5. 逻辑回归

“您怎么知道这些美丽的秋海棠并不同样重要呢？”

赫尔克里·波洛，在阿加莎·克里斯蒂的《斯泰尔斯的神秘事件》

侦探小说充斥着线索，就像文本充斥着单词一样。然而，对于可怜的读者来说，要知道如何权衡作者的线索来完成关键的分类任务：**决定谁是真凶(deciding whodunnit)**，这是很有挑战性的。

在本章中，我们介绍一种非常适合发现特征或线索与某些特定结果之间的联系的算法：**逻辑回归(logistic regression)**。实际上，逻辑回归是社会科学和自然科学中最重要的分析工具之一。在自然语言处理中，逻辑回归是用于分类的基准监督机器学习算法，并且与神经网络有着非常密切的关系。正如我们将在第 7 章中看到的那样，神经网络可以看作是一系列相互堆叠的逻辑回归分类器。因此，本文介绍的分类和机器学习技术将在整本书中发挥重要作用。

逻辑回归可用于将观察结果分为两类之一(例如“积极情感”和“消极情感”)，也可以分为许多类别之一。由于两类情况的数学比较简单，因此我们将在接下来的几节中首先介绍这种逻辑回归的特殊情况，然后在 5.6 节中简要总结多元逻辑回归在两个以上类中的使用。在接下来的几节中，我们将介绍逻辑回归的数学原理。但是，让我们从一些高级问题开始。

生成性(generative)和**区分性(discriminative)**分类器：朴素贝叶斯和逻辑回归之间的最重要区别是，逻辑回归是一种区分性分类器，而朴素贝叶斯是生成性分类器。

对于如何构建机器学习模型，这是两个非常不同的框架。考虑一个视觉隐喻：想象一下我们正在尝试区分狗图像和猫图像。生成模型的目标是了解狗的外观和猫的外观。您可能会从字面上要求这样的模型“生成”(绘)一只狗。给定测试图像后，系统会询问最适合该图像(对它不那么惊讶)的是猫模型还是狗模型，然后选择它作为标签。

相比之下，区分性模型仅仅是试着去学习区分这些类(也许并没有对他们有太多的了解)。所以也许训练数据中所有的狗都戴着项圈而猫没有。如果有一个特征能很好地将这些类分开，那么模型就满足了。如果你问模型，它对猫了解多少，它只能说猫不戴项圈。

更正式地说，回想一下，朴素贝叶斯不是通过直接计算 P(c | d)而是通过计算似然和先验来为文档 d 分配类 c：

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \tag{5.1}$$

像朴素贝叶斯这样的生成模型使用了这个似然项，它表示：如果我们知道文档属于 c 类(if we knew it was of class c)，则如何去生成文档的特征。

相比之下，在这个文本分类场景中，一个区分性模型尝试直接计算 P(c|d)。也许它将学会给文档特征赋予很高的权重，这些特征可以直接提高它在这些可能的类中作区分的能力，即便它不能生成其中一个类的一个示例。

概率机器学习分类器的组成部分：与朴素贝叶斯一样，逻辑回归是一种利用监督机器学习的概率分类器。机器学习分类器需要 m 个输入/输出对($x^{(i)}$，$y^{(i)}$)的训练语料。(对于情感分类，我们将使用括号中的上标来指代训练集中的单个实例，每个实例可能是一个要分类的单独文档)。用于分类的机器学习系统有四个组成部分:

1. 输入的一种特征表示。对于每个输入观测 $x^{(i)}$，这将是一个特征向量$[x_1, x_2, \ldots, x_n]$。我们通常将输入

$x^{(i)}$的特征 i 称为 $x_i^{(i)}$，有时简化为 x_i，但我们也会看到符号 f_i, $f_i(x)$，或者，对于多类分类，$f_i(c, x)$。

2. 一个分类函数，它通过 p(y | x)计算被估类的$\hat{y}$。下一节，我们将介绍用于分类的 sigmoid 和 softmax。
3. 一种用于学习的目标函数，通常包括最小化训练示例中的误差。我们将引入交叉熵损失函数。
4. 一种优化目标函数的算法。我们介绍了随机梯度下降算法。

Logistic 回归分为两个阶段:

训练:我们使用随机梯度下降和交叉熵损失训练系统(特别是权值 w 和 b)。

测试:给定一个测试示例 x，我们计算 p(y|x)并返回更高的概率标签 y = 1 或 y = 0。

5.1. 分类: sigmoid

二元逻辑回归的目标是训练一个分类器，使其能够对新的输入观察的类做出二元决策。在这里我们介绍 sigmoid 分类器，它将帮助我们做出这个决策。

考虑单个输入观测值 x，我们将用特征向量$[x_1, x_2, \ldots, x_n]$表示它(我们将在下一部分中显示示例特征)。分类器输出 y 可以为 1(表示观察值是该类的成员)或 0(表示观察值不是该类的成员)。我们想知道该观察是该类成员的概率 P(y=1|x)。因此，也许决策是“积极情感”和“消极情感”，这些特征表示文档中单词的数量，P(y=1|x)是文档具有积极情感的概率，P(y=0|x)是文档消极情感的可能性。

逻辑回归通过从训练集中学习权向量和偏差项来解决此任务。每个权重 w_i 是实数，并且与输入特征 x_i 之一相关联。权重 w_i 表示输入特征对分类决策的重要性，可以为正(提供证据表明正在被分类的实例属于肯定类)或负(提供证据表明正在被分类的实例属于否定类)。因此，我们可能期望在情感任务中，“awesome”这个词具有很高的正权重，而“absmal”则具有非常负的权重。**偏差项(bais term)**,也称为**截距(intercept)**,是添加到加权输入中的另一个实数。

为了在测试实例上做出决策(在我们学习了训练的权重之后)，分类器首先将每个 x_i 乘以权重 w_i，对加权特征求和，然后加上偏差项 b。所得的单个数字 z 表示该类别证据的加权总和。

