Introduction to Machine Learning - Decision Trees

Kairit Sirts

07.02.2014

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell, 1997

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.

Examples

- Email spam filtering
- Handwriting recognition
- Face detection and recognition
- Speech recognition
- Machine translation
- Computer games (Kinect, TrueSkill player ranking system)
- Recommender systems (Amazon)
- Ranking search results
- Fraud detection
- ... and many more

Goal of this course

- Provide an overview about the main methods in modern machine learning (lectures)
- Give some experience in applying these methods in practice (homeworks)
- Exemplify the usefulness of all the math you've learned so far
- Hopefully make you excited about this field.

Course at a glance

- Decision trees
- Nearest neighbours classification
- K-means classification
- Perceptron
- Neural networks
- Linear and polynomial regression
- Logistic regression
- Support vector machines
- Naïve Bayes
- Markov models
- Dimensionality reduction methods
- Reinforcment learning

Course organization

- Lectures every week (Friday at 16:00)
- 7 homeworks: programming assignment plus short write-up, assigned every other week.
- Practical every Friday after lecture – no organized activity but I'm staying around and can advise with homeworks.
- Homeworks give 10% of the final grade each.
- Final exam (written) will give the rest 30%.
- Course web-page: `http://courses.cs.ttu.ee/pages/Machine_learning`
- Assignment submission via moodle.

Any questions so far?

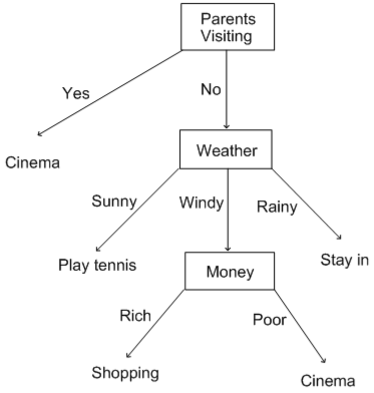
Decision trees example

Suppose we want to decide what to do based on the following information:

1. If parents are visiting then go to cinema.
2. If they're not visiting and it's sunny then play tennis ...
3. but if it's windy and have money then go shopping ...
4. but if no money then go to cinema.
5. If parents are not visiting and it's rainy then stay at home.

Decision trees example

We can draw a tree to make these decisions - a **decision tree**:



http://www.doc.ic.ac.uk/~sgc/teaching/pre2012/v231/lecture11.html

Data

- Internal tree nodes are questions.
- Leaves represent the answers - **labels** or **classes**.
- Data is usually given in tabulated form.
- Each row represents one data point or item.
- Questions in the rows are called **features** or **attributes**.
- Specific answers to each question are called **feature values**.

Parents	Weather	Money	Activity
yes	sunny	yes	cinema
no	windy	yes	shopping
no	sunny	no	tennis
yes	windy	yes	cinema
no	rainy	no	stay at home
...	...	...	...

Some more terminology

- This is a **classification** probem - the labels are discrete.
- The task with continuous labels is called **regression** problem.
- When we learn from labelled examples then it is **supervised learning** problem.
- When no labels are given then it is **unsupervised learning** problem.

Decision tree learning

- What questions to ask?
- In what order to ask?
- Which label to predict?

Let's assume a **binary decision tree**:

- Each question has two answers (yes or no).
- With **m** features, how many questions can we ask?
- How many possible orderings there are?
- What is the best ordering?
- Do we have to ask all the questions?

We cannot try out all possible trees most of the times.

How do we learn?

- We need **labelled training data** – data items with their correct answers.
- From this we must construct questions and the ordering.
- We can use the **greedy heuristic**:
  - **If I could ask only one question then what would it be?**
- This enables to find the most useful feature for guessing the answer.

How to greedily select the feature?

With each feature repeat the procedure:

- Divide the data into sets based on feature values.
- Construct the histogram of answers for each set.
- Compute a score of how useful is this feature in predicting answers.

For computing the score:

- Assing to each set it's majority label based on the histogram.
- Use this assignment to label each item in the training set.
- Compute the accuracy of the labelled data.

Choose the feature with the highest score.

