Statistical Learning and Data Science 个人总结

姓名：陈云　学号：3180105861 班级：测控1801

本次线上交流课程（加上第0次的见面课程）总共有6节（约10.5个小时），每次课程的主要内容我总结如下：

- √ 0：引入课程内容需要的线性代数和统计知识，介绍相关的参考资料。
- √ 1：引入课程内容，讲解线性回归模型的相关内容的概念和原理，以及数据成分分析的相关方法。
- √ 2：讲解模型的选择和评估、数据统计分析方法、线性回归的线性分类和逻辑回归的相关原理、KNN算法。
- √ 3：讲解非线性方法。
- √ 4：讲解决策树和支持向量机。
- √ 5：讲解神经网络、机器学习相关的算法，如前向传播、反向传播、梯度下降；以及一些优化方法如Regularisatioin防止过拟合，Normalization提高训练效率等。

下面我会对课程内容中我比较了解的一部分做一些补充总结。

Preliminaries

下面记录了常用的几个统计计算的公式

Expectation

$$E(x) = \sum_{i}^{n} x_i \cdot p_i$$

$$E(x) = \int_{-\infty}^{+\infty} x \cdot f(x) dx$$

Variance

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

$$var(x) = E(x - E(x))^2 = E(x^2) - E^2(x)$$

Covariance

$$cov(x, y) = E((x - E(x)) \cdot (y - E(y)))$$

Standard deviation

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}$$

Correlation coefficient

$$P_{x,y} = \frac{cov(x, y)}{\sqrt{var(x) \cdot var(y)}}$$

Moment

$$\text{Origin moment: } E(x^k)$$

$$\text{Central moment: } E(x - E(x))^k$$

Batch Normalization

- Input：

$$\{[x_1 \quad x_2 \quad x_3 \quad \dots x_n]\} \tag{1}$$

- Learn Parameters:

$$\begin{matrix} \beta \\ \gamma \end{matrix} \tag{2}$$

- Output:

$$y_i = BN_{\beta,\gamma}(x_i) \tag{3}$$

- Do what?

$$\begin{aligned} \mu_\beta &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \sigma_\beta^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_\beta)^2 \\ \hat{x}_i &= \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 - \epsilon}} \\ y_i &= \gamma\hat{x}_i + \beta \equiv BN_{\beta,\gamma}(x_i) \end{aligned} \tag{4}$$

Linear Regression

基本模型

假设输入为$X \in R^{m\times n}$，即m组数据，每组数据n个特征。而预测的输出(这里预测模型输出仅由w、b决定)：

$$\hat{y} = F(X) = f(X) + b = Xw + b \ \ (\hat{y}, w, b \in R^{m\times 1})$$

error为($y \in R^{m\times 1}$为真实输出)：

$$err = |\hat{y} - y|$$

这里假设当$b = 0$，可以很容易求出w(Normal Equation，证明见后面的梯度下降算法中)：

$$w = (X^\top X)^{-1} X^\top y$$

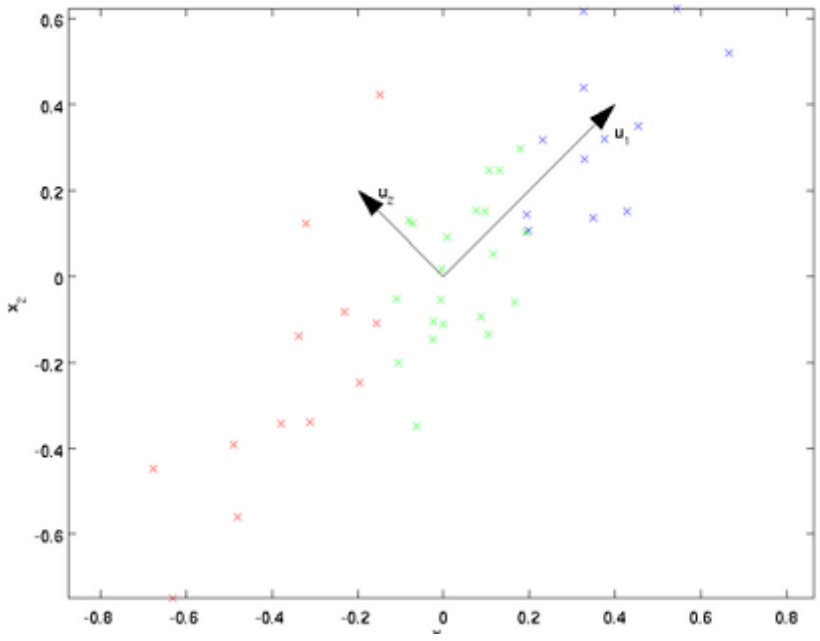
PCA降维

PCA（Principal components analysis 大概主成分分析）是我自己在做SRTP中也用过的数据降维方法，下面是我的结合以前的一些笔记做的一些总结。

