GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 19: REVIEW AND NEXT STEPS

Paul Burkard

01/13/2016

LAST TIME:

I. BIG DATA

II. HADOOP

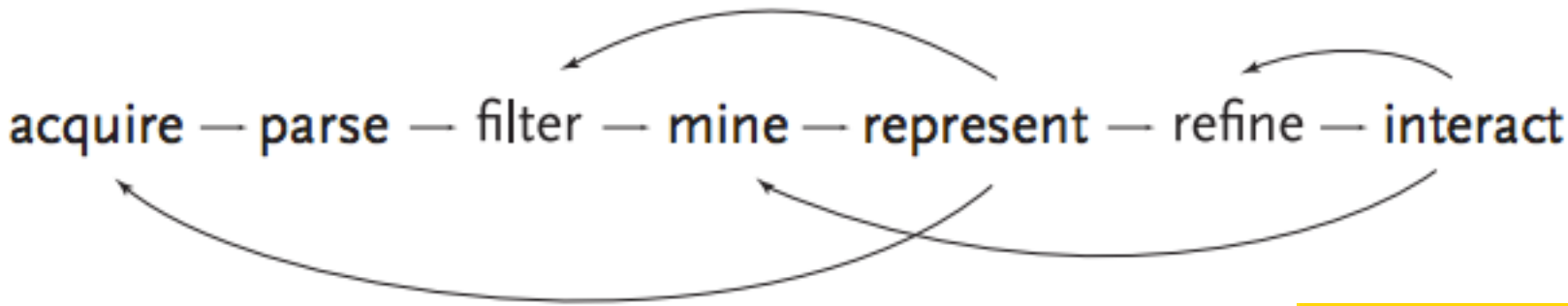
III. SPARK

IV. AWS

I. WHERE HAVE WE BEEN?
II. WHERE SHALL WE GO?

I. WHERE HAVE WE BEEN?

- A set of tools and techniques used to extract useful information from data.
- An interdisciplinary, problem-solving oriented subject.
- The application of scientific techniques to practical problems.
- A rapidly growing field.



NOTE

This diagram illustrates the *iterative* nature of problem solving

source: http://benfry.com/phd/dissertation-110323c.pdf

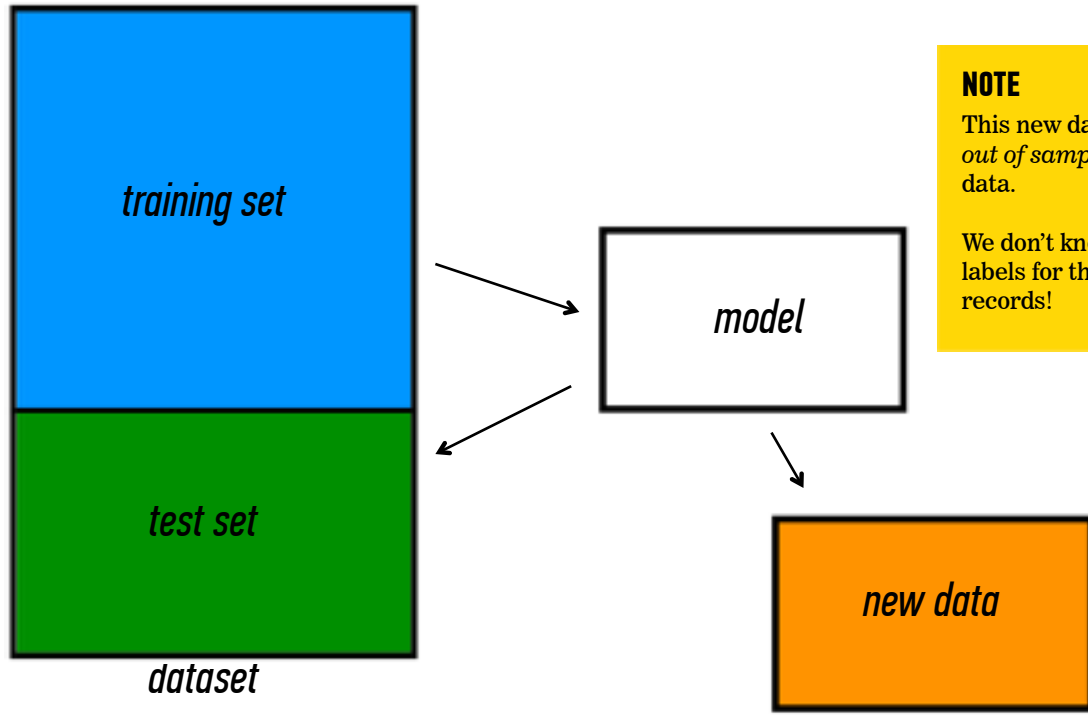
	continuous	*categorical*
supervised	*regression*	*classification*
unsupervised	*dim reduction*	*clustering*

supervised	*making predictions*
unsupervised	*discovering patterns*

supervised	*labeled examples*
unsupervised	*no labeled examples*

Q: What steps does a supervised learning problem require?

1) split dataset
2) train model
3) test model
4) make predictions



NOTE

This new data is called *out of sample* data.

We don't know the labels for these OOS records!

Q: What can go wrong if we don't follow these steps?

A: **Overfitting!**

- *If we test our model against the training set it might perform quite well on the training set, but fail to* **generalize** *to new data*
- *The model might be overly* **complex** *and tailored to the training data*

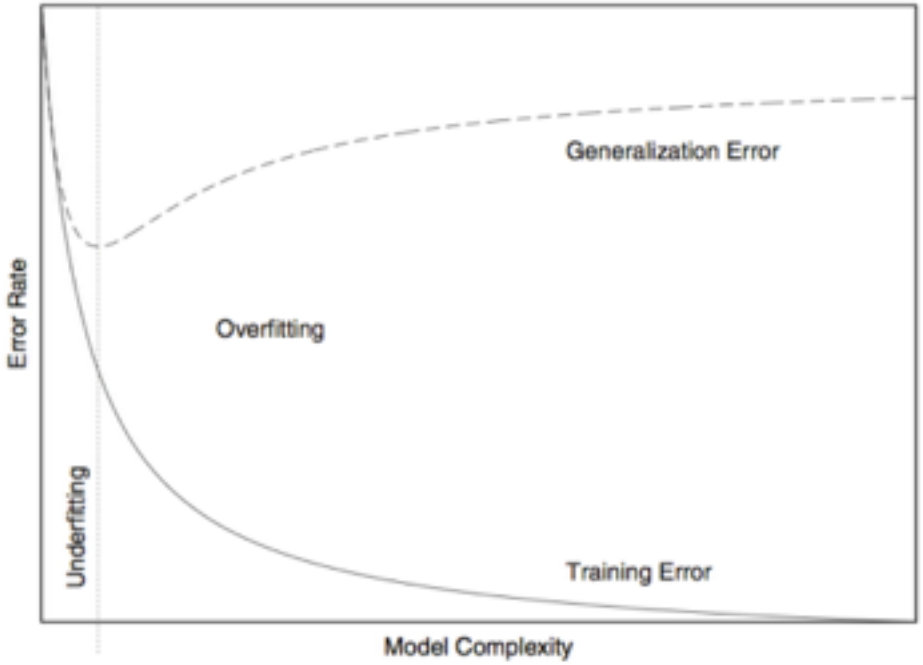
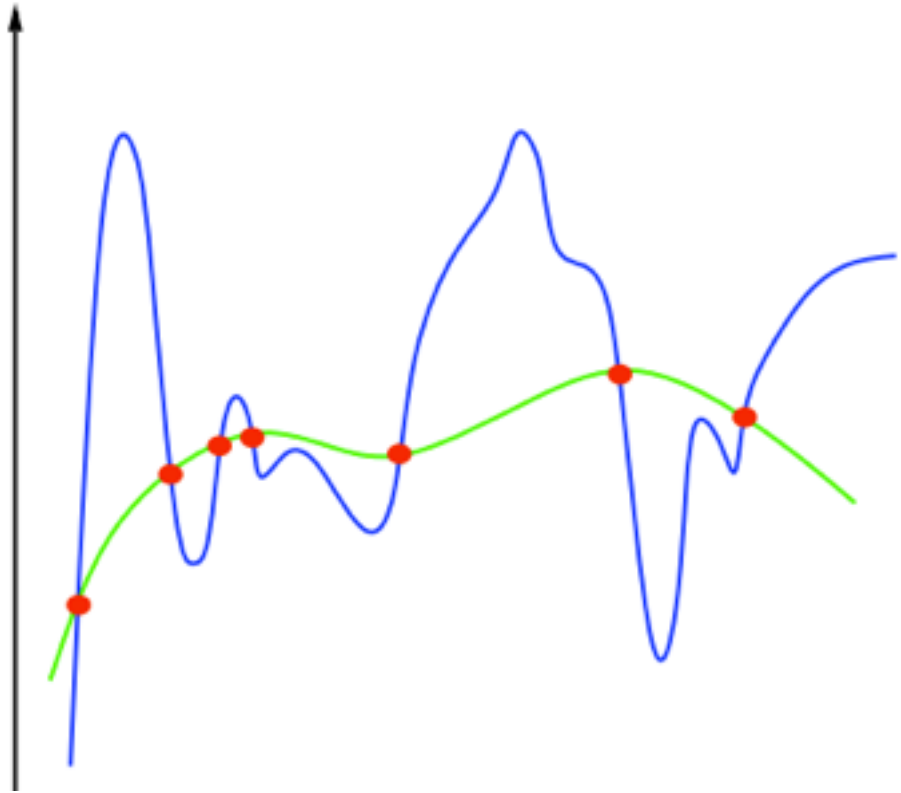


FIGURE 18-1. Overfitting: as a model becomes more complex, it becomes increasingly able to represent the training data. However, such a model is overfitted and will not generalize well to data that was not used during training.

source: Data Analysis with Open Source Tools, by Philipp K. Janert. O'Reilly Media, 2011.

OVERFITTING EXAMPLE (REGRESSION)



source: http://www.mit.edu/~9.520/spring12/slides/class02/class02.pdf

Q: How can we avoid overfitting?

A: One way is **Cross-Validation**

- *Pre-splitting the dataset into train/test sets is one form of cross-validation*
- *There are plenty of others (***k-fold***,* **leave-one-out***, etc)*

Steps of **k-fold Cross-Validation***:*

- *Partition dataset into k random, equal-sized subsets*
- *For each subset, hold it out as the test set and train on the rest*
- *Report the average of the testing performances as the model's estimated generalization performance*

Databases are a **structured** data source optimized for efficient **retrieval and persistent storage**

structured : we will have to define some pre-defined organization strategy

retrieval : the ability to read data our

storage: the ability to write data and save it

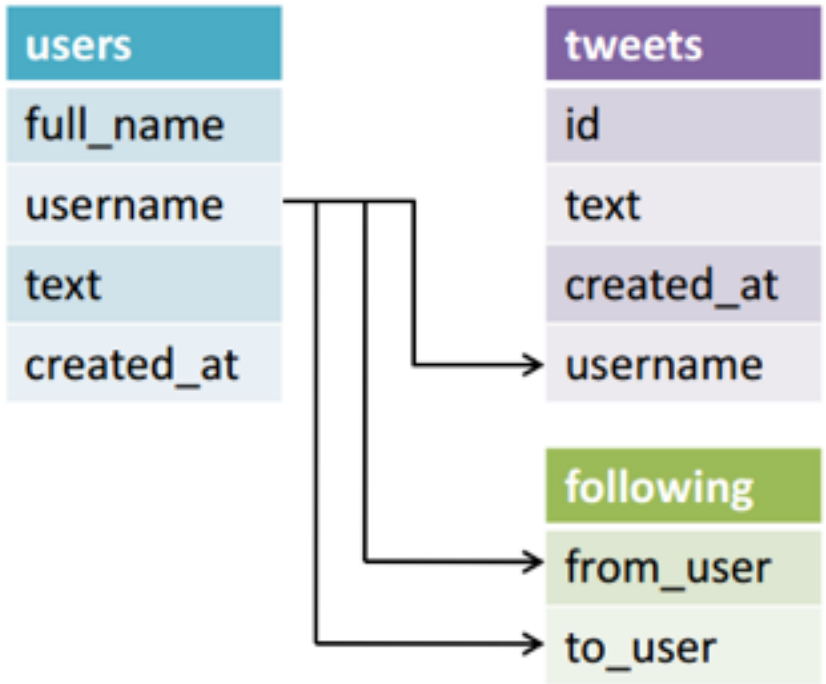
Relational databases are traditionally organized in the following manner:

A database has **tables** *which represent individual entities or objects* — "**Relations**"

Tables have a predefined **schema** - *rules that tell it what columns exist and what they look like*

Each table should have a **primary key** *column- a unique identifier for that row*

Additionally each table can have a **foreign key** *column- an id that references a unique entry in another table*


users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user

We could have had a table structure as follow:
Why is this different?

We would repeat the user information on each row.

This is called
denormalization

tweets
id
text
created_at
username
full_name
username
text
created_at

Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data:
Wide data, fields are often repeated but removes the need to join together multiple tables

Trade off of speed vs. storage

SELECT: *Allows you to* **retrieve** *information from a table*

Syntax:

SELECT col1, col2 FROM table WHERE <some condition>

Example:
SELECT poll_title, poll_date FROM polls WHERE romney_pct > obama_pct

GROUP BY: *Allows you to* **aggregate** *information from a table*

Syntax:

SELECT col1, AVG(col2) FROM table GROUP BY col1

There are usually a few common built-in operations:
SUM, AVG, MIN, MAX, COUNT

JOIN: *Allows you to* **combine** *multiple tables*

Syntax:

```
SELECT table1.col1, table1.col2, table2.col2
FROM  (JOIN table1, table2 ON table1.col1 = table2.col2)
```

Tutorial: http://www.w3schools.com/sql/default.asp

Other Commands: DISTINCT, ORDER BY, AND/OR, UPDATE, DELETE, LIKE, IN, HAVING, CREATE, DROP, ALTER…

NO-SQL databases are a new old trend in databases

The title **NOSQL** *refers to the lack of a relational structure between stored objects*

Most importantly, they often attempt to minimize the need for **JOIN** *operations*

Memcached	::	*LiveJournal*
Apache HBase	::	*Google BigTable*
Cassandra	::	*Amazon Dynamo*
MongoDB		

Key Takeaways:

- *Each Database has it's strengths*
- *Choose the right one for* **your use case**

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

high-level: *interpreted (not compiled)*

dynamic: *things that would typically happen at compile time happen at runtime instead (eg, dynamic typing)*

Q: What is Pandas?

