神经计算的数学原理

“海棠”系列丛书

作者：刘俊宏

组织：哈尔滨工业大学（深圳）

目录

第一章 回归模型

内容提要

- 线性模型
- 量化评价
- 模型求解
- 多项式回归
- 正则化处理
- 算法实验

让我们以一个**线性回归**（Linear Regression）的例子开始我们的神经计算之旅。假设我们要预测去哈工大的通勤时间，并有如下数据：

- 距离 (Km): [2.7, 4.1, 1.0, 5.2, 2.8]
- 时段 (1: if weekday, 0: if weekend): [1, 1, 0, 1, 0]
- 通勤时间 (min): [25, 33, 15, 45, 22]

我们希望找到一个函数 $f: R^2 \to R$ 满足 $f(D, T_d) \approx T_c$，其中 D 表示距离，T_d 表示时段，T_c 表示通勤时间。

1.1 线性模型

我们首先尝试构建一个线性回归模型来预测通勤时间，其一般表达式如 (1.1) 所示，其中 d 为变量的个数。

$$f(x) = w_0 + w_1 x_1 + \ldots + w_d x_d \tag{1.1}$$

为了后续方便，我们将其转化为矩阵的形式：$f(x) = W^{\mathsf{T}} X$，其中

$$\left[w_0, w_1, w_2, \ldots, w_d\right] = W^{\mathsf{T}}, \begin{bmatrix} x_0 = 1 \\ x_1 \\ \ldots \\ x_d \end{bmatrix} = X$$

笔记 我们约定所有初始向量都是列向量。在上述通勤的例子中，线性回归模型中的参数 $d = 2$，x_1 和 x_2 分别对应于 D 和 T_d 中的元素，w_0 是一个偏置项（Bias Term）。

1.2 量化评价

一旦我们初始了一个线性回归模型，我们就需要一种方法来量化地评价我们模型效果的好坏，即在本例子中我们需要一种方法来评价我们构建的模型是否可以准确地预测通勤时间。在这里，我们使用**残差**（Residual）(1.2) 来衡量第 i 个数据点的期望输出（真实的通勤时间）和模型输出（预测的通勤时间）之间的差距。

$$e^{(i)} = y^{(i)} - W^{\mathsf{T}} x^{(i)} \tag{1.2}$$

同时，我们使用**均方误差** (Mean Square Error, MSE) (1.3) 来综合考虑所有数据点的残差情况。根据**一阶必要最优性条件**（First-Order Necessary Optimality Condition），当均方误差 $C(W)$ 的一阶导数 $C'(W) = 0$ 时 $C(W)$ 取得极值。又根据**二阶充分最优性条件**（Second-Order Sufficiency Conditions），$C(W)$ 的二阶导数 $C''(W) = \frac{1}{n} X^{\mathsf{T}} X$ 恒大于零，所以 $C'(W) = 0$ 时 $C(W)$ 取得最小值。

$$C(W) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - W^{\mathsf{T}} x^{(i)}\right)^2 \tag{1.3}$$

笔记 通常，标准的均方误差表达式为：$C(W) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - W^{\mathsf{T}} x^{(i)}\right)^2$。但在机器学习，特别是在使用梯度下降法进行优化时，将标准的均方误差缩放为 $C(W) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - W^{\mathsf{T}} x^{(i)}\right)^2$ 有利于计算的便利，这在下一小节的计算演示中大家会感受到。

1.3 模型求解

这里使用了一个巧妙地方法求解 $C'(W^*)=0$ 的解析解 W^*，首先我们假设第 i 个数据点的特征向量为 $x^{(i)}$，其中第 i 行表示第 i 个数据点的特征向量，第 j 列表示特征向量中的第 j 个特征。接着我们可以假设一个维度为 $n\times d$ 的特征矩阵 X 和目标输出向量 Y。

$$x^{(i)}=\begin{bmatrix} x_0^{(i)}=1 \\ \cdots \\ x_j^{(i)} \\ \cdots \\ x_d^{(i)} \end{bmatrix}, X=\begin{bmatrix} {x^{(1)}}^{\mathsf{T}} \\ \cdots \\ {x^{(n)}}^{\mathsf{T}} \end{bmatrix}, Y=\begin{bmatrix} y^{(1)} \\ \cdots \\ y^{(n)} \end{bmatrix}$$

