Machine Learning (Homework 2) Report
0860908 李少琪

1. **Sequential Bayesian Learning**

過程:

Step.1 $\emptyset_j(x) = \frac{1}{1+e^{\left(-\frac{x-\mu_j}{s}\right)}}$, $s = 0.1,\ \mu_j = \frac{2j}{M},\ M = 3,\ j = 0, \dots, (M-1)$

$$\rightarrow\ \Phi \in \mathbb{R}^{100\times 3}$$

Step.2 prior distribution $p(w) = N(w \mid m_0, S_0)$

$$m_0 = 0, S_0 = 10^{-6} I$$

以下步驟, X 中選前 N 個做訓練(N=5,10,30,80)

Step.3 posterior distribution $p(w|\boldsymbol{t}) = N(w \mid m_N, S_N), \beta = 1$

$$S_N = (S_0^{-1} + \beta \Phi^T \Phi)^{-1}$$
$$m_N = S_N(S_0^{-1} m_0 + \beta \Phi^T \boldsymbol{t})$$

Step.4 從 posterior distribution 隨機選出 5 組 random variables 作為 weights

$$\rightarrow w \in \mathbb{R}^3$$

Step.5 $y = w\Phi^T$, 計算 5 組 Root mean square error 的平均, 並劃出預測結果的曲線

Step.6 選前兩個 weight 畫出對應的 prior distribution

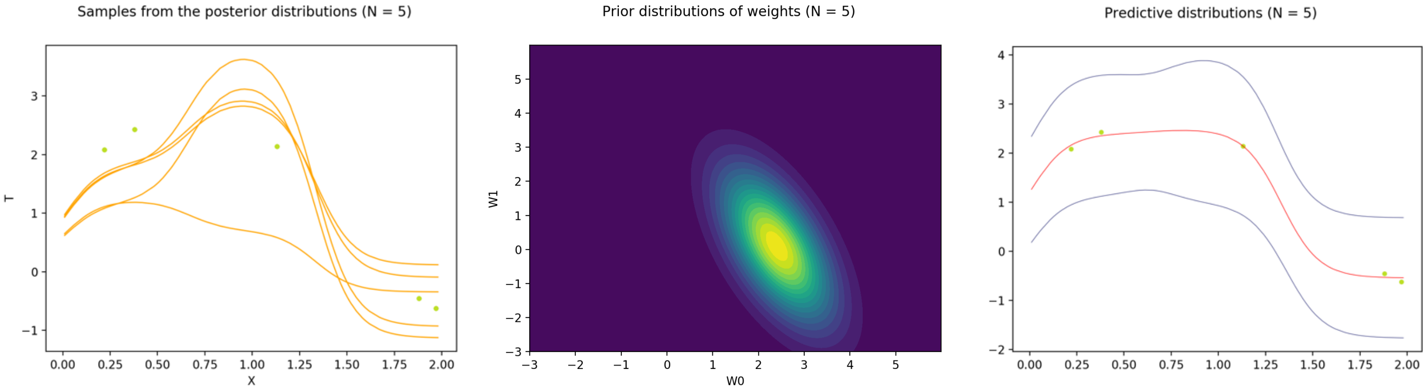
Step.7 predictive distribution $p(t|\mathrm{x}, \boldsymbol{t}, \beta) = N(t|m_N^T \emptyset(\mathrm{x}), \sigma_N^2)$

$$\sigma_N^2 = \frac{1}{\beta} + \emptyset(\mathrm{x})^T S_n \emptyset(\mathrm{x})$$

