Note of PRML and Scikit-learn

Subway Chan

2018 年 10 月 24 日

目录

1 Introduction **3**

1.1 main body 3

1.1.1 term 3

1.1.2 Paradigms of Machine Learning 4

1.1.3 学习理论 5

1.2 Example: Polynomial Curve Fitting 5

1.3 Probability Theory 5

1.3.1 贝叶斯概率 5

1.3.2 高斯分布 8

1.4 Model Selection 14

1.4.1 **TODO** 交叉验证 (cross validation) 14

1.5 The Curse of Dimensionality 14

1.6 Decision Theory 14

1.6.1 回归问题的损失函数 14

1.7 Notes 17

1.7.1 Curve fitting 为例子演示三种方法 17

2 Probability Distributions **17**

3 Linear Models for Regression **17**

3.1 Abstract 17

3.2 线性基函数模型 18

3.2.1 main body 19

3.2.2 最大似然与最小平方 21

List of Listings

1.1 ch01-init 12

1.2 generate data 12

1 Introduction

1.1 main body

PRML$_{\text{Chinesevision.pdf}}$, p. 9

1.1.1 term

1. 训练集 (training set)

   一个由 N 个数字 $\{x_1, \cdots, x_N\}$ 组成的大的集合被叫做训练集 (training set), 用来调节模型的参数。

2. 目标向量 (target vector)

   我们可以使用目标向量 (target vector) $\mathbf{t}$ 来表示数字的类别, 它代表对应数字的标签。

3. 训练 (training)& 学习 (learning)

   函数 $\mathbf{y}(\mathbf{x})$ 的精确形式在训练 (training) 阶段被确定, 这个阶段也被称为学习 (learning) 阶段, 以训练数据为基础。一旦模型被训练出来, 它就能确定新的数字的图像集合中图像的标签。

4. 测试集 (test set)

   这些新的数字的图像集合组成了测试集 (test set)。

5. 泛化 (generalization)

   正确分类与训练集不同的新样本的能力叫做泛化 (generalization)。

6. 特征抽取 (feature extraction)

   这个预处理阶段有时被叫做特征抽取 (feature extraction)。

7. 监督学习 (supervised learning)

   训练数据的样本包含输入向量以及对应的目标向量的应用叫做有监督学习 (supervised learning) 问题。

   (a) 分类 (classification) 问题

   (b) 回归 (regression)

8. 无监督学习 (unsupervised learning)

   在其他的模式识别问题中, 训练数据由一组输入向量 x 组成, 没有任何对应的目标值。

   (a) 聚类 (clustering)

   目标可能是发现数据中相似样本的分组, 这被称为聚类 (clustering)

   (b) 密度估计 (density estimation)

   决定输入空间中数据的分布

   (c) 数据可视化 (visualization)。

   把数据从高维空间投影到二维或者三维空间

9. **TODO** 反馈学习 (reinforcement learning)

   PRML$_{\text{Chinesevision.pdf}}$, p. 10

1.1.2 Paradigms of Machine Learning

1. supervised learning

   Given $\mathcal{D} = \{\mathbf{X}_i, \mathbf{Y}_i\}$

   learn $f(\cdot) : \mathbf{Y}_i = f(\mathbf{X}_i)$

   s.t. $\mathcal{D}^{\text{new}} = \{\mathbf{X}_j\} \Rightarrow \{\mathbf{Y}_j\}$

2. unsupervised learning

   Given $\mathcal{D} = \{\mathbf{X}_i$

   learn $f(\cdot) : \mathbf{Y}_i = f(\mathbf{X}_i)$

   s.t. $\mathcal{D}^{\text{new}} = \{\mathbf{X}_j\} \Rightarrow \{\mathbf{Y}_j\}$

3. reinforcement learning

Given $\mathcal{D} = \{\text{env, actions, rewards, simulator/trace/real game}\}$

learn policy : $\boldsymbol{e}, \boldsymbol{r} \to \boldsymbol{a}$, utility : $\boldsymbol{a}, \boldsymbol{e} \to \boldsymbol{r}$

s.t. $\{\text{env, new real game}\} \Rightarrow \boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \ldots$

4. supervised learning

Given $\mathcal{D} \sim G(\cdot)$

learn $\mathcal{D}^{\text{new}} \sim G'(\cdot) \text{and} f(\cdot)$

s.t. $\mathcal{D}^{\text{all}\prime}(\cdot), \text{policy}, \{\mathbf{Y}_j\}$

1.1.3 学习理论

一套标准的框架, 用统计学, 概率论, 数学的严格化语言去解释 (收敛速率与泛化性能) 或者比较不同学习⊠法不模型的性能。其中最经典的例子: 统计学习理论。统计学习理论的目标: 去研究所谓的泛化误差界 (Generalization Bounds) PRML 读书会合集打印版.pdf, p. 8

