CS5489
Lecture 2.1: Naive Bayes and Linear Discriminant Analysis

Kede Ma

City University of Hong Kong (Dongguan)

Slide template by courtesy of Benjamin M. Marlin

The Classification Task

Definition: The Classification Task

Given a feature vector $\mathbf{x} \in \mathbb{R}^N$ that describes an object that belongs to one of C classes from the set $\mathcal{Y}$, predict which class the object belongs to

Definition: Classifier Learning

Given a data set of example pairs $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, M\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^N$ is a feature vector and $y^{(i)} \in \mathcal{Y} = \{1, \ldots, C\}$ is a class label, learn a function $f : \mathbb{R}^N \mapsto \mathcal{Y}$ that accurately predicts the class label y for any feature vector $\mathbf{x}$

Classification Error and Accuracy

Definition: Classification Error Rate

Given a data set of example pairs $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, M\}$ and a function $f : \mathbb{R}^N \mapsto \mathcal{Y}$, the classification error rate of f on $\mathcal{D}$ is:

$$\text{Err}(f, \mathcal{D}) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}[y^{(i)} \neq f(\mathbf{x}^{(i)})]$$

The Bayes Optimal Classifier

- The Bayes optimal classifier uses the classification function:

$$
\begin{aligned}
f_{\mathrm{B}}(\mathbf{x}) &= \arg\max_{c \in \{1,\ldots,C\}} p(y = c|\mathbf{x}) \\
&= \arg\max_{c \in \{1,\ldots,C\}} p(\mathbf{x}|y = c)p(y = c) \\
&= \arg\max_{c \in \{1,\ldots,C\}} \log p(\mathbf{x}|y = c) + \log p(y = c)
\end{aligned}
$$

- The Bayes optimal classifier is the best classifier among all possible classifiers. However, it is not useful due to the difficulty of estimating $p(\mathbf{x}|y = c)$ in practice

