Sistemas de Percepción

Práctica 4

UNIVERSIDAD D SEVILLA

Realizado por:

Fernando Román Hidalgo

Andrés Martínez Márquez

Índice

1. Introducción

En esta práctica se ha trabajado el reconocimiento de imágenes a través de un clasificador Bayesiano. Este clasificador estará basado en un juego de imágenes, el *MNIST Dataset*, que se usará tanto para el entrenamiento del clasificador como para el test final.

2. Realización de la práctica

2.1. Versión básica

En cuanto a la versión básica, se nos pide construir el propio clasificador Bayesiano, y comprobar que funciona correctamente.

Para ello, tomaremos las imágenes de la base de datos proporcionada y las cargaremos a nuestro programa. Cada imagen viene acompañada de una etiqueta que indica el número que representa.

El primer paso a realizar será separar las imágenes en clases para su posterior procesamiento. Para ello se convertirán en vectores columna de 400x1 y se agruparán en sus distintas clases creando 10 matrices de 400xN siendo N el número de veces que aparece cada número en el dataset.

A continuación hallaremos los valores estadísticos que usara el clasificador Bayesiano, como las medias, las matrices de covarianza y las inversas de estas últimas. Una vez tengamos las matrices de covarianza, se pasará a calcular la función de decisión inicial con su determinante, teniendo cuidado de que este no sea 0.

Cuando ya se tienen esos parámetros estadísticos calculados se pasa a clasificar los números. Para ello se va imagen por imagen del dataset, primero pasándola a un vector columna para poder calcular la distancia de Mahalanobis de la imagen que se está analizando al número actual. Está acción se realiza con todos los números del 0 al 9, y el que obtenga una función de decisión máxima será la predicción para ese número.

Cuando se ha procesado una imagen comparándola del 0 al 9, se suma un acierto o un fallo y se repite el proceso realizado con la siguiente imagen, así hasta terminar el dataset.

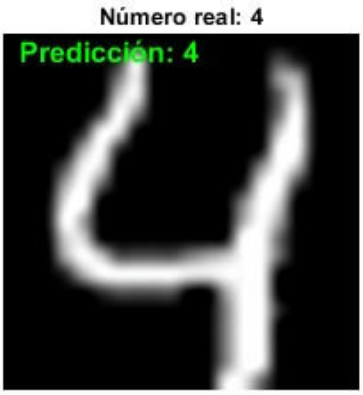

Figura 1: Imagen del dataset clasificada correctamente

Figura 2: Imagen del dataset clasificada erróneamente

Algo a tener en cuenta es que las matrices de covarianza tienden a ser singulares, y para regularizarlas se les tiene que sumar un término lambda para que de esta forma no den problemas al calcular su inversa. El lambda elegido en este caso ha sido 0.1, puesto que ha proporcionado las mejores tasas de acierto.

```
Número de train: 60000, Fallos: 2470, Acierto: 95.8833%
Número de test: 10000, Fallos: 445, Acierto: 95.55%
```

Figura 3: Resultado obtenido con la versión básica del programa

2.2. Versión avanzada

Para la versión avanzada, se nos pide reducir el tamaño del vector de características para así conseguir un clasificador más eficiente y ligero. Esto se consigue usando la estrategia de transformación discriminante.

El proceso seguido es inicialmente, muy similar al descrito en la versión básica, ya que se necesita de la misma manera procesar las imágenes en clases.

El siguiente paso seguido es el de calcular las medias, tanto de cada clase como la total, y calcular la dispersión dentro de las clases con respecto a la media de cada clase (S_w) y la dispersión entre las clases con respecto a la media global (S_b).

A continuación, se resuelve el problema de valores propios para encontrar las direcciones que maximizan la separación entre clases. Esto nos permite separar los diferentes autovalores, quedándonos solo con los de mayor capacidad discriminante, ordenándolos para crear así la matriz de proyección.

Acto seguido, se han proyectado los datos tanto de test como de entrenamiento al nuevo espacio reducido, permitiéndonos así realizar el clasificador Bayesiano con estos datos ya reducidos.

En cuanto al clasificador, se han calculado en primer lugar la media y la covarianza para cada clase en el espacio reducido, y se han guardado las matrices de covarianza, sus inversas, y la función de decisión f inicial.

Una vez realizado el paso anterior, ya se puede proceder a realizar la clasificación con las muestras, tanto de *test* como de *train*.

Se ha ido probando el número adecuado al que reducir el vector de características para realizar de manera adecuada y con una precisión correcta la clasificación, y tras varias pruebas, se ha encontrado que con un tamaño de 200 se consigue un resultado muy óptimo, y asegurando una tasa de acierto cercana a 94%.

```
Número de test: 10000, Fallos: 612, Acierto: 93.88%
Número de train: 60000, Fallos: 3116, Acierto: 94.8067%
Tasa de éxito: [train, test] = [94.8067% ,93.88%]
```

Figura 4: Resultado obtenido con el conjunto de características de tamaño 200

De una forma similar, si se quisiera obtener un resultado más eficiente, aunque con una ligera menor precisión, se puede tomar el tamaño proporcionado por la práctica, 151, que nos deja con una precisión cercana al 90%.

```
Número de test: 10000, Fallos: 987, Acierto: 90.13%
Número de train: 60000, Fallos: 5225, Acierto: 91.2917%
Tasa de éxito: [train, test] = [91.2917% ,90.13%]
```

Figura 5: Resultado obtenido con el conjunto de características de tamaño 151

3. Instrucciones para ver el programa

El programa básico se encuentra en el fichero de Matlab “Practica4Básica”, y el programa avanzado se encuentra en el fichero de Matlab “Practica4Avanzada”. Ambos contienen comentarios que facilitan el entendimiento de la práctica.

4. Conclusión

En esta práctica se ha aprendido a realizar un clasificador Bayesiano que permita reconocer las imágenes de un dataset inicial. Además, con la realización de la versión avanzada se ha comprendido como mejorar este clasificador, convirtiéndolo así en uno más eficiente. Gracias a esta práctica, hemos reforzado nuestros conocimientos adquiridos anteriormente en las clases de teoría, ampliando así nuestro aprendizaje.