第 3 单元 线性模型

支撑的课程目标

1. 能够基于智能信息处理的基本理论和技术，识别和理解数据处理与分析等问题的相关特性.
2. 能够运用智能信息处理的相关原理和专业知识，设计实验方案，为解决数据处理与分析等问题提供支持.

基本要求

1. 应用线性回归模型和逻辑回归模型，解决数据分析领域的回归问题.
2. 应用损失函数、优化器和模型可视化，分析回归模型的问题.

教学重点与难点

重点： 多元线性回归；逻辑回归.

难点： 归回模型的学习原理.

教学过程设计

新课导入、知识讲授、教学目标达成考核、总结.

教学过程设计

本单元教学通过“互动、开放”的课堂形式，采用探究式学习、问题导入的教学方法，激发学生的学习兴趣，促成课程目标的达成.

教学学时

6 学时.

一、导入新课（15 分钟）

回归（regression）是能为一个或多个自变量与因变量之间关系建模的一类方法。在自然科学和社会科学领域，回归经常用来表示输入和输出之间的关系。

在机器学习领域中的大多数任务通常都与预测（prediction）有关。当我们想预测一个数值时，就会涉及到回归问题。常见的例子包括：预测价格（房屋、股票等）、预测住院时间（针对住院病人等）、预测需求（零售销量等）。但不是所有的预测都是回归问题。在本节后面，将介绍分类问题。分类问题的目标是预测数据属于一组类别中的哪一个。

二、新课讲授 (210 分钟)

本单元要点:

* 回归的线性模型（一元线性回归和多元线性回归）

* 分类的线性模型（logistic 回归和 softmax 回归）

1. 线性回归

1.1 线性回归的基本元素

线性回归（linear regression）可以追溯到 19 世纪初，它在回归的各种标准工具中最简单而且最流行。线性回归基于几个简单的假设：首先，假设自变量 $\mathbf{x}$ 和因变量 y 之间的关系是线性的，即 y 可以表示为 $\mathbf{x}$ 中元素的加权和，这里通常允许包含观测值的一些噪声；其次，我们假设任何噪声都比较正常，如噪声遵循正态分布。

为了解释线性回归，我们举一个实际的例子：我们希望根据房屋的面积（平方英尺）和房龄（年）来估算房屋价格（人民币）。为了开发一个能预测房价的模型，我们需要收集一个真实的数据集。这个数据集包括了房屋的销售价格、面积和房龄。在机器学习的术语中，该数据集称为训练数据集（training data set）或训练集（training set）。每行数据（比如一次房屋交易相对应的数据）称为样本（sample），也可以称为数据点（data point）或数据样本（data instance）。我们把试图预测的目标（比如预测房屋价格）称为标签（label）或目标（target）。预测所依据的自变量（面积和房龄）称为特征（feature）或协变量（covariate）。

通常，我们使用 n 来表示数据集中的样本数。对索引为 i 的样本，其输入表示为 $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}]^\top$，其对应的标签是 $y^{(i)}$。

1.1.1 线性模型

线性假设是指目标（房屋价格）可以表示为特征（面积和房龄）的加权和，

如下面的式子：

$$\mathrm{price} = w_{\mathrm{area}} \cdot \mathrm{area} + w_{\mathrm{age}} \cdot \mathrm{age} + b. \tag{1}$$

(1) 中的 w_{area} 和 w_{age} 称为权重（weight），权重决定了每个特征对我们预测值的影响。b 称为偏置（bias）、偏移量（offset）或截距（intercept）。偏置是指当所有特征都取值为 0 时，预测值应该为多少。即使现实中不会有任何房子的面积是 0 或房龄正好是 0 年，我们仍然需要偏置项。如果没有偏置项，我们模型的表达能力将受到限制。严格来说，是输入特征的一个仿射变换（affine transformation）。仿射变换的特点是通过加权和对特征进行线性变换（linear transformation），并通过偏置项来进行平移（translation）。

给定一个数据集，我们的目标是寻找模型的权重 $\mathbf{w}$ 和偏置 b，使得根据模型做出的预测大体符合数据里的真实价格。输出的预测值由输入特征通过线性模型的仿射变换决定，仿射变换由所选权重和偏置确定。

而在机器学习领域，我们通常使用的是高维数据集，建模时采用线性代数表示法会比较方便。当我们的输入包含 d 个特征时，我们将预测结果 $\hat{y}$（通常使用“尖角”符号表示 y 的估计值）表示为：

$$\hat{y} = w_1 x_1 + ... + w_d x_d + b. \tag{2}$$

将所有特征放到向量 $\mathbf{x} \in \mathbb{R}^d$ 中，并将所有权重放到向量 $\mathbf{w} \in \mathbb{R}^d$ 中，我们可以用点积形式来简洁地表达模型：

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b. \tag{3}$$

在 (3) 中，向量 $\mathbf{x}$ 对应于单个数据样本的特征。用符号表示的矩阵 $\mathbf{X} \in \mathbb{R}^{n\times d}$ 可以很方便地引用我们整个数据集的 n 个样本。其中，$\mathbf{X}$ 的每一行是一个样本，每一列是一种特征。

