神经网络学习到的是什么？（Python）

机器学习算法与Python实战 2022-01-17 09:00

以下文章来源于算法进阶，作者泳鱼

算法进阶

有AI原创干货，也有code实践！通俗又不乏深度！专注于Python机器学习、深度学习...

机器学习算法与Python实战

长期跟踪关注统计学、数据挖掘、机器学习算法、深度学习、人工智能技术与行业...

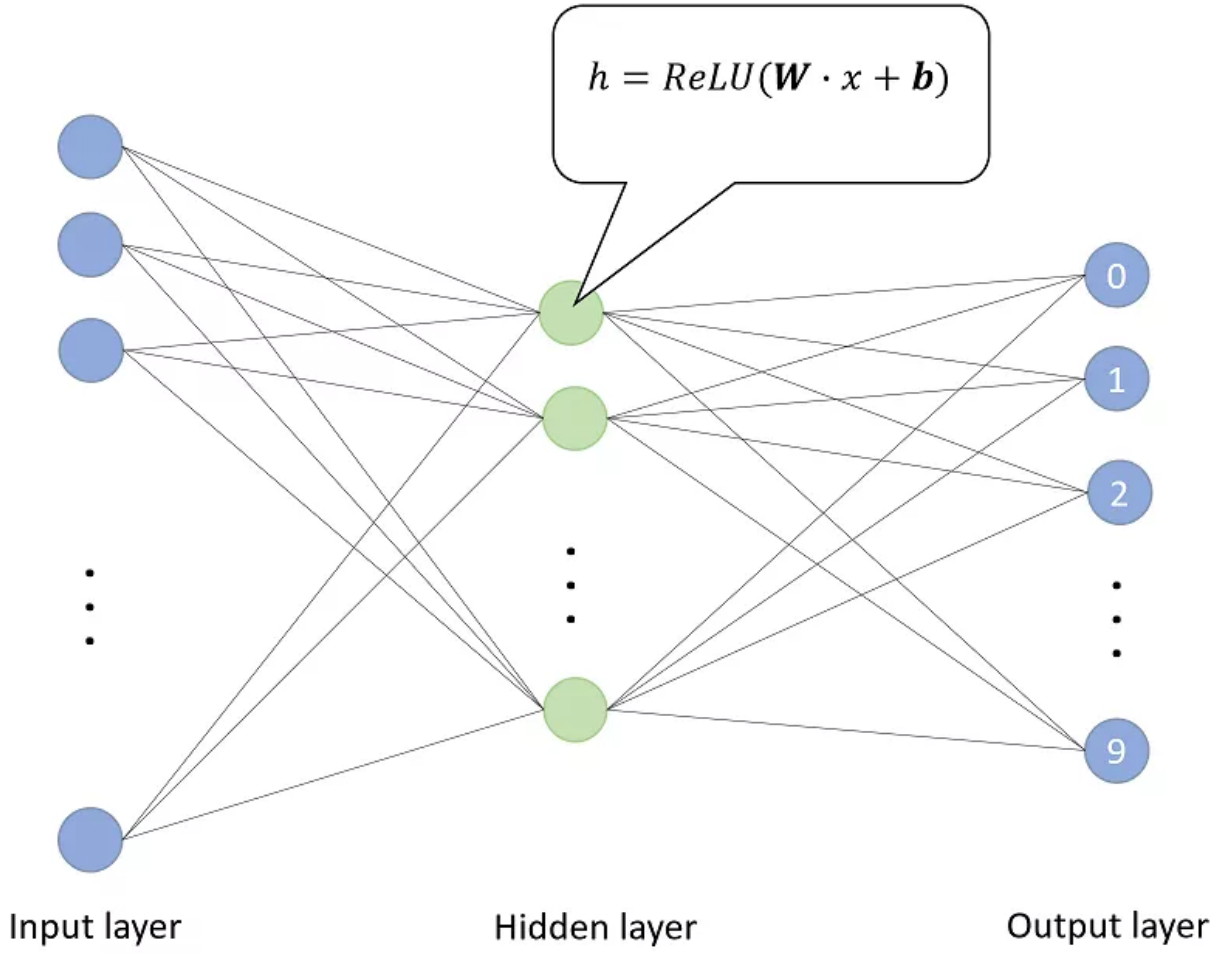
239篇原创内容

公众号

神经网络（深度学习）学习到的是什么？一个含糊的回答是，学习到的是数据的本质规律。但具体这本质规律究竟是什么呢？要回答这个问题，我们可以从神经网络的原理开始了解。

一、 神经网络的原理

神经网络学习就是一种特征的表示学习，把原始数据通过一些简单非线性的转换成为更高层次的、更加抽象的特征表达。深度网络层功能类似于“生成特征”，而宽度层类似于“记忆特征”，增加网络深度可以获得更抽象、高层次的特征，增加网络宽度可以交互出更丰富的特征。通过足够多的转换组合的特征，非常复杂的函数也可以被模型学习好。



可见神经网络学习的核心是，学习合适权重参数以对数据进行非线性转换，以提取关键特征或者决策。即模型参数控制着特征加工方法及决策。了解了神经网络的原理，我们可以结合如下项目示例，看下具体的学习的权重参数，以及如何参与抽象特征生成与决策。

二、神经网络的学习内容

2.1 简单的线性模型的学习

我们先从简单的模型入手，分析其学习的内容。像线性回归、逻辑回归可以视为单层的神经网络，它们都是广义的线性模型，可以学习输入特征到目标值的线性映射规律。

如下代码示例，以线性回归模型学习波士顿各城镇特征与房价的关系，并作出房价预测。数据是波士顿房价数据集，它是统计20世纪70年代中期波士顿郊区房价情况，有当时城镇的犯罪率、房产税等共计13个指标以及对应的房价中位数。

序号	变量名	说 明
1	CRIM	城镇人均犯罪率
2	ZN	超过25000平方英尺的住宅用地所占比例
3	INDUS	城镇非零售业的商业用地所占比例
4	CHAS	是否被Charles河流穿过（取值1：是；取值0：否）
5	NOX	一氧化碳浓度
6	RM	每栋住宅的平均房间数
