Module - 4

BAYESIAN LEARNING

Bayesian reasoning provides a probabilistic approach to inference. It is based on the assumption 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.

INTRODUCTION

Bayesian learning methods are relevant to our study of machine learning for two different reasons.

- Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems.
- Bayesian methods are important to our study of machine learning is that they pro vide 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.

BAYES THEOREM

from some space H, given the observed training data D. One way to specify what we mean by the ***best*** hypothesis is to say that we demand the ***most probable*** hypothesis, given the data ***D*** plus any initial knowledge about the prior probabilities of the various hypotheses in H. Bayes theorem provides a direct method for calculating such probabilities. More precisely, Bayes theorem provides a way to:

- Calculate the probability of a hypothesis based on its prior probability,
- The probabilities of observing various data given the hypothesis, and
- The observed data itself.

Bayes theorem 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) is the prior probability of h*
- *P(D) to denote the prior probability that training data D will be observed (i.e.,the probability of D given no knowledge about which hypothesis holds).*
- *P(D|h) to denote the probability of observing data D given some world in which hypothesis h holds.*
- *P (h1 D) is called the posterior probability of h, because it reflects our confidence that h holds after we have seen the training data D.*

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

Any such maximally probable hypothesis is ca lled 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 h_{MAP} is a MAP hypothesis provided:

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

Notice in the final step above we dropped the term *P(D)* because it is a constant independent of *h.*

In some cases, we will assume 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 we can further simplify the above equation and need to 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}.

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

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 cancer.

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 has this 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?

The above situation can be summarized by the following probabilities:

$$\begin{aligned} P(cancer) &= .008, & P(\neg cancer) &= .992 \\ P(\oplus|cancer) &= .98, & P(\ominus|cancer) &= .02 \\ P(\oplus|\neg cancer) &= .03, & P(\ominus|\neg cancer) &= .97 \end{aligned}$$

The maximum a posteriori hypothesis can be found using

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

Thus, h_{MAP} = patient does not cancer.

The result of Bayesian inference depends strongly on the prior probabilities. also that in this example the hypotheses are not completely accepted or rejected, but rather become more or less probable as more data is observed.

BAYES THEOREM AND CONCEPT LEARNING

What is the relationship between Bayes theorem and the problem of concept learning? This section considers such a brute- force Bayesian concept learning algorithm, then compares it to concept learning algorithms.

Brute-Force Bayes Concept Learning

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

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

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). Here let us choose them to be consistent with the following assumptions:*

1. *The training data D is noise free (i.e., $d_i = c(x_i)$).*
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 these assumptions, what values should we specify for P(h).

According to the assumption 2 and 3

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

What choice shall we make for P(D|h)? 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 $d_i \neq h(x_i)$. Therefore,

$$P(D|h) = \begin{cases} 1 \text{ if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 \text{ otherwise} \end{cases} \qquad (6.4)$$

In other words, the probability of data D given hypothesis h is 1 if D is consistent with h, and 0 otherwise. Let us consider the first step of this algorithm, Recalling Bayes theorem, we have

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

First consider the case where h is inconsistent with the training data D. Since Equation (6.4) defines *P(D)h)* to be *0* when *h* is inconsistent with *D*, we have

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

Now consider the case where *h* is consistent with *D*. Since Equation *(6.4)* defines *P(D|h)* to be 1 when *h* is consistent with *D*, we have

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

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

We can derive *P(D)* from the theorem of total probability and the fact that the hypotheses are mutually exclusive

$$\begin{aligned} P(D) &= \sum_{h_i \in H} P(D|h_i)P(h_i) \\ &= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} \\ &= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} \\ &= \frac{|VS_{H,D}|}{|H|} \end{aligned}$$

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

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases} \qquad (6.5)$$

where $|VS_{H,D}|$ is the number of hypotheses from H consistent with ***D.*** The evolution of probabilities associated with hypotheses is depicted schematically in Figure *6.1*.

