**EIE4105 Multimodal Human Computer Interaction Technologies**

**Lab 2: SVM, PCA and LDA for HCI (Colab version)**

**A. Objectives and Outcomes**

After finishing this lab, you should be able to

* Use support vector machines (SVMs) to build HCI systems.
* Understand the properties of different types of SVMs
* Understand how principal component analysis (PCA) and linear discriminant analysis (LDA) can be used for dimension reduction
* Use LDA to build HCI systems.

**B. Assessment Criteria**

* Ability to build pattern classification systems based on SVM, PCA, and LDA.
* Ability to produce correct results.
* Ability to explain the capability of different machine learning algorithms.
* Clarity of the report.

**C. Submission**

* Copy and paste the graphs and images that you obtain from Google Colab to a Word file and convert it to PDF.
* Submit your report to Blackboard before the deadline specified in Blackboard.

**D. Handwritten Digit Recognition**

This lab will use the same dataset in Lab 1. Finish Lab 1 before starting this one.

**E. Procedures**

***E.1 Support Vector Machines Classifier***

1. We will use Google Colab as the development environment.[[1]](#footnote-1) Colab runs on browsers. You need a Google account to use Colab. If you do not have one, visit https://support.google.com/mail/answer/56256?hl=en. The advantage of Colab is that no installation is required. The disadvantage is that you may need to remount the Google Drive after the session expired.
2. Display the Google Drive (https://drive.google.com/drive/) page in your browser. Create the following directory structures in your Google Drive:

My Drive/Learning/EIE4105/lab2/data

My Drive/Learning/EIE4105/lab2/python

After creating the folders, you should see something like this:

图形用户界面, 应用程序, 表格

描述已自动生成

Fig. 2: Directory structure in Google Drive

1. Go to http://bioinfo.eie.polyu.edu.hk/download/EIE4105/lab2/python and download all files in this folder and upload them to your Google Drive under “My Drive/Learning/EIE4105/lab2/python”.
2. Go to <http://bioinfo.eie.polyu.edu.hk/download/EIE4105/lab2/data> to download the file “faces.mat”. upload it to your Google Drive under “My Drive/Learning/EIE4105/lab2/data”.
3. Start reading from the file svm.ipynb.

***How many SVMs in this Handwritten Digit Recognizer? Draw a block diagram illustrating the architecture of the recognizer.***

1. Linear SVMs can be used to classify high-dimensional data, even when the number of training samples is significantly smaller than the feature dimension. Create this situation by editing svm.ipynb so that the number of training samples per digit is 10 (nSamples = 10) and that the SVM kernel is ‘linear’.

from mnist import load\_SampleMnist

nSamples = 10

trainpath = '../../lab1/data/noisy\_train\_digits.mat'

testpath = '../../lab1/data/noisy\_test\_digits.mat'

train\_data, train\_labels, test\_data, test\_labels =

load\_SampleMnist(trainpath,testpath,nSamples)

svm = LinearSVC(dual=True, multi\_class='ovr', loss='hinge')

svm.fit(train\_data, train\_labels)

pred = svm.predict(test\_data)

acc = accuracy\_score(test\_labels, pred)

print(accuracy)

Run the script svm.ipynb. Create a more extreme situation where the number of training samples per class to 1 and re-run the script.

***Report the accuracy for both ‘clean’ and ‘noisy’ digits. How many training samples have been used for training each of the SVMs? Explain why the linear SVMs can work (at least they can be trained) even if the number of training samples per SVM is much smaller than the feature dimension, whereas the Gaussian classifier will fail under such situation.***

***Hints: To answer this question, you can think of the case of binary classification where you have two samples (one for each class) on a 2-D space. Then, extend this to 3 linearly independent samples (one from the positive class and two from the negative class) on a 3-D space. Can you always find a plane that perfectly separates the two classes of data when the number of linearly independent training samples is less than or equal to the feature dimension?***

1. Change the SVMs to polynomial SVM with degree 2 by editing the following lines and run the script svm.ipynb.

nSamples = 1

train\_data, train\_labels, test\_data, test\_labels =

load\_SampleMnist(trainpath,testpath,nSamples)

kerType = 'poly';

kerPara = 2;

svm = SVC(kernel= kerType ,degree = kerPara)

svm.fit(train\_data, train\_labels)

***Report the accuracy for both ‘clean’ and ‘noisy’ digits. In theory, polynomial SVMs are more powerful than linear SVM because they can produce non-linear decision boundaries. However, you properly observe that the performance of poly-SVMs is poorer than that of linear SVMs. Explain why poly-SVMs are not suitable for such extreme case.***

1. Edit svm.ipynb so that you use 785 training samples per class. Re-run the script using linear SVMs and poly-SVM.

***Report your observation for linear SVMs and report the accuracy for poly-SVMs. Explain your observation for linear SVMs. How does the performance of poly-SVMs compared with that of the Gaussian classifier that uses the same number of training samples per class in Lab1? Briefly explain your observations.***

***How many support vectors for each of the SVMs? Hints: You may obtain this information by typing*** ***svm.n\_support\_, This Attribute refers to the Number of Support Vectors for each class.***