对于特征集合 $\mathbf{X}$，预测值 $\hat{\mathbf{y}} \in \mathbb{R}^n$ 可以通过矩阵-向量乘法表示为：

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b \tag{4}$$

这个过程中的求和将使用广播机制（广播机制后面会详细介绍）。给定训练数据特征 $\mathbf{X}$ 和对应的已知标签 $\mathbf{y}$，线性回归的目标是找到一组权重向量 $\mathbf{w}$ 和偏

置 b：当给定从 $\mathbf{X}$ 的同分布中取样的新样本特征时，这组权重向量和偏置能够使得新样本预测标签的误差尽可能小。

虽然我们相信给定 $\mathbf{x}$ 预测 y 的最佳模型会是线性的，但我们很难找到一个有 n 个样本的真实数据集，其中对于所有的 $1 \leq i \leq n$，$y^{(i)}$ 完全等于 $\mathbf{w}^{\top}\mathbf{x}^{(i)} + b$。无论我们使用什么手段来观察特征 $\mathbf{X}$ 和标签 $\mathbf{y}$，都可能会出现少量的观测误差。因此，即使确信特征与标签的潜在关系是线性的，我们也会加入一个噪声项来考虑观测误差带来的影响。

在开始寻找最好的模型参数（model parameters）$\mathbf{w}$ 和 b 之前，我们还需要两个东西：（1）一种模型质量的度量方式；（2）一种能够更新模型以提高模型预测质量的方法。

1.1.2 损失函数

在我们开始考虑如何用模型拟合（fit）数据之前，我们需要确定一个拟合程度的度量。损失函数（loss function）能够量化目标的实际值与预测值之间的差距。通常我们会选择非负数作为损失，且数值越小表示损失越小，完美预测时的损失为 0。回归问题中最常用的损失函数是平方误差函数。当样本 i 的预测值为 $\hat{y}^{(i)}$，其相应的真实标签为 $y^{(i)}$ 时，平方误差可以定义为以下公式：

$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2. \tag{5}$$

常数 $\frac{1}{2}$ 不会带来本质的差别，但这样在形式上稍微简单一些（因为当我们对损失函数求导后常数系数为 1）。由于训练数据集并不受我们控制，所以经验误差只是关于模型参数的函数。为了进一步说明，来看下面的例子。我们为一维情况下的回归问题绘制图像，如图 1 所示。

注意：由于平方误差函数中的二次方项，估计值 $\hat{y}^{(i)}$ 和观测值 $y^{(i)}$ 之间较大的差异将导致更大的损失值。(这可能是一把双刃剑。虽然它能使模型避免大的错误，但也可能导致对异常数据的过度敏感)。为了度量模型在整个数据集上的质量，我们需计算在训练集 n 个样本上的损失均值（也等价于求和）。

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\mathbf{w}^{\top}\mathbf{x}^{(i)} + b - y^{(i)}\right)^2. \tag{6}$$

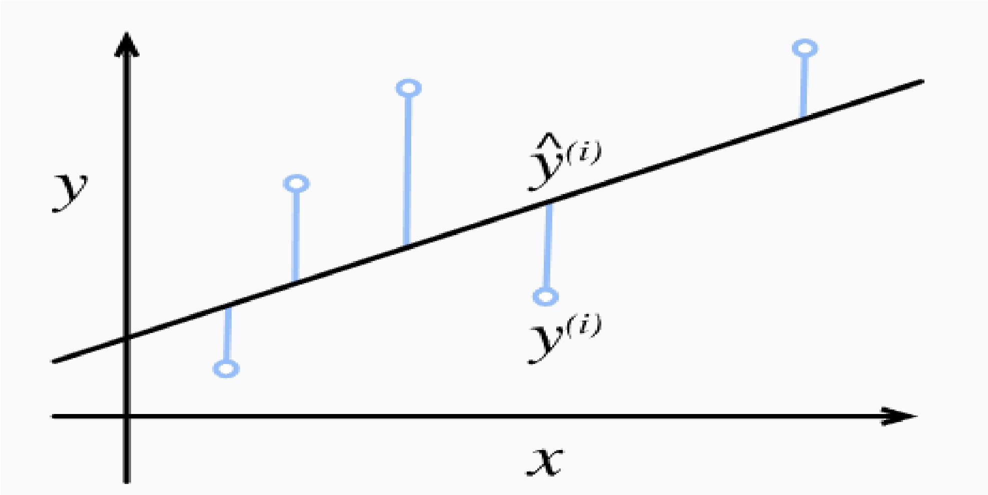

图 1: 用线性模型拟合数据

在训练模型时，我们希望寻找一组参数（$\mathbf{w}^*, b^*$），这组参数能最小化在所有训练样本上的总损失。如下式：

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\ L(\mathbf{w}, b). \tag{7}$$

