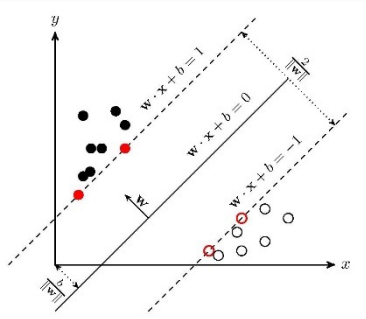
媒体与认知

Media and Cognition



V 1.0

清华大学电子工程系

唐沛 助教

Email: tp21@mails.tsinghua.edu.cn

第三次作业的考察目标

编程部分：

- 一、在神经网络中引入hinge loss实现支持向量机
- 二、熟悉支持向量机的代码实现
  - **hinge loss的前向、反向过程实现**
  - **libsvm库**
- 三、复习神经网络的训练过程
  - **数据加载**
  - **模型定义**
  - **训练/验证**

理论部分：

- 四、特征提取与降维
  - **尺度变换特性**
  - **特征变换：PCA/LDA**
- 五、统计模式识别
  - **贝叶斯决策**
  - **正态分布下的贝叶斯决策**
  - **KNN、KMeans、EM算法**
- 六、支持向量机
  - **线性可分：拉格朗日乘子法**
  - **非线性问题：核函数法**
  - **近似线性可分：引入松弛变量ξ**

