Prediction and Optimization

何景盛 B08901123

2023/6/2

Abstract

由於人工智慧的發展迅速，各式各樣的技術產生出來，而這些技術背後也有一定的原理，因此勾起我對這些技術原理的渴望，進而引發修習此專題的動機。而此專題修習的部分主要為預測和最優化，並且以數據模擬來驗證其數學模型和演算法。為期十三週，前五週主要為預測，接著一週為相關性，餘下的週數專注在最優化的部分。通過這次專題的學習，令我更深入了解預測模型和最優化算法的理論，這將有助提升我未來在此領域解決問題的能力。詳細程式在 https://github.com/JohnHo112/lab_prediction-and-optimization

1 Prediction Model

1. Linear Prediction with optimal parameters
2. Nonlinear Prediction with optimal parameters
3. Generalized Nonlinear Prediction with optimal parameters
4. Curve fitting with optimal parameters
5. Using PCA to prediction

預測模型的部分以以上五個模型作預測，以 kaggle 上的 Daily Temperature of Major Cities[1] 這個 dataset 來作溫度的預測，評比誤差以 mean square error(MSE) 來作為標準。

2 Optimization

1. Step Search Method
2. Golden Search
3. Newton Method
4. Gradient Descend
5. L_α-Norm

最優化算法的部分以以上五個算法，並用一些 convex function(拋物線、拋物面...) 來作為測試，並觀測其收斂性。

3 Linear Prediction with optimal parameters

3.1 Theory

Linear Prediction model 如下 Eq.1：

$$x_p[n] = \sum_{k=1}^{L} a_k x[n-k] \tag{1}$$

其中 $x_p[n]$ 為模型預測值，$x[n]$ 是輸入資料，a_k 為最佳化參數，L 為常數與預測值跟前幾筆 $x[n]$ 有關。我們的目標是找一組合適的 a_k 使得 MSE 最小。首先需要把 data 分成兩組

train data 和 test data。train data: $\{x[n]|n_1 \le n \le n_2\}$，test data: $\{x[n]|n_3 \le n \le n_4\}$，而 $n_1 < n_2 < n_3 < n_4$，從 train data 中找出 a_k，然後代入 Eq.1，進而對數據預測。

使用以下 MSE object function Eq.2作為 test data 的誤差衡量：

$$E = \sum_{n=n_3}^{n_4} (x[n] - x_p[n])^2 \tag{2}$$

我們目標是從 train data 找出誤差最小的 a_k。

$$E = \sum_{n=n_1}^{n_2} (x[n] - \sum_{k=1}^{L} a_k x[n-k])^2 \tag{3}$$

因此需要用 Eq.3對 a_k 進行偏微分等於 0，推導過程如下。

$$\frac{\partial E}{\partial a_k} = 0$$

for k = 1, 2, ..., L

$$\Rightarrow -2 \sum_{n=n_1}^{n_2} x[x-k](x[n] - \sum_{s=1}^{L} a_s x[n-s]) = 0$$

$$\Rightarrow \sum_{s=1}^{L} a_s \sum_{n=n_1}^{n_2} x[n-k]x[n-s] = \sum_{n=n_1}^{n_2} x[n-k]x[n]$$

$$\Rightarrow \begin{bmatrix} \sum_{n=n_1}^{n_2} x^2[n-1] & \sum_{n=n_1}^{n_2} x[n-1]x[n-2] & \dots & \sum_{n=n_1}^{n_2} x[n-1]x[n-L] \\ \sum_{n=n_1}^{n_2} x[n-2]x[n-1] & \sum_{n=n_1}^{n_2} x^2[n-2] & \dots & \sum_{n=n_1}^{n_2} x[n-2]x[n-L] \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{n=n_1}^{n_2} x[n-L]x[n-1] & \sum_{n=n_1}^{n_2} x[n-L]x[n-2] & \dots & \sum_{n=n_1}^{n_2} x^2[n-L] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_L \end{bmatrix} = \begin{bmatrix} \sum_{n=n_1}^{n_2} x[n-1]x[n] \\ \sum_{n=n_1}^{n_2} x[n-2]x[n] \\ \vdots \\ \sum_{n=n_1}^{n_2} x[n-L]x[n] \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_L \end{bmatrix} = \begin{bmatrix} \sum_{n=n_1}^{n_2} x[n-1]x[n] \\ \sum_{n=n_1}^{n_2} x[n-2]x[n] \\ \vdots \\ \sum_{n=n_1}^{n_2} x[n-L]x[n] \end{bmatrix} \begin{bmatrix} \sum_{n=n_1}^{n_2} x^2[n-1] & \sum_{n=n_1}^{n_2} x[n-1]x[n-2] & \dots & \sum_{n=n_1}^{n_2} x[n-1]x[n-L] \\ \sum_{n=n_1}^{n_2} x[n-2]x[n-1] & \sum_{n=n_1}^{n_2} x^2[n-2] & \dots & \sum_{n=n_1}^{n_2} x[n-2]x[n-L] \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{n=n_1}^{n_2} x[n-L]x[n-1] & \sum_{n=n_1}^{n_2} x[n-L]x[n-2] & \dots & \sum_{n=n_1}^{n_2} x^2[n-L] \end{bmatrix}^{-1} \tag{4}$$

求得 a_k 後，可以把它與 test data 代入 Eq.1，並求出與 $\{x[n_3], x[n_3+1], ..., x[n_4]\}$ 相同 index 的 $\{x_p[n_3], x_p[n_3+1], ..., x_p[n_4]\}$ 代入 Eq.2，從而觀測其誤差。

3.2 Simulation

用台灣 2013 年的溫度作為 training data 來找 a_k，並用 2018 年的溫度作為 testing data 去進行預測，得出 fig.1。最後得出 MSE 為 4.1227。

