Transformaciones basadas en el algoritmo Local Binary Pattern de imágenes capturadas con la *Kinect* para clasificación facial.

Enrique José Ramón Balmaseda

Tutores: Dr. Javier Lorenzo Navarro y Dr. Modesto F. Castrillón Santana

29 de noviembre de 2011

Índice general

Índice de figuras

Capítulo 1

Introducción

La irrupción de *Kinect* [Kin] en el mercado doméstico ha generado un rápido movimiento en la comunidad científica relacionada con los sistemas de visión artificial al poder disponer de un dispositivo 'económico' con capacidad de obtener información tridimensional de las escena, además de la imagen. ¿Es *Kinect* la respuesta?. *Kinect* ha supuesto toda una revolución, ya no sólo porque ha cambiado la forma de jugar a videojuegos sino porque, por 150 dólares, los investigadores disponen de un sistema de cámaras y sensores, un SDK para desarrollar, soporte para OpenCV 2.3.1 e integración con en el nuevo kernel de Linux para este dispositivo. Gracias a ello se han desarrollado proyectos tan sorprendentes como el control de un helicóptero, un sistema de telepresencia y videoconferencia, una extensión que permite a cualquier web interactuar a través de *Kinect* e incluso los inicios de un sistema holográfico, sistema róbotico lazarillo para invidentes en resumen están proliferando multitud de proyectos vinculados a este dispositivo diferentes para el que fue creado, el entretenimiento. Con este trabajo se pretende realizar una implementación de un sistema de reconocimiento facial al que se le entrenará con imágenes obtenidas a través de *Kinect* para determinar si mejoran el rendimiento respecto a los procedimientos de actuales de clasificación.

Para ello, a partir de imágenes capturadas por *Kinect*, se diseñará un sistema de reconocimiento facial, el proceso consistirá, básicamente en:

1. Grabación de vídeos que incluyan imágenes de rostros de personas con *Kinect*. Como salida tendremos vídeos en formato RGB estándar y vídeos con imágenes de profundidad, escala de grises.

2. A los vídeos se les aplicará ENCARA2 [CS11] para la detección de los rostros en las imágenes que han sido obtenidas a través de *Kinect*.

3. A las imágenes de los rostros se les aplicará el operador Patrón Binario Local (de sus siglas en inglés) (LBP) [Pie10] para generar los vectores de características de las imágenes de las caras.

4. Por último, se realizará un estudio sobre el rendimientos de varios clasificadores empleando WEKA[1][**?**] , que es una colección de algoritmos *machine learning* ideal para el desarrollo de nuevos sistemas de aprendizaje automático.

[1]Plataforma escrita en Java y desarrollada por la Universidad de Waikato y distribuido bajo licencia de software libre GNU-GPL

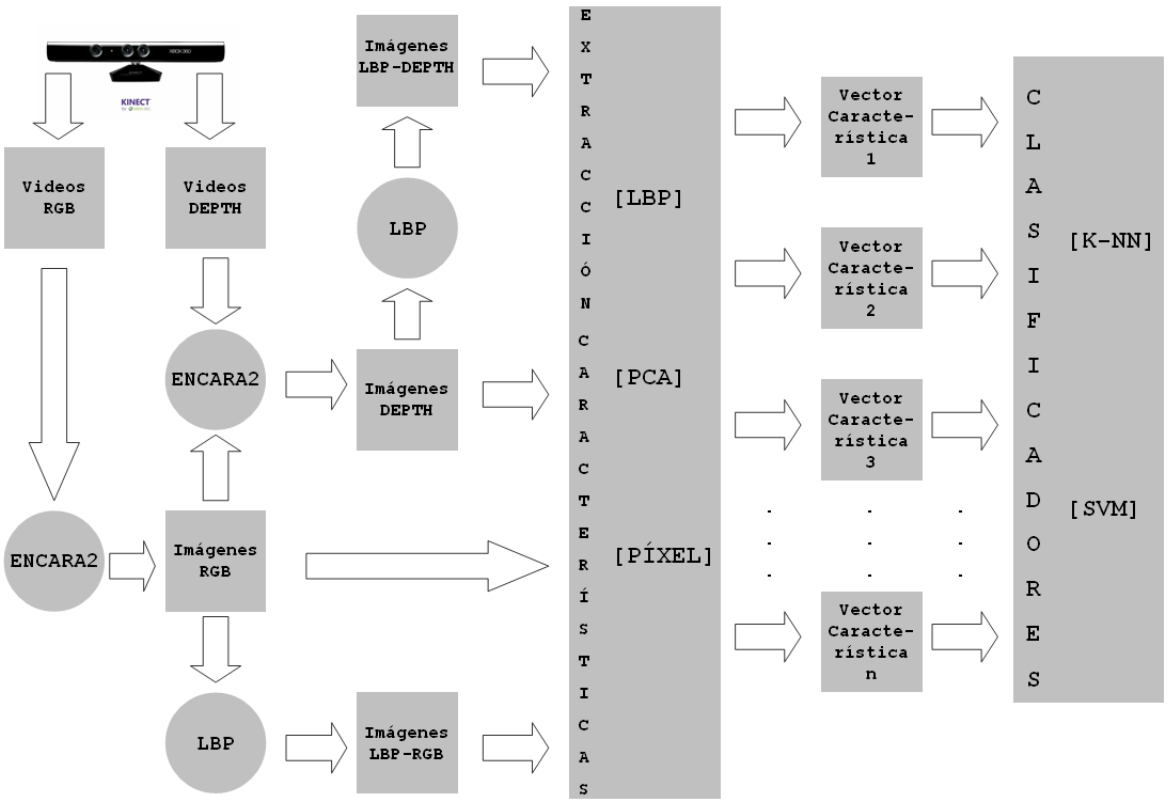


Figura 1.1: Esquema simplificado del proceso seguido en la implementación del sistema de reconocimiento facial

1.1. Reconocimiento facial

Recientemente se ha cumplido el décimo aniversario del 11 de septiembre de 2001, fecha en la que los Estados Unidos sufrieron el mayor ataque terrorista de la historia de forma simultánea a varias ciudades, entre ellas, a Nueva York, con el balance de cerca de 2.000 fallecidos, uno de los emblemas arquitectónicos de la ciudad: las Torres Gemelas, desaparecidas, y secuelas psicológicas directa e indirectamente a las sociedades americana y europea.

La amenaza terroristas, entre otros, ha servido como justificación a los gobiernos para la instalación de miles de cámaras digitales de vídeo de vigilancia, además de la necesidad de desarrollar nuevas aplicaciones basadas en rasgos biométricos para la identificación y reconocimiento facial, incrementándose la demanda de sistemas de este tipo en aras a **aumentar el nivel de seguridad**.

La **proliferación** de nuevas **cámaras digitales** de usos doméstico, de fotografía y vídeo, potentes y accesibles y la universalización del teléfono móvil con altas prestaciones han generado ingentes cantidades de imágenes de todo tipo.

Y por último: **Internet**, paradigma de la nueva sociedad, con las Redes Sociales como lugar de intercambio, de chismorreo, de comunicación, de información, de encentro el 'meeting point' global y, si sumamos, nuevas viejas aplicaciones como YouTube, Flickr,...

Todo este caldo de cultivo ha motivado que muchos equipos de investigación centren sus esfuerzo en resolver los retos que llevan aparejado el reconocimiento facial.

El reconocimiento facial es una tarea que, sin esfuerzo, realizamos los humanos diariamente y, en si mismo constituye un reto que ha ocupado a diferentes grupos de investigación de todo el mundo en las dos últimas década y esto es debido, además de lo comentado hasta ahora, a:

- Aumento de la potencia de cálculo de los equipos de proceso de datos.
- La disminución de los costes de los equipos tanto de procesos de datos como de obtención de imágenes de alta calidad.
- Y a la aparición de sistemas embebidos hardware-software muy potentes.

Todo ello ha hecho que se hayan implementado aplicaciones que se basan en el procesamiento de imágenes y de vídeo digital que incluyen:

1. Autentificación biométrica.
2. Soluciones de vigilancia o de seguridad.
3. Aplicaciones que interactuan con los humanos.
4. Gestión Multimedia.

El reconocimiento facial es una de las tecnologías biométricas que tiene las siguientes ventajas sobre otras:

- Es natural
- No es intrusivo.
- Fácil de usar

El desarrollo de sistemas comerciales de reconocimiento facial es la demostración de los importantes avances logrados en este campo de investigación pero, a pesar de ellos, el reconocimiento facial sigue siendo un reto para los investigadores de visión por computador. Esto se debe a que los sistemas actuales funcionan bien en ambientes relativamente controlados, pero tienden a empeorar sus resultados cuando cambian las condiciones ambientales o se modifican las circunstancias gestuales de los individuos, cambios en la expresión facial el envejecimiento, entre otros.

Las actuales líneas de investigación centran sus esfuerzos en aumentar la robustez de los sistemas frente a factores como, iluminación, ángulo, pose, elementos externos. Idealmente, el objetivo es desarrollar un sistema de reconocimiento facial que imite la capacidad de percepción visual humana.

1.2. Tipos de reconocimiento facial y sus fases

Un sistema de reconocimiento facial [LJ05] pretende identificar los rostros presentes en imágenes y vídeos de forma automática y puede funcionar en uno de los siguientes modos:

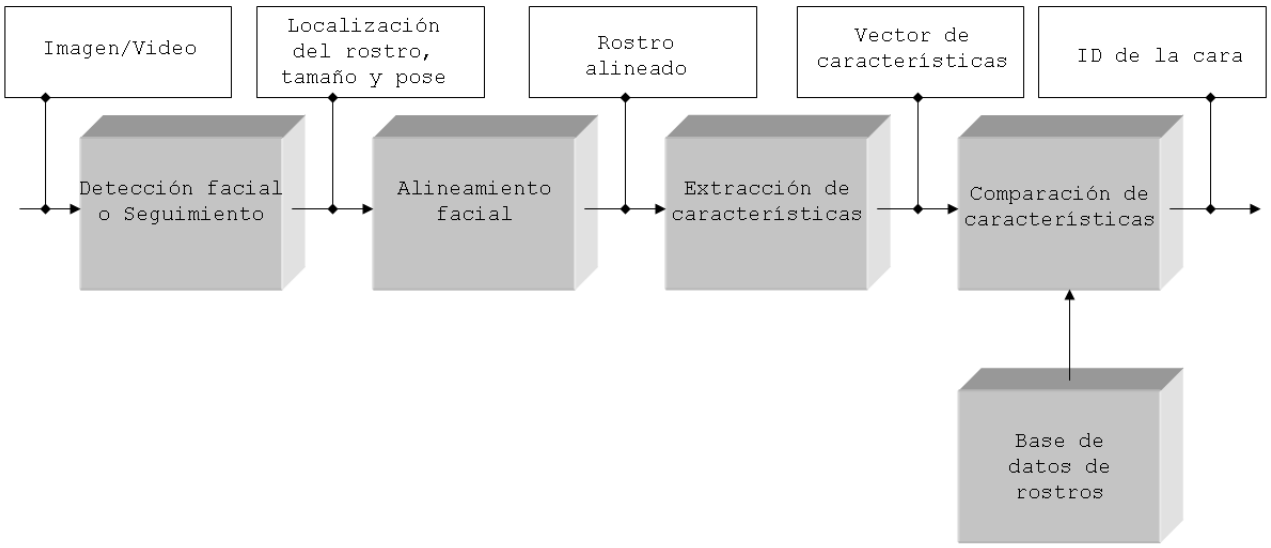


Figura 1.2: Flujo del proceso de reconocimiento facial

1. *Reconocimiento facial o autentificación*:
   El procedimiento consiste en una comparación **uno a uno**. Se dispone de la imagen de un sujeto en una base de datos y la comparamos con la nueva imagen captada por una cámara digital para determinar si existe coincidencia.

2. *Identificación facial o reconocimiento*:
   En este caso la comparación es **uno a muchos**. Se dispone, también, de una base de datos de imágenes, la nueva imagen captada es comparada con las imágenes de la base de datos para determinar la persona con la que hay mayor coincidencia. Existe otro escenario, en el que se utiliza una lista reducida de candidatos, esto es, una comparación **uno a pocos**.

Las principales dificultades en la tarea de reconocimiento facial son los grados de libertad que envuelve a la captura de las imágenes, entre otras:

- El punto de vista desde donde la imagen es tomada.
- La iluminación.
- La expresión de la cara en el momento de la toma de la imagen.
- La ocultación parcial del rostro.
- Posibles accesorios, gafas, pendientes.
- El paso del tiempo.
- Pose de la cabeza
- Cambio de escala

En la figura 1.2 se presenta el esquema estándar y simplificado de un sistema de reconocimiento facial. El reconocimiento facial es básicamente un problema de reconocimiento de patrones. Disponemos de un

rostro en tres dimensiones sujeto a variaciones de iluminación, pose, expresión y necesitamos realizar una identificación basada en una imagen de dos dimensiones. Las cuatro fases del sistema de reconocimiento son:

1. *Detección de la cara*. Consiste en detectar y separar el área de la cara del resto de la imagen. En el caso de tratamiento de un vídeo la detección de la cara además exige el seguimiento de la misma. La salida de este modulo proporciona una estimación poco refinada del rostro de la persona.

2. *Alineamiento de la cara*. Está dirigido a una localización más precisa de la cara, así como a la normalización. Los componentes de la cara como ojos, nariz, boca y contorno son utilizados para normalizar la cara con respecto a sus propiedades geométricas como tamaño, posición, usando transformaciones geométricas o técnicas de 'morphing'. Además también se normaliza con respecto a características fotométricas, es decir, iluminación, tamaño y escala de grises.

3. *Extracción de característsicas*. Es el tercer paso del proceso, se lleva a cabo para proporcionar información efectiva que permita distinguir las caras de diferentes personas y sea estable con respecto a la geometría y variaciones fotométricas. La salida de esta fase es el vector de características de la imagen.

4. *El último paso consiste en la comparación*. El vector de características obtenido en la fase anterior es comparado con todos los vectores de características almacenados en la base de datos de imágenes y, la salida de esta fase es la identidad de la cara, cuando se encuentra una correspondencia con suficiente confianza, o identificación fallida.

Luego el resultado del reconocimiento depende en gran medida de:

- Las extracción de las características de la cara.
- De los métodos de clasificación utilizados para distinguir entre los rostros almacenados en la base de datos de sujetos.

1.3. La *Kinect*

La *Kinect* para Xbox 360, o simplemente *Kinect* (originalmente conocido por el nombre en clave «Project Natal»), es « un controlador de juego libre y entretenimiento » creado por Alex Kipman, desarrollado por Microsoft para la videoconsola Xbox 360, y en un futuro para PC a través de Windows 8. *Kinect* permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional (joystick), mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, objetos e imágenes.

Microsoft Research invirtió veinte años en el desarrollo de la tecnología de *Kinect* de acuerdo con los comentarios de Robert J.Bach. *Kinect* fue anunciado por primera vez el 1 de junio de 2009 en la Electronic Entertainment Expo 2009 como "Project Natal". Este nombre en clave « Proyecto Natal » responde a la tradición de Microsoft de utilizar ciudades como nombres en clave. Alex Kipman, director de Microsoft, quien gestó el proyecto, decidió ponerle el nombre de la ciudad brasileña Natal como un homenaje a su país de origen y porque la palabra natal significa « de o en relación al nacimiento », lo que refleja la opinión de Microsoft en el proyecto como «el nacimiento de la próxima generación de dispositivos de entretenimiento en el hogar». Poco antes de la E3 2010 varios weblogs tropezaron con un anuncio que supuestamente se filtró en el sitio italiano de Microsoft que sugirió el título *Kinect* que confirmó más tarde junto con los detalles de una nueva Xbox 360 más delgada.

El sensor *Kinect* es una barra horizontal de aproximadamente 23 cm (9 pulgadas) conectada a una pequeña base circular con un eje de articulación de rótula, y está diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla de vídeo, como se puede observar en la figura 1.3.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un procesador personalizado que ejecuta el software que proporciona captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz. El micrófono de matrices del sensor de *Kinect* permite a la Xbox 360 llevar a cabo la localización de la fuente acústica y la supresión del ruido ambiente, permitiendo participar en el chat de Xbox Live sin utilizar auriculares. El sensor contiene un mecanismo de inclinación motorizado.

El sensor de profundidad es un proyector de infrarrojos combinado con un sensor CMOS monocromo que permite ver la habitación en 3D en cualquier condición de luz ambiental. El rango de detección de la profundidad es ajustable gracias al software de *Kinect* capaz de calibrar automáticamente el sensor, basado en la jugabilidad y en el ambiente físico del jugador, tal como la presencia de sofás.

Kinect utiliza técnicas de reconocimiento de voz y reconocimiento facial para la identificación automática de los usuarios. La aplicación puede utilizar la funcionalidad de seguimiento *Kinect* y el sensor de giro motorizado para ajustar la cámara para que el usuario se mantenga siempre en el marco.

Figura 1.3: Imagen de la *Kinect* de Xbox de Microsoft

En noviembre de 2010, Industrias Adafruit ofreció una recompensa por un controlador de código abierto para *Kinect*. El 10 de noviembre, se anunció al español Héctor Martín como el ganador, que utilizó métodos de ingeniería inversa con *Kinect* y desarrolló un controlador GNU/Linux que permite el uso de la cámara sistema de se[Pleaseinsertintopreamble]al de vídeo que utiliza la señal de rojo, verde y azul por separado (RGB) y las funciones de profundidad.

1.4. Principales tareas a desarrollar en el trabajo

Precisamente en este trabajo nos centramos básicamente en dos aspecto del proceso de reconocimiento facial. La caracterización de las imágenes mediante operador LBP en lugar de los que se usan habitualmente como Principal Component Analysis en castellano Análisis de Componentes Principales (PCA) o Análisis Discriminante Lineal (LDA) o simplemente el mapa de píxeles de las imágenes. Y utilizar como método de clasificación, y por ende, de detección, sistemas basados en aprendizaje automático.

Las técnicas de aprendizaje automático o *Machine Learning* se basan en la aplicación de algoritmos que mejora su rendimiento a través de la experiencia, como indica Tom M. Mitchell en [Mit97] , el cual

define el aprendizaje automático de la siguiente forma:

Un programa de computadora se dice que aprender de la experiencia E con respecto a la misma clase de tareas T y sea P la medida del rendimiento, si su desempeño en la tarea T, medida por P, mejora con la experiencia E.

En el diseño de cualquier sistema de aprendizaje automático nos enfrentamos a una serie de elecciones:

1. Elegir el **conjunto de aprendizaje** que se va a utilizar para que el sistema aprenda y, el éxito del sistema puede depender de este conjunto. Los puntos claves son:
   a) Si el conjunto de entrenamiento proporciona una realimentación directa o indirecta.
   b) El grado de control que se tiene sobre el conjunto de entrenamiento.
   c) Como de bien está representado la distribución de ejemplos sobre el sistema de rendimiento P que debe ser medido.
2. Elección de la f**unción objetivo**, es decir, determinar exactamente el tipo de conocimiento que se desea aprender y como este será utilizado por el programa
3. Elección de la **Representación de la función objetivo**, hemos de seleccionar una representación de la función para que el programa pueda utilizar la función para aprender. Para representar el conocimiento existen varias alternativas, por ejemplo: usar tablas, reglas, funciones polinómicas o mediante redes neuronales artificiales. En general existe un compromiso entre una elección de una representación muy expresiva, que nos permita una representación lo más exacta posible de la función objetivo, o por otro lado una representación que permita, dado el conjunto de entrenamiento, elegir sobre un conjunto alternativo de hipótesis que pueda ser representada.
4. Elección de una Aproximación Algorítmica de la Función Objetivo. **Determinar el algoritmo de aprendizaje** a utilizar.

Una vez realizado esto nos queda, como es lógico, verificar que el sistema es capaz que dado una muestra nueva mejora su rendimiento dando una respuesta adecuada a la tarea que se le ha encomendado.

El planteamiento que voy a utilizar en este trabajo es utilizar algoritmos de clasificación para la fase de comparación de características comparándola con la base de datos de rostros de la figura 1.2, con el objetivo de tener una correcta clasificación del rostro de la persona. En concreto utilizaremos los algoritmos Suport Vector Machine en castellano Máquina Vector Soporte (SVM) y K vecinos más próximos (de sus siglas en inglés (K-NN) que nos permitirán realizar una comparación de los resultados y medir el rendimiento total del sistema.

1.5. Diseño del sistema de reconocimiento de imágenes

Los dos hitos diferenciadores del sistema de reconocimiento que se han implantado son:

1. La elección del dispositivo para generar los vídeos de donde obtendremos las imágenes: *Kinect*
2. La elección del algoritmo de extracción de características: *Patrón Binario Local (de sus siglas en inglés) (LBP)*

El objetivo es el diseño de un sistema de reconocimiento facial a partir de imágenes capturadas a través de la *Kinect*, lo que nos permite procesar imágenes en formato RGB y en formato de profundidad, en escala de grises, que son las salidas que nos proporcionan este dispositivo. La imagen de profundidad, en escala de grises, nos permitirá intentar determinar si la misma es válida para la implementación de un sistema de reconocimiento facial.

Como se ha indicado el operador seleccionado para extraer el vector de características de las imágenes es el LBP, con esto pretendemos mitigar las dificultades de los sistema de reconocimiento facial vinculados: a la iluminación, pose, expresión, ya que este operador se define como una medida de la textura en escala de grises invariante, derivado de una definición general de la textura a través de un entorno local.

Con sus recientes extensiones, el operador LBP se ha convertido en una medida potente de la textura de la imagen, que muestra excelentes resultados en muchos estudios empíricos. El operador LBP puede ser visto como un enfoque unificador de los modelos tradicionalmente divergentes estadísticos y estructurales de análisis de texturas. **Quizá la propiedad más importante del operador LBP en aplicaciones del mundo real es su invariancia frente a cambios de nivel de gris monótono**. Otra no menos importante es su simplicidad computacional, que permite analizar las imágenes en tiempo real.

La representación de imágenes faciales utilizando el operador LBP [AHP06] se propuso en el 2004 y ha evolucionado siendo utilizado por varios grupos de investigación.

Hasta ahora, los resultados que han obtenido indican que la textura y las ideas que hay detrás de la metodología LBP podría tener un papel más amplio en la visión artificial y en el análisis de imágenes del que se pensaba y, es por ello, por lo que se ha seleccionado como método a utilizar en la implementación del sistema de reconocimiento o autentificación facial en este trabajo.

1.6. Objetivos

Cómo resumen de todo lo indicado y para centrar los trabajos a llevar acabo se pretenden obtener los siguientes objetivos en el desarrollo de este trabajo:

- Utilizar como software de detección facial ENCARA2 para los vídeos en formato RGB.
- Detectar las caras en los vídeos con imágenes con formato de profundidad generado con la *Kinect*.
- Utilizar como caracterización de las imágenes de los rostros diferentes versiones de los operadores LBP, explotando muchas de alternativas que proporcionan.
- Determinar si con las imágenes de profundidad generadas por *Kinect* obtenemos mejor rendimientos en el procesos de clasificación facial utilizando los algoritmo K-NN y SVM.

Capítulo 2

Descriptores, algoritmos y herramientas utilizadas

En este capítulo del trabajo vamos a desarrollar los aspectos teóricos de los métodos que hemos empleados durante la fase de experimentación con los datos para la obtención de resultados que expondremos en el apartado correspondiente. En concreto analizaremos los métodos de extracción de características, los algoritmos de clasificacióon usados y por último las otras herramientas software que hemos utilizado en concreto:

- Métodos de extracción de características.
  - LBP
  - Mapa de píxeles de una imagen.
  - PCA.
- Algoritmos de clasificación.
  - SVM.
  - K-NN.
- Herramienta software adicionales utilizadas
  - WEKA.
  - ENCARA2.

2.1. Local Binary Pattern

LBP [Pie10] [web] es un operador de textura simple pero muy eficiente que etiqueta los píxeles de una imagen por vecindad de umbral de cada píxel con el valor del píxel central, y considera el resultado como un número binario. Debido a su poder de discriminación y la simplicidad de cálculo, este operador de textura se ha convertido en un método popular que se usa en varias tipos de aplicaciones. Puede ser visto como un enfoque unificador de los modelos tradicionalmente divergentes del análisis de texturas: los estadísticos versus los estructurales. Quizá la propiedad más importante del operador, para aplicaciones del mundo real, es su robustez frente a cambios, en una escala de grises monótona, causados, por ejemplo, por las variaciones de iluminación y frente a rotaciones, en el caso de utilizar códigos circulares. Otra característica importante es su simplicidad computacional, lo que nos que permite analizar las

imágenes en tiempo real.

LBP proporciona un operador de análisis de textura que se define como una medida de la textura en una escala de grises invariante, derivado de una definición general de textura mediante vecinos locales. La forma actual del operador LBP es muy diferente de su versión básica: la definición original se extiende a un conjunto de vecinos arbitrarios circulares, y se han desarrollado nuevas versiones del mismo, sin embargo, la idea principal es la misma: un código binario que describe el patrón de la textura local que es construido por el umbral de un conjunto de vecinos por el valor de gris de su centro. El operador tiene que ver con muchos otros métodos conocidos de análisis de texturas en la figura 2.1 [Uni] se observan las relaciones del operador LBP con los otros métodos.

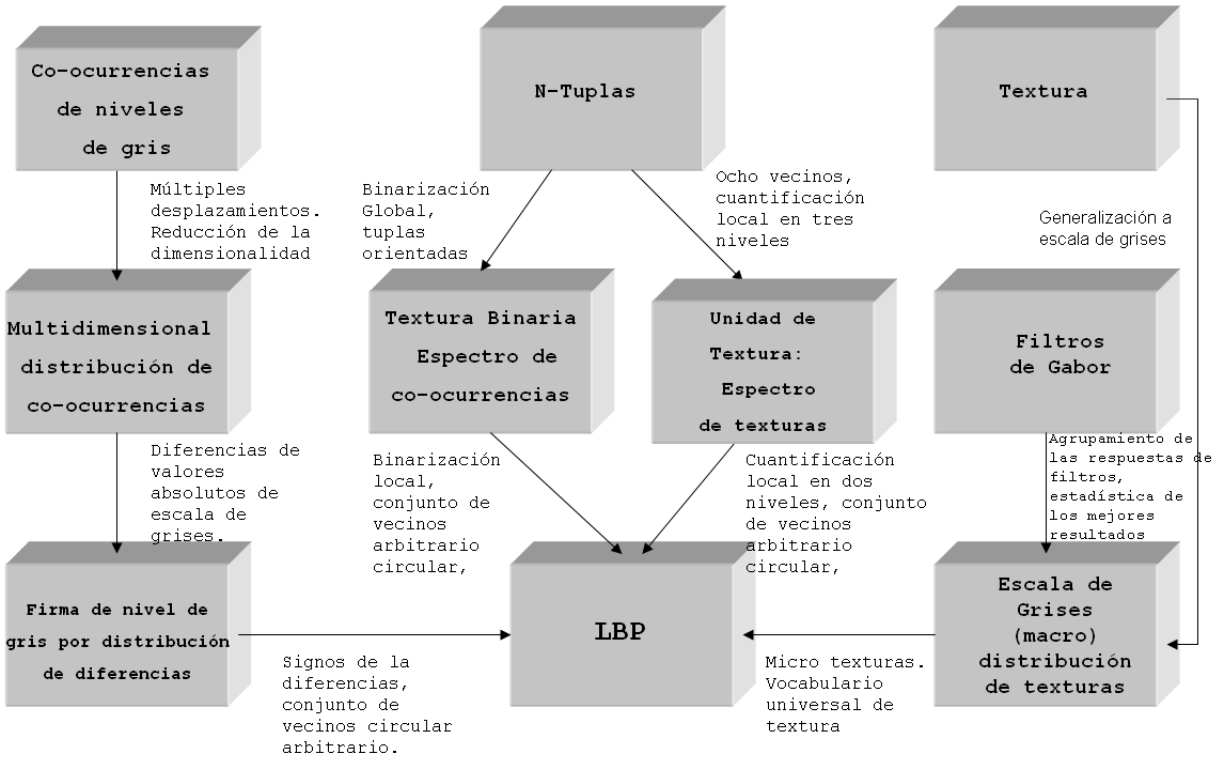


Figura 2.1: LBP operador en el campo de análisis de texturas

La primera versión del operador LBP

El operador original LBP, fue introducido por [OPH96] y es un método para describir texturas. El operador etiqueta cada uno de los píxeles de una imagen mediante el muestreo de matrices 3x3 comparando cada píxel con el valor central de la matriz y considerando el resultado como un número binario. A continuación, se calcula el histograma de las etiquetas y se concatenan los histogramas esto se puede utilizar como un descriptor de textura. Un ejemplo se puede observar en la figura 2.2 como ilustración de como funciona el operador LBP.

