

# 中山大學

## 本科实验报告

课程名称:	人工智能
实验名称:	神经网络图像识别
专业名称:	保密管理
学生姓名:	武自厚
学生学号:	20336014
实验地点:	东校园实验楼 D502
实验成绩:	
报告时间:	2022 年 6 月 4 日

## 一、 实验要求

在 MNIST 手写数字数据集完成图像分类任务, 在测试集完成测试, 计算准确率. 完成房价预测回归训练, 画出训练损失函数曲线图、预测函数图.

设计多层感知机 (至少两层全连接层), 并使用激活函数, 选择合适的损失函数, 并手动推导参数更新公式, 利用训练集完成网络训练, 并在测试集上计算准确率.

## 二、 实验过程

### 1. 神经网络算法原理

**神经网络**采用了仿生学的思想, 通过模拟生物神经网络的结构和功能来实现建模. 此次实验将完成多层感知机, 输入数据将通过若干**全连接层** (分别配有**激活函数层**) 之后得到输出数据. 训练数据 (以及测试数据) 将通过**正向传播**得到相应的分类, 而预测值与真实值的差异可以使用**损失函数**估算, 最后通过**反向传播**得到各个变量的梯度, 按照一定的**学习率**减去梯度完成学习.

#### (1) 全连接层

该层会接受一个向量  $\mathbf{x}$  作为输入, 并将其按照  $n$  组权重  $\mathbf{W}$  计算出  $n$  维向量  $\mathbf{y}$  作为输出, 有时输出向量还会加上偏置向量  $\mathbf{b}$  进行调整. 很容易可以得到:

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

#### (2) 激活函数层

该层会接收一个向量输入, 并对单个元素分别应用激活函数再输出. 激活函数是非线性函数, 用于在以线性计算为主的神经网络中引入非线性因素, 便于模拟非线性函数.

本次试验中用到的激活函数:

- ReLU 函数: 很简单的非线性函数, 能够最大程度上突出非线性特征.

$$\text{ReLU}(x_i) = \max\{0, x_i\}$$

- softmax 函数: 由于本次试验是分类问题, 因此采用了这个可以用来扩大特征的激活函数.

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_k \exp(x_k)}$$

#### (3) 损失函数

与最后的 softmax 层相对应, 损失函数采用交叉熵. 对于  $n$  维的真实值向量  $\mathbf{y}$  以及预测值向量  $\hat{\mathbf{y}}$ , 损失函数为:

$$L = -\frac{1}{n} \sum_{i=1}^n y_i \ln \hat{y}_i$$

## 2. 公式推导

### (1) 全连接层的正向传播

在实际训练的过程中会将很多训练数据拼接为一个矩阵输入，所以全连接层具有输入数据  $\mathbf{X}_{p \times m}$ ，输出数据  $\mathbf{Y}_{p \times n}$ ，权值矩阵  $\mathbf{W}_{m \times n}$  以及偏置向量  $\mathbf{b}_{1 \times n}$ 。

从单个数据正向传播公式容易推广出多个数据的公式：

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{1}_{p \times n}\mathbf{b}$$

### (2) 激活层的正向传播

由于 ReLU 是一元函数，所以容易推出：

$$\mathbf{Y} = [y_{i,j}] = [\max\{0, x_{i,j}\}]$$

同理可以推出 softmax 函数：

$$\hat{\mathbf{Y}} = [\hat{y}_{i,j}] = \left[ \frac{\exp(x_{i,j})}{\sum_k \exp(x_{i,k})} \right]$$

不过在实际编程中  $\exp(x_{i,j})$  可能会过大造成“指数发散”的问题，因此在实际使用时一般令其减去最大值再用指数函数处理：

$$\hat{\mathbf{Y}} = [y_{i,j}] = \left[ \frac{\exp(x_{i,j} - \max_t x_{i,t})}{\sum_k \exp(x_{i,k} - \max_t x_{i,t})} \right]$$

### (3) softmax 层反向传播

对于真实值矩阵  $\mathbf{Y}$  以及预测值矩阵  $\hat{\mathbf{Y}}$  (型号都为  $m \times n$ ) 由损失函数公式容易得到：

$$\frac{\partial L}{\partial \hat{y}_{i,j}} = -\frac{y_{i,j}}{n \hat{y}_{i,j}}$$

再已知 softmax 层输入矩阵  $\mathbf{X}$  根据 softmax 函数求偏导：

$$\frac{\partial \hat{y}_{i,k}}{\partial x_{i,j}} = \frac{\partial}{\partial x_{i,j}} \frac{\exp(x_{i,k})}{\sum_l \exp(x_{i,l})} = \begin{cases} -\hat{y}_{i,j} \hat{y}_{i,k}, & k \neq i \\ -\hat{y}_{i,j} \hat{y}_{i,k} + \hat{y}_{i,j}, & k = i \end{cases}$$

