

2025 模式识别 作业三

人工智能学院 221300079 王俊童

2025.4.15

221300079 王俊童, 人工智能学院

1 问题一

a. 用 python 完成, 结果如下:

10 维样本的 12 范数统计:
均值: 2.8871
最小值: 2.0136
最大值: 4.0669

b. 同样, 用 python 可以得到:

100 维样本的 12 范数统计:
均值: 9.9181
最小值: 8.8613
最大值: 11.0844

1000 维样本的 12 范数统计:
均值: 31.7088
最小值: 30.6157
最大值: 32.9135

10000 维样本的 12 范数统计:
均值: 99.9484
最小值: 98.8400
最大值: 101.5054

100000 维样本的 12 范数统计:
均值: 316.4926
最小值: 315.0430
最大值: 318.7029

c. 可以提出一个猜想是: 随着维度 d 变高, 样本的 12 范数更加接近 \sqrt{d} 的取值。他们的相对波动减少
可以理解为: 维度增加, 对于 d 维度的标准高斯分布, 12 范数越来越集中于 \sqrt{d} 附近。

相对波动:

$$relative\ variance = \frac{\max - \min}{mean}$$

100 维样本的 l2 范数统计：
波动：0.2241

1000 维样本的 l2 范数统计：
波动：0.0725

10000 维样本的 l2 范数统计：
波动：0.0267

100000 维样本的 l2 范数统计：
波动：0.0116

d. 如果用严谨的数学语言描述这个现象，可以解释为：

令 $X = (X_1, X_2, \dots, X_d)^\top \sim \mathcal{N}(0, I_d)$ ，且其 l2 范数满足 $\|X\|_2 = \sqrt{\sum_{i=1}^d X_i^2}$ ，其 l2 范数的平方服从自由度为 d 的 χ^2 分布。那么有以下形式的式子在概率意义下成立：

$$\frac{\|X\|_2}{\sqrt{d}} \xrightarrow{P} 1, \quad \text{when } d \rightarrow \infty$$

当然这是一个服从大数定律形式的，或者可以有一个 concentration 的形式：

$$P(|\frac{\|X\|_2}{\sqrt{d}} - 1| \geq \epsilon) \rightarrow 0, \quad \text{when } d \rightarrow \infty$$

e. 虽然集中不等式和大数定律忘的差不多了，但是可以知道这个东西服从一定的集中不等式，或者从大数定律来解释也是可以的，最后会有一个逼近性。

2 问题二

- 对于核函数的半正定性，有： $y^\top K_1 y \geq 0, y^\top K_2 y \geq 0$ ，对于 $K = K_1 + K_2$ ，有 $y^\top K y = y^\top K_1 y + y^\top K_2 y \geq 0$ ，所以合法
- 同理 $K = K_1 - K_2$ ，有 $y^\top K y = y^\top K_1 y - y^\top K_2 y$ ，但这个不是合法的核函数，可以有一个例子是如果 $K = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$ 那么对于一个 $y = (1, 0)^\top$ ，有 $K_1 = 0, K_2 = x^\top y$ ，这个是不合法的，不正定。
- 对于 $y^\top K_1 y \geq 0$ ，且 $\alpha \geq 0$ ，所以 $\alpha y^\top K_1 y \geq 0$ 成立，所以合法。
- 同上 $\alpha y^\top K_1 y \geq 0$ 成立，但是 $-\alpha y^\top K_1 y \leq 0$ 成立，所以不合法。
- 对于 $K = K_1 K_2$ ，这个是逐元素相乘，所以可以得到我们需要证明 $K = K_1 K_2 \rightarrow y^\top K_1 K_2 y \geq 0$ ，所以我们需要证明矩阵 K_1, K_2 是半正定的，我们知道他们分别是半正定，所以我们可以让 $K_1 = U^\top U$ 。所以 $y^\top K_1 K_2 y = \sum_{i=1}^n \sum_{j=1}^n y_i^\top K_1 K_2 y_j = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n (u_{ki} y_i) K_{ij} (u_{kj} y_j) \geq 0$ ，所以合法。
- 因为 ϕ 是一个合法的从 $d \rightarrow d'$ 的合法映射，所以对于 $\{x_1, x_2, \dots, x_n\}$ ，可以有 $\{\phi(x_1), \phi(x_2), \dots, \phi(x_n)\}$ 对于原核函数也是在核函数处理范围内，所以对于原来的 $K = \kappa_3(\phi(x), \phi(y))$ 可以得到 $K_3 = \kappa_3(\phi(x), \phi(y))$ ， $K = K_3$ ，由于 K_3 一定合法，所以 K 合法。