A: The "Python Data Analysis Library"

"pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language."

open source: *free! (both binaries and source files)*

data structures: *allows for working with table-like data directly in Python - functionality is similar to R*

data analysis: *interoperates smoothly with further machine learning and visualization packages expecting NumPy-style data structures*

Q: Why should we use Pandas?

A:

- *Python has traditionally been good at raw data preparation, "data munging", and manipulation*
- *A rich set of recent Python libraries provide various Machine Learning out of the box (scikitlearn, statsmodels, etc)*
  - *These packages expect NumPy arrays, which can be tough to work with*
  - *Because of this, people often switch over to R etc. for data analysis*
- *Pandas bridges the gap by providing easy-to-use Python data structures for datasets that play nicely with these machine learning packages*

Q: What is **Web Scraping***?*

A: Retrieving data from a website in a format suitable for analysis

- *Most involves parsing HTML, or occasionally XML*
- *Alternatively many websites offer public APIs (Application Program Interface) with open methods for common data retrieval operations*
- *Websites often contain rich data, but also mountains of extraneous content that we need to wade through to get the stuff that we want*

Web Scraping in Python

- *We will use BeautifulSoup*
- *Other options:*
  - *Scrapy, lxml, HTQL, Mechanize, Selenium*

Q: What is **HTML***?*

A: **HTML** *is a markup language for describing web documents*

- *HTML stands for Hyper Text Markup Language*
- *A markup language is a set of markup tags*
- *HTML documents are described by HTML tags*
- *Each HTML tag describes different document content*

Sample HTML snippet

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>

<table style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
</table>

</body>
</html>
```

Q: How is **HTML** *used?*

A:

- *Designers use it to create webpages*
- *Browsers interpret the HTML markup to display the webpages*
- *Different HTML tags can provide many different types of content*
  - *Headers, spacing, tables, audio, images, video, links, etc.*
- *Here is a sufficient HTML Tutorial*

Q: What is an **API***?*

A: When an application allows access to certain programmatic functions to interact with its system

- *API stands for Application Program Interface*
- *Web applications with APIs allow users to access them by hitting specific URLs with the appropriate HTTP Requests*
- *Results are returned in various prescribed data formats, commonly JSON*

Sample Yelp Web API call

```
def search(term, location):
    """Query the Search API by a search term and location.
    Args:
        term (str): The search term passed to the API.
        location (str): The search location passed to the API.
    Returns:
        dict: The JSON response from the request.
    """

