DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico

Polynomial Regression

Reconocimiento de patrones

Integrante	LU	Correo electrónico
Rodrigo Oscar Kapobel	695/12	rokapobel35@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción **3**
1.1. Clasificador cuadrático . . . 3
1.2. Regresión logística . . . 3
1.3. Fisher (Análisis discriminante lineal) . . . 4

2. Implementaciones **5**
2.1. Clasificador cuadrático . . . 5
2.2. Regresión logística . . . 5
2.2.1. Test A . . . 5
2.2.2. Test B . . . 5
2.3. Análisis discriminante lineal . . . 7
2.3.1. Test A . . . 7
2.3.2. Test B . . . 7

3. Casos de estudio **8**
3.1. Clasificador cuadrático . . . 8
3.2. Regresión logística . . . 8
3.2.1. Test A . . . 8
3.2.2. Test B . . . 9
3.3. Discriminante de Fisher . . . 11
3.3.1. Test A . . . 11
3.3.2. Test B . . . 11

4. Conclusiones **12**

1. Introducción

En este trabajo práctico se implementan diferentes algorítmos de clasificación y de reducción de dimensionalidad para luego realizar estudios sobre sus comportamientos.

1.1. Clasificador cuadrático

A partir de un set de datos $\{(x_n, t_n)\}$ con $n = 1...N$ con t_n en notación 1-de-K se asocia a cada clase C_k con $k = 1...K$ una función $y_k(x) = w_k^t x + w_{k0}$. Notando x_n de manera matricial como $(1, x_n^t)^t$ y w_k como $(w_{k0}, w_k^t)^t$ podemos obtener el clasificador de manera matricial como

$$y(x) = W^t x = T^t((X^t X)^{-1} X^t)^t x$$

Donde W es una matriz que contiene como columnas (w_{k0}, w_k) para todo k, X una matriz que contiene como filas $(1, x_n^t)^t$ para todo n y T una matriz que contiene como filas t_n^t para todo n.
Para clasificar un punto x_n simplemente se elige la clase que maximiza $y_k(x)$.

1.2. Regresión logística

Comencemos por plantear el algorítmo iterativo utilizado para regresión logística:

$$w^{\tau+1} = w^{\tau} - \eta(\sigma(w^t \phi(x_n)) - t_n)\ \phi(x_n)$$

Donde:

- $w \in \mathrm{R}^M$ es el vector de pesos buscado para luego clasificar nuevos datos.
- η es el hiperparámetro que ajusta el tamaño de los pasos que realiza el algorítmo.
- ϕ: $\mathrm{R}^D \rightarrow \mathrm{R}^M$
- $\sigma(a) = \frac{1}{1+e^{-a}}$ es la función logística.

La función logística para dos clases representa la probabilidad a posteriori de la clase C_1. Para clasificar un nuevo dato como perteneciente a la clase C_1, el valor de $\sigma(w^t \phi(x))$ debe ser mayor igual a 0.5. Si no, se considera perteneciente a la clase C_2.

En el caso de $K \geq 3$ clases, habrá K vectores de pesos w_k y por lo tanto cada clase tendrá su probabilidad a posteriori. Para clasificar un nuevo dato, simplemente se elige la clase k para la cual $\sigma(w^t \phi(x))$ es mayor.

La convergencia del algorítmo se puede definir de varias formas en la literatura. Para este trabajo práctico se elige la más naive, pero no por ello menos efectiva, la cual es por cantidad de iteraciones.

Dado que para valores negativos muy grandes en módulo del parámetro a se obtenían errores por overflow en el cálculo de la función logística, se aprovechó el hecho de que la misma es simétrica respecto de cero para calcular 1 - $\sigma(a)$ y así evitar estos errores.

Para realizar el estudio con datos circulares, por alguna razón, no se pudo obtener resultados correctos con la versión iterativa mencionada. A pesar de que la función de costo de regresión logística es convexa, el método no funcionó para este tipo de set de datos. No se pudo encontrar un posible error de aplicación del método, ni una configuración ganadora de los hiperparámetros learning rate (η) y la cantidad de iteraciones.

Por esta razón se implementó Newton-Raphson con el cual se obtuvieron resultados bastante precisos. Dado que Newton-Raphson utiliza el Hessiano en su fórmula, ajusta mucho mejor su learning rate y apunta en mejores direcciones para llegar al mínimo. Pero no quedó claro el motivo por el cual con este método si se obtuvieron los resultados esperados en la clasificación.

