【SLP3-I5】Logistic回归

0. 绪论

“您怎么知道这些美丽的秋海棠并不同等重要呢？”
"And how do you know that these fine begonias are not of equal importance?"

——埃尔克里·普瓦洛(阿加莎·克里斯蒂的小说《斯泰尔斯庄园奇案》中的人物)
Hercule Poirot, in Agatha Christie's *The Mysterious Affair at Styles*

侦探小说中充斥着线索，就像文本中充斥着单词一样。然而，对于阅读能力不足的读者来说，要知道如何衡量作者的线索以完成关键的分类任务——判定出谁是真凶——是颇具挑战性的。

在本章中，我们将介绍一种非常适用于发现特征或线索与某些特定结果之间的联系的算法：**Logistic回归(logistic regression)** 。事实上，Logistic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，Logistic回归是用于分类的baseline监督机器学习算法，也与神经网络有着非常密切的关系。正如我们将在【SLP3-I7】中所见，神经网络可被视为一系列堆叠在一起的Logistic回归分类器。因此，这里介绍的分类和机器学习技术将在整本书中发挥重要作用。

Logistic回归可用于将一个观测值分入两个(如“积极情感”和“消极情感”)或多个类别中的一类。由于二分类情形在数学上较为简单，我们将在接下来的几节中首先描述Logistic回归的这种特殊情形，然后在第3节中简要总结 **多项Logistic回归(multinomial logistic regression)** 在两个以上类别情形下的使用。

在接下来的几节中，我们将介绍Logistic回归的数学原理。但是，让我们先从一些高级问题谈起。

- **生成式(generative)和判别式(discriminative)分类器(classifier)** ：朴素贝叶斯与Logistic回归最重要的区别在于，Logistic回归是 **判别式** 分类器，而朴素贝叶斯是 **生成式** 分类器。

  这是两种截然不同的构建机器学习模型的框架。考虑这样一个视觉隐喻：想象我们正在试图区分狗和猫的图像。生成式模型的目标是理解狗和猫的外观。你甚至可以要求这样的模型去“生成“——或者说画出——一条狗。给定一幅测试图像，系统会询问是猫模型还是狗模型更好地拟合了该图像(而对图像没那么惊讶)，并选出二者之一作为它的标签。

  相比之下，判别式模型只是试图学习区分类别(也许对这些类别的具体情况了解得都不多)。因此，或许训练数据中所有的狗都戴着项圈，而猫没有。如果用这一个特征就能很好地区分类别，那么模型就满意了。如果你问这样的模型它对猫了解多少，它只能说猫不戴项圈。

  更正式地讲，回想一下，朴素贝叶斯并不是通过直接计算$P(c|d)$来为文档d分配类别c的，而是通过计算似然估计和先验概率

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{似然估计}} \overbrace{P(c)}^{\text{先验概率}} \tag{1}$$

  像朴素贝叶斯这样的 **生成式模型** 利用了这个 **似然估计(likelihood)** 项，它表达了“在我们已知文档属于类别c的前提下”，如何生成文档的特征。

  相比之下，在文本分类场景中， **判别式模型** 试图**直接**计算$P(c|d)$。也许它会学习为文档的某些特征分配高权重，这些特征能直接提高它在可能类别之间进行“区分”的能力，即使它无法生成其中某个类别的示例。

- **概率机器学习分类器(probabilistic machine learning classifier)的组成部分** ：与朴素贝叶斯一样，Logistic回归也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一个包含m个输入/输出对$(x^{(i)}, y^{(i)})$的训练语料库(我们将使用带括号的上标来指代训练集中的单个实例——对于情感分类，每个实例可能是一个待分类的单独文档)。一个用于分类的机器学习系统有四个组成部分：

1. 输入的 **特征表示** 。对于每个输入观测值$x^{(i)}$，这将是一个特征向量$[x_1, x_2, \ldots, x_n]$。我们通常将输入$x^{(j)}$的特征i表示为$x_i^{(j)}$，有时简化为x_i，但我们也会看到f_i、$f_i(x)$或多分类情形下的$f_i(c, x)$等表示法。
2. 分类函数。它通过$p(y|x)$计算估计的类别$\hat{y}$。在下一节中，我们将介绍用于分类的 **sigmoid** 和 **softmax** 工具。
3. 用于学习的目标函数。通常涉及最小化训练样本上的误差。我们将引入 **交叉熵损失函数 (cross-entropy loss function)** 。
4. 优化目标函数的算法。我们将介绍 **随机梯度下降(stochastic gradient descent)** 算法。

- Logistic回归分为两个阶段:
  - **训练(training)**： 我们使用随机梯度下降和交叉熵损失来训练系统(具体来说是权重w和b)。
  - **测试(test)**： 给定一个测试样本x，我们计算$p(y|x)$并返回概率较高的标签$y = 1$或$y = 0$。

1. sigmoid函数

二元Logistic回归的目标是训练一个分类器，使其能够对新输入观测值的类别做出二元决策。在这里，我们引入 **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，我们用特征向量$[x_1, x_2, \ldots, x_n]$去表示它(我们将在下一小节中展示样本特征)。分类器输出y可以为1(表示观测值是该类别的成员)或0(表示观测值不是该类别的成员)。我们想知道这个观测值是某一类别的成员的概率$P(y = 1|x)$。因此，假定要在“积极情感”和“消极情感”之间做出决策，这些特征值表示文档中单词的计数，$P(y = 1|x)$是文档带有积极情感的概率，而$P(y = 0|x)$是文档带有消极情感的概率。

Logistic回归通过从训练集中学习 **权重** 向量和 **偏置项(bias term)** 来解决此问题。每个权重w_i都是一个实数，且与输入特征之一x_i相关联。权重w_i表示输入特征对分类决策的重要程度，可以为正值(提供证据表明待分类的实例属于积极类)，也可以为负值(提供证据表明待分类的实例属于消极类)。因此，在情感分类任务中，我们可能期望单词“awesome”具有很高的正权重，而“abysmal”则具有很高的负权重。 **偏置项** ，也称为 **截距(intercept)** ，是另一个被添加到加权输入中的实数。

在我们了解了训练中的权重之后，为了在测试实例上做出决策，分类器首先将每个x_i乘以它的权重w_i，对加权后的特征值求和，然后加上偏置项b。所得的单个数值z表示该类别证据的加权和。

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \tag{2}$$

在本书的其余部分，我们将使用线性代数中的 **点乘(dot product)** 符号来表示这样的和。两个向量a和b的点积(记作$\mathsf{a} \cdot \mathsf{b}$)是每个向量对应元素的乘积之和(注意，我们这里使用粗体符号b来表示向量)。因此，下式是式(2)的等价形式:

$$z = \mathsf{w} \cdot \mathsf{x} + b \tag{3}$$

但要注意，式(3)中没有任何东西能强制z成为合法概率，即介于0和1之间。事实上，由于权重可取任意实数值，输出甚至可能是负数；也就是说，z的取值范围为$(-\infty, \infty)$。

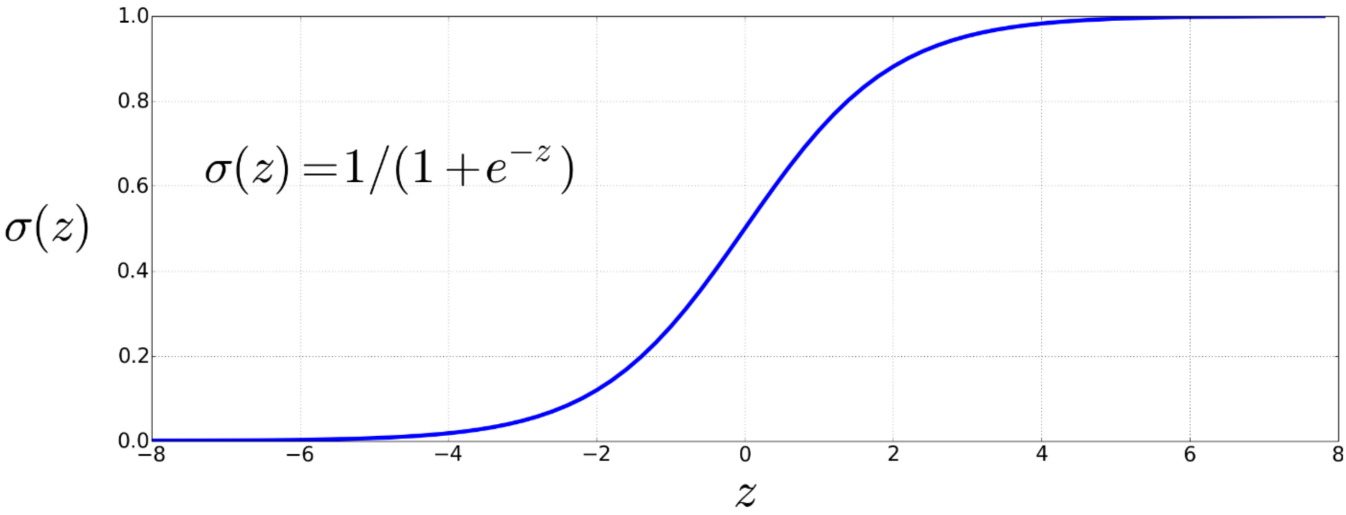


图1 sigmoid函数$\sigma(z)=\frac{1}{1+e^{-z}}$取一个实数值$z$. 将其映射到值域$(0,1)$内。它在0附近几乎是线性的，但是离群值被压向0或1。

为了产生概率，我们将z传入 **sigmoid** 函数$\sigma(z)$。sigmoid函数(本义是“S型函数”，因其函数图象形似字母“S”而得名)又被称为 **Logistic函数** ，Logistic回归由此得名。如图1所示，sigmoid函数的解析式如下：

$$\sigma(z)=\frac{1}{1+e^{-z}}=\frac{1}{1+\exp(-z)} \tag{4}$$

(在本书的其余部分，我们将用记号$\exp(x)$来指代e^x。)

当用于Logistic回归时，sigmoid函数具有许多优点。它将实数值映射到$(0,1)$的范围内，而这正是我们所希望得到的合法概率。由于其S形曲线在0附近几乎是线性的，但在两端趋于平缓，因此它倾向于将离群值压向0或1。此外，sigmoid函数是可微的，这会使它学习起来更简便，正如我们将在第10节中看到的那样。

