暨南大學
JINAN UNIVERSITY

本科实验报告

课程名称：机器学习与神经网络

课程编号：08060324

小组成员：郑泽琪 高彬彬

小组成员：陆盛权 曾晓灿

学院：信息科学技术学院

系：计算机科学系

专业：计算机科学与技术

指导教师：李展

教师单位：计算机科学系

开课时间：2018~ 2019 学年度 第二学期

暨南大学教务处

2019 年 06 月 07 日

暨南大学本科实验报告专用纸

课程名称 机器学习与神经网络 成绩评定

实验项目名称 基于 Keras 的波士顿房价预测 指导教师 李展

实验项目编号 08060324 实验项目类型 综合

一、 背景及研究意义

本实验利用从 1978 年开始统计的美国波士顿共 506 个房屋的资料数据来预测平均房价；波士顿房价数据集是统计了 20 世纪 70 年代中期波士顿郊区房价的中位数，同时统计了当时教区部分的犯罪率、房产税等共计 13 个指标，涵盖了用地情况、教育、人种、收入、环保、犯罪等多个方面。通过该数据集，我们的目标是找到每个指标与房价的关系。

数据集有 506 条记录，我们将其划分成 333 条记录的的训练集和 173 条记录的测试集。每个记录的特征取值范围各不相同。比如，犯罪率范围在[0,1]，平均房间数范围在[3,9]，房价范围在[10000，50000]等等。

利用马萨诸塞州波士顿郊区的房屋信息数据训练和测试一个模型，并对模型的性能和预测能力进行测试。通过该数据训练后的好的模型可以被用来对房屋做特定预测---尤其是对房屋的价值。对于房地产经纪等人的日常工作来说，这样的预测模型被证明非常有价值。

这是一个经典的多变量的回归预测问题，且预测结果为连续值。我们考虑使用 Keras 深度学习框架来搭建并训练神经网络完成此课题。

二、 数据及处理

1、了解数据

1.1 数据来源及维度含义

此项目的数据集源自 UCI 机器学习库，由 kaggle 的 Predicting Boston Housing Prices 赛题目所提供。波士顿的房屋数据是在 1978 年收集的，506 个条目中的每一个都代表了马萨诸塞州波士顿各郊区 14 个房屋特征的汇总数据。14 种特征的数据分为十三个特征指标和一个目标值房价 Medv，每个特征指标含义分别如下：

特征指标	具体含义
CRIM	人均犯罪率
ZN	25,000平方英尺以上民用土地的比例
INDUS	城镇非零售业商用土地比例
CHAS	是否邻近查尔斯河，1是邻近，0是不邻近
NOX	一氧化氮浓度（千万分之一）
RM	住宅的平均房间数
AGE	自住且建于1940年前的房屋比例
DIS	到5个波士顿就业中心的加权距离
RAD	到高速公路的便捷度指数
TAX	每万元的房产税率
PTRATIO	城镇学生教师比例
BLACK	城镇中黑人比例
LSTAT	低收入人群比例
MEDV	自住房价格中位数，单位是千元

