Chapter 18: Probabilistic Classification
Bayes Classifier

Let the training dataset $D$ consist of $n$ points $x_i$ in a $d$-dimensional space, and let $y_i$ denote the class for each point, with $y_i \in \{c_1, c_2, \ldots, c_k\}$.

The Bayes classifier estimates the posterior probability $P(c_i|x)$ for each class $c_i$, and chooses the class that has the largest probability. The predicted class for $x$ is given as

$$\hat{y} = \arg \max_{c_i} \{P(c_i|x)\}$$

According to the Bayes theorem, we have

$$P(c_i|x) = \frac{P(x|c_i) \cdot P(c_i)}{P(x)}$$

Because $P(x)$ is fixed for a given point, Bayes rule can be rewritten as

$$\hat{y} = \arg \max_{c_i} \{P(c_i|x)\} = \arg \max_{c_i} \left\{ \frac{P(x|c_i)P(c_i)}{P(x)} \right\} = \arg \max_{c_i} \{P(x|c_i)P(c_i)\}$$
Let $D_i$ denote the subset of points in $D$ that are labeled with class $c_i$:

$$D_i = \{x_j \in D \mid x_j \text{ has class } y_j = c_i\}$$

Let the size of the dataset $D$ be given as $|D| = n$, and let the size of each class-specific subset $D_i$ be given as $|D_i| = n_i$.

The prior probability for class $c_i$ can be estimated as follows:

$$\hat{P}(c_i) = \frac{n_i}{n}$$
Estimating the Likelihood: Numeric Attributes, Parametric Approach

To estimate the likelihood $P(x|c_i)$, we have to estimate the joint probability of $x$ across all the $d$ dimensions, i.e., we have to estimate $P(x = (x_1, x_2, \ldots, x_d)|c_i)$.

In the parametric approach we assume that each class $c_i$ is normally distributed, and we use the estimated mean $\hat{\mu}_i$ and covariance matrix $\hat{\Sigma}_i$ to compute the probability density at $x$

$$\hat{f}_i(x) = \hat{f}(x|\hat{\mu}_i, \hat{\Sigma}_i) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\hat{\Sigma}_i|}} \exp \left\{ -\frac{(x - \hat{\mu}_i)^T \hat{\Sigma}_i^{-1} (x - \hat{\mu}_i)}{2} \right\}$$

The posterior probability is then given as

$$P(c_i|x) = \frac{\hat{f}_i(x)P(c_i)}{\sum_{j=1}^{k} \hat{f}_j(x)P(c_j)}$$

The predicted class for $x$ is:

$$\hat{y} = \arg \max_{c_i} \left\{ \hat{f}_i(x)P(c_i) \right\}$$
Bayes Classifier Algorithm

\textsc{BayesClassifier} (D = \{(x_j, y_j)\}_{j=1}^{n}): 

1. \text{for } i = 1, \ldots, k \text{ do} 
2. \quad D_i \leftarrow \{x_j \mid y_j = c_i, j = 1, \ldots, n\} \text{ // class-specific subsets} 
3. \quad n_i \leftarrow |D_i| \text{ // cardinality} 
4. \quad \hat{P}(c_i) \leftarrow n_i / n \text{ // prior probability} 
5. \quad \hat{\mu}_i \leftarrow \frac{1}{n_i} \sum_{x_j \in D_i} x_j \text{ // mean} 
6. \quad Z_i \leftarrow D_i - 1_{n_i} \hat{\mu}_i^T \text{ // centered data} 
7. \quad \hat{\Sigma}_i \leftarrow \frac{1}{n_i} Z_i^T Z_i \text{ // covariance matrix} 
8. \text{return } \hat{P}(c_i), \hat{\mu}_i, \hat{\Sigma}_i \text{ for all } i = 1, \ldots, k 

\textsc{Testing} (x \text{ and } \hat{P}(c_i), \hat{\mu}_i, \hat{\Sigma}_i, \text{ for all } i \in [1, k]): 