7	AGE	早于1940年建成的自住房屋比例
8	DIS	到波士顿5个中心区域的加权平均距离
9	RAD	到达高速公路的便利指数
10	TAX	每10000美元的全值财产税率
11	PTRATIO	城镇中师生比例
12	B	反映城镇中的黑人比例的指标，越靠近0.63越小；B=1000*(BK-0.63) 2 ，其中BK是黑人的比例。
13	LSTAT	低收入人口的比例
14	MEDV	自住房屋房价的平均房价（单位为1000美元）

```
import pandas as pd
import numpy as np
from keras.datasets import boston_housing #导入波士顿房价数据集


(train_x, train_y), (test_x, test_y) = boston_housing.load_data()

from keras.layers import *
from keras.models import Sequential, Model
from tensorflow import random
from sklearn.metrics import  mean_squared_error

np.random.seed(0) # 随机种子
random.set_seed(0)

# 单层线性层的网络结构（也就是线性回归）：无隐藏层，由于是数值回归预测，输出层没有用激活函数：
model = Sequential()
model.add(Dense(1,use_bias=False))


model.compile(optimizer='adam', loss='mse')  # 回归预测损失mse


model.fit(train_x, train_y, epochs=1000,verbose=False)  # 训练模型
model.summary()

pred_y = model.predict(test_x)[:,0]
```

```
print("正确标签：",test_y)
print("模型预测：",pred_y )

print("实际与预测值的差异：",mean_squared_error(test_y,pred_y ))
```