我们说得差不多了。如果我们将sigmoid函数应用于加权特征值之和，我们就会得到一个介于0和1之间的数值。要使其成为合法概率，我们只需确保$p(y=1)$与$p(y=0)$之和为1即可。我们可以进行如下的操作：

$$\begin{aligned} P(y=1) &= \sigma(\mathsf{w}\cdot\mathsf{x}+b) \\ &= \frac{1}{1+\exp(-(\mathsf{w}\cdot\mathsf{x}+b))} \\ P(y=0) &= 1-\sigma(\mathsf{w}\cdot\mathsf{x}+b) \\ &= 1-\frac{1}{1+\exp(-(\mathsf{w}\cdot\mathsf{x}+b))} \\ &= \frac{\exp(-(\mathsf{w}\cdot\mathsf{x}+b))}{1+\exp(-(\mathsf{w}\cdot\mathsf{x}+b))} \end{aligned} \tag{5}$$

sigmoid函数具有如下性质：

$$1-\sigma(x)=\sigma(-x) \tag{6}$$

因此我们也可以将$P(y=0)$表示为$\sigma(-(\mathsf{w}\cdot\mathsf{x}+b))$。

2. 使用Logistic回归进行分类

现在，上一节引入的sigmoid函数为我们提供了一种给定实例x时计算概率$P(y=1|x)$的方法。

我们如何对测试实例x归属于哪一类别做出决策？对于给定的x，如果概率$P(y=1|x)$的值大于

0.5，我们就认为x属于该类；否则就认为不属于。我们称0.5为 **决策边界(decision boundary)** 值：

$$\text{decision}(x) = \begin{cases} 1, & \text{当} P(y=1|x) > 0.5 \text{时} \\ 0, & \text{其他} \end{cases}$$

下面，让我们来举例说明Logistic回归作为分类器在语言任务中的应用。

2.1. 情感分类

假设我们正在对电影评论文本进行二元情感分类，我们想知道是将情感类别“+”(积极)还是“-”(消极)分配给评论文档doc。我们将每个输入观测值都表示为下表中显示的6个输入特征值$x_1 \dots x_6$。图2显示了微型测试文档样本中的特征。

特征变量	定义	图2中的取值
x_1	doc中积极词汇的计数	3
x_2	doc中消极词汇的计数	2
x_3	$\begin{cases} 1, & \text{当} doc \text{包含单词“no”时} \\ 0, & \text{其他} \end{cases}$	1
x_4	doc中第一和第二人称代词的计数	3
x_5	$\begin{cases} 1, & \text{当} doc \text{包含标点符号“!”时} \\ 0, & \text{其他} \end{cases}$	0
x_6	$\ln(doc\text{的总单词数})$	$\ln(66) = 4.19$

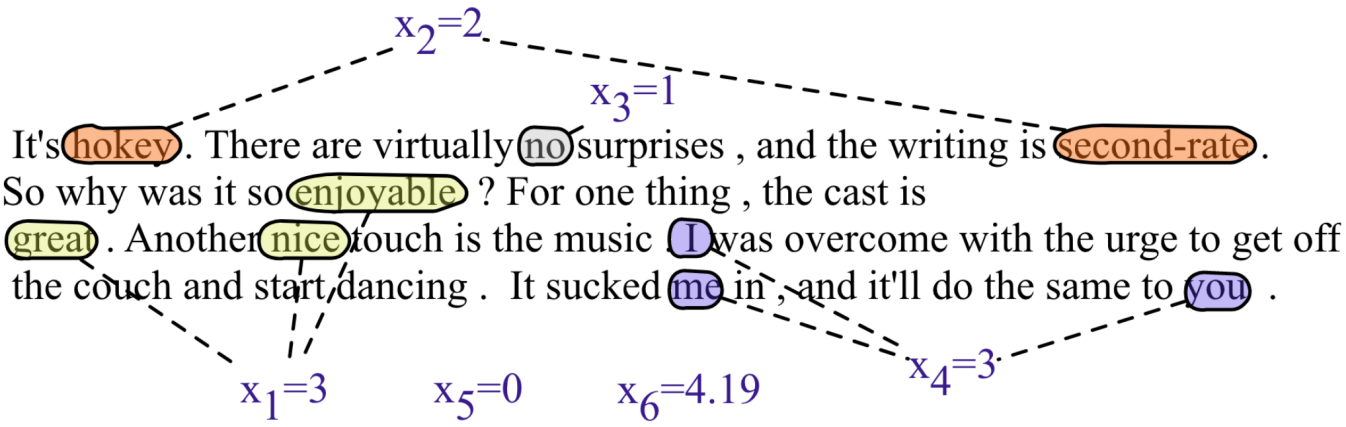


图2 一个微型测试文档样本展示了向量x中提取的特征。

假设我们已经通过学习为这其中每一个特征都分配了一个实值权重，这6个权值$[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$分别对应于6个特征，且$b = 0.1$。(我们将在下一节讨论权值是如何通过学习得到的。)例如，权重w_1表示积极词汇(如great、nice、enjoyable等)的数量对于积极情感决策的重要程度，而w_2则告诉我们消极词汇的重要性。注意到$w_1 = 2.5$是正数而$w_2 = -5.0$为负，这意味着消极词汇与积极情感决策呈负相关，并且大约是积极词汇的两倍重要。

给定这6个特征值以及输入评论x，代入式(5)便可计算出$P(+|x)$和$P(-|x)$的结果：

$$\begin{aligned} p(+|x) = P(y=1|x) &= \sigma(\mathsf{w}\cdot\mathsf{x}+b) \\ &= \sigma([2.5,-5.0,-1.2,0.5,2.0,0.7]\cdot[3,2,1,3,0,4.19]+0.1) \\ &= \sigma(0.833) \\ &= 0.70 \\ p(-|x) = P(y=0|x) &= 1-\sigma(\mathsf{w}\cdot\mathsf{x}+b) \\ &= 0.30 \end{aligned} \tag{7}$$

2.2. 其他分类任务和特征

Logistic回归常用于各种NLP任务，输入的任何属性都可以成为特征。考虑 **句点消歧(period disambiguation)** 的任务：通过将每个句点分入EOS(end-of-sentence，句尾)和not-EOS(非句尾)两类之一，确定句点是句子的结尾还是单词的一部分。我们可能会使用像下面的x_1这样的特征来表示当前单词是小写(可能具有正权重)，或者当前单词(如“Prof.”)已被收录于我们的缩写词典(AcronymDict)中(可能具有负权重)。特征也可以表达相当复杂的属性组合。例如，紧跟在首字母大写单词后面的句点很可能是EOS，但如果该单词本身是“St.”，并且前一个单词的首字母大写，则句点可能是单词“street”缩写形式的一部分。

$$\begin{aligned} x_1 &= \begin{cases} 1, & \text{当}Case(w_i)=\text{Lower}\text{时} \\ 0, & \text{其他} \end{cases} \\ x_2 &= \begin{cases} 1, & \text{当}w_i \in \text{AcronymDict}\text{时} \\ 0, & \text{其他} \end{cases} \\ x_3 &= \begin{cases} 1, & \text{当}w_i=\text{St.}\text{且}Case(w_{i-1})=\text{Cap}\text{时} \\ 0, & \text{其他} \end{cases} \end{aligned}$$

- **设计特征**：特征通常是通过仔细观察训练集并结合语言直觉和领域语言文献来设计的。对系统早期版本的训练集或开发集进行仔细的错误分析通常能够提供对特征的深入了解。

  对于某些任务，构建由更原始特征组合而成的复杂特征特别有用。我们在上面看到了这样一个用于句点消歧的特征，即如果前一个单词的首字母大写，则“St.”这个单词中的句点不太可能是句子的结尾。对于Logistic回归和朴素贝叶斯，这些组合特征或 **特征交互(feature interactions)** 必须手动设计。

  对于许多任务(特别是当特征值可以引用特定单词时)，我们需要大量的特征。这些特征通常是通过 **特征模板(feature templates)** (特征的抽象规范)自动创建的。例如，句点消歧的Bigram模板可能会为训练集中每个句点前出现的单词对创建一个特征。因此，这里的特征空间是稀疏的，因为我们只有在训练集中存在n-gram的位置才需要创建一个特征。一般是根据字符串描述将特征创建为哈希(hash)。用户对特征的描述(例如“bigram(American breakfast)”)被哈希为唯一的整数i[1]，成为特征f_i的编号(下标)。

  为了避免特征设计耗费大量人力，近期的NLP研究聚焦于 **表示学习(representation learning)** ——以无监督的方式从输入中自动学习特征的方法。我们将在【SLP3-I6】和【SLP3-I7】中介绍表示学习的方法。

- **缩放(scaling)输入特征**：当不同的输入特征具有极为不同的取值范围时，通常会对它们进行重新缩放，使它们具有可比较的取值范围。我们可以对输入值进行 **标准化处理(standardize)**，经过处理的输入值符合标准正态分布(均值为0，标准差为1)，这一变换有时也被称为 **Z-Score标准化**。也就是说，如果μ_i是输入数据集中m个观测值的特征x_i的平均值，σ_i是这些特征值的标准差，我们可

以用新的特征x_i'替换原来的特征x_i，计算方法如下：

$$\begin{aligned}\mu_i &= \frac{1}{m}\sum_{j=1}^{m} x_i^{(j)} \\ \sigma_i &= \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(\mathsf{x}_i^{(j)} - \mu_i\right)^2} \\ \mathsf{x}_i' &= \frac{\mathsf{x}_i - \mu_i}{\sigma_i}\end{aligned} \tag{8}$$

此外，我们也可以对输入特征值进行 **归一化处理(normalize)** ，使结果值映射到0和1之间：

$$\mathsf{x}_i' = \frac{\mathsf{x}_i - \min(\mathsf{x}_i)}{\max(\mathsf{x}_i) - \min(\mathsf{x}_i)} \tag{9}$$

使输入数据具有可比较的取值范围在比较不同特征的值时非常有用。数据缩放对于大型神经网络尤其重要，因为它有助于加速梯度下降。

2.3. 同时处理多个样本

我们已经给出了单样本Logistic回归方程。但在实际应用中，我们当然希望能够处理一个包含大量样本的完整测试集。假设我们有一个由m个测试样本组成的测试集，我们希望对每个测试样本进行分类。我们将继续使用绪论中定义的符号，其中带括号的上标值表示某组数据集(用于训练或测试)中的样本索引。因此，在这种情况下，每个测试样本$x^{(i)}$都有一个特征向量$\mathsf{x}^{(i)}$，$1 \leqslant i \leqslant m$。(和往常一样，我们用粗体表示向量和矩阵。)

计算每个输出值$\hat{y}^{(i)}$的一种方法就是使用for循环，一次计算一个测试样本：

$$\begin{gathered}\textbf{foreach}\ \ x^{(i)}\ \ \text{in input}\ [x^{(1)}, x^{(2)}, \ldots, x^{(m)}] \\ y^{(i)} = \sigma(\mathsf{w} \cdot \mathsf{x}^{(i)} + b)\end{gathered} \tag{10}$$

