

# Bayesian Learning

- A powerful approach in machine learning
- Combine data seen so far with prior beliefs
  - This is what has allowed us to do machine learning, have good inductive biases, overcome "No free lunch", and obtain good generalization on novel data
- We use it in our own decision making all the time
  - You hear a word which which could equally be “Thanks” or “Hanks”, which would you go with?
    - Combine sound likelihood and your prior knowledge
  - Texting Suggestions on phone
  - Spell checkers, speech recognition, etc.
  - Many applications

# Bayesian Classification

- $P(c|x)$  - Posterior probability of output class  $c$  given the input vector  $x$
- The discriminative learning algorithms we have learned so far try to “approximate” this directly
- $P(c|x) = P(x|c)P(c)/P(x)$  Bayes Rule – A true probability
- Seems like more work but often calculating the right hand side probabilities can be reasonable and advantageous
- $P(c)$  - Prior probability of class  $c$  – How do we know?
  - Just count up and get the probability for the Training Set – Easy!
- $P(x|c)$  - Probability “likelihood” of data vector  $x$  given that the output class is  $c$ 
  - Later we will discuss ways to calculate this *likelihood*
- $P(x)$  - Prior probability of the data vector  $x$ 
  - This is a normalizing term to get an actual probability. In practice we drop it because it is the same for each class  $c$  (i.e. independent), and we are mostly interested in which class  $c$  maximizes  $P(c|x)$ .

# Bayesian Classification Example

- Assume we have 100 examples in our Training Set with two output classes Good and Bad, and 80 of the examples are of class good. We want to figure out  $P(c|x) \sim P(x|c)P(c)$
- Thus our priors are:

# Bayesian Classification Example

- Assume we have 100 examples in our Training Set with two output classes Good and Bad, and 80 of the examples are of class good.
- Thus our priors are:
  - $P(\text{Good}) = .8$
  - $P(\text{Bad}) = .2$
- $P(c|x) = P(x|c)P(c)/P(x)$  Bayes Rule
- Now we are given an input vector  $x$  which has the following likelihoods
  - $P(x|\text{Good}) = .3$
  - $P(x|\text{Bad}) = .4$
- What should our output be?

# Bayesian Classification Example

- Assume we have 100 examples in our Training Set with two output classes Good and Bad, and 80 of the examples are of class good.
- Thus our priors are:
  - $P(\text{Good}) = .8$
  - $P(\text{Bad}) = .2$
- $P(c|x) = P(x|c)P(c)/P(x)$  Bayes Rule
- Now we are given an input vector  $x$  which has the following likelihoods
  - $P(x|\text{Good}) = .3$
  - $P(x|\text{Bad}) = .4$
- What should our output be?
- Try all possible output classes and see which one maximizes the posterior using Bayes Rule:  $P(c|x) = P(x|c)P(c)/P(x)$ 
  - Drop  $P(x)$  since it is the same for both
  - $P(\text{Good}|x) = P(x|\text{Good})P(\text{Good}) = .3 \cdot .8 = .24$
  - $P(\text{Bad}|x) = P(x|\text{Bad})P(\text{Bad}) = .4 \cdot .2 = .08$

# Bayesian Intuition

- Bayesian vs. Frequentist
- Bayesian allows us to talk about probabilities/beliefs even when there is little data, because we can use the prior
  - What is the probability of a nuclear plant meltdown?
  - What is the probability that BYU will win the national championship?
- As the amount of data increases, Bayes shifts confidence from the prior to the likelihood
- Requires reasonable priors in order to be helpful
- We use priors all the time in our decision making
  - Unknown coin: probability of heads? (over time?)

# Bayesian Learning of ML Models

- Assume  $H$  is the hypothesis space,  $h$  a specific hypothesis from  $H$ , and  $D$  is all the training data
- $P(h|D)$  - Posterior probability of  $h$ , this is what we usually want to know in a learning algorithm – (i.e. model selection)
- $P(h|D) = P(D|h)P(h)/P(D)$  Bayes Rule
- $P(h)$  - Prior probability of the hypothesis/model independent of  $D$  - do we usually know?
  - Could assign equal probabilities
  - *Could assign probability based on an inductive bias* (e.g. simple hypotheses have higher probability) – Thus regularization is in the equation!
- $P(D|h)$  - Probability “likelihood” of data given the hypothesis
  - Just use the accuracy of the model  $h$  on the data  $D$
- $P(D)$  - Prior probability of the data
- $P(h|D)$  increases with  $P(D|h)$  and  $P(h)$ . In learning when seeking to discover the best  $h$  given a particular  $D$ ,  $P(D)$  is the same and can be dropped.

