

暑期班总结报告

暑期学习内容总结

一、称手的环境和工具

- Python, Numpy和Pandas
 - 学习路上一门新的语言，Udacity上的课程将Python和两个数据分析工具快速地过了一遍。语言关在开始阶段是一个很大的障碍，Python里的集合，字符串，列表，元组，字典和各自的内置函数当时花费了大量的时间去做区分和熟悉。
 - Python For Data Analysis 和 NumPy 快速处理数据两本书在从0到1的阶段给予了很大帮助，第一周主要的时间也都放在了阅读入门教程上。两本书只挑了重点，没来得及读完，当然，对Python语言的学习也贯穿了整个暑假。
- Anaconda和Jupyter notebook
 - Anaconda没啥感觉，Jupyter notebook真心好用
- Keras
 - windows环境配置了好几次，安装失败，然后换到了虚拟机上，卷积的速度惨不忍睹。最后安装了双系统跑，顺带着重拾残存的linux知识。搭建环境耗费了不少时间，tensorflow的框架和函数式模型跑神经网络的确方便。

二、线性回归

- Linear Regression
 - 给定一个D属性描述的示例 $x = (x_1; x_2; \dots; x_d)$ ，其中 x_i 是 x 在第 i 个属性上的取值，linear model试图学得 $f(x_i) = wx_i + b$ ，使得 $f(x_i) \simeq y_i$ ，其中 loss function $L(f) = L(w^*, b^*) = \arg \min \sum_{i=1}^m (f(x_i) - y_i)^2$.

当然更一般的情形是用向量形式表示

- gradient descent

确定当前位置的损失函数的梯度 $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \frac{\partial L(\mathbf{b})}{\partial \mathbf{b}}$ ，用步长 α 乘以损失函数的梯度，得到当前位置下降的距离；将所有的 w 和 b 带入，若得出的梯度下降的距离都小于 ϵ ，算法终止，当前矩阵中所有的 w_i 和 b_i 即为最终结果。否则更新 w 和 b ，重复上述步骤。

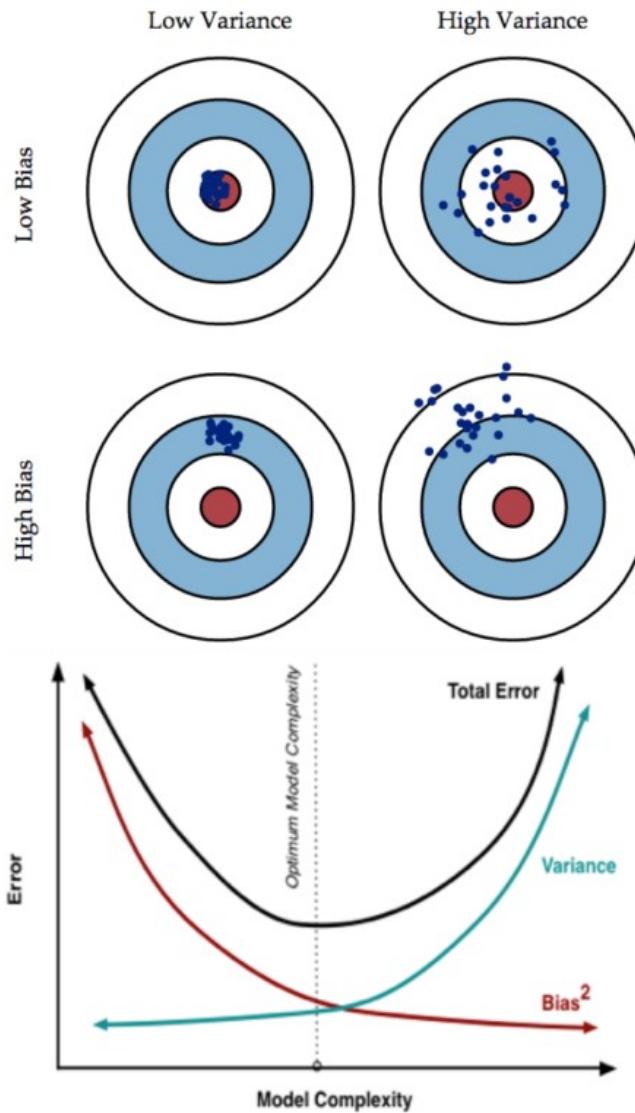
w 和 b 更新表达式如下：

$$(w_i^*, b_i^*) = (w_i, b_i) - \alpha \frac{1}{m} \sum_{j=0}^m (L_{w,b}(f) - y_j) x_i^j.$$

- bias, error and Variance

- 这个就不过多文字解释了，直接上图

准与确



bias描述真实与期望的差距

variance描述训练模型在测试集上的表现

error=bias+variance

- Learning rate α

◦ α 会随着梯度下降变得越来越小，故需要根据情况调整不同的学习率

批量梯度下降算法(BGD):在更新参数时使用所有的样本来进行更新,常用。

随机梯度下降算法(SGD):和批量梯度下降法原理类似，区别在与求梯度时没有用所有的m个样本的数据，而是仅仅选取一个样本j来求梯度

还有小批量梯度下降算法(MBGD) , Momentum和NAG动量法(第一次学到时感觉很神奇) , Adam , Adagrad , Adadelta , RMSprop , 牛顿法之类的 , 在后来搭建CNN时都很有用。

另外看概念时还get了一些看不太懂的 λ 强凸函数的纯数学知识

三、分类

- Logistic Regression

- linear Regression 类似于连续性的函数 , 而logistic Regression 则类似于概率。
- sigmoid : $g(z) = \frac{1}{1+e^{-z}}$, 取值在[0, 1]之间 , 在远离0的地方函数的值会很快接近0/1。
- 给定一个D属性描述的示例 $x[(x_1, y_1); (x_2, y_2); \dots (x_d, y_d)]$, 其中 x_i 是 x 在第 i 个属性上的取值, logistic model 试图学得 $P(y = 1|z) = g(z) = \frac{1}{1+e^{-z}}$, 其中 $z = \sum_i \mathbf{w}^T \mathbf{x} + b$.
- 最大似然估计 : $L(z) = P(D|z) = \prod P(y|x; z) = \prod g(z)^y (1 - g(z))^{1-y}$. 之后用gradient descent更新 w 和 b

w和b更新表达式如下 :

$$(w_i^*, b_i^*) = (w_i, b_i) - \alpha \frac{1}{m} \log [\sum_{j=0}^m (L_{w,b}(f) - y_j) x_i^j].$$

- softmax,是逻辑回归的多分类推广 , 也可称多元逻辑回归(Multi-logistic Regression)

$$P(y = i|x, \theta) = \frac{e^{\theta_i^T x}}{\sum_j^K e^{\theta_j^T x}}$$

而决策函数为 : $y^* = \operatorname{argmax}_i P(y = i|x, \theta)$

对应的损失函数为 :

$$J(\theta) = -\frac{1}{N} \sum_i^N \sum_j^K 1[y_i = j] \log \frac{e^{\theta_i^T x}}{\sum_k e^{\theta_k^T x}}$$

- K-Nearest-Neighbor -给定测试样本 x , 若其最近邻样本为 z , 则最近邻分类器出错的概率就是 x 与 z 类别标记不同的概率 , 即:

$$P(\text{err}) = 1 - \sum_{c \in y} P(c|\mathbf{x})P(c|\mathbf{z}).$$

令 $c^* = \arg \max_{c \in y} P(c|\mathbf{x})$ 表示贝叶斯最优分类器的结果 , 可以得到近邻分类器与贝叶斯最优分类器之间泛化错误率的比较。

$$\begin{aligned}
P(\text{err}) &= 1 - \sum_{c \in \mathcal{Y}} P(c \mid \mathbf{x}) P(c \mid \mathbf{z}) \\
&\simeq 1 - \sum_{c \in \mathcal{Y}} P^2(c \mid \mathbf{x}) \\
&\leq 1 - P^2(c^* \mid \mathbf{x}) \\
&= (1 + P(c^* \mid \mathbf{x})) (1 - P(c^* \mid \mathbf{x})) \\
&\leq 2 \times (1 - P(c^* \mid \mathbf{x}))
\end{aligned}$$