因此，对于前三个测试样本，我们将分别计算预测的$\hat{y}^{(i)}$如下：

$$\begin{aligned}P(y^{(1)} = 1 | x^{(1)}) &= \sigma(\mathsf{w} \cdot \mathsf{x}^{(1)} + b) \\ P(y^{(2)} = 1 | x^{(2)}) &= \sigma(\mathsf{w} \cdot \mathsf{x}^{(2)} + b) \\ P(y^{(3)} = 1 | x^{(3)}) &= \sigma(\mathsf{w} \cdot \mathsf{x}^{(3)} + b)\end{aligned}$$

但事实证明，我们可以稍微修改我们的原始方程式(5)，以更有效地完成这项工作。我们将使用矩阵运算一次性为所有样本分配类别！

首先，我们将每个输入x的所有输入特征向量打包到一个输入矩阵X中，其第i行就是一个由输入样本$x^{(i)}$的特征向量组成的行向量(即向量$\mathsf{x}^{(i)}$)。假设每个样本都有f个特征和权重，那么X将是一个形状为$[m \times f]$的矩阵，如下所示：

$$\mathsf{X} = \begin{bmatrix} \mathsf{x}_1^{(1)} & \mathsf{x}_2^{(1)} & \ldots & \mathsf{x}_f^{(1)} \\ \mathsf{x}_1^{(2)} & \mathsf{x}_2^{(2)} & \ldots & \mathsf{x}_f^{(2)} \\ \mathsf{x}_1^{(3)} & \mathsf{x}_2^{(3)} & \ldots & \mathsf{x}_f^{(3)} \\ \ldots & & & \end{bmatrix} \tag{11}$$

现在，如果我们引入一个维数为m的向量b，它由标量偏置项b重复m次组成，即$\mathsf{b} = [b, b, \ldots, b]$，并且$\hat{\mathsf{y}} = [\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(m)}]$作为输出向量(每个输入$x^{(i)}$及其特征向量$\mathsf{x}^{(i)}$都对应一个标量输出$\hat{y}^{(i)}$)，并将权重向量$\mathsf{w}$表示为列向量，我们可以通过一次矩阵乘法和一次加法计算所有输出：

$$\mathsf{y} = \mathsf{X}\mathsf{w} + \mathsf{b} \tag{12}$$

你应当确信式(12)的计算结果与我们在式(10)中的for循环相同。例如，输出向量y的第一项$\hat{y}^{(1)}$将被正确地计算为：

$$\hat{y}^{(1)} = [\mathsf{x}_1^{(1)}, \mathsf{x}_2^{(1)}, \ldots, \mathsf{x}_f^{(1)}] \cdot [\mathsf{w}_1, \mathsf{w}_2, \ldots, \mathsf{w}_f] + b \tag{13}$$

注意，我们必须将X和w在式(5)中出现的顺序重新排列，以使乘法正确进行。这里再次展示式(12)，这一次标注了各矩阵或向量的形状：

$$\underset{(m\times 1)}{\mathsf{y}} = \underset{(m\times f)(f\times 1)}{\mathsf{X}}\ \underset{}{\mathsf{w}} + \underset{(m\times 1)}{\mathsf{b}} \tag{14}$$

现代编译器和计算硬件可以非常高效地计算这个矩阵运算，使得计算速度更快，这对于在非常大的数据集上进行训练或测试是非常重要的。

2.4. 分类器的选择

相比于朴素贝叶斯，Logistic回归具有许多优势。朴素贝叶斯具有过于强的条件独立性假设。考虑两个强相关的特征；或者更彻底一些，想象我们只是将同一个特征f_1加了两次。朴素贝叶斯会将f_1的两个副本都视为独立的，将它们相乘，从而高估了证据。相比之下，Logistic回归应对相关特征时更加健壮。如果两个特征f_1和f_2完全相关，Logistic回归将简单地将部分权重分配给w_1，部分权重分配给$w2$。因此，当存在大量相关特征时，Logistic回归分配的概率将比朴素贝叶斯更加准确。所以，Logistic回归通常在较大的文档或数据集上表现得更好，并且是常见的默认选择。

尽管得出的概率不见得准确，朴素贝叶斯仍能经常做出正确的分类决策。并且，朴素贝叶斯在非常小的数据集[2]和短文档[3]上可以表现得非常好(有时甚至比Logistic回归更好)。此外，朴素贝叶斯易于实现且训练速度非常快(没有优化步骤)。因此，在某些情况下使用它仍是一个合理的选择。

3. 多项Logistic回归

有时我们需要超过两个类别。也许我们想进行三向(3-way)情感分类(积极、消极和中立)。或者我们可能要去分配【SLP3-I8】中将要介绍的一些标签，例如单词的词性(从10、30甚至50种不同的词性中选择)，或短语的命名实体类型(从人、地点、组织等标签中选择)。

在这种情况下，我们使用 **多项Logistic回归** ，也称为 **softmax回归** (在早期的NLP文献中，有时会看到“ **maxent分类器** ”的称法)。在多项Logistic回归中，我们希望为每个观测值标记一个类别k，k是从设定的K个类别中选出来的，且规定其中仅有一个类是正确的[有时称为 **硬分类(hard classification)** ，即观测值不能属于多个类]。让我们使用以下表示方法：每个输入x对应的输出y都将是维数为K的向量。如果类c是正确的类，则令$\mathsf{y}_c = 1$，并将y的所有其他分量设置为0，即$\begin{cases}\mathsf{y}_c = 1 \\ \mathsf{y}_j = 0 \quad \forall j \neq c\end{cases}$。像$\mathsf{y}$这样只有一个分量的值为1、其余分量的值都为0的向量被称为 **one-hot向量** 。分类器的任务就是产生估计向量$\hat{\mathsf{y}}$。对于每个类k，$\hat{\mathsf{y}}_k$的值就是分类器对概率$p(\mathsf{y}_k = 1|\mathsf{x})$的估计。

3.1. softmax函数

多项Logistic回归计算$p(\mathsf{y}_k = 1|\mathsf{x})$使用的是sigmoid函数的一个推广，称为 **softmax** 函数。softmax函数将向量$\mathsf{z} = [\mathsf{z}_1,\mathsf{z}_2,\ldots,\mathsf{z}_K]$的$K$个任意值分量映射到概率分布上，使得每个分量值都落入$[0,1]$的范围内，并且所有分量值之和为1。与sigmoid函数一样，softmax函数也是一个指数函数。

对于K维向量z，softmax函数定义如下：

$$\text{softmax}(\mathsf{z}_i) = \frac{\exp(\mathsf{z}_i)}{\sum_{j=1}^{K}\exp(\mathsf{z}_j)} \quad 1 \leqslant i \leqslant K \tag{15}$$

输入向量$\mathsf{z}=[\mathsf{z}_1,\mathsf{z}_2,\ldots,\mathsf{z}_K]$的softmax函数值本身也是一个向量：

$$\text{softmax}(\mathsf{z})=\left[\frac{\exp(\mathsf{z}_1)}{\sum_{i=1}^{K}\exp(\mathsf{z}_i)},\frac{\exp(\mathsf{z}_2)}{\sum_{i=1}^{K}\exp(\mathsf{z}_i)},\ldots,\frac{\exp(\mathsf{z}_K)}{\sum_{i=1}^{K}\exp(\mathsf{z}_i)}\right] \tag{16}$$

分母$\sum_{i=1}^{K}\exp(\mathsf{z}_i)$用于将所有分量值归一化为合法概率。举例来说，现给出一个向量

$$\mathsf{z}=[0.6,\,1.1,\,-1.5,\,1.2,\,3.2,\,-1.1]$$

则$\text{softmax}(\mathsf{z})$的计算结果(保留有效数字)为：

$$[0.055,0.090,0.006,0.099,0.74,0.010]$$

与sigmoid函数一样，softmax函数也具有将离群值压向0或1的性质。因此，如果输入之一大于其他输入，softmax函数倾向于将其概率推向1，同时抑制较小输入的概率。

3.2. 在Logistic回归中应用softmax函数

当我们将softmax函数应用于Logistic回归时，输入将(与sigmoid函数一样)是权重向量w与输入向量x的点积(再加上偏置项b)。但是现在我们需要为K个类别中的每一个分别设置权重向量w_k和偏置项b_k。因此，我们的每一个输出类别$\hat{y}_k$的概率可如下计算得到：

$$p(\mathsf{y}_k=1|\mathsf{x})=\frac{\exp(\mathsf{w}_k\cdot\mathsf{x}+b_k)}{\sum_{j=1}^{K}\exp(\mathsf{w}_j\cdot\mathsf{x}+b_j)} \tag{17}$$

式(17)的形式似乎表明我们将单独计算每个输出。相反，更为常见的是通过现代向量处理硬件来建立方程以实现更高效的计算。我们通过将K维权重向量表示为权重矩阵W和偏置向量b来实现这一点。W的第k行对应于权重向量w_k，故W是一个$K\times f$矩阵，其中K为输出类别数，f为输入特征数。偏置向量b对于K个输出类别中的每一个都有一个对应值。如果我们以这种方式表示权重，我们可以通过一个简洁的算式计算出K个类别中每个类别的输出概率向量$\hat{\mathsf{y}}$：

$$\hat{\mathsf{y}}=\text{softmax}(\mathsf{W}\mathsf{x}+\mathsf{b}) \tag{18}$$

如果你仔细完成矩阵运算，你会发现第一个输出类别$\hat{y}_1$的估计分数(在我们取softmax函数值之前)将正确地变为$\mathsf{w}_1\cdot\mathsf{x}+b_1$。