# Bayesian Learning

- Learning (finding) the best model the Bayesian way
- For Machine Learning the likelihood  $P(D|h)$  is usually measured using the accuracy of the hypothesis on the training data
  - If the hypothesis is very accurate on the data, that implies that the data is more likely given that particular hypothesis
- Maximum a posteriori (MAP) hypothesis
- $h_{\text{MAP}} = \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h)P(h)/P(D) = \operatorname{argmax}_{h \in H} P(D|h)P(h)$
- Maximum Likelihood (ML) Hypothesis  $h_{\text{ML}} = \operatorname{argmax}_{h \in H} P(D|h)$
- MAP = ML if all priors  $P(h)$  are equally likely (uniform priors)
- Note that the prior can be like an inductive bias (i.e. simpler hypotheses are more probable)
- For Bayesian learning, overfitting handled in the equation. Note regularization similarities.

# Bayesian Learning (cont)

- Brute force approach is to test each  $h \in H$  to see which maximizes  $P(D|h)P(h)$
- $P(D|h)P(h)$  is called the *Relative Posterior* and is not the real probability without the normalizing  $P(D)$
- Can still get the real probability (if desired) by normalization if there is a limited number of hypotheses
  - Assume only two possible hypotheses  $h_1$  and  $h_2$
  - The true posterior probability of  $h_1$  would be

$$P(h_1 | D) = \frac{P(D|h_1)P(h_1)}{P(D|h_1) + P(D|h_2)}$$

# Example of MAP Hypothesis

- Assume only 3 possible hypotheses in hypothesis space  $H$
- Given a data set  $D$  which  $h$  do we choose?
- Maximum Likelihood (ML):  $\operatorname{argmax}_{h \in H} P(D|h)$
- Maximum a posteriori (MAP):  $\operatorname{argmax}_{h \in H} P(D|h)P(h)$

$H$	Likelihood $P(D h)$	Prior $P(h)$	Relative Posterior $P(D h)P(h)$
$h_1$	.6	.3	.18
$h_2$	.9	.2	.18
$h_3$	.7	.5	.35

# Example of MAP Hypothesis – True Posteriors

- Assume only 3 possible hypotheses in hypothesis space  $H$
- Given a data set  $D$

$H$	Likelihood $P(D h)$	Prior $P(h)$	Relative Posterior $P(D h)P(h)$	True Posterior $P(D h)P(h)/P(D)$
$h_1$	.6	.3	.18	$.18/ (.18 + .18 + .35) = .18/.71 = .25$
$h_2$	.9	.2	.18	$.18/.71 = .25$
$h_3$	.7	.5	.35	$.35/.71 = .50$

# Prior Handles Overfit

- Prior can make it so that less likely hypotheses (those likely to overfit) are less likely to be chosen
- Similar to regularization
- Minimize  $F(h) = Error(h) + \lambda \cdot Complexity(h)$
- $P(h|D) = P(D|h)P(h)$
- The challenge is
  - Deciding on priors – subjective
  - Maximizing across  $H$  which is usually infinite – approximate by searching over "best  $h$ 's" in more efficient time

# Minimum Description Length

- Information theory shows that the number of bits required to encode a message  $i$  is  $-\log_2 p_i$
- Call the minimum number of bits to encode message  $i$  with respect to code  $C$ :  $L_C(i)$

$$\begin{aligned} h_{\text{MAP}} &= \operatorname{argmax}_{h \in H} P(h) P(D|h) = \\ &= \operatorname{argmin}_{h \in H} -\log_2 P(h) - \log_2 P(D|h) = \\ &= \operatorname{argmin}_{h \in H} L_{C_1}(h) + L_{C_2}(D|h) \end{aligned}$$

- $L_{C_1}(h)$  is a representation of hypothesis
- $L_{C_2}(D|h)$  is a representation of the data. Since you already have  $h$  all you need is the data instances which differ from  $h$ , which are the lists of misclassifications
- The  $h$  which minimizes the MDL equation will have a balance of a small representation (simple hypothesis) and a small number of errors

# Bayes Optimal Classifier

- *Best* question is what is the most probable classification  $c \in C$  for a new instance input  $x$ , rather than what is the most probable hypothesis for a data set
- Let *all* possible hypotheses vote for the instance in question, weighted by their posterior (an ensemble approach) - better than the single best MAP hypothesis

$$P(c_j|D, H, x) = \sum_{h_i \in H} P(c_j|x, h_i)P(h_i|D) = \sum_{h_i \in H} \frac{P(c_j|x, h_i)P(D|h_i)P(h_i)}{P(D)}$$

