Linear Models for Classification

Chia-Ping Chen

Professor
National Sun Yat-sen University
Department of Computer Science and Engineering

Machine Learning

Topics

- Discriminant Functions
- Generative Models
- Discriminative Models
- Laplace Approximation
- Bayesian Logistic Regression

Definition. The goal of **classification** is to take an input instance x and assign it to one discrete class. The assignment is achieved by a **classification function**.

- In **AlphaGo**, decide the next move given previous moves
- Given a **speech** waveform, decide its emotional category
- Given an **image** of written digit, decide the digit
- Decide whether to buy, sell, or keep for a **stock portfolio**

Decision Regions

Definition. By a classification function, an **input space** or **feature space** is divided into **decision regions** for the classes. The boundary between adjacent decision regions is a **decision boundary** or **decision surface**.

Definition. A **linear model for classification** means that the decision boundaries are linear functions of the input vector x. That is, the decision boundaries are hyperplanes. A **generalized linear model** means that the decision boundaries are linear functions of a feature vector ϕ.

Linearly Separable

Definition. A data set whose data points of different classes can be separated cleanly by linear decision boundaries is **linearly separable**.

- **Discriminants.** Classification via $y_k(\boldsymbol{x})$.

$$\hat{k} = \arg\max_k \ y_k(\boldsymbol{x})$$

- **Generative models.** Classification via $p(\mathcal{C}_k)$ and $p(\boldsymbol{x}|\mathcal{C}_k)$.

$$\hat{k} = \arg\max_k \ p(\mathcal{C}_k|\boldsymbol{x}) = \arg\max_k \ p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

- **Discriminative models.** Classification via $p(\mathcal{C}_k|\boldsymbol{x})$.

$$\hat{k} = \arg\max_k \ p(\mathcal{C}_k|\boldsymbol{x})$$

Labels for Binary Classification

In binary classification, we may use the 0/1 target values for the class labels, i.e. $t = 1$ for a data point of class $\mathcal{C}_1$ and $t = 0$ for a data point of class $\mathcal{C}_2$.

This labeling scheme allows the interpretation that t is the probability of class $\mathcal{C}_1$.

Labels for K-ary Classification

In K-ary classification, we often use K **1-hot vectors** for the class labels.

- The 1-hot vector $\boldsymbol{t}^{(k)}$ has components $t_j^{(k)} = \delta_{kj}$.
- We use $\boldsymbol{t} = \boldsymbol{t}^{(k)}$ for a data point of class $\mathcal{C}_k$.

This labeling scheme allows the interpretation that component t_j is the probability of class $\mathcal{C}_j$.

Definition. In binary classification, we may make decision based on the posterior probability of class $\mathcal{C}_1$. This is achieved by an **activation function** $f(\cdot)$ with range $(0, 1)$ such that

$$p(\mathcal{C}_1|\boldsymbol{x}) \approx f(a(\boldsymbol{x}))$$

Note that the activation function act on an **activation**, which is denoted by $a(\boldsymbol{x})$, of $\boldsymbol{x}$.

Definition. A **linear activation** is linear in the input variables

$$a(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

With linear activation, the decision boundary is determined by

$$a(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = f^{-1}\left(\frac{1}{2}\right)$$

Note that this is a hyperplane, and we have a linear model.

From Input Space to Feature Space

Consideration in the input space can be extended to a feature space defined by a set of basis functions

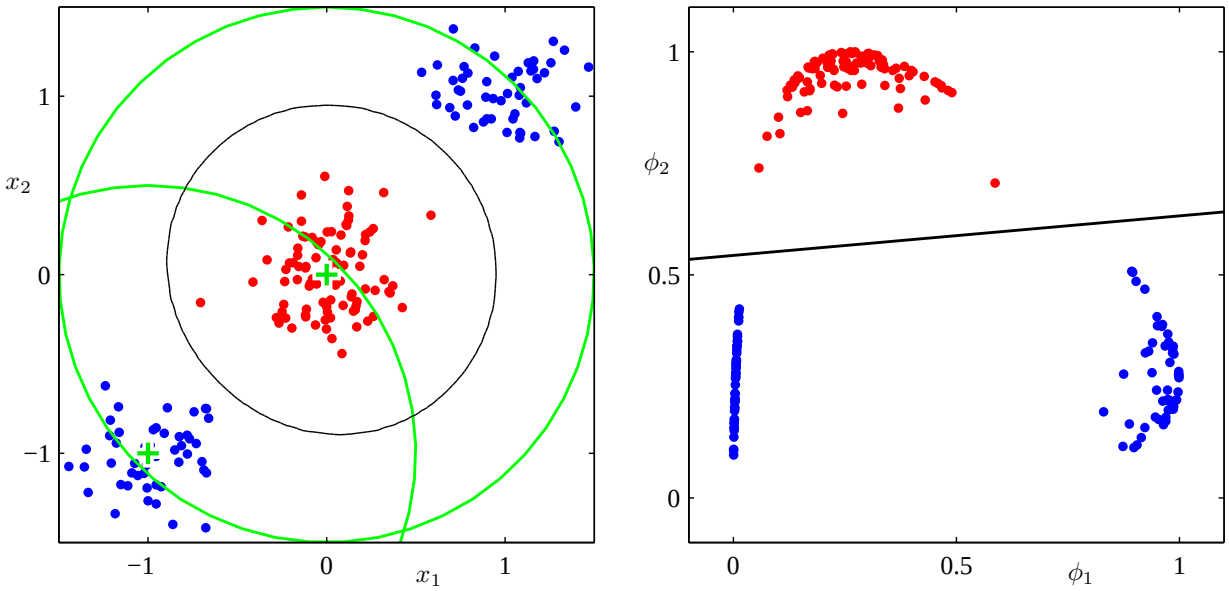
$$\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{x})$$

It is often assumed that **activation is linear in the features**. Then we have

$$p(\mathcal{C}_1|\boldsymbol{x}) \approx f(a(\boldsymbol{x})) = f(\boldsymbol{w}^T\boldsymbol{\phi} + w_0)$$

Note that the decision boundary is a hyperplane in a feature space, and non-flat in the input space.

Input Space and Feature Space



The basis functions $\phi_1(\boldsymbol{x})$ and $\phi_2(\boldsymbol{x})$ are Gaussian basis functions. A linear decision boundary in the feature space (right) corresponds to a non-linear decision boundary in the input space (left).

K-ary Classification and Activations

In K-ary classification, we may make decision based on the posterior probabilities of K classes. This is achieved by an activation function $\boldsymbol{y}(\boldsymbol{x})$ such that the components of $\boldsymbol{y}$ are non-negative and sum to 1, so

$$(p(\mathcal{C}_1|\boldsymbol{x}), \ldots, p(\mathcal{C}_K|\boldsymbol{x}))^T \approx \boldsymbol{f}(\boldsymbol{a}(\boldsymbol{x}))$$

Again, the activation function $\boldsymbol{f}(\cdot)$ acts on class activations

$$\boldsymbol{a}(\boldsymbol{x}) = (a_1(\boldsymbol{x}), \ldots, a_K(\boldsymbol{x}))^T$$

And again, if the class activations are linear functions of the input vector (or feature vector), then we have a (or generalized) linear model.

