CS485/685 Machine Learning
Lecture 3: Jan 10, 2012

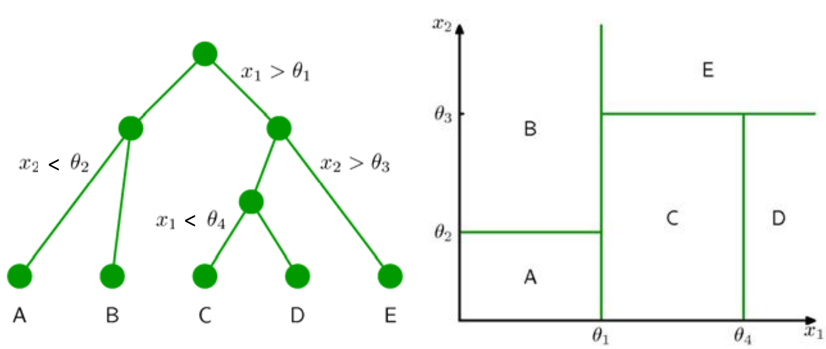
Nearest Neighbour and

Statistical Learning

[B] Section 2.5.2

Decision tree with continuous attributes

- Tree partitions the input space



Decision tree with continuous attributes

- How do we come up with good partitions?
- Common approach: thresholding
  - Single attribute: $x_j > \theta_j$
  - Multi-attribute: $f(x_1, \ldots, x_M) > \theta_j$
    - Where f can be linear or non-linear

Single Attribute Thresholding

- Idea:
  - Discretize continuous attribute into finite set of intervals.
  - Pick thresholds midway between pairs of consecutive values
- Example:

Full Tree

- In the limit, single attribute thresholding leads to a full tree with one example per leaf
  - Partition input space into bins or hypercubes
  - Future examples classified according to bins' labels
    - Close to "nearest neighbour" classification
- Picture:

Nearest Neighbour Classification

- Instead of building tree, find nearest neighbour

  $x^* = argmin_{x'}\ d(x, x')$

  Label: $y_x \leftarrow y_{x^*}$
- Distance measures: $d(x, x')$

  L_1: $d(x, x') = \sum_j^M |x_j - x_j'|$

  L_2: $d(x, x') = \sum_j^M |x_j - x_j'|^2$

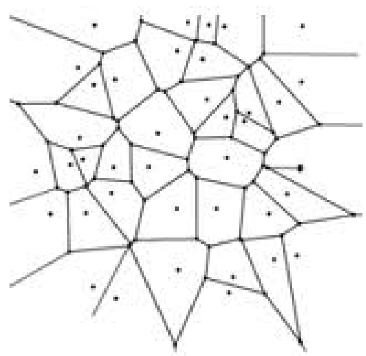
  …

  L_p: $d(x, x') = \sum_j^M |x_j - x_j'|^p$

  Weighted dimensions: $d(x, x') = \sum_j^M c_j |x_j - x_j'|^p$

Voronoi diagram

- Partition implied by nearest neighbour
  - Assuming Euclidean distance

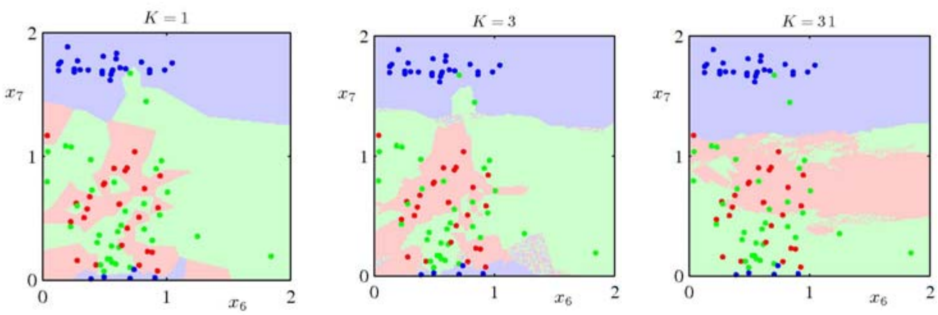


K-nearest neighbour

- Nearest neighbour often instable (overfitting)
- Idea: assign most frequent label among k-nearest neighbours
  - Let $knn(x)$ be the k-nearest neighbours of x according to distance d
  - Label: $y_x \leftarrow mode(\{y_{x'} | x' \in knn(x)\})$

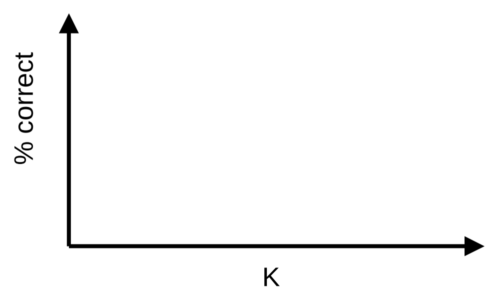
Effect of K

- K controls the degree of smoothing.
- Which partition do you prefer? Why?