通过线性回归模型学习训练集，输出测试集预测结果如下：

```
正确标签： [ 7.2 18.8 19.  27.  22.2 24.5 31.2 22.9 20.5 23.2 18.6 14.5 17.8 50.
 20.8 24.3 24.2 19.8 19.1 22.7 12.  10.2 20.  18.5 20.9 23.  27.5 30.1
  9.5 22.  21.2 14.1 33.1 23.4 20.1  7.4 15.4 23.8 20.1 24.5 33.  28.4
 14.1 46.7 32.5 29.6 28.4 19.8 20.2 25.  35.4 20.3  9.7 14.5 34.9 26.6
  7.2 50.  32.4 21.6 29.8 13.1 27.5 21.2 23.1 21.9 13.  23.2  8.1  5.6
 21.7 29.6 19.6  7.  26.4 18.9 20.9 28.1 35.4 10.2 24.3 43.1 17.6 15.4
 16.2 27.1 21.4 21.5 22.4 25.  16.6 18.6 22.  42.8 35.1 21.5 36.  21.9
 24.1 50.  26.7 25. ]
模型预测： [ 5.7905407 19.103935  22.303558  29.4067    23.599136  18.829319
 29.525255  25.243988  19.591068  21.946165  18.460178  22.358196
 19.083248  31.651276  18.31515   21.835949  21.427935  18.348156
 20.356106  27.467861  11.400978  11.061567  21.182583  16.22248
 23.640032  22.774853  30.073679  31.861498  10.1168995 20.269012
 21.275768  14.443459  33.817585  20.229748  17.763628   5.9011154
 13.665704  16.824749  17.909904  30.10262   26.207582  25.11956
 20.004086  28.846426  29.825506  21.750217  29.166645  17.947567
 25.703148  21.988855  34.022743  15.591615  11.579025  17.456547
 29.745672  25.91045   18.90581   32.470596  34.785862  24.35701
 24.635662  21.704132  16.519161  20.820751  25.131176  26.781944
 18.301697  26.270384   3.9031887 12.536167  27.62437   25.976814
 23.083326   9.177868  23.921764  20.58401   20.4917    21.885681
 28.52879    7.3671856 21.546711  31.921574  20.211231  20.363632
 21.066738  17.997557  22.190643  22.72239   25.276339  28.294157
 17.008148  22.061337  24.872896  29.96588   28.244148  20.67507
 31.640997  38.896126  20.792768  37.354774  31.28537   20.00523 ]
实际与预测值的差异： 31.943738352771547
```

分析预测的效果，用上面数值体现不太直观，如下画出实际值与预测值的曲线，可见，整体模型预测值与实际值的差异还是比较小的（模型拟合较好）。

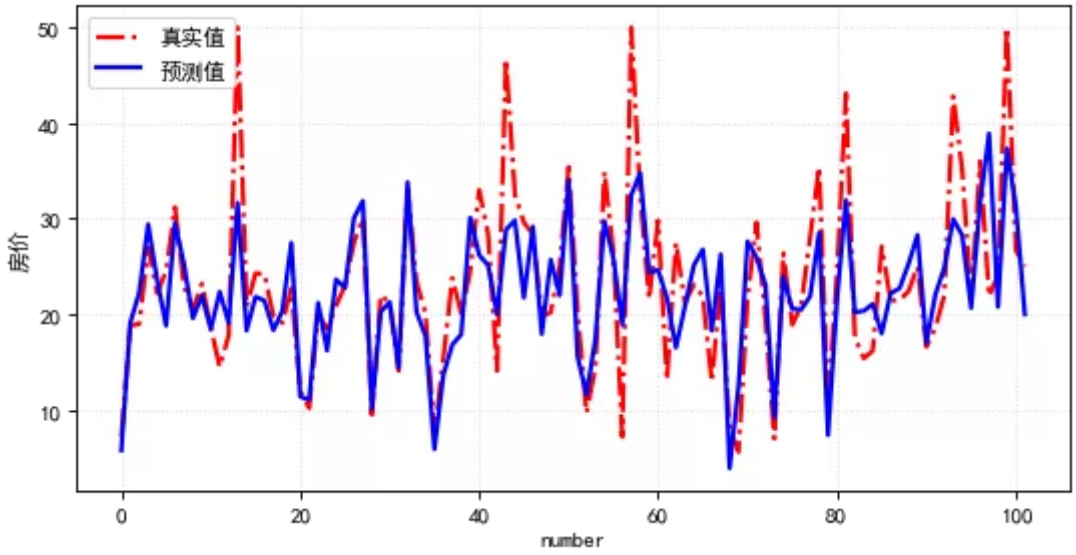
```
#绘图表示
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 设置图形大小
plt.figure(figsize=(8, 4), dpi=80)
plt.plot(range(len(test_y)), test_y, ls='-.',lw=2,c='r',label='真实值')
plt.plot(range(len(pred_y)), pred_y, ls='-',lw=2,c='b',label='预测值')
```

```
# 绘制网格
plt.grid(alpha=0.4, linestyle=':')
plt.legend()
plt.xlabel('number') #设置x轴的标签文本
plt.ylabel('房价') #设置y轴的标签文本

# 展示
plt.show()
```



