机器学习作业 3

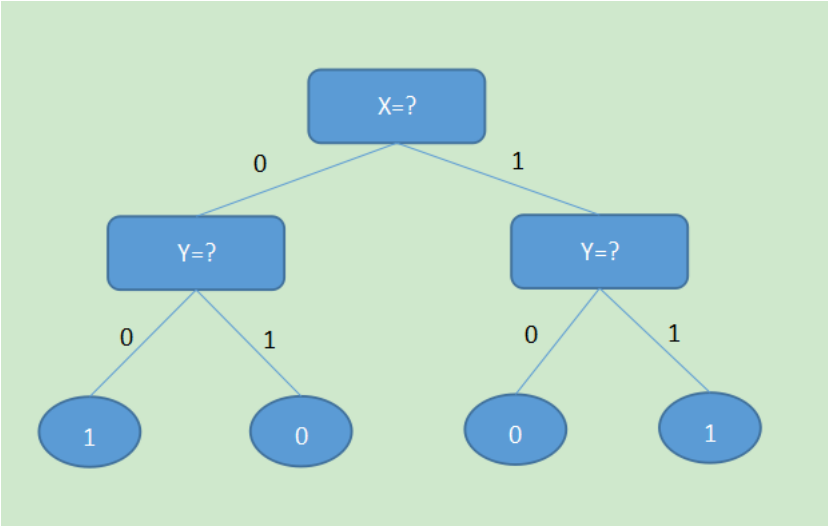
2020 年 11 月 12 日

181860152 周宇翔

1.[20pts]Decision Tree

(1)

f 是 realizable 的



(2)

$p_0 = 0.4, p_1 = 0.6$
$Ent(D) = -\sum_{k=0}^{1} p_k log_2 p_k$

$= -0.4log_20.4 - 0.6log_20.6 = 0.971$

设两个 Feature 分别为 $F_1, F_2(F_1, F_2$ 就是 x_1, x_2 我后来才发现但是懒得改了 ovo)

$Gain(D, F_1) = Ent(D) - 10 * \frac{1}{10} * 0 = 0.971$

$IV(a) = log_210 = 3.32$

$Gain\_ratio(D, F_1) = 0.292$

类似的 $Gain\_ratio(D, F_2) = 0.292$

但是如果把每个 feature 当作离散值的一个分类处理, 会导致虽然每个节点的纯度很高, 但是决策树的泛化能力很弱, 无法对新样本进行有效预测

因此我按照第 i 个分类的取值为 $(5 * i, 5 * (i + 1)], i = 0, 1, 2, ...$ 进行处理

$Gain(D, F_1) = Ent(D) - \frac{|D^4|}{|D|}Ent(D^4) - \frac{|D^6|}{|D|}Ent(D^6) - \frac{|D^8|}{|D|}Ent(D^8) - \frac{|D^9|}{|D|}Ent(D^9) - \frac{|D^{10}|}{|D|}Ent(D^{10})$

$= 0.97 - 0.4 * (-0.25log_20.25 - 0.75log_20.75) - 0.1 * 0 - 0.1 * 0 - 0.1 * 0 - 0.3 * (-0.33log_20.33 - 0.66log_20.66)$

$= 0.17$

$IV(F_1) = -0.4log_20.4 - 0.3log_20.3 - 3 * 0.1log_20.1 = 2.05$

$Gain(D, F_2) = Ent(D) - \frac{|D^7|}{|D|}Ent(D^7) - \frac{|D^8|}{|D|}Ent(D^8) - \frac{|D^9|}{|D|}Ent(D^9) - \frac{|D^{10}|}{|D|}Ent(D^{10}) - ...$

