

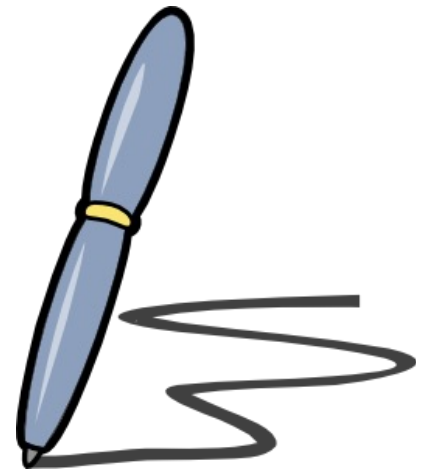
CS 444: Big Data Systems

Chapter 6. Big Data Analytics

Chase Wu

Outline

- An Introduction
 - Data, Information, Knowledge, and Wisdom
 - Artificial Intelligence, Machine Learning, Data Mining
 - Questions Data Science Can Answer
 - When to Use Machine Learning
- Big Data Analytics Algorithms
 - Recommendation
 - Clustering
 - Classification
 - Regression



Data, Information, Knowledge, and Wisdom

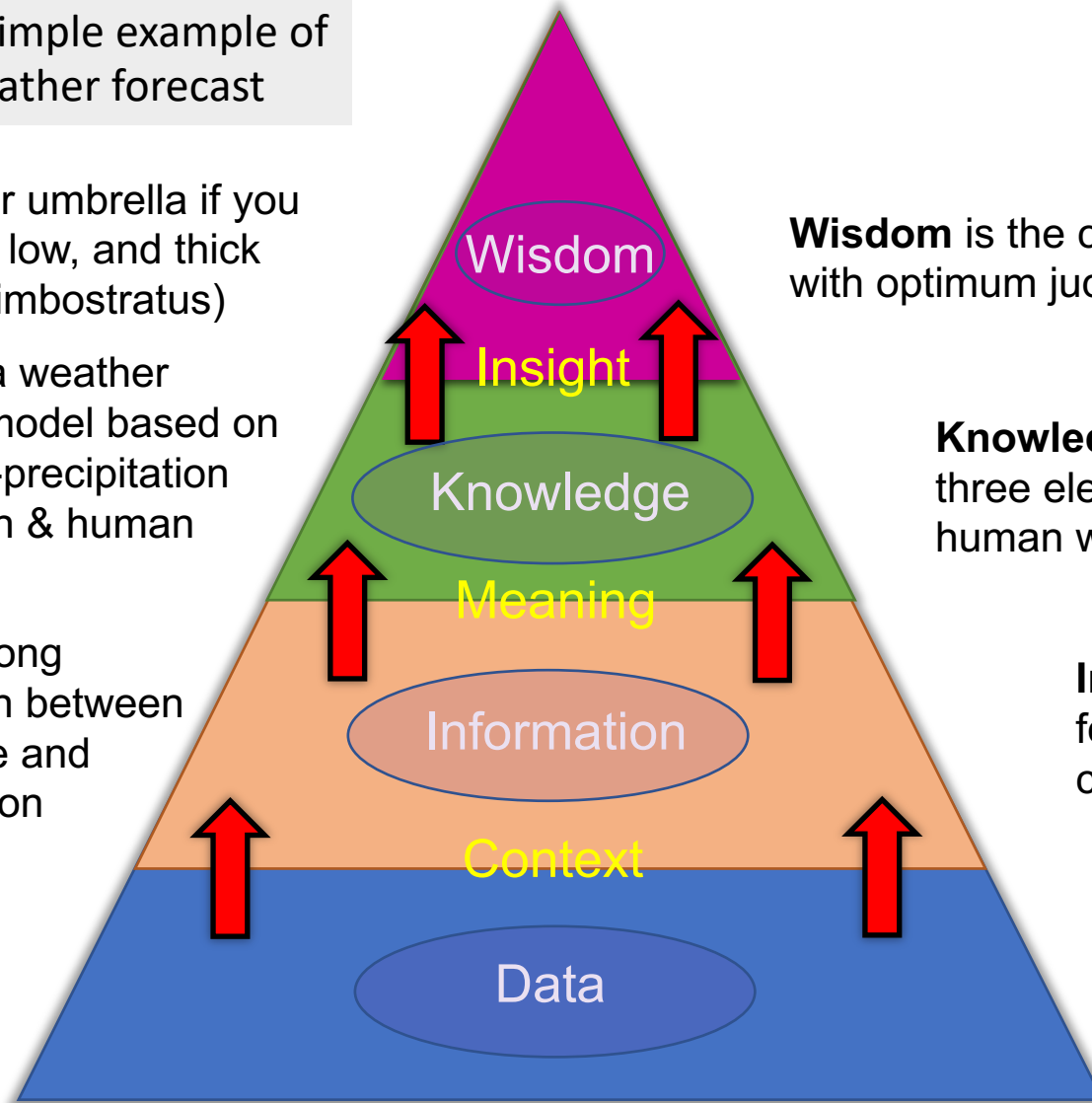
A simple example of weather forecast

Bring your umbrella if you see dark, low, and thick clouds (nimbostratus)

Develop a weather forecast model based on the cloud-precipitation correlation & human insights

Find a strong correlation between cloud type and precipitation

Collect historical weather data



Wisdom is the comprehension of what is true coupled with optimum judgement as to action taking

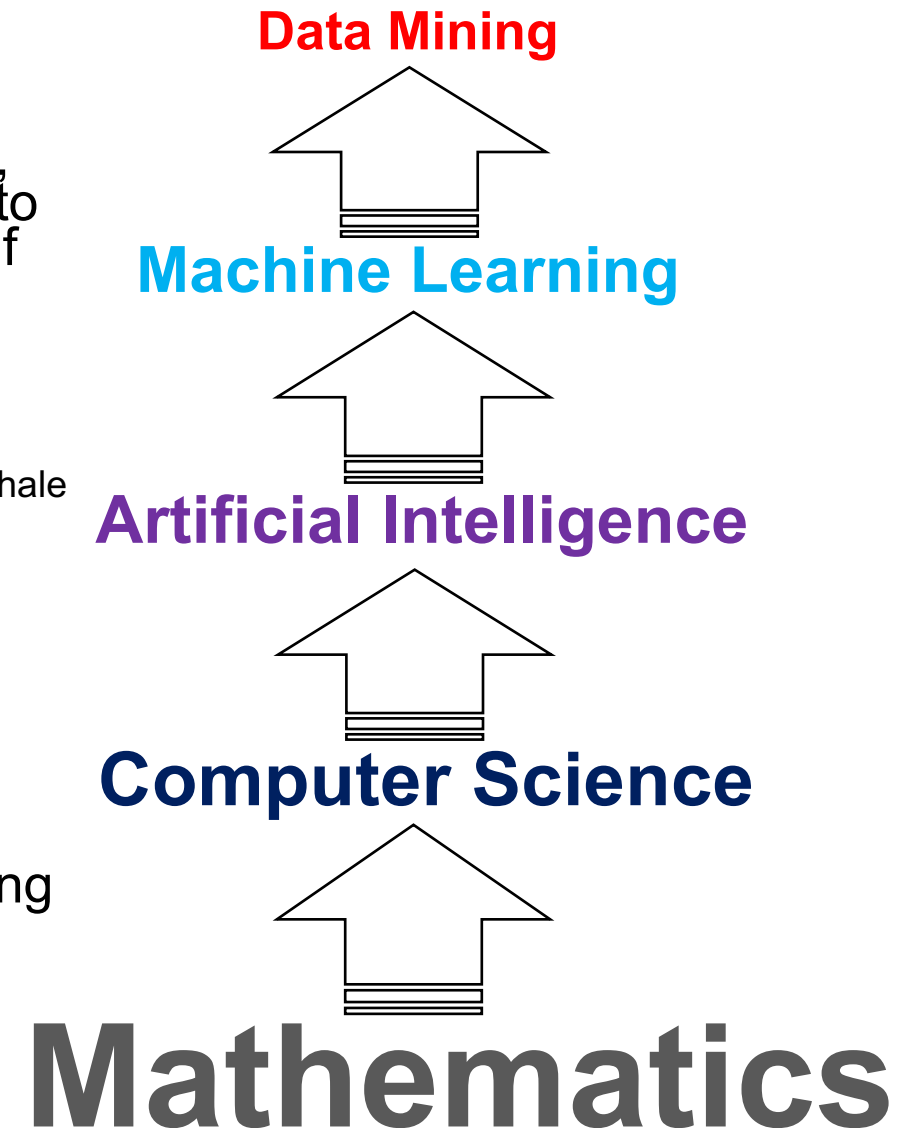
Knowledge is the result of the qualitative fusion of three elements; information, experience, and human wisdom

Information is the result of data analysis for the purpose of finding relations, indicators, correlations, upon which a decision can be made

Data are typically the results of measurements, stored/managed in different forms and structures, and can be *visualized* using *graphs* or *images*

Artificial Intelligence, Machine Learning, and Data Mining

- There is no common agreement
- Artificial intelligence (AI), also known as machine intelligence, focuses on building and managing technology that can learn to autonomously make decisions and carry out actions on behalf of a human being
 - Data-driven machine learning
 - Bio- or physics-inspired (randomization-based, probabilistic) algorithms
 - Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization, Whale Optimization Algorithm, Simulated Annealing, Spin Glass, etc.
- Machine learning, as one type of AI technology, focuses on designing algorithms that can learn from historical data and make predictions
- Data mining is a cross-disciplinary field that aims at discovering properties (useful information) of data sets
- Machine learning can be used for data mining



Software is becoming increasingly intelligent

Example: Computer Vision (CV) has surpassed human abilities

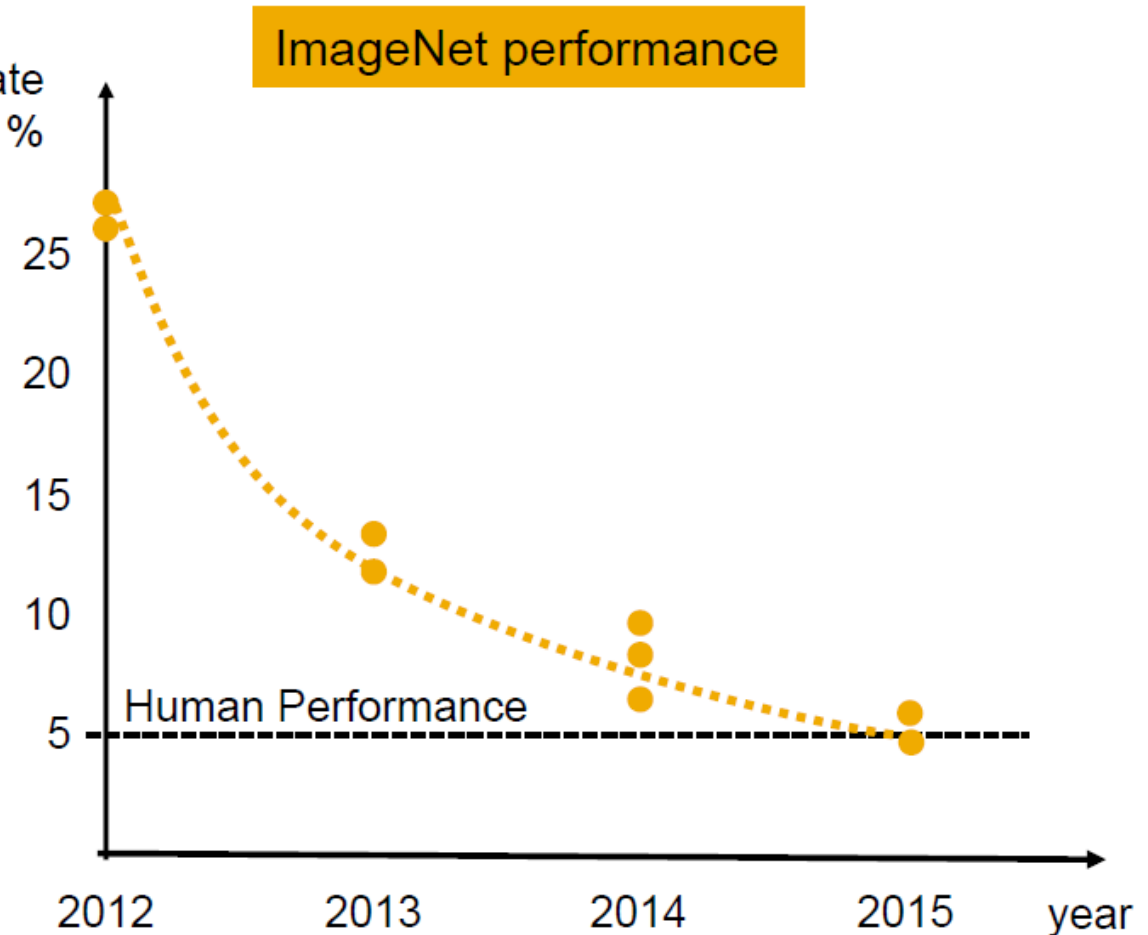


- Chair
- Dining table
- Person



- Dog
- Person
- Leaf

Error Rate
in %



ChatGPT



Examples

"Explain quantum computing in simple terms"

"Got any creative ideas for a 10 year old's birthday?"

"How do I make an HTTP request in Javascript?"



Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests



Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge of world and events after 2021

Machine learning example applications



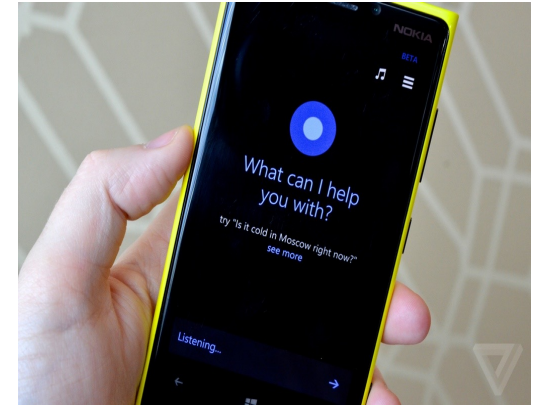
Self-Driving



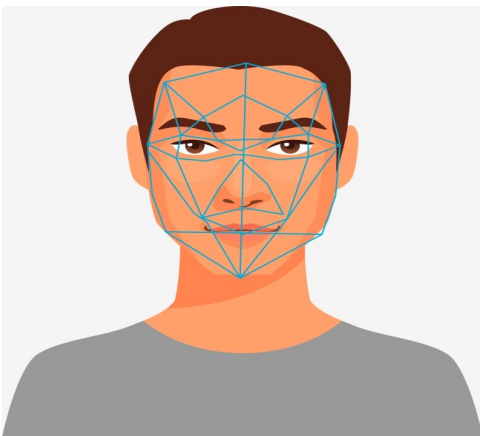
Spam Detection



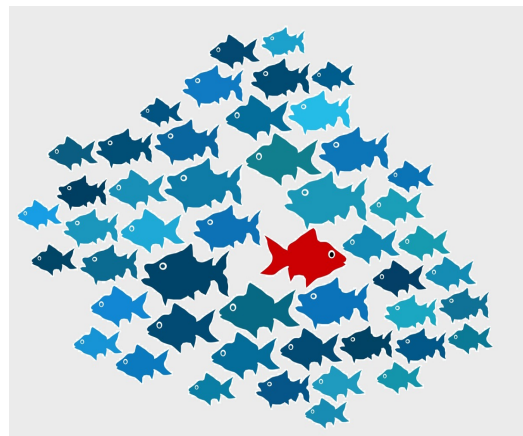
Fraud Detection



Voice Recognition



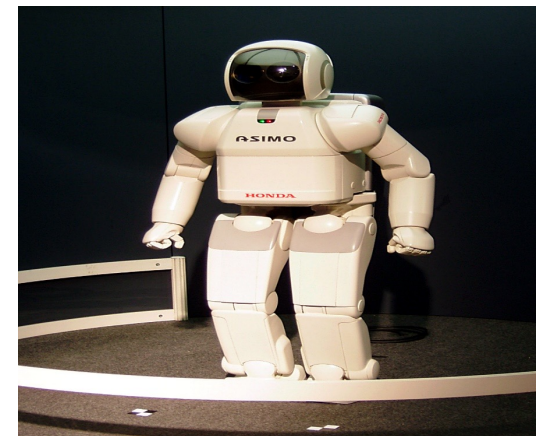
Face Recognition



Anomaly Detection



Sales Forecast



Robotics

Use predictive analytics to solve a variety of business challenges



- Churn reduction
- Customer acquisition
- Lead scoring
- **Product recommendation**
- Campaign optimization
- Customer segmentation
- Next best offer/action



- Predictive maintenance
- Load forecasting
- Inventory/Demand optimization
- **Price optimization**
- Manufacturing process optimization
- Quality management
- Yield management



- **Fraud and abuse detection**
- Collection and delinquency
- Credit scoring
- Operation risk modeling
- Crime threat analysis
- Revenue and loss analysis



- Cash flow and forecasting
- Budget simulation
- Profitability and margin analysis
- Financial risk modeling
- **Employee retention modeling**
- Succession planning



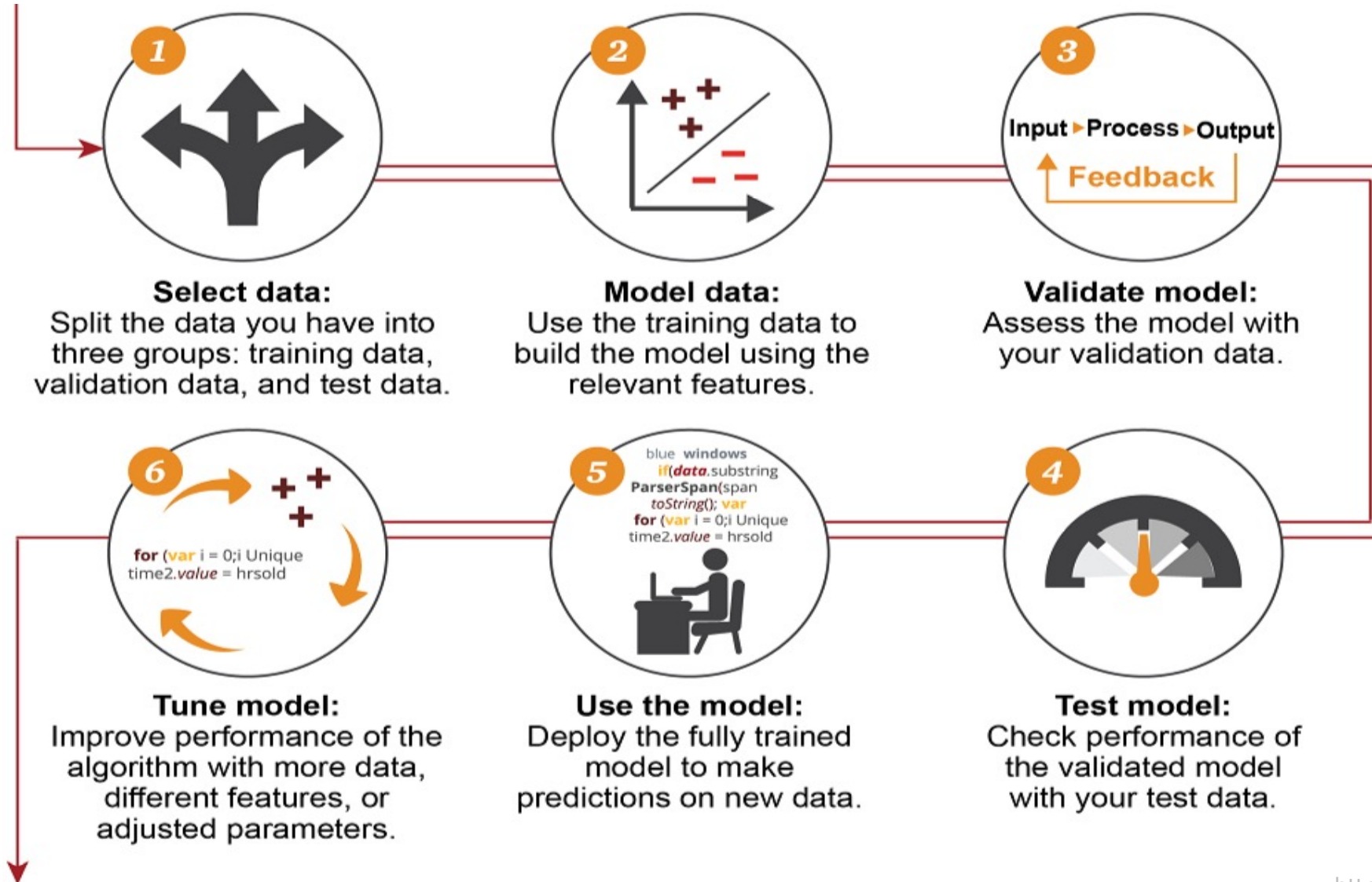
- Life sciences
- Healthcare
- Media
- Higher education
- Public section
- Social sciences
- Construction and mining
- **Travel and hospitality**
- Big data and IoT

What is Machine Learning?

- Machine learning is the science of getting computers to act without being *explicitly programmed*
- Machine learning is a technique of data science that helps computers *learn from existing data* in order to *forecast future behaviors, outcomes, and trends*



How machine learning works



An example of machine learning task

Let's say that you are in the car rental business.

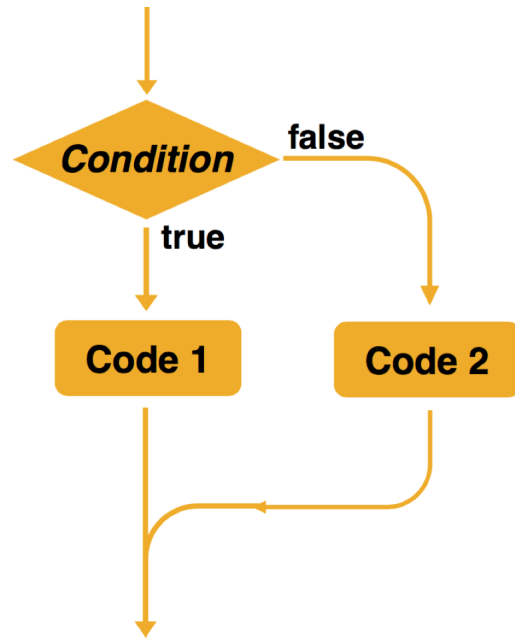
How can you accurately predict demands for different types of cars at different times?



Difference between traditional approach and machine learning approach

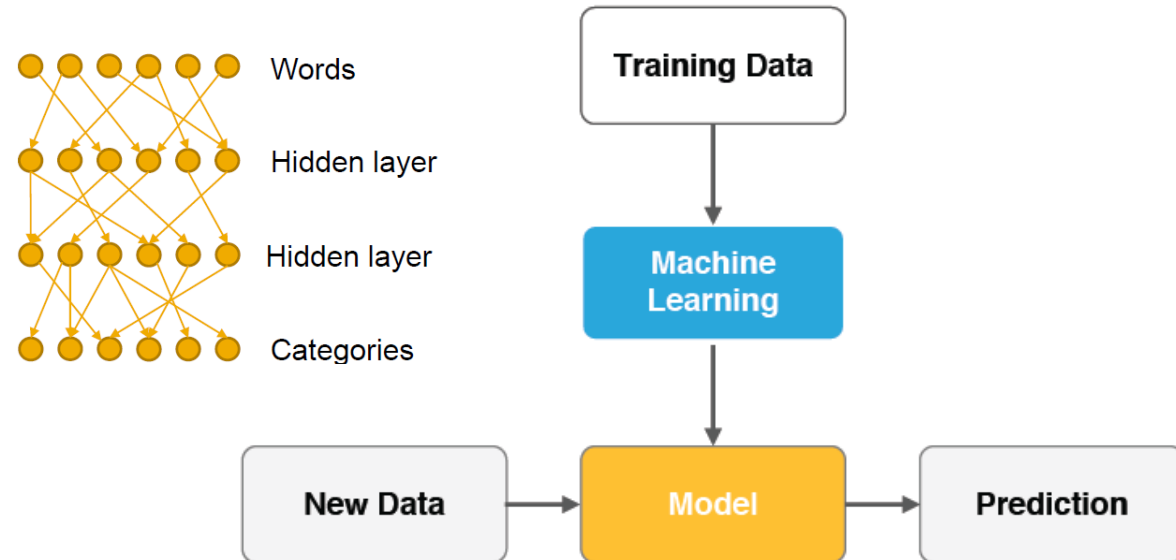
Rule-based approach

```
if (..) then {  
  ..  
} else {  
  ..  
}
```



- Explicitly programmed to solve problems
- Decision rules are **clearly defined** by humans

Machine learning approach



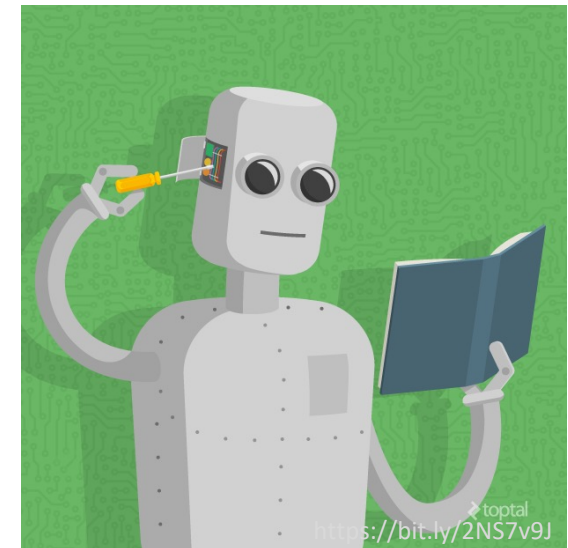
- Trained (i.e., learned) from examples
- Decision rules are **complex and fuzzy**
- Rules are not defined by humans but learned by machines from data

Summary

Machine learning uses historical data to make predictions

It is similar to data mining, but whereas data mining is the science of discovering unknown patterns and relationships in data; machine learning applies previously inferred knowledge to new data to make decisions in real-life applications

- Computers approximate complex functions from historical data
- Rules are not explicitly programmed but learned from data



Data Analytics Process

Data
Preprocessing

Data Sourcing/Representation

- Where are the datasets from

Data Cleaning (Veracity)

- To remove noise and inconsistent data

Data Integration (Variety)

- Multiple data sources may be combined

Data Selection/Reduction (Volume/Velocity)

- Data relevant to the analysis task are retrieved from the database

Data Transformation/Consolidation (Variety, Volume/Velocity)

- Data are transformed and consolidated into forms appropriate for mining or analysis by performing summarization or aggregation operations

Data Mining/Machine Learning

- The essential process where intelligent methods are applied to extract data patterns

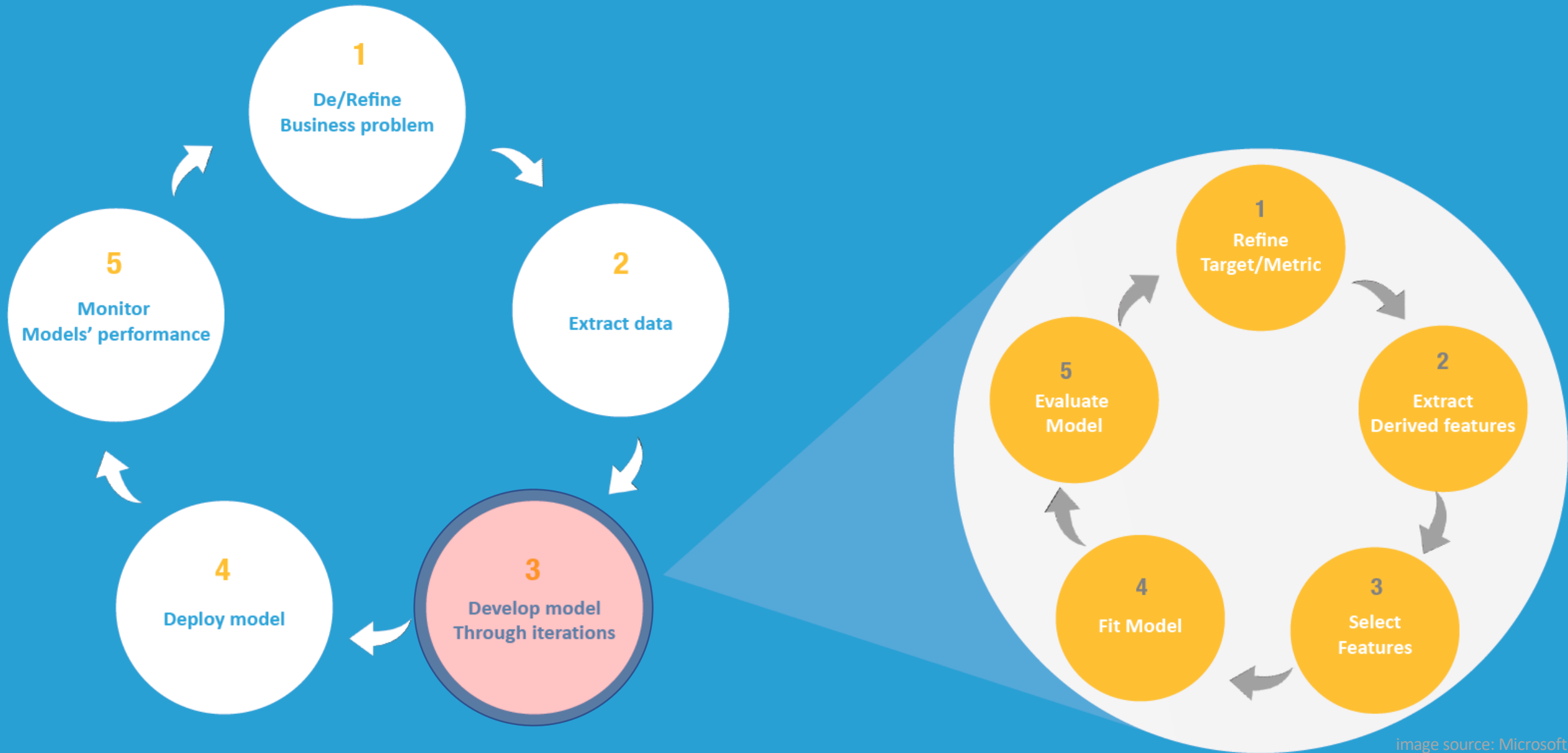
Pattern Evaluation

- To identify the truly interesting patterns representing knowledge

Knowledge Presentation

- Visualization and knowledge representation techniques are used to present mined knowledge to users

Steps to build a machine learning solution

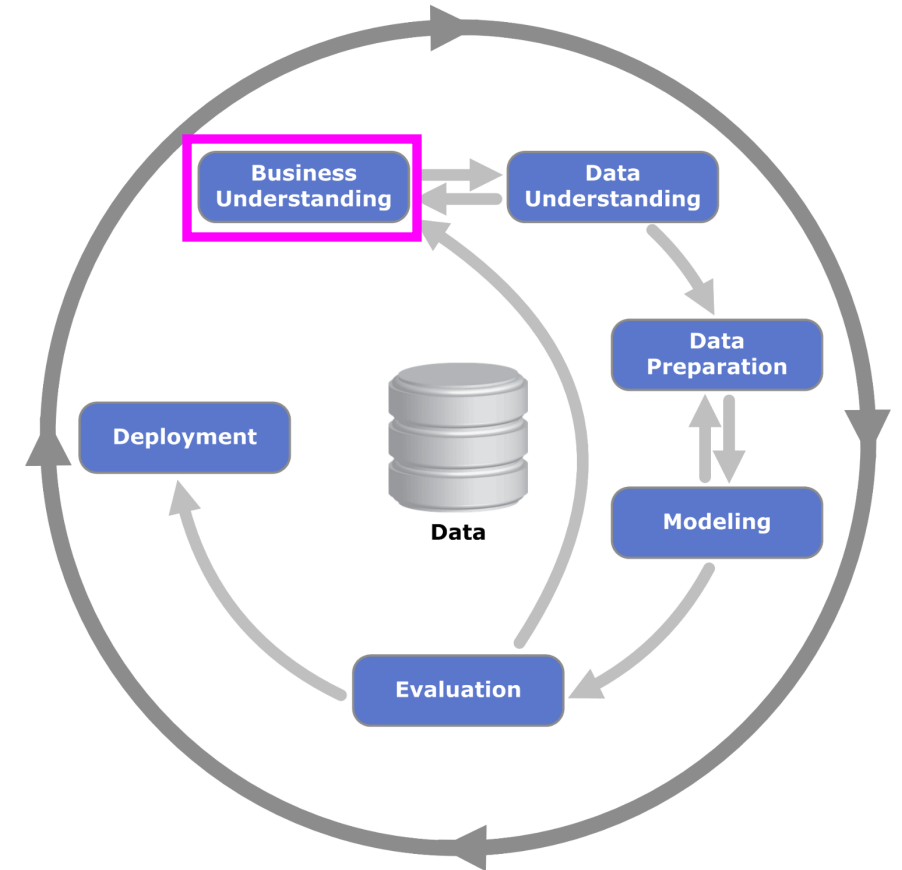


How to Start?

