UNIVERSITÄT OSNABRÜCK

Decision Trees and Random Forests

Kai-Uwe Kühnberger
University of Osnabrück
22.01.2021

UNIVERSITÄT OSNABRÜCK

Machine Learning Methodology: Schedule

- Overview Machine Learning Sessions
  - Basics (last week)
    - Machine Learning
    - Important Concepts
    - Clustering Methods
    - Properties of Hypotheses
  - Classification Methods (today)
    - Support Vector Machines
    - Example: Document Classification
  - Classification Methods (next week)
    - Decision Trees
    - Random Forests
  - Literature

Categorization Methods

Decision Trees

Decision Trees

- A *decision tree* is a classifier in the form of a tree structure
- There are two types of nodes
  - Decision nodes
    - Each decision node is labeled with an attribute name
    - Each arc is labeled with a value for the node attribute
  - Leaf nodes
    - Each leaf node is labeled with a class / category name

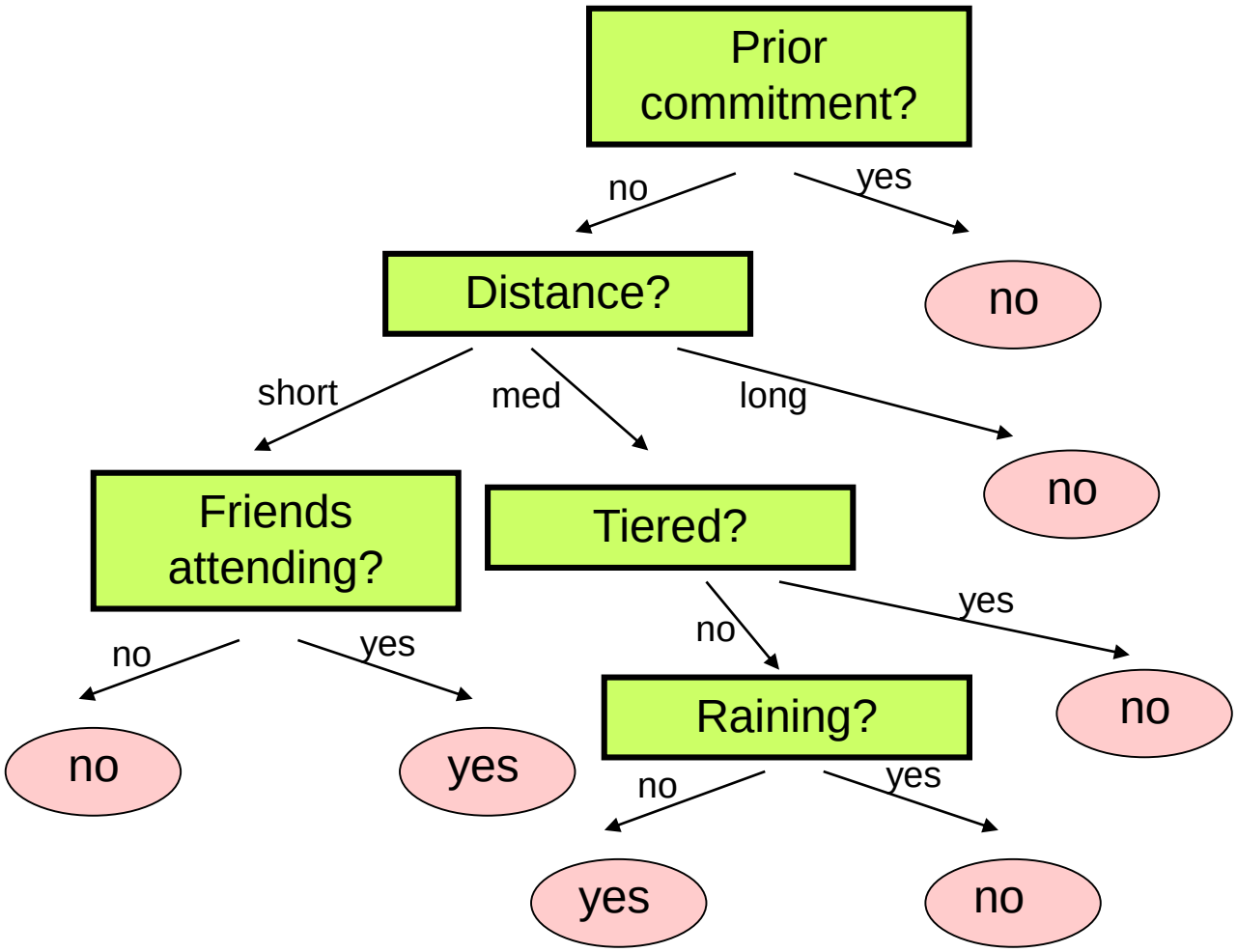
- Example: Whether or not we will go to a party depends on the following five attributes

