Recitation 6

3D Chromatin Structure & Dimensionality Reduction

Ge (Saber) Liu, Corban Swain

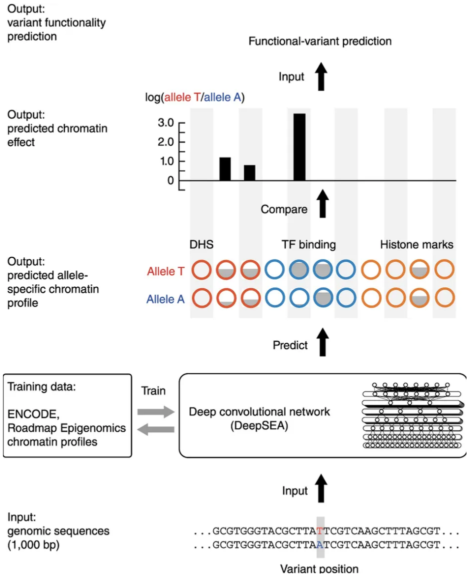
MIT - 6.802 / 6.874 / 20.390 / 20.490 / HST.506 - Spring 2020

2020-03-12 / 2020-03-13

Outline

- DeepSEA
- Linear algebra basics
- Principle component analysis (PCA)
- t-SNE and parametric t-SNE
- Auto-encoder
- U-MAP

deep learning-based sequence analyzer (DeepSEA)



Linear algebra basics

Eigenvector: An eigenvector or characteristic vector of a linear transformation T is a non-zero vector that changes by only a scalar factor when that linear transformation is applied to it.

$$T(\mathbf{v}) = \lambda\mathbf{v} \quad \text{or} \quad \mathbf{A}\mathbf{v} = \lambda\mathbf{v} \quad \text{if the transformation can be represented as a } \textbf{square matrix } A$$

where λ is a scalar, known as the **eigenvalue**, characteristic value, or characteristic root associated with the eigenvector $\mathbf{v}$. An NXN matrix has at most N **linearly independent** eigenvectors.

Eigen-decomposition: Factorization of a matrix into a canonical form, whereby the matrix is represented in terms of its eigenvalues and eigenvectors.

$$A = Q\Lambda Q^{-1}$$

where Q is the square NXN matrix whose i-th column is the eigenvector v_i of A, and Λ is the **diagonal matrix** whose diagonal elements are the corresponding eigenvalues, $\Lambda_{ii} = \lambda_i$. Note that $\mathbf{A}$ has to have N linearly independent eigenvectors (Only diagonalizable matrices can be factorized in this way).

Singular value decomposition(SVD): factorization of a real or complex matrix $\mathbf{M}_{m\times n}$ into $\mathbf{M} = \mathbf{U}\mathbf{V}^*$ where $\mathbf{U}_{m\times m}$ and $\mathbf{V}_{n\times n}$ are unitary matrix with orthonormal eigenvectors of $\mathbf{M}\mathbf{M}^*$ and $\mathbf{M}^*\mathbf{M}$, and $_{m\times n}$ is an rectangular diagonal matrix with non-negative real numbers on the diagonal.X^* means the conjugate transpose of X.

Linear algebra basics

Special matrices

Real symmetric matrices: Matrix $\mathbf{A}$ is symmetric if $\mathbf{A} = \mathbf{A^T}$

Theorem

Any symmetric matrix:

- has only real eigenvalues
- is always diagonalizable
- has **orthogonal** eigenvectors

Corollary: If matrix A is symmetric then there exists $Q^T Q = \mathcal{I}$ such that $A = Q^T \Lambda Q$.

Positive definite matrices: A symmetric matrix $\mathbf{A}$ is positive definite/semi-definite if all its eigenvalues are positive/non-negative.