- Descision Tree

- 信息熵(information entropy):衡量样本集合纯度的一种指标

假设样本D中第k类样本的占比为 $p_k (k = 1, 2, \dots, |y|)$, 则信息熵定义为:

$$Ent(D) = - \sum_{k=1}^{|y|} p_k \log_2 p_k$$

- IC3:越是小型的决策树越优于大型的决策树,用信息增益来衡量:

$$Gain(D, a) = Ent(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} Ent(D^v)$$

- C4.5:使用信息率来选择最优划分属性:

$$Gain_{ratio}(D, a) = \frac{Gain(D, a)}{IV(a)}$$

其中 $IV(a) = -\sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$

- CART:使用"基尼系数"(Gini index)选择划分属性:

$$Gini(D) = \sum_{k=1}^{|y|} p_k p_{\bar{k}}$$

- 基本算法:

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
属性集 $A = \{a_1, a_2, \dots, a_d\}$.
过程: 函数 TreeGenerate(D, A)
1: 生成结点 node;
2: if D 中样本全属于同一类别 C then
3: 将 node 标记为 C 类叶结点; return
4: end if
5: if $A = \emptyset$ OR D 中样本在 A 上取值相同 then
6: 将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; return
7: end if
8: 从 A 中选择最优划分属性 a_* ;
9: for a_* 的每一个值 a_*^v do
10: 为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11: if D_v 为空 then
12: 将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; return
13: else
14: 以 TreeGenerate($D_v, A \setminus \{a_*\}$) 为分支结点
15: end if
16: end for
输出: 以 node 为根结点的一棵决策树

四、支持向量机(SVM)

- 给定训练样本集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, $y_i \in \{-1, +1\}$ 在训练集中找到一个划分超平面，将不同的样本区分开，SVM算法需要找到具有最大间隔(maximum margin)的划分超平面。

在样本空间中，划分超平面可以通过方程 $w^T x + b = 0$ 描述，其中 $w = (w_1; w_2; \dots; w_d)$ 为法向量，决定超平面的方向。

样本中任意点 x 到超平面 (w, b) 的距离为：

$$r = |w^T x + b| / \|w\|$$

则两个异类支持向量到超平面的距离为：

$$r = 2 / \|w\|$$

找到具有最大间隔的划分超平面，则要使得 r 最大，即：

$$\max_{w,b} 2 / \|w\|$$

$$s.t. y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, m$$

或者可写为：

$$\min_{w,b} 1/2 \|w\|^2$$

$$s.t. y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, m$$

- KKT 条件：非线性规划问题能有最优化解法的必要和充分条件

标准形式中的最小点 x^* 必须满足下面的条件：

- $h_j(x_*) = 0, j = 1, \dots, p, g_k(x_*) \leq 0, k = 1, \dots, q,$
 - $\nabla f(x_*) + \sum_{j=1}^p \lambda_j \nabla h_j(x_*) + \sum_{k=1}^q \mu_k \nabla g_k(x_*) = 0,$
- $$\lambda \neq 0, \mu_k \geq 0, \mu_k g_k(x_*) = 0$$

- SMO 算法求解对偶因子

$$\begin{aligned} \mathcal{L}(w, b, \alpha) &= \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

五、聚类分析

- Unsupervised Learning

- 聚类算法将样本集划分为 k 个不相交的簇，聚类的结果需要通过优化使簇内相似度 (intra-cluster similarity) 高，同时降低簇间相似度 (inter-cluster similarity)

- K-Means

给定样本集 $D = \{x_1; x_2; \dots; x_m\}$ ，k-means 对聚类所得簇划分 $C = \{C_1; C_2; \dots; C_k\}$ 的二次方误差：

$$E = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|_2^2,$$

其中 $\mu_i = 1/|C_i| \sum_{x \in C_i} x$ 是簇 C_i 的均值向量

输入：样本集 $D = \{x_1, x_2, \dots, x_m\}$;
聚类簇数 k .

过程：

```

1: 从  $D$  中随机选择  $k$  个样本作为初始均值向量  $\{\mu_1, \mu_2, \dots, \mu_k\}$   $\mu_1$  的均值
2: repeat
3:   令  $C_i = \emptyset$  ( $1 \leq i \leq k$ )
4:   for  $j = 1, 2, \dots, m$  do
5:     计算样本  $x_j$  与各均值向量  $\mu_i$  ( $1 \leq i \leq k$ ) 的距离:  $d_{ji} = \|x_j - \mu_i\|_2$ ;
6:     根据距离最近的均值向量确定  $x_j$  的簇标记:  $\lambda_j = \arg \min_{i \in \{1, 2, \dots, k\}} d_{ji}$ ;
7:     将样本  $x_j$  划入相应的簇:  $C_{\lambda_j} = C_{\lambda_j} \cup \{x_j\}$ ;
8:   end for
9:   for  $i = 1, 2, \dots, k$  do
10:    计算新均值向量:  $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ ;
11:    if  $\mu'_i \neq \mu_i$  then
12:      将当前均值向量  $\mu_i$  更新为  $\mu'_i$ 
13:    else
14:      保持当前均值向量不变
15:    end if
16:  end for
17: until 当前均值向量均未更新
输出：簇划分  $C = \{C_1, C_2, \dots, C_k\}$ 

```

- Mixture-of-Gaussian

对 n 维样本空间 χ 中随机向量 x ，若 x 服从高斯分布，其概率密度函数为：

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)},$$

然后可定义高斯混合分布：

$$pM(x) = \sum_{i=1}^k a_i \cdot p(x)$$

- 主成分分析(PCA)

假定数据样本进行了中心化，即 $\sum_i x_i = 0$ ；再假定投影变换后得到新的坐标系 $\{w_1; w_2; \dots; w_d\}$ ，其中 $\|w_i\|_2 = 1$, $w_i^T w_j = 0$ ($i \neq j$)。将样本维度降低到 $d < n$ ，则样本点在 x_i 的低维坐标系中的投影是 $z_i = (z_{i1}; z_{i2}; \dots; z_{id})$ ，其中 z_{ij} 是 x_i 在低维坐标系下第 j 维的坐标。

用 z_i 来重构 x_i ，即可得到原样本点 x_i 和重构样本点 \hat{x}_i 之间的距离：

$$\sum_{i=1}^m \left\| \sum_{j=1}^d z_{ij} w_j - x_i \right\|^2 = \sum_{i=1}^m z_i^T z_i - 2 \sum_{i=1}^m z_i^T W^T x_i + \text{const} - \text{tr}(W^T (\sum_{i=1}^m x_i x_i^T) W).$$

则主成分的优化目标：

$$\min_W - \text{tr}(W^T X X^T W)$$

$$s.t. W^T W = I$$

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

过程:

- 1: 对所有样本进行中心化: $\mathbf{x}_i \leftarrow \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$;
- 2: 计算样本的协方差矩阵 \mathbf{XX}^T ;
- 3: 对协方差矩阵 \mathbf{XX}^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 $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$.