1.1.3 解析解

线性回归刚好是一个很简单的优化问题。与后续所讲到的其他大部分模型不同，线性回归的解可以用一个公式简单地表达出来，这类解叫作解析解(analytical solution)。首先，我们将偏置 b 合并到参数 $\mathbf{w}$ 中，合并方法是在包含所有参数的矩阵中附加一列。即令 $\mathbf{w} = (w_0, w_1, \cdots, w_d) \in \mathbb{R}^{d+1}$，$\mathbf{x} = (1, x_1, \cdots, x_d) \in \mathbb{R}^{d+1}$，其中 $b = w_0$，则

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \cdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d)} \\ & & \cdots & & \\ 1 & x_n & x_n & \cdots & x_{nd} \end{bmatrix}$$

因此，式 (4) 可以写成向量形式：

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} \tag{8}$$

由此，求得损失函数的向量形式

$$\begin{aligned} L(\mathbf{w}) &= (\mathbf{y}-\hat{\mathbf{y}})^\top(\mathbf{y}-\hat{\mathbf{y}}). \\ &= (\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}). \\ &= \|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2 \end{aligned} \tag{9}$$

根据函数极值问题的求解方法，我们的预测问题变成最小化 $\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2$。这在损失平面上只有一个临界点，这个临界点对应于整个区域的损失极小点。将损失函数关于 $\mathbf{w}$ 求导，

$$\begin{aligned} \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}}\left[\mathbf{y}^\top\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\right] \\ &= 0 - \mathbf{X}^T\mathbf{y} - \mathbf{X}^\top\mathbf{y} + (\mathbf{X}^\top\mathbf{X} + \mathbf{X}^T\mathbf{X})\mathbf{w} \\ &= 2\mathbf{X}^\top\mathbf{X}w - 2\mathbf{X}^\top\mathbf{y} \\ &= 2\mathbf{X}^\top(\mathbf{X}\mathbf{w}-\mathbf{y}) \end{aligned} \tag{10}$$

其中使用的矩阵微分公式如下：

$$\begin{aligned} \frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} &= \frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \mathbf{a} \\ \frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} &= (\mathbf{A}+\mathbf{A}^T)\mathbf{x} \end{aligned} \tag{11}$$

当矩阵 $\mathbf{X}^\top\mathbf{X}$ 为满秩矩阵或正定矩阵时，令 $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = 0$，得到解析解 $\mathbf{w}$ 为

$$\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{12}$$

像线性回归这样的简单问题存在解析解，但并不是所有的问题都存在解析解。解析解可以进行很好的数学分析，但解析解对问题的限制很严格，导致它无法广泛应用在深度学习里。

1.1.4 随机梯度下降

即使在我们无法得到解析解的情况下，我们仍然可以有效地训练模型。在许多任务上，那些难以优化的模型效果要更好。因此，弄清楚如何训练这些难以优化的模型是非常重要的。

我们用到一种名为梯度下降（gradient descent）的方法，这种方法几乎可以优化所有深度学习模型。它通过不断地在损失函数递减的方向上更新参数来降低误差。

梯度下降最简单的用法是计算损失函数（数据集中所有样本的损失均值）关于模型参数的导数（在这里也可以称为梯度）。但实际中的执行可能会非常慢：因为在每一次更新参数之前，我们必须遍历整个数据集。因此，我们通常会在每次需要计算更新的时候随机抽取一小批样本，这种变体叫做小批量随机梯度下降（minibatch stochastic gradient descent）。

在每次迭代中，我们首先随机抽样一个小批量 $\mathcal{B}$，它是由固定数量的训练样本组成的。然后，我们计算小批量的平均损失关于模型参数的导数（也可以称为梯度）。最后，我们将梯度乘以一个预先确定的正数 η，并从当前参数的值中减掉。

我们用下面的数学公式来表示这一更新过程（∂ 表示偏导数）：

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{(\mathbf{w}, b)} l^{(i)}(\mathbf{w}, b). \tag{13}$$

总结一下，算法的步骤如下：

（1）初始化模型参数的值，如随机初始化；

（2）从数据集中随机抽取小批量样本且在负梯度的方向上更新参数，并不断迭代这一步骤。

对于平方损失和仿射变换，我们可以明确地写成如下形式：

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) = \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{x}^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right), \\ b &\leftarrow b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_b l^{(i)}(\mathbf{w}, b) = b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right). \end{aligned} \tag{14}$$

公式中的 $\mathbf{w}$ 和 $\mathbf{x}$ 都是向量。在这里，更优雅的向量表示法比系数表示法（如 $w_1, w_2, \ldots, w_d$）更具可读性。$|\mathcal{B}|$ 表示每个小批量中的样本数，这也称为批量大

小（batch size）。η 表示学习率（learning rate）。批量大小和学习率的值通常是手动预先指定，而不是通过模型训练得到的。这些可以调整但不在训练过程中更新的参数称为超参数（hyperparameter）。调参（hyperparameter tuning）是选择超参数的过程。超参数通常是我们根据训练迭代结果来调整的，而训练迭代结果是在独立的验证数据集（validation dataset）上评估得到的。

在训练了预先确定的若干迭代次数后（或者直到满足某些其他停止条件后），我们记录下模型参数的估计值，表示为 $\hat{\mathbf{w}}, \hat{b}$。但是，即使我们的函数确实是线性的且无噪声，这些估计值也不会使损失函数真正地达到最小值。因为算法会使得损失向最小值缓慢收敛，但却不能在有限的步数内非常精确地达到最小值。

