Ensemble Learning and Statistical Learning

CS 486/686
Introduction to AI
University of Waterloo

Outline

- Ensemble Learning
- Statistical learning
  - Bayesian learning
  - Maximum a posteriori (MAP)
  - Maximum likelihood

Ensemble learning

- So far our learning methods have had the following general approach
  - Choose a single hypothesis from the hypothesis space
  - Use this hypothesis to make predictions

- Maybe we can do better by using a lot of hypothesis from the hypothesis space and combine their predictions

Ensemble Learning

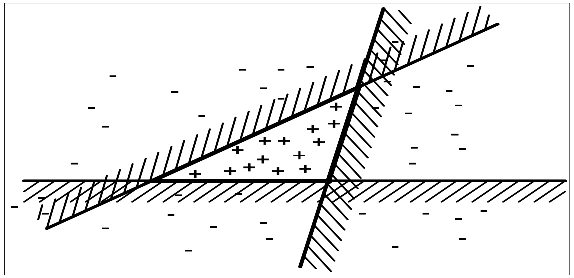
- Analogies
  - Elections
  - Committees
- Intuitions:
  - Individuals may make mistakes
    - The majority may be less likely to make a mistake
  - Individuals have partial information
    - Committes pool experise

Ensemble expressiveness

- Using ensembles can also enlarge the hypothesis space
  - Ensemble as hypothesis
  - Set of all ensembles as hypothesis space

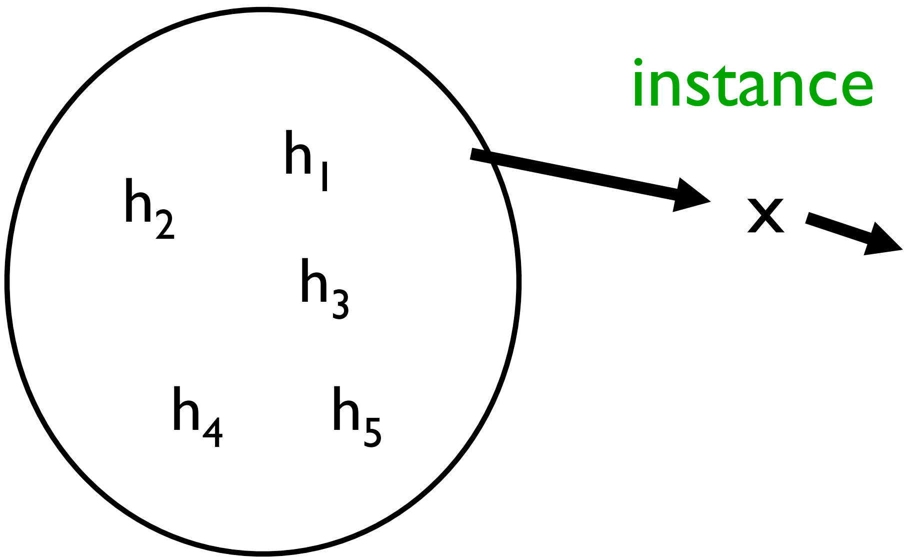
Original hypothesis space: linear threshold hypothesis

- Simple, efficient learning algorithms but not particularly expressive



Bagging

- Majority voting:



classification

Majority(h_1(x),h_2(x),h_3(x),h_4(x),h_5(x))

Ensemble of hypothesis

For the classification to be wrong, at least 3 out of 5 hypothesis have to be wrong

Bagging

- Assumptions:
  - Each h_i makes an error with probability p
  - Hypotheses are independent

- Majority voting of n hypotheses
  - Probability k make an error?
  - Probability majority make an error?

