Never Stand Still

Classification (2)

COMP9417 Machine Learning & Data Mining

Aims

This lecture will continue your exposure to machine learning approaches to the problem of classification. Following it you should be able to:

- 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
- outline the Naive Bayes classification algorithm
- outline the logistic regression classification algorithm

Introduction

What do we understand about the problem of learning classifiers ? . . .

how can we know when classifier learning succeeds ?

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

Inductive Bias

"All models are wrong, but some models are useful."

Box & Draper (1987)

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 ?
    - 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 is 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

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 $P\,(y = spam|Viagra, lottery)$, 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

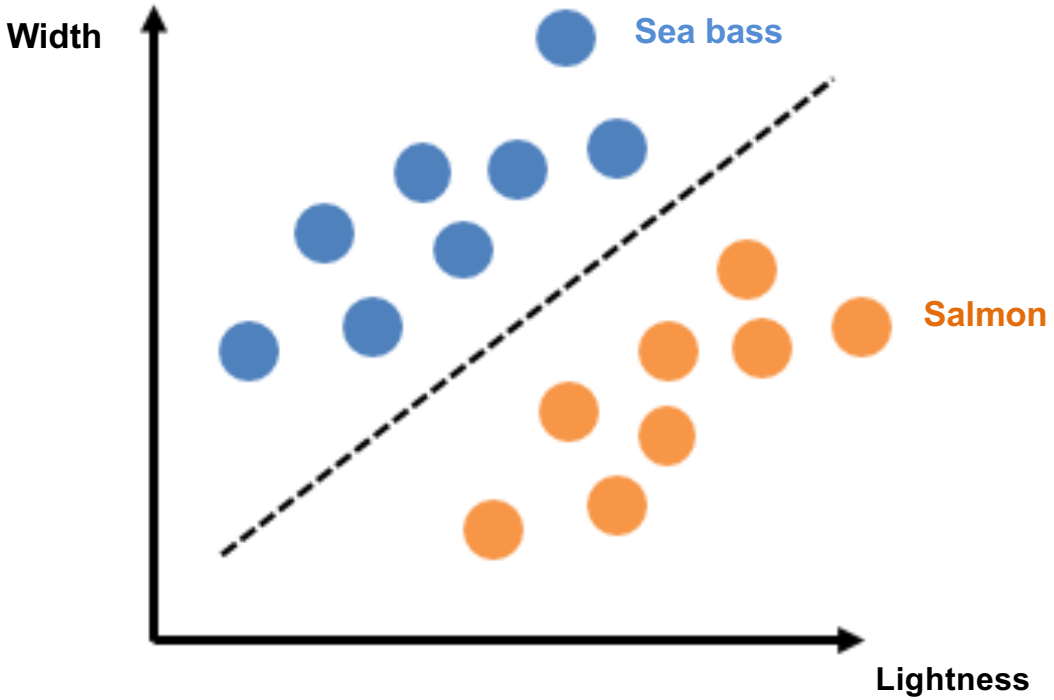
Provides useful conceptual framework:

- Provides a “gold standard” for evaluating other learning algorithms
- Some additional insight into Occam’s razor

Classification

Question:

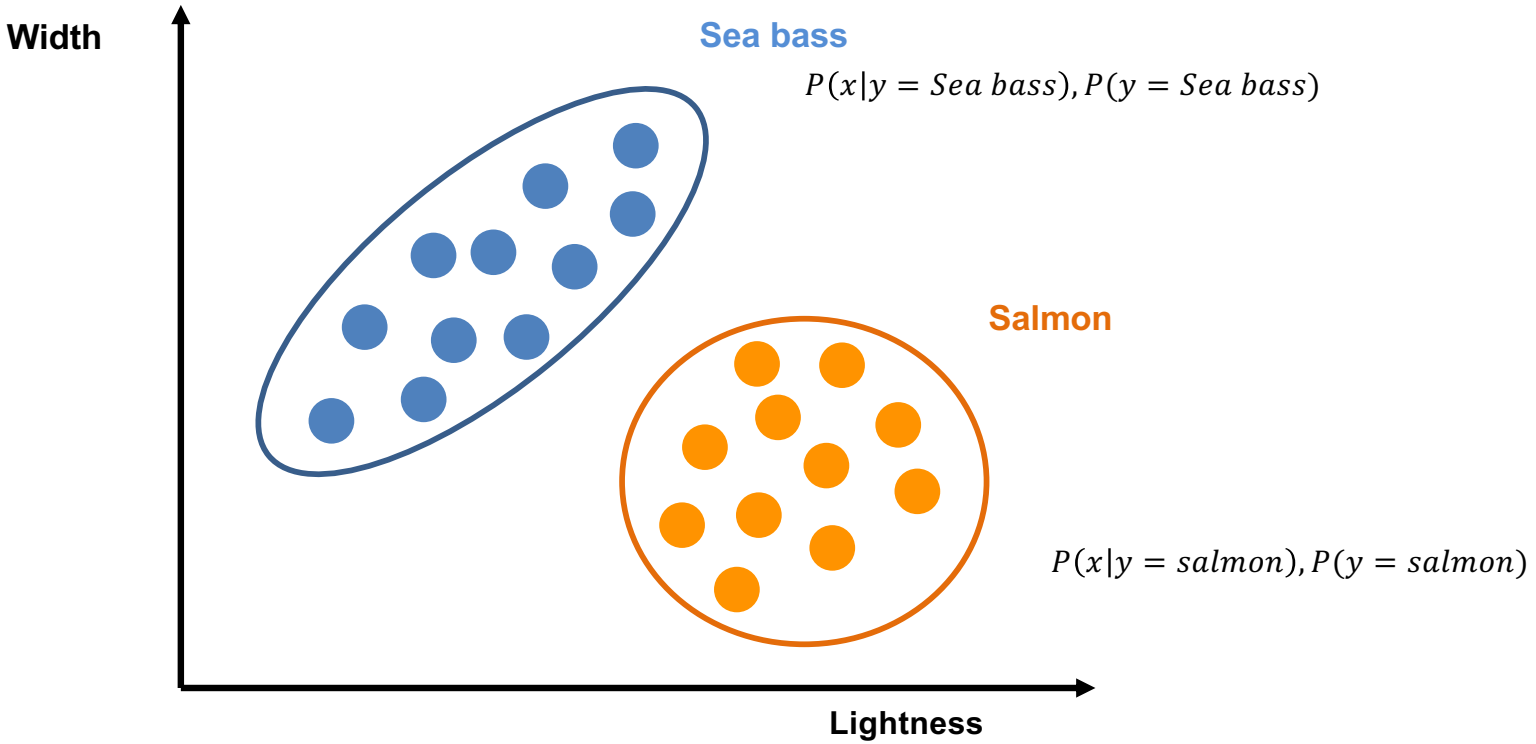
Can we do something different than finding the discriminative line (or some boundary) to be able to separate the two groups?



Classification

Answer:

Yes, we can. Instead of finding a discriminative line, maybe we can focus on one class at a time and build a model that describes how that class looks like; and then do the same for the other class. This type of models are called *generative learning algorithm*.



Classification – Bayesian methods

Example: Imagine, we want to classify fish type: Salmon , Sea bass

If from the past experience we have (C_i *is the class*):

P(c_i)	Salmon	Sea bass
Prior	0.3	0.7

- If we decide only based on prior, we always have to choose "sea bass". This is called "*decision rule based on prior*"
  - This can behave vey poorly
  - It never predicts other classes

Classification – Bayesian methods

Example: now if we have some more information on the length of the fish in each class, then how can we update our decision, if we want to predict the class for a fish with 70cm length?

These are called “class conditionals”, “class conditioned probabilities”

$P(x\|c_i)$	Salmon	Sea bass
length > 100 cm	0.5	0.3
50 cm < length < 100 cm	0.4	0.5
length < 50 cm	0.1	0.2

What we are interested in is $P(c_i|x)$, but we have $P(c_i)$ and $P(x|c_i)$, so how should we go about this?

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

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

Decision Rule from Posteriors

Example:

If the output belongs to a set of k classes: $y \in \{C_1, C_2, \ldots, C_k\}$ for $1 \leq i \leq k$

Then in Bayesian framework:

$$P(y = C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

- $P(y = C_i|x)$: posterior probability
- $P(x|C_i)$: class conditional (class likelihood)
- $P(C_i)$: prior
- $P(x)$: Marginal ($P(x) = \sum_i p(x|C_i)P(C_i)$)

The decision rule is to select a class which maximizes the posterior probability for the prediction

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} :

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

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

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

Choosing Hypotheses

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

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

Classification – Bayesian methods

Example: now if we have some more information on the length of the fish in each class, then how can we update our decision, if we want to predict the class for a fish with 70cm length?