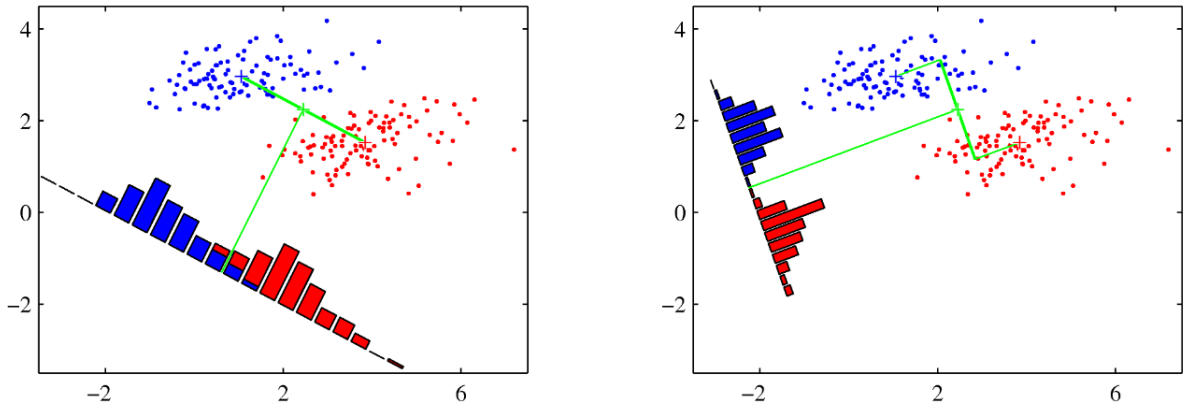
1.3. Fisher (Análisis discriminante lineal)

En general, dadas K clases y datos en dimension D, el efecto de aplicar Fisher a los datos será proyectar los mismos en dimensión $K-1$ aplicando una transformación definida por $y(x) = w^t x$.

Para el caso de dos clases, la fórmula para obtener el vector de pesos se define como:

$$w \propto S_w^{-1}(m_2 - m_1)$$

Donde m_i es la media de los datos en la clase $i \in \{1,2\}$ y S_w es la matriz de dispersión intra clase. El efecto realizado al aplicar la transformación de los datos es una proyección de los mismos en un espacio unidimensional como puede apreciarse en la siguiente imágen:



A partir de esto, los datos proyectados pueden utilizarse para construir un discriminante, eligiendo un threshold y_o y clasificando un nuevo punto como perteneciente a C_1 si $y(x) \geq y_o$ y como C_2 en caso contrario. En este caso el threshold se define como $y_0 = \frac{1}{2} w^t (m_1 + m_2)$.

Para el caso de $K > 2$, se realiza el siguiente procedimiento: Dadas las matrices $\boldsymbol{S}_W$ y $\boldsymbol{S}_B$ definidas como:

$$\mathbf{S}_{\mathrm{W}} = \sum_{k=1}^{K} \mathbf{S}_k$$

$$\mathbf{S}_{\mathrm{B}} = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^{\mathrm{T}}.$$

Donde $\boldsymbol{m}$ y $\boldsymbol{m}_k$ y $\boldsymbol{S}_k$ se definen como:

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k$$

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

$$\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathrm{T}}$$

Y por último N_k es la cantidad de datos en la clase k y N es el total de datos.

Dado que S_b es una matriz de rango a lo sumo $K-1$, tendrá por lo tanto a lo sumo $K-1$ autovalores y podrán obtenerse $K-1$ autovectores. El método define a W como los autovectores de $S_w^{-1}S_b$. La reducción de dimensionalidad se logra aplicando $y(x) = Wx$ para un nuevo dato x.

Para lograr la clasificación de los datos, en este informe se eligió aplicar regresión logística para el caso de $K = 3$.

