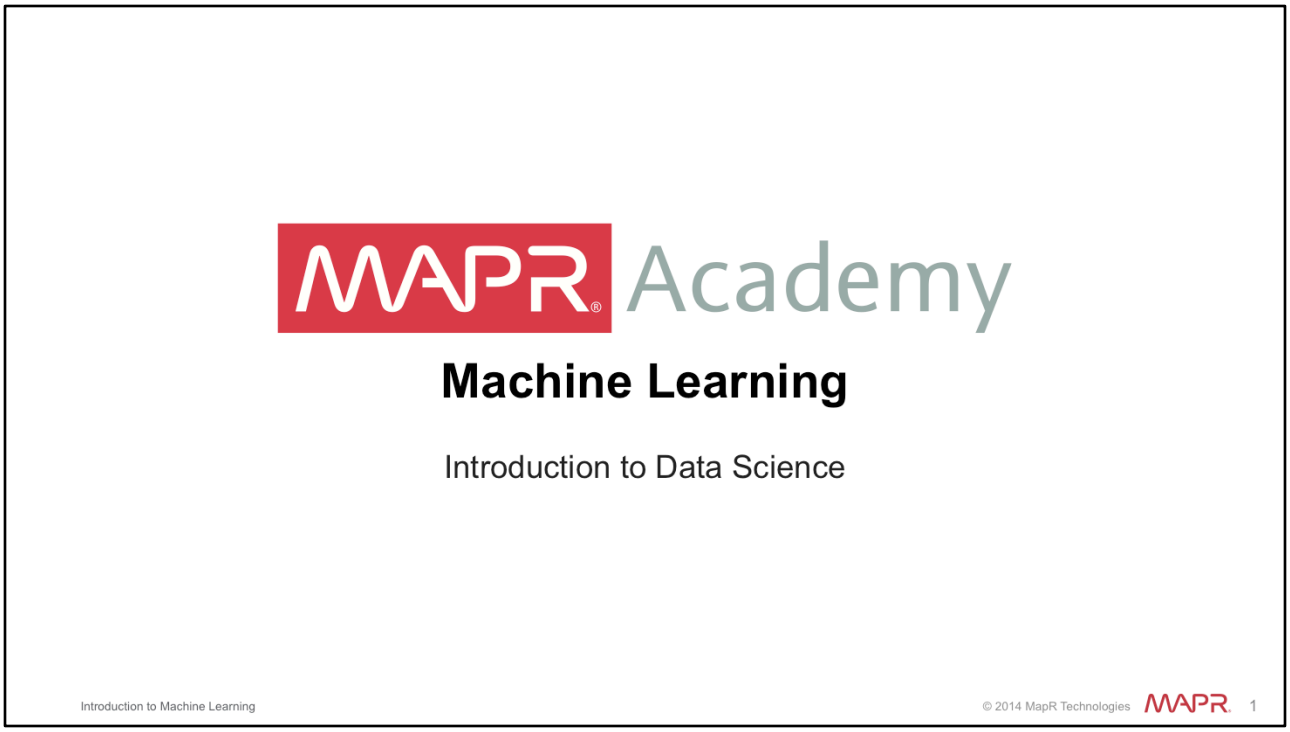
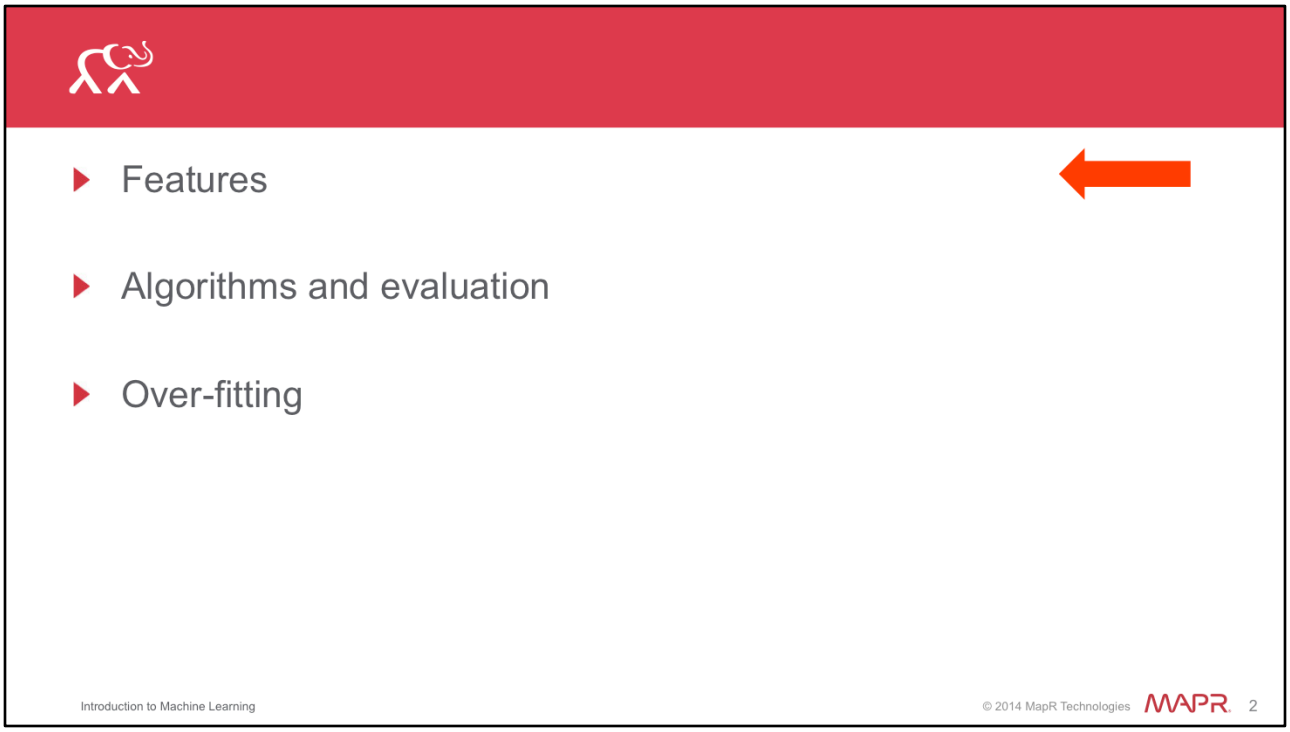
