BAYESIAN LEARNING, CONT:

Danica Kragic

Most Probable Classification of New Instances

- So far we’ve sought the most probable hypothesis given the data $D$ (i.e., $h_{MAP}$).
- Assumption: training set consists of instances described as conjunctions of attribute values, target classification based on finite set of classes $V$.
- Task: predict the correct class for a new instance $x$, $x = <a_1, a_2, ..., a_n>$.

Consider:
- Three possible hypotheses:
  - $P(h_1|D) = 4$, $P(h_2|D) = .3$, $P(h_3|D) = .3$.
  - Given new instance $x$, $h_1(x) = +$, $h_2(x) = -$, $h_3(x) = -$.
- What’s most probable classification of $x$?

Bayesian Classification

- Instead of asking “What is the most probable hypothesis given training data?” ask.
- “What is the most probable classification of the new instance given training data?”
- Instead of learning “hypothesis function”, $h$, the Bayes optimal classifier assigns the most probable class to the input data.

Types:
- Bayes Optimal Classifier
- Gibbs Classifier
- Naive Bayes Classifier
- Bayesian Belief Network (Bayes Net)

Bayes Optimal Classifier

- We want to determine the most probable classification based on the combined prediction of all hypotheses, weighted by their posterior probabilities.
- Let $V$ be a set of all possible classifications.

$$P(v|D) = \sum_{h \in H} P(v|h)P(h|D)$$

- Bayes Optimal Classification

$$v = \arg \max_{v \in V} P(v|D) = \arg \max_{v \in V} \sum_{h \in H} P(v|h)P(h|D)$$

- also known as model averaging - averaging predictions of lots of models based on the posterior probability of the model’s parameters.

Simple Bayesian Reasoning

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

$$P(D) = \sum_{h \in H} P(D|h)P(h)$$

- For each hypothesis $h$, we need to know the probability of generating any combination of data $D$.
- The number of all possible data sets is exponential in the number of basic data attributes.
- Requires huge amount of data usually not available (sparse data problem).
Bayes Optimal Classifier

Bayes optimal classification:

\[ \text{arg max}_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) \]

Example:

- \( P(h_1|D) = 0.4 \)
- \( P(-h_1|D) = 0 \)
- \( P(h_2|D) = 0.6 \)
- \( P(-h_2|D) = 0 \)

therefore

\[ \sum_{h \in H} P(+h_i|P(h_i|D)) = 0.4, \sum_{h \in H} P(-h_i|P(h_i|D)) = 0.6 \]

and

\[ \text{arg max}_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = - \]

Gibbs Classifier

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.

Gibbs algorithm:

1. Choose one hypothesis at random, according to \( P(h_i|D) \)
2. Use this to classify new instance

Surprising fact: Assume target concepts are drawn at random from \( H \) according to priors on \( H \). Then:

\[ E[\text{error Gibbs}] \leq 2 E[\text{error Bayes Optimal}] \]

- Its expected error no worse than twice Bayes optimal

Naive Bayes Classifier

Assume discrete target function \( f(x) : X \rightarrow V \), where each instance \( x \) described by attributes \( (a_1, a_2, \ldots, a_n) \) and \( f(x) \) takes any value from finite set \( V \).

Most probable target value of \( h \) is:

\[ \text{max}_{v_j \in V} \prod_{i \in C} P(a_i|v_j) \]

Naive Bayes classifier:

\[ v_{\text{naive}} = \text{max}_{v_j \in V} \prod_{i \in C} P(a_i|v_j) \]

Example

Naive Bayes assumption:

\[ P(a_1, a_2, \ldots, a_n|v_j) = \prod_{i} P(a_i|v_j) \]

Which gives

Naive Bayes classifier:

\[ v_{\text{naive}} = \text{max}_{v_j \in V} P(v_j) \prod P(a_i|v_j) \]
Consider PlayTennis again, and new instance

(Outlook = sun, Temp = cool, Humid = high, Wind = strong)

Want to compute:

\[ v_{NB} = \arg\max_v P(v) \prod_i P(a_i | v) \]

- playtennis (9+,5-): \( P(\text{yes}) = 9/14, P(\text{no}) = 5/14 \)
- wind = strong (3+, 3-): \( P(\text{strong} | \text{yes}) = 3/9, P(\text{strong} | \text{no}) = 3/5 \)
- 

\[ P(y)P(\text{sun} | y)P(\text{cool} | y)P(\text{high} | y)P(\text{strong} | y) = 0.005 \]

\[ P(y)P(\text{sun} | y)P(\text{cool} | y)P(\text{high} | y)P(\text{strong} | y) = 0.021 \]

\[ \rightarrow v_{NB} = \text{no} \]

