Tarea 5: Descenso Gradiente Parte 2

Optimización I

Maestría en Computación

Centro de Investigación en Matemáticas

Esteban Reyes Saldaña
esteban.reyes@citmat.mx

9 de marzo de 2021

Resumen

En esta tarea se utilizó el método de descneso gradiente para maximixar la función log-likelihood del dataset *mnist.pkl.gz*. La función log-likelihood es conocida en la literatura por su capacidad de clasificación binaria. En este caso el conjunto de datos corresponde a imágenes de números dígitos escritos a mano. Para maximizar esta función se minimizó su negativo. Se presenta a continuación una descripción general la función log-likelihood, así como el pseudocódigo de los métodos implementados. En los resultados se incluyen pruebas de descenso gradiente tanto como para búsqueda por backtracking y bisección.Finalmente se incluyen conclusiones observadas a partir de la experimentación.

1 Introducción

1.1 Función Log-Likelihood

Dado un conjunto de datos de entrenamiento,

$$\{x_i, y_i\}_{i=1}^{n}$$

donde $y_i \in \{0, 1\}$, la función **log-likelihood** está dada por

$$\begin{aligned} H(\beta, \beta_0) &= \sum_{i=1}^{n} y_i \log(\pi_i) + (1 - y_i)\log(1 - \pi_i) \\ \pi_i(\beta, \beta_0) &= \frac{1}{1 + \exp(-x_i^T \beta - \beta_0)} \end{aligned}$$

Sea $m = dim(x_0) = dim(x_1) = \ldots, dim(x_n)$. Notemos que $\beta \in \mathbb{R}^m$ y $\beta_0 \in \mathbb{R}$. Entonces, por motivos prácticos y como más adelante estaremos interesados en calcular el gradiente con respecto a los valores beta se introduce el cambio de variable,

$$\phi = [\beta, \beta_0]^T \tag{1}$$

como el vector que concatena a β y a β_0. Además

$$\begin{aligned} -x_i^T \beta - \beta_0 &= -\left[x_i^1, x_i^2, \ldots, x_i^m\right] \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} - \beta_0 \\ &= -\left[x_i^1, x_i^2, \ldots, x_i^m, 1\right] \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \\ \beta_0 \end{bmatrix} \\ &= -\hat{x}_i^T \phi. \end{aligned}$$

Entonces reescribimos $h(\beta, \beta_0)$ como

$$
\begin{aligned}
H(\phi) &= \sum_{i=1}^{n} y_i \log(\pi_i) + (1-y_i)\log(1-\pi_i) \\
\pi_i(\phi) &= \frac{1}{1+\exp(-\hat{x}_i^T \phi)}
\end{aligned}
$$

Algunos resultados relevantes sobre el gradiente, vistos en la tarea pasada son

Teorema 1.1. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. Entonces para toda $x \in dom(f)$, $\nabla f(x)$ es perpendicular al conjunto de nivel

$$S = \{x \in \mathbb{R}^n | f(x) = c, c \text{ constante.}\}$$

Teorema 1.2. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. Entonces la dirección donde $f(x)$ crece más rápido es $\nabla f(x)$.

Corolario 1.1. Bajo las condiciones del Teorema (1.2), $f(x)$ decrece más rápido en la dirección $-\nabla f(x)$.

Definición 1.1. Una **dirección de descenso** $d \in \mathbb{R}^n$ para $f \in \mathcal{C}^1$ es un vector tal que

$$f(x+td) < f(x)$$

para $t \in (0, T)$. Es decir, permite que el punto x más cerca al mínimo local x^* de la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$.

Teorema 1.3. Si $g(x)^T d < 0$ entonces d es una dirección de descenso.

Observación. La dirección

$$d_k = -g(x_k)$$

es la elección más obvia de una dirección de búsqueda.

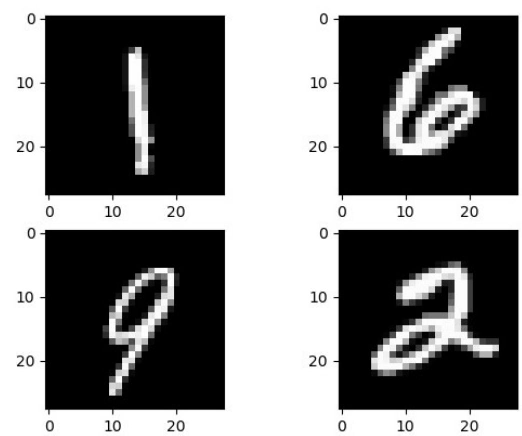
Observación. El objetivo es **maximizar** la función $h(\phi)$ usando descenso gradiente. Por los resultados anteriores es fácil ver que esto es equivalente a **minimizar** $-h(\phi)$.