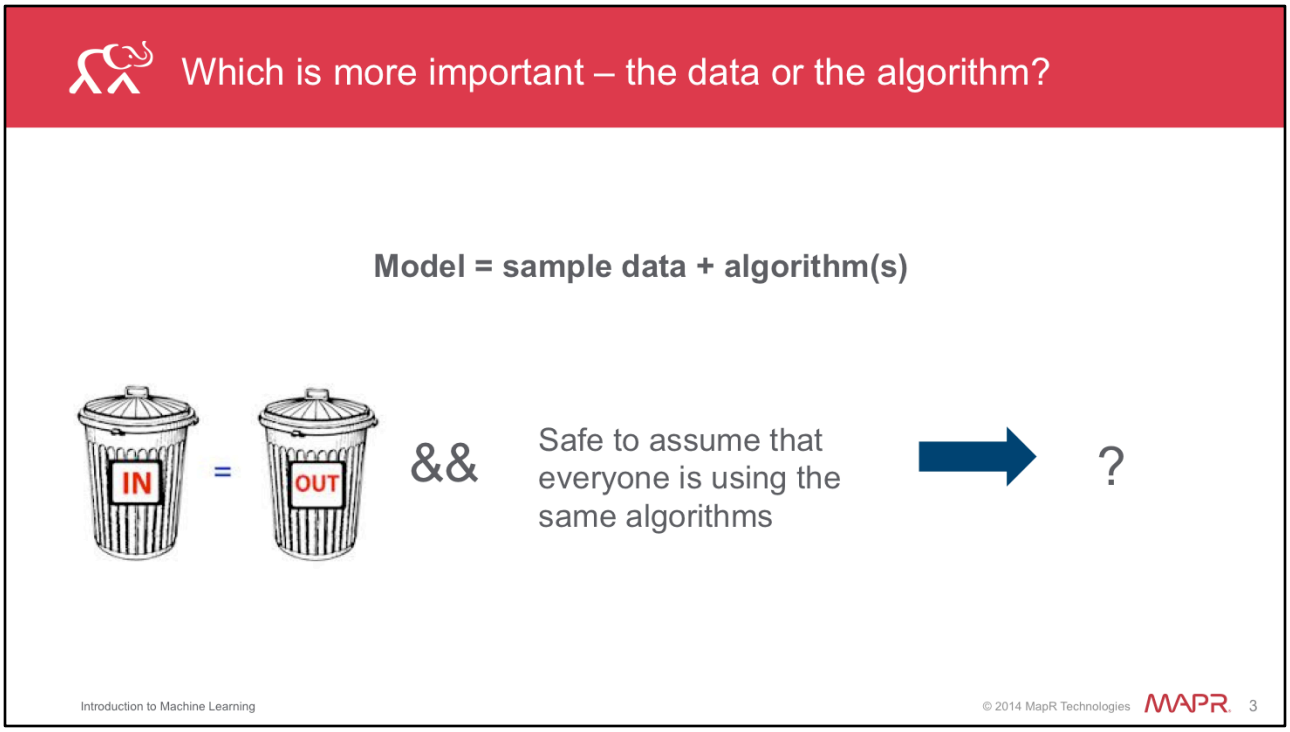
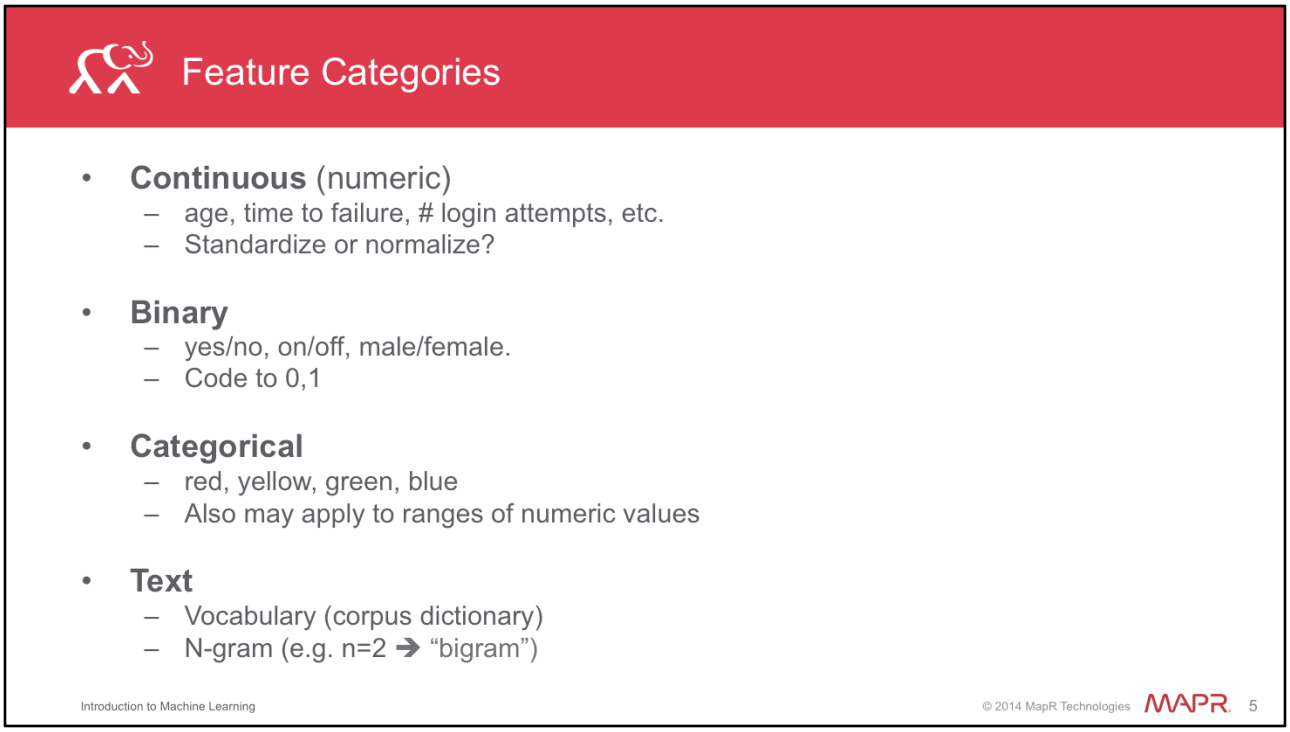

