机器学习报告

小组#3

小组成员：帅灿宇 20354240

万俊麟 20354123

廖逸霖 20354079

任务完成概况：

1 使用多种超参数组合训练模型得到最佳组合：

Table: Best Hyper-parameters

Hyper-parameter	max_iter	solver	activation	early-stopping	Alpha
	500	Sgd	‘ReLU’	False	10^{-5}
Hyper-parameter	tol	momentum	Hidden_layer_sizes	learning_rate_init	
	10^{-8}	0.9	（5, 3）	0.007	

2 使用得到的最佳超参数组合训练模型得到神经隐层尺寸为（5, 3）的神经网络分类模型。

3 描述模型训练算法

4 模型训练结果：

Table： Result

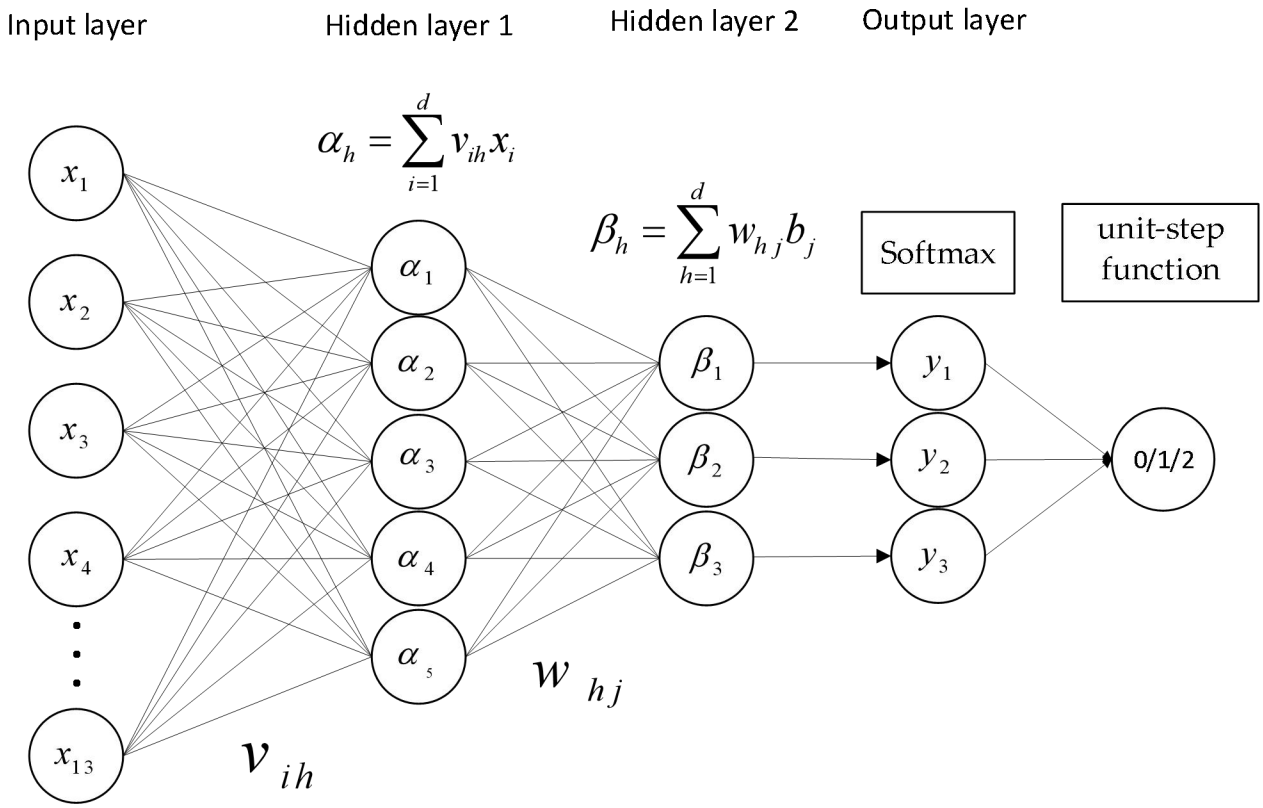
	Train Log Loss Mean	Test Log Loss Mean	Train Accuracy Mean	Test Accuracy Mean	Estimator Loss
Test1	0.04412	0.10298	0.99375	0.9875	0.04419
Test2	0.04824	0.09775	0.99375	0.9875	0.04830
Test3	0.04902	0.08316	0.99375	0.9875	0.04908
Test4	0.04507	0.11518	0.99375	0.9750	0.04514
Test5	0.04526	0.09121	0.9922	0.9875	0.04533

