Bayesian learning

- incremental, noise-resistant method
- can combine prior Knowledge (the K is probabilistic)
- predictions are probabilistic

Courtesy of Eammon Keogh, UCR,
eamonn@cs.ucr.edu

Naïve Bayes Classifier

Bayes theorem

prior probability

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

posterior probability

Thomas Bayes
1702 - 1761

Let us start with an example of "Bayesian inference":…

Bayes' law of conditional probability:

results in a simple "learning rule": choose
the most likely (Maximum Aposteriori)hypothesis

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

Example:
Two hypo:
(1) the patient has cancer
(2) the patient is healthy

Priors: 0.8% of the population has cancer;

⊕ is 98% reliable: it returns positive in 98% of cases when the the disease is present, and returns 97% negative when the disease is actually absent.

P(cancer) = .008

P(+ |cancer) = .98

P(+|not cancer) = .03

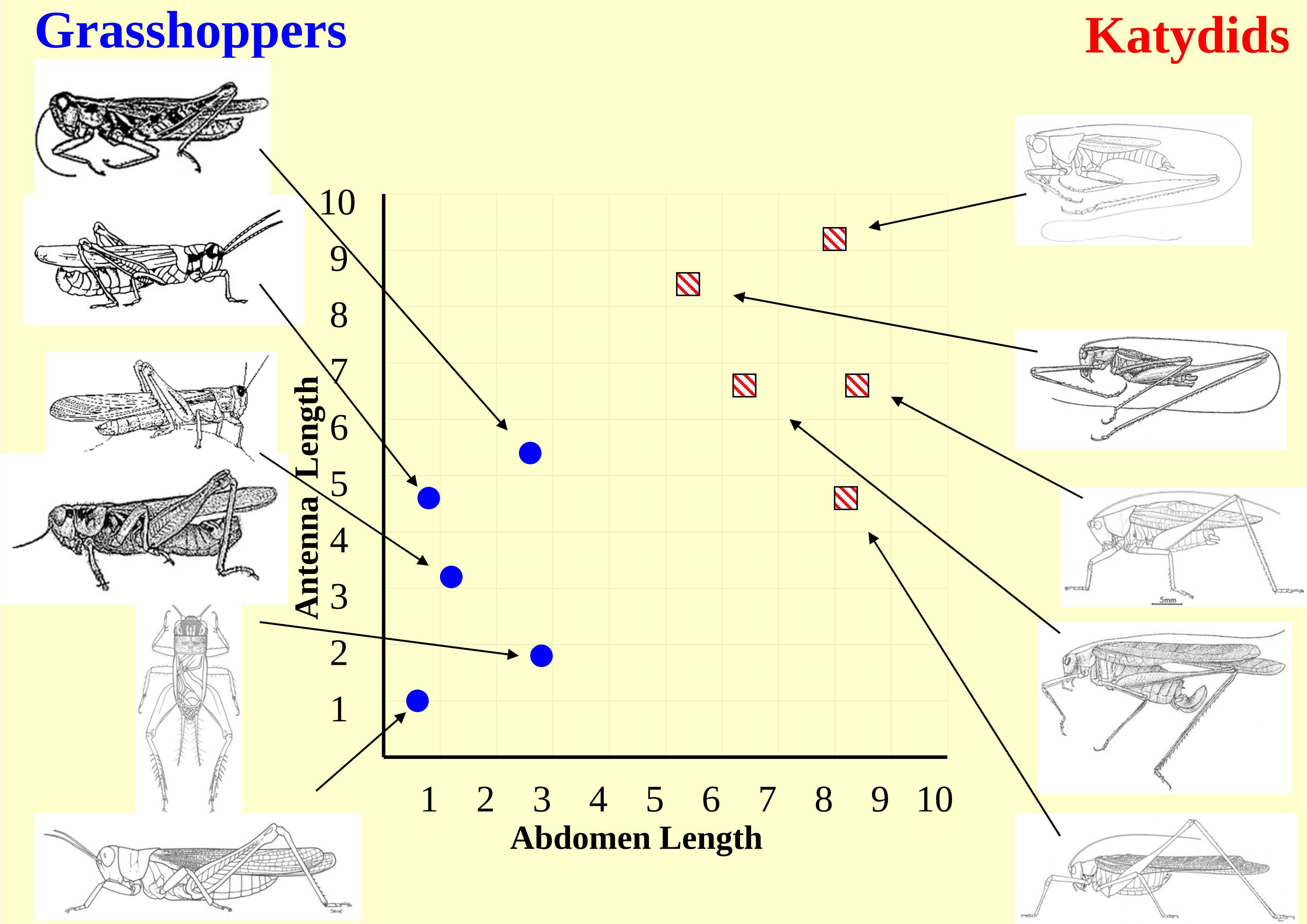
P(not cancer) = .992

P(- |cancer) = .02

P(-|not cancer) = .97

We observe a new patient with a positive test.
How should they be diagnosed?

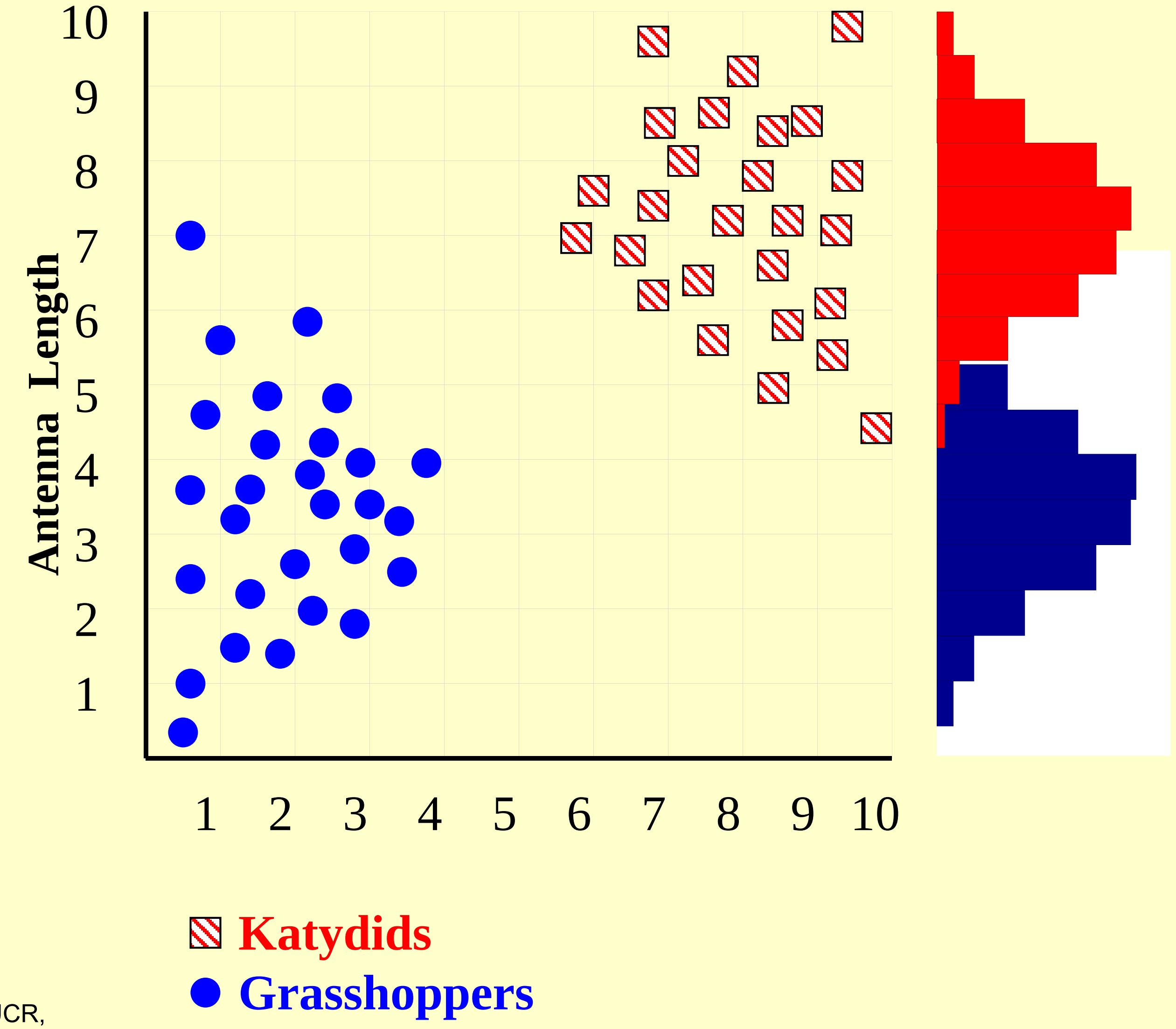
P(+|cancer)P(cancer) = .98*.008 = .0078
P(+|not cancer)P(not cancer) = .03*.992=.0298



