ML-05

201300086史浩男

第三题附代码Iris-Bayes.py

一、贝叶斯决策论

教材 7.1 节介绍了贝叶斯决策论, 它是一种解决统计决策问题的通用准则. 考虑一个带有 “拒绝” 选项的 N 分类问题, 给定一个样例, 分类器可以选择预测这个样例的标记, 也可以选择拒绝判断并将样例交给人类专家处理. 设类别标记的集合为 $\mathcal{Y}=\{c_1,c_2,\ldots,c_N\}$, λ_{ij} 是将一个真实标记为 c_i 的样例误分类为 c_j 所产生的损失, 而人类专家处理一个样例需要额外 λ_h 费用. 假设后验概率 $P(c\mid\boldsymbol{x})$ 已知, 且 $\lambda_{ij}\geq 0$, $\lambda_h\geq 0$. 请思考下列问题:

1. 基于期望风险最小化原则, 写出此时贝叶斯最优分类器 $h^\star(\boldsymbol{x})$ 的表达式;

(1*)最优分类器

$$h^*(\boldsymbol{x})=\arg\min_{c_i\in\mathcal{Y}}\sum_{j=1}^{N}\lambda_{ij}P\left(c_j\mid\boldsymbol{x}\right)$$

2. 人类专家的判断成本 λ_h 取何值时, 分类器 $h^\star$ 将一直拒绝分类? 当 λ_h 取何值时, 分类器 $h^\star$ 不会拒绝分类任何样例?

(2)代价bound

(i)一直拒绝分类:

$$\lambda_h\leq\min_i\sum_{j=1}^{N}\lambda_{ij}P\left(c_j\mid\boldsymbol{x}\right)$$

(ii)不会拒绝分类:

$$\lambda_h\geq\max_i\sum_{j=1}^{N}\lambda_{ij}P\left(c_j\mid\boldsymbol{x}\right)$$

3. 考虑一个具体的二分类问题, 其损失矩阵为

$$\boldsymbol{\Lambda}=\begin{pmatrix}\lambda_{11}&\lambda_{12}\\\lambda_{21}&\lambda_{22}\end{pmatrix}=\begin{pmatrix}0&1\\1&0\end{pmatrix},\tag{1}$$

且人类专家处理一个样例的代价为 $\lambda_h=0.3$. 对于一个样例 $\boldsymbol{x}$, 设 $p_1=P(c_1\mid\boldsymbol{x})$, 证明存在 $\theta_1,\theta_2\in[0,1]$, 使得贝叶斯最优决策恰好为: 当 $p_1<\theta_1$ 时, 预测为第二类, 当 $\theta_1\leq p_1\leq\theta_2$ 时, 拒绝预测, 当 $\theta_2<p_1$ 时, 预测为第一类.

(3)

验证当$\theta_1=0.3,\theta_2=0.7$时满足要求:

$$R(c_1|x)=\lambda_{11}P(c_1|x)+\lambda_{12}P(c_2|x)=P(c_2|x)=1-p_1$$
$$R(c_2|x)=\lambda_{21}P(c_1|x)+\lambda_{22}P(c_2|x)=P(c_1|x)=p_1$$

- 当$p_1<\theta_1$时，$R(c_2|x)=p_1<0.3<1-p_1=R(c_1|x)$，预测为第二类
- 当$\theta_1\leq p_1\leq\theta_2$时，$0.3<R(c_2|x),0.3<R(c_1|x)$，拒绝
- 当$\theta_2<p_1$时，$R(c_1|x)=1-p_1<0.3<p_1=R(c_2|x)$，预测为第一类

二、极大似然估计

教材 7.2 节介绍了极大似然估计方法用于确定概率模型的参数. 其基本思想为: 概率模型的参数应当使得当前观测到的样本是最有可能被观测到的, 即当前数据的似然最大. 本题通过抛硬币的例子理解极大似然估计的核心思想.

1. 现有一枚硬币, 抛掷这枚硬币后它可能正面向上也可能反面向上. 我们已经独立重复地抛掷了这枚硬币 99 次, 均为正面向上. 现在, 请使用极大似然估计来求解第 100 次抛掷这枚硬币时其正面向上的概率;

(1)极大似然估计

设正面朝上概率为θ_c

训练集中99个样本的类别都是"正面向上"，所以θ_c对于D_c的对数似然为

$$LL\left(\boldsymbol{\theta}_c\right)=\log P\left(D_c \mid \boldsymbol{\theta}_c\right)=\sum_{\boldsymbol{x} \in D_c} \log P\left(\boldsymbol{x} \mid \boldsymbol{\theta}_c\right)=99 \log \boldsymbol{\theta}_c$$

极大似然估计为

$$\hat{\boldsymbol{\theta}}_c=\arg \max_{\boldsymbol{\theta}_c} LL\left(\boldsymbol{\theta}_c\right)$$

解为：$P\left(1 \mid \boldsymbol{\theta}_c\right)=1$，即正面向上的概率为1

因此预测第100次正面向上概率为1

2. 仍然考虑上一问的问题. 但现在, 有一位抛硬币的专家仔细观察了这枚硬币, 发现该硬币质地十分均匀, 并猜测这枚硬币 "肯定有 50% 的概率正面向上". 如果同时考虑已经观测到的数据和专家的见解, 第 100 次抛掷这枚硬币时, 其正面向上的概率为多少?

