Supervised Learning
- Part 2 -

Road Map

- Classification using class association rules
- Naïve Bayesian classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging, Boosting, Random Forest
- Summary

CAR definition

IF:

- I is a set of items, $I = \{i_1, i_2, \ldots, i_n\}$,
- C a set of classes ($C \cap I = \varnothing$), and
- T a set of transactions, $T = \{t_1, t_2, \ldots, t_m\}$ where each transaction is labeled with a class label $c \in C$,

THEN:

- a class association rule (CAR) is a construction with the following syntax:

$$X \rightarrow y$$

where $X \subseteq I$ and $y \in C$.

Example: Dataset

- a set of six transactions labeled with classes from C = C = {database, datamining, programming}:

Doc1	{rule, tree, classification}	datamining
Doc2	{relation, tuple, join, algebra, recommendation}	database
Doc3	{variable, loop, procedure, rule}	programming
Doc4	{clustering, rule, tree, recommendation}	datamining
Doc5	{join, relation, selection, projection, classification}	database
Doc6	{rule, tree, recommendation}	datamining

Support and confidence

- The following constructions are valid CARs:

$$\text{rule} \rightarrow \text{datamining};$$

$$\text{recommendation} \rightarrow \text{database}$$

- For each CAR the support and confidence may be computed:

$$\sup(X \rightarrow y) = \frac{\textit{Number of transactions containing } X \textit{ and labeled with } y}{\textit{Total number of transactions}}$$

$$conf(X \rightarrow y) = \frac{\textit{Number of transactions containing } X \textit{ and labeled with } y}{\textit{Number of transactions containing } X}$$

Example: support and confidence

Doc1	**{rule, tree, classification}**	**datamining**
Doc2	**{relation, tuple, join, algebra, recommendation}**	**database**
Doc3	**{variable, loop, procedure, rule}**	**programming**
Doc4	**{clustering, rule, tree, recommendation}**	**datamining**
Doc5	**{join, relation, selection, projection, classification}**	**database**
Doc6	**{rule, tree, recommendation}**	**datamining**

Using these expressions:

- sup(rule → datamining) = 3/6 = 50%, and
- conf(rule → datamining) = 3/4 = 75%.

Using CARs

- There are presented two methods for using CARs in classification (see [Liu 11]):
  - Use CARs for building classifiers
    - Strongest rule
    - Subset of rules
  - Use CARs to build new attributes of the dataset

Strongest rule

- In this case after CARs are obtained by data mining (as described in chapter 3) the CARs set is used for classifying new examples:
  - For each new example the strongest rule that covers that example is chosen for classification (its class will be assigned to the test example.
- Strongest rule means rule with the highest confidence and/or support. There are also other measures for rule strength (chi-square test from statistics for example).
- This is the simplest method to use CARs for classifications: Rules are ordered by their strength and for each new test example the ordered rule list is scanned and the first rule covering the example is picked up.
- A CAR covers an example if the example contains the left side of the rule (the transaction contains all the items in rule left side).

Strongest rule: example

- For example if we have an ordered rule list:

$$\text{rule} \rightarrow \text{datamining};$$

$$\text{variable} \rightarrow \text{programming}$$

$$\text{recommendation} \rightarrow \text{database}$$

- Then the transaction

Doc-ex	{rule, variable, loop, recommendation}	?

will be labeled with class ‘datamining’ because of the first rule-strongest than the other rules.

Subset of rules

This method is used in *Classification Based on Associations* (CBA). In this case, having a training dataset D and a set of CARs R, the objectives are:

A. to order R using their support and confidence, R = $\{r_1, r_2, \ldots, r_n\}$:
   1. First rules with highest confidence
   2. For the same confidence use the support to order the rules
   3. For the same support and confidence order by rule generation-time (rules generated first are 'greater' than rules generated later).

