COMPSCI762: Foundations of Machine Learning

Decision Trees

Jörg Simon Wicker

The University of Auckland

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SCIENCE
SCHOOL OF COMPUTER SCIENCE
MACHINE LEARNING

Today we will cover...

Partially based on Slides from University of British Columbia

Motivating Example

Motivating Example: Food Allergies

- You frequently start getting an upset stomach and suspect an adult-onset food allergy
- To solve the mystery, you start a food journal

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	...	Sick?
0.0	0.7	0.0	0.3	0.0	0.00	...	1
0.3	0.7	0.0	0.6	0.0	0.01	...	1
0.0	0.0	0.0	0.8	0.0	0.00	...	0
0.3	0.7	1.2	0.0	0.1	0.01	...	1
0.3	0.0	1.2	0.3	0.1	0.01	...	1

Motivating Example: Food Allergies

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	...	Sick?
0.0	0.7	0.0	0.3	0.0	0.00	...	1
0.3	0.7	0.0	0.6	0.0	0.01	...	1
0.0	0.0	0.0	0.8	0.0	0.00	...	0
0.3	0.7	1.2	0.0	0.1	0.01	...	1
0.3	0.0	1.2	0.3	0.1	0.01	...	1

- What can we learn from this?
- It is hard to find the pattern
  - You can't isolate and only one food at a time
  - You may be allergic to more than one food
  - The quantity matters: a small amount may be OK
  - You may be allergic to specific interactions

Supervised Learning

Supervised Learning

- We can formulate this as a supervised learning problem

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	...		Sick?
0.0	0.7	0.0	0.3	0.0	0.00	...		1
0.3	0.7	0.0	0.6	0.0	0.01	...	$\Rightarrow$	1
0.0	0.0	0.0	0.8	0.0	0.00	...		0
0.3	0.7	1.2	0.0	0.1	0.01	...		1
0.3	0.0	1.2	0.3	0.1	0.01	...		1

- Input for an example (day of the week) is a set of features (quantities of food)
- Output is a desired class label (whether or nor we got sick)
- Goal of supervised learning
  - Use data to find a model that outputs the right label based on the features
  - Model predicts whether foods will make you sick (even with new combinations)

Supervised Learning

- General supervised learning problem:
  - Take features of examples and corresponding labels as inputs
  - Find a model that can accurately predict the labels of new examples
- This is the most successful or widely used machine learning technique
  - Spam filtering, optical character recognition, speech recognition, classifying tumours, etc.
- We'll first focus on categorical labels, which is called classification
  - The model is a called a classifier

Naïve Supervised Learning: Predict Mode

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	...		Sick?
0.0	0.7	0.0	0.3	0.0	0.00	...	$\Rightarrow$	1
0.3	0.7	0.0	0.6	0.0	0.01	...		1
0.0	0.0	0.0	0.8	0.0	0.00	...		0
0.3	0.7	1.2	0.0	0.1	0.01	...		1
0.3	0.0	1.2	0.3	0.1	0.01	...		1

- A very naïve supervised learning method?
  - Count how many times each label occurred in the data (4 vs. 1 above)
  - Always predict the most common label, the "mode" ("sick" above)
- This ignores the features, so is only accurate if we only have 1 label
- We want to use the features, and there are MANY ways to do this
  - First, let's look at Decision Trees

Decision Trees

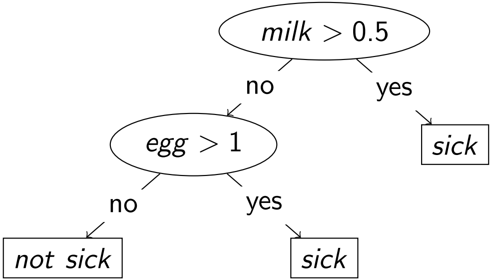
Decision Trees

- Decision trees are simple models consisting of
  - A nested sequence of “if-else” decisions based on the features (splitting rules)
  - A class label as a return value at the end of each sequence

- Example decision tree

if $milk > 0.5$ **then**
    return *sick*
else
    if $egg > 1$ **then**
        return *sick*
    else
        return *not sick*
    end
end