Discriminants

Definition. A **discriminant** is a function that takes an input vector (or feature vector) and assign it to a class. A discriminant is **linear** if the corresponding decision boundaries are hyperplanes.

Simplest Linear Discriminant

The simplest linear discriminant is a linear function of the input vector

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

Here $\boldsymbol{w}$ is the **weight vector**, and w_0 is the **bias**. The negative bias is the **threshold**.

Sometimes we write

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}$$

where $\tilde{\boldsymbol{w}}^T = (w_0, \boldsymbol{w}^T)$ and $\tilde{\boldsymbol{x}}^T = (1, \boldsymbol{x}^T)$.

Definition. In a **binary discriminant**, there is a discriminant function $y(\mathbf{x})$, and an input vector $\mathbf{x}$ is assigned to class $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$ and class $\mathcal{C}_2$ otherwise. A **binary linear discriminant** has a linear discriminant function.

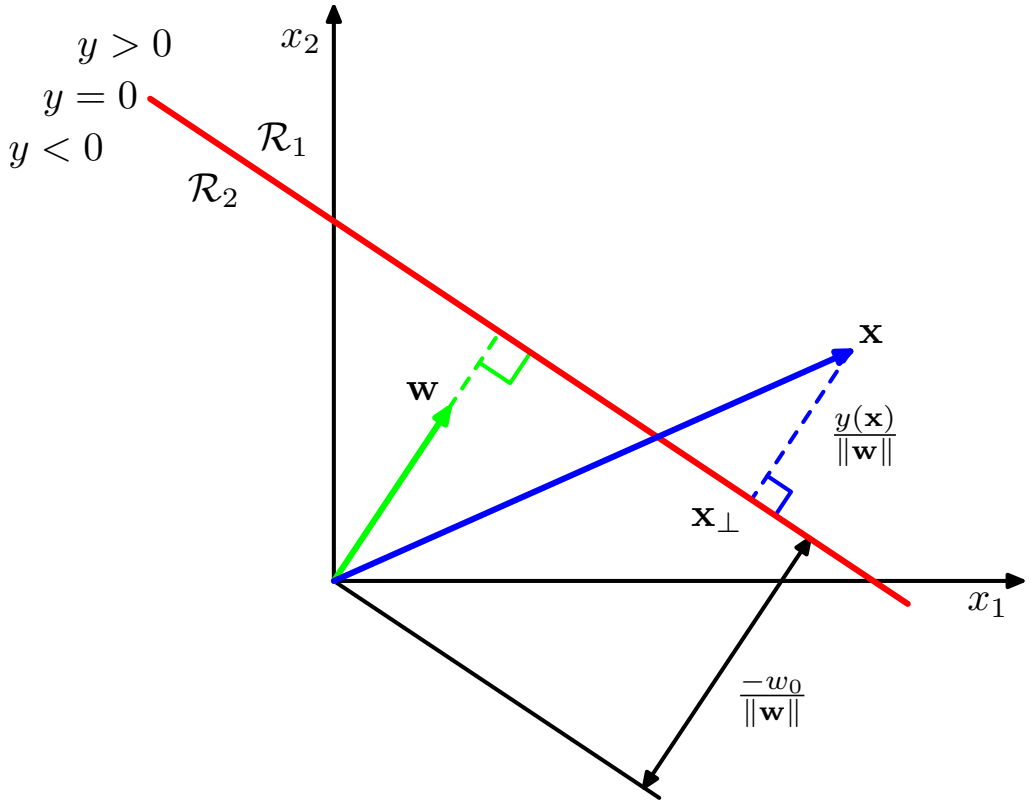
The decision boundary of a binary discriminant is given by

$$y(\boldsymbol{x}) = 0$$

The decision boundary is a hyperplane for a binary linear discriminant.

Properties of a Binary Linear Discriminant

- $\boldsymbol{w}$ is orthogonal to the decision boundary.
- The distance from $\boldsymbol{x}$ to the decision boundary is $\frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$.



K-ary Discriminant

Definition. In a K-**ary discriminant**, there are K discriminant functions $y_1(\mathbf{x}), \ldots, y_K(\mathbf{x})$, and an input vector $\mathbf{x}$ is assigned to class $\mathcal{C}_k$ if

$$y_k(\mathbf{x}) > y_j(\mathbf{x}) \text{ for all } j \neq k$$

A K-**ary linear discriminant** has K linear discriminant functions of the input variables

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}, \quad k = 1, \ldots, K$$

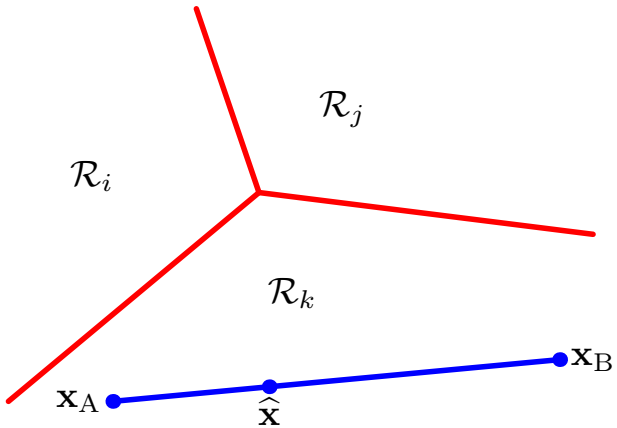
Decision Boundaries and Decision Regions

With a K-ary linear discriminant, the decision boundaries are hyperplanes and the decision regions are convex.

The decision boundary between class $\mathcal{C}_k$ and class $\mathcal{C}_j$ is given by $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$, i.e.

$$(\boldsymbol{w}_k - \boldsymbol{w}_j)^T \boldsymbol{x} + (w_{k0} - w_{j0}) = 0$$

which is a hyperplane. Convex decision regions are illustrated below.



The parameters in a linear discriminant (binary or K-ary) can be learned from data.

- Let $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{t}_n)\}_{n=1}^N$ be a data set
- Let $\tilde{\boldsymbol{X}}$ be the input matrix whose nth row is

$$\tilde{\boldsymbol{x}}_n^T = (1, x_{n1}, \ldots, x_{nD})$$

- Let $\boldsymbol{T}$ be the target matrix whose nth row is

$$\boldsymbol{t}_n^T = (t_{n1}, \ldots, t_{nK})$$

- Let $\tilde{\boldsymbol{W}}$ be the weight matrix whose kth column is

$$\tilde{\boldsymbol{w}}_k = \begin{pmatrix} w_{k0} \\ w_{k1} \\ \vdots \\ w_{kD} \end{pmatrix}$$

Minimizing the Sum of Squared Errors

For a K-ary linear discriminant, define the output $\boldsymbol{y}(\boldsymbol{x}_n)^T$ of $\boldsymbol{x}_n$ as the nth row of $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{W}}$. The sum of squared errors between the outputs and the targets of $\mathcal{D}$ is

$$\begin{aligned} E(\tilde{\boldsymbol{W}}) &= \sum_{n=1}^{N} \|\boldsymbol{y}(\boldsymbol{x}_n) - \boldsymbol{t}_n\|^2 \\ &= \text{tr}\left\{(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{W}} - \boldsymbol{T})(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{W}} - \boldsymbol{T})^T\right\} \end{aligned}$$

