Machine Learning I

3. Linear Classification

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

Contents

感知机

回归用于分类

最简单的想法：根据**取值范围**分类

- 例如$[0, 10)$的坐标区间：$[0, 1)$代表第一个类

回归用于分类

最简单的想法：根据**取值范围**分类

- 例如$[0, 10)$的坐标区间：$[0, 1)$代表第一个类

问题：不容易度量

- 目标：同类尽量聚在一起；**异类尽量分散开**
  - 但线性映射：边界附近很难区分

回归用于分类

最简单的想法：根据**取值范围**分类

- 例如$[0, 10)$的坐标区间：$[0, 1)$代表第一个类

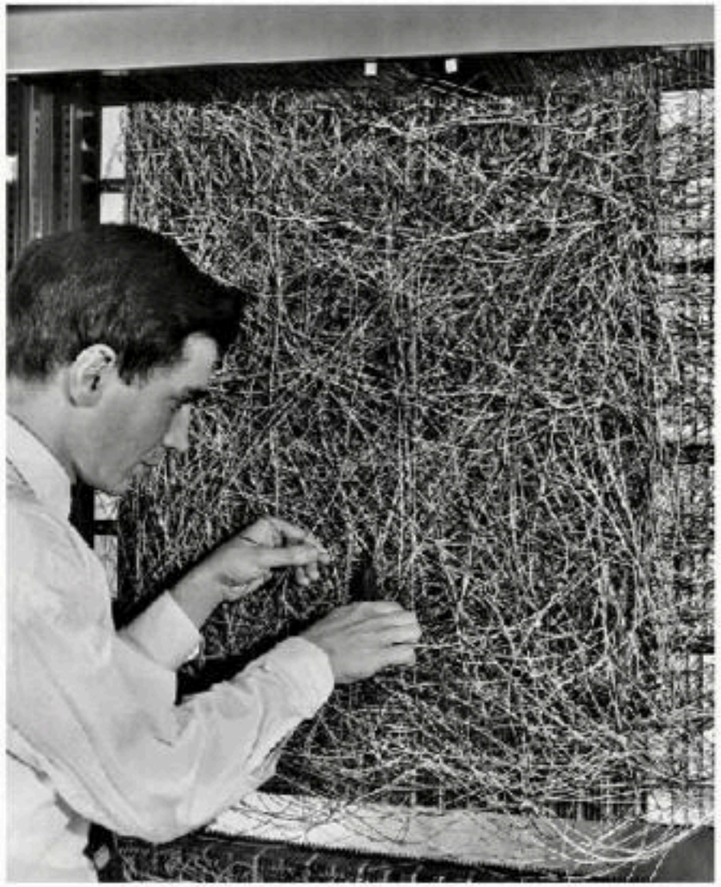
问题：不容易度量

- 目标：同类尽量聚在一起；**异类尽量分散开**
  - 但线性映射：边界附近很难区分
- 解决关键：如何构造目标（损失）函数?
  - 连续值，却需要聚在几个点附近

I型感知机

Frank Rosenblatt 和I型感知机，1960

- **感知机算法**，1957



感知机

- “感”：感受，输入
- “知”：知识，输出
- 对输入进行处理，输出结果的机器

灵感来源 I

生物神经系统：信息反馈通路、多层处理单元

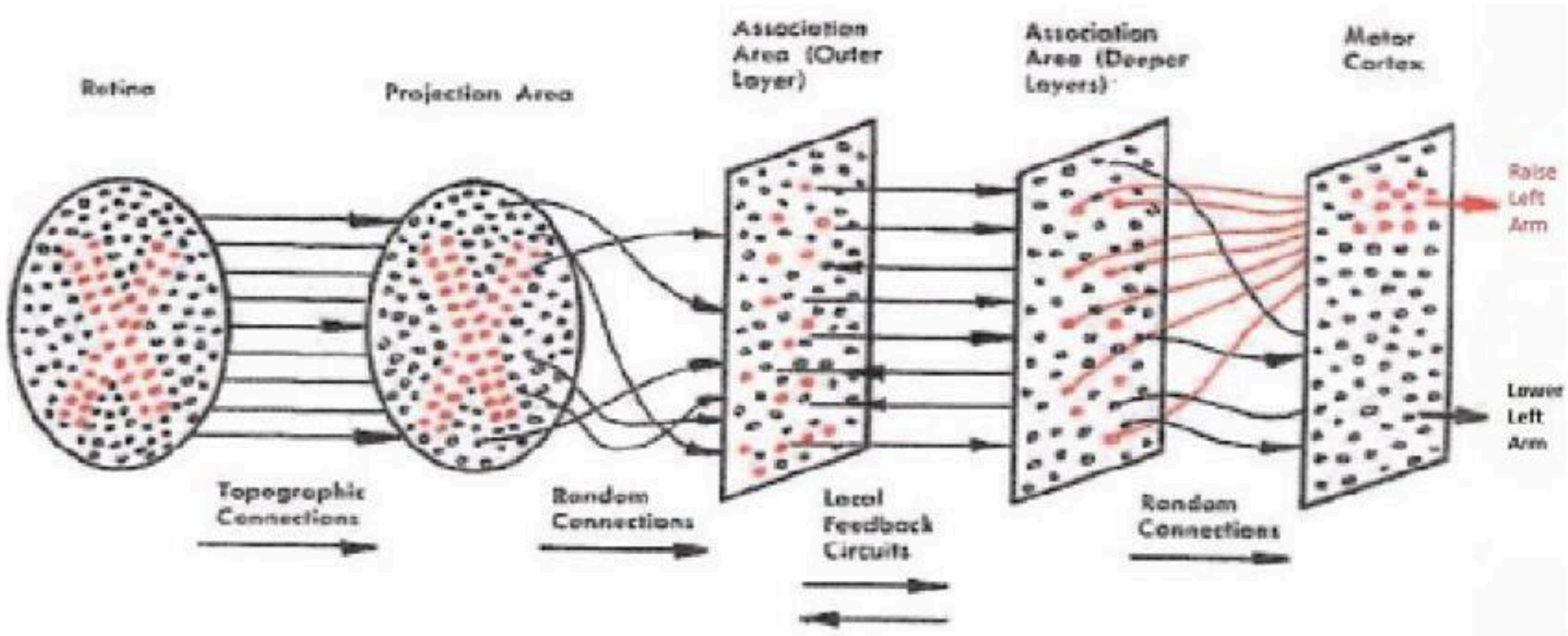


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

灵感来源 II

感知机的结构：单向信息通路、单层处理单元

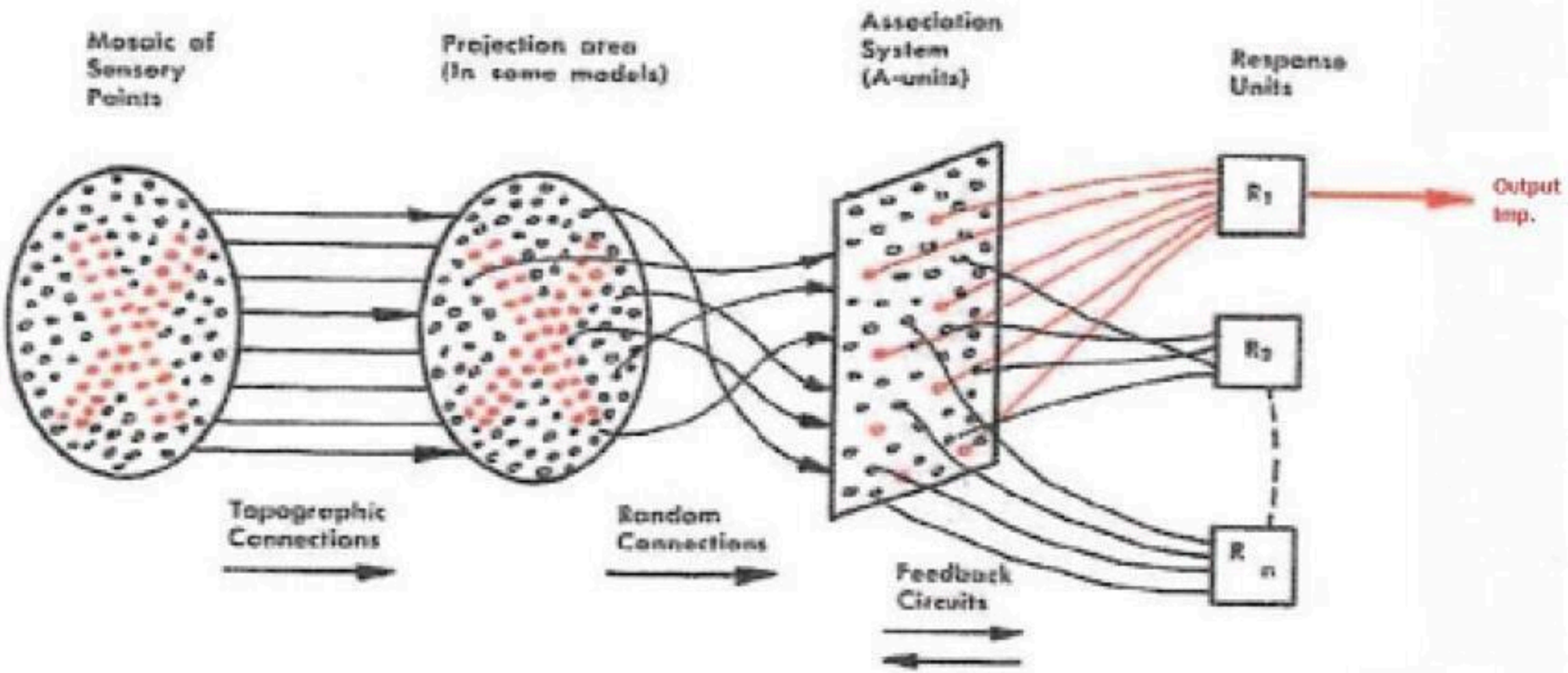


FIG. 2 — Organization of a perceptron.

灵感来源 III

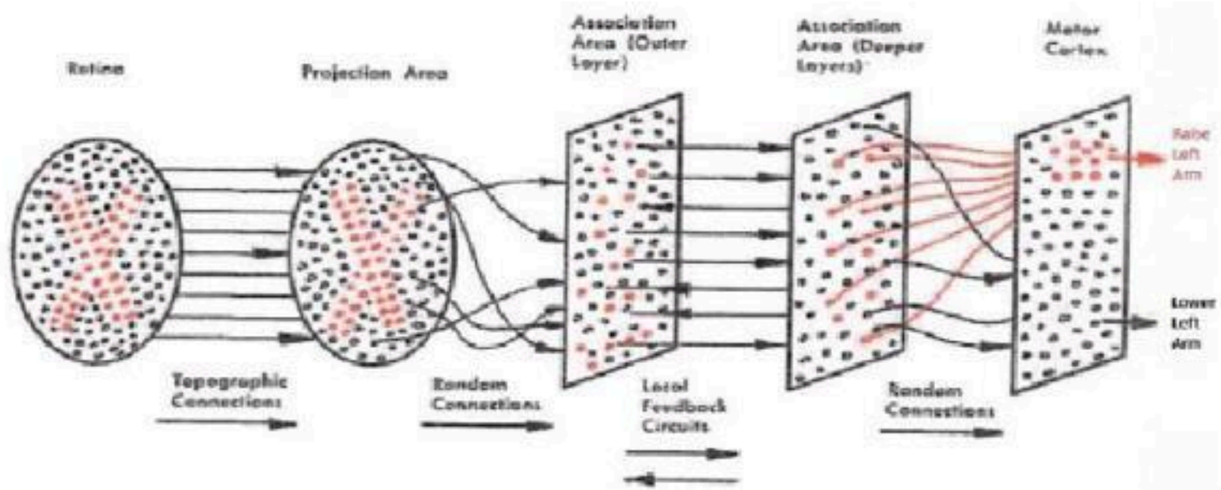


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

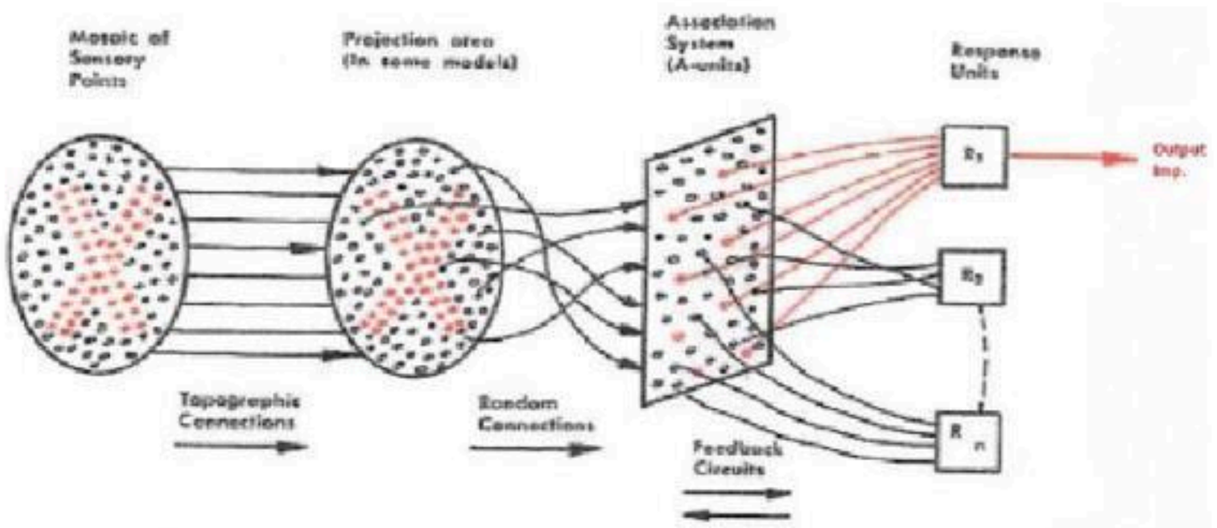


FIG. 2 — Organization of a perceptron.

