Bayesian Learning

Features of Bayesian learning methods:

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.
  - This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting
  - a prior probability for each candidate hypothesis, and
  - a probability distribution over observed data for each possible hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

Difficulties with Bayesian Methods

- 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.
- Significant computational cost is 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.

Bayes Theorem

- In machine learning, we try to determine the ***best hypothesis*** from some hypothesis space H, given the observed training data D.
- In Bayesian learning, the ***best hypothesis*** means 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 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

P(h) is ***prior probability of hypothesis h***

- P(h) to denote the initial probability that hypothesis h holds, before observing training data.
- P(h) may reflect any background knowledge we have about the chance that h is correct. If we have no such prior knowledge, then each candidate hypothesis might simply get the same prior probability.

P(D) is ***prior probability of training data D***

- The probability of D given no knowledge about which hypothesis holds

P(h|D) is ***posterior probability of h given 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.***
- 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.

P(D|h) is ***posterior probability of D given h***

- The probability of observing data ***D*** given some world in which hypothesis ***h*** holds.
- Generally, we write ***P(xly)*** to denote the probability of **event x** given **event y.**

Bayes Theorem

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

Bayes Theorem: $$P(h \mid D) = \frac{P(D \mid h)\, P(h)}{P(D)}$$

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

Bayes Theorem - Example

Sample Space for events A and B

A holds	T	T	F	F	T	F	T
B holds	T	F	T	F	T	F	F

P(A) = 4/7 P(B) = 3/ 7 P(B|A) = 2/4 P(A|B) = 2/3

Is Bayes Theorem correct?

P(B|A) = P(A|B)P(B) / P(A) = (2/3 * 3/7) / 4/7 = 2/4 ➔ CORRECT

P(A|B) = P(B|A)P(A) / P(B) = (2/4 * 4/7) / 3/7 = 2/3 ➔ CORRECT

Maximum A Posteriori (MAP) Hypothesis, h_{MAP}

- The learner considers some set of candidate hypotheses H and it is interested in finding the ***most probable hypothesis*** h $\in$ H given the observed data D
- Any such maximally probable hypothesis is called a ***maximum a posteriori (MAP) hypothesis*** h_{MAP}.
- We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.

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

Maximum Likelihood (ML) Hypothesis, h_{ML}

- If we assume that every hypothesis in H is equally probable

  i.e. $P(h_i) = P(h_j)$ for all h_i and h_j in H

  We can only consider P(D|h) to find the most probable hypothesis.

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

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

Example - Does patient have cancer or not?

- The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present.
- Furthermore, .008 of the entire population have cancer.

P(cancer) = .008	P(notcancer) = .992
P(+|cancer) = .98	P(-|cancer) = .02
P(+|notcancer) = .03	P(-|notcancer) = .97

- A patient takes a lab test and the result comes back positive.

P(+|cancer) P(cancer) = .98 * .008 = .0078

P(+|notcancer) P(notcancer) = .03 * .992 = .0298 ➔ ***h_{MAP} is notcancer***

- Since P(cancer|+) + P(notcancer|+) must be 1

P(cancer|+) = .0078 / (.0078+.0298) = .21

P(notcancer|+) = .0298 / (.0078+.0298) = .79

Basic Formulas for Probabilities

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

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

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

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

Brute-Force Bayes Concept Learning

- A Concept-Learning algorithm considers a finite hypothesis space **H** defined over an instance space **X**
- The task is to learn the target concept (a function) **c : X → {0,1}.**
- The learner gets a set of training examples $(<x_1,d_l> \dots <x_m,d_m>)$ where x_i is an instance from **X** and d_i is its target value (i.e. $c(x_i) = d_i$).

- *Brute-Force Bayes Concept Learning Algorithm* finds the maximum a posteriori hypothesis (h_{MAP}), based on Bayes theorem.

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

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

- This algorithm may require significant computation, because it applies Bayes theorem to each hypothesis in H to calculate P(h|D) .
  - While this is impractical for large hypothesis spaces,
  - The algorithm is still of interest because it provides a standard against which we may judge the performance of other concept learning algorithms.

Brute-Force MAP Learning Algorithm

- BF MAP learning algorithm must specify values for $P(h)$ and $P(D|h)$.
- $P(h)$ and $P(D|h)$ must be chosen to be consistent with the 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.

