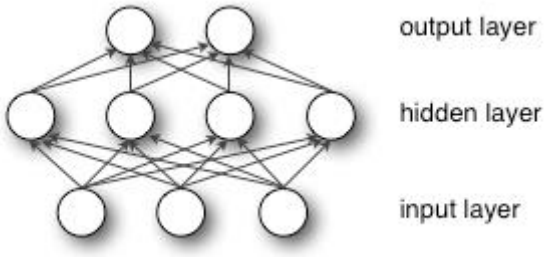
Hmoework 6: MLP

——宋悦溪 & 郑丽珊

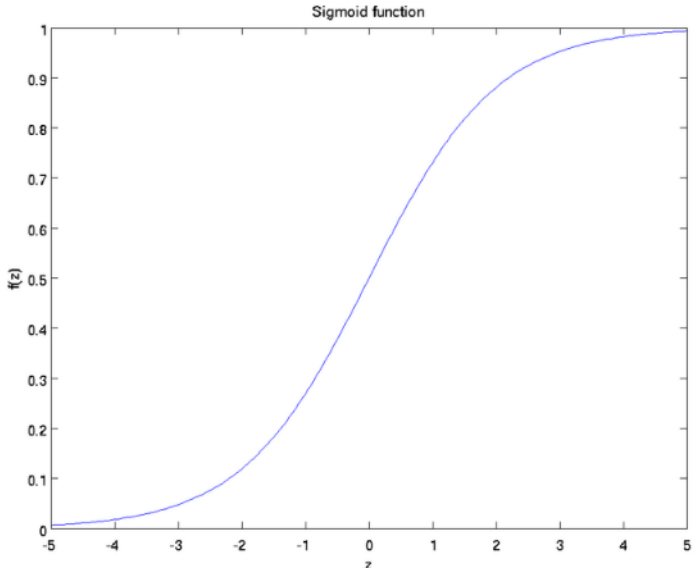
神经网络是一种模拟人脑的神经网络系统，以期能够实现类人工智能的机器学习技术。神经元模型是一个包含输入、输出与计算功能的模型。将前一个神经元的输出作为后一个神经元的输入，得到的输出再作为后一个神经元的输入，不断叠加，就构成了多层感知器 MLP（Multi-Layer Perceptron），它是一种前向结构的人工神经网络。

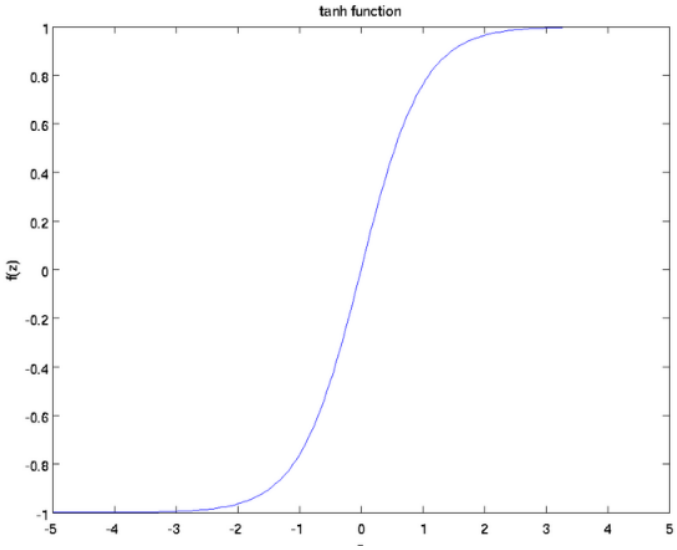
MLP 可以被看做是一个有向图，由多个节点层组成，每一层全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元。MLP 映射一组输入向量到一组输出向量。除了输入输出层，它中间可以有多个隐层，最简单的 MLP 只含一个隐层，即三层的结构，如下图：