PCA的目标是找出数据最主要方面代替原始数据。

设数据集为m个n维数据$[x_1, x_2, \cdots, x_m](dim(x_i, i \in [1, m]) = n)$，降维后的数据集的数据维度为$n'$。

设数据从2维降到1维，如下图（图片来自网络），要找某个维度方向（u_1优于u_2），u_1可以代表这两个维度的数据。



要得到原始数据新的表示空间，最简单的是对原始数据进行**线性变换（基变换）**

$$Y = PX$$

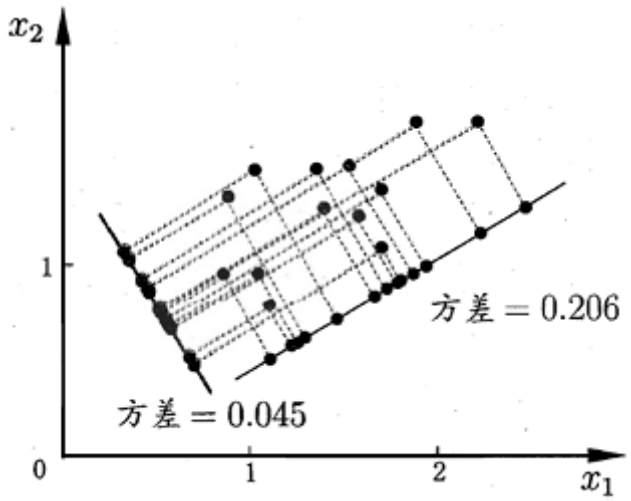
其中Y是样本在新空间的表达， P是基向量， X是原始样本。选择不同的基可以对一组数据给出不同的表示，同时**当基的数量少于原始样本本身的维数则可达到降维的效果**，矩阵表示如下：

(其中$p_i \in R^{1\times N}$是一个**行向量**，表示第 i 个基，$a_j \in R^{N\times 1}$是一个**列向量**，表示第 j 个原始数据记录。)

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{pmatrix} \begin{pmatrix} a_1 & a_2 & \cdots & a_M \end{pmatrix} = \begin{pmatrix} p_1a_1 & p_1a_2 & \cdots & p_1a_M \\ p_2a_1 & p_2a_2 & \cdots & p_2a_M \\ \vdots & \vdots & \ddots & \vdots \\ p_Ra_1 & p_Ra_2 & \cdots & p_Ra_M \end{pmatrix}$$

两个矩阵相乘的意义是将右边矩阵中的每一列向量变换到左边矩阵中以每一行行向量为基所表示的空间中去，即一个矩阵可以表示一种线性变换。,并且从原来的$X \in R^{N\times M}$变为了$Y \in R^{R\times M}$。

如何选择一个最优方向或者基? 我们将所有的点分别向两条直线做投影，如下图（图片来自网络），我们要找的方向是降维后损失最小，可以理解为**投影后的数据尽可能的分开，分散程度可以用数学上的方差来表示，方差越大数据越分散。**



方差计算：$var(x)=\frac{1}{m}\sum_{i=1}^{m}(x_i-\mu)^2$，中心化后（方便后续操作）：$var(x)=\frac{1}{m}\sum_{i=1}^{m}x_i^2$

对基变换后新的样本求其方差，**选取使其方差最大的基**。

从2维降到1维可用方差最大来选出能使基变换后数据分散最大的基（方向），但高维的变换，当完成第一个方基（方向）选择后，第二个投影方向会与第一个几乎重合在一起，因此要有其它的约束条件，表示更多的信息，使其不存在相关性。

数学上用协方差表示其相关性：$cov(a,b)=\frac{1}{m}\sum_{i=1}^{m}a_ib_i$，**当**$cov(a,b)=0$，**即a、b完全独立，即为目标。**

最终目标和**字段内方差及字段间协方差有密切关系**，假如只有a、b两个字段，那么我们将它们按行组成矩阵 X:

$$X=\begin{pmatrix}a_1 & a_2 & \cdots & a_m\\ b_1 & b_2 & \dots & b_m\end{pmatrix}$$