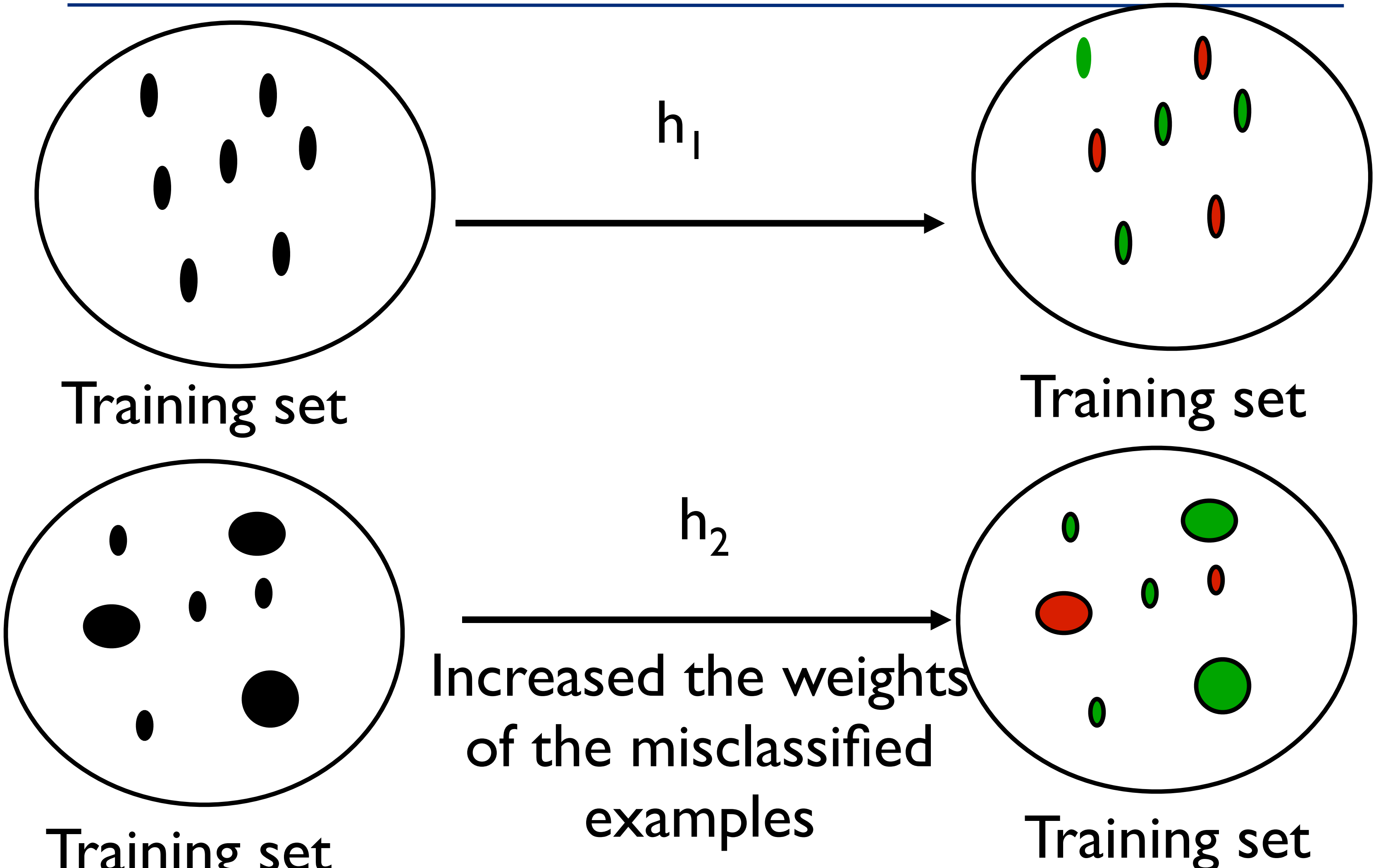
Weighted Majority

- In practice
  - Hypotheses are rarely independent
  - Some hypotheses have less errors than others
- Weighted majority
  - Intuition
    - Decrease weights of correlated hypotheses
    - Increase weights of good hypotheses

Boosting

- Boosting is the most commonly used form of ensemble learning
  - Very simple idea, but very powerful
    - Computes a weighted majority
    - Operates on a weighted training set

Boosting



h_1
Training set
Training set
h_2
Increased the weights of the misclassified examples
Training set
Training set

AdaBoost

- w_j<- 1/N
- For m=1 to M do
  - h_m <- learn(data,w)
  - err<- 0
  - For each (x_i,y_i) in data do
    - If $h_m(x_i) \neq y_i$ then err <-err + w_i
  - For each (x_i,y_i) in data do
    - If $h_m(x_i)=y_i$ then w_i<-w_i*err/(1-err)
  - w <- normalize(w)
  - z_m<-log[(1-err)/err]
- Return weighted-majority(h,z)

