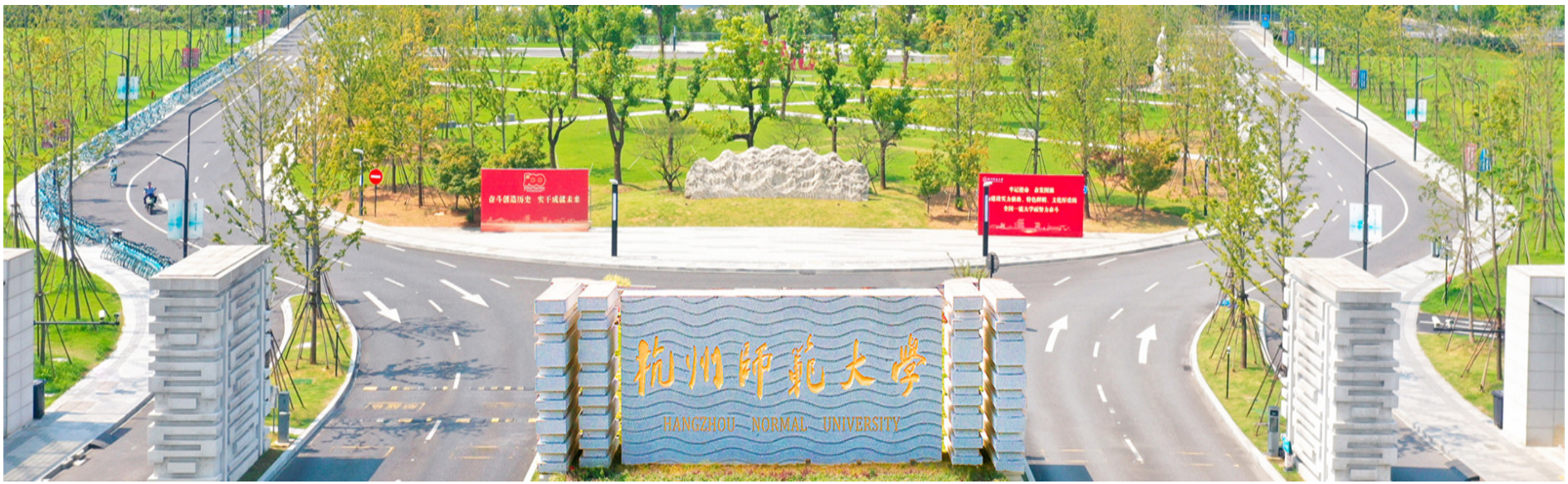
第五讲 - NLP基础算法：逻辑回归

张建章

阿里巴巴商学院
杭州师范大学

2025-02-01



目录

1. 生成模型与判别模型的对比

朴素贝叶斯是一种生成模型，它通过建模类别如何生成输入数据(即计算似然$P(d|c)$与先验$P(c)$)，来间接推导后验概率$P(c|d)$；而逻辑回归则是一种判别模型，直接学习输入与类别之间的映射，即，直接估计$P(c|d)$。

直观比喻说明：生成模型会尝试了解“狗长什么样”和“猫长什么样”，而判别模型只关心“如何区分狗和猫”。这种区别使得逻辑回归能更直接地关注那些对区分不同类别最有用的特征，而不必刻意去理解每个类别的完整分布。

2. 逻辑回归的特点

逻辑回归是一种用于**发现输入特征与某个特定结果之间联系的监督学习算法**，可以用于将观察实例划分为两个类别，也可以扩展为多类别情形，即，**多项逻辑回归 (softmax回归)**。

它不仅是社会科学和自然科学中极为重要的分析工具，也是NLP领域中**最常用的分类基线模型之一**。此外，逻辑回归与神经网络有密切关系，神经网络可以看作是由多个逻辑回归分类器逐层堆叠而成的。

3. 概率分类器的组成要素

构建一个概率分类器所需的四个核心组件 (以逻辑回归为例)：

① **特征表示**：将每个输入实例 $x^{(i)}$ 表示为一个特征向量，例如 $[x_1, x_2, \ldots, x_n]$；

② **分类函数**：一个通过计算 $p(y \mid x)$ 来产生估计类别 $\hat{y}$ 的函数，如，用于二分类的Sigmoid函数以及多分类中的Softmax函数；

③ **目标函数**：通常以损失函数形式定义，用来衡量模型输出与真实标签之间的差异，进而作为模型学习的优化目标，如，交叉熵损失函数；

④ **优化算法**：用于最小化目标函数，教材介绍了随机梯度下降（SGD）等算法作为参数更新的方法。

训练与测试阶段：训练阶段旨在通过优化目标函数来学习分类函数中的参数，而测试阶段则利用学到的参数对新的输入实例进行类别概率的计算，并依据概率大小作出最终分类决策。

目录

二元逻辑回归的核心思想：通过对输入特征进行线性加权求和得到一个实数得分 z，再利用Sigmoid函数将 z 映射为一个介于0和1之间的数值，从而实现概率估计，并最终用于二元分类决策。Sigmoid函数是将线性模型输出转化为概率的关键步骤。

1. 二元逻辑回归分类的背景

对每个输入实例 x（通常以特征向量 $[x_1, x_2, \ldots, x_n]$ 表示）进行二分类决策，输出 y 的取值为 1（表示正类）或 0（表示负类）。模型的目标是估计 $P(y = 1 \mid x)$，即**给定输入 x 属于正类的概率**。

2. 线性组合与得分（Logit）的计算

为了做出分类决策，模型首先为每个特征分配一个权重 w_i（权重可以为正也可以为负，反映了该特征对分类决策的正向或负向影响），并设有一个偏置项 b。将所有特征与对应权重进行线性组合，再加上偏置项，得到一个实数值得分 z，即