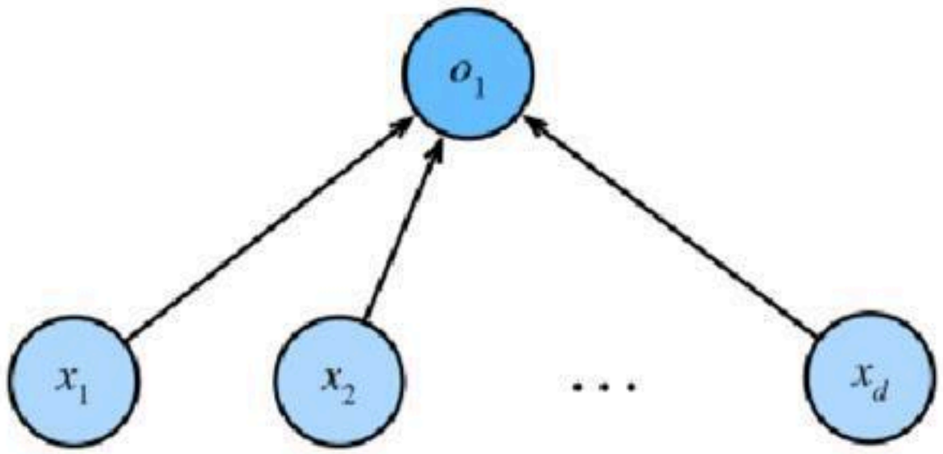
感知机原型

输入：$\mathbf{x}$；参数：$\mathbf{w}, b$；输出：

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right), \sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

