6.7900 Fall 2024: Lecture Notes 19

1 PCA Setup (Review from Last Lecture)

We have a dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, \dots x^{(n)}\}$ on which we'd like to perform "dimensionality reduction".

- $x^{(n)}$ is a $D \times 1$ vector
- Pre-process by zero-centering so $\frac{1}{N}\sum_{n=1}^{N} x^{(n)} = 0_D$
- Assume: we want to approximate the data with its projection onto a low-dimensional subspace with orthonormal basis $w_1, ..., w_L$, which are the principal components. $x^{(n)} \approx \sum_{\ell=1}^{L} z_\ell^{(n)} w_\ell = W z^{(n)} =: \hat{x}^{(n)}$
- Goal: the projection should be "close" to the original data (square loss):

$$\min \sum_{n=1}^{N} \|x^{(n)} - \hat{x}^{(n)}\|^2 = \min_{W,Z} \|X^T - WZ^T\|_F^2 = \min_{W,Z} \|X - ZW^T\|_F^2$$

> **Note:** The Frobenius Norm of a matrix is defined as the square root of the sum of the absolute squares of its elements.

In the previous lecture, we solved the $L = 1$ case, where we approximate our data with only one dimension. We found that the solution to the objective is

$$z_1^{(n)} = w_1^T x^{(n)}$$

where w_1 is the solution to the optimization problem

$$\arg\min_{w_1} -\sum_{n=1}^{N} (w_1^T x^{(n)})^2$$

2 PCA as Finding Direction of Maximum Variance

We claim that PCA finds the direction that maximizes the variance of the projected data. To see this for the $L = 1$ case, let's compute the empirical mean and variance of the projected data (the z's).

- $L = 1$, empirical mean.

$$\sum_{n=1}^{N} z_1^{(n)} = \sum_{n=1}^{N} w_1^T x^{(n)} = w_1^T \sum_{n=1}^{N} x^{(n)} = 0$$

  In the above, we first simplify our expression for the mean by plugging in our solution from last lecture. Next, we recognize that w_1^T is constant in the sum.

  Exercise: How did we get the final step? Hint: what did we do to our data?

- $L = 1$, empirical variance. In writing the empirical variance, we use the fact that the empirical mean is 0.

$$\sum_{n=1}^{N} (z_1^{(n)})^2 = \sum_{n=1}^{N} (w_1^T x^{(n)})^2$$

  Exercise: How does this relate to our PCA objective from last lecture?

We see that our PCA objective, which finds w_1 such that

$$\arg\min_{w_1} - \sum_{n=1}^{N} (w_1^T x^{(n)})^2$$

maximizes the empirical variance of the projected data.

Note: In general, PCA returns the L eigenvectors of the empirical covariance matrix $\hat{\Sigma}$ with the L largest eigenvalues. Intuitively, it finds L the orthogonal directions along which our data has the most variance.

3 When does PCA work and not work?

Going back to our ball example, running PCA with $L = 1$ correctly the direction associated with the ball's motion (Figure 1). However, we should be aware of the following limitations of PCA:

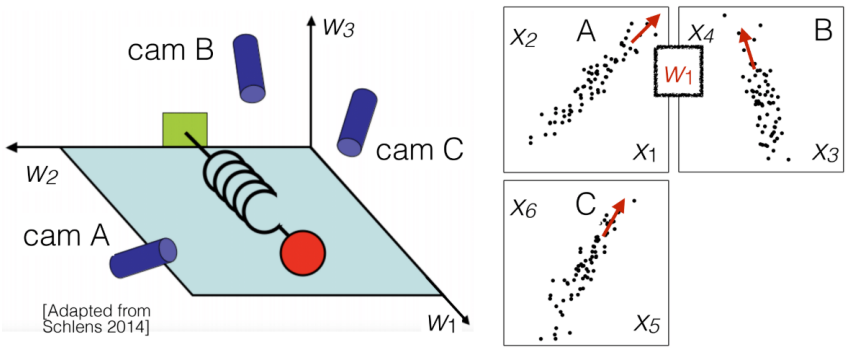

Figure 1: PCA correctly identifies the direction associated with the ball's motion

- Challenge 1: Determining how to pick L. If we have domain knowledge, as in our ball example, determining L may be easy. However, without this domain knowledge, it is hard to pick an "optimal" L (or even define what that might mean). In fact, as L increases, the "error" measuring how well PCA fits the data strictly decreases.

