University of Stavanger, DAT550

Summary of lecture notes

Håvard Godal
Spring 2021

Contents

1 Introduction to Data Mining

1.1 Data Mining

Data mining can be described as:

"Non-trivial extraction of implicit,
previously unknown and potential useful information from data"

"Exploration and analysis, by automatic or semi-automatic means,
of large quantities of data in order to discover meaningful patterns"

1.2 Machine Learning

Machine learning can be described as:

"The field of study that gives computers the ability to learn
without being explicitly programmed"

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

1.3 Data Mining or Machine Learning

Data mining is the process of discovering patterns in large data sets involving methods at the interseciton of machine learning, statistics, and database systems.

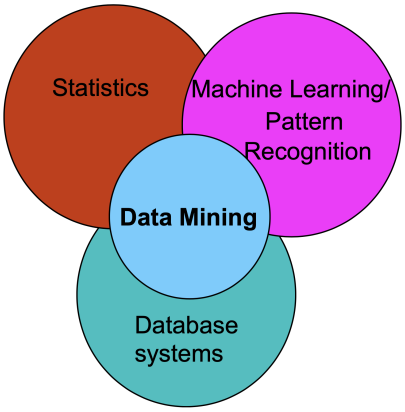


Figure 1.1: Diagram of Data Mining

1.4 Properties of Data Mining

1.4.1 Commercial viewpoint

Lots of data is being collected and warehoused, such as web data, purchases, and transactions. Computation power has become cheap and powerful, and the competitive pressure is strong. It is therefore necessary to provide better, customized servicees for an edge.

1.4.2 Scientific Viewpoint

Data is collected and stored at enormous speeds, up to multiple terrabytes per hour. Such data can be generated from the large hydron collider and from scanning the universe. Such data can not be interpreted by traditional techniques, and requires data mining to classify and segment the data, and form hypothesis formations.

1.4.3 KDD Process

Knowledge Discovery in Databases (KDD) refers to the overall process of discovering useful knowledge from data.

KDD is an integration of multiple technologies for data management such as database management and data warehousing, statistic machine learning, decision support, and others such as visualisation and parallel computing.

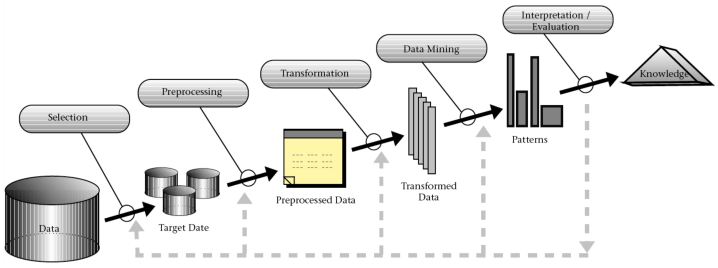


Figure 1.2: The KDD Process

1.5 Data Mining tasks

- Prediction methods
  - Use some variables to predict unknown or future values of other variables.
    - Supervised (classification or regression)
    - Unsupervised (clustering)
    - Semi-supervised
- Descriptive methods
  - Find human-interpretable patterns that describe the data.
    - Rule mining
    - Frequent patterns
    - Anomaly detection

1.6 Regression

1. Given a set of data points/instances along with "correct" answers or labels (training data)
2. Feed it to an algorithm which can learn a model and predict the correct value for an unseen data point (test data)
3. The learned model is an approximate representation of the training data

4. Predict continuous valued output (price in the previous example)
5. The algorithm should produce more right answers (goal)

1.7 Unsupervised Learning

- Unsupervised learning includes unlabeled data
- One way of doing this would be to cluster data into to groups
  - Group data points which are similar together, while separating dissimilar items as much as possible
  - This is a clustering algorithm

1.8 Association Rule Mining

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

Such rules can be used for marketing and sales promotion. Rule mining can also be used for inventory management, and much more.

1.9 Challenges in Data Mining

- Scalability
- Dimensionality
- Heterogenous or complex data
- Data ownership and distribution
- Privacy concern
- Data quality
- Evolving/streaming data

2 Data

2.1 Attributes

- An attribute is a property or characteristic of an object
  - Attribute are also known as variables, fields, characteristics, or features
- A collection of attributes describe an object
  - Objects are also known as records, points, cases, samples, entries, or instances

2.1.1 Values

Attribute values are numbers or symbols assigned to an attribute.
The distinction between attributes and attribute values are:

- Same attribute can be mapped to different attribute values
- Different attributes can be mapped to the same set of values

2.1.2 Types

Attribute Type	Operations	Transformations
Nomial	Mode, Entropy, Contingency, Correlation, χ^2-test	Any permutation of values
Ordinal	Median, Rank correlation, Percentiles, Run tests, Sign tests	Order-preserving change of values $new\_value = f(old\_value)$ where f is a monotonic function.
Interval	Mean, Standard deviation, t-tests Pearson's correlation, F-tests	$new\_value = a * old\_value + b$ where a and b are constants
Ratio	Geometric mean, Harmonic mean, Percent variation	$new\_value = a * old\_value$

2.1.3 Discrete and Continuous Attributes

Discrete attributes:

- Finite or countably infinite set of values
- Often integer or binaryvariables

Continuous attributes:

- Real numbers as attribute values
- Typically floating-point variables

2.2 Datasets

Some important characteristics of structured data are:

- Dimensionality
  - Curse of dimensionality
    - As the number of features or dimensions grow, the amount of data needed to generalize accurately grows exponentially.
    - When data moves from one dimensions to i.e. three dimensions, the given data fills less and less of the data space. In order to maintain an accurate representation of the space, the data for analysis grows exponentially.
    - When sorting or classifying data, low dimensional spaces tend to show the data as very similar, but in higher dimensions, the data might be further away from each other.
- Sparsity
  - Only presence counts
- Resolution
  - Patterns depend on the scale

2.2.1 Record Data

Data that consists of a collection of records, each of which consists of a fixed set of attributes.

Data Matrix

If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute.

Such data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute.

Document Data

Each document becomes a *term* vector. Each term is a component (attribute) of the vector.

The value of each component is the number of times the corresponding term occurs in the document.

Transaction Data

A special type of record data, where each record (transaction) involves a set of items.

For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

2.2.2 Graph Data

I.e. chemical structures.

2.2.3 Ordered Data

I.e. sequential data or gene sequences.

2.3 Data Quality

2.3.1 Noise

Noise refers to modification of original values.

2.3.2 Outliers

Outliers are data objects with characteristics that are considerably different than most of the other data objects in the data set.

2.3.3 Missing Values

Reasons for missing values may be that some information is not collected or aapplicable to all cases.

Handling missing values can be done by:

- Eliminating data objects
- Estimating the missing values
- Ignoring the missing values during analysis
- Replacing the missing values with all possible values weighted by their probabilities

2.3.4 Duplicate Data

Data set may include data objects that are duplicates, or almost duplicates of one another. This is a major issue when merging data from heterogenous sources.

2.4 Distance/Similarity Functions

- Similarity
  - Numerical measure of how alike two data objects are
  - Higher value means more alike
  - Often falls in the range $[0, 1]$
- Dissimilarity
  - Numerical measure of how different two data objects are
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies
- Proximity refers to a similarity or dissimilarity

2.4.1 Similarity/Dissimilarity for Simple Attributes

Given that p and q are the attribute values for two data objects:

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & if \quad p = q \\ 1 & if \quad p \neq q \end{cases}$	$s = \begin{cases} 1 & if \quad p = q \\ 0 & if \quad p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval/Ratio	$d = \lvert p - q \rvert$	$s = -d,\ s = \frac{1}{1+d},\ s = 1 - \frac{d-min(d)}{max(d)-min(d)}$

2.4.2 Euclidean distance

Euclidean distance is defined in the following theorem, where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the k_th attributes (components) or data objects p and q.

Theorem 2.1: Euclidean Distance

$$distance = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

Standarization is necessary if scales differ.

2.4.3 Minkowski Distance

Minkowski Distance is a generalization of Euclidean Distance where r is a parameter, n is the number of dimensions (attributes) and p_k and q_k are, respectively, the k_th attributes (components) or data objects p and q.

Theorem 2.2: Minkowski Distance

$$distance = \sqrt[r]{\sum_{k=1}^{n}|p_k - q_k|^r}$$

- $r = 1$: Manhattan distance
- $r = 2$: Manhattan distance
- $r \to \infty$: "Supremum" distance
  - This is the maximum difference between any component of the vectors

2.4.4 Properties of Distance Functions

Distances, such as the Euclidean distance, have some well known properties.

- Positive definiteness
  - $d(p, q) \geq 0$ for all p and q
  - $d(p, q) = 0$ only if $p = q$
- Symmetry
  - $d(p, q) = d(q, p)0$ for all p and q
- Triangle Inequality
  - $d(p, r) \leq d(p, q) + d(q, r)$ for all p. q and r

Important 2.3: Metric

A distance that satisfies the properties mentioned above is a metric.

2.4.5 Properties of Similarity Functions

Well known properties of similarities.