Understand the business

Tasks include

- Identify your business goals
- Assess your situation
- Define your data mining goals
- Produce your project plan



How to start?

Ask a question you can answer with data

- **Sharp questions** can be answered with a name or a number
 - What will my stock's sale price be next week?
 - Which car in my fleet is going to fail first?
- **Vague questions** cannot be answered with a name or a number
 - How can I increase my profits?
 - What can my data tell me about my business?
- How you ask a question is a clue to which algorithm can give you an answer

The 5 questions data science can answer

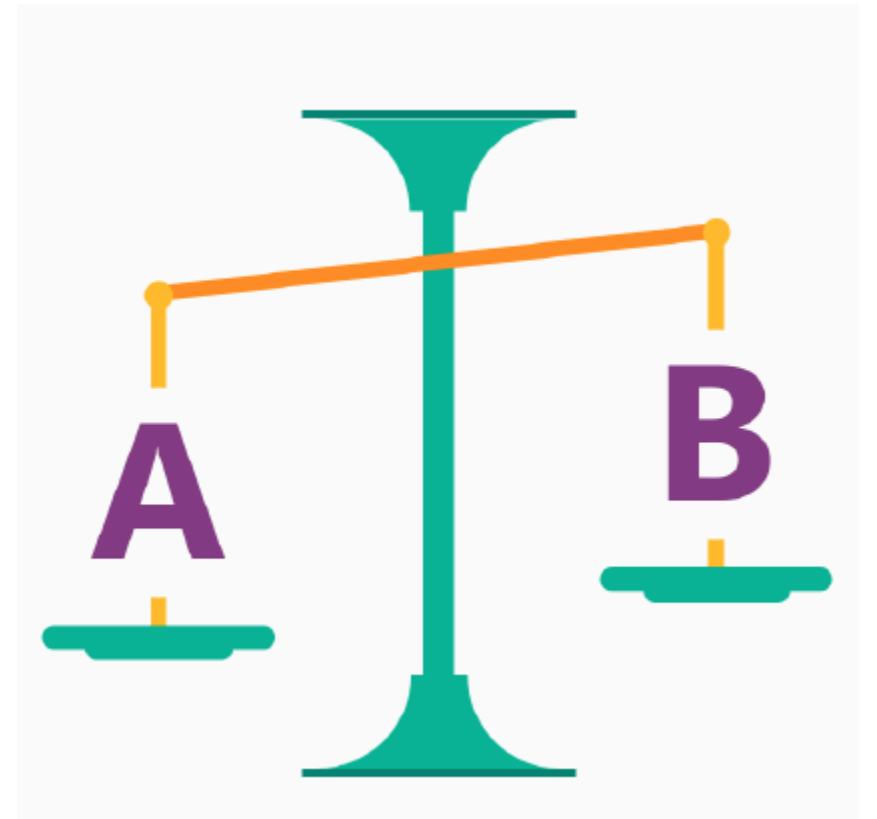
Surprise? but there are only five questions DS can answer

- Is this A or B?
- Is this weird?
- How much or how many?
- How is this organized?
- What should I do now?



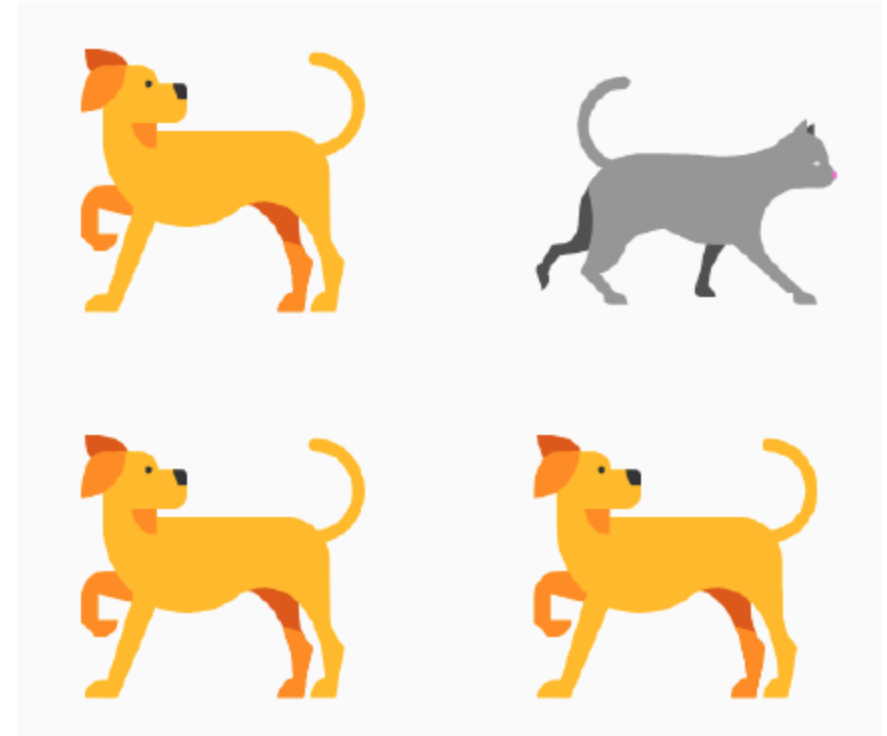
Q1: Is this A or B?

- Use Classification algorithms
- Will this tire fail in next 1,000 miles?
 - Yes or No?
- Which brings in more customers?
 - A \$5 coupon or a 25% discount?



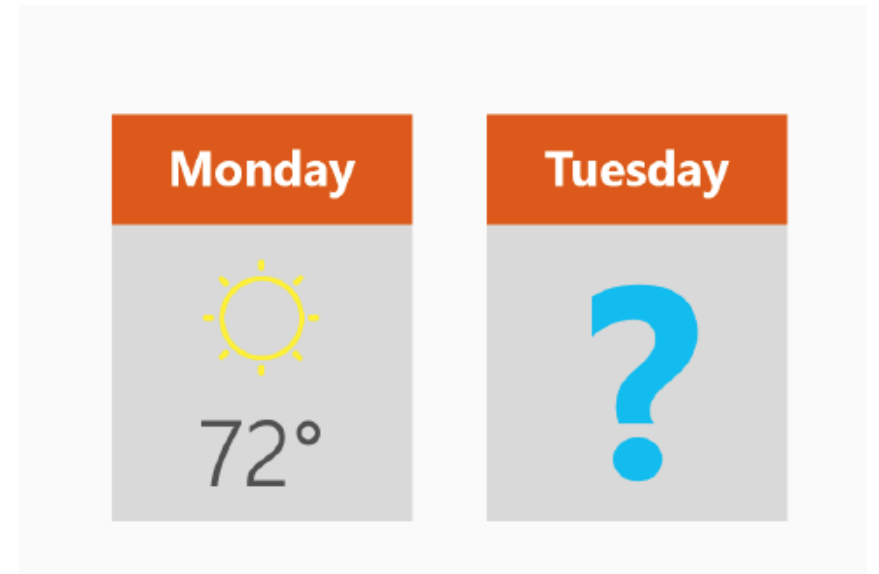
Q2: Is this weird?

- Use Anomaly Detection algorithms
- Your credit company analyzes your purchase pattern, so that they can alert you to possible fraud
- Charges that are “weird” might be a purchase at a store where you do not normally shop or buying an unusually pricey item



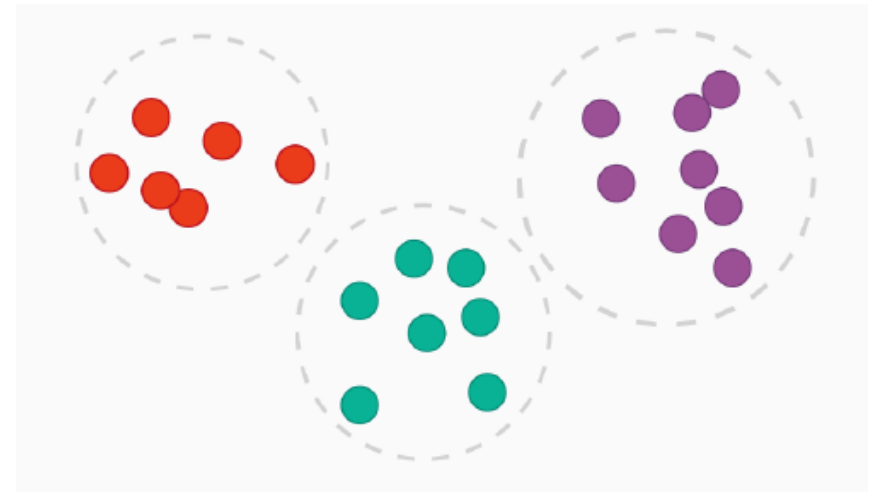
Q3: How much? or How many?

- Use Regression algorithms
- Regression algorithms make numerical predictions, such as
 - What will the temperature be next Tuesday?
 - What will my fourth-quarter sales be?
- They help answer any question that asks for a number



Q4: How is this organized?

- Use Clustering algorithms
- Common examples of clustering questions are:
 - Which viewers like the same types of movies?
 - Which printer models fail the same way?
- Sometimes you want to understand the structure of a data set

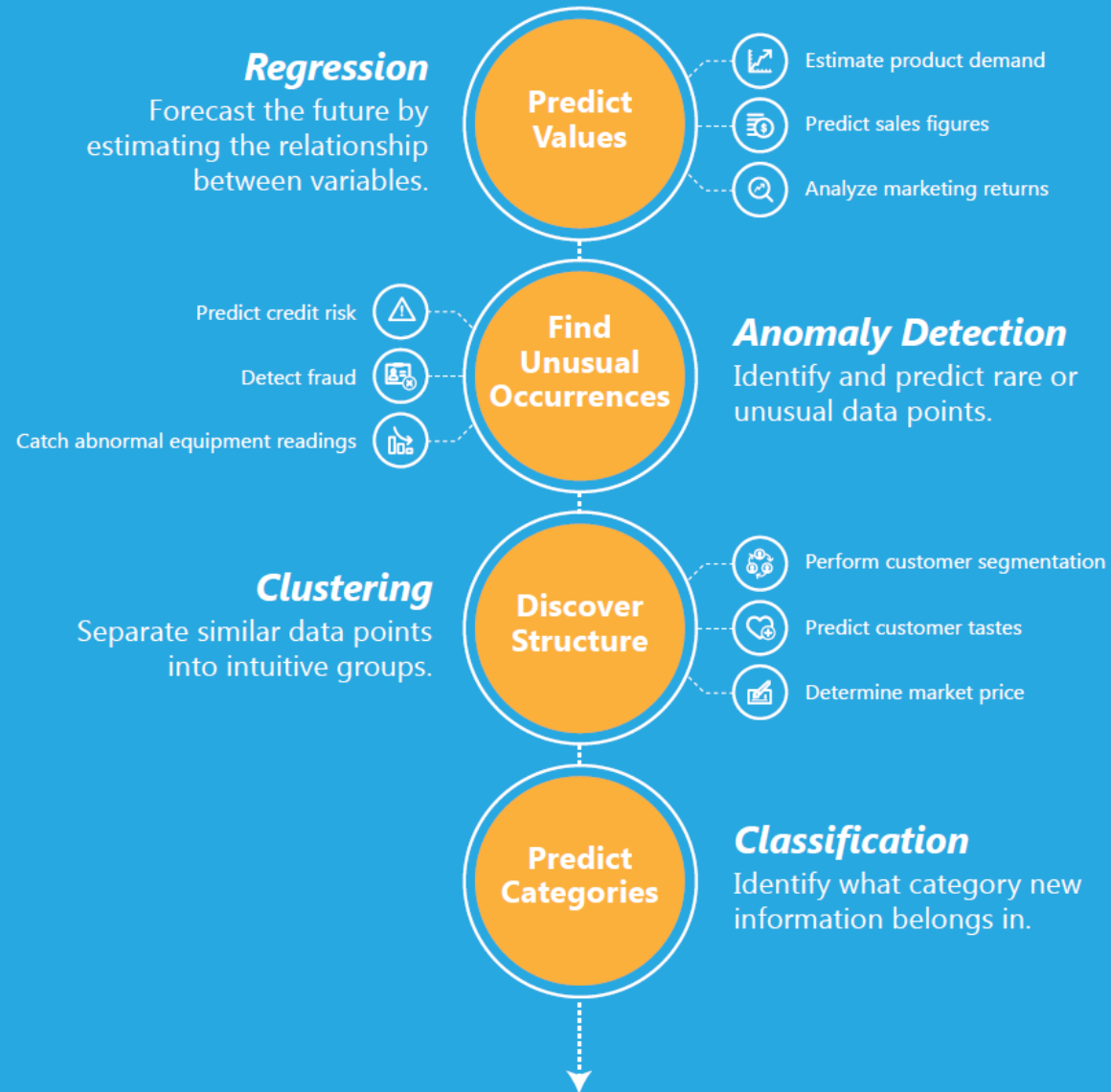


Q5: What should I do now?

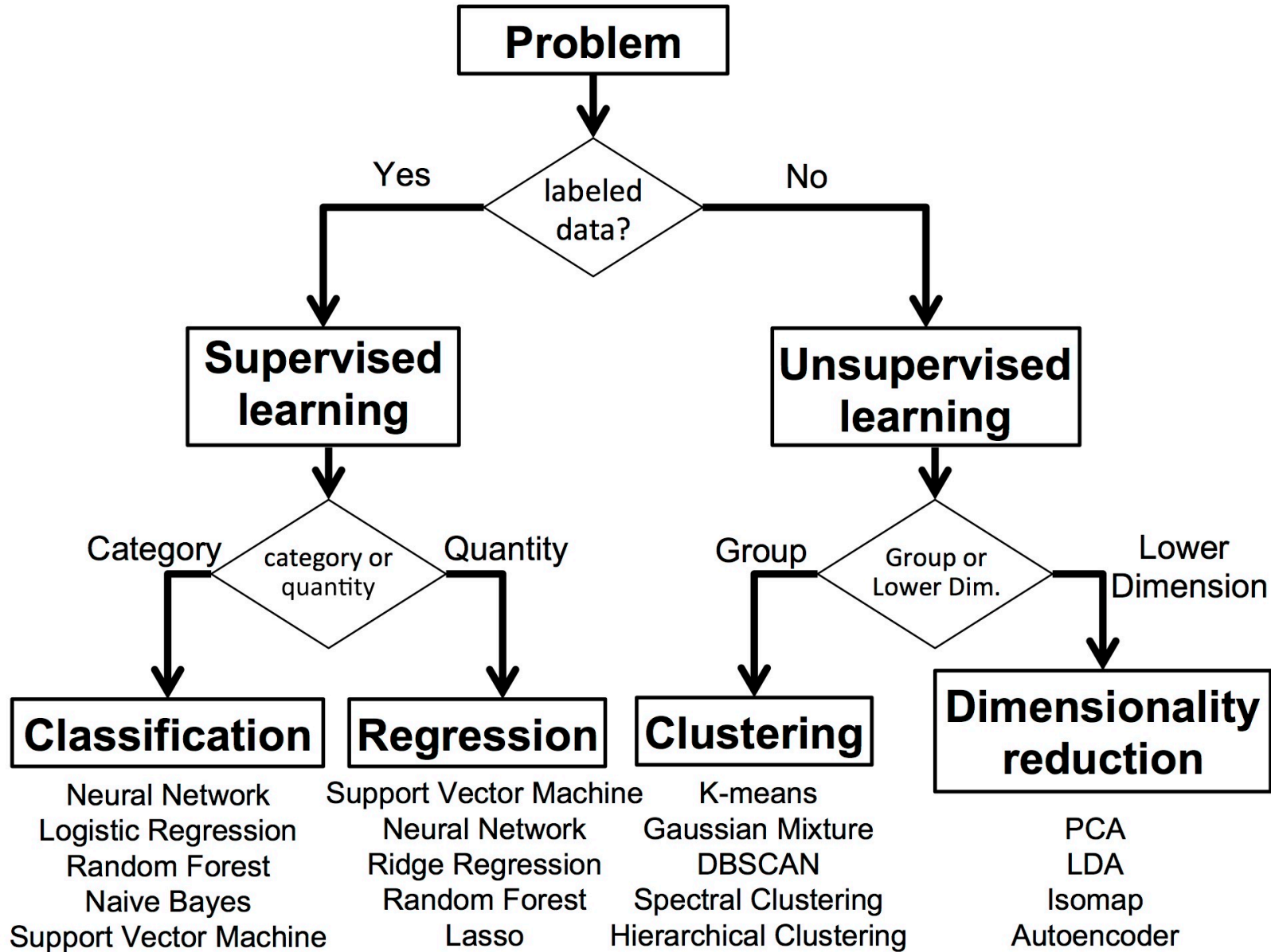
- Use Reinforcement Learning algorithms
- Questions it answers are always about what action should be taken – usually by a machine or robot, e.g.,
 - For a self-driving car: at a yellow light, brake or accelerate?
 - For a robot vacuum: keep vacuuming, or go back to the charging station



So, what do you want to find out?

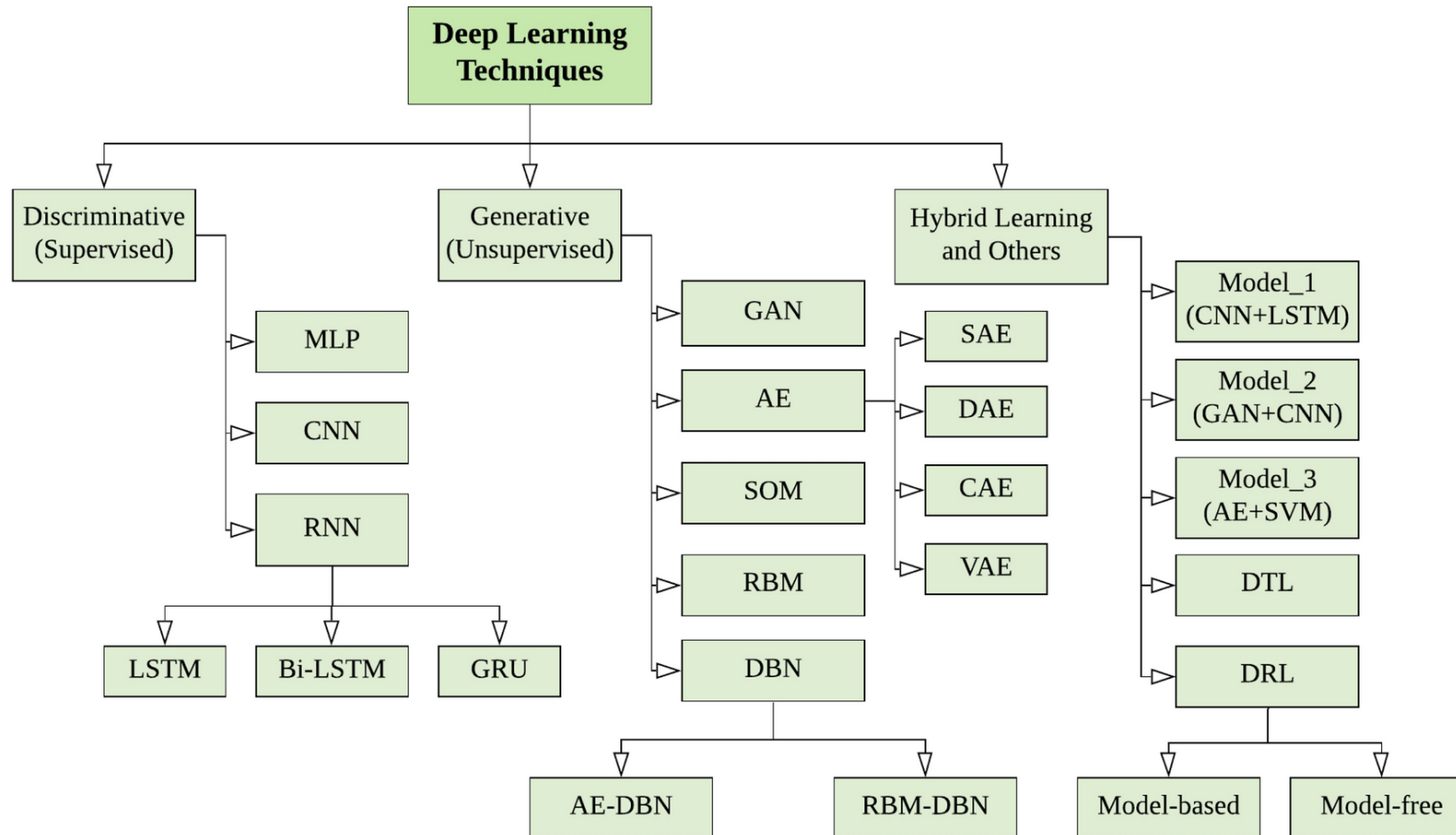


Machine Learning Categorization



Deep Learning

- Three major categories
 - Deep networks for supervised or discriminative learning
 - Deep networks for unsupervised or generative learning
 - Deep networks for hybrid learning and relevant others



When to use machine learning?

From business problem to machine learning problem: a recipe

1. Do you need machine learning?
2. Can you formulate your problem clearly?
3. Do you have sufficient examples?
4. Does your problem have a regular pattern?
5. Can you find meaningful representations of your data?
6. How do you define success?



When to use machine learning?

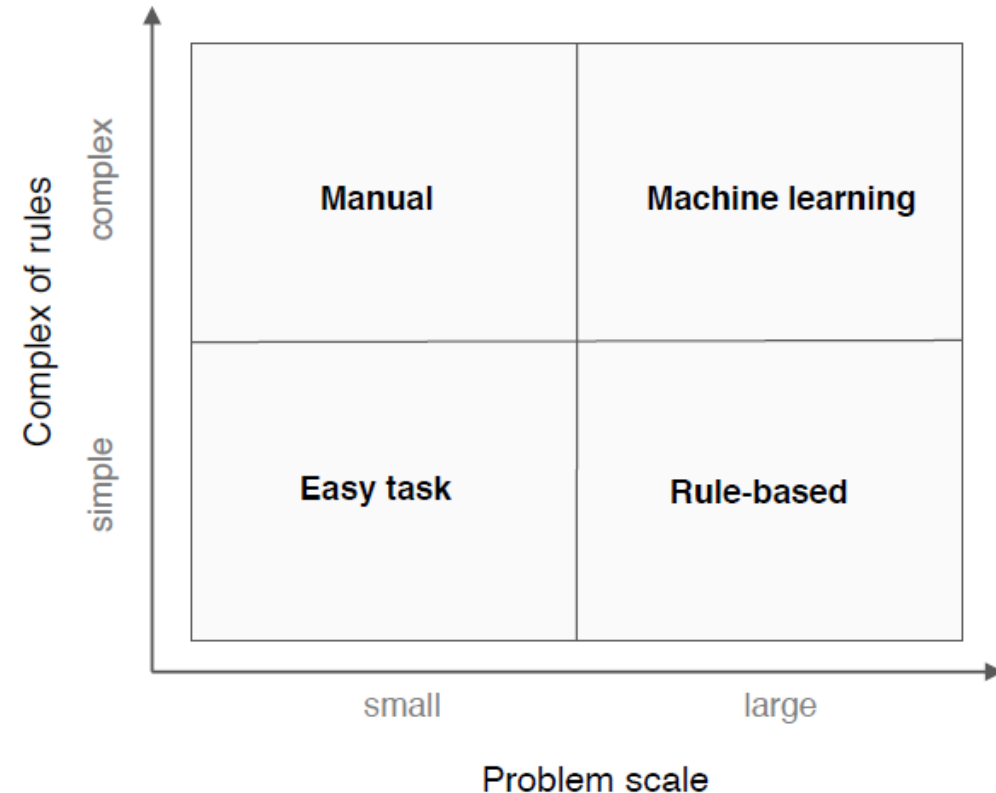
From business problem to machine learning problem: a recipe

1 Do you need machine learning?

- Do you need to automate the task?
- High-volume tasks with complex rules and unstructured data are good candidates

Example: sentiment analysis

- High volume of reviews on the Web
- Unstructured text
- Human language is complex and ambiguous



When to use machine learning?

From business problem to machine learning problem: a recipe

2 Can you formulate your problem clearly?

- What do you want to predict given which input?
- Pattern: “given X, predict Y”
 - What is the input?
 - What is the output?

Example: sentiment analysis

- Given a customer review, predict its sentiment
 - Input: customer review text
 - Output: positive, negative, neutral



When to use machine learning?

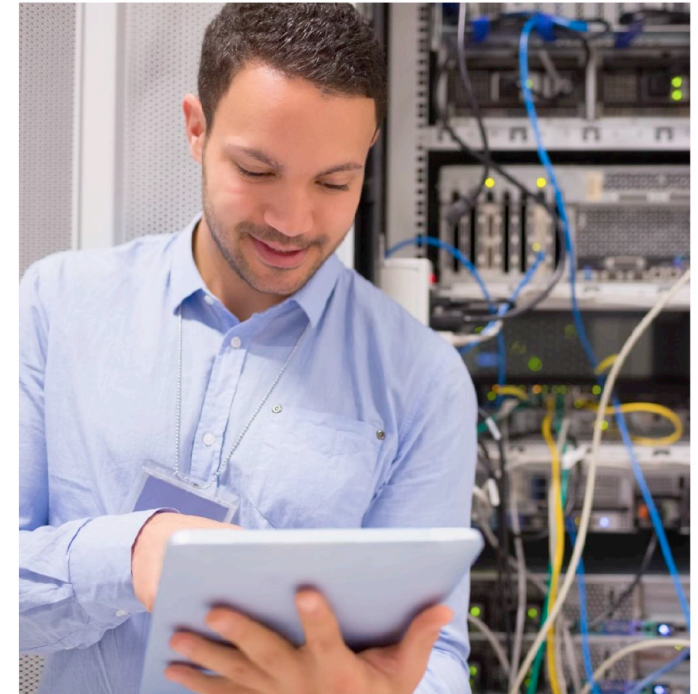
From business problem to machine learning problem: a recipe

3 Do you have sufficient examples?

- Machine learning always requires data
- Generally, the more data, the better
- Each example must contain two parts (supervised learning)
 - Features: attributes of the example
 - Label: the answer you want to predict

Example: sentiment analysis

- Thousands of customer reviews and ratings from the Web



When to use machine learning?

From business problem to machine learning problem: a recipe

4 Does your problem have a regular pattern?

- Machine learning learns regularities and patterns
- Hard to learn patterns that are rare or irregular

Example: sentiment analysis

- Positive words like *good*, *awesome*, or *love it* appear more often in highly-rated reviews
- Negative words like *bad*, *lousy*, or *disappointed* appear more often in poorly-rated reviews



When to use machine learning?

