Neural network theory and Applications

Homework Assignment 1

汪旭鸿

017032910027

March 11, 2018

Problem 1

One variation of the perceptron rule is:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{e}\mathbf{p}^T$$
$$\mathbf{b}^{new} = \mathbf{b}^{old} + \alpha \mathbf{e}$$

where α is the learning rate. Prove convergence of this algorithm. Does the proof require a limit on the learning rate? Explain.

解答：

Part 1 记号说明：

设线性可分数据集：$\{(p_1, t_1), (p_2, t_2), \dots, (p_N, t_N)\}$，标签值$t_i \in \{-1, +1\}$ $i \in 1,2, \dots, N$

感知机的输出：$a = handlim(wp^T + b)$

为了便于推导，记 $\widehat{w} = (w^T, b)^T$ 同样地，记 $\hat{p} = (p^T, 1)^T$。显然，$\widehat{w} \cdot \hat{p} = wp^T + b.$

Part 2 证明

根据 Novikoff [1]在 1963 年发表的文章，我可以仿照他的工作进行以下推导：

Step 1

因为数据集线性可分，所以一定存在的超平面将训练数据集完全正确分开。

设该超平面为：$\widehat{w}_{opt}\hat{p}^T = w_{opt}p^T + b_{opt} = 0$，且满足条件$\|\widehat{w}_{opt}\| = 1$（范数为 1 便于计算，$\|\widehat{w}_{opt}\|$的大小不影响超平面分布）。

因为数据集被正确分割，所以对于有限的$i \in 1,2, \dots, N$，正样本分布在超平面以上，负样本分布在超平面以下，所以一定有：

$$t_i(\widehat{w}_{opt}\hat{p}_i) = t_i(w_{opt}p_i + b_{opt}) > 0$$

所以上述不等式一定存在下界

$$\gamma = \min_i\{t_i(w_{opt}p_i + b_{opt})\} \tag{1}$$

$$即：\begin{cases} \widehat{w}_{opt}\hat{p}_i \ge \gamma & t_i = 1(正样本) \\ \widehat{w}_{opt}\hat{p}_i \le -\gamma & t_i = -1(负样本) \end{cases}$$

Step 2

跟据题目给定的变种感知机算法，假设初始参数$\widehat{w}_0 = 0$，如果样本被误分类，则参数$\widehat{w}_0$就必更新。参数$\widehat{w}_0$每更新一次，其下标加 1，例如经过第 k-1 次更新，参数$\widehat{w}$会被表示成$\widehat{w}_{k-1}$，即$\widehat{w}_{k-1} = (w_{k-1}^T, b_{k-1})^T$。

假设参数$\widehat{w}$在经过 k-1 次更新之后，发现了样本(p_i, t_i)被参数为$\widehat{w}_{k-1}$的感知机超平面误分类，那么感知机必须进行下一次更新。此时，存在不等式：

$$t_i(\widehat{w}_{k-1}\hat{p}_i) = t_i(w_{k-1}p_i + b_{k-1}) \le 0 \tag{2}$$

并且此时，根据题目要求，参数的更新为：

$$\widehat{w}_k = \widehat{w}_{k-1} + \alpha e\hat{p}_i = \widehat{w}_{k-1} + \alpha(t_i - a_i)\hat{p}_i \tag{3}$$

当误分类时，总是存在关系：$t_i - a_i = 2t_i$

(1) 由公式(1)及公式(3)可知，

$$\widehat{w}_k\widehat{w}_{opt} = \widehat{w}_{k-1}\widehat{w}_{opt} + \alpha(t_i - a_i)\widehat{w}_{opt}\hat{p}_i \ge \widehat{w}_{k-1}\widehat{w}_{opt} + 2\alpha\gamma$$

由上述不等式递归可得：

$$\widehat{w}_k\widehat{w}_{opt} \ge \widehat{w}_{k-1}\widehat{w}_{opt} + 2\alpha\gamma \ge \widehat{w}_{k-2}\widehat{w}_{opt} + 4\alpha\gamma \ge \cdots \ge 2k\alpha\gamma \tag{4}$$

