MA3700 Mathematical Methods for Data Mining - Lecture1

A. Balinsky

Cardiff School of Mathematics

Spring 2018

Standard Formulation:

Data for data mining comes in many forms: from computer files typed in by human operators, business information in SQL or some other standard database format, information recorded automatically by equipment such as fault logging devices, from streams of binary data transmitted from satellites.

For purposes of data mining we will assume that the data takes a particular **standard form** which is described below.

- We will assume that for any data mining application we have a ***universe of objects*** that are of interest.

Standard Formulation:

Data for data mining comes in many forms: from computer files typed in by human operators, business information in SQL or some other standard database format, information recorded automatically by equipment such as fault logging devices, from streams of binary data transmitted from satellites.

For purposes of data mining we will assume that the data takes a particular **standard form** which is described below.

- We will assume that for any data mining application we have a ***universe of objects*** that are of interest.

- The universe of objects is normally **very large** and we have only a **small part** of it. Usually we want to extract information from the data available to us that *we hope is applicable* to the large volume of data that we have not yet seen.

- Each object is described by a number of variables that correspond to its properties. In data mining variables are often called attributes.We will use both terms in this course.

- The set of variable values corresponding to each of the objects is called a record or (more commonly) an instance. The complete set of data available to us for an application is called a dataset. A dataset is often depicted as a table, with each row representing an instance.

- Labelled data: where one attribute is given special significance and the aim is to predict its value (e.g. "spam", "not spam").

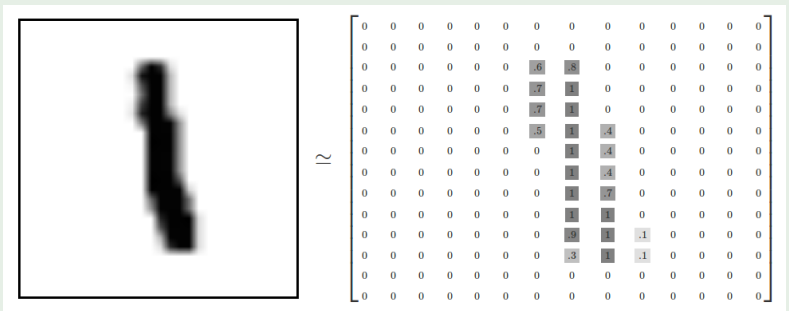
- Unlabelled data: when there is no such significant attribute.

- The set of variable values corresponding to each of the objects is called a record or (more commonly) an instance. The complete set of data available to us for an application is called a dataset. A dataset is often depicted as a table, with each row representing an instance.

- Labelled data: where one attribute is given special significance and the aim is to predict its value (e.g. "spam", "not spam").

- Unlabelled data: when there is no such significant attribute.

Example (MNIST (60000 train, 10000 test), Labelled data)



Example (Netflix Prize (Unlabelled data))

Types of Variable

In general there are many types of variable that can be used to measure the properties of an object. At least six main types of variable can be distinguished.

- **Nominal Variables**: *A variable used to put objects into categories*, e.g. the name or colour of an object.
  A nominal variable may be numerical in form, but the numerical values have no mathematical interpretation. *They are simply labels*.

- **Binary Variables**: A binary variable is a special case of a nominal variable that takes only two possible values: true or false, 1 or 0 etc.

Types of Variable

In general there are many types of variable that can be used to measure the properties of an object. At least six main types of variable can be distinguished.

- **Nominal Variables**: *A variable used to put objects into categories*, e.g. the name or colour of an object.
  A nominal variable may be numerical in form, but the numerical values have no mathematical interpretation. *They are simply labels*.

- **Binary Variables**: A binary variable is a special case of a nominal variable that takes only two possible values: true or false, 1 or 0 etc.

Types of Variable

In general there are many types of variable that can be used to measure the properties of an object. At least six main types of variable can be distinguished.

- **Nominal Variables**: *A variable used to put objects into categories*, e.g. the name or colour of an object.
  A nominal variable may be numerical in form, but the numerical values have no mathematical interpretation. *They are simply labels*.

- **Binary Variables**: A binary variable is a special case of a nominal variable that takes only two possible values: true or false, 1 or 0 etc.

- **Ordinal Variables**: Ordinal variables are similar to nominal variables, except that an ordinal variable has values that can be arranged in a meaningful order, e.g. small, medium, large.

- **Integer Variables**: Integer variables are ones that take values that are genuine integers, for example "number of children".

- **Interval-scaled Variables**: Interval-scaled variables are variables that take numerical values which are measured at equal intervals from a zero point or origin. However the origin does not imply a true absence of the measured characteristic.

- **Ratio-scaled Variables**: Ratio-scaled variables are similar to interval-scaled variables except that the zero point does reflect the absence of the measured characteristic.

- **Ordinal Variables**: Ordinal variables are similar to nominal variables, except that an ordinal variable has values that can be arranged in a meaningful order, e.g. small, medium, large.

- **Integer Variables**: Integer variables are ones that take values that are genuine integers, for example "number of children".

- **Interval-scaled Variables**: Interval-scaled variables are variables that take numerical values which are measured at equal intervals from a zero point or origin. However the origin does not imply a true absence of the measured characteristic.

- **Ratio-scaled Variables**: Ratio-scaled variables are similar to interval-scaled variables except that the zero point does reflect the absence of the measured characteristic.

- **Ordinal Variables**: Ordinal variables are similar to nominal variables, except that an ordinal variable has values that can be arranged in a meaningful order, e.g. small, medium, large.

- **Integer Variables**: Integer variables are ones that take values that are genuine integers, for example "number of children".

- **Interval-scaled Variables**: Interval-scaled variables are variables that take numerical values which are measured at equal intervals from a zero point or origin. However the origin does not imply a true absence of the measured characteristic.

- **Ratio-scaled Variables**: Ratio-scaled variables are similar to interval-scaled variables except that the zero point does reflect the absence of the measured characteristic.

- **Ordinal Variables**: Ordinal variables are similar to nominal variables, except that an ordinal variable has values that can be arranged in a meaningful order, e.g. small, medium, large.

- **Integer Variables**: Integer variables are ones that take values that are genuine integers, for example "number of children".

- **Interval-scaled Variables**: Interval-scaled variables are variables that take numerical values which are measured at equal intervals from a zero point or origin. However the origin does not imply a true absence of the measured characteristic.

- **Ratio-scaled Variables**: Ratio-scaled variables are similar to interval-scaled variables except that the zero point does reflect the absence of the measured characteristic.

Although the distinction between different categories of variable can be important in some cases, many practical data mining systems divide attributes into just two types:

- **Categorical**: corresponding to nominal, binary and ordinal variables.
- **Continuous**: corresponding to integer, interval-scaled and ratio-scaled variables.

It is *important* to choose methods that are appropriate to the types of variable stored for a particular application.

Although the distinction between different categories of variable can be important in some cases, many practical data mining systems divide attributes into just two types:

- **Categorical**: corresponding to nominal, binary and ordinal variables.
- **Continuous**: corresponding to integer, interval-scaled and ratio-scaled variables.

It is *important* to choose methods that are appropriate to the types of variable stored for a particular application.