  Exercise: As L increases, why must the "error" measuring how well PCA fits the data strictly decrease?

- Challenge 2: PCA is sensitive to units of measurement. Since PCA captures directions with maximum variance, it is sensitive to changes in units, or normalization.

- Challenge 3: PCA does not necessarily learn "useful" features. Directions of maximum variance may be completely irrelevant to some machine learning objective you care about.

- Challenge 4: PCA has limited representational power. PCA finds an orthogonal basis in the original feature space – it can only rotate our data! Example: say your data is best described by polar coordinates. Unfortunately, PCA will not be able to find this solution.

4 t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is another kind of dimensionality reduction approach. It's different from PCA in the following ways:

- Non-linear. PCA is linear.

- Designed for visualization – maps to 2 dimensions.
- Aims to preserve local relationships. PCA gives a map based on all training points.
- Unlike PCA, the computed mapping cannot be applied to new data-points.

Let's first talk about the SNE in t-SNE. SNE's mapping algorithm attempts to map data from the original high-dimensional space to 2D, while preserving neighbors. Let $x^{(n)}$ denote each data point in the original space and $z^{(n)}$ be each corresponding point in 2D. SNE defines:

$$p_{m|n} := \frac{exp\Big(-\frac{1}{2\sigma_n^2}||x^{(n)} - x^{(m)}||^2\Big)}{\sum_{n':n'\neq n} exp\Big(-\frac{1}{2\sigma_n^2}||x^{(n)} - x^{(n')}||^2\Big)}$$

which is a the similarity of each point $x^{(n)}$ in the original space to each other point in the space $x^{(m)}$ as a probability distribution. It also defines:

$$q_{m|n} := \frac{exp\Big(-||z^{(n)} - z^{(m)}||^2\Big)}{\sum_{n':n'\neq n} exp\Big(-||z^{(n)} - z^{(n')}||^2\Big)}$$

which is the same but for the corresponding points in 2D.

Note: Notice the $-\frac{1}{2\sigma_n^2}$ which have a hyperparameter σ_n for each n. These hyperparameters controls the distance "units" for each datapoint and can be used to tune the number of relevant neighbors. This flexibility is useful if in our dataset, some datapoints are close to many other datapoints in high dimensional space while others are not.

SNE will optimize the z's to make the distributions p and q as close as possible for all n. Specifically, the Kullback-Leibler (KL) divergence is used as the metric to optimize:

$$\min_{z_1,z_2,\ldots,z_n} \sum_{n=1}^{N} \mathcal{D}_{KL}(p_{\cdot|n}||q_{\cdot|n})$$

Note that the KL divergence is asymmetric:

$$\mathcal{D}_{KL}(p_{\cdot|n}||q_{\cdot|n}) = \sum_{m:m\neq n} p_{m|n}\frac{p_{m|n}}{q_{m|n}}.$$

Exercise: What relative values of p and q will cause us to incur the most penalty? What does this intuitively tell us about how the algorithm will preserve relationships among neighbors and non-neighbors?

The asymmetry implies that when p is large (m is a neighbor of n in the original space), we also need q large (m is a neighbor of n in the 2D space), otherwise we will incur a large cost. This property preserves local relationships! Similarly, we see that when p is small, we don't really care about the value of q, so global relationships are not necessarily likely to be preserved.

Some final notes about t-SNE:

- The SNE optimization problem is not convex; may be hard to optimize
- In practice, the user sets a perplexity value, which is then used to tune σ_n for each n. Intuitively, this does something approximately close to standardizing the number of neighbors for each n.
- A heavier-tailed distribution like the student's t distribution is often used instead of Gaussian to get better performance in practice (hence the name t-SNE).

Note: You can play around with t-SNE here:
`https://distill.pub/2016/misread-tsne/`