2. Implementaciones

2.1. Clasificador cuadrático

Los sets de datos se generan a partir de una distribución normal multivariada en R^2 y medias generadas a partir de una distribución uniforme. Para probar el algorítmo de clasificación cuadrática se debe insertar en el terminal desde el directorio TP2 el siguiente comando:

```
python LinearClassificationTest.py
```

Podrá elegirse la cantidad de datos o elegir si testear el clasificador usando los datos de training o un nuevo set de datos, ingresando los comandos indicados por help. Por default, se utilizan dos clases y se testea el clasificador usando un nuevo set de datos.

```
-k NUMBEROFCLASSES   Number of classses.
-e TESTUSINGTRAININGDATA
1: Test de classifier using a different data set.
0: Test using the training data set.
```

2.2. Regresión logística

2.2.1. Test A

El test A realiza separación lineal de k clases. El valor por defecto para k es 2.

Los sets de datos se generan a partir de una distribución normal multivariada en R^2 y medias generadas a partir de una distribución uniforme.

Para probar el algorítmo de regresión logística se debe insertar en el terminal desde el directorio TP2 el siguiente comando:

```
python LogisticRegressionTest.py -t a
```

2.2.2. Test B

El test B realiza separación elíptica de datos utilizando Newton-Raphson.
La función $\phi(x)$ elegida es:

$$\phi(x) = (\ 1, x_0, x_1, x_0x_1, x_0^2, x_1^2\ )$$

Para poder probar este algoritmo de regresión debe insertarse en el terminal desde el directorio TP2 el siguiente comando:

```
python LogisticRegressionTest.py -t b
```

Podrá elegir el número de clases para el test A o si utilizar el set de datos de entrenamiento para testear el clasificador. Por default, la cantidad de clases es dos y se utiliza un set de datos de validación nuevo.

```
-t TEST t in [a, b].
Test a: Linear classification.
Test b: Elliptic classification.
-k NUMBEROFCLASSES Number of classses used in test a.
Default = 2.
-e TESTUSINGTRAININGDATA
1: Test de classifier using a different data set.
0: Test using the training data set.
```

2.3. Análisis discriminante lineal

2.3.1. Test A

Para probar Fisher para dos clases debe ingresar en el terminal desde el directorio TP2 el siguiente comando:

```
python LinearClassificationTest.py -t a
```

2.3.2. Test B

Para probar Fisher para tres clases debe ingresar en el terminal desde el directorio TP2 el siguiente comando:

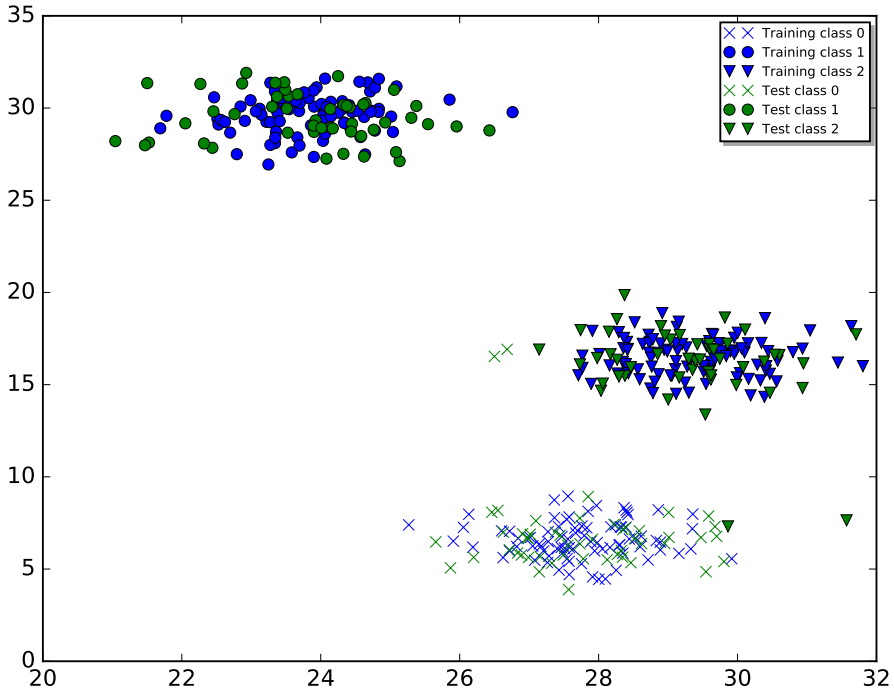
```
python LinearClassificationTest.py -t b
```