- Maximum Similarity
  - $s(p, q) = 1$ (or maximum similarity) only if $p = q$
- Symmetry
  - $s(p, q) = s(q, p)0$ for all p and q

2.5 Similarity/Coefficient

A common situation is that objects, p and q, have only binary attributes.
Let's define:

- M_{01} - Number of attributes where p is 0 and q is 1
- M_{10} - Number of attributes where p is 1 and q is 0
- M_{00} - Number of attributes where p is 0 and q is 0
- M_{11} - Number of attributes where p is 1 and q is 1

2.5.1 Simple Matching Coefficient

$$SMC = \frac{Numberofmatches}{Numberofattributes} \tag{2.1}$$

$$SMC = \frac{M_{00} + M_{11}}{M_{01} + M_{10} + M_{00} + M_{11}} \tag{2.2}$$

2.5.2 Jaccard Similarity/Coefficient

The Jaccard Similarity/Coefficient is used for categorical attributes and sets.

$$J = \frac{Numberof M_{11}}{Numberofnon - both - zeroattributes} \tag{2.3}$$

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{2.4}$$

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cup B|} \tag{2.5}$$

2.5.3 Cosine Similarity

If $\boldsymbol{A}$ and $\boldsymbol{B}$ are two documented vectors, then

$$similarity = \cos(\theta) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.6}$$

2.6 Dot Product

Given two vectors a and b:

$$\overrightarrow{a} \cdot \overrightarrow{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{2.7}$$

2.7 Data Preprocessing

2.7.1 Aggregation

Aggregation is to combine two or more attributes (or objects) into a single attribute (or object).

Purpose:

- Data reduction
- Reduce the number of attributes or objects
- Change of scale
- More "stable" data
- Often reduce variability

2.7.2 Data Sampling

Data sampling is the main technique employed for data selection. It is often used for both the preliminary investigation of the data and the final data analysis. Statisticians sample because obtaining the entire set of data of interest is too expensive or time consuming. Sampling is used in data mining because processing the entire set of data of interest is too expensive or time consuming.

A good stategy for choosing the sample size is to aim for a sample size that is 10% of the population, as long as the sample size is smaller than 1000. It is also important to have a large enough sample size so that all attributes/groups are represented.

Sampling is effective when:

- The sample is representative
- The sample has approximately the same properties (of interest) as the original set of data

Types of sampling:

- Simple random sampling
  - Equal probability of selecting any item.

- Sampling without replacement
  - As each item is selected, it is removed from the population.
- Sampling with replacement
  - Objects are not removed from the population as they are selected for the sample.
- Stratified sampling
  - Split the data into several partitions, then draw random samples from each partition.

Reservoir Sampling

Theorem 2.4: Reservoir Sampling

Keep a reservoir of r samples

1. Keep the first r items in memory
2. When the i^{th} item arrives ($i > r$)
   a) Keep the new item with probability $\frac{r}{i}$,
   or discard the new item with probability $1 - \frac{r}{i}$
   b) Discard one of the items in the reservoir at random
   if the new item was kept.

This means that as i increases, the probability of a new item being kept in the reservoir reduces.

2.8 Dimensionality Reduction

- Purpose
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data mining
  - Allow data to be more easily visualized
  - May help eliminate irrelevant features or reduce noise
- Techniques
  - Pronciple Component analysis
  - Singular Value Decomposition

3 Exploring Data

3.1 Summary Statistics

- Frequency
- Mode
- Percentiles
- Mean
- Median
- Range
- Variance

Theorem 3.1: Mean

$$mean(x) = \bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

Theorem 3.2: Median

$$median(x) = \begin{cases} x_{(r+1)} & \text{if } m = 2r+1 \\ \frac{1}{2}\left(x_{(r)} + x_{(r+1)}\right) & \text{if } m = 2r \end{cases}$$

Theorem 3.3: Variance

$$variance(x) = s_x^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2$$

Theorem 3.4: Average Absolute Deviation

$$AAD(x) = \frac{1}{m}\sum_{i=1}^{m}|x_i - \bar{x}|$$

Theorem 3.5: Mean Absolute Deviation

$$MAD(x) = median\left(\{|x_1 - \bar{x}| \cdots |x_m - \bar{x}|\}\right)$$

3.2 Visualisation

Visualisation is the conversion of data into a visual or tabular format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported. Visualisation allows humans to detect general patterns and trends, as well as detect outliers and unusual patterns.

3.2.1 Representation

The representation is the mapping of information to a visual format. Data objects, their attributes, and the relationships among data objects are translated into graphichal elements such as points, lines, shapes, and colors.

3.2.2 Arrangement

Arrangement is the placement of visual elements within a display. This can make a large difference in how easy it is to understand the data.

3.2.3 Selection

Selection is the elimination or the de-emphasis of certain objects and attributes. Such selections may involve choosing a subset of attributes, or choosing a subset of objects. Only selecting some attributes can be done through dimensionality reduction, while selecting some objects can be done through stratified sampling. Some things to keep in mind when using data selection is to contain the diversity of the objects.

3.2.4 Visualisation Techniques

- Histogram plot
- Box plot
- Scatter plot

- Matrix plot
- Parallel coordinates plot
- Star plot

4 Decision Trees

4.1 Classification Definition

Given a collection of records (a training set) where each record contains a set of attributes, the classification has to find a model for one particular attribute (the class) as a function of the values of other attributes. The goal is for previously unseen records to have the class attribute assigned as accurately as possible. A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with the training set, used to build the model, and the test set used to validate it.

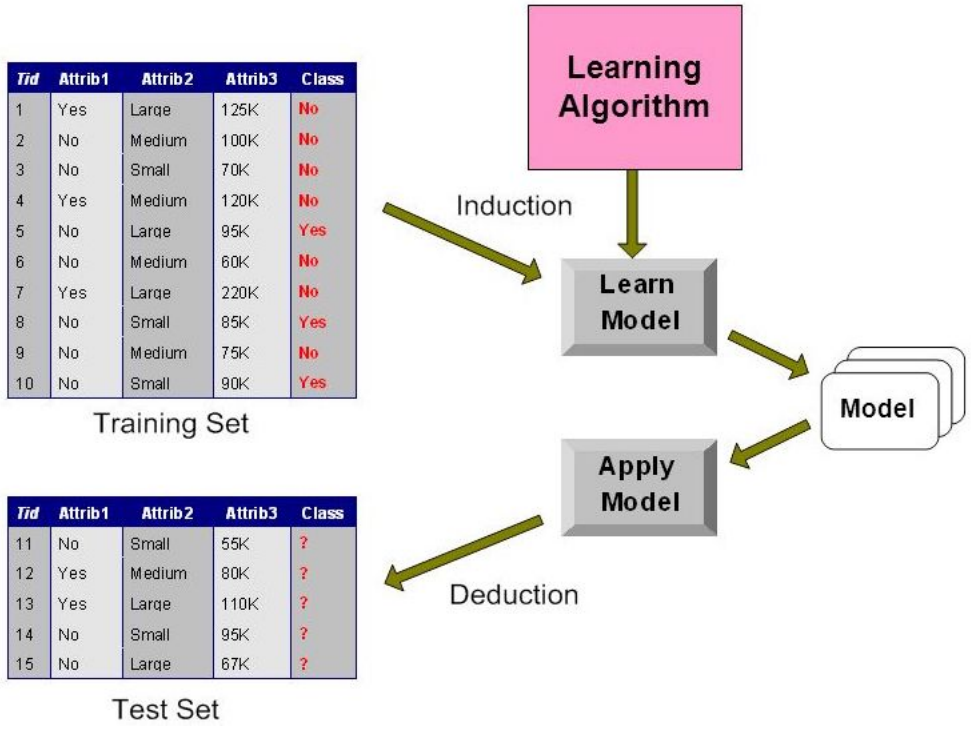


Figure 4.1: Typical Classification Task

4.1.1 Decision Tree

A tree is built by splitting the source set, constituting the root node of the tree, into subsets, which constitute the successor children. The splitting is based on a set of splitting

rules based on classification features. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.

Decision trees used in data mining are of two main types:

- Classification tree analysis - When the predicted outcome is the class (discrete) to which the data belongs.
- Regression tree analysis - When the predicted outcome can be considered a real number.

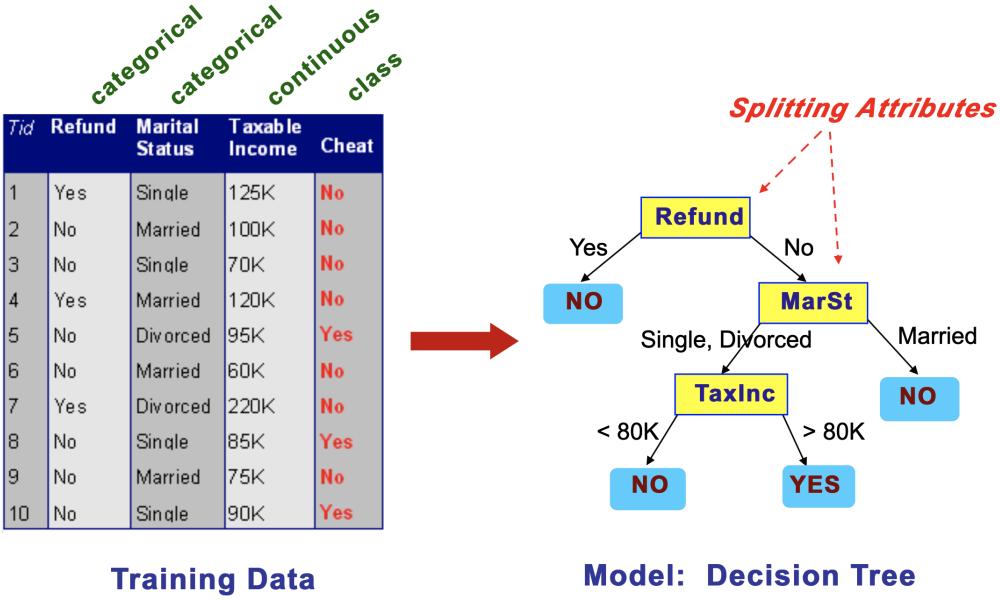


Figure 4.2: Typical Decision Tree

4.2 Hunt's Algorithm

- Let D_t be the set of training records that reach a node t.
- General procedure:
  - If D_t contains records that belong to the same class as y_t, then t is a leaf node labeled as y_t.
    - This is a pure node.
  - If D_t is an empty set, then t is a leaf node labeled by the default class y_d.
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.
    - This is an impure node.

