

NTUEE, WAN - CYUAN FAN 范萬泉

“no collaborators, with reference”

Homework 1

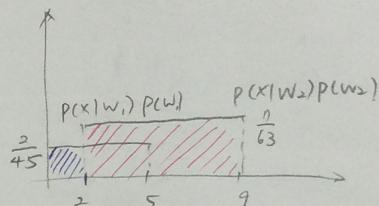
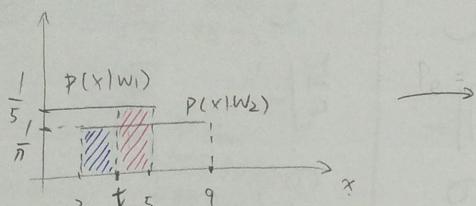
Problem 1 : Bayes Decision Rule

Problem 1. Bayes Decision Rule (20%)

$$P(x|w_1) = \begin{cases} \frac{1}{5-0}, & (0, 5) \\ 0, & \text{o.w.} \end{cases}$$

$$P(x|w_2) = \begin{cases} \frac{1}{9-2}, & (2, 9) \\ 0, & \text{o.w.} \end{cases}$$

$$P(w_1) = \frac{2}{9}, \quad P(w_2) = \frac{7}{9}.$$



Decide w_1 , $P(w_1|x) > P(w_2|x)$; o.w., w_2 .

$$P_e = (5-t) \times \frac{2}{45} = \frac{2}{15}.$$

$$P_e = \int_t^\infty P(x|w_1)P(w_1)dx + \int_0^t P(x|w_2)P(w_2)dx$$

$$= \int_t^5 \frac{1}{5} \times \frac{2}{9} dx + \int_0^t \frac{1}{7} \times \frac{7}{9} dx$$

$$= \frac{2x}{45} \Big|_t^5 + \frac{x}{9} \Big|_0^t = \frac{2}{9} - \frac{2t}{45} + \frac{t}{9} - \frac{2}{9}$$

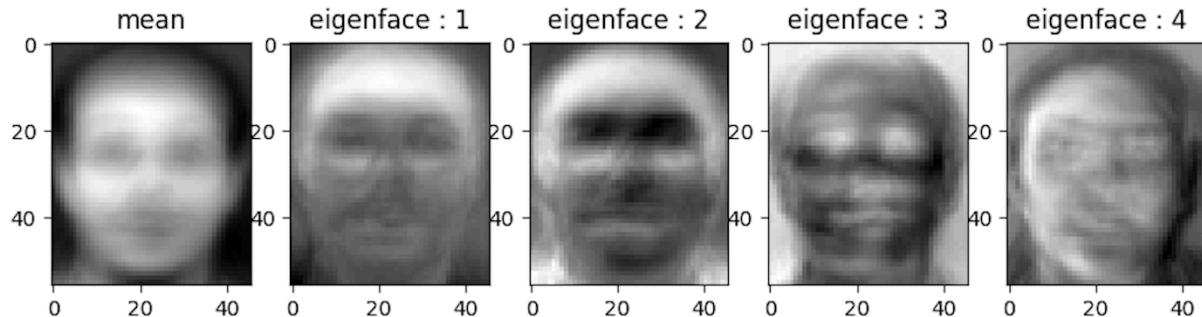
$$= \frac{34}{45}t - \frac{2}{9}$$

$$P_e \min \Rightarrow \min_{2 \leq t \leq 5} \left(\frac{34}{45}t - \frac{2}{9} \right) \quad t = 2 \text{ 时}, \min = \frac{68}{45} = \frac{2}{15}$$

