MODULE -4

BAYSEIAN LEARNING

CONTENT

- Introduction
- Bayes theorem
- Bayes theorem and concept learning
- Maximum likelihood and Least Squared Error Hypothesis
- Maximum likelihood Hypotheses for predicting probabilities
- Minimum Description Length Principle
- Naive Bayes classifier
- Bayesian belief networks
- EM algorithm

INTRODUCTION

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

Features of Bayesian Learning Methods

- 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
- 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.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- 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.

Practical difficulty in applying Bayesian 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.

- A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case. In certain specialized situations, this computational cost can be significantly reduced.

BAYES THEOREM

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.

Notations

- P(h) *prior probability of* h, reflects any background knowledge about the chance that h is correct
- P(D) *prior probability of* D, probability that D will be observed
- P(D|h) probability of observing D given a world in which h holds
- P(h|D) *posterior probability of* h, reflects confidence that h holds after D has been observed

Bayes theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability ***P(h|D),*** from the prior probability ***P(h),*** together with ***P(D)*** and ***P(D(h).***

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.

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.***

Maximum a Posteriori (MAP) Hypothesis

- 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. Any such maximally probable hypothesis is called a ***maximum a posteriori (MAP) hypothesis***.
- Bayes theorem to calculate the posterior probability of each candidate hypothesis is h_{MAP} is a MAP hypothesis provided

$$h_{MAP} = \underset{h \in H}{argmax}\ P(h|D)$$

$$= \underset{h \in H}{argmax}\ \frac{P(D|h)P(h)}{P(D)}$$

$$= \underset{h \in H}{argmax}\ P(D|h)P(h)$$

- P(D) can be dropped, because it is a constant independent of h

Maximum Likelihood (ML) Hypothesis

In some cases, it is assumed that every hypothesis in H is equally probable a priori ***($P(h_i) = P(h_j)$*** for all ***h_i*** and ***h_j*** in H).

In this case the below equation can be simplified and need only consider the term ***P(D|h)*** to find the most probable hypothesis.

$$h_{MAP} = \underset{h \in H}{argmax}\ P(D|h)P(h)$$

the equation can be simplified

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

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

Example

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

- The patient has a particular form of cancer (denoted by ***cancer***)
- The patient does not (denoted by ¬ ***cancer***)

The available data is from a particular laboratory with two possible outcomes: + (positive) and - (negative)

$$P(cancer) = .008 \qquad P(\neg cancer) = 0.992$$

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

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

- Suppose a new patient is observed for whom the lab test returns a positive (+) result.
- Should we diagnose the patient as having cancer or not?

$$P(\oplus|cancer)P(cancer) = (.98).008 = .0078$$
$$P(\oplus|\neg cancer)P(\neg cancer) = (.03).992 = .0298$$
$$\Rightarrow h_{MAP} = \neg cancer$$

BAYES THEOREM AND CONCEPT LEARNING

What is the relationship between Bayes theorem and the problem of concept learning?

Since Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data, and can use it as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable.

Brute-Force Bayes Concept Learning

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on Bayes theorem, as follows:

Brute-Force MAP LEARNING algorithm

1. For each hypothesis h in H calculate the posterior probability

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

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{argmax}\ P(h|D)$$

In order specify a learning problem for the **BRUTE-FORCE MAP LEARNING** algorithm we must specify what values are to be used for ***P(h)*** and for ***P(D|h) ?***

Lets choose ***P(h)*** and for ***P(D|h)*** to be consistent with the following assumptions:

- The training data ***D*** is noise free (i.e., $d_i = c(x_i)$)
- The target concept ***c*** is contained in the hypothesis space H
- We have no a priori reason to believe that any hypothesis is more probable than any other.

What values should we specify for ***P(h)?***

- Given no prior knowledge that one hypothesis is more likely than another, it is reasonable to assign the same prior probability to every hypothesis ***h*** in ***H.***
- Assume the target concept is contained in H and require that these prior probabilities sum to 1.

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

What choice shall we make for ***P(D|h)***?

- ***P(D|h)*** is the probability of observing the target values D = ***(d_1 . . .d_m)*** for the fixed set of instances ***(x_1 . . . x_m),*** given a world in which hypothesis ***h*** holds
- Since we assume noise-free training data, the probability of observing classification ***d_i*** given ***h*** is just ***1*** if ***d_i*** = ***h(x_i)*** and 0 if ***d_i*** # ***h(x_i).*** Therefore,

$$P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \in D \\ 0 & \text{otherwise} \end{cases}$$

Given these choices for ***P(h)*** and for ***P(D|h)*** we now have a fully-defined problem for the above **BRUTE-FORCE MAP LEARNING** algorithm.

