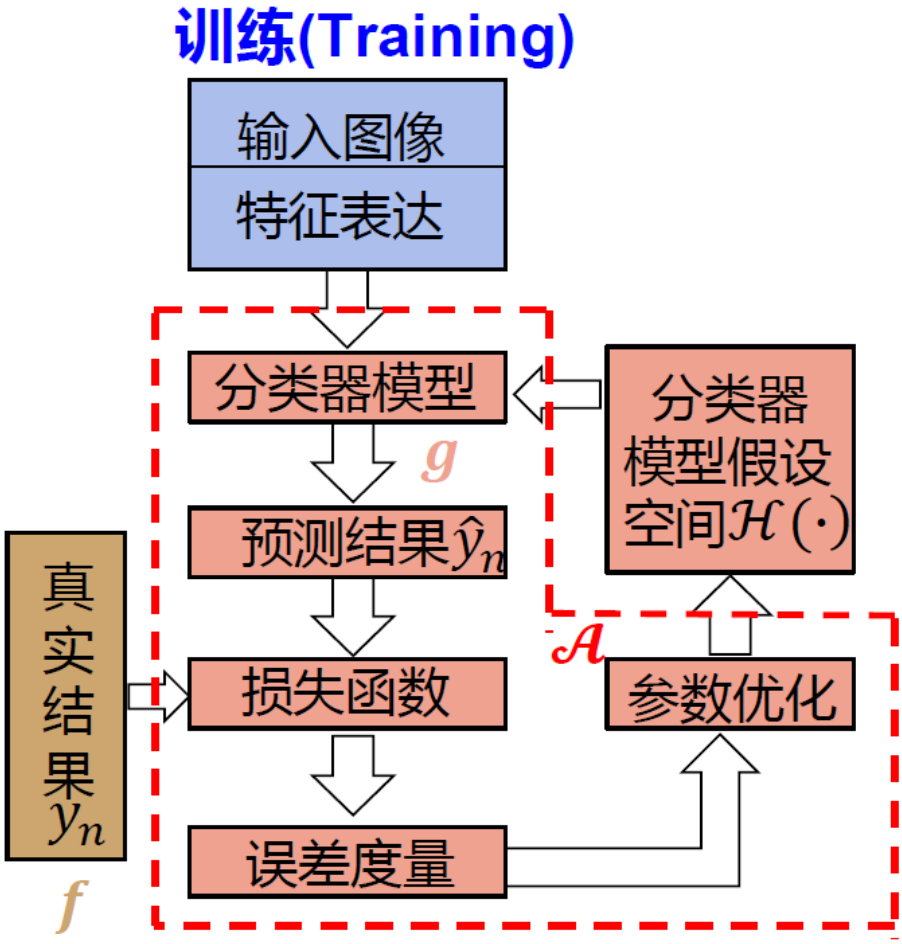
Lecture2：感知机PLA(Perceptron Learning Algorithm)

一句话概括：在高维空间中找出某个线性超平面，使得样本能被完全正确的二分类

由上面概括可大概判断PLA算法的一些特性：

1. 线性分类器
2. 解决二分类问题
3. 迭代至所有训练样本能够被完全正确分类才停止



从这幅流程图可看出，PLA算法需要具体讨论的点有：

1. 分类器模型具体是什么？
2. 如何定义损失函数（Loss）？
3. 如何寻找超平面，即如何参数优化$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t$？
4. 算法收敛性如何？

2.1 分类器模型

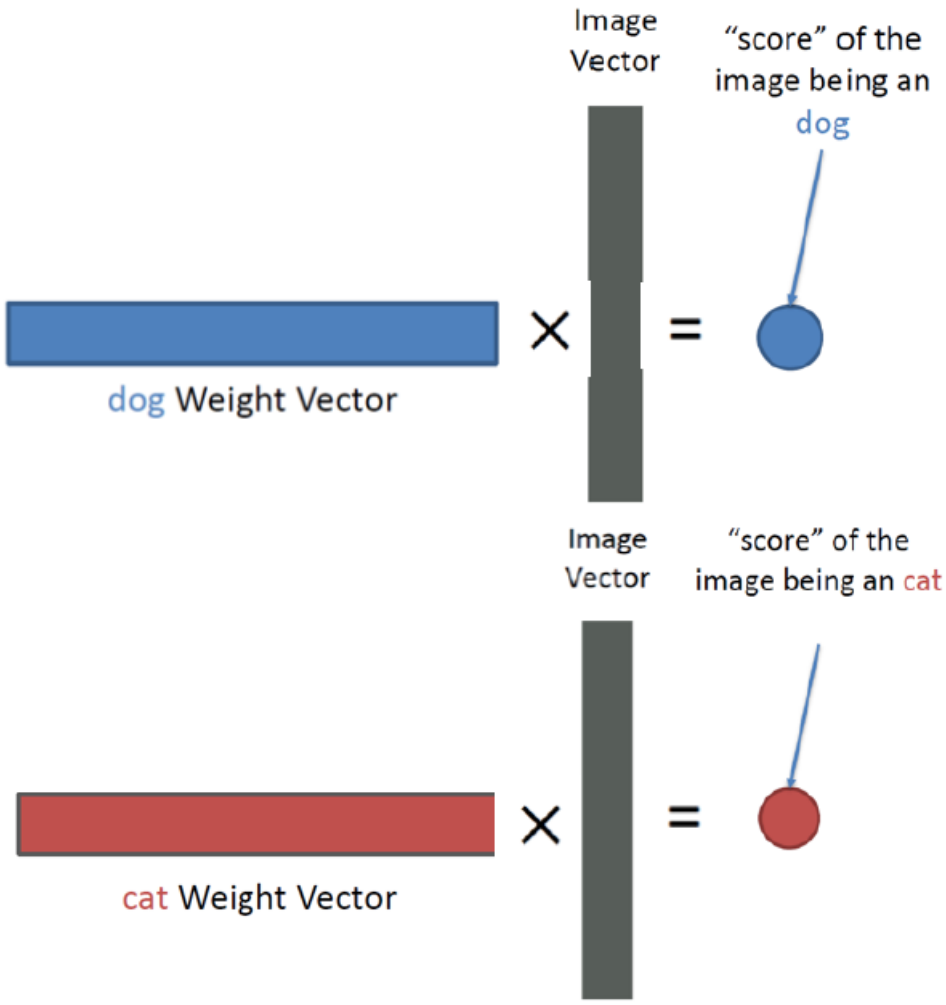
感知机模型本质上就是将**输入向量**$\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$**与一组权重系数**$\mathbf{w} = (w_1, w_2, \ldots, w_d)$**做内积，加上偏置**$b$**，判断其正负。**数学表示为：

$$\begin{aligned} \hat{y} &= sgn(\sum_{i=1}^{d} w_i x_i + b) \\ \hat{y} &= sgn(\mathbf{W}_d \mathbf{x}_d + b) \end{aligned} \tag{1}$$