- Can draw sequences of decisions as a tree



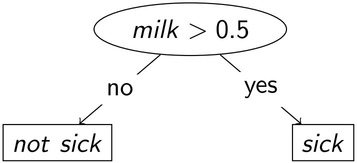
Supervised Learning as Writing a Program

- There are many possible decision trees
  - We're going to search for one that is good at our supervised learning problem
- So our input is data and the output will be a program
  - This is called "training" the supervised learning model
  - Different than usual input/output specification for writing a program
- Supervised learning is useful when you have lots of labeled data BUT:
  1. Problem is too complicated to write a program ourselves,
  2. Human expert can't explain why you assign certain labels, OR
  3. We don't have a human expert for the problem.
- So how would you train a decision tree?

Decision Stumps

Learning A Decision Stump: "Search and Score"

- We'll start with "decision stumps"
  - Simple decision tree with one splitting rule based on thresholding one feature



- How do we find the best "rule" (feature, threshold, and leaf labels)?
  1. Define a 'score' for the rule
  2. Search for the rule with the best score
- What would you suggest as a score?

Learning a Decision Stump: Accuracy Score

- Maybe most intuitive score: classification accuracy
  - "If we use this rule, how many examples do we label correctly?"
- Computing classification accuracy for $(egg > 1)$:
  - Find most common labels if we use this rule:
    - When $(egg > 1)$, we were "sick" 2 times out of 2
    - When $(egg \leq 1)$, we were "not sick" 3 times out of 4
  - Compute accuracy:
    - The accuracy ("score") of the rule $(egg > 1)$ is 5 times out of 6
- This "score" evaluates quality of a rule
  - We "learn" a decision stump by finding the rule with the best score.

Egg	Milk	Fish	...	Sick?
1	0.7	0.0	...	1
2	0.7	0.0	...	1
0	0.0	1.2	...	0
0	0.7	1.2	...	0
2	0.0	1.3	...	1
0	0.0	0.0	...	0

Learning a Decision Stump: By Hand

- Let's search for the decision stump maximizing classification score:
  - First we check "baseline rule" of predicting mode (no split): this gets 3/6 accuracy
  - If ($milk > 0$) predict "sick" (2/3) else predict "not sick" (2/3): 4/6 accuracy
  - If ($fish > 0$) predict "not sick" (2/3) else predict "sick" (2/3): 4/6 accuracy
  - If ($fish > 1.2$) predict "sick" (1/1) else predict "not sick" (3/5): 5/6 accuracy
  - If ($egg > 0$) predict "sick" (3/3) else predict "not sick" (3/3): 6/6 accuracy
  - If ($egg > 1$) predict "sick" (2/2) else predict "not sick" (3/4): 5/6 accuracy
- Highest-scoring rule: ($egg > 0$), then "sick", else "not sick"
- Notice we only need to test feature thresholds that happen in the data
  - There is no point in testing the rule ($egg > 3$), it gets the "baseline" score
  - There is no point in testing the rule ($egg > 0.5$), it gets the ($egg > 0$) score
  - Also note that we don't need to test "$<$", since it would give equivalent rules

Egg	Milk	Fish	...	Sick?
1	0.7	0.0	...	1
2	0.7	0.0	...	1
0	0.0	1.2	...	0
0	0.7	1.2	...	0
2	0.0	1.3	...	1
0	0.0	0.0	...	0

Supervised Learning Notation

Supervised Learning Notation

$$X = \begin{array}{c} \begin{array}{ccccccc} \textit{Egg} & \textit{Milk} & \textit{Fish} & \textit{Wheat} & \textit{Shellfish} & \textit{Peanuts} & \dots \end{array} \\ \underbrace{\left.\begin{bmatrix} 0.0 & 0.7 & 0.0 & 0.3 & 0.0 & 0.00 & \dots \\ 0.3 & 0.7 & 0.0 & 0.6 & 0.0 & 0.01 & \dots \\ 0.0 & 0.0 & 1.2 & 0.8 & 0.0 & 0.00 & \dots \\ 0.3 & 0.7 & 1.2 & 0.0 & 0.1 & 0.01 & \dots \\ 0.3 & 0.0 & 1.3 & 0.3 & 0.1 & 0.01 & \dots \end{bmatrix}\right\} n}_{d} \end{array} \qquad y = \begin{array}{c} \textit{Sick?} \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{array}$$