4.3 Building Trees

4.3.1 Tree Induction

The greedy strategy splits the records based on an attribute test that optimizes a certain criterion.

Issues with this approach include:

- How to split the records. How should the attribute test condition be specified, and how is the best split determined?
- Determining when to stop.

Specifying the Attribute Test Condition

A proper test condition depends on the attribute type, which can be nominal, ordinal, or continuous.

The condition also depends on the number of ways to split. This can be either a two-way split (binary) or a multi-way split. The binary split is often preferred as of the bushiness of the multi-way split. This means that very few records will match the leaf-nodes, and will therefore be overfitted.

Ordinal attributes can be split the same way as nominal attributes, with some limitations if the order is to be preserved.

For continuous attributes, there are two ways of handling splitting:

- Discretization to form an ordinal catergorical attribute.
  - Static - discretize once at the beginning.
  - Dynamic - ranges can be found by equal interval bucketing, equal frequency bucketing, or clustering.

- Binary decision.
  - Consider all possible splits and find the best cut.
  - Can be more compute intensive.

Determine Best Split

When it comes to greedy approaches, nodes with a homogeneous class distribution are preffered.

Figure 4.3: Class distributions

4.4 Measures of Node Impurity

Finding the best split follows closely to this process:

1. Define the set of records as $M0$, where some records belong to class $C0$, while the rest belongs to $C1$.
2. Define some splitting criterias A and B.
3. Use each criteria to generate new sets of records. The new sets of records now has a different (or equal) distribution of $C0$- and $C1$-records.
   - $M1$ and $M2$ from A, collected into $M12$
   - $M3$ and $M4$ from B, collected into $M34$
4. The gain in purity can be measured as such: $Gain = M0 - M12$ vs. $M0 - M34$
5. The greedy algorithm can then choose the attribute that gives the greatest gain in purity.

4.4.1 Gini

The Gini Index measures the degree of probability of a particular variable being wrongly classified when it is randomly chosen.

Theorem 4.1: Gini Index

$$GINI(t) = 1 - \sum_{j}^{n_c} [p(j|t)]^2$$

Maximum: $(1 - \frac{1}{n_c})$ Minimum: 0.0 (pure)

Note: $p(j|t)$ is the relative frequency of class j at node t.

Gini can be used when splitting nodes. When a node p is split into k partitions (children), the quality of split follows the theorem:

Theorem 4.2: Gini Split

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

n_i = Number of records at child i.
n = Number of records at node p.

Compute the Gini Index for continuous attributes can often be quite expensive as all values have to be evaluated for splitting. One process (to be done for each attribute) for finding a good split is as such:

1. Sort the attribute on values.
2. Linearly scans the values, each time choosing the mean of the current value and the next value as the splitting point position and compute the Gini Index.
3. Choose the split position that has the least Gini Index.

Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No	
Sorted Values →	Annual Income																			
	60		70		75		85		90		95		100		120		125		220	
Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400		0.420	

Figure 4.4: Gini Index of Continuous Attributes

4.4.2 Entropy

Entropy is a measure of the uncertainty associated with a random variable. Entropy is a splitting criterion based on information gain and measures the homogeneity of a node.

Theorem 4.3: Entropy

$$Entropy(t) = -\sum_{j}^{n_c} p(j|t) \log_2 \ p(j|t)$$

Maximum: $(\log_2 \ n_c)$ Minimum: 0.0

Note: $p(j|t)$ is the relative frequency of class j at node t.

In the same way Gini can be used to split nodes, so can Entropy:

Theorem 4.4: Entropy Split

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent node p is split into k partitions.
n_i is the number of records in partition i

The gain measures reduction in entropy achieved from the split. Choose the split that achieves the most reduction, that is, maximizes gain. The disadvantage is that Entropy tends to prefer splits that result in a large number of partitions, each being small but pure (overfitting). This is fixed by including the number of partitions in the equation.

Theorem 4.5: Entropy Split Ratio

$$GainRation_{split} = \frac{Gain_{split}}{SplitInfo}$$

$$SplitInfo = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent node p is split into k partitions.
n_i is the number of records in partition i

4.4.3 Misclassification error

The Classification error measures the misclassification error made by a node.
Classification error at node t:

Theorem 4.6: Classification Error

$$Error(t) = 1 - \max P(i|t)$$

Maximum: $(1 - \frac{1}{n_c})$ Minimum: 0.0

4.4.4 Comparison among Splitting Criterias

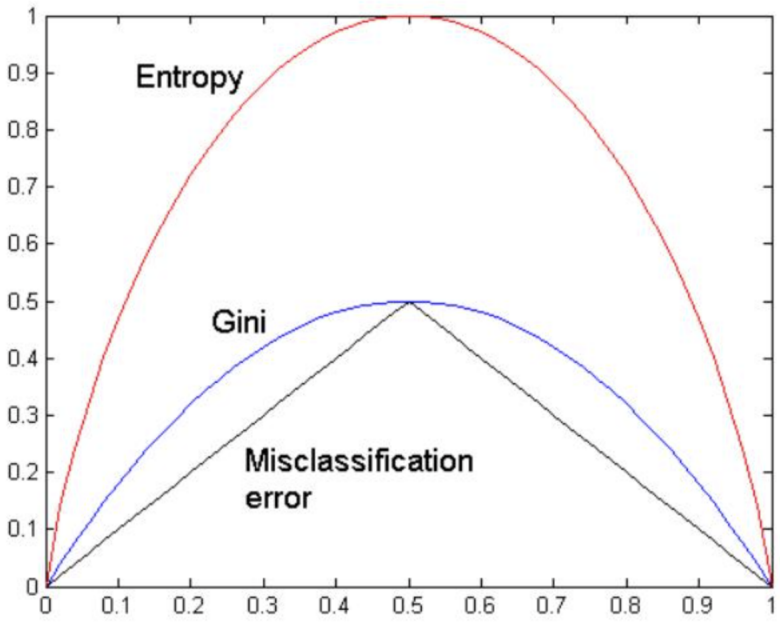


Figure 4.5: Splitting Criterias for a two-class problem

4.5 Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class.
- Stop expanding a node when all the records have similar attribute values.
- Early termination (probably for limiting tree depth).

4.6 Advantages of Decision Tree based Classification

- Inexpensive to construct.
- Extremely fast at classifying unknown records.
- Easy to interpret for small-sized trees.
- Accuracy is comparable to other classification techniques for many simple data sets.

4.7 Classification Issues

4.7.1 Underfitting and Overfitting

- Underfitting: When a model cannot capture the underlying trend of the data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. Specifically, underfitting occurs if the model or algorithm shows low variance but high bias. Underfitting is often a result of an excessively simple model. Overfitting can also arise from insufficient data points.
- Overfitting: When a model captures the noise of the data. Intuitively, overfitting occurs when the model or the algorithm fits the data too well. Specifically, overfitting occurs if the model or algorithm shows low bias but high variance. Overfitting is often a result of an excessively complicated model, and it can be prevented by fitting multiple models and using validation or cross-validation to compare their predictive accuracies on test data.

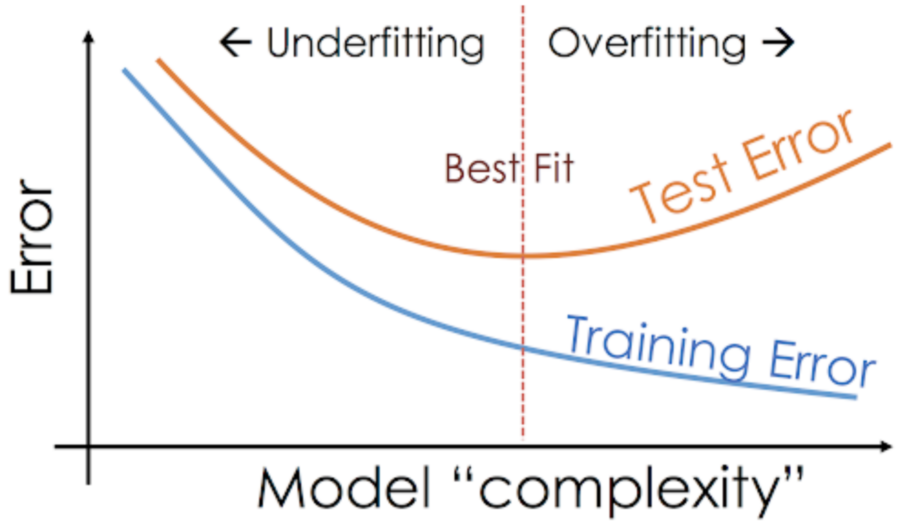


Figure 4.6: Overfitting and underfitting

4.7.2 Address Overfitting

Decision Tree Pruning

Pre-Pruning (Early Stopping Rule):

- Stop the algorithm before it becomes a fully-grown tree.
  - Stop if all instances belong to the same class.
  - Stop if all the attribute values are the same.
  - Stop if the number of records in a node are lower than the user-specified threshold.
  - Stop if splitting a node does not improve the impurity meassure.