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \tag{5.2}$$

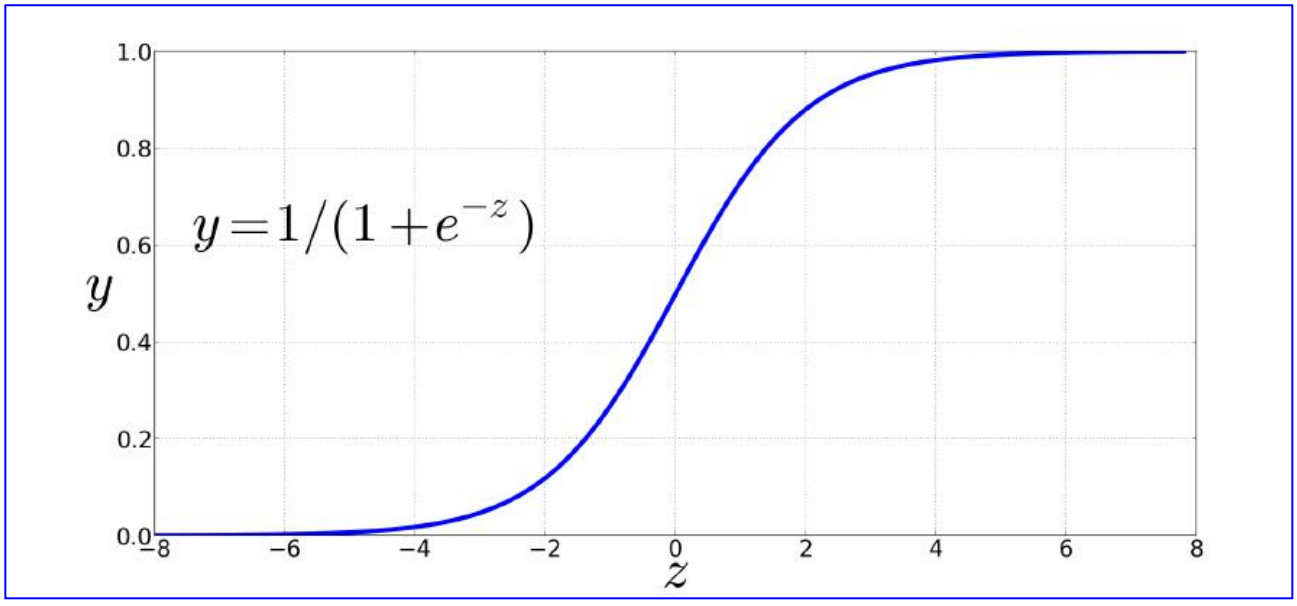

图 5-1：sigmoid 函数

图注：y 取一个实数，并将其映射到范围[0, 1]。它在 0 附近几乎是线性的，但是离群值被压向 0 或 1。

在本书的其余部分，我们将使用线性代数中的**点积(dot product)**符号来表示这样的和。两个向量 a 和 b 的点积，写成 a·b，是每个向量对应元素的乘积的和。因此，以下是与公式 5.2 等价的形式:

$$z = w \cdot x + b \tag{5.3}$$

但请注意，公式 5.3 中没有任何东西强制 z 为合法概率，即介于 0 和 1 之间。事实上，由于权值是实数值，输出甚至可能是负的;z 的取值范围是-∞到∞。

为了产生概率，我们将通过 **sigmoid** 函数σ(z)传递 z。S 形函数(sigmoid)(之所以称呼它，是因为它看起来像一个 s)，也称为**逻辑函数(logistic function)**，并赋予逻辑回归这个名称。如图 5.1 所示,S 型具有以下公式：

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)} \tag{5.4}$$

(在本书的其余部分中，我们将使用符号 exp(x)表示 e^x。)S 形有很多优点； 它采用一个实数值并将其映射到[0,1]范围内，这正是我们想要的概率。由于它在 0 附近几乎是线性的，但在两端逐渐变平，因此倾向于将离群值压缩到 0 或 1。而且它是可微的，正如我们在 5.8 节中将看到的那样，它很容易学习。

我们差不多了。如果将 Sigmoid 应用于加权特征的总和，我们将得到一个介于 0 和 1 之间的数字。要使其成为概率，我们只需要确保 p(y = 1)和 p(y = 0)的总和为 1。我们可以这样进行操作：

$$\begin{aligned} P(y=1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1+\exp(-(w \cdot x + b))} \\ P(y=0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1+\exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1+\exp(-(w \cdot x + b))} \end{aligned} \tag{5.5}$$

S 型函数有这样的性质:

$$1 - \sigma(x) = \sigma(-x) \tag{5.6}$$

所以我们也可以把 P(y = 0)表示为σ(- (w·x+b))。

现在我们有了一个算法，给定一个实例 x 计算概率 P(y = 1 | x)。我们如何做出决策？对于测试实例 x，如果概率 P(y = 1 | x)大于 0.5，则说“是”，否则说“否”。我们将 0.5 称为**决策边界(decision boundary)**：

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

5.1.1. 例子：情感分类

让我们举个例子。假设我们正在对电影评论文本进行二进制情感分类，并且我们想知道是否将情感类别+或-分配给评论文档 doc。我们将通过下表中显示的输入的 6 个特征 x_1 ... x_6 来表示每个输入观察值；图 5.2 显示了微型测试文档样本中的特征。

现在，假设我们已经为每个特征学习了一个实数值权重，并且与这 6 个特征相对应的 6 个权重为[2.5，5.0 ，1.2，0.5，2.0，0.7]，而 b = 0.1。(我们将在下一节中讨论如何学习权重。)例如，权重 w_1 表示积极词汇(great, nice, enjoyable 等)的数量的特征对一个积极情绪决策的重要性，而 w_2 则告诉我们消极词汇的重要性。注意，w_1 = 2.5 是积极的，而 w_2 =-5.0 意味着消极词汇与积极情绪决策呈负相关，其重要性大约

是积极词汇的两倍。

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

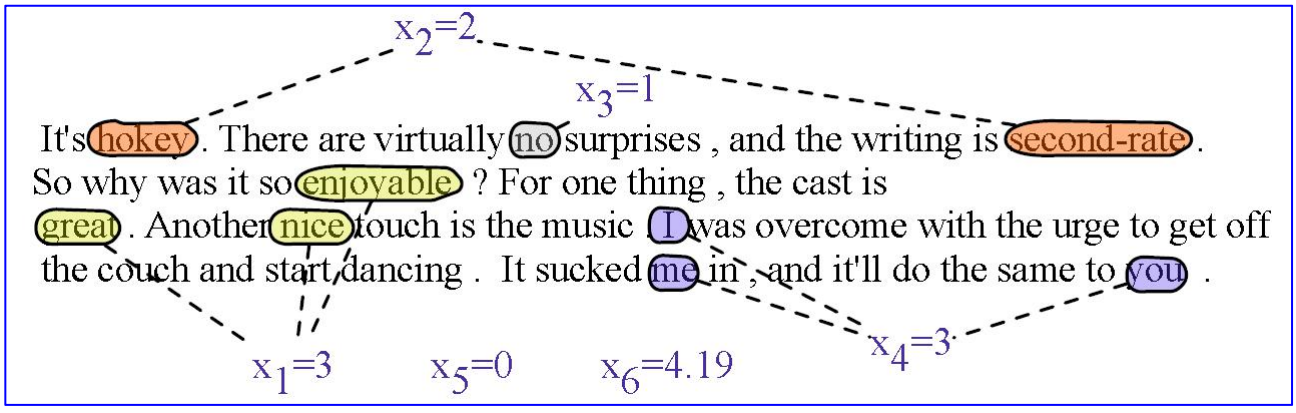

图 5-2：微型测试文档样本向量 x 中的提取特征

图注：微型测试文档的一个样本(显示了向量 x 中的提取特征)