线性回归恰好是一个在整个域中只有一个最小值的学习问题。但是对像深度神经网络这样复杂的模型来说，损失平面上通常包含多个最小值。深度学习实践者很少会去花费大力气寻找这样一组参数，使得在训练集上的损失达到最小。事实上，更难做到的是找到一组参数，这组参数能够在我们从未见过的数据上实现较低的损失，这一挑战被称为泛化（generalization）。

1.1.5 用模型进行预测

给定“已学习”的线性回归模型 $\hat{\mathbf{w}}^\top\mathbf{x} + \hat{b}$，现在我们可以通过房屋面积 x_1 和房龄 x_2 来估计一个（未包含在训练数据中的）新房屋价格。给定特征估计目标的过程通常称为预测（prediction）或推断（inference）。

在本课程中将尝试坚持使用预测这个词。虽然推断这个词已经成为深度学习的标准术语，但其实推断这个词有些用词不当。在统计学中，推断更多地表示基于数据集估计参数。当深度学习从业者与统计学家交谈时，术语的误用经常导致一些误解。

1.2 最小化平方误差函数的统计意义

接下来，我们通过对噪声分布的假设来解读平方损失目标函数。

正态分布和线性回归之间的关系很密切。正态分布（normal distribution），也称为高斯分布（Gaussian distribution），最早由德国数学家高斯（Gauss）应用于天文学研究。简单的说，若随机变量 x 具有均值 μ 和方差 σ^2（标准差 σ），其正

态分布概率密度函数如下：

$$p(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right). \tag{15}$$

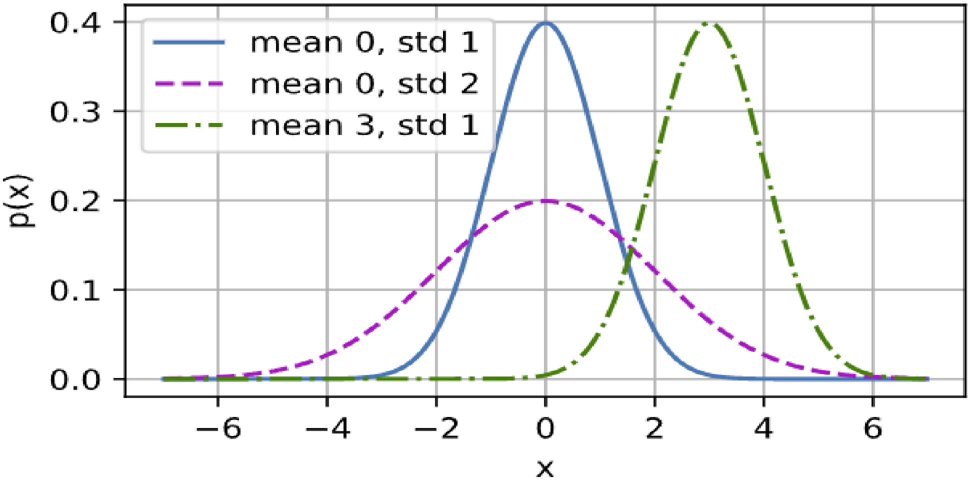

图 2: 用线性模型拟合数据

就像我们在图 2 所看到的，改变均值会产生沿 x 轴的偏移，增加方差将会分散分布、降低其峰值。

均方误差损失函数（简称均方损失）可以用于线性回归的一个原因是：我们假设了观测中包含噪声，其中噪声服从正态分布。噪声正态分布如下式:

$$y=\mathbf{w}^\top\mathbf{x}+b+\epsilon, \tag{16}$$

其中，$\epsilon\sim\mathcal{N}(0,\sigma^2)$。

因此，我们现在可以写出通过给定的 $\mathbf{x}$ 观测到特定 y 的似然（likelihood）：

$$P(y\mid\mathbf{x})=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(y-\mathbf{w}^\top\mathbf{x}-b)^2\right). \tag{17}$$

现在，根据极大似然估计法，参数 $\mathbf{w}$ 和 b 的最优值是使整个数据集的似然最大的值：

$$P(\mathbf{y}\mid\mathbf{X})=\prod_{i=1}^{n}p(y^{(i)}|\mathbf{x}^{(i)}). \tag{18}$$

根据极大似然估计法选择的估计量称为极大似然估计量。虽然使许多指数函数的乘积最大化看起来很困难，但是我们可以在不改变目标的前提下，通过最大化似然对数来简化。由于历史原因，优化通常是说最小化而不是最大化。我们可以改为最小化负对数似然 $-\log P(\mathbf{y} \mid \mathbf{X})$。由此可以得到的数学公式是：

$$-\log P(\mathbf{y} \mid \mathbf{X}) = \sum_{i=1}^{n} \frac{1}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} - b\right)^2. \tag{19}$$

