

3.1.1~3.1.7_周哲煒

- 3.1.1~3.1.7 周哲煒
 - 3.1.1 text sample data I/O
 - 3.1.2 What is machine learning do?
 - 機器學習主要分三類
 - 3.1.3線性回歸
 - 3.1.4正規方程式法求解(一堆矩陣數學)
 - 3.1.5梯度下降法加速求解
 - 3.1.6偵錯學習率
 - 3.1.7梯度驗證
 - code

3.1.1 text sample data I/O

```
1 x,y=[],[]
2 with open('food_truck_data.txt') as data_f: #file IO
3     content = data_f.readlines() #read file content, to a list
4     for line in content:
5         tmp = line.split(",")
6         x.append(tmp[0])
7         y.append(tmp[1])
8
```

3.1.2 What is machine learning do?

機器學習的目標:找出函數(eg: $y = f(x) = ax+b$)描述資料的關係

x:=樣本特徵

y:=樣本標籤

求解的過程-->訓練模型(找出預測值和目標值的最小誤差)

機器學習主要分三類

1. 監督式學習: 用答案去訓練
 - 需要訓練樣本
 - 設計可以 **良好 刻劃** sample data <-> sample label關係的函數
 - 選擇**合理的**損失函數，描述誤差
 - 訓練模型
 - 使用模型
 - p.s.分類問題、回歸問題
2. 非監督式學習:自己找規律猜答案
3. 強化學習:與環境互動的經驗回饋來訓練

3.1.3 線性回歸

對樣本 $(x^{(i)}, y^{(i)})$ ，用記號 $f^{(i)}$ 表示假設函數 $f(x)$ 的預測值 $f(x^{(i)})$ 。對於「餐車利潤問題」，可用方差 $(f^{(i)} - y^{(i)})^2$ 表示單一樣本的預測誤差。所有樣本的預測誤差，公式如下。

$$L = \frac{1}{m} \sum_{i=1}^m (f^{(i)} - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m (wx^{(i)} + b - y_i)^2$$

L 可以看成未知參數 w 、 b 的函數 $L(w, b)$ ，用於刻畫在樣本資料上模型預測的誤差。 $L(w, b)$ 稱為損失函數（也稱為誤差函數）。模型訓練就是求解使損失函數 $L(w, b)$ 的值最小的參數 w 、 b 。

當然，對一個函數乘以一個常數，不會影響其最小值的參數。有些時候，為了使導數更簡單，會將上式第一個等號右邊的式子除以 2，作為損失函數，具體如下。

$$L(w, b) = \frac{1}{2m} \sum_{i=1}^m (f^{(i)} - y^{(i)})^2 = \frac{1}{2m} \sum_{i=1}^m (wx^{(i)} + b - y_i)^2$$

線性回歸就是求使這個損失函數的值最小的參數 w 、 b 。求最小值有兩種方法，分別是正規方程式法和梯度下降法。

3.1.4 正規方程式法求解(一堆矩陣數學)

損失函數 $L(w, b)$ 是一個關於 (w, b) 的多變數函數。 $L(w, b)$ 關於 (w, b) 的偏導數如下。

$$\frac{\partial L}{\partial w} = \frac{1}{2m} \frac{\partial \left(\sum (wx^{(i)} + b - y^{(i)})^2 \right)}{\partial w} = \frac{1}{m} \sum (wx^{(i)} + b - y^{(i)}) x^{(i)}$$
$$\frac{\partial L}{\partial b} = \frac{1}{2m} \frac{\partial \left(\sum (wx^{(i)} + b - y^{(i)})^2 \right)}{\partial b} = \frac{1}{m} \sum (wx^{(i)} + b - y^{(i)})$$

函數 $L(w, b)$ 的最小值必須滿足的條件是： $L(w, b)$ 關於引數（即 (w, b) 的梯度，或說偏導數）等於 0，公式如下。

$$\frac{\partial L}{\partial w} = \frac{1}{m} \sum (wx^{(i)} + b - y^{(i)}) x^{(i)} = 0$$
$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum (wx^{(i)} + b - y^{(i)}) = 0$$

令

$$X = \begin{pmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & \vdots \\ 1 & x^{(m)} \end{pmatrix}, \quad W = \begin{pmatrix} b \\ w \end{pmatrix}, \quad Y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