(2) 由公式(3)和公式(2)得：

$$\|\widehat{w}_k\|^2 = \|\widehat{w}_{k-1}\|^2 + 2\alpha(t_i - a_i)\widehat{w}_{k-1}\hat{p}_i + \alpha^2(t_i - a_i)^2\|\hat{p}_i\|^2$$

$$\le \|\widehat{w}_{k-1}\|^2 + 4\alpha^2\|\hat{p}_i\|^2$$

令$R = \max_{1\le i\le N}\|\hat{p}_i\|$，得：

$$\le \|\widehat{w}_{k-1}\|^2 + 4\alpha^2R^2$$

根据递归：$\le \|\widehat{w}_{k-2}\|^2 + 8\alpha^2R^2 \le \cdots \le$

$$\le 4k\alpha^2R^2 \tag{5}$$

(3) 结合不等式(4)(5)可得，

$$2k\alpha\gamma \le \widehat{w}_k\widehat{w}_{opt} \le \|\widehat{w}_k\|\|\widehat{w}_{opt}\| \le 2\sqrt{k}\alpha R$$

结论: 最大误分类次数 k：$k \le \left(\frac{R}{\gamma}\right)^2$

其中$R = \max_{1\le i\le N}\|\hat{p}_i\|$，$\gamma = \min_i\{t_i(\widehat{w}_{opt}\hat{p}_i)\}$

Problem 2.

Suppose the output of each neuron in a multilayer perceptron network is

$$x_{kj} = f\left(\sum_{i=1}^{N_{k-1}} (u_{kji}x_{k-1,i}^2 + v_{kji}x_{k-1,i}) + b_{kj}\right)$$

$$\text{for } k = 2, 3, \cdots, M \text{ and } j = 1, 2, \cdots, N_k$$

where both u_{kji} and v_{kji} are the weights connecting the ith unit in the layer $k-1$ to the jth unit in the layer k, b_{kj} is the bias of the jth unit in the layer k, N_k is the number of units in the k $(1 \le k \le M)$, and $f(.)$ is the sigmoidal activation function.

The structure of the unit is shown as the following figure, and this network is called multi-layer quadratic perceptron (MLQP).

Please derive the back-propagation algorithms for MLQPs in both on-line learning and batch learning ways.

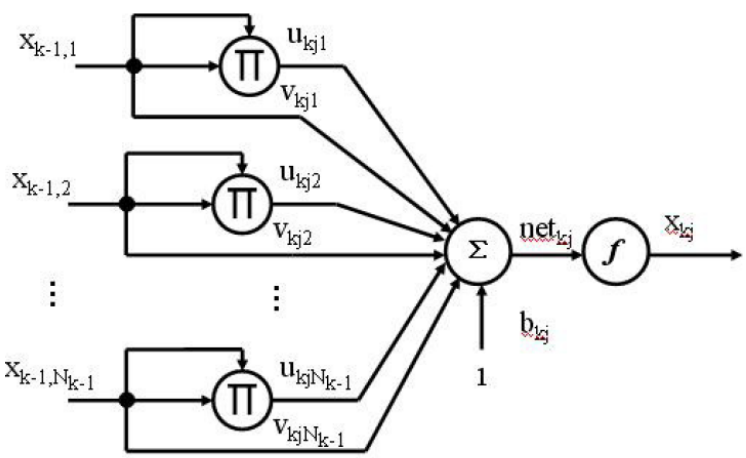

图 1: MLQP 神经元结构

Solution Part a): 当第 k 层为输出层时：

设第 n 次迭代时：$k = 2, 3, ..., M; j = 1, 2, ..., N_k$

第 k 层第 j 个神经元的输出误差为：

$$e_{kj}(n) = d_{kj}(n) - x_{kj}(n) \tag{1}$$

第 k 层的整体输出误差为：

$$\varepsilon(n)=\frac{1}{2}\sum_{j}e_{kj}^{2}(n) \tag{2}$$

$$x_{kj}(n)=f(net_{kj}(n)) \tag{3}$$

其中激活函数 f 为 Sigmoid：$f(x)=\frac{1}{1+e^{-x}}$
其导数 $f'(x)=\frac{e^{-x}}{(1+e^{-x})^{2}}=f(x)(1-f(x))$

$$f'(net_{kj}(n))=f(net_{kj}(n))\big(1-f(net_{kj}(n))\big) \tag{4}$$

$$net_{kj}(n)=\sum_{i=1}^{N_{k-1}}(u_{kji}x_{k-1,i}^{2}+v_{kji}x_{k-1,i})+b_{kj} \tag{5}$$

应用求导链式法则：

考虑在线学习 online learning 的情况：

(1) 对于参数 u_{kji}

$$\frac{\partial\varepsilon(n)}{\partial u_{kji}(n)}=\frac{\partial\varepsilon(n)}{\partial e_{kj}(n)}\frac{\partial e_{kj}(n)}{\partial x_{kj}(n)}\frac{\partial x_{kj}(n)}{\partial net_{kj}(n)}\frac{\partial net_{kj}(n)}{\partial u_{kji}(n)} \tag{6}$$

