

Naive Bayes Classifier

- Naive Bayes Classifier
- Performance evaluation

1

Bayesian recipe for classification

- We could design an optimal classifier if we knew:
 - $P(\omega_i)$ (priors)
 - $p(x | \omega_i)$ (class-conditional densities)
- We have some knowledge and training data $\{(x_i, \omega_i)\}$
- Use the samples to estimate the unknown probability distributions
- X is typically high-dimensional
- Need to estimate $P(X | \omega)$ from *limited* data
- Use Bayesian networks to model $P(X, \omega)$?

Naive Bayes Classifier

- Along with decision trees, neural networks, nearest neighbor, one of the most practical learning methods.
- Categories $\{\omega_1, \omega_2, \dots, \omega_J\}$,
- Feature/attribute vector $\mathbf{x} = [x_1, x_2, \dots, x_d]^t$
- Naive Bayes assumption:

$$P(x_1, \dots, x_d | \omega_j) = \prod_i P(x_i | \omega_j)$$

- Naive Bayes classifier:

$$\omega_{NB} = \arg \max_{\omega_j} P(\omega_j) \prod_i P(x_i | \omega_j)$$

- Performs optimally under certain assumptions

Naive Bayes Classifier

- Given training data set D
- Need to estimate probabilistic parameters, no need for complicated training process as in neural networks
- Estimate $P(\omega_j) = n_j/n$ (Maximum Likelihood estimation)
- Estimate $P(x_i=k | \omega_j)$
 - ML Estimation N_{jik}/n_j – discrete feature

Training Examples

4

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

5

- Consider PlayTennis problem, and new instance
<Outlk=sun, Temp=cool, Humid=high, Wind=strong>

We estimate parameters

$$P(\text{yes}) = 9/14, \quad P(\text{no}) = 5/14$$

$$P(\text{Wind=strong}|\text{yes}) = 3/9$$

$$P(\text{Wind=strong}|\text{no}) = 3/5$$

...

We have

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

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

- Therefore this new instance is classified to "no"

Estimation of Probabilities from Small Samples

- Estimate $P(x_i=k | \omega_j)$ – discrete feature
 - ML Estimation N_{jik}/n_j
 - Poor estimates when n_j is small
 - What if none of the training instances with category ω_j have feature value $x_i=k$?
 $P(x_i=k | \omega_j) = 0$, which lead to $P(\omega_j | \dots, x_i=k, \dots) = 0$
 - Typical solution is Bayesian estimate

$$P(X_i = k | D_j) = \frac{N_{jik} + Mp_k}{n_j + M}$$

Bayesian estimates - Recall

- Bayesian estimates for estimating $\theta_i = P(X=i)$ from data set D

$$P(X = i | D) = \frac{N_i + Mp_i}{\sum_j N_j + M} \equiv \hat{\theta}_{iBE}$$

- *Laplace estimates*: $M=|Dm(X)|$, $Mp_i=1$
- M : imaginary equivalent sample size
- p_i : prior belief about θ_i , summation to 1
- The larger the equivalent sample size M , the more confident we are in our prior

- Estimate $P(x_i | \omega_j)$ – continuous feature
 - Assume e.g. Gaussian distribution $N(\mu, \sigma^2)$, then estimate μ , σ , or
 - Discretize into $\{1, \dots, k\}$
 - Equal-width interval: $Width = (x_{max} - x_{min})/k$
 - Convert x to i if x is in i th interval

Naive Bayes Classifier

- Conditional independence assumption is often violated
- But it works surprisingly well anyway (Domingos and Pazzani, 1997)
- Successful applications:
 - Diagnosis
 - Learn which news articles are of interest.
 - Learn to classify web pages by topic.
 - Learn to assign proteins to functional families
- Performance often comparable to that of neural networks, decision tree, etc.

Learning to Classify Text

- E.g., Learn which news articles are of interest
- Target concept *Interesting?* : Documents $\rightarrow \{+, -\}$
- Learning: Use training examples to estimate
 $P(+)$, $P(-)$, $P(doc|+)$, $P(doc|-)$
- What attributes shall we use to represent text documents?

