3.1 线性回归

线性回归输出是一个连续值，因此适用于回归问题。回归问题在实际中很常见，如预测房屋价格、气温、销售额等连续值的问题。与回归问题不同，分类问题中模型的最终输出是一个离散值。我们所说的图像分类、垃圾邮件识别、疾病检测等输出为离散值的问题都属于分类问题的范畴。softmax回归则适用于分类问题。

由于线性回归和softmax回归都是单层神经网络，它们涉及的概念和技术同样适用于大多数的深度学习模型。我们首先以线性回归为例，介绍大多数深度学习模型的基本要素和表示方法。

3.1.1 线性回归的基本要素

我们以一个简单的房屋价格预测作为例子来解释线性回归的基本要素。这个应用的目标是预测一栋房子的售出价格（元）。我们知道这个价格取决于很多因素，如房屋状况、地段、市场行情等。为了简单起见，这里我们假设价格只取决于房屋状况的两个因素，即面积（平方米）和房龄（年）。接下来我们希望探索价格与这两个因素的具体关系。

3.1.1.1 模型定义

设房屋的面积为 x_1，房龄为 x_2，售出价格为 y。我们需要建立基于输入 x_1 和 x_2 来计算输出 y 的表达式，也就是模型（model）。顾名思义，线性回归假设输出与各个输入之间是线性关系：

$$\hat{y} = x_1 w_1 + x_2 w_2 + b$$

其中 w_1 和 w_2 是权重（weight），b 是偏差（bias），且均为标量。它们是线性回归模型的参数（parameter）。模型输出 $\hat{y}$ 是线性回归对真实价格 y 的预测或估计。我们通常允许它们之间有一定误差。

3.1.1.2 模型训练

接下来我们需要通过数据来寻找特定的模型参数值，使模型在数据上的误差尽可能小。这个过程叫作模型训练（model training）。下面我们介绍模型训练所涉及的3个要素。

(1) 训练数据

我们通常收集一系列的真实数据，例如多栋房屋的真实售出价格和它们对应的面积和房龄。我们希望在这个数据上面寻找模型参数来使模型的预测价格与真实价格的误差最小。在机器学习术语里，该数据集被称为训练数据集（training data set）或训练集（training set），一栋房屋被称为一个样本（sample），其真实售出价格叫作标签（label），用来预测标签的两个因素叫作特征（feature）。特征用来表征样本的特点。

假设我们采集的样本数为 n，索引为 i 的样本的特征为 $x_1^{(i)}$ 和 $x_2^{(i)}$，标签为 $y^{(i)}$。对于索引为 i 的房屋，线性回归模型的房屋价格预测表达式为

$$y^{(i)} = x_1^{(i)} w_1 + x_2^{(i)} w_2 + b$$

(2) 损失函数

在模型训练中，我们需要衡量价格预测值与真实值之间的误差。通常我们会选取一个非负数作为误差，且数值越小表示误差越小。一个常用的选择是平方函数。它在评估索引为 i 的样本误差的表达式为

$$\ell^{(i)}(w_1, w_2, b) = \frac{1}{2}\left(y^{(i)} - y^{(i)}\right)^2$$

其中常数 $\frac{1}{2}$ 使对平方项求导后的常数系数为1，这样在形式上稍微简单一些。显然，误差越小表示预测价格与真实价格越相近，且当二者相等时误差为0。给定训练数据集，这个误差只与模型参数相关，因此我们将它记为以模型参数为参数的函数。在机器学习里，将衡量误差的函数称为损失函数（loss function）。这里使用的平方误差函数也称为平方损失（square loss）。

通常，我们用训练数据集中所有样本误差的平均来衡量模型预测的质量，即

$$\ell(w_1, w_2, b) = \frac{1}{n}\sum_{i=1}^{n} \ell^{(i)}(w_1, w_2, b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}\right)^2$$