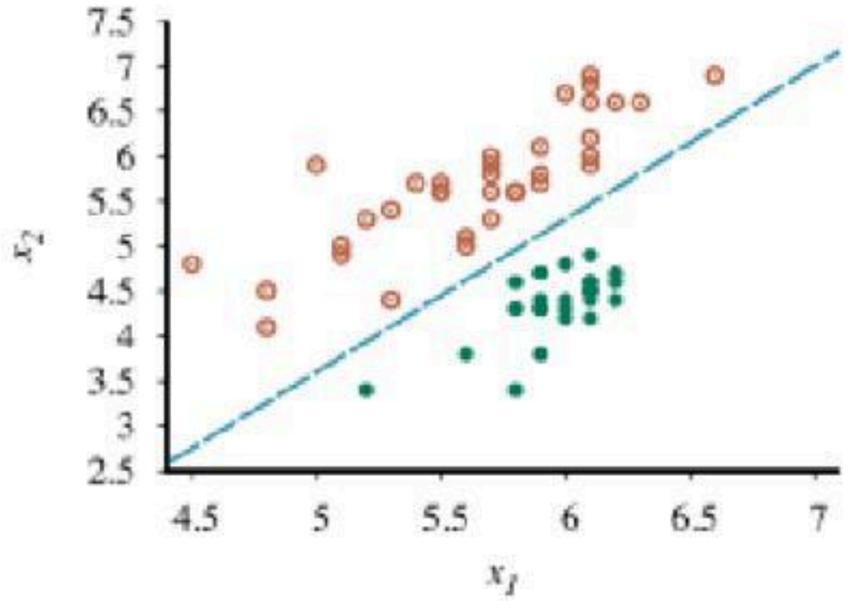
激活函数σ：输出并传递电信号，激活/未激活两种状态

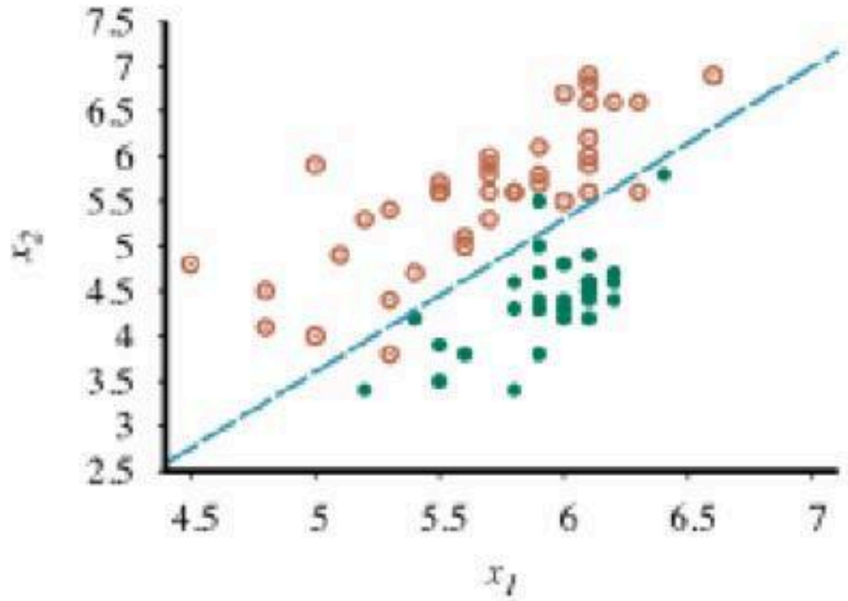
- 回归：实数；感知机：两种离散状态（类别）



地震问题

以下数据收集自地震监测部门

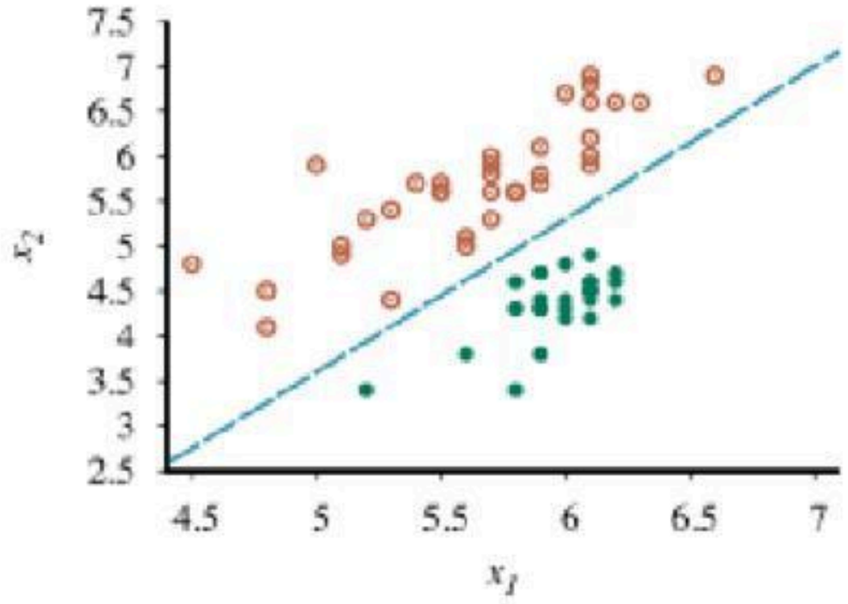


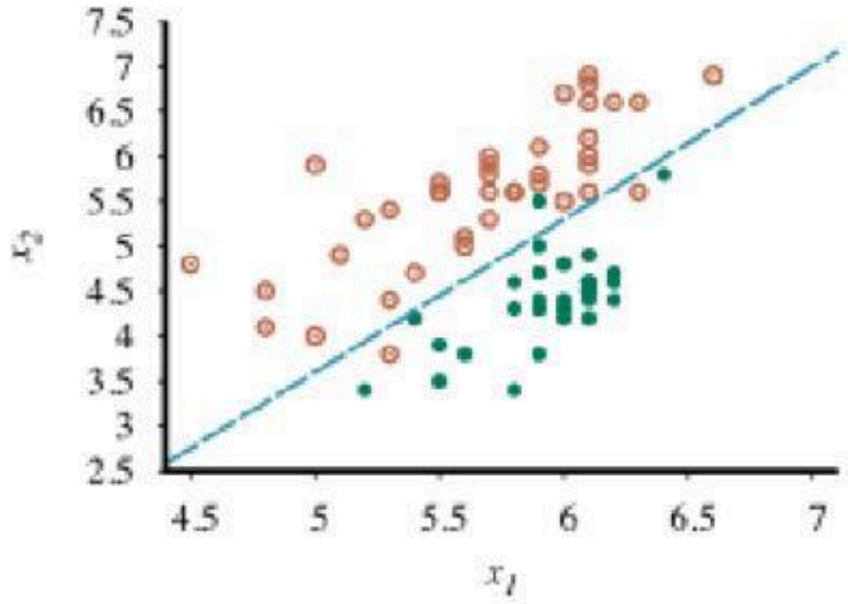


- 红、绿点分别代表两种不同情况（0、1）
- 横、纵坐标代表两种特征描述的取值

地震问题

以下数据收集自地震监测部门





- 红、绿点分别代表两种不同情况（0、1）
- 横、纵坐标代表两种特征描述的取值

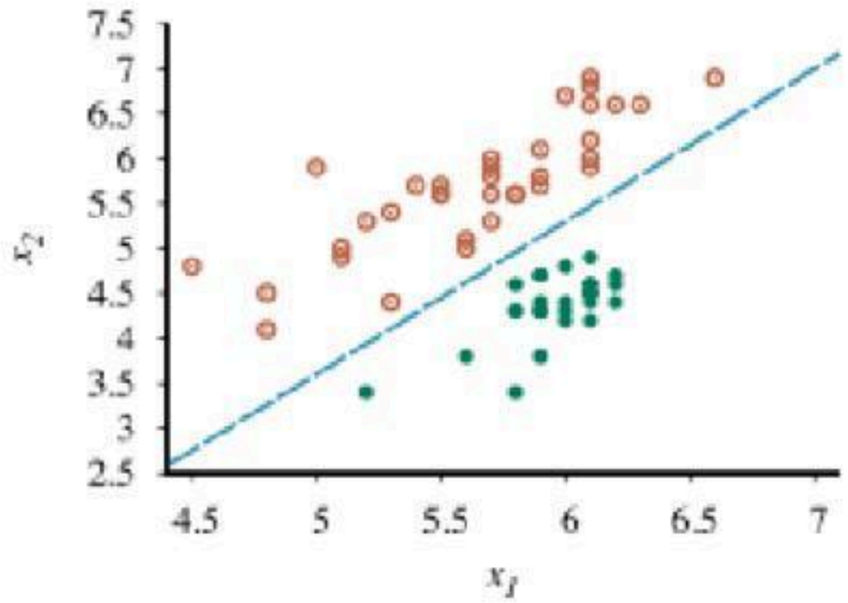
分类问题：将两种情况区分出来（假设边界线右下方规定为类别 1）

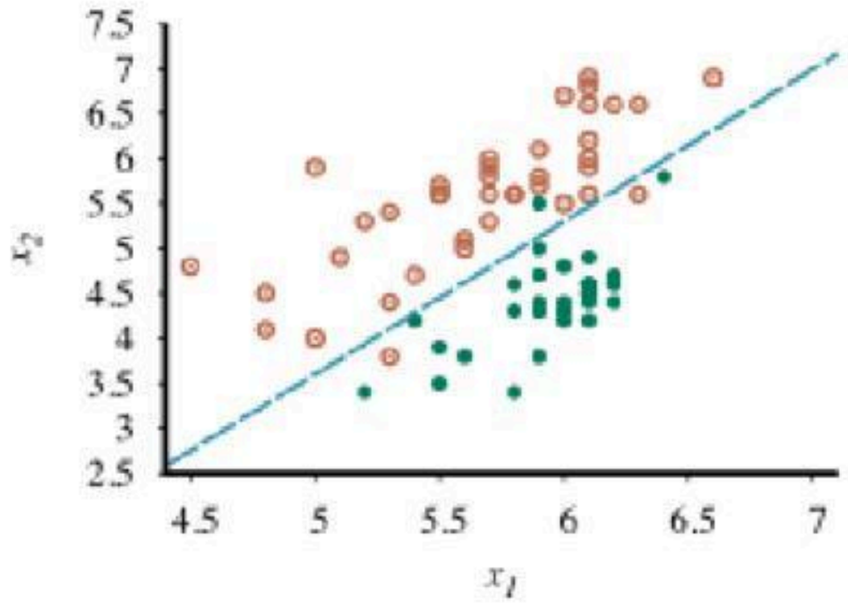
- 最直接的想法就是画一条分界线：人脑不擅长处理非线性运算

决策边界

决策边界将数据分成两个类别，对应分类器的判断依据

- 如果数据是**线性可分**（左图）： 线性模型可以完美区分两个类别

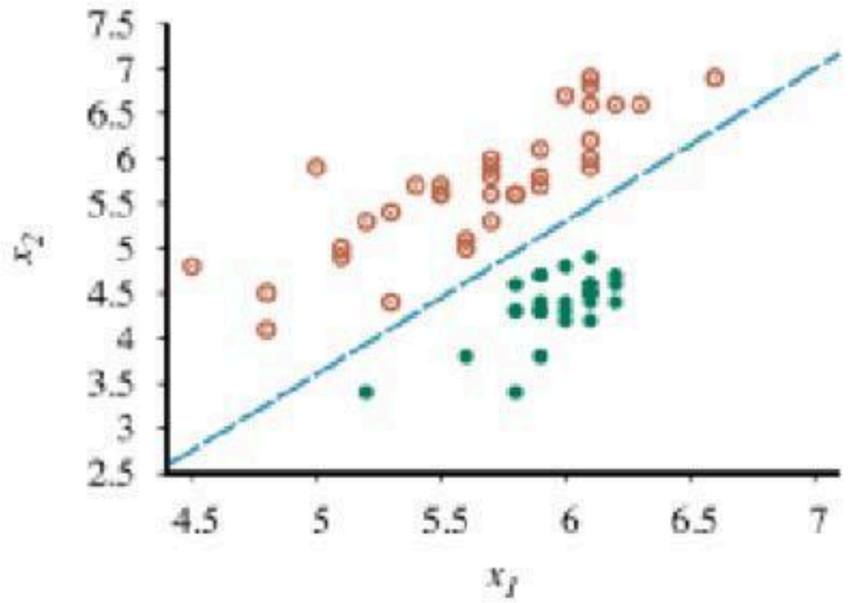


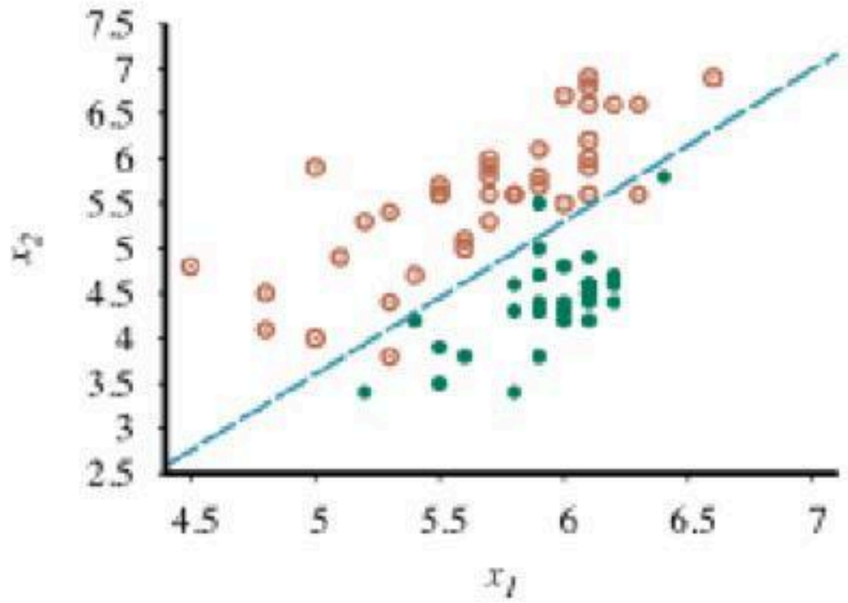


决策边界

决策边界将数据分成两个类别，对应分类器的判断依据

- 如果数据是**线性可分**（左图）： 线性模型可以完美区分两个类别

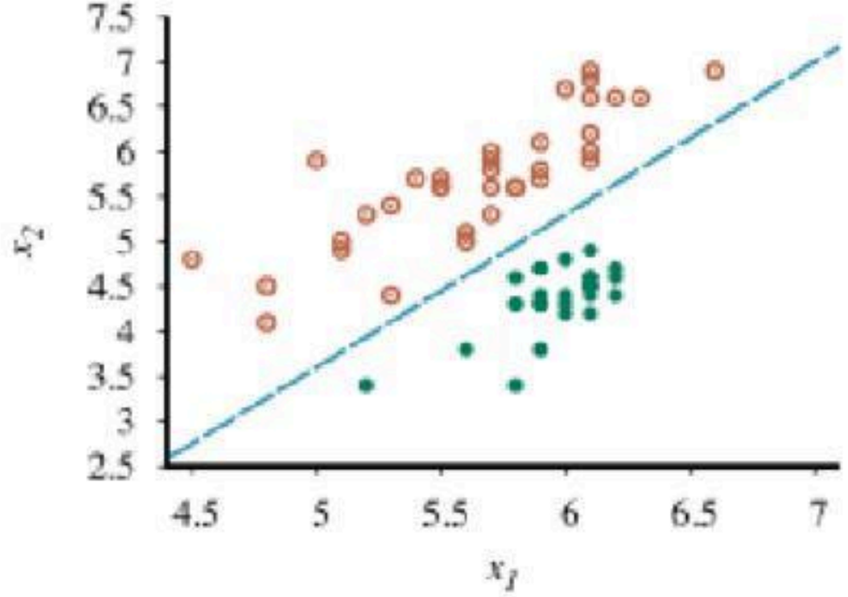


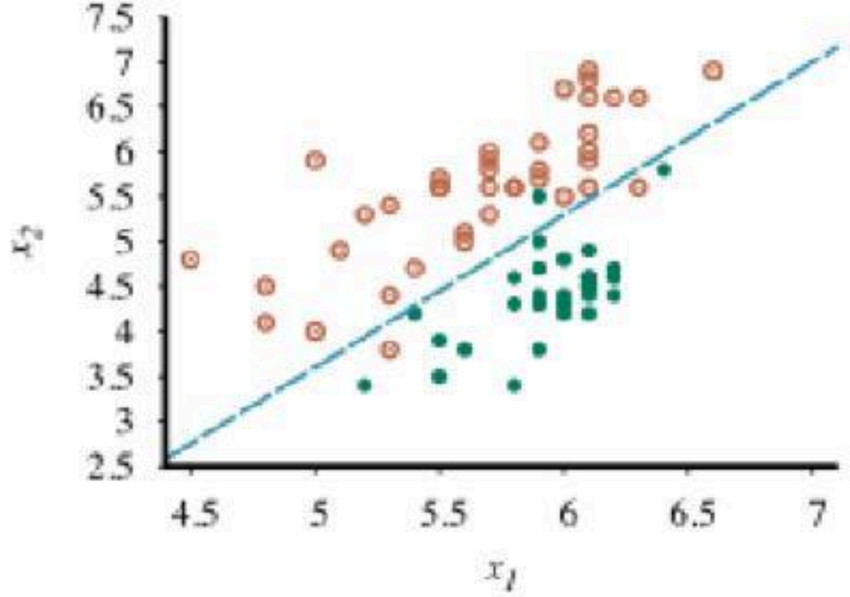


线性边界表达式：$-4.9 + 1.7x_1 - x_2 = 0$。为了后面讨论方便：

- 添加定值变量 $x_0 = 1$：$-4.9x_0 + 1.7x_1 - x_2 = 0$
- 记 $\mathbf{x} = \langle x_0, x_1, x_2 \rangle$, $\mathbf{w} = \langle -4.9, 1.7, -1 \rangle$

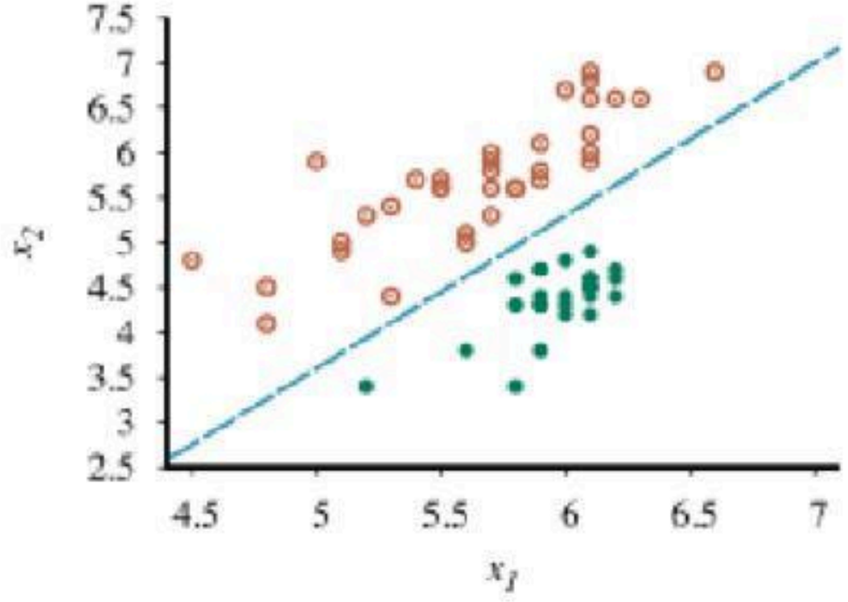
线性边界判别式

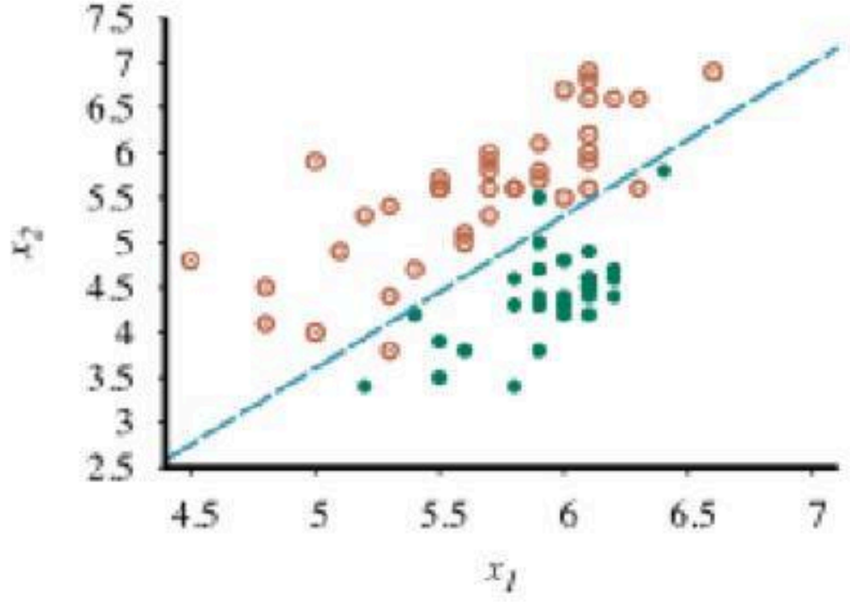




- 边界线右下方规定为类别 1：$-4.9x_0 + 1.7x_1 - x_2 = \mathbf{w} \cdot \mathbf{x} > 0$
  - 因此可以构造**线性分类模型**：$h_{\mathbf{w}}(\mathbf{x}) = 1$, if $\mathbf{w} \cdot \mathbf{x} > 0$

线性边界判别式

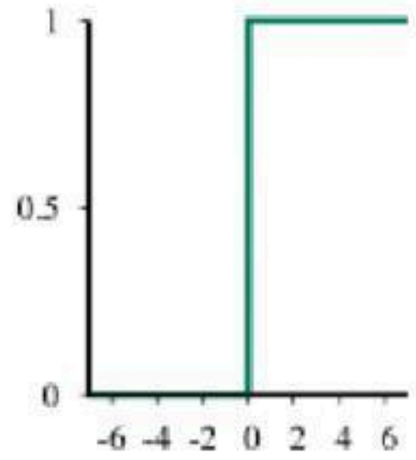




- 边界线右下方规定为类别 1：$-4.9x_0 + 1.7x_1 - x_2 = \mathbf{w} \cdot \mathbf{x} > 0$
  - 因此可以构造**线性分类模型**：$h_{\mathbf{w}}(\mathbf{x}) = 1$, if $\mathbf{w} \cdot \mathbf{x} > 0$

也可理解为 $\mathbf{w} \cdot \mathbf{x}$ 作为**阈值函数**的输入

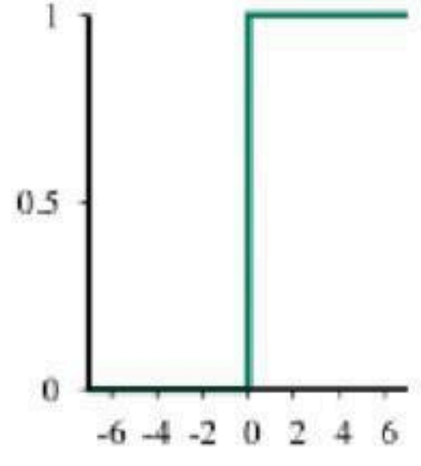
- $h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$
  - $Threshold(z) = 1$ if $z > 0$，即阶梯函数
    - 感知机原型：电信号的激活状态
- 思考：如何优化参数 $\mathbf{w}$？



决策边界：优化问题

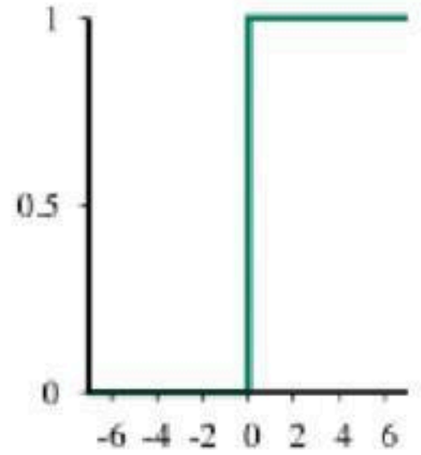
感知机模型 $h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$

- 在数学上是良定义的
- 但梯度几乎处处为0：无法推进梯度下降



决策边界：优化问题

感知机模型 $h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$



- 在数学上是良定义的
- 但梯度几乎处处为0：无法推进梯度下降

- 唯一不为0的点：梯度不存在
  - 无法求解析解

感知机更新规则

定理：对于0/1分类问题，只要数据**线性可分**，以下更新规则**必定收敛**

$$w_i \leftarrow w_i - \eta(\hat{y} - y)x_i$$

- 回顾：此式与线性回归梯度下降法更新规则完全相同；η：学习率
  - 即：感知机模型完全不受梯度不存在的问题影响

感知机更新规则

定理：对于0/1分类问题，只要数据**线性可分**，以下更新规则**必定收敛**

$$w_i \leftarrow w_i - \eta(\hat{y} - y)x_i$$

- 回顾：此式与线性回归梯度下降法更新规则完全相同；η：学习率
  - 即：感知机模型完全不受梯度不存在的问题影响

0/1分类问题分情况讨论（注意：$y, \hat{y}$ 取值只能是 $0, 1$）：

- 如果预测正确（$y = \hat{y}$）：无变化

感知机更新规则

定理：对于0/1分类问题，只要数据**线性可分**，以下更新规则**必定收敛**

$$w_i \leftarrow w_i - \eta(\hat{y} - y)x_i$$

- 回顾：此式与线性回归梯度下降法更新规则完全相同；η：学习率
  - 即：感知机模型完全不受梯度不存在的问题影响

0/1分类问题分情况讨论（注意：$y, \hat{y}$ 取值只能是 $0, 1$）：

- 如果预测正确（$y = \hat{y}$）：无变化
- 如果 $y = 1, \hat{y} = 0$：$x_i > 0$ 时权重增加，反之亦然
  - 此时预测值 $\mathbf{w} \cdot \mathbf{x}$ 小于真实值，故希望预测值增大

感知机更新规则

定理：对于0/1分类问题，只要数据**线性可分**，以下更新规则**必定收敛**

$$w_i \leftarrow w_i - \eta(\hat{y} - y)x_i$$

- 回顾：此式与线性回归梯度下降法更新规则完全相同；η：学习率
  - 即：感知机模型完全不受梯度不存在的问题影响

0/1分类问题分情况讨论（注意：$y, \hat{y}$ 取值只能是 $0, 1$）：

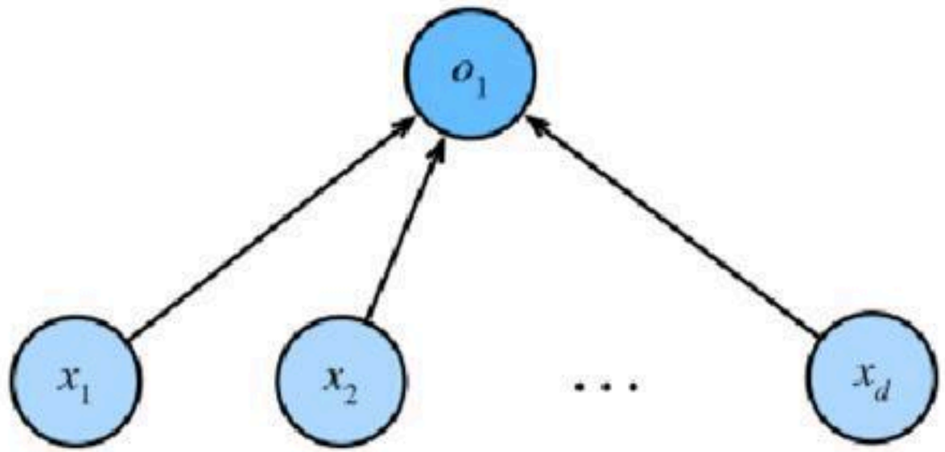
- 如果预测正确（$y = \hat{y}$）：无变化
- 如果 $y = 1, \hat{y} = 0$：$x_i > 0$ 时权重增加，反之亦然
  - 此时预测值 $\mathbf{w} \cdot \mathbf{x}$ 小于真实值，故希望预测值增大
- 思考（同理）：如果 $y = 0, \hat{y} = 1$？

感知机二分类模型

输入：$\mathbf{x}$；参数：$\mathbf{w}, b$；输出：

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right), \sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{otherwise} \end{cases}$$

- 区别：激活函数通常输出：$\{-1, 1\}$

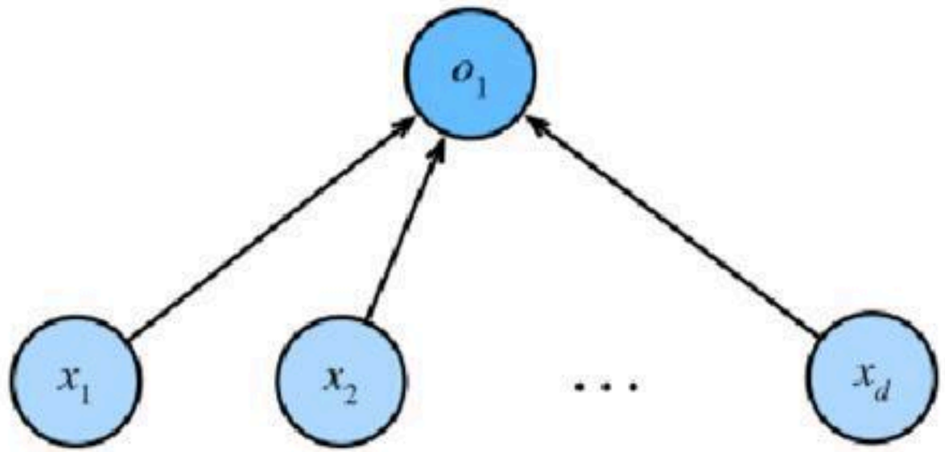


感知机二分类模型

输入：$\mathbf{x}$；参数：$\mathbf{w}, b$；输出：

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right), \sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{otherwise} \end{cases}$$