默认$y=1$表示分类为正，$y=-1$表示分类为负。此时可将W和X做**增广化**，将常数项并入向量乘法中，使表达式更简洁：

$$\begin{aligned}\mathbf{W}_{d+1} &= (1, w_i, w_2, \ldots, w_d) \\ \mathbf{x}_{d+1} &= (b, x_1, x_2, \ldots x_d) \\ \hat{y} &= sgn(\mathbf{W}_{d+1}\mathbf{x}_{d+1})\end{aligned} \tag{2}$$

从模型数学公式可看出，PLA本身仅仅为权重与输入的线性运算，故其本身是一个**线性二分类模型**，只能解决比较简单的**线性二分类问题**。但可以将多个学习到的感知机模型重叠在一起形成多分类模型。



从数学本质上来看，模型将Weight Vector与Feature Vector作内积，而内积一定程度上可以刻画两个向量之间的**相似度**，所以可以认为Weight Vector就是模型所学习到的**模式**或者称为**特征**

2.2 损失函数(Loss)/参数优化

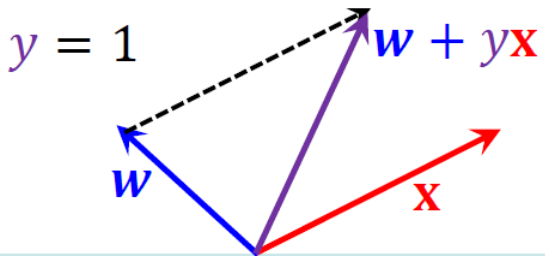
PLA损失函数非常直接清晰，即未被正确分类的训练样本的个数

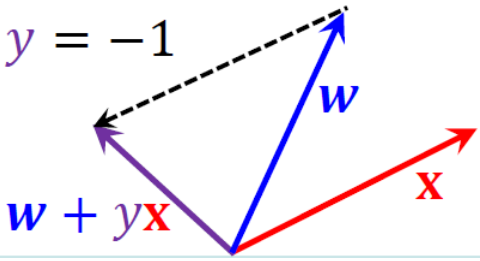
$$L = bool(y_n \neq \hat{y}_n) \tag{3}$$

优化过程也非常直接：

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y_n \mathbf{x}_{n(t)} \tag{4}$$

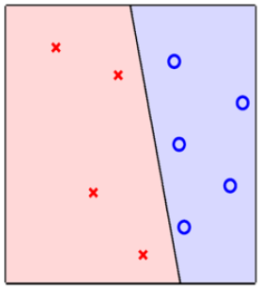
从内积的角度看，**内积越大表示两个向量之间夹角越小，从而反应两向量更加相似。**模型希望最终学习到的**权重与样本尽可能相似**，因为只有这样，其内积结果才能尽可能大（正数），最终通过符号函数变为1（划分到正样本中）。所以，当$y=1$，w会向更接近x的方向移动，**使模式与数据特征更匹配或相似**；反之，当$y=-1$，w会远离该负样本。



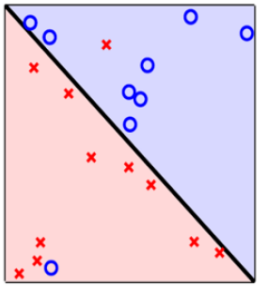


2.3 算法收敛性

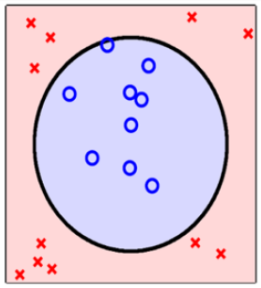
算法停止的条件为：所有样本均被正确分类。这自然就引出了一个问题：该模型在怎样的情况下会收敛？显然，对于非线性可分的数据，算法无论如何优化也不能通过一个线性超平面完全正确分类训练样本。那么，当训练样本为线性可分数据时，算法一定收敛吗？



线性可分



线性不可分



线性不可分 Ref.

可从数学上证明：

假设 $\boldsymbol{w}_f$ 是理想的分类面：

$\mathcal{D}$ 是线性可分的 $\Longleftrightarrow$ $y_n = sign(\boldsymbol{w}_f^T \mathbf{x}_n)$

↓ 第t次迭代时，任意一个样本$\mathbf{x}_{n(t)}$满足

$$y_{n(t)} \boldsymbol{w}_f^T \mathbf{x}_{n(t)} \geq \min_n y_n \boldsymbol{w}_f^T \mathbf{x}_n > 0$$

↓ 迭代次数增加

$$\boldsymbol{w}_f^T \boldsymbol{w}_{t+1} = \boldsymbol{w}_f^T \left(\boldsymbol{w}_t + y_{n(t)} \mathbf{x}_{n(t)} \right) \geq \boldsymbol{w}_f^T \boldsymbol{w}_t + \min_n y_n \boldsymbol{w}_f^T \mathbf{x}_n > \boldsymbol{w}_f^T \boldsymbol{w}_t + 0$$

结论：随着迭代次数增加， $\boldsymbol{w}_f^T \boldsymbol{w}_t$随之增加，意味着$\boldsymbol{w}_t$与 $\boldsymbol{w}_f$越来越接近

假设 $\boldsymbol{w}_t$ 是第 t 次迭代得到的分类面：

当 $\mathbf{x}_n$被分类错误时，才更新 $\boldsymbol{w}_t$ $\Longleftrightarrow$ $sign(\boldsymbol{w}_t^T \mathbf{x}_n) \neq y_n$ $\Longleftrightarrow$ $y_{n(t)} \boldsymbol{w}_t^T \mathbf{x}_{n(t)} \leq 0$

↓ 假设 $\mathbf{x}_n$在样本集中模值最大，随着迭代次数增加，$\|\boldsymbol{w}_t\|$?

