Machine Learning

David Kauchak, CS151, Fall 2010

Admin

- CS colloquium tomorrow
- Literature review due Friday

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

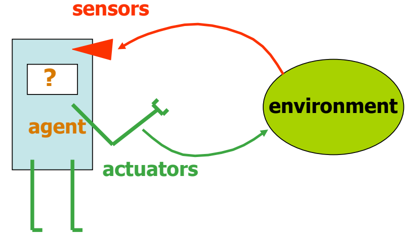
Project ideas

- 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/ml/ (lots of data sets)

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

Learning



As an agent interacts with the world, it should learn about it's environment

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

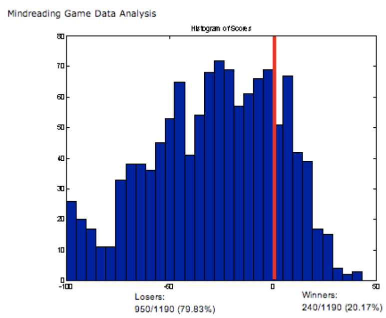
How good are you at guessing random numbers?

```
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…



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

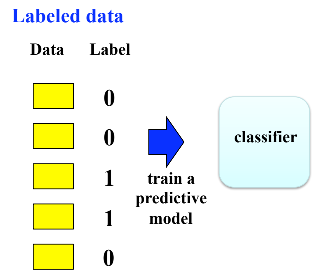
APPLE or BANANA?

Supervised learning: learn to classify unlabeled

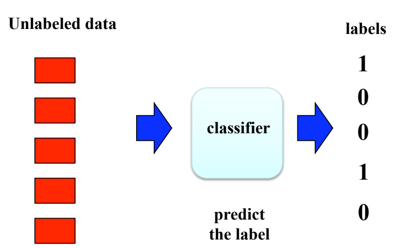
Lots of learning problems

- 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

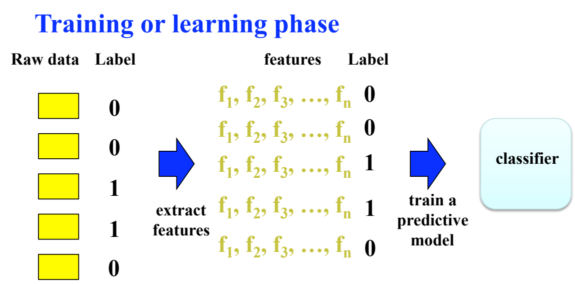
Supervised classification: training



Supervised learning: testing/ classifying

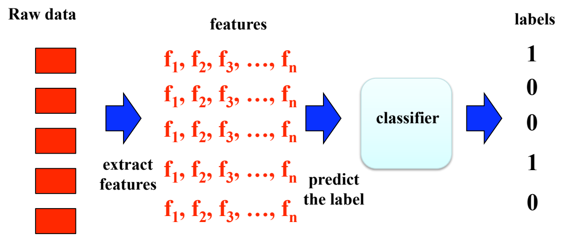


Feature based classification



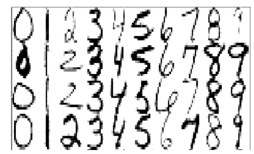
Feature based classification

Testing or classification phase



An example

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



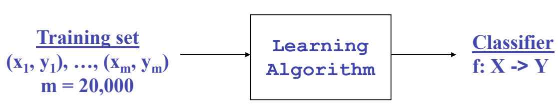
[28 x 28 greyscale; pixel values 0-255; labels 0-9]

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

The learning problem

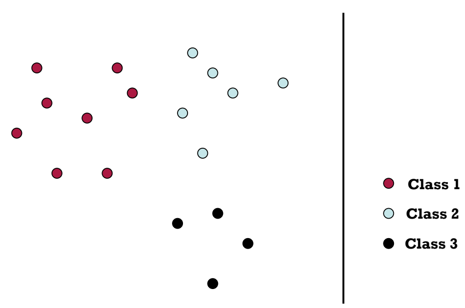
Input space $X = \{0,1,\ldots,255\}^{784}$

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

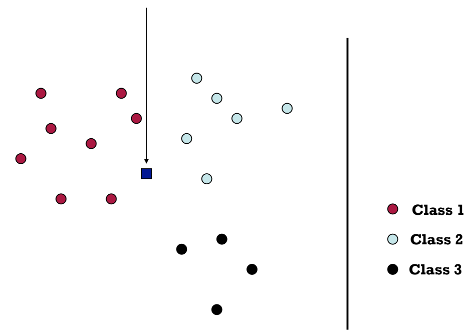


Points in a feature space

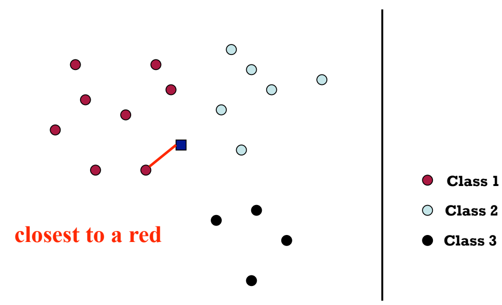
Note most actual feature spaces are much, much larger!



Test example: what class?

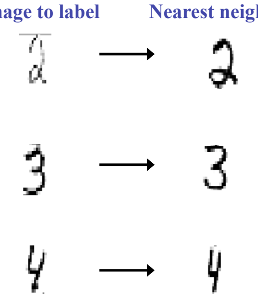


Test example = Class 1



Nearest neighbor

Image to label **Nearest neighbor**



Overall:

error rate = 6%

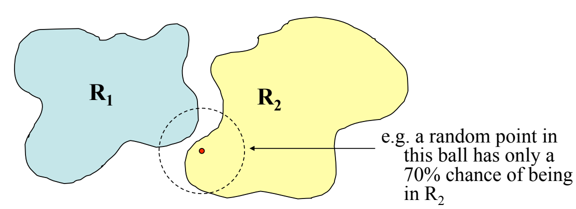
(on test set)

