教材《机器学习与大语言模型》配套电子教案

机器学习

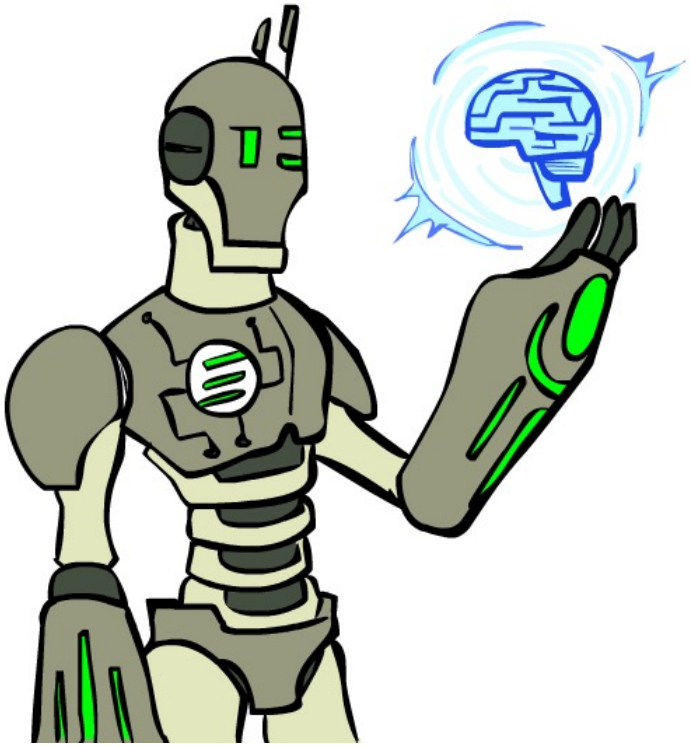
第六讲: 逻辑回归

顾小东
上海交通大学计算机学院

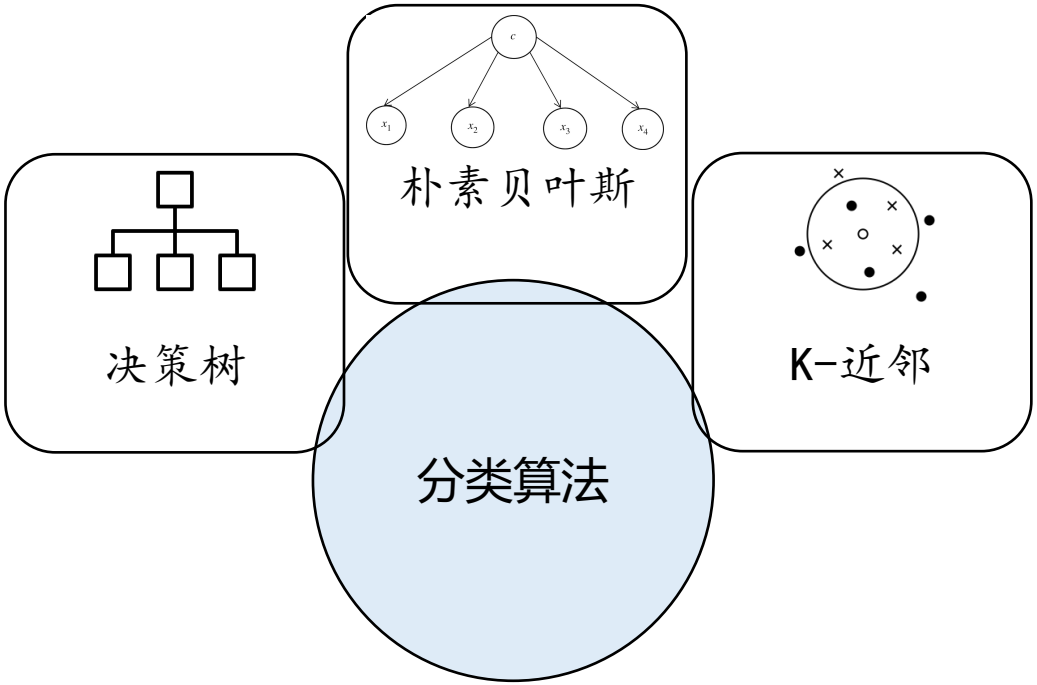
本讲纲要

线性分类器

- (线性)判别函数
- 感知机
- Logistic回归
- Softmax回归

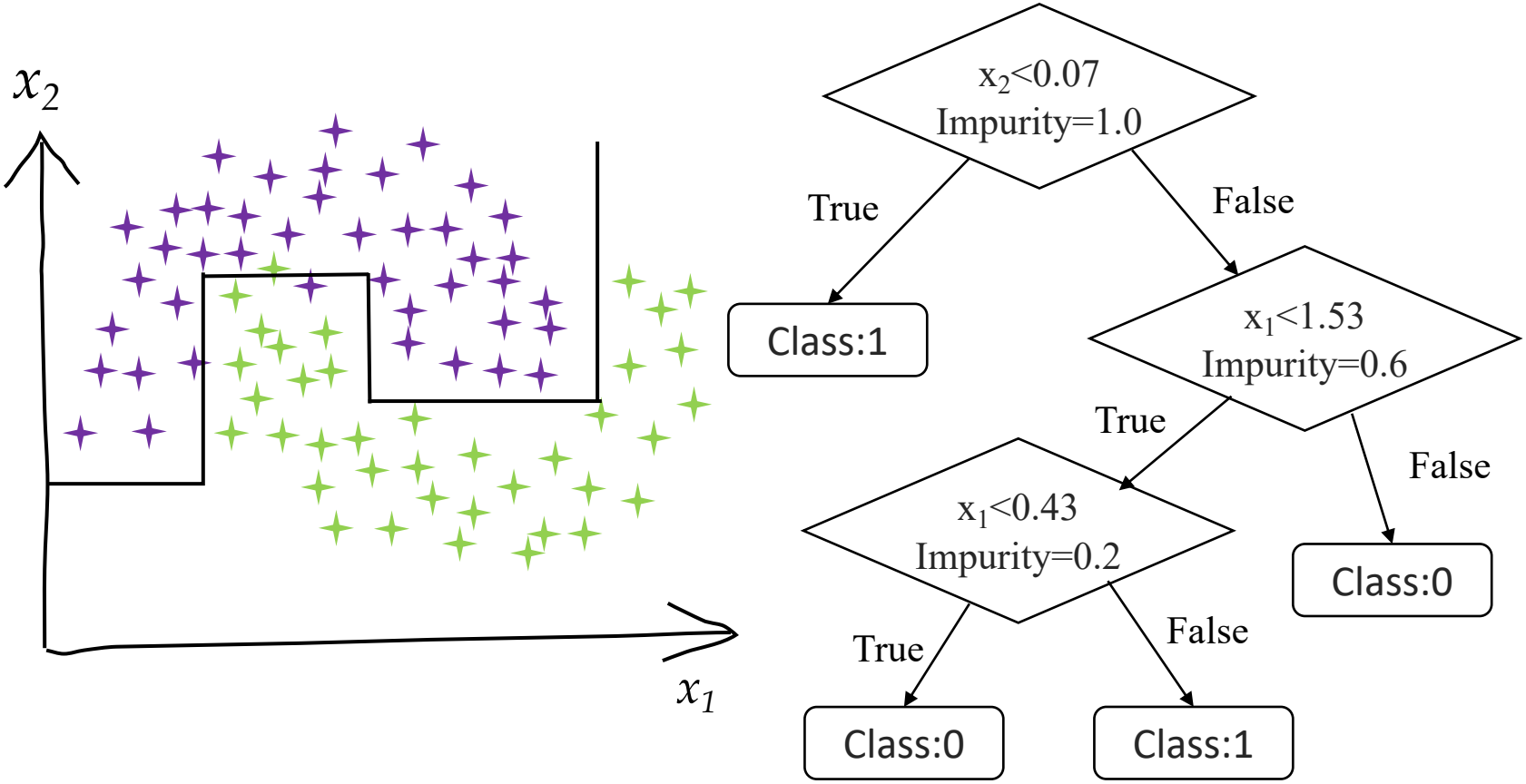


分类器家族



回顾: 决策树

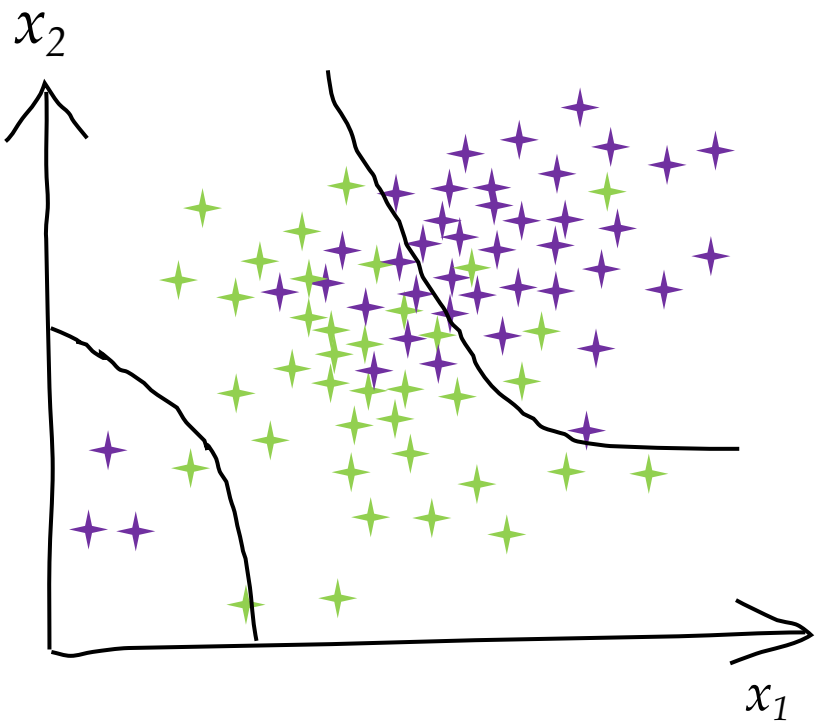