Choosing features

- The first chosen feature is the **root** of the decision tree.
- Continue by using **divide and conquer** strategy: for each set obtained by choosing the feature choose the next feature from the remaining features.
- How long should we continue? When should be stop dividing?
  - No more features left.
  - When some feature at some point in the tree classifies the data 100% accurately.
  - When some other stopping criterion is met.

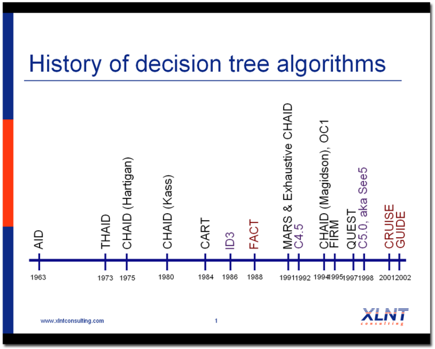
Example - when to play tennis?

Let's take adata set with two labels - **binary classification problem**.

outlook	temperature	humidity	wind	play
sunny	warm	high	weak	no
sunny	warm	high	strong	no
rain	warm	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
sunny	cool	normal	strong	yes
sunny	warm	high	weak	no
sunny	cool	normal	weak	yes
rain	warm	normal	weak	yes
sunny	warm	normal	strong	yes
rain	warm	high	strong	yes
sunny	warm	normal	weak	yes
rain	warm	high	strong	no

A little bit about history

- One of the oldest and simplest machine learning models.
- Introduced in sixties, main methods developed in eighties.
- Many methods proposed proposed since then: ID3, C4.5, CART, CHAID, MARS



http://www.xlntconsulting.com/newsletter-archive/history-of-decision-trees-algorithms.htm

Advantages of decision trees

- Decision trees are self-explanatory, they can be converted to a set of rules also understandable to non-professionals.
- They can handle both numeric and nominal inputs.
- They make no assumptions about space distribution and classifier structure.
- Decision trees can handle missing data (some attribute values are missing).
- The representation is rich enough to represent any discrete-value classifier.

Some disadvantages

- Decision trees can be quite unstable - small variations in the data can result in a completely different tree.
- As optimal learning is intractable then heuristics (for example greedy) must be used. This means that at each node locally optimal decisions are made and this does not guarantee globally optimal result.
- There are concepts that are hard to learn, leading to overly complex trees.

Evaluating the results

How do we know how well we are doing?

- **Training data** - why can't we use it to evaluate the resulting tree?
  - Because we used it to train this tree at first hand.
- We need **test data**: originating from the same source as training data.
- **Don't use the test data when still developing the system, use it only for final evaluation**
- How to evaluate the system during development?
  - We need **development data**: from the same source as training and test data.

Data splitting

- Usually the data is split first into training, development and test sets.
  - For example 80/10/10 or 80/20 when no development data is needed
  - or 70/10/20 or 90/10.
  - The split depends on how much training data is available (how much can we spare for testing and development).
- An alternative to separating development data is to use **k-fold cross-validation**.

Cross-validation

- Divide the training data (after removing test data) randomly into **k folds**.
- Run k experiments:
  - Compose the training data by concatenating k-1 folds leaving one fold out in turn
  - Train the model on those k-1 folds
  - Test it on the left-out fold
  - Record the result
- Report the average of the k experiments.

Models and parameters

- Here we attempted to solve the task of **classificaton**
- We decided to use a particular type of **model** - decision trees.
- Usually, models have **parameters**.
- **Training** a model means learning or **inferring** the parameters from the training data.
- Parameters in the decision tree include:
    - specific questions asked;
    - the order of the questions asked;
    - classification decisions at the leaves.

Hyperparameters

Models may also have **hyperparameters**:

- These are like parameters but usually not learned from the training data
- Learned parameters depend on the hyperparameters
- They are usually set by hand or tuned on development data
- A hyperparameter for the decision trees can be the **maximum depth** of the tree.