给定这 6 个特征和输入评论 x, P(+|x)和 P(−|x)可以通过公式 5.5 计算:

$$\begin{aligned} p(+|x) = P(Y=1|x) &= \sigma(w\cdot x+b) \\ &= \sigma([2.5,-5.0,-1.2,0.5,2.0,0.7]\cdot[3,2,1,3,0,4.19]+0.1) \\ &= \sigma(.833) \\ &= 0.70 \\ p(-|x) = P(Y=0|x) &= 1-\sigma(w\cdot x+b) \\ &= 0.30 \end{aligned} \tag{5.7}$$

逻辑回归通常应用于各种 NLP 任务，输入的任何属性都可以是一个特征。考虑**句点消歧(period disambiguation)**的任务:通过将每个句点分为 EOS(句尾)和非 EOS 两类之一，确定句点是句子的结尾还是单词的一部分。我们可以使用下面的 x_1 这样的特征来表示：当前单词是小写的，其类为 EOS(可能有一个正的权重)；或者，当前单词在我们的缩写字典中("Prof")，其类为 EOS(可能有一个负的权重)。特征还可以表示相当复杂的属性组合。例如，大写单词后面的句号可能是 EOS，但如果单词本身是 St.，并且前面的单词大写，那么句号可能是单词 street 的缩写的一部分。

$$x_1 = \begin{cases} 1 & \text{if "}Case(w_i) = \text{Lower"} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if "}w_i \in \text{AcronymDict"} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if "}w_i = \text{St.} \ \& \ Case(w_{i-1}) = \text{Cap"} \\ 0 & \text{otherwise} \end{cases}$$

设计特征：特征的设计通常是通过观察语言直觉和该领域的语言文献来检查训练集。对系统早期版本

的训练集或开发集进行仔细的误差分析，通常可以洞悉特征。

对于某些任务，构建由更多原始特征组合而成的复杂特征特别有用。我们在上面看到了用于句点消歧的特征，如果前一个单词是大写的，则单词 St.上的句点不太可能是句子的结尾。对于逻辑回归和朴素贝叶斯，必须手动设计这些组合特征或**特征交互(feature interactions)**。

对于许多任务(特别是当特征值可以引用特定的单词时)，我们需要大量的特征。这些通常是通过**特征模板(feature templates)**(特征的抽象规范)自动创建的。例如，用于句点消歧的 bigram 模板可能为出现在训练集中句点之前的每对单词创建一个特征。因此，特征空间是稀疏的，因为我们仅需在训练集中该 n-gram 存在于该位置的情况下创建特征。通常根据字符串的描述将特征创建为哈希(hash)。用户对诸如“bigram(American breakfast)”特征的描述为被哈希为唯一的整数 i，该整数成为特征数 f_i。

为了避免大量人为的特征设计工作，最近的 NLP 研究集中于**表示学习(representation learning)**:从输入中以无监督的方式自动学习特征的方法。我们将在第 6 章和第 7 章中介绍表示学习的方法。

选择分类器 与朴素贝叶斯相比，逻辑回归具有许多优势。朴素贝叶斯有过分强的条件独立性假设。考虑两个密切相关的特征； 实际上，假设我们只是将相同的特征 f_1 加了两次。朴素贝叶斯会将 f_1 的两个副本视为分开的，将它们相乘，从而高估了证据。相比之下，逻辑回归对相关特征更为健壮。如果两个特征 f_1 和 f_2 完全相关，则回归将简单地将部分权重分配给 w_1，将部分权重分配给 w_2。因此，当存在许多相关特征时，逻辑回归将比朴素贝叶斯分配更准确的概率。因此，逻辑回归通常在较大的文档或数据集上效果更好，这是常见的默认设置。

尽管概率不太准确，朴素贝叶斯仍然经常做出正确的分类决策。此外，朴素贝叶斯可以在非常小的数据集(Ng 和 Jordan, 2002)或简短的文档(Wang 和 Manning, 2012)上工作得非常好(有时甚至比逻辑回归更好)。此外，朴素贝叶斯很容易实现和非常快的训练(没有优化步骤)。所以在某些情况下，这仍然是一个合理的方法。

5.2. 在逻辑回归中学习

如何学习模型的参数(权重 w 和偏差 b)？ 逻辑回归是监督分类的一个实例，其中我们知道每个观测值 x 的正确标签 y(0 或 1)。系统通过公式 5.5 产生的结果是$\hat{y}$，即系统对真实 y 的估计。我们想学参数(即 w 和 b)，使每个训练观察的$\hat{y}$尽可能接近真实的 y。

这需要在本章的简介中提到的两个组件。

第一个度量标准是当前标签($\hat{y}$)与真实黄金标签 y 的接近程度。我们通常考虑的不是度量相似度，而是与之相反的情况：系统输出与黄金输出之间的距离，我们将此距离称为损失函数或成本函数。在下一节中，我们将介绍通常用于逻辑回归和神经网络的损失函数，即交叉熵损失。

第二个是用于迭代更新权重的优化算法，以便最小化此损失函数。对此的标准算法是梯度下降。在以下部分中，我们将介绍随机梯度下降算法。

5.3. 交叉熵损失函数

对于观察 x，我们需要一个损失函数来表示分类器输出($\hat{y} = \sigma(w \cdot x + b)$)与正确的输出(y，即 0 或 1)有多接近。我们将其称为：

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y \tag{5.8}$$

我们通过损失函数来做到这一点，损失函数更倾向于训练示例的正确类标签。这称为条件最大似然估计：我们选择参数 w，b，该参数在给定观察值 x 的情况下最大化训练数据中真实 y 标签的对数概率。最终的损失函数为负对数似然损失，通常称为**交叉熵损失(cross-entropy loss)**。

让我们导出这个损失函数，将其应用于单个观测值 x。我们想学习权重，以最大化正确标签 p(y | x)的概率。由于只有两个离散的结果(1 或 0)，所以这是一个伯努利分布，对于我们的分类器为一个观察而产生的概率 p(y | x)(请注意，如果 y = 1，则公式 5.9 简化为$\hat{y}$；如果 y = 0，则公式 5.9 简化为 1-$\hat{y}$)，我们可以如下表达:

$$p(y|x) = \hat{y}^y (1-\hat{y})^{1-y} \tag{5.9}$$

现在我们对两边取对数。这在数学上是很方便的，而且不会对我们造成伤害;使概率最大化的值也将使概率的对数最大化:

$$\begin{aligned}\log p(y|x) &= \log\left[\hat{y}^y (1-\hat{y})^{1-y}\right] \\ &= y\log\hat{y} + (1-y)\log(1-\hat{y})\end{aligned} \tag{5.10}$$