Boosting

- Many variations of boosting
  - ADABOOST is a specific boosting algorithm
  - Takes a weak learner L (classifies slightly better than just random guessing)
  - Returns a hypothesis that classifies training data with 100% accuracy (for large enough M)

Robert Schapire and Yoav Freund
Kanellakis Award for 2004

Boosting Paradigm

- Advantages
  - No need to learn a perfect hypothesis
  - Can boost any weak learning algorithm
  - Easy to program
  - Good generalization
- When we have a bunch of hypotheses, boosting provides a principled approach to combine them
  - Useful for sensor fusion, combining experts…

Statistical Learning

- Statistical learning
  - Bayesian learning
  - Maximum a posteriori (MAP)
  - Maximum likelihood

Motivation: Things you know

- Agents model uncertainty in the world and utility of different courses of actions
- Bayes nets are models of probability distributions
- Models involve a graph structure **annotated** with probabilities
- Bayes nets for realistic applications have hundreds of nodes and tens of links…

- **Where do these numbers come from?**

Recall: Pathfinder
(Heckerman, 1991)

- Medical diagnosis for lymph node disease
- Large net
  - 60 diseases, 100 symptoms and test results, 14000 probabilities
- Built by medical experts
  - 8 hours to determine the variables
  - 35 hours for network topology
  - 40 hours for probability table values

Knowledge acquisition bottleneck

- In many applications, Bayes net structure and parameters are set by experts in the field
  - Experts are scarce and expensive
  - Experts can be inconsistent
  - Experts can be non-existent
- But data is cheap and plentiful (usually)

- **Goal of learning**:
  - Build models of the world directly from data
  - We will focus on learning models for probabilistic models

Candy Example (from R&N)

- Favorite candy sold in two flavors
  - Lime
  - Cherry
- Same wrapper for both flavors
- Sold in bags with different ratios
  - 100% cherry
  - 75% cherry, 25% lime
  - 50% cherry, 50% lime
  - 25% cherry, 75% lime
  - 100% lime

Candy Example

- You bought a bag of candy but do not know its flavor ratio
- After eating k candies
  - What is the flavor ratio of the bag?
  - What will be the flavor of the next candy?

Statistical Learning

- Hypothesis H: probabilistic theory about the world
  - h_1: 100% cherry
  - h_2: 75% cherry, 25% lime
  - h_3: 50% cherry, 50% lime
  - h_4: 25% cherry, 75% lime
  - h_5: 100% lime

- Data D: evidence about the world
  - d_1: 1st candy is cherry
  - d_2: 2nd candy is lime
  - d_3: 3rd candy is lime
  - …

Bayesian learning

- Prior: P(H)
- Likelihood: P(d|H)
- Evidence: $d=<d_1,d_2,\ldots,d_n>$

- Bayesian learning
  - Compute the probability of each hypothesis given the data
  - $P(H|d)=\alpha\, P(d|H)P(H)$