运行下面的代码单元以加载 Boston Housing 数据集，以及该可视化数据所需的一些必要的 Python 库。如果打印出了 features 的首部，则成功加载了数据集。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
# Seperate the features and target value from the train.csv
# features 十三维的特征
# 目标值是房价： prices
data = pd.read_csv('../Row_Data/train.csv')
prices = data['medv']
features = data.drop('medv', axis = 1)
test_features = pd.read_csv('../Row_Data/test.csv')
features.head()
```

由此成功加载了训练数据集 train.csv 和待测试数据集 test.csv.

训练数据：train.csv (333 x 15)：

ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
1	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
2	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
5	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
11	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
12	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
13	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
14	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
15	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2

测试数据：test.csv (173 x 14)：

ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
3	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
6	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21
8	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15
9	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93
10	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1
18	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67
20	0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21	390.95	11.28
25	0.75026	0	8.14	0	0.538	5.924	94.1	4.3996	4	307	21	394.33	16.3
26	0.84054	0	8.14	0	0.538	5.599	85.7	4.4546	4	307	21	303.42	16.51

1.3 数据可视化

对部分数据特征进行数据可视化，通过图像我们可以更加方便地了解和分析数据。同时观察是否存在错误记录的数据，比如出现不应该为负数得特征或者严重偏离常规值的数据。代码块如下：

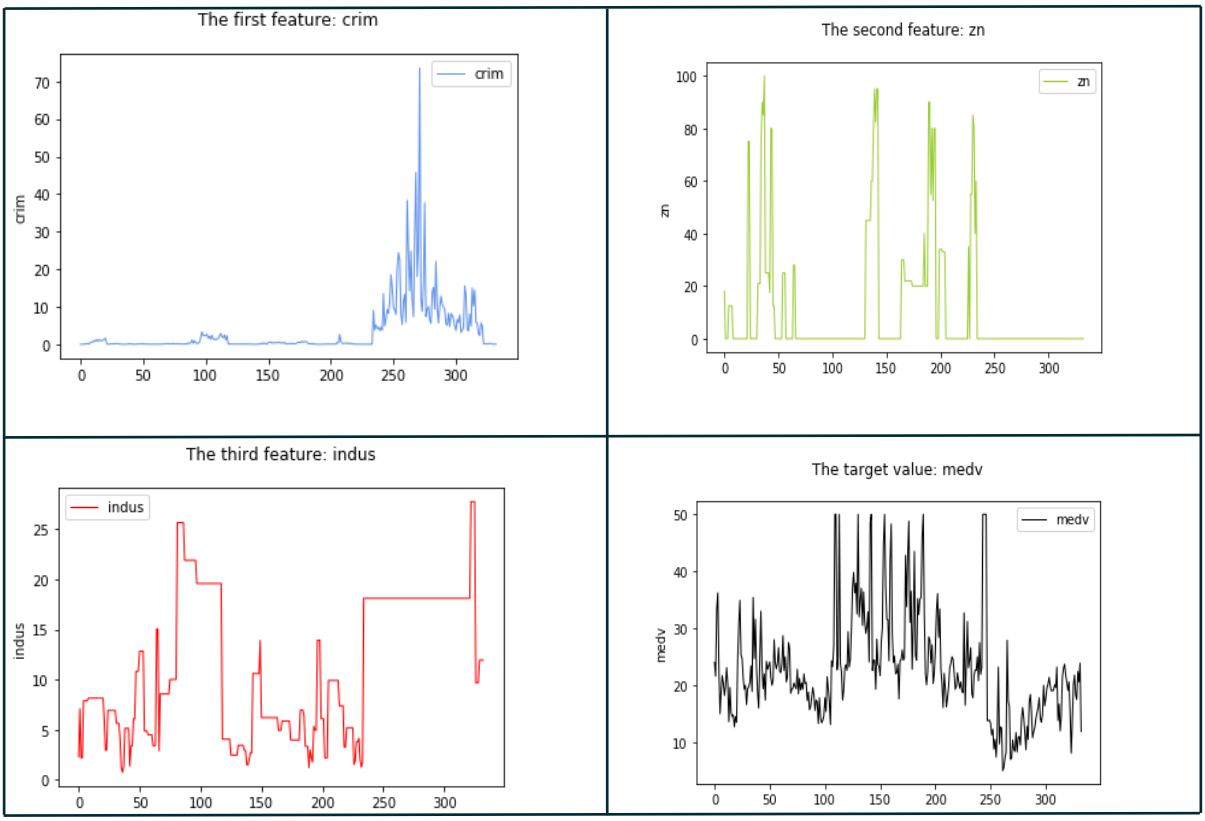
```
# 画出 features.crim 图像
plt.figure()
plt.plot(features.crim, color="cornflowerblue", label="crim", linewidth=1)
plt.ylabel("crim")
plt.title("The first feature: crim\n")
plt.legend()
plt.show()

# 画出 features.zn 图像
plt.figure()
plt.plot(features.zn, color="yellowgreen", label="zn", linewidth=1)
plt.ylabel("zn")
plt.title("The second feature: zn\n")
plt.legend()
plt.show()

# 画出 features.indus 图像
plt.figure()
plt.plot(features.indus, color="red", label="indus", linewidth=1)
plt.ylabel("indus")
plt.title("The third feature: indus\n")
plt.legend()
plt.show()