因此可以得出：

$$\frac{\partial L}{\partial x_{i,j}} = \sum_k \frac{\partial L}{\partial \hat{y}_{i,k}} \frac{\partial \hat{y}_{i,k}}{\partial x_{i,j}} = \frac{1}{n} \sum_k \frac{y_{i,k}}{\hat{y}_{i,k}} \hat{y}_{i,k} \hat{y}_{i,j} - \frac{1}{n} y_{i,j} = \frac{1}{n} (\sum_k y_{i,k} \hat{y}_{i,j} - y_{i,j})$$

由于真实值同一行中只有一个元素是 1，其余都是 0，所以  $\forall k: y_{i,k}$  只有一个为 1。所以，

$$\frac{\partial L}{\partial x_{i,j}} = \frac{1}{n} (\hat{y}_{i,j} - y_{i,j})$$

扩展为梯度：

$$\nabla_{\mathbf{X}} L = \left[ \frac{\partial L}{\partial x_{i,j}} \right] = \frac{1}{n} (\hat{\mathbf{Y}} - \mathbf{Y})$$

## (4) ReLU 层的反向传播

由于 ReLU 是分段函数, 所以:

$$\frac{\partial y}{\partial x} = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \implies \frac{\partial L}{\partial x_{i,j}} = \begin{cases} 0, & x < 0 \\ \frac{\partial L}{\partial y_{i,j}}, & x \geq 0 \end{cases} \implies \nabla_{\mathbf{X}} L = \begin{cases} 0, & x < 0 \\ \nabla_{\mathbf{Y}} L, & x \geq 0 \end{cases}$$

注: ReLU 在  $x = 0$  处并不连续, 为了方便计算此处取右导数.

## (5) 全连接层的反向传播

由公式可以得出: 已知  $y_{i,j} = \sum_k x_{i,k} w_{k,j} + b_j$ :

首先是对权重的梯度,

$$\nabla_{\mathbf{W}} L = \left[ \frac{\partial L}{\partial w_{k,j}} \right] = \left[ \sum_i \frac{\partial L}{\partial y_{i,k}} \frac{\partial y_{i,k}}{\partial w_{k,j}} \right] = \left[ \sum_i \frac{\partial L}{\partial y_{i,k}} x_{i,k} \right] = \left[ \sum_i x_{k,i}^T \frac{\partial L}{\partial y_{i,k}} \right] = \mathbf{X}^T (\nabla_{\mathbf{Y}} L)$$

其次是对偏置的梯度:

$$\nabla_{\mathbf{b}} L = \left[ \frac{\partial L}{\partial b_{1,j}} \right]^T = \left[ \sum_i \frac{\partial L}{\partial y_{i,j}} \right]^T = \left[ \sum_i 1_{1,i} \frac{\partial L}{\partial y_{i,j}} \right]^T = ((\mathbf{1}_{1 \times i})(\nabla_{\mathbf{Y}} L))^T$$

注: 其实就是对  $\nabla_{\mathbf{Y}} L$  进行逐行求和得到的向量

最后是对输入的梯度:

$$\nabla_{\mathbf{X}} L = \left[ \frac{\partial L}{\partial x_{i,j}} \right] = \left[ \sum_j \frac{\partial L}{\partial y_{i,j}} \frac{\partial y_{i,j}}{\partial x_{i,k}} \right] = \left[ \sum_j \frac{\partial L}{\partial y_{i,j}} w_{k,j} \right] = \left[ \sum_j \frac{\partial L}{\partial y_{i,j}} w_{j,k}^T \right] = (\nabla_{\mathbf{Y}} L) \mathbf{W}^T$$

## (6) SME 损失函数

前向传播很简单:

$$L_{\text{SME}} = \frac{1}{p} (\hat{\mathbf{Y}} - \mathbf{Y})^2$$

其中  $p$  是矩阵中元素的个数.

反向传播:

$$\nabla_{\mathbf{Y}} = \frac{1}{p} \left[ \frac{\partial L}{\partial y_{i,j}} \right] = \frac{2}{p} [y_{i,j} - \hat{y}_{i,j}] = \frac{2}{p} (\mathbf{Y} - \hat{\mathbf{Y}})$$

## (7) 参数学习

给定学习率  $\eta$  每次反向传播之后全连接层的权重矩阵和偏置向量都会进行学习:

$$\mathbf{W}' = \mathbf{W} - \eta \nabla_{\mathbf{W}} L, \quad \mathbf{b}' = \mathbf{b} - \eta \nabla_{\mathbf{b}} L$$

### 三、 实验结果

#### 1. 分类问题

此问题中数据来源为手写数字图片生成的灰度矩阵, 目标为将手写的数字分类为 “0” 到 “9”.

对于该问题采用四层全连接层, 前三层采用 ReLU 激活, 输出层采用 softmax 激活. 采用与 softmax 函数匹配的交叉熵作为损失函数. 学习率设置为 0.005, 每次训练都随机从训练集中抽取 100 个数据进行训练以防止过拟合.