$P(x\|c_i)$	Salmon	Sea bass
length > 100 cm	0.5	0.3
50 cm < length < 100 cm	0.4	0.5
length < 50 cm	0.1	0.2

$$P(c = salmon|x = 70cm) \propto P(70cm|salmon) * P(salmon) = 0.4*0.3=0.12$$
$$P(c = sea\ bass|x = 70cm) \propto P(70cm|sea\ bass) * P(sea\ bass) = 0.5*0.7=0.35$$

So base on these probabilities, our model predict the type as “sea bass”

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.

$P(cancer) = ?$ $\quad P(not\ cancer) = ?$

$P(\oplus \mid cancer) = ?$ $\quad P(\ominus \mid cancer) = ?$

$P(\oplus \mid not\ cancer) = ?$ $\quad P(\ominus \mid not\ cancer) = ?$

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.

$P(cancer) = .008$ $\quad P(not\ cancer) = .992$

$P(\oplus\ |cancer) = .98$ $\quad P(\ominus\ |cancer) = 0.02$

$P(\oplus\ |not\ cancer) = .03$ $\quad P(\ominus\ |not\ cancer) = .97$

Applying Bayes Theorem

Does patient have cancer or not?

$$P(cancer|\oplus) = ?$$

$$P(not\ cancer|\oplus) = ?$$

We can find the maximum a posteriori (MAP) hypothesis

$$P(\oplus|cancer)P(cancer) = 0.98 \times 0.008 = 0.00784$$

$$P(\oplus|not\ cancer)P(not\ cancer) = 0.03 \times 0.992 = 0.02976$$

Thus $h_{MAP} = \cdots$

Applying Bayes Theorem

Does patient have cancer or not?

$$P(cancer|\oplus) = ?$$

$$P(not\ cancer|\oplus) = ?$$

We can find the maximum a posteriori (MAP) hypothesis

$$P(\oplus|cancer)P(cancer) = 0.98 \times 0.008 = 0.00784$$

$$P(\oplus|not\ cancer)P(not\ cancer) = 0.03 \times 0.992 = 0.02976$$

Thus $h_{MAP} = not\ cancer$

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).

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:

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

- Rearrange sum rule to get:

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

Posterior Probabilities & Decision

If we have two competing hypothesis h_1 and h_2, then

- $P(h_1|D) =$ posterior probability of h_1
- $P(h_2|D) =$ posterior probability of h_2

- So far, the potential decision is made based on the higher posterior probability
  - Decide h_1 if $P(h_1|D) > P(h_2|D)$ and else decide h_2
- An alternative: using a loss function $L(h)$, where $L(h)$ is the loss that occurs when decision h is made.
  - In this setup the Bayesian testing procedure minimizes the posterior expected loss

Bayesian Expected Loss

Example: If in the example of patient with positive test result, we know that the cost of misclassifying a patient who has cancer as “not cancer” is 10 times more than misclassifying a patient who doesn’t have cancer as “cancer”, how that will affect our decision?

Bayesian Expected Loss (Risk)

If the cost of misclassification is not the same for different classes, then instead of maximizing a posteriori, we have to minimize the expected loss:

- So, if we define the loss associated to action α_i as $\lambda(\alpha_i|h)$
- Then the expected loss associated to action α_i is:

$$E[L(\alpha_i)] = R(\alpha_i|x) = \sum_{h \in H} \lambda(\alpha_i|h)\ P(h|x)$$

An optimal Bayesian decision strategy is to minimize the expected loss. And if the loss associated to misclassification is the same for different classes, then maximum a posteriori is equal to minimizing the expected loss.

Bayesian Expected Loss

Example: let's revisit the example of patient with positive result for cancer, given the loss function below:

$\lambda(\alpha_i \mid c_i)$	Cancer	Not cancer
If predicted cancer	0	1
If predicted not cancer	10	0

$$R(predict\ cancer|\oplus) = \lambda(predict\ cancer|cancer)P(cancer|\oplus)$$
$$+\lambda(predict\ cancer|not\ cancer)P(not\ cancer|\oplus) \propto 0 + 1{\times}0.02976 = 0.02976$$

$$R(predict\ not\ cancer|\oplus) = \lambda(predict\ not\ cancer|cancer)P(cancer|\oplus)$$
$$+\lambda(predict\ not\ cancer|not\ cancer)P(not\ cancer|\oplus) \propto 10{\times}0.00784 + 0 = 0.0784$$

$$R(predict\ cancer|\oplus) < R(predict\ not\ cancer|\oplus)$$

Therefor the expected loss is less if we predict that the patient has cancer.