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

这里的 z 称为logit，它反映了**输入 x 对正类的“证据总量”**，但 z 本身可能取任意实数值，不能直接作为概率解释。

3. Sigmoid函数及其作用

为了将得分 z 映射到概率区间 $(0,1)$ 内，逻辑回归采用了Sigmoid（或称Logistic）函数，其定义为

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

Sigmoid函数具有以下3个重要性质：

- **区间映射**：它能将任意实数 z 映射到区间 $(0,1)$，正好符合概率的定义；

- **非线性特性**：在 z 附近近似线性，而当 z 远离 0 时迅速饱和于0或1，这有助于抑制极端得分的影响；

- **可微性**：Sigmoid函数是光滑且处处可导的，为后续的梯度下降优化提供了数学基础。

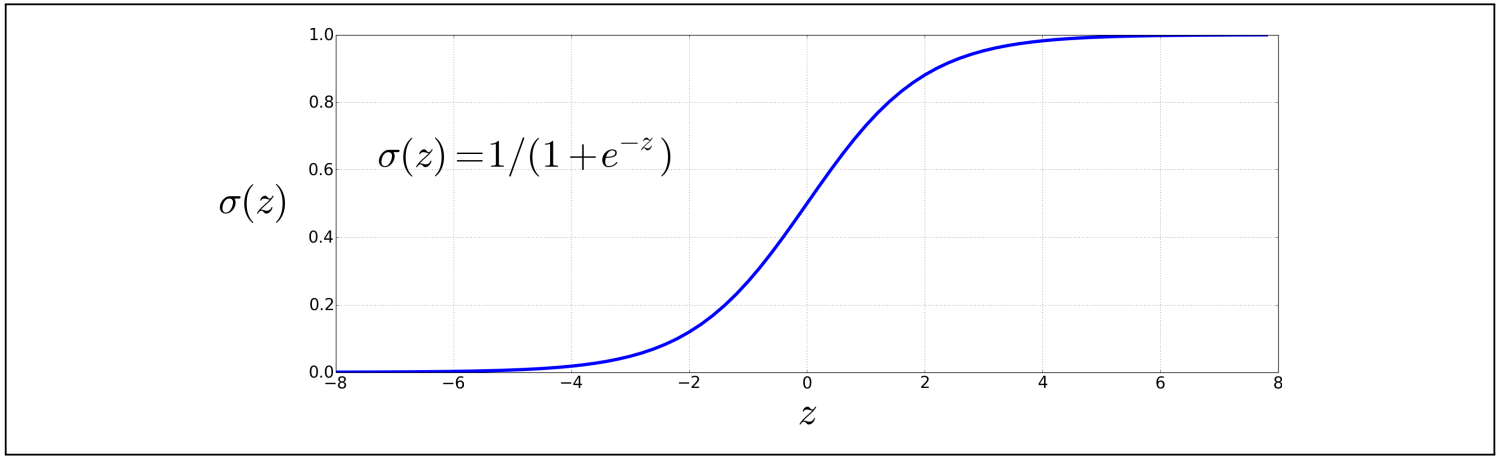


Figure 5.1 The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $(0, 1)$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

4. 概率输出与对偶关系

应用Sigmoid函数后，逻辑回归模型将得分 z 转换为正类概率：

$$P(y = 1 \mid x) = \sigma(w \cdot x + b)$$

由于 $1 - \sigma(z) = \sigma(-z)$，负类的概率可以表示为：

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

在逻辑回归中，线性组合 $z = w \cdot x + b$被称为 logit，它实际上代表了正类概率 p 的对数几率，即

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

而 Sigmoid 函数正好是这个 logit 函数的逆函数，它将任意实数 z 映射回一个介于 0 和 1 之间的概率值，即

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

因此，称 z 为 logit 意味着**可以将 z 看作是由概率通过 logit 转换得到的数值，再通过 Sigmoid 函数将其逆转换回来得到原始的概率。**

目录

利用逻辑回归模型进行分类决策

1. 利用Sigmoid函数进行概率估计

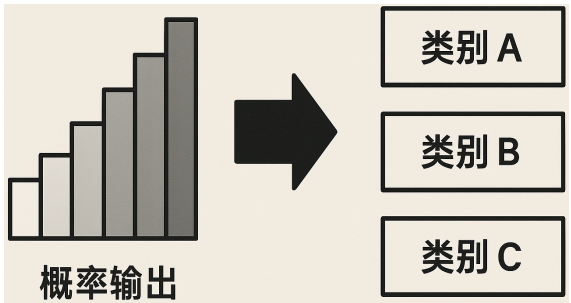
- 给定一个输入实例 x（通常表示为特征向量 $[x_1, x_2, \ldots, x_n]$），通过将输入与模型参数（权重 w 和偏置 b）线性组合得到得分 $z = w \cdot x + b$；

- 将得分 z 通过Sigmoid函数 $\sigma(z) = \frac{1}{1+\exp(-z)}$ 转化为一个介于 0 和 1 之间的数值，从而估计正类概率 $P(y = 1 \mid x) = \sigma(w \cdot x + b)$；

2. 分类决策规则

- 决策边界设定为 0.5：如果 $P(y = 1 \mid x) > 0.5$ 则判定输入 x 属于正类（例如正面情感），否则判定为负类（例如负面情感）；

- 这种基于概率阈值的决策机制使得逻辑回归能够直接将连续的概率输出转换为离散的类别标签。



情感分类应用示例

以电影评论情感分类为例，提取文档的多个特征（如正向词计数、负向词计数、否定词标识、代词计数、感叹号出现与否、文档长度等）构成特征向量。

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon words $\in$ doc)	3
x_2	count(negative lexicon words $\in$ 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(word count of doc)	$\ln(66) = 4.19$

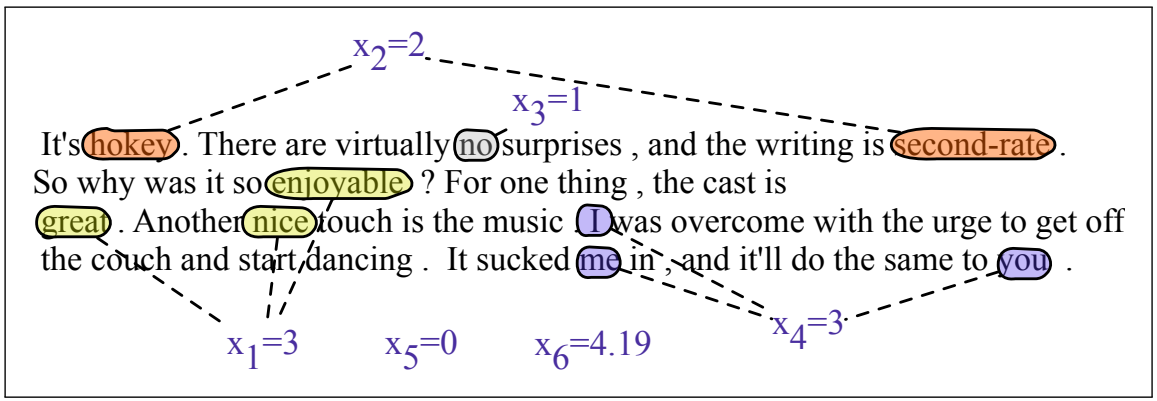


Figure 5.2 A sample mini test document showing the extracted features in the vector x.

假设已经通过训练学得权重向量$[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$）和偏置 $b = 0.1$，模型计算得到 $z = w \cdot x + b$ 后，通过Sigmoid函数转化为 $P(y = 1 \mid x) = 0.70$；因此，该电影评论被判定为正面情感。

$$
\begin{aligned}
P(y = 1 \mid x) &= \sigma(w \cdot x + b) \\
&= \sigma\Big([2.5,\ -5.0,\ -1.2,\ 0.5,\ 2.0,\ 0.7] \cdot [3,\ 2,\ 1,\ 3,\ 0,\ 4.19] + 0.1\Big) \\
&= \sigma(0.833) \\
&= 0.70 \\
P(y = 0 \mid x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.30
\end{aligned}
$$

特征值标准化 (standardization)

1. 为何需要对输入特征进行缩放

不同特征可能**取值范围差异很大**，这会导致在模型训练过程中，各特征对损失函数梯度的贡献不均，从而影响参数更新的效率和最终模型的收敛速度。当特征值处于可比较的数值范围内时，模型更容易平衡各个特征的影响，进而加快梯度下降的过程，提高学习效率。

