28.68
29	0.62	1.85			0.63	9.02				
30	0.67	2.22								
31	0.73	2.36	0.70	3.55						
32	0.75	1.88								
33	0.80	1.84								
34	0.86	1.49								
35	0.94	1.75	0.89	3.21			0.92	11.01		
36	0.97	1.43			0.99	5.86				
37	1.05	1.44								
38	1.07	1.10								
39	1.17	1.57	1.13	2.61						
40	1.26	1.18								
41	1.30	1.12							1.31	13.24
42	1.33	1.45								
43	1.44	1.37	1.41	2.23	1.42	4.19				
44	1.49	0.86								
45	1.54	1.18					1.55	7.24		
46	1.60	0.98								
47	1.70	1.06	1.70	2.05						
48	1.78	1.09								
49	1.81	0.80								
50	1.98	1.13			1.90	3.26				
51	2.05	0.78	2.03	1.58						
52	2.09	0.90								
53	2.20	0.79								
54	2.36	0.89							2.27	7.88
55	2.37	0.91	2.36	1.55			2.42	5.31		
56	2.49	0.78								
57	2.65	1.11			2.52	2.58				
58	2.65	0.87								
59	2.77	0.72	2.74	1.46						
60	2.88	0.79								
61	2.97	0.66								
62	3.05	0.60								
63	3.13	0.74	3.08	1.25						
64	3.21	0.57			3.18	2.14				
65	3.40	0.49					3.43	3.83		
66	3.42	0.58								
67	3.45	0.55	3.60	1.17					3.54	5.31
68	3.52	0.58								
69	3.72	0.52								
70	3.77	0.54								
71	3.95	0.46	3.97	1.02	3.96	1.79				

Tabla E.16: Aprendizaje incremental en la base de datos Usps, cuando muestras se incorporan a la base de conocimiento a partir de IDCV-O. En este caso el algoritmo base de IDCV-O es DCV-GSO.

Usps, IDCV-O										
m_j	$n_j = 1$		$n_j = 2$, CPU		$n_j = 3$, CPU		$n_j = 4$, CPU		$n_j = 5$, CPU	
	CPU(seg.)	CPU %	seg.	%	seg.	%	seg.	%	seg.	%
7	0.01		0.01		0.01		0.02		0.01	
8	0.02	17.91								
9	0.02	5.92	0.02	11.26						
10	0.03	4.83			0.03	24.49				
11	0.03	3.78	0.04	9.25			0.04	21.79		
12	0.04	3.40							0.05	29.69
13	0.05	4.31	0.05	4.81	0.05	11.93				
14	0.06	3.47								
15	0.07	6.71	0.07	4.05			0.07	11.84		
16	0.08	4.69			0.08	6.42				
17	0.09	3.47	0.09	3.82					0.09	13.34
18	0.10	3.45								
19	0.11	3.39	0.11	2.42	0.11	6.37	0.11	8.66		
20	0.13	2.61								
21	0.14	3.67	0.14	2.63						
22	0.15	1.30			0.15	4.08			0.16	9.49

Tabla E.17: Aprendizaje incremental de la base de datos AR, cuando nuevas clases se incorporan a la base de conocimiento a partir de IDCV-O. En este caso el algoritmo base de IDCV-O es DCV-GSO.

AR, IDCV-O								
c_x	$c_y = 1$		$c_y = 3$		$c_y = 6$		$c_y = 8$	
	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %
26	55.41		55.64		57.16		55.85	
27	59.28	0.82						
28	62.16	0.82						
29	69.28	0.72	69.56	2.00				
30	74.44	0.82						
31	78.39	0.73						
32	83.24	0.74	81.99	1.92	86.38	4.85		
33	87.86	0.67						
34	91.67	0.69					90.75	7.08
35	97.85	0.63	100.81	1.60				
36	104.07	0.61						
37	107.78	0.64						
38	116.18	0.59	117.02	1.41	121.32	3.48		
39	123.67	0.54						
40	128.83	0.51						
41	133.81	0.50	134.05	1.24				
42	142.89	0.46					144.93	4.52
43	147.91	0.51						
44	155.94	0.48	157.42	1.08	160.03	2.81		
45	161.59	0.49						
46	172.05	0.47						
47	181.44	0.43	170.31	1.07				
48	185.39	0.43						
49	195.09	0.44						
50	205.79	0.40	197.78	0.94	206.41	2.20	198.26	3.48

Tabla E.18: Aprendizaje incremental de la base de datos ORL, cuando nuevas clases se incorporan a la base de conocimiento a partir de IDCV-O. En este caso el algoritmo base de IDCV-O es DCV-GSO.