1.2 Example: Polynomial Curve Fitting

1.3 Probability Theory

1.3.1 贝叶斯概率

PRML$_{\text{Chinesevision.pdf}}$, p. 22

1. 贝叶斯概率

贝叶斯定理现在有了一个新的意义。回忆一下, 在水果盒子的例子中, 水果种类的观察提供了相关的信息, 改变了选择了红盒子的概率。在那个例子中, 贝叶斯定理通过将观察到的数据融合, 来把先验概率转化为后验概率。正如我们将看到的, 在我们对数量 (例如多项式曲线拟合例子中的参数 w) 进行推断时, 我们可以采用一个类似的方法。在观察到数据之前, 我们有一些关于参数 w 的假设, 这以先验概率 $p(w)$ 的形式给出。观测数据 $\mathcal{D} = \{t_1, \cdots, t_N\}$ 的效果可以通过条件概率 $p(D|w)$

表达, 我们将在 1.2.5 节看到这个如何被显式地表达出来。贝叶斯定理的形式为

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

它让我们能够通过后验概率 $p(w|D)$, 在观测到 D 之后估计 w 的不确定性。

贝叶斯定理右侧的量 $p(D|w)$ 由观测数据集 D 来估计, 可以被看成参数向量 w 的函数, 被称为似然函数 (likelihood function)。它表达了在不同的参数向量 w 下, 观测数据出现的可能性的大小。注意, 似然函数不是 w 的概率分布, 并且它关于 w 的积分并不 (一定) 等于 1。

给定似然函数的定义, 我们可以用自然语言表述贝叶斯定理

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{1.1}$$

其中所有的量都可以看成 w 的函数。

公式 (1.1) 的分母是一个归一化常数, 确保了左侧的后验概率分布是一个合理的概率密度, 积分为 1。实际上, 对公式 (1.1) 的两侧关于 w 进行积分, 我们可以用后验概率分布和似然函数来表达贝叶斯定理的分母

$$p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{1.2}$$

2. 贝叶斯频率学比较

在贝叶斯观点和频率学家观点中, 似然函数 $p(D|w)$ 都起着重要的作用。然而, 在两种观点中, 使用的方式有着本质的不同。在频率学家的观点中, w 被认为是一个固定的参数, 它的值由某种形式的“估计”来确定, 这个估计的误差通过考察可能的数据集 D 的概率分布来得到。相反, 从贝叶斯的观点来看, 只有一个数据集 D (即实际观测到的数据集), 参数的不确定性通过 w 的概率分布来表达。

3. 频率学观点

   (a) 最大似然 (maximum likelihood) 估计

   其中 w 的值是使似然函数 $p(D|w)$ 达到最大值的 w 值。这对应于选择使观察到的数据集出现概率最大的 w 的值。在机器学习的文献中, 似然函数的负对数被叫做误差函数 (error function)。由于负对数是单调递减的函数, 最大化似然函数等价于最小化误差函数。

   (b) 自助法 (bootstrap)

   这种方法中, 多个数据集使用下面的方式创造。假设我们的原始数据集由 N 个数据点 $X = \{x_1, \cdots, x_N$ 组成。我们可以通过随机从 X 中抽取 N 个点的方式, 创造一个新的数据集 X_B 。抽取时可以有重复, 因此某些 X 中的数据点可能在 X_B 中有重复, 而其他的在 X 中的点会在 X_B 中缺失。这个过程可以重复 L 次, 生成 L 个数据集, 每个数据集的大小都是 N , 每个数据集是通过对原数数据集 X 采样得到的。参数估计的统计准确性之后就可以通过考察不同的自助数据集之间的预测的变化性来进行评估。

4. 贝叶斯优点

   贝叶斯观点的一个优点是对先验概率的包含是很自然的事情。例如, 假定投掷一枚普通的硬币 3 次, 每次都是正面朝上。一个经典的最大似然模型在估计硬币正面朝上的概率时, 结果会是 1, 表示所有未来的投掷都会是正面朝上! 相反, 一个带有任意的合理的先验的贝叶斯的方法将不会得出这么极端的结论。

5. 贝叶斯缺点

   针对贝叶斯方法的一种广泛的批评就是先验概率的选择通常是为了计算的方便而不是为了反映出任何先验的知识。某些人甚至把贝叶斯观点中结论对于先验选择的依赖性的本质看成困难的来源。减少对于先验的依赖性是所谓无信息 (noninformative) 先验的一个研究动机。然而, 这会导致比较不同模型时的困难, 并且实际上当先验选择不好的时

