DIMENSION REDUCTION: FEATURE SELECTION VS FEATURE EXTRACTION

Guoqing Tang
Department of Mathematics & Statistics
North Carolina A&T State University
February 18, 2021
Introduction

Feature Selection Methods
- **Filter**: extract features from the data without any learning involved
- **Wrapper**: use learning techniques to evaluate which features are useful
- **Embedded**: combine the feature selection step and the classifier construction

Feature Extraction Methods
- Supervised PCA and PCR
- Partial least squares
- LDA
- General low rank models (GLRM)
- Isometric Mapping (Isomap)
In machine learning as the dimensionality of the data rises, the amount of data required to provide a reliable analysis grows exponentially.

Bellman referred to this phenomenon as the “curse of dimensionality” when considering problems in dynamic optimization.

In practice, the curse of dimensionality means that for a given sample size, there is a maximum number of features above which the performance of a classifier will degrade rather than improve.

A popular approach to this problem of high-dimensional datasets, known as dimensionality reduction, is to search for a projection of the data onto a smaller number of features which preserves the information as much as possible.
The dimensionality reduction can be made in two different ways:

- by only keeping the most relevant features from the original dataset (this technique is called **feature selection**) or
- by exploiting the redundancy of the input data and by finding a smaller set of new features, each being a combination of the original features, containing basically the same information as the original features (this technique is called **feature extraction**).

**Feature selection:** In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:

- **Filter:** extract features from the data without any learning involved
- **Wrapper:** use learning techniques to evaluate which features are useful
- **Embedded:** combine the feature selection step and the classifier construction
Feature extraction: This reduces the data in a high dimensional space to a lower dimension space, i.e., a space with lesser no. of dimensions.

One objective for both feature subset selection and feature extraction methods is to avoid overfitting the data in order to make further analysis possible.
Feature Selection

- Feature selection works by removing features that are not relevant or are redundant.

- The size of the data to be processed has increased exponentially in the past few years and therefore feature selection has become a requirement before any kind of regression or classification takes place.

- Unlike feature extraction methods, feature selection techniques do not alter the original representation of the data.

- Mathematically speaking, given a set of features \( F = \{f_1, f_2, ..., f_p\} \), the feature selection problem is to find a subset \( F^* = \{f_{i_1}, f_{i_2}, ..., f_{i_k}\} \), where \( k < p \), that "maximizes the learner’s ability to classify patterns."
Implementing Feature Selection

- In theory, the goal of feature selection is to find an optimal feature-subset (one that maximizes the scoring function).

- In real world applications this is usually not possible.
  - For most problems it is computationally intractable to search the whole space of possible feature subsets
  - One usually has to settle for approximations of the optimal subset

- Feature Selection is necessary in a number of situations
  - Features may be expensive to obtain
  - Want to extract meaningful rules from your classifier
  - When you transform or project, measurement units (length, weight, etc.) are lost
  - Features may not be numeric (e.g. strings)
Feature selection requires
- A search strategy to select candidate feature subset
- An objective function to evaluate these candidate features

Search strategy: Exhaustive evaluation of feature subsets involves \( \binom{p}{k} \) combinations for a fixed value of \( k \), and if \( 2^p \) if \( k \) must be optimized as well.
- This number of combination is unfeasible, even for moderate values of \( p \) and \( k \), so a search procedure must be used in practice
- For example, exhaustive evaluation of 10 out of 20 features involves 184,756 feature subsets

A search strategy is therefore needed to direct the FS process as it explores the space of all possible combinations of features.
The objective function evaluates candidate subsets and returns a measure of their “goodness”.

This feedback is used by the search strategy to select new candidates.

Simple Objective function: Cross-validation error rate.
Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance.

These methods are faster and less computationally expensive than wrapper methods.

When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

Some of filter methods:
» Correlation coefficient
» Information gain
» Chi-square
» Variance threshold
» Mean absolute difference (MAD)
One of the simplest criteria is the Pearson’s correlation coefficient defined by

$$R(i) = \frac{\text{Cov}(X_i, Y)}{\sqrt{\text{Var}(X_i)} \sqrt{\text{Var}(Y)}}$$

where $X_i$ is the $i$th feature, $Y$ is the response.

- We check the absolute value of the Pearson’s correlation between the target and numerical features in our dataset.
- We keep the top $k$ features based on this criterion.
- Correlation ranking can only detect linear dependencies between features and target.
**Information gain (IG)** for a given set $S$ of observed $Y$ values for a targeted feature $X$ is calculated:

$$IG(Y, X) = H(Y) - H(Y \mid X),$$

where $H(Y)$ is the **entropy** for $Y$, and $H(Y \mid X)$ is the **conditional entropy** for $Y$ given the variable $X$.

IG can be used to judge how important a given categorical variable $X$ is in explaining the binary $Y$ variable.

