Andrés Marín Galán

Aprendizaje automático
Proyecto:

En este proyecto se realizará un estudio de los resultados de error en los algoritmos de mixtura de gaussianas, vectores soporte y redes neuronales para clasificar imágenes pertenecientes al conjunto de datos de MNIST donde se pueden consultar los datos de error en http://yann.lecun.com/exdb/mnist/, además del efecto del algoritmo de P.C.A. sobre el error del clasificador de algunos de estos clasificadores.

Mixtura de gaussianas

Emplearemos un conjunto de distribuciones gaussianas del mismo tipo que las que se vieron en la asignatura de Percepción, pero usaremos varias para clasificar los datos mediante la técnica de aprendizaje no supervisado, creando tantas gaussianas como se como se indique en la componente k (además de los parámetros propios de la gaussiana como el alpha).

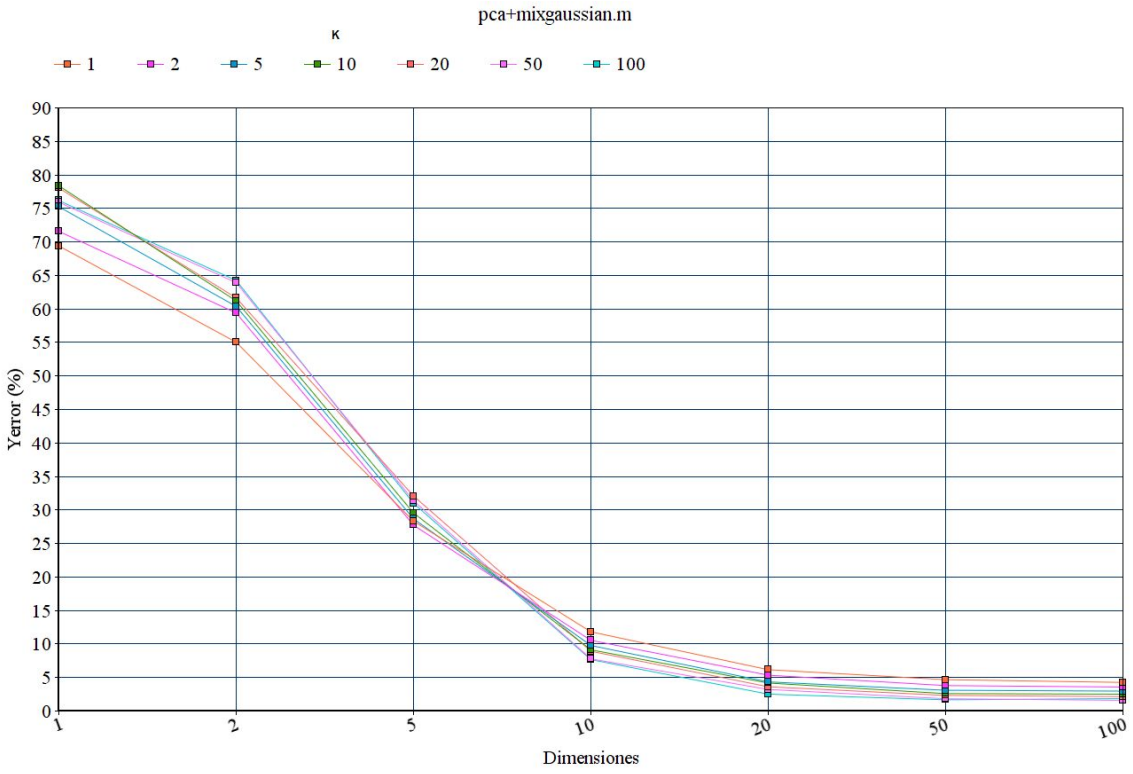
El código se puede encontrar en el archivo **mixgaussian.m** adjunto a este documento. (Ejercicio 2.1.)

Aplicaremos primero PCA y después emplearemos mixgaussian.m para estimar el error mediante el alpha que dio error más bajo en Percepción que es el alpha 1e-4 pero variando K y las dimensiones como indica el ejercicio 2.3.