Naïve Bayes classifier: the very foundation

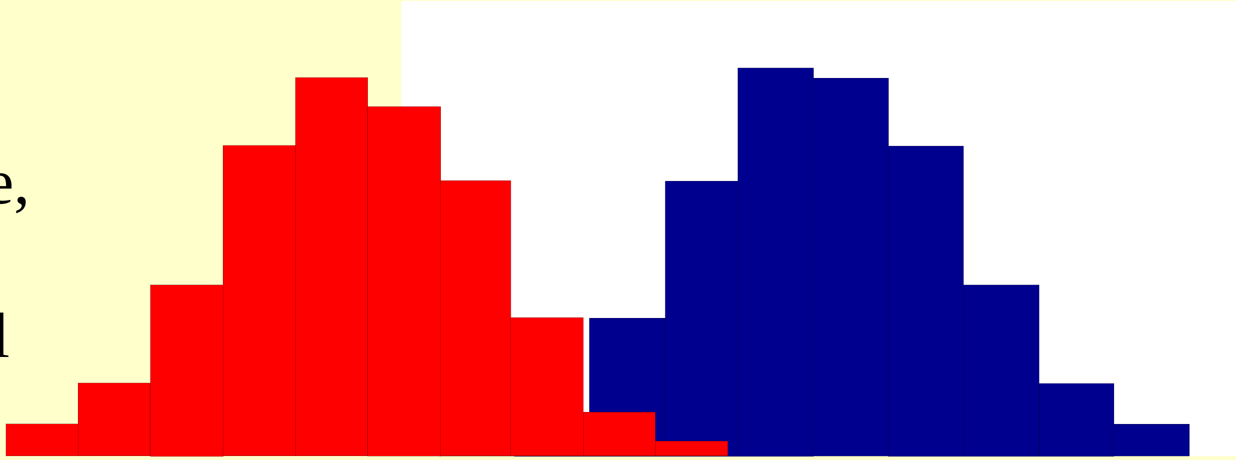
Courtesy of Eammon Keogh, UCR,
eamonn@cs.ucr.edu

With a lot of data, we can build a histogram. Let us just build one for “Antenna Length” for now…

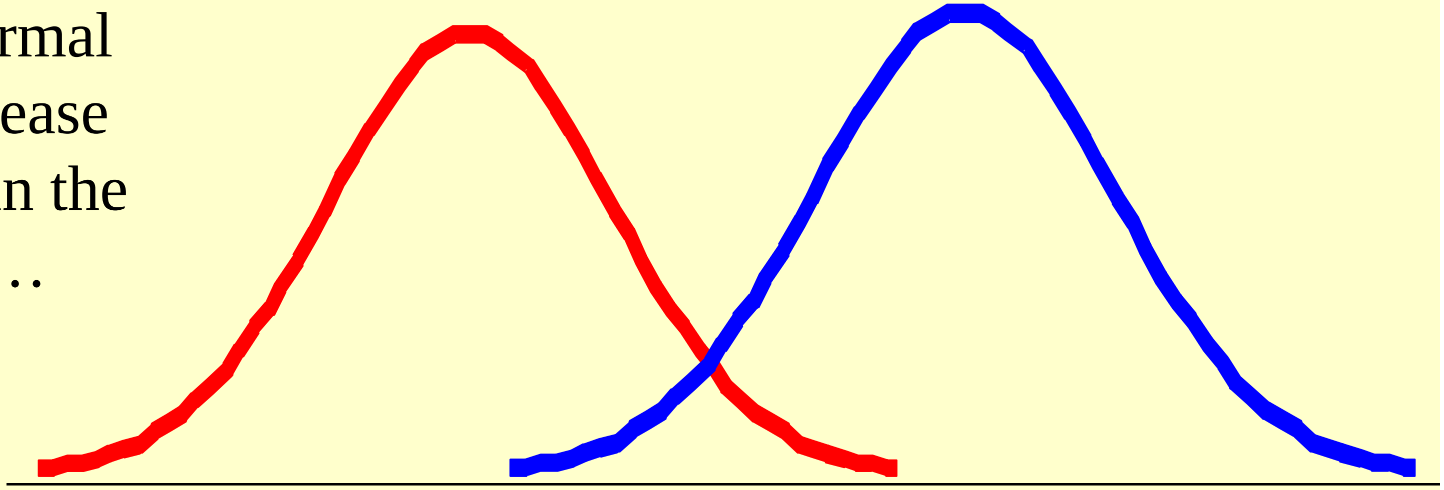


Courtesy of Eammon Keogh, UCR, eamonn@cs.ucr.edu

We can leave the histograms as they are, or we can summarize them with two normal distributions.



