机器学习与数据挖掘

线性分类器

课程大纲

- 二分类
- 多分类

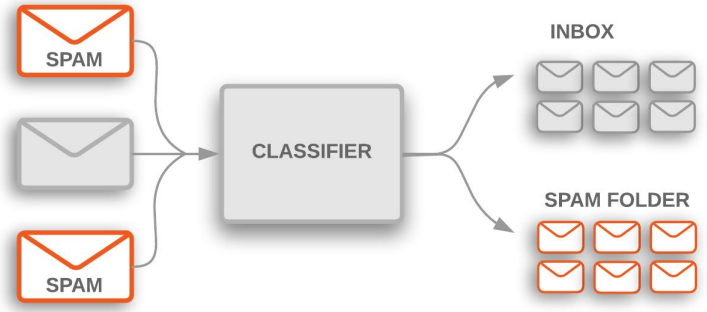
什么是分类

- 分类是监督学习的一个核心任务
- 目标是学习一个模型（称为分类器）
- 该模型能将输入数据（特征）映射到一个预定义的离散类别
- 与回归的区别： 回归预测的是连续值（如房价、温度），而分类预测的是类别标签（如“猫”或“狗”，“垃圾邮件”或“非垃圾邮件”）

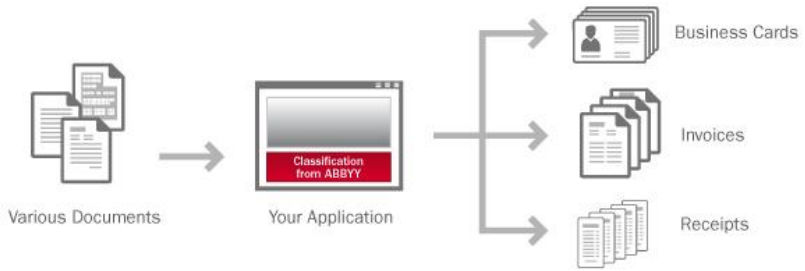
示例

- 图像识别

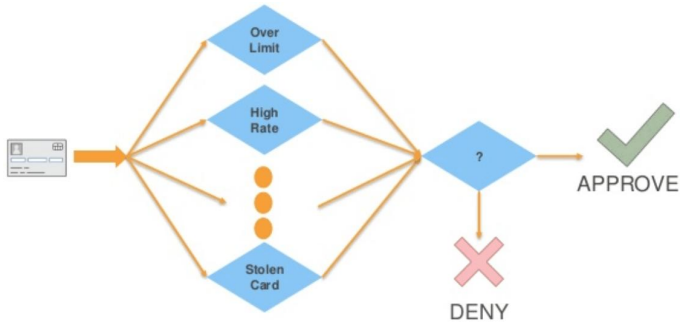
- 垃圾邮件检测



- 文档分类

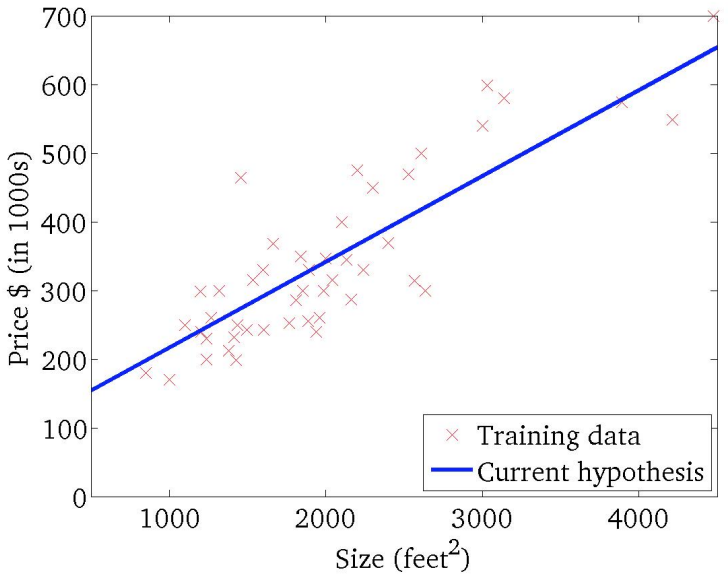


- 交易欺诈检测



逻辑回归

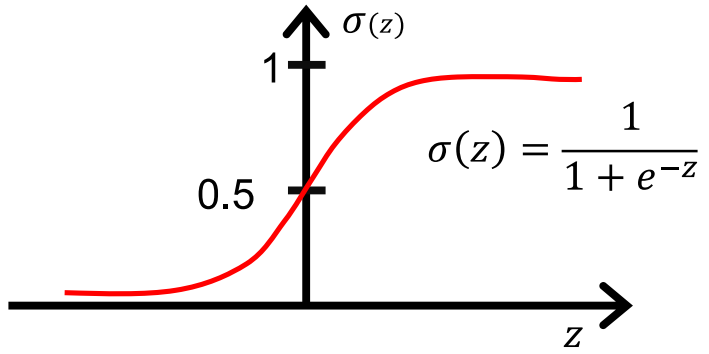
- 在分类问题中，目标变量 $y \in \{0, 1\}$
- 在线性回归中，输出 $f(\boldsymbol{x}) = \boldsymbol{xw}$ 的取值范围是 $[-\infty, +\infty]$