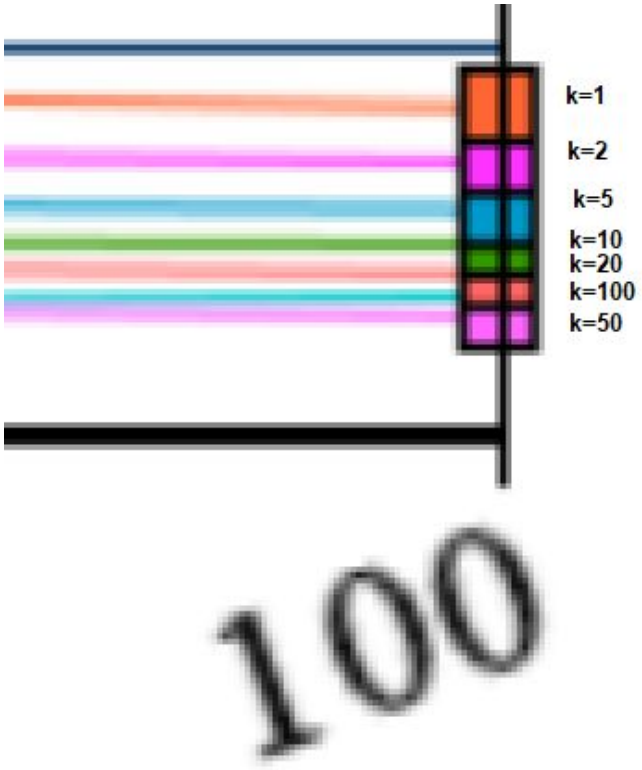
El código de esto reside en el archivo **pca+mixgaussian-exp.m** adjunto a este documento.

A continuación veremos una gráfica con los resultados obtenidos con la ejecución de:

./pca+mixgaussian-exp.m train-images-idx3-ubyte.mat.gz train-labels-idx1-ubyte.mat.gz "[1 2 5 10 20 50 100]" "[1e-4]" "[1 2 5 10 20 50 100]" 90 10



Como podemos apreciar en la gráfica el mejor valor (de mínimo error) de uso de dimensiones en PCA es 100 aunque no logra apreciarse a simple vista que valor de K vamos a ampliar la imagen para verlo.



Como se puede ver en la anterior imagen el mejor valor de k es 50 y no 100 como parecía deducirse así por tanto con este estudio podemos decir que los mejores parámetros para este problema son k=50, dim=100 y alpha=1e-4 con un valor de error de **1.60**%

Vamos a probar estos parámetros para clasificar otro conjunto de muestras para ver cuánto error se obtiene.

El código de esta prueba se puede encontrar en el archivo **pca+mixgaussian-eva.m** adjunto en este documento (Ejercicio 2.4.)

Mediante la ejecución de la orden:

./pca+mixgaussian-eva.m train-images-idx3-ubyte.mat.gz train-labels-idx1-ubyte.mat.gz t10k-images-idx3-ubyte.mat.gz t10k-labels-idx1-ubyte.mat.gz 50 1e-4 100

Se obtiene un error de **1.92% (0.0191 sobre 1)**
Con intervalo de confianza del 95% que es 1.96*sqrt((0.0191*(1-0.0191))/N) como N son 10000 da un valor de 0.0027 (0.27%) [1.92-0.27,1.92+0.27]
Probabilidad de error: 1.92%
Con intervalo de confianza: [1.65,2.19]

Valor obtenido con 1 gaussiano (supervisado) en Percepción 4.11% / Por MNIST 3.3%

Podemos ver una clara mejoría en el error, sorprendente por el hecho de que esta parte no sea aprendizaje supervisado lo que debería dificultar la clasificación. Queda plasmada la eficacia de estos sistemas que trabajan con varias gaussianas.

Valor obtenido por MNIST la gaussiana kernel con svm es 1.4%
Aunque está hecha con la técnica de vectores soporte emplea el kernel gaussiano por lo tanto es lo más cercano tanto teóricamente como experimentalmente dado que la diferencia de error es solo de un 0.42%

Vectores Soporte

Emplearemos los Multiplicadores de Lagrange para obtener unos vectores sobre los cuales obtenemos theta y theta0 para así poder clasificar las muestras.

