# MODULE – 4

# BAYESIAN LEARNING

## 4.1. INTRODUCTION

Bayesian reasoning provides a probabilistic approach to inference. It assumes that 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. It is important to machine learning because it provides a quantitative approach to weighing the evidence supporting alternative hypotheses.

Bayesian learning methods are *relevant* to our 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 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 include:*

- 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 (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").
- 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*

1. Bayesian methods 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.
2. The significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses). In certain specialized situations, this computational cost can be significantly reduced.

## 4.2. BAYESIAN THEOREM

In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D. 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.

To define Bayes theorem precisely, let us first introduce a little notation.

- **P(h)** → initial probability that hypothesis h holds, before we have observed the training data. 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.
- **P(D)** → **prior probability** that **training data D** will be observed
- P(D|h) → probability of observing data D given some world in which hypothesis h holds.

In general, we write P(x|y) to denote the probability of x given y. 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 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)."***

**Bayesian Theorem:** $$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \quad \textbf{---(1)}$$

Here, 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.

**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 ∈ 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. We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis. More precisely, we will say that **hMAP** is a MAP hypothesis provided,

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

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

$$\equiv \underset{h \in H}{argmax} P(D|h)P(h) \quad \text{---(2)}$$

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

**Maximum Likelihood (ML) Hypothesis**

In some cases, we will assume that every hypothesis in H is equally probable a priori ( P(hi) = P(hj) for all hi and hj in H). In this case we can further above equation and need only consider the term P(D|h) to find the most probable hypothesis. **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, $h_{ML}$.

$$h_{ML} \equiv \underset{h \in H}{argmax} P(D|h) \textbf{---(3)}$$

In order to make clear the connection to machine learning problems, we introduced Bayes theorem above by referring to the data D as training examples of some target function and referring to H as the space of candidate target functions

**Summary of Basic Probability Formulae**

- *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

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

An Example

To illustrate Bayes rule, 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*.

The available data is from a particular laboratory test with two possible outcomes:

⊕ (positive) and

⊖ (negative).

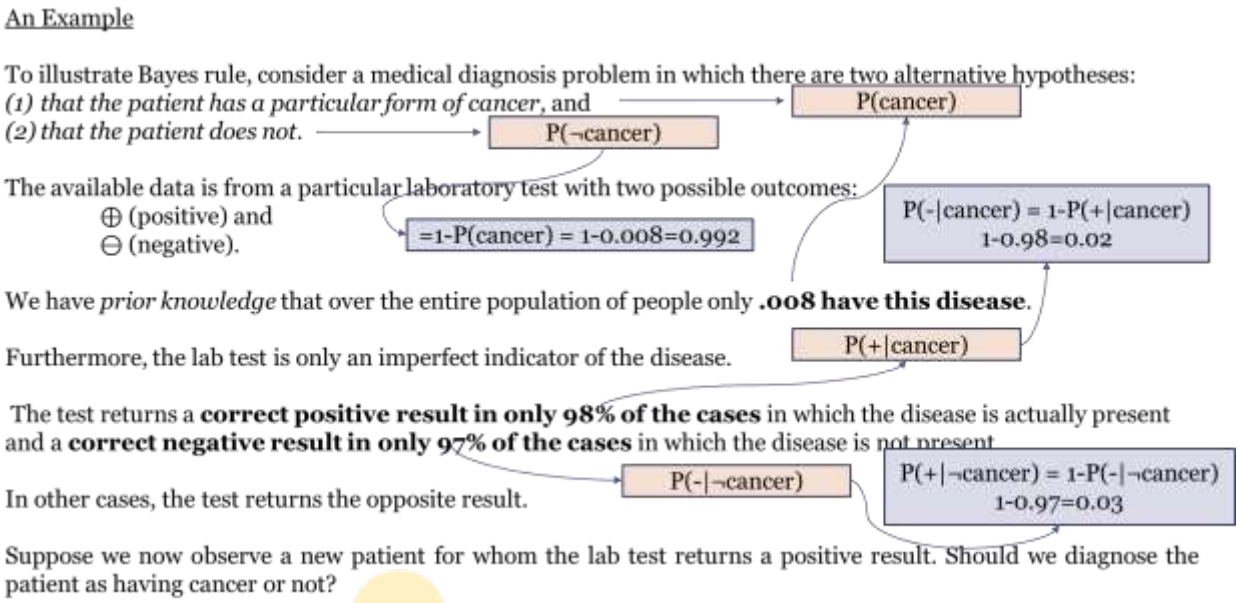
We have *prior knowledge* that over the entire population of people only .008 have this disease.

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.

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?




**NOTE: This figure is for the reference to obtain values.**

***Solution***

The given situation can be summarized by the following probabilities:

P(cancer) = 0.008 P(¬cancer) = 0.992

P(+|cancer) = 0.98 P(-|cancer) = 0.02

P(-|¬cancer) = 0.97 P(+|¬cancer) = 0.03

The maximum a posteriori hypothesis can be found using Equation (2):

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

D → + → true positive and false positive

$h_{cancer}$= P(+|cancer)P(cancer) = 0.98 ×0.008 = 0.0078

$h_{cancer}$= P(+|¬cancer)P(¬cancer) = 0.03 × 0.992 = **0.0298**

OUTPUT → $h_{MAP}$ = ¬*cancer* because false positive has the highest values.

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

$$P(cancer|+) = \frac{TP}{TP+FP} = \frac{0.0078}{0.0078+0.0298} = 0.21$$

$$P(\neg cancer|+) = \frac{FP}{TP+FP} = \frac{0.0298}{0.0078+0.0298} = 0.79$$

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(+). Although P(+) was not provided directly as part of the problem statement, we can calculate it in this fashion because we know that P(cancer|+) and P(¬cancer|+) must sum to 1.

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.

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.

## 4.3. BAYESIAN THEOREM AND CONCEPT LEARNING

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

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

WHAT IS UNDER DISCUSSION?

Consider a brute-force Bayesian concept learning algorithm, then compare it to concept learning algorithms

WHAT IS NOTICED?

Under certain conditions several algorithms output the same hypotheses as brute-force Bayesian algorithm

**Brute-Force Bayes Concept Learning**

Consider a finite hypothesis space *H* defined over the instance space *X*

TASK→ learn a target concept $c : X \rightarrow \{0,1\}$

Given→ Sequence of training examples, $\langle\langle x_1, d_1\rangle \ldots \langle x_m, d_m\rangle\rangle$

Where, $x_i$ –instance from *X*

$d_i$ → target value of $x_i$ → $d_i = c(x_i)$

Assumption: Sequence of instances $\langle x_1 \dots x_m \rangle$ is held **fixed**.

→ D can be written as a sequence of target values→ $\langle d_1 \dots d_m \rangle$

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

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

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

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

The algorithm requires significant computation → Bayes Theorem is applied to each hypothesis in H for P(h|D)

- *To specify Learning problem*→ Specify what values to be used for P(h) and P(D|h)
- P(D) will be determined after P(h) and P(D|h)

**P(h)** and **P(D|h)** should be consistent with the following assumptions

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

Given the assumptions, 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* .**
- Since we assume the target concept is contained in H we should require that these prior
  probabilities sum to 1.

