

# Bayesian Learning

In Bayesian learning, we are interested in the probability of a hypothesis  $h$  given the dataset  $D$ . By Bayes theorem:

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

Other useful formulas to remember are:

Product Rule:  $P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$

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

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

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

---

## Bayesian Learning Criteria

Different criteria for specifying the optimal hypothesis use Bayes theorem in different ways. Choose  $h$  to:

**MAP: Maximum a posteriori:** maximize  $P(h | D)$

**ML: Maximum Likelihood:** maximize  $P(D | h)$

**MDL: Minimum Description Length:** minimize Coding length of  $h$  + Coding length of  $D$  assuming  $h$

**Bayes Optimal Classification** of  $\mathbf{x} \in X$ :

Choose  $y$  that maximizes  $P(\mathbf{x}, y)$

**Parameter Estimation:**  $P(\mathbf{x}, y)$  has a known type, but parameters are unknown.

## Maximum Likelihood and Least Squared Error

Suppose outcomes normally dist. around  $f(\mathbf{x})$

$$p(y \mid \mathbf{x}) = \frac{e^{-(f(\mathbf{x})-y)^2/(2\sigma^2)}}{\sqrt{2\pi\sigma}}$$

$$h_{ML} = \operatorname{argmax}_{h \in H} p(D \mid h) = \operatorname{argmax}_{h \in H} \prod_{i=1}^m p((\mathbf{x}_i, y_i) \mid h)$$

Assume  $x$  is independent of  $h$  (skipping steps)

$$\begin{aligned} &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(y_i \mid h, \mathbf{x}_i) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{e^{-(h(\mathbf{x}_i)-y)^2/(2\sigma^2)}}{\sqrt{2\pi\sigma}} \\ &= \operatorname{argmin}_{h \in H} \sum_{i=1}^m (h(\mathbf{x}_i) - y)^2 \end{aligned}$$

---

## Maximum Likelihood and Probability Prediction

Assume  $Y = \{0, 1\}$  and  $f(\mathbf{x}) = P(y = 1 \mid \mathbf{x})$

$$h_{ML} = \operatorname{argmax}_{h \in H} p(D \mid h) = \operatorname{argmax}_{h \in H} \prod_{i=1}^m p((\mathbf{x}_i, y_i) \mid h)$$

Assume  $\mathbf{x}$  is independent of  $h$  (skipping steps)

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m P(y_i \mid h, \mathbf{x}_i)$$

$P(y \mid h, \mathbf{x})$  is  $h(\mathbf{x})$  if  $y = 1$ , or  $1 - h(\mathbf{x})$  if  $y = 0$ .

$$\begin{aligned} &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m h(\mathbf{x}_i)^{y_i} (1 - h(\mathbf{x}_i))^{1-y_i} \\ &= \operatorname{argmin}_{h \in H} \sum_{i=1}^m -y_i \ln h(\mathbf{x}_i) - (1 - y_i) \ln(1 - h(\mathbf{x}_i)) \end{aligned}$$

The last expression is called cross entropy. By a trick, we can convert this to least squares error again.

# Minimum Description Length Principle

$$\begin{aligned}h_{MAP} &= \operatorname{argmax}_{h \in H} P(h \mid D) = \operatorname{argmax}_{h \in H} \frac{P(D \mid h)P(h)}{P(D)} \\ &= \operatorname{argmax}_{h \in H} P(D \mid h)P(h) \\ &= \operatorname{argmax}_{h \in H} \log_2 P(D \mid h) + \log_2 P(h) \\ &= \operatorname{argmin}_{h \in H} -\log_2 P(D \mid h) - \log_2 P(h)\end{aligned}$$

MDL makes the following assumptions:

- $-\log_2 P(h)$  = description length of  $h$  in bits
- $-\log_2 P(D \mid h)$  = length of  $D$  assuming  $h$   
= length of outcomes  $y_1, \dots, y_m$  assuming  $h$

---

## Bayes Optimal Classification

Previous criteria focus on the most probable hypothesis. BOC focuses on the most probable classification. Each hypothesis is weighted by its posterior probability.

$$\operatorname{argmax}_{y \in Y} P((\mathbf{x}, y) \mid D)$$

By theorem of total probability:

$$= \operatorname{argmax}_{y \in Y} \sum_{h \in H} P((\mathbf{x}, y) \wedge h \mid D)$$

After skipping a few steps:

$$= \operatorname{argmax}_{y \in Y} \sum_{h \in H} P(y \mid h, \mathbf{x})P(h \mid D)$$

# Gibbs Algorithm

BOC is too expensive because it requires a summation over the whole hypothesis space. A simpler alternative is Gibbs algorithm:

For each instance  $x$ ,

    Select  $h$  randomly according to  $P(h | D)$

    Use  $h$  to classify  $x$

Gibbs algorithm has at most twice the error rate of BOC. However, Gibbs algorithm is still a difficult algorithm to apply because it requires random sampling based on  $P(h | D)$ .