根据公式 (1)(2)(3)(4)(5), 可以得到以下结果：

$$\frac{\partial\varepsilon(n)}{\partial u_{kji}(n)}=-e_{kj}(n)(1-x_{kj}(n))\sum_{i=1}^{N_{k-1}}x_{k-1,i}^{2}\cdot x_{kj}(n) \tag{7}$$

$$\delta_{uj}(n)=e_{kj}(n)(1-x_{kj}(n))\sum_{i=1}^{N_{k-1}}x_{k-1,i}^{2} \tag{8}$$

(2) 对于参数 v_{kji}

$$\frac{\partial\varepsilon(n)}{\partial v_{kji}(n)}=\frac{\partial\varepsilon(n)}{\partial e_{kj}(n)}\frac{\partial e_{kj}(n)}{\partial x_{kj}(n)}\frac{\partial x_{kj}(n)}{\partial net_{kj}(n)}\frac{\partial net_{kj}(n)}{\partial v_{kji}(n)} \tag{9}$$

同理可以得到以下结果：

$$\frac{\partial\varepsilon(n)}{\partial v_{kji}(n)}=-e_{kj}(n)(1-x_{kj}(n))\sum_{i=1}^{N_{k-1}}x_{k-1,i}\cdot x_{kj}(n) \tag{10}$$

$$\delta_{vj}(n)=e_{kj}(n)(1-x_{kj}(n))\sum_{i=1}^{N_{k-1}}x_{k-1,i} \tag{11}$$

(3) 对于参数 b_{kj}

$$\frac{\partial \varepsilon(n)}{\partial b_{kj}(n)} = \frac{\partial \varepsilon(n)}{\partial e_{kj}(n)} \frac{\partial e_{kj}(n)}{\partial x_{kj}(n)} \frac{\partial x_{kj}(n)}{\partial net_{kj}(n)} \frac{\partial net_{kj}(n)}{\partial b_{kj}(n)} \tag{12}$$

同理可以得到以下结果:

$$\frac{\partial \varepsilon(n)}{\partial b_{kj}(n)} = -e_{kj}(n)(1 - x_{kj}(n)) \cdot x_{kj}(n) \tag{13}$$

$$\delta_{bj}(n) = e_{kj}(n)(1 - x_{kj}(n)) \tag{14}$$

考虑 batch learning 的情况:

如果使用 batch learning，那么公式 (2) 会被替换为:

$$\varepsilon(n) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j} e_{kj}^2(n) \tag{15}$$

注：N 为样本数

所以对于 batch learning，上述的三条结论应该改为:

$$\frac{\partial \varepsilon(n)}{\partial u_{kji}(n)} = -\frac{1}{N} \sum_{n=1}^{N} (e_{kj}(n))(1 - x_{kj}(n)) \sum_{i=1}^{N_{k-1}} x_{k-1,i}^2 \cdot x_{kj}(n) \tag{16}$$

$$\delta_{uj}(n) = \frac{1}{N} \sum_{n=1}^{N} (e_{kj}(n))(1 - x_{kj}(n)) \sum_{i=1}^{N_{k-1}} x_{k-1,i}^2 \tag{17}$$

$$\frac{\partial \varepsilon(n)}{\partial v_{kji}(n)} = -\frac{1}{N} \sum_{n=1}^{N} (e_{kj}(n))(1 - x_{kj}(n)) \sum_{i=1}^{N_{k-1}} x_{k-1,i} \cdot x_{kj}(n) \tag{18}$$

$$\delta_{vj}(n) = \frac{1}{N} \sum_{n=1}^{N} (e_{kj}(n))(1 - x_{kj}(n)) \sum_{i=1}^{N_{k-1}} x_{k-1,i} \tag{19}$$

$$\frac{\partial \varepsilon(n)}{\partial b_{kj}(n)} = -\frac{1}{N} \sum_{n=1}^{N} (e_{kj}(n))(1 - x_{kj}(n)) \cdot x_{kj}(n) \tag{20}$$

$$\delta_{bj}(n) = \frac{1}{N} \sum_{n=1}^{N} (e_{kj}(n))(1 - x_{kj}(n)) \tag{21}$$

Part b): 当第 k 层为隐含层时，假设第 k+1 层是输出层，并存在第 r 个神经元：

$$\frac{\partial \varepsilon(n)}{\partial x_{kj}(n)} = \sum_r e_r(n) \frac{\partial e_r(n)}{\partial x_{kj}(n)} = \sum_r e_r(n) \frac{\partial e_r(n)}{\partial net_{k+1,r}(n)} \frac{\partial net_{k+1,r}(n)}{\partial x_{kj}(n)} \tag{22}$$