2. z-score标准化方法

将每个特征的数据转化为均值为0、标准差为1的分布，使得各特征具有相似的尺度，具体操作是：

首先，计算特征 x_i 在所有 m 个样本中的均值 $\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_i^{(j)}$ 和标准差 $\sigma_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(x_i^{(j)} - \mu_i)^2}$；然后，用公式

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}$$

将原始特征 x_i 转换为标准化后的特征 x_i'。

3. 归一化 (normalization)方法

将特征值缩放到固定区间内（通常是0到1），这种方法有助于在模型中直接比较不同特征的数值。归一化操作的公式为：

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

其中，$\min(x_i)$ 和 $\max(x_i)$ 分别表示特征 x_i 在数据集中出现的最小值和最大值。

4. 其他任务与特征设计

- 除了情感分类，逻辑回归还可以应用于其他文本分类任务（如句末标点的歧义消解等），在这些任务中，任何能够反映输入信息的属性都可作为特征；

- 特征既可以由**人工设计**，也可以**通过表示学习自动获得**；此外，还可以**考虑特征之间的交互作用**，以提高分类性能。

5. 批量处理与向量化实现

为了提高计算效率，逻辑回归通常采用矩阵运算对多个测试实例进行并行处理。将所有测试实例的特征向量构成一个矩阵 X（每行为一个实例），利用矩阵乘法 $Xw + b$ 计算所有实例的得分，再对每个得分应用Sigmoid函数，快速获得所有实例的预测概率。这种向量化实现充分利用了现代计算硬件的并行处理能力，显著加快了模型的预测速度。

6. 分类器选择的讨论

- 相较于生成模型（如朴素贝叶斯），逻辑回归作为判别模型能更直接地关注对类别判定有帮助的特征，尤其在处理多个相关特征时更稳健。

- 尽管朴素贝叶斯在某些小规模数据或短文本任务上可能表现不错，但在大规模数据集或长文本中，逻辑回归通常能提供更精确的概率估计和更好的分类性能。

目录

1. 多类别问题背景与模型表述

- 当分类任务中类别数大于2时（例如新闻多分类），传统的二分类逻辑回归不足以直接处理，此时采用多类别 (多项式)逻辑回归模型。

- 在多类别逻辑回归中，每个输入实例 x 被映射为一个长度为 K 的输出向量，其中 K 表示类别数。对于正确类别 c，将其输出设为1，而其他类别均为0，这种表示方式称为 one-hot 编码。

2. Softmax函数的引入

- 多类别逻辑回归使用softmax函数来将模型输出（得分向量）转化为概率分布。softmax函数是sigmoid函数的推广，能够接受一个 K 维的任意实数向量 $z = [z_1, z_2, \ldots, z_K]$，并将其映射到一个所有元素非负、总和为1的概率向量：

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \quad (1 \leq i \leq K)$$

这一公式确保了当某个 z_i 显著大于其他分量时，其对应的概率接近1，而其他分量的概率则接近0。

3. softmax函数在逻辑回归中的应用

- 在多项逻辑回归中，对于每个类别 k，模型为输入 x 分别学习一个权重向量 w_k 以及偏置 b_k。对于每个类别，其得分为 $w_k \cdot x + b_k$；

- 利用softmax函数，类别 k 的预测概率被计算为：

$$p(y_k = 1 \mid x) = \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

为了充分利用现代向量处理硬件，多个类别对应的权重向量 $\{w_1, w_2, \ldots, w_K\}$ 被组合成一个权重矩阵 W（其每一行对应一个 w_k），偏置项则构成一个向量 b。于是，所有类别的预测概率可以通过一条矩阵运算表达为：

$$\hat{y} = \text{softmax}(Wx + b)$$

上式实现了对所有 K 个类别概率的并行计算。

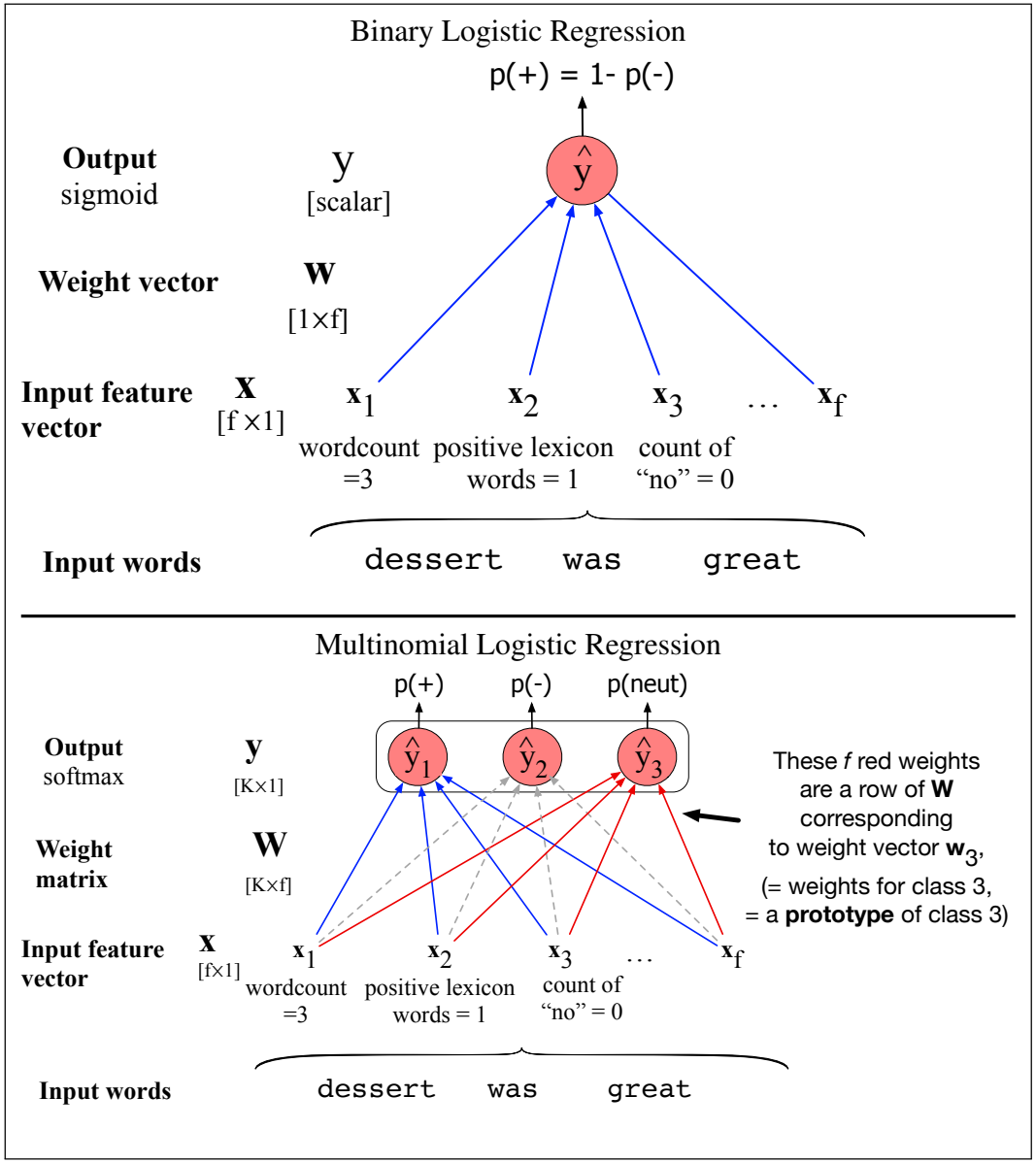


Figure 5.3 Binary versus multinomial logistic regression. Binary logistic regression uses a single weight vector $\mathbf{w}$, and has a scalar output $\hat{y}$. In multinomial logistic regression we have K separate weight vectors corresponding to the K classes, all packed into a single weight matrix $\mathbf{W}$, and a vector output $\hat{\mathbf{y}}$. We omit the biases from both figures for clarity.

