Chapter 13
Dimensionality Reduction / PCA

13.1 The Curse of Dimensionality

In your learning up until now, you have likely encountered the **curse of dimensionality**. The curse of dimensionality is when a data set has many features, making it difficult to create and train models, interpret them, and visualize the data. In addition, high dimensional data requires a greater number of samples to be able to accurately fit a model to it.

The number of samples that are required increases exponentially in relation to the dimension. This is similar to filling the volume of the dimensional space. For instance, a 2x2 plot has area 4, while a cube has a volume of 8, and in the 4th dimension 16, etc. This illustrates how in a higher dimension the amount of samples required to fit well in a low dimension will scale exponentially.

Example(s)

Some examples of high dimensional data occur in computer vision, where each pixel of an image represents a feature. For color images, each pixel could be three features for the corresponding red, green, and blue channels.
Another example is DNA analysis, which for different types of models can vary in size immensely, going from only a few base pairs to the size of the human genome which is 6.4 billion base pairs long.

Despite all having a huge amount of data acquired, being able to understand and make predictions on such high dimensional data is not an easy task. Many features correlates to slower training and more complex models.

13.2 Dimensionality Reduction

However, hope is not completely lost for interpreting high dimensional data. A key tool for understanding and visualizing such data is dimensionality reduction. The key idea here is that low dimensional data is easy to work with, so if there was some way to convert our high dimensional data into a lower one, without throwing away too much information, it would be much better to analyze that data in a reduced dimension.

Definition 13.1: Dimensionality Reduction

Dimensionality Reduction is the the task of representing the data with fewer dimensions, while keeping meaningful relations between data.

It is not always the case that data has some low dimensional representation that is accurate, however, in many cases dimensionality reduction is incredibly applicable.

Example(s)

For example, in natural language processing, we would like to understand relations between words (which ones are similar and which ones are not). We can create word embeddings which can group certain types of words together.

We previously learned about the flaws of bag of words model for predicting the sentiment of a restaurant review. Ideally, words like "bad", "angry", "hate", etc. should be grouped together, however a bag of words format would ignore these similarities. By using some type of dimensionality reduction, words as features could be better represented by the types of sentiment they are associated with which can better help train some classifier.

Dimensionality reduction is also incredibly useful for visualization and clustering. Imagine we would like to group some 2,000 cells into specific types using gene expression levels for 2 genes in each cell. This sounds pretty easy. We can simply plot each gene expression on the x and y axis and see if certain cells form clusters. In the real world though, analyzing cells only using 2 gene expression levels is trivial, usually we might analyze some 10,000 gene expression levels. All of a sudden, visualizing the differences between cells is not so easy.

We would like to see if we could convert this 10,000 dimension data into 2 or 3 dimension so that we could plot the cells to identify differences as we could have done if we only analyzed 2 gene expression levels. Using dimensionality reduction, there are ways to convert such high dimensional data down to 2 dimensions to show differences and similarities between data points by capturing some axis as a combination of features.

To conclude, here are some benefits dimensionality reduction can provide:

- **Easier learning.** Fewer parameters, no curse of dimensionality
- **Visualization.** Beyond the 3rd dimension, visualization becomes difficult.
- **"Discovering intrinsic dimensionality".** Often high dimensional data is truly better represented in a lower dimension, as there can be a lot of redundant information.

13.3 PCA

Definition 13.2: Principal Component Analysis

PCA is a common dimensionality reduction algorithm which linearly projects d-dimensionality data to k-dimension data where $k \leq d$.

Our goal in PCA is to find some linear projection which minimizes reconstruction error for the projected data. This reconstruction error can be thought of as the information that is lost when you attempt to recreate the original data from the projection ("undo" the projection).

A linear projection simply uses a linear combination of features, each feature weighted by some value, similar to a linear regression, which can be used to convert a high dimensional data set to a lower one by using some lower number of these linear combination vectors.

In essence, this creates a new feature which comprises many of the original features. Although this seems naive, often this can be useful as a certain combination of features might contain meaningful information.

