VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Fundamentals of Machine Learning

Classification

Jan Platos

November 15, 2023

Classification

Classification

- Classification process is a process where objects are assigned to different class by their properties using a model.
- The process is divided in two or three phases - training, validation and testing.
- The training phase uses a predefined class assignment to create a model that is able to fit on the training set of objects.
- The validation phase uses a validation set of object to evaluate the quality of the model.
- The testing phase evaluates the model on a testing set of objects and evaluates the precision.

Classification

- The quality of the model depends on the principle of the algorithm and the quality of the training data.
- A no-free-lunch theorem[1] applies here.
- Each algorithm makes different types of errors.
- The goal is to train as good model as possible for a specific data.
- Auto-ML tries to solve the problem with brute force.

[1] https://en.wikipedia.org/wiki/No_free_lunch_theorem

Classification - Nearest Neighbors Classification

- Simple algorithm that utilizes a distance function for classification.
- The predicted class is the majority class of the nearest neighbors.
- Finding the nearest neighbors is the crucial part.
- Efficient data structure for space division are used.
- The main algorithms are quadrant and octrees, and kD-tree or other data structures.

Classification - Simple Probabilistic Modeling

- Simple probabilistic modeling express the resulting probability based on all attributes together.
- The probability is computed separately according to the class/label.
- Prediction is computed as a multiplication of the particular probabilities.

Classification - Simple Probabilistic Modeling

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Classification - Simple Probabilistic Modeling

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Classification - Simple Probabilistic Modeling

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$Likelihood(yes) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$Likelihood(no) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$$

$$Probability(yes) = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%$$

$$Probability(no) = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

Classification - Simple Probabilistic Modeling

- The described process works only when the attributes are equally important and independent - in theory.
- The equation is defined according to the Bayes' rule of conditional probability.

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

- The Naive assumptions allows the decomposition of the $P(B|A)$ into a multiplication of the particular probabilities.

Classification - Simple Probabilistic Modeling

- Bayes rule and Naive Bayes classifier may be used also on text data.
- One of the basic Anti Spam Filter.
- Each word is taken as a single attribute.
- The probability is processed as previously.

Classification - Inferring Rudimentary Rules

- Extract very simple classification rules from a set of instances.
- They are called 1R (1-rule).
- The rule is base don a single attribute.
- Extracted rules well characterize the structure of the data.
- The defined rules usually achieve a high precision.
- The procedure is straight-forward.
- The best rule set is selected as a result.

Classification - Inferring Rudimentary Rules

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Classification - Inferring Rudimentary Rules

Attribute	Rules	Errors	Total error
Outlook	Sunny $\rightarrow$ no	2/5	
	Overcast $\rightarrow$ yes	0/4	4/14
	Rainy $\rightarrow$ yes	2/5	
Temperature	Hot $\rightarrow$ no	2/4	
	Mild $\rightarrow$ yes	2/6	5/14
	Cool $\rightarrow$ yes	1/4	
Humidity	High $\rightarrow$ no	3/7	4/14
	Normal $\rightarrow$ yes	1/7	
Windy	False $\rightarrow$ yes	2/8	5/14
	True $\rightarrow$ no	3/6	

Classification - Inferring Rudimentary Rules

- For each attribute,
    - For each value of that attribute, make a rule as follows:
        - count how often each class appears
        - find the most frequent class
        - make the rule assign that class to this attribute-value.
    - Calculate the error rate of the rules.
- Choose the rules with the smallest error rate.

Classification - Inferring Rudimentary Rules

- The algorithm works on categorical data.
- Numeric attributes must be categorized.
- Not precise enough for a complex dataset.
- Good baseline model.

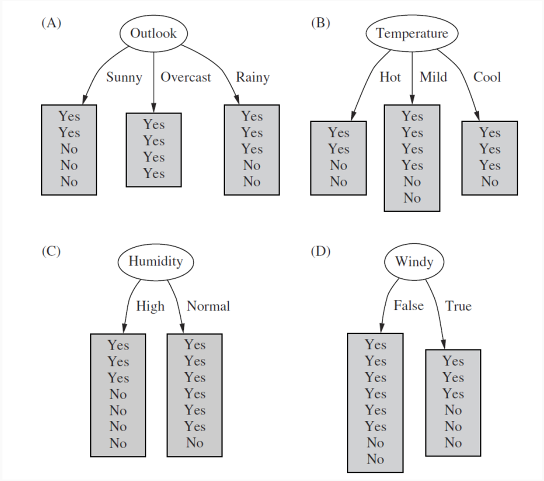
Classification - Decision Trees

- Decision Tree is a special type of n-ary tree with attributes in internal nodes and set of instances in a leaves.
- The tree is built using recursive approach.
- An attribute is selected and placed into a root node.
- The set of instances splits into subsets according to the values of the attribute.
- Each subset is processed similarly to the whole dataset.
- The process ends when a subset has the same label.

