Machine Learning Overview

Introduction to Machine Learning

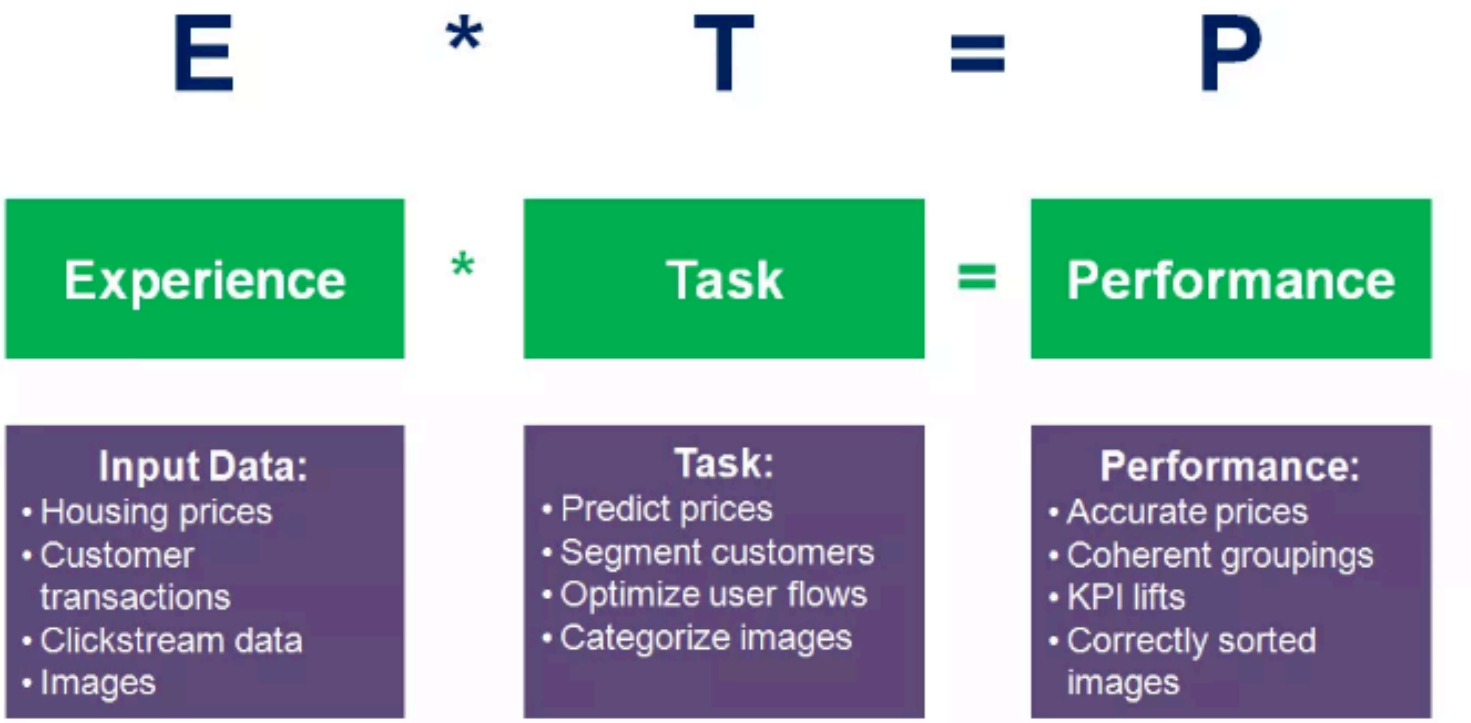
Machine Learning is…

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

Machine Learning is…

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

Tom Mitchell



Machine Learning is…

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

-- Ethem Alpaydin

The goal of machine learning is to develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest.

-- Kevin P. Murphy

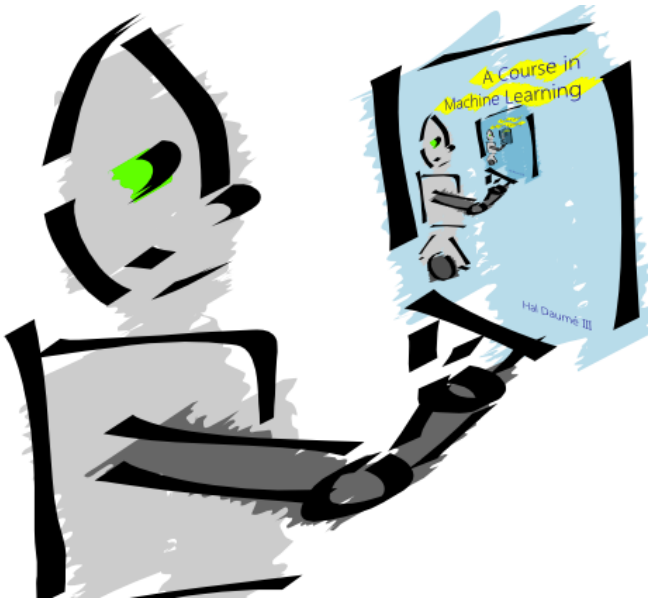
The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions.

-- Christopher M. Bishop

Machine Learning is…

Machine learning is about predicting the future based on the past.

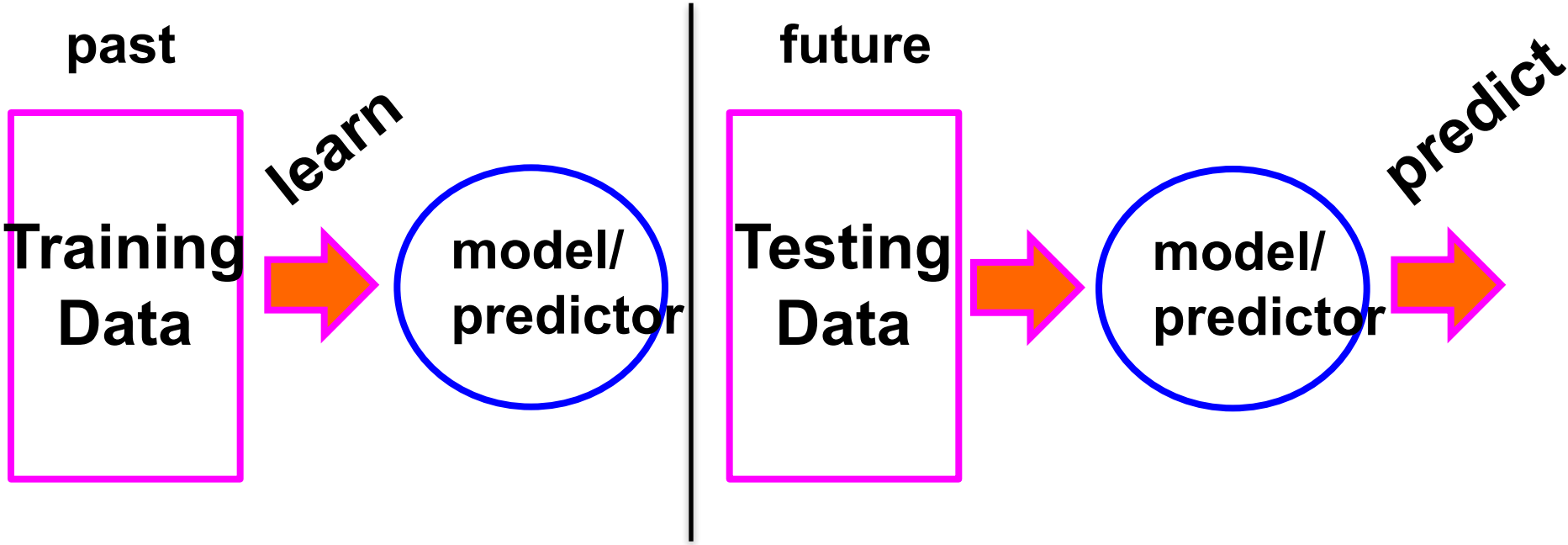
-- Hal Daume III



Machine Learning is…

Machine learning is about predicting the future based on the past.

-- Hal Daume III



The machine learning framework

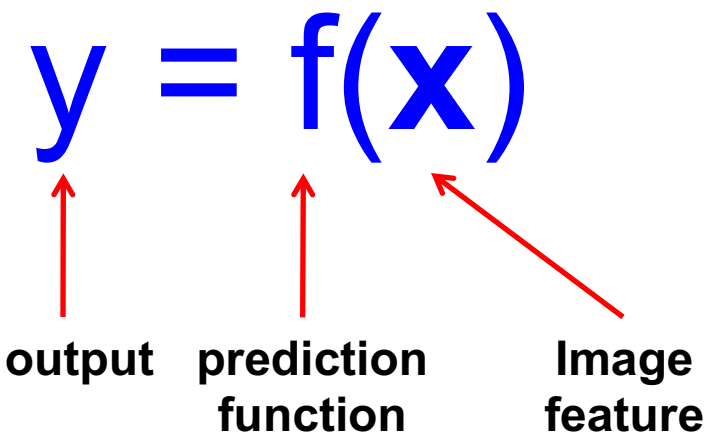
- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

The machine learning framework



- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* **x** and output the predicted value y = f(**x**)