$$\begin{aligned} \|\boldsymbol{w}_{t+1}\|^2 &= \left\| \boldsymbol{w}_t + y_{n(t)} \mathbf{x}_{n(t)} \right\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2 y_{n(t)} \boldsymbol{w}_t^T \mathbf{x}_{n(t)} + \left\| y_{n(t)} \mathbf{x}_{n(t)} \right\|^2 \\ &\leq \|\boldsymbol{w}_t\|^2 + 0 + \left\| y_{n(t)} \mathbf{x}_{n(t)} \right\|^2 \leq \|\boldsymbol{w}_t\|^2 + \max_n \|y_n \mathbf{x}_n\|^2 \end{aligned}$$

结论：随着迭代次数增加， $\boldsymbol{w}_t$模值增长不会太快， 意味着$\boldsymbol{w}_t$与 $\boldsymbol{w}_f$的接近是方向上在靠近，而非模值的贡献

2.4 线性不可分情况下的修正 -- Pocket算法

一句话概括：不要求训练样本被完全正确分类，只需按照与PLA相同的参数优化策略，找到最佳的w，

- 对样本的特征向量x和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)，并任意选一个“Pocket”向量$\hat{\boldsymbol{w}}$
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ (t 代表迭代次数)
  ① 进行到第t 次迭代时权向量为$\boldsymbol{w}_t$，它对样本$(\mathbf{x}_{n(t)}, y_{n(t)})$错分

$$\text{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}) \neq y_n$$

  ② 通过下式对权向量$\boldsymbol{w}_t$进行更新： $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$

  ③ 如果 $\boldsymbol{w}_{t+1}$在所有样本集上错分的样本少于$\hat{\boldsymbol{w}}$，则用$\boldsymbol{w}_{t+1}$代替$\hat{\boldsymbol{w}}$，并在错分样本中随机选一个对权向量进行更新

...达到指定的迭代次数

返回此时的 “Pocket”向量$\hat{\boldsymbol{w}}$ 作为算法学到的g

Ref.: NTU-LIN

Lecture3：线性回归

一句话概括：作为线性回归任务算法，损失函数定义为L2损失，模型输出的y不作任何处理

3.2 线性回归算法

优化目标

$$\min_{w} L_{in}(\mathbf{w}) = \frac{1}{N}||\mathbf{X}\mathbf{w} - \mathbf{Y}||^2 \tag{5}$$

由于L2损失函数是凸函数，所以$\mathbf{w}$取0的时候，损失函数值为0，于是优化该函数即求：

$$\nabla L_{in}(\mathbf{w}) = \mathbf{0}^T \tag{6}$$

可以通过矩阵求导进行推导，如此通过数学方法建立方程—解方程得到的$\mathbf{w}$被称为“**解析解**”

$$\begin{aligned}\nabla L_{in}(\mathbf{w}) &= \frac{L_{in}(\mathbf{w})}{\partial(\mathbf{X}\mathbf{w}-Y)} \cdot \frac{\mathbf{X}\mathbf{w}-Y}{\partial \mathbf{X}\mathbf{w}} \cdot \frac{\partial \mathbf{X}\mathbf{w}}{\partial \mathbf{w}} \\ &= \frac{2}{N}(\mathbf{X}\mathbf{w}-Y)^T \cdot \mathbf{1} \cdot \mathbf{X} \\ &= 0\end{aligned} \tag{7}$$

可解得

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y \tag{8}$$

在矩阵求导时，先明确链式法则中每一步的形状

3.3 梯度下降法

当样本量巨大时，使用解析解的方法计算$\mathbf{w}$较为困难，此时多采用迭代的方法求解。而梯度下降算法中最重要的便是学习率η，此处不加证明的给出“**当梯度较小时，我们希望有更大的学习率；当梯度较大时，我们希望有较小的学习率**”。故在实际应用中，学习率的调整也十分重要，以下分别介绍：

a. Adagrad

由上面内容可知，我们可以根据梯度来修正学习率，当$\mathbf{w}$是多维向量时，我们也希望在每个维度的学习率有所不同。

$$
\begin{aligned}
w_{i,t+1} &\leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}} \\
\sigma_{i,t} &= \sqrt{\frac{1}{t+1} \sum_{t=0}^{t} (\frac{\partial L_{in}}{\partial w_{i,t}})^2 + \epsilon}
\end{aligned}
\tag{9}
$$

使用该优化方法时，需要输入从开始迭代到目前为止的每一次梯度，ϵ为保证数值稳定性，加在分母上。

优点可以自适应的调整学习率；缺点，随着迭代次数的增加，自适应调整能力减弱，并且所有过去时刻对当前的σ影响相同，导致不能及时地对当前的梯度做出反应，**反应迟滞**

b. RMSProp

只需要把当前梯度和过往梯度进行加权，便可以使最近更新的梯度值有着更大的影响力

$$
\sigma_{i,t} = \sqrt{\alpha(\sigma_{i,t-1}) + (1-\alpha)(\frac{\partial L_{in}}{\partial w_{i,t}})^2}
\tag{10}
$$

c. Momentum

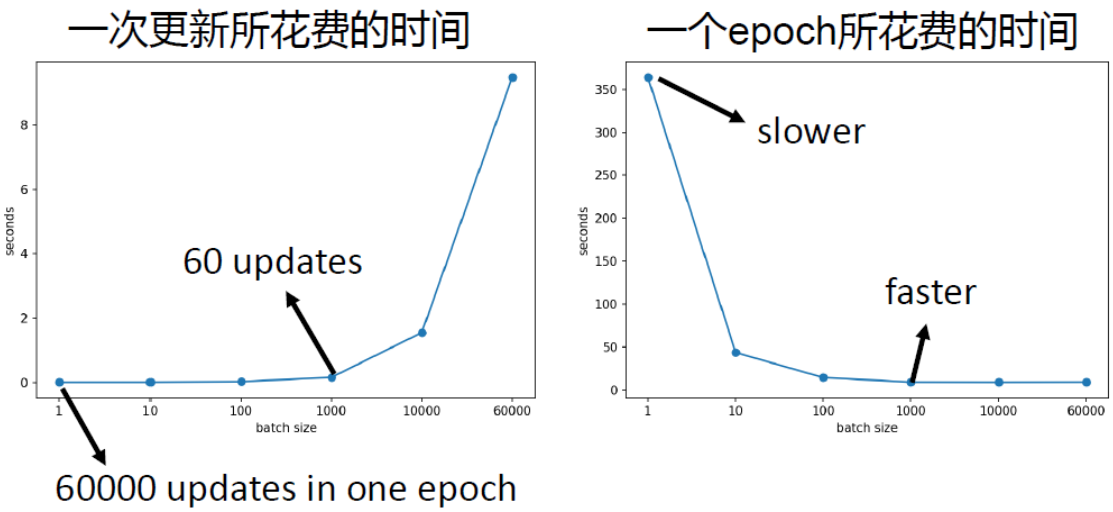
以上两种方法都会遇到一个问题，当梯度接近0时，学习率都十分低。但是，梯度接近于0并不代表找到了最优解，因为此时有可能是**局部最优(Local optimal)**或者处于**鞍点(Saddle point)**，此时，我们希望优化过程能具有一定的**动量(Momentum)**，在梯度为0的时候仍然可以继续更新跳出局部最优或者快速渡过鞍点。**所谓动量，就是前一步的更新向量**，将其乘上比例系数与本次迭代的更新向量相加