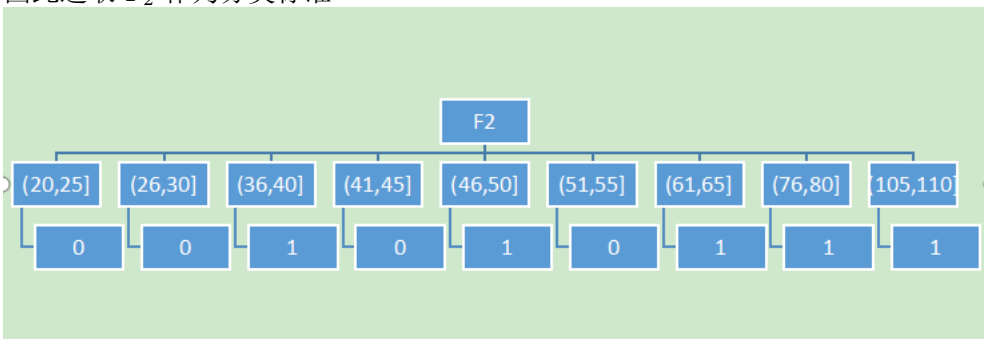
$= 0.97$

$IV(F_2) = -0.2log_20.2 - 9 * 0.1log_20.1 = 3.12$

$Gain\_ratio(D, F_1) = \frac{Gain(D,F_1)}{IV(F_1)} = 0.083$

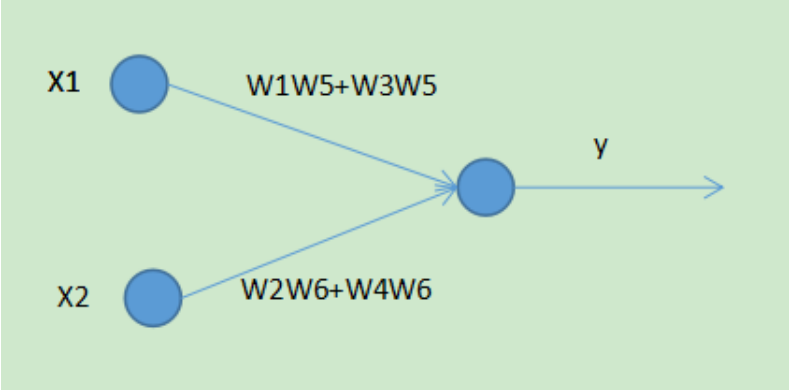
$Gain\_ratio(D, F_2) = \frac{Gain(D,F_2)}{IVF_2} = 0.31$

因此选取 F_2 作为分类标准



2.[20pts]Neural Network

(1)

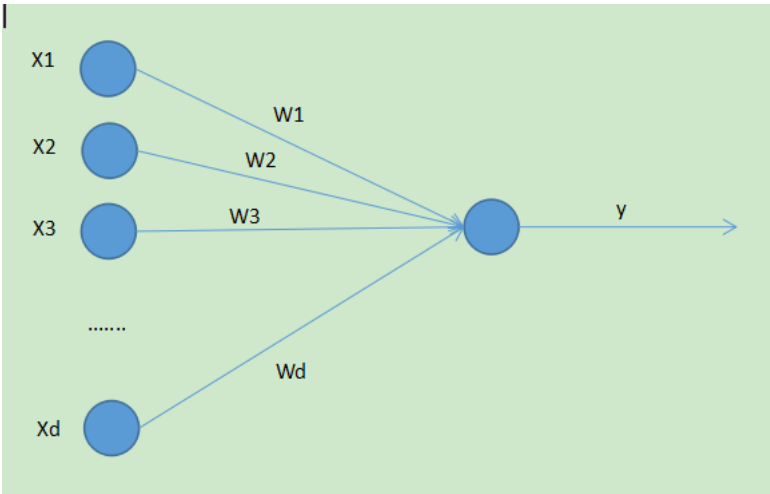


(2)

可以, 因为在这个计算过程中始终只会有输入变量 $X_1, ..., X_d$ 的一次项存在, 不会产生交叉项, 高次项或者非线性项, 因此可以不用隐层表示

(3)

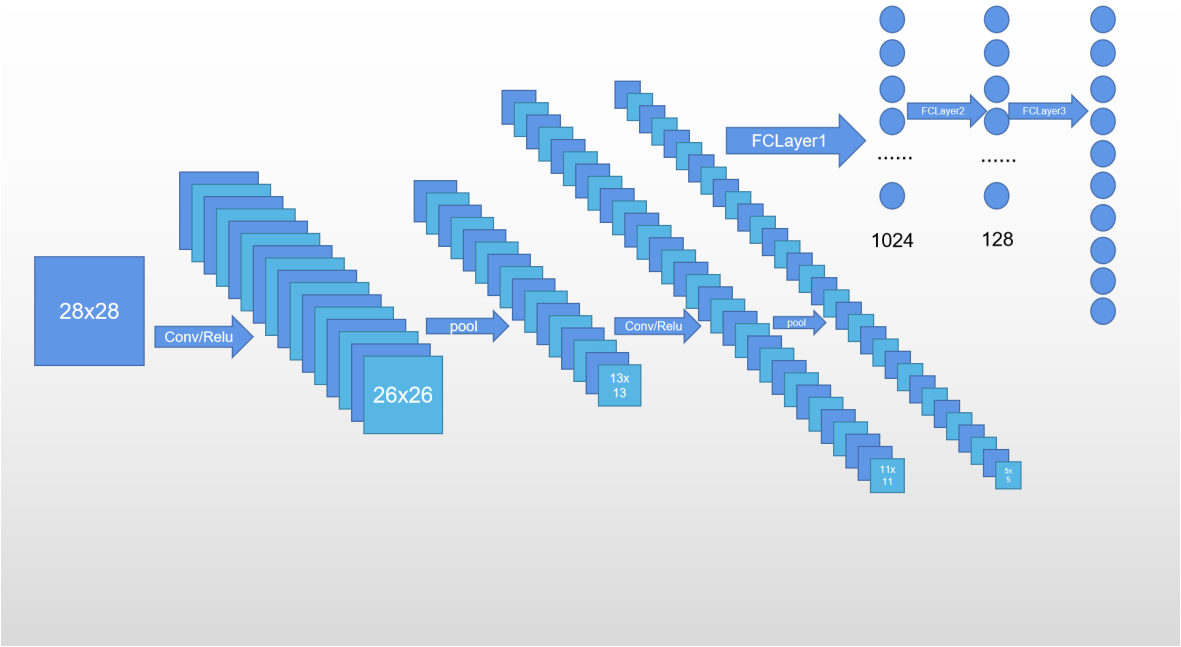
对于 Logistic Regression, 输入 $\mathbf{X} = X_1, ..., X_d$
输出为 $z = \mathbf{w}^T X + b$, 并用 sigmoid 函数 $y = \frac{1}{1+e^{-z}}$ 来将 z 值转化为一个接近 0 或 1 的 y 值
Logistic Regression 通过令 $(\mathbf{w},\mathbf{b}) = argmin_{\mathbf{w},\mathbf{b}} \sum_{i=1}^{d} (\hat{y}_i - y_i)^2$ 来逼近 $\mathbf{w}$ 和 $\mathbf{b}$ 的最优解
在求解过程中使用梯度下降法, 梯度为 $\nabla f(w) = X^T X w - X^T y$