4. 多元逻辑回归模型的解释性

- 一个有趣的解释是，将 W 的每一行 w_k 看作是类别 k 的原型或模板。**输入向量与 w_k 的点积衡量了输入与类别 k 的相似度，因而逻辑回归实际上是在寻找与每个类别原型最相似的输入实例；**

- 此外，多项逻辑回归中，每个特征在不同类别下可以拥有不同的权重，这使得同一特征能够为不同类别提供正向或负向的证据，从而更细粒度地区分各类别。

目录

参数学习的过程：利用交叉熵损失函数作为目标指标，将逻辑回归参数学习问题转化为一个凸优化问题，并通过梯度下降（特别是随机梯度下降）算法不断更新参数，即权重向量 w 和偏置 b，使得模型在训练数据上的预测概率尽可能接近真实标签，从而实现有效的分类。

1. 参数学习目标

逻辑回归是一种监督学习方法，每个训练样本 $(x^{(i)}, y^{(i)})$ 已经带有正确标签 $y^{(i)}$（对于二元分类，y 为0或1）。模型根据输入 x 通过计算 $z = w \cdot x + b$ 并应用 Sigmoid 函数，输出预测概率 $\hat{y} = \sigma(w \cdot x + b)$；**学习目标就是使得 $\hat{y}$ 尽可能接近真实标签 y。**

2. 损失函数的引入

为了衡量模型输出 $\hat{y}$ 与真实标签 y 之间的差距，引入损失函数（cost function）。逻辑回归采用了交叉熵损失 (cross-entropy loss)函数，对于单个样本，其形式为：

$$L_{CE}(\hat{y}, y) = -\left[y \log \hat{y} + (1-y) \log(1-\hat{y})\right]$$

这一损失函数的设计鼓励模型为正确标签赋予高概率，而对错误标签赋予低概率。

3. 优化问题的形式化

逻辑回归模型的学习问题可以形式化为一个凸优化问题，其目标是找到参数 $\theta = \{w, b\}$ 使得所有训练样本上的平均交叉熵损失最小：

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

由于交叉熵损失函数在逻辑回归中是凸的，这意味着不存在局部最小值，任何初始点通过适当的梯度下降都能达到全局最优解。

4. 梯度下降法及其变种

为了最小化损失函数，通常使用梯度下降算法，利用损失函数关于参数的梯度（即偏导数）来更新参数：

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L_{CE}\big(f(x^{(i)}; \theta), y^{(i)}\big)$$

其中 η 是学习率，决定了每次更新的步长大小。

实践中，为了提高计算效率，经常使用随机梯度下降算法，即每次只利用单个或一小批训练样本来近似计算梯度，从而实现在线式参数更新，这在大规模数据训练中尤为常用。

目录

1. 学习目标与参数定义

逻辑回归的学习 (训练)目标就是使得 $\hat{y}$ 尽可能接近真实标签 y，即，对于任一观测 x，**分类器的输出** $\hat{y} = \sigma(w \cdot x + b)$ **(取值为0-1之间的概率值)** 与**正确输出** y（**取值为0或1**）之间的差距：

$$L(\hat{y}, y) = \text{预测值与真实值之间的差异}$$

参数为Sigmoid函数中的w和b。

2. 损失函数的引入

为了实现上述目标，采用一种**使得训练样本中正确类别标签的概率更高**的损失函数。这种方法被称为**条件最大似然估计**，即选择参数 w 和 b 来最大化在给定观测 x 条件下训练数据中真实 y 标签的对数概率。最终得到的损失函数为**负对数似然损失**，通常称为**交叉熵损失**。

交叉熵损失函数推导

1. 概率模型的构建

针对单个观测 x，目标是学习出一组权重，最大化**模型对单个观测 x 生成正确标签的概率** $p(y \mid x)$，

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

当 $y = 1$ 时，$p(y \mid x) = \hat{y}$；当 $y = 0$ 时，$p(y \mid x) = 1 - \hat{y}$。

2. 对数似然的计算

为了使数学处理更简洁，同时将乘法转换为加法，对概率 $p(y \mid x)$ 取自然对数，得到对数似然：

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

对数函数保留了最大化概率的性质，即最大化概率的 $p(y \mid x)$ 等价于最大化其对数。

3. 构造损失函数：负对数似然

在机器学习中，通常是通过最小化一个损失函数来训练模型。为此，将对数似然取负，构造出损失函数：

$$L_{CE}(\hat{y}, y) = -\left[y \log \hat{y} + (1-y) \log(1-\hat{y})\right]$$

上式称为交叉熵损失函数。损失值越低，说明模型输出的概率分布与真实标签越接近。

4. 将模型输出与损失函数联系起来

在逻辑回归中，模型通过计算线性组合 $w \cdot x + b$ 并将其传入Sigmoid函数得到预测概率 $\hat{y} = \sigma(w \cdot x + b)$。将这一结果代入交叉熵损失函数，得到：