    url_params = {
        'term': term.replace(' ', '+'),
        'location': location.replace(' ', '+'),
        'limit': SEARCH_LIMIT
    }
    return request(API_HOST, SEARCH_PATH, url_params=url_params)
```

Some Public Web APIs:

- *Yelp*
- *Facebook*
- *Twitter*
- *ESPN*
- *StubHub*
- *EchoNest*
- *Spotify*
- *Many more!*

Q: What is **JSON***?*

A: **JSON** *is a syntax for storing and exchanging data*

- *JSON stands for JavaScript Object Notation*
- *Many programming languages (including Python) contain easy functions for converting JSON into usable objects*
- *JSON is "self-describing" and easy to understand*
- *Doesn't require as strict schema structure as XML*

Sample JSON snippet

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Q: How is **JSON** *used with Web APIs?*

A:

- *Users make appropriate Web API calls*
- *Web Applications return results of queries in JSON*
- *JSON is converted into programming objects to be manipulated*
- *Here is a sufficient JSON Tutorial*

Q: What is a **regression** *model?*
A: A functional relationship between input & response variables

The **simple linear regression** *model captures a linear relationship between a single input variable x and a response variable y:*

$$y = \alpha + \beta x + \varepsilon$$

Q: What do the terms in this model mean?

$$y = \alpha + \beta_1 x_1 + \ldots + \beta_n x_n + \varepsilon$$

A: y = **response variable** *(the one we want to predict)*

x = **input variable** *(the one we use to train the model)*

α = **intercept** *(where the line crosses the y-axis)*

β = **regression coefficient**s *(the model "parameters")*

ε = **residual** *(the prediction error)*

	continuous	*categorical*
supervised	*regression*	*classification*
unsupervised	*dim reduction*	*clustering*

Q: What is regularization?

A: Any built-in method to **reduce complexity** *of a model in an effort to* **lower the risk of overfitting**

These measures of complexity lead to the following **regularization** *techniques:*

L1 regularization*:* $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, |\beta_i| < s$

L2 regularization*:* $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, \beta_i^2 < s$

Regularization *refers to the method of preventing* **overfitting** *by explicitly controlling model* **complexity***.*

These measures of complexity lead to the following **regularization** *techniques:*

Lasso **regularization***:* $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, |\beta_i| < s$

Ridge regularization*:* $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, \beta_i^2 < s$

Regularization *refers to the method of preventing* **overfitting** *by explicitly controlling model* **complexity***.*

Q: What is a **Classification** *model/problem?*
A: A functional relationship between input & response variables...

Where the target variables are **categorical***!*

$$y = f(X)$$

The function we seek in a classification problem maps feature vectors to **qualitative/categorical target classes**

Here's (part of) an example dataset:

independent variables

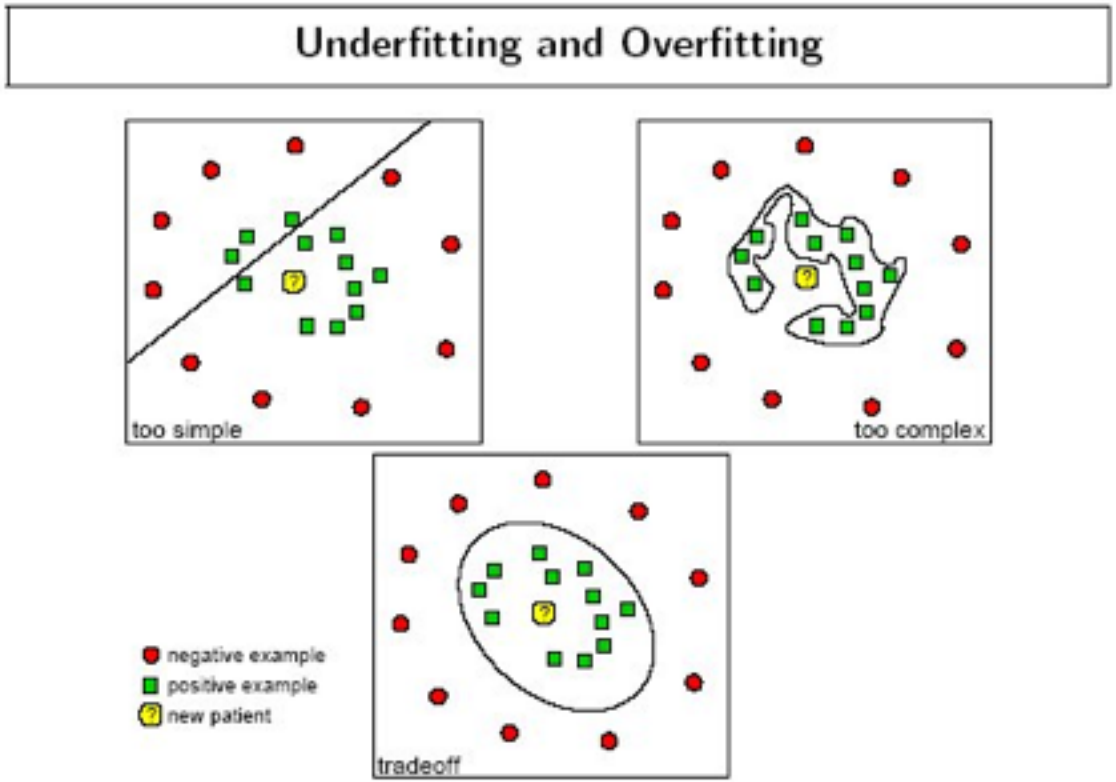
Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*

class labels (qualitative)

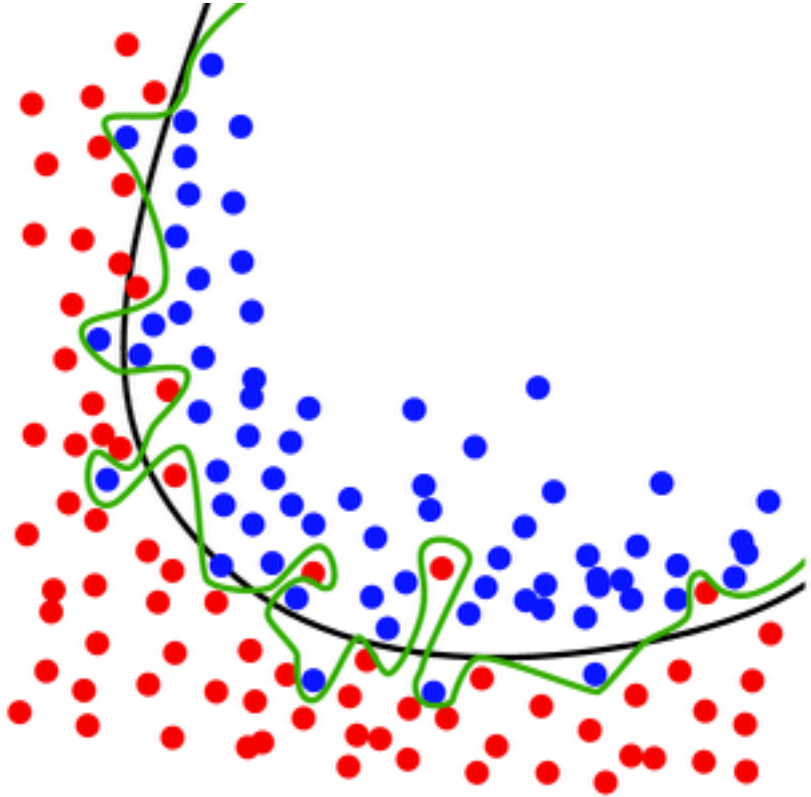
	continuous	*categorical*
supervised	*regression*	*classification*
unsupervised	*dim reduction*	*clustering*

CLASSIFICATION OVERFITTING – EXAMPLE



source: http://www.dtreg.com

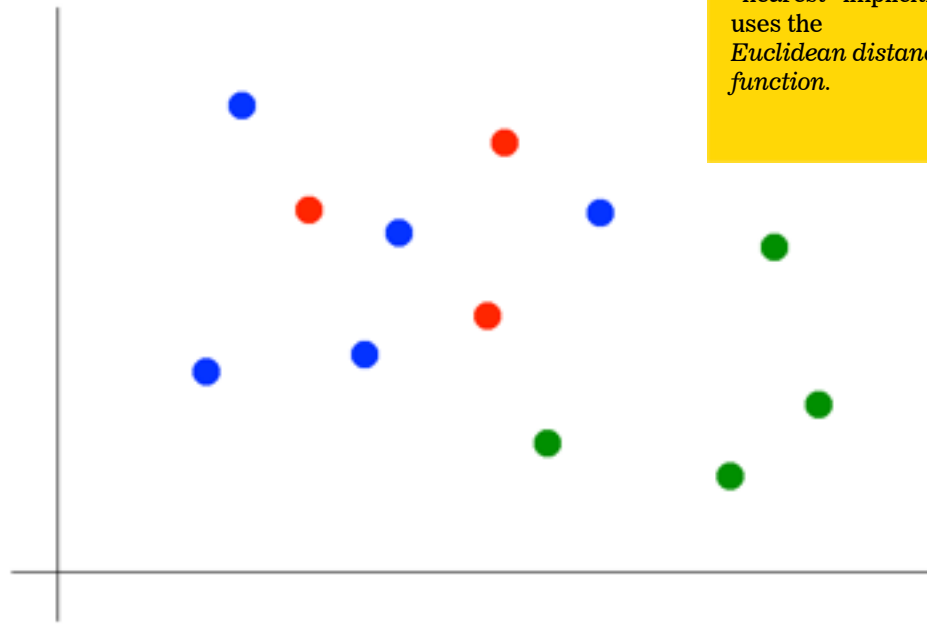
CLASSIFICATION OVERFITTING – EXAMPLE



source: http://www.dtreg.com

Suppose we want to predict the color of the grey dot.

1) *Pick a value for k.*
2) *Find colors of k nearest neighbors.*
3) *Assign the most common color to the grey dot.*



OPTIONAL NOTE

Our definition of "nearest" implicitly uses the *Euclidean distance function.*

Q: What is a **probability***?*

A: A number between 0 and 1 that characterizes the likelihood that some event will occur.

The probability of event A *is denoted* $P(A)$*.*

Q: What is the set of all possible events called?
A: This set is called the **sample space** Ω*. Event* A *is a member of the sample space, as is every other event.*

The probability of the sample space $P(\Omega)$ *is 1.*

Q: What is a probability distribution?
A: A function that assigns probability to each event in the sample space.

A distribution can be **discrete** *or* **continuous**
Ex: Discrete – Uniform distribution

$$X \sim \{1, \dots, N\} \quad \longrightarrow P(X = x) = 1/N$$

Continuous – Normal distribution – $N(\mu, \sigma^2)$

$$X \sim N(0, 1) \quad \longrightarrow \quad P(X = x) = 0$$

Discrete Probability Distributions:

- *These are* **probability mass functions**
- *Each value P(X=x) represents the probability of observing a given value x for variable X*

$$P(\Omega) = \sum_{X=x} P(X = x) = 1$$

Continuous Probability Distributions:

- *These are* **probability density functions (PDF)**
- *Each value P(X=x) represents the* **relative probability** *of observing a given value x for variable X*

$$P(x_0 < x < x_1) = \int_{x_0}^{x_1} P(x)\,dx$$

$$P(\Omega) = \int_{-\infty}^{+\infty} P(x)\,dx = 1$$

Q: What is a random variable?

A: Essentially, a measurable whose possible values have a probability distribution

Values of these are the "Events" for which we're looking to measure the probabilities

Q: What is expected value?

A: It is the average value of a random variable

For discrete distributions

$$E(X) = \sum x * p(x)$$

For continuous distributions

$$E(X) = \int (x * p(x))dx$$

Q: Consider two events A & B*. How can we characterize the* **intersection** *of these events?*

A: With the **joint probability** *of* A *and* B*, written* $P(A, B)$*.*

Q: Suppose event B has occurred. What quantity represents the probability of A **given** *this information about B?*

A: The intersection of A & B divided by region B.

This is called the **conditional probability** *of A given B, written $P(A|B) = P(A \cap B) / P(B)$.*

Q: What does it mean for two events to be **independent***?*
A: Information about one does not affect the probability of the other.

This can be written as $P(A|B) = P(A)$.

Using the definition of the conditional probability, we can also write:

$$P(A|B) = P(A \cap B) / P(B) = P(A) \rightarrow \quad P(A \cap B) = P(A) * P(B)$$

This result is called **Bayes' theorem**. *Here it is again:*

$$P(A|B) = P(B|A) * P(A) / P(B)$$

Some facts:

- This is a simple algebraic relationship using elementary definitions.

- It's a very powerful computational tool.

Each term in this relationship has a name, and each plays a distinct role in any probability calculation (including ours). Here's how it might look in the context of classification.

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

This term is the **likelihood function**. *It represents the joint probability of observing features* $\{x_i\}$ *given that that record belongs to class* C.

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

This term is the **likelihood function**. *It represents the joint probability of observing features* $\{x_i\}$ *given that that record belongs to class* C.

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

We can observe the value of the likelihood function from the training data.

This term is the **prior probability** *of C. It represents the probability of a record belonging to class C before the data is taken into account.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

This term is the **prior probability** *of C. It represents the probability of a record belonging to class C before the data is taken into account.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

