Midterm Review, Machine Learning II

Today’s Class

• Robotics class
• Quick midterm review
• Machine learning: Evaluation
• Machine learning: Beyond decision trees

Robotics Class

• Similar to this class
  • Midterm, final exam, 5-6 homeworks, and group project
  • Intro to a really big area
  • Probability and statistical modeling are important
• Dissimilarities
  • Lots more robot videos
  • Projects involve hardware, sometimes actual robots
  • Somewhat more in-class time spent on projects
• Is it easier or harder?
  • Goal: about the same… but.

Midterm Review: Definitions

• **Induction**: using past data to predict the future
  • The approach to reasoning that says “If it happened this way before, it will happen this way again.”
  • Frequentist, objectivist, and subjectivist/Bayesian reasoning.
• **Objective function**: Measure of what an agent is trying to achieve
  • A function that looks at the world and determines how “good” it is according to goals.
  • In search, applied to a state.

Bookkeeping

• Midterms at end of class
  • Reminder: 24 hours before questions
  • But! you should check point summation
  • We don’t hand out keys but do take questions
• Extra office hours: 9:15-10:15 W, Th this week, M, T, Th next week
• Final covers all material
• No, there’s no Part III (sigh 9.9)

Score / 95 ~Grade

<table>
<thead>
<tr>
<th>Score</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90+</td>
<td>A+</td>
</tr>
<tr>
<td>74+</td>
<td>A</td>
</tr>
<tr>
<td>65+</td>
<td>A-</td>
</tr>
<tr>
<td>60+</td>
<td>B+</td>
</tr>
<tr>
<td>50+</td>
<td>B</td>
</tr>
<tr>
<td>45+</td>
<td>C</td>
</tr>
</tbody>
</table>

Average: 65
Midterm Review: Concepts

- **Value function**: In decision theory, gives a ranking of the “goodness” (desirability) of states
  - E.g.: Italian > pizza > burgers > sandwiches
- **Utility function** gives a number, not just a ranking
  - E.g.: Pizza = 19, burgers = 9, sandwiches = 5
  - Lottery outputs $5000, $100, $5

Midterm Review: Concepts

- **Hill-climbing search**: only looks at immediate neighborhood to see what looks “more” good
  - Can a problem get “stuck” this way?
    - All successors “look” worse but are on the way to better?
    - If “it has to get worse before it gets better,” it can get stuck
    - Hill climbing can never get worse!
  - N-Queens is in the book … as a bad example, of how hill-climbing can get “stuck”

Midterm Review: CSPs

- Defining a CSP:
  - What are the variables we are trying to assign values to?
  - What are the values they could take?
  - How do the assignments for some of them constrain assignments for others?
  - If you have three developers and 5 pieces of work, what do you, the project manager, have to decide?
  - Who works on what in what order?

Midterm Review: CSPs

- Who works on what in what order?
  - Variables should capture developers, work, time
  - Constraints should capture ordering, developers not being able to work on more than one thing at a time
  - There are many ways to do this
    - Assign “phase, current month” to developers
      - Variables = <dev#, phase#, month#>
      - Assign “phase, months until completion” to developers
    - Assign “current month” to developer/phase pairs
      - Variables = <dev#, phaser>, Values = <month#>

Midterm Review: State Spaces

- The set of all states reachable from an initial state (any legal one!) by any sequence of actions.
- Informally: all possible combinations of tile rotations
  - Each arrangement of the board is a state.
- Formally: a start state; a set of actions; and the transition model (what state an action takes us to)
  - What state is this puzzle in initially?
  - What actions can you take?
  - How many arrangements can you reach?

Midterm Review: Heuristics

- Admissible: always underestimates actual cost from a state
  - Need estimate of how many tiles must rotate
  - For this puzzle, “always answer 1” is admissible
- Can you come up with a state where your heuristic gives too high a number?
  - Important: what’s wrong with “number of tiles that need to be rotated” as a heuristic?
    - You don’t know this until you’ve completed the search
    - It’s a “holy grail” answer!
**α-β Pruning and Chance**

- α-βPruning for chance trees:
  - Bound the possible values a chance node can take, given current average
  - Consider whether \( n \) more values averaged into the first value can change that bound
- This requires known bounds on the utility function
- I didn't specify that 🧐
  - So, full credit for either “standard” α-β pruning or “no pruning”

---

**Last Time on Our Show…**

- Decision trees and how to build them
- Information Gain
- Entropy
- Next up:
  - Elements of a Learning System
  - What can go wrong?
  - How do we know how it went?