Let us us two normal distributions for ease of visualization in the following slides…



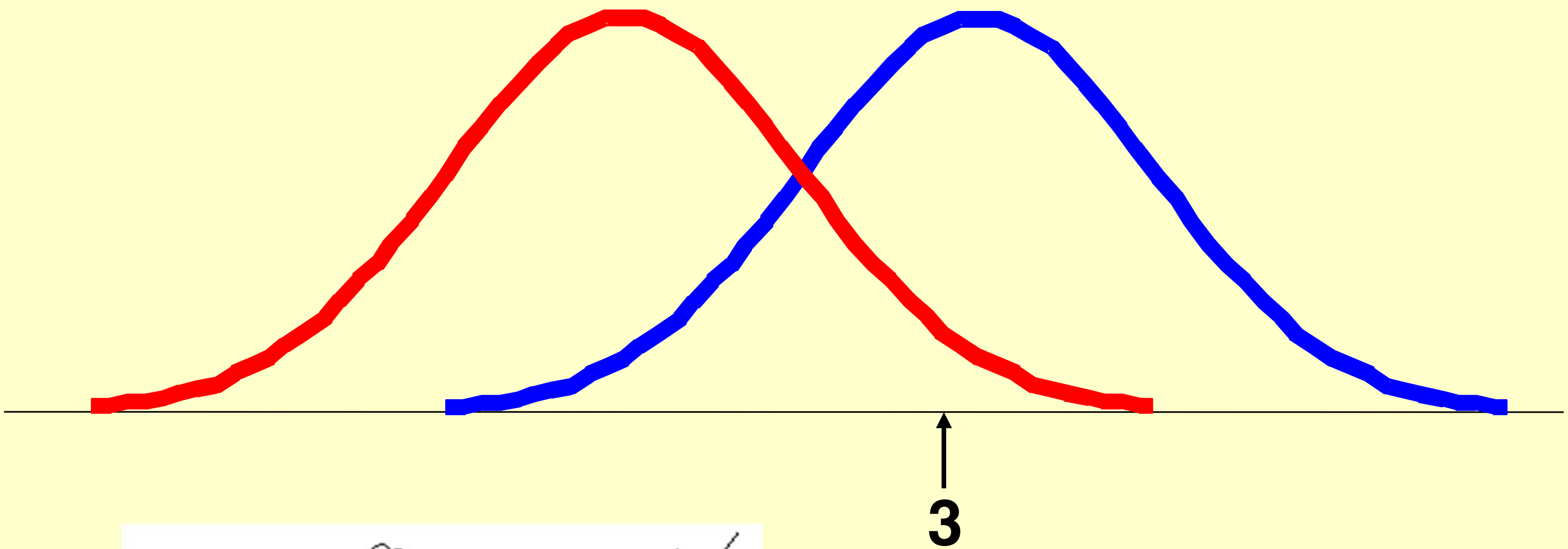
Courtesy of Eammon Keogh, UCR, eamonn@cs.ucr.edu

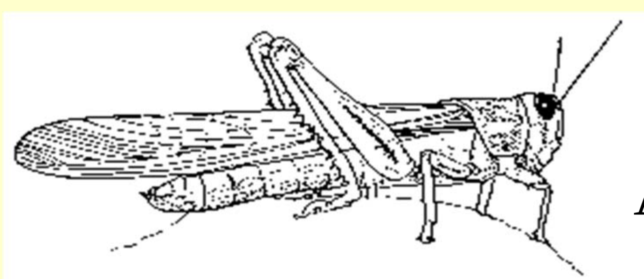
• We want to classify an insect we have found. Its antennae are 3 units long. How can we classify it?

• We can just ask ourselves, give the distributions of antennae lengths we have seen, is it more *probable* that our insect is a **Grasshopper** or a **Katydid**.

• There is a formal way to discuss the most *probable* classification…

$p(c_j \mid d)$ = probability of class c_j, *given* that we have observed d





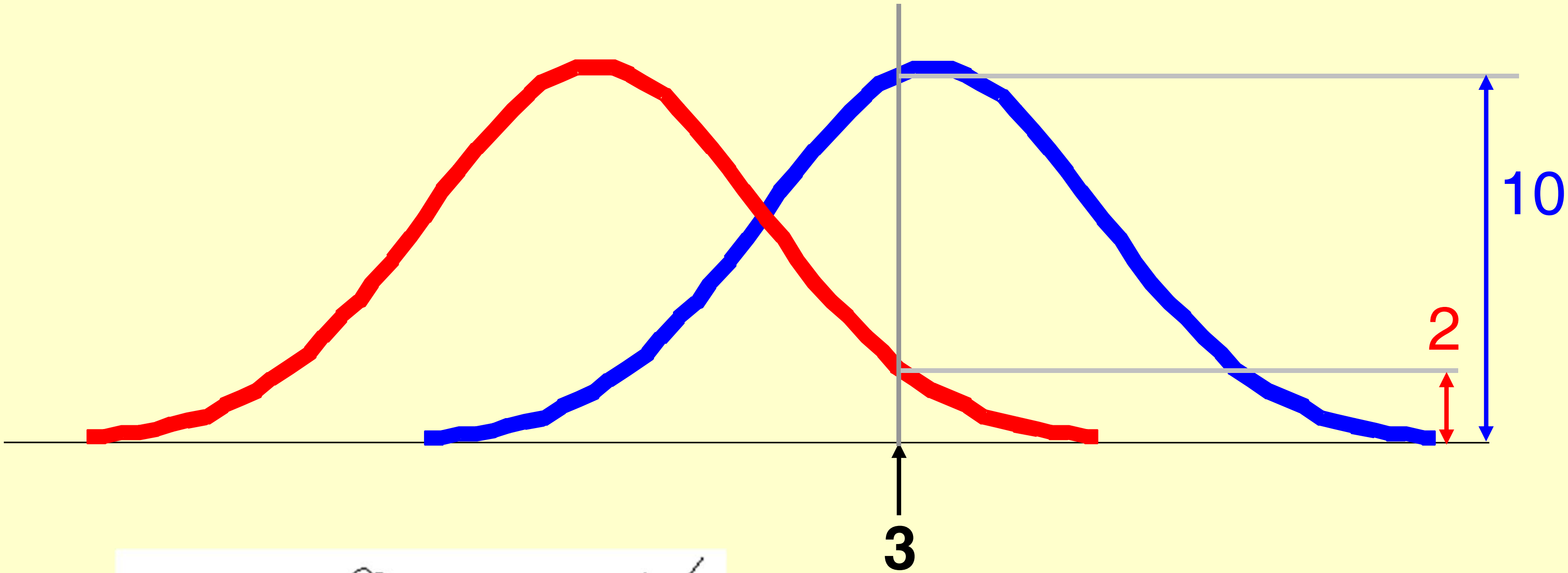
Antennae length is **3**

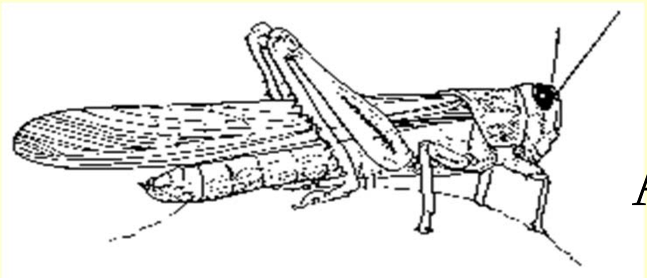
Courtesy of Eammon Keogh, UCR, eamonn@cs.ucr.edu

