[Summary]Dimensionality Reduction

Motivation I: Data Compression

- We may want to reduce the dimension of our features if we have a lot of redundant data.
- To do this, we find two highly correlated features, plot them, and make a new line that seems to describe both features accurately. We place all the new features on this single line.

Doing dimensionality reduction will reduce the total data we have to store in computer memory and will speed up our learning algorithm.

Note: in dimensionality reduction, we are reducing our features rather than our number of examples. Our variable m will stay the same size; n, the number of features each example from x^1 to x^m carries, will be reduced.

Motivation II: Visualization

It is not easy to visualize data that is more than three dimensions. We can reduce the dimensions of our data to 3 or less in order to plot it.

We need to find new features, z_1, z_2(and perhaps z_3) that can effectively **summarize** all the other features.

Example: hundreds of features related to a country's economic system may all be combined into one feature that you call "Economic Activity."

Principal Component Analysis Algorithm

Before we can apply PCA, there is a data pre-processing step we must perform:

Data preprocessing

- Given training set: x(1),x(2),…,x(m)
- Preprocess (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^i$$

- Replace each x_j^i with $x_j^i - \mu_j$
- If different features on different scales (e.g., x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

Above, we first subtract the mean of each feature from the original feature. Then we scale all the features $x_j^i = \frac{x_j^i - \mu_j}{s_j}$

We can define specifically what it means to reduce from 2D to 1D data as follows:

$$\sum = \frac{1}{m}\sum_{i=1}^{m}(x^i)(x^i)^T$$

The z values are all real numbers and are the projections of our features onto u^1

So, PCA has two tasks: figure out $u^1, \ldots, u^k$ and also to find $z_1, z_2, \ldots, z_m$

1. Compute "covariance matrix"

$$\sum = \frac{1}{m}\sum_{i=1}^{m}(x^i)(x^i)^T$$

2. Compute "eigenvectors" of covariance matrix Σ

[U,S,V] = svd(Sigma);

svd() is the 'singular value decomposition', a built–in Octave function.

What we actually want out of svd() is the 'U' matrix of the Sigma covariance matrix: $U \in \mathbb{R}^{n \times n}$. U contains $u^1, \ldots, u^n$, which is exactly what we want.

3. Take the first k columns of the U matrix and compute z

We'll assign the first k columns of U to a variable called 'Ureduce'. This will be an n ×k matrix. We compute z with:

$$z^i = Ureduce^T \cdot x^i$$

$UreduceZ^T$ will have dimensions k×n while x(i) will have dimensions n × 1. The product $Ureduce^T \cdot x^i$ will have dimensions k x 1.

To summarize, the whole algorithm in octave is roughly:

```
Sigma = (1/m) * X' * X;     % compute the covariance matrix
[U,S,V] = svd(Sigma);       % compute our projected directions
Ureduce = U(:,1:k);         % take the first k directions
Z = X * Ureduce;            % compute the projected data points
```

Reconstruction from Compressed Representation

If we use PCA to compress our data, how can we uncompress our data, or go back to our original number of features?

To go from 1–dimension back to 2d we do: $z \in \mathbb{R} \rightarrow x \in \mathbb{R}^2$

We can do this with the equation: $x_{approx}^1 = U_{reduce} \cdot z^1$

Note that we can only get approximations of our original data.

Choosing the Number of Principal Components

How do we choose k, also called the *number of principal components*? Recall that k is the dimension we are reducing to.

One way to choose k is by using the following formula:

- Given the average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}\left||x^i - x_{approx}^i\right||^2$
- Also given the total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}\left||x^i\right||^2$
- Choose k to be the smallest value such that: $\frac{\frac{1}{m}\sum_{i=1}^{m}\left||x^i - x_{approx}^i\right||^2}{\frac{1}{m}\sum_{i=1}^{m}\left||x^i\right||^2} \leq 0.01$

In other words, the squared projection error divided by the total variation should be less than one percent, so that **99% of the variance is retained**.

Algorithm for choosing k

- Try PCA with k=1,2,…
- Compute U_{reduce}, z, x
- Check the formula given above that 99% of the variance is retained. If not, go to step one and increase k.

This procedure would actually be horribly inefficient. In Octave, we will call svd:

```
[U,S,V] = svd(Sigma)
```

Which gives us a matrix S. We can actually check for 99% of retained variance using the S matrix as follows:

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

Advice for Applying PCA

- Compressions

Reduce space of data

Speed up algorithm

- Visualization of data

Choose k = 2 or k = 3

Don't assume you need to do PCA. **Try your full machine learning algorithm without PCA first.** Then use PCA if you find that you need it.