- Feature matrix X has rows as examples, columns as features
  - x_{ij} is feature j for example i (quantity of food j on day i)
  - x_i is the list of all features for example i (all the quantities on day i)
  - x_j is column j of the matrix(the value of feature j across all examples)
- Label vector y contains the labels of the examples
  - y_i is the label of example i (1 for "sick", 0 for "not sick")

Supervised Learning Notation

$$X = \begin{array}{c} \begin{array}{ccccccc} \textit{Egg} & \textit{Milk} & \textit{Fish} & \textit{Wheat} & \textit{Shellfish} & \textit{Peanuts} & \ldots \end{array} \\ \hline \underbrace{\left.\begin{bmatrix} 0.0 & 0.7 & 0.0 & 0.3 & 0.0 & 0.00 & \ldots \\ 0.3 & 0.7 & 0.0 & 0.6 & 0.0 & 0.01 & \ldots \\ 0.0 & 0.0 & 1.2 & 0.8 & 0.0 & 0.00 & \ldots \\ 0.3 & 0.7 & 1.2 & 0.0 & 0.1 & 0.01 & \ldots \\ 0.3 & 0.0 & 1.3 & 0.3 & 0.1 & 0.01 & \ldots \end{bmatrix}\right\} n}_{d} \end{array} \qquad y = \begin{array}{c} \textit{Sick?} \\ \hline \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{array}$$

- Training phase
  - Use X and y to find a model (like a decision stump)
- Prediction phase
  - Given an example x_i, use model to predict a label $\hat{y}_i$ ("sick" or "not sick")
- Training error
  - Fraction of times our prediction $\hat{y}_i$ does not equal the true y_i label

Decision Tree Learning

Decision Tree Learning

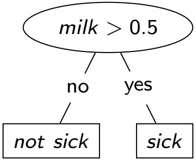
- Decision stumps have only 1 rule based on only 1 feature
  - Very limited class of models: usually not very accurate for most tasks
- Decision trees allow sequences of splits based on multiple features
  - Very general class of models: can get very high accuracy
  - However, it's computationally infeasible to find the best decision tree
- How would you build the tree?
  - Most common decision tree learning algorithm in practice:
    - Greedy recursive splitting

Example of Greedy Recursive Splitting

Start with the full data set

Egg	Milk	...	Sick?
0	0.7	...	1
1	0.7	...	1
0	0.0	...	0
1	0.6	...	1
1	0.0	...	0
2	0.6	...	1
0	1.0	...	1
2	0.0	...	1
0	0.3	...	0
1	0.6	...	0
2	0.0	...	1

Find the decision stump with the best score



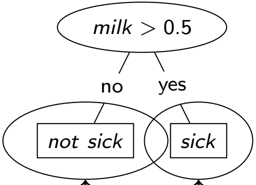
Split into two smaller data sets based on stump

Egg	Milk	...	Sick?
0	0.0	...	0
1	0.0	...	0
2	0.0	...	1
0	0.3	...	0
2	0.0	...	1

Egg	Milk	...	Sick?
0	0.7	...	1
1	0.7	...	1
1	0.6	...	1
2	0.6	...	1
0	1.0	...	1
1	0.6	...	0

Greedy Recursive Splitting

We now have a decision stump and two data sets



milk ≤ 0.5

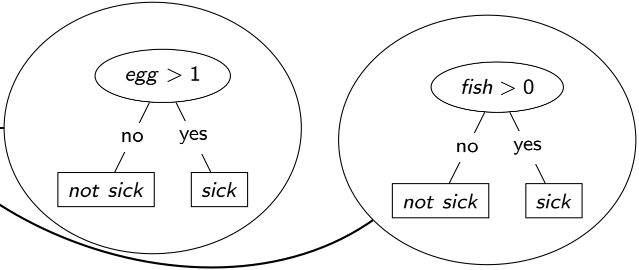
Egg	Milk	...	Sick?
0	0.0	...	0
1	0.0	...	0
2	0.0	...	1
0	0.3	...	0
2	0.0	...	1