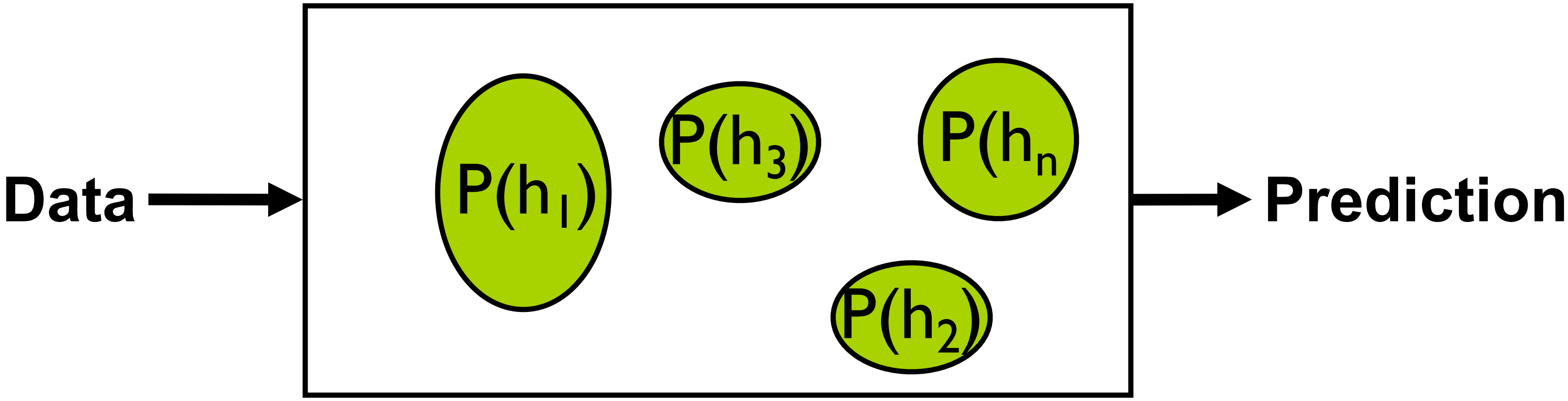
Bayesian learning

- Suppose we want to make a prediction about some unknown quantity x
  - i.e. flavor of next candy
- $P(x|d)=\sum_i P(x|d,h_i)P(h_i|d)$

  $= \sum_i P(x|h_i)P(h_i|d)$
- Predictions are weighted averages of the predictions of the individual hypothesis