作业说明

- 任务一：单选题

共5题：涵盖了统计模式识别、特征提取与降维和支持向量机的相关内容。

- 任务二：计算题

共5题：贝叶斯决策的简单应用；PCA算法；正态分布贝叶斯决策的分界面计算；KMeans算法；Lagrange乘子法。

- 任务三：实现hinge loss模拟支持向量机并运行自动评判程序

需要同学们补全svm_hw.py中的代码，主要包括：Linear类的前向、反向计算；Hinge类的前向、反向计算；SVM_HINGE类的模型参数定义；

补全后运行：python check.py 自动检查svm_hw.py的正确性。

作业说明

➢ 任务四：使用支持向量机完成线性分类任务：字符图片前景/背景特征分类，对比libsvm库的分类结果和使用线性层+hinge loss的模拟结果

1. Hinge loss模拟SVM的训练及验证：

使用线性模型+hinge loss模拟SVM，补全classify_hw.py中的代码，主要包括：

a) FeatureDataset类：实现数据特征的载入方法

b) train_val_hinge函数：实现线性模型的训练、验证过程

2.可视化分类结果：

通过指定参数，分别使用线性模型+hinge loss模拟SVM、以及libsvm库对数据集进行分类，例如：

作业说明

2.可视化分类结果：

通过指定参数，分别使用线性模型+hinge loss模拟SVM、以及libsvm库对数据集进行分类，例如：

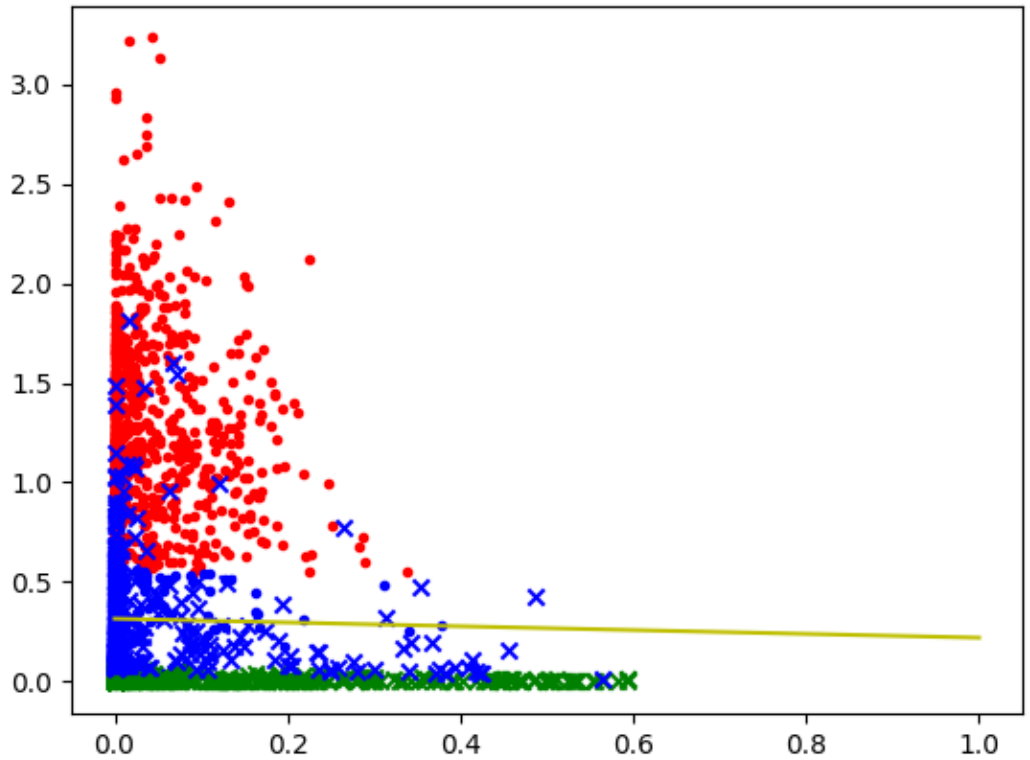
python classify_hw.py --mode hinge

训练/验证线性模型+hinge loss

python classify_hw.py --mode baseline

训练/验证libsvm库的SVM模型

蓝色部分为支持向量



作业说明

3. 调整正则化系数C，体会不同的C对分类效果的影响：

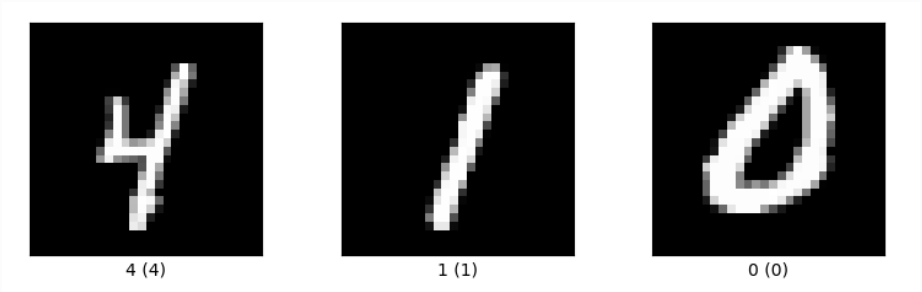
分别设置不同的正则化系数C=0.0001,0.001,0.01,0.1,1,10, 运行两种模型，例如：

python classify_hw.py --mode hinge --C 0.1

➢任务五：使用SVM算法在MNIST数据集上进行分类

提供了手写数字数据集MNIST，和处理数据集的代码proces_mnist.py。

MNIST数据集每一张图片维度为28x28，包含从0到9一共10个数字。



作业说明

需要先补全process_mnist.py中关于PCA降维的代码，对高维数据进行降维，再使用之前实现的SVM算法进行分类处理。提取特征的命令如下：

python process_mnist.py --class_0 0 --class_1 1 --feat_dim 10
提取0和1对应的图片特征，并降维到10维

python classify_hw.py --feat_dim 10 --train_file_path 'MNIST/train.npy' --val_file_path 'MNIST/val.npy'
使用SVM算法对提取的特征进行分类

补全PCA代码时，需要注意我们仅通过训练集计算均值和协方差，且将数据降维处理时，一般需要先将数据中心化，即减去均值。

作业说明

➢任务六：撰写作业报告

将HW3目录（含源代码文件）和作业报告打包为一个文件（例如 *.zip）提交到网络学堂。作业报告中包括选择题答案，计算题的解题步骤及答案、任务三、四、五运行结果及分析，本次作业遇到的问题及解决方法，对本次作业的意见及建议。

作业说明

➢ 特别说明：

参照前几次作业，本次作业需要使用pytorch，matplotlib， numpy库。

此外，本次作业还需要安装libsvm库，在windows、macos或ubuntu环境下，均可在命令提示符/终端执行下述指令安装libsvm库：

pip install –U libsvm-official

本次作业在cpu上运行只需约10s，建议同学们直接在cpu上运行即可。

打包文件不用放入data、MNIST等数据文件。

选择自选课题的同学完成选择题和计算题即可，编程部分汇报一下自选课题完成情况。

编程部分

1.1 Hinge Loss与SVM

➢ 引入松弛变量后的支持向量机：

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad s.t.\, y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \Rightarrow \xi_i \geq \max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0)$$

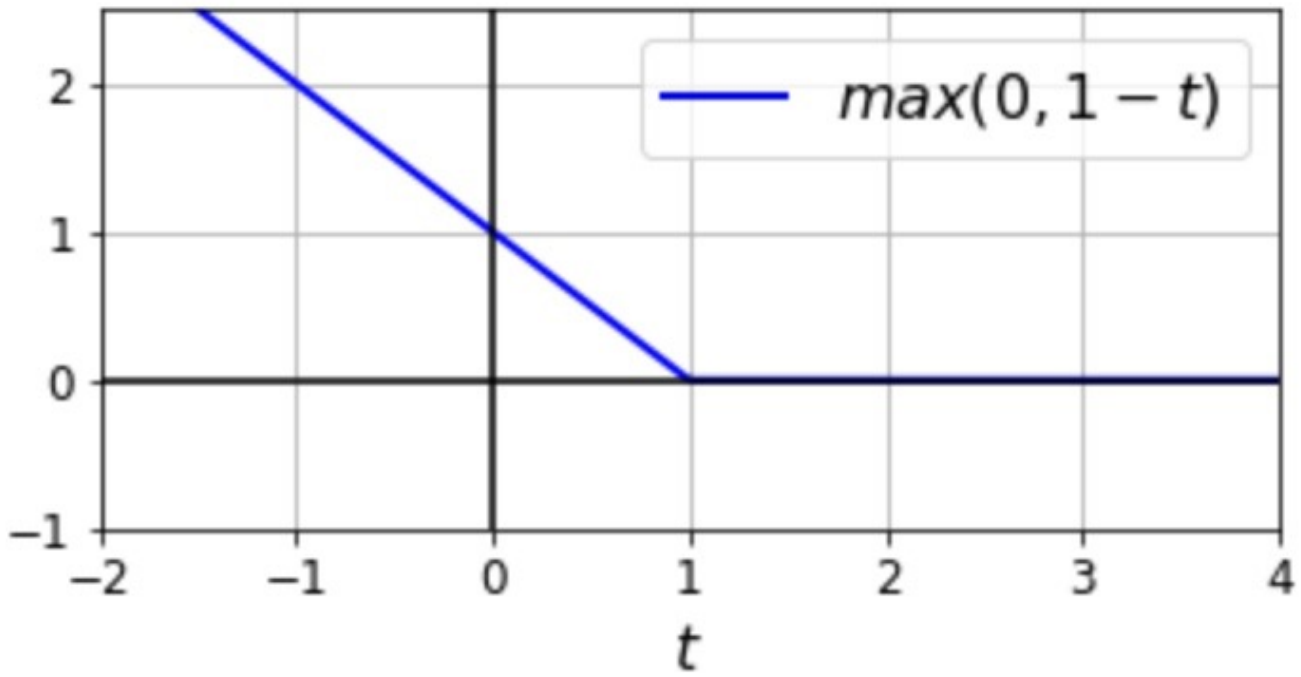
$$L(\mathbf{w}, b, \boldsymbol{\xi}, \gamma) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\gamma_i(max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0) - \xi_i)$$