进一步计算可以得到协方差矩阵C:

$$C=\frac{1}{m}XX^{\top}=\begin{pmatrix}\frac{1}{m}\sum_{i=1}^{m}a_i^2 & \frac{1}{m}\sum_{i=1}^{m}a_ib_i\\ \frac{1}{m}\sum_{i=1}^{m}a_ib_i & \frac{1}{m}\sum_{i=1}^{m}b_i^2\end{pmatrix}$$

协方差矩阵是一个对称的矩阵，而且对角线是各个维度的方差，而其它元素是a、b的协方差。

优化目标是使$con(a,b)=0$，**这与协方差矩阵**C**对角化等价。即**C**的对角线外的元素为0，并且在对角线上将元素按大小从上到下排列。**

设原始数据矩阵X对应的协方差矩阵为C，P是一组基按行组成的矩阵，$Y=PX$，Y的协方差矩阵为D，则：

$$D=\frac{1}{m}YY^{\top}=\frac{1}{m}(PX)(PX)^{\top}=\frac{1}{m}PXX^{\top}P^{\top}=P\left(\frac{1}{m}XX^{\top}\right)P^{\top}=PCP^{\top}$$

明显要寻找P使原始协方差矩阵C对角化*($PCP^{\top}$为对角矩阵，并且对角元素按从大到小依次排列)*，用P的前K行为基组成的矩阵可以将初始数据集X从N维降到K维。

需要投影后的方差最大化，即优化目标为求得P使$PCP^{\top}$的迹最大：

$$\max_{P}\operatorname{tr}\left(PCP^{\top}\right),PP^{\top}=E$$

利用拉格朗日函数求极值，令：

$$L(P)=\operatorname{tr}\left(PCP^{\top}\right)+\lambda(PP^{\top}-E)$$

求导并令导数为0，有：

$$\begin{aligned}\frac{\partial L}{\partial P}&=CP^{\top}+\lambda P^{\top}=0\\CP^{\top}&=-\lambda P^{\top}\end{aligned}$$

对协方差矩阵C进行特征分解，对求得的特征值进行排序，取$P^{\top}=(P_1,P_2,\cdots,P_R)$前$K$列组成的矩阵乘以原始数据矩阵$X$，即得降维后的数据矩阵$Y$。