In a first step, we have to determine the probabilities for P(h|D)

h is **inconsistent** with training data D

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

h is **consistent** with training data D

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

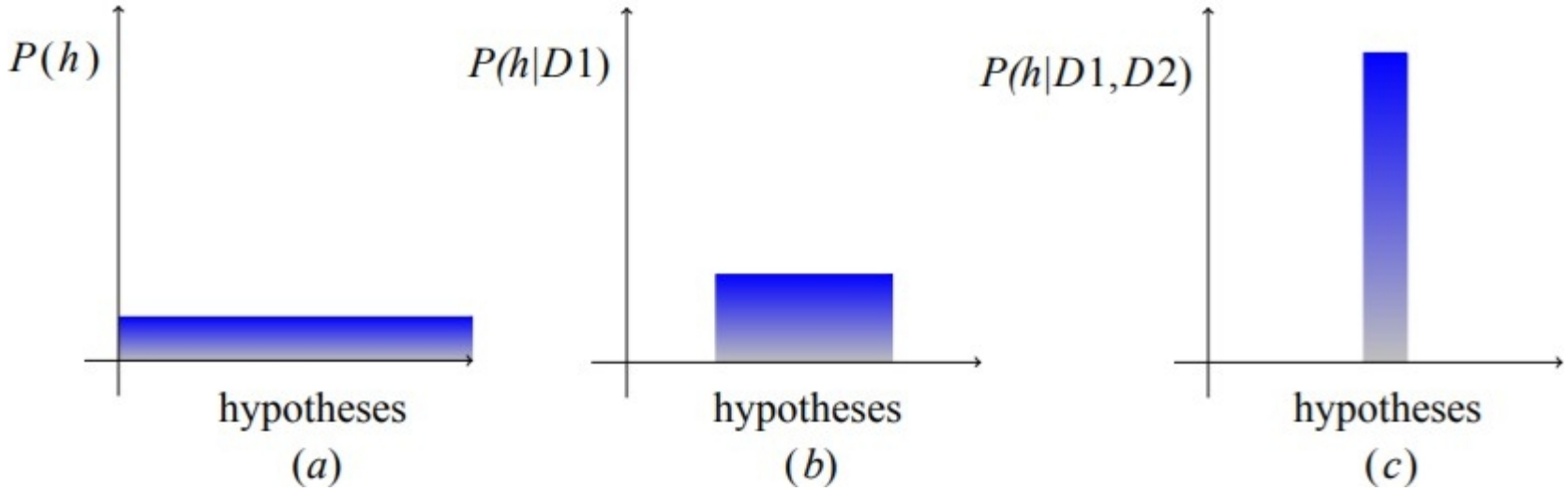
To summarize, Bayes theorem implies that the posterior probability P(h|D) under our assumed P(h) and P(D|h) is

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

where $|VS_{H,D}|$ is the number of hypotheses from H consistent with D

The Evolution of Probabilities Associated with Hypotheses

- Figure (a) all hypotheses have the same probability.
- Figures (b) and (c), As training data accumulates, the posterior probability for inconsistent hypotheses becomes zero while the total probability summing to 1 is shared equally among the remaining consistent hypotheses.



MAP Hypotheses and Consistent Learners

A learning algorithm is a consistent learner if it outputs a hypothesis that commits zero errors over the training examples.

Every consistent learner outputs a MAP hypothesis, if we assume a uniform prior probability distribution over H ($P(h_i) = P(h_j)$ for all i, j), and deterministic, noise free training data (P(D|h) =1 if D and h are consistent, and 0 otherwise).

Example:

- FIND-S outputs a consistent hypothesis, it will output a MAP hypothesis under the probability distributions P(h) and P(D|h) defined above.
- Are there other probability distributions for P(h) and P(D|h) under which FIND-S outputs MAP hypotheses? Yes.
- Because FIND-S outputs a maximally specific hypothesis from the version space, its output hypothesis will be a MAP hypothesis relative to any prior probability distribution that favours more specific hypotheses.

- Bayesian framework is a way to characterize the behaviour of learning algorithms
- By identifying probability distributions P(h) and P(D|h) under which the output is a optimal hypothesis, implicit assumptions of the algorithm can be characterized (Inductive Bias)
- Inductive inference is modelled by an equivalent probabilistic reasoning system based on Bayes theorem

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Consider the problem of learning a continuous-valued target function such as neural network learning, linear regression, and polynomial curve fitting

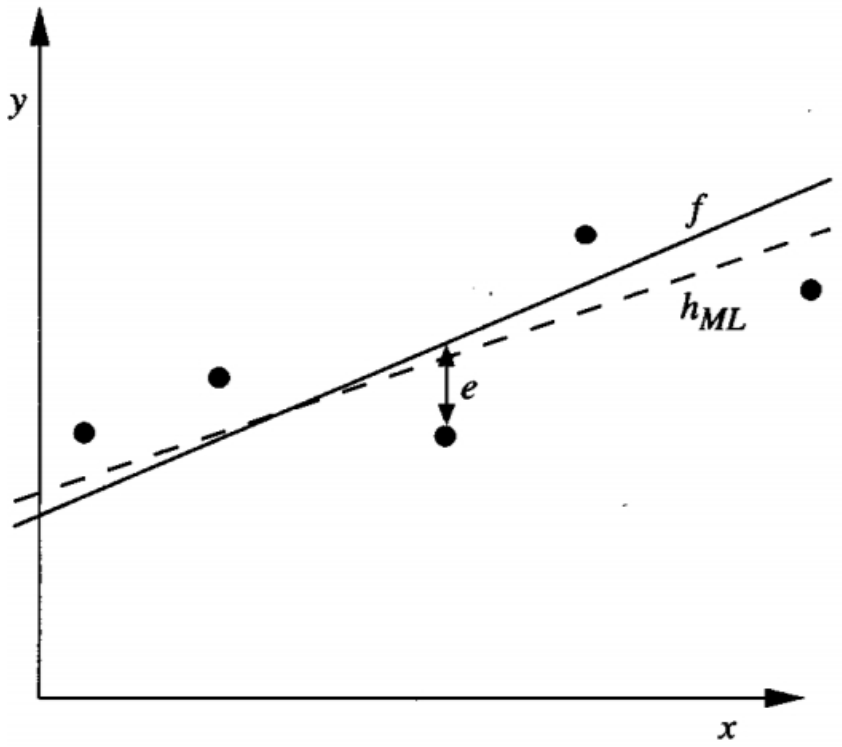
A straightforward Bayesian analysis will show that under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a maximum likelihood (ML) hypothesis

