Universidad de Ingeniería y Tecnología

Inteligencia Artificial
CS2601

Regresión Logística y Clasificación

Mauricio Pinto
Francesco Ucelli
Juan Manuel Navarro
Profesor: Cristian López Del Álamo

May 7, 2021

1. Introducción

2. Modelo y Técnicas utilizados

2.1. Regresión No Lineal

Para el trabajo realizado, se utilizó un modelo de **regresión no lineal** para predecir los valores de salida de una función no lineal en relación a una variable x. De esta manera, el valor de salida estaría dado por:

$$h(x_i) = w_0 + w_1 x_i^1 + w_2 x_i^2 + \ldots + w_p x_i^p$$

donde p representa la cantidad total de parámetros en la función.

La regresión busca determinar los valores apropiados de w de tal manera que, para cada valor x_i la función $h(x_i)$ retorne un valor que se aproxime lo más que se pueda a los valores reales de la salida y. De esta forma, se obtiene una función que pueda predecir el comportamiento de y para cualquier valor de x.

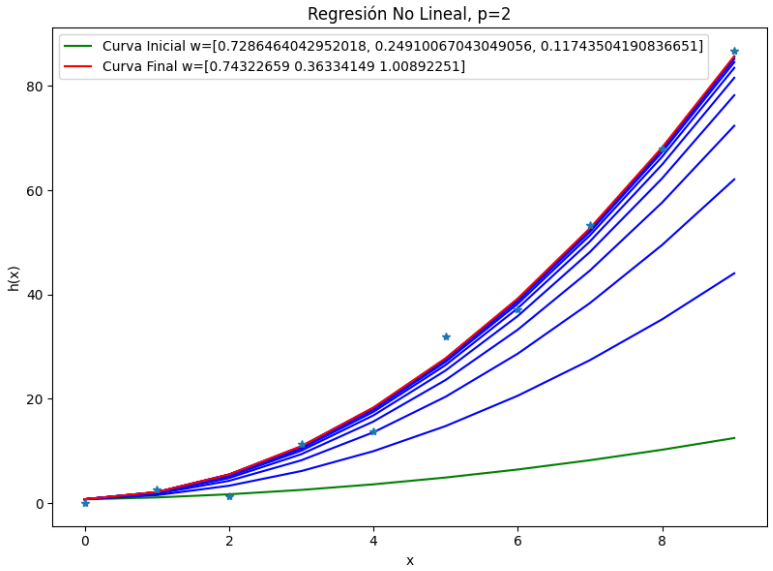

Figura 1: Regresión no lineal donde $p = 2$. Como se observa, al aproximar los valores de w logramos obtener una curva que representa mejor el comportamiento de $h(x)$.

2.2. Funciones de Pérdida

Para realizar modificaciones apropiadas a los valores de w, es necesario contar con una noción de qué tan precisa es la curva generada en relación al comportamiento de $h(x)$. Esta precisión se calcula a través de **funciones de pérdida**. Estas representan el grado de error de una decisión — que en este caso es representada por el valor de w en cierta iteración — con respecto a valores reales obtenidos.

Para la experimentación realizada en este trabajo, se utilizaron específicamente dos funciones de pérdida:

- Mean Square Error (MSE)

$$\frac{1}{2n}\sum_{i=0}^{n}(y_i - h(x_i))^2$$

- Mean Absolute Error (MAE)

$$\frac{1}{n}\sum_{i=0}^{n}|y_i - h(x_i)|$$

Ambas fueron utilizadas y los resultados obtenidos con ambas fueron comparados y analizados.

2.3. Técnicas de Regularización

Al obtener valores de w que minimicen el grado de error, generamos curvas que se ajustan muy deseablemente al comportamiento de los puntos observables. Sin embargo, este tipo de curvas suele no representar apropiadamente el comportamiento de puntos que no han sido observados debido a un sobre-ajuste.

Para contrarrestar este efecto, se utilizan **técnicas de regularización**. Estas alteran la función de error de modo que la curva resultante, a pesar de no aproximarse tanto al comportamiento de los puntos observables, presenta una mejor aproximación cuando se trata de valores no vistos.