候, 贝叶斯方法有很大的可能性会给出错误的结果。频率学家估计方法在一定程度上避免了这一问题, 并且例如交叉验证的技术在模型比较等方面也很有用。

贝叶斯方法的实际应用在很长时间内都被执行完整的贝叶斯步骤的困难性所限制, 尤其是需要在整个参数空间求和或者求积分, 这在做预测或者比较不同的模型时必须进行。取样方法的发展, 例如马尔科夫链蒙特卡罗 (在第 11 章讨论), 以及计算机速度和存储容量的巨大提升, 打开了在相当多的问题中使用贝叶斯技术的大门。蒙特卡罗方法非常灵活, 可以应用于许多种类的模型。然而, 它们在计算上很复杂, 主要应用于小规模问题。

最近, 许多高效的判别式方法被提出来, 例如变种贝叶斯 (variational Bayes) 和期望传播 (expectation propagation)。这些提供了一种可选的补充的取样方法, 让贝叶斯方法能够应用于大规模的应用中 (Blei et al., 2003)。

1.3.2 高斯分布

1. 偏移 (bias)

   最大似然的偏移问题是我们在多项式曲线拟合问题中遇到的过拟合问题的核心。

2. 重新考察曲线拟合问题

   在高斯噪音下:

   (a) 通过最大似然方法, 求 w 和 β

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \tag{1.3}$$

   似然函数为:

$$p(\boldsymbol{t}|\boldsymbol{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}((t_n)|y(x_n, \mathbf{w}), \beta^{-1}) \tag{1.4}$$

似然函数的对数:

$$\ln p(\boldsymbol{t}|\boldsymbol{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1} N\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) \tag{1.5}$$

首先考虑确定多项式系数的最大似然解 (记作 w_{ML})。这些由公式 (1.5) 关于 w 来确定。为了达到这个目的, 我们可以省略公式 (1.5) 右侧的最后两项, 因为他们不依赖于 w 。并且, 我们注意到, 使用一个正的常数系数来缩放对数似然函数并不会改变关于 w 的最大值的位置, 因此我们可以用 1 来代替系数 β 。最后, 我们不去最大化似然函数, 而是等价地去最小化负对数。于是我们看到, 目前为止对于确定 w 的问题来说, 最大化似然函数等价于最小化由公式定义的平方和误差函数。因此, 在高斯噪声的假设下, 平方和误差函数是最大化似然函数的一个自然结果。

我们也可以使用最大似然方法来确定高斯条件分布的精度参数 β 。关于 β 来最大化函数 (1.5), 我们有

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{ML}) - t_n\}^2 \tag{1.6}$$

我们又一次首先确定控制均值的参数向量 w_{ML} , 然后使用这个结果来寻找精度 β_{ML} 。这与简单高斯分布时的情形相同。

(b) 最大后验 (maximum posterior), 简称 MAP

$$p(t|x, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(t|y(x, \mathbf{w}_{ML}), \beta_{ML}^{-1}) \tag{1.7}$$

现在让我们朝着贝叶斯的方法前进一步, 引入在多项式系数 w 上的先验分布。简单起见, 我们考虑下面形式的高斯分布

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2}\exp\left\{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\} \tag{1.8}$$

使用贝叶斯定理, w 的后验概率正比于先验分布和似然函数的乘积。

$$p(\mathbf{w}|\boldsymbol{x},\boldsymbol{t},\alpha,\beta) \propto p(\boldsymbol{t}|\boldsymbol{x},\mathbf{w},\beta)p(\mathbf{w}|\alpha) \tag{1.9}$$

取公式 (1.9) 的负对数, 结合公式 (1.5) 和公式 (1.8), 我们可以看到, 最大化后验概率就是最小化下式:

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w})-t_n\}^2+\frac{\alpha}{2}\mathbf{w}^T\mathbf{w} \tag{1.10}$$

因此我们看到最大化后验概率等价于最小化正则化的平方和误差函数 (之前在公式 (1.4) 中提到), 正则化参数为 $\lambda=\alpha/\beta$.

i. 超参数 (hyperparameters)
像 这样控制模型参数分布的参数, 被称为超参数 (hyperparameters)。

3. 贝叶斯曲线拟合 PRML$_{\text{Chinesevision.pdf}}$, p. 28

虽然我们已经谈到了先验分布 $p(w|\alpha)$, 但是我们目前仍然在进行 w 的点估计, 这并不是贝叶斯观点。在一个纯粹的贝叶斯方法中, 我们应该自始至终地应用概率的加和规则和乘积规则。我们稍后会看到, 这需要对所有 w 值进行积分。对于模式识别来说, 这种积分是贝叶斯方法的核心。

