Homework 6: Cell as a bag of programs and single cell mRNA-seq

Bi 183
Winter 2020
Due: Feb. 27th (noon)

Introduction

In this problem set, you will apply PCA and non-negative matrix factorization to real and synthetic gene expression data sets. I want to give you the key facts that you will need to do the problem set first.

PCA review

Principal component analysis uses the eigenvectors of the data covariance matrix as a basis set in which to describe and analyze the data. In class we studied a matrix factorization formulation of PCA to determine why the eigenvectors of the covariance matrix provide a low error representation of the data.

For a gene expression matrix, D ('n' genes by 'm' cells), you can perform PCA by, first, mean centering the data by row. For each row in D calculate (here shown for row 1),

$$D'_{1i} = D_{1i} - \frac{1}{m}\sum_{i=1}^{m} D_{1i},$$

where each element in a row, D_{1i} is mapped to a new mean-centered value D'_{1i}, and $\frac{1}{m}\sum_{i=1}^{m} D_{1i}$ is the row mean. Here, I am showing you the calculation for the first row, and you would repeat for all rows (genes) in the matrix.

Now, using the mean centered data, I can calculate the gene-gene covariance matrix Σ (n by n) as:

$$\Sigma = \frac{1}{(m-1)} D' \; D'^{T}$$

where D'^{T} is the transpose of D. Then, I find the eigenvalues, $\{\lambda_i\}$, and eigenvectors, $\{\mathbf{v_i}\}$ of Σ. Each eigenvector $\mathbf{v_i}$ is associated with an eigenvalue λ_i. This eigenvalue quantifies the variance in the data explained by that direction in gene expression space. In this way, you can order the $\mathbf{v_i}$ by their importance by ordering them by their associated eigenvalue λ_i. Vectors with relatively large λ_i are the 'most important' where importance is measured by variance. It is useful and common practice to plot λ_i vs i. In lecture we showed that there is a close relationship between variance and 'error', so that the reduction in representation error provided by an eigenvector, $\mathbf{v_i}$, is $\sqrt{\lambda_i}$.

To project a data point, $\mathbf{g}$ ($\mathbf{g}$ is a genes by 1 vector), onto a specific principal component vector, we calculate $\mathbf{v_i^T g}$ (or $\mathbf{g^T v_i}$) where again $\mathbf{g}$ is a column of your data matrix and $\mathbf{v_i}$ is an eigenvector of Σ. In exploratory data analysis, we often project data onto principal components to study the structure of the resulting point set.

Note: Notice that I have performed PCA by thinking of the data points as *cells* embedded in an n dimensional Euclidean vector space. I could have, alternately, performed an identical analysis working with D^T instead of D. in this analysis, I am effectively thinking of my points as genes and the space as being the m dimensional space defined by the cells. In either case, I can obtain a covariance matrix and eigenvectors and eigenvalues. The eigenvectors of DD^T and D^TD have a very close relationship. If v is an eigenvector of DD^T, for example, you can show that D^Tv is an eigenvector of D^TD.

Non-negative matrix factorization

In class, we said that PCA can be viewed as solving the following optimization problem:

$$\text{argmin}_{V_k} \; ||D - V_k \; V_k^T \; D||_2 \tag{1}$$

where V_k is a set of vectors of rank$k$, and $||||_2$ represents a matrix norm known as the Frobenius norm. By rank, we mean that the number of linearly independent vectors is k.For a matrix X, $||X||_2 = \sqrt{\sum_{ij} X_{ij}^2}$. PCA (and the related Singular Value Decomposition) provides the best rank(k) approximation to your data by generating $\hat{D} = V_k \; V_k^T \; D$ (see the Eckart Young theorem) that minimizes $||D - \hat{D}||_2$. Notice that if $k = n$ (n is the number of genes), then $\hat{D} = D$. When we take a smaller number of eigenvectors, $\hat{D}$ approximates D. Equation (1) is important because it frames the principal component decomposition as an optimization problem.

The problem with PCA is that it finds a set of vectors, $\{\mathbf{v_i}\}$, that are orthogonal, so that $v_i^T \; v_j = 0$ if $i \neq j$. This is a problem because, in reality, gene expression programs are often not orthogonal. Programs share component genes, and so PCA, by construction, cannot extract feature vectors that contain shared components. Furthermore, PCA will find a set of basis vectors with positive and negative entries. Gene expression data is only positive, and so the PCA basis vectors can be challenging to interpret. Thus, we often hope to extract non-orthogonal vectors that are constrained to be positive to describe data. You will see this in problem 1 and 2 below.

