(Multinomial): Se nos pide implementar la función multinomial y luego realizar un experimento con él sobre los datos del dataset MNIST. Así pues el experimento de esta parte consta de 3 programas, el primero es '*multinomialEXP.m*' el cual contendrá el codigo que lanzará el '*multinomial.m*' sobre los datos del training set separándolos en 90% para el Training Set y el 10% restante para el Validation Set. Así pues a continuación de ejecutar el experiment del multinomial para los datos dados, es decir, usando las siguientes epsilons: *ε = [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]*, los resultados de error que nos dió el programa son los que siguen:

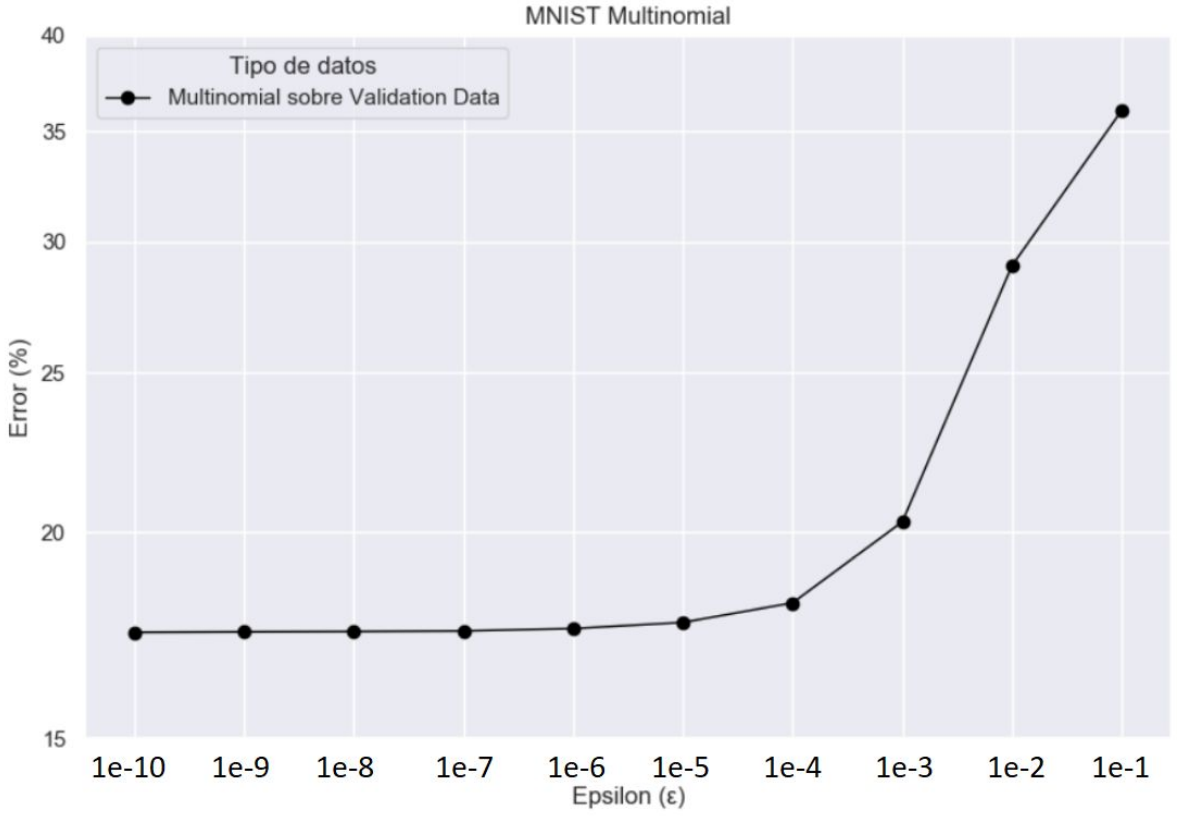


Imagen 1

Si observamos más detenidamente la solución proporcionada por el Octave, (imagen 2), los errores en el rango de 1e-10 a 1e-6 para ver cual es el menor. En la imagen se observa que se calculó el error tanto para ese 90% (el cual no se utilizará en ningún momento) y el VS, que corresponden al validation data, es decir, a ese 10% del MNIST, tenemos que los primeros tres errores son 17.85%, con lo que la mejor epsilon sería la más grande, es decir, la *ε = 1e-8*, así pues, trataríamos los calculos con números más pequeños, ya que recordemos que usamos las epsilons dentro de un logaritmo. Así pues, una vez obtenida nuestro epsilon nueva (*ε = 1e-8*), vamos a proceder a experimentar con dicha epsilon sobre el dataset de testeo, para lo que creamos un nuevo fichero que es el multinomialEVA.m . Así pues, teniendo la epsilon en mente,

```
Error TS para epsilon 1.000000e-10 es: 17.40
Error VS para epsilon 1.000000e-10 es: 17.85
============================================
Error TS para epsilon 1.000000e-09 es: 17.41
Error VS para epsilon 1.000000e-09 es: 17.85
============================================
Error TS para epsilon 1.000000e-08 es: 17.42