现在我们只需要假设 σ 是某个固定常数就可以忽略第一项，因为第一项不依赖于 $\mathbf{w}$ 和 b。现在第二项除了常数 $\frac{1}{\sigma^2}$ 外，其余部分和前面介绍的均方误差是一样的。幸运的是，上面式子的解并不依赖于 σ。因此，在高斯噪声的假设下，最小化均方误差等价于对线性模型的极大似然估计。

1.3 从线性回归到深度网络

到目前为止，我们只谈论了线性模型。尽管神经网络涵盖了更多更为丰富的模型，我们依然可以用描述神经网络的方式来描述线性模型，从而把线性模型看作一个神经网络。首先，我们用“层”符号来重写这个模型。

深度学习从业者喜欢绘制图表来可视化模型中正在发生的事情。在图 3 中，我们将线性回归模型描述为一个神经网络。需要注意的是，该图只显示连接模式，即只显示每个输入如何连接到输出，隐去了权重和偏置的值。

在图 3 所示的神经网络中，输入为 $x_1, \ldots, x_d$，因此输入层中的输入数（或称为特征维度，feature dimensionality）为 d。网络的输出为 o_1，因此输出层中的输出数是 1。需要注意的是，输入值都是已经给定的，并且只有一个计算神经元。由于模型重点在发生计算的地方，所以通常我们在计算层数时不考虑输入层。也就是说，图 3 中神经网络的层数为 1。我们可以将线性回归模型视为仅由单个人工神经元组成的神经网络，或称为单层神经网络。

对于线性回归，每个输入都与每个输出（在本例中只有一个输出）相连，我们将这种变换（图 3 中的输出层）称为全连接层（fully-connected layer）或称为稠密层（dense layer）。后面将详细讨论由这些层组成的网络。

2.softmax 回归

回归可以用于预测多少的问题。比如预测房屋被售出价格，或者棒球队可

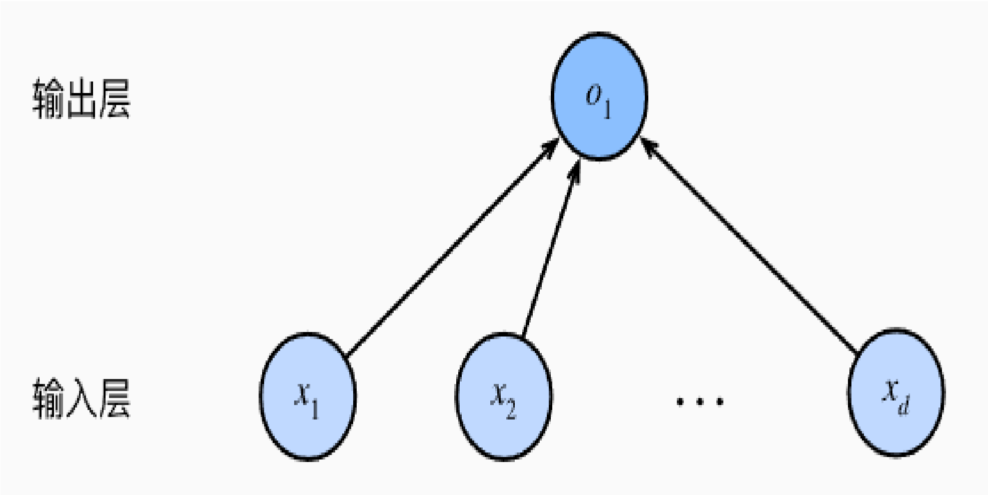

图 3: 线性回归是一个单层神经网络

能获得的胜场数，又或者患者住院的天数。

事实上，我们也对分类问题感兴趣：不是问“多少”，而是问“哪一个”：

* 某个电子邮件是否属于垃圾邮件文件夹？

* 某个图像描绘的是驴、狗、猫、还是鸡？

* 某人接下来最有可能看哪部电影？

通常，机器学习实践者用分类这个词来描述两个有微妙差别的问题：

1. 我们只对样本的“硬性”类别感兴趣，即属于哪个类别；

2. 我们希望得到“软性”类别，即得到属于每个类别的概率。

这两者的界限往往很模糊。其中的一个原因是：即使我们只关心硬类别，我们仍然使用软类别的模型。

2.1 分类问题

我们从一个图像分类问题开始。假设每次输入是一个 2×2 的灰度图像。我们可以用一个标量表示每个像素值，每个图像对应四个特征 x_1, x_2, x_3, x_4。此外，假设每个图像属于类别“猫”“鸡”和“狗”中的一个。

接下来，我们要选择如何表示标签。我们有两个明显的选择：最直接的想法是选择 $y \in \{1, 2, 3\}$，其中整数分别代表 $\{狗, 猫, 鸡\}$。这是在计算机上存

储此类信息的有效方法。如果类别间有一些自然顺序，比如说我们试图预测 {婴儿, 儿童, 青少年, 青年人, 中年人, 老年人}，那么将这个问题转变为回归问题，并且保留这种格式是有意义的。