图3直观地展示了权重向量与权重矩阵在计算二元与多项Logistic回归的输出类别概率时所起到的作用。

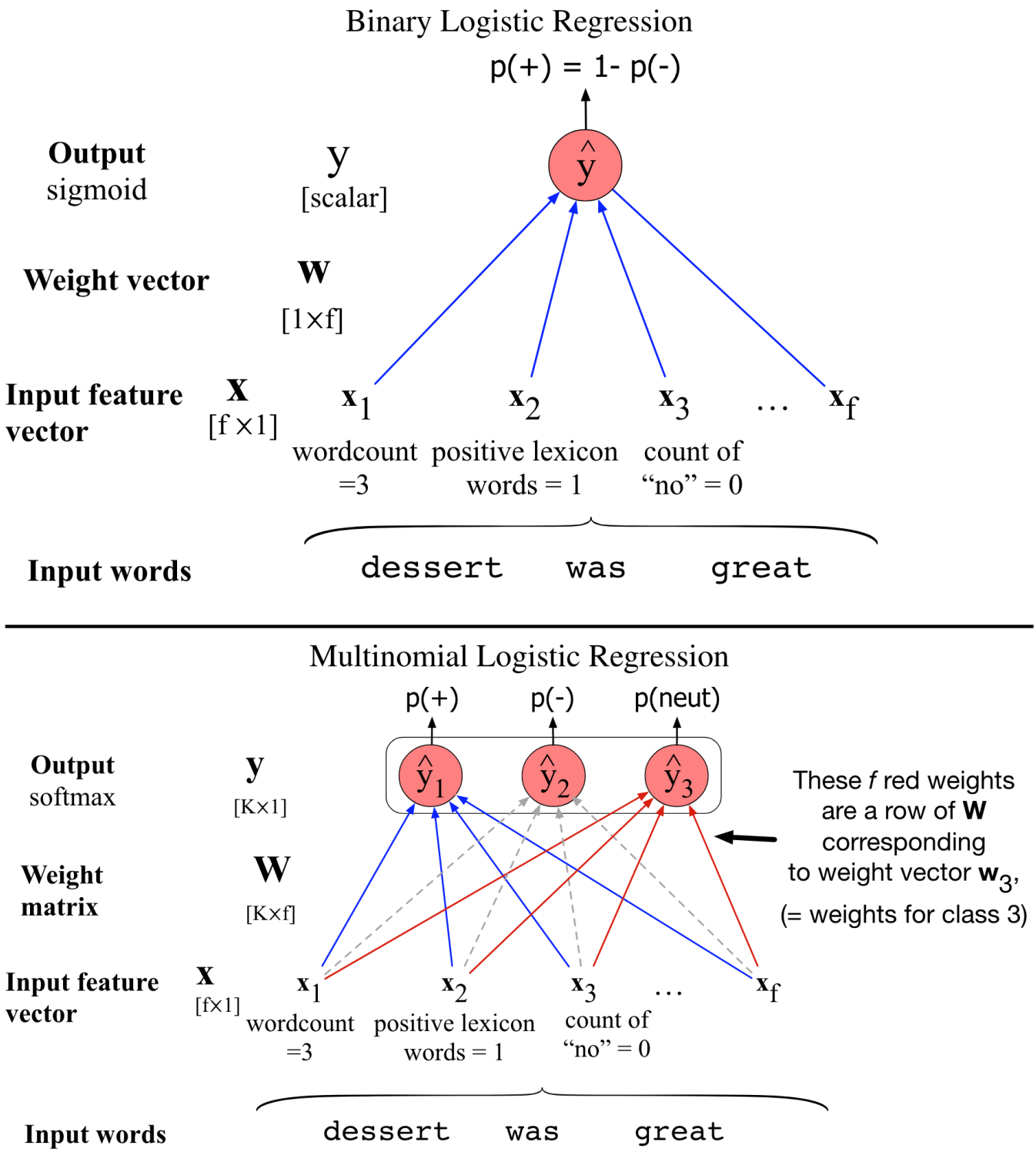


图3　二元与多项Logistic回归。二元Logistic回归使用单个权重向量w，并具有标量输出$\hat{y}$。在多项Logistic回归中我们有K个单独的权重向量，分别对应于K个类别，所有这些向量都被打包到一个权重矩阵W中，并且具有向量输出$\hat{\mathbf{y}}$。

3.3. 多项Logistic回归中的特征

多项Logistic回归中的特征类似于二元Logistic回归，不同之处在于我们需要为K类中的每一类分别设置权重向量和偏差项。回想一下我们在2.1小节中提到的二元感叹号特征x_5：

$$x_5 = \begin{cases} 1, & \text{当}doc\text{包含标点符号“!”时} \\ 0, & \text{其他} \end{cases}$$

在二分类中，特征上的正权重w_5会影响分类器趋向$y=1$(积极情感)，而负权重会影响其趋向$y=0$(消极情感)，权重的绝对值表示该特征的重要程度。相比之下，在多项Logistic回归中，每个类别都有

单独的权重，一个特征可以作为每个类别的证据或反证。

例如，在三向多类别情感分类中，我们必须为每个文档分配三个类别[+、-或0(中性)]中的一个。现在，与感叹号相关的特征可能对0文档有负权重，而对+或-文档有正权重：

特征	定义	$w_{5,+}$	$w_{5,-}$	$w_{5,0}$
$f_5(x)$	$\begin{cases}1, & \text{当}doc\text{包含标点符号“!”时}\\0, & \text{其他}\end{cases}$	3.5	3.1	−5.3

由于这些特征权重既依赖于输入文本，也依赖于输出类别，我们有时会明确表示这种依赖关系，并将特征本身表示为$f(x,y)$——一个既依赖于输入又依赖于类别的函数。使用这种符号，上面的$f_5(x)$可以用三个特征$f_5(x,+)$、$f_5(x,-)$和$f_5(x,0)$来表示，每个特征都有一个权重。我们将在【SLP3-I8】中描述CRF时使用这种符号。

4. Logistic回归中的学习

如何学习模型的参数(权重w和偏置项b)？Logistic回归是监督分类的一个实例，在其中我们知道每个观测值x的正确标签y(0或1)。系统通过式(5)产生的是$\hat{y}$，即系统对真实y的估计。我们希望学习参数(w和b)，使得每个训练观测值的$\hat{y}$尽可能接近真实的y。

这就需要用到我们在本章绪论中预告过的两个组成部分(组件)了。其一是一个度量标准，衡量当前标签($\hat{y}$)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性，而是与之相反的，系统输出与黄金输出之间的距离(差异、差距)。我们称这种距离为 **损失(loss)** 函数或 **成本函数(cost function)** 。在下一节中，我们将介绍常用于Logistic回归和神经网络的损失函数—— **交叉熵损失** 。

其二是一个优化算法，用于迭代更新权重，以最小化此损失函数。对此的标准算法是 **梯度下降** 算法；我们将在随后的一节中介绍 **随机梯度下降** 算法。

我们将在接下来的两节中描述这些算法，用于更简单的二元Logistic回归情形，然后在第8节中转向多项Logistic回归情形。

5. 交叉熵损失函数

对于一个观测值x，我们需要用一个函数来表示分类器输出$\hat{y}=\sigma(\mathsf{w}\cdot\mathsf{x}+b)$与正确输出($y=0$或1)的接近程度。我们将其称为：

$$L(\hat{y},y)=\hat{y}\text{与真正的}y\text{相差多少} \tag{19}$$

我们通过一个损失函数来实现这一点，它倾向于让训练样本的类别标签“更有可能”被正确预测。这被称为 **条件最大似然估计(conditional maximum likelihood estimation)** ：我们选定参数w,b，使得在给定观测值x的情况下， **训练数据中真实标签y的对数概率最大化** 。由此产生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)** 。

让我们推导出这个损失函数，将其应用于单个观测值x。我们希望学习能够最大化正确标签概率$p(y|x)$的权重。由于只有两个离散结果(1或0)，这是一个伯努利分布，我们可以将我们的分类器为一个观测值产生的概率$p(y|x)$表示为以下形式(记住，当$y=1$时式(20)简化为$\hat{y}$，当$y=0$时式(20)简化为$1-\hat{y}$)：

$$p(y|x)=\hat{y}^y\left(1-\hat{y}\right)^{1-y} \tag{20}$$

现在我们取等式两边的对数。这在数学上会很方便，而且不会对我们造成损害；无论变量取什么值，在最大化概率的同时，也会最大化概率的对数：

$$\begin{aligned}\log p(y|x)&=\log\left[\hat{y}^y\left(1-\hat{y}\right)^{1-y}\right]\\&=y\log\hat{y}+(1-y)\log\left(1-\hat{y}\right)\end{aligned} \tag{21}$$

式(21)描述了一个应该被最大化的对数似然值。为了将其转化为损失函数(我们需要最小化的东西)，我们只需将式(21)中的符号翻转即可。得到的结果就是交叉熵损失L_{CE}：

$$L_{CE}\left(\hat{y}, y\right) = -\log p(y|x) = -\left[y \log \hat{y} + (1-y) \log\left(1-\hat{y}\right)\right] \tag{22}$$

最后，我们可以代入定义式$\hat{y} = \sigma(\mathsf{w} \cdot \mathsf{x} + b)$：

$$L_{CE}\left(\hat{y}, y\right) = -\left[y \log \sigma(\mathsf{w} \cdot \mathsf{x} + b) + (1-y) \log\left(1-\sigma(\mathsf{w} \cdot \mathsf{x} + b)\right)\right] \tag{23}$$

让我们看看这个损失函数是否对我们在图2中的实例起到了正确的作用。我们希望当模型的估计接近正确时损失较小，当模型感到困惑[4]时损失较大。因此，首先让我们假设图2中情感实例的正确黄金标签[5]是积极的，即$y = 1$。在这种情况下，我们的模型运行良好，因为从式(7)可以看出，它确实对这个实例给出了比消极概率(0.30)更高的积极概率(0.70)。如果我们将$\sigma(\mathsf{w} \cdot \mathsf{x} + b)$=0.70和$y = 1$代入式(23)，二项式的右半边就会消掉，计算出如下损失(当底数未指定时，我们用log表示自然对数)：

$$\begin{aligned} L_{CE}\left(\hat{y}, y\right) &= -\left[y \log \sigma(\mathsf{w} \cdot \mathsf{x} + b) + (1-y) \log\left(1-\sigma(\mathsf{w} \cdot \mathsf{x} + b)\right)\right] \\ &= -[\log \sigma(\mathsf{w} \cdot \mathsf{x} + b)] \\ &= -\log(0.70) \\ &= 0.36 \end{aligned}$$

相反，让我们假装图2中的实例实际上是消极的，即$y = 0$(或许评论者接着说："但最后我想说的是，这部电影太糟糕了！我求你不要去看它！")。在这种情况下，我们的模型会感到困惑，我们希望损失更大。现在，如果我们将式(7)中的$y = 0$和$1 - \sigma(\mathsf{w} \cdot \mathsf{x} + b)$=0.30代入式(23)，二项式的左半边就会消掉：

$$\begin{aligned} L_{CE}\left(\hat{y}, y\right) &= -\left[y \log \sigma(\mathsf{w} \cdot \mathsf{x} + b) + (1-y) \log\left(1-\sigma(\mathsf{w} \cdot \mathsf{x} + b)\right)\right] \\ &= -[\log(1-\sigma(\mathsf{w} \cdot \mathsf{x} + b))] \\ &= -\log(0.30) \\ &= 1.20 \end{aligned}$$

