TP1

Tomás Hüttebräucker
Aprendizaje Profundo y Redes Neuronales Artificiales
(7 de septiembre de 2020)

1. MÍNIMOS CUADRADOS Y MALDICIÓN DE LA DIMENSIONALIDAD

Se aplicó un ajuste lineal a un conjunto de M datos n-dimensionales. Los datos se generaron de la forma $y_i = a_1 x_{1i} + \ldots + a_n x_{ni} + b + w_i$ donde y se ve como función de los datos n-dimensionales x, w_i es una componente de ruido y el vector $\mathbf{a} = a_1, \ldots, a_n$ junto con el bias b son los parámetros a ajustar. El ajuste por cuadrados mínimos se realiza de forma cerrada según:

$$\hat{\mathbf{a}} = (X^T X)^{-1} X^T \mathbf{y}$$

donde X es una matriz de $M \times (n+1)$ cuyas filas estas formadas por los vectores n-dimensionales de observación con un 1 incluido al final para el bias e y es la observación de los M datos. El error cuadrático medio del ajuste es de la forma $\|\hat{\mathbf{a}} - \mathbf{a}\| / n$, donde $\hat{\mathbf{a}}$ es el vector estimado y $\mathbf{a}$ es el vector original, usando la norma euclidiana. Con esta definición de error, se procedió a calcular el error del ajuste en función de la dimensión n del problema. Para esto se utilizaron muestras de datos M de diferentes tamaños.

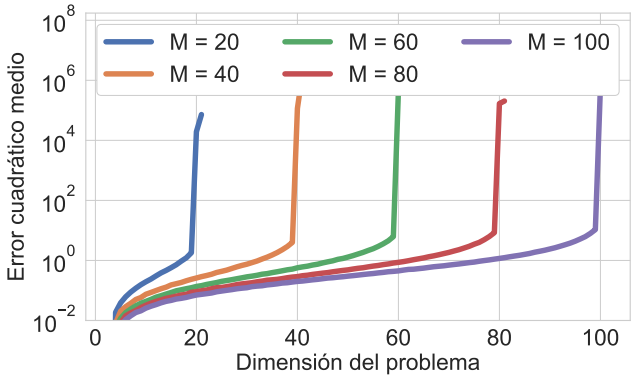
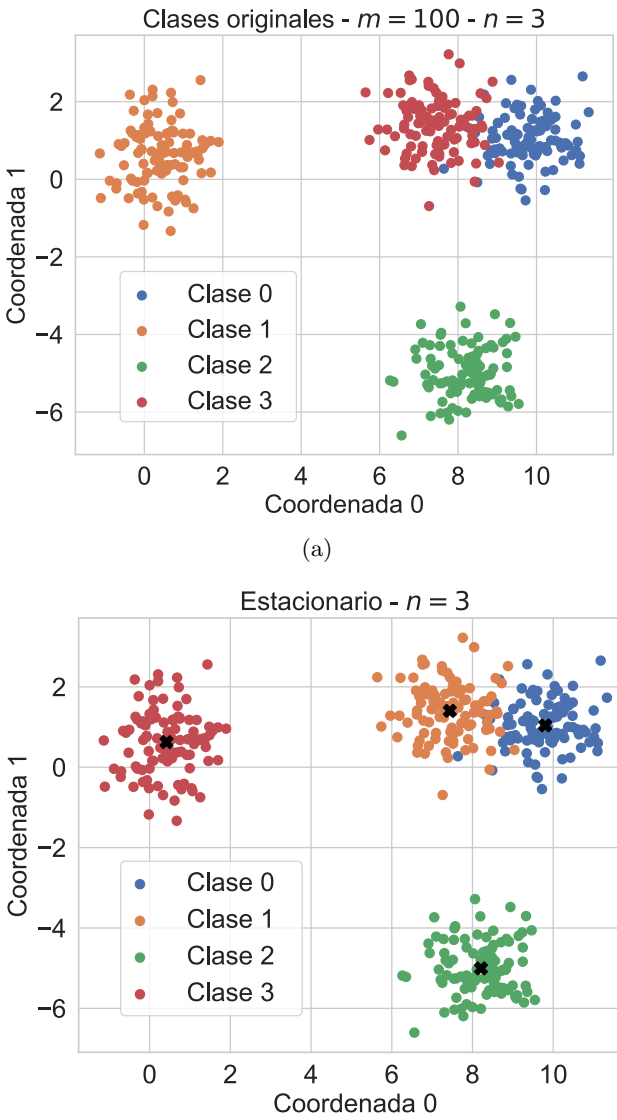


Figura 1: Error del ajuste en función de la dimensión n del problema para diferentes tamaños de datos M usados.