$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **3**) = 10 / (10 + 2) = 0.833

P(**Katydid** | **3**) = 2 / (10 + 2) = 0.166





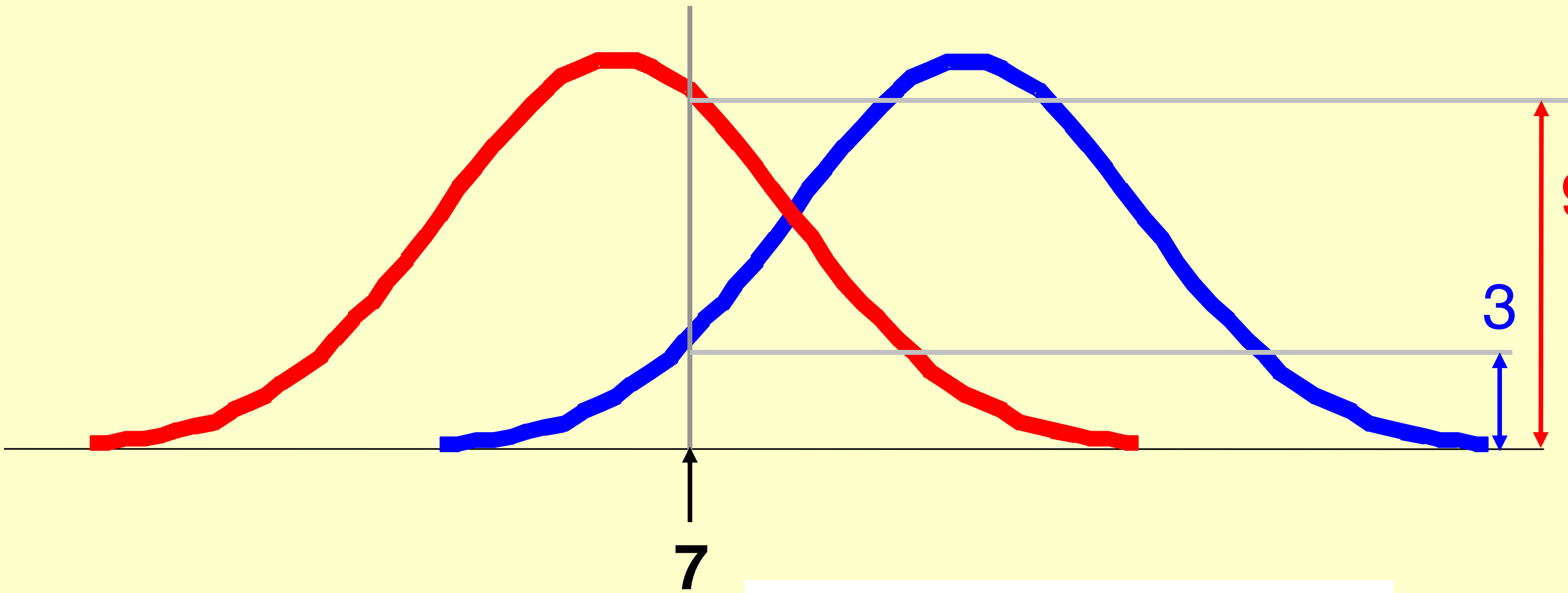
Antennae length is **3**

Courtesy of Eammon Keogh, UCR, eamonn@cs.ucr.edu

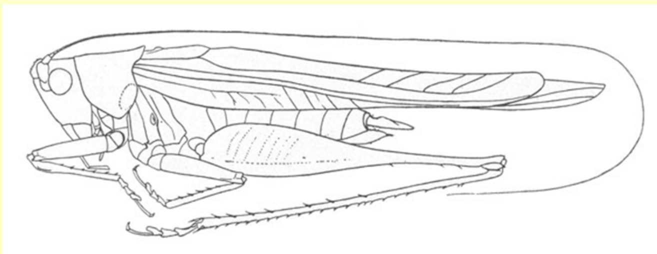
$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **7**) = 3 / (3 + 9) = 0.250

P(**Katydid** | **7**) = 9 / (3 + 9) = 0.750



Antennae length is **7**

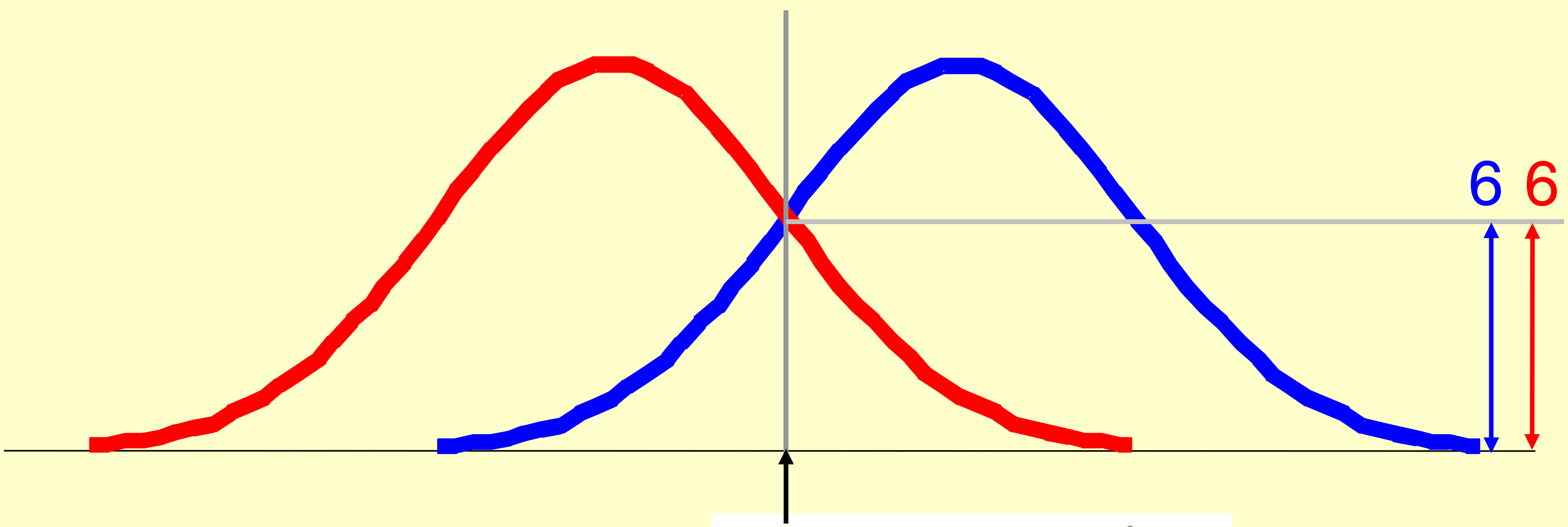


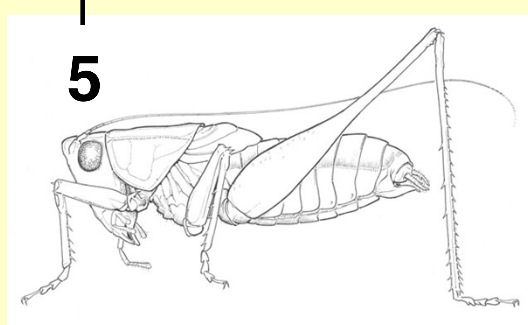
Courtesy of Eammon Keogh, UCR, eamonn@cs.ucr.edu

