EEP 596 Homework 1 (100%) due: 10/12 11:59 pm

Submission Instructions

You will need to submit the following materials:

1. **Questions and Answers** in PDF format. Remember to put your name in your report.
2. A folder containing all ipynb files with your results for the the problems. Please remember to keep all results and logs in the submitted ipynb files to get full credit.

All submitted files should be put into one single zip file named as HW#_xxx.zip, e.g. HW1_George_Clooney.zip**,** including all ipynb files with answers (both results and discussions), and the **Questions and Answers** report.

Problem 1: Practices of Python Numpy (25%)

You will go through the Python, Numpy, Scikit Learn Colab tutorial introduced by TA in class.

Problem 2: Polynomial Regression (25%)

In this problem, we will implement second order polynomial regression of (a) one variable (10%) and (b) two variable (15%) from scratch.
Then, we are going to fit the randomly generated data to quadratic functions.
Please reference the course slide when doing this problem.

Problem 3: Classification on Digital Images Using Traditional Machine Learning (25%)

We will go through a definition called Principal Component Analysis (PCA) for dimensionality reduction for high dimensional data. We do dimensionality reduction to convert the high d-dimensional dataset into k-dimensional data where $k < d$. Data variance on one axis may be very large but relatively smaller on another axis. Higher variance usually means greater information in this direction. Therefore, we can skip the dimensions having less variance because having less information.

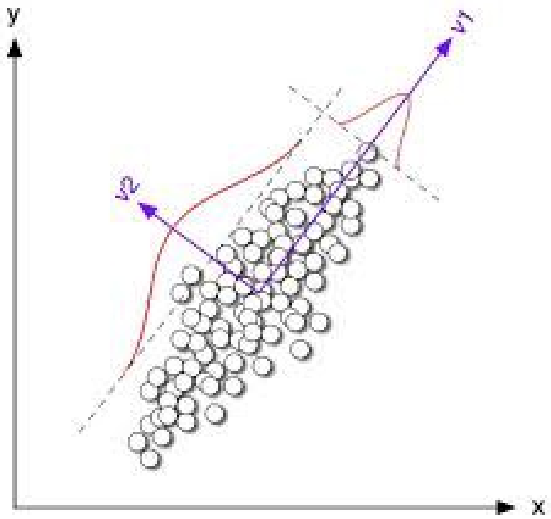


Fig. 1: Variances of 2D data.

Fig. 1 is an example of a set of 2D data. The direction of v1 is maximum while v2 is minimum, so that v1 has more information about the dataset. Thus, the 2D data with (x,y) variables can be converted to 1D variables in the direction of v1.

Projections (Orthogonal)

Input: $x_1, x_2, \ldots, x_n \in R^d$, target dim k.

Output: a k-dim subspace by orthonormal basis $q_1, q, \ldots, q_k \in R^d$.

Orthogonal projection:

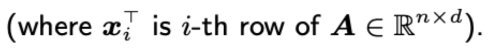
$$\underbrace{\left(\sum_{i=1}^{k} \boldsymbol{q}_i \boldsymbol{q}_i^\top\right)}_{\boldsymbol{\Pi}} \boldsymbol{x} \;=\; \sum_{i=1}^{k} \langle \boldsymbol{q}_i, \boldsymbol{x} \rangle \boldsymbol{q}_i \;\in\; \mathbb{R}^d .$$

It can be also represented in terms of coefficients w.r.t. the orthonormal basis $q_1, q, \ldots, q_k \in R^d$:

$$\boldsymbol{\phi}(\boldsymbol{x}) \;:=\; \begin{bmatrix} \langle \boldsymbol{q}_1, \boldsymbol{x} \rangle \\ \langle \boldsymbol{q}_2, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{q}_k, \boldsymbol{x} \rangle \end{bmatrix} \;\in\; \mathbb{R}^k .$$

Minimize residual squared error

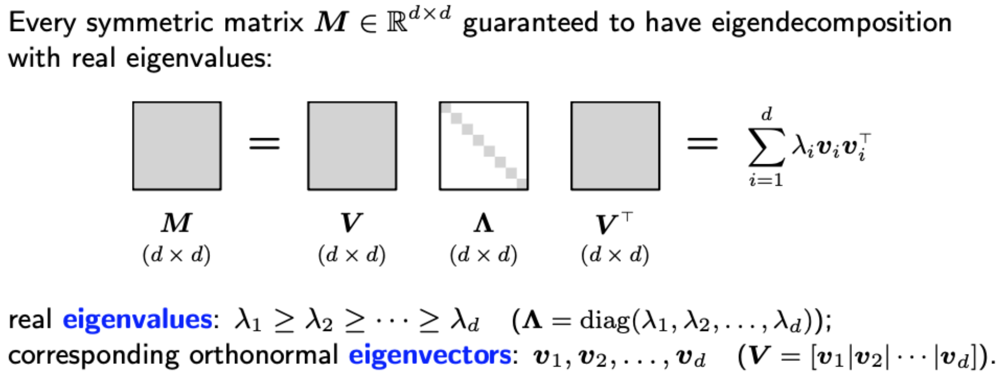
$$\underset{\substack{\boldsymbol{Q} \in \mathbb{R}^{d \times k}: \\ \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}}}{\arg\min} \; \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \boldsymbol{Q}\boldsymbol{Q}^\top \boldsymbol{x}_i \right\|_2^2 \quad \equiv \quad \underset{\substack{\boldsymbol{Q} \in \mathbb{R}^{d \times k}: \\ \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}}}{\arg\max} \; \sum_{i=1}^{k} \boldsymbol{q}_i^\top \left(\frac{1}{n} \boldsymbol{A}^\top \boldsymbol{A} \right) \boldsymbol{q}_i .$$



(where $\boldsymbol{x}_i^\top$ is i-th row of $\boldsymbol{A} \in \mathbb{R}^{n \times d}$).