- Bayes Optimal Classification:

$$\begin{aligned} c_{BayesOptimal} &= \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|x, h_i)P(h_i|D) \\ &= \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|x, h_i)P(D|h_i)P(h_i) \end{aligned}$$

- Also known as the *posterior predictive*

# Bayes Optimal Classifier

- *Best* question is what is the most probable classification  $c \in C$  for a new instance input  $x$ , rather than what is the most probable hypothesis for a data set
- Let *all* possible hypotheses vote for the instance in question, weighted by their posterior (an ensemble approach) - better than the single best MAP hypothesis

$$P(c_j|D, H, x) = \sum_{h_i \in H} P(c_j|x, h_i)P(h_i|D) = \sum_{h_i \in H} \frac{P(c_j|x, h_i)P(D|h_i)P(h_i)}{P(D)}$$

- Bayes Optimal Classification:

$$\begin{aligned} c_{BayesOptimal} &= \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|x, h_i)P(h_i|D) \\ &= \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|x, h_i)P(D|h_i)P(h_i) \end{aligned}$$

- Also known as the *posterior predictive*

# Example of Bayes Optimal Classification

$$c_{BayesOptimal} = \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j | x, h_i) P(D | h_i) P(h_i)$$

- Assume same 3 hypotheses with priors and posteriors as shown for a data set  $D$  with 2 possible output classes (A and B)
- Assume novel input instance  $x$  where  $h_1$  and  $h_2$  output B and  $h_3$  outputs A as class – 1/0 output case. Which class wins and what are the probabilities?

$H$	Likelihood $P(D h)$	Prior $P(h)$	Posterior $P(D h)P(h)$	$P(A)$	$P(B)$
$h_1$	.6	.3	.18	$0 \cdot .18 = 0$	$1 \cdot .18 = .18$
$h_2$	.9	.2	.18		
$h_3$	.7	.5	.35		
Sum					

# Example of Bayes Optimal Classification

$$c_{BayesOptimal} = \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j | x, h_i) P(D | h_i) P(h_i)$$

- Assume same 3 hypotheses with priors and posteriors as shown for a data set  $D$  with 2 possible output classes (A and B)
- Assume novel input instance  $x$  where  $h_1$  and  $h_2$  output B and  $h_3$  outputs A as class – 1/0 output case. Which class wins and what are the probabilities?

$H$	Likelihood $P(D h)$	Prior $P(h)$	Posterior $P(D h)P(h)$	$P(A)$	$P(B)$
$h_1$	.6	.3	.18	$0 \cdot .18 = 0$	$1 \cdot .18 = .18$
$h_2$	.9	.2	.18	$0 \cdot .18 = 0$	$1 \cdot .18 = .18$
$h_3$	.7	.5	.35	$1 \cdot .35 = .35$	$0 \cdot .35 = 0$
Sum				.35	.36

# Example of Bayes Optimal Classification

- Assume probabilistic outputs from the hypotheses for new a instance  $x$

$H$	$P(A)$	$P(B)$
$h_1$	.3	.7
$h_2$	.4	.6
$h_3$	.9	.1

$H$	Likelihood $P(D h)$	Prior $P(h)$	Posterior $P(D h)P(h)$	$P(A)$	$P(B)$
$h_1$	.6	.3	.18	$.3 \cdot .18 = .054$	$.7 \cdot .18 = .126$
$h_2$	.9	.2	.18	$.4 \cdot .18 = .072$	$.6 \cdot .18 = .108$
$h_3$	.7	.5	.35	$.9 \cdot .35 = .315$	$.1 \cdot .35 = .035$
Sum				.441	.269

# Bayes Optimal Classifiers (Cont.)

- *No other classification method using the same hypothesis space can outperform a Bayes optimal classifier on average, given the available data and prior probabilities over the hypotheses*
- Large or infinite hypothesis spaces make this impractical in general
- Also, it is only as accurate as our knowledge of the priors (background knowledge) for the hypotheses, which we often do not know
  - But if we do have insights, priors can really help
  - For example, it would automatically handle overfit, with no need for a validation set, early stopping, etc.
  - Note that using accuracy, etc. for likelihood  $P(D|h)$  is also an approximation
- If our priors are bad, then Bayes optimal will not be optimal for the actual problem. For example, if we just assumed uniform priors, then you might have a situation where the many lower posterior hypotheses could dominate the fewer high posterior ones.
- However, Bayes optimal is an important theoretical concept, and it leads to many practical algorithms which are simplifications based on the concepts of full Bayes optimality (e.g. ensembles)