Dado que LBP, por definición, es invariante a los cambios monótonos en escala de grises, se complementó con una medida ortogonal de contraste local. La figura 2.3 muestra cómo se calcula la medida de contraste C: el promedio de los niveles de gris por debajo del píxel central se resta el promedio de los

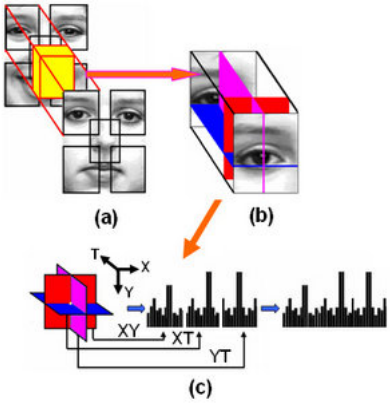


Figura 2.2: Descripción del patrón facial con LBP

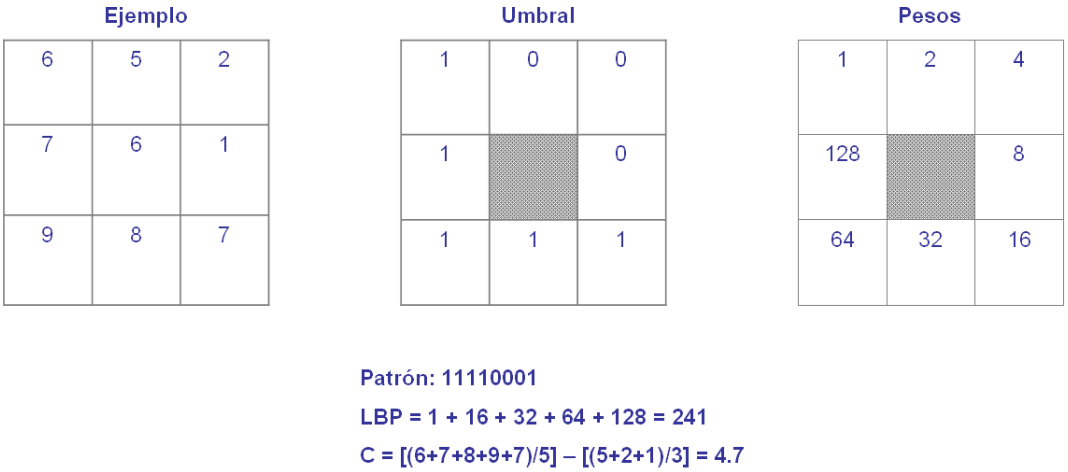


Figura 2.3: Procedimiento de cálculo de LBP original y la medida de contraste

niveles de gris por encima (o igual) que el píxel central. Las distribuciones bidimensionales del LBP y las medidas locales de contraste se utilizan como características de la imagen y es por ello que al operador también se le llama LBP/C.

Evolución del operador LBP

La evolución del operador LBP fue desarrollada por [OPM02] y posteriormente, el operador se extendió el uso de diferentes tamaños, no solo a ocho puntos, sino a muestreos circulares y, la bilinealidad se consigue con la interpolación de los valores de los píxeles, lo que permite utilizar cualquier radio y por lo tanto cualquier número de píxeles vecinos. Para los entornos circulares se usa la notación (P, R), donde P indica el número de puntos de muestreo y R el radio del círculo, en la figura 2.4 se pueden observar ejemplos para diferentes tamaños , en concreto para $(8, 2); (12, 2{,}5)$ y $(16, 4)$ como valores de (P, R).

2.1.1. Derivación del operador LBP

Definimos T, ***Textura***, como la distribución conjunta de los niveles de gris de $P+1(P>0)$ píxeles de una imagen.

$T = t(g_c, g_0, g_1, \cdots, g_{p-1})$

Donde g_c se corresponde con el nivel de gris del píxel central del muestreo de una matriz nxn y $g_p(p = 1, 2, \cdots, p-1)$, corresponden a los valores de gris de los píxeles P equidistantes en un círculo de radio $R(R > 0)$ que forman un conjunto circular y simétrico de los vecinos con respecto al centro. La Figura 2.4 muestra tres conjuntos vecinos circularmente simétricos para diferentes valores de P y R, como ya hemos indicado. Sin perder información podemos restar g_c de los puntos g_p, quedando:

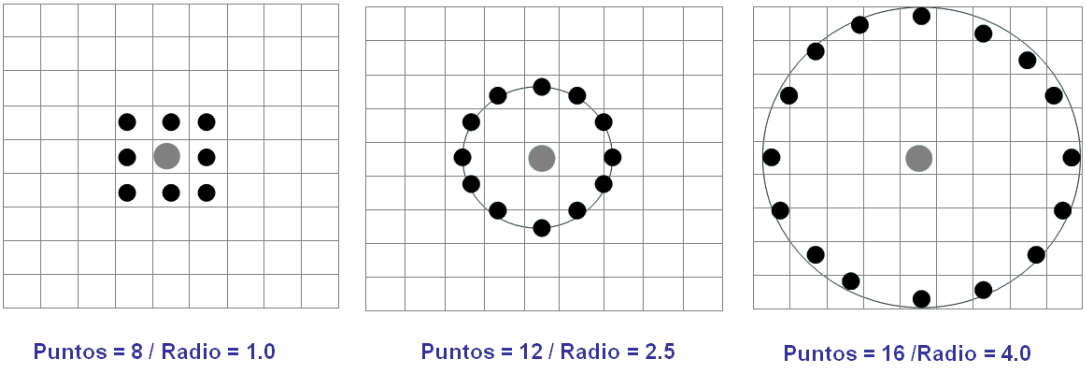


Figura 2.4: Simetría circular para conjuntos de vecinos

$T = t(g_c, g_0 - g_c, g_1 - g_c, \cdots, g_{p-1} - g_c)$, asumiendo que la diferencia son independientes la distribución puede ser factorizada:

$T \approx t(g_c)t(g_0 - g_c, g_1 - g_c, \cdots, g_{p-1} - g_c)$, como $t(g_c)$, describe la luminosidad general de una imagen, que no está relacionada a la textura de la imagen local, puede ser ignorada y quedando: $T \approx t(g_0 - g_c, g_1 - g_c, \cdots, g_{p-1} - g_c)$.

Aunque invariante frente a cambios de escala de grises, las diferencias se ven afectados por el cambio de escala. Para lograr invariancia con respecto a cualquier transformación monótona de la escala de grises, sólo es necesario considerar los signos de las diferencias:

$T \approx t(s(g_0 - g_c), s(g_1 - g_c), \cdots, s(g_{p-1} - g_c))$, donde:

$$S(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Ahora, un peso 2^p es asignado a cada signo $s(g_p - g_c)$ transformando la diferencia en el muestreo en un único código LBP:

$$LBP_{P,R} = \sum_{p=0}^{p-1} s(g_p - g_c)2^p$$

2.1.2. Extensiones del operador LBP

De este operador se han desarrollado extensiones reflejando el interés que en él han mostrado los diferentes grupos de investigación, en esta sección vamos a ver dos de ellas.

Patrones Uniformes

Otra versión para el operador original, es la llamada *patrones uniformes*, que puede ser usado para reducir la longitud del vector de características y para implementar un descriptor sencillo e invariante frente a rotaciones. Esta versión fue inspirada por que algunos patrones binarios son más frecuentes en las imágenes de textura que otros. Un código LBP se llama homogéneo si el patrón binario contiene un máximo de dos transiciones a nivel de bits, de 0 a 1 o viceversa, cuando el patrón de bits es atravesado de manera circular. Por ejemplo, los patrones de 00000000 (0 transiciones), 01110000 (2 transiciones) y 11001111 (2 transiciones) son uniformes, mientras que los patrones de 11001001 (4 transiciones) y 01010010 (6 transiciones) no lo son. En el cálculo de las etiquetas LBP, cuando los patrones uniformes son utilizados, se utiliza una etiqueta para cada uno de los patrones uniforme y todos los patrones no uniformes están etiquetados con una sola etiqueta. Por ejemplo, cuando se utiliza (8, R) de vecindad, hay un total de 256 patrones, 58 de los cuales son uniformes y el resto son no uniformes, por lo que resulta uno total de 59 etiquetas diferentes.

Espacio temporal- LBP

El operador original LBP se definió para tratar sólo la información espacial, más tarde, se extendió a una representación espacio-temporal para el análisis de texturas dinámicas. El nombre que recibe esta versión del operador es Volumen Local Binary Pattern (VLBP), esta extensión fue propuestas por [ZP07]. La idea que hay detrás de VLBP consiste en observar la textura dinámica como si fuera un conjunto de volúmenes en el espacio (X, Y, T) donde X e Y indican las coordenadas espaciales y T es el índice de marco (tiempo). Los vecinos de cada píxel se define así en un espacio tridimensional. Después, como para LBP en el dominio espacial, volumen de texturas se puede definir de forma similar y se extrae en histogramas. Por lo tanto, VLBP combina el movimiento y la apariencia en conjunto para describir texturas dinámicas.

Para hacer que el VLBP sea computacionalmente simple y fácil de extender su uso, el operador se basa en la co-ocurrencias de patrones binarios locales en tres planos ortogonales : XY, XT y YT. Los patrones LBP de cada plano se concatena LBP como se muestra en la figura 2.2. Los vecinos se han generalizado, para un muestreo circular, con la intención de adaptarse a las estadísticas del espacio-tiempo.

La Figura 2.5 muestra un ejemplo de imágenes en los tres planos. El plano XY representa la información de la apariencia, mientras que el plano XT ofrece una impresión visual de una fila cmbiante en el tiempo y YT describe el movimiento de una columna en el espacio temporal. Los códigos de LBP son extraídos de todos los píxeles de la XY, XT e YT, denominado XY-LBP, XT-LBP e YT-LBP, y los histogramas de estos planos se calculan y se concatenan en un solo histograma. En esta representación, una textura dinámica está codificada por una apariencia (XY-LBP) y dos espacio-temporal (XT-LBP) e (YT-LBP).

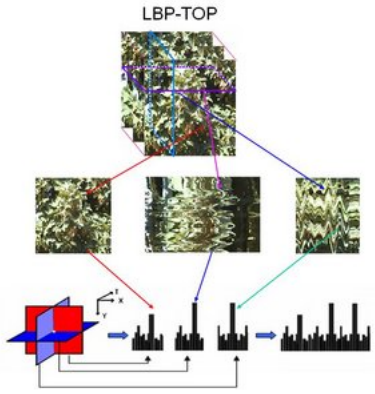


Figura 2.5: LBP de tres planos ortogonales

Que el ajuste del radio sea igual en el eje de tiempo al del eje de espacio no es razonable para las texturas dinámicas. Por lo tanto, tienen diferentes parámetros de radio en el espacio y en el tiempo. En el planos XT e YT, se puede asignar diferentes radios a la muestra y puntos vecinos diferentes en el espacio y en el tiempo. De manera general todos los parámetros pueden ser distintos: radios y número de vecinos en cada uno de los planos.

2.1.3. Invariante a rotaciones

El operador LBP_{PR} produce 2^P valores de salida diferentes, correspondientes a los 2^P patrones binarios diferentes que se pueden formar por los P píxeles vecinos. Cuando se rota la imagen, los valores g_p de gris correspondientes se moverá a lo largo del perímetro del círculo alrededor de g_0. Puesto que g_0 está destinado a ser el valor de gris de elemento (0, R) a la derecha de g_c la rotación de un patrón binario en particular, naturalmente, da lugar a un $LBP_{P,R}$ diferente. Esto no se aplica a los patrones que consta de sólo 0s (o 1s) que permanecen constantes en todos los ángulos de rotación. Para eliminar el efecto de la rotación, es decir, para asignar un identificador único para cada rotación invariable patrón binario local se define:

$$LBP^{ri}_{P,R} = min\{ROR(LBP_{P,R}, i)|i = 0, 1, \cdots, P-1\} \tag{2.1}$$

donde $ROR(x, i)$ realiza un desplazamiento circular, a nivel de bit, hacia la derecha P-bits de x i-veces. En términos de imagen de bits la ecuación 2.1, corresponde simplemente a girar el conjunto de vecinos hacia la derecha tantas veces como el número máximo de los bits más significativos, a partir de g_{P-1}, esta 0.

$LBP^{ri}_{P,R}$ cuantifica las ocurrencias de rotaciones invariables los patrones correspondientes a ciertas micro características de la imagen, por lo que los patrones pueden ser considerados como detectores de rasgos. Figura 2.6 ilustra las 36 únicas rotaciones invariantes de patrones binarios locales que pueden ocurrir en el caso de $P = 8$, es decir, $LBP^{ri}_{8,R}$ puede tener 36 valores distintos. Si $R = 1$, esto es, $LBP^{ri}_{8,1}$, se corresponde con el operador invariante de escala de grises y rotación que ha sido designado como LBPROT en [MTZ00].

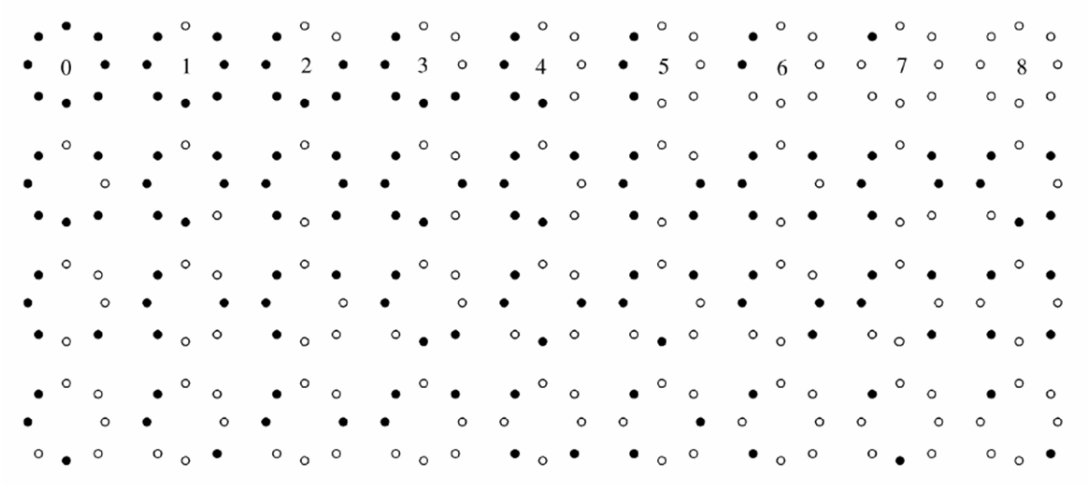


Figura 2.6: Rotaciones invariantes de patrones binarios que pueden ocurrir en el conjunto de ochos vecinos de la simetría circular $LBP^{ri}_{8,R}$. Los círculos en blanco y negro se corresponden a los valores de bit de 0 y 1 en la salida de 8 bits del operador. La primera fila contiene los nueve patrones de uniformes y los números dentro de ellos corresponden a su único código $LBP^{riu2}_{8,R}$.

Invariancia a rotación mejorada con los patrones uniformes y cuantificación más fina del espacio angular

La experiencia práctica, sin embargo, ha demostrado que LBPROT como tal, no proporciona una muy buena discriminación, conclusión a la que llegó también [MTZ00]. Hay dos razones: Las frecuencias de ocurrencia de los 36 patrones individuales incorporados en LBPROT varían mucho y la cuantificación en bruto del espacio angular en intervalos de 45 grados.

Se ha observado que existen ciertos patrones binarios que están presente en más del 90 % de las texturas generados por muestreos locales de 3x3 píxeles. A estos patrones se les conoce con el nombre de patrones uniformes, que tienen una cosa en común, su estructura circular uniforme tiene muy pocas transiciones espaciales. Los patrones uniformes se muestran en la primera fila de la figura 2.6. Ellos funcionan como una micro plantilla, con un punto brillante (0), superficie plana o punto negro (8), y los bordes de diferente curvatura positiva y negativa (1-7).

Para definir formalmente los patrones uniforme, se introduce una medida de la uniformidad U (patrón), que se corresponde con el número de transiciones espaciales (a nivel de bits 0 / 1) cambios el patrón. Por ejemplo, los patrones 00000000_2 y 11111111_2tienen un valor U de 0, mientras que los otros siete patrones en la primera fila de la figura 2.6 tienen un valor U de 2, ya que hay exactamente dos 0 / 1 en el patrón de las transiciones. Del mismo modo, los otros 27 modelos tienen un valor U de al menos 4. Designamos a los patrones que tienen un valor de U al menos de 2 como patrones uniformes y tenemos el siguiente operador invariante a escala de grises y rotaciones en lugar del usado hasta ahora $LBP^{ri}_{P,R}$:

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{P=0}^{P-1} s(g_P - g_c) \text{ si } U(LBP_{P,R}) \leq 2) \\ P+1 \text{ en cualquier otro caso.} \end{cases} \tag{2.2}$$

donde tenemos que

$$U(LBP_{P,R}) = |s(g_{P-1} - g_c) - (g_0 - g_c)| + \sum_{P=1}^{P-1} |s(g_P - g_c) - s(g_{P-1} - g_c)| \tag{2.3}$$

El superíndice ^{riu2} refleja el uso de los patrones invariable a rotaciones y uniformes que tienen un valor de U menor o igual a 2. Por definición, $P+1$ patrones binarios uniformes pueden existir en un conjunto de P píxeles vecinos con simetría circular. La ecuación (2.2 asigna una etiqueta única para cada una de ellas correspondiente al número de bits '1' en el patrón ($0 \rightrightarrows P$), mientras que los patrones no uniformes agrupan bajo la etiqueta genérica ($P+1$). En la figura 2.6, las etiquetas de los patrones uniformes se indican en el interior de los patrones. En la práctica, el mapeo de $LBP_{P,R}$ a $LPB_{P,R}^{riu2}$, que tiene $P+2$ valores de salida diferentes, será más eficaz que la búsqueda en una tabla de 2^P elementos.

La función de la textura final empleada en el análisis de la textura es el histograma de los resultados del operador (es decir, las etiquetas del patrón) acumulado a lo largo de una muestra textura. La razón por la que el histograma de los patrones uniformes proporciona una mejor discriminación en comparación con el histograma de todos los patrones individuales se reduce a las diferencias en sus propiedades estadísticas. La proporción relativa de los patrones no uniformes con respecto a todos los patrones, acumulados en un histograma, es tan pequeña que sus probabilidades no se pueden estimar de forma fiable.

La inclusión de estimaciones de ruido y de análisis de disimilitud de las muestras y el modelo de histogramas podrían deteriorar el rendimiento de este operador.

Hemos señalado anteriormente que la invariancia a rotaciones de $LBPROT(LBP_{8,1}^{ri})$ se ve obstaculizada por la cuantificación de rotaciones de 45 grados del espacio angular proporcionada por los vecinos del conjunto de ocho píxeles. Una solución es usar una P mayor y la cuantificación del espacio angular se define por $(360^o/P)$. Sin embargo, algunas consideraciones que deben tenerse en cuenta en la selección de P. En primer lugar, P y R se relacionan en el sentido de que el entorno circular que se corresponde a una R dada contiene un número limitado de píxeles (por ejemplo, nueve de R = 1) , que introduce un límite superior al número de puntos de muestreo no redundante en el entorno. En segundo lugar, una implementación eficiente, con una tabla de búsqueda de elementos 2^P establece un límite superior práctico para P .

Medida de la variaza de la rotación invariante del contraste de la textura local de una imagen

El operador$LPB_{P,R}^{riu2}$ es una medida invariante en una escala de grises, es decir, su salida no se ve afectada por cualquier transformación monótona de la escala de grises. Es una excelente medida de la distribución espacial por definición elimina el contraste. Si la invariancia de escala de grises no es necesaria y queremos incorporar el contraste de la textura de la imagen, esta se puede medir con una medida invariante de rotación de la varianza local:

$$VAR_{P,R} = \frac{1}{P}\sum_{P=0}^{P-1}(g_p - \mu)^2, \text{ donde } \mu = \frac{1}{P}\sum_{P=0}^{P-1} g_p \tag{2.4}$$

$VAR_{P,R}$ es, por definición, invariante a los cambios en la escala de grises. $LPB^{riu2}_{P,R}$ y $VAR_{P,R}$ son complementarios, se espera que su distribución conjunta $LPB^{riu2}_{P,R}/VAR_{P,R}$ sea una medida potente , invariante a rotaciones, de la textura de la imagen local.

2.1.4. Reconocimiento facial mediante el operador LBP

En el enfoque de LBP para la clasificación de la textura, las ocurrencias de los códigos LBP de una imagen se recogen en un histograma. La clasificación se realiza entonces mediante el cálculo sencillo de similitud de histogramas. Sin embargo, teniendo en cuenta la aproximación de similitud para el reconocimiento facial resulta una representación con pérdida de información espacial y, por lo tanto, se debe codificar la información de textura, manteniendo también su ubicación. Una forma de lograr este objetivo es con el uso de los descriptores de textura LBP para construir varias descripciones locales de la cara y combinarlos en una descripción global. Tales descripciones locales han ido ganando interés últimamente lo cual es comprensible dadas las limitaciones de las representaciones holísticas (El holismo enfatiza la importancia del todo, que es más grande que la suma de las partes, y da importancia a la interdependencia de éstas). Estos métodos basados en características locales son más robustos frente a variaciones en la postura o la iluminación de los métodos holísticos .

La metodología básica LBP para la extracción de características propuesto por [AHP06] es la siguiente: El algoritmo LBP introduce un nuevo enfoque para el reconocimiento facial al considerar como elementos para representar las imágenes de la cara tanto la forma como la textura. En este algoritmo el rostro se divide en pequeñas regiones de las que se extrae el patrón binario local y se concatenan en un único histograma que representa la imagen de la cara. Las texturas de las regiones faciales son localmente codificadas por LBP, mientras que toda la forma de la cara es recuperada por el histograma de la cara. La idea que subyace de usar las características LBP, es que las imágenes de la cara se puede ver como la composición de micro-patrones que son invariantes con respeto a la transformaciones monótonas de escala de grises. La combinación de estos micro-patrones generan una descripción global de la imagen de la cara, como se muestra en la figura 2.7. Este histograma tiene una descripción de la cara en tres

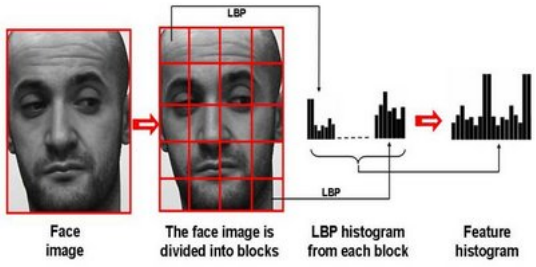


Figura 2.7: Descripción de la cara con el operador LBP.

niveles diferentes de la localidad:

1. Las etiquetas de LBP para el histograma contienen información acerca de los patrones en un nivel de píxeles.

2. Las etiquetas se resumen en una pequeña región de producción de información a nivel local.

3. Los histogramas regionales se concatenan para crear una descripción global de la cara.

Señalar que al utilizar métodos basados en histogramas las regiones no tienen porqué ser rectangulares, no tienen que tener el mismo tamaño ni ser de la misma forma, tampoco debe cubrir toda la imagen y es posible que diferentes regiones se puedan superponer parcialmente.

El método de reconocimiento facial en dos dimensiones se ha extendido en el dominio espacio-temporal [ZP07]. La Figura 2.2 muestra la descripción de la expresión facial con LBP-TOP. Con este operador se han obtenido excelentes resultados en problemas de reconocimiento de expresiones faciales.

Desde la publicación de la descripción LBP para reconocimiento facial este método ha alcanzado una posición relevante en la investigación de el análisis facial y aplicaciones, entre otros podemos citar las siguientes sistemas.

- Sistema de reconocimiento facial invariante a la iluminación propuesto por [SRSL07] combinado NIR de la imagen con LBP como vector de característica y Adaboost como clasificador.

- [WSW+05] propone la extracción de la características LBP a partir de imágenes obtenidas mediante el filtrado de una imagen facial con 40 filtros de Gabor de diferentes escalas y orientaciones, obteniendo excelentes resultados.

- [HP09] utiliza LBP espacio-temporal para el reconocimiento facial y de género a partir de secuencias de vídeo.

Notación

Usamos la siguiente notación para el operador LBP: $LBP^{u2}_{P,R}$, donde los subíndices P, R representa el operador vecindad (P, R), número de puntos y radio, el superíndice $u2$ se utiliza para representar sólo los *uniform pattern* y etiquetado todos los demás patrones (no uniformes) con una sola etiqueta.

Después de generar las etiquetas LBP de la imagen $f_l(x, y)$, el histograma LBP se puede definir como

$$H_i = \sum_{x,y} I\{f_l(x, y)\} = i, i = 0, 1, \cdots, n-1$$

donde n es el número de etiquetas diferentes producidos por el operador LBP e $I(A) = 1$ si A es cierto y 0 si A es falso. Cuando las imágenes, cuyo histogramas se van a comparar, tienen diferente tamaño, los histogramas se deben de normalizar para que la comparación sea coherente:

$$N_i = \frac{H_i}{\sum_{j=0}^{n-1} H_j}$$

El histograma contiene información sobre la distribución local de la imagen con información de: bordes, manchas y zonas planas. Para que la representación de la cara sea eficaz, también debe incluir información espacial. Para ello, la imagen se divide en regiones $R_0, R_1, \cdots, R_{M-1}$. El histograma tiene una descripción de la cara en tres niveles diferentes:

1. Las etiquetas para el histograma contienen información acerca de los patrones a nivel de píxeles
2. Las etiquetas resumen una pequeña región, esto es, se genera información a nivel regional
3. Los histogramas regional se concatenan para crear una descripción global de la cara.

Desde el punto de vista de la clasificación de patrones, un problema habitual en el reconocimiento facial es que existen muchas clases y pocas muestras de entrenamiento por clase, y en muchos casos solamente una. Por este motivo, no son necesarios clasificadores muy sofisticados, un clasificador del vecino más cercano puede ser suficiente. Se han propuesto varias medidas de disimilitud para histogramas:

1. Logaritmo de probabilidad (verosimilitud) estadística
2. Estadístico χ^2

De estas medidas se puede extender al histograma espacial mejorado, simplemente sumando i y j.

Cuando la imagen se ha dividido en regiones, es de esperar que algunas de las regiones contengan información más útil que otras, en términos de lo que realmente distingue a las personas. Por ejemplo, los ojos parecen ser una señal importante en el reconocimiento de rostros humanos. Para beneficiarnos de esto, se pueden asignar pesos a cada región en función de la la importancia de la información que contienen. Por ejemplo, el promedio ponderado estadística, sea W_j el peso de la región j, quedando una ecuación como sigue:

$$\chi^2_w(x, \xi) = \sum_{i,j} w_j \frac{(x_{i,j} - \xi_{i,i})^2}{x_{i,j} + \xi_{i,j}} \tag{2.5}$$

donde tenemos que x y ξ son los histogramas normalizados y mejorados para ser comparados, los índices i y j se refieren al i-ésimo en el histograma correspondiente a la j-ésima región local y w_j es el peso de la región j.

2.2. Mapa de píxeles de la image

Para realizar una comparación de resultados debido a las dimensiones de la imagen hemos incluido también como descriptor los píxeles de la imagen, este descriptor debería estar en la banda baja del clasificador.

Una imagen es una matriz de puntos, con un valor para cada uno de los puntos. Como sea que todas las imágenes que vamos a procesar son en blanco y negro (escala de grises), el valor de cada punto debe estar comprendido entre 0 y 255 (niveles de gris). Para cada imagen hemos de generar un vector con dimensión igual al producto de las dimensión de las imágenes, si la imagen tiene dimensión de nxm píxeles la dimensión del vector será la del producto de mxn y este vector caracteriza a la imagen del cual es deducida.