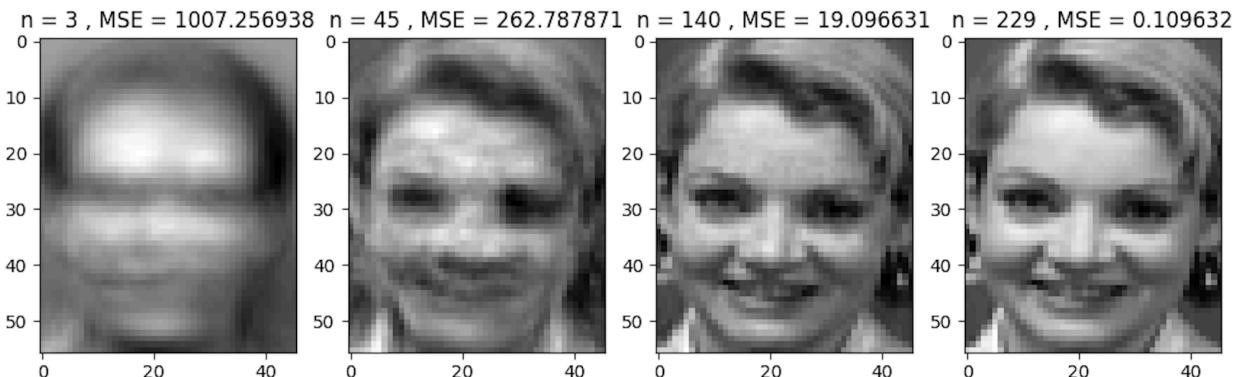
R1 : $x < 2$, R2 : $x \geq 2$

Problem 2 : Principal Component Analysis

- 這部分我將所有資料匯入np.array()中，先利用np.mean()計算平均值，再將此平均值，使用“sklearn.decomposition”套件的PCA.fit()來處理，得到以下：



- 此題，我將上面求出的eigenvectors當作空間的基底，把person1_1影像，投影到該空間向量中，依照不同的eigenvector數量限制投影，最後做圖如下：



- 利用“sklearn.metrics” 中的“mean_squared_error”計算差距，如上圖標號所表示。

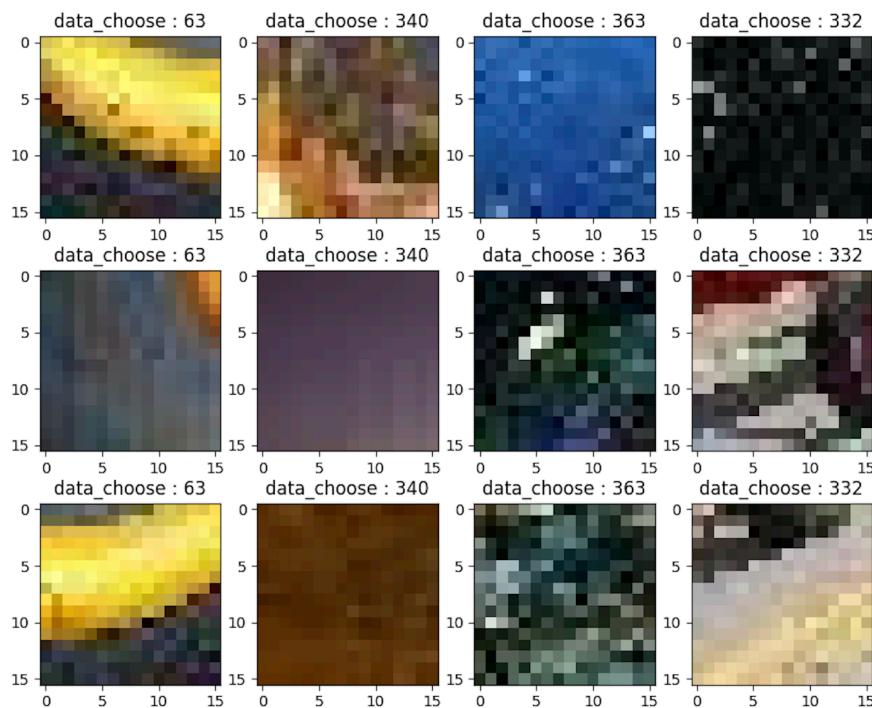
- 這部分我先將所有的training data投影到指定數量的eigenvector上頭，利用eigenVectors.transform()作轉換操作，再接著使用在“sklearn.model_selection”模組中的GridSearchCV(KNN(), param_grid, cv=3)函式找出每一個對應k所得到的結果，此函式可以將資料分成三堆，其中兩堆當作train，而另外一堆當作validation用。回傳結果如下：

Accuracy	$k = 1$	$k = 3$	$k = 5$
$n = 3$	0.704	0.617	0.521
$n = 45$	0.929	0.858	0.792
$n = 140$	0.929	0.858	0.754

