Prac 5 - Dimensionality Reduction using Principal Component Analysis and t-SNE

COMP4702/COMP7703 - Machine Learning

Aims:

- To complement lecture material in understanding the operation of PCA.
- To complement lecture material in understanding the operation of t-SNE.
- To gain experience with simulating and implementing these techniques in software.
- To produce some assessable work for this subject.

Part I: Principal Component Analysis:

PCA can be implemented very simply in Matlab or Python. Given a dataset (as a matrix X), the covariance matrix can be found using the Matlab covariance function (cov(X)). Then, the eigenvectors and eigenvalues of this covariance matrix are the principal component (vectors) and principal values respectively. The eigenvalues reflect the amount of variance accounted for by each principal component and are ordered. To perform dimensionality reduction (e.g. down to 2 dimensions), we need to multiply X by the two eigenvectors with the largest corresponding eigenvalues.

- Read section 6.3 of the Alpaydin text.
- Write a Matlab or Python function implementing PCA.

(**Q1**) List the code you have written that implements PCA.

(**Q2**) For the MNIST dataset, run your PCA function on the data.

(a) Produce a plot of the data in the space spanned by the first two principal components. Colour each point by its class.

(b) What percentage of the data variance is accounted for by the first two principal components?

(c) From the results, produce a Scree graph similar to that shown in Fig 6.2 of the Alpaydin text.

(**Q3**) Repeat the procedure in (**Q2**) using the Swissroll and Diabetes datasets. For MNIST, compare a plot visualising only 2 digits with a plot visualising 10 digits.

Part II: t-SNE:

Suppose we are given the task of placing objects existing in high dimensional space $\mathbb{R}^d$ into a low dimensional space $\mathbb{R}^l$ so that the neighbours of similar objects in high dimensional space are preserved in the new low dimensional space. Stochastic Neighbour Embedding (SNE) is an algorithm that accomplishes this. For each object x_i in $\mathbb{R}^d$, the conditional probability that x_i would pick x_j as its neighbour is calculated according to a Gaussian distribution. Similarly for each object y_i in $\mathbb{R}^l$, the conditional probability that y_i would pick y_j as its neighbour is calculated according to a Gaussian distribution. The algorithm proceeds to minimise the sum of KL-divergences over the probability distributions in $\mathbb{R}^l$ and $\mathbb{R}^d$. As we know from previous practicals, the KL-divergence is a measure for how diffferent two distributions are, so minimising it is intuitively motivated. Gradient Descent is the algorithm that is used to minimise the KL-divergence.

- Read the original paper on t-SNE available on blackboard at **Learning Resources > Articles (for COMP7704 Reading Assignments) > Week 7: van der Maaten and G. Hinton, Visualizing data using t-SNE**.
- Download the Matlab or Python implementation of t-SNE available at https://lvdmaaten.github.io/tsne/. Also download the MNIST data in .mat or other format.
- Run the t-SNE algorithm on 6000 datapoints from the MNIST dataset:

```
1 clear all; close all; clc;
2 load('mnist_train.mat');
3 idx = unidrnd(60000, 6000, 1);
4 x = train_X(idx, :);
5 labels = train_labels(idx);
6 tsne(x, labels, 2, 30, 30)
```

(**Q1**) In one or two sentences, explain how t-SNE differs from SNE.

(**Q2**) In one or two sentences, explain lines 3, 4 and 5 in the code snippet above.

(**Q3**) Provide a screenshot of the 2-dimensional visualisation after 300 iterations. Plot the error at each iteration up to 300 iterations in steps of 10.

(**Q4**) In 3-4 sentences, explain lines 51-53 and lines 87-89 in tsne_p.m in relation to any features you observe in your plot. Why has the code been written in this way? (Hint - read the paper).

(**Q5**) Comment out lines 51-53 and 87-89. Run t-SNE for 300 iterations for perplexity values ranging from 10 to 300 in steps of 10. Produce a 3D plot with perplexity and iterations on the horizontal axis and cost on the vertical axis.

(**Q6**) After consulting your plot, comment on the following statement: *A lower cost value always produces a better visualisation.* Choose a suitable perplexity value and provide the visualisation in 2D space after 300 iterations for your chosen perplexity. Compare this with your result in (**Q3**).

(**Q7**) Run t-SNE (without PCA as a preprocessing step and with lines $51-53$ and $87-89$ uncommented) on the Swissroll and Diabetes datasets. Comment on the visualisations it produces. Provide the parameters you used.

Additional Resources:

- How to use t-SNE Effectively - http://distill.pub/2016/misread-tsne/
  One of the first articles on the new research platform called Distill.
- Gradient-Based Optimization - Chapter 4.3 of Deep Learning available online
  http://www.deeplearningbook.org/contents/numerical.html
  We will be revisiting gradient descent later when training neural networks.

Useful Matlab Commands:

The following commands may be useful when writing code to answer the questions in this prac.
eig(), eigs(), mean(), cov(), repmat(), diag(), surf()
Similar functions exist in Python's numpy.