(2*)最大后验估计

$$\begin{aligned} P(\boldsymbol{\theta}_c \mid X) &= \frac{P(X \mid \boldsymbol{\theta}_c) P(\boldsymbol{\theta}_c)}{P(X)} \\ &\propto P(X \mid \boldsymbol{\theta}_c) P(\boldsymbol{\theta}_c) \end{aligned}$$

最大后验估计为

$$\hat{\boldsymbol{\theta}}_c \quad = \quad \arg \max_{\boldsymbol{\theta}_c} \sum_{\boldsymbol{x} \in D_c} \log P\left(\boldsymbol{x} \mid \boldsymbol{\theta}_c\right)+logP(\boldsymbol{\theta}_c)$$

专家的先验P（x）=0.5不连续不可求导，因此也无法准确地计算出$\hat{\boldsymbol{\theta}}_c$的值

因此预测第100次正面向上概率为0.5~1.0，无法得到具体值

3. 若同时考虑专家先验和实验数据来对硬币正面朝上的概率做估计. 设这枚硬币正面朝上的概率为 θ, 某抛硬币专家主观认为 $\theta \sim \mathcal{N}(\frac{1}{2}, \frac{1}{900})$, 即 θ 服从均值为 $\frac{1}{2}$, 方差为 $\frac{1}{900}$ 的高斯分布. 另一方面, 我们独立重复地抛掷了这枚硬币 400 次, 记第 i 次的结果为 x_i, 若 $x_i=1$ 则表示硬币正面朝上, 若 $x_i=0$ 则表示硬币反面朝上. 经统计, 其中有 100 次正面向上, 有 300 次反面向上. 现在, 基于专家先验和观测到的数据 $\boldsymbol{x}=\{x_1, x_2, \ldots, x_{400}\}$, 对参数 θ 分别做极大似然估计和最大后验估计;

(3)

极大似然:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg \max_{\boldsymbol{\theta}} LL\left(\boldsymbol{\theta}\right) \\ &= \arg \max_{\boldsymbol{\theta}} log(\boldsymbol{\theta}^{100}(1-\boldsymbol{\theta})^{300}) \\ &= \arg \max_{\boldsymbol{\theta}} 100log\boldsymbol{\theta}+300log(1-\boldsymbol{\theta}) \end{aligned}$$

解得$\hat{\boldsymbol{\theta}} = 0.25$

最大后验：

θ的密度函数：

$$f(\theta) = \frac{1}{\sqrt{\frac{\pi}{450}}} e^{-\frac{(\theta-0.5)^2}{\frac{1}{450}}}$$

因此最大后验估计：

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} log(\boldsymbol{\theta}^{100}(1-\boldsymbol{\theta})^{300}) + log(\frac{1}{\sqrt{\frac{\pi}{450}}} e^{-\frac{(\theta-0.5)^2}{\frac{1}{450}}}) \\ &= \arg\max_{\boldsymbol{\theta}} 100log\boldsymbol{\theta} + 300log(1-\boldsymbol{\theta}) - 450(\theta-0.5)^2 \end{aligned}$$

求极值，解方程

$$\frac{d\ln\left[P\left(x_0, x_1, \ldots, x_n \mid \theta\right) \times P(\theta)\right]}{d\theta} = 0$$

$$\frac{100}{\boldsymbol{\theta}} - \frac{300}{1-\boldsymbol{\theta}} - 450(2\boldsymbol{\theta} - 1) = 0$$

$$18\boldsymbol{\theta}^3 - 27\boldsymbol{\theta}^2 + \boldsymbol{\theta} + 2 = 0$$

```
x= [1/3, 7/12 - sqrt(97)/12, 7/12 + sqrt(97)/12]
```

其中符合$[0, 1]$范围内的解只有$\hat{\boldsymbol{\theta}} = \frac{1}{3}$

4. 如何理解上一小问中极大似然估计的结果和最大后验估计的结果？

(4)

- 极大似然估计MLE是频率主义学派的思想，认为参数 θ 是一个客观存在的固定值，可以通过优化似然函数等准则来确定参数值，只需要看样本的分布，不考虑先验概率，考虑的是可以使得当前抽样结果发生可能性最大的解θ，因此直观计算正例占总体样本比例也能算出概率估计值
- 最大后验估计MAP是贝叶斯学派对的估计，必须考虑先验分布的影响，需要引入概率密度函数，求导求最值。由于先验分布的均值表示了概率是0.5，而最大似然求出的概率是0.25，因此最终结果1\3是二者中和的结果