\hat{y} \leftarrow \arg\max_{c_i} \{f(x \mid \hat{\mu}_i, \hat{\Sigma}_i) \cdot P(c_i)\} 

9. \text{return } \hat{y}
Bayes Classifier: Iris Data

\(X_1: \text{sepal length} \text{ versus } X_2: \text{sepal width}\)
Bayes Classifier: Categorical Attributes

Let $X_j$ be a categorical attribute over the domain $\text{dom}(X_j) = \{a_{j1}, a_{j2}, \ldots, a_{jm_j}\}$. Each categorical attribute $X_j$ is modeled as an $m_j$-dimensional multivariate Bernoulli random variable $X_j$ that takes on $m_j$ distinct vector values $e_{j1}, e_{j2}, \ldots, e_{jm_j}$, where $e_{jr}$ is the $r$th standard basis vector in $\mathbb{R}^{m_j}$ and corresponds to the $r$th value or symbol $a_{jr} \in \text{dom}(X_j)$.

The entire $d$-dimensional dataset is modeled as the vector random variable $X = (X_1, X_2, \ldots, X_d)^T$. Let $d' = \sum_{j=1}^{d} m_j$; a categorical point $x = (x_1, x_2, \ldots, x_d)^T$ is therefore represented as the $d'$-dimensional binary vector

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} e_{1r_1} \\ \vdots \\ e_{dr_d} \end{pmatrix}$$

where $v_j = e_{jr_j}$ provided $x_j = a_{jr_j}$ is the $r_j$th value in the domain of $X_j$. 
The probability of the categorical point $x$ is obtained from the joint probability mass function (PMF) for the vector random variable $X$:

$$P(x|c_i) = f(v|c_i) = f(X_1 = e_{r_1}, \ldots, X_d = e_{d_{r_d}} | c_i)$$

The joint PMF can be estimated directly from the data $D_i$ for each class $c_i$ as follows:

$$\hat{f}(v|c_i) = \frac{n_i(v)}{n_i}$$

where $n_i(v)$ is the number of times the value $v$ occurs in class $c_i$. However, to avoid zero probabilities we add a pseudo-count of 1 for each value

$$\hat{f}(v|c_i) = \frac{n_i(v) + 1}{n_i + \prod_{j=1}^{d} m_j}$$
Discretized Iris Data: \textit{sepal length} and \textit{sepal width}

(a) Discretized \textit{sepal length}:

<table>
<thead>
<tr>
<th>Bins</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4.3, 5.2]</td>
<td>Very Short ($a_{11}$)</td>
</tr>
<tr>
<td>(5.2, 6.1]</td>
<td>Short ($a_{12}$)</td>
</tr>
<tr>
<td>(6.1, 7.0]</td>
<td>Long ($a_{13}$)</td>
</tr>
<tr>
<td>(7.0, 7.9]</td>
<td>Very Long ($a_{14}$)</td>
</tr>
</tbody>
</table>

(b) Discretized \textit{sepal width}:

<table>
<thead>
<tr>
<th>Bins</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2.0, 2.8]</td>
<td>Short ($a_{21}$)</td>
</tr>
<tr>
<td>(2.8, 3.6]</td>
<td>Medium ($a_{22}$)</td>
</tr>
<tr>
<td>(3.6, 4.4]</td>
<td>Long ($a_{23}$)</td>
</tr>
</tbody>
</table>
### Class-specific Empirical Joint Probability Mass Function

#### Class: $c_1$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$\hat{f}_{X_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Short ($e_{11}$)</td>
<td>1/50, 33/50, 5/50</td>
<td>39/50</td>
</tr>
<tr>
<td>Short ($e_{12}$)</td>
<td>0, 3/50, 8/50</td>
<td>13/50</td>
</tr>
<tr>
<td>Long ($e_{13}$)</td>
<td>0, 0, 0</td>
<td>0</td>
</tr>
<tr>
<td>Very Long ($e_{14}$)</td>
<td>0, 0, 0</td>
<td>0</td>
</tr>
<tr>
<td>$\hat{f}_{X_2}$</td>
<td>1/50, 36/50, 13/50</td>
<td></td>
</tr>
</tbody>
</table>