Model Assessment、Inferencing、Linear Classification

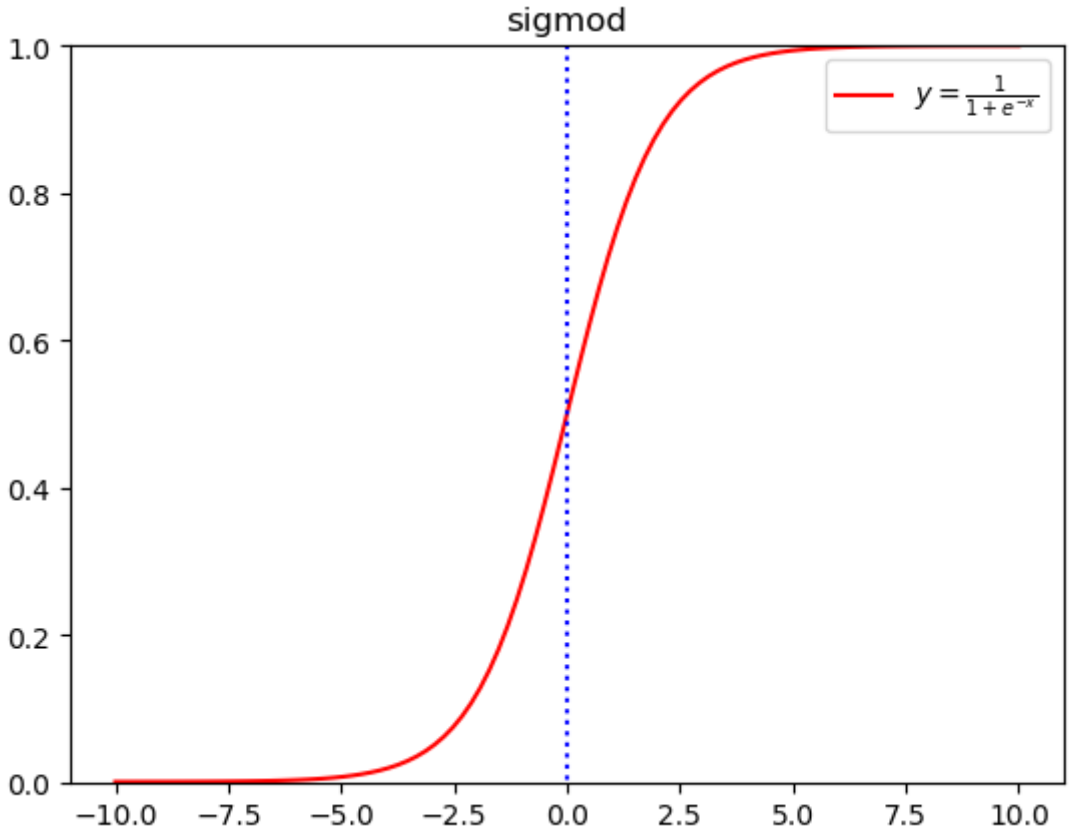
逻辑回归基本模型

逻辑回归就是将线性回归的输出向量的各个分量转为概率值，输出其中最大的概率，即逻辑回归进行分类的输出。

对于二分类任务，将线性回归输出$\hat{y}$通过$sigmoid$函数映射到$(0,1)$的概率范围内。

$$\sigma(x)=\frac{1}{1+e^{-x}}$$

$sigmoid$函数图像如下（利用python的matplotlib库绘制）：



对于多分类任务，将线性回归输出$\hat{y}$通过$softmax$函数（归一化指数函数）映射到$(0,1)$的概率范围内。

$$\mathbb{S}(x_i)=\frac{e^{x_i}}{\sum_{i=1}^{n}e^{x_i}}$$

容易证明（这里n为多分类任务的类数）：

$$\mathbb{S}(x_i) \in (0,1)\ and\ \sum_{i=1}^{n}\mathbb{S}(x_i) = 1$$

KNN

KNN（K-Nearest Neighbors，K邻近算法）采用测量不同特征值之间的距离进行分类。

这里的距离有多种不同的定义，下面用$dst(A,B)$表示A、B之间的距离，实际的距离可以用以下的方式计算：

$$dst(A,B) = \sum_i (a_i - b_i)^2 \ \ or \ \ \sum_i |a_i - b_i|_1 \ \ or \ \ \sum_i |a_i - b_i|_2 \cdots\cdots$$