- 线性回归的输出值与分类任务中的目标值不兼容

- Sigmoid/logistic 函数

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



- 逻辑回归

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{x}\boldsymbol{w})$$

- 线性回归

$$f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w}$$

- 输出范围从 $[-\infty, +\infty]$ 转换为 $[0, 1]$

代价函数

- 目标
  - 如果真实标签 $y = 1$,我们希望 $f(\boldsymbol{x}) = \sigma(\boldsymbol{x}\boldsymbol{w})$ 接近于1
  - 如果真实标签 $y = 0$,我们希望 $f(\boldsymbol{x}) = \sigma(\boldsymbol{x}\boldsymbol{w})$ 接近于0

- 为了实现这个目标，我们可以定义一个类似于回归中的代价函数：

$$L(\boldsymbol{w}) = (\sigma(\boldsymbol{x}\boldsymbol{w}) - y)^2$$

- 或者，我们也可以寻求最小化：

$$L(\boldsymbol{w}) = \begin{cases} -\log\left(\sigma(\boldsymbol{x}\boldsymbol{w})\right) & \text{if } y = 1 \\ -\log\left(1-\sigma(\boldsymbol{x}\boldsymbol{w})\right) & \text{if } y = 0 \end{cases}$$

- 上面的目标可以等价地写成：

$$L(\boldsymbol{w}) = -y\log\left(\sigma(\boldsymbol{x}\boldsymbol{w})\right) - (1-y)\log\left(1-\sigma(\boldsymbol{x}\boldsymbol{w})\right)$$

如果 $y = 1, L(w)$ *简化为* $L(w) = -log\left(\sigma(xw)\right);$
否则，如果 $y = 0, L(w)$ *简化为 to* $L(w) = -log\left(1-\sigma(xw)\right)$

- 上面的损失函数被称为交叉熵损失

- 交叉熵损失：

$$L(\boldsymbol{w}) = -y\log\left(\sigma(\boldsymbol{x}\boldsymbol{w})\right) - (1-y)\log\left(1-\sigma(\boldsymbol{x}\boldsymbol{w})\right)$$

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

- 二阶导数（海森矩阵）：

$$H = \frac{\partial^2 L(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^{\mathrm{T}}} = xx^T\sigma(xw)(1-\sigma(xw))$$

- 判断半正定性：

$$v^T H v = (x^T v)^2 \sigma(xw)(1-\sigma(xw)) \geq 0$$

- 交叉熵损失函数的海森矩阵是半正定的，所以该函数是一个凸函数

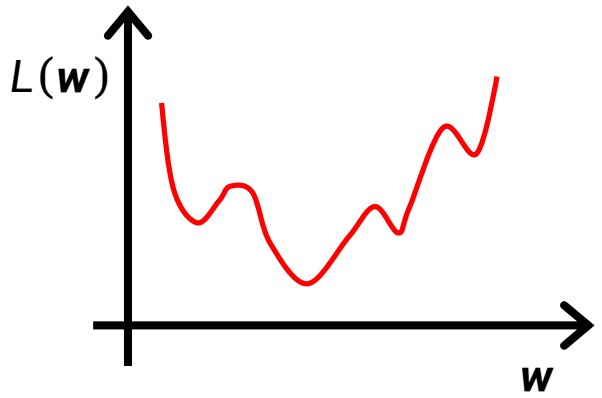
- 哪个代价函数更好？

函数 f(z) = $\log(\sigma(z))$ 的形状是什么样的？

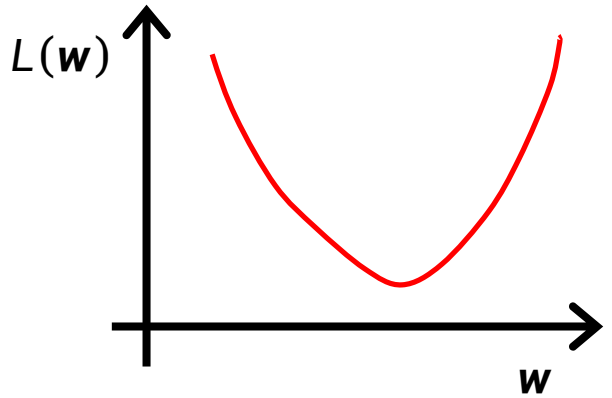
平方误差: $L(\boldsymbol{w}) = (\sigma(\boldsymbol{xw}) - y)^2$

交叉熵: $L(\boldsymbol{w}) = -[y\log(\sigma(\boldsymbol{xw})) + (1-y)\log(1-\sigma(\boldsymbol{xw}))]$