Learning A Continuous-Valued Target Function

- Learner L considers an instance space X and a hypothesis space H consisting of some class of real-valued functions defined over X, i.e., $(\forall\, h \in H)[\, h : X \rightarrow R]$ and training examples of the form $<x_i, d_i>$
- The problem faced by L is to learn an unknown target function $f : X \rightarrow R$
- 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 with zero mean ($d_i = f(x_i) + e_i$)
- Each training example is a pair of the form (x_i, d_i) where $di = f(x_i) + e_i$.
  - Here $f(x_i)$ is the noise-free value of the target function and e_i is a random variable representing the noise.
  - 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.

Learning A Linear Function

- The target function f corresponds to the solid line.
- The training examples (xi , di) are assumed to have Normally distributed noise ei with zero mean added to the true target value f (xi).
- 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



Before showing why a hypothesis that minimizes the sum of squared errors in this setting is also a maximum likelihood hypothesis, let us quickly review ***probability densities and Normal distributions***

Probability Density for continuous variables

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

e: a random noise variable generated by a Normal probability distribution

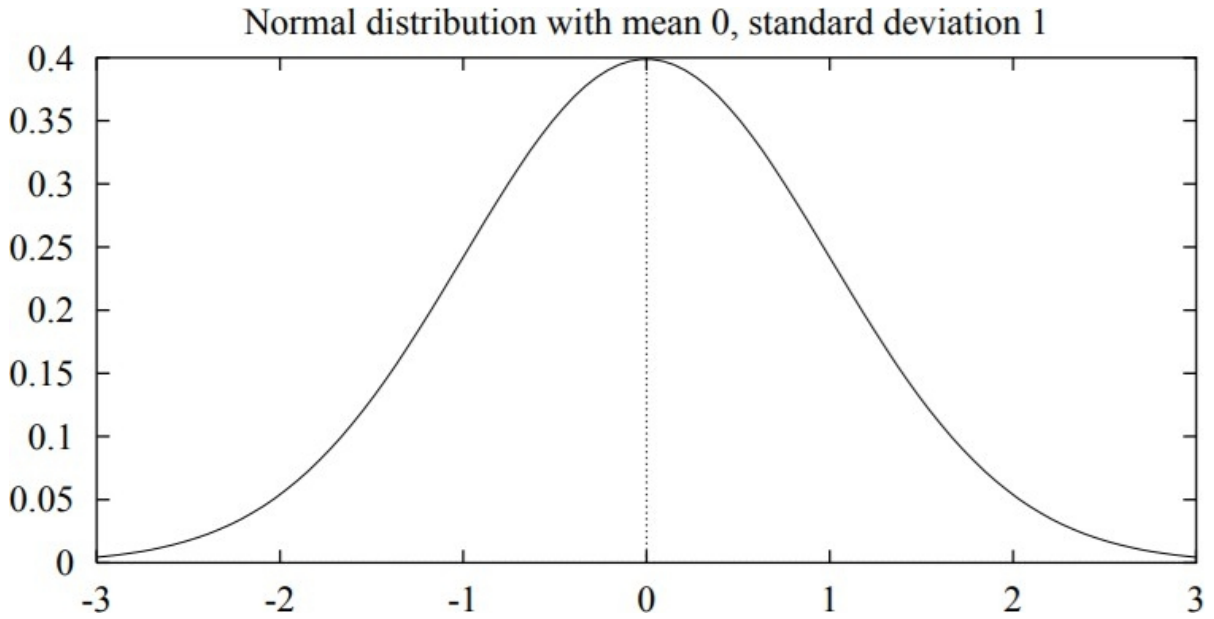
$<x_1 \ldots x_m>$: the sequence of instances (as before)

$<d_1 \ldots d_m>$: the sequence of target values with $d_i = f(x_i) + e_i$

Normal Probability Distribution (Gaussian Distribution)

A Normal distribution is a smooth, bell-shaped distribution that can be completely characterized by its mean μ and its standard deviation σ

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



- A Normal distribution is fully determined by two parameters in the formula: μ and σ .
- If the random variable X follows a normal distribution:
  - The probability that X will fall into the interval (a, b) is $\int_a^b p(x)dx$
  - The expected, or ***mean value of X***, $E[X] = \mu$
  - The ***variance of X***, $Var(X) = \sigma^2$
  - The ***standard deviation of X***, $\sigma_x = \sigma$
- The ***Central Limit Theorem*** states that the sum of a large number of independent, identically distributed random variables follows a distribution that is approximately ***Normal***.

