# Jose Antonio Mira García

## Introducción

Se evaluarán los resultados de tres métodos de clasificación (la mixtura de gaussianas no pude evaluarla).

El objetivo de este proyecto es investigar qué método es más apropiado para la clasificación de la colección de dígitos manuscritos MNIST.

# Mixturas de gaussianas

## mixgaussian.m Ej 2.1

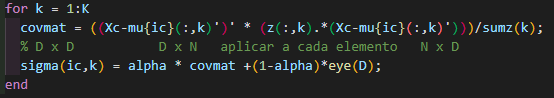
Se calcula la probabilidad a priori para cada índice de la clase y se normaliza respecto al número de clases. No se utiliza el operador ‘.’ porque octave aplica la normalización a toda la matriz.



La media para cada índice se calcula transponiendo la muestra de la clase c para que las dimensiones se ajusten a las requeridas para multiplicarlo por zk, y a cada elemento se le divide entre el sumatorio de zk para normalizar.

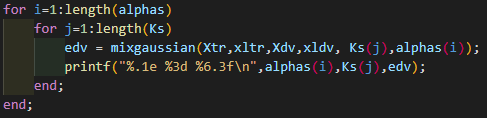


Se aplica la matriz de covarianzas a cada componente k. Finalmente se aplica un suavizado.



## mixgaussian-exp.m Ej 2.2

Se recorren tanto las alphas como las Ks y se guardan en edv. Para posteriormente imprimir la tasa de error.



## pca+mixgaussian-exp.m Ej 2.3

Aplicamos PCA sobre el conjunto de entrenamiento. Así obtenemos la matriz de proyección completa (W) y las medias (m).

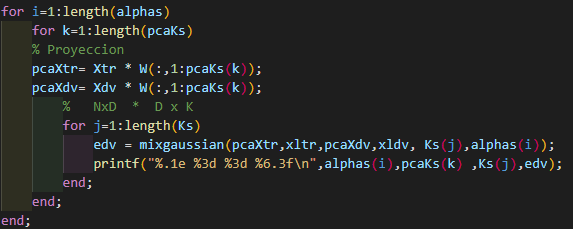


A cada dato se le resta su media (en ambos conjuntos).

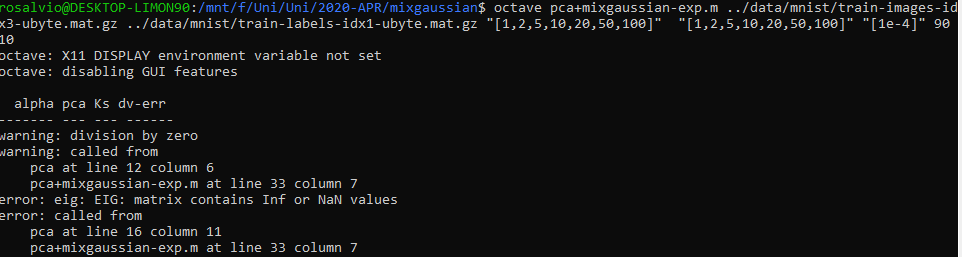


Se calculan los nuevos conjuntos en función de la proyección en PCA y luego se pasa como parámetro a mixgaussian.

A diferencia de mixgaussian-exp.m aquí se recorren también todas las dimensiones del PCA.



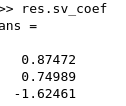
No he podido continuar con el experimento debido a que no me funciona el pca.



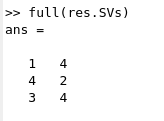
# Máquinas de vectores de soporte

## Ej 3.2

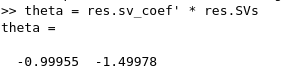
### Linealmente separable



Multiplicadores de Lagrange:



Vectores de soporte:

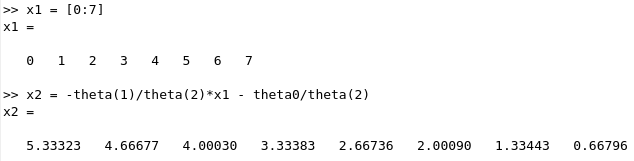


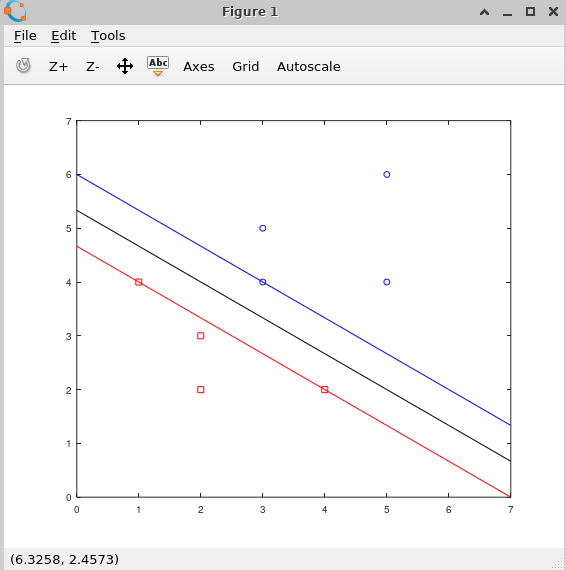
Vector de pesos:





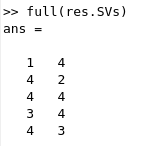
Margen:

Recta de separación: 

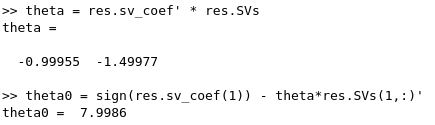


### No separable

Multiplicadores de Lagrange:

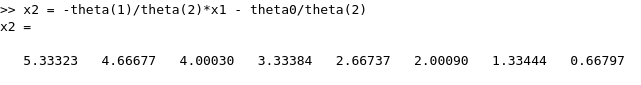


Vectores de soporte:

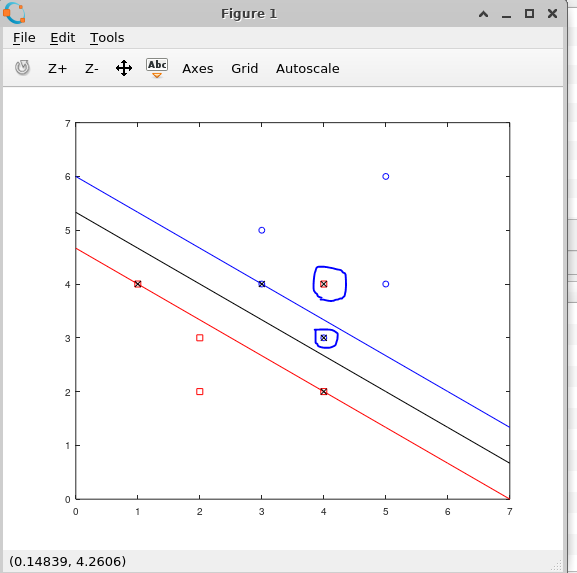


Vector de pesos:

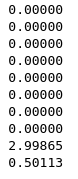
Margen:



Recta de separación:



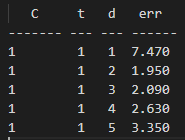
Redondeados los vectores de soporte erróneos



Tolerancia de margen:

## Ej 3.3

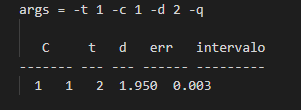
En mi caso sólo pude ejecutar el kernel polinomial por tiempo de cómputo.



Mediante la salida obtenida concluí que los mejores resultados se obtenían mediante un polinomio de segundo grado.

A partir del segundo grado se tiende a empeorar probablemente debido al sobreentrenamiento.