Bayesian Expected Loss

In summary:

- Bayesian framework allows for integration of losses into the decision-making rule
- In such framework, we can minimize the posterior expected loss

Bayes Theorem

- To compute $P(D|h)$ and $P(h)$, we can use an empirical method based on the given data
- Or we may assume a parametric model, then we estimate parameters using the data

Learning A Real Valued Function

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

- $y_i = \hat{f}(x_i) + \varepsilon_i$
- ε_i is random variable (noise) drawn independently for each x_i according to some Gaussian (normal) distribution with mean zero

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

$$h_{ML} = \arg\max_{h \in H} P(D|h) = \arg\max_{h \in H} \prod_{i=1}^{m} P(y_i|\hat{f})$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_i - \hat{f}(x_i)}{\sigma}\right)^2}$$

Where $\hat{f} = h_{ML}$

Learning A Real Valued Function

Maximize natural log to give simpler expression:

$$h_{ML} = \arg\max_{h \in H} \sum_{i=1}^{m} ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\left(\frac{y_i - \hat{f}(x_i)}{\sigma}\right)^2$$

$$= \arg\max_{h \in H} \sum_{i=1}^{m} -\frac{1}{2}\left(\frac{y_i - \hat{f}(x_i)}{\sigma}\right)^2$$

$$= \arg\max_{h \in H} \sum_{i=1}^{m} -(y_i - \hat{f}(x_i))^2$$

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

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

Discriminative vs Generative Probabilistic Models

- Discriminative models model the posterior probability distribution $P\ (y|x)$, where y is the target variable and x is the feature vector. 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 $P(x) = \sum_c P(y = c, x)$ it follows that the posterior distribution can be obtained as $P(y|x) = \frac{P(y,x)}{\sum_c P(y=c,x)}$
- 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.

Bayesian Approach

What should h be?

- A collection of possible predictions, or
- A collection of functions that map the data x to a possible prediction y

Most Probable Classification of New Instances

So far, we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})

But the most important question is: given new instance x, what is its most probable *classification*?

- Although it may seem that h_{MAP} can answer this question, but in fact we can do better
- $h_{MAP}(x)$ is not necessarily the most probable *classification*! (e.g if we are dealing with multiple hypotheses supporting each class)

Most Probable Classification of New Instances

Consider:

- Three possible hypotheses:

$$P(h_1|D) = 0.4, \qquad P(h_2|D) = 0.3, \qquad P(h_3|D) = 0.3$$

- Given new instance x,

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

- What's most probable classification of x?

Bayes Optimal Classifier

Bayes optimal classification:

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

Example:

$$P(h_1|D) = 0.4, \quad P(-|h_1) = 0, \quad \mathrm{P}(+|h_1) = 1$$

$$P(h_2|D) = 0.3, \quad P(-|h_2) = 1, \quad \mathrm{P}(+|h_2) = 0$$

$$P(h_3|D) = 0.3, \quad P(-|h_3) = 1, \quad \mathrm{P}(+|h_3) = 0$$

Bayes Optimal Classifier

therefore

$$\sum_{h_i \in H} P(+|\, h_i) \mathrm{P}(h_i|D) = 0.4$$

$$\sum_{h_i \in H} P(-|\, h_i) \mathrm{P}(h_i|D) = 0.6$$

and

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

v_j is the class value from the set of possible class values ($v_j \in V$)

Bayes Optimal Classifier

The Bayes optimal classifier is a probabilistic model that makes the most probable prediction for a new sample based on the training data. This is different than MAP inference.

- In MAP framework, we seek the most probable hypothesis, among the space of hypothesis.
- But in Bayesian optimal classification, the most probable classification of the new instance is obtained by combining the predictions of ALL hypotheses (in the hypothesis space), weighted by their posterior probabilities:

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

The optimal classification of the new instance is the value v_j, for which $P(v_j|D)$ is maximum

Bayes Optimal Classifier

It can be mathematically shown that no other classification method using the same hypothesis space and same prior knowledge can outperform the Bayes Optimal Classifier method on average.