公式 5.10 描述了应该最大化的对数似然。为了将其转化为损失函数(我们需要将其最小化)，我们只需将公式 5.10 上的符号翻转即可。结果是交叉熵损失 L_{CE}：

$$L_{CE}(\hat{y},y) = -\log p(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})] \tag{5.11}$$

最后，我们可以插入ŷ =σ(w·x + b)的定义：

$$L_{CE}(\hat{y},y) = -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))] \tag{5.12}$$

让我们看看这个损失函数是否可以正确完成图 5.2 中的示例。如果模型的估算值接近正确，我们希望损失会减少；如果模型混乱，则损失会更大。因此，首先让我们假设图 5.2 中情感示例的正确黄金标签为正，即 y =1。在这种情况下，我们的模型运行良好，从公式 5.7 可以看出，这个例子的积极(0.69)概率确实高于消极(0.31)概率。如果我们将σ(w • x+b) = 0.69 和 y = 1 代入公式 5.12 中，方程右边就会去掉，导致如下损失(我们将使用对数表示未指定底数时的自然对数):

$$\begin{aligned}L_{CE}(\hat{y},y) &= -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))] \\ &= -[\log\sigma(w\cdot x+b)] \\ &= -\log(.69) \\ &= .37\end{aligned}$$

相比之下，让我们假设图 5.2 中的例子实际上是消极的，即 y = 0(也许评论者接着说:“但最重要的是，这部电影很糟糕!我求你不要看它!”)。在这种情况下，我们的模型是混乱的，我们希望损失更大。现在，如果我们将公式 5.7 中的 y = 0 和 1-σ(w·x+b) =0.31 代入公式 5.12，方程左边就会得到:

$$\begin{aligned}L_{CE}(\hat{y},y) &= -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))] \\ &= -[\log(1-\sigma(w\cdot x+b))] \\ &= -\log(.31) \\ &= 1.17\end{aligned}$$

当然，第一个分类器的损失(0.37)小于第二个分类器的损失(1.17)。

为什么我们想要最大程度地降低这种负对数概率？完美的分类器会将概率 1 分配给正确的结果(y = 1 或 y = 0)，将概率 0 分配给不正确的结果。这意味着ŷ越高(它越接近 1)，则分类器越好；ŷ越低(越接近 0)，分类器越差。此概率的负对数是一种方便的损失度量，因为它从 0(负对数 1，无损失)到无穷大(负对数 0，无穷损失)。该损失函数还确保：当正确答案的概率最大化时，错误答案的概率最小化。由于两者之和为 1，因此正确答案的概率的任何增加都会以错误答案为代价。之所以称为交叉熵损失，是因为公式 5.10 也是一个交叉熵的公式，这个交叉熵横跨在真实概率分布 y 和估计分布ŷ之间。

现在我们知道了要最小化的内容； 在下一部分中，我们将介绍如何找到最小值。

5.4. 梯度下降

我们使用梯度下降的目标是找到最优的权值:最小化我们为模型定义的损失函数。在下面的公式 5.13

中，我们将明确表示这样一个事实，即损失函数 L 是由权值参数化的，在机器学习中我们通常将其称为θ(在逻辑回归θ= w,b 的情况下)。所以我们的目标是找到一组使损失函数最小的权值，在所有例子上取平均值:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)};\theta), y^{(i)}) \tag{5.13}$$

我们如何找到这个(或任何)损失函数的最小值？ 梯度下降法是通过找出函数的斜率在哪个方向上(在参数θ的空间内)最陡峭地上升，并在相反方向上移动来找到函数的最小值。直觉是，如果您在峡谷中徒步旅行，并试图最快速地下降到底部的河流，则可能会环顾四周 360 度，找到地面最陡峭的方向，然后沿该方向下坡 。

对于逻辑回归，此损失函数通常是**凸的(convex)**。凸函数只有一个最小值。没有局部最小值可以陷入，因此保证从任何一点开始的梯度下降都可以找到最小值。(相比之下，多层神经网络的损失是非凸的，梯度下降可能会卡在用于神经网络训练的局部最小值中，而找不到全局最优值。)

尽管这个算法(和梯度的概念)是为方向矢量设计的，但让我们首先考虑一下可视化的情况，其中我们的系统参数仅为单个标量 w，如图 5.3 所示。

给定 w 在某个值 w^1 处的随机初始化，并假设损失函数 L 恰好具有图 5.3 的形状，我们需要一种算法来告诉我们在下一次迭代时是否应向左移动(使 w^2 小于 w^1) 或向右(使 w^2 大于 w^1)以便达到最小值。

梯度下降算法通过找到损失函数在当前点的梯度并向相反的方向移动来回答这个问题。多变量函数的梯度是指向函数中增量最大方向的向量。梯度是对斜率的多变量概括，所以对于像图 5.3 这样的单变量函数，我们可以非正式地把梯度看作斜率。图 5.3 中的虚线显示了这个假设的损失函数在 $w = w^1$ 点的斜率。可以看到这条虚线的斜率是负的。因此，为了找到最小值，梯度下降告诉我们以相反的方向:将 w 移动到正的方向。

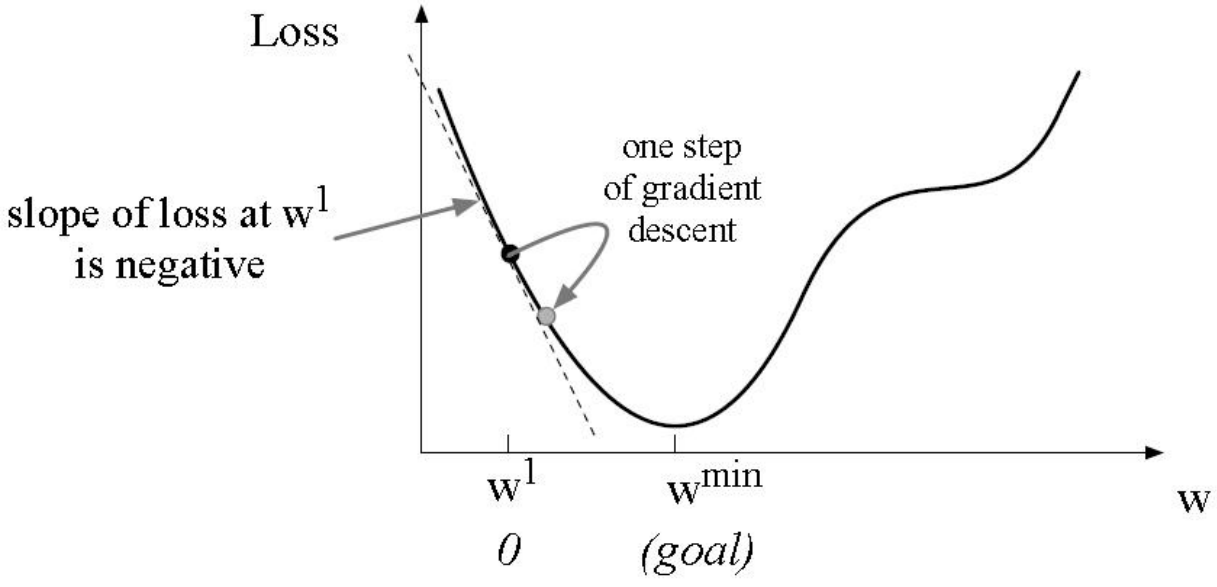

图 5-3：梯度下降算法的可视化

图注：通过从函数的斜率反向移动 w 来迭代找到此损失函数的最小值的第一步。由于斜率为负，因此我们需要将 w 沿正方向向右移动。这里的上标用于学习步骤，因此 w^1 表示 w 的初始值(为 0)，第二步为 w^2，依此类推。