milk > 0.5

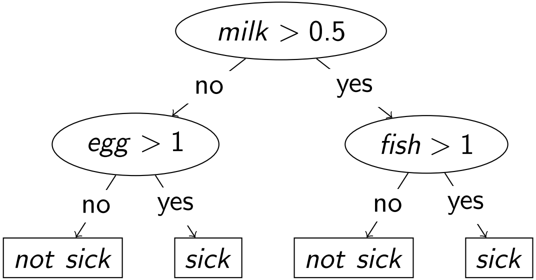
Egg	Milk	...	Sick?
0	0.7	...	1
1	0.7	...	1
1	0.6	...	1
2	0.6	...	1
0	1.0	...	1
1	0.6	...	0

Fit a decision stump to each leaf's data

Then add these stumps to the tree



Greedy Recursive Splitting



- We can continue increasing the depth, when do we stop?
  - Leaves each have only one label
  - User-defined maximum depth

Which score function should a decision tree use?

- How about accuracy?
  - For leafs: no issue
  - For internal nodes: not the best choice
- What if no simple rule improves accuracy?
  - This does not necessarily mean we should stop

Which score function should a decision tree use?

- Most common score in practise is "information gain"
  - Choose split that decreases entropy of labels the most

$$\textit{information gain} = \underbrace{\textit{entropy}(y)}_{\text{entropy before split}} - \frac{\overbrace{n_{yes}}^{\text{number of examples satisfying rule}}}{n} \underbrace{\textit{entropy}(y_{yes})}_{\text{entropy examples satisfying rule}} - \frac{n_{no}}{n} \textit{entropy}(y_{no})$$

with

$$\textit{entropy}(s) = -p_{sick} \log_2 p_{sick} - p_{not\ sick} \log_2 p_{not\ sick}$$

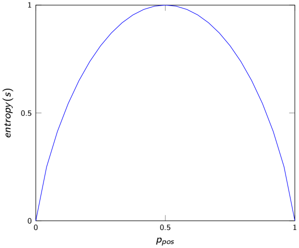
- Information gain for baseline rule ("do nothing") is 0
  - Infogain is large if labels are "more predictable" ("less random") in next layer
- Even if it does not increase classification accuracy at one depth, we hope that it makes the classification easier at the next depth

Entropy

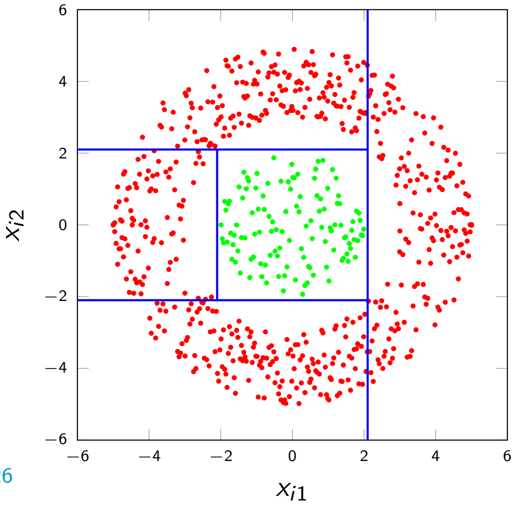
$$\text{information gain} = \underbrace{\mathit{entropy}(y)}_{\text{entropy before split}} - \frac{\overbrace{n_{yes}}^{\text{number of examples satisfying rule}}}{n} \underbrace{\mathit{entropy}(y_{yes})}_{\text{entropy examples satisfying rule}} - \frac{n_{no}}{n} \mathit{entropy}(y_{no})$$

with

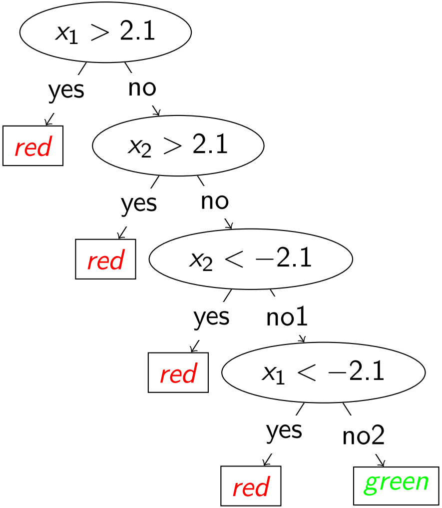
$$\mathit{entropy}(s) = -p_{s_{pos}} \log_2 p_{s_{pos}} - p_{s_{neg}} \log_2 p_{s_{neg}}$$