MAPR Academy
Machine Learning
Introduction to Data Science
Introduction to Machine Learning
© 2014 MapR Technologies
MAPR
1


Features
Algorithms and evaluation
Over-fitting
Introduction to Machine Learning
© 2014 MapR Technologies
2


Which is more important – the data or the algorithm?
Model = sample data + algorithm(s)
IN
=
OUT
&&
Safe to assume that everyone is using the same algorithms
?
Introduction to Machine Learning
© 2014 MapR Technologies
3

What is a feature?

- **Feature**
  - individual measurable property of an observation
  - Used for predicting the target (supervised learning)
- Also known as
  - ***Predictor***
  - ***Attribute***
  - ***Variable***
- **Examples** (for a person)
  - Height (in inches)
  - Weight (in pounds)
  - Credit score (unit-less)
  - Sex (male/female)
  - Married (yes/no)
  - Nationality (American, French, Chinese, …)
  - ZIP code (94086, …)



YOU are the expert in determining good features for YOUR model

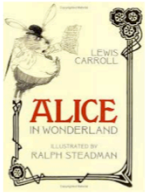
Another category – words in a document, TD-IDF (would fall under transformation?)

Vectorizing Data

- Given an observation, assign features to dimensions

Example: vectorize a text document (bag of words approach)

Dictionary: [a, advance, after, …, you, yourself, youth, zigzag]



[223,1,1,0,…,12,10,6,1]

Example: vectorize an apple

Features: [size, color, weight]

[3.2, 167771848, 100]

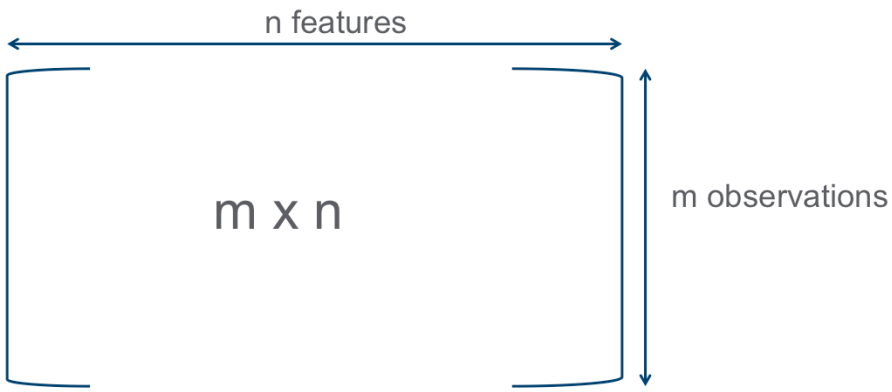
In order to cluster data, the data set must be vectorized. For example, if you wish to cluster text documents, you must define a dictionary of words and associate each word to a dimension. Such a vector might well be constituted of tens of thousands of dimensions. Given a text document (like the text representation of "Alice in Wonderland"), you'd perform a wordcount against the document to determine the vector that represents the document. Similarly, if you wish to cluster apples, you would identify a set of features that can be used to uniquely represent a given apple (for example, the radial size of the apple, the color in hex, and the weight in grams). There may be words you don't care about that just introduce noise into the model – these are called "stop words". Examples of stop words are those you'll find in any document (e.g. a, an, the, some, most, all, …).

What makes a feature “good”?

- The features are ***correlated to the target***
- Features are ***independent of each other***
- The features ***do not leak the target***
- The ***number*** of features ***is minimized***
- ***The number of features is much less than the number of data points***

Target leak example = zip code in sioux falls (changed to fraud center)
Find targets by isolating best-performing variables, then remove & determine perf impact.

Data set dimensionality



***Ideally,* |*features*| <<<<< |*observations*|**

Dimensionality reduction: Principal Component Analysis (Conceptual)

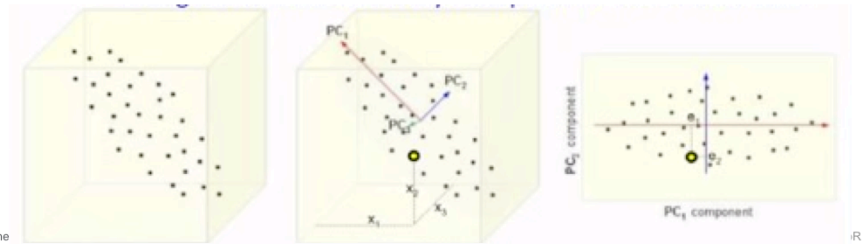
Calculate covariance matrix of original input matrix

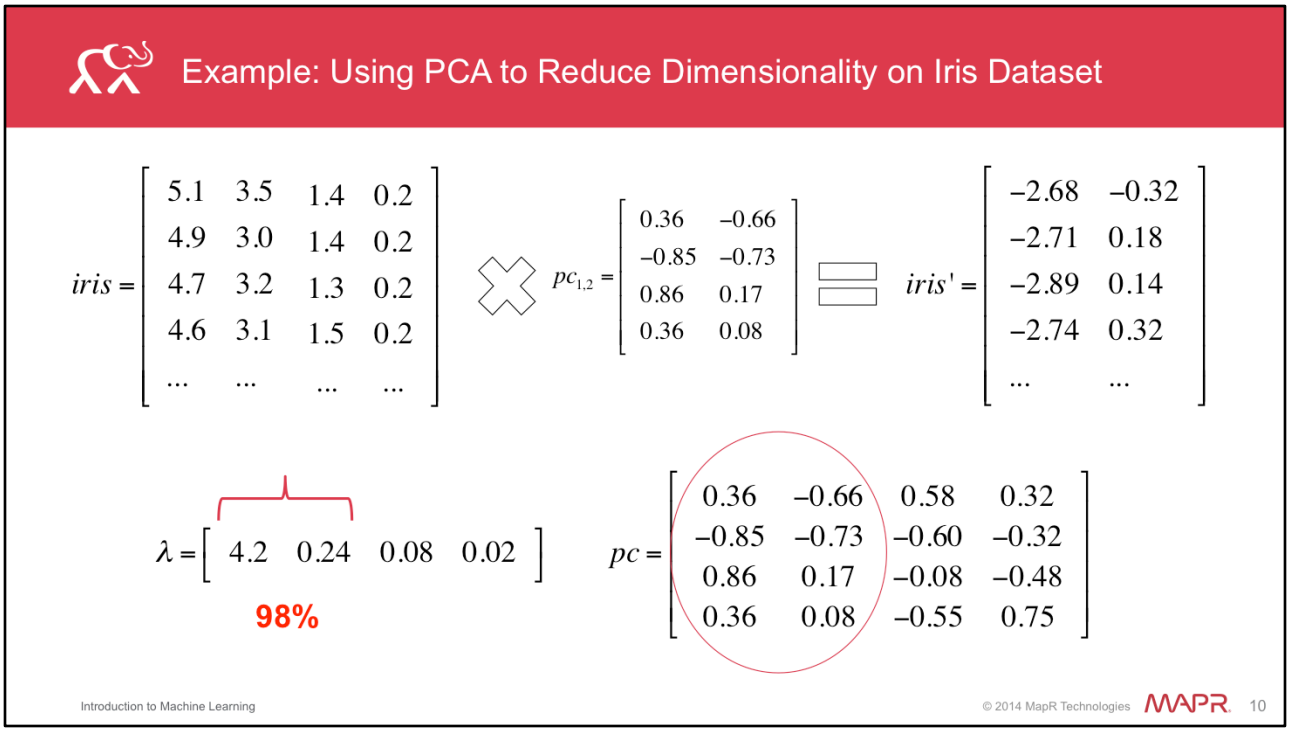
Perform eigenvalue decomposition of covariance matrix

Look for orthogonal dimensions with greatest variance (principal)

Project input matrix into those dimensions (reduction)

Relative covariance is preserved across projection





Given iris dataset (n=150) with 4 features – sepal/petal height/ width. Execute PCA.

Here are 4 eigenvalues and 4 principal components.

The first 2 account for 98 of total variance. So we select only the first 2 PC.

Multiplying the original matrix by PC2 matrix gives us our reconstructed data – we could use this as inputs to a model.

One drawback – if your method allows you to explain results, you will lose that since the new features are difficult to understand.

Feature scale

- **relative magnitude**
  - Some features have much different magnitude than others
  - Model will be dominated by largest magnitude features
- **absolute magnitude**
  - Some features are extremely small and others are extremely large
  - Model may lose information to underflow and overflow