设在N维空间中有点集S，然后这些点被分为n类，容易理解相同类的点之间的距离相比于和其他类的点来说很近。

设现有新点s，且已知其坐标，计算其与已与S中所有点的距离的集合DST，以DST最近的K个点中数量最多的类作为s的类别。

Non-Linear Methods

课程中的这方面的内容我目前了解与使用的比较少，需要结合课程录像做一定程度的加深。

我接触过的非线性方法用过的主要是分段将一些函数进行分段、拼接达到非线性。

比如现在用的最多的激活函数$RELU(x)=\begin{cases}x,x>0\\0,else\end{cases}$

包括各种各样的激活函数、损失函数（如Cross Entropy Loss（分类）、Total Variation loss（图像平滑）等）

Tree-Based Learning Methods、Support Vector Machines

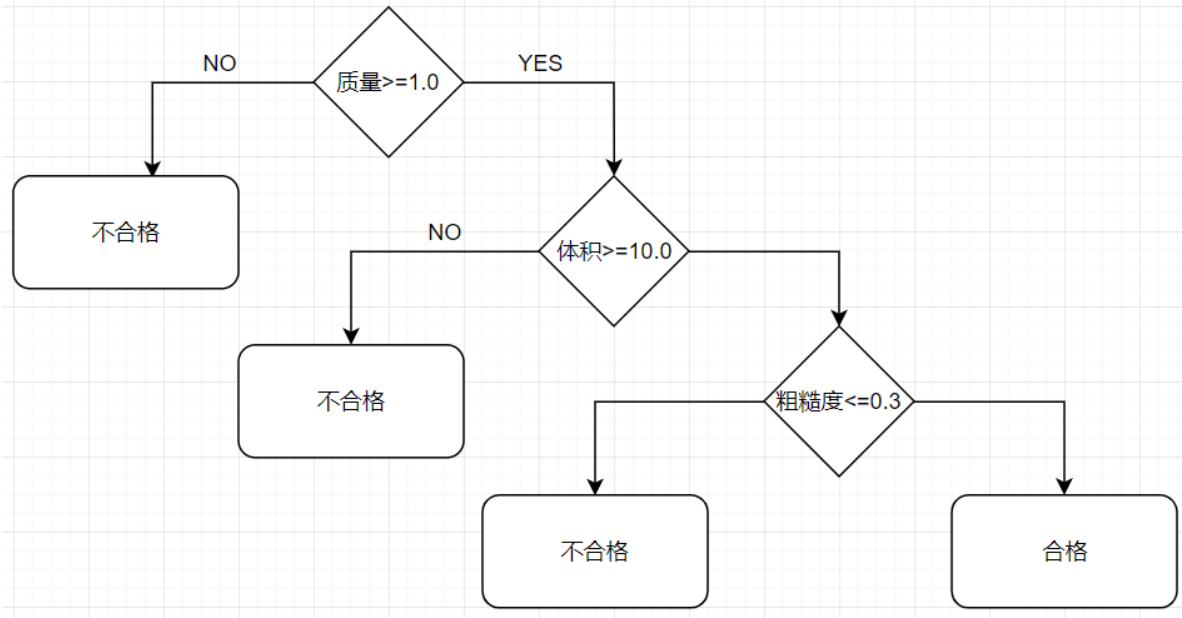
决策树

决策树就是根据对象的不同属性综合判断对象是否是属于某类，之所以叫决策树是因为如果用图来表示这一过程就是该图为树的形状。

下面用判断某种产品是否合格（当且仅当条件都满足才合格）来举例：

质量(>=1.0)	体积(>=10.0)	粗糙度(<=0.3)
1.1	10.3	0.2

决策树如下图（自己绘制）：



假设有n组产品已知合格与否，但不知道是否合格的判断标准（即上图中>=1.0、>=1.0、<=0.3未知的），可以通过训练得到合格的判断标准，即得到决策树。

决策树训练方式：

- 当某个节点所代表的属性无法判断时，将此节点分成n个子节点（n为类别）
- 确定判断标准（阈值），这里用基础的ID3算法说明。一批数据X的熵(Entropy)的定义：

$$\mathbb{EN}(\mathbb{X}) = -\sum P(x_i) log_2 P(x_i)$$

$P(x_i)$为x_i出现概率，熵表示体系的混乱程度，当$\mathbb{EN}(\mathbb{X})_{max} = 1$体系最混乱，分类效果最差，当$\mathbb{EN}(\mathbb{X})_{max} = 0$体系最有序，分类效果最好。这里要求得的判断标准（阈值）为向量w,即：

$$\underset{w}{argmin}\, \mathbb{EN}(\mathbb{X}|\mathrm{w})$$

支持向量机

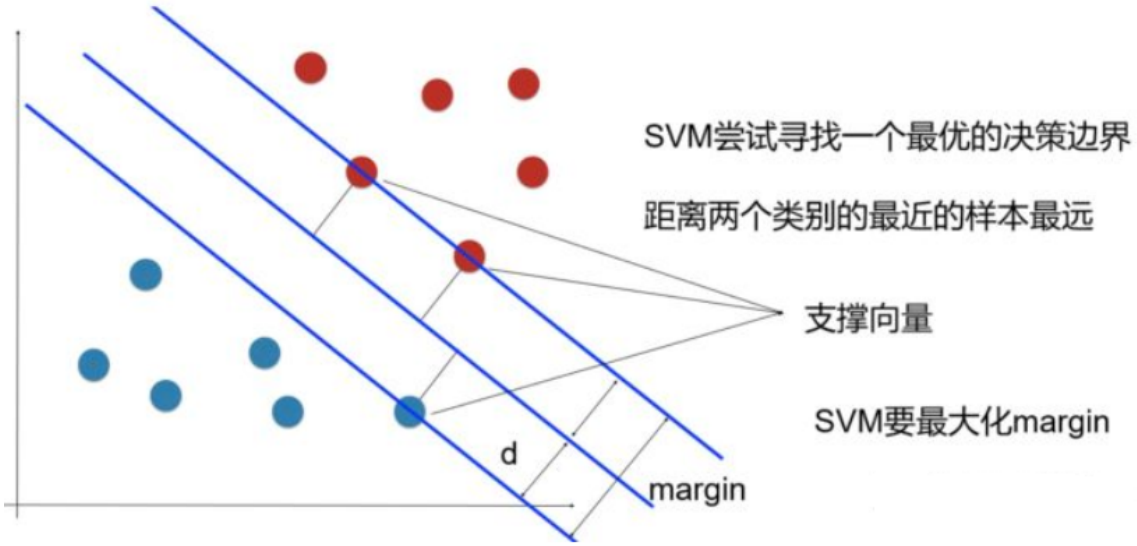
用于二分类

n维欧式空间上点集S_1、S_2，满足存在$w \in R^{n\times 1}$、$b \in R$，有（这里大小关系可以相反）：

$$x_{s_1} w + b > 0 \ (each \ x_{s_1} \in S_1) \wedge x_{s_2} w + b < 0 \ (each \ x_{s_2} \in S_2)$$

