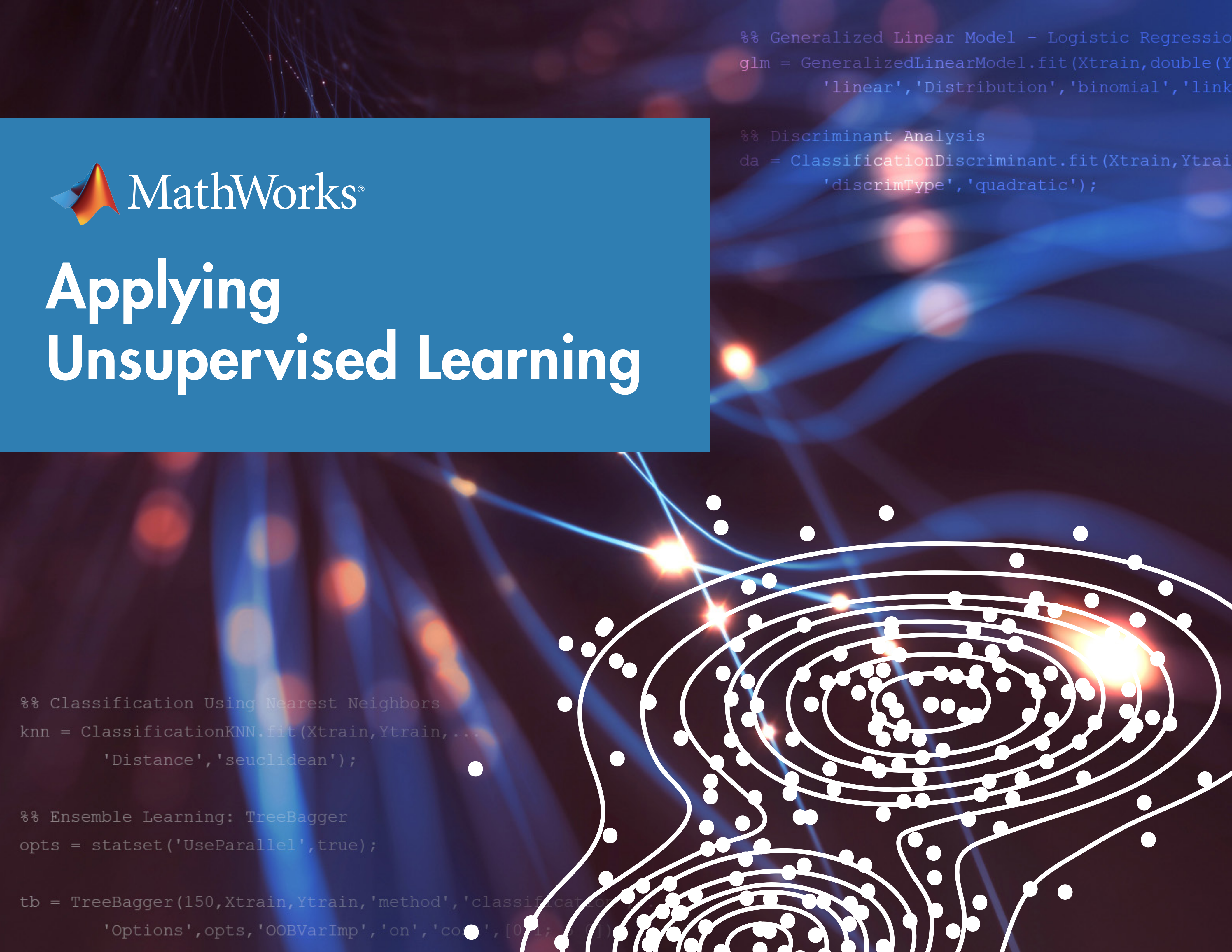
MathWorks®

Applying Unsupervised Learning



```
%% Generalized Linear Model - Logistic Regressio
glm = GeneralizedLinearModel.fit(Xtrain,double(Y
        'linear','Distribution','binomial','link

%% Discriminant Analysis
da = ClassificationDiscriminant.fit(Xtrain,Ytrai
        'discrimType','quadratic');
```

```
%% Classification Using Nearest Neighbors
knn = ClassificationKNN.fit(Xtrain,Ytrain,...
        'Distance','seuclidean');

%% Ensemble Learning: TreeBagger
opts = statset('UseParallel',true);

tb = TreeBagger(150,Xtrain,Ytrain,'method','classification',...
        'Options',opts,'OOBVarImp','on','cost',[0 1; 1 0]);
```

MathWorks®

When to Consider Unsupervised Learning

Unsupervised learning is useful when you want to explore your data but don't yet have a specific goal or are not sure what information the data contains. It's also a good way to reduce the dimensions of your data.