- With these assumptions:

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

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

Brute-Force MAP Learning Algorithm

- So, the values of P(h|D) will be:

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

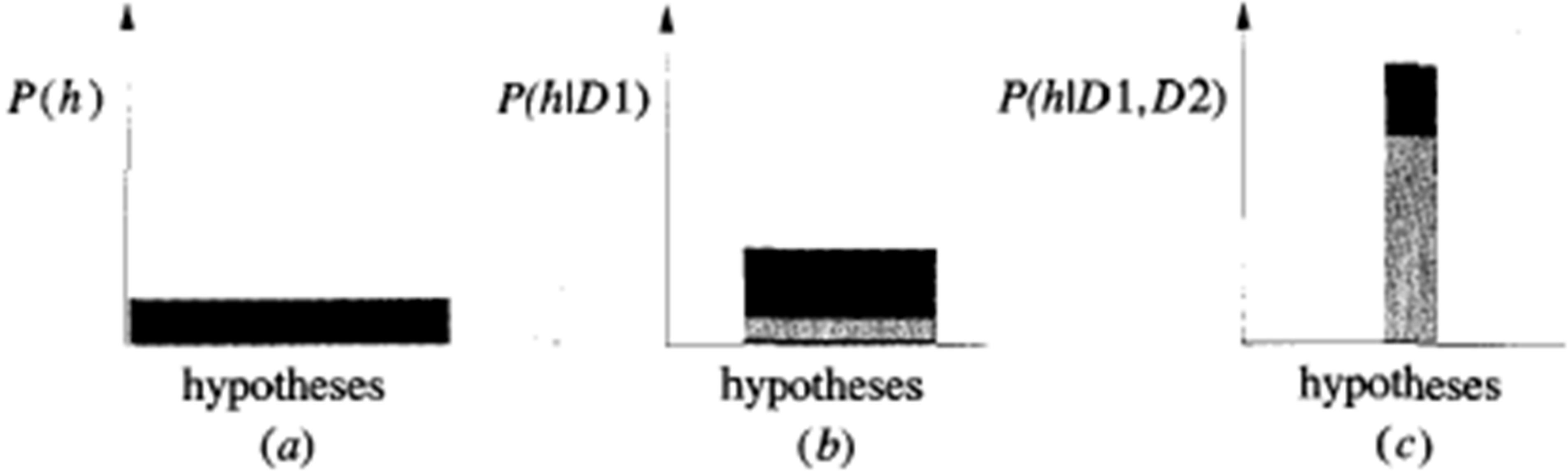
where $VS_{H,D}$ is the version space of H with respect to D.

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

- $P(D) = |VS_{H,D}| / |H|$ because
  - the sum over all hypotheses of P(h|D) must be one and the number of hypotheses from H consistent with D is $|VS_{H,D}|$, or
  - we can derive P(D) from *the theorem of total probability* and the fact that the hypotheses are mutually exclusive (i.e., $(\forall i \neq j)(P(h_i \wedge h_j) = 0)$

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

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 Dl (b), then to Dl $\wedge$ 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*** if it outputs a hypothesis that commits zero errors over the training examples.
- Every consistent learner outputs a MAP hypothesis, if we assume
  - a uniform prior probability distribution over H (i.e., $P(h_i) = P(h_j)$ for all i, j), and
  - deterministic, noise free training data (i.e., $P(D|h) = 1$ if D and h are consistent, and 0 otherwise).
- Because FIND-S outputs a consistent hypothesis, it will output a MAP hypothesis under the probability distributions P(h) and P(D|h) defined above.
- Are there other probability distributions for P(h) and P(D|h) under which FIND-S outputs MAP hypotheses? Yes.
  - Because FIND-S outputs a maximally specific hypothesis from the version space, its output hypothesis will be a MAP hypothesis relative to any prior probability distribution that favors more specific hypotheses.
  - More precisely, suppose we have a probability distribution P(h) over H that assigns $P(h_1) \geq P(h_2)$ if h_1 is more specific than h_2.

Maximum Likelihood and Least-Squared Error Hypotheses

- Many learning approaches such as neural network learning, linear regression, and polynomial curve fitting try to learn a continuous-valued target function.