这里分开S_1、S_2为超平面$S: xw + b = 0$

支持向量为距离超平面最近的一些点（如下图，图片来自网络）



n维空间上点$x = (x_1, x_2, \cdots, x_n)$到超平面$xw + b = 0$的距离：

$$\text{dst} = \frac{|xw + b|}{\sqrt{\sum_{i=1}^{n} w_i^2}}$$

对于支持向量而言，其超平面dst为最小，即：

$$\frac{|xw + b|}{\sqrt{\sum_{i=1}^{n} w_i^2}} \geq \text{dst} \Longrightarrow \frac{|xw + b|}{\text{dst}\sqrt{\sum_{i=1}^{n} w_i^2}} \geq 1$$

注意到$\text{dst}\sqrt{\sum_{i=1}^{n} w_i^2} > 0$,为便于推导，令$\text{dst}\sqrt{\sum_{i=1}^{n} w_i^2} > 0 = 1$，即

$$xw+b\geq 1$$

记$y\in\{1,-1\}$为二元变量，表示两类，目标将两类分开，且使任意样本点到超平面距离满足：

$$(xw+b)y\geq 1\Longleftrightarrow\begin{cases}xw+b\geq 1,y=1\\xw+b\leq -1,y=-1\end{cases}$$

则最大间隔超平面$xw+b=1$上两个超平面$xw+b=\pm 1$

因为：$(xw+b)y\geq 1>0$，即$|xw+b|=(xw+b)y$，可以将dst分子绝对值替换，即：

$$\mathrm{dst}=\frac{(xw+b)y}{\sqrt{\sum_{i=1}^{n}w_i^2}}$$

目标为：

$$\underset{w}{argmax}(\mathrm{dst})=\underset{w}{argmax}=(\frac{1}{\sqrt{\sum_{i=1}^{n}w_i^2}}),(xw+b)y\geq 1$$

即：

$$min(\sqrt{\sum_{i=1}^{n}w_i^2})=min(\sum_{i=1}^{n}w_i^2),(xw+b)y\geq 1$$

Neural Networks

梯度下降算法

设输入$X \in R^{m \times n}$为m组数据，每组数据$x_i (i \in [1,m]) \in R^{1 \times n}$有$n$个特征：

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

权重为$w \in R^{n \times 1}$，预测输出$\hat{y} \in R^{m \times 1}$（这里为了简化，未考虑偏置参数$bias$）

$$\hat{y} = Xw = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} w = \begin{bmatrix} x_1 w \\ x_2 w \\ \vdots \\ x_m w \end{bmatrix}$$

实际输出值$y \in R^{m \times 1}$，均方误差为（$\frac{1}{2}$是为了后面约分）：

$$\mathbb{MSE} = \frac{1}{2}(Xw - y)^T (Xw - y) = \frac{1}{2}(w^T X^T Xw - w^T X^T y - y^T Xw + y^T y)$$

注意到$\mathbb{MSE} \in R^{1 \times 1} = \mathbb{SCALAR}$为标量，则：

$$\mathbb{MSE} = tr(\mathbb{MSE})$$

求梯度有：

$$\frac{\partial \mathbb{MSE}}{\partial w} = \frac{\partial tr(\mathbb{MSE})}{\partial w} = \frac{1}{2} \frac{\partial (tr(w^T X^T Xw - w^T X^T y - y^T Xw + y^T y))}{\partial w}$$

$$= \frac{1}{2}(\frac{\partial (tr(w^T X^T Xw))}{\partial w} - \frac{\partial (tr(w^T X^T y))}{\partial w} - \frac{\partial (tr(y^T Xw))}{\partial w} + \frac{\partial (tr(y^T y))}{\partial w})$$

由矩阵的迹的相关性质可以很容易计算出：

$$\frac{\partial(tr(w^TX^TXw))}{\partial w}=\frac{\partial(tr(w^T(X^TX)w))}{\partial w}=X^TXw+(X^TX)^Tw=2X^TXw$$

$$\frac{\partial(tr(y^TXw))}{\partial w}=\frac{\partial(tr(w^TX^Ty))}{\partial w}=X^Ty$$

$$\frac{\partial(tr(y^TXw))}{\partial w}=\frac{\partial(tr((y^TXw)^T))}{\partial w}=\frac{\partial(tr(w^TX^Ty))}{\partial w}=X^Ty$$