The value of the prior is also observed from the data.

This term is the **normalization constant.** *It doesn't depend on* C*, and is generally ignored until the end of the computation.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

Maximum likelihood estimator (MLE):

What parameters **maximize** *the likelihood function?*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

Maximum a posteriori estimate (MAP):

What parameters **maximize** *the likelihood function* **AND** *prior?*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

Binomial Distribution:

$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

P (4 heads, 3 tails | q) = P (X = 4, n = 7)

= (7 choose 4) * q^4 * $(1-q)^3$

Optimize w.r.t. q —> **MLE:** q = 4/7

A prior distribution is known as **conjugate prior** if its from the same family as the posterior for a certain likelihood function

For the binomial distribution, the conjugate prior is the **Beta distribution**

$$= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

$$= \frac{1}{\mathrm{B}(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

Q: What assumption does Naive Bayes make?

A: We assume that the features x_i are conditionally independent from each other:

$$P(\{x_i\}|C) = P(x_1, x_2, \ldots, x_n|C) \approx P(x_1|C) * P(x_2|C) * \ldots * P(x_n|C)$$

This **"naïve" assumption** *simplifies the likelihood function to make it tractable.*

Q: What is **logistic regression***?*

A: A generalization of the linear regression model to classification problems.

In linear regression, we used a set of features to predict the value of a (continuous) outcome variable.

In logistic regression, we use a set of features to predict probabilities of (binary) class membership.

These probabilities are then mapped to class labels, thus solving the classification problem.

In logistic regression, we've seen that the conditional mean of the outcome variable takes values only in the unit interval $[0, 1]$.

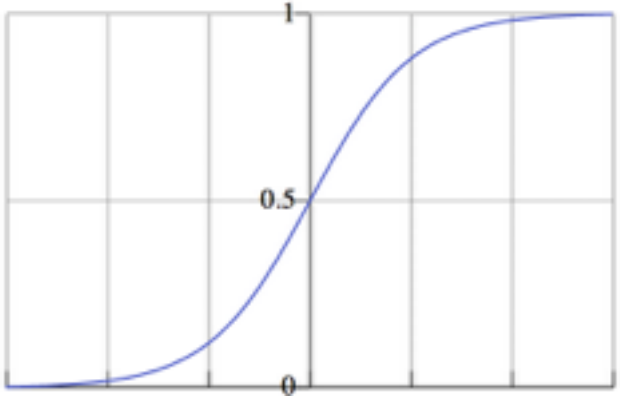
The first step in extending the linear regression model to logistic regression is to map the outcome variable $E(y \mid x)$ *into the unit interval.*

Q: How do we do this?

A: By using a transformation called the **logistic function***:*

$$E(y|x) = \pi(x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

We've already seen what this looks like:



NOTE

For any value of x, y is in the interval [0, 1]

This is a nonlinear transformation!

The **logit function** *is an important transformation of the logistic function. Notice that it returns the linear model!*

$$g(x) = ln(\frac{\pi(x)}{1-\pi(x)}) = \alpha + \beta x$$

NOTE

This name hints at its usefulness in interpreting our results.

We will see why shortly.

The logit function is also called the **log-odds function**.

We can now state the following:

$$e^{g(x)} = OR = e^{\alpha+\beta x}$$

So that if,

$$e^{\beta_i} = n$$

then the odds ratio is increased by a factor of n for a unit increase of x_i

Q: What is a decision tree?
A: A non-parametric hierarchical classification technique.

non-parametric*: no parameters, no distribution assumptions*

hierarchical*: consists of a sequence of questions which yield a class label when applied to any record*

EXAMPLE – DECISION TREE

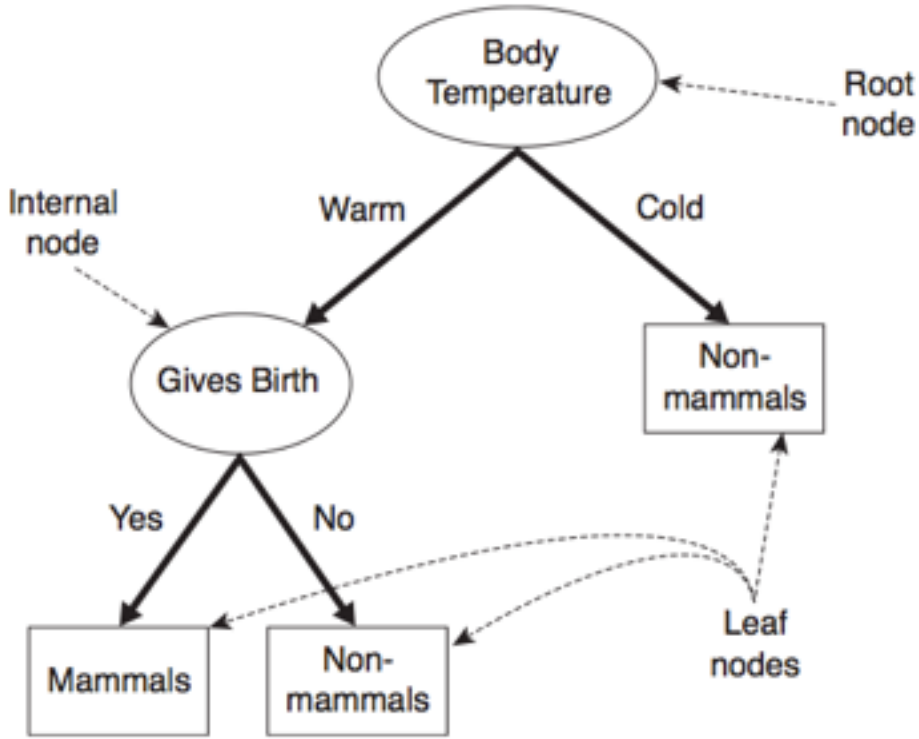


Figure 4.4. A decision tree for the mammal classification problem.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

NOTE

Internal nodes represent test conditions which *partition the records* at that node.

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm**.

This is a **greedy recursive** *algorithm that leads to a* **local optimum**.

greedy – *algorithm makes locally optimal decision at each step*
recursive – *splits task into subtasks, solves each the same way*
local optimum – *solution for a given neighborhood of points*

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called **purity***.*

A partition is 100% pure when all of its records belong to a single class.

Consider a binary classification problem with **classes X, Y**. *Given a* **set of records D_t** *at* **node t**, **Hunt's algorithm** *proceeds as follows:*

1) If all records in D_t belong to class X, then t is a leaf node corresponding to class X.

2) If D_t contains records from both classes, then a test condition is created to partition the records further. In this case, t is an internal node whose outgoing edges correspond to the possible outcomes of this test condition.

These outgoing edges terminate in **child nodes**. *A record d in D_t is assigned to one of these child nodes based on the outcome of the test condition applied to d.*

3) Recursively apply 1 & 2 to each child node

Q: How do we partition the training records?

A: There are a few ways to do this.

Test conditions can create **binary splits***:*

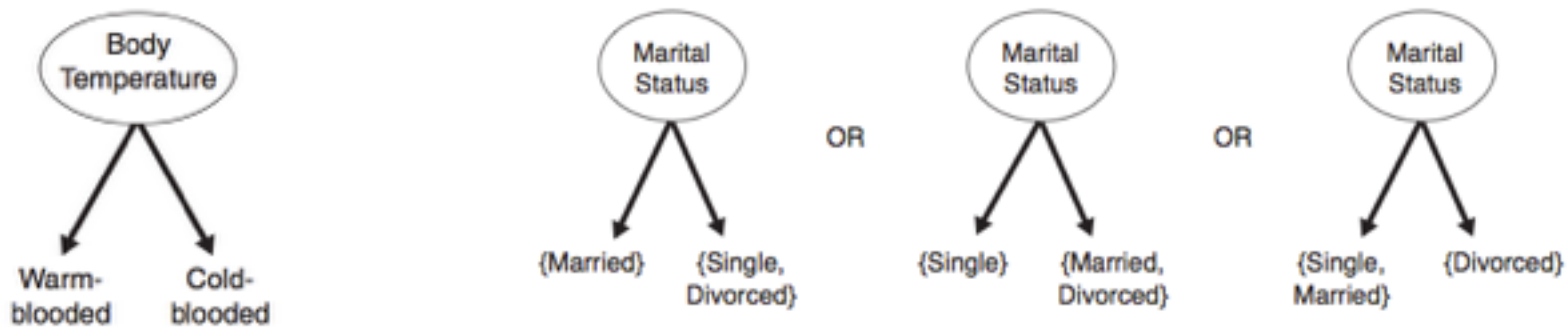


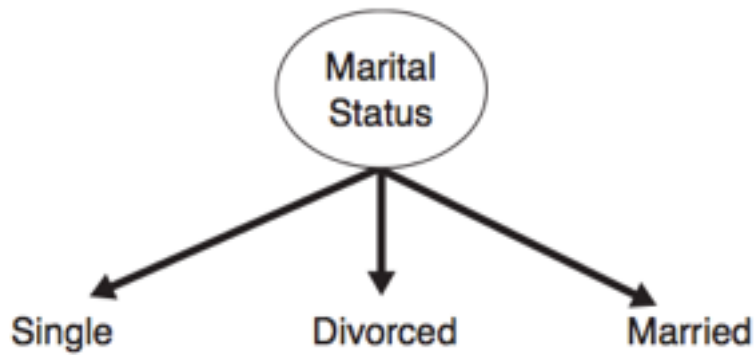
Figure 4.8. Test condition for binary attributes.

(b) Binary split {by grouping attribute values}

Q: How do we partition the training records?

A: There are a few ways to do this.

Alternatively, we can create **multiway splits***:*



(a) Multiway split

NOTE

Multiway splits can produce purer subsets, but may lead to overfitting!

Q: How do we partition the training records?

A: There are a few ways to do this.

For continuous features, we can use either method:

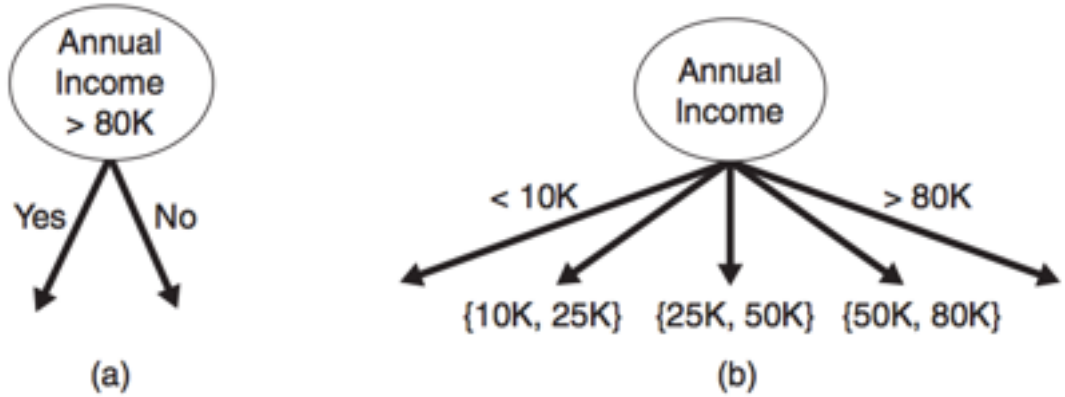


Figure 4.11. Test condition for continuous attributes.

NOTE

There are optimizations that can improve the naïve quadratic complexity of determining the optimum split point for continuous attributes.

Q: How do we determine the best split?
A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the highest possible purity (the most class-homogeneous splits).

We need an objective function for **purity** *to optimize!*

Some measures of impurity include:

$$\begin{aligned}
\text{Entropy}(t) &= -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t), \\
\text{Gini}(t) &= 1-\sum_{i=0}^{c-1}[p(i|t)]^2, \\
\text{Classification error}(t) &= 1-\max_i[p(i|t)],
\end{aligned}$$

Note that each measure achieves its max at 0.5, min at 0 & 1.

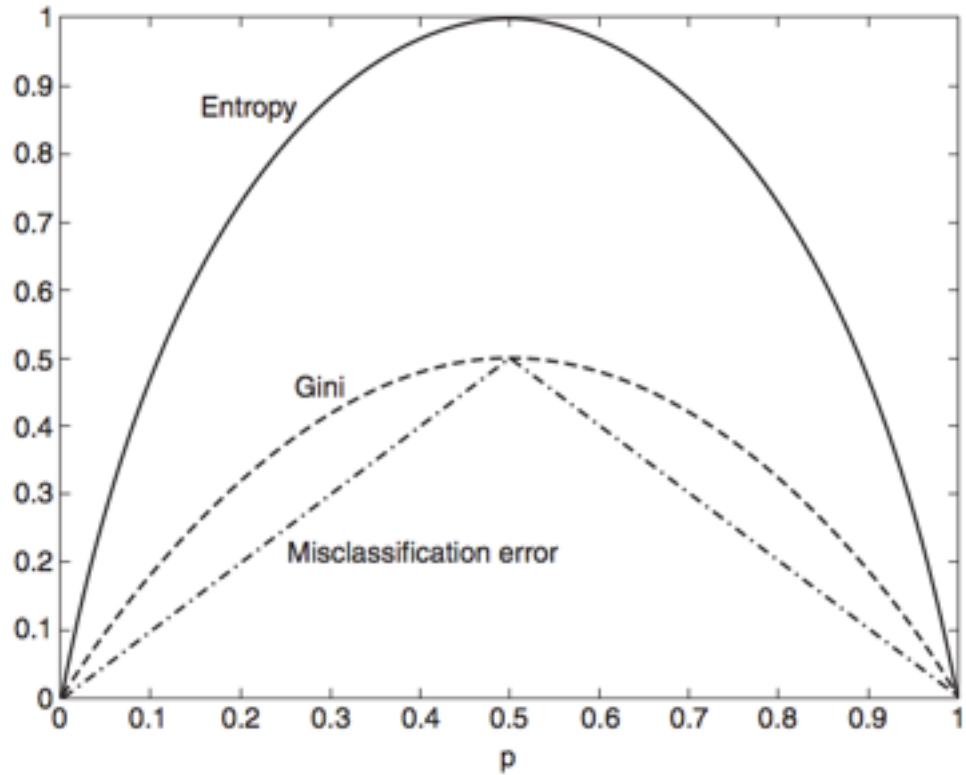


Figure 4.13. Comparison among the impurity measures for binary classification problems.

NOTE

Despite consistency, different measures may create different splits.

We can make this comparison using the **gain***:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I is the impurity measure, N_j denotes the number of records at child node j, and N denotes the number of records at the parent node.)

When I is the entropy, this quantity is called the **information gain***.*

We can use a function of the information gain called the **gain ratio** *to explicitly penalize high numbers of outcomes:*

$$\text{gain ratio} = \frac{\Delta_{info}}{-\sum p(v_i) log_2 p(v_i)}$$

NOTE

This is a form of regularization!

(Where $p(v_i)$ refers to the probability of label i at node v)

Q: *What is a support vector machine?*
A: *A* **binary linear classifier** *whose decision boundary is explicitly constructed to minimize generalization error.*

recall:

binary classifier – solves two-class problem
linear classifier – creates linear decision boundary (in 2d)

Q: *How is the decision boundary derived?*

A: *Using* **geometric reasoning** *(as opposed to the algebraic reasoning we've used to derive other classifiers).*

The goal of an SVM is to create the linear decision boundary with the **largest margin**. *This is commonly called the* **maximum margin hyperplane**.

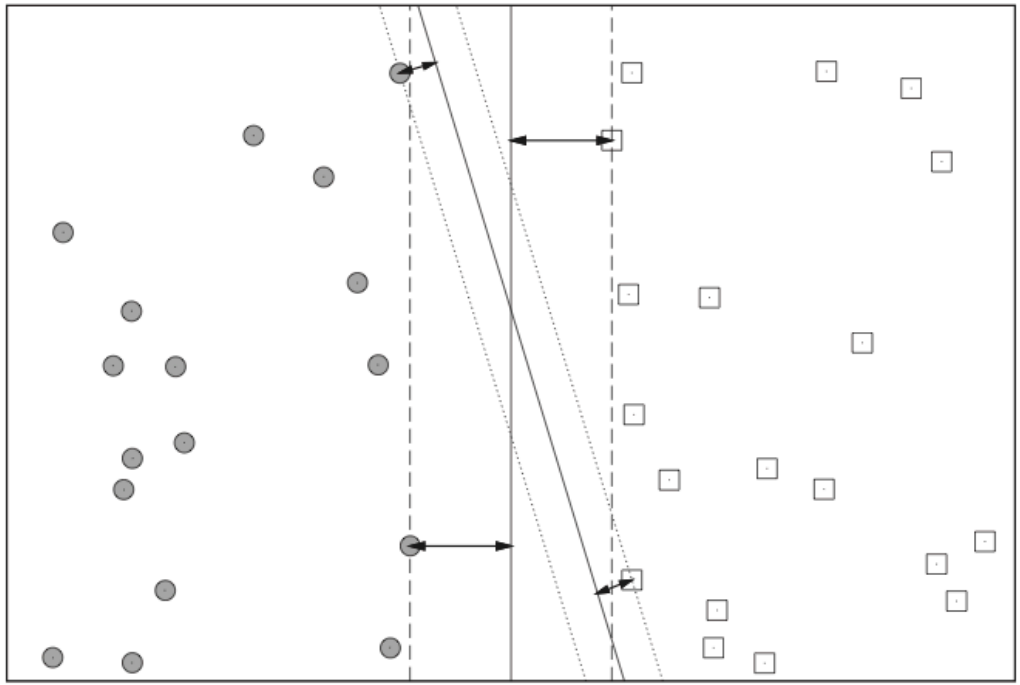