Choosing K

- Best K depends on
  - Problem
  - Amount of training data
- Choose K by k-fold cross validation



Complexity

- Nearest neighbour computation:
  - Training: no computation (simply store examples)
  - Testing: return label of nearest example
- Complexity with respect to
  - N: size of training set
  - M: number of attributes

	Training	**Testing**
Decision tree		
Nearest neighbour		

Statistical Learning

- View: we have uncertain knowledge of the world
- Idea: **learning simply reduces this uncertainty**

Candy Example

- Favorite candy sold in two flavors:
  - Lime (hugh)
  - Cherry (yum)
- 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 don't know its flavor ratio
- After eating k candies:
  - What's the flavor ratio of the bag?
  - What will be the flavor of the next candy?

Statistical Learning

- **Hypothesis H:** probabilistic theory of 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
- **Examples E:** evidence about the world
  - e_1: 1st candy is cherry
  - e_2: 2nd candy is lime
  - e_3: 3rd candy is lime
  - …

Bayesian Learning

- **Prior:** $\Pr(H)$
- **Likelihood:** $\Pr(e|H)$
- **Evidence:** $\boldsymbol{e} = < e_1, e_2, \dots, e_N >$

- **Bayesian Learning** amounts to computing the posterior using Bayes' Theorem:

$$\Pr(H|\boldsymbol{e}) = k \Pr(\boldsymbol{e}|H)\Pr(H)$$

Terminology

- **Probability distribution:**
  - A specification of a probability for each event in our sample space
  - Probabilities must sum to 1
- Assume the world is described by two (or more) random variables
  - **Joint probability distribution**
    - Specification of probabilities for all combinations of events

Joint distribution

- Given two random variables A and B:
- Joint distribution:

  $\Pr(A = a \land B = b)$ for all a, b

- **Marginalisation (sumout rule):**

  $\Pr(A = a) = \Sigma_b \Pr(A = a \land B = b)$

  $\Pr(B = b) = \Sigma_a \Pr(A = a \land B = b)$

Example: Joint Distribution

	sunny		~sunny	
	cold	~cold	cold	~cold
headache	0.108	0.012	0.072	0.008
~headache	0.016	0.064	0.144	0.576

P(headache∧sunny∧cold) = P(~headache∧sunny∧~cold) =

P(headache∨sunny) =

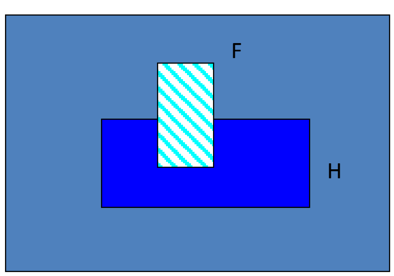
P(headache) =

marginalization

Conditional Probability

- $\Pr(A|B)$: fraction of worlds in which B is true that also have A true

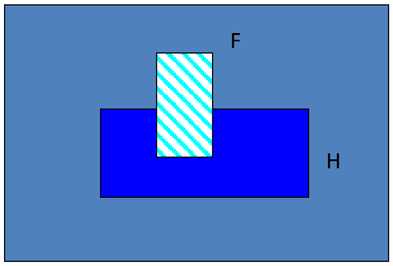
H="Have headache"
F="Have Flu"



$$\Pr(H) = 1/10$$
$$\Pr(F) = 1/40$$
$$\Pr(H|F) = 1/2$$

Headaches are rare and flu is rarer, but if you have the flu, then there is a 50-50 chance you will have a headache

Conditional Probability



H="Have headache"
F="Have Flu"

$\Pr(H) = 1/10$
$\Pr(F) = 1/40$
$\Pr(H|F) = 1/2$

$\Pr(H|F)$ = Fraction of flu inflicted worlds in which you have a headache

=(# worlds with flu and headache)/ (# worlds with flu)

= (Area of "H and F" region)/ (Area of "F" region)

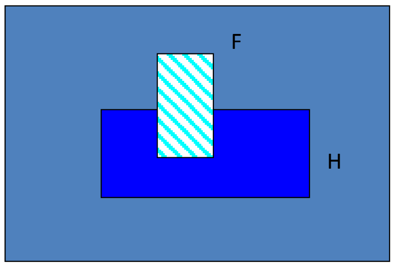
= $\Pr(H \wedge F) / \Pr(F)$

Conditional Probability

- Definition:
  $$\Pr(A|B) = \Pr(A \wedge B) / \Pr(B)$$
- Chain rule:
  $$\Pr(A \wedge B) = \Pr(A|B) \Pr(B)$$

Memorize these!