  - ***Distance*** - distance of the party (**short, medium, long**)
  - ***Friends*** - whether or not any of our friends are going (**yes, no**)
  - ***Prior commitment*** - whether or not we have any prior plans (**yes, no**)
  - ***Rain*** - whether or not it is raining (**yes, no**)
  - ***Tired*** - whether or not we are tired (**yes, no**)

Decision Trees

prior commitment	friend attending	raining	tired	distance	go to party
no	yes	no	no	short	yes
no	yes	no	yes	medium	no
yes	no	yes	no	long	no
yes	yes	yes	yes	short	no
no	yes	no	no	short	yes
no	no	no	no	medium	yes
yes	no	yes	no	long	no
yes	yes	no	yes	short	no
no	yes	yes	no	short	yes
no	no	yes	no	medium	no
no	no	yes	no	long	no
yes	no	yes	yes	short	no

Decision Trees



Remarks

Goal predicate: *attend-party*

Decision trees implicitly define logical statements (conjunctions of implications) like:

$\forall P$ *attend-party*(P) if *not prior*(P) & *dist*(P,*short*) & *friends*(P)

Decision Trees: Learning Algorithm

- Start with a set of examples (training set), set of attributes SA, default value
- for goal predicate, do
  - If the set of examples is empty, then add a leaf with the default value for the goal predicate and terminate, otherwise
  - If all examples have the same classification, then add a leaf with that classification and terminate, otherwise
  - If the set of attributes SA is empty, then return the default value for the goal predicate and terminate, otherwise
  - *Choose* an attribute A to split on (next slide)
    - Add a corresponding test to the tree
    - Create new branches for each value of the attribute
    - Assign each example to the appropriate branch
  - Continue on each branch, with the set of attributes SA-{A} and a default value for this branch corresponding to the majority value for the current set of examples

Decision Trees: ID3

- Which attribute should be chosen?
  - Choose the attribute with the highest information gain on the training set

- Informal formulation of ID3
  - Determine the attribute with the highest information gain on the training set
  - Use this attribute as the root of the tree, create a branch for each of the values that the attribute can take
  - For each of the branches, repeat this process with the subset of the training set that is classified by this branch

- ID3 performs (approximately) a top-down, breadth-first, greedy search through the search space.
- Its inductive bias is:
  - Shorter trees are preferred over longer trees
  - Trees that place high information gain attributes close to the root are preferred over those that don't

Information Gain

- Assume the following is given
  - S: training data set
  - c: number of target classes
  - p_i: proportion of examples in S belonging to target class i
- Entropy

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

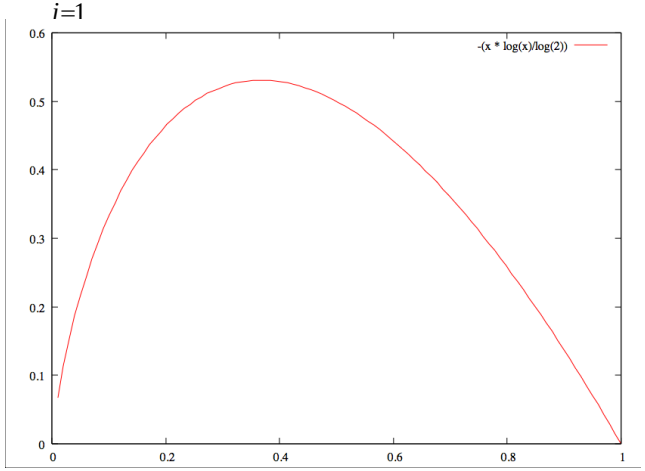
- Remarks
  - Information theoretic interpretation: number of bits required to encode the classification of an arbitrary member of S (measure of uncertainty)
  - If all instances in S belong to the same class, then $E(S) = 0$
- Example
  - Out of 14 instances, 9 are classified as yes, 5 as no
  - $p_{yes} \log_2(p_{yes}) = - (9/14) \log_2(9/14) = 0.41$
  - $p_{no} \log_2(p_{no}) = - (5/14) \log_2(5/14) = 0.53$
  - $E(S) = p_{yes} \log_2(p_{yes}) + p_{no} \log_2(p_{no}) = 0.94$