3.2.
El código a ejecutar dentro de la consola en una carpeta con los archivos **svmpredict.mex**, **svmtrain.mex**, **tr.dat**, **trlabels.dat**, **trSep.dat** y **trSeplabls.dat** esta en el archivo **script.txt.**

3.2.a (Separables)
Obtenemos así:

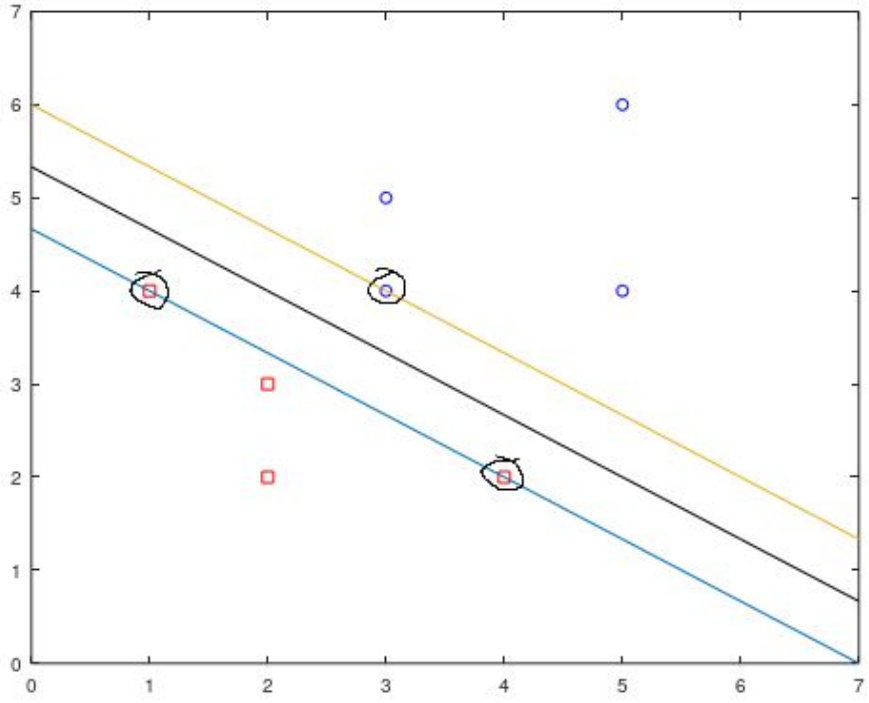
a) sv_coef= Multiplicadores de Lagrange
   [0.87472 índice 1
   0.74989 índice 4
   -1.62461] índice 5

b) SVs= Vectores soporte
   [1, 4
   4, 2
   3, 4]

c) theta(t) = (-1,-1.5) Vector de pesos
   theta0(t0) = 8 Humbral

d) Margen(m) = 1.1097

Representación con vectores soporte erróneos marcados



3.2.b (No separables) C=1000
Obtenemos así:

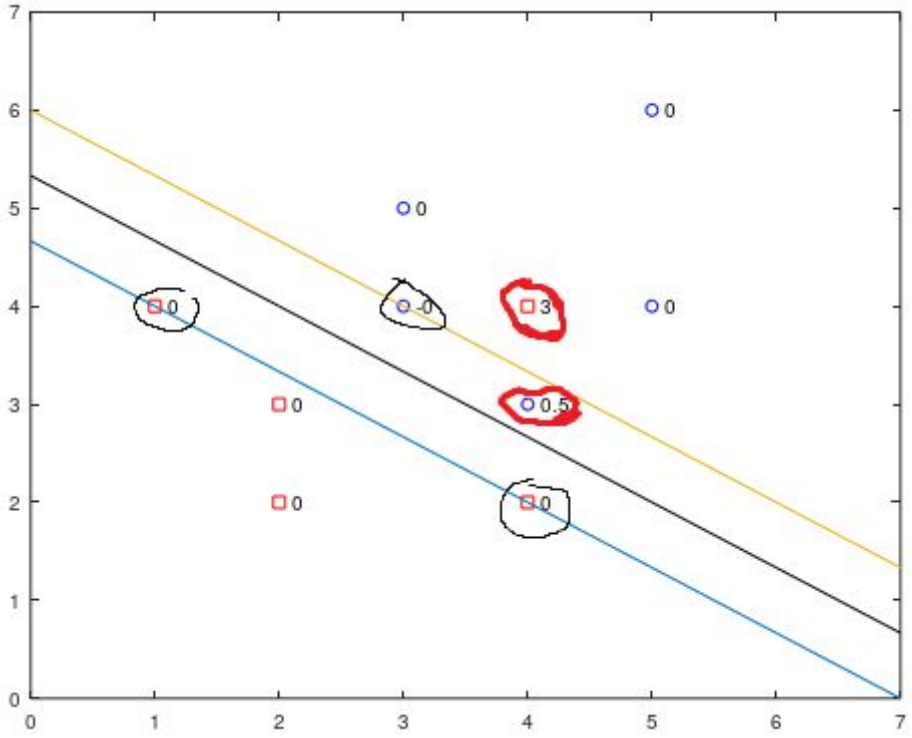
a) sv_coef= Multiplicadores de Lagrange
   [250.87 índice 1
   500.75 índice 4
   1000.00 índice 9
   -751.62 índice 5
   -1000.00] índice 10