Therefore,

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

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

- P(D|h) is the probability of observing the target values D = $\langle d_1 \dots d_m \rangle$ for fixed set of instances $\langle x_1 \dots x_m \rangle$ given a world in which hypothesis ***h*** holds.
- Assumption → 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 $d_i \neq h(x_i)$
- Therefore,

$$P(D|h) = \begin{cases} 1 \; if \; d_i = h(x_i) for \; all \; d_i \; in \; D \\ 0 \qquad\qquad\qquad\qquad Otherwise \end{cases} \text{--(1)}$$

That is, the probability of data ***D*** given hypothesis ***h*** is 1if ***D*** is consistent with ***h,*** and 0 otherwise.

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

From Bayes Theorem

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

Case 1: *h* is inconsistent with the training data *D*

From (1)

*P(D|h)* is *0* when *h* is inconsistent with *D*.

Therefore,

$$P(h|D) = \frac{0.P(h)}{P(D)} = 0 \text{ if } h \text{ is inconsistent with D}$$

That is, the posterior probability of a hypothesis inconsistent with *D* is zero.

Case 2: *h* is consistent with the training data *D*

From (1)

*P(D|h)* is *1* when *h* is consistent with *D*.

Therefore,

$$P(h|D) = \frac{1.\frac{1}{|H|}}{P(D)}$$

$$= \frac{1.\frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|} \text{ if } h \text{ is consistent with } D$$

Where $VS_{H,D}$ is the subset of hypotheses from $H$ that are consistent with $D$ .

**How do we get $P(D) = \frac{|VS_{H,D}|}{|H|}$ ?**

- The sum over all hypotheses of $P(h|D)$ must be one
- The number of hypotheses from H consistent with $D$ is by definition $|VS_{H,D}|$

We can derive $P(D)$ from the theorem of total probability→hypotheses are mutually exclusive → $(\forall i \neq j)(P(h_i \wedge h_j) = 0)$

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

$$= \sum_{h_i \in VS_{H,D}} 1.\frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0.\frac{1}{|H|}$$

$$= \sum_{h_i \in VS_{H,D}} 1.\frac{1}{|H|}$$

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

Therefore,

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

Every consistent hypothesis is, therefore, a MAP hypothesis.

The evolution of probabilities associated with hypotheses is depicted schematically in Figure given below. Initially (Figure 6.1a) all hypotheses have the same probability. As training data accumulates (Figures 6.1b and 6.lc), the posterior probability for inconsistent hypotheses becomes zero while the total probability summing to one is shared equally among the remaining consistent hypotheses.

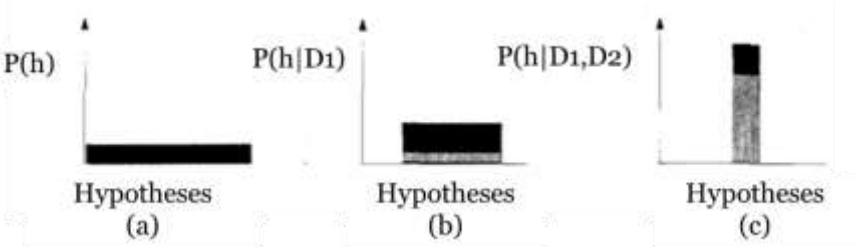



**Figure 1:** Evolution of posterior probabilities *P(hlD)* with increasing training data. *(a)* Uniform priors assign equal probability to each hypothesis. As training data increases first to *Dl* (b), then to $D_l$ A $D_2$ **(c),** the posterior probability of inconsistent hypotheses becomes zero, while posterior probabilities increase for hypotheses remaining in the version space

**MAP Hypothesis and Consistent Learners**

From the previous analysis

"every hypothesis consistent with *D* is a MAP hypothesis"

**Definition**

This implies a definition for ***consistent learners.***

"A learning algorithm is a *consistent learner* provided it outputs a hypothesis that commits zero errors over the training examples"

Therefore,

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

For examples,

Consider the concept learning algorithm FIND- S. We know that, FIND-S searches the hypothesis space H from specific to general hypotheses, outputting a maximally specific consistent hypothesis → it will output a MAP hypothesis under the probability distributions *P(h)* and *P(D|h)* .

Are there other probability distributions for ***P(h)*** and ***P(D1h)*** under which FIND-S outputs MAP hypotheses? Yes.

Find-S output → MAP hypothesis relative to any prior probability distribution that

favors more specific hypotheses.

Consider H → probability distribution $P(h)$ over H that assigns $P(h_1) \geq P(h_2)$ if $h_1$ is more specific than $h_2$ . It can be shown that FIND-S outputs a MAP hypothesis assuming the prior distribution H and the same distribution $P(D|h)$.

*From concept learning,*

An *Inductive bias* of a learning algorithm to be the set of assumptions B sufficient to *deductively* justify the inductive inference performed by the learner.

Consider Candidate Elimination algorithm with an assumption that the target concept *c* is included in the hypothesis space H. Its output follows deductively from its inputs plus this implicit inductive bias assumption.

Alternative,

→ Bayesian interpretation →model inductive bias by an equivalent ***probabilistic reasoning*** system based on Bayes theorem.

→ Implicit assumptions

- Prior probabilities over H are given by the distribution $P(h)$, and
- Strength of data in rejecting or accepting a hypothesis is given by $P(D|h)$

## 4.4. MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

**Learning Problem:** learning a continuous-valued target function

**Bayesian Analysis**

*"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.*"

**Problem Setting**

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

$h : X \rightarrow R$

*L* → should learn an unknown target function f :X → *R* drawn from H

Consider the following problem.

Given,

Set of *m* training examples where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution . → $\langle x_i, d_i \rangle$ where

$d_i = f(x_i) + e_i$.

$f(x_i)$ → noise-free value of the target function

$e_i$ → random variable representing the noise

**Assumption**

- The values of the $e_i$ are drawn independently**.**
- They are distributed according to a Normal distribution with zero mean.

**Task of the Learner**

Output a maximum likelihood hypothesis, or, equivalently, a MAP hypothesis assuming all hypotheses are equally probable a priori .

**Example:** Learning a linear function, though the analysis applies to learning arbitrary real-valued functions.

Figure 6.2 illustrates the whole scenario. Here 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.