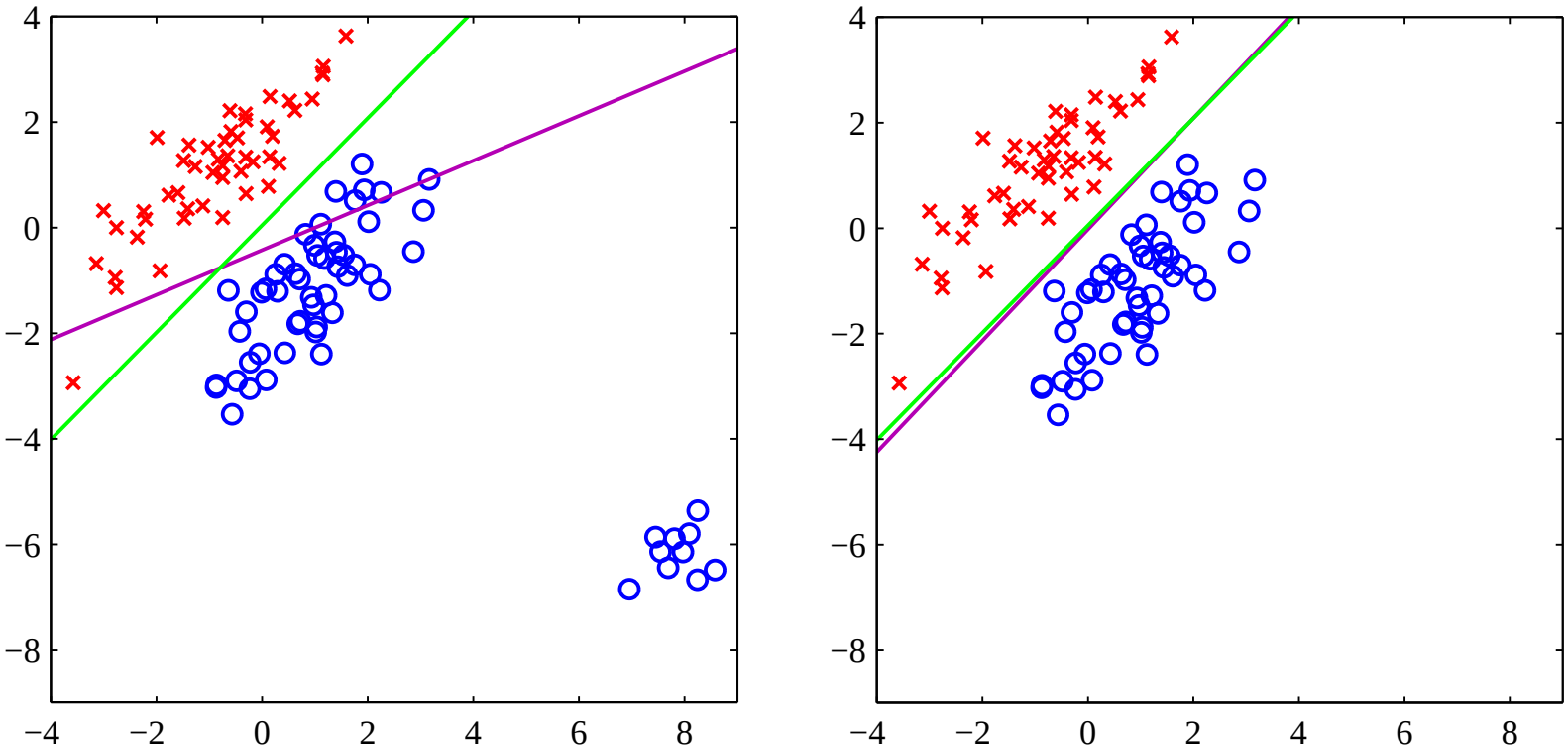
Set the derivative with respect to $\tilde{\boldsymbol{W}}$ to $\boldsymbol{0}$

$$\tilde{\boldsymbol{W}} = (\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^T\boldsymbol{T} = \tilde{\boldsymbol{X}}^\dagger\boldsymbol{T}$$

For a test input vector $\boldsymbol{x}$, the output $\boldsymbol{y}(\boldsymbol{x})$ is

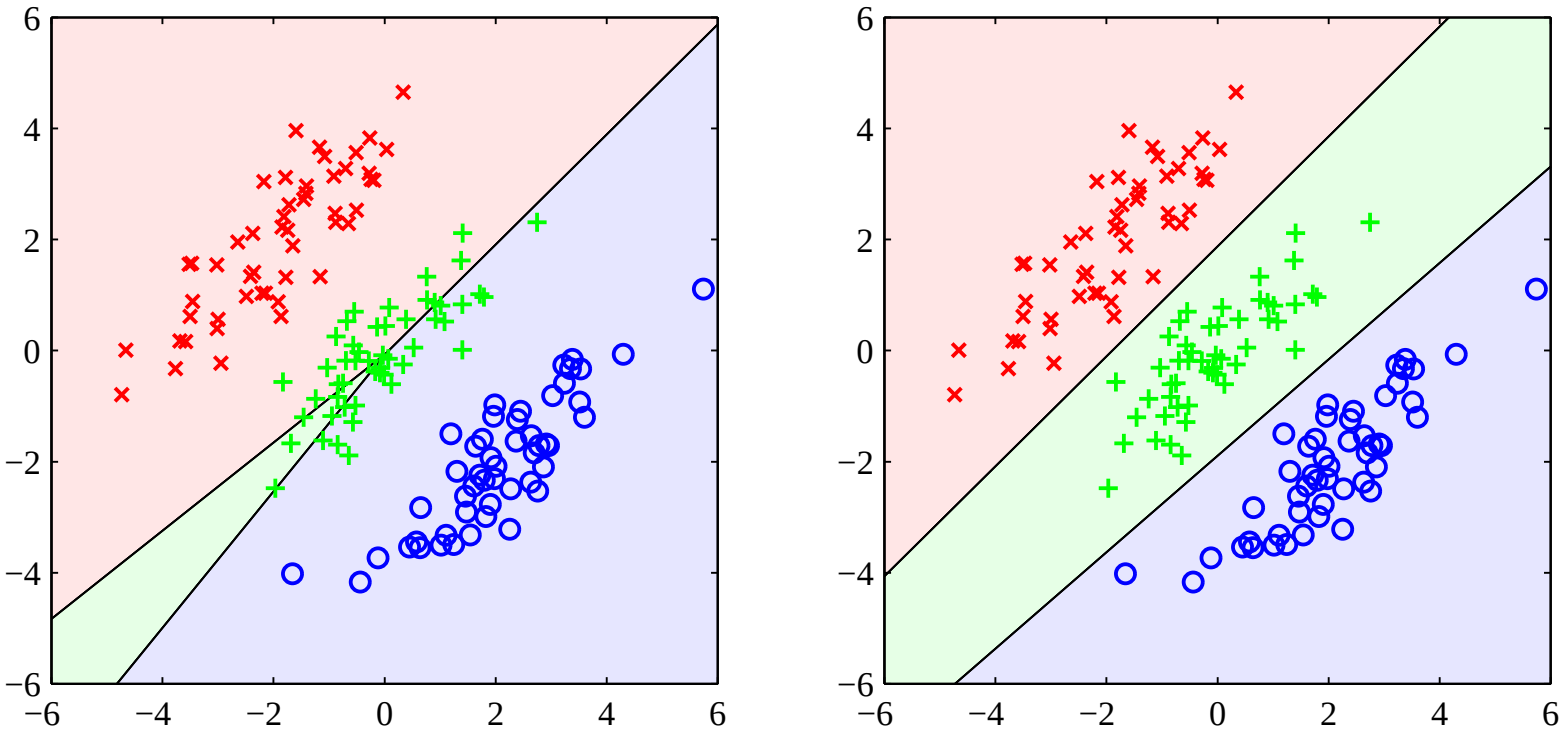
$$\boldsymbol{y}(\boldsymbol{x}) = \tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}} = \boldsymbol{T}^T\left(\tilde{\boldsymbol{X}}^\dagger\right)^T\tilde{\boldsymbol{x}}$$