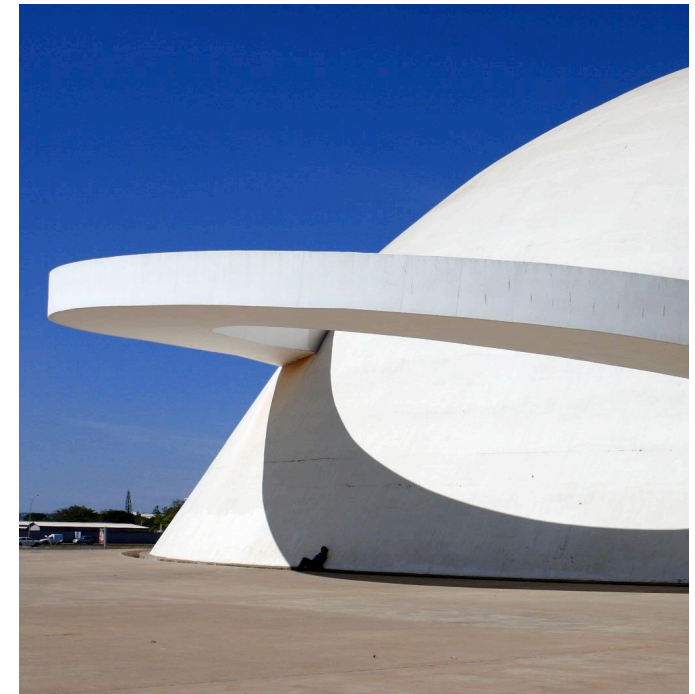
From business problem to machine learning problem: a recipe

5 Can you find meaningful representations of your data?

- Machine learning algorithms ultimately operate on numbers
- Generally, **examples are represented as feature vectors**
- Good features often determine the success of machine learning

Example: sentiment analysis

- Represent customer review as vector of word frequencies
- Label is positive (4-5 stars), negative (1-2 stars), neutral (3 stars)



When to use machine learning?

From business problem to machine learning problem: a recipe

6 How do you define success?

- Machine learning optimizes a training criterion (loss function or objective function)
- The evaluation function must support the business goals

Example: sentiment analysis

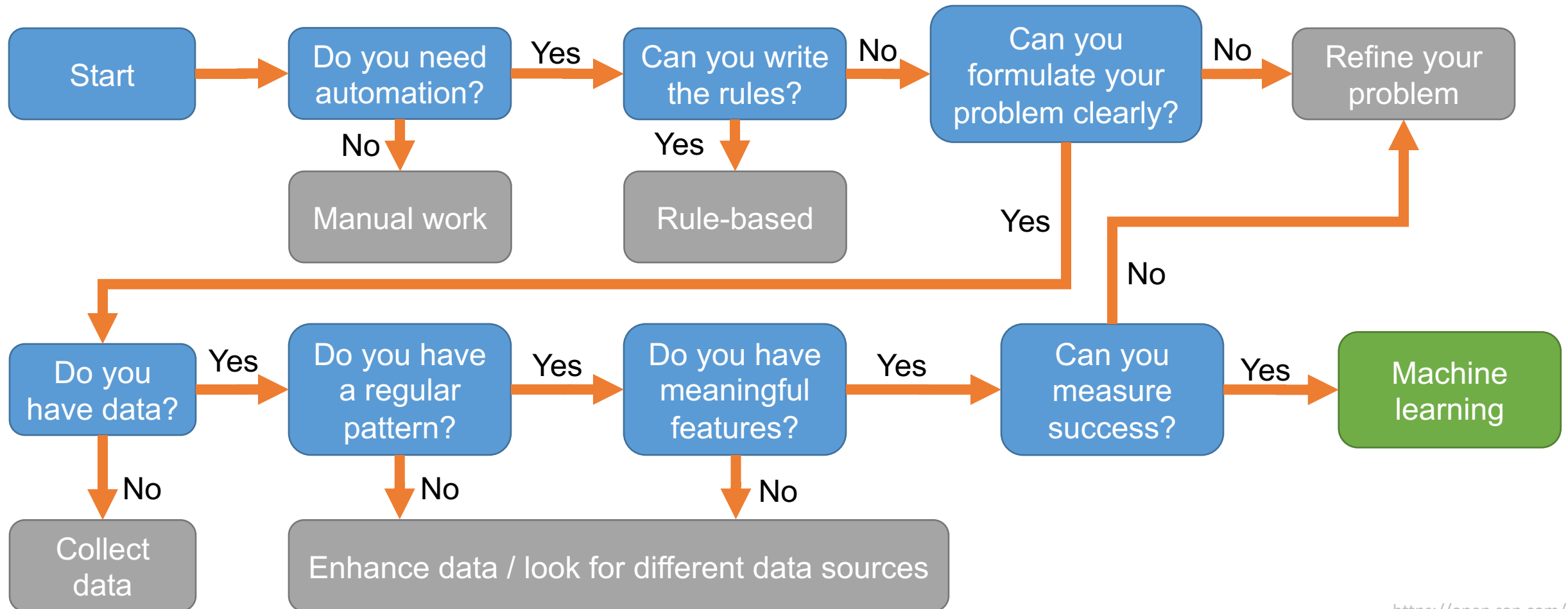
- Accuracy: percentage of correctly predicted labels



When to use machine learning?

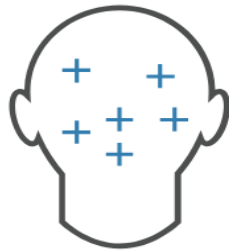
From business problem to machine learning problem: a recipe

The “cheat sheet”



Summary

- Consider using machine learning when you have a complex task or problem involving a large amount of data and lots of variables, but no existing formula or equation.
- For example, machine learning is a good option if you need to handle situations like these:



Hand-written rules and equations are too complex—as in face recognition and speech recognition.



The rules of a task are constantly changing—as in fraud detection from transaction records.



The nature of the data keeps changing, and the program needs to adapt—as in automated trading, energy demand forecasting, and predicting shopping trends.

Takeaway question



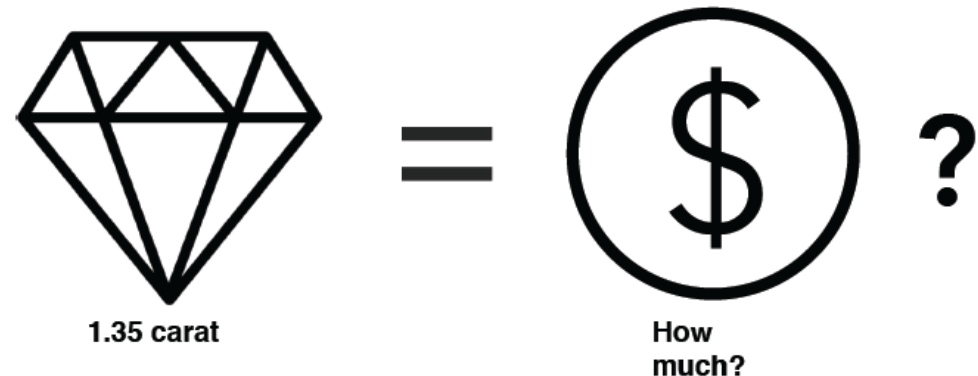
Example: predict the price of a diamond

Suppose you want to shop for a diamond, and you want to get an idea of how much it will cost. I take a notepad and pen into the jewelry store, and I write down the price of all of the diamonds in the case and how much they weigh in carats. Starting with the first diamond – it's 1.01 carats and \$7,366. Now I go through and do this for all the other diamonds in the store.

Prices of diamonds in jewelry stores

Carats	Price (\$)
1.01	7,366
0.49	985
0.31	544
1.51	9,140
0.37	493
0.73	3,011
1.53	11,413
0.56	1,814
0.41	876
0.74	2,690
0.63	1,190
0.6	4,172
2.06	11,764
1.1	4,682
1.31	6,172

How much will it cost to buy a 1.35 carat diamond?

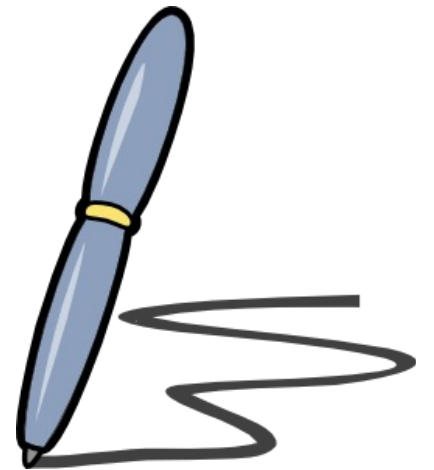


Summary

- Machine learning enables computers to learn from data
 - Computers approximate complex functions from historical data
 - Rules are not explicitly programmed but learned from data
- Intelligent applications are expected to have a significant effect on the future of knowledge work and employment
 - Large number of work activities can be automated with today's technology
- Identify if machine learning can solve your business problem
 - High-volume, repetitive tasks on unstructured data are good candidates
 - You cannot explicitly write the rules, but you have examples
- Machine learning comes with new architecture principles
 - Machine learning requires separate training and prediction steps
 - Micro-services and APIs are a common architecture principle for ML services

Outline

- Big Data Analytics Algorithms
 - Recommendation
 - Clustering
 - Classification
 - Regression



Remind -- Hadoop-related Apache Projects

- [Ambari™](#): A web based tool for provisioning, managing, and monitoring Hadoop clusters. It also provides a dashboard for viewing cluster health and ability to view MapReduce, Pig, and Hive applications visually.
- [Avro™](#): A data serialization system.
- [Cassandra™](#): A scalable multi-master database with no single points of failure.
- [Chukwa™](#): A data collection system for managing large distributed systems.
- [HBase™](#): A scalable, distributed database that supports structured data storage for large tables.
- [Hive™](#): A data warehouse infrastructure that provides data summarization and ad hoc querying
- [Mahout™](#): A scalable machine learning and data mining library.
- [Pig™](#): A high-level data-flow language and execution framework for parallel computation
- [Spark™](#): A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- [Tez™](#): A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases.
- [ZooKeeper™](#): A high-performance coordination service for distributed applications.



Key Components of Mahout



Collaborative Filtering

- User-Based Collaborative Filtering - **single machine**
- Item-Based Collaborative Filtering - **single machine / MapReduce**
- Matrix Factorization with Alternating Least Squares - **single machine / MapReduce**
- Matrix Factorization with Alternating Least Squares on Implicit Feedback- **single machine / MapReduce**
- Weighted Matrix Factorization, SVD++, Parallel SGD - **single machine**

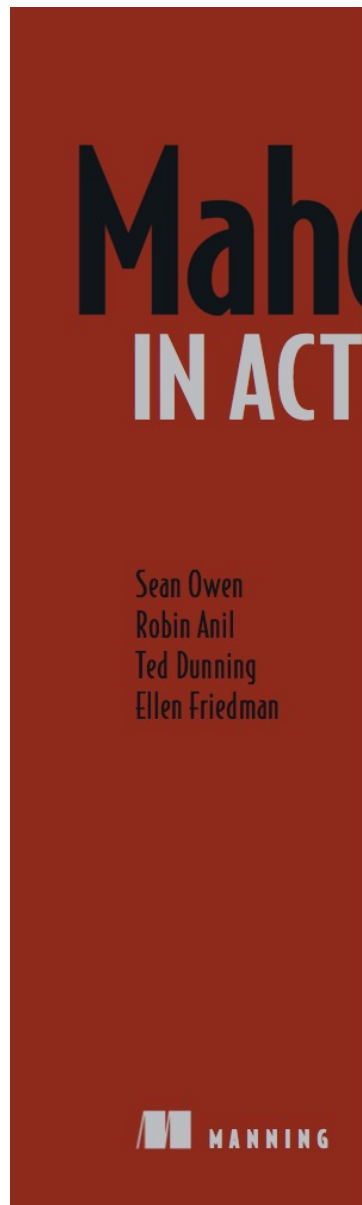
Classification

- Logistic Regression - trained via SGD - **single machine**
- Naive Bayes/ Complementary Naive Bayes - **MapReduce**
- Random Forest - **MapReduce**
- Hidden Markov Models - **single machine**
- Multilayer Perceptron - **single machine**

Clustering

- Canopy Clustering - **single machine / MapReduce** (deprecated, will be removed once Streaming k-Means is stable enough)
- k-Means Clustering - **single machine / MapReduce**
- Fuzzy k-Means - **single machine / MapReduce**
- Streaming k-Means - **single machine / MapReduce**
- Spectral Clustering - **MapReduce**

Mahout reference book



Requires Adobe Acrobat Reader to play audio and video links

- 1 ■ Meet Apache Mahout 1

PART 1 RECOMMENDATIONS 11

- 2 ■ Introducing recommenders 13
- 3 ■ Representing recommender data 26
- 4 ■ Making recommendations 41
- 5 ■ Taking recommenders to production 70
- 6 ■ Distributing recommendation computations 91

PART 2 CLUSTERING 115

- 7 ■ Introduction to clustering 117
- 8 ■ Representing data 130
- 9 ■ Clustering algorithms in Mahout 145
- 10 ■ Evaluating and improving clustering quality 184
- 11 ■ Taking clustering to production 198
- 12 ■ Real-world applications of clustering 210

PART 3 CLASSIFICATION 225

- 13 ■ Introduction to classification 227
- 14 ■ Training a classifier 255
- 15 ■ Evaluating and tuning a classifier 281
- 16 ■ Deploying a classifier 307
- 17 ■ Case study: Shop It To Me 341

Mahout Overview



↓ download mahout

Latest release version 0.9 has

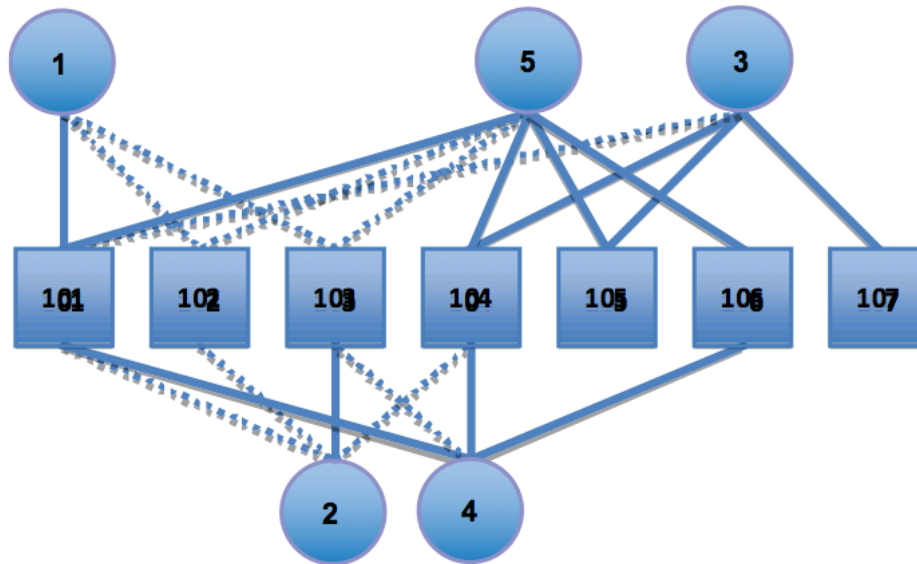
- User and Item based recommenders
- Matrix factorization based recommenders
- K-Means, Fuzzy K-Means clustering
- Latent Dirichlet Allocation
- Singular Value Decomposition
- Logistic regression classifier
- (Complementary) Naive Bayes classifier
- Random forest classifier
- High performance java collections
- A vibrant community

25 April 2014 - Goodbye MapReduce

The Mahout community decided to move its codebase onto modern data processing systems that offer a richer programming model and more efficient execution than Hadoop MapReduce. **Mahout will therefore reject new MapReduce algorithm implementations from now on.** We will however keep our widely used MapReduce algorithms in the codebase and maintain them.

We are building our future implementations on top of a [DSL for linear algebraic operations](#) which has been developed over the last months. Programs written in this DSL are automatically optimized and executed in parallel on [Apache Spark](#).

Recommender -- inputs



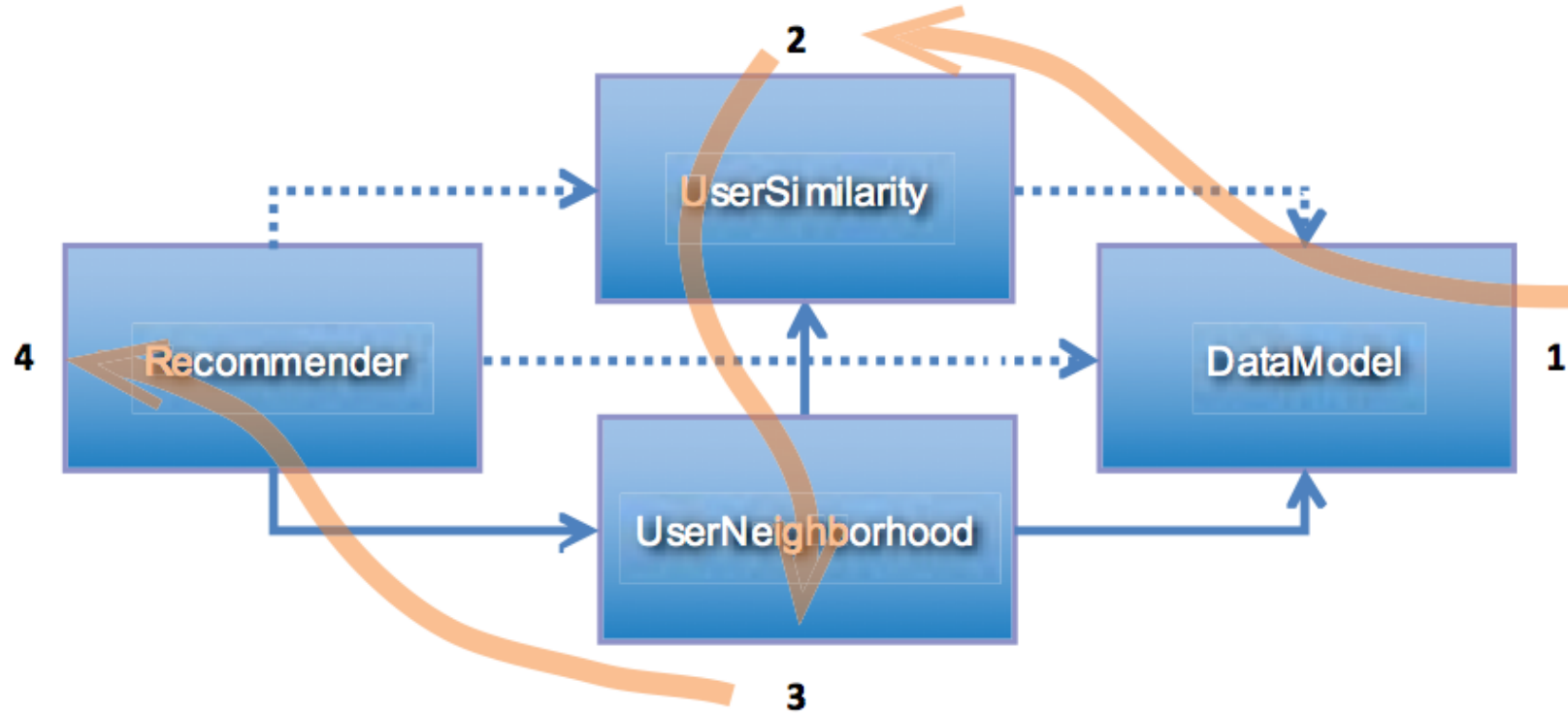
User	Item	Rating
1,	101,	5.0
1,	102,	3.0
1,	103,	2.5
2,	101,	2.0
2,	102,	2.5
2,	103,	5.0
2,	104,	2.0
3,	101,	2.5
3,	104,	4.0
3,	105,	4.5
3,	107,	5.0
4,	101,	5.0
4,	103,	3.0
4,	104,	4.5
4,	106,	4.0
5,	101,	4.0
5,	102,	3.0
5,	103,	2.0
5,	104,	4.0
5,	105,	3.5
5,	106,	4.0

User-based Recommendation – Scenario I

- *Adult: I'm looking for a CD for a teenager.*
- *Employee: OK, what does this teenager like?*
- *Adult: Oh, you know, what all the young kids like this days.*
- *Employee: What kind of music or bands?*
- *Adult: It's all noise to me. I don't know.*
- *Employee: Uh, well...I guess lots of young people are buying this boy band album here by New 2 Town?*
- *Adult: Sold!*



Process and output of the example



Predict user u 's rating for a new item:

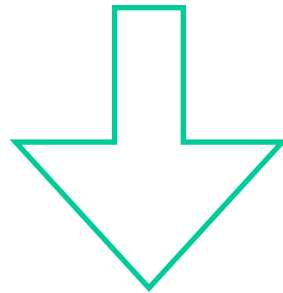
1. Compute user similarity
2. Decide u 's neighborhood
3. Among all users in u 's neighborhood, compute their average rating for this item weighted by their respective similarity with u .

`RecommendedItem [item:104, value:4.257081]`

*Recommendation for Person 1:
Item 104 > Item 106
Item 107 is not favored*

User-based Recommendation Algorithms

```
for every item i that u has no preference for yet
  for every other user v that has a preference for i
    compute a similarity s between u and v
    incorporate v's preference for i, weighted by s, into a running average
return the top items, ranked by weighted average
```



Consider a
neighborhood only,
instead of all other users

```
for every other user w
  compute a similarity s between u and w
  retain the top users, ranked by similarity, as a neighborhood n
for every item i that some user in n has a preference for,
  but that u has no preference for yet
  for every other user v in n that has a preference for i
    compute a similarity s between u and v
    incorporate v's preference for i, weighted by s, into a running average
```

Example Recommender Code via Mahout

```
class RecommenderIntro {  
    public static void main(String[] args) throws Exception {  
        DataModel model =  
            new FileDataModel (new File("intro.csv"));           ← Load data file  
  
        UserSimilarity similarity =  
            new PearsonCorrelationSimilarity (model);  
        UserNeighborhood neighborhood =  
            new NearestNUserNeighborhood (2, similarity, model);  
  
        Recommender recommender = new GenericUserBasedRecommender (  
            model, neighborhood, similarity);                       ← Create  
recommender engine  
  
        List<RecommendedItem> recommendations =  
            recommender.recommend(1, 1);                            ← For user I,  
recommend  
I item  
  
        for (RecommendedItem recommendation : recommendations) {  
            System.out.println(recommendation);  
        }  
    }  
}
```

User Similarity Measurements

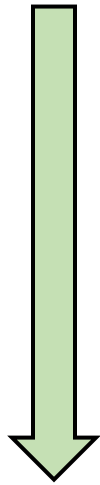
- Pearson Correlation Similarity
- Euclidean Distance Similarity
- Cosine Measure Similarity
- Spearman Correlation Similarity
- Tanimoto Coefficient Similarity (Jaccard coefficient)
- Log-Likelihood Similarity



Pearson Correlation Similarity

Covariance of the two variables divided by the product of their standard deviations

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$



$$\mu_X = E[X]$$

$$\mu_Y = E[Y]$$

$$\sigma_X^2 = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

$$\sigma_Y^2 = E[(Y - E[Y])^2] = E[Y^2] - E[Y]^2$$

$$E[(X - \mu_X)(Y - \mu_Y)] = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

$$\rho_{X,Y} = \frac{\sum XY - \frac{(\sum X)(\sum Y)}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right) \left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}$$

Pearson Correlation Similarity

$$\rho_{X,Y} = \frac{\sum XY - \frac{(\sum X)(\sum Y)}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right) \left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}$$

$$\sum XY = (5.0)(2.0) + (3.0)(2.5) + (2.5)(5.0) = 30$$

$$\sum X = 5.0 + 3.0 + 2.5 = 10.5$$

$$\sum Y = 2.0 + 2.5 + 5.0 = 9.5$$

$$\sum X^2 = 5.0^2 + 3.0^2 + 2.5^2 = 40.25$$

$$\sum Y^2 = 2.0^2 + 2.5^2 + 5.0^2 = 35.25$$

$$N = 3$$

$$\rho_{X,Y} = \frac{30 - \frac{(10.5)(9.5)}{3}}{\sqrt{\left(40.25 - \frac{(10.5)^2}{3}\right) \left(35.25 - \frac{(9.5)^2}{3}\right)}} = -0.7643$$

Calculate each component

X = [5.0, 3.0, 2.5]

Y = [2.0, 2.5, 5.0]

ratings of common items

User	Item	Rating
1,	101,	5.0
1,	102,	3.0
1,	103,	2.5
2,	101,	2.0
2,	102,	2.5
2,	103,	5.0
2,	104,	2.0
3,	101,	2.5
3,	104,	4.0
3,	105,	4.5
3,	107,	5.0
4,	101,	5.0
4,	103,	3.0
4,	104,	4.5
4,	106,	4.0
5,	101,	4.0
5,	102,	3.0
5,	103,	2.0
5,	104,	4.0
5,	105,	3.5
5,	106,	4.0

	Item 101	Item 102	Item 103	Correlation with user 1
User 1	5.0	3.0	2.5	1.000
User 2	2.0	2.5	5.0	
User 3	2.5	-	-	-
User 4	5.0	missing data	3.0	
User 5	4.0	3.0	2.0	

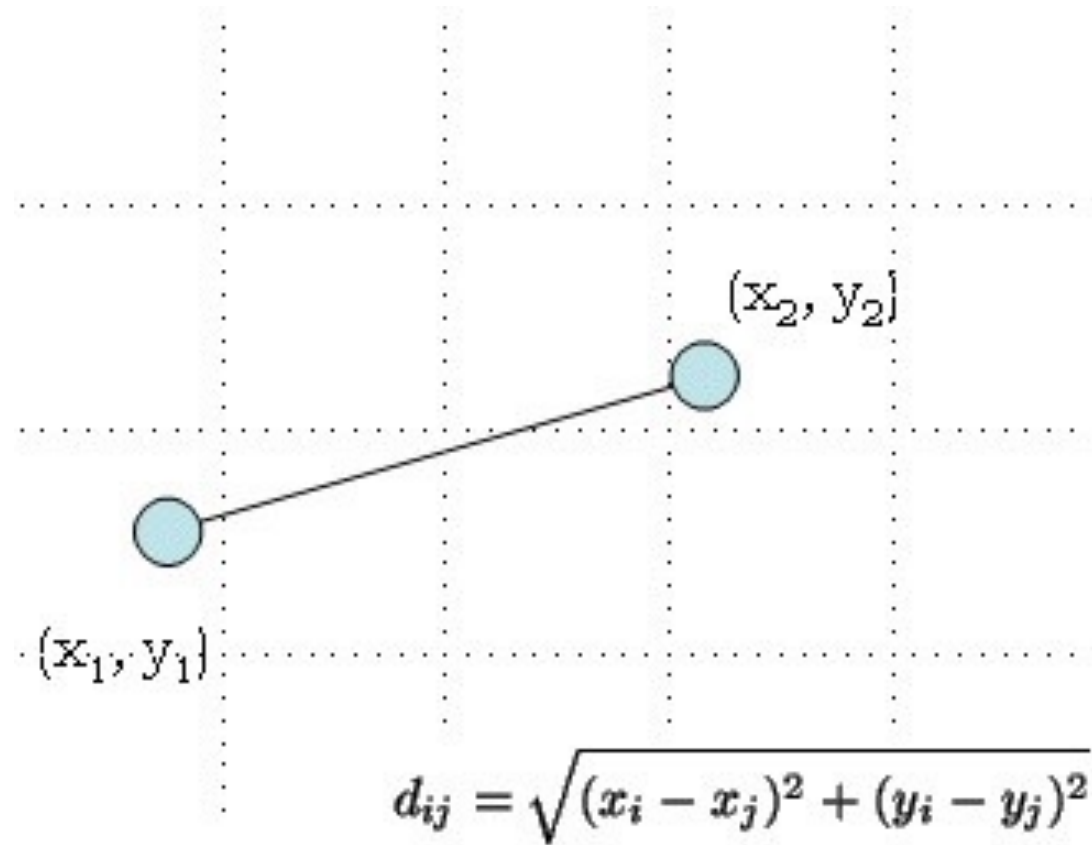
Three problems with the Pearson Similarity

- Not take into account of the number of items, in which two users' preferences overlap
 - 1,000 overlapped items vs. 5 overlapped items, which recommendation is more trustworthy?
- If two users overlap on only one item, no correlation can be computed
- The correlation is undefined if either series of preference values are identical

Adding Weighting WEIGHTED as the 2nd parameter of the constructor can cause the resulting correlation to be pushed towards 1.0, or -1.0, depending on how many points are used.