- 区别：激活函数通常输出：$\{-1, 1\}$



- 注意线性部分不变，**只改变激活函数**
  - 有何意义？改变激活函数，可能找到对特定数据更好的模型解释

训练感知机

训练过程等价于使用**批量大小为1**的梯度下降

1. 初始化： $\mathbf{w} = \mathbf{0}$;
2. 循环，直到所有数据都被分类正确:
    1. **if** $y_i \left(\langle \mathbf{w_i}, \mathbf{x_i} \rangle\right) \leq 0$:
        1. $\mathbf{w_i} = \mathbf{w_i} + y_i \mathbf{x_i}$;

- 二分类输出$\{-1, 1\}$： $\hat{y}_i = \langle \mathbf{w_i}, \mathbf{x_i} \rangle$为正时输出1
    - 所以可以用$y_i \left(\langle \mathbf{w_i}, \mathbf{x_i} \rangle\right) = y_i \hat{y}_i$判断分类是否正确

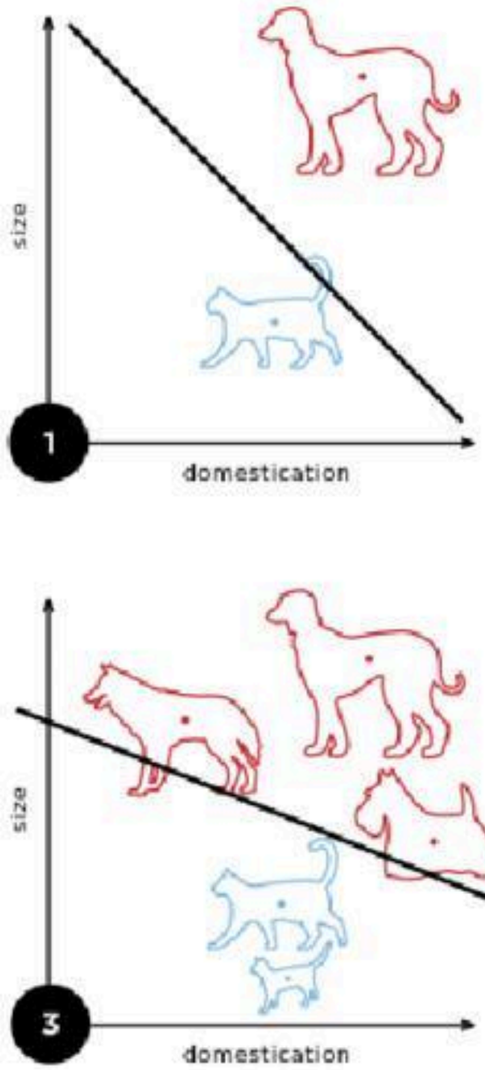
训练感知机

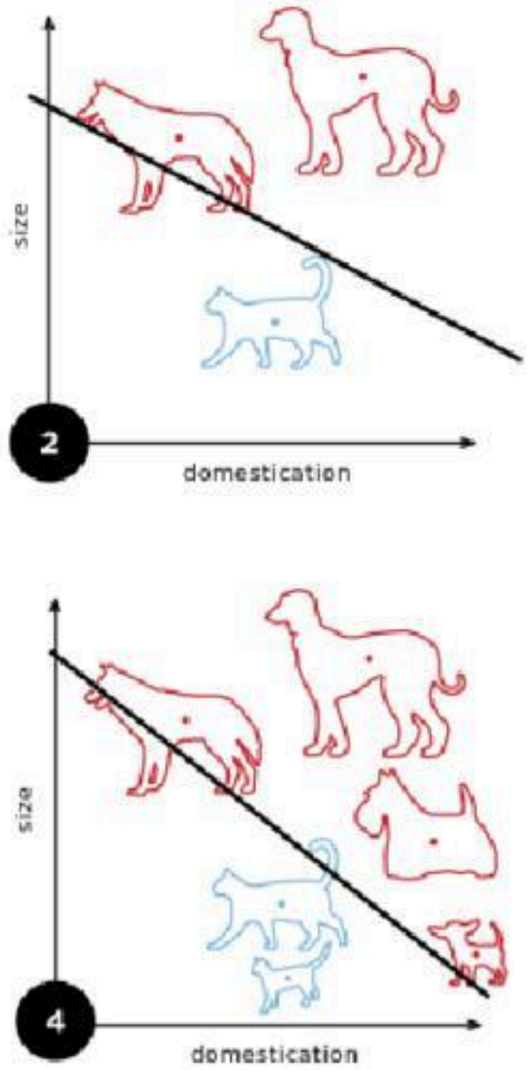
训练过程等价于使用**批量大小为1**的梯度下降

1. 初始化：$\mathbf{w} = \mathbf{0}$;
2. 循环，直到所有数据都被分类正确:
    1. **if** $y_i \left(\langle \mathbf{w_i}, \mathbf{x_i} \rangle\right) \leq 0$:
        1. $\mathbf{w_i} = \mathbf{w_i} + y_i \mathbf{x_i}$;

- 二分类输出$\{-1, 1\}$：$\hat{y}_i = \langle \mathbf{w_i}, \mathbf{x_i} \rangle$为正时输出1
    - 所以可以用$y_i \left(\langle \mathbf{w_i}, \mathbf{x_i} \rangle\right) = y_i \hat{y}_i$判断分类是否正确
- (*)**损失函数**：$l(y, \mathbf{x}; \mathbf{w}) = \max(0, -y \langle \mathbf{w}, \mathbf{x} \rangle)$
    - 梯度：$y_i \mathbf{x_i}$，进而得到步长为1的更新公式

感知机训练示例



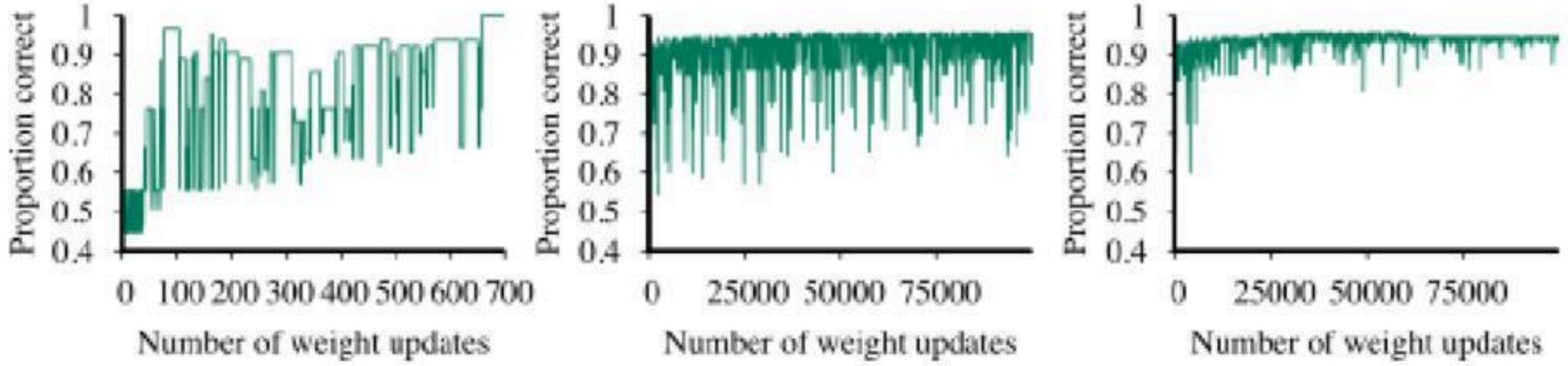


- 每步解决一个负例
- 第三步没出现负例

感知机：学习曲线

通常感知机**每次更新只处理一个样本**，速度慢但总能收敛（左图）

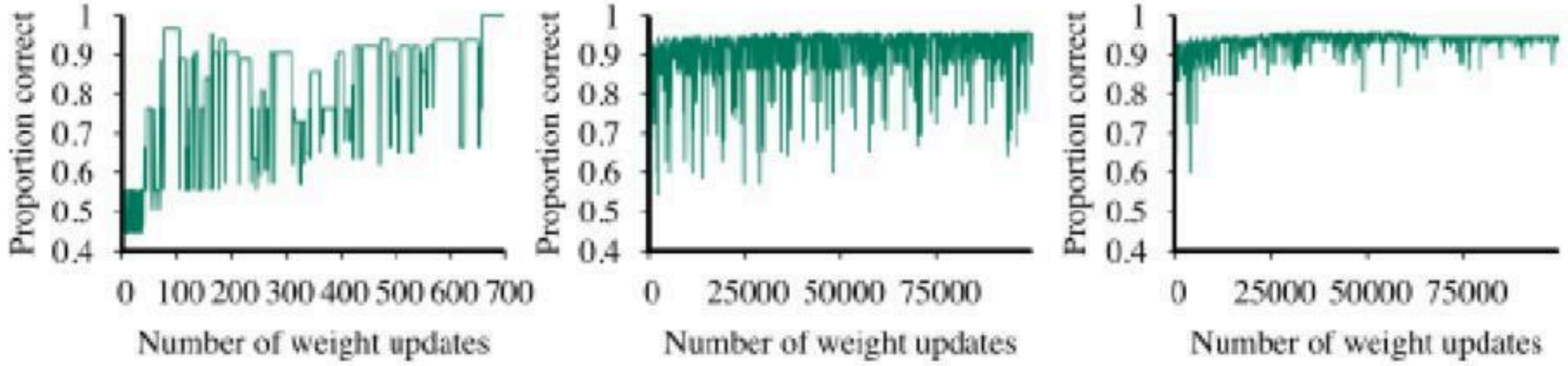
- 收敛过程很不稳定；63个样本消耗657步更新，平均每个样本被处理10次



感知机：学习曲线

通常感知机**每次更新只处理一个样本**，速度慢但总能收敛（左图）

- 收敛过程很不稳定；63个样本消耗657步更新，平均每个样本被处理10次



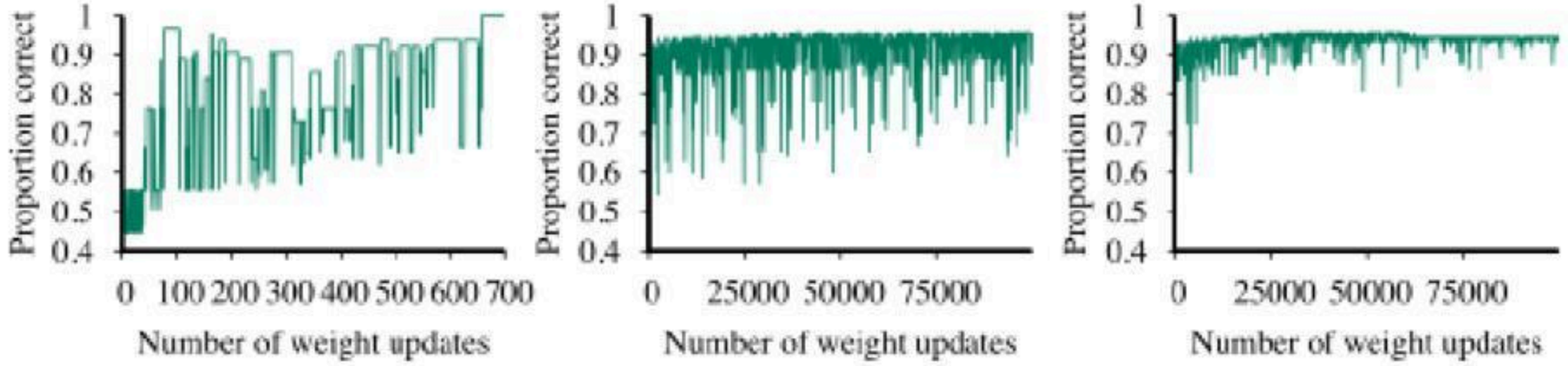
如果数据非线性可分，学习率 η 固定时算法不收敛（中图）

- 注意：横轴（更新步数）的量级不同

感知机：学习曲线

通常感知机**每次更新只处理一个样本**，速度慢但总能收敛（左图）

- 收敛过程很不稳定；63个样本消耗657步更新，平均每个样本被处理10次



如果数据非线性可分，学习率 η 固定时算法不收敛（中图）

- 注意：横轴（更新步数）的量级不同
- (*)如果学习率 η 以 $O(1/t)$ 的速率衰减，理论上可以收敛到最小误差解
  - 但搜索最小误差解是NP难：可能需要大量的样本步骤

小结

- 感知机是二分类模型，最早的AI模型之一
- 训练过程等价于使用批量大小为1的梯度下降
- 不能拟合XOR函数，间接导致第一次AI寒冬

逻辑回归

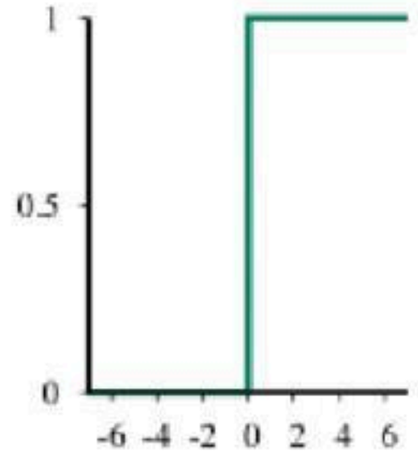
阶梯函数：硬性阈值

阈值函数：$h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$

- $Threshold(z) = 1$ if $z > 0$，即阶梯函数

硬性阈值：不连续，也就不可微

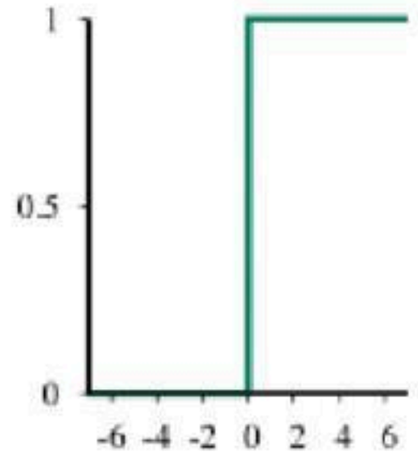
- 导致感知机的学习过程不可预测，很难收敛



阶梯函数：硬性阈值

阈值函数：$h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$

- $Threshold(z) = 1$ if $z > 0$，即阶梯函数



硬性阈值：不连续，也就不可微

- 导致感知机的学习过程不可预测，很难收敛

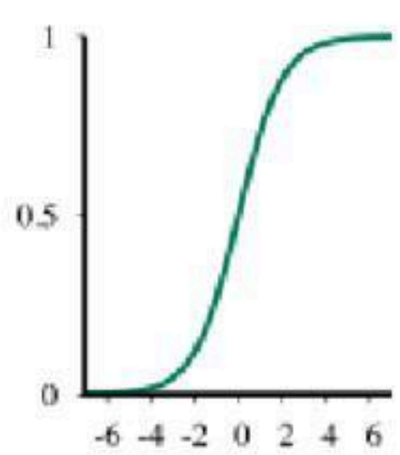
另外：感知机模型的预测结果非0即1，即使在决策边界附近也不动摇

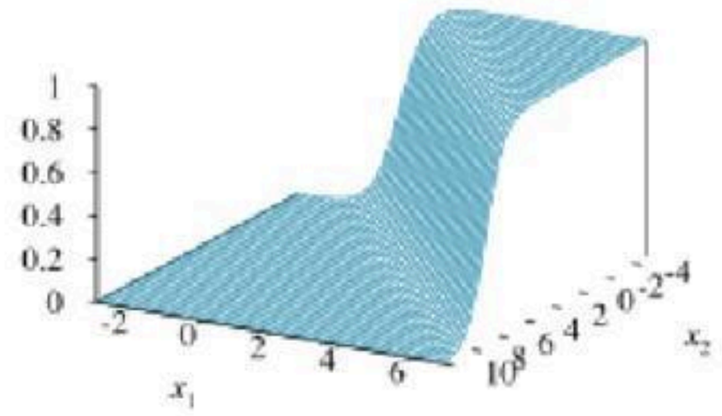
- 常识来说，决策边界附近有**不确定性**，容易受噪音干扰
  - 模型应该能把这类样本解释为“临界情况”