- 平方损失是非凸的



- 交叉熵是凸的



凸函数更易于优化

- 在下一次课中，我们将从精确建模的角度，展示使用交叉熵损失的另一个优点

解析解

- 交叉熵损失

$$L(\boldsymbol{w}) = -y\log\left(\sigma(\boldsymbol{x}\boldsymbol{w})\right) - (1-y)\log\left(1-\sigma(\boldsymbol{x}\boldsymbol{w})\right)$$

- 交叉熵损失的梯度

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{1}{N}\sum_{\ell=1}^{N}\left[\sigma(\boldsymbol{x}^{(\ell)}\boldsymbol{w}) - y^{(\ell)}\right]\boldsymbol{x}^{(\ell)T}$$

- 求解 $\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0$ 可以得到最优的 $\boldsymbol{w}^*$。将 Sigmoid 函数代入，我们得到：

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{1}{N}\sum_{\ell=1}^{N}\left[\frac{1}{1+e^{-\boldsymbol{x}^{(\ell)}\boldsymbol{w}}} - y^{(\ell)}\right]\boldsymbol{x}^{(\ell)T} = \boldsymbol{0}$$

- 这个方程是关于$\boldsymbol{w}$的一个非线性方程，无法直接求解这个方程，不能得到一个解析解

梯度下降

- 交叉熵损失的梯度

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{1}{N}\sum_{\ell=1}^{N}\left[\sigma(\boldsymbol{x}^{(\ell)}\boldsymbol{w}) - y^{(\ell)}\right]\boldsymbol{x}^{(\ell)T}$$

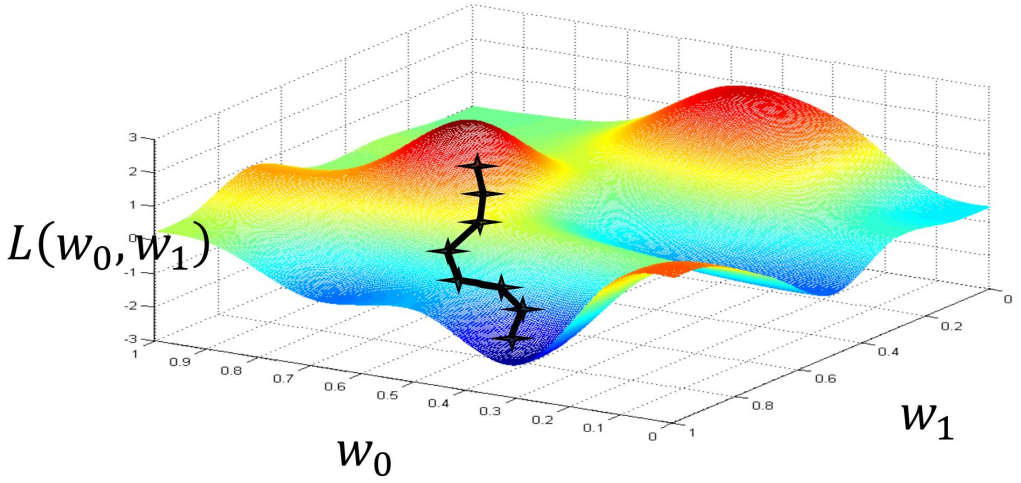
- 求解 $\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0$ 可以得到最优的 $\boldsymbol{w}^*$。但在这里，*解析解不存在*

- 因此，我们只能求助于数值方法
  - 梯度下降
  - 牛顿法
  - 坐标下降
  - ……

- 梯度下降法
- 从一个初始的 $\boldsymbol{w_0}$ 开始，沿着梯度的反方向迭代更新参数，以逐步逼近最小值点

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - r \cdot \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}$$

- r是学习率，控制每一步更新的幅度
- 收敛性： 由于交叉熵损失是凸函数，梯度下降法保证能收敛到最优值 $\boldsymbol{w}^*$



- 梯度的直观理解

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{1}{N}\sum_{\ell=1}^{N}\left[\underbrace{\sigma\left(\boldsymbol{x}^{(\ell)}\boldsymbol{w}\right) - y^{(\ell)}}_{\text{预测误差}}\right]\boldsymbol{x}^{(\ell)T}$$

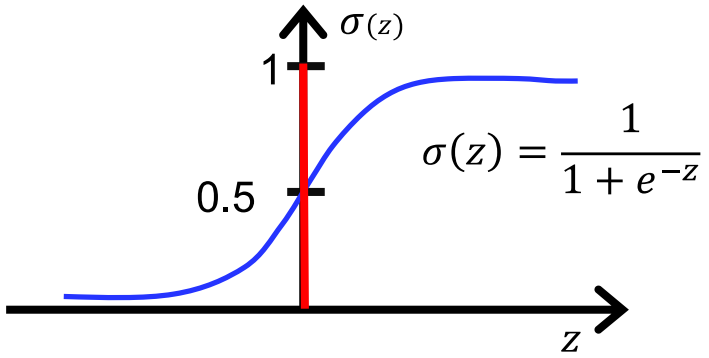
可以发现梯度下降法（GD）总是试图减小预测误差

- 如果 $y^{(\ell)} = 1$，算法会驱使 $\sigma\left(\boldsymbol{x}^{(\ell)}\boldsymbol{w}\right)$ 趋近于 1
- 如果 $y^{(\ell)} = 0$，算法会驱使 $\sigma\left(\boldsymbol{x}^{(\ell)}\boldsymbol{w}\right)$ 趋近于 0

