Machine Learning

Part III

A Biswas
IIEST, Shibpur

Syllabus

Introduction

Learning Problems, Well-posed learning problems, Designing learning systems.

Concept Learning

Concept learning task, Inductive hypothesis, Ordering of Hypothesis, General-to-specific ordering of hypotheses. Version spaces, Inductive Bias.

Learning Rule Sets

Sequential Covering Algorithm, First Order Rules, Induction, First Order Resolution, Inverting Resolution.

Regression

Linear regression, Notion of cost function, Logistic Regression, Cost function for logistic regression, application of logistic regression to multi-class classification.

Continued …

Syllabus (continued)

Supervised Learning

Support Vector Machine, Decision tree Learning, Representation, Problems, Decision Tree Learning Algorithm, Attributes, Inductive Bias, Overfitting.

Bayes Theorem, Bayesian learning, Maximum likelihood, Least squared error hypothesis, Gradient Search, Naive Bayes classifier, Bayesian Network, Expectation Maximization Algorithm.

Unsupervised learning

Clustering, K-means clustering, hierarchical clustering.

Instance-Based Learning

k-Nearest Neighbour Algorithm, Radial Basis Function**, Locally Weighted Regression, Locally Weighted Function.**

Neural networks

Linear threshold units, Perceptrons, Multilayer networks and back-propagation, recurrent networks. Probabilistic Machine Learning, Maximum Likelihood Estimation.

Regularization, Preventing Overfitting, Ensemble Learning: Bagging and Boosting, Dimensionality reduction

Bayesian Learning

Bayesian reasoning provides a **probabilistic approach** to **inference.**

Bayesian Learning

The quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data.

Bayesian Learning

It is important to machine learning because it provides a quantitative approach to weighing the evidence supporting alternative hypotheses.

Bayesian Learning

Bayesian learning methods are relevant for two different reasons.

First, Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the **most practical approaches to certain types of learning problems.**

Bayesian Learning

The second reason that Bayesian methods are important to our study of machine learning is that they provide a **useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.**

Features of Bayesian learning methods

1. Each observed training example can incrementally **decrease or increase the estimated probability** that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.

Features of Bayesian learning methods

2. **Prior knowledge** can be **combined** with **observed data** to determine the final probability of a hypothesis.

In Bayesian learning, prior knowledge is provided by asserting

(1) a prior probability for each candidate hypothesis, and

(2) a probability distribution over observed data for each possible hypothesis

Features of Bayesian learning methods

3. Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").

Features of Bayesian learning methods

4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

Features of Bayesian learning methods

5. Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

Features of Bayesian learning methods

One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities.

When these probabilities are not known in advance they are often estimated based on **background knowledge, previously available data, and assumptions about the form of the underlying distributions**.

Features of Bayesian learning methods

A second practical difficulty is the **significant computational cost** required to determine the Bayes optimal hypothesis in the general case (**linear in the number of candidate hypotheses**).

Bayes Theorem

Objective: Determining the **best hypothesis** from some space H, given the observed training data D.

Bayes Theorem

What do we mean by the best hypothesis?

The **most probable** hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H.

Bayes Theorem

Bayes theorem provides a direct method for calculating such probabilities. More precisely, Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

Bayes Theorem

Notations:

P(h): initial probability that hypothesis h holds, before we have observed the training data.

Bayes Theorem

Notations:

P(h) is often called the **prior probability** of h and may reflect any background knowledge we have about the chance that h is a correct hypothesis.

If we have **no such prior knowledge**, then we might simply **assign the same prior probability** to each candidate hypothesis.

Bayes Theorem

Notations:

P(D): the **prior probability that training data** D will be observed (i.e., the probability of D given no knowledge about which hypothesis holds).

P(D|h): the probability of observing data D given some world in which hypothesis h holds.

Bayes Theorem

In machine learning problems, we are interested in the probability **P(h|D)** that h holds given the observed training data D .

P(h|D) is called the **posterior probability** of h because it reflects our confidence that h holds after we have seen the training data D.

Notice **the posterior probability P(h|D) reflects the influence of the training data D**, in contrast to the prior probability P(h), which is independent of D.

Bayes Theorem

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

Bayes Theorem

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

P(h|D) increases with P(h) and with P(D|h) according to Bayes theorem. It is also reasonable to see that P(h|D) decreases as P(D) increases, because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

Bayes Theorem

In many learning scenarios, the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis $h \in H$ given the observed data D (or at least one of the maximally probable if there are several). Any such maximally probable hypothesis is called a Maximum a Posteriori (MAP) hypothesis.

Bayes Theorem

We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.

Bayes Theorem

h_{MAP} is a MAP hypothesis provided

$$
\begin{aligned}
h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \\
&= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \\
&= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h)
\end{aligned}
$$

the term P (D) is dropped because it is a constant independent of h.

Bayes Theorem

If we assume, every hypothesis in H is equally probable a priori $P(h_i) = P(h_j) \;\; \forall \; h_i, h_j \; \in \; H.$

In this case, we need only to consider the term P(D|h) to find the most probable hypothesis.

P(D|h) is often called the likelyhood of the data D given h, and any hypothesis that maximizes P(D|h) is called a **maximum likelihood** (ML) hypothesis, h_{ML}.

$$h_{ML} \equiv argmax_{h \in H} \; P(D \mid h)$$

Bayes Theorem

Connection of Bayes theorem with machine learning:

The data D as training examples of some target function and referring to H as the space of candidate target functions.