Post-Pruning:

1. Grow decision tree to its entirety.
2. Trim the nodes of the decision tree in a bottom-up fashion.
3. If generalization error improves after trimming, replace sub-tree by a leaf node.
4. The class label of a leaf node is determined from the majority class of instances in the sub-tree.

Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.
- For complex models, there is a greater chance that it was fitted accidentally by the error in data.
- Therefore, one should include model complexity when evaluating a model.

4.7.3 Generalization Errors

Generalization error is a measure of how accurately a model can predict outcome values for previously unseen data. The measure is calculated from the ratio of misclassifications that can occur when choosing the majority class for all records in the node.

- Training errors: Error on training ($\sum_{t=1}^{n} e(t)$)
- Generalization errors: Error on validation ($\sum_{t=1}^{n} e'(t)$)

Methods for estimating generalization errors:

- Optimistic approach: $e'(t) = e(t)$

- Pessimistic approach:
  - For leaf each leaf node, $e'(t_i) = e(t_i) + \frac{1}{2}$
  - Total error: $e'(T) = e(T) + N\frac{1}{2}$ (N = number of leaf nodes)
- Reduced error pruning (REP)
  - Uses validation data set to estimate generalization error.

4.7.4 Handling Missing Attribute Values

Missing attribute values affect the decision tree in three major ways:

- How impurity measures are computed.
- How to distribute instance with missing value to child nodes.
- How a test instance with missing value is classified.

Computing Impurity Measure (Impurity Gain)

One way of computing the information gain is as normal, without including the instances with the missing attribute. This means that the entropy-computation for the parent is as normal, but for the children, instances with the missing attributes should be omitted, but included in the denominator. Finally, the gain is computed as normal but is multiplied by the fraction of "valid" records.

Distributing Instances

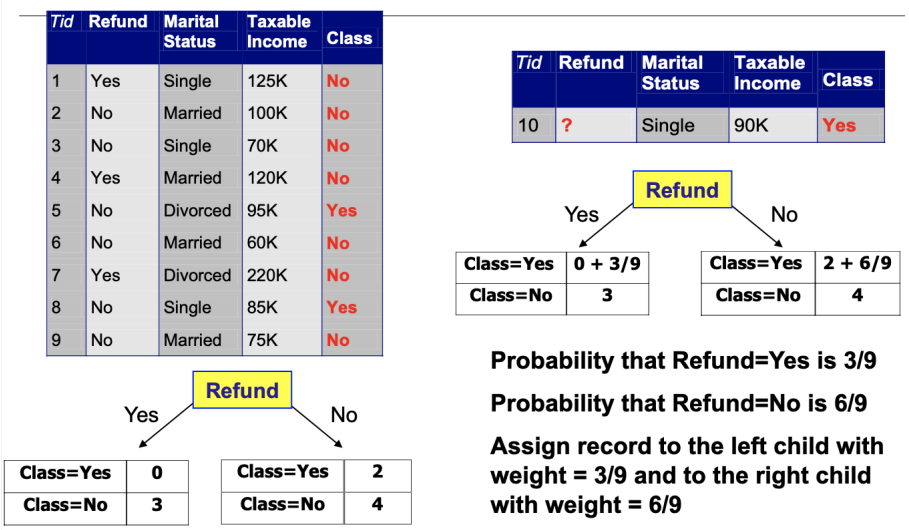


Figure 4.7: Distributing Instances

The figure explains that since the instance with the unknown attribute corresponds to the class yes, it is distributed on both nodes, affecting the yes-class in each node. The distribution ratio is based on the number of instances of the same class, divided by the total number of instances.

Classify Instances

To classify an instance with an unknown class, as well as having the current attribute unknown, can be done by understanding what option is more likely. If most of the instances are of attribute 1 the instance with the unknown attribute is assumed to also have attribute 1 and follows the tree accordingly.

4.7.5 Other Issues

Data Fragmentation

- The number of instances gets smaller as you traverse down the tree.
- Number of instances at the leaf node could be too small to make any statistically significant decision.

Search Strategy

- Finding an optimal decision tree is NP-hard (exponentially).
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution.
- Could use bottom-up or bi-directional algorithms.

Expressiveness

Decision trees provide expressive representation for learning discrete-valued functions, but they do not generalize well to certain types of boolean functions like parity functions.

Also, decision trees are not expressive enough for modeling continuous variables.

Decision Boundary

The borderline between two neighboring regions of different classes is known as a decision boundary.

A decision boundary is parallel to the axes because the test conditions only involve a single attribute at a time.

This could be fixed with oblique decision trees, but includes more expressive representation and can be computationally expensive.

Tree Replication

The same subtree appears in multiple branches.

5 Model Evaluation

5.1 Metrics for Performance Evaluation

Metrics for performance evaluation focuses on the predictive capability of a model, that is, how well a model can predict the class of the test data, and evaluate the performance of the training-data. This focus is instead of focusing on how fast it takes to classify or build models, scalability, etc.

5.1.1 Confusion Matrix

This can be done through a confusion matrix:

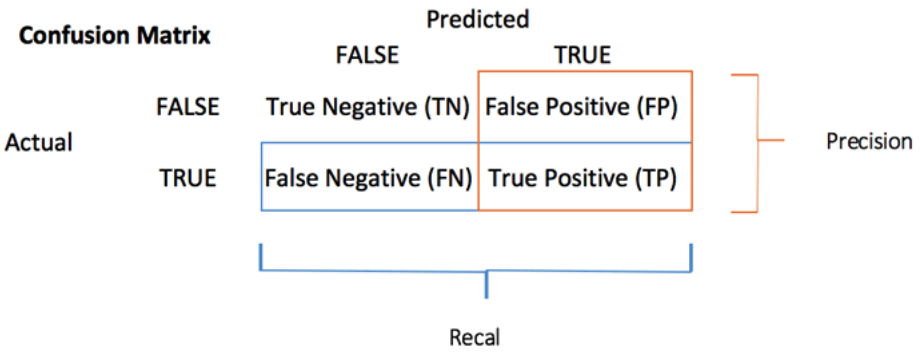


Figure 5.1: Confusion Matrix

One of the most common and widely-used metric for evaluating the performance is accuracy:

Theorem 5.1

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

The accuracy metric has some limitations, among those are the ability to evaluate two-class problems where almost all instances are of class 0, while only a few are of class 1. The accuracy metric will result in 99.9% accuracy, which is misleading if the model predicts all instances to belong to class 0.

5.1.2 Cost Matrix

One way of solving the issue is to use a cost matrix instead:

		PREDICTED CLASS	
	C(i|j)	**Class=Yes**	**Class=No**
ACTUAL CLASS	**Class=Yes**	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

Figure 5.2: Cost Matrix

Here, $C(i|j)$ is the cost of misclassifying class j as class i. The cost equals the weighted sum of all predictions.

If true positives and true negatives are equally weighted, as well as the fasle positives and the false negatives, the accuracy is proportional to the cost. This results in the following equation:

$$Cost = N[q - (q - p) * Accuracy] \tag{5.1}$$

Where $N = TP + TN + FP + FN$, $q = FP$, and $p = TP$.

5.1.3 Cost-Sensitive Measures

Some cost-sensitive measures that are preferred over accuracy:

$$Precision(p) = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall(r) = \frac{TP}{TP + FN} \tag{5.3}$$

$$F - measure(F) = \frac{2TP}{2TP + FP + FN} \tag{5.4}$$

- Precision is biased towards TP and FP
- Recall is biased towards TP and FN
- F-measure is biased towards all except TN

$$Weighted_Accuracy = \frac{\omega_1 TP + \omega_4 TN}{\omega_1 TP + \omega_2 FN + \omega_3 FP + \omega_2 TN} \tag{5.5}$$

5.2 Methods for Performance Evaluation

Methods on how to obtain a reliable estimate of performance.

The performance of a model may depend on other factors besides the learning algorithm:

- Class distribution.
- Cost of misclassification.
- Size of training and test sets.

5.2.1 Learning Curve

Learning curves show how accuracy changes with varying sample sizes.

