

Regressão Logística

Prof. Dr. Thiago de Paulo Faleiros

Roteiro da apresentação

1 Regularização

Regressão Logística

Ferramenta analítica importante na natureza e ciência social
Baseline para a tarefa de classificação supervisionada
É também a fundação de redes neurais

Generativo e Discriminativo

Naive Bayes é um classificador Generativo

Por outro lado

Regressão Logística é discriminativo

Generativo e Discriminativo

Suponha que estamos distinguindo cachorro de cão



imagenet



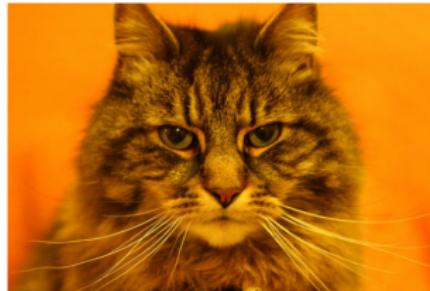
imagenet

Classificador Generativo

- Construa um modelo do que é uma imagem de gato
 - Sabe sobre bigode, orelhas, olhos
 - Atribui uma probabilidade a imagem
- Construa um modelo do que é uma imagem de cachorro
- Agora dada uma nova imagem
 - Rode ambos os modelos e veja qual é o melhor.

Classificador Discriminativo

Apenas tente distinguir cachorro de gatos



Veja, cachorro tem coleira!
Vamos ignorar todo o resto

Procurando a classe c correta para o documento d no classificador Discriminativo vs Generativo

Naive Bayes

$$c = \arg \max_{c \in C} P(d|c)P(c)$$

onde $P(c)$ é a priori

Regressão Logística

$$c = \arg \max_{c \in C} P(c|d)$$

onde $P(c|d)$ é a posteriori

Componentes de um classificador probabilístico

Dado m pares de entradas/saídas $(x^{(i)}, y^{(i)})$:

- ➊ As características que representam a entrada. Para cada entrada observada $x^{(i)}$, um vetor de características $[x_1, x_2, x_3, \dots, x_n]$. A característica j para a entrada $x^{(i)}$ é x_j , $x_j^{(i)}$, ou $f_j(x)$.
- ➋ Uma função de classificação que computa \bar{y} , a classe estimada, via $P(y|x)$, como as funções sigmoid ou softmax.
- ➌ Uma função objetivo, como entropia cruzada (**cross-entropy loss**).
- ➍ Um algoritmo de otimização para a função objetivo: *stochastic gradient descent*.

Duas fases da regressão logística

Treinamento: Aprendemos os pesos w e b usando *stochastic gradient descent* e *cross-entropy loss*.

Teste: Dado um exemplo de teste x , computamos $P(y|x)$ usando os pesos aprendidos w e b , e retornamos qualquer rótulo com maior probabilidade.

Relembrando Classificação

análise de sentimentos (Positivo/Negativo)
Spam/não spam

Definição da tarefa de classificação

Entrada:

- um documento x
- um conjunto fixo de classes $C = \{c_1, c_2, \dots, c_J\}$

Saída: uma classe predita $\bar{y} \in C$

Classificador Binário em Regressão Logística

Dada uma série de pares entradas/saídas:

- $(x^{(i)}, y^{(i)})$

Para cada observação $x^{(i)}$

- Nós representamos $x^{(i)}$ por um vetor de características $[x_1, x_2, \dots, x_n]$
- Nós computamos uma saída: um preditor de classe $\bar{y}^{(i)} \in \{0, 1\}$

Características em Regressão Logística

- Para característica x_i , o peso w_i diz o quanto importante é x_i
 - $x_i = \text{contém a palavra "awesome"}: w_i = +10$
 - $x_j = \text{contém a palavra "absmado"}: w_j = -10$
 - $x_k = \text{contém a palavra "medíocre"}: w_k = -2$

Regressão Logística para uma observação x

Entrada observada: vetor $x = [x_1, x_2, \dots, x_n]$

Pesos: um por característica: $W = [w_1, w_2, \dots, w_n]$

- Algumas vezes os pesos são descritos como $\theta = [\theta_1, \theta_2, \dots, \theta_n]$

Saída: a classe predita $\bar{y} \in \{0, 1\}$

(regressão logística multinomial: $\bar{y} \in \{0, 1, 2, 3, 4\}$)