b) Svs= Vectores soporte
   [1,4
   4,3
   4,4
   2,4
   4,3]

c) theta(t) = (-1,-1.5) Vector de pesos
   theta0(t0) = 8 Humbral

d) Margen(m) = 1.1097

Representación con vectores soporte erróneos marcados



3.2.b (No separables) C=1

Obtenemos así:

e) sv_coef= Multiplicadores de Lagrange

[0.65306 índice 1
0.73472 índice 4
1.00000 índice 9
-1.00000 índice 5
-0.38778 índice 6
-1.00000] índice 10

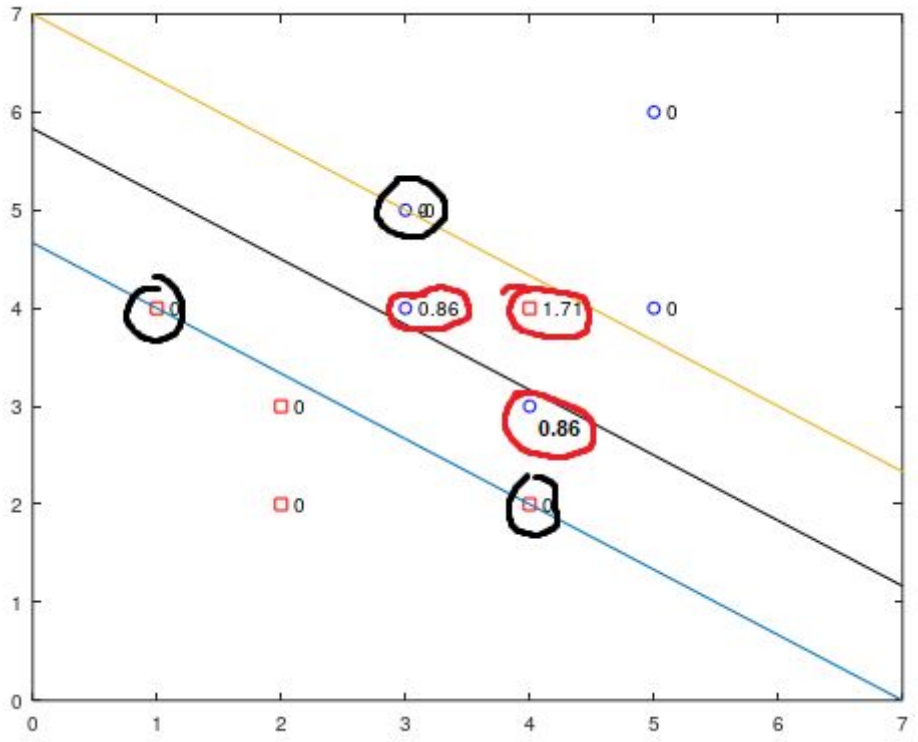
f) Svs= Vectores soporte

[1,4
4,3
3,4
4,2
4,4
5,3]