但是一般的分类问题并不与类别之间的自然顺序有关。幸运的是，统计学家很早以前就发明了一种表示分类数据的简单方法：独热编码（one-hot encoding）。独热编码是一个向量，它的分量和类别一样多。类别对应的分量设置为 1，其他所有分量设置为 0。在我们的例子中，标签 y 将是一个三维向量，其中 $(1,0,0)$ 对应于“猫”、$(0,1,0)$ 对应于“鸡”、$(0,0,1)$ 对应于“狗”：

$$y \in \{(1,0,0),(0,1,0),(0,0,1)\}.$$

2.2 网络架构

为了估计所有可能类别的条件概率，我们需要一个有多个输出的模型，每个类别对应一个输出。为了解决线性模型的分类问题，我们需要定义和输出一样多的仿射函数（affine function）。每个输出对应于它自己的仿射函数。在我们的例子中，由于我们有 4 个特征和 3 个可能的输出类别，我们将需要 12 个标量来表示权重（带下标的 w），3 个标量来表示偏置（带下标的 b）。下面我们为每个输入计算三个未规范化的预测（logit）：o_1、o_2 和 o_3。

$$\begin{aligned}
o_1 &= x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14} + b_1,\\
o_2 &= x_1 w_{21} + x_2 w_{22} + x_3 w_{23} + x_4 w_{24} + b_2,\\
o_3 &= x_1 w_{31} + x_2 w_{32} + x_3 w_{33} + x_4 w_{34} + b_3.
\end{aligned} \tag{20}$$

我们可以用神经网络图 4 来描述这个计算过程。与线性回归一样，softmax 回归也是一个单层神经网络。由于计算每个输出 o_1、o_2 和 o_3 取决于所有输入 x_1、x_2、x_3 和 x_4，所以 softmax 回归的输出层也是全连接层。

为了更简洁地表达模型，我们仍然使用线性代数符号。通过向量形式表达为 $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$，这是一种更适合数学和编写代码的形式。由此，我们已经将所有权重放到一个 3×4 矩阵中。对于给定数据样本的特征 $\mathbf{x}$，我们的输出是由权重与输入特征进行矩阵-向量乘法再加上偏置 $\mathbf{b}$ 得到的。

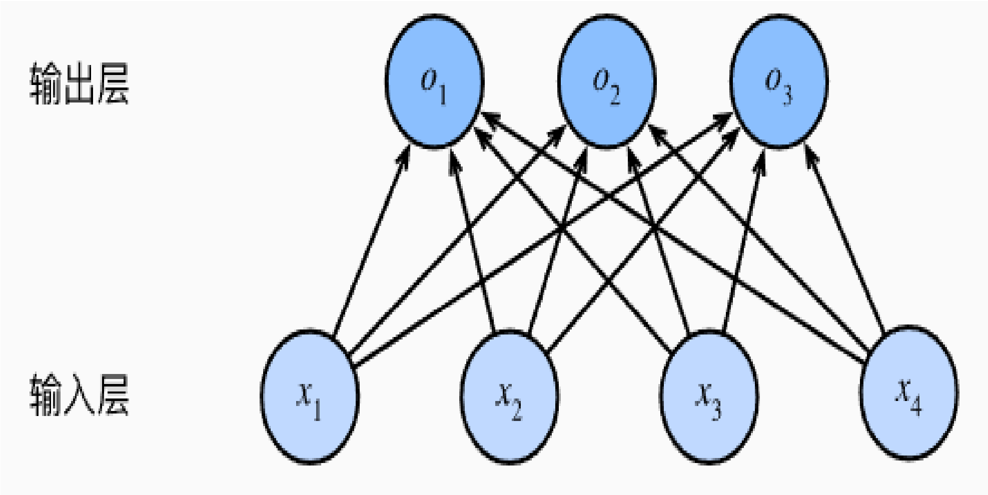

图 4: softmax 回归是一种单层神经网络

2.3 全连接层的参数开销

在深度学习中，全连接层无处不在。然而，顾名思义，全连接层是“完全”连接的，可能有很多可学习的参数。具体来说，对于任何具有 d 个输入和 q 个输出的全连接层，参数开销为 $\mathcal{O}(dq)$，这个数字在实践中可能高得令人望而却步。幸运的是，将 d 个输入转换为 q 个输出的成本可以减少到 $\mathcal{O}(\frac{dq}{n})$，其中超参数 n 可以由我们灵活指定，以在实际应用中平衡参数节约和模型有效性。

2.4softmax 运算

现在我们将优化参数以最大化观测数据的概率。为了得到预测结果，我们将设置一个阈值，如选择具有最大概率的标签。

我们把模型的输出 $\hat{y}_j$ 视为属于类 j 的概率，然后选择具有最大输出值的类别 $\operatorname{argmax}_j y_j$ 作为我们的预测。例如，如果 $\hat{y}_1$、$\hat{y}_2$ 和 $\hat{y}_3$ 分别为 0.1、0.8 和 0.1，那么我们预测的类别是 2，在我们的例子中代表“鸡”。

然而我们能否将未规范化的预测 o 直接视作我们感兴趣的输出呢？答案是否定的。因为将线性层的输出直接视为概率时存在一些问题：

一方面，我们没有限制这些输出数字的总和为 1。

另一方面，根据输入的不同，它们可以为负值。这些违反了概率论中的概

率基本公理。

要将输出视为概率，我们必须保证在任何数据上的输出都是非负的且总和为 1。此外，我们需要一个训练的目标函数，来激励模型精准地估计概率。例如，在分类器输出为 0.5 的所有样本中，我们希望这些样本是刚好有一半实际上属于预测的类别。这个属性叫做校准（calibration）。