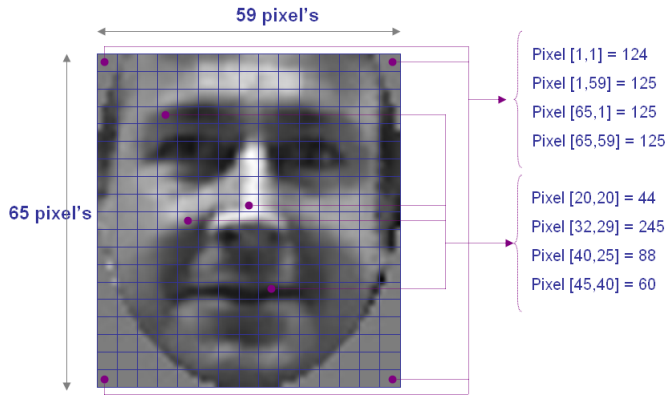


Figura 2.8: Matriz de una imagen. Ejemplo de descriptor basado en píxeles

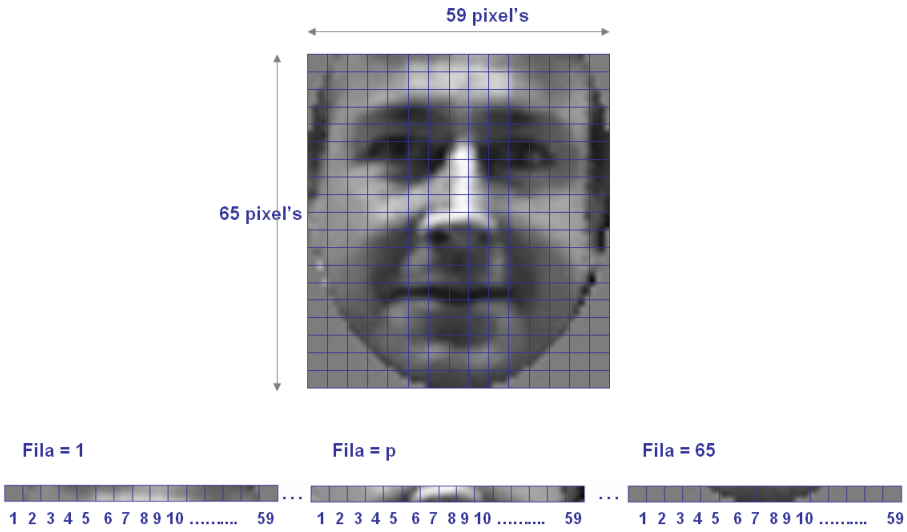


Figura 2.9: Imagen descompuesta píxeles

2.3. Análisis de componentes principales PCA

PCA [Shl09] es el principal método utilizado para disminuir la dimensión de las características asociadas a una imagen reduciendo la carga computacional y, como sea que la imágenes de las caras en este trabajo tienen el tamaño 59 x 65 píxeles, lo que da un total de 3.835 puntos por imagen, utilizar un procedimiento que permita limitar la dimensión del vector de características,a priori, se antoja una estrategia adecuada.

PCA es una herramienta estándar en el análisis de datos y se utiliza en diversos campos de la neurociencia, reconocimiento de patrones e imágenes, indexación de contenidos en la web, en la meteorología y oceanografía, entre otras, debido a que es simple y no paramétrico, y permite extraer información relevante de un conjunto de datos, con una ingente cantidad de información que además es confusa.

Con un mínimo esfuerzo PCA proporciona un método que a partir de un conjunto de datos complejos genera un nuevo espacio de datos, con una dimensión inferior que permite que florezcan estructuras simplificadas. Este método reduce la dimensionalidad del espacio de datos, extrayendo la información relevante del mismo. Hay otro fenómeno a tener en cuenta que afecta a los datos, el ruido, que todavía dificulta más la posibilidad de extraer conclusiones de los datos

2.3.1. Cambio de base

PCA se centra en la identificación de la base más significativa que permita volver a expresar el conjunto de datos. La nueva base elimina el ruido y revela estructuras ocultas en el conjunto de datos. El objetivo de PCA es determinar el vector de la base unitario que permitirá discernir entre lo importante y lo redundante o ruidoso.

Cada una de las imágenes o muestras ejemplo se pueden expresar en forma de un vector columna de dimensión nxm de la forma:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ x_{m+1} \\ \vdots \\ x_{nxm} \end{bmatrix}$$

Cada imagen es un vector que se encuentra en un espacio vectorial de dimensión nxm y dicho espacio vectorial es generado por una base ortonormal. De álgebra lineal, sabemos que todos los vectores de espacio vectorial se forman como una combinación lineal de este conjunto de vectores (base de dicho espacio vectorial) con norma unitaria.

PCA se basa en un pre-requisito estricto: **la linealidad**. Esta condición simplifica el problema, restringiendo el conjunto de bases potenciales. Con esta suposición PCA se limita a la hora de volver a expresar los datos como una combinación lineal de los vectores de su base.

Sea X el conjunto de datos original, donde cada columna es una sola muestra, sea Y otra matriz m $\times$ n relacionada mediante un transformación lineal P. La matriz Y es la nueva representación de los datos, esto es, Y son los componentes principales de X.

$$XP = Y \tag{2.6}$$

Se definen los siguientes términos:

p_i : filas de P.
x_i: las columnas de X [nuestras imágenes].
Y_i : las columnas de Y.

La ecuación (2.6) representa un cambio de base y puede tener, entre otras, las siguientes interpretaciones:

1. P es una matriz que transforma X en Y.

2. Geométricamente, P es una rotación y un tramo que a su vez transforma X en Y.

3. Las filas de $P, p_1, ..., p_m$, son un conjunto de vectores de la nueva base términos de referencia para expresar las columnas de X.

La tercera interpretación no es obvia, pero puede ser vista como el producto explícito de PX.

$$PX = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \tag{2.7}$$

por lo que Y queda de la siguiente forma:

$$Y = \begin{bmatrix} p_1 \cdot x_1 & \cdots & p_1 \cdot x_n \\ p_2 \cdot x_1 & \cdots & p_2 \cdot x_n \\ \vdots & \ddots & \vdots \\ p_m \cdot x_1 & \cdots & p_m \cdot x_n \end{bmatrix}$$

Cada vector columna de Y lo podemos representar como:

$$Y_i = \begin{bmatrix} p_1 \cdot x_i \\ p_2 \cdot x_i \\ \vdots \\ p_m \cdot x_i \end{bmatrix}$$

Por lo tanto cada coeficiente y_i es un producto de la x_i con la fila correspondiente en la matriz P. En otras palabras, el coeficiente -$j^{th} y_i$ es una proyección sobre la fila j-ésima de P. Esto es la propia forma de una ecuación donde y_i es una proyección sobre la base $p_1, \cdots, p_m$. Por lo tanto, las filas de P son el nuevo conjunto de vectores de la base para representar las columnas de X.

Al asumir como pre-requisito la linealidad, el problema se reduce a encontrar el cambio de base apropiado. Los vectores fila $p_1, \cdots, p_m$, en esta transformación, se convertirán en las componentes principales de X.

2.3.2. La varianza y los objetivos

Los objetivos PCA son eliminar el ruido y los datos redundantes de forma que aflore la información realmente relevante y que permitan expresar lo que significan los datos.

Ruido

La medición del ruido en cualquier conjunto de datos debe ser baja o de lo contrario, no importa la técnica de análisis que se utilice, no será posible extraer información acerca de una señal. No existe una escala absoluta de ruido, una medida común que se utiliza es la relación entre la señal y el ruido (SNR), es una relación de varianzas σ^2:

$$SNR = \frac{\sigma^2_{\text{señal}}}{\sigma^2_{\text{ruido}}} \tag{2.8}$$

Una alta relación señal ruido ($SNR \gg 1$) indica una medición de alta precisión, mientras que si el índice es bajos indica que los datos son muy ruidoso.

Redundancia

La redundancia es otro de los problemas que nos podemos encontrar a la hora de extraer información relevante por lo que la misma deberá ser eliminada y esta es la idea central que hay detrás de reducir la dimensionalidad de los datos, que es el objetivo de PCA.

2.3.3. Matriz de covarianza

En el caso de dos variables es fácil identificar los casos redundantes, para ello calculamos la pendiente de la recta que mejor los ajuste y juzgamos la calidad de dicho ajuste. Para cualquier número de dimensiones:

Consideremos dos conjuntos de mediciones con media cero:

$A = a_1, a_2, \cdots, a_n, B = b_1, b_2, \cdots, b_n$
donde el subíndice indica el número de muestra. Las varianzas de A y B se calculan como:

$$\sigma^2_A = \frac{1}{n}\sum_i a_i^2 \, , \sigma^2_B = \frac{1}{n}\sum_i b_i^2$$

Y la covarianza entre las muestras A y B es una generalización directa:

$$\text{Covarianza de A y B} = \sigma^2_{AB} = \frac{1}{n}\sum_i a_i b_i$$

La covarianza mide el grado de relación lineal entre dos variables. Un valor positivo indica grandes posibilidades que los datos estén correlacionados. Del mismo modo, un valor negativo, elevado, indica que los datos de las variables no están correlacionados. La magnitud absoluta de la covarianza mide el grado de redundancia. Otras propiedades de la covarianza .

- σ_{AB} es cero si y sólo si A y B están correlacionados .
- $\sigma^2_{AB} = \sigma^2_A$ si $A = B$.

Convertir A y B en vectores fila:
$a = [a_1 a_2 \cdots a_n]$
$b = [b_1 b_2 \cdots b_n]$
de manera que podemos expresar la covarianza como un producto de matrices, de la siguiente forma:

$$\sigma^2_{ab} = \frac{1}{n} a b^T \tag{2.9}$$

Generalizamos a partir de dos vectores a un número arbitrario para ello se cambia el nombre de los vectores fila a y b por x_1 y x_2, respectivamente, y consideran otros vectores fila $x_3, ..., x_m$. Se Define una nueva matriz X de dimensiones m $\times$ n.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Cada fila de X se corresponde con todas las mediciones de un tipo particular. Cada columna de X corresponde a una muestra en particular. Y la definición de la matriz de covarianza C_X.

$$C_X = \frac{1}{n} X X^T \tag{2.10}$$

Consideremos la matriz de la ecuación (2.10) el elemento i^{jth} de C_X como el producto escalar entre el vector del tipo de medición i con el vector del tipo de medición j . Algunas de las propiedades de la covarianza C_X son:

- C_X es una matriz cuadra y simétrica de dimensión m $\times$ m.
- Los términos de la diagonal de C_X son las varianzas particulares de los tipos de medición.
- Los términos fuera de la diagonal de C_X son las covarianzas entre los tipos de medición.

C_X calcula la covarianza entre todos los posibles pares de mediciones. Los valores de covarianza reflejan el ruido y la redundancia en nuestras mediciones y los valores los podemos interpretar de la siguiente forma:

- En la diagonal: los valores grandes se corresponden a estructuras interesante.
- Fuera de la diagonal: Valores elevados corresponden a alta redundancia.

2.3.4. Cálculo de PCA mediante descomposición autovectores

Si podemos manipular C_X, podríamos definir una nueva matriz de covarianza C_Y de forma que:

- Aquella en la que todos los términos fuera de la diagonal en C_Y son cero. Por lo tanto, C_Y debe ser una matriz diagonal. O, dicho de otra manera, Y no está correlacionada.
- Cada dimensión de Y debe ser jerarquizada de acuerdo a la varianza.

Por el álgebra lineal podemos calcular PCA mediante la descomposición de vectores propios. El conjunto de datos X, es una matriz de dimensión m $\times$ n, donde m es el número de tipos de medidas o atributos y n es el número de muestras. El objetivo se resume como sigue.

Encontrar una matriz ortonormal P que $Y = PX$ tal que $C_Y \equiv \frac{1}{n}YY^T$ sea una matriz diagonal, y las filas de P son los componentes principales de X.

Empezamos por re-escribir C_Y en términos de variables desconocidas.

$$\begin{aligned} C_Y &= \frac{1}{n}YY^T \\ &= \frac{1}{n}(PX)(PX)^T \\ &= \frac{1}{n}PXX^TP^T \\ &= P\frac{1}{n}(XX^T)P^T \\ C_Y &= PC_XP^T \end{aligned} \tag{2.11}$$

En la última línea de las ecuación (2.11) tenemos la matriz de covarianza de X.

Del álgebra lineal sabemos que cualquier matriz simétrica A es diagonalizable por una matriz ortogonal de sus vectores propios. Una matriz simétrica A se puede descomponer como: $A = EDE^T$, donde D es una matriz diagonal y E es una matriz de vectores propios de A dispuestas en forma de columnas

Si seleccionamos la matriz P como una matriz donde cada fila p_i es un vector propio de una XX^T. Con esta decisión, $P \equiv E$. Con esta relación y por el álgebra lineal sabemos que ($P^{-1} = P^T$) terminamos de evaluar C_Y:

$$\begin{aligned} C_Y &= PC_XP^T \\ &= P(E^TDE)P^T \\ &= P(P^TDP)P^T \\ &= (PP^T)D(PP^T) \\ &= (PP^{-1})D(PP^{-1}) \\ C_Y &= D \end{aligned} \tag{2.12}$$

Es evidente que la elección de P diagonaliza C_Y que era el objetivo de PCA. Podemos resumir los resultados de PCA en las matrices P y C_Y.

- Los componentes principales de X son los vectores propios de la $C_X = XX^T$.
- Y el valor i^{th} de la diagonal de C_Y es la varianza de X a lo largo de p_i.

En la práctica para calcular PCA de un conjunto de datos X implica:

1. Restar la media de cada tipo de medición.
2. Calcular los vectores propios de la C_X.

Esta solución se muestra en el código de MATLAB incluido en el anexo

2.3.5. SVD para el cálculo de PCA

Otra solución algebraica para el cálculo de PCA es mediante Descomposición de Valores Singulares (SVD), en el proceso de cálculo de esta solución, observamos que PCA está estrechamente relacionado con SVD, a menudo se utilizan de forma intercambiable.SVD proporciona un método general para entender el cambio de base.

SVD

Sea X una matriz arbitraria n $\times$ m y X^TX sea de rango r, cuadrada, simétrica m $\times$ m. Se definen los siguiente valores:

- Los vectores $\{\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_r\}$ es el conjunto de vectores propios ortonormales $m \times 1$ con valores propios asociados $\{\lambda_1, \lambda_2, \cdots, \lambda_r\}$ para la matriz simetrica X^TX.

$$(X^TX)\widehat{v}_i = \lambda_i \widehat{v}_i$$

- $\sigma_i \equiv \sqrt{\lambda_i}$ son reales positivos y valores singulares.
- $\{u_1, u_2, \cdots, u_r\}$ es el conjunto de $n \times 1$ vectores definidos por: $u_i \equiv \frac{1}{\sigma_i} X\widehat{v}_i$.

La definición final incluye dos nuevos e inesperados propiedades.

- $|\widehat{u}_i \cdot \widehat{u}_j| = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{en otro caso} \end{cases}$
- $\|X\widehat{v}_i\| = \sigma_i$

Con esto disponemos de todas las piezas para la construcción de la descomposición. La versión escalar de descomposición de valor singular es sólo una re-afirmación de la tercera definición.

$$X\widehat{v}_i = \sigma_i \widehat{u}_i \tag{2.13}$$

Este resultado dice que X multiplicado por el vector propio de X^TX es igual a un escalar por otro vector. El conjunto de vectores $v_1, v_2, \cdots, v_r$ y el conjunto de vectores $u_1, u_2, \cdots, u_r$ son dos conjuntos ortonormales o bases en r el espacio r-dimensional.

Podemos resumir este resultado para todos los vectores en una multiplicación de matrices, siguiendo la construcción indicadas en la figura 2.10 Empezamos por la construcción de una nueva matriz diagonal Σ.

$$\Sigma \equiv \begin{bmatrix} \sigma_i & 0 & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ 0 & & & & & 0 \end{bmatrix} \tag{2.14}$$

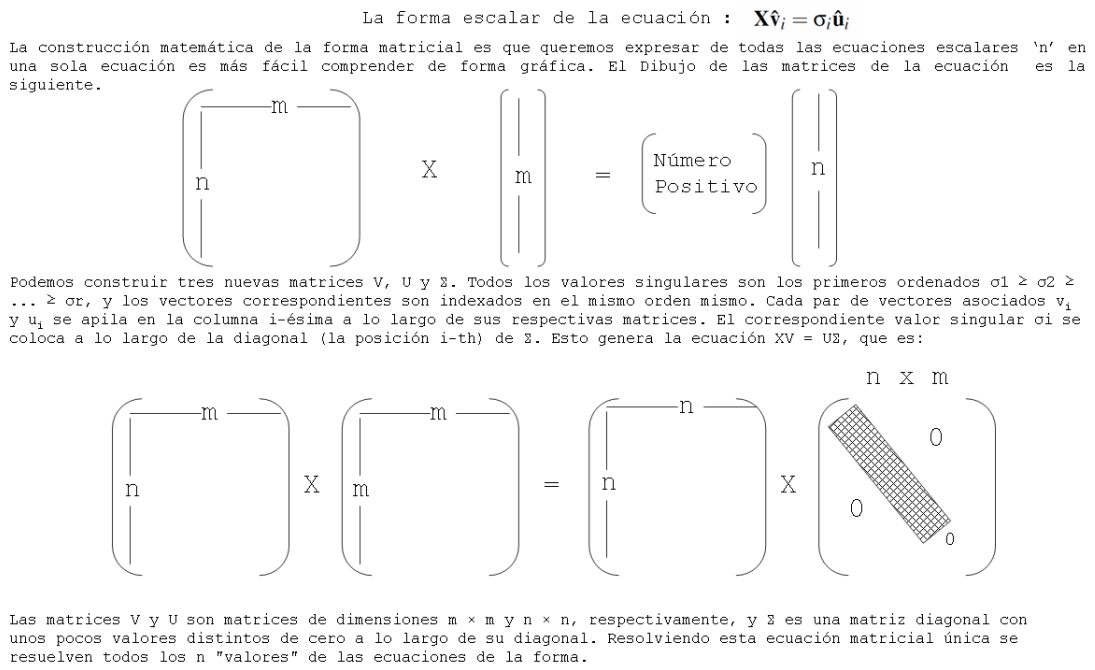


Figura 2.10: La construcción de la forma de la matriz de SVD (Ecuación (2.15) de la forma escalar (Ecuación (2.13)).

donde $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ son el rango del conjunto valores singulares. Del mismo modo se construyen las matrices ortogonales,

$$\begin{aligned} V &= \begin{bmatrix} \widehat{v_1} & \widehat{v_2} & \cdots & \widehat{v_m} \end{bmatrix} \\ U &= \begin{bmatrix} \widehat{u_1} & \widehat{u_2} & \cdots & \widehat{u_n} \end{bmatrix} \end{aligned}$$

donde se han añadido $(m-r)$ y $(n-r)$- vectores ortonormales adicionales para 'completar' las matrices de V y U, respectivamente (es decir, para hacer frente a los problemas de degeneración).

$$XV = U\Sigma$$

donde cada columna de U y V son la versión escalar de las descomposición de la ecuación (2.13). Debido a que V es ortogonal, se puede multiplicar ambos lados de la igualdad por $V^{-1} = V^T$ para llegar a la forma final de la descomposición.

$$X = U\Sigma V^T \tag{2.15}$$

Esta descomposición es bastante potente. La ecuación (2.15) establece que toda matriz arbitraria X se puede convertir en una matriz ortogonal, una matriz diagonal y otra matriz ortogonal (una rotación, un estiramiento y una segunda rotación).

Interpretación de SVD

La forma final de la ecuación SVD es una fórmula concisa pero poco refinada. Para entenderla mejor vamos re-interpretar la ecuación (2.13).

$$Xa = Kb$$

Donde a y b son vectores columna y K es una constante escalar, el conjunto $\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_m$ es análogo a a y $\widehat{u}_1, \widehat{u}_2, \cdots, \widehat{u}_n$ lo es a b. Lo que es único es, sin embargo, que $\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_m$ y $\widehat{u}_1, \widehat{u}_2, \cdots, \widehat{u}_n$ son conjuntos ortonormales de vectores que abarcan un espacio de m o n dimensiones, respectivamente. En particular, en términos generales estos conjuntos parecen abarcar todas las posibles 'entradas' (es decir, a) y 'productos' (es decir, b). ¿Se puede formalizar que $\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_m$ y $\widehat{u}_1, \widehat{u}_2, \cdots, \widehat{u}_n$ abarquen todas los posibles 'entradas' y 'salidas'?

Se puede manipular la ecuación (2.15) para que esta hipótesis difusa sea más precisa.

$$\begin{aligned} X &= U\Sigma V^T \\ U^T X &= \Sigma V^T \\ U^T X &= Z \end{aligned}$$

Donde Z es igual ΣV^T

Ahora las columnas anteriores$\widehat{u}_1, \widehat{u}_2, \cdots, \widehat{u}_n$ son filas en U^T. Comparando esta ecuación con la ecuación (2.6),$\widehat{u}_1, \widehat{u}_2, \cdots, \widehat{u}_n$ realiza el mismo papel que $\widehat{p}_1, \widehat{p}_2, \cdots, \widehat{p}_m$, por lo tanto, U^T es un cambio de base de X a Z. Al igual que antes se puede inferir transformando en vectores columna. El echo de que la base sea ortonormal U^T (o P) se transformen en vectores columna significa que U^T es una base que se extiende por las columnas de X. Las bases que extienden columnas se denomina como el espacio columna de X. El espacio de columnas formaliza la noción de cuáles son las posibles 'salidas' de cualquier matriz.

Hay una simetría interesante en SVD de tal manera que podemos definir de forma similar - el espacio de filas-.

$$\begin{aligned} XV &= \Sigma V \\ (XV)^T &= (\Sigma V)^T \\ V^T X^T &= V^T \Sigma^T \\ V^T X^T &= Z \end{aligned}$$

donde hemos definido $Z \equiv U^T\Sigma$. Una vez más las filas de la V^T (o las columnas de V) son una base ortonormal para la transformación de X^T en Z. A causa de la transposición de X, se deduce que V es una base ortonormal que abarca el espacio de filas de X. El espacio de filas también formaliza el noción de lo que es posible 'entradas' en una matriz arbitraria.

SVD y PCA

Es evidente que PCA y SVD están estrechamente ligados, volvamos a la matriz original X de dimensiones bxm, se puede definir una nueva matriz Y de dimensiones nxm

$$Y \equiv \frac{1}{\sqrt{n}} X^T \tag{2.16}$$

donde cada columna de Y tiene de media cero. La elección de Y empieza ser clara analizando $Y^T Y$:

$$\begin{aligned} Y^T Y &= (\frac{1}{\sqrt{n}} X^T)^T (\frac{1}{\sqrt{n}} X^T) \\ &= \frac{1}{n} X X^T \\ Y^T Y &= C_X \end{aligned}$$

La construcción de Y es igual a la matriz de covarianza de X, de la sección 3.2.5 se sabe que los componentes principales de X son los vectores propios de la C_X. Si calculamos SVD de Y, las columnas de la matriz V contiene los vectores propios de $Y^T Y = C_X$. Por lo tanto, las columnas de V son los componentes principales de X. Este algoritmo se resume en segundo código en Matlab incluyó en el Anexo.

¿Qué significa esto? V se extiende por el espacio de filas $Y \equiv \frac{1}{\sqrt{n}} X^T$. Por lo tanto, V también debe abarcar el espacio de columnas de $\frac{1}{\sqrt{n}} X$. Se puede concluir que la búsqueda de componentes principales es encontrar una base ortonormal que se extiende por el espacio de columnas de X.

2.3.6. Discusión y límites de PCA

PCA se ha extendido a diferentes aplicaciones, porque revela estructuras simples subyacentes de datos complejos y establece el uso de soluciones analíticas de álgebra lineal. Un breve resumen de un algoritmo para el cálculo de PCA sería:

1. Organizar el conjunto de datos como una matriz de dimensión mxn, donde m es el número de medidas y n es el número de muestras.

2. Restar la media a cada tipo de medida.

3. Calcular SVD o los vectores propios de la matriz de covarianza

Una de las ventajas de PCA es que determina la importancia de cada dimensión para describir la variabilidad de un conjunto de datos. En particular, la medición de la variación a lo largo de cada componente principal proporciona un medio para comparar la importancia relativa de cada dimensión. En este método la variación a lo largo de un pequeño número de componentes principales (menor que el número de tipos de medición) presenta una caracterización razonable del conjunto completo de datos.

A pesar de que PCA se ha utilizando en multitud de problemas reales no por ello debemos de hacernos la siguiente pregunta, ¿Cuándo falla PCA? Antes de responder a esta pregunta, veamos una característica destacada de este método, PCA es completamente no-paramétrico: A cualquier conjunto de datos se le

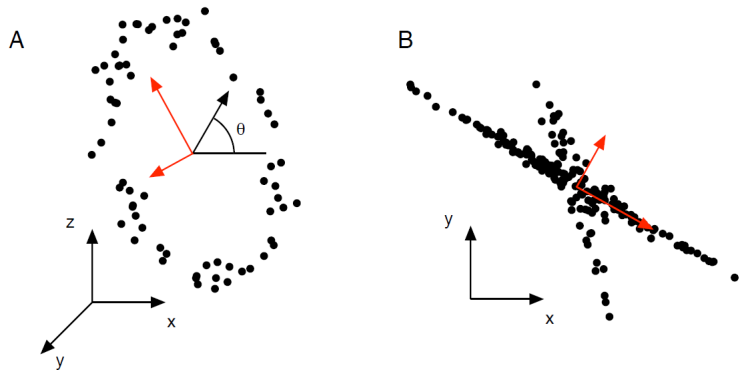


Figura 2.11: Ejemplo de PCA cuando falla (líneas rojas). (a) Seguimiento de una persona en una noria (puntos negro). Toda la dinámica puede ser descrita por la fase de la noria θ, una combinación no lineal de la base sencilla. (b) En este conjunto de datos de ejemplo, una función no gaussiana de datos distribuida en ejes no ortogonales causas que la PCA no funcione. Los ejes con mayor varianza no se corresponden con la respuesta adecuada.

puede aplicar PCAy obtendremos una salida. El que PCA sea no paramétrica puede considerarse como un rasgo positivo porque la respuesta es única e independiente del usuario. Desde otra perspectiva que PCA sea independiente de la fuente de los datos es también una debilidad. Por ejemplo, considere el seguimiento de una persona en una noria como aparece en en la figura 2.11 a. Los puntos de datos puede ser limpiamente descrito por una sola variable, el ángulo de la noria θ, sin embargo, PCA no puede recuperar esta variable.

Un análisis más profundo de los límites de PCA requiere algunas consideraciones acerca de los supuestos subyacentes y una descripción más rigurosa de la fuente de datos. En términos generales, la principal motivación que hay detrás de este método es des-correlacionar el conjunto de datos, es decir, eliminar el segundo orden de dependencia. La manera de abordar este objetivo es ligeramente similar a cómo se podría explorar una ciudad: reducir el camino más largo que atraviesa la ciudad. Cuando uno ve otro camino, girar a la izquierda o la derecha y seguir por este camino, y así sucesivamente. En esta analogía, PCA requiere que cada nuevo camino explorado debe ser perpendicular a la anterior, pero es evidente que este requisito es demasiado estricto y los datos se puede organizar a lo largo de ejes no ortogonales, como por ejemplo la figura 2.11 b. La figura 2.11 proporciona dos ejemplos de este tipo de datos donde PCA no proporciona resultados satisfactorios. Para abordar estos problemas es necesario que definamos lo que consideramos como los mejores resultados. Si hablamos de disminuir la dimensional una medida del éxito es el grado en que el espacio de representación se reduce. En términos estadísticos, es necesario definir una función de error (o pérdida de función). Se puede demostrar que en una función de pérdida común, el error cuadrático medio (es decir, L2 norma), PCA proporciona la óptima reducción del espacio de representación de los datos. Esto significa que la selección de direcciones ortogonales de los componentes principales es la mejor solución para predecir los datos originales. ¿Cómo podemos asegurar que esta declaración es cierta? La figura 2.11 indica que este resultado es de alguna manera engañosa. La solución a esta paradoja radica en el objetivo que hemos seleccionado para el análisis, descorrelacionar los datos, o dicho en otros términos, el objetivo es eliminar el segundo orden de dependencia. En los conjuntos de datos de la figura 2.11, existe un aumento del orden de dependencia entre las variables. Por lo tanto, la eliminación del segundo orden de dependencias no es suficiente, para que afloren todas las

estructuras de los datos.

Existen soluciones para la eliminación de orden superior dependencias. Mediante la aplicación de una transformación a los datos para resolver el problema de la no linealidad, para los datos de la figura 2.11 se podría aplicar la polar para su representación. Este enfoque paramétrico a veces se denomina *Kernel* PCA. Y otra posibilidad es la de imponer más estadística general sobre definiciones de dependencia dentro de un conjunto de datos. Esta clase de algoritmos, llamados, Análisis de Componentes Independientes (ICA), ha demostrado tener éxito en muchos campos donde no tiene éxito el PCA.