Example(s)

Here are k new features z created from the original d features x where the coefficients are something

random (real projections from PCA will not have these values):

$$z_i[1] = 2x_i[1] + 3x_i[2] - 7x_i[3] + ... + 5x_i[d]$$
$$z_i[2] = 5x_i[1] + 0x_i[2] + 4x_i[3] + ... - 2x_i[d]$$
$$...$$
$$z_i[k] = ...$$

The most simple projection that PCA can accomplish is projecting 2 dimensions of data down to one dimension. Remember that our goal is to fit some line (1 dimension) to the 2 dimensional data which minimizes reconstruction error, which is our standard sum of squared differences.

$$\text{PCA reconstruction error} = ||x_i - \hat{x}_i||_2^2 \text{ where } \hat{x}_i = z_i u_1 \tag{13.27}$$

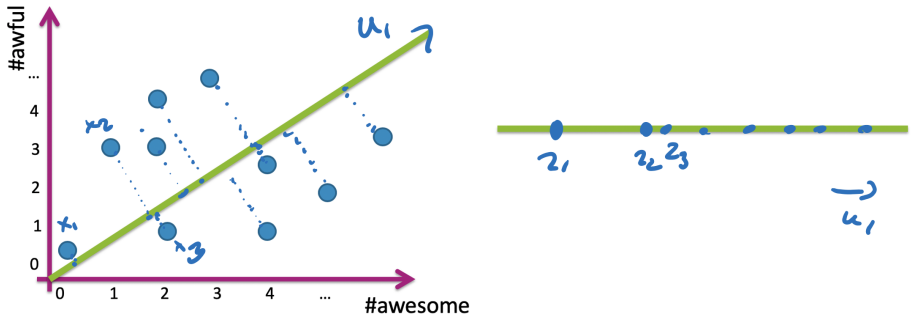

Figure 13.72: Fitting a line u_1 which minimizes reconstruction error.

In PCA, the first feature we project onto is the one that offers the lowest reconstruction error out of any single linear projection (onto 1 dimension), or similarly captures the most variance of the original data in that projected space. In the figure above we see that best line as u_1. This is called our 1st **principal component**.

If we wish to project data to some dimension greater than 1, then we can use more principal components. Each principal component is orthogonal to the previous. For instance, the next principal component we would select would look like an orthogonal line on our data from before.

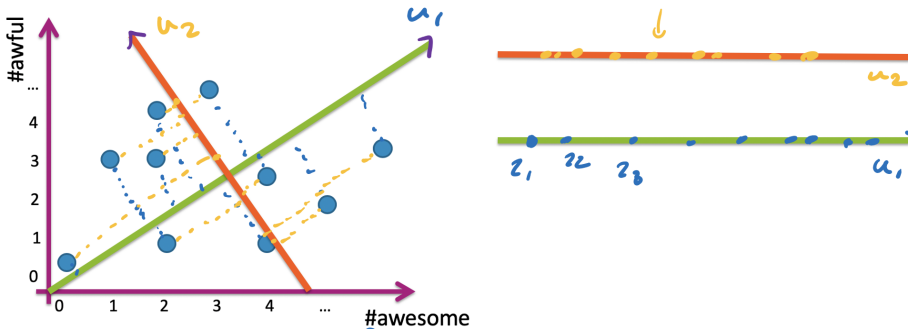

Figure 13.73: Fitting the second principal component.

As you may have guessed, with $k = d$ our reconstruction error goes to 0 and we have captured 100 percent of the variance from the original data.

To compute some of the data's principal components we must run the PCA algorithm.

Algorithm 5 PCA Algorithm

Step 1: Recenter the data by subtracting the mean from every row.
- $X_c = X - \bar{X}[1:d]$

Step 2: Compute spread/orientation (covariance matrix Σ)
- $\Sigma[t,s] = \frac{1}{n}\sum_{i=1}^{n} x_{c,i}[t]\, x_{c,i}[s]$

Step 3: Find basis for orientation (computer eigenvectors of Σ)
- Select k eigenvectors $u_1, u_2, ..., u_k$ with the largest corresponding eigenvalues (u_1 will have the largest eigenvalue and u_2 the second and so on).

Step 4: Project Data onto principal components
- $z_i[1] = u_1^T x_{c,i} = u_1[1]\, x_{c,i}[1] + ... + u_1[d]\, x_{c,i}[d]$

  ...