回到正题，我们的单层神经网络模型（线性回归），在数据（波士顿房价）、优化目标（最小化预测误差mse）、优化算法（梯度下降）的共同配合下，从数据中学到了什么呢？

我们可以很简单地用决策函数的数学式来概括我们学习到的线性回归模型，预测y=w1*x1* + *w2*x2 + wn*xn。通过提取当前线性回归模型最终学习到的参数：

```
model.layers[0].get_weights()[0]
```

```
array([[-0.09546997],
       [ 0.09558205],
       [-0.01804003],
       [ 3.8479505 ],
       [ 1.0180658 ],
       [ 2.8623202 ],
       [ 0.05667834],
       [-0.47793597],
       [ 0.20240606],
       [-0.01002822],
       [ 0.23102441],
       [ 0.0190283 ],
       [-0.66846687]], dtype=float32)
```

将参数与对应输入特征组合一下，我们忙前忙后训练模型学到内容也就是——权重参数，它可以对输入特征进行加权求和输出预测值决策。如下决策公式，我们可以看出预测的房价和犯罪率、弱势群体比例等因素是负相关的：

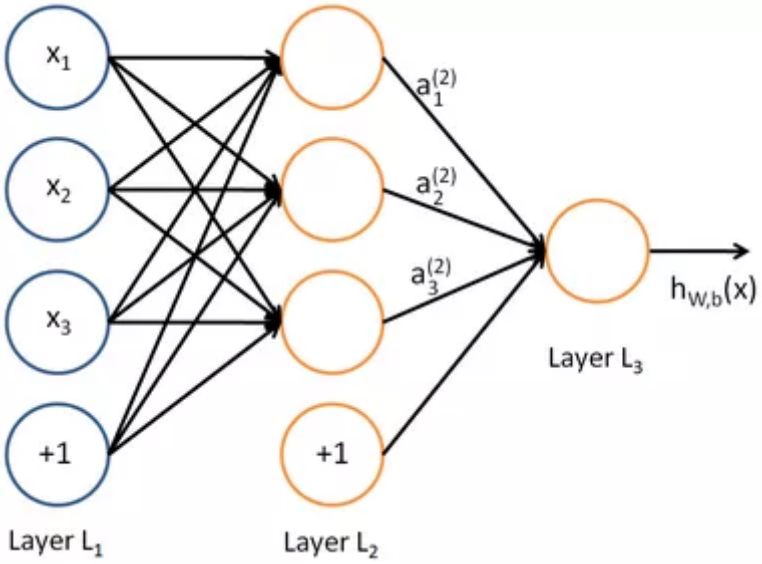
房价预测值 = [-0.09546997]*CRIM|住房所在城镇的人均犯罪率+[0.09558205]*ZN|住房用地超过 25000 平方尺的比例+[-0.01804003]*INDUS|住房所在城镇非零售商用土地的比例+[3.8479505]*CHAS|有关查理斯河的虚拟变量（如果住房位于河边则为1,否则为0 ）+[1.0180658]*NOX|一氧化氮浓度+[2.8623202]*RM|每处住房的平均房间数+[0.05667834]*AGE|建于 1940 年之前的业主自住房比例+[-0.47793597]*DIS|住房距离波士顿五大中心区域的加权距离+[0.20240606]*RAD|距离住房最近的公路入口编号+[-0.01002822]*TAX 每 10000 美元的全额财产税金额+[0.23102441]*PTRATIO|住房所在城镇的师生比例+[0.0190283]*B|1000(Bk|0.63)^2,其中 Bk 指代城镇中黑人的比例+[-0.66846687]*LSTAT|弱势群体人口所占比例