2.4. Algoritmos de aprendizaje. Máquina vector soporte (SVM)

En la última década, la Máquina Vector Soporte [SVM] [Bur98] se ha convertido en una de las técnicas más utilizadas para tareas de clasificación automática de imágenes y de reconocimiento facial, debido a los buenos resultados que se han obtenido. Este algoritmo de aprendizaje automático se basa en la representación de las imágenes en un espacio vectorial y asume que las imágenes de cada clase se agrupan en regiones separables del espacio de representación y, teniendo en cuenta esto, el algoritmo busca un hiperplano que separe cada clase, maximizando la distancia entre las imágenes y el propio hiperplano, lo que se denomina margen. El hiperplano se define mediante la siguiente función:

$$f(x) = wx + b$$

La optimización de esta función supone tener en cuenta todos los valores posibles de w y b, para después quedarse con aquéllos que maximicen los márgenes. Esta función resulta difícil de optimizar, por lo que en la práctica se utiliza la siguiente función de optimización equivalente:

$$min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} x_i^d$$

Sujeto a:

$$y_i(wx_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

donde C es la penalización y ξ_i es la distancia entre el hiperplano y la imagen i.

De esta forma, únicamente, se resuelven problemas linealmente separables, por lo que en muchos casos se requiere la utilización de una función de *Kernel* para la redimensión del espacio. Así, el nuevo espacio obtenido resultará linealmente separable. Posteriormente, la redimensión se deshace de modo que el hiperplano encontrado será transformado al espacio original, constituyendo la función de clasificación.

Es importante destacar que esta función únicamente puede resolver problemas binarios y el aprendizaje debe ser supervisado.

La Máquina Vector Soporte ofrece un enfoque al problema de reconocimiento de patrones, con claras conexiones con la teoría del aprendizaje estadístico subyacente. SVM se caracteriza por la elección

de su *Kernel*, y por lo tanto SVM, se puede utilizar para afrontar los problemas que han sido diseñados sobre los métodos basados en *Kernel*.

Las características principales de SVM son:

- Utiliza *Kernel*
- Ausencia de mínimos locales.
- Escasa densidad de la solución.
- Control de la solución obtenida mediante la optimización de los márgenes.

2.4.1. Máquina vector soporte- lineal

Caso simple: datos separables

El caso más simple que podemos encontrarnos es el caso en el que los datos son separables. Primero se etiquetan los datos, de entrenamiento (nos encontramos utilizando un algoritmo de aprendizaje supervisado):

$x_i, y_i, i = 1, \cdots, l, y_i \epsilon \{-1, 1\}, x_i \epsilon R^d$.

Ahora suponemos que existe un hiperplano que separa las muestras positivas de las negativos (un 'hiperplano de separación'). Los puntos x que se encuentran en el hiperplano deben cumplir la ecuación $w\Delta x + b = 0$, donde w es normal al hiperplano, y $\frac{|b|}{\|w\|}$ es la distancia perpendicular del hiperplano al origen, y $\|w\|$ es la norma Euclidea de w . Se define $d_+(d_-)$ como la distancia más corta desde el hiperplano de separación a la muestra más cercana positiva (negativa). Se define el margen de un hiperplano como: $d_+ + d_-$.

Para el caso linealmente separable, el algoritmo SVM busca el hiperplano de separación con mayor margen.

Esto se puede formular de la siguiente manera:
supongamos que todos los datos de entrenamiento satisfacen las siguientes restricciones:

$$x_i w + b \geq +1 \text{ para } y_i = +1 \tag{2.17}$$
$$x_i w + b \leq -1 \text{ para } y_i = -1 \tag{2.18}$$

Estas dos desigualdades se pueden unir , quedando:

$$y_i(x_i w + b) - 1 \geq 0 \forall i \tag{2.19}$$

Consideremos ahora los puntos que cumplen la igualdad (2.17), estos puntos se encuentran en el hiperplano $H_1 : xi\Delta w + b = 1$ con w normal y la distancia perpendicular desde el origen $\frac{|1-b|}{\|w\|}$. Del mismo modo, los puntos que cumplen la igualdad (2.18) se encuentran en el hiperplano: $H_2 : xi\Delta w + b = -1$,con

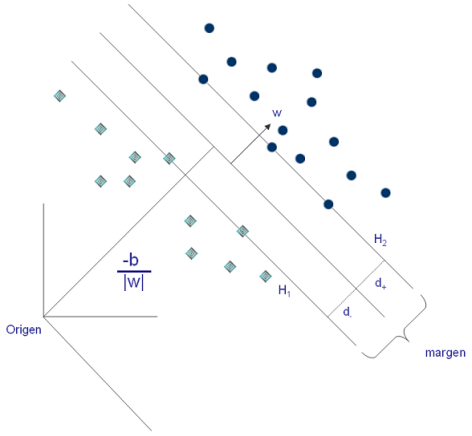


Figura 2.12: SVM lineal, caso separables.

w normal y la distancia perpendicular desde el origen $\frac{|-1-b|}{\|w\|}$. Por lo tanto $d_+ = d_- = \frac{1}{\|w\|}$ y el margen es más que $\frac{2}{\|w\|}$. Los hiperplanos H_1 y H_2 son paralelos (tienen la misma normal) y no hay puntos de los ejemplos que se encuentren entre ellos. Se puede encontrar el par de hiperplanos con el margen máximo, reduciendo al mínimo $\|w\|^2$, sujeto a la ecuación (2.17). Por lo tanto esperamos que la solución para un caso típico de dos dimensiones, tenga la forma que se indica en la figura 2.12. El conjunto de entrenamiento para los que se cumple la igualdad de la ecuación (2.19), y cuya eliminación cambiaría la solución encontrada, se denominan **vectores soporte**, que son los puntos que se encuentran en los planos H_1 y H_2 como se puede observar en la figura 2.12.

Ahora cambiamos a una fórmula de Lagrange por:

1. La restricción de la ecuación (2.19) será sustituido por las limitaciones de los multiplicadores de Lagrange, que serán más fáciles de usar.

2. En la reformulación del problema los datos de entrenamiento sólo aparecen en forma de productos entre vectores. Esta es una propiedad fundamental que nos permitirá generalizar el procedimiento para el caso no lineal.

Por lo tanto,se introducen cambios positivos con los multiplicadores de Lagrange $\alpha_i, i = 1, \cdots, l$, uno para cada una de las restricciones de desigualdad de la ecuación (2.19). Para las ecuaciones de la forma $c_i \geq 0$, las ecuaciones de restricción se multiplican por los multiplicadores de Lagrange positivos y restan al margen de la función objetivo, para formar el lagrangiano. Esto da la siguiente ecuación de Lagrange:

$$L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i w + b) + \sum_{i=1}^{l} \alpha_i \tag{2.20}$$

Ahora, minimizamos L_P con respecto a w y b simultáneamente y se debe cumplir que la derivada de L_P con respecto a todos los multiplicadores de Lagrange α_i desaparecen siempre que $\alpha_i \geq 0$ lo llamamos (Caso A).

Ahora bien, este es un problema de programación cuadrática convexa, ya que la función objetivo es convexa en sí, y los puntos que satisfacen las restricciones también forman un conjunto convexo (cualquier restricción lineal define un conjunto convexo, y un conjunto de N restricciones lineales simultáneas define la intersección de N conjuntos convexos, que también es un conjunto convexo). Esto significa que se puede resolver, de forma equivalente, el siguiente problema dual: maximizar el L_P, pero sujeto a las restricciones de las que la pendiente de L_P con respecto a w y b se eliminan, y sujeto también a la limitación de que el $\alpha_i \geq 0$ lo llamamos (Caso B). Esta formulación dual del problema se llama la doble Wolfe [Fle87] . Tiene la propiedad de que el máximo de L_P, está sujeto a las limitaciones del (Caso B), se produce, en los mismos valores de la w, b y α, como el mínimo de L_P, sujeto a las restricciones del (Caso A).
La exigencia de que el gradiente de L_P con respecto a w y b desaparezca generan las siguientes condiciones:

$$w = \sum_i \alpha_i y_i X_i \tag{2.21}$$

$$\sum_i \alpha_i y_i = 0 \tag{2.22}$$

Como se trata de restricciones de igualdad en la formulación dual, se pueden sustituir en la ecuación (2.20) quedando:

$$L_D = \sum_i \alpha_i - \frac{1}{2}{}_{i,j}\alpha_i \alpha_j y_i y_j X_i \cdot X_j \tag{2.23}$$

Se asignan etiquetas para la ecuaciones de Lagrange (P para la primera y D para la segunda) Las formulaciones para L_P y L_D son diferentes: L_P y L_D aunque se derivan de la misma función objetivo pero con diferentes restricciones, y la solución se encuentra al minimizar L_P o al maximizar L_D. Si se formula el problema con $b = 0$, lo que equivale a exigir que todos los hiperplanos contengan el origen, la restricción de la ecuación (2.22) no aparece. Esto equivale a reducir el número de grados de libertad a uno.

La formación de vectores soporte equivale a maximizar L_D con respecto a α_i y sujeto a la restricción de la ecuación (2.22) y a que los α_i sean positivos, con la solución dada por la ecuación (2.21). Se tienen un multiplicador (α_i) de Lagrange para cada uno de los puntos (muestras). En la solución, los puntos para los que $\alpha_i > 0$ se llaman 'vectores de soporte', y se encuentran en uno de los hiperplanos $H1$ o $H2$. El resto de puntos tienen $\alpha_i = 0$ y se encuentran ya sea en $H1$ o $H2$ como en ese lado del $H1$ o $H2$ que es la desigualdad estricta y se mantiene en la ecuación (2.19). Para estas, los vectores de soporte, son los elementos críticos del conjunto de entrenamiento. Se encuentran más cerca de la frontera de decisión, y si todos los puntos de otro tipo de formación se eliminan y se repite el experimento se obtiene el mismo hiperplano de separación.

Condición de Karush-Kuhn-Tucker (KKT)

La condición Karush-Kuhn-Tucker desempeñan un papel fundamental en la teoría y la práctica de la optimización con restricciones. Para el problema anterior las condiciones de KKT [Fle87] son:

$$\frac{\partial}{\partial w_v} L_P = w_v - \sum_i \alpha_i y_i x_{iv} = 0, v = 1, \cdots, d \tag{2.24}$$

$$\frac{\partial}{\partial b} L_p = -\sum_i \alpha_i y_i = 0 \tag{2.25}$$

$$y_i(x_i \cdot w + b) - 1 \geq 0, i = 1, \cdots, l \tag{2.26}$$

$$\alpha_i \geq 0 \forall i \tag{2.27}$$

$$\alpha_i(y_i(w \cdot x_i + b) - 1) = 0 \forall i \tag{2.28}$$

Las condiciones KKT se cumplen en la solución de cualquier problema de optimización con restricciones (convexo o no), con cualquier tipo de restricciones, siempre que la intersección del conjunto de direcciones posibles con el sistema de direcciones descenso coincida con la intersección del conjunto de posibles direcciones de las limitaciones lineal con el conjunto de direcciones de descenso (ver [Fle87] ; [McC83]). Esta hipótesis de regularidad es válida para todas las máquinas de vectores soporte, ya que las limitaciones siempre son lineales. Además, el problema de SVM, es convexo (una función objetivo convexa, con las limitaciones que le da una región factible convexa), y para problemas convexos (si tiene la condición de regularidad), las condiciones KKT son necesarias y suficientes para w, b, y α a ser una solución [Fle87]. Resolviendo así el problema de SVM es equivalente a encontrar una solución a las condiciones KKT.

Como una aplicación inmediata mientras que w está expresamente determinado por el procedimiento de entrenamiento, el umbral de b no, a pesar de que está implícitamente determinado. Sin embargo, b es fácil de encontrar mediante el uso de la condición de KKT de la 'complementariedad' de la ecuación (2.28), mediante la elección cualquier i para que $\alpha_i \neq 0$ y calcular b (es numéricamente más seguro tomar el valor promedio de b como resultado de todas las ecuaciones de este tipo).

Hasta ahora lo que se ha hecho es convertir el problema en un problema de optimización, donde las restricciones son bastante más manejable que las de las ecuaciones (2.17) y (2.18). Encontrar la solución para los problemas del mundo real que, por lo general, requieren métodos numéricos.

Caso no separable

Si aplicamos el algoritmo anterior cuando los datos son no separables, no se podrá encontrar ninguna solución viable: esto lo evidencia la función objetivo (Lagrange) ya que crece de forma arbitraria. Entonces, ¿cómo podemos extender estas ideas para manejar datos no linealmente separables? Se deben reducir las limitaciones de las ecuaciones (2.17) y (2.18), pero sólo cuando sea necesario, es decir, se introduce un costo adicional (es decir, un aumento en la función objetivo) para hacerlo. Esto se puede hacer mediante la introducción de variables de holgura positivas $\xi_i, i = 1, \cdots, l$ a las restricciones [Cor95]), que luego se convierten en:

$$x_i \cdot w + b \geq +1 - \xi_i \text{ para } y_i = +1 \quad (2.29)$$
$$x_i \cdot w + b \leq -1 - \xi_i \text{ para } y_i = -1 \quad (2.30)$$
$$\xi_i \geq 0 \forall i \quad (2.31)$$

Así, por un error que se produzca la correspondiente ξ_i debe superar la unidad, por lo que $\sum_i \xi_i$ es el límite superior en el número de errores de entrenamiento. Por lo tanto, una manera natural de asignar un costo extra a los errores es cambiar la función objetivo a minimizar de $\frac{\|w\|^2}{2}$ a $\frac{\|w\|^2}{2} + C(\sum_i \xi_i)^k$, donde C es un parámetro que se elegido por el usuario, una C más grande se corresponde a la asignación de una penalización superior a los errores. Tal como está, este es un problema de programación convexa para cualquier entero positivo k, para $k = 2$ y $k = 1$ es también un problema de programación cuadrática, y la elección de $k = 1$ tiene la ventaja adicional de que ni el ξ_i, ni sus multiplicadores de Lagrange , aparecen en el problema dual de Wolfe, que se convierte en

Maximizar:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \quad (2.32)$$

sujeto a

$$0 \leq \alpha_i \leq C, \quad (2.33)$$

$$\sum_i \alpha_i y_i = 0 \quad (2.34)$$

La solución se da de nuevo por:

$$w = \sum_{i=1}^{N_s} \alpha_i y_i x_i \quad (2.35)$$

Donde N_S es el número de vectores soporte.

Así, la única diferencia con el caso hiperplano óptimo es que el α_i ahora tienen un límite superior, C. En la figura 2.13 resume lo comentado. Ahora necesitamos las condiciones de Karush-Kuhn-Tucker (KKT) para el problema principal. El problema principal de Lagrange es:

$$L_P = \frac{1}{2}\|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i \{y_i(x_i \cdot w + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i \quad (2.36)$$

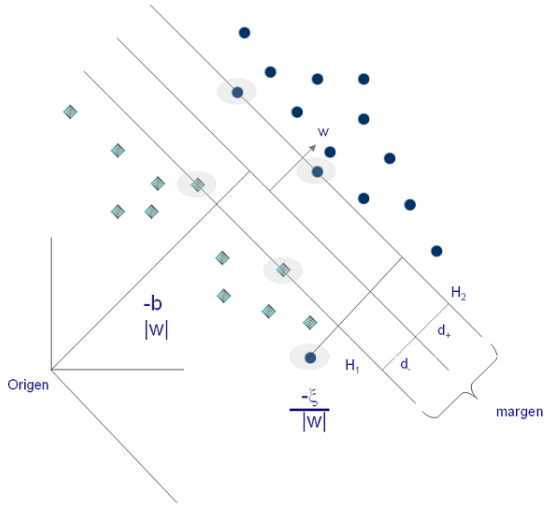


Figura 2.13: Hiperplanos separables linealmente para el caso de muestras no separables

donde los μ_i son los multiplicadores de Lagrange introducido para forzar que ξ_i sean positivos. Las condiciones de KKT para el problema principal es por lo tanto:

$$\frac{\partial L_P}{\partial w_v} = w_v - \sum_i \alpha_i y_i x_{iv} = 0 \tag{2.37}$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i = 0 \tag{2.38}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \tag{2.39}$$

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \tag{2.40}$$

$$\xi_i \geq 0 \tag{2.41}$$

$$\alpha_i \geq 0 \tag{2.42}$$

$$\mu_i \geq 0 \tag{2.43}$$

$$\alpha_i\{y_i(x_i \cdot w + b) - 1 + \xi_i\} = 0 \tag{2.44}$$

$$\mu_i \xi_i = 0 \tag{2.45}$$

como antes, podemos usar condiciones KKT complementáreas, ecuaciones (2.44) y (2.45) para determinar el umbral b. Si combinamos las ecuaciones (2.39) con la ecuación (2.45) hacen que $\xi_i = 0$ si $\alpha_i < C$, así podemos coger cualquier punto para entrenar para el cual, $0 < \alpha_i < C$ y usando la ecuación (2.44) con $(\xi_i = 0)$ para calcular b. Como antes es recomendable tomar la medida de tales puntos de entrenamiento.

2.4.2. Máquina vector soporte - no lineal

¿Cómo pueden los métodos anteriores generalizarse al caso en que el función decisión sea no lineal? [BGV92a], demuestran que un truco bastante antiguo [ABR64] que se puede utilizar para lograr esto de una manera sencilla.

Primero, ver que la única forma en que los datos aparecen en el problema de la ecuación (2.32) a la ecuación (2.34), es en forma de productos '$\cdot$' , $x_i \cdot x_j$. Ahora supongamos que primero se asigna los datos a otro espacio eucídeo H, mediante:

$$\Phi : R^d \longmapsto H \tag{2.46}$$

El algoritmo de entrenamiento sólo depende de los datos a través del producto en H, es decir, en funciones de la forma: $\Phi(x_i) \cdot \Phi(x_j)$, ahora bien, si hay una función *Kernel*, tal que: $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ sólo deberemos utilizar la función K en los algoritmos de entrenamiento y no será necesario utilizar de forma explicita Φ. La pregunta ahora es Pero ¿cómo podemos usar esta máquina?. Después de todo, tenemos que w , y los que quieren vivir en H también (ver ecuación (2.35)). Sin embargo, en fase de prueba una SVM es utilizado para calcular el producto '$\cdot$' de un x dado con w , o más específicamente mediante el cálculo del signo de:

$$f(x) = \sum_{i=1}^{N_s} \alpha_i y_i \Phi(s_i) \cdot \Phi(x) + b = \sum_{i=1}^{N_s} \alpha_i y_i K(s_i, x) + b \tag{2.47}$$

donde los s_i son los vectores soporte. De esta forma podemos evitar tener que calcular $\Phi(x)$ de forma explícita y usamos $K(s_i, x) = \Phi(s_i) \cdot \Phi(x)$. Vamos a llamar el espacio de los datos L (aquí y más adelante se utiliza L como una combinación de 'baja dimensión', y H para 'dimensión': por lo general es el caso de que el rango de Φ es de mayor dimensión que su dominio). Hay que tener en cuenta que, además del echo de que w se encuentra en H, no habrá, en general, en el vector L el mapeo Φ , a w. Si lo hubiera, $f(x)$ en la ecuación (2.47) puede ser calculada en un solo paso, evitando la suma (y haciendo las veces SVM correspondiente N_S más rápido). A pesar de esto, la idea que se ha indicado se puede utilizar para acelerar significativamente la fase de prueba de SVM [Bur96]. Hay que tener también en cuenta que es fácil de encontrar un *Kernel* (por ejemplo, las funciones *Kernel* del tipo $x_i \cdot$ Ł) tal que el algoritmo de entrenamiento y la solución encontrada es independiente de la dimensión de L y H. En la sección siguiente veremos qué funciones K son permisibles y cuáles no.

En la literatura sobre Máquinas Vector Soporte, por lo general, se refiere a H como el espacio Hilbert. Un espacio de Hilbert puede ser visto como una generalización del espacio Euclídeo. En concreto, es un espacio lineal, con un producto interno definido y completo con respecto a la norma y definido por el producto interno. Algunos autores requieren que el espacio de Hilbert sea separable (es decir, debe tener un subconjunto contable cuyo cierre es el espacio en sí mismo), y algunos (por ejemplo, [Hal67]) no lo hacen.

Condición de Mercer

¿Para qué *Kernel* existe un par: $\{H, \Phi\}$, con las propiedades descritas anteriormente y para cuáles no?. La respuesta a esta pregunta nos la da las condiciones de Mercer. Existe un mapeo y una expansión:

$$K(x, y) = \sum_i \Phi(x)_i \Phi(y)_i \tag{2.48}$$

Si y solo si, para cualquier $g(x)$ se cumple que:

$$\int (g(x)^2 dx \text{ es finita} \tag{2.49}$$

entonces

$$\int K(x,y)g(x)g(y)dxdy \geq 0 \tag{2.50}$$

No es fácil probar que la condición de Mercer's, se debe cumplir para la ecuación (2.50) para g con norma finita L_2 (debe satisfacer la ecuación (2.49)). Es fácil probar que se cumple la condición para potencias enteras positivas del producto $K(x,y) = (x \cdot y)^p$. Luego hay que demostrar que:

$$\int (\sum_{i=1}^{d} x_i y_i)^p g(x)g(y)dxdy \geq 0 \tag{2.51}$$

La expansión del termino $(\sum_{i=1}^{d})x_i y_i)^p$ contribuye de la forma:

$$\frac{p!}{r_1!r_2!\cdots(p-r_1-r_2\cdots)!}\int x_1^{r_1}x_2^{r_2}\cdots y_1^{r_1}y_2^{r_2}\cdots g(x)g(y)dxdy \tag{2.52}$$

el lado izquierdo de la ecuación (2.50) se puede factorizar:

$$\frac{p!}{r_1!r_2!\cdots(p-r_1-r_2\cdots)!}(\int x_1^{r_1}x_2^{r_2}\cdots g(x)dx)^2 \geq 0 \tag{2.53}$$

Una consecuencia de todo esto es que cualquier *Kernel* que puede ser expresado de la forma $K(x,y) = \sum_{p=0}^{\infty} c_p(x \cdot y)^p$ donde c_p son coeficientes reales positivos y la serie converge uniformemente y satisface la condición de Mercer

Por último, ¿qué ocurre si se utiliza un núcleo que no satisface la condición de Mercer? En general, pueden existir datos de tal manera que el grupo de acción es de carácter indefinido, y para el que el problema de programación cuadrática no tendrá solución (la función de doble objetivo puede llegar a ser muy grandes). Sin embargo, incluso para los núcleos que no satisfacen la condición de Mercer, se podría encontrar un conjunto de entrenamiento de una manera positiva y semidefinida de Hessiano, en cuyo caso el entrenamiento converge perfectamente. En este caso, sin embargo, la interpretación geométrica descrita anteriormente es insuficiente.

La condición de Mercer nos dice que si el *Kernel* cumple con la propiedad de un producto escalar en algún espacio, se puede usar para SVM aunque no nos dice cómo construir Φ, o incluso lo que H es. Utilizando un polinomio de segundo grado de forma explícita se puede construir el mapeo de algunos *Kernel*.

Por lo general, en el mapeo de los datos en un 'espacio de características' de un enorme número de dimensiones no sería un buen presagio para un SVM eficaz. Después de todo, el conjunto de todos los hiperplanos w, b son parametrizados por dim (H) + 1. La mayoría de los sistemas de reconocimiento de patrones con miles de millones, o incluso un número infinito de parámetros no lo hace más allá de la puerta de salida. ¿Cómo es que SVM puede hacerlo así? Uno podría argumentar que, dada la forma de solución, existen en la mayoría de $l + 1$ parámetros ajustables (donde l es el número de muestras de entrenamiento), pero esto parece ser una declaración de principios. Tiene que ver con la exigencia de que el margen de los hiperplanos sea máximo.

Puesto que la superficie asignada es de dimensión intrínseca dim (L), a menos que dim (L) = dim (H), es obvio que la proyección no puede ser (sobreyectiva). También no tiene que ser uno a uno (biyectiva), considerar $x1, \rightarrow -x1, x2 \rightarrow -x2$. La imagen de Φ no necesita ser un espacio vectorial:

Ejemplos de SVM no lineales

Los primeros *Kernel* investigados, para el problema de reconocimiento de patrones, fueron los siguientes:

$$K(x, y) = (x \cdot y + 1)^p \tag{2.54}$$

$$K(x, y) = e^{-\|x-y\|^2/2\sigma^2} \tag{2.55}$$

$$K(x, y) = tanh(kx \cdot y - \delta) \tag{2.56}$$

La ecuación (2.54) da como resultado un clasificador que es un polinomio de grado p, la ecuación (2.55) da un clasificador gaussiano y la función de base radial, y la ecuación (2.56) da una clase particular de sigmoidal de dos capas de redes neuronales. Para el caso de RBF, el número de centros (N_s en la ecuación (2.47)), Los propios centros (el s_i), los pesos (α_i), y el umbral (b) son producidos de forma automática en el entrenamiento de SVM y da un excelente resultados en comparación con RBFs clásica, para el caso de Gauss RBF's). Para el caso de redes neuronales, la primera capa se compone de conjuntos de N_s pesos, cada conjunto formado por d_L pesos (la dimensión de los datos), y la segunda capa se compone de N_s pesos (el α_i), por lo que la evaluación sólo requiere tomar una suma ponderada de sigmoides, se evaluaron en los productos '$\cdot$' de los datos de prueba con los vectores de soporte. Así, para el caso de redes neuronales, la arquitectura (número de pesos) se determina por el entrenamiento de SVM. Sin embargo, el *Kernel* de tangente hiperbólica sólo satisface la condición de Mercer para ciertos valores de los parámetros k y δ (y de los datos $|x|^2$). Esto fue señalado por primera vez de forma experimental por [Vap95], sin embargo algunas condiciones necesarias en estos parámetros son ahora conocidos.

La figura 2.15 muestra los resultados para el problema de reconocimiento de patrones, los mismos que se muestra en la figura 2.14, pero el *Kernel* que ha sido elegido es un polinomio de tercer grado. Hay que considerar que, a pesar de que el número de grados de libertad es mayor, para el caso linealmente separable (izquierda), la solución es aproximadamente lineal, lo que indica que la capacidad está siendo controlada, y que la forma lineal y el caso no separable (figura derecha) se ha convertido en separable.

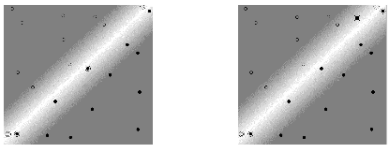


Figura 2.14: Ejemplo Lineal: Izquierda caso separable y derecha caso no separable.

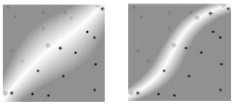


Figura 2.15: *Kernel* de un polinomio grado 3. El color de fondo muestra la forma de la superficie de decisión

Por último, señalar que a pesar de que los clasificadores SVM descritos anteriormente son clasificadores binarios, que son fáciles de combinar para tratar el caso multiclase. Para los ejemplos simples es eficaz con combinación del tipo N uno-versus-resto clasificadores (por ejemplo, 'un' positivo 'el resto' negativo) para el caso de la $N - clases$ toma la clase de un punto de prueba para calcular la correspondiente a la mayor distancia positiva [BGV92b].

2.4.3. Método y solución

El cálculo del vector soporte es básicamente un problema de optimización que se pueden resolver analíticamente sólo cuando el número de muestras de entrenamiento es muy pequeño, o para el caso de que los datos de entrenamiento sean separables y cuando se conocen a priori qué datos de entrenamiento se convierten en vectores de soporte. Pero para el caso general la complejidad computacional, en el peor de los casos, es del orden de N_S^3 , donde N_S es el número de vectores de soporte.

Sin embargo, en la mayoría de casos reales, las ecuaciones (2.32) (con productos de puntos de entrenamiento reemplazada por una función *Kernel*), ecuaciones (2.33) y (2.34) se deben resolver numéricamente. Para pequeños problemas, cualquier aplicación de propósito general de optimización puede resolverlo siempre que sea lineales y cumplan la restricción de que sean formas cuadráticas convexa. Para problemas de mayor envergadura, hay disponibles una amplia variedad de técnicas que se utilizar. Nos centramos en el problema generales, y para ser concretos, se dará una breve explicación de la técnica en concreto . Descendiendo por una 'cara', con un conjunto de puntos que se encuentran en el límite de la región factible, y una 'restricción activa' que es un obstáculo para que la igualdad se mantenga.

El procedimiento básico es:

1. Se debe de verificar la condición de optimalidad (KKT) para que la solución sea factible

2. Se define una estrategia de acercamiento al óptimo de manera uniforme el aumento en objetivo dual y sujeto a las restricciones.