逻辑函数

逻辑函数（也称S曲线）：可以看成柔化的阈值函数

- 连续、可微的逼近函数（左图）：$Logistic(z) = \frac{1}{1+\exp^{-z}}$
  - 注：仅作为激活函数时，应当称为**Sigmoid函数**
- **逻辑回归模型**（右图）：$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1+\exp^{-\mathbf{w} \cdot \mathbf{x}}}$

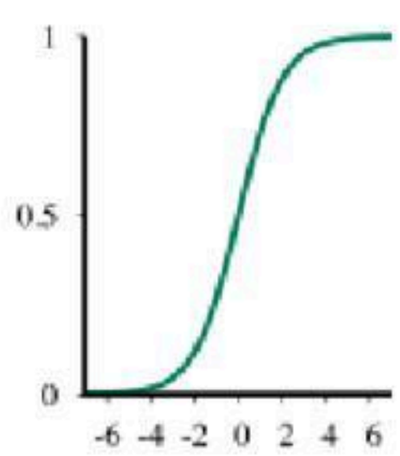


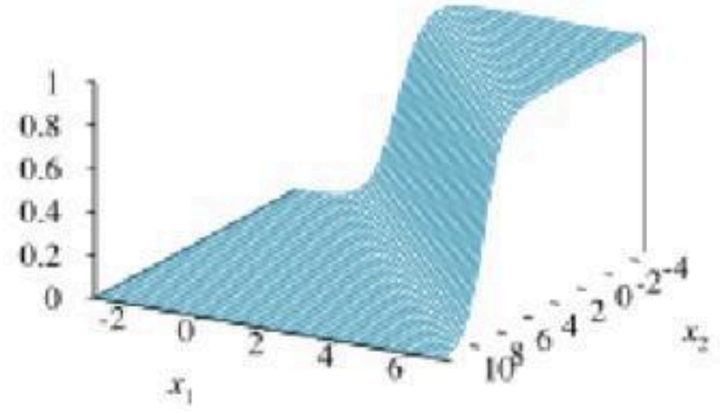


逻辑函数

逻辑函数（也称S曲线）：可以看成柔化的阈值函数

- 连续、可微的逼近函数（左图）：$Logistic(z) = \frac{1}{1+\exp^{-z}}$
  - 注：仅作为激活函数时，应当称为**Sigmoid函数**
- **逻辑回归模型**（右图）：$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1+\exp^{-\mathbf{w} \cdot \mathbf{x}}}$





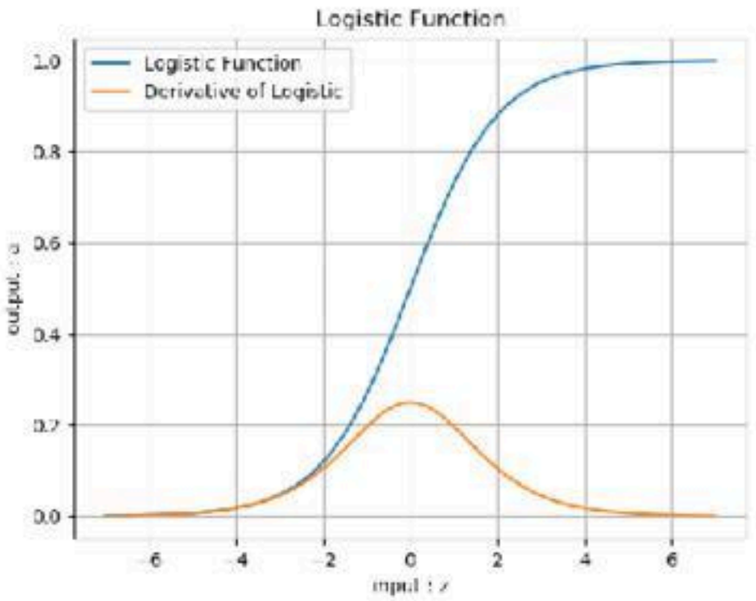
输出值在0，1之间，可以解释为（属于1类的）**概率值**

- 决策边界上概率为0.5：不确定性最大

逻辑函数：使用方法

返回概率值：接近1时认为是正例，否则是负例

- 训练时：预测值越接近标签值，那么误差越小，反向传播的力度就越小
- (*)推理时：阈值越大，准确率越高，召回率越低；阈值越小则相反，准确度越低，召回率越高。



比如：

- input=2时，output=0.88，而0.88>0.5，算作正例
- input=-1时，output=0.27，而0.27<0.5，算作负例

(*)逻辑回归：平方损失

逻辑回归很难计算解析解。下面采用平方损失计算梯度

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_i} &= \frac{\partial}{\partial w_i}(y-\hat{y})^2 \\ &= 2(y-\hat{y})\frac{\partial}{\partial w_i}(y-\hat{y}) \\ &= -2(y-\hat{y})g'(\mathbf{w}\cdot\mathbf{x})\frac{\partial}{\partial w_i}\mathbf{w}\cdot\mathbf{x} \\ &= -2(y-\hat{y})g'(\mathbf{w}\cdot\mathbf{x})x_i\end{aligned}$$

- g' 为逻辑函数的导数

(*)逻辑回归：平方损失

逻辑回归很难计算解析解。下面采用平方损失计算梯度

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_i} &= \frac{\partial}{\partial w_i}(y-\hat{y})^2 \\ &= 2(y-\hat{y})\frac{\partial}{\partial w_i}(y-\hat{y}) \\ &= -2(y-\hat{y})g'(\mathbf{w}\cdot\mathbf{x})\frac{\partial}{\partial w_i}\mathbf{w}\cdot\mathbf{x} \\ &= -2(y-\hat{y})g'(\mathbf{w}\cdot\mathbf{x})x_i\end{aligned}$$

- g' 为逻辑函数的导数

已知 $g'(z) = g(z)(1-g(z))$：$g'(\mathbf{w}\cdot\mathbf{x}) = \hat{y}(1-\hat{y})$

- 逻辑回归更新规则：$w_i \leftarrow w_i + \eta(y-\hat{y})\hat{y}(1-\hat{y})x_i$

(*)逻辑回归：交叉熵

实际应用通常使用**交叉熵**构造损失函数：$loss(w) = -[y \log \hat{g} + (1-y) \log(1-g)]$

- 交叉熵：度量概率分布之间的差异
- 标记：$z = \mathbf{w} \cdot \mathbf{x}, g = Logistic(z) = \frac{1}{1+e^{-z}}$

(*)逻辑回归：交叉熵

实际应用通常使用**交叉熵**构造损失函数：$loss(w) = -[y \log \hat{g} + (1-y) \log(1-g)]$

- 交叉熵：度量概率分布之间的差异
- 标记：$z = \mathbf{w} \cdot \mathbf{x}, g = Logistic(z) = \frac{1}{1+e^{-z}}$

巧妙之处在于，求导结果只剩一项减法：

$$\frac{\partial loss}{\partial z} = \frac{\partial loss}{\partial g} \frac{\partial g}{\partial z} = \frac{g-y}{g(1-g)} \cdot g(1-g) = g - y$$

- 损失函数满足二分类的要求，无论是正例还是反例，都是单调的
- 损失函数可导，便于使用反向传播算法
- 计算过程非常简单，一个减法就可以搞定

对数几率

几率 $odds = \frac{1}{1-g}$：正例相对负例的可能性比值

- 例如公平硬币：几率 $0.5/0.5 = 1$

对数几率

几率 $odds = \frac{1}{1-g}$：正例相对负例的可能性比值

- 例如公平硬币：几率 $0.5/0.5 = 1$

对数几率 log odds, logit：几率的对数

概率	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
反概率	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
几率	0	0.11	0.25	0.43	0.67	1	1.5	2.33	4	9	∞
对数几率	N/A	-2.19	-1.38	-0.84	-0.4	0	0.4	0.84	1.38	2.19	N/A

- 注意：几率不是线性增长，但对数几率接近线性关系

对数几率：线性逼近

假设对数几率可以用线性关系建模：$\log(odds) = \log \frac{g}{1-g} = \mathbf{w} \cdot \mathbf{x}$

- 两边取对数后整理得到：$g = \frac{1}{1+e^{-(\mathbf{w} \cdot \mathbf{x})}} = \frac{1}{1+e^{-z}}$
  - 注意：此式正是逻辑函数的定义！

对数几率：线性逼近

假设对数几率可以用线性关系建模：$\log(odds) = \log \frac{g}{1-g} = \mathbf{w} \cdot \mathbf{x}$

- 两边取对数后整理得到：$g = \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x})}} = \frac{1}{1+e^{-z}}$
  - 注意：此式正是逻辑函数的定义！
- 本质：用线性回归模型的预测结果来**逼近样本分类的对数几率 logit**
  - 即“逻辑 logistic 回归”命名的原因

对数几率：线性逼近

假设对数几率可以用线性关系建模：$\log(odds) = \log \frac{g}{1-g} = \mathbf{w} \cdot \mathbf{x}$

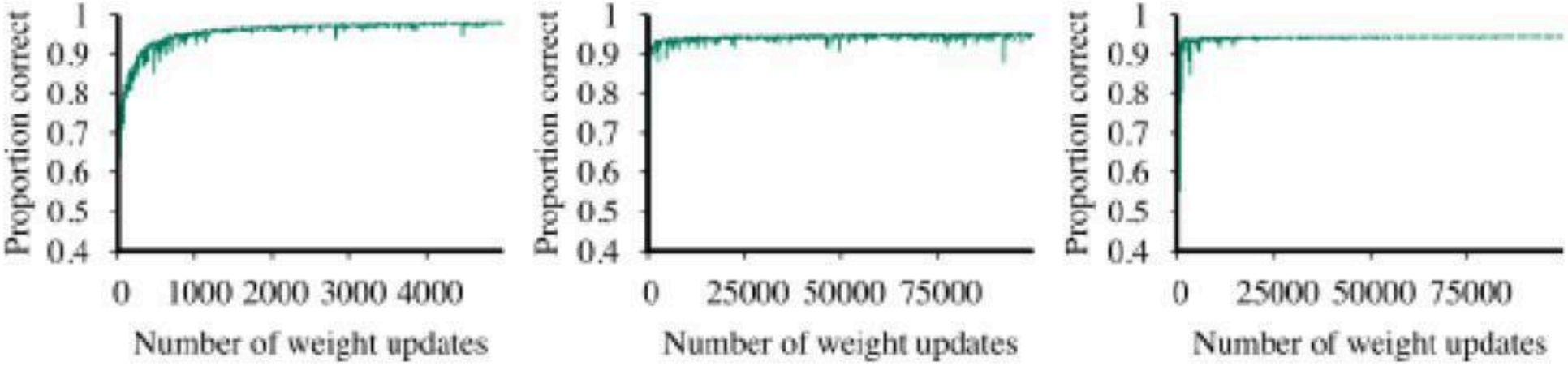
- 两边取对数后整理得到：$g = \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x})}} = \frac{1}{1+e^{-z}}$
  - 注意：此式正是逻辑函数的定义！
- 本质：用线性回归模型的预测结果来**逼近样本分类的对数几率 logit**
  - 即“逻辑 logistic 回归”命名的原因

把线性回归的成功经验引入到分类问题中的优点：

- 不仅预测出类别，而且得到了近似的概率值：对许多需要利用概率辅助决策的任务很有用
- 对率函数是任意阶可导的凸函数：有很好的数学性质，易于数值优化算法求解

逻辑回归：学习曲线

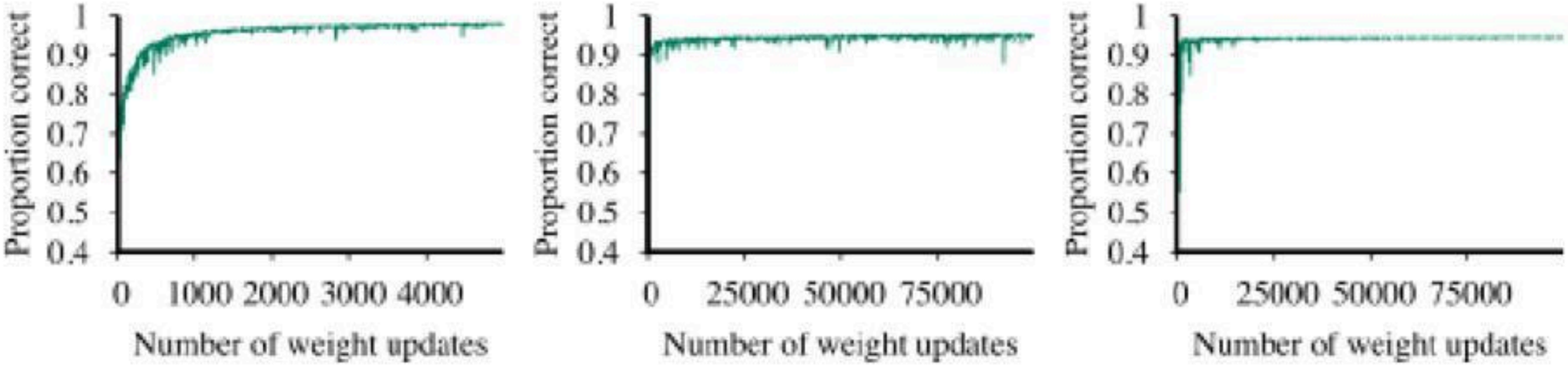
逻辑回归模型在业界是常用基础模型



- 线性可分时（左图）：比感知机收敛慢，但非常稳定

逻辑回归：学习曲线

逻辑回归模型在业界是常用基础模型



- 线性可分时（左图）：比感知机收敛慢，但非常稳定
- 当数据有噪音且线性不可分时：比感知机收敛更快、更可靠

逻辑门运算问题

下面我们尝试使用逻辑回归解决**逻辑门运算**问题：

样本	x_1	x_2	与	与非	或	或非
1	0	0	0	1	0	1
2	0	1	0	1	1	0
3	1	0	0	1	1	0
4	1	1	1	0	1	0

- 输入变量有两个特征x_1, x_2
- 输出值为对应逻辑运算的结果

逻辑门运算问题

下面我们尝试使用逻辑回归解决**逻辑门运算**问题：

样本	x_1	x_2	与	与非	或	或非
1	0	0	0	1	0	1
2	0	1	0	1	1	0
3	1	0	0	1	1	0
4	1	1	1	0	1	0

- 输入变量有两个特征x_1, x_2
- 输出值为对应逻辑运算的结果

理论上，给定确定真值表的**4个样本**，就可以得出逻辑运算的全部规律

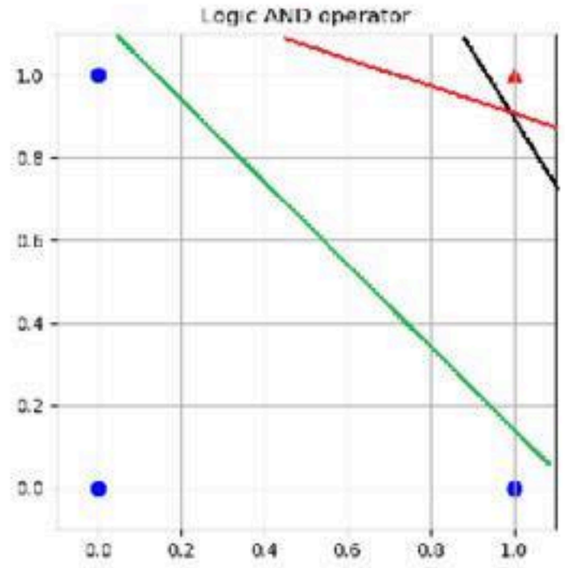
- 思考：如何将问题转化成一个线性分类模型的描述?