Subset of rules

B. to select a subset S of R covering D:

1. Start with an empty set S
2. Consider ordered rules from R in sequence: for each rule **r**
   - If **r** covers at least an example in D add **r** to S and remove covered examples from D
3. Stop when D is empty
4. Add the majority class as default classification.

The result is:

Classifier = **<r_{i1}, r_{i2}, ..., r_{ik}, majority-class>**

Using CARs

- There are presented two methods for using CARs in classification (see [Liu 11]):
  - Use CARs for building classifiers
    - Strongest rule
    - Subset of rules
  - Use CARs to build new attributes of the dataset

Build new attributes (features)

- In this approach the training dataset is enriched with new attributes, one for each CAR:

FOREACH transaction
IF transaction is covered by the left part of the CAR
THEN the value of the attribute is 1 (or TRUE)
ELSE the value of the new attribute is 0 (or FALSE)
ENDIF
ENDFOR

- There are many other methods to use a set of CARs for building classifiers, for examples grouping rules and measuring the strength of each group, etc.

Use of association rules

- Usual association rules may be also used in recommendation systems:
- The rules are ordered by their confidence and support and then may be used, considering them in this order, for labeling new examples
- Labels are not classes but other items (recommendations).
- For example, based on a set of association rules containing books, the system may recommend new books to customers based on their previous orders.

Road Map

- Classification using class association rules
- Naïve Bayesian classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging, Boosting, Random Forest
- Summary

Naïve Bayes: Overview

- This approach is a probabilistic one.
- The algorithms based on Bayes theorem compute for each test example not a single class but a probability of each class in C (the set of classes).
- If the dataset has k attributes, A_1, A_2, …, A_k, the objective is to compute for each class $c \in C = \{c_1, c_2, \ldots, c_n\}$ the probability of the test example $(a_1, a_2, \ldots, a_k)$ to belong to the class c:

$$\mathbf{Pr(Class = c \mid A_1 = a_1, ..., A_k = a_k)}$$

- If classification is needed, the class with the highest probability may be assigned to that example.

Bayes theorem

- Thomas Bayes (1701 – 1761) was a Presbyterian minister and an English mathematician.
- The theorem named after him may be expressed as follows:

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

where P(A|B) means probability of A given B.

Example

- “Students in EC101 are 30% from the ABD M.Sc. module and 70% from other modules.
- 20% of the students are placed in the first 5 rows of seats but for ABD this percent is 40%.
- When the dean enters the class and sits somewhere in the first 5 rows, near a student, compute the probability that its neighbor is from ABD?”

Solution

$$\Pr(ABD) = 0.3$$

$$\Pr(5rows \mid ABD) = 0.4$$

$$\Pr(5rows) = 0.2$$

- So:

$$\Pr(ABD \mid 5rows) = \frac{\Pr(5rows \mid ABD) * \Pr(ABD)}{\Pr(5rows)} = \frac{0.4 * 0.3}{0.2} = 0.6 = 60\%$$

Building classifiers

- The objective is to compute

$$\Pr(\text{Class} = c \mid A_1 = a_1, \ldots, A_k = a_k).$$

- Applying Bayes theorem:

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_k = a_k) =$$

$$\frac{\Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_j) * \Pr(C = c_j)}{\Pr(A_1 = a_1, \ldots, A_k = a_k)} =$$

$$\frac{\Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_j) * \Pr(C = c_j)}{\sum_{x=1}^{n} \Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_x) \Pr(C = c_x)}$$

Building classifiers

- Making the following assumption: “all attributes are conditionally independent given the class $C=c_j$” then:

$$\Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_j) = \prod_{i=1}^{k} \Pr(A_i = a_i \mid C = c_j)$$

- Because of this assumption the method is called “naïve”.
- Not in all situations the assumption is valid.
- The practice shows that the results obtained using this simplifying assumption are good enough in most of the cases.

