1. Gradient\_descent.ipynb(梯度縮減)

**import numpy as np**

**import matplotlib.pyplot as plt**

**# 目標函數:y=x^2**

**def func(x): return np.square(x)**

**# 目標函數一階導數:dy/dx=2\*x**

**def dfunc(x): return 2 \* x**

**def GD(x\_start, df, epochs, lr):**

**""" 梯度下降法。給定起始點與目標函數的一階導函數，求在epochs次反覆運算中x的更新值**

**:param x\_start: x的起始點**

**:param df: 目標函數的一階導函數**

**:param epochs: 反覆運算週期**

**:param lr: 學習率**

**:return: x在每次反覆運算後的位置（包括起始點），長度為epochs+1**

**"""**

**xs = np.zeros(epochs+1)**

**x = x\_start**

**xs[0] = x**

**for i in range(epochs):**

**dx = df(x)**

**# v表示x要改變的幅度**

**v = - dx \* lr**

**x += v**

**xs[i+1] = x**

**return xs**

**# Main**

**# 起始權重**

**x\_start = 5**

**# 執行週期數**

**epochs = 15**

**# 學習率**

**lr = 0.3**

**# 梯度下降法**

1. **x = GD(x\_start, dfunc, epochs, lr=lr)**

**print (x)**

**# 輸出：[-5. -2. -0.8 -0.32 -0.128 -0.0512]**

**color = 'r'**

**#plt.plot(line\_x, line\_y, c='b')**

**from numpy import arange**

**t = arange(-6.0, 6.0, 0.01)**

**plt.plot(t, func(t), c='b')**

**plt.plot(x, func(x), c=color, label='lr={}'.format(lr))**

**plt.scatter(x, func(x), c=color, )**

**plt.legend()**

**plt.show()**

一張含有 圖表 的圖片

自動產生的描述

2.Keras\_Tutorial.ipynb(利用KERAS套件 產生一 32個神經元模型進行訓練)

**from tensorflow import keras**

**from keras.models import Sequential #layers為keras堆疊層**

**from keras.layers import Dense**

***~#模型宣告~***

**import numpy as np**

**data = np.random.random((1000, 10)) *#隨機產生1000筆資料，每筆資料包含十個float數字***

**labels = np.random.randint(2, size=(1000, 1)) *#隨機產生1000筆資料，每筆資料上限為2(意思就是0 或 1)***

**print(np.shape(data))**

**print(data[0])**

**print(np.shape(labels))**

**print(labels[0])**

**model = Sequential()**

**model.add(Dense(32, activation='relu', input\_dim=10))**

**model.add(Dense(1, activation='sigmoid'))**

**~#模型建立~**

**model.compile(optimizer='rmsprop', loss='binary\_crossentropy', metrics=['accuracy'])**

**~#模型編譯~**

**model.fit(data, labels, epochs=10, batch\_size=32)**

**~#模型訓練~**

**score=model.evaluate(data, labels)**

**print(score[1])**

**一張含有 資料表 的圖片

自動產生的描述**

3.Keras\_tutorial2.ipynb

**from tensorflow import keras**

**from keras.models import Sequential**

**from keras.layers import Dense, Dropout**

**model = Sequential()**

**model.add(Dense(512, activation='relu', input\_dim=784))**

**model.add(Dropout(0.2))**

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

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

**import numpy as np**

**data = np.random.random((1000, 784))**

**labels = np.random.randint(10, size=(1000, 10))**

**model.fit(data, labels, epochs=10, batch\_size=32)**

**score=model.evaluate(data, labels)**

**print(score[1])**

**一張含有 資料表 的圖片

自動產生的描述**

4.keras\_CNN\_mnist.ipynb

**from tensorflow import keras**

**from keras.models import Sequential**

**from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten**

**model=Sequential()**

**model.add(Conv2D(filters=16, kernel\_size=(5, 5), input\_shape=(28, 28, 1), activation='relu'))**

**model.add(MaxPooling2D(pool\_size=(2, 2)))**

***#model.add(Conv2D(filters=36, kernel\_size=(5, 5), activation='relu'))***

***#model.add(MaxPooling2D(pool\_size=(2, 2)))***

**model.add(Flatten())**

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

**model.summary()**

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

**一張含有 文字 的圖片

自動產生的描述**

**mnist=keras.datasets.mnist**

**(train\_data, train\_label), (test\_data, test\_label) = mnist.load\_data()**

**test\_label\_original=test\_label *#for crosstab***

**train\_label = keras.utils.to\_categorical(train\_label)**

**test\_label = keras.utils.to\_categorical(test\_label)**

**train\_history=model.fit(train\_data,train\_label,epochs=10,batch\_size=512)**

一張含有 資料表 的圖片

自動產生的描述

**score=model.evaluate(test\_data, test\_label)**

**print(score[1])**

****

***#繪製正確性與損失立歷史圖***

**import matplotlib.pyplot as plt**

**import numpy as np**

**def show\_train\_history(train\_history, train, validation):**

**plt.plot(train\_history.history[train])**

**plt.plot(train\_history.history[validation])**

**plt.title('Train History')**

**plt.ylabel('Train')**

**plt.xlabel('Epoch')**

**plt.legend(['train', 'validation'], loc='center right')**

**plt.show()**

**show\_train\_history(train\_history, 'accuracy', 'accuracy')**

**show\_train\_history(train\_history, 'loss', 'loss')**

***# Confusion Matrix混淆矩陣***

**import pandas as pd**

**prediction = model.predict(test\_data)**

**prediction\_label=np.argmax(prediction,axis=1)**

**print(test\_label.shape)**

**pd.crosstab(test\_label\_original, prediction\_label, rownames=['label'], colnames=['predict'])**

**一張含有 圖表 的圖片

自動產生的描述**

**一張含有 圖表 的圖片

自動產生的描述**

**(10000, 10)**

**一張含有 文字 的圖片

自動產生的描述**

5.keras\_mnist.ipynb

**from tensorflow import keras**

**from keras.models import Sequential**

**from keras.layers import Dense, Flatten**

**model=Sequential()**

**model.add(Flatten())**

**model.add(Dense(32, activation='relu', input\_dim=784))**

**model.add(Dense(10, activation='sigmoid'))**

**model.compile(optimizer='rmsprop', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])**

**mnist=keras.datasets.mnist**

**(train\_data, train\_label), (test\_data, test\_label) = mnist.load\_data()**

**print(type(train\_data))**

**print(train\_data.shape)**

**print(type(train\_label))**

**print(train\_label.shape)**

**print(test\_data.shape)**

**print(test\_label.shape)**

**一張含有 文字, 信 的圖片

自動產生的描述**

**print(train\_label[0])**

**print(train\_data[0])**

一張含有 圖表 的圖片

自動產生的描述

**import matplotlib.pyplot as plt**

**plt.imshow(train\_data[0], cmap='binary')**

**plt.show()**

**一張含有 圖表 的圖片

自動產生的描述**

**model.fit(train\_data,train\_label,epochs=20,batch\_size=512)**

**一張含有 資料表 的圖片

自動產生的描述**

**score=model.evaluate(test\_data, test\_label)**

**print(score[1])**

****