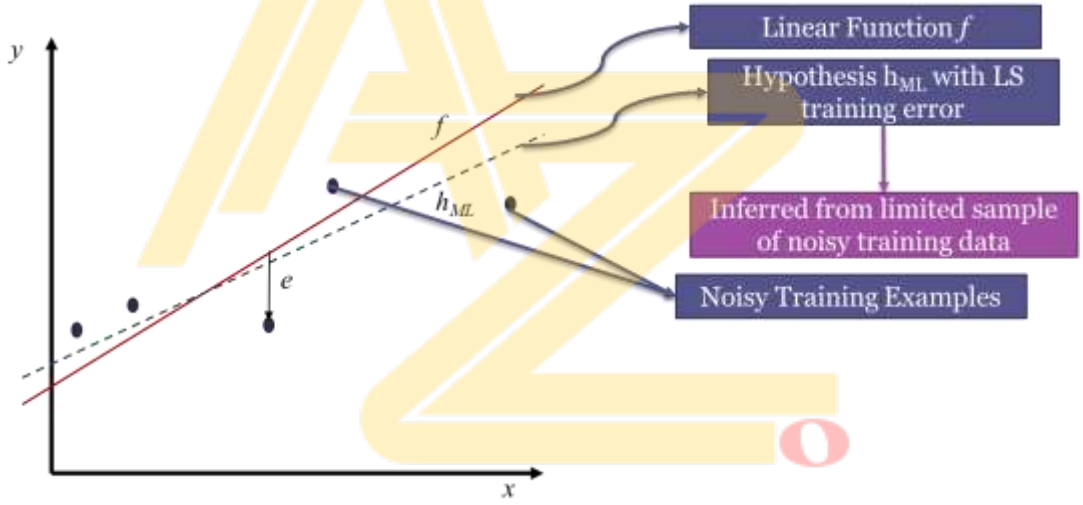



**Figure 2:** Learning a real-valued function. The target function f corresponds to the solid line. The training examples ***(xi,di***) are assumed to have Normally distributed noise ***ei*** with zero mean added to the true target value f ***(xi).***The dashed line corresponds to the linear function that minimizes the sum of squared errors. Therefore, it is the maximum likelihood hypothesis $h_{ML}$ , given these five training examples.

**To show that,**

Last-squared error hypothesis is, in fact, the maximum likelihood hypothesis

**Probability densities:**

- For probabilities over continuous variables such as *e*.

Requirement for our problem→ Total probability over all possible values of the random variable must sum to one.

*Why is Probability Density needed?*

Reason→ For continuous variable it is difficult to achieve by assigning a finite probability to each of the infinite set of possible values for the random variable.

*What is done with Probability Densities?*

Apply and expect the integral of this probability density over all possible values to be one.

Representation:

p→ Probability Density Function

P→ Finite Probability

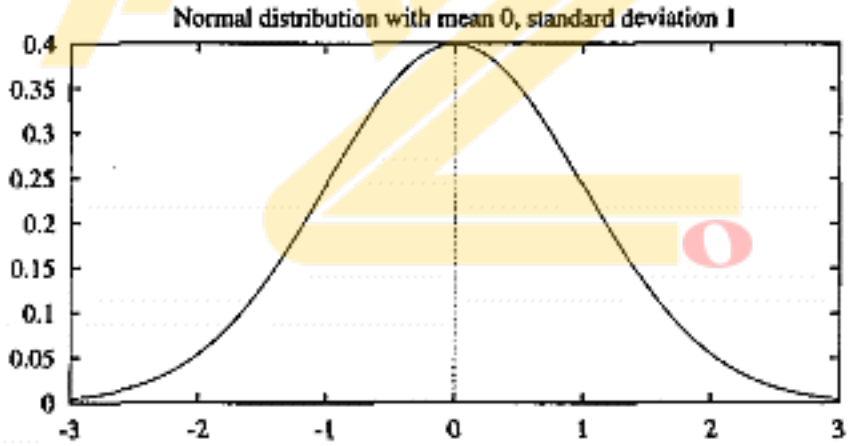
**Definition:**

The probability density ***p(x₀)*** is the limit as $\epsilon$ goes to zero, of $\frac{1}{\epsilon}$ times the probability that *x* will take the value in the interv$[x_0, x_0+ \epsilon]$ .

Probability density function:

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

**Normal Distribution:** Random noise variable e is generated by a Normal probability distribution. A Normal distribution (also called a Gaussian distribution) is a smooth, bell-shaped distribution that can be completely characterized by its mean μ and its standard deviation σ. It can be 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] = μ***
- The variance of X, Var(X), is ***Var(X) = σ²***
- The standard deviation of X, $\sigma_x$, is ***$\sigma_x$ = σ***

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.

**Prove:** Maximum likelihood hypothesis $h_{ML}$ minimizes the sum of the squared errors between the observed training values $d_i$ and the hypothesis predictions $h(x_i)$

**Proof:** From equation (3) we have

Deriving the maximum likelihood hypothesis starting with our earlier definition of $h_{ML}$, but using lower case p to refer to the probability density

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

Assumptions

Fixed set of training instances $\langle x_1 \dots x_m \rangle$

D→ corresponding sequence of target values $D=\langle d_1 \dots d_m \rangle$

$$d_i = f(x_i) + e_i$$

Training examples are mutually independent given h → P(D|h) → product of various $p(d_i|h)$

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

$e_i$ obeys Normal distribution with zero mean and unknown variance $\sigma^2$ .

$d_i$ must also obey Normal distribution with variance $\sigma^2$ centered around the true target value $f(x_i)$ rather than zero.

Hence,

$p(d_i|h)$ can be written as a Normal distribution with variance $\sigma^2$ and mean $\mu = f(x_i)$ .

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 $p = f(x_i) = h(x_i)$, yielding

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

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

Rather than maximizing the above complicated expression we shall choose to maximize its (less complicated) logarithm

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

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

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

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

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

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

Above 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 $h(x_i)$.

**Limitations:** 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.

## 4.5. MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

*Proven:*

Maximum likelihood hypothesis is the one that minimizes the sum of squared errors over the training examples.

*NOW!!*

Derive an analogous criterion for a second setting that is common in neural network learning→ learning to predict probabilities.

Setting!!

Learn a nondeterministic (probabilistic) function ***f*** : X -+ {0,1}, which has two discrete output values

Example 1,

X→ medical patients in terms of their symptoms

f(x) → Target Function →= 1 → if patient survives

= 0 → if not.

Example 2,

X→ loan applicants in terms of their past credit history

f(x) → Target Function →= 1 → if the applicant successfully repays their next loan

= 0 → if not.

Here, *f* can be expected to be probabilistic

***PROBABILISTIC f !!!***

For example,

Among a collection of patients exhibiting the same set of observable symptoms, we might find that 92% survive, and 8% do not. → the output of the target function *f(x)* is a probabilistic function of its input.

LEARNING PROBLEM!!!

Learn a neural network (or other real-valued function approximator) whose output is the ***probability*** that ***f***(x)=1

i.e., to learn the target function, f' : X → [0,1] such that f'(x) = P(f(x)=1).

From the examples,

f'(x) = 0.92

Probabilistic function ***f***(x) will be equal to 1 in 92% of cases and equal to 0 in the remaining 8%.

**How can we learn f' using, say, a neural network?**

Solution

- first collect the observed frequencies of 1's and 0's for each possible value of x
- then train the neural network to output the target frequency for each x

Further to be Proven!!

It is possible to train a neural network directly from the observed training examples of ***f,*** yet still derive a maximum likelihood hypothesis for *f'* .

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

To answer → first obtain an expression for P(D|h).

Assumptions

- D→ training Data → $\{\langle x_1, d_1\rangle \ldots \langle x_m, d_m\rangle\}$
  - $d_i$ → observed 0 or 1 for $f(x_i)$
- Both $x_i$ and $d_i$ are random variables
- Each training example is drawn independently

Therefore, we can write

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

- Probability of encountering any particular instance xi is independent of the hypothesis h
    - Example→ probability that our training set contains a particular *patient* $x_i$ is independent of our hypothesis about survival rates
    - *survival* $d_i$ of the patient depends strongly on *h*

When x is independent of h we can rewrite the above expression as

$$P(D|h) = \prod_{i=1}^{m} P(d_i|h, x_i)P(x_i) \ldots(6.8)$$

What is the probability $P(d_i|h,x_i)$ of observing $d_i = 1$ for a single instance $x_i$, given a world in which hypothesis *h* holds?