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

六、神经网络

- DNN
- CNN
- RNN
- 来不及写公式了，详见附录手稿

七、其它

- 标准化(standardization), 归一化(normalization), 正则化(regularization)
 - 标准化：把数据变为 (0, 1) 之间的小数，方便数据处理；把有量纲表达式变换为无量纲表达式，成为纯量。
 - 归一化：将数据按比例缩放，使之落入一个特定区间
 - 正则化：在处理过程中引入正则化因子(regulator)，增加引导约束，最小化误差
 - L0范数是指向量中非0的元素的个数。如果我们用L0范数来规则化一个参数矩阵 \mathbf{W} 的话，就是希望 \mathbf{W} 的大部分元素都是0。换句话说，让参数 \mathbf{W} 是稀疏的。
- L1范数是指向量中各个元素绝对值之和，也称“稀疏规则算子”(Lasso regularization)。L1范数和L0范数可以实现稀疏，L1因具有比L0更好的优化求解特性而被广泛应用。
- L2范数是指向量各元素的平方和然后求平方根。L2范数可以防止过拟合，提升模型的泛化能力。

- 皮尔逊相关系数(Pearson Correlation Coefficient)

相关公式：

$$\rho_{x,y} = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y} = \frac{E((X - \mu_x)(Y - \mu_y))}{\sigma_x \sigma_y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)} \sqrt{E(Y^2) - E^2(Y)}}$$

$$\rho_{x,y} = \frac{N \sum XY - \sum X \sum Y}{\sqrt{N \sum X^2 - (\sum X)^2} \sqrt{N \sum Y^2 - (\sum Y)^2}}$$

$$\rho_{x,y} = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

- 各种距离计算

八、没来得及整理的知识

- HOG, LBP, Haar
- 剪枝，随机森林
- 核函数
- adaboost, XGBoost
- Neighbor Embedding
- 强化学习
- GAN

个人感想

有幸从同学那里得知暑期班的消息，一个暑假没有选择碌碌无为，回顾自己收集的资料，里面丰富的知识也足够让我继续学很长一阵子，当然如果中途没有志愿者之类任务打乱计划的话应该还能把知识点继续完善和总结一下。

第一周的语言关起步比较慢，基础知识的建立耗费至少4天时间，week1的任务有点赶。之后对语言的学习一直都断断续续穿插在每周的间隙。之后有时间还是需要把基本参考书看完，Python还是非常好用的，尤其是数据的抓取(暑期班用的都是现成的数据，个人觉得学会自己去网站上扒取数据还是很有意思的，而且可以做自己感兴趣的数据分析，当然爬虫还在抽空学习中)

决策树的几周是志愿者活动有些打乱计划，同时刚好学到中期，思路已经开始有些跟不上。所以分类问题是知识环节里最不熟练的一部分，直接导致来不及上传kaggle的比赛数据

对于知识点的学习更偏重于理论部分了，像SVM，PCA之类的耗费了很多时间在理清理论上，对于代码的练习不太够，刚开始对numpy和pandas的语法和公式运用也有些不熟练，直接导致前期代码的进度缓慢。

DNN和CNN部分花了很长时间看视频去理解，tensorflow的框架开始一直搭不上浪费了不少时间，不过最后还是看懂了(花了不少力气)，CNN也实现和优化了几遍(慢到怀疑电脑配置，结果也有些差强人意，不过好歹提交上kaggle了)到RNN部分已经有些看不懂了，可能还要花多一点的时间吧(开学初又有好多事处理耗去不少时间，主要是想回家了……)

电脑有时候直接晚上放实验室了这样第二天还有动力回去

还是离不开手写笔记，所以到最后收集的资料手写的比打字的还多

github和Linux重学了一遍，markdown和latex新学了一遍，感觉贼好用

书还是得多看，顺带着还得学会看论文，尤其是英文的表达

个人认为这一大块领域必定作为主导，至少也是今后重点发展的方向。大开眼界的一个暑假，也许这就是改变世界吧，是一种接触了就逃不开的命运

意见 建议

没啥意见吧，就是行程有点紧忙不过来，也有可能我学得慢。

其实最好能搬到本部去，东区一直都不大方便

规划

- 玩摄影，做视频，捣鼓摄像头 到大学开发了不少自己的兴趣，也学习了用PS，openCV，3Dsmax，Matlab(那时候很惊讶这东西能修图？)处理各种图像。所以在目标识别，图像分类和处理方面可以尝试更深入的学习
- 创新实践的课在玩硬件，所以在暑假其实一直想着怎么把电脑上跑的神经网络搬到硬件上去。tensorflow上有raspberry Pi的应用实例，也借鉴了国外一些电子dalao的开源代码，尝试在树莓派上搭建tensorflow框架，道路万分曲折。人脸检测已经能够实现了，打算让它自己学习目标识别(速度真的慢，识别一个东西要20-30s)

总结快写完的时候，iphone发布ai芯片了，再过几周huawei发布的mate10也要搭载kirin970智能芯片了，尝试去移植到移动端，也许又会是一场革命吧。

附录

- 18份手写稿：

Linear Regression, (R^2 vs)

$$\textcircled{1} \quad f(x) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b_{\text{bias}}$$

$$\rightarrow f(x) = [w]^T [x] + b.$$

② Loss function

$$L(w, b) = \sum_{i=1}^n (y_i - f(x_i))^2 \quad \text{if } L(w, b) = \sum_{i=1}^n (y_i - (w^T x_i + b))^2$$

(least square method)

③ pick best function

$$L(w^*, b^*) = \arg \min_{(w, b)} \sum_{i=1}^n (f(x_i) - y_i)^2 = \arg \min_{(w, b)} \sum_{i=1}^n (y_i - w^T x_i - b)^2$$

④ get w^*, b^* (Gradient Descent)

$$\frac{\partial L(w, b)}{\partial w} = 2 \left[\sum_{i=1}^n w x_i - \sum_{i=1}^n (y_i - b) x_i \right], \quad \frac{\partial L(w, b)}{\partial b} = 2 \left[\sum_{i=1}^n (y_i - w^T x_i - b) \right].$$

$$\text{令 } \frac{\partial L}{\partial w} = 0, \quad \frac{\partial L}{\partial b} = 0.$$

$$\boxed{w = \frac{\sum_{i=1}^n x_i (y_i - \bar{y})}{\sum_{i=1}^n x_i^2 - \frac{1}{n} (\sum x_i)^2}, \quad b = \frac{1}{n} \sum (y_i - w x_i)}$$

⑤ multivariate Linear Regression.

$$\hat{w}^* = \arg \min_{\hat{w}} \underbrace{([y] - [\hat{w} x])^T ([y] - [\hat{w} x])}_{\text{Error}}$$

$$\frac{\partial F_{\hat{w}}}{\partial \hat{w}} = 2 [x]^T ([\hat{w} x] - [y])$$

$$\hat{w}^* = (x^T x)^{-1} x^T y$$

$$\boxed{\hat{f}(x) = w^T x + b, \quad \hat{w}^* = (w^T x + b)^{-1} x^T y.}$$

⑥ 以线性模型: $y = g^{-1}(w^T x + b)$

$$\text{即 } g(y) = w^T x + b, \quad g(\cdot) \text{ 为单函数.}$$

$$P_{w, b}(c, | x) = \sigma(z).$$

$$\left\{ \begin{array}{l} z = \sum_i w_i x_i + b, \\ \sigma(z) = \frac{1}{1 + \exp(-z)}. \end{array} \right.$$

$$f_{w, b}(x) = P_{w, b}(c, | x) = \sigma(z).$$

$$\text{④ } f_{w, b}(x) = \frac{1}{1 + \exp(\sum_i w_i x_i + b)} \in \{0, 1\}$$

$$\text{TD: } \pi_1 \cdot x_1 \cdot \pi_2 \cdot \dots \cdot \pi_n \quad \text{based on } f_{w, b}(x)$$

$$\text{if } x_i \in C_1, \quad L(w, b) = f_{w, b}(x_1) f_{w, b}(x_2) \dots f_{w, b}(x_n)$$

$$\text{⑤ } \boxed{\begin{array}{l} \text{get } w^*, b^* \quad w^*, b^* = \arg \max_{w, b} L(w, b) \\ \text{or } w^*, b^* = \arg \min_{w, b} - \ln L(w, b) \end{array}}$$

$$\boxed{\text{Logistic Regression.}}$$

$$f_{w, b}(x) = \sigma(\sum_i w_i x_i + b)$$

$$\boxed{\text{Linear Regression.}}$$

$$f_{w, b}(x) = \sum_i w_i x_i + b$$

$$\text{Submission: } \hat{f}_1 = 1, \quad \hat{f}_2 = 0, \quad \hat{f}_3 = 1 \quad (\text{归一化处理})$$