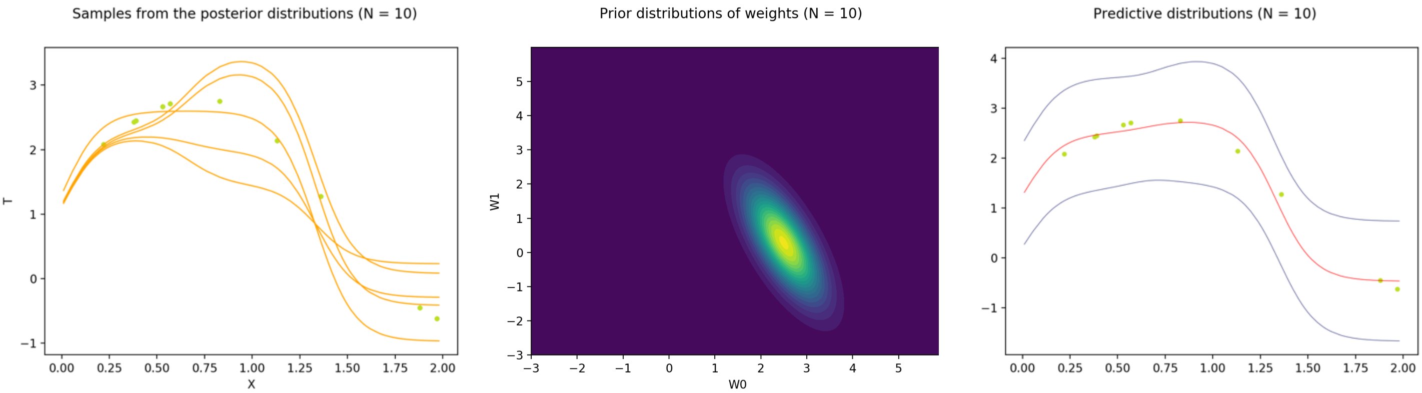
Step.8 畫出 predictive distribution (上界 mean+stddev, 下界 mean-stddev)

結果:

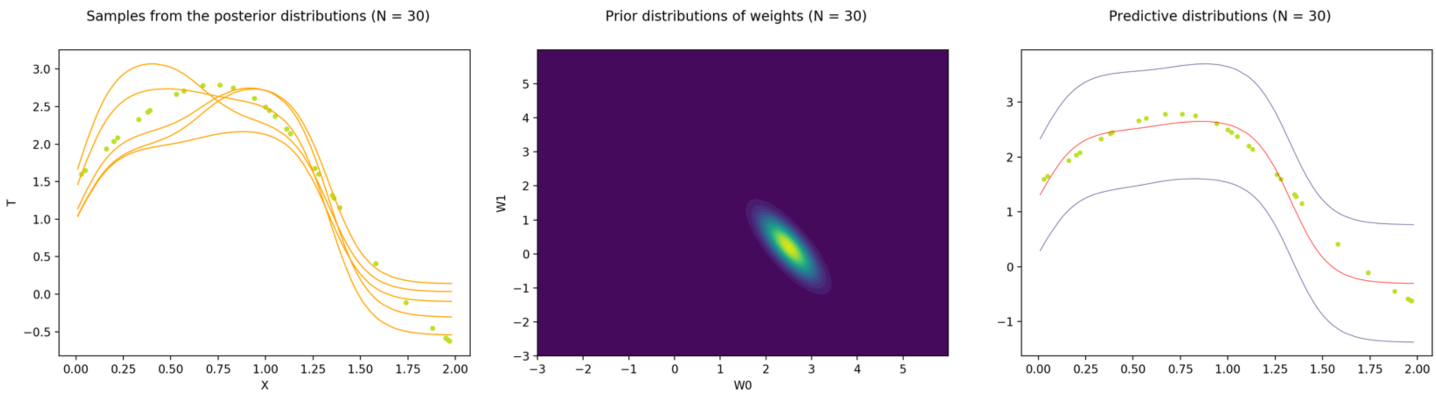
N = 5, RMS = 17.12



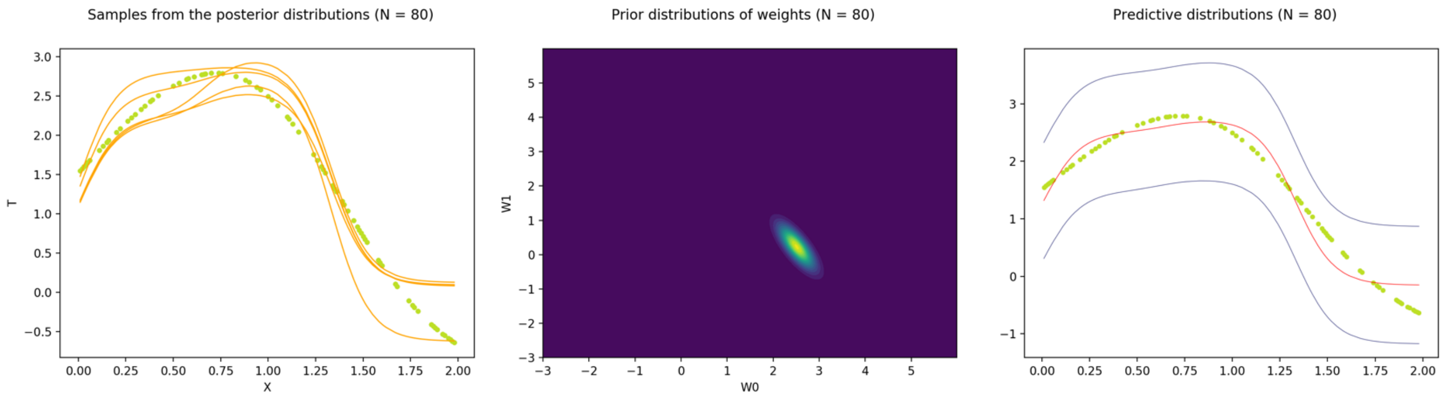
N = 10, RMS = 16.37



N = 30, RMS = 15.88



N = 80, RMS = 15.79



分析:

由上述的結果可以看出，當訓練資料越多，fit 出來的曲線越接近訓練資料，權重的高斯機率分佈的標準差也越來越小，預測的機率分佈的標準差也越來越小。

2. Logistic Regression

過程:

Step.1 讀取照片，並轉為一維陣列作為訓練特徵，對這些 features 做 Normalization

$$x' = \frac{x-\mu}{\sigma}$$

Step.2 切出 Training dataset, Testing dataset (每個類別各 5 筆資料)

Step.3 將 target 做 one-hot encoding

Step.4 給定 learning rate, epochs, $w_0=\mathbf{0}$

A. Gradient descent Algorithm

Step.5 $w^{\tau+1} = w^{\tau} + \eta\Phi\boldsymbol{t}$

Step.6 Softmax transformation (將 predict 的 target 轉為 0~1, 屬於該類的機率)

$$p(C_k|\emptyset) = y_k(\emptyset) = \frac{e^{a_k}}{\sum_j a_j}$$

Step.7 計算 Accuracy, Cross-entropy $E(W) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} ln y_{nk}$

Step.8 predict the class of testing data

B. Principal component analysis (PCA)

Step.9 選出 n 個特徵值最大對應的 eigenvectors

Step.10 PCA_x = x * eigenvectors

Step.11 畫出特徵向量 (由 mean 指向 mean+stddev*eigenvector)

C. Newton-Raphson Algorithm

Step.12 用 PCA 降維到 2,5,10

Step.13 $w^{\tau+1} = (\Phi^T R\Phi)^{-1}\Phi^T Rz$

$$R_{NN} = y_n(1-y_n)$$
$$z = \Phi w^{\tau} - R^{-1}(\boldsymbol{y}-\boldsymbol{t})$$

Step.14 計算 Accuracy, Cross-entropy

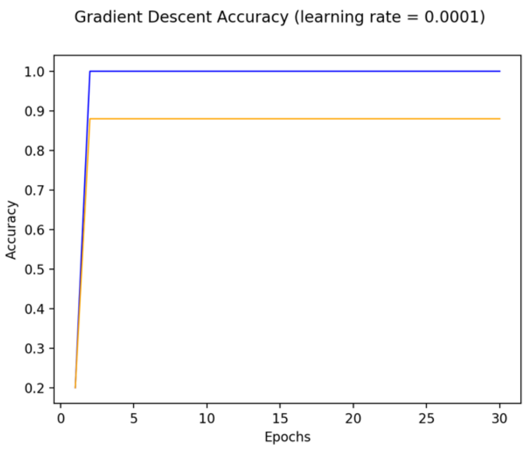
結果:

Gradient descent Algorithm (epochs=30)

Learning rate= 0.0001, Testing accuracy = 0.88

True: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5]