En la Figura 1 se muestra el error cuadrático medio del ajuste en función de la dimensión del problema para diferentes tamaños de datos. Se ve claramente que a medida que la dimensión del problema se acerca a la cantidad de datos, el error diverge. Esto tiene sentido ya que al aumentar la dimensión del problema aumenta la cantidad de incógnitas a ajustar, de hecho si la dimensión del problema es mayor que la cantidad de datos, es imposible realizar un ajuste correcto ya que hay coordenadas de $\mathbf{a}$ sobre las que no hay información.

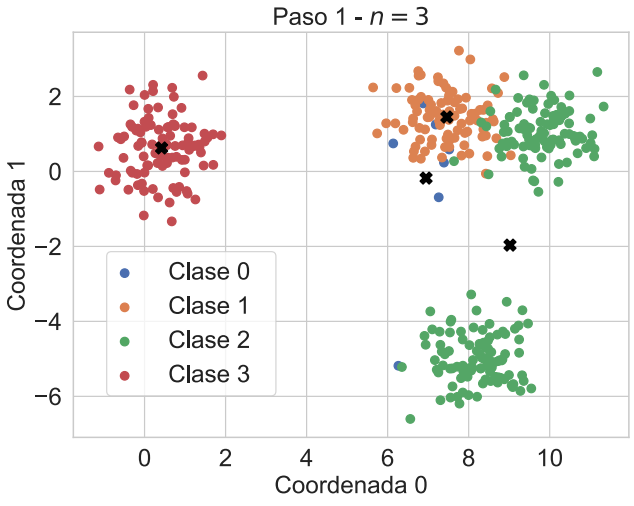
2. K-MEANS



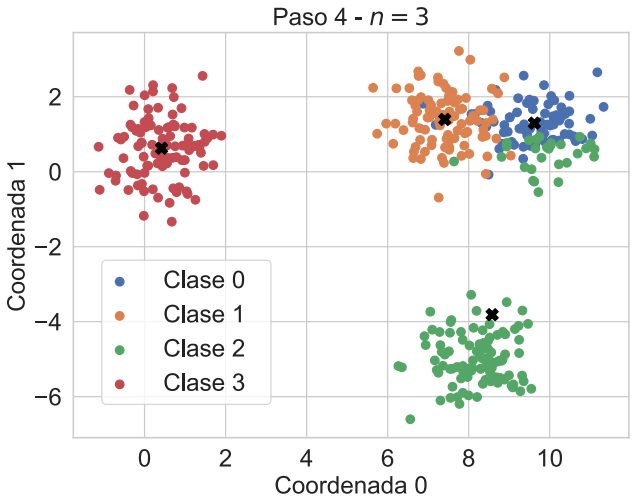
(a)

(b)

Figura 2: Clasificación correcta de cuatro conjuntos compuestos cada uno por 100 puntos de 3 dimensiones mediante el algoritmo K-means. En la imagen (a) se muestran los conjuntos de clasificación originales y en (b) la clasificación realizada con los centros de masa correspondientes en cruz negra. Ambas imágenes se hicieron proyectando sobre las coordenadas 0 y 1.



(a)

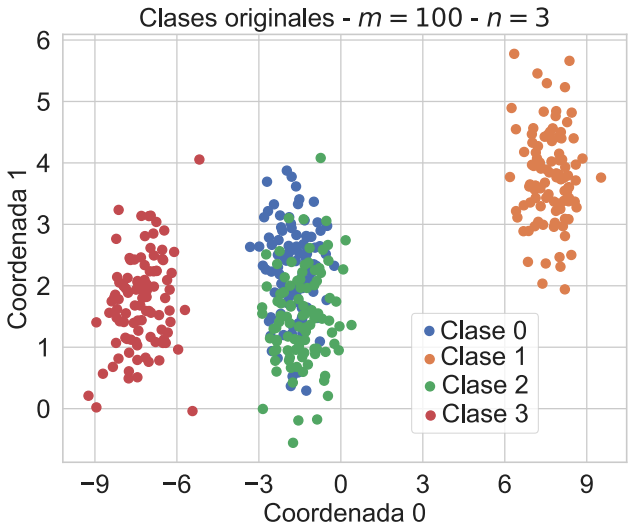


(b)