# 画出 values 图像
plt.figure()
plt.plot(prices, color="black", label="medv", linewidth=1)
plt.ylabel("medv")
plt.title("The target value: medv\n")
plt.legend()
plt.show()
```

所得结果如下：



注：其他维度的可视化过程类似，为简化实验报告将不再赘述。

2、 数据预处理

该数据集的数据是 1978 年开始在美国波士顿进行统计收集的，由于年代较为久远，可能会存在错误数据或者不合理数据，所以在训练前要进行数据的预处理，以避免这些不合理数据导致训练的结果受到影响。

2.1 缺失值处理

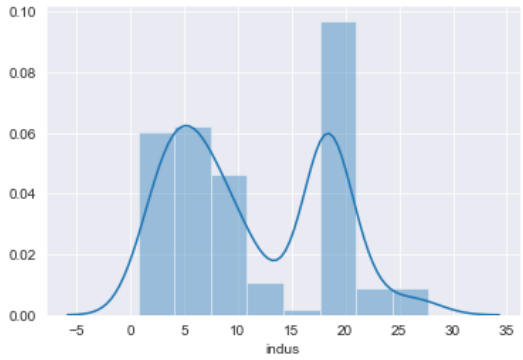
```
# 缺失值处理
count = 0
for i in features:
    count += features[i].isnull().sum()
    print(features[i].isnull().sum())
print("空值个数为: ", count)
if(count == 0):
    print("无缺失值")
data.info()
```

结果：空值个数为 0，即没有缺失值。

2.2 异常值处理

```
# 异常值处理
import seaborn as sns
sns.set_style('darkgrid')
sns.distplot(features.indus)
sns.distplot(features.zn)
sns.distplot(features.black)
sns.distplot(features.crim)
```

处理完成后，结果如下：（以 indus 列数据为例）



笔记：Seaborn 是基于 matplotlib 的 Python 可视化库。 它提供了一个高级界面来绘制有吸引力的统计图形。Seaborn 其实是在 matplotlib 的基础上进行了更高级的 API 封装，从而使得作图更加容易，不需要经过大量的调整就能使你的图变得精致。但应强调的是，应该把 Seaborn 视为 matplotlib 的补充，而不是替代物。从上图可以看到与使用 matplotlib 作的直方图最大的区别在于有一条密度曲线（KDE），可以通过设置参数去掉这条默认的曲线。通过分布可以发现，indus 基本分布在 1 和 27 之间，indus 指城镇非零售业商用土地比例。通过 Seaborn 可以更为丰富地制图。

2. 2 PCA 数据降维

使用 PCA（主成分分析 Principal Component Analysis）对数据进行降维，将数据从 13 维度降至 10 维度。

笔记：PCA 是一种在尽可能减少信息损失的情况下找到某种方式降低数据维度的方法。通常来说，是把原始数据的特征空间（高维度）投影到一个小一点的子空间（低维度）里去，并尽可能表达得很好。通过减少特征空间的维度，抽取子空间的数据来最好的表达我们的数据，从而减少参数估计的误差。

对训练数据集 PCA 操作如下，如果代码块执行后打印出前三行，说明 PCA 操作运行成功：

```
# 用 PCA 从 13 维度降至 10 维度
# 先去掉ID
features = features.drop('ID', axis = 1)
features
from sklearn.decomposition import PCA
pca = PCA(n_components = 10)
newfeatures = pca.fit_transform(features)
# 查看前 3 行
newfeatures[:3]

pca = PCA(n_components = 10)
test_newfeatures = pca.fit_transform(test_features)
# 查看前 3 行
test_newfeatures[:3]
```

所得结果前三行如下：

```
array([[-119.75386175,   -7.83917519,   -3.16445772,    6.19441862,
          -4.07667334,    5.37050041,   -0.74070188,    5.42597745,
           1.75017944,   -0.27564841],
       [-169.50276671,    5.83441619,  -30.51712067,    1.48637459,
          -0.76489749,    3.13517314,   -0.23434761,    1.03205475,
          -0.50255232,    1.39595369],
       [-191.19936724,   12.90321916,   -5.63890319,  -18.76641735,
          -1.22476344,    7.09343101,   -0.37123921,    1.00983758,
          -1.19008723,    1.03688806]])
```

注：由于训练数据和测试需要保持一致，所以测试集也要进行 PCA 降维操作。

待测试集 PCA 操作之后所得结果前三行：