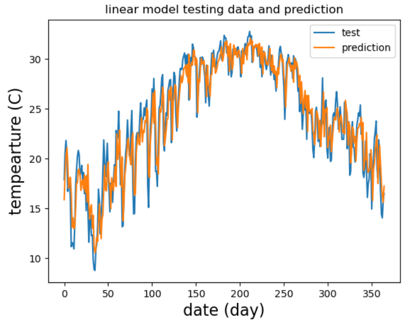


Figure 1: linear model result

4 Nonlinear Prediction with optimal parameters

4.1 Theory

Nonlinear Prediction model 如下 Eq.5

$$x_p[n] = \sum_{k=1}^{L} a_k x[n-k] + \sum_{k=1}^{L-1} b_k (x[n-k] - x[n-k-1])^2 \tag{5}$$

其中 $x_p[n]$ 為模型預測值，$x[n]$ 是輸入資料，a_k 和 b_k 為最佳化參數，L 為常數與預測值跟前幾筆 $x[n]$ 有關。我們的目標是找一組合適的 a_k 和 b_k 使得 MSE 最小。首先需要和 Linear Prediction 一樣，把 data 分為 train data 和 test data，從 train data 中找出 a_k 和 b_k，然後代入 Eq.5，進而對數據預測。

同樣也是使用 MSE object function Eq.6作為 test data 的誤差衡量：

$$E = \sum_{n=n_3}^{n_4} (x[n] - x_p[n])^2 \tag{6}$$

目標是從 train data 找出使得 Eq.7最小的 a_k 和 b_k。

$$E = \sum_{n=n_1}^{n_2} (x[n] - \sum_{k=1}^{L} a_k x[n-k] - \sum_{k=1}^{L-1} b_k (x[n-k] - x[n-k-1])^2)^2 \tag{7}$$

因此需要對 Eq.7分別對 a_k 和 b_k 進行偏微分等於 0，推導過程如下。

$$\frac{\partial E}{\partial a_k} = 0 \quad for\ k = 1, 2, ..., L$$

$$\begin{aligned} &\Rightarrow \sum_{s=1}^{L} a_s \sum_{n=n_1}^{n_2} x[n-k]x[n-s] + \sum_{s=1}^{L-1} b_s \sum_{n=n_1}^{n_2} x[n-k](x[n-s] - x[n-s-1])^2 \\ &= \sum_{n=n_1}^{n_2} x[n-k]x[n] \end{aligned} \tag{8}$$

$$\frac{\partial E}{\partial b_k} = 0 \quad for\ k = 1, 2, ..., L-1$$

$$\begin{aligned} &\Rightarrow \sum_{s=1}^{L} a_s \sum_{n=n_1}^{n_2} (x[n-k] - x[n-k-1])^2 x[n-s] \\ &+ \sum_{s=1}^{L-1} b_s \sum_{n=n_1}^{n_2} (x[n-k] - x[n-k-1])^2 (x[n-s] - x[n-s-1])^2 \\ &= \sum_{n=n_1}^{n_2} (x[n-k] - x[n-k-1])^2 x[n] \end{aligned} \tag{9}$$

綜合 Eq.8和 Eq.9可得出

$$\begin{bmatrix} a_1 \\ \vdots \\ a_L \\ b_1 \\ \vdots \\ b_{L-1} \end{bmatrix} = \begin{bmatrix} A_{1,1} & \dots & A_{1,L} & B_{1,1} & \dots & B_{1,L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{L,1} & \dots & A_{L,L} & B_{L,1} & \dots & B_{L,L-1} \\ B_{1,1} & \dots & B_{L,1} & C_{1,1} & \dots & C_{1,L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{1,L-1} & \dots & B_{L,L-1} & C_{L-1,1} & \dots & C_{L-1,L-1} \end{bmatrix}^{-1} \begin{bmatrix} A_{1,0} \\ \vdots \\ A_{L,0} \\ B_{0,1} \\ \vdots \\ B_{0,L-1} \end{bmatrix}$$

$$\begin{aligned} where \quad A_{i,j} &= \sum_{n=n_1}^{n_2} x[n-i]x[n-j] \\ B_{i,j} &= \sum_{n=n_1}^{n_2} x[n-i](x[n-j]-x[n-j-1])^2 \\ C_{i,j} &= \sum_{n=n_1}^{n_2} (x[n-i]-x[n-i-1])^2(x[n-j]-x[n-j-1])^2 \end{aligned}$$

求得 a_k 和 b_k 後，方法和 Linear Prediction 一樣，可以把它與 test data 代入 Eq.5，並求出與 $\{x[n_3], x[n_3+1], ..., x[n_4]\}$ 相同 index 的 $\{x_p[n_3], x_p[n_3+1], ..., x_p[n_4]\}$ 代入 Eq.6，從而觀測其誤差。

4.2 Simulation

同上 Linear Prediction 一樣用台灣 2013 年的溫度作為 training data 來找 a_k,b_k，並用 2018 年的溫度作為 testing data 去進行預測，得出 Fig.2，得出 MSE 為 4.329

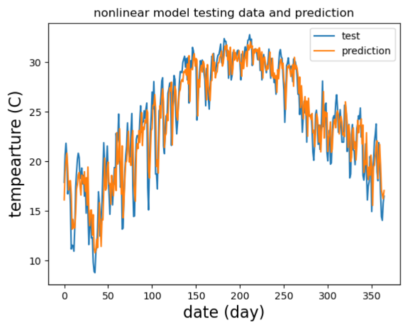


Figure 2: nonlinear model result

5 Generalized Nonlinear Prediction with optimal parameters

5.1 Theory

Generalized Nonlinear Prediction Model 基本上跟 Nonlinear Prediction Model 樣，差別在非線性項不是 2 次方，而是 α 次方，如下 Eq.10

$$x_p[n] = \sum_{k=1}^{L} a_k x[n-k] + \sum_{k=1}^{L-1} b_k(x[n-k]-x[n-k-1])^{\alpha} \tag{10}$$