Como classificar

Para cada feature x_i , o peso w_i diz a importância do x_i

- Temos o bias b

Soma dos pesos e do bias

$$z = \left(\sum_{i=1}^n w_i x_i + b \right)$$

$$z = w \cdot x + b$$

Se a soma é alta, dizemos que $y = 1$; se baixo, então $y = 0$

Mas queremos um classificador probabilístico

Precisamos formalizar “a soma é alta”

Semelhante ao Naive Bayes

Queremos um modelo que nos diga:

$$p(y = 1|x, \theta)$$

$$p(y = 0|x, \theta)$$

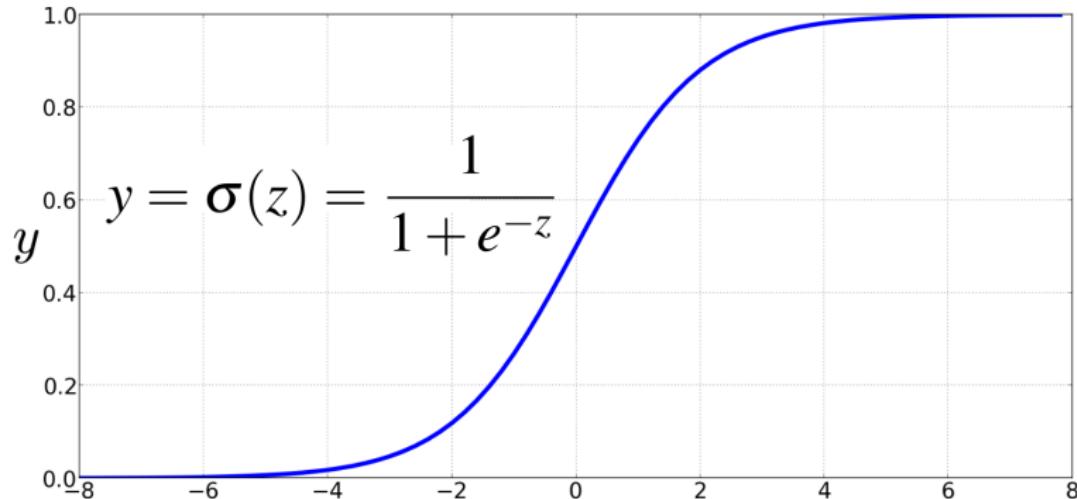
O problema: z não é uma probabilidade, é apenas um número

$$z = x \cdot x + b$$

Solução: use uma função de z que vai de 0 até 1

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

Funções sigmoid e softmax



Ideal para regressão logística

Iremos computar $w \cdot x + b$
E então passar pela função sigmoid

$$\sigma(w \cdot x + b)$$

E então tratar como probabilidade

Criando valores de probabilidade com sigmoid

$$\begin{aligned}P(y = 1) &= \sigma(w \cdot x + b) \\&= \frac{1}{1 + \exp(-(w \cdot x + b))}\end{aligned}$$

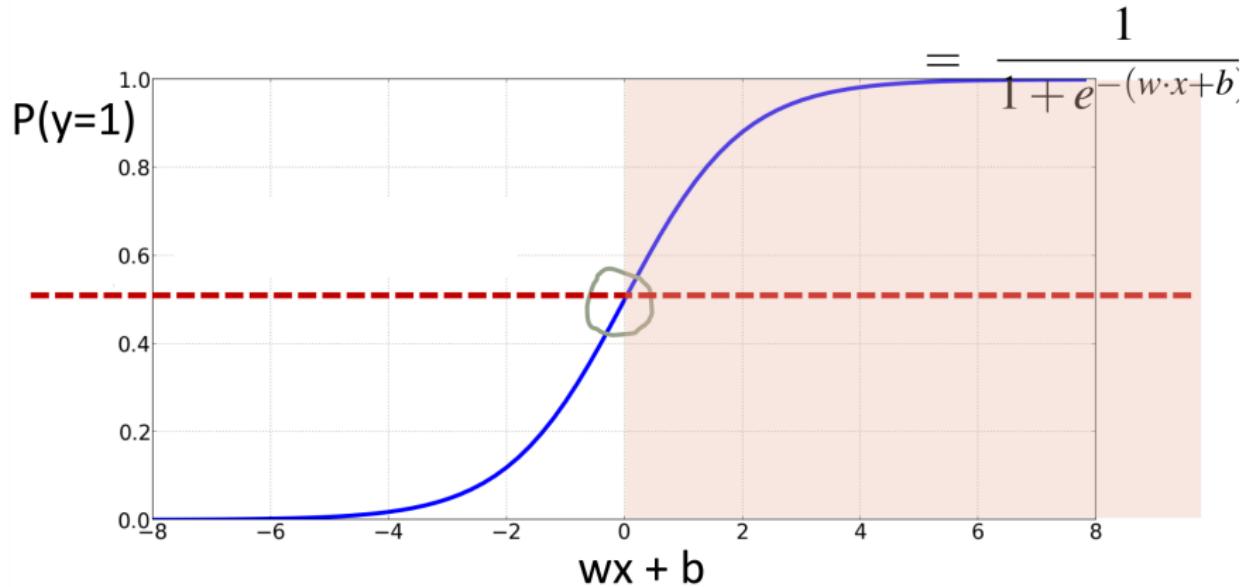
$$\begin{aligned}P(y = 0) &= 1 - \sigma(w \cdot x + b) \\&= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))} \\&= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))}\end{aligned}$$