Unsupervised Learning Techniques

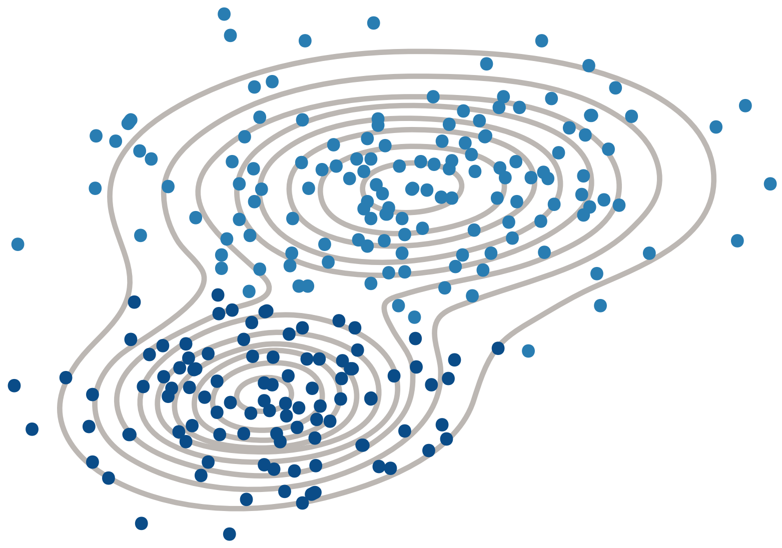
As we saw in section 1, most unsupervised learning techniques are a form of cluster analysis.

In cluster analysis, data is partitioned into groups based on some measure of similarity or shared characteristic. Clusters are formed so that objects in the same cluster are very similar and objects in different clusters are very distinct.

Clustering algorithms fall into two broad groups:

- Hard clustering, where each data point belongs to only one cluster
- Soft clustering, where each data point can belong to more than one cluster

You can use hard or soft clustering techniques if you already know the possible data groupings.



Gaussian mixture model used to separate data into two clusters.

If you don't yet know how the data might be grouped:

- Use self-organizing feature maps or hierarchical clustering to look for possible structures in the data.
- Use cluster evaluation to look for the "best" number of groups for a given clustering algorithm.

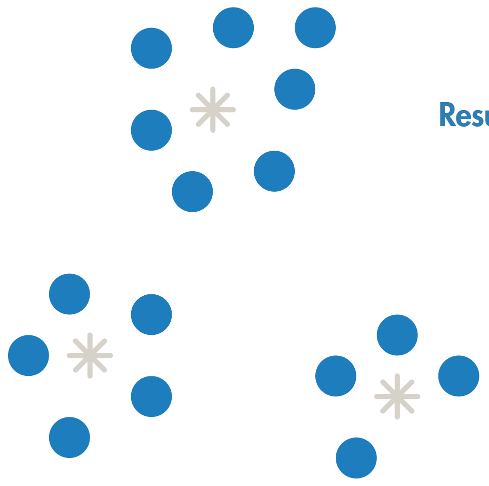
Common Hard Clustering Algorithms

k-Means

How it Works
Partitions data into k number of mutually exclusive clusters. How well a point fits into a cluster is determined by the distance from that point to the cluster's center.

Best Used...

- When the number of clusters is known
- For fast clustering of large data sets



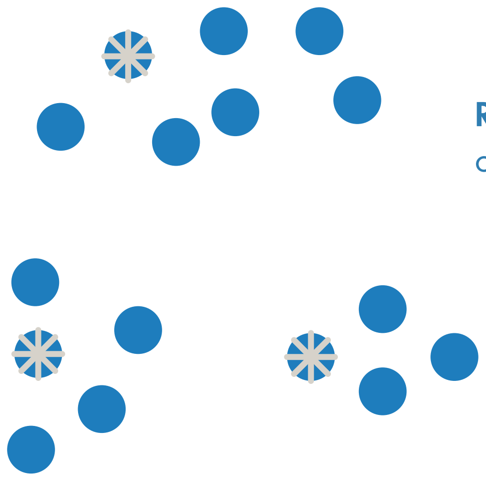
Result: Cluster centers

k-Medoids

How It Works
Similar to k-means, but with the requirement that the cluster centers coincide with points in the data.

Best Used...