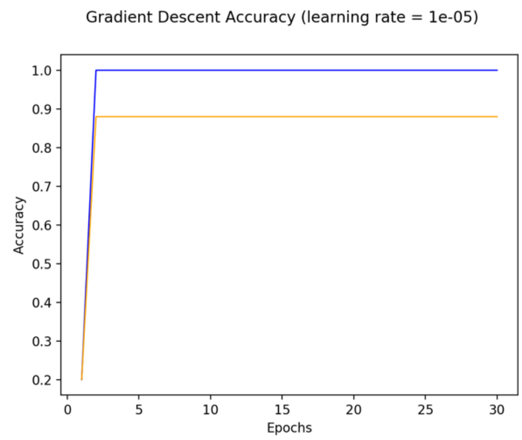
Predict: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4 5 4 5 4]

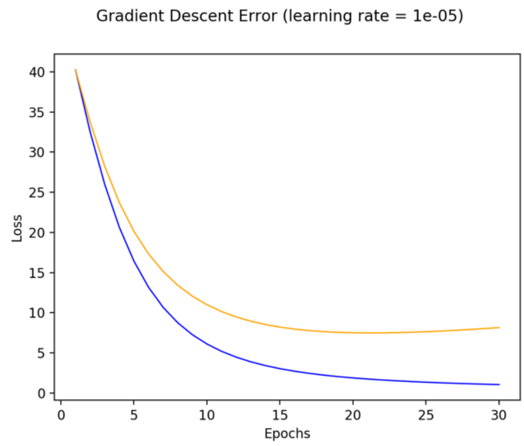


Learning rate = 0.00001, Testing accuracy = 0.88

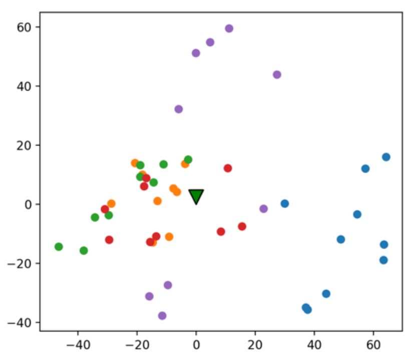
True: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5]

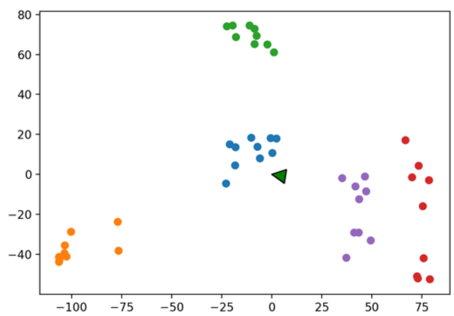
Predict: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4 5 4 5 4]