接着，我们首先推导 $XW-Y$ 这个公式看看会得到什么：

$$XW-Y=\begin{bmatrix} {x^{(1)}}^{\mathsf{T}} \\ \cdots \\ {x^{(n)}}^{\mathsf{T}} \end{bmatrix} W-\begin{bmatrix} y^{(1)} \\ \cdots \\ y^{(n)} \end{bmatrix}=\begin{bmatrix} {x^{(1)}}^{\mathsf{T}}W-y^{(1)} \\ \cdots \\ {x^{(n)}}^{\mathsf{T}}W-y^{(n)} \end{bmatrix}$$

由此，我们可以得到下述等式：

$$\begin{aligned}(XW-Y)^{\mathsf{T}}(XW-Y)&=\begin{bmatrix} {x^{(1)}}^{\mathsf{T}}W-y^{(1)}, & \cdots & , {x^{(n)}}^{\mathsf{T}}W-y^{(n)} \end{bmatrix}\begin{bmatrix} {x^{(1)}}^{\mathsf{T}}W-y^{(1)} \\ \cdots \\ {x^{(n)}}^{\mathsf{T}}W-y^{(n)} \end{bmatrix} \\ &=\sum_{i=1}^{n}\left({X^{(i)}}^{\mathsf{T}}W-y^{(i)}\right)^2 \\ &=2nC(W)\end{aligned}$$

上述等式 $2nC(W)=(XW-Y)^{\mathsf{T}}(XW-Y)$ 描述了 $C(W)$ 与 X，W，Y 之间的关系，于是我们可以进一步推理得到一个更明确的等式 (1.4)：

$$\begin{aligned}C(W)&=\frac{1}{2n}(XW-Y)^{\mathsf{T}}(XW-Y) \\ &=\frac{1}{2n}(W^{\mathsf{T}}X^{\mathsf{T}}-Y^{\mathsf{T}})(XW-Y) \\ &=\frac{1}{2n}(W^{\mathsf{T}}X^{\mathsf{T}}XW-W^{\mathsf{T}}X^{\mathsf{T}}Y-Y^{\mathsf{T}}XW+Y^{\mathsf{T}}Y) \\ &=\frac{1}{2n}(W^{\mathsf{T}}X^{\mathsf{T}}XW-2W^{\mathsf{T}}X^{\mathsf{T}}Y+Y^{\mathsf{T}}Y)\end{aligned} \tag{1.4}$$

笔记 在推导等式 (1.4) 的时候可能会对最后两步推导过程感到疑惑：

$$\frac{1}{2n}(W^{\mathsf{T}}X^{\mathsf{T}}XW-W^{\mathsf{T}}X^{\mathsf{T}}Y-Y^{\mathsf{T}}XW+Y^{\mathsf{T}}Y)=\frac{1}{2n}(W^{\mathsf{T}}X^{\mathsf{T}}XW-2W^{\mathsf{T}}X^{\mathsf{T}}Y+Y^{\mathsf{T}}Y)$$

这里解释一下，这是因为 $W^{\mathsf{T}}X^{\mathsf{T}}Y$ 实际上会得到一个实数，并且 $W^{\mathsf{T}}X^{\mathsf{T}}=(XW)^{\mathsf{T}}$。于是我们可以获得等价关系：

$$W^{\mathsf{T}}X^{\mathsf{T}}Y=(W^{\mathsf{T}}X^{\mathsf{T}}Y)^{\mathsf{T}}=Y^{\mathsf{T}}XW$$