Tortando probabilidade em classe

$$\bar{y} = \begin{cases} 1 & \text{se } P(y=1|x) > 0.5 \\ 0 & \text{caso contrário} \end{cases}$$

0.5 é definido como borda de limite

The probabilistic classifier $P(y=1) = \sigma(w \cdot x + b)$



Criando valores de probabilidade com sigmoid

$$\bar{y} = \begin{cases} 1 & \text{se } P(y=1|x) > 0.5 \\ 0 & \text{caso contrário} \end{cases} \quad \begin{array}{ll} \text{se } w \cdot x > 0 & \\ \text{se } w \cdot x \leq 0 & \end{array}$$

Exemplo – análise de sentimento

It's **hokey**. There are virtually **no** surprises , and the writing is **second-rate**.
 So why was it so **enjoyable**? For one thing , the cast is
great. Another **nice** touch is the music **I** was overcome with the urge to get off
 the **couch** and start/dancing . It sucked **me** in , and it'll do the same to **you** .

$x_1=3$ $x_5=0$ $x_6=4.19$ $x_2=2$ $x_3=1$ $x_4=3$

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) \in doc	3
x_2	count(negative lexicon) \in doc)	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns \in doc)	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\ln(\text{word count of doc})$	$\ln(66) = 4.19$

Var	Definition	Val
x_1	count(positive lexicon) \in doc)	3
x_2	count(negative lexicon) \in doc)	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns \in doc)	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\log(\text{word count of doc})$	$\ln(66) = 4.19$

Suppose $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$

$$b = 0.1$$

Classificando sentimento para entrada X

$$\begin{aligned}P(+|x) &= P(Y = 1|x) = \sigma(w \cdot x + b) \\&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\&= \sigma(0.833) \\&= 0.70\end{aligned}$$

$$\begin{aligned}P(-|x) &= P(Y = 0|x) = 1 - \sigma(w \cdot x + b) \\&= 0.30\end{aligned}$$

Podemos construir característica para a regressão logística

This ends in a period.



The house at 465 Main St. is new.



Not end

$$x_1 = \begin{cases} 1 & \text{se } \text{Caso}(w_i) = \text{Lower} \\ 0 & \text{caso contrário} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{se Acrônimo } w_i \\ 0 & \text{caso contrário} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{se } w_i = \text{St.} \& \text{Caso}(w_i) \\ 0 & \text{caso contrário} \end{cases}$$

Classificação binária para Regressão Logística

Dado:

- um conjunto de classe $(+, -)$
- um vetor de características $x = [x_1, x_2, \dots, x_n]$
 - $x_1 = \text{count}(\text{awesome})$
 - $x_2 = \log(\text{número de palavras})$
- Um vetor de pesos $w = [w_1, w_2, \dots, w_n]$
 - w_i para cada característica f_i

$$\begin{aligned}P(y = 1) &= \sigma(w \cdot x + b) \\&= \frac{1}{1 + e^{-(w \cdot x + b)}}\end{aligned}$$

De onde os pesos W vieram?