Issue: Robustness



Left: With outlier data points
Right: Without outlier data

Issue: Correctness



Left: Least squares learning
Right: Logistic regression learning

Projection is often used in dimension reduction. We want to look for an optimum **direction**.

Consider projection to a line defined by a unit vector $\boldsymbol{w}$. Let $m_k\boldsymbol{w}$ is the mean of the projection points of $\mathcal{C}_k$, i.e.

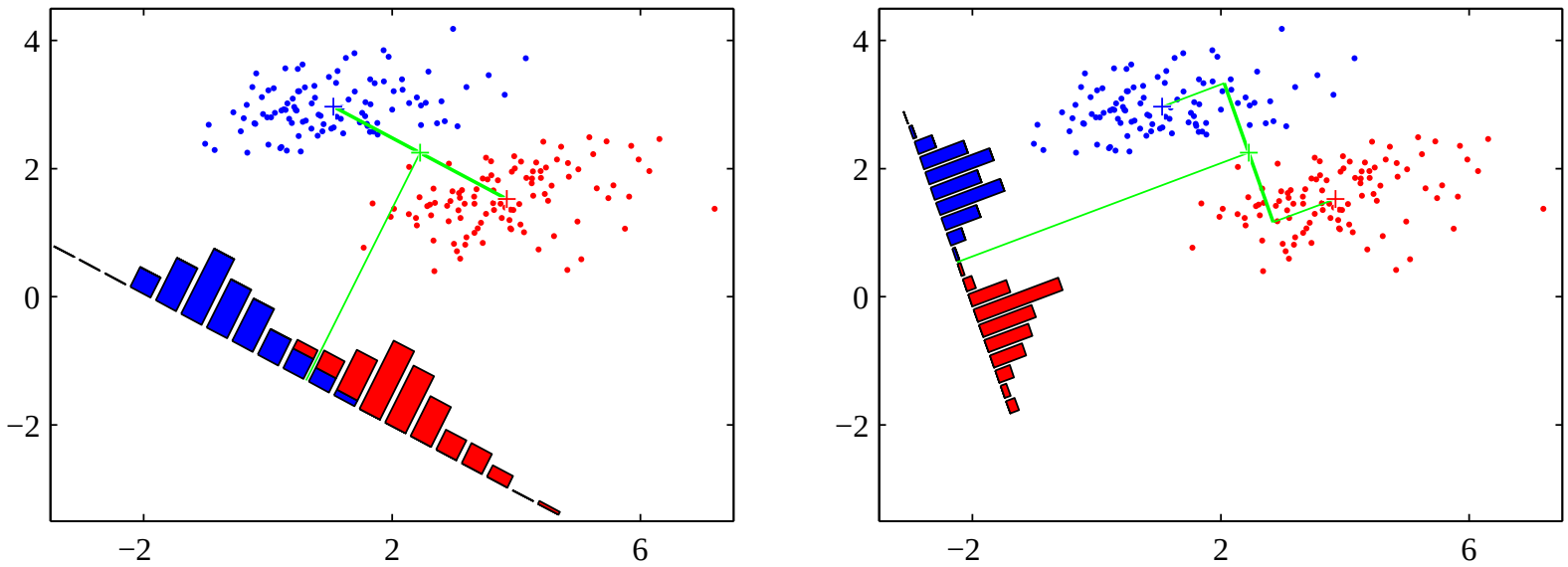
$$m_k\boldsymbol{w} = \frac{1}{n_k}\sum_{\boldsymbol{x}_n\in\mathcal{C}_k}(\boldsymbol{w}^T\boldsymbol{x}_n)\boldsymbol{w} = \boldsymbol{w}^T\left(\frac{1}{n_k}\sum_{\boldsymbol{x}_n\in\mathcal{C}_k}\boldsymbol{x}_n\right)\boldsymbol{w} = (\boldsymbol{w}^T\boldsymbol{m}_k)\boldsymbol{w}$$

where $\boldsymbol{m}_k = \frac{1}{n_k}\sum_{\boldsymbol{x}_n\in\mathcal{C}_k}\boldsymbol{x}_n$ is the mean of data points of class $\mathcal{C}_k$. The separation between classes $\mathcal{C}_1$ and $\mathcal{C}_2$ after the projection can be measured by $(m_2 - m_1)^2$. Since $(m_2 - m_1)^2 = (\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1))^2$, it is maximized when

$$\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

As is shown, there may be considerable overlap between the classes when

$$\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$



Idea. Minimize the class overlap by large separation between class means and small variance within each class.

Let $\boldsymbol{w}$ be a unit vector defining a line and $y_n = \boldsymbol{w}^T \boldsymbol{x}_n$, so $y_n \boldsymbol{w}$ is the projection of $\boldsymbol{x}_n$ on the line. We define the **within-class variance** of class $\mathcal{C}_k$ as

$$s_k^2 = \sum_{\boldsymbol{x}_n \in \mathcal{C}_k} (y_n - m_k)^2$$

and the **total within-class variance** as $s_1^2 + s_2^2$.

Criterion. Maximize

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

The numerator of $J(\boldsymbol{w})$ can be written as

$$(m_2 - m_1)^2 = \boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w} = \boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w}$$

where $\boldsymbol{S}_B$ is the **between-class covariance matrix** defined by

$$\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$$

The denominator can be written as

$$s_1^2 + s_2^2 = \sum_{\boldsymbol{x}_n \in \mathcal{C}_1} (y_n - m_1)^2 + \sum_{\boldsymbol{x}_n \in \mathcal{C}_2} (y_n - m_2)^2 = \boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w}$$

where $\boldsymbol{S}_W$ is the **total within-class covariance matrix** defined by

$$\boldsymbol{S}_W = \sum_{\boldsymbol{x}_n \in \mathcal{C}_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{\boldsymbol{x}_n \in \mathcal{C}_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$$

Thus, the Fisher's criterion can be written as

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$$

Setting the gradient of $J(\boldsymbol{w})$ to $\mathbf{0}$, we get

$$\begin{aligned}(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}) \boldsymbol{S}_W \boldsymbol{w} &= (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) \boldsymbol{S}_B \boldsymbol{w} \\ &\propto \boldsymbol{S}_B \boldsymbol{w} \\ &\propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)\end{aligned}$$

Hence

$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

For K classes, we define the **between-class covariance matrix** as

$$\boldsymbol{S}_B = \sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T$$

the **total within-class covariance matrix** as

$$\boldsymbol{S}_W = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_n \in \mathcal{C}_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T$$

and the **total covariance matrix** as

$$\boldsymbol{S}_T = \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^T$$

Note that $\boldsymbol{S}_B + \boldsymbol{S}_W = \boldsymbol{S}_T$.