又因为，$\varepsilon(n) = \frac{1}{2}\sum_r e_r^2(n)$，$e_r(n) = d_r(n) - x_{k+1,r}(n) = d_r(n) - f(net_{k+1,r}(n))$，

所以

$$\begin{aligned}\frac{\partial e_r(n)}{\partial net_{k+1,r}(n)} &= -f'\big(net_{k+1,r}(n)\big) \\ &= -f\big(net_{k+1,r}(n)\big)\Big(1 - f\big(net_{k+1,r}(n)\big)\Big) \\ &= -x_{k+1,r}(n)\big(1 - x_{k+1,r}(n)\big)\end{aligned} \tag{23}$$

$$\frac{\partial net_{k+1,r}(n)}{\partial x_{kj}(n)} = 2u_{k+1,rj}x_{kj}(n) + v_{k+1,rj} \tag{24}$$

所以公式 (22) 可最终化为：

$$\frac{\partial \varepsilon(n)}{\partial x_{kj}(n)} = -\sum_r e_r(n)x_{k+1,r}(n)\big(1 - x_{k+1,r}(n)\big)(2u_{k+1,rj}x_{kj}(n) + v_{k+1,rj}) \tag{25}$$

所以,

$$\delta_{uj}(n) = -\frac{\partial \varepsilon(n)}{\partial x_{kj}(n)} \frac{\partial x_{kj}(n)}{\partial net_{kj}(n)} \tag{26}$$

所以,

$$\frac{\partial \varepsilon(n)}{\partial u_{kji}(n)} = \delta_{uj}(n) \cdot x_{kj}(n) \tag{27}$$

同理，$\frac{\partial \varepsilon(n)}{\partial v_{kji}(n)}$ $\frac{\partial \varepsilon(n)}{\partial b_{kji}(n)}$ 也可根据式 (25)(26)(27) 求得。参考Part a)，同理易知batch learning

Problem 3. 在任务 3 中, 为了解决这个表情识别问题，使用 Tensorflow 编写一个神经网络。

Solution 为了解决这个表情分类问题，我建立了如图 2 所示的前馈型神经网络。代码请见附件 **HW1 Problem3.py**

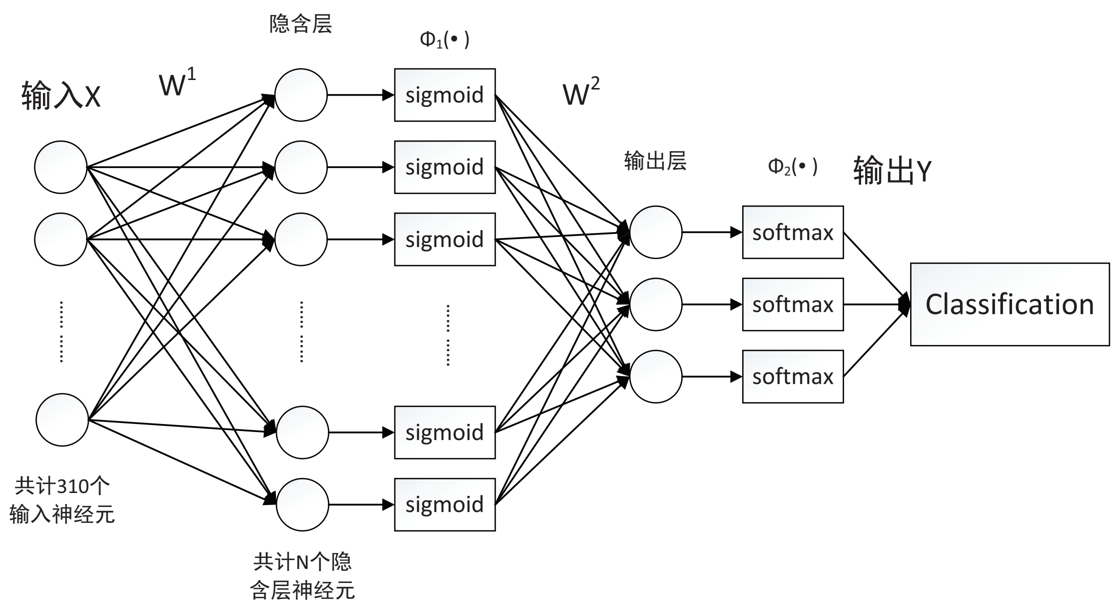

图 2: 三层神经网络结构

如果假设网络的输入是向量 X，输出是 Y 的话，此网络可以用如下的解析式表示 (其中，W 代表的是添加了偏置项的权重向量 $[w_1, w_2, ..., b]^T$,X 和 Y 同样也添加了偏置项'1')：