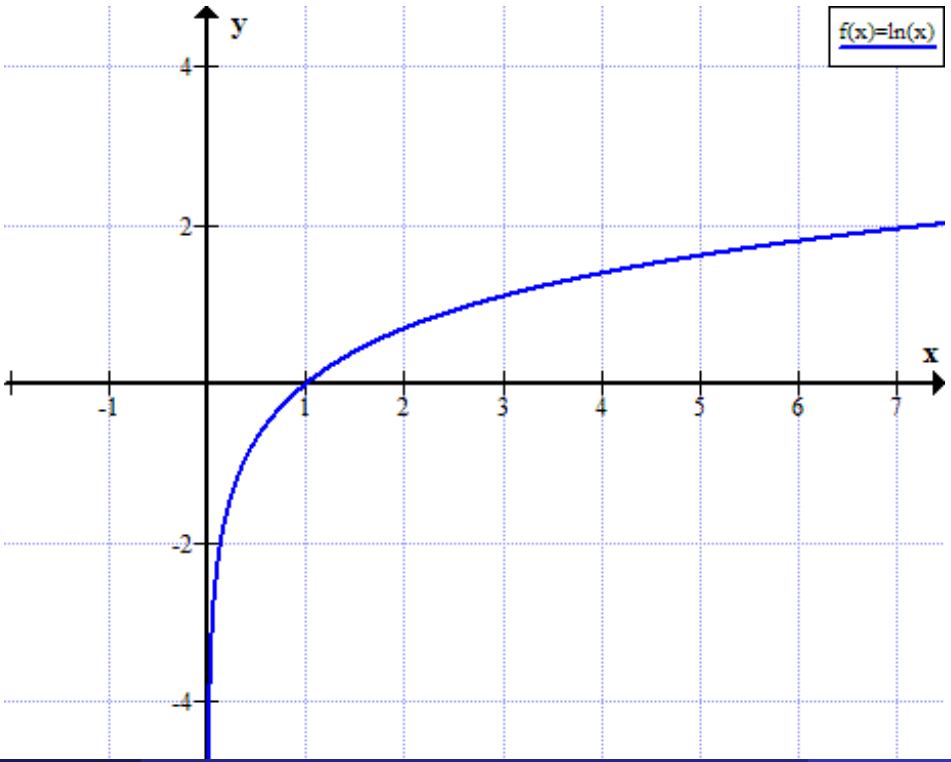
$$L_{CE}(\hat{y}, y) = -\left[y \log \sigma(w \cdot x + b) + (1-y) \log\left(1 - \sigma(w \cdot x + b)\right)\right]$$

通过最小化这个损失函数，模型会调整参数 w 和 b，使得对于每个训练样本，**预测概率** $\sigma(w \cdot x + b)$ **尽可能接近真实标签** y。

5. 交叉熵损失的意义

- 交叉熵损失函数不仅衡量了模型输出与真实分布之间的“距离”，对数似然衡量模型输出与真实分布之间的“相似度”；

- 交叉熵损失函数具有较好的数学性质——当预测概率接近真实标签（即为1或0）时，损失趋近于0；当预测概率与真实标签相差较大时，损失会迅速增大 (如下图所示)。这一特性促使模型在训练过程中不断提升对正确标签的置信度，从而提高分类性能。



目录

1. 目标与背景

- 在逻辑回归中，目标是通过调整参数 $\theta = \{w, b\}$ 使得模型对训练数据的预测概率 $\hat{y} = \sigma(w \cdot x + b)$ 尽可能接近真实标签 y；

- 损失函数（交叉熵损失）刻画了模型输出与真实标签之间的差距，而梯度下降法则用于在参数空间中寻找使得**平均损失最小**的参数集，即：

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L_{CE}\big(f(x^{(i)}; \theta), y^{(i)}\big).$$

其中，m为测试集中的样本数量。

- **逻辑回归的交叉熵损失函数是凸函数** (在函数图像上任意两点连线都不会低于函数图像本身)，凸函数最多只有一个全局最小值，也就是说不存在多个局部最小值使得优化过程陷入困境。因此，**从任意初始点出发使用梯度下降法都能保证找到全局最小值**。多层神经网络的损失函数通常是非凸的，具有多个局部最小值。在这种情况下，梯度下降可能会在某个局部最小值停滞，无法达到全局最优解，这也是深度学习模型训练中常见的挑战之一。

2. 梯度下降的基本原理

- **梯度的概念：**多变量函数在某一点的梯度是所有偏导数组成的向量，指向该函数**上升最快的方向**，为了使损失函数值下降，需要**向反方向更新参数**，即向梯度的相反方向移动。

- **直观解释：**类似于在峡谷中寻找下坡路径，梯度给出了上升最快的方向，因此我们选择沿着其反方向更新参数，从而快速降低损失。

- **更新规则：**考虑一个由参数 θ（如权重 w 和偏置 b）构成的损失函数L，为了使损失函数下降，梯度下降法按照下面的规则更新参数：

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

其中 η 是学习率，控制每次更新的步长大小。**逻辑回归交叉熵损失函数对单个权重 w_j 的偏导数为：**

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

因此，预测误差与输入特征值的乘积决定了参数更新的方向和幅度。

单变量损失函数梯度下降示例

在单变量情况下，梯度可以理解为损失函数在该点的斜率。

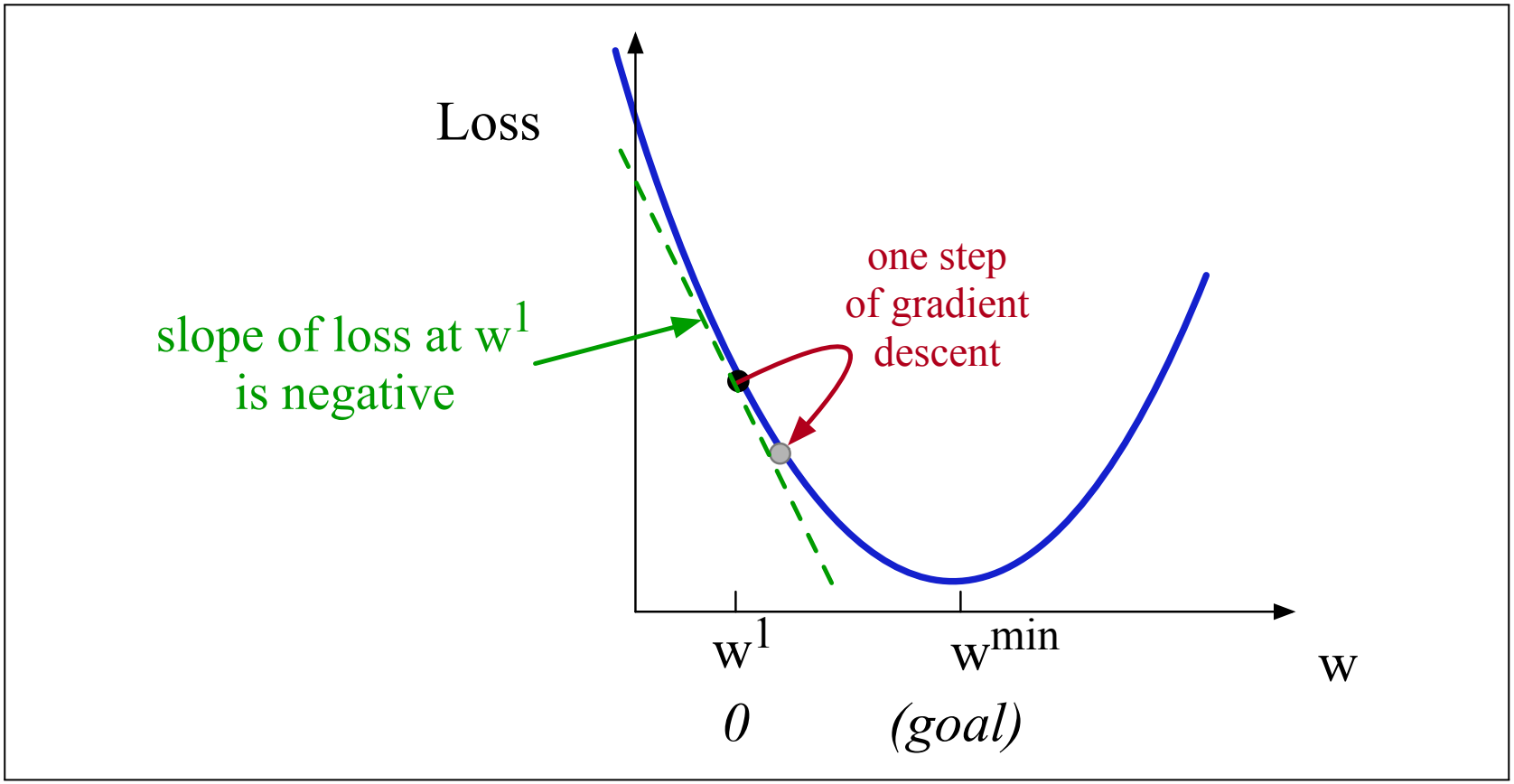


Figure 5.4 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 the value at the second step, and so on.