因为y、y^T与w无关，即：

$$\frac{\partial(tr(y^Ty))}{\partial w}=0$$

因此：

$$\frac{\partial\mathbb{MSE}}{\partial w}=\frac{1}{2}(2X^TXw-X^Ty-X^Ty)=X^T(Xw-y)$$

令$\frac{\partial\mathbb{MSE}}{\partial w}=0$，可以求解$w$（前面线性回归中提到的Normal Equation）

$$X^T(Xw-y)=0\Longrightarrow X^TXw=X^Ty\Longrightarrow w=(X^\top X)^{-1}X^\top y$$

用梯度下降的方式求解w，设学习率为$\mathbb{LR}$，迭代次数为$\mathbb{EPOCH}$，则：

> $\mathbb{INIT}:\mathbb{C}=1$
>
> $\mathbb{LOOP}(\mathbb{WHILE}:\mathbb{C}\leq\mathbb{EPOCH}):$
>
> $\quad\mathbb{C}=\mathbb{C}+1$
>
> $\quad w=w-\mathbb{LR}\cdot\frac{\partial\mathbb{MSE}}{\partial w}=w-\mathbb{LR}\cdot X^T(Xw-y)$

激活函数

上面的算法由于只涉及到矩阵的相关运算，因此最后的模型不具有线性成分。但神经网络显然不是线性的，因此需要添加一些非线性单元（函数），即激活函数。常见的激活函数有：

$$RELU(x) = \begin{cases} x, x > 0 \\ 0, else \end{cases} \in [0, \infty)$$

$$ELU(x) = \begin{cases} x, x > 0 \\ \alpha(e^x - 1), else \end{cases} \in (\alpha, \infty)$$

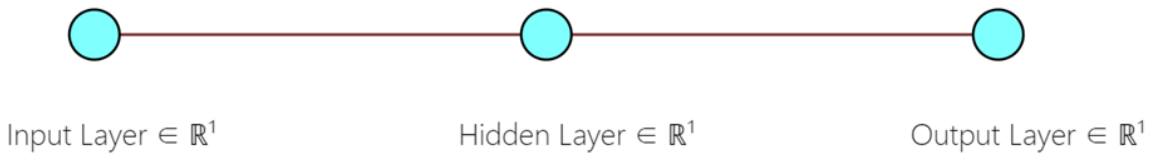
$$sigmoid(x) = \frac{1}{1 + e^{-x}} \in (0, 1)$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in (-1, 1)$$

反向传播

反向传播本质上就是函数的链式求导法则，只是在机器学习中的链式求导变成了对矩阵的求导。

为了便于理解与说明，这里仅仅以每层只有一个神经元为例说明（如下图，自己绘制）：



输入层Input Layer为x，$w^{(i)}, b^{(i)}$对应层的权重参数和偏置参数，这里f为激活函数，如$sigmoid$

输出层不经过激活函数输出为：

$$y^{(2)} = w^{(2)} l_{out}^{(1)} + b^{(2)}$$

输出层输出为：

$$l_{out}^{(2)} = f(y^{(2)})$$

隐藏层不经过激活函数输出为：

$$y^{(1)} = w^{(1)} x + b^{(1)}$$

输出层输出为：

$$l_{out}^{(1)} = f(y^{(1)})$$

损失函数为：

$$\text{loss} = (l_{out}^{(2)} - y)^2$$

求偏导：

$$\frac{\partial \text{loss}}{\partial l_{out}^{(2)}} = 2(l_{out}^{(2)} - y)$$

$$\frac{\partial l_{out}^{(2)}}{\partial y^{(2)}} = \frac{\partial f}{\partial y^{(2)}}$$

$$\frac{\partial y^{(2)}}{\partial w^{(2)}} = l_{\text{out}}^{(1)}$$

最终结果为：

$$\frac{\partial \text{loss}}{\partial w^{(2)}} = \frac{\partial \text{loss}}{\partial l_{out}^{(2)}} \frac{\partial l_{out}^{(2)}}{\partial y^{(2)}} \frac{\partial y^{(2)}}{\partial w^{(2)}} = 2(l_{out}^{(2)} - y) \frac{\partial f}{y^{(2)}} l_{out}^{(1)}$$