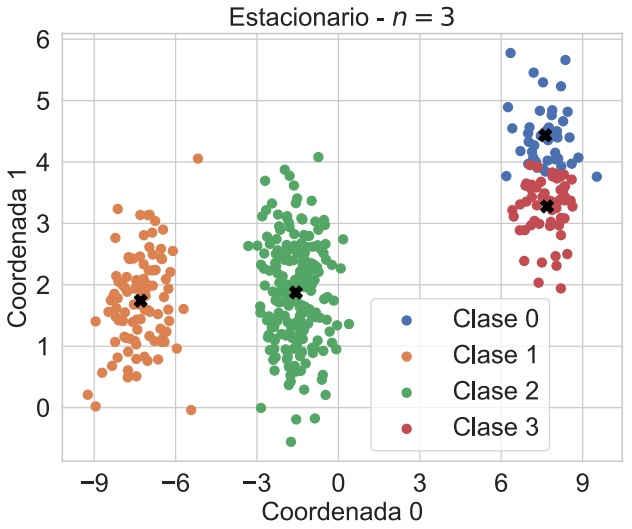
Figura 3: Paso inicial y paso numero 4 de la clasificación K-means sobre cuatro conjuntos compuestos cada uno por 100 puntos de 3 dimensiones. Los puntos de diferentes clases se muestran con diferentes colores y los centros de masa de cada clase se muestran con una cruz negra. Se muestra una proyección sobre las coordenadas 0 y 1.

Se crearon p conjuntos de datos n-dimensionales con distribuciones normales diferentes de forma que 2 clases se superpongan levemente. Una vez hecho esto se realizó una clasificación de los datos usando el método K-means, donde se eligió $k = p$. Para cada conjunto de datos se eligió una distribución normal multivariada con medias aleatorias en cada dirección y matriz de covarianza diagonal y de un valor dado (igual para todas las distribuciones). Para implementar el algoritmo se mezclaron todos los puntos, perdiéndose la información sobre la clase de cada uno, y se eligieron k al azar como centros de masa iniciales. Se utilizo la métrica euclidiana para medir las distancias.

En la figura 2 se muestran los conjuntos originales de puntos tridimensionales y la clasificación realizada, proyectados sobre las coordenadas 0 y 1 del problema. Se ve que el algoritmo clasificó satisfactoriamente los conjuntos, pudiendo diferenciar perfectamente entre puntos de diferentes clases.



(a)



(b)

Figura 4: Clasificación errónea de cuatro conjuntos compuestos cada uno por 100 puntos de 3 dimensiones mediante el algoritmo K-means. En la imagen (a) se muestran los conjuntos de clasificación originales y en (b) la clasificación realizada con los centros de masa correspondientes en cruz negra. Ambas imágenes se hicieron proyectando sobre las coordenadas 0 y 1.

En la figura 3 se muestra la pre clasificación del algoritmo K-Means en el primer y cuarto paso. Se ve como el

algoritmo avanza y corrige los centros de masa a medida que es necesario, haciendo una clasificación mas acertada. El conjunto rojo son puntos que por estar tan alejados nunca cambian su clasificación, básicamente no ven al resto de los puntos.

En la figura 4 se muestra una clasificación de datos errónea. En este caso los datos originales de dos grupos distintos poseían poca distancia entre si. Esto provoco que el algoritmo clasifique como un solo conjunto a datos que provenían originalmente de dos, y ademas, al estar forzado a ser $k = 4$, dividió un conjunto que era único, en dos diferentes.