Although the distinction between different categories of variable can be important in some cases, many practical data mining systems divide attributes into just two types:

- **Categorical**: corresponding to nominal, binary and ordinal variables.
- **Continuous**: corresponding to integer, interval-scaled and ratio-scaled variables.

It is *important* to choose methods that are appropriate to the types of variable stored for a particular application.

Although the distinction between different categories of variable can be important in some cases, many practical data mining systems divide attributes into just two types:

- **Categorical**: corresponding to nominal, binary and ordinal variables.
- **Continuous**: corresponding to integer, interval-scaled and ratio-scaled variables.

It is *important* to choose methods that are appropriate to the types of variable stored for a particular application.

Instance-Based Versus Model-Based Learning

There are two main approaches to generalization: **instance-based** learning and **model-based** learning.

Instance-based learning: the system learns the examples by heart, then generalizes to new cases using a similarity measure

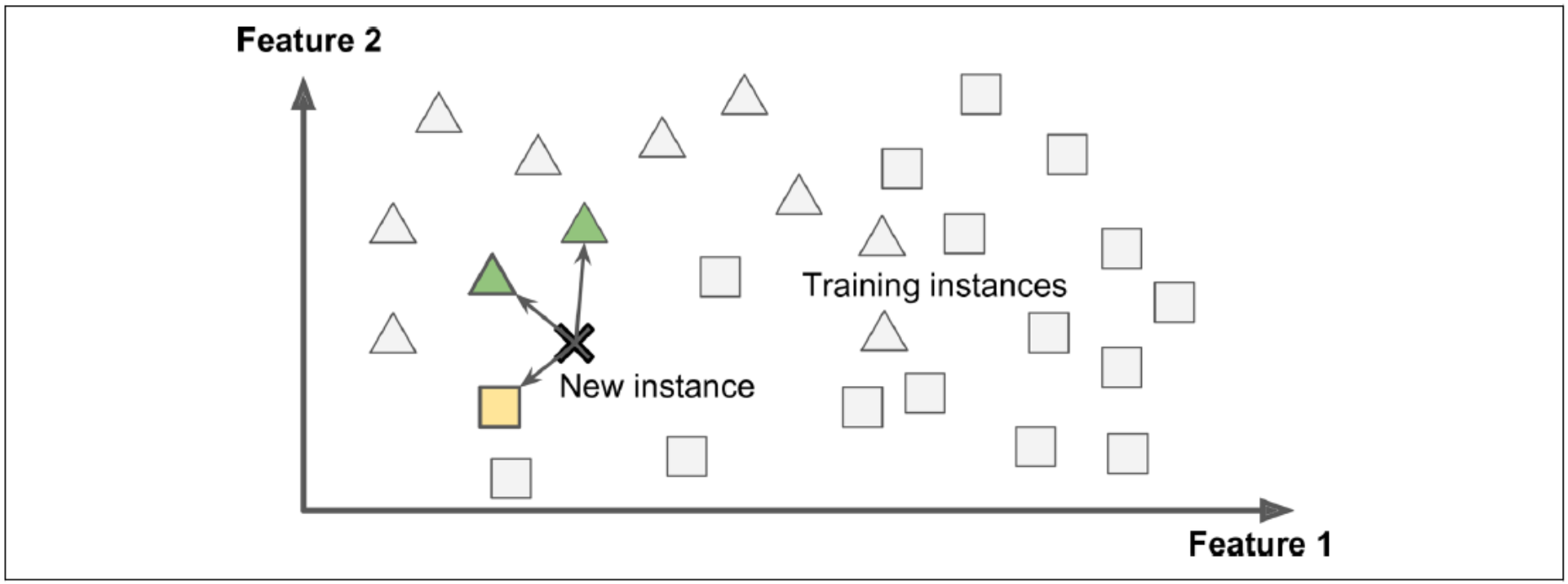


Figure 1-15. Instance-based learning

Model-based learning

Another way to generalize from a set of examples is to build a model of these examples, then use that model to make predictions. This is called model-based learning

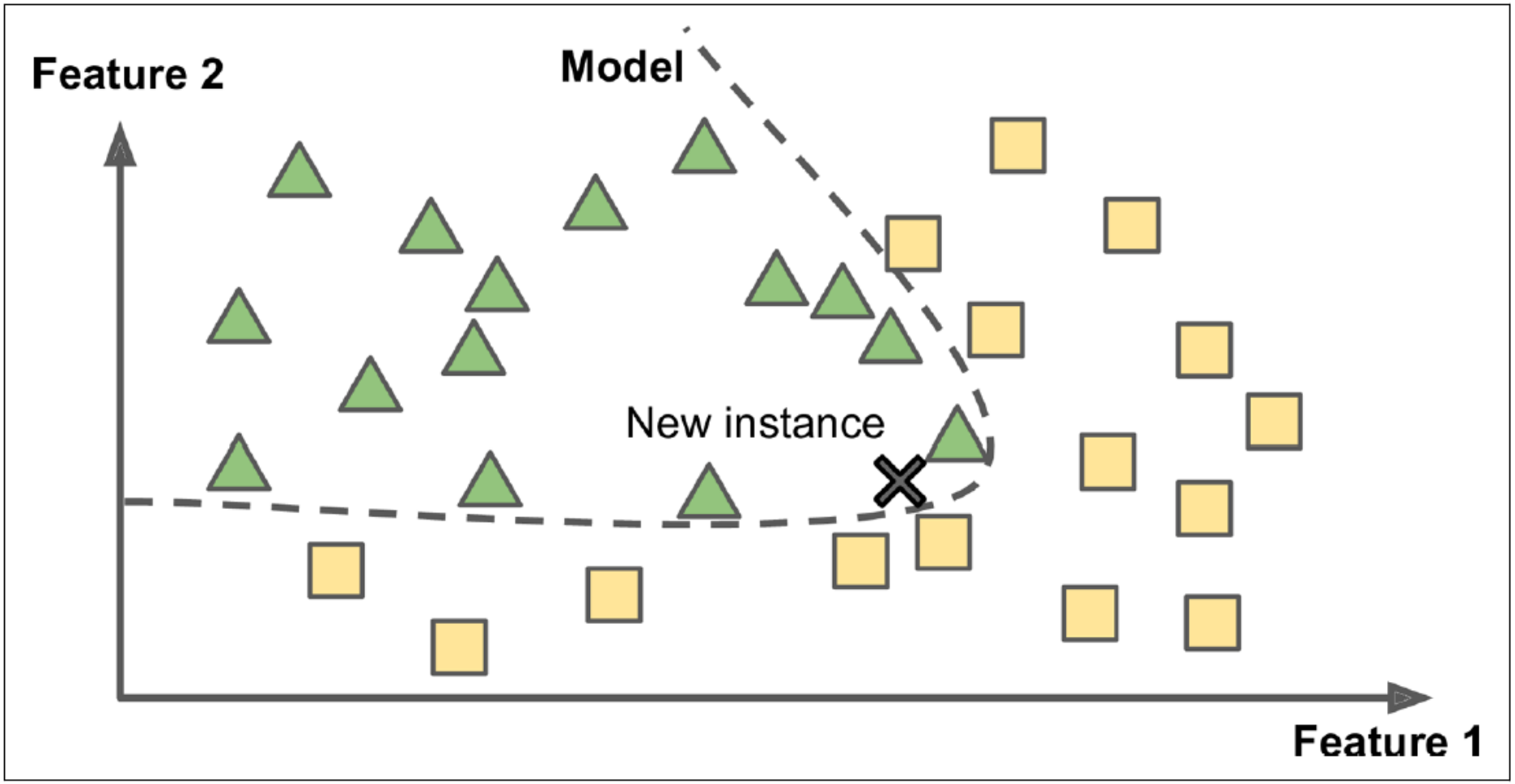


Figure 1-16. Model-based learning

Formalization of Model-based Learning

First, let us formalize the most common *mathematical framework for learning*.