笔记 同时，请注意看 X，W，Y 的组织形式，所以这里写成 $XW-Y$ 的形式才是正确并方便我们的推导，而不是 $Y-W^{\mathsf{T}}X$ 的形式。

通过等式 (1.4) 我们可以进一步求解 $C'(W)$ 与 X,W,Y 之间的关系。我们首先通过观察可以发现：1. $W^{\mathsf{T}}X^{\mathsf{T}}XW$ 是二次项，2. $W^{\mathsf{T}}X^{\mathsf{T}}Y$ 是一次项，3. $Y^{\mathsf{T}}Y$ 是常数项。于是我们便可以快速得到等式 (1.5)：

$$C'(W)=\frac{1}{2n}(2X^{\mathsf{T}}XW-2X^{\mathsf{T}}Y) \tag{1.5}$$

根据上一小节的推论，当 $C'(W^*)=0$ 时取得最优的 W^*，此时 $C(W)$ 取得最小值。于是根据等式 (1.5) 可以得到

$X^\mathsf{T}XW^* = X^\mathsf{T}Y$。当 $X^\mathsf{T}X$ 可逆时便可以获得 W^* 的解析解 (1.6) 和模型的输出 Y^* (1.6)：

$$\begin{cases} W^* = (X^\mathsf{T}X)^{-1}X^\mathsf{T}Y \\ Y^* = XW^* = X(X^\mathsf{T}X)^{-1}X^\mathsf{T}Y \end{cases} \tag{1.6}$$

证明 看完了 W^* 的解析解的整个推导过程，有些读者可能会有这样一个疑问：为什么 W 作为一个向量却可以将其视为一个“整体”进行求导操作？这涉及到多元微积分和矩阵求导的核心概念，下面开始阐述其数学原理。

1. 标量对向量的导数：梯度的定义

当函数 $f(X)$ 的输出是一个标量，而输入 X 是一个向量时，其导数被称为梯度（Gradient），记作 $\nabla_X f(X)$。梯度本身是一个向量，其每个分量是函数 f 对 X 的每个分量的偏导数。

例如，如果 $W = [w_0, w_1, ..., w_d]^\mathsf{T}$，那么：

$$\nabla_W C(W) = \frac{\partial C(W)}{\partial W} = \left[\frac{\partial C(W)}{\partial w_0}, \frac{\partial C(W)}{\partial w_1}, \cdots, \frac{\partial C(W)}{\partial w_d}\right]^\mathsf{T}$$

2. 为什么推导过程看起来像是在对“一维未知量”求导？

要回答这个问题，我们需要理解的一个关键问题是：为什么 XW（或 $W^\mathsf{T}X^\mathsf{T}$）对 W 的梯度是 X（或 X^T）。

我们假设

$$Z(W) = XW$$

其中：

$$X = \begin{bmatrix} x_0^{(1)}, x_1^{(1)}, ..., x_d^{(1)} \\ ... \\ x_0^{(n)}, x_1^{(n)}, ..., x_d^{(n)} \end{bmatrix}, W = \begin{bmatrix} w_0 \\ ... \\ w_d \end{bmatrix}$$

接着我们对 $Z(W)$ 求一阶导，即求 $Z(W)$ 的雅可比矩阵：

$$\nabla_W Z(W) = \frac{\partial Z_i}{\partial W_k} = \frac{\partial(\sum_{j=0}^{d} X_{ij}W_j)}{\partial W_k} = X_{ik}$$

即

$$J = \begin{bmatrix} \frac{\partial Z_1}{\partial w_0} & \frac{\partial Z_1}{\partial w_1} & \cdots & \frac{\partial Z_1}{\partial w_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Z_n}{\partial w_0} & \frac{\partial Z_n}{\partial w_1} & \cdots & \frac{\partial Z_n}{\partial w_d} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(n)} & x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} = X$$