Building classifiers

- Finally, replacing in the above expression we obtain:

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_k = a_k) =$$

$$\frac{\Pr(C = c_j) * \prod_{i=1}^{k} \Pr(A_i = a_i \mid C = c_j)}{\sum_{x=1}^{n} \Pr(C = c_x) \prod_{i=1}^{k} \Pr(A_i = a_i \mid C = c_j)}$$

- All probabilities in the above expression may be obtained by counting.

Building classifiers

- When only classification is needed, the denominator of the above expression may be ignored (is the same for all c_j) and the labeling class is obtained by maximizing the numerator:

$$C = argmax_{cj} \Pr(c_j) * \prod_{i=1}^{k} \Pr(A_i = a_i | C = c_j)$$

Example

- Consider a simplified version of the PlayTennis table :

Outlook	Wind	Play Tennis
Overcast	Weak	Yes
Overcast	Strong	Yes
Overcast	Absent	No
Sunny	Weak	Yes
Sunny	Strong	No
Rain	Strong	No
Rain	Weak	No
Rain	Absent	Yes

Example

$Pr(Yes) = 4/8$
$Pr(Overcast \mid C = Yes) = 2/4$
$Pr(Overcast \mid C = No) = 1/4$
$Pr(Sunny \mid C = Yes) = 1/4$
$Pr(Sunny \mid C = No) = 1/4$
$Pr(Rain \mid C = Yes) = 1/4$
$Pr(Rain \mid C = No) = 2/4$

$Pr(No) = 4/8$
$Pr(Weak \mid C = Yes) = 2/4$
$Pr(Weak \mid C = No) = 1/4$
$Pr(Strong \mid C = Yes) = 1/4$
$Pr(Strong \mid C = No) = 2/4$
$Pr(Absent \mid C = Yes) = 1/4$
$Pr(Absent \mid C = No) = 1/4$

- If the test example is:

Sunny	**Absent**	**???**

Example

For C = Yes

$$\Pr(Yes) * \Pr(Sunny \mid Yes) * \Pr(Absent \mid Yes) = \frac{4}{8} * \frac{1}{4} * \frac{1}{4} = \frac{1}{32}$$

For C = No

$$\Pr(No) * \Pr(Sunny \mid No) * \Pr(Absent \mid No) = \frac{4}{8} * \frac{1}{4} * \frac{1}{4} = \frac{1}{32}$$

- The result is that both probabilities have the same value and the class may be any of them.

Special case: division by 0

- Sometimes a class does not occur with a specific attribute value.
- In that case one term Pr(Ai = ai | C = cj) is zero, so the above expression for probabilities of each class evaluates to 0/0.
- For avoiding this situation, the expression:

$$\Pr(A_i = a_i | C = c_j) = \frac{a}{b}$$

must be modified.

(a = number of training examples with $A_i = a_i$ and $C = c_j$ and b = number of training examples with $C = c_j$)

Special case: division by 0

- The modified expression is:

$$\Pr(A_i = a_i | C = c_j) = \frac{a + s}{b + s * r}$$

where:

- s = 1 / Number of examples in the training set
- r = Number of distinct values for A_i

- In this case all product terms are greater than zero.

Special case: values

Non-categorical values or absent values:

- All non-categorical attributes must be discretized (replaced with categorical ones).
- Also, if some attributes have a missing value, these values are ignored.

Road Map

- Classification using class association rules
- Naïve Bayesian classification
- Support vector machines (SVMs)
- K-nearest neighbor
- Ensemble methods: Bagging, Boosting, Random Forest
- Summary

SVM: Overview

- In this course is presented only the general idea of the Support Vector Machines (SVM) classification method.
- SVMs are described in detail in many documentations and books, for example [Liu 11] or [Han, Kamber 06].
- The method was discovered in Soviet Union in '70 by Vladimir Vapnik and was developed in USA after Vapnik joined AT&T Bell Labs in early '90 (see [Cortes, Vapnik 95]).