1. The support vectors in SVMs are either very close to the decision boundaries or on the wrong side of the decision boundaries. It is possible to understand how the SVMs solve a binary classification problem by inspecting the support vectors. The scoring function of an SVM is



where  is the class label of the training vector , *b* is a bias term (making the decision plane not passing through the origin), *S* contains the indexes of support vectors, ’s are Lagrange multiplies, and  is a kernel function. Let’s focus on the SVM that separates Digit ‘0’ from Digits ‘1’ to ‘9’. We can display the most influential support vectors from Digit ‘0’ and from Digits ‘1’ to ‘9’ as follows:

import matplotlib.pyplot as plt

d = 0

labels = np.copy(train\_labels) # Deep copy

labels[labels != d] = d-1 # Make it not Digit d

kerType = 'poly';

kerPara = 2;

svm = SVC(kernel=kerType ,degree=kerPara, C=10)

svm.fit(train\_data, labels)

alphaY = svm.dual\_coef\_[0,:]

plt.plot(alphaY)

plt.xlabel('Sample index')

plt.ylabel('alphaY')

plt.show()

fig, ax = plt.subplots(nrows=1, ncols=2, sharex=True, sharey=True, )

pos = np.argmax(alphaY)

neg = np.argmin(alphaY)

pos\_img = svm.support\_vectors\_[pos].reshape(28,28)

neg\_img = svm.support\_vectors\_[neg].reshape(28,28)

ax[0].imshow(pos\_img, interpolation='nearest')

ax[1].imshow(neg\_img, interpolation='nearest')

print(f'Positive Class: Digit {train\_labels[svm.support\_[pos]]}')

print(f'Negative class: Digit {train\_labels[svm.support\_[neg]]}')

***What are the vectors who corresponding***  ***are zero? Do they contribute to the SVM scoring function? Explain your answer. Does the most influential support vector from Digit ‘0’ look like a ‘0’? Does the most influential support vector from Digits ‘1’ to ‘9’ look like a ‘0’? Explain your observation.***

1. Repeat Step 7 and Step 8 but using RBF-SVM instead. Note that for this set of data, you need a largeσ for the RBF kernel parameter. In the class sklearn.svm.SVC, “gamma” is a parameter the RBF kernel. It implicitly determines the distribution of in the feature space. The larger the gamma, the fewer support vectors. The smaller the gamma, the more support vectors. The number of support vectors affects the speed of training and prediction.

RBF function formula:



Note the relationship between sigma and gamma. Gamma determines the width of the Gaussians, thereby affecting the generalization performance. If gamma is too large, will be very small, and the Gaussian distribution with a small will become tall and thin, and the classification performance on unknown samples will be poor.

kerType = 'rbf'

kerPara = 1/500

svm = SVC(kernel=kerType, gamma=kerPara, C=10)

svm.fit(train\_data, train\_labels)

print(svm.get\_params())

***Report the results and your observations.***

***E.2 Principal Component Analysis***

1. Read PCA\_projection.ipynb to understand how to perform PCA. Run PCA\_projection.ipynb to find the eigenvectors for different numbers of principal components. Determine the eigenvalues found by PCA\_projection.ipynb.

***Report the results and your observations. Plot the eigenvalues. Based on your plot, explain why we only need the first few eigenvectors to represent the digits.***

1. Read eigenface.ipynb to understand how to perform PCA on face images. Pay special attention to the method that avoid computing a large covariance matrix (read the Wiki page as stated in the .m file if necessary). Run eigenface.ipynb to find the eigenvectors for different numbers of principal components. Determine the eigenvalues found by eigenface.ipynb.

***Report the results and your observations. Plot the eigenvalues. Based on your plot, explain why we only need the first few eigenvectors to represent the digits.***

1. When projecting facial images onto low-dimensional subspace, it is necessary to remove the global mean (mean face) of facial images, and during reconstruction, it necessary to add the global mean as an offset. Specifically, the projection and reconstruction processes are as follows:





where **μ** is the global mean of facial images. Investigate the effect of peforming this mean removal by setting **μ=0.** Note that you still need subtract the mean when computing the covariance matrix for eigen-decomposition.

***Display some reconstructed face when setting  in the above equations (use the maximum number of eigenfaces). Also display the reconstructed faces using different numbers of eigenfaces. Report the results and your observations. Why is it necessary to remove the mean when performing projection and to add back the mean during reconstruction?***

1. Read LDA\_projection.ipynb to understand how to use LDA to project data onto low-dimensional subspace. Similar to Step 12, you may investigate the effect of not performing mean subtraction on the projected images by setting the global mean to zero.

***Report the results and your observations. By setting the number of principal components (PC) to 9, compare the images reconstructed by PCA and LDA. Which one looks better? Can the LDA-reconstructed image be improved by increasing the number of PC?***

1. Step 13 shows that LDA may not be suitable for dimension reduction. However, it is very good for classification. Read get\_lda\_accuracy.m to understand how to use LDA as a multi-class classifier.

***Report the results and your observations. How does the performance of LDA + Gaussian Classifier compare with that of the Gaussian Classifier?***

1. (***Optional, for those whole would like to explore more***) Now, you know how to perform PCA and LDA and how to build SVM classifiers. Construct an SVM classifier that classifies the PCA-projected data and LDA-projected data. Compare the performance with those you obtained in Step 7.

-- END --

1. You may also install Python on your computer. For example, “Anaconda 3” contains a full set of Python packages and the core Python. It also has integrated development environments (IDE) such as Spyder and VSCode. If you use Mac or Linux, Python has been pre-installed. But you still need to install an IDE. [↑](#footnote-ref-1)