Classification - Decision Trees

- The question is which attribute have to be chosen for a set of instances.
- Each split generates a new level of a tree.
- Lets say that the small trees are better than large trees.
- Small trees are generated when a subset contains only single class - not necessary to split this node again.
- The non-diversity of a subset is called purity.

Classification - Decision Trees



Classification - Decision Trees

- How to measure the purity of the tree?
- Information (Entropy) is one of the popular option.
- Measurement of the change before and after the split is used, it is called Information Gain.
- The split that maximizes Gain is the most suitable.

Classification - Decision Trees

- Information of a set of instances with two possible labels - Info([X, Y].
- X and Y represents the number of first, resp. second label.

$$Info([X, Y]) = -p_x \log p_x - p_y \log p_y$$

$$p_x = \frac{X}{X+Y}, \quad p_y = \frac{Y}{X+Y}$$

Classification - Decision Trees

- Information of a multiple subsets is computed using weighted sum of individual information.
- Information Gain is then computed for each possible split.
- The split with the largest gain is selected and the set is done.
- Each subset is then processed with the same algorithm.
- Other attributes are used.

Classification - Decision Trees - Split Criteria

Definitions:

- S is a set of points in a branch of a tree.
- $|S|$ is size of the set (number of points in a set).
- r-way split has r subsets $S_1, \ldots, S_r$ of set S.
- k is the number of classes.

Classification - Decision Trees - Split Criteria

Error rate:

- On a set:

$$Err\left(S\right) = 1 - p$$

  - where the p is a fraction of points that belongs to the dominant class from S.

- On r-way split:

$$Err\left(S \Rightarrow S_1, \ldots, S_r\right) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} \left(1 - p\right)$$

Classification - Decision Trees - Split Criteria

Gini index:

- On a set:

$$G(S) = 1 - \sum_{j=1}^{k} p_j^2$$

  - where the p_j is a fraction of points that belongs to the class j from S.

- On r-way split:

$$G(S \Rightarrow S_1, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} G(S_i)$$

Classification - Decision Trees - Split Criteria

Entropy:

- On a set:

$$E(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

  - where the p_j is a fraction of points that belongs to the class j from S.

- On r-way split:

$$E(S \Rightarrow S_1, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} E(S_i)$$

- Decision Trees - Numeric Attributes

- Dealing with numeric attributes brings a new problem.
- Where to split the attribute and how to maintain the best possible tree.
- The validation criteria is the same - the purity of the subsets according to the class label.
- The optimal point is always between two successive values of the attribute.

- Decision Trees - Iris Example

Iris Setosa

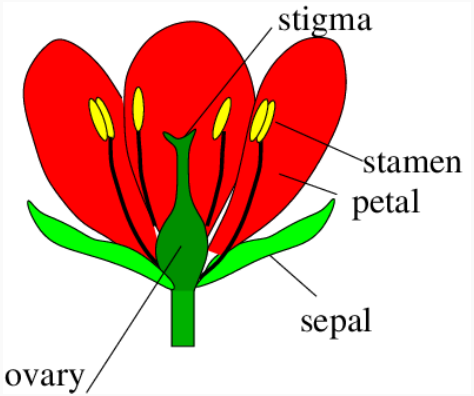
Iris Versicolor

Iris Virginica

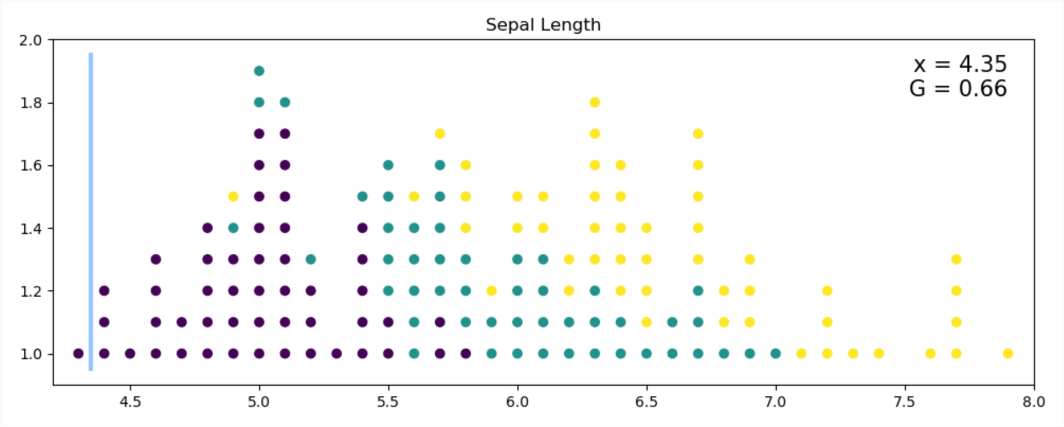
- Decision Trees - Iris Example

Features:

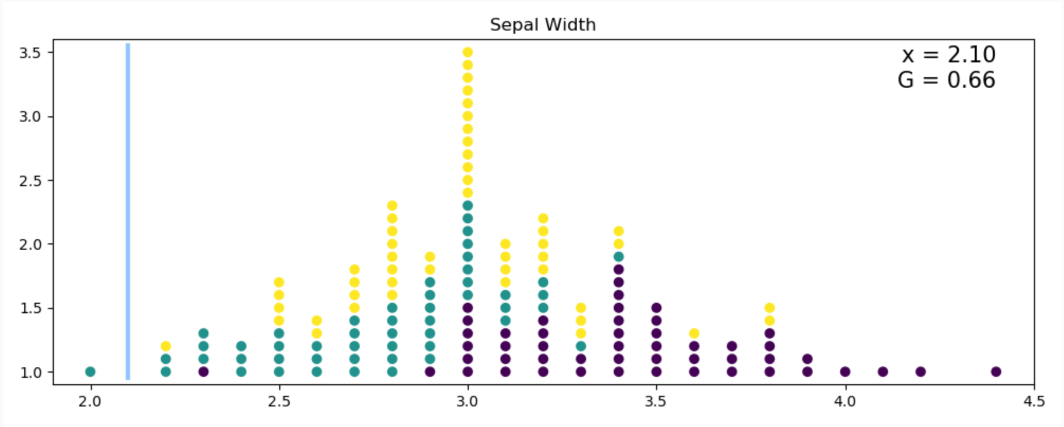
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
  - Iris Setosa
  - Iris Versicolour
  - Iris Virginica