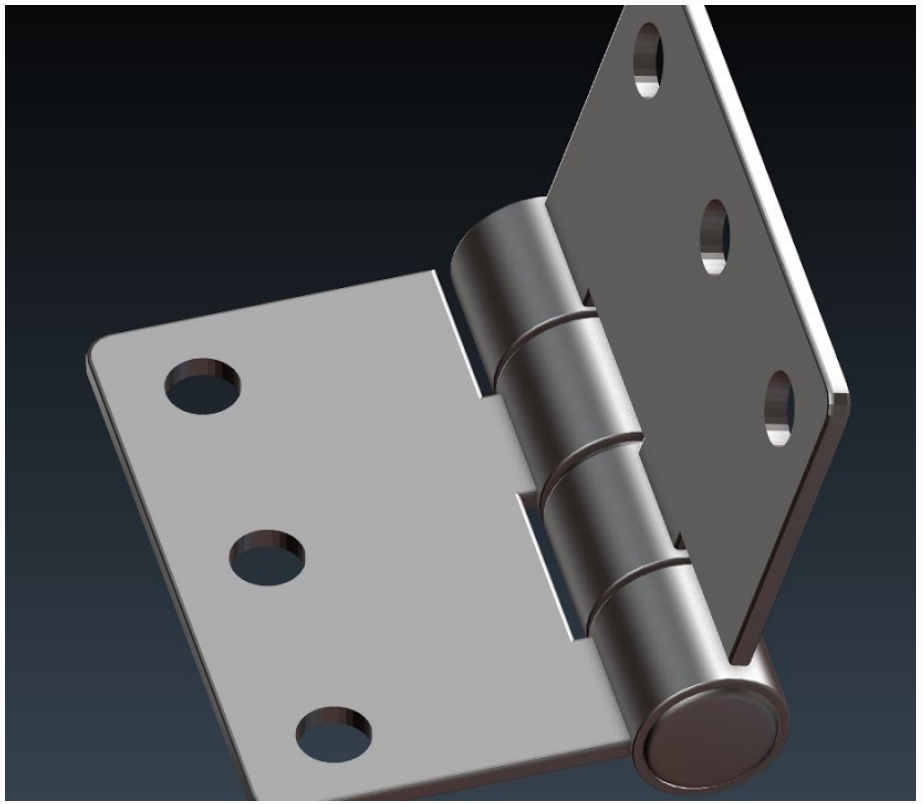
$$\frac{\partial L}{\partial \xi_i} = C - \gamma_i = 0 \Rightarrow \gamma_i = C \Rightarrow L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0)$$

1.2 Hinge Loss图像

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0)$$

$$hinge(t) = max(1 - t, 0)$$





2.1 Hinge Loss前向计算与误差反向传播

- 前向计算过程：

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} max(1 - y_i f(\mathbf{x}_i), 0)$$

其中max(·,0)操作可以用torch.nn.functional.relu(·)实现

- 误差反向传播过程：记$output_i = f(\mathbf{x}_i)$

$$\frac{\partial L}{\partial \mathrm{w}_i} = \mathrm{w}_i$$

$$\frac{\partial L}{\partial output_i} = C * \mathbb{I}\big(1 - y_i f(\mathbf{x}_i)\big) * (-y_i)，\ 其中\mathbb{I}(\cdot)为阶跃函数$$

$$grad\_w = grad\_loss * \frac{\partial L}{\partial \mathbf{w}}$$

$$grad\_\mathbf{output} = grad\_loss * \partial L/\partial \mathbf{output}$$

2.2 libsvm库

- Libsvm具有C++，Matlab，Python等多种调用接口

```
## 训练支持向量机
m = svm_train(train_labels, train_data, f'-c {C} -t 0')
## 根据支持向量计算决策函数
sv = m.get_sv_indices()
sv = [x-1 for x in sv]
sv_feature = train_data[sv, :]
sv_labels = train_labels[sv]
sv_coef = m.get_sv_coef()
sv_coef = [x[0] for x in sv_coef]
W = (np.reshape(sv_coef, (-1, 1)) * np.reshape(sv_feature, (-1, 2))).sum(0)
W = np.reshape(W, (1, 2))
b = (np.reshape(sv_labels, (-1, )) - (W * np.reshape(sv_feature, (-1, 2))).sum(1)).mean()
## 使用训练好的支持向量机预测验证集数据
svm_predict(val_labels, val_data, m)
```