Bayesian learning

- Hypothesis are “intermediaries” between raw data and prediction

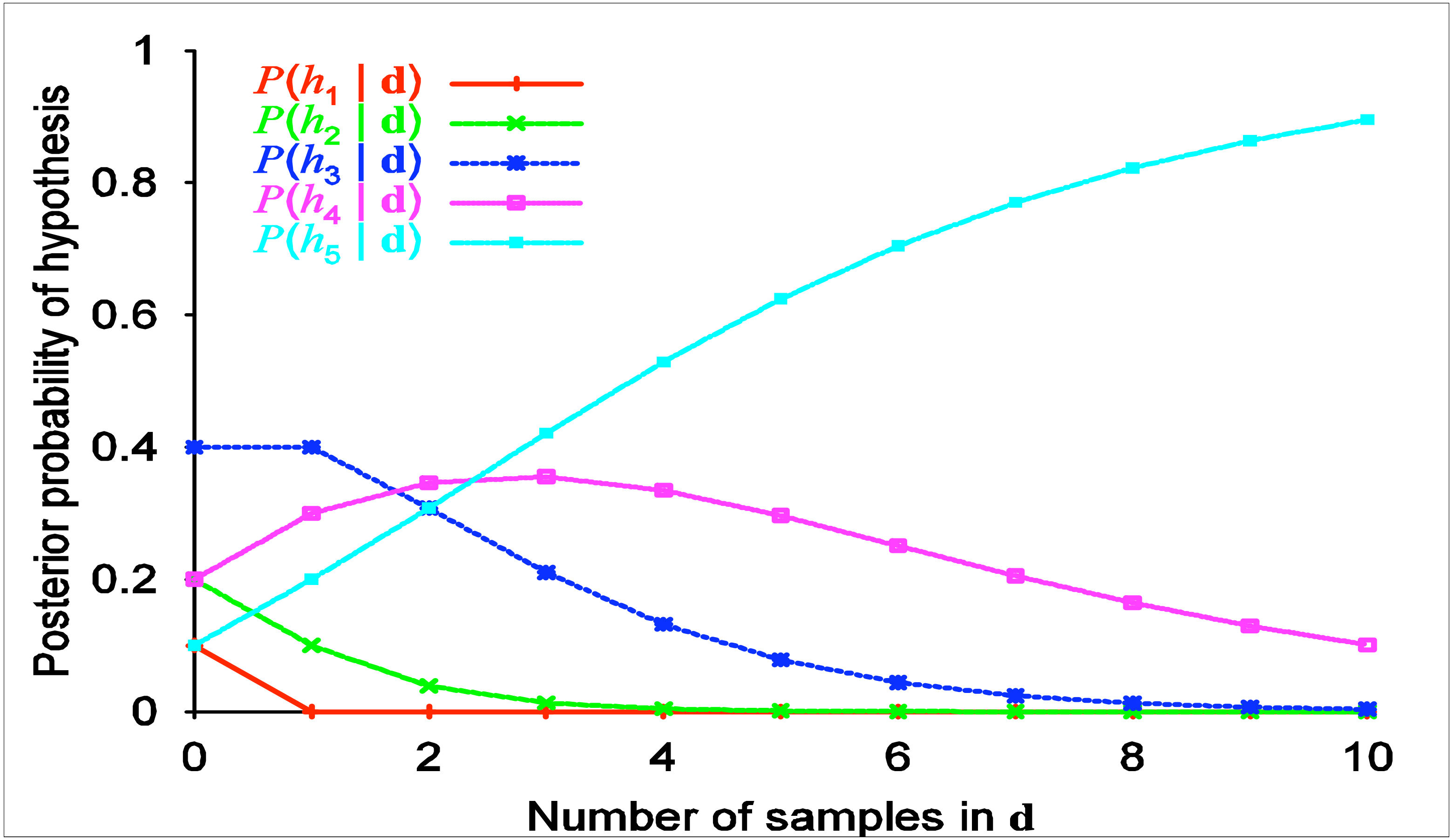


Candy Example

- Assume prior P(H)=<0.1,0.2,0.4,0.2,0.1>
- Assume candies are i.i.d (identically and independently distributed)
  - $P(d|h_i)=\Pi_j P(d_j|h_i)$
- Suppose first 10 candies are all lime
  - $P(d|h_1)=0^{10}=0$
  - $P(d|h_2)=0.25^{10}=0.00000095$
  - $P(d|h_3)=0.5^{10}=0.00097$
  - $P(d|h_4)=0.75^{10}=0.056$
  - $P(d|h_5)=1^{10}=1$

