Machine Learning

David Kauchak, CS151, Fall 2010

Project ideas

- Improved mancala player
  - examine improvements from other game playing components
    - end-game table
    - transposition table
    - learned weights for evaluation function
- Examine the performance of other search algorithms for an application (for example theta*)
- Play with games with different characteristics
  - games with chance
  - unobservability (blind tic-tac-toe, stratego)
  - games with betting
- Compare local search methods on an application
- Compare CSP heuristics

Project ideas

- Bayes nets
  - Variable elimination algorithm
    - compare vs. enumeration
    - compare different variable ordering heuristics
  - implement MCMC
    - lots of applications here
- SPAM identification/detection
- Improved sentiment classification
- Compare document classification techniques (NB vs. multinomial NB)
- Play with some machine learning approach(es)
  - http://archive.ics.uci.edu/ (lots of data sets)

Admin

- CS colloquium tomorrow
- Literature review due Friday
Project ideas

• HMMs
  – Applications
    • HMM part of speech tagger
    • Phone texting prediction (e.g. T9) – other models besides HMM might also be interesting
  – HMM smoothing for tracking movement
  – HMM prediction
    • Predicting movement in video
    • Stock market prediction
    • Sports prediction
• Clustering
  – Compare clustering approaches
  – See how clustering does on a particular dataset
  – State space search hierarchical clustering

The plan

• We looked in a lot of detail at HMMs
• For the next few classes, going to look at machine learning techniques
• More an emphasis on breadth
• You’ll get to play with some of the techniques in the homeworks (and maybe your final projects)

The mind-reading game

How good are you at guessing random numbers?

Repeat 100 times:
Computer guesses whether you’ll type 0/1
You type 0 or 1

http://seed.ucsd.edu/~mindreader/
[written by Y. Freund and R. Schapire]
The mind-reading game

The computer is right much more than half the time…

Strategy: computer predicts next keystroke based on the last few (maintains weights on different patterns)

There are patterns everywhere… even in “randomness”!

Supervised learning

APPLES
BANANAS

Supervised learning: given labeled data

Lots of learning problems

• Given labeled examples, learn to label unlabeled examples

APPLE or BANANA?

Supervised learning: learn to classify unlabeled
Lots of learning problems

- We’ll focus on supervised learning this class and next
- Some unsupervised
- Many others
  - semi-supervised learning: some labeled data and some unlabeled data
  - active learning: unlabeled data, but we can pick some examples to be labeled
  - reinforcement learning: maximize a cumulative reward. Learn to drive a car, reward = not crashing
- and variations
  - online vs. offline learning: do we have access to all of the data or do we have to learn as we go
  - classification vs. regression: are we predicting between a finite set or are we predicting a score/value

Supervised classification: training

Labeled data

<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>train a predictive model</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Feature based classification

Training or learning phase

<table>
<thead>
<tr>
<th>Raw data</th>
<th>Label</th>
<th>features</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>f₁, f₂, f₃, ..., fₙ</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>f₁, f₂, f₃, ..., fₙ</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>extract features</td>
<td>f₁, f₂, f₃, ..., fₙ</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>f₁, f₂, f₃, ..., fₙ</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>f₁, f₂, f₃, ..., fₙ</td>
<td>0</td>
</tr>
</tbody>
</table>

Unlabeled data

<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

predict the label
Feature based classification

Testing or classification phase

- Raw data
- Extract features
- Features: $f_1, f_2, f_3, ..., f_n$
- Predict the label
- Classifier
- Labels: 1, 0, 0, 1, 0

An example

Database of 20,000 images of handwritten digits, each labeled by a human

[28 x 28 greyscale; pixel values 0-255; labels 0-9]

Use these to learn a classifier which will label digit-images automatically...

The learning problem

Input space $X = \{0, 1, ..., 255\}^{784}$

Output space $Y = \{0, 1, ..., 9\}$

Training set $(x_1, y_1), ..., (x_m, y_m)$ $m = 20,000$

Learning Algorithm

Classifier $f: X \rightarrow Y$

Points in a feature space

Note most actual feature spaces are much, much larger!
Test example: what class?

Test example = Class 1

Class 1
Class 2
Class 3

Class 1
Class 2
Class 3

Nearest neighbor

<table>
<thead>
<tr>
<th>Image to label</th>
<th>Nearest neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Overall:
error rate = 6%  
(on test set)

Question: what is the error rate for random guessing?

What does it get wrong?

Who knows… but here’s a hypothesis:
Each digit corresponds to some connected region of $\mathbb{R}^{784}$. Some of the regions come close to each other; problems occur at these boundaries.

e.g. a random point in this ball has only a 70% chance of being in $R_2$
Boost probability of success

Analogy: suppose a (biased) coin has
Pr(heads) = 0.70
Flip it 11 times and return the majority vote:
Pr(heads) = 0.92

Therefore: to classify x, find its k nearest neighbors (in the training set) and return their majority vote

Large deviation theory: the foundation of machine learning...

k-Nearest Neighbor (k-NN)

• To classify an example \( d \):
  – Find \( k \) nearest neighbors of \( d \)
  – Choose as the class the majority class within the \( k \) nearest neighbors
• Can get rough approximations of probability of belonging to a class as fraction of \( k \)
• Does not explicitly compute boundary or model
• Also called:
  – Case-based learning
  – Memory-based learning
  – Lazy learning

Example: k=6 (6-NN)

k Nearest Neighbor

• What value of \( k \) should we use?
  – Using only the closest example (1NN) to determine the class is subject to errors due to:
    • A single atypical example
    • Noise
  – Pick \( k \) too large and you end up with looking at neighbors that are not that close
  – Value of \( k \) is typically odd to avoid ties; 3 and 5 are most common.
**k-NN**

<table>
<thead>
<tr>
<th>k</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.0</td>
</tr>
<tr>
<td>3</td>
<td>5.9</td>
</tr>
<tr>
<td>5</td>
<td>6.0</td>
</tr>
<tr>
<td>7</td>
<td>6.0</td>
</tr>
<tr>
<td>9</td>
<td>5.8</td>
</tr>
</tbody>
</table>

**k-NN decision boundaries**

k-NN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, etc.)