2.2 libsvm库

m = svm_train(train_labels, train_data, f'-c {C} -t 0’)

-t 核函数的类型：

0：线性核

1：多项式核

2：高斯核

3：sigmoid核

4：自定义核

-c 松弛变量的正则化系数C

任务四要求大家使用线性核，并调整C的参数，观察C对结果的影响

2.2 libsvm库

sv = m.get_sv_indices()

sv = [x-1 for x in sv]

获得支持向量的索引

sv_coef = m.get_sv_coef()

sv_coef = [x[0] for x in sv_coef]

获得支持向量对应的权重$\alpha_i\ y_i$

svm_predict(val_labels, val_data, m)

使用训练好的支持向量机对新的数据进行预测

3.1 数据加载

- torch.utils.data.Dataset：定义数据集，并定义载入一个样本的方法

```
class FeatureDataset(torch.utils.data.Dataset):

    def __init__(self, file_path):

        self.data = np.load(file_path)
        self.labels = np.concatenate([np.ones(self.data.shape[0] // 2).astype(np.float32), -1.0 *
np.ones(self.data.shape[0] // 2).astype(np.float32)], axis=0)

    # define the __len__ function of the Dataset class
    def __len__(self):
        return ???

    # define the __getitem__ function of the Dataset class
    def __getitem__(self, item):
        feature = ???
        label = ???
        return feature, label
```

3.1 数据加载

- torch.utils.data.DataLoader：实现批量载入样本的功能

```
dataset = FeatureDataset(file_path)
return torch.utils.data.DataLoader(dataset, batch_size=batch_size, shuffle=True if 'train' in file_path
else False, num_workers=workers)
```

batch_size：指定批次的大小
shuffle：随机打乱数据的顺序（通常只在训练时打乱）
num_workers：指定载入数据的线程数量

3.2 模型定义

➢ 用线性模型实现支持向量机：

```
class Linear(torch.autograd.Function):

    @staticmethod
    def forward(ctx, x, W, b):
        output = ???
        ctx.save_for_backward(x, W, b)
        return output

    @staticmethod
    def backward(ctx, grad_output):
        x, W, b = ctx.saved_tensors
        batch, channels = x.shape
        grad_W = ???
        grad_b = ???
        return None, grad_W, grad_b
```

```
class SVM_HINGE(nn.Module):

    def __init__(self, in_channels, C):
        super().__init__()
        self.W = ???
        self.b = ???
        self.C = torch.tensor([[C]], requires_grad=False)

    def forward(self, x, label=None):
        output = Linear.apply(x, self.W, self.b)
        if label is not None:
            loss = Hinge.apply(output, self.W, label, self.C)
        else:
            loss = None
        output = (output > 0.0).type_as(x) * 2.0 - 1.0
        return output, loss
```

3.3 网络训练/验证

- 实现模型的训练和验证：

过程与前几次作业类似，进一步熟悉神经网络的训练过程：

实例化数据载入模块；

实例化模型（device、训练/验证模式，加载参数，…）；

实例化优化器（Adam， SGD，…）；

载入数据训练：

前向计算、计算损失函数、误差反向传播、更新参数，…

载入数据验证：

前向计算、计算准确率，…

保存模型；

理论部分

4.1 小波变换

- 对于平方可积函数$\psi(t)$，其傅里叶变换$\Psi(w)$满足容许条件

$$C_\psi = \int_0^\infty \frac{|\Psi(\omega)|^2}{\omega} d\omega < \infty$$

则称函数族$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$为连续小波函数。a>0为尺度因子；b为位移因子，其值可正可负（保持C_ψ不变）

小波变换$W_s(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} s(t)\,\overline{\psi(\frac{t-b}{a})}dt$, 横线表示取共轭

- 若窗函数g(t)的傅里叶变换为$G(\omega)$，则g(at)的傅里叶变换为$\frac{1}{|a|}G(\frac{\omega}{a})$（思考与尺度因子的关系）

4.2 特征变换

	主成分分析PCA	线性判别分析LDA
优化问题	$\min\{\varepsilon(M)\}$ $s.t.\Phi_i^T\Phi_i - 1 = 0$	$\max\frac{tr(W^T S_b W)}{tr(W^T S_w W)}$
优点	• 采用样本协方差矩阵的特征向量作为变换的基向量，与样本的统计特性完全匹配在 • 最小均方误差准则下，是最佳变换	• 在降维过程中可以使用类别的先验知识经验 • 在样本分类信息依赖均值而不是方差的时候，比PCA之类的算法较优
缺点	• 变换矩阵随样本数据而异，无快速算法	• 在样本分类信息依赖方差而不是均值的时候，降维效果不好 • 可能过度拟合数据 • 得到的投影向量 X 的有效维数最多是C－1， C为类别数（与协方差矩阵的秩有关）
不同点	• 无监督 • 最小均方误差准则下保留原始数据信息 • 取数据投影方差最大的方向	• 有监督 • 最小均方误差准则下区分多类数据 • 取分类性能最好的投影方向
相同点	降维方法，线性特征变换，特征值分解，高斯假设	

4.2 主成分分析PCA

➢PCA算法如下:

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
低维空间维数 d'.