经过几次测试, 分类准确率平均达到 96% 以上, 损失函数随训练次数变化情况如图所示:

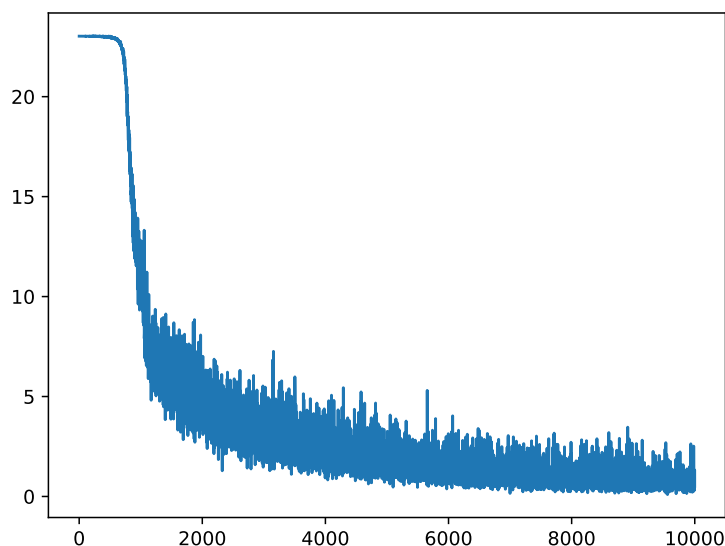


图 1: 交叉熵损失函数随训练次数的变化情况

#### 2. 回归问题

此问题的数据来自于一个房屋面积、房间数量以及房价的表格, 根据这些数据回归拟合房价关于房屋面积以及房间数量的二元函数.

对于该问题采用两层全连接层, 第一层采用 ReLU 激活, 第二层直接输出. 采用 SME 函数作为损失函数. 学习率设置为 0.01, 每次训练都随机抽取 10 个数据.

以下是损失函数下降情况以及拟合函数的图像.

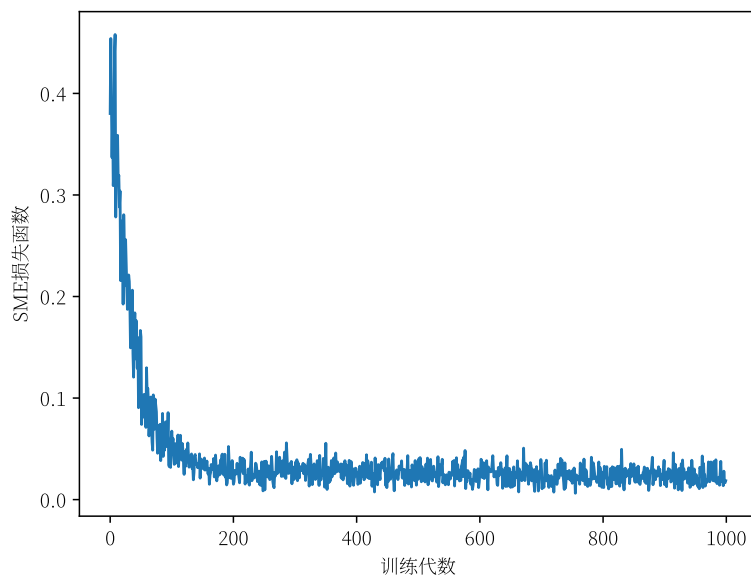


图 2: SME 损失函数随训练次数的变化情况

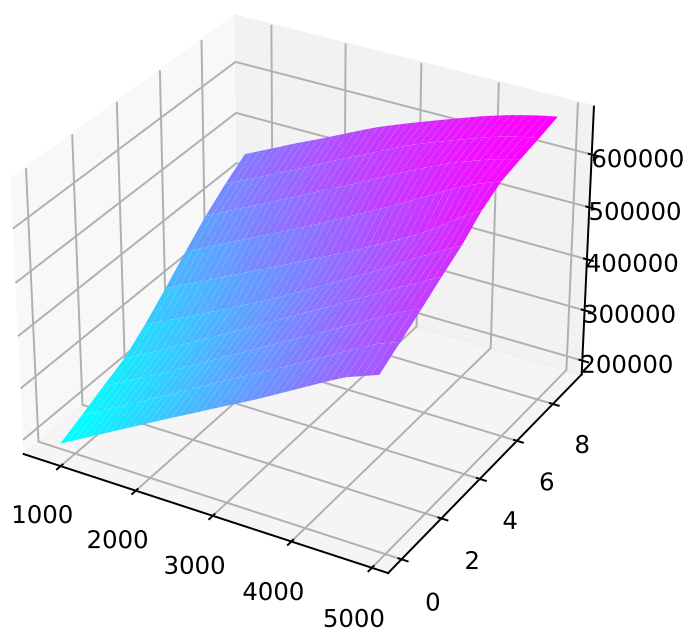


图 3: 房价回归预测函数示意图