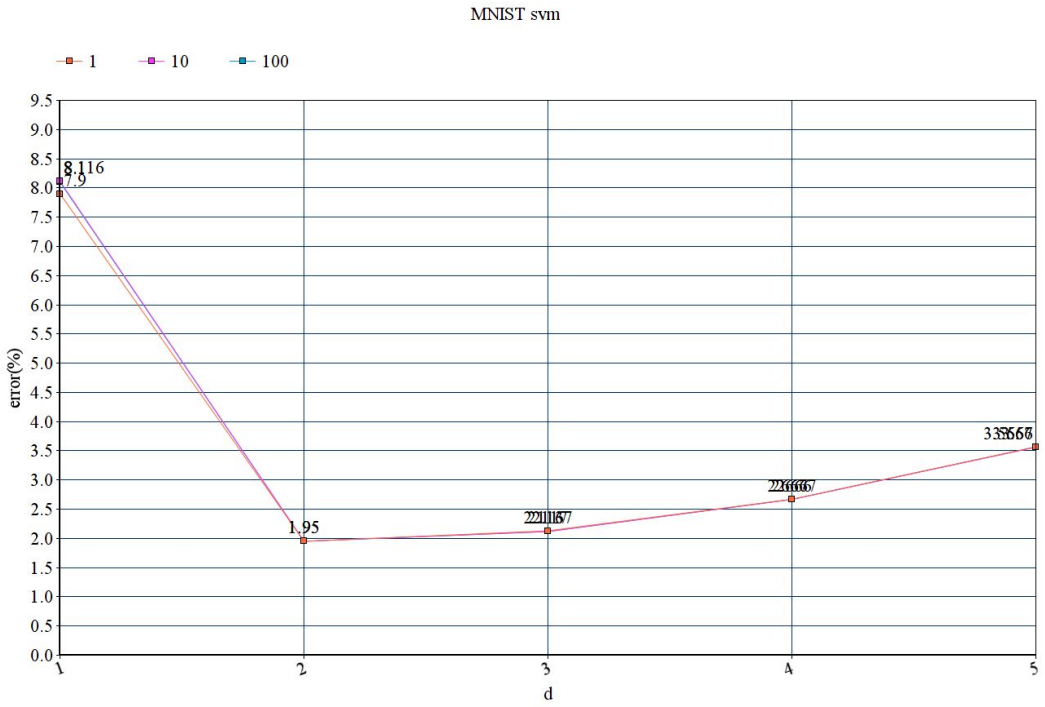
g) theta(t) = (-0.57139, -0.85722) Vector de pesos
theta0(t0) = 5 Humbral

h) Margen(m) = 1.9414

Representación con vectores soporte erróneos marcados



Vamos a resolver el problema MNIST mediante vectores soporte, el código se encuentra en el archivo **svm-exp.m** adjunto con este archivo he probado el error con el polinomial aplicado a distintos valores de C y d.



Que plasmado que el kernel polinomial (t=1) es mucho más efectivo si observamos los valores vemos que el mínimo valor obtenido es el d=2 y c=1 o 10 o 100 dan los 3 valores muy similares **1.95%**

Ejecutamos con estos valores el archivo **svm-eva.m** adjunto a este documento y obtenemos un error del **1.94%** (0.0194 sobre 1)
Con intervalo de confianza del 95% que es 1.96*sqrt((0.0194*(1-0.0194))/N) como N son 10000 da un valor de 0.0027 (2.7%) [1.94-0.27,1.94+0.27]
Probabilidad de error: 1.94%
Con intervalo de confianza: [1.67, 2.21]

Comparándolo con los valores de MNIST vemos que nuestros valores de error son algo más altos en este caso que cualquier ejemplo de MNIST pero el más similar podría ser “Reduced Set SVM deg 4 polynomial” con error 1.1 qué es 0.84 mejor que lo obtenido en esta prueba.

Redes Neuronales

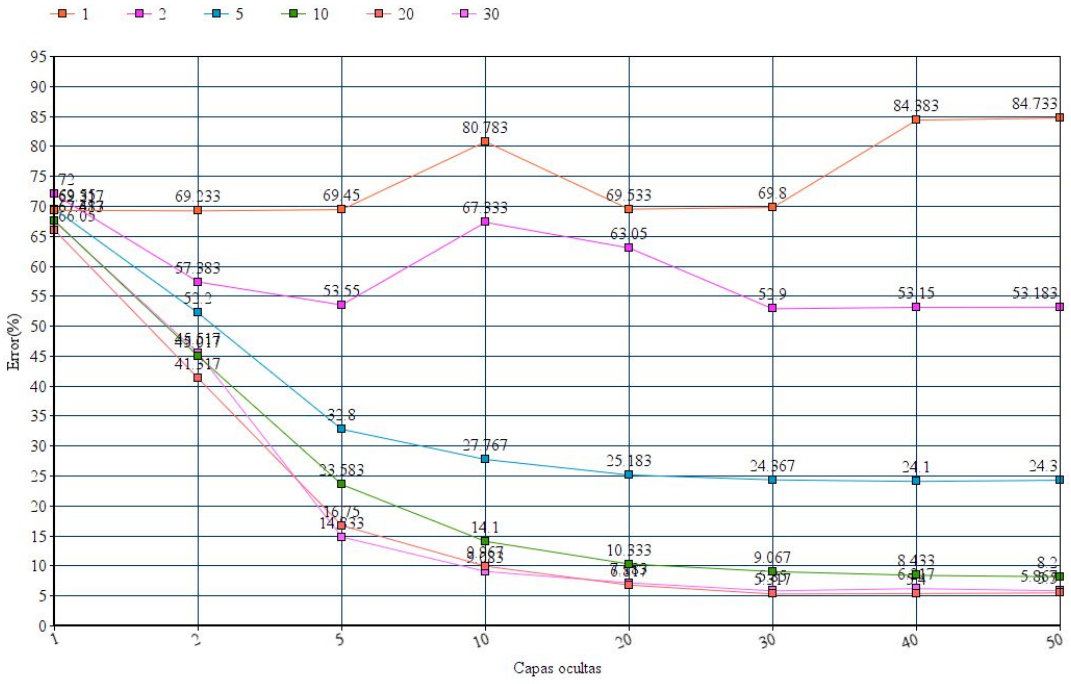
Emplearemos Redes Neuronales Profundas para clasificar cada una de las muestras de MNIST con distintos valores de PCA y distinto número de capas ocultas.