Underfitting and overfitting

- Basically we learn a function (currently in the form of a tree).
- **Underfitting** occurs if the learned function is too simple:
  - Decision tree is empty, we do not ask any questions, just give all items a predefined label - constant function.
- **Overfitting** occurs if the learned function is too complex and does not generalize well to new data
  - Decision tree is full, all questions are asked in each branch.
  - When there is no training data at some branch, we must choose labels in leaves arbitrarily
  - This tree scores well on training data, what about development or test data?

What is learning anyway?

- If a student learns the solutions of the exercises solved in class but fails to solve slighly different but analogous problems in exam, did he really learn anything?
- If the model classifies all the training examples correctly but does terribly on test data, dit it really learn anything?
- **Memorizing** versus **learning**
- Learning enables to generalize to new data, simply memorizing does not.
- When we overfit then we rather memorize than learn.
- When we underfit then we learn too little.

Cost function

- **Cost funcion** or **loss function** tells us how badly the model is working - the worse the model the higher the loss or cost.
- Learning should make the cost smaller.
- The form of the cost function depends on the model and on the problem.
- For binary classification the cost can be 1 or 0, depending whether the item was classified correctly or not.
- Previsously we maximized the accuracy of the decision made by a specific feature, this is equivalent to minimizing **misclassification rate**.

More formally

We want to choose a feature j^* for subset S of data among the set of features F:

$$j^*(S) = \arg\min_{j \in F} \text{cost}(\{\mathbf{x}_i, y_i : \mathbf{x}_i \in S, x_{ij} = c_k\}) \\ + \text{cost}(\{\mathbf{x}_i, y_i : \mathbf{x}_i \in S, x_{ij} \neq c_k\})$$

Probability that item x_i with label y_i belongs to class c:

$$\hat{\pi}_c = \frac{1}{|S|} \sum_{x_i \in S} \mathbf{1}\{y_i = c\}$$

Most probable class label:

$$\hat{y} = \arg\max_c \hat{\pi}_c$$

Misclassification rate (one possible cost function):

$$\frac{1}{|S|} \sum_{x_i \in S} \mathbf{1}\{y_i \neq \hat{y}\} = 1 - \hat{\pi}_{\hat{y}}$$

Another cost function - entropy

- Entropy is an information-theoretic measure that measures uncertainty.
- The bigger the entropy the more uncertain we are.
- When we know for sure then the entropy is zero.
- When we are completely ambivalent (uniform distribution over classes) then the entropy has maximal value.
- Entropy is computed over **probability distribution**.
- We want to minimize entropy in order to minimize uncertainty in our decisions.

$$H(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi}_c \log_2 \hat{\pi}_c$$

More popular cost functions

- Minimizing entropy is equivalent to maximizing **information gain** between test $X_j = k$ and class label Y.

$$\text{infoGain}(X_j = k, Y) = H(Y) - H(Y|X_j)$$

- **Gini index** is the expected error rate:

$$\sum_{c=1}^{C} \hat{\pi}_c(1 - \hat{\pi}_c) = \sum_{c=1}^{C} \hat{\pi}_c - \sum_{c=1}^{C} \hat{\pi}_c^2 = 1 - \sum_{c=1}^{C} \hat{\pi}_c^2$$

Cost functions in two-class case

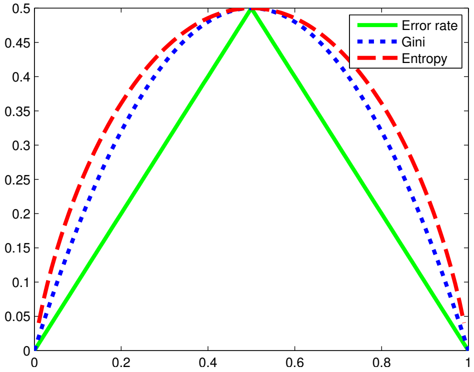


Figure 16.3 from Machine Learning: A Proabilistic Approach (Murphy)

Pruning

- To prevent overfitting stop growing the tree when the decrease in error is below some threshold.
- Often none of the splits produces a significant reduction in error, but after several splits a substantial error reduction is found.
- Thus a more common approach is to grow a big tree:
  - full tree,
  - use a stopping criterion based on number of data points in leaves.
- Then prune the tree by eliminating the branches giving the least increase in the error.