Classification (2)

COMP9417 Machine Learning and Data Mining

Term 2, 2019

Acknowledgements

Material derived from slides for the book
"Elements of Statistical Learning (2nd Ed.)" by T. Hastie,
R. Tibshirani & J. Friedman. Springer (2009)
http://statweb.stanford.edu/~tibs/ElemStatLearn/

Material derived from slides for the book
"Machine Learning: A Probabilistic Perspective" by P. Murphy
MIT Press (2012)
http://www.cs.ubc.ca/~murphyk/MLbook

Material derived from slides for the book
"Machine Learning" by P. Flach
Cambridge University Press (2012)
http://cs.bris.ac.uk/~flach/mlbook

Material derived from slides for the book
"Bayesian Reasoning and Machine Learning" by D. Barber
Cambridge University Press (2012)
http://www.cs.ucl.ac.uk/staff/d.barber/brml

Material derived from slides for the book
"Machine Learning" by T. Mitchell
McGraw-Hill (1997)
http://www-2.cs.cmu.edu/~tom/mlbook.html

Material derived from slides for the course
"Machine Learning" by A. Srinivasan
BITS Pilani, Goa, India (2016)

Aims

This lecture will continue your exposure to machine learning approaches to the problem of *classification*. Following it you should be able to reproduce theoretical results, outline algorithmic techniques and describe practical applications for the topics:

- explain the concept of inductive bias in machine learning
- outline Bayes Theorem as applied in machine learning
- define MAP and ML inference using Bayes theorem
- define the Bayes optimal classification rule in terms of MAP inference
- outline the Naive Bayes classification algorithm
- describe typical applications of Naive Bayes for text classification
- outline the logistic regression classification algorithm

Introduction

What do we understand about the problem of learning classifiiers ?

. . . how can we know when classifier learning succeeds ?

and . . . can we use this to build practical algorithms ?

Step back

Inductive Bias

All models are wrong, but some models are useful.

Box & Draper (1987)

Inductive bias 归纳偏向

The inductive bias (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered.

Inductive Bias

Confusingly, "inductive bias" is *NOT* the same "bias" as in the "bias-variance" decomposition.

"Inductive bias" is the combination of assumptions and restrictions placed on the models and algorithms used to solve a learning problem.

Essentially it means that the algorithm and model combination you are using to solve the learning problem is appropriate for the task.

Success in machine learning requires understanding the inductive bias of algorithms and models, and choosing them appropriately for the task.

Inductive Bias

Unfortunately, for most machine learning algorithms it is not always easy to know what their inductive bias is.

For example, what is the inductive bias of:

- Linear Regression ?
  -
  -
- Nearest Neighbour ?
  -
  -

Inductive Bias

Unfortunately, for most machine learning algorithms it is not always easy to know what their inductive bias is.

For example, what is the inductive bias of:

- Linear Regression ?
  - target function has the form $y = ax + b$
  - approximate by fitting using MSE
- Nearest Neighbour ?
  - target function is a complex non-linear function of the data
  - predict using nearest neighbour by Euclidean distance in feature space

Inductive Bias

What we would really like:

- a framework for machine learning algorithms
- with a way of representing the inductive bias
- ideally, should be a **declarative** specification
- also should quantify **uncertainty** in the inductive bias

A probabilistic approach

Uncertainty

As far as the laws of mathematics refer to reality, they are not certain; as far as they are certain, they do not refer to reality.

–Albert Einstein

A simple probabilistic model

'Viagra' and 'lottery' are two Boolean features; Y is the class variable, with values 'spam' and 'ham'. In each row the most likely class is indicated in bold.

Viagra	lottery	$P(Y = \text{spam}\|\text{Viagra, lottery})$	$P(Y = \text{ham}\|\text{Viagra, lottery})$
0	0	0.31	**0.69**
0	1	**0.65**	0.35
1	0	**0.80**	0.20
1	1	0.40	**0.60**

Decision rule

Assuming that X and Y are the only variables we know and care about, the posterior distribution $P(Y|X)$ helps us to answer many questions of interest.

- For instance, to classify a new e-mail we determine whether the words 'Viagra' and 'lottery' occur in it, look up the corresponding probability $P(Y = \text{spam}|\text{Viagra}, \text{lottery})$, and predict spam if this probability exceeds 0.5 and ham otherwise.
- Such a recipe to predict a value of Y on the basis of the values of X and the posterior distribution $P(Y|X)$ is called a *decision rule*.

Bayesian Machine Learning

Two Roles for Bayesian Methods

Provides practical learning algorithms:

- Naive Bayes classifier learning
- Bayesian network learning, etc.
- Combines prior knowledge (prior probabilities) with observed data
- How to get prior probabilities ?

Provides useful conceptual framework:

- Provides a "gold standard" for evaluating other learning algorithms
- Some additional insight into Occam's razor

Bayes Theorem

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

where

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data D
- $P(h|D)$ = probability of h given D 在给定D的情况下h发送的概率
- $P(D|h)$ = probability of D given h

Choosing Hypotheses

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

Generally want the most probable hypothesis given the training data
Maximum a posteriori hypothesis h_{MAP}:

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(h|D) \\ &= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg\max_{h \in H} P(D|h)P(h) \end{aligned}$$

hypothesis 和 probability 的区别在于,hypothesis表示的是一个比例如H_map,他们的和不为1
而probabilities的和为1

Choosing Hypotheses

If assume $P(h_i) = P(h_j)$ then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i)$$

Applying Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, $.008$ of the entire population have this cancer.

$$P(cancer) = \qquad P(\neg cancer) =$$
$$P(\oplus \mid cancer) = \qquad P(\ominus \mid cancer) =$$
$$P(\oplus \mid \neg cancer) = \qquad P(\ominus \mid \neg cancer) =$$

Applying Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, $.008$ of the entire population have this cancer.

$$P(cancer) = .008 \qquad P(\neg cancer) = .992$$
$$P(\oplus \mid cancer) = .98 \qquad P(\ominus \mid cancer) = .02$$
$$P(\oplus \mid \neg cancer) = .03 \qquad P(\ominus \mid \neg cancer) = .97$$

Applying Bayes Theorem

Does patient have cancer or not?

We can find the maximum a posteriori (MAP) hypothesis

$$\begin{aligned} P(\oplus \mid cancer)P(cancer) &= 0.98 \times 0.008 = 0.00784 \\ P(\oplus \mid \neg cancer)P(\neg cancer) &= 0.03 \times 0.992 = 0.02976 \end{aligned}$$

Thus $h_{MAP} = \ldots$.

Applying Bayes Theorem

Does patient have cancer or not?