SVM: Model

- Consider the training data set D = {(X_1, y_1), (X_2, y_2), ..., (X_k, y_k)} where:
  - $X_i = (x_1, x_2, ..., x_n)$ is a vector in R^n (all x_i components are real numbers)
  - y_i is the class label, $y_i \in \{-1, +1\}$. If Xi is labeled with +1 it belongs to the ***positive class***, else to the ***negative class*** (-1).

SVM: Model

- A possible classifier is a linear function:

$$f(X) = \langle w \cdot X \rangle + b$$

such as:

$$y_i = \begin{cases} 1 \; if \; \langle w \cdot X \rangle + b \geq 0 \\ -1 \; if \; \langle w \cdot X \rangle + b < 0 \end{cases}$$

where:

- **w** is a weight vector,
- $\langle w \cdot X \rangle$ is the dot product of vectors **w** and **X**,
- **b** is a real number and
- **w** and **b** may be scaled up or down as shown below.

SVM: Model

- The meaning of **f** is that the hyperplane

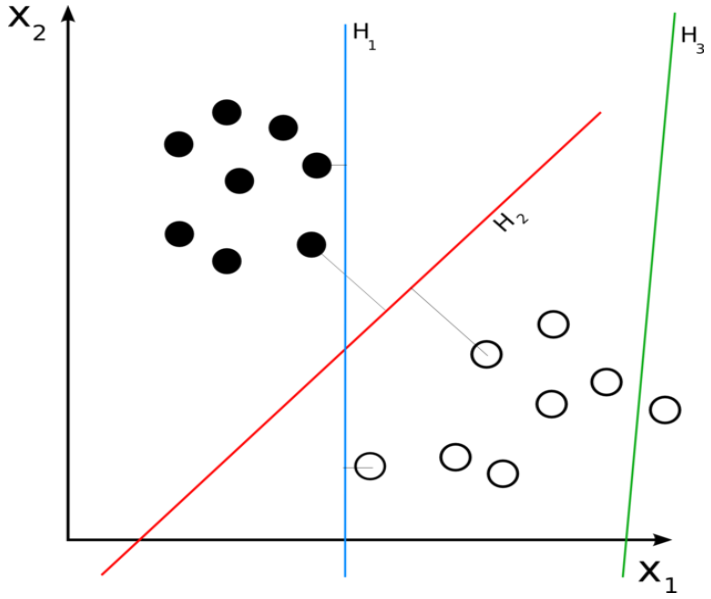
$$< w \cdot X > + b = 0$$

separates the points of the training set D in two:

  - one half of the space contains the positive values and
  - the other half the negative values in D (like hyperplanes H1 and H2 in the next figure).

- All test examples can now be classified using f: the value of f is the label for the example.