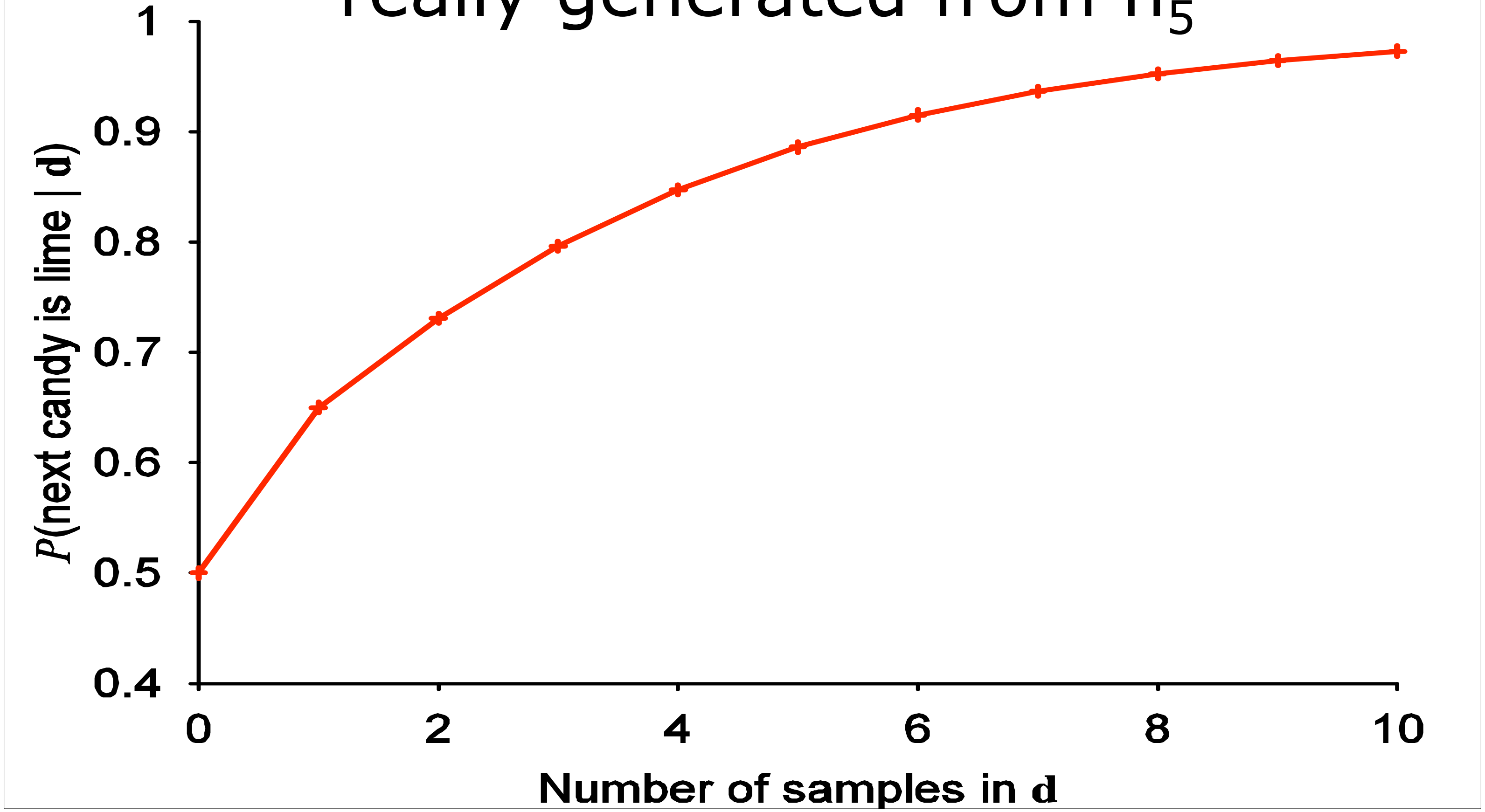
Candy Example: Posterior

Posteriors given that data is really generated from h_5



Candy Example: Prediction

Prediction next candy is lime given that data is really generated from h_5



Bayesian learning

- **Good news**
  - Optimal
    - Given prior, no other prediction is correct more often than the Bayesian one
  - No overfitting
    - Use prior to penalize complex hypothesis (complex hypothesis are more unlikely)
- **Bad news**
  - If hypothesis space is large, Bayesian learning is intractable
    - Large summation (or integration) problem
- Use approximations
  - Maximum a posteriori (MAP)

Maximum a posteriori (MAP)

- Idea: Make prediction on most probable hypothesis h_{MAP}
  - $h_{MAP}=\text{argmax}_{h_i} P(h_i|d)$
  - $P(x|d)=P(x|h_{MAP})$

- Compare to Bayesian learning
  - Bayesian learning makes prediction on all hypothesis weighted by their probability

MAP – Candy Example

MAP Properties

- MAP prediction is less accurate than Bayesian prediction
  - MAP relies on only one hypothesis
- MAP and Bayesian predictions converge as data increases
- No overfitting
  - Use prior to penalize complex hypothesis
- Finding h_{MAP} may be intractable
  - h_{MAP}=argmax $P(h|d)$
  - Optimization may be hard!

MAP computation

- Optimization
  - $h_{MAP} = \text{argmax}_h\ P(h|d)$

    $= \text{argmax}_h\ P(h)P(d|h)$

    $= \text{argmax}_h\ P(h)\Pi_i\ P(d_i|h)$
- Product introduces non-linear optimization
- Take log to linearize
  - $h_{MAP} = \text{argmax}_h \log P(h) + \sum_i \log P(d_i|h)$

Maximum Likelihood (ML)

- Idea: Simplify MAP by assuming uniform prior (i.e. $P(h_i)=P(h_j)$ for all i,j)
  - $h_{MAP}=\text{argmax}_h\ P(h)\ P(d|h)$
  - $h_{ML}=\text{argmax}_h\ P(d|h)$
- Make prediction on h_{ML} only
  - $P(x|d)=P(x|h_{ML})$

ML Properties

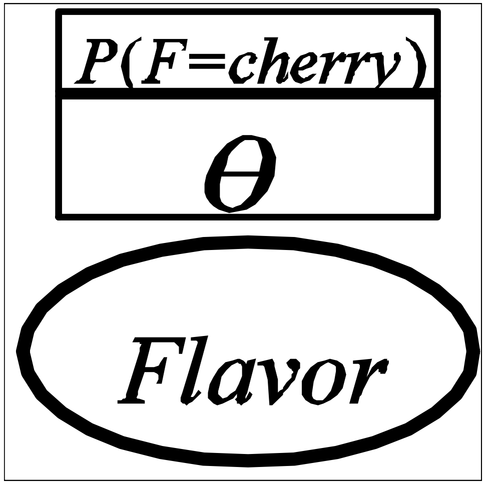
- ML prediction is less accurate than Bayesian and MAP
  - Ignores prior information
  - Relies only on one hypothesis h_M
- ML, MAP and Bayesian predictions converge as data increases
- Subject to overfitting
  - Does not penalize complex hypothesis
- Finding h_{ML} is often easier than h_{MAP}
  - $h_{ML} = \text{argmax}_j \sum_i \log P(d_i | h_j)$

Learning with complete data

- Parameter learning with complete data
  - Parameter learning task involves finding numerical parameters for a probability model whose structure is fixed
  - Example
    - Learning CPT for a Bayes net with a given structure

Simple ML Example

- Hypothesis h_θ
  - P(cherry)=θ and P(lime)=1-θ
  - θ is our parameter
- Data d:
  - N candies (c cherry and l=N-c lime)
- What should θ be?