- When the number of clusters is known
- For fast clustering of categorical data
- To scale to large data sets



Result: Cluster centers that coincide with data points

Common Hard Clustering Algorithms *continued*

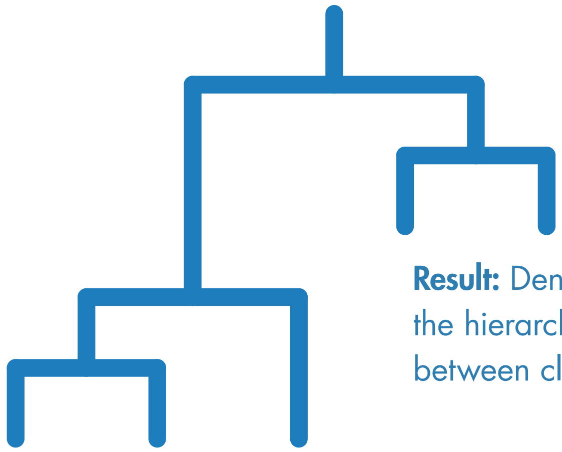
Hierarchical Clustering

How it Works

Produces nested sets of clusters by analyzing similarities between pairs of points and grouping objects into a binary, hierarchical tree.

Best Used...

- When you don't know in advance how many clusters are in your data
- You want visualization to guide your selection



Result: Dendrogram showing the hierarchical relationship between clusters

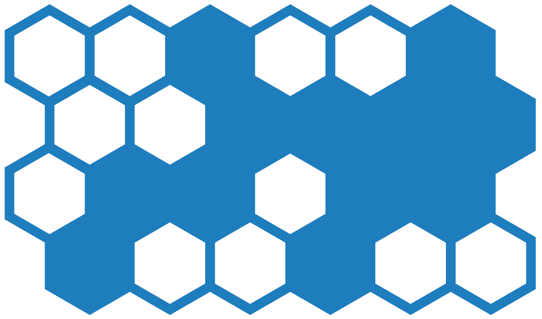
Self-Organizing Map

How It Works

Neural-network based clustering that transforms a dataset into a topology-preserving 2D map.

Best Used...

- To visualize high-dimensional data in 2D or 3D
- To deduce the dimensionality of data by preserving its topology (shape)



Result: Lower-dimensional (typically 2D) representation

Common Hard Clustering Algorithms *continued*

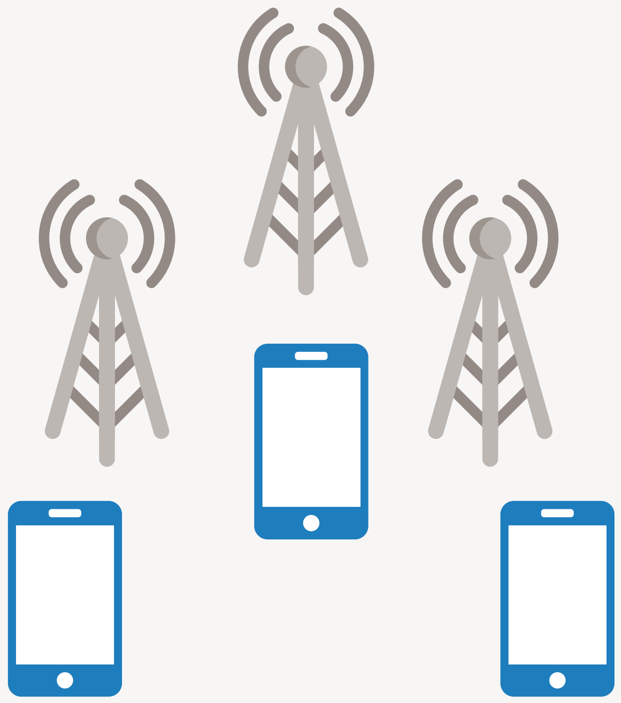
Example: Using k-Means Clustering to Site Cell Phone Towers

A cell phone company wants to know the number and placement of cell phone towers that will provide the most reliable service. For optimal signal reception, the towers must be located within clusters of people.

The workflow begins with an initial guess at the number of clusters that will be needed. To evaluate this guess, the engineers compare service with three towers and four towers to see how well they're able to cluster for each scenario (in other words, how well the towers provide service).

A phone can only talk to one tower at a time, so this is a hard clustering problem. The team uses k-means clustering because k-means treats each observation in the data as an object having a location in space. It finds a partition in which objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible.

After running the algorithm, the team can accurately determine the results of partitioning the data into three and four clusters.