其中 $x_p[n]$ 為模型預測值，$x[n]$ 是輸入資料，a_k 和 b_k 為最佳化參數，L 為常數與預測值跟前幾筆 $x[n]$ 有關，α 可以任意選。我們的目標是找一組合適的 a_k 和 b_k 使得 MSE 最小。首先需要和前面一樣把 data 分為 train data 和 test data，從 train data 中找出 a_k 和 b_k，然後代入 Eq.10，進而對數據預測。

使用 Eq.11來衡量誤差

$$E = \sum_{n=n_3}^{n_4} (x[n] - x_p[n])^2 \tag{11}$$

這次使用的 object function 有點不一樣，在每筆資料前加一個 $w[n]$(weight)，可以增加其重要性，例如預測溫度，在 train datas 找最佳化參數時，可以分為每年季節，可能 train datas 的春天和 test datas 的春天比較有關聯，因此可以增加其權重。其 object function 如下 Eq.12：

$$E = \sum_{n=n_1}^{n_2} w[n](x[n] - \sum_{k=1}^{L} a_k x[n-k] - \sum_{k=1}^{L-1} b_k (x[n-k] - x[n-k-1])^{\alpha})^2 \tag{12}$$

對 Eq.12取偏微分可求得 a_k 和 b_k

$$\frac{\partial E}{\partial a_k} = 0 \quad for\ k = 1, 2, ..., L$$

$$\begin{aligned} &\Rightarrow \sum_{s=1}^{L} a_s \sum_{n=n_1}^{n_2} w[n]x[n-k]x[n-s] + \sum_{s=1}^{L-1} b_s \sum_{n=n_1}^{n_2} w[n]x[n-k](x[n-s] - x[n-s-1])^{\alpha} \\ &= \sum_{n=n_1}^{n_2} w[n]x[n-k]x[n] \end{aligned} \tag{13}$$

$$\frac{\partial E}{\partial b_k} = 0 \quad for\ k = 1, 2, ..., L-1$$

$$\begin{aligned} &\Rightarrow \sum_{s=1}^{L} a_s \sum_{n=n_1}^{n_2} w[n](x[n-k] - x[n-k-1])^{\alpha} x[n-s] \\ &+ \sum_{s=1}^{L-1} b_s \sum_{n=n_1}^{n_2} w[n](x[n-k] - x[n-k-1])^{\alpha}(x[n-s] - x[n-s-1])^{\alpha} \\ &= \sum_{n=n_1}^{n_2} w[n](x[n-k] - x[n-k-1])^{\alpha} x[n] \end{aligned} \tag{14}$$

綜合 Eq.13和 Eq.14可得出

$$\begin{bmatrix} a_1 \\ \vdots \\ a_L \\ b_1 \\ \vdots \\ b_{L-1} \end{bmatrix} = \begin{bmatrix} A_{1,1} & \dots & A_{1,L} & B_{1,1} & \dots & B_{1,L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{L,1} & \dots & A_{L,L} & B_{L,1} & \dots & B_{L,L-1} \\ B_{1,1} & \dots & B_{L,1} & C_{1,1} & \dots & C_{1,L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{1,L-1} & \dots & B_{L,L-1} & C_{L-1,1} & \dots & C_{L-1,L-1} \end{bmatrix}^{-1} \begin{bmatrix} A_{1,0} \\ \vdots \\ A_{L,0} \\ B_{0,1} \\ \vdots \\ B_{0,L-1} \end{bmatrix}$$

$$\begin{aligned} where \quad A_{i,j} &= \sum_{n=n_1}^{n_2} w[n]x[n-i]x[n-j] \\ B_{i,j} &= \sum_{n=n_1}^{n_2} w[n]x[n-i](x[n-j] - x[n-j-1])^2 \\ C_{i,j} &= \sum_{n=n_1}^{n_2} w[n](x[n-i] - x[n-i-1])^2(x[n-j] - x[n-j-1])^2 \end{aligned}$$

求得 a_k 和 b_k，與 test data 代入 Eq.10，並將 $\{x[n_3], x[n_3+1], ..., x[n_4]\}$ 相同 index 的 $\{x_p[n_3], x_p[n_3+1], ..., x_p[n_4]\}$ 代入 Eq.11，從而觀測其誤差。

5.2 Simulation

這次嘗試增加 training data 的數量，用 2007 年至 2016 年的溫度，把每年平均分為 5 份，分別對應 5 個 step function weight，如下 Fig.3，並用 2018 年的溫度作為 testing data 去進行預測，得出 Fig.4，其 MSE 為 3.8694，可見得增加 training data 數和 weight 可以降低其誤差。

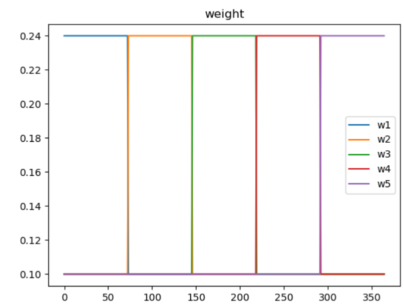


Figure 3: generalized nonlinear model weight

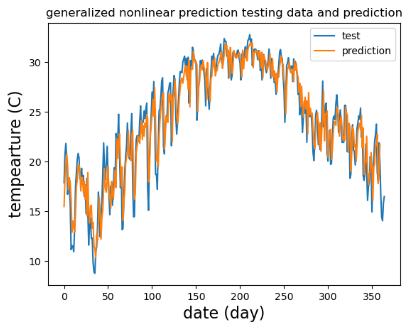


Figure 4: generalized nonlinear model result

6 Curve fitting with optimal parameters

6.1 Theory

以下為多項式的 Curve fitting 的模型：

$$x[n] = \sum_{k=0}^{M} a_k n^k \tag{15}$$

其中 $x[n]$ 為模型預測值，a_k 為最佳化參數，M 為多項式數目。求 a_k 的方和前面的一樣，用 object function 取偏微分等於 0，而這次嘗試有 weights 和沒有 weights，object function 如下 Eq.17和 Eq.19
使用 Eq.16來衡量誤差

