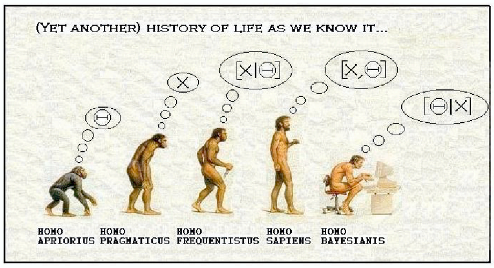
Bayesian Learning (cont.)

Lecture 7, DD2431 Machine Learning

Hedvig Kjellström
071121

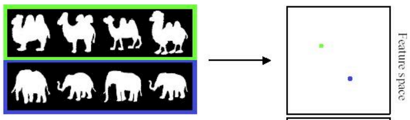
Bayesian Reasoning - a New Way of Thinking



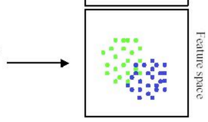
Classification Revisited

Sources of Noise

- 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*

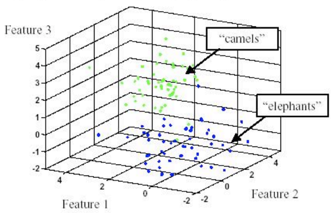


- However:
  - Measurement, discretisation noise
  - Variation between samples
  - Poor features



Feature Space

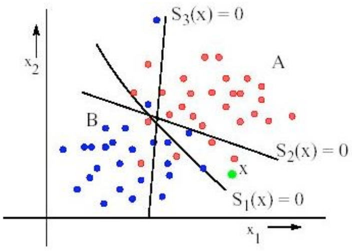
- End result: a *k*-dimensional space,
  - in which each dimension is a **feature**
  - containing *N* (labelled) **samples** (objects)



Problem 1: Large Feature Space

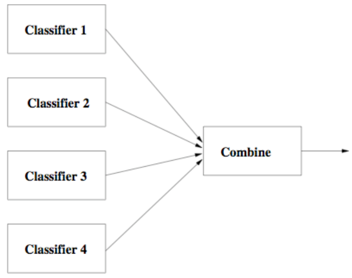
- Size of feature space exponential in number of features |x|.
- More features mean better description of the objects, but also larger feature space:
  Difficult to model likelihood P(v|x) in a large=highdim space.
- One solution is to look at parts of the feature space:
  Naive Bayes: Each feature separately

Problem 2: Non-Separable Classes



Problem 2: Non-Separable Classes

- One solution is to combine classifiers:



Coping with High Dimensionality

Naive Bayes Classifier

- One of the most common learning methods together with decision trees, neural networks and nearest neighbor.
- When to use:
  - Moderate or large training set available
  - Attributes x_i of a data instance x are conditionally independent given classification (or at least reasonably independent, works with a little dependence)
- Successful applications:
  - Medical diagnoses (symptoms independent)
  - Classification of text documents (words independent)

Naive Bayes Classifier

- An instance x is described by attributes $<x_1,x_2,\ldots,x_k>$.
- As before, let V be the set of possible classes. The MAP estimate of v is:

$$v_{MAP} = \arg\max_{v_i \in V} P(v_i | x_1,x_2,\ldots,x_k)$$

$$= \arg\max_{v_i \in V} \frac{P(x_1,x_2,\ldots,x_k | v_i)\, P(v_i)}{P(x_1,x_2,\ldots,x_k)}$$

$$= \arg\max_{v_i \in V} P(x_1,x_2,\ldots,x_k | v_i)\, P(v_i)$$

- Naive Bayes assumption: $P(x_1,x_2,\ldots,x_k | v_i) = \prod_j P(x_j | v_i)$
- This gives a *naive Bayes estimate*:

$$v_{MAP} = \arg\max_{v_i \in V} P(v_i) \prod_j P(x_j | v_i)$$

Example: Play Tennis?

- Task: To tell whether I should go for tennis given the forecast.
- An instance x has attributes
  $<outlook \in \{sunny, overcast, rainy\}$,
  $temp. \in \{hot, mild, cool\}$,
  $humidity \in \{high, normal\}$,
  $windy \in \{false, true\}>$.
- Its class label v is the variable $play \in \{yes, no\}$.