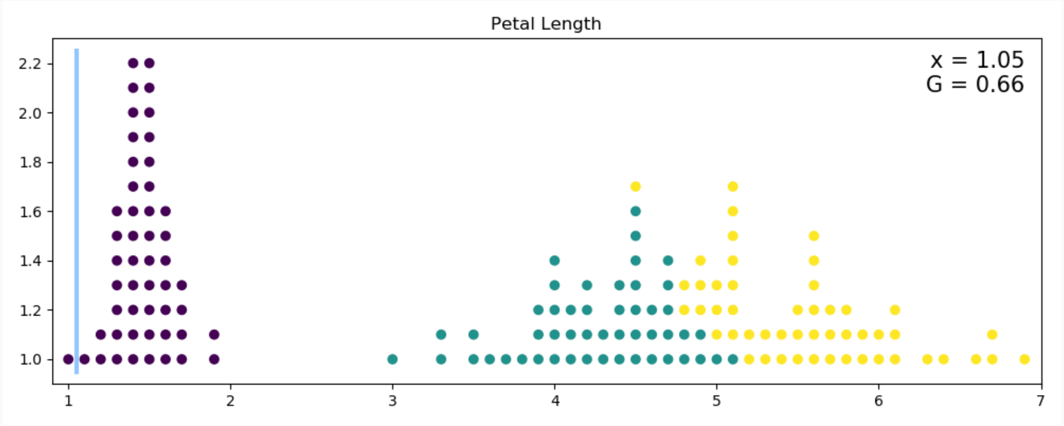
- Decision Trees - Iris Example



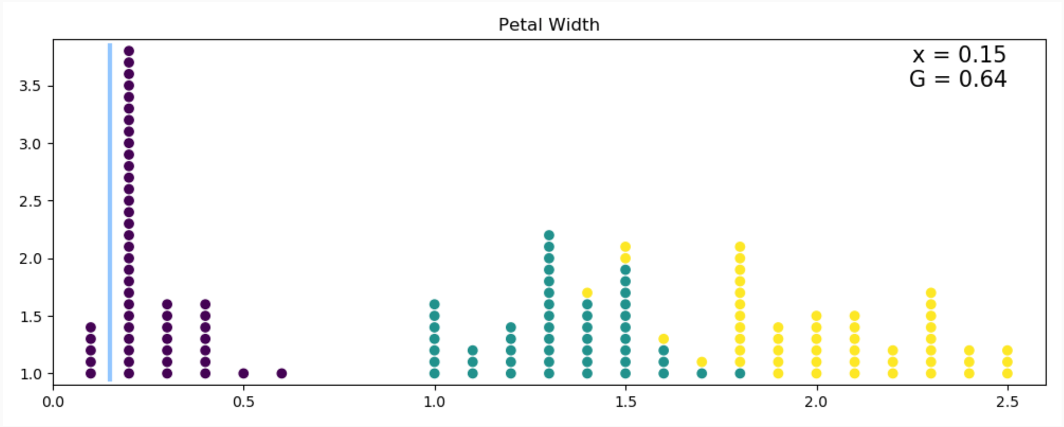
- Decision Trees - Iris Example



- Decision Trees - Iris Example



- Decision Trees - Iris Example



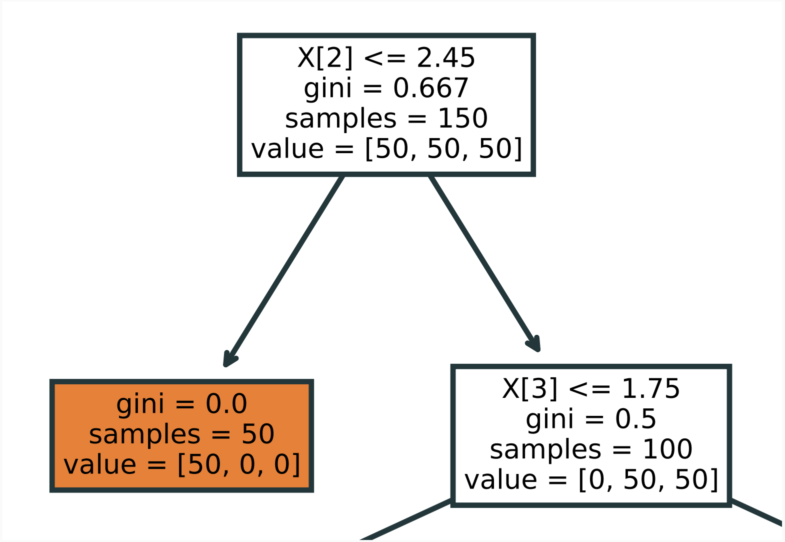
- Decision Trees - Iris Example

- index = 0, Sepal Length $<$ 5.45, Gini = 0.44
- index = 1, Sepal Width $<$ 3.35, Gini = 0.54
- index = 2, Petal Length $<$ 2.45, Gini = 0.33
- index = 3, Petal Width $<$ 0.80, Gini = 0.33

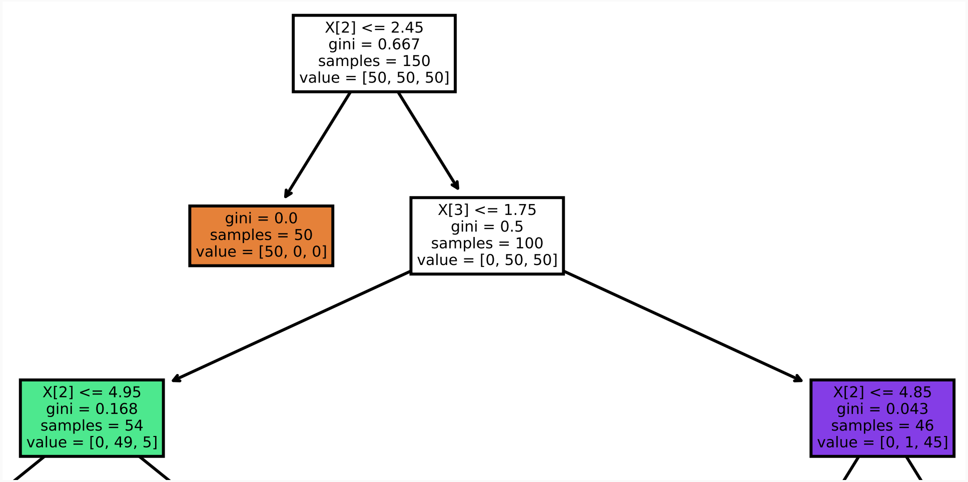
- Decision Trees - Iris Example

- index = 0, Sepal Length $<$ 5.45, Gini = 0.44
- index = 1, Sepal Width $<$ 3.35, Gini = 0.54
- index = 2, Petal Length $<$ 2.45, Gini = 0.33
- index = 3, Petal Width $<$ 0.80, Gini = 0.33