决策边界

- 决策边界是特征空间中的一个“边界”，它将不同类别的点分隔开
- 样本按以下方式被分为 1 和 0 两类：

$$\hat{y} = \begin{cases} 1, & \text{if } \sigma(\boldsymbol{xw}) \geq 0.5 \\ 0, & \text{if } \sigma(\boldsymbol{xw}) < 0.5 \end{cases}$$

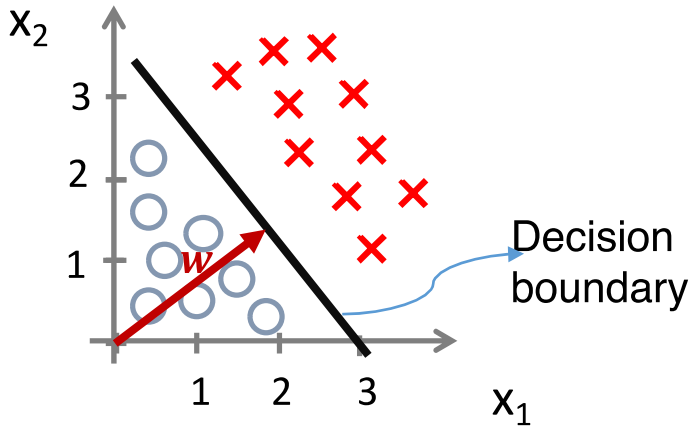


- 这等价于按以下方式对样本进行分类：

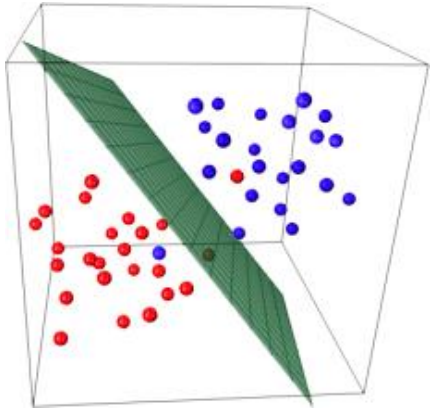
$$\hat{y} = \begin{cases} 1, & \text{if } \boldsymbol{xw} \geq 0 \\ 0, & \text{if } \boldsymbol{xw} < 0 \end{cases}$$

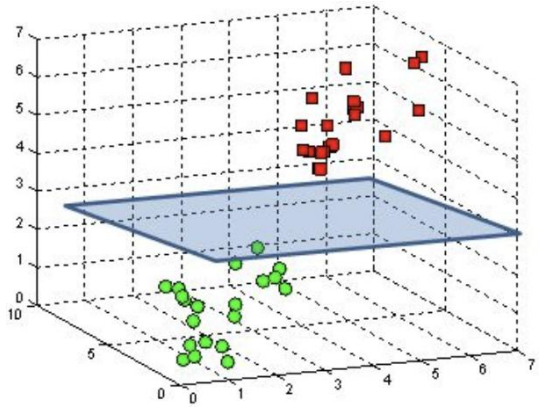
- 决策边界由满足 $\boldsymbol{xw} = 0$ 的 $\boldsymbol{x}$ 组成

- 由于 $\boldsymbol{w}$ 是一个向量，所有满足 $\boldsymbol{xw} = 0$ 的 $\boldsymbol{x}$ 构成一个与 $\boldsymbol{w}$ 正交的空间