社会科学家邓肯·卢斯于 1959 年在选择模型（choice model）的理论基础上发明的 softmax 函数正是这样做的：softmax 函数能够将未规范化的预测变换为非负数并且总和为 1，同时让模型保持可导的性质。

为了完成这一目标，我们首先对每个未规范化的预测求幂，这样可以确保输出非负。为了确保最终输出的概率值总和为 1，我们再让每个求幂后的结果除以它们的总和。如下式：

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \quad \text{其中} \quad \hat{y}_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)} \tag{21}$$

这里，对于所有的 j 总有 $0 \leq \hat{y}_j \leq 1$。因此，$\hat{\mathbf{y}}$ 可以视为一个正确的概率分布。softmax 运算不会改变未规范化的预测 $\mathbf{o}$ 之间的大小次序，只会确定分配给每个类别的概率。因此，在预测过程中，我们仍然可以用下式来选择最有可能的类别。

$$\operatorname*{argmax}_j \hat{y}_j = \operatorname*{argmax}_j o_j. \tag{22}$$

尽管 softmax 是一个非线性函数，但 softmax 回归的输出仍然由输入特征的仿射变换决定。因此，softmax 回归是一个线性模型（linear model）。

2.5 小批量样本的矢量化

为了提高计算效率并且充分利用 GPU，我们通常会对小批量样本的数据执行矢量计算。假设我们读取了一个批量的样本 $\mathbf{X}$，其中特征维度（输入数量）为 d，批量大小为 n。此外，假设我们在输出中有 q 个类别。那么小批量样本的特征为 $\mathbf{X} \in \mathbb{R}^{n \times d}$，权重为 $\mathbf{W} \in \mathbb{R}^{d \times q}$，偏置为 $\mathbf{b} \in \mathbb{R}^{1 \times q}$。softmax 回归的矢量计算

表达式为:

$$\begin{aligned}\mathbf{O} &= \mathbf{X}\mathbf{W} + \mathbf{b},\\ \hat{\mathbf{Y}} &= \mathrm{softmax}(\mathbf{O}).\end{aligned} \tag{23}$$

相对于一次处理一个样本，小批量样本的矢量化加快了 $\mathbf{X}\Box\mathbf{W}$ 的矩阵-向量乘法。由于 $\mathbf{X}$ 中的每一行代表一个数据样本，那么 softmax 运算可以按行（rowwise）执行：在中，$\mathbf{X}\mathbf{W} + \mathbf{b}$ 的求和会使用广播机制，小批量的未规范化预测 $\mathbf{O}$ 和输出概率 $\hat{\mathbf{Y}}$ 都是形状为 $n \times q$ 的矩阵。

2.6 损失函数

接下来，我们需要一个损失函数来度量预测的效果。我们将使用最大似然估计，这与在线性回归中的方法相同。

2.6.1 对数似然

softmax 函数给出了一个向量 $\hat{\mathbf{y}}$，我们可以将其视为"对给定任意输入 $\mathbf{x}$ 的每个类的条件概率"。例如，$\hat{y}_1$=$P(y=$ 猫 $\mid \mathbf{x})$。由前面内容可知 $\mathbf{y}$ 为 one-hot 编码。如果我们表示 $y_j = 1$ 的概率为 $\hat{y}_j$，则 $\mathbf{y}$ 的分布为

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{j=1}^{q} \hat{y}_j^{y_j}$$

其中 $\hat{\mathbf{y}} = (\hat{y}_1, \cdots, \hat{y}_q)$，$\hat{y}_j \geq 0$，$\sum_{i=1}^{q} \hat{y}_j = 1$

假设整个数据集 $\{\mathbf{X}, \mathbf{Y}\}$ 具有 n 个样本，其中索引 i 的样本由特征向量 $\mathbf{x}^{(i)}$ 和独热标签向量 $\mathbf{y}^{(i)}$ 组成。我们可以得到似然函数:

$$P(\mathbf{Y} \mid \mathbf{X}) = \prod_{i=1}^{n} P(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}). \tag{24}$$

根据最大似然估计，我们最大化 $P(\mathbf{Y} \mid \mathbf{X})$，相当于最小化负对数似然:

$$-\log P(\mathbf{Y} \mid \mathbf{X}) = \sum_{i=1}^{n} -\log P(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}) = \sum_{i=1}^{n} l(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}), \tag{25}$$

其中，对于任何标签 $\mathbf{y}$ 和模型预测 $\hat{\mathbf{y}}$，损失函数为:

$$l(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{q} y_j \log \hat{y}_j. \tag{26}$$