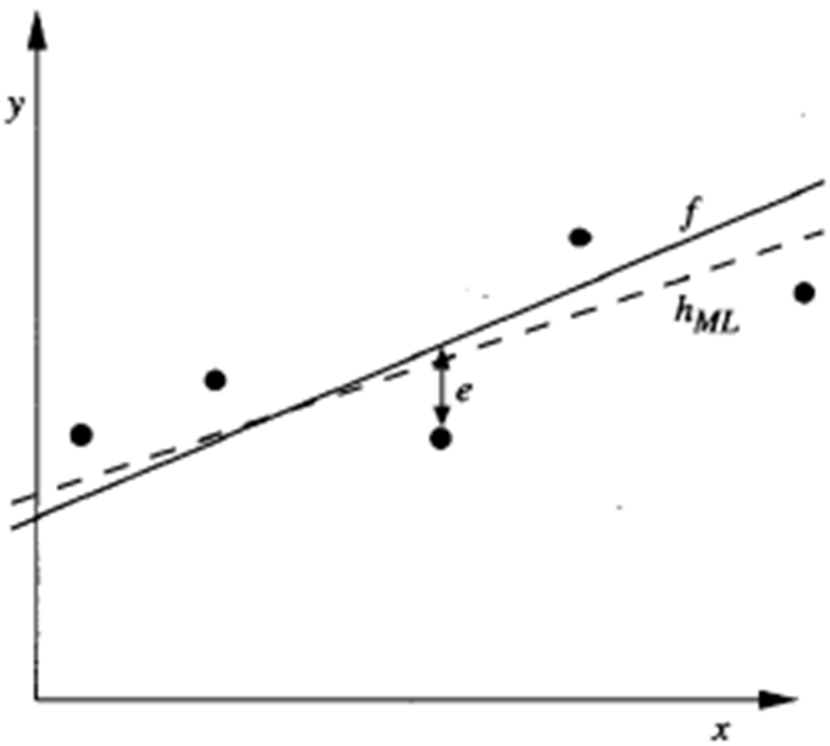
- **Under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a *MAXIMUM LIKELIHOOD HYPOTHESIS*.**

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

Learning A Continuous-Valued Target Function

- Learner L considers an instance space X and a hypothesis space H consisting of some class of real-valued functions defined over X.
- The problem faced by L is to learn an unknown target function f drawn from H.
- A set of m training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution
- Each training example is a pair of the form (x_i, d_i) where $d_i = f(x_i) + e_i$.
  - Here $f(x_i)$ is the noise-free value of the target function and e_i is a ***random variable*** representing the noise.
  - It is assumed that the values of the e_i are ***drawn independently*** and that they are distributed according to a ***Normal distribution*** with zero mean.
- The task of the learner is to output a ***maximum likelihood hypothesis***, or, equivalently, a MAP hypothesis assuming all hypotheses are equally probable a priori.

Learning A Linear Function



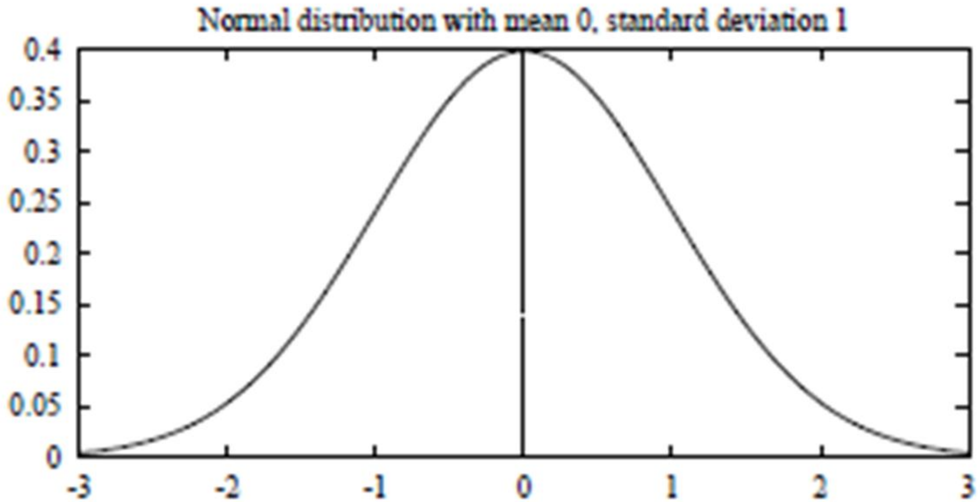
- The target function f corresponds to the solid line.
- The training examples (x_i, d_i) are assumed to have Normally distributed noise e_i with zero mean added to the true target value f (x_i).
- The dashed line corresponds to the hypothesis h_{ML} with least-squared training error, hence the maximum likelihood hypothesis.
- Notice that the maximum likelihood hypothesis is not necessarily identical to the correct hypothesis, f, because it is inferred from only a limited sample of noisy training data.