The conditional entropy can be calculated by splitting the dataset into groups for each observed value of $x$:

$$H(S \mid X) = \sum_{x \in \text{value}(X)} \frac{|S(X=x)|}{|S|} H(S(X=x)) = \sum_{x} p(x)[-\sum_{y} p(y \mid x) \log_2 (p(y \mid x))],$$

where $S(X=x) = \{y \text{ in } S \mid X = x\}$, and $H(S(X=x))$ is the entropy of group of samples in $S$ corresponding to $X = x$. 
The following is what the *quantum* of Information Gain means:

- Less than 0.02, then the predictor is not useful for modeling $Y$.
- 0.02 to 0.1, then the predictor has only a weak relationship with $Y$.
- 0.1 to 0.3, then the predictor has a medium strength relationship with $Y$.
- 0.3 or higher, then the predictor has a strong relationship with $Y$. 
Let’s do an example to make this clear. In the below mini-dataset, the label we’re trying to predict is the type of fruit. This is based off the size, color, and shape variables.

<table>
<thead>
<tr>
<th>Class</th>
<th>Suspicious Words</th>
<th>Unknown Sender</th>
<th>Contain Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>span</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>span</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>span</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>ham</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>ham</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>ham</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

\[
H(\text{class}) = - (P(\text{class} = \text{span}) \times \log_2(P(\text{class} = \text{span})) + P(\text{class} = \text{ham}) \times \log_2(P(\text{class} = \text{ham}))) = -(3/6 \times \log_2(3/6) + 3/6 \times \log_2(3/6)) = - \log_2(1/2) = 1.
\]
To compute $IG(class, \text{words})$, we need to compute $H(class \mid \text{words})$:

$$H(class \mid (\text{words=true}), \text{words}) = -(3/4\log_2(3/4) + 1/4\log_2(1/4)) = 0.8113$$

$$H(class \mid (\text{words=false}), \text{words}) = -2/2\log_2(2/2) = 0,$$ and

$$H(class \mid \text{words}) = 4/6 \times H(class \mid (\text{words=true}), \text{words}) +$$

$$2/6 \times H(class \mid (\text{words=false}), \text{words})$$

$$= 2/3 \times 0.8113 + 1/3 \times 0 = 0.5409.$$

Hence $IG(class,\text{words}) = H(class) - H(class \mid \text{words}) = 1 - 0.5409 = 0.4591$. 
Similarly, to compute $IG(\text{class, sender})$, we need to compute $H(\text{class | sender})$:

\[
H(\text{class | (sender=true), sender}) = -(2/3\log_2(2/3) + 1/3\log_2(1/3)) = 0.9183,
\]

\[
H(\text{class | (sender=false), sender}) = -(1/3\log_2(1/3) + 2/3\log_2(2/3)) = 0.9183, \text{ and}
\]

\[
H(\text{class | sender}) = 3/6 \cdot H(\text{class | (sender=true), sender}) + 3/6 \cdot H(\text{class | (sender=false), sender})
\]

\[
= 1/2 \cdot 0.9183 + 1/2 \cdot 0.9183 = 0.9183.
\]

Hence $IG(\text{class, sender}) = H(\text{class}) - H(\text{class | sender}) = 1 - 0.9183 = 0.0817$.
Finally, to compute $IG(\text{class, image})$, we need to compute $H(\text{class | image})$:

$$H(\text{class |(image=true), sender}) = -(1/3 \cdot \log_2(1/3) + 2/3 \cdot \log_2(2/3)) = 0.9183,$$

$$H(\text{class |(image=false), sender}) = -(2/3 \cdot \log_2(2/3) + 1/3 \cdot \log_2(1/3)) = 0.9183,$$

and

$$H(\text{class |image}) = 3/6 \cdot H(\text{class |(image=true), sender}) + 3/6 \cdot H(\text{class |(image=false), sender})$$

$$= 1/2 \cdot 0.9183 + 1/2 \cdot 0.9183 = 0.9183.$$

Hence $IG(\text{class, image}) = H(\text{class}) - H(\text{class | image}) = 1 - 0.9183 = 0.0817$.

The information gain of the Suspicious Words feature is 0.4591, which suggests that this feature has a strong relationship to the response Class.

Both the features Unknown Sender and Contain Image has the IG of 0.0817, indicating both features have very weak correlations with the response Class, and they are not very helpful in explaining the response Class.
In this method, we calculate Chi-square between each feature and the target and select the desired number of features with the best Chi-square scores.

In order to correctly apply the chi-squared to test the relation between various features in the dataset and the target variable, the following conditions have to be met: the variables have to be categorical, sampled independently and values should have an expected frequency greater than 5.

This is a non-parametric test. We typically use it to find how the observed value of a given event is significantly different from the expected value.

We arrange the data by using the contingency table consists of both the observed frequencies $O_i$ and expected frequencies $E_i$. 
We next compute the value of the chi-square statistic

$$\chi^2_{df} = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

where $df = (k - 1)$ is the degree of freedom, $O_i$'s are the observed frequencies from the sample and $E_i$'s are the expected frequencies computed by allocating the sample to the response categories according to the distribution specified in $H_0: p_1 = p_{10}, p_2 = p_{20}, \ldots, p_k = p_{k0}$. That is, $E_i = n \times p_{i0}$.

Once we have calculated the chi-square value, the next task is to compare it with the critical chi-square value from the chi-square table against the degrees of freedom and the level of significance.

Let us work out a simple example to understand the Chi-square method.
An organization claims that the experience of the employees of different departments is distributed in the following categories:

- 11 – 20 Years = 20%
- 21 – 40 Years = 17%
- 6 – 10 Years = 41% and
- Up to 5 Years = 22%

A random sample of 1470 employees is collected. Does this random sample provide evidence against the organization’s claim?