ORL, IDCV-O								
c_x	$c_y = 1$		$c_y = 3$		$c_y = 6$		$c_y = 8$	
	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %	CPU (seg.)	CPU %
16	0.69		0.69		0.69		0.74	
17	0.78	4.11						
18	0.87	3.87						
19	0.96	4.17	0.99	7.23				
20	1.04	3.42						
21	1.15	3.40						
22	1.25	2.77	1.31	6.52	1.25	13.05		
23	1.35	3.07						
24	1.50	2.65					1.52	16.80
25	1.65	2.48	1.63	4.86				
26	1.75	2.43						
27	1.92	2.15						
28	1.97	2.02	1.97	4.56	2.02	8.33		
29	2.12	2.03						
30	2.26	2.07						
31	2.46	1.98	2.47	3.70				
32	2.67	1.76					2.72	9.80
33	2.79	1.58						
34	2.84	1.63	2.90	3.31	2.84	6.88		
35	3.12	1.55						
36	3.29	1.45						
37	3.54	1.33	3.49	2.92				
38	3.63	1.37						
39	3.78	1.23						
40	3.97	1.28	4.04	2.62	4.02	5.97	3.95	7.57

E.4. Validación IRDCV

Las tablas E.21 y E.26 presentan los resultados de validar el método IRDCV, cuando nuevas muestras se incorporan a la base de conocimiento de Mnist y Usps, respectivamente. En Mnist α =0.05. En Usps α =0.10. m_j y n_j son el número de muestras por clase en el entrenamiento por lotes e incremental, respectivamente. $(1 - e)$ es la tasa de acierto. CPU (seg.) y std(CPU) es el tiempo de entrenamiento y la desviación estándar en segundos del método RDCV. CPU % y std(CPU %) es el tiempo de CPU relativo y la desviación estándar del algoritmo incremental respecto al algoritmo original. El primer modelo precalculado se obtiene a partir del método original RDCV.

Tabla E.19: Aprendizaje incremental en la base de datos Mnist. Con $n_j = 1$.

Mnist, IRDCV								
	RDCV				IRDCV			
m_j	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
15	0.88	0.02	0.09	0.01				
16	0.89	0.02	0.09	0.01	0.89	0.02	34.04	22.04
17	0.89	0.02	0.10	0.01	0.89	0.02	28.86	8.76
18	0.90	0.02	0.12	0.03	0.90	0.02	22.96	9.96
19	0.90	0.02	0.12	0.01	0.90	0.02	34.72	12.37
20	0.90	0.02	0.12	0.01	0.90	0.02	25.67	5.26
21	0.91	0.02	0.14	0.03	0.90	0.02	19.68	7.43
22	0.91	0.02	0.15	0.01	0.90	0.01	36.83	26.28
23	0.90	0.02	0.17	0.04	0.91	0.01	20.91	13.52
24	0.90	0.02	0.17	0.02	0.90	0.02	20.17	9.78
25	0.90	0.02	0.19	0.02	0.91	0.02	17.09	8.88
26	0.91	0.02	0.20	0.01	0.91	0.02	17.66	4.09
27	0.91	0.01	0.23	0.05	0.91	0.02	13.73	4.53
28	0.91	0.01	0.23	0.02	0.91	0.02	16.19	8.90
29	0.91	0.01	0.25	0.03	0.91	0.02	15.33	8.22
30	0.92	0.01	0.27	0.02	0.91	0.01	19.54	11.57
31	0.92	0.01	0.30	0.03	0.91	0.01	14.83	4.34
32	0.91	0.02	0.30	0.01	0.91	0.01	14.73	5.85
33	0.91	0.01	0.34	0.04	0.91	0.01	12.11	2.97
34	0.91	0.02	0.37	0.03	0.91	0.02	13.70	7.36
35	0.91	0.02	0.38	0.03	0.91	0.01	10.86	3.07
36	0.91	0.02	0.39	0.01	0.91	0.02	11.34	3.66
37	0.91	0.02	0.42	0.02	0.91	0.01	9.57	1.74
38	0.91	0.01	0.49	0.03	0.92	0.01	8.25	1.44
39	0.91	0.01	0.53	0.03	0.92	0.01	10.50	4.82
40	0.91	0.01	0.59	0.04	0.92	0.01	7.59	1.36
41	0.92	0.02	0.63	0.04	0.92	0.01	7.00	1.49
42	0.92	0.01	0.66	0.04	0.92	0.02	7.02	1.76
43	0.92	0.02	0.71	0.04	0.92	0.01	6.76	1.28
44	0.92	0.01	0.74	0.04	0.92	0.01	6.76	1.94
45	0.92	0.01	0.80	0.06	0.92	0.01	5.71	0.80
46	0.92	0.01	0.82	0.05	0.92	0.01	5.91	1.02
47	0.92	0.01	0.86	0.04	0.92	0.01	6.26	1.97
48	0.92	0.01	0.93	0.04	0.92	0.01	6.26	2.34
49	0.92	0.01	0.99	0.06	0.92	0.01	5.21	0.88
50	0.92	0.01	1.02	0.06	0.92	0.01	4.69	1.16
51	0.91	0.01	1.12	0.09	0.92	0.01	5.95	3.16
52	0.92	0.02	1.32	0.05	0.92	0.01	4.58	0.70
53	0.92	0.01	1.32	0.09	0.92	0.01	5.20	1.38
54	0.92	0.01	1.39	0.12	0.92	0.01	5.81	1.53
55	0.92	0.01	1.45	0.10	0.92	0.01	4.94	1.30
56	0.92	0.01	1.52	0.12	0.92	0.01	4.49	0.95
57	0.92	0.01	1.60	0.10	0.92	0.01	4.41	0.82
58	0.92	0.01	1.70	0.11	0.92	0.01	4.09	0.83
59	0.92	0.01	1.77	0.11	0.92	0.01	3.82	0.99
60	0.92	0.01	1.85	0.17	0.92	0.02	4.23	1.40
61	0.93	0.01	1.90	0.17	0.92	0.01	3.67	0.93
62	0.93	0.01	1.99	0.15	0.92	0.01	3.33	0.68
63	0.92	0.01	2.03	0.14	0.92	0.02	3.48	0.85
64	0.93	0.01	2.09	0.15	0.93	0.01	3.31	0.49
65	0.93	0.01	2.18	0.15	0.92	0.01	3.23	1.05
66	0.93	0.01	2.42	0.14	0.92	0.01	3.31	0.84
67	0.93	0.01	2.41	0.15	0.92	0.01	3.01	0.54
68	0.93	0.01	2.59	0.20	0.92	0.01	2.83	0.90
69	0.93	0.01	2.69	0.16	0.92	0.01	3.66	1.68
70	0.93	0.01	2.77	0.18	0.92	0.01	2.41	0.65
71	0.93	0.01	2.87	0.19	0.92	0.01	2.30	0.34

