CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Machine Learning

Machine Learning

Lecture: Bayesian Classification

Ted Scully

Naïve Bayes Classifier Example (Weather Dataset)

In order to see the probability estimates in action we will look at a simple dataset called the weather dataset. We will look at the process by which it creates a model and then classifies unseen instances of data such as the following.

Outlook =sunny, Temp = cool, Humidity = high, Windy = true: **Play = ?**

Anyone for Tennis?

ID	Outlook	Temp	Humidity	Windy	Play?
A	sunny	hot	high	false	no
B	sunny	hot	high	true	no
C	overcast	hot	high	false	yes
D	rainy	mild	high	false	yes
E	rainy	cool	normal	false	yes
F	rainy	cool	normal	true	no
G	overcast	cool	normal	true	yes
H	sunny	mild	high	false	no
I	sunny	cool	normal	false	yes
J	rainy	mild	normal	false	yes
K	sunny	mild	normal	true	yes
L	overcast	mild	high	true	yes
M	overcast	hot	normal	false	yes
N	rainy	mild	high	true	no

Anyone for Tennis?

ID	Outlook	Temp	Humidity	Windy	Play?
A	sunny	hot	high	false	no
B	sunny	hot	high	true	no
C	overcast	hot	high	false	yes
D	rainy	mild	high	false	yes
E	rainy	cool	normal	false	yes
F	rainy	cool	normal	true	no
G	overcast	cool	normal	true	yes
H	sunny	mild	high	false	no
I	sunny	cool	normal	false	yes
J	rainy	mild	normal	false	yes
K	sunny	mild	normal	true	yes
L	overcast	mild	high	true	yes
M	overcast	hot	normal	false	yes
N	rainy	mild	high	true	no

ML Algorithm

Conditional Probabilities

1. P(Outlook = Sunny | Play = Y)
2. P(Outlook = Sunny | Play = N)
3. P(Outlook = Overcast | Play = Y)
4. P(Outlook = Overcast | Play = N)
5. …..

Our Naïve Bayes algorithm takes as input the data set and produces the following **model.**

```
P(Outlook=s | Play=y) = 2/9  P(Outlook=s | Play=n) = 3/5
P(Outlook=o | Play=y) = 4/9  P(Outlook=o | Play=n) = 0/5
P(Outlook=r  | Play=y) = 3/9 P(Outlook=r  | Play=n) = 2/5
```

```
P(Wind=t | Play=y) = 3/9   P(Wind=t | Play=n) = 3/5
P(Wind=f | Play=y) = 6/9   P(Wind=f | Play=n) = 2/5
```

```
P(Temp=h | Play=y) = 2/9  P(Temp=h  | Play=n) = 2/5
P(Temp=m | Play=y) = 4/9  P(Temp=m | Play=n) = 2/5
P(Temp=c  | Play=y) = 3/9 P(Temp=c  | Play=n) = 1/5
```

P(Humidity =high| Play = yes) = 3/9
P(Humidity =normal| Play = yes) = 6/9

P(Humidity =high| Play = no) = 4/5
P(Humidity =normal| Play = no) = 1/5

```
Play=y) = 9/14
Play=n) = 5/14
```

Classify a New Instance

Outlook =sunny, Temp = cool, Humidity = high, Windy = true: **Play = ?**

Classify a New Instance

Outlook =<u>sunny</u>, Temp = <u>cool</u>, Humidity = <u>high</u>, Windy = <u>true</u>: **Play = ?**

Play is y or n. Evaluate probability of each given data.

P(Play = y | Outlook = s, Temp =c, Humidity = h, Wind = t) =
P(Play = y) * P(Outlook = s | Play = y) * P(Temp=c | Play = y) * P(Humidity= h | Play = y) * P(Wind = t | Play = y)
= 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = **0.005291**

P(Play = n | Outlook = s, Temp =c, Humidity = h, Wind = t) =
P(Play = n) * P(Outlook = s | Play = n) * P(Temp=c | Play = n) * P(Humidity= h | Play = n) * P(Wind = t | Play = n)
= 5/14 * 3/5 * 1/5 * 4/5 *3/5 = **0.020571**

$$P(c_j)\prod_{x \in X} P(x \mid c)$$

Consider the following data instance:

Outlook =<u>overcast</u>, Temp = mild, Humidity = normal, Windy = false: Play = ?

```
P(Outlook=s | Play=y) = 2/9  P(Outlook=s | Play=n) = 3/5
P(Outlook=o | Play=y) = 4/9  P(Outlook=o | Play=n) = 0/5
P(Outlook=r  | Play=y) = 3/9 P(Outlook=r  | Play=n) = 2/5
```

Problem with Using Frequencies for Probability Calculations

- So far we estimated probabilities using the following:

- $P(X = x_1 | C = c_1) = \frac{N_{x1c1}}{N_{c1}}$
  - N_{x1c1} = counts of cases where $X=x_1$ and $C=c_1$
  - N_{c1} = count of cases where $C=c_1$

- To avoid the problem of zero probabilities we can applying basic smoothing techniques to the above formula.