$$E = \sum_{n=n_3}^{n_4} (x[n] - x_p[n])^2 \tag{16}$$

1. without weight

$$E = \sum_{n=n_1}^{n_2} (x[n] - \sum_{k=0}^{M} a_k n^k)^2 \tag{17}$$

$$\frac{\partial E}{\partial a_k} = 0 \quad for\ k = 0, 1, 2, ..., M$$

$$\Rightarrow -2 \sum_{n=n_1}^{n_2} n^k (x[n] - \sum_{s=0}^{M} a_s n^s) = 0$$

$$\Rightarrow \sum_{s=0}^{M} a_s \sum_{n=n_1}^{n_2} n^k n^s = \sum_{n=n_1}^{n_2} n^k x[n]$$

$$\begin{bmatrix} \sum_{n=n_1}^{n_2} n^{0+0} & \sum_{n=n_1}^{n_2} n^{0+1} & \dots & \sum_{n=n_1}^{n_2} n^{0+M} \\ \sum_{n=n_1}^{n_2} n^{1+0} & \sum_{n=n_1}^{n_2} n^{1+1} & \dots & \sum_{n=n_1}^{n_2} n^{1+M} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{n=n_1}^{n_2} n^{M+0} & \sum_{n=n_1}^{n_2} n^{M+1} & \dots & \sum_{n=n_1}^{n_2} n^{M+M} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \sum_{n=n_1}^{n_2} n^0 x[n] \\ \sum_{n=n_1}^{n_2} n^1 x[n] \\ \vdots \\ \sum_{n=n_1}^{n_2} n^M x[n] \end{bmatrix} \tag{18}$$

2. with weight

$$E = \sum_{n=n_1}^{n_2} w[n](x[n] - \sum_{k=0}^{M} a_k n^k)^2 \tag{19}$$

$$\frac{\partial E}{\partial a_k} = 0 \quad for\ k = 0, 1, 2, ..., M$$

$$\Rightarrow -2 \sum_{n=n_1}^{n_2} w[n] n^k (x[n] - \sum_{s=0}^{M} a_s n^s) = 0$$

$$\Rightarrow \sum_{s=0}^{M} a_s \sum_{n=n_1}^{n_2} w[n] n^k n^s = \sum_{n=n_1}^{n_2} w[n] n^k x[n]$$

$$\begin{bmatrix} \sum_{n=n_1}^{n_2} w[n]n^{0+0} & \sum_{n=n_1}^{n_2} w[n]n^{0+1} & \dots & \sum_{n=n_1}^{n_2} w[n]n^{0+M} \\ \sum_{n=n_1}^{n_2} w[n]n^{1+0} & \sum_{n=n_1}^{n_2} w[n]n^{1+1} & \dots & \sum_{n=n_1}^{n_2} w[n]n^{1+M} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{n=n_1}^{n_2} w[n]n^{M+0} & \sum_{n=n_1}^{n_2} w[n]n^{M+1} & \dots & \sum_{n=n_1}^{n_2} w[n]n^{M+M} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \sum_{n=n_1}^{n_2} w[n]n^0 x[n] \\ \sum_{n=n_1}^{n_2} w[n]n^1 x[n] \\ \vdots \\ \sum_{n=n_1}^{n_2} w[n]n^M x[n] \end{bmatrix} \tag{20}$$

求得 a_k 可代入 Eq.15進行預測，最後代入 Eq.16來判斷模型誤差。

6.2 Simulation

同樣 training data 也是用 10 年，這次有兩組的結果，一組是沒有 weight Fig.6(a)，其 MSE 為 7.3696，另外一組是有 weight Fig.6(b)，而 weight function 如下 Fig.5，其 MSE 為 6.7631。從結果可以觀察到加了 weight 相對的誤差也會降低。

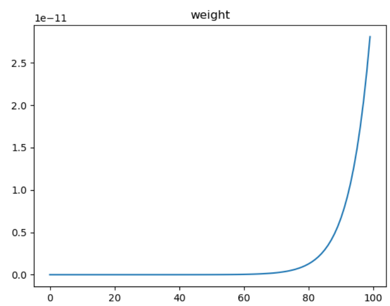
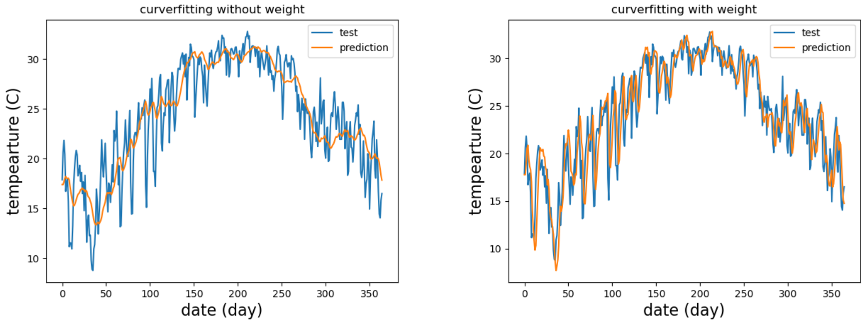


Figure 5: curvefitting weight



(a) curvefitting result without weight　(b) curvefitting result with weight

Figure 6: Curve fitting result

7 Using PCA to prediction

7.1 Theory

用 PCA 來進行預測就跟前面的方法有點不同，不是用偏微分來求。假設現在有 M 筆資料，而每一筆資料為 N-dimension

$$\begin{gathered} g_1 = [f_{1,1}, f_{1,2}, ..., f_{1,N}] \\ g_2 = [f_{2,1}, f_{2,2}, ..., f_{2,N}] \\ \vdots \\ g_M = [f_{M,1}, f_{M,2}, ..., f_{M,N}] \end{gathered}$$