$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **5**) = 6 / (6 + 6) = 0.500

P(**Katydid** | **5**) = 6 / (6 + 6) = 0.500





Antennae length is **5**

Courtesy of Eammon Keogh, UCR, eamonn@cs.ucr.edu

Minimum Description Length

revisiting the def. of h*MAP*:

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

we can rewrite it as:

$$h_{MAP} = \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h)$$

or

$$h_{MAP} = \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h)$$

But the first log is the cost of coding the data *given* the theory, and the second - the cost of coding the theory

Observe that:

for data, we only need to code the exceptions; the others are correctly predicted by the theory

MAP principles tells us to choose the theory which encodes the data in the shortest manner

the MDL states the trade-off between the complexity of the hypo. and the number of errors

Bayes optimal classifier

- so far, we were looking at the "most probable hypothesis, given a priori probabilities". But we really want the most probable classification
- this we can get by combining the predictions of all hypotheses, weighted by their posterior probabilities:

$$P(v_j|D) = \sum_{h_i} P(v_j|h_i)P(h_i|D)$$

- this is the bayes optimal classifier BOC:

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

Example of hypotheses
h1, h2, h3 with posterior probabilities
.4, .3. .3
A new instance is classif. pos. by h1 and
neg. by h2, h3

Bayes optimal classifier

V = {+, -}

P(h1|D) = .4, P(-|h1) = 0, P(+|h1) = 1

…

Classification is " –" (show details!)

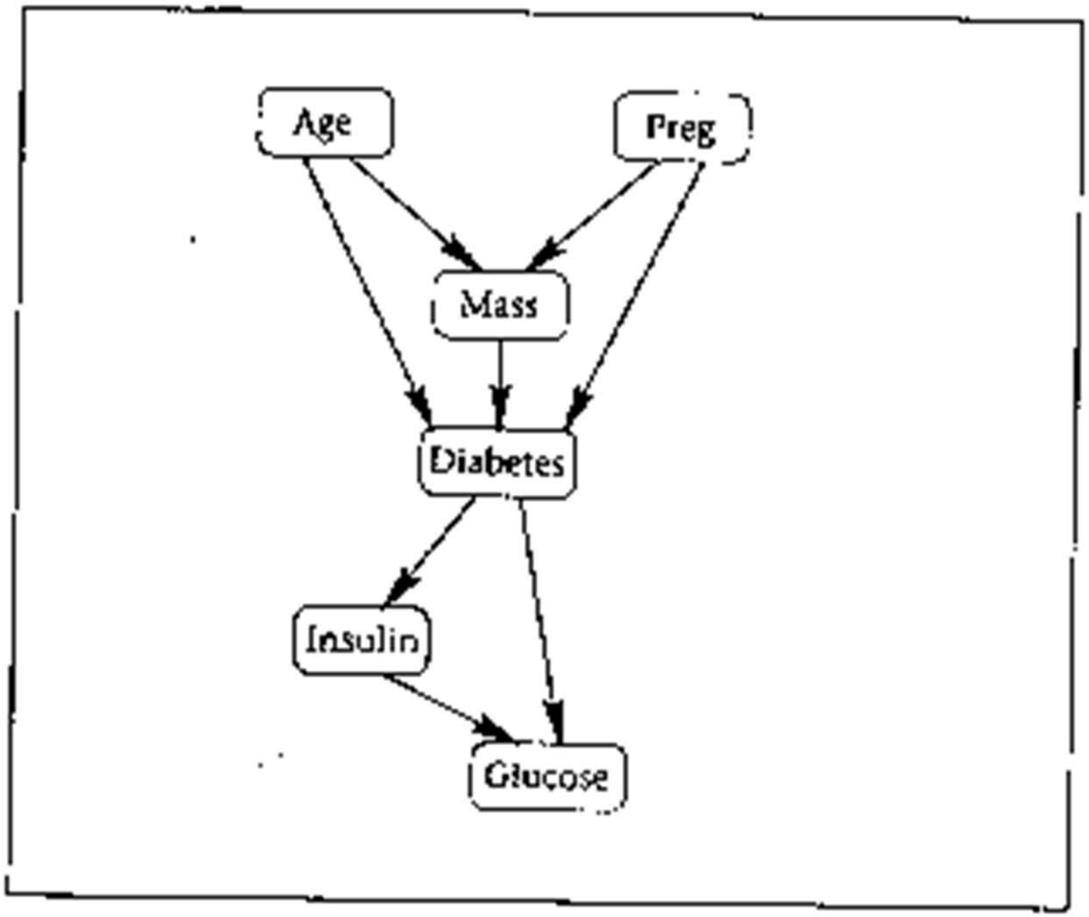


Figure 19. A Probabilistic Network for Diabetes Diagnosis.

- **Captures probability dependencies**
- **ea node has probability distribution: the task is to determine the join probability on the data**
- **In an appl. a model is designed manually and forms of probability distr. Are given**
- **Training set is used to fit the model to the data**
- **Then probabil. Inference can be carried out, eg for prediction**

First five variables are observed, and the model is Used to predict diabetes

$$P(A, N, M, I, G, D)=P(A)*P(n)*P(M|A, n)*P(D|M, A, N)*P(I|D)*P(G|I,D)$$

Age	P(A)
0–25	
26–50	
51–75	
> 75	

Preg	P(N)
0	
1	
>1	

		P(M|A, N)		
Age	Preg	0–50	51–100	>100
0–25	0			
0–25	1			
0–25	> 1			
26–50	0			
26–50	1			
26–50	>1			
51–75	0			
51–75	1			
51–75	>1			
>75	0			
>75	1			
>75	> 1			

Table 3. Probability Tables for the Age, Preg, and Mass Nodes from Figure 19.
A learning algorithm must fill in the actual probability values based on the observed training data.