过程:

1: 对所有样本进行中心化: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i$;

2: 计算样本的协方差矩阵 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$;

3: 对协方差矩阵 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ 做特征值分解;

4: 取最大的 d' 个特征值所对应的特征向量 $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'}$.

输出: 投影矩阵 $\mathbf{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'})$.

4.2 主成分分析PCA

- 思考：如果已知的不是样本集D，而是样本均值和协方差矩阵，如何进行PCA降维？

- （计算题）：给定样本集合，其均值为 $\mu = [1,2]^T$，样本协方差矩阵为 C，且已知 CU = U λ。其中 $U = \begin{bmatrix} 0.5 & -0.4 \\ 0.5 & 0.4 \end{bmatrix}, \lambda = \begin{bmatrix} 10.7 & 0 \\ 0 & 0.4 \end{bmatrix}$。试用主成分分析 PCA 将样本 $x = [3,1]^T$ 变换至一维。

注意：在计算时，样本数据应减去均值；特征向量应当归一化。

- SVD分解

$$X = U\Sigma V^h \rightarrow XX^h = U\Sigma V^h V\Sigma U^h = U\tilde{\Sigma}U^h$$

4.2 线性判别分析LDA

- 步骤：
  - 类内散度矩阵：$S_w = \sum_{i=1}^{C}\{\frac{n_i}{N}\cdot\frac{1}{n_i}\sum_{k=1}^{n_i}(x_k^i-\mu_i)(x_k^i-\mu_i)^T\}$
  - 类间散度矩阵：$S_b = \sum_{i=1}^{C}\{\frac{n_i}{N}(\mu_i-\mu)(\mu_i-\mu)^T\}$
  - 对$S_w^{-1}S_b$做广义特征值分解，选取投影方向w
- 对于两类样本的线性判别分析，投影方向为$w = S_w^{-1}(\mu_1-\mu_2)$
  - $S_b = \lambda(\mu_1-\mu_2)(\mu_1-\mu_2)^T$($\lambda$为常数)
  - $S_w^{-1}S_b w = S_w^{-1}\lambda(\mu_1-\mu_2)(\mu_1-\mu_2)^T S_w^{-1}(\mu_1-\mu_2) = [\lambda(\mu_1-\mu_2)^T S_w^{-1}(\mu_1-\mu_2)]S_w^{-1}(\mu_1-\mu_2)$
- 数据集总散度：$S_t = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T = S_w + S_b$

5.1 贝叶斯决策

➢Bayes公式：

先验概率

后验概率 $p(\omega_i \mid x)=\dfrac{p(\omega_i)p(x \mid \omega_i)}{p(x)}$ 类条件概率密度函数

样本出现概率

◆ p(x)可采用全概率公式计算。

例: 假设某种疾病检测技术，确诊病人检测为阳性的概率为80%，健康人检测为阳性的概率为5%。若在某地区，参与检测人群中只有0.01%是确诊病人，若某人参加检测且结果为阳性，这个人是确诊病人的概率是多少？

解:假设某人确诊为事件A（健康为事件$\bar{A}$），检测阳性为事件B

则待求结果为$P(A|B)$。

根据贝叶斯公式，$P(A|B)=\dfrac{P(A)P(B|A)}{P(B)}$，其中$P(A)=0.01\%, P(B|A)=80\%$

而$P(B)=P(B|A)P(A)+P(B|\bar{A})P(\bar{A})=80\%\times0.01\%+5\%\times(1-0.01\%)$

◆ 贝叶斯模型学习类条件概率密度$p(x|w_i)$，因此是生成式模型

5.1 贝叶斯决策

- 贝叶斯决策：已知先验概率$P(\omega_i)$和类条件概率密度函数$P(x|\omega_i)$，计算后验概率$P(w_i|x)$对未知样本分类。选择后验概率最大的类别，可实现最小错误率的判决：
  - $\omega(x) = \arg\max p(\omega_i|x) = \arg\max p(x|\omega_i)p(\omega_i) = \arg\max g_i(x)$
- 判别函数：
  - 若$g_i(x) > g_j(x)$对一切$j \neq i$成立，则将x归入ω_i类
- 两类问题决策面方程：
  - 判别函数定义为$g(x) = g_1(x) - g_2(x)$，若$g(x) > 0$判别为w_1
  - 决策面方程为$g(x) = 0$

5.2 正态分布下的贝叶斯决策

- 正态分布：$$p(\boldsymbol{x}\mid\omega_i)=\frac{1}{(2\pi)^{\frac{d}{2}}\left|\Sigma_i\right|^{\frac{1}{2}}}\exp\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\}$$
- 判别函数定义为：$$G_i(x)=p(\boldsymbol{x}\mid\omega_i)p(\omega_i)=\frac{1}{(2\pi)^{\frac{d}{2}}\left|\Sigma_i\right|^{\frac{1}{2}}}\exp\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\}\cdot p(\omega_i)$$
- 取对数后判别函数为：$$g_i(x)=-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln\left|\Sigma_i\right|+\ln p(\omega_i)$$
  - (计算题2.3)分类界面$g(x)=g_1(x)-g_2(x)$, $g(x)>0$为第一类,$g(x)<0$为第二类
- 协方差矩阵的三种情况：
  - （1）$\Sigma_i=\sigma^2 I$ 最小欧氏距离分类器
  - （2）$\Sigma_i=\Sigma$ 最小马氏距离分类器
  - （3）$\Sigma_i\neq\Sigma_j,\ i\neq j$ 二次判别函数