FIGURE 18-4. Two decision boundaries and their margins. Note that the vertical decision boundary has a wider margin than the other one. The arrows indicate the distance between the respective support vectors and the decision boundary.

Q: *How is the decision boundary (**mmh**) derived?*

A: *By the* **discriminant function**,

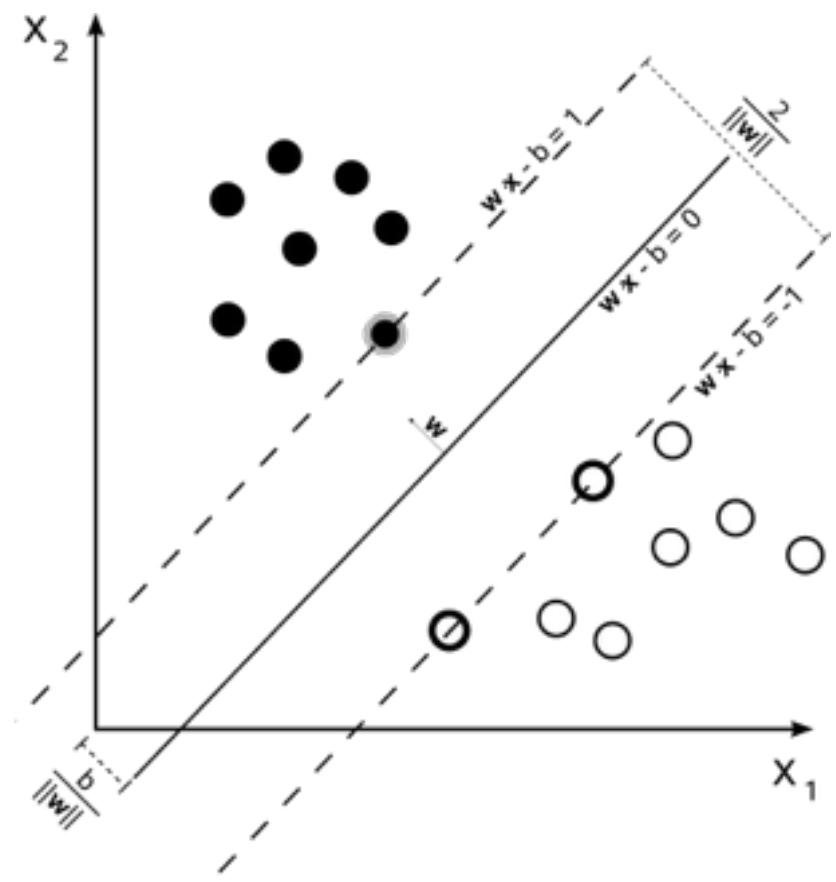
$$f(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b.$$

such that w is the weight vector and b is the bias.

The sign of f(x) determines the (binary) class label of a record x.

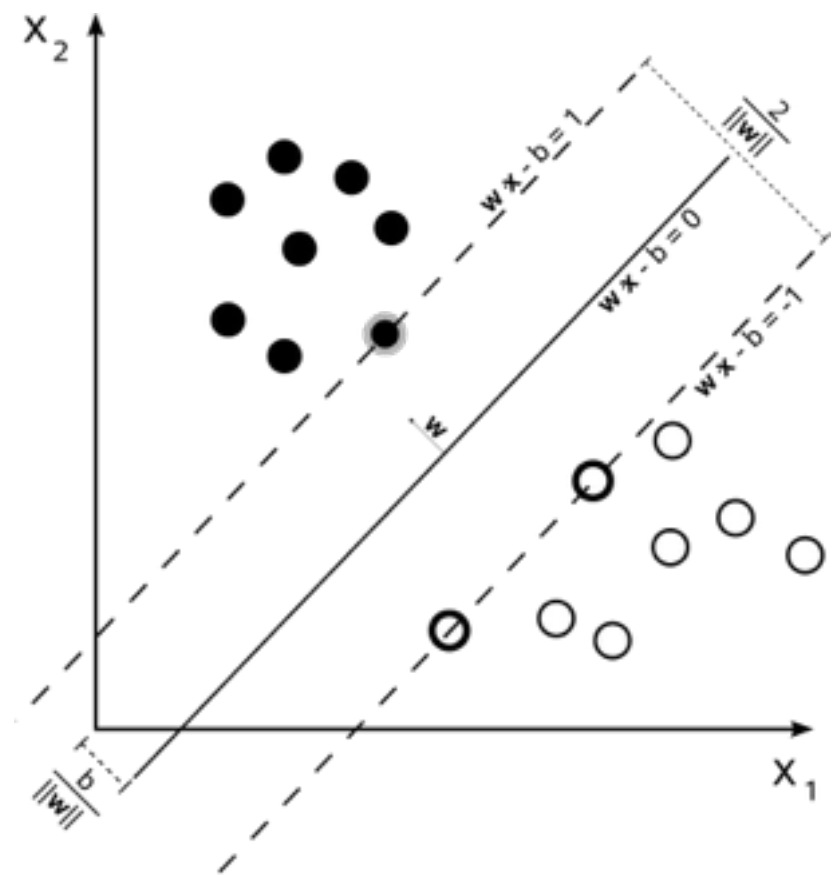
*How is the decision boundary (**mmh**) derived?*

- Any hyperplane can be written as the set of points where $\mathbf{w}^T\mathbf{x} - \mathbf{b} = 0$
  - **w** sets the plane's orientation (it's perpendicular to the plane)
  - **b** sets the offset from the origin
- Set the margin planes for each class such that $\mathbf{w}^T\mathbf{x} - \mathbf{b} = +/-1$
  - +1 for positive class, -1 for negative class



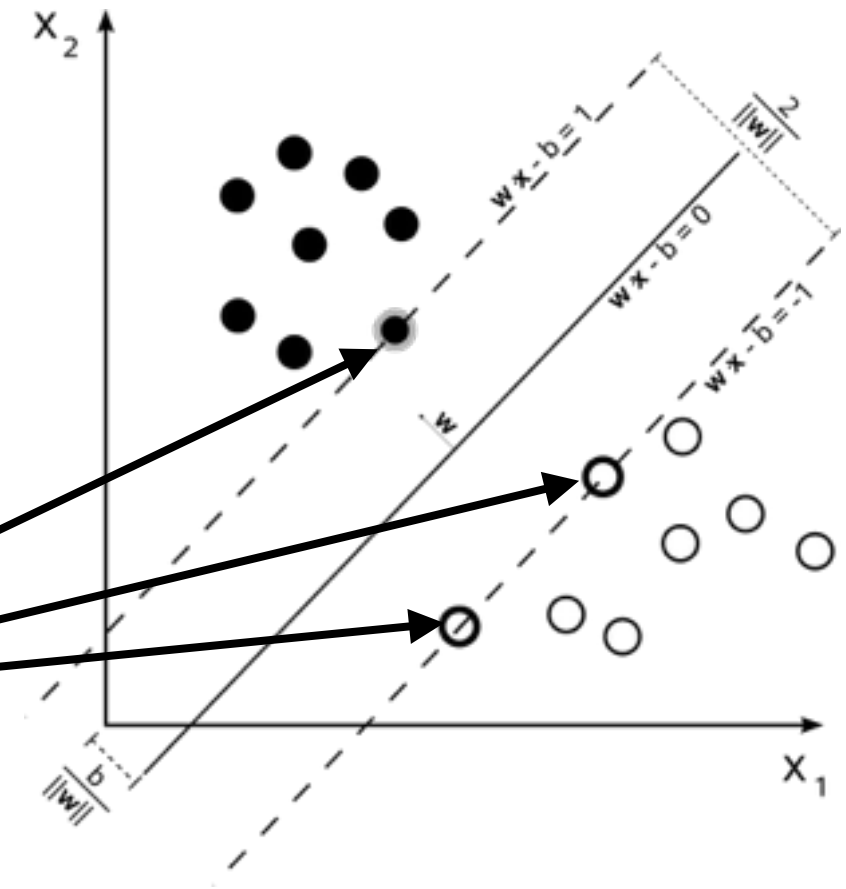
*How is the decision boundary (**mmh**) derived?*

- Thus, our goal of maximizing the margin width amounts to maximizing **2/||w||**
  - To do this, we must minimize **||w||**
  - Easier to minimize **$||w||^2/2$**
- We do this subject to the constraint: **$y_i*(w^T x_i - b) \geq 1$**
- This is a **Quadratic Programming (QP)** optimization problem, yielding **w** and **b**



*How is the decision boundary (**mmh**) derived?*

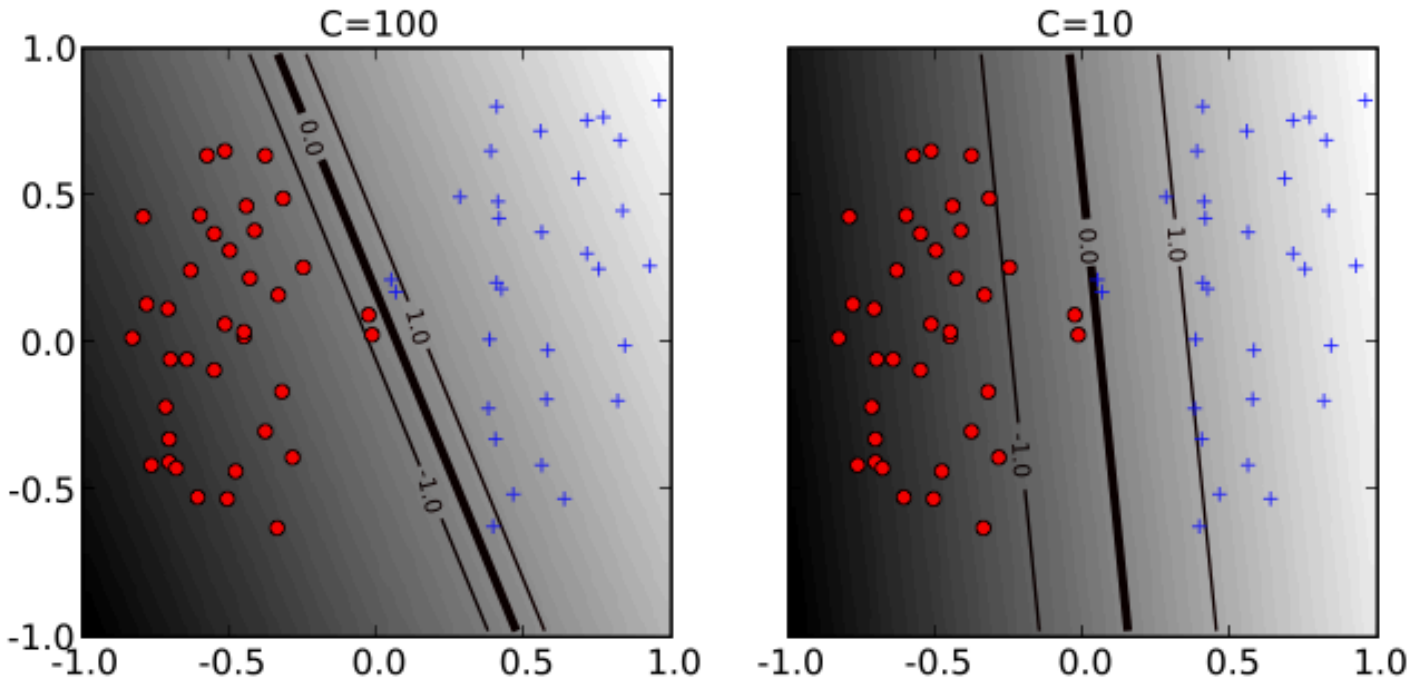
- This is a **Quadratic Programming (QP)** optimization problem, yielding **w** and **b**
- Solving this QP problem yields a solution of the form: $$w = \sum_i \alpha_i y_i x_i$$
- Most of these alpha are 0
- Only the "**support vectors**" will be nonzero
- Thus only they contribute!



Q: *How do we handle potentially inseparable data?*
A: *By training a* **soft margin SVM** *rather than a hard margin*

soft margin – basically allows for a fuzzy boundary, where some proportion of elements may be misclassified in order to maintain a simpler boundary

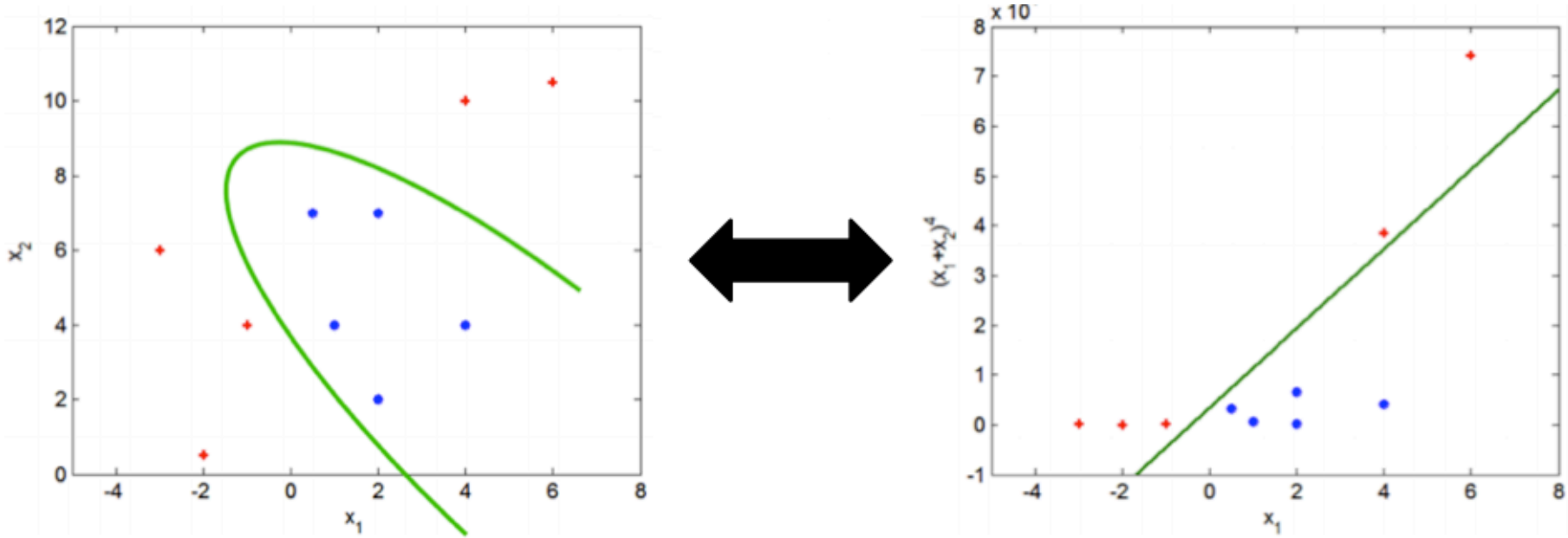
Remember, simpler boundaries are probably more likely to generalize.


C=100
C=10
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0
0.0
1.0
-1.0

Suppose we need a more complex classifier than a linear decision boundary allows.

One possibility is to add nonlinear combinations of features to the data, and then to create a linear decision boundary in the enhanced (higher-dimensional) feature space.

This **linear** decision boundary will be mapped to a **nonlinear** decision boundary in the original feature space.



original feature space *K*

higher-dim feature space *K'*

We can use a kernel function to *implicitly* train our model in a higher-dimensional feature space, *without* incurring additional computational complexity!

As long as the kernel function satisfies certain conditions, our conclusions above regarding the mmh continue to hold.

NOTE

These conditions are contained in a result called *Mercer's theorem*.

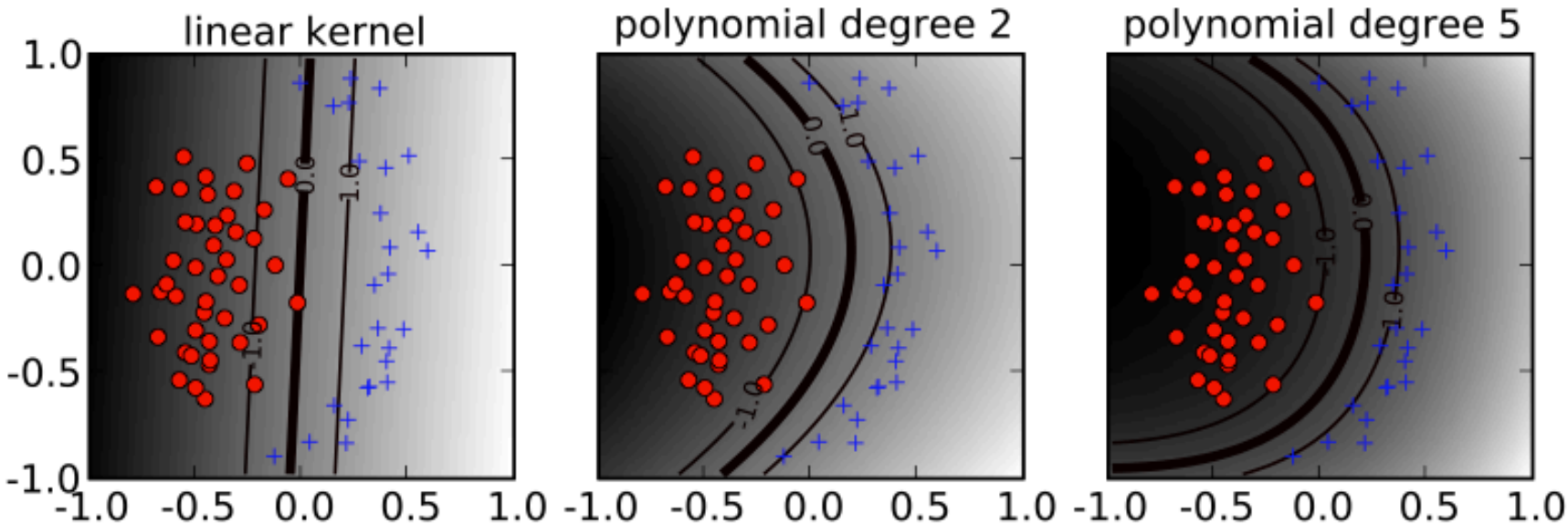
some popular kernels:

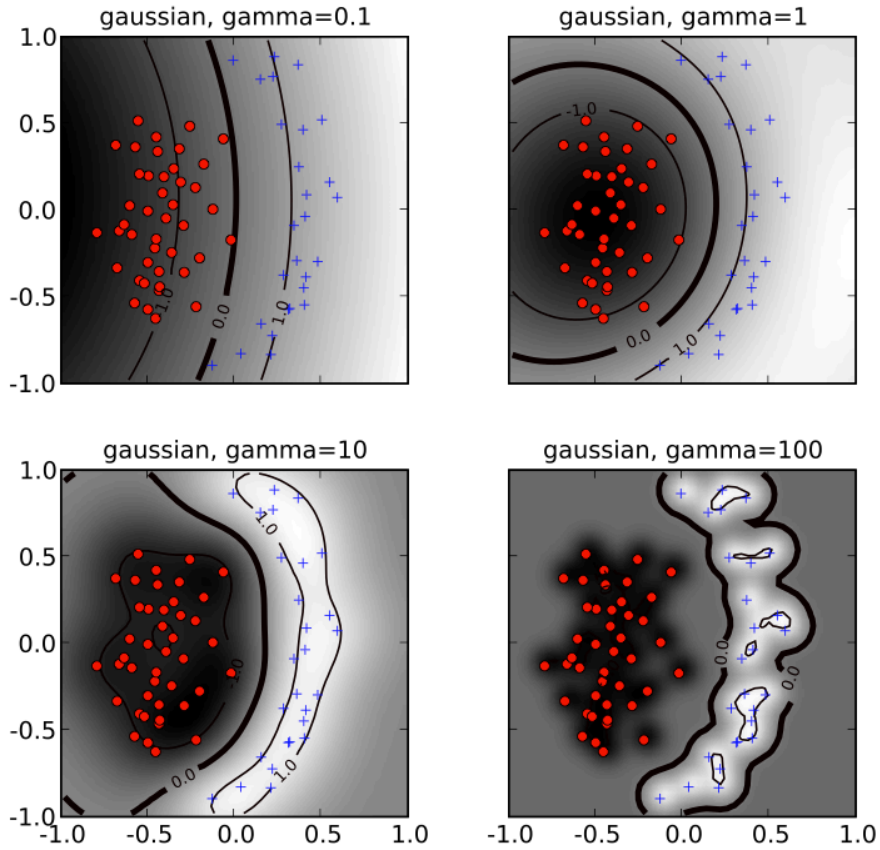
linear kernel $$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$$

polynomial kernel $$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d$$

Gaussian (RBF) kernel $$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$

The **hyperparameters d, γ** affect the flexibility


linear kernel
polynomial degree 2
polynomial degree 5
1.0
0.5
0.0
-0.5
-1.0
-1.0 -0.5 0.0 0.5 1.0
0.0
-1.0


gaussian, gamma=0.1
gaussian, gamma=1
gaussian, gamma=10
gaussian, gamma=100
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0

	continuous	*categorical*
supervised	*regression*	*classification*
unsupervised	*dim reduction*	*clustering*

Q: *So what is cluster analysis?*

A: **Unsupervised learning algorithms** *that seek to discover patterns in data by grouping unlabeled observations into coherent subsets.*

Clustering provides a layer of abstraction from individual data points.

The goal is to enhance the natural structure of the data (not to impose arbitrary structure!)

Q: *When should we use cluster analysis?*
A: *Clustering is often useful in the* **data exploration stage** *of the data analysis pipeline to get a better feel for your data.*

Does it have inherent groups of observations?

Do these groups have different behaviors for building further models?

Can I build better models by taking these groups into consideration?

Q: *What is K-Means Clustering?*

A: *Probably the most famous* **clustering algorithm**, *a* **greedy** *learner that* **partitions** *a data set into k clusters.*

greedy - only makes locally optimal decisions
partitions - each point belongs to one cluster (usually)

1) *choose k initial centroids (note that k is an input)*

2) *for each point:*
   - *find distance to each centroid*
   - *assign point to nearest centroid*

3) *recalculate centroid positions*
4) *repeat steps 2-3 until stopping criteria met*

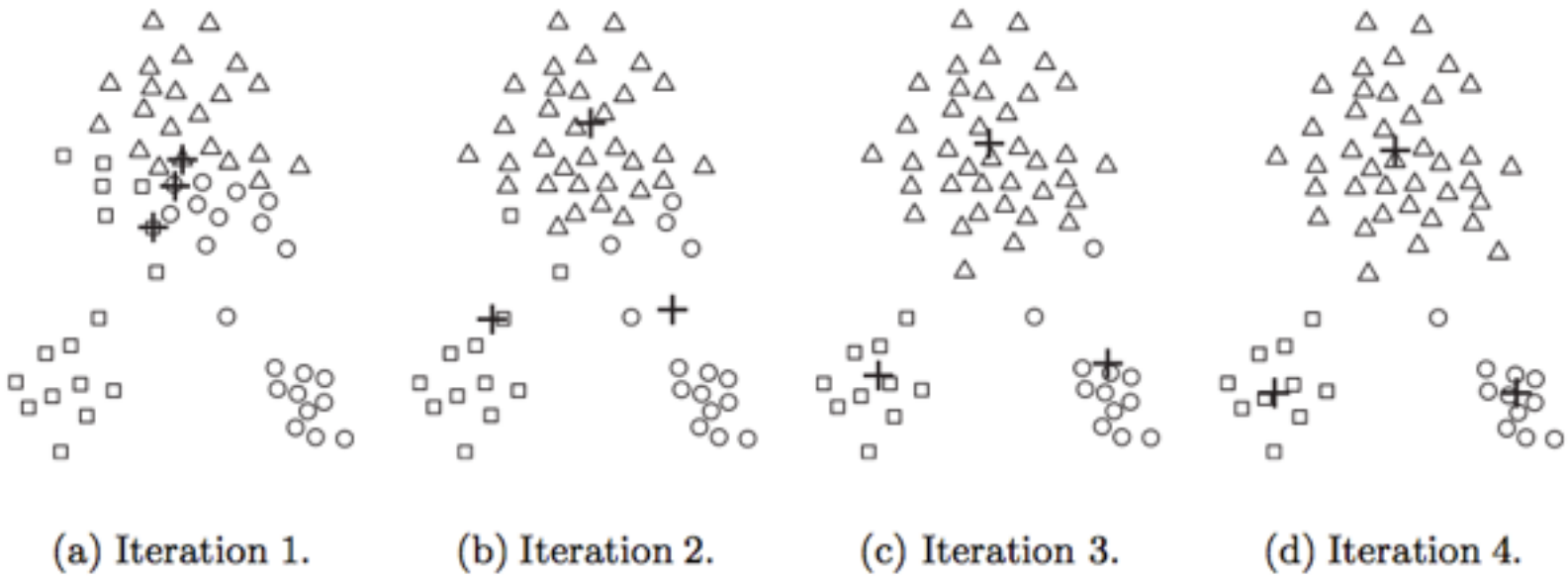


Figure 8.3. Using the K-means algorithm to find three clusters in sample data.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

There are as many as hundreds of different clustering algorithms.

They generally fall into a handful of classes:

- *Density-Based Clustering*
- *Hierarchical Clustering (Connective Models)*
- *Distribution-Based Clustering*
- *Graphical Models*

In general, k-means will converge to a solution and return a partition of k clusters, even if no natural clusters exist in the data.

How do we evaluate the usefulness or performance of our resulting clusters?

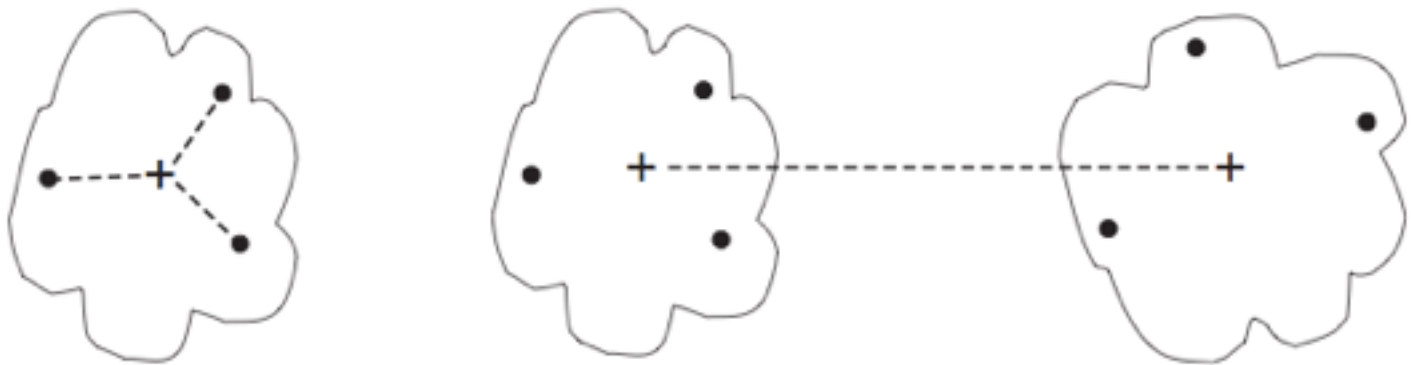
We will look at two validation metrics useful for partitional clustering: **cohesion** *and* **separation***.*

Cohesion *measures clustering effectiveness within a cluster.*

$$\hat{C}(C_i) = \sum_{x \in C_i} d(x, c_i)$$

Separation *measures clustering effectiveness between clusters.*

$$\hat{S}(C_i, C_j) = d(c_i, c_j)$$



(a) Cohesion. (b) Separation.

Figure 8.28. Prototype-based view of cluster cohesion and separation.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

One useful measure than combines the ideas of cohesion and separation is the **silhouette coefficient**. *For point x_i, this is given by:*

$$SC_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

such that:

a_i = *average in-cluster distance to* x_i

b_{ij} = *average between-cluster distance to* x_i

$b_i = min_j(b_{ij})$

The silhouette coefficient can take values between -1 and 1.

In general, we want separation to be high and cohesion to be low. This corresponds to a value of SC *close to +1.*

A negative silhouette coefficient means the cluster radius is larger than the space between clusters, and thus clusters overlap.

The silhouette coefficient for the cluster C_i is given by the average silhouette coefficient across all points in C_i:

$$SC(C_i) = \frac{1}{m_i} \sum_{x \in C_i} SC_i$$

The overall silhouette coefficient is given by the average silhouette coefficient across all points:

$$SC_{total} = \frac{1}{k} \sum_{1}^{k} SC(C_i)$$

NOTE

This gives a summary measure of the overall clustering quality.

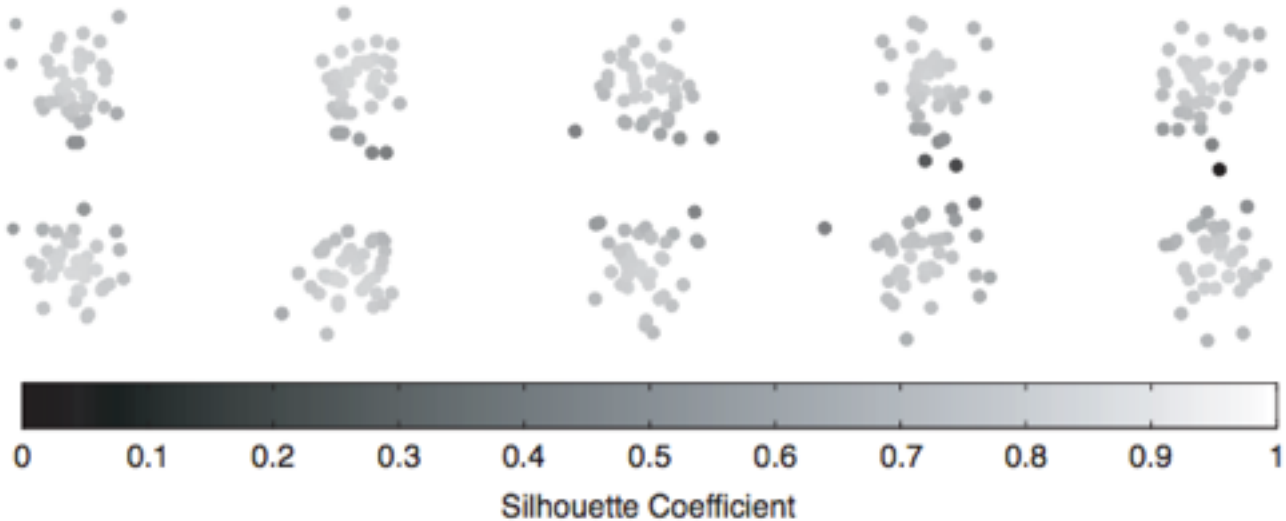


Figure 8.29. Silhouette coefficients for points in ten clusters.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

One useful application of cluster validation is to determine the best number of clusters for your dataset.

Q: *How would you do this?*
A: *By computing the overall SSE or SC for different values of k.*

Then treat k as a model parameter and **cross-validate***!*