去掉方程式的係數 $\frac{1}{m}$ ，有

$$(x^{(1)} \ x^{(2)} \ \dots \ x^{(m)}) \begin{pmatrix} wx^{(1)} + b - y^{(1)} \\ wx^{(2)} + b - y^{(2)} \\ \vdots \\ wx^{(m)} + b - y^{(m)} \end{pmatrix} = 0$$

$$(1 \ 1 \ \dots \ 1) \begin{pmatrix} wx^{(1)} + b - y^{(1)} \\ wx^{(2)} + b - y^{(2)} \\ \vdots \\ wx^{(m)} + b - y^{(m)} \end{pmatrix} = 0$$

即

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{pmatrix} \begin{pmatrix} b + wx^{(1)} - y^{(1)} \\ b + wx^{(2)} - y^{(2)} \\ \vdots \\ b + wx^{(m)} - y^{(m)} \end{pmatrix} = 0$$

令

$$\mathbf{X} = \begin{pmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & \vdots \\ 1 & x^{(m)} \end{pmatrix}, \quad \mathbf{W} = \begin{pmatrix} b \\ w \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

因此，有 $\mathbf{X}^T(\mathbf{XW} - \mathbf{y}) = 0$ ，即

$$\mathbf{X}^T \mathbf{X} \mathbf{W} = \mathbf{X}^T \mathbf{y}$$

等號兩邊同時乘以 $\mathbf{X}^T \mathbf{X}$ 的反矩陣 $(\mathbf{X}^T \mathbf{X})^{-1}$ ，結果如下。

$$\mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

求得 $\mathbf{W} = (b, w)$ 。

求解 \mathbf{W} 的正規方程式 (Normal Equation) 如下。

$$\frac{\partial L}{\partial w} = \frac{1}{m} \sum (wx^{(i)} + b - y^{(i)}) x^{(i)} = 0$$

正規方程式法求解餐車利潤問題

```
1 import numpy
2
3 #data是mx2的矩陣，每行表示一個樣本
4 data = np.loadtxt('food_truck_data.txt', delimiter=',')
5 train_x = data[:,0] #城市人口，mx1的矩陣
6 train_y = data[:,1] #餐車利潤，mx1的矩陣
7
8 X = np.ones(shape=len(train_x), 2)
9 X[:,1] = train_x
10 y = train_y
11
12 XT = X.transpose()
13 XTy = XT@y
14
15 w = np.linalg.inv(XT@X)@ XTy
16 print(w)
```

3.1.5 梯度下降法加速求解

正規方法需要計算矩陣乘法&反矩陣→若特徵或樣本較多，則太耗時。

一般用梯度下降法來求解(較快速)

從(w₀,b₀)出發，透過下列公式更新

$$w_{i+1} := w_i - \alpha \frac{\partial L}{\partial w_i}$$

$$b_{i+1} := w_i - \alpha \frac{\partial L}{\partial b_i}$$

其中

$$\frac{\partial L}{\partial w_i} = np.mean((w_i x + b_i - y) \cdot x)$$

$$\frac{\partial L}{\partial b_i} = np.mean(w_i x + b_i - y)$$

觀察疊代情況->繪製**loss curve**

3.1.6 偵錯學習率

偵錯學習率的過程就是用不同的學習率嘗試學習

- 一樣是用**loss curve**來觀察

3.1.7 梯度驗證

在執行梯度下降法前，應進行梯度驗證，以保證梯度和函數數值的計算正確

對線性回歸問題，應使用以下數值梯度公式來檢驗：

(數值梯度)

$$\frac{\partial L(w, b)}{\partial w} = \lim_{\epsilon \rightarrow 0} \frac{L(w + \epsilon, b) - L(w - \epsilon, b)}{2\epsilon}$$

$$\frac{\partial L(w, b)}{\partial b} = \lim_{\epsilon \rightarrow 0} \frac{L(w, b + \epsilon) - L(w, b - \epsilon)}{2\epsilon}$$

CODE

分析梯度

```
1 import numpy as np
2
3 dw = np.mean((w*x+b-y)*x)
4 db = np.mean(w*x+b-y)
```

數值梯度

```
1 df_approx = lambda x,y,w,b,eps:((loss(x,y,w+eps,b)\
2 -loss(x,y,w-eps,b))/(2*eps),(loss(x,y,w,b+eps)\
3 -loss(x,y,w,b-eps))/(2*eps))
```

在任意點，如 $(w,b)=(1.0,-2.0)$ ，比較數值梯度和分析梯度，
若非常接近則可以使用

```
1 import numpy
2
3 w,b,eps = 1.0,-2.0,1e-8
4 dw = np.mean((w*X+b-y)*X)
5 db = np.mean(w*X+b-y)
6 grad = np.array([dw,db])
7 grad_approx = df_approx(X,y,w,b,eps)
8 print(grad)
9 print(grad_approx)
10
```