3 问题三

a. 因为要施加一个 k ，根据题目描述，可以得到 k 对于样本 i 有：

$$k_i = \begin{cases} 1, & y_i = 1 \\ k, & y_i = -1 \end{cases}$$

若 $f(x) = \mathbf{w}^\top \mathbf{x}_i + b$ 所以优化问题变为:

$$\begin{aligned} \min_{w,b} & \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^n k_i \xi_i \\ \text{s.t.} & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad 1 \leq i \leq n \\ & \xi_i \geq 0, \quad 1 \leq i \leq n \end{aligned}$$

b. 把这个问题写成 lagrange 形式:

$$L = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^n k_i \xi_i + \sum_{i=1}^n \alpha_i (1 - \xi_i - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) - \sum_{i=1}^n \mu_i \xi_i$$

求 \mathbf{w}, b, ξ_i 的导数:

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \\ 0 &= \sum_{i=1}^n \alpha_i y_i \\ k_i C &= \alpha_i + \mu_i \end{aligned}$$

所以带入可以得到:

$$L = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

所以对偶问题为:

$$\begin{aligned} \max_{\alpha} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} & \sum_{i=1}^n \alpha_i y_i = 0, \quad 1 \leq i \leq n, \\ & 0 \leq \alpha_i \leq k_i C, \quad 1 \leq i \leq n, \end{aligned}$$

KKT 条件为:

$$\begin{cases} \mu_i \geq 0, \alpha_i \geq 0 \\ (1 - \xi_i - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \geq 0 \\ \alpha_i (1 - \xi_i - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) = 0 \\ \xi_i \geq 0 \\ \mu_i \xi_i = 0 \end{cases}$$

4 问题四

a. 基本假设: 属性的条件独立性假设。

优点: 分类效率稳定, 可以增量学习, 对缺失不太敏感, 可以用小数据算置信度高的结果。

缺点: 条件独立性假设一般是不成立的, 所以如果有关联的时候, 朴素贝叶斯效果不好。

这个算法是参数化的, 参数空间是有限的。

b. 根据 bayes 算法, 可以计算出: $P(A) = \frac{1}{3}, P(B) = \frac{1}{3}, P(C) = \frac{1}{3}$.

因为都是高斯分布, 可以计算每个类别属性对应的均值和方差。

- (A,1): $\mu = 2.5, \sigma = 1.25$
- (A,2): $\mu = 3.5, \sigma = 1.25$
- (B,1): $\mu = 2.5, \sigma = 1.25$
- (B,2): $\mu = 5.5, \sigma = 1.25$
- (C,1): $\mu = 5.5, \sigma = 1.25$
- (C,2): $\mu = 2.5, \sigma = 1.25$

根据 bayes, 由于分母一样:

$$P(Y_k|X) = \frac{P(XY_k)}{P(X)} = \frac{P(X|Y_k)P(Y_k)}{\sum_j P(X|Y_j)P(Y_j)} \sim P(X|Y_k)P(Y_k)$$

所以 (只给出第一个, 后面都一样):

- $P_{(x_1|A)} = 1/(\sqrt{2\pi}\sqrt{1.25}) * \exp(-(2 - 2.5)^2/(2 * 1.25)) = 0.323$
- $P_{(x_1|B)} = 0.323$
- $P_{(x_1|C)} = 0.003$
- $P_{(x_2|A)} = 0.145$
- $P_{(x_2|B)} = 0.003$
- $P_{(x_2|C)} = 0.323$
- $P_{(y_1|A)} = 0.003$
- $P_{(y_1|B)} = 0.003$
- $P_{(y_1|C)} = 0.323$
- $P_{(y_2|A)} = 0.029$
- $P_{(y_2|B)} = 0.0001$
- $P_{(y_2|C)} = 0.145$