2 Datos MNIST

Los datos utilizados corresponden a la base de datos MNIST. Que son imágenes de números dígitos escritos a mano. Las imágenes tiene dimensión 28×28 y la distribución de los datos está dada por

Datos	Cantidad
Entrenamiento	50000
Pruebas	10000
Validación	10000

A continuación se muestra un ejemplo del conjunto de entrenamiento



Dado que la función $h(\phi)$ corresponde a clasificación binaria, la experimentaición se realizó con las imágenes de etiqueta cero y uno.

3 Método

3.1 Gradiente

Usando la relación derivada-gradiente y regla de la cadena tenemos que

$$
\begin{aligned}
\nabla \pi_i(\phi) &= -\frac{\exp(-\hat{x}_i^T \phi)}{[1+\exp(-\hat{x}_i^T \phi)]^2}\hat{x}_i \\
&= -\frac{1+\exp(-\hat{x}_i^T \phi) - 1}{[1+\exp(-\hat{x}_i^T \phi)]^2}\hat{x}_i
\end{aligned}
$$

$$
\begin{aligned}
&= \left(-\frac{1}{1+\exp(-\hat{x}_i^T\phi)} + \frac{1}{[1+\exp(-\hat{x}_i^T\phi)]^2}\right)\hat{x}_i \\
&= [-\pi_i(\phi) + \pi_i^2(\phi)]\hat{x}_i \\
&= (1-\pi_i(\phi))\pi_i(\phi)\hat{x}_i.
\end{aligned}
$$

Así que

$$
\nabla\pi_i(\phi) = (1-\pi_i(\phi))\pi_i(\phi)\hat{x}_i. \qquad (2)
$$

Ahora,

$$
\begin{aligned}
\nabla h(\phi) &= \sum_{i=1}^{n}\left[y_i\frac{\nabla\pi_i(\phi)}{\pi_i(\phi)} + (1-y_i)\frac{\nabla(1-\pi_i(\phi))}{1-\pi_i(\phi)}\right] \\
&= \sum_{i=1}^{n}\left[y_i\frac{(1-\pi_i(\phi))\pi_i(\phi)\hat{x}_i}{\pi_i(\phi)}\right. \\
&\qquad \left. -(1-y_i)\frac{(1-\pi_i(\phi))\pi_i(\phi)\hat{x}_i}{1-\pi_i(\phi)}\right] \\
&= \sum_{i=1}^{n}[y_i(1-\pi_i(\phi))\hat{x}_i - (1-y_i)\pi_i(\phi)\hat{x}_i]
\end{aligned}
$$

Entonces

$$
\nabla h(\phi) = \sum_{i=1}^{n}[y_i(1-\pi_i(\phi))\hat{x}_i - (1-y_i)\pi_i(\phi)\hat{x}_i]
$$

3.2 Pseudocódigo

3.2.1 Búsqueda con BackTracking

Input: $\hat{\alpha}$, $\rho \in (0,1)$, $c_1 \in (0,1)$
Output: Tamaño de paso α_k

Haga $alpha = \hat{\alpha}$
$inum = 0$
while $f(x_k + \alpha d_k) > f(x_k) + c_1\alpha\nabla f_k^T d_k$ **do**
  $alpha \to \rho\alpha$
end while
Regresa $\alpha_k = \alpha$

3.2.2 Búsqueda en Línea con Bisección

Input: α_0, $0 < c_1 < c_2 < 1$
Output: Tamaño de paso α_k

Haga $alpha = 0$, $\beta = \infty$, $\alpha^i = \alpha_0$
$inum = 0$
while $f(x_k + \alpha_k d_k) > f(x_k) + c_1\alpha_k\nabla f_k^T d_k$ **o** $\nabla f(x_k + \alpha_k d_k)^T d_k < c_2\nabla f(x_k)^T d_k$ **do**
  if $f(x_k + \alpha_k d_k) > f(x_k) + c_1\alpha_k\nabla f_k^T d_k$ **then**
    $\beta = \alpha_k$
    $\alpha_k = \dfrac{\alpha+\beta}{2}$
  else
    $\alpha = \alpha_k$
    if $\beta = \infty$ **then**
      $\alpha_k = 2\alpha$
    else
      $\alpha_k = \dfrac{\alpha+\beta}{2}$
    end if
  end if
end while
Regresa α_k