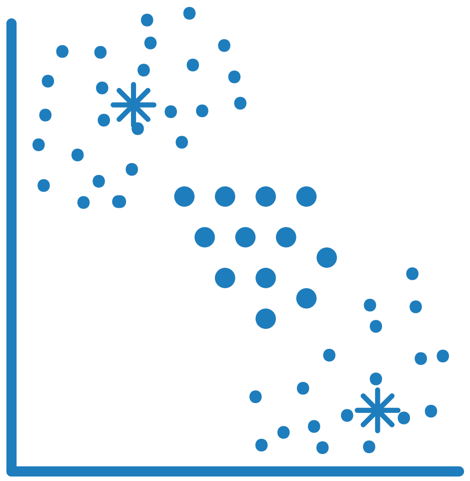
Common Soft Clustering Algorithms

Fuzzy c-Means

How it Works
Partition-based clustering when data points may belong to more than one cluster.

Best Used...

- When the number of clusters is known
- For pattern recognition
- When clusters overlap



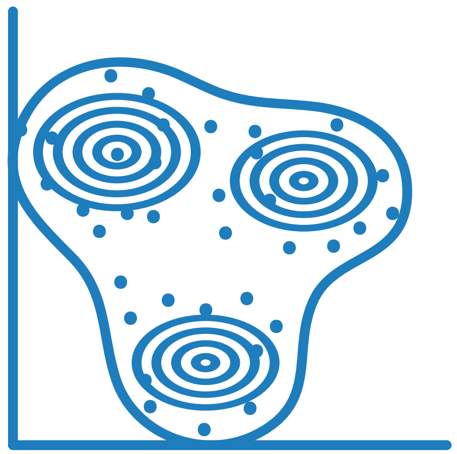
Result: Cluster centers (similar to k-means) but with fuzziness so that points may belong to more than one cluster

Gaussian Mixture Model

How It Works
Partition-based clustering where data points come from different multivariate normal distributions with certain probabilities.

Best Used...

- When a data point might belong to more than one cluster
- When clusters have different sizes and correlation structures within them



Result: A model of Gaussian distributions that give probabilities of a point being in a cluster

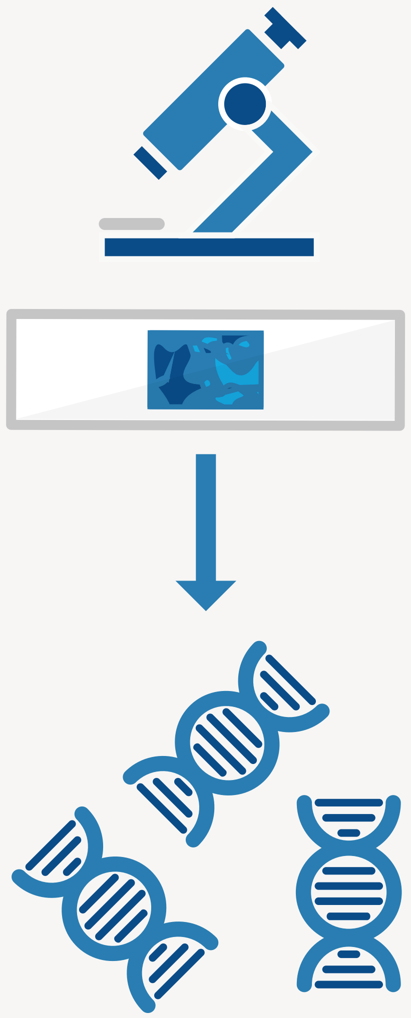
Common Soft Clustering Algorithms *continued*

Example: Using Fuzzy c-Means Clustering to Analyze Gene Expression Data

A team of biologists is analyzing gene expression data from microarrays to better understand the genes involved in normal and abnormal cell division. (A gene is said to be "expressed" if it is actively involved in a cellular function such as protein production.)

The microarray contains expression data from two tissue samples. The researchers want to compare the samples to determine whether certain patterns of gene expression are implicated in cancer proliferation.

After preprocessing the data to remove noise, they cluster the data. Because the same genes can be involved in several biological processes, no single gene is likely to belong to one cluster only. The researchers apply a fuzzy c-means algorithm to the data. They then visualize the clusters to identify groups of genes that behave in a similar way.



Improving Models with Dimensionality Reduction

Machine learning is an effective method for finding patterns in big datasets. But bigger data brings added complexity.