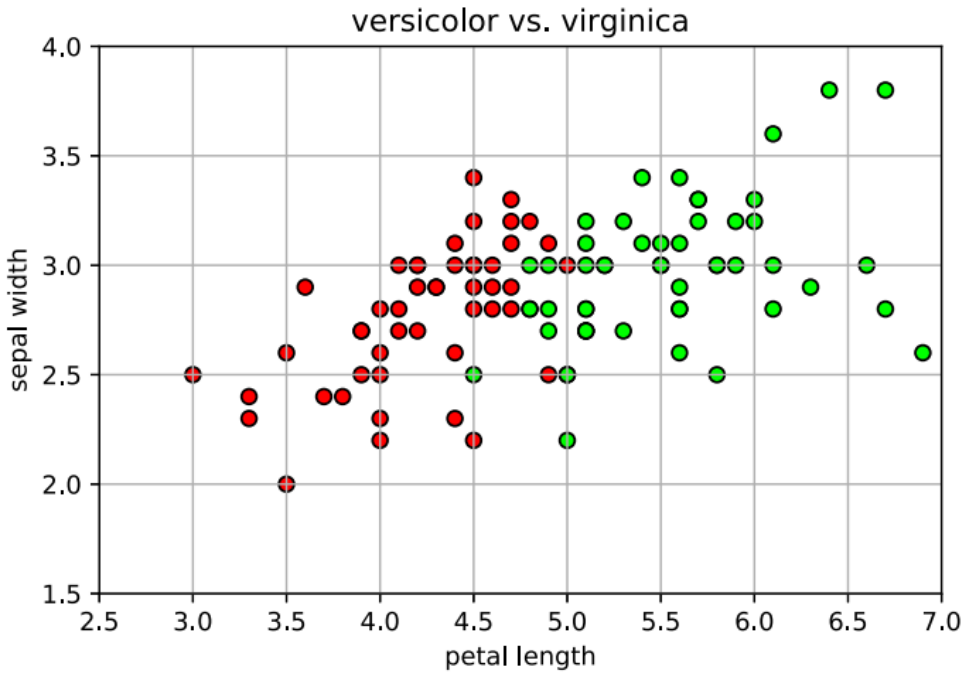
Naive Bayes

- How to deal with multiple features? E.g., $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$
- Naive Bayes assumption
  - The naive Bayes classifier approximates the Bayes optimal classifier using a simple form for the functions $p(\mathbf{x}|y = c)$
  - This form assumes that all of the feature dimensions are statistically independent given the value of the class variable:
    - E.g., for 2 dimensions, $p(x_1, x_2|y) = p(x_1|y)p(x_2|y)$
  - Accumulates evidence from each feature dimension:
    - E.g., $\log p(x_1, x_2|y) = \log p(x_1|y) + \log p(x_2|y)$
  - Allows us to model each dimension of the observation with a simple univariate distribution
- The general form for the classification function is:

$$f_{\text{NB}}(\mathbf{x}) = \underset{c \in \{1,\ldots,C\}}{\arg\max}\, p(y = c) \prod_{j=1}^{N} p(x_j|y = c)$$

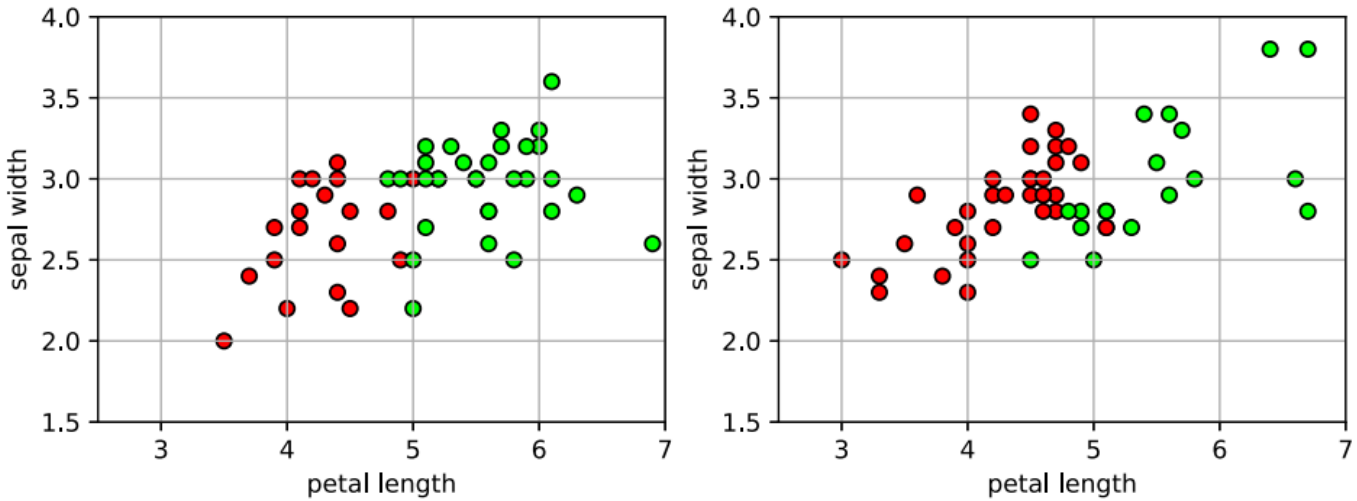
Example: Gaussian Classifier

- We will consider the 2-dimensional iris data shown in the beginning of the lecture