- Classificação supervisionada
 - Conhecemos o valor correto de y (0 ou 1) para cada x
 - Mas o que o sistema produz é uma estimativa \bar{y}
- Queremos definir o valor de w e b para minimizar a distância entre nossa estimativa \bar{y} e o valor real y
 - Precisamos de um estimador de distância: a **loss function** ou **cost function**
- Precisamos de um algoritmo de otimização que atualiza os pesos w e b para minimizar a perda

Componentes de Aprendizagem

A função de perda

- **cross-entropy loss**

Um algoritmo de otimização

- **stochastic gradient descent**

A distância entre \hat{y} e y

Queremos saber o quanto distante está a resposta do classificador

$$\hat{y} = \sigma(w \cdot x + b)$$

da resposta verdadeira

$$y \quad [= 0 \text{ ou } 1]$$

Chamaremos essa diferença:

$$L(\hat{y}, y) = \text{o quanto } \hat{y} \text{ difere de } y$$

Intuição do negative log likelihood loss = cross-entropy loss

Estimador condicional de máxima verossimilhança

Escolhemos os parâmetros w , b que maximiza

- log-probability
- o rótulo verdadeira y no conjunto de treino
- dada a observação x

Derivando a entropia cruzada para a simples observação x

Objetivo: maximizar a probabilidade do rótulo correto $p(y|x)$

Desde que existam apenas 2 resultados discretos (0 ou 1) podemos expressar a probabilidade $p(y|x)$ do nosso classificador (o que queremos maximizar) como

$$p(y|x) = \hat{y}^y(1 - \hat{y})^{1-y}$$

note que:

se $y = 1$, temos \hat{y}

se $y = 0$, temos $1 - \hat{y}$

Derivando a entropia cruzada para uma simples observação x

Objetivo: maximiza a probabilidade do rótulo $p(y|x)$

Maximize: $p(y|x) = \hat{y}^y(1 - \hat{y})^{1-y}$

tirando o log

Maximize:

$$\begin{aligned}\log p(y|x) &= \log [\hat{y}^y(1 - \hat{y})^{1-y}] \\ &= y \log \hat{y} + (1 - y) \log (1 - \hat{y})\end{aligned}$$

Qualquer valor que maximize $\log p(y|x)$ irá maximizar também $p(y|x)$.

Derivando a entropia cruzada para uma simples observação x

Objetivo: maximiza a probabilidade do rótulo $p(y|x)$

Maximize:

$$\begin{aligned}\log p(y|x) &= \log [\hat{y}^y (1 - \hat{y})^{1-y}] \\ &= y \log \hat{y} + (1 - y) \log (1 - \hat{y})\end{aligned}$$

Agora mudando o sinal: tornando em problema em minimização

Cross-entropy loss – Minimização

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

Ou,

$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))]$$

Vamos ver se isso funciona para nosso exemplo para análise de sentimentos

Queremos que a perda seja:

- Menor se a estimativa do modelo é próxima da correta
- Maior se o modelo é confuso

Vamos assumir o rótulo verdadeira $y = 1$ (positivo)

It's hokey . There are virtually no surprises , and the writing is second-rate . So why was it so enjoyable ? For one thing , the cast is great . Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you .

Vamos ver se funciona para nosso exemplo

O valor verdadeiro é $y = 1$. O quanto bom é nosso modelo?

$$\begin{aligned} p(+|x) &= p(Y = 1|x) = \sigma(w \cdot x + b) \\ &= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\ &= \sigma(0.833) \\ &= 0.70 \end{aligned}$$

Muito bom! Qual é a perda?

$$\begin{aligned} L_{CE}(\hat{y}, y) &= [y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\ &= -[\log \sigma(w \cdot x + b)] \\ &= -\log(0.70) \\ &= 0.36 \end{aligned}$$

Vamos ver se funciona para nosso exemplo

Suponha verdadeiro para $y=0$

$$\begin{aligned} p(-|x) &= p(Y = 0|x) = 1 - \sigma(w \cdot x + b) \\ &= 0.30 \end{aligned}$$

Qual é a perda?

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\ &= -[\log(1 - \sigma(w \cdot x + b))] \\ &= -\log(0.30) \\ &= 1.2 \end{aligned}$$

Vamos ver se funciona para nosso exemplo

A perda quando o modelo está certo ($y=1$)

$$\begin{aligned}L_{CE}(\hat{y}, y) &= [y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\&= -[\log \sigma(w \cdot x + b)] \\&= -\log(0.70) \\&= 0.36\end{aligned}$$

É menor quando o modelo está errado ($y=0$)

$$\begin{aligned}L_{CE}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\&= -[\log(1 - \sigma(w \cdot x + b))] \\&= -\log(0.30) \\&= 1.2\end{aligned}$$

A perda é maior quando o modelo está errado!