Setting up the hypothesis:

- **Null hypothesis**: The true proportions of the experience of the employees of different departments are distributed in the following categories: 11 – 20 Years = 20%, 21 – 40 Years = 17%, 6 – 10 Years = 41% and up to 5 Years = 22%
- **Alternative hypothesis**: The distribution of experience of the employees of different departments differs from what the organization states
Use the data to compute the observed frequencies $O = [340, 207, 607, 316]$.

- 11 – 20 Years = $\frac{340}{1470} = 23.12925\%$
- 21 – 40 Years = $\frac{207}{1470} = 14.08163\%$
- 6 – 10 Years = $\frac{607}{1470} = 41.29252\%$, and
- Up to 5 Years = $\frac{316}{1470} = 21.49660\%$

The expected frequencies are: $E = 1470 \times [0.2, 0.17, 0.41, 0.22] = [294, 259.9, 602.7, 323.4]$

Calculate the chi-square value: $\chi^2 = \sum_{i=1}^{4} \frac{(O_i - E_i)^2}{E_i} = 14.7619$, $df = 3$, and $p$-value = 0.002032

The $p$-value here is less than 0.05. Therefore, we will reject our null hypothesis. Hence, the distribution of experience of the employees of different departments differs from what the organization states.
Wrapper methods use the predictor as a black box and the predictor performance as the objective function to evaluate the variable subset.

Wrappers tend to perform better in selecting features since they take the model hypothesis into account by training and testing in the feature space.

Wrapper methods can be divided into **Sequential Selection Algorithms** and **Heuristic Search Algorithms**.

The sequential selection algorithms start with an empty set (full set) and add features (remove features) until the maximum objective function is obtained.

To speed up the selection, a criterion is chosen which incrementally increases the objective function until the maximum is reached with the minimum number of features.
The heuristic search algorithms evaluate different subsets to optimize the objective function.

Different subsets are generated either by searching around in a search space or by generating solutions to the optimization problem.

First we will look at sequential selection algorithms.

The **Sequential Feature Selection** (SFS) algorithm starts with an empty set and adds one feature for the first step which gives the highest value for the objective function.

From the second step onwards the remaining features are added individually to the current subset and the new subset is evaluated.
The individual feature is permanently included in the subset if it gives the maximum classification accuracy.

The process is repeated until the required number of features are added. This is a naive SFS algorithm since the dependency between the features is not accounted for.

A *Sequential Backward Selection* (SBS) algorithm can also be constructed but the algorithm starts from the complete set of variables and removes one feature at a time whose removal gives the lowest decrease in predictor performance.

The *Sequential Floating Forward Selection* (SFFS) algorithm is more flexible than the naive SFS because it introduces an additional backtracking step.
The basic flowchart is given in the figure on right where \( k \) is the current subset size and \( d \) is the required dimension.

The first step of the algorithm is the same as the SFS algorithm which adds one feature at a time based on the objective function.

The SFFS algorithm adds another step which excludes one feature at a time from the subset obtained in the first step and evaluates the new subsets.
If excluding a feature increases the value of the objective function, then that feature is removed and goes back to the first step with the new reduced subset or else the algorithm is repeated from the top.

This process is repeated until the required number of features are added or required performance is reached.

The SFS and SFFS methods suffer from producing nested subsets since the forward inclusion was always unconditional which means that two highly correlated variables might be included if it gave the highest performance in the SFS evaluation.

To avoid the nesting effect, adaptive version of the SFFS was developed.
The Adaptive Sequential Forward Floating Selection (ASFFS) algorithm used a parameter $r$ which would specify the number of features to be added in the inclusion phase which was calculated adaptively.

The parameter $o$ would be used in the exclusion phase to remove maximum number of features if it increased the performance.

The ASFFS attempted to obtain a less redundant subset than the SFFS algorithm. It can be noted that a statistical distance measure can also be used as the objective function for the search algorithms.

Theoretically, the ASFFS should produce a better subset than SFFS but this is dependent on the objective function and the distribution of the data.
The Plus-$L$-Minus-$r$ search method also tries to avoid nesting.

In the Plus-$L$-Minus-$r$ search, in each cycle $L$ variables were added and $r$ variables were removed until the desired subset was achieved.

The parameters $L$ and $r$ have to be chosen arbitrarily.
- **Genetic Algorithm** (GA) can be used to find the subset of features wherein the chromosome bits represent if the feature is included or not.
- The global maximum for the objective function can be found which gives the best suboptimal subset. Here again the objective function is the predictor performance.
- The GA parameters and operators can be modified within the general idea of an evolutionary algorithm to suit the data or the application to obtain the best performance or the best search result.
- A modified version of the GA called the **CHCGA** can be used for feature selection.
The CHCGA is a non-traditional GA which differs from GA in the following ways:

» The best \( N \) individuals are chosen from the pool of parents and offspring i.e. better offspring replaces lesser fit parents.

» A highly disruptive *half uniform crossover* (HUX) operator is used which crosses over exactly half of the non-matching alleles, wherein the bits to be crossed over are selected at random.