Bayes Theorem

It can be applied equally well to **any set H of mutually exclusive propositions** whose probabilities sum to one (e.g., "the sky is blue," and "the sky is not blue").

Bayes Theorem

We will at times consider cases where H is a hypothesis space containing possible target functions and the data D are training examples.

At other times we will consider cases where H is some other set of mutually exclusive propositions, and D is some other kind of data.

Bayes Theorem

An Example:

Consider a medical diagnosis problem in which there are two alternative hypotheses:

(1) that the patient has a particular form of cancer and

(2) that the patient does not.

Available data is from laboratory with two possible outcomes positive ($\oplus$) and negative ($\ominus$).

Bayes Theorem

An Example: ...

We have prior knowledge that over the entire population of people only .008 have this disease. Furthermore, the lab test is only an imperfect indicator of the disease. 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. In other cases, the test returns the opposite result.

Bayes Theorem

An Example: ...

Summarised as:

$$P(cancer) = .008, \qquad P(\neg cancer) = .992$$

$$P(\oplus|cancer) = .98, \qquad P(\ominus|cancer) = .02$$

$$P(\oplus|\neg cancer) = .03, \qquad P(\ominus|\neg cancer) = .97$$

Suppose we now observe a new patient for whom the lab test returns a positive result. Should we diagnose the patient as having cancer or not?

Bayes Theorem

An Example: ...

The maximum a posteriori hypothesis can be found using Equation

$$
\begin{aligned}
h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \\
&= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \\
&= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h)
\end{aligned}
$$

$$
\begin{aligned}
P(\oplus|cancer)P(cancer) &= (.98).008 = .0078 \\
P(\oplus|\neg cancer)P(\neg cancer) &= (.03).992 = .0298
\end{aligned}
$$

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

Bayes Theorem

An Example: ...

The maximum a posteriori hypothesis can be found using Equation

$$
\begin{aligned}
h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \\
&= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \\
&= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h)
\end{aligned}
$$

$$
\begin{aligned}
P(\oplus|cancer)P(cancer) &= (.98).008 = .0078 \\
P(\oplus|\neg cancer)P(\neg cancer) &= (.03).992 = .0298
\end{aligned}
$$

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

Bayes Theorem

An Example: ...

The exact posterior probabilities can also be determined by normalizing the above quantities so that they sum to 1.

$$P(cancer \mid \oplus) = \frac{0.0078}{0.0078 + 0.0298} = 0.21$$

Bayes Theorem

An Example: ...

This step is warranted because Bayes theorem states that the posterior probabilities are just the above quantities divided by the probability of the data, $P(\oplus)$. Although $P(\oplus)$ was not provided directly as part of the problem statement, we can calculate it in this fashion because we know that $P(\text{cancer} \mid \oplus)$ and $P(\neg\ \text{cancer} \mid \oplus)$ must sum to 1 (i.e., either the patient has cancer or they do not).

Bayes Theorem

An Example: ...

Notice that while the posterior probability of cancer is significantly higher than its prior probability, the most probable hypothesis is still that the patient does not have cancer.

Bayes Theorem

An Example: ...

As this example illustrates, the result of Bayesian inference depends strongly on the prior probabilities, which must be available in order to apply the method directly.

Note also that in this example the hypotheses are not completely accepted or rejected, but rather become more or less probable as more data is observed.

Bayes Theorem

An Example: ...

- *Product rule*: probability $P(A \wedge B)$ of a 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 a disjunction of two events A and B

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

- *Bayes theorem*: the posterior probability $P(h|D)$ of h given D

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

- *Theorem of total probability*: if events $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

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

Summary of basic probability formulas

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Bayesian analysis can sometimes be used to show that a particular learning algorithm outputs MAP hypotheses even though it may not explicitly use Bayes rule or calculate probabilities in any form.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Now, we consider the problem of **learning a continuous-valued target function** - a problem faced by many learning approaches such as neural network learning, linear regression, and polynomial curve fitting.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

A straight forward Bayesian analysis will show that under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a maximum likelihood hypothesis.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

The significance of this result is that it provides a **Bayesian justification** (under certain assumptions) for many neural network and other curve fitting methods that attempt to minimize the sum of squared errors over the training data.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Consider the problem:

Learner L considers an instance space X and a hypothesis space H consisting of some class of real-valued functions defined over X.

That is, each h ∈ H is a function of the form

$$h : X \rightarrow \mathbb{R}$$

$\mathbb{R}$ is the set of real numbers.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

The problem faced by L is to learn an unknown target function $f : X \to \mathbb{R}$ drawn from H.

A set of **m** training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

More precisely, each training example is a pair of the form (x_i, d_i) where $d_i = f(x_i) + e_i$. Here $f(x_i)$ is the noise-free value of the target function and **ei** is a random variable representing the noise.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

It is assumed that the values of the e_i are drawn independently and that they are distributed according to a Normal distribution with zero mean.

The task of the learner is to output a maximum likelihood hypothesis, or, equivalently, a MAP hypothesis assuming all hypotheses are equally probable a priori.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

A simple example: learning a linear function (though our analysis applies to learning arbitrary real-valued functions)

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Learning a real-valued function.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Linear target function f depicted by the solid line, and a set of noisy training examples of this target function. The dashed line corresponds to the hypothesis h_{ML} with least-squared training error, hence the maximum likelihood hypothesis. Notice that the maximum likelihood hypothesis is not necessarily identical to the correct hypothesis, f, because it is inferred from only a limited sample of noisy training data.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

A hypothesis that minimizes the sum of squared errors in this setting is also a maximum likelihood hypothesis - why?