ROYAL INSTITUTE OF TECHNOLOGY

Example: Play Tennis?

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes

outlook	temp.	humidity	windy	play
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

ROYAL INSTITUTE OF TECHNOLOGY

Example: Play Tennis?

- New instance x = <outlook=sunny, temp.=cool, humidity=high, windy=true>:

  $v_{MAP} = \arg\max_{v_i \in V} P(v_i) \prod_j P(x_j|v_i)$

  P(yes) P(sunny|yes) P(cool|yes) P(high|yes) P(true|yes) =
  9/14 2/9 3/9 3/9 3/9 = 0.005
  P(no) P(sunny|no) P(cool|no) P(high|no) P(true|no) =
  5/14 3/5 1/5 4/5 3/5 = 0.021

  $\Rightarrow v_{MAP} = no$

ROYAL INSTITUTE OF TECHNOLOGY

Naive Bayes: Independence Violation

- Conditional independence assumption:

  $P(x_1,x_2,\ldots,x_k|v_i) = \prod_j P(x_j|v_i)$

  often violated - but it works surprisingly well anyway!

- Note: Do not need the posterior estimates $\wp(v_i|x)$ to be correct, need only correct $v_{MAP} = \arg\max_{v_i \in V} \wp(v_i|x)$.

- Since dependencies ignored, naive Bayes posteriors often unrealistically close to 0 or 1. *(Different attributes say the same thing to a higher degree than we expect, since they are correlated in reality.)*

ROYAL INSTITUTE OF TECHNOLOGY

Naive Bayes: Estimating Probabilities

- What if non of the training instances with target value v_i have attribute $D_j = x_j$? Then:

  $\wp(x_j|v_i) = 0$ and $\wp(v_i) \prod_j \wp(x_j|v_i) = 0$

- A solution is to add as prior knowledge that $\wp(x_j|v_i)$ must be larger than 0 - *m-estimate of probability*:

  $$\wp(x_j|v_i) \leftarrow \frac{n_v + mp}{n + m}$$

  where
  n = total number of training samples with $v = v_i$
  n_v = total number of training samples with $v = v_i$ and $D_j = x_j$
  p = prior estimate of $\wp(x_j|v_i)$ (set to 1/k if no other info)
  m = weight given to prior estimate (in relation to data)

ROYAL INSTITUTE OF TECHNOLOGY

Example: Spam Detection

- Instances x are emails, that are classified as spam (v_1 = +) or not spam (v_2 = -). *(The random vector Email is denoted D.)*

- Email is represented by vector of words:
  One attribute x_j per word position in email.

- Assumptions:
  $P(D = x|v_i) = \prod_j P(D_j = x_j|v_i)$ (Naive Bayes)
  $P(D_i = x_m|v_i) = P(D_j = x_m|v_i)\ \forall\ i,j$ (Word order insignificant)

ROYAL INSTITUTE OF TECHNOLOGY

Example: Spam Detection

- Learn the probability terms $P(v_i)$ and $P(x_m|v_i)$:

```
LearnNaiveBayesText (Examples, V)
    Vocabulary ← all distinct words and tokens from dataset
    For each class v_i ∈ V
      Emails(i) ← subset of Examples classified as v_i
      P(v_i) ← |Emails(i)| / |Examples|
      text(i) ← concatenation of all members of Emails(i)
      n ← number of words in text(i)
      For each word x_m ∈ Vocabulary
         n_m ← number of times word occurs in text(i)
         P(x_m|v_i ) ← (n_m + 1) /  (n + [Vocabulary])
      End
    End
End
```

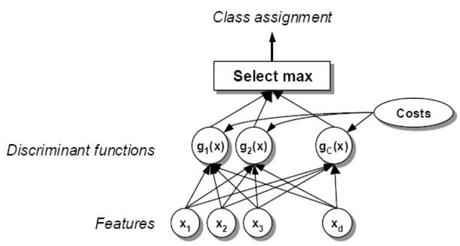
Example: Spam Detection

- Classify a new email instance Email as spam (v_1) or not (v_2):

ClassifyNaiveBayesText (Email)
  Positions ← all words and token positions in Email found in Vocabulary
  Email is a vector of words x_j, $j \in$ Positions
  $v_{NB} \leftarrow \arg\max_{vi \in V} P(v_i) \prod_{j \in Positions} P(x_j|v_i)$
End

Combination of Classifiers: Bayes Optimal and Gibbs

Combination of Classifiers



How weigh together the votes from the discriminant functions?