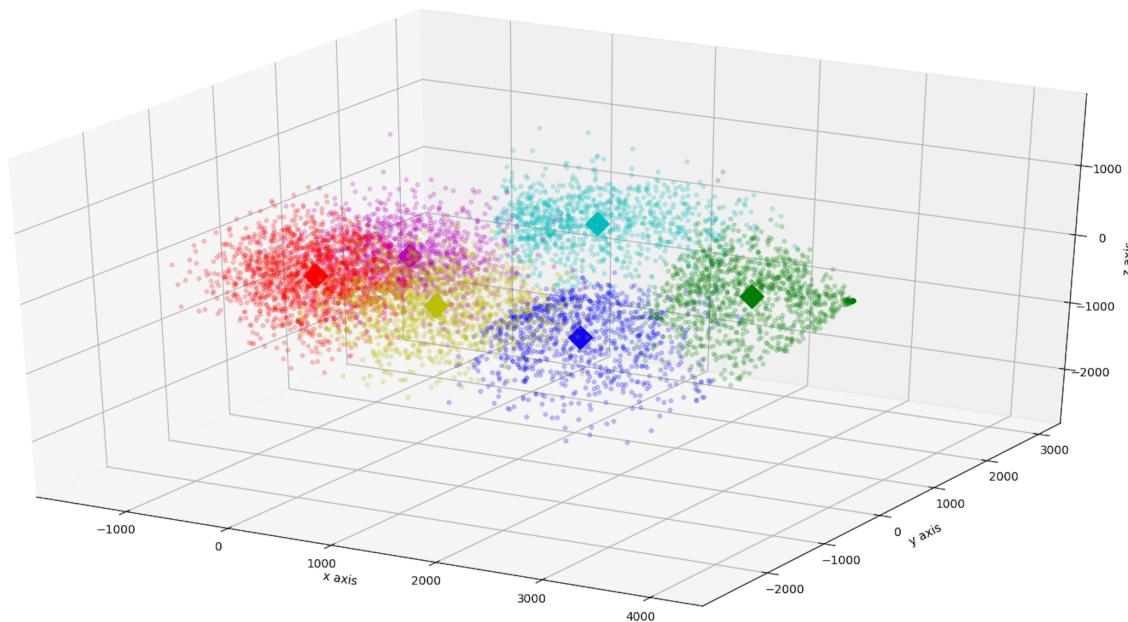
5. 這部分呈上題，因為我們發現最好的分辨效果落在 $(k,n) = (1,45)$ ，因此我們取值 k,n 。接著我們將test data一樣做投影到eigenvectors上頭，再藉由“sklearn.neighbors”模組中的函式“KNeighborsClassifier”，此我們用training data以及 k,n 參數訓練好後，使用 $kN\text{-}N.\text{predict}()$ 預測testing data的結果，再跟正確的答案比對，看Accuracy多少，測試出後 Accuracy : 0.95625。

Problem 3 : Visual Bag-of-Words

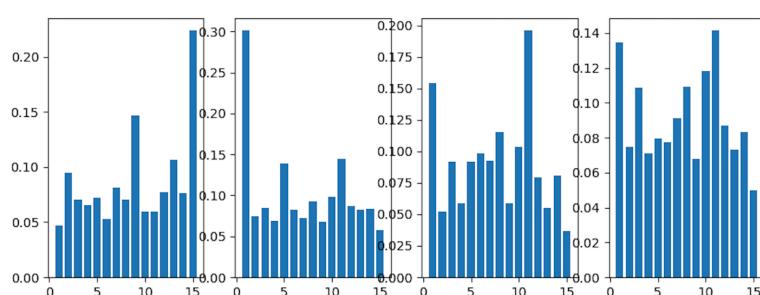
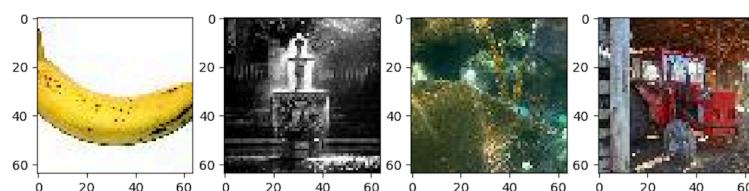
1. 這部分比較沒有什麼模組套用，我使用的方法就是先將資料讀到一個np.array()中，基本training data.shape = (1500,64,64,3)，testing data的方法一樣，然後我們做兩次for迴圈，將每一筆照片資料每16*16單位的patch資料擷取放到一個新的np.array()中，因此會形成shape = (24000,16,16,3)的patches資料大小，再將後面的個別patches的資料 reshape = (24000,768)存起來後面使用，此部分隨機取照片，就使用random完成，如下圖所示：其照片分別為[“banana”, “fountain”, “reef”, “tractor”]，基本上已經不太容易直接從這街patches看出來原本是什麼，因為我們不見得可以選到這街圖片中關鍵的feature，像這裡香蕉還可以分辨，因為我們恰好擷取到關鍵「蕉身」。



