Lecture 6
Curse of Dimensionality
PCA
Outline

- Curse of Dimensionality
- Dimensionality reduction with PCA
Curse of Dimensionality

- Problems of high dimensional data, “the curse of dimensionality”
  - running time
  - overfitting
  - number of samples required
- Dimensionality Reduction Methods
  - Principle Component Analysis
Curse of Dimensionality: Complexity

- Complexity (running time) increases with dimension $d$

- A lot of methods have at least $O(nd^2)$ complexity, where $n$ is the number of samples

- For example if we need to estimate covariance matrix

- So as $d$ becomes large, $O(nd^2)$ complexity may be too costly
Curse of Dimensionality: Number of Samples

- Suppose we want to use the nearest neighbor approach with $k = 1$ (1NN)
  - Suppose we start with only one feature
    
    ![Diagram showing 0 and 1 features]
    
  - This feature is not discriminative, i.e. it does not separate the classes well
  - We decide to use 2 features. For the 1NN method to work well, need a lot of samples, i.e. samples have to be dense
  - To maintain the same density as in 1D (9 samples per unit length), how many samples do we need?
Curse of Dimensionality: Number of Samples

- We need $9^2$ samples to maintain the same density as in 1D.
Of course, when we go from 1 feature to 2, no one gives us more samples, we still have 9

This is way too sparse for 1NN to work well
Things go from bad to worse if we decide to use 3 features:

If 9 was dense enough in 1D, in 3D we need $9^3=729$ samples!
Curse of Dimensionality: Number of Samples

- In general, if \( n \) samples is dense enough in 1D
- Then in \( d \) dimensions we need \( n^d \) samples!
- And \( n^d \) grows really really fast as a function of \( d \)
- Common pitfall:
  - If we can’t solve a problem with a few features, adding more features seems like a good idea
  - However the number of samples usually stays the same
  - The method with more features is likely to perform worse instead of expected better
We should try to avoid creating lot of features
- Often no choice, problem starts with many features
- Example: Face Detection
  - One sample point is $k$ by $m$ array of pixels
  - Feature extraction is not trivial
  - Say pixel intensities are taken as a feature
  - Typical dimension is 20 by 20 = 400
  - Suppose **10** samples are dense enough for 1 dimension. Need only $10^{400}$ samples
The Curse of Dimensionality

- Face Detection, dimension of one sample point is $km$

  $\begin{bmatrix}
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
\end{bmatrix}

- The fact that we set up the problem with $km$ dimensions (features) does not mean it is really a $km$-dimensional problem.
- Space of all $k$ by $m$ images has $km$ dimensions.
- Space of all $k$ by $m$ faces must be much smaller, since faces form a tiny fraction of all possible images.
- Most likely we are not setting the problem up with the right features.
- If we used better features, we are likely need much less than $km$-dimensions.
Dimensionality Reduction

- High dimensionality is challenging and redundant
- It is natural to try to reduce dimensionality
- Reduce dimensionality by feature combination: combine old features $x$ to create new features $y$

$$
x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \rightarrow f \left( \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \right) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = y \quad \text{with } k < d
$$

- For example,

$$
x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \end{bmatrix} = y
$$

- Ideally, the new vector $y$ should retain from $x$ all information important for classification
Dimensionality Reduction

- The best $f(x)$ is most likely a non-linear function
- Linear functions are easier to find though
- For now, assume that $f(x)$ is a linear mapping
- Thus it can be represented by a matrix $W$:

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_d 
\end{bmatrix} \Rightarrow W \begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_d 
\end{bmatrix} = \begin{bmatrix}
  w_{11} & \cdots & w_{1d} \\
  \vdots & \ddots & \vdots \\
  w_{k1} & \cdots & w_{kd} 
\end{bmatrix} \begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_d 
\end{bmatrix} = \begin{bmatrix}
  y_1 \\
  \vdots \\
  y_k 
\end{bmatrix} \text{ with } k < d
\]
**Principle Component Analysis (PCA)**

- **Main idea:** seek most accurate data representation in a lower dimensional space

- **Example in 2-D**
  - Project data to 1-D subspace (a line) which minimize the projection error

- Notice that the the good line to use for projection lies in the direction of largest variance
PCA

- After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector $y$

- Note that new data $y$ has the same variance as old data $x$ in the direction of the green line

- PCA preserves largest variances in the data
PCA: Approximation of Elliptical Cloud in 3D

**best 2D approximation**

**best 1D approximation**
PCA

- What is the direction of largest variance in data?
- Recall that if $x$ has multivariate distribution $N(\mu, \Sigma)$, direction of largest variance is given by eigenvector corresponding to the largest eigenvalue of $\Sigma$

- This is a hint that we should be looking at the covariance matrix of the data (note that PCA can be applied to distributions other than Gaussian)
PCA: Linear Algebra Review

- Let $V$ be a $d$ dimensional linear space, and $W$ be a $k$ dimensional linear subspace of $V$
- We can always find a set of $d$ dimensional vectors \{$e_1, e_2, \ldots, e_k$\} which forms an orthonormal basis for $W$
  - $\langle e_i, e_j \rangle = 0$ if $i$ is not equal to $j$ and $\langle e_i, e_i \rangle = 1$
- Thus any vector in $W$ can be written as
  \[
  \alpha_1 e_1 + \alpha_2 e_2 + \ldots + \alpha_k e_k = \sum_{i=1}^{k} \alpha_i e_i \quad \text{for scalars } \alpha_1, \ldots, \alpha_k
  \]

Let $V = \mathbb{R}^2$ and $W$ be the line $x-2y=0$. Then the orthonormal basis for $W$ is
\[
\begin{bmatrix}
  2 / \sqrt{5} \\
  1 / \sqrt{5}
\end{bmatrix}
\]
PCA: Linear Algebra

- Recall that subspace $W$ contains the zero vector, i.e. it goes through the origin.