We can find the maximum a posteriori (MAP) hypothesis

$$\begin{aligned} P(\oplus \mid cancer)P(cancer) &= 0.98 \times 0.008 = 0.00784 \\ P(\oplus \mid \neg cancer)P(\neg cancer) &= 0.03 \times 0.992 = 0.02976 \end{aligned}$$

Thus $h_{MAP} = \neg cancer$.

Also note: posterior probability of hypothesis $cancer$ higher than prior.

Applying Bayes Theorem

How to get the posterior probability of a hypothesis h ?
Divide by $P(\oplus)$, probability of data, to normalize result for h:

$$P(h|D) = \frac{P(D|h)P(h)}{\sum_{h_i \in H} P(D|h_i)P(h_i)}$$

Denominator ensures we obtain posterior probabilities that sum to 1.
Sum for all possible numerator values, since hypotheses are mutually exclusive (e.g., patient either has cancer or does not).
Marginal likelihood (marginalizing out likelihood over all possible values in the hypothesis space) — prior probability of the data.

Basic Formulas for Probabilities

Product Rule: probability $P(A \wedge B)$ of conjunction of two events A and B:

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

Sum Rule: probability of 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 $\sum_{i=1}^{n} P(A_i) = 1$, then:

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Basic Formulas for Probabilities

Also worth remembering:

- *Conditional Probability*: probability of A given B:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

- Rearrange sum rule to get:

$$P(A \wedge B) = P(A) + P(B) - P(A \vee B)$$

Exercise: (if you haven't done it before) derive Bayes Theorem.

Brute Force MAP Hypothesis Learner

Idea: view learning as finding the *most probable* hypothesis

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

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

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

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

Relation to Concept Learning (i.e., classification)

Canonical concept learning task:

- instance space X, hypothesis space H, training examples D
- consider a learning algorithm that outputs most specific hypothesis from the *version space* $VS_{H,D}$ (i.e., set of all consistent or "zero-error" classification rules)

What would Bayes rule produce as the MAP hypothesis?

Does this algorithm output a MAP hypothesis??

Relation to Concept Learning (i.e., classification)

Brute Force MAP Framework for Concept Learning:

Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$

Assume D is the set of classifications $D = \langle c(x_1), \ldots, c(x_m) \rangle$

Choose $P(h)$ to be *uniform* distribution:

- $P(h) = \frac{1}{|H|}$ for all h in H

Choose $P(D|h)$:

- $P(D|h) = 1$ if h consistent with D
- $P(D|h) = 0$ otherwise

Relation to Concept Learning (i.e., classification)

Then:

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Relation to Concept Learning (i.e., classification)

Note that since likelihood is zero if h is inconsistent then the posterior is also zero. But how did we obtain the posterior for consistent h ?

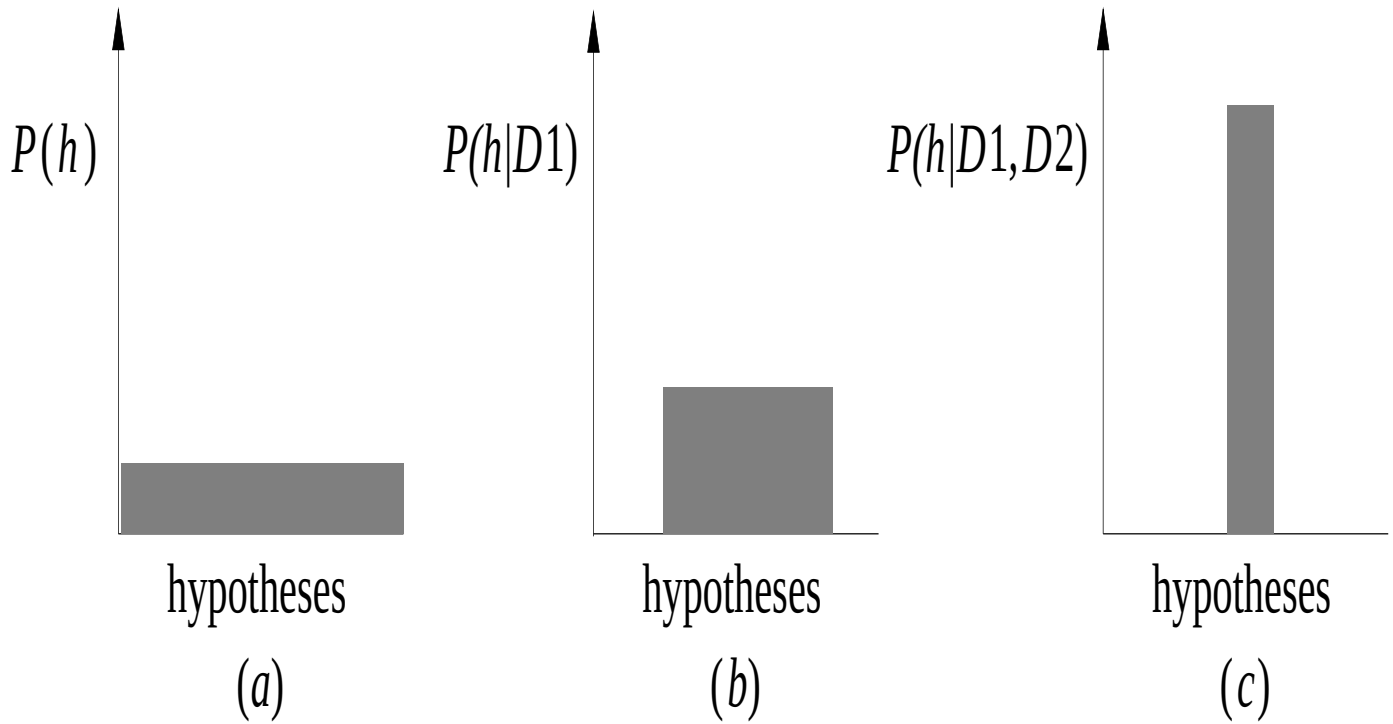
$$
\begin{aligned}
P(h|D) &= \frac{1 \cdot \frac{1}{|H|}}{P(D)} \\
&= \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} \\
&= \frac{1}{|VS_{H,D}|}
\end{aligned}
$$

Relation to Concept Learning (i.e., classification)

How did we obtain $P(D)$? From theorem of total probability:

$$
\begin{aligned}
P(D) &= \sum_{h_i \in H} P(D|H_i)P(h_i) \\
&= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} \\
&= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} \\
&= \frac{|VS_{H,D}|}{|H|}
\end{aligned}
$$

Evolution of Posterior Probabilities



Relation to Concept Learning (i.e., classification)