梯度下降的移动量的大小是由**学习率(learning rate)**η加权的斜率$\frac{d}{dw}$f(x;w)的值。更高(更快)的学习速度意味着我们应该在每一步上增加更多。我们在参数中所做的更改是学习率乘以梯度(或斜率，在我们的单变量示例中)：

$$w^{t+1} = w^t - \eta \frac{d}{dw} f(x; w) \tag{5.14}$$

现在让我们将直觉地从一个标量变量 w 的函数扩展到许多变量，因为我们不只是想向左或向右移动，我们想知道在 N 维空间中(构成θ的 N 个参数) ，我们应该如何移动。梯度就是这样一个向量，它表示沿这 N 个维度中的每个维度的最大斜率的方向分量。如果我们仅想象两个权重维度(例如，对于一个权重 w 和一个偏置 b)，则梯度可能是具有两个正交分量的向量，每个分量都告诉我们 w 维度和 b 上的地面坡度尺寸。图 5.4 显示了在红点处获取的二维梯度向量值的可视化。

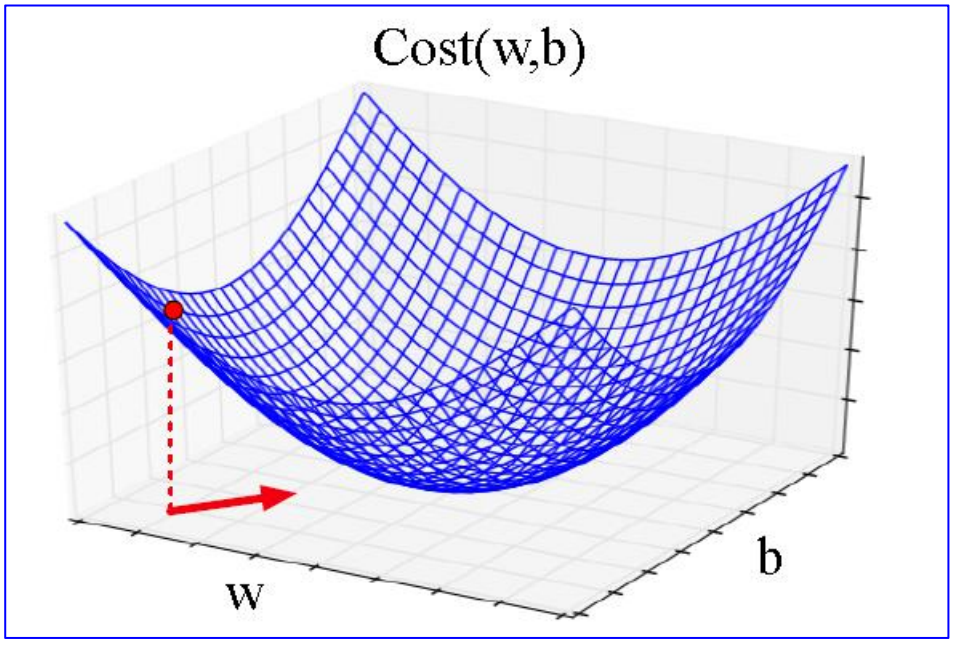

图 5-4：梯度矢量的可视化

图注：二维 w 和 b 上红色点处的梯度矢量的可视化，在 x-y 平面上以红色箭头显示了梯度。

在实际的逻辑回归中，参数向量 w 比 1 或 2 长得多，因为输入特征向量 x 可能很长，并且每个 x_i 需要权重 w_i。对于 w 中的每个维度/变量 w_i(加上偏差 b)，梯度将具有一个分量，该分量告诉我们相对于该变量的斜率。本质上，我们在问：“该变量 w_i 的微小变化会对总损失函数 L 产生多大影响？”

在每个维度 w_i 中，我们将斜率表示为损失函数的偏导数$\partial l/\partial w_i$。然后将梯度定义为这些偏导的向量。我们将 y 表示为 f(x;θ)，以使对θ的依赖性更加明显：

$$\nabla_\theta L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix} \tag{5.15}$$

因此，θ是基于梯度而更新的，其最终方程为：

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta),y) \tag{5.16}$$

5.4.1. 逻辑回归的梯度

为了更新θ，我们需要定义梯度∇L(f(x;θ)，y)。回想一下，对于逻辑回归，交叉熵损失函数为：

$$L_{CE}(\hat{y},y) = -[y \log \sigma(w \cdot x + b) + (1-y) \log (1 - \sigma(w \cdot x + b))] \tag{5.17}$$

事实证明，该函数对一个观测向量 x 的导数为公式 5.18(对此公式的推导，读者可以参见 5.8 节)：

$$\frac{\partial L_{CE}(\hat{y},y)}{\partial w_j} = [\sigma(w\cdot x+b)-y]x_j \tag{5.18}$$

注意，在公式 5.18 中，相对于单个权重 w_j 的梯度，表达一个非常直观的值：真实的 y 与该观测值估算的$\hat{y}=\sigma(w\cdot x + b)$之差乘以相应的输入值 x_j。

