Machine Learning

8 – Naïve Bayes, Logistic Regression

Avisek Gupta
Postdoctoral Fellow, IAI, TCG CREST
avisek003@gmail.com

September 09, 2022

Recap of the previous class: Discriminative vs. Generative Models

Bayes Decision Rule to attain the Bayes Risk R^*:

Decide class w_i where

$$P(w_i|x) > P(w_j|x) \ \forall j \neq i$$

Estimate posterior probabilities

Discriminative Methods:

- Logistic Regression
- k-Nearest Neighbours
- Multi-Layered Perceptrons
- Support Vector Machines
- Random Forests
- …

Decide class w_i where

$$p(x|w_i)\,P(w_i) > p(x|w_j)\,P(w_j) \ \forall j \neq i$$

Estimate (i) class-conditional densities and (ii) prior probabilities

Generative Methods:

- Naive Bayes Classifier
- Hidden Markov Models
- Variational Autoencoders
- Generative Adversarial Networks
- …

Recap of the previous class: Discriminative vs. Generative Models

Discriminative Models

Decide class w_i where

$$P(w_i|x) > P(w_j|x) \; \forall j \neq i$$

Estimate posterior probabilities

Generative Models

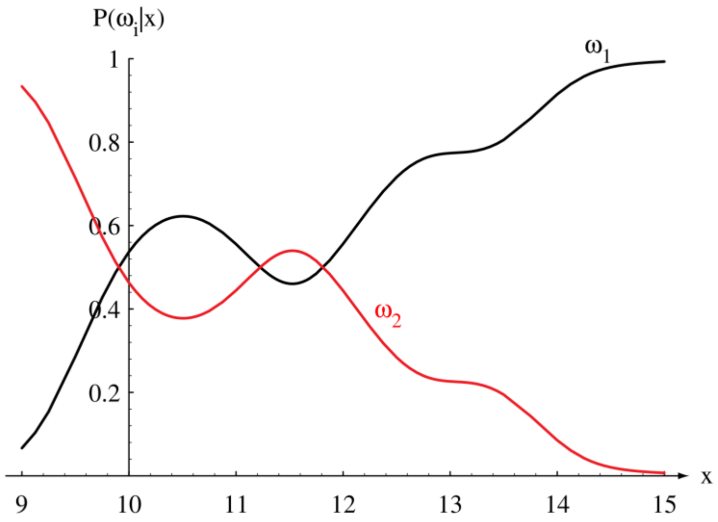
Decide class w_i where

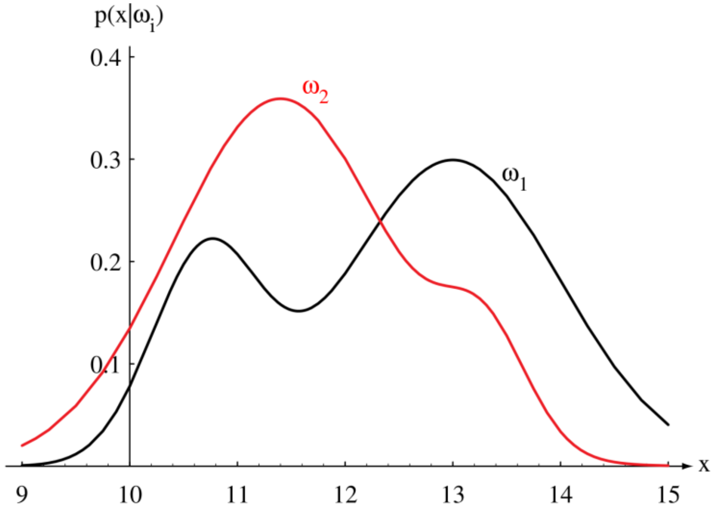
$$p(x|w_i)\, P(w_i) > p(x|w_j)\, P(w_j) \; \forall j \neq i$$

Estimate (i) class-conditional densities and (ii) prior probabilities

How can we estimate posterior or prior or class-conditional densities?

- w_i are discrete random variables, x is continuous.





*Images Source: Duda Hart Stork - Pattern Classification

Recap of the previous class: Discriminative vs. Generative Models

<u>Generative Models</u>

Decide class w_i where

$$p(x|w_i)\,P(w_i) > p(x|w_j)\,P(w_j)\ \forall j \neq i$$

Estimate (i) class-conditional densities and
(ii) prior probabilities

How can we estimate probability densities?