Review of two concepts:

1. Probability density
2. Normal Distribution

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Probability Density:

First, in order to discuss probabilities over continuous variables such as e, we must introduce probability densities. The reason, roughly, is that we wish for the total probability over all possible values of the random variable to sum to one. In the case of continuous variables we cannot achieve this by assigning a finite probability to each of the infinite set of possible values for the random variable.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Probability Density:

Instead, we speak of a probability density for continuous variables such as e and require that the integral of this probability density over all possible values be one.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Probability Density Function (p.d.f.)

The probability density $p(x_0)$ is the limit as ϵ goes to zero, of $1/\epsilon$ times the probability that x will take on a value in the interval $[x_0, x_0 + \epsilon)$.

$$p(x_0) \equiv \lim_{\epsilon \to 0} \frac{1}{\epsilon} P(x_0 \leq x < x_0 + \epsilon)$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Normal Distribution

Second, we stated that the random noise variable e is generated by a Normal probability distribution. A Normal distribution is a smooth, bell-shaped distribution that can be completely characterized by its mean μ and its standard deviation σ.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

A Normal distribution (also called a Gaussian distribution) is a bell-shaped distribution defined by the probability density function

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

A Normal distribution is fully determined by two parameters in the above formula: μ and σ.

If the random variable X follows a normal distribution, then:

- The probability that X will fall into the interval (a, b) is given by

$$\int_a^b p(x)dx$$

- The expected, or mean value of X, $E[X]$, is

$$E[X] = \mu$$

- The variance of X, $Var(X)$, is

$$Var(X) = \sigma^2$$

- The standard deviation of X, σ_X, is

$$\sigma_X = \sigma$$

The Central Limit Theorem (Section 5.4.1) states that the sum of a large number of independent, identically distributed random variables follows a distribution that is approximately Normal.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

To showing that the least-squared error hypothesis is, in fact, the maximum likelihood hypothesis within our problem setting.

By definition, Maximum Likely hypothesis using probability density p:

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}}\, p(D|h)$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Assume a fixed set of training instances $< x_1, \ldots, x_m >$.

Data D is the corresponding sequence of target values

$D = < d_1, \ldots, d_m >$ Here, $d_i = f(x_i) + e_i$

Assuming the training examples are mutually independent given h, we can write P (D | h) as the product of the various $p(d_i | h)$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i | h)$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Given that the noise e_i obeys a Normal distribution with zero mean and unknown variance σ^2 each d_i must also obey a Normal distribution with variance σ^2 centered around the true target value $f(x_i)$ rather than zero. Therefore, $p(d_i | h)$ can be written as a Normal distribution with variance σ^2 and mean $\mu = f(x_i)$.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Let us write the formula for this Normal distribution to describe $p(d_i|h)$, beginning with the general formula for a Normal distribution and substituting the appropriate μ and σ^2.

Because we are writing the expression for the probability of d_i given that h is the correct description of the target function f, we will also substitute μ = f (xi)= h(xi)

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Hence,

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - \mu)^2}$$

$$= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2}$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Now, apply a common transformation:

Rather than maximizing the above complicated expression we shall choose to maximize its (less complicated) logarithm. This is justified because ln p is a monotonic function of p. Therefore maximizing ln p also maximizes p.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

The first term in this expression is a constant independent of h and can therefore be discarded.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Maximizing this negative quantity is equivalent to minimizing the corresponding positive quantity.

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

Discard constants independent of h

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

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Thus, this Equation shows that the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of the squared errors between the observed training values d_i and the hypothesis predictions $h(x_i)$. This holds under the assumption that the observed training values d_i are generated by adding random noise to the true target value, where this random noise is drawn independently for each

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

This holds under the assumption that the observed training values d_i are generated by adding random noise to the true target value, where this random noise is drawn independently for each example from a Normal distribution with zero mean.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

As the above derivation makes clear, the squared error term $(d_i - h(x_i))^2$ follows directly from the exponent in the definition of the Normal distribution.

Similar derivations can be performed starting with other assumed noise distributions, producing different results.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Why is it reasonable to choose the Normal distribution to characterize noise?

1. it allows for a mathematically straightforward analysis
2. the smooth, bell-shaped distribution is a good approximation to many types of noise in physical systems

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

The Central Limit Theorem shows that the sum of a sufficiently large number of independent, identically distributed random variables itself obeys a Normal distribution, regardless of the distributions of the individual variables.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

This implies that noise generated by the sum of very many independent, but identically distributed factors will itself be Normally distributed.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Of course, in reality, different components that contribute to noise might not follow identical distributions, in which case this theorem will not necessarily justify our choice.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Minimizing the sum of squared errors is a common approach in many neural network, curve fitting, and other approaches to approximating real-valued functions.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Gradient descent methods that seek the least-squared error hypothesis in neural network learning

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Some limitations of this problem setting: The above analysis considers noise only in the **target value** of the training example and does not consider noise in the **attributes describing the instances themselves.** For example, if the problem is to learn to predict the weight of someone based on that person's age and height, then the above analysis assumes noise in measurements of weight, but perfect measurements of age and height.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

The analysis becomes significantly more complex as these simplifying assumptions are removed.