#### Class: $c_2$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$\hat{f}_{X_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Short ($e_{11}$)</td>
<td>6/100, 0, 0</td>
<td>6/100</td>
</tr>
<tr>
<td>Short ($e_{12}$)</td>
<td>24/100, 15/100, 0</td>
<td>39/100</td>
</tr>
<tr>
<td>Long ($e_{13}$)</td>
<td>13/100, 30/100, 0</td>
<td>43/100</td>
</tr>
<tr>
<td>Very Long ($e_{14}$)</td>
<td>3/100, 7/100, 2/100</td>
<td>12/100</td>
</tr>
<tr>
<td>$\hat{f}_{X_2}$</td>
<td>46/100, 52/100, 2/100</td>
<td></td>
</tr>
</tbody>
</table>
Iris Data: Test Case

Consider a test point \( \mathbf{x} = (5.3, 3.0)^T \) corresponding to the categorical point (Short, Medium), which is represented as \( \mathbf{v} = (\mathbf{e}_{12}^T \mathbf{e}_{22}^T)^T \).

The prior probabilities of the classes are \( \hat{P}(c_1) = 0.33 \) and \( \hat{P}(c_2) = 0.67 \). The likelihood and posterior probability for each class is given as

\[
\begin{align*}
\hat{P}(\mathbf{x}|c_1) &= \hat{f}(\mathbf{v}|c_1) = 3/50 = 0.06 \\
\hat{P}(\mathbf{x}|c_2) &= \hat{f}(\mathbf{v}|c_2) = 15/100 = 0.15 \\
\hat{P}(c_1|\mathbf{x}) &\propto 0.06 \times 0.33 = 0.0198 \\
\hat{P}(c_2|\mathbf{x}) &\propto 0.15 \times 0.67 = 0.1005
\end{align*}
\]

In this case the predicted class is \( \hat{y} = c_2 \).
Iris Data: Test Case with Pesudo-counts

The test point \( \mathbf{x} = (6.75, 4.25)^T \) corresponds to the categorical point (Long, Long), and it is represented as \( \mathbf{v} = (\mathbf{e}_{13}^T, \mathbf{e}_{23}^T)^T \).

Unfortunately the probability mass at \( \mathbf{v} \) is zero for both classes. We adjust the PMF via pseudo-counts noting that the number of possible values are \( m_1 \times m_2 = 4 \times 3 = 12 \).

The likelihood and prior probability can then be computed as

\[
\hat{P}(\mathbf{x}|c_1) = \hat{f}(\mathbf{v}|c_1) = \frac{0 + 1}{50 + 12} = 1.61 \times 10^{-2}
\]

\[
\hat{P}(\mathbf{x}|c_2) = \hat{f}(\mathbf{v}|c_2) = \frac{0 + 1}{100 + 12} = 8.93 \times 10^{-3}
\]

\[
\hat{P}(c_1|\mathbf{x}) \propto (1.61 \times 10^{-2}) \times 0.33 = 5.32 \times 10^{-3}
\]

\[
\hat{P}(c_2|\mathbf{x}) \propto (8.93 \times 10^{-3}) \times 0.67 = 5.98 \times 10^{-3}
\]

Thus, the predicted class is \( \hat{y} = c_2 \).
The main problem with the Bayes classifier is the lack of enough data to reliably estimate the joint probability density or mass function, especially for high-dimensional data.

For numeric attributes we have to estimate $O(d^2)$ covariances, and as the dimensionality increases, this requires us to estimate too many parameters.