```
array([[-2.85930687e+02, -1.10470514e+02, -1.20487981e+01,
         -1.74238056e+00,  1.26762752e+01,  2.22742100e+00,
         -6.80047276e+00, -3.87880878e-01, -5.19409555e-01,
          2.21114340e-01],
       [-2.99800734e+02, -9.89771125e+01, -2.54333106e+00,
         -3.11094455e+00,  1.36107221e+01,  3.40242673e+00,
         -5.58048307e+00, -5.25331323e+00, -1.55450473e-02,
          8.66134949e-01],
       [-2.30243717e+02, -1.36657567e+02, -4.98453689e+01,
         -9.62168300e+00, -1.90663105e+01, -1.01370445e+00,
          3.94229280e+00, -3.59280627e+00, -4.76769364e-02,
         -2.23420842e+00]])
```

2. 3 标准化

因为每个指标的取值范围区别较大，所以需要对数据进行标准化，所谓数据的标准化就是将数据“去量纲化”，简单来说就是要把这些数据指标的单位的影响去掉，使得他们都在同一个量纲上——即无量纲上进行讨论，这样数据之间的相互关系还在，但其他的一些干扰因素就少了很多。这里我们采用的标准化方法如下：

$$\hat{x} = \frac{x - \overline{x}}{s}$$

- 其中:
- $\overline{x}$：**平均数**
- s：**标准差**

标准化操作如下：

```
# 自定义一个最小最大规范化的函数
def minmax_normalization(data):
    xs_max = np.max(data, axis=0)
    xs_min = np.min(data, axis=0)
    xs = (1 - 0) * (data - xs_min) / (xs_max - xs_min) + 0
    return xs
```

标准化后的结果如下：（截取部分数据）

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	0.000000	0.000000	0.180	0.058148	0.0	0.314815	0.583656	0.629787	0.308996	0.000000	0.206501	0.313953	1.0000
1	0.001980	0.000285	0.000	0.234444	0.0	0.172840	0.553834	0.775532	0.400545	0.043478	0.103250	0.604651	1.0000
2	0.005941	0.000354	0.000	0.053333	0.0	0.150206	0.665569	0.423404	0.514848	0.086957	0.065010	0.709302	0.9942
3	0.007921	0.000853	0.000	0.053333	0.0	0.150206	0.694423	0.512766	0.514848	0.086957	0.065010	0.709302	1.0000
4	0.011881	0.001115	0.125	0.264074	0.0	0.286008	0.474632	0.644681	0.462482	0.173913	0.235182	0.302326	0.9966

注：同样，测试数据集也要进行标准化：

```
# 将 test_features 传入上面的规范化函数
# 先去掉 test_features 的 ID 列
test_features = test_features.drop('ID', axis = 1)
# 规范化得到的数据存放在 m_n_newfeatures 中
m_n_newfeature = minmax_normalization(test_features);
m_n_newfeature.head()
```

结果如下：（截取部分数据）

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0.000152	0.000000	0.242302	0.0	0.160752	0.656398	0.599382	0.346078	0.043478	0.104962	0.553191	0.989737	0.0650
0.000180	0.000000	0.063050	0.0	0.137787	0.493753	0.574665	0.446102	0.086957	0.066794	0.648936	0.992990	0.1013
0.001470	0.131579	0.271628	0.0	0.275574	0.438173	0.959835	0.435899	0.173913	0.236641	0.276596	1.000000	0.5309
0.002219	0.131579	0.271628	0.0	0.275574	0.321629	1.000000	0.447919	0.173913	0.236641	0.276596	0.974104	0.8631
0.001756	0.131579	0.271628	0.0	0.275574	0.401982	0.854789	0.494501	0.173913	0.236641	0.276596	0.974305	0.4677

2. 4 将 train.csv 划分为训练集和验证集：

进行 DNN 网络的训练，需要做数据分割与重排，因此将波士顿房屋测试数据集 train.csv 划分成训练数据集和验证数据集两个子集，其中训练数据集和验证数据集按 8:2 的比例进行划分。在这个过程中，数据也会被重排列，以消除数据集中由于顺序而产生的偏差。训练数据集用于训练神经网络模型，而验证数据集用于结果验证。

笔记：将数据集分为训练数据集和测试数据集所带来的好处：一方面对我们创建的模型进行训练拟合，并测试以检测我们所创建模型的可行性并及时调整，也可以防止过拟合；另一方面，验证数据集为我们的模型训练过程所没有见过的，更具代表性，能及时地验证我们当前所得模型的好坏。相关代码块如下：