PCA 預測流程如下：

1. 扣掉平均值，形成新 data
   $d_m = [e_{m,1}, e_{m,2}, ..., e_{m,N}] \quad m = 1, 2, ..., M$
   其中 $e_{m,n} = f_{m,n} - \hat{f}_n$, $\hat{f}_n = \frac{1}{M}\sum_{m=1}^{M} f_{m,n}$
2. 形成 $M \times N$ 矩陣 A
   A 的第 m 個 row 為 d_m，$m = 1, 2, ..., M$
3. 對 A 進行 SVD 分解
   $A = USV^H = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + ... + \lambda_k u_k v_k^T, \quad k = min(M, N)$
   $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k$
4. 將 A 近似成 $A = USV^H = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + ... + \lambda_h u_h v_h^T, \quad for\ h \leq k$

則每一筆資料可近似為

$$g_m = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + ... + \lambda_h u_h v_h^T + [\hat{f}_1, \hat{f}_2, ..., \hat{f}_N]$$

v_1^T 是資料的最主要成分，v_2^T 是資料的次要成分，v_3^T 是資料的再次要成分，如此類推。

以預測溫度為例的二維的資料，x 為日子，y 為溫度。

$$\begin{gathered} g_1 = [x_1 \quad y_1] \\ g_2 = [x_2 \quad y_2] \\ \vdots \\ g_M = [x_M \quad y_M] \end{gathered}$$

根據上述方法可求得主要成份 $v_1^T = [a_1 \quad a_2]$，並用主成分來求得迴歸線，式子如下：

$$\begin{gathered} [x_0 \quad y_0] + cv_1^T = [x_1 \quad y_1] \\ [x_0 \quad y_0] + c[a_1 \quad a_2] = [x_1 \quad y_1] \end{gathered}$$

c 為常數，x_0 為已知日子，y_0 為 x_0 的溫度，x_1 為預測的日子，目標是求 y_1，可以經下列運算求得 y_1：

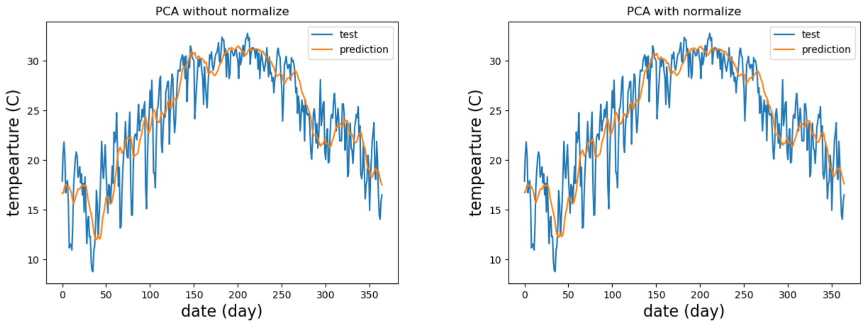
$$x_1 = x_0 + ca_1 \Rightarrow c = \frac{x_1 - x_0}{a_1}$$

$$y_1 = y_0 + ca_2 = y_0 + \frac{x_1 - x_0}{a_1} a_2$$

得到 y_1 後可用相同方法求得 $y_2, ..., y_n$。最後再用 MSE 來判斷其誤差。

7.2 Simulation

同樣 training data 也是用 10 年，這次有兩組的結果，一組是沒有 normalize data Fig.7(a)，其 MSE 為 7.3653，另外一組是有 normalize data Fig.7(b)，其 MSE 為 7.2818。從結果可以觀察到有做 normalize 相對的誤差也會下降。



(a) PCA without normalize

(b) PCA with normalize

Figure 7: PCA result

8 Correlation

8.1 Theory

以下 Eq.21為相關性：

$$corr_{X,Y} = \frac{cov_{X,Y}}{\sigma_X \sigma_Y} = \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y} = \frac{\sum_{n=1}^{N}(x[n] - \mu_X)(y[n] - \mu_Y)}{\sqrt{\sum_{n=1}^{N}(x[n] - \mu_X)^2 \sum_{m=1}^{N}(y[m] - \mu_Y)^2}} \tag{21}$$

其中 $\mu_X = \frac{1}{N}\sum_{n=1}^{N} x[n]$, $\mu_Y = \frac{1}{N}\sum_{n=1}^{N} y[n]$

當中 $x[n]$ 為一筆資料，$y[n]$ 為另一筆資料，我們可以用 $corr_{X,Y}$ 來觀察這兩筆資料的相關性，而 $corr_{X,Y} \in [-1 \quad 1]$，得出 $corr_{X,Y}$ 可分為以下的相關性：

1. Full Correlation: $|corr_{X,Y}| \geq 0.9$
2. High Correlation: $0.6 \leq |corr_{X,Y}| < 0.9$
3. Middle Correlation: $0.3 \leq |corr_{X,Y}| < 0.6$
4. Low Correlation: $|corr_{X,Y}| < 0.3$
5. Positive Correlation: $corr_{X,Y} > 0$
6. Negative Correlation: $corr_{X,Y} < 0$

8.2 Simulation

嘗試用 2018 的 data 來當作主要的 label 與 2018 向之前 shift 1 至 356 天 data 來觀察它們之間的 correlation，如下圖 Fig.8，並觀察最大、最小和接近 0 的 correlation，如下圖 Fig.9，Fig.9(a) corr = 0.9262，可觀察到 Fig.9(d) label 和 shift data 機乎一樣，Fig.9(b) corr = -0.801，可觀察到 Fig.9(e) label 和 shift data 呈反相，Fig.9(c) corr = 0.0016，可觀察到 Fig.9(f) label 和 shift data 並沒有相關。

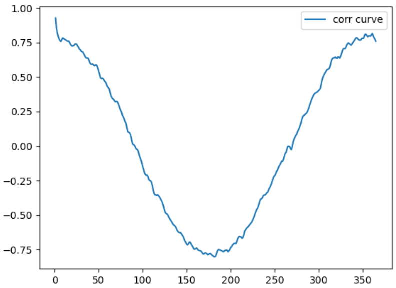
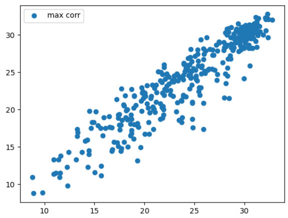
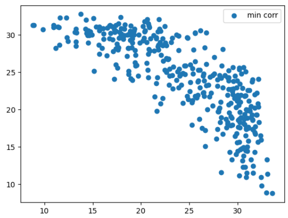