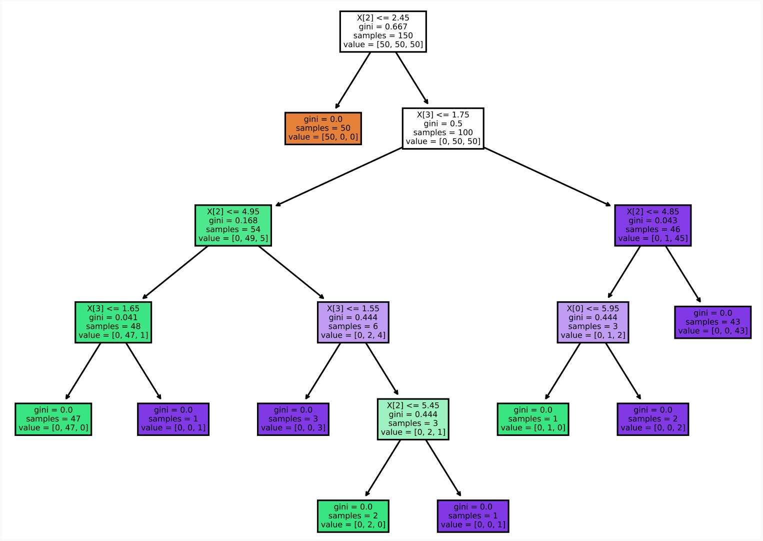
- Decision Trees - Iris Example



- Decision Trees - Iris Example



- Decision Trees - Iris Example



- Decision Trees - Pruning

- Fully trained decision tree is a 100% classifier.
- Fully grown decision tree contains many unnecessary nodes.
- Pruning removes the unnecessary nodes to improve efficiency on the testing dataset.
- Two variants: Prepruning and Postpruning.

- Decision Trees - Pruning

- Pre-pruning:
    - Applied during constructing the tree.
    - Stops split when some condition are satisfied.
- Post-pruning:
    - More frequent variant
    - Two operation: Subtree replacement and subtree raising.

- Decision Trees - Pruning

- Each node is processed and evaluated.
- Sub-tree replacement operation replaces the sub-tree by the leaf node.
- Sub-tree raising operation moves upward a node with its children and replaces the parent nodes. Sibling nodes need to be reclassified.

- Decision Trees - Pruning

- Evaluation of the pruning operation is a major question.
- Simplest possibility if the hold-out set of training objects.
- Hold-out set is set of objects removed from the training data and used strictly in pruning process.
- Other option is to estimate the error produced by the tree and making the changes.

Classification - Covering rules

- Decision Trees works in a divide-and-conquer principle.
- Bottom-up approach may focus on specific classes.
- And covering all instances belonging to the specified class.
- The covering is done using set of rules belonging to a specified class.

Classification - Covering rules

- Covering algorithm add a test to the rule under construction which maximizes the accuracy.
- Searching for the optimal test is similar to the decision trees.
- Suppose the new rule will cover a total of t instances, of which p are positive examples of the class and $t - p$ are in other classes.
- The error rate of the new rule is then p/t.
- The test which maximizes the ration p/t is chosen as best test.

Classification - Covering rules

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Classification - Covering rules

if ? then Play = Yes	
Outlook = Sunny	2/5
Outlook = Overcast	4/4
Outlook = Rainy	3/5
Temperature = Hot	2/4
Temperature = Mild	4/6
Temperature = Cool	3/4
Humidity = High	3/7
Humidity = Normal	6/7
Windy = True	3/6
Windy = False	6/8

Classification - Covering rules

if ? then Play = Yes	
Outlook = Sunny	2/5
Outlook = Overcast	4/4
Outlook = Rainy	3/5
Temperature = Hot	2/4
Temperature = Mild	4/6
Temperature = Cool	3/4
Humidity = High	3/7
Humidity = Normal	6/7
Windy = True	3/6
Windy = False	6/8

Classification - Covering rules

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Classification - Covering rules

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Rainy	Mild	High	True	No

Classification - Covering rules

if ? then Play = Yes	
Outlook = Sunny	2/5
Outlook = Rainy	3/5
Temperature = Hot	0/2
Temperature = Mild	3/5
Temperature = Cool	2/3
Humidity = High	1/5
Humidity = Normal	4/5
Windy = True	1/4
Windy = False	4/6

Classification - Covering rules

$$if\ Humidity = Normal\ AND\ ?\ then\ Play = Yes$$

Outlook	Temperature	Humidity	Windy	Play
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Classification - Covering rules

$$if\ Humidity = Normal\ AND\ ?\ then\ Play = Yes$$

Outlook	Temperature	Humidity	Windy	Play
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

$$if\ Humidity = Normal\ AND\ Outlook = Rainy\ AND\ Temperature = Cool\ AND\ Windy = True\ then\ Play = Yes$$

Classification - Covering rules

- The previous principle is called a PRISM.
- Only 100% accuracy rules are generated.
- The classification works well for non-ambiguous instances.
- All classes are processed separately.
- The rules are evaluated in a ordered manner.