5.2 正态分布下的贝叶斯决策

- 例：正态分布下有两类先验概率相等的样本集

$$D=\{(\boldsymbol{x}_n,y_n)\}_{n=1}^{N}=\begin{Bmatrix}([0.5,1.5]^{\mathrm{T}},1),([1.5,1.5]^{\mathrm{T}},1),([0.5,0.5]^{\mathrm{T}},1),([1.5,0.5]^{\mathrm{T}},1),\\([1.5,0.5]^{\mathrm{T}},0),([2.5,0.5]^{\mathrm{T}},0),([1.5,-0.5]^{\mathrm{T}},0),([2.5,-0.5]^{\mathrm{T}},0)\end{Bmatrix}$$

满足$\Sigma_1=\Sigma_2=\Sigma$。

$$\mu_1=\frac{1}{4}([0.5,1.5]^T+[1.5,1.5]^T+[0.5,0.5]^T+[1.5,0.5]^T)=[1,1]^T$$

$$\mu_2=\frac{1}{4}([1.5,0.5]^T+[2.5,0.5]^T+[1.5,-0.5]^T+[2.5,-0.5]^T)=[2,0]^T$$

$$\Sigma=\frac{1}{8}\left[\left(\begin{bmatrix}0.5\\1.5\end{bmatrix}-\begin{bmatrix}1\\1\end{bmatrix}\right)\left(\begin{bmatrix}0.5\\1.5\end{bmatrix}-\begin{bmatrix}1\\1\end{bmatrix}\right)^T+\cdots\right]=\begin{bmatrix}0.25&0\\0&0.25\end{bmatrix}$$

$$g(x)=g_1(x)-g_2(x)=-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)+\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)-\frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|}+\ln\frac{p(w_1)}{p(w_2)}$$

$$=(\mu_1-\mu_2)^{\mathrm{T}}\Sigma^{-1}x-\frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1-\mu_2^T\Sigma^{-1}\mu_2)=-4x_1+4x_2+4$$

5.2 正态分布下的贝叶斯决策

对某一类样本估计均值和方差

$x_1, x_2, \ldots, x_n \sim N(\mu, \Sigma)$，其中$\mu, \Sigma$是待估计的参量

- 正态分布中均值和协方差矩阵的最大似然估计：

$$\hat{\mu} = \hat{\theta}_1 = \frac{1}{n}\sum_{k=1}^{n} x_k \qquad \hat{\Sigma} = \hat{\theta}_2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^T$$

通常用最大似然估计计算，但最大似然的协方差矩阵估计是有偏的

- 协方差矩阵的无偏估计：

$$\hat{\Sigma}_{unbiased} = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^T$$

知识拓展:有偏估计与无偏估计

- 有偏/无偏？
  - 若θ的估计$\hat{\theta}$满足$E(\hat{\theta})=\theta$，则称估计$\hat{\theta}$是无偏的；反之，则称$\hat{\theta}$是有偏的
- 为何最大似然估计得到的协方差矩阵是有偏的？

$$E(\hat{\Sigma})=E\left(\frac{1}{n}\sum_{k=1}^{n}(x_k-\hat{\mu})(x_k-\hat{\mu})^T\right)=\frac{1}{n}E\left(\sum_{k=1}^{n}\left(x_kx_k^T-x_k\hat{\mu}^T-\hat{\mu}x_k^T+\hat{\mu}\hat{\mu}^T\right)\right)$$

$$E(x_kx_k^T)=\Sigma+\mu\mu^T, E(x_kx_i^T)=\mu\mu^T, E(x_k\hat{\mu}^T)=E\left(\frac{1}{n}\sum_{i=1}^{n}x_k\,x_i^T\right)=\frac{1}{n}(\Sigma+n\mu\mu^T)$$

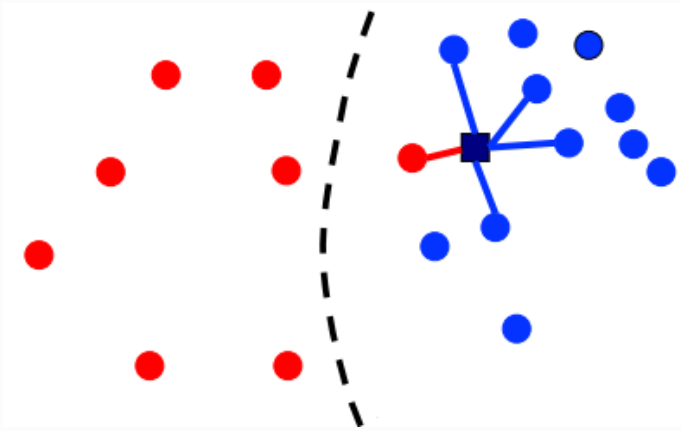
$$E(\hat{\mu}\hat{\mu}^T)=E\left(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}x_ix_j^T\right)=\frac{1}{\mathrm{n}^2}(\mathrm{n}\Sigma+n^2\mu\mu^T)=\frac{1}{\mathrm{n}}(\Sigma+n\mu\mu^T)$$

$$E(\hat{\Sigma})=\frac{1}{n}\left(E\left(\sum_{k=1}^{n}x_kx_k^T\right)-2E\left(\sum_{k=1}^{n}x_k\hat{\mu}^T\right)+E\left(\sum_{k=1}^{n}\hat{\mu}\hat{\mu}^T\right)\right)=\frac{1}{n}\left(n(\Sigma+\mu\mu^T)-2(\Sigma+n\mu\mu^T)+(\Sigma+n\mu\mu^T)\right)=\frac{n-1}{n}\Sigma$$