# Naïve Bayes

## Revisit Bayesian Classification

- $P(c|x) = P(x|c)P(c)/P(x)$
- $P(c)$  - Prior probability of class  $c$  – How do we know?
  - Just count up and get the probability for the Training Set – Easy!
- $P(x|c)$  - Probability “likelihood” of data vector  $x$  given that the output class is  $c$ 
  - We use  $P(x_1, \dots, x_n|c_j)$  as short for  $P(x_1 = val_1, \dots, x_n = val_n|c_j)$
  - How do we really do this?
  - If  $x$  is real valued?
  - If  $x$  is nominal we can just look at the training set and count to see the probability of  $x$  given the output class  $c$  but how often will all  $x$ 's values be the same?
    - Which will also be the problem if we bin real valued inputs

# Naïve Bayes Classifier

$$c_{MAP} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j | x_1, \dots, x_n) = \operatorname{argmax}_{c_j \in \mathcal{C}} P(x_1, \dots, x_n | c_j) P(c_j)$$

- Note we are not considering  $h \in H$ , rather just collecting statistics from the data set
- Given a training set,  $P(c_j)$  is easy to calculate
- How about  $P(x_1, \dots, x_n | c_j)$ ? Most cases would be either 0 or 1
- Key "Naïve" leap: Assume conditional independence of the attributes

$$P(x_1, \dots, x_n | c_j) = \prod_i P(x_i | c_j)$$

$$c_{NB} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j) \prod_i P(x_i | c_j)$$

- $P(\text{Thin, Red, Meat} | \text{Good}) = P(\text{Thin} | \text{Good}) * P(\text{Red} | \text{Good}) * P(\text{Meat} | \text{Good})$
- There is usually sufficient data to get accurate values for independent terms

# Naïve Bayes Classifier

- While conditional independence is not typically a reasonable assumption... (heart rate and blood pressure)
  - Low complexity simple approach
  - Need only store all  $P(c_j)$  and  $P(x_i|c_j)$  terms
  - Assume nominal features for the moment
  - Easy to calculate the  $|attribute\ values| \times |classes|$  terms
  - There is usually enough data to make the independent terms reasonably accurate
  - Effective and common for many large applications (Document classification, etc.)

# Naïve Bayes Example

Size (L, S)	Color (R,G,B)	Output (P,N)
L	R	N
S	B	P
S	G	N
L	R	N
L	G	P

For the given training set:

1. Create a table of the statistics needed to do Naïve Bayes
2. What would be the best output for a new instance which is Large and Blue? (e.g. the class which wins the argmax)
3. What is the true probability for each output class (P or N) for Large and Blue?

$$c_{NB} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j) \prod_i P(x_i | c_j)$$

What do we need?

Size (L, S)	Color (R,G,B)	Output (P,N)
L	R	N
S	B	P
S	G	N
L	R	N
L	G	P

$P(P)$	
$P(N)$	
$P(\text{Size=L} P)$	
$P(\text{Size=S} P)$	
$P(\text{Size=L} N)$	
$P(\text{Size=S} N)$	
$P(\text{Color=R} P)$	
$P(\text{Color=G} P)$	
$P(\text{Color=B} P)$	
$P(\text{Color=R} N)$	
$P(\text{Color=G} N)$	
$P(\text{Color=B} N)$	

$P(c_j)$

$P(x_i|c_j)$

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i|c_j)$$

## \*\*Challenge Question\*\*

Finish and give the true Probabilities of P and N for Size = L and Color = B

Size (L, S)	Color (R,G,B)	Output (P,N)
L	R	N
S	B	P
S	G	N
L	R	N
L	G	P

$P(P)$	$2/5$
$P(N)$	$3/5$
$P(\text{Size}=L P)$	$1/2$
$P(\text{Size}=S P)$	$1/2$
$P(\text{Size}=L N)$	$2/3$
$P(\text{Size}=S N)$	
$P(\text{Color}=R P)$	
$P(\text{Color}=G P)$	
$P(\text{Color}=B P)$	
$P(\text{Color}=R N)$	
$P(\text{Color}=G N)$	
$P(\text{Color}=B N)$	