Avoiding Zeros

- To avoid the problem outlined on the previous slide we typically use +1 or laplace smoothing.

- Often some basic softening of the equation is performed. For example (**+1 smoothing**), $(N_{x1c1} + \mathbf{1}) / (N_{c1} + \mathbf{2})$

- ***Laplace*** **Smoothing** (m-estimate) : $(N_{x1c1} + \mathbf{1}) / (N_{c1} + |\mathbf{X}|)$

  - N_{x1c1} = counts of cases where X=x1 and C=c1
  - N_{c1} = count of cases where C=c1
  - |X| = count of cases of X (number of features(attributes))

Avoiding Zeros

- Remember we worked out P(Outlook = o | Play = n) = 0/5

- +1 smoothing $(N_{x1c1} + 1) / (N_{c1} + 2)$
- If we use +1 smoothing **P(Outlook = o | Play = n) would be (0+1)/(5+2) = 1/7**

- *Laplace* Smoothing (m-estimate) : $(N_{x1c1} + 1) / (N_{c1} + |X|)$
- **P(Outlook = o | Play = n) would be (0+1)/(5+4) = 1/9**
  - Remember |X| is the number of features

Problems with Probabilities for Naïve Bayes

$$P(c_j)\prod_{x \in X} P(x \mid c)$$

Can you see any computational problem that may occur from this formula? Hint: What might happen if you have a large amount of features?

The computation issue is that of underflow: doing too many multiplications of small numbers.

When we go to calculate the product p(w0|ci)p(w1|ci)p(w2|ci)...p(wN|ci) and many of these numbers are very small, we'll get underflow (multiply many small numbers in a programming language and eventually it rounds off to 0.)

Using Log

- The most common solution to the problem on the previous slide is to calculate the logarithm of this product.
- Doing this allows us to avoid the underflow or round-off error problem. Why? Because we end up adding the individual probabilities rather than multiplying them
- In other word we now get the log of the Bayes equation

$$\mathbf{log}(P(c)\prod_{x\in X} P(x|c))$$

  - We now use

$$\mathbf{log}\,P(c) + \sum_{x\in X} \mathbf{log}\,P(x \mid c)$$

Contents

1. Probability distributions, rules and Bayes theorem

2. Classification Example using Naïve Bayes

3. <u>Text Classification Using Naïve Bayes</u>

Document Classification

- Naive Bayes is a very successful and effective approach to learning to classify text documents.
- In document classification **each word is treated as an feature**.
- Document Classification
  - Spam Filtration
  - Author Identification
  - Sentiment Analysis (movie review, product reviews, important applications)

Document Classification

- A Bayesian classifier will typically either adopt a **bag** of words or **set** of words approach.
  - (<u>Bernoulli model</u>) **Set of words**, counts the number of documents where a word occurs
  - (<u>Multinomial Model</u>) **Bag of words**, counts the total occurrences of a word across all documents.

- When classifying a test document, the Bernoulli model uses **binary occurrence** information, ignoring the number of occurrences of a word in a document , whereas the multinomial model keeps track of multiple occurrences in a single document.
- The models also differ in how **<u>non-occurring terms</u>** are used in classification. They do not impact the classification decision in the multinomial model; but in the Bernoulli model the probability of non-occurrence is factored in when computing probabilities

Calculating Prior Probabilities

$$c_{MAP} = argmax_{c \in C} \log P(c) + \sum_{w \in W} \log P(w \mid c)$$

- The first thing we need to do is calculate the prior probabilities (that is, the probability of the class). This calculation is the same for both multinomial and binomial.

$$P(c) = \frac{\text{Number of documents of class c}}{Total\ Number\ of\ documents}$$

Naïve Bayes - Multinomial Model

$$c_{MAP} = argmax_{c \in C} \log P(c) + \sum_{w \in W} \log P(w \mid c)$$

- Calculation of the probabilities in the multinomial model as are follows (notice we use laplace smoothing here):

- $P(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$

count(w, c) is the number of occurrences of the word w in all documents of class c.

count(c) The total number of words in all documents of class c (**including duplicates**).

|V| The number of words in the vocabulary, which is all unique words irrespective of class.

Exercise

- The table below shows a very simple training set containing 4 documents and the words contained within those documents.
- It also contains the class of each of the document.

- Objective is to classify the new Test as either class Comp or class Politics.
  - We will use **laplace** for calculating the Multinomial probabilities
  - We will use simple **+1 smoothing** for calculating the Bernoulli probabilities

	Doc	Words	Class
Training	1	Cloud Java Cloud	Comp
	2	Cloud Cloud Spring	Comp
	3	Cloud Software	Comp
	4	Referendum Software Election	Politics
Test	5	Java Software Java Election	?

