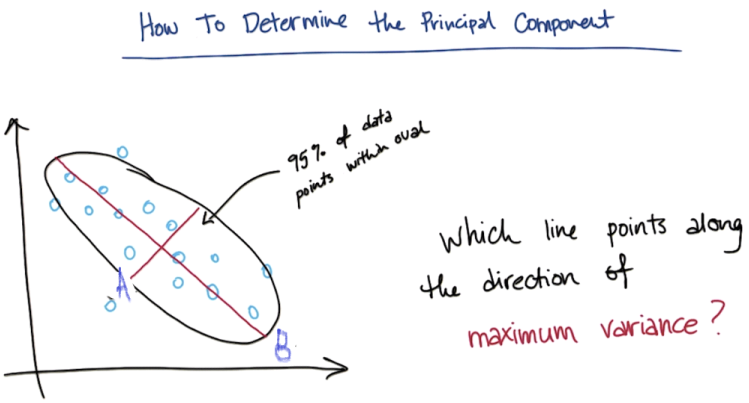
National University of Singapore
School of Computing
Tutorial 5:
PCA and GMM

Introduction to the Principal Component Analysis

- Systematized way to transform input features into Principal Component
- These Principal Components acts as new features
- Principal Components are directions in data that maximizes the variance and minimize the information loss, when you project or compress down to them
- More variance of data along Principal Component, higher the principal component is raked
- Maximum number of PC = Number of Input Features



When to use PCA

- To find latent features driving the patterns in Data
- Dimensional Reduction
  - To Visualize High Dimensional Data
  - To Reduce Noise
  - To make algorithms like Classification and Regression work better with reduced dimensionality

Application: Facial recognition using PCA + SVM.

1. Download the 'fetch_lfw_people' dataset from sklearn datasets using 'fetch_lfw_people(min_faces_per_person=70, resize=0.4)'. Introspect the parameters of dataset. Print the target_names parameters. Visualize a few images at random.

2. Create your X variable (the features) and the y variable (the labels).

3. Create a train-test split in your data using the SKLearn Train-Test split library.

4. Compute a PCA on the face dataset with n_component=150. This will help in dimensionality reduction. Create new features after PCA for the train and test data.

5. Now using the new features fit the SVM classifier predict the targets. Try using GridSearchCV to tune your C and gamma parameters. Print the best_estimator_ of the GridSearchSV.

6. Create predictions on the test set using the best estimated fitted classifier and use the SKLearn Classification_report library to generate a classification report. Discuss your results.

7. Visualize your prediction by plotting few images and its corresponding actual target and predicted target.

Introduction to the GMM

- A GMM attempts to model the data as a collection of Gaussian blobs.
- You can use it as unsupervised clustering algorithm which attempts to find distinct groups of data without reference to any labels.

Application: Segmentation of Image using GMM Clustering, i.e. giving each pixel a label

8. Generally, Image consists of 3 frames (Blue, Green, Red), with each pixel ranging from 0-255. Load a sample image from sklearn dataset with '`load_sample_image('china.jpg')`'. Visualize the image using '`plt.imshow`'. Print and Save your original image shape. Let the shape of image be `(h, w, 3)`.

9. Assign each pixel inside `(h, w, 3)` a label from 0-5 using sklearn 'GaussianMixture' clustering. To do that first flatten the image numpy array using '`.reshape(-1, 3)`'. Now fit the GaussianMixture with `n_clusters=5`, to assign labels to flattened array.

10. Reshape your predicted labels to (h, w) shape, i.e. size of original image. Now visualize your segmented image using '`plt.imshow(new_image, cmap='gray')`'.