Inference



One day you wake up with a headache. You think "Drat! 50% of flues are associated with headaches so I must have a 50-50 chance of coming down with the flu"

H="Have headache"
F="Have Flu"

$\Pr(H) = 1/10$
$\Pr(F) = 1/40$
$\Pr(H|F) = 1/2$

Is your reasoning correct?

$\Pr(F \wedge H) =$

$\Pr(F|H) =$

Example: Joint Distribution

sunny

	cold	~cold
headache	0.108	0.012
~headache	0.016	0.064

~sunny

	cold	~cold
headache	0.072	0.008
~headache	0.144	0.576

$\Pr(headache \wedge cold \mid sunny) =$

$\Pr(headache \wedge cold \mid {\sim}sunny) =$

Bayes Rule

- Note
  $\Pr(A|B)\Pr(B) = \Pr(A \land B) = \Pr(B \land A) = \Pr(B|A)Pr(A)$

- Bayes Rule
  $\Pr(B|A) = [(\Pr(A|B)\Pr(B)]/\Pr(A)$

Memorize this!

Using Bayes Rule for inference

- Often we want to form a hypothesis about the world based on what we have observed
- Bayes rule is vitally important when viewed in terms of stating the belief given to hypothesis H, given evidence e

Likelihood

Prior probability

$$P(H|e) = \frac{P(e|H)P(H)}{P(e)}$$

Posterior probability

Normalizing constant

Bayesian Learning

- **Prior:** $\Pr(H)$
- **Likelihood:** $\Pr(e|H)$
- **Evidence:** $\boldsymbol{e} = < e_1, e_2, \ldots, e_N >$

- **Bayesian Learning** amounts to computing the posterior using Bayes' Theorem:
  $$\Pr(H|\boldsymbol{e}) = k \Pr(\boldsymbol{e}|H)\Pr(H)$$

Bayesian Prediction

- Suppose we want to make a prediction about an unknown quantity X (i.e., the flavor of the next candy)

- $$\begin{aligned}\Pr(X|\boldsymbol{e}) &= \Sigma_i \Pr(X|\boldsymbol{e}, h_i)P(h_i|\boldsymbol{e}) \\ &= \Sigma_i \Pr(X|h_i)P(h_i|\boldsymbol{e})\end{aligned}$$

- Predictions are weighted averages of the predictions of the individual hypotheses
- Hypotheses serve as "intermediaries" between raw data and prediction

Candy Example

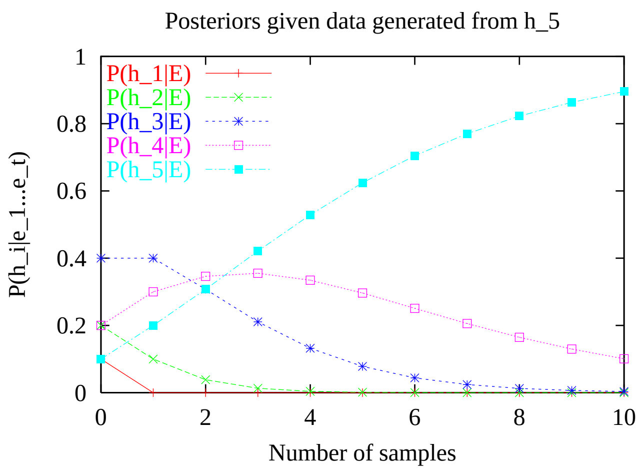
- Assume prior $\Pr(H) = < 0.1, 0.2, 0.4, 0.2, 0.1 >$
- Assume candies are **i.i.d. (identically and independently distributed)**

$$\Pr(\boldsymbol{e}|h) = \Pi_n P(e_n|h)$$

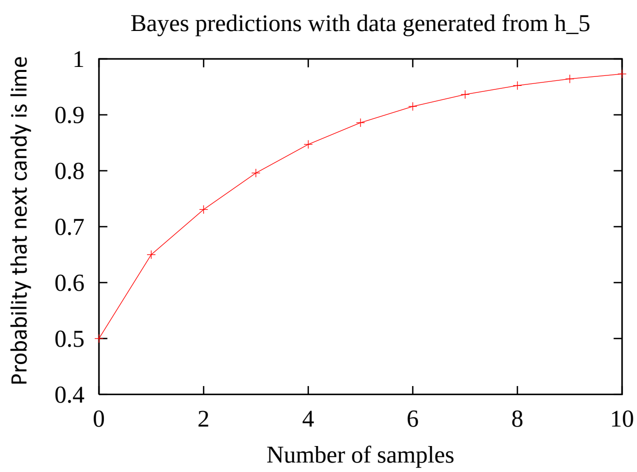
- Suppose first 10 candies all taste lime:

$$\Pr(\boldsymbol{e}|h_5) =$$
$$\Pr(\boldsymbol{e}|h_3) =$$
$$\Pr(\boldsymbol{e}|h_1) =$$