Learning curves require a sampling schedule:

- Arithmetic sampling. (Linear)
- Geometric sampling. (Exponential)

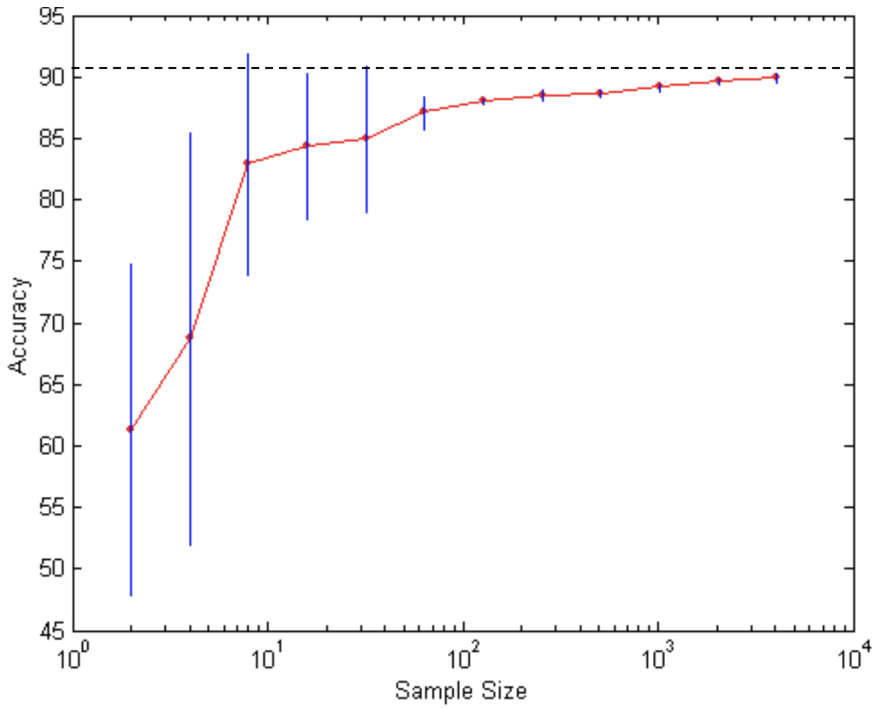


Figure 5.3: A Learning Curve

The effects of small sample sizes includes:

- Bias in the estimate (Underfitting)
- Variance of estimate (Overfitting)

5.2.2 Other Methods of Estimation

- Holdout
  - Reserve 80% for training and 20% for testing.
- Random subsampling
  - Repeated holdouts
- Cross validation (popular)
  - Partition data into k disjoint subsets.
  - k-fold: Train on $k-1$ partitions, test on the remaining one.
  - Leave-one-out: $k = n$
- Stratified sampling
  - Oversampling
  - Undersampling
- Bootstrap
  - Sampling with replacement

5.3 Methods for Model Comparison

Methods for model comparison explains how to compare the relative performance among competing models.

5.3.1 Receiver Operating Characteristic (ROC)

ROC was developed in the 1950s to characterize the trade-off between true positives and false positives.

A ROC curve plots TP on the y-axis against FP on the x-axis, where the performance of each classifier is represented as a point on the ROC curve. The point changes based on the threshold of the algorithm, the sample distribution, and/or the cost matrix.

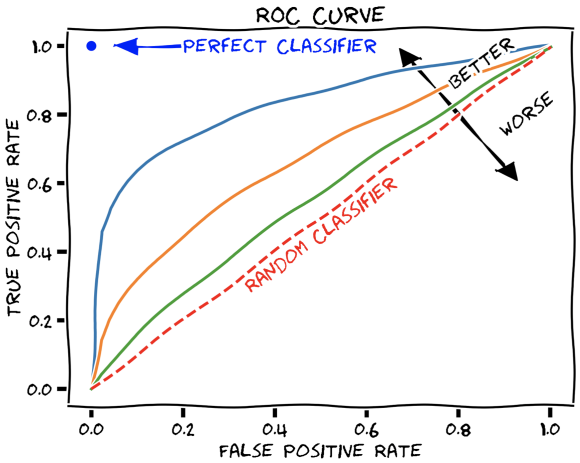


Figure 5.4: ROC Curve

In some cases when comparing the ROC curve for different models, no models consistently outperform the others. Evaluating such models can be based on the area under the ROC curve (AUC), or which model fits better to the desired outcome.

Constructing a ROC curve

1. Use a classifier that produces posterior probability for each test instance $P(+|A)$.
2. Sor the instances according to $P(+|A)$ in decending order.
3. Apply threshold at each unique value of $P(+|A)$.
4. Count the number of TP, FP, TN and FN at each treshold.

- $TPR = \frac{TP}{TP+FN}$
- $FPR = \frac{FP}{FP+TN}$

6 Regression Trees and Ensemble methods

6.1 Regression Trees

In a decision tree, there is typically a data set with N features $(X_1, X_2, \cdots, X_N)$. Each of the features has a domain, with the domain type being dependent on the type of data the feature contains (categorical or numerical). The output variable Y will have a domain D_y.

- Categorical - Classification
- Numerical - Regression

6.1.1 Constructing a Regression Tree

1. Find a split (X_i, v) that creates two child nodes D_L and D_R from the parent node D. The split should maximize the weighted sum of the variances.
   - For ordered domains, sort X_i and consider a split between each pair of adjacent values.
   - For categorical X_i find the best split based on subsets (Breiman's algorithm).

$$Weighted\_Variance = |D| * Var(D) - (|D_L| * Var(D_L) + |D_R| * Var(D_R)) \quad (6.1)$$

$$Var(D) = \frac{1}{n} \sum_{i=1}^{|D|} (y_i - \overline{y})^2 \quad (6.2)$$

Stopping a Tree

- When the leaf is pure, i.e. $Var(D) < \epsilon$
- When the number of instances in the leaf ($|D|$) is too small.

Predictor

- Regression - Average value of y_i of the instances in the leaf.
- Classification - Choose the most common y_i in the leaf.

6.2 Ensemble Methods

Ensemble methods construct a set of classifiers from a given training data. The classification is done by predicting class values from previously unseen records by aggregating predictions made by multiple classifiers.

The general idea of ensemble methods is to have the original training data and then:

1. Create multiple data sets from the training data.
2. Build multiple classifiers corresponding to the data sets.
3. Combine the predictions of the classifiers.

Ensemble methods work the sum of the error rate of multiple independent classifiers are lower than the error rate itself. The following equation calculates the probability that the ensemble classifier makes a wrong prediction, having ϵ as the error rate.

$$\sum_{i=N/2(+1)}^{N} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \tag{6.3}$$

6.3 Manipulating the Training set

Bagging

Bagging is sampling with replacement

1. Let k be the number of bootstrap samples.
2. `for` i `in range(`k`):`
   a) Create a bootstrap sample of size N, D_i.
   b) Train a base classifier C_i on the bootstrap sample D_i.

$C^*(x) = argmax\left(\sum_i \delta\left(C_i(x) = y\right)\right)$

$\delta = 1$ if the argument is true and 0 otherwise (the majority class is chosen).
Bagging can also increase the complexity of simple classifiers such as decision stumps.

Boosting

Boosting is an iterative procedure to adaptively change the distribution of training data by focusing more on previously misclassified records. Initially, all N records are assigned equal weights.

Unlike bagging, the weights of the records may change at the end of each boosting round.

- Records that are wrongly classified will have their weights increased.
- Records that are classified correctly will have their weights decreased.

One way of boosting is to use the AdaBoost algorithm:

Algorithm 5.7 AdaBoost algorithm.

1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Initialize the weights for all N examples.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all examples in the original training set, D.
7: $\epsilon_i = \frac{1}{N}\left[\sum_j w_j \, \delta\big(C_i(x_j) \neq y_j\big)\right]$ {Calculate the weighted error.}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Reset the weights for all N examples.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each example according to Equation 5.69.
14: **end for**
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.

Figure 6.1: AdaBoost Algorithm

6.4 Manipulating the Features

Random Forest

A random forest is a class of ensemble methods specifically designed for decision tree classifiers. It combines the predictions made by multiple decision trees where each tree is generated based on the values of an independent sample set. The samples are generated from a fixed probability distribution.

As in bagging, we build decision trees on bootstrapped training samples. Each time a split in a tree is considered, a random sample of F features is chosen as split candidates from the full set of m features.
(If $F = m$, we have bagging)

A random forest algorithm can look like this:

- `for` i `in range(`B`):`

1. Draw a bootstrap sample D^* of size N from the training data.
   a) Grow a random forest tree to the bootstrapped data.
      i. Select $\boldsymbol{F}$ features at random from the m variables.
      ii. Pick the best variable/split point among the $\boldsymbol{F}$s.
      iii. Split the node into two child nodes.

- Output the ensemble of trees.

Prediction at a new point x can be done:

- For regression - Average of the results.
- For classification - Majority vote.

Random Forest Recommendations

- For classification, the default value for F is $\sqrt{m}$ and the minimum node size is one.
- For regression, the default value for F is $\frac{m}{3}$ and the minimum node size is five.

7 Locality-Sensitive Hashing

7.1 Definition

Locality-sensitive hashing (LSH) is an algorithmic technique that hashes similar input items into the same "buckets" with high probability. Since similar items end up in the same buckets, this technique can be used for data clustering and nearest neighbor search. It differs from conventional hashing techniques in that hash collisions are maximized, not minimized. Alternatively, the technique can be seen as a way to reduce the dimensionality of high-dimensional data; high-dimensional input items can be reduced to low-dimensional versions while preserving relative distances between items.

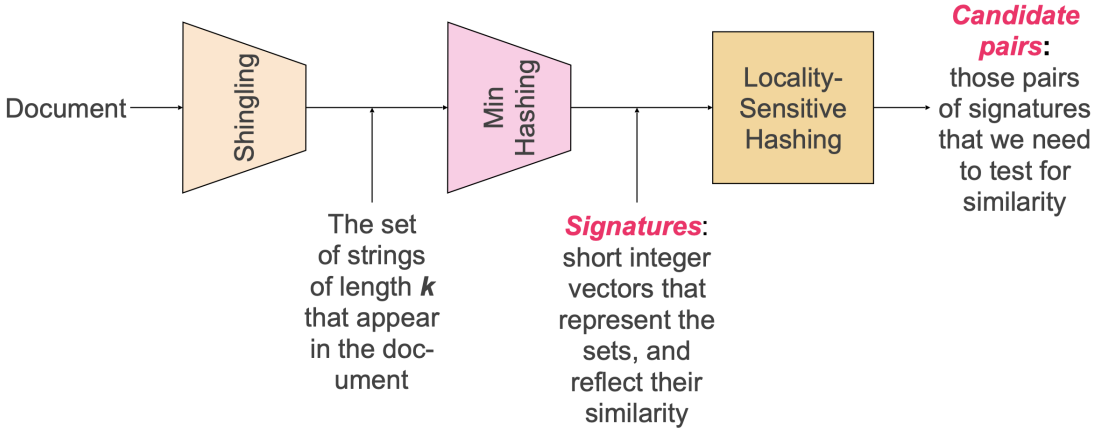


Figure 7.1: Locality-Sensitive Hashing

7.2 LSH Steps

1. Shingling: Convert documents into sets of shingles.
2. Min-Hashing: Convert large sets of short signatures, while preserving similarity (i.e. Jaccard similarity).
3. Locality-Sensitive Hashing: Focus on pairs of signatures likely to be from similar documents.