Tabla E.20: Aprendizaje incremental en la base de datos Mnist. Con $n_j = 4$.

Mnist, IRDCV								
	RDCV				IRDCV			
m_j	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
15	0.89	0.02	0.11	0.04				
19	0.91	0.02	0.12	0.02	0.91	0.02	60.46	19.39
23	0.91	0.01	0.16	0.01	0.91	0.02	50.17	10.12
27	0.92	0.01	0.23	0.05	0.92	0.01	34.05	7.13
31	0.92	0.02	0.28	0.01	0.92	0.01	35.39	9.64
35	0.91	0.02	0.38	0.04	0.92	0.01	23.95	4.13
39	0.91	0.01	0.52	0.04	0.92	0.01	18.19	2.66
43	0.92	0.02	0.68	0.05	0.92	0.02	15.90	4.63
47	0.92	0.01	0.89	0.07	0.92	0.01	12.90	4.07
51	0.92	0.02	1.13	0.07	0.92	0.01	14.07	6.90
55	0.92	0.02	1.35	0.09	0.93	0.02	11.25	3.87
59	0.93	0.01	1.67	0.12	0.93	0.01	9.25	2.15
63	0.92	0.02	1.97	0.10	0.92	0.01	8.22	1.87
67	0.93	0.01	2.36	0.13	0.92	0.01	7.02	1.73
71	0.92	0.01	2.80	0.23	0.92	0.01	6.38	1.52

Tabla E.21: Aprendizaje incremental en la base de datos Mnist. Con $n_j = 7$.

Mnist, IRDCV								
	RDCV				IRDCV			
m_j	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
15	0.89	0.00	0.08	0.01				
22	0.91	0.01	0.15	0.01	0.91	0.01	86.18	7.15
29	0.91	0.01	0.25	0.01	0.91	0.01	52.36	3.44
36	0.91	0.01	0.39	0.01	0.92	0.01	37.66	1.96
43	0.92	0.01	0.70	0.03	0.92	0.01	21.86	1.59
50	0.92	0.00	1.08	0.06	0.93	0.01	17.58	1.30
57	0.93	0.00	1.54	0.09	0.93	0.01	11.13	0.83
64	0.93	0.01	2.13	0.16	0.93	0.01	8.59	0.75
71	0.93	0.01	2.84	0.18	0.93	0.01	6.53	0.60

Tabla E.22: Aprendizaje incremental en la base de datos Mnist. Con $n_j = 10$.

Mnist, IRDCV								
	RDCV				IRDCV			
m_j	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
15	0.90	0.02	0.10	0.02				
25	0.92	0.02	0.21	0.02	0.92	0.02	115.51	19.77
35	0.92	0.02	0.42	0.02	0.92	0.02	67.41	9.43
45	0.92	0.02	0.85	0.04	0.93	0.01	33.62	3.71
55	0.93	0.01	1.48	0.11	0.93	0.02	21.85	2.76
65	0.93	0.01	2.28	0.17	0.93	0.01	13.87	1.93

Tabla E.23: Aprendizaje incremental en la base de datos Mnist. Con $n_j = 13$.