Steps

Training

Training Images

Training Labels

Image Features → Training → Learned model

Testing

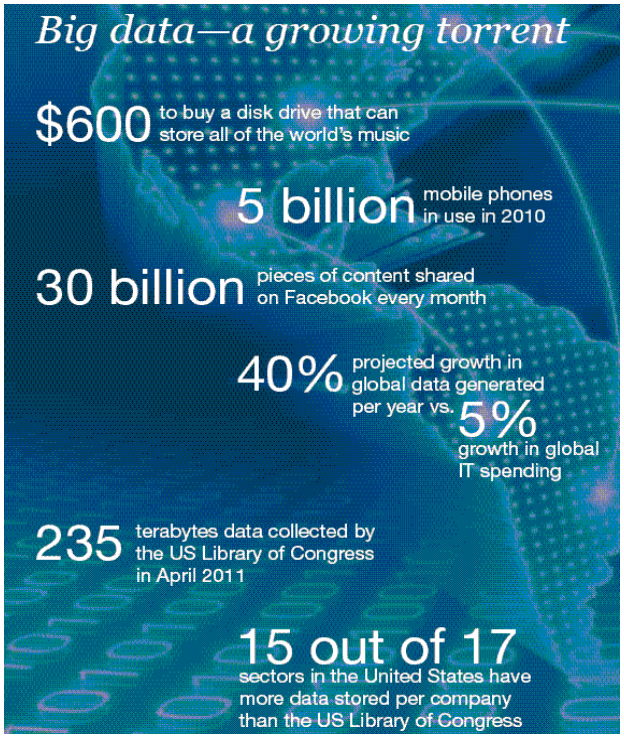
Test Image

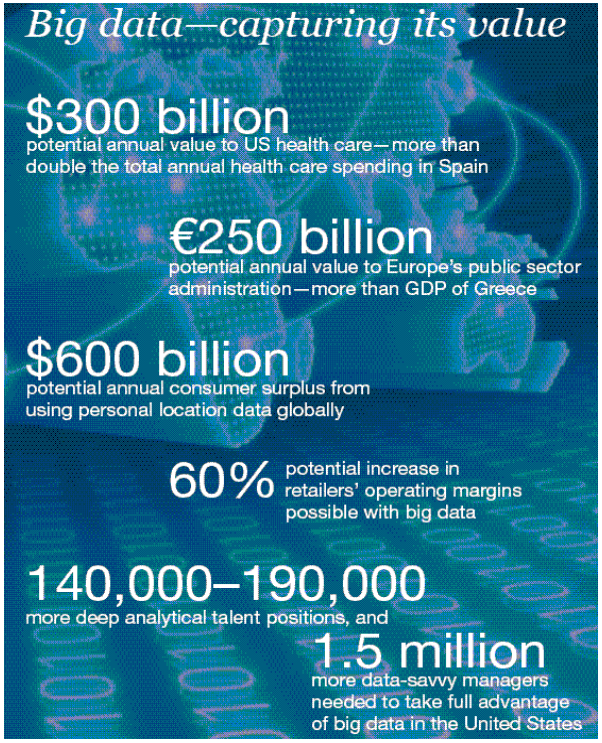
Image Features → Learned model → Prediction

Great opportunities to improve productivity in all walks of life

McKinsey Global Institute

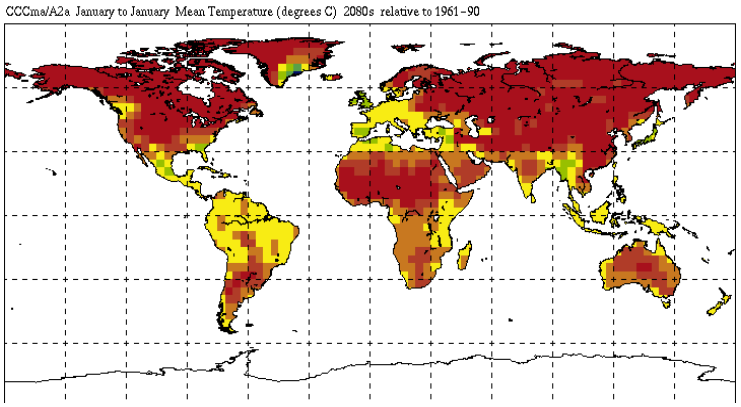
Big data: The next frontier for innovation, competition, and productivity





Great Opportunities to Solve Society's Major Problems

Improving health care and reducing costs



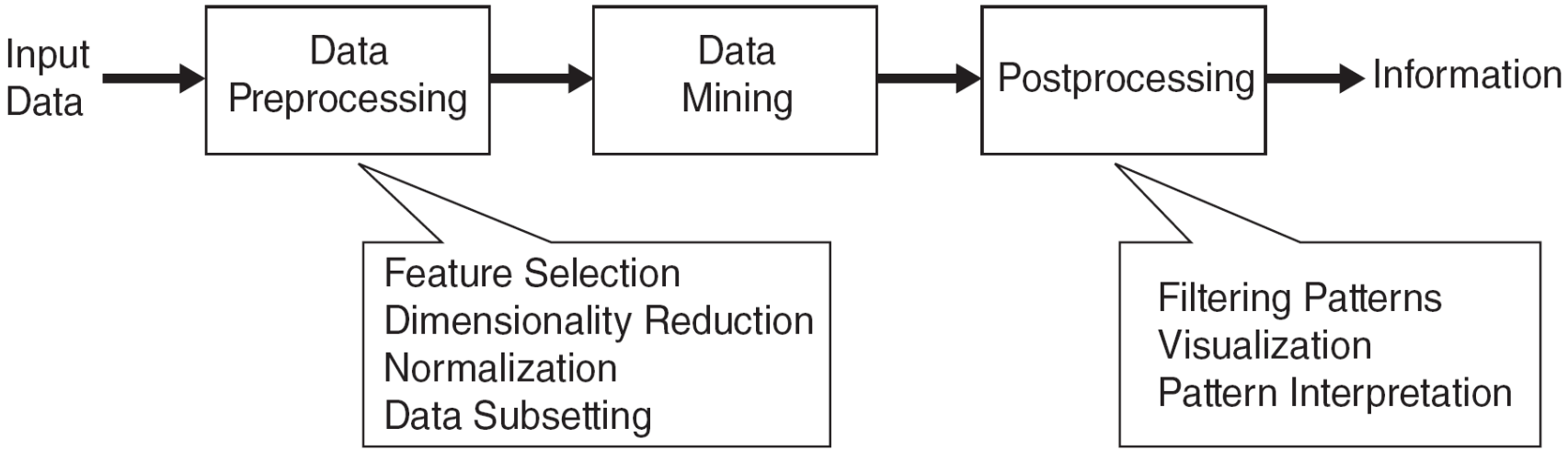
Predicting the impact of climate change

Finding alternative/ green energy sources

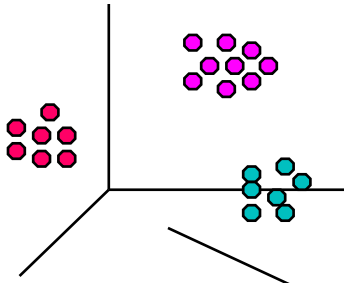
Reducing hunger and poverty by increasing agriculture production

What is Data Mining?

- Many Definitions
  - Non-trivial extraction of implicit, previously unknown and potentially useful information from data
  - Exploration & analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns



ML Tasks ...

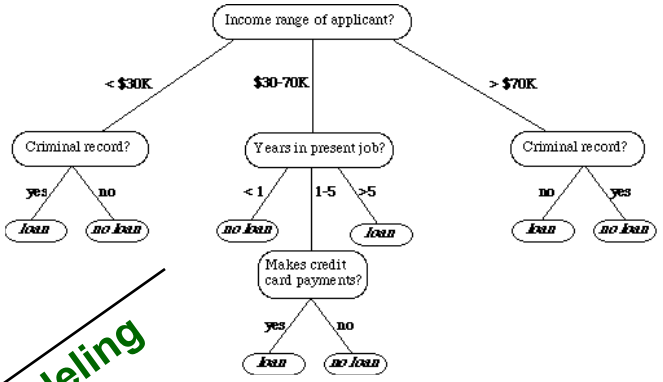


Clustering

Data

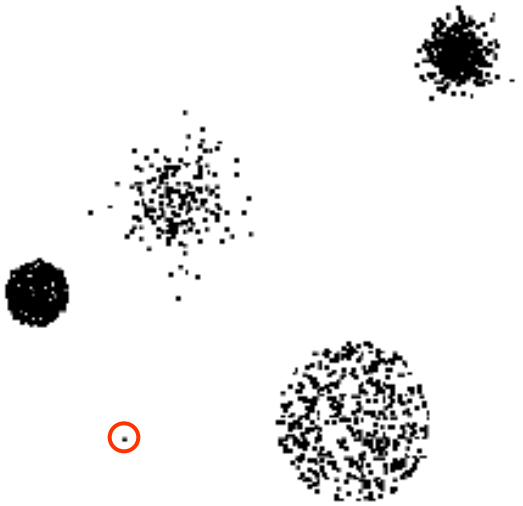
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes
11	No	Married	60K	No
12	Yes	Divorced	220K	No
13	No	Single	85K	Yes
14	No	Married	75K	No
15	No	Single	90K	Yes

