Classification (2)

COMP9417 Machine Learning and Data Mining

Term 2, 2021

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:

- outline the logistic regression classification algorithm
- describe the use of Bayes Theorem for decision-making under uncertainty
- 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

Linear classifiers

The "classic" classification learning algorithm is a *linear discriminant* — many forms of linear discriminant from statistics and machine learning, e.g.,

- Fisher's Linear Discriminant Analysis
  - basic linear classifier (last lecture) is a simplified version of this
- Logistic Regression
  - a probabilistic linear classifier (next slides)
- Perceptron
  - a linear threshold classifier (last lecture)
  - an early version of an artificial "neuron"
  - still a useful method, and source of ideas (later lecture)
- Winnow
  - like Perceptron, another linear threshold classifier
  - uses a different learning method, with interesting properties
  - comes from Learning Theory (later lecture)

Linear Regression for classification

Question: can we use Linear Regression for classification ?

Answer: not really, unless we use an approach like the following, e.g.,

Training: train a separate linear regression model for each class

- set $y = 1$ if example in class
- set $y = 0$ otherwise

Prediction: for each example

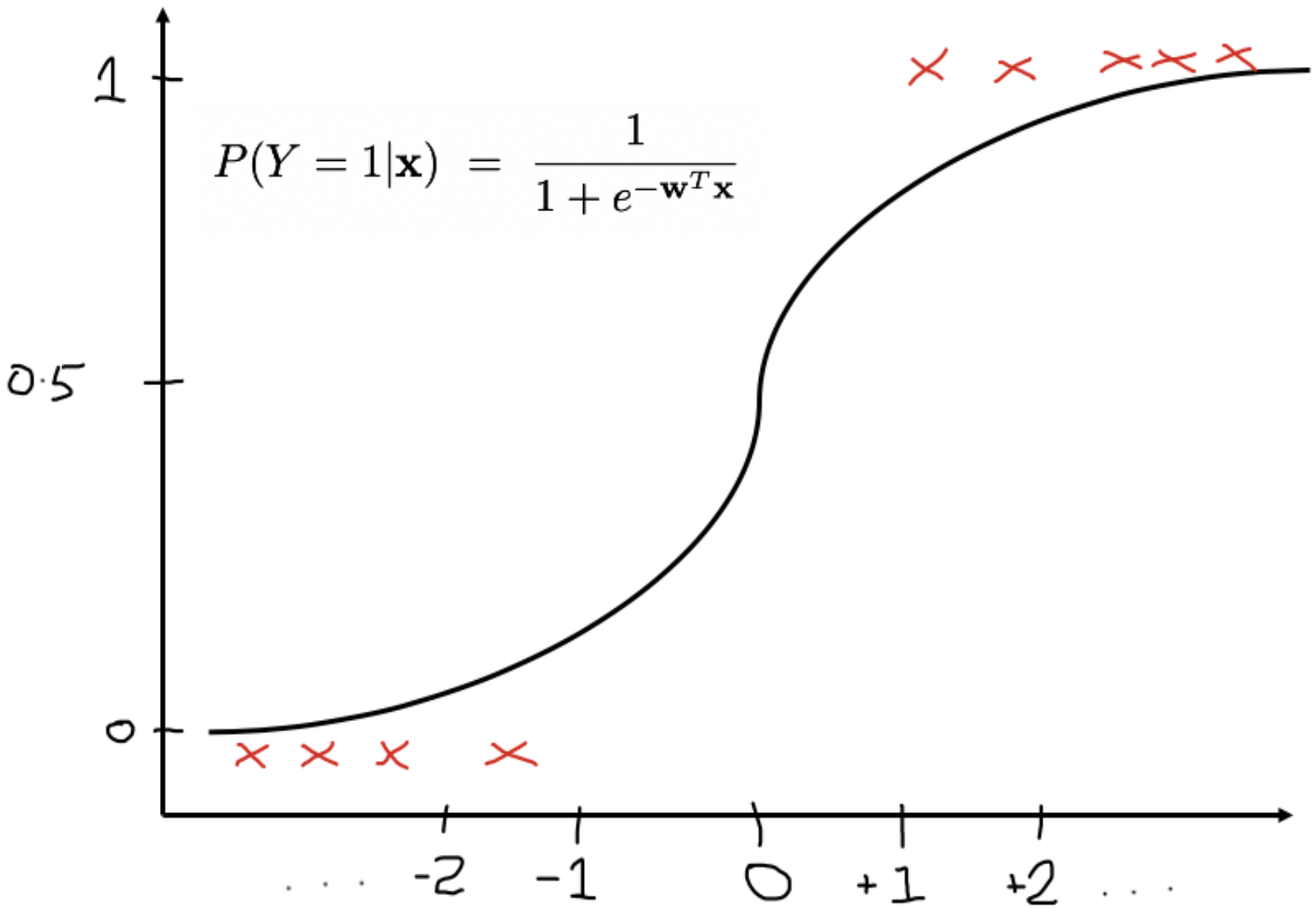
- run example through all regression models
- predict the class with the largest output value for y

Called *multi-response* linear regression.[1]

Note: does not obey linear regression assumptions, but may work in practice.