Non-negative matrix factorization frames a new problem motivated by (1). In non-negative matrix factorization, we attempt to find a positive and non-orthogonal set of basis vectors, $\{w_i\}$, the columns of a matrix W, that allows us to generate a low-error factorization of our data.

Specifically, non-negative matrix factorization attempts to solve the following problem:

$$\begin{aligned} &\text{argmin}_{\{W\ H\}}\ ||D - W\ H||_2 \\ &W_{i,j} \geq 0, \\ &H_{i,j} \geq 0, \\ &|\{w_i\}| = k, \ \text{ we extract k vectors.} \end{aligned} \tag{2}$$

Here, W is the analogue of V_k in PCA. W is a genes by k matrix that contains k basis vectors. H is the analogue of $V_k^T\ D$, H (k by cells) represents the coefficients of these basis vectors for each cell. H is a k by cells matrix. Critically, in non-negative matrix factorization, k is a parameter selected by the user. Equation (2) attempts to find an approximation of D as $\hat{D} = W\ H$ that as before minimizes the error between data and approximation. However, we solve the problem with the constraint that W and H are matrices with positive entries, and we constrain the number of vectors in the decomposition to be k.

Note: the number k controls the dimension of W and H, this is a critical parameter in NNMF and is user defined. Think about how to pick k and how this relates to the framework of PCA.

The problem (2) is not a convex optimization problem. There, can, in fact, be many solutions with equivalent error, and solutions are a topic of current research (see Joel Tropp in CMS). For W or H individually, the problem is convex and can be solved by a variety of methods. Conceptually, an attractive approach is to use non-negative least squares. In this approach, you first fix W and solve for H using non-negative least-squares to minimize error. Then, you fix H and solve for W.

In the problem set, you will use a method first proposed by Lee and Seung called the multiplicative up-date rule. In this method, H and W are updated iteratively using the following update rule:

$$\begin{aligned} H_{ij}\ \frac{(W^T\ D)_{ij}}{(W^T W H)_{ij}} &\rightarrow H_{ij} \\ W_{ij}\ \frac{(D\ H^T)_{ij}}{(W\ H\ H^T)_{ij}} &\rightarrow W_{i,j} \end{aligned} \tag{3}$$

In practice, start W and H at random but positive initial values and apply these update rules iteratively. Common practice is to try many different initial conditions. Notice that the updates are done on each element ij of H and W. For example, $(W^T\ D)_{i,j}$ takes the element (i, j) of the matrix that results from $W^T\ D$. The operations in (3) can be done very efficiently as matrix multiplications. You just need to be careful to perform element-wise division.

Finally, keep in mind that the optimization problem framework has many interesting extensions. For example, sparse PCA attempts to find a solution to (1) that has a small number of non-zero terms by adding an 'L1' penalty to the optimization. See the optional problem if you are interested in understanding this more deeply.

$$\text{argmin}_{V_k} ||D - V_k\ V_k^T\ D||_2 + \alpha\ ||V_k||_1$$

where $||V_k||_1 = \sum_{i,j} |V_{i,j}|$, and $||$ is absolute value. The parameter α controls the weighting of the L1 norm in the optimization. This L1 norm is commonly used to push entries towards 0. See the optional problem if interested.

Problem 1: Gene expression programs for an orthogonal cell

Consider a cell that has two gene expression programs 'growth' and 'starvation'. When a cell is in 'growth', it expresses three genes, g_1, g_2, g_3, and the genes are Poisson distributed with means $100, 125, 130$ respectively. When a cell is in starvation, it expresses three genes s_1, s_2, s_3, and s_1, s_2, s_3 are Poisson distributed with means $20, 30, 40$. For a given cell, consider the state of the cell to be a vector $(g_1, g_2, g_3, s_1, s_2, s_3)$ of integer gene expression counts and the mean expression vectors are $\mu_g = (100, 125, 130, 0, 0, 0)$ and $\mu_s = (0, 0, 0, 20, 30, 40)$.