For K classes, more than one directions are needed.

Through $\boldsymbol{w}$, a **linear feature** $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ is extracted from $\boldsymbol{x}$. Multiple linear features can be extracted through multiple $\boldsymbol{w}$'s

$$y_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x}, \quad i = 1, \ldots, D'$$

These features form a **feature vector**

$$\boldsymbol{y}(\boldsymbol{x}) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{D'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T\boldsymbol{x} \\ \boldsymbol{w}_2^T\boldsymbol{x} \\ \vdots \\ \boldsymbol{w}_{D'}^T\boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \\ \vdots \\ \boldsymbol{w}_{D'}^T \end{bmatrix} \boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{x}$$

The weight matrix $\boldsymbol{W}$ has the weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{D'}$ as columns.

Define $\boldsymbol{y}_n = \boldsymbol{W}^T \boldsymbol{x}_n$ and

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{\boldsymbol{x}_n \in \mathcal{C}_k} \boldsymbol{y}_n, \ \boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{y}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k$$

The total within-class covariance matrix in the feature space is

$$\begin{aligned} \boldsymbol{s}_W &= \sum_{k=1}^{K} \sum_{\boldsymbol{x}_n \in \mathcal{C}_k} (\boldsymbol{y}_n - \boldsymbol{\mu}_k)(\boldsymbol{y}_n - \boldsymbol{\mu}_k)^T \\ &= \boldsymbol{W}^T \left(\sum_{k=1}^{K} \sum_{\boldsymbol{x}_n \in \mathcal{C}_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T \right) \boldsymbol{W} = \boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W} \end{aligned}$$

The between-class covariance matrix is

$$\boldsymbol{s}_B = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T = \boldsymbol{W}^T \boldsymbol{S}_B \boldsymbol{W}$$

Fisher's Criterion for Multiple Features

$$
\begin{aligned}
J(\boldsymbol{W}) &= \text{tr}\{\boldsymbol{s}_W^{-1}\boldsymbol{s}_B\} \\
&= \text{tr}\{(\boldsymbol{W}^T\boldsymbol{S}_W\boldsymbol{W})^{-1}(\boldsymbol{W}^T\boldsymbol{S}_B\boldsymbol{W})\}
\end{aligned}
$$

- $J(\boldsymbol{W})$ is large when the between-class covariance is large and when the within-class covariance is small.
- $\boldsymbol{W}$ is determined by the eigenvectors of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ with the D' largest eigenvalues.

Perceptron

Class Labels in Perceptron

In perceptron, we use class label $t \in \{-1, 1\}$, where $t = 1$ represents $\mathcal{C}_1$ and $t = -1$ represents $\mathcal{C}_2$, i.e.

$$t_n = \begin{cases} 1, & \text{if } \boldsymbol{x}_n \in \mathcal{C}_1 \\ -1, & \text{if } \boldsymbol{x}_n \in \mathcal{C}_2 \end{cases}$$

and the decision function of

$$y(\boldsymbol{x}) = \text{sgn}(\boldsymbol{w}^T \boldsymbol{\phi})$$

where $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{x})$ is the feature vector based on fixed basis functions.

The parameters $\boldsymbol{w}$ are learned by a set $\mathcal{D} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^{N}$.

A perceptron makes an error for a data point $(\boldsymbol{x}, t)$ if

$$\boldsymbol{w}^T \boldsymbol{\phi} t < 0$$

The **error measure** of data point $(\boldsymbol{x}_n, t_n)$ can be defined by

$$E_n(\boldsymbol{w}) = \begin{cases} -\boldsymbol{w}^T \boldsymbol{\phi}_n t_n, & \boldsymbol{w}^T \boldsymbol{\phi}_n t_n < 0 \\ 0, & \text{otherwise} \end{cases}$$

For a misclassified example $(\boldsymbol{x}_n, t_n)$, update $\boldsymbol{w}$ by

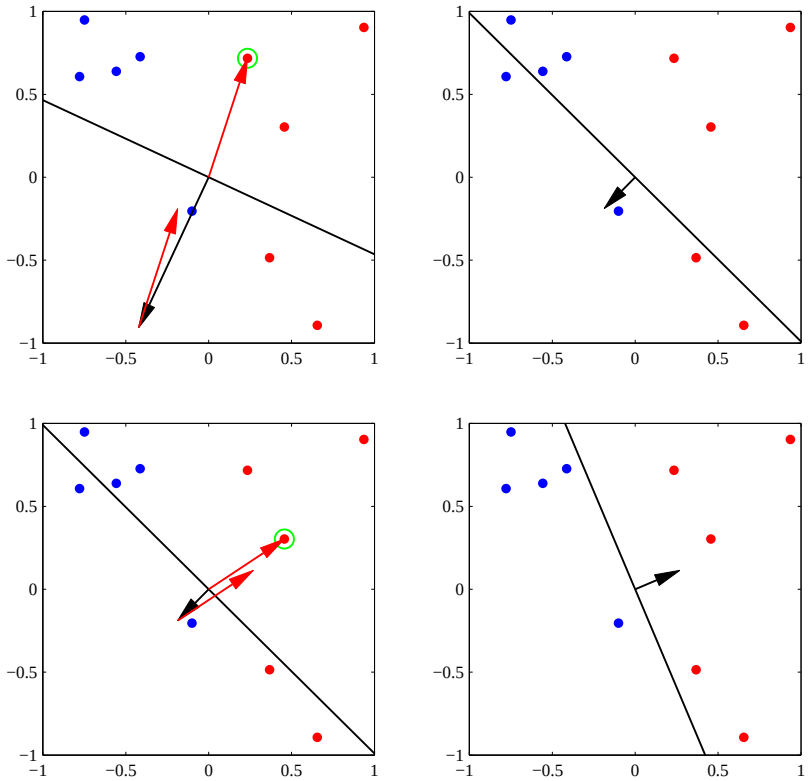
$$\begin{aligned}\boldsymbol{w}^{(\tau+1)} &= \boldsymbol{w}^{(\tau)} - \eta \nabla E_n(\boldsymbol{w}) \\ &= \boldsymbol{w}^{(\tau)} + \eta \boldsymbol{\phi}_n t_n\end{aligned}$$

where $\eta > 0$ is a learning rate.

- For misclassified $\boldsymbol{x}_n$ with $t_n = -1$, subtract $\eta \boldsymbol{\phi}_n$ from $\boldsymbol{w}$.
- For misclassified $\boldsymbol{x}_n$ with $t_n = 1$, add $\eta \boldsymbol{\phi}_n$ from $\boldsymbol{w}$.

Perceptron Convergence Theorem

For linearly separable data, the perceptron algorithm learns a perfect decision boundary in finite steps.



Probabilistic Generative Model

Definition. A **generative model** consists of the **class priors** $p(\mathcal{C}_k)$ and the **class-conditional distributions** $p(\boldsymbol{x}|\mathcal{C}_k)$.