小结：单层神经网络学习到各输入特征所合适的权重值，根据权重值对输入特征进行加权求和，输出求和结果作为预测值（注：逻辑回归会在求和的结果再做sigmoid非线性转为预测概率）。

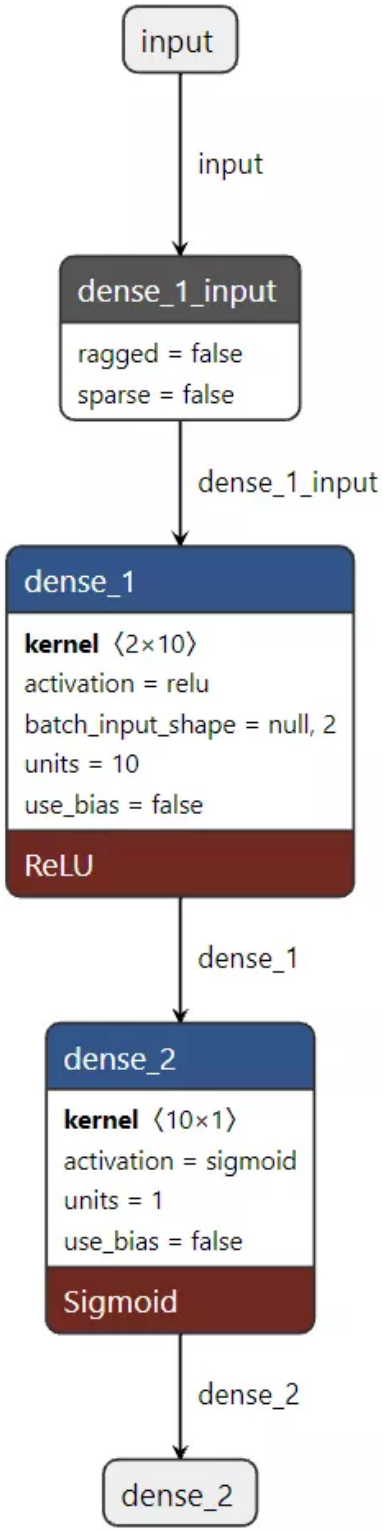
2.2 深度神经网络的学习

深度神经网络（深度学习）与单层神经网络的结构差异在于，引入了层数>=1的非线性隐藏层。从学习的角度上看，模型很像是集成学习方法——以上层的神经网络的学习的特征，输出到下一层。而这种学习方法，就可以学习到非线性转换组合的复杂特征，达到更好的拟合效果。

对于学习到的内容，他不仅仅是利用权重值控制输出决策结果--f(WX)，还有比较复杂多层次的特征交互, 这也意味着深度学习不能那么直观数学形式做表示--它是一个复杂的复合函数$f(f..f(W$X)$)$。