For categorical attributes we have to estimate the joint probability for all the possible values of $v$, given as $\prod_j |\text{dom}(X_j)|$. Even if each categorical attribute has only two values, we would need to estimate the probability for $2^d$ values. However, because there can be at most $n$ distinct values for $v$, most of the counts will be zero.

Naive Bayes classifier addresses these concerns.
The naive Bayes approach makes the simple assumption that all the attributes are independent, which implies that the likelihood can be decomposed into a product of dimension-wise probabilities:

$$P(x|c_i) = P(x_1, x_2, \ldots, x_d|c_i) = \prod_{j=1}^{d} P(x_j|c_i)$$

The likelihood for class $c_i$, for dimension $X_j$, is given as

$$P(x_j|c_i) \propto f(x_j|\hat{\mu}_{ij}, \hat{\sigma}^2_{ij}) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_{ij}} \exp \left\{ -\frac{(x_j - \hat{\mu}_{ij})^2}{2\hat{\sigma}^2_{ij}} \right\}$$

where $\hat{\mu}_{ij}$ and $\hat{\sigma}^2_{ij}$ denote the estimated mean and variance for attribute $X_j$, for class $c_i$. 
The naive assumption corresponds to setting all the covariances to zero in $\hat{\Sigma}_i$, that is,

$$\Sigma_i = \begin{pmatrix}
\sigma_{i1}^2 & 0 & \cdots & 0 \\
0 & \sigma_{i2}^2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma_{id}^2
\end{pmatrix}$$

The naive Bayes classifier thus uses the sample mean $\hat{\mu}_i = (\hat{\mu}_{i1}, \ldots, \hat{\mu}_{id})^T$ and a diagonal sample covariance matrix $\hat{\Sigma}_i = \text{diag}(\sigma_{i1}^2, \ldots, \sigma_{id}^2)$ for each class $c_i$. In total $2d$ parameters have to be estimated, corresponding to the sample mean and sample variance for each dimension $X_j$. 
Naive Bayes Algorithm

\[
\text{NAIVEBAYES (D = \{(x_j, y_j)\}_{j=1}^n)}:
\]

1. for \( i = 1, \ldots, k \) do
2. \( D_i \leftarrow \{x_j \mid y_j = c_i, j = 1, \ldots, n\} \) // class-specific subsets
3. \( n_i \leftarrow |D_i| \) // cardinality
4. \( \hat{P}(c_i) \leftarrow n_i/n \) // prior probability
5. \( \hat{\mu}_i \leftarrow \frac{1}{n_i} \sum_{x_j \in D_i} x_j \) // mean
6. \( Z_i = D_i - \mathbf{1} \cdot \hat{\mu}_i^T \) // centered data for class \( c_i \)
7. for \( j = 1, \ldots, d \) do // class-specific variance for \( X_j \)
8. \( \hat{\sigma}_{ij}^2 \leftarrow \frac{1}{n_i} Z_{ij}^T Z_{ij} \) // variance
9. \( \hat{\sigma}_i = (\hat{\sigma}_{i1}^2, \ldots, \hat{\sigma}_{id}^2)^T \) // class-specific attribute variances
10. return \( \hat{P}(c_i), \hat{\mu}_i, \hat{\sigma}_i \) for all \( i = 1, \ldots, k \)

\[
\text{TESTING (x and } \hat{P}(c_i), \hat{\mu}_i, \hat{\sigma}_i, \text{for all } i \in [1, k]):\n\]

\( \hat{y} \leftarrow \arg \max_{c_i} \left\{ \hat{P}(c_i) \prod_{j=1}^d f(x_j | \hat{\mu}_{ij}, \hat{\sigma}_{ij}^2) \right\} \)