可以检验$E(\hat{\Sigma}_{unbiased})=\Sigma$

5.3 KNN、KMeans

➢K-Nearest-Neighbor (KNN, K-近邻)



- 利用k个邻近样本处理当前数据
- 思考：KNN可以处理回归问题吗？

➢KMeans

- 无监督聚类算法
- 聚类中心初始化会影响聚类结果（思考：初始化为相同的值可行吗？）
- KMeans++：随机选取一个样本点作为初始中心，再从剩下的样本中选取距离已有的初始中心最远的点作为新的初始中心
- KNN和Kmeans都涉及距离度量（还有哪些距离度量函数？）

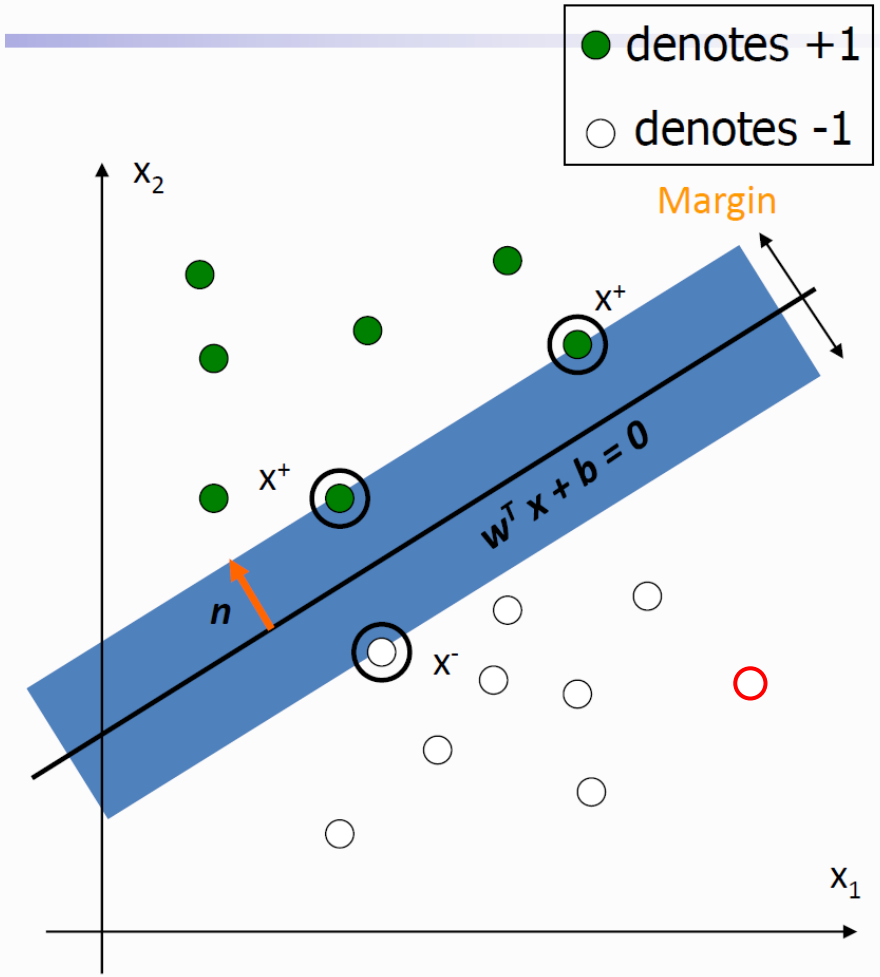
5.3 EM算法

➢期望-最大化算法（Expectation-Maximum）

- 一种迭代优化的参数估计算法，分为E步（通过已有变量的估计值，计算隐变量的估计值），M步（通过所有变量的估计值，最大化似然函数，求出已有变量新的估计值）
- 对于GMM模型，已有变量为高斯分布的均值和方差，隐变量为每个高斯分布的权重
- 对于KMeans算法，E步为计算当前聚类下的聚类中心，M步为计算新的聚类
- EM算法的收敛性有理论保证

6.1 线性支持向量机

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 \quad s.t.\, y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad (*)$$



➢如何理解支持向量？

- 直观理解：

  距离分类边界最近的样本点

- 数学理解：$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

  只有少数$\alpha_i \neq 0$的样本对分类边界有贡献

6.1 线性支持向量机

➢拉格朗日乘子法：

$$\text{定义 } L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

$$(*) \Leftrightarrow \underset{\mathbf{w},b}{\text{minimize}}\ \underset{\boldsymbol{\alpha}}{\text{maximize}}\ L(\mathbf{w}, b, \boldsymbol{\alpha}) \quad s.t.\ \alpha_i \geq 0, \forall i = 1, \cdots, n$$