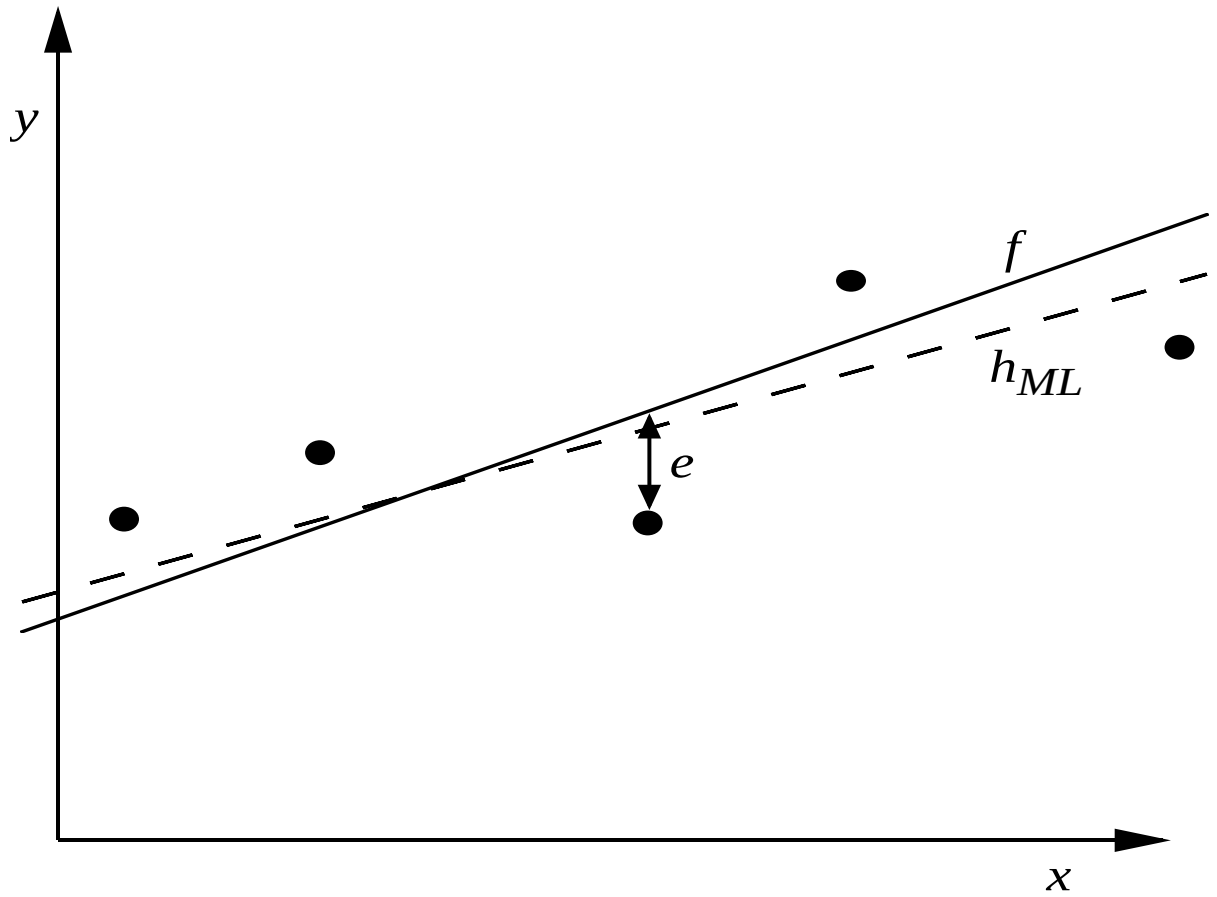
Every hypothesis consistent with D is a MAP hypothesis, if we assume

- uniform probability over H
- target function $c \in H$
- deterministic, noise-free data
- etc. (see above)

So, this learning algorithm *will* output a MAP hypothesis, even though it does not explicitly use *probabilities* in learning.

Learning A Real Valued Function

E.g., learning a linear target function f from noisy examples:



Learning A Real Valued Function

Consider any real-valued target function f
Training examples $\langle x_i, d_i \rangle$, where d_i is noisy training value

- $d_i = f(x_i) + e_i$
- e_i is random variable (noise) drawn independently for each x_i according to some Gaussian (normal) distribution with mean=0

Then the **maximum likelihood** hypothesis h_{ML} is the one that **minimizes the sum of squared errors**:

$$h_{ML} = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

Learning A Real Valued Function

How did we obtain this ?

$$
\begin{aligned}
h_{ML} &= \arg\max_{h \in H} p(D|h) \\
&= \arg\max_{h \in H} \prod_{i=1}^{m} p(d_i|h) \\
&= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}
\end{aligned}
$$

Recall that we treat each probability $p(D|h)$ **as if** $h = f$, i.e., **we assume** $\mu = f(x_i) = h(x_i)$, which is the key idea behind maximum likelihood !

Learning A Real Valued Function

Maximize natural log to give simpler expression:

$$
\begin{aligned}
h_{ML} &= \arg\max_{h \in H} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2 \\
&= \arg\max_{h \in H} \sum_{i=1}^{m} -\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2 \\
&= \arg\max_{h \in H} \sum_{i=1}^{m} -\left(d_i - h(x_i)\right)^2
\end{aligned}
$$

Equivalently, we can minimize the positive version of the expression:

$$
h_{ML} = \arg\min_{h \in H} \sum_{i=1}^{m} \left(d_i - h(x_i)\right)^2
$$

Discriminative and generative probabilistic models

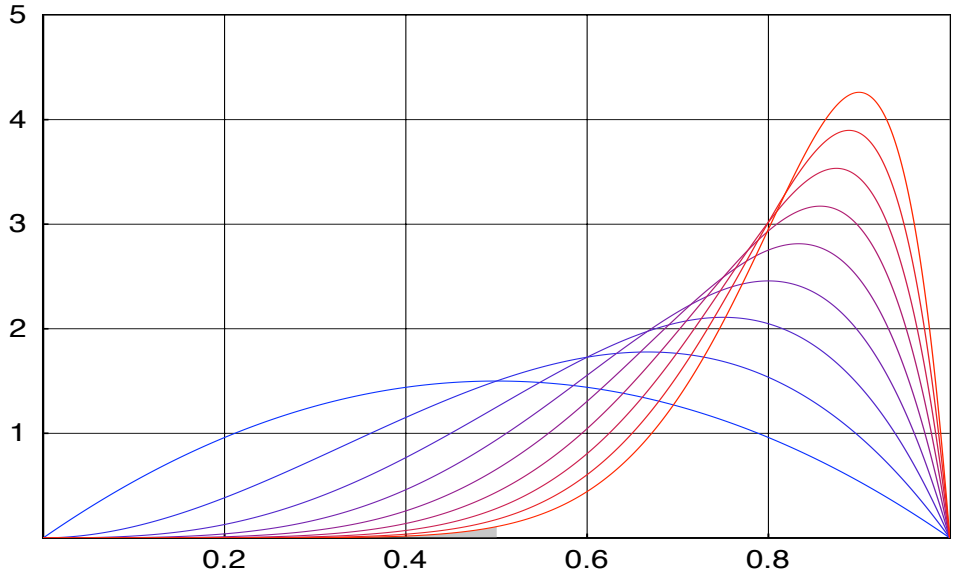
- *Discriminative models* model the posterior probability distribution $P(Y|X)$, where Y is the target variable and X are the features. That is, given X they return a probability distribution over Y.
- *Generative models* model the joint distribution $P(Y, X)$ of the target Y and the feature vector X. Once we have access to this joint distribution we can derive any conditional or marginal distribution involving the same variables. In particular, since $P(X) = \sum_y P(Y = y, X)$ it follows that the posterior distribution can be obtained as $P(Y|X) = \frac{P(Y,X)}{\sum_y P(Y=y,X)}$.
- Alternatively, generative models can be described by the likelihood function $P(X|Y)$, since $P(Y, X) = P(X|Y)P(Y)$ and the target or prior distribution (usually abbreviated to 'prior') can be easily estimated or postulated.
- Such models are called 'generative' because we can sample from the joint distribution to obtain new data points together with their labels. Alternatively, we can use $P(Y)$ to sample a class and $P(X|Y)$ to sample an instance for that class.

Assessing uncertainty in estimates

Suppose we want to estimate the probability θ that an arbitrary e-mail is spam, so that we can use the appropriate prior distribution.

- The natural thing to do is to inspect n e-mails, determine the number of spam e-mails d, and set $\hat{\theta} = d/n$; we don't really need any complicated statistics to tell us that.
- However, while this is the most likely estimate of θ – the maximum a posteriori (MAP) estimate – this doesn't mean that other values of θ are completely ruled out.
- We model this by a probability distribution over θ (a Beta distribution in this case) which is updated each time new information comes in. This is further illustrated in the figure for a distribution that is more and more skewed towards spam.
- For each curve, its bias towards spam is given by the area under the curve and to the right of $\theta = 1/2$.

Assessing uncertainty in estimates



Each time we inspect an e-mail, we are reducing our uncertainty regarding the prior spam probability θ. After we inspect two e-mails and observe one spam, the possible θ values are characterised by a symmetric distribution around $1/2$. If we inspect a third, fourth, ..., tenth e-mail and each time (except the first one) it is spam, then this distribution narrows and shifts a little bit to the right each time. The distribution for n e-mails reaches its maximum at $\hat{\theta}_{\mathrm{MAP}} = \frac{n-1}{n}$ (e.g., $\hat{\theta}_{\mathrm{MAP}} = 0.8$ for $n = 5$).

The Bayesian perspective

Explicitly modelling the posterior distribution over the parameter θ has a number of advantages that are usually associated with the 'Bayesian' perspective:

- We can precisely characterise the uncertainty that remains about our estimate by quantifying the spread of the posterior distribution.
- We can obtain a generative model for the parameter by sampling from the posterior distribution, which contains much more information than a summary statistic such as the MAP estimate can convey – so, rather than using a single e-mail with $\theta = \theta_{\mathrm{MAP}}$, our generative model can contain a number of e-mails with θ sampled from the posterior distribution.

The Bayesian perspective

- We can quantify the probability of statements such as 'e-mails are biased towards ham' (the tiny shaded area in the figure demonstrates that after observing one ham and nine spam e-mails this probability is very small, about 0.6%).
- We can use one of these distributions to encode our prior beliefs: e.g., if we believe that the proportions of spam and ham are typically 50–50, we can take the distribution for $n = 2$ (the lowest, symmetric one in the figure on the previous slide) as our prior.

The key point is that probabilities do not have to be interpreted as estimates of relative frequencies, but can carry the more general meaning of (possibly subjective) degrees of belief.

Consequently, we can attach a probability distribution to almost anything: not just features and targets, but also model parameters and even models.

Minimum Description Length Principle

Once again, the MAP hypothesis

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

Which is equivalent to

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

Or

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

Minimum Description Length Principle

Interestingly, this is an expression about a quantity of *bits*.

$$h_{MAP} = \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \quad (1)$$

From information theory:

> *The optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.*

Minimum Description Length Principle

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

Note well: assumes *optimal* encodings, when the priors and likelihoods are known. In practice, this is difficult, and makes a difference.

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

$$h_{MDL} = \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$$

where $L_C(x)$ is the description length of x under optimal encoding C

Minimum Description Length Principle

Without loss of generality, classifier is here assumed to be a decision tree

Example: H = decision trees, D = training data labels

- $L_{C_1}(h)$ is # bits to describe tree h
- $L_{C_2}(D|h)$ is # bits to describe D given h
  - Note $L_{C_2}(D|h) = 0$ if examples classified perfectly by h. Need only describe exceptions
- Hence h_{MDL} trades off tree size for training errors
  - i.e., prefer the hypothesis that minimizes

$$length(h) + length(misclassifications)$$

Most Probable Classification of New Instances

So far we've 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!

Most Probable Classification of New Instances

Consider:

- Three possible hypotheses:

$$P(h_1|D) = .4,\ P(h_2|D) = .3,\ P(h_3|D) = .3$$

- Given new instance x,

$$h_1(x) = +,\ h_2(x) = -,\ h_3(x) = -$$

- What's most probable classification of x?

Bayes Optimal Classifier

Bayes optimal classification:

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

Example:

$$\begin{aligned} P(h_1|D) &= .4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1 \\ P(h_2|D) &= .3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0 \\ P(h_3|D) &= .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0 \end{aligned}$$

Bayes Optimal Classifier

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

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

No other classification method using the same hypothesis space and same prior knowledge can outperform this method on average

Bayes Error

What is the best performance attainable by a (two-class) classifier ?

Define the probability of error for classifying some instance x by

$$\begin{aligned} P(\text{error}|x) &= P(\text{class}_1|x) \quad \text{if we predict class}_2 \\ &= P(\text{class}_2|x) \quad \text{if we predict class}_1 \end{aligned}$$

This gives

$$\Sigma_x \quad P(\text{error}) = P(\text{error}|x)\ P(x)$$

So we can justify the use of the decision rule

$$\begin{aligned} \text{if} \quad P(\text{class}_1|x) > P(\text{class}_2|x) \quad &\text{then} \quad \text{predict class}_1 \\ &\text{else} \quad \text{predict class}_2 \end{aligned}$$

On average, this decision rule minimises probability of classification error.