- **how do we specify prob. distributions?**
- **discretize variables and represent probability distributions as a table**
- **Can be approximated from frequencies, eg table P(M|A, N) requires 24parameters**
- **For prediction, we want (D|A, n, M, I, G): we need a large table to do that**

- no other classifier using the same hypo. space e and prior K can outperform BOC
- the BOC has mostly a theoretical interest; practically, we will not have the required probabilities
- another approach, Naive Bayes Classifier (NBC)

$$v_{MAP} = \operatorname*{argmax}_{v_j \in V} P(v_j \mid a_1, \ldots a_n) = \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, \ldots a_n \mid v_j) P(v_j)}{P(a_1, \ldots a_n)} =$$

$$\operatorname*{argmax}_{v_j \in V} P(a_1, \ldots a_n \mid v_j) P(v_j)$$

To estimate this, we need (#of possible values)*(#of possible instances) examples

under a **simplifying** assumption of independence of the attribute values given the class value:

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

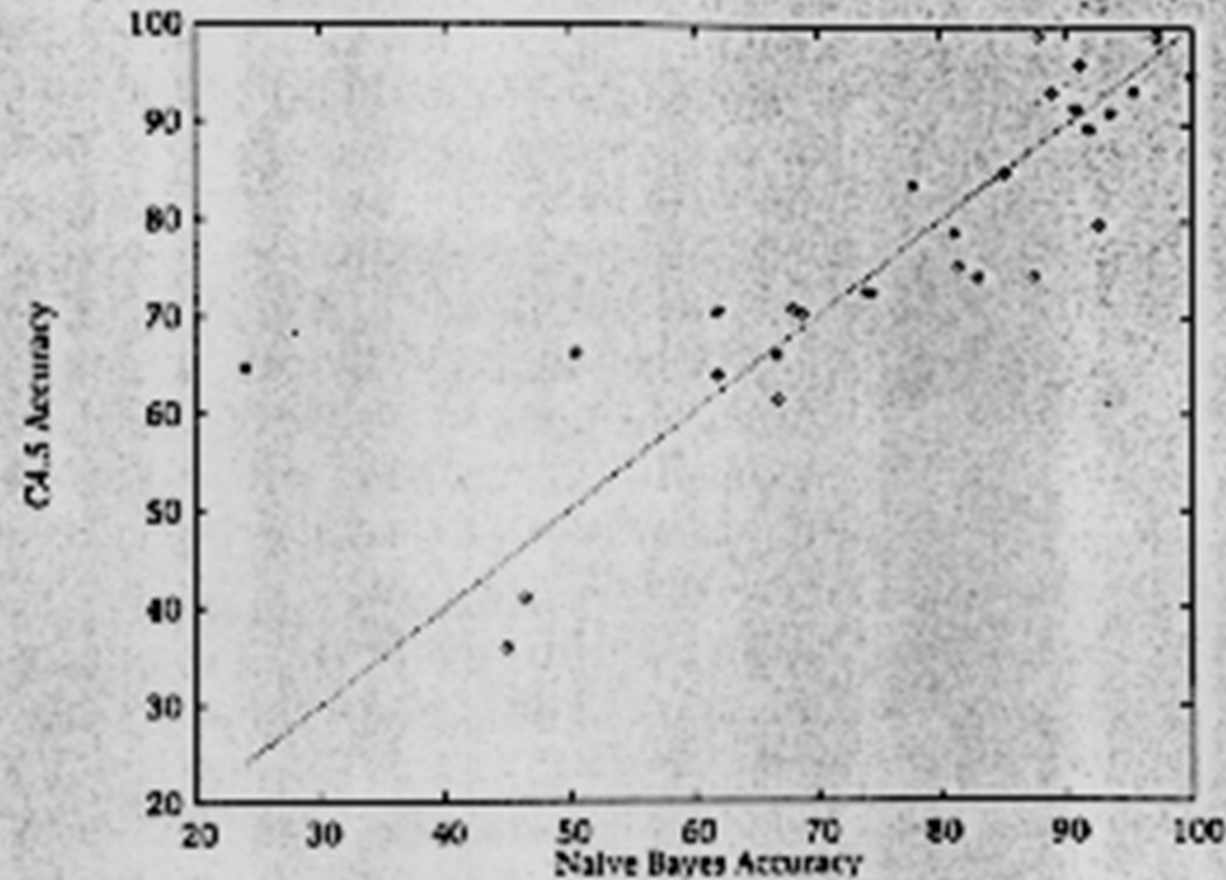


Figure 21. Comparison of C4.5 and the Naive Bayesian Classifier on 28 Data Sets.

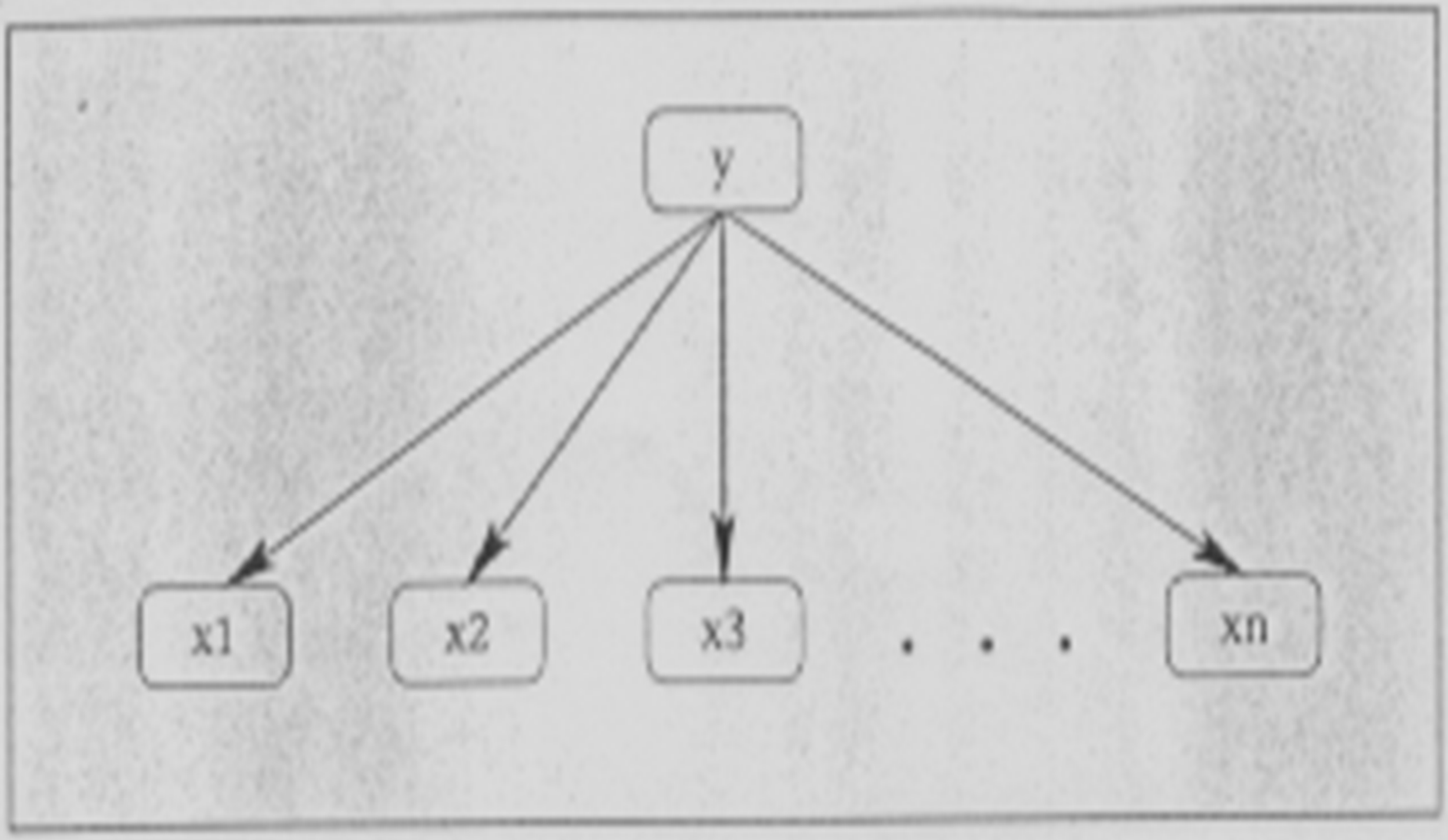


Figure 20. Probabilistic Network for the Naive Bayes Classifier.

