# logistic regression

## 1.the function


`discriminative classifier` | `naive Bayes is a generative classifier`

| | |
|---|---|
| 判别模型 | $D \xrightarrow{\text{经验误差最小原则}} \text{决策边界}$ |
| 生成模型 | $D \rightarrow \text{正、负类的分布} \rightarrow \text{决策边界}$ |

区别在于最后模型的输出结果是$P(X,Y)|P(Y|X)$

LR, NB共同点都是基于有监督概率学的分类器:

基于给定的input feature $[x_1, x_2, \ldots, x_n]$有标注$f_i(x)$, $f_i$或者多分类标注$f_i(c,x)$预估类$\hat{y} = P(y|x)$

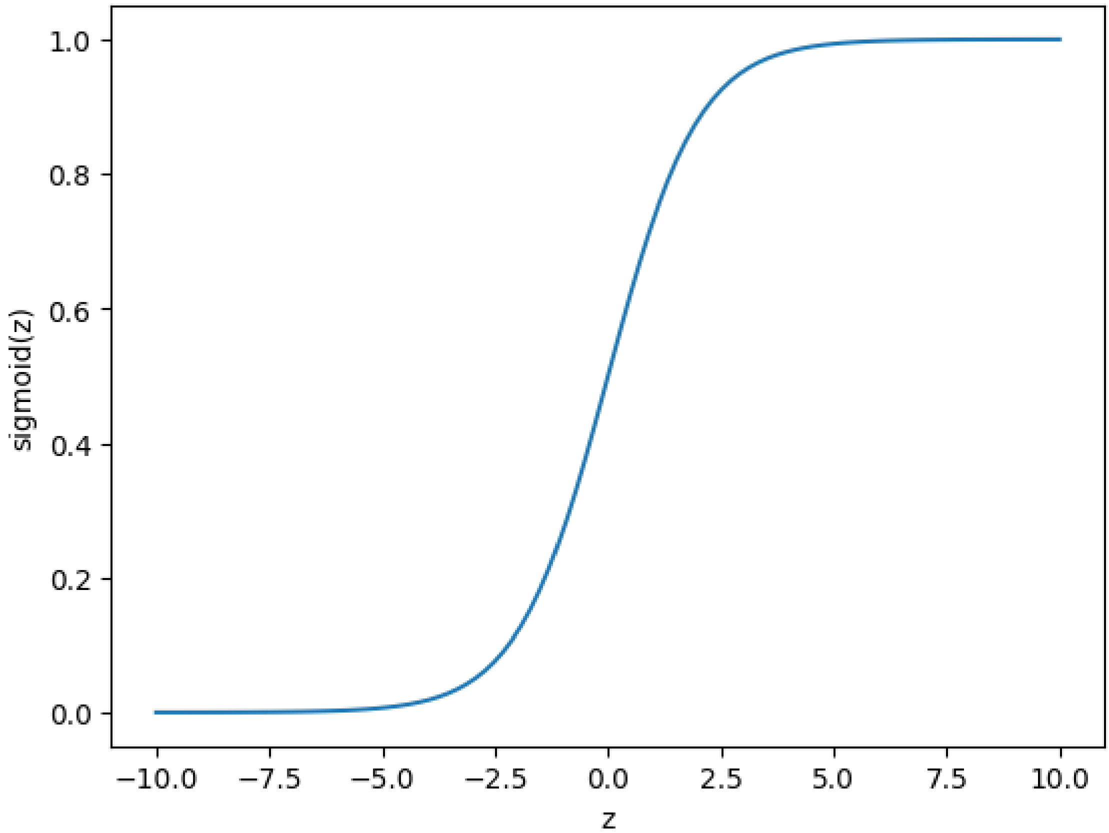
### sigmoid

$$z = (\sum_{i=1}^{n} w_i x_i) + \mathbf{b} = \mathbf{w} \cdot \mathbf{x} + \mathbf{b}$$

z表示的是对于不同输入特征的加权和在与偏差值bias相加的标量

直接相加导致z的范围尺度可能非常大 、scale z

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




将sigmoid(z)映射到概率分布上

$$P(y=1)=\delta(w\cdot x+b)$$
$$P(y=0)=1-\delta(w\cdot x+b)$$
$$also: 1-\sigma(x)=\sigma(-x)$$