[1] See: Witten et al. (2017).

Logistic regression



Logistic regression

In the case of a two-class classification problem, if we model the probability $P(Y = 1)$ of an instance $\mathbf{x}$ being a positive example like this:

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

then this probability vs. the alternative $(1 - P(Y = 1))$ can be written like this:

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

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

Logistic regression

Unlike linear regression, no analytical maximum likelihood (ML) solution to find weights $\mathbf{w}$.

An iterative gradient ascent method can be used to maximize log likelihood.

The (conditional) log likelihood is:

$$\sum_{i=1}^{N} y^{(i)} \log P(1|\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - P(1|\mathbf{x}^{(i)}))$$

Over a set of N examples, choose $\mathbf{w}$ to maximise this expression, noting that $y^{(i)}$ is always either 0 or 1.

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

Step back

Classifier learning

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 [2].

[2] Even true for "deep learning"; watch Andrew Ng's talk on this at `http://www.youtube.com/watch?v=F1ka6a13S9I&t=48s`.

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

Bayesian Machine Learning

Two Roles for Bayesian Methods

Provides useful conceptual framework:

- a general framework for decision making under uncertainty
- a "gold standard" for evaluating other learning algorithms

Provides practical learning algorithms:

- Naive Bayes classifier learning
- Bayesian network learning, etc. (not in this course)
- combines prior knowledge (prior probabilities) with observed 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.

Bayes Theorem

For estimating the probability of a hypothesis (model) from data:

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

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

Choosing Hypotheses

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

Generally, we want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP}:

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

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 probability of data, $P(\oplus)$, 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.

A Bayesian framework for Classification

1. Define a prior on hypotheses
2. Define the likelihood, i.e., the probability of the data given the hypothesis
3. Learning is finding the required parameters by fitting hypotheses to data
4. Predict (classify) using, e.g., the MAP hypothesis

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)

Consider a basic form of classification, *concept learning*[3]

- two classes: positive examples are in the concept
  - e.g., pink elephants
- instances x_i described by Boolean features
  - e.g., $\text{has\_trunk}(x_{17})$, $\text{pink}(x_{17})$, ...
- hypotheses x_i described by predicates
  - e.g., $\text{has\_trunk}(X) \wedge \text{pink}(X)$, where X is a variable
- examples are instances labelled with class values
- assume perfect classification is possible
  - noise-free examples
- a hypothesis is *consistent* with a set of examples if
  - the hypothesis is true for all positive examples
  - the hypothesis is false for all negative examples

[3] See: Mitchell (1997).

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 ?

