# 1.线性回归

<https://blog.csdn.net/googler_offer/article/details/78726571>

详解

# coding: utf-8

# In[ ]:

import keras

import numpy as np

import matplotlib.pyplot as plt

# Sequential按顺序构成的模型

from keras.models import Sequential

# Dense全连接层

from keras.layers import Dense

# In[ ]:

# 使用numpy生成100个随机点

x\_data = np.random.rand(100)

noise = np.random.normal(0,0.01,x\_data.shape)

y\_data = x\_data\*0.1 + 0.2 + noise

# 显示随机点

plt.scatter(x\_data,y\_data)

plt.show()

# In[ ]:

# 构建一个顺序模型

model = Sequential()

# 在模型中添加一个全连接层

model.add(Dense(units=1,input\_dim=1))

# sgd:Stochastic gradient descent，随机梯度下降法

# mse:Mean Squared Error，均方误差

model.compile(optimizer='sgd',loss='mse')

# 训练3001个批次

for step in range(3001):

# 每次训练一个批次

cost = model.train\_on\_batch(x\_data,y\_data)

# 每500个batch打印一次cost值

if step % 500 == 0:

print('cost:',cost)

# 打印权值和偏置值

W,b = model.layers[0].get\_weights()

print('W:',W,'b:',b)

# x\_data输入网络中，得到预测值y\_pred

y\_pred = model.predict(x\_data)

# 显示随机点

plt.scatter(x\_data,y\_data)

# 显示预测结果

plt.plot(x\_data,y\_pred,'r-',lw=3)

plt.show()

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 2.非线性回归

# coding: utf-8

# In[8]:

import keras

import numpy as np

import matplotlib.pyplot as plt

# Sequential按顺序构成的模型

from keras.models import Sequential

# Dense全连接层

from keras.layers import Dense,Activation

from keras.optimizers import SGD

# In[2]:

# 使用numpy生成200个随机点

x\_data = np.linspace(-0.5,0.5,200)

noise = np.random.normal(0,0.02,x\_data.shape)

y\_data = np.square(x\_data) + noise

# 显示随机点

plt.scatter(x\_data,y\_data)

plt.show()

# In[10]:

# 构建一个顺序模型

model = Sequential()

# 在模型中添加一个全连接层

# 1-10-1

model.add(Dense(units=10,input\_dim=1,activation='relu'))

# model.add(Activation('tanh'))

model.add(Dense(units=1,activation='relu'))

# model.add(Activation('tanh'))

# 定义优化算法

sgd = SGD(lr=0.3)

# sgd:Stochastic gradient descent，随机梯度下降法

# mse:Mean Squared Error，均方误差

model.compile(optimizer=sgd,loss='mse')

# 训练3001个批次

for step in range(3001):

# 每次训练一个批次

cost = model.train\_on\_batch(x\_data,y\_data)

# 每500个batch打印一次cost值

if step % 500 == 0:

print('cost:',cost)

# x\_data输入网络中，得到预测值y\_pred

y\_pred = model.predict(x\_data)

# 显示随机点

plt.scatter(x\_data,y\_data)

# 显示预测结果

plt.plot(x\_data,y\_pred,'r-',lw=3)

plt.show()

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 3.MNIST数据集分类=0.918

# coding: utf-8

# In[1]:

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

# In[3]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 创建模型，输入784个神经元，输出10个神经元

model = Sequential([

Dense(units=10,input\_dim=784,bias\_initializer='one',activation='softmax')

])

# 定义优化器

sgd = SGD(lr=0.2)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(

optimizer = sgd,

loss = 'mse',

metrics=['accuracy'],

)

# 训练模型

model.fit(x\_train,y\_train,batch\_size=32,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('accuracy',accuracy)

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 4.交叉熵=0.922

# coding: utf-8

# In[1]:

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

# In[2]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 创建模型，输入784个神经元，输出10个神经元

model = Sequential([

Dense(units=10,input\_dim=784,bias\_initializer='one',activation='softmax')

])

# 定义优化器

sgd = SGD(lr=0.2)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(

optimizer = sgd,

loss = 'categorical\_crossentropy',

metrics=['accuracy'],

)

# 训练模型

model.fit(x\_train,y\_train,batch\_size=32,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('accuracy',accuracy)

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 5.Dropout应用=0.978

# coding: utf-8

# In[3]:

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense,Dropout

from keras.optimizers import SGD

# In[6]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 创建模型

model = Sequential([

Dense(units=200,input\_dim=784,bias\_initializer='one',activation='tanh'),

Dropout(0.4),

Dense(units=100,bias\_initializer='one',activation='tanh'),

Dropout(0.4),

Dense(units=10,bias\_initializer='one',activation='softmax')