Dealing with feature scale

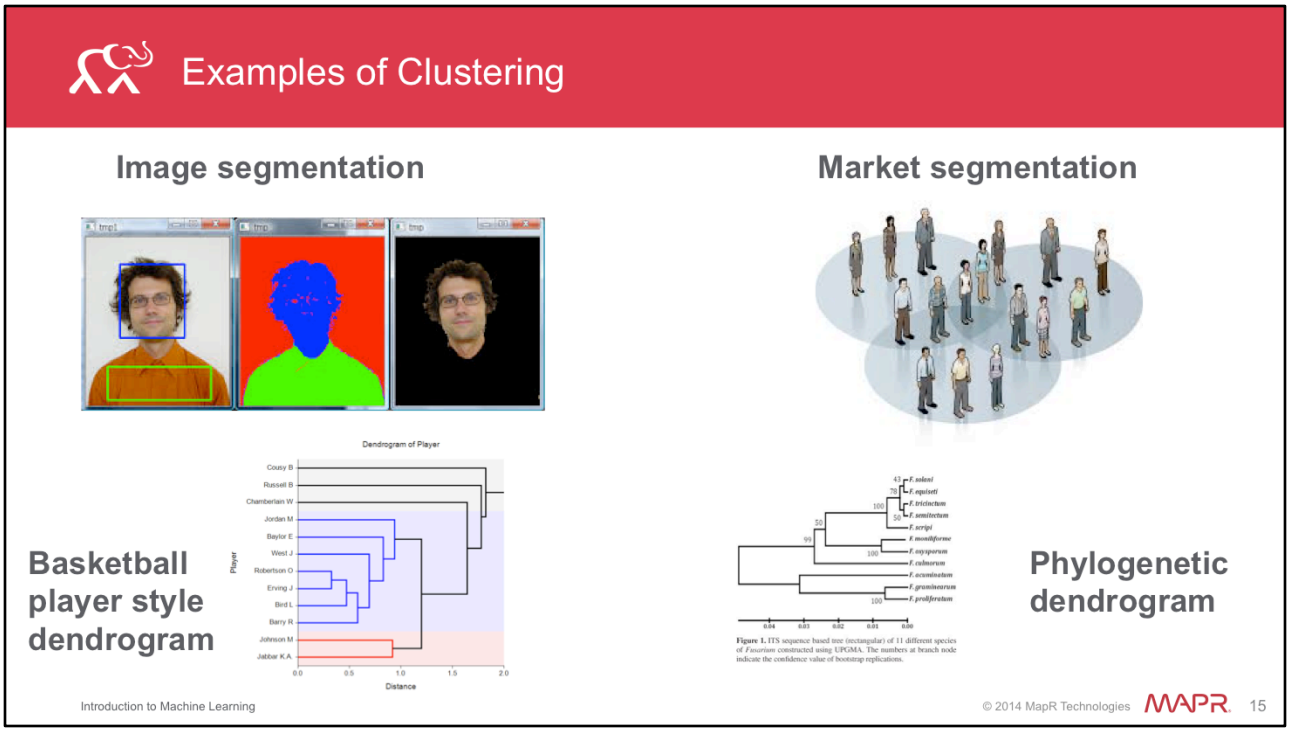
- **Standardization**
  - transform data to measure how many standard deviations from mean
  - Example: $x' = (x - x_{mean}) / (x_{SD})$
- **Normalization**:
  - transform data such that range is between 0 and 1
  - Example: $x' = (x - x_{min}) / (x_{max} - x_{min})$
- ***Both methods preserve covariance***

Feature anomalies

- **Target leaks**
  - In-sample data has “extra” information not found out of sample
  - (hopefully) detected during testing ➔ model too good
  - E.g. using feature “selected free shipping” to predict whether a customer will purchase or not
- **Bias**
  - Sampling bias (e.g. self-selected clinical trial)
  - Observer bias (e.g. measuring stick is off)
  - Funding bias (e.g. fudge the data)
- **Simpson’s Paradox**
  - Trend that appears in different groups disappears (or reverses) when groups are combined
  - E.g. average of averages != average of all points

	1995		1996		1997		Combined	
Derek Jeter	12/48	.250	183/582	.314	190/654	.291	385/1284	**.300**
David Justice	104/411	**.253**	45/140	**.321**	163/495	**.329**	312/1046	.298

- Features
- Algorithms and evaluation
- Over-fitting

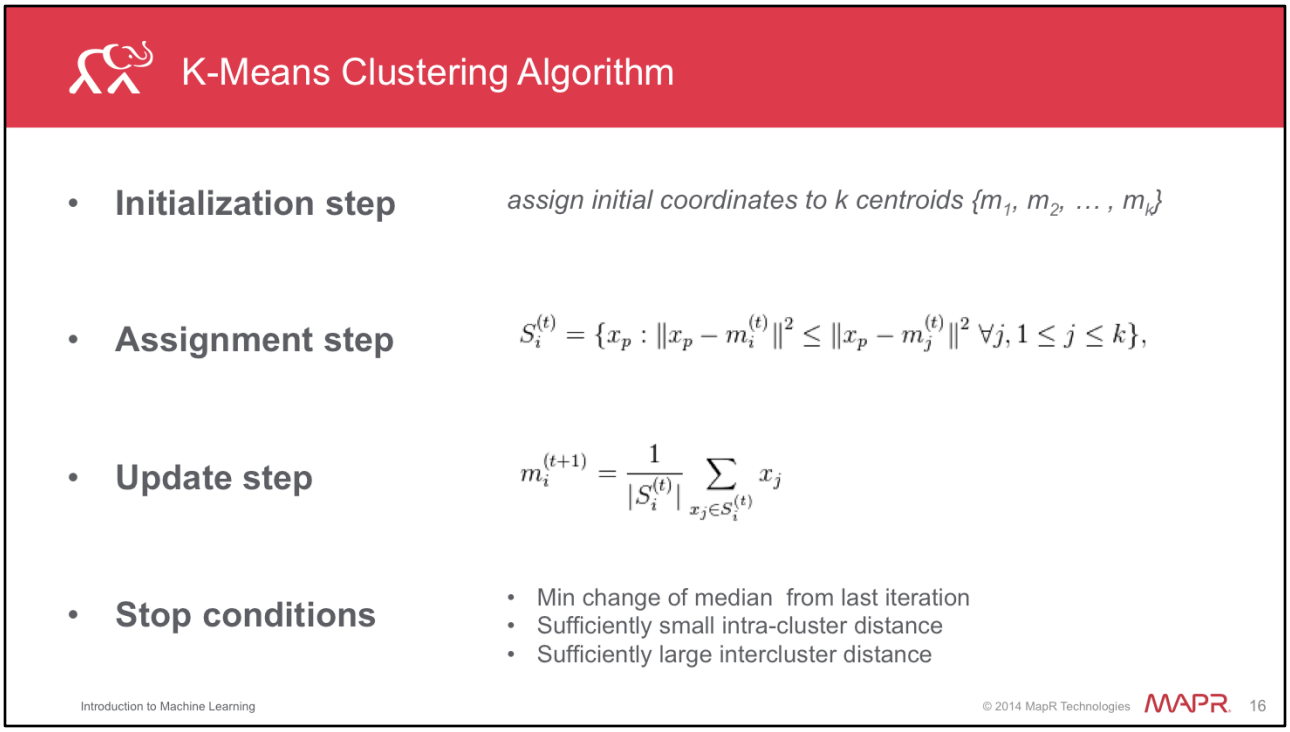


Clustering analysis is used in a variety of domains.

For example, in image segmentation, the objective is to identify borders of objects in an image based on the pixel data. In this example, the user has asked the app to cluster the "head" as a blue cluster and the "shirt" as a green cluster. All other data is represented in red. This could be used for computer vision.

In the case of market segmentation, the objective is to identify similarities between users based on their market habits. This could be used to target ad campaigns for specific groups of users.