在本节稍后的内容会讲到，26 中的损失函数通常被称为交叉熵损失（cross-entropy loss）。由于 $\mathbf{y}$ 是一个长度为 q 的独热编码向量，所以除了一个项以外的所有项 j 都消失了。由于所有 $\hat{y}_j$ 都是预测的概率，所以它们的对数永远不会大于 0。因此，如果正确地预测实际标签，即如果实际标签 $P(\mathbf{y} \mid \mathbf{x}) = 1$，则损失函数不能进一步最小化。注意，这往往是不可能的。例如，数据集中可能存在标签噪声（比如某些样本可能被误标），或输入特征没有足够的信息来完美地对每一个样本分类。

2.6.2 softmax 及其导数

由于 softmax 和相关的损失函数很常见，因此我们需要更好地理解它的计算方式。将 21 代入损失 26 中。利用 softmax 的定义，我们得到：

$$\begin{aligned} l(\mathbf{y}, \hat{\mathbf{y}}) &= - \sum_{j=1}^q y_j \log \frac{\exp(o_j)}{\sum_{k=1}^q \exp(o_k)} \\ &= \sum_{j=1}^q y_j \log \sum_{k=1}^q \exp(o_k) - \sum_{j=1}^q y_j o_j \\ &= \log \sum_{k=1}^q \exp(o_k) - \sum_{j=1}^q y_j o_j. \end{aligned} \tag{27}$$

对未规范化的预测 o_j 求导数，我们得到：

$$\partial_{o_j} l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\exp(o_j)}{\sum_{k=1}^q \exp(o_k)} - y_j = \mathrm{softmax}(\mathbf{o})_j - y_j. \tag{28}$$

换句话说，导数是我们 softmax 模型分配的概率与实际发生的情况（由独热标签向量表示）之间的差异。从这个意义上讲，这与我们在回归中看到的非常相似，其中梯度是观测值 y 和估计值 $\hat{y}$ 之间的差异。这不是巧合，在任何指数族分布模型中，对数似然的梯度正是由此得出的。这使梯度计算在实践中变得容易很多。

2.6.3 交叉熵损失

现在让我们考虑整个结果分布的情况，即观察到的不仅仅是一个结果。对于标签 $\mathbf{y}$，我们可以使用与以前相同的表示形式。唯一的区别是，我们现在用

一个概率向量表示，如 $(0.1, 0.2, 0.7)$，而不是仅包含二元项的向量 $(0, 0, 1)$。我们使用来定义损失 l，它是所有标签分布的预期损失值。此损失称为交叉熵损失（cross-entropy loss），它是分类问题最常用的损失之一。本节我们将通过介绍信息论基础来理解交叉熵损失。

2.7 重新审视交叉熵

如果把熵 $H(P)$ 想象为“知道真实概率的人所经历的惊异程度”，那么什么是交叉熵？交叉熵从 P 到 Q，记为 $H(P, Q)$。我们可以把交叉熵想象为“主观概率为 Q 的观察者在看到根据概率 P 生成的数据时的预期惊异”。当 $P = Q$ 时，交叉熵达到最低。在这种情况下，从 P 到 Q 的交叉熵是 $H(P, P) = H(P)$。

简而言之，我们可以从两方面来考虑交叉熵分类目标：（i）最大化观测数据的似然；（ii）最小化传达标签所需的惊异。

2.8 模型预测和评估

在训练 softmax 回归模型后，给出任何样本特征，我们可以预测每个输出类别的概率。通常我们使用预测概率最高的类别作为输出类别。如果预测与实际类别（标签）一致，则预测是正确的。在接下来的实验中，我们将使用精度（accuracy）来评估模型的性能。精度等于正确预测数与预测总数之间的比率。

三、教学目标考核（60 分钟）

讨论：

1. 线性回归与 Softmax 回归有哪些区别？为什么 Softmax 回归也是线性模型？

2. 假设我们有一些数据 $x_1, \ldots, x_n \in \mathbb{R}$。我们的目标是找到一个常数 b，使得最小化 $\sum_i (x_i - b)^2$。请找到最优值 b 的解析解，这个问题及其解与正态分布有什么关系？

3. 请问使用平方误差的线性回归优化问题的解析解，在什么时候可能比使用随机梯度下降更好？这种方法何时会失效？

4. 我们可以更深入地探讨指数族与 softmax 之间的联系。请计算 softmax 交叉熵损失 $l(\mathbf{y}, \hat{\mathbf{y}})$ 的二阶导数，以及计算 $\mathrm{softmax}(\mathbf{o})$ 给出的分布方差，并与上面计算的二阶导数匹配。

四、总结（15 分钟）

机器学习模型中的关键要素是训练数据、损失函数、优化算法，还有模型本身。最小化目标函数和执行极大似然估计等价。线性回归模型也是一个简单的神经网络。Logistic 回归是深度学习中最基础的非线性模型之一。作为铺垫，在介绍 Logistic 回归以前，首先介绍了线性回归。线性回归的预测目标是连续变量，而 Logistic 回归的预测目标是二元变量。为了应对这一差异，Logistic 回归在线性回归的基础上加入了 Sigmoid 激活函数。softmax 运算获取一个向量并将其映射为概率。softmax 回归适用于分类问题，它使用了 softmax 运算中输出类别的概率分布。交叉熵是一个衡量两个概率分布之间差异的很好的度量，它测量给定模型编码数据所需的比特数。