- Parametric Estimation: We assume a probability density function can be estimated by a parametric distribution, where parameters of the distribution can fully describe the distribution. E.g., μ, $\mathbf{\Sigma}$ completely describe a Gaussian distribution.
- Non-parametric Estimation: We estimate a function that describes a desired probability density function as closely as possible.

Recap of the previous class: Discriminative vs. Generative Models

<u>Generative Models</u>

Decide class w_i where

$$p(x|w_i)\,P(w_i) > p(x|w_j)\,P(w_j)\ \forall j \neq i$$

Estimate (i) class-conditional densities and
(ii) prior probabilities

How can we estimate probability densities?

- Parametric Estimation: We assume a probability density function can be estimated by a parametric distribution, where parameters of the distribution can fully describe the distribution. E.g., μ, $\mathbf{\Sigma}$ completely describe a Gaussian distribution.

Parametric Estimation Procedures:

- Maximum Likelihood Estimation
- Bayesian Estimation
- …

Recap of the previous class: Maximum Likelihood Estimation

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be n observed samples. We wish to estimate the parameters Θ of a distribution so that the probability of observing the n samples is maximized. This objective is also expressed as: we wish to estimate the distribution parameters so that the *likelihood* of observing the n samples is maximized. Formally, a *likelihood function* is defined as:

$$L(\Theta) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n | \Theta)$$

The Maximum Likelihood Estimation procedure is described as:

Estimate $\hat{\Theta}$ that maximizes $L(\Theta)$

Recap of the previous class: Maximum Likelihood Estimation

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be n observed samples. We wish to estimate the parameters Θ of a distribution so that the probability of observing the n samples is maximized. This objective is also expressed as: we wish to estimate the distribution parameters so that the *likelihood* of observing the n samples is maximized. Formally, a *likelihood function* is defined as:

$$L(\Theta) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n | \Theta)$$

The Maximum Likelihood Estimation procedure is described as:

Estimate $\hat{\Theta}$ that maximizes $L(\Theta)$

If we assume $\mathbf{x}_1, \ldots, \mathbf{x}_n$ were drawn independently, then the likelihood can be written as,

$$L(\Theta) = \prod_{i=1}^{n} p(\mathbf{x}_i | \Theta)$$

We can then consider the log likelihood function:

$$\ell(\Theta) = \ln L(\Theta) = \sum_{i=1}^{n} \ln p(\mathbf{x}_i | \Theta)$$

Recap of the previous class: Maximum Likelihood Estimation

If we assume $\mathbf{x}_1, \ldots, \mathbf{x}_n$ were drawn independently, then the likelihood can be written as,

$$L(\Theta) = \prod_{i=1}^{n} p(\mathbf{x}_i|\Theta)$$

We can then consider the log likelihood function:

$$\ell(\Theta) = \ln L(\Theta) = \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\Theta)$$

The Maximum Likelihood Estimation procedure can then be equivalently described as:

Estimate $\hat{\Theta}$ that maximizes $\ell(\Theta)$

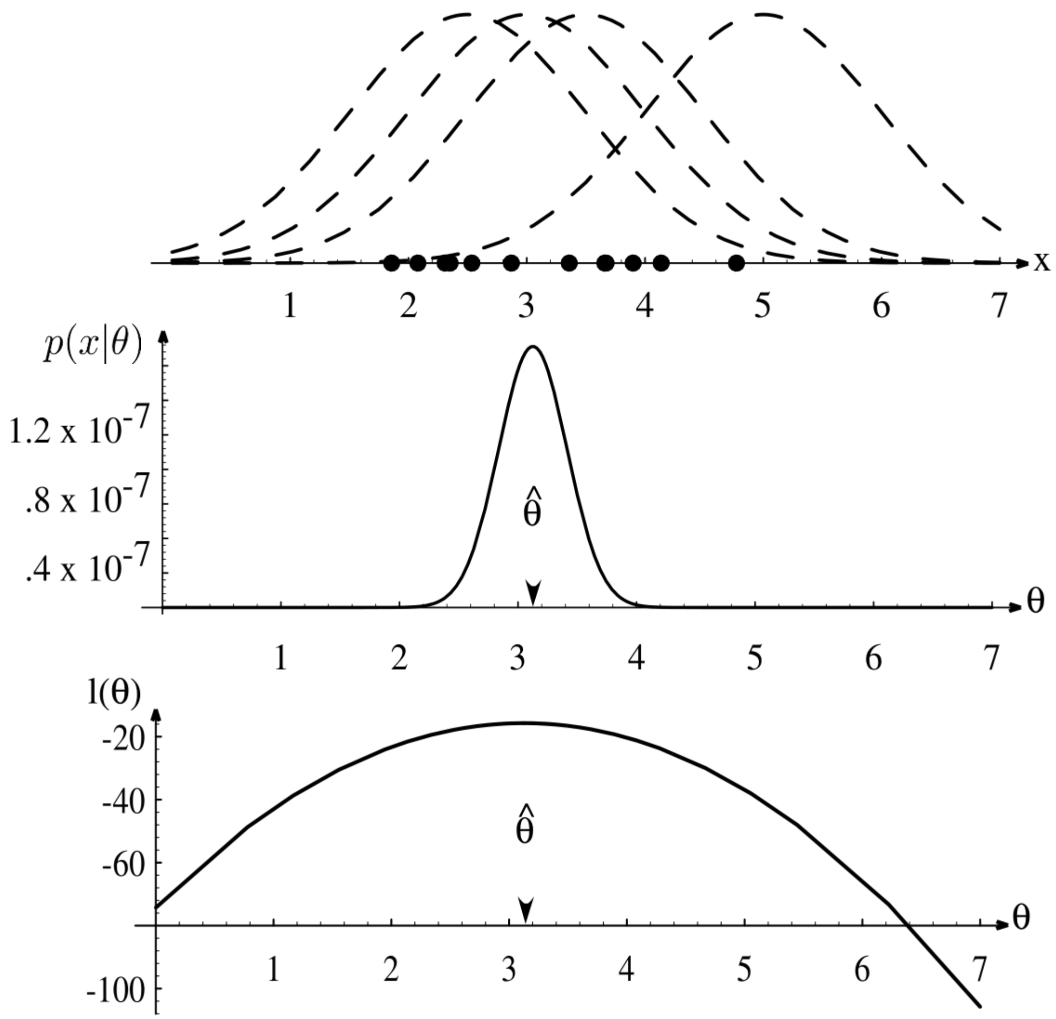


*Image Source: Duda Hart Stork - Pattern Classification

Recap of the previous class: Maximum Likelihood Estimation

If we assume $\mathbf{x}_1, \ldots, \mathbf{x}_n$ were drawn independently, then the likelihood can be written as,

$$L(\Theta) = \prod_{i=1}^{n} p(\mathbf{x}_i|\Theta)$$

We can then consider the log likelihood function:

$$\ell(\Theta) = \ln L(\Theta) = \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\Theta)$$

The Maximum Likelihood Estimation procedure can then be equivalently described as:

Estimate $\hat{\Theta}$ that maximizes $\ell(\Theta)$

How can we estimate the MLE of Θ?