As datasets get bigger, you frequently need to reduce the number of features, or *dimensionality.*

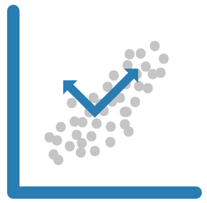
Example: EEG Data Reduction

Suppose you have electroencephalogram (EEG) data that captures electrical activity of the brain, and you want to use this data to predict a future seizure. The data was captured using dozens of leads, each corresponding to a variable in your original dataset. Each of these variables contains noise. To make your prediction algorithm more robust, you use dimensionality reduction techniques to derive a smaller number of features. Because these features are calculated from multiple sensors, they will be less susceptible to noise in an individual sensor than would be the case if you used the raw data directly.

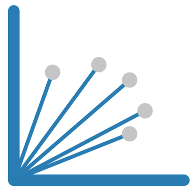
Common Dimensionality Reduction Techniques

The three most commonly used dimensionality reduction techniques are:

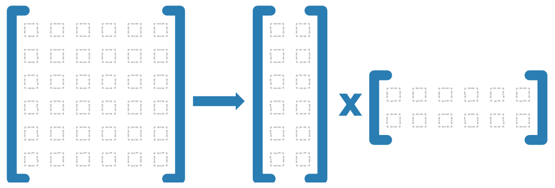
Principal component analysis (PCA)—performs a linear transformation on the data so that most of the variance or information in your high-dimensional dataset is captured by the first few principal components. The first principal component will capture the most variance, followed by the second principal component, and so on.



Factor analysis—identifies underlying correlations between variables in your dataset to provide a representation in terms of a smaller number of unobserved latent, or common, factors.



Nonnegative matrix factorization—used when model terms must represent nonnegative quantities, such as physical quantities.



Using Principal Component Analysis

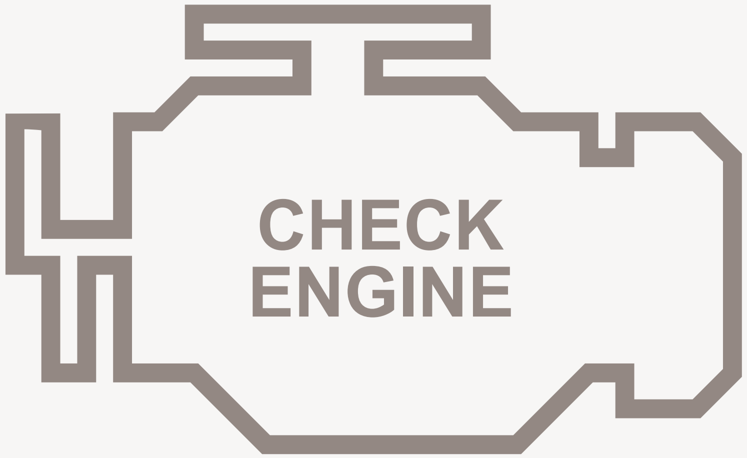
In datasets with many variables, groups of variables often move together. PCA takes advantage of this redundancy of information by generating new variables via linear combinations of the original variables so that a small number of new variables captures most of the information.

Each principal component is a linear combination of the original variables. Because all the principal components are orthogonal to each other, there is no redundant information.

Example: Engine Health Monitoring

You have a dataset that includes measurements for different sensors on an engine (temperatures, pressures, emissions, and so on). While much of the data comes from a healthy engine, the sensors have also captured data from the engine when it needs maintenance.

You cannot see any obvious abnormalities by looking at any individual sensor. However, by applying PCA, you can transform this data so that most variations in the sensor measurements are captured by a small number of principal components. It is easier to distinguish between a healthy and unhealthy engine by inspecting these principal components than by looking at the raw sensor data.



Using Factor Analysis

Your dataset might contain measured variables that overlap, meaning that they are dependent on one another. Factor analysis lets you fit a model to multivariate data to estimate this sort of interdependence.

In a factor analysis model, the measured variables depend on a smaller number of unobserved (latent) factors. Because each factor might affect several variables, it is known as a common factor. Each variable is assumed to be dependent on a linear combination of the common factors.

Example: Tracking Stock Price Variation

Over the course of 100 weeks, the percent change in stock prices has been recorded for ten companies. Of these ten, four are technology companies, three are financial, and a further three are retail. It seems reasonable to assume that the stock prices for companies in the same sector will vary together as economic conditions change. Factor analysis can provide quantitative evidence to support this premise.

Using Nonnegative Matrix Factorization

This dimension reduction technique is based on a low-rank approximation of the feature space. In addition to reducing the number of features, it guarantees that the features are nonnegative, producing models that respect features such as the nonnegativity of physical quantities.