1. We are given **training examples**

$$\mathcal{D} = \{z_1, z_2, \ldots, z_n\}$$

with the z_i being examples sampled from an **unknown process** $P(Z)$.

2. We are also given a **loss functional** L which takes as argument a **decision function** f from a **hypothesis space**, $\mathcal{H}$, and an example z, and returns a real-valued scalar.
3. We want to minimize the **expected value** of $L(f, Z)$ under the unknown generating process $P(Z)$.

Formalization of Model-based Learning

First, let us formalize the most common *mathematical framework for learning*.

1. We are given **training examples**

$$\mathcal{D} = \{z_1, z_2, \ldots, z_n\}$$

   with the z_i being examples sampled from an **unknown process** $P(Z)$.
2. We are also given a **loss functional** L which takes as argument a **decision function** f from a **hypothesis space**, $\mathcal{H}$, and an example z, and returns a real-valued scalar.
3. We want to minimize the **expected value** of $L(f, Z)$ under the unknown generating process $P(Z)$.

Formalization of Model-based Learning

First, let us formalize the most common *mathematical framework for learning*.

1. We are given **training examples**

$$\mathcal{D} = \{z_1, z_2, \ldots, z_n\}$$

   with the z_i being examples sampled from an **unknown process** $P(Z)$.
2. We are also given a **loss functional** L which takes as argument a **decision function** f from a **hypothesis space**, $\mathcal{H}$, and an example z, and returns a real-valued scalar.
3. We want to minimize the **expected value** of $L(f, Z)$ under the unknown generating process $P(Z)$.

Formalization of Model-based Learning

First, let us formalize the most common *mathematical framework for learning*.

1. We are given **training examples**

$$\mathcal{D} = \{z_1, z_2, \ldots, z_n\}$$

   with the z_i being examples sampled from an **unknown process** $P(Z)$.
2. We are also given a **loss functional** L which takes as argument a **decision function** f from a **hypothesis space**, $\mathcal{H}$, and an example z, and returns a real-valued scalar.
3. We want to minimize the **expected value** of $L(f, Z)$ under the unknown generating process $P(Z)$.

Ockham's razor principle: We prefer the simplest hypothesis consistent with the data.

Supervised Learning:

In supervised learning, each examples is an (*input*,*target*) pair: $z = (x, y)$ and f takes an x as argument.

The most common examples are

- **regression**: y is a real-valued scalar or vector, the output of f is in the *same set* of values as y, and we often take as loss functional the **squared error**: $L(f, (x, y)) = \|f(x) - y\|^2$.
- **classification**: y is a finite integer (e.g. a symbol) corresponding to a *class index*, and we often take as loss function the **negative conditional log-likelihood**, with the interpretation that $f_i(x)$ estimates $P(y = i|x)$: $L(f, (x, y)) = -\log f_y(x)$, where we have the constraints $f_y(x) \geq 0 \quad \sum_i f_i(x) = 1$

Ockham's razor principle: We prefer the simplest hypothesis consistent with the data.

Supervised Learning:

In supervised learning, each examples is an (*input*,*target*) pair: $z = (x, y)$ and f takes an x as argument.

The most common examples are

- **regression**: y is a real-valued scalar or vector, the output of f is in the *same set* of values as y, and we often take as loss functional the **squared error**: $L(f, (x, y)) = \|f(x) - y\|^2$.
- **classification**: y is a finite integer (e.g. a symbol) corresponding to a *class index*, and we often take as loss function the **negative conditional log-likelihood**, with the interpretation that $f_i(x)$ estimates $P(y = i|x)$: $L(f, (x, y)) = -\log f_y(x)$, where we have the constraints $f_y(x) \geq 0 \quad \sum_i f_i(x) = 1$

Ockham's razor principle: We prefer the simplest hypothesis consistent with the data.

Supervised Learning:

In supervised learning, each examples is an (*input*,*target*) pair: $z = (x, y)$ and f takes an x as argument.

The most common examples are

- **regression**: y is a real-valued scalar or vector, the output of f is in the *same set* of values as y, and we often take as loss functional the **squared error**: $L(f, (x, y)) = \|f(x) - y\|^2$.
- **classification**: y is a finite integer (e.g. a symbol) corresponding to a *class index*, and we often take as loss function the **negative conditional log-likelihood**, with the interpretation that $f_i(x)$ estimates $P(y = i|x)$: $L(f, (x, y)) = -\log f_y(x)$, where we have the constraints $f_y(x) \geq 0 \quad \sum_i f_i(x) = 1$

Ockham's razor principle: We prefer the simplest hypothesis consistent with the data.

Supervised Learning:

In supervised learning, each examples is an (*input*,*target*) pair: $z = (x, y)$ and f takes an x as argument.

The most common examples are

- **regression**: y is a real-valued scalar or vector, the output of f is in the *same set* of values as y, and we often take as loss functional the **squared error**: $L(f, (x, y)) = \|f(x) - y\|^2$.
- **classification**: y is a finite integer (e.g. a symbol) corresponding to a *class index*, and we often take as loss function the **negative conditional log-likelihood**, with the interpretation that $f_i(x)$ estimates $P(y = i|x)$: $L(f, (x, y)) = -\log f_y(x)$, where we have the constraints $f_y(x) \geq 0 \quad \sum_i f_i(x) = 1$

Unsupervised Learning

In unsupervised learning we are learning a function f which helps to characterize the unknown distribution $P(Z)$.

Sometimes f is directly an estimator of $P(Z)$ itself (this is called **density estimation**).
In many other cases f is an attempt to characterize where the density **concentrates**.
Clustering algorithms divide up the input space in regions. Some clustering algorithms create a hard partition (e.g. the k-means algorithm) while others construct a soft partition (e.g. a Gaussian mixture model) which assign to each Z a probability of belonging to each cluster.

Another kind of unsupervised learning algorithms are those that construct a new **representation** for Z.

Unsupervised Learning

In unsupervised learning we are learning a function f which helps to characterize the unknown distribution $P(Z)$.

Sometimes f is directly an estimator of $P(Z)$ itself (this is called **density estimation**).
In many other cases f is an attempt to characterize where the density **concentrates**.
Clustering algorithms divide up the input space in regions. Some clustering algorithms create a hard partition (e.g. the k-means algorithm) while others construct a soft partition (e.g. a Gaussian mixture model) which assign to each Z a probability of belonging to each cluster.

Another kind of unsupervised learning algorithms are those that construct a new **representation** for Z.

Unsupervised Learning

In unsupervised learning we are learning a function f which helps to characterize the unknown distribution $P(Z)$.