三、朴素贝叶斯分类器

朴素贝叶斯算法有很多实际应用，本题以 sklearn 中的 Iris 数据集为例，探讨实践中朴素贝叶斯算法的技术细节. 可以通过 sklearn 中的内置函数直接获取 Iris 数据集，代码如下：

```python
def load_data():
    # 以feature, label的形式返回数据集
    feature, label = datasets.load_iris(return_X_y=True)
    print(feature.shape) # (150, 4)
    print(label.shape) # (150,)
    return feature, label
```

上述代码返回 Iris 数据集的特征和标记, 其中 feature 变量是形状为 $(150, 4)$ 的 numpy 数组, 包含了 150 个样本的 4 维特征, 而 label 变量是形状为 (150) 的 numpy 数组, 包含了 150 个样本的类别标记. Iris 数据集中一共包含 3 类样本, 所以类别标记的取值集合为 $\{0, 1, 2\}$. Iris 数据集是类别平衡的, 每类均包含 50 个样本. 我们进一步将完整的数据集划分为训练集和测试集, 其中训练集样本量占总样本量的 80%, 即 120 个样本, 剩余 30 个样本作为测试样本.

```python
feature_train, feature_test, label_train, label_test = \
    train_test_split(feature, label, test_size=0.2, random_state=0)
```

朴素贝叶斯分类器会将一个样例的标记预测为类别后验概率最大的那一类对应的标记, 即:

$$\hat{y} = \arg\max_{y \in \{0,1,2\}} P(y) \prod_{i=1}^{d} P(x_i \mid y) \,. \tag{2}$$

因此, 为了构建一个朴素贝叶斯分类器, 我们需要在训练集上获取所有类别的先验概率 $P(y)$ 以及所有类别所有属性上的类条件概率 $P(x_i \mid y)$.

1. 请检查训练集上的类别分布情况, 并基于多项分布假设对 $P(y)$ 做极大似然估计;

使用`Counter()`函数统计类别分布情况:

```
Counter(label_train): Counter({2: 44, 0: 39, 1: 37})
Counter(label_test): Counter({1: 13, 0: 11, 2: 6})
```

极大似然估计为:

$$\hat{\boldsymbol{\theta}}_c = \arg\max_{\boldsymbol{\theta}_c} \sum_{\boldsymbol{x} \in D_c} \log P\left(\boldsymbol{x} \mid \boldsymbol{\theta}_c\right)$$

$$P(y=0) = \frac{39}{120} = 0.325$$

$$P(y=1) = \frac{37}{120} = 0.308$$

$$P(y=2) = \frac{44}{120} = 0.367$$

2. 在 Iris 数据集中, 每个样例 $\boldsymbol{x}$ 都包含 4 维实数特征, 分别记作 x_1, x_2, x_3 和 x_4. 为了计算类条件概率 $P(x_i \mid y)$, 首先需要对 $P(x_i \mid y)$ 的概率形式做出假设. 在本小问中, 我们假设每一维特征在给定类别标记时是独立的 (朴素贝叶斯的基本假设), 并假设它们服从高斯分布. 试基于 sklearn 中的 GaussianNB 类构建分类器, 并在测试集上测试性能;

```python
clf = GaussianNB()
clf=clf.fit(feature_train, label_train)
print("==Predict result in GaussianNB by predict==")
print(clf.predict(feature_test))
print(clf.score(feature_test, label_test))
```

使用`fit`方法训练, 使用`predict`方法直接给出测试集的预测类别输出

```
==Predict result in GaussianNB by predict==
[2 1 0 2 0 2 0 1 1 1 1 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
0.9666666666666667
```

与测试集标记`label_test`对比，发现只有一项不同，准确率`score`为96.7%

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
```

3. 在 GaussianNB 类中手动指定类别先验为三个类上的均匀分布，再次测试模型性能；

指定类别先验，三类上的均匀分布即每个类都是$\frac{1}{3}$