Bayes Optimal Classifier

Why do we need any other methods?

Bayes Optimal Classifier

The Bayes rule depends on unknown quantities, so we need to use the data to find some approximation of those quantities.

Gibbs Classifier

Bayes optimal classifier is very inefficient.

An alternative, less optimal method is Gibbs algorithm.

Gibbs algorithm:

1. Choose one hypothesis at random, according to $P(h|D)$
2. Use this to classify new instance

Surprising fact: Assuming target concepts are drawn at random from H $(h \in H)$ according to priors on H. Then

$$E[error_{Gibbs}] \leq 2 \times E[error_{Bayes\ Optimal}]$$

Bayes Error

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

Define the probability of error for classifying some instance x by

$$P(error|x) = P(class_1|x) \; if \; we \; predict \; class_2$$
$$= P(class_2|x) \; if \; we \; predict \; class_1$$

This gives

$$\sum P(error) = \sum_x P(error|x)P(x)$$

So we can justify the use of the decision rule:

if $\hat{P}(class_1|x) > \hat{P}(class_2|x)$ then predict $class_1$

else predict $class_2$

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 $f : X \rightarrow V$, where each instance x described by attributes $\langle x_1, x_2, \ldots, x_n \rangle$.

Most probable value of $f(x)$ is:

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j | x_1, x_2, \ldots, x_n)$$

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(x_1, x_2, \ldots, x_n | v_j) P(v_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \arg\max_{v_j \in V} P(x_1, x_2, \ldots, x_n | v_j) \mathrm{P}(v_j)$$

Naive Bayes Classifier

Naive Bayes assumption:

$$P(x_1, x_2, \ldots, x_n | v_j) = \prod_i P(x_i | v_j)$$

- 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:**

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

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 x_i:

$\hat{P}(x_i|v_j) \leftarrow$ estimate $P(x_i|v_j)$

Classify New Instance (for sample x)

$$v_{NB} = arg \max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(x_i|v_j)$$

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:

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

Naive Bayes Example: *PlayTennis*

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

For “yes” $= P(\text{“}yes\text{”}) \times P(sun|\text{”}yes\text{”}) \times P(cool|\text{”}yes\text{”}) P(high|\text{”}yes\text{”}) \times P(true|\text{”}yes\text{”})$

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

For “no” $= P(\text{“}no\text{”}) \times P(sun|\text{”}no\text{”}) \times P(cool|\text{”}no\text{”}) P(high|\text{”}no\text{”}) \times P(true|\text{”}no\text{”})$

$$0.0206 = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{9} \times \frac{3}{9}$$

Naive Bayes Example: PlayTennis

Then convert to a probability by normalisation

$$P(\text{“}yes\text{”}|x) = \frac{0.0053}{(0.0053 + 0.0206)} = 0.205$$

$$P(\text{“}no\text{”}|x) = \frac{0.0206}{(0.0053 + 0.0206)} = 0.795$$

The Naive Bayes classification is “no”.

Naive Bayes: Subtleties

Conditional independence assumption is often violated

$$P(x_1, x_2, \ldots, x_n | v_j) = \prod_i P(x_i | v_j)$$

– …but it works surprisingly well anyway. Note that you don't need the estimated posteriors $\hat{P}(v_j|x)$ to be correct; You need only that

$$arg \max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(x_i|v_j) = \arg \max_{v_j \in V} P(v_j) P(x_1, x_2, \ldots, x_n | v_j)$$

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 x_i? Then

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

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

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

(In some cases adding a constant different from 1 might be more appropriate)

Naive Bayes: numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)

$$x|v_j \sim N(\mu, \sigma^2)$$

- 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:

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

These parameters have to be defined for each class separately.

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 $standard\ deviation = 6.2$ for class of "yes". Density value

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

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

Categorical random variables

Categorical variables or features (also called discrete or nominal) are ubiquitous in machine learning. For example, in text classification.

- Perhaps the most common form is 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.