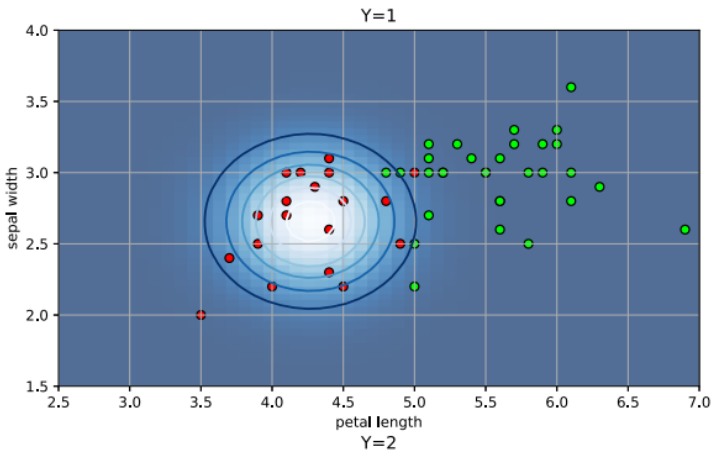
Example: Gaussian Classifier

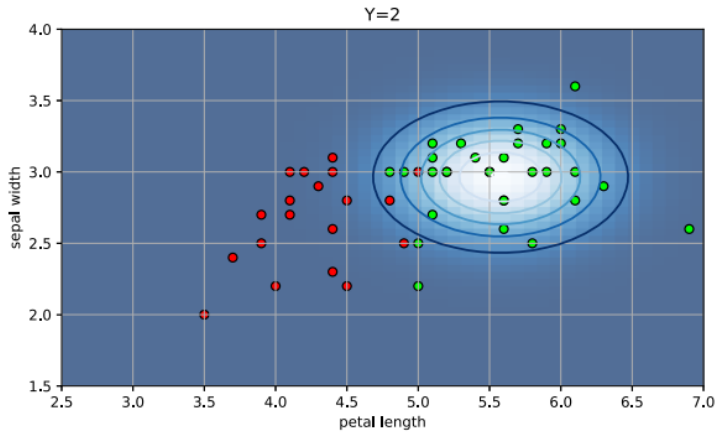
- We will select 50% of the data for training, and 50% for testing



Example: Gaussian Classifier

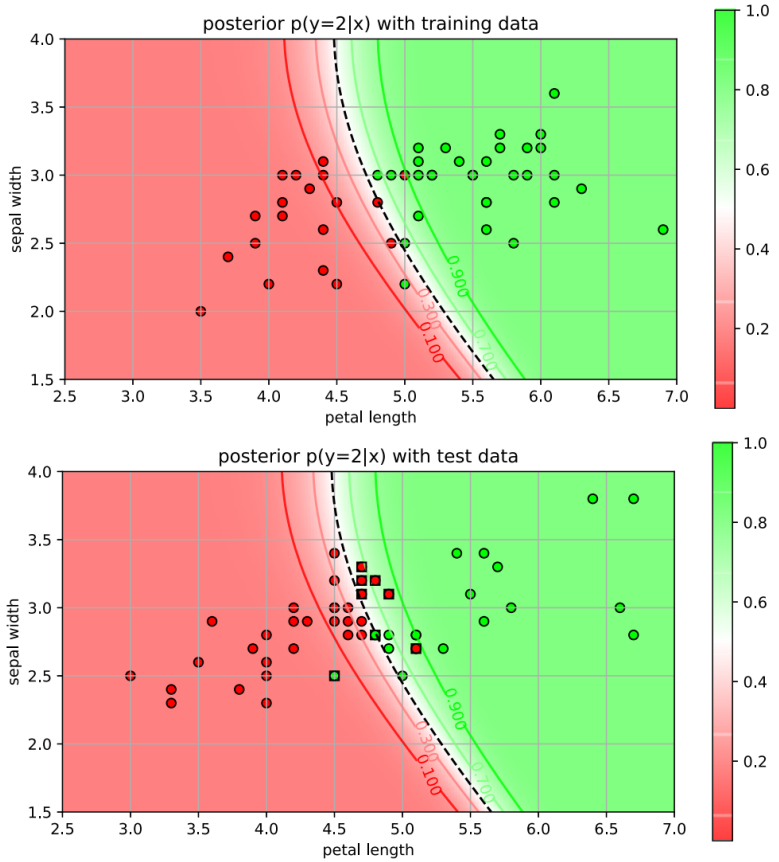
- Learn Gaussian Naive Bayes model
  - Treat each dimension as an independent Gaussian and fit the Gaussian parameters with MLE
- View 2d class conditionals:
  - $p(x_1, x_2|y = c) = p(x_1|y = c)p(x_2|y = c)$