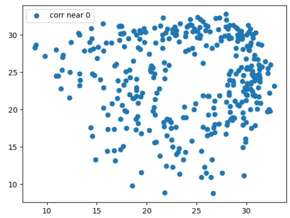
Figure 8: 2018 and shift data correlation



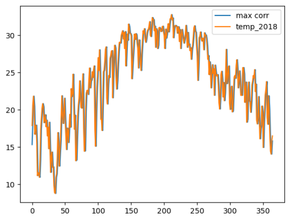
(a) max correlation



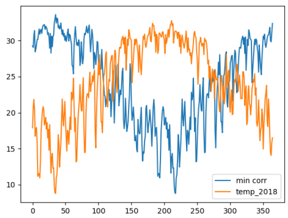
(b) min correlation



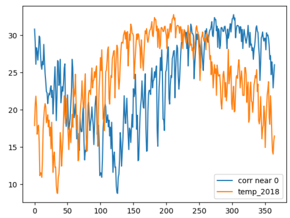
(c) near 0 correlation



(d) max correlation data with label



(e) min correlation data with label



(f) near 0 correlation data with label

Figure 9: max, min and near 0 correlation data with label

9 Step Search Method

9.1 Theory

Alg.1為 Step Search Method 的算法，用於尋找函數最小值。首先，算法需要接收以下輸入參數：函數 f，初始搜索範圍起點 x_0 和終點 x_1，以及一個閾值 $Threshold$，用於判斷何時停止搜索。

Algorithm 1 Step Search Method

Step-Search(f, x_0, x_1, $Threshold$)

Set Init Δ, $n = 1$, $f_{min,0} \to \infty$;
while $f_{min,n-1} - f_{min,n} > Threshold$ **do**
  Determine $f(x_0)$, $f(x_0 + \Delta)$, $f(x_0 + 2\Delta)$, ..., $f(x_1)$
  Find c such that $f(x_0 + c\Delta) < f(x_0 + (c-1)\Delta)$ *and* $f(x_0 + c\Delta) < f(x_0 + (c+1)\Delta)$
  Set $f_{min,n} = f(x_0 + c\Delta)$
  Set $x_0 \Leftarrow x_0 + (c-1)\Delta$, $x_1 \Leftarrow x_1 + (c+1)\Delta$
  $\Delta \Leftarrow \Delta / S$
  n++
end while
$x = x_0 + c\Delta$
$y = f(x)$
return x, y

以下將逐步解釋每個步驟：

1. 初始化變量，設定初始步長 Δ，將迭代次數 n 設為 1，並將 $f_{min,0}$ 設為正無窮大。這些變量將在算法的迭代過程中被更新。
2. 進入一個循環，直到 $f_{min,n-1} - f_{min,n}$ 的差值小於閾值 $Threshold$ 時停止。
3. 在每次迭代中，首先計算函數 f 在 $x_0, x_0 + \Delta, x_0 + 2\Delta$，一直到 x_1 處的值。
4. 找到一個位置 c，滿足 $f(x_0 + c\Delta) < f(x_0 + (c-1)\Delta)$ 並且 $f(x_0 + c\Delta) < f(x_0 + (c+1)\Delta)$。這個位置 c 代表了當前搜索範圍內的極小值點的位置。
5. 將 $f_{min,n}$ 更新為 $f(x_0 + c\Delta)$，即當前搜索範圍內的最小函數值。
6. 更新搜索範圍，將 x_0 更新為 $x_0 + (c-1)\Delta$，將 x_1 更新為 $x_1 + (c+1)\Delta$。這樣，下一次迭代將在新的搜索範圍內進行。
7. 將步長 Δ 更新為 Δ/S，其中 S 是一個縮放因子。通過縮小步長，可以使算法更加精細地搜索極小值點。
8. 重複步驟 2-7，直到 $f_{min,n-1} - f_{min,n}$ 的差值小於閾值 $Threshold$，表示已經找到了一個滿足要求的極小值點。
9. 返回找到的極小值點的坐標，$x = x_0 + c\Delta$，以及對應的函數值 $y = f(x)$。

總體而言，這個算法通過不斷縮小搜索範圍和步長來逐步逼近函數的極小值點。它在每次迭代中通過比較函數值來確定當前搜索範圍內的極小值點位置，並不斷更新搜索範圍和步長以獲得更準確的結果。

9.2 Simulation

以 $f(x) = 2(x-1.5)^2 - 2.5$ 作為目標函數，並使用 Step Search 來找最小值，可以觀察到 Fig.10是可以找到最小值 $(1.5, -2.5)$。執行時間為 $0.001s$。

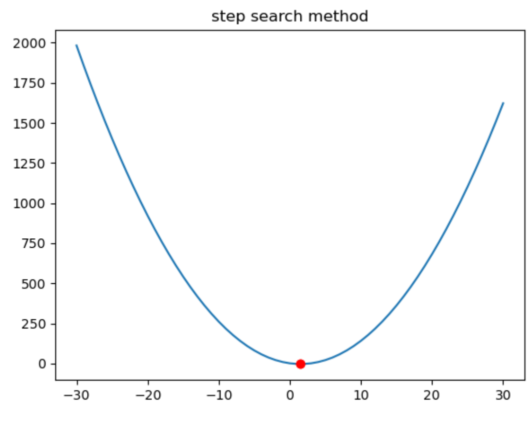


Figure 10: step search method result

10 Golden Search

10.1 Theory

Alg.2為黃金搜索（Golden Search）算法。它用於在一個區間內尋找一個函數的最小值。首先算法需要接收以下輸入參數：函數 f，初始搜索範圍起點 x_0 和終點 x_1，以及一個閾值 $Threshold$，用於判斷何時停止搜索。