$$
m_t = \lambda m_{t-1} - \eta \nabla L_{in}(w)
\tag{11}
$$

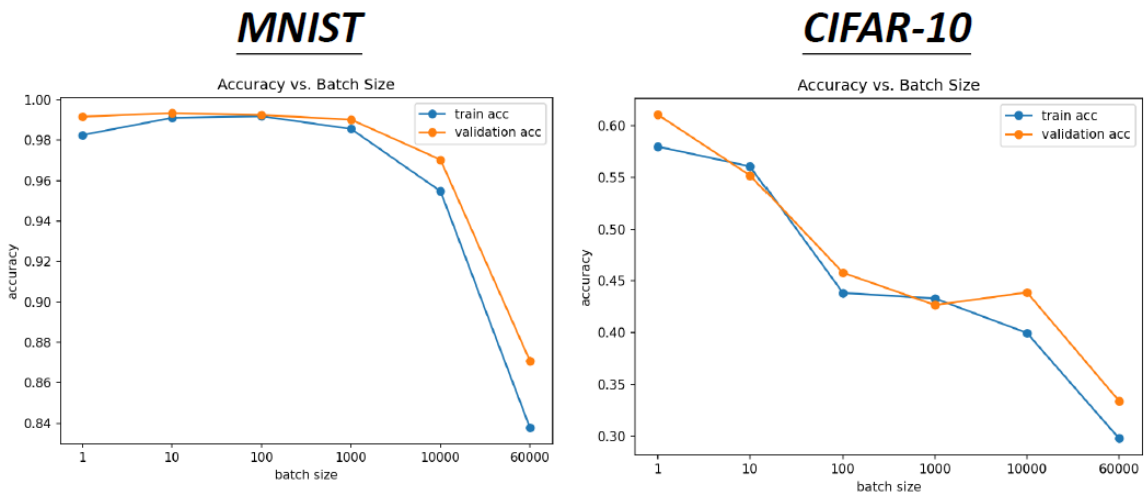
d. Adam: RMPProp + Momentum

$$
\begin{aligned}
m_t &= \lambda m_{t-1} - \nabla L_{in}(w_) \\
\sigma_{i,t} &= \sqrt{\alpha(\sigma_{i,t-1}) + (1-\alpha)(\frac{\partial L_{in}}{\partial w_{i,t}})^2} \\
w_{i,t+1} &\leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} m_t
\end{aligned}
\tag{12}
$$

除此之外，还需要考虑批量大小对梯度下降法的影响。计算梯度时，每次使用所有样本为**梯度下降法**，若每次只使用一个样本为**随机梯度下降法(SGD)**，若每次使用一个batch，叫做**批量梯度下降法**，其中，batch在每个epoch后会被reshuffle



实验证明，当使用batch时，一次更新花费的时间和一个epoch花费的时间都是比较小的。同时，batchsize对性能也有一定的影响



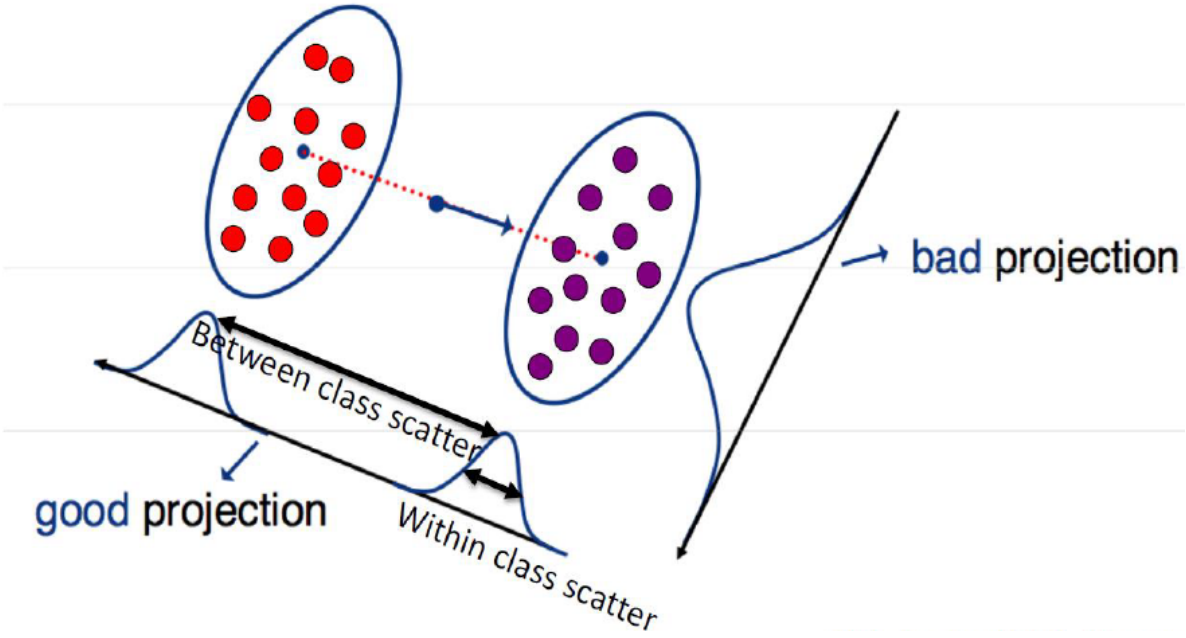
原因可以解释为：基于batch算出来的梯度具有一定的随机性，这样的随机性相当于在训练过程中添加了一定量的噪声，能够帮助优化过程快速**走出鞍点。**同时，由于batch数据分布仅仅为full batch的一个部分，在full batch上梯度较小时在batch上梯度可能较大，便于进一步优化