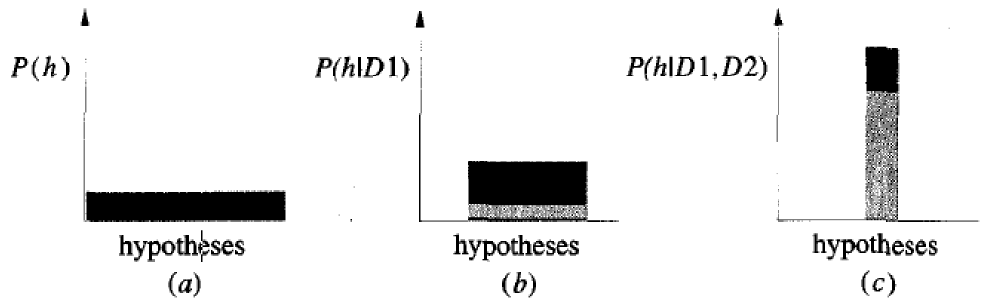


FIGURE 6.1
Evolution of posterior probabilities $P(h|D)$ with increasing training data. (*a*) Uniform priors assign equal probability to each hypothesis. As training data increases first to $D1$ (*b*), then to $D1 \wedge D2$ (*c*), the posterior probability of inconsistent hypotheses becomes zero, while posterior probabilities increase for hypotheses remaining in the version space.

MAP Hypotheses and Consistent Learners

A learning algorithm is a consistent learner provided it outputs a hypothesis that commits zero errors over the training examples. Given the above analysis, we can conclude that 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).

The Bayesian framework allows one way to characterize the behavior of learning algorithms (e.g., FIND-S), even when the learning algorithm does not explicitly manipulate probabilities. By identifying probability distributions ***P(h)*** and ***P(Dlh)*** under which the algorithm outputs optimal (i.e., MAP) hypotheses, we can characterize the implicit assumptions, under which this algorithm behaves optimally.

NAIVE BAYES CLASSIFIER

One highly practical Bayesian learning method is the naive Bayes learner, often called the naive Bayes classifier. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values (al, a2.. .a,). The learner is asked to predict the target value, or classification, for this new instance.

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

$$v_{MAP} = \underset{v_j \in V}{\text{argmax}}\ P(v_j \mid a_l, a_2 \ldots a_n)$$

We can use Bayes theorem to rewrite this expression as

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{\text{argmax}} \frac{P(a_1, a_2 \ldots a_n|v_j)P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \underset{v_j \in V}{\text{argmax}}\ P(a_1, a_2 \ldots a_n|v_j)P(v_j) \qquad (6.19) \end{aligned}$$

The assumption is that given the target value of the instance, the probability of observing the conjunction **al, a2.. .a,** is just the product of the probabilities for the individual attributes: $\mathbf{P(a_1, a_2 \ldots a_n \mid v_j) = \prod_i P(a_i|v_j)}$. Substituting this into Equation (6.19), we have the approach used by the naive Bayes classifier.

Naive Bayes classifier:

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

Where V_{NB} denotes the target value output by the naive Bayes classifier.

An Illustrative Example

Consider the dataset of 14 instances and 4 attributes that we have used in Decision tree learning module.

Here we use the naive Bayes classifier and the training data from this table to classify the following novel instance:

(Outlook = s unny, Te mpe rature = cool, Humidity = high, Wind = strong)

Our task is to predict the target value (yes or no) of the target concept PlayTennis for this new instance. Instantiating Equation (6.20) to fit the current task, the target value v_{NB} is given by

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

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

$$P(Humidity = high|v_j) P(Wind = strong|v_j) \qquad (6.21)$$

To calculate VNB we now require 10 probabilities that can be estimated from the training data. First, the probabilities of the different target values can easily be estimated based on their frequencies over the **14** training examples

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

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

Similarly, we can estimate the conditio nal probabilities. For example, those for *Wind* = *trong* are

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

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

Using these probability estimates and similar estimates for the remaining attribute values, we calculate ***VNB*** according to Equation ***(6.21)*** as follows