果然，第一个分类器的损失(0.36)小于第二个分类器的损失(1.20)。

最小化这个负对数概率为什么能够达到我们想要的效果呢？一个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分配给错误的结果。这意味着如果y等于1，那么$\hat{y}$越高(越接近1)，分类器就越好；$\hat{y}$越低(越接近0)，分类器就越差。相反地，如果y等于0，那么$1 - \hat{y}$越高(越接近1)，分类器就越好。$\hat{y}$的负对数(如果真实的y等于1)或$1 - \hat{y}$的负对数(如果真实的y等于0)是一种方便的损失度量，因为它的值可从0(1的负对数，无损失)取到无穷大(0的负对数，无穷损失)。这个损失函数还确保了当正确答案的概率最大化时，错误答案的概率最小化；由于二者之和为1，所以正确答案概率的任何增加都是以错误答案概率的减小为代价的。之所以称其为交叉熵损失，是因为式(21)也是真实概率分布y和我们估计的分布$\hat{y}$之间的 **交叉熵** 公式。

现在我们知道了我们想要最小化的对象；在下一节中，我们将介绍如何找到其最小值。

6. 梯度下降

我们使用梯度下降的目标是找到最优权重——最小化我们为模型定义的损失函数。在下面的式(24)中，我们将明确表示这样一个事实，即损失函数L是由权重参数化的，在机器学习中我们一般将其称为θ(在Logistic回归中$\theta = w, b$)。因此，目标是找到一组权重，使得损失函数最小化，并在所有样本上进行平均：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(f(x^{(i)}; \theta), y^{(i)}) \tag{24}$$

我们如何找到这个(或任何)损失函数的最小值呢？梯度下降法是一种通过确定函数的斜率在[参数(θ)空间内的]哪个方向上最陡峭地上升，并朝相反方向移动来找到函数最小值的方法。直观地说，假如你正

在峡谷中徒步旅行，试图以最快的速度下降到底部的河流，你可能会360度环顾四周，找到地面倾斜最陡峭的方向，然后沿着那个方向下山。

对于Logistic回归，很方便的一点[6]是这个损失函数是 **凸的(convex)** 。一个凸函数最多只有一个最小值；由于不存在局部最小值，(在使用梯度下降法寻找最小值的过程中)不会被它们卡住，因此可以保证从任意点开始的梯度下降都能找到最小值。(相比之下，多层神经网络的损失是非凸的，在神经网络的训练过程中梯度下降可能会在局部最小值处卡住，导致永远找不到全局最优解。)

尽管该算法(以及梯度的概念)是为方向向量设计的，但让我们首先考虑一下在我们系统的参数仅仅是一个标量w的情况下的可视化，如图4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的图象恰好具有图4中的形状，我们需要一个算法来告诉我们在下一次迭代中应该向左移动(使w^2小于w^1)还是向右移动(使w^2大于w^1)以达到最小值。

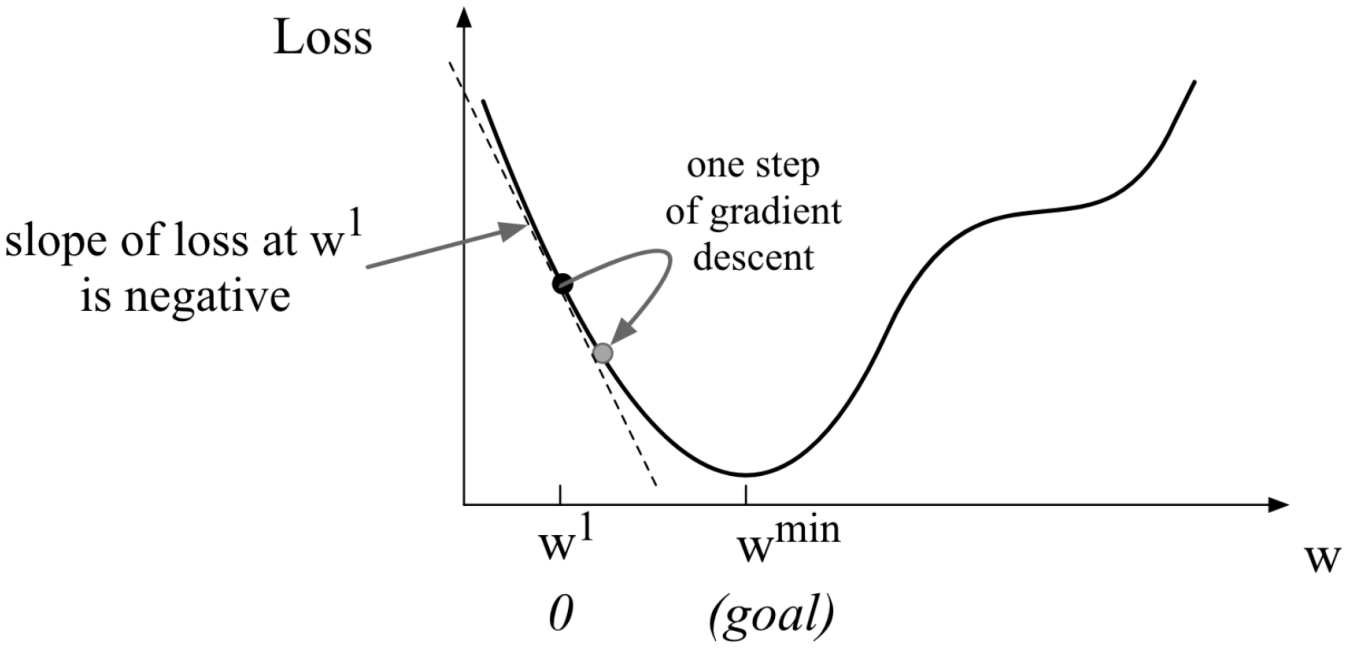


图4　通过将w沿着函数斜率的反方向移动，迭代地找到这个损失函数的最小值的第一步。由于斜率是负的，我们需要将w沿正方向向右移动。这里用上标表示学习步骤，所以w^1表示w的初始值(为0)，w^2表示第二步的值，依此类推。

梯度下降算法通过找到损失函数在当前点的 **梯度(gradient)** 并朝相反方向移动来回答这个问题。一个多变量函数的梯度是一个指向函数值增长最快方向的向量。梯度是斜率的多变量推广，因此对于一个像图4中这样的单变量函数，我们可以非正式地认为梯度就是斜率。图4中的虚线显示了这个假设的损失函数在点$w = w^1$处的斜率。你可以看到这条虚线的斜率是负的。因此，为了找到最小值，梯度下降告诉我们要朝相反方向走——将w向正方向移动。

在梯度下降中移动量的大小是斜率$\frac{\mathrm{d}}{\mathrm{d}w}L(f(x;w),y)$按 **学习率(learning rate)** η加权后的值。更高(更快)的学习率(学习速率)意味着我们应该在每一步上将w移动更多。我们对参数所做的改变是学习率乘以梯度(或斜率，在我们的单变量示例中)：

$$w^{t+1} = w^t - \eta\frac{\mathrm{d}}{\mathrm{d}w}L(f(x;w),y) \tag{25}$$

现在让我们将直觉从一个标量变量w的函数扩展到多个变量，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(对应于构成θ的N个参数)中我们应该向哪里移动。梯度就是这样一个向量，它表示了沿着(N变量函数)在这N个维度中的每个维度上函数值增长最快方向的方向分量。如果我们仅想

象两个权重维度(比方说一个权重w和一个偏置项b)，那么梯度可能是一个具有两个正交分量的向量，每个分量都告诉我们函数曲面在w维度和b维度上的坡度是多少。图5显示了在红点处取得的二维梯度向量值的可视化。

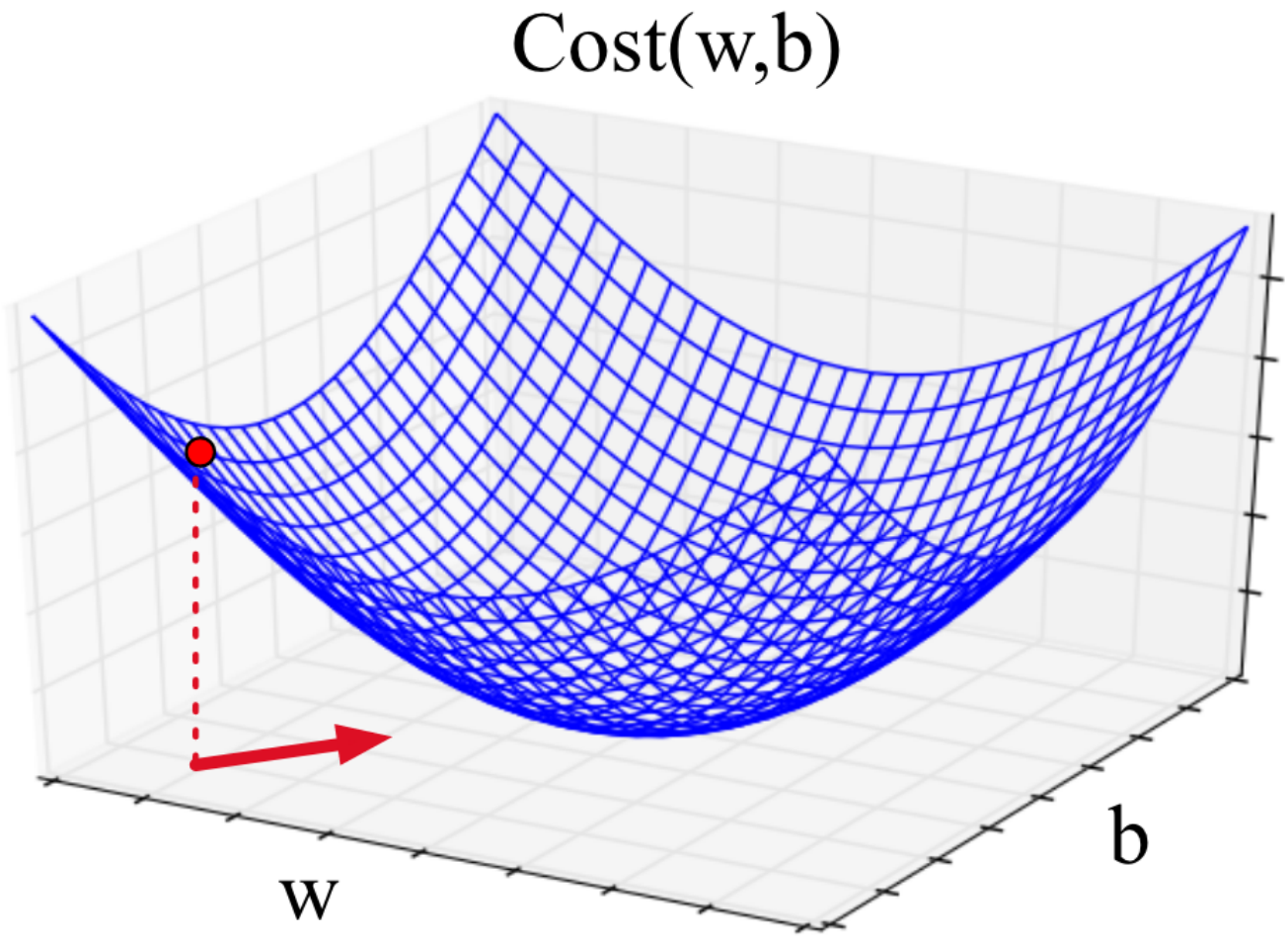


图5 在w和b两个维度上对红点处的梯度向量进行可视化，在x-y平面上标出了一个红色箭头指向我们将要去寻找最小值的方向——梯度的相反方向(请记住，梯度指向函数值增加而非减少的方向)。