- 在二维情况下，这个空间是一条直线
- 在三维情况下，这个空间是一个平面



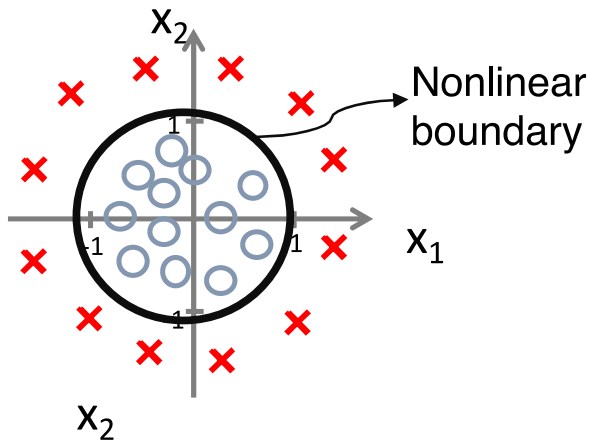


- 对于一个固定的向量 $\boldsymbol{w} \in \mathbb{R}^K$，点集

$$\boldsymbol{x} \in \{\boldsymbol{x} | \boldsymbol{x}\boldsymbol{w} = 0\}$$

构成一个 $(K-1)$ 维的超平面

- 超平面永远无法表示非线性的决策边界，例如：

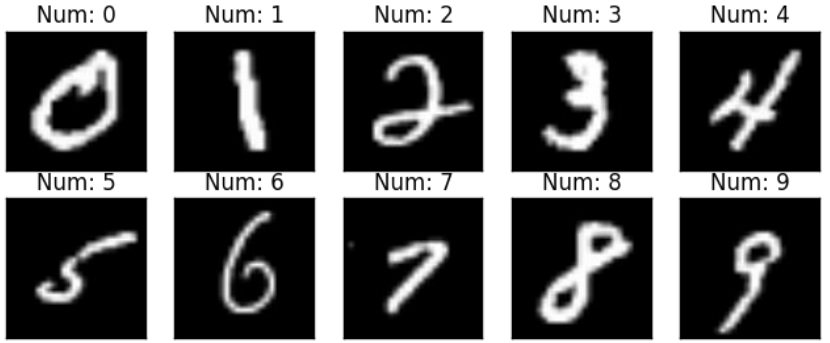


这就是为什么逻辑回归被称为线性分类器的原因

课程大纲

- 二分类
- 多分类

- 许多应用包含多于 2 个类别



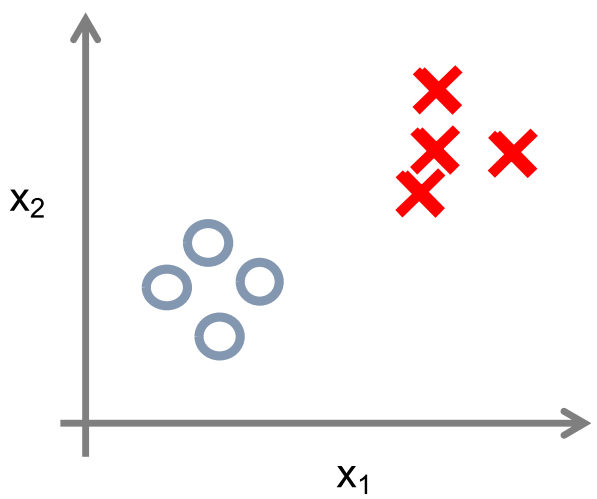
- 处理多类别分类的两种方法：
  - One-vs-All

    将多类别问题转化为多个二元分类问题
  - Softmax 函数

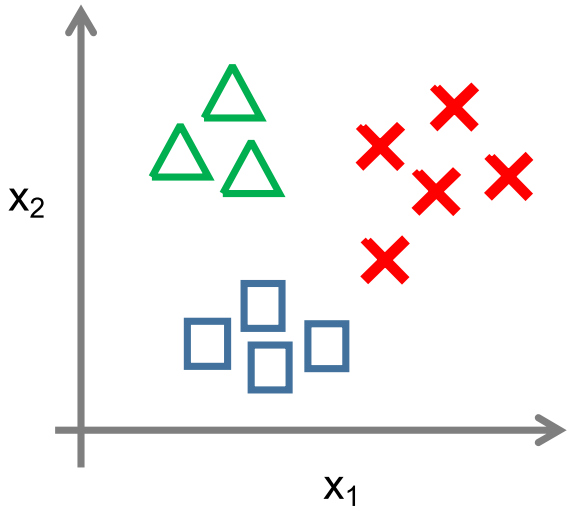
    直接将样本分到其中一个类别

One-vs-All

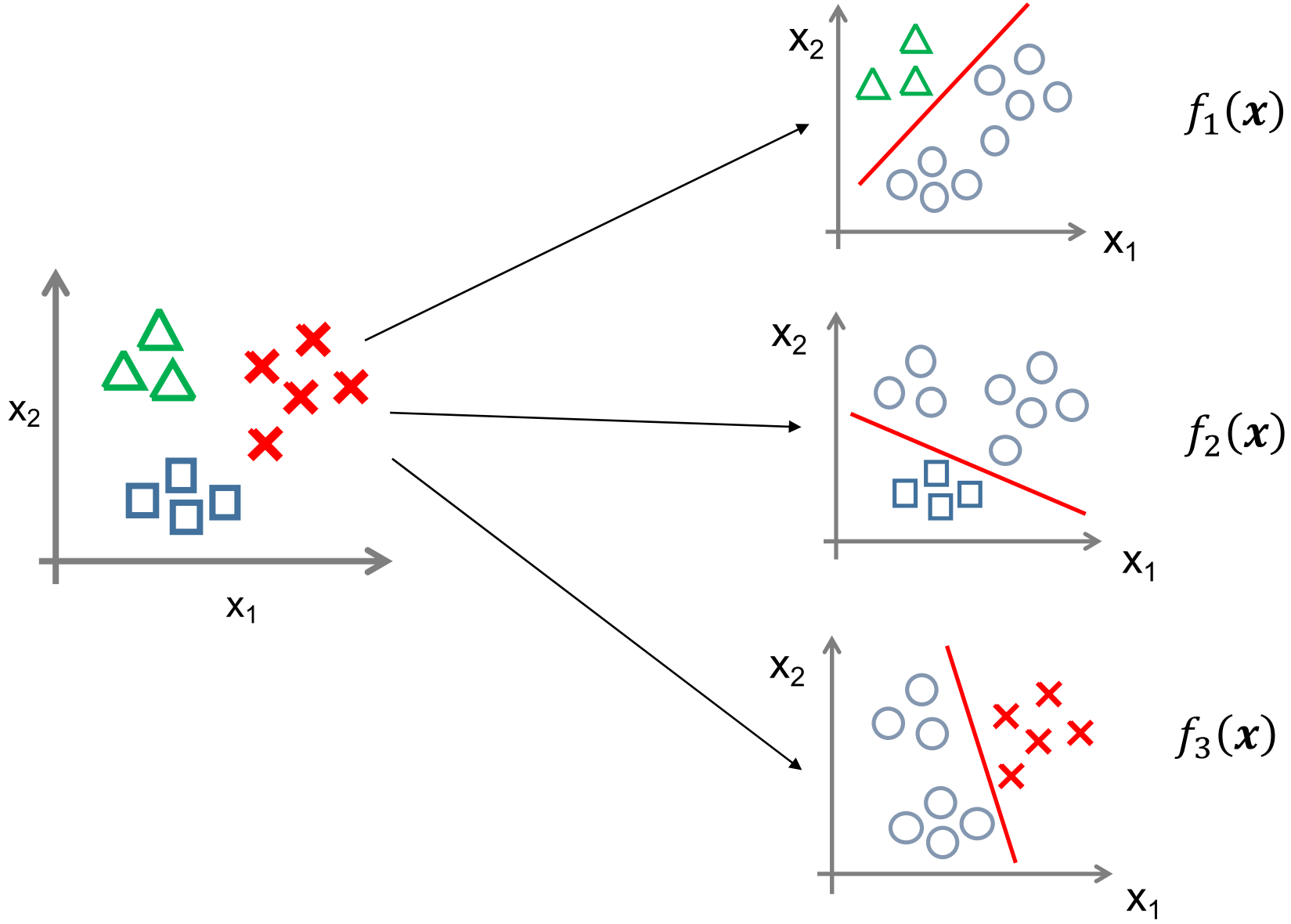
二分类



多分类

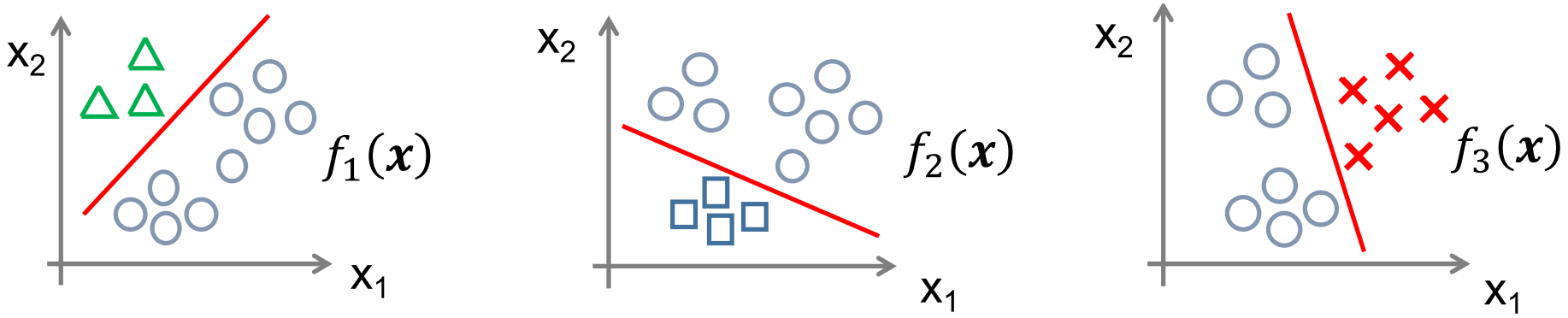


- 为每个类别训练一个分类器