其它的梯度求法类似，这里不赘述。

我用过的优化措施

Regularization（正则化）

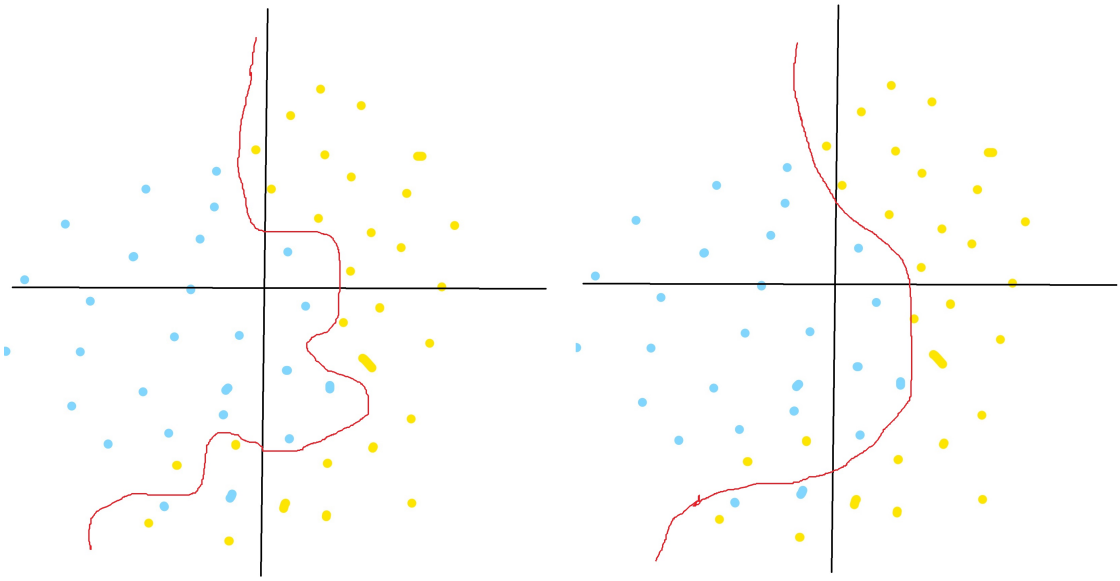
损失函数添加正则化项：

$$\mathbb{MSELOSS} = \frac{1}{2}(Xw-y)^T(Xw-y) + \lambda||w||_2$$

这里我通常在训练中取$\lambda = 100$（一个较大的值），$||w||_2 = w^Tw$

因为$||w||_2 > 0$，优化后$\mathbb{MSELOSS}$要尽可能小，由于λ值较大，因此w会尽可能小。

Regularization前（下图左，图像自己绘制）和Regularization后（下图右）：



明显经过正则化后的图像更平滑，梯度更小，因为参数w更小，这样的模型有更强的泛化能力。

学习率衰减

当学习率较大时，梯度下降时容易冲出最低点（局部最低点），导致梯度下降最终效果变差。

当学习率较小时，梯度下降时速度会较慢，导致达到相同结果时消耗时间变多，效率变低。

一个简单的解决办法是开始训练时设置学习率较大，随着迭代次数的增多逐渐衰减学习率，这样既能在一定程度上保证梯度下降的最终效果，又会有不错的效率。

数据Normalization

常见的方式是将数据利用公式将其转为标准的正太分布$N(0,1)$

$$\hat{x}_i = \frac{x_i - \overline{x}}{\hat{\sigma}}$$

其中，$\overline{x}$、$\hat{\sigma}$由样本估计：

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

提提一些其它的优化措施

除了普通的梯度下降法，我目前见过的有共轭梯度法、动量法、牛顿法，这些算法都是很好用的梯度下降方法，但原理都比普通的梯度下降法复杂一些，限于篇幅，这里不详说了。

除此之外，将数据增强也是比较好的措施。以CNN举例，很多时候很难搜集到足够的图片供我们训练，因此需要在已有的图片进行数据增强，比如将图片进行旋转、翻转、镜像、裁剪、提取、提取图像的某个通道等，这样可以将1张图片数据变成10张甚至更多的图片数据。

附录

关于公式的证明

某些证明在绝大多数情况下都成立（即训练的时候是基本不会出现问题的），但从数学的严谨性的角度来说，这些证明都存在着不完善的地方，如下所示：

- √ 未考虑两矩阵相乘是否满足矩阵相乘条件
- √ 未证明某矩阵的是否可逆
- √ 未证明某矩阵、函数是否在定义域内是否可导

自己用python实现的梯度下降算法

利用numpy库做相关的矩阵运算实现的算法代码网址如下：

https://github.com/morisa66/gradient_down