Using the previous definition of h_{ML} we have

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

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

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

Given 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 around the true targetvalue $f(x_i)$. Because we are writing the expression for P(D|h), we assume h is the correct description of f. Hence, $\mu = f(x_i) = h(x_i)$

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

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

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

It is common to maximize the less complicated logarithm, which is justified because of the monotonicity of function p.

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

The first term in this expression is a constant independent of $\boldsymbol{h}$ and can therefore be discarded

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

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

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

Finally Discard constants that are independent of ***h***

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

- the h_{ML} is one that minimizes the sum of the squared errors

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

- good approximation of many types of noise in physical systems
- Central Limit Theorem shows that the sum of a sufficiently large number of independent, identically distributed random variables itself obeys a Normal distribution

Only noise in the target value is considered, not in the attributes describing the instances themselves

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

Consider the setting in which we wish to learn a nondeterministic (probabilistic) function $f : X \rightarrow \{0, 1\}$, which has two discrete output values.

We want a function approximator whose output is the probability that $f(x) = 1$

In other words , learn the target function

$$f' : X \rightarrow [0, 1] \text{ such that } f'(x) = P(f(x) = 1)$$

How can we learn f' using a neural network?

Use of brute force way would be to first collect the observed frequencies of 1's and 0's for each possible value of x and to then train the neural network to output the target frequency for each x.

What criterion should we optimize in order to find a maximum likelihood hypothesis for f' in this setting?

- First obtain an expression for P(D|h)
- Assume the training data D is of the form $D = \{(x_1, d_1) \ldots (x_m, d_m)\}$, where d_i is the observed 0 or 1 value for $f(x_i)$.
- Both x_i and d_i as random variables, and assuming that each training example is drawn independently, we can write P(D|h) as

$$P(D \mid h) = \prod_{i=1}^{m} P(x_i, d_i \mid h)$$

Applying the product rule

$$P(D \mid h) = \prod_{i=1}^{m} P(d_i \mid h, x_i) P(x_i)$$

The probability $P(d_i|h, x_i)$

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{if } d_i = 0 \end{cases} \qquad \text{equ (3)}$$

Re-express it in a more mathematically manipulable form, as

$$P(d_i|h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \qquad \text{equ (4)}$$

Equation (4) to substitute for $P(d_i|h, x_i)$ in Equation (5) to obtain

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i) \qquad \text{equ (5)}$$

We write an expression for the maximum likelihood hypothesis

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

The last term is a constant independent of h, so it can be dropped

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \qquad \text{equ (6)}$$

It easier to work with the log of the likelihood, yielding

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \qquad \text{equ (7)}$$

Equation (7) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting

Gradient Search to Maximize Likelihood in a Neural Net

Derive a weight-training rule for neural network learning that seeks to maximize G(h, D) using gradient ascent

- The gradient of G(h, D) is given by the vector of partial derivatives of G(h, D) with respect to the various network weights that define the hypothesis h represented by the learned network
- In this case, the partial derivative of G(h, D) with respect to weight w_{jk} from input k to unit j is

$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m} \frac{\partial G(h,D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{\partial (d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}} \qquad \text{equ (1)}$$

Suppose our neural network is constructed from a single layer of sigmoid units. Then,

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i)x_{ijk} = h(x_i)(1 - h(x_i))x_{ijk}$$

where x_{ijk} is the k^{th} input to unit j for the i^{th} training example, and d(x) is the derivative of the sigmoid squashing function.

Finally, substituting this expression into Equation (1), we obtain a simple expression for the derivatives that constitute the gradient

$$\frac{\partial G(h, D)}{\partial w_{jk}} = \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ijk}$$

Because we seek to maximize rather than minimize P(D|h), we perform ***gradient ascent*** rather than ***gradient descent search***. On each iteration of the search the weight vector is adjusted in the direction of the gradient, using the weight update rule

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

Where,

$$\Delta w_{jk} = \eta \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ijk} \qquad \text{equ (2)}$$

where η is a small positive constant that determines the step size of the i gradient ascent search

It is interesting to compare this weight-update rule to the weight-update rule used by the BACKPROPAGATION algorithm to minimize the sum of squared errors between predicted and observed network outputs.

The BACKPROPAGATION update rule for output unit weights, re-expressed using our current notation, is

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

Where,

$$\Delta w_{jk} = \eta \sum_{i=1}^{m} h(x_i)(1 - h(x_i))(d_i - h(x_i))\ x_{ijk}$$

MINIMUM DESCRIPTION LENGTH PRINCIPLE

- A Bayesian perspective on Occam's razor
- Motivated by interpreting the definition of h_{MAP} in the light of basic concepts from information theory.

$$h_{MAP} = \underset{h \in H}{argmax}\ P(D|h)P(h)$$

which can be equivalently expressed in terms of maximizing the $\log_2$

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

or alternatively, minimizing the negative of this quantity

$$h_{MAP} = \underset{h \in H}{argmin}\ -\log_2 P(D|h) - \log_2 P(h) \qquad \text{equ (1)}$$

- This equation can be interpreted as a statement that short hypotheses are preferred, assuming a particular representation scheme for encoding hypotheses and data

Introduction to a basic result of information theory

- Consider the problem of designing a code to transmit messages drawn at random
- i is the message
- The probability of encountering message i is p_i
- Interested in the most compact code; that is, interested in the code that minimizes the expected number of bits we must transmit in order to encode a message drawn at random
- To minimize the expected code length we should assign shorter codes to messages that are more probable
- Shannon and Weaver (1949) showed that the optimal code (i.e., the code that minimizes the expected message length) assigns - log, pi bits to encode message i.
- The number of bits required to encode message i using code C as the ***description length of message i with respect to C***, which we denote by $L_c(i)$.