Solution: k eigenvectors of $A^\top A$ corresponding to k largest eigenvalues.

Eigen Decompositions



Every symmetric matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ guaranteed to have eigendecomposition with real eigenvalues:

$$= \quad = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

real **eigenvalues**: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \quad (\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d))$;
corresponding orthonormal **eigenvectors**: $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d \quad (\boldsymbol{V} = [\boldsymbol{v}_1 | \boldsymbol{v}_2 | \cdots | \boldsymbol{v}_d])$.

PCA step-by-step

Step 1: Standardize the dataset.
Step 2: Calculate the covariance matrix for the features in the dataset.
Step 3: Calculate the eigenvalues and eigenvectors for the covariance matrix.
Step 4: Sort eigenvalues and their corresponding eigenvectors.
Step 5: Pick k eigenvalues and form a matrix of eigenvectors.
Step 6: Transform the original matrix.

Find data representation for MNIST dataset

Handwritten classification is a basic task for early stage image processing. In this problem, we will work with MNIST dataset, which contains 60,000 training and 10,000 testing images of handwritten numbers 0-9.

Fig. 1: Some sample images in MNIST dataset.

(a) Prepare MNIST dataset (5%)

Download MNIST dataset mnist.mat from the provided google drive link and follow the instruction in HW1.ipynb to load and visualize the dataset.

In this part, you will practice how to load files from your Google Drive to the Google Colab Notebook. To achieve this, you will need to:

- Upload file(s) to your own Google Drive.
- Mount your Google Drive to the corresponding Colab Notebook.
- Find the path of your uploaded file(s) and access them.
- Split the validation set from the training set (train: 50000, valid: 10000, test: 10000).

The detailed instructions for this part is shown in the HW1.ipynb file.

(b) PCA on MNIST (10%)

Implement PCA on MNIST dataset to reduce the dimension of the digit images.

1) Each data in the MNIST dataset is a 784-dimensional vector (flatten from 28x28). Please use PCA to reduce the dimension to a smaller value *k*.
2) How to **select** a good reduced dimension *k*? Keep 80% information after the reduction.
3) Transform all the data into the reduced dimension (train: 50000 x *k*, validation: 10000 x *k*, test: 10000 x *k*).

Note that you will use this reduced data for digit classification in the following problem.

(c) Support Vector Machine (SVM) Classifier (10%)

Follow the steps on the notebook HW1.ipynb to build a SVM Classifier for MNIST dataset. Here you should use the representative data (after PCA) for training and inference. For more details, please refer to https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

1) Implement a SVM classifier using the scikit-learn package: sklearn.svm.SVC with L2 regularization parameters C = 1.0, kernel type 'linear'.

2) Evaluate the classification accuracies on the validation set.

3) Try using a different kernel type, change the kernal from 'linear' to 'rbf' (radial basis function) and evaluate the classification accuracy on the validation set. Which one ('linear' or 'rbf') can give you higher accuracy?

4) Fix the kernel type to be 'rbf' and try different sets of regularization parameters C $\in$ {0. 1, 0. 5, 1. 0, 5. 0, 10. 0}, and report all the classification accuracies on the validation set. What's the meaning of changing the C here? Which C in your case can give your the best accurarcy?

5) Run your classifier on the testing set with the model which achieves the best performance on the validation dataset, and visualize some of the images with their predicted labels.

Problem 4: Questions and Answers (25%)

1. (5%) Given a trained classifier for 4 object classes (C_1, C_2, C_3 C_4), an input data belongs to C2 generates (0.15, 0.7, 0.1, 0.05) output likelihood, what are the corresponding loss values (L1 loss, L2 loss and cross-entropy loss.) associated with this data?

2. (5%) Given the **confusion matrix** of this 4-class classifier

	Predicted Classes				
Actual Classes		C_1	C_2	C_3	C_4
	C_1	68	12	9	11
	C_2	14	74	5	7
	C_3	12	3	82	3
	C_4	6	10	12	72

Please compute the overall average accuracy. Followed by per class precision and recall, F1 score of C4, and the micro-average precision of all 4 classes.

3. (5%) Multiple Linear Regression: Given the following dataset with one response variable y and two predictor variables x_1 and x_2:

y	x_1	x_2
140	60	22
155	62	25
159	67	24
179	70	20
192	71	15
200	72	14
212	75	14
215	78	11

Please find the linear regression model $y=b_0+b_1x_1+b_2x_2$, i.e., determine the linear regression coefficients, b_0, b_1 and b_2 based on least squares solution.

4. (5%) Explain what is Support Vector Machine (SVM) (3%), and when and how do we use nonlinear SVM (Hint: Kernel Trick) (2%)?

5. (5%) Explain what are discriminative and generative classifiers?