多变量损失函数梯度下降示例

对于多维参数，每个参数都有对应的偏导数，其集合构成梯度向量。

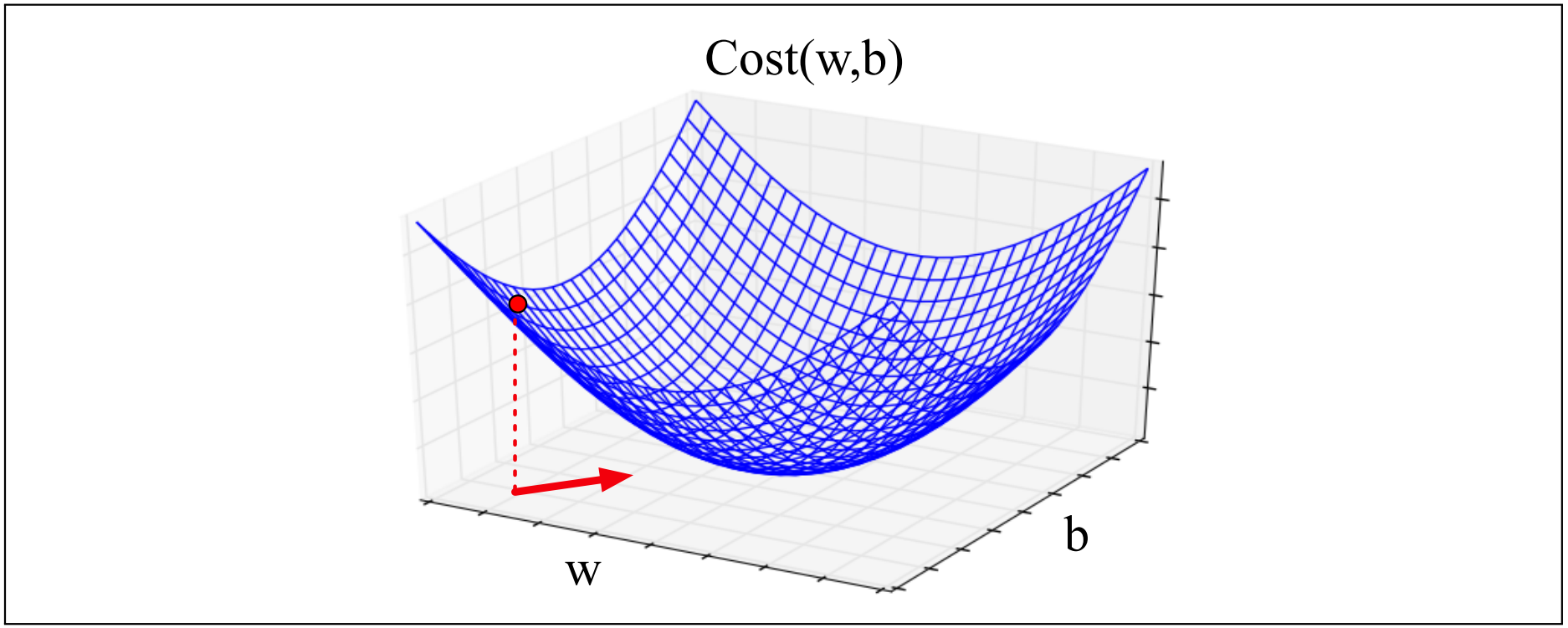


Figure 5.5 Visualization of the gradient vector at the red point in two dimensions w and b, showing a red arrow in the x-y plane pointing in the direction we will go to look for the minimum: the opposite direction of the gradient (recall that the gradient points in the direction of increase not decrease).

梯度下降的实际实现与优化策略

- **随机梯度下降（SGD）**：为了提高大规模数据训练的效率，梯度下降通常采用随机梯度下降，即每次仅利用一个随机训练样本来计算梯度，并据此更新参数；

- **Mini-batch训练**：为了平衡梯度估计的噪声和计算效率，还可以采用 mini-batch 训练方法，每次利用一组小批量样本进行参数更新，从而获得更稳定的梯度估计并加速并行计算；

- **学习率调整**：选择合适的学习率 η 至关重要，过高可能导致参数更新过快而越过最优点，过低则会使收敛速度缓慢。常见策略是初始采用较高学习率，随后逐渐降低（例如，使学习率成为训练迭代次数的函数）。

随机梯度下降 (Stochastic Gradient Descent)

随机梯度下降之所以被称为“随机”，是因为它在每一步迭代中会随机选取单个样本进行计算。

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

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) $y^{(1)}$, $y^{(2)}$,..., $y^{(m)}$

$\theta \leftarrow 0$ # (or small random values)
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 θ to maximize loss?
3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead
return θ

Figure 5.6 The stochastic gradient descent algorithm. Step 1 (computing the loss) is used mainly to report how well we are doing on the current tuple; we don't need to compute the loss in order to compute the gradient. The algorithm can terminate when it converges (when the gradient norm $< \epsilon$), or when progress halts (for example when the loss starts going up on a held-out set). Weights are initialized to 0 for logistic regression, but to small random values for neural networks, as we'll see in Chapter 7.

随机梯度下降 (Stochastic Gradient Descent)

随机梯度下降针对单个样本进行参数优化，可能会导致参数非常不稳定的移动，因此通常计算批量训练实例而不是单个实例的梯度。

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

```
    # 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)

θ ← 0          # (or small random values)
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 ŷ^(i) = f(x^(i); θ)        # What is our estimated output ŷ?
       Compute the loss L(ŷ^(i), y^(i))   # How far off is ŷ^(i) from the true output y^(i)?
    2. g ← ∇_θ L(f(x^(i); θ), y^(i))      # How should we move θ to maximize loss?
    3. θ ← θ − η g                        # Go the other way instead
return θ
```

Figure 5.6 The stochastic gradient descent algorithm. Step 1 (computing the loss) is used mainly to report how well we are doing on the current tuple; we don't need to compute the loss in order to compute the gradient. The algorithm can terminate when it converges (when the gradient norm $< \epsilon$), or when progress halts (for example when the loss starts going up on a held-out set). Weights are initialized to 0 for logistic regression, but to small random values for neural networks, as we'll see in Chapter 7.