在实际的Logistic回归中，参数向量w的维数要比1或2大得多，因为输入特征向量x的分量可能很多，我们需要为每个分量x_i设置一个权重w_i。对于w中的每个分量/变量w_i(加上偏置项b)，梯度都会有一个分量告诉我们关于该变量的斜率。在每一个w_i维度上，我们将斜率表示为损失函数的偏导数$\frac{\partial}{\partial w_i}L$。本质上我们是在问："变量$w_i$的一个微小变化会对总的损失函数$L$产生多大的影响？"

因此，正式地说，一个多变量函数f的梯度是一个向量，该向量的每个分量表示f关于其中一个变量的偏导数。我们将使用哈密顿算子(或称"Nabla算子")∇(希腊字母Δ的倒置)来表示梯度，并将$\hat{y}$表示为$f(x;\theta)$，以使对θ的依赖更加明显：

$$\nabla L(f(x;\theta),y) = \begin{bmatrix} \frac{\partial}{\partial w_1}L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2}L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n}L(f(x;\theta),y) \\ \frac{\partial}{\partial b}L(f(x;\theta),y) \end{bmatrix} \tag{26}$$

因此，θ是基于梯度而更新的，其最终公式为：

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta), y) \tag{27}$$

6.1. Logistic回归的梯度

为了更新θ，我们需要定义梯度$\nabla L(f(x;\theta), y)$。回想一下，对于Logistic回归，交叉熵损失函数为：

$$L_{CE}\left(\hat{y}, y\right) = -\left[y \log \sigma(\mathsf{w} \cdot \mathsf{x} + b) + (1-y) \log\left(1 - \sigma(\mathsf{w} \cdot \mathsf{x} + b)\right)\right] \tag{28}$$

可以推导出该函数对一个观测值向量x的偏导数为式(29)(对该偏导数的推导过程感兴趣的读者可以参见第10节的内容)：

$$\begin{aligned} \frac{\partial L_{CE}(\hat{y}, y)}{\partial \mathsf{w}_j} &= [\sigma(\mathsf{w} \cdot \mathsf{x} + b) - y]\mathsf{x}_j \\ &= (\hat{y} - y)\mathsf{x}_j \end{aligned} \tag{29}$$

你有时也会看到这个式子的等价形式：

$$\frac{\partial L_{CE}(\hat{y}, y)}{\partial \mathsf{w}_j} = -(y - \hat{y})\mathsf{x}_j \tag{30}$$

注意，在以上这两个式子中，关于单个权重w_j的梯度表现出一个非常直观的值：对于该观测值，真实的y与我们估计的$\hat{y} = \sigma(\mathsf{w} \cdot \mathsf{x} + b)$之间的差值，乘以相应的输入值$x_j$。

6.2. 随机梯度下降算法

随机梯度下降是一种在线算法(online algorithm)，它通过在每个训练样本之后计算损失函数的梯度并将θ推向正确的方向(梯度的相反方向)来最小化损失函数。(“在线算法”是指逐个处理其输入样本的算法，而不是等看到整个输入后再处理。)图6以伪代码的形式展示了该算法的运作过程。

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6 随机梯度下降算法。步骤1(计算损失)主要用于报告我们在当前元组上的表现；我们不需要通过计算损失来计算梯度。该算法可以在收敛时(当梯度范数$< \epsilon$时)，或当进度停止时(例如，当保留集[7]上的损失开始上升时)终止。

学习率η是一个必须调整的 **超参数(hyperparameter)** 。如果它过高，学习者迈开的步子会太大，可能越过损失函数的最小值。如果它太低，学习者迈开的步子会太小，导致花费过于长的时间才能到达最小值[8]。通常从较高的学习率开始，然后缓慢降低它，使其成为训练迭代次数k的函数；可将第k次迭代时学习率的值记为η_k。

我们会在【SLP3-I7】中更详细地讨论超参数，但简而言之，它们对于任何机器学习模型都是一种特殊的参数。与模型的常规参数(如权重w和偏置项b，它们是由算法从训练集中学习的)不同，超参数是由算法设计者选择的特殊参数，它们会影响算法的工作方式。

6.3. 举例说明

让我们通过梯度下降算法的一个步骤来演示。我们将使用图2中实例的一个简化版本，它考察了一个观测值x，其正确值为$y=1$(这是一则积极的评论)，并且具有由下面这两个特征组成的特征向量$\mathsf{x}=[\mathsf{x}_1,\mathsf{x}_2]$:

$$\begin{aligned}\mathsf{x}_1 &= 3 \quad \text{(积极词汇的计数)}\\ \mathsf{x}_2 &= 2 \quad \text{(消极词汇的计数)}\end{aligned}$$

假设θ^0中的初始权重和偏置项都设置为0，初始学习率η为0.1:

$$\begin{aligned}\mathsf{w}_1 = \mathsf{w}_2 &= b = 0\\ \eta &= 0.1\end{aligned}$$

单个更新步骤需要我们计算梯度，并乘以学习率:

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

在我们的微型样本中有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:

$$\nabla_{w,b} L = \begin{bmatrix}\frac{\partial L_{CE}(\hat{y},y)}{\partial \mathsf{w}_1}\\ \frac{\partial L_{CE}(\hat{y},y)}{\partial \mathsf{w}_2}\\ \frac{\partial L_{CE}(\hat{y},y)}{\partial b}\end{bmatrix} = \begin{bmatrix}(\sigma(\mathsf{w}\cdot\mathsf{x}+b)-y)\mathsf{x}_1\\ (\sigma(\mathsf{w}\cdot\mathsf{x}+b)-y)\mathsf{x}_2\\ \sigma(\mathsf{w}\cdot\mathsf{x}+b)-y\end{bmatrix} = \begin{bmatrix}(\sigma(0)-1)\mathsf{x}_1\\ (\sigma(0)-1)\mathsf{x}_2\\ \sigma(0)-1\end{bmatrix} = \begin{bmatrix}-0.5\mathsf{x}_1\\ -0.5\mathsf{x}_2\\ -0.5\end{bmatrix} = \begin{bmatrix}-1.5\\ -1.0\\ -0.5\end{bmatrix}$$

现在我们有了一个梯度，我们通过朝梯度的相反方向移动θ^0来计算出新的参数向量θ^1:

$$\theta^1 = \begin{bmatrix}\mathsf{w}_1\\ \mathsf{w}_2\\ b\end{bmatrix} - \eta\begin{bmatrix}-1.5\\ -1.0\\ -0.5\end{bmatrix} = \begin{bmatrix}0.15\\ 0.10\\ 0.05\end{bmatrix}$$

因此，在经过一步梯度下降后，权值变为(被“移动”到)：$\mathsf{w}_1=0.15$，$\mathsf{w}_2=0.10$，$b=0.05$。

请注意，这个观测值x恰好是一个积极的样本。我们期望在看到更多具有高计数消极词汇的消极样本之后，权重w_2会变为(“移动”至)负值。

6.4. 小批量训练

随机梯度下降之所以被称为随机，是因为它每次选择一个随机的样本，移动权重以提高对该单个样本的性能。这可能会导致非常不稳定的移动，因此通常计算批量训练实例而不是单个实例的梯度。

例如，在 **批量训练(batch training)** 中，我们计算整个数据集的梯度。通过考察如此多的样本，批量训练可以很好地估计出要朝哪个方向移动权重，代价是要花费大量时间处理训练集中的每一个样本以计算出这个完美的方向。

一种折中的方法是 **小批量(mini-batch)** 训练：我们在一组小于整个数据集的m个(可能是512或1024个)样本上进行训练。(如果m是数据集的大小，那么我们进行的是 **批量** 梯度下降；如果$m = 1$，我们就回到了随机梯度下降。)小批量训练还具有计算效率上的优势。小批量样本容易被向量化，根据计算资源选择小批的大小。这使得我们能够并行处理一个小批中的所有样本，然后累积损失，而这在单样本或批量训练中是不可能实现的。

我们只需要定义第5节中定义的交叉熵损失函数以及6.1小节中定义的梯度的小批量版本。让我们将式(22)中单样本的交叉熵损失扩展到大小为m的小批。我们将继续使用符号$x^{(i)}$和$y^{(i)}$来分别表示第i个训练特征和训练标签。我们假设训练样本是独立的：

$$\begin{aligned}\log p(\text{training labels}) &= \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) \\ &= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}) \\ &= -\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)}, y^{(i)})\end{aligned} \tag{31}$$

现在，m个样本的小批量成本函数是每个样本的平均损失：

$$\begin{aligned}Cost(\hat{y}, y) &= \frac{1}{m}\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)}, y^{(i)}) \\ &= -\frac{1}{m}\sum_{i=1}^{m}\Big(y^{(i)} \log \sigma(\mathsf{w} \cdot \mathsf{x}^{(i)} + b) + (1 - y^{(i)}) \log(1 - \sigma(\mathsf{w} \cdot \mathsf{x}^{(i)} + b))\Big)\end{aligned} \tag{32}$$

小批量梯度是式(29)中各个梯度的平均值：

$$\frac{\partial Cost(\hat{y}, y)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\Big(\sigma(\mathsf{w} \cdot \mathsf{x}^{(i)} + b) - y^{(i)}\Big)\mathsf{x}_j^{(i)} \tag{33}$$

我们可以更有效地以矩阵形式计算梯度，而不是使用求和符号，遵循我们在2.3小节中提到的向量化，其中我们有一个大小为$[m \times f]$的矩阵X，表示批中的m个输入，以及一个大小为$[m \times 1]$的向量y，表示正确的输出：

$$\begin{aligned}\frac{\partial Cost(\hat{y}, y)}{\partial \mathsf{w}} &= \frac{1}{m}(\hat{\mathsf{y}} - \mathsf{y})^{\mathsf{T}}\mathsf{X} \\ &= \frac{1}{m}(\sigma(\mathsf{X}\mathsf{w} + \mathsf{b}) - \mathsf{y})^{\mathsf{T}}\mathsf{X}\end{aligned} \tag{34}$$