Entropy

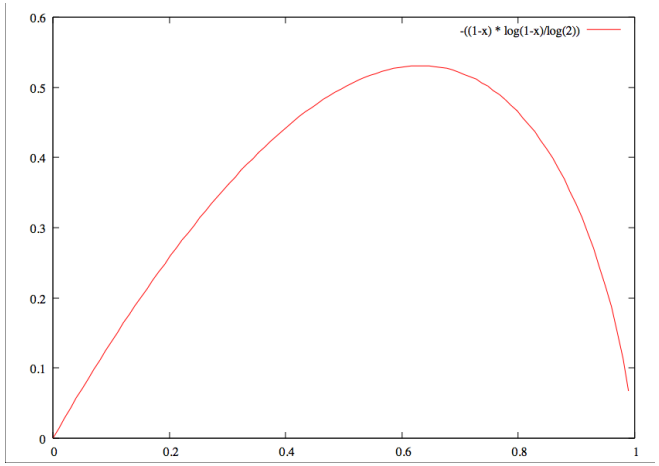
$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

for $c = 2$:

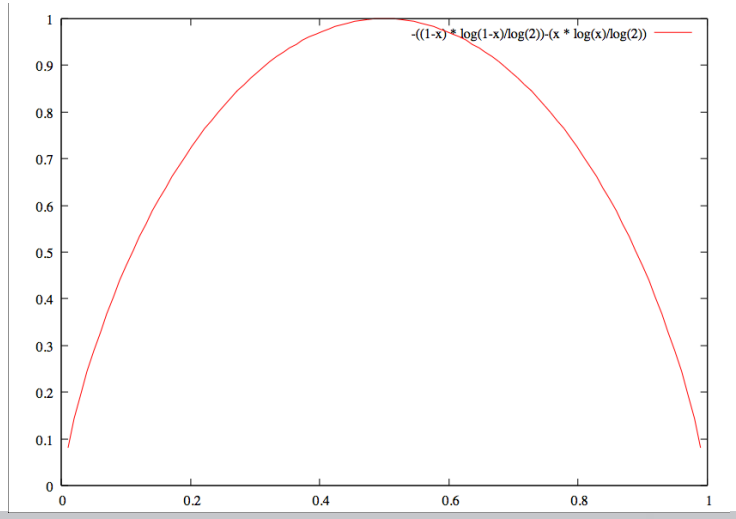
$$E(S) = -(p_1 \log_2 p_1 + (1 - p_1)\log_2(1 - p_1))$$



+



=



(number of bits to code the information)

Information Gain

- Assume the following is given
  - *Values*(*A*): set of all positive values of an attribute *A*
  - S_v: subset of *S* for which *A* has value *v*
  - |*S*|: size of *S*, $|S_v|$: size of S_v
- The information gain *Gain*(*S*,*A*) is the *expected reduction in entropy* caused by knowing the value of attribute *A*

$$Gain(S,A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$$

- Using the formula to calculate the information gain, we can determine the attribute with the highest information gain
- This is the attribute with the smallest sum of entropies of the partitioned data!

UNIVERSITÄT OSNABRÜCK

prior commitment	friend attending	raining	tired	distance	go to party
no	yes	no	no	short	yes
no	yes	no	yes	medium	no
yes	no	yes	no	long	no
yes	yes	yes	yes	short	no
no	yes	no	no	short	yes
no	no	no	no	medium	yes
yes	no	yes	no	long	no
yes	yes	no	yes	short	no
no	yes	yes	no	short	yes
no	no	yes	no	medium	no
no	no	yes	no	long	no
yes	no	yes	yes	short	no