所以雅可比矩阵的第 i 行就是 X 的第 i 行，即证：

$$\frac{\partial XW}{\partial W} = X$$

1.4 多项式回归

在推导出线性回归模型的解后，我们其实也可以用上述公式求解多项式回归的 W^* 的解析解。多项式回归是一种基础的非线性回归模型，其基本形式如公式 (1.7) 所示：

$$f(x) = w_0 + w_1x_1 + w_2x^2 + ... + w_mx^m = W^\mathsf{T}\phi(x) \tag{1.7}$$

其中 $\phi(x) = \left[1, x, x^2, \cdots, x^m\right]^\mathsf{T}$，我们同样设：

$$X = \begin{bmatrix} \phi(x)^{(1)\mathsf{T}} \\ ... \\ \phi(x)^{(n)\mathsf{T}} \end{bmatrix}, Y = \begin{bmatrix} y^{(1)} \\ ... \\ y^{(n)} \end{bmatrix}$$

同样可以得到相同的解析解形式：

$$\begin{cases} W^* = (X^\mathsf{T}X)^{-1}X^\mathsf{T}Y \\ Y^* = XW^* = X(X^\mathsf{T}X)^{-1}X^\mathsf{T}Y \end{cases}$$

1.5 正则化处理

理想情况下，我们希望模型在训练样本上表现良好同时又不至于太复杂，从而实现良好的泛化（既不过拟合也不过欠拟合）。解决这个问题的一种方法是鼓励较小的权重（这样任何特征都不会对预测产生太大的影响）。这被称为**正则化**（Regularization）。

给定一个数据集 $S = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ 和一个正则化强度 $\lambda > 0$，我们同样希望找到下列**正则化最小二乘回归**函数（Regularized Least Square Regression）(1.8)，也被称为岭回归（Ridge Regression），的最小值：

$$C(W) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - W^{\mathsf{T}}X^{(i)}\right)^2 + \frac{\lambda}{2}||W||_2^2 \tag{1.8}$$

其中 $||W||_2$ 表示 W 的二范数（2-norm）。现在函数 (1.8) 有两个优化目标：函数右边第一项 $\frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - W^{\mathsf{T}}X^{(i)}\right)^2$ 是误差项（经验风险），用来衡量模型对训练数据的拟合程度；函数右边第二项 $\frac{\lambda}{2}||W||_2^2$ 是正则化项（结构风险），用来对模型的复杂度进行惩罚，当模型的某些权重变得过大，正则化项会变大并超过误差项的影响，从而导致总成本 $C(W)$ 变大。

定义 1.1 (范数)

对于一个向量 $v = (v_1, ..., v_d) \in R^d$，范数被定义为：

$$||V||_n = (|v_1|^n + |v_2|^n + ... + |v_d|^n)^{\frac{1}{n}}$$

二范数也被称为欧几里得范数（Euclidean Norm）。

接下来我们开始求解上述函数的 W^* 值，前面我们已经证明过 $2nC(W) = (XW - Y)^{\mathsf{T}}(XW - Y)$，于是我们便可以得知：

$$C(W) = \frac{1}{2n}(XW - Y)^{\mathsf{T}}(XW - Y) + \frac{\lambda}{2}||W||_2^2$$

同时通过公式 (1.5)，我们可知 $C(W)$ 的一阶导为：

$$\begin{aligned}
C'(W) &= \frac{1}{n}(X^{\mathsf{T}}XW - X^{\mathsf{T}}Y) + (||W||_2^2)' \\
&= \frac{1}{n}(X^{\mathsf{T}}XW - X^{\mathsf{T}}Y) + ((|w_0|^2 + |w_1|^2 + ... + |w_d|^2)^{\frac{1}{2}\times 2})' \\
&= \frac{1}{n}(X^{\mathsf{T}}XW - X^{\mathsf{T}}Y) + (|w_0|^2 + |w_1|^2 + ... + |w_d|^2)' \\
&= \frac{1}{n}(X^{\mathsf{T}}XW - X^{\mathsf{T}}Y) + 2(|w_0| + |w_1| + ... + |w_d|) \\
&= \frac{1}{n}(X^{\mathsf{T}}XW - X^{\mathsf{T}}Y) + 2W
\end{aligned}$$