3. Decidir sobre una descomposición del algoritmo de modo que sólo las porciones de los datos de entrenamiento debe ser asumidas en un momento dado

Se puede ver el problema como la solución de una secuencia de restricciones y de igualdades limitadas. Un problema de igualdades dada y limitado se puede resolver en un solo paso mediante el método de Newton, o en l pasos mediante el método del gradiente conjugado [PFV92] (l es el número de datos para el problema). Algunos algoritmos se mueven en una dirección dada hasta se encuentre una nueva restricción, en cuyo caso el algoritmo se reinicia con la nueva restricción de la lista de restricciones de igualdad. Este método tiene la desventaja de que sólo se tiene en cuenta una restricción a la vez. 'Los métodos de proyección' también se han considerado [MeT91], donde un punto fuera de la región factible se calcula, a continuación, se busca en línea y las proyecciones se llevan a cabo para que el movimiento actual se mantenga dentro de la región factible. Este enfoque puede agregar varias nuevas restricciones a la vez. En ambos casos, varias restricciones activas pueden llegar a desactivar en un solo paso. En todos los algoritmos, sólo la parte principal del Hessiano (las columnas correspondientes a $\alpha_i \neq 0$) tiene que ser calculados. Para el enfoque de Newton, también se puede aprovechar el echo de que la Hessiano es semidefinido positivo para la diagonalización con el algoritmo de Bunch-Kaufman ([BK77]; [BK80]) (si el grupo de acción fuera indefinido, se podría reducir fácilmente formando un bloque diagonal 2x2 con este algoritmo). En este algoritmo, cuando una nueva restricción se activa o inactiva, la factorización de la proyección del Hessiano es fácil de actualizar (en lugar de volver a calcular la factorización a partir de cero). Finalmente, en los métodos de punto interior, las variables son re-escaladas a fin de que permanezca siempre dentro de la región factible. Un ejemplo es el algoritmo 'LOQO' algoritmo de ([Van94b]; [Van94a]). Este último método es probable que sea útil para problemas en los que se espera que el número de vectores soporte, como parte del tamaño de la muestra de entrenamiento, sea grande.

Se describe someramente un método de optimización. Se trata de un método que utiliza la combinación de la pendiente y el ascenso de gradiente conjugado. Siempre se calcula la función objetivo con un costo computacional adicional pequeño. En la fase 1, búsqueda de direcciones **s** lo largo del gradiente. Se encuentra la cara más próxima a lo largo de la dirección de búsqueda. Si el producto escalar del gradiente con s es máximo indica que se encuentra a lo largo de **s**, entre el punto actual y la cara más próxima, entonces el punto óptimo se calcula analíticamente a lo largo de la dirección de búsqueda (esto no requiere una línea de búsqueda), y entramos en la fase 2, sino, se salta a una nueva cara y repetimos la fase 1. En la fase 2, Polak-Ribiere se asciende por el gradiente conjugado [PFV92], hasta que una nueva cara se encuentra (en cuyo caso volvemos a la fase 1) o se cumple el criterio de parada. Se tiene que tener en cuenta lo siguiente:

- Las búsquedas de direcciones siempre son proyectados de manera que las α_i continúan satisfaciendo la ecuación de restricción de igualdad (2.34). El algoritmo de gradiente conjugado seguirá funcionando, simplemente estamos buscando en un subespacio. Sin embargo, es importante que esta proyección se lleva a cabo de tal manera que no sólo la ecuación (2.34) se cumple (fácil de calcular), sino también para que el ángulo entre la dirección de búsqueda que aparece, y la dirección de búsqueda antes de la proyección, se reduce al mínimo (no tan fácil de calcular).

- Además, usamos el algoritmo 'sticky faces': cuando a una cara dada se accede más de una vez, las direcciones de búsqueda se ajustan de manera que todas las búsquedas posteriores se llevan a cabo por las caras que se enfrentan. Todas las 'sticky faces' se restauran cuando la tasa de incremento de la función objetivo está por debajo de un umbral.

- El algoritmo se detiene cuando la tasa de aumento de la función objetivo F cae por debajo de la tolerancia (por lo general $1e^{-10}$). También se pueden utilizar como criterio de parada la condición

de que el tamaño de la dirección de búsqueda proyectada caiga por debajo de un cierto umbral. Sin embargo, este criterio no trata bien el problema de escaldo.

- En opinión de [Bur98] lo más difícil es el manejo de la 'precisión' correcta en todas partes. Si esto no se hace el algoritmo o no converge o lo hace de forma muy lenta.

Una buena manera de comprobar que el algoritmo está funcionando es comprobar que la solución satisface todas las condiciones Karush-Kuhn-Tucker para el problema principal, ya que estas son condiciones necesarias y suficientes para que la solución sea óptima. Las condiciones KKT son de la ecuación (2.37) a la (2.45), con los productos 'punto-escalar' entre vectores de datos reemplazados por *Kernel* de donde sea que aparezcan (w debe ser ampliado, como en la ecuación (2.37). En primer lugar, puesto que w no es, en general, la asignación de un punto en L). Así, para comprobar las condiciones de KKT, es suficiente comprobar que los α_i satisfacer $0 \leq \alpha_i \leq C$, que la restricción de igualdad (2.38) sostiene que todos los puntos que $0 \leq \alpha_i < C$ satisfacen la ecuación (2.40) con $\xi_i = 0$, y que todos los puntos con $\alpha_i = C$ satisfacen la ecuación (2.40) para algunos $xi_i \geq 0$. Estas son condiciones suficientes para que todas las condiciones KKT se cumplan.

2.4.4. Limitaciones, extensiones y conclusiones de la máquina vector soporte

Limitaciones

Tal vez la mayor limitación del enfoque de vectores de soporte se encuentra en la elección del *Kernel*. Una vez que el núcleo es fijo, los clasificadores SVM sólo tiene un parámetro elegido por el usuario (la penalización del error), pero el *Kernel* es una alfombra muy grandes en las que barrer parámetros. Algunos trabajos se han realizado en los *Kernel* limitando con el conocimiento previo ([SSSV98]), pero la mejor opción del *Kernel* para un determinado problema sigue siendo un tema de investigación.

Una segunda limitación es la velocidad y el tamaño, tanto para el entrenamiento como para las pruebas. Mientras que el problema de la velocidad en fase de prueba está en gran medida resuelto en [Bur96], esto aún requiere dos pases para la fase de entrenamiento. Para grupos de datos muy grandes (millones de vectores de soporte) es un problema que todavía no resuelto.

Los datos discretos presenta otro problema, aunque se han obtenido resultados excelentes con adecuado redimensionamiento ([Joa97].

Por último, aunque algunos han trabajado en la formación de una SVM multiclase en un solo paso, el diseño óptimo para los clasificadores SVM multiclase es un área de investigación futura.

Extensiones

Veamos de forma resumida dos métodos sencillos y eficaces para mejorar el rendimiento de SVM:

- El método de vectores soporte virtual ([SBV96];[BS97]), trata de incorporar los 'invariantes' conocidos del problema (por ejemplo, la invarianza de traducción para el problema de reconocimiento de imágenes) primero se capacita el sistema, y luego se crean nuevos datos a través de vectores que distorsionan el apoyo resultante (su traducción, en el caso mencionado), y, finalmente, se entrena al nuevo sistema con los datos distorsionados (y sin distorsionar). La idea es fácil de implementar

y parece funcionar mejor que otros métodos para la incorporación de las invariantes propuestas hasta el momento.

- El método que se deduce de ([Bur96]; [BS97]) se introdujo para hacer frente a la velocidad de las máquinas de vectores soporte en fase de prueba, y también se inicia con una SVM entrenado. La idea es reemplazar la suma de la ecuación (2.35) por una suma similar, donde en lugar de vectores de soporte, los vectores calculados (que no son elementos del conjunto de entrenamiento) se utilizan, y en lugar de la α_i, un conjunto diferente de los pesos se calculan. El número de parámetros se elige de antemano para dar la aceleración deseada. El vector resultante sigue siendo un vector H, y los parámetros se encuentran minimizando la norma euclidiana de la diferencia entre el vector original w y la aproximación a ella. La misma técnica podría ser utilizada para la regresión SVM para encontrar una representación de la función mucho más eficiente (que podría ser utilizado, por ejemplo, en la compresión de datos).

La combinación de estos dos métodos dieron un factor de aceleración de 50 (mientras que la tasa de error de un aumento del $1,0$ al $1,1$ por ciento) en los dígitos del NIST.

Conclusiones

La Máquina Vector Soporte ofrece un nuevo enfoque al problema de reconocimiento de patrones (junto con la estimación de regresión lineal y la inversión de operador), con claras conexiones con la teoría del aprendizaje estadístico subyacente. Difiere radicalmente de los enfoques comparables, tales como las redes neuronales: el entrenamiento de un SVM siempre encuentra un mínimo global, y su geometría simple de interpretación proporciona un terreno fértil para futuras investigaciones. Un SVM en gran parte se caracteriza por la elección del *Kernel* y por lo tanto SVM está relacionado con los problemas que están diseñados sobre los métodos basados en el uso de *Kernel*.

2.5. Algoritmo de clasificación: K-NN

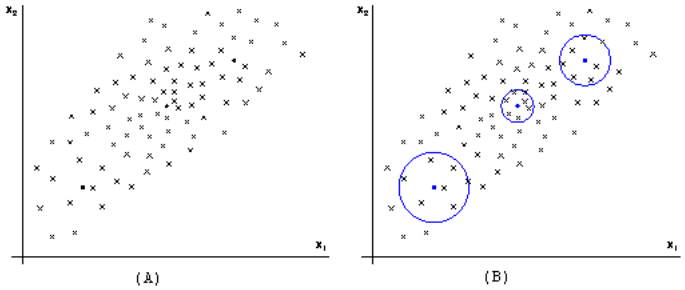
En el método K-NN [Fix51] es un método de clasificación supervisada (aprendizaje, estimación basada en un conjunto de entrenamiento y prototipos) que sirve para estimar la función de densidad $F(x/C_j)$ de las predictoras x por cada clase C_j.

K-NN es un método de clasificación no paramétrico, que estima el valor de la función de densidad de probabilidad o directamente la probabilidad a posteriori de que un elemento x pertenezca a la clase C_j a partir de la información proporcionada por el conjunto de prototipos. En el proceso de aprendizaje no se hace ninguna suposición acerca de la distribución de las variables predictoras.

En el reconocimiento de patrones, el algoritmo K-NN es usado como método de clasificación de objetos (elementos) basado en un entrenamiento mediante ejemplos cercanos en el espacio de los elementos. K-NN es un tipo de "Lazy Learning", donde la función se aproxima sólo localmente y todo el cómputo es diferido a la clasificación.

2.5.1. Estimación mediante los K-NN

Supongamos, por simplicidad, un espacio de representación bidimensional y una serie de muestras de una misma clase representadas en la figura 2.16.(A)]. Dado un patrón cualquiera X, si consideramos las K muestras más próximas a X, éstas estarán localizadas en un círculo centrado en X. En la figura



[h!]

Figura 2.16: Patrones en un espacio bidimensional (A). Los patrones considerados para la estimación de $P(X|\omega)$ mediante los 7 vecinos más cercanos. (B)

2.16.(B) resaltamos los 7 vecinos más cercanos a tres patrones.

Parece sensato pensar que el área del círculo que encierra un número fijo de puntos, k, es menor en regiones densamente pobladas que en regiones donde los puntos están más dispersos. Este sencillo planteamiento es la base de la estimación mediante los k vecinos más próximos. En espacios multidimensionales, el círculo se convierte en una hiperesfera, y el planteamiento anterior se puede extender fácilmente ya que el volumen de la hiperesfera que encierra a k puntos está relacionado con el valor de la función de densidad de probabilidad en el centro de la hiperesfera. Veamos de qué manera.

Sea $K_i(X), i = 1, 2, \cdots, J$ es el número de prototipos de la clase ω_i que se encuentran en una vecindad de X, el valor de $\hat{p}(X|\omega_i)$ puede calcularse como la proporción de los N_i prototipos de la clase ω_i que se encuentran en esa vecindad:

$$\widehat{P}(X|\omega_i) = \frac{K_i(X)}{N_i} \tag{2.57}$$

La vecindad depende del patrón considerado para la estimación:

- Si el patrón se encuentra en una región muy poblada, la vecindad a considerar tendrá un radio menor.
- Si el patrón se encuentra en una región poco poblada, la vecindad a considerar tendrá un radio mayor.

Con esta idea, la ecuación (2.57) debe modificarse de manera que se pondere inversamente con el área de la región considerada. Para el caso d-dimensional, el área se convierte en el volumen de una hiperesfera centrada en X, $V(X)$, por lo que el nuevo estimador se formula como:

$$\hat{P}(X|\omega_i) = \frac{K_i(X)}{N_i V(X)} \tag{2.58}$$

Para clarificar más acerca de la hiperesfera es necesario especificar que es la que está centrada en X y contiene los k vecinos más cercanos a X. En el caso bidimensional como aparece en la figura 2.16, la esfera se convierte en un círculo y el volumen se convierte en superficie.

2.5.2. Elección del valor adecuado de K

El problema que se plantea ahora es el de si existe un valor óptimo para k o en su defecto si se pueden dar algunas reglas para fijar este valor. La elección de k está determinado por la densidad de los puntos y debería hacerse en relación a N_i. Puede demostrarse que el estimador dado en (2.58) es no sesgado y consistente si se verifican las siguientes condiciones sobre k:

$$\lim_{N_i \to \infty} kN_i = \infty \tag{2.59}$$

$$\lim_{N_i \to \infty} \frac{KN_i}{N_i} = 0 \tag{2.60}$$

Así, una elección adecuada de $k(N_i)$ es:

$$K(N_i) = cte\sqrt{N_i} \tag{2.61}$$

2.5.3. Clasificador del vecino más próximo

Las probabilidades a priori pueden estimarse por la frecuencia relativa global

$$\hat{\pi_i} = \frac{N_i}{N} \tag{2.62}$$

y considerando que el estimador de la densidad de probabilidad $\hat{P}(X|\omega_i)$ viene dado por la ecuación (2.58) un estimador de la probabilidad a posteriori será:

$$\widehat{P}(\omega_i|X) = \hat{P}(X|\omega_i)\widehat{\pi}_i = \frac{K_i(N)}{N_i v(X)}\frac{N_i}{N} = \frac{K_i(X)}{v(X)N} = \frac{K_i(X)}{k} \tag{2.63}$$

A partir de este resultado se formula la siguiente regla de clasificación:

$$\text{Seleccionar } \omega_c \text{ si } K_c(X) = max\{K_i(X)\} \text{ para } i = 1, 2, \cdots, J \tag{2.64}$$

conocida como regla de clasificación por los k vecinos más cercanos o simplemente K-NN . Cuando k = 1, la regla anterior se conoce como la conocida como regla de clasificación del vecino más cercano o simplemente 1 vecino más próximo (de sus siglas en inglés (1-NN). Con otras palabras, podemos afirmar que los prototipos cercanos tienden a ser de la misma clase 1-NN o bien a tener una probabilidad a posteriori similar K-NN. Estas reglas proporcionan una estimación directa de la probabilidad a posteriori de cada una de las clases y las reglas de clasificación son sencillas y fácilmente interpretables.

2.5.4. Regla de clasificación 1-NN

La regla de clasificación por vecindad más simple es la regla de clasificación del vecino más cercano o simplemente 1-NN. Se basa en la suposición de que la clase de la muestra a etiquetar: X, es la del prototipo más cercano en R, al que notaremos por X_{NN}. Si $|R| = N$ esta regla puede expresarse como:

$$d(X) = \omega_c \text{ si } \begin{cases} \delta(X, X_{NN}) & = min\{\delta(X, X_i)\} \text{ para } i = 1, 2, \cdots, N \\ (X_{NN}) & \epsilon R \end{cases} \tag{2.65}$$

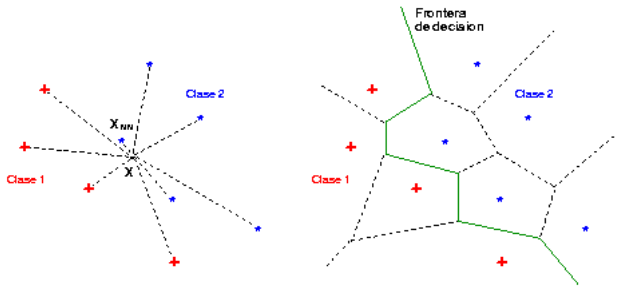


Figura 2.17: Ejemplos de Clasificación 1-NN

En la figura 2.17 en la parte izquierda se muestra cómo se clasificaría el patrón X con la regla $1-NN$ para un problema de clasificación de dos clases. Existen cuatro prototipos de clase 1 (representados por cruces) y cinco prototipos de clase dos (representados por asteriscos). El prototipo más cercano es de la clase '2', por lo que ésta será la clase asociada a X.

El efecto de esta regla es el de dividir el espacio de representación en N-'regiones de influencia', una por cada prototipo. Cada una de esas regiones tiene forma poligonal y los bordes corresponden a los puntos situados a igual distancia entre prototipos. Cada una de estas regiones se conoce como región de Voronoi y la partición poligonal del espacio de representación se conoce como partición de Voronoi. En la figura 2.17, en la parte derecha aparece la partición de Voronoi asociada. Cada región de Voronoi puede considerarse como una región de decisión (restringida al prototipo central), por lo que la región de decisión de una clase será la unión de todas las regiones de Voronoi de los prototipos de esa clase. La consecuencia es que las fronteras de decisión serán fronteras lineales a trozos.

2.5.5. Regla de clasificación K-NN

La regla de clasificación por vecindad más general es la regla de clasificación de los k vecinos más cercanos o simplemente K-NN. Se basa en la suposición de que los prototipos más cercanos tienen una probabilidad a posteriori similar.

Si $K_i(X)$ es el número de muestras de la clase ω_i presentes en los k vecinos más próximos a X, esta regla puede expresarse como:

$$d(X) = \omega_c \text{ si } K_c(X) = max\{K_i(X)\} \text{ para } i = 1, 2, \cdots, J \tag{2.66}$$

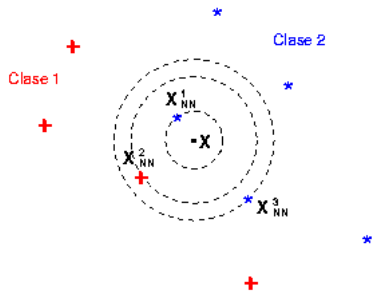


Figura 2.18: Ejemplo de clasificación K-NN, para $k = 3$

En la figura 2.18 se muestra cómo se realizaría la clasificación $3 - NN$ del mismo patrón que se utilizó como ejemplo de clasificación 1-NN en la figura 2.17. En este caso, $K_1(X) = 1$ y $K_2(X) = 2$ por lo que X se etiquetará como de clase 2.

2.6. Otras herramientas software utilizadas

Durante el desarrollo de este trabajo se han utilizado dos herramientas software que han permitido centrarnos en los retos principales del trabajo descargando sobre las mismas determinadas tareas y reduciendo el tiempo total invertido. En esta sección del trabajo describiremos de manera somera en qué consisten.

2.6.1. WEKA

La experiencia demuestra que ningún sistema de aprendizaje automático es apropiado para todos los problemas. El aprendizaje universal es, desde luego, una utopía. El framework WEKA [WFH10] es una colección de técnicas de aprendizaje automático y herramientas de pre-procesamiento de datos que incluye muchos de los algoritmos de clasificación. Está diseñado de manera que permite probar los métodos existentes en nuevos conjuntos de datos de manera flexible. Existe un amplio soporte para todo el proceso de aprendizaje automático incluyendo:

- La preparación de los datos de entrada.
- Evaluación de los sistema de aprendizaje.
- Visualización de los datos de entrada.
- Visualización del resultado del proceso de aprendizaje.
- Una amplia gama de algoritmos de aprendizaje

A este conjunto de herramientas se accede a través de una interfaz común para que sus usuarios puedan comparar los distintos métodos e identificar aquellos que son más apropiados para el problema en cuestión.

WEKA se desarrolló en la Universidad de Waikato en Nueva Zelanda, y el nombre es sinónimo de Medio Ambiente Waikato para el análisis del conocimiento. Fuera de la universidad WEKA, es un ave

que no vuela con una naturaleza inquisitiva que habita exclusivamente en las islas de Nueva Zelanda.

El sistema está escrito en Java y distribuido bajo los términos de la Licencia GNU. Funciona en casi cualquier plataforma y ha sido probado bajo Linux, Windows y Mac. Proporciona una interfaz uniforme a diferentes algoritmos de aprendizaje, junto con los métodos de pre-y post-procesado, para evaluar el resultado de aprender con los algoritmos con cualquier conjunto de datos dado.

Figura 2.19: Pantalla inicial de WEKA, versión para windows

Qué nos encontramos en WEKA

El framework WEKA proporciona implementaciones de algoritmos de aprendizaje que se pueden aplicar fácilmente a cualquier conjunto de datos. También incluye una variedad de herramientas para la transformación de conjuntos de datos, tales como los algoritmos de discretización. Se puede pre-procesar un conjunto de datos, alimentar a un algoritmo de aprendizaje, y analizar el clasificador resultante y su rendimiento, sin necesidad de escribir una sola linea de código de programación.

El framework incluye los métodos para numerosos problemas de aprendizaje automático y minería de datos estándar: regresión, clasificación, clustering, la minería de reglas de asociación, y la selección de atributos. Conocer los datos es una parte importante en el proceso de aprendizaje por lo que se incluyen sistemas de visualización de datos y herramientas de pre-procesamiento de datos. El formato por excelencia utilizado por WEKA es ARFF que puede ser leído desde un archivo.

Con WEKA podemos:

1. Aplicar un método de aprendizaje de un conjunto de datos y analizar el resultado para obtener más información acerca de los datos.

2. A un modelo entrenado utilizarlo para generar predicciones sobre nuevas muestras de test.

3. Aplicar varios algoritmos de aprendizaje y compara su rendimiento con el objetivo de seleccionar el óptimo para realizar predicciones.

Los métodos de aprendizaje son los llamados clasificadores, y desde su interfaz de menú se puede seleccionar el que desee. Los clasificadores disponen de parámetros ajustables, a los que se accede de forma fácil a través de un formulario de propiedades o el editor de objetos. Un módulo común se utiliza para la evaluación del rendimiento de todos los clasificadores.

Las implementaciones de los sistemas de aprendizaje automático son sin duda el recurso más valioso que ofrece WEKA. Pero las herramientas para el pre-procesamiento de los datos, llamados filtros, vienen en segundo lugar. Al igual que los clasificadores, se selecciona un menú de filtros y se pueden adaptar a las necesidades.WEKA también incluyen:

- Algoritmos para el aprendizaje de reglas de asociación.
- Agrupación de datos para los que no se especifica un valor de una clase.
- Selección de los atributos relevantes de los datos

Cómo se utiliza

La manera más fácil de usar WEKA es a través de una interfaz gráfica de usuario llamado 'Explorer'. Este da acceso a todas sus recursos mediante la selección de menú y la cumplimentación de los formularios. El interface 'Explorer' le ayuda a hacer precisamente eso, nos guía por la presentación de opciones como los menús, forzando a trabajar en la secuencia adecuada entre las diferentes opciones hasta que las mismas son aplicables, y mediante la presentación de formularios para seleccionar los diferentes parámetros. Ayudas contextuales se visualizan en la pantalla cuando el ratón pasa por encima de los elementos para explicar lo que hacen. Valores razonables por defecto están pre-cargados lo que asegura que se pueden obtener resultado con un mínimo de esfuerzo.

Hay otras dos interfaces gráfica de usuario en WEKA

1. La interfaz de flujo de conocimiento: Permite diseñar las configuraciones para el procesamiento de datos de entrada. Una desventaja fundamental de la interface 'Explorer' es que tiene todo en la memoria principal cuando se abre un conjunto de datos, inmediatamente se lo carga todo adentro lo que significa que sólo se puede aplicar a los problemas de pequeño y mediano tamaño. Sin embargo, WEKA contiene algunos algoritmos incrementales que pueden ser utilizados para procesar grandes bases de datos. Este interfaz permite mediante la técnica de arrastrar 'cajas' que representan algoritmos de aprendizaje y fuentes de datos alrededor de la pantalla y unirlas como se desea. Permite especificar una secuencia de datos conectando los componentes que representan las fuentes de datos, herramientas de pre-procesamiento, algoritmos de aprendizaje, métodos de evaluación, y los módulos de visualización. Si los filtros y algoritmos de aprendizaje son capaces de aprender incremental, los datos se cargan y se procesan de forma incremental.

2. El experimentador: Esta interfaz está diseñado para ayudar a responder a una cuestión práctica fundamental en la aplicación de técnicas de clasificación y regresión: ¿cuáles son los métodos y valores de los parámetros mejores para el problema dado? Generalmente, no hay manera de responder a esta pregunta, a priori, y una de las razones de la existencia de este framework es para proporcionar un entorno que permite a los usuarios comparar una variedad de técnicas de aprendizaje. Esto se puede hacer de forma interactiva utilizando el 'Explorer', como se ha indicado. Sin embargo, el experimentador le permite automatizar el proceso por lo que es fácil de ejecutar los clasificadores y los filtros con parámetros diferentes en un mismos conjunto de datos, para recopilar estadísticas del rendimiento, y realizar las pruebas. Los usuarios avanzados pueden utilizar el experimentador para distribuir la carga computación a través de múltiples máquinas utilizando Java Remote Method Invocation (RMI). De esta forma se puede configurar experimentos a gran escala estadística y dejar que se ejecuten.

3. Modo línea de comando: A todas las funciones básicas de WEKA se puede acceder en modo consola mediante la introducción de comando de texto, que da acceso a todas las funciones del sistema.

Al ejecutar WEKA hay que elegir con que interface de usuarios se desea trabajar.

Qué más puedo hacer con WEKA

Un recurso importante cuando se trabaja con WEKA es la documentación en línea, que ha sido generada automáticamente a partir del código fuente y es concisa y refleja su estructura. Hay una única lista completa de los algoritmos disponibles porque WEKA está en continuo crecimiento y se genera automáticamente a partir del código fuente, la documentación en línea está siempre al día. Por otra parte, se hace imprescindible si se desea continuar al siguiente nivel y acceder a la biblioteca a partir de sus propios programas en Java o escribir y probar los sistemas de aprendizaje por su cuenta.

En muchas aplicaciones, el componente de aprendizaje automático es sólo una parte de un sistema software mucho más grande. Si la intención es escribir este tipo de aplicaciones, hay que acceder a los programas en WEKA desde nuestro propio código. De esta manera, se puede resolver el subproblema de aprendizaje automático con un mínimo de programación adicional.

En el caso de desarrollar algoritmos de aprendizaje propios sin tener que abordar tareas como la lectura de los datos de un archivo, la aplicación de filtrado de algoritmos, o proporcionar código para evaluar los resultados. WEKA ya incluye todo esto.

Donde puedo conseguirlo

WEKA está disponible en http://www.cs.waikato.ac.nz/ml/weka. Y se puede descargar e instalar o descargar un archivo java, extensión 'jar' para su posterior instalación siempre que tengamos instalada la máquina virtual de java. Además señalar que los autores del proyecto han editado un libro [WFH10] que explica en profundidad las diferentes técnicas disponibles en WEKA e incluyen ejemplos. Al libro se puede enlazar desde http://www.cs.waikato.ac.nz/ml/weka/book.html.

2.6.2. ENCARA2

ENCARA2 [CSDSHTGA07] es un detector de rostros en tiempo real desarrollado en el SIANI. Este detector es una evolución del detector ENCARA [MFCG03] que basaba la detección facial en la búsqueda de zonas de color de piel.

Los problemas encontrados en la detección del color de piel hacían inviable conseguir un detector robusto ante cualquier condición de luces, cámaras, etcétera y la posible presencia de fondos con colores similares por ello, en lugar de emplear un modelo genérico del color de piel, se utilizó otro esquema (Viola-Jones incluido desde 2003 en OpenCV) para detectar la primera cara frontal lo que nos permite modelar el color de piel de la persona en las condiciones actuales, llevando al equipo de investigación a la versión ENCARA2.

ENCARA2 realiza una primera detección basada en clasificadores de Viola-Jones de diferente finalidad (caras, cabeza-hombros). A partir de esta detección se realiza una confirmación detectando elementos faciales (ojos, nariz y boca). La confirmación permite modelar el rostro detectados en base a su posición,

tamaño, apariencia, etcétera. El modelo se reutiliza en imágenes sucesivas para acelerar, con mayor robustez, el proceso de detección.

La combinación de técnicas ha mejorado la tasa de detección correcta manteniendo la exigencia de tiempo real. ENCARA2 supone que se dispone en la Universidad de Las Palmas de Gran Canaria (ULPGC) de un módulo válido y probado para proporcionar datos, a otros, de análisis facial. Y ya se han abordado las primeras experiencias en descripción facial.

ENCARA2 sigue siendo un proyecto abierto con margen de mejora y sus lineas futuras de trabajo van en las siguientes líneas : mejorar la detección, integrar la descripción, ampliar los conjuntos de aprendizaje e integrar segmentación y seguimiento por color. La búsqueda de los elementos faciales da mayor fiabilidad a la detección de rostros, por lo que se puede plantearse buscar rostros en base a sus elementos.