» During reproduction step, each member of the parent population is randomly selected without replacement and paired for mating. Not all the pairs are crossed over but before mating the Hamming distance between the parents are calculated and if half this distance does not exceed a threshold \( d \), they are not mated. The threshold is usually initialized to \( L=4 \) where \( L \) is the chromosome length. If no offspring is obtained in the generation, the threshold is decremented by one. Due to these mating criteria of mating only diverse parents, the population converges as the threshold decreases.
If there are no offspring generated and the threshold drops to zero, a cataclysmic mutation is introduced to create a new population. The best individual in the current parent population is taken as the template to create the new population. The rest $N-1$ individuals are obtained by randomly flipping a percentage (35–40%) of bits of the template. The regular mutation after crossover step is skipped each time and the above mentioned mutation is carried if required.

- The CHCGA converges on the solution faster and provides a more effective search by maintaining the diversity and avoiding stagnation of the population.
The main drawback of wrapper methods is the number of computations required to obtain the feature subset.

For each subset evaluation, the predictor creates a new model, i.e., the predictor is trained for each subset and tested to obtain the classifier accuracy.

If the number of samples is large, most of the algorithm execution is spent in training the predictor.

In some algorithms such as GA feature selection, the same feature subset might be evaluated multiple times since the classifier accuracies for evaluated subsets are not stored for future retrieval.
Limitations of Wrapper Methods

- Another drawback of using the classifier performance as the objective function is that the classifiers are prone to overfitting.
- Overfitting occurs if the classifier model learns the data too well and provides poor generalization capability.
- The classifier can introduce bias and increases the classification error.
- Using classification accuracy in subset selection can result in a bad feature subset with high accuracy but poor generalization power.
- To avoid this, a separate holdout test set can be used to guide the prediction accuracy of the search.
Embedded methods aim to reduce the computation time taken up for reclassifying different subsets which is done in wrapper methods.

The main approach is to incorporate the feature selection as part of the training process.

In Filters methods we mentioned that IG is an important concept but the ranking using IG yielded poor results since the IG between each of the features and the response only was considered.

A greedy search algorithm is proposed to evaluate the subsets.

The objective function is designed such that choosing a feature will maximize the IG between the feature and the response while the IG between the selected feature and the subset of the so far selected features is a minimum.
This is formulated as follows:

\[ IG(Y, X) - \beta \sum_{s \in S} IG(X, s) \]  

where \( Y \) is the response, \( X \) is the current selected feature, \( s \) is the feature in the already selected subset \( S \), and \( \beta \) controls the importance of the IG between the current feature \( X \) and the features in the subset \( S \).

Eq. (1) will select better subset since the inter-feature IG is used in the calculation to select the nonredundant features.
Max-Relevancy Min-Redundancy

- The mRMR (max-relevancy, min-redundancy) is another method based on IG.
- It uses similar criteria as in Eq. (1) given as:

\[ IG(x_j, C) - \frac{1}{m} \sum_{x_j \in S_{m-1}} IG(x_j, x_l) \]  

where \( x_j \) is the \( m \)th feature in the subset \( S \) and the set \( S_{m-1} \) is the so far selected subset with \( m-1 \) features.
- Instead of a greedy algorithm a two-stage approach is implemented.
- First the criterion (2) is used to select a number \( k \) which is the optimal number of features which gives the lowest cross-validation classification error.
- In the second stage, wrapper methods are used to evaluate different subsets of size \( k \) or direct evaluations are done on different subsets to find the subset which consistently yields the smallest classification error.
Another method used in literature is to use the weights of a classifier to rank features for their removal.

Let \( w_j \) be defined as:

\[
    w_j = \frac{\mu_j(+) - \mu_j(-)}{\sigma_j(+) + \sigma_j(-)}
\]

(3)

where \( \mu_j(+) \) and \( \mu_j(-) \) are the means of the sample in class (+) and (-), and \( \sigma_j(+) \) and \( \sigma_j(-) \) are the variances of the respective classes, and \( j = 1, \ldots, D \).

Eq. (3) can be used as a ranking criteria to sort the features.

The rank vector \( w \) can be used to classify since features rank proportionally contributes to the correlation.
A voting scheme given as:

\[ D(x) = w(x - \mu) \quad (4) \]

where \( w \) is the rank of the features or weight, \( D(x) \) is the decision and \( mu \) is the mean of the \( x \) data.

Hence the weights (rank) of the features can be used as classifier weights.

Feature selection can be done by looking at the change in the weight \( w_j \) to determine removing a feature \( X_j \).

It is also suggested to use the change in the objective function, a linear discriminant function \( J \) which is a function of \( w_j \).
Choosing the Optimal Model