---

## Parameter Estimation

If the type of probability distribution is known/assumed, use  $D$  to estimate its parameters.

Example: Each class  $y$  follows a multivariate normal distribution:

$$P(y | \mathbf{x}) = \frac{1}{(2\pi)^n |\boldsymbol{\Sigma}|} \exp\left\{-\frac{(\mathbf{x} - \mathbf{m})^t \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{m})}{2}\right\}$$

where  $\mathbf{x}$  is a  $n \times 1$  vector,  $\mathbf{m}$  are the means,  $\boldsymbol{\Sigma}$  is an  $n \times n$  covariance matrix,  $|\boldsymbol{\Sigma}|$  is its determinant, and  $\boldsymbol{\Sigma}^{-1}$  is its inverse.

Use  $D$  to estimate  $\mathbf{m}$  and  $\boldsymbol{\Sigma}$  for each class.

# Example Parameter Estimation: Iris2

Assume each class is generated by a normal distribution.  
Sample covariance  $\sigma_{x_j x_k}^2$  is

$$\sum_{\mathbf{x} \in D} (x_j - \mu_{x_j})(x_k - \mu_{x_k}) / (|D| - 1)$$

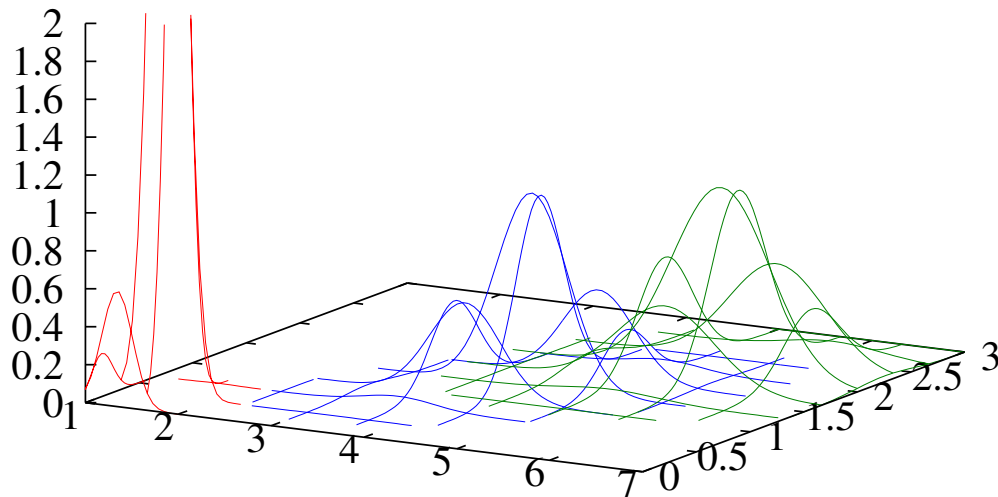
$$\boldsymbol{\mu}_1 = [1.464 \ 0.244] \quad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.0301 & 0.0057 \\ 0.0057 & 0.0115 \end{bmatrix}$$

$$\boldsymbol{\mu}_2 = [4.260 \ 1.326] \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.2208 & 0.0731 \\ 0.0731 & 0.0391 \end{bmatrix}$$

$$\boldsymbol{\mu}_3 = [5.552 \ 2.026] \quad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 0.3046 & 0.0488 \\ 0.0488 & 0.0754 \end{bmatrix}$$

---

## Iris2 Distributions



# Discriminant Functions

Discriminant functions  $d_i(x)$  for classes  $Y$  are designed so that the decision for  $\mathbf{x}$  is class:

$$\operatorname{argmax}_{y \in Y} P(y \mid \mathbf{x})$$

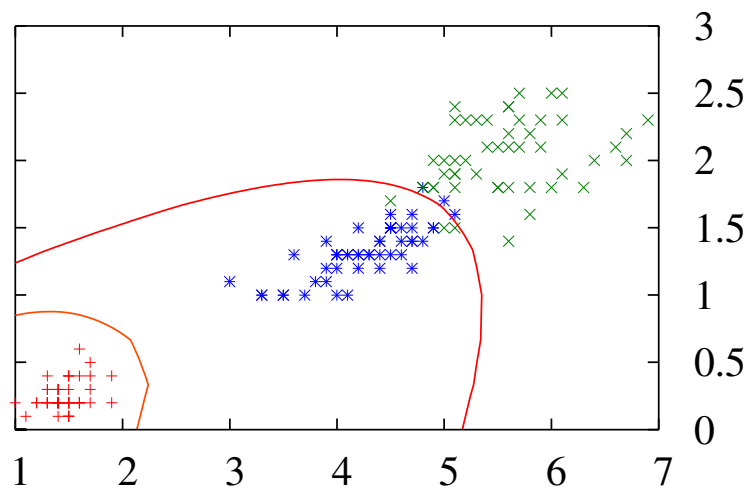
The decision boundary between two classes  $y_1$  and  $y_2$  is where  $d_1(\mathbf{x}) = d_2(\mathbf{x})$ .

Typically,  $d_i(\mathbf{x}) \propto \ln P(\mathbf{x} \mid y_i) + \ln P(y_i)$

For normal distributions,  $d_i(\mathbf{x})$  is equal to:

$$-\frac{1}{2} \ln |\boldsymbol{\Sigma}_i| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i) \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)^T + \ln P(y_i)$$

## Iris2 Decision Boundaries



# Naive Bayes Classifier

Naive Bayes classification is derived by:

$$\begin{aligned} y_{MAP} &= \operatorname{argmax}_{y \in Y} P(y \mid \mathbf{x}) = \operatorname{argmax}_{y \in Y} \frac{P(\mathbf{x})P(y)}{P(\mathbf{x})} \\ &= \operatorname{argmax}_{y \in Y} P(\mathbf{x} \mid y)P(y) \end{aligned}$$

Now “naively” assume that attributes are conditionally independent given the outcome. Let  $\mathbf{x} = (x_1, \dots, x_n)$

$$\begin{aligned} &= \operatorname{argmax}_{y \in Y} P(y) \prod_{j=1}^n P(x_j \mid y) \\ &= \operatorname{argmax}_{y \in Y} \log P(y) + \sum_{j=1}^n \log P(x_j \mid y) \end{aligned}$$

Note that the result is a linear function.

---

## Estimating Probabilities

Let  $N = |D|$  and let  $N(\textit{condition})$  be the number of exs. in  $D$  for which *condition* is true.

$$\hat{P}(y) = \frac{N(y)}{N} \quad \hat{P}(x_j \mid y) = \frac{N(x_j \wedge y)}{N(y)}$$

is problematic when  $N(x_j \wedge y) = 0$ .

It is better to use Laplace’s law of succession:

$$\hat{P}(x_j \mid y) = \frac{N(x_j \wedge y) + 1}{N(y) + k} \text{ where } x_j \text{ has } k \text{ possible vals.}$$

Or a more sophisticated  $m$ -estimate:

$$\hat{P}(x_j \mid y) = \frac{N(x_j \wedge y) + m/k}{N(y) + m}$$

## Naive Bayes Example: Learning

	$P(\text{good}) = 9/14$	$P(\text{bad}) = 5/14$
Attribute	$P(x \mid \text{good})$	$P(x \mid \text{bad})$
Outlook = sunny	3/12	4/8
Outlook = overcast	5/12	1/8
Outlook = rain	4/12	3/8
Temp = hot	3/12	3/8
Temp = mild	4/12	3/8
Temp = cool	5/12	2/8
Humidity = high	4/11	5/7
Humidity = normal	7/11	2/7
Windy = true	4/11	4/7
Windy = false	7/11	3/7

## Naive Bayes Example: Classification

$$P(\text{good} \mid (\text{rain, cool, normal, true}))$$

$$\propto (9/14)(4/12)(5/12)(7/11)(4/11) \approx 0.0207$$

$$P(\text{bad} \mid (\text{rain, cool, normal, true}))$$

$$\propto (5/14)(3/8)(2/8)(2/7)(4/7) \approx 0.0055$$

$$P(\text{good} \mid (\text{rain, mild, high, false}))$$

$$\propto (9/14)(4/12)(4/12)(4/11)(7/11) \approx 0.0165$$

$$P(\text{bad} \mid (\text{rain, mild, high, false}))$$

$$\propto (5/14)(3/8)(3/8)(5/7)(3/7) \approx 0.0154$$

## Comments on Multivariate Normal and Naive Bayes

- Both are very efficient! Only need one pass through examples.
- Surprisingly, NB is more effective. Often, several attributes give strong evidence for the class. Numeric attributes should be discretized. MV can perform badly when assumptions are wrong.
- NB is easy to interpret. One can identify attributes that support and oppose a classification.
- NB cannot model combinations of attributes properly.
- NB has linear decision boundary for nominal attributes. MN has hyperquadric decision boundary.