- 为了预测新样本 $\boldsymbol{x}$ 的类别，选择满足以下条件的类别：

$$k = \arg\max_i f_i(\boldsymbol{x})$$

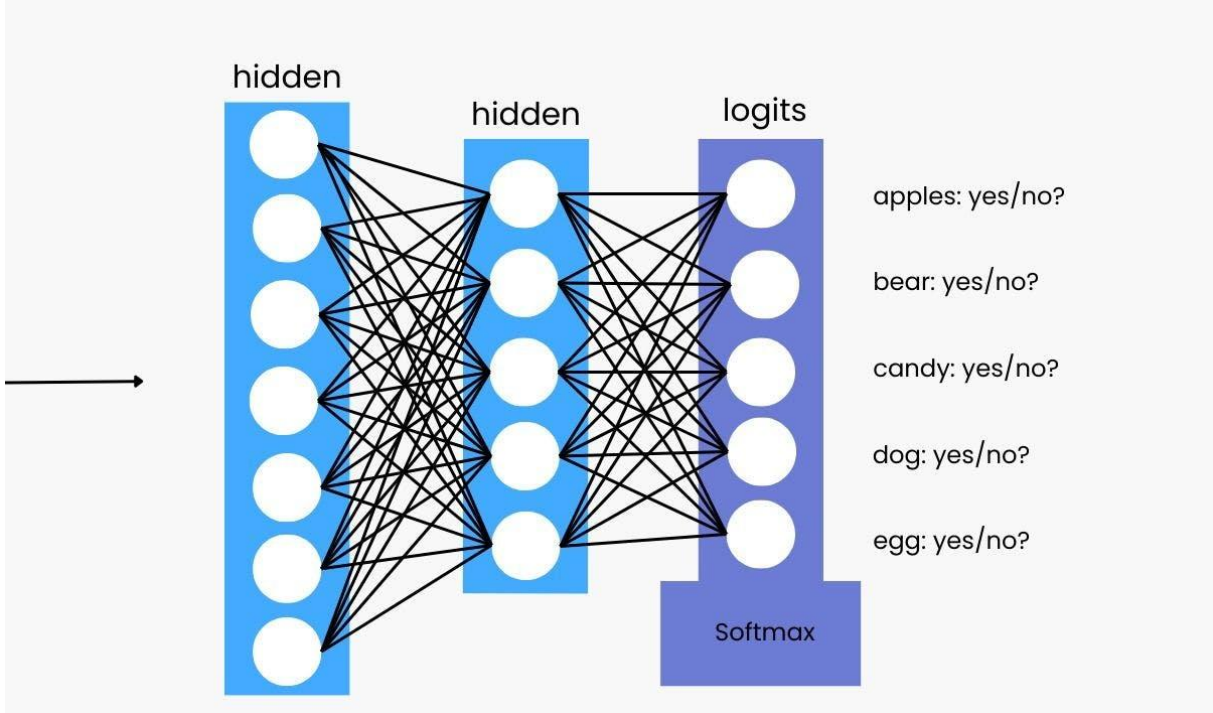


Softmax 函数

- Softmax 函数

$$softmax_i(\boldsymbol{z}) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

可以看出，$\sum_{i=1}^{K} softmax_i(\boldsymbol{z}) = 1$



$$softmax_i(\boldsymbol{z}) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

- 数据 $\boldsymbol{x}$ 被分类到第 i 个类别的概率是

$$f_i(\boldsymbol{x}) = softmax_i(\boldsymbol{xW}) = \frac{e^{\boldsymbol{x}\boldsymbol{w}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{x}\boldsymbol{w}_k}}$$

  其中 $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_K]$