Mnist, IRDCV								
	RDCV				IRDCV			
m_j	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
15	0.89	0.02	0.10	0.02				
28	0.92	0.01	0.26	0.03	0.92	0.02	133.46	19.27
41	0.92	0.02	0.64	0.03	0.93	0.02	55.77	4.03
54	0.93	0.02	1.34	0.07	0.93	0.02	28.48	3.20
67	0.93	0.02	2.33	0.13	0.93	0.01	17.18	1.59

Tabla E.24: Aprendizaje incremental en la base de datos Usps. Con $n_j = 1$.

Usps, IRDCV								
m_j	RDCV				IRDCV			
	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
7	0.67	0.08	0.02	0.01				
8	0.70	0.08	0.03	0.01	0.69	0.08	29.75	26.49
9	0.67	0.09	0.03	0.01	0.70	0.09	39.78	27.37
10	0.71	0.09	0.03	0.01	0.70	0.10	26.73	18.04
11	0.68	0.10	0.03	0.00	0.70	0.10	23.33	13.41
12	0.70	0.08	0.05	0.02	0.71	0.10	18.37	9.21
13	0.70	0.09	0.04	0.01	0.70	0.10	24.29	13.15
14	0.72	0.10	0.05	0.01	0.72	0.11	22.47	12.15
15	0.72	0.06	0.05	0.01	0.73	0.08	21.54	9.78
16	0.72	0.07	0.06	0.01	0.74	0.07	18.41	6.25
17	0.71	0.06	0.07	0.01	0.72	0.08	15.07	5.65
18	0.73	0.08	0.08	0.01	0.71	0.08	14.70	7.08
19	0.71	0.06	0.09	0.02	0.72	0.08	16.81	9.80
20	0.70	0.07	0.10	0.01	0.72	0.08	10.99	7.49
21	0.71	0.04	0.10	0.01	0.71	0.07	12.48	6.87
22	0.70	0.06	0.12	0.01	0.71	0.07	8.87	3.36

Tabla E.25: Aprendizaje incremental en la base de datos Usps. Con $n_j = 2$.

Usps, IRDCV								
m_j	RDCV				IRDCV			
	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
7	0.67	0.07	0.02	0.01				
9	0.68	0.08	0.03	0.01	0.69	0.06	77.02	44.07
11	0.69	0.05	0.03	0.01	0.69	0.03	76.03	28.24
13	0.70	0.08	0.04	0.01	0.70	0.05	59.29	8.50
15	0.72	0.07	0.05	0.01	0.70	0.07	48.83	7.51
17	0.72	0.11	0.06	0.01	0.72	0.08	44.36	9.90
19	0.71	0.09	0.08	0.01	0.72	0.08	36.25	7.65
21	0.72	0.12	0.09	0.01	0.71	0.09	33.46	4.80

Tabla E.26: Aprendizaje incremental en la base de datos Usps. Con $n_j = 3$.

Usps, IRDCV								
m_j	RDCV				IRDCV			
	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
7	0.81	0.01	0.02	0.01				
10	0.83	0.01	0.03	0.01	0.84	0.01	67.44	16.97
13	0.85	0.01	0.04	0.01	0.85	0.01	67.52	17.04
16	0.87	0.01	0.06	0.01	0.87	0.01	50.21	11.84
19	0.88	0.02	0.08	0.01	0.87	0.01	36.21	7.87
22	0.90	0.01	0.10	0.01	0.89	0.01	29.37	7.87

Tabla E.27: Aprendizaje incremental en la base de datos Usps. Con $n_j = 4$.

Usps, IRDCV								
m_j	RDCV				IRDCV			
	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
7	0.69	0.08	0.02	0.01				
11	0.73	0.08	0.03	0.01	0.71	0.08	97.09	38.85
15	0.73	0.08	0.05	0.01	0.72	0.06	88.23	22.20
19	0.70	0.06	0.08	0.01	0.68	0.04	66.77	19.95

Tabla E.28: Aprendizaje incremental en la base de datos Usps. Con $n_j = 5$.

Usps, IRDCV								
m_j	RDCV				IRDCV			
	$(1-e)$	std$(1-e)$	CPU (seg.)	std(CPU)	$(1-e)$	std$(1-e)$	CPU %	std(CPU %)
7	0.68	0.07	0.02	0.01				
12	0.70	0.06	0.03	0.01	0.72	0.08	143.96	36.02
17	0.72	0.08	0.07	0.01	0.71	0.08	95.69	22.08
22	0.70	0.04	0.12	0.02	0.71	0.07	56.25	14.16

Bibliografía

[1] Mark Aronovich Aizerman, E.A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control*, number 25 in Automation and Remote Control, pages 821–837, 1964.

[2] Shotaro Akaho. A kernel method for canonical correlation analysis. In *In Proceedings of the International Meeting of the Psychometric Society (IMPS2001)*. Springer-Verlag, 2001.

[3] Enrique Alexandre-Cortizo, Manuel Rosa-Zurera, and Francisco Lopez-Ferreras. Application of fisher linear discriminant analysis to speech/music classification. In *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, volume 2, pages 1666 –1669, nov. 2005.

[4] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

[5] Joaquim Arlandis, Juan C. Perez-Cortes, and Rafael Llobet. Handwritten character recognition using the continuous distance transformation. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 940–943, 2000.