---

**A Learning System**

- Four components of a machine learning system:
  1. Representation: how do we describe the problem space?
  2. Actor: the part of the system that actually does things.
  3. Critic: Provides the experience we learn from.
  4. Learner: the actual learning algorithm.

---

**Representing The Problem**

- Representing the problem to be solved is the first decision to be made (and most important)
- Requires understanding the domain – the field in which the problem is set
- There are two aspects of representing a problem:
  - Behavior that we want to learn
  - Inputs we will learn from
Representation: Examples to think about

- How do we describe a problem?
  - Guessing an animal
  - Playing checkers
  - Labeling spam email
  - OCRing a check
  - Noticing new help desk topics
- What data do you need to represent for each of these? What model might you learn?

Representation: Examples

- One set of possible answers
  - Guessing an animal: a tree of questions and answers
  - Labeling spam email: the frequencies of words used in this email and in our entire mailbox (TF/IDF). Naive Bayes.
  - OCRing: matrix of light/dark pixels; % light pixels; # straight lines, etc. Neural net.
  - Noticing new help desk topics: Clustering algorithm such as K-Means.

Actor

- Want a system to do something.
  - Make a prediction
  - Sort into categories
  - Look for similarities
- Once a system has learned, or been trained, this is the component we continue to use.
- It may be as simple as a formula to be applied, or it may be a complex program

Actor

- How do we take action?
  - Guessing an animal: walk the tree and ask associated questions
  - Playing checkers: look through the rules and weights to identify a move; choose one; make it.
  - Identifying spam: examine the set of features (word frequencies), calculate the probability of spam.
  - OCRing a check: input the features for a digit, output probability for each of 0 through 9.
  - Help desk topics: output a graphic representation of clusters

Critic

- This component provides the experience we learn from.
- Typically, it is a set of examples with the decision that should be reached or action that should be taken.
- But may be any kind of feedback that indicates how close we are to where we want to be.
- Feedback may be after a single action, or after a sequence.

Critic: Think About

- How do we judge correct actions?
  - Guessing an animal:
  - OCRing digits:
  - Identifying spam:
  - Playing checkers:
  - Grouping documents:
Critic: Possible Answers

- How do we judge correct actions?
  - Guessing an animal: human feedback
  - OCRing digits: Human-categorized training set.
  - Identifying spam: match to a set of human-categorized test documents.
  - Playing checkers: who won?
  - Grouping documents: which are most similar in language or content?
- Can be generally categorized as supervised, unsupervised, reinforcement.

Learner

- The learner is the core of a machine learning system. It will:
  - Examine the information provided by the critic
  - Use it to modify the representation to move toward a more desirable action the next time.
  - Repeat until the performance is satisfactory, or until it stops improving.
- The learner component is what people mean when they refer to a machine learning algorithm or method.

Learner

- What does the learner do?
  - Guessing an animal: ask the user for a question and add it to the binary tree
  - OCRing digits: modify the importance of different input features.
  - Identifying spam: change the set of words likely to be in spam.
  - Playing checkers: increase the chance of using some rules and decrease the chance for others.
  - Grouping documents: find clusters of similar documents

Information Gain

- Concept: make decisions that increase the homogeneity of the data subsets (for outcomes)
- Information gain is based on:
  - Decrease in entropy
  - After a dataset is split on an attribute.
  - High homogeneity – e.g., likelihood samples will have the same class (outcome)

Extensions of the Decision Tree Learning Algorithm

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

Using Gain Ratios

- Information gain favors attributes with a large number of values
  - If we have an attribute D that has a distinct value for each record, then $\text{Info}(D,T) = 0$, thus $\text{Gain}(D,T)$ is maximal.
  - To compensate, use the following ratio instead of Gain: $\text{GainRatio}(D,T) = \text{Gain}(D,T) / \text{SplitInfo}(D,T)$
  - $\text{SplitInfo}(D,T) = I(|T_1|/|T|, |T_2|/|T|, ..., |T_m|/|T|)$
  - where $\{T_1, T_2, ..., T_m\}$ is the partition of T induced by value of D
Real-Valued Data

• Select a set of thresholds defining intervals
  • Each interval becomes a discrete value of the attribute
• How?
  • Use simple heuristics…
  • Always divide into quartiles
  • Use domain knowledge…
  • Divide age into infant (0-2), toddler (3-5), school-aged (5-8)
  • Or treat this as another learning problem
  • Try a range of ways to discretize the continuous variable and see which yield “better results” w.r.t. some metric
  • E.g., try midpoint between every pair of values

Noisy Data

• Many kinds of “noise” can occur in the examples:
  • Two examples have same attribute/value pairs, but different classifications
  • Some values of attributes are incorrect
  • Errors in the data acquisition process, the preprocessing phase, ...
  • Classification is wrong (e.g., + instead of -) because of some error
  • Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome
  • Some attributes are missing (are pangolins bipedal?)