Simple ML example

- Likelihood of this particular data set
  - $P(d|h_\theta)=\theta^c(1-\theta)^l$
  - ML hypothesis is one that maximizes the above expression
    - Equivalent to maximizing log likelihood

- Log likelihood
  - $L(d|h_\theta)=\log P(d|h_\theta)=c \log \theta + l \log (1-\theta)$

Simple ML example

- Find θ that maximizes log likelihood

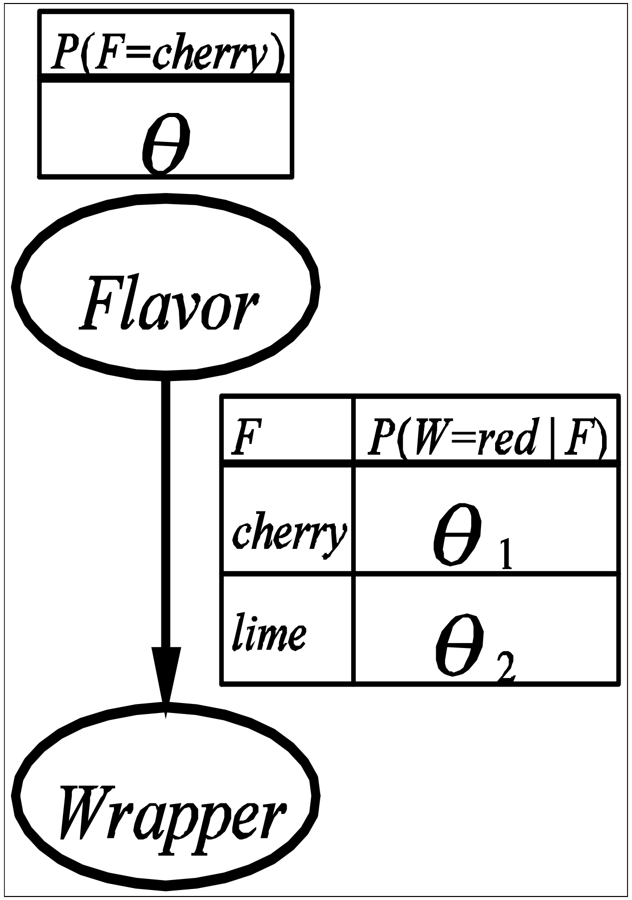
$$\frac{\partial L(d|h_\theta)}{\partial \theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0$$

$$\theta = \frac{c}{c+l} = \frac{c}{N}$$

- ML hypothesis asserts that actual proportion of cherries is equal to observed proportion

More complex ML example

- Hypothesis: $h_{\theta, \theta_1, \theta_2}$
- Data:
  - c cherries
    - G_c green wrappers
    - R_c red wrappers
  - l limes
    - G_l green wrappers
    - R_l red wrappers



More complex ML example

- $P(d|h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^l \theta_1^{R_c} (1-\theta_1)^{G_c} \theta_2^{R_l} (1-\theta_2)^{G_l}$
- $L = [c \log\theta + l \log(1-\theta)] + [R_c \log\theta_1 + G_c \log(1-\theta_1)] +$

  $[R_l \log\theta_2 + G_l \log(1-\theta_2)]$
- Take derivatives with respect to each parameter and set to zero
  - $\theta = c/(c+l)$
  - $\theta_1 = R_c/(R_c+G_c)$
  - $\theta_2 = R_l/(R_l+G_l)$

ML Comments

- This approach can be extended to any Bayes net whose conditional probabilities are represented as tables