3. KNN

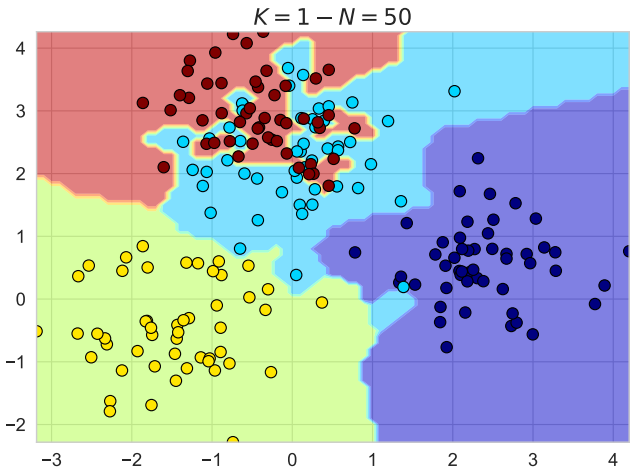
Se implemento el algoritmo de K-nearest neighbours para clasificación de datos en dos problemas. Primero se utilizo para clasificar imágenes de las bases de datos CIFAR10 y MNIST. Luego se utilizo el mismo algoritmo para clasificar distribuciones de puntos bidimensionales creadas aleatoriamente.

3.1. Clasificación de imágenes

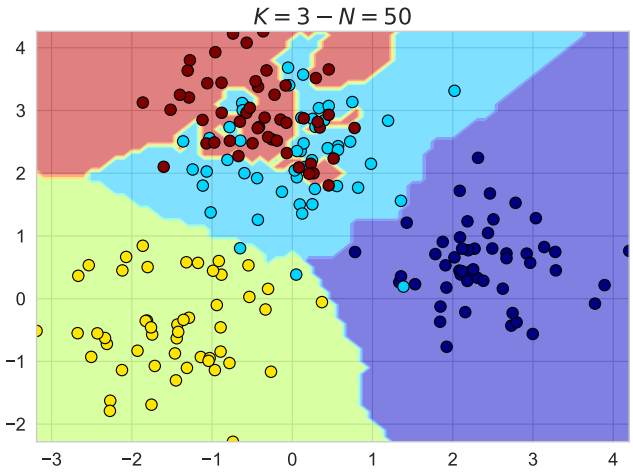
Utilizando la base de datos CIFAR10 y NMIST se entrenó un modelo de K vecinos. Probando el algoritmo sobre los primeros 20 datos de la base MNIST se obtuvo el resultado esperado, 100 % de precisión.

3.2. Clasificación de datos bidimensionales

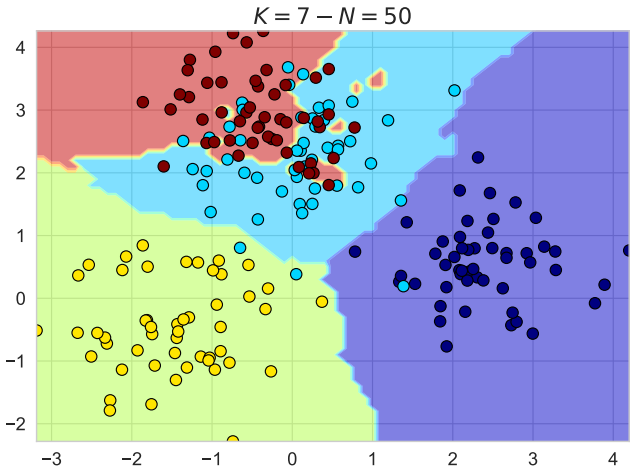
Se crearon cuatro grupos de datos bidimensionales con distribuciones normales de varianza fija y medias aleatorias. Estos conjuntos sirvieron como datos de entrenamiento para el modelo de *K-nearest neighbours*. Luego, generando y clasificando una grilla de puntos, se procedió a graficar las fronteras de decisión para valores K=1, 3 y 7. En la figura 5 se muestran las fronteras de decisión del algoritmo para los diferentes valores de K elegidos. Se observa que los resultados de la clasificación son altamente dependientes del valor del hiperparámetro K elegido.