Classification - Credibility and Algorithm evaluation

- Dataset composition:
    - Training data
    - Testing data
    - Validation data
- Not all sets are required.
- Evaluation may be done on the Training data only.

Classification - Credibility and Algorithm evaluation

- The using of the same data for training and testing is not possible due to over-fitting and overestimation.
- The validation part is used for parameter tuning or model solution.
- When the parameter tuning is done, the model is reconstructed on the whole dataset.
- The knowledge from the testing dataset should not be used in parameter tuning.

Classification - Credibility and Algorithm evaluation

- *Accuracy* - the fraction of test instances in which the predicted value matched the ground-truth value.

$$Accuracy = \frac{1}{N}\sum_{i=0}^{N} 1\ if\ (y^i_{pred} == y^i_{truth})\ else\ 0$$

Classification - Credibility and Algorithm evaluation

- *Cost-sensitive accuracy*
  - Not all cases are equally important in all scenarios while comparing the accuracy, e.g. Imbalanced data, ill vs. healthy patients, etc.
  - This is frequently quantified by imposing different costs $c_1, \ldots, c_k$ on the misclassification on the different classes.
  - Let $n_1, \ldots, n_k$ be the number of test instances belonging to each class.
  - Let $a_1, \ldots, a_k$ be the accuracy (expressed as a fraction) on the subset of test instances belonging to each class.
  - The overall accuracy A can be computed as a weighted combination of the accuracy over the individual labels:

$$A = \frac{\sum_{i=1}^{k} c_i n_i a_i}{\sum_{i=1}^{k} c_i n_i}$$

Classification - Credibility and Algorithm evaluation

- *Confusion matrix*

	Ground truth	
Predicted	True	False
True	TP	FP
False	FN	TN

Classification - Credibility and Algorithm evaluation

Holdout

- The labeled data is randomly divided into two disjoint sets (training and testing).
- Typically 60% to 75% is used for training set.
- This partition may be repeated several times to get he final estimation.
- The over-presented samples in the training set are under-presented in the testing sets.
- Due to not using of the whole data set for training the estimation are pessimistic.

Classification - Credibility and Algorithm evaluation

Holdout

- By repeating the process over *b* different holdout samples the mean and the variance of the error estimates may be determined.
- These information may be used for building the confidence intervals on the error.
- In case of imbalanced data, an independent sampling (for each class separately) have to be used to ensure the similarity between whole dataset and the testing dataset.

Classification - Credibility and Algorithm evaluation

Cross-Validation

- The data is divided into m disjoint subsets of equal size n/m.
- A typical choice for m is around 10.
- One segment is used as a testing set the the remaining $m-1$ as a training set.
- This process is repeated by selection each of the m subsets as a testing sets.
- The average accuracy over the m different test sets is reported.
- The size of the training set is $(m-1) * n/m$.

Classification - Credibility and Algorithm evaluation

Cross-Validation

- When m is chosen large, the training set size is close to the whole dataset and the reported prediction is very close to the whole data set.
- The estimate of the accuracy tends to be highly representative but pessimistic.
- A special case is when $m = n$, this is called a *leave-one-out* cross-validation.
- *Stratified cross-validation* uses proportional representation of each class in the different folds and usually provides less pessimistic results.

Classification - Credibility and Algorithm evaluation

Bootstrap

- The labeled data are sampled uniformly with replacement to create a training set that may contain a duplicates.
- The labeled data of size n is sampled n times with replacement.
- The probability that a particular data point is not included in a sample is given by $(1 - 1/n)$
- The probability that the point is not included in n samples is then $(1 - 1/n)^n$.

Classification - Credibility and Algorithm evaluation

Bootstrap

- For large values of n the expression is approximately $1/e$.
- The fraction of labeled points included included at least once in the dataset is $1 - 1/e = 0.632$.
- The training model is constructed on the bootstrapped sample with duplicates.
- The overall accuracy is computed using the whole dataset.
- The estimate is highly optimistic due to large overlap between training and testing set.