7. 正则化

Numquam ponenda est pluralitas sine necessitate
'Plurality should never be proposed unless needed'
“如无必要，避重趋轻”

William of Occam
——奥卡姆的威廉

学习使模型与训练数据完美匹配的权重存在一个问题。如果一个特征因为恰好只出现在一个类别中而完美预测结果，那么它将被分配一个非常高的权重。特征的权重将试图完美地(实际上是过于完美了)拟合训练集的细节，对偶然与类别相关的噪声因素进行建模。这个问题被称为 **过拟合(overfitting)** 。一个好的模型应该能够很好地从训练数据 **泛化(generalize)** 到未见过的测试集，但是过拟合的模型将具有较差的泛化能力。

为了避免过拟合，在式(24)的目标函数中添加了一个新的 **正则化(regularization)** 项$R(\theta)$，得到了一批m个样本的目标(稍微改写了式(24)，使其最大化对数概率而不是最小化损失，并去掉了不影响argmax的$\frac{1}{m}$项)：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta) \tag{35}$$

新的正则化项$R(\theta)$用于惩罚较大的权重。因此，与一个跟数据匹配得稍差一些但使用较小权重的设置相比，一个跟训练数据完美匹配的权重设置——但使用了许多高权值来实现——将受到更多惩罚。计算该正则化项$R(\theta)$的常用方法有两种。 **L2正则化** 是权值的二次函数，因其使用了权值的L2范数(的平方)而得名。L2范数，即$||\theta||_2$，就是从原点到向量θ的 **欧几里得距离(Euclidean distance)** 。如果θ由n个权重组成，则：

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n} \theta_j^2 \tag{36}$$

L2正则化的目标函数变为：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} \theta_j^2 \tag{37}$$

L1正则化 是权值的线性函数，得名于L1范数$||W||_1$，即权重的绝对值之和，或 **曼哈顿距离(Manhattan distance)** (曼哈顿距离是指你在像纽约这样有街道网格的城市中的两点之间必须走过的距离)：

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i| \tag{38}$$

L1正则化的目标函数变为：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j| \tag{39}$$

这些类型的正则化来自统计学，其中L1正则化被称为 **套索回归(lasso regression)** [9]，L2正则化被称为 **岭回归(ridge regression)** ，两者都常用于语言处理。L2正则化因其导数简单(θ^2的导数就是2θ)而更容易优化，而L1正则化更复杂($|\theta|$的导数在0处不连续)。但是，L2偏爱具有许多小权重的权重向量，L1则偏爱具有一些较大权重但更多权重设置为零的稀疏解。因此，L1正则化导致更为稀疏的权重向量，即特征要少得多。

L1和L2正则化都具有贝叶斯解释，作为对权重分布形态的先验约束。L1正则化可以看作是权重的拉普拉斯先验。L2正则化对应于假设权重服从均值$\mu = 0$的高斯分布。在高斯(正态)分布中，一个值偏离均值越远，其出现的概率就越低(以方差σ为标度)。通过对权重使用高斯先验，我们说权重更倾向于取值0。权重θ_j的高斯函数为：

$$\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2} \right) \tag{40}$$

如果我们将每个权重乘以其高斯先验，我们就最大化了下面这个约束：

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} P(y^{(i)}|x^{(i)}) \times \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\theta_j - \mu_j)^2}{2\sigma_j^2} \right) \tag{41}$$

在对数空间中，当$\mu=0$，且假设$2\sigma^2=1$时，它对应于：

$$\hat{\theta}=\underset{\theta}{\operatorname{argmax}}\sum_{i=1}^{m}\log P(y^{(i)}|x^{(i)})-\alpha\sum_{j=1}^{n}\theta_j^2 \tag{42}$$

其形式与式(37)相同。

8. 多项Logistic回归中的学习

多项Logistic回归的损失函数将二元Logistic回归的损失函数从2类推广到K类。回想一下，二元Logistic回归的交叉熵损失(重复式(22))为：

$$L_{CE}\left(\hat{y},y\right)=-\log p(y|x)=-\left[y\log\hat{y}+(1-y)\log\left(1-\hat{y}\right)\right] \tag{43}$$

多项Logistic回归的损失函数将式(43)中的两个项(当$y=1$时非零的一个项和当$y=0$时非零的一个项)推广到K个项。如前所述，对于多项回归，我们将y和$\hat{\mathsf{y}}$都表示为向量。真实标签y是一个具有K个分量的向量，每个分量对应一个类别，如果正确类别为c，则$y_c=1$，而y的所有其他分量都为0。我们的分类器将产生一个具有K个分量的估计向量$\hat{\mathsf{y}}$，其中分量$\hat{\mathsf{y}}_k$表示估计概率$p(\mathsf{y}_k=1|\mathsf{x})$。

由二元Logistic回归推广而来的单个样本x的损失函数，是K个输出类别的对数之和，每个类别都由它们的概率y_k加权(式(44))。这恰好是正确类别c的负对数概率(式(45))：

$$\begin{aligned}L_{CE}(\hat{\mathsf{y}},\mathsf{y})&=-\sum_{k=1}^{K}\mathsf{y}_k\log\hat{\mathsf{y}}_k &(44)\\&=-\log\hat{\mathsf{y}}_c &(45)\\&=-\log\hat{p}(\mathsf{y}_c=1|\mathsf{x})\\&=-\log\frac{\exp(\mathsf{w}_c\cdot\mathsf{x}+b_c)}{\sum_{j=1}^{K}\exp\left(\mathsf{w}_j\cdot\mathsf{x}+b_j\right)} &(46)\end{aligned}$$

我们如何由式(44)推导出式(45)？因为只有一个类别(我们称之为c)是正确的，所以向量y仅对k的这个值取1，即$y_c=1$且$y_j=0\ \forall j\neq c$。这意味着式(44)的求和项中除与真实类别c对应的项外其他都为0。因此，交叉熵损失仅仅是正确类别对应的输出概率的对数，因此我们也称式(45)为 **负对数似然损失(negative log likelihood loss)** 。

当然，对于梯度下降，我们不需要损失，我们需要它的梯度。单样本的梯度与我们在式(29)中看到的二元Logistic回归的梯度非常相似，即$(\hat{y}-y)x$。让我们考虑梯度的一部分，关于单个权重的偏导数。对于类别k，输入x的第i个分量的权重为$\mathsf{w}_{k,i}$。损失关于$\mathsf{w}_{k,i}$的偏导数是什么？这个偏导数恰好是类别k的真实值(要么是1，要么是0)与分类器输出类别k的概率之间的差值，由类别k权重向量的第i个分量对应的输入值x_i来加权：

$$\begin{aligned}\frac{\partial L_{CE}}{\partial\mathsf{w}_{k,i}}&=-(\mathsf{y}_k-\hat{\mathsf{y}}_k)\mathsf{x}_i\\&=-(\mathsf{y}_k-p(\mathsf{y}_k=1|\mathsf{x}))\mathsf{x}_i\\&=-\left(\mathsf{y}_k-\frac{\exp\left(\mathsf{w}_k\cdot\mathsf{x}+b_k\right)}{\sum_{j=1}^{K}\exp\left(\mathsf{w}_j\cdot\mathsf{x}+b_j\right)}\right)\mathsf{x}_i\end{aligned} \tag{47}$$

我们将在【SLP3-I7】中介绍神经网络时回顾softmax回归中梯度的这种情况，届时我们还将在彼章的式(35)—式(43)中讨论这个梯度的推导。

9. 解释模型

通常，我们想知道的不仅仅是对观测值的正确分类。我们还想知道分类器为什么做出了它所做的决策。也就是说，我们希望我们的决策是可解释的。可解释性可能很难严格定义，但其核心思想是作

为人类，我们应该知道我们的算法为什么会得出它们所得出的结论。由于Logistic回归的特征通常是人为设计的，因此理解分类器决策的一种方法是理解每个特征在决策中所起的作用。Logistic回归可以与统计检验(似然比检验或Wald检验)结合使用；通过其中一种检验调查特定特征是否显著，或检查其大小(与该特征相关联的权重w有多大？)可以帮助我们解释分类器为什么做出它所做的决策。这对于构建透明模型非常重要。

此外，除了作为分类器使用外，在NLP以及其他诸多领域，Logistic回归还广泛用作分析工具，用于检验对于各种解释变量(特征)效果[10]的假设。在文本分类中，也许我们想知道逻辑否定词(no、not、never)是否更可能与消极情感相关联，或者电影的消极(负面)评论是否更可能讨论电影摄影。然而，在这样做时有必要控制潜在混杂因素——其他可能影响情感的因素[如电影类型、制作年份，还有可能是评论的长度(词数)]。或者我们可能正在研究NLP提取的语言特征与非语言结果(医院再入院、政治结果或产品销售)之间的关系，但需要控制混杂因素(患者年龄、投票县、产品品牌)。在这种情况下，Logistic回归允许我们检验某个特征是否在其他特征的影响之外与某个结果相关联。

10. 拓展阅读：推导梯度公式

在本节中，我们给出Logistic回归的交叉熵损失函数L_{CE}的梯度推导。让我们从一些快速的微积分复习开始。首先是$\ln(x)$的导数：

$$\frac{\mathrm{d}}{\mathrm{d}x}\ln(x)=\frac{1}{x} \tag{48}$$

其次是sigmoid函数的[非常优雅[11]的(very elegant)]导数：

$$\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z}=\sigma(z)(1-\sigma(z)) \tag{49}$$

最后是导数的 **链式法则(chain rule)** 。假设我们正在使用链式法则计算复合函数$f(x)=u(v(x))$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：

$$\frac{\mathrm{d}f}{\mathrm{d}x}=\frac{\mathrm{d}u}{\mathrm{d}v}\cdot\frac{\mathrm{d}v}{\mathrm{d}x} \tag{50}$$

第一步，我们想知道损失函数关于单个权重w_j的导数(我们需要对每个权重以及偏置项进行计算)：

$$\begin{aligned}\frac{\partial L_{CE}}{\partial \mathsf{w}_j}&=\frac{\partial}{\partial \mathsf{w}_j}-[y\log\sigma(\mathsf{w}\cdot\mathsf{x}+b)+(1-y)\log\left(1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)\right)]\\&=-\left[\frac{\partial}{\partial \mathsf{w}_j}y\log\sigma(\mathsf{w}\cdot\mathsf{x}+b)+\frac{\partial}{\partial \mathsf{w}_j}(1-y)\log[1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)]\right]\end{aligned} \tag{51}$$

第二步，使用链式法则，并借助自然对数的导数：

$$\frac{\partial L_{CE}}{\partial \mathsf{w}_j}=-\frac{y}{\sigma(\mathsf{w}\cdot\mathsf{x}+b)}\frac{\partial}{\partial \mathsf{w}_j}\sigma(\mathsf{w}\cdot\mathsf{x}+b)-\frac{1-y}{1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)}\frac{\partial}{\partial \mathsf{w}_j}(1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)) \tag{52}$$