$$\nabla_{\Theta}\, \ell(\Theta) = 0$$

$$\implies \sum_{i=1}^{n} \nabla_{\Theta}\, \ln p(x_i|\Theta) = 0$$

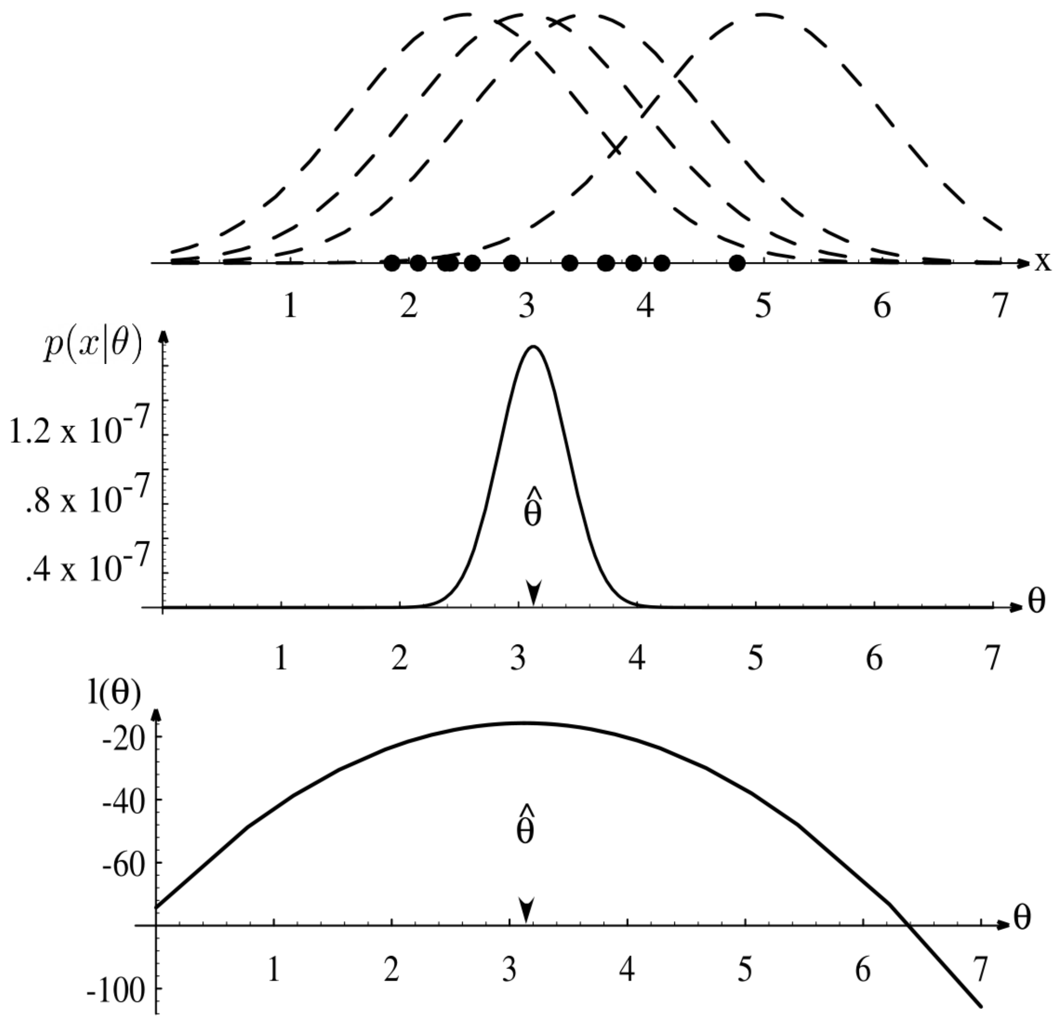


*Image Source: Duda Hart Stork - Pattern Classification

Recap of the previous class: Maximum Likelihood Estimation

If we assume $\mathbf{x}_1, \ldots, \mathbf{x}_n$ were drawn independently, then the likelihood can be written as,

$$L(\Theta) = \prod_{i=1}^{n} p(\mathbf{x}_i|\Theta)$$

We can then consider the log likelihood function:

$$\ell(\Theta) = \ln L(\Theta) = \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\Theta)$$

The Maximum Likelihood Estimation procedure can then be equivalently described as:

Estimate $\hat{\Theta}$ that maximizes $\ell(\Theta)$

How can we estimate the MLE of Θ ?

$$\nabla_{\Theta}\, \ell(\Theta) = 0$$

$$\implies \sum_{i=1}^{n} \nabla_{\Theta} \ln p(x_i|\Theta) = 0$$

Related: Maximum A Posteriori Estimation (MAP)

Estimate $\hat{\Theta}$ that maximizes $L(\Theta)P(\Theta)$

MAP with equal priors is equivalent to MLE

Recap of the previous class: MLE of Gaussian distribution parameters

Let us assume the class-conditional densities follow Gaussian distributions $p(x|w_j) \sim N(\mu_j, \Sigma_j)$. We wish to estimate the parameters of the distribution $\theta_j = \{\mu_j, \Sigma_j\}$ that completely describe the distribution. The Gaussian density function is,

$$p(\mathbf{x}_i|\theta_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{-1}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j)\right\}$$