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

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

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

Estimating Probabilities

Up to this point we have estimated probabilities by the fraction of times the event is observed to occur over the total number of opportunities. For example, in the above case we estimated P(Wind = strong|Play Tennis = no) by the fraction n_c/n where n = 5 is the total number of training examples for which PlayTennis = no, and n, = 3 is the number of these for which Wind = strong. it provides poor estimates when $\mathbf{n_c}$ is very small. This raises two difficulties. First, n_c/n produces a biased underestimate of the probability. Second, when this probability estimate is zero, this probability term will dominate the Bayes classifier if the future query contains Wind = strong. To avoid this difficulty we can adopt a Bayesian approach to estimating the probability, using the m-estimate defined as follows.

***m*-estimate of probability:**

$$\frac{n_c + mp}{n + m} \tag{6.22}$$

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

BAYESIAN BELIEF N ETWORKS

- 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.
- In contrast to the naive Bayes classifier, which assumes that ***all*** the variables are conditionally independent given the value of the target variable, Bayesian belief networks allow stating conditional independence assumptions that apply to ***subsets*** of the variables.
- Thus, Bayesian belief networks provide an intermediate approach that is less constraining than the global assumption of conditional independence made by the naive Bayes classifier, but more tractable than avoiding conditional independence assumptions altogether.

Conditional Independence

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 $P(A_l, A_2 \mid V)$ in Equation (6.20) as follows

$$\begin{aligned} P(A_1, A_2|V) &= P(A_1|A_2, V)P(A_2|V) && (6.23) \\ &= P(A_1|V)P(A_2|V) && (6.24) \end{aligned}$$

Equation (6.23) is just the general form of the product rule of probability. Equation (6.24) follows because if A_1 is conditionally independent of A_2 given V, then by our definition of conditional independence $P(A_1|A_2, V) = P(A_1|V)$.

Representation

A Bayesian belief network (Bayesian network for short) represents the joint probability distribution for a set of variables. In general, a Bayesian network represents the joint probability distribution by specifying a set of conditional independence assumptions (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. For each variable two types of information are specified. First, the network arcs represent the assertion that the variable is conditionally independent of its nondescendants in the network given its immediate predecessors in the network. Second, a conditional probability table 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, . . . , y_n)*** to the tuple of network variables ***(Y_1 . . . Y_n)*** can be computed by the formula

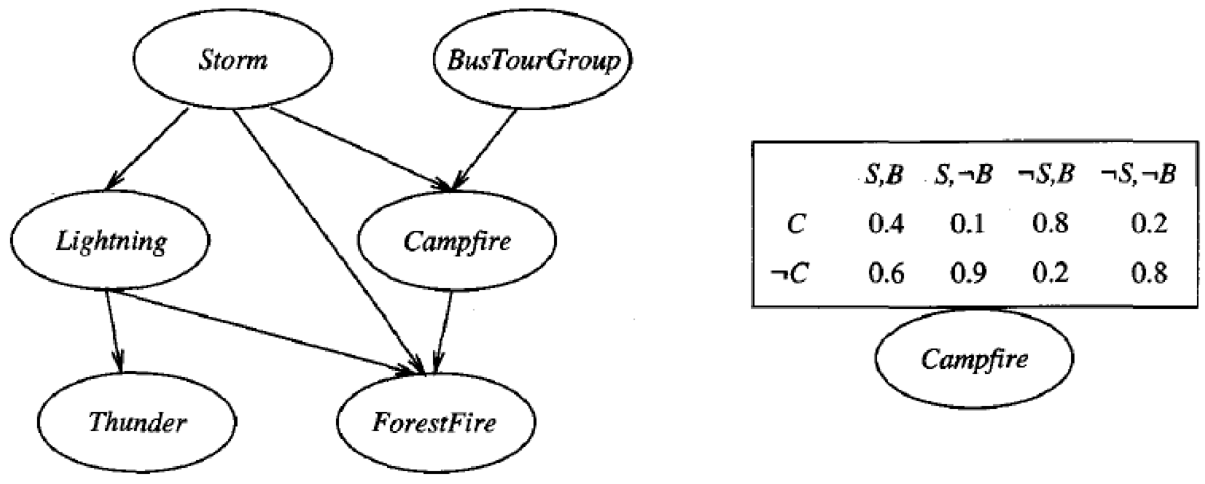
$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