在模型训练中，我们希望找出一组模型参数，记为 w_1^*, w_2^*, b^*，来使训练样本平均损失最小：

$$w_1^*, w_2^*, b^* = \underset{w_1, w_2, b}{\arg\min}\, \ell(w_1, w_2, b)$$

(3) 优化算法

当模型和损失函数形式较为简单时，上面的误差最小化问题的解可以直接用公式表达出来。这类解叫作解析解（analytical solution）。本节使用的线性回归和平方误差刚好属于这个范畴。然而，大多数深度学习模型并没有解析解，只能通过优化算法有限次迭代模型参数来尽可能降低损失函数的值。这类解叫作数值解（numerical solution）。

在求数值解的优化算法中，小批量随机梯度下降（mini-batch stochastic gradient descent）在深度学习中被广泛使用。它的算法很简单：先选取一组模型参数的初始值，如随机选取；接下来对参数进行多次迭代，使每次迭代都可能降低损失函数的值。在每次迭代中，先随机均匀采样一个由固定数目训练数据样本所组成的小批量（mini-batch）$\mathcal{B}$，然后求小批量中数据样本的平均损失有关模型参数的导数（梯度），最后用此结果与预先设定的一个正数的乘积作为模型参数在本次迭代的减小量。

在训练本节讨论的线性回归模型的过程中，模型的每个参数将作如下迭代：

$$w_1 \leftarrow w_1 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_1} = w_1 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} x_1^{(i)}(x_1^{(i)}w_1 + x_2^{(i)}w_2 + b - y^{(i)}),$$

$$w_2 \leftarrow w_2 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_2} = w_2 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} x_2^{(i)}(x_1^{(i)}w_1 + x_2^{(i)}w_2 + b - y^{(i)}),$$

$$b \leftarrow b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial b} = b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}(x_1^{(i)}w_1 + x_2^{(i)}w_2 + b - y^{(i)}).$$

在上式中，$|\mathcal{B}|$ 代表每个小批量中的样本个数（批量大小，batch size），η 称作学习率（learning rate）并取正数。需要强调的是，这里的批量大小和学习率的值是人为设定的，并不是通过模型训练学出的，因此叫作超参数（hyperparameter）。我们通常所说的“调参”指的正是调节超参数，例如通过反复试错来找到超参数合适的值。在少数情况下，超参数也可以通过模型训练学出。本书对此类情况不做讨论。

3.1.1.3 模型预测

模型训练完成后，我们将模型参数 w_1, w_2, b 在优化算法停止时的值分别记作 $\hat{w}_1, \hat{w}_2, \hat{b}$。注意，这里我们得到的并不一定是最小化损失函数的最优解 w_1^*, w_2^*, b^*，而是对最优解的一个近似。然后，我们就可以使用学出的线性回归模型 $x_1\hat{w}_1 + x_2\hat{w}_2 + \hat{b}$ 来估算训练数据集以外任意一栋面积（平方米）为 x_1、房龄（年）为 x_2 的房屋的价格了。这里的估算也叫作模型预测、模型推断或模型测试。

3.1.2 线性回归的表示方法

我们已经阐述了线性回归的模型表达式、训练和预测。下面我们解释线性回归与神经网络的联系，以及线性回归的矢量计算表达式。

3.1.2.1 神经网络图

在深度学习中，我们可以使用神经网络图直观地表现模型结构。为了更清晰地展示线性回归作为神经网络的结构，图3.1使用神经网络图表示本节中介绍的线性回归模型。神经网络图隐去了模型参数权重和偏差。

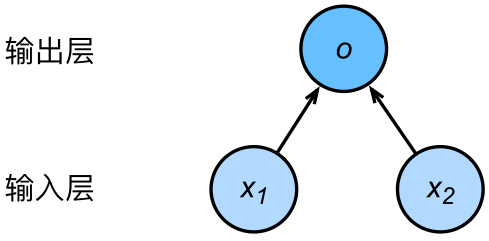

图3.1 线性回归是一个单层神经网络

在图3.1所示的神经网络中，输入分别为 x_1 和 x_2，因此输入层的输入个数为2。输入个数也叫特征数或特征向量维度。图3.1中网络的输出为 o，输出层的输出个数为1。需要注意的是，我们直接将图3.1中神经网络的输出 o 作为线性回归的输出，即 $\hat{y} = o$。由于输入层并不涉及计算，按照惯例，图3.1所示的神经网