We know that , h → computes this probability

Therefore,

$$P(d_i = 1 \,|h, x_i) = h(x_i)$$

Hence,

$$P(d_i|h, x_i) = \begin{cases} h(x_i) \; if \; d_i = 1 \\ 1 - h(x_i) \; if \; d_i = 0 \end{cases} \ldots.(6.9)$$

In order to substitute for P(D|h) in (8), let us first "re-express it in a more mathematically manipulable form, as

$$P(d_i|h, x_i) = \mathrm{h}(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \ldots(6.10)$$

It is easy to verify that the expressions in Equations (6.9) and (6.10) are equivalent. Notice that when $d_i = 1$, the second term from Equation (6.10), $(1 - h(x_i))^{1-d_i}$, becomes equal to 1. Hence $P(d_i = 1|h, x_i) = h(x_i)$, which is equivalent to the first case in Equation (6.9). A similar analysis shows that the two equations are also equivalent when $d_i = 0$.

We can use Equation (6.10) to substitute for $P(d_i|h, x_i)$ in Equation (6.8) to obtain

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i) \qquad (6.11)$$

Now 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

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \qquad (6.12)$$

The expression on the right side of Equation (12) can be seen as a generalization of the **Binomial distribution**. The expression in Equation (12) describes the probability that flipping each of m distinct coins will produce the outcome (dl . . .dm), assuming that each coin xi has probability h(xi) of producing a heads. Note the Binomial distribution is similar, but makes the additional assumption that the coins have identical probabilities of turning up heads (i.e., that h(xi) = h(xj), for every i, j). In both cases we assume the outcomes of the coin flips are mutually independent-an assumption that fits our current setting.

It is easy to work with log of the likelihood, we get

$$h_{ML} = \underset{h\epsilon H}{argmax} \sum_{i=1}^{m} d_i ln\, h(x_i) + (1 - d_i) \ln\big(1 - h(x_i)\big) \quad \ldots(6.13)$$

Equation (6.13) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting.

$\sum_{i=1}^{m} d_i ln\, h(x_i)$ → Similar to Entropy expression and hence named as ***Cross Entropy***.

**Gradient Search to Maximize Likelihood in a Neural Net**

Let $d_i lnh(x_i) + (1 - d_i) \ln\big(1 - h(x_i)\big)$ be represented by G(h,D)

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

From ANN 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.

Therefore, he partial derivative of G(h, D) with respect to weight $w_{jk}$ from input k to unit j is

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

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

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

$$\frac{\partial G(h|D)}{\partial w_{jk}} = \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}} \quad \text{---(1)}$$

Considering that neural network is constructed from a single layer of sigmoid units, we get

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(h(x_i))x_{ijk} = h(x_i)(1 - h(x_i))x_{ijk}$$---(2)

Where,

$x_{ijk}$→ k[th] input to unit j for the i[th] training example

$\sigma'(h(x_i))$→ Derivative of sigmoid squashing function

Substitute (2) in (1)

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

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

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

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

Where,

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

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.

Re-express it using the current notations:

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

Where,

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

CONCLUSION

The rule that minimizes sum of squared error seeks the maximum likelihood hypothesis under the assumption that the training data can be modeled by normally distributed noise added to the target function value.

The rule that minimizes cross entropy seeks the maximum likelihood hypothesis under the assumption that the observed Boolean value is a probabilistic function of the input instance.

## 4.6. MINIMUM DESCRIPTION LENGTH PRINCLIPLE

**Occam's razor**

Choose the shortest explanation for the observed data.

Bayesian perspective → Minimum Description Length (MDL) principle

Motivation→ interpreting the definition of $h_{MAP}$ in the light of basic concepts from information theory.

Consider,

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

$h_{MAP}$ can be equivalently expressed in terms of maximizing the log,

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

Or alternatively minimize the above quantity

$$h_{MAP} = \underset{h \in H}{argmin} -log_2 P(D|h) - log_2 P(h) \qquad (16)$$

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

Consider the problem of designing a code to transmit messages drawn at random, where the probability of encountering message ***i*** is $p_i$.

***What is needed?***

Most compact code → the code that minimizes the expected number of bits we must transmit in order to encode a message drawn at random.

***How?***

Assign shorter codes to messages that are more probable.

***Existing Proof!!***

Shannon and Weaver (1949)→ Optimal code (i.e., the code that minimizes the expected message length) assigns $-\log_2 p_i$ bitst to encode message ***i***.

Let, number of bits required to encode message ***i*** using code ***C*** → *description length of message i with respect to C* → $L_c(i)$.

Consider Eq. 16.

Interpret the above expression with coding theory.

- $-\log_2 P(h)$ → description length of ***h*** under the optimal encoding for the hypothesis space H.

This is the size of the description of hypothesis ***h*** using this optimal representation.

$$L_{C_H}(h) = -\log_2 P(h)$$

$C_H$ → Optimal code for hypothesis space H.

- $-\log_2 P(D|h)$ → description length of the training data ***D*** given hypothesis ***h,*** under its optimal encoding.

$$L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$$

$C_{D|h}$→ optimal code for describing data ***D*** assuming that both the sender and receiver know the hypothesis ***h***

Rewrite (16)

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

CONCLUSION:

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

Consider that the codes $C_1$ and $C_2$ to represent the hypothesis and the data given the hypothesis, we can state the MDL principle as

**Minimum Description Length principle:** Choose $h_{MDL}$ where

$$h_{MDL} = \underset{h}{argmin} L_{C_1}(h) + L_{C_2}(D|h)$$

If we choose C1 to be the optimal encoding of hypotheses $C_H$, and if we choose C2 to be the optimal encoding $C_{D|h}$, then $h_{MDL} = h_{MAP}$ .

Example,

To 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?

$C_1$ → choose some obvious encoding of decision trees such that description length grows with the number of nodes in the tree and with the number of edges

How shall we choose the encoding C2 of the data given a particular decision tree hypothesis?

Consider a sequence of instances $<x_1 , \ldots , x_m>$ known to both transmitter and receiver. Therefore, transmit only the classification $<f(x_1),\ldots, f(x_m)>$

If the he training classifications $<f(x_1),\ldots, f(x_m)>$ 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 → 0.

**Consider, examples are misclassified by h**

For each misclassification transmit a message that identifies which example is misclassified along with its correct classification.

Misclassified classification→ $\log_2 m$

Correct classification → $\log_2 k$

The hypothesis $h_{MDL}$ under the encoding C1 and C2 is just the one that minimizes the sum of these description lengths

Therefore MDL principle provides a way of **trading off hypothesis complexity for the number of errors committed by the hypothesis.**

It might select a shorter hypothesis that makes a few errors over a longer hypothesis that perfectly classifies the training data. → overfitting addressed.

**Conclusion on MDL!!**

Does this prove once and for all that short hypotheses are best?

NO

What is shown?

If a representation of hypotheses is chosen so that the size of hypothesis h is $-\log_2 P(h)$, and if a representation for exceptions is chosen so that the encoding length of D given h is equal to $-\log_2 P(D|h)$, *then* the MDL principle produces MAP hypotheses.