在曲线拟合问题中, 我们知道训练数据 $\boldsymbol{x}$ 和 $\boldsymbol{t}$, 以及一个新的测试点 x , 我们的目标是预测 t 的值。因此我们想估计预测分布 $p(t|x,\boldsymbol{x},\boldsymbol{t})$ 。这里我们要假设参数 α 和 β 是固定的, 事先知道的 (后续章节中我们会讨论这种参数如何通过贝叶斯方法从数据中推断出来)。

简单地说, 贝叶斯方法就是自始至终地使用概率的加和规则和乘积规则。因此预测概率可以写成下面的形式:

$$p(t|x,\boldsymbol{x},\boldsymbol{t}) = \int p(t|x,\mathbf{w})p(\mathbf{w}|\boldsymbol{x},\boldsymbol{t})\mathrm{d}\mathbf{w} \tag{1.11}$$

其中:

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}),$$

并且我们省略了对于 和 的依赖, 简化记号, 和归 1 化:

$$p(\mathbf{w}|\boldsymbol{x}, \boldsymbol{t}, \alpha, \beta) \propto p(\boldsymbol{t}|\boldsymbol{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

我们在 3.3 节将看到, 对于曲线拟合这样的问题, 后验分布是一个高斯分布, 可以解析地求出。类似地, 公式 (1.11) 中的积分也可以解析地求解。因此, 预测分布由高斯的形式给出:

$$p(t|x, \boldsymbol{x}, \boldsymbol{t}) = \mathcal{N}(t|m(x), s^2(x)) \tag{1.12}$$

where the mean and variance are given by

$$m(x) = \beta\boldsymbol{\phi}(x)^T\mathbf{S}\sum_{n=1}^{N}\boldsymbol{\phi}(x_n)t_n \tag{1.13}$$

$$s^2(x) = \beta^{-1} + \boldsymbol{\phi}(x)^T\mathbf{S}\boldsymbol{\phi}(x) \tag{1.14}$$

Here the matrix $\mathbf{S}$ is given by

$$\mathbf{S} = \alpha\mathbf{I} + \beta\sum_{n=1}^{N}\boldsymbol{\phi}(x_n)\boldsymbol{\phi}^T(x) \tag{1.15}$$

where $\mathbf{I}$ is the unit matrix, and we have defined the vector $\boldsymbol{\phi}(x)$ with elements $\phi_i(x) = x^i$ for $i = 0, \cdots, M$.

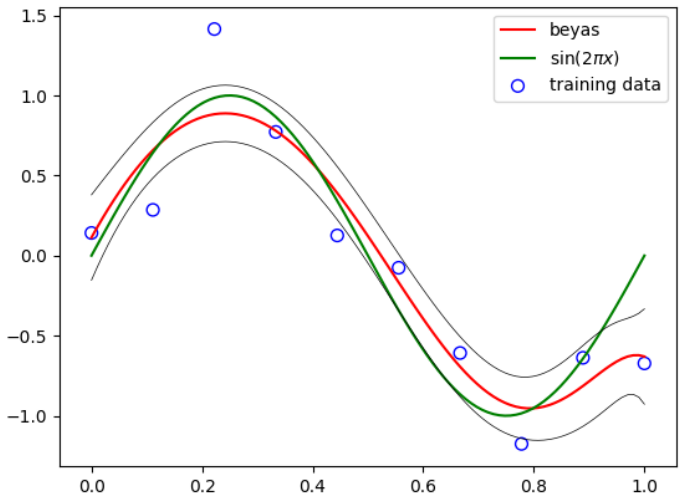
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import norm
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression, Ridge, BayesianRidge
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import mean_squared_error
import sys
sys.path.append("my-packages")
from bayesian_regressor import BayesianRegressor
from polynomial import PolynomialFeatures
pd.options.display.max_rows = 10
from tabulate import tabulate
tbl = lambda x: tabulate(x,headers="keys",tablefmt="orgtbl")
```

Listing 1.1: ch01-init

```
def create_toy_data(func, sample_size=10, std=1):
    x = np.linspace(0, 1, sample_size)
    t = func(x) + np.random.normal(scale=std, size=x.shape)
    return x, t


def func(x):
    return np.sin(2 * np.pi * x)


std = 0.3
np.random.seed(1234)
data_train = pd.DataFrame(
    dict(zip(["x", "t"], create_toy_data(func, std=std, sample_size=10))))
data_test = pd.DataFrame(
    dict(zip(["x", "t"], create_toy_data(func, std=std, sample_size=100))))
data_plot = pd.DataFrame({"x": np.linspace(0, 1, 100)})
```

Listing 1.2: generate data