根据数据的不纯净度（Impurity）作分类决策

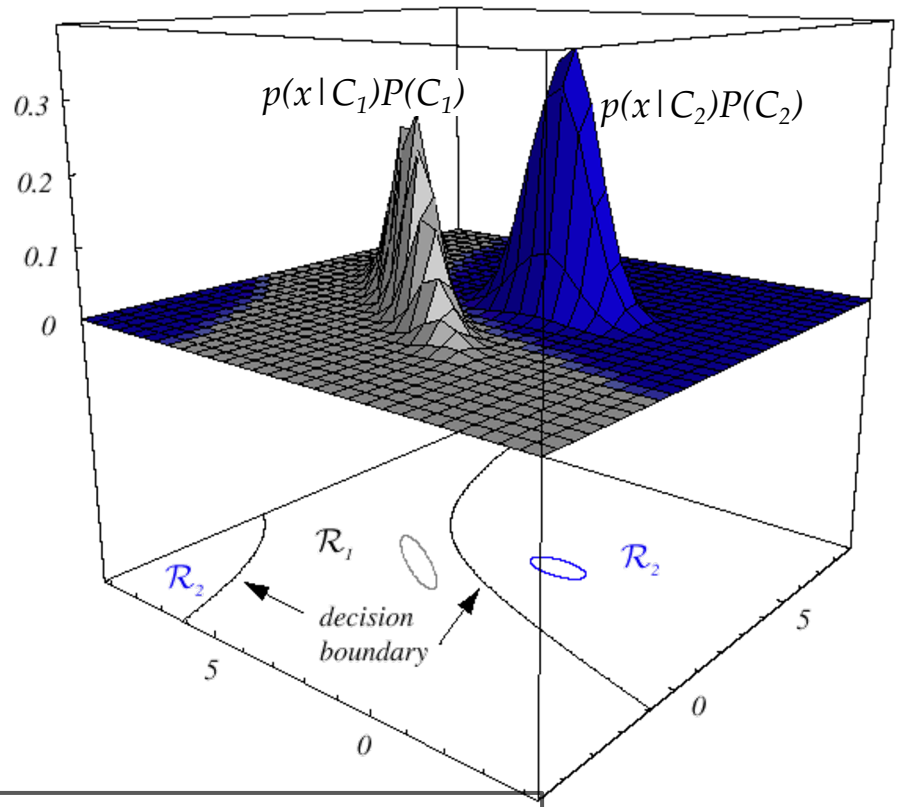


回顾: 贝叶斯分类

根据数据的概率密度作分类决策

$$P(C_i|x)=\frac{p(x|C_i)P(C_i)}{p(x)},\ \ i=1,\ 2$$

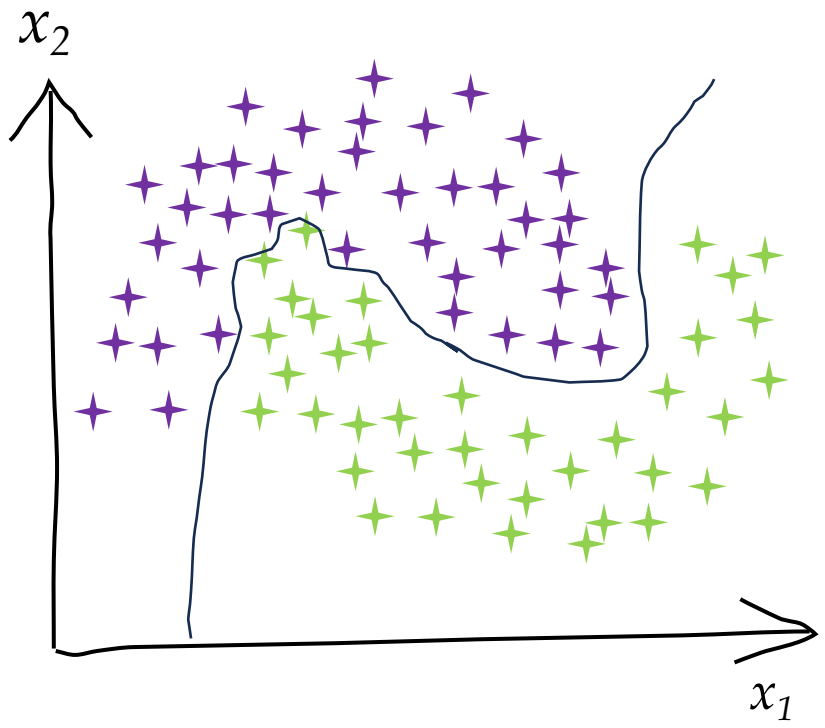




决策规则: Classify x into C_i if $p(C_i|x) > p(C_j|x)$ else C_j.