Sometimes f is directly an estimator of $P(Z)$ itself (this is called **density estimation**).
In many other cases f is an attempt to characterize where the density **concentrates**.
Clustering algorithms divide up the input space in regions. Some clustering algorithms create a hard partition (e.g. the k-means algorithm) while others construct a soft partition (e.g. a Gaussian mixture model) which assign to each Z a probability of belonging to each cluster.

Another kind of unsupervised learning algorithms are those that construct a new **representation** for Z.

Main Steps of Data Mining

Step 1: Representing Data in the "Standard Form".

For "**standard**" data mining tasks the data is presented to the data mining system in the *standard form*. There are a fixed number of *attributes* (or *features*) which were chosen before the data was collected.

For example, for text mining the dataset usually comprises the documents themselves and the *features are extracted from the documents automatically based on their content before the classification algorithm is applied*.

There are generally a very large number of features, most of them only occurring rarely, with a high proportion of noisy and irrelevant features.

Main Steps of Data Mining

Step 1: Representing Data in the "Standard Form".

For "**standard**" data mining tasks the data is presented to the data mining system in the *standard form*. There are a fixed number of *attributes* (or *features*) which were chosen before the data was collected.

For example, for text mining the dataset usually comprises the documents themselves and the *features are extracted from the documents automatically based on their content before the classification algorithm is applied*.

There are generally a very large number of features, most of them only occurring rarely, with a high proportion of noisy and irrelevant features.

Step 2: Distance Measures and Similarity Measures.

A fundamental data-mining problem is to examine data for "similar" items. Local Generalization!

This is the **main idea** of Data Mining!

Step 3: Algorithms.

- k-nearest neighbour classification,
- Naive Bayes,
- the k-means algorithm,
- PageRank algorithm,
- support vector machines,
- AdaBoost,
- the expectation-maximization (EM) algorithm,
- graph mining algorithms,
- ...

Step 2: Distance Measures and Similarity Measures.

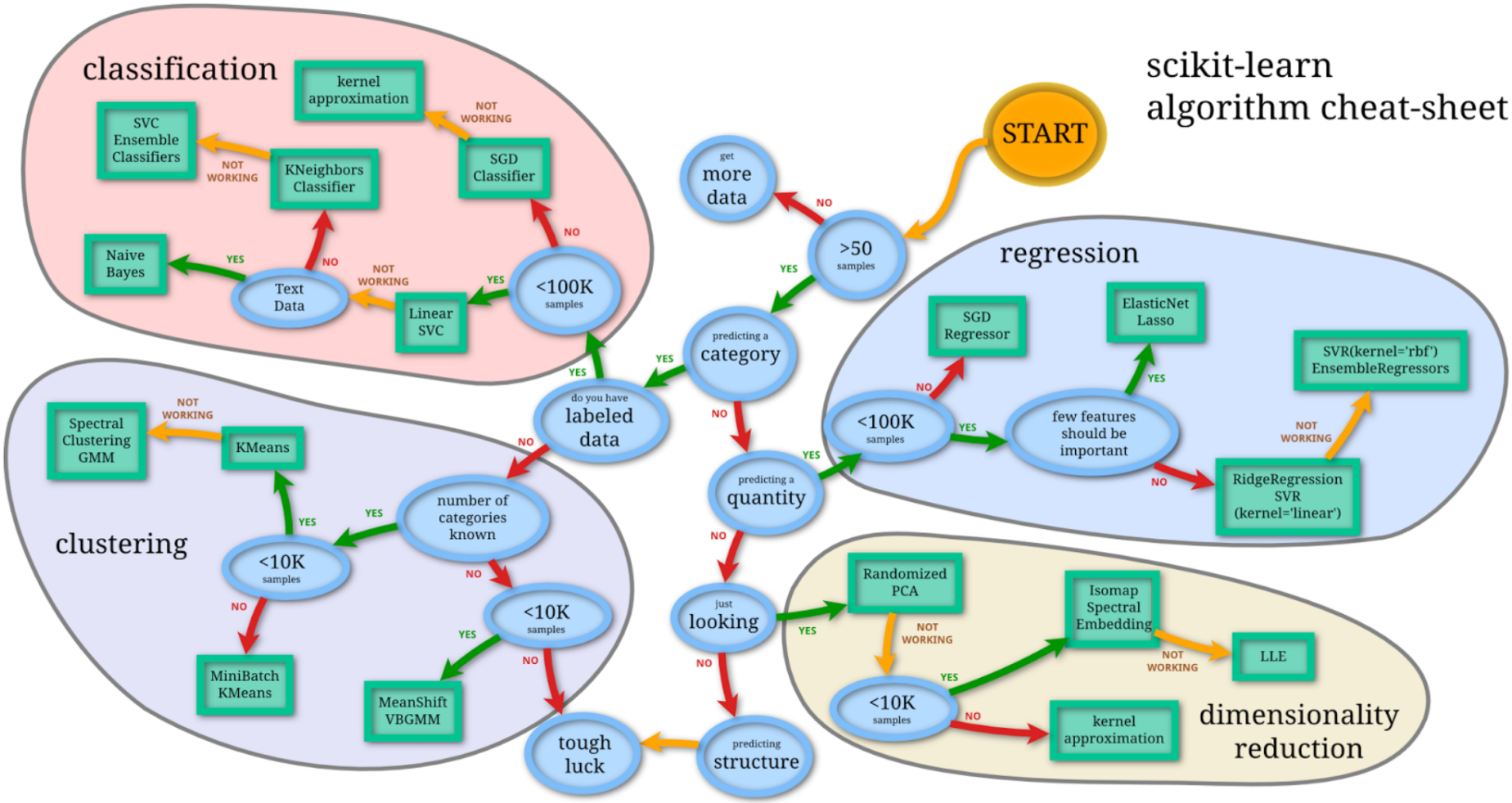
A fundamental data-mining problem is to examine data for "similar" items. Local Generalization!

This is the **main idea** of Data Mining!

Step 3: Algorithms.

- k-nearest neighbour classification,
- Naive Bayes,
- the k-means algorithm,
- PageRank algorithm,
- support vector machines,
- AdaBoost,
- the expectation-maximization (EM) algorithm,
- graph mining algorithms,
- ...

Selecting an Algorithm



http://peekaboo-vision.blogspot.com

Things Useful to Know

In this section, we offer brief introductions to subjects that you may or may not have seen in your study of other courses. Each will be useful in the study of data mining. They include:

- Softmax function.
- Cross entropy.
- The TF.IDF measure of word importance.
- Glossary and concepts of graph theory.

Softmax function

In mathematics, the *softmax function*, or *normalized exponential function*, is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $softmax(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function is given by

$$softmax : \mathbb{R}^K \to [0,1]^K$$

$$softmax(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

for $j = 1, \ldots, K$. In probability theory, the output of the softmax function can be used to represent a categorical distribution, that is, a probability distribution over K different possible outcomes.

Cross entropy In information theory, the *cross entropy* between two probability distributions p and q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an *"unnatural"* probability distribution q, rather than the "*true*" distribution p.