$P(c_j)$

$P(x_i|c_j)$

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i|c_j)$$

$$P(P) = 2/5 * 1/2 * P(\text{Color} = B|P) = ? \quad P(N) = ?$$

## \*\*Challenge Question\*\*

Finish and give the true Probabilities of P and N for Size = L and Color = B

Size (L, S)	Color (R,G,B)	Output (P,N)
L	R	N
S	B	P
S	G	N
L	R	N
L	G	P

$P(P)$	$2/5$
$P(N)$	$3/5$
$P(\text{Size=L} P)$	$1/2$
$P(\text{Size=S} P)$	$1/2$
$P(\text{Size=L} N)$	$2/3$
$P(\text{Size=S} N)$	$1/3$
$P(\text{Color=R} P)$	$0/2$
$P(\text{Color=G} P)$	$1/2$
$P(\text{Color=B} P)$	$1/2$
$P(\text{Color=R} N)$	$2/3$
$P(\text{Color=G} N)$	$1/3$
$P(\text{Color=B} N)$	$0/3$

$P(c_j)$

$P(x_i|c_j)$

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i|c_j)$$

$$P(P) = 2/5 * 1/2 * 1/2 = 1/10$$

$$P(N) = 3/5 * 2/3 * 0/3 = 0$$

True Probabilities

$$P(P) = .1 / (.1 + 0) = 1$$

$$P(N) = 0 / (.1 + 0) = 0$$

# Naïve Bayes Homework

Size (L, S)	Color (R,G,B)	Output (P,N)
L	R	P
S	B	P
S	B	N
L	R	N
L	B	P
L	G	N
S	B	P

For the given training set:

1. Create a table of the statistics needed to do Naïve Bayes
2. What would be the best output for a new instance which is Small and Blue? (e.g. the class which wins the argmax)
3. What is the true probability for each output class (P or N) for Small and Blue?

$$c_{NB} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j) \prod_i P(x_i | c_j)$$

# Naïve Bayes (cont.)

- Can normalize to get the actual naïve Bayes probability
- Continuous data? - Can discretize a continuous feature into bins, thus changing it into a nominal feature and then gather statistics normally
  - How many bins? - More bins is good, but need sufficient data to make statistically significant bins. Thus, base it on data available
  - Could also assume data is Gaussian and compute the mean and variance for each feature given the output class, then each  $P(x_i|c_j)$  becomes  $\mathcal{N}(x_i|\mu_{xi|c_j}, \sigma^2_{xi|c_j})$ 
    - Not good if data is multi-modal

# Infrequent Data Combinations

- Would if there are 0 or very few cases of a particular  $x_i=v|c_j$  ( $n_v/n$ )? ( $n_v$  is the number of instances with output  $c_j$  where  $x_i =$  attribute value  $v$  and  $n$  is the total number of instances with output  $c_j$ )
- Should usually allow every case at least some finite probability since it could occur in the test set, else the 0 terms will dominate the product
- Could replace  $n_v/n$  with the Laplacian:  $(n_v+1)/(n+|V|)$ 
  - Where  $|V|$  is the number of possible attribute values
- Thus if  $n_v/n$  is 0/10 and  $x_i$  has three attribute values, the Laplacian would be 1/13.

## Naïve Bayes (cont.)

- No training - Just gather the statistics from the data set and then apply the Naïve Bayes classification equation to any new instance
- Easier to have many attributes since not building a model (eg MLP), and the amount of statistics gathered grows linearly with the number of attribute values ( $\# \text{ attribute values} \times \# \text{ classes}$ ) - Thus natural for applications like text classification which can be represented with huge numbers of input attributes.
- Though Naïve Bayes is limited by first order assumptions, it is still often used and gives reasonable results in many large real-world applications

# Text Classification Example

- A text classification approach
  - Want  $P(\textit{class}/\textit{document})$  – How to represent document?

# Text Classification Example

- A text classification approach
  - Want  $P(\text{class}/\text{document})$  - Use a "Bag of Words" approach – order independence assumption (valid?)
    - Variable length input of query document is fine
  - Calculate bag of words for every word/token in the language and each output class based on the training data. Words that occur in testing but do not occur in the training data are ignored.
  - Good empirical results. Can drop filler words (the, and, etc.) and words found less than  $z$  times in the training set.