2. 第二部分我們承接第一題中已經分好的training data然後直接使用k mean alg.將資料不斷的平均後分堆，差別於 $k = 15$ 組，並且限定iterations = 5000，我們可以馬上得到15個center數值。在使用Problem 2使用到的PCA降維，降到最主要的3個eigenvectors上頭，使用“pca.fit_transform()”直接做投影，有別於fit()這個函式可以直接投影到我們要的eigenvectors上頭，再接著我們利用random隨意選取6組center以及training data，用“mpl_toolkits.mplot3d”模組的3D scatter繪圖功能，轉換成三維圖像如下：



3. 這部分分成兩個重點：(1) 利用patches與centers的“Euclidean distance”當作轉換介質，找出每個patches 對應的BoW轉換(15維度向量，因為一共有15個centers) (2)再使用soft-assignment + Max pooling strategy也就是每一個照片一共會有16個patches，在每一個對應center計算出來的BoW轉換值上只取出最大的那個數值，因此我們最後可以得到每一個照片一組15維度的向量轉換值(1500,C=15)，再取出每個種類一組做比較，如下：



4. 最後我們一樣將testing data與centers做距離運算並轉倒數成normalization形式，也可以得到testing data = (500, 15)的np.array()，這樣我們便可以用kNN的技巧，k=5，將training data放入kNN model中訓練，分堆後再把testing data放到訓練好的model中predict，可以得到如下正確率：

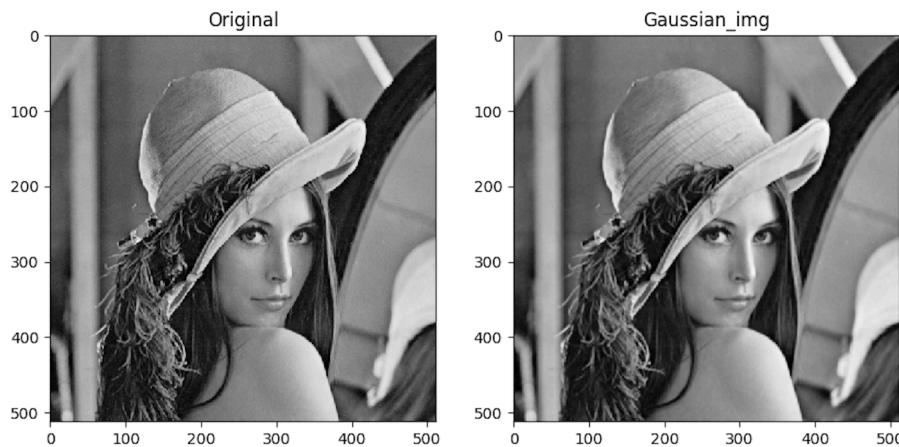
Total Accuracy	0.52
Banana Accuracy	0.696
Fountain Accuracy	0.424
Reef Accuracy	0.624
Tractor Accuracy	0.336

Problem 4 : Image Filtering

1. 證明如下：

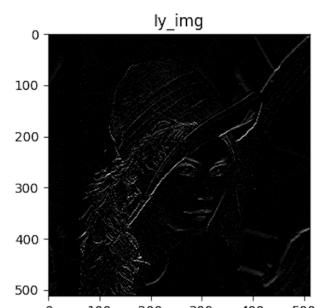
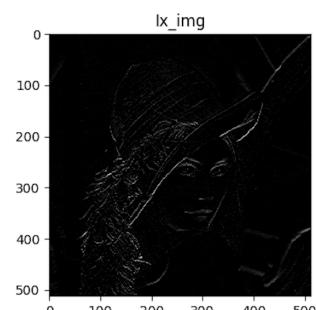
$$\begin{aligned}
 g(x, y) &= f(x, y) * G(x, y) \\
 &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(z_u, z_v) G(x - z_u, y - z_v) dz_u dz_v \\
 &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(z_u, z_v) \cdot \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x-z_u)^2 + (y-z_v)^2}{2\sigma^2}} dz_u dz_v \\
 &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(z_u, z_v) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_u-x)^2}{2\sigma^2}} dz_u \right] \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-z_v)^2}{2\sigma^2}} dz_v \\
 &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(z_u, z_v) \cdot G(x - z_u) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-z_v)^2}{2\sigma^2}} dz_v \\
 &= \int_{-\infty}^{\infty} [f(x, z_v) * G(x)] \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(y-z_v)^2}{2\sigma^2}} dz_v \\
 &= [f(x, y) * G(x)] * G(y)
 \end{aligned}$$