You can construct dendrograms using cluster analysis. A dendrogram is a hierarchical structure in which the most similar elements are closest to each other in the tree, and the most

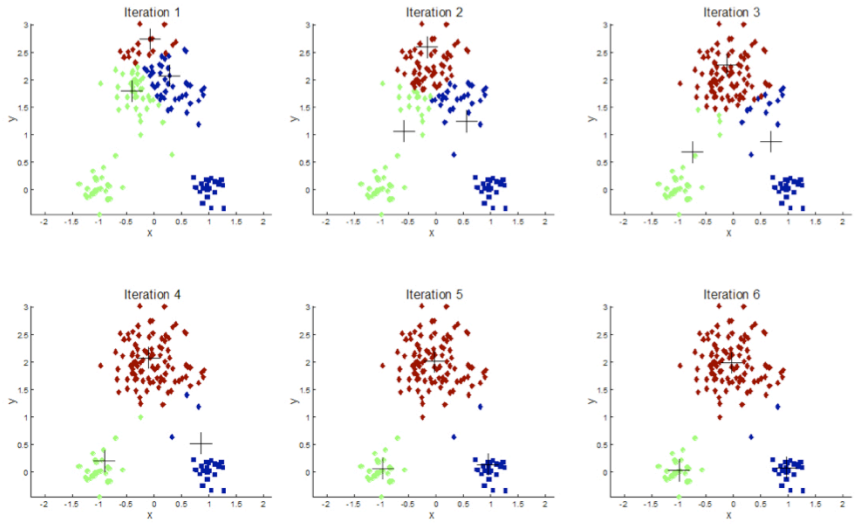


The steps for the k-means algorithm are given above. The algorithm begins by initializing all the coordinates to centroids. There are various ways you can initialize the points, including randomly.

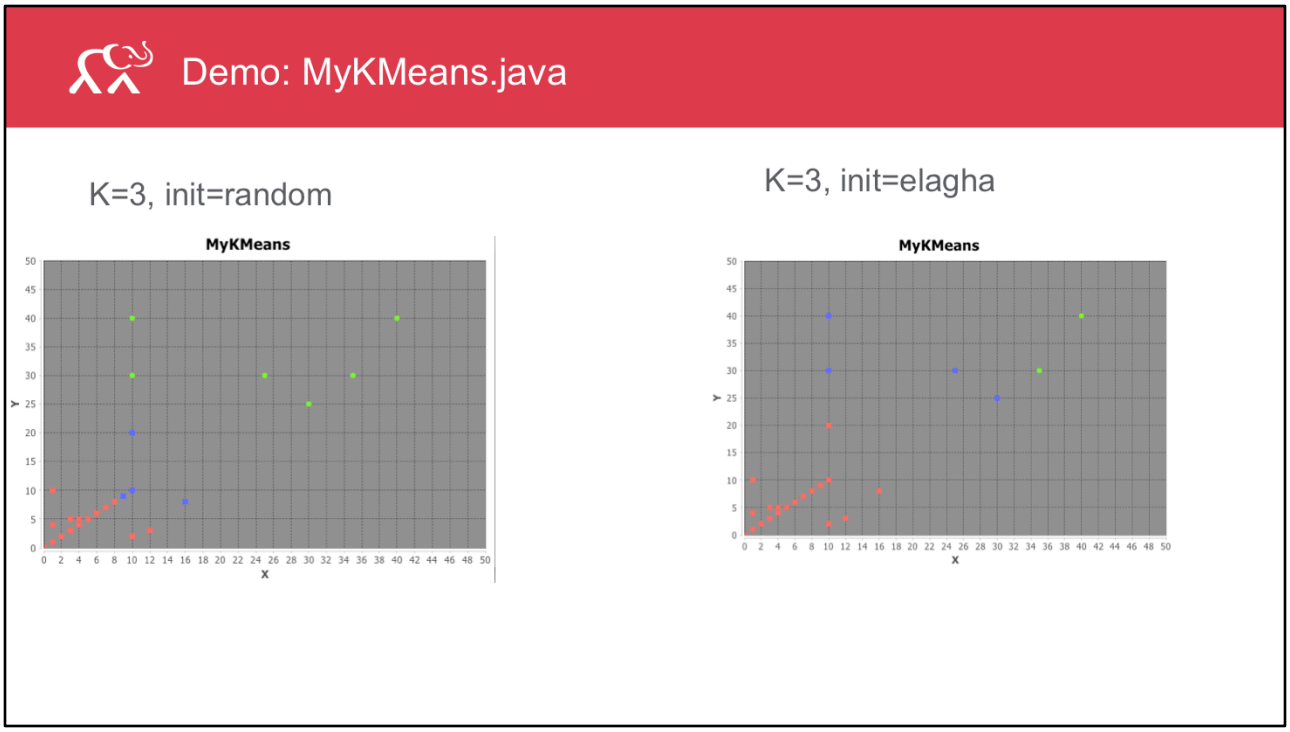
At each pass in the algorithm, you assign each point to its nearest centroid based on some distance metric (usually euclidean distance). The assignment statement reads as follows: “The ith set S at time t is equal to the set of all points x-sub-p such that the distance between x-sub-p and the mean of that set S is less than or equal to the distance between x-sub-p and the mean of all other sets S sub j where j ranges from 1 to k.”

You then update the centroids to be the “centers” of all the points assigned to it in that pass. The update step reads as

Iterations of K-Means Algorithm



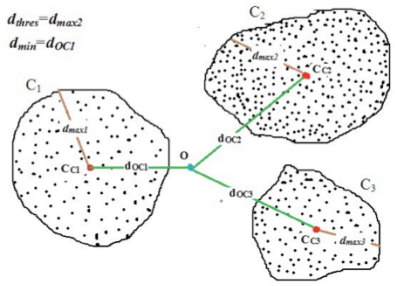
The graphic above shows 6 iterations of the k-means algorithm for a given data set. In this example, k=3. The centroids "move" in each iteration, and the input points stay stationary.



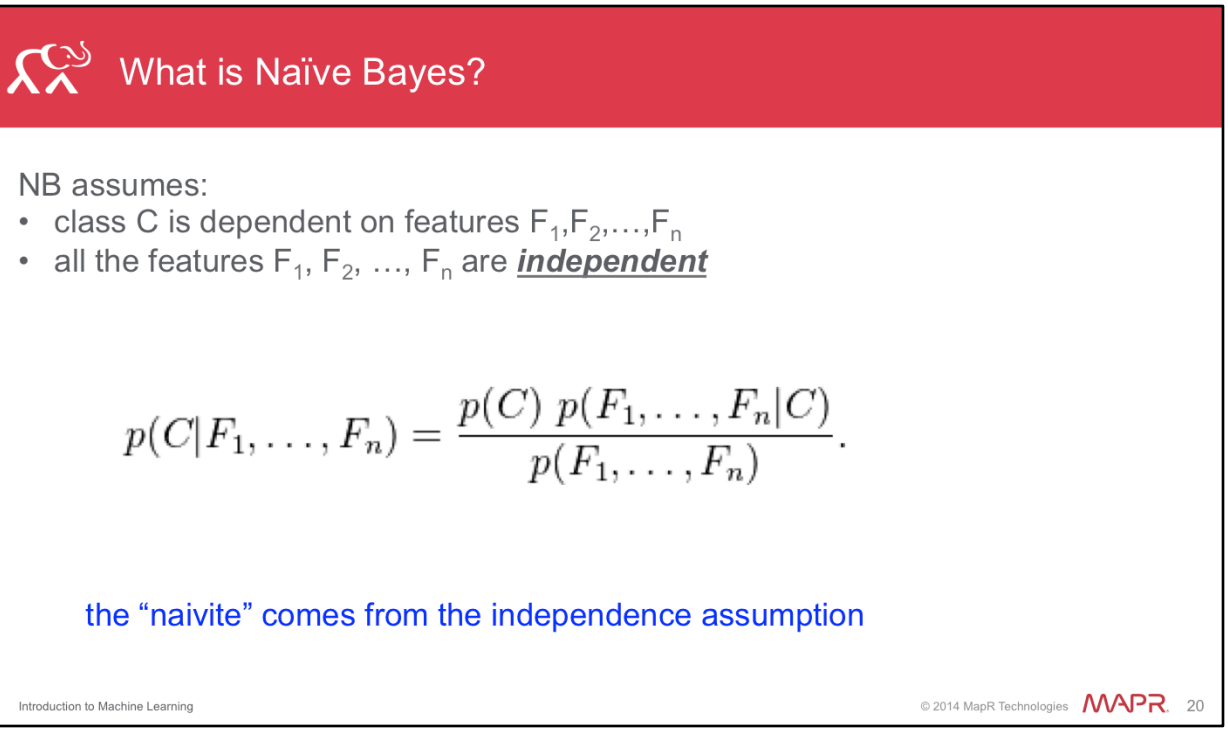
This demonstration runs the k-means algorithm using 2 different initialization methods. The first is random and the second uses an initialization method developed by El Agha et al. Note that the algorithm converges on different clusterings depending on the initialization method.