Nosso objetivo: Minimizar a perda

Vamos deixar explícito que a função de perda é parametrizada pelo peso $\theta = (w, b)$

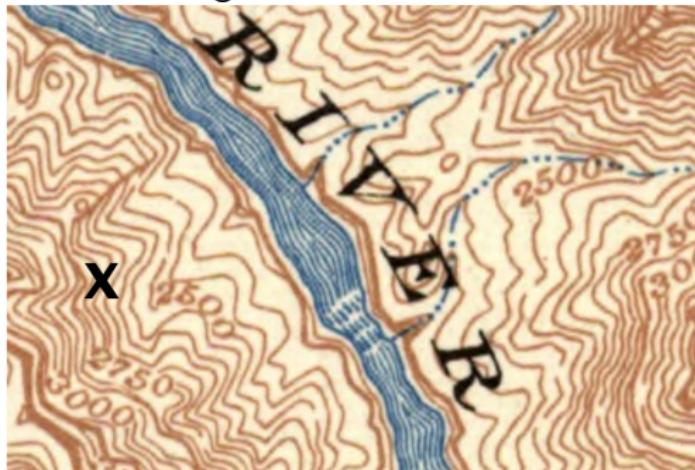
- E iremos representar \hat{y} como $f(x; \theta)$ para fazer a dependência de θ mais óbvia.

Queremos os pesos que minimizam a perda, média entre todos os exemplos

$$\theta = \arg \min_{\theta} \frac{1}{m} \sum_{i=1}^m L_{CE} \left(f(x^{(i)}, \theta), y^{(i)} \right)$$

Intuição de Gradiente descendente

Como chego ao fundo deste cânion fluvial?



Olhe em volta
Encontre a direção
declive mais
íngreme para baixo
Vá naquela direção

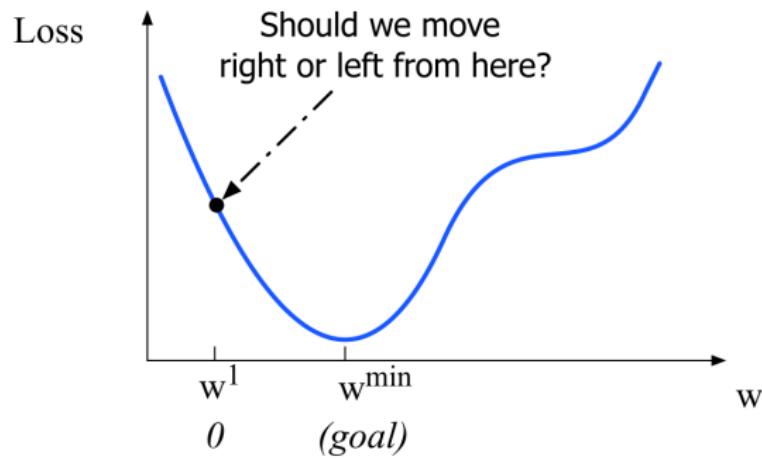
Nosso objetivo: minimizar a loss

Para regressão logística, a função de perda é convexa

- Uma função convexa tem apenas um mínimo
- O Gradiente descendente começando de qualquer ponto tem garantia de encontrar o mínimo
 - (Loss para redes neurais artificiais é não-convexa)

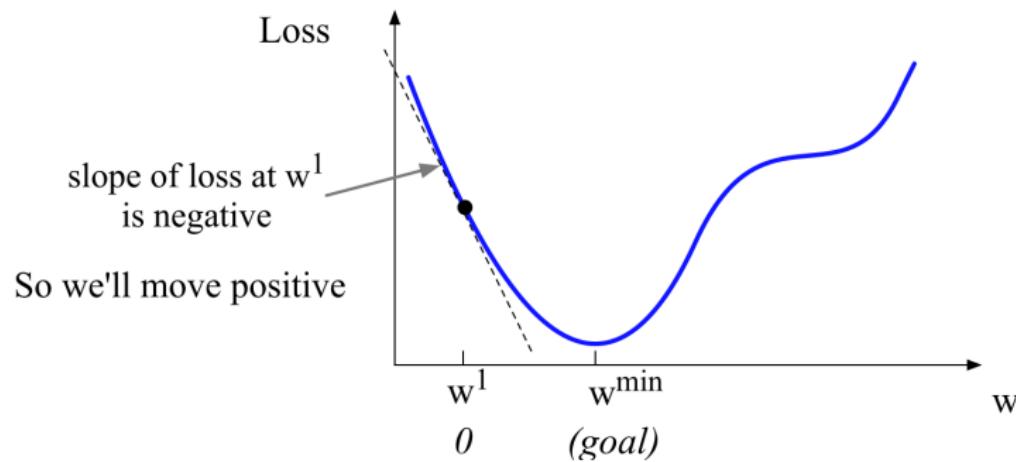
Vamos visualizar para um escalar simples w