小批量训练 (Mini-batch training)

1. 背景与动机

- **随机梯度下降 (SGD)的问题：**随机梯度下降每次仅选取单个训练样本来计算梯度并更新参数，这种“在线”更新方法虽然简单，但由于每次仅依赖单个样本的信息，导致参数更新过程可能“颠簸”较大，梯度估计不稳定。

- **全批量训练 (batch training)的局限：**与之相对，全批量梯度下降利用整个训练集计算梯度，能获得准确的梯度方向，但其计算成本高昂，尤其在面对大规模数据集时效率较低。

2. 小批量训练的基本思想

- **折衷方案：**为平衡梯度估计的准确性和计算效率，引入小批量训练的策略。该方法在每次迭代中不是只使用一个样本，也不是使用全部样本，而是采用一部分样本组成的小批量来计算梯度。

- **优点：**① 通过同时处理多个样本，可以利用向量化操作大幅提升计算效率；② 小批量的梯度平均可以降低单个样本带来的噪声，使得梯度更新更加稳定，从而加速收敛。

3. 小批量损失函数

假设一个小批量包含 m 个样本，每个样本的输入与标签分别记作 $x^{(i)}$ 和 $y^{(i)}$，并假定这些样本相互独立。单个样本的交叉熵损失函数为

$$L_{\mathsf{CE}}(\hat{y}, y) = -\left[y \log \hat{y} + (1-y) \log(1-\hat{y})\right]$$

则整个小批量的对数概率为

$$\log p(\text{training labels}) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)})$$

这可以表示为小批量损失的负和，即

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

为了得到平均损失，定义小批量的代价函数为

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

4. 小批量梯度的计算

每个样本的梯度为 $(\hat{y}-y)x$。因此，小批量梯度就是各样本梯度的平均值：

$$\frac{\partial \text{Cost}(\hat{y},y)}{\partial w_j}=\frac{1}{m}\sum_{i=1}^{m}\left[\sigma(w\cdot x^{(i)}+b)-y^{(i)}\right]x_j^{(i)}.$$

同时，这一求和形式可以利用矩阵运算进一步向量化表达，提高计算效率。设 X 为形状为 $[m\times f]$ 的输入矩阵，y 为形状为 $[m\times 1]$ 的标签向量，则梯度可写为：

$$\frac{\partial \text{Cost}(\hat{y},y)}{\partial w}=\frac{1}{m}(\hat{y}-y)^T X=\frac{1}{m}\left(\sigma(Xw+b)-y\right)^T X.$$

小批量训练是介于随机梯度下降与全批量梯度下降之间的一种折衷方法。它既能利用多个样本的统计信息提高梯度估计的准确性和更新的稳定性，又能通过向量化操作降低计算成本，从而在大规模数据训练中兼顾效率与效果。

梯度下降计算实例

一、示例背景与初始设置

1. 样本选择与特征构造

选取一个简化示例：对单个训练样本，其特征向量仅包含两个分量：

- $x_1 = 3$：表示该文档中正面情感词（例如“great”、“awesome”等）的计数；

- $x_2 = 2$：表示该文档中负面情感词（例如“poor”、“terrible”等）的计数

该样本的真实标签为 $y = 1$，即表示正面情感。

2. 初始参数设置

模型参数包括权重向量和偏置项。初始权重和偏置均设为0：

$$w_1 = 0, \quad w_2 = 0, \quad b = 0$$

同时，初始学习率设为 $\eta = 0.1$。

二、单步梯度下降更新过程

1. 梯度计算公式

根据逻辑回归交叉熵损失的求导结果，对于单个样本，其梯度关于每个参数的偏导数表达式为：

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

其中，$\sigma(z)$ 表示 sigmoid 函数，其定义为 $\sigma(z) = \frac{1}{1+e^{-z}}$。

2. 在初始参数下计算梯度

由于初始参数均为0，则有：

$$z = w \cdot x + b = 0 \quad \Longrightarrow \quad \sigma(0) = 0.5$$

因此，对于该样本（$y = 1$）：

- 对 w_1 的偏导数为：

$$\frac{\partial L_{\text{CE}}}{\partial w_1} = (\sigma(0) - 1)x_1 = (0.5 - 1) \times 3 = -1.5$$

- 对 w_2 的偏导数为：

$$\frac{\partial L_{\text{CE}}}{\partial w_2} = (\sigma(0) - 1)x_2 = (0.5 - 1) \times 2 = -1.0$$

- 对偏置 b 的偏导数为：

$$\frac{\partial L_{\text{CE}}}{\partial b} = \sigma(0) - 1 = 0.5 - 1 = -0.5$$

整体梯度向量为：$\nabla_{w,b} L = [-1.5, -1.0, -0.5]^T$。

3. 参数更新

根据梯度下降更新规则：

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta L,$$

其中 θ 表示包含 w_1, w_2 以及 b 的参数向量。将初始参数 $\theta^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ 和计算得到的梯度代入，利用学习率 $\eta = 0.1$ 得到：

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - 0.1 \times \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.1 \\ 0.05 \end{bmatrix}.$$

这表明在经过一次梯度下降更新后，参数从初始的全零状态调整为 $w_1 = 0.15$，$w_2 = 0.1$，$b = 0.05$。

三、结果解释

1. 参数更新的意义

更新后的参数表明模型对正面情感的证据进行了正向修正。具体来说：

- w_1（与正面词计数相关）变为正值，增加了正类的概率；

- w_2（与负面词计数相关）在本次更新中也变为正值，理想情况下如果遇到更多负例（其中 x_2 较大且 $y = 0$），w_2 将会向负方向调整，从而更准确地区分正负情感。

2. 学习过程的动态性

本次更新仅基于单个正样本。整体模型训练过程中，梯度下降（或小批量训练）会不断利用多个样本的梯度信息，对所有参数进行迭代更新，使得模型最终能够以较低的平均损失拟合整个训练集。

目录

1. 正则化的背景与动机

过拟合问题：在模型训练过程中，如果模型参数（权重）能够使模型在训练数据上达到完美拟合，则可能会出现“过拟合”现象。也就是说，某些特征如果在训练集中完全区分了各类别，就会被赋予极大的权重，从而模型**不仅捕捉到了数据中真实的模式，还过分“记住”了训练数据中的噪声和偶然相关性，导致在新数据（测试数据）上表现不佳**。

为了使模型能够**更好地泛化到未见数据**，需要对模型参数进行约束，从而防止权重过大或过于复杂。这就是引入正则化项的根本动机。

2. 正则化项的基本构造

在优化目标（损失函数）中，在原有的最大化训练数据对数似然的基础上，额外加入一个正则化项，以惩罚权重过大的情况。(最大化)目标函数可以写为：

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

其中：- $\theta = \{w, b\}$ 表示模型的全部参数；

- $R(\theta)$ 为正则化项，α 为正则化超参数，控制正则化项在整体目标中的权重。

正则化项的作用：如果模型完美拟合训练数据但需要使用极大权重，则正则化项会对这样的参数设置施加惩罚，从而使得模型更倾向于选择那些虽然拟合效果稍逊但权重较小、泛化能力更好的参数组合。