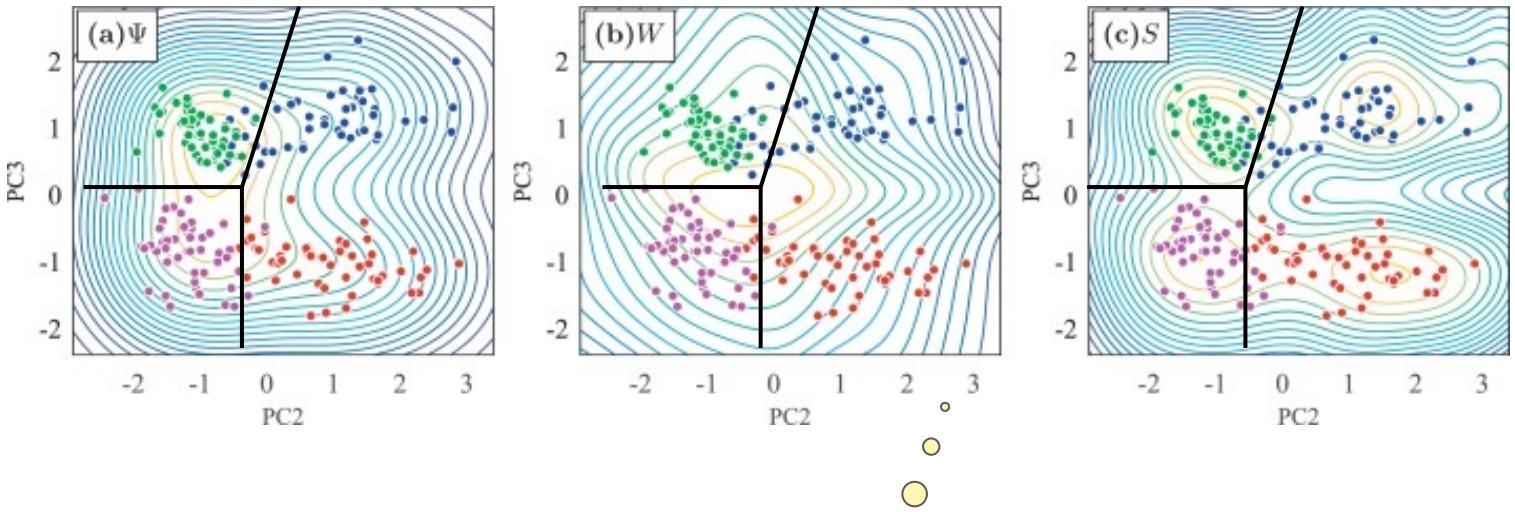
回顾: K-近邻

根据数据点的距离作分类决策



Decision Boundaries Are All You Need

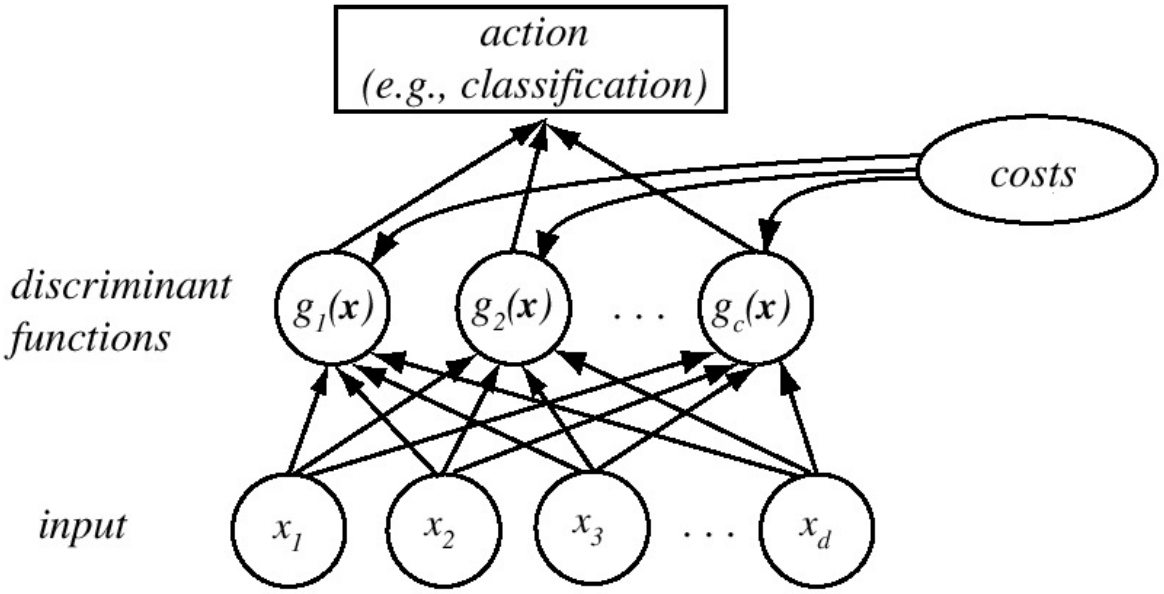
Do we really need to know the structure (impurity/density /distance/…) of data ?



Why not estimate the decision boundary directly?

