Práctica de Laboratorio N◦1

Tópicos de Inteligencia Artificial ...
CComp9-1

Yeroen Félix Medina Vilca
yeroen.medina@ucsp.edu.pe
Yessica Chuctaya Zamata
yessica.chuctaya@ucsp.edu.pe

Universidad Católica San Pablo

1 Introducción

El presente Informe describe la experiencia y resultados de la segunda práctica de Laboratorio del curso de Tópicos ed Inteligencia Artificial referida a Regresión Logística.

La práctica consiste en la implementación de diversas rutinas para un modelo de Regresión Logística de dos conjuntos de datos.

Partimos del mismo modelo de Regersión lineal pero con algunas variaciones. Se conserva la representación de la hipótesis h. Por lo tanto podemos aproximar y como una función lineal de x:

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n \tag{1}$$

donde $\theta_0, \theta_1, \theta_2 ... \theta_n$ son los parámetros, también llamados pesos, los cuales parametrizan el espacio de la función lineal mapeando de X a Y.
Esta función puede ser simplificada así:

$$h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T x \tag{2}$$

Pero tomamos un umbral de clasificación en 0.5:

- Si $h_\theta(x) \geq 0.5$ predecimos $y = 1$
- Si $h_\theta(x) < 0.5$ predecimos $y = 0$

Por lo tanto necesitamos que $0 \leq h_\theta(x) \leq 1$
Para ello hacemos uso de la función sigmoidea:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{3}$$

La función de costo a minimizar ahora está definida por:

$$J(\theta) = \frac{-1}{m} \big[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \big] \tag{4}$$

El algoritmo de Gradiente descendiente se representa como:
Repetir:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{5}$$

donde: α es la tasa de aprendizaje y:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{6}$$

La métrica utilizada será *Accuracy* definido por:

$$Accuracy = \frac{\text{número de predicciones correctas}}{\text{número de predicciones hechas}} \tag{7}$$

Un *accuracy* igual uno (1) indica un modelo perfecto con predicciones sin errores. Un *accuracy* igual a cero (0) indica que no se obtuvo ninguna predicción acertada.

2 Implementación

La implementación[1] fue realizada en Python 3.6, con uso de bibliotecas como numpy y pandas.

La data utilizada corresponde a dos *datasets* "Clima Australia" (weatherAUS) y "Titanic" (titanic_test y gender_submission).

La data fue normalizada por medio de la puntuación estándar definida por:

$$\frac{X - \mu}{\sigma} \tag{8}$$

donde:
μ es la media aritmética
σ es la desviación estándar del conjunto de datos.

3 Experimentos y Resultados

3.1 Experimento 1

Consiste en la búsqueda de los mejores parámetros de entrenamiento para los conjuntos "Clima Australia" y "Titanic" utilizando validación cruzada(k) (*k-fold cross validation*, con k = 3).

En la validación cruzada se entrenó con los parámetros específicos según tabla 1 y se calculó el promedio de los *accuracies* obtenido al ejecutar el algoritmo del gradiente descendiente k veces. En cada vez, uno de los *folds* es usado como conjunto de prueba y el resto, los otros dos, como conjunto de entrenamiento. Los *folds* son conjuntos disjuntos dos a dos del conjunto de datos original.

[1] El código de la implementación se encuentra disponible en https://github.com/giulianodelagala/RegresionLogistica

Table 1: Parámetros para el Experimento 1

Parámetros	Valores
Tasa de Aprendizaje	0.01, 0.05, 0.1, 0.2, 0.3, 0.4
Iteraciones	[500,3500] en incrementos de 500

La tabla 2 resumen los resultados del experimento. Cada fila representa la variación de la tasa de aprendizaje, y las columnas la variación del número de iteraciones.

Table 2: Accuracy de la Regresión Logística para diferentes parámetros

Clima Australia							
	500	1000	1500	2000	2500	3000	3500
0.01	0.589709	0.601245	0.606627	0.609026	0.611032	0.611824	0.612959
0.05	0.611039	0.614729	0.615829	0.615657	0.615650	0.615807	0.615779
0.10	0.614708	0.615657	0.615807	0.615800	0.615772	0.615814	0.615772
0.20	0.615672	0.615800	0.615814	0.615772	0.615772	0.615772	0.615772
0.30	0.615807	0.615814	0.615772	0.615772	0.615772	0.615772	0.615772
0.40	0.615800	0.615772	0.615772	0.615772	0.615772	0.615772	0.615772
Titanic							
	500	1000	1500	2000	2500	3000	3500
0.01	0.873109	0.870106	0.870106	0.870106	0.873137	0.873137	0.876140
0.05	0.873137	0.879170	0.879170	0.873109	0.873109	0.873109	0.873109
0.10	0.879170	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109
0.20	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109
0.30	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109
0.40	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109	0.873109

De la tabla 2 se puede apreciar que la convergencia depende de la tasa de aprendizaje, para valores menores $\alpha < 0.1$, la convergencia requiere mayor número de iteraciones e incluso para valores muy pequeños, esta convergencia no es estable como se aprecia con $\alpha = 0.01$, donde el *accuracy* varía de un conjunto de iteraciones al siguiente.

Luego de analizar la tabla, una tasa de aprendizaje de 0.10 para un número de iteraciones igual a 2500, son parámetros adecuados para el caso del *dataset* "Clima Australia"; y para el caso de "Titanic" se puede reducir a 1000 iteraciones para lograr un costo estabilizado.

También se puede ver que para "Clima Australia" solo se puede predecir con aproximadamente 61% de confiabilidad; y para el caso de "Titanic" se tiene una confiabilidad del 87%.