Para la realización de nuestros experimentos, se utilizaron dos técnicas de regularización, conocidas como:

- Regularización Ridge

$$\lambda||\hat{w}||^2$$

- Regularización Lasso

$$\lambda||\hat{w}||$$

2.4. Optimización de Gradiente

El descenso de gradiente es uno es los métodos de optimización más utilizados y definitivamente el más popular cuando se trata de redes neuronales. A continuación presentaremos una serie de optimizaciones de descenso de gradiente que buscan mejorar el desempeño de este método. Entonces, si definimos el descenso de gradiente como una manera de minimizar la función $J(\theta)$ que está parametrizada por el conjunto de parámetros θ debemos ir actualizando los parámetros en la dirección opuesta de la función objetivo. Dentro de cada método de optimización existen variables únicas, una común es α que corresponde al $learning_rate$ (tamaño de los pasos que tomamos).Finalmente, el método sobre el cual se trabajan las optimización es el descenso de gradiente estocástico, que se diferencia en que actualiza los parámetros uno a la vez y permite más flexibilidad sobre la función objetivo. A continuación presentamos los algoritmos de optimización con los que trabajamos:

2.4.1. Momentum

El método momentum trae su nombre de un concepto físico, este dicta que cuan más empinada una curva, más velocidad gana un objeto que cae por ella. Este mismo concepto se usa para la actualización de parámetros garantizándonos mejor convergencia. Esta definida por:

$$v_t = \gamma v_{t-1} + \alpha \nabla J(\theta) \tag{1}$$

Donde el término de momentum es γ y usualmente se le da un valor como 0.9 . En términos de implementación, simplemente se calcula el valor de cada actualización y se reemplaza en w

2.4.2. Adagrad

El algoritmo Adagrad tiene un objetivo claro y simple: Adapta la tasa de aprendizaje a los parámetros.Esta adaptación se hace definiendo el cambio según la frecuencia de los parámetros. Esta dado por la ecuación :

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} * g_t \tag{2}$$

Donde G_t equivale a la suma de cuadrados de las gradientes, ϵ es un parámetro arbitrario de suavización que evita la división entre 0 (usualmente 1e-8) y g_t es la gradiente en t.

2.4.3. Adadelta

Adadelta es básicamente una extensión de Adagrad que busca modificar su cambio de α tan agresivo.En cambio,se limita el tamaño de las gradientes pasadas que se almacenan, suavizando la taza con la que cambian los parámetros y dándole aun más flexibilidad. Dado que este algoritmo es más extenso, insertaremos el pseudocódigo obtenido en el paper donde se definió [2]

Algorithm 1 Computing ADADELTA update at time t

Require: Decay rate ρ, Constant ϵ
Require: Initial parameter x_1
1: Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta x^2]_0 = 0$
2: **for** $t = 1 : T$ **do** %% Loop over # of updates
3: Compute Gradient: g_t
4: Accumulate Gradient: $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2$
5: Compute Update: $\Delta x_t = -\frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t}\, g_t$
6: Accumulate Updates: $E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)\Delta x_t^2$
7: Apply Update: $x_{t+1} = x_t + \Delta x_t$
8: **end for**

ADADELTA

2.4.4. Adam

ADAM, por sus siglas en ingles Adaptive Moment Estimation es también un método que calcula la taza de aprendizaje según cada parámetro. Este algoritmo se podría ver como una combinación de adadelta con momentum.Ya que ,además de calcular todo lo adadelta, también almacena el promedio de gradientes pasadas(que decrece exponencialmente).

$$m_t = \beta_1^t m_{t-1} + (1 - \beta_1^t) g_t \tag{3}$$

$$v_t = \beta_2^t m_{t-1} + (1 - \beta_2^t) g_t^2 \tag{4}$$

El único problema sería que dado que todos los vectores se inicializan en 0, este algoritmo estaría sesgado hacia él. Por supuesto, los autores lo tenían en cuenta y definieron el siguiente conjunto de ecuaciones.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{5}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{6}$$

Por lo tanto se define la regla de actualización de Adam:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m} \tag{7}$$

3. Experimentación

A continuación, se explicará la metodología utilizada para los experimentos realizados.