- in NB, the conditional probabilities are *estimated* from training data simply as normalized frequencies: how many times a given attribute value is associated with a given class wrt to all classes: $\frac{n_c}{n}$
- no search!
- example

Example we are trying to predict ***yes*** or ***no*** for ***Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong***

$$v_{NB} = \arg\max_{v_j \in [yes,no]} P(v_j) \prod_i P(a_i \mid v_j) = \arg\max_{v_j \in [yes,no]} P(v_j) P(Outlook = sunny \mid v_j)$$

$$P(Temperature = cool \mid v_j) P(Humidity = high \mid v_j) P(Wind = strong \mid v_j)$$

P(yes)=9/14   P(no)=5/14

P(Wind=strong|yes)=3/9 P(Wind=strong|no)=3/5 etc.

P(yes)P(sunny|yes)P(cool|yes)P(high|yes)Pstrong|yes)=.0053

P(no)P(sunny|no)P(cool|no)P(high|no)Pstrong|no)=.0206

so we will predict *no*

Geometric decision boundary

- Assume a binary NB classifier *f* with instances $[x_1,\ldots,x_n,y]$, $y=0$ or $y=1$. Denote by v_0 (v_1) the vector of probabilities of all instances belonging to class *0 (1),* respectively.

$$f(x) = \log \frac{P(y=1\,|\,x)}{P(y=0\,|\,x)} = \log P(y=1\,|\,x) - \log P(y=0\,|\,x) =$$

$$(\log v_1 - \log v_0)x + \log p(y=1) - \log p(y=0)$$

- This expression is linear in x*.* Therefore the decision boundary of the NB classifier is linear in the feature space *X,* and is defined by $f(x) = 0$*.*

- Further, we can not only have a decision, but also the prob. of that decision: $\frac{.0206}{.0206+.0053} = .795$
- we rely on $\frac{n_c}{n}$ for the conditional probability, where n is the total number of instances for a given class, n_c is how many among them have a specific attribute value
- if we do not observe any values of , or very few, this is a problem for the NB classifier (multiplications!)
- So: smoothen; see Witten p. 91

- we will use the estimate $\frac{n_c + mp}{n + m}$

  where p is the <u>prior</u> estimate of probability,

  m is *p=1/k* for *k* values of the attribute; *m* has the effect of augmenting the number of samples of class ;

  large value of m means that priors p are important wrt training data when probability estimates are computed, small – less important
- In practice often *1* is used for *mp* and *m*

Application: text classification

- setting: newsgroups, preferences, etc. Here: ‘like’ and ‘not like’ for a set of documents
- text representation: “bag of words”: Take the union of all words occurring in all documents. A specific document is represented by a binary vector with 1’s in the positions corresponding to words which occur in this document
- high dimensionality (tens of thou. of features)

$$v_{NBC} = \max_{v_j \in like, notlike} \{P(like)P(w_1 \mid like)...P(w_n \mid like),$$
$$(P(notlike)(P(w_1 \mid notlike)...P(w_n \mid notlike)\}$$

- We will estimate $P(w_k|v_j)$ as m-estimate with equal priors

$$\frac{n_k + 1}{n + |vocabulary|}$$

- incorrectness of NB for text classification (e.g. if ‘Matwin’ occurs, the previous word is more likely to be ‘Stan’ than any other word; violates independence of features)
- but amazingly, in practice it does not make a big difference

LEARN_NAIVE_BAYES_TEXT(*Examples*, *V*)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

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

- *Vocabulary* ← the set of all distinct words and other tokens occurring in any text document from *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
  - $docs_j$ ← the subset of documents from *Examples* for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← a single document created by concatenating all members of $docs_j$
  - n ← total number of distinct word positions in $Text_j$
  - for each word w_k in *Vocabulary*
    - n_k ← number of times word w_k occurs in $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

Return the estimated target value for the document Doc. a_i denotes the word found in the ith position within Doc.

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB}, where

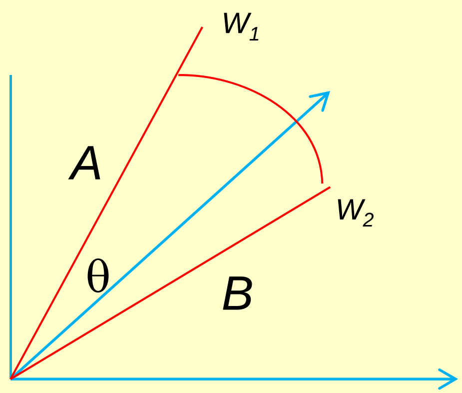
$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}}\, P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Taking into account frequencies of words

- In order to determine the weight of term *k* for the representation of document *j*, the *term frequency inverted document frequency* (*tfidf*) is often used. This function is defined as:
- $tfidf(t_k,d_j) = \#(t_k, d_j) * \log (|Tr| / \#(t_k))$
- where Tr is the training set, $\#(t_k, d_j)$ is the number of times t_k occurs in d_j, and $\#(t_k)$ is the number of documents in Tr in which t_k occurs at least once (the document frequency of tk.) Meaning?
- To make the weights fall in the [0,1] interval and for the documents to be represented by vectors of equal length, the following cosine normalization is used:
- $w_{k,j} = tfidf(t_k, d_j) / sqrt(\sum_{s=1..r} (tfidf(t_s,d_j))^2)$

Geometric interpretation

- *n*-dimensional space, where *n* = |V|
- Documents are *n*-dimensional vectors
- Distance (similarity) between documents – cosine:



$$\cos(A,B) = \frac{A \bullet B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

- distance(1 – highest, 0 – most independent); similarity = 1 – *cos* distance;

Measures for text classification

Refer to the contingency table:

- Precision (Pr) = TP / (TP + FP)
- Recall (Re) = TP / (TP + FN)

Complementarity of R & P, break-even

- Also, the F_α-measure:= $(1+\alpha)P*R/(\alpha P+R)$
- For $\alpha=1$, F-measure

Bayesian algorithms for text categorization

Naive Bayes for and against

- Naive Bayes attractive features: simple model, easy to implement and fast
- Naive Bayes has its share of shortcomings, primarily due to its strict assumptions
- If only presence/absence of word is represented, we have a multi-variate Bernoulli model for NB