所以:

- $P(A|(x_1, x_2)) = 1/4 + 0.323 + 0.145 = 0.016$
- $P(B|(x_1, x_2)) = 0.00029$
- $P(C|(x_1, x_2)) = 0.00029$
- $P(A|(y_1, y_2)) = 0.0000259$
- $P(B|(y_1, y_2)) = 0.0000259$
- $P(C|(y_1, y_2)) = 0.016$

所以 x 会为 A 类, y 会为 C 类。

5 问题五

a. 信息熵如下:

$$\begin{aligned}x : H &= -(0.5 \log(0.5) + 0.25 \log(0.25) + 0.25 \log(0.25)) = 1.5 \\y : H &= -(0.5 \log(0.5) + 0.5 \log(0.5)) = 1 \\\hat{x} : H &= -(0.5 \log(0.5) + 0.5 \log(0.5)) = 1\end{aligned}$$

有损, 因为在 x 输入是 3 的时候, 会被还原为 2. 只有这种情况有损, 因为信息熵减少了。

b. D 如果用 MSE 的话, 且 X 是一个遵守 $x \sim P(X), \text{for } i = 1, \dots, n$, y 遵守 $y \sim Q(Y), \text{for } j = 1, \dots, m$, 可以表示如下:

$$J = \frac{1}{n} \sum_{i=1}^n P_i (x_i - g(f(x_i)))^2 - \lambda \sum_{j=1}^m Q_j \log Q_j$$

对于这个式子有一个可行的优化方式是, 从 x 到 y 进来不要引入损失, 让码率最小, 所以 $f=0$, 所以 $g(y)$ 有两个取值:

$$g(y) = 1: D = 1.25/3 = 0.4167$$

$$g(y) = 2: D = 0.75/3 = 0.25$$

所以可以计算 a 中的这个系统性能如下: $J_a = 0.25/3 + \lambda$

b 的性能为: $J_b = 0.25$

所以

- $\lambda = 0.1 : J_a = 0.25/3 + 0.1 = 0.183, J_b = 0.25$
- $\lambda = 1 : J_a = 0.25/3 + 1 = 1.083, J_b = 0.25$
- $\lambda = 10 : J_a = 0.25/3 + 10 = 10.183, J_b = 0.25$

在 0.1 的时候 a 好, 1 和 10 的时候 b 好。

c. 如果 y 是个两类的判别, 那么输入类别只有在 2 类概率的时候才会无损, 因为只有两类别的时候信息熵才会为 1, 这样才能保证无损, 如果是 a 的话, 信息熵铁定大于 1 的, 所以肯定不能保证无损。

如果 $y \in \mathbb{Z}$:

$$J = \frac{1}{n} \sum_{i=1}^n P_i (x_i - g(f(x_i)))^2 - \lambda \sum_{j=1}^{\infty} Q_j \log Q_j$$

那么就可以取到无穷了。

如果 $y \in \mathbb{R}$, 意味着他可以取连续值了 (其实我 x 的取值写的不标准啊, 应该是个积分而不是级数形式):

$$J = \frac{1}{n} \sum_{i=1}^n P_i (x_i - g(f(x_i)))^2 - \lambda \int_{-\infty}^{\infty} q(y) \ln q(y) dy$$

- 表达能力: R 比 Z 厉害, 因为他可以取连续值。
- 编码难度: R 更难, 因为连续值更难编码, 一般离散值是直接存储的。
- 编解码器设计难度: R 更难编码, 也更难设计, 因为连续值不好评估, Z 更好设计一些。

d. 可以。根据定义可以得到:

$$J = D + \lambda R \rightarrow \text{loss} = \text{MSE}(x, \hat{x}) + \lambda Q(y)$$

其中 Q 是一个可以设计的码率的 loss 比如 $Q(y) = -\ln(y)$, 那么这样 loss 就可以出现了, 那么类似于 attention 的机制那种, 然后这个模型就变成了神经网络的训练, encoder 一个, decoder 一个。那么 $\ln(y)$ 这种东西就可以用求分布和迭代来做了。