将 $C'(W^*) = 0$ 代入上式可得：

$$\begin{aligned}
\frac{1}{n}(X^{\mathsf{T}}XW^* - X^{\mathsf{T}}Y) + \lambda W^* &= 0 \\
\frac{1}{n}X^{\mathsf{T}}XW^* + \lambda W^* &= \frac{1}{n}X^{\mathsf{T}}Y \\
(\frac{1}{n}X^{\mathsf{T}}X + \lambda\mathbb{I})W^* &= \frac{1}{n}X^{\mathsf{T}}Y \\
W^* &= (\frac{1}{n}X^{\mathsf{T}}X + \lambda\mathbb{I})^{-1}(\frac{1}{n}X^{\mathsf{T}}Y)
\end{aligned}$$

于是我们得到了公式 (1.8) 的解析解 $W^* = (\frac{1}{n}X^{\mathsf{T}}X + \lambda\mathbb{I})^{-1}(\frac{1}{n}X^{\mathsf{T}}Y)$。这个正则化后的解析解有一个非常好的性质那就是不要求 $X^{\mathsf{T}}X$ 是可逆的（invertible），因为 $\frac{1}{n}X^{\mathsf{T}}X + \lambda\mathbb{I}$ 几乎总是可逆的（一个矩阵可逆的充要条件是它的所有特征值都不为零，而 $\frac{1}{n}X^{\mathsf{T}}X + \lambda\mathbb{I}$ 的所有特征值都严格大于零。）

1.6 算法实验

现在让我们回到最开头的案例，我们想预估我们上学的通勤时间，现在我们有下列数据：

- 距离 (Km): [2.7, 4.1, 1.0, 5.2, 2.8]
- 时段 (1: if weekday, 0: if weekend): [1, 1, 0, 1, 0]
- 通勤时间 (min): [25, 33, 15, 45, 22]

代如上述公式 (1.6) 求解出的输出应该为：

$$W^* = \begin{bmatrix} 6.08613445378149 \\ 6.53361344537816 \\ 2.11274509803922 \end{bmatrix}, Y^* = \begin{bmatrix} 25.8396358543418 \\ 34.9866946778712 \\ 12.6197478991597 \\ 42.1736694677872 \\ 24.3802521008403 \end{bmatrix}, Y = \begin{bmatrix} 25 \\ 33 \\ 15 \\ 45 \\ 22 \end{bmatrix}$$

其中 W^* 为模型的最优权重，Y^* 为模型的最优预测输出，Y 为模型的期望输出（即真实数据）。上述例子在 GitHub 仓库提供了 Matlab 代码（Linear_Regression.m）实现，其可视化的结果如图 1.1 所示。