- This results in candidate pairs, which are likely candidates with a high probability of being similar to the input document.

7.3 Shingling

Shingling is the process of converting documents to sets. Some simple approaches are to transform the document into sets of unique words, or "important" words. The main problem is that the ordering of words has to be taken into account:

7.3.1 Shingles

A k-shingle (or k-gram) for a document is a sequence of k tokens that appears in the document.

- Tokens can be characters, words, or something else, depending on the application.

A modification of the k-shingle is to use shingles as a bag (multiset), and therefore counting equal sequences multiple times instead of only one.

Similarity Metric for Shingles

- Document D_i is a set of its k-shingles $C_i = S(D_i)$
- In this setting, each unique shingle is a dimension.
- The vectors will be very sparse as most documents does not contain most of the shingles.

A natural similarity measure is the **Jaccard Similarity**:

$$sim(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \tag{7.1}$$

Working Assumptions

- Documents that have lots of shingles in common have similar text, even if the text appears in different orders.
- $k = 5$ is ok for short documents.
- $k = 10$ is better for long documents.

7.4 Min-Hashing

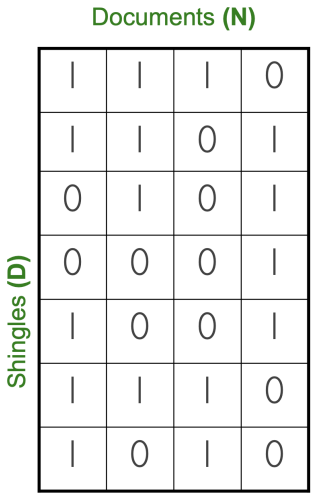
Using pairwise Jaccard similarity directly to find near-duplicate documents would result in operations too complex and time consuming to handle. This is why min-hashing is used.

Min-hashing provides signatures (short integer vectors) that represent the sets and their similarity.

Encoding Sets as Bit Vectors

Many similarity problems can be formalized as:
Finding subsets that have a significant intersection.

This can be done efficiently by encoding sets using boolean vectors.



Note: $d(C_1, C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} = 1 - \frac{3}{6}$

Figure 7.2: Boolean Matrix

7.4.1 Hashing Columns

The key idea of hashing is to hash each column C to a small signature $h(C)$ so that:

- $h(C)$ is small enough to fit in the RAM.
- The Jaccard similarity is preserved.

The goal is to find a hash function so that if $sim(C_1, C_2)$ is high, there is a high probability that $h(C_1) = h(C_2)$, and vice versa. The hash function places columns into a bucket, with the expected outcome being that most pairs of near-duplicate documents hash into the same bucket.

A suitable hash function for the Jaccard similarity is called **Min-Hashing**.

7.4.2 Definition

- Imagine the rows of the boolean matrix permuted under random permutation π.
- Define a hash function $h_\pi(C)$ = the index of the first row in which column C is $True$.

- Use several independent hash functions (that is, permutations) to create a signature of a column.

$$\boldsymbol{h_\pi(C) = min_\pi \pi(C)} \tag{7.2}$$

Each unique combination of 1s and 0s can be classified as different types.
I.e. $A = 1,1$, $B = 1,0$, $C = 0,1$, $D = 0,0$. If lower case letters denote the number of rows corresponding to the corresponding type, we get the following Jaccard similarity:

$$sim(C_1, C_2) = \frac{a}{a+b+c} \tag{7.3}$$

And the probability:

$$Pr\left[h_\pi(C_1) = h_\pi(C_2)\right] = sim(C_1, C_2) \tag{7.4}$$

And the thought-process is:

1. Look down the columns C_1 and C_2 until the first 1 appears.
2. If the row is of type A, then $h(C_1) = h(C_2)$.
3. If the row is of type B or C, then $h(C_1) \neq h(C_2)$.

Similarity for Signatures

- The similarity of two signatures is the fraction of the hash functions in which they agree.
- Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures.

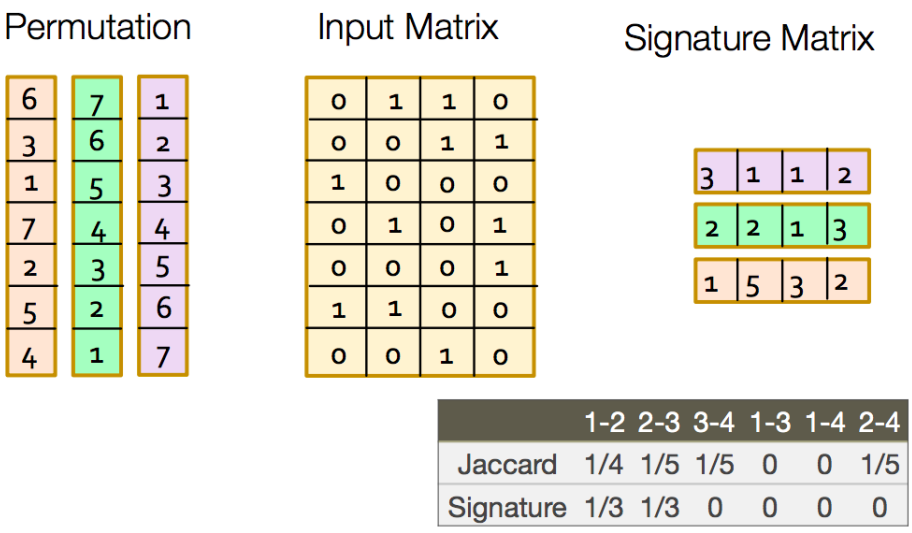


Figure 7.3: Min-Hashing

7.4.3 Process

1. Pick $k = 100$ random permutations of the rows.
2. Think of $\boldsymbol{sig(C)}$ as a column vector.
3. $\boldsymbol{sig(C)[i]}$ = The index of the first row that has the value of 1 in column C, given the $\boldsymbol{i}_{th}$ permutation.
4. $\boldsymbol{sig(C)[i] = min\,(\pi_i(C))}$

7.5 LSH

The goal of the locality-sensitive hashing process is to find documents with Jaccard similarity over some similarity threshold s.

The general idea is to use a function $f(x, y)$ that tells whether x and y is a candidate pair; a pair of elements whose similarity must be evaluated.

For Min-Hash matrices:

- Hash columns of the signature matrix M into many buckets,
- Each pair of documents that hashes into the same bucket is a candidate pair.

Columns x and y of M are a candidate pair if their signatures agree on at leas the fraction s of their rows. $\boldsymbol{M}(i, x) = \boldsymbol{M}(i, y)$ for at least the fraction s of i. It is expected that the documents x and y have the same Jaccard similarity as their signatures.

The main idea of LSH for Min-hashing is to hash columns of the signature matrix M several times. Then, arrange the columns so that only similar columns are likely to hash to the same bucket, with a high probability. Finally, the candidate pairs are those columns that hash to the same bucket.

7.5.1 Process of LSH

1. Divide matrix M into b bands of r rows.
2. For each band, hash its portion of each column to a hash table with k buckets, while making k as large as possible.
3. The candidate column pairs are those bands that hash to the same bucket for more than one band.
4. Tune b and r to catch most similar pairs, and minimize the number of non-similar pairs.

Simplified Assumptions

- There are enough buckets that columns are unlikely to hash to the same bucket unless they are identical in a particular band.
- It is assumed that all bands in the bucket are identical.
- These assumptions are only needed to simplify the analysis, not for the correctness of the algorithm.

Some Probabilities

Given the columns C_1 and C_2 with the similarity s, and that b bands with r rows is selected:

- The probability that all rows in the band are equal $= s^r$
- The probability that some rows in the band are unequal $= 1 - s^r$
- The probability that no bands are identical after b bands $= (1 - s^r)^b$
- The probability that at least one band is identical after b bands $= 1 - (1 - s^r)^b$

7.5.2 LSH Decisions

Some parameters have to be adjusted to balance the false positives and negatives and get the most desired result when using LSH. This includes:

- The number of Min-Hashes (rows of M).
- The number of bands b.
- The number of rows r per band.

7.5.3 Summary

- Tune M, b, and r to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.
- Check in the main memory that candidate pairs do have similar signatures.
- Optional: In another pass through the data, check that the remaining candidate pairs represent similar documents.

8 Dimensionality reduction

8.1 Dimensionality Reduction

The purpose of dimensionality reduction is to:

- Avoid curse of dimensionality.
- Reduce the amount of time and memory required by data mining algorithms.
- Allow data to be more easily visualized.
- May help to eliminate irrelevant features or reduce noise.
- Recommender systems

Some of the techniques used for dimensionality reduction are:

- Principle Component Analysis
- Singular Value Decomposition

This is done by searching for a plane with a lower count of dimensions, which approximately fits the data values.