如下以2层的神经网络为例，继续波士顿房价的预测：



注：本可视化工具来源于https://netron.app/

```
from keras.layers import *
from keras.models import Sequential, Model
from tensorflow import random
from sklearn.metrics import  mean_squared_error

np.random.seed(0) # 随机种子
random.set_seed(0)
```

```
# 网络结构：输入层的特征维数为13，1层relu隐藏层，线性的输出层；
model = Sequential()
model.add(Dense(10, input_dim=13, activation='relu',use_bias=False))   # 隐藏层
model.add(Dense(1,use_bias=False))


model.compile(optimizer='adam', loss='mse')  # 回归预测损失mse


model.fit(train_x, train_y, epochs=1000,verbose=False)  # 训练模型
model.summary()

pred_y = model.predict(test_x)[:,0]

print("正确标签：",test_y)
print("模型预测：",pred_y )

print("实际与预测值的差异：",mean_squared_error(test_y,pred_y ))
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 10)                130
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 10
=================================================================
Total params: 140
Trainable params: 140
Non-trainable params: 0
_________________________________________________________________
正确标签： [ 7.2 18.8 19.  27.  22.2 24.5 31.2 22.9 20.5 23.2 18.6 14.5 17.8 50.
 20.8 24.3 24.2 19.8 19.1 22.7 12.  10.2 20.  18.5 20.9 23.  27.5 30.1
  9.5 22.  21.2 14.1 33.1 23.4 20.1  7.4 15.4 23.8 20.1 24.5 33.  28.4
 14.1 46.7 32.5 29.6 28.4 19.8 20.2 25.  35.4 20.3  9.7 14.5 34.9 26.6
  7.2 50.  32.4 21.6 29.8 13.1 27.5 21.2 23.1 21.9 13.  23.2  8.1  5.6
 21.7 29.6 19.6  7.  26.4 18.9 20.9 28.1 35.4 10.2 24.3 43.1 17.6 15.4
 16.2 27.1 21.4 21.5 22.4 25.  16.6 18.6 22.  42.8 35.1 21.5 36.  21.9
 24.1 50.  26.7 25. ]
模型预测： [ 7.5146527 18.902946  22.210932  30.507252  23.852276  20.170218
 29.643345  24.621902  19.098114  21.61975   18.639257  20.64216
 18.517044  34.346615  17.271315  20.432007  22.640913  18.909649
 19.200155  27.687202  10.283243  12.209352  21.246788  13.735024
 21.834309  21.950676  32.135685  28.651934  11.355534  19.833866
 21.634579  15.6553135 35.910152  21.338562  17.272541   5.5597277
 14.533072  17.419893  16.717453  28.986017  24.882612  26.159771
 19.236229  32.476814  29.60453   22.550388  29.927797  17.60147
 24.738838  22.612709  33.42963   14.817703  10.340087  16.021805
 31.227316  26.336601  18.761383  36.42126   36.888638  23.67204
 25.286743  21.435474  17.541449  21.335087  24.977837  24.534794
```

模型参数

```
18.263994  25.33643    2.898717  12.495476  28.163792  25.809155
21.601257   7.997617  25.14667   19.2655    19.681389  23.150776
29.958298   6.414797  21.666973  34.24747   16.979721  20.177526
20.036276  17.03515   21.477913  23.83873   25.758814  27.688416
16.098408  19.41324   25.091585  31.317312  30.211792  21.333977
33.506325  42.303986  22.249466  39.572956  33.190773  20.164354 ]
实际与预测值的差异： 27.401029802308354
```

mse

可见，其模型的参数（190个）远多于单层线性网络（13个）；学习的误差（27.4）小于单层线性网络模型（31.9），有着更高的复杂度和更好的学习效果。

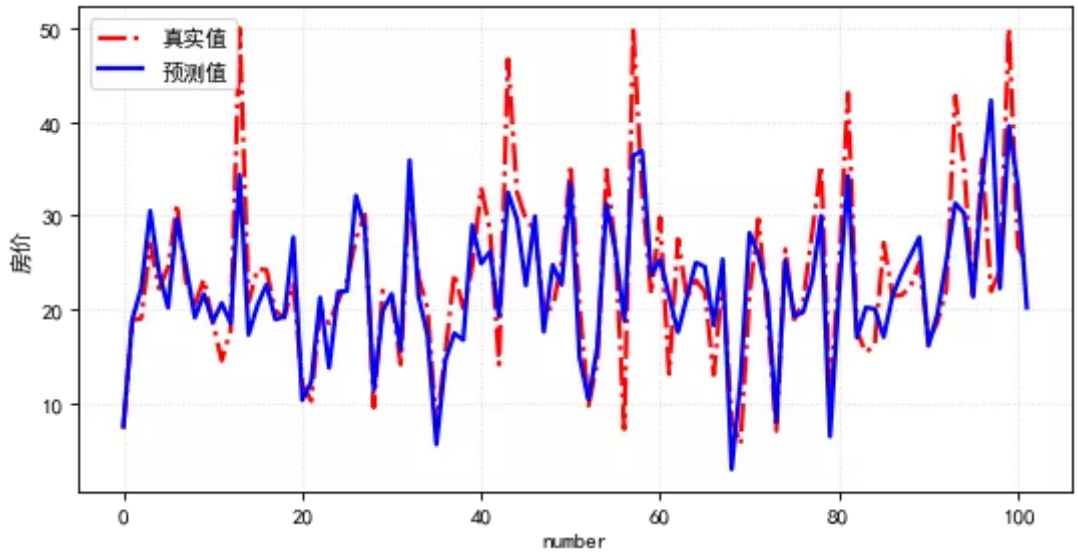
```
#绘图表示
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 设置图形大小
plt.figure(figsize=(8, 4), dpi=80)
plt.plot(range(len(test_y)), test_y, ls='-.',lw=2,c='r',label='真实值')
plt.plot(range(len(pred_y)), pred_y, ls='-',lw=2,c='b',label='预测值')

# 绘制网格
plt.grid(alpha=0.4, linestyle=':')
plt.legend()
plt.xlabel('number') #设置x轴的标签文本
plt.ylabel('房价') #设置y轴的标签文本

# 展示
plt.show()
```