$$\text{if } c_1: \quad \hat{f} = 1$$

$$\text{else: } \quad \hat{f} = 0$$

$$\downarrow$$

$$\begin{aligned} \rightarrow -\ln L(w, b) &= -\ln f_{w, b}(x_1) - \ln f_{w, b}(x_2) - \dots - \ln f_{w, b}(x_n) \\ &= - \left[\hat{f}_1 \ln f_{w, b}(x_1) + (1 - \hat{f}_1) \ln (1 - f_{w, b}(x_1)) \right] \\ &\quad - \left[\hat{f}_2 \ln f_{w, b}(x_2) + (1 - \hat{f}_2) \ln (1 - f_{w, b}(x_2)) \right] \end{aligned}$$

$$\dots$$

$$\begin{aligned} \therefore -\ln L(w, b) &= \ln f_{w, b}(x_1) + \ln f_{w, b}(x_2) + \dots + \ln (1 - f_{w, b}(x_n)) \\ &= \sum_{i=1}^n \left[\hat{f}_i \ln f_{w, b}(x_i) + (1 - \hat{f}_i) \ln (1 - f_{w, b}(x_i)) \right] \\ &\quad (\text{cross entropy loss } \mathcal{L}[f(x_i), y_i]) \\ &= \sum_{i=1}^n \mathcal{L}(f(x_i), y_i) \end{aligned}$$

$$\text{⑥ } \boxed{\frac{\partial \ln L(w, b)}{\partial w_i} = \sum_{i=1}^n \left[\hat{f}_i \frac{\ln(1)}{\partial w_i} + (1 - \hat{f}_i) \frac{\ln(1)}{\partial w_i} \right]} \quad (\text{Gradient Descent})$$

$$\rightarrow = \sum_{i=1}^n \left[\hat{f}_i \frac{\partial \ln f_{w, b}(x_i)}{\partial w_i} + (1 - \hat{f}_i) \frac{\partial \ln (1 - f_{w, b}(x_i))}{\partial w_i} \right]$$

$$(z = w^T x + b) = \sum_i w_i x_i + b, \quad \frac{\partial z}{\partial w_i} = x_i$$

$$= \sum_i (-(\hat{f}_i - f_{w, b}(x_i)) x_i)$$

$$w_i \leftarrow w_i - \eta \sum_i (-(\hat{f}_i - f_{w, b}(x_i)) x_i)$$

样本 $(x_i, j_i), (x_0, j_0), \dots, (x_n, j_n), j \in \{0, 1\}$

$$P(y_i, x_i) = P(j_i=1|x_i) = [P(j_i=1|x_i)]^{j_i} [1-P(j_i=1|x_i)]^{1-j_i}$$

似然函数: $L(\theta) = \prod P(y_i, x_i)$, 求 θ^*

$$\begin{aligned} \ln L(\theta) &= \ln \left(\prod P(j_i=1|x_i) \right)^{j_i} \left(1 - P(j_i=1|x_i) \right)^{1-j_i} \\ &= \sum_{i=1}^n j_i \ln \frac{P(j_i=1|x_i)}{1 - P(j_i=1|x_i)} + \sum_{i=1}^n \ln (1 - P(j_i=1|x_i)) \\ &= \sum_{i=1}^n j_i (\theta^T x_i) - \sum_{i=1}^n \ln (1 + e^{\theta^T x_i}) \\ \frac{\partial \ln L(\theta)}{\partial \theta} &= \sum_{i=1}^n (j_i - \sigma(\theta^T x_i)) x_i \quad \left(\sigma(z) = \frac{1}{1 + \exp(-z)} \right) \end{aligned}$$

$$\begin{aligned} \text{Gradient Descent: } \theta^{t+1} &= \theta^t - \alpha \frac{\partial \ln L(\theta)}{\partial \theta} \\ &= \theta^t - \alpha \sum_{i=1}^n (j_i - \sigma(\theta^T x_i)) x_i \end{aligned}$$

Linear Regression, (R^2 vs)

$$\textcircled{1} \quad f(x) = w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b_{\text{bias}}$$

$$\rightarrow f(x) = [w]^T [x] + b$$

\textcircled{2} Loss function

$$L(f) = \sum_{i=1}^n (f(x_i) - y_i)^2 \quad \text{if } L(w, b) = \sum_{i=1}^n (f(x_i) - (w^T x_i + b))^2$$

(least square method)

\textcircled{3} pick best function

$$L(w^*, b^*) = \arg \min_{(w, b)} \sum_{i=1}^n (f(x_i) - y_i)^2 = \arg \min_{(w, b)} \sum_{i=1}^n (y_i - w^T x_i - b)^2$$

\textcircled{4} get w^*, b^* (Gradient descent)

$$\frac{\partial L(\theta)}{\partial w} = 2 \left[\sum_{i=1}^n w^T x_i^2 - \sum_{i=1}^n (y_i - b) x_i \right], \quad \frac{\partial L(\theta)}{\partial b} = 2 \left[\sum_{i=1}^n (y_i - b) \right]$$

$$\therefore \frac{\partial L}{\partial w} = 0, \quad \frac{\partial L}{\partial b} = 0$$

$$\boxed{w = \frac{\sum j_i (x_i - \bar{x})}{\sum x_i^2 - \frac{1}{n} (\sum x_i)^2}, \quad b = \frac{1}{n} \sum (y_i - w^T x_i)}$$

multivariate Linear Regression.

$$\hat{w}^* = \arg \min_{\hat{w}} \underbrace{([y] - [\hat{w}^T x])^T ([y] - [\hat{w}^T x])}_{\text{F}(\hat{w})}$$

$$\frac{\partial F(\hat{w})}{\partial \hat{w}} = 2 [x]^T ([\hat{w}^T x] - [y])$$

$$\therefore \hat{w}^* = ([x]^T [x])^{-1} [x]^T [y]$$

$$\therefore f(x) = w^T x + b, \quad \boxed{\hat{w}^* = (w^T x + b)^{-1} x^T y}$$

以线性模型: $y = g^{-1}(w^T x + b)$

$$\therefore f(y) = w^T y + b, \quad f(\cdot) \text{ 为损失函数.}$$

Gradient Descent (单元)

$y = h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \dots$ (因为参数)

$h_{\theta}(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$

$h_{\theta}(x) \rightarrow y \Rightarrow \left| \int_{\theta} \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \right| \min \text{ cost function 成本函数.}$

gradient descent. $(\theta_j) \approx \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \Rightarrow \theta$.

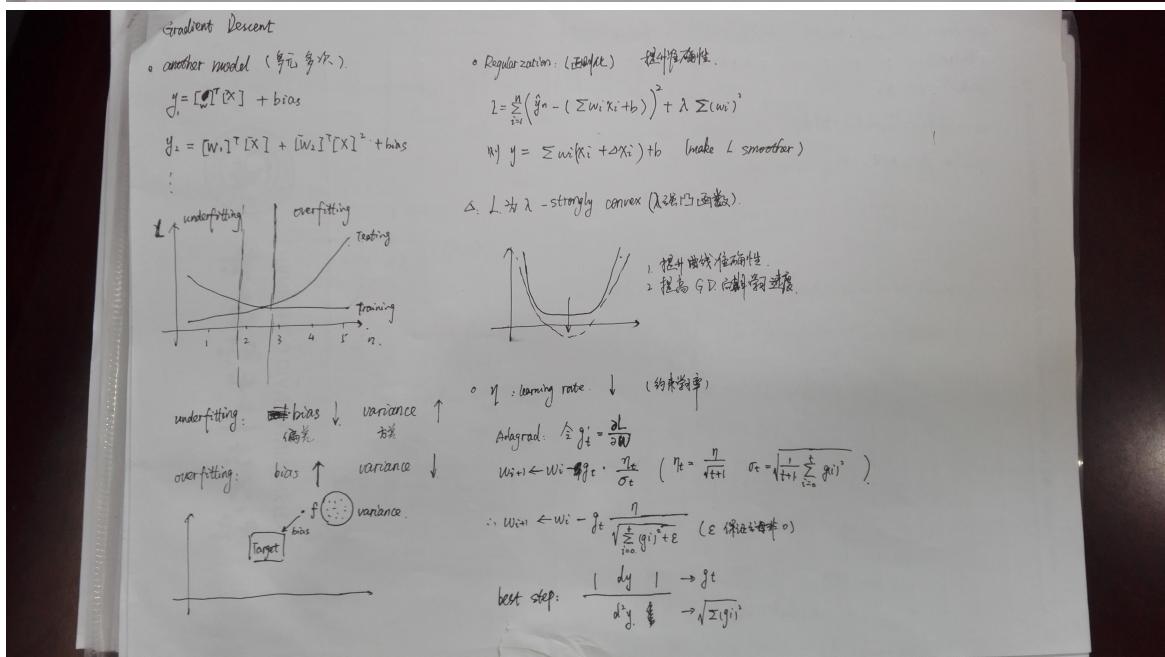
(α : 学习速率)