Distance-based metrics for cluster evaluation

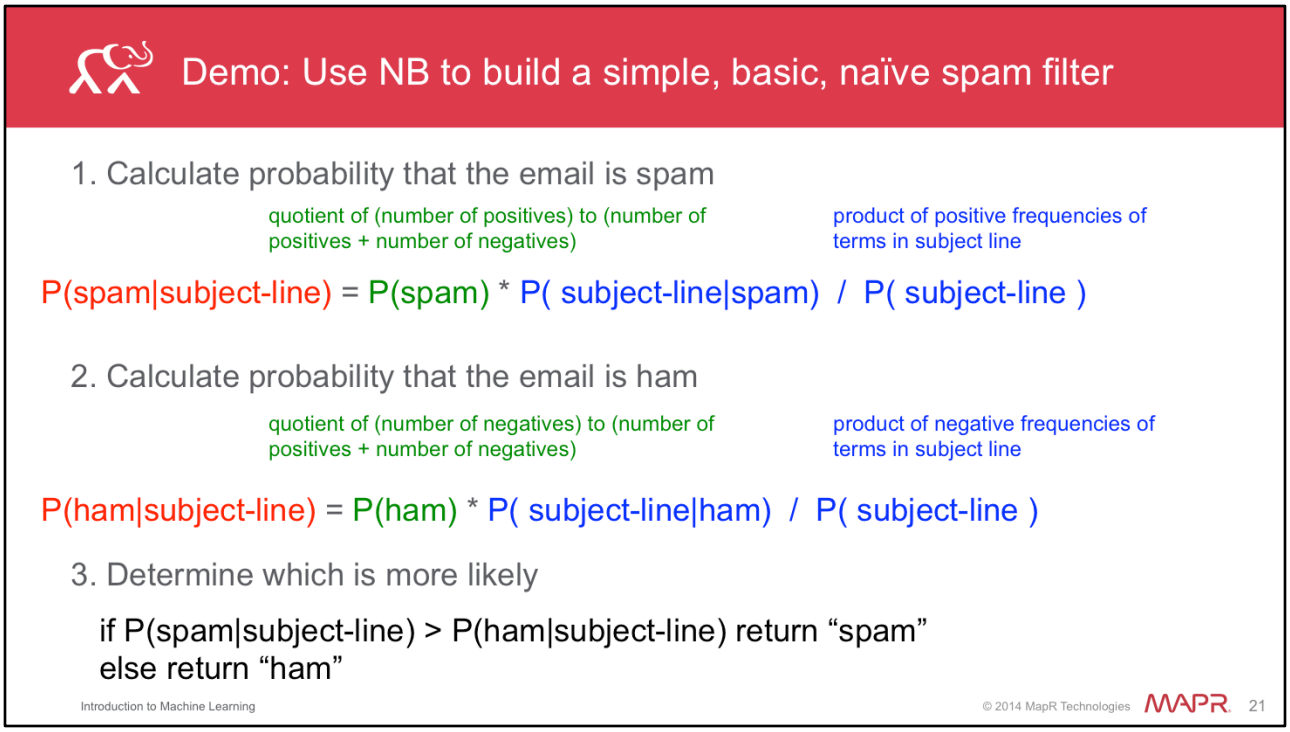
- **Average inter-cluster distance (do I want them close or far?)**
  - Distance between centroids
  - Measures data separation between clusters
- **Average intra-cluster distance (do I want them close or far?)**
  - Distance between points in same cluster
  - Measures data similarity within clusters



In order to evaluate an algorithm, you have to have metrics with which to evaluate. These are either minimization or maximization metrics. For example, you can consider the average distance between centroids as the metric you wish to optimize. Depending on your use case, you may wish to minimize or maximize this distance. You may also consider the average distance between points and their centroid as the metric to optimize. Again, depending on your use case, you may wish to minimize or maximize this distance.



Naïve Bayes is a machine learning algorithm that is based on Bayes theorem. The crux of Naïve Bayes (the "naïve'" part) is based on the assumption that all the features are independent of each other. Even though this may be too strong an assumption, Naïve Bayes has been shown to work well in several domains.



In this demo, we train and query a Naïve Bayes classifier for detecting spam email. This particular algorithm only considers the contents of the subject line in the model.

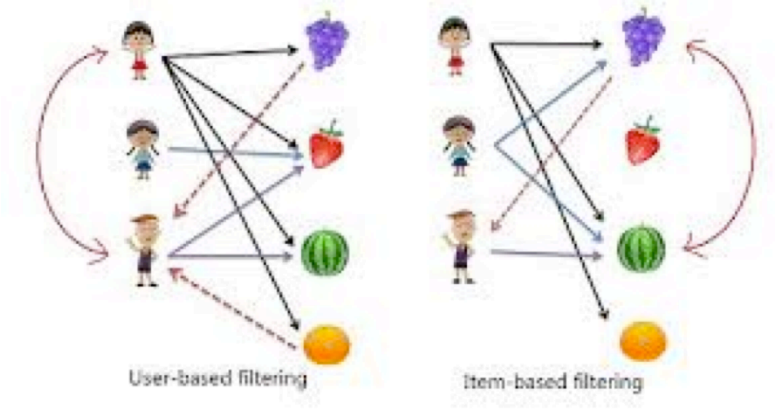
This particular NB program takes as input a line of text which represents the subject line of an email and outputs information regarding whether the model considers it spam or not. Interestingly, the string “vote now” is not considered spam while the same subject line with an exclamation point is considered spam. This is due to the fact that many of the spam emails in the training corpus contain the exclamation mark in their subject line.

Measures based on TP/FP/TN/FN

- **Sensitivity** (aka true positive rate)
  TPR = TP / (TP + FN)
- **Specificity** (aka true negative rate)
  TNR = TN / (TN + FP)
- **Accuracy**
  ACC = (TP + TN) / (P + N)
- Different domains have different “penalties”
  e.g. spam detection vs cancer detection

Based on TP, FP, TN, and FN, you can calculate rate metrics to evaluate your classifier, as shown in the slide above. The higher these values, the better the model.