Predictive Modeling



Association Rules

Anomaly Detection



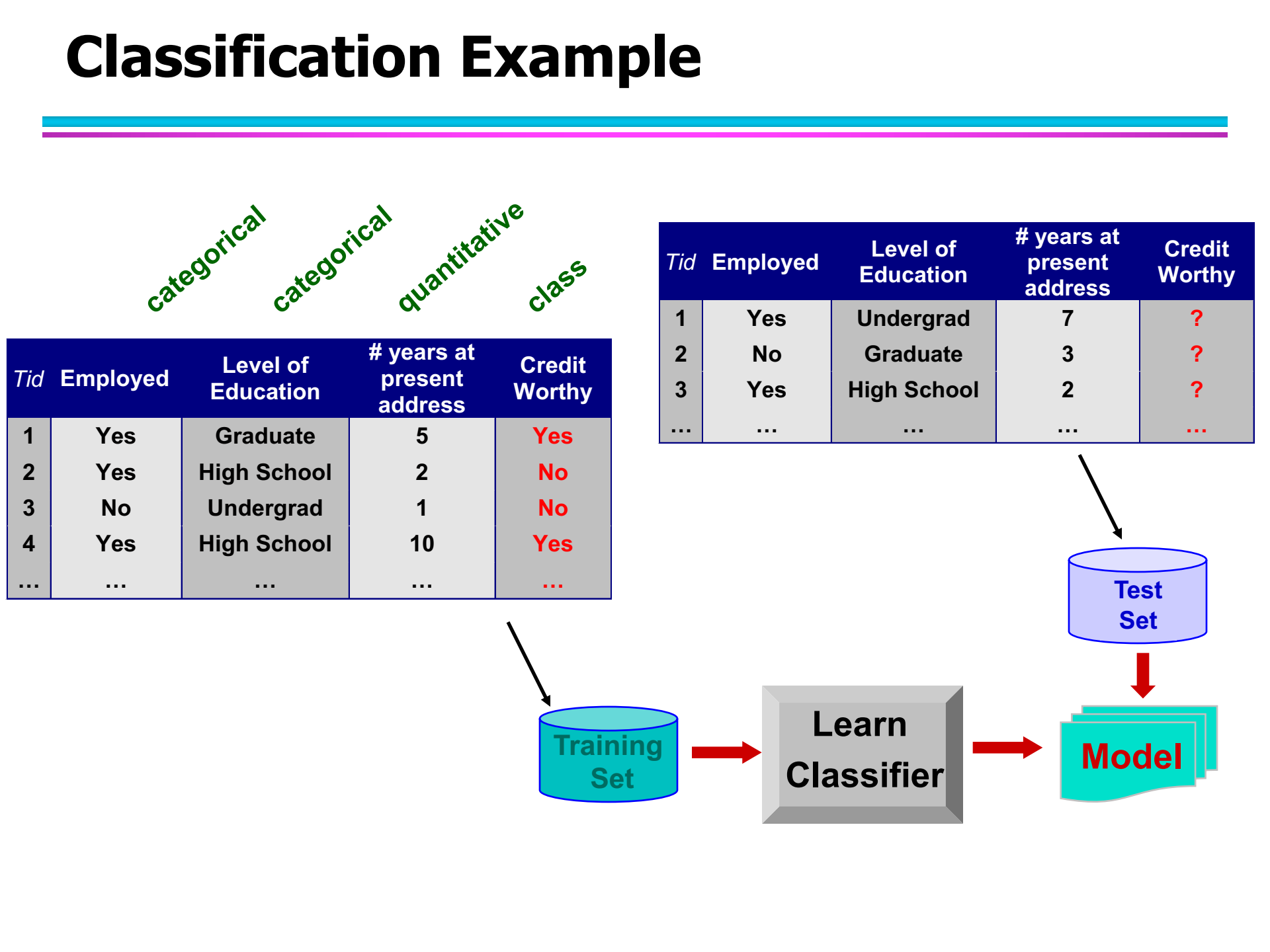
Classification

Classification Example

categorical categorical quantitative class

Tid	Employed	Level of Education	# years at present address	Credit Worthy
1	Yes	Graduate	5	Yes
2	Yes	High School	2	No
3	No	Undergrad	1	No
4	Yes	High School	10	Yes
…	…	…	…	…

Tid	Employed	Level of Education	# years at present address	Credit Worthy
1	Yes	Undergrad	7	?
2	No	Graduate	3	?
3	Yes	High School	2	?
…	…	…	…	…

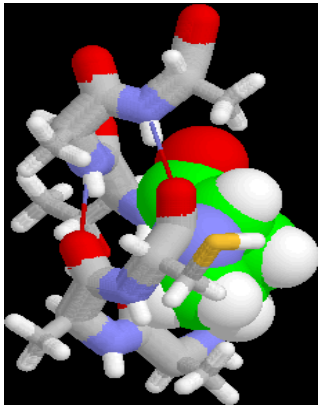


Examples of Classification Task

- Classifying credit card transactions as legitimate or fraudulent

- Classifying land covers (water bodies, urban areas, forests, etc.) using satellite data

- Categorizing news stories as finance, weather, entertainment, sports, etc
- Identifying intruders in the cyberspace
- Predicting tumor cells as benign or malignant



- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil

Classification: Application 1

- Fraud Detection
  - **Goal:** Predict fraudulent cases in credit card transactions.
  - **Approach:**
    - Use credit card transactions and the information on its account-holder as attributes.
      - When does a customer buy, what does he buy, how often he pays on time, etc
    - Label past transactions as fraud or fair transactions. This forms the class attribute.
    - Learn a model for the class of the transactions.
    - Use this model to detect fraud by observing credit card transactions on an account.

Classification: Application 2

- Churn prediction for telephone customers
  - **Goal:** To predict whether a customer is likely to be lost to a competitor.
  - **Approach:**
    - Use detailed record of transactions with each of the past and present customers, to find attributes.
      - How often the customer calls, where he calls, what time-of-the day he calls most, his financial status, marital status, etc.
    - Label the customers as loyal or disloyal.
    - Find a model for loyalty.

From [Berry & Linoff] Data Mining Techniques, 1997

Classification: Application 3

- Sky Survey Cataloging
  - **Goal:** To predict class (star or galaxy) of sky objects, especially visually faint ones, based on the telescopic survey images (from Palomar Observatory).
    - 3000 images with 23,040 x 23,040 pixels per image.
  - **Approach:**
    - Segment the image.
    - Measure image attributes (features) - 40 of them per object.
    - Model the class based on these features.
    - Success Story: Could find 16 new high red-shift quasars, some of the farthest objects that are difficult to find!

From [Fayyad, et.al.] Advances in Knowledge Discovery and Data Mining, 1996

Classifiers in MLlib

Linear models

- classification (SVMs, logistic regression)
- linear regression (least squares, Lasso, ridge)

Decision trees

Ensembles of decision trees

- random forests
- gradient-boosted trees

Naive Bayes

Isotonic regression

https://spark.apache.org/docs/latest/mllib-classification-regression.html

Mathematical Foundation

- Most classifiers can be expressed as finding the minimum of a convex objective function

$$f(\boldsymbol{w}) = \lambda R(\boldsymbol{w}) + \frac{1}{n}\sum_{i=1}^{n} L(\boldsymbol{w}; \boldsymbol{x_i}, y_i)$$

Where

$f(\boldsymbol{w})$ is the objective function

$L(\boldsymbol{w}; \boldsymbol{x_i}, y_i)$ is the loss function - difference between actual (y_i) and predicted value using set of feature vector $(\boldsymbol{x_i})$, using the choice of weights $(\boldsymbol{w})$

$\lambda R(\boldsymbol{w})$ regularization term that controls tradeoff between minimizing loss and model complexity.

Loss Functions

Types of Loss functions:

	loss function $L(\mathbf{w}; \mathbf{x}, y)$	gradient or sub-gradient
hinge loss	$\max\{0, 1 - y\mathbf{w}^T\mathbf{x}\}, \quad y \in \{-1, +1\}$	$\begin{cases} -y \cdot \mathbf{x} & \text{if } y\mathbf{w}^T\mathbf{x} < 1, \\ 0 & \text{otherwise.} \end{cases}$
logistic loss	$\log(1 + \exp(-y\mathbf{w}^T\mathbf{x})), \quad y \in \{-1, +1\}$	$-y\left(1 - \frac{1}{1+\exp(-y\mathbf{w}^T\mathbf{x})}\right) \cdot \mathbf{x}$
squared loss	$\frac{1}{2}(\mathbf{w}^T\mathbf{x} - y)^2, \quad y \in \mathbb{R}$	$(\mathbf{w}^T\mathbf{x} - y) \cdot \mathbf{x}$

Types of Regularizer

	regularizer $R(\mathbf{w})$	gradient or sub-gradient
zero (unregularized)	0	$\mathbf{0}$
L2	$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\mathbf{w}$
L1	$\|\mathbf{w}\|_1$	$\text{sign}(\mathbf{w})$
elastic net	$\alpha\|\mathbf{w}\|_1 + (1-\alpha)\frac{1}{2}\|\mathbf{w}\|_2^2$	$\alpha\text{sign}(\mathbf{w}) + (1-\alpha)\mathbf{w}$

Logistic Regression

Logistic Regression is a linear classifier with following loss function:

$$L(\mathbf{w}; \mathbf{x}, y) := \log(1 + \exp(-y\mathbf{w}^T\mathbf{x})).$$

Given a data point, classification is done by evaluating:

$$\mathrm{f}(z) = \frac{1}{1 + e^{-z}}$$

where $z = \boldsymbol{w}^T \boldsymbol{x}$

If f(z) > 0.5, assign class 0, otherwise class 1

More details at:

https://spark.apache.org/docs/2.2.0/mllib-linear-methods.html#logistic-regression

Logistic Regression in Spark MLlib

Easiest way to get started:
LogisticRegressionWithLBFGS
- limited memory BFGS algorithm

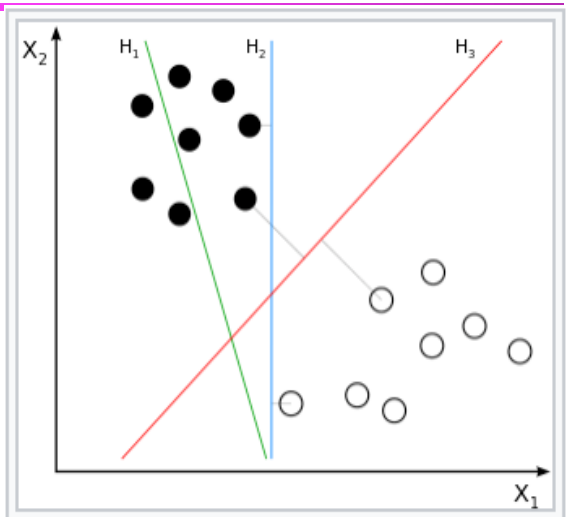
```
val model = new LogisticRegressionWithLBFGS()
  .setNumClasses(10)
  .run(training)
```

- just need to specify number of classes.
- should be tried first as base model
- More details at:

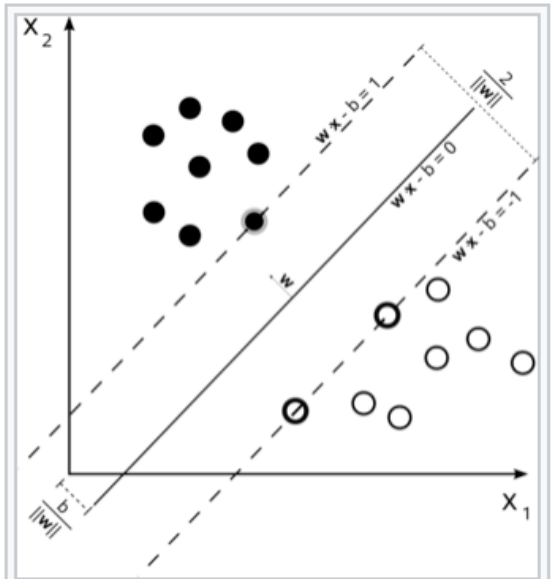
https://spark.apache.org/docs/2.2.1/api/java/org/apache/spark/mllib/classification/LogisticRegressionWithLBFGS.html

Support Vector Machine

Linear Case:
- We want to separate
  two classes
- Various possible choices



- SVM chooses one that provides maximum Margin of separation



SVM in Spark MLlib

```
public static SVMModel train(RDD<LabeledPoint> input,
                             int numIterations,
                             double stepSize,
                             double regParam)
```

Parameters:

input - RDD of (label, array of features) pairs.

stepSize - Step size to be used for each iteration of Gradient Descent.

regParam - Regularization parameter.

numIterations - Number of iterations of gradient descent to run.

