# Naive Bayes Classifier



## Bayesian recipe for classification

• The Bayesian recipe is simple, optimal, and in principle, straightforward to apply

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

2

# Naive Bayes Classifier

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

$$P(x_1, \cdots, x_d \mid \omega_j) = \prod_i P(x_i \mid \omega_j)$$

- Naive Bayes classifier:

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

- Performs optimally under certain assumptions

3

# 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 Likelyhood estimation)
- Estimate $P(x_i=a_{ik} \mid \omega_j)$
  - 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 |

5

## Example

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

We estimate parameters

P(yes) = 9/14, P(no) = 5/14
P(Wind=strong|yes) = 3/9
P(Wind=strong|no) = 3/5
…

We have

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

- Therefore this new instance is classified to “no”

6

## Estimation of Probabilities from Small Samples

- Estimate $P(x_i=a_{ik}|\ \omega_j)$ – discrete feature
  - ML Estimation $N_{jik}/n_j$
  - Poor estimates when $n_j$ is small
  - What if none of the training instances with category $\omega_j$ have feature value $x_i=a_{ik}$?

    $P(x_i=a_{ik}\ |\ \omega_j) = 0$, which lead to $P(\omega_j|\ldots, x_i=a_{ik},\ldots)=0$
  - Typical solution is Bayesian estimate

7

- Bayesian estimates for estimating $\theta_k$=P($X_i$=k) from data set Dj

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

- *Laplace estimates:* $M=|Dm(X_i)|$, $Mp_k=1$
- $M$: imaginary equivalent sample size
- $p_k$: prior belief about $\theta_k$, summation to 1
- The larger the equivalent sample size $M$, the more confident we are in our prior

- Estimate density $p(x_i | \omega_j)$ – continuous feature
  - Assume e.g. Gaussian distribution $N(\mu,\sigma^2)$, then estimate $\mu$, $\sigma$, or
  - Discretize into $\{1, ..., k\}$
    - 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.

10

# Learning to Classify Text

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

11

# Text Representation

- Represent each document by vector of words
  - one attribute per word position in document

$$P(doc|\omega_j) = P(length(doc)|\omega_j)\prod_{i=1}^{|doc|} P(X_i = \mathrm{w_k} \,|\omega_j)$$

- We need a probability for each word occurrence in each position in the document: *2 x length x |vocabulary|*
  - Too many probabilities to estimate!
  - Limited samples

12

# Binary Independence Model

- Given a vocabulary V: ($w_1$, …, $w_{|V|}$)
- A document is a vector of binary features ($X_1$, …, $X_{|V|}$)
- $X_i$ is 1 if $w_i$ appears in the document, 0 otherwise

$$P(doc|\omega_j) = \prod_{i=1}^{|V|} P(x_i \mid \omega_j) = \prod_{i=1}^{|V|} \theta_{ji}^{x_i}(1-\theta_{ji})^{1-x_i}$$

$$\hat{\theta}_{ji} = \frac{N_{ji}+c_1}{N_j+c} \quad N_{ji} :\text{\# of documents in class } j \text{ with word } w_i$$

- *Multi-variate Bernoulli Model*
- The number of times a word occurs in a document is not captured

13

# 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 x |vocabulary|*
- Treat each document as a bag of words!
- Each document *d* results from |d| draws on a multinomial variable *X* with |V| values
- The number of times a word occurs in a document is captured

14

# Multinomial Model

- Assume that the lengths of documents are independent of class

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

$\mathrm{N_k}$ is the # of occurences of $\mathrm{w_k}$ in document $d$

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

$\mathrm{N_{jk}}$ is the # of occurences of $\mathrm{w_k}$ in documents in class $j$

15

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 $\omega_j$ do
     - $docs_j$ : subset of *Examples* for which the target value is $\omega_j$
     - $P(\omega_j) = |docs_j|/|Examples|$
     - $Text_j$ : 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 $Text_j$
       - $P(w_k|\omega_j) = (n_k + 1)/(n + |Vocabulary|)$

16

Classify_naive_Bayes_text(*Doc*)

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

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

17

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

# 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