DW_共读AI圣经：第四章 Part 1 线性回归

第一章中有一个核心思想：

> 机器学习就是从数据中学习概率

这里的概率需要通过函数（抑或是称为模型）来求解，那么”线性“就是最简单的函数关系。
通过线性模型的数学原理推导，我们就可以看到机器学习从数学角度来说到底在做什么。
虽然更复杂的回归模型我们已经无法人力计算，但是从思路上是一致的。
这就是学习本章的意义：**以线性回归模型为例，探究机器学习的数学本质。**

目录

细节推导和概念引入

问题引入

$$y(x,w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$

这是一个简单的线性函数可以用来描述一个线性回归模型，其中含有自变量的部分叫做**基函数。**
（常见基函数在后面介绍）

$$y(x,w) = \sum_{j=0}^{M-1} w_j \phi_j(x)$$

加入一个偏置参数就可以统一为以上函数，就可以看做是一个简单的神经网络。

那么核心问题就是如何确定该函数中的参数 w

似然函数和误差函数的关系推导细节

需要预测的目标变量 t 是由包含高斯噪声的函数 $y(x,w)$ 给出的

$$t=y(\boldsymbol{x}, \boldsymbol{w})+\varepsilon$$

其中 ε 是方差 σ^2 的零均值高斯随机变量，那么预测值的概率求解就可以是

$$p\left(t \mid \boldsymbol{x}, \boldsymbol{w}, \sigma^2\right)=\mathcal{N}\left(t \mid y(\boldsymbol{x}, \boldsymbol{w}), \sigma^2\right)$$

当我们有多个观测数据时，就可以把 t 看成时一个列向量，由于数据点都是从分布式中独立产生的，那么就可以得到似然函数

$$p\left(\boldsymbol{t} \mid \boldsymbol{X}, \boldsymbol{w}, \sigma^2\right)=\prod_{n=1}^N \mathcal{N}\left(t_n \mid \boldsymbol{w}^{\mathrm{T}} \phi\left(\boldsymbol{x}_n\right), \sigma^2\right)$$

以下证明最大化似然函数就是最小化误差函数，可以由此确定参数的值：

证明：在高斯噪声分布下， $t = y(x, w) + \varepsilon$

(不属于严格的等价证明) 最大化似然函数等价于最小化误差函数.

假设观测值 t_n 服从高斯分布，其均值为模型预测值 $w^T\phi(x_n)$，方差为 σ^2：

$$p(t_n | x_n, w, \sigma^2) = N(t_n | w^T\phi(x_n), \sigma^2).$$

① 假设 n 个样本独立同分布：(概率条件).

$$p(t | x_n, w, \sigma^2) = \prod_{n=1}^{N} N(t_n | w^T\phi(x_n), \sigma^2)$$

② 取对数 (运算技巧)

$$\ln p(t | x_n, w, \sigma^2) = \sum_{n=1}^{N} \ln[N(t_n | w^T\phi(x_n), \sigma^2)]$$

③ 展开高斯分布的概率密度函数

$$\ln N(\cdot) = \ln\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] + \ln\left[\exp\left(-\frac{(t_n - w^T\phi(x_n))^2}{2\sigma^2}\right)\right]$$

$= e^{-(\ )}$.

④ 简化对数 (涉及对数的基本运算)

$$\ln N(\cdot) = -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma^2) - \frac{1}{\sigma^2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2$$

⑤ 代入原式.

$$\ln p(t | x_n, w, \sigma^2) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2$$

$$= -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{\sigma^2}E_D(w)$$

前一个task 讲到的误差函数.

故最大化似然函数

需要最小化误差函数.

似然函数的求解

明确目标：求解似然函数的最大值，下面就是数学计算的问题了。

我们选择求梯度的方式来求解：

$$\nabla_{\boldsymbol{w}} \ln p(\boldsymbol{t} \mid \boldsymbol{X}, \boldsymbol{w}, \sigma^2) = \frac{1}{\sigma^2}\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_n)\}\phi(\boldsymbol{x}_n)^{\mathrm{T}}$$

用梯度的方法求解最大似然.（以求解w为例）.

$$\nabla_W \ln p(t|X,w,\sigma^2) = \nabla_W\left[-\frac{1}{\sigma^2}E_D(w)\right]$$ 实际是求偏导的运算

$$= \nabla_W\left[-\frac{1}{\sigma^2}\frac{\sum_{n=1}^{N}(t-w^T\phi(x_n))^2}{2}\right]$$

$$= \nabla_W\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left[t^2 + (w^T\phi(x_n))^2 - 2tw^T\phi(x_n)\right]\right\}$$

求w的偏导

$\nabla_W(w^Ta) = a$ $\nabla_W(w^TAw) = (A+A^T)w$

$$= -\frac{1}{2\sigma^2}\sum_{n=1}^{N}\{2\phi(x_n)\phi(x_n)w - 2t_n\phi(x_n)\}$$

$$= -\frac{1}{\sigma^2}\sum_{n=1}^{N}\left[t_n\phi(x_n) - \phi(x_n)\phi(x_n)^Tw\right]$$ （一些向量运算）

$\phi(x_n)\phi(x_n)^Tw = \phi(x_n)(\phi(x_n)^Tw)$.

$$= \frac{1}{\sigma^2}\sum_{n=1}^{N}\left[t_n - w^T\phi(x_n)\right]\phi(x_n)$$ （因式分解）

（后续步骤书中已给出）.