where **Parents(Y_i)** denotes the set of immediate predecessors of **Y_i** in the network. To illustrate, the Bayesian network in Figure 6.3 represents the joint probability distribution over the boolean variables Storm, Lightning, Thunder, Forest Fire, Campfire, and BusTourGroup. Consider the node Campfire. The network nodes and arcs represent the assertion that Campfire is conditionally independent of its nondescendants Lightning and Thunder, given its immediate parents Storm and BusTourGroup. 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 right side of the figure shows the conditional probability table associated with the variable Campfire. The top left entry in this table, for example, expresses the assertion that

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

this table provides only the conditional probabilities of *Campfire* given its parent variables *Storm* and *BusTourGroup.* The set of local conditional probability tables for all the variables, together with the set of conditional independence assumptions described by the network, describe the full joint probability distribution for the network.

One attractive feature of Bayesian belief networks is that they allow a convenient way to represent causal knowledge such as the fact that *Lightning* causes *Thunder.* In the terminology of conditional independence, we express this by stating that *Thunder* is conditionally independent of other variables in the network, given the value of *Lightning.*



	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

FIGURE 6.3
A Bayesian belief network. The network on the left represents a set of conditional independen assumptions. In particular, each node is asserted to be conditionally independent of its nondesce dants, given its immediate parents. Associated with each node is a conditional probability tabl which specifies the conditional distribution for the variable given its immediate parents in the grap The conditional probability table for the *Campfire* node is shown at the right, where *Campfire* abbreviated to *C*, *Storm* abbreviated to *S*, and *BusTourGroup* abbreviated to *B*.

Infe rence

We might wish to use a Bayesian network to infer the value of some target variable (e.g., *ForestFire)* given the observed values of the other variables. This inference step can be straightforward if values for all of the other variables in the network are known exactly. In the more general case we may wish to infer the probability distribution for some variable (e.g., ***ForestFire)*** given observed values for only a subset of the other variables e.g., ***Thunder*** and ***BusTourGroup*** may be the only observed values available). In general, a Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables. Exact inference of probabilities in general for an arbitrary Bayesian network is known to be NP-hard (Cooper 1990). N umerous methods have been proposed for probabilistic inference in Bayesian networks, including exact inference methods and approximate inference methods that sacrifice precision to gain efficiency.

Learning Bayesian Belief Networks

Can we devise effective algorithms for learning Bayesian belief networks from training data?. Several different settings for this learning problem can be considered. First, the network structure might be given in advance, or it might have to be inferred from the training data. Second, all the network variables might be directly observable in each training example, or some might be unobservable.

In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward. We simply estimate the conditional probability table entries just as we would for a naive Bayes classifier.

In the case where the network structure is given but only some of the variable values are observable in the training data, the learning problem is more difficult. This problem is somewhat analogous to learning the weights for the hidden units in an artificial neural network. In fact, Russell et al. (1995) propose a similar gradient ascent procedure that learns the entries in the conditional probability tables.

Learning the Structure of Bayesian Networks

Learning Bayesian networks when the network structure is not known in advance is also difficult. Cooper and Herskovits (1992) present a Bayesian scoring metric for choosing among alternative networks. They also present a heuristic search algorithm called **K2** for learning network structure when the data is fully observable. Like most algorithms for learning the structure of Bayesian networks, K2 performs a greedy search that trades off network complexity for accuracy over the training data. Constraint-based approaches to learning Bayesian network structure have also been developed. These approaches infer independence and dependence relationships from the data, and then use these relationships to construct Bayesian networks.

THE EM ALGORITHM

In many practical learning settings, only a subset of the relevant instance features might be observable. EM algorithm is widely used approach to learning in the presence of unobserved variables. The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known. The EM algorithm is also the basis for many unsupervised clustering algorithms.