Pruning Decision Trees

• Replace a whole subtree by a leaf node
• If a decision rule establishes that he expected error rate in the subtree is greater than in the single leaf. E.g.,
  • Training: one training red success and two training blue failures
  • Test: one red failure and one blue success
  • Consider replacing this subtree by a single Failure node (leaf)
• After replacement we will have only two errors instead of five:

Training

<table>
<thead>
<tr>
<th>Color</th>
<th>1 success 0 failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue</td>
<td>2 failures</td>
</tr>
</tbody>
</table>

Test

<table>
<thead>
<tr>
<th>Color</th>
<th>1 success 3 failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue</td>
<td>1 failure</td>
</tr>
</tbody>
</table>

Pruned

<table>
<thead>
<tr>
<th>Color</th>
<th>2 success 4 failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue</td>
<td></td>
</tr>
</tbody>
</table>

Converting Decision Trees to Rules

• It is easy to derive a rule set from a decision tree:
  • Write a rule for each path in the decision tree from the root to a leaf
  • Left-hand side is label of nodes and labels of arcs
  • The resulting rules set can be simplified:
    • Let LHS be the left hand side of a rule
    • Let LHS’ be obtained from LHS by eliminating some conditions
    • We can replace LHS by LHS’ in this rule if the subsets of the training set that satisfy respectively LHS and LHS’ are equal
  • A rule may be eliminated by using metaconditions such as “if no other rule applies”

Measuring Model Quality

• How good is a model?
  • Predictive accuracy
  • False positives / false negatives for a given cutoff threshold
  • Loss function (accounts for cost of different types of errors)
  • Area under the curve
  • Minimizing loss can lead to problems with overfitting

• Training error
  • Train on all data; measure error on all data
  • Subject to overfitting (of course we’ll make good predictions on the data on which we trained!)
• Regularization
  • Attempt to avoid overfitting
  • Explicitly minimize the complexity of the function while minimizing loss
  • Tradeoff is modeled with a regularization parameter

Measuring Model Quality

• Training error
  • Train on all data; measure error on all data
  • Subject to overfitting (of course we’ll make good predictions on the data on which we trained!)
• Regularization
  • Attempt to avoid overfitting
  • Explicitly minimize the complexity of the function while minimizing loss
  • Tradeoff is modeled with a regularization parameter
Cross-Validation

• Holdout cross-validation:
  • Divide data into training set and test set
  • Train on training set; measure error on test set
  • Better than training error, since we are measuring generalization to new data
  • To get a good estimate, we need a reasonably large test set
  • But this gives less data to train on, reducing our model quality!

Cross-Validation, cont.

• k-fold cross-validation:
  • Divide data into k folds
  • Train on k-1 folds, use the kth fold to measure error
  • Repeat k times; use average error to measure generalization accuracy
  • Statistically valid and gives good accuracy estimates

• Leave-one-out cross-validation (LOOCV)
  • k-fold cross validation where k=N (test data = 1 instance!)
  • Quite accurate, but also quite expensive, since it requires building N models

Correctness

• True positive
• True negative
• False positive
• False negative

Precision/Recall

Noisy Data

• Many kinds of “noise” can occur in the examples:
  • Two examples have same attribute/value pairs, but different classifications
  • Some values of attributes are incorrect because of errors in data acquisition or preprocessing phase
  • The classification is wrong (e.g., + instead of -) because of some error
  • Attributes irrelevant to the decision-making process
    • Color of a die is irrelevant to its outcome
    • Can still be in training data, can be chosen as an attribute

Summary: Decision Tree Learning

• One of the most widely used learning methods in practice
• Can out-perform human experts in many problems

• Strengths include
  • Fast
  • Simple to implement
  • Can convert result to a set of easily interpretable rules
  • Empirically valid in many commercial products
  • Handles noisy data

• Weaknesses:
  • Univariate splits/partitioning using only one attribute at a time
  • Large decision trees may be hard to understand
  • Requires fixed-length feature vectors
  • Non-incremental (i.e., batch method)