Q: Dado um peso corrente w , podemos fazer esse valor menor ou maior? A: Mova w na direção reversa do declive da função



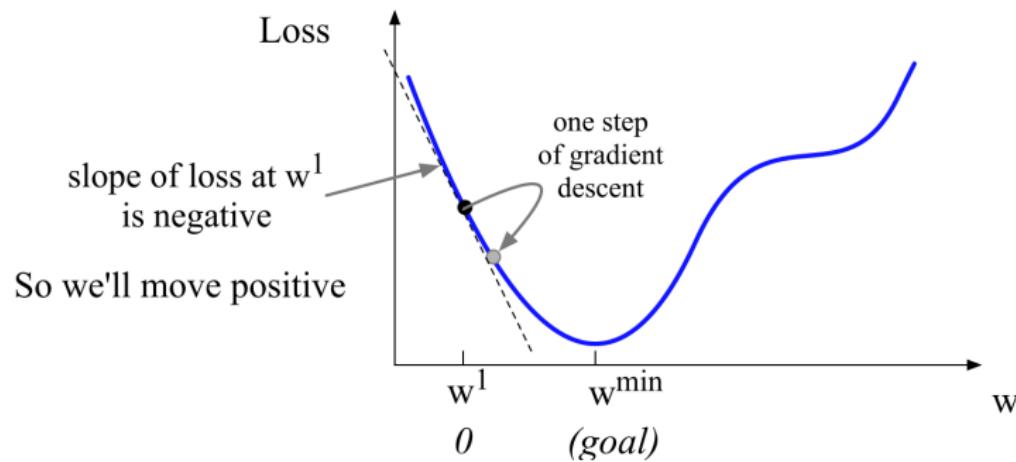
Vamos visualizar para um escalar simples w

Q: Dado um peso corrente w , podemos fazer esse valor menor ou maior? A: Mova w na direção reversa do declive da função



Vamos visualizar para um escalar simples w

Q: Dado um peso corrente w , podemos fazer esse valor menor ou maior? A: Mova w na direção reversa do declive da função



Gradientes

O **gradiente** de uma função de múltiplas variáveis é um vetor na direção do maior incremento da função

Gradiente descendente: Encontre o gradiente da função de perda do ponto corrente e move na direção oposta.

O quanto movemos naquela direção?

- O valor do gradiente $\frac{d}{dw} L(f(x, w), y)$ é ponderado pela taxa de aprendizagem η
- Maior a taxa de aprendizagem mais rápido w se move

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x, w), y)$$

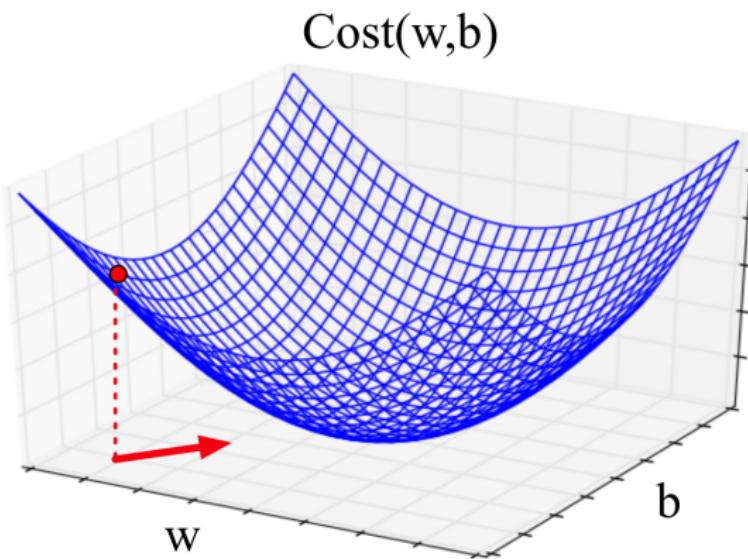
Agora vamos considerar N dimensões

Queremos saber onde no espaço N -dimensional (dos N parâmetros que montam θ) podemos mover.