Naive Bayes. Next step ahead

- improving Naive Bayes by

    1. Learning better classification weights
    2. Modeling text better (transforming the data)

- The final goal is to have a fast classifier that performs almost as well as the SVM (on text)

Multinomial Naïve Bayes (MNB)

- designed for text categorization - requires BOW input data
- attempts to improve the performance of text classification by the incorporation the words frequency information
- models the distribution of words (features) in a document as a multinomial distribution

Multinomial model and classifying documents

- We assume the *generative* model: a “source” generates an *n*-word long document, from a vocabulary of *k* words ($|V| = k$)
- Here we usually find the hypothesis (model) *most likely to have generated the data* (whereas in MAP we are looking for a model most likely *given* the observed data)
- Word occurrences are ***independent***
- A new document can then be modeled by a multinomial distribution

Multinomial distribution

- in probability theory, the multinomial distribution is a generalization of the binomial distribution.
- The binomial distribution is the probability distribution of the number of "successes" in n independent Bernoulli trials, with the same probability of "success" on each trial. (n tosses of a coin)
- In a multinomial distribution, each trial results in exactly one of some fixed finite number k of possible outcomes, with probabilities p1, ..., pk (so that pi ≥ 0 for i = 1, ..., k and $\sum_{j=1}^{k} p_j$ = 1), and there are n independent trials. Then let the random variables Xi indicate the number of times outcome number i was observed over the n trials. X=(X1,…,Xk) follows a multinomial distribution with parameters n and p, where p = (p1, ..., pk).

Multinomial Distribution

The probability mass function of the multinomial distribution is:

$$f(x_1, \ldots\ldots x_k, n, p_1, \ldots\cdot, p_k) =$$

$$Pr\left(X_1 = x_1 \, and \, \ldots \, and \, X_k = x_k\right) =$$

$$\begin{cases} \dfrac{n!}{x_1! \ldots x_k!} p_1^{x_k} \ldots\cdot p_k^{x_k}, & when \ \sum_{i=1}^{k} x_i = n \\ 0 & otherwise, \end{cases}$$

for non-negative integers $x_1, \ldots x_k$

$c \in \{1, 2, \ldots m\}$

Multinomial parameters

Each class $c \in \{1, 2, \ldots m\}$ has a fixed set of multinomial parameters $\theta_c = \{\theta_{c1}, \theta_{c2}, \ldots, \theta_{cN}\}$ (N is th size of the vocabulary) :

θ_{ci} is the probability that word *i* occurs in documents of class *c*,

$$\sum_i \theta_{ci} = 1$$

Multinomial parameters

$$\theta_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha}$$

N_{ci} is the number of times word *i* appears in the documents of class *c*

N_c is the total number of word occurrences in class *c*

α_i is a smoothing parameter

α denotes the sum of the α_i

The likelihood of a document in multinomial model

$$p\left(d|\theta_c\right) = \frac{\left(\sum_i f_i\right)!}{\prod_i f_i!} \prod_i \left(\theta_{ci}\right)^{f_i}$$

f_i is the frequency count of feature *i* in document *d*

Classification rule for MNB

$$l_{MNB}(d) = argmax_c \left[\log p\left(\theta_c\right) + \sum_i f_i \log \frac{N_{ci} + \alpha_i}{N_c + \alpha} \right]$$

$$= argmax_c \left[b_c + \sum_i f_i w_{ci} \right]$$

$p(\theta_c)$ is the class prior estimate

Threshold term : $b_c = \log p\left(\theta_c\right)$

Class ***c*** weight for word ***i***: $w_{ci} = \log \theta_{ci}$

(weights for the MNB decision boundary)

MNB (Multinomial naïve Bayes classifier)

- MNB model:

$$P(d \mid c) = \frac{(\sum_i f_i)!}{\prod_i f_i!} \prod_{i=1} P(w_i \mid c)^{f_i}$$

- where f_i = # of occurrences of word w_i in d
- Three independence assumptions:
  - occurrence of w_i is independent of occurrences of all the other words
  - occurrence of w_i is independent of itself
  - $|d|$ is independent of class of d
- MNB classifier:

$$P(c \mid d) = \frac{P(c)\prod_{i=1}^{n} P(w_i \mid c)^{f_i}}{P(d)} \qquad (1)$$

Unbalanced Training data problem (Skewed Data Bias)

- Skewed data – more training examples for one class then another
- Problem: NB and MNB heavily favor classes with more training
- Fewer samples ->smaller weights
- Frequently, the class of interest is significantly smaller, and as a result could be underweighted. It leads to the poor performance of the classifier
- Solution: Calculate score for class using statistics from all other classes; pick class with minimum score
  (More examples -> smaller bias)

Complement Naïve Bayes (CNB)

- Addressed to text categorization on unbalanced training set
- Based on heuristic solution to modify the estimation and classification rules by using the “complement class”
- complement class to the current class includes all other classes except the current class
- imbalanced class estimation is based on a more even amount of training data (as result, more stable weights estimation)

CNB parameters

are estimated as:

$$\theta_{\bar{c}\,i} = \frac{N_{\bar{c}\,i} + \alpha_i}{N_{\bar{c}} + \alpha}$$

$N_{\bar{c}i}$ is the number of times feature *i* occurred in documents of classes other than *c*

$N_{\bar{c}}$ is the total number of feature occurrences in classes other than *c*

Classification rule for CNB

$$l_{CNB}(d) = argmax_c \left[\log p(\theta_c) - \sum_i f_i \log \frac{N_{\bar{c}i} + \alpha_i}{N_{\bar{c}} + \alpha} \right]$$

The negative sign represents the fact that we want to assign to class ***c*** documents that poorly match the complement parameter estimates.

Weight Magnitude Errors

When the magnitude of Naive Bayes' weight vector w_c is larger in one class than the others, the larger magnitude class may be preferred

Since the weight differences could be partially an artifact of applying the independence assumption to dependent data, Naive Bayes gives more influence to classes that most violate the independence assumption

For Example: Class 1 is “Boston,” Class 2 is “San Francisco”. Since “San” and “Francisco” are counted independently, single occurrence of “San Francisco" will contribute twice the weight of the occurrence of Boston. It leads to incorrect classification

Solution: Normalize weight vector:

Better weights: Normalization

correct for the fact that some classes have greater dependencies by normalizing the weight vectors

$$w_{ci} = \frac{\log\theta_{ci}}{\sum_k |\log\theta_{ck}|}$$

Weight-normalized Complement Naive Bayes (WCNB).

References

- McCallum, A., & Nigam, K. (1998). A comparison of event models for naive Bayes text classification. Proceedings of AAAI '98.
- J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger (2003). Tackling the poor assumptions of Naive Bayes text classifiers. In T. Fawcett and N. Mishra (eds.), International Conference on Machine Learning Washington D.C.: Morgan Kaufmann