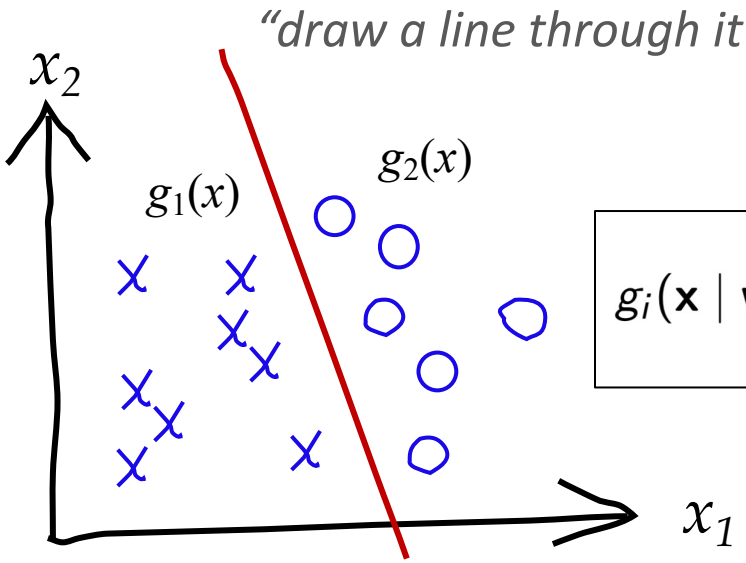
Classification as Discriminants

- consider a classifier with c classes
- define c discriminant functions $g_i(x)$
- **decision rule**: assign x to C_j if $g_j(x) > g_i(x)$, $\forall i \neq j$



线性判别函数

- 采用一个线性函数拟合数据的类别归属。



$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

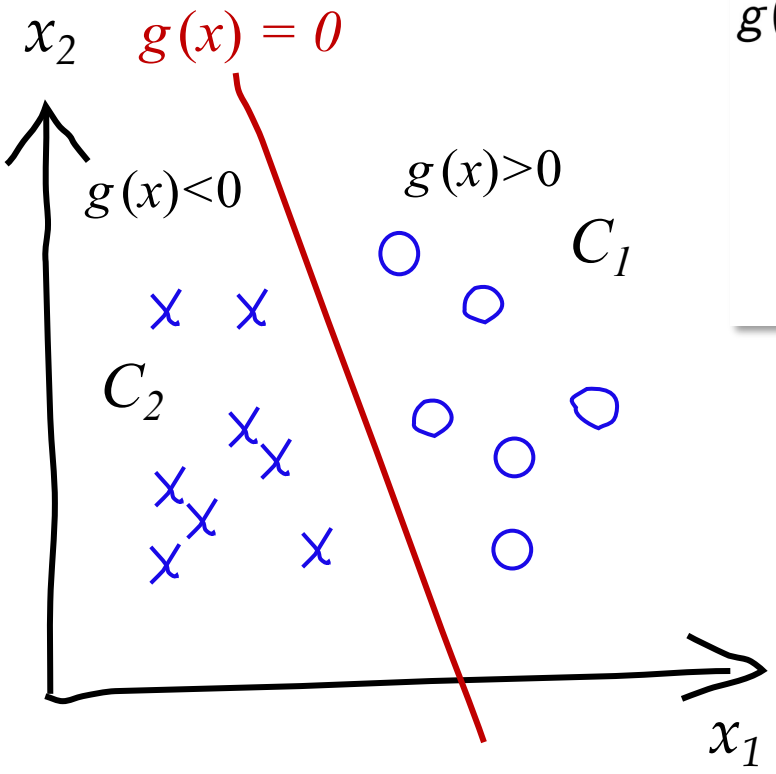
Decide C_1 if $g_1(x) > g_2(x)$ else C_2.

Advantages:

- simplicity: $O(d)$ time and space complexity;
- understandability: final output is a weighted sum of attributes;
- accuracy: quite accurate if some assumptions are satisfied.

Two Classes

- A linear discriminant function models a **linear decision boundary** of two classes.



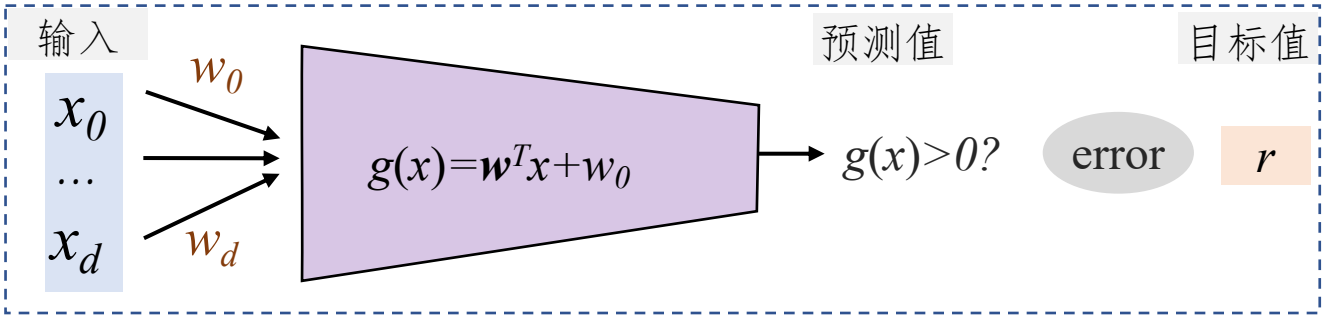
$$
\begin{aligned}
g(\mathbf{x}) &= g_1(\mathbf{x}) - g_2(\mathbf{x}) \\
&= (\mathbf{w}_1^T\mathbf{x} + w_{10}) - (\mathbf{w}_2^T\mathbf{x} + w_{20}) \\
&= (\mathbf{w}_1 - \mathbf{w}_2)^T\mathbf{x} + (w_{10} - w_{20}) \\
&= \mathbf{w}^T\mathbf{x} + w_0
\end{aligned}
$$

- Decision rule:

$$
\text{Choose} \begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}
$$

Perceptron (感知机)

- The first, naïve linear classifier. (to introduce in future lectures)