The cross entropy for the distributions p and q over a given set is defined as follows:

$$H(p, q) = \mathrm{E}_p[-\log q] = H(p) + D_{\mathrm{KL}}(p \| q),$$

where $H(p)$ is the entropy of p, and $D_{\mathrm{KL}}(p \| q)$ is the *Kullback-Leibler divergence* of q from p (also known as the relative entropy of p with respect to q).

For discrete p and q this means

$$H(p, q) = -\sum_x p(x) \log q(x).$$

Example (MNIST simple Demo, Jupyter notebook)

Importance of Words in Documents and Vector Space Model

In several applications of data mining, we shall be faced with the problem of categorizing documents (sequences of words) by their topic.

Textual Data Preparation:

- Words and Sentences Extraction.
- Removing *stop words*.
- *Stemming*.

Importance of Words in Documents and Vector Space Model

In several applications of data mining, we shall be faced with the problem of categorizing documents (sequences of words) by their topic.

Textual Data Preparation:

- Words and Sentences Extraction.
- Removing *stop words*.
- *Stemming*.

Importance of Words in Documents and Vector Space Model

In several applications of data mining, we shall be faced with the problem of categorizing documents (sequences of words) by their topic.

Textual Data Preparation:

- Words and Sentences Extraction.
- Removing *stop words*.
- *Stemming*.

Importance of Words in Documents and Vector Space Model

In several applications of data mining, we shall be faced with the problem of categorizing documents (sequences of words) by their topic.

Textual Data Preparation:

- Words and Sentences Extraction.
- Removing *stop words*.
- *Stemming*.

Typically, topics are identified by finding the special words that characterize documents about that topic.

For instance, articles about baseball would tend to have many occurrences of words like "*ball*", "*bat*", "*pitch*", "*run*" and so on.

Once we have classified documents to determine they are about baseball, it is not hard to notice that words such as these appear unusually frequently.

However, until we have made the classification, it is not possible to identify these words as characteristic.

Typically, topics are identified by finding the special words that characterize documents about that topic.

For instance, articles about baseball would tend to have many occurrences of words like "*ball*", "*bat*", "*pitch*", "*run*" and so on.

Once we have classified documents to determine they are about baseball, it is not hard to notice that words such as these appear unusually frequently.

However, until we have made the classification, it is not possible to identify these words as characteristic.

Remember:

Machine learning has not proved successful in situations where we can describe the goals of the mining more directly.

Thus, classification often starts by looking at documents, and finding the **significant words** in those documents.

Our first guess might be that the words appearing most frequently in a document are the most significant. However, that intuition is exactly opposite of the truth.

In fact, the indicators of the topic are relatively rare words.

Remember:

Machine learning has not proved successful in situations where we can describe the goals of the mining more directly.

Thus, classification often starts by looking at documents, and finding the **significant words** in those documents.

Our first guess might be that the words appearing most frequently in a document are the most significant. However, that intuition is exactly opposite of the truth.

In fact, the indicators of the topic are relatively rare words.

However, not all rare words are equally useful as indicators.

There are certain words, for example "*notwithstanding*" or "*albeit*," that appear rarely in a collection of documents, yet do not tell us anything useful.

On the other hand, a word like "*chukker*" is probably equally rare, but tips us off that the document is about the sport of polo.

The *difference* between rare words that tell us something and those that do not has to do with the ***concentration of the useful words in just a few documents***.

However, not all rare words are equally useful as indicators.

There are certain words, for example "*notwithstanding*" or "*albeit*," that appear rarely in a collection of documents, yet do not tell us anything useful.

On the other hand, a word like "*chukker*" is probably equally rare, but tips us off that the document is about the sport of polo.

The *difference* between rare words that tell us something and those that do not has to do with the ***concentration of the useful words in just a few documents***.

However, not all rare words are equally useful as indicators.

There are certain words, for example "*notwithstanding*" or "*albeit*," that appear rarely in a collection of documents, yet do not tell us anything useful.

On the other hand, a word like "*chukker*" is probably equally rare, but tips us off that the document is about the sport of polo.

The *difference* between rare words that tell us something and those that do not has to do with the ***concentration of the useful words in just a few documents***.

That is, the presence of a word like "albeit" in a document does not make it terribly more likely that it will appear multiple times.

However, if an article mentions "chukker" once, it is likely to tell us what happened in the "first chukker," then the "second chukker," and so on.

That is, **the word is likely to be repeated if it appears at all**.

The formal measure of how concentrated into relatively few documents are the occurrences of a given word is called TF.IDF

That is, the presence of a word like "albeit" in a document does not make it terribly more likely that it will appear multiple times.

However, if an article mentions "chukker" once, it is likely to tell us what happened in the "first chukker," then the "second chukker," and so on.

That is, **the word is likely to be repeated if it appears at all**.

The formal measure of how concentrated into relatively few documents are the occurrences of a given word is called TF.IDF

Term Frequency times Inverse Document Frequency: $TF \times IDF$

Suppose we have a collection of N documents.
Define f_{ij} to be the *frequency* (number of occurrences) of term (word) i in document j.

Then, define the term frequency TF_{ij} to be:

Term Frequency

$$TF_{ij} := \frac{f_{ij}}{\max_k f_{kj}}.$$

Thus, the most frequent term in document j gets a TF of 1, and other terms get fractions as their term frequency for this document.

Term Frequency times Inverse Document Frequency: $TF \times IDF$

Suppose we have a collection of N documents.
Define f_{ij} to be the *frequency* (number of occurrences) of term (word) i in document j.

Then, define the term frequency TF_{ij} to be:

Term Frequency

$$TF_{ij} := \frac{f_{ij}}{\max_k f_{kj}}.$$

Thus, the most frequent term in document j gets a TF of 1, and other terms get fractions as their term frequency for this document.

The *IDF* for a term is defined as follows.

Suppose term i appears in n_i of the N documents in the collection. Then

Inverse Document Frequency

$$IDF_i = \log_2 \left(\frac{N}{n_i} \right).$$

The TF.IDF score for term i in document j is then defined to be

$TF \times IDF$

$$TF_{ij} \times IDF_i.$$

The terms with the highest TF.IDF score are often the terms that best characterize the topic of the document.

Example

Suppose our repository consists of $2^{20} = 1,048,576$ documents.
Suppose word w appears in $2^{10} = 1024$ of these documents.
Then $IDF_w = \log_2(2^{20}/2^{10}) = \log_2(2^{10}) = 10$.

Consider a document j in which w appears 20 times, and that is the maximum number of times in which any word appears (*perhaps after eliminating stop words*).
Then $TF_{wj} = 1$, and the TF.IDF score for w in document j is 10.

Suppose that in document k, word w appears once, while the maximum number of occurrences of any word in this document is 20.
Then $TF_{wk} = 1/20$, and the TF.IDF score for w in document k is $1/2$.

Example

Suppose our repository consists of $2^{20} = 1,048,576$ documents.
Suppose word w appears in $2^{10} = 1024$ of these documents.
Then $IDF_w = \log_2(2^{20}/2^{10}) = \log_2(2^{10}) = 10$.

Consider a document j in which w appears 20 times, and that is the maximum number of times in which any word appears (*perhaps after eliminating stop words*).
Then $TF_{wj} = 1$, and the TF.IDF score for w in document j is 10.

