Final Exam: Due March 15th

Matt Thomson

3/8/2018

Introduction

The final exam covers material from the last section of the course. The exam will walk you through problems that apply SNE based embeddings to data.
The exam has two possible tracks a computationally focused track which is oriented on developing native procedures for applying (t)SNE to simulated data. This track will require students to think through details of the method and implement several detailed aspects of the method. You will be graded, in this case, on your effort to address each key conceptual step in the method.

The bio track applies these methods to the human immune data set that we have worked on in the last two problem sets. The goal of this track is to understand how to apply some of the methods we have used to break a cell population into a primitive set of cell groups and analyze the biological content of these groups. Im hoping that students will read the primary paper, think scientifically about the data, and methods. An important aspect in this track is having students engage with the underlying data and formulate biological questions about the immune system in health and disease.

Computational Track

SNE based methods

My goal is to walk you through an implementation of SNE and tSNE. These closely related methods construct a low dimensional embedding of a high dimensionality data set through a small set of fundamental ideas.

Please also read:
https://www.cs.toronto.edu/ fritz/absps/sne.pdf
http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

(Introduction) Write a function to construct the conditional probability matrix for the input data points.

For two points, x_i and x_j in a data set, we construct a conditional probability, $p(x_i, x_j)$ (Hinton calls this p_{ij}), which we think of as the probability of observing the point x_j, given a Gaussian density at x_i. To construct, $p(x_i, x_j)$ we center a Gaussian distribution around x_i, and then we calculate:

$$p(x_i, x_j) = \frac{\exp(\frac{-||x_i - x_j||_2}{2\sigma_i^2})}{Z}$$

$$Z = \sum_{k \neq i} \exp(\frac{-||x_i - x_k||_2}{2\sigma_k^2}).$$

Here, Z_i is a normalization factor. We set $p(x_i, x_i) = 0$. For each point in the data set, we now have a probability distribution over edges connecting to all other points. Edges are weighted by a Gaussian kernel that decays with distance, $\frac{-||x_i - x_j||_2}{2\sigma^2}$.

The choice of σ_i is an important implementation detail. The authors make this choice by developing a concept they call the local perplexity of the data.

The key idea is that we can think about the distribution of $p(x_i, x_j)$ for a point x_j, and think about its entropy or how spread out this distribution is. If σ_i is large, then more edges contribute to the probability density. Smaller σ_i, causes fewer edges to contribute to the density.

So the developers of this method think of tuning σ_i locally (for each x_i) to enforce a specified neighborship or perplexity locally in the data. They call this number k. For each x_i in the data set, you can think of setting σ_i so that:

$$H[P_i(X)] = -\sum_{k \neq i} p(x_i, x_k) \log p(x_i, x_k)$$

so that we are calculating the entropy of the edge distribution at a point. The authors tune

Remember that x_i is fixed above. In a data set with n points, the entropy can be at most $\log n$ and at least 0. For each point in the data set, the authors construct a procedure to set $H[P(x_i, x_j)] = \log(k)$ where k is the user specific perplexity and ranges between 5 and 50. Practically, you can perform a simple search over data points to set σ_i locally.

In the initial implementation of your method, skip this step (just as the authors did) and set $\sigma_i > 30$.

Now, think of embedding the points x_i into a lower dimensional space, so that each x_i will get mapped to a $y_i \in \mathcal{R}^2$.

To perform this embedding, we seek to minimize the KL-divergence between $p(x_i, x_j)$ in the native space and a distribution of the identical form constructed in the lower dimensional space.

Thus, we construct $q(y_i, y_j)$ as:

$$q(y_i, y_j) = \frac{\exp(\frac{-||y_i - y_j||_2}{2\sigma_i^2})}{Z}$$
$$Z = \sum_{k \neq i} \exp(\frac{-||y_i - y_k||_2}{2\sigma_k^2}).$$

Now, we can write the KL divergence between p and q simply as:

$$D(P_i||Q_i) = \sum_{k \neq i} p(x_i, x_j) \ \log(\frac{p(x_i, x_j)}{q(y_i, y_j)})$$

It is standard practice, to fix $p(x_i, x_j)$ and attempt to minimize $D(p_i||q_i)$ by adjusting q.

The authors call the total KL divergence $C = \sum_k D(P_k||Q_k)$ and construct a gradient $\frac{dC}{dy_i}$ which they show to be:

$$\frac{dC}{dy_i} = 2 * \sum_j (y_i - y_j)(p_{i,j} - q_{i,j} + p_{j,i} - q_{j,i})$$

Thus, the KL divergence between data points can be minimized by performing gradient descent on C numerically.

The authors comment that this gradient descent for SNE is challenging and recommend proceeding by initializing y_i points to be close to the origin and then performing an annealed descent with noise on C.

In my implementation I add some random Gaussian jitter to the gradient at each point.

(1) Implement SNE and provide your code as part of your exam.