5.4.2. 随机梯度下降算法

随机梯度下降是一种在线算法，通过在每个训练示例之后计算其梯度，并在正确的方向(梯度的相反方向)上推动θ来最小化损失函数。图 5.5 显示了该算法。

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
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
```

图 5-5：随机梯度下降算法

图注：步骤 1(计算损失)用于报告我们在当前元组上的表现。该算法可以在收敛时(或在梯度范数<ϵ时)或进度停止时(例如，损失在保留集上开始上升)而终止。

学习速率η是必须调整的**超参数(hyperparameter)**。如果它的值太高，学习者将采取太大的步子，超出损失函数的最小值。如果它的值太低，学习者将采取太小的步子，并且花费太长时间才能达到最低要求。通常从较高的学习率开始，然后缓慢降低它，使其成为训练迭代 k 的函数。符号η_k可用于表示迭代 k 处学习率的值。

我们将在第 7 章中更详细地讨论超参数，但简要地讲，它们对于任何机器学习模型都是一种特殊的参数。与算法从训练集中学习的模型的常规参数(权重 w 和 b)不同，超参数是算法设计者选择的特殊参数，会影响算法的工作方式。

5.4.3. 弄通一个例子

让我们逐步了解一下梯度下降算法。我们将使用图 5.2 中示例的简化版本，因为它看到一个观测值 x，其正确值为 y = 1(这是积极的评论)，并且只有两个特征：

x_1 = 3(积极词汇数)

x_2 = 2(消极词汇数)

假设θ^0的初始权重和偏差都设置为 0，初始学习率η为 0.1：

$w_1 = w_2 = b = 0$

$\eta = 0.1$

单个更新步骤要求我们计算梯度乘以学习率：

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

在我们的小型示例中，有 3 个参数，因此梯度向量具有 3 个维度，分别是 w_1，w_2 和 b。我们可以如下计算第一个梯度：

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w\cdot x+b)-y)x_1 \\ (\sigma(w\cdot x+b)-y)x_2 \\ \sigma(w\cdot x+b)-y \end{bmatrix} = \begin{bmatrix} (\sigma(0)-1)x_1 \\ (\sigma(0)-1)x_2 \\ \sigma(0)-1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

现在我们有了一个梯度，我们通过从梯度的相反方向移动θ^0来计算新的参数矢量θ^1：

$$\theta^1 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} .15 \\ .1 \\ .05 \end{bmatrix}$$

因此，在经过一个梯度下降步骤之后，权重已变为：$w_1 = .15$，$w_2 = .1$ 和 $b = .05$。

注意，该观察值 x 恰好是一个积极示例。我们希望看到更多带有大量消极词的消极示例后，权重 w_2 会变为负值。

5.4.4. 小批量训练

随机梯度下降法之所以被称为随机，是因为它一次选择单个随机示例，移动权重以提高该示例的性能。这可能会导致很不稳定的移动，因此通常在成批训练实例而非单个实例上计算梯度。

例如，在**批量(batch)**训练中，我们计算整个数据集的梯度。通过查看如此多的示例，批处理训练可以很好地估计权重的移动方向，计算此完美方向的代价是，要花费大量时间来处理训练集中的每个示例。

一个折衷方案是**小批量(mini-batch)**训练：我们训练一组小于整个数据集的 m 个示例(也许是 512 个或 1024 个)。(如果 m 是数据集的大小，那么我们将进行批量梯度下降；如果 m = 1，我们将返回随机梯度下降)。小批量训练还具有计算效率的优势。小批量可以很容易地进行矢量化，根据计算资源选择小批量的大小。这样一来，我们就可以在一个小批量处理中并行处理所有示例，然后累积损失，而这对于单个或批量训练而言是不可能的。

我们只需要定义在 5.3 节中定义的交叉熵损失函数和在 5.4.1 节中定义的梯度的小批量版本。让我们从公式 5.11 中把一个示例的交叉熵损失扩展到 m 大小的小批量。我们将继续使用 $x^{(i)}$和 $y^{(i)}$分别表示第 i 个训练特征和训练标签。我们假设训练示例是独立的：

$$\begin{aligned} \log p(\text{training labels}) &= \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) \\ &= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}) \\ &= -\sum_{i=1}^{m} L_{\text{CE}}(\hat{y}^{(i)}, y^{(i)}) \end{aligned} \tag{5.19}$$

现在，m 个示例的小批量生产的成本函数是每个示例的平均损失：

$$Cost(\hat{y},y) = \frac{1}{m}\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)},y^{(i)})$$
$$= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log\sigma(w\cdot x^{(i)}+b)+(1-y^{(i)})\log\left(1-\sigma(w\cdot x^{(i)}+b)\right) \quad (5.20)$$

小批量梯度是来自等式 5.18 的各个梯度的平均值：

$$\frac{\partial Cost(\hat{y},y)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left[\sigma(w\cdot x^{(i)}+b)-y^{(i)}\right]x_j^{(i)} \quad (5.21)$$

5.5. 正则化

Numquam ponenda est pluralitas sine necessitate

'Plurality should never be proposed unless needed'

"除非有必要，否则绝不应提出多个建议"

William of Occam

有一个学习权值的问题，使模型完全匹配训练数据。因为一个特征只出现在一个类别中，所以，如果它能够完美地预测结果，那么它将被赋予非常高的权重。特征的权重将试图完美(事实上过于完美)地拟合训练集的细节，对偶然与类相关的噪声因素进行建模。这个问题叫做**过拟合(overfitting)**。一个好的模型应该能够很好地从训练数据**概括(generalize)**到看不见的测试集，但一个过拟合的模型将会有很差的概括。

为了避免过拟合,一个新的**正则化(regularization)**项 R(θ)被添加到公式 5.13 的目标函数中，得出了一批 m 个示例的以下目标(从等式 5.13 略微重写，以使对数概率最大化而不是使损失最小化，并删除不影响 argmax 的 1/m 项)：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\sum_{i=1}^{m}\log P(y^{(i)}|x^{(i)}) - \alpha R(\theta) \quad (5.22)$$

新的正则化项 R(θ)用于惩罚较大的权重。

因此，假设有两种权重方案：

A：一个完全匹配训练数据的权重设置，但使用许多高值的权重；

B：一个稍好匹配训练数据的权重设置，但使用一些较小的权重。

那么，A 方案将受到更大的惩罚。

有两种常见的方法可以计算此正则项 R(θ)。

第一种是 L2 正则化。**L2 正则化(L2 regularization)**是权重值的二次函数，之所以这样命名，是因为它使用权重值的 L2 范数(的平方)。L2 范数，$||\theta||_2$，就是从原点到向量θ的距离(即向量的长度)，这个距离叫欧氏距离。如果θ由 n 个权重组成，则：

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n}\theta_j^2 \quad (5.23)$$

L2 正则化目标函数变为：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\left[\sum_{i=1}^{m}\log P(y^{(i)}|x^{(i)})\right] - \alpha\sum_{j=1}^{n}\theta_j^2 \quad (5.24)$$

第二种是 L1 正则化。**L1 正则化(L1 regularization)**是权重值的线性函数，以 L1 范数$||W||_1$命名，它是权重的绝对值之和或曼哈顿距离(曼哈顿距离是在像纽约这样有街道网格的城市里，你在两点之间所走的

