Práctica 1B: Reconocimiento de letras y dígitos mediante el Análisis de Componentes Principales (PCA)

Ingeniería del Conocimiento

Curso académico 2023-2024

Eduardo Ezponda Igea

ÍNDICE:

INTRODUCCIÓN:

El problema para resolver trata de reconocer un nuevo dígito o letra del abecedario a partir de un clasificador previamente entrenado. Para ello, se utiliza el algoritmo PCA (Análisis de las Componentes Principales). Se utiliza un proceso de aprendizaje para clasificar nuevos ejemplos en el futuro a partir de unos ejemplos de entrenamiento. Además, se intenta reducir la información manteniendo al máximo las propiedades.

EXPLICACIÓN DEL PROBLEMA:

Se van a utilizar dos conjuntos de datos. En primer lugar, se encuentra MNIST, que es un conjunto de datos de dígitos manuscritos en forma de imagen. Además, se tiene 10 clases, cada una perteneciente a cada dígito, y se tiene una etiqueta para cada imagen, representando el dígito que representa. A través de la etiqueta, se comprobará si el clasificador acierta o falla en su reconocimiento del dígito. El otro conjunto de datos es EMNIST, que es una extensión de EMNIST, añadiendo imágenes de letras mayúsculas y minúsculas escritas a mano. Únicamente se utilizará el conjunto de mayúsculas, formado por 26 letras.

Para MNIST dispondremos de 60000 imágenes de entrenamiento y de 10000 imágenes de testeo, mientras que para EMNIST dispondremos de 1000 imágenes de entrenamiento y de 106 imágenes de testeo (4 por letra mayúscula). Se utilizarán distintas distribuciones entre imágenes de testeo y de entrenamiento para comprobar cuál es la proporción que adquiere un mayor porcentaje de acierto. Sin embargo, la distribución más común entre entrenamiento y testeo es de 80:20, respectivamente. Por último, se comprobará si un mayor número de imágenes implicaría un mayor o menor porcentaje de acierto, utilizando la distribución 160:40 (el doble de imágenes que 80:20).

Para la resolución de la técnica PCA, se divide el código en distintas fases. Lo único que cambia entre MNIST y EMNIST es la forma en la que se cargan los datos, y la distribución que se utiliza para cada uno de ellos (80:20 y 160:40 en MNIST, mientras que 1000:104 en EMNIST).

En primer lugar, se cogen las imágenes de los dígitos con sus respectivas etiquetas de los archivos *idx3-ubyte* para después cargarlos en *loadData* tras su previo procesamiento en *processImagesMNIST* y *processLabelsMNIST*. Se obtiene el número determinado de ejemplos por clase para testear y para entrenar con la

función *get_n_samples_per_class*. Por último, se convierten los valores de las imágenes en números y se filtran las clases a través de *filter_classes*. A *loadData* se le pasará como argumento una tabla con las clases con las que se quiere realizar la comparación, además de la distribución utilizada.

Para el conjunto EMNIST, se cargarán los datos en *loadDataEMNIST* utilizando un fichero de datos de MATLAB denominado *emnist-letters*.

En segundo lugar, se realiza un cambio de base y se obtienen las coordenadas de los datos originales en la nueva base. Estas coordenadas serán los nuevos atributos del problema. El cambio de base de debe a querer representar de una forma óptima las características del problema ante posibles atributos redundantes o ruido.

En la función *aprendeBase* se consigue una matriz *nuevaBase* de dimensiones NumeroPixeles x NumeroFotos formada por vectores columna. Para ello, se crea una matriz R formada por vectores imágenes como columnas. Cada vector tendrá el tamaño de su número de píxeles. Se calcula la media por filas de R y se calculan los vectores con media nula para añadirlos a la matriz A. Se calcula la matriz de correlación C. Por último, se obtienen los vectores y valores propios de la matriz cuadrada C para insertar los vectores en la nueva base. Se necesitará realizar una previa conversión de los vectores para modificar su tamaño de NumeroFotos componentes a NumeroPixeles componentes.

En tercer lugar, se crean los prototipos para cada clase. Se necesita previamente proyectar los ejemplos de las imágenes en la nueva Base para luego ir cogiendo los ejemplos de cada clase y calcular su media correspondiente por clase. Cada vector medio de cada clase se añadirá como columna a una matriz prototipos. El objetivo de calcular los prototipos es obtener una nueva imagen que será la imagen media de todas las de esa clase.

Para finalizar, se clasificarán las imágenes de testeo no utilizadas anteriormente para comprobar si el aprendizaje es efectivo. El primer paso será modificar la imagen de una matriz a un vector con la función *reshape*. Se proyectará la imagen de testeo a la nueva base, y se calculará la distancia entre la imagen y cada uno de los prototipos. Por último, se cogerá el mínimo de las distancias que será el prototipo de la clase que más parecidos tenga. Será necesario calcular posteriormente si ha acertado la clase con la respectiva etiqueta de la imagen que se ha utilizado.

RESULTADOS:

A continuación, se va a explicar cada una de las pruebas que se van a realizar. En cada una de ellas se obtendrá su determinado porcentaje de aciertos.

