Machine Learning

Prof. Dr. Stefan Kramer
Johannes Gutenberg-Universität Mainz

Acknowledgements

- Eibe Frank
- Ian Witten
- Tom Mitchell

Evaluation and Validation in Predictive Mining

Outline

- ROC and recall-precision curves
- Loss functions and error measures
- Bayesian learning and Naive Bayes
- Application of Naive Bayes to text categorization

ROC Curves

- “ROC” stands for “receiver operating characteristic”
- Used in signal detection to show tradeoff between hit rate and false alarm rate over a noisy channel
- x axis shows percentage of false positives in sample (rather than sample size)
- y axis shows percentage of true positives in sample (rather than absolute number)

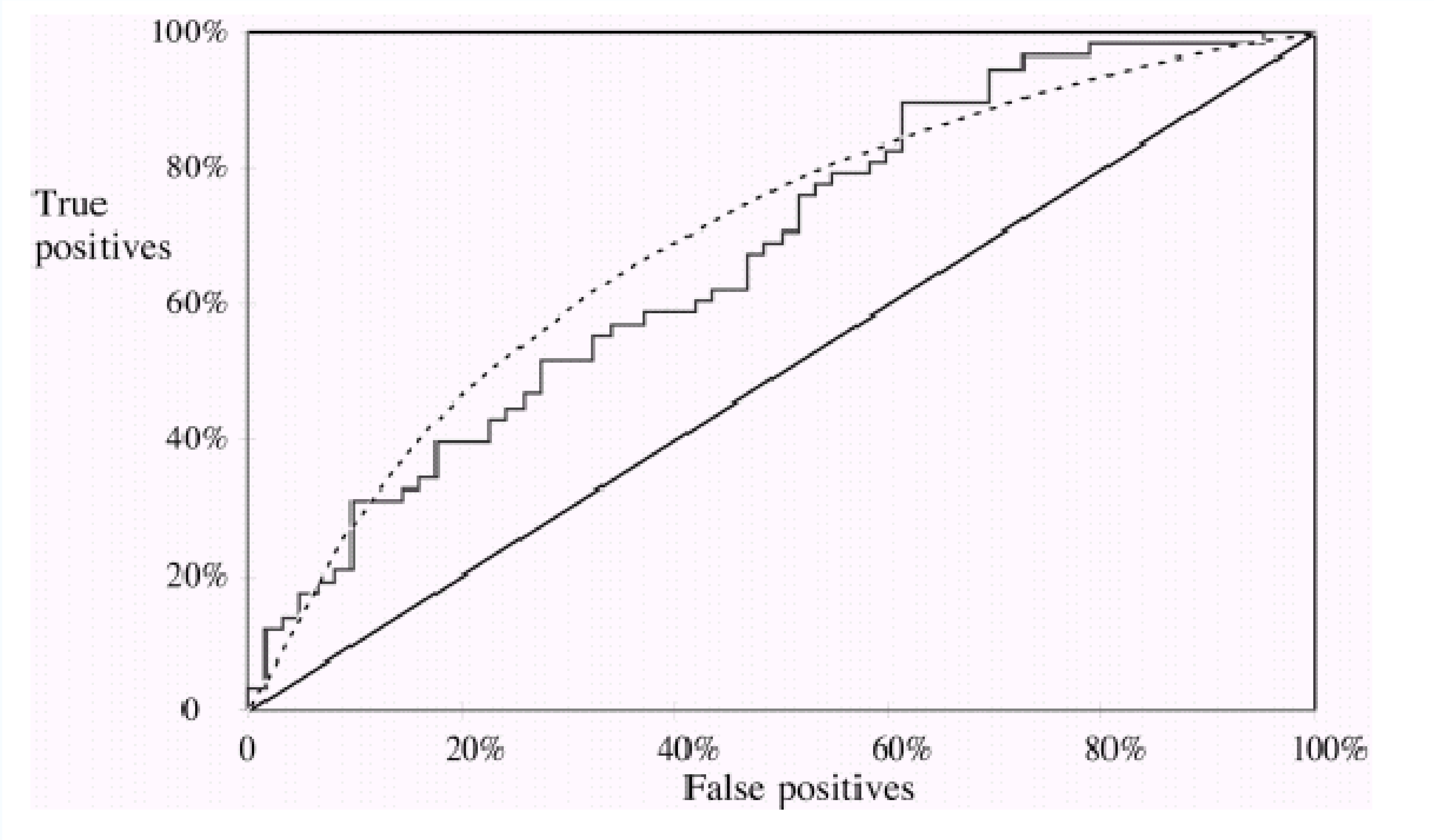
Measures and Curves

	Predicted Class			
Actual Class		Pos	Neg	
	Pos	TP	FN	P
	Neg	FP	TN	N
		PP	PN	

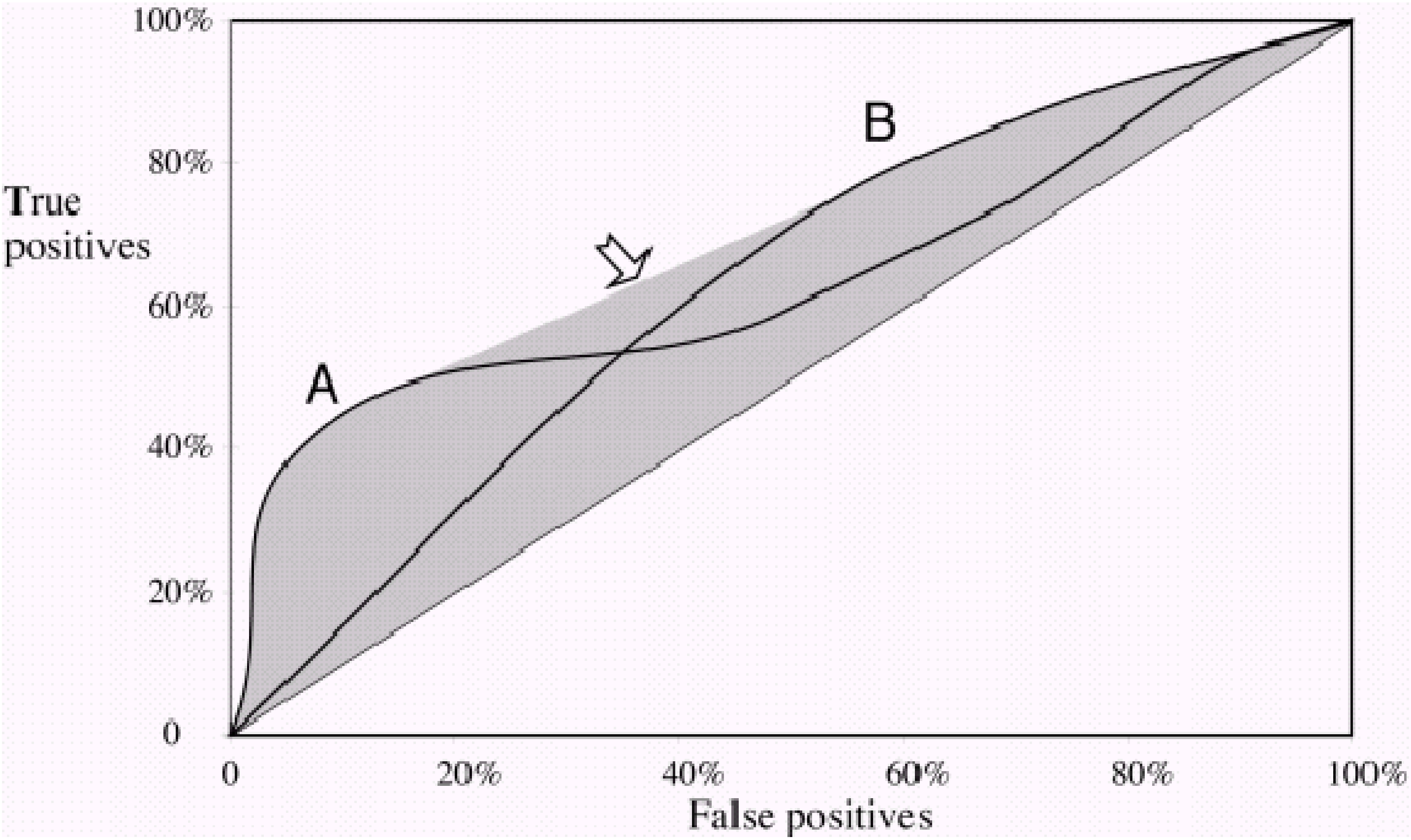
<u>ROC Curves:</u>
x-axis: False Positive Rate = FP /(FP + TN) = FP / N
y-axis: True Positive Rate = TP /(TP + FN) = TP / P