- The model containing all of the predictors will always have the smallest RSS and the largest $R^2$, since these quantities are related to the training error.
- We wish to choose a model with low test error, not a model with low training error. Recall that training error is usually a poor estimate of test error.
- Therefore, RSS and $R^2$ are not suitable for selecting the best model among a collection of models with different numbers of predictors.
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Supervd</th>
<th>Linear</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t-test feature selection</td>
<td>Filter</td>
<td>Yes</td>
<td></td>
<td>It finds features with a maximal difference of mean value btw groups and a minimal variability within each group</td>
</tr>
<tr>
<td>Correlation-based feature selection</td>
<td>Filter</td>
<td>Yes</td>
<td></td>
<td>It finds features that are highly correlated with the class but are uncorrelated with each other</td>
</tr>
<tr>
<td>Bayesian networks</td>
<td>Filter</td>
<td>Yes</td>
<td>No</td>
<td>They determine the causal relationships among features and remove the ones that do not have any causal relationship with the class</td>
</tr>
<tr>
<td>Information gain</td>
<td>Filter</td>
<td>No</td>
<td>Yes</td>
<td>It measures how common a feature is in a class compared to all other classes</td>
</tr>
<tr>
<td>Genetic algorithms</td>
<td>Wrapper</td>
<td>Yes</td>
<td>No</td>
<td>They find the smaller set of features for which the optimization criterion (classification accuracy) does not deteriorate</td>
</tr>
<tr>
<td>Sequential search</td>
<td>Wrapper</td>
<td></td>
<td></td>
<td>Heuristic base search algorithm that finds the features with the highest criterion value (classification accuracy) by adding one new feature to the set every time</td>
</tr>
<tr>
<td>SVM w/ recursive feature elimination</td>
<td>Embedded</td>
<td>Yes</td>
<td>Yes</td>
<td>It constructs the SVM classifier and eliminates the features based on their “weight” when constructing the classifier</td>
</tr>
<tr>
<td>Random forests</td>
<td>Embedded</td>
<td>Yes</td>
<td>Yes</td>
<td>They create a number of decision trees using different samples of the original data and use different averaging algorithms to improve accuracy</td>
</tr>
<tr>
<td>LASSO</td>
<td>Embedded</td>
<td>Yes</td>
<td>Yes</td>
<td>It constructs a linear model that sets many of the feature coefficients to zero and uses the nonzero ones as the selected features.</td>
</tr>
</tbody>
</table>
Feature extraction techniques are used to extract a subset of new features from the original feature set by means of some functional mapping by keeping as much information in the data as possible.

Feature extraction creates new variables as combinations of others to reduce the dimensionality of the selected features.

The problem of feature extraction can be stated as:

» Give a feature space \( x_j \in \mathbb{R}^n \), find a mapping \( f : x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^m \) with \( m < n \) such that the transformed feature \( z = f(x) \) preserves most of the information or structure in \( \mathbb{R}^m \).

» An optimal mapping \( z = f(x) \) will be one that results in no increase in the minimum probability of error.

There are two broad categories for feature extraction algorithms: linear and nonlinear.
Linear feature extraction assumes that the data lies on a lower-dimensional linear subspace.

It projects them on this subspace using matrix factorization.

Given a dataset $X$ of size $n \times p$, find a project matrix $U$ of size $p \times K$ and a projection $Z$ of $n \times K$ with $K < p$ such that $Z = XU$.

If we choose $U$ to be orthogonal, i.e., $UU^T = I$, then we have $X = ZU^T$. 
The most well-known dimensionality reduction algorithm is principal component analysis (PCA).

Using the covariance matrix and its eigenvalues and eigenvectors, PCA finds the “principal components” in the data which are uncorrelated eigenvectors each representing some proportion of variance in the data.

PCA and many variations of it have been applied as a way of reducing the dimensionality of the data.

It has been argued that when computing the principal components (PCs) of a dataset there is no guarantee that the PCs will be related to the class variable.

A supervised version of the principal component analysis (Bair et. al., 2006) selects the PCs based on the class variables, and the SPCA performs better than the unsupervised PCA.
The SPCA works as follows: Assume that we have a data matrix $X$ of size $n \times p$, and we denote the $j$th feature by $X_j$.

1. Compute standard univariate regression coefficients for measuring the univariate effect of each feature $X_j$ on $Y$ separately as $w_j = X_j Y^T / \sqrt{X_j X_j^T}$

2. Reduce the data matrix $X$ to include only those features whose univariate coefficients $w_j$ exceed a threshold $\theta$ in absolute value.
   - Let $C_\theta$ be the collection of such indices that $|w_j| > \theta$. The threshold $\theta$ is estimated by cross-validation of the likelihood (or log-likelihood) ratio statistic. The matrix $X_\theta$ is denoted as the one consisting of the columns of $X$ corresponding to $C_\theta$.

3. Compute the first few principal components of the reduced data matrix.
   - The SVD of $X_\theta$ is $X_\theta = U_\theta S_\theta V_\theta^T \Rightarrow U_\theta = X_\theta V_\theta S_\theta^{-1} = X_\theta W_\theta$.
   - Write $U_\theta = (u_{\theta,1}, u_{\theta,2}, \ldots, u_{\theta,m})$. We call $u_{\theta,1}$ the first supervised principal component of $X$, $u_{\theta,2}$ the second supervised principal component of $X$, and so on.
4. Use the principal components calculated in step 3 in a regression model or a classification algorithm to predict the outcome.

- One can fit a univariate linear regression model with response $y$ and predictor $u_{\theta,1}$,

$$\hat{y}^{\text{spec}, \theta} = \bar{y} + \hat{\beta}u_{\theta,1}.$$ 

- Note that $u_{\theta,1}$ is a left singular vector of $X_{\theta}$, so it has mean zero and unit norm. Hence $\hat{\beta} = u_{\theta,1}^T y$, and $\bar{y}$ is the mean of the response $y$ as the intercept.