Learning to Classify Text

For example:

- 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

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

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

one more assumption:

$$P(x_i = w_k|v_j) = P(x_m = w_k|v_j), \forall i, m$$

"bag of words"

Application: 20 Newsgroups

Given: 20000 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	

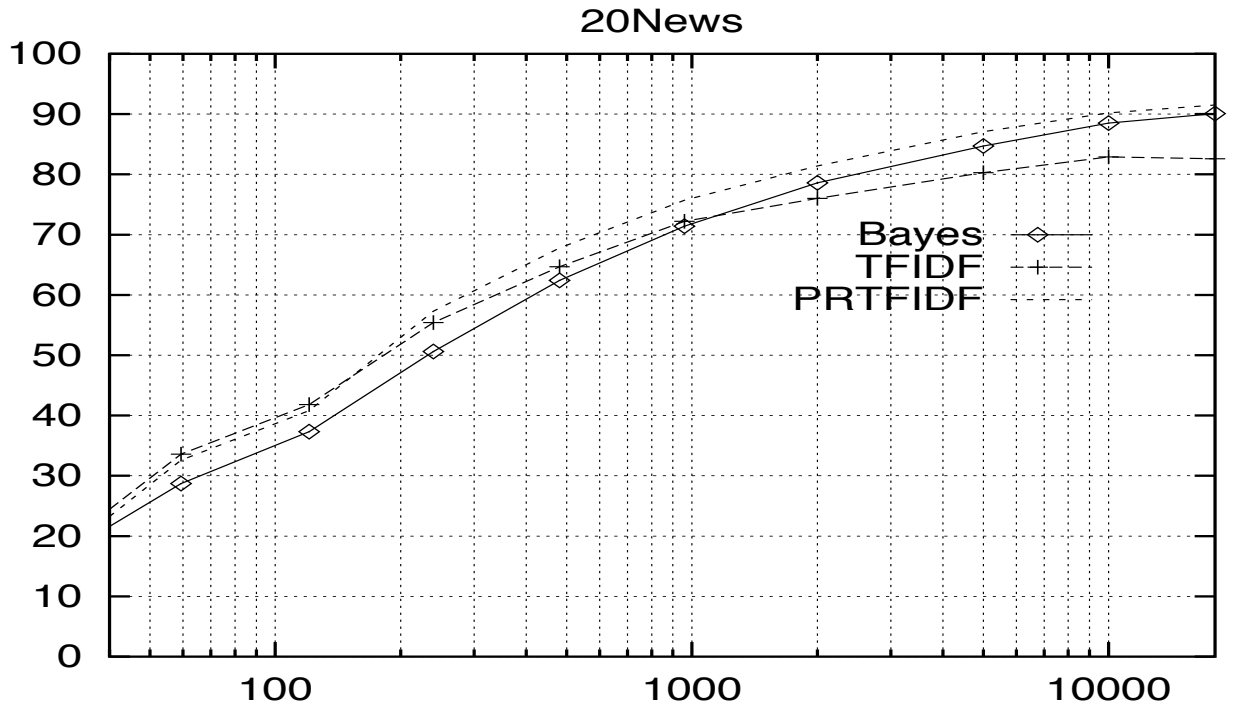
Naive Bayes: 89% classification accuracy

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

Training a naive Bayes model

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

$e1$: $b\ d\ e\ b\ b\ d\ e$	$e5$: $a\ b\ a\ b\ a\ b\ a\ e\ d$
$e2$: $b\ c\ e\ b\ b\ d\ d\ e\ c\ c$	$e6$: $a\ c\ a\ c\ a\ c\ a\ e\ d$
$e3$: $a\ d\ a\ d\ e\ a\ e\ e$	$e7$: $e\ a\ e\ d\ a\ e\ a$
$e4$: $b\ a\ d\ b\ e\ d\ a\ b$	$e8$: $d\ e\ d\ e\ d$

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 data for naive Bayes

The same data set described by bit vectors (presence of words).
We can use Bernoulli model.

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

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

$$\hat{\theta}^{\oplus} = \left(\frac{3}{6}, \frac{4}{6}, \frac{2}{6}\right) = (0.5, 0.67, 0.33) \quad \text{for spam}$$

$$\hat{\theta}^{\ominus} = \left(\frac{4}{6}, \frac{2}{6}, \frac{2}{6}\right) = (0.67, 0.33, 0.33) \quad \text{for ham}$$

Training data for naive Bayes

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

We can use multinomial model.