ENCARA2 aporta a este proyecto

Esta herramienta desarrollada en C++ en entorno Windows nos proporciona las siguiente funcionalidades:

- La entrada a esta aplicación los constituyen los vídeos obtenidos.
- Detección de las caras en cada vídeo.
- Re-escaldado de cada imagen a una dimensión fija: 59x65 píxeles.
- Genera una imagen en blanco y negro de cada cara.
- Normaliza las imágenes.
- Salida del programa son las muestras de las caras que posteriormente aplicaremos técnicas para obtener el vector de características de las mimas (LBP/Mapa de píxeles/PCA) que nos permitan entrenar nuestros algoritmos de clasificación.

En definitiva nos proporciona las dos primeras fase del procedimiento de reconocimiento facial que se incluye en la figura 1.2

Capítulo 3

Experimentos Realizados

Este capítulo del Trabajo Fin de Máster es el núcleo experimental del mismo y en el abordaremos los siguientes hitos:

1. Se indicará el procedimiento de obtención de las imágenes necesarias para entrenar y probar el proceso de aprendizaje.

2. Se comentarán las funciones principales vinculadas al operador LBP.

3. Se incluirán nuevos conjuntos de datos que se utilizarán para realizar comparaciones entre los diferentes métodos obteniendo nuevos resultados.

4. Y por último se presentarán los datos obtenidos

Esta secuencia de pasos describe fielmente el proceso seguido para el desarrollo de los experimentos vinculados a este trabajo.

3.1. Proceso para obtención del conjunto de imágenes de los rostros

Al objeto de disponer de las muestras necesarias se ha realizado dos sesiones de grabaciones de vídeo con hasta nueve individuos. En cada grabación se capturan dos vídeos, el primero de ellos en formato estándar a color RGB y el segundo de ellos en escala de grises que refleja la profundidad. Para el caso de de los vídeos en formato de profundidad se utiliza como factor conversión de cambio de escala $\frac{255}{2047}$ [1]. En la primera sesión se obtiene diez vídeos uno para cada de las personas que han colaborado que además sus caras constituirán el conjunto de entrenamiento. En la segunda sesión de grabación se obtiene vídeos de cinco personas, las imágenes de las caras de esta sesión formaran parte del conjunto de test en el proceso de aprendizaje.

Finalmente se decide que el número de individuos de entrenamiento y de test coincidan eliminándose, del conjunto de aprendizaje, los rostros de las cinco personas que no tenían vídeo de test. En resumen, se dispone de cinco vídeos para generar las imágenes de entrenamiento y de test.

Una vez disponemos de todos los vídeos necesarios hemos de obtener las imágenes de las caras, para ello utilizamos la herramienta ENCARA2 detector facial que es capaz de generar las imágenes separada en niveles de gris para cada una de las caras detectadas en los vídeos en formato RGB. Para extraer la

[1] Esta información se obtiene de http://kgxpx834.blog58.fc2.com

caras en los vídeos en formato de profundidad nos basaremos en las detecciones realizadas en los vídeos en formato RGB, se utilizan las coordenadas de la posición en el frame para marcar la zona de la cara en formato de profundidad. De forma empírica se detectó un desplazamiento de la posición del rostro en la imagen de profundidad con respecto a la imagen RGB, esto se corrigió ampliando el rectángulo que contenía la cara de la persona.
El número de imágenes, por persona, para el conjunto de entrenamiento son:

Id	Caras
1	235
2	174
3	93
4	163
5	76
Total	741

y para el conjunto de test:

Id	Caras
1	60
2	144
3	81
4	120
5	114
Total	519

En resumen, el total de imágenes de caras en formato RGB 1.260 de las que 741 (58,81 %) son para entrenar y 519 (41,19 %) son de pruebas y, de las imágenes en formato de profundidad, se tienen un total de 1.260 de las que 741 (58,81 %) constituyen el conjunto de aprendizaje y 519 (41,19 %) se utilizarán para la verificación del sistema de reconocimiento facial, como se puede observar en la figura 3.1.
De forma gráfica en la figura 3.2 se observa la distribución de las muestras por cada uno de los individuos tanto para entrenamiento como para verificación, estos datos son válidos tanto para imágenes en formato RGB como para formato en profundidad.

Como hemos comentado estas imágenes se obtienen a partir de los vídeo obtenidos, en la figura 3.3 se observan imágenes del conjunto de vídeos obtenidos para el formato RGB para generar las imágenes del conjunto de aprendizaje.

El conjunto de vídeos obtenidos de la *Kinect* en formato de profundidad son los que se pueden observar en la figura 3.4 A continuación en las figuras 3.5 y 3.6 se pueden observar imágenes de los vídeos obtenidos para generar los conjuntos de test tanto en formato RGB como en escala de grises de profundidad.

De los vídeos y con el programa ENCARA2 se han extraído las imágenes individuales de las caras que forma parte de los conjuntos de aprendizaje y de test que se van a utilizar para entrenar a los clasificadores. Todas las imágenes obtenidas de los vídeos son de 59x65 píxeles. En las figuras 3.7, 3.8, 3.9 y 3.10 tenemos el resultado de aplicar ENCARA2 a los vídeos, tanto en formato RGB como en formato de profundidad y para los los conjuntos de aprendizaje y de test.

El proceso de extracción de las imágenes de las caras de los vídeos en formato de profundidad se ha realizado de forma manual y ha consistido en:

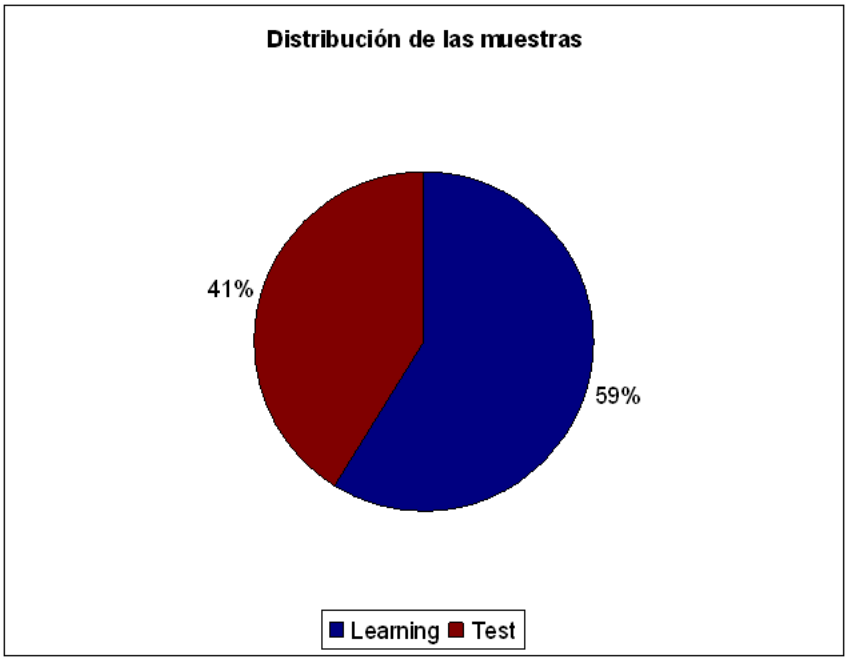


Figura 3.1: Distribución porcentual de las imágenes de muestras de entrenamiento y de test

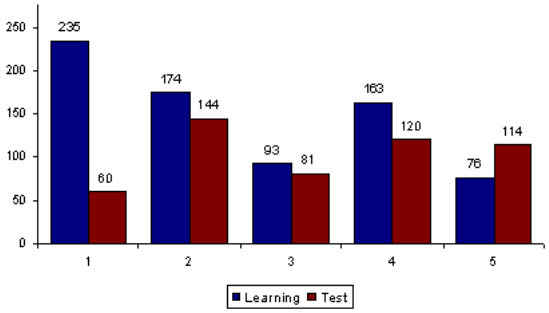


Figura 3.2: Distribución de las imágenes de las muestras tanto en formato RGB como en formato profundidad y para entrenamiento y test

- Se ha verificado que visualmente que había una sincronización entre el vídeo en formato RGB y el vídeo en formato de profundidad, que además ha sido constatado por la propia coincidencia de en la duración de los vídeos.
- Se ha modificado la aplicación para que una vez detectada una cara en un frame del vídeo en formato RGB extraer los vértices del rectángulo y extraer la imagen de la cara del correspondiente frame del vídeo en formato de profundidad.
- Al realizar esta extracción se detecta que las imágenes en los vídeos en formato profundidad son de mayor tamaño. Para resolver este problema a los vértices del rectángulo se les da cierta holgura para incluir en el proceso de detección todo el contorno del rostro de la persona.

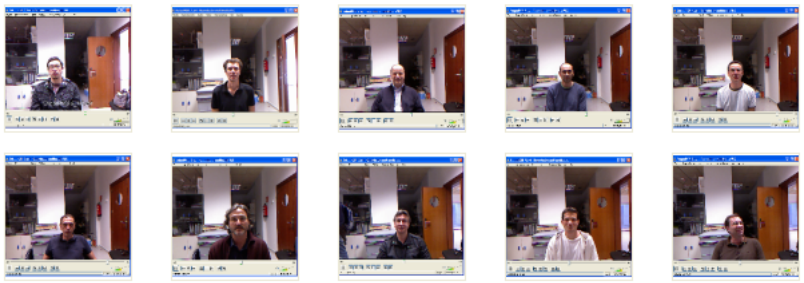


Figura 3.3: Muestras de imágenes de los vídeos del conjunto de aprendizaje en formato RGB

Figura 3.4: Muestras de imágenes de los vídeos del conjunto de aprendizaje en formato de profundidad.

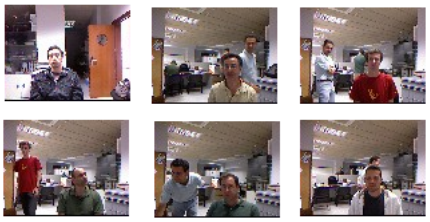


Figura 3.5: Ejemplos de vídeos obtenidos para obtener las imágenes de TEST en formato RGB

A estas imágenes se le aplicarán diferentes procesos con el fin de obtener resultados que nos permitan medir el rendimiento de los operadores LBP en comparación con otros caracterizadores que se utilizan en sistemas de reconocimiento facial, que es el objetivo de este trabajo.

3.2. Operadores LBP

En la página web de la Universidad de OULU[web], se han obtenido las siguientes funciones **MATLAB**, Abreviatura de MATrix LABoratory, 'laboratorio de matrices'. Es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio 'lenguaje M'. Disponible para las plataformas Unix, Windows y Apple Mac OS X, en las que nos basaremos para todo el desarrollo software del trabajo:

- lbp.m
- getmapping.m

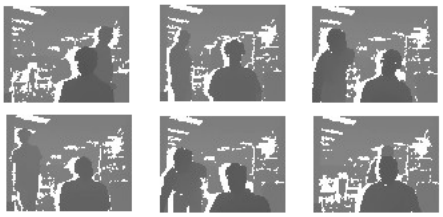


Figura 3.6: Ejemplos de imagen de los vídeos en formato profundidad para generar el conjunto de pruebas

Figura 3.7: Ejemplos de imágenes en formato RGB del conjunto aprendizaje generadas por ENCARA2

Figura 3.8: Ejemplos de imágenes en formato profundidad del conjunto de aprendizaje obtenidas por ENCARA2

Figura 3.9: Muestras de imágenes en formato RGB generadas por ENCARA2 de conjunto de test

Figura 3.10: Muestras de imágenes en formato de profundidad generadas por ENCARA2 de conjunto de test

- cont.m

que permiten generar los histogramas de las imágenes y de las nuevas imágenes en formato LBP. En los siguientes aparatados se detallan algunos de los aspectos relevantes de estas funciones que he usado de alguna forma para el desarrollo de los experimentos.

3.2.1. Función lbp.m

La función lbp que hemos utilizado es la versión 0.3.2 y de Marko Heikkilääa y Timo Ahonen de la Universidad Finlandesa de OULU.
La función lbp devuelve la imagen en formato LBP o el histograma de la imagen. El formato de llamada a esta función es:

$J = LBP(I, R, N, MAPPING, MODE)$, donde tenemos que las variables de entradas son:

I es la imagen a la que vamos aplicar el operador LBP.

Los códigos LBP se calculan utilizando 'N' puntos de muestreo en un circulo de radio 'R', por defecto se utiliza 8 puntos de muestreo alrededor del punto central de radio 1, en el caso de utilizar otros valores es necesario ejecutar previamente la función getmapping.m que analizaremos posteriormente. Los posibles valores de MODE son:

- 'h' o 'hist', para obtener el histograma de la imagen
- 'nh' , para obtener el histograma normalizado de la imagen

En caso contrario se devuelve una imagen en formato LBP

Esta función la utilizaremos para:

- Generar imágenes en formato LBP tanto en formato RGB como de profundidad para los conjuntos de entrenamiento y de TEST
- Para generar los histogramas normalizados de las imágenes
- En conjunción con la función getmapping para generar nuevos histogramas diferentes del estándar

En el Anexo I se incluye el texto integro de la versión 0.3.2 de esta función obtenido de la url indicada.

3.2.2. Función getmapping.m

Los autores de la función son: Marko Heikkilä y Timo Ahonen y la versión que proporcionan es la 0.1.1.

Esta función devuelve una estructura que contiene la tabla de asignación de códigos LBP, la forma de la llamada es:

$MAPPING = GETMAPPING(SAMPLES, MAPPINGTYPE)$

Mediante la variable SAMPLES indicamos el número de puntos vecinos, que luego utilizaremos en la función lbp para el parámetro P.
Los valores permitidos para la variable MAPPINGTYPE , que definen los diferentes características del algoritmo LBP son:

- u2 genera código LBP uniforme.
- ri genera código LBP invariante a rotaciones.
- riu2 geneta código LBP uniforme e invariante a rotaciones.

Esta función se utiliza previamente a la función lbp indicando los puntos vecinos necesarios para aplicar el operador LBP y el tipo de operador LBP que se desea aplicar (u2,ri,riu2).

Se incluye en el Anexo I el contenido integro de esta función.

3.2.3. Función cont.m

La versión de esta función es la 0.1.0, Esta función devuelve o la varianza, una rotación invariante local o la varianza del histograma de la imagen. La forma de la llamada de esta función es:
$J = CONT(I, R, N, LIMS, MODE)$
Donde tenemos que:

- I es la imagen
- R el radio del circulo de puntos vecinos.
- N el Número de puntos de muestreos, número de puntos espaciados al rededor de la circunferencia de radio R.
- LIMS (Límites): Si se informan los límites la función devuelve los valores del descriptor de forma discreta sino, lo hace de forma continua.
- MODE: El único valor permitido para esta variable es 'i', para que devuelva la image.

En el Anexo I tenemos el código integro de esta función.

3.3. Nuevos conjunto de datos

Con el objetivo de explotar al máximo lo que el operador LBP puede aportar y analizar las posibilidades del mismo, a las imágenes obtenidas de los vídeos se les aplica diferentes operadores LBP al objeto de generar nuevas representaciones del espacio facial usando distintas aproximaciones generando a las que identificaremos con el prefijo o sufijo LBP, que formaran parte del conjunto de datos a los que les aplicaremos los procesos de extracción de características, al objeto de analizarlas.

Se incluyen nuevas imágenes en el conjunto de datos a los que aplicaremos los diferentes algoritmos de clasificación que nos permitirán medir y comparar resultados aprovechando la funcionalidad para generar imágenes en formato LBP a partir de imágenes en formato RGB y de profundidad. En concreto:

1. A partir de las imágenes en formato RGB se obtienen mediante el operador LBP estándar un nuevo conjunto de imágenes.

2. A partir de las imágenes en formato de profundidad se obtienen mediante el operador LBP estándar un nuevo conjunto de imágenes.

3. Se ha generado nuevas imágenes a partir de las imágenes en formato de profundidad y aplicándole el operador *cont.m*.

4. A partir de las imágenes en formato RGB, se han generado nuevas imágenes utilizando el parámetro LBP uniforme, con R = 1 y P= 8, los valores estándar del LBP de R y P.

5. A partir de las imágenes en formato de profundidad, se han generado nuevos conjuntos de imágenes utilizando los parámetros LBP uniforme y con R = 1 y P = 8, los valores estándar de LBP de R y P.

6. A partir de las imágenes en formato de RGB se generan nuevos conjuntos de imágenes utilizando los parámetros LBP uniforme con R = 2 y P =16.

7. A partir de las imágenes en formato de profundidad se generan nuevos conjuntos de imágenes utilizado los parámetros de LBP uniforme y R = 2 y P =16.

Las siguientes figuras muestran el resultado de aplicar diferentes operadores LBP a las imágenes RGB y de profundidad del conjunto de entrenamiento. De la misma forma se han generado las correspondientes

Figura 3.11: Muestras de imágenes de entrenamiento en formato LBP a partir de imágenes en formato RGB

versiones de los conjuntos de test tanto de las imágenes RGB como de la versión de profundidad, siendo el resultado el que se muestra en las figuras 3.13, 3.14, 3.15, 3.16, 3.17 y 3.18.

3.4. Procedimiento para la caracterización mediante los operadores LBP

El vector de características vinculado al operador LBP consiste en un histograma de escala de grises, que dependerá del número de puntos, si es uniforme y de la invarianza a rotaciones. Para los diferentes conjuntos de datos se han generado los histogramas tanto paras la imágenes en formato RGB como a la imágenes en formato de profundidad, para los conjuntos de entrenamiento y de test:

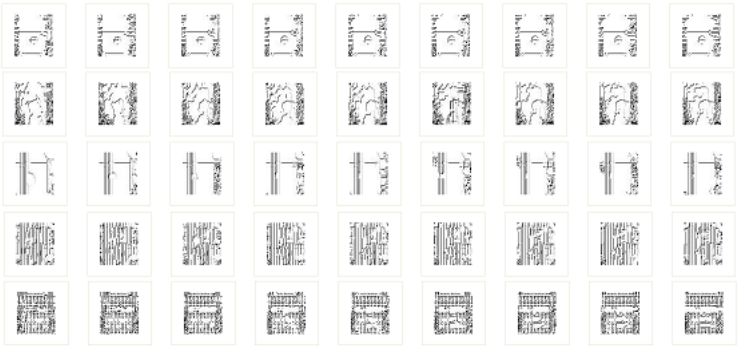


Figura 3.12: Muestras de imágenes de entrenamiento en formato LBP a partir de imágenes en formato de profundidad

Figura 3.13: Muestras de imágenes de test en formato LBP a partir de imágenes en formato RGB

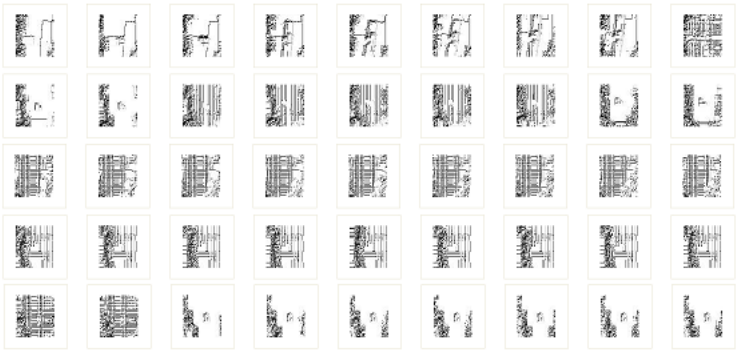


Figura 3.14: Muestras de imágenes de test en formato LBP a partir de imágenes en formato de profundidad

- Histogramas para R =1 y P = 8: $lbp(Imagen, 1, 8,' nh')$, recordemos que 'nh' implica histograma normalizado
- Histogramas para R = 2 y P = 16: $lbp(Imagen, 2, 16,' nh')$.
- Histograma para R=1 y P = 8, de LBP uniforme, con las siguientes llamadas a las funciones map-

Figura 3.15: Muestras de imágenes LBP con parámetros u2 y Radio = 1 y Puntos = 8 a partir de imágenes en formato RGB, con diferente 'map' en matlab

Figura 3.16: Muestras de imágenes LBP con parámetros u2 y Radio = 2 y Puntos = 16 a partir de imágenes en formato RGB

ping y lbp:

$mapping = getmapping(8,'u2');$
$lbp(imagen, 1, 8, mapping,'nh');$

- Histograma para R=1 y P = 8, de LBP invariante a rotaciones, las llamadas a las funciones realizadas fueron:

Figura 3.17: Muestras de imágenes LBP con parámetros u2 y Radio = 1 y Puntos = 8 a partir de imágenes en profundidad, , con diferente 'map' en matlab

$mapping = getmapping(8,'ri');$
$lbp(imagen, 1, 8, mapping,'nh');$

- Histograma para R=1 y P = 8, de LBP normalizado e invariante a rotaciones, las llamadas a las funciones fueron:

  $mapping = getmapping(8,'riu2');$
  $lbp(imagen, 1, 8, mapping,'nh');$

Como resultado de la aplicación de los operadores obtenemos un histograma de valores que son los atributos de la imagen. Con el conjunto de histogramas uno por imagen, se genera el fichero en formato 'arff' a los que les aplicaremos los algoritmos de aprendizaje.

Veamos un ejemplo para ilustrar el proceso de generación de un histograma para una imagen. Como ejemplo vamos de forma detallada.

1. Primero selección de los parámetros:
   $R = 1$
   P =8
   Tipo= U2 -normalizado
   Histograma: Normalizado.

2. Selección de la imagen a la que le vamos aplicar los operadores LBP.

3. Se le aplica la función $getmapping$, con los siguientes parámetros:
   mapping = getmapping (8,'u2');

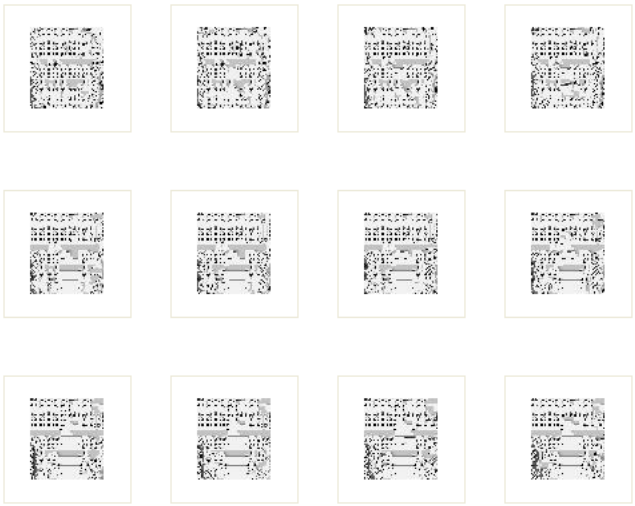


Figura 3.18: Muestras de imágenes LBP con parámetros u2 y Radio = 2 y Puntos = 16 a partir de imágenes en profundidad

Figura 3.19: Ejemplo de imágenes a la que se le aplica el operador LBP

4. Se aplica la función *lbp*, con los siguientes parámetros:
   H = lbp (imagen,1,8,mapping,'nh');
5. Se obtiene el histograma, que se puede observar en la figura 3.20. El último atributo vinculado al histograma LBP es la clase a la qué pertenece el rostro de la persona, es decir, quien es.

Como resultado final de este proceso se obtiene un ficheros en formato 'arff' para WEKA por cada aproximación de representación del espacio facial.

3.5. Otros descriptores de caracterización de la imágenes

Para poder realizar la comparación de resultados se utilizan otros dos métodos de extracción de características en concreto PCA y el mapa de píxeles. A cada uno de los conjuntos datos les aplicamos los procesos que permiten obtener su vectores de características.

3.5.1. Generación análisis de componentes principales

Antes de realizar el proceso para obtener la caracterización mediante PCA de las muestras, primero se ha de preparar los datos de una forma determinada para automatizar todo el procesos, para ello, organizamos las imágenes por tipos en directorios:

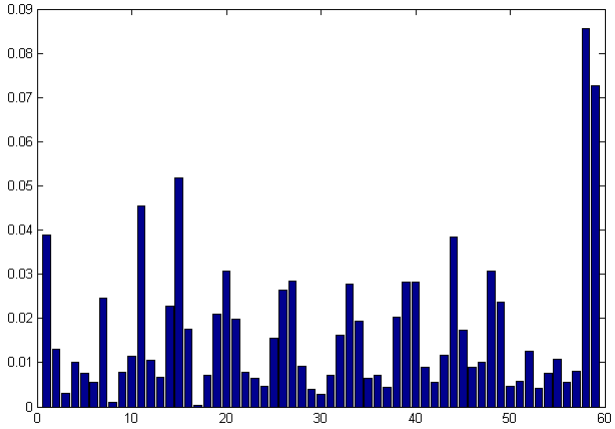


Figura 3.20: Ejemplo de histograma LBP normalizado

- $\backslash RGB$
- $\backslash DEPTH$
- $\backslash LBP - RGB$
- $\backslash LBP - DEPTH$
- $\backslash LBP - DEPTH - CONT$
- $\backslash LBP - RGB - U2(1, 8)$
- $\backslash LBP - DEPTH - U2(1, 8)$
- $\backslash LBP - RGB - U2(2, 16)$
- $\backslash LBP - DEPTH - U2(2, 16)$

Y tenemos la misma estructura para los datos de test que nos sirven para verificar la bondad de los sistemas de aprendizaje.

En cada uno de los directorios anteriores tendremos una carpeta por cada una de las clases, que contiene directamente las imágenes, en concreto:

- $\backslash RGB \backslash 1$
- $\backslash RGB \backslash 2$
- $\backslash RGB \backslash 3$
- $\backslash RGB \backslash 4$
- $\backslash RGB \backslash 5$

Y de estos directorios cuelgan las imágenes de cada clase.

- $\backslash RGB \backslash 1 \backslash imagen_1$
- $\backslash RGB \backslash 1 \backslash imagen_2$
- $\backslash RGB \backslash 1 \backslash imagen_2$
- $\backslash RGB \backslash 1 \backslash \cdots\cdots$
- $\backslash RGB \backslash 1 \backslash imagen_n$

Objetivos

Uno de los objetivos de PCA es reducir la dimensionalidad del espacio de representación de los datos sin pérdida de información, que permitirá la caracterización de las muestras en un espacio de dimensión menor con la información realmente relevante, eliminando el ruido y la redundancia. En este apartado determinaremos la dimensión del espacio de características mínimo para que, sin pérdida de información, nos garantice un nivel óptimo de clasificación de los rostros. Para ello, a partir de la imágenes procedemos:

- Obtener la matriz PCA
- Generar el fichero en formato 'arff' que es el formato que utiliza WEKA.

Proceso

La matriz PCA tiene un número de filas igual al número de imágenes de entrenamiento o de test y como número de columnas el producto de los píxeles de la imagen, esto es si la imagen tiene 59x65 el número de columnas es 3835, que son los atributos de la imagen. Para el proceso de generar fichero 'arff' se utiliza una función proporcionada en la asignatura de Biometría Computacional del Máster que, en concreto, utiliza el método SVD para genera PCA y que permite obtener vectores de características con diferente número de atributos.

Para seleccionar el número de atributos mínimo que se necesitan para una óptima caracterización del conjunto de imágenes se generan los ficheros 'arff' PCA con los siguientes números de atributos:

- 50 atributos
- 100 atributos
- 250 atributos
- 500 atributos
- 1000 atributos
- Atributos igual al número de píxeles de la imagen.