[6] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *Journal of Machine Learning*, 3:1–48, 2002.

[7] Reinhold Baer. *Linear algebra and projective geometry*. Academic Press, New York, 1965.

[8] Eric Bair, Trevor Hastie, Debashis Paul, and Robert Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101:119–137, 2006.

[9] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, December 2010.

[10] Juan Bekios, José Miguel Buenaposada, and Luis Baumela. Revisiting linear discriminant techniques in gender recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(4):858–864, 2011.

[11] Peter N. Belhumeur, João Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[12] Tijl De Bie, Nello Cristianini, and Roman Rosipal. Eigenproblems in pattern recognition. In *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics*, pages 129–170. Springer, 2005.

[13] Yu Bing, Jin Lianfu, and Chen Ping. A new lda-based method for face recognition. *Pattern Recognition, International Conference on*, 1:10168, 2002.

[14] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. United States by Oxford University, 1996.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[16] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM Press, 1992.

[17] James R. Bunch and Christopher P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31:111–129, 1978.

[18] James R. Bunch, Christopher P. Nielsen, and Danny C. Sorenson. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31:31–48, 1978.

[19] Hakan Cevikalp, Marian Neamtu, and Atalay Barkana. The kernel common vector method: A novel nonlinear subspace classifier for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(4):937–951, 2007.

[20] Hakan Cevikalp, Marian Neamtu, and Mitch Wilkes. Discriminative common vector method with kernels. *IEEE Transactions on Neural Networks*, 17(6):1550–1565, 2006.

[21] Hakan Cevikalp, Marian Neamtu, Mitch Wilkes, and Atalay Barkana. A novel method for face recognition. In *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th*, pages 579–582, 2004.

[22] Hakan Cevikalp, Marian Neamtu, Mitch Wilkes, and Atalay Barkana. Discriminative common vectors for face recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(1):4–13, 2005.

[23] Li-Fen Chen, Hong-Yuan Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33(10):1713–1726, October 2000.

[24] Yong-Qing Cheng, Yong-Ming Zhuang, and Jing-Yu Yang. Optimal fisher discriminate analysis using the rank decomposition. *Pattern Recognition*, 25:101–111, 1992.

[25] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[26] Marcos del Pozo-Baños, Carlos M. Travieso, Jesús B. Alonso, and Miguel A. Ferrer. Discriminative multi-projection vectors: Modifying the discriminative common vectors approach for face verification. In *Security Technology (ICCST), 2010 IEEE International Carnahan Conference on*, pages 190–197, oct. 2010.

[27] Katerine Diaz-Chito and Francesc J. Ferri. Empirical evaluation of class-updating incremental discriminative common vector based face recognition. In *II Workshop de Reconocimiento de Formas y Análisis de Imágenes (AERFAI), CEDI2010.*, pages 125–132, 2010.

[28] Katerine Diaz-Chito and Francesc J. Ferri. Incremental discriminative common vectors. Technical report, 2010.

[29] Katerine Diaz-Chito, Francesc J. Ferri, and Wladimiro Díaz-Villanueva. An empirical evaluation of common vector based classification methods and some extensions. In *SSPR/SPR*, pages 977–985, 2008.

[30] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[31] Katerine Díaz-Chito, Francesc J. Ferri, and Wladimiro Díaz-Villanueva. Extracción de características mediante vectores comunes discriminantes extendidos con kernel. *Revista Iberoamericana de Inteligencia Artificia*, 13(43):1–15, 2009.

[32] Q. Emad ul Haq, M.Y. Javed, and Q. Sami ul Haq. Efficient and robust approach of iris recognition through fisher linear discriminant analysis method and principal component analysis method. In *Multitopic Conference, 2008. INMIC 2008. IEEE International*, pages 218 –225, dec. 2008.

[33] S. Ergin and Bilginer M. Gülmezoglu. Face recognition based on face partitions using common vector approach. In *Communications,*

Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on, pages 624–628, 2008.

[34] Francesc J. Ferri, Katerine Díaz-Chito, and Wladimiro Díaz-Villanueva. Using subspace-based learning methods for medical drug design and characterization. In *IEEE Intl. Conf on Systems, Man and Cybernetics, SMC 2008*, pages 2111–2115, 2008.

[35] Ronald Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[36] Evelyn Fix and J.L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989.

[37] Annalisa Franco, Alessandra Lumini, and Dario Maio. Eigenspace merging for model updating. In *International Conference on Pattern Recognition, ICPR (2)*, pages 156–159, 2002.

[38] Jerome H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[39] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2 edition, 1990.

[40] A.S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[41] Philip E. Gill, G. H. Golub, Walter Murray, and Michael A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974.

[42] Luc Giraud, Julien Langou, and Miroslav Rozloznik. The loss of orthogonality in the gram-schmidt orthogonalization process. *Comput. Math. Appl.*, 50:1069–1075, October 2005.