Estimating Means of k Gaussians

The easiest way to introduce the EM algorithm is via an example. Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normal distributions. This problem setting is for the case where k = 2. The learning task is to output a hypothesis **$h = (\mu_1, \ldots \mu_k)$** that describes the means of each of the k distributions. We would like to find a maximum likelihood hypothesis for these means; It is easy to calculate the maximum likelihood hypothesis for the mean of a single Normal distribution.

The maximum likelihood hypothesis is the one that minimizes the sum of squared errors over the m training instances. We have

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2 \qquad (6.27)$$

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

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad (6.28)$$

Our problem here, however, involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution. Thus, we have a prototypical example of a problem involving hidden variables. We can think of the full description of each instance as the triple ***(x_i, z_{i1}, z_{i2}),*** where ***x_i*** is the observed value of the ith instance and where ***z_{il}*** 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 jth 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.

Applied to the problem of estimating the two means the EM algorithm first initializes the hypothesis to **h = (μ₁ , μₖ),** where **μ₁** and **μₖ** are arbitrary initial values. It then iteratively re-estimates h by repeating the following two steps until the procedure converges to a stationary value for h.

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.

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

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.

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

General Statement of EM Algorithm

➔ More generally, the EM algorithm can be applied in many settings where we wish to estimate some set of parameters **θ** that describe an underlying probability distribution, given only the observed portion of the full data produced by this distribution. In the above two- means example the parameters of interest were $\theta = (\mu_1, \mu_2)$, and the full data were the triples (x_i, z_{i1}, z_{i2}) of which only the x_i were observed.

➔ In general let $X = \{x_1, \ldots, x_m\}$ denote the observed data in a set of m independently drawn instances, let $Z = \{z_1, \ldots, z_m\}$ denote the unobserved data in these same instances, and let $Y = X \cup Z$ denote the full data.

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

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

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

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

➔ 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*.

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

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

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

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

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

Derivation of the k Means Algorithm

Let us use EM Algorithm to derive the algorithm for estimating the means of a mixture of k Normal distributions. To apply EM we must derive an expression for ***Q(h|h')*** that applies to our k- means problem.

First, let us derive an expression for ***1n p(Y|h').*** Note the probability ***p(y_i|h')*** of a single instance $y_i = (x_i, z_{i1}, \ldots z_{ik})$ of the full data can be written

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

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

$$\begin{aligned}\ln P(Y|h') &= \ln \prod_{i=1}^{m} p(y_i|h') \\ &= \sum_{i=1}^{m} \ln p(y_i|h') \\ &= \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} z_{ij}(x_i - \mu_j')^2 \right)\end{aligned}$$

Note the above expression for In P(Y|h') is a linear function of these z_{ij}. In general, for any function f (z) that is a linear function of z, the following equality holds

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

This general fact about linear functions allows us to write

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

To summarize, the function Q(h'|h) for the k means problem is

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

where $h' = \langle \mu_1', \ldots, \mu_k' \rangle$ and where $E[z_{ij}]$ is calculated based on the current hypothesis h and observed data X. As discussed earlier

$$E[z_{ij}] = \frac{e^{-\frac{1}{2\sigma^2}(x_i-\mu_j)^2}}{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i-\mu_n)^2}} \tag{6.29}$$

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

$$\begin{aligned}\underset{h'}{\text{argmax}}\, Q(h'|h) &= \underset{h'}{\text{argmax}} \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} E[z_{ij}](x_i - \mu_j')^2 \right) \\ &= \underset{h'}{\text{argmin}} \sum_{i=1}^{m} \sum_{j=1}^{k} E[z_{ij}](x_i - \mu_j')^2 \end{aligned} \tag{6.30}$$

Thus, 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 μ_j' is weighted by $E[z_{ij}]$. The quantity given by Equation (6.30) is minimized by setting each μ_j' to the weighted sample mean

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}]\ \ x_i}{\sum_{i=1}^{m} E[z_{ij}]} \tag{6.31}$$