```
# Split the dataset as traing set and testing set
m_n_features = m_n_features.drop('ID', axis = 1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(m_n_features, prices, test_size = 0.2, random_state = 1)
# random_state=1:不会随机划分
X_train.head()
```

三、 训练神经网络

3. 1 定义模型评估方法

对于模型准确性的的评估，常见描述误差的函数有 RMSE（均方根误差）、MSE（均方误差）、MAE（平均绝对误差）、SD（标准差）等，对于房价预测等此类回归问题，最常用的损失函数是均方误差（MSE）和均方根误差(RMSE)。

a) MSE（均方误差 Mean Squared Error）：

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(y_i - \hat{y}_i)^2$$

MSE 是指参数估计值与参数真值之差平方的期望值，MSE 可以评价数据的变化程度，MSE 的值越小，说明预测模型描述实验数据具有更好的精确度。

b) RMSE（均方根误差 Root Mean Squared Error）：

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE 是均方误差的算术平方根，采用 RMSE 的值作为评估依据，RMSE 的值越接近 0，代表网络模型训练效果越好；RMSE 的值越接近 1，则代表预测值与真实值之间地差距越大，即网络模型训练效果越差。

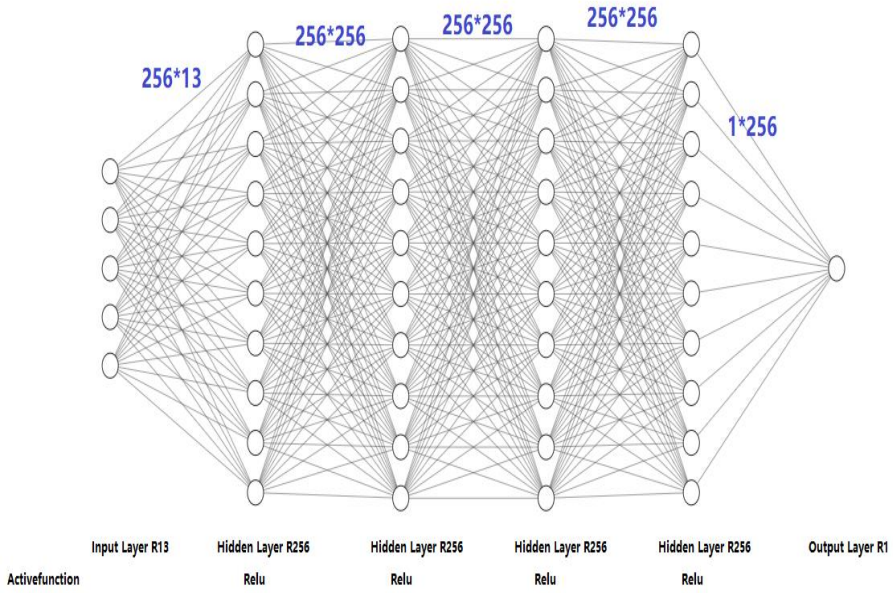
3. 2 DNN 网络结构

我们搭建了一个 DNN 网络，可分为输入层、4 个隐含层和输出层，输入层有 13 个节点，对应 13 维度的 feature，第一个隐含层是 256 x 13 的结构，另外三个隐含层都是 256 x 256 的结构，激活函数都是 Relu 函数，而输出层只有一个节点，就是我们所需的输出目标房价，因此没有激活函数。

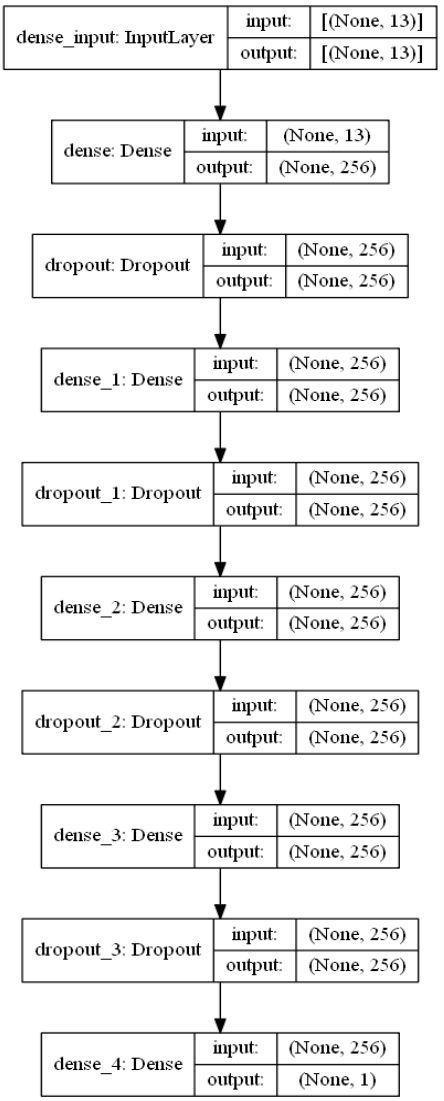
网络各层情况：

Layer	Remark	Activitation Function
Input	$13 nodes$	
Hidden 1	$256 \times 13 nodes$	Relu