在神经网络中, 我们如图考虑:
输出节点的输入为 $\sum_{i=1}^{d} X_i W_i$, 对应到对数几率回归中的 $w^T X$, 激活函数为 $y = f(\sum_{i=1}^{d} X_i W_i - \theta)$, 如果让 θ 对应到 $-b$,f 为 sigmoid 函数
则 $w^T X + b$ 可表示为 $\sum_{i=1}^{d} X_i W_i - \theta$
则此时的 y 对应到对数几率回归中的 y 值, 此时采用逆传播法, 通过最小化 $E_k = \frac{1}{2} \sum_{i=1}^{d} (\hat{y_i}^k - {y_i}^k)$ 来逼近最优解, 由于此时仍然对各系数取值采用梯度下降法进行调整, 因此若控制所有系数的学习率一致, 该神经网络学习的就是一个对数几率回归的模型

3.[60pts]Neural Network in Practice

(2)

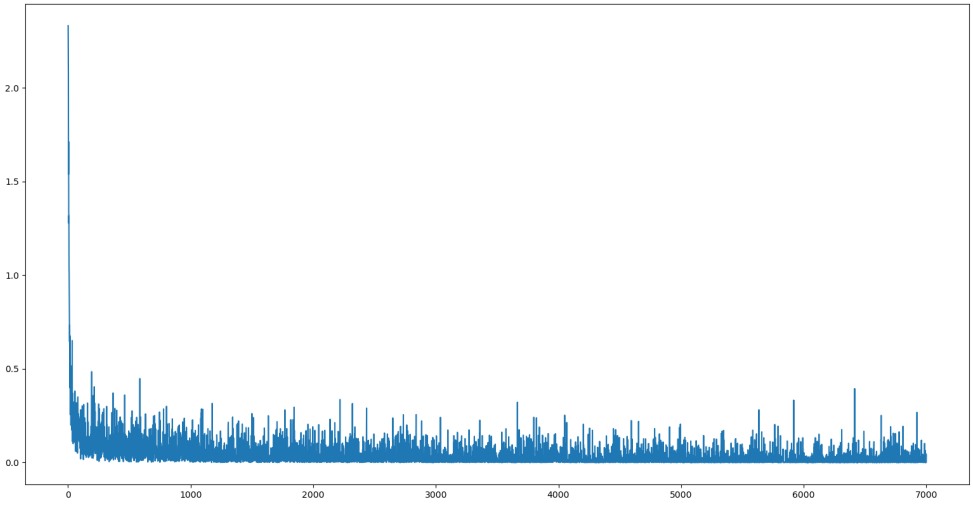


(3)

选取 epochs∈{1,2,3,4,5,6,7,8},Learning Rate∈{0.001,0.002,0.003,0.004,0.005}
进行测试
具体数据见附表 params2.txt
综合 Accuracy,AverageLoss 以及 Epochs 考虑, 选取
Epochs=7,Learning Rate=0.002 进行作为该模型的参数

(4)

对于 training_loss 的变化, 我采用对每一批 (60 个) 数据都记录一次 loss 来刻画, 并使用
matplotlib.pylab 来绘制图像
图大致如下, 具体可见附件 training_loss.png


2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000
6000
7000