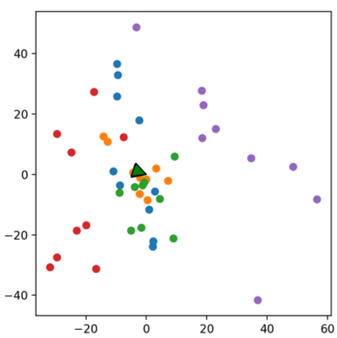


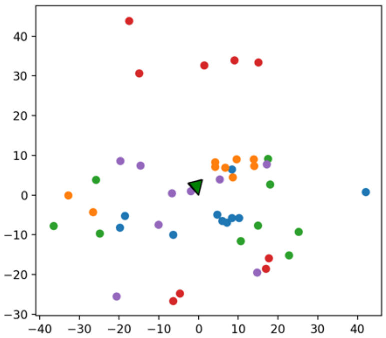


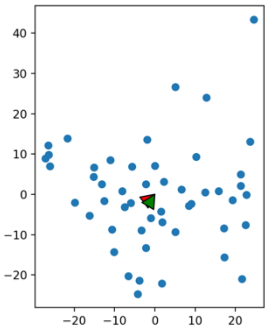
Eigenvectors









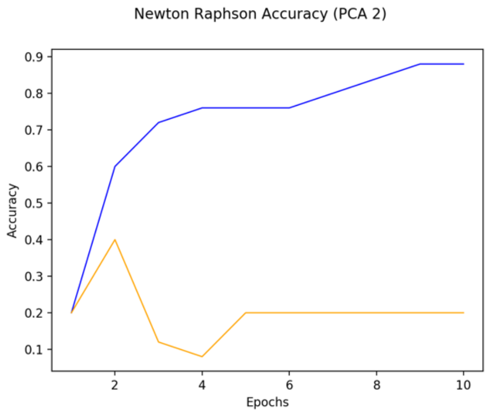


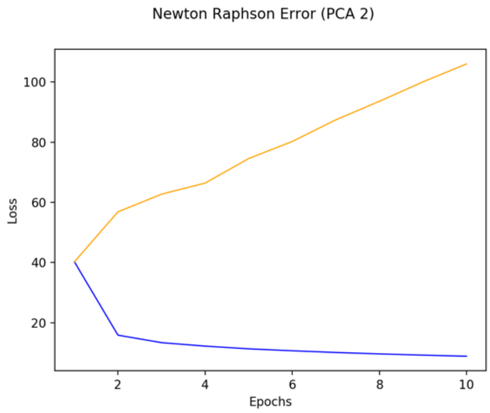
Newton-Raphson Algorithm

PCA dim = 2, Testing accuracy = 0.2

True: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5]

Predict: [2 2 4 5 1 1 1 1 1 1 4 4 2 4 2 5 5 5 5 5 5 5 3 5 5]

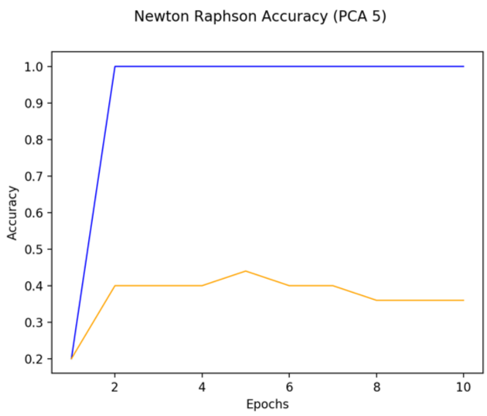


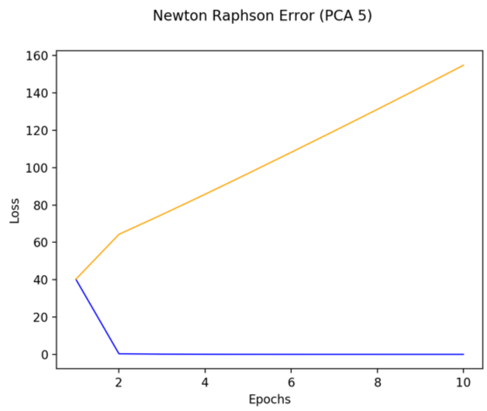


PCA dim = 5, Testing accuracy = 0.36

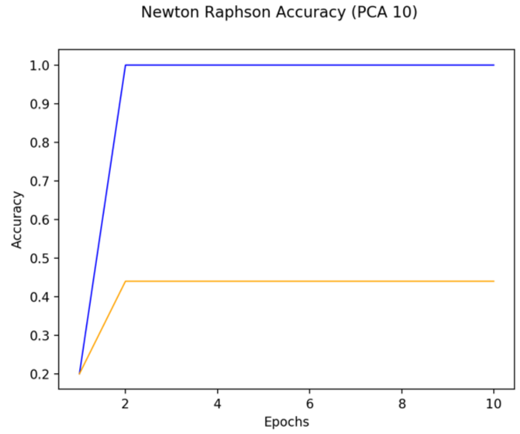
True: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5]

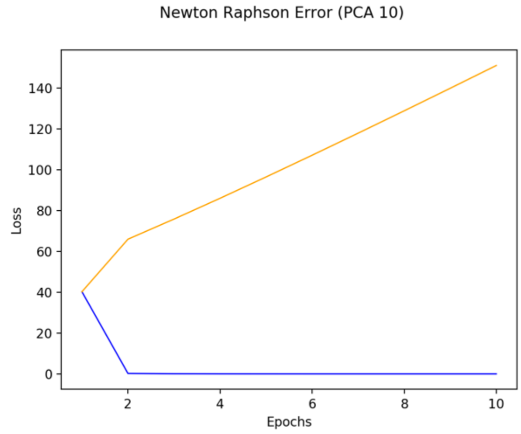
Predict: [1 1 1 1 1 2 3 3 3 3 2 2 2 2 2 4 4 4 5 5 3 4 3 1 3]