```
clf.class_prior_ = [1./3.,1./3.,1./3.]
```

性能结果不变

```
==Predict result in GaussianNB by predict==
[2 1 0 2 0 2 0 1 1 1 1 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
0.9666666666666667
```

但如果把`predict`方法改为`predict_proba`，即观察更细节的对每个类别估计，可以发现指定类别先验后确实改变了预测结果，只不`predict`方法预测结果过不明显而已：（左图为添加类别先验前，右图为添加后）

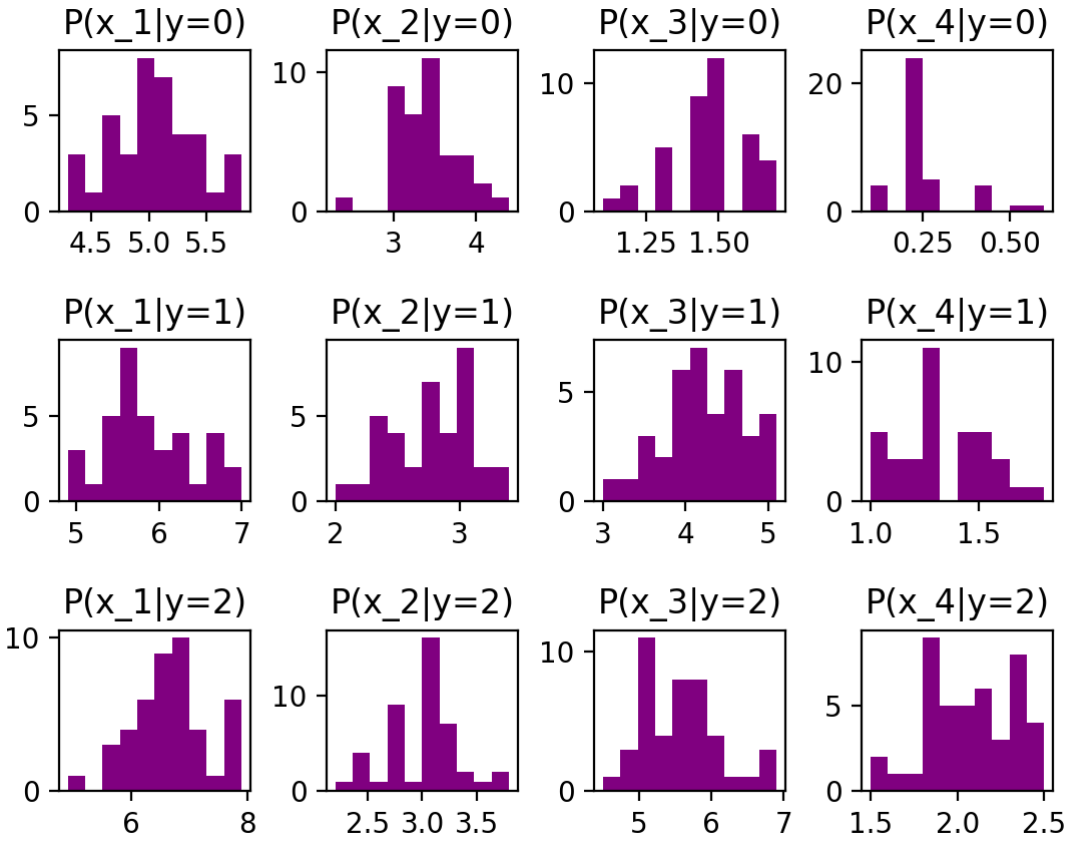
```
[[1.63380783e-232 2.18878438e-006 9.99997811e-001]
 [1.82640391e-082 9.99998304e-001 1.69618390e-006]
 [1.00000000e+000 7.10250510e-019 3.65449801e-028]
 [1.58508262e-305 1.04649020e-006 9.99998954e-001]
 [1.00000000e+000 8.59168655e-017 4.22159374e-027]
 [6.39815011e-321 1.56450314e-010 1.00000000e+000]
 [1.00000000e+000 1.09797313e-016 5.30276557e-027]
 [1.25122812e-146 7.74052109e-001 2.25947891e-001]
 [5.34357526e-150 9.07564955e-001 9.24350453e-002]
 [5.67261712e-093 9.99882109e-001 1.17891111e-004]
 [2.38651144e-210 5.29609631e-001 4.70390369e-001]
 [8.12047631e-132 9.43762575e-001 5.62374248e-002]
 [5.25177109e-132 9.98864361e-001 1.13563851e-003]
 [1.24498038e-139 9.49838641e-001 5.01613586e-002]
 [4.08232760e-140 9.88043864e-001 1.19561365e-002]
 [1.00000000e+000 7.12837229e-019 4.10162749e-029]
 [4.19553996e-131 9.87944980e-001 1.20550201e-002]
 [4.13286716e-111 9.99942383e-001 5.76167389e-005]
 [1.00000000e+000 2.24933112e-015 3.63624519e-026]
 [1.00000000e+000 9.86750131e-016 2.42355087e-025]
 [1.85930865e-186 1.66966805e-002 9.83303319e-001]
 [8.83060167e-130 9.92757232e-001 7.24276827e-003]
 [1.00000000e+000 4.26380344e-013 4.34222344e-023]
 [1.00000000e+000 1.28045851e-016 1.26708019e-027]
 [2.43739221e-168 1.83516225e-001 8.16483775e-001]
 [1.00000000e+000 2.62431469e-018 6.72573168e-029]
 [1.00000000e+000 3.20605389e-011 1.52433420e-020]
 [2.20964201e-110 9.99291229e-001 7.08771072e-004]
 [1.39297338e-046 9.99999972e-001 2.81392389e-008]
 [1.00000000e+000 1.85943966e-013 1.58833385e-023]]
0.9666666666666667
```

