Machine Learning

Dia 3 - Regressão Logística e Gradient Descent

ImageU - Grupo de Pesquisa em Machine Learning e Visão Computacional
https://imageu.github.io/

Curso de Verão 2022

Instituto de Matemática e Estatística - IME USP

Programa

1. Regressão Logística
2. Função de Custo
3. Otimização utilizando Gradient Descent

Regressão Logística

Regressão Logística

- Modelo linear para classificação binária
- Similar à Regressão Linear, mas usa a função logística em sua formulação
- Treinado de forma iterativa com Gradient Descent

Regressão Logística

- Suponha classificação binária: $y \in \{-1, +1\}$
- Observações $(\mathbf{x}, y)$ seguem uma distribuição $P(X, Y)$
- O target que queremos aprender não é mais uma função que determina y para cada $\mathbf{x}$
- Nosso target agora é uma distribuição $P(y|\mathbf{x})$

Caracterização de $P(y|\mathbf{x})$

- Se definirmos $f(\mathbf{x}) = P(y = +1|\mathbf{x})$, a distribuição $P(y|\mathbf{x})$ pode ser caracterizada por:

$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{se } y = +1, \\ 1 - f(\mathbf{x}), & \text{se } y = -1 \end{cases}$$

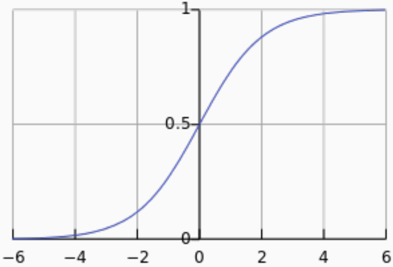
- (note que, alternativamente, poderíamos ter definido $f(\mathbf{x}) = P(y = -1|\mathbf{x})$)
- Note que não temos acesso a $f(\mathbf{x})$; só sabemos que y segue a distribuição $P(y|\mathbf{x})$
- Se conseguirmos aprender $f(\mathbf{x})$, teremos aprendido $P(y|\mathbf{x})$

Função logística ou sigmóide

- Ideia é considerarmos hipóteses do tipo

$$h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$$

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$



$$0 \leq \sigma(z) \leq 1 \quad \Longrightarrow \quad 0 \leq h_{\mathbf{w}}(\mathbf{x}) \leq 1$$

Truque

- Supondo que $h_{\mathbf{w}}(\mathbf{x}) \approx f(\mathbf{x})$, é natural supor que

$$\hat{P}(y|\mathbf{x}) \approx \begin{cases} h_{\mathbf{w}}(\mathbf{x}), & \text{se } y = +1, \\ 1 - h_{\mathbf{w}}(\mathbf{x}), & \text{se } y = -1 \end{cases}$$

- é uma boa aproximação de $P(y|\mathbf{x})$
- (note que $1 - \sigma(z) = \sigma(-z)$)
- Usando isso, podemos escrever

$$\hat{P}(y|\mathbf{x}) = \sigma(y\, \mathbf{w}^T \mathbf{x})$$

Aprendizado do target $f(\mathbf{x}) = P(y = 1|\mathbf{x})$

- Dados disponíveis:

$$\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)}) \in X \times Y, n = 1, \ldots, N\}$$

- Dada uma observação $(\mathbf{x}, y)$, nós estamos supondo que y segue uma distribuição $P(y|\mathbf{x}) = \sigma(y\,\mathbf{w}^T\mathbf{x})$

- Princípio da máxima verossimilhança:
  - Dentre todas as distribuições $\sigma(y\,\mathbf{w}^T\mathbf{x})$, qual é a que gerou D?
  - É aquela que maximiza a probabilidade conjunta

$$\prod_{n=1}^{N} P(y^{(n)}|\mathbf{x}^{(n)}) = \prod_{n=1}^{N} \sigma(y^{(n)}\,\mathbf{w}^T\mathbf{x}^{(n)})$$

Função de Custo

Problema de Otimização

- Encontrar $\mathbf{w}$ que <u>maximiza</u>

$$\prod_{n=1}^{N} \sigma\big(y^{(n)}\,\mathbf{w}^T\mathbf{x}^{(n)}\big)$$

- Ou, equivalentemente, <u>maximiza</u>

$$\frac{1}{N}\ln\Big(\prod_{n=1}^{N} \sigma\big(y^{(n)}\,\mathbf{w}^T\mathbf{x}^{(n)}\big)\Big)$$