example

# 2.classification with logistic function

## 情感分类

特征设计、特征分析、bias调整、特征模板

## 其他任务

input scaling and normalize

1.batch norm

2.layer norm

## 选择classifier

LR相较于朴素贝叶斯分类器有数值优势，后者需要较强的独立假设。

但是后者训练速度快不需要迭代参数

# 3.mutinomial logistic regression

## 多项式分类

1.硬分类

$$\text{given input } \mathbf{X} \text{ output}->\text{ one}-\text{hot vector}: \mathbf{v}[0,\ldots,1,\ldots 0]$$
$$\text{where } i_k = 1$$

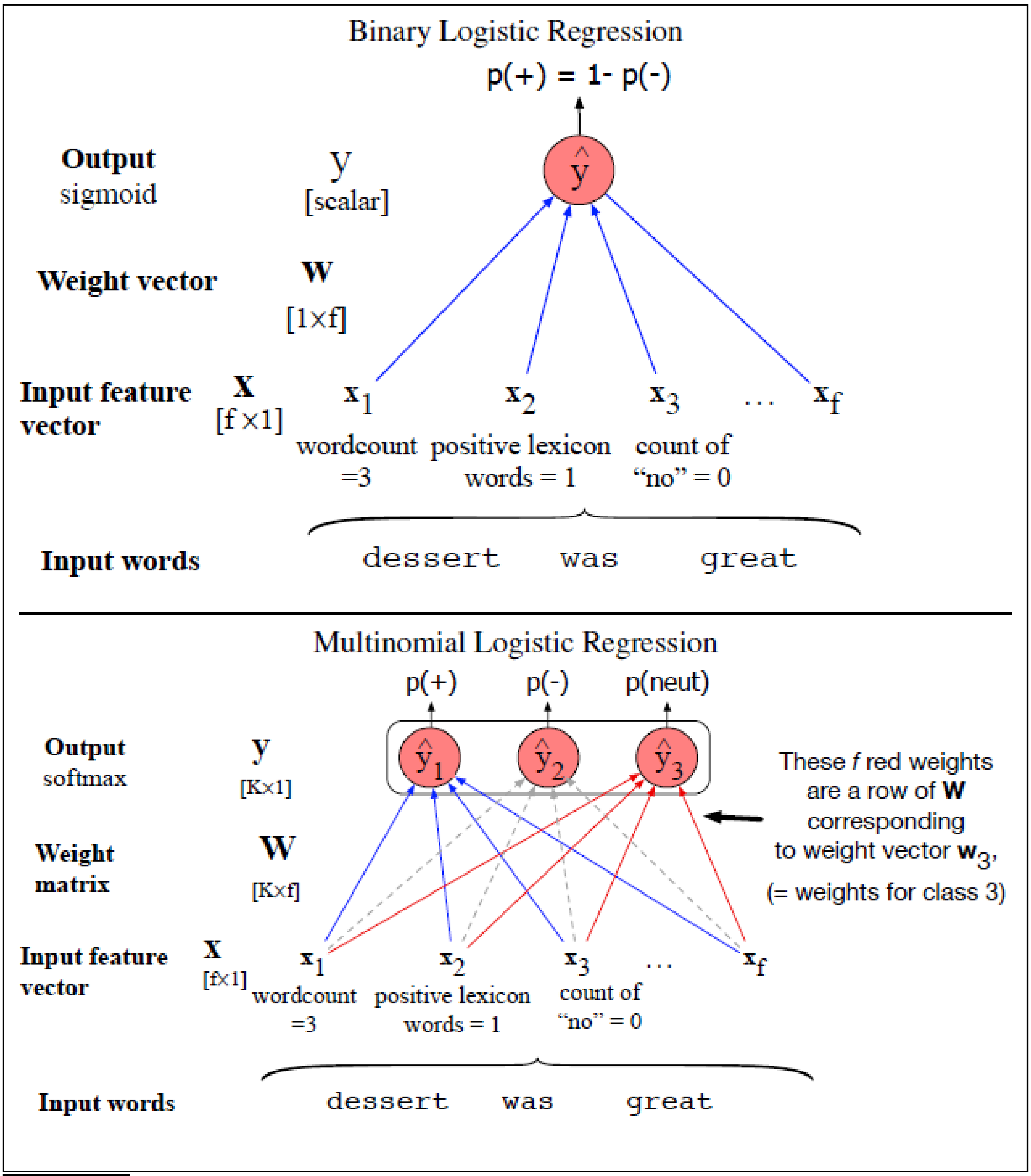
2.softmax

$$\hat{y} = \operatorname{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

exp非零性

$$\text{given input } \mathbf{X} \text{ output}->\text{ possiblity vector } \mathbf{v} = [\frac{\exp(z_i)}{\sum_{i=1}^{K}\exp(z_i)}]$$
$$\text{where } z = f_i(X)$$