距离)：

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i| \tag{5.25}$$

L1 正则化目标函数变为：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{1=i}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j| \tag{5.26}$$

这些类型的正则化来自统计学，其中 L1 正则化称为**套索(lasso)**回归(Tibshirani，1996)，而 L2 正则化称为**岭脊(ridge)**回归，这两者都常用于语言处理中。L2 正则化因其简单的导数(θ^2 的导数仅为 2θ)而更易于优化，而 L1 正则化则更复杂(|θ|的导数在零处不连续)。但是，在 L2 偏爱具有许多较小权重的权重向量的情况下，L1 偏爱具有一些较大权重但又有更多权重设置为零的稀疏解。因此，L1 正则化导致稀疏的权重向量，即特征少得多。

L1 和 L2 正则化均具有贝叶斯(Bayesian)解释，作为对权重外观的先验约束。L1 正则化可以被视为权重上的拉普拉斯先验。L2 正则化对应于假设权重根据均值μ = 0 的高斯分布进行分配。在高斯或正态分布中，一个值与平均值的距离越远，其概率(由方差σ缩放)越低。通过对权重使用高斯先验，我们说权重更喜欢值为 0。权重θ_j的高斯函数为：

$$\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2} \right) \tag{5.27}$$

如果将每个权重乘以权重的高斯先验，则将使以下约束最大化：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{M} P(y^{(i)}|x^{(i)}) \times \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2} \right) \tag{5.28}$$

在对数空间中，μ= 0，并假设 $2\sigma^2$= 1，它对应于：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha \sum_{j=1}^{n} \theta_j^2 \tag{5.29}$$

其形式与公式 5.24 相同。

5.6. 多元逻辑回归

有时我们需要两个以上的类。也许我们可能想进行 3 种情感分类(正面，负面或中性)。或者，我们可以分配一些我们将在第 8 章中介绍的标签，例如单词的词类(从 10、30 甚至 50 个不同的词类中进行选择)，或短语的命名实体类型(选择来自人物、位置、组织等标签)。

在这种情况下，我们使用**多元逻辑回归(multinomial logistic regression)**，也称为 **softmax 回归**(或历史上称为 maxent 分类器)。在多元逻辑回归中，目标 y 是一个变量，范围超过两个类别。我们想知道 y 在每个潜在的类 c∈C，p(y = c | x)中的概率。

多元逻辑分类器使用广义的 Sigmoid，称为 **softmax** 函数，来计算概率 p(y = c | x)。softmax 函数采用 k 个任意值的向量 z = [z_1，z_2，...，z_k]并将其映射到概率分布，每个值的范围为(0,1)，所有值的总和为 1。像 S 形一样，它是一个指数函数。

对于维数为 k 的向量 z，softmax 定义为：

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k}\exp(z_j)} \quad 1 \le i \le k \tag{5.30}$$

输入向量 z = [z_1，z_2，...，z_k]的 softmax 因此是向量本身：

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k}\exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k}\exp(z_i)}, \ldots, \frac{\exp(z_k)}{\sum_{i=1}^{k}\exp(z_i)}\right] \tag{5.31}$$

分母$\sum_{i=1}^{k}\exp(z_i)$用于将所有值归一化为概率。因此，例如给定一个向量：

z = [0.6，1.1，−1.5，1.2，3.2，−1.1]

结果(四舍五入)softmax(z)为：

[0.055，0.090，0.006，0.099，0.74，0.010]

再次像 Sigmoid 一样，softmax 的输入将是权重向量 w 和输入向量 x(加上偏置)之间的点积。但是，现在我们需要为 K 个类中的每个类使用单独的权重向量(和偏差)。

$$p(y=c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{k}\exp(w_j \cdot x + b_j)} \tag{5.32}$$

像 Sigmoid 一样，softmax 具有将值朝 0 或 1 压缩的特性。因此，如果输入之一大于其他输入，则它将倾向于将其概率推向 1，并抑制较小输入的概率。

5.6.1. 多元逻辑回归的特征

多元逻辑回归函数的特征类似于二元逻辑回归，不同之处在于我们需要为每一个 K 类单独的权向量(和偏差)。回想一下 5.1.1 的二进制感叹号特征 x_5:

$$x_5 = \begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$$

在二元分类中，一个特征的正权重 w_5 对分类器的影响趋向为 y = 1(积极情感)，一个负权重对分类器的影响趋向为 y = 0(消极情感)，其绝对值表示该特征的重要程度。相比之下，在多元逻辑回归中，每个类都有单独的权重，一个特征可以作为支持或反对每个单独类的证据。

例如，在三向(3-way)多类别情感分类中，我们必须为每个文档分配+，-或 0(中性)这 3 个类别之一。现在，与感叹号相关的特征可能对 0 文档的权重为负，而对+或-文档的权重为正：

Feature	Definition	$w_{5,+}$	$w_{5,-}$	$w_{5,0}$
$f_5(x)$	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	3.5	3.1	−5.3

5.6.2. 多元逻辑回归的学习

多元逻辑回归的损失函数概括了从 2 类到 K 类的二进制逻辑回归的损失函数。回想一下，二进制逻辑回归的交叉熵损失(从公式 5.11 重复)为：

$$L_{\text{CE}}(\hat{y}, y) = -\log p(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})] \tag{5.33}$$

多元逻辑回归的损失函数概括了方程式 5.33 中的两个项(当 y = 1 时有一个非零项，而当 y = 0 时也有一个非零项)到 K 个项。因此，单个示例 x 的损失函数是 K 个输出类的对数的总和，每个对数的权重为 y_k，

即真实类的概率：

$$\begin{aligned} L_{\mathrm{CE}}(\hat{y}, y) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\ &= -\sum_{k=1}^{K} y_k \log \hat{p}(y=k|x) \end{aligned} \tag{5.34}$$

由于只有一个类(我们称其为 i)是正确的类，因此向量 y 仅针对此 k 值取值为 1，即 $y_i=1$ 且 $y_j=0\ \forall j \neq i$。像这样的一个向量，其一个值=1，其余值为 0，被称为**独热向量**。因此，公式 5.34 中的总和项将是 0(除了对应于真实类的项)，即:

$$\begin{aligned} L_{\mathrm{CE}}(\hat{y}, y) &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\} \log \hat{p}(y=k|x) \\ &= -\sum_{k=1}^{K} \mathbb{1}\{y=k\} \log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)} \end{aligned} \tag{5.35}$$

因此，交叉熵损失仅仅是对应于正确类的输出概率的对数，因此我们也将其称为**负对数似然损失**：

$$\begin{aligned} L_{\mathrm{CE}}(\hat{y}, y) &= -\log \hat{y}_k, \quad (\text{where } k \text{ is the correct class}) \\ &= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)} \quad (\text{where } k \text{ is the correct class}) \end{aligned} \tag{5.36}$$