Problema de Otimização

- Quero **w** que minimiza

$$-\frac{1}{N}\ln\Big(\prod_{n=1}^{N}\sigma\big(y^{(n)}\,\mathbf{w}^T\mathbf{x}^{(n)}\big)\Big)$$

$$\frac{1}{N}\sum_{n=1}^{N}\ln\Big(\frac{1}{\sigma\big(y^{(n)}\,\mathbf{w}^T\mathbf{x}^{(n)}\big)}\Big) \qquad (\text{ pois } \ln\tfrac{1}{a} = -\ln a\ )$$

$$\frac{1}{N}\sum_{n=1}^{N}\ln\Big(1+e^{-y^{(n)}\,\mathbf{w}^T\mathbf{x}^{(n)}}\Big) \qquad (\text{ pois } \tfrac{1}{\sigma z} = \tfrac{1}{\frac{1}{1+e^{-z}}}\ )$$

Problema de Otimização: Função de Custo

- Função de custo na regressão logística:

$$E_{in} = \frac{1}{N} \sum_{n=1}^{N} \underbrace{\ln\left(1 + e^{-y^{(n)}\,\mathbf{w}^T \mathbf{x}^{(n)}}\right)}_{err(y^{(n)},\hat{y}^{(n)})}$$

- Interpretação:
  - Se $y^{(n)}$ e $\mathbf{w}^T \mathbf{x}^{(n)}$ concordam, o expoente em $e^{-y^{(n)}\,\mathbf{w}^T \mathbf{x}^{(n)}}$ é negativo $\rightsquigarrow$ $err(y^{(n)}, \hat{y}^{(n)})$ tende a ser próximo de zero
  - Se $y^{(n)}$ e $\mathbf{w}^T \mathbf{x}^{(n)}$ discordam, o expoente em $e^{-y^{(n)}\,\mathbf{w}^T \mathbf{x}^{(n)}}$ é positivo $\rightsquigarrow$ $err(y^{(n)}, \hat{y}^{(n)})$ tende a ser grande

Problema de Otimização: Entropia Cruzada

- Em vez de $Y = \{-1, +1\}$ podemos considerar $Y = \{0, 1\}$ e escrever:

$$\begin{aligned} P(y|\mathbf{x}) &= P(y=1|\mathbf{x})^y \, P(y=0|\mathbf{x})^{1-y} \\ &= P(y=1|\mathbf{x})^y \, [1 - P(y=1|\mathbf{x})]^{1-y} \end{aligned}$$

- Função de máxima verossimilhança (tirando o índice (n) para limpar a notação)

$$\begin{aligned} \prod_{(\mathbf{x},y)\in D} P(y|\mathbf{x}) &= \prod_{(\mathbf{x},y)\in D} P(y=1|\mathbf{x})^y \, [1 - P(y=1|\mathbf{x})]^{1-y} \\ &\approx \prod_{(\mathbf{x},y)\in D} [\sigma(\mathbf{w}^T\mathbf{x})]^y \, [1 - \sigma(\mathbf{w}^T\mathbf{x})]^{1-y} \\ &= \prod_{(\mathbf{x},y)\in D} \hat{y}^y \, (1 - \hat{y})^{1-y} \end{aligned}$$

Problema de Otimização: Entropia Cruzada

- Maximizar

$$\prod_{(\mathbf{x},y)\in D} \hat{y}^y (1-\hat{y})^{1-y}$$

- é minimizar

$$-\ln \prod_{(\mathbf{x},y)\in D} \hat{y}^y (1-\hat{y})^{1-y}$$

- Que é equivalente a minimizar

$$-\sum_{(\mathbf{x},y)\in D} \ln\left(\hat{y}^y (1-\hat{y})^{1-y}\right)$$

$$-\sum_{(\mathbf{x},y)\in D} \ln(\hat{y}^y) + \ln((1-\hat{y})^{1-y})$$

$$-\sum_{(\mathbf{x},y)\in D} y \ln \hat{y} + (1-y)\ln(1-\hat{y})$$

Problema de Otimização: Entropia Cruzada

- Função de Custo - Entropia Cruzada (cross-entropy):

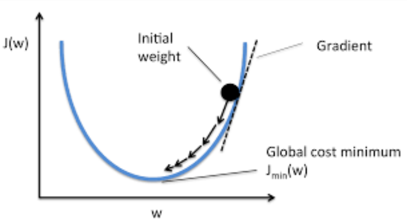
$$J(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} y^{(n)} \ln \hat{y}^{(n)} + (1 - y^{(n)}) \ln(1 - \hat{y}^{(n)})$$