- e. 因为训练过程中: $\hat{y} = y + \epsilon$, 相当于噪声, 这种小规模噪声可以训练出有效的梯度:

$$\frac{d\hat{y}}{dy} = 1, \quad \frac{d[y]}{dy} = 0$$

这样就可以回传梯度, 不然梯度为 0. 从概率分布角度来说:

$$f_{y+\epsilon}(y) = \int_{-0.5}^{0.5} f_y(y-x)dx, \quad P([y] = y) = \int_{-0.5}^{0.5} f_y(y-x)dx$$

他们的概率分布确实是一样的, 这样就合理了。

- f. 我们构造两个神经网络的 encoder 和 decoder 神经网络如下, 基于 e 问:

2. 定义压缩模型 (集成加性噪声离散化)

```
class CompressionModel(nn.Module):
    def __init__(self, latent_dim=2, lambda_val=1.0):
        super().__init__()
        self.lambda_val = lambda_val
        self.encoder = nn.Sequential(
            nn.Linear(1, 32),
            nn.ReLU(),
            nn.Linear(32, 16),
            nn.ReLU(),
            nn.Linear(16, latent_dim) # 输出连续值
        )
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 16),
            nn.ReLU(),
            nn.Linear(16, 32),
            nn.ReLU(),
            nn.Linear(32, 1)
        )

    # 根据第 e 问题的实现
    def quantize(self, y, training=False):
        if training:
            # 训练时: 添加均匀噪声  $U(-0.5, 0.5)$ 
            noise = torch.rand_like(y) - 0.5
            return y + noise
        else:
            # 测试时: 直接取整
            return torch.round(y)

    def forward(self, x, training=False):
        y = self.encoder(x)
        y_quantized = self.quantize(y, training)
        x_recon = self.decoder(y_quantized)
        return x_recon, y_quantized

    def compute_loss(self, x):
        #  $J = D + \lambda R$ 
        x_recon, y_quantized = self(x, training=True)
        # 重构误差 (MSE)
        D = torch.mean((x - x_recon) ** 2)
```

```
# 码率 (信息熵估计)
hist = torch.histc(y_quantized, bins=20, min=-10, max=10)
prob = hist / torch.sum(hist)
R = -torch.sum(prob * torch.log2(prob + 1e-10)) # 防止 log(0)
# 总损失
return D + self.lambda_val * R
```

我们完全参考上面的代码，把训练集弄一个 ϵ ，然后熵的计算参考上面 $D + \lambda R$.

$\lambda = 0.1, 1, 10$ 上面取值做实验，结果如下：我们把重构和重构前的数据可视化一下：可以看出，MSE

```
Final Results:
λ=0.1: J=0.41, MSE=0.01, Entropy=4.06
λ=1: J=4.13, MSE=0.01, Entropy=4.13
λ=10: J=39.55, MSE=0.01, Entropy=3.95
```

Figure 1: result

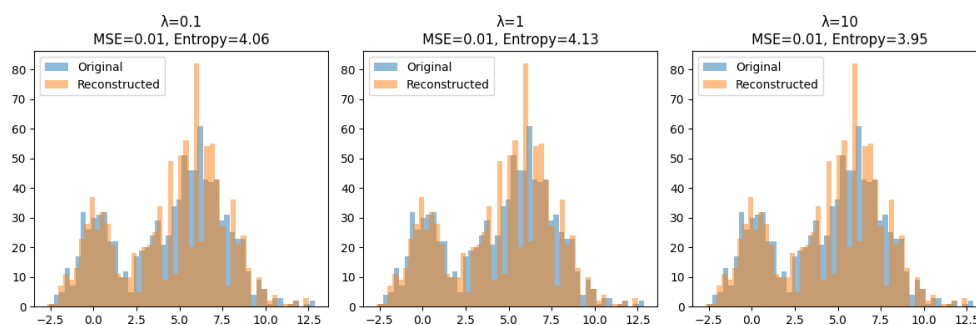


Figure 2: compression

都比较小， λ 对于码率的限制确实是有效果的。