## Ej 3.4



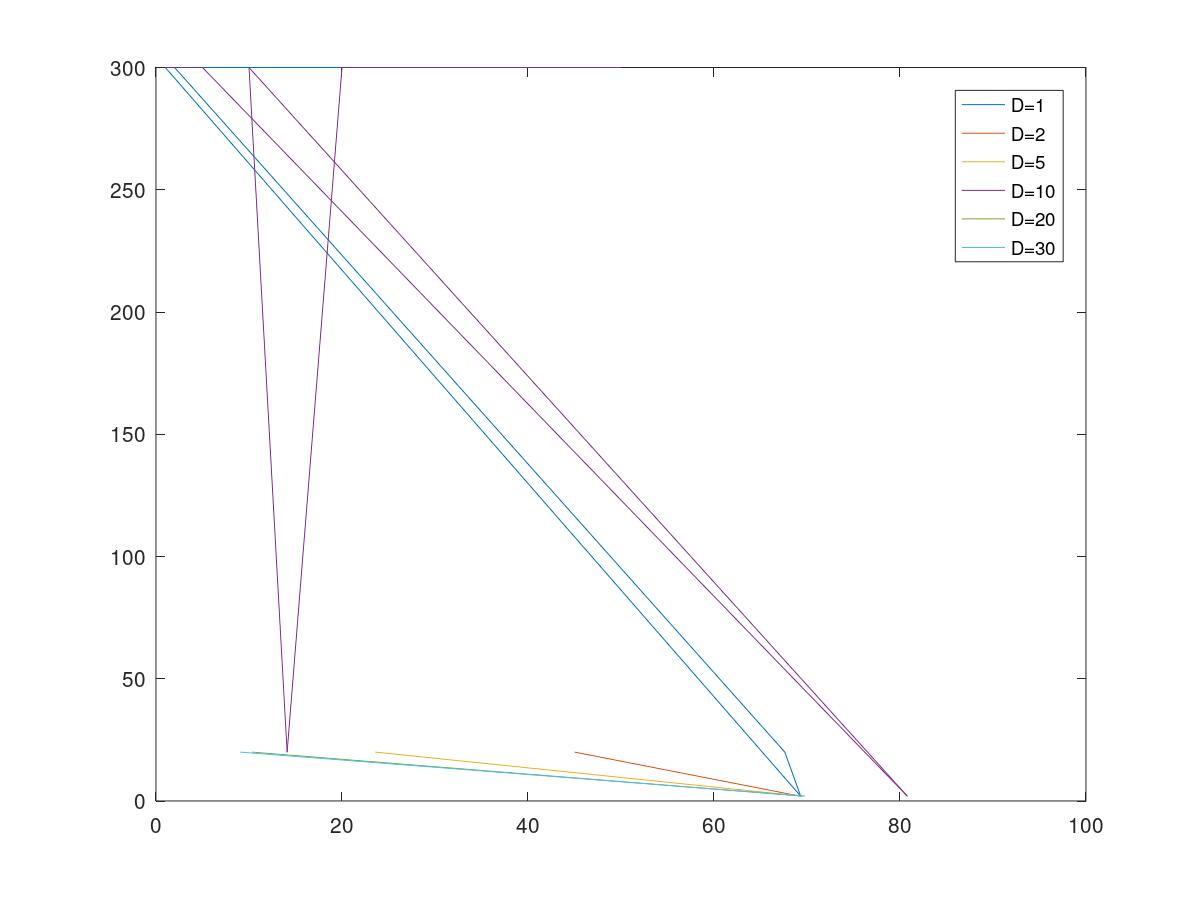
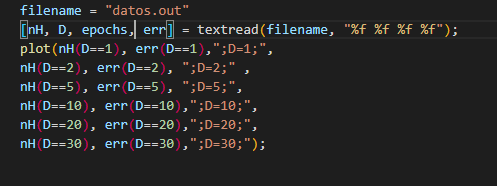
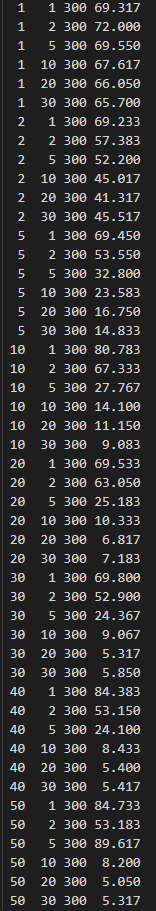
En la documentación de MNIST se alcanza un error mínimo de 1.1 mediante un kernel polinomial de grado 4, sin embargo, esto es porque se ha utilizado deskewing como preproceso de los datos.

# Redes neuronales multicapa

## Ej 4.5

Experimento realizado con 40% de test y 10% de validación. El mínimo se consigue con 20 dimensiones de pca y 50 neuronas ocultas. Error = 5.05

NHiddens D Epochs Error



## Ej 4.5



Se consigue un error similar al aplicar una red neuronal de 2 capas y 300 neuronas ocultas (4.7), sin embargo se consiguen mejores resultados mediante SVMs, también probablemente por sobreentrenamiento como hemos visto en el caso de SVMs con polinomios de grado alto.

## Conclusión

Para la tarea MNIST resulta mucho más eficiente usar SVMs, probablemente podría haber alcanzado incluso mejores resultados si hubiera podido computar otros tipos de kernel.