O gradiente é tal vetor; ele expressa a direção do componente com o declive mais acentuado ao longo de cada dimensão.

Imagine 2 dimensões, w e b

Visualizando o vetor do gradiente no ponto vermelho
Ele tem duas dimensões no plano $x-y$



Gradiente real

São muito mais longos; vários e vários pesos

Para cada dimensão w_i o componente i do gradiente nos diz o declive em relação àquela variável.

- “O quanto uma pequena mudança no w_i irá influenciar a função total de perda L ? ”
- Expressamos o declive como derivada parcial ∂ da perda ∂w_i .

O gradiente é então definido como um vetor dessas parciais

O gradiente

Iremos representar \hat{y} como a função $f(x; \theta)$ para fazer a dependência em θ mais óbvia:

$$\nabla_{\theta} L(f(x, \theta), y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x, \theta), y) \\ \frac{\partial}{\partial w_2} L(f(x, \theta), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x, \theta), y) \end{bmatrix}$$

A equação final de atualização de θ é baseada no seguinte gradiente

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x, \theta), y)$$

Quais são as derivadas para a Regressão Logística

A função de perda:

$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x) + b)]$$

A derivada é a seguinte:

$$\frac{\partial L_{CE}(\hat{y}, y)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j$$

```
function STOCHASTIC GRADIENT DESCENT( $L()$ ,  $f()$ ,  $x$ ,  $y$ ) returns  $\theta$ 
```

where: L is the loss function

f is a function parameterized by θ

x is the set of training inputs $x^{(1)}$, $x^{(2)}$, ..., $x^{(m)}$

y is the set of training outputs (labels) $y^{(1)}$, $y^{(2)}$, ..., $y^{(m)}$

$\theta \leftarrow 0$

repeat til done

For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

1. Optional (for reporting): # How are we doing on this tuple?

Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output \hat{y} ?

Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?

2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?

3. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

Hiperparâmetros

A taxa de aprendizagem η é um hiperparâmetro

- muito alta: o aprendizado dará grandes passos
- muito baixa: o aprendizado irá demorar muito

Hiperparâmetros:

- um tipo especial de parâmetro para modelos de ML
- Em vez de ser aprendido pelo algoritmo, eles são escolhidos pelo projetista do algoritmo

Exemplo

Um passo da descida do gradiente

Um pequeno exemplo, onde o valor verdadeiro é $y = 1$

Duas características:

- $x_1 = 3$ (conta os lexicons positivos)
- $x_2 = 2$ (conta os lexicons negativos)

Assume 3 parâmetros (2 pesos e 1 bias) em θ^0 iniciando com zero:

- $w_1 = w_2 = b = 0$
- $\eta = 0.1$

Exemplo

$$w_1 = w_2 = b = 0; \quad x_1 = 3; \quad x_2 = 2$$

Passo de atualização para θ é:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x, \theta), y)$$

onde $\frac{\partial L_{CE}(\hat{y}, y)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j$

O gradiente do vetor tem 3 dimensões:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{CE}(\hat{y}, y)}{\partial w_1} \\ \frac{\partial L_{CE}(\hat{y}, y)}{\partial w_2} \\ \frac{\partial L_{CE}(\hat{y}, y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{CE}(\hat{y}, y)}{\partial w_1} \\ \frac{\partial L_{CE}(\hat{y}, y)}{\partial w_2} \\ \frac{\partial L_{CE}(\hat{y}, y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

Agora que temos os gradientes, vamos computar o novo vetor de parâmetros θ^1 movendo θ^0 na direção oposta do gradiente:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x, \theta), y)$$

para $\eta = 0.1$

$$\theta^1 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.1 \\ 0.05 \end{bmatrix}$$

Note que exemplos negativos poderia fazer w_2 ficar negativo

Treinamento em Mini-batch

- O algoritmo de descida de gradiente escolhe aleatoriamente um simples exemplo por vez
- Isso pode resultar em um momento de distúrbio
- É comum computar o gradiente sobre batches de instâncias de treinamento
- **Batch training:** conjunto inteiro
- **Mini-batch training:** m amostras (512 ou 1024)

Overfitting

Um modelo que perfeitamente “se encaixa” perfeitamente no conjunto de treinamento tem um problema.