Algorithm 2 Golden Search

Golden Search(f, x_0, x_1, $Threshold$)

Let $e = \frac{-1+\sqrt{5}}{2}$, $x_2 = x_0 + \frac{x_1 - x_0}{1+e}$, $x_3 = x_0 + \frac{x_2 - x_0}{1+e}$
while $|x_0 - x_1| > Threshold$ **do**
  if $f(x_2) < f(x_3)$ **then**
    $x_0 = x_3$
    $x_3 = x_2$
    $x_2 = x_1 - \frac{x_1 - x_3}{1+e}$
  else
    $x_1 = x_2$
    $x_2 = x_3$
    $x_3 = x_0 + \frac{x_2 - x_0}{1+e}$
  end if
end while
$x = \frac{x_1 + x_0}{2}$
$y = f(x)$
return x, y

以下將逐步解釋這個偽代碼的每個步驟：

1. 首先，設定一個常數 e，它的值是 $(\sqrt{5}-1)/2$，然後計算兩個中點 x_2 和 x_3，它們是根據黃金分割法得出的。具體來說，x_2 被設置為 x_0 和 x_1 的黃金分割點，而 x_3 被設置為 x_0 和 x_2 的黃金分割點。
2. 進入一個循環，直到 x_0 和 x_1 之間距離差小於閾值 $Threshold$ 時停止。
3. 接下來，評估函數 f 在 x_2 和 x_3 上的值，並比較它們。如果 $f(x_2)$ 小於 $f(x_3)$，則執行以下步驟，更新 x_0 的值為 x_3，更新 x_3 的值為 x_2，計算一個新的 x_2，它是 x_1 和 x_3 的黃金分割點。如果 $f(x_2)$ 不小於 $f(x_3)$，則執行以下步驟，更新 x_1 的值為 x_2，更新 x_2 的值為 x_3，計算一個新的 x_3，它是 x_0 和 x_2 的黃金分割點。
4. 當循環結束時，計算 x 和 y 的值。其中，x 是區間的中點 $(x_1+x_0)/2$，y 是在 x 上計算函數 f 得到的值。
5. 最後，返回的結果是找到的極小值點的座標 (x, y)。

10.2 Simulation

以 $f(x) = 2(x-1.5)^2 - 2.5$ 作為目標函數，並使用 Golden Search 來找最小值，可以觀察到 Fig.11是可以找到最小值 $(1.5, -2.5)$，並可以觀察到 x_2 和 x_3 在每次迭遞後逐步收斂到最小值。執行時間為 $0.0009s$，因為不用每個區間做計算，所以比 Step Search 快。

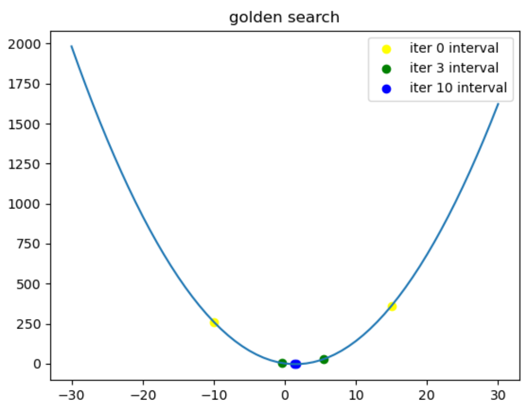


Figure 11: golden search result result

11 Newton Method

11.1 Theory

Alg.3為牛頓法（Newton Method）。首先算法需要接收以下輸入參數，函數 f、初始估計值 x、為維持收斂的穩定性，需要加一個 $\lambda < 1$ 和迭代次數 n。

Algorithm 3 Newton Method

Newton Method(f, x, λ, n)

```
1: for i=0; i<n; i++ do
2:     x = x − λ f'(x)/f''(x)
3: end for
4: y = f(x)
5: return x, y
```

以下將逐步解釋這個偽代碼的每個步驟：

1. 使用 for 循環進行迭代，從 $i = 0$ 開始，直到 i 達到 n，執行以下步驟。
2. 更新估計值，使用牛頓法的更新公式 $x = x - \lambda \frac{f'(x)}{f''(x)}$，將當前的估計值 x 更新為新的值。在求解最小值的情況下，這個更新步驟使用函數的一階導數和二階導數來調整估計值，以使其朝著函數最小值的方向移動。具體而言，我們根據當前估計值處的斜率和曲率來調整估計值，使其向梯度下降的方向前進。
3. 迭代完成後，得到最終的估計值 x。
4. 計算最終的函數值，將最終的估計值 x 代入函數 f 中，計算函數值 $y = f(x)$。
5. 最後，返回最終的估計值 x 和函數值 y。

在原始的牛頓法中，我們通過求解方程 $f(x) = 0$ 來找到根，但是在求解最小值的情況下，我們需要調整估計值的更新公式和迭代終止條件。常見的做法是使用函數的一階導數和二階導數來更新估計值，並在達到某個停止準則時終止迭代，如梯度接近零或達到預設的迭代次數。

11.2 Simulation

以 $f(x, y) = (0.5x + 1)^2 + 0.5y^2 + 10$ 作為目標函數，(x_0, y_0) 設為 $(6, 8)$、$\lambda = 0.5$、$n = 50$，可以觀察到 Fig.12是可以找到最小值 $(-2, 0, 10)$，並可以觀察到 Newton Method 在每次迭遞後會逐步接近最小值。執行時間為 $0.0019s$。