[43] Ming Gu and Stanley C. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Technical Report YALEU/DCS/RR-966, Dept. of Computer Science, Yale University, New Haven, CT, jun 1993.

[44] Bilginer M. Gulmezoglu, Vakıf Dzhafarov, M. Keskin, and Atalay Barkana. A novel approach to isolated word recognition. *IEEE Trans. Speech and Audio Processing*, 7(6):620–618, 1999.

[45] Bilginer M. Gülmezoglu, Vakıf Dzhafarov, and Atalay Barkana. The common vector approach and its relation to principal component analysis. *IEEE Trans. Speech and Audio Processing*, 9(6):655–662, 2001.

[46] Witten Ian H. and Frank Eibe. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, 2005.

[47] Zhao Hai-long, Mu Zhi-chun, Xia Zhang, and Dun Wen-jie. Ear recognition based on wavelet transform and discriminative common vectors. In *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*, volume 1, pages 713–716. IEEE Computer Society, 2008.

[48] Li Haifeng, Jiang Tao, and Zhang Keshu. Efficient and robust feature extraction by maximum margin criterion. In *In Advances in Neural Information Processing Systems, NIPS*, 2003.

[49] Lu Haiping, Konstantinos N. Plataniotis, and Anastasios Venetsanopoulos. Mpca: Multilinear principal component analysis of tensor objects. *IEEE Transactions on Neural Networks*, 19(1):18–39, 2008.

[50] Peter Hall, David Marshall, and Ralph Martin. Incremental eigenanalysis for classification. In *in British Machine Vision Conference*, pages 286–295, 1998.

[51] Peter Hall, David Marshall, and Ralph Martin. Merging and splitting eigenspace models. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

[52] David R. Hardoon, Sandor R. Szedmak, and John R. Shawe-taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.*, 16:2639–2664, December 2004.

[53] Yunhui He, Li Zhao, and Cairong Zou. Face recognition using common faces method. *Pattern Recognition*, 39(11):2218–2222, 2006.

[54] Zi Quan Hong and Jing Yu Yang. Optimal discriminant plane for a small number of samples and design method of classifier on the plane. *Pattern Recognition*, 24:317–324, 1991.

[55] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psycholog*, 24:417–441 and 498–520, 1933.

[56] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, December 1936.

[57] Rui Huang, Qingshan Liu, Hanqing Lu, and Songde Ma. Solving the small sample size problem of lda. In *In: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02*, pages 29–32, 2002.

[58] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:550–554, 1994.

[59] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[60] Ye Jieping. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6:483–502, 2005.

[61] Ye Jieping and Tao Xiong. Computational and theoretical analysis of null space and orthogonal linear discriminant analysis. *Journal of Machine Learning Research*, 7:1183–1204, 2006.

[62] I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, NY, USA, 2002.

[63] Lu Juwei, Kostantinos N. Plataniotis, and Anastasios Venetsanopoulos. Regularized discriminant analysis for the small sample size problem in face recognition. *Pattern Recogn. Lett.*, 24(16):3079–3087, 2003.

[64] K. Karhunen. Zur spektraltheorie stochastischer prozesse. *Annales Academiae Scientiarum Fennicae, Series A1*, Mathematica-Physica(34):1–7, 1946.

[65] Pearson Karl. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[66] Tae-Kyun Kim, Josef Kittler, and Roberto Cipolla. Discriminative learning and recognition of image set classes using canonical correlations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1005–1018, 2007.

[67] Tae-Kyun Kim, Björn Stenger, Josef Kittler, and Roberto Cipolla. Incremental linear discriminant analysis using sufficient spanning sets and its applications. *International Journal of Computer Vision*, 91(2):216–232, 2011.

[68] Tae-Kyun Kim, Shu-Fai Wong, Björn Stenger, Josef Kittler, and Roberto Cipolla. Incremental linear discriminant analysis using sufficient spanning set approximations. In *CVPR 2007. 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. IEEE Computer Society, 2007.

[69] Mehmet Koç, Atalay Barkana, and Ömer Gerek. A fast method for the implementation of common vector approach. *Journal Information Sciences*, 180:4084–4098, October 2010.

[70] Tae kyun Kim, Josef Kittler, and Roberto Cipolla. Incremental learning of locally orthogonal subspaces for set-based object recognition. bmvc06. In *Proceeding British Machine Vision Conference*, pages 559–568, 2006.

[71] C. Lakshmi, M. Sundararajan, and P. Manikandan. Hierarchical approach of discriminative common vectors for bio metric security. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 2, pages 784–790, feb. 2010.

[72] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 –2324, nov. 1998.

[73] Chih-Jen Lee, Tzu-Yin Chen, Jenn-Dong Sun, Tai-Ning Yang, and A.Y. Chang. Combining gradientfaces, principal component analysis, and fisher linear discriminant for face recognition. In *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, pages 622 –625, aug. 2010.

[74] Yixiong Liang, Weiguo Gong, Yingjun Pan, and Chengrong Li. Generalizing relevance weighted lda. *Pattern Recognition*, 38(11):2217–2219, 2005.