➢对偶问题：满足KKT条件，可以交换内外层顺序

$$(*) \Leftrightarrow \underset{\boldsymbol{\alpha}}{\text{maximize}}\ \underset{\mathbf{w},b}{\text{minimize}}\ L(\mathbf{w}, b, \boldsymbol{\alpha}) \quad s.t.\ \alpha_i \geq 0, \forall i = 1, \cdots, n$$

什么是KKT条件？

6.1 线性支持向量机

➢KKT条件：对于凸优化问题，KKT条件可以保证，对偶问题的最优解就是原问题的最优解。

一般地，对于一个凸优化问题：

$$\min f(\mathbf{x})$$
$$s.t.\ g_i(\mathbf{x}) \geq 0, i = 1,2,\cdots,m$$
$$h_j(\mathbf{x}) = 0, j = 1,2,\cdots,l$$

可以定义拉格朗日函数：

$$L(\mathbf{x},\boldsymbol{\omega},\boldsymbol{\nu}) = f(\mathbf{x}) - \sum_{i=1}^{m} \omega_i\ g_i(\mathbf{x}) - \sum_{j=1}^{l} \nu_j\ h_j(\mathbf{x})$$

拉格朗日对偶函数：

$$\theta(\boldsymbol{\omega},\boldsymbol{\nu}) = \inf_{\mathbf{x}\in\boldsymbol{D}} L(\mathbf{x},\boldsymbol{\omega},\boldsymbol{\nu}),\ D\text{为集约束}$$

6.1 线性支持向量机

对偶问题：

$$\max_{\omega,\nu} \theta(\boldsymbol{\omega}, \mathbf{v})$$
$$s.t.\ \ \omega \geq 0$$

对偶问题的最优解是原问题最优解的充要条件是：

(1) $g_i(\mathbf{x}^*) \geq 0, i = 1,2,\cdots,m$

(2) $h_j(\mathbf{x}^*) = 0, j = 1,2,\cdots,l$

(3) $\omega_i^* \geq 0, i = 1,2,\cdots,m$

(4) $\omega_i^* g_i(\mathbf{x}^*) = 0, i = 1,2,\cdots,m$

(5) $\nabla_x L(\mathbf{x}, \boldsymbol{\omega}, \mathbf{v}) = \nabla f(\mathbf{x}^*) - \sum_{i=1}^{m} \omega_i^* \nabla g_i(\mathbf{x}^*) - \sum_{j=1}^{l} \nu_j^* \nabla h_j(\mathbf{x}^*) = 0$

6.1 线性支持向量机

对于支持向量机的优化问题，KKT条件第(3)、(4)、(5)条可以表示为：

$$\alpha_i \geq 0, i = 1,2,\cdots,n$$

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1) = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

着重观察第二个式子，可以看出，对于$\alpha_i > 0$的情况，必须有$y_i(\mathbf{w}^T\mathbf{x_i} + b) = 1$，即对于所有对分类界面有贡献的支持向量，有$y_i(\mathbf{w}^T\mathbf{x_i} + b) = 1$。

求解出 $\mathbf{w}$ 之后，可以任意选择一个支持向量，代入求解b的值。

6.1 支持向量机

将$\mathbf{w}=\sum_{i=1}^{n}\alpha_i\, y_i\mathbf{x_i}$，$\sum_{i=1}^{n}\alpha_i\, y_i$=0代入

$$L(\mathbf{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i+b)-1)$$

可得

$$L(\mathbf{w},b,\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\mathbf{x}_i^T\mathbf{x}_j=\theta(\boldsymbol{\alpha})$$

因此原问题变为求解

$$\max_{\boldsymbol{\alpha}\geq 0}\theta(\boldsymbol{\alpha}),s.t.\sum_{i=1}^{n}\alpha_i\, y_i=0$$

6.2 支持向量机与核方法

- 考虑线性不可分的情况：先对特征做非线性变换，$\mathbf{x} \to \phi(\mathbf{x})$，再对$\phi(\mathbf{x})$求解最大间隔分类器。

  引入核函数：$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$，记$\mathbf{K}$为核函数在训练数据上的矩阵形式，即$\mathbf{K}_{i,j} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$。

  几个例子：

$$\mathbf{K}_{i,j} = (1 + x_i^T x_j)^p; \quad \mathbf{K}_{i,j} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

  是否任意一个二元向量函数，都可以是核函数？

- Mercer定理：$K(\mathbf{x}_i, \mathbf{x}_j)$是有效核函数的充要条件是：**$K$为对称半正定矩阵**。

6.3 支持向量机与松弛变量

➢引入松弛变量ξ_i和正则化系数C：

$$minimize\ \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + C\sum_{i=1}^{n}\xi_i,\quad s.t.\ y_i(\boldsymbol{w}^T x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

$C \to \infty$时，$\xi_i \to 0$

$C \to 0$时，ξ_i可以很大

引入松弛变量后，不等式约束条件变为：

$$y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1 + \xi_i \geq 0, \xi_i \geq 0$$

相应的，KKT条件的第(4)条变为：

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1 + \xi_i) = 0$$

对于$\alpha_i > 0$的支持向量，有 $y_i(\mathbf{w}^T\mathbf{x_i} + b) = 1 - \xi_i \leq 1$。

引入松弛变量之后，对于支持向量，有$y_i(\mathbf{w}^T\mathbf{x_i} + b) \leq 1$。

谢谢大家！