Data compression

By compressing the data, data mining algorithms can be trained faster. Besides, redundant data such as equal measures, but in different units, could be compressed/merged.

Visualization

Dimensionality reduction can improve how we display information in a tractable manner for human consumption. This is important because visually understandable information often helps to develop algorithms.

8.2 Singular Value Decomposition (SVD)

Theorem 8.1: SVD

$$A_{[m \cdot n]} = U_{[m \cdot r]} \Sigma_{[r \cdot r]} V^T_{[n \cdot r]}$$

Where:

A = Input data matrix
I.E m documents, n terms

U = Left Singular vectors
I.E m documents, r concepts

Σ = Singular values
r - rank of matrix A

V = Right Singular vectors
I.E n terms, r concepts

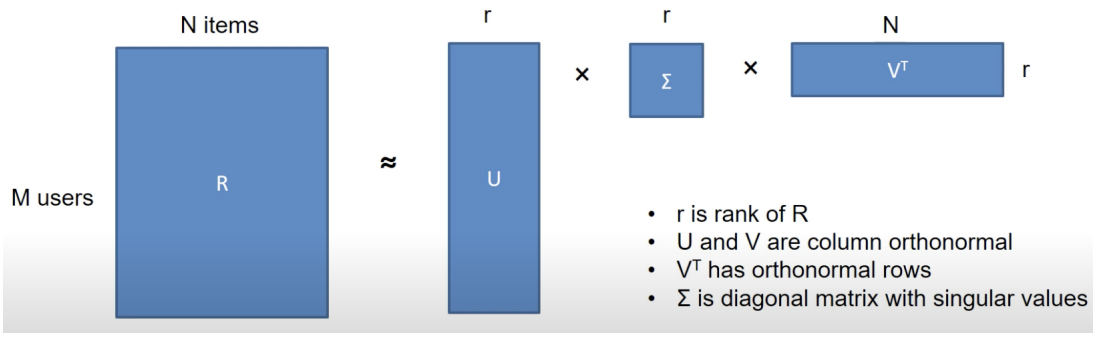


Figure 8.1: Single Value Decomposition

For information on how to compute the singular value decomposition (with examples), visit: Singular Value Decomposition (SVD) tutorial.

8.3 Principal Component Analysis (PCA)

PCA is the most commonly used dimensionality reduction method.

8.3.1 Goal

The goal of the PCA is to find a single line (or plane in higher dimensions) onto which to project the input data.

The line/plane can be found by finding a lower-dimensional surface so the sum of squares onto that surface is minimized.

8.3.2 General Case

- Given an $m * n$ matrix, the goal is to find an $m * k$ matrix.
- Find a set of vectors which we project the data onto the linear subspace spanned by that set of vectors.
- We can define a point in a plane with k dimensional vectors.

8.3.3 Algorithm

The PCA Algorithm requires some preprocessing to work properly.

- Step 1: Preprocessing
  - Mean normalization - Replace each x_{ji} with x_{ji}-μ_j, subtract the mean from the value, so the mean is re-scaled to be 0.
  - Feature scaling (dependent on data) - If the features have very different scales, normalize them by changing x_{ji} to $\frac{x_{ij}-\mu_j}{s_j}$, where s_j is some measure of the range, i.e. *max*, *min*, or *standard deviation*.
- Step 2: Compute the covariance matrix
  - $COV(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \hat{x})(y_i - \hat{y})$
  - This results in an $[n * n]$ matrix.
  - x_i and y_i are $[m * 1]$ vectors.
- Step 3: SVD
  - Compute the SVD on the covariance matrix C.
  - $\left[U, \Sigma, V^T\right] = SVD(C)$
  - The U matrix is an $[n * r]$ matrix.
  - To reduce the system from n-dimensions to k-dimensions, take the first k vectors from U (The first k columns).

8.3.4 Transformation

To change X (which is n dimensional) to z (which is k dimensional):

1. Take the first k columns of the U matrix and stack in columns. This gives the $n * k$ matrix $U_{Reduced}$.
2. Calculate z: $z = U_{Reduced}^T * X$
3. $[[k * n]] * [[n * 1]]$
4. This generates a matrix which is $k * 1$ per data row.

8.3.5 Meassure Quality of PCA

To meassure how good the implemented PCA is, we can meassure the reconstruction error:

- Given the reduced data $U_{Reduced}$ in K dimensions .
- $X_{Approx} = U_{Reduce} * Z$

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x_i - x_i^{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x_i\|^2} \leq 0.01 \tag{8.1}$$

8.3.6 Choosing the Parameter k

We want to choose the k value which results in the smallest ratio between the averaged squared projection error with the total variation in the data. This makes sense, as we want $x_i \approx x_i^{Approx}$ to retain as much information as possible.

9 Advanced Classifiers (Naive Bayes)

9.1 Bayes Theorem

Theorem 9.1

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

9.2 Bayes Theorem for Continuous Values

We need to assume that the values we are interested in have a Gaussian distribution.

- Find the mean μ
- Find the variance σ^2
- Use the Gaussian distribution to calculate the probability $P(A)$

For conditional probabilities:

- Find the mean μ_B for only the condition B
- Find the variance σ_B^2 for only the condition B
- Use the Gaussian distribution to calculate the conditional probability $P(A|B)$

For inequalities, we need to integrate the Gaussian density (preferably by computing the standard Gaussian density first).

9.3 Bayesian Classifiers

9.3.1 Goal

Given a record with attributes $(A_1, A_2, \cdots, A_n)$, the goal of the bayesian classifier is th predict the class C. Specifically, we want the classifier to fint the value of C that maximizes $P(C|A_1, A_2, \cdots, A_n)$.

9.3.2 Approach

1. Compute the posterior probability for all values of C based on Bayes theorem.
2. Choose the value of C that maximizes the posterior probability.
    - This is equivalent to choosing the value of C that maximizes $P(A_1, A_2, \cdots, A_n|C)P(C)$, because the denominators are equal

Computing $P(A_1, A_2, \cdots, A_n|C)P(C)$ is done as such:

1. Assume independence among attributes A_i when the class is given:
    - $P(A_1, A_2, \cdots, A_n|C) = P(A_1|C_j)P(A_2|C_j)\cdots P(A_n|C_j)$
2. Estimate $P(A_i|C_j)$ for all A_i and C_j
3. The new point is classified to C_j if $P(C_j)\prod P(A_i|C_j)$ is maximal

9.3.3 Estimating Probabilities from Data

For continuous attributes:

1. Discretize the range into bins:
    - One ordinal attribute per bin
    - Violates independence assumption
2. Two-way split
    - Choose only one of the two splits as a new attribute
3. Probability density estimation
    - Assume that the attribute follows a normal distribution
    - Estimate the mean and the variance
    - Estimate the conditional probability $P(A_i|C)$ from the known distribution

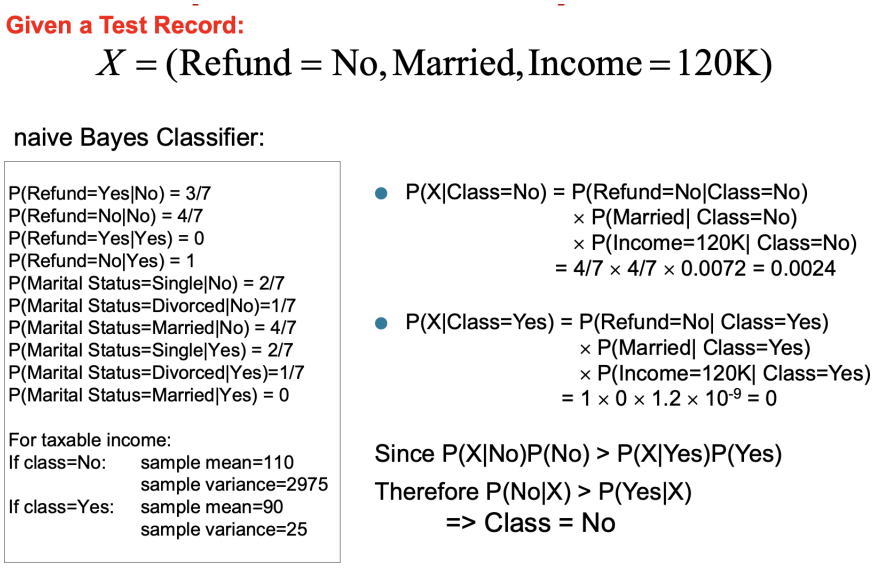


Figure 9.1: Example of deciding class

9.3.4 Naive Bayes Classifier problems

In the previous figure, one of the conditional probabilities is zero, meaning that the entire expression becomes zero. This is often an effect of having an incomplete training set. This means that the class, given the conditional probability, will never be chosen.

This can be "fixed" by using some type of smoothing:

- Original: $P(A_i|C) = \frac{N_{ic}}{N_c}$
- Laplace: $P(A_i|C) = \frac{N_{ic}+1}{N_c+c}$
- m-estimate: $P(A_i|C) = \frac{N_{ic}+mp}{N_c+m}$

Where c = number of classes, p = prior probability, and m = own parameter.

9.4 Bayesian Belief Network (BBN)

- Conditional independence assumption is not always practical.
- A Bayesian belief network provides a graphical representation of dependencies.
  - A directed acyclic graph (dag) encoding the dependence relationships among a set of variables.
  - A probability table associating each node to its immediate parent node.
- Calculating the probability of a variable, which is dependent on some other variables, can often be simplified to only depend on the direct parent(s).