Para determinar el fichero PCA con menor número de atributos pero que nos permita obtener resultados equivalentes al que tiene el mayor número de atributos procederé a realizar las pruebas con las imágenes en formato RGB y realizar la comparación con los algoritmos de aprendizaje K-NN y SVM utilizando el conjunto de test de control. Se han obtenido los siguientes resultados:

	Aciertos	
Atributos	K-NN	SVM
50	102	238
100	85	171
250	91	146
500	126	151
1000	134	178
3835	101	145

Para el caso del algoritmo K-NN se utiliza una medida de distancia en concreto he utilizado la distancia de Chebyshev, que es la que me ha dado mejores resultados que la distancia Euclidea (la más común), la fórmula para el cálculo de esta distancias es:

$$D(i,j) = Max|x_{ki} - x_{kj}|$$

Conclusión

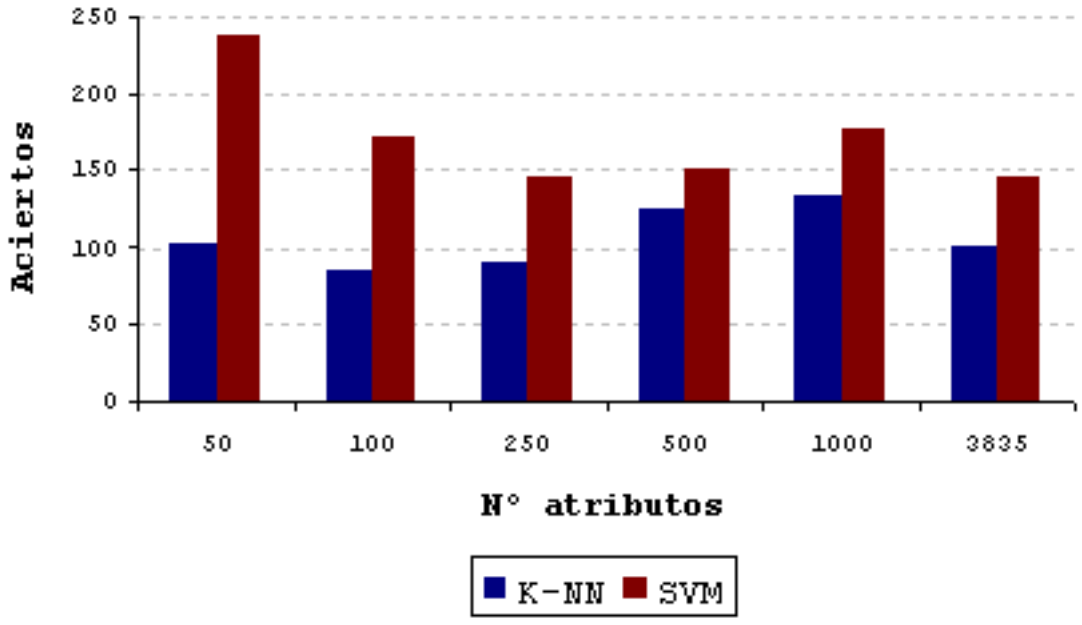


Figura 3.21: Evolución de los algoritmos de clasificación en función del número de atributos PCA

Como reflejan los datos de la tabla y la gráfica de la figura 3.21 y los resultados obtenidos con 50 atributos y los resultados obtenidos con 1000 atributos dan mejores resultados, sin embargo se seleccionar la versión PCA con 50 atributos por:

1. Da mejor resultado en algoritmo SVM que es en el que vamos a centrar el trabajo frente al K-NN.
2. Al ser menor el número de atributos tiene menos coste computacional.

3. El tamaño de los ficheros, con los que vamos a trabajar, son menores, el tiempo de producción de los datos es menor.

Una vez que se ha seleccionado el número de atributos con el que vamos a trabajar generamos los ficheros correspondientes para los diferentes tipos de imágenes que tenemos.

3.5.2. Generación de mapas de píxeles

A los efectos de establecer comparaciones se ha determinado utilizar directamente el mapa de píxeles de las imágenes. El procedimiento que se ha utilizado para este vector de características del conjunto de datos es el siguiente. Se crea un matriz con una fila por muestras y con nxm columnas, siendo n y m las dimensiones de la imagen, a partir de esta matriz se genera fichero en formato 'arff' para WEKA, de forma que cada una de las columnas son los atributos.

3.6. Algoritmos de clasificación

Como ya hemos mencionados a todos los vectores de características les vamos aplicar dos algoritmos de aprendizaje, el K-NN y el SVM, que han sido vistos en profundidad en secciones anteriores.

Antes de pasar a los resultados obtenidos indicar que el algoritmo de aprendizaje K-NN, permite parametrizar qué medida de distancia utilizar, en concreto he utilizado, en algunos casos, tres:

1. Euclidea
   La distancia euclídea es la medida más conocida y sencilla de comprender, pues su definición coincide con el concepto más común de distancia.

$$D(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.1}$$

   Donde $P = (p_1, p_2, \cdots, p_n)$ y $Q = (q_1, q_2, \cdots, q_n)$

   La distancia Euclídea, a pesar de su sencillez de cálculo y de que verifica algunas propiedades interesantes tiene dos graves inconvenientes:

   - El primero de ellos es que la euclídea es una distancia sensible a las unidades de medida de las variables: las diferencias entre los valores de variables medidas con valores altos contribuirán en mucha mayor medida que las diferencias entre los valores de las variables con valores bajos. Como consecuencia de ello, los cambios de escala determinarán, también, cambios en la distancia entre los individuos. Una posible vía de solución de este problema es la tipificación previa de las variables, o la utilización de la distancia euclídea normalizada.
   - El segundo inconveniente no se deriva directamente de la utilización de este tipo de distancia, sino de la naturaleza de las variables. Si las variables utilizadas están correlacionadas, estas variables nos darán una información, en gran medida redundante. Parte de las diferencias entre los valores individuales de algunas variables podrían explicarse por las diferencias en otras variables. Como consecuencia de ello la distancia euclídea inflará la disimilaridad o divergencia entre los individuos.

La solución a este problema pasa por analizar las componentes principales (que están incorrelacionadas) en vez de las variables originales. Otra posible solución es ponderar la contribución de cada par de variables con pesos inversamente proporcionales a las correlaciones, lo que nos lleva, como veremos a la utilización de la distancia de Mahalanobis.

La distancia euclídea será, en consecuencia, recomendable cuando las variables sean homogéneas y estén medidas en unidades similares y/o cuando se desconozca la matriz de varianzas.

2. Chevyshev
   Es un resultado que ofrece una cota inferior a la probabilidad de que el valor de una variable aleatoria con varianza finita esté a una cierta distancia de su esperanza matemática.

$$D(i,j) = Max|x_{ki} - x_{kj}|$$

3. Manhattan
   La distancia entre dos puntos es la suma de las diferencias (absolutas) de sus coordenadas.

$$d(p,q) = \sum_{i=1}^{n} |p - q| \tag{3.2}$$

3.7. Experimentos realizados

Una vez hemos procedido a generar todos los datos y ficheros que necesitamos en nuestro proceso de análisis, nos queda la fase principal que es realizar los procesos de aprendizajes, a partir de los diferentes vectores de características y verificar el sistema de aprendizaje, de forma comparada para por último analizar los datos que nos permitan, por último, extraer las características.

Los experimentos han sido realizados en las siguientes fases:

1. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas mediante PCA y mapa de píxeles.
2. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas mediante el operador LBP aplicado a la imagen completa.
3. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas mediante el operador LBP aplicado a la imagen dividida en regiones.
4. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas de forma conjunta (RGB y profundidad) mediante el operador LBP aplicado a la imagen dividida en regiones.

3.7.1. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas mediante PCA y mapa de píxeles

Proceso

En esta primera fase, a las imágenes en formato RGB y de profundidad y, caracterizadas mediante el mapa de pixeles y PCA, le aplicaremos los algoritmos de clasificación K-NN y SVM. Posteriormente,

a las imágenes que hemos obtenido de aplicar diferentes operadores LBP les aplicaremos la mismas técnicas de caracterización y clasificación y formarán también parte del grupo de contraste.

Objetivos

En esta fase se pretende:

- Obtener datos de contraste
- Determinar si existe mejora en el proceso de clasificación al utilizar imágenes transformadas mediante diferentes operadores LBP y caracterizadas mediante el mapa de píxeles y PCA.
- Analizar si se obtienen mejores resultados en las imágenes en formato de profundidad que en formato RGB.

Datos Obtenidos

TIPOS DE IMÁGENES	PCA DE LOS PÍXEL DE LAS IMÁGENES			MAPA DE PÍXELES			
	K-NN		SVM	K-NN			SVM
	CHEBYSHEV	EUCLIDEA		CHEBYSHEV	MANHATTAN	EUCLIDEA	
RGB	19,65%	16,57%	45,86%	26,78%	46,63%	44,32%	42,97%
DEPTH	36,42%	38,15%	27,36%	41,62%	42,58%	34,30%	32,76%
LBP-RGB	43,55%	34,68%	51,06%	27,75%	69,36%	66,09%	55,49%
LBP-DEPTH	32,95%	40,66%	39,69%	27,75%	29,67%	28,90%	42,39%
LBP-DEPTH-CONT	26,01%	29,87%	22,93%	27,75%	30,44%	30,25%	30,64%
LBP-RGB-U2 (1,8)	19,65%	27,94%	51,45%	23,51%	64,74%	59,15%	56,45%
LBP-RGB-U2 (2,16)	37,38%	40,08%	48,94%	20,42%	69,36%	57,03%	56,65%
LBP-DEPTH-U2 (1,8)	47,01%	44,70%	38,54%	12,33%	14,26%	14,45%	59,34%
LBP-DEPTH-U2 (2,16)	45,28%	35,84%	24,08%	13,10%	15,03%	13,29%	44,89%

Figura 3.22: Resultados al aplicar K-NN y SVM a los datos de las imágenes.

En la tabla 3.22 y los gráficos que aparecen en la figuras 3.23 y 3.24 se muestran los resultados obtenidos después de aplicar los algoritmos de aprendizaje a los datos, se incluyen, el algoritmo K-NN con dos tipos de distancias, la de Chebyshev y la distancias Euclidea, y el algoritmo SVM, el cuál se ha parametrizado mediante la utilización de un *Kernel* lineal y el parámetro normalizar a cierto, como se observa en la figura 3.25, el resto de los parámetros son los que vienen por defecto en WEKA.

Recordar que a los datos a los que hemos aplicado los resultados son:

- RGB: Imágenes en formato RGB estándar a los que se les ha hecho la conversión a escala de grises.
- DEPTH: Imágenes en formato de profundidad.

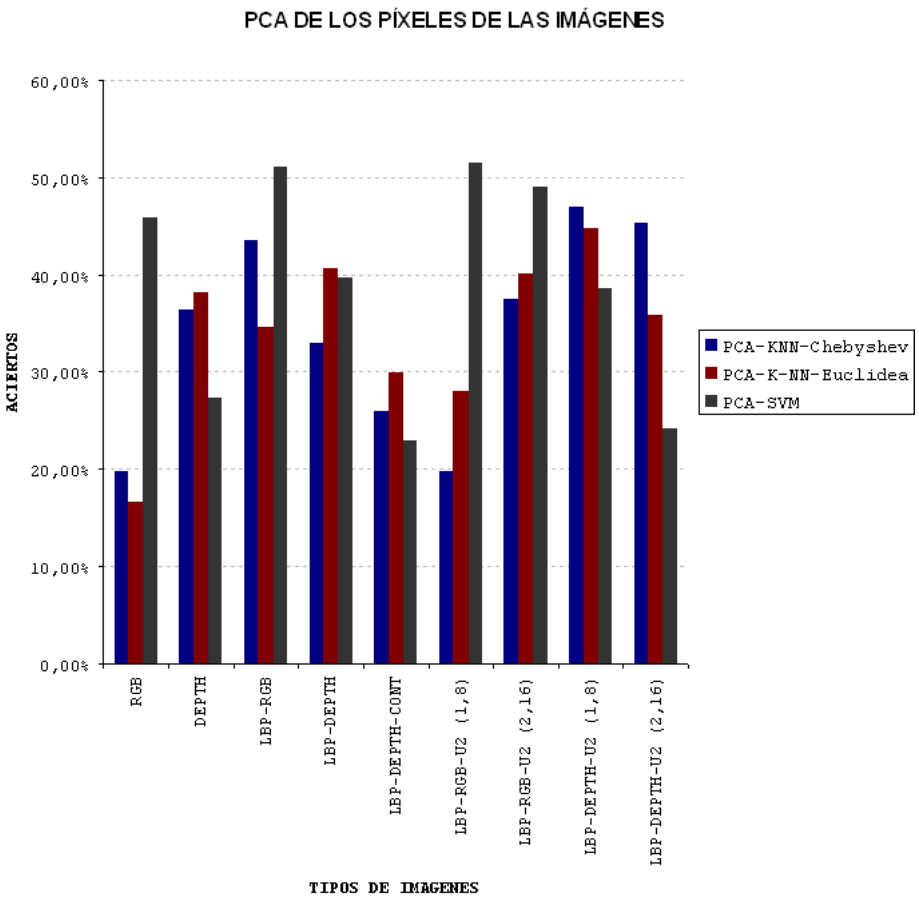


Figura 3.23: Gráfica de los resultados obtenidos de aplicar los algoritmos de clasificación a la imágenes caracterizadas por PCA.

- LBP-RGB: Imagen RGB a la que se le aplica el operador LBP para generar una nueva imagen
- LBP-DEPTH: Imagen de profundidad a la que se aplica el operador LBP para generar una nueva imagen.
- LBP-DEPTH-CONT: Imagen en profundidad a la que se le aplica el operador LBP/C para generar nueva imagen
- LBP-RGB-U2 (1,8) :
- LBP-RGB-U2 (2,16):
- LBP-DEPTH-U2 (1,8) :
- LBP-DEPTH-U2 (2,16) :

Conclusiones

De la tabla de resultados que aparece en la figura 3.22 y de los gráficos de las figuras 3.23 y 3.24 podemos concluir que:

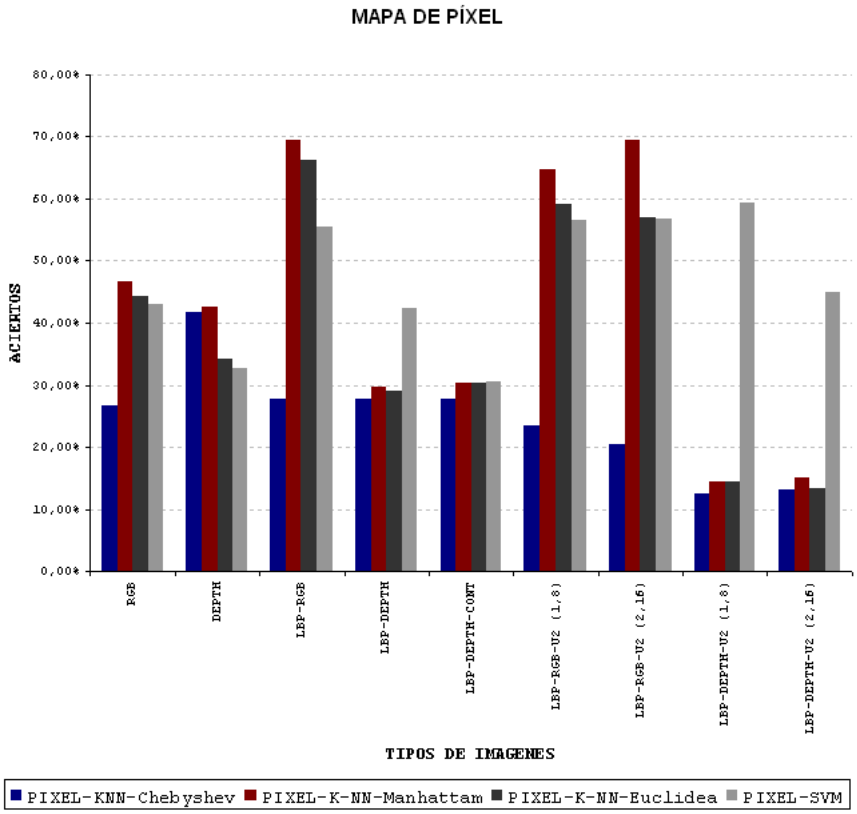


Figura 3.24: Gráfica de los resultados obtenidos de aplicar los algoritmos de clasificación a la imágenes caracterizadas por el mapa de píxeles.

1. Disponemos de datos de contraste que nos permitirán comparar con el resto de métodos de caracterización.
2. Para este conjunto de datos obtenemos mejores resultados con el mapa de píxeles que con la caracterización mediante PCA.
3. Con la caracterización PCA se obtienen mejores resultados utilizando el algoritmo SVM que el K-NN
4. Con estas dos caracterizaciones no podemos concluir de forma clara que un formato de imagen de mejores resultados que otro, ya que depende incluso del tipo de distancia escogida.
5. Con caracterización PCA e imágenes en formato RGB se obtienen mejores resultados con el clasificador SVM que con el clasificador K-NN.

Los valores máximo más destacados que obtenemos son:

- $69,36\,\%$ con caracterización mediante mapa de píxeles, algoritmos de clasificación K-NN y distancia Manhattan y con el tipo de imagen LBP-RGB y LBP-RGB-U2 (2,16).
- $59,34\,\%$ con caracterización mediante mapa de píxeles, algoritmos de clasificación SVM y con el tipo de imagen LBP-DEPTH-U2 (1,8).

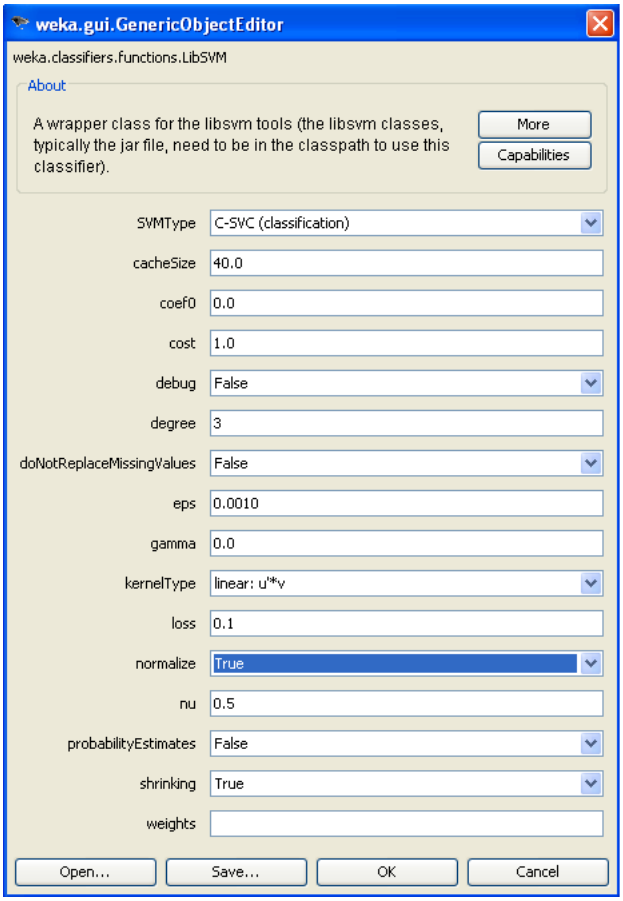


Figura 3.25: Parametrización del algoritmo SVM- Tipo de kernel y normalizar a cierto

3.7.2. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas mediante el operador LBP aplicados a la imagen completa.

Proceso

Se han calculado los histogramas LBP de las imágenes en formato RGB y en profundidad (DEPTH) tanto de las imágenes que componen el conjunto de entrenamiento como las de verificación a las que se les ha aplicado diferentes parámetros, además todos los histogramas se han normalizado. Recordemos que han quedando el siguiente conjunto de histogramas:

1. Histogramas normalizados de las imágenes en formato RGB, en la tabla de resultados utilizamos el nombre RGB.

2. Histogramas normalizados de las imágenes en formato de profundidad, en la tabla de resultados utilizamos el nombre $DEPTH$.

3. Histogramas normalizados de las imágenes en formato RGB al que se han indicado los parámetros $R = 2$ y $P = 16$, en la tabla de resultados se ha etiquetado con el nombre 2_RGB.

4. Histogramas normalizados de las imagenes en formato de profundidad que ha sido generado con los parámetros $R = 2$ y $P = 16$, en la tabla de resultados se ha etiquetado como 2_DEPTH.

5. Histogramas normalizados de las imágenes en formato RGB a los que se les ha aplicado el parámetro de uniforme LBP, en la tabla de resultados utilizamos el nombre RGB_U2.

6. Histogramas normalizados de las imágenes en formato de profundidad a los que se les ha aplicado el parámetro de uniforme LBP,a en la tabla de resultados utilizamos el nombre $DEPTH\_U2$.

7. Histogramas normalizados de las imágenes en formato RGB al que se le ha aplicado el parámetro de invarianza rotaciones, en la tabla de resultados se utiliza el nombre RGB_RI.

8. Histogramas normalizados de las imágenes en formato de profundidad al que se le ha aplicado el parámetro de invarianza a rotaciones, en la tabla de resultados se utiliza el nombre $DEPTH\_RI$.

9. Histogramas normalizados de las imágenes en formato RGB al que se le ha aplicado el parámetro de invarianza rotaciones y uniforme LBP, en la tabla de resultados se utiliza el nombre RGB_RIU2.

10. Histogramas normalizados de las imágenes en formato de profundidad al que se le ha aplicado el parámetro de invarianza rotaciones y uniforme LBP, en la tabla de resultados se utiliza el nombre $DEPTH\_RIU2$.

A los histogramas LBP obtenidos se les han aplicado los algoritmo de aprendizaje K-NN y SVM. Del algortimo K-NN se han utilizado tres cálculos de distancias, la Euclidea, Mahattan y la de Chebyshev, en la figura 3.26 se pueden ver los resultados obtenidos.

Objetivos

Se prentede:

- Obtener los resultados, aciertos, de aplicar los algoritmos de clasificación a los diferentes conjuntos de imágenes caracterizadas mediante los operadores LBP
- Determinar si mejoramos resultados con respecto a los datos obtenidos con las otras caracterizaciones.
- Determinar si se obtienen mejores resultados con la imágenes de profundidad que con las de formato RGB.
- Por último ver si es posible establecer alguna conclusión sobre el comportamiento de un algoritmo de aprendizaje.

Resultados obtenidos de los experimentos

En la tabla incluida en a figura 3.26 y en el gráfico de la figura 3.27 reflejan los datos obtenidos, como se observa muy pocos apenas superan el $50\,\%$ de acierto.

Conclusiones

En comparación con los datos de contraste y los objetivos que planteé señalamos.

- En general los datos son peores que los obtenidos con el caracterizador de mapa de píxeles, aunque para algún tipo en concreto pudiera ser que mejorar para algún clasificador en concreto.
- Obtenemos mejores resultados en la imágenes de profundidad que en las RGB.

TIPOS DE IMÁGENES	HISTOGRAMAS			
	K-NN			SVM
	CHEBYSHEV	MANHATTAN	EUCLIDEA	
RGB	29,09%	39,50%	42,39%	42,00%
DEPTH	29,48%	34,10%	32,76%	51,25%
2-RGB	38,15%	43,35%	42,39%	38,73%
2-DEPTH	29,67%	34,49%	32,76%	50,29%
U2_RGB	37,96%	43,35%	42,39%	38,73%
U2_DEPTH	29,48%	34,49%	32,76%	50,29%
RGB_RI	32,56%	26,20%	29,48%	25,82%
DEPTH_RI	26,01%	33,72%	30,25%	28,13%
RGB_RIU2	26,59%	21,19%	23,51%	19,46%
DEPTH_RIU2	35,84%	33,14%	32,76%	30,25%

Figura 3.26: Resultados obtenidos de los histogramas LBP aplicando los algoritmos K-NN y SVM

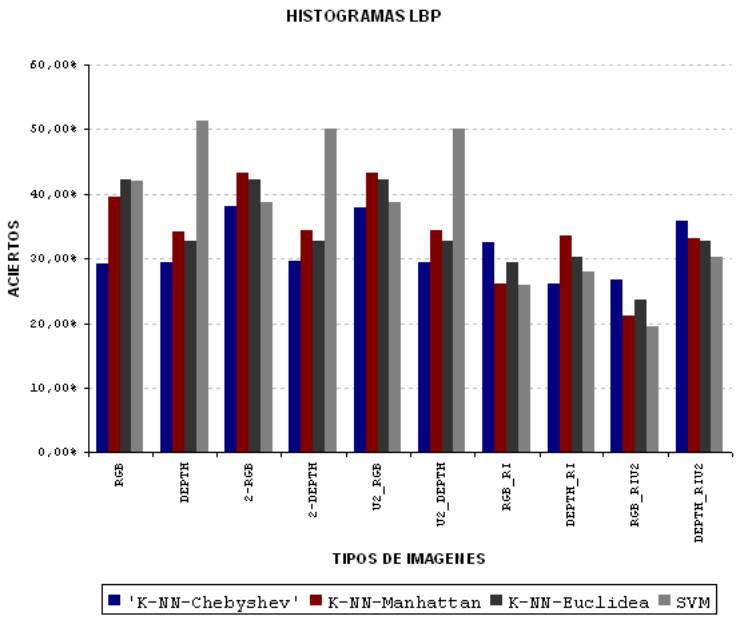


Figura 3.27: Gráfico de los resultados obtenidos aplicando el operador LBP a los diferentes conjuntos de imágenes.

- Para imágenes en formato RGB los algoritmos K-NN y SVM tiene comportamiento parecido, aunque ligeramente mejor el clasificador K-NN.

- Para la imágenes en formato de profundidad tiene mejor comportamiento el clasificador SVM

3.7.3. Aplicación de los algoritmos de clasificación a las imágenes caracterizadas mediante el operador LBP aplicados a la imagen dividida en regiones.

Procedimiento

Se propone una mejora en la caracterización de las imágenes de los rostros que consiste básicamente en que se debe aprovechar, en el proceso de clasificación, las características espaciales de las partes de la cara y para ellos se propone:

- Dividir el rostros en regiones
- A cada región aplicarle el operador LBP.
- Concatenar los histogramas de las partes para formar el histograma del conjunto.

Basándonos en esta metodología se realizan un nuevo conjunto de experimentos para determinar si la misma mejora los resultados obtenidos, para ello, procederemos de la siguiente forma:

- Se procederá a dividir las imágenes de la cara, y como sea que no hay indicación en el método del número de divisiones del rostros que mejore el proceso de reconocimiento facial, en 3x3, 5x5 y 7x7 regiones.
- A cada una de las regiones se le aplicará el operador LBP con diferentes parámetros:
  - Operador LBP con P = 8 y R = 2. $LBP_{8,2}$
  - Operador LBP uniforme con P = 8 y R = 2, $LPR^{u2}_{8,2}$
- Finalmente se concatenaran los histogramas para formar la caracterización de la imagen de la cara.

Objetivos

Con la utilización de esta técnica de aplicación del operador LBP para el reconocimiento facial nos planteamos los siguientes objetivos:

- Obtener los datos correspondientes a este operador para los tipos de imágenes y algoritmos de clasificación utilizados.
- Determinar si obtenemos mejorar con respecto al operador LBP aplicado a la imagen íntegra.
- Determinar, para nuestro conjunto, de datos el número de regiones óptima para dividir los rostros.
- Determinar si con esta versión obtenemos mejores resultados que a los obtenidos cuando se caracterizaba con mapa de píxeles y PCA.

Datos obtenidos de los experimentos

Como se recoge en la tabla de la figura 3.28, se han aplicado dos operadores LBP a cuatro tipos de imágenes en concreto tenemos:

- A las imágenes en formato RGB se le aplica el operador $LBP_{8,2}$
- A las imágenes en formato de profundidad se le aplica el operador $LBP_{8,2}$.
- A las imágenes RGB a las que se aplicó el operador LBP, se le aplica el operador $LBP_{8,2}$.

- A las imágenes en formato profundidad a las que se aplicó el operador LBP, se le aplica el operador $LBP_{8,2}$.
- A las imágenes en formato RGB se le aplica el operador $LBP^{u2}_{8,2}$
- A las imágenes en formato de profundidad se le aplica el operador $LBP^{u2}_{8,2}$.
- A las imágenes RGB a las que se aplicó el operador LBP, se le aplica el operador $LBP^{u2}_{8,2}$.
- A las imágenes en formato profundidad a las que se aplicó el operador LBP, se le aplica el operador $LBP^{u2}_{8,2}$.

TIPOS DE IMÁGENES	N° Regiones					
	3X3		5x5		7x7	
	K-NN	SVM	K-NN	SVM	K-NN	SVM
RGB	57,8	77,07	21,96	59,53	20,42	51,64
DEPTH	41,62	41,04	14,84	26,59	11,75	21,39
LBP-RGB	50,29	60,5	24,86	50,1	26,78	47,98
LBP-DEPTH	31,02	25,82	18,11	31,79	13,68	25,05
RGB-U2	76,11	74,37	69,17	63,97	49,13	55,49
DEPTH-U2	35,83	39,11	32,18	25,63	41,43	25,43
RGB-LBP-U2	48,94	60,89	42,39	53,56	35,07	49,32
DEPTH-LBP-U2	28,71	23,7	32,37	24,28	25,24	26,59

Figura 3.28: Resultados de los experimentos obtenidos al aplicar diferentes operadores LBP a la imágenes divididas en regiones

En la figura 3.29 se observa gráficamente los resultados obtenidos y, junto a los datos que se observan en la tabla, se ha obtenido un nivel de aciertos por encima de un $70\,\%$ en al menos tres de los experimentos, lo que implica una mejora sustancial con respecto a las anteriores experimentos.

Conclusiones

A la vista de los datos obtenidos podemos inferir las siguientes conclusiones.