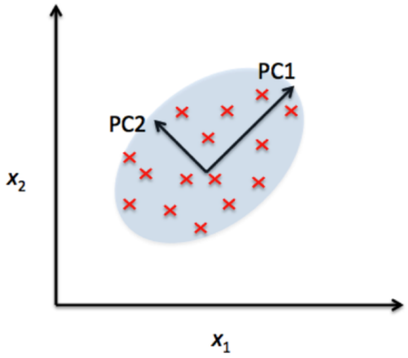
Theorem

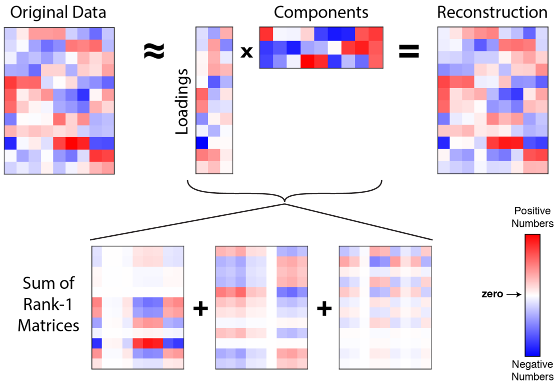
$\mathbf{A}$ is positive definite if and only if $x^T A x > 0, \forall x \neq 0$.

Diagonalizable $\supset$ Symmetric $\supset$ Positive semi-definite $\supset$ Positive definite

PCA

Principal component analysis: Orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components. This transformation is defined such that the first principal component has the **largest possible variance**, and each succeeding component in turn has the highest variance possible under the constraint that it is **orthogonal** to the preceding components. The resulting vectors are **linear combination** of the variables and form an **orthogonal basis set**.





PCA

Principal component analysis: Consider a data matrix $\mathbf{X}_{mxn}$ with m examples of n dimensional features (each dimension has been z-centered such that the mean is zero).Try to find a set of n-dimensional vectors of weights or coefficients $\mathbf{w}_{(k)} = (w_1, \dots, w_n)_{(k)}$ that projects each row vector $\mathbf{x}_{(i)}$ of $\mathbf{X}$ to a new vector of principal component scores $\mathbf{t}_{(i)} = (t_1, \dots, t_n)_{(i)}$, given by $t_{k(i)} = \mathbf{x_{(i)}} \cdot \mathbf{w}_{(k)}$

$$\begin{bmatrix} - & \mathbf{x_{(1)}} & - \\ & \vdots & \\ - & \mathbf{x_{(m)}} & - \end{bmatrix}_{m\times n} \times \begin{bmatrix} | & & | \\ \mathbf{w_{(1)}} & \dots & \mathbf{w_{(n)}} \\ | & & | \end{bmatrix}_{n\times n} = \begin{bmatrix} - & \mathbf{t_{(1)}} & - \\ & \vdots & \\ - & \mathbf{t_{(m)}} & - \end{bmatrix}_{m\times n} = \begin{bmatrix} | & & | \\ \mathbf{PC_{(1)}} & \dots & \mathbf{PC_{(n)}} \\ | & & | \end{bmatrix}_{m\times n}$$

$$\mathbf{T} = \mathbf{XW}$$

The weights are constrained to be a unit vector such that $\mathbf{w_{(i)}^T w_{(i)}} = 1$.

PCA

To solve for projection with maximized variance, the **first** weight vector has to satisfy:

$$\mathbf{w_1} = \underset{\mathbf{w^T w}=1}{\operatorname{argmax}} \sum_i (\mathbf{x_i} \cdot \mathbf{w_1})^2 = \underset{\mathbf{w^T w}=1}{\operatorname{argmax}} \mathbf{w^T X^T X w} = \operatorname{argmax} \frac{\mathbf{w^T X^T X w}}{\mathbf{w^T w}}$$

A standard solution to this optimization for a positive semidefinite matrix such as $X^T X$ is the largest eigenvalue of the matrix, which occurs when w is the corresponding eigenvector. The rest of the component can be given as:

$$\mathbf{w_k} = \operatorname{argmax} \frac{\mathbf{w^T X_k^T X_k w}}{\mathbf{w^T w}} \quad \text{where} \quad \mathbf{X_k} = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X w_s w_s^T}$$

It turns out that this gives the remaining eigenvectors of $\mathbf{X^T X}$, with the maximum values equal to the corresponding eigenvalues. Thus, solving weight vectors for PCA is equivalent to finding the eigenvectors of $\mathbf{X^T X}$ and sorting by its corresponding eigenvalues.