The log of the Gaussian density is,

$$\ln p(\mathbf{x}_i|\theta_j) = -\ln\left\{(2\pi)^{d/2}|\Sigma_j|^{-1}\right\} - \frac{1}{2}(\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j)$$

Solving for the MLE of μ_j :

$$\nabla_{\mu_j} \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\theta_j) = \sum_{i=1}^{n} \Sigma_j^{-1}(\mathbf{x}_i - \mu_j) = 0$$

$$\implies \hat{\mu_j} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

Recap of the previous class: MLE of Gaussian distribution parameters

Let us assume the class-conditional densities follow Gaussian distributions $p(x|w_j) \sim N(\mu_j, \Sigma_j)$. We wish to estimate the parameters of the distribution $\theta_j = \{\mu_j, \Sigma_j\}$ that completely describe the distribution. The Gaussian density function is,

$$p(\mathbf{x}_i|\theta_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{-1}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j)\right\}$$

The log of the Gaussian density is,

$$\ln p(\mathbf{x}_i|\theta_j) = -\ln\left\{(2\pi)^{d/2}|\Sigma_j|^{-1}\right\} - \frac{1}{2}(\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j)$$

Solving for the MLE of μ_j :

$$\nabla_{\mu_j} \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\theta_j) = \sum_{i=1}^{n} \Sigma_j^{-1} (\mathbf{x}_i - \mu_j) = 0$$

$$\implies \hat{\mu_j} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

Similarly, if all variances are equal and all covariances are zero, we can solve for the MLE of σ_j:

$$\hat{\sigma_j}^2 = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \mu_j)^2$$

What is the MLE of Σ_j ?

Recap of the previous class: Naïve Bayes Classifier

Conditional Independence: X is conditionally independent of Y given Z if and only if the probability distribution of X is independent of Y given Z, i.e.,

$$P(X|Y,Z) = P(X|Z)$$

Naïve Bayes Classifier

- Generative Classification Model: Estimates class-conditional densities $p(x|w_j)$ and prior probabilities $P(w_j)$.
- Assumes each features is *conditionally independent* of others, given the class, i.e.,

$$p(x_i|x_k, w_j) = p(x_i|w_j) \ \forall i, j, k$$

- The consequence of the conditional independence assumption is that the class-conditional densities $p(\mathbf{x}|w_j)$ can be estimated in terms of the class-conditional density of all the features $p(x_i|w_j)$.

$$p(\mathbf{x}|w_j) = p(x_1, \ldots, x_d|w_j) = p(x_1, \ldots, x_{d-1}|x_d, w_j)p(x_d|w_j)$$

Recap of the previous class: Naïve Bayes Classifier

Naïve Bayes Classifier

- Generative Classification Model: Estimates class-conditional densities $p(x|w_j)$ and prior probabilities $P(w_j)$.
- Assumes each features is *conditionally independent* of others, given the class, i.e.,

$$p(x_i|x_k, w_j) = p(x_i|w_j)\, \forall i, j, k$$

- The consequence of the conditional independence assumption is that the class-conditional densities $p(\mathbf{x}|w_j)$ can be estimated in terms of the class-conditional density of all the features $p(x_i|w_j)$.

$$\begin{aligned} p(\mathbf{x}|w_j) &= p(x_1, ..., x_d|w_j) = p(x_1, ..., x_{d-1}|x_d, w_j)p(x_d|w_j) \\ &= ... \\ &= p(x_1|w_j)...p(x_d|w_j) = \prod_{i=1}^{d} p(x_i|w_j) \end{aligned}$$

- If $p(\mathbf{x}|w_j) \sim N(\mu_j, \Sigma_j)$, we would have to estimate $\mu_j \in \mathbb{R}^d, \Sigma_j \in \mathbb{R}^{d\times d}$, for a total of $d + d^2$ parameters for each class. By assuming the features are conditionally independent given the class, we only need to estimate $2d$ parameters $(\mu_{ij} \in \mathbb{R}, \sigma_{ij} \in \mathbb{R},\ i = 1, ..., d)$ for each class instead.