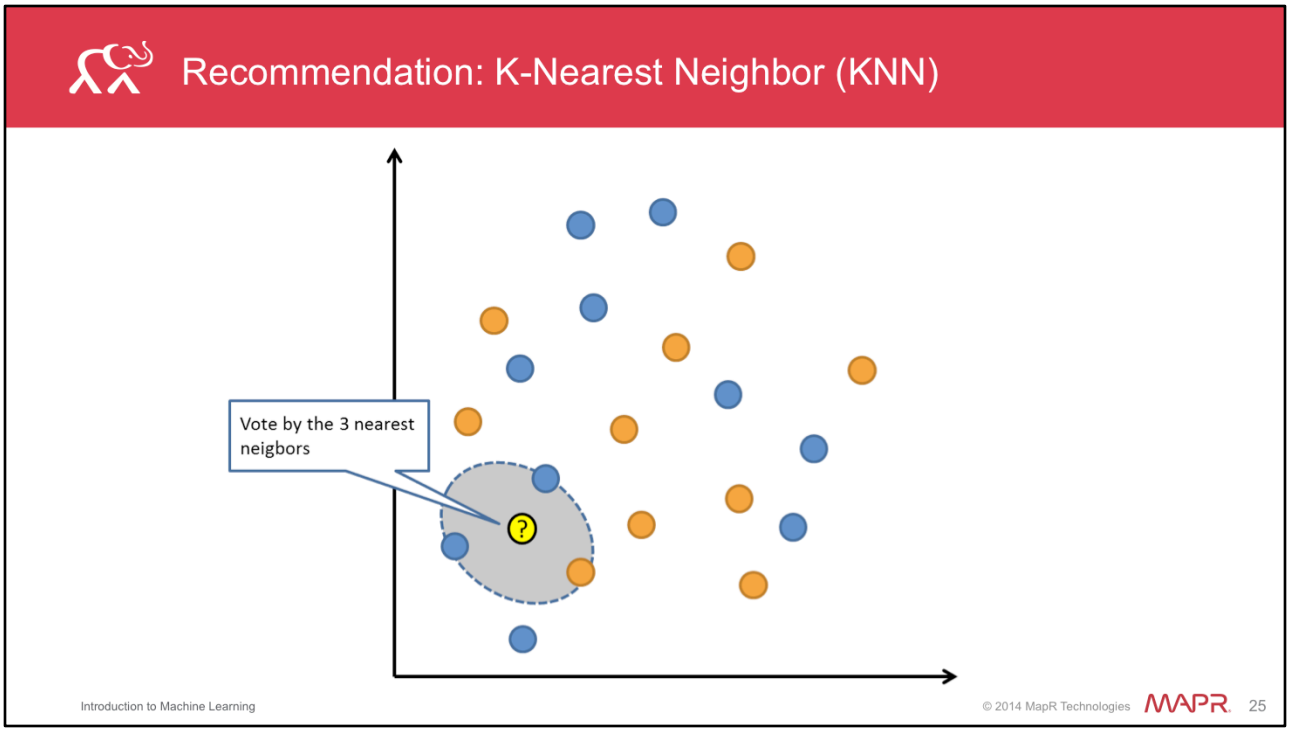
Recommendation: User-based vs item-based filtering



Users are associated with item ***preferences*** through history

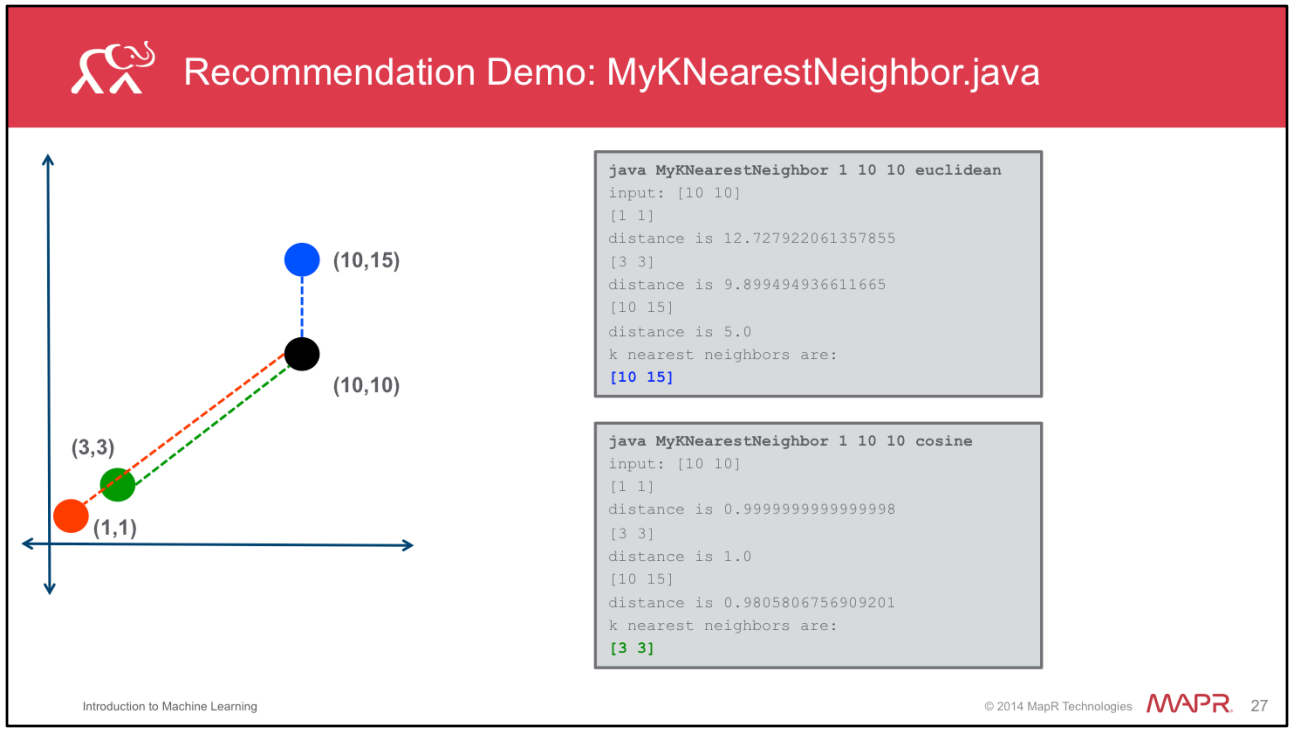
Similarity is constructed between users (or items)

Recommendations based on ***similarity to other users (or items)***


Recommendation: K-Nearest Neighbor (KNN)
Vote by the 3 nearest neigbors
?
Introduction to Machine Learning
© 2014 MapR Technologies
25

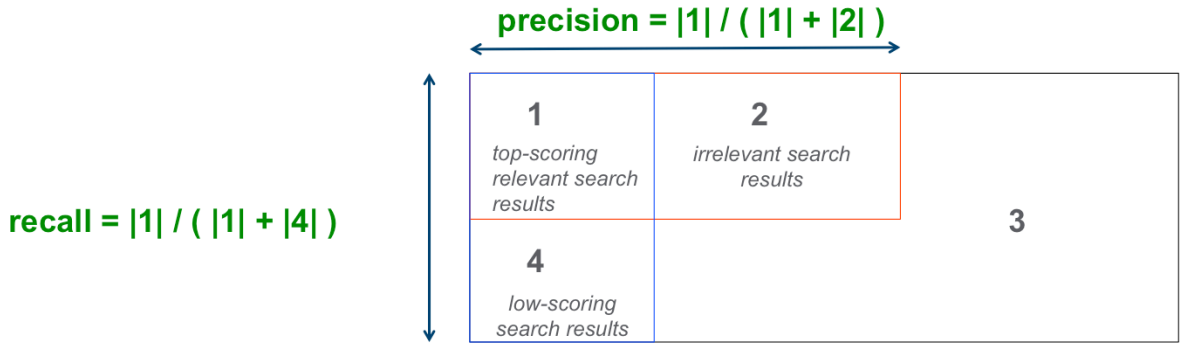
Recommendation: KNN algorithm

- **Distance**
  - Euclidean
  - Manhattan
  - Cosine
- **Majority voting**
  - Problematic if class distribution is skewed
  - Could use distance as weight to compensate
  - Pick odd number for k to avoid ties