Suppose that in document k, word w appears once, while the maximum number of occurrences of any word in this document is 20.
Then $TF_{wk} = 1/20$, and the TF.IDF score for w in document k is $1/2$.

Example

Suppose our repository consists of $2^{20} = 1,048,576$ documents.
Suppose word w appears in $2^{10} = 1024$ of these documents.
Then $IDF_w = \log_2(2^{20}/2^{10}) = \log_2(2^{10}) = 10$.

Consider a document j in which w appears 20 times, and that is the maximum number of times in which any word appears (*perhaps after eliminating stop words*).
Then $TF_{wj} = 1$, and the TF.IDF score for w in document j is 10.

Suppose that in document k, word w appears once, while the maximum number of occurrences of any word in this document is 20.
Then $TF_{wk} = 1/20$, and the TF.IDF score for w in document k is $1/2$.

Vector Space Model.
Document-Term Matrix for a Corpus of Documents.

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word vector (passage vector)

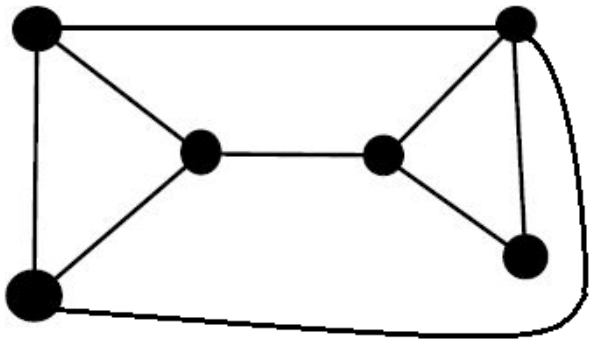
Document vector

$$
\mathrm{M}\begin{pmatrix}
 & D_1 & D_2 & D_3 & D_4 & D_5 & D_6 & \cdots & D_n \\
T_1 & 0.00060 & 0.00012 & 0.00003 & 0.00003 & 0.00333 & 0.00048 & \cdots & a_{1n} \\
T_2 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_{2n} \\
T_3 & 0 & 2.98862 & 0 & 0 & 0 & 1.49431 & \cdots & a_{3n} \\
T_4 & 0 & 0 & 0 & 13.32555 & 0 & 0 & \cdots & a_{4n} \\
T_5 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_{5n} \\
T_6 & 1.03442 & 1.03442 & 0 & 0 & 0 & 3.10326 & \cdots & a_{6n} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
T_m & a_{m1} & a_{m2} & a_{m3} & a_{m4} & a_{m5} & a_{m6} & \cdots & a_{mn}
\end{pmatrix}
$$

Glossary and Concepts of Graph Theory

Formally, a **graph** $G = (V, E)$ is a mathematical structure consisting of a finite nonempty set V of **vertices** or **nodes**, and a set $E \subseteq V \times V$ of **edges** consisting of *unordered* pairs of vertices.

An edge from a node to itself, (v_i, v_i), is called a **loop**.
An undirected graph without loops is called a **simple graph**. Unless mentioned explicitly, we will consider a graph to be simple.



An edge $e = (v_i, v_j)$ between v_i and v_j is said to be **incident** with nodes v_i and v_j ; in this case we also say that v_i and v_j are **adjacent** to one another, and that they are **neighbours**. We denote this by $v_i \backsim v_j$.

The number of nodes in the graph G, given as $|V| = n$, is called the **order** of the graph, and the number of edges in the graph, given as $|E| = m$, is called the **size** of G.

A **directed graph** or *digraph* has an edge set E consisting of *ordered pairs* of vertices.
A directed edge (v_i, v_j) is also called an **arc**, and is said to be from v_i to v_j .
We also say that v_i is the *tail* and v_j the *head* of the arc.

An edge $e = (v_i, v_j)$ between v_i and v_j is said to be **incident** with nodes v_i and v_j ; in this case we also say that v_i and v_j are **adjacent** to one another, and that they are **neighbours**. We denote this by $v_i \backsim v_j$.

The number of nodes in the graph G, given as $|V| = n$, is called the **order** of the graph, and the number of edges in the graph, given as $|E| = m$, is called the **size** of G.

A **directed graph** or *digraph* has an edge set E consisting of *ordered pairs* of vertices.
A directed edge (v_i, v_j) is also called an **arc**, and is said to be from v_i to v_j .
We also say that v_i is the *tail* and v_j the *head* of the arc.

A **weighted graph** consists of a graph together with a weight w_{ij} for each edge $(v_i, v_j) \in E$.

Such weights might represent, for example, costs, lengths or capacities, etc., depending on the problem at hand.
Every graph can be considered to be a weighted graph in which the edges have weight one.

Weights are usually real numbers. **We always assume that weights are strictly positive**.

Subgraphs

A graph $H = (V_H, E_H)$ is called a **subgraph** of $G = (V, E)$ if $V_H \subseteq V$ and $E_H \subseteq E$.
We also say that G is a **supergraph** of H.

Given a subset of the vertices $V' \subseteq V$, the **induced subgraph** $G' = (V', E')$ consists exactly of all the edges present in G between vertices in V'.
More formally, for all $v_i, v_j \in V'$, $(v_i, v_j) \in E' \iff (v_i, v_j) \in E$. In other words, two nodes are adjacent in G' if and only if they are adjacent in G.
A (sub)graph is called **complete** (or a **clique**) if there exists an edge between all pairs of nodes.

Subgraphs

A graph $H = (V_H, E_H)$ is called a **subgraph** of $G = (V, E)$ if $V_H \subseteq V$ and $E_H \subseteq E$.
We also say that G is a **supergraph** of H.

Given a subset of the vertices $V' \subseteq V$, the **induced subgraph** $G' = (V', E')$ consists exactly of all the edges present in G between vertices in V'.
More formally, for all $v_i, v_j \in V'$, $(v_i, v_j) \in E' \iff (v_i, v_j) \in E$. In other words, two nodes are adjacent in G' if and only if they are adjacent in G.
A (sub)graph is called **complete** (or a **clique**) if there exists an edge between all pairs of nodes.

Degree

The **degree** of a node $v_i \in V$ is the number of edges incident with it, and is denoted as $d(v_i)$ or just d_i .

The *degree sequence* of a graph is the list of the degrees of the nodes sorted in *non-increasing* order.

Let N_k denote the number of vertices with degree k. The **degree frequency** distribution of a graph is given as $(N_0, N_1, \ldots, N_t)$ where t is the maximum degree for a node in G.

Let X be a random variable denoting the degree of a node. The *degree distribution* of a graph gives the probability mass function f for X, given as $(f(0), f(1), \ldots, f(t))$, where $f(k) = P(X = k) = \frac{N_k}{n}$ is the probability of a node with degree k, given as the number of nodes N_k with degree k, divided by the total number of nodes n.

Degree

The **degree** of a node $v_i \in V$ is the number of edges incident with it, and is denoted as $d(v_i)$ or just d_i .

The *degree sequence* of a graph is the list of the degrees of the nodes sorted in *non-increasing* order.

Let N_k denote the number of vertices with degree k. The **degree frequency** distribution of a graph is given as $(N_0, N_1, \ldots, N_t)$ where t is the maximum degree for a node in G.