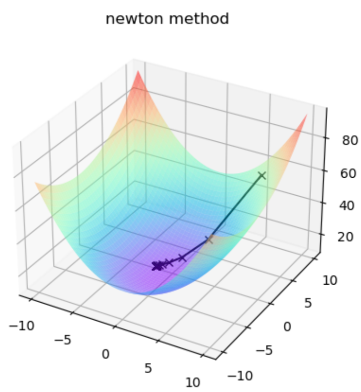


Figure 12: newton method result

12 Gradient Descend

12.1 Theory

Alg.4為梯度下降算法（Gradient Descend）的偽代碼。梯度下降算法是一種用於優化函數的迭代方法，通過不斷調整自變量的值來最小化目標函數。首先算法接受四個輸入參數，函數 f、x 為初始值，即算法開始時的起始點、η 為學習率（learning rate）是控制每次迭代步長的參數和 n 為迭代次數，確定算法運行的迭代次數。

Algorithm 4 Gradient Descend

Gradient Descend(f, x, η, n)

```
for i=0; i<n; i++ do
    x = x − η ▽ f(x)
end for
y = f(x)
return x, y
```

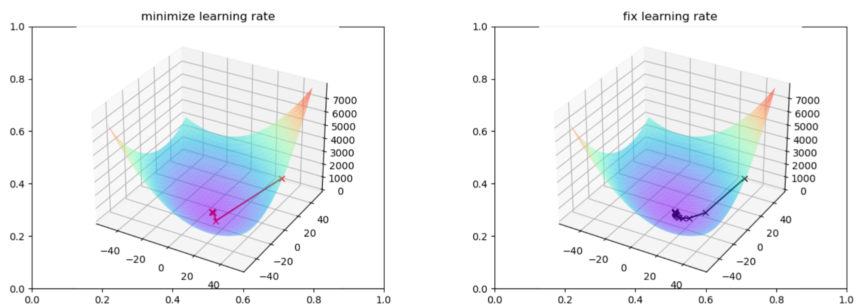
算法的主要步驟如下：

1. 使用 for 循環進行迭代，從 $i = 0$ 開始，直到 i 達到 n，執行以下步驟。
2. 計算當前 x 的梯度 $\triangledown f(x)$。然後更新 x 的值，通過從 x 中減去學習率 η 乘以梯度 $\triangledown f(x)$ 得到新的 x 值。這一步是梯度下降算法的核心步驟，通過向負梯度方向調整 x 的值來逐步逼近函數的最小值。
3. 最後，結束 for 循環返回 x 和 y。

總結來說，該算法通過迭代調整 x 的值，以找到使函數最小化的 x 值。每次迭代中，通過計算目標函數在當前 x 處的梯度，來確定下降的方向，並通過學習率控制下降的步長。最終，算法返回找到的 x 和對應的最小值的 y。

12.2 Simulation

以 $f(x,y) = (x-1)^2 + (y+5)^2 + xy$ 作為目標函數，(x_0, y_0) 設為 $(49, 10)$，並設定 $n = 100$，當得出 gradient 的 norm 小於某個 $threshold = 10^{-6}$ 時會停止迭遞，learning rate 分為兩部分進行試驗，一部分是每次迭遞都算出最小值的 learning rate，這裡使用 Golden Search，另一個是 fix learning rate 為 0.2，可見得 Fig.13為其路徑，其中 13(b)的迭遞次數較 13(a)多。但執行時間並沒有較快，因為需要消耗額外的時間在做 Golden Search。



(a) minimize learning rate

(b) fixe learning rate

Figure 13: gradient descend result

13 L_α-Norm

13.1 Theory

L_α-Norm 的方法跟前面 Linear 和 Non-Linear Prediction 計算 MSE 很像，但是在這裡就不用 MSE，而是用 L_α-Norm 來計算誤差的距離，以下 Eq.22為 L_α-Norm。

$$\|z\|_\alpha = (\sum_n |z[n]|^\alpha)^{1/\alpha} \tag{22}$$

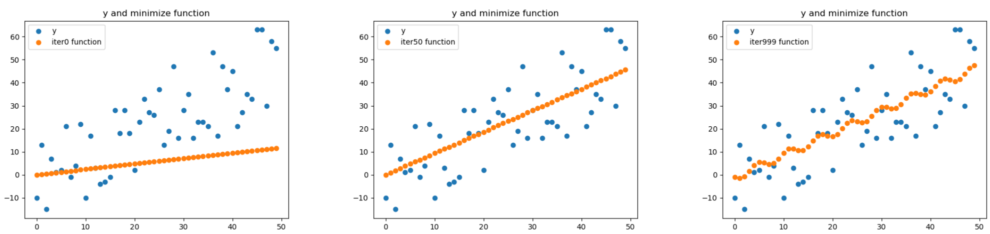
我們該如何使用 L_α-Norm 來最優化，以下有一個例子說明：

$$E = \|y[n] - x_1 b_1[n] - x_2 b_2[n] - x_3 b_3[n]\|_\alpha \tag{23}$$

如上式 Eq.23，假設有一個目標函數 $y[n]$，其中用 $b_1[n]$、$b_2[n]$ 和 $b_3[n]$ 的線性組合來逼近 $y[n]$，我們需要找出令 Eq.23最小的常數 x_1、x_2 和 x_3，首先，可以對 Eq.23取偏微分，從而求後最小值。另外，求 $\|y[n]-x_1b_1[n]-x_2b_2[n]-x_3b_3[n]\|_\alpha$ 的最小值是等價於 $\|y[n]-x_1b_1[n]-x_2b_2[n]-x_3b_3[n]\|_\alpha^\alpha$ 的最小值，從而簡化其運算。

13.2 Simulation

以 $f(n) = n + r$ 作為目標函數，其中 r 為-20 至 20 的隨機數，用 $b_1(x) = sin(x)$、$b_2(x) = 0.5x$ 和 $b_3(x) = 1$ 進行線性組合。得出 Fig.14結果，其中 iteration 增加，$f_{opt}(n) = x_1 b_1[n] - x_2 b_2[n] - x_3 b_3[n]$ 的會慢慢迫近 $f(n)$，如圖 Fig.14(a)、Fig.14(b)、Fig.14(c)，它的 E 變化為 287、153 和 151。



(a) iter 0 (b) iter 50 (c) iter 999

Figure 14: L_α-Norm result

References

[1] 專題資料

[2] https://www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-major-cities