# CS 486 — Lecture 6: Machine Learning and Decision Trees

## 1 Introduction

- Some applications:
  - Medical diagnosis
  - Spam filtering
  - Facial recognition
  - Speech understanding
  - Handwriting recognition
- Learning is the ability of an agent to improve its performance on future tasks based on experience.
- So the things we want an agent to do are to do more (expand range of behaviours), do things better (improve accuracy), and do things faster.
- Why learn over just hardcoding?
  - There may be situations which cannot be anticipated.
  - Situations may change over time which cannot be anticipated.
  - Sometimes, some solutions just can't be programmed — how do you program image recognition, for example?
- Any learning problem consists of these parts:
  - The problem/task
  - Experience/data that we are using to improve the performance of our agent
  - Background knowledge/bias (make assumptions about the world/task)
  - Measure of improvement — how do we know if we are doing better/worse?
- Types of learning:
  - Supervised learning — given input features, target features, and training examples, predict the value of the target features for new examples given their values on the input features.
  - Unsupervised learning — learning classifications when the examples do not have the targets defined.
  - Reinforcement learning — learning what to do based on rewards and punishments.
- Let us focus on two types of supervised learning problems:
  1. Classification — target features are discrete (ie: is the weather tomorrow sunny, cloudy, or rainy).
  2. Regression — target features are continuous (ie: tomorrow's temperature).

## 2 Supervised Learning

- Given training examples of the form $(x, f(x))$, we want to return a function $h$ (hypothesis) that approximates $f$.
- This does assume that some true function $f(x)$ exists; but during training we never get to actually see $f$ (otherwise we would just use it).

- We can see learning as a search problem if we see it as looking for one best hypothesis given a hypothesis "space".
- The search space is often too large for a systematic search, so most ML techniques are some form of a local search.
- The goal of ML is to find a hypothesis that can predict unseen examples correctly. If it can do so well, then we say it generalizes well.
- How can we choose a hypothesis that generalizes well?
- Occam's razor — assume the simplest hypothesis is the best.
- Cross-validation — use a test set later to validate your hypothesis after using the training set on it. Or, if we don't have test data, split up part of your training set as test data. For example, $k$-fold cross validation will split the training set into $k$ parts; then use one part as validation and the remaining data as training. Repeat $k$ times.
- After running cross-validation, we can take our best $k$-trained hypothesis or train a new hypothesis on all of the data, using parameters selected by cross-validation.
- There is often a trade-off between complex hypotheses that fit the training data better, and simpler hypotheses that generalize better but fit worse.
- The bias-variance trade-off — error is at its highest when the model is too complex (bias low, variance high) OR if it is too low (bias high, variance low) (so a U shape). We want to find a sweet spot in the middle.
- The bias is about the effect of the hypothesis on how well we can fit the data.
- A hypothesis with high bias will be too simplistic, have too few degrees of freedom, assumes too much, and does not fit the training data well.
- If there is too high bias, it cannot use the training data well.
- Meanwhile, the variance is about how much the learned hypothesis varies given different training data.
- A hypothesis with high variance has many degrees of freedom, is very flexible, fits the training data very well, and is very sensitive — small changes in the training data can cause the hypothesis to change a lot.
- This also means that it cannot adapt to test data; it is overfitted to training data and cannot generalize.
- Overfitting will occur if we make our hypothesis too complex — the expected error on training sets should decrease over time, but during cross-validation, if it is overfit, it will start to *increase* the cross-validation error!
- We want a model complexity in the middle; we can use cross-validation to do so by watching it to see if the error rate starts increasing instead of decreasing.

# 3 Decision Trees

- Decision trees are a tree-like structure where we follow branches to reach a leaf as our final "choice".
- How do we build a decision tree?
- We need to determine an order of testing the input features, and given an order of testing the input features, we can build a decision tree by splitting the examples.
- Which decision tree should we generate?

  - Which order of testing the input features should we use? The search space is too big for systematic searches, so we would use a greedy/myopic search.
  - Should we grow a full tree or not? A decision tree can represent any discrete function of input features.
  - We also need a bias — for example, perhaps we prefer the smallest tree, or the one with the least depth, or fewest nodes, etc.

- We stop building a tree when we hit a node that only has one choice (ie: only yes, or only no).
- More generally, when all examples belong to the same class.
- Other stopping criterion are when there are no more features to test, or there are no more examples.