```
plt.scatter(
    data_train["x"],
    data_train["t"],
    facecolor="none",
    edgecolor="b",
    s=50,
    label="training data")
beta_=11.1
alpha_=5e-3

def phi(x):
    phi_ = np.mat(np.vander([x], 10,increasing=True).T)
    return phi_
Phi_=np.mat(np.zeros((10,10)))
sum_phi_=np.mat(np.zeros((10,1)))
for sample,sample2 in zip(data_train["x"],data_train["t"]):
    Phi_ += phi(sample)*phi(sample).T
    sum_phi_ += phi(sample)*sample2
S=(alpha_*np.mat(np.eye(10))+beta_*Phi_).I
def m(x):
    m_x = (beta_*phi(x).T*S*sum_phi_).A1
    return m_x
def s(x):
    s_x = np.sqrt(phi(x).T * S * phi(x)).A1
    return s_x
x_test = data_plot["x"].values
y = data_plot["x"].apply(m).values
y_err = data_plot["x"].apply(s).values
fill_low = y-y_err
fill_up = y+y_err
plt.plot(data_plot["x"], data_plot["x"].apply(m), c="r", label="beyas")
plt.plot(data_plot["x"], data_plot.apply(func), c="g", label="$\sin(2\pi x)$")
plt.plot(x_test,y+y_err,c="k",linewidth=.5)
plt.plot(x_test,y-y_err,c="k",linewidth=.5)
plt.legend()
plt.savefig("test.png")
plt.close("all")
```



1.4 Model Selection

1.4.1 TODO 交叉验证 (cross validation)

将其划分为 S 组 (最简单的情况下, 等于数据的个数)。然后, $S-1$ 组数据被用于训练一组模型, 然后在剩余的一组上进行评估。然后对于所有 S 的可能选择重复进行这一步骤, 使用剩余的一组进行评估, 之后, 对 S 轮运行结果的表现得分求平均值。

1.5 The Curse of Dimensionality

1.6 Decision Theory

1.6.1 回归问题的损失函数

PRML$_{\text{Chinesevision.pdf}}$, p. 38

1. loss function

   决策阶段包括对于每个输入 $\mathbf{x}$, 选择一个对于 t 值的具体的估计 $y(\mathbf{x})$ 。假设这样做之后, 我们造成了一个损失 $L(t, y(\mathbf{x}))$ 。平均损失 (或者

说期望损失) 就是

$$\mathbb{E}[L] = \int\int L(t, y(\mathbf{x}))p(\mathbf{x}, t)\mathrm{d}\mathbf{x}\mathrm{d}\mathrm{t} \tag{1.16}$$

2. loss-function-for-regression

   回归问题中, 损失函数的一个通常的选择是平方损失, 定义为 $L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$ 。这种情况下, 期望损失函数可以写成

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\mathrm{d}\mathbf{x}\mathrm{d}\mathrm{t} \tag{1.17}$$

3. 回归函数 (regression function)

   我们的目标是选择 $y(\mathbf{x})$ 来最小化 $\mathbb{E}[L]$ 。如果我们假设一个完全任意的函数 $y(\mathbf{x})$, 我们能够形式化地使用变分法求解:

$$\frac{\delta\mathbb{E}[L]}{\delta y(\mathbf{x})} = 2\int \{y(\mathbf{x} - t)\}p(\mathbf{x}, t)\mathrm{d}\mathrm{t} = 0 \tag{1.18}$$

   求解 $y(\mathbf{x})$, 使用概率的加和规则和乘积规则, 我们得到

$$y(\mathbf{x}) = \frac{\int tp(\mathbf{x}, t)\mathrm{d}\mathrm{t}}{p(\mathbf{x})} = \int tp(t|\mathbf{x}) \tag{1.19}$$

   这是在 x 的条件下 t 的条件均值, 被称为回归函数 (regression function)。结果如图 1.1 所示。这个结果可以扩展到多个目标变量 (用向量 $\mathbf{t}$) 的情形。这种情况下, 最优解是条件均值 $y(\mathbf{x}) = \mathbb{E}_t[\mathbf{t}|\mathbf{x}]$ 。