---

**k-NN: The good and the bad**

- **Good**
  - No training is necessary
  - No feature selection necessary
  - Scales well with large number of classes
    - Don’t need to train n classifiers for n classes

- **Bad**
  - Classes can influence each other
    - Small changes to one class can have ripple effect
  - Scores can be hard to convert to probabilities
  - Can be more expensive at test time
  - “Model” is all of your training examples which can be large

---

**Choosing the correct model capacity**

Which separating line should we use?
**Bias/Variance**

- Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

**Bias/Variance**

- Another way to think about it is model complexity
- Simple models
  - may not model data well
  - high bias
- Complicated models
  - may overfit to the training data
  - high variance
- Why do we care about bias/variance?

---

**Bias/variance trade-off**

We want to fit a polynomial to this, which one should we use?

**Bias/variance trade-off**

High variance OR high bias?

- Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model
Bias/variance trade-off

High bias

- Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

High variance

- Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

Bias/variance trade-off

High variance OR high bias?

- Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

What do we want?

- Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model
Bias/variance trade-off

• Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
• Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

k-NN vs. Naive Bayes

How do k-NN and NB sit on the variance/bias plane?

• Bias: How well does the model predict the training data?
  - high bias – the model doesn’t do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
• Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

k-NN vs. Naive Bayes

How do k-NN and NB sit on the variance/bias plane?

• k-NN has high variance and low bias.
  - more complicated model
  - can model any boundary
  - but very dependent on the training data
• NB has low variance and high bias.
  - Decision surface has to be linear
  - Cannot model all data
  - but, less variation based on the training data

Playing tennis

• You want to decide whether or not to play tennis today
  - Outlook: Sunny, Overcast, Rain
  - Humidity: High, normal
  - Wind: Strong, weak
• What might be a reasonable decision?
**Decision tree** is an intuitive way of representing a decision

- Tree with internal nodes labeled by features
- Branches are labeled by tests on that feature
  - outlook = sunny
  - $x > 100$
- Leaves labeled with classes

Another decision tree

**Document classification:** wheat or not wheat?

**Decision tree learning**

Features are $x$ and $y$
A node in the tree is threshold on that dimension

How could we learn a tree from data?
Decision tree learning

Features are x and y
A node in the tree is threshold on that dimension

How could we learn a tree from data?

Decision Tree Learning

• Start at the top and work our way down
  – Examine all of the features to see which feature best separates the data
  – Split the data into subsets based on the feature test
  – Test the remaining features to see which best separates the data in each subset
  – Repeat this process in all branches until:
    • all examples in a subset are of the same type
    • there are no examples left
    • there are no attributes left

Ideas?

KL-Divergence

• Given two probability distributions P and Q

\[ D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \]

When is this large? small?
**KL-Divergence**

- Given two probability distributions $P$ and $Q$
  
  \[
  D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}
  \]

  When $P = Q$, $D_{KL}(P||Q) = 0$

**KL-Divergence**

- Given two probability distributions $P$ and $Q$
  
  \[
  D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}
  \]

  \[
  \begin{align*}
  P(1) &= 0.999 \\
  P(2) &= 0.001 \\
  Q(1) &= 0.001 \\
  P(2) &= 0.999
  \end{align*}
  \]

  \[
  D_{KL}(P||Q) = 6.89
  \]

  KL-divergence is a measure of the distance between two probability distributions (though it’s not a distance metric!)

**Information Gain**

\[
D_{KL}(P(class \mid f) \parallel P(class)) = \sum_{c \in class} P(c \mid f) \log \frac{P(c \mid f)}{P(c)}
\]

- What is the distance from the probability of a class (i.e., the prior) and the probability of that class conditioned on $f$?
- What information do we gain about the class decision, given the feature $f$?
- Use information gain to decide the most informative feature to split on

**Decision tree learning**

Features are $x$ and $y$

A node in the tree is threshold on that dimension

What would be the learned tree?
**Decision tree learning**

Features are $x$ and $y$
A node in the tree is threshold on that dimension

Do you think this is right?

**Overfitting**

- Decision trees can have a high variance
- The model can be too complicated and *overfit* to the training data

**Pruning**

- Just like in our game tree pruning (alpha-beta) we can also prune a decision tree
- Lots of ways to do this
- One common way:
  - Measure accuracy on a hold-out set (i.e. not used for training)
    - Stop splitting when accuracy decreases
    - Prune tree from the bottom up, replacing split nodes with majority label, while accuracy doesn’t decrease
- Other ways look at complexity of the model with respect to characteristics of the training data

**Decision trees**

- **Good**
  - Very human friendly
  - easy to understand
  - people can modify
  - fairly quick to train
- **Bad**
  - overfitting/pruning can be tricky
  - greedy approach: if you make a split you’re stuck with it
  - performance is ok, but can do better
A good review

• Top Gear reviews Ford Fiesta
  – http://www.youtube.com/watch?v=6Zy78tFPQwQ

• Different sections highlighting different aspects
• In each case, cites specific examples of good and bad things
• Keeps the end-user of the review in mind (in our case, pretend it’s either the authors of the paper or someone on a conference committee)