(a)



(b)

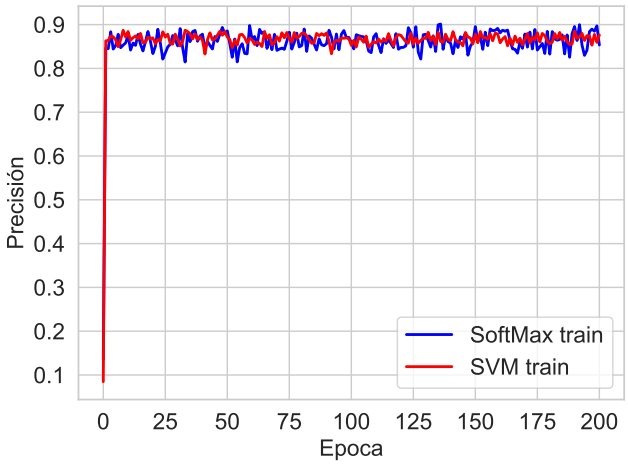


(c)

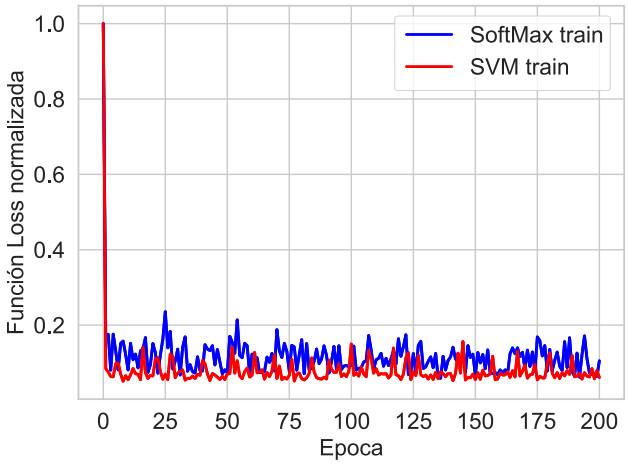
Figura 5: Fronteras de decisión dadas por el algoritmo KNN para 4 conjuntos de datos bidimensionales de N=50 puntos cada uno y diferentes valores de K. Cada color diferente representa una clase diferente

4. CLASIFICADORES LINEALES SVM Y SOFT MAX

Se implementaros dos clasificadores lineales con diferentes funciones de costo, uno de tipo *Support Vector Machine* (SVM) y uno de tipo *Soft Max*. Se entrenaron ambos modelos con las bases de datos CIFAR10 y MNIST mediante el metodo BGD.



(a)



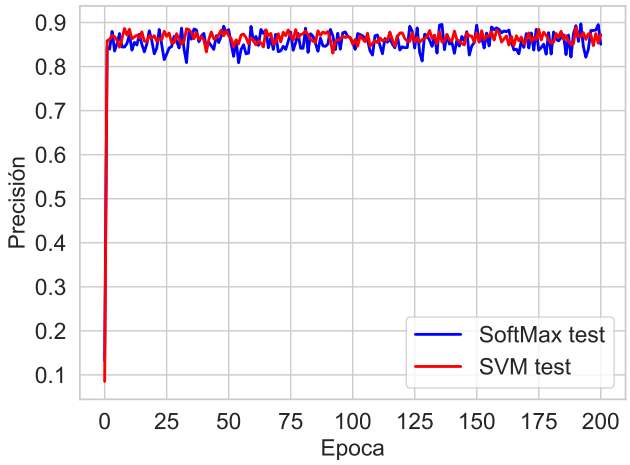
(b)

Figura 6: Precisión en la clasificación (a) y función de costo (b) para los métodos *SoftMax* y *SVM* sobre los datos de entrenamiento de la base MNIST en función de la época de entrenamiento. Para entrenar se usó un *learning rate* de 10^{-3},Δ=1 (SVM), *batch size* de 50 y un parámetro de regularización de 0.1.

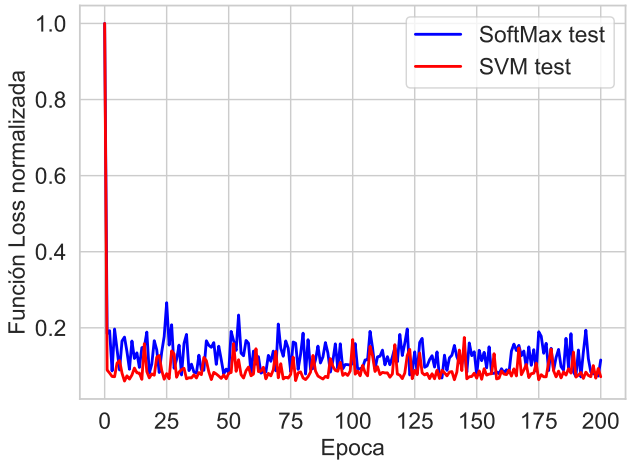
Los parámetros usados para las simulaciones fueron

Batch size:	50
Learning rate:	10^{-3}
λ:	0.1
Δ (SVM):	1

Donde λ es el parámetro de regularización.