Otimização utilizando Gradient Descent

Gradient Descent

- Muitos nomes em Português: método do gradiente, método do máximo declive, gradiente descendente
- Algoritmo em linha gerais:
  - Seja $J(\mathbf{w})$ a função custo a ser minimizada
  - Chutar um valor para $\mathbf{w}$
  - Calcular o gradiente de J no ponto $\mathbf{w}$ ("direção de maior inclinação")
  - Alterar $\mathbf{w}$ no sentido oposto ao do vetor gradiente
  - Repetir até atingir algum critério de convergência ou número de iterações

Ilustração do Gradient Descent



Exemplo: Regressão Linear

- Função de custo a ser otimizada com GD:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\Big(y^{(n)} - \underbrace{h_{\mathbf{w}}(\mathbf{x}^{(n)})}_{\hat{y}^{(n)}=\mathbf{w}^T\mathbf{x}^{(n)}}\Big)^2$$

- Solução iterativa no lugar da solução analítica

Exemplo: Regressão Linear

- Vetor gradiente de J:

$$\nabla J(\mathbf{w}) = \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_d}\right]^T$$

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_n (y^{(n)} - \hat{y}^{(n)})^2 \\
&= \frac{1}{2}\sum_n \frac{\partial}{\partial w_j}(y^{(n)} - \hat{y}^{(n)})^2 \\
&= \frac{1}{2}\sum_n 2(y^{(n)} - \hat{y}^{(n)})\frac{\partial}{\partial w_j}(y^{(n)} - \hat{y}^{(n)}) \\
&= \sum_n (y^{(n)} - \hat{y}^{(n)})\frac{\partial}{\partial w_j}\Big(y_n - \big(w_0 + w_1 x_1^{(n)} + \ldots + w_j x_j^{(n)} + \ldots + w_d x_d^{(n)}\big)\Big) \\
&= -\sum_n (y^{(n)} - \hat{y}^{(n)})\, x_j^{(n)}
\end{aligned}$$

(o componente j do vetor gradiente depende do erro $(y^{(n)} - \hat{y}^{(n)})$ e os componentes j de todos os pontos $\mathbf{x}^{(n)}$)

Exemplo: Regressão Linear

- Solução com Gradient Descent:
  - Vetor gradiente de J: $\nabla J(\mathbf{w}) = \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_d}\right]$

    $\frac{\partial J}{\partial w_j} = -\sum_n (y^{(n)} - \hat{y}^{(n)})\, x_j^{(n)}$

Peso inicial: $\mathbf{w}(0)$

Regra de atualização (iteração r):

$$\mathbf{w}(r+1) = \mathbf{w}(r) + \eta \Delta \mathbf{w}(r)$$

$$\Delta \mathbf{w}(r) = -\nabla J(\mathbf{w}), \qquad \Delta w_j(r) = \sum_n (y^{(n)} - \hat{y}^{(n)})\, x_j^{(n)}$$

η : taxa de aprendizado (em geral um valor pequeno)

Pseudo-código do Gradient Descent

Algorithm 1 GradientDescent

Input: D, η, *iter*

Output: $\mathbf{w}$

$\mathbf{w} \leftarrow$ small random value

repeat

$\Delta w_j \leftarrow 0, \quad j = 0, 1, 2, \ldots, d$

for all $(\mathbf{x}, y)$ in D **do**

compute $\hat{y} = \mathbf{w}^T \mathbf{x}$

$\Delta w_j \leftarrow \Delta w_j + (y - \hat{y})\, x_j, \quad j = 0, 1, 2, \ldots, d$

end for

$w_j \leftarrow w_j + \eta \Delta w_j, \quad j = 0, 1, 2, \ldots, d$

until number of iterations $=$ *iter*

Exemplo: Regressão Logística

- Função de custo:

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} y^{(n)} \ln \hat{y}^{(n)} + (1 - y^{(n)}) \ln(1 - \hat{y}^{(n)})$$

$$\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

- Derivadas parcias: $\dfrac{\partial}{\partial w_j} J(\mathbf{w}) = \displaystyle\sum_{n=1}^{N} (\hat{y}^{(n)} - y^{(n)}) x_j^{(n)}$
- Update de peso: $\Delta w_j(r) = \displaystyle\sum_{n=1}^{N} (y^{(n)} - \hat{y}^{(n)}) \mathbf{x}_j^{(i)}$

Próxima Aula: Implementando Regressão Linear e Logística do Zero