Example: Gaussian Classifier

- View the posterior $p(y = c|\mathbf{x}) = p(y = c|x_1, x_2)$



Geometric Interpretation

- What is the geometry of the decision boundary?

$$
\begin{aligned}
f_{\mathrm{NB}}(\mathbf{x}) &= \arg\max_{c\in\{1,2\}} p(y=c)\prod_{j=1}^{2} p(x_j|y=c) \\
&= \arg\max_{c\in\{1,2\}} \log p(y=c) + \sum_{j=1}^{2} \log p(x_j|y=c) \\
&= \arg\max_{c\in\{1,2\}} \log p(y=c) + \sum_{j=1}^{2} \left(-\frac{1}{2}\log(2\pi\sigma_{j|c}^2) - \frac{(x_j-\mu_{j|c})^2}{2\sigma_{j|c}^2} \right)
\end{aligned}
$$

Geometric Interpretation

- The decision boundary consists of the set of points (x_1, x_2) where $\log p(y=1) + \sum_{j=1}^{2} \log p(x_j|y=1) = \log p(y=2) + \sum_{j=1}^{2} \log p(x_j|y=2)$:

$$\log p(y=1) + \sum_{j=1}^{2} \left(-\frac{1}{2} \log(2\pi\sigma_{j|1}^2) - \frac{1}{2\sigma_{j|1}^2}(x_j - \mu_{j|1})^2 \right)$$
$$- \log p(y=2) - \sum_{j=1}^{2} \left(-\frac{1}{2} \log(2\pi\sigma_{j|2}^2) - \frac{1}{2\sigma_{j|2}^2}(x_j - \mu_{j|2})^2 \right) = 0$$

- It's not hard to see that the decision boundary is a quadratic function of (x_1, x_2) with the form: $\sum_{j=1}^{2} \left(a_j x_j^2 + b_j x_j \right) + c = 0$

Naive Bayes Model For Boolean Vectors

- Model each feature independently
  - Absence/presence of a feature x_j in an input example
  - Bernoulli distribution
    - Present: $p(x_j = 1|y = c) = \varphi_{j|c}$
    - Absent: $p(x_j = 0|y = c) = 1 - \varphi_{j|c}$
  - MLE paramters: $\varphi_{j|c} = M_{j|c}/M_c$
    - $M_{j|c}$ is the number of training examples in Class c that contain feature x_j
    - M_c is the number of training examples in Class c
- Class-conditional distribution
  - $p(x_1, \ldots, x_N|y = c) = \prod_{j=1}^{N} p(x_j|y = c)$
  - $\log p(x_1, \ldots, x_N|y = c) = \sum_{j=1}^{N} \log p(x_j|y = c)$
- For an input example $\mathbf{x}$, the log-probabilities of features present given $y = c$ adds

Smoothing

- Some features may not be present in any training examples for a given class
  - $M_{j|c} = 0$ and thus $\varphi_{j|c} = 0$
    - I.e., examples in the class **definitely** will not contain the feature
    - Can be a problem since we simply may not have seen an example with that feature due to the limited size of the training set
- Smoothed MLE
  - Add a smoothing parameter α, i.e., adding a "virtual" count
  - Parameter: $\varphi_{j|c} = (M_{j|c} + \alpha)/(M_c + N\alpha)$
  - If $\alpha = 1$, this is called **Laplace smoothing** (or additive smoothing)
- In general, regularizing or smoothing of the estimate helps to prevent **overfitting** of the parameters

Multinomial Naive Bayes

- What if x_j has a finite set of categories, i.e., $x_j \in \mathcal{X}_j = \{1, \ldots, |\mathcal{X}_j|\}$, instead of only two states of absence and presence with $x_j \in \{1, 2\}$
  - Use a multinomial (or called categorical) distribution as class conditional
    - $p(x_j = x | y = c) = \varphi_{x,j|c}$
    - $\sum_{x \in \mathcal{X}_j} \varphi_{x,j|c} = 1$
  - MLE paramters: $\varphi_{x,j|c} = M_{x,j|c} / M_c$
    - $M_{x,j|c}$ is the number of training examples in Class c with feature $x_j = x$
    - M_c is the number of training examples in Class c

Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) is a classification technique due to Fisher that dates back to the 1930's
- It can be interpreted as a different approximation to the Bayes optimal classifier for real-valued data
- Instead of a product of independent Gaussians in Naive Bayes, LDA assumes $p(\mathbf{x}|y=c) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma})$, *i.e.*, multivariate Gaussian with a shared covariance matrix

$$p(\mathbf{x}|y=c) = \frac{1}{|(2\pi)^N \boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)\right)$$

- The classification function is

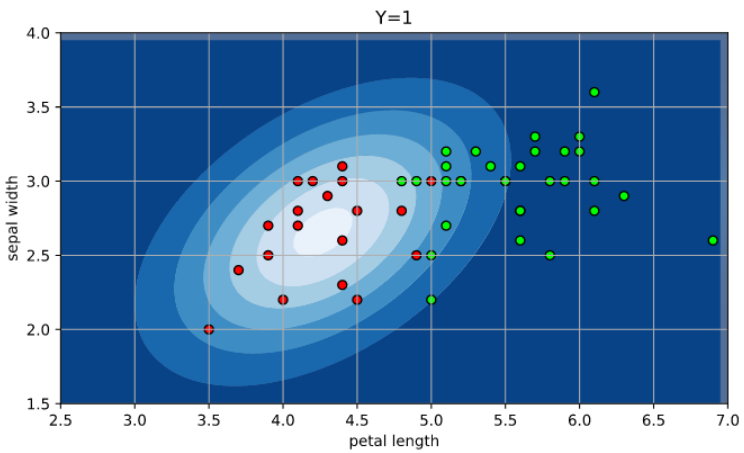
$$f_{\text{LDA}}(\mathbf{x}) = \arg\max_{c \in \{1,\ldots,C\}} p(y=c)p(\mathbf{x}|y=c)$$

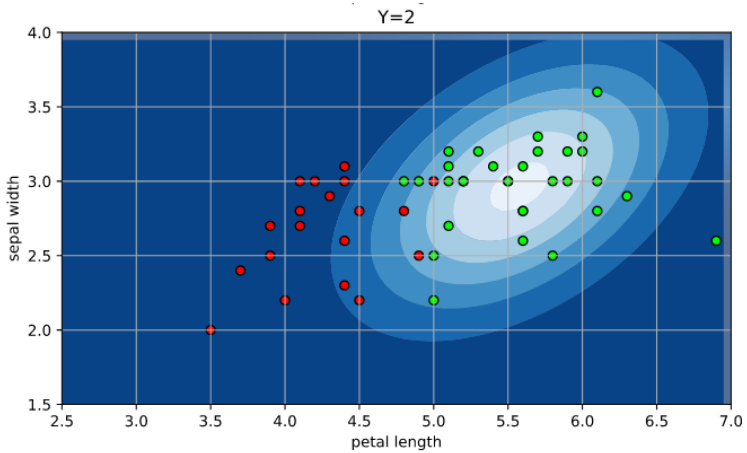
Learning for LDA

- As with naive Bayes, LDA parameters are learned using MLE, which reduces to using sample estimates
- Class probabilities: $p(y = c) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}[y^{(i)} = c] = \frac{M_c}{M}$
- Class means: $\boldsymbol{\mu}_c = \frac{\sum_{i=1}^{M} \mathbb{I}[y^{(i)}=c]\mathbf{x}^{(i)}}{\sum_{i=1}^{M} \mathbb{I}[y^{(i)}=c]} = \frac{\sum_{i=1}^{M_c} \mathbf{x}^{(i)}}{M_c}$, where $\mathbf{x}^{(i)}$ belongs to Class c
- Shared covariance: $\boldsymbol{\Sigma} = \frac{1}{M} \sum_{i=1}^{M} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})^T$