The SVD of $\mathbf{X} = \mathbf{UW^T}$, so $\mathbf{T} = \mathbf{XW} = \mathbf{UW^T W} = \mathbf{U}$. Each column of $\mathbf{T}$ is given by one of the left singular vectors of $\mathbf{X}$ multiplied by the corresponding singular value.

SNE

Stochastic neighbor embedding (SNE): An unsupervised nonlinear dimensionality reduction technique where the goal is to find a low-dimensional (2-dimensional) representation of the original inputs such that pairwise similarity are best preserved and the inherent clustering structure can be visualized.

Similarity in original input space: the similarity of datapoint x_i to datapoint x_j is defined as the conditional probability, $p(x_j|x_i)$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i. Given an input matrix $\mathbf{X}$, in which each row is a sample x_i and each column represent a feature dimension, the pairwise similarity is defined as:

$$P_{i,j} = p(x_j|x_i) = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

We define $\beta_i = 1/\sigma_i$ which is equivalent to the **precision** of a multivariant Gaussian. We further set constraints on the perplexity of the conditional distribution where **perplexity** is defined using Shannon entropy of P_i:

$$Perplexity(P_i) = 2^{H(P_i)} \quad \text{and} \quad H(P_i) = -\sum_j p_{x_j|x_i} \log_2 p_{x_j|x_i}$$

In order to gaurantee the perplexity constraint for each P_i, we need to find the corresponding β_i such that the resulting distribution has the desired perplexity. Given that $Perplexity(P_i)$ is a **monotonically decreasing** function of β_i, we could use binary search to estimate the solution for β_i.

non-parametric t-SNE

t-distribution Stochastic neighbor embedding (t-SNE)

There are several modifications we need to make:

- **Use symmetric similarity matrix instead:** Since the gradients for the conditional distribution is hard to compute, people use the joint distribution as an alternative that is "just as good" and this gives symmetric distribution matrix:

$$P_{ij}^{symmetric} = p(x_i, x_j) = \frac{\exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} \exp(-||x_l - x_k||^2/2\sigma^2)} = \frac{p(x_j|x_i) + p(x_i|x_j)}{2N}$$

- **Use Student t-distribution for similarity in embedded space:** We look for a 2D representation of **X** which is **Y** (Nx2 matrix), such that the pairwise similarity on **Y**:

$$Q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

  is similar to P_{ij}. Which is is equivalent to minimizing the **KL-divergence** between P and Q:

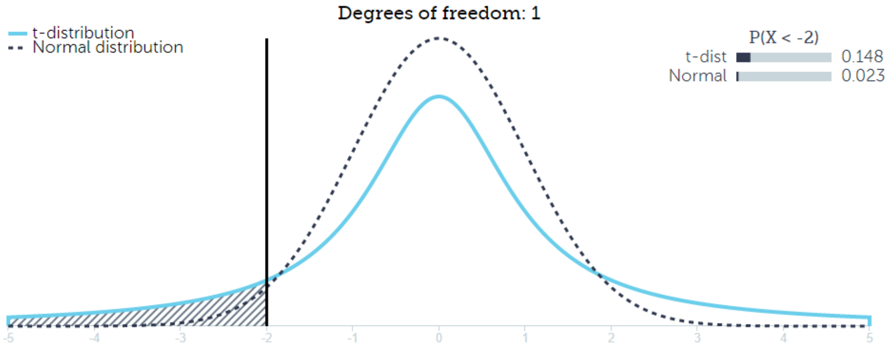
$$C = KL(P||Q) = \sum_i \sum_j P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