Naïve Bayes Classifier

Decide class w_j if,

$$\prod_{i=1}^{n} p(x_i|w_j)P(w_j) > \prod_{i=1}^{n} p(x_i|w_k)P(w_k)\, \forall k \neq j$$

- If $p(x_i|w_j) \sim N(\mu_j, \sigma_j)$, the MLE estimates of μ_j, σ_j are:

Naïve Bayes Classifier

Decide class w_j if,

$$\prod_{i=1}^{n} p(x_i|w_j)P(w_j) > \prod_{i=1}^{n} p(x_i|w_k)P(w_k) \,\forall k \neq j$$

- If $p(x_i|w_j) \sim N(\mu_j, \sigma_j)$, the MLE estimates of μ_j, σ_j are:

$$\hat{\mu}_j = \frac{1}{\sum_{i=1}^{n} \delta_{w_j}(y_i)} \sum_{i=1}^{n} x_i \delta_{w_j}(y_i)$$

$$\hat{\sigma}_j^2 = \frac{1}{\sum_{i=1}^{n} \delta_{w_j}(y_i)} \sum_{i=1}^{n} (x_i - \mu_j)^2 \delta_{w_j}(y_i)$$

where,

$$\delta_{w_j}(y_i) = \begin{cases} 1 & , y_i = w_j \\ 0 & , \text{o/w} \end{cases}$$

- The estimated prior probabilities are:

Naïve Bayes Classifier

Decide class w_j if,

$$\prod_{i=1}^{n} p(x_i|w_j)P(w_j) > \prod_{i=1}^{n} p(x_i|w_k)P(w_k)\,\forall k \neq j$$

- If $p(x_i|w_j) \sim N(\mu_j, \sigma_j)$, the MLE estimates of μ_j, σ_j are:

$$\hat{\mu}_j = \frac{1}{\sum_{i=1}^{n} \delta_{w_j}(y_i)} \sum_{i=1}^{n} x_i \delta_{w_j}(y_i)$$

$$\hat{\sigma}_j^2 = \frac{1}{\sum_{i=1}^{n} \delta_{w_j}(y_i)} \sum_{i=1}^{n} (x_i - \mu_j)^2 \delta_{w_j}(y_i)$$

where,

$$\delta_{w_j}(y_i) = \begin{cases} 1 & , y_i = w_j \\ 0 & , \text{o/w} \end{cases}$$

- The estimated prior probabilities are:

$$P(w_j) = \frac{\sum_{i=1}^{n} \delta_{w_j}(y_i)}{n}$$

Naïve Bayes Classifier

Decide class w_j if,

$$\prod_{i=1}^{n} p(x_i|w_j)P(w_j) > \prod_{i=1}^{n} p(x_i|w_k)P(w_k)\,\forall k \neq j$$

- If all x_i are discrete valued:

- The estimated prior probabilities are:

Naïve Bayes Classifier

Decide class w_j if,

$$\prod_{i=1}^{n} p(x_i|w_j)P(w_j) > \prod_{i=1}^{n} p(x_i|w_k)P(w_k)\ \forall k \neq j$$

- If all x_i are discrete valued:

$$\hat{p}(x_i = x_{ik}|w_j) = \frac{\#D\{x_i = x_{ik} \wedge y_i = w_j\}}{\#D\{y_i = w_j\}}$$

- The estimated prior probabilities are:

$$\hat{P}(w_j) = \frac{\#D\{y_i = w_j\}}{n}$$

Logistic Regression Classifier

Decide class w_i where

$$P(w_i|x) > P(w_j|x)\ \forall j \neq i$$

Binary Classification:

Logistic Regression assumes the following parametric model to estimate the posterior probabilities of the two classes:

$$P(y=1|\mathbf{x}) = \frac{1}{1+\exp(w_0 + \sum_{t=1}^{d} w_t x_t)}$$

and,

$$P(y=0|\mathbf{x}) = \frac{\exp(w_0 + \sum_{t=1}^{d} w_t x_t)}{1+\exp(w_0 + \sum_{t=1}^{d} w_t x_t)}$$

The assumption of the above parametric model leads to a *linear* classifier, that classifies the data based on a hyperplane between the two classes.