Primero, se realizan clasificaciones binarias MNIST entre las clases “1” y “7”, “2” y “7”, y, por último, “4” y “9”. Se cogen 80 imágenes para entrenar y 20 imágenes para testear. El resultado del porcentaje de aciertos es 100%, 90% y 90%, respectivamente.

Como se puede comprobar a simple vista, los números 2 y 7, y 4 y 9 tienen un gran parecido a la hora de escribirse. La forma es muy similar, y eso puede provocar mayor incertidumbre y dificultad a la hora de analizar la clase resultante. Consecuentemente, la tasa de aciertos será menor. En cambio, 1 y 7 son números mucho más diferentes, lo que provoca una tasa de aciertos mayor. Además, al ser una clasificación binaria y sólo tener que distinguir entre 2 elementos, el aprendizaje será más sencillo y óptimo, a pesar de que puedan ser más o menos parecidos los dígitos. Por último, al ser binario y utilizar una proporción 80:20, el proceso de aprendizaje tendrá muchos más ejemplos para aprender y poder distinguir entre pequeñas diferencias que en el caso de utilizar una clasificación multiclase no se podría dar.

Segundo, se realiza una clasificación multiclase MNIST entre las 10 clases del dataset correspondientes a cada dígito (0,1,2,3,4,5,6,7,8,9). Se utilizan las distribuciones 80:20 y 160:40. En este caso, la tasa de aciertos para las distribuciones anteriores es 70% y 63.125%, respectivamente. Al tener que reconocer una imagen entre 10 clases, el clasificador tenderá a realizar muchísimos más errores porque cuanto mayor sea el número de clases, mayor parecido habrá entre distintas clases y, por tanto, una menor tasa de acierto.

Al aplicar una distribución con el doble de datos de entrenamiento y testeo (160:40), el porcentaje de acierto es menor. Este menor porcentaje puede ser debido a que al incluir una mayor cantidad de imágenes, el clasificador obtenga un análisis mayor de los atributos por clase, haciendo que le sea más difícil decantarse entre una clase y otra, y, como consecuencia, realizar un fallo. Además, se puede deber a que justo las imágenes que escoja el clasificador de forma aleatoria sean mayormente conflictivas que con la distribución 80:20, y pongan más complicado el reconocimiento del dígito.

Finalmente, se realiza una clasificación multiclase EMNIST entre las 26 clases correspondientes a las 26 letras mayúsculas. Se utilizan 1000 imágenes de entrenamiento y 104 imágenes para testear. Al utilizar dicha distribución, el porcentaje de acierto es 48.5577%. Como se puede observar, cada dos imágenes, el clasificador aproximadamente fallará una, y acertará otra. Por lo tanto, el porcentaje de fallo es muy alto. Este porcentaje se debe a que el número de letras mayúsculas o clases es muy alto, haciendo confundir con facilidad al clasificador. Si se redujera el número de clases, lo más probable sería que el porcentaje de aciertos aumente al tener que reconocer entre menor cantidad de atributos.

Si se modifica el número de imágenes a 2000, 5000 o 10000, el porcentaje de aciertos es de 49.085%, 47.1154% y 50%. Como se puede observar, en 2 de los 3 casos, al aumentar la cantidad de imágenes para entrenar al clasificador, el porcentaje de aciertos aumenta. Sobre todo, aumenta la tasa de aciertos al cambiar de 1000 a 10000 imágenes al ser una amplia diferencia en comparación con 2000 o 5000 imágenes. En el segundo caso, la tasa de aciertos disminuye, y esto se puede deber a que elija aleatoriamente imágenes que le haga confundirse de forma más fácil al tener posibles similitudes. A pesar de todo, la diferencia al aumentar el número de imágenes en el clasificador es mínima debido a que lo más importante en este caso es el número de clases que tenga el clasificador. Y, como en este caso son 26 clases, que es un número muy alto, el clasificador tenderá a realizar errores y el proceso de aprendizaje será óptimo.

CONCLUSIÓN FINAL:

Como conclusión final podemos habernos dado cuenta durante el proceso del algoritmo PCA que hay tres factores que afectan en gran medida en el proceso.

En primer lugar, las imágenes. Siempre el clasificador tenderá a realizar más errores cuando las clases utilizadas tengan mayores similitudes. Esto se debe a que el proceso de aprendizaje será defectuoso al no aprender correctamente, y asociar atributos similares a clases distintas produciendo errores en el reconocimiento final.

En segundo lugar, afecta la cantidad de clases que haya que analizar. Cuánta mayor cantidad de clases haya, más complejo será el proceso de aprendizaje, y, por lo tanto, mayores errores cometerá el clasificador. Si las clases son muy diferentes, puede que no afecte que haya mayor cantidad de clases, pero en general, no ocurrirá tal cosa.

En tercer lugar, tenemos la cantidad de imágenes que se utilicen en la distribución para entrenar al clasificador y para posteriormente testearlo. En general, cuanta mayor cantidad de imágenes mejor para enseñar al clasificador el máximo número de atributos. El problema podría darse al usar imágenes de poca calidad que pudieran resultar en malinterpretaciones por parte del clasificador al haber entorpecido el proceso de aprendizaje.

Teniendo en cuenta estos tres factores, y aportando imágenes que no puedan confundir al clasificador, se podría conseguir un buen porcentaje de acierto dependiendo del problema y de las clases correspondientes.