回到分析深度神经网络学习的内容，这里我们输入一条样本，看看每一层神经网络的输出。

```
from numpy import exp


x0=train_x[0]
print("1、输入第一条样本x0:\n", x0)
 # 权重参数可以控制数据的特征表达再输出到下一层
w0= model.layers[0].get_weights()[0]
print("2、第一层网络的权重参数w0:\n", w0)

a0 = np.maximum(0,np.dot(w0.T, x0))
# a0可以视为第一层网络层交互出的新特征，但其特征含义是比较模糊的
print("3、经过第一层神经网络relu(w0*x0)后输出:\n",a0)
w1=model.layers[1].get_weights()[0]
print("4、第二层网络的权重参数w1:\n", w1)
 # 预测结果为w1与ao加权求和
a1 = np.dot(w1.T,a0)
print("5、经过第二层神经网络w1*ao后输出预测值:%s,实际标签值为%s"%(a1[0],train_y[0]))
```

运行代码，输出如下结果

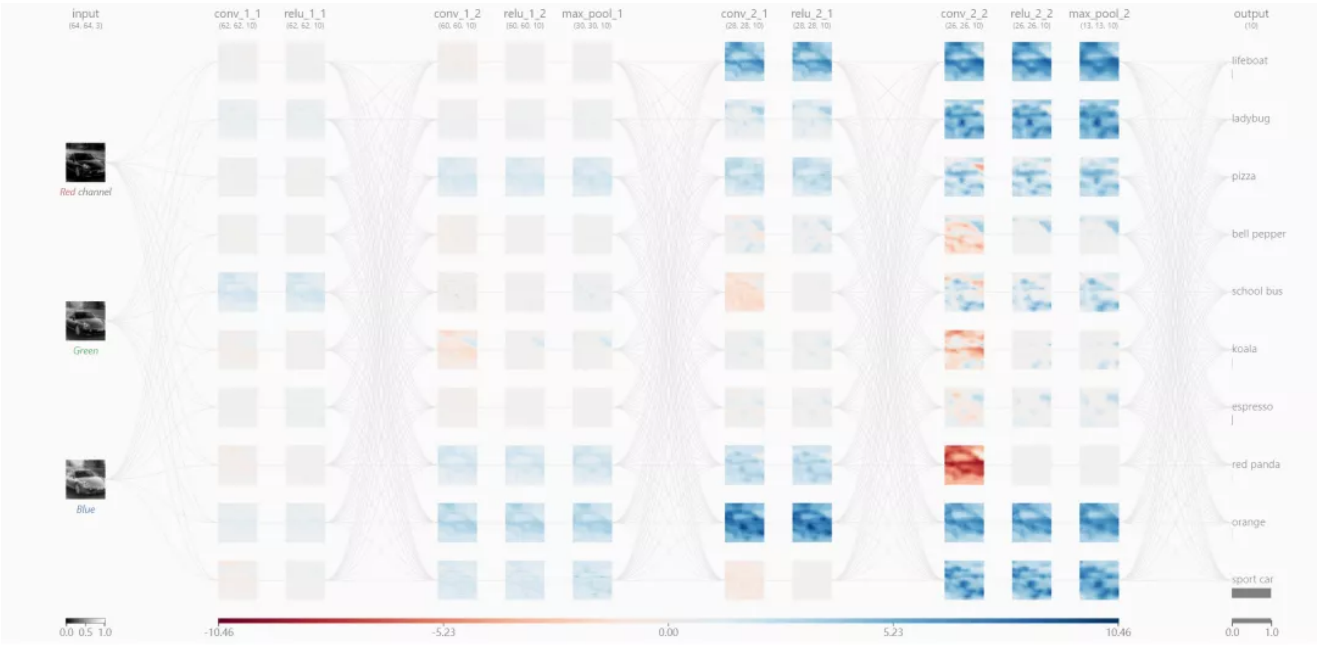
```
1、输入第一条样本x0:
 [  1.23247    0.         8.14       0.         0.538      6.142     91.7
    3.9769     4.       307.        21.       396.9       18.72    ]
2、第一层网络的权重参数w0:
 [[-0.16263646 -0.33749306  0.03615189  0.04744884 -0.12909177  0.24638008
   0.02984359  0.5088429   0.01279008 -0.3826838 ]
 [-0.08980344  0.03859001  0.07381064 -0.13815537  0.34057978 -0.03152593
   0.10607474  0.16512388  0.34557223 -0.42499015]
 [ 0.04424414 -0.5848086   0.23124236  0.25747713  0.26661664  0.2120363
   0.20227864 -0.46231204 -0.42497548  0.25715792]] ...
3、经过第一层神经网络relu(w0*x0)后输出a0:
 [ 70.09896981   0.           0.           0.         336.81448645
  68.34481658 121.285138     0.          30.16801369   0.        ]
4、第二层网络的权重参数w1:
 [[-0.3117117 ]
 [-0.3397185 ]
 [ 0.5390298 ]
 [ 0.3905266 ]
 [ 0.4214618 ]
 [-0.30084056]
 [-0.6513426 ]
```

```
[ 0.5517096 ]
[-0.0838832 ]
[-0.1673125 ]]
```