```
[[1.84326961e-232 2.60287765e-006 9.99997397e-001]
 [1.73274264e-082 9.99998574e-001 1.42633685e-006]
 [1.00000000e+000 7.48642430e-019 3.23921414e-028]
 [1.78829798e-305 1.24447458e-006 9.99998756e-001]
 [1.00000000e+000 9.05610204e-017 3.74186718e-027]
 [7.21829909e-321 1.86049022e-010 1.00000000e+000]
 [1.00000000e+000 1.15732303e-016 4.70017857e-027]
 [1.23132406e-146 8.02913855e-001 1.97086145e-001]
 [5.14520902e-150 9.21110413e-001 7.88895867e-002]
 [5.38181461e-093 9.99900862e-001 9.91375659e-005]
 [2.44726706e-210 5.72448736e-001 4.27551264e-001]
 [7.77359088e-132 9.52282516e-001 4.77174835e-002]
 [4.98334984e-132 9.99044859e-001 9.55141309e-004]
 [1.19063678e-139 9.57479535e-001 4.25204649e-002]
 [3.88035835e-140 9.89926816e-001 1.00731842e-002]
 [1.00000000e+000 7.51368971e-019 3.63553346e-029]
 [3.98803249e-131 9.89843345e-001 1.01566549e-002]
 [3.92096120e-111 9.99951549e-001 4.84508837e-005]
 [1.00000000e+000 2.37091659e-015 3.22303551e-026]
 [1.00000000e+000 1.04008798e-015 2.14814736e-025]
 [2.09107619e-186 1.97929892e-002 9.80207011e-001]
 [8.38741477e-130 9.93902464e-001 6.09753562e-003]
 [1.00000000e+000 4.49427930e-013 3.84878896e-023]
 [1.00000000e+000 1.34967248e-016 1.12309380e-027]
 [2.65760813e-168 2.10912770e-001 7.89087230e-001]
 [1.00000000e+000 2.76616953e-018 5.96144399e-029]
 [1.00000000e+000 3.37935410e-011 1.35111440e-020]
 [2.09656345e-110 9.99403921e-001 5.96079251e-004]
 [1.32153885e-046 9.99999976e-001 2.36625419e-008]
 [1.00000000e+000 1.95994991e-013 1.40784137e-023]]
0.9666666666666667
```

4. 在朴素贝叶斯模型中，对类条件概率的形式做出正确的假设也很重要. 请检查每个类别下特征的数值分布，并讨论该如何选定类条件概率的形式.

对于训练集中每个类别下每个特征数值分布画出频数直方图

其中$P(x_i|y=j)$表示j类别中特征x_i的分布



我们发现每个特征的都大致符合高斯分布的特点，即中间高两边低，因此条件概率的形式选择为高斯分布可以增大模型性能

四、集成--Boosting

Boosting 算法有序地训练一批弱学习器进行集成得到一个强学习器, 核心思想是使用当前学习器 “提升” 已训练弱学习器的能力. 教材 8.2 节介绍的 AdaBoost 是一种典型的 Boosting 算法, 通过调整数据分布使新学习器重点关注之前学习器分类错误的样本. 教材介绍的 AdaBoost 关注的是二分类问题, 即样本 $\boldsymbol{x}$ 对应的标记 $y(\boldsymbol{x}) \in \{-1, +1\}$. 记第 t 个基学习器及其权重为 h_t 和 α_t, 采用 T 个基学习器加权得到的集成学习器为 $H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$. AdaBoost 最小化指数损失: $\ell_{\exp} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-y(\boldsymbol{x})H(\boldsymbol{x})}\right]$.

1. 在 AdaBoost 训练过程中, 记前 t 个弱学习器的集成为 $H_t(\boldsymbol{x}) = \sum_{i=1}^{t} \alpha_i h_i(\boldsymbol{x})$, 该阶段优化目标为:

$$\ell_{\exp,t} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-y(\boldsymbol{x})H_t(\boldsymbol{x})}\right]. \tag{3}$$

   如果记训练数据集的初始分布为 $\mathcal{D}_0 = \mathcal{D}$, 那么第一个弱学习器的训练依赖于数据分布 $\mathcal{D}_0$. AdaBoost 根据第一个弱学习器的训练结果将训练集数据分布调整为 $\mathcal{D}_1$, 然后基于 $\mathcal{D}_1$ 训练第二个弱学习器. 依次类推, 训练完前 $t-1$ 个学习器之后的数据分布变为 $\mathcal{D}_{t-1}$. 根据以上描述并结合 “加性模型”(Additive Model), 请推导 AdaBoost 调整数据分布的具体过程, 即 $\mathcal{D}_t$ 与 $\mathcal{D}_{t-1}$ 的关系;

$$\begin{aligned}
\mathcal{D}_t(\boldsymbol{x}) &= \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]} \\
&= \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-2}(\boldsymbol{x})}e^{-f(\boldsymbol{x})\alpha_{t-1}h_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]} \\
&= \mathcal{D}_{t-1}(\boldsymbol{x}) \cdot e^{-f(\boldsymbol{x})\alpha_{t-1}h_{t-1}(\boldsymbol{x})} \frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-2}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}