Puede elegir si utilizar un set de datos de validación nuevo o si utilizar los datos de entrenamiento para testear los algorítmos. Por defecto se utiliza un set de datos nuevo. Pero puede cambiar esta opción ingresando -e 0. Podrá elegir la dimensión de los datos del test B ingresando -d <valor entero>

3. Casos de estudio

3.1. Clasificador cuadrático

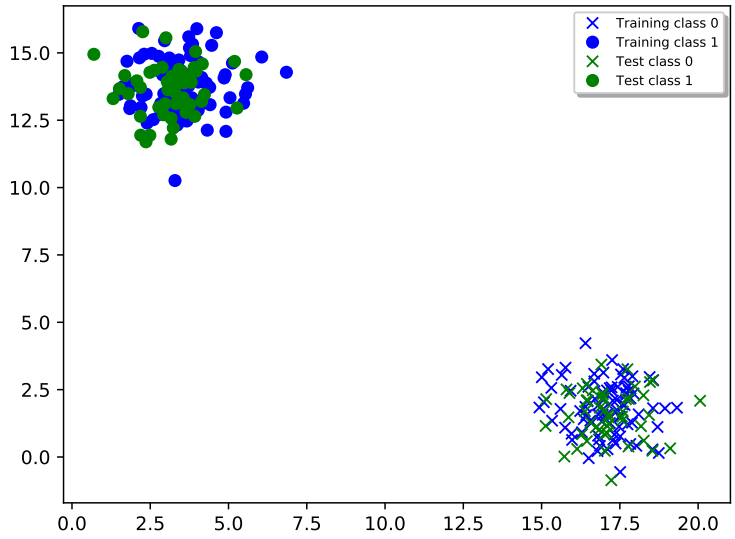
Puede verse que el clasificador cuadrático funciona bastante bien para clases bien definidas. En este caso, se ha realizado un test con 3 clases de training y 3 clases de testeo. Solo ha clasificado mal, visiblemente, 4 datos de test. Por lo que el error para estas clases puede definirse como despreciable aunque no se tomen en este caso medidas rigurosas sobre su funcionamiento.



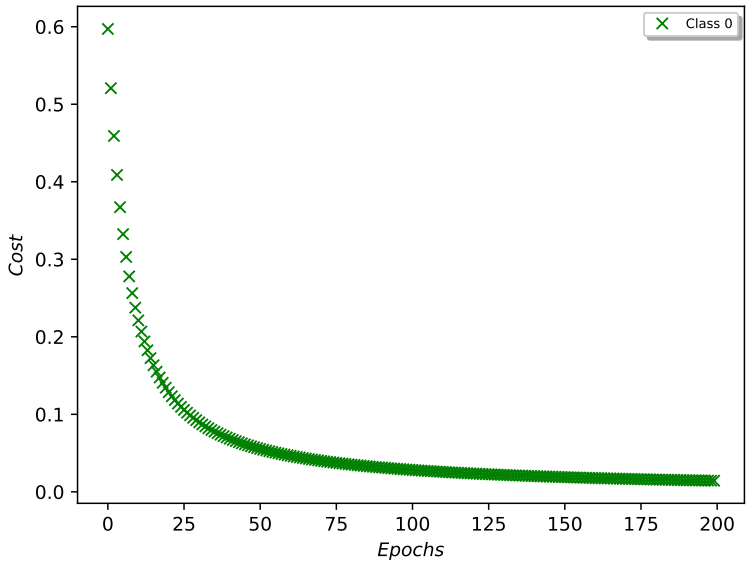
3.2. Regresión logística

3.2.1. Test A

En el siguiente gráfico puede verse el regresor logístico en funcionamiento para separación lineal de datos. Se entrena para dos clases y luego se utiliza un set de datos de validación. Puede verse que, estando las clases bien distanciadas, el error de clasificación es cero.