单个示例的梯度与二进制逻辑回归的梯度非常相似，尽管我们在此处未显示推导。它是真实类 k 的值(为 1)与分类器为类 k 输出的概率之间的差值，用 k 类中的第 i 个元素权重 $w_{k,i}$ 对应的输入 x_i 的值，进行加权计算:

$$\begin{aligned} \frac{\partial L_{\mathrm{CE}}}{\partial w_{k,i}} &= -(\mathbb{1}\{y=k\} - p(y=k|x))x_i \\ &= -\left(\mathbb{1}\{y=k\} - \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}\right) x_i \end{aligned} \tag{5.37}$$

5.7. 解释模型

通常，我们不仅要了解观察的正确分类，还希望了解更多。我们想知道为什么分类器做出了决策。也就是说，我们希望我们的决策是可以**解释的(interpretable)**。可解释性可能很难严格定义，但是核心思想是作为人类，我们应该知道为什么我们的算法得出他们得出的结论。由于逻辑回归的特征通常是人为设计的，因此了解分类器决策的一种方法是了解每个特征在决策中所扮演的角色。逻辑回归可以与统计检验(似然比率检验或 Wald 检验)结合使用；通过这些测试之一调查某个特定特征是否很重要，或者检查其大小(与该特征相关联的权重 w 多大？)可以帮助我们解释分类器为何做出决策。这对于建立透明模型非常重要。

此外，除了将其用作分类器外，NLP 和许多其他领域的逻辑回归还广泛用作分析工具，以测试有关各种解释变量(特征)的影响的假设。在文本分类中，也许我们想知道逻辑上否定的词(不、不是、从不)更可能与否定的情感相关联，或者电影的否定评论更可能讨论电影摄影。但是，这样做有必要控制潜在的混杂因素：其他可能影响情感的因素(电影体裁、制作年份或评论的长度)。或者，我们可能正在研究 NLP 提取的

语言特征与非语言结果(再次入院，政治结果或产品销售)之间的关系，但需要控制混杂因素(患者的年龄、县区的投票、产品的品牌)。在这种情况下，逻辑回归允许我们测试某些特征是否与某些结果相关，并且超出了其他特征的影响。

5.8. 高级专题：推导梯度公式

在本节中，我们为逻辑回归给出了交叉熵损失函数 L_{CE} 的梯度的推导。

让我们从一些快速的演算复习开始[1]。

5.8.1. 复习

首先，ln(x)的导数：

$$\frac{d}{dx}\ln(x) = \frac{1}{x} \tag{5.38}$$

第二，sigmoid 的(非常优雅的)导数:

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z)) \tag{5.39}$$

最后，导数的**链式规则(chain rule)**。假设我们正在计算复合函数 f(x)= u(v(x))的导数。f(x)的导数是 u(x)关于 v(x)的导数乘以 v(x)关于 x 的导数：

$$\frac{df}{dx} = \frac{du}{dv}\cdot\frac{dv}{dx} \tag{5.40}$$

5.8.2. 推导

首先，我们想知道损失函数相对于单个权重 w_j 的导数(我们需要针对每个权重和偏差来计算它)：

$$\begin{aligned}\frac{\partial L_{\text{CE}}}{\partial w_j} &= \frac{\partial}{\partial w_j} -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))] \\ &= -\left[\frac{\partial}{\partial w_j}y\log\sigma(w\cdot x+b)+\frac{\partial}{\partial w_j}(1-y)\log[1-\sigma(w\cdot x+b)]\right]\end{aligned} \tag{5.41}$$

接下来，利用链式规则，借助对数的导数:

$$\frac{\partial L_{\text{CE}}}{\partial w_j} = -\frac{y}{\sigma(w\cdot x+b)}\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) - \frac{1-y}{1-\sigma(w\cdot x+b)}\frac{\partial}{\partial w_j}1-\sigma(w\cdot x+b) \tag{5.42}$$

重新整理:

$$\frac{\partial L_{\text{CE}}}{\partial w_j} = -\left[\frac{y}{\sigma(w\cdot x+b)}-\frac{1-y}{1-\sigma(w\cdot x+b)}\right]\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) \tag{5.43}$$

现在，插入 Sigmoid 的导数，再使用链式规则一次，得出公式 5.44：

[1] 译者注：“5.8.1 复习，5.8.2 推导”是译者自己添加的三级标题，以便增加可读性。

$$\begin{aligned}\frac{\partial L_{\text{CE}}}{\partial w_j} &= -\left[\frac{y-\sigma(w\cdot x+b)}{\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]}\right]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]\frac{\partial(w\cdot x+b)}{\partial w_j} \\ &= -\left[\frac{y-\sigma(w\cdot x+b)}{\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]}\right]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]x_j \\ &= -[y-\sigma(w\cdot x+b)]x_j \\ &= [\sigma(w\cdot x+b)-y]x_j \end{aligned} \tag{5.44}$$

5.9. 总结

本章介绍了**分类**的**逻辑回归**模型。

•逻辑回归是一种受监督的机器学习分类器，可从输入中提取实值特征，将每个特征与权重相乘，对其求和，然后将总和传递给 **Sigmoid** 函数以生成概率。阈值用于做出决定。

•逻辑回归可以用于两个类别(例如，正面情感和负面情感)，也可以用于多个类别(**多元逻辑回归**，例如用于 n 元(n-ary)文本分类，词类标注等)。

•多元逻辑回归使用 **softmax** 函数计算概率。

•权重(向量 w 和偏差 b)是从标记的训练集中通过损失函数(如**交叉熵损失**)学习的，必须将其最小化。

•最小化此损失函数是一个**凸优化**问题，并且使用诸如**梯度下降**的迭代算法来找到最佳权重。

•**正则化**用于避免**过拟合**。

•逻辑回归也是最有用的分析工具之一，因为它能够透明地研究各个特征的重要性。

5.10. 文献和历史说明

逻辑回归是在统计学领域发展起来的，它在 20 世纪 60 年代被用于二进制数据的分析，在医学中尤其常见(Cox, 1969)。从 20 世纪 70 年代后期开始，作为语言变异研究的正式基础之一，它在语言学中得到了广泛的应用(Sankoff 和 Labov, 1979)。

然而，逻辑回归直到 20 世纪 90 年代才在自然语言处理中变得普遍，当时它似乎同时出现在两个方向：第一个是邻近的信息检索，第二个是语音处理领域。这两个方向都使用了回归，都为自然语言处理提供了许多其他的统计技术。实际上，很早就将逻辑回归用于文档传送是最早使用(LSI)嵌入作为单词表示的 NLP 应用之一(Schutze 等人 1995)。

与此同时，在 20 世纪 90 年代早期，IBM 研究中心以最大熵模型或 maxent (Berger 等人，1996)的名义开发了逻辑回归并应用于自然语言处理，似乎独立于统计文献。在这个名称下，它被应用于语言建模(Rosenfeld, 1996)、词类标注(Ratnaparkhi, 1996)、解析(Ratnaparkhi, 1997)、关联解析(Kehler, 1997b)和文本分类(Nigam 等人 1999)。

更多关于分类的内容可以在机器学习教科书中找到(Hastie 等人 2001, Witten 和 Frank 2005, Bishop 2006, Murphy 2012)。

5.11. 练习

(空缺)