与门问题

样本的两个特征可以用2维坐标轴表示

例如与门：

- 4个样本分布在角落
- 两种输出值用不同的符号表示
- 此问题可以有无数条分界边界

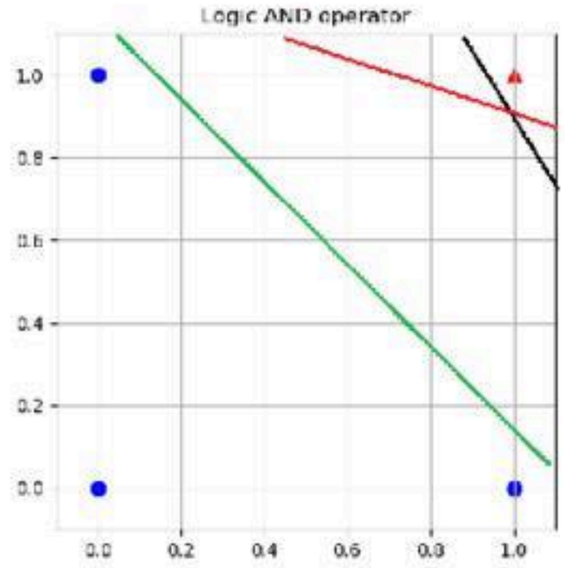


与门问题

样本的两个特征可以用2维坐标轴表示

例如与门:

- 4个样本分布在角落
- 两种输出值用不同的符号表示
- 此问题可以有无数条分界边界



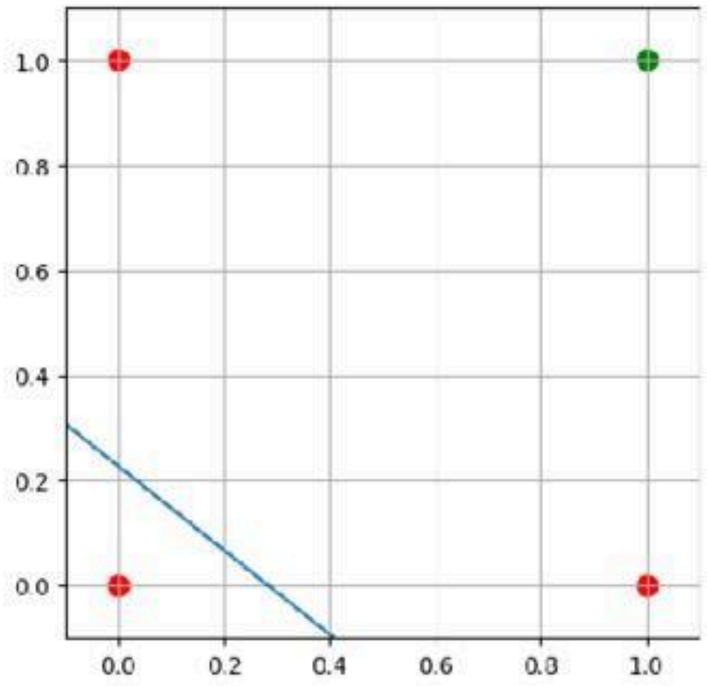

```
X = np.array([0,0,0,1,1,0,1,1]).reshape(4,2)
Y = np.array([0,0,0,1]).reshape(4,1)
```

与门问题：训练

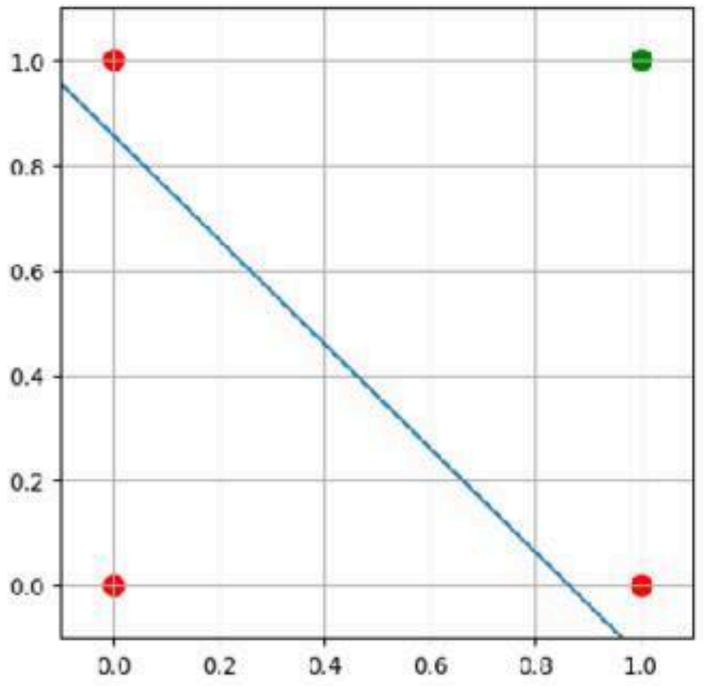
```
def logit(w, x):
    return 1/ (1 + np.exp(- np.dot(w, x)))

def train(X, Y, lr=1, epoch=10000):
    Xh = np.hstack((X, np.ones((X.shape[0],1))))
    w = np.random.rand(Xh.shape[1])
    for ei in range(epoch):
        loss = 0
        for di, x in enumerate(Xh):
            d = logit(w, x) - np.squeeze(Y[di])
            loss += abs(d)
```

```
epoch #0: loss = 2.700277
epoch #1000: loss = 0.016769
epoch #2000: loss = 0.008438
...
epoch #9000: loss = 0.001885
```