$\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \quad \theta = (x^T x)^{-1} x^T y$

$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial \theta} \\ \frac{\partial L}{\partial b} \end{bmatrix} \text{ gradient}$

地址: 浙江杭州市下沙高教园区 电话: 86-571-86915072 网址: <http://www.hdu.edu.cn> 电子信箱: office@hdu.edu.cn

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY



KNN (k最近邻)

训练样本 X , 测试样本 z ,
 $P(\text{error}) = 1 - \sum_{c \in \mathcal{C}} P(c|x) P(c|z)$

$$\approx 1 - \sum_{c \in \mathcal{C}} P^2(c|x)$$

$$\leq 1 - P^2(c^*|x)$$

$$\leq 2(1 - P(c^*|x))$$

$$(c^* = \operatorname{argmax}_{c \in \mathcal{C}} P(c|x))$$

泛化错误概率 $\leq 2 \times \text{bias}$ (最优分类器错误概率)

降维/度量

令 $B = Z^T Z \in \mathbb{R}^{m \times m}$, 则 $b_{ij} = z_i^T z_j$

$$\begin{aligned} \text{dist}_{i,j}^2 &= \|z_i\|^2 + \|z_j\|^2 - 2 z_i^T z_j \\ &= b_{ii} + b_{jj} - 2 b_{ij}. \end{aligned}$$

令样本已中心化, 即 $\sum_{i=1}^m z_i = 0$

$$\sum_{i=1}^m \text{dist}_{i,j}^2 = \text{tr}(B) + m b_{jj}$$

$$\sum_{i=1}^m \sum_{j=1}^m \text{dist}_{i,j}^2 = \text{tr}(B) + m b_{ii}$$

$$\sum_{i=1}^m \sum_{j=1}^m \text{dist}_{i,j}^2 = 2m \text{tr}(B)$$

$$(\text{tr}(B) = \sum_{i=1}^m \|z_i\|^2)$$

降维处理 (MDS)

训练样本 m , 距离矩阵 $D \in \mathbb{R}^{m \times m}$, $\text{dist}_{i,j} = \|x_i - x_j\|$

高维空间表示 $Z \in \mathbb{R}^{d \times m}$
 且 Z 在空间与原空间中样点距离一致
 即 $\text{dist}_{i,j} = \|z_i - z_j\|$

$b_{ij} = -\frac{1}{2} (\text{dist}_{ij}^2 - \text{dist}_{i,i}^2 - \text{dist}_{j,j}^2 + \text{dist}_{i,j}^2)$

支持向量机 (SVM)

样本集 $D = \{(x_i, y_i) | (x_1, y_1), \dots, (x_m, y_m)\}$, $y_i \in \{-1, +1\}$

1. $\mathbf{线性可分}$: $[w]^T \cdot [x] + b = 0$ (w 为法向量, b 为偏移量)

(w, b) 的最优点面: $\text{Distance } x \rightarrow (w, b)$, $\gamma = \frac{1 - \|w\|^2 + b}{\|w\|}$

$\text{Distance } x_i \rightarrow (w, b) = \gamma = \frac{2}{\|w\|}$

2. $\mathbf{核 SVM}$: $\max_{w,b} \frac{2}{\|w\|}$

s.t. $y_i (w^T x_i + b) \geq 1 \quad i=1, 2, \dots, m$

$\min \frac{1}{2} \|w\|^2$

s.t. $y_i (w^T x_i + b) \geq 1 \quad i=1, 2, \dots, m$

3. **核编程 (对偶问题)**

核编程求解方法: $\alpha_i \geq 0, \alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)$

$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i (w^T x_i + b))$ (求 $\max_{w,b} L(w, b, \alpha)$)

$\frac{\partial L}{\partial w} = 0, \frac{\partial L}{\partial b} = 0 \rightarrow [w] = \sum_{i=1}^m \alpha_i y_i x_i$

$\alpha = \sum_{i=1}^m \alpha_i y_i$

$\frac{\partial}{\partial \alpha} \max_{\alpha} \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j)$

$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0, \quad \alpha_i \geq 0, i=1, 2, \dots, m$

$\frac{\partial}{\partial \alpha} \min_{\alpha} \left(\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j) - \sum_{i=1}^m \alpha_i \right)$

$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0, \quad \alpha_i \geq 0, i=1, 2, \dots, m$

4. **核编程 (SMO)**

固定 α_{i+1} 并将 α_i 代入, 求 α_i 的极值, 通过 α_i 变更操作后找出 \max_{α}

对偶的解: $\begin{cases} \alpha_i \geq 0 \\ y_i f(\alpha_{i+1}) \geq 1 \\ \alpha_i (y_i f(\alpha_{i+1}) - 1) = 0 \end{cases} \Rightarrow \begin{cases} \alpha_i^* = \frac{1}{2} \alpha_i \\ \frac{\partial L}{\partial w} = 0, \frac{\partial L}{\partial b} = 0 \\ (f_i^* f_{i+1}^*) - 1 = 0 \end{cases}$

$w^* \cdot x + b^* = 0$

SVM 核函数 拉格朗日核空间

非线性支持向量机

$\phi(x)$ 为 x 映射后的特征向量

$$\max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

等价于 $f(x) = w^T \phi(x) + b$

$$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C \quad i=1, 2, \dots, N$$

$$\text{求 } \min \frac{1}{2} \|w\|^2$$

$$\text{t.e. } \alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$$

$$\text{s.t. } y_i(w^T \phi(x_i) + b) \geq 1 \quad i=1, 2, \dots, m$$

$$\text{对偶问题} \quad \max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

$$\text{核函数 } K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

等价样本 $D = \{x_1, x_2, \dots, x_m\}$

$$K = \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_m) \\ \vdots & \ddots & \vdots \\ K(x_m, x_1) & \dots & K(x_m, x_m) \end{bmatrix}$$

线性核 $K = x_i^T x_j$

多项式核 $(x_i^T x_j)^d$ (d 为数 $d \geq 1$)

Semi-Supervised learning (soft-training)

1. initialization $\theta = \{P(c_1), P(c_2), \mu_1, \mu_2, \Sigma\}$

2. 计算概率 $P_B(c_i | x_u)$

$$3. \text{ update model: } P(c_i) = \frac{N_i + \sum_{x_u} P(c_i | x_u)}{N} \quad \left(\frac{N_1 \cdot 70\% + 1}{N} \text{ 有 1 例 } \right)$$

$$\mu_i = \frac{1}{N_i} \sum_{x_r \in c_i} x_r + \frac{1}{\sum_{x_u} P(c_i | x_u)} \sum_{x_u} P(c_i | x_u) x_u$$

same as μ_2

return step 2

1. labelled data $\{(x_r, \hat{y}_r)\}_{r=1}^R$

unlabelled data $\{x_u\}_{u=1}^{R+U}$

2. Repeat:

train f^* from labelled data

Apply f^* into unlabelled data

Obtain $\{(x_u, \hat{y}_u)\}_{u=1}^{R+U}$



Remove $\{\text{a set of data}\}$ from unlabelled data
and add in \downarrow labelled data