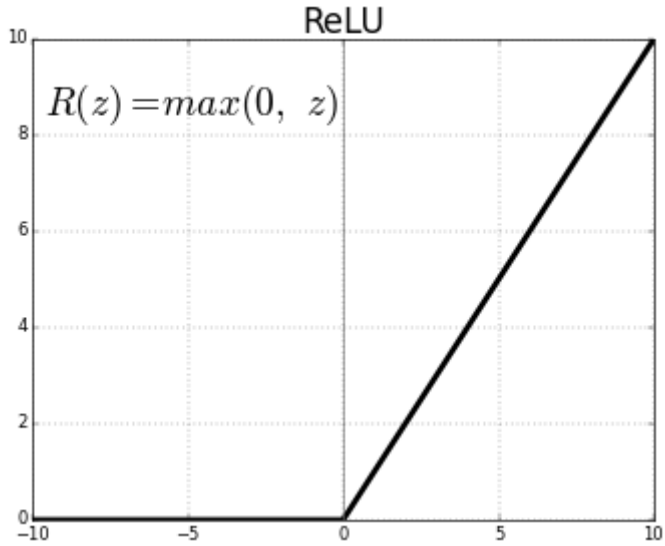


上图中，最底层是输入层，输入一个 n 维向量，就有 n 个神经元。

中间是隐藏层，它与输入层是全连接的，假设输入层用向量 X 表示，则隐藏层的输出就是 f(W1X+b1)，其中 W1 是权重（也叫连接系数），b1 是偏置，函数 f 可以是常用的 sigmoid 函数或者 tanh 函数以及 ReLU 函数。以下分别是 sigmoid，tanh 及 ReLU 函数的图像。







需要注意的是 tanh 函数是 sigmoid 函数的一种特殊变体，它的取值范围为[-1,1]，而不是 sigmoid 函数的[0,1]。

最上面一层是输出层，隐藏层到输出层可以看成是一个多类别的逻辑回归，也就是 softmax 回归，所以输出层的输出就是 softmax(W2X1+b2)，X1 表示隐藏层的输出 f(W1X+b1)。

MLP 的优点是它可以用来学习非线性模型，此外它能使用 partial_fit 进行实时学习。但是 MLP 的缺点是有隐藏层的 MLP 包含一个非凸性损失函数，存在超过一个最小值，所以不同的随机初始权重可能导致不同验证精确度 ；MLP 要求调整一系列超参数，比如隐藏神经元，隐藏层的个数以及迭代的次数；以及 MLP 对特征缩放比较敏感。

神经网络是智能计算领域最伟大的发明之一，它模仿了人类大脑的神经元，用于解决分类问题和进行数据预测。在大数据和机器学习盛行的时代背景下，我们可以建立多层感知器神经网络模型（MLP）学习，使用学习完毕的模型用于评估或预测，得到相应的结果，用来解决实际问题，这是十分有意义的。在此过程中，全程使用计算机编程自动处理，相比人工分析可以极大地缩短时间，同时减轻相应的人力投入。此外更为重要的是，该方法可以避免选取特定评估

指标标准以及人为主观因素对结果的影响，使结果更具有客观性和说服力。

二、模型实践

在实验方面我们选择的房地产销售数据，“房地产估价” 是一个回归问题，设计的变量包括时间、位置、交通等，不能用常规的线性模型来拟合，我们对比测试了多层感知机模型以及常规的 ensemble 方法来比较该方法的优势。

1、数据及变量说明

（1）数据说明

数据来源：Real estate valuation data set Dataset，
加利福尼亚大学尔湾分校（2018 年 8 月 18 日），
网址：http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set
房地产估价的市场历史数据集来自台湾新北市新店区。将数据集随机分成训练数据集和测试数据集合（8:2）。

（2）变量说明

自变量：
X1 =交易日期（例如，2013.250 = 2013 年 3 月，2013.500 = 2013 年 6 月等）
X2 =房屋年龄（单位：年）
X3 =到最近地铁站的距离（单位：米）
X4 =徒步生活圈中便利店的数量（整数）
X5 =地理坐标，纬度。（单位：度）
X6 =地理坐标，经度。（单位：度）