- The linear regression model estimate can be regarded as a restricted linear model estimate using all the predictors in $X_{\theta}$,

$$\hat{y}^{\text{spec}, \theta} = \bar{y} + \hat{\beta}u_{\theta,1} = \bar{y} + \hat{\beta}X_{\theta}w_{\theta,1} = \bar{y} + X_{\theta}\hat{\beta}_{\theta},$$

where $\hat{\beta}_{\theta} = \hat{\beta}w_{\theta,1}$. In fact, by padding $w_{\theta,1}$ with zeros corresponding to the variables excluded $C_{\theta}$, and the prediction is linear in all $p$ variables.
5. Use this model to predict the outcome. A test variable vector $\mathbf{x}^*$ is given in this step, and a prediction can be made from the regression model as follows:

- Each of components of $\mathbf{x}^*$ should be centered by using the means we derived from the training data: $x_j^* \leftarrow x_j^* - \overline{x}_j$.
- The predicted outcome is
  $$\hat{y}^* = \overline{y} + \beta_j \hat{x}_j = \overline{y} + x_j^* \hat{\beta}_j,$$
  where $\hat{x}_j$ is the appropriate sub-vector of $\mathbf{x}^*$, i.e., $\hat{x}_j$ is the extracting vector of $\mathbf{x}^*$ based on the variables in $\mathbf{X}_\theta$.

In the case of uncorrelated predictors, it is easy to verify that the supervised principal components procedure has the desired behavior:

- it yields all predictors whose standardized univariate coefficients exceed $\theta$ in an absolute value.
Principal component regression (PCR) is a regression analysis technique that is based on principal component analysis (PCA).

The prime idea of PCR is to use scores rather than the original data for the regression step.

In PCR, instead of regressing the response on the features directly, the principal components of the features are used as regressors.

One typically uses only a subset of all the principal components for regression, making PCR a kind of regularized procedure and also a type of shrinkage estimator.

Often the principal components with higher variances (the ones based on eigenvectors corresponding to the higher eigenvalues of the sample variance-covariance matrix of the features) are selected as regressors.
One major use of PCR lies in overcoming the multicollinearity problem which arises when two or more of the explanatory variables are close to being collinear.

PCR can deal with such situations by excluding some of the low-variance principal components in the regression step.

In addition, by usually regressing on only a subset of all the principal components, PCR can result in *dimension reduction* through substantially lowering the effective number of parameters characterizing the underlying model.

This can be particularly useful in settings with high-dimensional covariates.
The PCR method is divided into three major steps:

1. Perform PCA on the observed data matrix $X$ for the features to obtain the principal components, and then (usually) select a subset, based on some appropriate criteria, of the principal components so obtained for further use.

2. Now regress the observed vector of outcomes on the selected principal components as covariates, using ordinary least squares regression (linear regression) to get a vector of estimated regression coefficients (with dimension equal to the number of selected principal components).

3. Finally transform this vector back to the scale of the actual covariates, using the selected PCA loadings (the eigenvectors corresponding to the selected principal components) to get the final PCR estimator (with dimension equal to the total number of covariates) for estimating the regression coefficients characterizing the original model.
PCR identifies linear combinations, or directions, that best represent the predictors $X_1, X_2, \ldots, X_p$.

These directions are identified in an *unsupervised* way, since the response $Y$ is not used to help determine the principal component directions. That is, the response does not *supervise* the identification of the principal components.

Partial least squares (PLS) is a supervised alternative to PCR.

Like PCR, PLS is a dimension reduction method, which first identifies a new set of features $Z_1, \ldots, Z_M$ that are linear combinations of the original features, and then fits a linear model via OLS using these $M$ new features.

But unlike PCR, PLS identifies these new features in a supervised way - that is, it makes use of the response $Y$ in order to identify new features that not only approximate the old features well, but also that are related to the response.
The whole process of PLS is as follows:

1. Standardize each of the variables to have zero mean and unit norm, and compute the univariate regression coefficient \( w = X^T y \).

2. Define \( u = Xw \) and use it in a linear regression model with \( y \).

3. Compute \( q = y^T u \) and \( p = X^T u \), when the suitable latent variables are obtained, regression vector \( \beta_{PLS} \) could be obtained by the following formula:

\[
\beta_{PLS} = W (p^T W)^{-1} q^T.
\]

- This regression vector \( \beta_{PLS} \) can be applied to predict the new samples.
- The number of latent variables in PLS model should be optimized by cross-validation in order to avoid the underfitting or overfitting.
Linear Discriminant Analysis

- Linear discriminant analysis (FDA) is a traditional method of supervised dimensionality reduction and continues to be practically useful.
- LDA aims to find a linear combination of features that characterizes or separates two or more classes of objects or events.
- The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.
- LDA considers maximizing the following objective: $J(w) = w^T S_B w / (w^T S_W w)$ where $S_B = \sum (\mu_c - \bar{x})(\mu_c - \bar{x})^T$ is the “between classes scatter matrix”, and $S_W = \sum \sum (x_j^c - \mu_c)(x_j - \mu_c)^T$ is the “within classes scatter matrix”.
- The objective $J$ is invariant w.r.t. rescaling of the vectors $w$ so that we can choose $w$ such that the denominator is simply $w^T S_W w = 1$. 
Hence we can transform the problem of maximizing $J$ into the following constrained optimization problem:

$$\min_w (-w^T S_B w / 2) \text{ subject to } w^T S_w w = 1.$$  

The corresponding Lagrangian is

$$L(w, \lambda) = [-w^T S_B w + \lambda (w^T S_w w - 1)] / 2.$$  

Differentiating $L$ wrt $w$ yields

$$S_B w = \lambda S_w w \Rightarrow S_w^{-1} S_B w = \lambda w \Rightarrow (S_w^{-1} S_B - \lambda I)w = 0.$$  