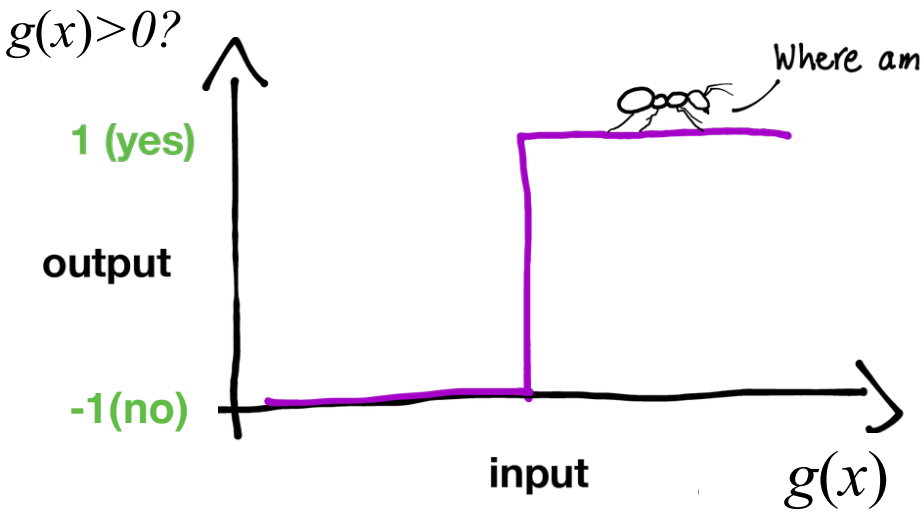


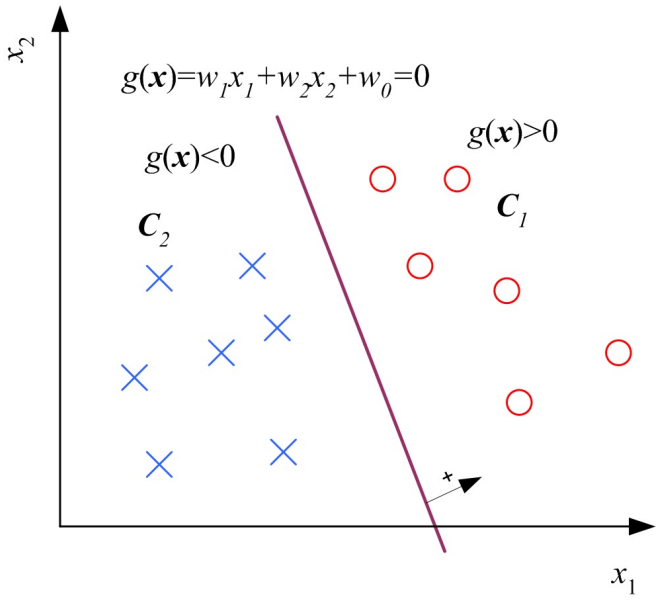
- 训练目标: 估计线性函数的参数(系数) $\boldsymbol{w}$ 和 w_0
- 预测新样本: 对于新数据样本x, 计算 $g(x) = \boldsymbol{w}^T x+w_0$ 按照g(x)的正负号判定类别： $g(x) > 0$代表C_1类，否则属于 C_2 类.

perceptron classifies data based on which **side** of the plane the new point lies on.

感知机的局限

硬决策不利数值计算和优化





I know it belongs to C_1 but to what extent?
(0-1 decision is too hard to optimize)

Linear Classification with Uncertainty

- What is the posterior probability of choosing C1 (or C2)?

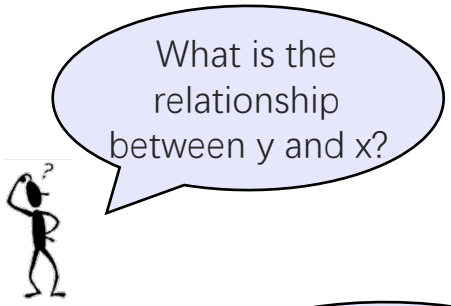
Let

model output

$$P(C_1 \mid \boldsymbol{x}) = y \qquad P(C_2 \mid \boldsymbol{x}) = 1 - y$$

Classification rule:

$$\text{Choose} \begin{cases} C_1 & \text{if } y > 0.5 \\ C_2 & \text{otherwise} \end{cases}$$

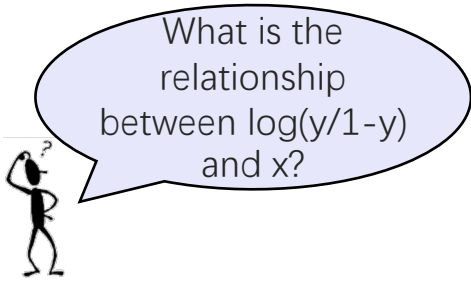


Equivalent Rule:

$$\frac{y}{1-y} > 1 \qquad \text{or} \qquad \log \frac{y}{1-y} > 0$$

odds of y

Log odds (logit function) of y

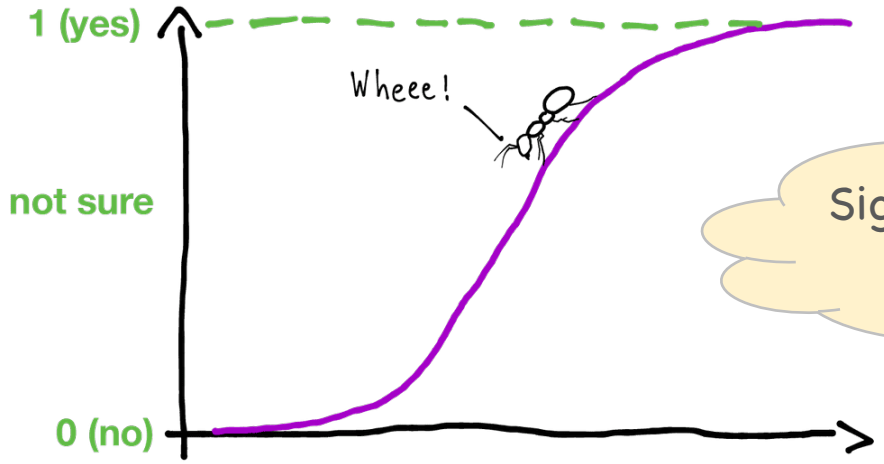


The Sigmoid Function

$$\log \frac{y}{1-y} = \mathbf{w}^{\mathrm{T}} x + w_0$$

$$\Rightarrow \qquad y = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0) = \frac{1}{1 + \exp[-(\mathbf{w}^T \mathbf{x} + w_0)]}$$