(a)

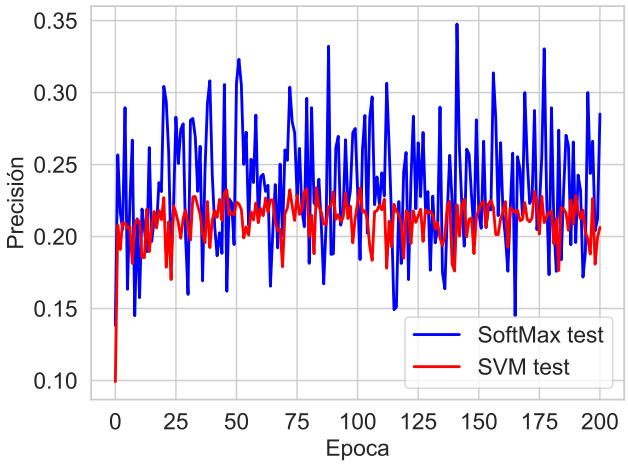


(b)

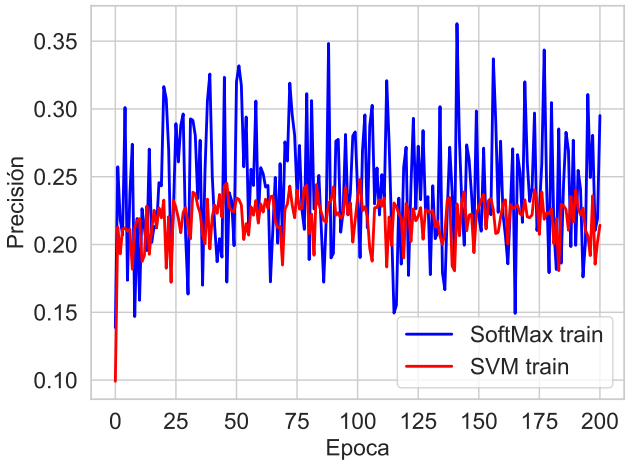
Figura 7: Precisión en la clasificación (a) y función de costo (b) para los métodos *SoftMax* y *SVM* sobre los datos de prueba de la base MNIST en función de la época de entrenamiento. Para entrenar se usó un *learning rate* de 10^{-3},Δ=1 (SVM), *batch size* de 50 y un parámetro de regularización de 0.1.

Se implemento una clase llamada `LinearClassifier` que representa un clasificador de datos lineal. En principio este clasificador cuenta con la matriz de clasificación aleatoria (de la dimensión que se le especifique), también cuenta con las funciones `score`, `predict` y `accuracy` que sirven para obtener los scores, la clasificación realizada de los datos y la precisión en la clasificación respectivamente. Por ultimo incluye una función `fit` que realiza el entrenamiento del clasificador mediante el *gradient descent*, de forma en BGD. Para utilizar este entrenamiento es necesario especificar una función de costo y computar su gradiente. Para esto se implementaron dos clases nuevas, la clase `SVM` y la clase `SoftMax`, cada una heredó

las funciones de `LinearClassifier`, agregando también sus propias implementaciones de función costo y su gradiente, llamadas `loss` y `loss_gradient`. Ambas clases de clasificadores lineales se entrenaron con los datos de las bases CIFAR10 y MNIST.



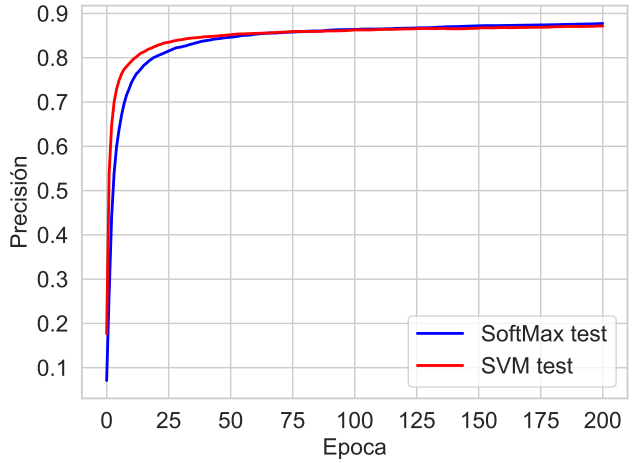
(a)