PCA dim = 10, Testing accuracy = 0.44
True: [1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5]
Predict: [1 1 5 1 1 2 2 3 2 3 2 2 2 2 2 4 4 4 5 5 3 4 1 1 5]





分析:

從 gradient descent algorithm 的實驗上可以發現，當 learning rate 越大，cross-entropy 下降越快，但是也越容易 overfitting，所以當要選擇 learning rate 時需要避免 loss 下降過快，以及訓練時間過長。

從 newton-raphson algorithm 的實驗可以發現，選擇的特徵越多，Testing data 的表現越好，但若是過多不必要的特徵，也可能影響訓練，所以選擇特徵的數量也需要不斷嘗試。

比較 gradient descent algorithm 和 newton-raphson algorithm，可以發現 gradient descent 訓練結果比較好，可能的因素是 gradient descent 是作一次微分，而 newton-raphson 是作二次微分運算較為複雜，使得 testing 的效果並不好。

3. Nonparametric Methods

過程:

Step.1 讀取.csv，將 Type 1 轉為 Psychic = 0, Normal = 1, Water = 2, 將 Legendary 轉為 False = 0, True = 1

Step.2 Type 1 之後的 columns 作為 features，並作 Normalization

Step.3 切出 Training dataset(120), Testing dataset(38)

Step.4 用 features 計算 Euclidean distance 找出離 testing data 最近的 K 個 training data 作為 neighbors

$$distance(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_m - y_m)^2}$$

Step.5 以 K 個 neighbors 中最多的類別作為 testing data 的結果

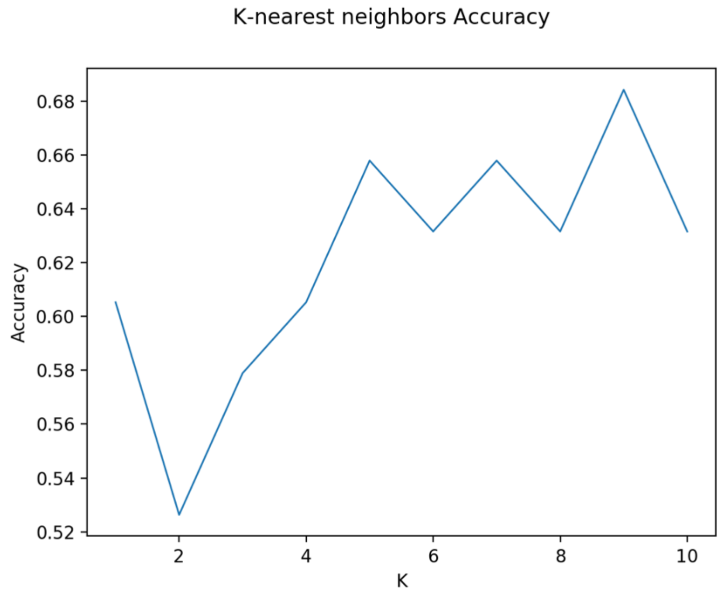
Step.6 計算 accuracy

Step.7 用 PCA 降維至 7,6,5，重複 Step.4~6

結果:

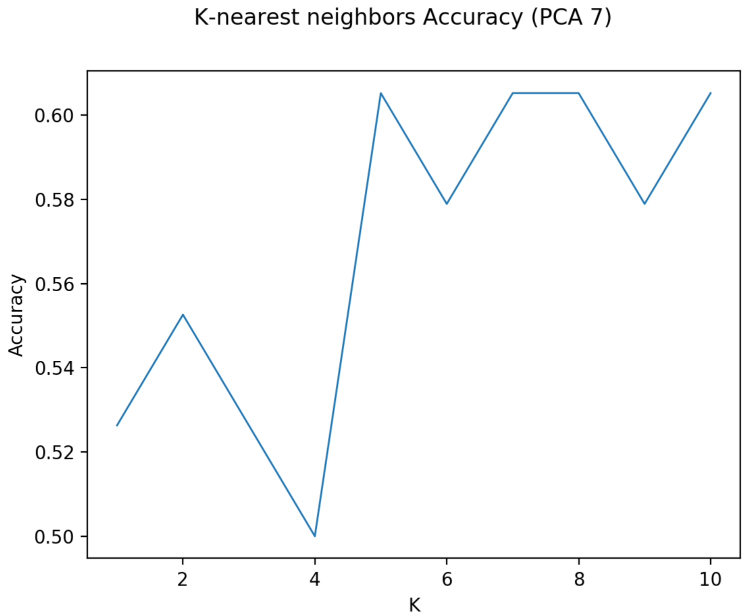
KNN (all features)