Interpreting the equation

$$h_{MAP} = \underset{h \in H}{argmin} \; -\log_2 P(D|h) - \log_2 P(h) \qquad \text{equ (1)}$$

- $-\log_2 P(h)$: the description length of h under the optimal encoding for the hypothesis space H $L_{CH}(h) = -\log_2 P(h)$, where C_H is the optimal code for hypothesis space H.
- $-\log_2 P(D \mid h)$: the description length of the training data D given hypothesis h, under the optimal encoding fro the hypothesis space H: $L_{CH}(D|h) = -\log_2 P(D| h)$, where $C_{D|h}$ is the optimal code for describing data D assuming that both the sender and receiver know the hypothesis h.

Rewrite Equation (1) to show that h_{MAP} is the hypothesis h that minimizes the sum given by the description length of the hypothesis plus the description length of the data given the hypothesis.

$$h_{MAP} = \underset{h \in H}{argmin} \; L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

where C_H and $C_{D|h}$ are the optimal encodings for H and for D given h

The Minimum Description Length (MDL) principle recommends choosing the hypothesis that minimizes the sum of these two description lengths of equ.

$$h_{MAP} = \underset{h \in H}{argmin}\ L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

Minimum Description Length principle:

$$h_{MDL} = \underset{h \in H}{\text{argmin}}\ L_{C_1}(h) + L_{C_1}(D \mid h)$$

Where, codes C_1 and C_2 to represent the hypothesis and the data given the hypothesis

The above analysis shows that if we choose C_1 to be the optimal encoding of hypotheses C_H, and if we choose C_2 to be the optimal encoding $C_{D|h}$, then $h_{MDL} = h_{MAP}$

Application to Decision Tree Learning

Apply the MDL principle to the problem of learning decision trees from some training data.

What should we choose for the representations C_1 and C_2 of hypotheses and data?

- **For C_1:** C_1 might be some obvious encoding, in which the description length grows with the number of nodes and with the number of edges
- **For C_2:** Suppose that the sequence of instances (x_1 . . .x_m) is already known to both the transmitter and receiver, so that we need only transmit the classifications ($f(x_1)$ . . . $f(x_m)$).

  Now if the training classifications ($f(x_1)$ . . .$f(x_m)$) are identical to the predictions of the hypothesis, then there is no need to transmit any information about these examples. The description length of the classifications given the hypothesis ZERO

  If examples are misclassified by h, then for each misclassification we need to transmit a message that identifies which example is misclassified as well as its correct classification

The hypothesis h_{MDL} under the encoding C_1 and C_2 is just the one that minimizes the sum of these description lengths.

- MDL principle provides a way for trading off hypothesis complexity for the number of errors committed by the hypothesis
- MDL provides a way to deal with the issue of overfitting the data.
- Short imperfect hypothesis may be selected over a long perfect hypothesis.

MOD 4

BAYESIAN BELIEF NETWORKS

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

- This assumption dramatically reduces the complexity of learning the target function

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 allow stating conditional independence assumptions that apply to subsets of the variables

Notation

- Consider an arbitrary set of random variables $Y_1 \ldots Y_n$, where each variable Yi can take on the set of possible values $V(Y_i)$.
  The joint space of the set of variables Y to be the cross product $V(Y_1) \times V(Y_2) \times \ldots V(Y_n)$.
- 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.

Conditional Independence

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

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

$$x_i \in V(X),\ y_j \in V(Y),\ \text{and}\ z_k \in V(Z).$$

The above expression is written in abbreviated form as

$$P(X \mid Y, Z) = P(X \mid Z)$$

Conditional independence can be extended to sets of variables. The set of variables $X_1 \ldots X_l$ is conditionally independent of the set of variables $Y_1 \ldots Y_m$ given the set of variables $Z_1 \ldots Z_n$ if