Error VS para epsilon 1.000000e-08 es: 17.85
============================================
Error TS para epsilon 1.000000e-07 es: 17.43
Error VS para epsilon 1.000000e-07 es: 17.90
============================================
Error TS para epsilon 1.000000e-06 es: 17.49
Error VS para epsilon 1.000000e-06 es: 17.92
============================================
Error TS para epsilon 1.000000e-05 es: 17.64
Error VS para epsilon 1.000000e-05 es: 18.00
============================================
Error TS para epsilon 1.000000e-04 es: 18.13
Error VS para epsilon 1.000000e-04 es: 18.72
============================================
Error TS para epsilon 1.000000e-03 es: 20.30
Error VS para epsilon 1.000000e-03 es: 20.78
============================================
Error TS para epsilon 1.000000e-02 es: 29.02
Error VS para epsilon 1.000000e-02 es: 29.08
============================================
Error TS para epsilon 1.000000e-01 es: 36.03
Error VS para epsilon 1.000000e-01 es: 35.67
```

Imagen 2

procedemos a la parte de la evaluación sobre el dataset de testing, es decir, el dataset bueno con datos nuevos nunca vistos, para ver como realmente se comporta nuestro modelo. Así pues después de ejecutar el multinomialEVA.m obtenemos el siguiente error:

```
============================================================
Error sobre Test Data es: 16.36
============================================================
```

Lo primero que vemos, es que el error sobre el dataset de testing es menor que sobre el validation set y es normal, suponiendo que el clasificador que usamos es un clasificador estadistico, con lo que al usar todo el dataset de entrenamiento para el entrenamiento en el experiment, pues conseguimos mejorar la predicción y, por lo tanto, mejorar los resultados en la parte del test. Para ver los resultados de una manera más cómoda, los mostraremos en la siguiente gráfica:

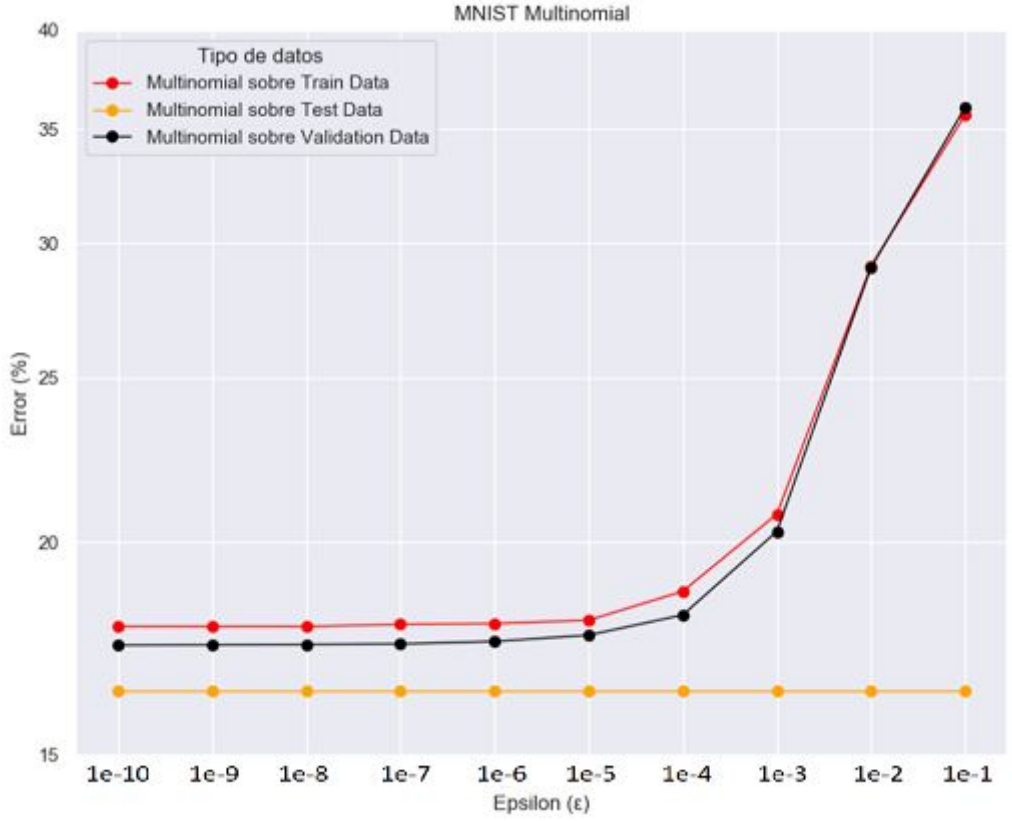
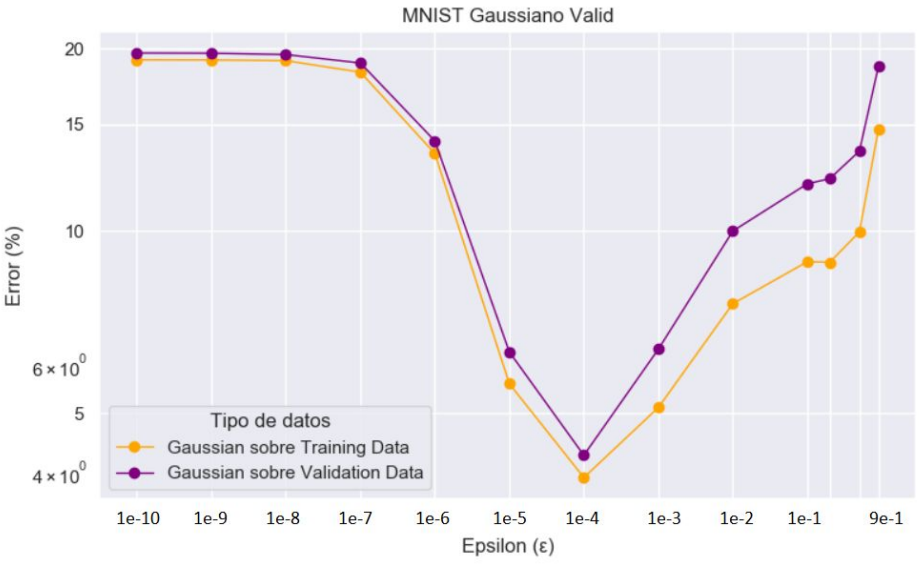


Imagen 3

Si comparamos los resultados obtenidos con los dados en la web de MNIST, nos damos rápidamente cuenta de que el clasificador no es el más bueno, incluso es peor que el clasificador lineal que tiene un 12% de error conseguido con una capa. El resultado es lógico, ya que téngase en cuenta la formulación para conseguir el clasificador multinomial, se basa en la mayor parte de una estadística básica de casos favorables entre los casos totales, así pues es más propenso a errores, ya sea por la introducción de un dataset no balanceado en el entrenamiento que podría ser obtenido por las permutaciones aleatorias que hicimos al comienzo, o directamente con un dataset no balanceado desde el principio, con lo que las probabilidades para cada clase fueron ligeramente modificadas.