- It is convenient to project to subspace $W$: thus we need to shift everything.
PCA Derivation: Shift by the Mean Vector

- Before PCA, subtract sample mean from the data
  \[ x - \frac{1}{n} \sum_{i=1}^{n} x_i = x - \hat{\mu} \]

- The new data has zero mean: \( E(X - E(X)) = E(X) - E(X) = 0 \)

- All we did is change the coordinate system

- Another way to look at it:
  - first step of getting \( y \) is to subtract the mean of \( x \)
  \[ x \rightarrow y = f(x) = g(x - \hat{\mu}) \]
PCA: Derivation

- We want to find the most accurate representation of data \(D = \{x_1, x_2, \ldots, x_n\}\) in some subspace \(W\) which has dimension \(k < d\).
- Let \(\{e_1, e_2, \ldots, e_k\}\) be the orthonormal basis for \(W\). Any vector in \(W\) can be written as \(\sum_{i=1}^{k} \alpha_i e_i\).
- Thus \(x_1\) will be represented by some vector in \(W\) \(\sum_{i=1}^{k} \alpha_1_i e_i\).
- Error this representation:

\[
\text{error} = \left\| x_1 - \sum_{i=1}^{k} \alpha_1_i e_i \right\|^2
\]
PCA: Derivation

- To find the total error, we need to sum over all $x_j$'s.
- Any $x_j$ can be written as $\sum_{i=1}^{k} \alpha_{ji}e_i$.
- Thus the total error for representation of all data $D$ is:

$$J(e_1, ..., e_k, \alpha_{11}, ..., \alpha_{nk}) = \sum_{j=1}^{n} \left| x_j - \sum_{i=1}^{k} \alpha_{ji}e_i \right|^2$$

unknowns | error at one point
sum over all data points
PCA: Derivation

- A lot of math……to finally get:

- Let $S$ be the scatter matrix, it is just $n-1$ times the sample covariance matrix

\[
\hat{\Sigma} = \frac{1}{n-1} \sum_{j=1}^{n} (x_j - \hat{\mu})(x_j - \hat{\mu})^t
\]

- To minimize $J$ take for the basis of $W$ the $k$ eigenvectors of $S$ corresponding to the $k$ largest eigenvalues
PCA

- The larger the eigenvalue of $S$, the larger is the variance in the direction of corresponding eigenvector.

- This result is exactly what we expected: project $\mathbf{x}$ into subspace of dimension $k$ which has the largest variance.

- This is very intuitive: restrict attention to directions where the scatter is the greatest.

\[ \lambda_1 = 30 \]
\[ \lambda_2 = 0.8 \]
Thus PCA can be thought of as finding new orthogonal basis by rotating the old axis until the directions of maximum variance are found.
PCA as Data Approximation

- Let \( \{e_1, e_2, \ldots, e_d\} \) be all \( d \) eigenvectors of the scatter matrix \( S \), sorted in order of decreasing corresponding eigenvalue.

- Without any approximation, for any sample \( x_i \):

\[
x_i = \sum_{j=1}^{d} \alpha_j e_j = \alpha_1 e_1 + \ldots + \alpha_k e_k + \alpha_{k+1} e_{k+1} + \ldots + \alpha_d e_d
\]

- Coefficients \( \alpha_m = x_i^t e_m \) are called *principle components*.
  - The larger \( k \), the better is the approximation.
  - Components are arranged in order of importance, more important components come first.

- Thus PCA takes the first \( k \) most important components of \( x_i \) as an approximation to \( x_i \).
PCA: Last Step

- Now we know how to project the data
- Last step is to change the coordinates to get final $k$-dimensional vector $y$

Let matrix $E = [e_1 \cdots e_k]$

Then the coordinate transformation is $y = E^t x$

Under $E^t$, the eigenvectors become the standard basis:

$$E^t e_i = \begin{bmatrix} e_1 \\ \vdots \\ e_i \\ \vdots \\ e_k \end{bmatrix} e_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$
Recipe for Dimension Reduction with PCA

Data \( D = \{ x_1, x_2, \ldots, x_n \} \). Each \( x_i \) is a \( d \)-dimensional vector. Wish to use PCA to reduce dimension to \( k \)

1. Find the sample mean \( \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \)
2. Subtract sample mean from the data \( z_i = x_i - \hat{\mu} \)
3. Compute the scatter matrix \( S = \sum_{i=1}^{n} z_i z_i^t \)
4. Compute eigenvectors \( e_1, e_2, \ldots, e_k \) corresponding to the \( k \) largest eigenvalues of \( S \)
5. Let \( e_1, e_2, \ldots, e_k \) be the columns of matrix \( E = [e_1 \cdots e_k] \)
6. The desired \( y \) which is the closest approximation to \( x \) is \( y = E^t z \)
Drawbacks of PCA

- PCA was designed for accurate *data representation*, not for *data classification*
- Preserves as much variance in data as possible
- If directions of maximum variance is important for classification, will work

- However, the directions of maximum variance may be useless for classification

*apply PCA to each class*