Naive Bayes Classifier

Along with decision trees, neural networks, nearest neighbour, one of the most practical learning methods.

When to use

- Moderate or large training set available
- Attributes that describe instances are conditionally independent given classification

Successful applications:

- Classifying text documents
- Gaussian Naive Bayes for real-valued data

Naive Bayes Classifier

Assume target function $f : X \to V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$.
Most probable value of $f(x)$ is:

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

Naive Bayes Classifier

Naive Bayes assumption:

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

- Attributes are statistically independent (given the class value)
  - which means knowledge about the value of a particular attribute tells us nothing about the value of another attribute (if the class is known)

which gives

Naive Bayes classifier: $v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

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)

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Naive Bayes Example: *PlayTennis*

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Naive Bayes Example: *PlayTennis*

What are the required probabilities to predict *PlayTennis* ?

Outlook			Temperature			Humidity			Windy		
	Yes	No		Yes	No		Yes	No		Yes	No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3
Rainy	3	2	Cool	3	1						
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5
Rainy	3/9	2/5	Cool	3/9	1/5						
Play											
Yes	No										
9	5										
9/14	5/14										

Naive Bayes Example: *PlayTennis*

Say we have the new instance:

$$\langle Outlk = sun, Temp = cool, Humid = high, Wind = true \rangle$$

We want to compute:

$$v_{NB} = \underset{v_j \in \{\text{“yes”}, \text{“no”}\}}{\arg\max} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Example: *PlayTennis*

So we first calculate the likelihood of the two classes, "yes" and "no"

$$\text{for "yes"} = P(y) \times P(sun|y) \times P(cool|y) \times P(high|y) \times P(true|y)$$

$$0.0053 = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}$$

$$\text{for "no"} = P(n) \times P(sun|n) \times P(cool|n) \times P(high|n) \times P(true|n)$$

$$0.0206 = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}$$

Naive Bayes Example: *PlayTennis*

Then convert to a probability by normalisation

$$
\begin{aligned}
P(\text{“yes”}) &= \frac{0.0053}{(0.0053 + 0.0206)} \\
&= 0.205 \\
P(\text{“no”}) &= \frac{0.0206}{(0.0053 + 0.0206)} \\
&= 0.795
\end{aligned}
$$

The Naive Bayes classification is “no”.

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. Note don't need estimated posteriors $\hat{P}(v_j|x)$ to be correct; need only that

$$\arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = \arg\max_{v_j \in V} P(v_j) P(a_1 \ldots, a_n | v_j)$$

i.e. maximum probability is assigned to correct class

- see [Domingos & Pazzani, 1996] for analysis
- Naive Bayes posteriors often unrealistically close to 1 or 0
- adding too many redundant attributes will cause problems (e.g. identical attributes)

Naive Bayes: "zero-frequency" problem

What if none of the training instances with target value v_j have attribute value a_i? Then

$$\hat{P}(a_i|v_j) = 0, \text{ and...}$$

$$\hat{P}(v_j)\prod_i \hat{P}(a_i|v_j) = 0$$

Pseudo-counts add 1 to each count (a version of the *Laplace Estimator*)

Naive Bayes: "zero-frequency" problem

- In some cases adding a constant different from 1 might be more appropriate
- Example: attribute *outlook* for class *yes*

Sunny	*Overcast*	*Rainy*
$\frac{2+\frac{\mu}{2}}{9+\mu}$	$\frac{4+\frac{\mu}{3}}{9+\mu}$	$\frac{3+\frac{\mu}{3}}{9+\mu}$

- Weights don't need to be equal (if they sum to 1) – a form of *prior*

Sunny	*Overcast*	*Rainy*
$\frac{2+\mu p_1}{9+\mu}$	$\frac{4+\mu p_2}{9+\mu}$	$\frac{3+\mu p_3}{9+\mu}$

Naive Bayes: numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The *probability density function* for the normal distribution is defined by two parameters:

  The sample mean μ:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  The standard deviation σ:

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

Naive Bayes: numeric attributes

Then we have the density function $f(x)$:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \; e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Example: continuous attribute *temperature* with mean $= 73$ and standard deviation $= 6.2$. Density value

$$f(\text{temperature} = 66 | \text{“yes”}) = \frac{1}{\sqrt{2\pi}6.2} \; e^{-\frac{(66-73)^2}{2\times 6.2^2}} = 0.0340$$

Missing values during training are not included in calculation of mean and standard deviation.

Naive Bayes: numeric attributes

Note: the normal distribution is based on the simple exponential function

$$f(x) = e^{-|x|^m}$$

As the power m in the exponent increases, the function approaches a step function.
Where $m = 2$

$$f(x) = e^{-|x|^2}$$

and this is the basis of the normal distribution – the various constants are the result of scaling so that the integral (the area under the curve from $-\infty$ to $+\infty$) is equal to 1.
from "Statistical Computing" by Michael J. Crawley (2002) Wiley.

Categorical random variables

Categorical variables or features (also called discrete or nominal) are ubiquitous in machine learning.

- Perhaps the most common form of the Bernoulli distribution models whether or not a word occurs in a document. That is, for the i-th word in our vocabulary we have a random variable X_i governed by a Bernoulli distribution. The joint distribution over the *bit vector* $X = (X_1, \ldots, X_k)$ is called a *multivariate Bernoulli distribution*.
- Variables with more than two outcomes are also common: for example, every word position in an e-mail corresponds to a categorical variable with k outcomes, where k is the size of the vocabulary. The multinomial distribution manifests itself as a *count vector*: a histogram of the number of occurrences of all vocabulary words in a document. This establishes an alternative way of modelling text documents that allows the number of occurrences of a word to influence the classification of a document.

Categorical random variables

Both these document models are in common use. Despite their differences, they both assume independence between word occurrences, generally referred to as the *naive Bayes assumption*.

- In the multinomial document model, this follows from the very use of the multinomial distribution, which assumes that words at different word positions are drawn independently from the same categorical distribution.
- In the multivariate Bernoulli model we assume that the bits in a bit vector are statistically independent, which allows us to compute the joint probability of a particular bit vector $(x_1, \ldots, x_k)$ as the product of the probabilities of each component $P(X_i = x_i)$.
- In practice, such word independence assumptions are often not true: if we know that an e-mail contains the word 'Viagra', we can be quite sure that it will also contain the word 'pill'. Violated independence assumptions reduce the quality of probability estimates but may still allow good classification performance.