Example



6
4
2
0
−2
−4
−6
x_{i2}
−6 −4 −2 0 2 4 6
x_{i1}



$x_1 > 2.1$
yes
no
red
$x_2 > 2.1$
yes
no
red
$x_2 < -2.1$
yes
no1
red
$x_1 < -2.1$
yes
no2
red
green

Pruning

Decision Tree Pruning

- There are different stopping criteria that are used in practice
  - You can not always achieve a clean split in the leaves
  - You can use threshold for information gain to decide to stop
- However, sometimes the information gain is low for several levels and then becomes high again (splits become more meaningful
  - You typically grow the tree “too large” and then “prune” it back
- Reduced error pruning

Reduced Error Pruning

Input: decision Tree T; labelled data D
Output: Pruned tree T'
for *every internal node N of T, starting from the bottom* **do**
  $T_N \leftarrow$ subtree of T rooted at N;
  $D_N \leftarrow \{x \in D | x \text{ is covered by } N\}$;
  if *accuracy of T_N over D_N is worse than majority class in D_N* **then**
    replace T_N in T by a leaf labelled with the majority class in D_N;
  end
end
return *pruned version T*

Decision Trees

- Decision Trees
  - Advantages:
    - Easy to implement
    - Interpretable
    - Learning is fast prediction is very fast
    - Can elegantly handle a small number missing values during training
  - Disadvantages
    - Hard to find optimal set of rules
    - Greedy splitting often not accurate, requires very deep trees

Hypothesis Space

Hypothesis Space – Learning as Search

THE UNIVERSITY OF AUCKLAND NEW ZEALAND | SCIENCE SCHOOL OF COMPUTER SCIENCE MACHINE LEARNING

- Learning can be defined as searching the best hypothesis for all observed data
- For decision trees, the hypothesis space are all possible decision trees that can be generated for a data set
- The learner searches through the space and returns the best hypothesis, for decision trees, the tree that potentially best predicts new data
- If the space is small enough, it is possible to test all hypotheses (then no Machine Learning needed)
- So how do we search the space to find the "best" decision tree?

Unsupervised Learning

Unsupervised Learning

SCIENCE
SCHOOL OF COMPUTER SCIENCE
MACHINE LEARNING

- Supervised learning:
  - We have features x_i and class labels y_i
  - Write a program that produces y_i from x_i
- Unsupervised learning
  - We only have x_i values, but no explicit target labels
  - You want to do "something" with them
- Some unsupervised learning tasks
  - Outlier detection: Is this a 'normal' x_i?
  - Similarity search: Which examples look like this x_i?
  - Association rules: Which x_j occur together?
  - Latent-factors: What 'parts' are the x_i made from?
  - Data visualization: What does the high-dimensional X look like?
  - Ranking: Which are the most important x_i?
  - Clustering: What types of x_i are there?

Summary

- Supervised learning
  - Using data to write a program based on input/output examples
- Decision trees: predicting a label using a sequence of simple rules
- Decision stumps: simple decision tree that is very fast to fit
- Greedy recursive splitting: uses a sequence of stumps to fit a tree
  - Very fast and interpretable, but not always the most accurate
- Information gain: splitting score based on decreasing entropy
- Unsupervised Learning
  - Unsupervised learning: fitting data without explicit labels

Literature

- Machine Learning – Tom Mitchell
- Pattern Recognition and Machine Learning – Christopher Bishop
- Machine Learning – The Art and Science of Algorithms that Make Sense of Data – Peter Flach
- Data Mining – Jiawei Han, Micheline Kamber, Jian Pei
- Data Mining – Ian Witten, Eibe Frank, Mark Hall, Christopher Pal

SCIENCE
SCHOOL OF COMPUTER SCIENCE
MACHINE LEARNING

Thank you for your attention!

https://ml.auckland.ac.nz
https://wicker.nz