9.5 Summary of Naive Bayes

- Robust to isolated noise points.
- Handle missing values by ignoring the instance during probability estimate calculations.
- Robust to irrelevant attributes.
- Independence assumption may not hold for some attributes.
  - Use other techniques such as Bayesian Belief Networks (BBN).

10 Advanced Classifiers (Linear Regression)

10.1 Notation

- m = The number of training examples
- x = The input variables / features
- y = The output variable/"target" variables
- (x_i, y_i) = A specific example (i_{th} training example)

Passing the features into a learning algorithm that outputs a function (denoted h = hypothesis), resulting in the function:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

where θ_i are parameters, θ_0 is the zero condition, and θ_1 is the gradient.

10.2 Cost Function

A cost function lets us figure out how to fit the best straight line to our data. We want to solve a minimization problem where we minimize $(h_\theta(x) - y)^2$.

The most common cost function is the mean squared error (MSE):

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Note that multiplying the cost function by $\frac{1}{2}$ does not change the location of the minima, which is why we use it to form the following cost function:

One half mean squared error:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

The cost function determines the parameters, and the values associated with the parameters determine how the hypothesis behaves. Different values generate different hypotheses.

The MSE cost function is convex.

Finding the global minima in linear regression can be done with a gradient descent algorithm.

10.2.1 Cost Function for Multiple Variables

$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

10.3 Batch Gradient Descent Algorithm (BGD)

- In each step you look at all the training data

Progress

1. Start with initial guesses, i.e. $(0, 0)$
2. Keep changing θ_0 and θ_1 a little bit to reduct $J(\theta_0, \theta_1)$
3. Each time the parameters change, select the gradient which reduces $J(\theta_0, \theta_1)$ the most
4. Repeat until we converge to a local/global minima

Different starting points will often result in different minimas.

Formal progress

- Do the following until convergence:
  - $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
  - For $j = 0$ and $j = 1$ in this example.
- α is the learning rate
- α controls how aggressive the gradient descent is

"For $j = 0$ and $j = 1$" means that we update both parameters simultaneously. This is done by first solving the derivative for θ_0 and storing it in a temporary variable. Do the same for θ_1, and then update the original θ_j-values.

Partial Derivatives

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i) * 1$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i) x_i^{(1)}$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \cdots, \theta_n) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \cdots + \theta_n x_i - y_i) x_i^{(j)}$$

10.3.1 Gradient Descent in Practice

- Feature Scaling (normalize)
  - Problems with multiple features should be scaled to a more even scale across the features.
  - One common way is to use the mean to scale all data into the range $[-1, 1]$.
  - Try to create a range between $[-3, 3]$ and $\left[-\frac{1}{3}, \frac{1}{3}\right]$.
  - This is done by $x_i = \frac{x_i - \hat{x}}{max(x) - min(x)}$
- Running gradient descent on this kind of cost function can take a long time to find the global minima.

10.4 Practical Tips

- Plot $min(J(\theta))$ with the number of iterations to clearly illustrate the convergence.
- Start with a low learning rate, and gradually increase it. It is normal to go for roughly treefold increments (multiply by 3).
- We could also use a polynomial regression on the form $(\theta_0 + \theta_1 x + \theta_2 x^2)$.

11 Advanced Classifiers (Logistic Regression)

11.1 The Logistic Classifier

The logistic regression model is a binary classification problem.

Linear regression for binary classification is not a viable option as the decision boundary will not separate the two categories correctly because of the nature of the mean square error.

The classifier should output either 1 or 0, depending on the data.

For the classification hypothesis representation, we have:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

This is the sigmoid function (the logistic function).

The function is used as such:

$$\begin{cases} 1 & \text{if } h_\theta(x) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{11.1}$$

11.1.1 Logistic Classifier for Higher-Order Terms

As in polynomial regression, the logistic classifier can be used for higher order terms by adding higher order terms. I.e. $g(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2 + \theta_4 x_2^2)$

11.2 Problems with the Linear Regression Cost Function

The mean square error function results in a convex PDF when given it is given linear values. This is not the case for logistic regression. This means that the MSE would most likely only find a local minimum, not a global one.

11.3 Logistic Regresssion Cost Function

$$Cost(h_\theta(x), y) = \begin{cases} -ln(h_\theta(x)) & \text{if } y = 1 \\ -ln(1 - h_\theta(x)) & \text{if } y = 0 \end{cases} \tag{11.2}$$

This can be rewritten into a single function:

$$Cost(h_\theta(x), y) = -yln(h_\theta(x)) - (1-y)ln(1-h_\theta(x)) \tag{11.3}$$

11.4 Process

The process forwards is the same as in the linear regression chapter.

11.5 Multiclass Classifier for Logistic Regression

Extending the logistic regression from a binary classifier to a multiclass classifier can be done in a "one vs. all" classification.

This means that the algorithm is conducted multiple times, each time changing which class is the positive class (where all the other classes are negative).

12 Classification (Regularization)

12.1 Addressing Overfitting

Multiple strategies can be used to check if overfitting occurs:

- Plot the hypothesis to check for overfitting. Might not always work.
- Reduce the number of features
- Regularization

12.2 Regularization

- Small values for parameters correspond to a simpler hypothesis (you effectively get rid of some of the terms).
- A simpler hypothesis is less prone to overfitting.

12.2.1 Cost function optimalization

We can penalize some of the higher-order θ parameters and make them very small:

$$min \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 + 1000\theta_3^2 + 1000\theta_4^2$$

12.2.2 Regularization on features

Unlike in a polynomial, we don't know which parameters are the high-order terms. The solution for this is to penalize all the parameters.

By convention, we don't penalize θ_0, but all other parameters, with the regularization parameter λ.

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

12.2.3 Regularized Gradient Descent

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right) x_i^{(j)} + \frac{\lambda}{m} \theta_j \right]$$

$$\theta_j := \theta_j \left(1 - \alpha \frac{\lambda}{m} \right) - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right) x_i^{(j)}$$

Given a typical small learning rate λ and a large number of terms m, we see that θ_j is multiplied by a value close to 1 (around $0.95 - 0.99$).

13 Deep Learning (Neural Network)

13.1 Advantages

Logistic regression with a polynomial hypothesis is often way to expensive. This is because each feature space has to be compared to each other, which leads to way too many input features to optimize.

13.2 Artificial Neuron

- Feed input via input nodes
- Logistic unit does some computation
- Sends output

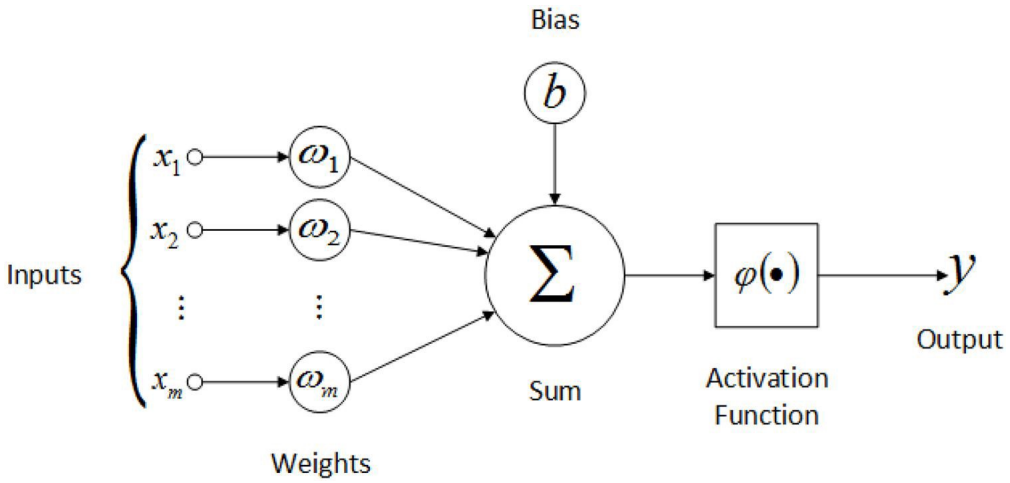


Figure 13.1: Artificial Neuron

The activation function is typically the sigmoid function, a hyperbolic tangent, or a rectified linear unit (ReLU).

13.3 Notations

- $a_i^{(j)}$ - Activation of unit i in layer j (the output of the node)
- $\theta^{(j)}$ - Matrix of parameters (controlling the activation from one layer to the next)
- $g(\theta^T x)$ - The activation function

13.4 Multiclass Classification

- Given N classes:
  - Build a neural network with N output units
  - Use one-hot encoding to encode the output as an array of $N-1$ zeros and one 1.

13.5 Cost function for Neural Networks

The cost function can generate k outputs, meaning that the sigmoid function (or any other cost functions) returns a k dimensional vector.

The cost function is combined by two parts.

The first part finds the mean of the sum of each training sample m, over the sum of each class k. All classes has a "one-vs-all" comparison. This gives the following expression:

$$-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta(x^{(i)})\right)_k + (1-y_k^{(i)}) \log\left(1-(h_\Theta(x^{(i)}))_k\right)\right]$$

The second part is the regularization term. In this part, we sum all layers $L-1$, over all units in each layer s_t, over all the input layers j:

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_t}\sum_{j=1}^{s_l+1}(\Theta_{ji}^{(l)})^2$$

13.6 Learn this better...