- Los mejores resultados los obtenemos al dividir las imágenes de las caras en 3x3 regiones, con algunas excepciones poco significativas. Los resultados obtenidos al dividir en regiones 5x5 son mejores que los obtenidos al dividir las imágenes en 7x7 regiones.
- Se obtiene mejores resultados en los clasificadores sobre las imágenes en formato RGB que con las imágenes en formato de profundidad.
- No en todas las ocasiones se obtiene mejores resultados con el algoritmo de clasificación SVM.
- Se obtiene los mejores resultados al aplicar el operador $LBP_{8,2}$ al conjunto de datos formato RGB, con un $77,07\,\%$ y con el algoritmo de clasificación SVM, y habiendo dividido la imagen en 3x3 regiones.

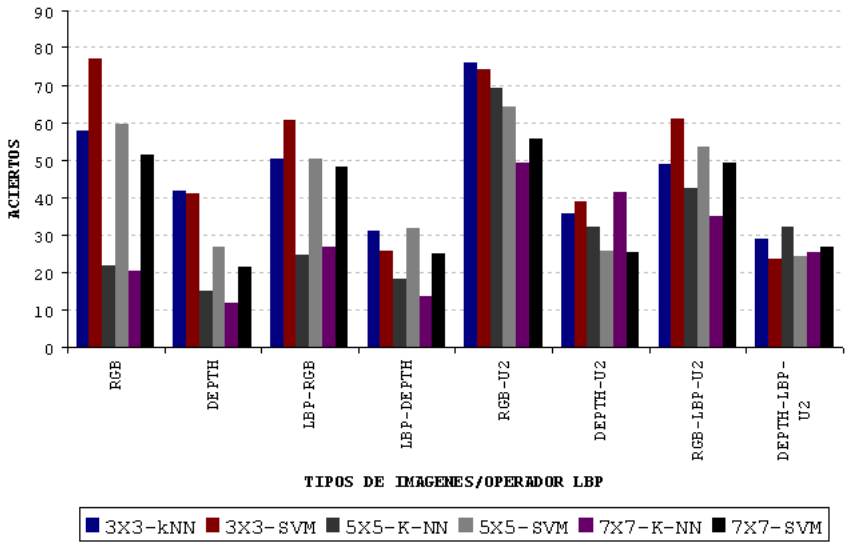


Figura 3.29: Gráficos con los resultados de los algoritmos de clasificación con las caras divididas en regiones aplicados a diferentes conjuntos de imágenes.

- También destacar los resultados obtenidos al aplicar el operador $LBP^{u2}_{8,2}$ a las imágenes en formato RGB, alcanzado los valores máximos tanto para cualquier división y algoritmo de clasificación, excepto para el valor indicado en el punto anterior.
- Como en los otros experimentos se concluye que no se obtiene mejores resultados, para este conjunto de muestras, con las imágenes en formato de profundidad, por lo que no podemos concluir que el uso de esta imágenes mejora los índices de reconocimiento facial, como se esperaban.

3.7.4. Aplicación de los algoritmos de clasificación a una caracterización conjunta de las imágenes RGB y profundida

Procedimiento

En este apartado de los experimentos se pretende determinar si una caracterización conjunta de las imágenes en ambos formato mejora el rendimiento en el proceso de clasificación. Para ello vamos a proceder de la siguiente forma.

- Seleccionar el conjunto de histogramas a concatenar. Como sea que los mejores resultados se han obtenido con las imágenes divididas en regiones se utilizan estos histogramas para ser concatenados.
- Como se ha indicado se va a realizar la caracterización conjunta imágenes RGB y de profundidad, mediante la concatenación de los histogramas.
- Los tipos de imágenes resultantes serán:
  - RGB+DEPTH.
  - LBP-RGB+LBP-DEPTH.
  - RGB-U2 + DEPTH-U2.

- LBP-RGB-U2+LBP-DEPTH-U2.

- En este caso también se utilizaran tres divisiones de las imágenes: 3x3; 5x5 y 7x7.
- Por último se le aplicarán los algoritmos de clasificación K-NN y SVM .

Objetivo

Con este conjunto de experimentos se pretende determinar si la categorización conjunta de las imágenes en formatos RGB y en formato de profundidad generados por la *Kinect* mejoran los resultados usando RGB y profundidad por separados.

Resaltos obtenidos en los experimentos

En las figuras 3.30 y 3.31 podemos observar los datos obtenidos con este experimento y, si los comparamos con los resultados que se observan en las figuras 3.28 y 3.29, en general no se obtiene mejores resultados. Hay que destacar, sin embargo, que en el caso de la división en 7x7 regiones la imagen si se obtiene un mejor rendimiento en la categorización conjunta cuando se utiliza SVM, aunque los resultados se encuentran alrededor del 50 % muy por debajo de los mejores resultados obtenidos sin la caracterización conjunta. Señalar, también, que en la mayoría de las medidas tomadas, se obtiene mejores resultados cuando el histogramas es la concatenación de las imágenes RGB y de profundidad, que cuando la categorización es sólo de las imágenes en profundidad.

TIPOS DE IMÁGENES	N° Regiones					
	3X3		5x5		7x7	
	K-NN	SVM	K-NN	SVM	K-NN	SVM
RGB+DEPTH	45,09%	62,24%	27,17%	51,64%	19,46%	50,29%
LBP-RGB + LBP-DEPTH	37,76%	53,37%	27,36%	26,78%	24,86%	52,60%
RGB-U2 + DEPTH-U2	60,12%	55,30%	49,52%	49,52%	44,89%	51,06%
RGB-LBP-U2 + DEPTH-LBP-U2	34,49%	53,18%	28,32%	46,43%	18,11%	50,10%

Figura 3.30: Tabla con los resultados de la categorización conjunta de las imágenes en profundidad y en formato RGB

Conclusiones

Como ya se ha indicado no se puede concluir que se consigue mejorar los resultados de clasificación con la categorización conjunta mediante la concatenación de histogramas. Y podemos concluir:

- La categorización conjunta da peores resultados, en general, que la extracción de características de las imágenes en formato RGB.
- La categorización conjunta da mejores resultados que la extracción de características mediante el operador LBP de las imágenes en profundidad.
- En el caso de división de la imagen en 7x7 regiones y con el algoritmo de clasificación basado en el SVM da resultados equivalentes independientemente del tipo de imágenes y, además, comparativamente, da mejores resultados que la categorización de imágenes en formato RGB o de profundidad de manera independiente.

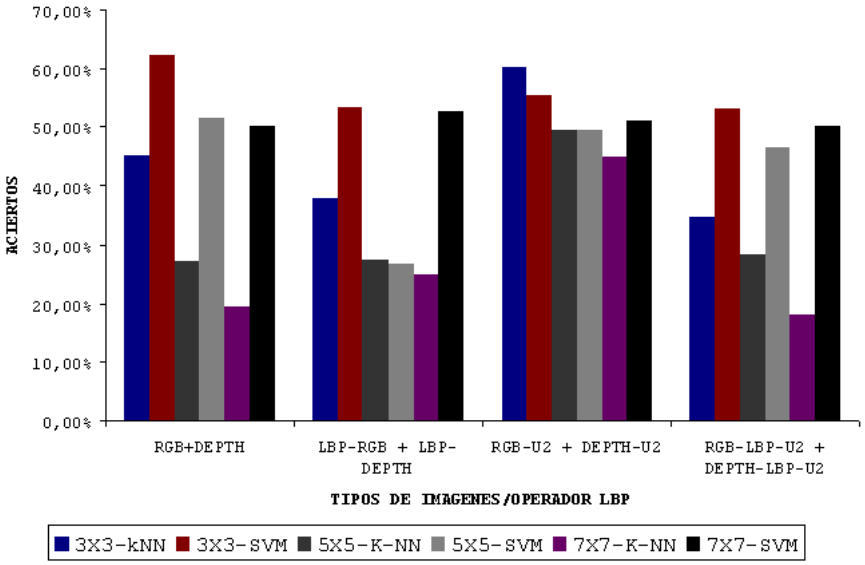


Figura 3.31: Histogramas de resultados de la categorización conjunta de las imágenes en profundidad y en formato RGB

Capítulo 4

Conclusiones y nuevas líneas de trabajo

Los experimentos realizados sobre este conjunto de muestras permite extraer las siguientes conclusiones:

1. ENCARA2 no es capaz de detectar las caras directamente en los vídeos en formato de profundidad.

2. Con los caracterizadores de imágenes de caras más utilizados en los últimos años en los problemas de reconocimiento facial, mapa de píxeles y PCA, no se obtienen mejores resultados con las imágenes de profundidad obtenidos con la *Kinect*.

3. Las imágenes generadas con el operador LBP generan mejor rendimiento en los algoritmos de clasificación que las imágenes sin transformar. Con PCA se obtienen peores resultados que con el mapa de píxeles.

4. Con la caracterización LBP de la imagen de cara completa, se han obtenido mejores resultados con las imágenes en formato de profundidad originales y las generadas mediante transformaciones con el operador LBP que las con las imágenes originales RGB y sus transformadas. Aún así, los mejores resultados obtenidos son bajos, apenas superan ligeramente el $50\,\%$.

5. Se ha obtenidos mejores resultados al aplicar LBP sobre las imágenes divididas en regiones, lo que permite aprovechar las características espaciales.

6. Con las imágenes divididas en regiones 3x3 se obtienen, mejores resultados que cuando se dividen en 5x5 regiones y mucho mejores cuando se dividen en 7x7 regiones.

7. Con las imágenes divididas en regiones se obtienen mejores resultados en las imágenes RGB y sus versiones transformadas que en las de formato de profundidad y sus versiones transformadas.

8. El mejor registro se ha obtenido con la imágenes en formato RGB y con el algoritmo de clasificación SVM y con la imagen dividida en 3x3 regiones.

9. El mejor conjunto de datos se obtiene con el operador $LBP^{u2}_{8,2}$ aplicado a las imágenes en formato RGB, siendo:

   - Algoritmo de clasificación: K-NN:
     - Imágenes divididas en 3x3 regiones: $76,11\,\%$
     - Imágenes divididas en 5x5 regiones: $69,17\,\%$
     - Imágenes divididas en 7x7 regiones: $49,13\,\%$

- Algoritmo de clasificación: SVM
  - Imágenes divididas en 3x3 regiones: $74,37\,\%$
  - Imágenes divididas en 5x5 regiones: $63,97\,\%$
  - Imágenes divididas en 7x7 regiones: $55,49\,\%$

A pesar de que varias pruebas han mostrado resultados con una tasa de aciertos superior al $70\,\%$ los resultados de las pruebas son pobres y peores con las imágenes de profundidad y sus transformaciones, por lo que concluimos que, con estas imágenes de profundidad, no se obtienen resultados óptimos con esta caracterización de las imágenes y los algoritmos de clasificación seleccionados para las pruebas.

10. Con el conjunto de muestras que disponemos de forma general no podemos concluir que la categorización conjunta de las imágenes en profundidad y en formato RGB mejore el rendimiento de los clasificadores faciales entrenados.

11. Los vídeos generados por la *Kinect* duran el mismo tiempo y por lo tanto están sincronizados.

Líneas futuras de trabajo

Utilizando el operador LBP uniforme con la imagen dividida en regiones y concatenando los histogramas de cada región se han conseguido los mejores resultados, lo que invita a pensar que profundizando en este tipo de operador puede mejorar los niveles de acierto, por ejemplo, primando determinadas regiones con mayor información relevante frente a otras.

Otra segunda alternativa sería concatenando a los histogramas de las regiones en las que se dividen las imágenes el histograma de la imagen sin dividir.

Usar esta metodología para otro tipo de clasificadores de características biométricas por ejemplo clasificadores de género, podrían dar buenos resultados máxime si utilizamos pesos para dar prioridad a determinadas regiones frente otras dando más énfasis en aquellas regiones en las que pueden existir más diferencias entre los géneros: ojos, orejas y boca.

Otra línea de mejora sería disponer de un sistema de detección facial para imágenes en profundidad específico.

Capítulo 5

Anexos

5.1. Funciones Matlab vinculadas a los operadores LBP

Se incluyen las tres funciones Matlab incluidas en la http://www.cse.oulu.fi/MVG/Downloads/LBPMatlab de la Universidad de OULU, que son:

- lbp.mat
- getmapping.mat
- cont.m

LBP.MAT

```
function result = lbp(varargin) % image, radius, neighbors, mapping, mode)
% Version 0.3.2
% Authors: Marko Heikkila and Timo Ahonen

% Changelog
% Version 0.3.2: A bug fix to enable using mappings together with a
% predefined spoints array
% Version 0.3.1: Changed MAPPING input to be a struct containing the mapping
% table and the number of bins to make the function run faster with high number
% of sampling points. Lauge Sorensen is acknowledged for spotting this problem.


% Check number of input arguments.
error(nargchk(1,5,nargin));

image=varargin{1};
d_image=double(image);

if nargin==1
    spoints=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
    neighbors=8;
```

```
    mapping=0;
    mode='h';
end

if (nargin == 2) && (length(varargin{2}) == 1)
    error('Input arguments');
end

if (nargin > 2) && (length(varargin{2}) == 1)
    radius=varargin{2};
    neighbors=varargin{3};

    spoints=zeros(neighbors,2);

    % Angle step.
    a = 2*pi/neighbors;

    for i = 1:neighbors
        spoints(i,1) = -radius*sin((i-1)*a);
        spoints(i,2) = radius*cos((i-1)*a);
    end

    if(nargin >= 4)
        mapping=varargin{4};
        if(isstruct(mapping) && mapping.samples ~= neighbors)
            error('Incompatible mapping');
        end
    else
        mapping=0;
    end

    if(nargin >= 5)
        mode=varargin{5};
    else
        mode='h';
    end
end

if (nargin > 1) && (length(varargin{2}) > 1)
    spoints=varargin{2};
    neighbors=size(spoints,1);

    if(nargin >= 3)
        mapping=varargin{3};
        if(isstruct(mapping) && mapping.samples ~= neighbors)
            error('Incompatible mapping');
        end
    else
```

```
        mapping=0;
    end

    if(nargin >= 4)
        mode=varargin{4};
    else
        mode='h';
    end
end

%Determine the dimensions of the input image.
[ysize xsize] = size(image);

miny=min(spoints(:,1));
maxy=max(spoints(:,1));
minx=min(spoints(:,2));
maxx=max(spoints(:,2));

%Block size, each LBP code is computed within a block of size bsizey*bsizex
bsizey=ceil(max(maxy,0))-floor(min(miny,0))+1;
bsizex=ceil(max(maxx,0))-floor(min(minx,0))+1;

%Coordinates of origin (0,0) in the block
origy=1-floor(min(miny,0));
origx=1-floor(min(minx,0));

%Minimum allowed size for the input image depends
%on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
  error('Too small input image. Should be at least (2*radius+1) x (2*radius+1)');
end

%Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;

%Fill the center pixel matrix C.
C = image(origy:origy+dy,origx:origx+dx);
d_C = double(C);

bins = 2^neighbors;

%Initialize the result matrix with zeros.
result=zeros(dy+1,dx+1);

%Compute the LBP code image

for i = 1:neighbors
```

```
  y = spoints(i,1)+origy;
  x = spoints(i,2)+origx;
  % Calculate floors, ceils and rounds for the x and y.
  fy = floor(y); cy = ceil(y); ry = round(y);
  fx = floor(x); cx = ceil(x); rx = round(x);
  % Check if interpolation is needed.
  if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
    % Interpolation is not needed, use original datatypes
    N = image(ry:ry+dy,rx:rx+dx);
    D = N >= C;
  else
    % Interpolation needed, use double type images
    ty = y - fy;
    tx = x - fx;

    % Calculate the interpolation weights.
    w1 = (1 - tx) * (1 - ty);
    w2 =      tx  * (1 - ty);
    w3 = (1 - tx) *      ty ;
    w4 =      tx  *      ty ;
    % Compute interpolated pixel values
    N = w1*d_image(fy:fy+dy,fx:fx+dx) + w2*d_image(fy:fy+dy,cx:cx+dx) + ...
        w3*d_image(cy:cy+dy,fx:fx+dx) + w4*d_image(cy:cy+dy,cx:cx+dx);
    D = N >= d_C;
  end
  % Update the result matrix.
  v = 2^(i-1);
  result = result + v*D;
end

%Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end

if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result=hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result=result/sum(result);
    end
else
    %Otherwise return a matrix of unsigned integers
```

```
    if ((bins-1)<=intmax('uint8'))
        result=uint8(result);
    elseif ((bins-1)<=intmax('uint16'))
        result=uint16(result);
    else
        result=uint32(result);
    end
end

end
```

GETMAPPING.MAT

```
function mapping = getmapping(samples,mappingtype)
%Version 0.1.1
%Authors: Marko Heikkila and Timo Ahonen

%Changelog
%0.1.1 Changed output to be a structure
%Fixed a bug causing out of memory errors when generating rotation
%invariant mappings with high number of sampling points.
%Lauge Sorensen is acknowledged for spotting this problem.

table = 0:2^samples-1;
newMax  = 0; %number of patterns in the resulting LBP code
index   = 0;

if strcmp(mappingtype,'u2') %Uniform 2
  newMax = samples*(samples-1) + 3;
  for i = 0:2^samples-1
    j = bitset(bitshift(i,1,samples),1,bitget(i,samples)); %rotate left
    numt = sum(bitget(bitxor(i,j),1:samples)); %number of 1->0 and
                                               %0->1 transitions
                                               %in binary string
                                               %x is equal to the
                                               %number of 1-bits in
                                               %XOR(x,Rotate left(x))
    if numt <= 2
      table(i+1) = index;
      index = index + 1;
    else
      table(i+1) = newMax - 1;
    end
  end
end

if strcmp(mappingtype,'ri') %Rotation invariant
  tmpMap = zeros(2^samples,1) - 1;
```

```
  for i = 0:2^samples-1
    rm = i;
    r  = i;
    for j = 1:samples-1
      r = bitset(bitshift(r,1,samples),1,bitget(r,samples)); %rotate
                                                              %left
      if r < rm
        rm = r;
      end
    end
    if tmpMap(rm+1) < 0
      tmpMap(rm+1) = newMax;
      newMax = newMax + 1;
    end
    table(i+1) = tmpMap(rm+1);
  end
end

if strcmp(mappingtype,'riu2') %Uniform & Rotation invariant
  newMax = samples + 2;
  for i = 0:2^samples - 1
    j = bitset(bitshift(i,1,samples),1,bitget(i,samples)); %rotate left
    numt = sum(bitget(bitxor(i,j),1:samples));
    if numt <= 2
      table(i+1) = sum(bitget(i,1:samples));
    else
      table(i+1) = samples+1;
    end
  end
end

mapping.table=table;
mapping.samples=samples;
mapping.num=newMax;
```

CONT.MAP

```
function result = cont(varargin)
% Version: 0.1.0

% Check number of input arguments.
error(nargchk(1,5,nargin));

image=varargin{1};
d_image=double(image);

if nargin==1
    spoints=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
```

```
    neighbors=8;
    lims=0;
    mode='i';
end

if (nargin > 2) && (length(varargin{2}) == 1)
    radius=varargin{2};
    neighbors=varargin{3};
    spoints=zeros(neighbors,2);
    lims=0;
    mode='i';
    %Angle step.
    a = 2*pi/neighbors;

    for i = 1:neighbors
        spoints(i,1) = -radius*sin((i-1)*a);
        spoints(i,2) = radius*cos((i-1)*a);
    end

    if(nargin >= 4 && ~ischar(varargin{4}))
        lims=varargin{4};
    end

    if(nargin >= 4 && ischar(varargin{4}))
        mode=varargin{4};
    end

    if(nargin == 5)
        mode=varargin{5};
    end
end

if (nargin == 2) && ischar(varargin{2})
    mode=varargin{2};
    spoints=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
    neighbors=8;
    lims=0;
end

%Determine the dimensions of the input image.
[ysize xsize] = size(image);

miny=min(spoints(:,1));
maxy=max(spoints(:,1));
minx=min(spoints(:,2));
maxx=max(spoints(:,2));

%Block size, each LBP code is computed within a block of size bsizey*bsizex
```

```
bsizey=ceil(max(maxy,0))-floor(min(miny,0))+1;
bsizex=ceil(max(maxx,0))-floor(min(minx,0))+1;

%Coordinates of origin (0,0) in the block
origy=1-floor(min(miny,0));
origx=1-floor(min(minx,0));

%Minimum allowed size for the input image depends
%on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
    error('Too small input image. Should be at least (2*radius+1) x (2*radius+1)'
end

%Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;

%Compute the local contrast

for i = 1:neighbors
    y = spoints(i,1)+origy;
    x = spoints(i,2)+origx;
    %Calculate floors and ceils for the x and y.
    fy = floor(y); cy = ceil(y);
    fx = floor(x); cx = ceil(x);

    %Use double type images
    ty = y - fy;
    tx = x - fx;

    %Calculate the interpolation weights.
    w1 = (1 - tx) * (1 - ty);
    w2 =      tx  * (1 - ty);
    w3 = (1 - tx) *      ty ;
    w4 =      tx  *      ty ;
    %Compute interpolated pixel values
    N = w1*d_image(fy:fy+dy,fx:fx+dx) + w2*d_image(fy:fy+dy,cx:cx+dx) + ...
        w3*d_image(cy:cy+dy,fx:fx+dx) + w4*d_image(cy:cy+dy,cx:cx+dx);
    %Compute the variance using on-line algorithm
    %( http://en.wikipedia.org/wiki/Algorithms_for_calculating_variance#On-line_a
    if i == 1
        MEAN=zeros(size(N));
        DELTA=zeros(size(N));
        M2=zeros(size(N));
    end
    DELTA=N-MEAN;
    MEAN=MEAN+DELTA/i;
    M2=M2+DELTA.*(N-MEAN);
```

```
end

% Compute the variance matrix.
% Optional estimate for variance:
% VARIANCE_n=M2/neighbors;
result=M2/(neighbors-1);


% Quantize if LIMS is given
if lims
    [q r s]=size(result);
    quant_vector=q_(result(:),lims);
    result=reshape(quant_vector,q,r,s);
    if strcmp(mode,'h')
        % Return histogram
        result=hist(result, length(lims)-1);
    end

end

if strcmp(mode,'h') && ~lims
    % Return histogram
    %epoint = round(max(result(:)));
    result=hist(result(:),0:1:1e4);
end

end

function indx = q_(sig,partition)

[nRows, nCols] = size(sig);
indx = zeros(nRows, nCols);

for i = 1 : length(partition)
    indx = indx + (sig > partition(i));
end

end
```

5.2. Funciones Matlab para el cálculo de PCA

Se incluyen el código de las funciones Matlab para el cálculo de PCA.

```
function [signals,PC,V] = pca1(data)

% data-MxN matrices de entrada M dimensiones, N muestras.
signals Matriz M por N proyeccion de los datos.
PC - cada columna es un PC
```

```
V- Matriz Mx1 de varianzas.
[M,N] = size(data)
%Resta la media a cada dimension
mn = mean(data,2)
data = data - repmat(mn,1,N)
%Calculo de la Matriz de Covarianza
 covariance = 1 / (N-1) * data * data'
%En cuentra los eigevectores y eigevalores
[PC, V] = eig(covariance);
%Extraer la diagonal de la matriz como un vector
V = diag(V);
%Ordenar la varianzas en orden decreciente.
[junk, rindices] = sort(-1*V);
V = V(rindices); PC = PC(:,rindices);
%Prpyectar los datos originales
signals = PC' * data;
```

This second version follows section 6 computing PCA through SVD.

```
function [signals,PC,V] = pca2(data)
% % % % %
PCA2: Perform PCA using SVD. data - MxN matrix of input data (M dimensions, N tri
signals - MxN matrix of projected data PC - each column is a PC

%       V - Mx1 matrix of variances
[M,N] = size(data);
%subtract off the mean for each dimension
mn = mean(data,2);
 data = data - repmat(mn,1,N);
%construct the matrix Y
Y = data'/ sqrt(N-1);
%SVD does it all
[u,S,PC] = svd(Y);
%calculate the variances
S = diag(S);
 V = S .* S;
%project the original data
signals = PC' * data;
```

Bibliografía

[ABR64] M.A. Aizerman, E.M. Braverman, and L.I Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[AHP06] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12), December 2006.

[BGV92a] B.E. Boser, I.M. Guyon, and V Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, 1992.

[BGV92b] B.E. Boser, I.M. Guyon, and V Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory, Pittsburgh*, 1992.

[BK77] J.R. Bunch and L Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of computation*, 31(137):163–179, 1977.

[BK80] J.R. Bunch and L Kaufman. A computational method for the indefinite quadratic programming problem. *Linear Algebra and its Applications*, 34:341–370, 1980.

[BS97] C.J.C. Burges and B Schölkopf. Improving the accuracy and speed of support vector learning machines. *Advances in Neural Information Processing Systems*, 9:375–381, 1997.

[Bur96] C.J.C Burges. Simplified support vector decision rules. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 71–77. Morgan Kaufman, 1996.

[Bur98] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. 1998.

[Cor95] C. & Vapnik Cortes. Support vector networks. *Machine Lear*, 20:273–297, 1995.

[CS11] Modesto F. Castrillón Santana. *Apuntes de Biomtríacomputacional. Reconocimiento Facial*. SIANI, 2011.

[CSDSHTGA07] M. Castrillón Santana, O. Déniz Suárez, M. Hernández Tejera, and C. Guerra Artal. ENCARA2: Real-time detection of multiple faces at different resolutions in video streams. *Journal of Visual Communication and Image Representation*, pages 130–140, April 2007.

[Fix51] Hodges Fix. Discriminatory analysis, nonparametric estimation: consistency properties. Technical report, 1951.

[Fle87] R Fletcher. *Practical Methods of Optimization.* 1987.

[Hal67] P.R Halmos. *A Hilbert Space Problem Book*. D. Van Nostrand Company, Inc, 1967.

[HP09] Abdenour Hadid and Matti Pietikäinen. Combining appearance and motion for face and gender recognition from videos. *Pattern Recognition*, 42(11):2818–2827, November 2009.

[Joa97] T Joachims. Text categorization with support vector machines. Technical report LS VIII 23, University of Dortmund,, 1997.

[Kin] Página de wikipedia con información de kinect.

[LJ05] S.Z. Li and A.K. Jain. *Handbook of face recognition*. Springer, 2005.

[McC83] G.P McCormick. *Non Linear Programming: Theory, Algorithms and Applications.* John Wiley and Sons, Inc., 1983.

[MeT91] J.J. Mor˜e and G Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM J. Optimization*, 1:93–113, 1991.

[MFCG03] Castrillón Santana M., Hernández Tejera F.M., and J. Cabrera Gámez. ENCARA: real-time detection of frontal faces. In *International Conference on Image Processing*, Barcelona, Spain, September 2003.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[MTZ00] Pietikäinen M., Ojala T., and Xu Z. Rotation-invariant texture classification using feature distributions. 2000.

[OPH96] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29:51–59, 1996.

[OPM02] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[PFV92] W.H. Press, S.A. Flannery, B.P.and Teukolsky, and W.T Vettering. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 2 edition, 1992.

[Pie10] Matti Pietikäinen. Local binary pattern- scholpedia, 2010.

[SBV96] B. Schölkopf, C. Burges, and V Vapnik. *Incorporating invariances in support vector learning machines.*, volume 1112 of *Springer Lecture Notes in Computer Science*. Artificial Neural Networks — ICANN'96, Berlin, 1996.

[Shl09] Jonathon Shlens. A tutorial on principal component analysis. 2009.

[SRSL07] Li SZ., Chu RF., Liao SC., and Zhang L. Illumination invariant face recognition using near-infrared images. 2007.

[SSSV98] B. Schölkopf, P. Simard, A. Smola, and V Vapnik. *Prior knowledge in support vector kernels.* Advances in Neural Information Processing Systems 10, Cambridge, MA,. MIT Press., 1998.

[Uni] University of OULU. *LBP METHODOLOGY*.

[Van94a] R.J Vanderbei. An interior point code for quadratic programming. Technical report, Program in Statistics & Operations Research, Princeton Universit, 1994.

[Van94b] R.J Vanderbei. Interior point methods : Algorithms and formulations. *ORSA J. Computing*, 6:32–34, 1994.

[Vap95] V Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[web] Web del departament of computer science and engineering. www.cse.oulu.fi/mvg/downloads/lbpmatlab.

[WFH10] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann publishers, 3 edition, 2010.

[WSW+05] Zhang W., Shan S., Gao W., Chen Z., and Zhang H. Local gabor binary pattern histogram sequence (lgbphs): a novel non-statistical model for face representation and recognition. Proc. Tenth IEEE International Conference on Computer Vision (ICCV 05) Vol 1, 786-791, 2005.

[ZP07] Guoying Zhao and Matti Pietikäinen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):915–928, 2007.