Machine Learning Homework 4

TA hours

ML TAs ntu.mlta@gmail.com

May 11, 2017

NTUEE

Outline

Eigenfaces with PCA

Visualization of Word Vectors

Estimation of Intrinsic Dimension

Eigenfaces with PCA

Q1: Dimensionality Reduction with PCA

1. Perform PCA using the first 10 faces of the first 10 subjects to obtain the eigenfaces. Plot the average face. Also plot the top 9 eigenfaces in a figure (3-by-3, left to right & top to bottom). Show the figures in your report. (1%)

- Load data (flatten): $X \in \mathbb{R}^{N \times d}$
- Calculate mean: `X_mean = X.mean(axis=0, keepdims=True)`
- Subtract mean: `X_ctr = X - X_mean`
- SVD: `u, s, v = np.linalg.svd(X_ctr)`
  $\rightarrow$ rows of `v` are the eigenvectors (eigenfaces)
- Plot the first 10 eigenfaces by reshaping back to 64×64
- You can use `cmap=pyplot.get_cmap('gray')` from `matplotlib.pyplot`

Q1: Dimensionality Reduction with PCA

2. Project the 100 faces onto the top 5 eigenfaces, and then reconstruct the original images. Plot the 100 original faces (10-by-10) and the recovered faces (also 10-by-10). Show the two figures side-by-side in your report. (1%)

- Project each face $\mathbf{x}$ onto i^{th} eigenfaces by calculating the dot product between $\mathbf{x} - \mu$ and $\mathbf{v_i}$ to obtain x_i:

$$x_i^{\text{reduced}} = (\mathbf{x} - \mu)^{\mathsf{T}} \mathbf{v_i}$$

- You can recover each image $\hat{x}$ by a weighted sum of the 5 eigenfaces with the elements of x^{reduced} as weights, plus the mean:

$$\hat{\mathbf{x}} = \mu + \sum_{i=1}^{5} x_i^{\text{reduced}} \mathbf{v_i}$$

Q1: Dimensionality Reduction with PCA

3. In problem 2, we can choose top k eigenfaces and check the reconstruction error (RMSE). Find the smallest k such that the error is less than 1%. (1%)

   - For from $k = 1$ to 100, follow steps in problem 2 to obtain the recovered images. Calculate the RMSE:

$$\sqrt{\frac{1}{N}\frac{1}{2500}\sum_{N}\sum_{i=1}^{2500}|x_i - \hat{x}_i|^2}$$

Visualization of Word Vectors

Word2Vec Training

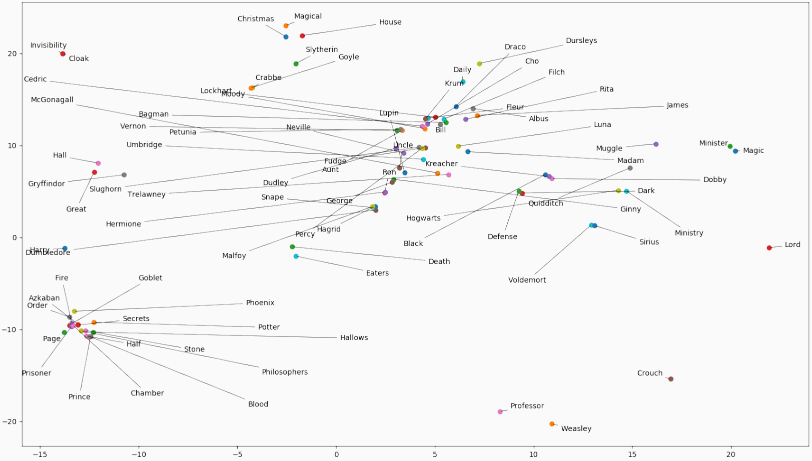
- Train model: `word2vec.word2vec` (Set your parameters!)
- Load your trained model: `word2vec.load`
- Obtain vocab: `model.vocab` (in the order of word frequency)

Visualization

Visualization Tips

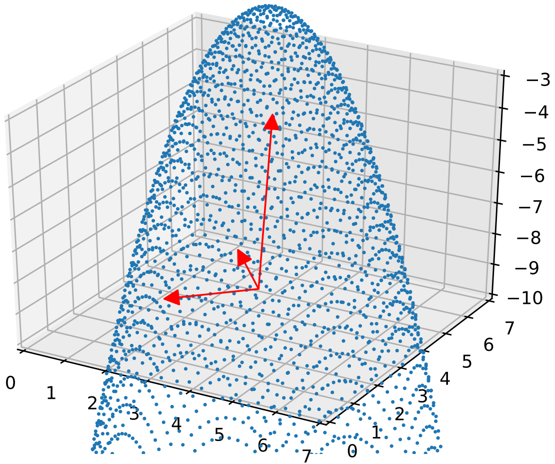
1. Use `PCA` or `TSNE`(preferred) from the `scikit-learn` library to project the vectors of the k-most-frequent words (a value of 500∼1000 is recommended).
2. Use the `NLTK` library to obtain the POS tags of the words. You should install `NLTK` and run `nltk.download()`. Choose “Models” and download `averaged_perceptron_tagger`, `maxent_treebank_pos_tagger`, and `punkt`.
3. Plot the words that have tags `JJ`, `NNP`, `NN` and `NNS`, and ignore those that contain punctuations (“ , . : ; ’ ! ? ”).
4. Use “adjust_text” (link) to prevent plot labels from overlapping.