---

## Bayesian Networks

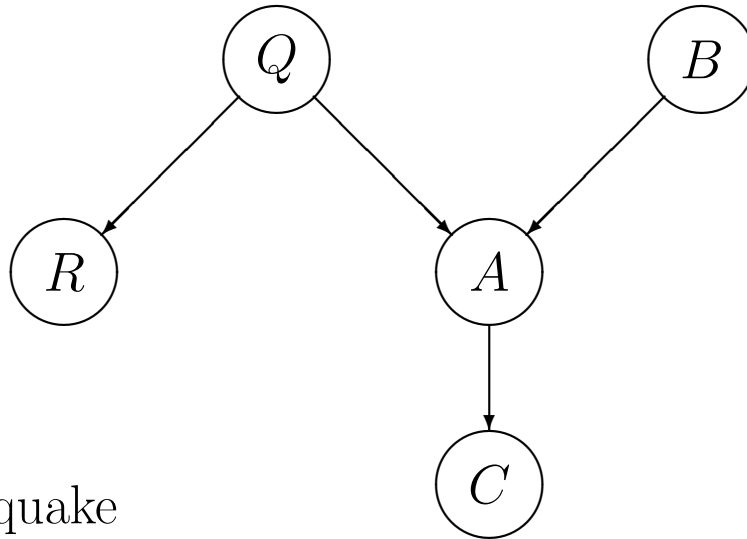
In Bayesian networks, the probability distribution is a product of conditional probabilities.

$$\begin{aligned} &P(x_1, \dots, x_n, y) \\ &= P(y \mid \text{parents}(y)) \prod_{j=1}^n P(x_j \mid \text{parents}(x_j)) \end{aligned}$$

where  $\text{parents}(y)$  and  $\text{parents}(x_j)$  are subsets of the variables, and the “parent-child” relation forms a directed acyclic graph.

The values of the parents should “cause” the value of the child.

# Example Bayesian Network



$Q$  = earthquake

$B$  = burglary

$A$  = house alarm

$R$  = radio announcement of earthquake

$C$  = phone call from neighbor

$P(Q)$	
<i>True</i>	<i>False</i>
0.0001	0.9999

$P(B)$	
<i>True</i>	<i>False</i>
0.001	0.999

$Q$	$P(R   Q)$	
	<i>True</i>	<i>False</i>
<i>True</i>	0.999	0.001
<i>False</i>	0.0005	0.9995

$Q$	$B$	$P(A   Q, B)$	
		<i>True</i>	<i>False</i>
<i>True</i>	<i>True</i>	0.999	0.001
<i>True</i>	<i>False</i>	0.8	0.2
<i>False</i>	<i>True</i>	0.95	0.05
<i>False</i>	<i>False</i>	0.001	0.999

$A$	$P(C   A)$	
	<i>True</i>	<i>False</i>
<i>True</i>	0.9	0.1
<i>False</i>	0.01	0.99

$$P(\neg Q \wedge B \wedge R \wedge \neg A \wedge C)$$

$$= P(\neg Q) P(B) P(R | \neg Q) P(\neg A | \neg Q \wedge B) P(C | \neg A)$$

$$= (0.9999) (0.001) (0.0005) (0.05) (0.01)$$

# Comments on Bayesian Networks

Variables can be nominal or numeric. Numeric variables need a probability density function (like normal dist.).

Inference algorithms can compute the probabilities of unknown variables from the known variables.

In practice, the probability computation is usually efficient, but, in theory, not guaranteed to be so.

Learning a Bayesian network depends on 2 conditions:

- Is the network structure known or unknown?
- Are all the variables observable or are some hidden?

---

## Bayes Risk Criterion

Choosing the most probable class might not be the best decision.

Consider deciding on turning onto an access road based on the probability of an accident.

Let  $L_{ij}$  = loss of deciding class  $i$  when class  $j$  is true.  $L_{ij}$  is the extra cost or risk of being wrong.

Bayes' risk criterion is to choose:

$$\operatorname{argmin}_i \sum_j L_{ij} P(y=j \mid \mathbf{x})$$

the class that minimizes loss.

Assuming two classes and  $L_{ii} = 0$ , choose class 1 if:

$$L_{21} P(y=1 | \mathbf{x}) > L_{12} P(y=2 | \mathbf{x})$$

This is equivalent to:

$$L_{21} P(\mathbf{x} | y=1) P(y=1) > L_{12} P(\mathbf{x} | y=2) P(y=2)$$

0-1 loss is defined as  $L_{ii} = 0$  otherwise  $L_{ij} = 1$ .

Note that Bayes' risk criterion with 0-1 loss is equivalent to MAP.