11. return \( \hat{y} \)
Naive Bayes versus Full Bayes Classifier: Iris 2D Data

$x_1$: sepal length versus $x_2$: sepal width

(a) Naive Bayes

(b) Full Bayes

$\mathbf{x} = (6.75, 4.25)^T$
Naive Bayes: Categorical Attributes

The independence assumption leads to a simplification of the joint probability mass function

\[ P(x|c_i) = \prod_{j=1}^{d} P(x_j|c_i) = \prod_{j=1}^{d} f(x_j = e_{jr_j} | c_i) \]

where \( f(x_j = e_{jr_j} | c_i) \) is the probability mass function for \( X_j \), which can be estimated from \( D_i \) as follows:

\[ \hat{f}(v_j|c_i) = \frac{n_i(v_j)}{n_i} \]

where \( n_i(v_j) \) is the observed frequency of the value \( v_j = e_{jr_j} \) corresponding to the \( r_j \)th categorical value \( a_{jr_j} \) for the attribute \( X_j \) for class \( c_i \).

If the count is zero, we can use the pseudo-count method to obtain a prior probability. The adjusted estimates with pseudo-counts are given as

\[ \hat{f}(v_j|c_i) = \frac{n_i(v_j) + 1}{n_i + m_j} \]

where \( m_j = |\text{dom}(X_j)| \).
Nonparametric Approach: $K$ Nearest Neighbors Classifier

We consider a non-parametric approach for likelihood estimation using the nearest neighbors density estimation. Let $D$ be a training dataset comprising $n$ points $x_i \in \mathbb{R}^d$, and let $D_i$ denote the subset of points in $D$ that are labeled with class $c_i$, with $n_i = |D_i|$.

Given a test point $x \in \mathbb{R}^d$, and $K$, the number of neighbors to consider, let $r$ denote the distance from $x$ to its $K$th nearest neighbor in $D$.

Consider the $d$-dimensional hyperball of radius $r$ around the test point $x$, defined as

$$B_d(x, r) = \{ x_i \in D \mid \delta(x, x_i) \leq r \}$$

Here $\delta(x, x_i)$ is the distance between $x$ and $x_i$, which is usually assumed to be the Euclidean distance, i.e., $\delta(x, x_i) = \| x - x_i \|_2$. We assume that $|B_d(x, r)| = K$. 
Nonparametric Approach: $K$ Nearest Neighbors Classifier

Let $K_i$ denote the number of points among the $K$ nearest neighbors of $x$ that are labeled with class $c_i$, that is

$$K_i = \left\{ x_j \in B_d(x, r) \mid y_j = c_i \right\}$$

The class conditional probability density at $x$ can be estimated as the fraction of points from class $c_i$ that lie within the hyperball divided by its volume, that is

$$\hat{f}(x|c_i) = \frac{K_i}{n_i} = \frac{K_i}{n_i V}$$

where $V = \text{vol}(B_d(x, r))$ is the volume of the $d$-dimensional hyperball.

The posterior probability $P(c_i|x)$ can be estimated as

$$P(c_i|x) = \frac{\hat{f}(x|c_i) \hat{P}(c_i)}{\sum_{j=1}^{k} \hat{f}(x|c_j) \hat{P}(c_j)}$$

However, because $\hat{P}(c_i) = \frac{n_i}{n}$, we have

$$\hat{f}(x|c_i) \hat{P}(c_i) = \frac{K_i}{n_i V} \cdot \frac{n_i}{n} = \frac{K_i}{n V}$$
Nonparametric Approach: \( K \) Nearest Neighbors Classifier

The posterior probability is given as

\[
P(c_i|x) = \frac{K_i}{nV} = \frac{K_i}{K}
\]

Finally, the predicted class for \( x \) is

\[
\hat{y} = \arg \max_{c_i} \{ P(c_i|x) \} = \arg \max_{c_i} \left\{ \frac{K_i}{K} \right\} = \arg \max_{c_i} \{ K_i \}
\]

Because \( K \) is fixed, the KNN classifier predicts the class of \( x \) as the majority class among its \( K \) nearest neighbors.
Iris Data: $K$ Nearest Neighbors Classifier