<u>Recall-Precision Curves:</u>
x-axis: Recall = TP / (TP + FN) = TP / P = TPR
y-axis: Precision = TP / (TP + FP) = TP / PP

A Sample ROC Curve



ROC Curves for Two Schemes



Area Under ROC (AUC or AUROC)

- Sometimes, the area under the ROC curve is taken as a measure of quality
- AUROC can also be represented as the ratio of the number of correct pairwise rankings vs. the number of all possible pairs, a quantity known as called the *Wilcoxon-Mann-Whitney (WMW)* statistic

Cross-Validation and ROC

- Simple method of getting a ROC curve using cross-validation:
  - collect probabilities for instances in test folds
  - sort instances according to probability of being positive
- This is the method implemented, for instance, in the WEKA workbench

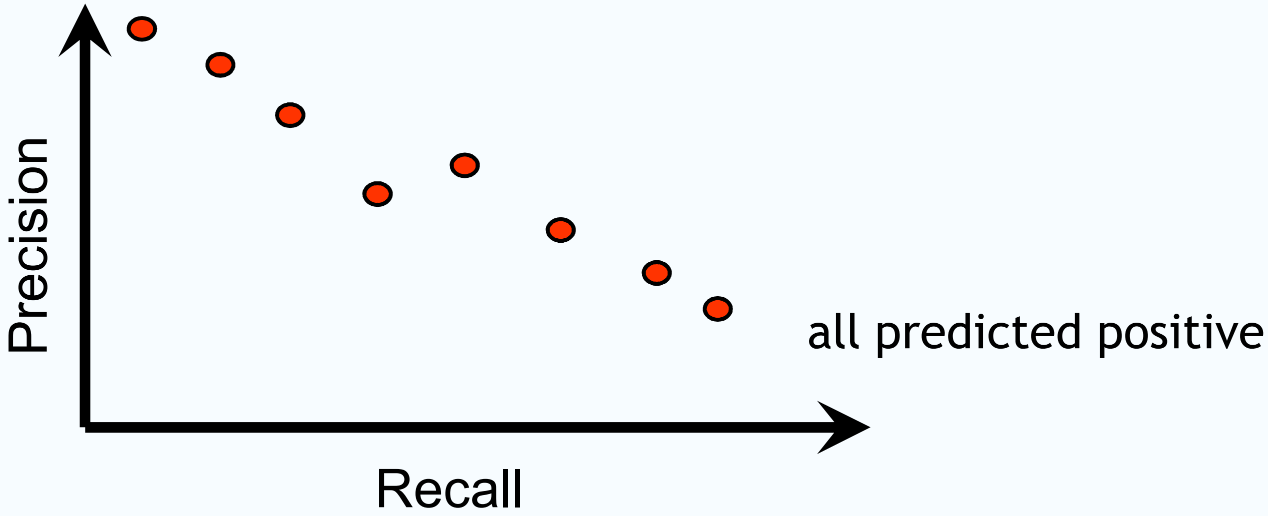
Measures and Curves

	Predicted Class			
Actual Class		Pos	Neg	
	Pos	TP	FN	P
	Neg	FP	TN	N
		PP	PN	

ROC Curves:
x-axis: False Positive Rate = FP /(FP + TN) = FP / N
y-axis: True Positive Rate = TP /(TP + FN) = TP / P