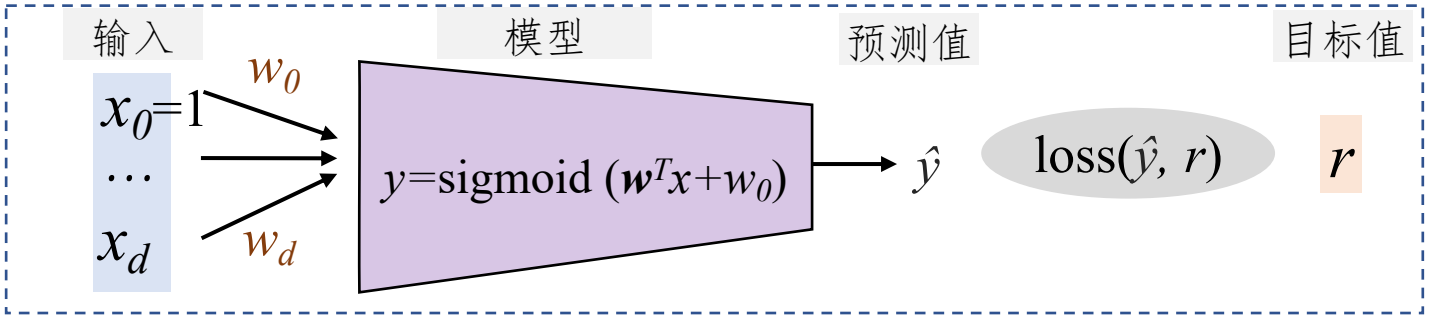
- sigmoid函数, 也称为逻辑函数(logistic function) , 是logit的反函数,它根据输入数据直接计算其属于第1类的后验$P(C_1|x)$:



Sigmoid平滑，利于优化算法作数值优化

逻辑回归模型(Logistic Regression)

- **一种使用Sigmoid函数刻画分类决策边界的分类器:**



- 训练目标: 估计线性函数的参数(系数) $\boldsymbol{w}$ 和 w_0
- 预测新样本: 对于新数据样本x, 计算 $y = \text{sigmoid}\ (\boldsymbol{w}^T x+w_0)$ ： $y > 0.5$代表C_1类，否则属于 C_2 类。

损失函数

$y \equiv p(C_1|x) = \text{sigmoid}(w^T x + w_0)$ is the estimated probability of $x \in C_1$.

- **对于任意输入**x, 模型预测x属于第1类的概率y。**令** $r \in \{0, 1\}$ **为x的真实类别标签** ($r = 1: x \in C_1$, $r = 0: x \in C_2$):
  - 如果 $r = 1$: 我们希望$\log p(C_1|x) = \log y$**尽可能大**, 意味着损失**为**$-\log y$
  - 如果 $r = 0$: 我们希望$\log p(C_2|x) = \log (1-y)$尽可能大, 意味着损失为$-\log (1-y)$
- 合并以上两个条件，得到一般**形式**的交叉熵损失函数:

$$\ell(\boldsymbol{w}, w_0 | x, r) = \underbrace{-r \log y}_{\text{nonzero only if r=1}} \underbrace{-(1-r) \log (1-y)}_{\text{nonzero only if r=0}}$$

模型训练

输入: 数据集 $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(N)}, r^{(N)})\}$

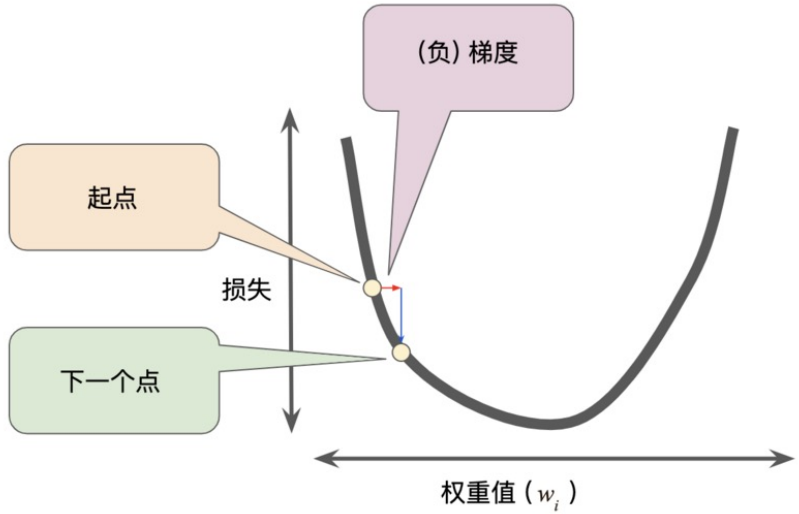
采用梯度下降法最小化交叉熵损失函数:

- 目标:

$$\min_w L(\boldsymbol{w})$$

- 迭代:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \frac{\partial L}{\partial w}$$



如何计算 $\frac{\partial L}{\partial w}$?

优化算法 – 梯度下降法

$$\ell(\boldsymbol{w}, w_0 \mid x, r) = -\, r \log y - (1-r) \log (1-y)$$

$$L(\boldsymbol{w}, w_0 \mid D) = -\sum_{\ell=1}^{N} r^{(\ell)} \log y^{(\ell)} + (1-r^{(\ell)}) \log (1-y^{(\ell)})$$

如何计算$\frac{\partial L}{\partial w}$？

Hint: if y=sigmoid(a) = 1/[1+exp(-a)], its derivative is $\frac{dy}{da}$ = y(1-y)

For each $\boldsymbol{w}_j$ (j =1,…,d):

$$\frac{\partial L}{\partial w_j} = -\sum_{\ell}\left(\underbrace{\frac{\partial \mathrm{L}}{\partial y^{(\ell)}}\frac{\partial y^{(\ell)}}{\partial a^{(\ell)}}\frac{\partial a^{(\ell)}}{\partial w_j}}_{\text{Chain rule}}\right) = -\sum_{l}\left(\frac{r^{(\ell)}}{y^{(\ell)}} - \frac{1-r^{(\ell)}}{1-y^{(\ell)}}\right)\frac{\partial y^{(\ell)}}{\partial a^{(\ell)}}\frac{\partial a^{(\ell)}}{\partial w_j}$$

优化算法 – 梯度下降法 (2)

For each $\boldsymbol{w}_j$ (j =1,…,d):

$$\frac{\partial L}{\partial w_j} = -\sum_{\ell}\left(\frac{r^{(\ell)}}{y^{(\ell)}} - \frac{1-r^{(\ell)}}{1-y^{(\ell)}}\right)\frac{\partial y^{(\ell)}}{\partial a^{(\ell)}}\frac{\partial a^{(\ell)}}{\partial w_j}$$

Since $a^{(l)} = \boldsymbol{w}^{\mathrm{T}}x^{(l)} + w_0$, we have $\frac{\partial a^{(\ell)}}{\partial w_j} = x_j^{(\ell)}$

So,

$$\frac{\partial L}{\partial w_j} = -\sum_{\ell}\left(\frac{r^{(\ell)}}{y^{(\ell)}} - \frac{1-r^{(\ell)}}{1-y^{(\ell)}}\right)y^{(\ell)}\left(1-y^{(\ell)}\right)x_j^{(\ell)}$$

$$= -\sum_{\ell}(r^{(\ell)} - y^{(\ell)})x_j^{(\ell)}$$

逻辑回归算法总结

Gradient Descend for LR

Input: $D = \{(\mathrm{x}^{(l)}, \mathrm{r}^{(l)})\}$ $(l = 1{:}N)$
for $j = 0,\ldots,d$
    $\boldsymbol{w}_j \leftarrow rand\ (-0.01, 0.01)$
repeat
    for $j = 0,\ldots,d$
        $\Delta \boldsymbol{w}_j \leftarrow 0$
    for $l = 1,\ldots,N$
        $a \leftarrow 0$
        for $j = 0, \ldots, d$
            $a \leftarrow a + \boldsymbol{w}_j x_j^{(l)}$
        $y \leftarrow \text{sigmoid}\ (a)$
        $\Delta \boldsymbol{w}_j \leftarrow \Delta \boldsymbol{w}_j + (r^{(l)} - y) x_j^{(l)}$
    for $j = 0,\ldots,d$
        $w_j \leftarrow w_j + \eta \Delta w_j$
until convergence

推广到多个类别

- 如果不止2个类别该如何分类?