  The gradients for conducting gradient descent are:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

non-parametric t-SNE

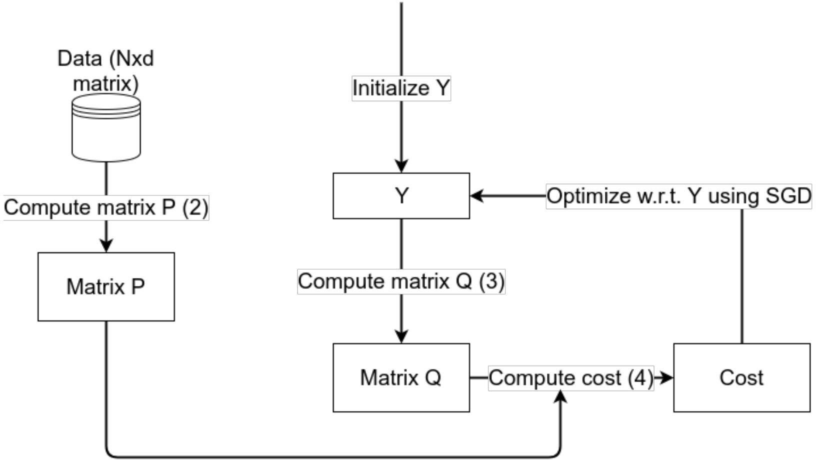
Evaluating distance using the *t*-statistic allows for pairs of points seperated by large distances to maintain significant probability mass compared to that for a gaussian distance. This is because the *t*-distribution has a "heavier tail" than the gaussian distribution. We want to maintain the probability mass for these distances so during the gradient update of embedded locations there is a "force" to bring locally clustered points in high-D space together in low-D space, even if they are initialized as far apart in the t-SNE scheme.



non-parametric t-SNE

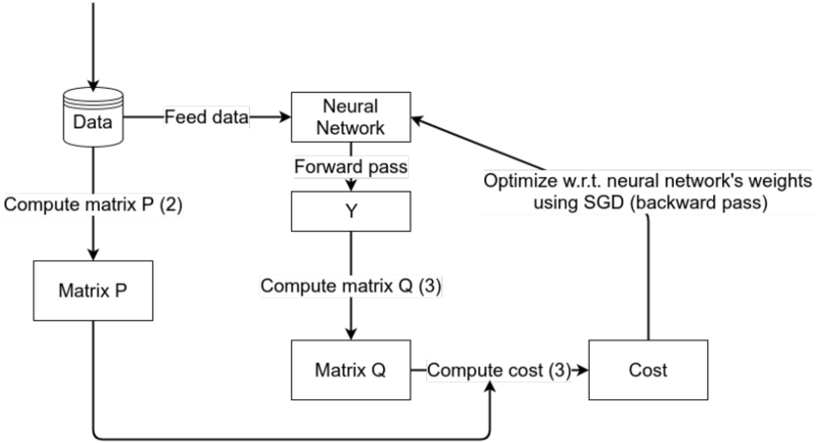
The non-parametric t-SNE has several drawbacks for being non-parametric.

- You can't embed new points that weren't used in the training phase without running the algorithm from scratch again and without preserving the previous embedding results.
- It is not scalable because the more points you have the larger memory you need to store **D**,**P** and **Q**.



parametric t-SNE

We could think of a parametric approach instead, in which we want to build a model $y = f(x, \Theta)$ that maps any given input x_i to low-dimensional output y_i. A useful family of model we would consider is of course the **neural networks**. The good thing about this is that we can calculate P on a smaller batch of X and train model using batched data. Also once the model is trained, we will have a deterministic embedding that can be calculated within linear time.

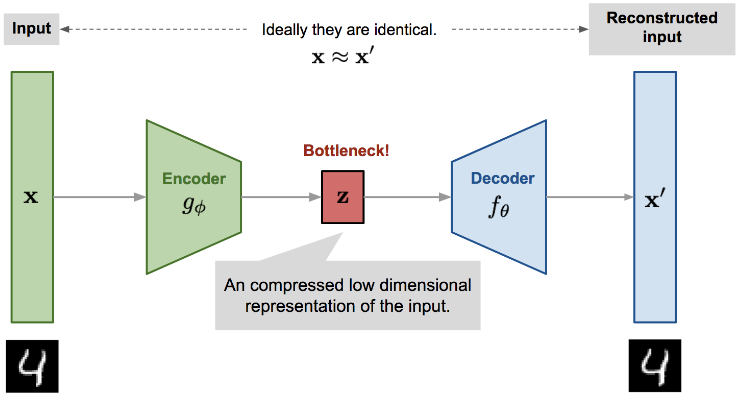


Practical Considerations for Dimensionality Reduction

- It may be necessary to first process raw data using principal component analysis then use the top k PCA loadings as a proxy for the raw data to perform t-SNE visualization. This could prevent issues and provide computational speed-up when trying to down-map from very high dimensional data to very low dimensnioal spaces.