Recall-Precision Curves:
x-axis: Recall = TP / (TP + FN) = TP / P = TPR
y-axis: Precision = TP / (TP + FP) = TP / PP

Recall-Precision Curve/Space



- x-axis is y-axis from ROC curve (TPR)
- Summary measures: average precision at 20%, 50% and 80% recall (“three-point average recall”)
- F-measure=(2xRxP)/(R+P)

Summary of Measures and Curves

Lift Charts:

- x-axis: subset size (TP+FP)/(TP+FP+TN+FN)
- y-axis: TP

ROC Curves:

- x-axis: *False Positive Rate* FPR = FP /(FP + TN) = FP / N
- y-axis: *True Positive Rate* TPR = TP /(TP + FN) = TP / P

Recall-Precision Curves:

- x-axis: *Recall* = TP / (TP + FN) = TP / P = TPR
- y-axis: *Precision* = TP / (TP + FP) = TP / PP

How about the *false negatives* in ROC and recall-precision curves?

Sensitivity = TP / P = Recall = TPR
Specificity = TN / N = (1 – FPR)

Bayesian Learning and Naive Bayes

Bayesian Learning

- Useful in two ways
- As a *theoretical tool* to analyze learning
  - MAP hypothesis
  - Bayes optimal classifier
  - to analyze learning algorithms from a Bayesian point of view:
    - candidate elimination algorithm
    - regression
    - ...
  - to derive schemes for *model selection* like the *minimum description length (MDL)* principle
- To derive *practical machine learning* schemes
  - Naive Bayes, Semi-Naive Bayes, Bayes nets

Bayesian Theorem

likelihood

posterior

$$P(h \mid D) = \frac{P(D \mid h) P(h)}{P(D)}$$

prior(s)

P(h) = prior probability of hypothesis h

P(D) = prior probability of training data D

P(h|D) = conditional probability of h given D

P(D|h) = conditional probability of D given h

Application Bayesian Theorem

- P(cancer) = 0.008
- P(not cancer) = 0.992
- P(pos test|cancer) = 0.98
- P(neg test|cancer) = 0.02
- P(pos test|not cancer) = 0.03
- P(neg test|not cancer) = 0.97

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

- P(cancer|pos test) = P(pos test|cancer)P(cancer)/P(pos test) = 0.00784/P(pos test)
- P(not cancer|pos test) = P(pos test|not cancer)P(not cancer)/P(pos test)= 0.02976/P(pos test)

Remember Basic Rules of Probabilities

- *Product Rule*: probability $P(A \wedge B)$ of a conjunction of two events A and B:
  $P(A \wedge B) = P(A \mid B)\, P(B) = P(B \mid A)\, P(A)$
- *Sum Rule*: probability of a disjunction of two events A and B:
  $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- *Theorem of total probability*: if events $A_1, \ldots, A_n$ are mutually exclusive with $\Sigma P(A_i) = 1$ then $P(B) = \Sigma P(B|A_i)\, P(A_i)$

Brute Force MAP Hypothesis Learner

- For each hypothesis h in H, calculate the posterior probability

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

- Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \arg\max_{h \in H} P(h \mid D)$$

Most Probable Classification of New Instances

- So far we have sought the *most probable hypothesis* given the data D (i.e., h_{MAP})
- Given new instance x, what is its *most probable classification*?
- $h_{MAP}(x)$ is not the most probable classification!
- Consider: three possible hypotheses:
  $P(h_1|D) = 0.4, \quad P(h_2|D) = 0.3, \quad P(h_3|D) = 0.3$
- Given new instance x,
  $h_1(x) = +, \; h_2(x) = -, \; h_3(x) = -$
- What is the *most probable* ***classification*** of x?