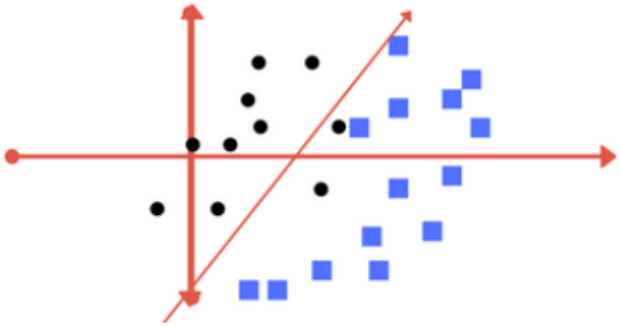
Returns:

a SVMModel which has the weights and offset from training.

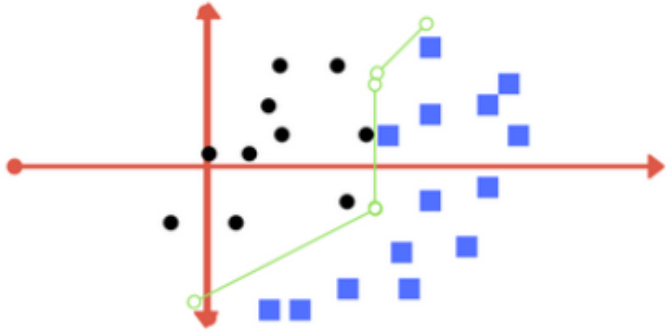
Note:

Labels used in SVM should be {0, 1}

SVM in Spark MLlib



Low Regularization Parameter
- model will tolerate error because the R part is already low, so L part will tolerate error.



High Regularization Parameter
- model will work very hard to avoid errors because R part is very high and we want to minimize L part

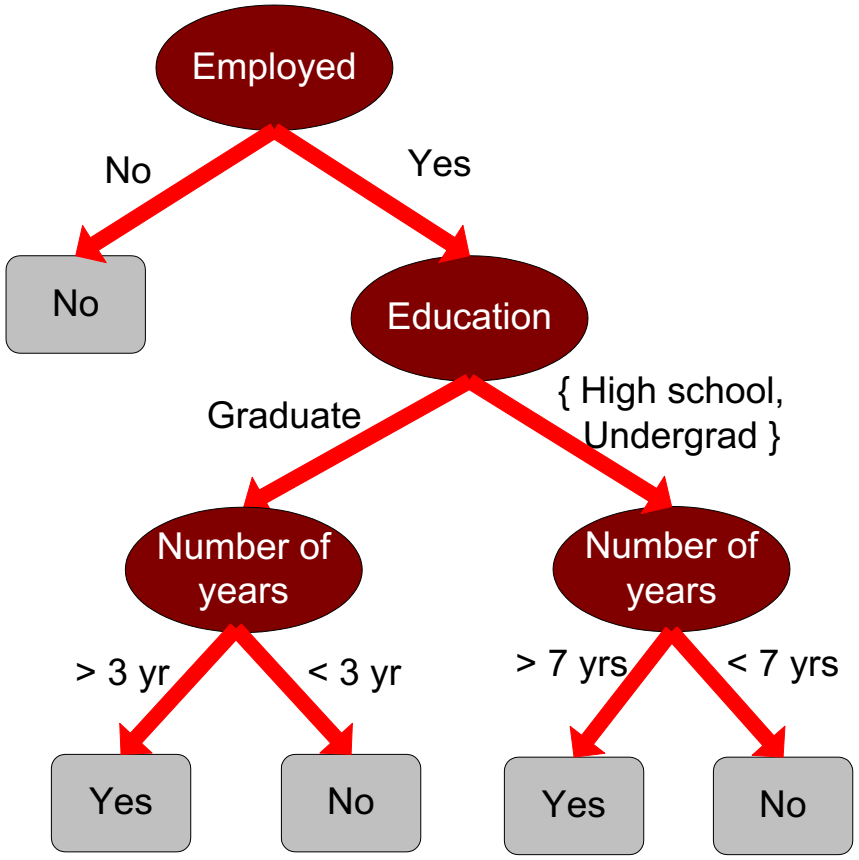
Decision Tree

- Find a model for class attribute as a function of the values of other attributes

Class

Tid	Employed	Level of Education	# years at present address	Credit Worthy
1	Yes	Graduate	5	Yes
2	Yes	High School	2	No
3	No	Undergrad	1	No
4	Yes	High School	10	Yes
…	…	…	…	…

Model for predicting credit worthiness



Decision Tree in Spark MLlib

```
public static DecisionTreeModel train(RDD<LabeledPoint> input,
                                       scala.Enumeration.Value algo,
                                       Impurity impurity,
                                       int maxDepth,
                                       int numClasses)
```

Parameters:

`input` - Training dataset: RDD of `LabeledPoint`. For classification, labels should take values {0, 1, ..., numClasses-1}. For regression, labels are real numbers.

`algo` - Type of decision tree, either classification or regression.

`impurity` - Criterion used for information gain calculation.

`maxDepth` - Maximum depth of the tree (e.g. depth 0 means 1 leaf node, depth 1 means 1 internal node + 2 leaf nodes).

`numClasses` - Number of classes for classification. Default value of 2.

More details at:
https://spark.apache.org/docs/latest/mllib-decision-tree.html

Decision Tree in Spark MLlib

```
public static DecisionTreeModel train(RDD<LabeledPoint> input,
                                       scala.Enumeration.Value algo,
                                       Impurity impurity,
                                       int maxDepth,
                                       int numClasses)
```

Parameters:

`input` - Training dataset: RDD of `LabeledPoint`. For classification, labels should take values {0, 1, ..., numClasses-1}. For regression, labels are real numbers.

`algo` - Type of decision tree, either classification or regression.

`impurity` - Criterion used for information gain calculation.

`maxDepth` - Maximum depth of the tree (e.g. depth 0 means 1 leaf node, depth 1 means 1 internal node + 2 leaf nodes).

`numClasses` - Number of classes for classification. Default value of 2.

More details at:
https://spark.apache.org/docs/latest/mllib-decision-tree.html

Combination of Trees

Random Forests(RF):

- Combine many decision trees in order to reduce the risk of overfitting.
- Train a set of decision trees separately, so the training can be done in parallel.
- Injects randomness into the training process so that each decision tree is a bit different.
- Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data.

More details at:
https://spark.apache.org/docs/latest/mllib-decision-tree.html

Combination of Trees

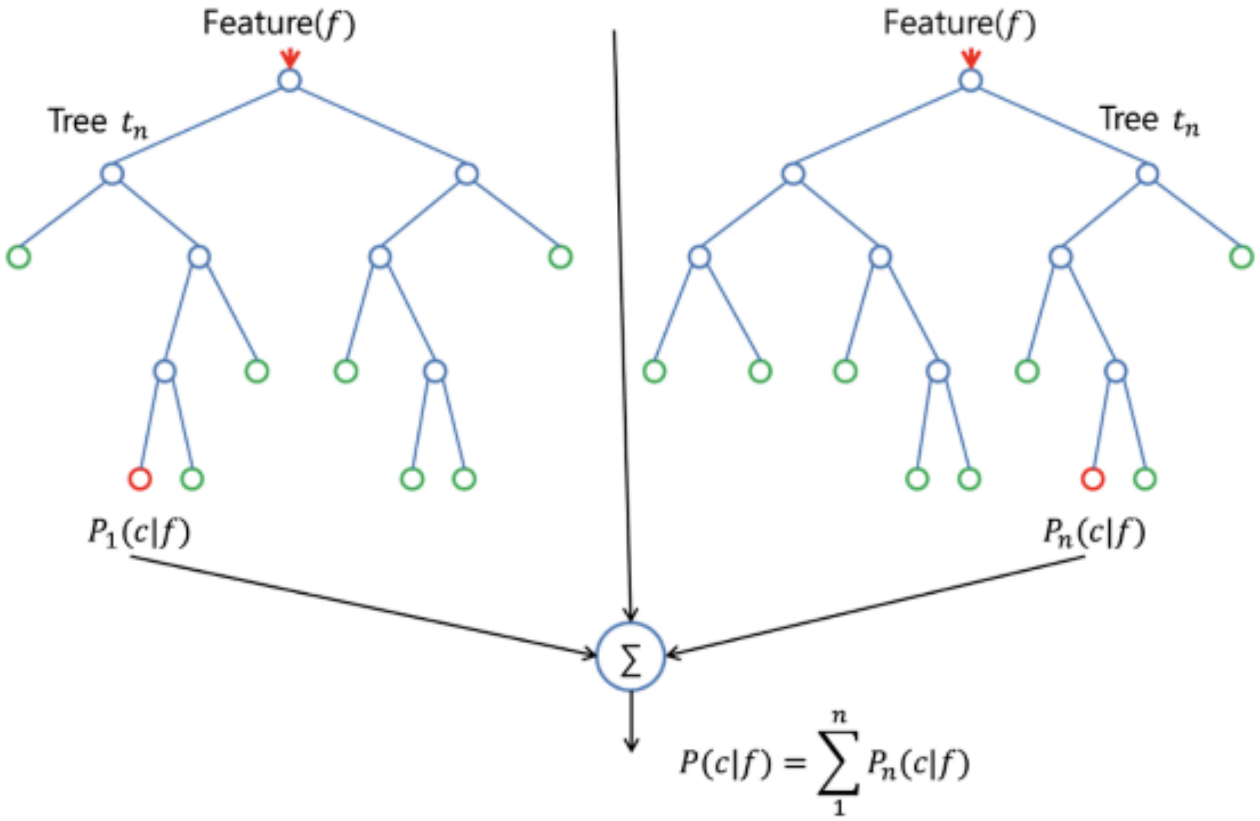
Random Forests(RF):

- Combine many decision trees in order to reduce the risk of overfitting.
- Train a set of decision trees separately, so the training can be done in parallel.
- Injects randomness into the training process so that each decision tree is a bit different.
- Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data.