2. 此題僅作filter，因此我們利用openCV提供的“cv.GaussianBlur()”函式直接將設定設好後輸出圖片，基本上就是做簡單的convolution，圖如下：其效果會使圖片的一些不均勻雜訊消除，就如同高斯分佈，會將超出平均太多的點降低影響，使整個影像看起來相對較平滑。

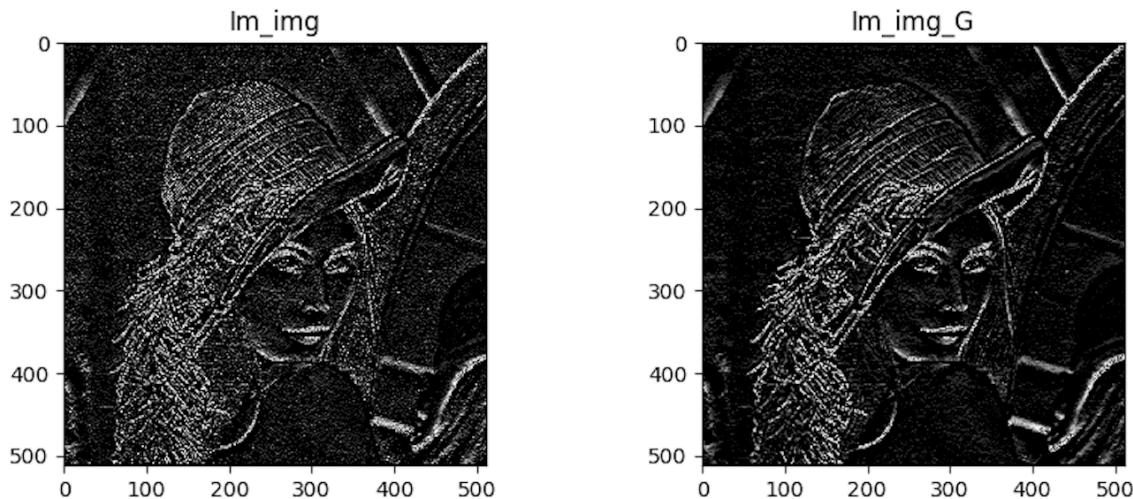


3. 證明如下圖，另外得出式子後，利用類似第二題的方法我們使用”cv.filter2D()”函式，將圖片做filter，成型圖片如下：

$$\begin{aligned}
 I_x(x,y) &= \frac{\partial I}{\partial x} = \frac{1}{2}(I(x+1,y) - I(x-1,y)) \\
 I_y(x,y) &= \frac{\partial I}{\partial y} = \frac{1}{2}(I(x,y+1) - I(x,y-1)) \\
 I_x &\in I * k_x \quad (f*g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m] \\
 &= I * \left(\frac{1}{2} [\delta(x+1) - \delta(x-1)] \right) \\
 &= \sum_{m=-\infty}^{\infty} I[m,y] \cdot \frac{1}{2} [\delta(x-m+1) - \delta(x-m-1)] \\
 &= \frac{1}{2} (I[x+1,y] - I[x-1,y]) \\
 k_x &= \frac{1}{2} [\delta(x+1) - \delta(x-1)] \\
 \text{同理 } k_y &= \frac{1}{2} [\delta(y+1) - \delta(y-1)]
 \end{aligned}$$



4. 接下來，承上題我們帶入公式後，得到一組新的filter，然後分別將原圖與Gaussian作圖過濾filter後，呈現如下：我們可以發現，經過Gaussian filter後的圖片，在gradient magnitude image會比較少雜點，這是因為Gaussian filter的用處就是減少雜訊，使整體圖片更對比度下降、在edge的地方可以稍微模糊圓滑。因此反映在gradient上頭會有較少的雜點。



My Code is available on my Github :

“https://github.com/Christine0312/Computer-vision-project/tree/master/mini_01”

reference:

1. *Sklearn*, “<https://scikit-learn.org/stable/documentation.html>”
2. *Python tutorial*, “<https://docs.python.org/3/tutorial/>”
3. *Matplotlib*, “https://matplotlib.org/users/pyplot_tutorial.html”
4. *PCA*, “<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>”
5. *openCV*, “https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html”