Lecture 13: Naive Bayes. Instance-based learning

- ♢ Naive Bayes learning
- ♢ k-Nearest Neighbor
- ♢ Locally weighted regression
- ♢ Case-based reasoning
- ♢ Lazy and eager learning

Naive Bayes Classifier

Assume target function $f : X \to V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$.

Most probable value of $f(x)$ is:

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n)$$

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)}$$
$$= \arg\max_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

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: $v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

Naive Bayes Classifier

Along with decision trees, neural networks, nearest neighbor, it is one of the most practical learning methods!

When to use it:

- A moderate or large training set is available (need enough data to get reliable probability estimates)
- The attributes that describe the instances are conditionally independent given the classification

Successful applications:

- Diagnosis (medical and other)
- Classifying text documents

Naive Bayes Algorithm

Naive_Bayes_Learn($examples$)

For each target value v_j

$\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$

For each attribute value a_i of each attribute a

$\hat{P}(a_i | v_j) \leftarrow$ estimate $P(a_i | v_j)$

It is easy to estimate these probabilities just by counting!

Classify_New_Instance(x)

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i | v_j)$$

Naive Bayes: Example

Consider $PlayTennis$ again, and new instance

$$\langle Outlk = sun, Temp = cool, Humid = high, Wind = strong \rangle$$

Want to compute:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$$

$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$$

$$\rightarrow v_{NB} = n$$

Typical solution is Bayesian estimate for $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- n is number of training examples for which $v = v_j$,
- n_c number of examples for which $v = v_j$ and $a = a_i$
- p is prior estimate for $\hat{P}(a_i|v_j)$
- m is weight given to prior (i.e. number of "virtual" examples)

Naive Bayes: Subtleties

1. Conditional independence assumption is often violated

   $$P(a_1, a_2 \ldots a_n|v_j) = \prod_i P(a_i|v_j)$$

   But it works surprisingly well anyway! Note that we do not need the estimated posteriors $\hat{P}(v_j|x)$ to be correct; we need only that

   $$\arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = \arg\max_{v_j \in V} P(v_j) P(a_1 \ldots, a_n|v_j)$$

   Naive Bayes posteriors are often unrealistically close to 1 or 0

2. What if none of the training instances with target value v_j have attribute value a_i? Then

   $\hat{P}(a_i|v_j) = 0$, and... $\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$

Learning to Classify Text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text documents?

Learning to Classify Text

Target concept $Interesting? : Document \rightarrow \{+, -\}$

1. Represent each document by vector of words: one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position i is w_k, given v_j

One more assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Using the Naive Bayes Classifier

Input: a new document Doc]

1. $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
2. Return v_{NB}, where

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Naive Bayes Learning for Text

Input: $Examples$ (the set of documents), V (the appropriate classifications)

1. Collect all words and other tokens that occur in $Examples$ into a $Vocabulary$
2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms, as follows: for each target value v_j in V do
   - $docs_j \leftarrow$ subset of $Examples$ for which the target value is v_j
   - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
   - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
   - $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
   - for each word w_k in $Vocabulary$
     - $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
     - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

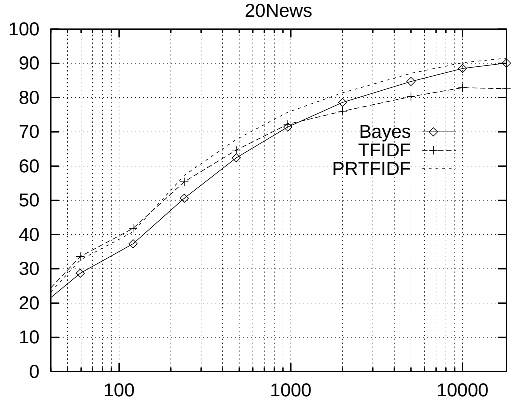
Twenty NewsGroups

Given 1000 training documents from each group, learn to classify new documents according to which newsgroup they came from

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	
talk.politics.guns	

Naive Bayes: 89% classification accuracy

Learning Curve for 20 Newsgroups



Accuracy vs. Training set size (1/3 withheld for test)

When To Consider Nearest Neighbor

- Instances map to points in $\Re^n$
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Don't lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

Instance-Based Learning

Key idea: just store all training examples $\langle x_i, f(x_i) \rangle$

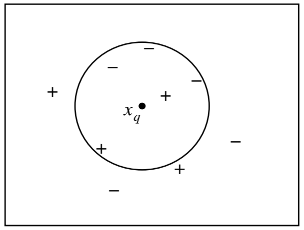
Nearest neighbor: Given query instance x_q, first locate nearest training example x_n, then estimate $\hat{f}(x_q) \leftarrow f(x_n)$

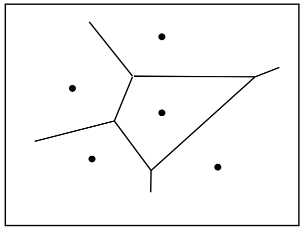
k-Nearest neighbor:

- Take vote among its k nearest neighbours (if discrete-valued target function)
- Take mean of f values of k nearest neighbours (if real-valued)

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Voronoi Diagram





Behavior in the Limit

Consider $p(x)$ defines probability that instance x will be labeled 1 (positive) versus 0 (negative).