Basic Concepts from Probability Theory

- Before showing why a hypothesis that minimizes the sum of squared errors in this setting is also a maximum likelihood hypothesis, let us quickly review basic concepts from probability theory

- A *random variable* can be viewed as the name of an experiment with a probabilistic outcome. Its value is the outcome of the experiment.
- A *probability distribution* for a random variable Y specifies the probability $\Pr(Y = y_i)$ that Y will take on the value y_i, for each possible value y_i.
- The *expected value*, or *mean*, of a random variable Y is $E[Y] = \sum_i y_i \Pr(Y = y_i)$. The symbol μ_Y is commonly used to represent E[Y].
- The *variance* of a random variable is $Var(Y) = E[(Y - \mu_Y)^2]$. The variance characterizes the width or dispersion of the distribution about its mean.
- The *standard deviation* of Y is $\sqrt{Var(Y)}$. The symbol σ_Y is often used used to represent the standard deviation of Y.
- The *Normal distribution* is a bell-shaped probability distribution that covers many natural phenomena.
- The *Central Limit Theorem* is a theorem stating that the sum of a large number of independent, identically distributed random variables approximately follows a Normal distribution.

Basic Concepts from Probability Theory



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

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

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

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

Maximum Likelihood and Least-Squared Error Hypotheses – Deriving h_{ML}

- In order to find the maximum likelihood hypothesis, we start with our earlier definition but using lower case ***p*** to refer to the probability density function.

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

- We assume a fixed set of training instances $(x_1 \ldots x_m)$ and therefore consider the data D to be the corresponding sequence of target values $D = (d_1 \ldots d_m)$.
- Here $d_i = f(x_i) + e_i$. Assuming the training examples are mutually independent given h, we can write $p(D|h)$ as the product of the various $p(d_i|h)$

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

Maximum Likelihood and Least-Squared Error Hypotheses – Deriving h_{ML}

- Given that the noise e_i obeys a Normal distribution with zero mean and unknown variance σ^2, each d_i must also obey a Normal distribution with variance σ^2 centered around the true target value $f(x_i)$ rather than zero.
- $p(d_i|h)$ can be written as a Normal distribution with variance σ^2 and mean $\mu = f(x_i)$.
- Let us write the formula for this Normal distribution to describe $p(d_i|h)$, beginning with the general formula for a Normal distribution and substituting appropriate μ and σ^2 .
- Because we are writing the expression for the probability of d_i given that h is the correct description of the target function f, we will also substitute $\mu = f(xi) = h(xi)$,

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

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

Maximum Likelihood and Least-Squared Error Hypotheses – Deriving h_{ML}

- Maximizing **ln p** also maximizes **p**.

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

- First term is constant, discard it.

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

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

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

- Finally, we can again discard constants that are independent of ***h***.

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

Maximum Likelihood and Least-Squared Error Hypotheses

- The maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of the squared errors between observed training values d_i and hypothesis predictions $h(x_i)$.
- This holds under the assumption that the observed training values d_i are generated by adding random noise to the true target value, where this random noise is drawn independently for each example from a Normal distribution with zero mean.
- Similar derivations can be performed starting with other assumed noise distributions, producing different results.
- Why is it reasonable to choose the Normal distribution to characterize noise?
  - One reason, is that it allows for a mathematically straightforward analysis.
  - A second reason is that the smooth, bell-shaped distribution is a good approximation to many types of noise in physical systems.
- Minimizing the sum of squared errors is a common approach in many neural network, curve fitting, and other approaches to approximating real-valued functions.

Bayes Optimal Classifier

- Normally we consider:
  - What is the most probable ***hypothesis*** given the training data?
- We can also consider:
  - what is the most probable ***classification*** of the new instance given the training data?

- Consider a hypothesis space containing three hypotheses, hl, h2, and h3.
  - Suppose that the posterior probabilities of these hypotheses given the training data are .4, .3, and .3 respectively.
  - Thus, hl is the MAP hypothesis.
  - Suppose a new instance x is encountered, which is classified positive by *hl,* but negative by *h2* and h3.
  - Taking all hypotheses into account, the probability that x is positive is .4 (the probability associated with *h1),* and the probability that it is negative is therefore .6.
  - The most probable classification (negative) in this case is different from the classification generated by the MAP hypothesis.

Bayes Optimal Classifier

- 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 the new example can take on any value $\boldsymbol{v_j}$ from some set V, then the probability $P(v_j \mid D)$ that the correct classification for the new instance is $\boldsymbol{v_j}$:

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

- **Bayes optimal classification:**

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

Bayes Optimal Classifier - Ex

$$P(h_1|D) = .4, \;\; P(\ominus|h_1) = 0, \;\; P(\oplus|h_1) = 1$$

$$P(h_2|D) = .3, \;\; P(\ominus|h_2) = 1, \;\; P(\oplus|h_2) = 0$$

$$P(h_3|D) = .3, \;\; P(\ominus|h_3) = 1, \;\; P(\oplus|h_3) = 0$$

Probabilities:

$$\sum_{h_i \in H} P(\oplus|h_i) P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(\ominus|h_i) P(h_i|D) = .6$$

Result:

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

Bayes Optimal Classifier

- Although the Bayes optimal classifier obtains the best performance that can be achieved from the given training data, it can be quite costly to apply.
  - The expense is due to the fact that it computes the posterior probability for every hypothesis in H and then combines the predictions of each hypothesis to classify each new instance.