softmax可归一并且可以根据sigmoid(x)完成多分类任务



## 4.learning in logistic regression

loss function gradient descent algorithm to updating weights and bias

## 5.the cross-entropy

given an observation $x$, we have output $\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + \mathbf{b})$ and correct output $y$

we need a function to be the metrics for how much differs from two outputs

1.conditional maximum likelihood estimation

for params $w, b$, and given observation x, the model should maximize the $\log(p(y = \hat{y}|x))$

对于简单的二分布模型满足伯努利分布：

$$p(y|x) = \hat{y}^y(1-\hat{y})^{1-y} \qquad y, \hat{y} \in (0,1)$$

after log function:

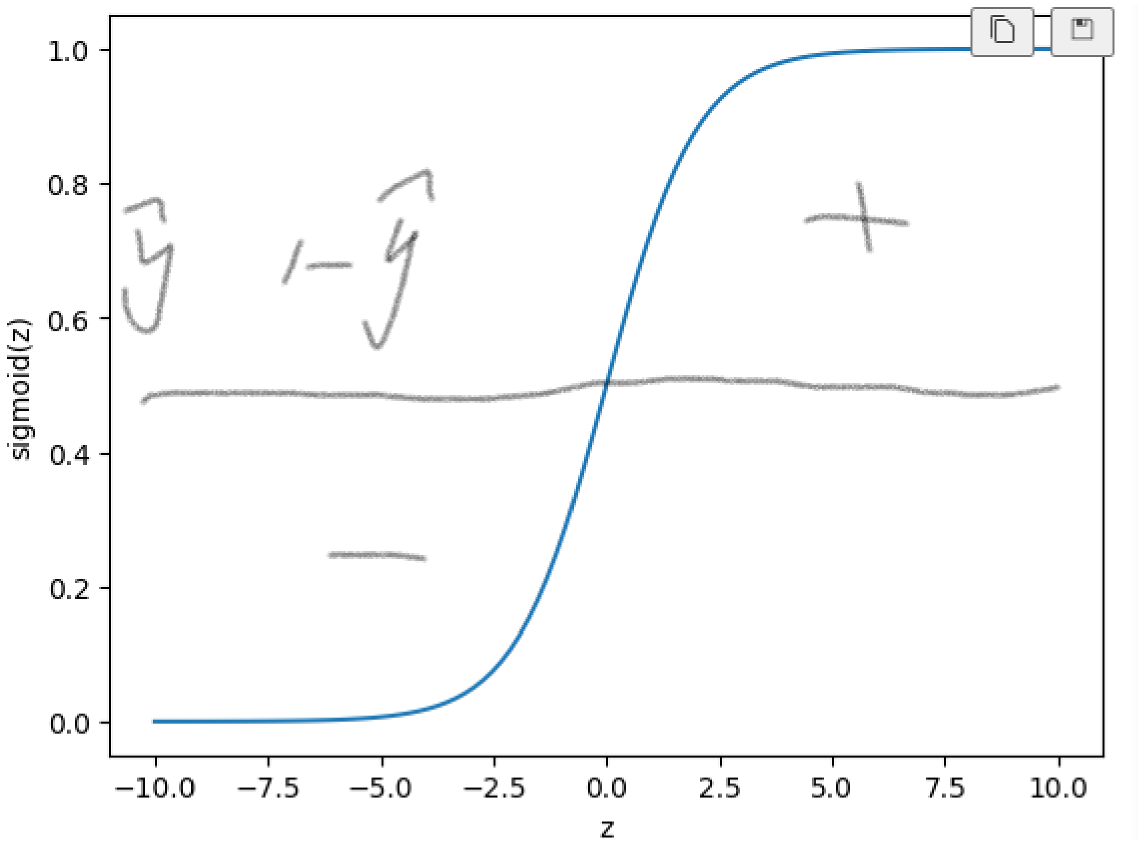
$$\log(p(y|x)) = \log p(y|x) = y\log\hat{y} + (1-y)\log(1-\hat{y})$$