5、经过第二层神经网络w1*ao后输出预测值:18.014112015448397,实际标签值为15.2

从深度神经网络的示例可以看出，神经网络学习的内容一样是权重参数。由于非线性隐藏层的作用下，深度神经网络可以通过权重参数对数据非线性转换，交互出复杂的、高层次的特征，并利用这些特征输出决策，最终取得较好的学习效果。但是，正也因为隐藏层交互组合特征过程的复杂性，学习的权重参数在业务含义上如何决策，并不好直观解释。

对于深度神经网络的解释，常常说深度学习模型是“黑盒”，学习内容很难表示成易于解释含义的形式。在此，一方面可以借助shap等解释性的工具加于说明。另一方面，还有像深度学习处理图像识别任务，就是个天然直观地展现深度学习的过程。如下展示输入车子通过层层提取的高层次、抽象的特征，图像识别的过程。注：图像识别可视化工具来源于https://poloclub.github.io/cnn-explainer/



在神经网络学习提取层次化特征以识别图像的过程：

- 第一层，像是各种边缘探测特征的集合，在这个阶段，激活值仍然是保留了几乎原始图像的所有信息。

- 更高一层，激活值就变得进一步抽象，开始表示更高层次的内容，诸如“车轮”。有着更少的视觉表示（稀疏），也提取到了更关键特征的信息。

这和人类学习（图像识别）的过程是类似的——从具体到抽象，简单概括出物体的本质特征。就像我们看到一辆很酷的小车，

然后凭记忆将它画出来，很可能没法画出很多细节，只有抽象出来的关键特征表现，类似这样：

我们的大脑学习输入的视觉图像的抽象特征，而不相关忽略的视觉细节，提高效率的同时，学习的内容也有很强的泛化性，我们只要识别一辆车的样子，就也会辨别出不同样式的车。这也是深度神经网络学习更高层次、抽象的特征的过程。

1. **准备写本书**
2. **属实逼真，决策树可视化！**
3. **21个深度学习开源数据集汇总！**
4. **耗时一个月，做了一个纯粹的机器学习网站**
5. **用 Python 从 0 实现一个神经网络**
6. **40篇AI论文！附PDF下载，代码、视频讲解**

三连在看，月入百万👇

阅读原文

喜欢此内容的人还喜欢

神经网络的5种常见求导，附详细的公式过程

AI蜗牛车

优化神经网络训练的17种方法！

机器学习实验室