Logistic Regression Classifier

Decide class w_i where

$$P(w_i|x) > P(w_j|x)\ \forall j \neq i$$

Binary Classification:

Logistic Regression assumes the following parametric model to estimate the posterior probabilities of the two classes:

$$P(y=1|\mathbf{x}) = \frac{1}{1+\exp(w_0+\sum_{t=1}^{d} w_t x_t)}$$

and,

$$P(y=0|\mathbf{x}) = \frac{\exp(w_0+\sum_{t=1}^{d} w_t x_t)}{1+\exp(w_0+\sum_{t=1}^{d} w_t x_t)}$$

The assumption of the above parametric model leads to a *linear* classifier, that classifies the data based on a hyperplane between the two classes.

Classification rule to assign class $y=0$:

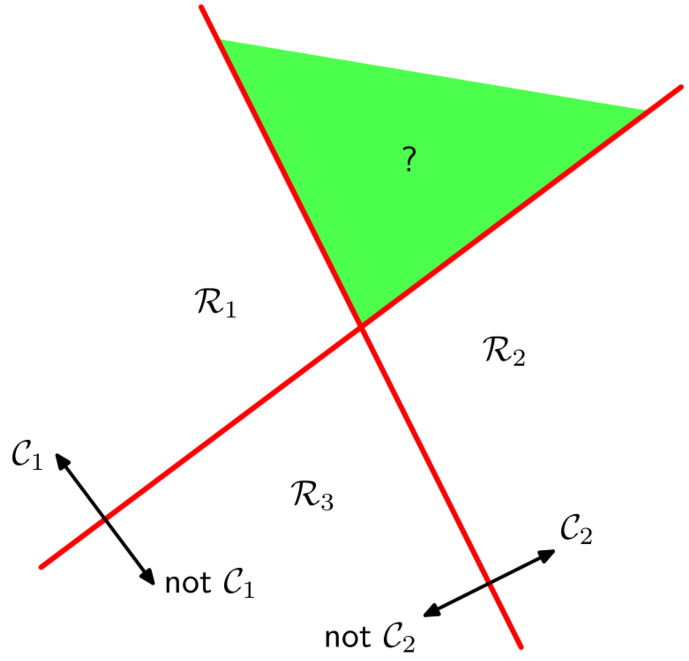
$$\frac{P(y=0|\mathbf{x})}{P(y=1|\mathbf{x})} > 1 \implies \exp(w_0+\sum_{t=1}^{d} w_t x_t) > 1 \implies w_0+\sum_{t=1}^{d} w_t x_t > 0$$

Multi-class Classification

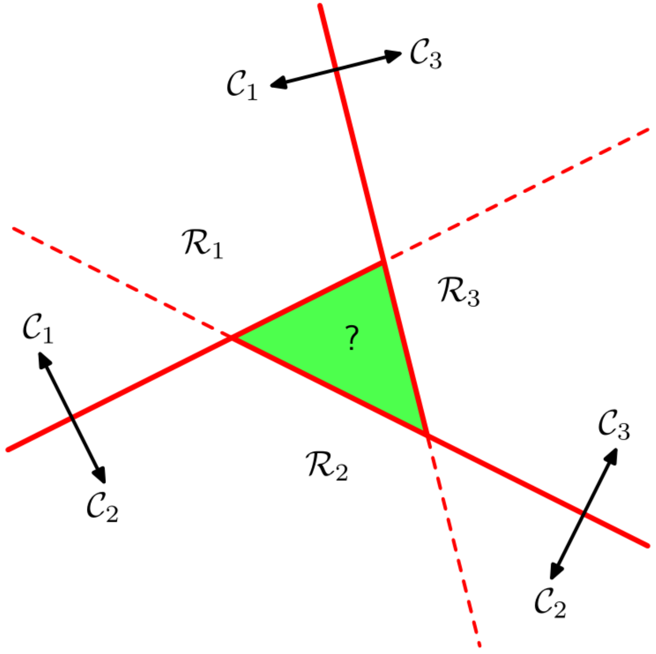
Given $c > 2$ number of classes, we can consider building:

- $(c - 1)$ number of binary classifiers (one-vs-all classification)
- $c(c - 1)$ number of binary classifiers (one-vs-one classification)

Both approaches have the drawback of leading to ambiguous regions that become difficult to classify.



one-vs-all



one-vs-one

*Image Source:
Bishop - Pattern Recognition and Machine Learning

Multi-class Classification

Given $c > 2$ number of classes, we consider building a single c-class discriminating classifier that is comprised of c linear functions of the form

$$y_j = \mathbf{w}_j^T \mathbf{x} + w_{j0}\,,\ j = 1, \dots, c$$

Data $\mathbf{x}$ is assigned to class j if $y_j > y_k \, \forall k \neq j$.