Euclidean Distance



Euclidean Distance Similarity

$$d(X, Y) = d(Y, X) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \dots + (y_n - x_n)^2} = \sqrt{\sum_{i=1}^n (y_i - x_i)^2}$$

$$\text{Similarity} = \rho_{X,Y} = \frac{1}{1 + d} = \frac{1}{1 + \sqrt{\sum_{i=1}^n (y_i - x_i)^2}}$$

User	Item	Rating
1,	101,	5.0
1,	102,	3.0
1,	103,	2.5
2,	101,	2.0
2,	102,	2.5
2,	103,	5.0
2,	104,	2.0
3,	101,	2.5
3,	104,	4.0
3,	105,	4.5
3,	107,	5.0
4,	101,	5.0
4,	103,	3.0
4,	104,	4.5
4,	106,	4.0
5,	101,	4.0
5,	102,	3.0
5,	103,	2.0
5,	104,	4.0
5,	105,	3.5
5,	106,	4.0

	Item 101	Item 102	Item 103	Distance	Similarity to user 1
User 1	5.0	3.0	2.5	0.000	1.000
User 2	2.0	2.5	5.0	3.937	0.203
User 3	2.5	-	-	2.500	0.286
User 4	5.0	-	3.0	0.500	0.667
User 5	4.0	3.0	2.0	1.118	0.472

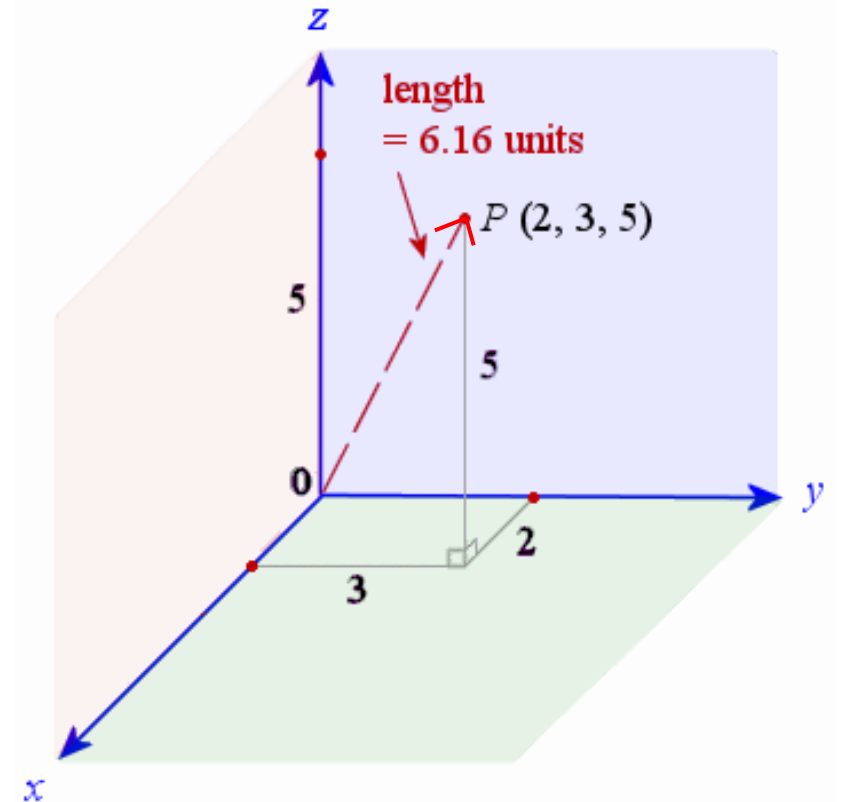
Cosine Measure Similarity

Cosine similarity and Pearson similarity get the same results if data are normalized (i.e., mean is zero)

Map the input data into a high-dimensional space

- Each point in the space represents a different user
- What does each dimension represent?

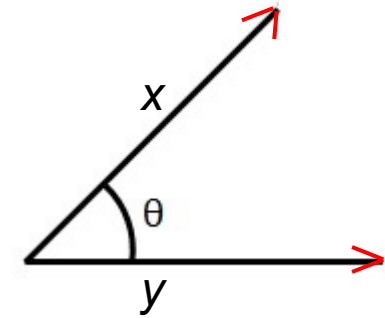
$$\rho_{X,Y} = \cos(\theta_{X,Y}) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|} = \frac{\sum_{i=1}^n (x_i \cdot y_i)}{\sqrt{\sum_{i=1}^n x_i^2} \cdot \sqrt{\sum_{i=1}^n y_i^2}}$$



Cosine Measure Similarity – A numerical example

Cosine similarity and Pearson similarity get the same results if data are normalized (i.e., mean is zero)

$$\rho_{X,Y} = \cos(\theta_{X,Y}) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|} = \frac{\sum_{i=1}^n (x_i \cdot y_i)}{\sqrt{\sum_{i=1}^n x_i^2} \cdot \sqrt{\sum_{i=1}^n y_i^2}}$$



Geometric illustration of the cosine measure

- $x = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$
- $y = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$
- $x \cdot y = 3 \cdot 1 + 2 \cdot 0 + \dots + 0 \cdot 2 = 5$
- $\|x\| = (3 \cdot 3 + 2 \cdot 2 + \dots + 0 \cdot 0)^{1/2} = 6.48$
- $\|y\| = (1 \cdot 1 + 0 \cdot 0 + \dots + 2 \cdot 2)^{1/2} = 2.24$
- Cosine Similarity of x and y is: $(5 / (6.48 \cdot 2.24)) = 0.34$

Spearman Correlation Similarity

Calculated based on Pearson value about relative ranks

In addition to actual rating values, their order (rank) also matters

- Consider two users: one super nice (high ratings 3-5), the other mean (low ratings 1-3)
- Their similarity is low if using actual rating values, but not really!

$$\rho_{X,Y} = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

$$d_i = x_i - y_i$$

Variable X_i	Position in the ascending order	Rank x_i
0.8	1	1
1.2	2	$\frac{2+3}{2} = 2.5$
1.2	3	$\frac{2+3}{2} = 2.5$
2.3	4	4
18	5	5

Example for ties

$X = [1, 2, 3]$
 $Y = [3, 2, 1]$
 $d = [2, 0, 2]$

	Item 101	Item 102	Item 103	Correlation to user 1
User 1	3.0	2.0	1.0	1.0
User 2	1.0	2.0	3.0	-1.0
User 3	1.0	-	-	-
User 4	2.0	-	1.0	1.0
User 5	3.0	2.0	1.0	-

User	Item	Rating
1,	101,	5.0
1,	102,	3.0
1,	103,	2.5
2,	101,	2.0
2,	102,	2.5
2,	103,	5.0
2,	104,	2.0
3,	101,	2.5
3,	104,	4.0
3,	105,	4.5
3,	107,	5.0
4,	101,	5.0
4,	103,	3.0
4,	104,	4.5
4,	106,	4.0
5,	101,	4.0
5,	102,	3.0
5,	103,	2.0
5,	104,	4.0
5,	105,	3.5
5,	106,	4.0

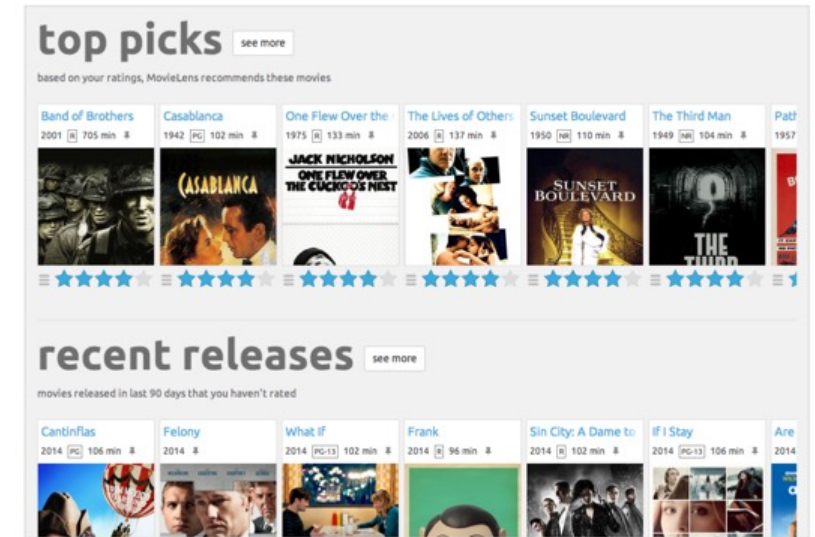
Performance Metrics

- Using GroupLens data (<http://grouplens.org>) -- 10 million MovieLens rating dataset

- Spearman: 0.8
- Tanimoto: 0.82
- Log-Likelihood: 0.73
- Euclidean: 0.75
- Pearson (weighted): 0.77
- Pearson: 0.89

recommendations

MovieLens helps you find movies you will like. Rate movies to build a custom taste profile, then MovieLens recommends other movies for you to watch.



	Item 1	Item 2	Item 3
Actual	3.0	5.0	4.0
Estimate	3.5	2.0	5.0
Difference	0.5	3.0	1.0
Average difference	= (0.5 + 3.0 + 1.0) / 3 = 1.5		
Root-mean-square	= $\sqrt{((0.5^2 + 3.0^2 + 1.0^2) / 3)} = 1.8484$		

Performance metrics



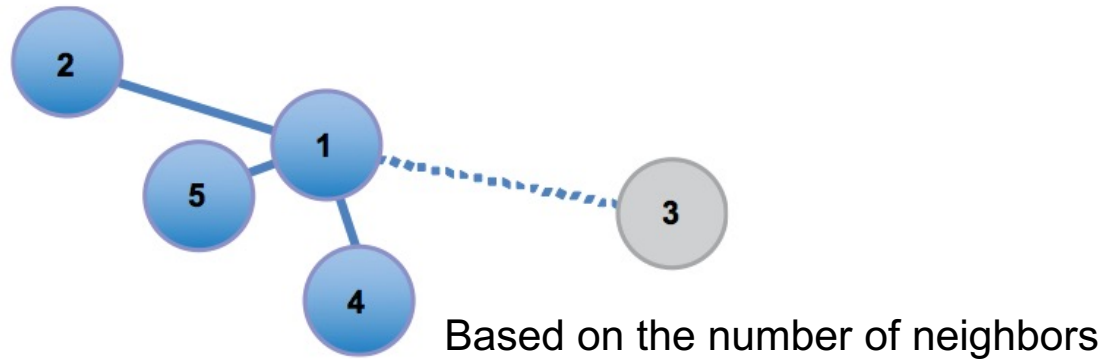
Mahout Code with Performance Measurements

```
DataModel model = new GroupLensDataModel (new File("ratings.dat"));
RecommenderEvaluator evaluator =
    new AverageAbsoluteDifferenceRecommenderEvaluator ();
RecommenderBuilder recommenderBuilder = new RecommenderBuilder() {
    @Override
    public Recommender buildRecommender(
        DataModel model) throws TasteException {
        UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
        UserNeighborhood neighborhood =
            new NearestNUserNeighborhood(100, similarity, model);
        return new GenericUserBasedRecommender(
            model, neighborhood, similarity);
    }
};
double score = evaluator.evaluate(
    recommenderBuilder, null, model, 0.95, 0.05);
System.out.println(score);
```

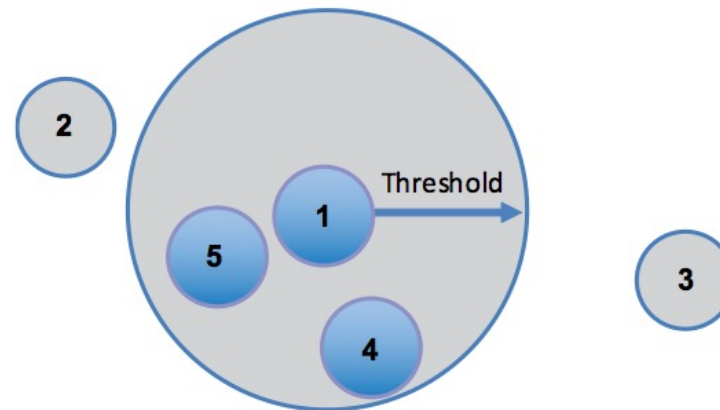
10 nearest neighbors: 0.98
100 nearest neighbors: 0.89
500 nearest neighbors: 0.75

95% of training; 5% of testing

Selecting neighborhood



```
new NearestNUserNeighborhood(100, similarity, model);
```



Based on a fixed threshold, e.g., 0.7 or **0.5**

```
new ThresholdUserNeighborhood(0.7, similarity, model)
```

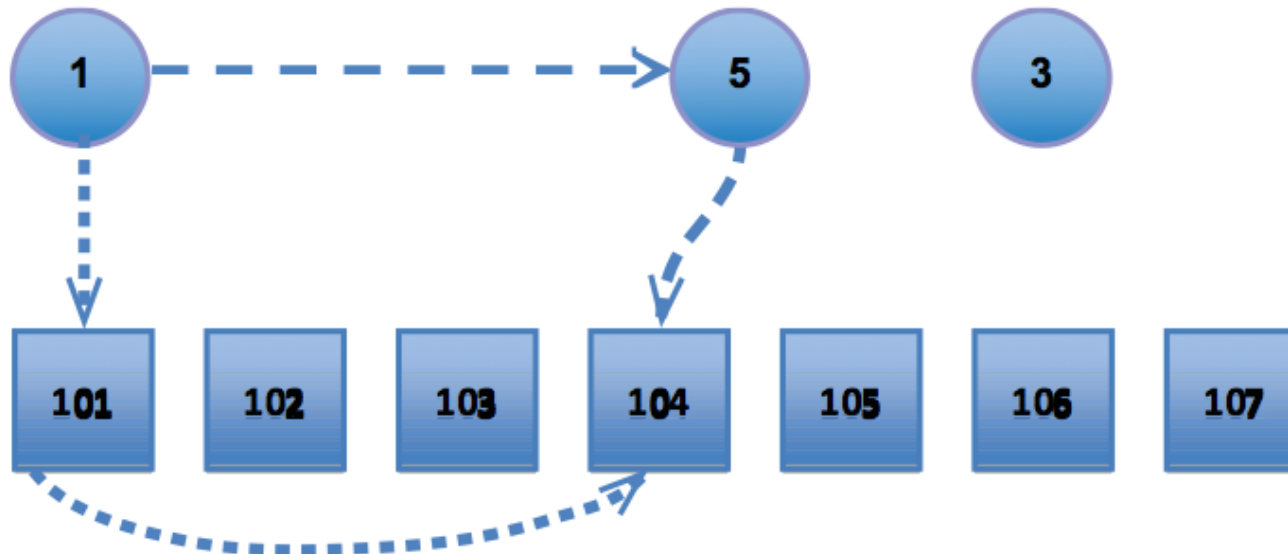
Item-based Recommendation – Scenario I

- *Adult: I'm looking for a CD for a teenager boy.*
- *Employee: What kind of music or bands does he like?*
- *Adult: He wears a Bowling In Hades T-shirt all the time and seems to have all of their albums. Anything else you'd recommend?*
- *Employee: Well, about everyone I know that likes Bowling In Hades seems to like the new Rock Mobster album.*



Item-based Recommendation Algorithm

```
for every item  $i$  that  $u$  has no preference for yet
  for every item  $j$  that  $u$  has a preference for
    compute a similarity  $s$  between  $i$  and  $j$ 
    add  $u$ 's preference for  $j$ , weighted by  $s$ , to a running average
return the top items, ranked by weighted average
```



Code and Performance of Item-Based Recommendation

```
public Recommender buildRecommender(DataModel model)
    throws TasteException {
    ItemSimilarity similarity = new PearsonCorrelationSimilarity(model);
    return new GenericItemBasedRecommender(model, similarity);
}
```

Performance:

Implementation	Similarity
PearsonCorrelationSimilarity	0.75
PearsonCorrelationSimilarity + weighting	0.75
EuclideanDistanceSimilarity	0.76
EuclideanDistanceSimilarity + weighting	0.78
TanimotoCoefficientSimilarity	0.77
LogLikelihoodSimilarity	0.77

Which method to use, user-based or item-based?

One thing you may notice is that this recommender setup runs significantly faster. That's not surprising, given that the data set has about 70,000 users and 10,000 items.

Item-based recommenders are generally faster when there are fewer items than users.

Clustering

- Given a set of data points, each having a set of attributes, and a *similarity* measure among them, find clusters such that
 - Data points in one cluster are more similar to one another
 - Data points in separate clusters are less similar to one another
- Similarity (distance) measures:
 - *Euclidean distance* if attributes are continuous
 - Other problem-specific measures

Clustering a collection involves three things



An algorithm -- This is the method used to group the objects together.

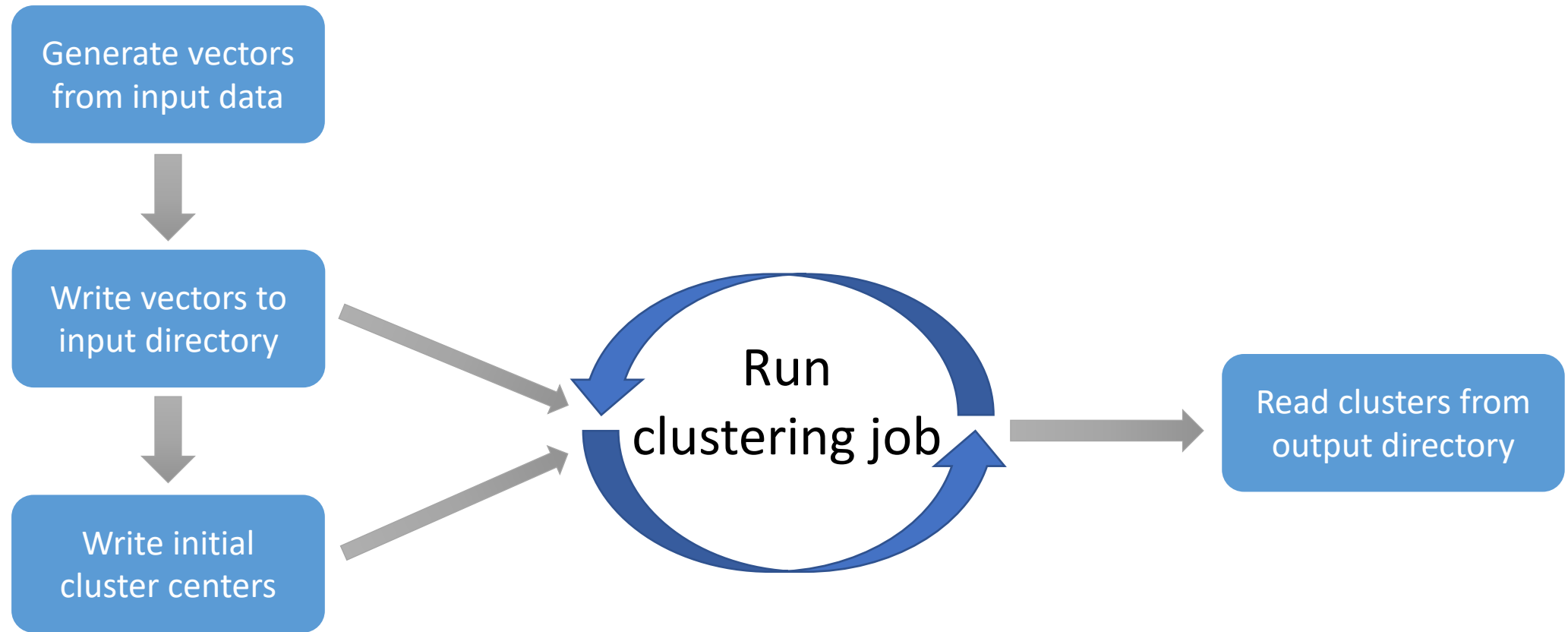


A notion of both similarity and dissimilarity -- This determines which objects belong to an existing group (cluster) and which should start a new one.



A stopping condition -- This might be the point beyond which objects cannot be grouped (clustered) anymore, or when the objects are already quite dissimilar.

Steps on Clustering

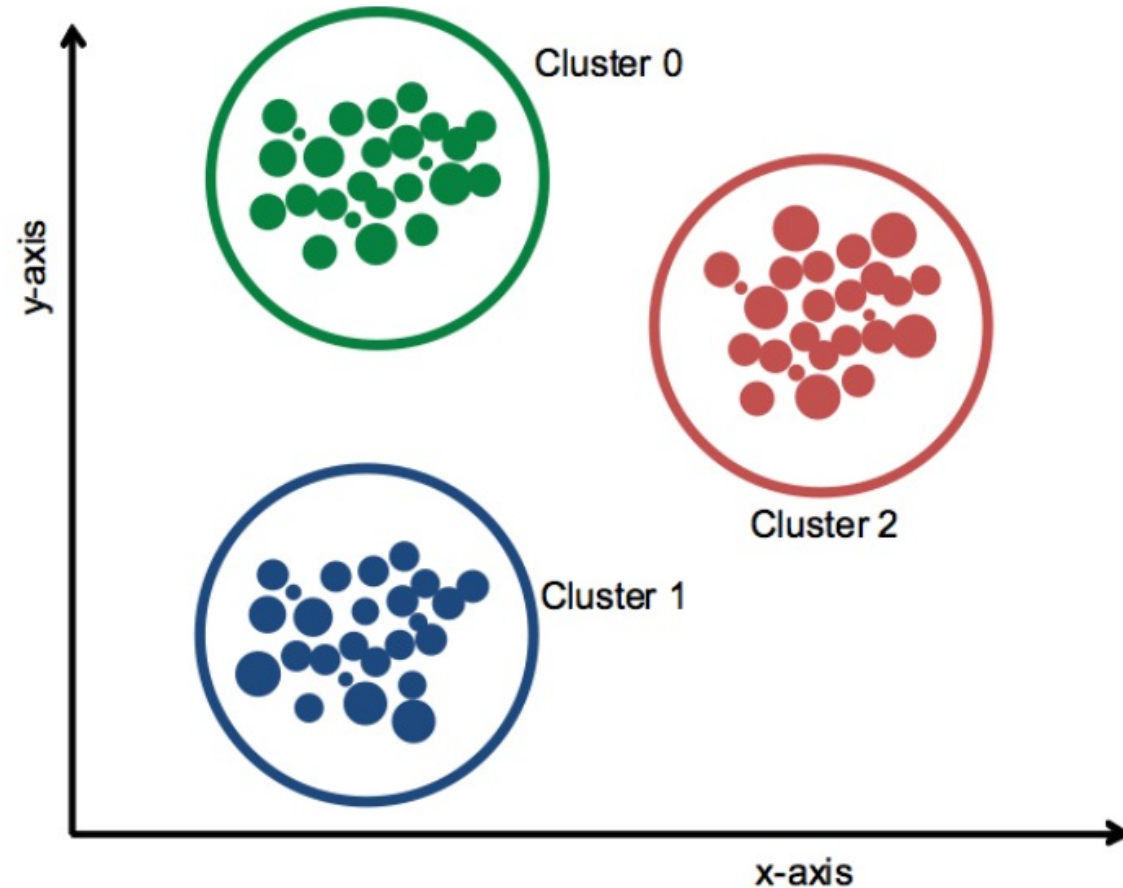


Clustering on a 2D Feature Plane

- Euclidean distance-based clustering in 2D space
 - 2 objectives

Intra-cluster distances
are minimized

Inter-cluster distances
are maximized

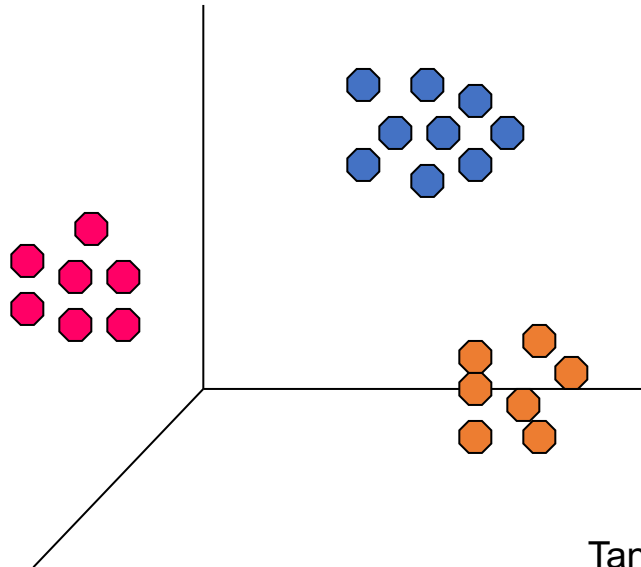


Clustering on a 3D Feature Plane

- Euclidean distance-based clustering in 3D space

Intra-cluster distances
are minimized

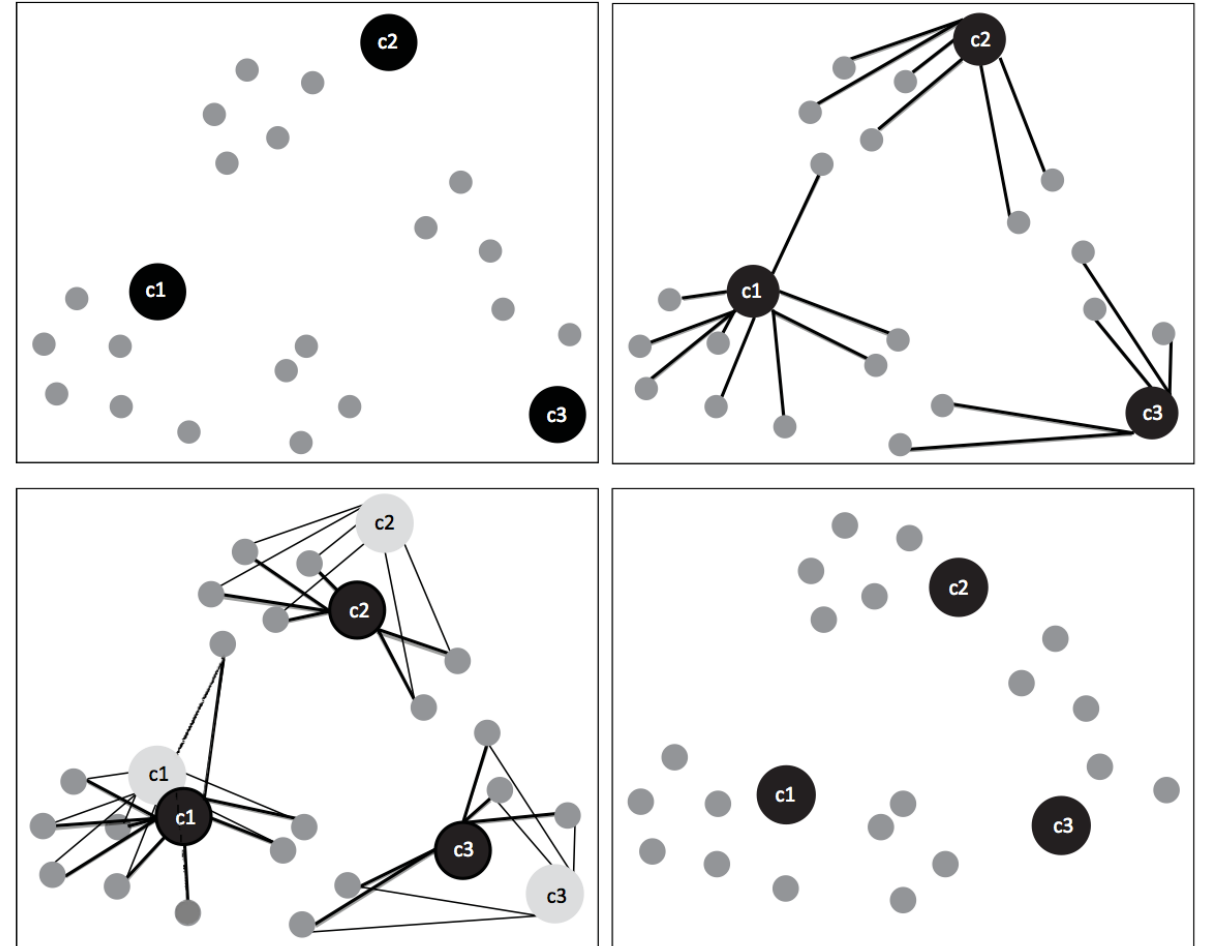
Inter-cluster distances
are maximized



A representative clustering algorithm: K-means

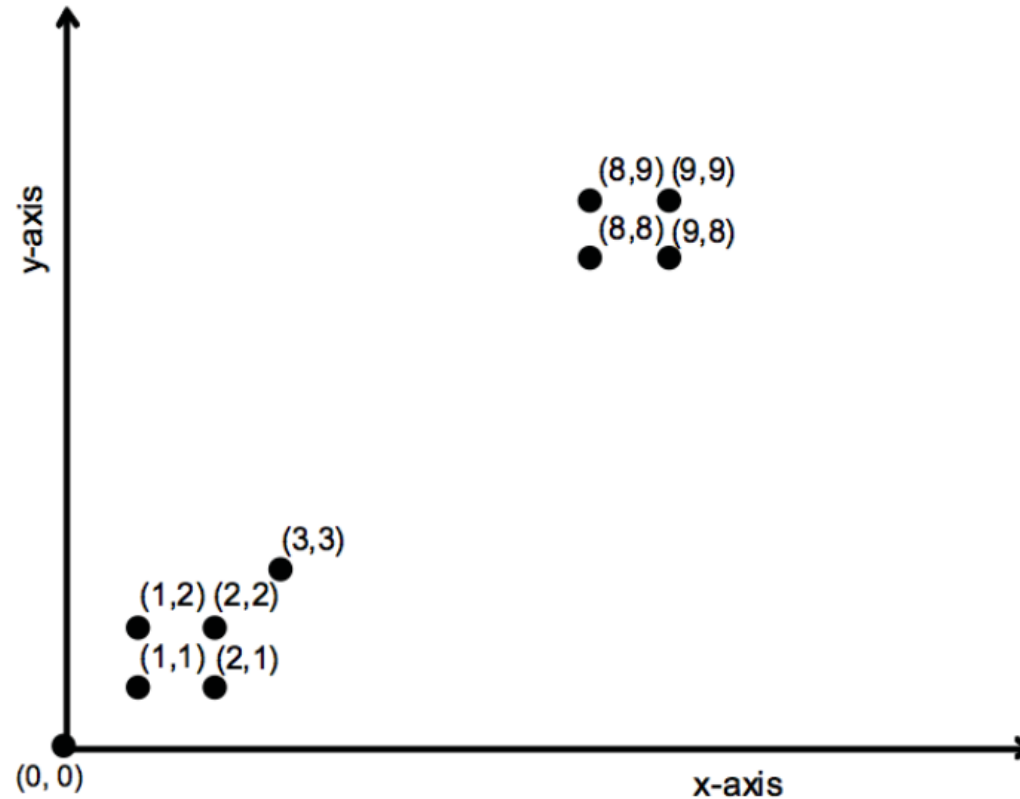
K-means clustering in action

- Starting with three random points as centroids (top left)
- The map stage (top right) assigns each point to the cluster nearest to it.
- In the reduce stage (bottom left), the associated points are averaged out to produce the new location of the centroid, leaving you with the final configuration (bottom right).
- After each iteration, the final configuration is fed back into the same loop until the centroids come to rest at their final positions.