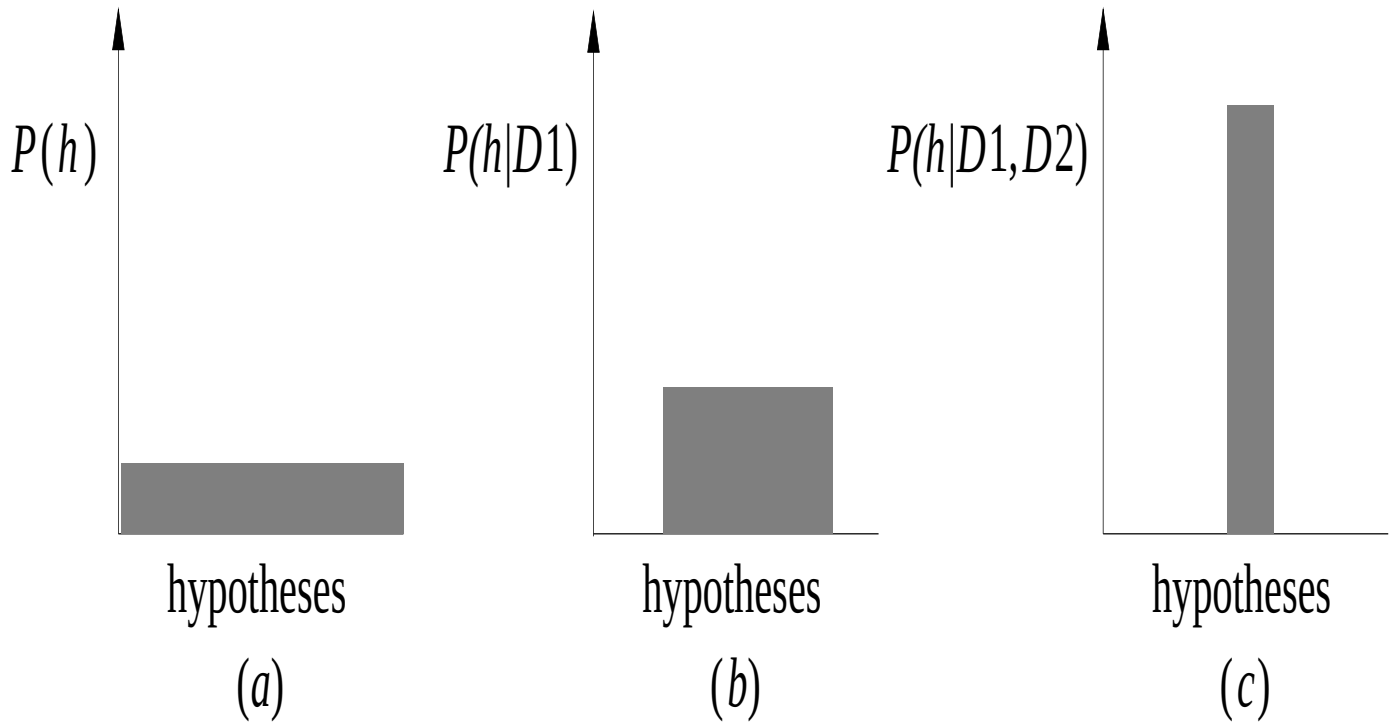
$$
\begin{aligned}
P(h|D) &= \frac{1 \cdot \frac{1}{|H|}}{P(D)} \\
&= \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} \\
&= \frac{1}{|VS_{H,D}|}
\end{aligned}
$$

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)

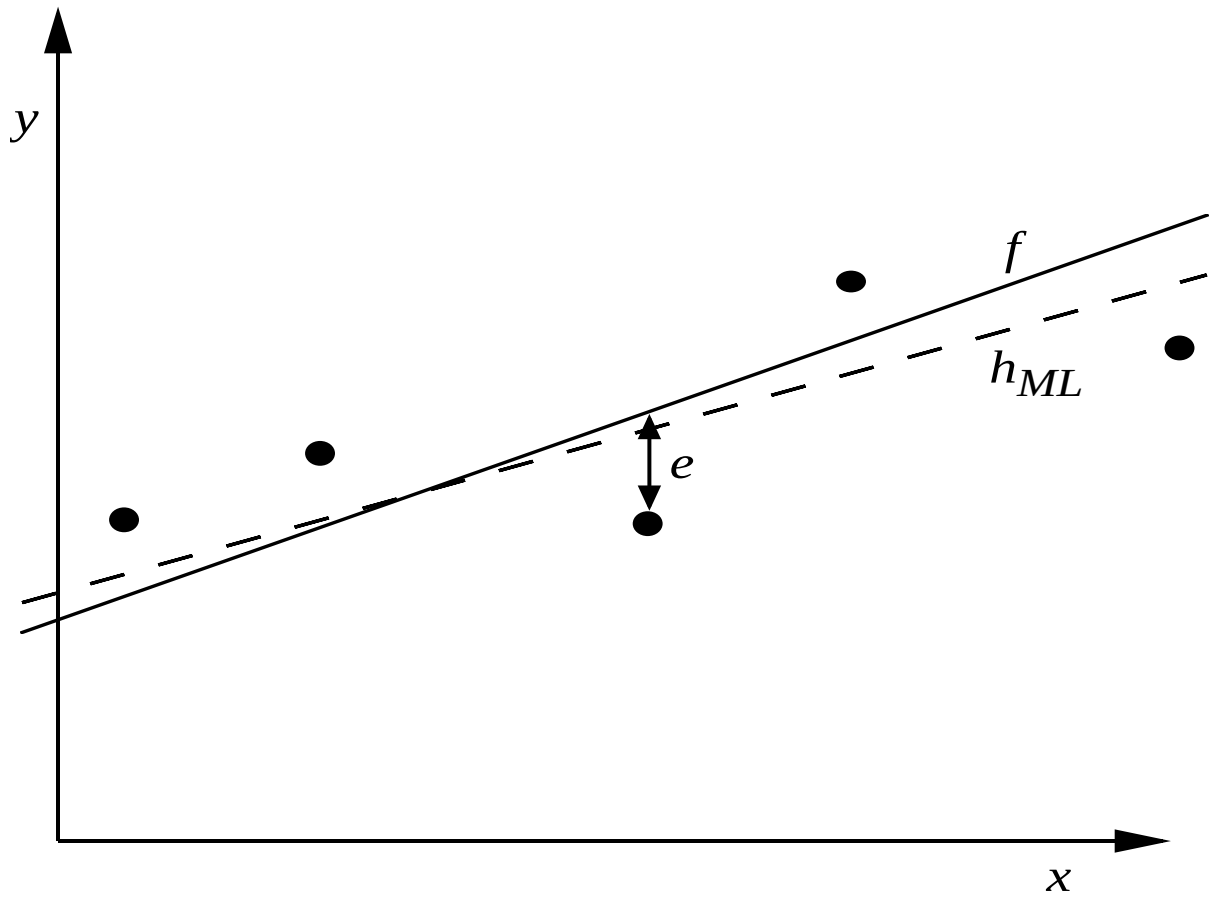
Every hypothesis consistent with D is a MAP hypothesis, if we assume