There is no reason to believe that the MDL hypothesis relative to *arbitrary* encodings C1 and C2 should be preferred.

## 4.7. BAYES OPTIMAL CLASSIFIER

What is the most probable ***hypothesis*** given the training data?

What is the most probable ***classification*** of the new instance given the training data?

It is possible to answer better than with MAP hypothesis.

*How to develop more intuitions?*

Consider H having $h_1$, $h_2$, and $h_3$.

Let their posterior probabilities be 0.4, 0.3. and 0.3.

MAP hypothesis → $h_1$.

Consider an instance *x* → classified +ve by $h_1$

→ classified –ve by $h_2$, and $h_3$

- That is, the probability that *x* is positive is 0.4 and probability that *x* is negative is 0.6.
- **The most probable classification is different from MAP classification.**
- The most probable classification of the new instance is obtained by combining the predictions of all hypotheses, weighted by their posterior probabilities**.**
- If the possible classification of a new example can take on any value $v_j$ from set *V*,

then the probability $P(v_j|D)$ that the correct classification for the new instance is $v_j$

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

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

**Bayes Optimal Classification**

$$\underset{v_j \in V}{argmax} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) --(18)$$

Example,

The set of possible classifications of the new instance is $V=\{\oplus, \ominus\}$

$$P(h_1|D) = 0.4\,,\ P(\ominus|h_i) = 0, P(\oplus|h_1) = 1$$
$$P(h_2|D) = 0.3\,,\ P(\ominus|h_2) = 1, P(\oplus|h_2) = 0$$
$$P(h_3|D) = 0.3\,,\ P(\ominus|h_3) = 1, P(\oplus|h_3) = 0$$

Therefore,

$$\sum_{h_i \in H} P(\oplus|h_i)P(h_i|D) = 0.4$$
$$\sum_{h_i \in H} P(\ominus|h_i)P(h_i|D) = 0.6$$

And

$$\underset{v_j \in \{\oplus, \ominus\}}{argmax} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = \ominus$$

Any system that classifies new instances according to Equation (6.18) is called a ***Bayes optimal classifier,*** or **Bayes optimal learner.** Therefore, Bayes Optimal Classifier maximizes the probability that the new instance is classified correctly, given the available data, hypothesis space, and prior probabilities over the hypotheses.

## 4.8. GIBBS ALGORITHM

Bayes optimal classifier obtains the best performance that can be achieved from the given training data.

Disadvantage!!

It can be costly to apply.

**WHY??**

- It computes the posterior probability for every hypothesis in H
- then combines the predictions of each hypothesis to classify each new instance.

**ALTERNATIVE!!**

***GIBBS ALGORITHM***

**Definition:**

*Gibbs Algorithm*

1. Choose a hypothesis ***h*** from H at random, according to the posterior probability distribution over H.
2. Use ***h*** to predict the classification of the next instance ***x***.

Haussler et al. 1994 → it can be shown that under certain conditions the expected misclassification error for the Gibbs algorithm is at most twice the expected error of the Bayes optimal classifier

Implication for the concept learning problem

Consider,

- Uniform prior probability over H
- Target concepts are drawn with uniform distribution

Classification→ done with hypothesis drawn at random from $V_{H,D}$

***→ has expected error at most twice that of the Bayes optimal classifier.***

## 4.9. NAÏVE BAYES CLASSIFIER

Highly practical learning method→ Naïve Bayes Classifier (NBC)

Where can NBC be applied??

Learning tasks where

- Each instance *x* is described by a conjunction of attribute values
- The target function *f(x)* can take on any value from some finite set V.

Given

A set of training examples of target function

A new instance described by the tuple of attribute values <$a_1$, $a_2$, …..,$a_n$>

Task!!

To predict the target value, or classification for the new instance

Bayesian approach!!!

Assign the most probable target value, $V_{MAP}$, given the attribute values **<$a_1$,$a_2$ . . .$a_n$>** that describe the instance.

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

Rewrite the above expression using Bayes theorem

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

$$v_{MAP} = \underset{v_j \in V}{argmax} P(a_1, a_2, .., a_n | v_j) P(v_j) \text{--(6.19)}$$

Count the frequency of $v_j$ to get the value of $P(v_j)$

*How to estimate* $P(a_1, a_2, .., a_n | v_j)$*??*

Takes a larger set of examples to follow the procedure of $P(v_j)$

Problem→ The number of these terms is equal to the number of possible instances times the number of possible target values.

Naïve Bayes Classifier simplifies the assumption that the attribute values are conditionally independent given the target value.

Given,

The target value of the instance

Therefore,

The probability of observing the conjunction $a_1, a_2 .. .a_n$, is the product of the probabilities for the individual attributes: $P(a_1, a_2, .., a_n | v_j) = \prod_i P(a_i | v_j)$.

Substitute in (6.19)

$$v_{NB} = \underset{v_j \in V}{argmax} P(v_j) \prod_i P(a_i | v_j) \text{ --(6.20)}$$

### 4.9.1. An Illustrative Example

Apply the naive Bayes classifier to a concept learning problem→ Decision tree learning

| Example | 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 |

Use naive Bayes classifier and the training data from the table to classify the following novel instance

<Outlook =sunny, Temperature =cool, Humidity = high, Wind = strong>

**Task!!**

Predict the target value (yes or no) of the target concept PlayTennis for this new instance.

We know that ,

$$v_{NB} = \underset{v_j \in V}{argmax} P(v_j) \prod_i P(a_i|v_j) --(20)$$

Here,

$$v_{NB} = \underset{v_j \in \{yes,no\}}{argmax} P(v_j) P(Outlook = sunny|v_j) P(Temperature = cool|v_j) P(Humidity = high|v_j) P(Wind = strong|v_j)$$

--(6.21)

Requires 10 probabilities for estimating $v_{NB}$.

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

$$P(PlayTennis = yes) = \frac{9}{14} = 0.64$$

$$P(PlayTennis = no) = \frac{5}{14} = 0.36$$

Now estimate the conditional probabilities

$P(Outlook$ =sunny|PlayTennis=yes) = 2/9=0.22

$P(Outlook$ =sunny|PlayTennis=no) = 3/5=0.6

$P$(Temperature =cool|PlayTennis=yes) = 3/9=0.33

$P$(Temperature =cool|PlayTennis=no)=1/5=0.2

$P$(Humidity = high|PlayTennis=yes) = 3/9=0.33

$P$(Humidity = high|PlayTennis=no)=4/5=0.8

$P$(Wind = strong|PlayTennis=yes) = 3/9=0.33

$P$(Wind = strong|PlayTennis=no)=3/5=0.6

Substituting in (6.21) we get,

P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)

=0.64×0.22×0.33×0.33 ×0.33=0.0051

P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no)

=0.36 ×0.6 ×0.2 ×0.8 ×0.6

**=0.0207**

NBC applies the target value as ***no***.

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.

$$\frac{0.0207}{0.0051 + 0.0207} = 0.80$$

**4.9.1.1. Estimating Probabilities**

The probabilities were estimated by the fraction of times the event is observed to occur over the total number of opportunities.

Example,

***P(Wind*** = ***strong|Play Tennis*** = ***no)*** is calculated using $\frac{n_c}{n}$

where

n=5 (total number of training examples with target value = no)

$n_c$ = 3 number of examples in *n* samples where Wind=strong.