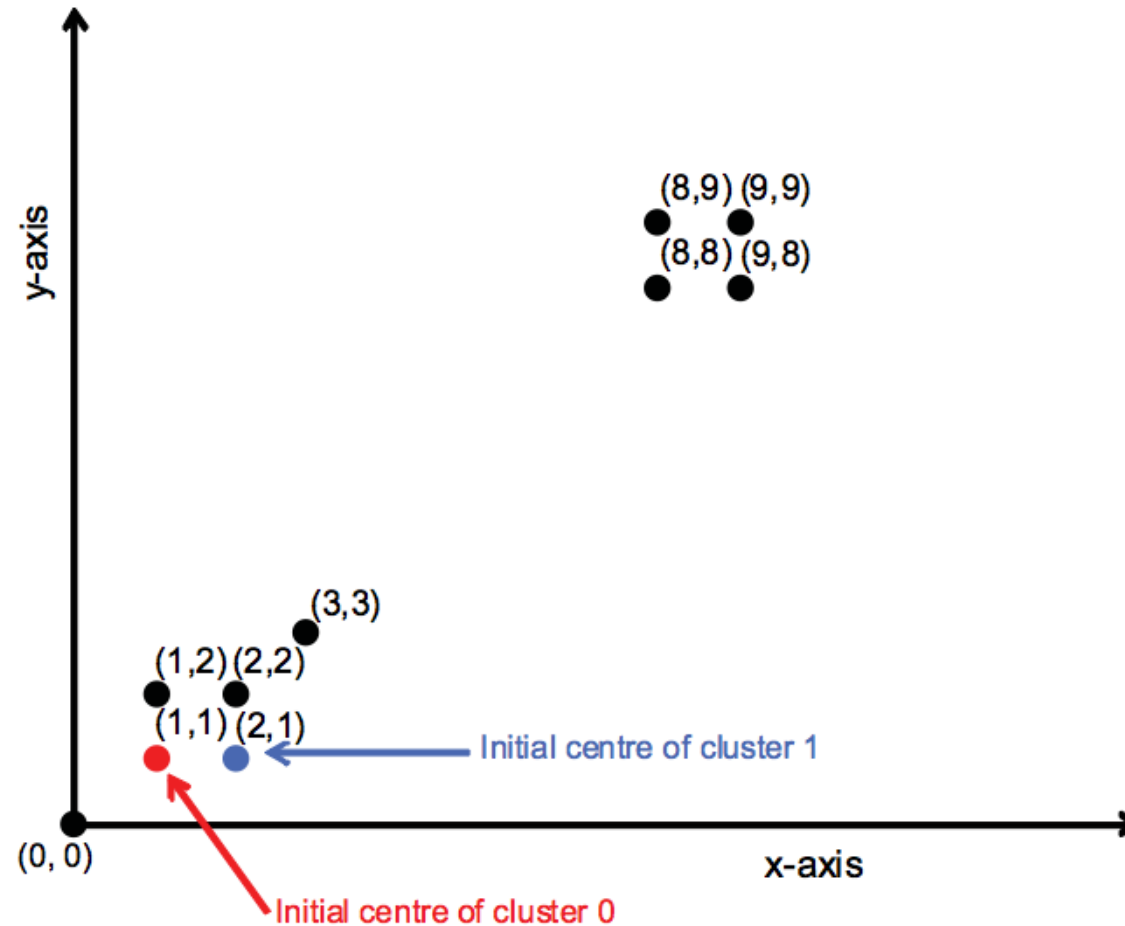


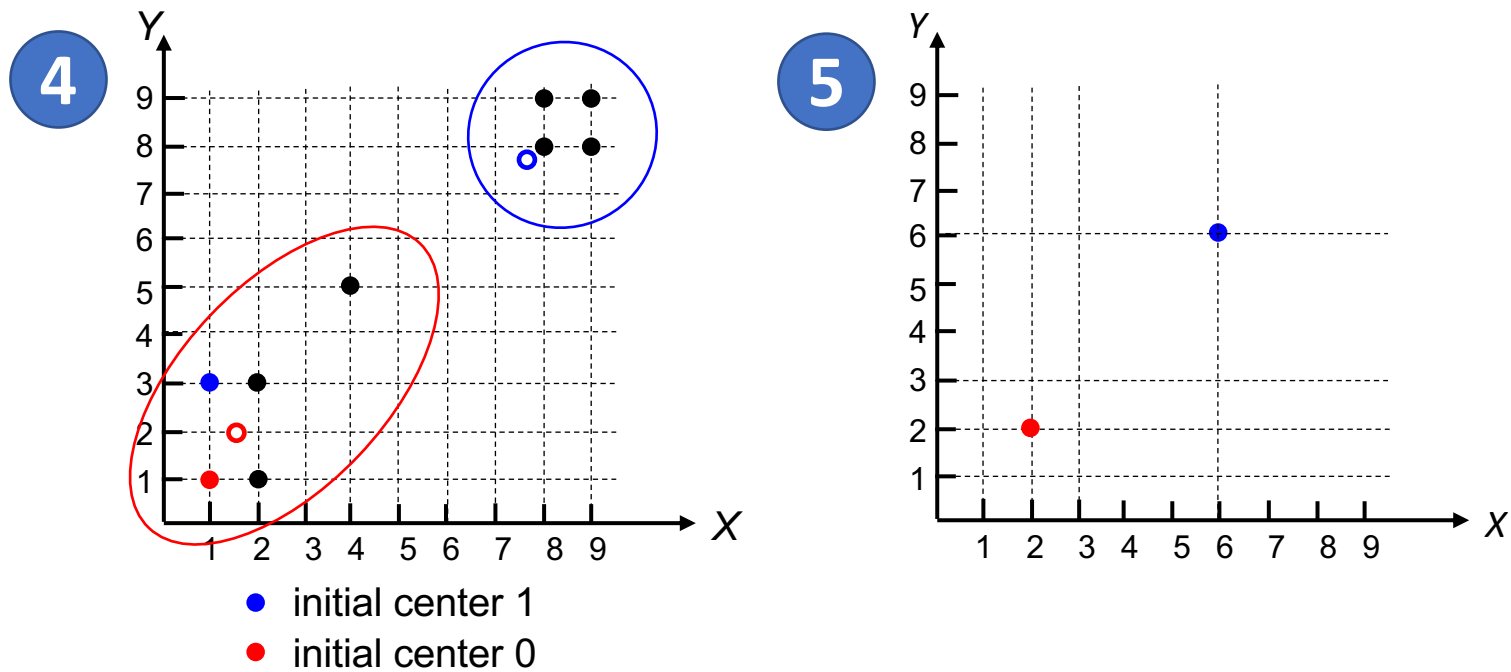
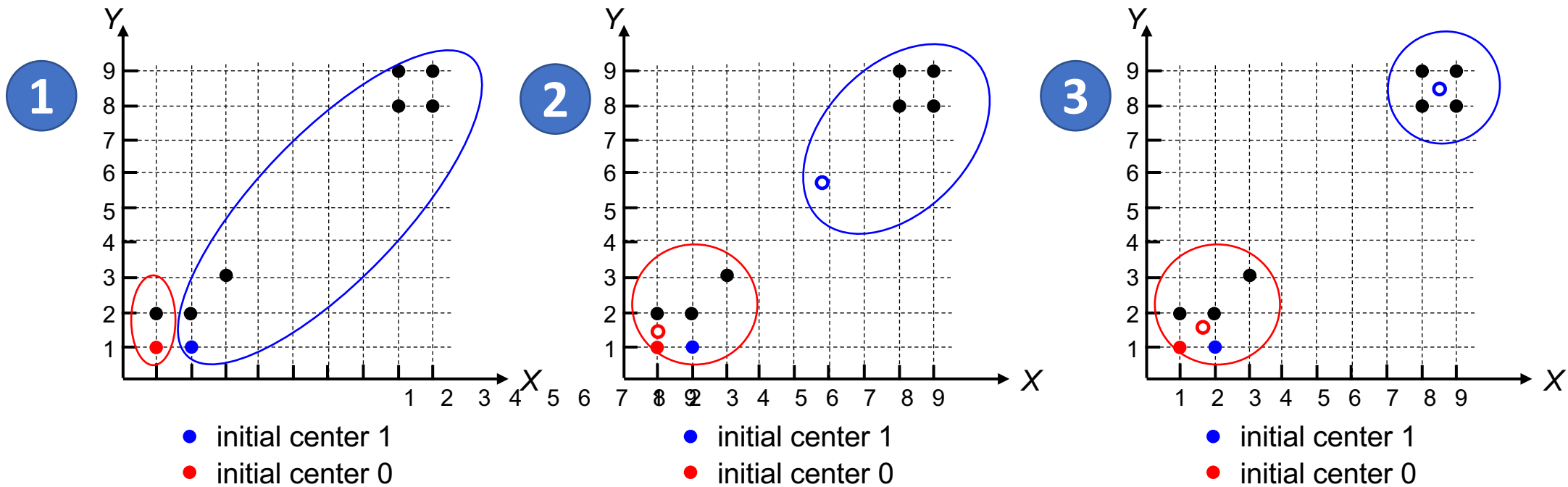
Clustering Example Using K-means

9 data points: (1, 1)
(2, 1)
(1, 2)
(2, 2)
(3, 3)
(8, 8)
(8, 9)
(9, 8)
(9, 9)



Making Initial Clustering Centers

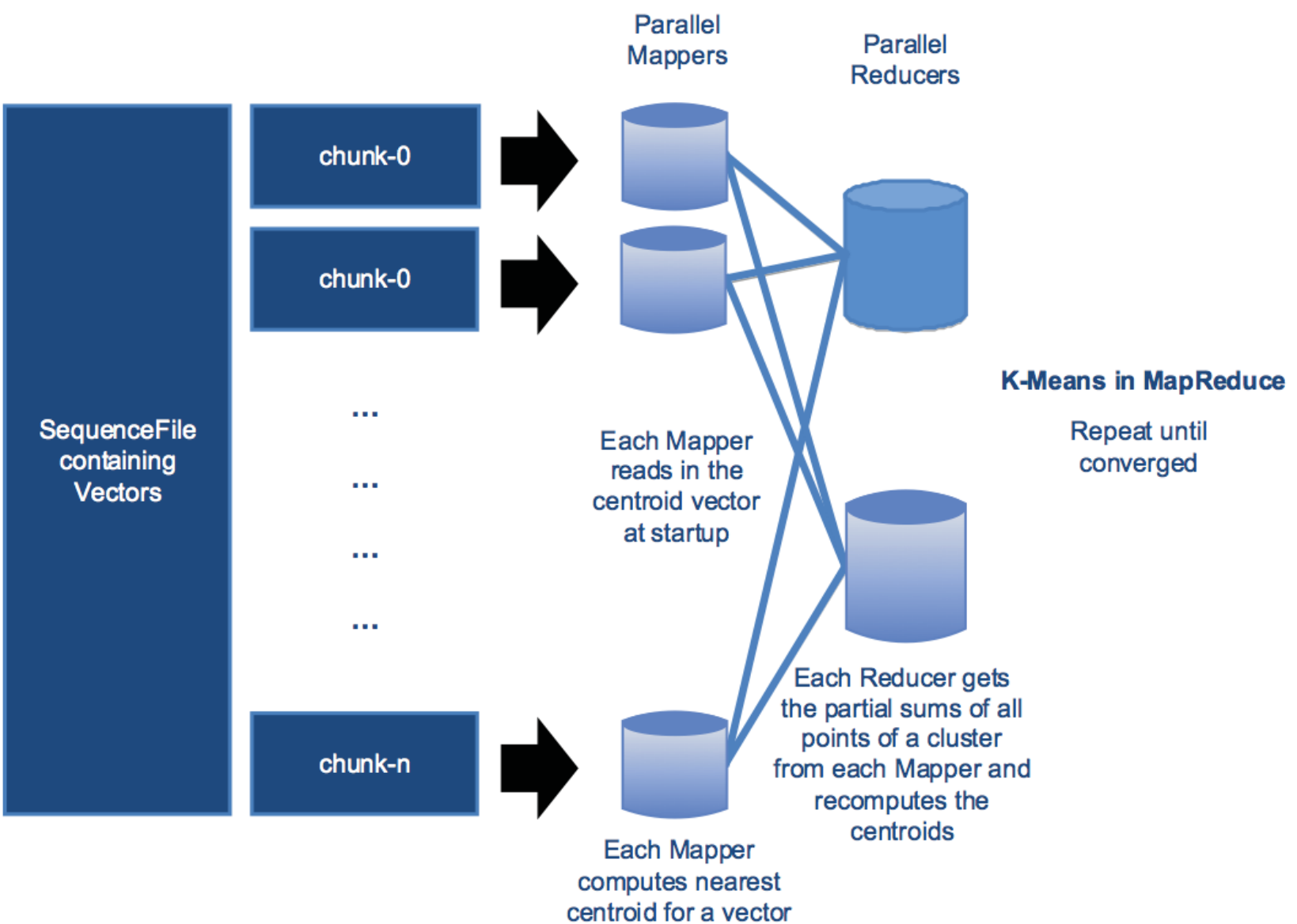




Hadoop k-means clustering jobs

- In Mahout, the MapReduce version of the K-means algorithm is instantiated using `KMeansDriver` class. The class has just a single entry point -- the `runJob` method.
 - The Hadoop configuration.
 - The `SequenceFile` containing the input `Vectors`.
 - The `SequenceFile` containing the initial `Cluster` centers.
 - The similarity measure to be used. We will use `EuclideanDistanceMeasure` as the measure of similarity and experiment with the others later.
 - The `convergenceThreshold`. If in an iteration, the centroids do not move more than this distance, no further iterations are done and clustering stops.
 - The number of iterations to be done. This is a hard limit; the clustering stops if this threshold is reached.

Implementation of K-means clustering using MapReduce



HelloWorld clustering scenario

```
public static final double[][] points = { {1, 1}, {2, 1}, {1, 2},
                                           {2, 2}, {3, 3}, {8, 8},
                                           {9, 8}, {8, 9}, {9, 9}};

public static void writePointsToFile(List<Vector> points,
                                     String fileName,
                                     FileSystem fs,
                                     Configuration conf) throws IOException {
    Path path = new Path(fileName);
    SequenceFile.Writer writer = new SequenceFile.Writer(fs, conf,
        path, LongWritable.class, VectorWritable.class);
    long recNum = 0;
    VectorWritable vec = new VectorWritable();
    for (Vector point : points) {
        vec.set(point);
        writer.append(new LongWritable(recNum++), vec);
    }
    writer.close();
}

public static List<Vector> getPoints(double[][] raw) {
    List<Vector> points = new ArrayList<Vector>();
    for (int i = 0; i < raw.length; i++) {
        double[] fr = raw[i];
        Vector vec = new RandomAccessSparseVector(fr.length);
        vec.assign(fr);
        points.add(vec);
    }
    return points;
}
```

HelloWorld clustering scenario -- II

```
public static void main(String args[]) throws Exception {  
    int k = 2;  
  
    List<Vector> vectors = getPoints(points);  
  
    File testData = new File("testdata");  
    if (!testData.exists()) {  
        testData.mkdir();  
    }  
    testData = new File("testdata/points");  
    if (!testData.exists()) {  
        testData.mkdir();  
    }  
  
    Configuration conf = new Configuration();  
    FileSystem fs = FileSystem.get(conf);  
    writePointsToFile(vectors,  
        "testdata/points/file1", fs, conf);  
  
    Path path = new Path("testdata/clusters/part-00000");  
    SequenceFile.Writer writer  
        = new SequenceFile.Writer(  
        fs, conf, path, Text.class, Cluster.class);  
  
    for (int i = 0; i < k; i++) {  
        Vector vec = vectors.get(i);  
        Cluster cluster = new Cluster(  
            vec, i, new EuclideanDistanceMeasure());  
        writer.append(new Text(cluster.getIdentifier()), cluster);  
    }  
    writer.close();  
}
```

Specify number
of clusters to
be formed

Create input
directories for data

Write initial centers

HelloWorld clustering scenario -- III

```
KMeansDriver.run(conf, new Path("testdata/points"),
    new Path("testdata/clusters"),
    new Path("output"), new EuclideanDistanceMeasure(),
    0.001, 10, true, false);

SequenceFile.Reader reader
    = new SequenceFile.Reader(fs,
        new Path("output/" + Cluster.CLUSTERED_POINTS_DIR
            + "/part-m-00000"), conf);

IntWritable key = new IntWritable();
WeightedVectorWritable value = new WeightedVectorWritable();
while (reader.next(key, value)) {
    System.out.println(
        value.toString() + " belongs to cluster "
            + key.toString());
}
reader.close();
}
```

← **Run k-means
algorithm**

← **Read output,
print vector,
cluster ID**

Hadoop k-means clustering code

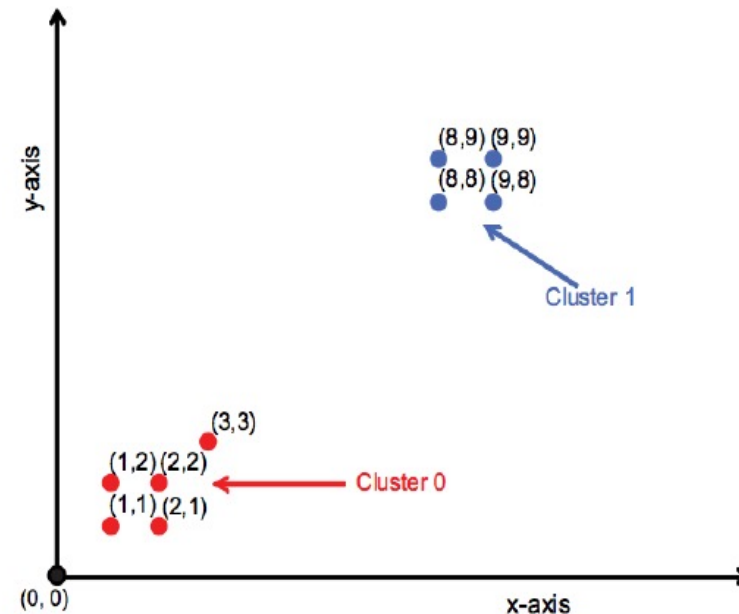
```
KmeansDriver.runJob(hadoopConf,  
    inputVectorFilesDirPath, clusterCenterFilesDirPath,  
    outputDir, new EuclideanDistanceMeasure(),  
    convergenceThreshold, numIterations, true, false);
```

TIP Mahout reads and writes data using the Hadoop `FileSystem` class. This provides seamless access to both the local filesystem (via `java.io`) and distributed filesystems like HDFS and S3FS (using internal Hadoop classes). This way, the same code that works on the local system will also work on the Hadoop filesystem on the cluster, provided the paths to the Hadoop configuration files are correctly set in the environment variables. In Mahout, the `bin/mahout` shell script finds the Hadoop configuration files automatically from the `$HADOOP_CONF` environment variable.

```
$ bin/mahout kmeans -i REUTERS-vectors/tfidf-vectors/ \  
-c REUTERS-initial-clusters \  
-o REUTERS-kmeans-clusters \  
-dm org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure \  
-cd 1.0 -k 20 -x 20 -cl
```

HelloWorld clustering scenario result

```
1.0: [1.000, 1.000] belongs to cluster 0
1.0: [2.000, 1.000] belongs to cluster 0
1.0: [1.000, 2.000] belongs to cluster 0
1.0: [2.000, 2.000] belongs to cluster 0
1.0: [3.000, 3.000] belongs to cluster 0
1.0: [8.000, 8.000] belongs to cluster 1
1.0: [9.000, 8.000] belongs to cluster 1
1.0: [8.000, 9.000] belongs to cluster 1
1.0: [9.000, 9.000] belongs to cluster 1
```



Testing Different Distance Measurements

Euclidean distance measure

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

Squared Euclidean distance measure

$$d = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2$$

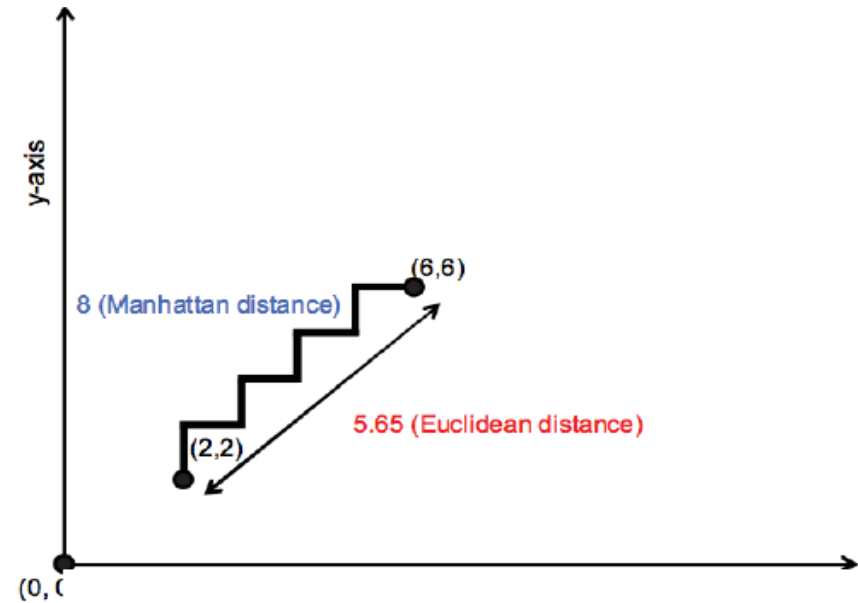
Manhattan distance measure

$$d = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$

Manhattan and Cosine Distances

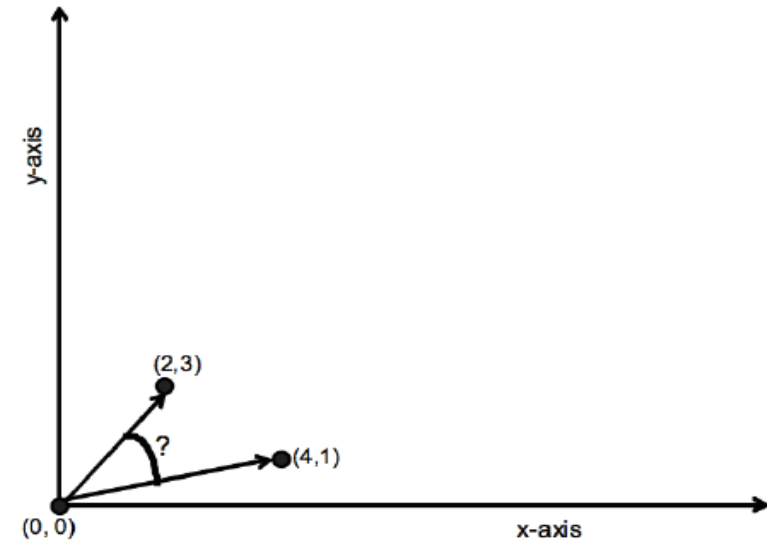
Manhattan distance measure

$$d = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$



Cosine distance measure

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}{(\sqrt{(a_1^2 + a_2^2 + \dots + a_n^2)} \sqrt{(b_1^2 + b_2^2 + \dots + b_n^2)})}$$



Tanimoto distance and weighted distance

Tanimoto distance measure

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}{\sqrt{(a_1^2 + a_2^2 + \dots + a_n^2)} + \sqrt{(b_1^2 + b_2^2 + \dots + b_n^2)} - (a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}$$

Weighted distance measure

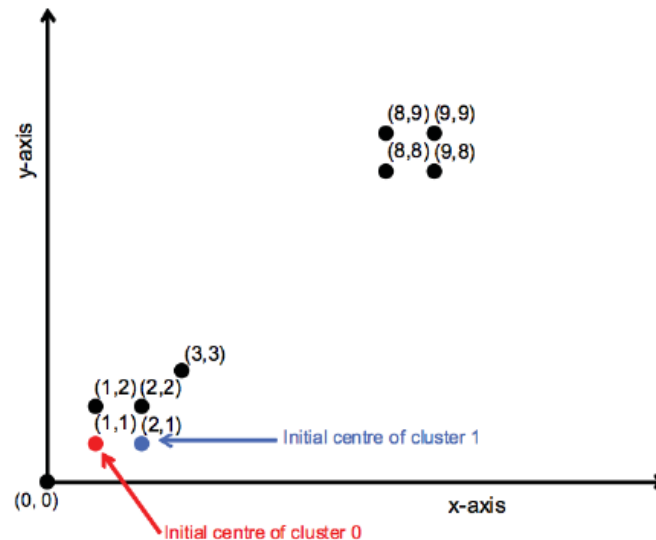
Mahout also provides a `WeightedDistanceMeasure` class, and implementations of Euclidean and Manhattan distance measures that use it. A weighted distance measure is an advanced feature in Mahout that allows you to give weights to different dimensions in order to either increase or decrease the effect of a dimension

Comparison of Clustering Results

Distance measure	Number of iterations	Vectors ^a in cluster 0	Vectors in cluster 1
EuclideanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
SquaredEuclideanDistanceMeasure	5	0, 1, 2, 3, 4	5, 6, 7, 8
ManhattanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
CosineDistanceMeasure	1	1	0, 2, 3, 4, 5, 6, 7, 8
TanimotoDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8

Why?

(1, 1)
 (2, 1)
 (1, 2)
 (2, 2)
 (3, 3)
 (8, 8)
 (8, 9)
 (9, 8)
 (9, 9)



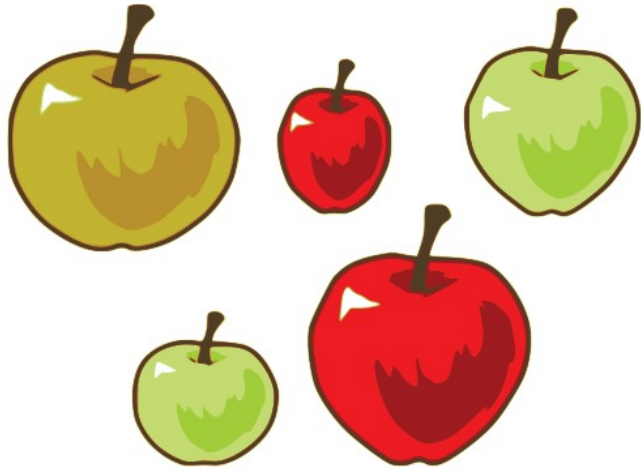
Data preparation in Mahout — vectors

- In Mahout, vectors are implemented as three different classes, each of which is optimized for different scenarios: `DenseVector`, `RandomAccessSparseVector`, and `SequentialAccessSparseVector`.
 - `DenseVector` can be thought of as an array of doubles, whose size is the number of features in the data. Because all the entries in the array are pre-allocated regardless of whether the value is 0 or not, we call it *dense*.
 - `RandomAccessSparseVector` is implemented as a `HashMap` between an integer and a double, where only nonzero valued features are allocated. Hence, they are called as `SparseVector`.
 - `SequentialAccessSparseVector` is implemented as two parallel arrays, one of integers and the other of doubles. Only nonzero valued entries are kept in it. Unlike the `RandomAccessSparseVector`, which is optimized for random access, this one is optimized for linear reading.