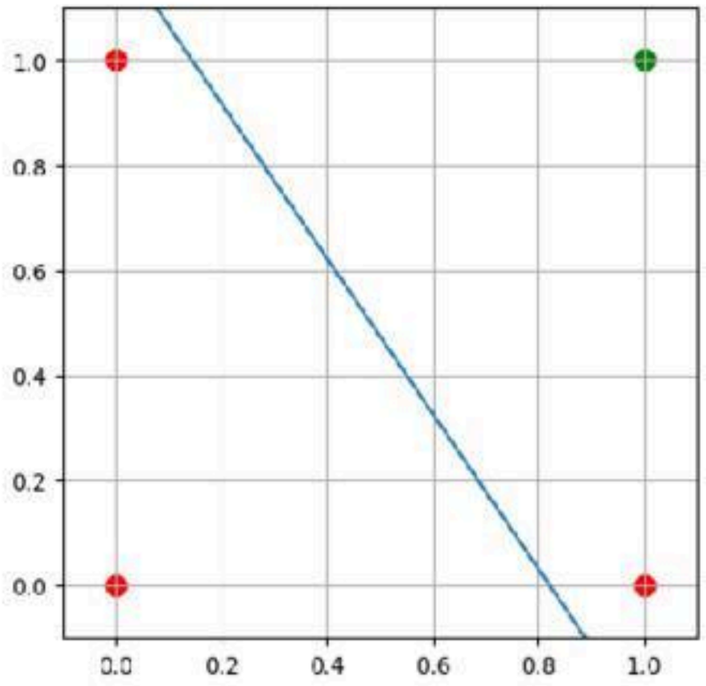
与门问题：图示1



- epoch 1

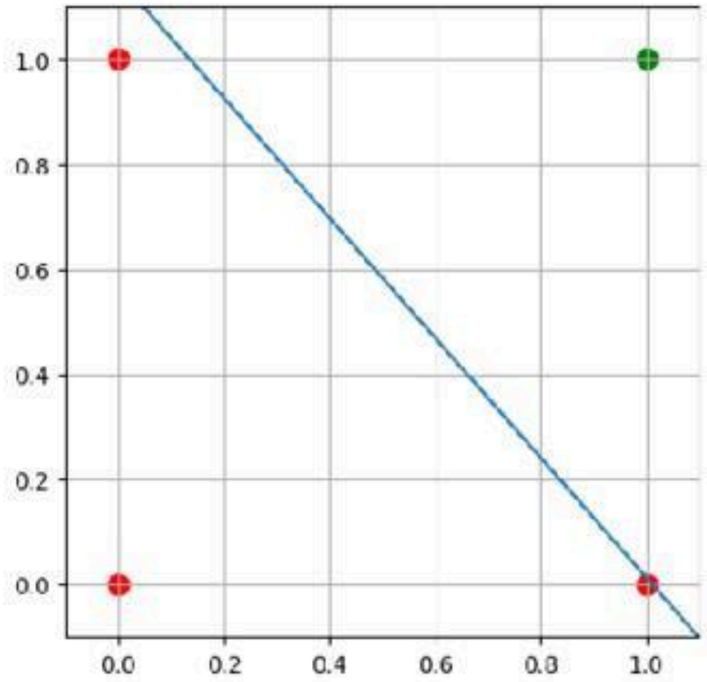


- epoch 2

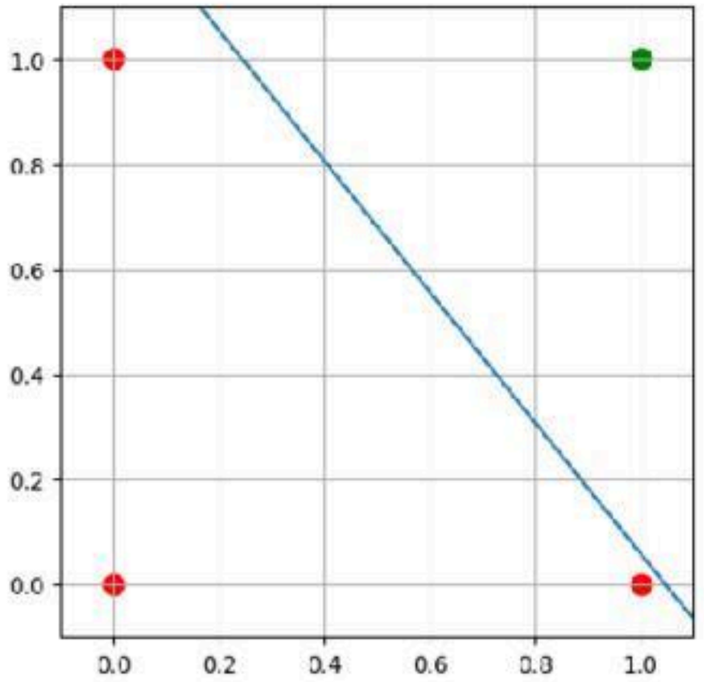


- epoch 3

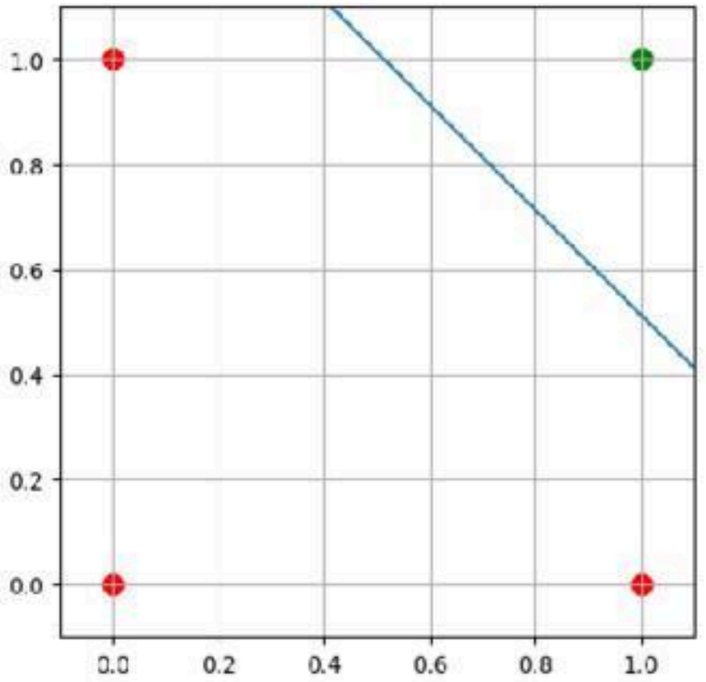
与门问题：图示2



- epoch 4



- epoch 5

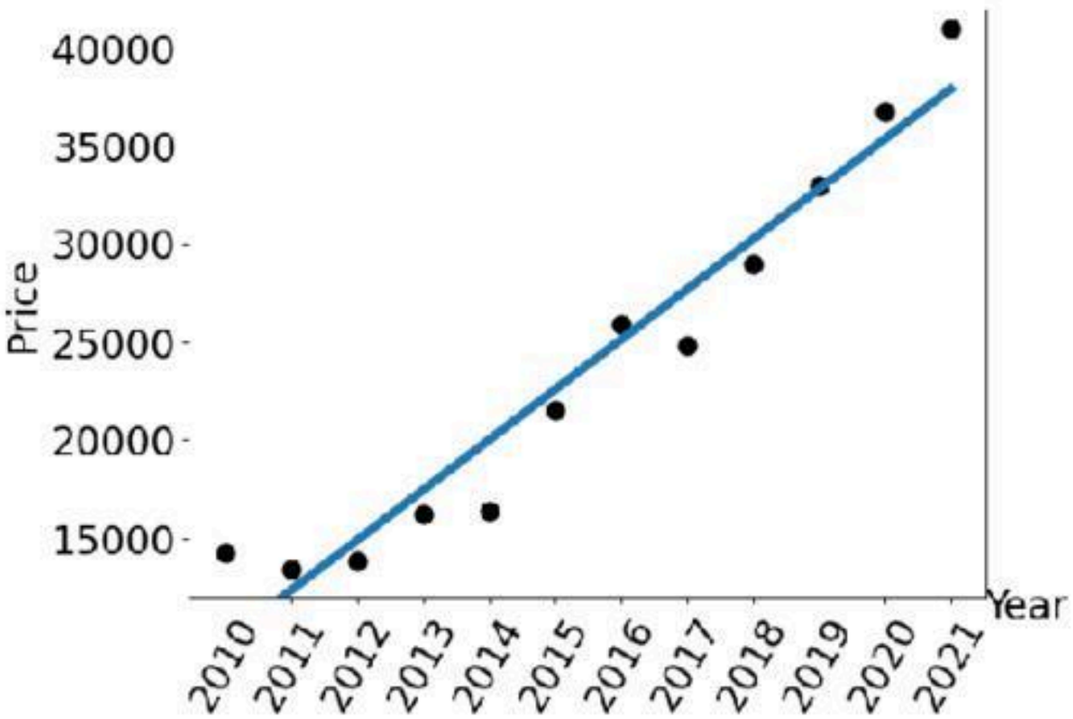


- epoch 10000

softmax回归

回归、分类

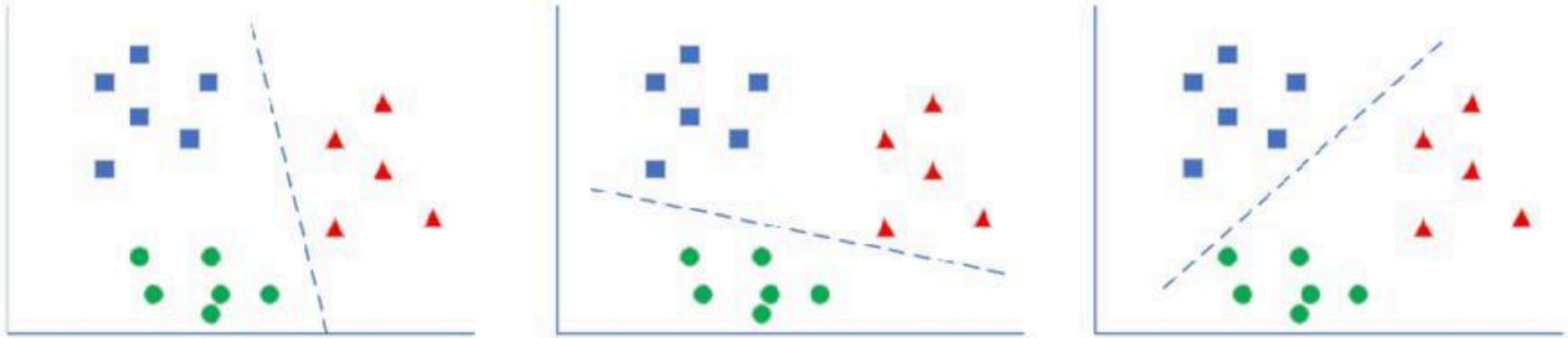
回归：输出连续数值



分类：输出离散类别

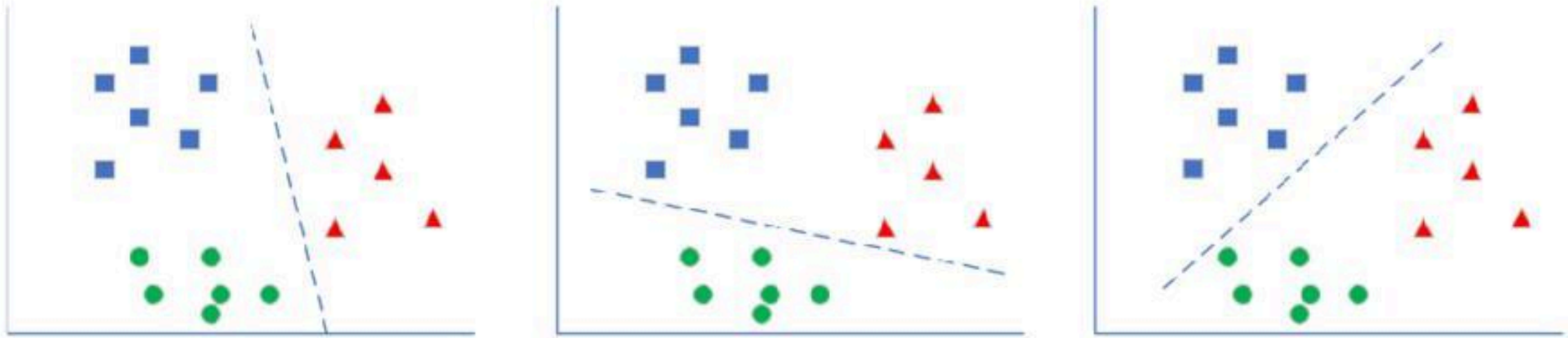
使用二分类解决多分类问题

多分类问题可以用二分类算法解决，例如**一对一**策略：

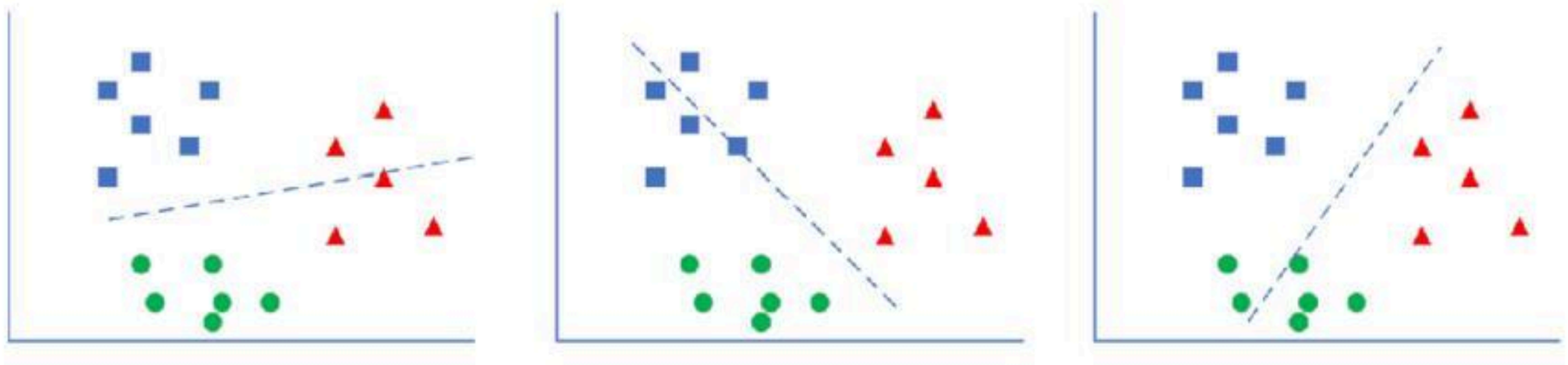


使用二分类解决多分类问题

多分类问题可以用二分类算法解决，例如**一对一**策略：

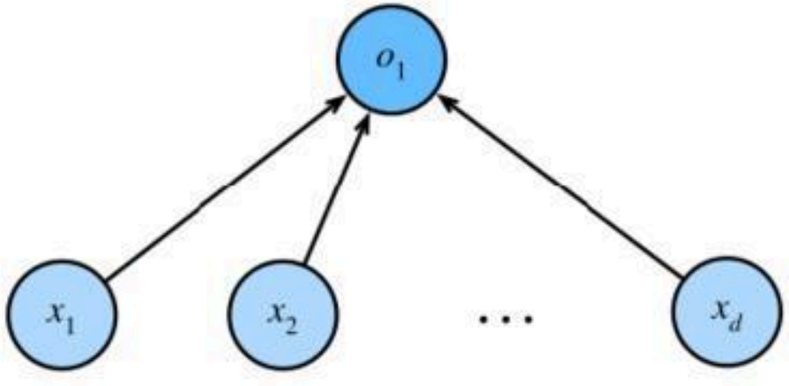


一对多策略可以更加强调每个类别：



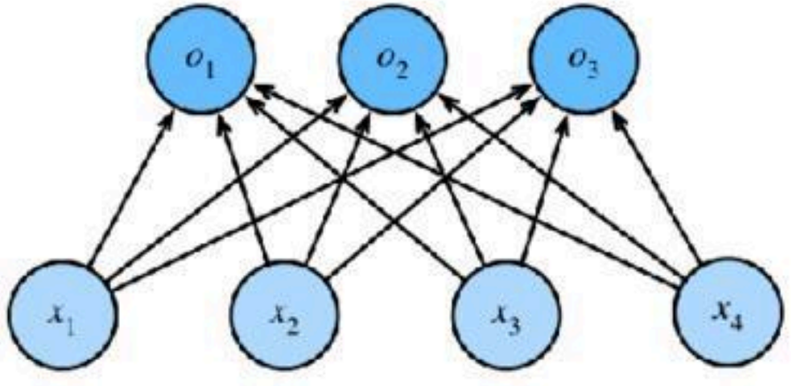
从回归到多类分类

回归



- 输出单个连续值
  - 输出：预测值
- 误差：输出与真实值的差别

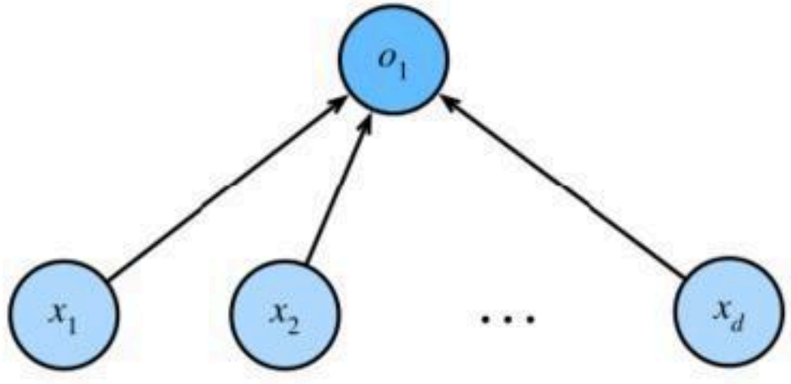
分类



- 输出多个不同的连续值
  - 每个输出：预测类别的**可信程度**
- 误差：输出与真实概率的差别

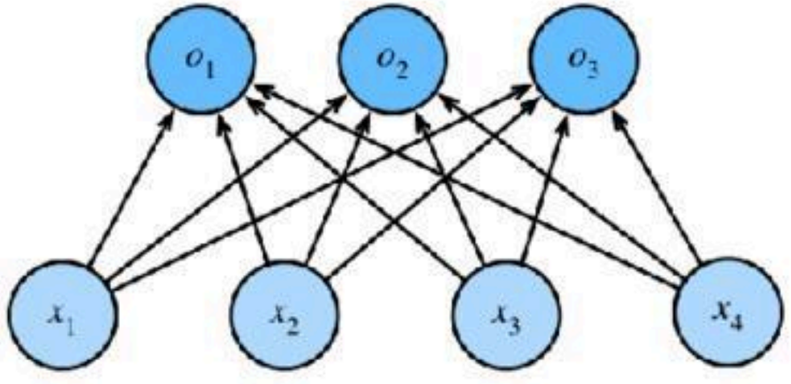
从回归到多类分类

回归



- 输出单个连续值
  - 输出：预测值
- 误差：输出与真实值的差别

分类



- 输出多个不同的连续值
  - 每个输出：预测类别的**可信程度**
- 误差：输出与真实概率的差别

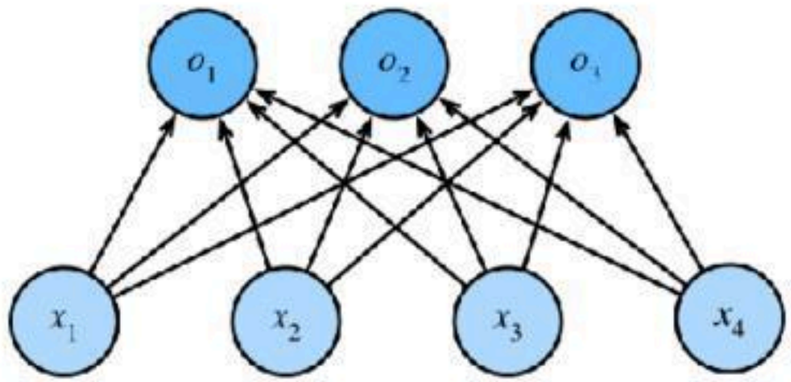
问题（即马上要讨论的内容）：

- 如何从输出的多个实数值构造概率?
- 如何表达“真实概率”?
- 如何计算概率之间的差别?

独热编码

独热 one-hot 编码：单有效位的向量，编码真实类别归属

- C个类；$\mathbf{y} = [y_1, y_2, .., y_C]$
- $y_c = 1$表示正确类别是c；否则$y_c = 0$



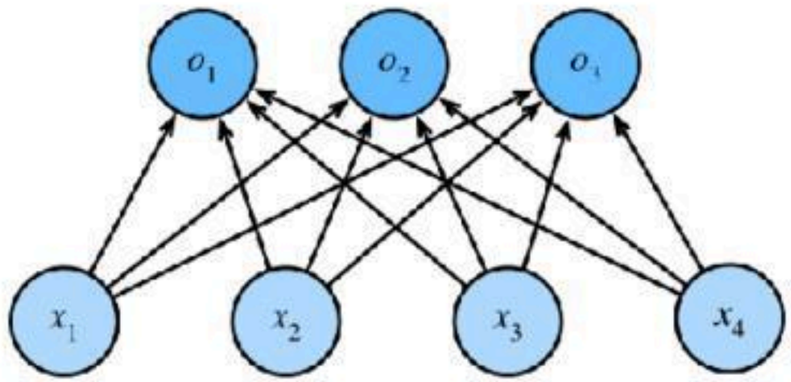
例如类别3的编码，假设$C = 10, c = 3$:

- $[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

独热编码

独热 one-hot 编码：单有效位的向量，编码真实类别归属

- C个类；$\mathbf{y} = [y_1, y_2, .., y_C]$
- $y_c = 1$表示正确类别是c；否则$y_c = 0$



例如类别3的编码，假设$C = 10, c = 3$：

- $[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

我们希望预测正确：预测值$\hat{\mathbf{y}}$与真实值$\mathbf{y}$越接近越好

- 思考：（回顾二类分类的建模过程）如何计算一个取值为0/1的向量?