Example: Text Mining

Suppose you want to explore variations in vocabulary and style among several web pages. You create a matrix where each row corresponds to an individual web page and each column corresponds to a word ("the","a","we", and so on). The data will be the number of times a particular word occurs on a particular page.

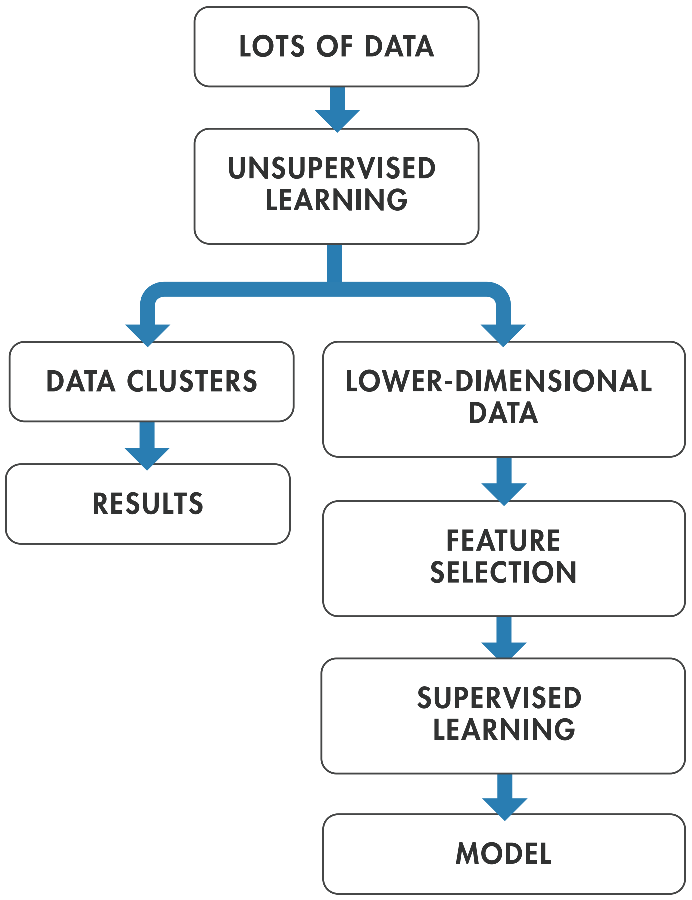
Since there more than a million words in the English language, you apply nonnegative matrix factorization to create an arbitrary number of features that represent higher-level concepts rather than individual words. These concepts make it easier to distinguish between, say, news, educational content, and online retail content.

Next Steps

In this section we took a closer look at hard and soft clustering algorithms for unsupervised learning, offered some tips on selecting the right algorithm for your data, and showed how reducing the number of features in your dataset improves model performance. As for your next steps:

- Unsupervised learning might be your end goal. For example, if you are doing market research and want to segment consumer groups to target based on web site behavior, a clustering algorithm will almost certainly give you the results you're looking for.
- On the other hand, you might want to use unsupervised learning as a preprocessing step for supervised learning. For example, apply clustering techniques to derive a smaller number of features, and then use those features as inputs for training a classifier.

In section 4 we'll explore supervised learning algorithms and techniques, and see how to improve models with feature selection, feature reduction, and parameter tuning.



Learn More

Ready for a deeper dive? Explore these unsupervised learning resources.

Clustering Algorithms and Techniques

k-Means

Use *K*-Means and Hierarchical Clustering to Find Natural Patterns in Data

Cluster Genes Using K-Means and Self-Organizing Maps

Color-Based Segmentation Using K-Means Clustering

Hierarchical Clustering

Connectivity-Based Clustering

Iris Clustering

Self-Organizing Maps

Cluster Data with a Self-Organizing Map

Fuzzy C-Means

Cluster Quasi-Random Data Using Fuzzy C-Means Clustering

Gaussian Mixture Models

Gaussian Process Regression Models

Cluster Data from Mixture of Gaussian Distributions

Cluster Gaussian Mixture Data Using Soft Clustering

Tune Gaussian Mixture Models

Image Processing Example: Detecting Cars with Gaussian Mixture Models

Dimensionality Reduction

Analyze Quality of Life in U.S. Cities Using PCA

Analyze Stock Prices Using Factor Analysis

Nonnegative Factorization

Perform Nonnegative Matrix Factorization

Model Suburban Commuting Using Subtractive Clustering

80823v00