1. LATENT SEMANTIC INDEXING

Monday, March 14, 2016:

1.1. **Intuition.** Objective is to classify bodies of text.

**Idea:** Form a matrix $B$ that records the incidence of words (“terms”) in your database. Then every term frequency - $t_{f_{ij}}$ of that matrix will be the number of times term $i$ appears in document $j$. Typically, we would filter out the terms that don’t carry much meaning.

1.2. **Method.** From $B$, we can construct a matrix $A$ defined by:

$$A_{ij} = \log(t_{f_{ij}} + 1)(1 + \frac{\sum_{j=1}^{n} p_{ij} \log p_{ij}}{\log(n)})$$

where: $p_{ij} = \frac{t_{f_{ij}}}{g_{fi}}$, $g_{fi} = \sum_{j=1}^{n} t_{f_{ij}} = 1$ for any $j$.

Perform SVD on $A$:

$$A \approx T S D^*.$$  
$m \times n$  
$m \times k$  
k \times k  
k \times n$

where, $T$ - term concept matrix, $D$ - concept-document matrix

1.3. **Non-linear techniques.** Often, the relationship between the data is nonlinear. Approach to resolve such problem is *Kernel PCA*:

**Kernel PCA:**

We construct a non-linear transformation on the given data $X$ such that PCA algorithm can pick out clusters. Problem with such algorithm is that it’s hard to come up with a non-linear map that solved the problem of non-linearity.

**Example:** In the following example, we are given a data that forms two circles with different radius. A kernel trick that could easily distinguish the two datasets is to add a third dimension that is formed by $x^2 + y^2$ from existing dimensions of $x$ and $y$. We see the result of such kernel map on the image.

2. DIFFUSION MAPS

2.1. **Markov Chains review.** Use distance between 2 points as a probability of jump in Markov chain matrix.

Markov Chains give us a way to estimate what event will be likely after arbitrary number of steps that the systems will take. For more intuition, let’s consider an example.

**Stock Market:**
Say we have 3 states:

1. Bull
2. Bear
3. Stagnant

Given $P$ - probability transition matrix, and $S_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ - initial distribution:

$$P = \begin{bmatrix} 0.9 & 0.15 & 0.25 \\ 0.075 & 0.8 & 0.25 \\ 0.025 & 0.05 & 0.5 \end{bmatrix}.$$
We can estimate that after 1 step, our distribution of each state will be as following:

\[
\mathbf{s}_1 = \mathbf{P}\mathbf{s}_0 = \begin{bmatrix} 0.9 \\ 0.075 \\ 0.025 \end{bmatrix}
\]

After infinitely many steps, in this example we would get a stationary distribution of. We can argue that a Markov Chain admits SVD:

\[
\mathbf{P} = \mathbf{V}\Lambda\mathbf{V}^{-1}
\]

where:

\[
\Lambda = \begin{bmatrix}
\lambda_1 = 1 & 0 & \cdots & 0 \\
0 & \lambda_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_n
\end{bmatrix}
\]

Then,

\[
\mathbf{P}^k = \mathbf{V}\lambda^k\mathbf{V}^{-1} = \mathbf{V} \begin{bmatrix}
\lambda_1^k & 0 & \cdots & 0 \\
0 & \lambda_2^k & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_n^k
\end{bmatrix} \mathbf{V}^{-1} \to \mathbf{V}^{-1} \begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix} \mathbf{V}^{-1}
\]

In our example:

\[
\mathbf{P}^k = \begin{bmatrix}
0.625 & 0.625 & 0.625 \\
0.3125 & 0.3125 & 0.3125 \\
0.0625 & 0.0625 & 0.0625
\end{bmatrix}.
\]
We see that we lost any knowledge of where we started, because all columns converged to the same vector.