[75] Zhi-Zheng Liang and You-Fu Li. Multiple kernels for generalised discriminant analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 4(2):117–128, June 2010.

[76] Ke Liu, Yong-Qing Cheng, and Jing-Yu Yang. A generalized optimal set of discriminant vectors. *Pattern Recognition*, 25(7):731–739, 1992.

[77] Ke Liu, Yong-Qing Cheng, Jing-Yu Yang, and Xiao Liu. An efficient algorithm for foley-sammon optimal set of discriminant vectors by algebraic method. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 6(5):817–829, 1992.

[78] Xiaoming Liu and Tsuhan Chen. Shot boundary detection using temporal statistics modeling. In *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, volume 4, pages IV–3389–IV–3392, 2002.

[79] M. Loéve. *Probability Theory*. Van Nostrand, 1955.

[80] H. Lu, Konstantinos N. Plataniotis, and Anastasios Venetsanopoulos. A taxonomy of emerging multilinear discriminant analysis solutions for biometric signal recognition. *in Biometrics: Theory, Methods, and Applications*, pages 21–45, 2009.

[81] B. S. Manjunath, Shivkumar Chandrasekaran, and Yuan-Fang Wang. An eigenspace update algorithm for image analysis. In *Proceedings of the International Symposium on Computer Vision*, pages 551–556, Washington, DC, USA, 1995. IEEE Computer Society.

[82] Jianchang Mao and Anil Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 1995.

[83] Aleix Martinez and Robert Benavente. The ar face database. Technical Report 24, Computer Vision Center CVC, 1998.

[84] Yang M.H. Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods. In *In Proc. of IEEE Internation Conf. on Face and Gesture Recognition*, pages 215–220, 2002.

[85] Sebastian Mika, Gunnar Rätsch, and Klaus-Robert Müller. A mathematical programming approach to the kernel fisher algorithm. In *Neural Information Processing Systems, NIPS*, pages 591–597, 2000.

[86] Hiroyasu Murakami and Vijaya Kumar. Efficient calculation of primary images from a set of images. *IEEE Trans. Patt. Analysis and Machine Intell*, 4(5):511–515, 1982.

[87] Sameer Nene, Shree Nayar, and Hiroshi Murase. Columbia object image library (coil-100). Technical report, 1996.

[88] Sameer Nene, Shree Nayar, and Hiroshi Murase. Columbia object image library (coil-20). Technical report, 1996.

[89] Mark Ordowski and Gerard G. L. Meyer. Geometric linear discriminant analysis for pattern recognition. *Pattern Recognition*, 37(3):421–428, 2004.

[90] Seiichi Ozawa, Shaoning Pang, and Nikola K. Kasabov. Incremental learning of chunk data for online pattern classification systems. *IEEE Transactions on Neural Networks*, 19(6):1061–1074, 2008.

[91] Shaoning Pang, Seiichi Ozawa, and Nikola Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(5):905–914, 2005.

[92] Haesun Park and Peg Howl. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:995–1006, 2004.

[93] Volker Roth and V. Steinhage. Nonlinear discriminant analysis using kernel functions. In S. A. Solla, T. K. Leen, and K. R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 568–574. MIT Press, 1999.

[94] Ferdinando Samaria and Andy Harter. Parameterisation of a stochastic model for human face identification. In *Workshop on Applications of Computer Vision, WACV94*, pages 138–142, 1994.

[95] Bernhard Schölkopf, Sebastian Mika, Chris Burges, Phil Knirsch, Klaus-Robert Müller, Gunnar Rätsch, and Alex J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions On Neural Networks*, 10(5):1000–1017, 1999.

[96] Bernhard Schölkopf and Alex J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[97] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 1996.

[98] Nicu Sebe, Ira Cohen, Ashutosh Garg, and Thomas Huang. *Machine Learning in Computer Vision (Computational Imaging and Vision)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[99] John Shawe-Taylor and Cristianini Nello. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[100] Danijel Skočaj and Aleš Leonardis. Incremental approach to robust learning of eigenspaces. In *Vision with non-traditional sensors, 26th Workshop of the Austrian Association for Pattern Recognition*, pages 111–118, Graz, Austria, September 2002.

[101] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974.

[102] Gilbert Strang. *Linear Algebra and Its Applications*. Fourth edition, February 2009.

[103] Ruei sung Lin, David Ross, Jongwoo Lim, and Ming hsuan Yang. Adaptive discriminative generative model and its applications. In *In Proc. Conf. on Neural Information Processing Systems*, pages 801–808. The MIT Press, 2005.

[104] Daniel Swets and John Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[105] Akihiko Tamura and Qiangfu Zhao. Rough common vector: A new approach to face recognition. In *IEEE Intl. Conf. on Syst, Man and Cybernetics,*, pages 2366–2371, 2007.

[106] Hong Tang, Tao Fang, and Peng-Fei Shi. Laplacian linear discriminant analysis. *Pattern Recognition*, 39(1):136–139, 2006.