Example:

maximum likelihood (最大似然估计) with labelled data:

$$\log L(\theta) = \sum_{x_r} \log P_B(x_r, \hat{y}_r) = \sum_{x_r} \log P_B(x_r | \hat{y}_r) \cdot P_B(\hat{y}_r)$$

max ... + unlabelled data.

$$\log L(\theta) = \sum_{x_r} \log P_B(x_r, \hat{y}_r) + \sum_{x_u} \log P_B(x_u, \hat{y}_u)$$

$$P_B(x_u | c_1) P_B(c_1) + P_B(x_u | c_2) P_B(c_2)$$

e.g. $\theta: \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} \leftarrow$ hard class 1: $x_u [1]$ ✓

soft class 1: 70% $x_u [0.7]$
class 2: 30% $x_u [0.3]$

(low-density separation)

OMO 算法

1. α, α_2 二值，松弛 $\lfloor (\alpha, \alpha_2 \text{ 为变量, 其它为常量}) \rfloor$

2. $W(\alpha_1, \alpha_2) = \frac{1}{2} K_{11} \alpha_1^2 + \frac{1}{2} K_{22} \alpha_2^2 + \sum_j j_i K_{ij} \alpha_i \alpha_j - (\alpha_1 + \alpha_2)$

α_1, α_2

$\alpha_1, \alpha_2 \text{ 为变量, 其它为常量}$

$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + \frac{j_1(E_1 - E_2)}{K_{11} + K_{22} - 2K_{12}}$

$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} + \frac{j_2(E_1 - E_2)}{K_{11} + K_{22} - 2K_{12}}$ ②

$\| \alpha_1^{\text{new}} - \alpha_1^{\text{old}} \|$

$\alpha_1^{\text{old}} + \alpha_2^{\text{old}} = - \sum_{i=3}^n j_i \alpha_i \quad 0 \leq \alpha_i \leq C \quad i=1, 2, \dots, n$

$(K_{ij} = K(x_i, x_j), i, j = 1, 2, \dots, n, C \text{ 为常数})$

$\alpha_2^{\text{new}} = \begin{cases} \min(0, \alpha_2^{\text{old}}) & \alpha_2^{\text{new}} > \min \\ \alpha_2^{\text{old}} & \alpha_2^{\text{new}} \leq \alpha_2^{\text{old}} \leq \min \\ \max(0, \alpha_2^{\text{old}}) & \alpha_2^{\text{new}} < \max \end{cases}$

$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + j_1 j_2 (y_2^{\text{old}} - \alpha_2^{\text{new}})$

$\alpha_1^{\text{old}} \neq j_1$

$\nabla_{\alpha_1} W = 0 \Rightarrow \alpha_1^{\text{new}}$

$\nabla_{\alpha_2} W = 0 \Rightarrow \alpha_2^{\text{new}}$

$\frac{\partial}{\partial \alpha_1} + j_1 \cdot \max(0, \alpha_1^{\text{old}} - \alpha_2^{\text{old}}) \leq \alpha_1^{\text{new}} \leq \min(C, C + \alpha_1^{\text{old}} - \alpha_2^{\text{old}})$ ③

$\frac{\partial}{\partial \alpha_2} + j_2 \cdot \max(0, \alpha_2^{\text{old}} + \alpha_1^{\text{old}} - C) \leq \alpha_2^{\text{new}} \leq \min(C, \alpha_2^{\text{old}} + \alpha_1^{\text{old}})$ ④

$\frac{\partial}{\partial \alpha_1} j(x) = \sum_{i=1}^n \alpha_i j_i K(x_i, x) + b$

$E_i = g(\alpha_1) - y_i = \left(\sum_{j=1}^n \alpha_j j_i K(x_j, x_i) + b \right) - y_i \quad i=1, 2$

(E_i : 偏差, y_i : 对输入值 x_i 的预测与真实的 y_i 间的差)

○ 神经网络尺寸 (全连接)

输入层 中间层 损失层
 $m \cdot k_1 \cdot k_2 \cdot k_3 \cdot n$

神经元 $\sum k_i + n$ (输入层不算)

权重 $m \cdot k_1 + \sum k_{i+1} \cdot k_i + k_3 \cdot n$

偏置 $\sum k_i + n = \text{神经元}$

可学习参数 = 神经元 + 权重

$f = 1 / (1 + \exp(-x))$

$x = np.random.randn(3, 1)$

$h_1 = f(np.dot(w_1, x) + b_1)$ $w_1 = 4 \times 3$

$h_2 = f(np.dot(w_2, h_1) + b_2)$ $w_2 = 5 \times 4$

$out = np.dot(w_{out}, h_2) + b_{out}$

前向传播 = 激活函数 (缩放乘法 + 偏置)

地 址: 浙江杭州下沙高教园区 电 话: 0571-86915072 网 址: <http://www.hdu.edu.cn> 电子 邮 箱: office@hdu.edu.cn

○ 数据预处理

Normalisation (归一化) $x_{norm} = \frac{x - \text{mean}(x)}{\text{std}(x)}$

PCA 降维 $x = np.mean(x, axis=0)$

Cov = $\text{np.dot}(x.T - x) / x.shape[0]$

U, S, V = np.linalg.svd(cov)

(U为特征向量, S为维数矩阵)

$X_{rot_reduced} = \text{np.dot}(X, U[:, :100])$

将 $X \in [N \times D]$ 转到 $[N \times 100]$

Whitening (白化): $X_{white} = X_{rot_reduced} / np.sqrt(S + 1e-5)$

将 Cov矩阵变为单位矩阵

○ 标准化和规范化 (standardization)

1. 小值归一化, $w = \frac{w - \text{min}(w)}{\text{max}(w) - \text{min}(w)}$ 但会使得小数溢出

2. $\sqrt{c} \text{option}$ 规范化: $w = np.random.randn(n) / \sqrt{c}$ (n为输入数据的数量)

梯度初值化, 偏置初始化, 权重归一化

梯度为 \sqrt{n} 的高斯分布梯度初始化方法

杭州电子科技大学

HANGZHOU DIAZI UNIVERSITY

○ 正则化 regularization 防止 overfitting

L1 正则 $w^T x + \lambda \|w\|_1$ (L1正则化惩罚)

L2 正则 $w^T x + \frac{1}{2} \lambda \|w\|^2$

最大范数约束 $\|w\|_2 < c$ (C-范数约束)

随机梯度 超过概率 P 被激活或为 0

$H_1 = np.maximum(0, np.dot(w_1, x) + b_1)$

$U_1 = (np.random.rand(*H_1.shape) < p) / p$ 随机梯度矩阵

$H_1 *= U_1$

$H_2 = np.maximum(0, np.dot(w_2, H_1) + b_2)$

$U_2 = \dots$

$out = np.dot(w_{out}, H_2) + b_{out}$

反向传播

参数更新

Def predict(x):

○ 损失函数 $L = \frac{1}{N} \sum_i L_i$ $f = f(w, x_i)$

1. 分类问题损失: $L_i = \sum_j \max(0, f_j - f_{j+1} + 1)$ (SVM)

$L_i = -\log\left(\frac{e^{f_j}}{\sum_j e^{f_j}}\right)$ (softmax)

2. 二分类问题损失: $L_i = \sum_j \max(0, 1 - y_{ij} f_j)$ $y_{ij} \in \{-1, 1\}$

$L_i = \sum_j [y_{ij} \log(\sigma(f_j)) + (1 - y_{ij}) \log(1 - \sigma(f_j))]$

$(\sigma(f_j) = \sigma(w^T x + b) = P(j=1 | x^T w, b))$

梯度 $\nabla_{f_j} L_i = y_{ij} - \sigma(f_j)$

3. 回归问题损失: $L_i = \|f - y_i\|_2^2$ (L2范数)

$L_i = \|f - y_i\|_1 = \sum_j |f_j - y_{ij}|$ (L1范数)