\end{aligned}$$

2. AdaBoost 算法可以拓展到 N 分类问题. 现有一种设计方法, 将样本标记编码为 N 维向量 $\boldsymbol{y}$, 其中目标类别对应位置的值为 1, 其余类别对应位置的值为 $-\frac{1}{N-1}$. 这种编码的一种性质是 $\sum_{n=1}^{N} \boldsymbol{y}_n = 0$, 即所有类别对应位置的值的和为零. 同样地, 学习器的输出为一个 N 维向量, 且约束其输出结果的和为零, 即: $\sum_{n=1}^{N}[h_t(\boldsymbol{x})]_n = 0$. $[h_t(\boldsymbol{x})]_n$ 表示基分类器输出的 N 维向量的第 n 个值. 在这种设计下, 多分类情况下的指数损失为:

$$\ell_{\text{multi-exp}} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{y}_n[H(\boldsymbol{x})]_n}\right] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-\frac{1}{N}\boldsymbol{y}^{\top}H(\boldsymbol{x})}\right]. \tag{4}$$

请分析为何如此设计;

设计优点:

- 指数损失函数$e^{-\frac{1}{N}\boldsymbol{y}^{\mathrm{T}}H(\boldsymbol{x})}$: 预测值 $H(\boldsymbol{x})$ 与真实值 $\boldsymbol{y}$越接近，$e^{-\frac{1}{N}\boldsymbol{y}^{\mathrm{T}}H(\boldsymbol{x})}$ 就越小，满足损失函数应有的语义
- $[H(\boldsymbol{x})]_n = 0$与$\sum_{n=1}^{N}\boldsymbol{y}_n = 0$相统一，起到了去中心化的作用，去除均值对结果的影响，使过程和结果更加直观
- $\frac{1}{N}$ 用于标准化损失函数，避免损失函数$e^{-\frac{1}{N}\boldsymbol{y}^{\mathrm{T}}H(\boldsymbol{x})}$ 的值受N的大小的影响

3. 教材 8.2 节已经证明 AdaBoost 在指数损失下得到的决策函数 $\operatorname{sign}(H(\boldsymbol{x}))$ 可以达到贝叶斯最优误差. 仿照教材中的证明, 请从贝叶斯最优误差的角度验证式(4)的合理性.

贝叶斯最优分类器:

$$\arg\max_{\boldsymbol{y}} P(\boldsymbol{y} \mid \boldsymbol{x})$$

验证指数损失的合理性即要证明:

$$\arg\min \ell_{\text{multi-exp}} = \arg\max_{\boldsymbol{y}}\left[\boldsymbol{y}^{\mathrm{T}}H(\boldsymbol{x})\right] = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y} \mid \boldsymbol{x})$$

对$H(\boldsymbol{x})$求偏导:

$$\begin{aligned}\frac{\partial \ell_{\text{multi-exp}}}{\partial H(\boldsymbol{x})} &= -\frac{1}{N}e^{-\frac{1}{N}\boldsymbol{y}^{\mathrm{T}}H(\boldsymbol{x})}\boldsymbol{y}\\ &= -e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x})=1 \mid \boldsymbol{x}) + (N-1)e^{\frac{1}{N-1}H(\boldsymbol{x})}P(f(\boldsymbol{x})=-\frac{1}{N-1} \mid \boldsymbol{x})\\ &= 0\end{aligned}$$

解得:

$$H(\boldsymbol{x}) = \frac{N-1}{N}\ln\frac{1}{N-1}\ln\frac{P(f(x)=1 \mid \boldsymbol{x})}{P(f(x)=-\frac{1}{N-1} \mid \boldsymbol{x})}$$

因此有

$$\begin{aligned}\operatorname{sign}(H(\boldsymbol{x})) &= \operatorname{sign}\left(\frac{N-1}{N}\ln\frac{1}{N-1}\ln\frac{P(f(x)=1 \mid \boldsymbol{x})}{P(f(x)=-\frac{1}{N-1} \mid \boldsymbol{x})}\right)\\ &= \begin{cases}1, & P(f(x)=1 \mid \boldsymbol{x}) > P(f(x)=-\frac{1}{N-1} \mid \boldsymbol{x})\\ -1, & P(f(x)=1 \mid \boldsymbol{x}) < P(f(x)=-\frac{1}{N-1} \mid \boldsymbol{x})\end{cases}\\ &= \arg\max_{y\in\{-\frac{1}{N-1},1\}} P(f(x)=y \mid \boldsymbol{x}),\end{aligned}$$

所以若指数损失函数最小化，则分类错误率也最小化。即指数损失是合理的

五、Bagging

考虑一个回归学习任务 $f:\mathbb{R}^d \to \mathbb{R}$. 假设已经学得 T 个学习器 $\{h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x})\}$. 将学习器的预测值视为真实值项加上误差项:

$$h_t(\boldsymbol{x}) = y(\boldsymbol{x}) + \epsilon_t(\boldsymbol{x}). \tag{5}$$

每个学习器的期望平方误差为 $\mathbb{E}_{\boldsymbol{x}}[\epsilon_t(\boldsymbol{x})^2]$. 所有学习器的期望平方误差的平均值为:

$$E_{av} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}}[\epsilon_t(\boldsymbol{x})^2]. \tag{6}$$

T 个学习器得到的 Bagging 模型为:

$$H_{bag}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}h_t(\boldsymbol{x}). \tag{7}$$

Bagging 模型的误差为:

$$\epsilon_{bag}(\boldsymbol{x}) = H_{bag}(\boldsymbol{x}) - y(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}\epsilon_t(\boldsymbol{x}), \tag{8}$$

其期望平均误差为:

$$E_{bag} = \mathbb{E}_{\boldsymbol{x}}[\epsilon_{bag}(\boldsymbol{x})^2]. \tag{9}$$

1. 假设 $\forall t \neq l$, $\mathbb{E}_{\boldsymbol{x}}[\epsilon_t(\boldsymbol{x})] = 0$, $\mathbb{E}_{\boldsymbol{x}}[\epsilon_t(\boldsymbol{x})\epsilon_l(\boldsymbol{x})] = 0$. 证明:

$$E_{bag} = E_{av}. \tag{10}$$

$$\begin{aligned}
E_{bag} &= \frac{1}{T^2}\mathbb{E}_{\boldsymbol{x}}\left[\left(\sum_{t=1}^{T}\epsilon_t(\boldsymbol{x})\right)^2\right] \\
&= \frac{1}{T^2}\mathbb{E}_{\boldsymbol{x}}\left[\sum_{t=1}^{T}\epsilon_t(\boldsymbol{x})^2 + 2\sum_{t<l}\epsilon_t(\boldsymbol{x})\epsilon_l(\boldsymbol{x})\right] \\
&= \frac{1}{T^2}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})^2\right] + \frac{2}{T^2}\sum_{t<l}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})\epsilon_l(\boldsymbol{x})\right] \\
&= \frac{1}{T^2}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})^2\right] \\
&= \frac{1}{T}E_{av}
\end{aligned}$$

2. 请证明无需对 $\epsilon_t(\boldsymbol{x})$ 做任何假设, $E_{bag} \leq E_{av}$ 始终成立

$$\begin{aligned}
E_{bag} &= \frac{1}{T^2}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})^2\right] + \frac{2}{T^2}\sum_{t<l}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})\epsilon_l(\boldsymbol{x})\right] \\
&= \frac{1}{T}E_{av} + \frac{2}{T^2}\sum_{t<l}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})\epsilon_l(\boldsymbol{x})\right] \\
&\leq \frac{1}{T}E_{av} + \frac{1}{T^2}\sum_{t<l}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})^2 + \epsilon_l(\boldsymbol{x})^2\right] \\
&= \frac{1}{T}E_{av} + \frac{T-1}{T^2}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}}\left[\epsilon_t(\boldsymbol{x})^2\right] \\
&= E_{av}
\end{aligned}$$

六、k-means

教材 9.4.1 节介绍了最经典的原型聚类算法—k 均值算法 (k-means). 给定包含 m 个样本的数据集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$, 其中 k 是聚类簇的数目, k 均值算法希望获得簇划分 $\mathcal{C}=\{C_1,C_2,\cdots,C_k\}$ 使得教材式 (9.24) 最小化, 目标函数如下:

$$E=\sum_{i=i}^{k}\sum_{\boldsymbol{x}\in C_i}\|\boldsymbol{x}-\boldsymbol{u}_i\|^2\ . \tag{11}$$

其中 $\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k$ 为 k 个簇的中心. 目标函数 E 也被称作均方误差和 (Sum of Squared Error, SSE), 这一过程可等价地写为最小化如下目标函数

$$E(\ ,\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k)=\sum_{i=1}^{m}\sum_{j=1}^{k}\Gamma_{ij}\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2\ . \tag{12}$$

其中 $\ \in\mathbb{R}^{m\times k}$ 为指示矩阵 (indicator matrix) 定义如下: 若 $\boldsymbol{x}_i$ 属于第 j 个簇, 即 $\boldsymbol{x}_i\in C_j$, 则 $\Gamma_{ij}=1$, 否则为 0. k 均值聚类算法流程如算法1中所示 （即教材中图 9.2 所述算法）. 请回答以下问题:

算法 1 k 均值算法

1: 初始化所有簇中心 $\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k$;
2: **repeat**
3: **Step 1:** 确定 $\{\boldsymbol{x}_i\}_{i=1}^m$ 所属的簇, 将它们分配到最近的簇中心所在的簇.

$$\Gamma_{ij}=\begin{cases}1, & \|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2\le\|\boldsymbol{x}_i-\boldsymbol{\mu}_{j'}\|^2,\forall j'\\ 0, & \text{otherwise}\end{cases} \tag{13}$$

4: **Step 2:** 对所有的簇 $j\in\{1,\cdots,k\}$, 重新计算簇内所有样本的均值, 得到新的簇中心 $\boldsymbol{\mu}_j$:

$$\boldsymbol{\mu}_j=\frac{\sum_{i=1}^{m}\Gamma_{ij}\boldsymbol{x}_i}{\sum_{i=1}^{m}\Gamma_{ij}} \tag{14}$$

5: **until** 目标函数 J 不再变化.