使得上述最大即使得$-\log(p(y|x))$最小所以以此作为loss函数

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

extention:

$$L = \frac{1}{N}\sum_i L_i = -\frac{1}{N}\sum_i \sum_{c=1}^{M} y_{ic}\log p_{ic}$$



## 6.gradient descent

with gradient descent to find optimal weights and minimize the loss function for out model

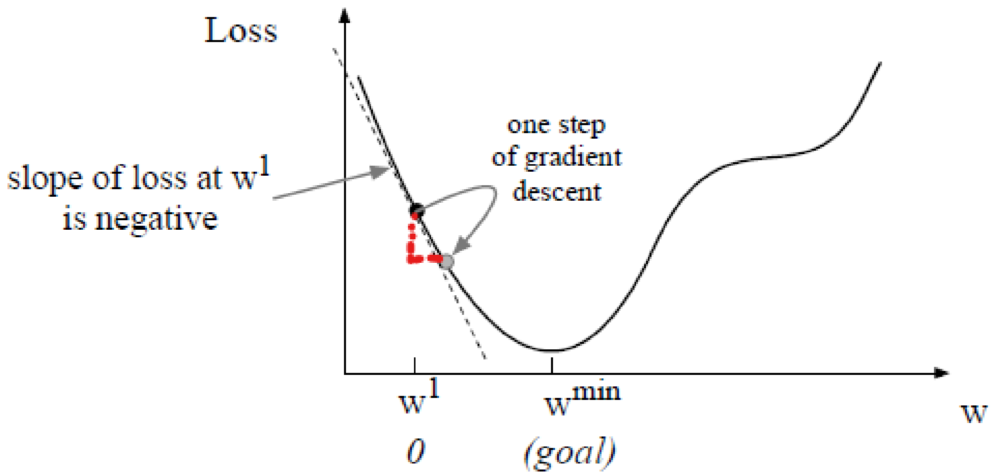
equation:

$$\hat{\theta} = \underset{\theta}{argmin} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(f(x^{(i)}; \theta), y^{(i)})$$