○ BN 算法

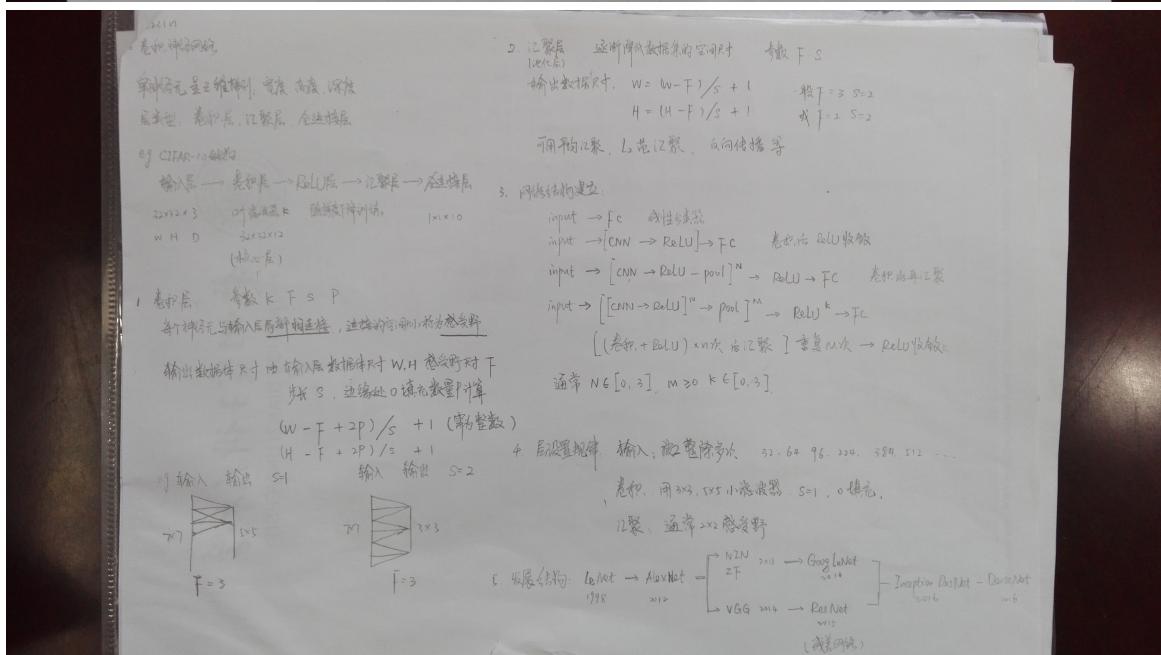
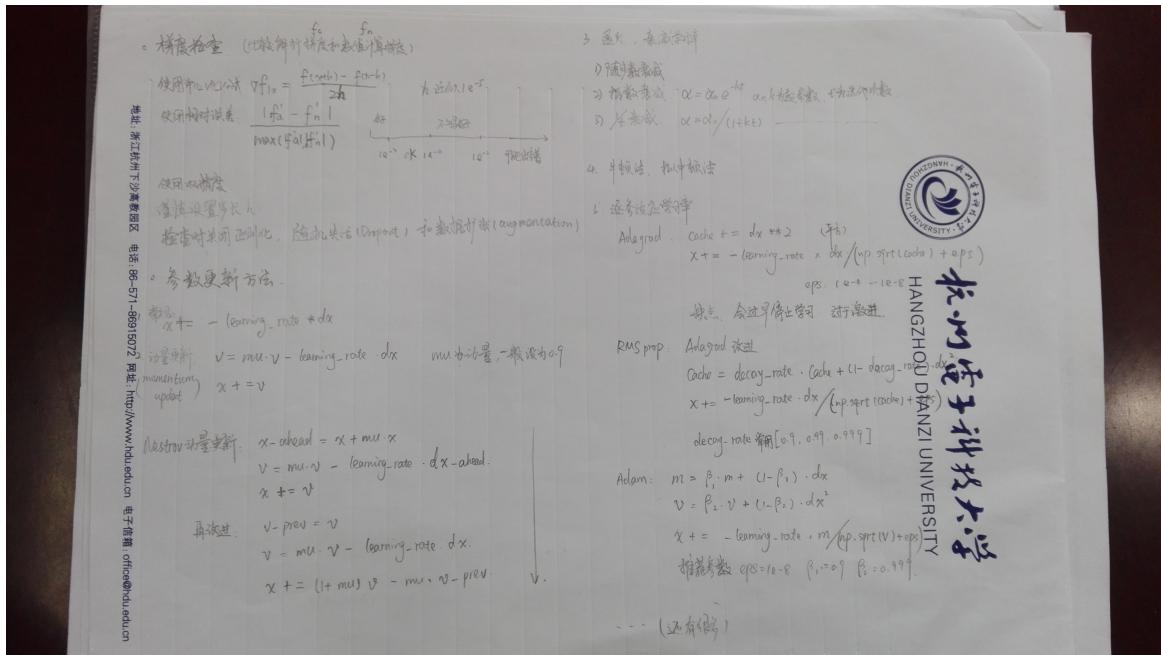
$m = k \cdot \text{mean}(x, axis=1)$

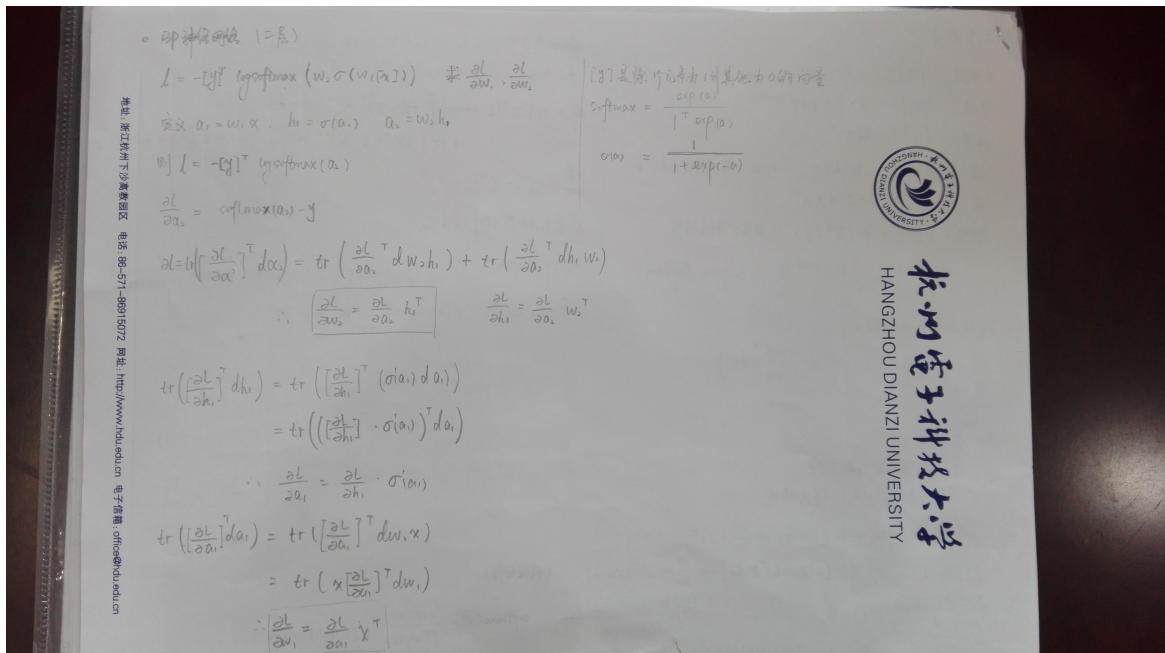
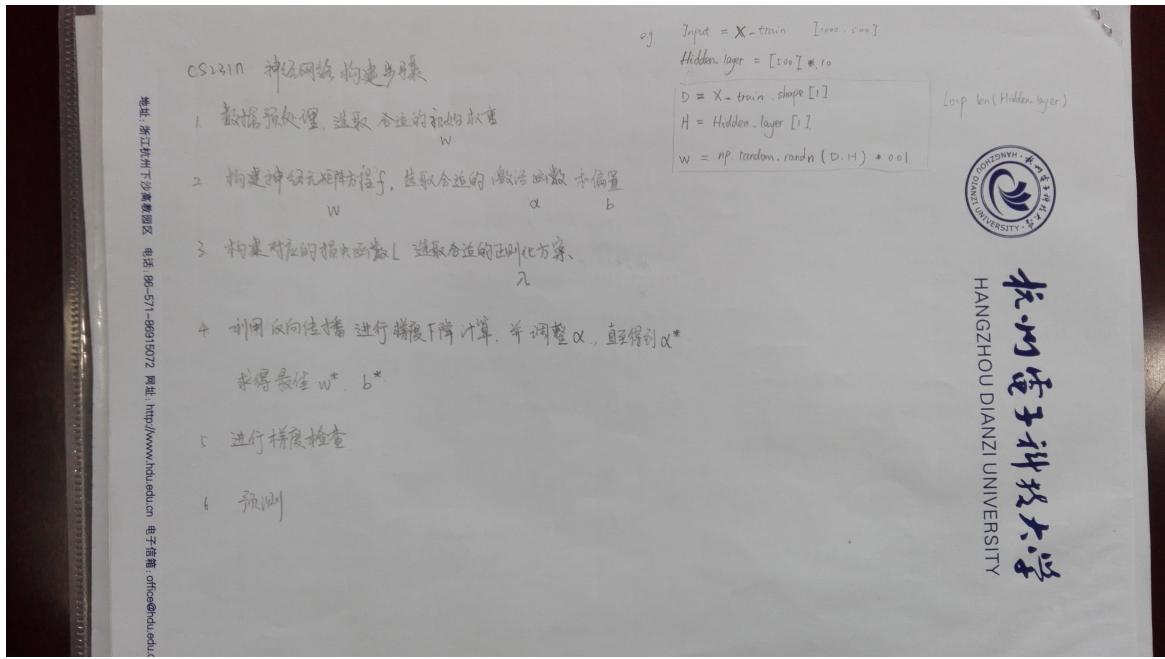
$std = k \cdot \text{std}(x, axis=1)$