上式可整理(化简、合并、重排)为：

$$\frac{\partial L_{CE}}{\partial \mathsf{w}_j}=-\left[\frac{y}{\sigma(\mathsf{w}\cdot\mathsf{x}+b)}-\frac{1-y}{1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)}\right]\frac{\partial}{\partial \mathsf{w}_j}\sigma(\mathsf{w}\cdot\mathsf{x}+b) \tag{53}$$

第三步，代入sigmoid函数的导数，并再次使用链式法则，我们最终会得到式(54)：

$$\begin{aligned}\frac{\partial L_{CE}}{\partial \mathsf{w}_j} &= -\left[\frac{y-\sigma(\mathsf{w}\cdot\mathsf{x}+b)}{\sigma(\mathsf{w}\cdot\mathsf{x}+b)\left[1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)\right]}\right]\sigma(\mathsf{w}\cdot\mathsf{x}+b)\left[1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)\right]\frac{\partial\left(\mathsf{w}\cdot\mathsf{x}+b\right)}{\partial \mathsf{w}_j} \\ &= -\left[\frac{y-\sigma(\mathsf{w}\cdot\mathsf{x}+b)}{\sigma(\mathsf{w}\cdot\mathsf{x}+b)\left[1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)\right]}\right]\sigma(\mathsf{w}\cdot\mathsf{x}+b)\left[1-\sigma(\mathsf{w}\cdot\mathsf{x}+b)\right]\mathsf{x}_j \\ &= -\left[y-\sigma(\mathsf{w}\cdot\mathsf{x}+b)\right]\mathsf{x}_j \\ &= \left[\sigma(\mathsf{w}\cdot\mathsf{x}+b)-y\right]\mathsf{x}_j\end{aligned} \tag{54}$$

11. 本章小结

本章介绍了用于 **分类(classification)** 的 **Logistic回归(logistic regression)** 模型。

- Logistic回归是一种有监督的机器学习分类器，它从输入中提取实值特征，将每个特征乘以一个权重，对它们求和，然后将总和传入 **sigmoid(S型)函数** 来生成概率。使用一个阈值来进行决策。
- Logistic回归可用于二分类(例如积极和消极情感)或多分类[即 **多项Logistic回归(multinomial logistic regression)** ，例如n元文本分类、词性标注等]。
- 多项Logistic回归使用 **softmax函数** 来计算概率。
- 权重(向量w和偏置项b)是通过一个必须最小化的损失函数[如 **交叉熵损失(cross-entropy loss)**]从标记的训练集中学习的。
- 最小化这个损失函数是一个 **凸优化(convex optimization)** 问题，使用迭代算法如 **梯度下降(gradient descent)** 来寻找最优权重。
- **正则化(regularization)** 用于避免 **过拟合(overfitting)** 。
- Logistic回归也是最有用的分析工具之一，因为它能够透明地研究单个特征的重要性。

12. 参考文献和历史说明

Logistic回归是在统计学领域发展起来的，在20世纪60年代曾被用于二进制数据的分析，在医学中尤为常见(Cox, 1969[12])。从20世纪70年代末开始，它在语言学中得到了广泛的应用，成为语言变异研究的正式基础之一(Sankoff and Labov, 1979[13])。

然而，直到20世纪90年代，Logistic回归才在自然语言处理中变得普遍起来，当时似乎是从两个方向同时出现的。第一个来源是信息检索和语音处理这两个邻近领域，它们都使用了回归，并且都为NLP提供了许多其他的统计技术。事实上，很早就将Logistic回归用于文档路由(传送)是最早使用(LSI)嵌入作为单词表示的NLP应用之一(Schütze et al., 1995[14])。

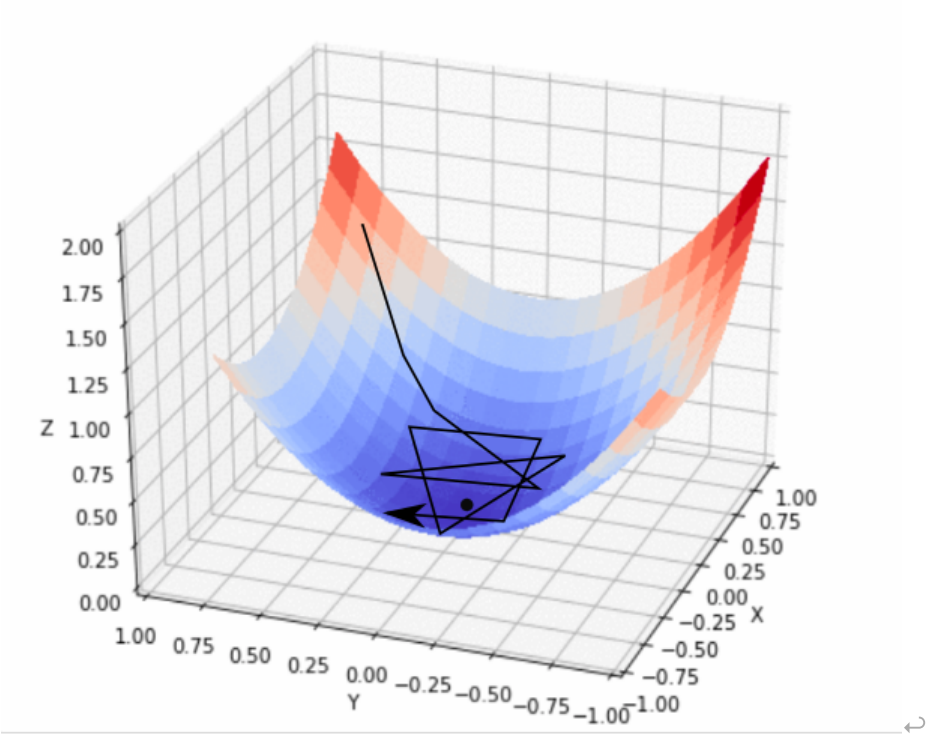
与此同时，在20世纪90年代初，IBM研究院以 **最大熵(maximum entropy)** 模型或 **maxent** (Berger et al., 1996[15])的名义将Logistic回归开发出来并应用于NLP，似乎与统计学文献无关。在这个名称下，它被应用于语言建模(Rosenfeld, 1996[16])、词性标注(Ratnaparkhi, 1996[17])、句法分析(Ratnaparkhi, 1997[18])、共指消解(Kehler, 1997b[19])和文本分类(Nigam et al., 1999[20])。

更多关于分类的内容可以在机器学习教材中找到(Hastie et al. 2001[21], Witten and Frank 2005[22], Bishop 2006[23], Murphy 2012[24])。

---

1. “被哈希为唯一的整数”(be hashed into a unique integer)意为“通过哈希函数(将用户对特征的字符串描述)转换为唯一的整数“。——译者注↩
2. A. Y. Ng and M. I. Jordan. 2002. **On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes**. *NeurIPS*.↩
3. S. Wang and C. D. Manning. 2012. **Baselines and bigrams: Simple, good sentiment and topic classification**. *ACL*.↩
4. “模型感到困惑”(the model is confused)意为“模型做出的估计离正确结果相去甚远”。——译者注↩

5. “黄金标签”(gold label)一般指在监督或半监督学习中，由专家手工标注的训练数据。这些标注被认为是可靠的基准真值(ground truth)，相当于“可信的正确标签”。在NLP中，由于很多时候不同的人对同样的语言会有不同的理解，人们往往会对数据的标注有不同的意见。因此，NLP中“可信的正确标签”被称为gold label或gold standard(黄金标准)，而不是ground truth。——译者注↩
6. “很方便”(conveniently)的意思是：由于Logistic回归的损失函数是凸函数，这使得我们在使用梯度下降法寻找其最小值时更加方便。后文会对“方便”的原因做出解释。——译者注↩
7. 保留集(held-out set)是指在机器学习中，从原始数据集中分离出来的一部分数据，用于评估模型的性能。——译者注↩
8. 在这里，学习率相当于梯度下降的移动步幅。如果学习率过高，移动得过快，我们就可能直接越过最小值，在“谷底”附近“来回蹦跳”(如下图所示)，永远到达不了最小值；如果学习率过低，移动得过慢，就会导致训练过程过于漫长而不可行，还有可能使算法卡在(“陷入”)局部最小值处。因此，我们必须谨慎地选择学习率。——译者注



9. R. J. Tibshirani. 1996. **Regression shrinkage and selection via the lasso**. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.↩
10. 此处的“效果”(effect)一词指的是解释变量对因变量(响应变量)的影响程度。解释变量是指在统计模型中用来解释因变量变化的变量。——译者注↩
11. 在数学中，“elegant”(优雅)是一个审美术语，指一种能够提供对数学的洞察力的想法，无论是通过统一不同的领域，引入单一领域的新视角，还是提供一种证明技巧，该技巧要么特别简单，要么捕捉到为什么它证明的结果是正确的直觉或想象力。可参考维基百科：https://en.wikipedia.org/wiki/List_of_mathematical_jargon 。——译者注↩
12. D. Cox. 1969. *Analysis of Binary Data*. Chapman and Hall, London.↩
13. D. Sankoff and W. Labov. 1979. **On the uses of variable rules**. *Language in society*, 8(2-3):189–222.↩
14. H. Schütze, D. A. Hull, and J. Pedersen. 1995. **A comparison of classifiers and document representations for the routing problem**. *SIGIR-95*.↩
15. A. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. **A maximum entropy approach to natural language processing**. *Computational Linguistics*, 22(1):39–71.↩
16. R. Rosenfeld. 1996. **A maximum entropy approach to adaptive statistical language modeling**. *Computer Speech and Language*, 10:187–228.↩
17. A. Ratnaparkhi. 1996. **A maximum entropy part-of-speech tagger**. *EMNLP*.↩
18. A. Ratnaparkhi. 1997. **A linear observed time statistical parser based on maximum entropy models**. *EMNLP*.↩
19. A. Kehler. 1997b. **Probabilistic coreference in information extraction**. *EMNLP*.↩
20. K. Nigam, J. D. Lafferty, and A. McCallum. 1999. **Using maximum entropy for text classification**. *IJCAI99 workshop on machine learning for information filtering*.↩

21. T. Hastie, R. J. Tibshirani, and J. H. Friedman. 2001. *The Elements of Statistical Learning*. Springer.↩
22. I. H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition*. Morgan Kaufmann.↩
23. C. M. Bishop. 2006. *Pattern recognition and machine learning*. Springer.↩
24. K. P. Murphy. 2012. *Machine learning: A probabilistic perspective*. MIT Press.↩