Text Representation

- Represent each document by vector of words
 - one attribute per word position in document
- $$P(doc|\omega_j) = P(length(doc)|\omega_j) \prod_{i=1}^{length(doc)} P(X_i = w_k | \omega_j)$$
- Given a vocabulary $V: (w_1, \dots, w_{|V|})$
 - We need a probability for each word occurrence in each position in the document: $length \times |V|$
 - Too many probabilities to estimate!
 - Limited samples

Multinomial Model

- Assume that probability of encountering a specific word in a particular position is independent of the position, $P(w_k | \omega_j)$
 - The number of probabilities to be estimated drops to $2 \times |V|$
- Treat each document as a bag of words!
- Each document d results from $|d|$ draws on a multinomial variable X with domain V
- The number of times a word occurs in a document is captured

Multinomial Model

- Assume that the lengths of documents are independent of class

$$P(d | \omega_j) = P(|d|) \frac{(\sum_k N_{dk})!}{\prod_k N_{dk}!} \prod_{k=1}^{|V|} P(w_k | \omega_j)^{N_{dk}}$$

N_{dk} is the # of occurrences of w_k in document d

$$P(w_k | \omega_j) = \hat{\theta}_{jk} = \frac{N_{jk} + c_k}{\sum_k N_{jk} + c}$$

N_{jk} is the # of occurrences of w_k in documents in class j

Learn_naive_Bayes_text(*Examples*)

1. collect all words and other tokens that occur in *Examples*
 - *Vocabulary*: all distinct words and other tokens in *Examples*
2. calculate the required probability terms
 - For each target value ω_j do
 - $docs_j$: subset of *Examples* for which the target value is ω_j
 - $P(\omega_j) = |docs_j|/|Examples|$
 - $Text_j$: a single document created by concatenating all members of $docs_j$
 - n total number of words in $Text_j$ (counting duplicate words multiple times)
 - for each word w_k in *Vocabulary*
 - n_k : number of times word w_k occurs in $Text_j$
 - $P(w_k|\omega_j) = (n_k + 1)/(n + |Vocabulary|)$

Classify_naive_Bayes_text(*Doc*)

1. *positions*: all word positions in *Doc* that contain tokens found in *Vocabulary*
2. Return ω_{NB} , where

$$\omega_{NB} = \arg \max_{\omega_j} P(\omega_j) \prod_{i \text{ in } positions} P(x_i | \omega_j)$$

Naive Bayes Classifier

- Twenty NewsGroups
- Given 1000 training documents from each group. Learn to classify new documents according to which newsgroup it came from

comp.graphics

comp.os.ms-windows.misc

comp.sys.ibm.pc.hardware

comp.sys.mac.hardware

comp.windows.x

alt.atheism

soc.religion.christian

talk.politics.mideast

talk.politics.misc

talk.politics.guns

misc.forsale

rec.autos

rec.motorcycles

rec.sport.baseball

rec.sport.hockey

talk.religion.misc

sci.space sci.crypt sci.electronics

sci.med

Naive Bayes Classifier

- Use 2/3 documents as training examples
- Performance was measured over the remaining third
- Naive Bayes: 89% classification accuracy
 - 100 most frequent words were removed from Vocabulary ("the", "of")
 - Any word occurring fewer than 3 times was removed

Evaluating the performance of a classifier

- How well can a classifier be expected to perform on *novel* data?
- Performance on the training data is not a good indicator of performance on future data
- An empirical approach for estimating the generalization error is to use test data
- Test set: independent instances that have not been used in any way to create the classifier
- We can *estimate* the performance (e.g., accuracy) of the classifier using a test data set
- Assumption: both training data and test data are representative samples of the underlying problem

Evaluating the performance of a classifier

- *Hold out method*: Randomly split the set of training samples into two parts
 - Training set (e.g. 2/3 of data) for setting model parameters in the classifier
 - *Test set* (e.g. 1/3 of data) used to estimate the generalization error
- Ideally both training set and test set should be large
- What to do if the amount of data is limited?

- *m*-fold cross-validation:
 - The training set is randomly divided into *m* disjoint sets of equal size n/m
 - The classifier is trained *m* times, each time with a different set held out as a test set.
 - The estimated performance is the mean of these *m* errors
- *Leave-one-out approach*: $m=n$
- Such techniques can be applied to virtually every classification method

Evaluating the performance of a classifier

- How close is the estimated performance to the true performance?
 - Assume the estimated accuracy is 75%. How close is this to the true accuracy?
 - Depends on the amount of test data
- Report confidence Interval:
 - $p=.75$ with 95% confidence the true value in $[.73, .77]$
- There is extensive literature on how to estimate the performance of a hypothesis from limited samples and how to assign confidence to estimates – take Stat 430x