Information Gain

- Compute the information gain for the attributes *raining* and *tired*
  - Calculation of $E(S)$
    - $E(S) = -(4/12)\log_2(4/12) + (-1)(8/12)\log_2(8/12) = 0.918$
  - Calculation of $E(S_{yes})$ and $E(S_{no})$ for attribute *tired*
    - $E(S_{yes}) = -(0/4)\log_2(0/4) + (-1)(4/4)\log_2(4/4) = 0$
    - $E(S_{no}) = -(4/8)\log_2(4/8) + (-1)(4/8)\log_2(4/8) = 1$
  - Now we calculate *Gain*(*S*,*tired*)
    - $Gain(S,tired) = E(S) - (|S_{yes}| / |S|)E(S_{yes}) - (|S_{no}| / |S|)\, E(S_{no})$
    - $= 0.918 - (4/12 \cdot 0) - (8/12 \cdot 1)$
    - $= 0.251$
  - Similarly we calculate *Gain*(*S*,*raining*)
    - $Gain(S,raining) = E(S) - (|S_{yes}| / |S|)E(S_{yes}) - (|S_{no}| / |S|)\, E(S_{no})$
    - $= 0.918 - (7/12)\ 0.592 - (5/12)\ 0.971$
    - $= 0.168$
  - *Tired* provides greater information gain than *raining* relative to the target *go to party*
    - Hence we would choose *tired* instead of *raining* in the ID3 algorithm

Some further Remarks on Decision Trees

- ID3 was proposed in Quinlan (1986) and is still a state-of-the-art learning algorithm
  - The “modern” version of ID3 is called C4.5
  - C4.5 is a benchmark machine learning algorithm

- There is an incremental version of decision trees called CAL2 proposed by Unger and Wysotzki (1981)
  - Motivation: Psychological Findings
    - Humans start with a simple hypothesis
    - They modify the hypothesis if new examples arrive
  - Differences to ID3
    - The algorithms builds the tree depending on the order of incoming examples (ID3 operates on whole training set)
    - This yields usually larger decision trees
    - Therefore pruning strategies need to be developed

Categorization Methods

Random Forests

Motivation for Random Forsts

- Decision trees with discrete sets of values for target variables are called classification trees.
  - In such decision trees, leaves represent class labels and branches represent conjunctions of features (see slides above).
- Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.
- Decision trees / regression trees have some nice features:
  - Robustness relative to irrelevant features
  - Models are transparent (white box model)
- Decision trees have also some suboptimal features:
  - Deep decision trees tend to learn irregular patterns (overfitting).
  - A possible solution for this problem are random forests.

Random Forests

- Idea:
  - Train many different decision trees on subsets of the training set while randomly varying certain aspects of the training for each tree.
  - Each decision tree votes for a classification. The class getting the majority of votes is selected (in the classification case).
- Why are random forests often better in performance than decision trees?
  - There are multiple theoretical reasons for this.
  - One reason: individual classifiers in the ensemble should be both **accurate** and **diverse**. In this context, a classifier is **accurate** if it performs better than random guessing and multiple classifiers are **diverse** if they make different errors for new examples.
  - To build a good ensemble, one needs a way to make sure that individual classifiers are substantially different from one another while still having at least some accuracy on their own.
  - The classical method for doing this is **tree bagging**.

Random Forests

- **Tree Bagging**
  - Bagging stands for **B**ootstrap **agg**regat**ing** (Breiman, 1996)
  - The decision trees are not trained on the whole training set, but on different subsets of the training set.
  - Bagging uses sampling of a new training set with replacement.
    - Sampling without replacement is also possible.
  - More formally: If D is a training set of size n, bagging generates m new training sets D_i of size n' by sampling from D with replacement.
  - One possible choice is to take the same size of the bagged sets as the original set: i.e., on average, 63.2 % of all samples will be present at least once in the new set (the others are duplicates). If a smaller size of the bagged sets is chosen, this is called sub-bagging.

Random Forests

- **Feature Subset Selection**
  - To get a random forest by bagging **feature subset selection** is used.
  - Training of individual trees does not use all available features at each split. Instead, for each new split in consideration, some features are randomly selected and presented to the decision tree algorithm.
  - The number of features randomly selected is controlled by a parameter.
  - For a classification problem with p features, often $p^{1/2}$ (rounded down) features are used in each split. For regression problems the inventors recommend $p/3$ (rounded down) with a minimum node size of 5 as the default.

Random Forests: Hyperparameters

Unfortunately, random forests have a rather large list of possible parameters. The following table is taken from a Bachelor Thesis that was recently written at our institute.