Binary Classification

- Spam
- Not spam

Multiclass Classification

- Dog
- Cat
- Horse
- Fish

目标是一个二值集合 {0, 1}.

目标是K个类的离散集合 {1, . . . ,K}.

多分类问题的预测概率 *

输入数据x属于第 i 类(i=1,...,K)的概率是多少?

$$y_i = \hat{P}(C_i \mid \mathbf{x}) = \frac{\exp\left(\mathbf{w}_i^T \mathbf{x} + w_{i0}\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w}_j^T \mathbf{x} + w_{j0}\right)}, \quad i = 1, \ldots, K$$

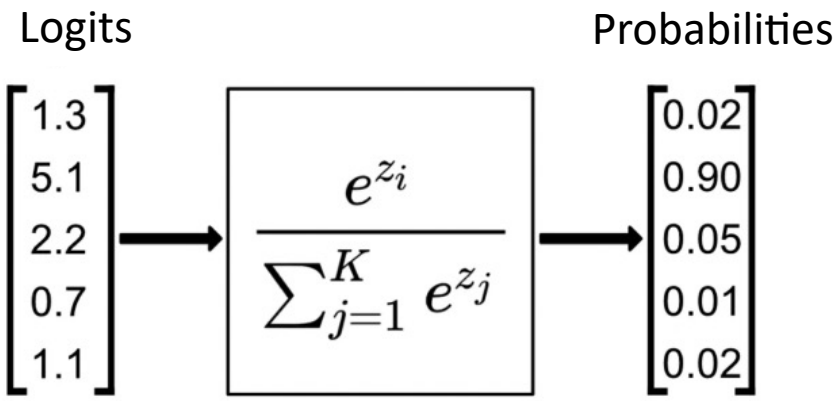
Softmax函数

- **softmax 函数定义为：**

$$y_i = \text{softmax}(a_1, \ldots, a_K)_i = \frac{e^{a_i}}{\sum_{j=1}^{K} e^{a_j}}$$

其中输入 $a_i = \boldsymbol{W}_i x + w_{i0}$ 称为**logits**. $\boldsymbol{W}_i$ 和 w_{i0} 是可训练参数。

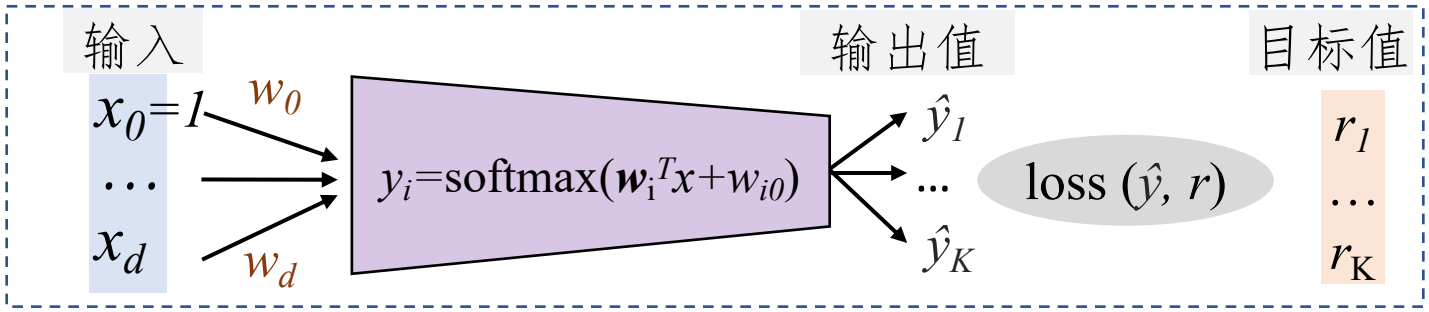
- Softmax 将一组logits (**分数**) 转换为概率分布。



Softmax回归

- 一种**使用softmax函数刻画分类决策边界的分类器**:

$$y_i = \text{softmax}\left(\boldsymbol{w}_i^{\mathrm{T}} x + w_{i0}\right) \qquad i = 1,..,K$$



- 训练目标: 评估线性函数的参数(系数) $\boldsymbol{w}$ 和 w_{i0} (i=1:K)
- 预测新样本: 对于新数据样本x, 计算 $y_i = \text{softmax}\left(\boldsymbol{w}_i^{\mathrm{T}} x + w_0\right)$: 如果$y_i = \max\{y_{1:K}\}$则代表数据属于第i类(x$\in C_i$).

损失函数

- **对于给定的输入x, 模型输出向量$y = (y_1, \ldots y_K)$**代表各个类的概率, 我们用**一个独热向量 $r = (r_1, \ldots, r_K)$表示数据的真实类别标签**(r_i=1:$x \in C_i$, r_i=0: $x \notin C_i$)
  - if $r_1 = 1$: we aim to maximize $\log p(C_1 | x) = \log y_1$, cost is $-\log y_1$
  - if $r_2 = 1$: we aim to maximize $\log p(C_2 | x) = \log y_2$, cost is $-\log y_2$
  - ……
- 整合成一般形式的交叉熵损失函数:

$$L_{CE}(\boldsymbol{y}, \boldsymbol{r}) = -\sum_{i=1}^{K} r_i \log y_i = -\boldsymbol{r}^T(\log \boldsymbol{y})$$

其中*log*代表逐元素对数

模型训练

- **输入数据集**: $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(N)}, r^{(N)})\}$
- 使用梯度下降法最小化损失函数:

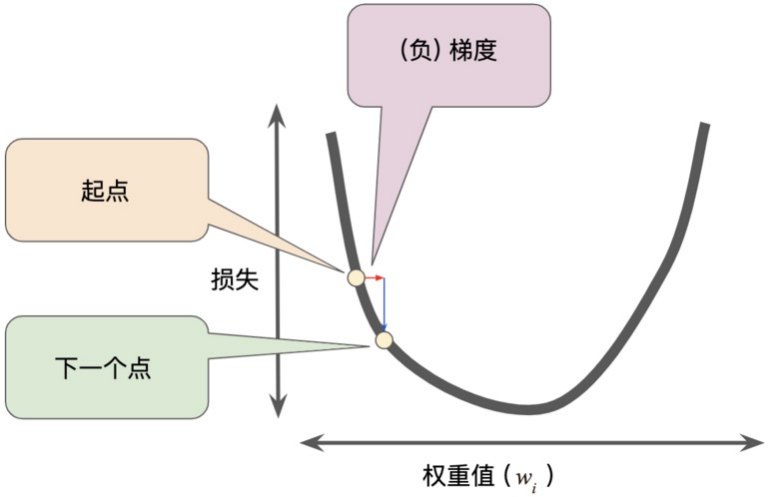
- 目标:

$$\min_w L(\boldsymbol{w})$$

- 迭代:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \frac{\partial L}{\partial w}$$

如何计算 $\frac{\partial L}{\partial w}$?



采用梯度下降法优化

$$L(\boldsymbol{w}|\,\boldsymbol{x},\boldsymbol{r}) = -\sum_{i=1}^{K} r_i \log y_i$$

$$L(\boldsymbol{w}|\,D) = -\sum_{\ell=1}^{N}[\sum_{i=1}^{K} r_i^{(\ell)} \log y_i^{(\ell)}]$$

What is $\frac{\partial L}{\partial w}$?

Hint: if $y_i = \exp(a_i)/\Sigma_j \exp(a_j)$, its derivative is $\frac{\partial y}{\partial a} = y_i(\delta_{ij} - y_j)$ where $\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$

For each w_j and w_{j0} (j=1,…,K), given $\Sigma_i r_i^{(\ell)} = 1$:

$$-\frac{\partial L}{\partial w_j} = \sum_l \sum_i \frac{\partial L}{\partial y_i^{(\ell)}} \frac{\partial y_i^{(\ell)}}{\partial a^{(\ell)}} \frac{\partial a^{(\ell)}}{\partial w_j} = \sum_l \sum_i r_i^{(\ell)} (\delta_{ij} - y_j^{(l)}) \boldsymbol{x}^{(\ell)}$$

$$= \sum_l \left[\sum_i r_i^{(\ell)} \delta_{ij} - y_j^{(\ell)} \sum_i r_i^{(\ell)} \right] x^{(\ell)} = \sum_l (r_j^{(\ell)} - y_j^{(\ell)}) \boldsymbol{x}^{(\ell)}$$

Softmax回归算法总结

Gradient Descend for Softmax Regression

Input: $D = \{(x^{(l)}, r^{(l)})\}$ $(l = 1{:}N)$
for $i = 1, \ldots, K$, **for** $j = 0,\ldots,d$,
  $w_{ij} \leftarrow$ rand (-0.01, 0.01) // initialization
repeat
  for $i = 1,\ldots,K$, **for** $j = 0,\ldots, d$, $\Delta w_{ij} \leftarrow 0$
  for $l = 1,\ldots,N$
    for $i = 1,\ldots,K$
      $a_i \leftarrow 0$
      for $j = 0,\ldots, d$
        $a_i \leftarrow a_i + w_{ij} x_j^{(l)}$
    for i $= 1,\ldots, K$
      $y_i \leftarrow \exp(a_i)/\boldsymbol{\Sigma}_j \exp(a_j)$
    for $i = 1,\ldots,K$
      for $j = 0,\ldots, d$
        $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^{(l)} - y_i) x_j^{(l)}$
  for $i = 1,\ldots,K$
    for $j = 0,\ldots, d$
      $w_{ij} \leftarrow w_{ij} + \boldsymbol{\eta} \Delta w_{ij}$
until convergence

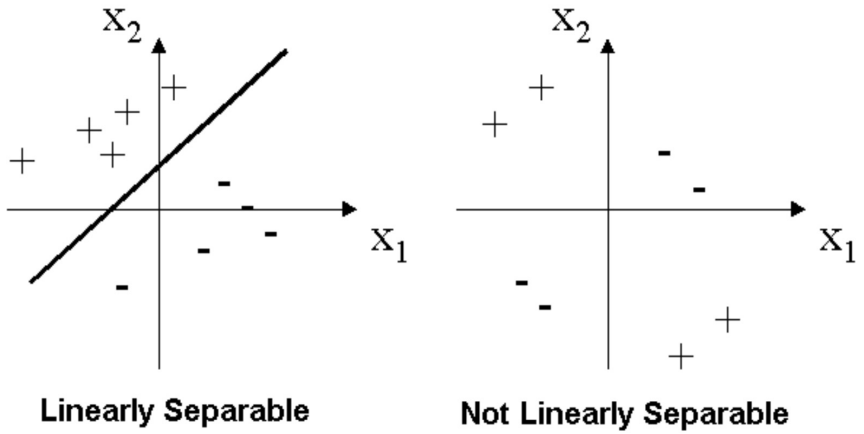
Tutorial: Logistic regression from scratch with Python

https://www.kaggle.com/code/ujjalkumarmaity/logistic-regression-from-scratch

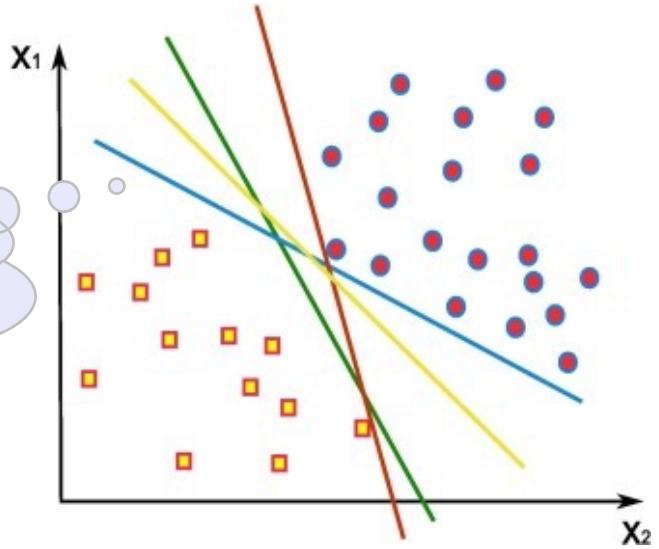
IT'S TIME FOR *coding*

Limitations of Linear Classifiers

Requires data to be linearly separable



Even if the data is linearly separable, there can be many separations.



What's Next?

Support Vector Machine

Find a decision boundary that maximizes the margin between two classes.