Visualization Example

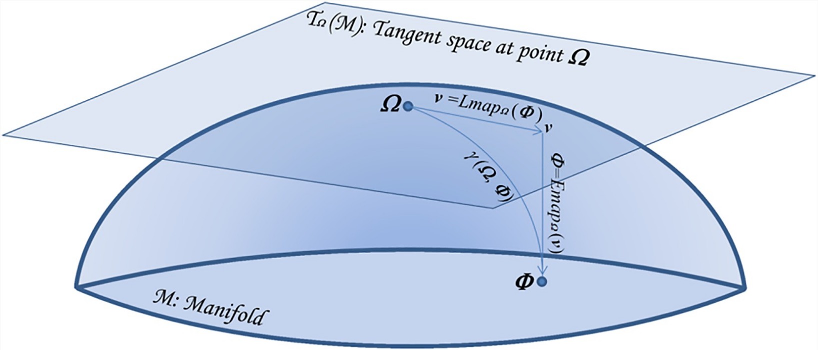


Estimation of Intrinsic Dimension

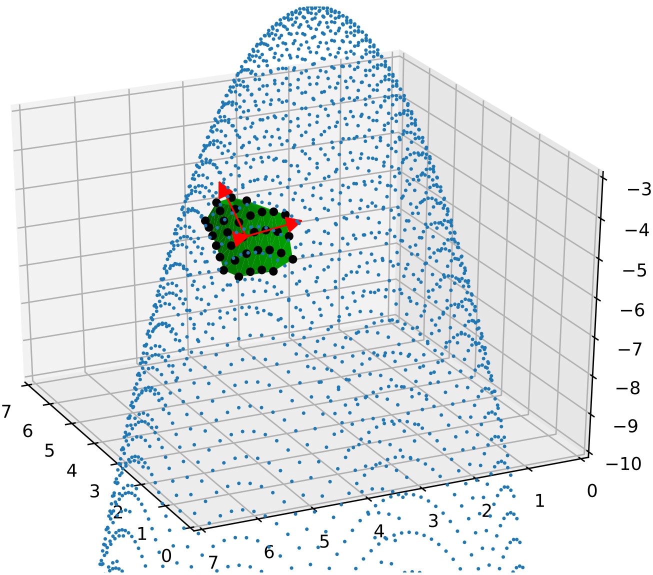
Idea - PCA



Idea - Tangent Space



Nearest Neighbors



About Complexity

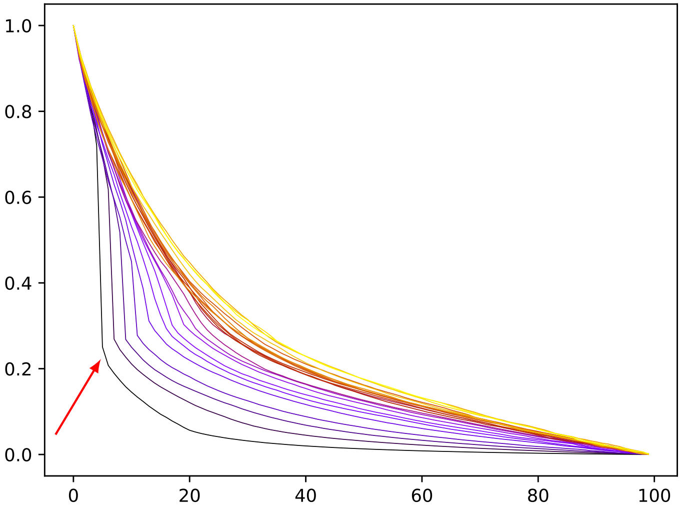
Since the original dimension is large ($= 100$), KD-tree is not a good choice. You can choose Ball-tree instead.

For more information, check the sklearn docs.

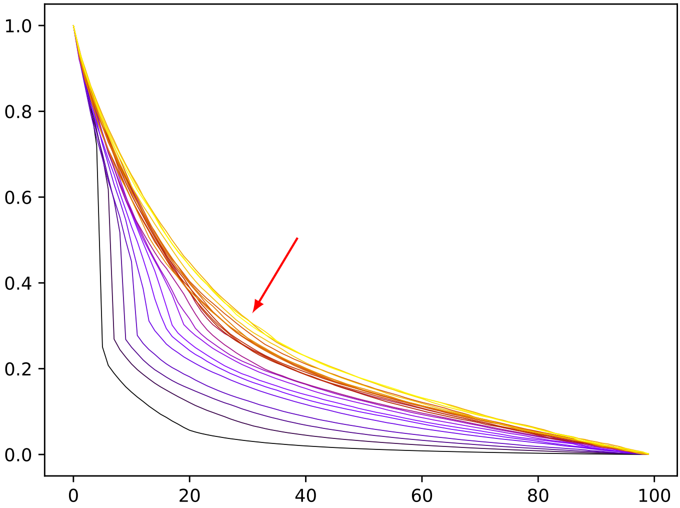
Measure the Dimension

1. Sample N points from a dataset randomly.
2. For each point, find k nearest neighbors of it.
3. Compute the eigenvalues of this subset of data.
4. We can sort the eigenvalues to see if there exists any obvious gap.
5. Normalize eigenvalues.
6. For all i, average i-th eigenvalue over N samples.

Result



The Problem



Simplest Method

1. Generate Data.
2. For each dataset, compute average eigenvalues.
3. Use Linear SVR to estimate the dimension.
4. Tune the parameter C.

Tricks

- Since the cost is relative, we can just predict $\ln d$ instead of d.
- Since all dimensions are integer, rounding the values may get a better result.

Sample Code

1. PCA: Too easy, no sample code.
2. very simple example
3. very simple example

Note: you must modify them for the code submission!

Questions?