Vectorization

Computers only understand numbers

- Must represent non-numerical features with numbers



[0 => 100 gram, 1 => red, 2 => small]

Apple	Weight (kg) (0)	Color (1)	Size (2)	Vector
Small, round, green	0.11	510	1	[0.11, 510, 1]
Large, oval, red	0.23	650	3	[0.23, 650, 3]
Small, elongated, red	0.09	630	1	[0.09, 630, 1]
Large, round, yellow	0.25	590	3	[0.25, 590, 3]
Medium, oval, green	0.18	520	2	[0.18, 520, 2]

Canopy Clustering to Estimate the Number of Clusters

- K-means limitation: must know the number of clusters a priori
- Canopy
 - Tell what size clusters to look for (two thresholds)
 - The algorithm will find the number of clusters that have approximately that size
 - The algorithm uses two distance thresholds T_1 and T_2
 - This method prevents all points close to an already existing canopy from being the center of a new canopy

$T_1 > T_2$

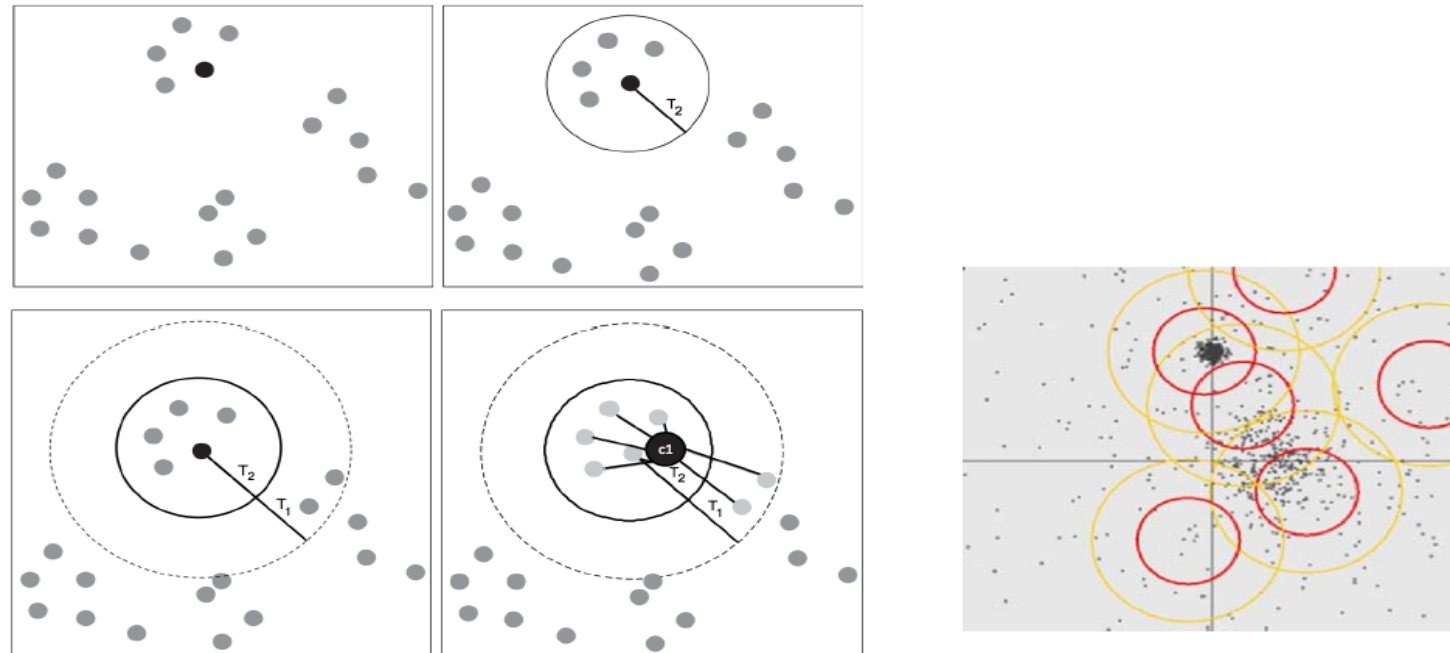
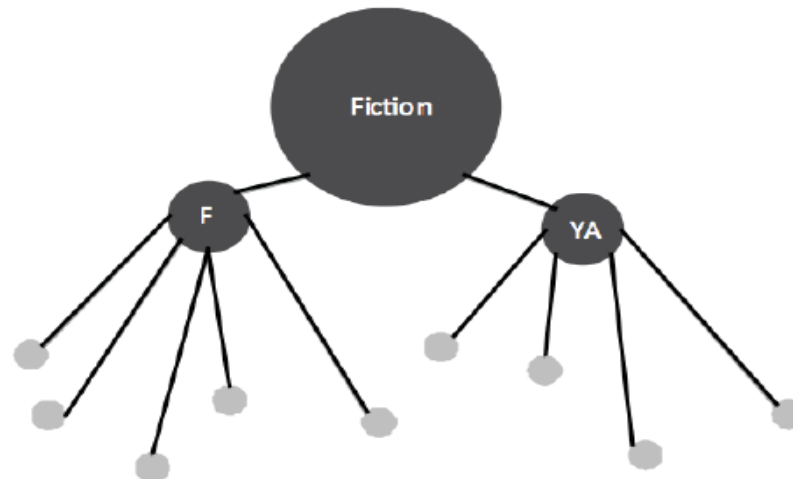
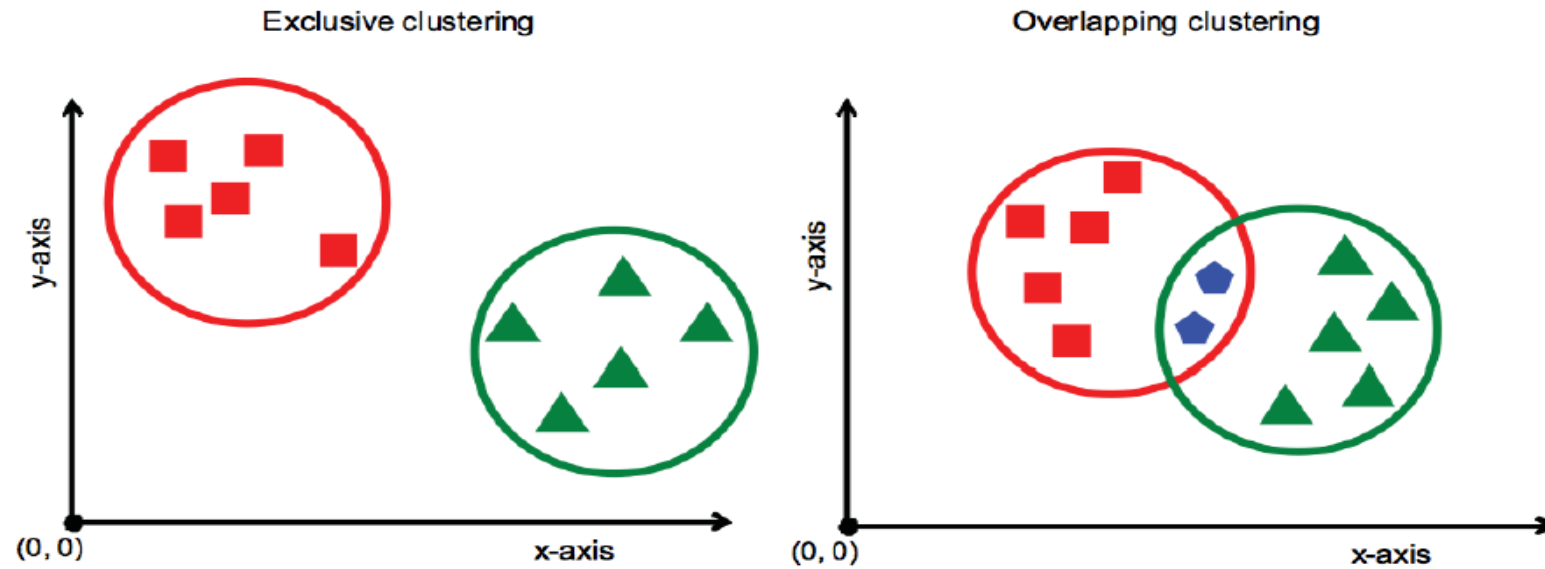


Figure 9.3 Canopy clustering: if you start with a point (top left) and mark it as part of a canopy, all the points within distance T_2 (top right) are removed from the data set and prevented from becoming new canopies. The points within the outer circle (bottom-right) are also put in the same canopy, but they're allowed to be part of other canopies. This assignment process is done in a single pass on a mapper. The reducer computes the average of the centroid (bottom right) and merges close canopies.

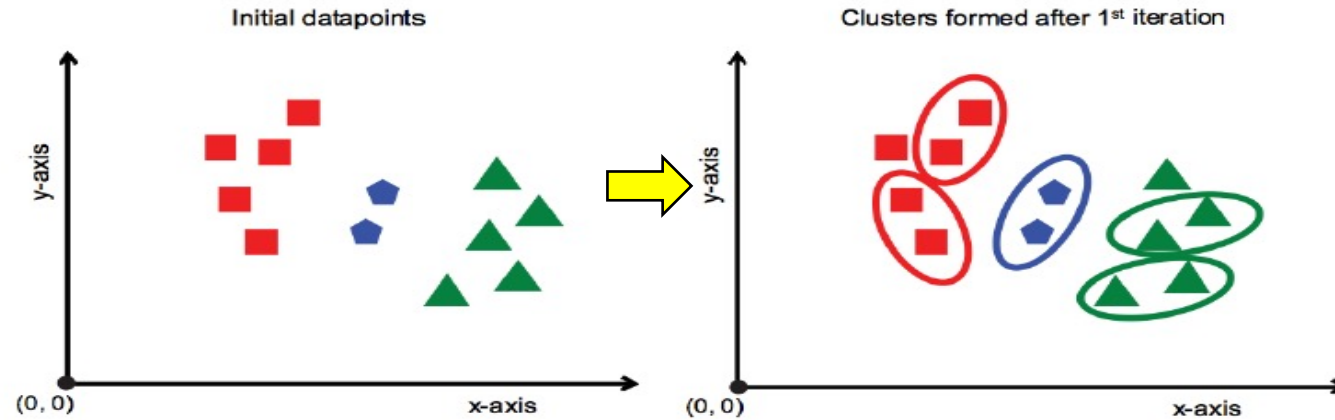
Other Clustering Algorithms



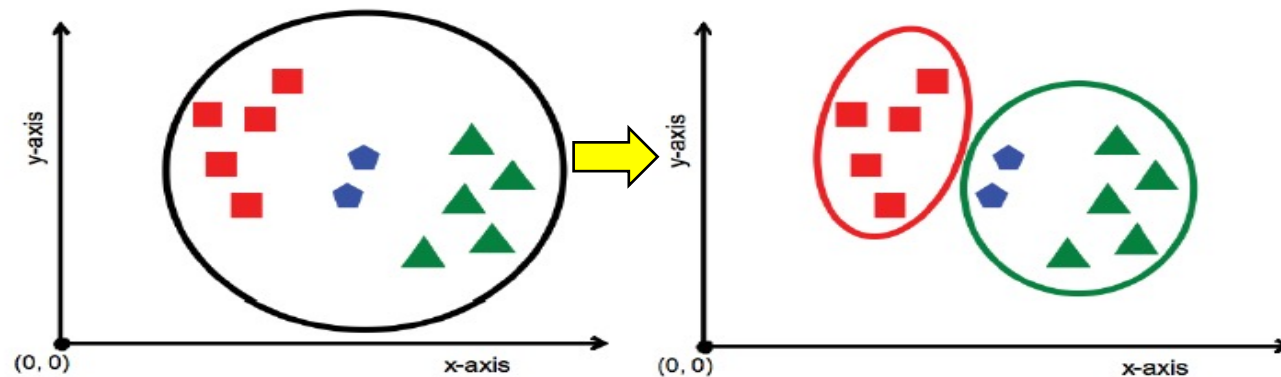
Different Clustering Approaches

FIXED NUMBER OF CENTERS

BOTTOM-UP APPROACH: FROM POINTS TO CLUSTERS VIA GROUPING



TOP-DOWN APPROACH: SPLITTING THE GIANT CLUSTER

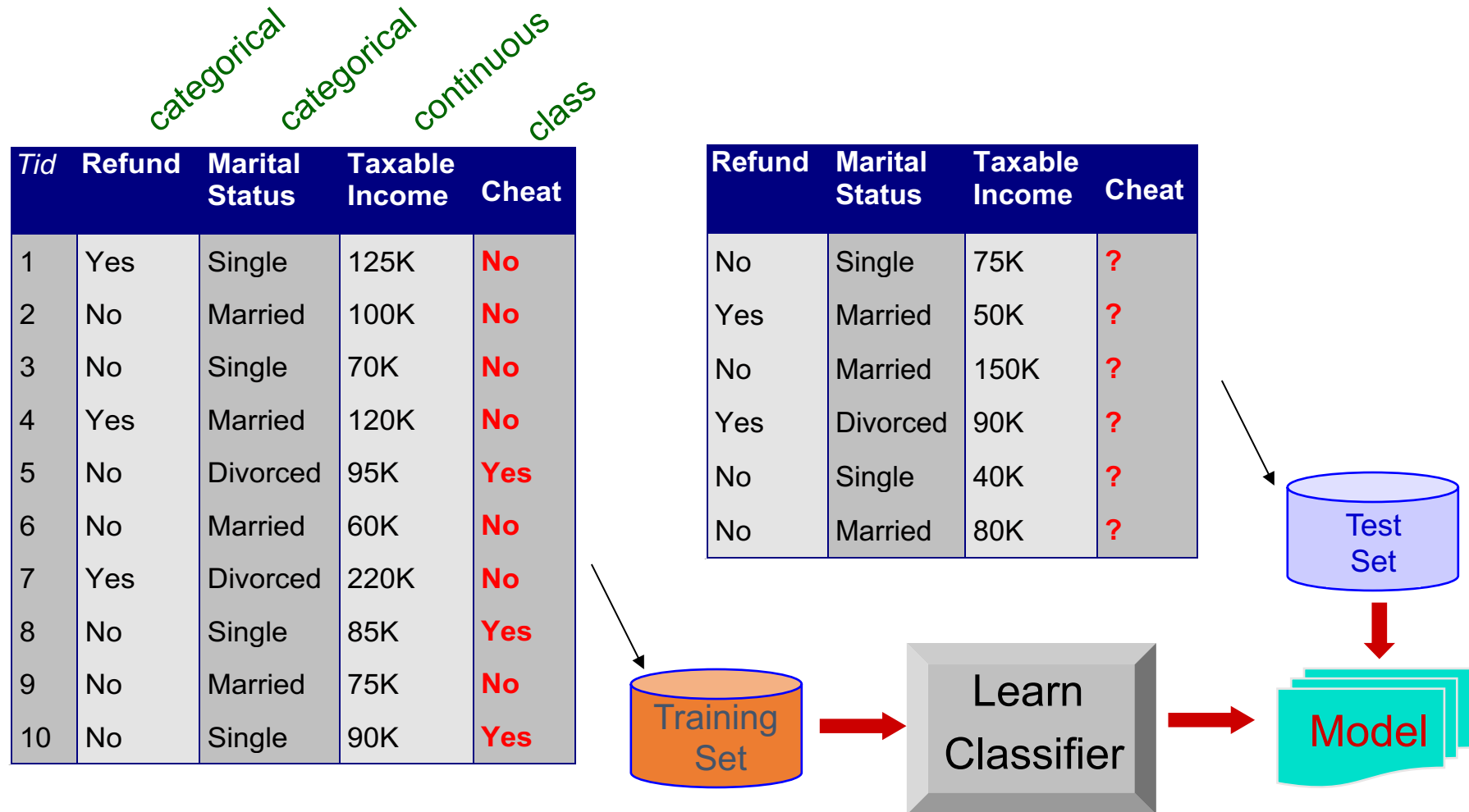


Classification - Definition

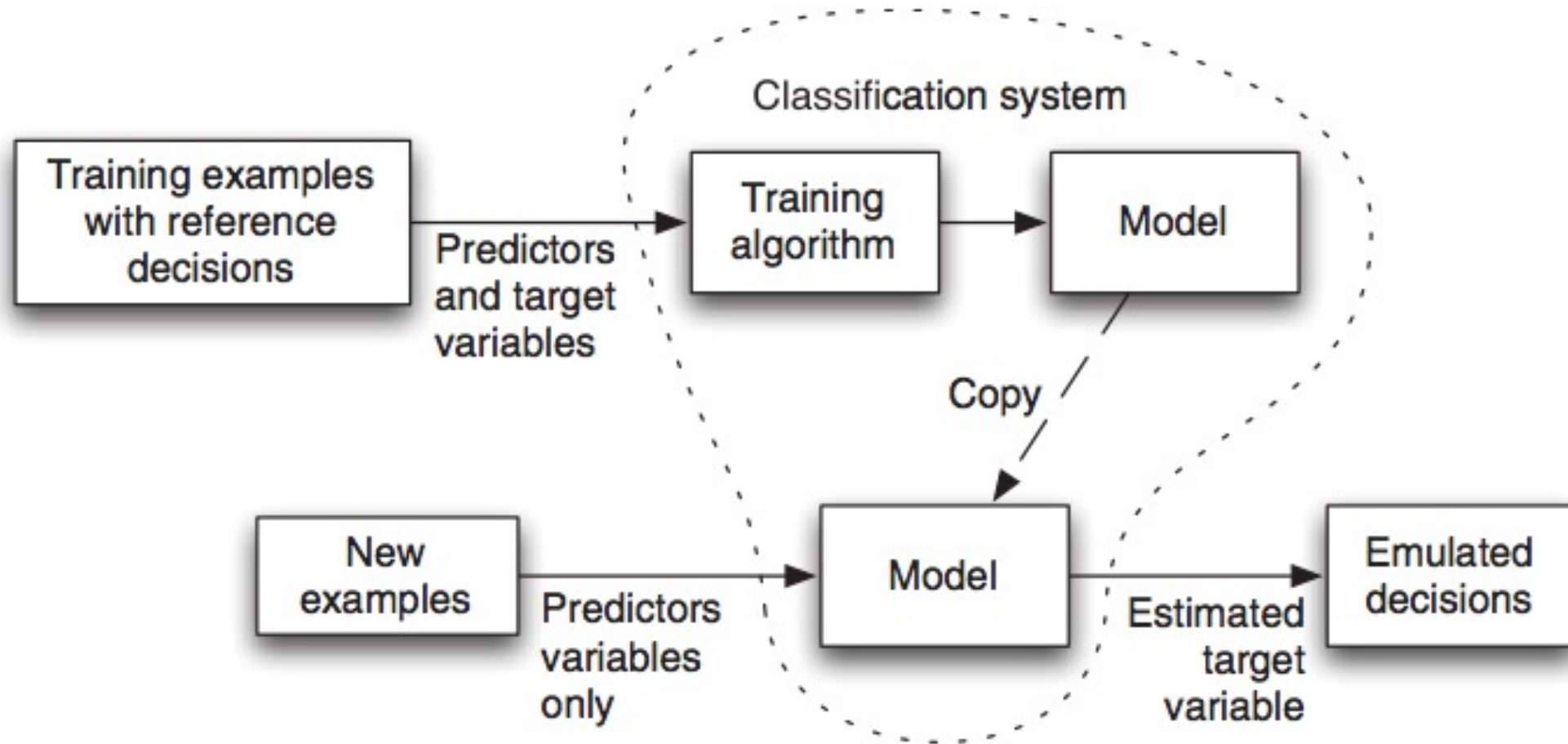
Computer classification systems are a form of machine learning that use learning algorithms to provide a way for computers to make decision based on experience and, in the process, emulate certain forms of human decision making

- Given a collection of records (*training set*)
 - Each record contains a set of *attributes*, one of the attributes is the (label) *class*
- Find a *model* for the class attribute as a function of the values of other attributes and employ a *learning algorithm* to compute the model *parameters*
- Goal: *previously unseen* records should be assigned a class as accurately as possible
 - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it

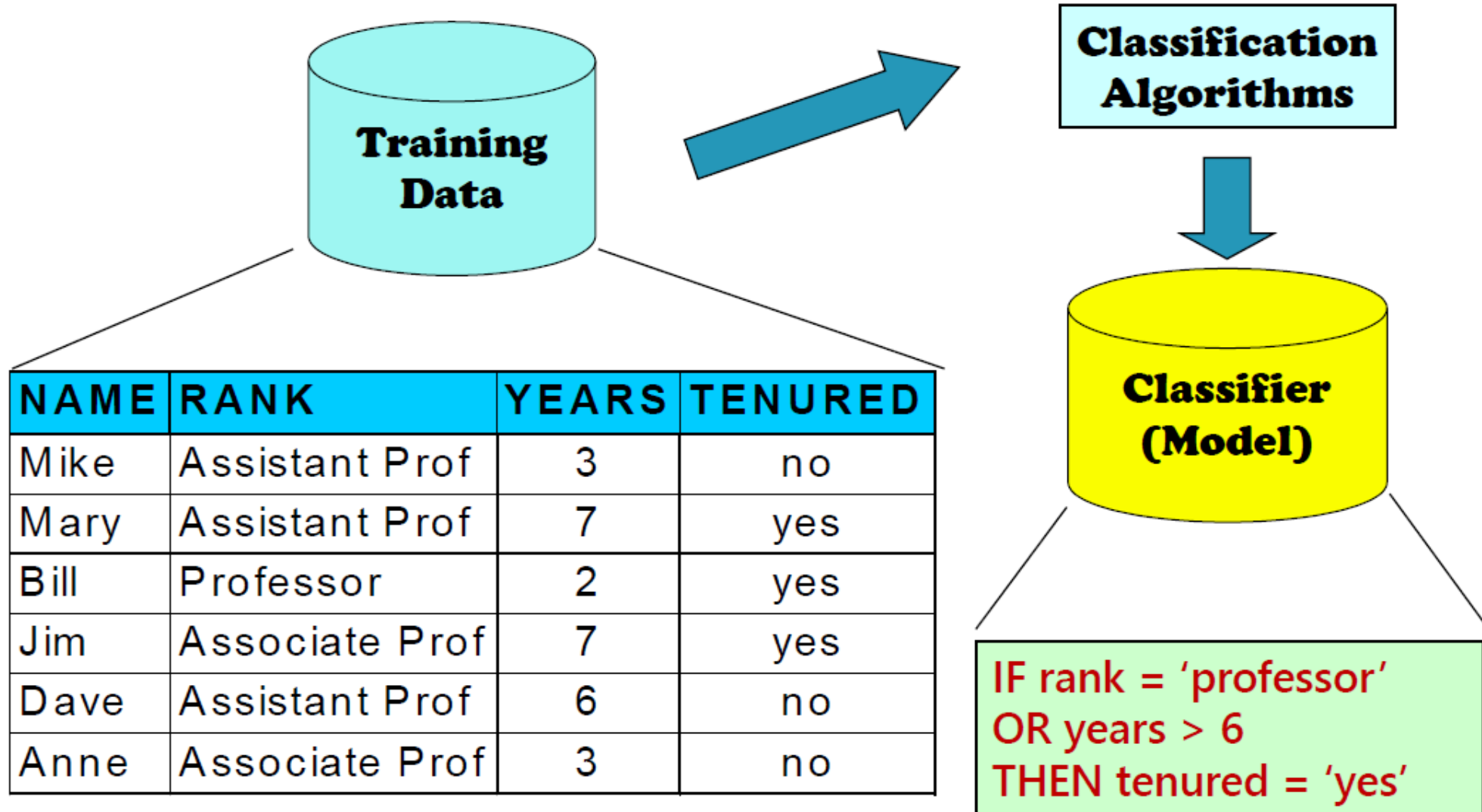
Classification - Example



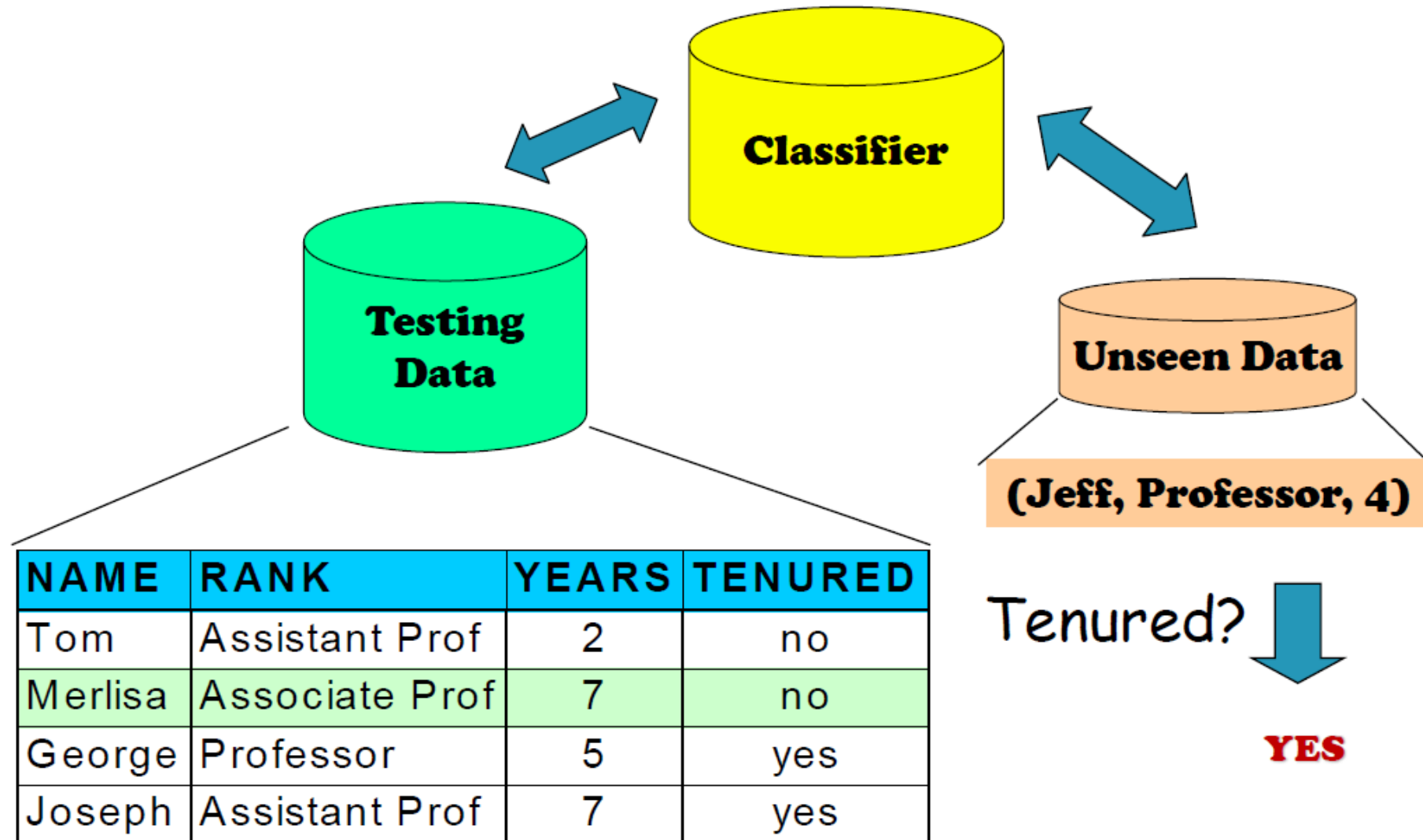
How does a classification system work?



Process 1: Model Construction



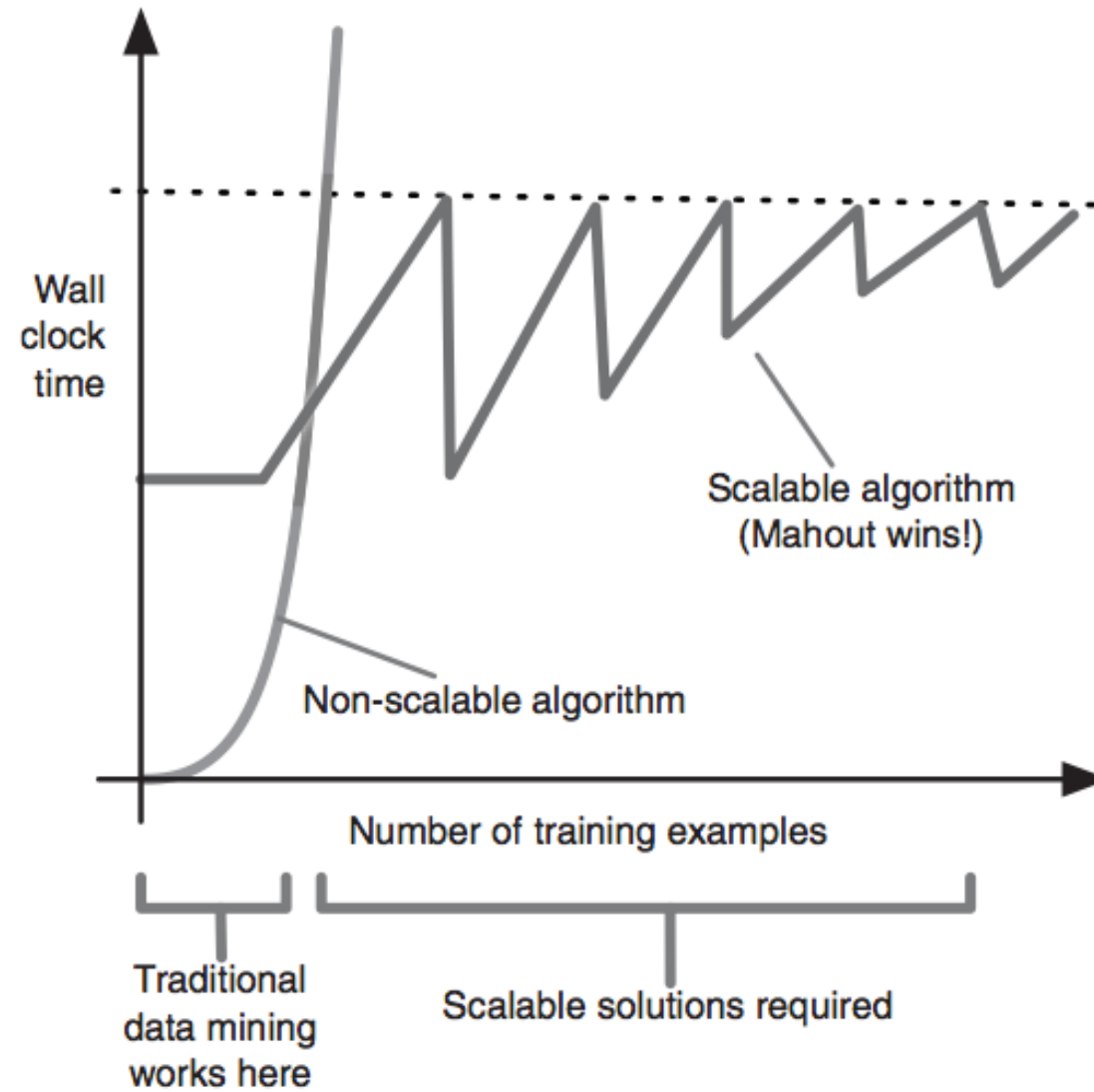
Process 2: Using the Model in Prediction



When to use Mahout for classification?

System size in number of examples	Choice of classification approach
< 100,000	Traditional, non-Mahout approaches should work very well. Mahout may even be slower for training.
100,000 to 1 million	Mahout begins to be a good choice. The flexible API may make Mahout a preferred choice, even though there is no performance advantage.
1 million to 10 million	Mahout is an excellent choice in this range.
> 10 million	Mahout excels where others fail.