络的层数为1。所以，线性回归是一个单层神经网络。输出层中负责计算 o 的单元又叫神经元。在线性回归中，o 的计算依赖于 x_1 和 x_2。也就是说，输出层中的神经元和输入层中各个输入完全连接。因此，这里的输出层又叫全连接层（fully-connected layer）或稠密层（dense layer）。

3.1.2.2 矢量计算表达式

在模型训练或预测时，我们常常会同时处理多个数据样本并用到矢量计算。在介绍线性回归的矢量计算表达式之前，让我们先考虑对两个向量相加的两种方法。a

下面先定义两个1000维的向量。

```
import torch
from time import time

a = torch.ones(1000)
b = torch.ones(1000)
```

向量相加的一种方法是，将这两个向量按元素逐一做标量加法。

```
start = time()
c = torch.zeros(1000)
for i in range(1000):
    c[i] = a[i] + b[i]
print(time() - start)
```

输出：

```
0.02039504051208496
```

向量相加的另一种方法是，将这两个向量直接做矢量加法。

```
start = time()
d = a + b
print(time() - start)
```

输出：

```
0.0008330345153808594
```

结果很明显，后者比前者更省时。因此，我们应该尽可能采用矢量计算，以提升计算效率。

让我们再次回到本节的房价预测问题。如果我们对训练数据集里的3个房屋样本（索引分别为1、2和3）逐一预测价格，将得到

$$\begin{aligned} \hat{y}^{(1)} &= x_1^{(1)} w_1 + x_2^{(1)} w_2 + b, \\ \hat{y}^{(2)} &= x_1^{(2)} w_1 + x_2^{(2)} w_2 + b, \\ \hat{y}^{(3)} &= x_1^{(3)} w_1 + x_2^{(3)} w_2 + b. \end{aligned}$$

现在，我们将上面3个等式转化成矢量计算。设

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ x_1^{(3)} & x_2^{(3)} \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

对3个房屋样本预测价格的矢量计算表达式为$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w} + b$, 其中的加法运算使用了广播机制（参见2.2节）。例如：

```
a = torch.ones(3)
b = 10
print(a + b)
```

输出：

```
tensor([11., 11., 11.])
```

广义上讲，当数据样本数为 n，特征数为 d时，线性回归的矢量计算表达式为

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w} + b$$

其中模型输出 $\hat{\boldsymbol{y}} \in \mathbb{R}^{n \times 1}$ 批量数据样本特征 $\boldsymbol{X} \in \mathbb{R}^{n \times d}$，权重 $\boldsymbol{w} \in \mathbb{R}^{d \times 1}$，偏差 $b \in \mathbb{R}$。相应地，批量数据样本标签 $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$。设模型参数 $\boldsymbol{\theta} = [w_1, w_2, b]^\top$，我们可以重写损失函数为

$$\ell(\boldsymbol{\theta}) = \frac{1}{2n}(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

小批量随机梯度下降的迭代步骤将相应地改写为

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell^{(i)}(\boldsymbol{\theta}),$$

其中梯度是损失有关3个为标量的模型参数的偏导数组成的向量：

$$\nabla_{\boldsymbol{\theta}} \ell^{(i)}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_1} \\ \frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_2} \\ \frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial b} \end{bmatrix} = \begin{bmatrix} x_1^{(i)}(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}) \\ x_2^{(i)}(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}) \\ x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)} \end{bmatrix} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ 1 \end{bmatrix} (\hat{y}^{(i)} - y^{(i)})$$

小结

- 和大多数深度学习模型一样，对于线性回归这样一种单层神经网络，它的基本要素包括模型、训练数据、损失函数和优化算法。
- 既可以用神经网络图表示线性回归，又可以用矢量计算表示该模型。
- 应该尽可能采用矢量计算，以提升计算效率。