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 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

$$\hat{\theta}^{\oplus} = \left(\frac{6}{20}, \frac{10}{20}, \frac{4}{20}\right) = (0.3, 0.5, 0.2) \quad \text{for spam}$$

$$\hat{\theta}^{\ominus} = \left(\frac{12}{20}, \frac{4}{20}, \frac{4}{20}\right) = (0.6, 0.2, 0.2) \quad \text{for ham}$$

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

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

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 $x = (1, 1, 0)$. We obtain likelihoods

$$P(x|\oplus) = 0.50 \times 0.67 \times (1 - 0.33) = 0.222$$

$$P(x|\ominus) = 0.67 \times 0.33 \times (1 - 0.33) = 0.148$$

The ML classification of 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

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

- This means that the MAP classification of 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 $\frac{P(\oplus)}{P(\ominus)} = \frac{0.33}{0.67} = \frac{1}{2}$, resulting in a posterior odds of

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

In this case the likelihood ratio for 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

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

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 $x = (3,1,0)$. The total number of vocabulary word occurrences is $n = 4$. We obtain likelihoods:

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

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

The likelihood ratio is $(\frac{0.3}{0.6})^3(\frac{0.5}{0.2})^1(\frac{0.2}{0.2})^0 = \frac{5}{16}$. The ML classification of 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.

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

Let's use the spam-ham data in slide 9. 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 slide 9) 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):

$$P(y|lottery) = P(y|Viagra = 0, lottery)P(Viagra = 0) \\ +P(y|Viagra = 1, lottery)P(Viagra = 1)$$

Missing values

For instance, suppose for the sake of argument that one in ten e-mails contain the word 'Viagra', then $P(Viagra = 1) = 0.10$ and $P(Viagra = 0) = 0.90$. Using the above formula, we obtain

$$P(y = spam|lottery = 1) = 0.65\times0.90 + 0.40\times0.10 = 0.625$$

and

$$P(y = ham|lottery = 1) = 0.35\times0.90 + 0.60\times0.10 = 0.375$$

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 $P(x|y = spam) = 3.5\times10^{-5}$ and $P(x|y = ham) = 7.4\times10^{-6}$, 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

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

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

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

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

Using a MAP decision rule we predict ham in the top two cases and spam in the bottom two.

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 for Naïve Bayes as follows (the previously calculated odds from the full posterior distribution are shown in brackets):

$$\frac{P(Viagra = 0|y = spam)}{P(Viagra = 0|y = ham)}\frac{P(lottory = 0|y = spam)}{P(Lottery = 0|y = ham)} = \frac{0.60}{0.88}\frac{0.79}{0.87} = 0.62 \qquad (0.45)$$

$$\frac{P(Viagra = 0|y = spam)}{P(Viagra = 0|y = ham)}\frac{P(lottory = 1|y = spam)}{P(Lottery = 1|y = ham)} = \frac{0.60}{0.88}\frac{0.21}{0.13} = 1.1 \qquad (1.9)$$

$$\frac{P(Viagra = 1|y = spam)}{P(Viagra = 1|y = ham)}\frac{P(lottory = 0|y = spam)}{P(Lottery = 0|y = ham)} = \frac{0.40}{0.12}\frac{0.79}{0.87} = 3.0 \qquad (4.0)$$

$$\frac{P(Viagra = 1|y = spam)}{P(Viagra = 1|y = ham)}\frac{P(lottory = 1|y = spam)}{P(Lottery = 1|y = ham)} = \frac{0.40}{0.12}\frac{0.21}{0.13} = 5.4 \qquad (0.67)$$

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

Naive Bayes

Pros:

- Easy and fast at prediction. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression with less training data.
- Performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

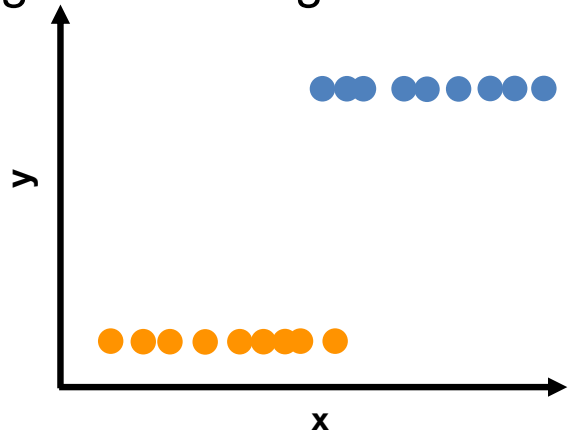
- If zero-frequency case happens, it is unable to make a prediction. (solution: use the smoothing technique).
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs can not to be taken too seriously.
- Strong assumption of independent attributes. In real life, it is almost impossible that we get a set of attributes which are completely independent.