Best accuracy = 0.9875

目录

1 神经网络模型

对数据处理后训练得到的分类器神经网络模型如下：



该神经网络模型隐层数目为 2 层，分别由 5 个及 3 个神经元构成。将数据输入模型，经过两层隐层神经元处理后得到的三个数据由 softmax 函数：

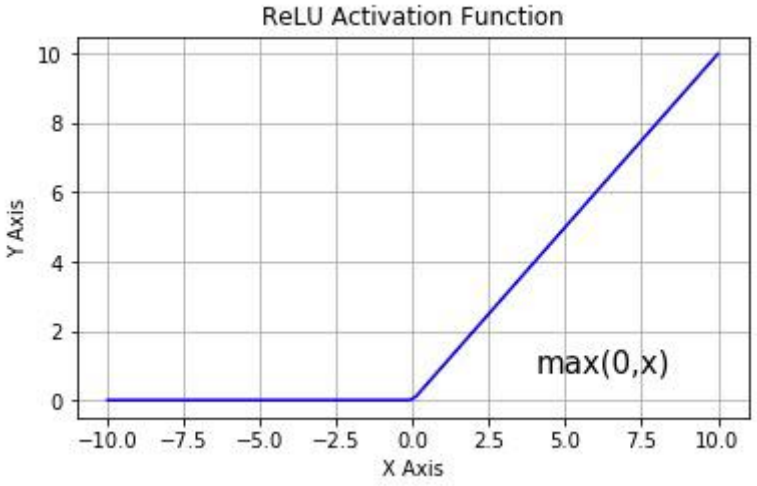
$$Softmax:\ y_i = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}}$$

$$Probability:$$

- $1 > y_i > 0$
- $\sum_i y_i = 1$

求得对应概率，最后由得到的概率值经激活函——ReLU 函数判断后得到最终分类结果。ReLU 函数计算效率高，兼具线性及非线性特性，其表达式及图像如下：

$$f_{ReLU}(x) = \max(0,\ x)$$



2 模型训练算法描述

2.1.1 随机梯度下降法（固定学习率）

分析可知，超平面优化问题即找到最佳的参数组合(ω^*,b^*)，使得对数误差 log loss 最小，

由随机梯度下降法求解最佳参数的算法步骤为：

算法随机梯度下降(SGD)在第 k 个训练迭代的更新

Require: 学习率α_k

Require: 初始参数θ

while 停止准则未满足 do

从训练集中采包含 m 个样本$\{x^{(1)},...,x^{(m)}\}$的小批量，其中$x^{(i)}$对应目标为$y^{(i)}$。

计算梯度估计: $\hat{g}\leftarrow+\frac{1}{m}\nabla_\theta\sum_i L(f(x^{(i)};\theta),y^{(i)})$

应用更新: $\theta\leftarrow\theta-\alpha\hat{g}$

end

问题：

我们发现，使用固定学习率的随机梯度下降算法存在一定问题,对于某些参数，

通过算法已经优化到了极小值附近，但是有的参数仍然有着很大的梯度。如果学习率太小，则梯度很大的参数会有一个很慢的收敛速度；如果学习率太大，则参数可能会出现不稳定的情况。

策略：

使用自适应学习率算法，即对每个参与训练的参数设置不同的学习率，在整个学习过程中通过一些算法自动适应这些参数的学习率。

2.1.2 随机梯度下降法（带 Nesterov 动量）

动量算法描述

Require: 学习率 ϵ，动量参数 α

Require: 初始参数 $\boldsymbol{\theta}$，初始速度 $\boldsymbol{v}$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度估计：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

计算速度更新：$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while

原始 SGD 每次更新的步长只是梯度乘以学习率；现在，步长还取决于历史梯度序列的大小和排列；当许多连续的梯度指向相同的方向时，步长会被不断增大；

带 Nesterov 动量算法描述

Require: 全局学习率 ϵ，衰减速率 ρ，动量系数 α

Require: 初始参数 $\boldsymbol{\theta}$，初始参数 $\boldsymbol{v}$

初始化累积变量 $\boldsymbol{r} = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算临时更新：$\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$

累积梯度：$\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$

计算速度更新：$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$ （$\frac{1}{\sqrt{\boldsymbol{r}}}$ 逐元素应用）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while