- **Data generation.** A generative model can be used to generate artificial data.
- **Class posteriors.** By Bayes' rule, the posterior of class $\mathcal{C}_k$ is

$$\begin{aligned} p(\mathcal{C}_k|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}, \mathcal{C}_k)}{p(\boldsymbol{x})} \\ &= \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\boldsymbol{x})} \\ &\propto p(\boldsymbol{x}|\mathcal{C}_k)\, p(\mathcal{C}_k) \end{aligned}$$

$p(\mathcal{C}_1|\boldsymbol{x}), \ldots, p(\mathcal{C}_K|\boldsymbol{x})$ are used in making decisions.

In binary classification of classes $\mathcal{C}_1$ and $\mathcal{C}_2$, the posterior of $\mathcal{C}_1$ is

$$\begin{aligned} p(\mathcal{C}_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \frac{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} \\ &= \frac{1}{1 + \exp(-a)} \\ &= \sigma(a) \end{aligned}$$

where $\exp(-a) = \frac{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}$ or $a = \log \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$.

Logistic Sigmoid Function

Definition. The function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

is the **logistic sigmoid function**.

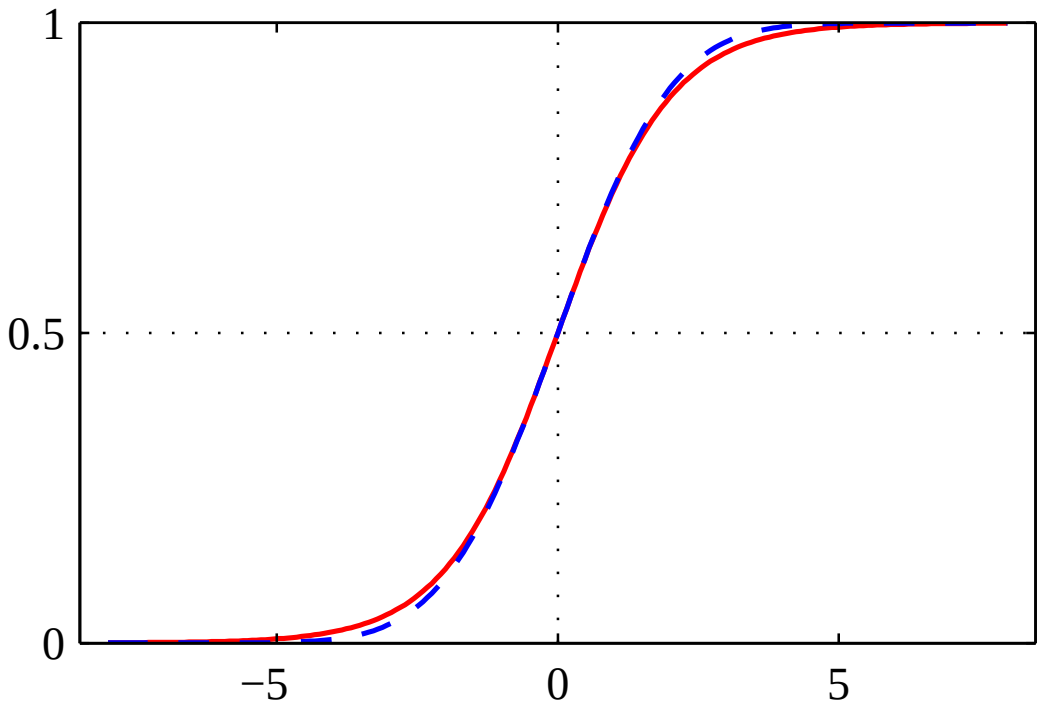
- S-shaped
- squashing: mapping $(-\infty, \infty)$ to $(0, 1)$
- symmetry property

$$\sigma(-a) = 1 - \sigma(a)$$

- a differentiable approximation of the step function with

$$\frac{d\sigma}{da} = \sigma(1 - \sigma)$$

Sigmoid and Inverse Probit Functions



Plot of the logistic sigmoid function $\sigma(a)$ (as shown in red) and a scaled inverse probit function $\Phi(\lambda a)$ for $\lambda = \sqrt{\frac{\pi}{8}}$ (blue)

Definition. The **logit function** is the inverse of the logistic sigmoid function.

$$a(\sigma) = \log\left(\frac{\sigma}{1-\sigma}\right)$$

In binary classification of classes $\mathcal{C}_1$ and $\mathcal{C}_2$, we often have $\sigma = p(\mathcal{C}_1|\boldsymbol{x})$. Then the logit

$$a = \log\left(\frac{\sigma}{1-\sigma}\right) = \log\left(\frac{p(\mathcal{C}_1|\boldsymbol{x})}{1-p(\mathcal{C}_1|\boldsymbol{x})}\right) = \log\left(\frac{p(\mathcal{C}_1|\boldsymbol{x})}{p(\mathcal{C}_2|\boldsymbol{x})}\right)$$

is the **log odds**.

In K-ary classification of classes $\mathcal{C}_1, \ldots, \mathcal{C}_K$, the posterior of $\mathcal{C}_k$ is

$$\begin{aligned} p(\mathcal{C}_k|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}, \mathcal{C}_k)}{p(\boldsymbol{x})} \\ &= \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\boldsymbol{x})} \\ &= \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

where $\exp(a_k) = p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)$ or $a_k = \log\left(p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)\right)$. We call a_k the **activation** of class $\mathcal{C}_k$.

Definition. A common transformation from class actions to class posterior probabilities is the **normalized exponential function** defined by

$$(a_1, \ldots, a_K) \mapsto \frac{(\exp(a_1), \ldots, \exp(a_K))}{\sum_{j=1}^{K} \exp(a_j)}$$

The normalized exponential is also known as the **softmax**: for the class $\mathcal{C}_k$ with the largest a_k, we have

$$\frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)} \approx 1, \quad \frac{\exp(a_{j \neq k})}{\sum_{j=1}^{K} \exp(a_j)} \approx 0$$

if $a_k \gg a_j$ for $j \neq k$.

Priors vs. Class-Conditionals

In a generative model

- The class priors are simply the relative frequencies.
- The key elements are the class-conditional distributions.
- We now introduce a few instances.

Definition. A **Gaussian class-conditional distribution** is

$$\begin{aligned} p(\boldsymbol{x}|\mathcal{C}_k) &= \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)} \end{aligned}$$

where $\boldsymbol{\mu}_k$ is class mean and $\boldsymbol{\Sigma}_k$ is class covariance matrix.

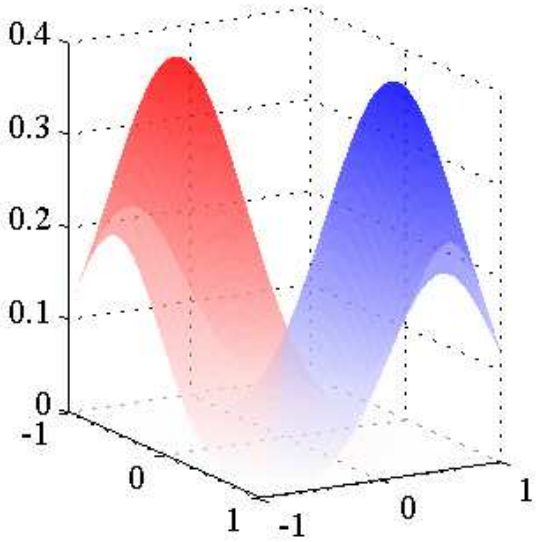
A **Gaussian class-conditional model** assumes a Gaussian class-conditional distribution for each class.