Nearest neighbor:

- As number of training examples $\to \infty$, approaches Gibbs Algorithm

  Gibbs: with probability $p(x)$ predict 1, else 0

k-Nearest neighbor:

- As number of training examples $\to \infty$ and k gets large, approaches Bayes optimal

  Bayes optimal: if $p(x) > .5$ then predict 1, else 0

Note Gibbs has at most twice the expected error of Bayes optimal

Curse of Dimensionality

Imagine instances described by 20 attributes, but only 2 are relevant to target function

Curse of dimensionality: nearest neighbour is easily mislead when high-dimensional X

One approach (Moore & Lee, 1994):

- Stretch jth axis by weight z_j, where $z_1, \ldots, z_n$ chosen to minimize prediction error
- Use cross-validation to automatically choose weights $z_1, \ldots, z_n$

Note setting z_j to zero eliminates this dimension altogether

Distance-Weighted kNN

Might want weight nearer neighbors more heavily...

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

and $d(x_q, x_i)$ is distance between x_q and x_i

Note now it makes sense to use *all* training examples instead of just k (Shepard's method)

Locally Weighted Regression

Note kNN forms local approximation to f for each query point x_q

Why not form an explicit approximation $\hat{f}(x)$ for region surrounding x_q

- Fit linear function to k nearest neighbors
- Fit quadratic, ...
- Produces "piecewise approximation" to f

Error functions

- Squared error over k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in\ k\ nearest\ nbrs\ of\ x_q} (f(x) - \hat{f}(x))^2$$

- Distance-weighted squared error over all neighbours

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2\ K(d(x_q, x))$$

- Other schemes are possible too

Note that Radial Basis functions (RBFs) are also locally weighted approximations

Case-Based Reasoning

Can apply instance-based learning even when $X \neq \Re^n$, we just need a different "distance" metric

Case-Based Reasoning is instance-based learning applied to instances with symbolic logic descriptions

```
((user-complaint error53-on-shutdown)
 (cpu-model PowerPC)
 (operating-system Windows)
 (network-connection PCIA)
 (memory 48meg)
 (installed-applications Excel Netscape VirusScan)
 (disk 1gig)
 (likely-cause ???))
```

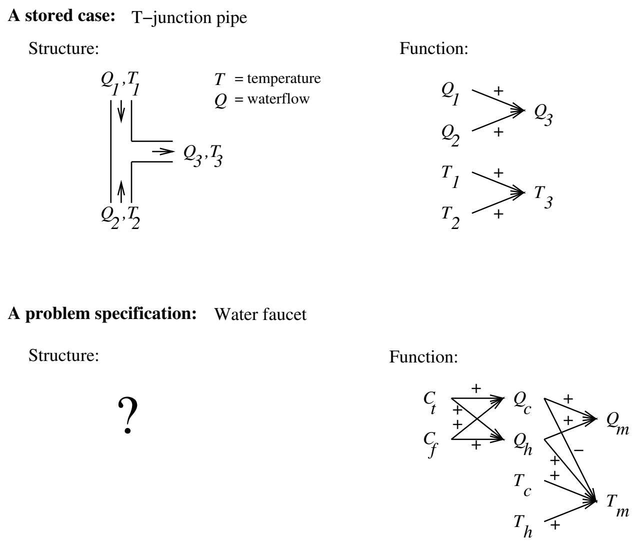
Case-Based Reasoning in CADET

CADET: 75 stored examples of mechanical devices

- each training example: $\langle$ qualitative function, mechanical structure$\rangle$
- new query: desired function,
- target value: mechanical structure for this function

Distance metric: match qualitative function descriptions

Case-Based Reasoning in CADET



Case-Based Reasoning in CADET

- Instances represented by rich structural descriptions
- Multiple cases retrieved (and combined) to form solution to new problem
- Tight coupling between case retrieval and problem solving

Bottom line:

- Simple matching of cases useful for tasks such as answering help-desk queries
- Area of ongoing research

Lazy and Eager Learning

Lazy: wait for query before generalizing

E.g. k-Nearest Neighbor, Case based reasoning

Eager: generalize before seeing query

E.g. Radial basis function networks, Decision trees, Backpropagation, Naive Bayes, ...

Does it matter?

- Eager learner must create global approximation
- Lazy learner can create many local approximations
- If they use same hypothesis space H, a lazy learner can represent more complex functions (e.g., consider $H =$ linear functions)