Let X be a random variable denoting the degree of a node. The *degree distribution* of a graph gives the probability mass function f for X, given as $(f(0), f(1), \ldots, f(t))$, where $f(k) = P(X = k) = \frac{N_k}{n}$ is the probability of a node with degree k, given as the number of nodes N_k with degree k, divided by the total number of nodes n.

Degree

The **degree** of a node $v_i \in V$ is the number of edges incident with it, and is denoted as $d(v_i)$ or just d_i .

The *degree sequence* of a graph is the list of the degrees of the nodes sorted in *non-increasing* order.

Let N_k denote the number of vertices with degree k. The **degree frequency** distribution of a graph is given as $(N_0, N_1, \ldots, N_t)$ where t is the maximum degree for a node in G.

Let X be a random variable denoting the degree of a node. The *degree distribution* of a graph gives the probability mass function f for X, given as $(f(0), f(1), \ldots, f(t))$, where $f(k) = P(X = k) = \frac{N_k}{n}$ is the probability of a node with degree k, given as the number of nodes N_k with degree k, divided by the total number of nodes n.

Degree

The **degree** of a node $v_i \in V$ is the number of edges incident with it, and is denoted as $d(v_i)$ or just d_i .

The *degree sequence* of a graph is the list of the degrees of the nodes sorted in *non-increasing* order.

Let N_k denote the number of vertices with degree k. The **degree frequency** distribution of a graph is given as $(N_0, N_1, \ldots, N_t)$ where t is the maximum degree for a node in G.

Let X be a random variable denoting the degree of a node. The *degree distribution* of a graph gives the probability mass function f for X, given as $(f(0), f(1), \ldots, f(t))$, where $f(k) = P(X = k) = \frac{N_k}{n}$ is the probability of a node with degree k, given as the number of nodes N_k with degree k, divided by the total number of nodes n.

For *directed graphs*, the **indegree** of node v_i , denoted as $id(v_i)$, is the number of edges with v_i as head, that is, the number of incoming edges at v_i .
The **outdegree** of v_i , denoted $od(v_i)$, is the number of edges with *vi* as the tail, that is, the number of outgoing edges from v_i .

Path and Distance

A *walk* in a graph G between nodes x and y is an ordered sequence of vertices, starting at x and ending at y,

$$x = v_0, v_1, \ldots, v_{t-1}, v_t = y$$

such that there is an edge between every pair of *consecutive* vertices, that is, $(v_{i-1}, v_i) \in E$ for all $i = 1, 2, \ldots, t$.

For *directed graphs*, the **indegree** of node v_i , denoted as $id(v_i)$, is the number of edges with v_i as head, that is, the number of incoming edges at v_i .
The **outdegree** of v_i , denoted $od(v_i)$, is the number of edges with *vi* as the tail, that is, the number of outgoing edges from v_i .

Path and Distance

A *walk* in a graph G between nodes x and y is an ordered sequence of vertices, starting at x and ending at y,

$$x = v_0, v_1, \ldots, v_{t-1}, v_t = y$$

such that there is an edge between every pair of *consecutive* vertices, that is, $(v_{i-1}, v_i) \in E$ for all $i = 1, 2, \ldots, t$.

The length of the walk, t , is measured in terms of hops - the number of edges along the walk. In a walk, there is no restriction on the number of times a given vertex may appear in the sequence; thus both the vertices and edges may be repeated.

A walk starting and ending at the same vertex (i.e., with $y = x$) is called **closed**. A **trail** is a walk with distinct edges, and a **path** is a walk with **distinct** vertices (with the exception of the start and end vertices).

A closed path with length $t \geq 3$ is called a **cycle**, that is, a cycle begins and ends at the same vertex and has distinct nodes.
A path of *minimum length* between nodes x and y is called a **shortest path**, and the length of the shortest path is called the **distance** between x and y, denoted as $d(x, y)$. If no path exists between the two nodes, the distance is assumed to be $d(x, y) = \infty$.

The length of the walk, t , is measured in terms of hops - the number of edges along the walk. In a walk, there is no restriction on the number of times a given vertex may appear in the sequence; thus both the vertices and edges may be repeated.

A walk starting and ending at the same vertex (i.e., with $y = x$) is called **closed**. A **trail** is a walk with distinct edges, and a **path** is a walk with **distinct** vertices (with the exception of the start and end vertices).

A closed path with length $t \geq 3$ is called a **cycle**, that is, a cycle begins and ends at the same vertex and has distinct nodes.
A path of *minimum length* between nodes x and y is called a **shortest path**, and the length of the shortest path is called the **distance** between x and y, denoted as $d(x, y)$. If no path exists between the two nodes, the distance is assumed to be $d(x, y) = \infty$.

The length of the walk, t , is measured in terms of hops - the number of edges along the walk. In a walk, there is no restriction on the number of times a given vertex may appear in the sequence; thus both the vertices and edges may be repeated.

A walk starting and ending at the same vertex (i.e., with $y = x$) is called **closed**. A **trail** is a walk with distinct edges, and a **path** is a walk with **distinct** vertices (with the exception of the start and end vertices).

A closed path with length $t \geq 3$ is called a **cycle**, that is, a cycle begins and ends at the same vertex and has distinct nodes.
A path of *minimum length* between nodes x and y is called a **shortest path**, and the length of the shortest path is called the **distance** between x and y, denoted as $d(x, y)$. If no path exists between the two nodes, the distance is assumed to be $d(x, y) = \infty$.

Connectedness

Two nodes v_i and v_j are said to be **connected** if there exists a path between them.
A graph is **connected** if there is a path between all pairs of vertices.

A **connected component**, or just component, of a graph is a maximal connected subgraph.

For a *directed* graph, we say that it is **strongly connected** if there is a (*directed*) path between all ordered pairs of vertices.

We say that it is **weakly connected** if there exists a path between node pairs only by considering edges as undirected.

Connectedness

Two nodes v_i and v_j are said to be **connected** if there exists a path between them.
A graph is **connected** if there is a path between all pairs of vertices.

A **connected component**, or just component, of a graph is a maximal connected subgraph.

For a *directed* graph, we say that it is **strongly connected** if there is a (*directed*) path between all ordered pairs of vertices.

We say that it is **weakly connected** if there exists a path between node pairs only by considering edges as undirected.