More details at:
https://spark.apache.org/docs/latest/mllib-ensembles.html#random-forests

Combination of Trees

Random Forests(RF):



More details at:
https://spark.apache.org/docs/latest/mllib-ensembles.html#random-forests

RF in MLlib

```
public static RandomForestModel trainClassifier(RDD<LabeledPoint> input,
                                                 int numClasses,
                                                 scala.collection.immutable.Map<Object,Object> categoricalFeaturesInfo,
                                                 int numTrees,
                                                 String featureSubsetStrategy,
                                                 String impurity,
                                                 int maxDepth,
                                                 int maxBins,
                                                 int seed)
```

input - Training dataset: RDD of LabeledPoint. Labels should take values {0, 1, ..., numClasses-1}.
numClasses - Number of classes for classification.
categoricalFeaturesInfo - Map storing arity of categorical features.
numTrees - Number of trees in the random forest.
featureSubsetStrategy - Number of features to consider for splits at each node.
impurity - Criterion used for information gain calculation. Supported values: "gini" (recommended) or "entropy".
maxDepth - Maximum depth of the tree
maxBins - Maximum number of bins used for splitting features (suggested value: 100)
seed - Random seed for bootstrapping and choosing feature subsets.
Returns:
RandomForestModel that can be used for prediction.

Combination of Trees

Gradient Boosting Trees(GBT):

- Iteratively train decision trees in order to minimize a loss function.
- On each iteration, the algorithm uses the current ensemble to predict the label of each training instance
- Compares the prediction with the true label.
- The dataset is re-labeled to put more emphasis on training instances with poor predictions.

More details at:
https://spark.apache.org/docs/latest/mllib-ensembles.html#random-forests

GBT in Spark MLlib

```
public static GradientBoostedTreesModel train(RDD<LabeledPoint> input,
                                              BoostingStrategy boostingStrategy)
```

Method to train a gradient boosting model.

Parameters:

`input` - Training dataset: RDD of `LabeledPoint`. For classification, labels should take values {0, 1, ..., numClasses-1}. For regression, labels are real numbers.

`boostingStrategy` - Configuration options for the boosting algorithm.

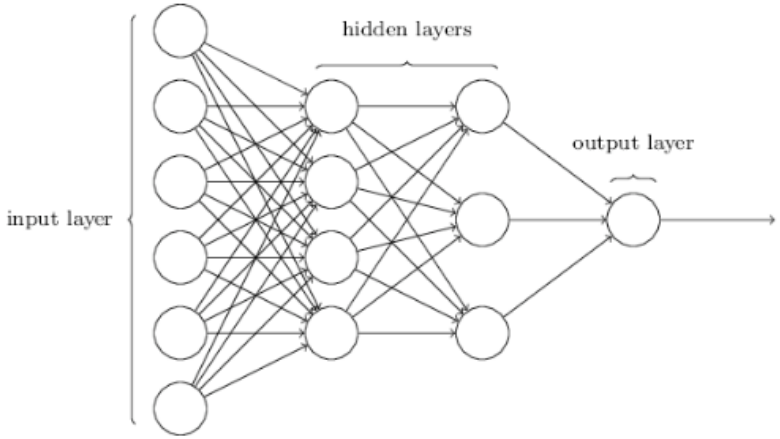
Returns:

GradientBoostedTreesModel that can be used for prediction.

More details at:
https://spark.apache.org/docs/latest/mllib-ensembles.html#gradient-boosted-trees-gbts

MultiLayer Perceptron

- Also known as neural network
- Various layers transform input to train the whole network to represent a function



More details at:
https://en.wikipedia.org/wiki/Multilayer_perceptron
https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

MultiLayer Perceptron in Spark MLlib

Parameter setters

- def **setFeaturesCol**(value: String): MultilayerPerceptronClassifier
- def **setLabelCol**(value: String): MultilayerPerceptronClassifier
- def **setLayers**(value: Array[Int]): MultilayerPerceptronClassifier.this.type
  Sets the value of param layers.
- def **setMaxIter**(value: Int): MultilayerPerceptronClassifier.this.type
  Set the maximum number of iterations.
- def **setPredictionCol**(value: String): MultilayerPerceptronClassifier
- def **setProbabilityCol**(value: String): MultilayerPerceptronClassifier
- def **setRawPredictionCol**(value: String): MultilayerPerceptronClassifier
- def **setSeed**(value: Long): MultilayerPerceptronClassifier.this.type
  Set the seed for weights initialization if weights are not set
- def **setStepSize**(value: Double): MultilayerPerceptronClassifier.this.type
  Sets the value of param stepSize (applicable only for solver "gd").
- def **setThresholds**(value: Array[Double]): MultilayerPerceptronClassifier
- def **setTol**(value: Double): MultilayerPerceptronClassifier.this.type
  Set the convergence tolerance of iterations.

More details at:
https://spark.apache.org/docs/2.3.1/api/scala/#org.apache.spark.ml.classification.MultilayerPerceptronClassifier
https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

naïve Bayes (NB) Classifier

- family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- Data point having features $\{x_1, x_2, \ldots, x_n\}$ is assigned class $\hat{y}$ among a set of classes $\{C_1, C_2, \ldots, C_n\}$ such that:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

More details at:
https://en.wikipedia.org/wiki/Naive_Bayes_classifier
https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

naïve Bayes in Spark MLlib

```
def train(input: RDD[LabeledPoint], lambda: Double, modelType: String): NaiveBayesModel
```

Trains a Naive Bayes model given an RDD of `(label, features)` pairs.

The model type can be set to either Multinomial NB (see here) or Bernoulli NB (see here). The Multinomial NB can handle discrete count data and can be called by setting the model type to "multinomial". For example, it can be used with word counts or TF_IDF vectors of documents. The Bernoulli model fits presence or absence (0-1) counts. By making every vector a 0-1 vector and setting the model type to "bernoulli", the fits and predicts as Bernoulli NB.

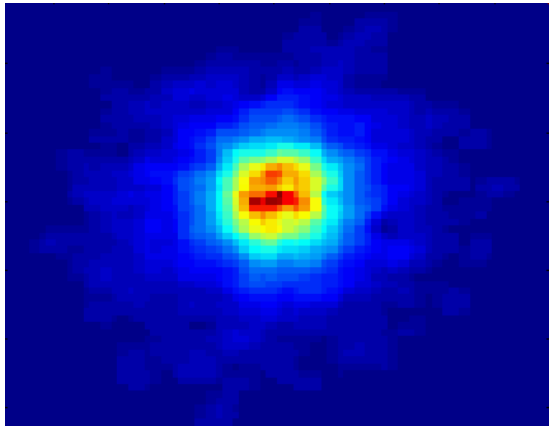
input	RDD of `(label, array of features)` pairs. Every vector should be a frequency vector or a count vector.
lambda	The smoothing parameter
modelType	The type of NB model to fit from the enumeration NaiveBayesModels, can be multinomial or bernoulli

More details at:
https://spark.apache.org/docs/2.2.0/api/scala/index.html#org.apache.spark.mllib.classification.NaiveBayes

Classifying Galaxies

Courtesy: http://aps.umn.edu

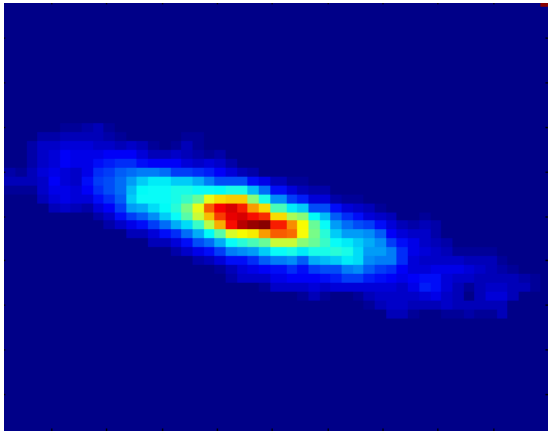
Early



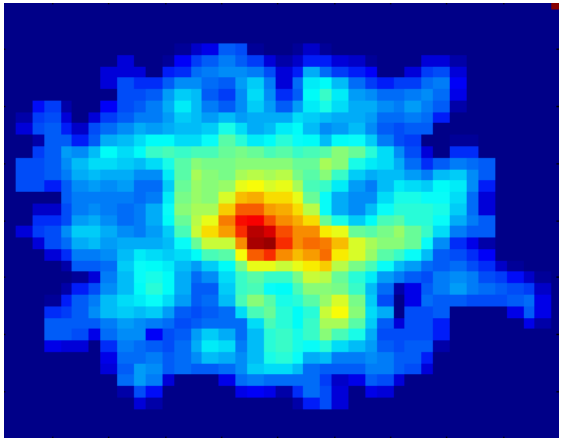
Class:
- **Stages of Formation**

Attributes:
- **Image features,**
- **Characteristics of light waves received, etc.**

Intermediate



Late



Data Size:
- **72 million stars, 20 million galaxies**
- **Object Catalog: 9 GB**
- **Image Database: 150 GB**

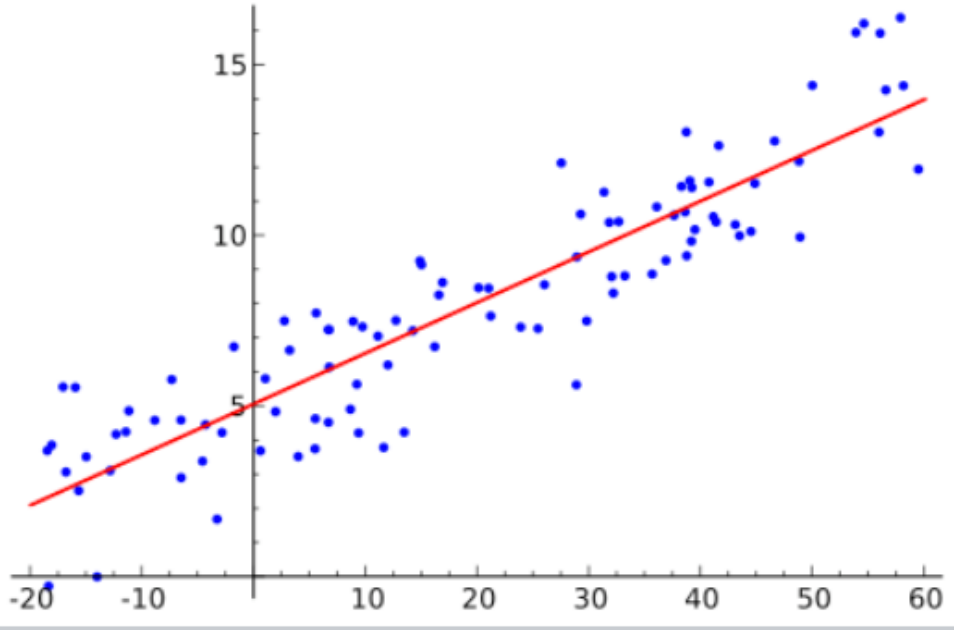
Regression

Regression

- Predict a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency.
- Extensively studied in statistics, neural network fields.
- Examples:
  - Predicting sales amounts of new product based on advetising expenditure.
  - Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
  - Time series prediction of stock market indices.