Ensemble methods

- The main idea is that different classifiers may make different predictions on test instances with the same train data.
- This is caused by the specific characteristics of the classifiers, their sensitivity to the random artifacts in the data, etc.
- The basic approach is to apply basic ensemble learners multiple times by using different models or the same model on different subsets of data.
- Two basic approaches exists:
  - Data-centered ensembles
  - Model-centered ensembles

Ensemble methods

- Data-centered ensembles
  - Single classification model is used.
  - The dataset is derived into set of subsets.
  - The method of dataset derivation differs - sampling, incorrectly classified data from previous set, manipulation with features, manipulation with class labels, etc.
- Model-centered ensembles
  - Many different algorithms are used in each ensemble iteration.
  - The dataset used by each model is the same as the original dataset.
  - The motivation is that different classifiers works better on particular part of data.
  - This approach is valid as long as the specific errors are not reflected by the majority of the ensembles.

Ensemble methods - Bias

- Every classifier makes its own modeling assumptions about the nature of the decision boundary between classes:
    - The classifier may incorrectly classify data even with large training dataset.
    - The modeled decision boundary does not match the real boundary.
    - Therefore, the classifier has an inherent error - **inherent bias**.
- When a classifier has **high bias**, it will make **consistently incorrect predictions** over particular choices of test instances near the incorrectly modeled

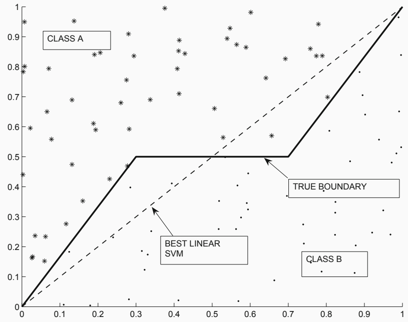


Figure 1: Bias on Linear SVM example

Ensemble methods - Variance

- Random variations in the choices of the training data will lead to different models.
    - Test instances such as X are **inconsistently classified** by decision trees which were created by different choices of training data sets.
    - This is a manifestation of model **variance**.
    - Model variance is closely related to **over-fitting**.
- When a classifier has an over-fitting tendency, it will make inconsistent

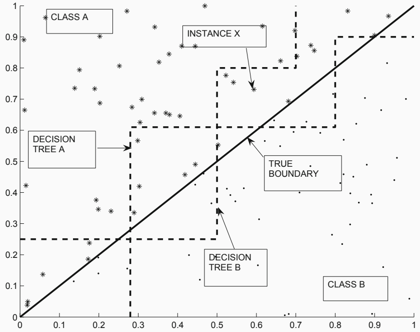


Figure 2: Variance on Decision Tree example

Classification- Bagging

- Also known as bootstrapped aggregation.
- It is focused on variance reduction of the prediction.
- With the variance of the prediction equals to σ^2, the variance of the average of k independent and identically distributed (i.i.d.) prediction is reduced to $\frac{\sigma^2}{k}$.
- The i.i.d. predictors are approximated with bootstrapping (sampling with replacement).

Classification- Bagging

- The k different sets are constructed from the original dataset.
- Each set is used for model training.
- The predicted class is the dominant class over all classifiers.
- This approach decreases the variance, but may increase the bias.
- More detailed models need to be used to reduce bias as well, otherwise, slightly degradation in accuracy may be achieved.
- The i.i.d. is usually not fully satisfied.
- The performance limit of the bagging is done by the pairwise correlation between models ρ

$$\rho \cdot \sigma^2 + \frac{(1-\rho) \cdot \sigma^2}{k}$$

Classification - Bagging - Random Forrest

- Random forests can be viewed as a generalization of the basic bagging method, as applied to decision trees.
- The main drawback of using decision-trees directly with bagging is that the split choices at the top levels of the tree are statistically likely to remain approximately invariant to bootstrapped sampling.
- Therefore, the trees are more correlated, which limits the amount of error reduction obtained from bagging.
- The idea is to use a randomized decision tree model with less correlation between the different ensemble components.
- The final results are often more accurate than a direct application of bagging on decision trees.

Classification - Bagging - Random Forrest

- The *random-split-selection* introduces randomness into split criterion.
- The coefficient $q \leq d$ is used to regulate the randomness.
- The split-point selection is preceded by the random selection of q features.
- Smaller number of q reduces the correlation between different trees but decreases the accuracy.
- Moreover, this improves the construction process because only subset of features need to be investigated.

Classification - Bagging - Random Forrest

- The good trade-off between correlation reduction and accuracy was investigated as
$$q = \log_2(d) + 1$$
- Low-dimension data does not benefit from this approach due to large q with respect to the d.
- The trees are grown without pruning to reduce bias of the prediction.
- Random trees are resistant to noise and outliers and usually better than pure bagging.