Figure 1

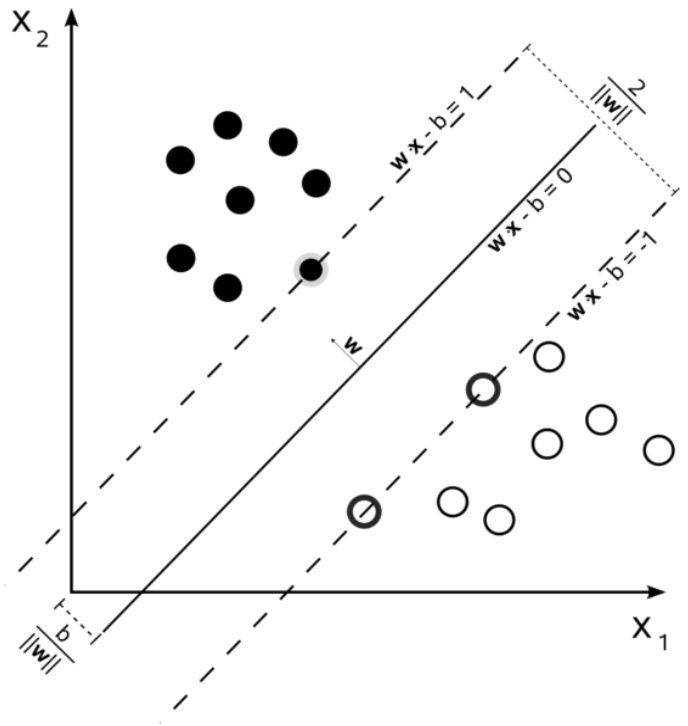


- Source: Wikipedia

Best hyperplane

- SVM tries to find the ‘best’ hyperplane of that form.
- The theory shows that the best plane is the one maximizing the so-called ***margin*** (the minimum orthogonal distance between a positive and negative point from the training set – see next figure for an example.

Figure 2



- Source: Wikipedia

The model

- Consider X^+ and X^- the nearest positive and negative points for the hyperplane

$$<w \cdot X> + b = 0$$

- Then there are two other parallel hyperplanes, H_+ and H_- passing through X^+ and X^- and their expression is:

$H_+ : <w \cdot X> + b = 1$

$H_- : <w \cdot X> + b = -1$

- These two hyperplanes are with dotted lines in Figure 1. Note that **w** and **b** must be scaled such as:

$<w \cdot Xi> + b \geq 1$ for $yi = 1$

$<w \cdot Xi> + b \leq -1$ for $yi = -1$

The model

- The margin is the distance between these two planes and may be computed using vector space algebra obtaining:

$$\text{Margin} = \frac{2}{\|w\|}$$

- Minimizing the margin means maximizing the value of

$$\frac{< w \cdot w >}{2}$$

- The points X^+ and X^- are called ***support vectors*** and are the only important points from the dataset.

Definition: separable case

- When positive and negative points are linearly separable, the SVM definition is the following:
  - Having a training data set D = $\{(X_1, y_1), (X_2, y_2), ..., (X_k, y_k)\}$
  - Minimize the value of expression (1) above
  - With restriction: y_i ($<w \cdot X_i> + b) \geq 1$, knowing the value of y_i: +1 or -1
- This optimization problem is solvable by rewriting the above inequality using a Lagrangian formulation and then finding solution using Karush-Kuhn-Tucker (KKT) conditions.
- This mathematical approach is beyond the scope of this course.

Non-linear separation

- In many situations there is no hyperplane for separation between the positive and negative examples.
- In such cases there is possible to map the training data points (examples) in another space, a higher dimensional one.
- Here data points may be linearly separable.
- The mapping function gets examples (vectors) from the input space X and maps them in the so-called ***feature space*** F:

$$\phi : X \rightarrow F$$

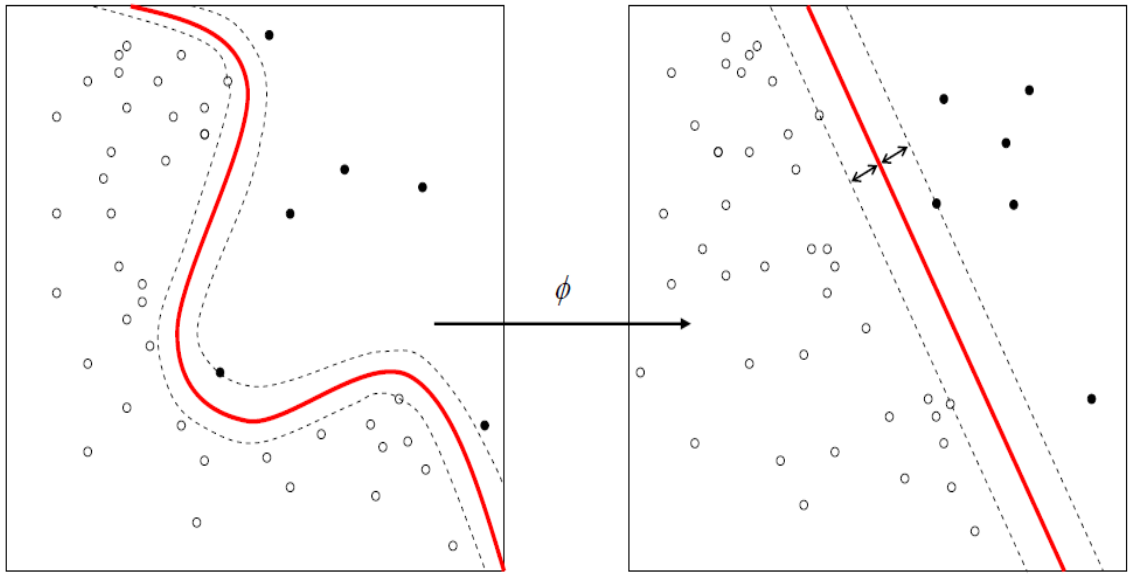
Non-linear separation

- Each point X is mapped in $\phi(X)$. So, after mapping the whole D there is another training set, containing vectors from F and not from X, with $\dim(F) \geq n = \dim(X)$:

$$D = \{(\phi(X_1), y_1), (\phi(X_2), y_2), ..., (\phi(X_k), y_k)\}$$

- For an appropriate ϕ, these points are linearly separable.
- An example is the next figure.

Figure 3



- Source: Wikipedia

Kernel functions

- But how can we find this mapping function?
- In solving the optimization problem for finding the linear separation hyperplane in the new feature space F all terms containing training examples are only of the form $\phi(X_i) \cdot \phi(X_j)$.
- By replacing this dot product with a function in both X_i and X_j the need for finding ϕ disappears. Such a function is called a ***kernel function***:

$$K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$$

- For finding the separation hyperplane in F we must only replace all dot products with the chosen kernel function and then proceed with the optimization problem like in separable case.

Kernel functions

Some of the most used kernel functions are:

- Linear kernel

$$K(X, Y) = <X \cdot Y> + b$$

- Polynomial Kernel

$$K(X, Y) = (a * <X \cdot Y> + b)^p$$

- Sigmoid Kernel

$$K(X, Y) = \tanh(a * <X \cdot Y> + b)$$

Other aspects concerning SVMs

- SVM deals with continuous real values for attributes.
  - When categorical attributes exists in the training data a conversion to real values is needed.
- When more than two classes are needed SVM can be used recursively.
  - First use separates one class; the second use separates the second class and so on. For N classes N-1 runs are needed.
- SVM are a very good method in hyper dimensional data classification.

Road Map

- Classification using class association rules
- Naïve Bayesian classification
- Support vector machines
- K-nearest neighbor (kNN)
- Ensemble methods: Bagging, Boosting, Random Forest
- Summary

kNN

- K-nearest neighbor (kNN) ***does not produce a model*** but ***is a simple method*** for determining the class of an example based on the labels of its neighbors belonging to the training set.
- For running the algorithm a ***distance function*** is necessary for computing the distance from the test example to the examples in the training set.
- A function f(x, y) may be used as distance function if four conditions are met:
  - $f(x, y) \geq 0$
  - $f(x, x) = 0$
  - $f(x, y) = f(y, x)$
  - $f(x, y) \leq f(x, z) + f(z, y)$.

Algorithm

Input:

- A dataset **D** containing labeled examples (the training set)
- A distance function **f** for measuring the dissimilarity between two examples
- An integer **k** – parameter telling how many neighbors are considered
- A test example **t**

Output:

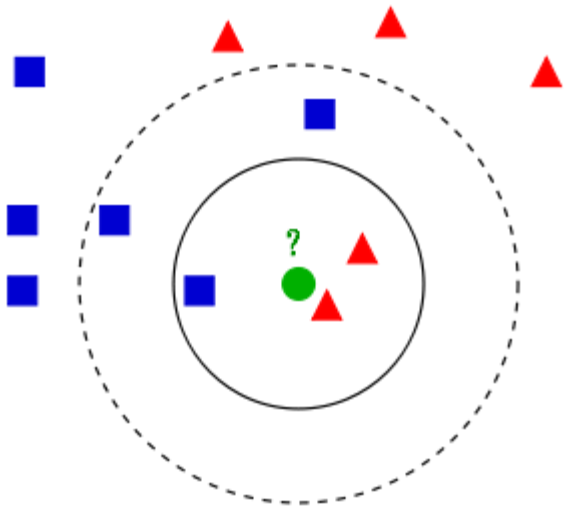
- The class label of **t**

Method:

- Use **f** to compute the distance between **t** and each point in D
- Select nearest **k** points
- Assign t the majority class from the set of **k** nearest neighbors.

Example

- K = 3 → Red
- K = 5 → Blue



- kNN is very sensitive to the value of parameter k.
- The best k may be found by cross validation for example.

Road Map

- Classification using class association rules
- Naïve Bayesian classification
- Support vector machines
- K-nearest neighbor (kNN)
- Ensemble methods: Bagging, Boosting, Random Forest
- Summary

Ensemble methods

- Ensemble methods combine multiple classifiers to obtain a better one.
- Combined classifiers are similar (use the same learning method) but the training datasets or the weights of the examples in them are different.

Bagging

- The name Bagging comes from **B**ootstrap **Agg**regat**ing**.
- As presented in the previous lesson bootstrap method is part of resampling methods and consists in getting a training set from the initial labeled data by sampling with replacement.

Example

Original dataset	A	b	c	d	e	f
Training set 1	A	b	b	c	e	f
Training set 2	B	b	c	c	d	e
Training set 3	A	b	c	c	d	f

Bagging

Bagging consists in:

- Starting with the original dataset, build **n** training datasets by sampling with replacement (bootstrap samples)
- For each training dataset build a classifier using the same learning algorithm (called ***weak classifiers***).
- The final classifier is obtained by combining the results of the weak classifiers (by voting for example).
- Bagging helps to improve the accuracy for unstable learning algorithms: decision trees, neural networks.
- It does not help for kNN, Naïve Bayesian classification or CARs.

Boosting

- Boosting consists in building a sequence of weak classifiers and adding them in the structure of the final strong classifier.
- The weak classifiers are weighted based on the weak learners' accuracy.
- Also data is reweighted after each weak classifier is built such as examples that are incorrectly classified gain some extra weight.
- The result is that the next weak classifiers in the sequence focus more on the examples that previous weak classifiers missed.

Random forest

- Random forest is an ensemble classifier consisting in a set of decision trees The final classifier output the modal value of the classes output by each tree.
- The algorithm is the following:

1. Choose T - number of trees to grow.
2. Choose m - number of variables used to split each node. m << M, where M is the number of input variables.
3. Grow T trees. When growing each tree do the following:
   - Construct a bootstrap sample from training data with replacement and grow a tree from this bootstrap sample.
   - When growing a tree at each node select m variables at random and use them to find the best split.
   - Grow the tree to a maximal extent. There is no pruning.
4. Predict new data by aggregating the predictions of the trees (e.g. majority votes for classification, average for regression).

Summary

This course presented:

- Classification using class association rules: CARs for building classifiers and using CARs for building new attributes (features) of the training dataset.
- Naïve Bayesian classification: Bayes theorem, Naïve Bayesian algorithm for building classifiers.
- An introduction to support vector machines (SVMs): model, definition, kernel functions.
- K-nearest neighbor method for classification
- Ensemble methods: Bagging, Boosting, Random Forest

- Next week: Unsupervised learning – part 1

References

- [Liu 11] Bing Liu, 2011. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data, Second Edition, Springer, chapter 3.
- [Han, Kamber 06] Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, Morgan Kaufmann Publishers, 2006
- [Cortes, Vapnik 95] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995. http://www.springerlink.com/content/k238jx04hm87j80g/
- [Wikipedia] Wikipedia, the free encyclopedia, en.wikipedia.org