1 回归问题

参考:https://zh-v2.d2l.ai/

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。

§ 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：

- 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；
- 任何噪声都比较正常，如噪声遵循正态分布。

根据房屋的面积（平方英尺）和房龄（年）来估算房屋价格（美元）

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:

- 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）。
- 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）。
- 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 $y^{(i)}$。

§ 1.2 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：

$$\text{price} = w_{\text{area}} \cdot \text{area} + w_{\text{age}} \cdot \text{age} + b.$$

其中的 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。

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：

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

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：

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

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。

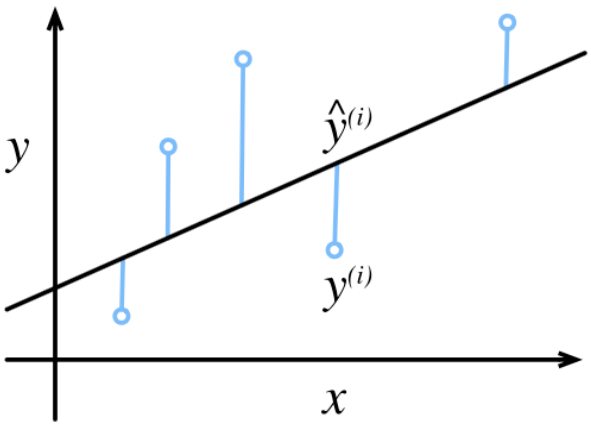
对于特征集合 $\mathbf{X}$，预测值 $\hat{\mathbf{y}} \in \mathbb{R}^n$ 可以通过矩阵-向量乘法表示为：

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

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：

- 一种模型质量的度量方式；
- 一种能够更新模型以提高模型预测质量的方法。

§ 1.3 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：

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

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。

我们为一维情况下的回归问题绘制图像，

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。

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）。

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

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：

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \ L(\mathbf{w}, b).$$

§ 1.4 解析解

线性回归刚好是一个很简单的优化问题。

与我们将在本书中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。

我们的预测问题是最小化 $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$。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。

将损失关于 $\mathbf{w}$ 的导数设为 0，得到解析解：

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

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。

§ 1.5 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。

本书中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。

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:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) = \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{x}^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)} \right),$$

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

$\mathbf{w}$ 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。

2 分类问题

参考:https://zh-v2.d2l.ai/

分类问题：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?
- 某个用户可能注册或不注册订阅服务?
- 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：

- 我们只对样本的“硬性”类别感兴趣，即属于哪个类别；
- 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。(用概率描述离散类别)

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。

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”：

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

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。

我们希望模型的输出 $\hat{y}_j$ 可以视为属于类 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、0.1 和 0.8，那么我们预测的类别是 3，在我们的例子中代表“狗”。

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。

$$o_1 = x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14} + b_1,$$

$$o_2 = x_1 w_{21} + x_2 w_{22} + x_3 w_{23} + x_4 w_{24} + b_2,$$

$$o_3 = x_1 w_{31} + x_2 w_{32} + x_3 w_{33} + x_4 w_{34} + b_3.$$

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。

这些违反了中所说的概率基本公理。

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 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）。

§ 2.1 softmax

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：

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

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。

$$\mathrm{argmax}_j \, \hat{y}_j = \mathrm{argmax}_j \, o_j.$$

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）。

§ 2.2 损失函数

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

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

损失函数被称为交叉熵损失（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。

§ 2.3 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。

利用 softmax 的定义，我们得到：

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

考虑相对于任何未规范化的预测 o_j 的导数，我们得到：

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

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。