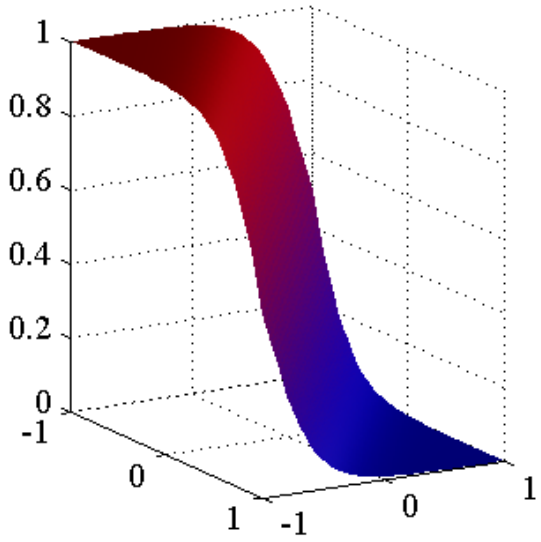
Consider a generative model with Gaussian class-conditional distributions.

- If the Gaussian class-conditional distributions do not share a common covariance matrix, then the decision boundaries are quadratic functions of the input vector.
- If the Gaussian class-conditional distributions share a common covariance matrix, i.e.

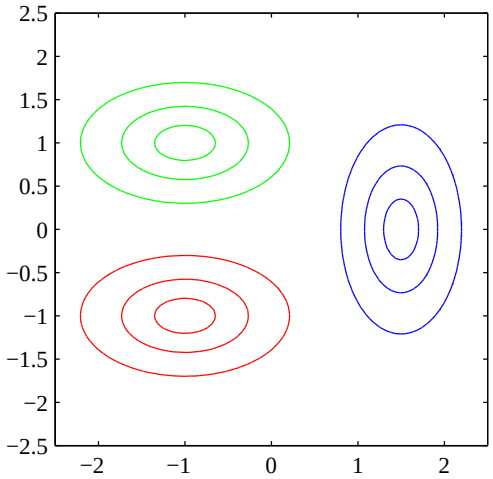
$$\begin{aligned} p(\boldsymbol{x}|\mathcal{C}_k) &= \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}) \\ &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)} \end{aligned}$$

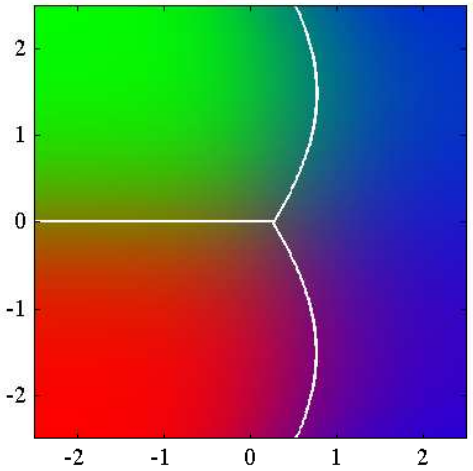
  then the decision boundaries are hyperplanes.





Class-conditionals $p(\boldsymbol{x}|\mathcal{C}_k)$ (left) and posterior $p(\mathcal{C}_1|\boldsymbol{x})$ (right).





Class-conditionals (left) and decision boundaries (right).

Substituting the Gaussian class-conditional distributions and eliminating common factors (the quadratic terms), we get

$$p(\mathcal{C}_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0) = \sigma(a)$$

where

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

The decision boundary is the hyperplane $a = \boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$.

Substituting the Gaussian class-conditional distributions and eliminating common factors (the quadratic terms), we get

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_k^T\boldsymbol{x} + w_{k0})}{\sum_j \exp(\boldsymbol{w}_j^T\boldsymbol{x} + w_{j0})} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where

$$\boldsymbol{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k,\ w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log p(\mathcal{C}_k)$$

The boundary between $\mathcal{R}_j$ and $\mathcal{R}_k$ is decided by $a_j = a_k$, i.e.

$$\boldsymbol{w}_j^T\boldsymbol{x} + w_{j0} = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0}$$

A **naïve Bayes model** is a generative model assuming independent features for the class-conditional distributions.

In particular, for **binary features** $\boldsymbol{x} \in \{0,1\}^D$, the class-conditional distributions are

$$p(\boldsymbol{x}|\mathcal{C}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i}(1-\mu_{ki})^{1-x_i}, \ k = 1, \ldots, K$$

A naïve Bayes model with binary features is a linear model for classification. The activations are linear in $\boldsymbol{x}$, as

$$\begin{aligned} a_k &= \log(p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)) \\ &= \sum_{i=1}^{D}\{x_i \log \mu_{ki} + (1-x_i)\log(1-\mu_{ki})\} + \log p(\mathcal{C}_k) \end{aligned}$$

Discriminative Model

From Generative to Discriminative Models

- In a generative model, we compute the posterior probabilities based on the prior probabilities and conditional distributions.
- In a discriminative model, we model the class posterior probabilities directly.
- In particular, in a **generalized linear discriminative model**, we assume that the class activations are linear functions of the feature vector

$$a_k = \boldsymbol{w}_k^T \boldsymbol{\phi} + w_{k0}$$

- Again, the activations are transformed to the posterior probabilities by a non-linear function, e.g. logistic sigmoid or normalized exponential.

Definition. A **logistic regression** models the posterior probability of class $\mathcal{C}_1$ as the **logistic sigmoid function** acting on a linear function of the feature vector

$$p(\mathcal{C}_1|\boldsymbol{x}) \approx y(\boldsymbol{x}, \boldsymbol{w}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

The posterior probability of class $\mathcal{C}_2$ is

$$p(\mathcal{C}_2|\boldsymbol{x}) = 1 - p(\mathcal{C}_1|\boldsymbol{x}) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

The parameters $\boldsymbol{w}$ in a logistic regression can be learned by a data set $\mathcal{D} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$.

The data likelihood of $(\boldsymbol{x}, t)$ is $p(\mathcal{C}_1|\boldsymbol{x})$ if $t = 1$ and $p(\mathcal{C}_2|\boldsymbol{x})$ if $t = 0$. Since $p(\mathcal{C}_1|\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w})$ and $p(\mathcal{C}_2|\boldsymbol{x}) = 1 - y(\boldsymbol{x}, \boldsymbol{w})$, the data likelihood of $(\boldsymbol{x}, t)$ can be written as

$$p(t|\boldsymbol{x}) = y^t(1-y)^{1-t}$$

where $y = y(\boldsymbol{x}, \boldsymbol{w})$. The likelihood of $\mathcal{D}$ is

$$p(\mathcal{D}|\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

where $y_n = y(\boldsymbol{x}_n, \boldsymbol{w})$.