3.2 Experimento 2

Para los conjuntos "Clima Australia" y "Titanic", se pide mostrar el costo histórico de cada uno de los conjuntos de entrenamiento.

Los resultados para el experimento se muestran en la figura 1 y la figura 2.

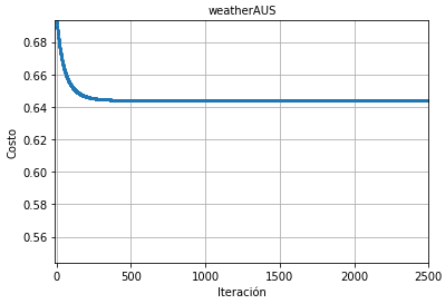
Las gráficas muestran el comportamiento de la función costo para un conjunto de pruebas de un fold, es decir se tiene tres pruebas donde un *fold* es considerado test y dos *folds* son el conjunto de entrenamiento. Por lo tanto se muestran tres gráficos.

Para el caso del conjunto "Clima Australia" se muestran los gráficos de costo correspondiente a una tasa de aprendizaje de 0.20 y 2500 iteraciones.

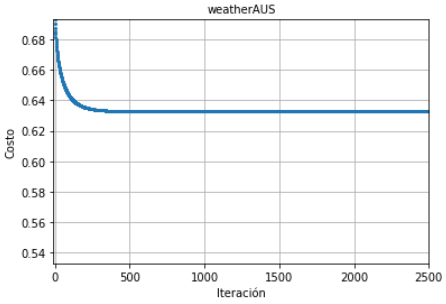
Como se pueden apreciar de la figura 1, la Curva de la Función Costo tienen un comportamiento esperado, por lo que podemos asegurar que la implementación es correcta ya que estamos convergiendo correctamente. Se puede apreciar que el costo se estabiliza se encuentra entre las 500 a 1000 iteraciones, esto debido a que la tasa que elegimos como muestra es alta por lo tanto la convergencia es más rapida. También podemos ver que el costo final obtenido es diferente en cada prueba, por lo que demuestra que el conjunto elegido influye en el modelo de aprendizaje.

Fig. 1: Función de Costo del conjunto "Clima Australia"

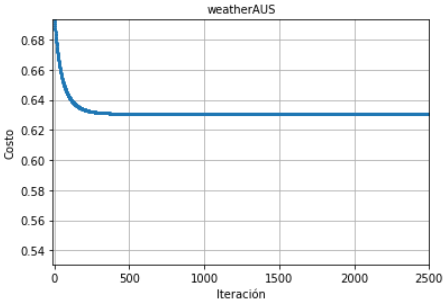
(a)



(b)



(c)

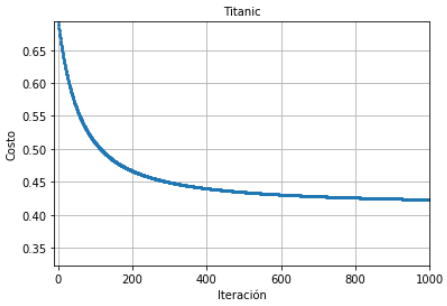


Para el caso del conjunto "Titanic" se muestran los gráficos de costo correspondiente a una tasa de aprendizaje de 0.10 y 1000 iteraciones.

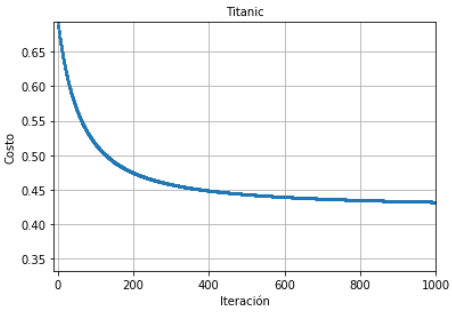
Se puede apreciar que el costo se estabiliza entre las 200 a 400 iteraciones, esto debido a que la tasa que elegimos como muestra no es tan pequeña como en los gráficos del *dataset* anterior por lo tanto la convergencia es más rápida. También podemos ver que el costo final obtenido es diferente en cada prueba, por lo que demuestra nuevamente que el conjunto elegido influye en el modelo de aprendizaje.

Fig. 2: Función de Costo del conjunto "Titanic"

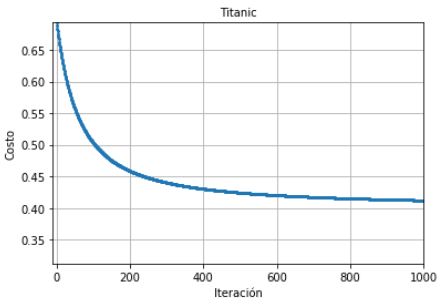
(a)



(b)



(c)



4 Conclusiones

La presente ha descrito los experimentos de la Práctica de Laboratorio 2 del Curso de Tópicos de Inteligencia Artificial. Los diversos experimentos han demostrado que se consigue un comportamiento esperado para las implementaciones de las funciones de Regresión Logística.

Luego de los experimentos se ha conseguido dos modelos de clasificación para los conjuntos de datos "Clima Australia" y "Titanic" con un *accuracy* de 61% y 87% respectivamente.

Las gráficas de la Función de Costo para la Gradiente Descendiente, señalan que la convergencia está relacionada con la tasa de aprendizaje, valores muy pequeños pueden demandar mayor número de iteraciones e incluso crear inestabilidad.

References

1. Andrew Ng, CS229 Lecture Notes