# Text Classification Example

- A text classification approach
    - Want  $P(\text{class}/\text{document})$  - Use a "Bag of Words" approach – order independence assumption (valid?)
      - Variable length input of query document is fine
    - Calculate bag of words for every word/token in the language and each output class based on the training data. Words that occur in testing but do not occur in the training data are ignored.
    - Good empirical results. Can drop filler words (the, and, etc.) and words found less than  $z$  times in the training set.
    - $P(\text{class}/\text{document}) \approx P(\text{class}/\text{BagOfWords})$  //assume word order independence  
=  $P(\text{BagOfWords}/\text{class}) * P(\text{class})/P(\text{document})$  //Bayes Rule
      - // But *BagOfWords* usually unique
      - //  $P(\text{document})$  same for all classes
      - // Thus Naïve Bayes
- $\approx P(\text{class}) * \prod P(\text{word}/\text{class})$

# Text Classification Example

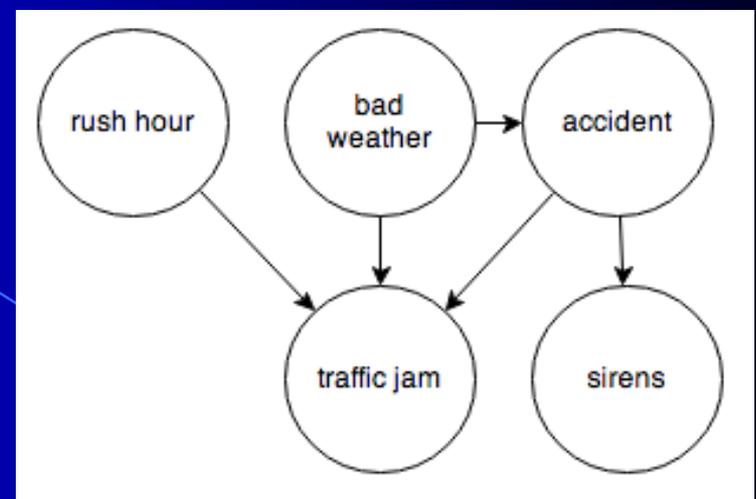
- A text classification approach
    - Want  $P(\text{class}/\text{document})$  - Use a "Bag of Words" approach – order independence assumption (valid?)
      - Variable length input of query document is fine
    - Calculate bag of words for every word/token in the language and each output class based on the training data. Words that occur in testing but do not occur in the training data are ignored.
    - Good empirical results. Can drop filler words (the, and, etc.) and words found less than  $z$  times in the training set.
    - $P(\text{class}/\text{document}) \approx P(\text{class}/\text{BagOfWords})$  //assume word order independence  
=  $P(\text{BagOfWords}/\text{class}) * P(\text{class})/P(\text{document})$  //Bayes Rule
      - // But *BagOfWords* usually unique
      - //  $P(\text{document})$  same for all classes
      - // Thus Naïve Bayes
- $\approx P(\text{class}) * \prod P(\text{word}/\text{class})$

# Less Naïve Bayes

- NB uses just 1st order features - assumes conditional independence
  - calculate statistics for all  $P(x_i|c_j)$
  - $|attributes| \times |attribute\ values| \times |output\ classes|$
- $n$ th order -  $P(x_1, \dots, x_n|c_j)$  - assumes full conditional dependence
  - $|attributes|^n \times |attribute\ values| \times |output\ classes|$
  - Too computationally expensive - exponential
  - Not enough data to get reasonable statistics - most cases occur 0 or 1 time
- 2nd order? - compromise -  $P(x_i x_k|c_j)$  - assume only low order dependencies
  - $|attributes|^2 \times |attribute\ values| \times |output\ classes|$
  - More likely to have cases where number of  $x_i x_k|c_j$  occurrences are 0 or few, could just use the higher order features which occur often in the data
  - 3<sup>rd</sup> order, etc.
- How might you test if a problem is conditionally independent?
  - Could just compare against 2<sup>nd</sup> or higher order. How far off on average is our assumption

$$P(x_i x_k|c_j) = P(x_i|c_j) P(x_k|c_j)?$$

# Bayesian Belief Nets



- Can explicitly specify where there is significant conditional dependence - intermediate ground (all dependencies would be too complex and not all are truly dependent). If you can get both of these correct (or close) then it can be a powerful representation. Important research area - CS 677
- Specify causality in a DAG and give conditional probabilities from immediate parents (causal)
  - Still can work even if causal links are not that accurate, but more difficult to get accurate conditional probabilities
- Belief networks represent the full joint probability function for a set of random variables in a compact space - Product of recursively derived conditional probabilities
- If given a subset of observable variables, then you can infer probabilities on the unobserved variables - general approach is NP-complete - approximation methods are used
- Gradient descent learning approaches for conditionals. Greedy approaches to find network structure.