- With complete data

1. ML parameter learning problem decomposes into separate learning problems, one for each parameter!

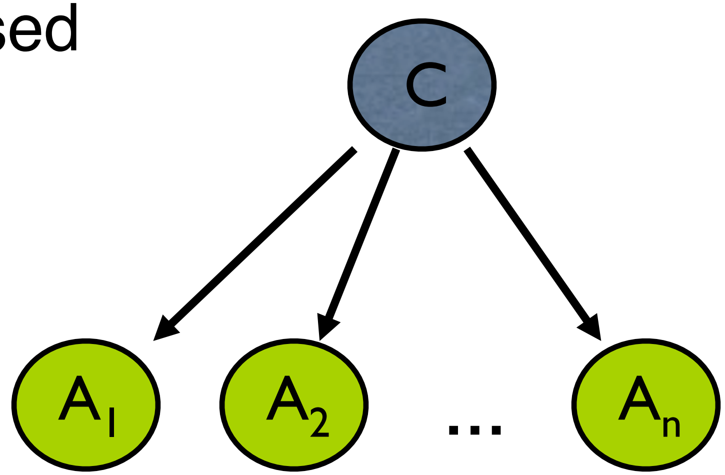
2. Parameter values for a variable, given its parents are just observed frequencies of variable values for each setting of parent values!

A problem: Zero probabilities

- What happens if we observed zero cherry candies?
  - θ would be set to 0
  - Is this a good prediction?
- Laplace smoothing
  - Instead of $\theta = c/(c+l)$ use $\theta=(c+1)/(c+l+2)$

Naïve Bayes model

- Want to predict a class C based on attributes A_i
- Parameters:
  - θ =P(C=true)
  - $\theta_{j,1}$=P(A_j=true|C=true)
  - $\theta_{j,2}$=P(A_j=true|C=false)



- Assumption: A_i's are independent given C

Naïve Bayes Model

- With observed attribute values $x_1, x_2, \ldots, x_n$
  - $P(C|x_1, x_2, \ldots, x_n) = \alpha\ P(C) \Pi_i\ P(x_i|C)$
- From ML we know what the parameters should be
  - Observed frequencies (with possible Laplace smoothing)
- Just need to choose the most likely class C

Naïve Bayes comments

- Naïve Bayes scales well
- Naïve Bayes tends to perform well
  - Even though the assumption that attributes are independent given class often does not hold
- Application
  - Text classification

Text classification

- Important practical problem, occurring in many applications
  - Information retrieval, spam filtering, news filtering, building web directories…
- Simplified problem description
  - Given: collection of documents, classified as "interesting" or "not interesting" by people
  - Goal: learn a classifier that can look at text of new documents and provide a label, without human intervention

Data representation

- Consider all possible significant words that can occur in documents
  - Words in English dictionary, proper names, abbreviations,…
- Do not include stopwords
  - Words that appear in all documents
    - E.g. prepositions, common verbs, “to be”, “to do”,…
- Stem words
  - Map words to their root
    - E.g. learn <-“learn”, “learning”, “learned”
- For each root, introduce common binary feature
  - specifying whether the word is present or not in the document

Example

- "Machine learning is fun"

Aardvark	0
M	
Fun	1
Funel	0
M	
Learn	1
M	
Machine	1
M	
Zebra	0

Use Naïve Bayes Assumption

- Words are independent of each other, given the class, y, of document

$$P(y|\text{document}) = \Pi_{i-1}^{|\text{Vocab}|} P(w_i|y)$$

How do we get the probabilities?

Use Naïve Bayes Assumption

- Words are independent of each other, given the class, y, of document

$$P(y|\text{document}) = \Pi_{i-1}^{|\text{Vocab}|} P(w_i|y)$$

- Use ML parameter estimation!
  - $P(w_i|y)$=(# documents of class y containing word w_i)/(# documents of class y)

- Count words over collections of documents
- Use Bayes rule to compute probabilities for unseen documents
- Laplace smoothing is very useful here

Observations

- We may not be able to find θ analytically

- **Gradient search** to find good value of θ
  - Start with guess θ
  - Update θ <- $\theta + \alpha \, \partial L(\theta \mid D)/\partial \theta$
    - α in (0,1) is learning rate or step size
  - Repeat until θ stops changing significantly

Conclusions

- What you should know
  - Bayesian learning
  - MAP
  - ML
  - How to learn parameters in Bayes Nets
  - Naïve Bayes assumption
  - Laplace smoothing