[107] Sim Terence, Baker Simon, and Bsat Maan. The CMU pose, illumination, and expression (PIE) database. In *Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition*, 2002.

[108] Q. Tian, M. Barbero, Z.H. Gu, and S.H. Lee. Image classification by the foley-sammon transform. *Optical Eng.*, 25(7):834–840, 1986.

[109] Ferdinand van der Heijden, Robert Duin, Dick de Ridder, and David M. J. Tax. *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. Wiley, 2004.

[110] Vladimir Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[111] Vladimir Vapnik and Lerner A. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[112] Xiaogang Wang and Xiaoou Tang. Dual-space linear discriminant analysis for face recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 2, pages 564–569, 2004.

[113] Yuan-Kai Wang and Chun-Hao Huang. Facial expression recognition with discriminative common vector. In *Intelligent Information Hiding and Multimedia Signal Processing, International Conference on*, volume 2, pages 431–434, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[114] Zhan Wang and Qiuqi Ruan. Facial expression recognition based orthogonal local fisher discriminant analysis. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 1358 –1361, oct. 2010.

[115] Harry Wechsler, Jonathon P. Phillips, Vicki Bruce, and Thomas S. (eds) Fogelman, Francoise (eds) Huang. Face recognition: From theory to applications. *NATO ASI Series F, Computer and Systems Sciences*, 163:446–456, 1998.

[116] Juyang Weng, Yilu Zhang, and Wey-Shiuan Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1034–1040, 2003.

[117] Zheng Wenming and Tang Xiaoou. Fast algorithm for updating the discriminant vectors of dual-space lda. *IEEE Transactions on Information Forensics and Security*, 4(3):418–427, 2009.

[118] Xinxiao Wu, Yunde Jia, and Wei Liang. Incremental discriminant-analysis of canonical correlations for action recognition. *Pattern Recognition.*, 43:4190–4197, December 2010.

[119] Yang Xiaochao, Dai Ji, Zhou Yue, and Yang Jie. Gabor-based discriminative common vectors for gait recognition. *Image and Signal Processing, Congress on*, 4:191–195, 2008.

[120] Yong Xu, David Zhang, Zhong Jin, Miao Li, and Jing-Yu Yang. A fast kernel-based nonlinear discriminant analysis for multi-class problems. *Pattern Recognition*, 39(6):1026 – 1033, 2006.

[121] Shuicheng Yan, Dong Xu, Qiang Yang, Lei Zhang 0001, Xiaoou Tang, and HongJiang Zhang. Multilinear discriminant analysis for face recognition. *IEEE Transactions on Image Processing*, 16(1):212–220, 2007.

[122] Jian Yang, A. F. Frangi, Jing-Yu Yang, David Zhang, and Zhong Jin. Kpca plus lda: a complete kernel fisher discriminant framework for feature extraction and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(2):230–244, 2005.

[123] Jian Yang, Jing-Yu Yang, David Zhang, and Jianfeng Lu. Feature fusion: parallel strategy vs. serial strategy. *Pattern Recognition*, 36(6):1369–1381, June 2003.

[124] Jian Yang, David Zhang, Alejandro F. Frangi, and Jing yu Yang. Two-dimensional pca: A new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:131–137, 2004.

[125] Jiepingm Ye, Qim Li, Huim Xiong, Haesunm Park, Ravim Janardan, and Kumar Vipin. Idr/qr: An incremental dimension reduction algorithm via qr decomposition. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1208–1222, 2005.

[126] Li Yongmin. On incremental and robust subspace learning. *Pattern Recognition*, 37:1509–1518, 2004.

[127] Hua Yu and Jie Yang. A direct lda algorithm for high-dimensional data with application to face recognition. *Pattern Recognition*, 34:2067–2070, 2001.

[128] Taiping Zhang, Bin Fang, Yuan Yan Tang, Zhaowei Shang, and Bin Xu. Generalized discriminant analysis: a matrix exponential approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(1):186–197, 2010.

[129] Haitao Zhao and Pong Chi Yuen. Incremental linear discriminant analysis for face recognition. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 38(1):210–221, 2008.

[130] Haitao Zhao, Pong Chi Yuen, and James T. Kwok. A novel incremental principal component analysis and its application for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics (Part B*, 36:873–886, 2006.

[131] Wenyi Zhao, Arvindh Krishnaswamy, Rama Chellappa, Daniel L. Swets, and John Weng. Discriminant analysis of principal components for face recognition. In *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, pages 336–341. IEEE Computer Society, 1998.

[132] Jianke Zhu, Steven C. H. Hoi, and Michael R. Lyu. Face annotation using transductive kernel fisher discriminant. *Multimedia, IEEE Transactions on*, 10(1):86 –96, jan. 2008.

[133] Lei Zhu, Yongying Jiang, and Lihua Li. Making discriminative common vectors applicable to face recognition with one training image per person. In *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, pages 385–387, sept. 2008.