Example application: Learning to Classify Text

In machine learning, the classic example of applications of Naive Bayes is learning to classify text documents.

Here is a simplified version in the multinomial document model.

Learning to Classify Text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text documents??

Learning to Classify Text

Target concept $Interesting? : Document \rightarrow \{+, -\}$

1. Represent each document by vector of words
   - one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Learning to Classify Text

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position i is w_k, given v_j

one more assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

"bag of words"

Application: 20 Newsgroups

Given: 1000 training documents from each group
Learning task: classify each new document by newsgroup it came from

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

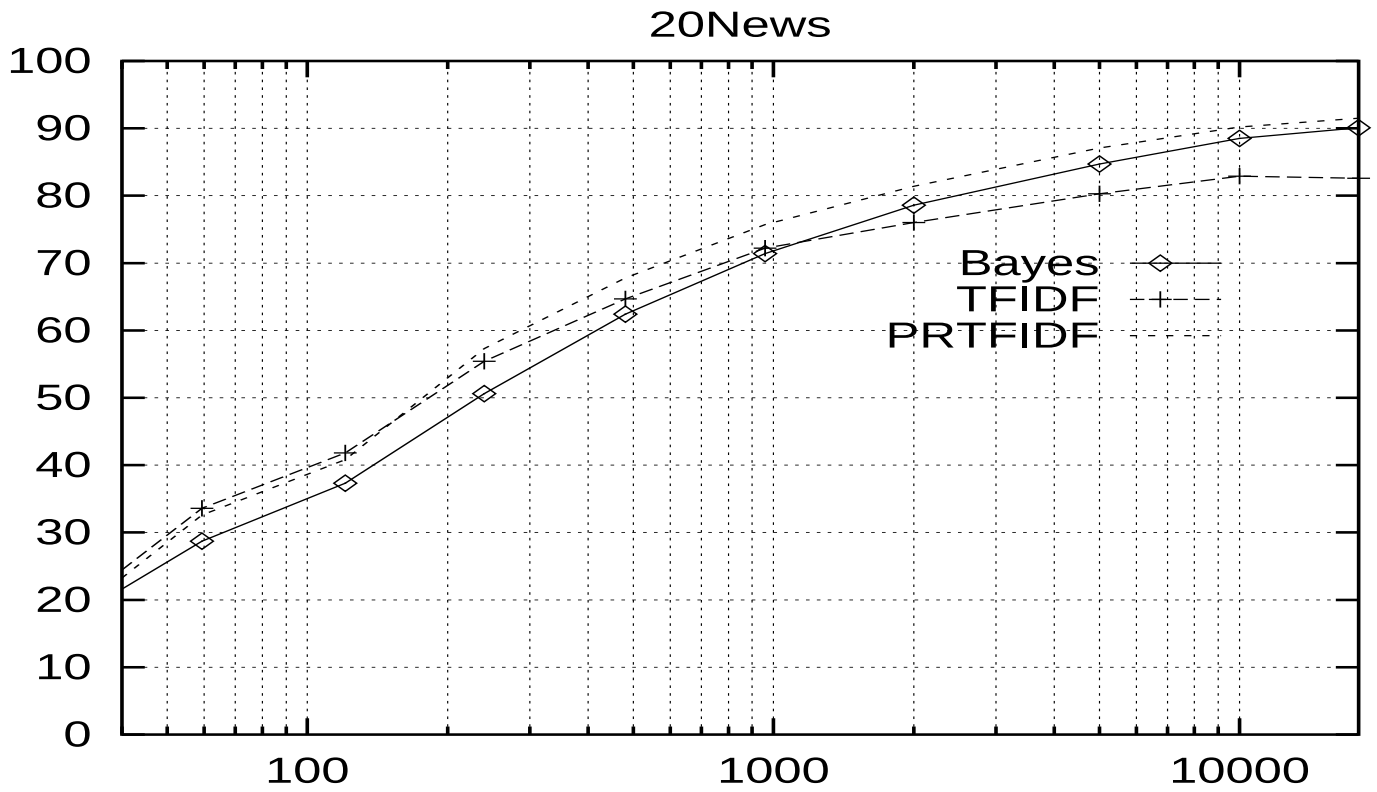
Naive Bayes: 89% classification accuracy

Article from rec.sport.hockey

```
Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!ogicse!uwm.edu
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)...
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate
for pleasant surprise is Alex Zhitnik. He came highly touted as
a defensive defenseman, but he's clearly much more than that.
Great skater and hard shot (though wish he were more accurate).
In fact, he pretty much allowed the Kings to trade away that
huge defensive liability Paul Coffey. Kelly Hrudey is only the
biggest disappointment if you thought he was any good to begin
with. But, at best, he's only a mediocre goaltender. A better
choice would be Tomas Sandstrom, though not through any fault
of his own, but because some thugs in Toronto decided ...
```

Learning Curve for 20 Newsgroups



Accuracy vs. Training set size (1/3 withheld for test)

Probabilistic decision rules

We have chosen one of the possible distributions to model our data X as coming from either class.

- The more different $P(X|Y = \text{spam})$ and $P(X|Y = \text{ham})$ are, the more useful the features X are for classification.
- Thus, for a specific e-mail x we calculate both $P(X = x|Y = \text{spam})$ and $P(X = x|Y = \text{ham})$, and apply one of several possible decision rules:

maximum likelihood (ML) – predict $\arg\max_y P(X = x|Y = y)$;

maximum a posteriori (MAP) – predict

$$\arg\max_y P(X = x|Y = y)P(Y = y);$$

The relation between the first two decision rules is that ML classification is equivalent to MAP classification with a uniform class distribution.

Probabilistic decision rules

We again use the example of Naive Bayes for text classification to illustrate, using both the multinomial and multivariate Bernoulli models.

Prediction using a naive Bayes model

Suppose our vocabulary contains three words a, b and c, and we use a multivariate Bernoulli model for our e-mails, with parameters

$$\boldsymbol{\theta}^{\oplus} = (0.5, 0.67, 0.33) \qquad \boldsymbol{\theta}^{\ominus} = (0.67, 0.33, 0.33)$$

This means, for example, that the presence of b is twice as likely in spam ($+$), compared with ham.
The e-mail to be classified contains words a and b but not c, and hence is described by the bit vector $\mathbf{x} = (1, 1, 0)$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 0.5 \cdot 0.67 \cdot (1 - 0.33) = 0.222$$
$$P(\mathbf{x}|\ominus) = 0.67 \cdot 0.33 \cdot (1 - 0.33) = 0.148$$

The ML classification of $\mathbf{x}$ is thus spam.