Lecture4：Fisher线性判别

一句话概括：找到某个投影超平面(super plane)，使得两个类别的特征在投影后，同类尽可能靠拢，异类尽可能分开

使用Fisher判别的需求是：在输入为高维向量时，直接对高维向量进行二分类问题需要很大的计算开销。于是我们希望找到一个超平面对输入数据进行降维，使得降维后我们依然能够较好的分类。

以下仅讨论二分类的Fisher线性判别



目标函数

$$
\begin{aligned}
J(w) &= \frac{between\ calss\ scatter}{within\ calss\ scatter} \\
w^* &= argmax_w j(w) \\
J(w) &= \frac{(E[s|y=1]-E[s|y=-1])^2}{var[s|y=1]+var[s|y=-1]}
\end{aligned} \tag{13}
$$

数学知识回顾

期望值

可以理解为服从某一概率分布的离散随机变量的加权平均值，其中权重为该变量出现概率的大小

协方差

用于衡量两个随机变量之间**联合变化程度**，计算公式为：

$$
cov(X,Y) = E((X-\mu)(Y-\gamma)) = E(X\cdot Y) - \mu\gamma \tag{14}
$$

当X与Y是统计独立时，二者之间的协方差为0，表示X的取值与Y的取值没有任何关系。而若X与Y不是统计独立，其协方差可能取非0值，表示X和Y之间存在某种**内在关联**。

协方差矩阵中

$$
\Sigma_{ij} = cov(X_i, X_j) = E[(X_i-\mu_i)(X_j-\mu_j)] \tag{15}
$$

即$X = [X_1, X_2, \ldots, X_n]^T$是由n个随机变量组成的向量

拉格朗日乘子法

在一组约束条件下最优化某一函数值，比如

$$
\begin{aligned}
&max f(\mathbf{x}) \text{ subjected to } g(\mathbf{x}) = 0 \\
&\max f(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})
\end{aligned} \tag{16}
$$

即引入新变量，将条件极值问题转化为新函数的自由极值问题

优化过程

根据方差和均值的定义可得

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})\,(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})^T\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{\Sigma}_1\boldsymbol{w}+\boldsymbol{w}^T\boldsymbol{\Sigma}_{-1}\boldsymbol{w}}$$

$$\boldsymbol{S}_B = (\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})\,(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})^T$$

$$\boldsymbol{S}_w = \boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_{-1}$$

最大化目标函数问题转化为：

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w} \quad Subject\ to\ \boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}=K$$

利用拉格朗日乘子法(*Lagrange multipliers*)：

$$L(\boldsymbol{w},\lambda) = \boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w}+\lambda(K-\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w})$$

$$= \boldsymbol{w}^T(\boldsymbol{S}_B-\lambda\boldsymbol{S}_w)\boldsymbol{w}+\lambda K$$

$$\nabla L_w(\boldsymbol{w},\lambda) = \frac{\partial L(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}} = \mathbf{0}^T$$

$$2(\boldsymbol{S}_B-\lambda\boldsymbol{S}_w)\boldsymbol{w}=\mathbf{0} \Rightarrow \boldsymbol{S}_B\boldsymbol{w}=\lambda\boldsymbol{S}_w\boldsymbol{w}$$

$$\boldsymbol{S}_B\boldsymbol{w}=\lambda\boldsymbol{S}_w\boldsymbol{w}$$

如果$\boldsymbol{S}_w^{-1}=(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_{-1})^{-1}$存在,则有：

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_B\boldsymbol{w}=\lambda\boldsymbol{w}$$

$$\boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})\,(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})^T\boldsymbol{w}=\lambda\boldsymbol{w}$$

$$\boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})\,a=\lambda\boldsymbol{w}$$

$$\boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1}) = \frac{\lambda}{a}\boldsymbol{w}$$

只关注投影向量的方向：

$$\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1})$$ Ref：CS131-Standford

即得到投影方向。

找到投影向量后，对任一测试样本 $\mathbf{x}$：

$$s = \boldsymbol{w}^{*T}\mathbf{x} = (\boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_{-1}))^T\mathbf{x}$$

假设类别的判别门限设为 s'：

$$s' = \frac{\boldsymbol{w}^{*T}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_{-1})}{2}$$

Lecture5：逻辑斯蒂回归(Logistic Regression)

5.1 逻辑斯蒂回归问题

在分类任务中，我们除了想知道分类的结果，还想知道将数据分为该类的置信度的大小。所以，逻辑斯蒂回归输出的是一个概率值，表示该样本为正样本的概率大小。

由于概率值取$[0,1]$，逻辑斯蒂函数可以将$\mathbf{w}^T\mathbf{x}+b$映射到$[0,1]$之间

5.2 逻辑斯蒂回归损失

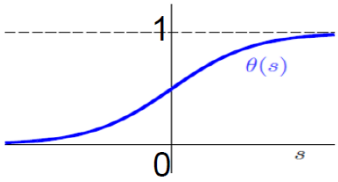
由于模型的输入为**概率值**，自然我们希望训练样本的标签值y也能是概率值，但实际上为1或-1，即标签为**含噪标签**。

逻辑斯蒂回归可以理解为**在给定样本下的极大似然估计**，其推导可参考此处。首先考虑能否使用EMS作为损失函数，即

$$L_{in}(w) = (\theta(w^T x) - y')^2 \tag{17}$$

$$L_{in}(\boldsymbol{w}) = (\theta(y\boldsymbol{w}^T\mathbf{x}) - 1)^2$$

$$\frac{\partial L_{in}(\boldsymbol{w},\mathbf{x},y)}{\partial w_i} = 2(\theta(y\boldsymbol{w}^T\mathbf{x}) - 1)\underbrace{\theta(y\boldsymbol{w}^T\mathbf{x})(1\text{-}\theta(y\boldsymbol{w}^T\mathbf{x}))}_{\frac{\partial\theta(z)}{\partial z}} y x_i$$



$$if\ \ (y\boldsymbol{w}^T\mathbf{x})) > 0 \quad \nabla L_{in}(\boldsymbol{w},\mathbf{x},y) = 0$$

$$if\ \ (y\boldsymbol{w}^T\mathbf{x})) < 0 \quad \nabla L_{in}(\boldsymbol{w},\mathbf{x},y) = 0$$