当梯度设为0的时候，就可以进行求解：

$$0 = \sum_{n=1}^{N} t_n\phi(\boldsymbol{x}_n)^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}\left(\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^{\mathrm{T}}\right)$$

$$\boldsymbol{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{t}$$

同理也可以求得其他参数。

补充信息

其他常见的基函数

在深度学习中，**基函数**通常指神经网络中用于引入非线性的**激活函数**。它们是神经网络能够学习复杂模式的关键组件。以下是几种常用基函数：

🧠 1. Sigmoid (Logistic Function)

- **公式**:
  `σ(z) = 1 / (1 + exp(-z))`
- **输出范围**: `(0, 1)`
- **特点**:
  - 将输入压缩到0-1之间，适合输出概率（二分类输出层）。
  - 历史重要，但现代深层网络较少使用。
- **缺点**:
  - **梯度消失**：当输入绝对值较大时，梯度接近0，导致深层网络训练困难。
  - **非零中心输出**：可能导致后续层输入分布偏移。

🔄 2. Tanh (Hyperbolic Tangent)

- **公式**:
  `tanh(z) = (exp(z) - exp(-z)) / (exp(z) + exp(-z))`
  (或 `2 * sigmoid(2z) - 1`)
- **输出范围**: `(-1, 1)`
- **特点**:
  - 类似Sigmoid，但**输出以0为中心**。
  - 梯度比Sigmoid稍强（最大梯度为1）。
- **缺点**:
  - 仍然存在**梯度消失**问题（尤其在绝对值大的区域）。
  - 常用于RNN、LSTM等循环网络。

⚡ 3. ReLU (Rectified Linear Unit)

- **公式**:
  `ReLU(z) = max(0, z)`
- **输出范围**: `[0, ∞)`
- **特点**:
  - **计算高效**：仅需比较和阈值操作。
  - **缓解梯度消失**：正区间梯度恒为1。
  - **稀疏激活**：约50%神经元在训练中被激活。
  - **现代深度网络最常用**。
- **缺点**:
  - **Dying ReLU问题**：输入为负时梯度为0，神经元可能永久"死亡"。
  - 输出非零中心。

💧 4. Leaky ReLU

- **公式**:
  `LeakyReLU(z) = max(αz, z)`
  (α 是一个小的正数，如0.01)
- **输出范围**: `(-∞, ∞)`
- **特点**:
  - 解决ReLU的"死亡"问题：负区间有微小梯度(α)。
  - 保留了ReLU在正区间的优点。
- **变种**:
  - **Parametric ReLU (PReLU)**：α作为可学习参数。

🌊 5. ELU (Exponential Linear Unit)

- **公式**:
  `ELU(z) = { z, if z > 0; α(exp(z) - 1), if z ≤ 0 }`
  (α通常设为1)
- **输出范围**: `(-α, ∞)`
- **特点**:
  - 负区间平滑渐近到-α，缓解Dying ReLU问题。
  - 输出接近零中心化。
  - 理论上可能比ReLU有更好的表现。
- **缺点**:

- 计算量稍大（涉及指数运算）。

🧪 6. Swish

- **公式**:
  `Swish(z) = z * sigmoid(βz)`
  (β可以是常数或可学习参数，常取1)
- **输出范围**: `(-∞, ∞)`
- **特点**:
  - 由Google提出，在部分任务上表现优于ReLU。
  - **平滑且非单调**（负区间有微小"凸起"）。
  - 负值不会被完全抑制。

📊 7. GELU (Gaussian Error Linear Unit)

- **公式** (近似):
  `GELU(z) ≈ 0.5 * z * (1 + tanh(√(2/π) * (z + 0.044715z³)))`
  (或使用Sigmoid近似)
- **输出范围**: `(-∞, ∞)`
- **特点**:
  - 受随机正则化思想启发（如Dropout）。
  - 在**Transformer模型**（如BERT, GPT）中广泛使用。
  - 表现常优于ReLU/ELU。

选择建议 & 总结

函数	适用场景	主要优势	主要劣势
Sigmoid	二分类输出层	输出概率直观	梯度消失严重
Tanh	RNN/LSTM隐藏层	零中心输出	梯度消失
ReLU	大多数CNN/MLP隐藏层 (默认首选)	计算高效，缓解梯度消失	Dying ReLU问题
Leaky ReLU/ELU	担心Dying ReLU时	解决负区间问题	计算稍复杂

函数	适用场景	主要优势	主要劣势
Swish/GELU	追求更高性能 (尤其 Transformer)	平滑，理论性质好	计算量最大

核心作用：引入非线性，使神经网络能够逼近任意复杂函数。没有它们，深度网络将退化为线性模型。