Posterior



Prediction



Bayesian Learning

- Bayesian learning properties:
  - **Optimal** (i.e. given prior, no other prediction is correct more often than the Bayesian one)
  - **No overfitting** (all hypotheses considered and weighted)
- There is a price to pay:
  - When hypothesis space is large Bayesian learning may be intractable
  - i.e. sum (or integral) over hypothesis often intractable
- Solution: approximate Bayesian learning

Maximum a posteriori (MAP)

- Idea: make prediction based on **most probable hypothesis** h_{MAP}

$$h_{MAP} = argmax_{h_i} \Pr(h_i|\boldsymbol{e})$$
$$\Pr(X|\boldsymbol{e}) \approx \Pr(X|h_{MAP})$$

- In contrast, Bayesian learning makes prediction based on **all** hypotheses weighted by their probability

Candy Example (MAP)

- Prediction after
  - 1 lime: $h_{MAP} = h_3$, $\Pr(lime|h_{MAP}) = 0.5$
  - 2 limes: $h_{MAP} = h_4$, $\Pr(lime|h_{MAP}) = 0.75$
  - 3 limes: $h_{MAP} = h_5$, $\Pr(lime|h_{MAP}) = 1$
  - 4 limes: $h_{MAP} = h_5$, $\Pr(lime|h_{MAP}) = 1$
  - …
- After only 3 limes, it correctly selects h_5

Candy Example (MAP)

- But what if correct hypothesis is h_4?
  - h_4: $\Pr(lime) = 0.75$ and $\Pr(cherry) = 0.25$
- After 3 limes
  - MAP incorrectly predicts h_5
  - MAP yields $\Pr(lime|h_{MAP}) = 1$
  - Bayesian learning yields $\Pr(lime|\boldsymbol{e}) = 0.8$

MAP properties

- MAP prediction **less accurate** than Bayesian prediction since it relies only on **one** hypothesis h_{MAP}
- But MAP and Bayesian predictions converge as data increases
- **Controlled overfitting** (prior can be used to penalize complex hypotheses)
- **Finding h_{MAP} may be intractable:**
  - $h_{MAP} = argmax_h \Pr(h|\boldsymbol{e})$
  - Optimization may be difficult

MAP computation

- Optimization:

$$\begin{aligned} h_{MAP} &= argmax_h \Pr(h|\boldsymbol{e}) \\ &= argmax_h \Pr(h) \Pr(\boldsymbol{e}|h) \\ &= argmax_h \Pr(h) \Pi_n \Pr(e_n|h) \end{aligned}$$

- Product induces non-linear optimization
- Take the log to linearize optimization

$$h_{MAP} = argmax_h \log \Pr(h) + \Sigma_n \log P(e_n|h)$$

Maximum Likelihood (ML)

- Idea: simplify MAP by assuming uniform prior (i.e., $\Pr(hi) = \Pr(h_j) \; \forall i, j$)
  - $h_{MAP} = argmax_h \Pr(h) \Pr(\boldsymbol{e}|h)$
  - $h_{ML} = argmax_h \Pr(\boldsymbol{e}|h)$
- Make prediction based on h_{ML} only:
  - $\Pr(X|\boldsymbol{e}) \approx \Pr(X|h_{ML})$

Candy Example (ML)

- Prediction after
  - 1 lime: $h_{ML} = h_5$, $\Pr(lime|h_{ML}) = 1$
  - 2 limes: $h_{ML} = h_5$, $\Pr(lime|h_{ML}) = 1$
  - …
- **Frequentist: "objective"** prediction since it relies only on the data (i.e., no prior)
- **Bayesian:** prediction based on data and uniform prior (since no prior ≡ uniform prior)

ML properties

- ML prediction **less accurate** than Bayesian and MAP predictions since it ignores prior info and relies only on **one** hypothesis h_{ML}
- But ML, MAP and Bayesian predictions converge as data increases
- Subject to **overfitting** (no prior to penalize complex hypothesis that could exploit statistically insignificant data patterns)
- Finding h_{ML} is often easier than h_{MAP}

$$h_{ML} = argmax_h \Sigma_n \log \Pr(e_n|h)$$