Regression

Linear regression tries to fit the best linear model to a set of data points and their continuous (real-valued) label.



Regression with Spark MLlib

Parameters

A list of (hyper-)parameter keys this algorithm can take. Users can set and get the parameter values through setters and getters, respectively.

- `final val` **`elasticNetParam:`** `DoubleParam`
  Param for the ElasticNet mixing parameter, in range [0, 1].
- `final val` **`featuresCol:`** `Param[String]`
  Param for features column name.
- `final val` **`fitIntercept:`** `BooleanParam`
  Param for whether to fit an intercept term.
- `final val` **`labelCol:`** `Param[String]`
  Param for label column name.
- `final val` **`loss:`** `Param[String]`
  The loss function to be optimized.
- `final val` **`maxIter:`** `IntParam`
  Param for maximum number of iterations (>= 0).
- `final val` **`predictionCol:`** `Param[String]`
  Param for prediction column name.
- `final val` **`regParam:`** `DoubleParam`
  Param for regularization parameter (>= 0).
- `final val` **`solver:`** `Param[String]`
  The solver algorithm for optimization.
- `final val` **`standardization:`** `BooleanParam`
  Param for whether to standardize the training features before fitting the model.
- `final val` **`tol:`** `DoubleParam`
  Param for the convergence tolerance for iterative algorithms (>= 0).
- `final val` **`weightCol:`** `Param[String]`
  Param for weight column name.

https://en.wikipedia.org/wiki/Elastic_net_regularization

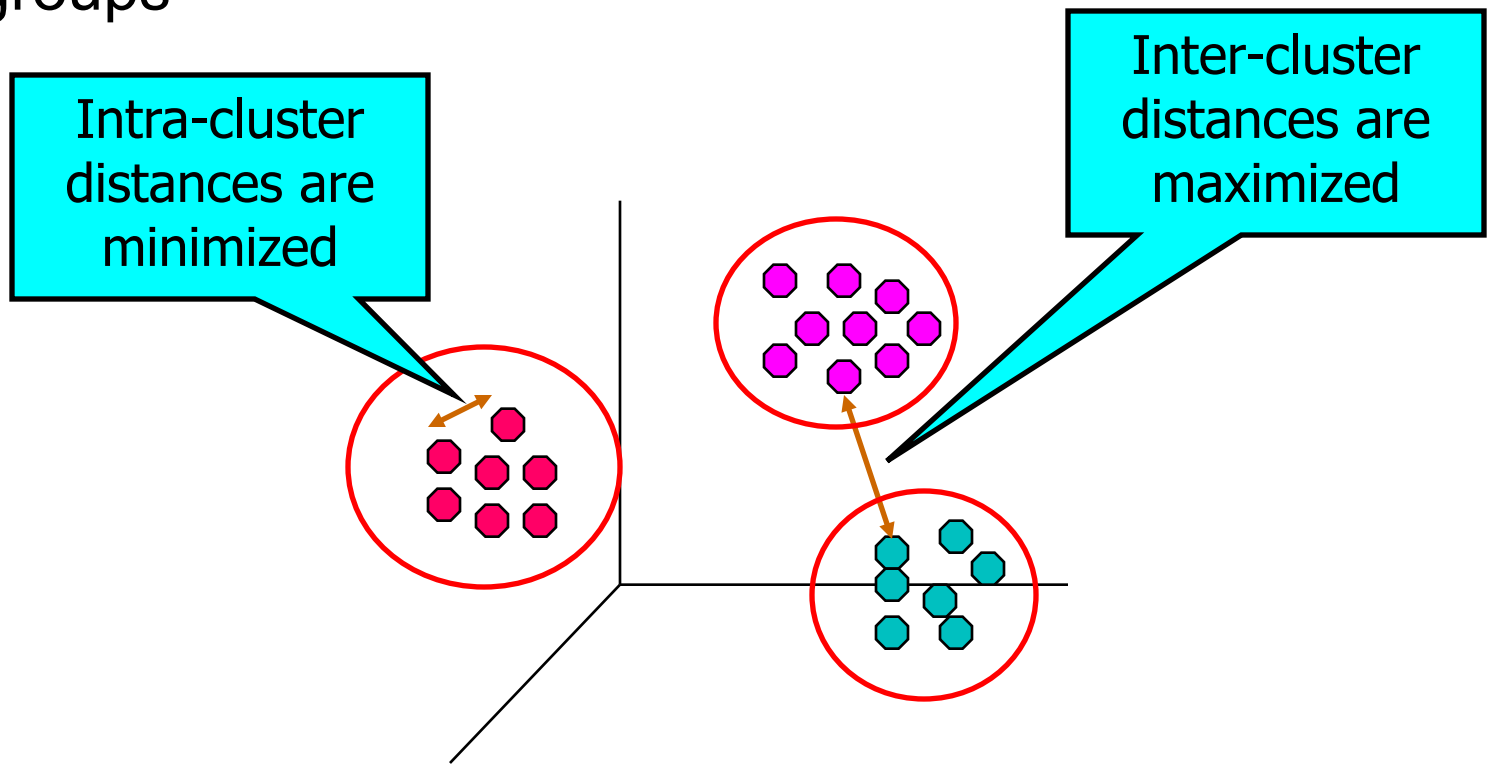
https://web.stanford.edu/~hastie/TALKS/enet_talk.pdf

More details at:
https://spark.apache.org/docs/latest/ml-classification-regression.html#linear-regression

Clustering

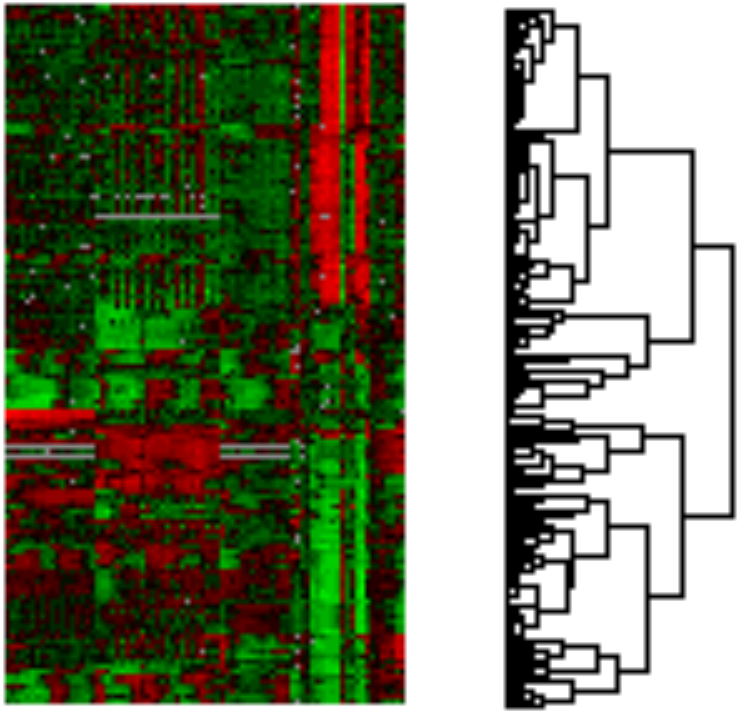
Clustering

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups

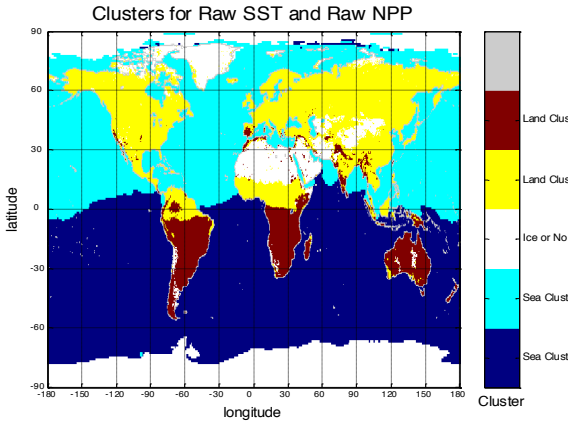


Applications of Cluster Analysis

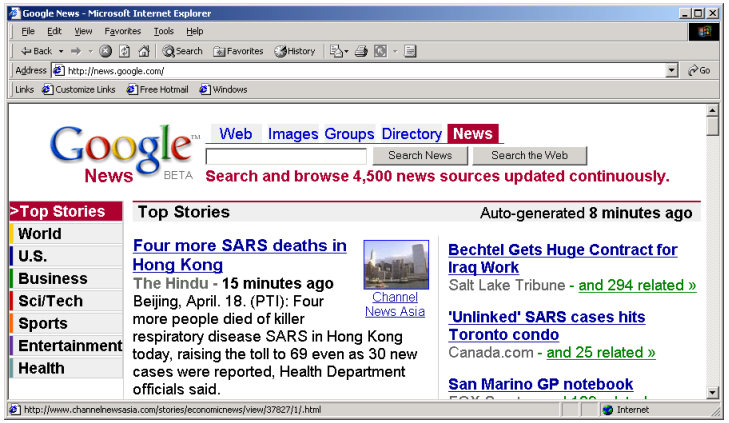
- **Understanding**
  - Custom profiling for targeted marketing
  - Group related documents for browsing
  - Group genes and proteins that have similar functionality
  - Group stocks with similar price fluctuations
- **Summarization**
  - Reduce the size of large data sets



Courtesy: Michael Eisen



Use of K-means to partition Sea Surface Temperature (SST) and Net Primary Production (NPP) into clusters that reflect the Northern and Southern Hemispheres.