This almost looks like an eigenvalue equation, if the matrix $S_w^{-1} S_B$ would have been symmetric (in fact, it is called a generalized eigen-problem).
However, we can apply the following transformation, using the fact that $S_B$ is symmetric positive definite and can hence be written as $S_B = S_B^{1/2} S_B^{1/2}$ where $S_B^{1/2}$ is constructed from its eigenvalue decomposition as

$$S_B = U \Lambda V \Rightarrow S_B^{1/2} = U \Lambda^{1/2} V.$$ 

Defining $v = S_B^{1/2} w$ we obtain $S_B^{1/2} S_W^{-1} S_B^{1/2} v = \lambda v$.

The above is a regular eigenvalue problem for a symmetric, positive definite matrix $S_B^{1/2} S_W^{-1} S_B^{1/2}$ and for which we can find eigenvalue-eigenvector pairs $(\lambda_k, v_k)$ that would correspond to solutions $w_k = S_B^{-1/2} v_k$. 
Remains to choose which eigenvalue and eigenvector corresponds to the desired solution.

Looking at the original objective $J$, we want the eigenvector corresponding to the largest eigenvalue.

This is easily verified by transforming to the dual formulation of the problem, where we use $w_k^T S_B w_k = 1$ and $w_k^T S_W w_k = 1/\lambda_k$, resulting in a dual Lagrangian

$$L_D = \text{constant} + \frac{\lambda_k}{2}$$

which we need to maximize over $\lambda_k$. 

$LDA$
Nonlinear dimensionality reduction works in different ways. For example, a low-dimensional surface can be mapped on a high-dimensional space so that a nonlinear relationship among the features can be found.

In theory, a lifting function $f(x)$ can be used to map the features onto a higher-dimensional space. On a higher space the relationship among the features can be viewed as linear and therefore is easily detected.

This is then mapped back on the lower dimensional space and the relationship can be viewed as nonlinear.

In practice kernel functions can be designed to create the same effect without the need to explicitly compute the lifting function.
A generalization of PCA and matrix factorization, called *generalized low rank models* (GLRMs) (Udell et al. 2016), has become a popular alternative approach to dimension reduction to capture nonlinear relationships.

GLRMs reduce the dimension of a data set by producing a condensed vector representation for every row and column in the original data.

Specifically, given a data set $A$ with $m$ rows and $n$ columns, a GLRM consists of a decomposition of $A$ into numeric matrices $X$ and $Y$.

The matrix $X$ has the same number of rows as $A$, but only a small, *user-specified* number of columns $k$.

The matrix $Y$ has $k$ rows and $n$ columns, where $n$ is equal to the total dimension of the embedded features in $A$. 
Each row of $Y$ is an archetypal feature formed from the columns of $A$, and each row of $X$ corresponds to a row of $A$ projected onto this smaller dimensional feature space.

We can approximately reconstruct $A$ from the matrix product $X \times Y$, which has rank $k$.

The number $k$ is chosen to be much less than both $m$ and $n$. The smaller $k$ is, the more compression we gain from our low rank representation.

To make this more concrete, let’s look at an example using the mtcars data where we have 32 data samples and 11 features.

If we want to reduce matrix $A$ to a rank of $k=3$ then our objective is to produce two matrices $X$ and $Y$ that when we multiply them together produce a near approximation to the original values in $A$. 

GLRMs
We call the condensed columns and rows in matrices $X$ and $Y$, respectively, “archetypes”, representations of the original features and observations.

The archetypes in $X$ represent each observation projected onto the smaller dimensional space, and the archetypes in $Y$ represent each feature projected onto the smaller dimensional space.
The resulting archetypes are similar in spirit to the PCs in PCA; as they are a reduced feature set that represents our original features.

» If our features truly behave in a linear and orthogonal manner than our archetypes produced by a GLRM will produce the same reduced feature set as PCA.

» However, if they are not linear, then GLRM will provide archetypes that are not necessarily orthogonal.

Here are two questions:

» How does GLRM produce the archetype values?

» How do you select the appropriate value for $k$?
There are a number of methods available to identify the optimal archetype values for each element in $X$ and $Y$; however, the most common is based on *alternating minimization*.

Alternating minimization simply alternates between minimizing some loss function for each feature in $X$ and $Y$.

In essence, random values are initially set for the archetype values in $X$ and $Y$. The loss function is computed, and then the archetype values in $X$ are slightly adjusted via gradient descent and the improvement in the loss function is recorded.

The archetype values in $Y$ are then slightly adjusted and the improvement in the loss function is recorded.

This process is continued until the loss function is optimized or some suitable stopping condition is reached.
The optimal archetypal values are selected based on minimizing some loss function.

The loss function should reflect the intuitive notion of what it means to “fit the data well”.

The most common loss function is the quadratic loss

$$\text{quadratic loss} = \min \{ \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{i,j} - X_i Y_j)^2 \}.$$ 

Quadratic loss, similar to SSE, can be heavily influenced by outliers.