Bayes Optimal Classifier

- Weigh the votes according to the reliability $P(h_i|D)$ of each node h_i.
- Let H be the set of node outputs, and V be the set of all possible classifications from the Bayes optimal classifier:

$$P(v_j|D) = \sum_i P(v_j|h_i) P(h_i|D)$$

- Bayes optimal classification:

$$v_{MAP} = \arg\max_{vj \in V} P(v_j|D) = \arg\max_{vj \in V} \sum_i P(v_j|h_i) P(h_i|D)$$

Example from Mitchell

- Task: Predict the class $v \in \{+,-\}$ for a new instance x.
- Three possible hypotheses:
  - $P(h_1|D) = 0.4$
  - $P(h_2|D) = 0.3$
  - $P(h_3|D) = 0.3$
- Given a, the nodes return:
  - $h_1(x) = + \Rightarrow P(+|h_1) = 1, P(-|h_1) = 0$
  - $h_2(x) = - \Rightarrow P(+|h_2) = 0, P(-|h_2) = 1$
  - $h_3(x) = - \Rightarrow P(+|h_3) = 0, P(-|h_3) = 1$
- Since:
  - $P(+|x) = \sum_i P(+|h_i) P(h_i|x) = 0.4$
  - $P(-|x) = \sum_i P(-|h_i) P(h_i|x) = 0.6$

  the Bayes optimal classification of x is -.
- Different from just choosing the most reliable node.

Gibbs Classifier

- Bayes optimal classifier returns the best result, but expensive with many hypotheses.
- Gibbs classifier:
  Choose one hypothesis h_i at random, by Monte Carlo sampling according to reliability $P(h_i|D)$.
  Use this hypothesis so that $v = h_i$.
- Surprising fact: The expected error is equal to or less than twice the Bayes optimal error!
  $E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$

Combination of Classifiers: Bagging and Boosting

Bagging and Boosting

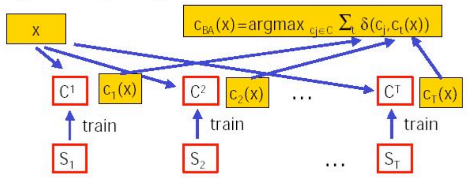
- Bagging and Boosting aggregate multiple hypotheses generated by instances of the same learning algorithm, trained with different selections of training data [Breiman 1996, Freund and Shapire 1995].
- Bagging and Boosting generate a classifier with small error by combining many weak (but easily computable) classifiers, each with large error individually.

Bagging and Boosting

- Bagging generates different training sets S_t by sampling with replacement from the original training set.
- Boosting uses all instances in S but weigh them in different ways for different classifiers.
- Classifiers are combined by voting:
  Bagging: classifiers have same votes.
  Boosting: vote dependent on classifiers' accuracy.

Bagging

- From the overall training S set randomly sample (with replacement) T different training sets $S_1,\ldots,S_T$ of size N.
- For each sample set S_t obtain a hypothesis C^t.
- To an unseen instance x assign the majority classification $c_{BA}(x)$ among the hypotheses C^t classifications $c_t(x)$.



Boosting

- Goes one step further than Bagging - uses performance of classifier C^t to improve classifier C^{t+1}.
- Maintains weight w_i^t for each training instance x_i.
- The higher the weight w_i^t, the more x_i influences the learning of C^t.
- At each trial t, weights are increased or decreased depending on if they are correctly or wrongly classified by C^t.