Esse modelo pode se sobreajustar sobre os dados

- Uma palavra aleatória que perfeitamente prediz y (isso pode acontecer por que ocorre apenas uma vez na classe) pode ter um peso alto
- Falhando ao generalizar no conjunto de teste (que não tem essa palavra)

Um bom modelo deve ser capaz de generalizar

Overfitting

+

Esse filme me cativou e irá fazer o mesmo com você.

-

Eu não consigo dizer o quanto odiei esse filme. Ele é péssimo.

Características (úteis ou danosas ao modelo?)

- x_1 = “Esse”
- x_2 = “filme”
- x_3 = “odiei”
- x_4 = “me cativou”
- x_5 = “o memo com você”
- x_6 = “dizer o quanto”

Overfitting

- O modelo 4 – *grams* com poucos dados irá apenas memorizar
 - 100% de acurácia no conjunto de treinamento
- Mas será surpreendido com os dados de teste
 - Baixa acurácia no conjunto de teste
- Modelos que são muitos poderosos podem ter sobreajustes nos dados
 - Ajustando tão forte aos detalhes do dados de treinamento que o modelo não generaliza bem para o conjunto de teste
 - Como evitar overfitting?
 - Regularização em regressão logística
 - Dropout em redes neurais

Regularização

Uma solução para overfitting

Adicione termo de regularização $R(\theta)$ para a função de perda

$$\hat{\theta} = \arg \max_{\theta} \left(\sum_{i=1}^m \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta) \right)$$

Ideia: escolha um $R(\theta)$ que penaliza pesos grandes

Regularização L2

A soma dos quadrados dos pesos

norma L2 $||\theta||_2$ = distância euclidiana de θ para a origem

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^n \theta_j^2$$

Função objetivo com regularização L2:

$$\hat{\theta} = \arg \max_{\theta} \left[\left(\sum_{i=1}^m \log P(y^{(i)} | x^{(i)}) \right) - \alpha \sum_{j=1}^n \theta_j^2 \right]$$

Regularização L1

A soma absoluta dos pesos

norma L1 $||\theta||_1$ = distância de Manhattan

$$R(\theta) = ||\theta||_1 = \sum_{j=1}^n |\theta_j|$$

Função objetivo com regularização L1:

$$\hat{\theta} = \arg \max_{\theta} \left[\left(\sum_{i=1}^m \log P(y^{(i)} | x^{(i)}) \right) - \alpha \sum_{j=1}^n |\theta_j| \right]$$

Régressão Logística Multinomial

- Frequentemente precisamos de mais do que duas classes
 - Positivo, negativo e neutro
 - Parts of speech (noun, verb, adjective, adverb, preposition, etc.)
 - Classificar notícias
- Se temos mais do que duas classes usamos a régressão logística multinomial

Régressão Logística Multinomial

A probabilidade de tudo deve somar para 1

$$P(\text{positive}|\text{doc}) + P(\text{negativo} \mid \text{doc}) + P(\text{neutro} \mid \text{doc}) = 1$$

Generalização da função σ é o softmax

- Seja um z um vetor $[z_1, z_2, \dots, z_k]$ com k valores arbitrários
- Retorne a distribuição de probabilidades
 - cada item deve estar no intervalo $[0,1]$
 - todos os valores devem somar 1

A função softmax

Transforma o vetor $[z_1, z_2, \dots, z_k]$ de k valores arbitrários em probabilidades

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^k \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^k \exp(z_i)}, \dots, \frac{\exp(z_k)}{\sum_{i=1}^k \exp(z_i)} \right]$$

$$[0.055, 0.090, 0.0067, 0.10, 0.74, 0.010]$$

Softmax em regressão logística multinomial

$$p(y = c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^k \exp(w_j \cdot x + b_j)}$$

A entrada é ainda o produto entre o vetor de peso w e o vetor de entrada x

Mas precisamos separar os pesos para cada uma das k classes.

Características em regressão logística binária versus multinomial

- Binário: pesos positivos $\rightarrow y=1$ peso negativo $\rightarrow y=0$

$$x_5 = \begin{cases} 1 & \text{se “!”} \in doc \\ 0 & \text{caso contrário} \end{cases}$$

(peso $w_5 = 3.0$)

Multinomial: separa os pesos para cada classe:

Característica	Definição	$w_5, +$	$w_5, -$	w_5, n
$f_5(x)$	$\begin{cases} 1 & \text{se “!”} \in doc \\ 0 & \text{caso contrário} \end{cases}$	3.5	3.1	-5.3