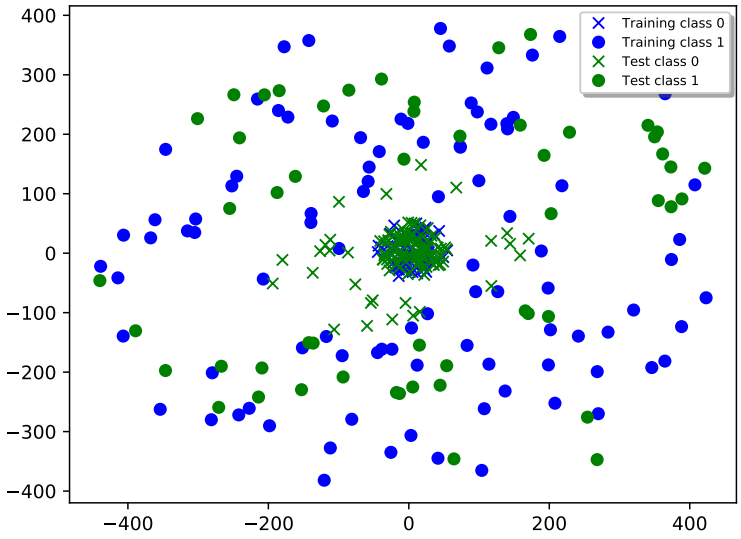


En el siguiente gráfico podemos ver la función de costo calculada para cada epoch del algorítmo. Puede verse que converge rápidamente en 200 iteraciones. Para poder obtener valores graficables, hubo que ajustar el learning rate η a 0.005 dado que para valores mayores, se obtenian overflows en el cálculo de la función de costo. A pesar de este ajuste, se obtuvieron buenos resultados sin cambiar el número de iteraciones.

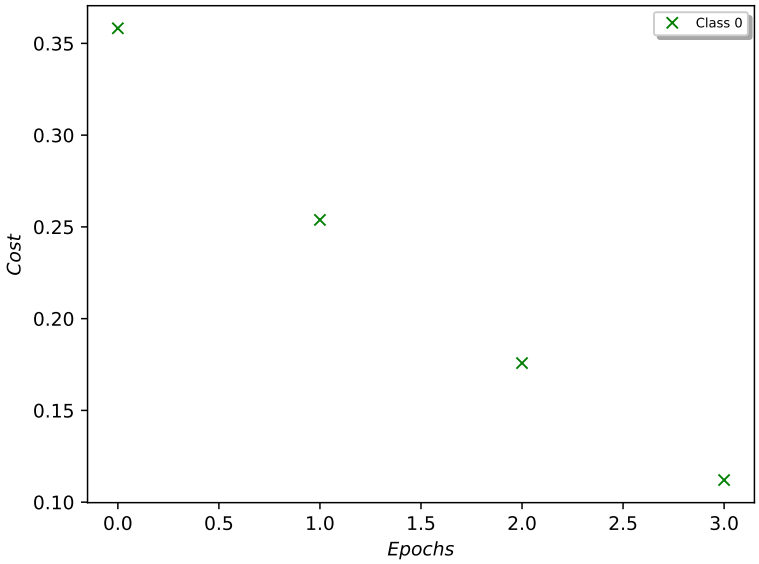


3.2.2. Test B

Para el caso de estudio de regresión logística para separación elíptica de datos utilizando dos clases se ha utilizado Newton-Raphson. Puede verse que en general, los datos son correctamente clasificados para un set de validación nuevo.