Auto-Encoder

Auto-encoder: An autoencoder learns to compress data from the input layer into a low dimensional representation, and then uncompress that representation into something that closely matches the original data.


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

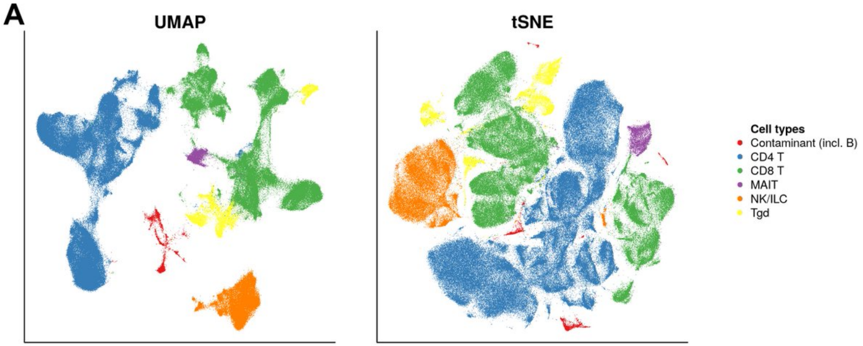
$$\mathbf{z} = g_\Phi(\mathbf{x}) \qquad \mathbf{x}' = f_\Theta(\mathbf{z}) = f_\Theta(g_\Phi(\mathbf{x})) \qquad \mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2$$

U-MAP

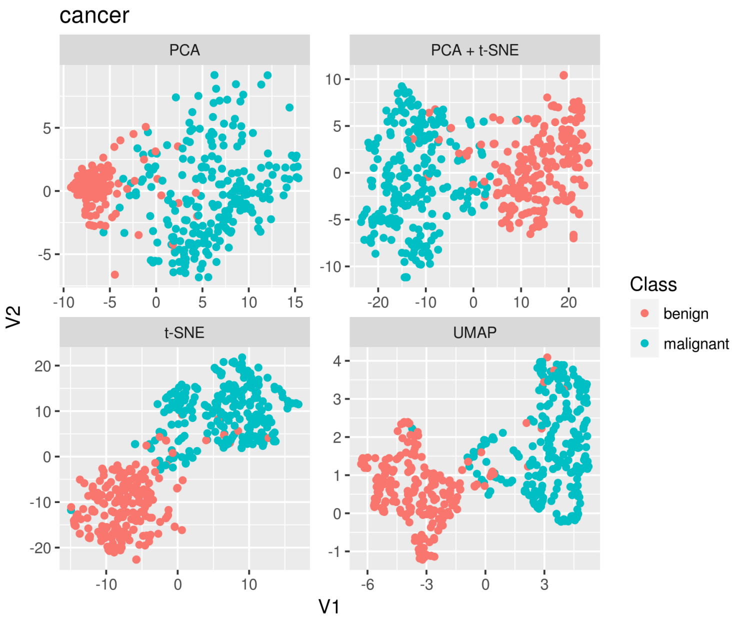
Uniform Manifold Approximation and Projection (UMAP): a general dimension reduction technique based on manifold learning techniques and topological data analysis. The algorithm is founded on three assumptions about the data:

- The data is uniformly distributed on Riemannian manifold
- The Riemannian metric is locally constant (or can be approximated as such)
- The manifold is locally connected

The embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure.



Dimension reduction algorithms


cancer
PCA
PCA + t-SNE
t-SNE
UMAP
V1
V2
Class
benign
malignant