$\theta(yw^T x)(1-\theta(yw^T x))$项非常容易等于0，造成学习率极低。

接下来分析为什么选择交叉熵，其本质是对数的极大似然

$$L(w) = argmax_w \prod_{n=1}^{N}(y_n w^T x_n) \tag{18}$$

取对数，再加负号，将极大变为极小

$$L(w) = argmin_w \frac{1}{N}\sum_{n=1}^{N}\ln\left(1+\exp\left(-y_n w^T x_n\right)\right) \tag{19}$$

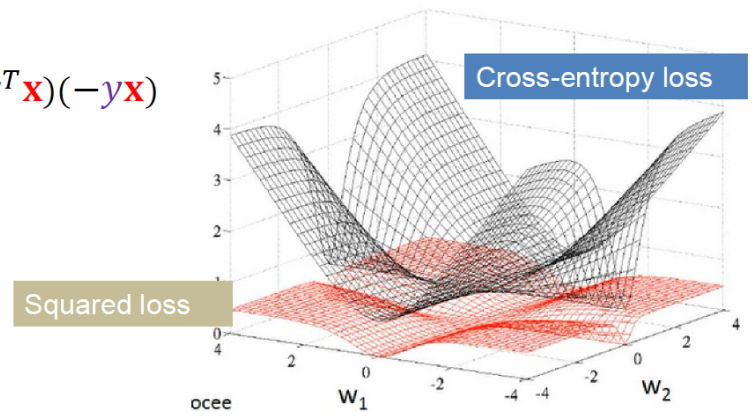
交叉熵用于概率分布的拟合

平方损失的梯度：

$$\nabla L_{in}(\boldsymbol{w},\mathbf{x},y) = 2(\theta(y\boldsymbol{w}^T\mathbf{x}) - 1)\theta(y\boldsymbol{w}^T\mathbf{x})(1\text{-}\theta(y\boldsymbol{w}^T\mathbf{x}))y\mathbf{x}$$

交叉熵损失的梯度：

$$\nabla L_{in}(\boldsymbol{w},\mathbf{x},y) = \theta(-y\boldsymbol{w}^T\mathbf{x})(-y\mathbf{x})$$



Lecture6：非线性变换(Nonlinear Transformation)

6.1 线性不可分问题

一句话概括：线性不可分的问题都可以通过一定的特征空间变换最终转化为线性可分的问题

首先考虑特殊情况，原本输入$\mathbf{x}$为一个二维特征$\mathbf{x}=[x_1,x_2]^T$，可以通过"**升维**"的方式升高到多维，即

$$\Phi_2(\mathbf{x})=(1,x_1,x_2,x_1^2,x_2^2,x_1x_2)^T \tag{20}$$

现在需要考虑的问题是：当输入向量维度为d，经过非线性变换到Q次多项式时，特征空间变为多少维度？

这个问题本质上可以看作是一个**"有放回且无序"**的抽样问题，在$(1,x_1,x_2,\dots,x_d)$中有放回地拿出Q个进行相乘，最终有多少种可能性。不加证明地给出

$$\widetilde{d}=C_{Q+d}^{Q} \tag{21}$$

证明以后再完善

6.2 非线性变换

非线性变换的根本目的在于：**将非线性可分问题转化为线性可分问题，使得线性分类算法能够work**

6.3 知识拓展

Lecture7：线性支撑向量机(Support Vecter Machine)

7.1 最大间隔分类面

在线性分类问题中，当训练样本均为无噪声输入时，我们希望模型能尽可能多的容忍测试样本中的噪声，即——获得一个更加鲁棒的分类面。用数学公式表述：

$$\begin{gathered}\max_{x}\operatorname{margin}(w)\\ \text{subject to every } y_n\mathbf{w}^T\mathbf{x}_n>0\\ \operatorname{margin}(w)=\min_{n=1,\dots,N}\operatorname{distance}(\mathbf{w},\mathbf{x}_n)\end{gathered} \tag{22}$$

7.2 标准的最大间隔问题

特征空间一向量$\mathbf{x}$距离超平面$\mathbf{w}$的距离可表述为：

$$|r|=\frac{\mathbf{w}^T\mathbf{x}+b}{||\mathbf{w}||} \tag{23}$$

即：

$$\operatorname{margin}(w)=\min_{n=1,\dots,N}\frac{y_n(\mathbf{w}^T\mathbf{x}_n+b)}{||\mathbf{w}||} \tag{24}$$

这一步比较难以理解：由于超平面的法向量$\mathbf{w}$和平移系数b可以比例缩放，不影响分类面本身以及分类的结果，所以可以令：

$$\min_{n=1,\dots,N}y_n(\mathbf{w}^T\mathbf{x}_n+b)=1 \tag{25}$$

所以问题转化为了：

$$\max\frac{1}{||\mathbf{w}||}$$

$$\begin{aligned} & \max_{x} \quad \frac{1}{||\mathbf{w}||} \\ & \text{subject to every } y_n \mathbf{w}^T \mathbf{x}_n > 0 \\ & \min_{n=1,\dots,N} y_n(\mathbf{w}^T \mathbf{x}_n + b) = 1 \end{aligned} \tag{26}$$

接下来进行条件松弛，这一步的目的是删去min，使得之后的数学推导更直观。即将上述问题中的约束条件转化为

$$y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \forall n \in N \tag{27}$$

以下用反证法说明条件松弛后不影响该问题的解，即在$y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$条件下解得的最优解$\mathbf{w}$和在条件$\min_{n=1,\dots,N} y_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$下解得的最优解相同

假设$\min_{n=1,\dots,N} y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1.3$可根据分 类面尺度不变性质得
$\min_{n=1,\dots,N} y_n(\frac{\mathbf{w}^T}{1.3} \mathbf{x}_n + \frac{b}{1.3}) \geq 1$

所以最终问题可以转化为：

$$\begin{aligned} & \min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} \\ & \text{subject to } y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \forall n \in N \end{aligned} \tag{28}$$