Problem !!

This provides a poor estimate if $n_c$ is too small.

Example,

Consider that P(Wind = strong|PlayTennis = no) = 0.8

Sample→ only 5 samples where PlayTennis = no.

Here, the most probable value for $n_c$ is 0.

Two difficulties

- $n_c$/n produces a biased underestimate of the probability.
- When $n_c$/n is zero, it will dominate the Bayes classifier if the future query contains ***Wind*** = ***strong***

***WHY??***

The quantity calculated in Equation ***(6.20)***requires multiplying all the other probability terms by this **zero** value.

**How to avoid this difficulty??**

Adopt a Bayesian approach to estimating the probability, using the m-estimate:

***m*-estimate of Probability**

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

Where,

p→ prior estimate of the probability

m→ constant→ equivalent sample size→ determines how heavily to weight *p* relative to the observed data

How to estimate *p* in the absence of other information?

Assume **uniform priors**.

NOTE: If $m = 0$ then *m*-estimate is equivalent to $n_c/n$.

## 4.10. BAYESIAN BELIEF NETWORKS

*Naïve Bayes Classifier* → Uses the assumption that the values of the attributes $a_1 \ldots a_n$, are **conditionally independent** given the target value v

OUTPUT→ optimal Bayes classification

*Bayesian Belief Networks* → Describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities

*Naïve Bayes Classifi*er → 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 network describes the probability distribution over a set of variables.

Example,

Consider,

An arbitrary set of random variables $Y_1 \ldots\ldots Y_n$.

For each $Y_i$ → $V(Y_i)$ → possible value

**Joint Space** of the set of variables $Y$ → cross product $\mathbf{V(Y_1) \times V(Y_2) \times \ldots\ldots \times V(Y_n)}$

Each item in the joint space → one of the possible assignments of values to the tuple of variables $< Y_1 \ldots\ldots Y_n >$

The probability distribution over the joint space is called as ***Joint Probability Distribution.*** It represents **p**robability for each of the possible variable bindings for the tuple $< Y_1 \ldots\ldots Y_n >$

***Bayesian Belief Networks → describes JDP for a set of variables.***

### 4.10.1. Conditional Independence

For 3 discrete-valued random variables X,Y, and Z,

"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 if

$$(\forall x_i, y_j, z_k) P(X = x_i | Y = y_j | Z = z_k) = P(X = x_i | Z = z_k) \quad \text{"----(1)}$$

Here $x_i \in V(X)$ , $y_i \in V(Y)$, and $z_i \in V(Z)$

- Eq.(1) Can be abbreviated as $P(X|Y,Z) = P(X|Z)$
- Definition for set of variables

"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

$$P(X_1 \ldots Xl | Y_1 \ldots Ym | Z_1 \ldots Zn) = P(X_1 \ldots Xl | Z_1 \ldots Zn)"$$

In NBC the instance attribute $A_1$ is conditionally independent of instance attribute $A_2$ given the target value V. Therefore

$$P(A_1, A_2|V) = P(A_1|A_2, V)P(A_2|V) \quad \text{---(2)}$$

$$= P(A_1|V)P(A_2|V) \quad \text{---(3)}$$

In Eq. (3) if $A_1$ is conditionally independent of $A_2$ given V, then by our definition of conditional independence then $P(A_1|A_2, V) = P(A_1|V)$

### 4.10.2. Representation

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

Example,

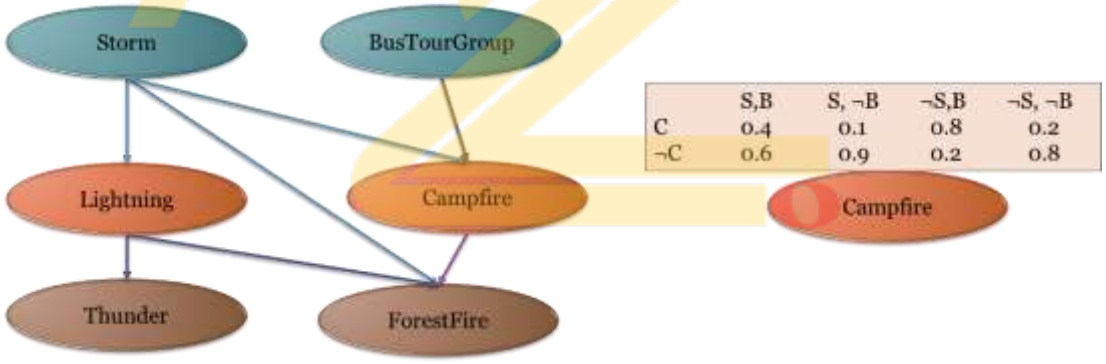



Figure: A Bayesian belief network. The network on the left represents a set of conditional independence assumptions. In particular, each node is asserted to be conditionally independent of its non-descendants, given its immediate parents. Associated with each node is a conditional probability table that specifies the conditional distribution for the variable given its immediate parents in the graph. The conditional probability table for the ***Campjire*** node is shown at the right, where ***Campjire*** is abbreviated to ***C, Storm*** abbreviated to ***S,*** and ***BusTourGroup*** abbreviated to ***B***

The joint probability for any desired assignment of values $\langle y_1, \ldots, y_n \rangle$ to the

tuple of network variables <$Y_1 \ldots Y_n$> can be computed by the formula

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

From the table for *Campfire* an assertion can be expressed as

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

***Full Joint Probability Distribution***

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

**Attractive feature** → BBN allows a convenient way to represent causal knowledge such as the fact that ***Lightning*** causes ***Thunder.***

### 4.10.3. Inference

Task→ To infer the value of some target variable (e.g., ***ForestFire)*** given the observed values of the other variables**.**

That is,

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.

→ Possible if values for all of the other variables in the network are known exactly.

*Possible cases for inference*

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).

*SOLUTION*

*Bayesian Network*

### 4.10.4. Learning Bayesian Belief Networks

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

Ans: Possible. Several settings can be done.

Example,

i. 1st setting

Network Structure may be given or need to be inferred

ii. 2nd setting

Network variables might be directly observable in each training

example, or some might be unobservable.

Case 1: Network structure is given and all the variables are fully observable in training examples

- Easy to learn conditional probability table

Case 2: Network structure is given and only some of the variable values are observable in the training data

- Learning is difficult
- Similar to learning weights for hidden units in ANN where input and output values are known.
- SOLUTION

  Gradient Ascent Procedure (GAP) that learns the entries in the conditional probability tables by Russel et. al. (1995)

  GAP → searches through a space of hypotheses that corresponds to the set of all possible entries for the conditional probability tables.

  → Maximizes probability ***P(D|h)*** of the observed training data D given the hypothesis h during gradient ascent.

### 4.10.5. Gradient Ascent Training of Bayesian Networks

By Russel et. al. (1995)

*What is done?*

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.

Consider,

$w_{ijk}$ → single entry in one of the conditional probability tables

→ conditional probability that the network variable $\boldsymbol{Y_i}$ will take on the value $\boldsymbol{y_{ij}}$ given that its immediate parents $\boldsymbol{U_i}$ take on the values given by $\boldsymbol{u_{ik}}$.