NAIVE BAYES CLASSIFIER

The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function f(x) can take on any value from some finite set V.

NAIVE BAYES CLASSIFIER

A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $< a_1, a_2, \ldots, a_n >$.

The learner is asked to predict the target value, or classification, for this new instance.

NAIVE BAYES CLASSIFIER

The Bayesian approach to classifying the new instance is to assign the most probable target value, v_{MAP} given the attribute values $< a_1, a_2, \ldots, a_n >$ that describe the instance.

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

NAIVE BAYES CLASSIFIER

Using Bayes theorem, we can write:

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)}$$

$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

NAIVE BAYES CLASSIFIER

Attempt to estimate the two terms in Equation based on the training data:

$$v_{MAP} = argmax_{v_j \in V} P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)$$

It is easy to estimate each of the $P(v_j)$ simply by counting the frequency with which each target value v_j occurs in the training data.

NAIVE BAYES CLASSIFIER

However, estimating the different $P(a_1, a_2, \ldots, a_n \mid v_j)$ terms in this fashion is not feasible unless we have a very, very large set of training data. The problem is that the number of these terms is equal to the number of possible instances times the number of possible target values. Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.

NAIVE BAYES CLASSIFIER

The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.

In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2, \ldots, a_n$ is just the product of the probabilities for the individual attributes:

NAIVE BAYES CLASSIFIER

In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2, \ldots, a_n$ is just the product of the probabilities for the individual attributes:

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

NAIVE BAYES CLASSIFIER

Naive Bayes classifier:

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

where

v_{NB} denotes the target value output by the naive Bayes classifier

NAIVE BAYES CLASSIFIER

Naive Bayes classifier:

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

Notice that in a naive Bayes classifier the number of distinct $P(a_i | v_j)$ terms that must be estimated from the training data is just the number of distinct attribute values times the number of distinct target values - a much smaller number than if we were to estimate the $P(a_1, a_2, \ldots, a_n | v_j)$ terms as first contemplated.

NAIVE BAYES CLASSIFIER

To summarize, the naive Bayes learning method involves a learning step in which the various P(vj) and P(ai|vj)terms are estimated, based on their frequencies over the training data. The set of these estimates corresponds to the learned hypothesis. This hypothesis is then used to classify each new instance by applying the rule in Equation $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$. Whenever the naive Bayes assumption of conditional independence is satisfied, this naive Bayes classification v_{NB} is identical to the MAP classification.

NAIVE BAYES CLASSIFIER

One interesting difference between the naive Bayes learning method and other learning methods we have considered is that there is no explicit search through the space of possible hypotheses (in this case, the space of possible hypotheses is the space of possible values that can be assigned to the various $P(v_j)$ and $P(a_i | v_j)$terms). Instead,the hypothesis is formed without searching, simply by counting the frequency of various data combinations within the training examples.

NAIVE BAYES CLASSIFIER

An example:

Whether someone will play tennis on a day?

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Training examples for the target concept Play Tennis.

NAIVE BAYES CLASSIFIER

Objective: to classify the new novel instance using Naive Bayes classifier

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

Our task is to predict the target value (yes or no) of the target concept PlayTennis for this new instance.

NAIVE BAYES CLASSIFIER

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

Instantiating Equation $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$ to fit the current task, the target value v_{NB} is given by

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

$$= \underset{v_j \in \{yes,no\}}{\operatorname{argmax}} P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$P(Humidity = high|v_j) P(Wind = strong|v_j)$$

NAIVE BAYES CLASSIFIER

$$v_{NB} = \underset{v_j \in \{yes, no\}}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

$$= \underset{v_j \in \{yes, no\}}{\operatorname{argmax}} P(v_j) \quad P(Outlook = sunny | v_j) P(Temperature = cool | v_j)$$

$$P(Humidity = high | v_j) P(Wind = strong | v_j)$$

Notice in the final expression that a_i has been instantiated using the particular attribute values of the new instance.

To calculate v_{NB} we now require 10 probabilities that can be estimated from the training data.

NAIVE BAYES CLASSIFIER

First, the probabilities of the different target values can easily be estimated based on their frequencies over the 14 training examples.

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

NAIVE BAYES CLASSIFIER

Similarly, we can estimate the conditional probabilities. For example, those for Wind= strong are

$$P(Wind = strong | PlayTennis = yes) = 3/9 = .33$$

$$P(Wind = strong | PlayTennis = no) = 3/5 = .60$$

NAIVE BAYES CLASSIFIER

We calculate v_{NB} as

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Thus, the naive Bayes classifier assigns the target value PlayTennis=no to this new instance, based on the probability estimates learned from the training data.

NAIVE BAYES CLASSIFIER

Furthermore, by normalizing the above quantities to sum to one we can calculate the conditional probability that the target value is no, given the observed attribute values.

Here, this probability is

$$\frac{.0206}{.0206+.0053} = .795$$

Estimating probabilities in NB

Up to this point we have estimated probabilities by the fraction of times the event is observed to occur over the total number of opportunities.

When we compute:

$P(wind = strong \mid PlayTennis = No)$ by the fraction n_c/n

n=5 and n_c=3.

Estimating probabilities in NB

While this observed fraction provides a good estimate of the probability in many cases, it provides poor estimates when n_c is very small.

Imagine: that $P(wind = strong \mid PlayTennis = No)$=0.08 and there is a sample of 5 examples for which PlayTennis=No.

Then the most probable value of n_c is 0.