***

(Gaussiano): Se nos plantea hacer un clasificador Gaussiano para MNIST al igual que con las otras modificaciones, para realizar dicho experimento utilizaremos el archivo '*gaussian.m*' donde hemos implementado el algoritmo encargado de entrenar el clasificador y calcular dos tipos de errores, uno sobre los datos de testeo y otro sobre los de entrenamiento. A continuación implementamos dos archivos más, '*gaussianEXP.m*' y '*gaussianEVA.m*', el primero se encargará de tomar el dataset de entrenamiento de MNIST y lo partirá en dos conjuntos, uno de 90% de datos, que será nuestro training set y el 10% restante serán los datos del validation set, así pues tras ejecutar el experiment, hemos obtenido los datos mostrados en la gráfica que sigue:




```
========= Alpha= 1e-10 =====
Error TS: 19.163
Error VS: 19.6667
========= Alpha= 1e-09 =====
Error TS: 19.1519
Error VS: 19.65
========= Alpha= 1e-08 =====
Error TS: 19.0667
Error VS: 19.55
========= Alpha= 1e-07 =====
Error TS: 18.287
Error VS: 18.9333
========= Alpha= 1e-06 =====
Error TS: 13.4241
Error VS: 14.0833
========= Alpha= 1e-05 =====
Error TS: 5.59815
Error VS: 6.31667
========= Alpha= 0.0001 ====
Error TS: 3.92037
Error VS: 4.26667
========= Alpha= 0.001 =====
Error TS: 5.11481
Error VS: 6.38333
========= Alpha= 0.01 ======
Error TS: 7.58889
Error VS: 10
========= Alpha= 0.1 =======
Error TS: 8.9
Error VS: 11.9667
========= Alpha= 0.2 =======
Error TS: 8.88148
Error VS: 12.2
========= Alpha= 0.5 =======
Error TS: 9.96481
Error VS: 13.55
========= Alpha= 0.9 =======
Error TS: 14.7074
Error VS: 18.6833
========= ========= ========
```

Imagen 4

Obsérvese que hay dos zonas bien diferenciadas, la que va desde 1e-10 hasta 1e-4, donde el error va bajando, y luego la que va desde 1e-4 hasta 1e-1 donde el error vuelve a crecer, por lo tanto, si queremos mejorar nuestro clasificador, deberíamos estar mirando por la zona mínima, 1e-4. Así pues, una vez vista la alpha mejor que puede minimizar el error, vamos a pasar a evaluar el dataset de Testing con el '*gaussianEVA.m*' con la alpha = 1e-4. Tras correr el algoritmo, el resultado que nos salió es: 4.18%, con lo que podemos confirmar que el clasificador funciona bien si lo comparamos con lo obtenido en el experiment, ya que está un poco mejor que el del experiment, aunque ese porcentaje de diferencia es muy pequeño y no se tiene que tener en cuenta, ya que no llega ni a 1%.

***

Por último, vamos a comparar los resultados con los proporcionados en la web de MNIST, así pues, comparando con un clasificador Gaussiano + 40 dimensiones de PCA, obtenemos que el error es de **3.3%**, es más pequeño del mínimo que nosotros tenemos, y es totalmente lógico, ya que además de aplicar el clasificador Gaussiano, aplica también PCA.

Non-Linear Classifiers			
40 PCA + quadratic classifier	none	3.3	LeCun et al. 1998

Imagen 5

Nuestro clasificador sin embargo, alcanza un error de **4.18%** que, es menor que el error de la web de MNIST, pero aun así sigue siendo muy pequeño, la diferencia entre los dos es tan solo de 0.88%, es decir, una diferencia no muy pequeña, pero aún así es pequeña, ya que dependiendo de los datos sobre los que tendremos que aplicar el resultado, podría darse el caso que nuestro clasificador diese mejores resultados. De todas maneras, podemos afirmar que el clasificador gaussiano implementado es satisfactorio, así que el experimento acaba aqui.