	Doc	Words	Class
Training	1	Cloud Java Cloud	Comp
	2	Cloud Cloud Spring	Comp
	3	Cloud Software	Comp
	4	Referendum Software Election	Politics
Test	5	Java Software Java Election	?

$$c_{MAP} = argmax_{c \in C} \log P(c) + \sum_{w \in W} \log P(w \mid c)$$

$$P(Comp) = \frac{3}{4}$$

$$P(Politics) = \frac{1}{4}$$

	Doc	Words	Class
Training	1	Cloud Java Cloud	Comp
	2	Cloud Cloud Spring	Comp
	3	Cloud Software	Comp
	4	Referendum Software Election	Politics
Test	5	Java Software Java Election	?

$$c_{MAP} = argmax_{c \in C} \log P(c) + \sum_{w \in W} \log P(w \mid c)$$

	Doc	Words	Class
Training	1	Cloud Java Cloud	Comp
	2	Cloud Cloud Spring	Comp
	3	Cloud Software Java	Comp
	4	Referendum Software Election	Politics
Test	5	Java Software Java Election	?

$$P(w \mid c) = \frac{count(w,c) + 1}{count(c) + |V|}$$

Notice we use Laplace smoothing here

	Doc	Words	Class
Training	1	Cloud Java Cloud	Comp
	2	Cloud Cloud Spring	Comp
	3	Cloud Software Java	Comp
	4	Referendum Software Election	Politics
Test	5	Java Software Java Election	?

$$P(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

Notice we use Laplace smoothing here

$$P(Cloud \mid Comp) = \frac{5+1}{9+6}$$

$$P(Java \mid Comp) = \frac{2+1}{9+6}$$

$$P(Software \mid Comp) = \frac{1+1}{9+6}$$

$$P(Spring \mid Comp) = \frac{1+1}{9+6}$$

$$P(Referendum \mid Comp) = \frac{0+1}{9+6}$$

$$P(Election \mid Comp) = \frac{0+1}{9+6}$$

	Doc	Words	Class
Training	1	Cloud Java Cloud	Comp
	2	Cloud Cloud Spring	Comp
	3	Cloud Software Java	Comp
	4	Referendum Software Election	Politics
Test	5	Java Software Java Election	?

$$P(w \mid c) = \frac{count(w,c) + 1}{count(c) + |V|}$$

Notice we use Laplace smoothing here

$$P(Cloud \mid Politics) = \frac{0+1}{3+6}$$

$$P(Java \mid Politics) = \frac{0+1}{3+6}$$

$$P(Software \mid Politics) = \frac{1+1}{3+6}$$

$$P(Spring \mid Politics) = \frac{0+1}{3+6}$$

$$P(Referendum \mid Politics) = \frac{1+1}{3+6}$$

$$P(Election \mid Politics) = \frac{1+1}{3+6}$$

	Doc	Words	Class
Test	5	Java Software Java Election	?

$$P(Cloud \mid Comp) = \frac{6}{15}$$

$$P(Java \mid Comp) = \frac{3}{15}$$

$$P(Software \mid Comp) = \frac{2}{15}$$

$$P(Spring \mid Comp) = \frac{2}{15}$$

$$P(Election \mid Comp) = \frac{1}{15}$$

$$P(Referendum \mid Comp) = \frac{1}{15}$$

$$P(Cloud \mid Politics) = \frac{1}{9}$$

$$P(Java \mid Politics) = \frac{1}{9}$$

$$P(Software \mid Politics) = \frac{2}{9}$$

$$P(Spring \mid Politics) = \frac{1}{9}$$

$$P(Election \mid Politics) = \frac{2}{9}$$

$$P(Referendum \mid Politics) = \frac{2}{9}$$

$$P(Comp) = \frac{3}{4} \qquad P(Politics) = \frac{1}{4}$$

	Doc	Words	Class
Test	5	Java Software Java Election	?

$$c_{MAP} = argmax_{c \in C} \log P(c) + \sum_{w \in W} \log P(w \mid c)$$

	Doc	Words	Class
Test	5	Java Software Java Election	?

$$P(c \mid W) = \log P(c) + \sum_{w \in W} \log P(w \mid c)$$

$P(Comp \mid Test) =$ log(3/4) + log(3/15) + log(2/15) + log (3/15) + log (1/15) = **-3.57**

$P(Politics \mid Test) =$ log(1/4) + log(1/9) + log(2/9)+log(1/9)+log(2/9)= **-3.81**

Classify the document as being of class Comp

Naïve Bayes: Text Classification for Multinomial

Examples are a set of training documents.

V is the set of classes (ex. Spam / NotSpam)

Learn_naive_Bayes_text($Examples$, V)

1. collect all words that occur in $Examples$

   $Vocabulary \leftarrow$ all distinct words 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

   - $docs_j \leftarrow$ 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 $Text_j$
     - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Document Classification

- Classify_naive_Bayes_text(newDoc)
  - We take in an unseen document *newDoc*, we extract all words from the document and store in *allWords* (the same word may appear multiple time)
  - Return V_{NB}, where:

$$V_{NB} = \underset{v_j \in V}{\mathrm{argmax}} \quad logP(v_j) + \sum_{x \in allWords} \log P(x \mid v_j)$$