])

# 定义优化器

sgd = SGD(lr=0.2)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(

optimizer = sgd,

loss = 'categorical\_crossentropy',

metrics=['accuracy'],

)

# 训练模型

model.fit(x\_train,y\_train,batch\_size=32,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('test accuracy',accuracy)

loss,accuracy = model.evaluate(x\_train,y\_train)

print('train loss',loss)

print('train accuracy',accuracy)

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 6.正则化应用=0.985

# coding: utf-8

# In[1]:

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

from keras.regularizers import l2

# In[2]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 创建模型

model = Sequential([

Dense(units=200,input\_dim=784,bias\_initializer='one',activation='tanh',kernel\_regularizer=l2(0.0003)),

Dense(units=100,bias\_initializer='one',activation='tanh',kernel\_regularizer=l2(0.0003)),

Dense(units=10,bias\_initializer='one',activation='softmax',kernel\_regularizer=l2(0.0003))

])

# 定义优化器

sgd = SGD(lr=0.2)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(

optimizer = sgd,

loss = 'categorical\_crossentropy',

metrics=['accuracy'],

)

# 训练模型

model.fit(x\_train,y\_train,batch\_size=32,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('test accuracy',accuracy)

loss,accuracy = model.evaluate(x\_train,y\_train)

print('train loss',loss)

print('train accuracy',accuracy)

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 8.CNN应用于手写数字识别=0.9919=时间长

# coding: utf-8

# In[9]:

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense,Dropout,Convolution2D,MaxPooling2D,Flatten

from keras.optimizers import Adam

# In[10]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)->(60000,28,28,1)

x\_train = x\_train.reshape(-1,28,28,1)/255.0

x\_test = x\_test.reshape(-1,28,28,1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 定义顺序模型

model = Sequential()

# 第一个卷积层

# input\_shape 输入平面

# filters 卷积核/滤波器个数

# kernel\_size 卷积窗口大小

# strides 步长

# padding padding方式 same/valid

# activation 激活函数

model.add(Convolution2D(

input\_shape = (28,28,1),

filters = 32,

kernel\_size = 5,

strides = 1,

padding = 'same',

activation = 'relu'

))

# 第一个池化层

model.add(MaxPooling2D(

pool\_size = 2,

strides = 2,

padding = 'same',

))

# 第二个卷积层

model.add(Convolution2D(64,5,strides=1,padding='same',activation = 'relu'))

# 第二个池化层

model.add(MaxPooling2D(2,2,'same'))

# 把第二个池化层的输出扁平化为1维

model.add(Flatten())

# 第一个全连接层

model.add(Dense(1024,activation = 'relu'))

# Dropout

model.add(Dropout(0.5))

# 第二个全连接层

model.add(Dense(10,activation='softmax'))

# 定义优化器

adam = Adam(lr=1e-4)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(optimizer=adam,loss='categorical\_crossentropy',metrics=['accuracy'])

# 训练模型

model.fit(x\_train,y\_train,batch\_size=64,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('test loss',loss)

print('test accuracy',accuracy)

# <h3 align = "center">欢迎大家关注我的公众号，或者加我的微信与我交流。</h3>

# <center><img src="wx.png" alt="FAO" width="300"></center>

# In[ ]:

# 9.RNN应用=0.90

# coding: utf-8

# In[5]:

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.layers.recurrent import SimpleRNN

from keras.optimizers import Adam

# In[4]:

# 数据长度-一行有28个像素

input\_size = 28

# 序列长度-一共有28行

time\_steps = 28

# 隐藏层cell个数

cell\_size = 50

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

x\_train = x\_train/255.0

x\_test = x\_test/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)#one hot

# 创建模型

model = Sequential()

# 循环神经网络

model.add(SimpleRNN(

units = cell\_size, # 输出

input\_shape = (time\_steps,input\_size), #输入

))

# 输出层

model.add(Dense(10,activation='softmax'))

# 定义优化器

adam = Adam(lr=1e-4)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(optimizer=adam,loss='categorical\_crossentropy',metrics=['accuracy'])

# 训练模型

model.fit(x\_train,y\_train,batch\_size=64,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('test loss',loss)

print('test accuracy',accuracy)

# 10. RNN应用lstm=0.94

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.layers.recurrent import LSTM

from keras.optimizers import Adam

# In[4]:

# 数据长度-一行有28个像素

input\_size = 28

# 序列长度-一共有28行

time\_steps = 28

# 隐藏层cell个数

cell\_size = 50

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

x\_train = x\_train/255.0

x\_test = x\_test/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)#one hot

# 创建模型

model = Sequential()