Example: Gaussian Classifier

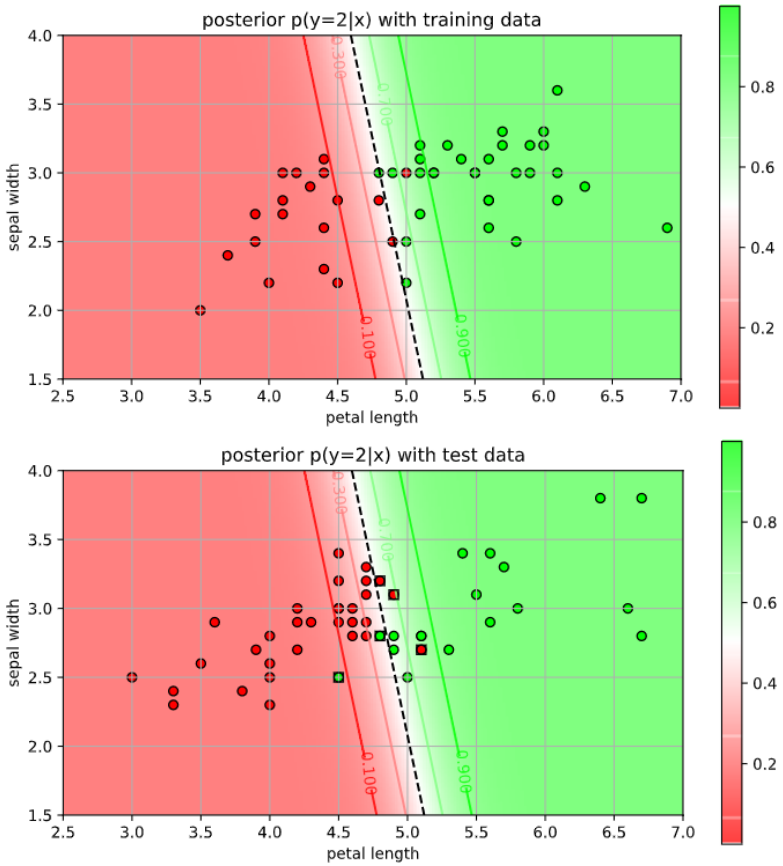
- View 2d class conditionals:
  - $p(x_1, x_2|y = c) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma})$





Example: Gaussian Classifier

- View the posterior



Geometric Interpretation

- What is the geometry of the decision boundary for LDA?
- The decision boundary consists of the set of points $\mathbf{x}$ where:

$$\log p(y=1) - \frac{1}{2}\log|(2\pi)^2\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)$$
$$-\log p(y=2) + \frac{1}{2}\log|(2\pi)^2\boldsymbol{\Sigma}| + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) = 0$$

- We can cancel a large number of terms because of the common covariance matrix and obtain the following result:

$$\log\frac{p(y=1)}{p(y=2)} - \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}\mathbf{x} = 0$$

- This shows that the decision boundary is actually linear in $\mathbf{x}$

NB Versus LDA

- Storage: NB requires $\mathcal{O}(N)$ parameters, whereas LDA requires $\mathcal{O}(N^2)$ parameters
- Speed: The quadratic dependence on N makes LDA slower than NB during learning and classification
- Interpretability: Both models have good interpretability since the parameters of $p(\mathbf{x}|y = c)$ correspond to class conditional averages
- Data: LDA will generally need more data than NB since it needs to estimate the $O(N^2)$ parameters in the shared covariance matrix

Summary

- Generative classification model
  - Estimate probability distributions of features from each class
  - Given feature, predict class with the largest posterior probability
- Advantages:
  - Works with small amount of data
  - Works with multiple classes
- Disadvantages:
  - Accuracy depends on selecting an appropriate probability distribution
    - If the probability distribution doesn't model the data well, then accuracy might be bad