4 Resultados

Del conjunto de entrenamiento se usaron un total de $10,610$ ejemplos de dígitos cero y uno. Como vector inicial se usó $\beta = [1,\ldots,1]^T$, $\beta_0 = 1$ y para los algoritmos de búsqueda en línea se usaron los parámetros

Parámetro	Valor
α	0.9
ρ	0.5
c_1	10^{-4}
c_2	0,9

Para un máximo de 500 iteraciones se obtuvo

Algoritmo	$\|\|G_k\|\|$	tiempo
BackTracking	0.0166	228.86 segundos
Bisección	0.0053	494.09 segundos

Para medir el error se usó la función

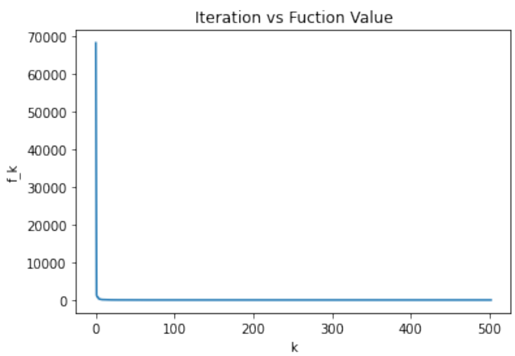
$$error = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{1}_{\pi(\hat{\beta},\hat{\beta}_0)>0,5}(x_i) - y_i|$$

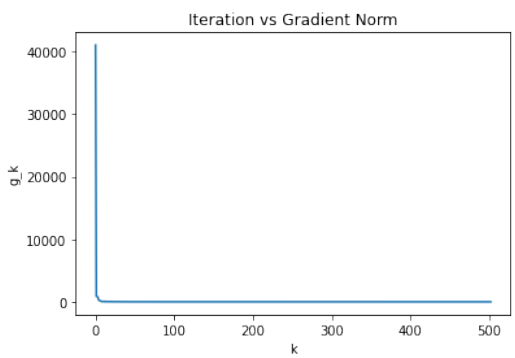
donde $\{(x_i, y_i)\}_{i=1}^{n}$ se obtiene del conjunto train_set y $x_i \in \mathbb{R}^{784}$ y $y_i \in \{0, 1\}$.

Para el punto mínimo encontrado en la experimentación se obtuvo

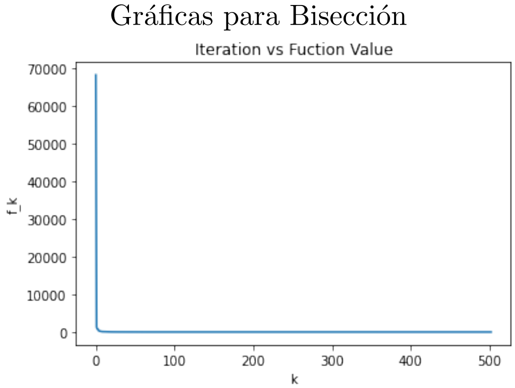
Algoritmo	*error*
BackTracking	0.0047
Bisección	0.0047

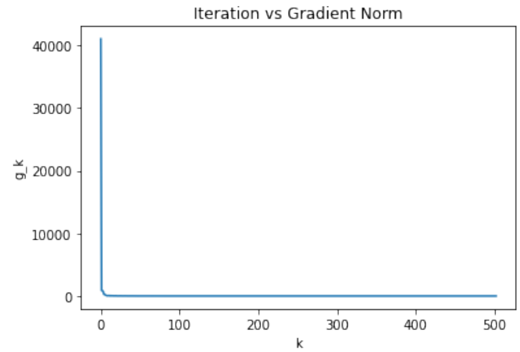
Gráficas para Backtracking





Gráficas para Bisección





5 Conclusiones

- El problema de clasificación binaria de un conjunto de entrenamiento se puede plantear como un problema de optimización. En este caso se usó la función de log-likelihood para clasificar un conjunto de imágenes de ceros y unos del dataset MNIST.

- En la experimentación se observó que el método de backtracking convergió en menor tiempo comparado con el método de bisección. La norma del gradiente quedó más cerca de cero para bisección y el error obtenido para ambos métodos se mantuvo igual.

- Observemos que el error es pequeño, considerando el número total de imágenes del conjunto de validación. Intuitivamente esto puede ocurrir porque la clasificación es clara para ceros y unos (dado que la forma de escribir estos dígitos es muy diferente). Lo que sugiere probar este algoritmo para clasificar unos y sietes o seis y nueves.