  $z_i[1] = u_k^T x_{c,i} = u_k[1]\, x_{c,i}[1] + ... + u_k[d]\, x_{c,i}[d]$

$=0$

When PCA is applied to different high dimension data sets, it can provide interesting and meaningful principal components.

Example(s)

In one data set of genes–500,568 base pairs each, for 3,191 people, labeled with their country of origin–PCA shows a low dimensional unexpected meaning. The first two principal components represent latitude and longitude, which can reconstruct the map of Europe using the original data colored in by their country of origin. This showcases the power of PCA as despite having a huge number of features, there is a valid and meaningful, low dimensional subspace for the data. This makes sense as genotypes are likely similar for people from the same areas around the world.
This is a unique and interesting example of how high dimension and complex data can have a meaningful low dimensional representation.

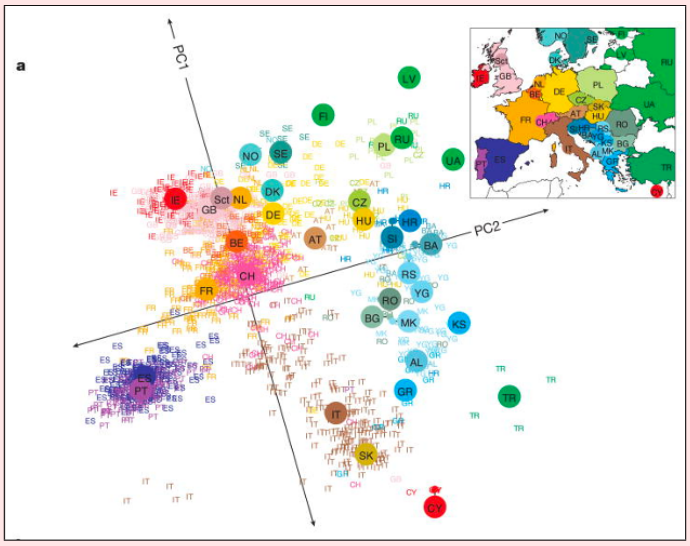

Figure 13.74: Genes for 3191 people projected using two principal components.

13.4 PCA Reflection

It is important to note that there are some major caveats of PCA. First, an integral part of being able to run PCA is computing the covariance matrix Σ which is a $d \times d$ matrix. For something with 10,000 features, this can be quite large and take a long time to compute.

In practice, singular value decomposition is used (SVD) to find k eigenvectors with the largest k eigenvalues, which can be done more quickly than computing the entire covariance matrix. While SVD will not be covered in this course, it is important to understand it is likely to be used in practice when using PCA.

Another major caveat of PCA is that it is simply a linear combination of features for each principal component, this means that PCA assumes there is a lower dimensional linear subspace that represents data well, but this is not always the case.

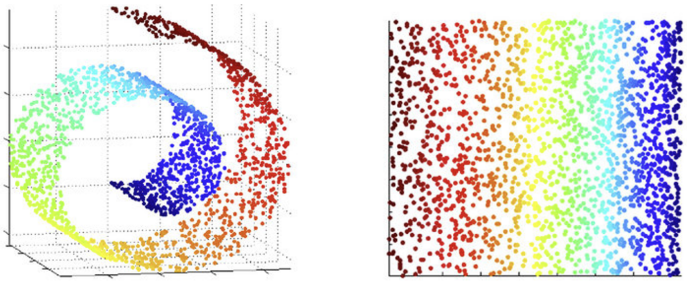

Figure 13.75: 3D to 2D to PCA example where there is no linear low dimension subspace.

In conclusion, PCA is an important algorithm for dimensional reduction which can help avoid the issues that come with the curse of dimensionality. We learned that the goal of PCA is to minimize reconstruction error / maximize captured variance for some k number of dimensions. While it has its drawbacks in that it

assumes a linear subspace, it is usually worth exploring what PCA can produce for high dimensional data sets as interesting and important meaning is often found. Note: for the following sections, readers can think of the word "consumed" being analogous to "bought" for storefronts like Amazon or "watched" for streaming services like Netflix.

13.5 Recommender Systems