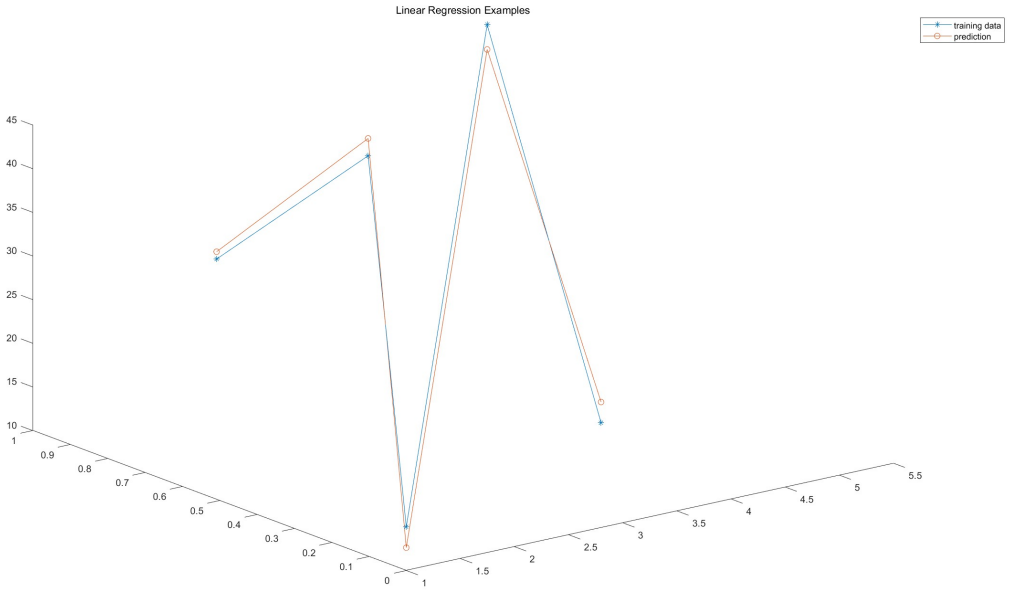

图 1.1: 通勤时间的回归建模结果

第二章 梯度下降

内容提要

- 随机梯度下降

梯度下降（Gradient Descent）是最小化目标函数 $C(W)$ 的通用算法之一（由柯西于 1847 年首次提出）。它是一种迭代算法，从 $W^{(t)}$ 开始，每次迭代都会产生一个新的 $W^{(t+1)}$，其公式 (2.1) 如下：

$$W^{(t+1)} = W^{(t)} - \eta_t \nabla C(W^{(t)}),\ (t = 0, 1, ..., n) \tag{2.1}$$

其中 η_t 称为**学习率**（Learning Rate）或**步长**（Step Size）。因此梯度下降 (2.1) 使用负梯度作为搜索方向，并沿着该方向移动 η_t 的距离。

接下来我们证明梯度下降 (2.1) 的合理性与可行性。根据一阶泰勒近似 $f(x) \approx f(a) + f'(a)(x - a)$ 的形式，那么存在下列近似关系：

$$C(W^{(t+1)}) \approx C(W^{(t)}) + \nabla C(W^{(t)})^{\mathsf{T}}(W^{(t+1)} - W^{(t)})$$

接着将公式 (2.1) 代入上述式子中可以得到：

$$\begin{aligned} C(W^{(t+1)}) &\approx C(W^{(t)}) + \nabla C(W^{(t)})^{\mathsf{T}}(W^{(t)} - \eta \nabla C(W^{(t)}) - W^{(t)}),\ (\eta \to 0) \\ &\approx C(W^{(t)}) + \nabla C(W^{(t)})^{\mathsf{T}}(-\eta \nabla C(W^{(t)})) \\ &\approx C(W^{(t)}) - \eta \nabla C(W^{(t)})^{\mathsf{T}} \nabla C(W^{(t)}) \end{aligned}$$

因为 $\nabla C(W)^{\mathsf{T}} \nabla C(W) = ||\nabla C(W)||_2^2 > 0$，所以即证 $C(W^{(t+1)}) < C(W^{(t)})$。

笔记 我们现在可以尝试利用梯度下降法来求解线性回归问题。

对于最小二乘回归 (1.4)，我们已经得到了 $C(W)$ 的一阶导形式 (1.5)，套用梯度下降公式 (2.1) 可以直接得到 W^* 数值解的求解方式：

$$\begin{aligned} W^{(t+1)} &= W^{(t)} - \eta_t \nabla C(W^{(t)}) \\ &= W^{(t)} - \frac{\eta}{n}(X^{\mathsf{T}} X W^{(t)} - X^{\mathsf{T}} Y) \end{aligned}$$

继续套用通勤时间计算的例子，GitHub 仓库提供了 Matlab 代码（Gradient_Descent_for_Linear_Regression.m）实现。

2.1 随机梯度下降