Chapter 6: Bayesian Learning (Part 2)

CS 536: Machine Learning

Littman (Wu, TA)

Bayesian Learning

[Read Ch. 6, except 6.3]

[Suggested exercises: 6.1, 6.2, 6.6]

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classier
- Naïve Bayes learner (today)
- Example: Learning over text data (today)
- Bayesian belief networks (skim)
- Expectation Maximization algorithm (later)

Naïve Bayes Classifier

Along with decision trees, neural networks, *k*NN, one of the most practical and most used learning methods.

When to use:

- Moderate or large training set available
- Attributes that describe instances are conditionally independent given classification

Successful applications:

- Diagnosis
- Classifying text documents

Naïve Bayes Classifier

Assume target function $f : X \rightarrow V$, where each instance x described by attributes $<a_1, a_2 \ldots a_n>$.

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

$$v_{MAP} = \text{argmax}_{v_j \text{ in } V} \text{P}(v_j | a_1, a_2 \ldots a_n)$$

$$= \text{argmax}_{v_j \text{ in } V} \text{P}(a_1, a_2 \ldots a_n, | v_j) \text{P}(v_j) / \text{P}(a_1, a_2 \ldots a_n)$$

$$= \text{argmax}_{v_j \text{ in } V} \text{P}(a_1, a_2 \ldots a_n, | v_j) \text{P}(v_j)$$

Naïve Bayes Assumption

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

which gives

Naïve Bayes classifier:

$$v_{NB} = \text{argmax}_{v_j \text{ in } V} P(v_j) \Pi_i P(a_i | v_j)$$

Note: No search in training!

Naïve Bayes Algorithm

Naïve_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 $P(a_i | v_j)$

Classify_New_Instance(*x*)

$$v_{NB} = \text{argmax}_{v_j \text{ in } V} \hat{P}(v_j) \Pi_i \hat{P}(a_i | v_j)$$

Naïve Bayes: Example

- Consider *PlayTennis* again, and new instance

<*Outlk* = *sun*, *Temp* = *cool*, *Humid* = *high*, *Wind* = *strong*>

Want to compute:

$$v_{NB} = \text{argmax}_{v_j \text{ in } V} P(v_j) \Pi_i P(a_i | v_j)$$

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

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

- So, $v_{NB} = n$

Naïve Bayes: Subtleties

1. Conditional independence assumption is often violated

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

- ...but it works surprisingly well anyway. Note don't need estimated posteriors $P(v_j | x)$ to be correct; need only that

$$\text{argmax}_{v_j \text{ in } V} P(v_j | a_1, a_2 \ldots a_n) = \text{argmax}_{v_j \text{ in } V} P(v_j) \Pi_i P(a_i | v_j)$$

- Domingos & Pazzani [1996] for analysis
- Naïve Bayes posteriors often unrealistically close to 1 or 0

Naïve Bayes: Subtleties

2. What if none of the training instances with target value v_j have attribute a_i?
   $P(a_i|v_j) = 0$, and... $P(v_j) \Pi_i P(a_i|v_j) = 0$

Solution is Bayesian estimate:

$$P(a_i|v_j) = (n_c + mp)/(n + m) \text{ where}$$

- 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 $P(a_i|v_j)$
- m is weight given to prior (i.e., number of "virtual" examples)

Learning to Classify Text

Why?

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

Naïve 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

- $P(+)$ $P(-)$
- $P(doc|+)$ $P(doc|-)$

Naïve Bayes for Text

- Naïve Bayes conditional independence assumption

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

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

One more assumption:

- $P(a_i = w_k|v_j) = P(a_m = w_k|v_j)\ \forall i, m$

"Bag of words" assumption.

Learning Algorithm

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* $P(v_j)$ *and* $P(w_k|v_j)$ *probability terms*

- For each target value v_j in V do
  - $docs_j$ ← subset of *Examples* for which the target value is v_j
  - $P(v_j) \leftarrow |docs_j| / |Examples|$
  - $Text_j$ ← a single document created by concatenating all members of $docs_j$
  - n ← total number of words in $Text_j$ (counting duplicate words multiple times) ("tokens" vs. "tokens")
  - for each word w_k in *Vocabulary*
    - n_k ← number of times word w_k occurs in $Text_j$
    - $P(w_k|v_j) \leftarrow (n_k + 1) / (n + |Vocabulary|)$

Classification Algorithm

CLASSIFY_NAÏVE_BAYES_TEXT (*Doc*)

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB}, where

$$v_{NB} = \text{argmax}_{v_j \text{ in } V} P(v_j) \Pi_{i \text{ in } positions} P(a_i|v_j)$$

Twenty NewsGroups

Given 1000 training documents from each group, learn to classify new documents according to newsgroup:

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

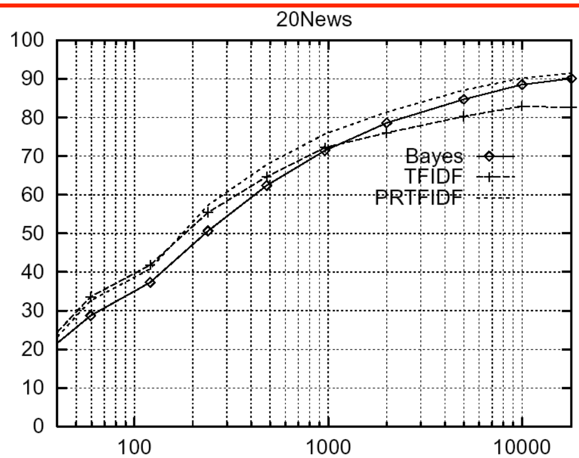
Naïve Bayes: 89% classification accuracy

Article in rec.sport.hockey

- Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu
- From: xxx@yyy.zzz.edu (John Doe)
- Subject: Re: This year's biggest and worst (opinion)
- Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he's clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he's only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided

Learning Curve



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

Bayesian Belief Networks

(Also called Bayes Nets, directed graphical models, BNs, ...). Interesting because:

- Naïve Bayes assumption of conditional independence too restrictive
- But it's intractable without some such assumptions...
- Belief networks describe conditional independence among subsets of variables

→ allows combining prior knowledge about (in)dependencies among variables with observed training data

Conditional Independence

Definition: X is *conditionally independent* of Y given Z if the probability distribution governing X is independent of the value of Y given the value of Z; that is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

more compactly, we write

$$P(X \mid Y, Z) = P(X \mid Z)$$

Independence Example

Example: *Thunder* is conditionally independent of *Storm*, given *Lightning*

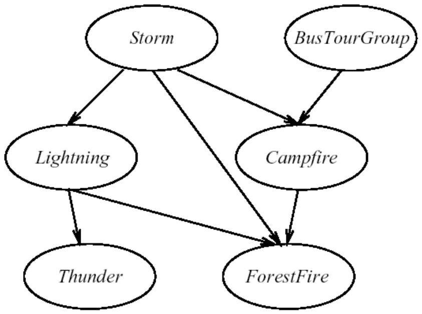
$P(Thunder \mid Storm, Lightning)$

$= P(Thunder \mid Lightning)$

Naïve Bayes uses conditional ind. to justify

$$P(X, Y \mid Z) = P(X \mid Y, Z)\, P(Y \mid Z) = P(X \mid Z)\, P(Y \mid Z)$$

Bayesian Belief Network

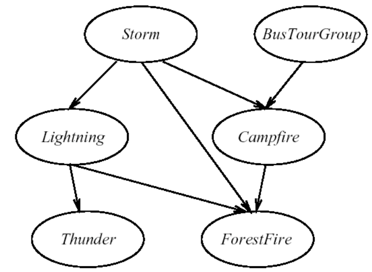


	S,B	*S,¬B*	*¬S,B*	*¬S,¬B*
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

Network represents a set of conditional ind. assertions:

- Each node is asserted to be conditionally ind. of its nondescendants, given its immediate predecessors.
- Directed acyclic graph

Bayesian Belief Network



	S,B	*S,¬B*	*¬S,B*	*¬S,¬B*
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

Represents joint probability distribution over all variables

- e.g., P(*Storm*, *BusTourGroup*, …, *ForestFire*)
- in general, $P(y_1, \ldots, y_n) = \Pi_{i=1}^{n} P(y_i | Parents(Y_i))$
  where $Parents(Y_i)$ denotes immediate predecessors of Y_i in graph
- so, joint distribution is fully defined by graph, plus the $P(y_i | Parents(Y_i))$ (CPTs)

Inference in Bayesian Nets

How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

- Bayes net contains all information needed for this inference
- If only one variable with unknown value, easy to infer it
- Easy if BN is a “polytree”
- In general case, problem is NP hard (#P-complete, Roth 1996).

Inference in Practice

In practice, can succeed in many cases

- Exact inference methods work well for some network structures (small “induced width”)
- Variational methods good approximation if nodes tightly coupled
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions

Now used as a primitive in more advanced learning and reasoning scenarios.

Learning of Bayesian Nets

Several variants of this learning task

- Network structure might be *known* or *unknown*
- Training examples might provide values of *all* network variables, or just *some*

observe \ **structure**	known	unknown
all	like Naïve Bayes	
some		

Learning Bayes Nets

Suppose structure known, variables partially observable

e.g., observe *ForestFire*, *Storm*, *BusTourGroup*, *Thunder*, but not *Lightning*, *Campfire*...

Similar to training neural network with hidden units

- In fact, can learn network conditional probability tables using gradient ascent!
- Converge to network h that (locally) maximizes $P(D|h)$

Gradient Ascent for BNs

Let w_{ijk} denote one entry in the conditional probability table for variable Y_i in the network

$w_{ijk} = P(Y_i = y_{ij} | Parents(Y_i) = u_{ik}$ values)

e.g., if $Y_i = Campfire$, then u_{ik} might be $<Storm = T, BusTourGroup = F>$

Perform gradient ascent by repeatedly:

1. Update all w_{ijk} using training data D

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \text{ in } D} P_h(y_{ij}, u_{ik} | d) / w_{ijk}$$

2. Then, renormalize the w_{ijk} to assure

$$\sum_j w_{ijk} = 1, \ 0 \leq w_{ijk} \leq 1$$

More on Learning BNs

EM algorithm can also be used. Repeatedly:

1. Calculate probabilities of unobserved variables, assuming h
2. Calculate new w_{ijk} to maximize

$$E [\ln P(D|h)]$$

where D now includes both observed and (calculated probabilities of) unobserved variables

Unknown Structure

When structure unknown...