4. 另一种方式推导出这个结果

   已经知道了最优解是条件期望, 我们可以把平方项按照下面的方式展开:

$$\begin{aligned}\{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x} - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t)\}^2 \\ &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} \\ &\quad + \{\mathbb{E}[t|\mathbf{x}] - t\}^2\end{aligned}$$

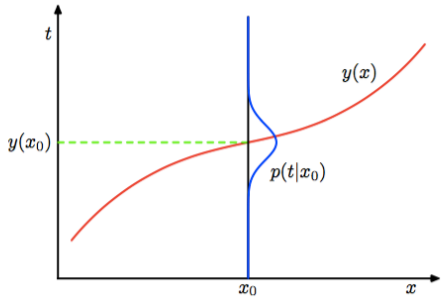


图 1.1: 最小化了期望平方损失的回归函数 $y(x)$ 由条件概率分布 $p(t|x)$ 的均值给出。

其中, 为了不让符号过于复杂, 我们使用 $\mathbb{E}[t|\mathbf{x}]$ 来表示 $\mathbb{E}_t[t|\mathbf{x}]$ 。代入损失函数中, 对 t 进行积分, 我们看到交叉项消失, 因而得到下面形式的损失函数

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int \mathrm{var[t|x]p(x)}\,\mathrm{d}\mathbf{x} \qquad (1.20)$$

注意 $\mathbb{E}[x]$ 与 x 无关

我们寻找的函数 $y(\mathbf{x})$ 只出现在第一项中。当 $y(\mathbf{x})$ 等于 $\mathbb{E}[t|\mathbf{x}]$ 时第一项取得最小值, 这时第一项会被消去。这正是我们之前推导的结果, 表明最优的最小平方预测由条件均值给出。第二项是 t 的分布的方差, 在 x 上进行了平均。它表示目标数据内在的变化性, 可以被看成噪声。由于它与 $y(\mathbf{x})$ 无关, 因此它表示损失函数的不可减小的最小值。

与分类问题相同, 我们可以确定合适的概率然后使用这些概率做出最优的决策, 或者我们可以建立直接决策的模型。实际上, 我们可以区分出三种解决回归问题的方法, 按照复杂度降低的顺序, 依次为:

(a) 首先解决确定联合概率密度 $p(\mathbf{x}, t)$ 的推断问题。

之后, 计算条件概率密度 $p(t|\mathbf{x})$ 。最后, 使用公式 (1.89) 积分, 求出条件均值。

(a) 首先解决确定条件概率密度 p(t | x) 的推断问题。之后使用公式 (1.89) 计算条件均值。(c) 直接从训练数据中寻找一个回归函数 y(x)。这三种方法的相对优势和之前所述的分类问题的情形很相似。平方损失函数不是回归问题中损失函数的唯一选择。实际上, 有些情况下, 平方损失函数会导致非常差的结果, 这时我们就需要更复杂的方法。这种情况的一个重要的例子就是条件分布 p(t | x) 有多个峰值, 这在解决反演问题时经常出现。这里我们简要介绍一下平方损失函数的一种推广, 叫做闵可夫斯基损失函数 (Minkowski loss), 它的期望为

1.7 Notes

1.7.1 Curve fitting 为例子演示三种方法

1. **MLE**, 直接对 likelihood function 求最大值, 得到参数 w 。该方法属于 point estimation。

2. **MAP** (poor man's bayes), 引入 prior probability, 对 posterior probability 求最大值, 得到 w。MAP 此时相当于在 MLE 的目标函数 (likelihood function) 中加入一个 L2 penalty。该方

法仍属于 point estimation。

2 Probability Distributions

3 Linear Models for Regression

3.1 Abstract

(Information Science and Statistics) Christopher M. Bishop-Pattern Recognition and Machine Learning-Springer (2007).pdf, p. 156

目前为止, 本书的关注点是无监督学习, 包括诸如 **概率密度估计和数据聚类**等话题。我们现在开始讨论有监督学习, 首先讨论的是回归问题。回归问题的目标是在给定 D 维输入 (input) 变量 x 的情况下, 预测一个或者多

个连续目标 (target) 变量 t 的值。在第 1 章中, 我们已经遇到了回归问题的一个例子: 多项式曲线拟合问题。多项式是被称为线性回归模型的一大类函数的一个具体的例子。线性回归模型有着可调节的参数, 具有线性函数的性质, 将会成为本章的关注点。线性回归模型的最简单的形式也是输入变量的线性函数。但是, 通过将一组输入变量的非线性函数进行线性组合, 我们可以获得一类更加有用的函数, 被称为基函数 (basis function)。这样的模型是参数的线性函数, 这使得其具有一些简单的分析性质, 同时关于输入变量是非线性的。