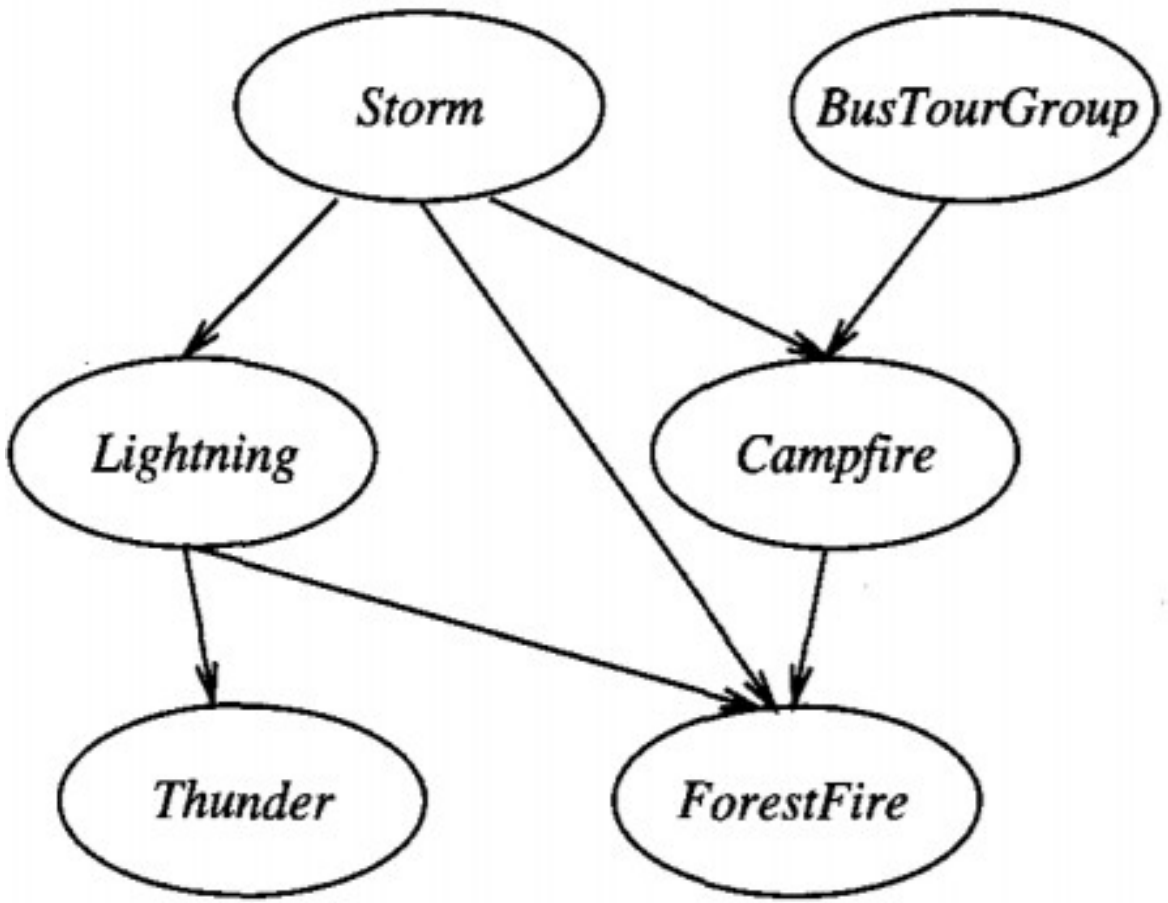
$$P(X_1 \ldots X_l | Y_1 \ldots Y_m, Z_1 \ldots Z_n) = P(X_1 \ldots X_l | Z_1 \ldots Z_n)$$

The naive Bayes classifier assumes that the instance attribute A_1 is conditionally independent of instance attribute A_2 given the target value V. This allows the naive Bayes classifier to calculate $P(A_1, A_2 \mid V)$ as follows,

$$\begin{aligned} P(A_1, A_2|V) &= P(A_1|A_2, V)P(A_2|V) \\ &= P(A_1|V)P(A_2|V) \end{aligned}$$

Representation

A Bayesian belief network represents the joint probability distribution for a set of variables. Bayesian networks (BN) are represented by directed acyclic graphs.



	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

The Bayesian network in above figure represents the joint probability distribution over the boolean variables *Storm, Lightning, Thunder, ForestFire, Campfire,* and *BusTourGroup*

A Bayesian network (BN) represents the joint probability distribution by specifying a set of *conditional independence assumptions*

- BN represented by a directed acyclic graph, together with sets of local conditional probabilities
- Each variable in the joint space is represented by a node in the Bayesian network
- The network arcs represent the assertion that the variable is conditionally independent of its non-descendants in the network given its immediate predecessors in the network.
- A ***conditional probability table*** **(CPT)** is given for each variable, describing the probability distribution for that variable given the values of its immediate predecessors