因变量：
Y =单位面积房价（10000 新台币/平，其中 Ping 为本地单位，1 平方= 3.3 平方米）

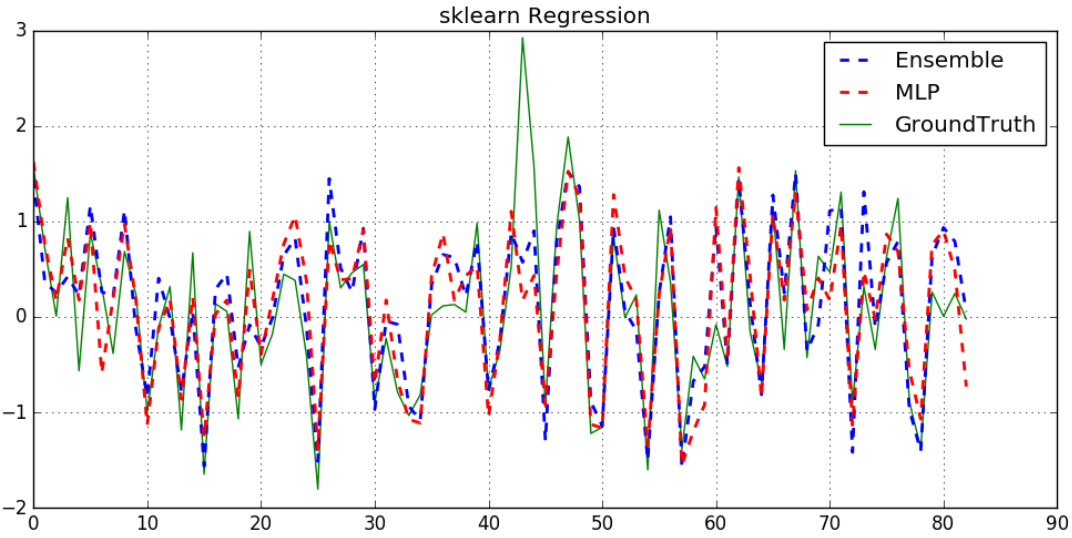
其中，我们对变量 X1 和 X2 进行预处理，具体操作如下：
属性 X1：以 2012 年为基准统一减去 2012
属性 X3：将单位转为千米统一除去 1000

2、实验结果

我们使用了具有三个隐藏层的多层感知机来预测单位面积房价，隐藏层的神经元个数分别为(3,26,28)，使用的参数见代码中的设置，我们对两种方法的进行参数选择以获取最好的模型，参数选择方法见代码。

实验中我们发现可能是由于数据量太少，所以该方法并不能显示出其良好的性能，同时在层数继续加深的情况下也不能获得更好的性能。整个实验是基于 sklearn 来实现的。

结果显示，MLP 能够很好达到比 ensemble 方法更好的效果。其中，MLP 的分数为 0.713，ensemble 的分数为 0.706。下面是预测数据的对比图。