Prediction using a naive Bayes model

In the case of two classes it is often convenient to work with likelihood ratios and odds.

- The likelihood ratio can be calculated as

$$\frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)} = \frac{0.5}{0.67}\frac{0.67}{0.33}\frac{1-0.33}{1-0.33} = 3/2 > 1$$

- This means that the MAP classification of $\mathbf{x}$ is also spam if the prior odds are more than $2/3$, but ham if they are less than that.
- For example, with 33% spam and 67% ham the prior odds are $\frac{P(\oplus)}{P(\ominus)} = \frac{0.33}{0.67} = 1/2$, resulting in a posterior odds of

$$\frac{P(\oplus|\mathbf{x})}{P(\ominus|\mathbf{x})} = \frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)}\frac{P(\oplus)}{P(\ominus)} = 3/2 \cdot 1/2 = 3/4 < 1$$

In this case the likelihood ratio for $\mathbf{x}$ is not strong enough to push the decision away from the prior.

Prediction using a naive Bayes model

Alternatively, we can employ a multinomial model. The parameters of a multinomial establish a distribution over the words in the vocabulary, say

$$\boldsymbol{\theta}^{\oplus} = (0.3, 0.5, 0.2) \qquad \boldsymbol{\theta}^{\ominus} = (0.6, 0.2, 0.2)$$

The e-mail to be classified contains three occurrences of word a, one single occurrence of word b and no occurrences of word c, and hence is described by the count vector $\mathbf{x} = (3, 1, 0)$. The total number of vocabulary word occurrences is $n = 4$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 4! \frac{0.3^3}{3!} \frac{0.5^1}{1!} \frac{0.2^0}{0!} = 0.054$$

$$P(\mathbf{x}|\ominus) = 4! \frac{0.6^3}{3!} \frac{0.2^1}{1!} \frac{0.2^0}{0!} = 0.1728$$

The likelihood ratio is $\left(\frac{0.3}{0.6}\right)^3 \left(\frac{0.5}{0.2}\right)^1 \left(\frac{0.2}{0.2}\right)^0 = 5/16$. The ML classification of $\mathbf{x}$ is thus ham, the opposite of the multivariate Bernoulli model. This is mainly because of the three occurrences of word a, which provide strong evidence for ham.

Training data for naive Bayes

A small e-mail data set described by count vectors.

E-mail	$\#a$	$\#b$	$\#c$	Class
e_1	0	3	0	$+$
e_2	0	3	3	$+$
e_3	3	0	0	$+$
e_4	2	3	0	$+$
e_5	4	3	0	$-$
e_6	4	0	3	$-$
e_7	3	0	0	$-$
e_8	0	0	0	$-$

Training data for naive Bayes

The same data set described by bit vectors.

E-mail	$a?$	$b?$	$c?$	Class
e_1	0	1	0	$+$
e_2	0	1	1	$+$
e_3	1	0	0	$+$
e_4	1	1	0	$+$
e_5	1	1	0	$-$
e_6	1	0	1	$-$
e_7	1	0	0	$-$
e_8	0	0	0	$-$

Training a naive Bayes model

Consider the following e-mails consisting of five words a, b, c, d, e:

e_1: $b\ d\ e\ b\ b\ d\ e$
e_2: $b\ c\ e\ b\ b\ d\ d\ e\ c\ c$
e_3: $a\ d\ a\ d\ e\ a\ e\ e$
e_4: $b\ a\ d\ b\ e\ d\ a\ b$

e_5: $a\ b\ a\ b\ a\ b\ a\ e\ d$
e_6: $a\ c\ a\ c\ a\ c\ a\ e\ d$
e_7: $e\ a\ e\ d\ a\ e\ a$
e_8: $d\ e\ d\ e\ d$

We are told that the e-mails on the left are spam and those on the right are ham, and so we use them as a small training set to train our Bayesian classifier.

- First, we decide that d and e are so-called *stop words* that are too common to convey class information.
- The remaining words, a, b and c, constitute our vocabulary.

Training a naive Bayes model

For the multinomial model, we represent each e-mail as a count vector, as before.

- In order to estimate the parameters of the multinomial, we sum up the count vectors for each class, which gives $(5, 9, 3)$ for spam and $(11, 3, 3)$ for ham.
- To smooth these probability estimates we add one pseudo-count for each vocabulary word, which brings the total number of occurrences of vocabulary words to 20 for each class.
- The estimated parameter vectors are thus
  $\hat{\boldsymbol{\theta}}^{\oplus} = (6/20, 10/20, 4/20) = (0.3, 0.5, 0.2)$ for spam and
  $\hat{\boldsymbol{\theta}}^{\ominus} = (12/20, 4/20, 4/20) = (0.6, 0.2, 0.2)$ for ham.

Training a naive Bayes model

In the multivariate Bernoulli model e-mails are represented by bit vectors, as before.

- Adding the bit vectors for each class results in $(2, 3, 1)$ for spam and $(3, 1, 1)$ for ham.
- Each count is to be divided by the number of documents in a class, in order to get an estimate of the probability of a document containing a particular vocabulary word.
- Probability smoothing now means adding two pseudo-documents, one containing each word and one containing none of them.
- This results in the estimated parameter vectors
  $\hat{\boldsymbol{\theta}}^{\oplus} = (3/6, 4/6, 2/6) = (0.5, 0.67, 0.33)$ for spam and
  $\hat{\boldsymbol{\theta}}^{\ominus} = (4/6, 2/6, 2/6) = (0.67, 0.33, 0.33)$ for ham.

Naive Bayes: "zero-frequency" problem

This generalisation is a Bayesian estimate for $\hat{P}(a_i|v_j)$

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

where

- 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 prior estimate for $\hat{P}(a_i|v_j)$
- m is weight given to prior (i.e. number of "virtual" examples)

This is called the m-estimate of probability (a form of "prior").

Naive Bayes: missing values

What about missing values ? A very basic approach is:

- Training: instance is not included in frequency count for attribute value-class combination
- Classification: attribute will be omitted from calculation

Can we do better ?

Missing values

Suppose we skimmed an e-mail and noticed that it contains the word 'lottery' but we haven't looked closely enough to determine whether it uses the word 'Viagra'. This means that we don't know whether to use the second or the fourth row in to make a prediction. This is a problem, as we would predict spam if the e-mail contained the word 'Viagra' (second row) and ham if it didn't (fourth row).