Clustering: Application 1

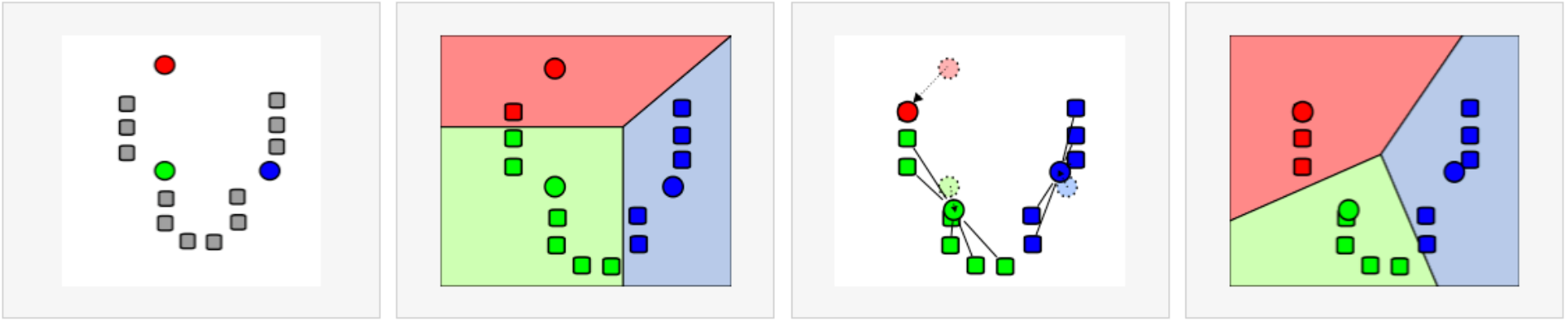
- Market Segmentation:
  - **Goal:** subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix.
  - **Approach:**
    - Collect different attributes of customers based on their geographical and lifestyle related information.
    - Find clusters of similar customers.
    - Measure the clustering quality by observing buying patterns of customers in same cluster vs. those from different clusters.

Clustering: Application 2

- Document Clustering:
  - **Goal:** To find groups of documents that are similar to each other based on the important terms appearing in them.
  - **Approach:** To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.

Enron email dataset

k-means Clustering



need to specify k (how many clusters)

k-means Clustering

```scala
import org.apache.spark.ml.clustering.KMeans
import org.apache.spark.mllib.linalg.Vectors

// Crates a DataFrame
val dataset: DataFrame = sqlContext.createDataFrame(Seq(
  (1, Vectors.dense(0.0, 0.0, 0.0)),
  (2, Vectors.dense(0.1, 0.1, 0.1)),
  (3, Vectors.dense(0.2, 0.2, 0.2)),
  (4, Vectors.dense(9.0, 9.0, 9.0)),
  (5, Vectors.dense(9.1, 9.1, 9.1)),
  (6, Vectors.dense(9.2, 9.2, 9.2))
)).toDF("id", "features")

// Trains a k-means model
val kmeans = new KMeans()
  .setK(2)
  .setFeaturesCol("features")
  .setPredictionCol("prediction")
val model = kmeans.fit(dataset)

// Shows the result
println("Final Centers: ")
model.clusterCenters.foreach(println)
```

Latent Dirichlet allocation

- Part of clustering library.
- Example Application: Topic discovery in a group of documents.
  - Can be a very interesting project topic
  - Read more at:
  https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
  https://en.wikipedia.org/wiki/Topic_model

Collaborative Filtering

Explicit Matrix Factorization Approach

Suppose you have following dataset :
(user, movie, review)

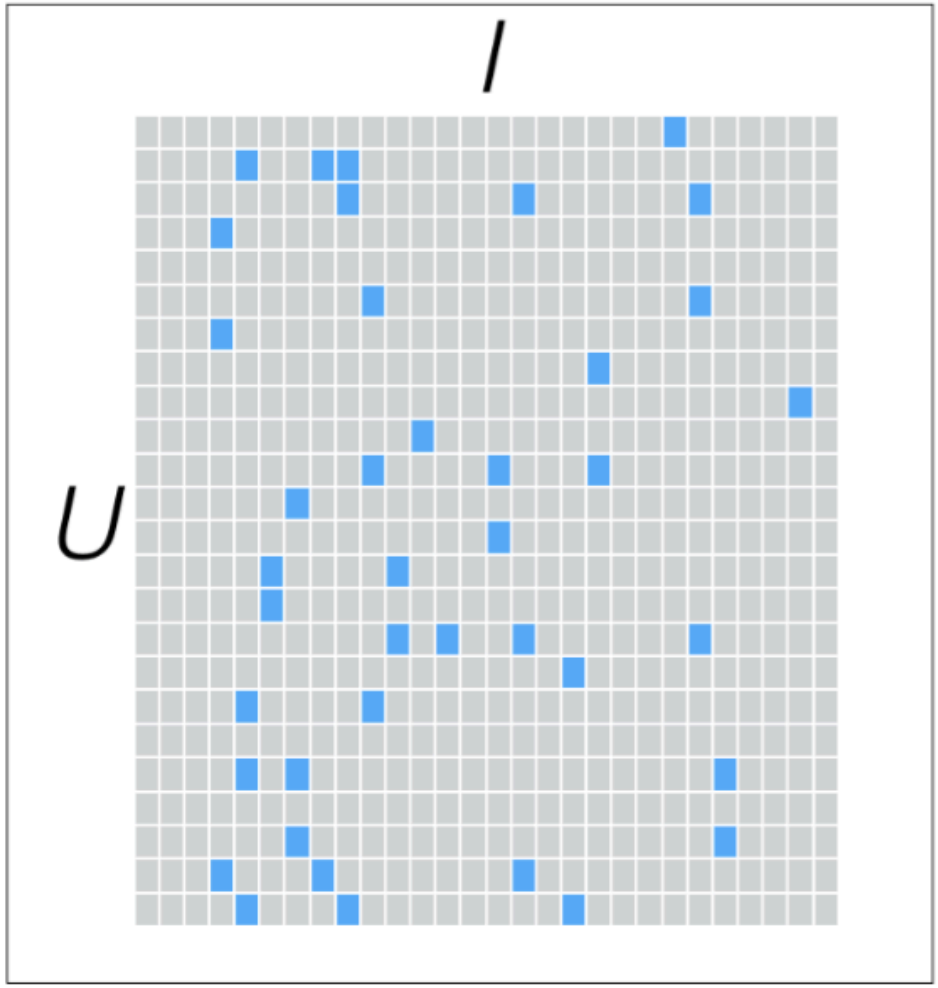
```
Tom, Star Wars, 5
Jane, Titanic, 4
Bill, Batman, 3
Jane, Star Wars, 2
Bill, Titanic, 3
```

It can be converted to a matrix:

User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

For large number of data points, this would be a sparse matrix

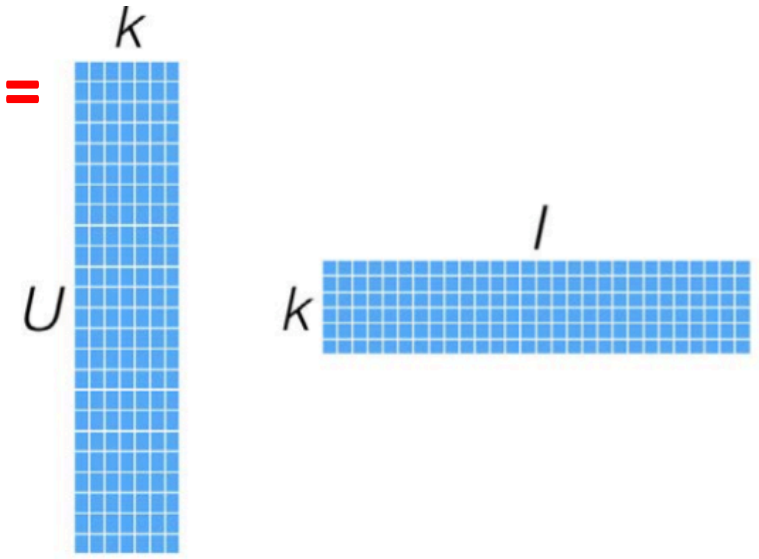
Factor Matrices



A sparse ratings matrix

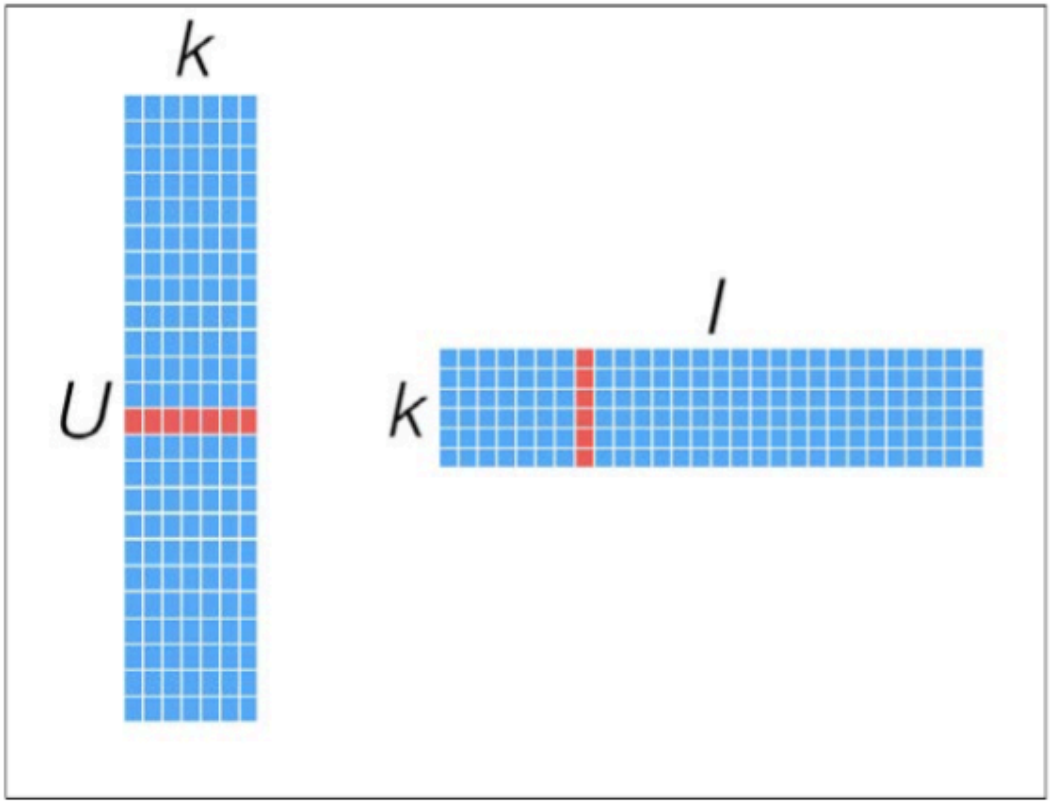
Do we really need to store it like this?