3.1. Experimento 1: Variación de funciones de pérdida

Para este experimento, desarrollamos un software de regresión no lineal utilizando las funciones de perdida MSE(Mean Square Error) y MAE(Mean Absolute error). En los siguientes gráficos, podemos observar la curva sinusoidal que generan ambas funciones:

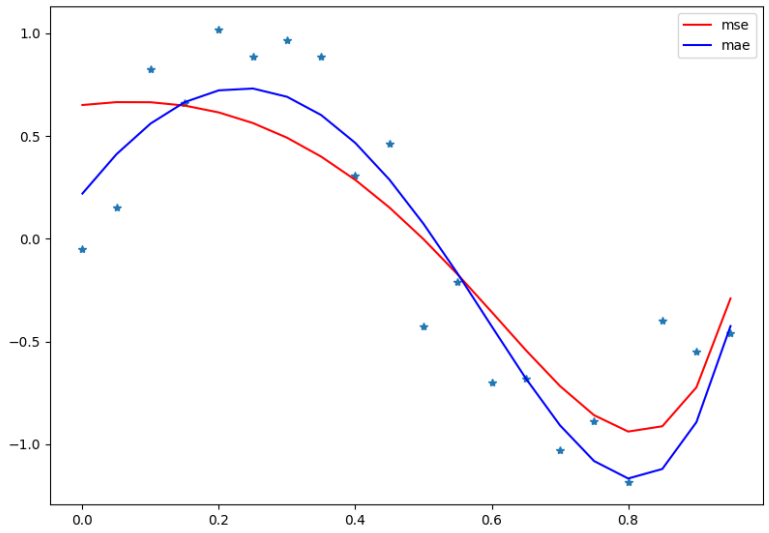

Figura 2: Regresión no lineal utilizando MSE y MAE donde $p = 10$ y $\alpha = 0{,}07$.

Se ejecutó varias veces con diferentes valores de α. Los resultados se muestran en la siguiente tabla.

	Loss ($\alpha = 0{,}07$)	Loss (α=0.1)	Loss ($\alpha = 0{,}4$)	Loss (α=0.7)
Búsqueda RT	0.060286	0.042823	0.042220	0.030743
Búsqueda BF	0.038892	0.02157	0.286250	0.472309

Como se observa en la tabla, la precisión de la curva generada con la función de error MAE es ligeramente más precisa para valores bajos de alpha.

3.2. Experimento 2: Variación de normas de regularización

Para este experimento, se ejecutó la regresión no lineal en tres escenarios: sin regularización, con regularización lasso y con regularización ridge. A continuación se observan las curvas generadas.

A continuación mostramos el error para valores fuera del dataset, para

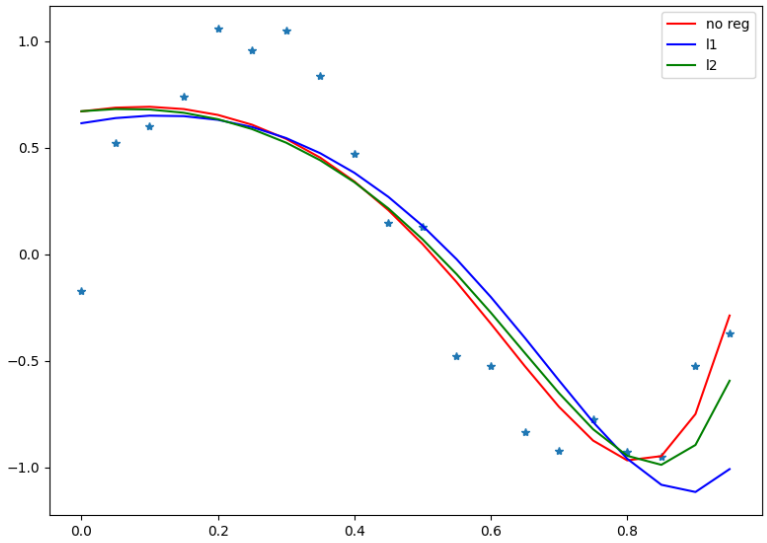

Figura 3: Regresión no lineal utilizando MSE, para regularizaciones lasso y ridge comparadas con una regresión sin regularización donde $p = 10$, $\alpha = 0{,}07$ y $\lambda = 5$.

Se midió el error en puntos generados fuera del dataset (desde $x = 1$ hasta $x = 1{,}4$ en 5 puntos, $y = sin(2\pi x) + \beta$, donde β representa ruido) para medir la precisión real de la curva. A continuación se presentan los resultados:

	Error $\lambda = 1$	Error $\lambda = 2$	Error $\lambda = 3$	Error $\lambda = 4$	Error $\lambda = 5$
Sin Regularización	223.390864	188.059109	257.460442	129.060439	234.262691
Regularización Lasso	5.135003	2.934790	1.294129	12.759626	1.998188
Regularización Ridge	67.214541	99.939136	146.150735	67.585041	81.350766

Como se observa, las curvas obtenidas con regularización tienen menos pérdidas para valores reales

3.3. Experimento 3: Variación de métodos de optimización

Para estos experimentos, se probaron los diferentes métodos de optimización sobre la regresión no lineal sin regularización. En este la tabla se refiere al error en la última iteración (MSE) y Eps al número de épocas.