- 如果 $\boldsymbol{x}$ 属于第 i 个类别，模型应该促使 $f_i(\boldsymbol{x})$ 尽可能地大

- 当 $K = 2$ 时，Softmax 函数等价于 logistic 函数
  - 在二分类情况下，我们有：

$$softmax_1(\boldsymbol{xW}) = \frac{e^{\boldsymbol{xw}_1}}{e^{\boldsymbol{xw}_1} + e^{\boldsymbol{xw}_2}} \qquad softmax_2(\boldsymbol{xW}) = \frac{e^{\boldsymbol{xw}_2}}{e^{\boldsymbol{xw}_1} + e^{\boldsymbol{xw}_2}}$$

$$= \frac{1}{1 + e^{-\boldsymbol{x}(\boldsymbol{w}_1 - \boldsymbol{w}_2)}} \qquad = \frac{e^{-\boldsymbol{x}(\boldsymbol{w}_1 - \boldsymbol{w}_2)}}{1 + e^{-\boldsymbol{x}(\boldsymbol{w}_1 - \boldsymbol{w}_2)}}$$

  - 可以看出：

$$softmax_1(\boldsymbol{xW}) = \sigma\big(\boldsymbol{x}(\boldsymbol{w}_1 - \boldsymbol{w}_2)\big)$$