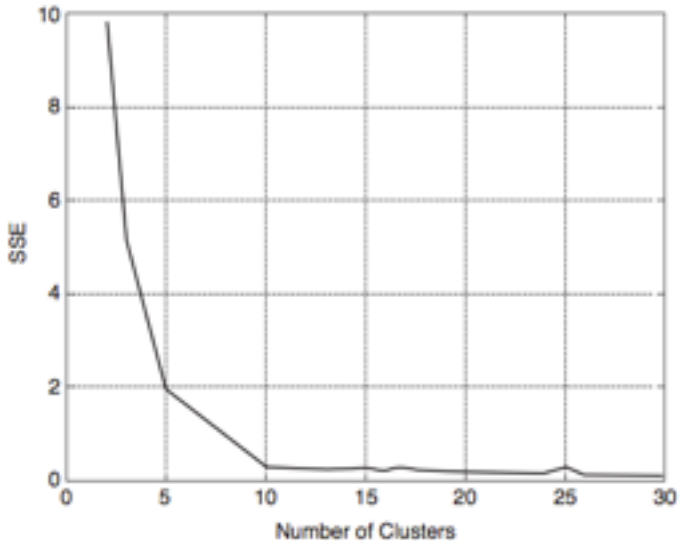


Figure 8.32. SSE versus number of clusters for the data of Figure 8.29.

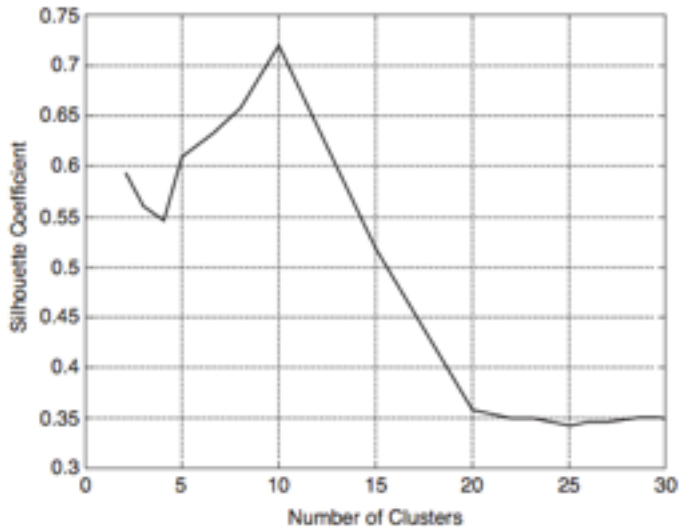


Figure 8.33. Average silhouette coefficient versus number of clusters for the data of Figure 8.29.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

	continuous	*categorical*
supervised	*regression*	*classification*
unsupervised	*dim reduction*	*clustering*

Q: *What is* **dimensionality reduction***?*

A: *A set of techniques for* **reducing the size** *(in terms of features, records, and/or bytes)* **of the dataset** *under examination.*

In general, the idea is to regard the dataset as a matrix and to **decompose the matrix** *into simpler, meaningful pieces.*

Dimensionality reduction is frequently performed as a **pre-processing step** *before another learning algorithm is applied.*

Q: *What are the motivations for dimensionality reduction?*

A: *The number of features in our dataset can be difficult to manage, or even misleading (eg, if the relationships are actually simpler than they appear).*

Q: *What is the goal of dimensionality reduction?*

A:

- *reduce computational expense*
- *reduce susceptibility to overfitting*
- *reduce noise in the dataset*
- *enhance our intuition*
- *reduce multicollinearity*

Q: *How is dimensionality reduction performed?*

A: *There are two approaches:* **feature selection** *and* **feature extraction.**

feature selection – *selecting a subset of features using an external criterion (filter) or the learning algorithm accuracy itself (wrapper)*

feature extraction – *mapping the features to a lower dimensional space*

Q: *How do we perform* **feature selection***?*

A: *By making use of* **wrappers, filters,** *or* **embedded methods**

wrappers *- potential feature subsets are compared based on the success of* **built models** *projected via* **cross-validation**

filters *- feature subsets are determined based on some simple prescribed metric over the features*

embedded *- feature selection happens within the model-building itself*

The goal of **feature extraction** *is to create a new set of coordinates that simplify the representation of the data.*

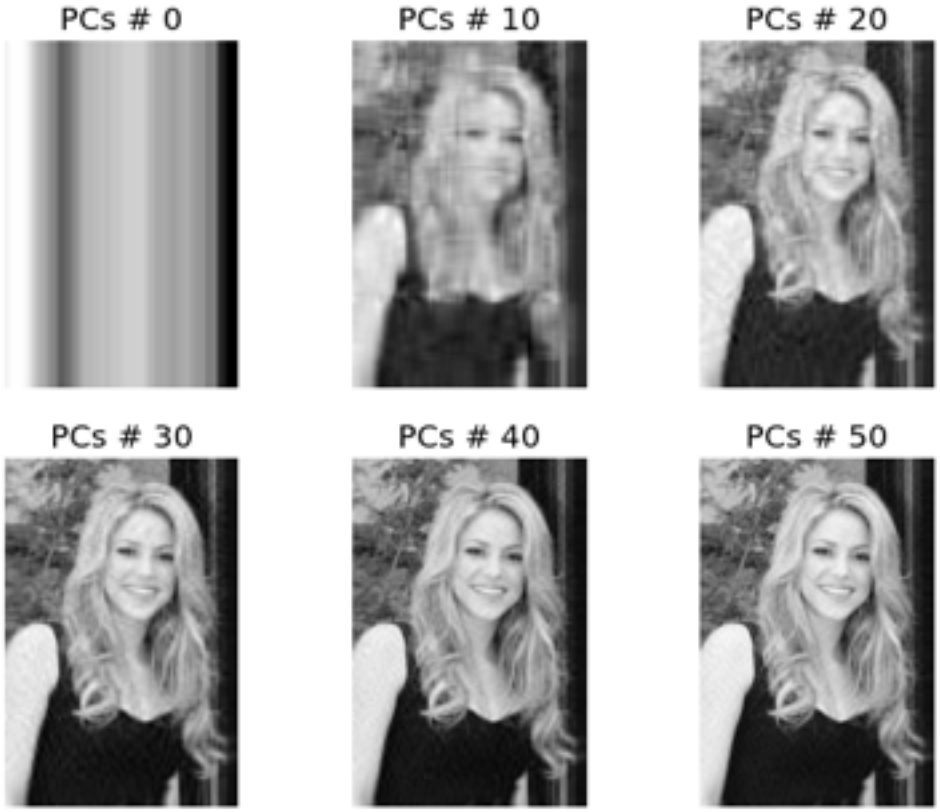
Typically we do this by using **matrix factorizations** *to map the features to a lower-dimensional space that minimizes information loss.*

Two prominent examples of such matrix factorization methods are **Principal Component Analysis (PCA)** *and* **Singular Value Decomposition (SVD)**

Q: *So what comes out of a PCA?*

A: *Eigenvectors and eigenvalues.*

- *Eigenvectors are linear combinations of the original feature vectors*
- *Each eigenvector represents a feature in our new transformed feature space*
- *The eigenvalues represent a measure of the amount of variance explained by each corresponding eigenvector ("new feature")*
- *We can choose only the first k (whatever we like) of our "new features" from the eigenvector space and work with them as our new data knowing we'll have minimal data loss for a feature space of that size*

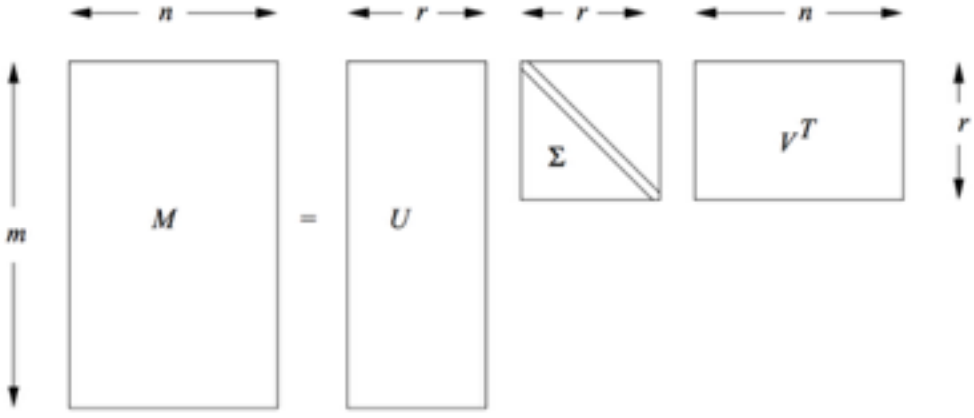


source: http://glowingpython.blogspot.it/2011/07/pca-and-image-compression-with-numpy.html

The **singular value decomposition** *of M is given by:*

$$M = U \Sigma V^T$$

(m x n) (m x r) (r x r) (r x n)



The nonzero entries of Σ are the **singular values** *of M. These are real, nonnegative, and rank-ordered (decreasing from left to right).*

NOTE

The number of singular values is equal to the *rank* of M.

The rank of a matrix measures its *non-degeneracy.*

How to reduce dimensions?

Drop Low Singular Values

$$\underset{M'}{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}} =$$

$$\underset{U}{\begin{bmatrix} .13 & .02 & -.01 \\ .41 & .07 & -.03 \\ .55 & .09 & -.04 \\ .68 & .11 & -.05 \\ .15 & -.59 & .65 \\ .07 & -.73 & -.67 \\ .07 & -.29 & .32 \end{bmatrix}} \underset{\Sigma}{\begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \\ .40 & -.80 & .40 & .09 & .09 \end{bmatrix}}$$

$$\begin{bmatrix} .13 & .02 \\ .41 & .07 \\ .55 & .09 \\ .68 & .11 \\ .15 & -.59 \\ .07 & -.73 \\ .07 & -.29 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \end{bmatrix}$$

$$= \begin{bmatrix} 0.93 & 0.95 & 0.93 & .014 & .014 \\ 2.93 & 2.99 & 2.93 & .000 & .000 \\ 3.92 & 4.01 & 3.92 & .026 & .026 \\ 4.84 & 4.96 & 4.84 & .040 & .040 \\ 0.37 & 1.21 & 0.37 & 4.04 & 4.04 \\ 0.35 & 0.65 & 0.35 & 4.87 & 4.87 \\ 0.16 & 0.57 & 0.16 & 1.98 & 1.98 \end{bmatrix}$$

Q: *What is* **NLP***?*

A: *A field of computer science, artificial intelligence, and linguistics concerned with the interaction between computers and human languages.*

The goal is for computers to derive meaning from human natural language input.

There are some general considerations that NLP faces...

Q: *What is* **tokenization***?*

A: *Tokenization is the process of breaking up streams up text into words, phrases, symbols, or other meaningful elements called* **tokens**

These tokens become inputs for further ML processing and generally allow us to put text information into data vectors (a vector space, this "vectorizing" of the data is always the first step in any ML problem).

Q: *What is* **stemming***?*

A: *Stemming is reducing words to that share the same root (verb forms, plurals, etc) to their root word for further processing.*

The idea is that the semantic information is captured by the root, and that retaining all the different forms just adds noise and complexity.

Q: *What is* **TFIDF** *weighting?*

A: Term-Frequency Inverse-Document-Frequency *assigns a weighting scheme that is proportional to the frequency of a token within a group of words and inversely proportional to the token frequency in the entire set of documents (corpus).*

We will apply TFIDF weighting to generate better vectors for our ML algorithms.

Q: *What is a* **Bag of Words Model***?*

A: *The idea that the order of the words in a document don't matter much in terms of semantic or conceptual meaning of the document.*

This will allow us to ignore word order and just treat each possible token as a static coordinate in a vector space (with (possibly weighted) word frequencies as the coordinate values).

Q: *What is a* **Cosine Similarity***?*

A: *A common metric for similarity used in vector spaces resulting from bag of words models on text analysis.*

It's the dot product of 2 vectors divided by the norms of the 2 vectors:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Q: *What is a* **Latent Semantic Analysis***?*

A: *LSA is a technique that combines an appropriate TFIDF weighting with a bag of words model to vectorize many documents of text data.*

The resulting **Term-Document Matrix** *is then reduced using an* **SVD***.*

The output yields reduced Term and Document vector spaces which allow Term-Term, Term-Document, and Document-Document similarity comparisons via cosine similarity in the (drastically) reduced-dimensionality space.

Q: *What are* **Recommender Systems***?*

A: *Automated systems that seek to suggest whether a given* **item** *(product, event, movie, song, etc) will be desirable to a* **user***.*

They often build on the back of machine learning concepts we've seen previously.

They've become ubiquitous in today's web-based world, so there are many different applications...

There are two general approaches to their design:

In **content-based filtering**, **items** *are mapped into a feature space, and recommendations depend on* **specified characteristics**.

In contrast, the only data under consideration in **collaborative filtering** *are* **user-item ratings**, *and recommendations depend on user preferences.*