Hidden 2	$256 \times 256 nodes$	Relu
Hidden 3	$256 \times 256 nodes$	Relu
Hidden 4	$256 \times 256 nodes$	Relu
Output	$1 \times 256 nodes$	

神经网络可视化：



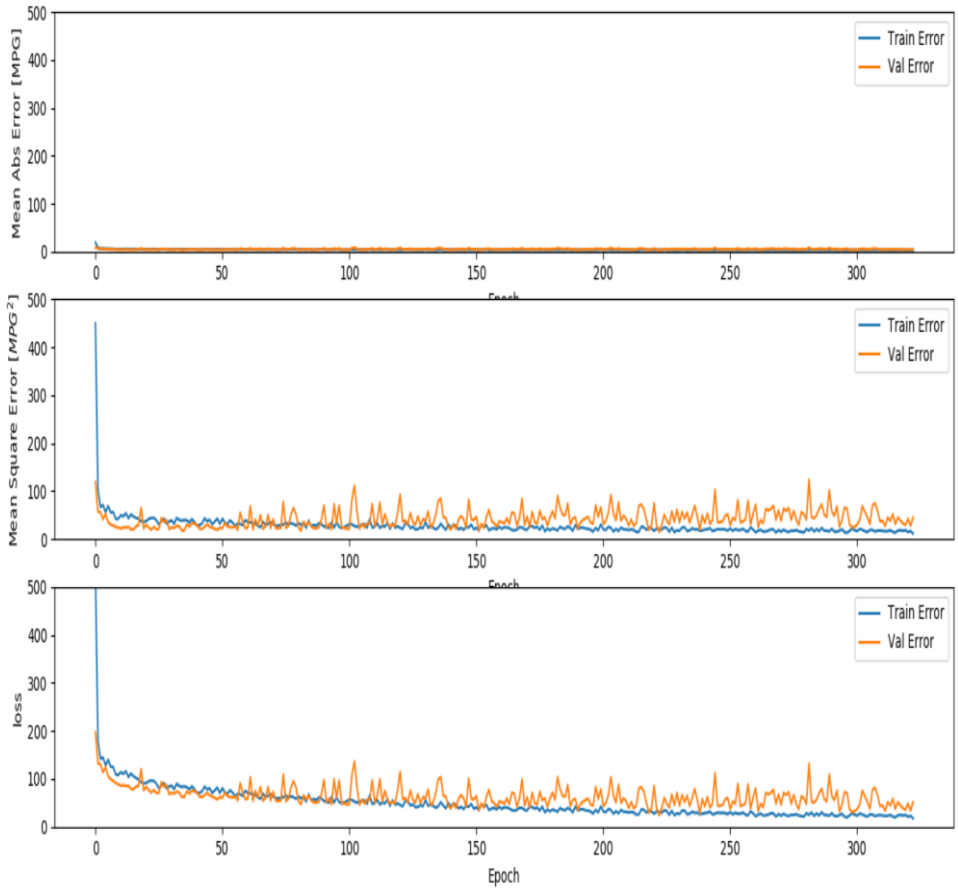
网络层次结构图：



3.3 神经网络超参数的选择

超参数	选择值
DropOut	**0.5**
正则化参数	**0.1**
EarlyStop	通过**EarlyStop**来提前结束训练
Loss Function	**RMSE**
Batch Size	**50**

四、 训练结果



从误差曲线的图像可以看出，由于训练集数据量较小，所以随着 Epoch 的增加，训练误差还是存在波动，但总体上误差还是在不断地减小，最后达到一定值后稳定下来。

注：因为这是 Kaggle 平台上地一个比赛，test.csv 中的正确结果，即待预测值并不知道，因此我们将模型训练结果提交至 Kaggle 获取最终成绩：

All Successful Selected

Submission and Description	Private Score	Public Score
output.csv 2 minutes ago by EricTseng97 add submission details	6.67829	6.21440
output1.csv 2 minutes ago by EricTseng97 add submission details	8.17292	8.25905
output2.csv 3 minutes ago by EricTseng97 add submission details	7.28540	8.10258
output2.csv 4 minutes ago by EricTseng97 add submission details	7.28540	8.10258

output3.csv 4 minutes ago by EricTseng97 add submission details	10.09035	8.10258
output4.csv 4 minutes ago by EricTseng97 add submission details	12.35913	10.94965