Boosting

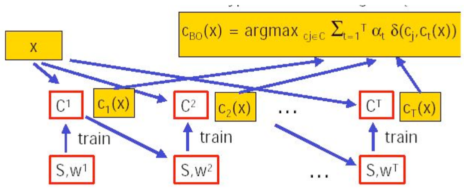
- AdaBoost:

Given: A training set $S = <(x_1,y_1),\ldots,(x_N,y_N)>$, $y_i \in \{-1, +1\}$
Initialize weights $w_i^1 = 1/N$
For trial $t = 1,\ldots,T$
  Train weak classifier C^t using weighted distribution w_i^t
  Compute error ε^t = share of x_i wrongly classified by C^t
  Compute weight $\alpha^t = 0.5 \ln((1 - \varepsilon^t)/ \varepsilon^t)$
  Compute weights $w_i^{t+1} \sim w_i^t \begin{cases} \exp(\alpha^t) & \textit{if } x_i \textit{ wrongly classified} \\ \exp(-\alpha^t) & \textit{if } x_i \textit{ correctly classified} \end{cases}$
  (Weight distribution is always normalized to sum to 1.)
End

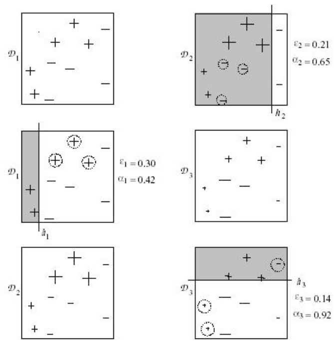
Combined classifier: $C^*(x) = \text{sign}(\sum_t \alpha^t C^t(x))$

Boosting

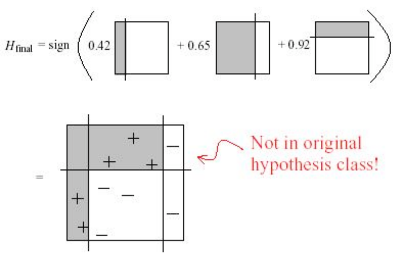
- For the more general multiclass case:



Toy Example

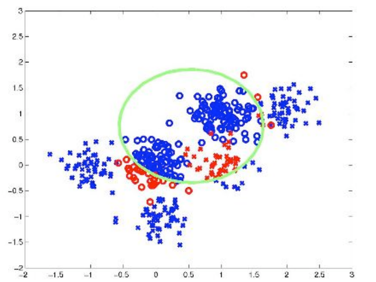


Toy Example



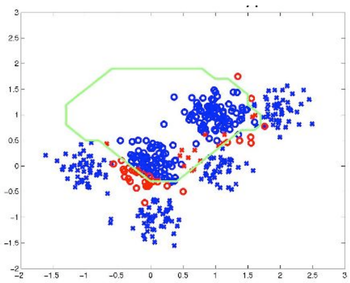
Bayes MAP Hypothesis

- Bayes MAP Hypothesis for two classes x and o.
- Red = wrongly classified instances.



Boosted Bayes MAP Hypothesis

- More complex decision surface than individual hypothesis alone.



Convergence and Generalization

- Bagging:
  - No proven convergence bound. *(Heuristically, the classifier must be reasonably "unstable" and dependent on the dataset, so that the T weak classifiers are different from each other.)*
  - No proven generalization error bound. *(That is, nothing can be promised about how the classifier handles previously unseen data.)*
- Boosting:
  - Combined classifier error decreases exponentially in AdaBoost for weak classifier errors $\varepsilon^t < 0.5$ (i.e. better than chance).
  - Generalization error is bounded by the training error with high probability. *(That is, the final classifier will with high probability perform well on any pattern the classifier has not seen before.)*

Summary

- Naive Bayes classifier: Treat all features/dimensions as independent conditioned on class.

- Bayes Optimal classifier: Combine different classifiers according to their prior reliability.
- Gibbs classifier: Probabilistic variant of BO where one of the classifiers are selected randomly.

- Bagging: Combine instances of the same (weak) classifier, trained with slightly different datasets.
- Boosting: Combine instances of the same (weak) classifier, trained with the same dataset but with different weights
  - Key idea: Iteratively select weights according to how ”hard” instances are to classify.