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 R^{784}. Some of the regions come close to each other; problems occur at these boundaries.



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

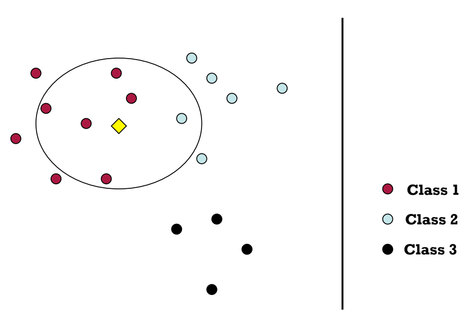
Therefore: to classify x, find its k nearest neighbors (in the training set) and return their majority vote

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)



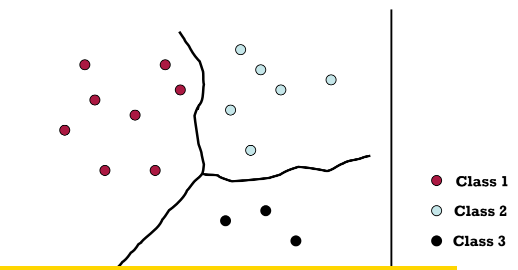
k Nearest Neighbor

- 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

k	error
1	6.0
3	5.9
5	6.0
7	6.0
9	5.8

k-NN decision boundaries

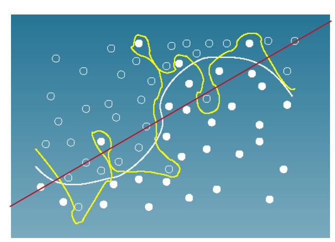


k-NN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, etc.)

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?

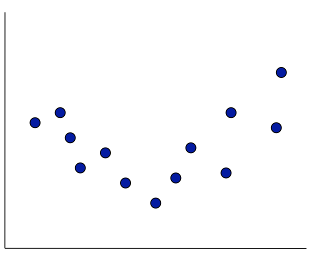
Bias/Variance

- 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

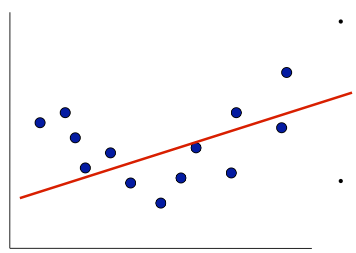
- Another way to think about it is model complexity
- Simple models
  - may not model data well
  - high bias
- Complicated models
  - may overfit to the training data
  - high variance
- Why do we care about bias/variance?

Bias/variance trade-off



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

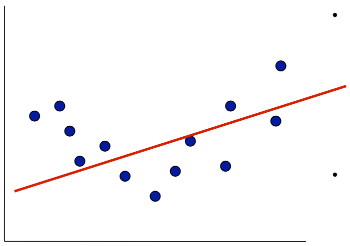
Bias/variance trade-off



- 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 OR high bias?

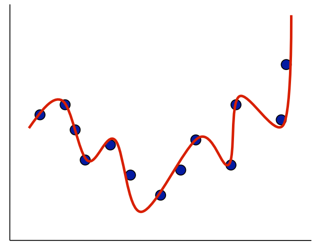
Bias/variance trade-off



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

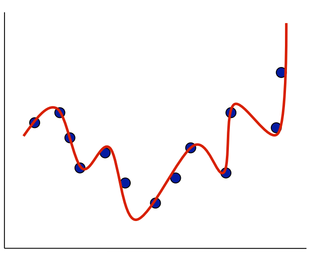
Bias/variance trade-off



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

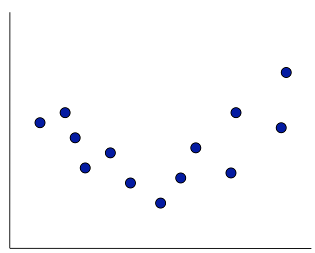
Bias/variance trade-off



High 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

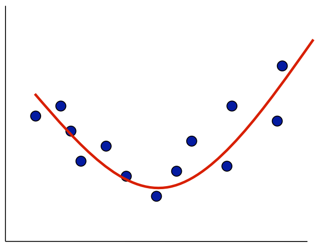
Bias/variance trade-off



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



Compromise between bias and 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

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?

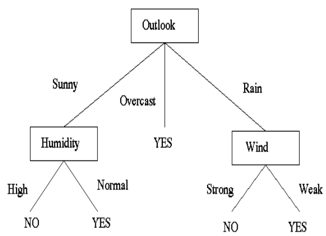
- k-NN has high variance and low bias.