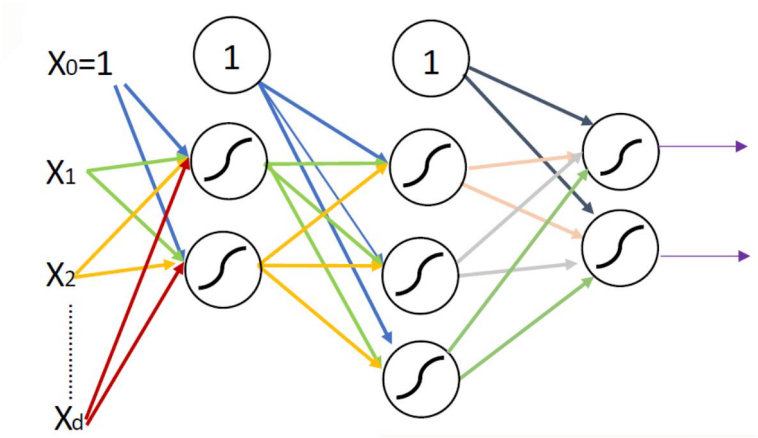
Nesterov 动量可以解释为往标准动量方法中添加了一个修正因子，带 Nesterov 动量的梯度下降算法把梯度计算放在对参数施加当前速度之后。这个“提前量”的设计让算法有了对前方环境“预判”的能力。

2.2 神经网络权重优化

BP 算法

算法基本思想：

BP 算法（误差逆传播算法）利用输出后的误差来估计输出层的前一层的误差，再用这个误差估计更前一层的误差，如此一层一层地反传下去，从而获得所有其它各层的误差估计。再根据估计的误差值对网络进行训练，直至得到一个较好的网络模型。



算法原理（在假设有 t 层神经元的情况下）：

①针对输入网络的数据集 $\{(\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),\cdots(\boldsymbol{x}_m,\boldsymbol{y}_m)\}$，根据原定的初始网络模型经过激活函数计算出初始的输出值 $\hat{y}_n = g(z_n^{(t)})$，并计算初始误差 E_m。在本题中损失函数使用对数误差（交叉熵损失）。

$$E_m = \sum_{i=1}^{m} y_i \log h(x_i) + (1-y_i)\log[1-h(x_i)]\text{。}$$

②采用广义的感知机学习规则，基于梯度下降的策略，BP 算法以目标的负梯度方向对权重进行调整：

$$\omega_{kn}^{(t)} = \omega_{kn}^{(t)} - \eta\delta_n^{(t)}x_k^{(t-1)}$$

其中 η 为学习率，代表每一轮迭代的学习步长，一般在 0-1 之间。

对于 δ_n^t，有：

$$\delta_n^t = (\hat{y}_n^m - y_n^m) \bullet y_n^m \bullet (1 - \hat{y}_n^m) \bullet x_k^{(t-1)}$$

③从第 t 层神经元的权重不断前推，直至修改所有层的权重，同时动态修改 δ_n^t 值

经过整一个阶段后，误差（损失值）会有所下降。

④不断重复②③两步，使误差（损失值）不断下降至所规定的误差范围 ε 内。

3 模型训练过程

3.1 超参数确定过程

模型训练过程中的超参数包括：

Table: Hyper-parameter

max_iter	最大迭代次数
solver	权重优化的求解算法，包括{‘lbfgs’, ‘sgd, ‘adamm’}三种算法
activation	激活函数，包括{‘ReLU’， ‘identity’…}函数
early-stopping	当验证评分没有改善时，是否使用提前停止来终止培训。
alpha	L2（欧氏距离）正则化惩罚参数
learning_rate_init	初始学习率，控制权重更新步长
tol	权重优化容忍度，容差优化
momentum	梯度下降更新的动量
Hidden_layer_sizes	隐层神经层尺寸

我们对数据设置多种超参数组合进行训练：

Hyper-parameter	
max_iter	[200, 500, 1000, 1500, 2000]
solver	{‘sgd’， ‘lgbfs’}
activation	{ ‘Identity’，‘Sigmoid’，‘Tanh’，‘Relu’ }

early_stopping	{‘True’, ‘False’}
alpha	10^{k}, $(-5 \le k \le 1)$
learning_rate_init	$k*10^{-3}$, $(1 \le k \le 15)$
tol	10^{k}, $(-8 \le k \le -4)$
momentum	$0.7 + k*10^{-2}$, $(0 \le k \le 20)$
Hidden_layer_sizes	$[k,\ (3 \le k \le 13)]$, $[(m,\ n),\ (3 \le m \le 13,\ 2 \le n \le 4)]$