$X_{-normalized} = (x - m) / \sqrt{std + \epsilon}$ $(\beta = 0.9)$

$out = \gamma * x + \beta$ 重参数化

BN 可用于任何神经网上, 处理激活函数 $Z = g(BN(w^T x))$





加法: $d(X \pm Y) = dX \pm dY$

乘法: $d(XY) = dXY + XdY$

转置: $d(X^T) = (dX)^T$

迹: $d \text{tr}(X) = \text{tr}(dX)$ ↑ 为吉布斯对角线元素之和

逆: $dX^{-1} = -X^{-1}dXX^{-1}$

行列式: $d|X| = \text{tr}(X^T dX)$ ↑ 为 X 伸缩矩阵

或 $d|X| = |X| \text{tr}(X^T dX)$ (Laplace 展开法则)

矩阵导数与微分关系:

$$df = \sum_{i,j} \frac{\partial f}{\partial X_{ij}} dX_{ij} = \text{tr} \left(\frac{\partial f}{\partial X}^T dX \right)$$

等

① 求微分

② 做上述 $\text{tr}(\cdot)$ 为支撑

③ 对矩阵微分关系得出结论

eg. $f = a^T X b$: $df = \text{tr}(ba^T dX) \rightarrow \frac{\partial f}{\partial X} = ab^T$

$$f = \|Xw - b\|^2, df = \text{tr}[2(Xw - b)^T X dw] \rightarrow \frac{\partial f}{\partial w} = 2X^T(Xw - b) \quad (\text{线性回归})$$

△ $f = -y^T \log \text{softmax}(wx)$: $df = -y^T dwx + \frac{f^T(\exp(wx) \cdot (dwx))}{f^T \exp(wx)} \quad \text{softmax}(x) = \frac{\exp(x)}{f^T \exp(x)} \quad (\text{Logistic 回归})$

$$\rightarrow \frac{\partial f}{\partial w} = [\text{softmax}(wx) - y]^T X^T$$

最大似然估计: $X_1, X_2, \dots, X_n \sim N(\mu, \Sigma)$

$$f = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^T \Sigma^{-1} (x_i - \bar{x})$$

$$df = \text{tr}(\Sigma^{-1} d\Sigma) + \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^T d\Sigma^{-1} (x_i - \bar{x})$$

$$+ \text{tr}((\Sigma^{-1} - \Sigma^{-1} S_n \Sigma^{-1}) d\Sigma) \quad (S_n := \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T)$$

$$\frac{\partial f}{\partial \Sigma} = (\Sigma^{-1} - \Sigma^{-1} S_n \Sigma^{-1})^T$$

Σ 最大似然估计值为 S_n

BP 神经网络: $f = -y^T \log \text{softmax}(w_2 \sigma(w_1 x)) \quad (\sigma(a) = \frac{1}{1 + \exp(-a)})$

令 $w_1 x = a, \sigma(w_1 x) = h, w_2 \cdot h = a$

$$df = \text{tr} \left(-[y]^T dw_2 [h] + \frac{\exp(w_1 x)^T dw_1 [x]}{1^T \exp(w_1 x)} \right) = \text{tr} \left[[y] (\text{softmax}(w_1 x))^T - [y]^T dw_2 \right]$$

$$\rightarrow \frac{\partial f}{\partial w} = \text{softmax}(w_1 x) - y [x]^T$$

• 参考文献 :

- [1] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks : The Official Journal of the International Neural Network Society*, 12(1), 145–151. [http://doi.org/10.1016/S0893-6080\(98\)00116-6](http://doi.org/10.1016/S0893-6080(98)00116-6).
- [2] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2121–2159. Retrieved from <http://jmlr.org/papers/v12/duchi11a.html>.
- [3] Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., ... Ng, A. Y. (2012). Large Scale Distributed Deep Networks. *NIPS 2012: Neural Information Processing Systems*, 1–11. <http://doi.org/10.1109/ICDAR.2011.95>.
- [4] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543. <http://doi.org/10.3115/v1/D14-1162>.
- [5] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. Retrieved from <http://arxiv.org/abs/1212.5701>.
- [6] Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2012). Advances in Optimizing Recurrent Networks. Retrieved from <http://arxiv.org/abs/1212.0901>.
- [7] Sutskever, I. (2013). Training Recurrent neural Networks. PhD Thesis.
- [8] McMahan, H. B., & Streeter, M. (2014). Delay-Tolerant Algorithms for Asynchronous Distributed Online Learning. *Advances in Neural Information Processing Systems (Proceedings of NIPS)*, 1–9. Retrieved from <http://papers.nips.cc/paper/5242-delay-tolerant-algorithms-for-asynchronous-distributed-online-learning.pdf>.
- [9] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems.

- [10] Zhang, S., Choromanska, A., & LeCun, Y. (2015). Deep learning with Elastic Averaging SGD. Neural Information Processing Systems Conference (NIPS 2015), 1–24. Retrieved from <http://arxiv.org/abs/1412.6651>.
- [11] Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1–13
- [12] Zaremba, W., & Sutskever, I. (2014). Learning to Execute, 1–25. Retrieved from <http://arxiv.org/abs/1410.4615>.
- [13] Ioffe, S., & Szegedy, C. (2015). Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv Preprint arXiv:1502.03167v3.
- [14] Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. arXiv, 1–14. Retrieved from <http://arxiv.org/abs/1406.2572>.
- [15] Sutskever, I., & Martens, J. (2013). On the importance of initialization and momentum in deep learning. <http://doi.org/10.1109/ICASSP.2013.6639346>.
- [16] Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., & Martens, J. (2015). Adding Gradient Noise Improves Learning for Very Deep Networks, 1–11. Retrieved from <http://arxiv.org/abs/1511.06807>.
- [17] Andrew Ng, CS 229 lecture notes
- [18] Andrew Ng et al. On discriminative vs. generative classifiers:a comparison of logistic regression and naïve bayes
- [19] Wikipedia, <http://en.wikipedia.org/wiki>
- [20] 知乎, <https://www.zhihu.com/>