$$Y = \Phi_2(W^2\Phi_1(W^1X))$$

网络的损失函数定义为交叉熵 (Crossentropy)。注意，为了使用交叉熵损失函数，我事先将给定的 *trainlabel* 和 *testlabel* 数据都转为了 one-hot encoding 的格式。具体来说我将原来的 1 位标签 {-1,0,1} 转换成了三位标签 {001,010,100}。只有这种标签格式才是被交叉熵函数所接受的。

图 3 是我认为的解决这个表情分类问题比较好的训练结果。在这次训练中，我使用了 128 个隐含层神经元，batch size 设置为 128，优化器采用的是学习率为 10^{-5} 的 Adam 优化方法, 在经过 300 个 Epochs 之后，这个模型取得了 80% 左右的准确率，以及 0.97 左右的交叉熵损失。

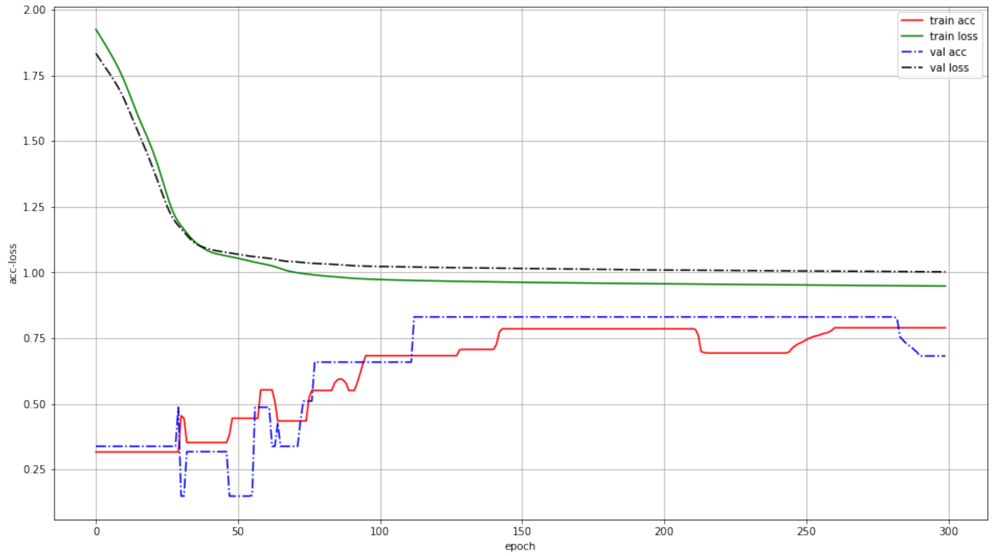

图 3: acc-loss 学习曲线

下表是本次实验中使用不同的参数得到的最终结果，在这里我采用了 earlystopping 机制，如果 loss 经过 5 次 epochs 不再下降，就停止训练。

隐含层个数	学习率	train_loss	test_loss	epochs to stable
512	10^{-3}	0.9384	0.9922	17
512	10^{-4}	0.8294	0.8910	59
512	10^{-5}	0.8360	0.9159	123
256	10^{-3}	1.0581	1.0854	98
256	10^{-4}	0.8799	0.9207	86
256	10^{-5}	0.8360	0.9159	120
128	10^{-3}	1.0642	1.0646	34
128	10^{-4}	0.9644	0.9937	39
128	10^{-5}	0.9943	1.0185	68

结论 1：根据上表以及其他实验过程可知，128 层的隐含层和 10^{-5} 的 Adam 学习率是最好的超参数搭配。

结论 2：当隐含层神经元个数过多时，网络训练非常容易过拟合，即便达到了较低的 loss 也只是假象；当隐含层神经元个数过少（如 64）时，网络不断震荡，根本无法得到靠谱的 loss。

结论 3：在这个表情识别任务、该三层神经网络的背景下，在大部分情况中 10^{-5} 是一个较好的学习率，它可以保证模型在尽量避免过拟合及震荡不收敛的条件下，达到较优的 loss 和准确率。

参考文献

[1] Novikoff A B J. On convergence proofs for perceptrons[R]. STANFORD RESEARCH INST MENLO PARK CALIF, 1963.