Estimating probabilities in NB

This raises two difficulties.

First, n_c/n produces a biased underestimate of the probability.

Second, when this probability estimate is zero, this probability term will dominate the Bayes classifier if the future query contains wind=strong.

The reason is that the quantity calculated in Equationrequires multiplying all the other probability terms by this zero value.

Estimating probabilities in NB

To avoid this difficulty we can adopt a Bayesian approach to estimating the probability, using the m-estimate defined as follows.

m-estimate of probability:

$$\frac{n_c + mp}{n + m}$$

Estimating probabilities in NB

Here, n_c and n are defined as before, p is our prior estimate of the probability we wish to determine, and m is a constant called the equivalent sample size which determines how heavily to weight p relative to the observed data.

Estimating probabilities in NB

A typical method for choosing p in the absence of other information is to assume uniform priors; that is, if an attribute has k possible values we set p=$1/k$.

For example, in estimating P(Wind = strong|PlayTennis = no) we note the attribute wind Wind has two possible values, so uniform priors would correspond to choosing p=.5. Note that if m is zero, the m-estimate is equivalent to the simple fraction n_c/n. If both n and m are nonzero, then the observed fraction n_c/n

Estimating probabilities in NB

and prior p will be combined according to the weight m.

The reason m is called the equivalent sample size is that

Equation $\frac{n_c + mp}{n + m}$ can be interpreted as augmenting the n actual observations by an additional m virtual samples distributed according to p.

BAYESIAN BELIEF NETWORKS

The naive Bayes classifier makes significant use of the assumption that the values of the attributes $a_1, a_2, \ldots, a_n$ are conditionally independent given the target value v.

This assumption dramatically reduces the complexity of learning the target function. When it is met, the naive Bayes classifier outputs the optimal Bayes classification.

However, in many cases this conditional independence assumption is clearly overly restrictive.

BAYESIAN BELIEF NETWORKS

A Bayesian belief network describes the **probability distribution** governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities.

BAYESIAN BELIEF NETWORKS

In contrast to the naive Bayes classifier, which assumes that all the variables are conditionally independent given the value of the target variable, Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables.

BAYESIAN BELIEF NETWORKS

Thus, **Bayesian belief networks** provide an **intermediate approach** that is **less constraining** than the **global assumption of conditional independence** made by the naive Bayes classifier, but **more tractable than avoiding conditional independence** assumptions altogether.

BAYESIAN BELIEF NETWORKS

In general, a Bayesian belief network describes the **probability distribution over a set of variables**.

BAYESIAN BELIEF NETWORKS

Consider an arbitrary set of random variables $Y_1, Y_2, \ldots, Y_n$ where each variable Y_i can take on the set of possible values $V(Y_i)$.

We define the joint space of the set of variables Y to be the cross product $V(Y_1) \times V(Y_2) \times \ldots \times V(Y_n)$.

BAYESIAN BELIEF NETWORKS

In other words, each item in the joint space corresponds to one of the possible assignments of values to the tuple of variables $< Y_1 \ldots Y_n >$ The probability distribution over this joint' space is called the joint probability distribution.

The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple $< Y_1 \ldots Y_n >$ A Bayesian belief network describes the joint probability distribution for a set of variables.

BAYESIAN BELIEF NETWORKS (Conditional Independence)

Let X, Y, and Z be three discrete-valued random variables.

We say that X is conditionally independent of Y given Z

if the probability distribution governing X is independent of the value of Y given a value for Z.

That is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

where $x_i \in V(X)$, $y_j \in V(Y)$, and $z_k \in V(Z)$

Commonly written as $P(X|Y,Z) = P(X|Z)$

BAYESIAN BELIEF NETWORKS (Conditional Independence)

This definition of conditional independence can be extended to sets of variables as well.

We say that the set of variables $X_1, X_2, \ldots, X_l$ is conditionally independent of the set of variables $Y_1, Y_2, \ldots, Y_m$ given the set of variables $Z_1, Z_2, \ldots, Z_n$ if

$$P(X_1 \ldots X_l | Y_1 \ldots Y_m, Z_1 \ldots Z_n) = P(X_1 \ldots X_l | Z_1 \ldots Z_n)$$

BAYESIAN BELIEF NETWORKS (Conditional Independence)

Note the correspondence between this definition and our use of conditional independence in the definition of the naive Bayes classifier.

The naive Bayes classifier assumes that the instance attribute A_1 is conditionally independent of instance attribute A_2 given the target value V.

BAYESIAN BELIEF NETWORKS (Conditional Independence)

This allows the naive Bayes classifier to calculate

$P(A_1, A_2 \mid V)$

$$P(A_1, A_2|V) = P(A_1|A_2, V)P(A_2|V)$$
$$= P(A_1|V)P(A_2|V)$$

The second line follows from the definition of conditional independence because A_1 is conditionally independent of A_2

$$P(A_1 \mid A_2, V) = P(A_1 \mid V)$$

BAYESIAN BELIEF NETWORKS (Representation)

A graph comprises nodes (also called vertices) connected by links (also known as edges or arcs).

BAYESIAN BELIEF NETWORKS (Representation)

In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relationships between these variables.

BAYESIAN BELIEF NETWORKS (Representation)

The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables.

BAYESIAN BELIEF NETWORKS (Representation)

Directed graphs are useful for expressing causal relationships between random variables, whereas undirected graphs are better suited to expressing soft constraints between random variables.