  - more complicated model
  - can model any boundary
  - but very dependent on the training data
- NB has low variance and high bias.
  - Decision surface has to be linear
  - Cannot model all data
  - but, less variation based on the training data

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?

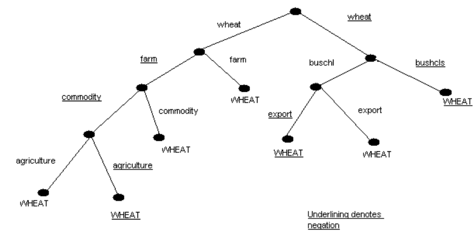
Decision tree **is an intuitive way of representing a decision**

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



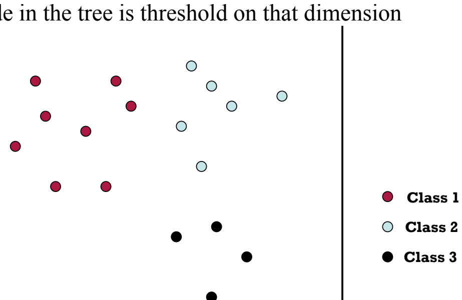
Another decision tree

Document classification:
wheat or not wheat?



Decision tree learning

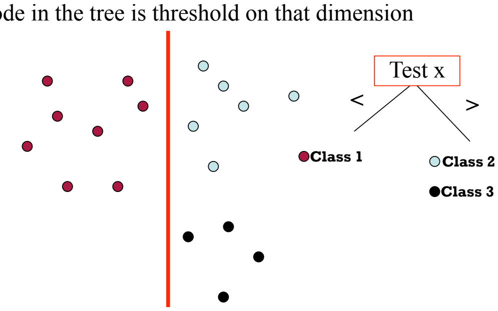
Features are x and y
A node in the tree is threshold on that dimension



How could we learn a tree from data?

Decision tree learning

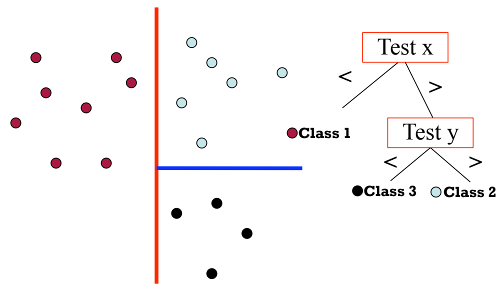
Features are x and y
A node in the tree is threshold on that dimension



How could we learn a tree from data?

Decision tree learning

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

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

$$D_{KL}(P \,\|\, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

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

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

P(1) = 0.999	**Q(1) = 0.001**
P(2) = 0.001	**P(2) = 0.999**

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!

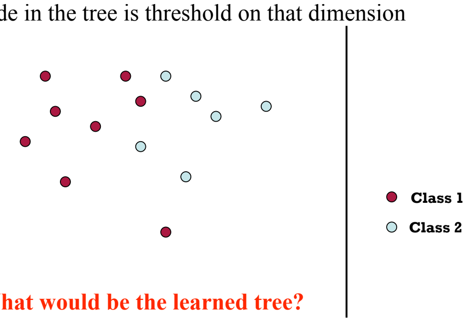
Information Gain

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

- 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

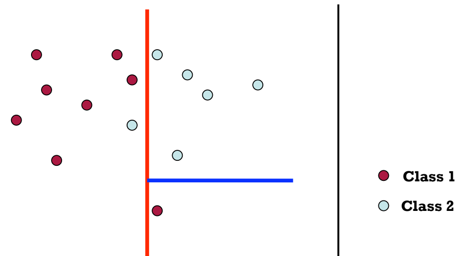
Features are x and y
A node in the tree is threshold on that dimension



What would be the learned tree?

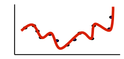
Decision tree learning

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

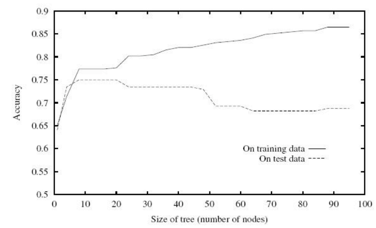


Do you think this is right?

Overfitting



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



Ideas?

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 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)