Example,

| | S,B | S, ¬B | ¬S,B | ¬S, ¬B |
|---|---|---|---|---|
| C | 0.4 | 0.1 | 0.8 | 0.2 |
| ¬C | 0.6 | 0.9 | 0.2 | 0.8 |

Consider $w_{ijk}$ is top right entry in the conditional probability table.

$\boldsymbol{Y_i}$ is the variable ***Campfire,***

$\boldsymbol{U_i}$ is the tuple of its parents ***(Storm,BusTourGroup),***

$\boldsymbol{y_{ij}}$ = ***True,*** and $\boldsymbol{u_{ik}}$ = ***(False, False).***

The gradient of P(D|h) is given by the derivatives $\frac{\partial lnP(D|h)}{\partial w_{ijk}}$ for each $w_{ijk}$.

Each can be calculated as,

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

---(4)

Example,

To calculate the derivative of In ***P(D|h)*** with respect to the upper rightmost entry in the table calculate the quantity ***P(Campfire*** = ***True, Storm*** = ***False, BusTourGroup*** = ***False|d)*** for each training example ***d*** in ***D***.

**Derivation of Eq. (4)**

Let $P_h(D)$ denote P(D|h)

To derive!

$$\frac{\partial lnP_h(D)}{\partial w_{ijk}}$$

Proof:

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

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

Since $\frac{\partial lnf(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$

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

Introduce the values of the variables $\boldsymbol{Y_i}$ and $\boldsymbol{U_i}$=***Parents(Y$_i$),***by summing over their possible values $\boldsymbol{y_{ij}}$**'** and $\boldsymbol{u_{ik}}$**'**

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

From product rule of probability

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

Consider the rightmost sum.

Given $w_{ijk} \equiv P_h(y_{ij}|u_{ik})$

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,

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

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

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

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

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

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

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

$$\frac{\partial lnP_h(d)}{\partial w_{ijk}} = \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}} \quad \text{--(5)}$$

Requirement,

- as the weights $w_{ijk}$ are updated they must remain valid probabilities in the interval [0,1]
- sum $\sum_j w_{ijk}$ remains 1 for all ***i, k***.

To satisfy the requirements!!

Update weights in a two-step process.

i. Update each $w_{ijk}$ by gradient ascent

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

ii. Renormalize the weights $w_{ijk}$ to assure that the above constraints are satisfied.

### 4.10.6. Learning the Structure of Bayesian Networks

Cooper and Herskovits (1992) → Bayesian scoring metric for choosing among alternative networks.

→ Also presented heuristic search algorithm called **K2** for learning network

structure when the data is fully observable.

→ K2

Performs a greedy search that trades off network complexity for accuracy over the training data

Experiment → given a set of 3,000 training examples generated at random from a manually constructed Bayesian network containing 37 nodes and 46 arcs.

→ given an initial ordering over the 37 variables that was consistent with the partial ordering of variable dependencies in the actual network.

OUTCOME → SUCCEEDED in reconstructing the correct Bayesian network structure almost exactly, with the exception of one incorrectly deleted arc and one incorrectly added arc

Spirtes et al. (1993)→ Constraint-based approaches to learning Bayesian network structure

→ Infer independence and dependence relationships from the data, and then use these relationships to construct Bayesian networks

## 4.11. The EM Algorithm

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

If some variable is sometimes observed and sometimes not, then the cases for which it has been observed is used to learn predict its values when it is not.

EM Algorithm→ Dempster et al. 1977

→ widely used approach to *learning in the presence of unobserved variables*.

→ used even for variables whose value is never directly observed, but requires the probability distribution.

→ has been used to train BBN

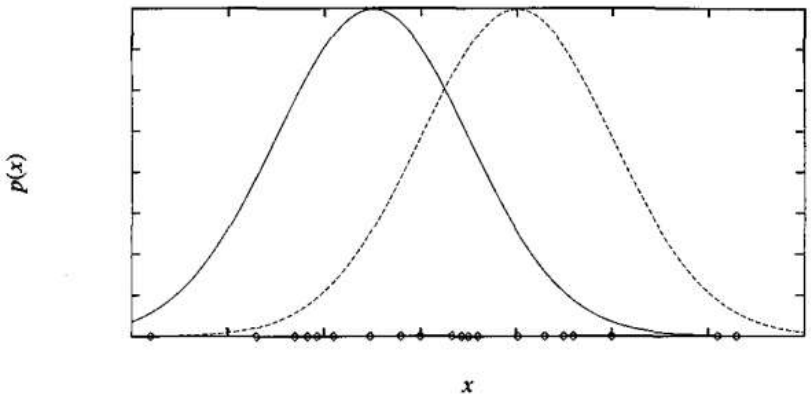
→ basis for many unsupervised clustering algorithms

→ basis for Baum-Welch forward-backward algorithm for learning Partially Observable Markov Models

### 4.11.1. Estimating the Means of *k* Gaussians

Example,

Given D→ data → set of instances generated by a probability distribution that is a mixture of *k* distinct Normal distributions




Here,

$k = 2$

instances→ points along *x*-axis

Each instance is generated using a two-step process.

i. One of the *k* Normal distributions is selected at random.

ii. A single random instance $x_i$ is generated according to this selected distribution

This process is repeated to generate a set of data points.

Consider a special case where, the selection of the single Normal distribution at each step is based on choosing each with uniform probability

Learning task→ output a hypothesis h = ($\mu_1$, .. .$\mu_k$) that describes the means of each of the k distributions.

Goal → a maximum likelihood hypothesis for these means; that is, a hypothesis h that maximizes p(D|h)

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

We know that,

$$\mu_{ML} = \underset{\mu}{argmin} \sum_{i=1}^{m} (x_i - \mu)^2 \quad \text{----(6)}$$

Also, the sum of squared errors is minimized by the sample mean.

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i \text{---(7)}$$

Problem in hand→ mixture of *k* different Normal distributions

→ cannot observe which instances were generated by which distribution

Let,

Full description of each instance be $\langle x_i, z_{i1}, z_{i2} \rangle$

Where,

$x_i$ → observed value of the $i^{th}$ instance

$z_{i1}$ and $z_{i2}$ → indicated which of the two Normal distributions was used to generate the value $x_i$

→ $z_{ij}$ = 1 if generated by $j^{th}$ Normal Distribution and 0 otherwise

Here,

$x_i$ → Observed Variable

$z_{i1}$ and $z_{i2}$ → Hidden Variables

If $z_{i1}$ and $z_{i2}$ were observed then Eq. (6) can be applied to solve for means $\mu_1$ and $\mu_2$ . Here we use EM Algorithm to find $\mu_1$ and $\mu_2$ .

EM Algorithm

Problem → *k*-Means

Task → Search for a maximum likelihood hypothesis by repeatedly re-estimating the expected values of the hidden variables $z_{ij}$ given its current hypothesis $< \mu_1 \ldots. \mu_k >$

→ Recalculate maximum likelihood hypothesis using these expected values for the hidden variables.

Procedure:

First initialize the hypothesis to $h = < \mu_1 , \mu_2 >$

Iteratively re-estimate *h* by repeating the following two steps until the procedure converges to a stationary value for h.

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

**Step 2:** Calculate a new maximum likelihood hypothesis $h' = < \mu_1' , \mu_2' >$ assuming the value taken on by each hidden variable $z_{ij}$ is its expected value $E[z_{ij}]$ calculated in Step 1.