For the purposes of solving inference problems, it is often convenient to convert both directed and undirected graphs into a different representation called a factor graph.

BAYESIAN BELIEF NETWORKS (Representation)

In order to motivate the use of directed graphs to describe probability distributions, consider first an arbitrary joint distribution p(a, b, c) over three variables a, b, and c.

By application of the product rule of probability

$$p(a, b, c) = p(c|a, b)p(a, b).$$

also

$$p(a, b, c) = p(c|a, b)p(b|a)p(a)$$

BAYESIAN BELIEF NETWORKS (Representation)

$$p(a, b, c) = p(c|a, b)p(b|a)p(a)$$

A directed graphical model representing the joint probability distribution over three variables a, b, and c, corresponding to the decomposition on the right-hand side

BAYESIAN BELIEF NETWORKS (Representation)

The joint distribution over K variables given by $p(x_1,\ldots,x_K)$. By repeated application of the product rule of probability, this joint distribution can be written as a product of conditional distributions, one for each of the variables

$$p(x_1, \ldots, x_K) = p(x_K|x_1, \ldots, x_{K-1}) \ldots p(x_2|x_1)p(x_1)$$

BAYESIAN BELIEF NETWORKS (Representation)

Example of a directed acyclic graph describing the joint distribution over variables $x_1, \ldots, x_7$. The corresponding decomposition of the joint distribution is given by

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5).$$

BAYESIAN BELIEF NETWORKS (Representation)

Thus, for a graph with K nodes, the joint distribution is given by

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

where pa_k denotes the set of parents of x_k, and $x = \{x_1, x_2, \ldots, x_k\}$.

BAYESIAN BELIEF NETWORKS (Representation)

Such graphs are directed acyclic graphs, or DAGs.

This is equivalent to the statement that there exists an ordering of the nodes such that there are no links that go from any node to any lower numbered node.

BAYESIAN BELIEF NETWORKS (Representation)

A Bayesian belief network (Bayesian network for short) represents the joint probability distribution for a set of variables.

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

Campfire

Each node is asserted to be conditionally independent of its nondescendants, given its immediate parents.

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

Campfire

Associated with each node is a conditional probability table, which specifies the conditional distribution for the variable given its immediate parents in the graph.

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

Campfire

The conditional probability table for the Campjire node is shown at the right, where **Campfire** is abbreviated to C, Storm abbreviated to S, and BusTourStop abbreviated to B

BAYESIAN BELIEF NETWORKS (Representation)

For each variable two types of information are specified.

First, the network arcs represent the assertion that the variable is conditionally independent of its nondescendants in the network given its immediate predecessors in the network. We say X is a **descendant** of Y if there is a **directed path** from Y to X.

Second, a **conditional probability table** is given for each variable, describing the probability distribution for that variable given the values of its immediate predecessors.

BAYESIAN BELIEF NETWORKS (Representation)

The joint probability for any desired assignment of values $< y_1, \ldots, y_n >$ to the tuple of network variables $< Y_1, \ldots, Y_n >$ can be computed by the formula:

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

where Parents(Yi) denotes the set of immediate predecessors of Yi in the network. The values of P(yi|Parents(Yi)) are precisely the values stored in the conditional probability table associated with node Y_i.

BAYESIAN BELIEF NETWORKS (Representation)

To illustrate, the Bayesian network in the Figure represents the joint probability distribution over the boolean variables **Storm, Lightning, Thunder, Forest-Fire, Campfire and BusTourGroup.**

Consider the node Campfire. The network nodes and arcs represent the assertion that Campfire is conditionally independent of its nondescendants Lightning and Thunder, given its immediate parents Storm and BusTourGroup.

BAYESIAN BELIEF NETWORKS (Representation)

This means that once we know the value of the variables Storm and BusTourGroup the variables Lighting and Thunder provide no additional information about Campfire.

The right side of the figure shows the conditional probability table associated with the variable Campfire.

BAYESIAN BELIEF NETWORKS (Representation)

The top left entry in this table, for example, expresses the assertion that

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

Campfire

$$P(Campfire = True | Storm = True, BusTourGroup = True) = 0.4$$

BAYESIAN BELIEF NETWORKS (Representation)

Note this table provides only the conditional probabilities of Campfire given its parent variables Storm and BustourGroup.

The set of local conditional probability tables for all the variables, together with the set of conditional independence assumptions described by the network, describe the full joint probability distribution for the network.

BAYESIAN BELIEF NETWORKS (Representation)

Bayesian belief networks allow a convenient way to represent causal knowledge such as the fact that Lighting causes Thunder

In the terminology of conditional independence, we express this by stating that Thunder is conditionally independent of other variables in the network, given the value of Lighting.

Note this conditional independence assumption is implied by the arcs in the Bayesian network of Figure.

BAYESIAN BELIEF NETWORKS (Inference)

Bayesian belief networks allow a convenient way to represent causal knowledge such as the fact that Lighting causes Thunder

In the terminology of conditional independence, we express this by stating that Thunder is conditionally independent of other variables in the network, given the value of Lighting.

Note this conditional independence assumption is implied by the arcs in the Bayesian network of Figure.

BAYESIAN BELIEF NETWORKS (Inference)

To use a Bayesian network to infer the value of some target variable (e.g., ForestFire) given the observed values of the other variables.

Of course, given that we are dealing with random variables it will not generally be correct to assign the target variable a single determined value.