Bayes Optimal Classifier

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- *Example*:
  $P(h_1|D) = 0.4$, $P(-|h_1) = 0.0$, $P(+|h_1) = 1.0$
  $P(h_2|D) = 0.3$, $P(-|h_2) = 1.0$, $P(+|h_2) = 0.0$
  $P(h_3|D) = 0.3$, $P(-|h_3) = 1.0$, $P(+|h_3) = 0.0$

- *How does the Bayes optimal classifier decide?*

Model Selection

- *Selecting a model from a given class of models*
- Methods:
  - cross-validation
  - minimum description length (MDL) principle
  - ...

Minimum Description Length (MDL) Principle

- Occam's razor:
  ~ ”prefer the shortest hypothesis”
- MDL: prefer the hypothesis h that minimizes

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} \, L_{C_1}(h) + L_{C_2}(D|h)$$

  where $L_C(x)$ is the description length of x under encoding C
- Measure for a model and its errors *in a common currency*, namely bits

Minimum Description Length (MDL) Principle

- *Example*: H = decision trees, D = training data
- $L_{C1}(h)$ is # bits to *describe* (encode) tree h
- $L_{C2}(D|h)$ is # bits to describe D given h
- Note $L_{C2}(D|h)=0$ if examples classified perfectly by h
- Need only to encode *exceptions, i.e., the errors made by the tree*
- Hence h_{MDL} *trades off tree size for training errors*

Cost Choose

- Coding cost to choose k elements from a set of n

$$L_{choose}(k, n) = n \times Entropy(k, n)$$

$$Entropy(k,n) = -\frac{k}{n}\log\left(\frac{k}{n}\right) - \frac{n-k}{n}\log\left(\frac{n-k}{n}\right)$$

Minimum Description Length (MDL) Principle

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(D|h)P(h) \\ &= \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg\min_{h \in H} - \log_2 P(D|h) - \log_2 P(h) \quad (1) \end{aligned}$$

Interesting fact from information theory: optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits. So interpret (1):

- $-\log_2 P(h)$ is length of h *under optimal code*
- $-\log_2 P(D|h)$ is length of D given h under optimal code
- prefers the hypothesis that minimize length(h) + length(misclassifications)
- In other words: under an optimal encoding with respect to the priors, $h_{MAP} = h_{MDL}$

Naive Bayes Classifier

- Along with nearest neighbor algorithms, one of the fastest baseline learning algorithms
- When to use
  - large training set available
  - attributes that describe instances are conditionally independent given classification
- Successful applications:
  - diagnosis
  - classifying text documents

Naive Bayes Classifier

- Assume target function f: X → V, where each instance x is described by attributes $\{a_1, \ldots, a_n\}$
- Most probable value of f(x) is:

$$
\begin{aligned}
v_{MAP} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n) \\
v_{MAP} &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\
&= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j)
\end{aligned}
$$

- The Naive Bayes assumption of conditional independence

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

gives the *Naive Bayes classifier*:

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Algorithm

Naive_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)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

More on Naive Bayes Algorithm

for each v_k **do**
  $c[v_k] := 0$
  for each a_i **do**
    $c[v_k, a_i] := 0$
for each x **in** D **do**
  $c[x.v] := c[x.v] + 1$
  for each $x.a_i$ **do**
    $c[x.v, a_i] := c[x.v, a_i] + 1$

Naive Bayes Example

- Consider PlayTennis again, and a new instance (Outlook=sunny, Temp=cool, Humid=high, Wind=strong)
- Want to compute:

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

- P(y) P(sunny|y) P(cool|y) P(high|y) P(strong|y) = 0.005
- P(n) P(sunny|n) P(cool|n) P(high|n) P(strong|n) = 0.021

Naive Bayes Subtleties

- Conditional independence assumption is often violated

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

...but it works surprisingly well anyway

- Estimated posteriors of $P(v_j|x)$ need not be correct
- Only

$$\underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) P(a_1 \ldots, a_n | v_j)$$

- See (Domingos and Pazzani, 1996) for analysis
- Naive Bayes posteriors often unrealistically close to 1 or 0

Naive Bayes Subtleties

- What if none of the training instances with target value v_j have attribute value a_i? (*Problem?*)
- Typical solution is Bayesian estimate where

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

- 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 a prior estimate for $P(a_i \mid v_j)$
- m is weight given to prior (i.e., number of "virtual" examples)

Application of Naive Bayes to Text Categorization

Learning to Classify Text

- Why?
  - learn which news articles are of interest
  - learn to classify web pages by topic
- Naive Bayes is a simple and useful baseline algorithm
- What attributes shall we use to represent text documents?

Learning to Classify Text

- Usually: represent each document in so-called bag-of-word (BOW) representation
- Target concept Interesting?: Document $\rightarrow$ {+,-}
- Learning: Use training examples to estimate P(+), P(-), P(doc|+), P(doc|-)
- Basic simplifying assumption: *position of word in document does not matter!*

Naive Bayes for Text Categorization

LEARN_NAIVE_BAYES_TEXT($Examples, V$)

1. collect all words and other tokens that occur in Examples

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

Testing Time

$\text{CLASSIFY\_NAIVE\_BAYES\_TEXT}(Doc)$

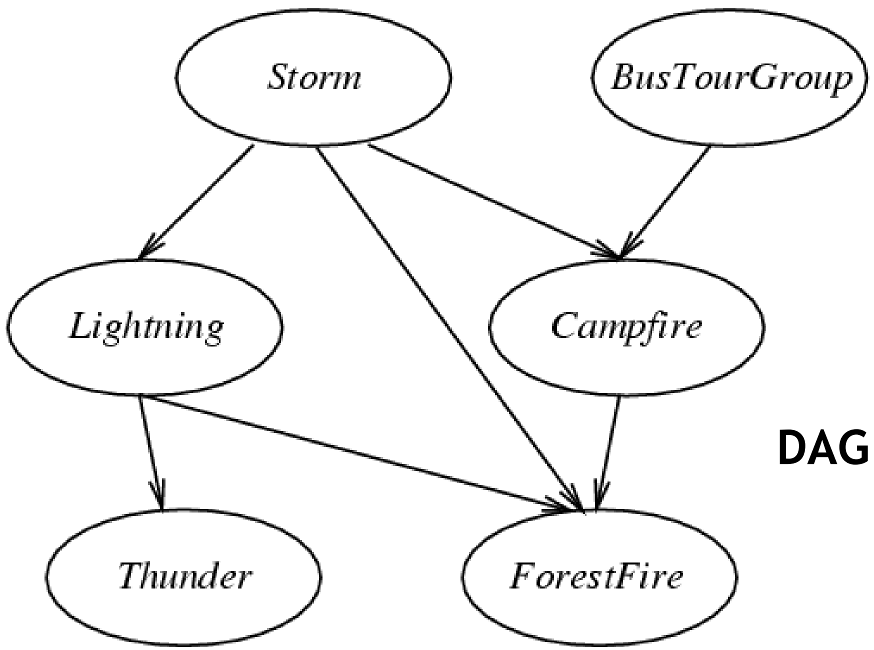
- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return v_{NB}, where

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

Bayesian Networks

Bayesian Networks

- Each node is asserted to be *conditionally independent* of its *non-descendants*, *given* its immediate *predecessors*
- Represents *joint probability distribution* over all variables, e.g., P(Storm, BusTourGroup, ..., ForestFire)



DAG

	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8