- How about dimensionality reduction.
- Representing it as a product of two smaller matrices (factor matrices)



Prediction using factor matrices

To find the prediction between a user and an item, simply take the dot product of corresponding row and column:



Computing recommendations from user- and item-factor vectors

Implicit Feedback

P

User / Item	Batman	Star Wars	Titanic
Bill	1	1	
Jane		1	1
Tom		1	

C

User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

Representation of an implicit preference and confidence matrix

You can still factorize the P matrix into users and items.
The C matrix indicates the "confidence" of the Preferences (P) matrix.

How is this achieved in Spark

- Spark has a library that implements ALS method for Matrix Factorization.
- See details at: http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html
- Good source of datasets for Recommender Systems: https://gist.github.com/entaroadun/1653794

Spark Collaborative Filtering

- Use ALS.train to train the model. Parameters are:

- `rank`: This refers to the number of factors in our ALS model, that is, the number of hidden features in our low-rank approximation matrices. Generally, the greater the number of factors, the better, but this has a direct impact on memory usage, both for computation and to store models for serving, particularly for large number of users or items. Hence, this is often a trade-off in real-world use cases. A rank in the range of 10 to 200 is usually reasonable.
- `iterations`: This refers to the number of iterations to run. While each iteration in `ALS` is guaranteed to decrease the reconstruction error of the ratings matrix, `ALS` models will converge to a reasonably good solution after relatively few iterations. So, we don't need to run for too many iterations in most cases (around 10 is often a good default).
- `lambda`: This parameter controls the regularization of our model. Thus, `lambda` controls over fitting. The higher the value of `lambda`, the more is the regularization applied. What constitutes a sensible value is very dependent on the size, nature, and sparsity of the underlying data, and as with almost all machine learning models, the regularization parameter is something that should be tuned using out-of-sample test data and cross-validation approaches.

Also look at trainImplicit method

https://databricks-training.s3.amazonaws.com/movie-recommendation-with-mllib.html

Dimensionality Reduction

Dimensionality Reduction

- Aim: Given a dataset with large number of features (dimensions), we wish to explain/summarize the underlying **variance-covariance structure** of variables through a **few linear combinations of these variables**.
- Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

Dimensionality Reduction

- Very useful technique in image processing, factor analysis, etc
- Spark has dimensionality reductions using SVD and PCA
- See details at:
  http://spark.apache.org/docs/latest/mllib-dimensionality-reduction.html

Frequent Pattern Mining

The Market-Basket Model

- A large set of *items*, e.g., things sold in a supermarket.
- A large set of *baskets*, each of which is a small set of the items, e.g., the things one customer buys on one day.

Support

- Simplest question: find sets of items that appear “frequently” in the baskets.
- *Support* for itemset *I* = the number of baskets containing all items in *I*.
  - Sometimes given as a percentage of the baskets.
- Given a *support threshold* *s*, a set of items appearing in at least *s* baskets is called a *frequent itemset*.

Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- Support = 3 baskets.

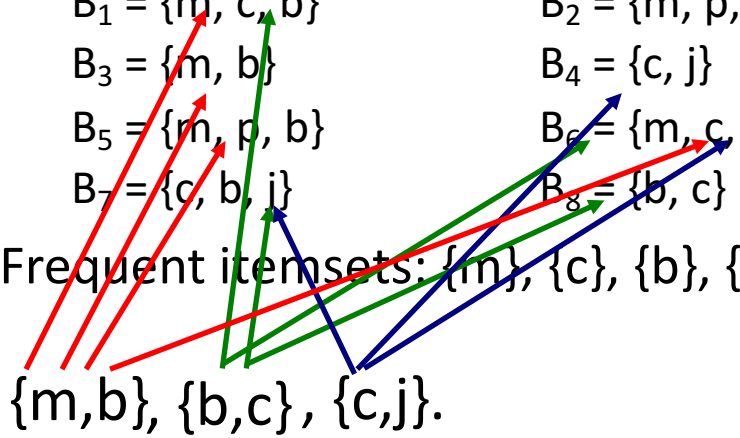
B_1 = {m, c, b} B_2 = {m, p, j}

B_3 = {m, b} B_4 = {c, j}

B_5 = {m, p, b} B_6 = {m, c, b, j}

B_7 = {c, b, j} B_8 = {b, c}

- Frequent itemsets: {m}, {c}, {b}, {j},



{m,b}, {b,c}, {c,j}.

Applications

- “Classic” application was analyzing what people bought together in a brick-and-mortar store.
  - Apocryphal story of “diapers and beer” discovery.
  - Used to position potato chips between diapers and beer to enhance sales of potato chips.
- Many other applications, including plagiarism detection; see MMDS.

Association Rules

- If-then rules about the contents of baskets.
- $\{i_1, i_2, \ldots, i_k\} \rightarrow j$ means: "if a basket contains all of $i_1, \ldots, i_k$ then it is *likely* to contain j."
- *Confidence* of this association rule is the probability of j given $i_1, \ldots, i_k$.
  - That is, the fraction of the baskets with $i_1, \ldots, i_k$ that also contain j.
- Generally want both high confidence and high support for the set of items involved.

Example: Confidence

+	B_1 = {m, c, b}	B_2 = {m, p, j}
	B_3 = {m, b}	B_4 = {c, j}
−	B_5 = {m, p, b}	B_6 = {m, c, b, j}
−	B_7 = {c, b, j}	+ B_8 = {b, c}

- An association rule: {m, b} → c.
  - Confidence = 2/4 = 50%.

Frequent Pattern using Spark

- Spark has built-in implementation for the FP Growth algorithm.
- Details are at: http://spark.apache.org/docs/latest/mllib-frequent-pattern-mining.html

Association Rule Discovery: Definition

- Given a set of records each of which contain some number of items from a given collection
  - Produce dependency rules which will predict occurrence of an item based on occurrences of other items.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:

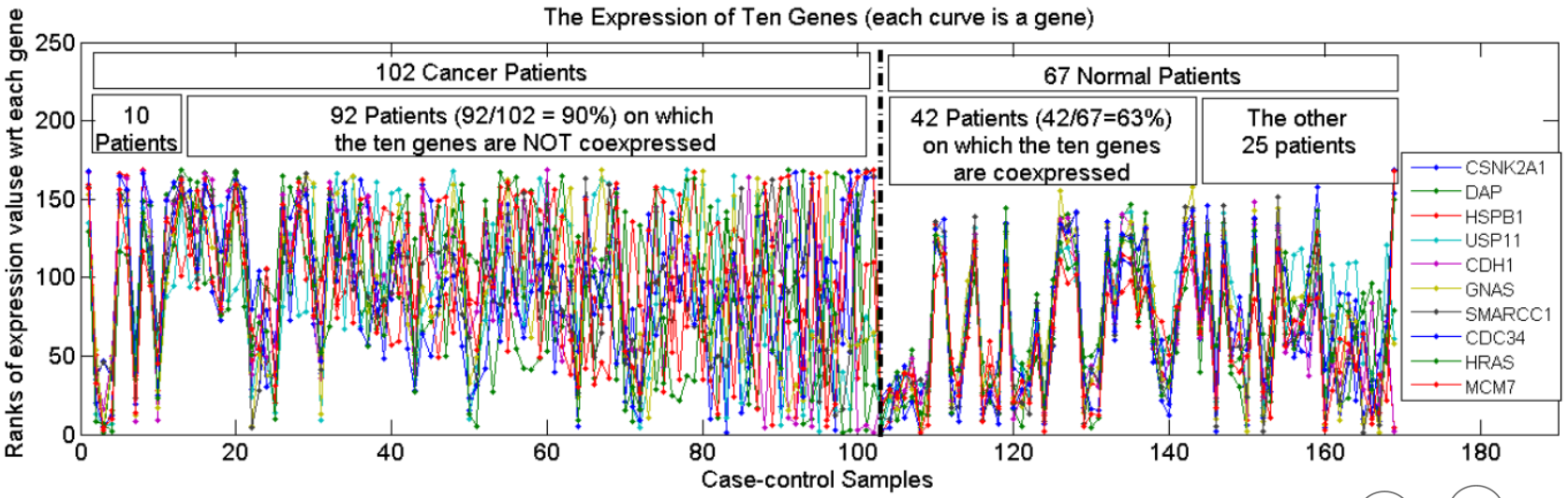
{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

Association Analysis: Applications

- An Example Subspace Differential Coexpression Pattern from lung cancer dataset

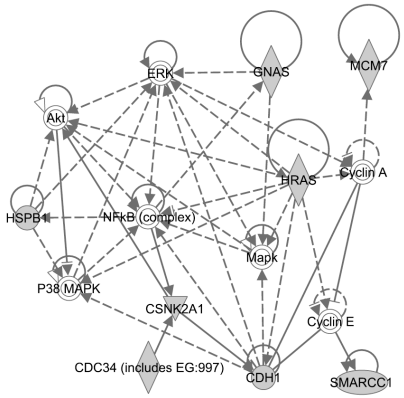
Three lung cancer datasets [Bhattacharjee et a 2001], [Stearman et al. 2005], [Su et al. 2007]



Enriched with the TNF/NFB signaling pathway

which is well-known to be related to lung cancer

P-value: $1.4*10^{-5}$ (6/10 overlap with the pathway)



[Fang et al PSB 2010]

Association Analysis: Applications

- Market-basket analysis
  - Rules are used for sales promotion, shelf management, and inventory management
- Telecommunication alarm diagnosis
  - Rules are used to find combination of alarms that occur together frequently in the same time period
- Medical Informatics
  - Rules are used to find combination of patient symptoms and test results associated with certain diseases

Evaluation Metrics

Evaluation Metrics

- For ML projects, it is definitely a good idea to present evaluation metrics.
- Details are at: http://spark.apache.org/docs/latest/mllib-evaluation-metrics.html

Motivating Challenges

- Scalability
- High Dimensionality
- Heterogeneous and Complex Data
- Data Ownership and Distribution
- Non-traditional Analysis