7.3 支撑向量机

该算法叫“支撑向量机”的原因是：超平面$\mathbf{w}$的取值仅由边界上的样本所决定。

损失函数为Hinge Loss:：$L_{SVM} = \max(0, 1 - ys)$

最终求解有两种方法：（1）梯度下降；（2）二次规划

首先考虑使用梯度下降求解SVM，损失函数定义为Hinge Loss

$$L_{SVM} = \max(0, 1 - ys) \tag{29}$$

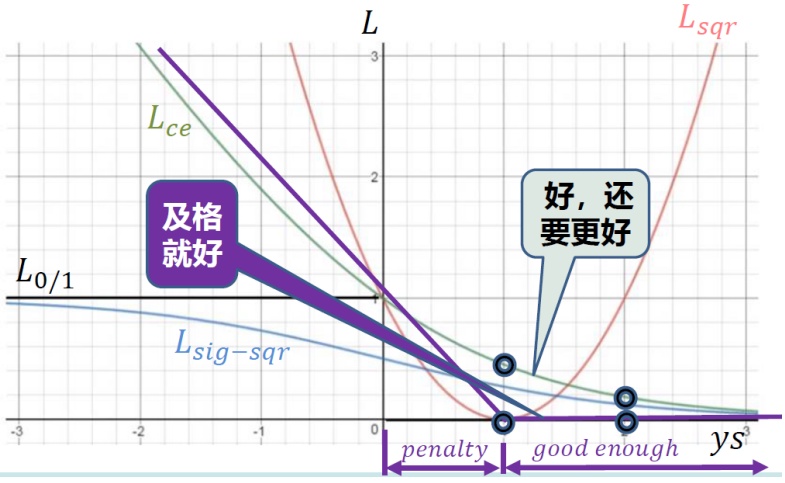
$L_{0/1} = [\![\hat{y} \neq y]\!]$

$L_{sqr} = (ys - 1)^2$

$L_{sig-sqr} = (\theta(ys) - 1)^2$

$L_{ce} = \ln(1 + \exp(-ys))$

$L_{SVM} = \max(0, 1 - ys)$



然后考虑二次规划求解

二次规划问题的特点是：（1）待优化的目标函数是凸函数（2）约束条件是线性函数

SVM的一般求解：

最佳的(w,b) =?

$$\min_{w} \quad \frac{1}{2}w^Tw$$
$$Subject\ to \quad y_n(w^Tx_n+b)\geq 1$$
$$for\ n=1,2,\dots N$$

二次规划(QP)的求解：

最佳的$\mathbf{u}\leftarrow \mathrm{QP}(\mathbf{Q},\mathbf{p},\mathbf{A},\mathbf{c})$

$$\min_{\mathbf{u}} \quad \frac{1}{2}\mathbf{u}^T\mathbf{Q}\mathbf{u}+\mathbf{p}^T\mathbf{u}$$
$$Subject\ to \quad \mathbf{a}_m^T\mathbf{u}\geq c_m$$
$$for\ m=1,2,\dots M$$

通过调用二次规划(QP)的求解函数就能得到SVM的最优解

$$\mathbf{u}=\begin{bmatrix}b\\ \mathbf{w}\end{bmatrix},\quad \mathbf{Q}=\begin{bmatrix}0 & \mathbf{0}_d^T\\ \mathbf{0}_d & \mathbf{I}_d\end{bmatrix},\quad \mathbf{p}=\mathbf{0}_{d+1}$$

$$\mathbf{a}_n^T=y_n[1\quad \mathbf{x}_n^T],\quad c_n=1,\quad M=N$$

此时只需要调用二次规划求解函数，并且将对应的参数匹配即可

Lecture8：对偶支撑向量机与核支撑向量机(Dual SVM & Kernel SVM)

8.1 对偶支撑向量机动机

SVM只能用于解决线性问题，当遇到非线性问题时，需要进行非线性变换，导致维数较高。于是我们希望SVM的求解中（二次规划）可以不依赖于升维后的$\tilde{d}$

首先利用拉格朗日乘子将约束条件下的优化问题转化为无约束条件下的优化问题：

$$\min_{w} \quad \frac{1}{2}w^Tw$$
$$Subject\ to \quad y_n(w^Tz_n+b)\geq 1$$
$$for\ n=1,2,\dots N$$

用Lagrange乘子 α_n 构造Lagrange函数

$$\mathcal{L}(b,\mathbf{w},\boldsymbol{\alpha})=\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{n=1}^{N}\alpha_n(1-y_n(\mathbf{w}^T\mathbf{z}_n+b))$$

约束项隐含在max中

$$\mathrm{SVM}\equiv \min_{b,\mathbf{w}}\left(\max_{all\ \alpha_n\geq 0}\mathcal{L}(b,\mathbf{w},\boldsymbol{\alpha})\right)=\min_{b,\mathbf{w}}(\infty\ if\ violating,\ \frac{1}{2}\mathbf{w}^T\mathbf{w}\ if\ feasible)=\frac{1}{2}\mathbf{w}^T\mathbf{w}$$

由于$1-y_n(w^Tz_n+b)\leq 0$，在$a_n\geq 0$时存在最大值，而Lagrange函数又存在最小值

8.2 对偶支撑向量机的拉格朗日分析

首先在所有可行的a_n中任选一组a'，可以得到

$$\min_{b,\mathbf{w}}\left(\max_{all\ \alpha_n\geq 0}\mathcal{L}(b,\mathbf{w},\boldsymbol{\alpha})\right)\geq \min_{b,\mathbf{w}}(\mathcal{L}(b,\mathbf{w},\alpha'))$$

如果此时的a'是使拉格朗日函数达到最优的那组值，则：

$$\min_{b,\mathbf{w}}\left(\max_{all\ \alpha_n\geq 0}\mathcal{L}(b,\mathbf{w},\boldsymbol{\alpha})\right)\geq \underbrace{\max_{all\ \alpha_n'\geq 0}\left(\min_{b,\mathbf{w}}(\mathcal{L}(b,\mathbf{w},\alpha'))\right)}_{Lagrange\ Dual\ Problem}$$

而二次规划问题又满足强对偶特性，即对等式两边最优化时，得到的最优(b,w,α)是相同的，那么原问题的最优化便可以转化为对偶问题的最优化。

于是我们的最优化问题现在变为了：

$$\max_{all\ \alpha_n \geq 0} \left(\min_{b,\boldsymbol{w}} \underbrace{\left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{n=1}^{N} \alpha_n (1 - y_n(\boldsymbol{w}^T \boldsymbol{z}_n + b)) \right)}_{\mathcal{L}(b,\boldsymbol{w},\boldsymbol{\alpha})} \right)$$

首先考虑括号内的min拉格朗日函数问题，分别对b和w求导，可以得到：

$$\frac{\partial \mathcal{L}(b,\boldsymbol{w},\boldsymbol{\alpha})}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\frac{\partial \mathcal{L}(b,\boldsymbol{w},\boldsymbol{\alpha})}{\partial w_i} = w_i - \sum_{n=1}^{N} \alpha_n y_n z_{n,i} = 0 \quad \Rightarrow \quad \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{z}_n$$