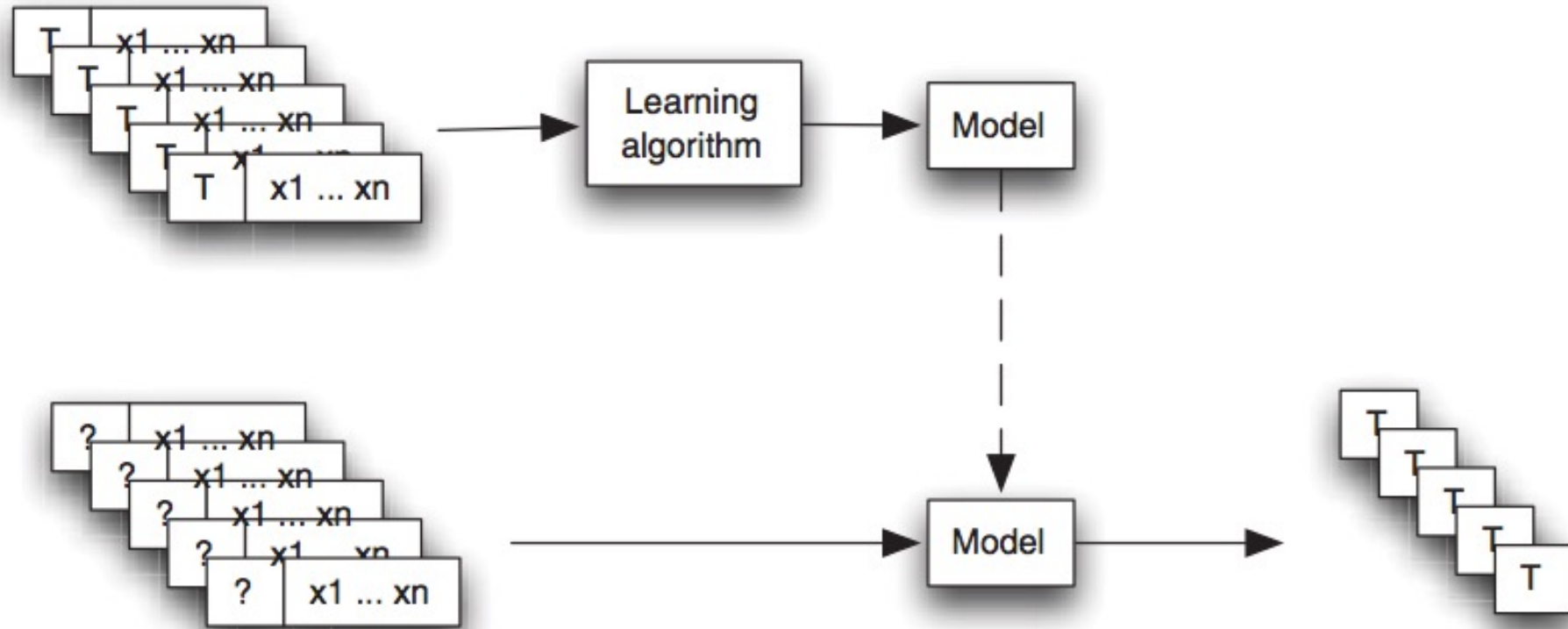
**The
advantage of
using Mahout
for
classification**



Key terminology for classification

Key Idea	Description
Model	A computer program that makes decisions; in classification, the output of the training algorithm is a model.
Training data	A subset of training examples labeled with the value of the target variable and used as input to the learning algorithm to produce the model.
Test data	A withheld portion of the training data with the value of the target variable hidden so that it can be used to evaluate the model.
Training	The learning process that uses training data to produce a model. That model can then compute estimates of the target variable given the predictor variables as inputs.
Training example	An entity with features that will be used as input for learning algorithm.
Feature	A known characteristic of a training or a new example; a feature is equivalent to a characteristic.
Variable	In this context, the value of a feature or a function of several features. This usage is somewhat different from the use of <i>variable</i> in a computer program.
Record	A container where an example is stored; such a record is composed of fields.
Field	Part of a record that contains the value of a feature (a variable).
Predictor variable	A feature selected for use as input to a classification model. Not all features need be used. Some features may be algorithmic combinations of other features.
Target variable	A feature that the classification model is attempting to estimate: the target variable is categorical, and its determination is the aim of the classification system.

Input and Output of a classification model



Four types of values for predictor variables

Type of value	Description
Continuous	This is a floating-point value. This type of value might be a price, a weight, a time, or anything else that has a numerical magnitude and where this magnitude is the key property of the value.
Categorical	A categorical value can have one of a set of prespecified values. Typically the set of categorical values is relatively small and may be as small as two, although the set can be quite large. Boolean values are generally treated as categorical values. Another example might be a vendor ID.
Word-like	A word-like value is like a categorical value, but it has an open-ended set of possible values.
Text-like	A text-like value is a sequence of word-like values, all of the same kind. Text is the classic example of a text-like value, but a list of email addresses or URLs is also text-like.

Sample data that illustrates all four value types

Name	Type	Value
from-address	Word-like	George <george@fumble-tech.com>
in-address-book?	Categorical (TRUE, FALSE)	TRUE
non-spam-words	Text-like	Ted, Mahout, User, lunch
spam-words	Text-like	available
unknown-words	Continuous	0
message-length	Continuous	31

Supervised vs. Unsupervised Learning

- Classification algorithms are related to, but still quite different from, clustering algorithms such as the k-means algorithm described in previous notes.
 - Classification algorithms are a form of supervised learning, as opposed to unsupervised learning, which happens with clustering algorithm.
 - A supervised learning algorithm is one that's given examples that contain the desired value of a target variable.
 - Unsupervised algorithms aren't given the desired answer, but instead must find something plausible on their own.
- **Supervised and unsupervised learning algorithms can often be usefully combined.**
 - A clustering algorithm can be used to create feature that can then be used by a learning algorithm, or the output of several classifiers can be used as features by a clustering algorithm.
 - Clustering systems often build a model that can be used to categorize new data. This clustering system model works much like the model produced by a classification system.
 - The difference lies in what data was used to produce the model.
 - For classification, the training data set includes the target variables; for clustering, the training data set doesn't include target variables.

Workflow in a typical classification project

Stage	Step
1. Training the model	Define target variable. Collect historical data. Define predictor variables. Select a learning algorithm. Use the learning algorithm to train the model.
2. Evaluating the model	Run test data. Adjust the input (use different predictor variables, different algorithms, or both).
3. Using the model in production	Input new examples to estimate unknown target values. Retrain the model as needed.

Mahout classification algorithms

- Mahout classification algorithms include:
 - Naive Bayesian
 - Complementary Naive Bayesian
 - Stochastic Gradient Descent (SDG)
 - Support Vector Machine (SVM)
 - Random Forest



**Choose
algorithm
via
Mahout**

Size of data set	Mahout algorithm	Execution model	Characteristics
Small to medium (less than tens of millions of training examples)	Stochastic gradient descent (SGD) family: OnlineLogisticRegression, CrossFoldLearner, AdaptiveLogisticRegression	Sequential, online, incremental	Uses all types of predictor variables; sleek and efficient over the appropriate data range (up to millions of training examples)
Medium to large (millions to hundreds of millions of training examples)	Support vector machine (SVM)	Sequential	Experimental still; sleek and efficient over the appropriate data range
	Naive Bayes	Parallel	Strongly prefers text-like data; medium to high overhead for training; effective and useful for data sets too large for SGD or SVM
	Complementary naive Bayes	Parallel	Somewhat more expensive to train than naive Bayes; effective and useful for data sets too large for SGD, but has similar limitations to naive Bayes
Small to medium (less than tens of millions of training examples)	Random forests	Parallel	Uses all types of predictor variables; high overhead for training; not widely used (yet); costly but offers complex and interesting classifications, handles nonlinear and conditional relationships in data better than other techniques

Stochastic Gradient Descent (SGD)

Both **statistical estimation** and **machine learning** consider the problem of minimizing an **objective function** that has the form of a sum:

$$Q(w) = \sum_{i=1}^n Q_i(w),$$

where the **parameter** w is to be **estimated** and where typically each summand function $Q_i()$ is associated with the i -th **observation** in the **data set** (used for training).

- Choose an initial vector of parameters w and learning rate α .
- Randomly shuffle examples in the training set.
- Repeat until an approximate minimum is obtained:
 - For $i = 1, 2, \dots, n$, do:
 - $w := w - \alpha \nabla Q_i(w)$.

Let's suppose we want to fit a straight line $y = w_1 + w_2x$ to a training set of two-dimensional points $(x_1, y_1), \dots, (x_n, y_n)$ using **least squares**. The objective function to be minimized is:

$$Q(w) = \sum_{i=1}^n Q_i(w) = \sum_{i=1}^n (w_1 + w_2x_i - y_i)^2.$$

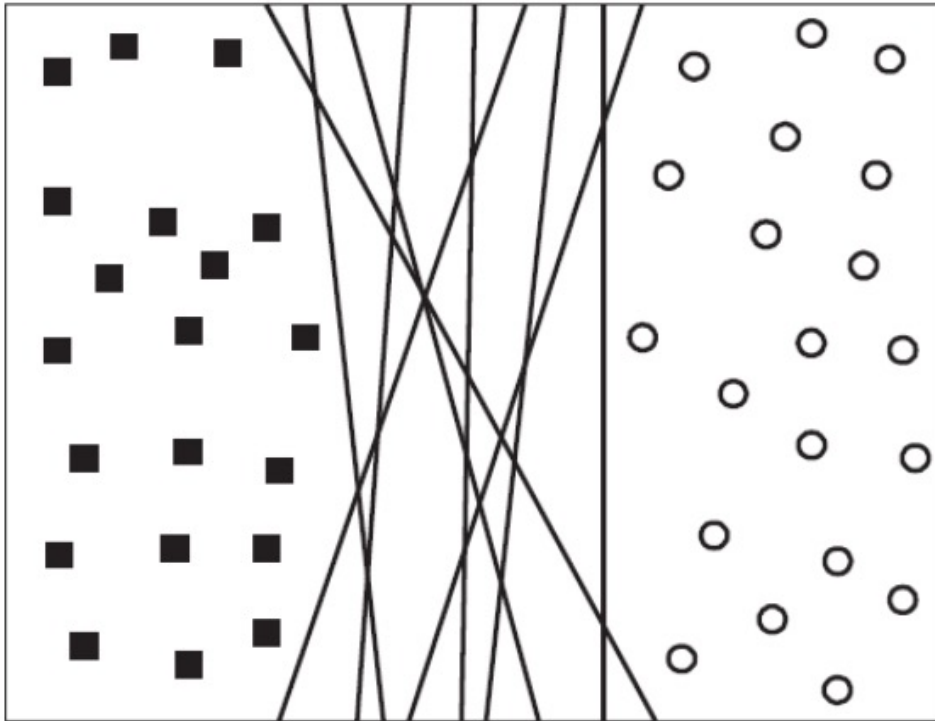
The last line in the above pseudocode for this specific problem will become:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \alpha \begin{bmatrix} \sum_{i=1}^n 2(w_1 + w_2x_i - y_i) \\ \sum_{i=1}^n 2x_i(w_1 + w_2x_i - y_i) \end{bmatrix}.$$

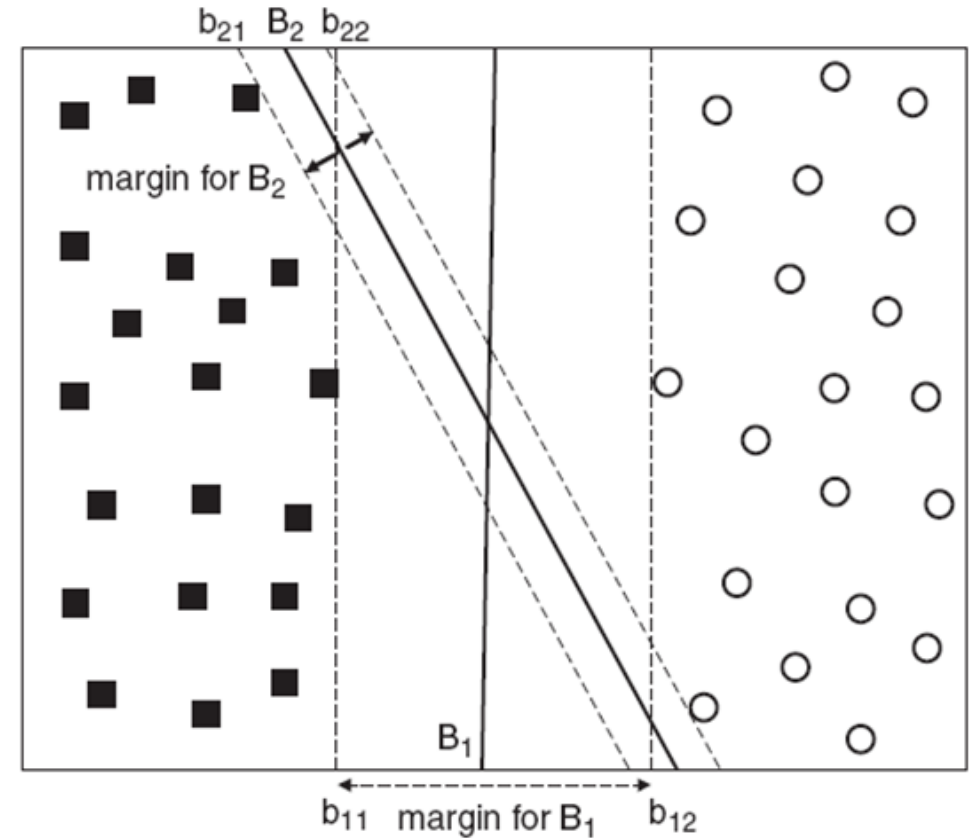
Stochastic Gradient Descent (SGD)

- Stochastic gradient descent (SGD) is a widely used learning algorithm, in which **each training example is used to tweak the model slightly to give a more correct answer for that one example.**
- This incremental approach is repeated over many training examples. With some special tricks to decide how much to nudge the model, the model accurately classifies new data after seeing only a modest number of examples.
- Although SGD algorithms are **difficult to parallelize** effectively, they are often so fast that for a wide variety of applications, parallel execution is not necessary.
- Importantly, because these algorithms do the same simple operation for each training example, they require a constant amount of memory. For this reason, each training example requires roughly the same amount of work. These properties make SGD-based algorithms scalable in the sense that twice as much data takes only twice as long to process.

Support Vector Machine (SVM)

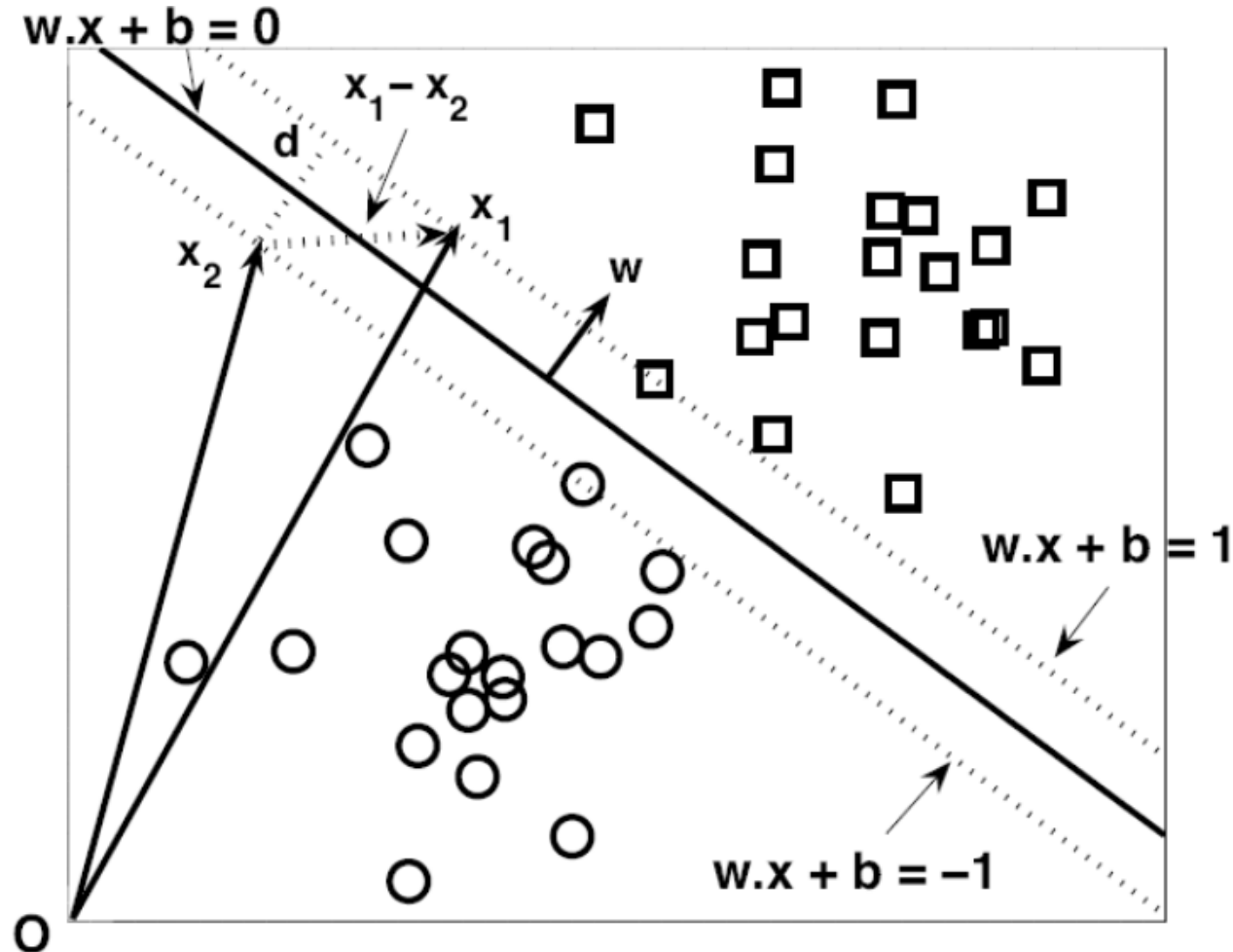


Possible decision boundaries for a linearly separable data set



Margin of a decision boundary

Support Vector Machine (SVM)

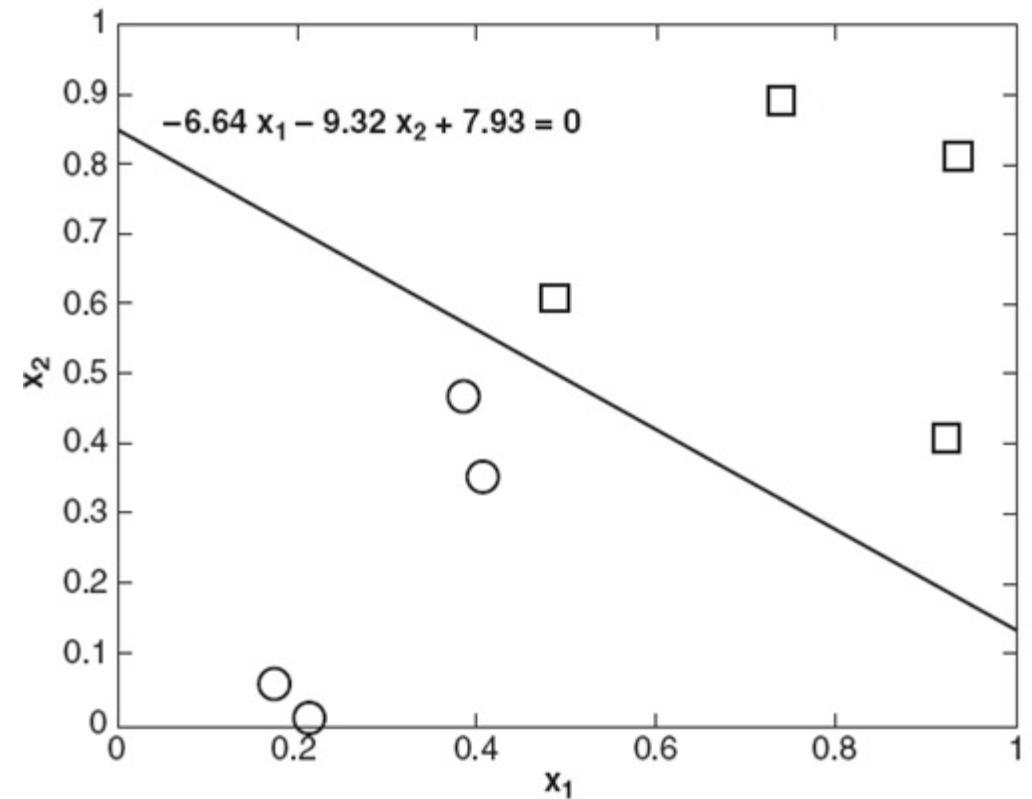


Decision boundary and margin of SVM

Support Vector Machine (SVM)

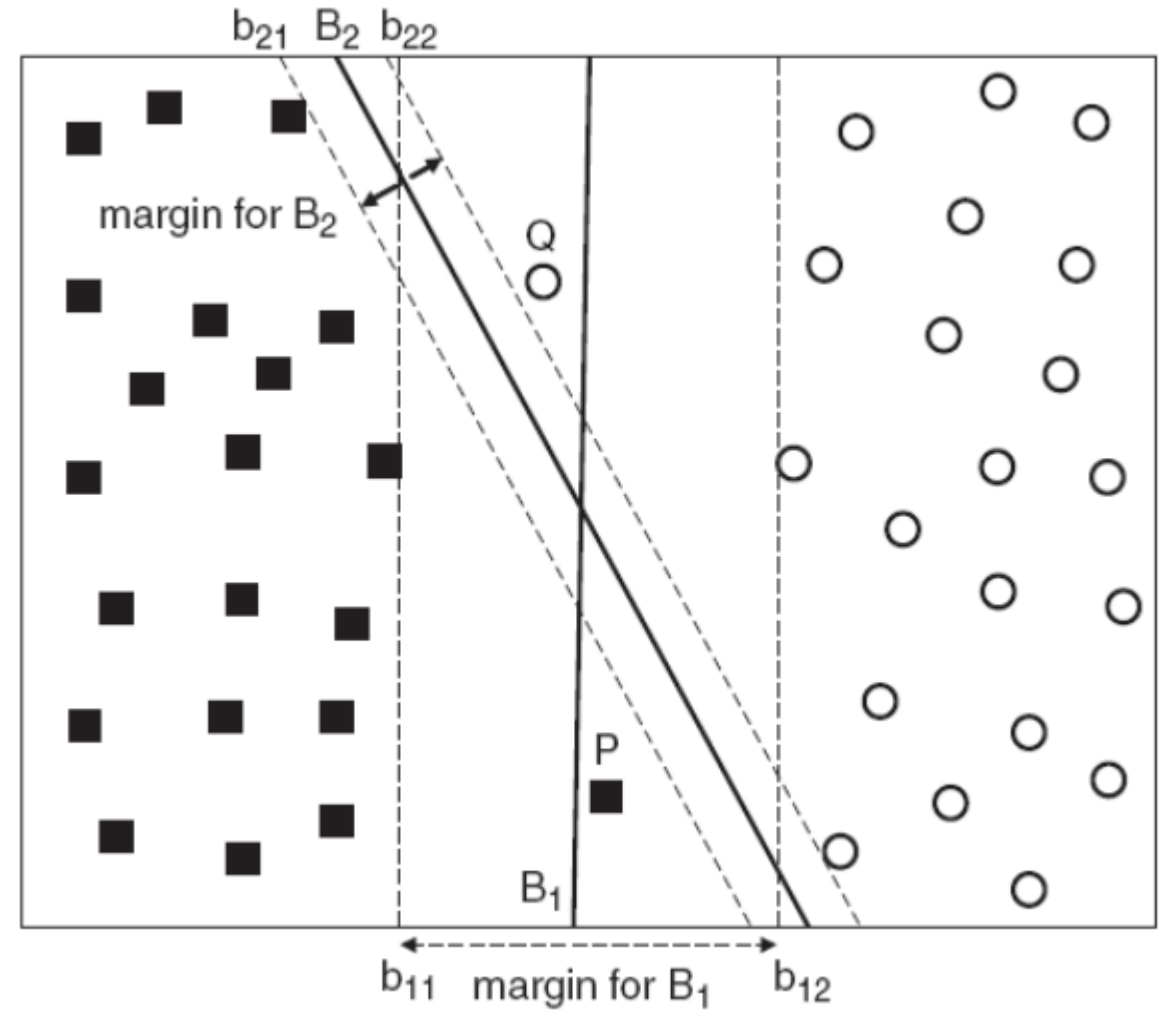
Example of a linearly separable data set

x_1	x_2	y	Lagrange Multiplier
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0



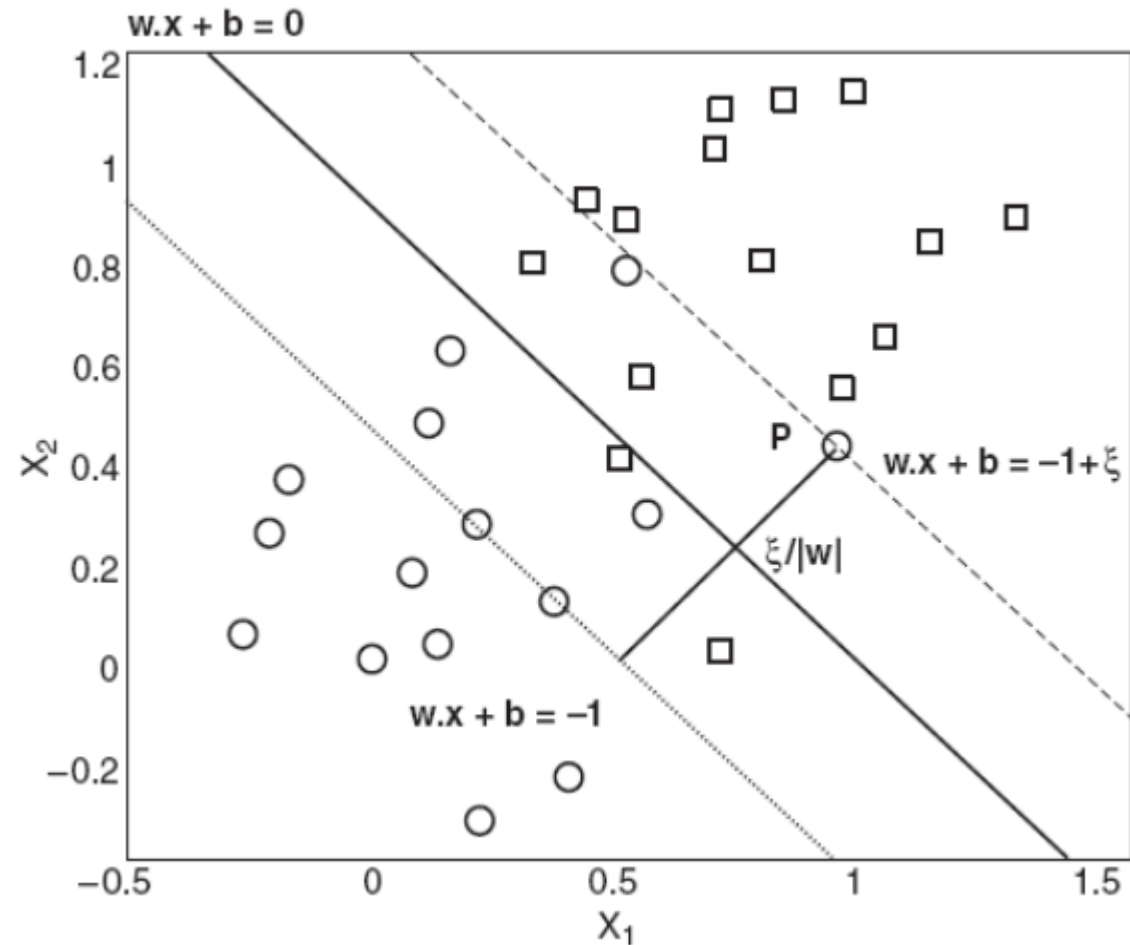
Support Vector Machine (SVM)

Decision boundary of SVM
for non-separable case



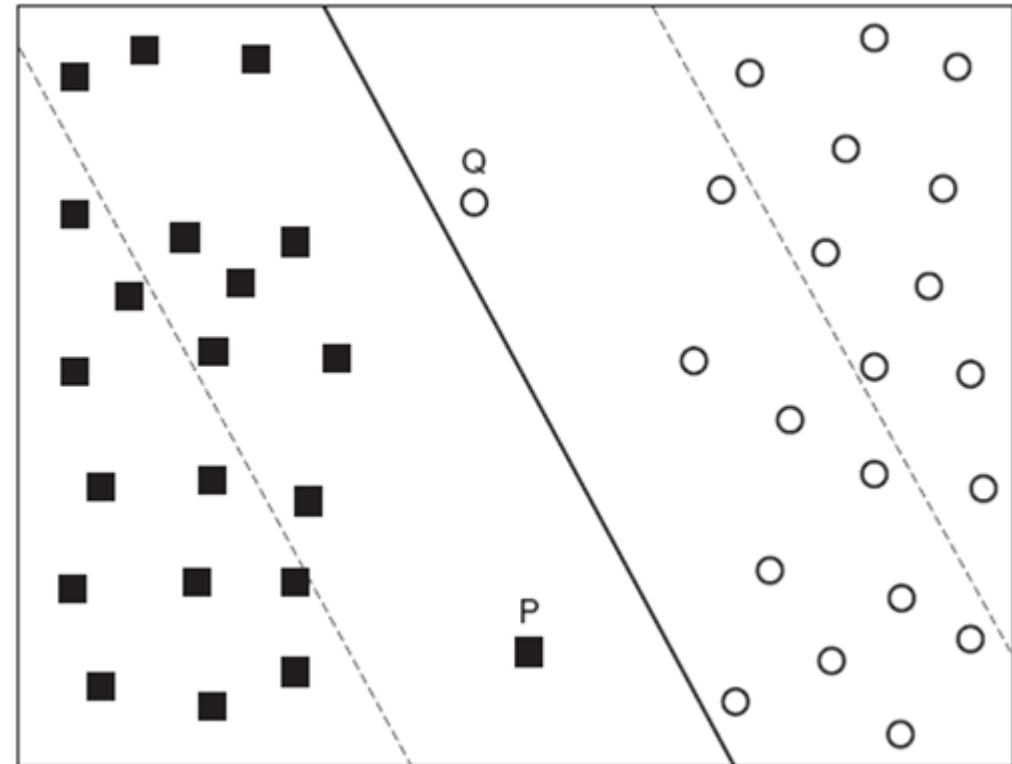
Support Vector Machine (SVM)

Slack variables for non-separable data



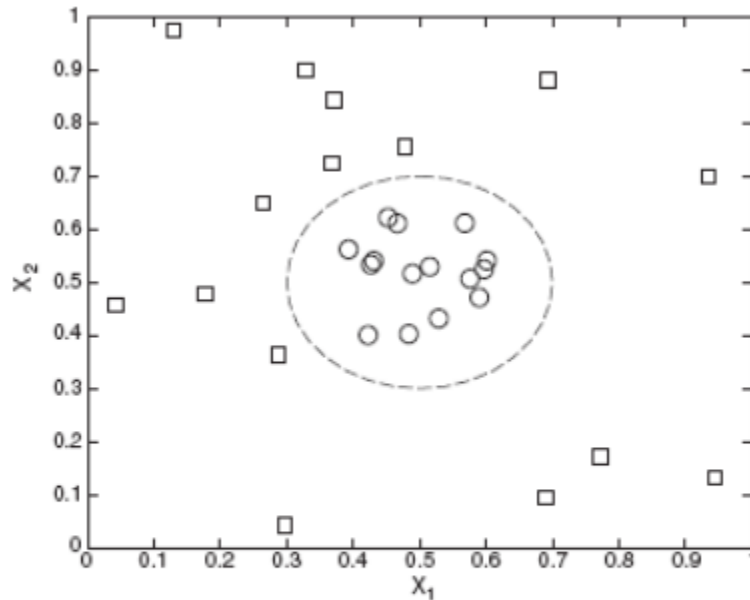
Support Vector Machine (SVM)