### Naive Bayes: Estimating Probabilities

What if none of the training instances with target value \( v \) have attribute value \( a \)? Then

\[ P(a | v) = 0 \quad \text{and} \quad P(v) \prod_i P(a_i | v) = 0 \]

Typical solution is Bayesian estimate for \( P(a | v) \): \( m \)-estimate of probability

\[ P(a | v) \approx \frac{n_i + mp}{n + mp} \]

where

- \( m \) is the total number of training examples for which \( v = v_j \).
- \( n_i \) number of examples for which \( i = v_j \) and \( a = a_i \).
- \( p \) is prior estimate for \( P(a | v) \).
- \( n \) is weight given to prior - how heavily to weight \( p \) relative to the observed data
- \( p \cdot n \) if no additional knowledge, set \( p = 1/k \), where \( k \) is the number of values an attribute can have.

### Naive Bayes: Independence Violation

- Conditional independence assumption is often violated

\[ P(a_1, a_2, \ldots, a_n | v) = \prod_i P(a_i | v) \]

- ...but it works surprisingly well anyway.
- Note! Don’t need estimated posteriors \( P(v_j | x) \) to be correct; need only

\[ \arg\max_{v_j} \prod_i P(a_i | v_j) = \arg\max_{v_j} P(v_j)P(a_1, \ldots, a_n | v_j) \]

- Naive Bayes posteriors often unrealistically close to 1 or 0

### Learning to Classify Email

Target concept \( SVM^+ \): \{+, -\}

1. Represent each email by vector of words
   - One attribute per word position in document
2. Learning: Use training examples to estimate
   - \( P(+) \)
   - \( P(-) \)

Naive Bayes conditional independence assumption

\[ P(\text{email} | v) = \prod_{i \in \text{positions}} P(a_i = v_i | v) \]

where \( P(a_i = v_i | v) \) is probability that word in position \( i \) is \( v_i \), given \( v \)

one more assumption: \( P(a_i = v_i | v) = P(a_i = v_i), \forall i, m \)

### Naive Bayes: Text Classification

```
Learn naïve Bayes, text(Examples, V)
1. collect all words and other tokens that occur in Examples
   • Vocabulary -- all distinct words and other tokens in Examples
2. calculate the required \( P(v_i) \) and \( P(w_i | v) \) probability terms
   • For each target value \( v_i \) in \( V \) do
     - \( \text{emails}_i \) -- subset of Examples for which the target value is \( v_i \)
     - \( P(v_i) = |\text{emails}_i| / n \)
     - \( \text{Test}_j \) -- a single document created by concatenating members of \( \text{emails}_i \)
     - \( n \) -- total number of word positions in all training examples whose target value is \( v_i \)
     - for each word \( w_k \) in Vocabulary
       - \( n_k \) -- number of times word \( w_k \) occurs in \( \text{Test}_j \)
       - \( P(w_i | v_i) = \frac{n_k + mp}{n + mp} \)
```

### Return the Estimated Target Value

Return the estimated target value for the email Email. \( a_i \) denotes the word found in the \( i \)th position within Email.

```
Classify naïve Bayes, text(Email)
• positions -- all word positions in Email that contain tokens found in Vocabulary
• Return \( v_{NB} \), where

\[ v_{NB} = \arg\max_{v_i} \prod_{i \in \text{positions}} P(a_i | v_i) \]
```
COMBINING CLASSIFIERS:

BAGGING and BOOSTING

Coping with “bad features”

- End result: a k-dimensional space,
  - in which each dimension is a feature
  - containing N (labelled) samples (objects)

Combine Classifiers

Choice: different training sets, different feature sets, ....

Discriminant functions

1. Choose class of decision functions in feature space.
2. Estimate the function parameters from the training set.
3. Classify a new pattern on the basis of this decision rule.

Computer Vision

- Sensors give measurements, which should be converted to features
  (can be pure measurements, e.g. pixels!)
- Ideally, a feature value is identical for all samples in one class
- However:
  - Measurement, discretisation noise
  - Variation between samples
  - Poor features
### Linear Discriminant Functions

A linear discriminant function is given by

\[ h(x) = w^T x + w_0 \]

where \( w \) determines the slope of the plane and \( w_0 \) determines the offset. If we define

\[ z = (1, x_1, ..., x_p)^T, \quad v = (w_0, w_1, ..., w_p)^T \]

then the function can be written as

\[ h(x) = v^T z. \]

---

### Classification problem

- Assume a set \( S \) of \( N \) instances \( x_i \in X \) each belonging to one of \( M \) classes \( \{c_1, ..., c_M\} \)
- The training set consists of pairs \( (x_i, c_i) \)
- A classifier \( C \) assigns a classification \( C(x) \in \{c_1, ..., c_M\} \) to an instance \( x \)
- The classifier learned in trial \( i \) is denoted \( C_i \) while \( C^* \) is the composite bagged or boosted classifier