给定一个由 N 个观测值 $\{x_n\}$ 组成的数据集, 其中 $n = 1, \cdots, N$, 以及对应的目标值 $\{t_n\}$, 我们的目标是预测对于给定新的 x 值的情况下, t 的值。最简单的方法是, 直接建立一个适当的函数 $y(x)$, 对于新的输入 x , 这个函数能够直接给出对应的 t 的预测。更一般地, 从一个概率的观点来看, 我们的目标是对预测分布 $p(t|x)$ 建模, 因为它表达了对于每个 x 值, 我们对于 t 的值的不确定性。从这个条件概率分布中, 对于任意的 x 的新值, 我们可以对 t 进行预测, 这种方法等同于最小化一个恰当选择的损失函数的期望值。正如在 1.5.5 节讨论的那样, 对于实值变量来说, 损失函数的一个通常的选择是平方误差损失, 这种情况下最优解由 t 的条件期望给出。

虽然线性模型对于模式识别的实际应用来说有很大的局限性, 特别是对于涉及到高维输入空间的问题来说更是如此, 但是他们有很好的分析性质, 并且组成了后续章节中将要讨论的更加复杂的模型的基础。

3.2 线性基函数模型

1. 线性回归 (linear regression)

$$y(x, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D \tag{3.1}$$

where $\mathbf{x} = (x_1, \cdots, x_D)^T$.

这通常被简单地称为线性回归 (linear regression)。

这个模型的关键性质是它是参数 $w_0, \cdots, w_D$ 的一个线性函数。但是, 它也是输入变量 x_i 的一个线性函数, 这给模型带来的极大的局限性。

3.2.1 main body

因此我们这样扩展模型的类别: 将输入变量的固定的非线性函数进行线性组合, 形式为

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \tag{3.2}$$

where $\phi_j(x)$ are known as basis functions. 通过把下标 j 的最大值记作 $M-1$, 这个模型中的参数总数为 M 。

通常, 定义一个额外的虚"基函数"$\phi_0(x) = 1$ 是很方便的, 这时

$$y(\mathbf{x}, w) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \tag{3.3}$$

where $\mathbf{w} = (w_0, \cdots, w_{M-1})^T$ and $\boldsymbol{\phi} = (\phi_0, \cdots, \phi_{M-1})^T$.

在许多模式识别的实际应用中, 我们会对原始的数据变量进行某种固定形式的预处理或者特征抽取。如果原始变量由向量 $\mathbf{x}$ 组成, 那么特征可以用基函数 $\{\phi_j(\mathbf{x})\}$ 来表示。

通过使用非线性基函数, 我们能够让函数 $y(\mathbf{x}, w)$ 成为输入向量 $\mathbf{x}$ 的一个非线性函数。但是, 形如 (3.2) 的函数被称为线性模型, 因为这个函数是 w 的线性函数。正是这种关于参数的线性极大地简化了对于这列模型的分析。然而, 这也造成了一些巨大的局限性, 正如我们在 3.6 节讨论的那样。

1. 偏置参数 (bias parameter)

   参数 w_0 使得数据中可以存在任意固定的偏置, 这个值通常被称为偏置参数 (bias parameter)。注意不要把这里的"偏置"与统计学中的"偏置"弄混淆。

2. 基函数例子

   (a) 幂基函数

   第 1 章中讨论的多项式拟合的例子是这个模型的一个特例, 那里有一个输入变量 x, 基函数是 x 的幂指数的形式, 即 $\phi_j(x) = x^j$ 。

多项式基函数的一个局限性是它们是输入变量的全局函数, 因此对于输入空间一个区域的改变将会影响所有其他的区域。

i. 样条函数 (spline function)
这个问题可以这样解决: 把输入空间切分成若干个区域, 然后对于每个区域用不同的多项式函数拟合。这样的函数叫做样条函数 (spline function)(Hastie et al., 2001)。

(b) 高斯基函数 (Information Science and Statistics) Christopher M. Bishop-Pattern Recognition and Machine Learning-Springer (2007).pdf, p. 158

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\} \tag{3.4}$$

其中 μ_j 控制了基函数在输入空间中的位置, 参数 s 控制了基函数的空间大小。这种基函数通常被称为"高斯"基函数, 但是应该注意它们未必一定是一个概率表达式。特别地, 归一化系数不重要, 因为这些基函数会与一个调节参数 w_j 相乘。

(c) sigmoid 基函数 (Information Science and Statistics) Christopher M. Bishop-Pattern Recognition and Machine Learning-Springer (2007).pdf, p. 158

另一种选择是 sigmoid 基函数, 形式为

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right) \tag{3.5}$$

其中 $\sigma(a)$ 是 logistic sigmoid 函数, 定义为

$$\sigma(a) = \frac{1}{1+\exp(-a)}. \tag{3.6}$$

等价地, 我们可以使用 tanh 函数, 因为它和 logistic sigmoid 函数的关系为 $\tanh(a) = 2\sigma(2a) - 1$, 因此 logistic sigmoid 函数的一般的线性组合等价于 tanh 函数的一般的线性组合。图 3.1 说明了基函数的不同选择情况。