Logistic Regression

a probabilistic linear classifier

Logistic Regression

- In a binary classification problem with one input variable, if we show the output on y-axis we may get something like below:



- Can we use a linear regression to model this data?

Logistic Regression

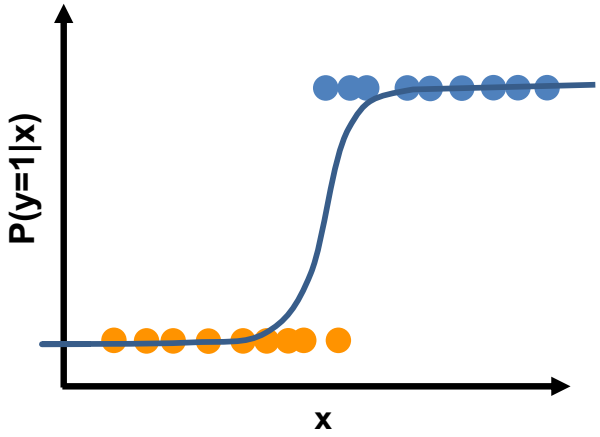
Why regular regression doesn't work here:

- Univariate or multivariate regressions are to predict a real-valued output from one or more independent variables
- Binary data doesn't have a normal distribution which is a condition needed for most other types of regression
- What we expect to get is an output of 0 or 1, but linear regression obviously can produce values beyond that range.

Logistic Regression

- In binary classification, we can transform the y values into probability values (vales are in range [0,1])

- We can model this with a s-curve (sigmoid curve) as above:



$$P(y=1|x) = \frac{1}{1+e^{-f(x)}} \Rightarrow f(x) = log\frac{P(y=1|x)}{1-P(y=1|x)}$$

Now $f(x)$ can have a value between $-\infty$ and $+\infty$ and in Logistic Regression we estimate $f(x)$ with a line.

$$\hat{f}(x) = x^T\beta \Rightarrow log\frac{P(y=1|x)}{1-P(y=1|x)} = x^T\beta$$

Logistic Regression

Logistic regression seeks to:

- Model the probability of a class given the values of independent input variables
- Estimate the probability that a class occurs for a random observation (versus the probability that the class doesn't occur)
- Classify an observation based on the probability estimations

Logistic Regression

$$\hat{P}(y = 1|x) = \frac{1}{1 + e^{-x^T\beta}}$$

If $P(y = 1|x) \geq 0.5$ (same as saying $x^T\beta \geq 0$) then predict as class 1

If $P(y = 1|x) < 0.5$ (same as saying $x^T\beta < 0$) then predict as class 0

- Interpretation of this model in the input space (feature space) is equivalent of having a linear decision boundary ($x^T\beta = 0$) separating the two class

- Now we have a linear solution to our problem, and this is what makes *Logistic Regression* a linear model.

Logistic Regression Parameter Estimation

- We can not use a similar cost function as we used in linear regression here, because it will result a non-convex function with many local minimums and would be very difficult to find the global minimum.
- Instead, the following cost function is used:

Let's define $\hat{P}(y = 1|x) = h_\beta\ (x)$

$$cost\big(h_\beta(x), y\big) = \begin{cases} -\log\big(h_\beta(x)\ \big) & if\ y = 1 \\ -\log\Big(1 - h_\beta\ (x)\Big) & if\ y = 0 \end{cases}$$

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\Big(h_\beta\big(x^{(i)}\big)\Big) + \big(1 - y^{(i)}\big)\log(1 - h_\beta\big(x^{(i)}\big))]$$

Now, to find the values of parameters to minimize $J(\beta)$, we can use the Gradient Descent algorithm.

Logistic Regression

Pros:

- Relatively easy to implement
- Easy to interpret
- Relatively fast at training and very fast at testing
- Can easily extend to multi-classes
- Provide probabilistic predictions

Cons:

- Prone to overfitting in high-dimensional data (one remedy: regularization)
- It provides a linear decision boundary. For non-linear decision boundaries, feature transformation is required
- Requires moderate or no correlation (collinearity) between input variables and may lead to poor model (dimensionality reduction is useful)
- Sensitive to outlier

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

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)