Connectedness

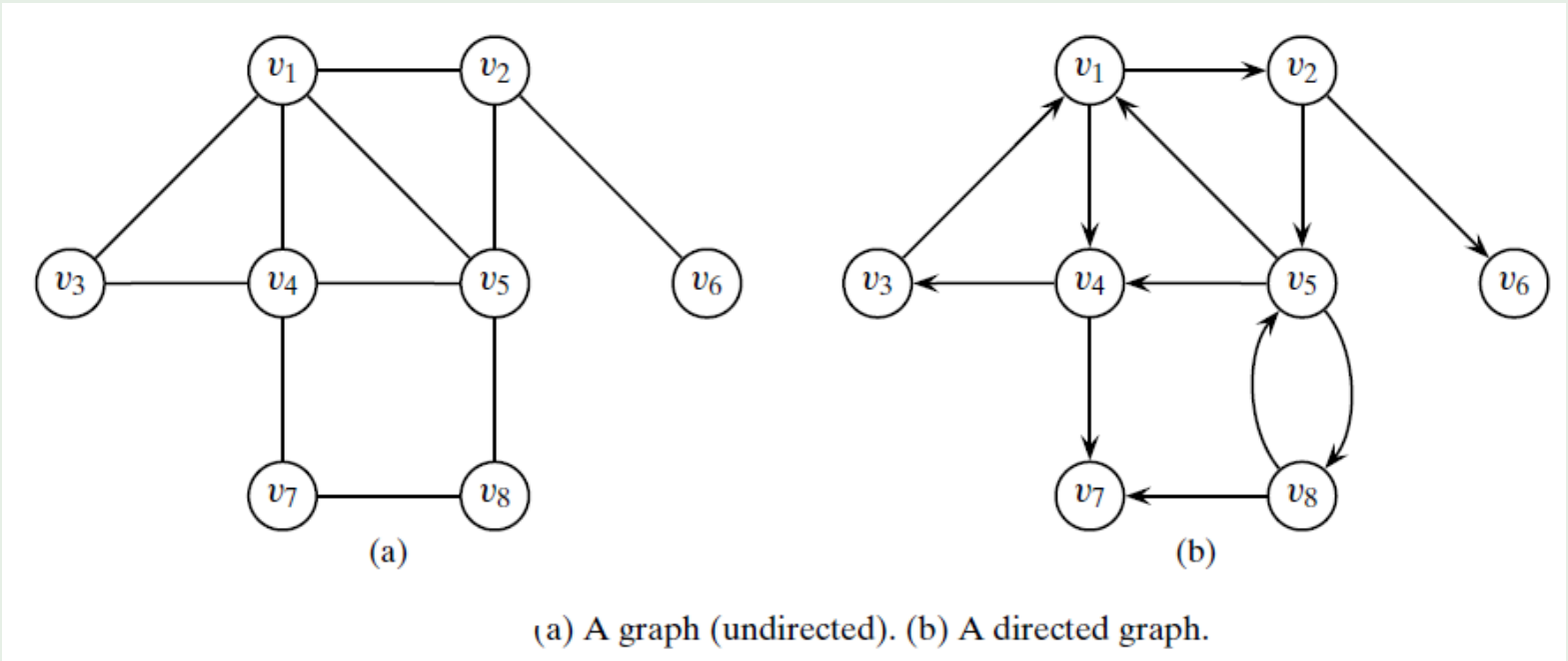
Two nodes v_i and v_j are said to be **connected** if there exists a path between them.
A graph is **connected** if there is a path between all pairs of vertices.

A **connected component**, or just component, of a graph is a maximal connected subgraph.

For a *directed* graph, we say that it is **strongly connected** if there is a (*directed*) path between all ordered pairs of vertices.

We say that it is **weakly connected** if there exists a path between node pairs only by considering edges as undirected.

Example



(a) A graph (undirected). (b) A directed graph.

(a) shows a graph with $|V| = 8$ vertices and $|E| = 11$ edges.
Because $(v_1, v_5) \in E$, we say that v_1 and v_5 are adjacent.
The degree of v_1 is $d(v_1) = d_1 = 4$.
The degree sequence of the graph is $(4, 4, 4, 3, 2, 2, 2, 1)$ and therefore its degree frequency distribution is given as
$(N_0, N_1, N_2, N_3, N_4) = (0, 1, 3, 1, 3)$

Example (cont.)

The degree distribution is given as

$$(f(0), f(1), f(2), f(3), f(4)) = (0, 0.125, 0.375, 0.125, 0.375)$$

The vertex sequence $(v_3, v_1, v_2, v_5, v_1, v_2, v_6)$ is a walk of length 6 between v_3 and v_6. We can see that vertices v_1 and v_2 have been visited more than once.

In contrast, the vertex sequence $(v_3, v_4, v_7, v_8, v_5, v_2, v_6)$ is a **path** of length 6 between v_3 and v_6.

However, this is not the shortest path between them, which happens to be (v_3, v_1, v_2, v_6) with length 3. Thus, the distance between them is given as $d(v_3, v_6) = 3$ (?).

Example (cont.)

The degree distribution is given as

$$(f(0), f(1), f(2), f(3), f(4)) = (0, 0.125, 0.375, 0.125, 0.375)$$

The vertex sequence $(v_3, v_1, v_2, v_5, v_1, v_2, v_6)$ is a walk of length 6 between v_3 and v_6. We can see that vertices v_1 and v_2 have been visited more than once.

In contrast, the vertex sequence $(v_3, v_4, v_7, v_8, v_5, v_2, v_6)$ is a **path** of length 6 between v_3 and v_6.

However, this is not the shortest path between them, which happens to be (v_3, v_1, v_2, v_6) with length 3. Thus, the distance between them is given as $d(v_3, v_6) = 3$ (?).

Example (cont.)

Figure (b) shows a **directed graph** with 8 vertices and 12 edges.
We can see that edge (v_5, v_8) is distinct from edge (v_8, v_5).
The *indegree* of v_7 is $id(v_7) = 2$, whereas its *outdegree* is $od(v_7) = 0$.
Thus, there is no (*directed*) path from v_7 to any other vertex.

Adjacency Matrix

A graph $G = (V, E)$, with $|V| = n$ vertices, can be conveniently represented in the form of an $n \times n$, symmetric binary **adjacency matrix**, A, defined as

$$A(i,j) = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

If the graph is directed, then the adjacency matrix A is not symmetric, as $(v_i, v_j) \in E$ obviously does not imply that $(v_j, v_i) \in E$.

Example (cont.)

Figure (b) shows a **directed graph** with 8 vertices and 12 edges.
We can see that edge (v_5, v_8) is distinct from edge (v_8, v_5).
The *indegree* of v_7 is $id(v_7) = 2$, whereas its *outdegree* is $od(v_7) = 0$.
Thus, there is no (*directed*) path from v_7 to any other vertex.

Adjacency Matrix

A graph $G = (V, E)$, with $|V| = n$ vertices, can be conveniently represented in the form of an $n \times n$, symmetric binary **adjacency matrix**, A, defined as

$$A(i,j) = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

If the graph is directed, then the adjacency matrix A is not symmetric, as $(v_i, v_j) \in E$ obviously does not imply that $(v_j, v_i) \in E$.

If the graph is *weighted*, then we obtain an $n \times n$ **weighted adjacency matrix**, A, defined as

$$A(i,j) = \begin{cases} w_{ij} & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

where w_{ij} is the weight on edge $(v_i, v_j) \in E$.

A weighted adjacency matrix can always be converted into a binary one, if desired, by using some threshold τ on the edge weights

$$A(i,j) = \begin{cases} 1 & \text{if } w_{ij} \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

If the graph is *weighted*, then we obtain an $n \times n$ **weighted adjacency matrix**, A, defined as

$$A(i,j) = \begin{cases} w_{ij} & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

where w_{ij} is the weight on edge $(v_i, v_j) \in E$.

A weighted adjacency matrix can always be converted into a binary one, if desired, by using some threshold τ on the edge weights

$$A(i,j) = \begin{cases} 1 & \text{if } w_{ij} \geq \tau \\ 0 & \text{otherwise} \end{cases}$$