(d) 傅里叶基函数

它可以用正弦函数展开。每个基函数表示一个具体的频率, 它在空间中有无限的延伸。相反, 限制在输入空间中的有限区域的基函数要由不同空间频率的一系列频谱组成。在许多信号处理的应用中, 一个吸引了研究者兴趣的问题是考虑同时在空间和频率受限的基函数。这种研究产生了一类被称为小波 (wavelet) 的函数。为了简化应用, 这些基函数被定义为相互正交的。当应用中的输入值位于正规的晶格中时, 应用小波最合适。这种应用包括时间序列中的连续的时间点, 以及图像中的像素。关于小波的有用的教科书包括 Ogden(1997),Mallat (1999) 和 Vidakovic(1999)。

3.2.2 最大似然与最小平方

(Information Science and Statistics) Christopher M. Bishop-Pattern Recognition and Machine Learning-Springer (2007).pdf, p. 159

与之前一样, 我们假设目标变量 t 由确定的函数 $y(\mathbf{x}, w)$ 给出, 这个函数被附加了高斯噪声, 即

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \tag{3.7}$$

其中 ϵ 是一个零均值的高斯随机变量, 精度 (方差的倒数) 为 β 。因此我们有

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}). \tag{3.8}$$

回忆一下, 如果我们假设一个平方损失函数, 那么对于 x 的一个新值, 最优的预测由目标变量的条件均值给出。在公式 (3.8) 给出的高斯条件分布的情况下, 条件均值可以简单地写成

$$\mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x})\, dx = y(\mathbf{x}, \mathbf{w}) \tag{3.9}$$

注意高斯噪声的假设表明, 给定 x 的条件下, t 的条件分布是单峰的, 这对于一些实际应用来说是不合适的。第 14.5.1 节将扩展到条件高斯分布的混合, 那种情况下可以描述多峰的条件分布。

PRML$_{\text{Chinesevision.pdf}}$, p. 103

现在考虑一个输入数据集 $\mathbf{X} = \{x_1, \cdots, x_N\}$, 对应的目标值为 $t_1, \cdots, t_N$ 。我们把目标向量 $\{t_n\}$ 组成一个列向量, 记作 t 。这个变量的字体与多元目标值的一次观测 (记作 t) 不同。假设这些数据点是独立地从分布 (3.8) 中抽取的, 那么我们可以得到下面的似然函数的表达式, 它是可调节参数 w 和 β 的函数, 形式为

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \tag{3.10}$$

其中我们使用了公式 (3.3)。**注意, 在有监督学习问题中 (例如回归问题和分类问题), 我们不是在寻找模型来对输入变量的概率分布建模**。 因此 $\mathbf{x}$ 总会出现在条件变量的位置上。因此从现在开始, 为了保持记号的简洁性, 我们在诸如 $p(\boldsymbol{t}|\mathbf{x}, w, \beta)$ 这类的表达式中不显式地写出 x 。取对数似然函数的对数, 使用一元高斯分布的标准形式 (2.146), 我们有

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned} \tag{3.11}$$

where the sum-of-squares error function is defined by

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\boldsymbol{\Phi}(\mathbf{x}_n)\}^2. \tag{3.12}$$

写出了似然函数, 我们可以使用最大似然的方法确定 w 和 β 。首先关于 w 求最大值。正如我们已经在 1.2.5 节中已经看到的那样, 我们看到在条件高斯噪声分布的情况下, 线性模型的似然函数的最大化等价于平方和误差函数的最小化。平方和误差函数由 $E_D(w)$ 给出。公式 (3.11) 给出的对数似然函数的梯度为

$$\nabla \ln p(\boldsymbol{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N}\{t_n - \mathbf{w}^T\boldsymbol{\phi}(x_n)\}\boldsymbol{\phi}(x_n)^T. \tag{3.13}$$

Setting this gradient to zero gives

$$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\mathbf{x}_n)^T - \mathbf{w}^T \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^T \right). \tag{3.14}$$

Solving for $\mathbf{w}$ we obtain

$$\mathbf{w}_{ML} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{t} \tag{3.15}$$