Q: What are ensemble techniques?
A: Methods of improving classification accuracy by aggregating predictions over several **base classifiers**.

Ensembles are often much more accurate than the base classifiers that compose them.

Of course we can extend these ideas to things like regression, clustering, or other ML techniques.

In order for an ensemble classifier to outperform a single base classifier, the following conditions must be met:

1) *the bc's must be* **accurate***: they must outperform random guessing*

2) *the bc's must be* **diverse***: their misclassifications must occur on different training examples*

Q: How do you create an ensemble classifier?

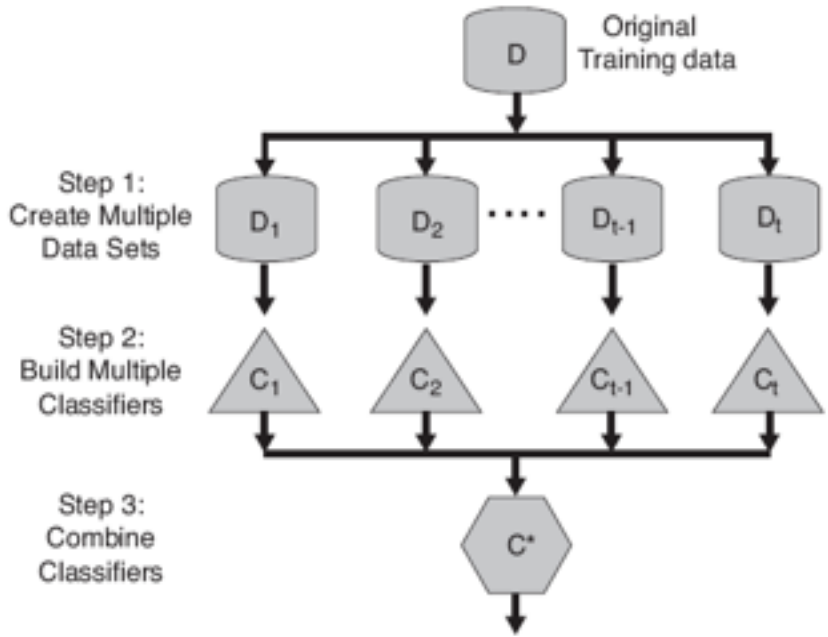


Figure 5.31. A logical view of the ensemble learning method.

Bagging *(bootstrap aggregating) is a method that involves manipulating the training set by* **resampling**.

NOTE

Resampling means that some training records may appear in a sample more than once, or even not at all.

We learn k base classifiers on k different samples of training data.

These samples are independently created by resampling the training data using uniform weights (eg, a uniform **sampling distribution***).*

A random forest is a common ensemble of decision trees where each base classifier is grown using a random effect.

1) *Select a sample of the original training set and build a tree as follows:*
    1) Randomly *select m features out of the M available*
    2) *Pick the best split based on just those m variables*
2) *Repeat (1) for N iterations*
3) *Predict based on majority vote of N trees*

Boosting is an iterative procedure that adaptively changes the sampling distribution of training records at each iteration.

The first iteration uses uniform weights (like bagging). In subsequent iterations, the weights are adjusted to emphasize records that were misclassified in previous iterations.

The final prediction is constructed by a weighted vote (where the weights for a bc depends on its training error).

1) *Start with a training set where each item has equal weight*
2) *Build a classifier - G*
3) *Compute the error e(G) of the classifier (% incorrect)*
4) *Build a new training set where the incorrect instances are weighted (repeated) by the error e(G) of the classifier*
5) *Repeat*
6) *Predict based on majority vote (or vote based on accuracy – inverse error)*

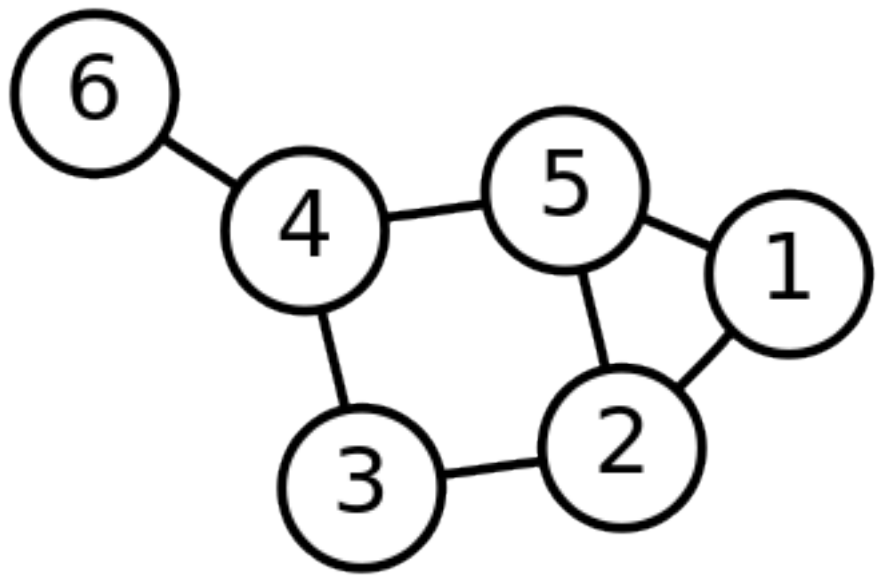
Q: What are **Networks***?*

A: A set of **pairwise relationships** *between* **objects***.*

The ubiquity of social networks gives rise to many interesting data-oriented questions that can be answered with analytical techniques.

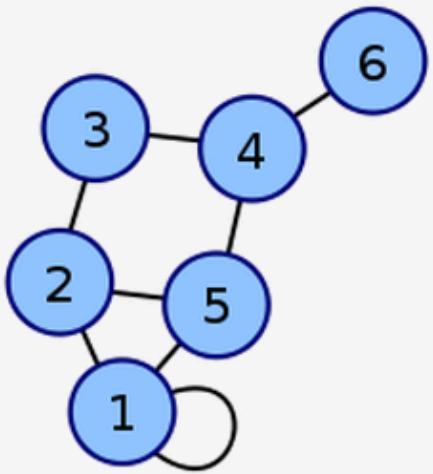
Given a large set of social network data, what types of questions do you think would be interesting to ask?

Nodes *represent* **actors** *in the graph, and* **edges** *represent the* **relationships** *between actors.*

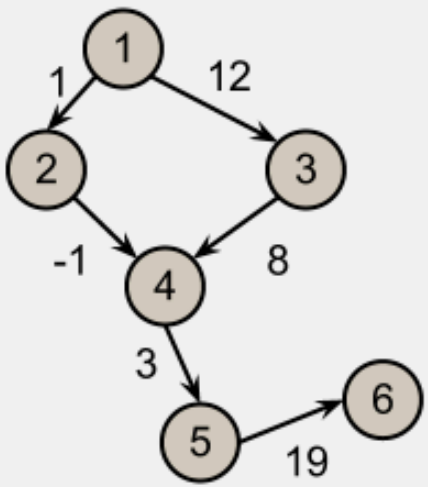


NOTE

An *undirected graph* has no directionality in its edges (bidirectional).

Labeled graph	Adjacency matrix
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ Coordinates are 1-6.

Weighted Directed Graph & Adjacency Matrix



Weighted Directed Graph

	1	2	3	4	5	6
1	0	1	12	0	0	0
2	-1	0	0	-1	0	0
3	-12	0	0	8	0	0
4	0	1	-8	0	3	0
5	0	0	0	-3	0	19
6	0	0	0	0	-19	0

Adjacency Matrix

NOTE

A *directed graph* has an asymmetric adjacency matrix. Can you see why?

One key concept in the study of network structure is **centrality**. *The* **centrality** *of a node is a measure of its* **importance** *in the network.*

The simplest centrality measure is the **degree** *of a node, which is simply the number of edges connected to it. Using the adjacency matrix notation for an undirected graph, we can express the degree* k_i *of node* i *as:*

$$k_i = \sum_{j=1}^{n} A_{ij}.$$

A more sophisticated measure called **eigenvector centrality** *allows important edges to give larger contributions to centrality:*

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j,$$

Here the eigenvector centrality x_i of node i is proportional to the average centrality of i's network neighbors.

Another useful centrality measure is based on the idea of shortest-distance (or geodesic) paths through the graph.

NOTE

Betweenness centrality measures the proportion of geodesic paths passing through a node.

This gives an idea of the node's *influence* in the network.

If σ_{st} is the number of geodesic paths from node s to node t, and $\sigma_{st}(v)$ is the number of these paths that cross node v, then the **betweenness centrality** *of node v is given by:*

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Q: What is a **time series***?*

A: A sequence of datapoints where each has an associated timestamp.

Q: How might we perform time series **forecasting***?*
A: **Autoregressive modeling**

Q: What are autoregressive models?
A: Regression models that use previous observations of the target variable as features to predict the target variable at the current time.

The Autoregressive Model:

$$X_t = \sum_{i=1}^{p} \theta_i X_{t-i}$$

We solve for the values of the thetas just like we would for any normal regression model.

Q: What is **data stream mining***?*

A: Extracting knowledge from ***continuous***, ***rapidly streaming*** *data sources.*

Can you think of any such questions that might be interesting?

Q: How can we accomplish data stream mining?

A: 2 general approaches:

- *Incremental Algorithms*
- *Periodic Batch Retraining*

Q: What are incremental algorithms?

A: ML algorithms that don't need to retrain on all of the data at once to maintain the model.

They can simply update themselves ***incrementally*** *based on the delta data.*

Q: **Big Data** *is a hot topic these days, what does it actually refer to?*

A:

- **Volume***: more data than can fit in memory on 1 machine*
- **Velocity***: data coming in faster than we can process (think Twitter)*
- **Variety***: Data across all different types*
- **Veracity***: Uncertainty of data*

Q: **Big Data** *is a hot topic these days, what does it actually refer to?*

A:

We're talking about more data than can fit on a single computer.

We have exponentially growing data AND exponentially growing computing power.

Q: What is **Hadoop***?*

A: A platform/framework for distributed computing.

Arose out of several papers from Google in the early 2000s.

Allows scaling big data analysis outwards over clusters of computers.

Handles fault tolerance, scheduling, monitoring

Q: What is **Hadoop***?*

A: At its core, it consists of 2 components:

- **Hadoop Distributed Filesystem (HDFS):**
  - *Filesystem optimized for dealing with big data on clusters of computers*
- **MapReduce:**
  - *Functional programming paradigm for simplifying large scale highly parallelizable computations*

As we've discussed, the map-reduce approach involves splitting a problem into subtasks and processing these subtasks in parallel.

This takes place in (approximately) two phases:

1) the **mapper** *phase*
1.5) shuffle/sort
2) the **reducer** *phase*

To implement MapReduce, you have to (at a minimum) write 2 functions:

1) the **mapper***: filter and transform data*

- (key, value) —> (key, value)

2) the **reducer***: outputs a new (key, value) by aggregating pairs with matching keys via some user-defined aggregation function*

- (key, List<values>) —> (key, value)

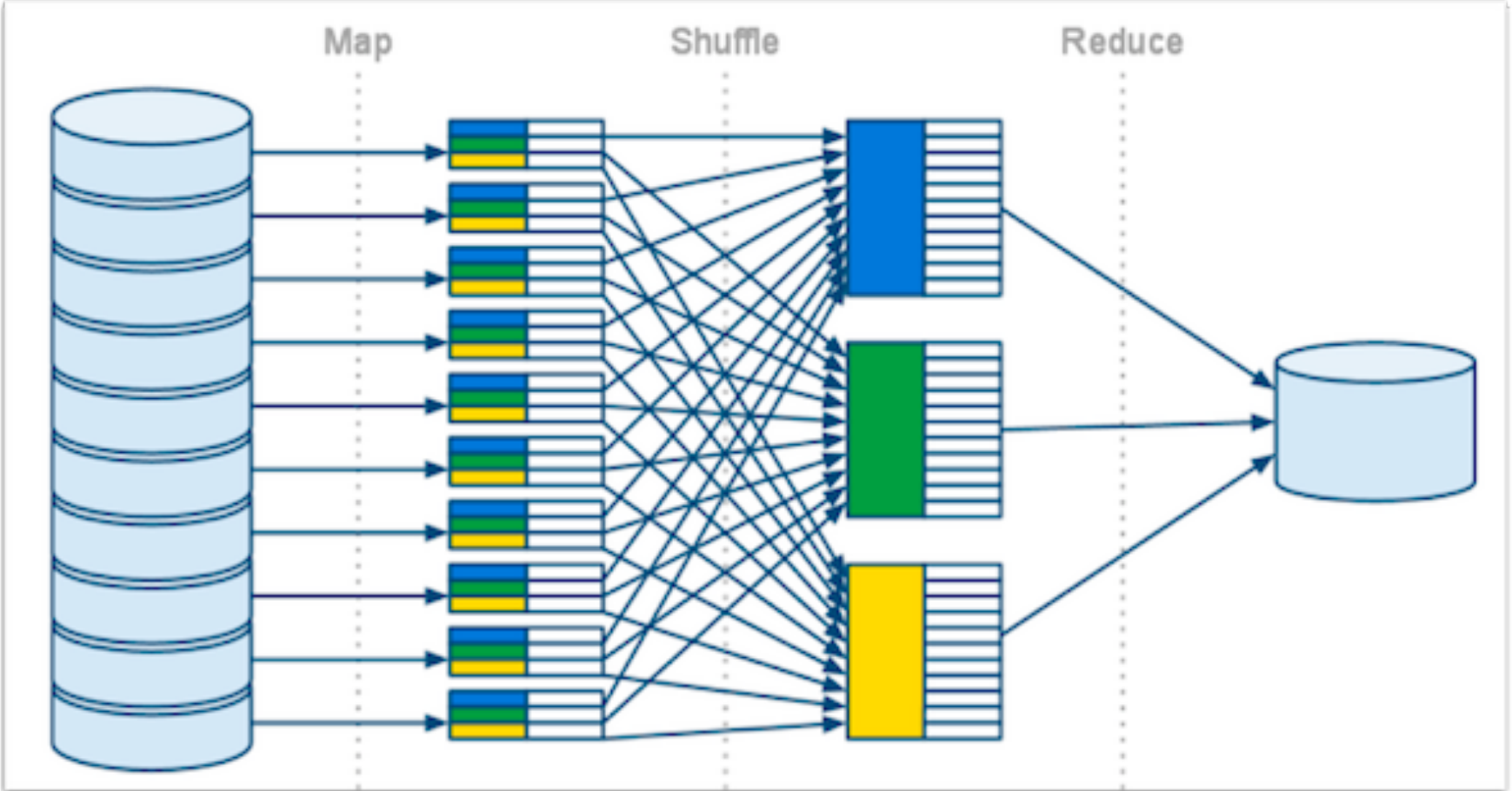
Hadoop will handle everything else for you if you want.

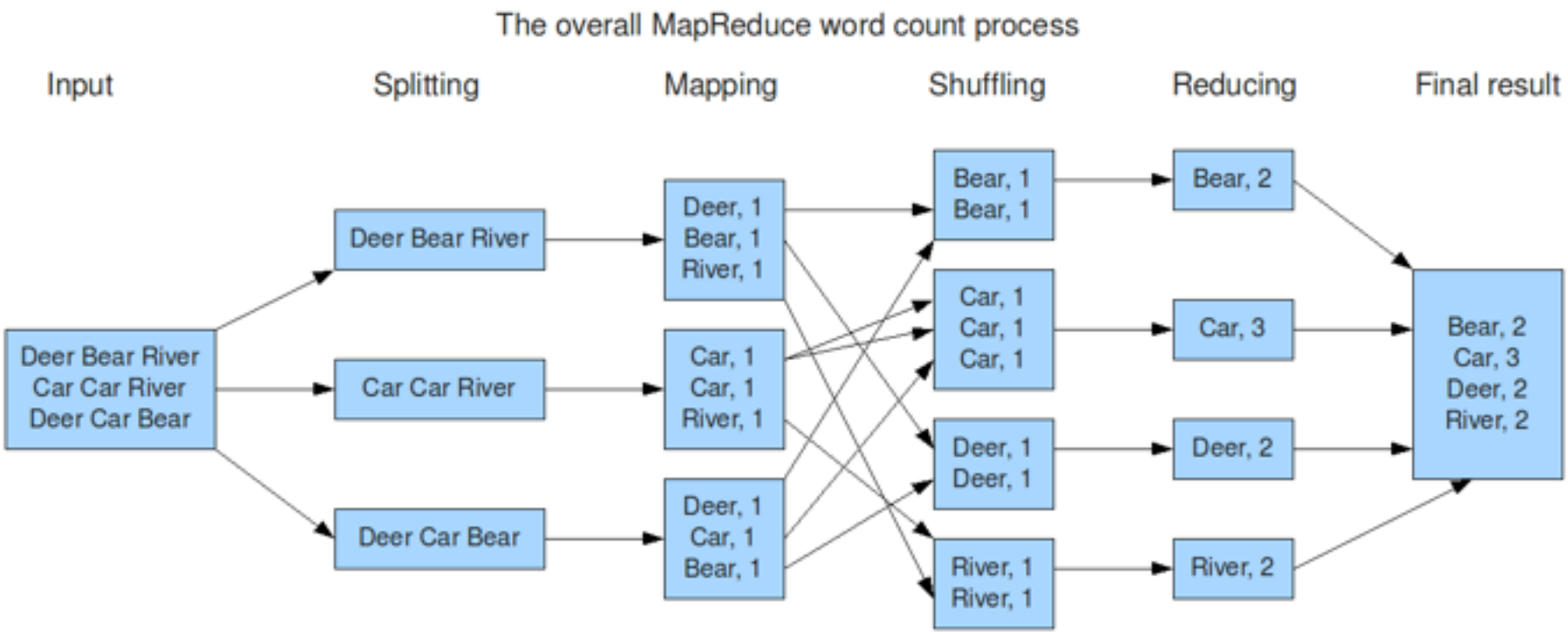
Map-reduce processes data in terms of key-value pairs:

```
input              <k1, v1>

mapper             <k1, v1> → <k2, v2>

(partitioner)      <k2, v2> → <k2, [all k2 values]>

reducer            <k2, [all k2 values]> → <k3, v3>
```

As our earlier diagram suggests, there are additional intermediate steps in a map-reduce workflow.

mappers – *filter & transform data*
combiners – *perform reducer operations on the mapper node (optional step, to reduce network traffic and disk I/O).*
partitioners – *shuffle/sort/redirect mapper output*
reducers – *aggregate results*


Map
Shuffle
Reduce


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Apache Hive

- SQL language to query data on HDFS

- Queries are translated behind the scenes into map-reduce jobs

- Data is stored on HDFS, but a metadata database contains the table schemas

Apache Pig

- Scripting language to operate on data on HDFS

- Queries are translated behind the scenes into map-reduce jobs

- Allows for similar declarative functionality to hive, but often in a simpler and more intuitive scripting format

Q: What is Spark?

A: An open source framework that combines an engine to distribute programs across clusters of machines with an elegant model for writing programs on top of it.

Arguably the first open source software that makes distributed programming truly accessible to the data science workflow!

- *Spark Core: contains the basic functionality of Spark*
  - *APIs that define Resilient Distributed Datasets (RDDs) and operations and actions that can be performed on them*
  - *The rest of Spark's libraries are built on top of the RDD and Spark Core*

- *Spark SQL:*
  - *APIs for interacting with data in Spark via the Apache Hive variant of SQL HiveQL (Hive Query Language)*
  - *Every DB table is represented as an RDD and Spark SQL queries are translated to Spark Operations*
  - *Can function as drop-in replacement for Hive*

- *Spark Streaming:*
  - *Enables the processing and manipulation of live data streams in real time*
  - *Many streaming data libraries (e.g. Apache Storm) exist for handling streaming data in real-time*
  - *Spark Streaming enables programs to leverage this data similar to how you would interact with a normal RDD as data is flowing in*

- *Spark MLLib:*
  - *A library of common ML algorithms implemented as Spark operations on RDDs*
  - *Contains scalable algorithms like classifications, regressions, etc*
  - *Mahout, the former choice for this task, will move to Spark for future things*

Amazon provides a suite of services to allow you to easily manage all of your servers (hardware, software, OS) completely in the Amazon cloud.

This is called Amazon Web Services (AWS)

Compute

- ec2: resizable compute capacity in the cloud

- lambda: run code immediately in response to events like website clicks, service requests

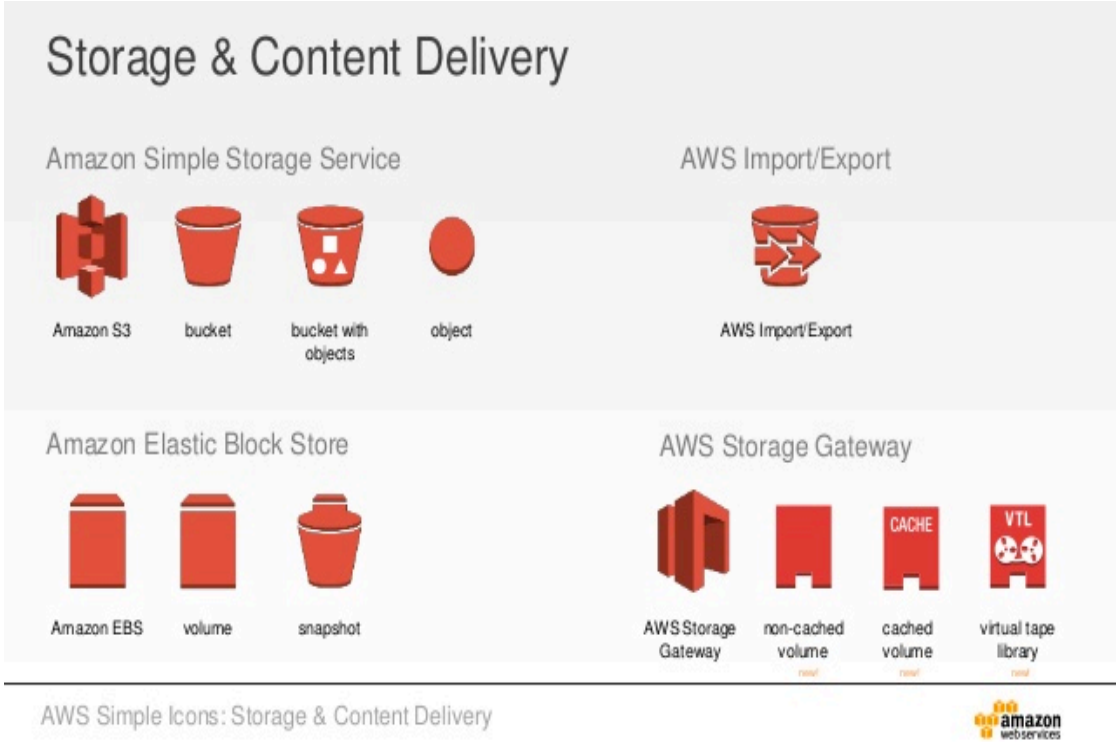
Storage

- s3: secure, durable, scalable object storage

- Storage Gateway: connect on-premises hardware with cloud

- Glacier: long term archival storage

- Cloudfront: deliver content to users



Databases

- RDS: run relational DBs

- DynamoDB: NoSQL DB in the cloud

- ElastiCache: In-memory key-value cache

- Redshift: Data Warehousing in the cloud

Networking

- VPC: provision segment of cloud for private network

- Direct Connect: private connection directly to AWS cloud

- Route 53: DNS Server

Analytics

- EMR: Managed Hadoop, Spark framework

- Kinesis: Data stream processing

- DataPipeline: Data processing/transfer at regular intervals

Amazon Elastic MapReduce

II. WHERE SHALL WE GO?

The more you know, the more you know you don't know.

– Aristotle

- *The world of Data Science/ML is HUGE and ever GROWING!*
- *We've covered the main algorithms in today's Data Science toolkit*
- *If you have a good understanding of these concepts, you should be well equipped to continue forward on your own.*
- *Basically, now you have a better idea of all the things you don't know! Pick a topic that interests you, and go be the world's best at it!*
- *Questions?*