**more details about convex function**： [here]

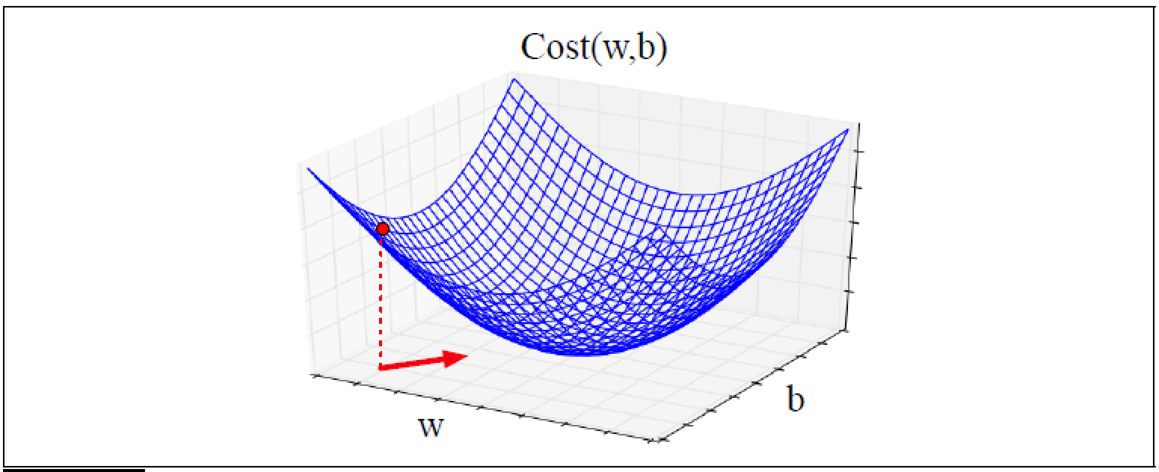
convex function: no local minimum

学习率：learning rate




$$
\begin{aligned}
w^{t+1} &= w^t - \eta \frac{d}{dw} L(f(x; w), y) \\
\mathbf{w} &= [w_1, \ldots, w_n] \\
gradient -> \nabla L(f(x; \theta), y) &= \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x; \theta), y) \\ \frac{\partial}{\partial w_2} L(f(x; \theta), y) \\ \ldots \\ \frac{\partial}{\partial w_n} L(f(x; \theta), y) \end{bmatrix} \\
\theta^{t+1} &= \theta^t - \eta \nabla L(f(x; \theta), y)
\end{aligned}
$$

$$L_{CE}(\hat{y}, y) \quad = \quad -[y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1 - y) \log (1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b))] \qquad (5.28)$$



对上式求$w_j$上的偏导

It turns out that the derivative of this function for one observation vector $x$ is Eq. 5.29 (the interested reader can see Section 5.10 for the derivation of this equation):

$$\begin{aligned}\frac{\partial L_{\mathrm{CE}}(\hat{y},y)}{\partial \mathbf{w}_j} &= [\sigma(\mathbf{w}\cdot\mathbf{x}+b)-y]\mathbf{x}_j \\ &= (\hat{y}-y)\mathbf{x}_j \end{aligned} \tag{5.29}$$

You'll also sometimes see this equation in the equivalent form:

$$\frac{\partial L_{\mathrm{CE}}(\hat{y},y)}{\partial \mathbf{w}_j} = -(y-\hat{y})\mathbf{x}_j \tag{5.30}$$

随机梯度下降：

**function** STOCHASTIC GRADIENT DESCENT($L()$, $f()$, $x$, $y$) **returns** $\theta$
  # where: L is the loss function
  #   f is a function parameterized by $\theta$
  #   x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
  #   y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$

$\theta \leftarrow 0$
**repeat** til done  # see caption
  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 $\theta$ to maximize loss?
    3. $\theta \leftarrow \theta - \eta\, g$  # Go the other way instead
return $\theta$

超参学习率$\eta$的选取

1.步幅太大可能在loss function极值点震荡无法到达

2.太大迭代次数多

`Mini-batch` | `batch`

batching_training

$$cost(\hat{y},y) = \frac{1}{m}\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)}, y^{(i)})$$

$$\frac{\partial Cost}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}[\sigma(\mathbf{w}\cdot\mathbf{x}^{\mathbf{i}}+b) - y^{(i)}]x_j^{(i)}$$

$$\begin{aligned}\frac{\partial Cost}{\partial \mathbf{w}} &= \frac{1}{m}(\hat{\mathbf{y}}-\mathbf{y})^T\mathbf{X} \\ &= \frac{1}{m}(\sigma(\mathbf{X}\mathbf{w}+\mathbf{b})-\mathbf{y})^{\mathbf{T}}\mathbf{X}\end{aligned}$$

## 7.Regularization

1.权重over-fitting the training data,提升泛化能力

2.为了避免过拟合在目标函数中加入含有参数的项式$R(\theta)$又称为惩罚系数

$$\hat{\theta} = \underset{\theta}{argmax} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta)$$

L2 **regularization**

$$R(\theta) = ||\theta||_2^2 = \sum \theta^2$$

L1 **regularization**

$$R(\theta) = ||\theta||_i$$

惩罚力度

**对于大权重L2>L1**

**对L1来说更适合处理稀疏的矩阵，稀疏的矩阵也意味着舍弃了很多输入的特征**

**对L2来说权重的分布要靠近高斯分布，处于均值的较多，较大的权重出现概率小**

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} P(y^{(i)}|x^{(i)}) \times \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2}\right)$$

## 8.learning mutinomial logistic regression

$$extending\ to\ muti-class:$$
$$L_{CE}(\hat{y}, y) = -\sum_{k=1}^{K} \mathbf{y_k} \log \hat{y}_k$$
$$= -\log \hat{\mathbf{y}}_\mathbf{c}\ \text{c is correct class}$$

## 9.Interpreting Models

## 10.Deriving the gradient Equation