	$Eps = 100$	$Eps = 1000$	$Eps = 10000$	$Eps = 50000$
Momentum	0.0720	0.0440	0.0121	0.0120
Adagrad	0.0713	0.0415	0.0121	0.0120
Adadelta	0.0710	0.0388	0.0120	0.0120
Adam	0.0587	0.0120	0.0120	0.0120

Como podemos ver en la tabla, para este caso la optimización Adam funciona mejor. Esto se debe a que llega a minimizar el error más rápido por sus características. A continuación mostraremos un gráfico donde se observa el comportamiento muy similar de los primeros tres métodos de optimización y el superior del método Adam. Además, al pasar de 10000 a 50000 iteraciones todos los métodos de optimización convergen en el mismo error y ya no mejoran. Se debe notar que el optimizador Adam llega al punto de convergencia 10 veces más rápido que los demás y probablemente a esto se debe su popularidad.

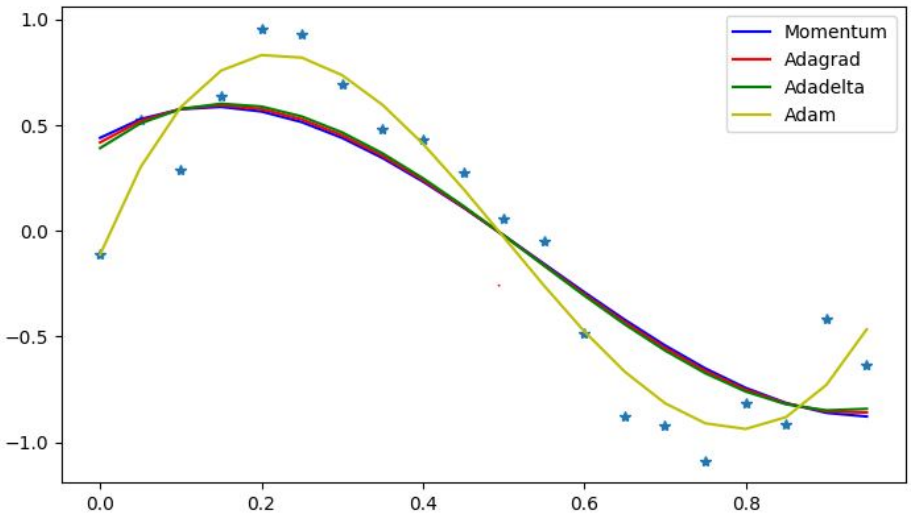

Figura 4: Regresión no lineal utilizando MSE, para optimizaciones con parámetros estandar y 1000 épocas

4. Conclusión

Primeramente, podemos asegurar que el uso del MSE es mucho mas sencillo, sin embargo, utilizar el MAE hace que nuestra regresión sea mas robusto a valores atípicos. Siempre que entrenamos un modelo, nuestro objetivo es encontrar el punto que minimice la función de pérdida. Por supuesto, ambas funciones alcanzan el mínimo cuando la predicción es exactamente igual al valor real. En conclusión, la pérdida de MAE es más robusta a los valores atípicos, pero sus derivados no son continuos, por lo que es ineficiente encontrar la solución. La

pérdida de MSE es sensible a valores atípicos, pero proporciona una solución de forma más estable y cerrada.

La regularización mostró una mejora para valores predecidos fuera del dataset, debido a que las curvas generadas sin regularización presentaban un ligero overfitting. En nuestro caso observamos que para estas curvas y para los valores predecidos, la regularización lasso mostró valores de predicción más aproximados a los reales.

Por otro lado, respecto a los métodos de optimización, si se observa una mejora significativa en comparación con el modelo de descenso de gradiente para regresión no lineal simple. Dentro de estos algoritmos optimizadores, se ve claramente que el Adam tiene un desempeño substancialmente mejor a los demás.

5. Referencias

1 arXiv:1609.04747 [cs.LG

2 Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701, 2012.

3 **Repositorio de Git:** https://github.com/mauriciopinto/IA$_Proyecto$1