- An alternative, less optimal method is the 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*.**

Naive Bayes Classifier

- One highly practical Bayesian learning method is Naive Bayes Learner (***Naive Bayes Classifier)***.
- The naive Bayes classifier applies to learning tasks where each instance ***x*** is described by a conjunction of attribute values and where the target function f (x) can take on any value from some finite set V.
- A set of training examples is provided, and a new instance is presented, described by the tuple of attribute values $(a_1, a_2 \ldots a_n)$.
- The learner is asked to predict the target value (classification), for this new instance.

Naive Bayes Classifier

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

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

- By Bayes theorem:

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

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

Naive Bayes Classifier

- It is easy to estimate each of the $\mathbf{P(v_j)}$ simply by counting the frequency with which each target value $\mathbf{v_j}$ occurs in the training data.
- However, estimating the different $\mathbf{P(a_1,a_2\ldots a_n \mid v_j)}$ terms is not feasible unless we have a very, very large set of training data.
  - The problem is that the number of these terms is equal to the number of possible instances times the number of possible target values.
  - Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.
- The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.
- For a given the target value of the instance, the probability of observing conjunction $\boldsymbol{a_1,a_2...a_n}$, is just the product of the probabilities for the individual attributes:

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

- **Naive Bayes classifier:** $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$

Naive Bayes Classifier - Ex

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes Classifier -Ex

- New instance to classify:

  (Outlook=sunny, Temperature=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.

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

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

Naive Bayes Classifier -Ex

- P(P1ayTennis = yes) = 9/14 = .64
- P(P1ayTennis = no) = 5/14 = .36

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

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

➔ 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.

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

.0206 / (.0206 + .0053) = .795

Estimating Probabilities

- **P(Wind=strong | PlayTennis=no)** by the fraction $\mathbf{n_c/n}$ where $\boldsymbol{n}$ = 5 is the total number of training examples for which **PlayTennis=no,** and $\boldsymbol{n_c}$ = 3 is the number of these for which **Wind=strong.**
- When $\boldsymbol{n_c}$ is zero
  - $\mathbf{n_c/n}$ will be zero too
  - this probability term will dominate
- 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: $\mathbf{(n_c + m*p) / (n + m)}$
- if an attribute has $\boldsymbol{k}$ possible values we set p = 1/k .
  - p=0.5 because Wind has two possible values.
- m is called the equivalent sample size
  - augmenting the $\boldsymbol{n}$ actual observations by an additional m virtual samples distributed according to p.

Learning To Classify Text

LEARN_NAIVE_BAYES_TEXT(Examples,V)

- Examples is a set of text documents along with their target values. V is the set of all possible target values.
- This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k.
- It also learns the class prior probabilities $P(v_j)$.

1. *collect all words, punctuation, and other tokens that occur in* ***Examples***
   - **Vocabulary** ← the set of all distinct words and other tokens occurring in any text document from **Examples**

LEARN_NAIVE_BAYES_TEXT(Examples,V)

2. *calculate the required* $\mathbf{P(v_j)}$ *and* $\mathbf{P(w_k|v_j)}$ *probability terms*

For each target value v_j in **V** do

- $\mathbf{docs_j}$ ← the subset of documents from **Examples** for which the target value is $\mathbf{v_j}$
- $\mathbf{P(v_j) \leftarrow |docs_j| \,/\, |Examples|}$
- $\mathbf{Text_j}$ ← a single document created by concatenating all members of $\mathbf{docs_j}$
- **n** ← total number of distinct word positions in **Examples**
- for each word $\mathbf{w_k}$ in **Vocabulary**
  - $\mathbf{n_k}$ ← number of times word $\mathbf{w_k}$ occurs in $\mathbf{Text_j}$
  - $\mathbf{P(w_k|v_j) \leftarrow (n_k + 1) \,/\, (n + |Vocabulary|)}$

CLASSIFY_NAIVE_BAYES_TEXT(Doc)

- *Return the estimated target value for the document* **Doc**.
- $\mathbf{a_i}$ *denotes the word found in the* i^{th} *position within* **Doc**.
  - **positions** ← all word positions in **Doc** that contain tokens found in **Vocabulary**
  - Return $\mathbf{V_{NB}}$, where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}}\, P(v_j) \prod_{i \in positions} P(a_i | v_j)$$