BAYESIAN BELIEF NETWORKS (Inference)

What we really wish to infer is the probability distribution for the target variable, which specifies the probability that it will take on each of its possible values given the observed values of the other variables.

This inference step can be straightforward if values for all of the other variables in the network are known exactly.

BAYESIAN BELIEF NETWORKS (Inference)

In the more general case we may wish to infer the probability distribution for some variable (e.g., ForestFire) given observed values for only a subset of the other variables (e.g., Thunder and BusTourGroup may be the only observed values available).

In general, a Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables.

BAYESIAN BELIEF NETWORKS (Inference)

Exact inference of probabilities in general for an arbitrary Bayesian network is known to be NP-hard (Cooper 1990).

Learning Bayesian Belief Networks

Can we devise effective algorithms for learning Bayesian belief networks from training data?

Learning Bayesian Belief Networks

This question is a focus of much current research. Several different settings for this learning problem can be considered.

First, the network structure might be given in advance, or it might have to be inferred from the training data.

Second, all the network variables might be directly observable in each training example, or some might be unobservable.

Learning Bayesian Belief Networks

In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward.

We simply estimate the conditional probability table entries just as we would for a naive Bayes classifier.

Learning Bayesian Belief Networks

In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward.

We simply estimate the conditional probability table entries just as we would for a naive Bayes classifier.

Learning Bayesian Belief Networks

In the case where the network structure is given but only some of the variable values are observable in the training data, the learning problem is more difficult.

This problem is somewhat analogous to learning the weights for the hidden units in an **artificial neural network**, where the input and output node values are given but the hidden unit values are left unspecified by the training examples

Learning Bayesian Belief Networks

In fact, Russell et al. (1995) propose a similar gradient ascent procedure that learns the entries in the conditional probability tables. This gradient ascent procedure searches through a space of hypotheses that corresponds to the set of all possible entries for the conditional probability tables.

Learning Bayesian Belief Networks

The objective function that is maximized during gradient ascent is the probability P(D1h) of the observed training data D given the hypothesis h. By definition, this corresponds to searching for the maximum likelihood hypothesis for the table entries.

The EM (Expectation Maximization) algorithm

In many practical learning settings, only a subset of the relevant instance features might be observable.

In the previous example of Bayesian Network, we might have data where only a subset of the network variables Storm, Lightning, Thunder, ForestFire, Campfire, and BusTourGroup have been observed.

Many approaches have been proposed to handle the problem of learning in the presence of unobserved (latent) variables.

The EM (Expectation Maximization) algorithm

The EM algorithm (Dempster et al. 1977), is a widely used approach to learning in the presence of unobserved variables.

The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known.

The EM algorithm is also the basis for many unsupervised clustering algorithms.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

An example:

Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normal distributions.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

k=2:

Instances generated by a mixture of two Normal distributions with identical variance σ.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

k=2:

If the means of the Normal distributions are unknown, the EM algorithm can be used to search for their maximum likelihood estimates.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

k=2:

Each instance is generated using a two-step process.

First, one of the k Normal distributions is selected at random.

Second, a single random instance x_i is generated according to this selected distribution.

This process is repeated to generate a set of data points as shown in the figure.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

k=2:

To simplify —>

Let's consider the special case where the selection of the single Normal distribution at each step is based on choosing each with uniform probability, where each of the k Normal distributions has the same variance a σ^2, and where σ^2 is known.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

The learning task is to output a hypothesis $h = < \mu_1, \ldots, \mu_k >$ that describes the means of each of the k distributions.

We would like to find a maximum likelihood hypothesis for these means; that is, a hypothesis h that maximizes p(D|h).

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Note it is easy to calculate the maximum likelihood hypothesis for the mean of a single Normal distribution given the observed data instances $x_1, x_2, \ldots, x_m$ drawn from this single distribution.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

This problem of finding the mean of a single distribution is a special case of the problem that the maximum likelihood hypothesis is the one that minimizes the sum of squared errors over the training instances.

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2 \longrightarrow \mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2$$

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

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

$\longrightarrow$

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2$$

In this case, the sum of squared errors is minimized by the sample mean

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Our problem here, however, involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution. Thus, we have a prototypical example of a problem involving hidden variables.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Here, we can think of the full description of each instance as the triple (x_i, z_{i1}, z_{i2}), where x_i is the observed value of the ith instance and where z_{i1} and z_{i2} indicate which of the two Normal distributions was used to generate the value x_i.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

In particular, z_{ij} has the value 1 if x_i was created by the jth Normal distribution and **0** otherwise.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Here x_i is the observed variable in the description of the instance, and z_{i1} and z_{i2} are hidden variables. If the values of z_{i1} and z_{i2} were observed, we could use Equation

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2$$

to solve for the means μ_1 *and* μ_2.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

We use the EM algorithm, hence.

Applied to our k-means problem the EM algorithm searches for a maximum likelihood hypothesis by repeatedly re-estimating the expected values of the hidden variables z_{ij} given its current hypothesis $< \mu_1, \mu_2, \ldots, \mu_k >$ then recalculating the maximum likelihood hypothesis using these expected values for the hidden variables.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

An instance of the EM algorithm for this problem:

The EM algorithm first initializes the hypothesis to $h = < \mu_1, \mu_2 >$ where μ_1 *and* μ_2 are two arbitrary initial values.