A decision boundary that has a wide margin but large training error

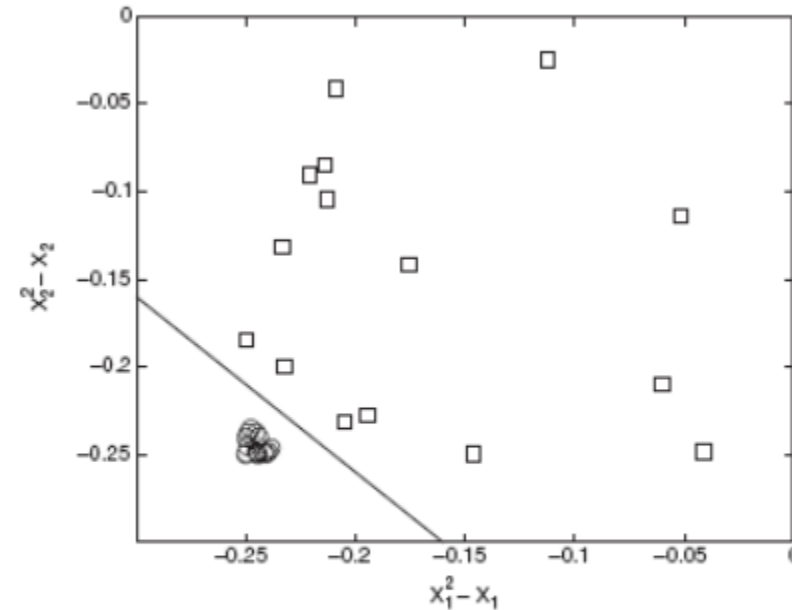


Support Vector Machine (SVM)

- Classifying data with a nonlinear decision boundary



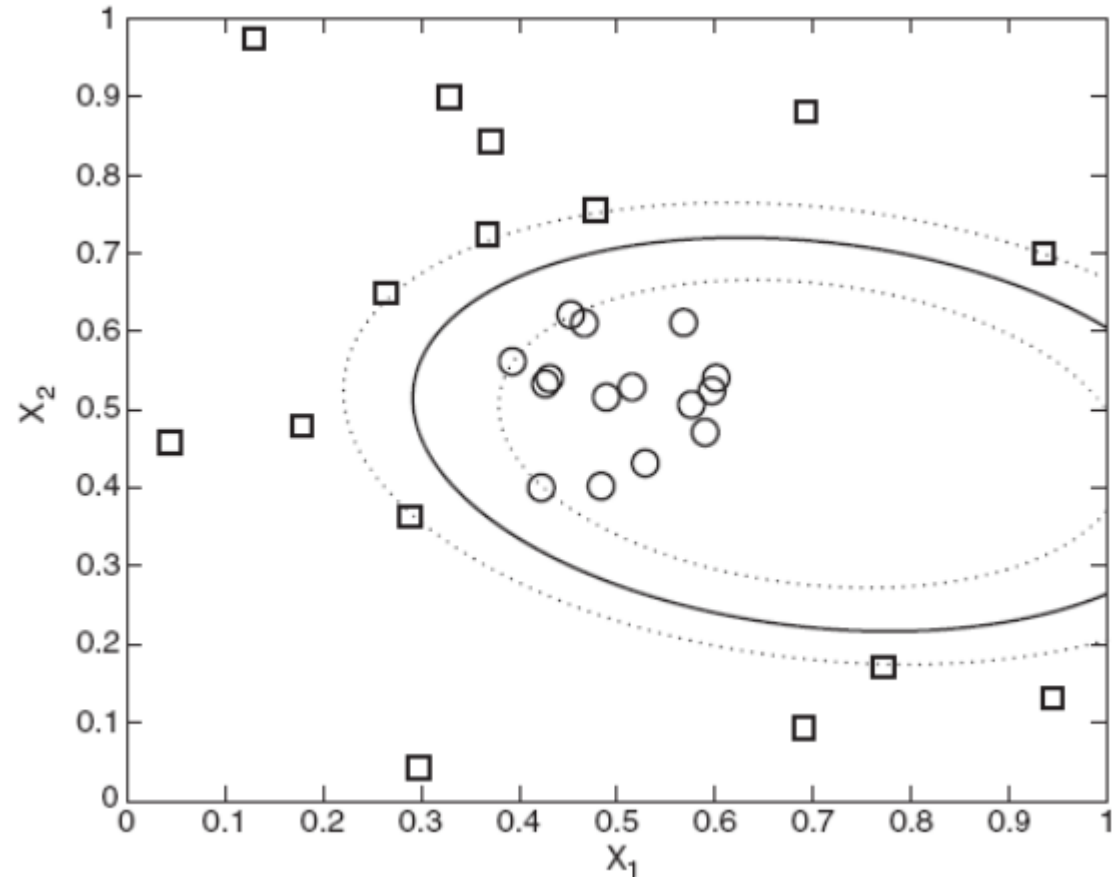
(a) Decision boundary in the original two-dimensional space.



(b) Decision boundary in the transformed space.

Support Vector Machine (SVM)

Decision boundary
produced by a nonlinear
SVM with polynomial kernel



Naive Bayes

Training set:

sex	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

Classifier using Gaussian distribution assumptions:

sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Test Set:

sex	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

$$\text{posterior}(\text{male}) = \frac{P(\text{male}) p(\text{height}|\text{male}) p(\text{weight}|\text{male}) p(\text{foot size}|\text{male})}{\text{evidence}}$$

$$\begin{aligned} \text{evidence} &= P(\text{male}) p(\text{height}|\text{male}) p(\text{weight}|\text{male}) p(\text{foot size}|\text{male}) \\ &+ P(\text{female}) p(\text{height}|\text{female}) p(\text{weight}|\text{female}) p(\text{foot size}|\text{female}) \end{aligned}$$

$$P(\text{male}) = 0.5$$

$$p(\text{height}|\text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6 - \mu)^2}{2\sigma^2}\right) \approx 1.5789,$$

$$p(\text{weight}|\text{male}) = 5.9881 \cdot 10^{-6}$$

$$p(\text{foot size}|\text{male}) = 1.3112 \cdot 10^{-3}$$

$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$

$$P(\text{female}) = 0.5$$

$$p(\text{height}|\text{female}) = 2.2346 \cdot 10^{-1}$$

$$p(\text{weight}|\text{female}) = 1.6789 \cdot 10^{-2}$$

$$p(\text{foot size}|\text{female}) = 2.8669 \cdot 10^{-1}$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

==> female

Example -- classifying 20 newsgroups data set (~ 20K docs)

- The 20 newsgroup data set is a standard data set commonly used for machine learning research. The data is from transcripts of several months of posting made in 20 Usenet newsgroups from the early 1990s. <http://qwone.com/~jason/20Newsgroups/>

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

- If you examine one of the files in the training data directory, such as 20news-bydate-train/sci.crypt/15524, you will see something like this:

```
From: rdippold@qualcomm.com (Ron "Asbestos" Dippold)
Subject: Re: text of White House announcement and Q&As
Originator: rdippold@qualcom.qualcomm.com
Nntp-Posting-Host: qualcom.qualcomm.com
Organization: Qualcomm, Inc., San Diego, CA
Lines: 12
```

```
ted@nmsu.edu (Ted Dunning) writes:
>nobody seems to have noticed that the clipper chip *must* have been
>under development for considerably longer than the 3 months that
>clinton has been president. this is not something that choosing
...
```

The predictor features are either headers or body

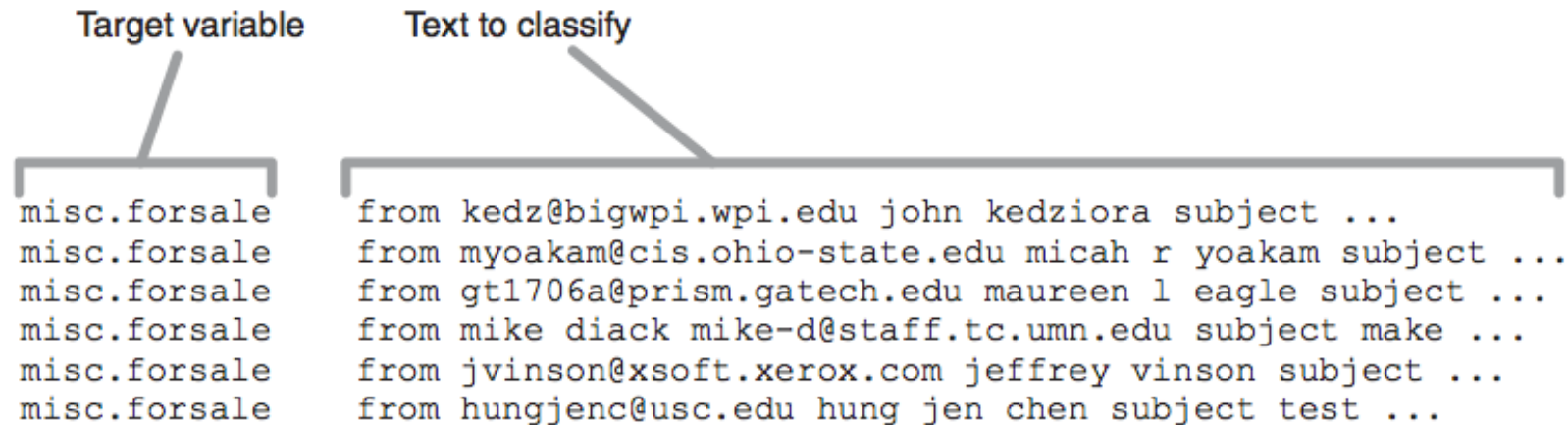
Classifying using Naive Bayes

```
$ bin/mahout prepare20newsgroups -p 20news-bydate-train/ \  
  -o 20news-train/ \  
  -a org.apache.lucene.analysis.standard.StandardAnalyzer \  
  -c UTF-8
```

```
no HADOOP_CONF_DIR or HADOOP_HOME set, running locally  
INFO: Program took 3713 ms
```

```
$ bin/mahout prepare20newsgroups -p 20news-bydate-test \  
  -o 20news-test \  
  -a org.apache.lucene.analysis.standard.StandardAnalyzer \  
  -c UTF-8
```

```
no HADOOP_CONF_DIR or HADOOP_HOME set, running locally  
INFO: Program took 2436 ms
```



Training and testing naive Bayes classifier

```
bin/mahout trainclassifier -i 20news-train \  
    -o 20news-model \  
    -type cbayes \  
    -ng 1 \  
    -source hdfs
```

...

```
INFO: Program took 250104 ms
```

The result is a model stored in the 20news-model directory, as specified by the `-o` option. The `-ng` option indicates that individual words are to be considered instead of short sequences of words. The model consists of several files that contain the components of the model. These files are in a binary format and can't be easily inspected directly, but you can use them to classify the test data with the help of the `testclassifier` program.

To run the naive Bayes model on the test data, you can use the following command:

```
bin/mahout testclassifier -d 20news-test \  
    -m 20news-model \  
    -type cbayes \  
    -ng 1 \  
    -source hdfs \  
    -method sequential
```

Here the `-m` option specifies the directory that contains the model built in the previous step. The `-method` option specifies that the program should be run in a sequential mode rather than using Hadoop. For small data sets like this one, sequential operation is preferred. For larger data sets, parallel operation becomes necessary to keep the runtime reasonably small.

See training results

When given this familiar data, the model is able to get nearly 98 percent correct, which is much too good to be true on this particular problem. The best machine learning researchers only claim accuracies for their systems around 84~86 percent.

```
bin/mahout testclassifier -d 20news-train -m 20news-model\  
  -type cbayes -ng 1 -source hdfs -method sequential  
...  
Correctly Classified Instances      :      11075    97.8876%  
Incorrectly Classified Instances   :         239     2.1124%  
Total Classified Instances         :      11314
```

Classification results

Confusion Matrix

```

-----
a b c d e f g h i j k l m n o p q r s t <--Classified as
388                                     | 397 a = rec.sport.baseball
  386                                     | 396 b = sci.crypt
    396                                     | 399 c = rec.sport.hockey
      347                                     | 364 d = talk.politics.guns
        377                                     | 398 e = soc.religion.christian
   12    304                                     | 393 f = sci.electronics
      281    14  43                                     | 394 g = comp.os.ms-windows.misc
        313    22 16                                     | 390 h = misc.forsale
   26 69    83 41                                     | 251 i = talk.religion.misc
      45    225 13                                     | 319 j = alt.atheism
        334    32                                     | 395 k = comp.windows.x
          367                                     | 376 l = talk.politics.mideast
   19 23 15    307 13                                     | 392 m =comp.sys.ibm.pc.hardware
      16 335                                     | 385 n = comp.sys.mac.hardware
        371                                     | 394 o = sci.space
          393                                     | 398 p = rec.motorcycles
   12 364                                     | 396 q = rec.autos
      11    22    305                                     | 389 r = comp.graphics
   102    160                                     | 310 s = talk.politics.misc
      362 | 396 t = sci.med
-----

```

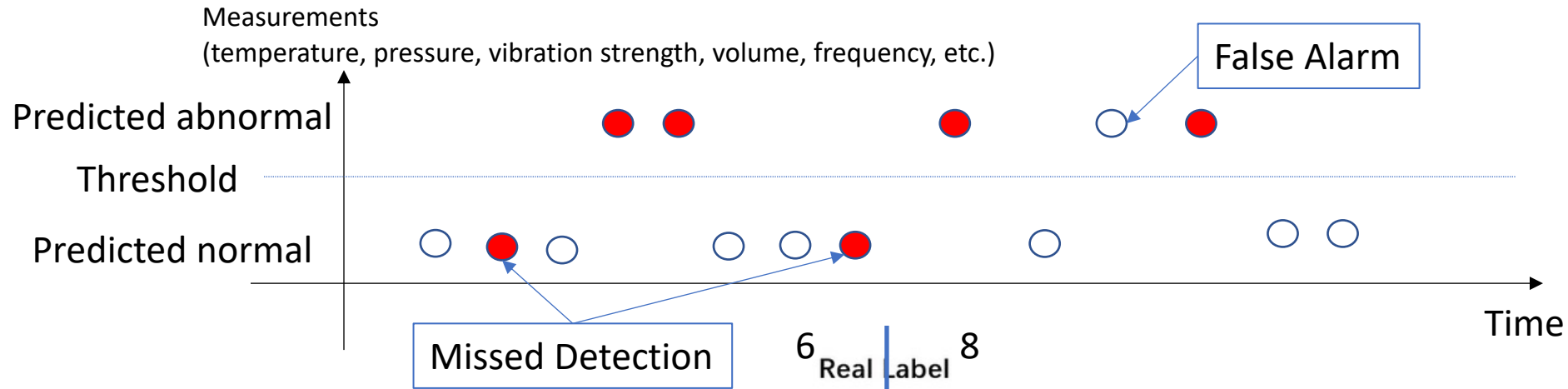
Default Category: unknown: 20

```

-----
Correctly Classified Instances          :      6398    84.9442%
Incorrectly Classified Instances       :      1134    15.0558%
Total Classified Instances             :      7532
=====

```


Performance Metrics for Binary Classification (Detection)



Missed Detection

False Alarm

		Real Label	
		Positive	Negative
Predicted Label	Positive	4 True Positive (TP)	1 False Positive (FP)
	Negative	2 False Negative (FN)	7 True Negative (TN)

Precision = $\frac{\sum TP}{\sum TP + FP}$ (Horizontal)

(Vertical) Recall = $\frac{\sum TP}{\sum TP + FN}$

Accuracy = $\frac{\sum TP + TN}{\sum TP + FP + FN + TN}$

$$F1 \text{ Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

H_1 (Alternative Hypothesis): Real Positive (Abnormal)

Missed Detection Rate (MDR) = $\frac{FN}{TP+FN}$



H_0 (Null Hypothesis): Real Negative (Normal)

False Alarm Rate (FAR) = $\frac{FP}{FP+TN}$

Note: MDR = 1 - Recall

MDR vs. FAR



- MDR and FAR are independent of each other as they are under two different hypotheses!


P
Real
N


100
100

Predicted	P 101	99	2
	N 99	1	98



MDR=1/100, FAR=2/100


P
Real
N


100
100

Predicted	P 197	1	98
	N 3	99	2



MDR=99/100, FAR=98/100


P
Real
N


100
100

Predicted	P 197	99	98
	N 3	1	2

MDR=1/100, FAR=98/100


P
Real
N


100
100

Predicted	P 3	1	2
	N 197	99	98

MDR=99/100, FAR=2/100

Precision+Recall vs. FAR+MDR

- Note that $\text{Recall} = 1 - \text{MDR}$

		P	Real	N
		100		3
Predicted	P 101	99		2
	N 2	1		1

High Precision: $2/101$ 👍
 High Recall: $99/100$ 👍

But high FAR: $2/3$ 👎

However, this scenario might be rare because $\#Real\ N \gg \#Real\ P$ in most cases.

		P	Real	N
		10		10,000
Predicted	P 19	9		10
	N 9991	1		9,990

Low MDR: $1/10$ 👍
 Low FAR: $10/10,000$ 👍

But low Precision: $9/19$ 👎

As we continuously make detection decisions, $\#Real\ N$ could become a really large number, which may eventually lead to a low Precision.

It's still acceptable as long as FAR is low!

Recognizing the difference in cost of classification errors

- False Alarm:
 - The cost of a false alarm may be much less than the cost of a false negative, e.g., Cancer, Missile
- Missed detection:
 - The cost of a false negative may be less than the cost of a false alarm, e.g., Spam

Confusion Matrix for Multi-Class Classification

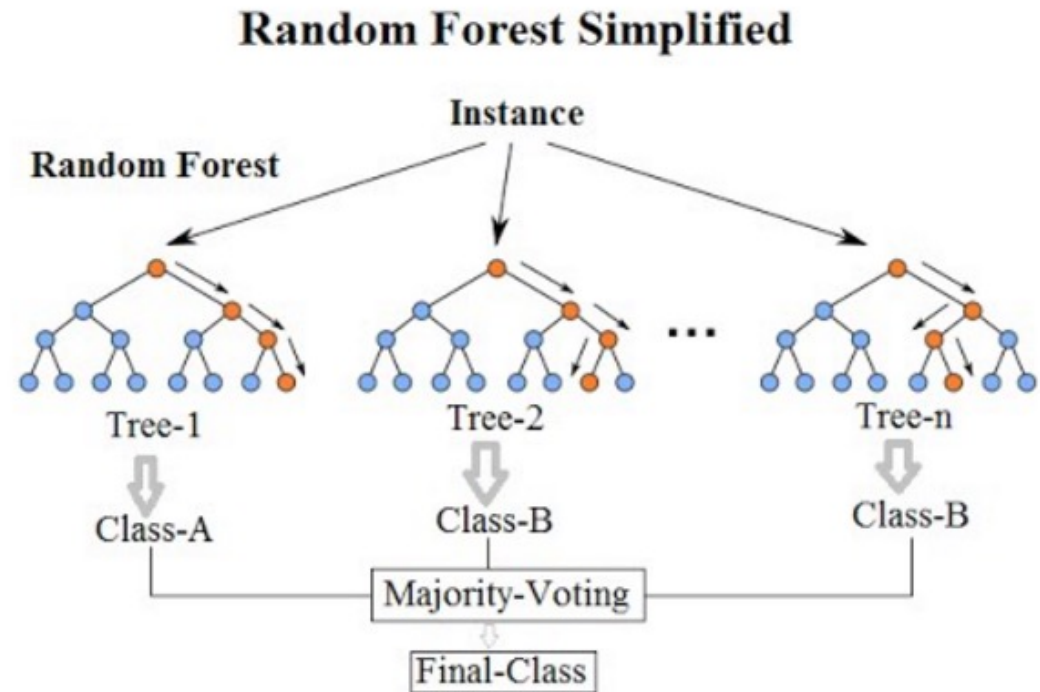
- For each class, calculate Precision (horizontal) and Recall (vertical)

		Ground truth				Precision
		Positive	Confusion	Negative	Neutral	
Prediction	Happiness	284 (85,29%)	46 (14,98%)	0	0	0,86
	Confusion	19 (5,71%)	257 (83,71%)	11 (35,48%)	43 (12,32%)	0,78
	Negative	0	4 (1,3%)	20 (64,52%)	6 (1,72%)	0,67
	Neutral	30 (9,01%)	0	0	300 (85,96%)	0,91
Recall		0,85	0,84	0,65	0,86	Accuracy:
F1-score		0,86	0,81	0,66	0,88	87,07%

Decision Tree and Random Forest

- Basic algorithm -- a greedy approach
 - Tree is constructed in a *top-down* recursive divide-and-conquer manner
 - At start, all the training examples are at the root
 - Attributes are categorical (if continuous-valued, discretize in advance)
 - Examples are partitioned recursively based on selected attributes
 - Test attributes are selected based on a heuristic or statistical measure (e.g., information gain, or Entropy)
- Conditions for stopping partitioning
 - All samples for a given node belong to the same class
 - There are no remaining attributes for future partitioning – majority voting is employed for classifying the leaf
 - There are no samples left

Random forest is a classification algorithm consisting of many decisions trees.



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 advertising expenditure
 - Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
 - Time series prediction of stock market indices

Regression -- example

- Non Exact Math → Approximation → Uncertainty Modeling
- Given a data set, input/output pairs with no function, i.e., no model, then how to find the *hypothesis!* (model)?
- Why do we need to do that?
- Example: suppose X is size of the house in acres, and Y is the price of houses already sold in hundred thousands, and you have a house that has 2.5 acres size, and you want to know how much you can get for it if you decide to sell it

$f?$

x	y
1	6
2	5
3	7
4	10

The task:

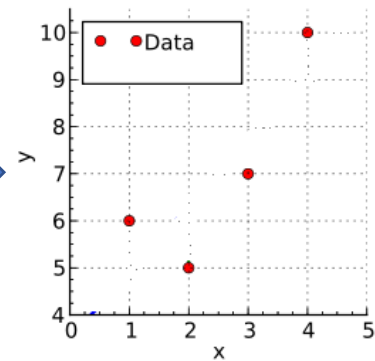
- Plot the data
- Use your eyes to suggest a *good* hypothesis
- Find the mathematical representation for the hypothesis
- Use it to predict the price for your house

Regression -- example

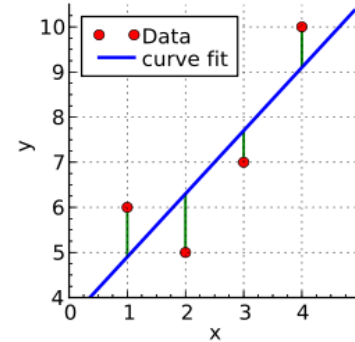
$f?$

x	y
1	6
2	5
3	7
4	10

Plot



Approximate with a model



Model is a Line?

$$y = \beta_1 + \beta_2 x$$

$$\beta_1 + 1\beta_2 = 6$$

$$\beta_1 + 2\beta_2 = 5$$

$$\beta_1 + 3\beta_2 = 7$$

$$\beta_1 + 4\beta_2 = 10$$

$$\begin{aligned} \min S(\beta_1, \beta_2) &= [6 - (\beta_1 + 1\beta_2)]^2 + [5 - (\beta_1 + 2\beta_2)]^2 \\ &\quad + [7 - (\beta_1 + 3\beta_2)]^2 + [10 - (\beta_1 + 4\beta_2)]^2 \\ &= 4\beta_1^2 + 30\beta_2^2 + 20\beta_1\beta_2 - 56\beta_1 - 154\beta_2 + 210 \end{aligned}$$

$$\begin{cases} \frac{\partial S}{\partial \beta_1} = 0 = 8\beta_1 + 20\beta_2 - 56 \\ \frac{\partial S}{\partial \beta_2} = 0 = 20\beta_1 + 60\beta_2 - 154 \end{cases}$$

$$\begin{cases} \beta_1 = 3.5 \\ \beta_2 = 1.4 \end{cases}$$

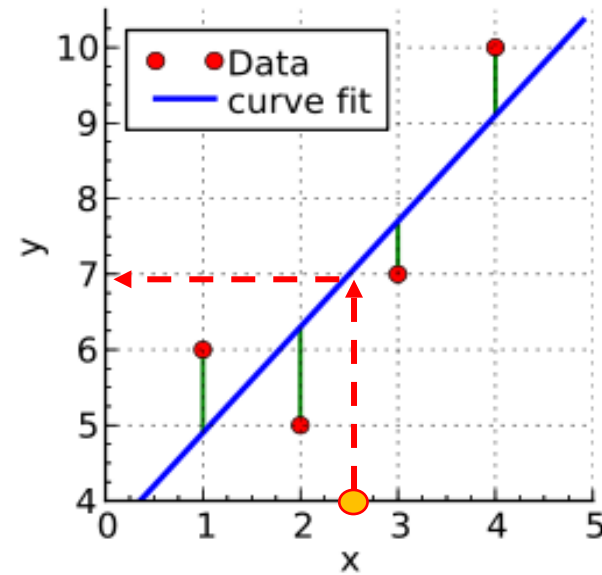
$$y = 3.5 + 1.4x$$

$$y = 3.5 + 1.4 \cdot (2.5) = 7$$

You will get 7 hundred thousand dollars if you decide to sell it now

Regression - Example

Can't you get more?



This is an example of using Polynomial Linear Regression for prediction, could this work with big data?

Regression -- example

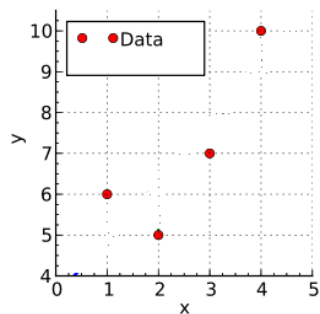
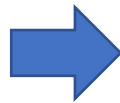
Modeling the 2nd Degree Polynomial Regression

- Let's try a quadratic model

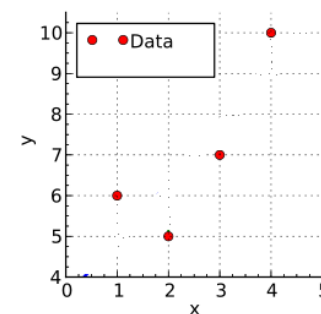
$f?$

x	y
1	6
2	5
3	7
4	10

Plot



with a
model



Quad.
Model

$$h = \beta_1 x^2$$



$$6 = \beta_1 (1)^2$$

$$5 = \beta_1 (2)^2$$

$$7 = \beta_1 (3)^2$$

$$10 = \beta_1 (4)^2$$



$$\frac{\partial S}{\partial \beta_1} = 0 = 708\beta_1 - 498$$



$$\beta_1 = .703$$



$$h = .703 (2.5)^2 = 4.39$$

You will get 4.39 hundred thousand dollars

Regression -- example

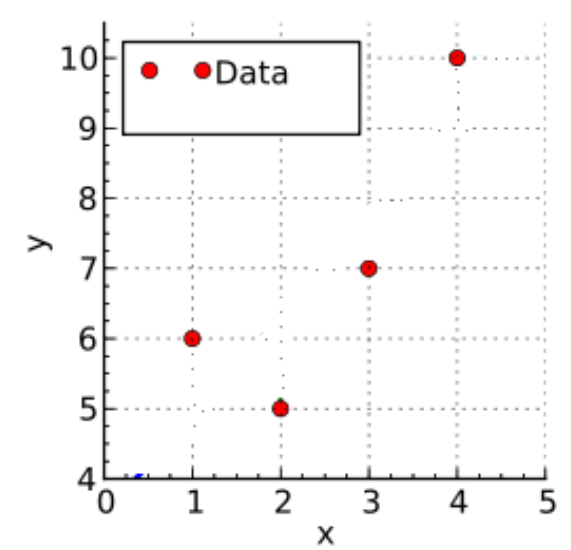
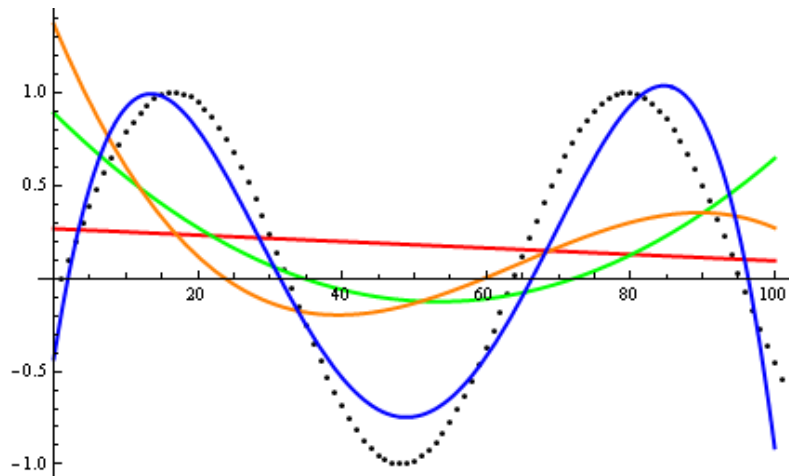
- Let's try higher degree polynomial models

$$h = \beta_0 + \beta_1x + \beta_2x^2$$

$$h = \beta_0 + \beta_1x + \beta_2x^2 + \beta_3x^3$$

⋮

$$h = \beta_0 + \beta_1x + \beta_2x^2 + \beta_3x^3 + \dots + \beta_nx^n$$



*The problem has become:
Finding the best solution (fit) and
optimize it as much as you can*



Thanks ! 😊

Questions ?