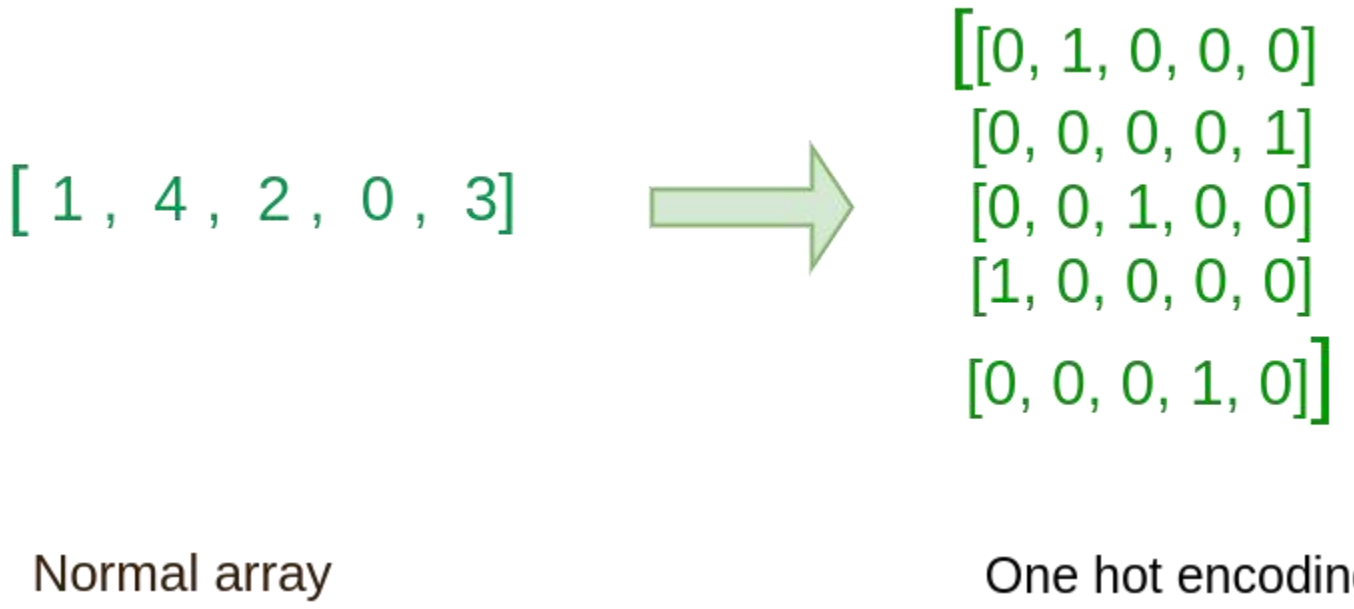
$$softmax_2(\boldsymbol{xW}) = 1 - \sigma\big(\boldsymbol{x}(\boldsymbol{w}_1 - \boldsymbol{w}_2)\big)$$

二分类 Softmax 分类等价于逻辑回归，其模型参数为 $\boldsymbol{w}_1 - \boldsymbol{w}_2$

代价函数

- 对于一个有 K 个类别的训练数据集，其标签 $\boldsymbol{y}$ 用一个独热向量（one-hot vector）

$$
\begin{bmatrix}1, 0, 0, \cdots, 0\end{bmatrix},\\
\begin{bmatrix}0, 1, 0, \cdots, 0\end{bmatrix},\\
\vdots\\
\begin{bmatrix}0, 0, 0, \cdots, 1\end{bmatrix}
$$



- 目标是最大化相应的概率 $f_i(\boldsymbol{x})$。因此，代价函数可以写为

$$L(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_K) = -\frac{1}{N}\sum_{\ell=1}^{N}\sum_{k=1}^{K} y_k^{(\ell)} \log\left[softmax_k(\boldsymbol{x}^{(\ell)}\boldsymbol{W})\right]$$

Cross-entropy 损失

  - $y_k^{(\ell)}$ 是 $\boldsymbol{y}^{(\ell)}$ 的第 k 个元素

梯度下降

- 关于 $\boldsymbol{w}_j$ 的梯度是

$$\frac{\partial L(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_K)}{\partial \boldsymbol{w}_j} = \frac{1}{N} \sum_{\ell=1}^{N} \left(\underbrace{softmax_j(\boldsymbol{x}^{(\ell)} \boldsymbol{W}) - y_j^{(\ell)}}_{\text{预测误差}} \right) \boldsymbol{x}^{(\ell)T}$$

注意，对于 $j = 1, \cdots, K$，所有的 $\boldsymbol{w}_j$ 都应该同时更新

$$\frac{\partial L(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_K)}{\partial \boldsymbol{w}_j} = \frac{1}{N}\sum_{\ell=1}^{N}\left(\underbrace{softmax_j(\boldsymbol{x}^{(\ell)}\boldsymbol{W}) - y_j^{(\ell)}}_{\text{预测误差}}\right)\boldsymbol{x}^{(\ell)T}$$

- 将 $\boldsymbol{W}$ 表示为 $[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_K]$，我们有：

$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{W}} = \frac{1}{N}\sum_{\ell=1}^{N}\boldsymbol{x}^{(\ell)T}\left(softmax(\boldsymbol{x}^{(\ell)}\boldsymbol{W}) - \boldsymbol{y}^{(\ell)}\right)$$

  - $softmax(\boldsymbol{x}^{(\ell)}\boldsymbol{W}) = [softmax_1(\boldsymbol{x}^{(\ell)}\boldsymbol{W}), \cdots, softmax_K(\boldsymbol{x}^{(\ell)}\boldsymbol{W})]$ 是一个行向量

- 更新：$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - r \cdot \left.\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{W}}\right|_{\boldsymbol{W}=\boldsymbol{W}_t}$