If you do not want to emphasize outliers in your data set, or if you just want to try minimize errors for lower values in addition to higher values (e.g., trying to treat low-cost products equally as important as high-cost products) then you can use the Huber loss function which essentially applies quadratic loss to small errors and uses the absolute value for errors with larger values.
Another important component to fitting GLRMs is **regularization**.

Regularization applied to GLRMs can be used to constrain the size of the archetypal values in $X$ with $r_x(X)$ and/or $Y$ with $r_y(Y)$.

$$\text{regularization} = \min \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{i,j} - X_i Y_j)^2 + r_x(X) + r_y(Y) \right\}.$$ 

This can help to create **sparse** $X$ and/or $Y$ matrices to mitigate the effect of negative features in the data (e.g., multicollinearity or excessive noise) which can help prevent overfitting.

As the above equation illustrates, we can regularize both matrices $X$ and $Y$. However, when performing dimension reduction we are mainly concerned with finding a condensed representation of the features, or columns.

We are more concerned with regularizing the $Y$ matrix using $r_y(Y)$ to force the $Y$ matrix to be column-sparse.
There are several regularizers to choose from.

- You can use a ridge regularizer to retain all columns but force many of the values to be near zero.
- You can also use a LASSO regularizer which will help zero out many of the columns; the LASSO helps you perform automated feature selection.

Lastly, how do we select the appropriate value for $k$? There are two main approaches, both of which will be illustrated in the section that follows.

- First, if we’re using GLRM to describe your data, then we can use many of the same approaches such as eigenvalue criterion, PVE criterion, or scree plot criterion, to assess how different values of $k$ minimize our loss function.
- If we are using GLRM to produce a model that will be used to assign future observations to the reduced dimensions, then we should use some form of CV.
Another approach to nonlinear dimensionality reduction is by using manifolds.

It is based on the assumption that the data lie on an embedded nonlinear manifold which has lower dimension than the raw data space and lies within it.

Several algorithms exist working in the manifold space.

A commonly used method of finding an appropriate manifold, isometric mapping (Isomap), constructs the manifold by joining each point only to its nearest neighbors.

Distances between points are then taken as geodesic distances on the resulting graph.

Isomap tries to preserve the geodesic distances in the lower dimension.
Isomap starts by creating a neighborhood network.

Next it uses graph distance to the approximate geodesic distance between all pairs of points.

Through eigenvalue decomposition of the geodesic distance matrix, it finds the low dimensional embedding of the dataset.

In non-linear manifolds, the Euclidean metric for distance holds good if and only if neighborhood structure can be approximated as linear. If neighborhood contains holes, then Euclidean distances can be highly misleading.

In contrast to this, if we measure the distance between two points by following the manifold, we will have a better approximation of how far or near two points are.
Let's understand this with an extremely simple 2-D example.

Suppose our data lies on a circular manifold in a 2-D structure like in the image below.
We will reduce the data to 1-D using the Euclidean distances and approximate geodesic distances.

Now, if we look at the 1-D mapping based on the Euclidean metric, we see that for points which are far apart (a & b) have been mapped poorly.

Only the points which can be approximated to lie on a linear manifold (c & d) give satisfactory results.

On the other hand, see the mapping with geodesic distances, it nicely approximates the close points as neighbors and far away points as distant.

The geodesic distances between two points in the image are approximated by graph distance between the two points.

Euclidean distances should not be used for approximating the distance between two points in non-linear manifolds while geodesic distances can be used.
- Isomap uses the local information to create a global similarity matrix.
- Isomap algorithm uses Euclidean metrics to prepare the neighborhood graph.
- It approximates the geodesic distance between two points by measuring shortest path between these points using graph distance.
- It approximates both global as well as the local structure of the dataset in the low dimensional embedding.
Steps of the Isomap algorithm are:

- Create a neighborhood graph and adjacency matrix from the dataset.
- After neighborhood search, we will calculate the geodesic distances between the points.
- While creating our neighborhood network, we have to make sure that the resulting graph is a single connected component.
- We need to iterate over the different values of neighborhood selection parameter to get the fully connected graph.
- Before eigenvalue decomposition, we have to square the distance and double center the squared similarities matrix.
- After eigenvalue decomposition select the first $K$ eigenvectors with $K$ highest eigenvalues.
The following is the plot of a subset of MNIST dataset after Isomap dimension reduction.
There are several other nonlinear manifold methods for feature extraction:

» **Multi-Dimension Scaling (MDS):** a distance-preserving manifold learning method

» **Locally Linear Embedding (LLE):** a topology preserving manifold learning method

» **Laplacian Eigenmaps:** use spectral techniques to perform dimensionality reduction with assumption that the data lies in a low-dimensional manifold in a high-dimensional space

» **Kernel PCA:** compute the covariance matrix of the data after being transformed into a higher-dimensional space, then project the transformed data onto the first $k$ eigenvectors of that matrix, and use the kernel trick to factor away much of the computation

» **Principal curves and manifolds:** give the natural geometric framework for nonlinear dimensionality reduction and extend the geometric interpretation of PCA by explicitly constructing an embedded manifold, and by encoding using standard geometric projection onto the manifold.
Thank you!

Questions?