(2) Generate a Gaussian mixture model with at least 4 Gaussian. You can select the mixture to have a structure of your choosing or consult with me for advice.

Please provide as a plot of your Gaussian mixture in PCA space.

(3) Run your SNE embedding on the Gaussian mixture data. please provide data from 4 different runs of the embedding for 4 different values of perplexity.

(4) Comment on the geometry of the SNE embeddings and specifically how distances between Gaussian centroids translates into the SNE plot.

(5) SNE was updated in two key ways. You will notice in SNE that $p_{ij} \neq p_{ji}$, so the matrix can be made symmetric. Second the Gaussian distribution in the low dimensional space was replaced by the Cauchy (or t distribution). The key aspect of this distribution is that it replaces the exponential tail decay with a power law decay.

Please take your version of SNE and modify it for tSNE and generate the same plots for your Gaussian mixture. What do you observe?

http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

Biology Track

(1) Before using a package for tSNE, I would like you to go through the first step of implementing SNE. Please construct the pair-wise conditional probability matrix for the healthy1 data set. Refer to the computational track description for details.

Construct $p(x_i, x_j)$ and plot the histogram of edge weights for a single node for two different values of σ_i. (**Two plots to show**)

(2) Perform PCA on the healthy1 data set as you did in the last problem set. However, this time, normalize each gene by mean centering and dividing by the standard deviation of the gene. **You do not need to produce any plots from this step.**

Use the representation of the healthy1 cells in the first 10 principle components as input into tSNE. You can run tSNE using sci-kit learn or matlab.

(3) **Please run tSNE a total of 3 times for each sample**. Chose 3 different values of the perplexity and for each value run the method 3 times. In each case plot the embedding and store the KL divergence or loss (**9 plots total**). Try adjusting the Perplexity between 1 and 100.
Generate these plots and comment on what you observe.

(4) Now, return to the original input data again. We are going to compare the tSNE decomposition to non-negative matrix factorization and use NNMF to identify gene clusters that occur in the different tSNE plots.

When doing, NNMF entries must be positive so mean centering is not appropriate. Instead, scale the data by dividing each gene row by the standard deviation of that row.

Now, perform NNMF setting the inner dimension k to detect the number of clusters you observe in the tSNE plot.

Plot a scatter plot of your tSNE embedding colored by the coefficient weighting of each cell in each row of the the H matrix defined by NNMF. If you choose 6 clusters, then you should generate 6 different scatter plots of your tSNE data **k plots to show**.

(5) You should find that your NNMF parts closely associate with the tSNE clusters. Pick three clusters of interest and their associated W vectorts (NNMF 'parts').

Using the corresponding vectors in the W matrix, examine the genes within the W vectors of interest. For a given vector w, find genes where the weighting, $w_i > \mu_w + 2\ \sigma_w$.

Select these genes and, then, go back to the original paper as well as the Immgen site and broad internet resources. Can you identify genes within the w vector that indicate which broad class of immune cells are represented in your clusters of interest?

http://rstats.immgen.org/MyGeneSet/

Also gene cards:
http://www.genecards.org/cgi-bin/carddisp.pl?gene=CD37

http://www.genecards.org/cgi-bin/carddisp.pl?gene=CD3D

The 10x analysis largely focuses on a small number of marker genes. Make sure to read the paragraph that begins:

'Cluster-specific genes were identified following the steps outlined'

They perform a broader analysis by clustering the genes into groups (as you do with NNMF). However, their biological significance is determined by using known immune markers.
Comment on identity of 3 clusters in your tSNE plots

(5) Perform a single analysis of the AML1 sample using the work-flow above. Perform PCA and then tSNE. Keep the perplexity set to the same number you used in healthy 1. Again perform NNMF on the sample to identify large clusters in the data.

Simply return 1 'tSNE' plot of your data, and answer the following question: What do you think the largest population of cells is in the healthy 1 sample vs the aml 1 sample? What evidence do you have of this?

Your answer to this question can simply explore some possibilities while reading the paper critically and thinking about your analysis.

Note: You are not required to do the following but if interested. A very common analysis is to perform k-means or spectral clustering on your tSNE plot to identify cell clusters. Following clustering, you can perform ML based differential expression (like in problem set 1) or perform standard expression tests to detect genes that are up or down-regulated in clusters. If you want to perform this analysis instead of NNMF, Im happy with that as well.

Generally we find matrix factorization to be a more direct way to address these problems because matrix factorization returns both cells and genes in one step and can be seen as an implicit clustering method that has direct interpretation in terms of loss function and optimization paradigm. Further, differential expression tests–like those done in bulk RNA-seq, are pair-wise tests. That said, differential expression based upon ratios can be easier to understand in a first approach and provide intuition for the data set.

Another approach is to separately identify gene and cell clusters in the data by performing spectral clustering (or some other clustering routine) on genes and cells separately and then linking gene clusters and cell clusters.