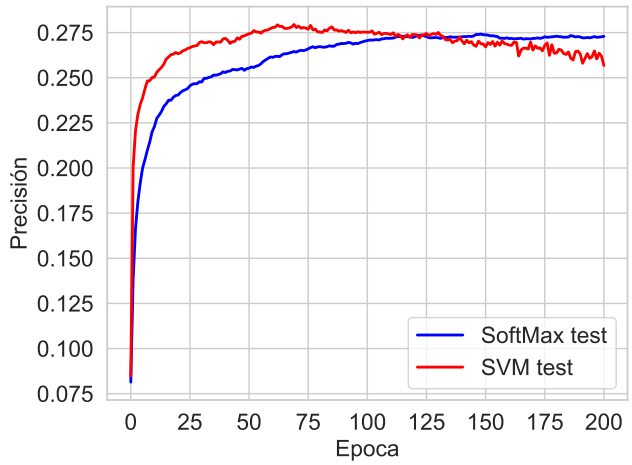
(b)

Figura 8: Precisión en la clasificación (a) y función de costo (b) para los métodos *SoftMax* y *SVM* sobre los datos de prueba de la base CIFAR10 en función de la época de entrenamiento. Para entrenar se usó un *learning rate* de 10^{-3}, Δ=1 (SVM), *batch size* de 50 y un parámetro de regularización de 0.1.

En la figura 6 se muestran la función costo y la precisión de ambos clasificadores sobre los datos de entrenamiento de MNIST. Se ve que, sobre los datos de entrenamiento, ambos clasificadores consiguen una precicion cercana al 90 %, lo que se condice en la reducción de las funciones de costo, normalizada por el valor mas alto. Se observa que tanto la evolución de la función costo como de la precisión del método convergen rápidamente a un valor y a partir de ese momento tienen una evolución ruidosa, esto puede deberse a un *learning rate* muy grande, que evita llegar a un mínimo real y hace que la función quede oscilando entre diferentes valores. En la figura 7 se muestra la evolución de la función de costo y la precisión de los métodos aplicados a los datos de evaluación de la base MNIST a medida que pasan las épocas. El comportamiento es muy similar al obtenido con los datos de entrenamiento, lo que permite suponer que no hay *overfitting* en el entrenamiento. Se ve que en las curvas de accuracy, en principio los modelos aciertan aproximadamente el 10 % de las veces, dado que la cantidad de clases de la base es 10. esta es la probabilidad de acertar aleatoriamente.



(a)



(b)

Figura 9: Precisión de los modelos *SoftMax* y *SVM* aplicados sobre los datos de prueba de MNIST (arriba) y CIFAR10 (abajo) al cambiar el *learning rate* por 10^{-6}

En la figura 8 se muestra la precisión de los modelos al entrenarse con la base de datos de CIFAR10. Se ve que la precisión es mucho peor que al utilizar la base de datos MNIST, lo que indica que estos modelos tienen problemas

para trabajar con colores y sería necesario adaptarlos. También se ve muy ruidosa la evolución, lo que parece indicar que el *learning rate* es muy elevado.

Se cambió el *learning rate* del problema, diminuyendolo hasta 10^{-6}. En la figura 9 se muestran las curvas de precisión para ambos modelos con los datos de prueba de MNIST y CIFAR10. Se ve que al disminuir el *learning rate* el aprendizaje toma mas tiempo (aumenta mas lento la precisión) y es mas estable, no cambiando bruscamete entre epocas. Todo esto tiene sentido y es el resultado que se esperaba obtener al disminuír este parámetro.

5. CONCLUSIONES

Se implementaron diversos métodos de clasificación de datos y se observo la influencia de los hiperparámetros y los datos a clasificar de los mismos. En el algoritmo de los K-Means se mostró que si los datos de clases diferentes estan muy superpuestos es imposible clasificarlos. En el algoritmo KNN se observo la clara dependencia de las fronteras de clasificación con el hiperparámetro K. Al trabajar con clasificadores lineales se observo la relación entre el parámetro *learning rate* y la evolución de la precisión del clasificador.