1. 请证明, 在算法1中, Step 1 和 Step 2 都会使目标函数 J 的值降低（或不增加）;
2. 请证明, 算法1会在有限步内停止;
3. 请证明, 目标函数 E 的最小值是关于 k 的非增函数.

(1)目标函数不增

此类证明做差最有说服力

Step1

对于新分配的簇中心有新的 Γ' :

$$\Gamma'_{ij}=\begin{cases}1, & \|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2\le\|\boldsymbol{x}_i-\boldsymbol{\mu}_{j'}\|^2,\forall j'\\ 0, & \text{otherwise}\end{cases}$$

因此

$$\begin{aligned}E'-E&=\sum_{i=1}^{m}\sum_{j=1}^{k}\Gamma'_{ij}\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2-\sum_{i=1}^{m}\sum_{j=1}^{k}\Gamma_{ij}\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2\\&=\sum_{i=1}^{m}\sum_{j=1}^{k}\left(\Gamma'_{ij}-\Gamma_{ij}\right)\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2\\&=\sum_{i=1}^{m}\left(\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2-\|\boldsymbol{x}_i-\boldsymbol{\mu}_{j'}\|^2\right)\\&\le 0\end{aligned}$$

Step2

因为每个点所属的簇不会改变，我们只需要证明在Step2中：

$$\forall j \in \{1,2,\ldots,k\},\ \sum_{i=1}^{m} \Gamma_{ij} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2 = \sum_{\boldsymbol{x} \in C_j} \|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2 \text{不增加}$$

我们有以下几条已知性质：

$$\boldsymbol{\mu}_j^{'} = \frac{\sum_{i=1}^{m} \Gamma_{ij} \boldsymbol{x}_i}{\sum_{i=1}^{m} \Gamma_{ij}},\ \text{满足} \sum_{\boldsymbol{x} \in C_j} (\boldsymbol{x} - \boldsymbol{\mu}_j^{'}) = 0$$

$$|C_j| = \sum_{i=1}^{m} \Gamma_{ij}$$

$$\sum_{i=1}^{m} \Gamma_{ij} \boldsymbol{x}_i = \boldsymbol{x}_i$$

因此

$$\begin{aligned}
E_j' - E_j &= \sum_{i=1}^{m} \Gamma_{ij} \left(\left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j' \right\|^2 - \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2 \right) \\
&= \sum_{i=1}^{m} \Gamma_{ij} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j' + \boldsymbol{x}_i - \boldsymbol{\mu}_j \right)^{\mathrm{T}} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j' - \boldsymbol{x}_i + \boldsymbol{\mu}_j \right) \\
&= \sum_{i=1}^{m} \Gamma_{ij} \left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j' \right)^{\mathrm{T}} \left(2\boldsymbol{x}_i - \boldsymbol{\mu}_j' - \boldsymbol{\mu}_j \right) \\
&= |C_j| \left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j' \right)^{\mathrm{T}} \left(2 \frac{\sum_{i=1}^{m} \Gamma_{ij} \boldsymbol{x}_i}{\sum_{i=1}^{m} \Gamma_{ij}} - \boldsymbol{\mu}_j' - \boldsymbol{\mu}_j \right) \\
&= |C_j| \left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j' \right)^{\mathrm{T}} \left(2\boldsymbol{\mu}_j' - \boldsymbol{\mu}_j' - \boldsymbol{\mu}_j \right) \\
&= -|C_j| \left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j' \right)^{\mathrm{T}} \left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j' \right) \\
&\leq 0
\end{aligned}$$

(2)划分有限--算法有限步截止

反设：算法不会在有限步停止，即目标函数E在本算法中每次运行都一定会变小，是严格单调减的

由于$E > 0$有下界，所以一定收敛

而每次算法结束得到的E都是不一样的，且都一一对应了样本的一种k-划分。由于m是有限的，所以样本的k-划分是有限的，所以算法结束后得到的结果是有限的。

而这与算法不会在有限步内结束矛盾，所以假设不成立，算法会在有限步内结束

(3)目标函数关于k不增

反设：当k变成k+1时，E的最小值反而更大，即$E_k < E_{k+1}$

下面我们基于使E_k最小的聚类结果来构造一个小于E_{k+1}的$E_{k+1}^{'}$，并推出矛盾：

由于样本并没有变化，所以我们可在k+1时仍然基于让E_k最小的那k个簇中心，划分情况也与E_k时完全相同，唯一的区别是，任意的在k个簇中选择一个包含大于一个点的簇，从中分离出一个点作为第k+1个簇，并把这个点设置为第k+1个簇的簇中心。

在这个构造中，只有第k+1个簇中这个点在目标函数中的表达式变成了0，其余点的距离表达式都没有变，因此此时的$E'_{k+1} < E_{k+1}$，与E_{k+1}的最小性矛盾

因此假设不成立，目标函数的最小值是关于k的非增函数