El código se puede encontrar en el archivo **mlp.m** adjunto a este documento.
Aplicaremos primero PCA y después emplearemos mlp.m para estimar el error.

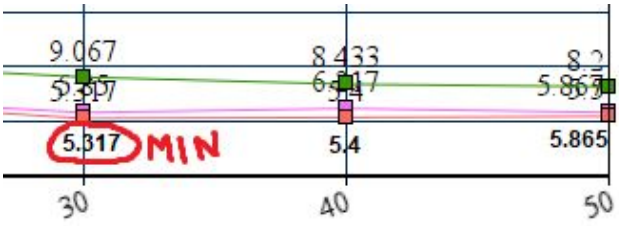
El código de esto reside en el archivo **pca+mlp-exp.m** adjunto a este documento.

A continuación veremos una gráfica con los resultados obtenidos con la ejecución de:

>octave pca+mlp-exp.m train-images-idx3-ubyte.mat.gz train-labels-idx1-ubyte.mat.gz "[1 2 5 10 20 30 40 50]" "[1 2 5 10 20 30]" 40 10



Ampliando la zona inferior derecha para ver mejor el valor mínimo de error:



Nota: Todos se han realizado con un 40% de los datos de entrenamiento y 10% de evaluación excepto:
Capas ocultas: 50 | PCA dim: 20 | Error: 5.500% 30% tr y 10% ev
Capas ocultas: 40 | PCA dim: 30 | Error: 6.217% 30% tr y 10% ev
Capas ocultas: 50 | PCA dim: 30 | Error: 5.867% 25% tr y 10% ev

Como se puede ver en la anterior imagen el mejor número de Capas ocultas es 30 y 20 dimensiones de PCA con un valor de error de **5.317**% porque es el que ofre menor número de error

Vamos a probar estos parámetros para clasificar otro conjunto de muestras para ver cuánto error se obtiene.

El código de esta prueba se puede encontrar en el archivo **pca+mlp-eva.m** adjunto en este documento

Mediante la ejecución de la orden:

>octave pca+mlp-eva.m train-images-idx3-ubyte.mat.gz train-labels-idx1-ubyte.mat.gz "[30]" "[20]" t10k-images-idx3-ubyte.mat.gz t10k-labels-idx1-ubyte.mat.gz

Se obtiene un error de **4.97% (0.0497 sobre 1)**
Con intervalo de confianza del 95% que es 1.96*sqrt((0.0497*(1-0.0497))/N) como N son 10000 da un valor de 0.00426 (0.426%) [4.97-0.426,4.97+0.426]
Probabilidad de error: 4.97%
Con intervalo de confianza: [4.544, 5.396]

Valor obtenido por MNIST para una red neuronal con NN de 2 capas, 300 unidades ocultas es 4.7%. Es un valor muy similar al obtenido experimentalmente dado que la diferencia de error es solo de un 0.27%

Conclusión

Podemos concluir que el mejor método para clasificar MNIST es vectores soporte al menos con los parámetros probados aunque las redes neuronales y en concreto las redes neuronales convolucionales las cuales no se han aplicado son las que teóricamente mejor resuelven el problema dada su capacidad para captar patrones mediante el uso de una malla o matriz que aplica a determinados pixeles. Debido a la simplicidad de las redes aplicadas a MNIST vectores soporte funciona mejor en este caso.

Gracias por su atención