# 循环神经网络

model.add(LSTM(

units = cell\_size, # 输出

input\_shape = (time\_steps,input\_size), #输入

))

# 输出层

model.add(Dense(10,activation='softmax'))

# 定义优化器

adam = Adam(lr=1e-4)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(optimizer=adam,loss='categorical\_crossentropy',metrics=['accuracy'])

# 训练模型

model.fit(x\_train,y\_train,batch\_size=64,epochs=10)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('test loss',loss)

print('test accuracy',accuracy)

# 11.模型保存

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 创建模型，输入784个神经元，输出10个神经元

model = Sequential([

Dense(units=10,input\_dim=784,bias\_initializer='one',activation='softmax')

])

# 定义优化器

sgd = SGD(lr=0.2)

# 定义优化器，loss function，训练过程中计算准确率

model.compile(

optimizer = sgd,

loss = 'mse',

metrics=['accuracy'],

)

# 训练模型

model.fit(x\_train,y\_train,batch\_size=64,epochs=5)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('accuracy',accuracy)

# 保存模型

model.save('model.h5') # HDF5文件，pip install h5py

# 12.模型载入

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

from keras.models import load\_model

# In[2]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)

print('x\_shape:',x\_train.shape)

# (60000)

print('y\_shape:',y\_train.shape)

# (60000,28,28)->(60000,784)

x\_train = x\_train.reshape(x\_train.shape[0],-1)/255.0

x\_test = x\_test.reshape(x\_test.shape[0],-1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 载入模型

model = load\_model('model.h5')

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('accuracy',accuracy)

# In[3]:

# 训练模型

model.fit(x\_train,y\_train,batch\_size=64,epochs=2)

# 评估模型

loss,accuracy = model.evaluate(x\_test,y\_test)

print('\ntest loss',loss)

print('accuracy',accuracy)

# In[4]:

# 保存参数，载入参数

model.save\_weights('my\_model\_weights.h5')

model.load\_weights('my\_model\_weights.h5')

# 保存网络结构，载入网络结构

from keras.models import model\_from\_json

json\_string = model.to\_json()

model = model\_from\_json(json\_string)

# In[5]:

print(json\_string)

# 13.绘制网络结构

import numpy as np

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense,Dropout,Convolution2D,MaxPooling2D,Flatten

from keras.optimizers import Adam

from keras.utils.vis\_utils import plot\_model

import matplotlib.pyplot as plt

# install pydot and graphviz

# In[23]:

# 载入数据

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

# (60000,28,28)->(60000,28,28,1)

x\_train = x\_train.reshape(-1,28,28,1)/255.0

x\_test = x\_test.reshape(-1,28,28,1)/255.0

# 换one hot格式

y\_train = np\_utils.to\_categorical(y\_train,num\_classes=10)

y\_test = np\_utils.to\_categorical(y\_test,num\_classes=10)

# 定义顺序模型

model = Sequential()

# 第一个卷积层

# input\_shape 输入平面

# filters 卷积核/滤波器个数

# kernel\_size 卷积窗口大小

# strides 步长

# padding padding方式 same/valid

# activation 激活函数

model.add(Convolution2D(

input\_shape = (28,28,1),

filters = 32,

kernel\_size = 5,

strides = 1,

padding = 'same',

activation = 'relu',

name = 'conv1'

))

# 第一个池化层

model.add(MaxPooling2D(

pool\_size = 2,

strides = 2,

padding = 'same',

name = 'pool1'

))

# 第二个卷积层

model.add(Convolution2D(64,5,strides=1,padding='same',activation = 'relu',name='conv2'))

# 第二个池化层

model.add(MaxPooling2D(2,2,'same',name='pool2'))

# 把第二个池化层的输出扁平化为1维

model.add(Flatten())

# 第一个全连接层

model.add(Dense(1024,activation = 'relu'))

# Dropout

model.add(Dropout(0.5))

# 第二个全连接层

model.add(Dense(10,activation='softmax'))

# # 定义优化器

# adam = Adam(lr=1e-4)

# # 定义优化器，loss function，训练过程中计算准确率

# model.compile(optimizer=adam,loss='categorical\_crossentropy',metrics=['accuracy'])

# # 训练模型

# model.fit(x\_train,y\_train,batch\_size=64,epochs=1)

# # 评估模型

# loss,accuracy = model.evaluate(x\_test,y\_test)

# print('test loss',loss)

# print('test accuracy',accuracy)

# In[31]:

plot\_model(model,to\_file="model.png",show\_shapes=True,show\_layer\_names=True,rankdir='TB')

plt.figure(figsize=(10,10))

img = plt.imread("model.png")

plt.imshow(img)

plt.axis('off')

plt.show()