- 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 a 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:

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

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.

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

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

Key point: no other classification method using the same hypothesis space and same prior knowledge can outperform this method on average

Bayesian classifier learning in practice

Problem: Bayes optimal classification is typically intractable in practice

- requires executing *all* hypotheses
- not practical in applications

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.
- example: Logistic regression
- *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.
- example: Naive Bayes classifier

Naive Bayes Classifier

A simplified version of a generative probabilistic model for classification

- widely-used algorithm
- easy to train practical learning methods
- not the most accurate, but better than others for some applications
- estimated probabilities may be off, but key issue is classification

When to apply

- 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:

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

By application of Bayes Theorem.

Naive Bayes Classifier

Naive Bayes assumption:

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

- attributes a_i are statistically independent, given the class value v_j
  - this 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: $v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

Binomial and Bernoulli distributions[4]

"Coin tossing"

Let $X \in \{0, \ldots, n\}$ be number of heads in a sequence of n tosses.

If the probability of heads is θ, then X has a binomial distribution

$$\text{Bin}(k|n, \theta) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

For a single coin toss, let $X \in \{0, 1\}$. If θ is the probability of heads, then X has a Bernoulli distribution

$$\text{Ber}(x|\theta) = \theta^{\mathbb{I}(x=1)} (1-\theta)^{\mathbb{I}(x=0)}$$

[4] "Machine Learning: A Probabilistic Perspective", K. Murphy (2012)

Multinomial and Multinoulli distributions

"K-sided Dice throwing"

Let $\mathbf{x} = (x_1, \ldots, x_K)$ be a random vector, where x_j is the number of times side j of the die is thrown.

If θ_j is the probability of side j, then $\mathbf{x}$ has a multinomial distribution

$$\mathrm{Mu}(\mathbf{x}|n, \theta) = \binom{n}{x_1 \ldots x_K} \prod_{j=1}^{K} \theta_j^{x_j}$$

For a single dice throw, $\mathbf{x}$ will be a vector of zeros except for a single '1' at index j indicating that side j of the die is thrown. If θ_j is as above, then $\mathbf{x}$ has a multivariate Bernoulli (Multinoulli) distribution

$$\mathrm{Mu}(\mathbf{x}|1, \theta) = \prod_{j=1}^{K} \theta_j^{\mathbb{I}(x_j=1)}$$

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.

Probabilistic decision rules

We choose 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 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:

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 (features).

Training data for Naive Bayes

In the multivariate Bernoulli model e-mails are represented by bit vectors. Here is our email 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

- Adding the bit vectors for each class results in $(2, 3, 1)$ for spam and $(3, 1, 1)$ for ham.
- Each count is divided by the number of documents in a class, 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.

Training data for Naive Bayes

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

Here is our 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 a Naive Bayes model

- 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.

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.

Naive Bayes: numeric attributes

Note that Naive Bayes can be used with probabilistic models other than categorical variables, e.g., for 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.

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.
  We discussed distance-based classification (Nearest Neighbour)

  Probabilistic. The key ideas are (mostly) Bayesian.
  We discussed generative (Naive Bayes) and discriminative (Logistic Regression) models
- So we have established a basis for learning classifiers
- Later we will see how to extend by building on these ideas

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York.

Witten, I., Frank, E., Hall, M., and Pal, C. (2017). *Data Mining (Fourth Edition)*. Morgan Kaufmann.