Classification - Bagging - Extra Trees

- Slightly different approach is used by the Extra Trees - Extremely Randomized Trees.
- The main changes are focused to increase the variance.
- The data are not sampled using bootstrapping - all data are used for each tree.
- First, the subset of randomly selected features of size q is randomly selected.
- The split of each feature is chosen randomly.
- The best split is selected from the sampled ones.
- Due to two random sampling, trees are really random and less

Classification - Boosting

- In boosting, a weight is associated with each input instance.
- Different classifiers are trained with these weights.
- The weights are modified iteratively based on classification performance.
- Each classifier is constructed using the same algorithm.
- The relative weights are increased on incorrectly classified instances, according to the hypothesis that the misclassification is caused by classifier bias.
- The overall bias is then decreased.

Classification - Boosting

- The predicted class is determined by the weighted aggregation of the particular prediction of each model.
- The primary purpose is to reduce bias of the classification.
- This approach is more sensitive to the noised datasets.
- A typical example is *AdaBoost* algorithm.

Ensemble methods - Boosting - AdaBoost (Adaptive Boosting)

- In binary classification, where labels are from $\{-1; 1\}$.
- The weights are initialized to $\frac{1}{n}$ for each of the n instances.
- The weights are in each iteration updated according the correctness of the prediction.
    - $W_{t+1}(i) = W_t(i)e^{\alpha_t}$ for incorrect classification.
    - $W_{t+1}(i) = W_t(i)e^{-\alpha_t}$ for correct classification.
- The α_t is defined as a function:

$$\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

- where ϵ_t is the fraction of incorrectly classified instances at t-th iterations.

Ensemble methods - Boosting - AdaBoost (Adaptive Boosting)

- The termination criterion are defined as:
  - $\epsilon_t = 0$ - all instances are correctly classified.
  - $\epsilon_t > 0.5$ - the classification is worse than random.
  - User-defined number of iterations is reached.
- The classification of test instance is done using aggregation over all models:

$$y_{pred} = \sum_t p_t \alpha_t$$

- where $p_t \in \{-1; 1\}$ is the prediction in the t-th iteration
- and the α_t is defined as a function:

$$\frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Classification - Boosting - Gradient Boosted Decision Trees

- Uses a Decision Trees as a weak learners.
- A loss function is used to detect the residuals, e.g. mean squared error (MSE) for a regression task and logarithmic loss (log loss) for a classification tasks.
- The existing trees are left unchanged when a new tree is added.
- The new tree is trained on the previous model residual.

Classification - Boosting - Gradient Boosted Decision Trees

- The increasing number of trees may lead in overfitting (this is a difference against the random forest).
- The learning rate affect the speed of learning (small value more robust model).
- Small learning rate requires more trees, more tree leads to overfitting....
- May lead to more precise models than the random forests.
- It is highly sensitive to learning rate and number of learners parameters.
- It is also very sensitive to outliers and noise.

Classification - Boosting - Light Gradient Boosting Machine

- Another gradient boosting algorithm that utilizes decision trees.
- The trees used grows leaf-wise - the leaf with maximal error is grown to achieve better results.
- The features are selected according the it nature - sparse features are combined.
- Designed to process large dataset with many features.
- Contains more than 100 parameters that may be tuned.

Ensemble methods - Bucket of models

- An method that combines several different algorithms together and removes the necessity of *a priori* selection of the particular classification algorithm.
- The dataset is divided into two subsets *A* and *B* (a hold-out principle).
- Each algorithm is trained on the *A* set and evaluated on *B* set.
- The best algorithm is selected as a winner and then it is retrained on the complete dataset.
- A cross-validation may be used instead of hold-out principle.
- Different algorithm may be represented by the same algorithm with different parameters.
- Due to *winner-take-all* principle, the best found classifier is selected.
- This approach reduces both bias and variance but it is limited by the

Ensemble methods - Stacking

- Stacking is a two-level classification approach.
- Several algorithm are used for classification.
- The dataset is divided into two subsets A and B (a hold-out principle).
- First level:
    - Training of the k different classifier (ensemble components) on the set A.
    - These components are generated using:
        - bagging,
        - k-rounds boosting,
        - k different decision tress,
        - k heterogeneous classifiers.

Ensemble methods - Stacking

- Second level:
    - Determine the k outputs of each trained classifier on a set B.
    - Create a new set of k features from these outputs.
    - The class label is known from the ground-truth data.
    - Train a classifier on this new representation of the set B.

Ensemble methods - Stacking

- Sometimes, the original features of B are combined with k generated features from the first level.
- The class predictions may be replaced with class probabilities.
- A m-way cross-validation may be used on the first level, where only $(m-1)$ folds are used for training and the second level classifier is trained on whole dataset.
- This approach is very flexible and reduces both bias and variance.
- Other ensemble approaches may be viewed as special cases of Stacking (i.e. majority voting in second level, etc.).

Questions