K	1	2	3	4	5	6	7	8	9	10
Acc	.605	.526	.579	.605	.658	.632	.658	.632	.684	.632



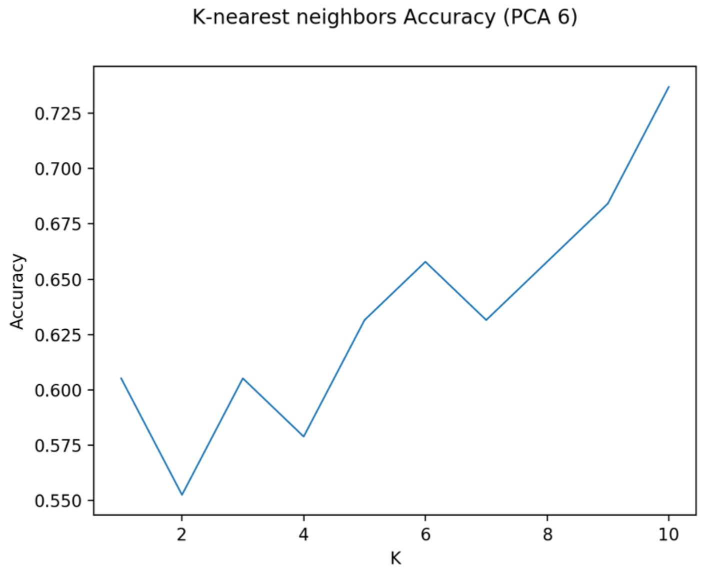
KNN (PCA 7)

K	1	2	3	4	5	6	7	8	9	10
Acc	.526	.553	.526	.5	.605	.579	.605	.605	.579	.605



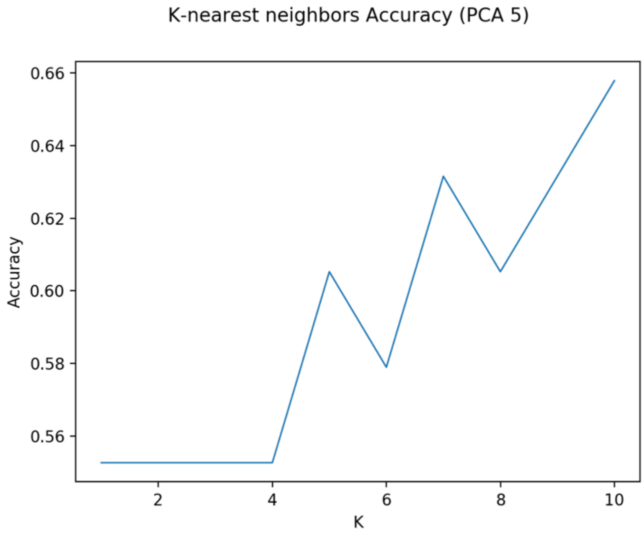
KNN (PCA 6)

K	1	2	3	4	5	6	7	8	9	10
Acc	.605	.553	.605	.579	.632	.658	.632	.657	.684	.737



KNN (PCA 5)

K	1	2	3	4	5	6	7	8	9	10
Acc	.553	.553	.553	.553	.605	.579	.632	.605	.632	.658



分析:

從上述 K-nearest neighbors 的實驗中，可以發現當 K 越小時，準確率越低，但也不能讓 K 太大，有可能因選擇過多鄰居超越分界線，產生 underfitting 的現象。

在 features 量的選擇上，如果計算過多的 features，計算出的距離也會增大，受到非相關性的 feature 也影響更大，找出最具相關性的幾個 features 來做計算最好。