将得到的w和b带入得到最终的优化问题

“括号”内的问题$\boldsymbol{w}$取最佳解时：

$$\max_{all\ \alpha_n \geq 0,\ \sum_{n=1}^{N} \alpha_n y_n = 0,\ \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{z}_n} \left(-\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{z}_n \right\|^2 + \sum_{n=1}^{N} \alpha_n \right)$$

8.3 求解对偶支撑向量机最佳值

在8.2中最终得到的优化问题又可以进一步看作是新的二次规划问题，即：

标准的硬间隔SVM的对偶问题----求解最佳$\boldsymbol{\alpha}$：

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \boldsymbol{z}_n^T \boldsymbol{z}_m - \sum_{n=1}^{N} \alpha_n$$

$$Subject\ to \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\alpha_n \geq 0, for\ n = 1,2,\dots,N$$

- $\boldsymbol{\alpha}$的目标函数是二次函数、凸函数！N个变量
- $\boldsymbol{\alpha}$的约束条件是线性函数！$N+1$个约束条件

----二次规划(QP)问题！

QP有成熟方便的办法求优化解！

Ref.: NTU-LIN

在求解出a_n后，便可以进一步求解w和b，此时可通过w的表达式看出，$a_n > 0$所对应的样本决定w的取值，即这些样本就是支撑向量

8.4 对偶支撑向量机讨论

SVM的原问题求解：

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$Subject\ to \quad y_n(\boldsymbol{w}^T\boldsymbol{z}_n + b) \geq 1$$

$$for\ n = 1,2,\dots N$$

- $(\tilde{d} + 1)$ 个变量和N 个约束条件
- 求解最佳$(b, \boldsymbol{w})$

SVM的对偶问题求解：

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \boldsymbol{z}_n^T \boldsymbol{z}_m - \sum_{n=1}^{N} \alpha_n$$

$$Subject\ to \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\alpha_n \geq 0, for\ n = 1,2,\dots,N$$

- N 个变量和 $N+1$ 个约束条件
- 求解最佳$\boldsymbol{\alpha}$，确定支撑向量

两种方法都能得到最佳解$(b, \boldsymbol{w})$获得最大间隔分类面$g_{SVM} = \text{sign}(\boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x}_n) + b)$

由原问题和对偶问题的比较中可以看出，将原问题转化为对偶问题可以不依赖于非线性变换后的维数$\tilde{d}$

8.5 核函数支撑向量机

我们将原问题转化为对偶问题的原因是因为不想依赖$\tilde{d}$，但在实际求解二次规划问题时，Q是一个稠密矩阵，且每一个元素都需要计算z_n的内积：

$$\mathbf{Q}=\begin{bmatrix} y_1y_1\mathbf{z}_1^T\mathbf{z}_1 & \cdots & y_1y_N\mathbf{z}_1^T\mathbf{z}_N \\ \vdots & y_ny_m\mathbf{z}_n^T\mathbf{z}_m & \vdots \\ y_Ny_1\mathbf{z}_N^T\mathbf{z}_1 & \cdots & y_Ny_N\mathbf{z}_N^T\mathbf{z}_N \end{bmatrix}$$

核函数的基本想法是，提高$z_n^Tz_m=\Phi(x_n)^T\Phi(x_m)$的计算效率。

二次多项式$\boldsymbol{\Phi}_2(\mathbf{x})$的快速内积计算

$$\boldsymbol{\Phi}_2(\mathbf{x})=(1,x_1,x_2,\dots,x_d,x_1^2,x_1x_2,\dots,x_1x_d,x_2x_1,x_2^2,\dots,x_2x_d,x_dx_1,x_dx_2,\dots,x_d^2)$$

$$\begin{aligned}\boldsymbol{\Phi}_2^T(\mathbf{x})\boldsymbol{\Phi}_2(\mathbf{x}')&=1+\sum_{i=1}^{d}x_ix_i'+\sum_{i=1}^{d}\sum_{j=1}^{d}x_ix_jx_i'x_j'\\&=1+\sum_{i=1}^{d}x_ix_i'+\sum_{i=1}^{d}x_ix_i'\sum_{j=1}^{d}x_jx_j'\\&=1+\mathbf{x}^T\mathbf{x}'+(\mathbf{x}^T\mathbf{x}')(\mathbf{x}^T\mathbf{x}')\end{aligned}$$

计算复杂度从$O(\tilde{d})$下降到$O(d)$

所以可以定义核函数：

核函数：Φ变换+内积计算 ⟶ $K_{\boldsymbol{\Phi}}(\mathbf{x},\mathbf{x}')=\boldsymbol{\Phi}^T(\mathbf{x})\boldsymbol{\Phi}(\mathbf{x}')$

$$K_{\boldsymbol{\Phi}_2}(\mathbf{x},\mathbf{x}')=\boldsymbol{\Phi}_2^T(\mathbf{x})\boldsymbol{\Phi}_2(\mathbf{x}')=1+\mathbf{x}^T\mathbf{x}'+(\mathbf{x}^T\mathbf{x}')(\mathbf{x}^T\mathbf{x}')$$

此后的计算便可以借助核函数完成

利用二次规划(QP)实现核函数支撑向量机求解

① $q_{n,m}=y_ny_mK_{\boldsymbol{\Phi}}(\mathbf{x}_n,\mathbf{x}_m)$，$\mathbf{p}=-1_N$，由约束条件得到$(\mathbf{A},\mathbf{c},\mathbf{r},v)$

② $\alpha\leftarrow \mathrm{QP}(\mathbf{Q},\mathbf{p},\mathbf{A},\mathbf{c},\mathbf{r},v)$

③ 任选一支撑SV$(\mathbf{x}_m,y_m)$：$b\leftarrow(y_m-\sum_{\mathrm{SV}}\alpha_ny_nK_{\boldsymbol{\Phi}}(\mathbf{x}_n,\mathbf{x}_m))$

④ 对于测试样本$\mathbf{x}$：$g_{SVM}=sign(\sum_{\mathrm{SV}}\alpha_ny_nK_{\boldsymbol{\Phi}}(\mathbf{x}_n,\mathbf{x})+b)$

- 步骤①的时间复杂度：$O(N^2)\cdot(kernel\ evaluation)$
- 步骤②的开销　　：N个变量，$N+1$个约束
- 步骤③和④的复杂度：$O(\#SV)\cdot(kernel\ evaluation)$

所以，即使原始数据进行无穷维的变换，核函数依然能够解决问题，其本质是不依赖$\tilde{d}$