output5.csv 5 minutes ago by EricTseng97 add submission details	13.22701	18.46576
output6.csv 5 minutes ago by EricTseng97	16.63672	20.99854

在 Kaggle 上，网络模型的得分(分数越低说明模型越好)，从一开始的 16.636，经过一步步对网络参数进行调整，模型训练效果越来越好，Kaggle 上的得分逐渐降低。最好的一次结果是：即 RMSE = 4.45220

Name	Submitted	Wait time	Execution time	Score
output.csv	just now	0 seconds	0 seconds	4.45220

Complete

五、小组分工：

郑泽琪：数据处理，讨论，训练模型，制作 PPT，写文档

高彬彬：训练模型，讨论，分析结果

陆盛权 & 曾晓灿：写文档，讨论，PPT

六、 总结与体会

（1）通过完成这样一个课程设计，我们对课上学习的理论内容进行实践，更好地理解了神经网络并能够使用神经网络来进行具体问题的回归预测；同时了解了一个完整的 Kaggle 比赛流程，并且对数据预处理，数据降维，数据标准化，以及构建神经网络中的损失函数，激活函数，等都有了更深刻的理解；

（2）对深度学习框架 Keras 有了初步的了解和学习。当时为什么选择 Keras 主要是基于以下考虑：Keras 是为人类而非机器设计的 API；Keras 遵循减少认知困难的最佳实践：它提供一致且简单的 API，它将常见用例所需的用户操作数量降至最低，并且在用户错误时提供清晰和可操作的反馈；最重要是的是，tf.keras 作为 Keras API 可以与 TensorFlow 工作流无缝集成。

（3）组员之间通过 Github 进行协同合作，基本掌握了 Github 的常用操作，并且最后将完整的流程以及代码上传至 Github。在制作上台展示的 PPT 时，为了使得 PPT 更为美观，在很多地方使用了 Markdown 语言和 Latex 编辑表格以及公式，

可以说在 PPT 的制作上也花费了很大的功夫。

（4）同时，这个课程设计还存在一些不足的地方，比如我们只是用 PCA 这种方法对数据进行降维，并且发现降维之后，在设定同样的超参数下，训练出的神经网络对测试数据的预测结果并不能够在 Kaggle 上取得很好的成绩，因此在最终的代码中我们取消了对数据的 PCA 处理。其原因一方面可能是由于 333 x 13 的测试数据不够多，且 13 维度相对于图像上千甚至上万的像素来讲微不足道。