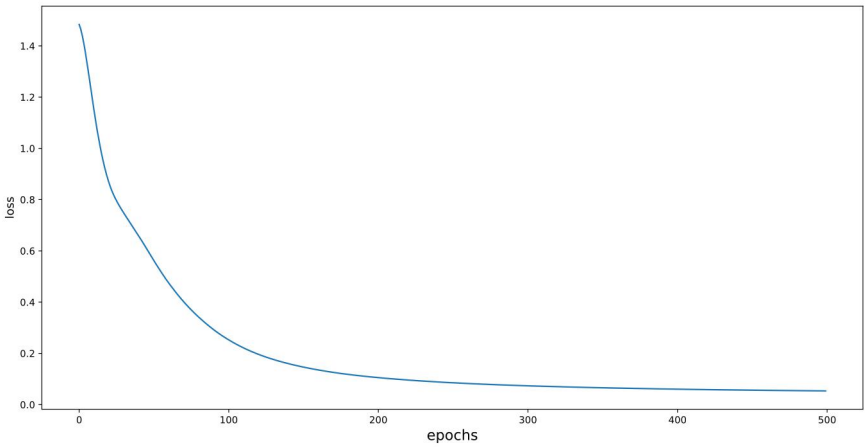
对比训练得出模型结果后选择出最佳超参数组合：

Table: Best Hyper-parameters

Hyper-parameter	max_iter	solver	activation	early-stopping	Alpha
	500	Sgd	‘ReLU’	False	10^{-5}
Hyper-parameter	tol	momentum	Hidden_layer_sizes	learning_rate_init	
	10^{-8}	0.9	（5, 3）	0.007	

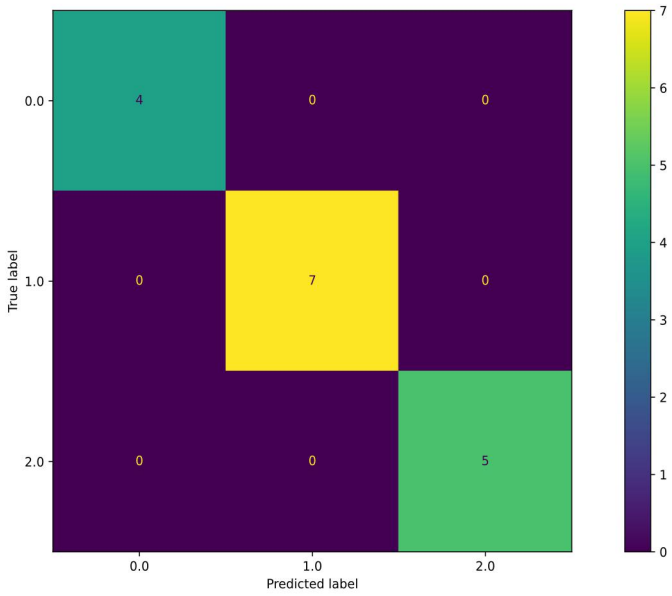
3.2 模型训练结果

得到最佳训练超参数组合后，对数据进行训练，得到对数损失log loss随迭代次数增加变化情况如下：



可以看出，随迭代次数增加，模型的log loss减小，在迭代次数接近500次时，log loss变化减小。

根据数据分类的预测值及真实值情况画出对应的混淆矩阵：（当数据在对角线上，即数据的预测值等于真实值时，说明分类效果好）



由混淆矩阵可以看出，训练出的模型对数据的分类时，预测值与真实值基本一致，说明分类器分类性能优。

用同样的参数训练模型，对数据进行五折交叉验证处理划分训练集以及测试集，分别测试得到五次五折训练结果的模型对应accuracy值

$$accuracy = \frac{TP + TN}{Total}$$

五次五折交叉验证结果：

Table： Result

	Train Log Loss Mean	Test Log Loss Mean	Train Accuracy Mean	Test Accuracy Mean	Estimator Loss
Test1	0.04412	0.10298	0.99375	0.9875	0.04419
Test2	0.04824	0.09775	0.99375	0.9875	0.04830
Test3	0.04902	0.08316	0.99375	0.9875	0.04908
Test4	0.04507	0.11518	0.99375	0.9750	0.04514
Test5	0.04526	0.09121	0.9922	0.9875	0.04533

Best accuracy = 1.0

5 模型训练收获

超参数设置时，要先通过多种超参数的组合来训练数据，对比训练得到的模型对应的拟合程度之后选择最优的模型对应的超参数组合进行验证。其中在设置超参数 alpha 时要按数据量设置，和学习率大小也有关，学习率不能太大也不能太小，sgd 在 fine-tuned 精细调整情况下效果比 adam 好，但是 adam 不用太细调超参数就能获得比较好的效果，网络尺寸好坏只能靠交叉验证看出来，但是大尺寸训练成本远高于小尺寸而效果未必会改善太多，所以要做出权衡。