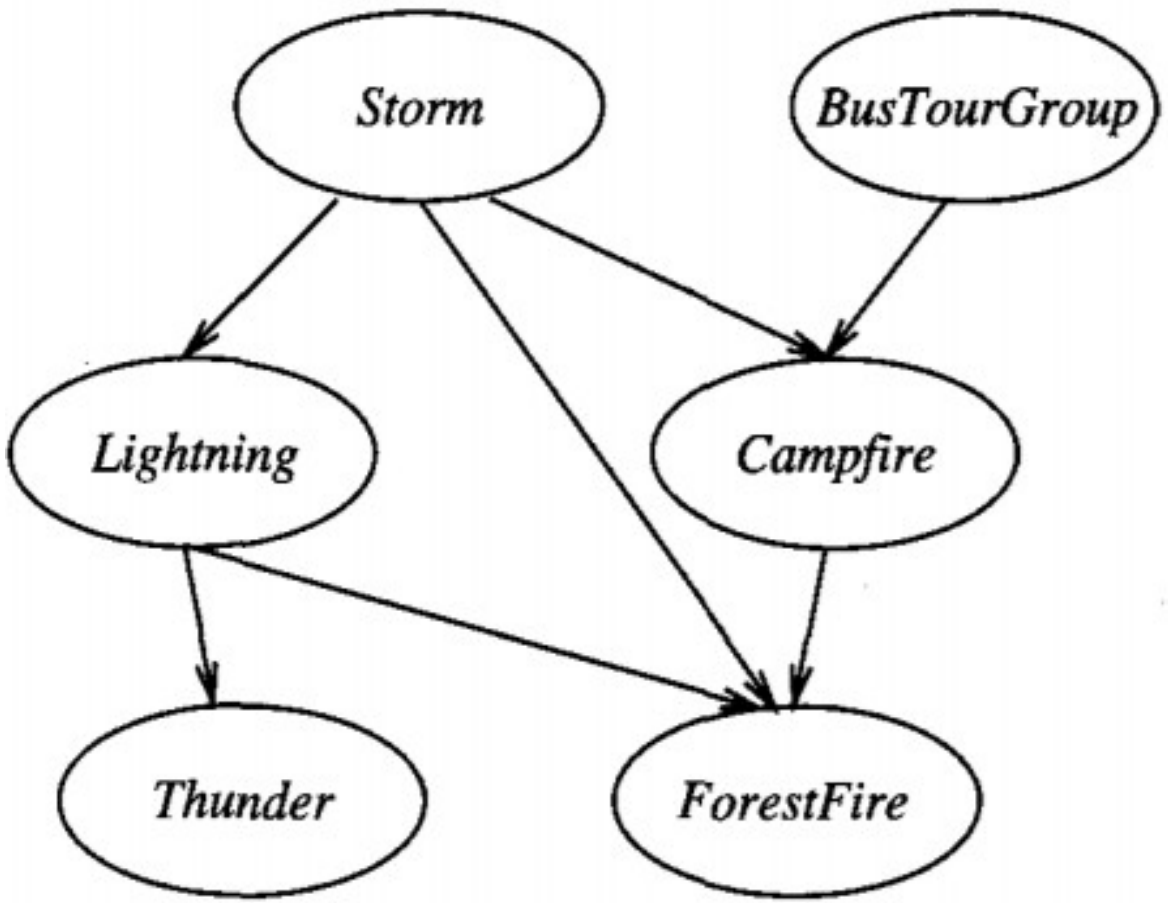
The joint probability for any desired assignment of values ($y_1, \ldots, y_n$) to the tuple of network variables ($Y_1 \ldots Y_m$) can be computed by the formula

Where, Parents(Y_i) denotes the set of immediate predecessors of y_i in the network.

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

Example:

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



	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

This means that once we know the value of the variables ***Storm*** and ***BusTourGroup***, the variables ***Lightning*** and ***Thunder*** provide no additional information about ***Campfire***
The conditional probability table associated with the variable ***Campfire.*** The assertion is

$$P(\text{Campfire} = \text{True} \mid \text{Storm} = \text{True}, \text{BusTourGroup} = \text{True}) = 0.4$$

Inference

- Use a Bayesian network to infer the value of some target variable (e.g., ForestFire) given the observed values of the other variables.
- Inference can be straightforward if values for all of the other variables in the network are known exactly.
- 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.
- An arbitrary Bayesian network is known to be NP-hard

Learning Bayesian Belief Networks

Effective algorithms can be considered for learning Bayesian belief networks from training data by considering several different settings for learning problem

- 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.
  - 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 and estimate the conditional probability table entries
  - 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. The learning problem can be compared to learning weights for an ANN.

Gradient Ascent Training of Bayesian Network

The gradient ascent rule which maximizes P(D|h) by following the gradient of ***ln P(D|h)*** with respect to the parameters that define the conditional probability tables of the Bayesian network.

Let w_{ijk} denote a single entry in one of the conditional probability tables. In particular w_{ijk} denote the conditional probability that the network variable Y_i will take on the value y_i, given that its immediate parents U_i take on the values given by u_{ik}.

The gradient of ***ln P(D|h)*** is given by the derivatives for each of the w_{ijk}. As shown below, each of these derivatives can be calculated as

$$\frac{\partial \ln P(D|h)}{\partial w_{ijk}}$$

$$\frac{\partial \ln P(D|h)}{\partial w_{ij}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_i = u_{ik}|d)}{w_{ijk}}$$

equ(1)

Derive the gradient defined by the set of derivativesfor all ***i, j,*** and ***k***. Assuming the training examples ***d*** in the data set D are drawn independently, we write this derivative as

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d)$$

$$= \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial w_{ijk}}$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial w_{ijk}}$$

We write the abbreviation $P_h(D)$ to represent $P(D|h)$.

This last step makes use of the general equality $\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$. We can now introduce the values of the variables Y_i and $U_i = Parents(Y_i)$, by summing over their possible values $y_{ij'}$ and $u_{ik'}$.