The solution is to average these two rows, using the probability of 'Viagra' occurring in any e-mail (spam or not):

$$\begin{aligned} P(Y|\text{lottery}) =& P(Y|\text{Viagra} = 0, \text{lottery})P(\text{Viagra} = 0) \\ &+ P(Y|\text{Viagra} = 1, \text{lottery})P(\text{Viagra} = 1) \end{aligned}$$

Missing values

For instance, suppose for the sake of argument that one in ten e-mails contain the word 'Viagra', then $P(\text{Viagra} = 1) = 0.10$ and $P(\text{Viagra} = 0) = 0.90$. Using the above formula, we obtain $P(Y = \text{spam}|\text{lottery} = 1) = 0.65 \cdot 0.90 + 0.40 \cdot 0.10 = 0.625$ and $P(Y = \text{ham}|\text{lottery} = 1) = 0.35 \cdot 0.90 + 0.60 \cdot 0.10 = 0.375$. Because the occurrence of 'Viagra' in any e-mail is relatively rare, the resulting distribution deviates only a little from the second row in the data.

Likelihood ratio

As a matter of fact, statisticians work very often with different conditional probabilities, given by the *likelihood function* $P(X|Y)$.

- I like to think of these as thought experiments: if somebody were to send me a spam e-mail, how likely would it be that it contains exactly the words of the e-mail I'm looking at? And how likely if it were a ham e-mail instead?
- What really matters is not the magnitude of these likelihoods, but their ratio: how much more likely is it to observe this combination of words in a spam e-mail than it is in a non-spam e-mail.
- For instance, suppose that for a particular e-mail described by X we have $P(X|Y = \text{spam}) = 3.5 \cdot 10^{-5}$ and $P(X|Y = \text{ham}) = 7.4 \cdot 10^{-6}$, then observing X in a spam e-mail is nearly five times more likely than it is in a ham e-mail.
- This suggests the following decision rule: predict spam if the likelihood ratio is larger than 1 and ham otherwise.

When to use likelihoods

Use likelihoods if you want to ignore the prior distribution or assume it uniform, and posterior probabilities otherwise.

Posterior odds

$$\frac{P(Y = \text{spam}|\text{Viagra} = 0, \text{lottery} = 0)}{P(Y = \text{ham}|\text{Viagra} = 0, \text{lottery} = 0)} = \frac{0.31}{0.69} = 0.45$$

$$\frac{P(Y = \text{spam}|\text{Viagra} = 1, \text{lottery} = 1)}{P(Y = \text{ham}|\text{Viagra} = 1, \text{lottery} = 1)} = \frac{0.40}{0.60} = 0.67$$

$$\frac{P(Y = \text{spam}|\text{Viagra} = 0, \text{lottery} = 1)}{P(Y = \text{ham}|\text{Viagra} = 0, \text{lottery} = 1)} = \frac{0.65}{0.35} = 1.9$$

$$\frac{P(Y = \text{spam}|\text{Viagra} = 1, \text{lottery} = 0)}{P(Y = \text{ham}|\text{Viagra} = 1, \text{lottery} = 0)} = \frac{0.80}{0.20} = 4.0$$

Using a MAP decision rule we predict ham in the top two cases and spam in the bottom two. Given that the full posterior distribution is all there is to know about the domain in a statistical sense, these predictions are the best we can do: they are *Bayes-optimal*.

Example marginal likelihoods

Y	$P(\text{Viagra} = 1 \mid Y)$	$P(\text{Viagra} = 0 \mid Y)$
spam	0.40	0.60
ham	0.12	0.88

Y	$P(\text{lottery} = 1 \mid Y)$	$P(\text{lottery} = 0 \mid Y)$
spam	0.21	0.79
ham	0.13	0.87

Using marginal likelihoods

Using the marginal likelihoods from before, we can approximate the likelihood ratios (the previously calculated odds from the full posterior distribution are shown in brackets):

$$\frac{P(\text{Viagra} = 0|Y = \text{spam})}{P(\text{Viagra} = 0|Y = \text{ham})}\frac{P(\text{lottery} = 0|Y = \text{spam})}{P(\text{lottery} = 0|Y = \text{ham})} = \frac{0.60}{0.88}\frac{0.79}{0.87} = 0.62 \quad (0.45)$$

$$\frac{P(\text{Viagra} = 0|Y = \text{spam})}{P(\text{Viagra} = 0|Y = \text{ham})}\frac{P(\text{lottery} = 1|Y = \text{spam})}{P(\text{lottery} = 1|Y = \text{ham})} = \frac{0.60}{0.88}\frac{0.21}{0.13} = 1.1 \quad (1.9)$$

$$\frac{P(\text{Viagra} = 1|Y = \text{spam})}{P(\text{Viagra} = 1|Y = \text{ham})}\frac{P(\text{lottery} = 0|Y = \text{spam})}{P(\text{lottery} = 0|Y = \text{ham})} = \frac{0.40}{0.12}\frac{0.79}{0.87} = 3.0 \quad (4.0)$$

$$\frac{P(\text{Viagra} = 1|Y = \text{spam})}{P(\text{Viagra} = 1|Y = \text{ham})}\frac{P(\text{lottery} = 1|Y = \text{spam})}{P(\text{lottery} = 1|Y = \text{ham})} = \frac{0.40}{0.12}\frac{0.21}{0.13} = 5.4 \quad (0.67)$$

We see that, using a maximum likelihood decision rule, our very simple model arrives at the *Bayes-optimal* prediction in the first three cases, but not in the fourth ('Viagra' and 'lottery' both present), where the marginal likelihoods are actually very misleading.

Linear discriminants

Many forms of linear discriminant from statistics and machine learning, e.g.,

- Fisher's Linear Discriminant Analysis
  - the basic linear classifier in last lecture is a version of this
- Logistic Regression
  - a probabilistic linear classifier
- Perceptron
  - a linear threshold classifier
  - an early version of an artificial "neuron"
  - still a useful method, and source of ideas (later lecture)

Logistic regression

In the case of a two-class problem, model the probability of one class $P(Y = 1)$ vs. the alternative $(1 - P(Y = 1))$:

$$P(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

or

$$\ln \frac{P(Y = 1|\mathbf{x})}{1 - P(Y = 1|\mathbf{x})} = \mathbf{w}^T\mathbf{x}$$

The quantity on the l.h.s. is called the *logit* and all we are doing is a linear model for the logit.

Note: to fit this is actually more complex than linear regression, so we omit the details.

Generalises to multiple class versions (Y can have more than two values).

Summary

- We described the classification problem in machine learning
- We also outlined the issue of Inductive Bias
- Two major frameworks for classification were covered

  Distance-based. The key ideas are geometric.
  Probabilistic. The key ideas are Bayesian.
- We also discussed Logistic Regression
- So we have established the basis for learning classifiers
- Later we will see how to extend by building on these ideas