感知机原型：多类推广

简单输出C个线性回归的激活值：

$$g_j = \sigma\left(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j\right), \sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

- 线性回归输出：$\mathbf{z} = [z_1, z_2, .., z_C]^T, z_j = \langle \mathbf{w}_j, \mathbf{x} \rangle + b_j$

感知机原型：多类推广

简单输出C个线性回归的激活值：

$$g_j = \sigma\left(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j\right), \sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

- 线性回归输出：$\mathbf{z} = [z_1, z_2, .., z_C]^T, z_j = \langle \mathbf{w}_j, \mathbf{x} \rangle + b_j$

问题：样本可能被划分到多个类别，即多个输出值为1

- 视觉中多物体检测问题；但我们只考虑单一归属

线性输出的最大值

另一简单方案是取C个线性回归输出的最大值作为预测的类别

$$\max_j z_j, z_j = \mathbf{w}_j \cdot \mathbf{x}$$

例如：$z = [1, 0, -1]$，取max操作变成$z = [1, 0, 0]$

- 三者相加为1，并且认为该样本属于第一类。

线性输出的最大值

另一简单方案是取C个线性回归输出的最大值作为预测的类别

$$\max_j z_j, z_j = \mathbf{w}_j \cdot \mathbf{x}$$

例如：$z = [1, 0, -1]$，取max操作变成$z = [1, 0, 0]$

- 三者相加为1，并且认为该样本属于第一类。

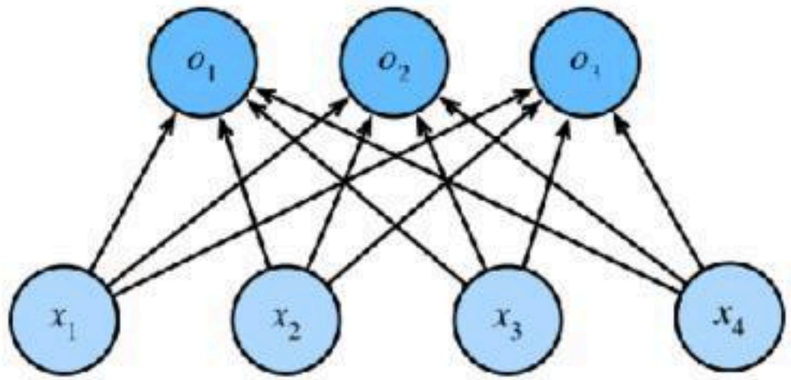
但是有两个不足：

- 分类结果只保留非0即1的信息，没有元素间相差量的信息，可以理解是“硬性最大值”
- max操作本身不可导，无法用在反向传播中

softmax运算

softmax计算线性回归输出$\mathbf{z} = [z_1, z_2, .., z_C]^T$中**每个分量的相对比重**:

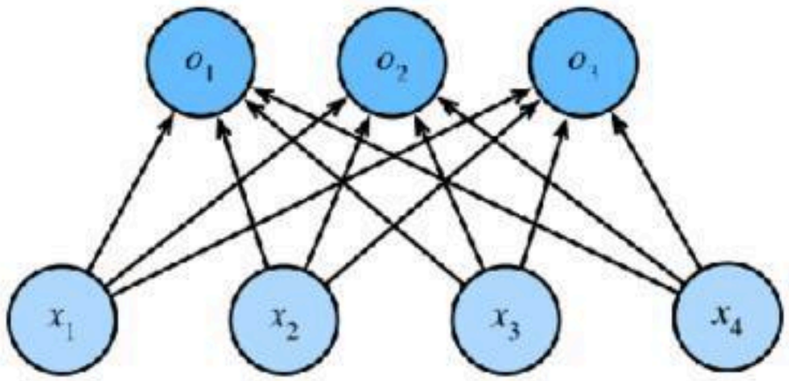
$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{z}), \hat{y}_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)}$$



softmax运算

softmax计算线性回归输出$\mathbf{z} = [z_1, z_2, .., z_C]^T$中**每个分量的相对比重**：

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{z}), \hat{y}_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)}$$



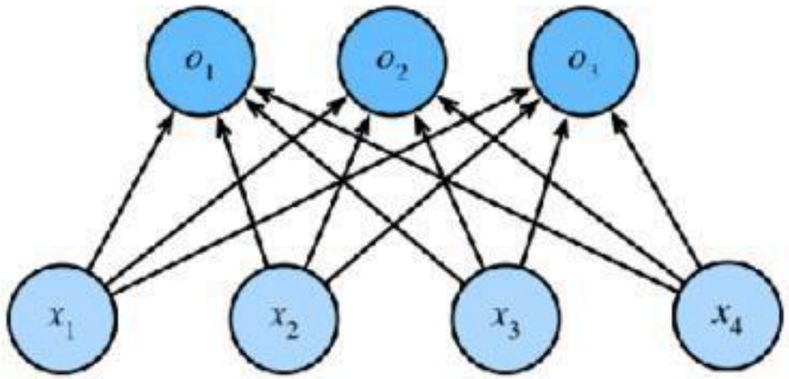
- softmax本质上是**归一化**
  - $0 \leq \hat{y}_j \leq 1$，并且 $\sum_j \hat{y}_j = 1$

(*)小批量样本

假设批量大小为B，特征维度为d

$$\mathbf{O} = \mathbf{X}\mathbf{W} + \mathbf{b}$$
$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{O})$$

- $\mathbf{X} \in \mathbb{R}^{B \times d}, \mathbf{W} \in \mathbb{R}^{d \times C}, \mathbf{b} \in \mathbb{R}^{1 \times C}$

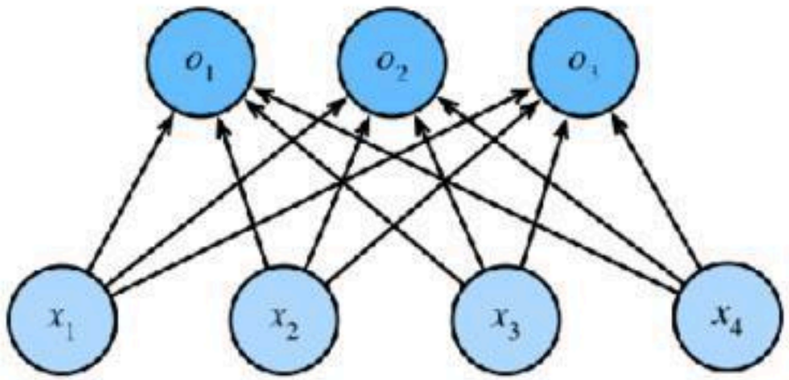


(*)小批量样本

假设批量大小为B，特征维度为d

$$\mathbf{O} = \mathbf{X}\mathbf{W} + \mathbf{b}$$
$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{O})$$

- $\mathbf{X} \in \mathbb{R}^{B \times d}, \mathbf{W} \in \mathbb{R}^{d \times C}, \mathbf{b} \in \mathbb{R}^{1 \times C}$



问题：如何计算误差（损失值）？

(*)度量概率相似程度

度量概率分布相似程度：采样点上的**重合度**

$$\|\mathbf{y}, \hat{\mathbf{y}}\| \propto \prod_c \hat{y}_c^{y_c}$$

- $a^0 = 1, a^1 = a$：等价于将 $y_c = 1$对应的$\hat{y}_c$ 选出来
  - 例如：$\hat{y}_c = y_c = 1$ 时，$\hat{y}_c^{y_c} = 1$
- 数值上与**内积**计算类似

(*)度量概率相似程度

度量概率分布相似程度：采样点上的**重合度**

$$\|\mathbf{y}, \hat{\mathbf{y}}\| \propto \prod_c \hat{y}_c^{y_c}$$

- $a^0 = 1, a^1 = a$：等价于将 $y_c = 1$对应的$\hat{y}_c$ 选出来
  - 例如：$\hat{y}_c = y_c = 1$ 时，$\hat{y}_c^{y_c} = 1$
- 数值上与**内积**计算类似

问题：大量概率值的乘积导致下溢出?

对数似然

度量概率分布相似程度：采样点上的**重合度**

$$\|\mathbf{y}, \hat{\mathbf{y}}\| \propto \prod_c \hat{y}_c^{y_c}$$

取负对数得到损失函数：

$$\begin{aligned} Loss(\mathbf{y}, \hat{\mathbf{y}}) &= -\log \prod_c \hat{y}_c^{y_c} \\ &= -\sum_c y_c \log \hat{y}_c \end{aligned}$$

对数似然

度量概率分布相似程度：采样点上的**重合度**

$$\|\mathbf{y}, \hat{\mathbf{y}}\| \propto \prod_c \hat{y}_c^{y_c}$$

取负对数得到损失函数：

$$\begin{aligned} Loss(\mathbf{y}, \hat{\mathbf{y}}) &= -\log \prod_c \hat{y}_c^{y_c} \\ &= -\sum_c y_c \log \hat{y}_c \end{aligned}$$

- 常用数学技巧。
  - 注意：取对数**不改变序关系**
- 数值计算：加法总是比乘法好

(*)信息论解释

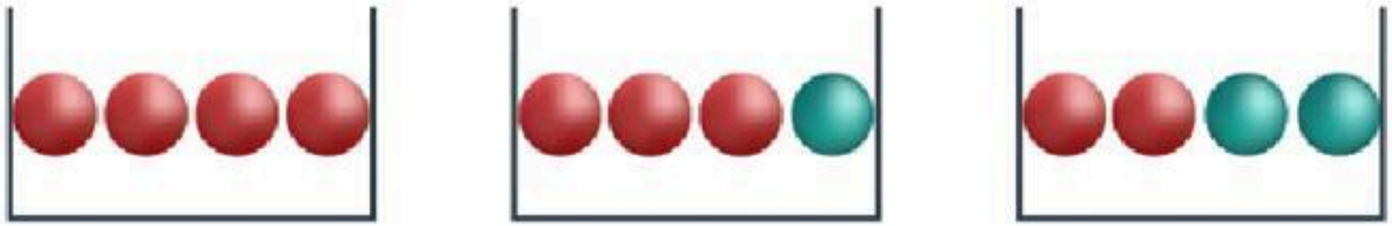
在信息论中，**熵 entropy**用于度量数据中的信息量

$$H(\mathbf{P}) = -\sum_i \mathbf{P}_i \log \mathbf{P}_i$$

对从分布 $\mathbf{P}$ 中随机抽样的数据编码，至少需要 $H[\mathbf{P}]$ “**纳特 nat**”。

(*)熵的理解

想象从盒子里抽奖



- 熵可以理解为：对数据流进行连续预测的**预期“惊异”**

交叉熵

交叉熵 cross entropy用于度量两个概率分布的差异

$$H(\mathbf{P}, \mathbf{Q}) = -\sum_i P_i \log Q_i$$

- 即前面损失函数的定义
  - 损失函数：$-\sum_c y_c \log \hat{y}_c$
  - 预测分布是$\mathbf{Q}$，但实际分布是$\mathbf{P}$

交叉熵

交叉熵 cross entropy用于度量两个概率分布的差异

$$H(\mathbf{P}, \mathbf{Q}) = -\sum_i P_i \log Q_i$$

- 即前面损失函数的定义
  - 损失函数：$-\sum_c y_c \log \hat{y}_c$
  - 预测分布是$\mathbf{Q}$，但实际分布是$\mathbf{P}$

(*)数学上漂亮：梯度是预测与真实概率的差值

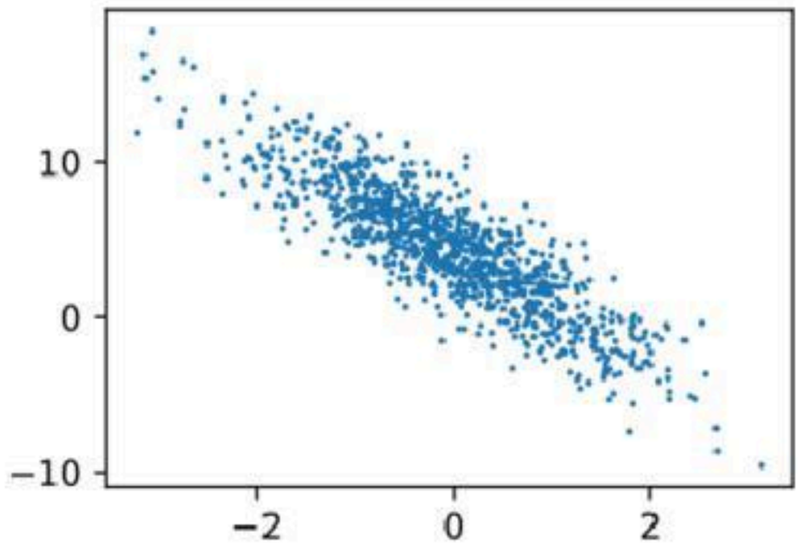
- $\frac{\partial}{\partial z_i} Loss(\mathbf{y}, \hat{\mathbf{y}}) = \text{softmax}(\mathbf{z})_i - y_i = \hat{y}_i - y_i$

实验：图像分类数据集

实际工程问题不是数学题

解决工程问题的思路需要考虑实际情况

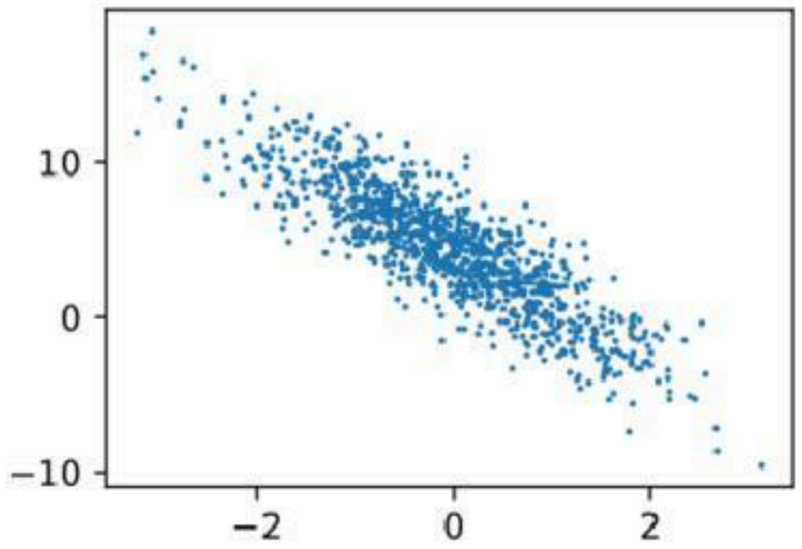
- 人造数据：训练出来的模型**只能拟合给定的数据**



实际工程问题不是数学题

解决工程问题的思路需要考虑实际情况

- 人造数据：训练出来的模型**只能拟合给定的数据**



优化目标：保证理论上参数对训练数据是最优的

- 实际应用：通常是**完全不同**的数据

训练集、测试集

测试集：最终评测时才能拿到的数据，只能用一次

- 对于训练过程来说是**未知**的

类比：训练集=日常习题；测试集=考试题

- 训练参数的过程：通过习题掌握**解题模式**

训练集、测试集

测试集：最终评测时才能拿到的数据，只能用一次

- 对于训练过程来说是**未知**的

类比：训练集=日常习题；测试集=考试题

- 训练参数的过程：通过习题掌握**解题模式**

模拟实际应用：将数据划分成两部分

- 对比：全部数据当成训练集，如何评测?

实验：softmax回归的从零开始实现

实验： softmax回归的简洁实现

Review

本章内容

1. 感知机模型
2. 逻辑回归模型
3. softmax回归模型
4. 实验：线性分类模型

重点：阶梯函数、逻辑函数、softmax函数；独热编码；求解线性分类问题并实现算法。

难点：交叉熵。

学习目标

1. 掌握基于硬性阈值函数的感知机原理
2. 掌握基于柔性决策边界的逻辑回归原理
3. 理解分类问题中softmax运算的作用
4. 理解softmax回归模型的一般形式、损失函数、优化算法
5. 掌握使用softmax回归解决分类问题的实现方法
6. 了解信息论中交叉熵用于度量概率分布的差异

问题

1. 简述感知机模型的原理。
2. 简述逻辑回归模型的原理。
3. 简述分类问题中softmax运算的作用。
4. 简述独热 one-hot 编码在分类问题中的应用。
5. 简述分类问题中损失函数的构造方法，及其计算方法。
6. (*) 简述信息论中交叉熵的作用，及其在分类问题中的应用。