### Bagging and Boosting

- Bagging replicates training sets by sampling with replacement from the training instances.
- Boosting uses all instances but weights them and therefore produces different classifiers.
- Classifiers are then combined by voting to create a composite classifier.
- Bagging: classifiers have same votes;
- Boosting: vote dependant on the classifiers’ accuracy

---

### Bagging

- For each trial, generate new training set of size \( N \) with replacements (multiple occurrence of instances)
- A classifier \( C_i \) is generated for each training set
- Final classifier \( C^* \) is formed by aggregating the \( T \) classifiers
- An instance \( x \) is classified by counting votes for which

\[ C_i(x) = k \]

and \( C^*(x) \) represents the class with most votes

### Boosting

- Bagging and Boosting aggregate multiple hypotheses generated by the same learning algorithm invoked over different distributions of training data [Breiman 1996, Freund and Schapire 1996]
- Bagging and Boosting generate a classifier with a smaller error on the training data as it combines multiple hypotheses which individually have a large error
Bagging

- For each trial $t=1,2,...,T$ create a bootstrap sample $S_t$.
- Obtain an hypothesis $C_t$ on the bootstrap sample $S_t$.
- The final classification is the majority class

$$c_{BA}(x) = \arg\max_{c\in C} \sum_{t=1}^{T} \delta(c_t, C_t(x))$$

Boosting

- Boosting maintains a weight $w_t$ for each instance $(x_i,\ldots,c_i)$ in the training set.
- The higher the weight $w_t$, the more the instance $x_i$ influences the next hypotheses learned.
- At each trial, the weights are adjusted to reflect the performance of the previously learned hypothesis, with the result that the weight of correctly classified instances is decreased and the weight of incorrectly classified instances is increased.

$$w_{t+1} = \frac{w_t}{N}$$

- Let $w^t_i$ denote the weight of an instance $x_i$ at trial $t$, for every $x_i$, $w^1_i = 1/N$.
- The weight $w^t_i$ reflects the importance (e.g. probability of occurrence) of the instance $x_i$ in the sample set $S_t$.
- At each trial $t = 1,\ldots,T$, an hypothesis $C^t$ is constructed from the given instances under the distribution $w^t$. This requires that the learning algorithm can deal with fractional examples.
Boosting

- The error of the hypothesis $C^i$ is measured with respect to the weights $\epsilon_i = \sum_{x \text{ such that } C_i(x) \neq c_i} w_i^t / \sum w_i^t$
  
  $$\alpha_i = \frac{1}{2} \ln \left( \frac{1 - \epsilon_i}{\epsilon_i} \right)$$

- Update the weights $w_i^t$ of correctly and incorrectly classified instances by
  
  $$w_i^{t+1} = w_i^t \cdot e^{-\epsilon_i} \text{ if } C^i(x) = c_i$$
  $$w_i^{t+1} = w_i^t \cdot e^{\epsilon_i} \text{ if } C^i(x) \neq c_i$$

- Afterwards normalize the $w_i^{t+1}$ such that they form a proper distribution
  
  $$\sum_i w_i^{t+1} = 1$$

Bayes MAP Hypothesis

- Bayes MAP hypothesis for two classes $x$ and $o$
- red: incorrect classified instances

Boosted Bayes MAP Hypothesis

- Boosted Bayes MAP hypothesis has more complex decision surface than individual hypothesis alone
Solution to the Prince and King problem

Under assumption that prince chooses door A initially and that king opens door B

The priori probability that the princess is behind any door X, \( P(X) = 1/3 \)

The probability that King opens door B if the princess was behind A, \( P(\text{King opens B} \mid A) = 1/2 \)

The probability that King opens door B if the princess was behind B, \( P(\text{King opens B} \mid B) = 0 \)

The probability that King opens door B if the princess was behind C, \( P(\text{King opens B} \mid C) = 1 \)

The probability that King opens door B is then

\[
P(\text{King opens B}) = P(A)\frac{P(K.B \mid A)}{P(B)} + P(B)\frac{P(K.B \mid B)}{P(B)} + P(C)\frac{P(K.B \mid C)}{P(B)} = 1/6 + 0 + 1/3 = 1/2
\]

- Then, by Bayes’ Theorem,
  \[
P(A \mid K.B) = \frac{P(A)\frac{P(K.B \mid A)}{P(B)}}{P(K.B)} = \frac{1/6}{1/2} = 1/3
  \]
  \[
P(C \mid K.B) = \frac{P(C)\frac{P(K.B \mid C)}{P(B)}}{P(K.B)} = \frac{1/3}{1/2} = 2/3
  \]

In other words, the probability that the princess is behind door C is higher when King opens door B, and prince SHOULD switch!