3. 常见的正则化方式

L2正则化（岭回归）：L2正则化项利用参数向量的L2范数（即欧几里得距离的平方）：

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

将L2正则化项加入到原始的目标函数中，得到正则化后的目标为：

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

特点：L2正则化倾向于使所有参数变得较小，即分散调整权重。其优点在于求导较简单（$\frac{d}{d\theta}\theta^2 = 2\theta$），便于数值优化。L2正则化对应于假设参数服从均值为0的高斯（Gaussian）分布，即大多数参数应集中在0附近，且偏离0的概率随着距离的增加而迅速下降。

L1正则化（Lasso回归）：L1正则化项利用参数向量的L1范数（即各参数绝对值之和）：

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

将L1正则化项加入到原始目标函数中，目标函数为：

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

特点：L1正则化倾向于使得部分参数精确地变为0，从而产生稀疏解，这在特征选择上有明显优势。然而，由于绝对值函数在0处不可微，求导处理相对复杂。L1正则化对应于对参数施加拉普拉斯（Laplace）先验，即认为参数分布服从尖峰且尾部较重的分布。

目录

1. 多项式逻辑回归损失函数的构建

在二元逻辑回归中，标签 y 只有两种取值（例如 0 与 1），其常用的交叉熵损失函数为

$$L_{CE}(\hat{y}, y) = -\big[y \log \hat{y} + (1-y) \log(1-\hat{y})\big].$$

对于多类别问题（多项式逻辑回归），标签不再是单一的0或1，而是用一个**one-hot向量**表示：假设共有 K 个类别，则对于正确类别 c 有 $y_c = 1$，其它类别 $y_j = 0$（$j \neq c$）。

损失函数表达：最大化**模型对单个观测 x 生成正确标签的概率**$p(y \mid x) = \prod_{k=1}^{K} \hat{y}_k^{y_k}$，多项式逻辑回归的交叉熵损失函数推广为：

$$L_{CE}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

由于只有正确类别 c 对应的 $y_c = 1$，其它 $y_j = 0$（$j \neq c$），故上述求和实际上只剩下正确类别一项，即

$$L_{CE}(\hat{y}, y) = -\log \hat{y}_c$$

输出概率计算：模型对于输入 x 的输出分数（也称为 logits）对于每个类别 k 由线性函数计算得到：

$$z_k = w_k \cdot x + b_k,$$

然后利用 softmax 函数将这些分数转换为概率分布：

$$\hat{y}_k = p(y_k = 1|x) = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

因此，对于正确类别 c 的预测概率为

$$\hat{y}_c = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

损失函数即为

$$L_{CE}(\hat{y}, y) = -\log \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

2. 梯度推导及参数更新

与二元逻辑回归类似，多项式逻辑回归的梯度推导目标在于计算损失函数关于各类别权重的偏导数，从而指导参数更新。对于每个类别 k 及对应权重 $w_{k,i}$（即第 i 个输入特征对应于类别 k 的权重），梯度的表达形式为：

$$\frac{\partial L_{CE}}{\partial w_{k,i}} = -(y_k - \hat{y}_k)\, x_i$$

这里，y_k 是真实标签（只有正确类别取1，其它类别取0），而 $\hat{y}_k$ 则是由 softmax 得到的预测概率。

参数更新：基于上式，多项式逻辑回归采用梯度下降（或其变种，如随机梯度下降）更新参数。令学习率为 η，则对于每个参数有：

$$w_{k,i}^{(t+1)} = w_{k,i}^{(t)} - \eta \frac{\partial L_{CE}}{\partial w_{k,i}},$$

即

$$w_{k,i}^{(t+1)} = w_{k,i}^{(t)} + \eta\,(y_k - \hat{y}_k)\, x_i$$

多类别逻辑回归梯度推导概要

① 从多类别的交叉熵损失函数

$$L_{CE} = -\sum_{k=1}^{K} y_k \log \hat{y}_k,$$

对 $w_{k,i}$ 求偏导；

② 利用链式法则和对对数函数求导的结果（$\frac{d}{dz}\log z = \frac{1}{z}$），可以将求导过程拆分为两部分：一部分是关于 $\hat{y}_k$ 的导数，另一部分是 $\hat{y}_k$ 关于 $w_{k,i}$ 的导数；

③ 又利用了 softmax 函数的导数性质（对于 sigmoid 函数，其导数为 $\sigma(z)(1-\sigma(z))$，softmax 的导数推广后会得到类似形式，一个雅可比矩阵）；

④ 最终得到的结果便是

$$\frac{\partial L_{CE}}{\partial w_{k,i}} = -(y_k - \hat{y}_k)\, x_i.$$

目录

1. 解释性的重要性

- 在实际应用中，不仅需要知道模型的分类结果，更希望理解模型为何做出这种决策。也就是说，模型的“可解释性”要求能够揭示**各个特征在决策过程中的作用和影响**。这种**解释能力对于构建透明、可信的模型至关重要**，特别是在商业和决策支持等领域，用户需要知道模型判断背后的依据。

2. 逻辑回归的可解释性优势

逻辑回归模型通常采用人为设计的特征，这使得每个特征对应的权重具有明确的含义。通过观察特征对应的权重（如某特征对应的权重大小及其正负符号），可以直观判断该特征对分类决策的正向或负向影响。例如，在情感分类中，某些词（如“好”、“喜欢”）可能具有正权重，而“差”、“糟糕”等词则具有负权重。

3. 统计检验与权重大小的作用

为了进一步解释模型决策，逻辑回归常常结合统计检验方法，如似然比检验或Wald检验，用以判断某一特征是否在统计上显著地影响分类结果。通过这些检验，可以确认哪些特征对模型预测起到了关键作用，从而帮助使用者理解模型的决策机制。

4. 逻辑回归作为分析工具的应用

除了作为分类器，逻辑回归还被广泛用作分析工具，帮助探讨不同解释变量（特征）与目标变量之间的关系。在文本分类等任务中，可以利用逻辑回归来检验某些语言特征（例如，否定词出现的频率）是否与情感极性、电影评论中的某些主题（如摄影技术）等结果有关，同时通过控制其他混杂因素（如电影类型）来验证这种关系的独立性。

课后练习

1. 复习下面微积分中的求导知识点，自行推导二分类逻辑回归和多分类逻辑回归中损失函数对模型参数的梯度。

- 对数函数的导数：

$$\frac{d}{dx}\ln(x) = \frac{1}{x}$$

- Sigmoid函数的导数：

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$

- 链式法则：对于复合函数 $f(x) = u(v(x))$，有

$$\frac{df}{dx} = \frac{du}{dv}\cdot\frac{dv}{dx}$$

- Softmax的导数涉及到一个雅可比矩阵（Jacobian Matrix），其中每个元素表示输出 s_i 对输入 z_j 的偏导数。经过求导计算，可以证明：

$$\frac{\partial s_i}{\partial z_j} = s_i(\delta_{ij} - s_j)$$

其中，δ_{ij} 是Kronecker δ函数，当 $i=j$ 时 $\delta_{ij}=1$，当 $i\neq j$ 时 $\delta_{ij}=0$。

THE END