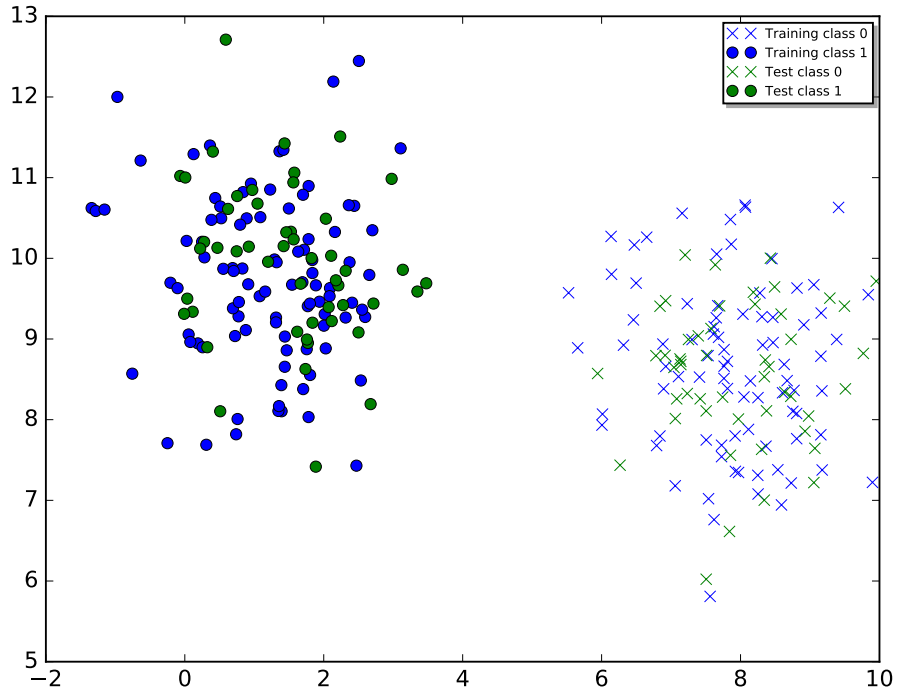


Además, dado que Newton-Raphson utiliza el Hessiano, puede verse en el siguiente gráfico que la función de costo converge rápidamente.

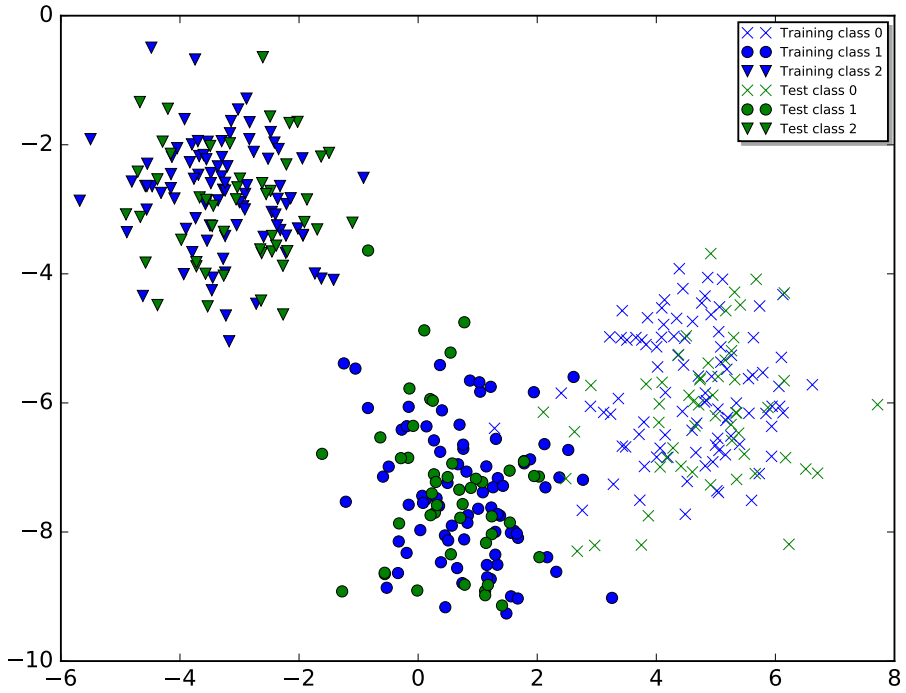


3.3. Discriminante de Fisher

3.3.1. Test A



3.3.2. Test B



4. Conclusiones

1. Los hiperparámetros de regresión logística pueden ser difíciles de configurar dependiendo de las clases que se estén estudiando. El learning rate es importante a la hora de analizar la convergencia de la función de costo. Y la cantidad de iteraciones va de la mano con el learning rate, ya que tal vez, cuanto más grande el learning rate, menor deba ser la cantidad de iteraciones para conseguir llegar al mínimo error.

2. En base a lo mencionado anteriormente, podrían darse casos de overfitting dependiendo de como configuremos los hiperparámetros y como sean las clases estudiadas aunque en este informe no fueron estudiadas. Además, métodos más avanzados para determinar la convergencia pueden ser utilizados y existen en la literatura.

3. Algo que no pudo analizarse, pero hubiese sido interesante para obtener más conclusiones sobre la efectividad de la regresión logística, es comparar contra clasificación cuadrática para clases de datos con outliers y notar que minimización cuadrática, es más sensible a outliers y en cambio, el regresor logístico, al usar como función de activación, la función logística, obtiene más localidad para los datos y es menos sensible a outliers.

4. El análisis discriminante lineal constituye una herramienta importante para la reducción de dimensionalidad y está fuertemente relacionada con el análisis de componentes principales. Reducir la dimensionalidad del problema y luego clasificar permite entre otras ventajas, una mejor visualización de los datos.