- Algorithms use greedy search to add/subtract edges and nodes
- Active research topic

Somewhat like decision trees: searching for a discrete graph structure

Summary: Belief Networks

- Combine prior knowledge with observed data
- Impact of prior knowledge (when correct!) is to lower the sample complexity
- Active research area (UAI)
  - Extend from Boolean to real-valued variables
  - Parameterized distributions instead of tables
  - Extend to first-order systems
  - More effective inference methods
  - ...

Expectation Maximization

When to use EM:

- Data is only partially observable
- Unsupervised clustering (target value unobservable)
- Supervised learning (some instance attributes or labels unobservable)

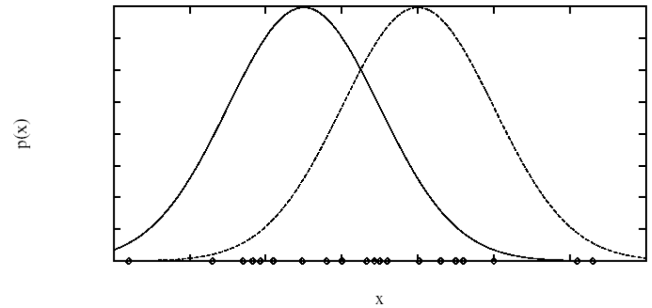
Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (AUTOCLASS)
- Learning Hidden Markov Models

Mixtures of k Gaussians

Each instance x generated by

1. Choosing one of the k Gaussians with uniform probability
2. Generating an instance at random according to that Gaussian



EM for Estimating k Means

Given:

- Instances from X generated by mixture of k Gaussian distributions

Not given:

- Means $<\mu_1, \ldots, \mu_k>$ of the k Gaussians
- Which instance x_i was generated by which Gaussian

Determine:

- ML estimates of $<\mu_1, \ldots, \mu_k>$

Missing Information

Think of full description of each instance as $y_i = <x_i, z_{i1}, z_{i2}>$, where

- z_{ij} is 1 if x_i generated by j th Gaussian (not observed)
- x_i observed data value

If we had full instances, how estimate μ_1, μ_2?

If we had μ_1, μ_2, how predict z_{ij}?

EM for Estimating k Means

EM Algorithm: Pick random initial $h = <\mu_1, \mu_2>$, then iterate

- E step: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = <\mu_1, \mu_2>$ holds.
- M step: Calculate a new maximum likelihood hypothesis $h' = <\mu_1', \mu_2'>$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated above. Replace $h = <\mu_1, \mu_2>$ by $h' = <\mu_1', \mu_2'>$.

E Step For k Means

$E[z_{ij}] = p(x=x_i|\mu=\mu_j) / \Sigma_{n=1}^{2} p(x=x_i|\mu=\mu_n)$

$p(x=x_i|\mu=\mu_j) = \exp(-1/(2\sigma^2)(x_i - \mu_j)^2)$

Derived via PDF for Gaussians and Bayes rule

M Step For *k* Means

$\mu_j' = (\Sigma_{n=1}^{m} E[z_{ij}]\ x_i) \ / \ (\Sigma_{n=1}^{m} E[z_{ij}])$

Estimates the mean by the sample mean (average of the observed data points, weighted by their probability of being generated by the Gaussian in question).

EM Algorithm

Converges to local maximum likelihood h and provides estimates of hidden variables z_{ij}

In fact, local maximum in $E[\ln P(Y \mid h)]$

- Y is complete (observable plus unobservable variables) data
- Expected value is taken over possible values of unobserved variables in Y

General EM Problem

Given:

- Observed data $X = \{x_1, \ldots, x_m\}$
- Unobserved data $Z = \{z_1, \ldots, z_m\}$
- Parameterized probability dist. $P(Y \mid h)$, where
  - $Y = \{y_1, \ldots, y_m\}$ is the full data $y_i = x_i \cup z_i$
  - h are the parameters

Determine:

- h that (locally) maximizes $E[\ln P(Y \mid h)]$

General EM Method

Define likelihood function $Q(h', h)$, which calculates $Y = X \cup Z$ using observed X and current parameters h to estimate Z

$$Q(h', h) \leftarrow E[\ln P(Y \mid h') \mid h, X]$$

Abstract Algorithm

EM Algorithm:

Estimation (E) step: Calculate $Q(h', h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

$$Q(h', h) \leftarrow E[\ln P(Y \mid h') \mid h, X]$$

Maximization (M) step: Replace hypothesis h by the hypothesis h' that maximizes this Q function.

$$h \leftarrow \mathrm{argmax}_{h'} Q(h', h)$$