The decision boundary between class j and class k is given by $y_j - y_k = 0$, and the equation of this (d-1) dimensional hyperplane is,

$$(\mathbf{w_j} - \mathbf{w_k})^T \mathbf{x} + (w_{j0} - w_{k0}) = 0$$

Each decision region is always a single connected and convex region.

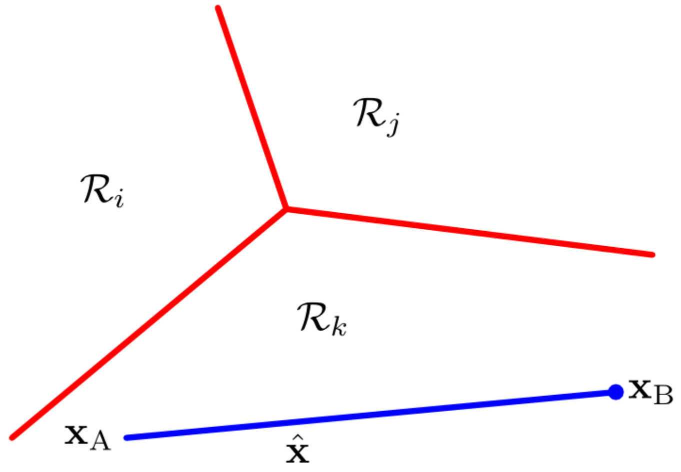


*Image Source:
Bishop - Pattern Recognition and Machine Learning

Multi-class Classification

Given $c > 2$ number of classes, we consider building a single c-class discriminating classifier that is comprised of c linear functions of the form

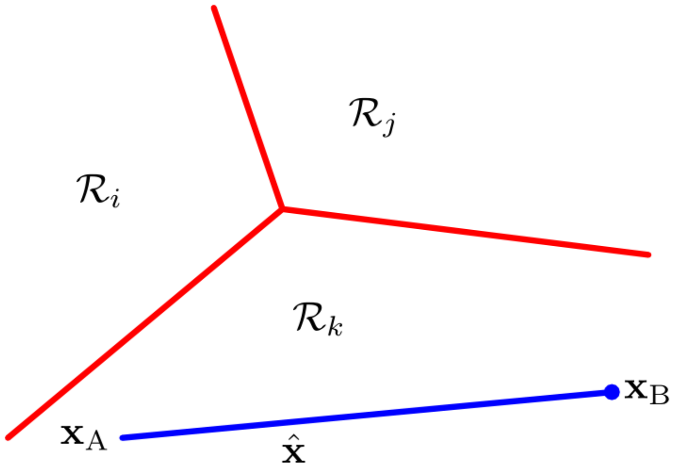
$$y_j = \mathbf{w}_j^T \mathbf{x} + w_{j0} \, , \; j = 1, ..., c$$

Data $\mathbf{x}$ is assigned to class j if $y_j > y_k \, \forall k \neq j$.

The decision boundary between class j and class k is given by $y_j - y_k = 0$, and the equation of this (d-1) dimensional hyperplane is,

$$(\mathbf{w_j} - \mathbf{w_k})^T \mathbf{x} + (w_{j0} - w_{k0}) = 0$$

Each decision region is always a single connected and convex region.



Proof: Let $\mathbf{x}_A, \mathbf{x}_B \in \mathcal{R}_k$, and any $\mathbf{x}$ lying on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ can be expressed as,

$$\mathbf{x} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B, 0 \leq \lambda \leq 1$$

$\mathbf{x}$ is classified to class $y_j(\mathbf{x})$, which can be written as,

$$y_j(\mathbf{x}) = \lambda y_j(\mathbf{x}_A) + (1 - \lambda) y_j(\mathbf{x}_B)$$

As $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A) \forall j \neq k,$ and $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B) \forall j \neq k,$ therefore, $y_k(\mathbf{x}) > y_j(\mathbf{x}) \forall j \neq k$.

*Image Source:
Bishop - Pattern Recognition and Machine Learning