It then iteratively re-estimates h by repeating the following two steps until the procedure converges to a stationary value for h

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Step 1: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

Step 2: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated in Step 1. Then replace the hypothesis $h = \langle \mu_1, \mu_2 \rangle$ by the new hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$ and iterate.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Implementation in practice:

Step 1: To calculate expected value of each z_{ij}. This $E(z_{ij})$ is just the probability that instance x_i was generated by the jth Normal distribution.

$$E[z_{ij}] = \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Implementation in practice:

$$E[z_{ij}] = \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

Thus the first step is implemented by substituting the current values μ_1 *and* μ_2 and the observed x_i into the above expression.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

Step 2: We use the $E(z_{ij})$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = < \mu'_1, \mu'_2 >$ given by

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \quad x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

(discussed later)

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \; x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

-Similar to the sample mean from Equation $\mu_{ML} = \frac{1}{m}\sum_{i=1}^{m} x_i$ that is used to estimate μ for a single Normal distribution. Here, it is just the weighted sample mean for μ_j with each instance weighted by the expectation $E(z_{ij})$ that it was generated by the jth Normal distribution.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

The above algorithm for estimating the means of a mixture of k Normal distributions illustrates the essence of the EM approach:

The **current hypothesis** is used to **estimate** the unobserved variables, and

the **expected values** of these variables are then used to calculate an **improved** hypothesis.

The EM (Expectation Maximization) algorithm

Estimating Means of k Gaussians

It can be proved that on each iteration through this loop, the EM algorithm increases the likelihood P(D|h) **unless it is at a local maximum**. The algorithm thus converges to a local maximum likelihood hypothesis for μ_1 *and* μ_2.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

More generally, the EM algorithm can be applied in many settings where we wish to estimate some set of parameters θ that describe an underlying probability distribution, given only the observed portion of the full data produced by this distribution.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

In the two-means example the parameters of interest were $\theta = < \mu_1, \mu_2 >$ and the full data were the triples $< x_i, z_{i1}, z_{i2} >$ of which only the x_i were observed.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

In general,

let $X = \{x_1, \ldots, x_m\}$ **—> the observed data in a set of m independently drawn instances,**

let $Z = \{z_1, \ldots, z_m\}$ **—> the unobserved data in these same instances, and**

let $Y = X \cup Z$ **—> the full data.**

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

Note the **unobserved Z** can be treated as **a random variable** whose **probability distribution depends on the unknown parameters** θ and **on the observed data X**.

Similarly, **Y is a random variable** because it is defined **in terms of the random variable Z**.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

We use **h** to denote the **current hypothesized** values of the parameters θ, and h' to denote the **revised hypothesis** that is estimated on each iteration of the EM algorithm.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

The EM algorithm searches for the maximum likelihood hypothesis h' by seeking the h' that maximizes $E[\ln P(Y|h')]$.

This expected value is taken over the probability distribution governing Y which is determined by the unknown parameters θ.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

Exact significance of $E[\ln P(Y|h')]$**?**

First, $P(Y|h')$ **is the likelihood of the full data Y given hypothesis** h'**.**
That is, we wish to find a h' **that maximizes some function of this quantity.**

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

Exact significance of $E[\ln P(Y|h')]$**?**

Second, maximizing the logarithm of this quantity $\ln P(Y|h')$ **also maximizes** $P(Y|h')$**.**

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

Exact significance of $E[\ln P(Y|h')]$?

Third, we use the expected value $E[\ln P(Y|h')]$ because the full data Y is itself a random variable. Given that the full data Y is a combination of the observed data X and unobserved data Z we average over the possible values of the unobserved Z, weighting each according to its probability.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

Exact significance of $E[\ln P(Y|h')]$?

That is, we take the expected value $E[\ln P(Y|h')]$ over the probability distribution governing the random variable Y.

The distribution governing Y is determined by the completely known values for X plus the distribution governing Z.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

What is the probability distribution governing Y?

In general we will not know this distribution because it is determined by the parameters θ that we are trying to estimate.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

What is the probability distribution governing Y?

Therefore, the EM algorithm uses its current hypothesis h in place of the actual parameters Θ to estimate the distribution governing Y.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

What is the probability distribution governing Y?

Let us define a function $Q(h'|h)$ **that gives** $E[\ln P(Y|h')]$ **as a function of** h' **under the assumption that** $\theta = h$ **and given the observed portion X of the full data Y :**

$$Q(h'|h) = E[\ln p(Y|h')|h, X]$$

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

What is the probability distribution governing Y?

$$Q(h'|h) = E[\ln p(Y|h')|h, X]$$

We write this function Q in the form Q(h|h') to indicate that it is defined in part by the assumption that the current hypothesis h is equal to Θ.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

In its general form, the EM algorithm repeats the following two steps until convergence:

Step 1: *Estimation (E) step:* Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

Step 2: *Maximization (M) step:* Replace hypothesis h by the hypothesis h' that maximizes this Q function.

$$h \leftarrow \underset{h'}{\operatorname{argmax}}\, Q(h'|h)$$

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

When the function Q is continuous, the EM algorithm converges to a stationary point of the likelihood function $P(Y|h')$**.**

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

When this likelihood function has a single maximum, EM will converge to this global maximum likelihood estimate for h'.

Otherwise, it is guaranteed only to converge to a local maximum.

The EM (Expectation Maximization) algorithm

General Statement of EM Algorithm

In this respect, EM shares some of the same limitations as other optimization methods such as gradient descent, line search, and conjugate gradient.