Replace $h = < \mu_1 , \mu_2 >$ by $h' = < \mu_1' , \mu_2' >$ and iterate

*Step 1* must calculate the expected value of $z_{ij}$.

$E[z_{ij}]$ → probability that instance $x_i$ was generated by the $j^{th}$ Normal distribution.

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

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

**First step** is implemented by substituting the current values $< \mu_1 , \mu_2 >$ and the observed $x_i$ into the above expression.

**Second step** → use $E[z_{ij}]$ calculated in Step 1 to derive a new maximum likelihood hypothesis $h' = < \mu_1' , \mu_2' >$.

It is

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

---(8)

The above expression is similar to

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

---(7)

(7) → used to estimate μ for a single Normal distribution.

(8) → the weighted sample mean for $\mu_j$, with each instance weighted by the expectation $E[z_{ij}]$ that it was generated by the $j^{th}$ Normal distribution .

**Conclusion:**

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.

**Further proof:**

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 $< \mu_1 , \mu_2 >$ .

### 4.11.2. General Statement of EM Algorithm

What have we learnt in the previous session!!

EM algorithm for the problem of estimating means of a mixture of Normal distributions.

**EM algorithm** can be applied in many settings to **estimate set of parameters, $\theta$,** that include **probability distribution** given **only the observed portion of the full data** produced by this distribution.

Example,

$$\theta = \langle \mu_1, \mu_2 \rangle$$

The full data were the triples

$$\langle x_i, z_{i1}, z_{i2} \rangle$$

Here , only $x_i$ was observed.

Let

$X = \{x_1, ....., x_m\}$ → observed data in a set of m independently drawn instances

$Z = \{z_1, ....., z_m\}$ → unobserved data in these same instances

$Y = X \cup Z$ → full data

Unobserved Z → random variable whose probability distribution depends on the unknown parameters $\theta$ and on the observed data X.

Y → random variable because it is defined in terms of the random variable Z

What do we learn further??

**General form of EM Algorithm**

Notations:

$h$ → current hypothesized values of the parameters $\theta$

$h'$ → revised hypothesis that is estimated on each iteration of the EM algorithm

Learning task of EM algorithm!!

The EM algorithm searches for the MLH, ***h'***, by seeking the ***h'*** that maximizes E[ln P(Y|h')]

E[ln P(Y|h')] → taken over the probability distribution governing ***Y***

$Y$ → determined by the unknown parameters, $\theta$.

What does this expression signify??

P(Y|h') → likelihood of the full data Y given hypothesis h'.

Maximizing the logarithm of this quantity *In P(Y|h')* also maximizes *P(Y|h')*

Introduce the expected value *E[ln P(Y|h')]* because the full data *Y* is itself a random variable

Given,

Full data ***Y*** is a combination of the observed data X and unobserved data Z .

What to be done??

*Average* over the possible values of the unobserved Z, weighting each according to its probability.

Means??

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.

What is the probability distribution governing *Y ?*

*Usually unknown* → determined by the parameters θ

Therefore,

EM algorithm uses its current hypothesis *h* in place of the actual parameters θ to estimate the distribution governing *Y* .

To define!!

A function *Q(h'|h)* that gives *E[ln P(Y|h')]* as a function of *h'* under the assumption that *θ=h* and given the observed portion of *X* of the full data *Y*.

*Q(h'|h) = E[ln P(Y|h')|h,X]*

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.

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

---

### 4.11.3. Derivation of *k*-Means Algorithm

*k*-means problem→ to estimate the parameters $\theta = \langle \mu_1 \dots \mu_k \rangle$ that define the means of *k* Normal distributions.

Given,

X = $\{\langle x_{ij} \rangle\}$ → observed data

Z = $\{\langle z_{i1}, \dots, z_{ik} \rangle\}$ →indicates which of the *k* Normal distributions was used to generate $x_{ij}$.

To apply EM algorithm → derive an expression for *Q(h'|h)*

Derive an expression for *p(Y|h')*

The probability $p(y_i|h')$ of a single instance $y_i = \langle x_i, z_{i1}, \dots, z_{ik} \rangle$ of the full data can be written as

$$p(y_i|h') = p(x_i, z_{i1}, \dots, z_{ik}|h') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\pi\sigma^2}\sum_{j=1}^{k} z_{ij}(x_i - \mu'_j)^2}$$

Here, only one of $z_{ij}$ can have the value 1 and all other must be 0.

Given this probability for a single instance $p(y_i|h')$,the logarithm of the probability *ln P(Y|h')* for all *m* instances in the data is

$$ln\ P(Y|h') = ln \prod_{i=1}^{m} p(y_i|h')$$

$$= \sum_{i=1}^{m} ln \quad p(y_i|h')$$

$$ln\ P(Y|h') = \sum_{i=1}^{m} \left( ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\pi\sigma^2}\sum_{j=1}^{k} z_{ij}(x_i - \mu'_j)^2} \right)$$

That is, *ln P(Y|h')* is a linear function of $z_{ij}$.

In general, for any function *f (z)* that is a linear function of *z*, the following equality holds

*E[f(z)] = f(E[z])*

And also

$$E[\ln P(Y|h'] = E\left[ \sum_{i=1}^{m} \left( ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\pi\sigma^2}\sum_{j=1}^{k} z_{ij}(x_i - \mu'_j)^2} \right) \right]$$

$$= \sum_{i=1}^{m} \left( ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\pi\sigma^2}\sum_{j=1}^{k} E[z_{ij}](x_i - \mu'_j)^2} \right)$$

Therefore, the function *Q(h'|h)* for the *k* means problem is

$$Q(h'|h) = \sum_{i=1}^{m} \left( ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\pi\sigma^2}\sum_{j=1}^{k} E[z_{ij}](x_i - \mu'_j)^2} \right)$$

Where, $h' = \langle \mu'_1, \ldots, \mu'_k \rangle$

*E* $[z_{ij}]$ → calculated based on current hypothesis *h* and observed data *X*.

From k-means Gaussians

$$E[z_{ij}] = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \quad \text{---(9)}$$

Thus,

- The first (estimation) step of the EM algorithm defines the Q function based on the estimated $E[z_{ij}]$ terms.
- The second (maximization) step then finds the values $\mu'_1, \ldots, \mu'_k$ that maximize this Q function.

In the current case

$$\underset{h'}{argmax} Q(h'|h) = \underset{h'}{argmax} \sum_{i=1}^{m} \left( ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\pi\sigma^2}\sum_{j=1}^{k} E[z_{ij}](x_i - \mu'_j)^2} \right)$$

$$= \underset{h'}{argmin} \sum_{i=1}^{m} \sum_{j=1}^{k} \sum_{j=1}^{k} E[z_{ij}] (x_i - \mu'_j)^2$$

---(10)

Therefore,

The maximum likelihood hypothesis here minimizes a weighted sum of squared errors, where the contribution of each instance $x_i$ to the error that defines $\mu'_j$ is weighted by $E[z_{ij}]$ .

The quantity given by Equation (10) is minimized by setting each $\mu'_j$ to the weighted sample mean

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

---(11)

Eq. 10 & 11 → Two steps in the k-means algorithm