代码：

```
# _*_ coding:utf-8 _*_
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import GradientBoostingRegressor

import pandas as pd

# read dataset
data = pd.read_excel('./RealEstateValuationDataSet.xlsx',index_col=0)
print(data.info())

# preprocess dataset
data['X1 transaction date'] = data['X1 transaction date'] - 2012
data['X3 distance to the nearest MRT station'] = data['X3 distance to the nearest MRT station'] / 1000
Y = data['Y house price of unit area']
X = data.drop(columns=['Y house price of unit area'])
x = X.values
y = Y.values

print('######################################################################')
# 随机挑选
train_x, test_x, train_y, test_y = train_test_split(x, y, train_size=0.8, random_state=33)

#数据标准化
ss_x = preprocessing.StandardScaler()
train_x = ss_x.fit_transform(train_x)
test_x = ss_x.transform(test_x)
```

```
ss_y = preprocessing.StandardScaler()
train_y = ss_y.fit_transform(train_y.reshape(-1, 1))
test_y = ss_y.transform(test_y.reshape(-1, 1))

model_mlp_best = MLPRegressor(hidden_layer_sizes=(3,26,28),  activation='relu', solver='adam', alpha=0.0001, batch_size='auto',
   learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
   random_state=1, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True,
   early_stopping=False,beta_1=0.9, beta_2=0.999, epsilon=1e-08)
model_mlp_best.fit(train_x,train_y.ravel())
mlp_score=model_mlp_best.score(test_x,test_y.ravel())
print('sklearn MLP',mlp_score) #准确率 0.713

model_gbr_best=GradientBoostingRegressor(learning_rate=0.1,max_depth=4,max_features=0.5, min_samples_leaf=14,n_estimators=30)
model_gbr_best.fit(train_x,train_y.ravel() )
gbr_score=model_gbr_best.score(test_x,test_y.ravel())
print('sklearn ensemble',gbr_score)#准确率 0.702

#使用最好的集成模型进行预测
gbr_pridict=model_gbr_best.predict(test_x)
#多层感知器
mlp_pridict=model_mlp_best.predict(test_x)

#画图
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(10, 5))
axes = fig.add_subplot(1, 1, 1)
line3,=axes.plot(range(len(test_y)), test_y, 'g',label='GroundTruth')
line1,=axes.plot(range(len(gbr_pridict)), gbr_pridict, 'b--',label='Ensemble',linewidth=2)
line2,=axes.plot(range(len(mlp_pridict)), mlp_pridict, 'r--',label='MLP',linewidth=2)
axes.grid()
fig.tight_layout()
plt.legend(handles=[line1, line2, line3])
plt.title("sklearn Regression")
plt.show()

# 多层感知器-回归模型 参数选择 3,26,28
'''
print('################################## 参 数 网 格 优 选 ##################################')
def genrator():
```

```
    for i in range(3,10):
        for j in range(3,30):
            for k in range(3,30):
                yield (i,j,k)
best_score = 0
best_list = []
num = 0
for i,j,k in genrator():
    model_mlp = MLPRegressor(hidden_layer_sizes=(i,j,k), activation='relu', solver='adam',
alpha=0.0001, batch_size='auto',
    learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
    random_state=1, tol=0.0001, verbose=False, warm_start=False, momentum=0.9,
nesterovs_momentum=True,
    early_stopping=False,beta_1=0.9, beta_2=0.999, epsilon=1e-08)
    model_mlp.fit(train_x,train_y.ravel())
    mlp_score=model_mlp.score(test_x,test_y.ravel())
    num = num+1
    if num%30 == 0:
        print('num:',num,i,j,k)
    if mlp_score>70.0:
        print('sklearn MLP',mlp_score)
        print(i,j,k)
    if mlp_score > best_score:
        best_list = [i,j,k]
        print('new_score',mlp_score,'and list',best_list)
        best_score = mlp_score
print('best_list',best_list)
'''

# 集成-回归模型 参数选择
'''
model_gbr=GradientBoostingRegressor()
model_gbr.fit(train_x,train_y.ravel())
gbr_score_disorder=model_gbr.score(test_x,test_y.ravel())
print('sklearn ensemble',gbr_score_disorder)
print('################################## 参 数 网 格 优 选
###################################')
model_gbr_GridSearch=GradientBoostingRegressor()
#设置参数池 参考 http://www.cnblogs.com/DjangoBlog/p/6201663.html
param_grid = {'n_estimators':range(20,81,10),
              'learning_rate': [0.2,0.1, 0.05, 0.02, 0.01 ],
              'max_depth': [4, 6,8],
              'min_samples_leaf': [3, 5, 9, 14],
              'max_features': [0.8,0.5,0.3, 0.1]}
```

```
#网格调参
from sklearn.model_selection import GridSearchCV
estimator = GridSearchCV(model_gbr_GridSearch,param_grid )
estimator.fit(train_x,train_y.ravel() )
print('最优调参：',estimator.best_params_)
# {'learning_rate': 0.1, 'max_depth': 4, 'max_features': 0.5, 'min_samples_leaf': 14, 'n_estimators':
30}
print('调参后得分',estimator.score(test_x, test_y.ravel()))
'''
```

参考文献：

【1】武斌, & 马晓娜. (2018). 基于多层感知器神经网络的学生校内消费评估研究. *中国教育信息化*(14), 85-89.

【2】晏福, 徐建中, & 李奉书. (2019). 混沌灰狼优化算法训练多层感知器.*电子与信息学报*(4).