Recommendation Demo: MyKNearestNeighbor.java
(10,15)
(10,10)
(3,3)
(1,1)
java MyKNearestNeighbor 1 10 10 euclidean
input: [10 10]
[1 1]
distance is 12.727922061357855
[3 3]
distance is 9.899494936611665
[10 15]
distance is 5.0
k nearest neighbors are:
[10 15]
java MyKNearestNeighbor 1 10 10 cosine
input: [10 10]
[1 1]
distance is 0.9999999999999998
[3 3]
distance is 1.0
[10 15]
distance is 0.9805806756909201
k nearest neighbors are:
[3 3]
Introduction to Machine Learning
© 2014 MapR Technologies
27

Recommendation: Precision and recall in document search

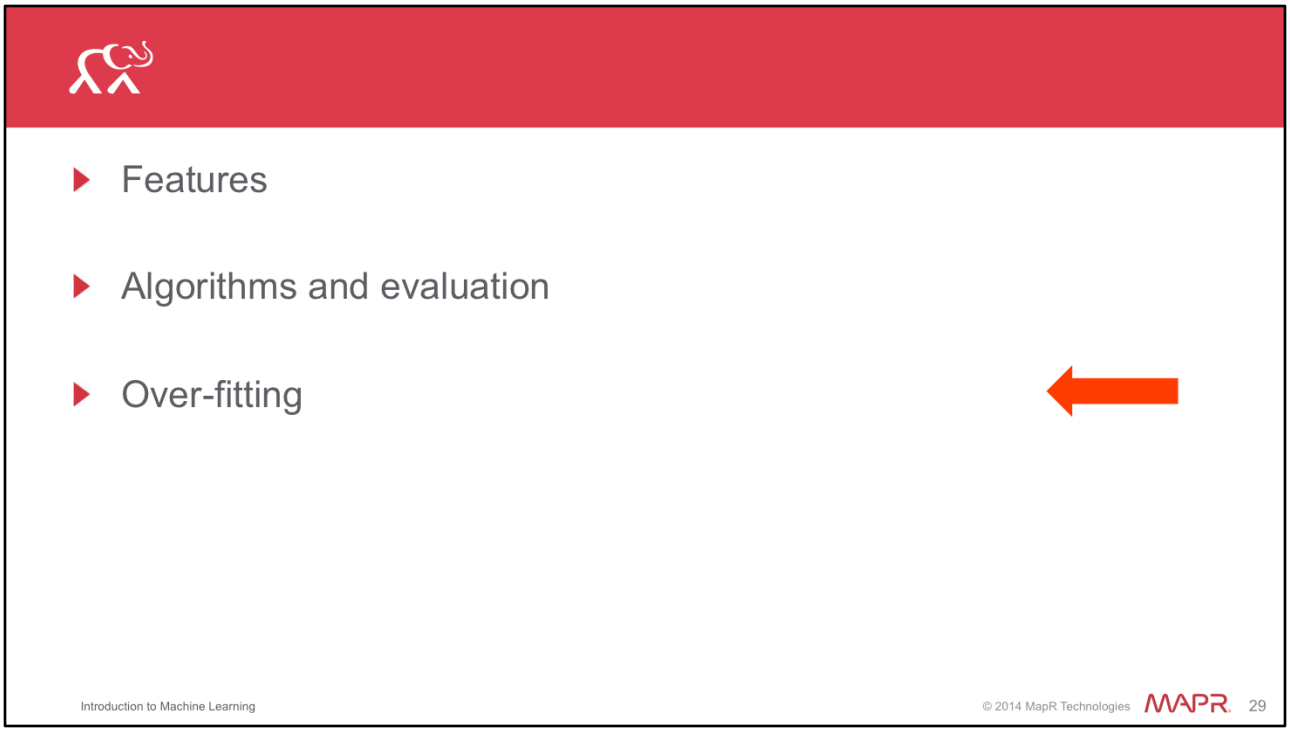
All docs in corpus = 1 + 2 + 3 + 4
All results from search = 1 + 2
Top results from search = 1
Relevant documents = 1 + 4



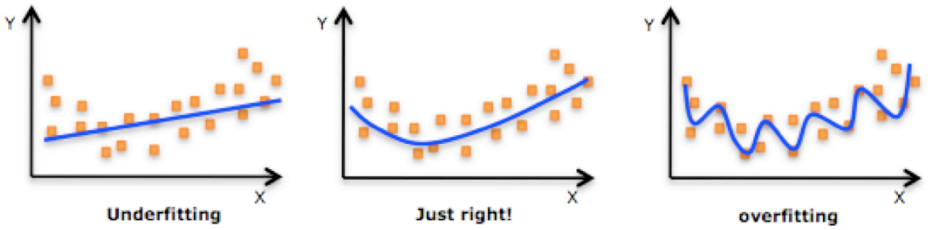
Precision and recall are used to ***evaluate search results***

Precision = proportion of top-scoring results that are relevant

Recall = proportion of relevant results that are top-scoring


Features
Algorithms and evaluation
Over-fitting
Introduction to Machine Learning
© 2014 MapR Technologies
29

Overfitting and underfitting



Dealing with overfitting

- **Regularization**
  - Occam's razor (William of Occam): most plausible explanation is the simplest
  - Impose penalty in cost function based on parameter complexity

    $\min(E_{out}(X, Y)) \rightarrow \min(E_{in}(X, Y) + \lambda \| w \|)$

  Smaller λ ➔ overfitting; larger λ ➔ underfitting

- **Smoothing**
  - Sunrise problem (Laplace): P(sun will rise tomorrow) = ?
  - Use probability distributions for parameter estimates for sparse data

    $$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

  Smaller α ➔ overfitting; larger α ➔ underfitting

Dealing with overfitting: cross-validation

- Leave k out
- Train on N-k points, validate on k points; gives estimate of out-of-sample error (which we want to minimize)
- Example (leave 1 out)
  - Take (x_1, y_1) out, train on N-1; calculate e_1
  - Put (x_1, y_1) back in training set
  - Take (x_2, y_2) out, train on N-1; calculate e_2
  - Put (x_2, y_2) back in training set
  - …
  - Take (x_N, y_N) out, train on N-1; calculate e_N
  - Calculate cross-validation error = $1/N * (e_1 + e_2 + \ldots + e_N)$
- Repeat above steps, each for a *different model* (e.g. higher-order polynomial, different λ , different α, …)
- select model with lowest average cross-validation error

K-fold cross-validation

- Leave 1 out ➔ lots of iterations per epoch ☹
- Cross-validation generalizes to higher values of k (k-fold cross-validation)
- k=N/10 (10-fold cross-validation) is empirically a good rule of thumb
- Leave first k out, train on N-k, calculate e_{k1}; repeat 10 times; calculate average e; perturb model and repeat cross-validation k times; choose model
- Question: what is the difference between validation and testing?