(A) Write a computer program to generate samples of cells from the growth and starvation condition by drawing vectors from appropriate Poisson distributions. Construct a data set that has at least 100 samples of growing cells and 100 starving cells. Represent your sampled data as a matrix and plot the matrix.

(B) Mean center the data and calculate the mean centered, gene-gene co-variance matrix. Plot this covariance matrix as a heatmap.

(C) Find the eigenvectors and eigenvalues of the gene-gene covariance matrix and display them. Comment on the eigenvectors and eigenvalues and their relationship to the gene expression programs. Do the eigenvector entries mirror the structure of the gene expression programs?

(D) Project the mean centered data along the first and second eigenvector from (C) and plot a scatter plot the projected data.

Problem 2: Gene expression programs for an non-orthogonal cell

Consider a cell as before with a 'growth' and starvation state. But now the gene expression programs are not orthogonal. In the growth state, the mean gene expression state is $(60, 0, 80, 10, 50, 50, 0, 0)$ (again gene expression is Poisson distributed). In the starvation state, the gene expression state is $(0, 30, 10, 0, 30, 50, 50, 10)$. Notice that these vectors are not orthogonal and share genes.

(A) Generate synthetic growth and starvation data by sampling Poission distributed values. As before, plot the gene-gene covariance matrix and raw data heatmap.

(B) Calculate the eigenvalues and eigenvectors of this matrix, plot, and comment.

(C) Implement non-negative matrix factorization (nnmf) using the multiplicative update rule described above. Perform nnmf, using your implementation for inner dimension $k = 1, 2, 3$. Plot a heat map of H and W and comment on your results.

Problem 3: PCA and NNMF on PBMC data

Down-load my version of the PBMC data from this link. I have organized the data to make interpretation as easy as possible. You may sub-sample cells from the data if you have trouble with computation. For example, select 1000 cells.

https://www.dropbox.com/sh/xnz1vvlt290lzap/AAAqlEnO6jsZCYrf1-FEysYFa?dl=0

(A) Perform PCA on the data by calculating the mean centered gene-gene covariance matrix and finding its eigenvectors and eigenvalues. Plot a heat map of the covariance matrix, and plot the first 20 eigenvalues arranged in decreasing order.

(B) Project the data into the first two principal components and plot a scatter plot of this projection.

(C) Project the data onto the first 3 principal components and plot a scatter plot. Note in B and C, you should see clusters in the data.

(D) Show a heatmap of the first 2 principal component vectors next to a heat map of the data matrix. Compare these vectors to the heatmap of the data matrix and comment. Do the principal components capture the blocks in the data?

(E) Use sklearn.decomposition.NMF to perform Non-negative matrix factorization on the data for $k = 4, 20, 40$ plot the H and W matrix for each k and comment on the results. For each value of k have the function to try multiple replicates. Do you capture the gene expression programs you observe in the data?

Note: I was able to run this on my laptop. However, if you have problems, sub-sample the cells by taking out 1000 cells.

(F) Look at the W matrix, for one of the W vectors select 5 genes with the top coefficient value. Write down their names and look up their function.

(G) perform NNMF for $k = 1$ to $k = 40$, and plot the reconstruction error as a function of k. What do you find?

(H) Look up sklearn.decomposition.SparsePCA. What does α mean in the function call?

Extra Problem: Sparse solutions and the L1 norm

The goal of this problem is to give you an intuition for the L1 norm and its connect with sparse solutions to systems of linear equations.

(A) We are going to solve the following system of linear equations using two different types of constraints:

$$2x - y = 5.$$

For part A, solve the following optimization problem:

$$\text{argmin}_{x,y}\ (2x - y - 5)^2 + .01\ (|x| + |y|).$$

You can solve by finding when $(2x - y - 5)^2$ and minimizing $(|x| + |y|)$ on this line.

(B) Represent the solution graphically from part (A) by (i) plotting the line $2x - y = 5$ and (ii) plotting (on the same plot) solutions to $|x| + |y| = 2.5$ (remember to consider all four quadrants of the graph).

(C) Now, solve the following optimization problem:

$$\text{argmin}_{x,y}\ (2x - y - 5)^2 + .01\ \sqrt{(x^2 + y^2)}.$$

You can solve using the same strategy as in part (A).

(D) As before, represent the solution graphically, by plotting the line and then plotting the solution to $\sqrt{(x^2 + y^2)} = 2.23$