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}|u_{ik'}) P_h(u_{ik'})$$

This last step follows from the product rule of probability . Now consider the rightmost sum in the final expression above. Given that $w_{ijk} \equiv P_h(y_{ij}|u_{ik})$, the only term in this sum for which $\frac{\partial}{\partial w_{ijk}}$ is nonzero is the term for which $j' = j$ and $i' = i$. Therefore

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) P_h(y_{ij}|u_{ik}) P_h(u_{ik})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) w_{ijk} P_h(u_{ik})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d|y_{ij}, u_{ik}) P_h(u_{ik})$$

Applying Bayes theorem to rewrite $P_h(d|y_{ij}, u_{ik})$, we have

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik}|d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})}$$

$$= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})}$$

$$= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{P_h(y_{ij}|u_{ik})}$$

$$= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}} \qquad \text{equ (2)}$$

Thus, we have derived the gradient given in Equation (1). There is one more item that must be considered before we can state the gradient ascent training procedure. In particular, we require that as the weights w_{ijk} are updated they must remain valid probabilities in the interval [0,1]. We also require that the sum $\sum_j w_{ijk}$ remains 1 for all i, k. These constraints can be satisfied by updating weights in a two-step process. First we update each w_{ijk} by gradient ascent

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}}$$

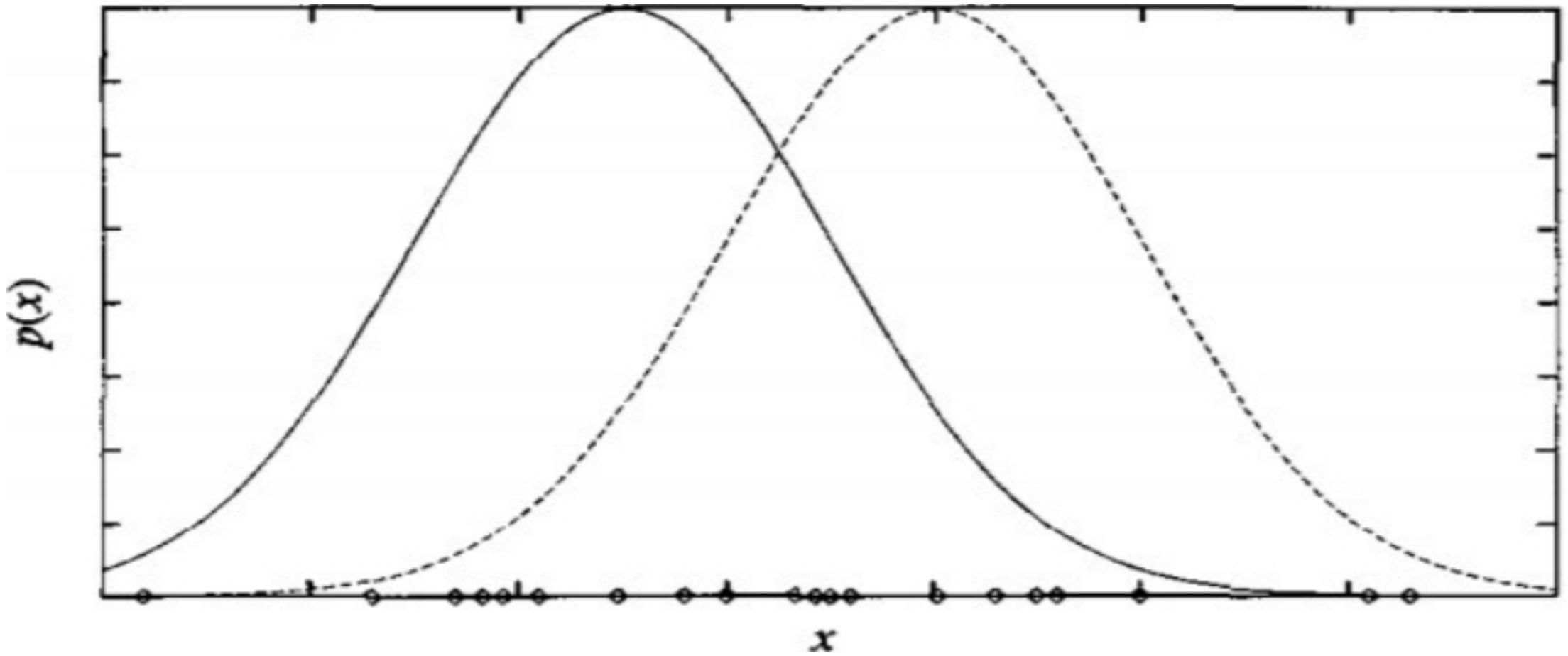
where η is a small constant called the learning rate. Second, we renormalize the weights w_{ijk} to assure that the above constraints are satisfied.
this process will converge to a locally maximum likelihood hypothesis for the conditional probabilities in the Bayesian network.

THE EM ALGORITHM

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.

Estimating Means of k Gaussians

- 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.


p(x)
x

- This problem setting is illustrated in Figure for the case where k = 2 and where the instances are the points shown along the x axis.
- 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.

To simplify, consider the special case

- The selection of the single Normal distribution at each step is based on choosing each with uniform probability
- Each of the k Normal distributions has the same variance σ^2, known value.

The learning task is to output a hypothesis h = (μ_1 , . . . ,μ_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).

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

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

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad (2)$$

- Our problem here, however, involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution.

  Consider 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

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

Here x_i is the observed variable in the description of the instance, and z_{il} and z_{i2} are hidden variables.

If the values of z_{il} and z_{i2} were observed, we could use following Equation to solve for the means p_1 and p_2

- Because they are not, we will instead use the EM algorithm

EM algorithm

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.

Let us examine how both of these steps can be implemented in practice. Step 1 must calculate the 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

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

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

Thus the first step is implemented by substituting the current values $\langle \mu_1, \mu_2 \rangle$ and the observed x_i into the above expression.

In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$. maximum likelihood hypothesis in this case is given by

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