Name[14]	Controls	SMILE default value	[Breiman 2001]
nTrees	How many decision trees will be grown.	No default	100
mtry	How many features will randomly be selected at each split.	$\sqrt{nFeatures}$	$1 \mid log_2(nFeatures) + 1$
subsample	How big the bagged set of samples is for each tree (as a multiple of the original number of samples)	1.0	1.0
nodeSize	How many examples a node must at least have for a further split to occur.	5	1
maxNodes[15]	How many leaf nodes a tree is allowed to contain before training is shut down.	100	Infinitely many
rule	Which heuristic to use for determining the goodness of a split.	Gini index	Gini index

- Choosing good parameters for random forests is sometimes a rather hard task.
- The choice is often domain dependent and is figured out by trial and error.

Random Forests: Hyperparameters

- Here is another list of parameters based on a Master thesis.

- **n_estimators (integer):** number of trees (estimators) in the forest.
- **bootstrap (boolean):** whether the samples used to building the trees are chosen with replacement.
- **max_features (string, float, int):** amount of features to consider for a split. If *None* is chosen then $max\_features = n\_features$, if *auto* is chosen then $max\_features = \sqrt{n\_features}$ which is the same as choosing *sqrt*.
- **class_weight (balanced or None):** whether the classes are weighted or not. If balanced is chosen then the classes are weighted inversely proportional to class frequencies with this formula: $\frac{N_s}{N_c \times nc_i}$. Where N_s is the number of samples, N_c is the number of classes and nc_i is the number of samples of class i.
- **min_samples_split (int or float):** minimum number of samples to split an internal node.
- **criterion (gini or entropy):** function to measure the degree of purity of the split. The formula to calculate gini is: $1 - \sum_j p_j^2$; while the formula for entropy is: $\sum_j -p_j \log_2 p_j$. Where p_j is the frequency or probability of class j in the sub-samples.

Random Forests

- Many machine learning libraries contain open source implementations of random forests:
  - There are libraries for R, Matlab, Python etc.
- Random forests show very good performance in many machine learning applications.
  - Examples are:
    - Document classification (several Bachelor and Master Theses were written in this context)
    - Bioinformatics
    - Video analysis
    - Image classification
    - Etc.

Further Remarks

- Different algorithms use different metrics for measuring the "best“ variable to perform a split.
  - We discussed in more detail the concept of information gain.
  - An alternative is Gini impurity
    - Gini Impurity can be understood as a criterion to minimize the probability of misclassification
    - Gini impurity: it is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- Remarks
  - History
    - ID3 was proposed in Quinlan (1986)
    - CART (regression tree learning) was proposed 1983 by Breiman
    - Random forests were proposed by Ho (1995) and later developed by Breiman and Cutler, e.g. Breiman (2001)

Bibliography

- Support Vector Machines
  - Schölkopf, Smola: Learning with Kernels, MIT Press, 2001.
  - Vapnik and Lerner, “Pattern Recognition Using Generalized Portrait Method,” Automation and Remote Control, Vol. 24, 1963, pp. 774-780.
  - Vapnik, Vladimir N (2000). The Nature of Statistical Learning Theory. Information Science and Statistics.
  - Geibel, Gust, Kühnberger (2007): Variants of Tree Kernels for XML Documents. In: A. Gelbukh and A.F. Kuri Morales (Eds.): MICAI 2007, LNAI 4827, pp. 850–860, 2007.
- Decision Trees
  - Quinlan (1986): Ross Quinlan: Induction of Decision Trees. Machine learning 1.1. vol. 1(1), August 1985, pp. 81–106.
  - L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone: CART: Classification and Regression Trees. Wadsworth: Belmont, CA, 1983.
  - Unger & Wysotzki, F. (1981): Lernfähige Klassifizierungssysteme (Classification Systems Being Able to Learn); Berlin, Germany: Akademie-Verlag

Bibliography

- Random Forest
  - Ho (1995): Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.
  - Breiman (2001): "Random Forests". Machine Learning. 45(1):5–32. doi:10.1023/A:1010933404324
  - Breiman, Leo (1996). "Bagging predictors". Machine Learning. 24 (2): 123–140.
- Similarities
  - Quesada, J. (2008). Human Similarity Theories for the Semantic Web. In: Nature inspired Reasoning for the Semantic Web, 2008.
  - Tversky (1977): Features of Similarity; Psychological Review 84(4):327-352.
- General Textbook for Machine Learning
  - Mitchell, T (1997). Machine Learning, McGraw-Hill, 1997.
  - Bishop, C. (2008). Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, Berlin 2008.