The negative log likelihood function (error function) is

$$\begin{aligned} E(\boldsymbol{w}) &= -\log p(\mathcal{D}|\boldsymbol{w}) \\ &= -\log\left(\prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}\right) \\ &= -\sum_{n=1}^{N}\{t_n \log y_n + (1-t_n)\log(1-y_n)\} \\ &= \sum_{n=1}^{N} E_n(\boldsymbol{w}) \end{aligned}$$

Note that $E_n(\boldsymbol{w}) = -\{t_n \log y_n + (1-t_n)\log(1-y_n)\}$ is the **cross entropy** between $(t_n, 1-t_n)$ and $(y_n, 1-y_n)$.

Recall that $y(\boldsymbol{x}, \boldsymbol{w}) = \sigma(\boldsymbol{w}^T \boldsymbol{\phi})$. The gradient of $E(\boldsymbol{w})$ is

$$\begin{aligned}
\boldsymbol{\nabla} E(\boldsymbol{w}) &= -\sum_{n=1}^{N} \boldsymbol{\nabla} \left\{ t_n \log y_n + (1 - t_n) \log(1 - y_n) \right\} \\
&= -\sum_{n=1}^{N} \frac{t_n}{y_n} \boldsymbol{\nabla} y_n + \left(\frac{1 - t_n}{1 - y_n} \right) \boldsymbol{\nabla} (1 - y_n) \\
&= -\sum_{n=1}^{N} \frac{t_n}{y_n} y_n (1 - y_n) \boldsymbol{\phi}_n - \left(\frac{1 - t_n}{1 - y_n} \right) y_n (1 - y_n) \boldsymbol{\phi}_n \\
&= \sum_{n=1}^{N} (y_n - t_n) \boldsymbol{\phi}_n \\
&= \boldsymbol{\Phi}^T (\mathbf{y} - \mathbf{t})
\end{aligned}$$

where $\boldsymbol{\Phi}$ is the design matrix.

In **gradient decent**, the parameters are updated by

$$\begin{aligned} \boldsymbol{w}' &= \boldsymbol{w} - \eta \boldsymbol{\nabla} E(\boldsymbol{w}) \\ &= \boldsymbol{w} - \eta \boldsymbol{\Phi}^T (\mathbf{y} - \mathbf{t}) \end{aligned}$$

In **stochastic gradient decent**, $\boldsymbol{w}$ are updated by

$$\begin{aligned} \boldsymbol{w}' &= \boldsymbol{w} - \eta \boldsymbol{\nabla} E_n(\boldsymbol{w}) \\ &= \boldsymbol{w} - \eta (y_n - t_n) \boldsymbol{\phi}_n \end{aligned}$$

where $(t_n, \boldsymbol{x}_n)$ is an example in $\mathcal{D}$.

In **Newton-Raphson method**, $\boldsymbol{w}$ are updated by

$$\boldsymbol{w}' = \boldsymbol{w} - \boldsymbol{H}^{-1}\boldsymbol{\nabla}E(\boldsymbol{w})$$

where $\boldsymbol{H}$ is the Hessian matrix. For the cross entropy error function

$$\boldsymbol{H} = \boldsymbol{\nabla}\boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) = \sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{\Phi}$$

where $\boldsymbol{R} = \text{diag}\{y_1(1 - y_1), \ldots, y_n(1 - y_n)\}$. Thus

$$\begin{aligned}
\boldsymbol{w}' &= \boldsymbol{w} - \boldsymbol{H}^{-1}\boldsymbol{\nabla}E(\boldsymbol{w}) = \boldsymbol{w} - (\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) \\
&= (\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{\Phi})^{-1}\{\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t})\} \\
&= (\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{z}
\end{aligned}$$

where $\boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{R}^{-1}(\mathbf{y} - \mathbf{t})$.

A solution of the **least-squares problem** $\boldsymbol{Ax} = \boldsymbol{b}$ satisfies the normal equation

$$\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b}$$

A solution of the **weighted least-squares problem** $\boldsymbol{CAx} = \boldsymbol{Cb}$, where $\boldsymbol{C}$ is diagonal with positive diagonal elements, satisfies the normal equation

$$\boldsymbol{A}^T\boldsymbol{C}^T\boldsymbol{CAx} = \boldsymbol{A}^T\boldsymbol{C}^T\boldsymbol{Cb}$$

That is

$$\boldsymbol{x} = \left(\boldsymbol{A}^T\boldsymbol{C}^T\boldsymbol{CA}\right)^{-1}\boldsymbol{A}^T\boldsymbol{C}^T\boldsymbol{Cb}$$

Recall that the weight is updated by

$$\boldsymbol{w}' = (\boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{z}$$

This is the solution of a weighted least-squares problem

$$\boldsymbol{C} \boldsymbol{\Phi} \boldsymbol{w} = \boldsymbol{C} \boldsymbol{z}$$

where

$$\boldsymbol{R} = \boldsymbol{C}^T \boldsymbol{C}$$

Since $\boldsymbol{R}$ depends on $\boldsymbol{w}$, the weights are updated in each iteration.

Definition. A **multiclass logistic regression** models the class posterior probabilities as the normalized exponential function acting on linear functions of the feature vector. Specifically, the posterior probability of class $\mathcal{C}_k$

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{\exp(a_k)}{\sum_{j=1}^{K}\exp(a_j)}$$

where the class **activations** are given by

$$a_k = \boldsymbol{w}_k^T\boldsymbol{\phi}$$

The parameters $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ in a multiclass logistic regression can be learned from data.

The likelihood of an example $(\boldsymbol{x}, \boldsymbol{t})$ is

$$p(\boldsymbol{t}|\boldsymbol{x}) = \prod_{k=1}^{K} y_k^{t_k}$$

where

$$y_k = y_k(\boldsymbol{x}) = \frac{\exp(a_k(\boldsymbol{x}))}{\sum_{j=1}^{K} \exp(a_j(\boldsymbol{x}))}$$

The likelihood of a data set $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{t}_n)\}_{n=1}^{N}$ is

$$p(\mathcal{D}|\boldsymbol{w}) = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

Maximizing the data likelihood is equivalent to minimizing the negative log data likelihood, so we define

$$\begin{aligned} E(\boldsymbol{w}) &= -\log \prod_{n=1}^{N} p(\boldsymbol{t}_n|\boldsymbol{x}_n) \\ &= -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log y_{nk} \\ &= \sum_{n=1}^{N} E_n(\boldsymbol{w}) \end{aligned}$$

Note that

$$E_n = -\sum_{k=1}^{K} t_{nk} \log y_{nk}$$

is the cross entropy between $\boldsymbol{t}_n$ and $\boldsymbol{y}_n$.

From $a_j = \boldsymbol{w}_j^T \boldsymbol{\phi}$ and $y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$, we have

$$\boldsymbol{\nabla}_{\boldsymbol{w}_j} a_l = \delta_{jl} \boldsymbol{\phi} \quad \text{and} \quad \frac{\partial y_k}{\partial a_j} = y_k(\delta_{kj} - y_j)$$

It follows that

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{w}_j} y_k &= \sum_l \frac{\partial y_k}{\partial a_l} \left(\boldsymbol{\nabla}_{\boldsymbol{w}_j} a_l \right) \\ &= \sum_l y_k(\delta_{kl} - y_l)\delta_{jl} \boldsymbol{\phi} \\ &= y_k(\delta_{kj} - y_j) \boldsymbol{\phi} \end{aligned}$$

Consider the first-order derivatives of the error $E = -\sum_{k=1}^{K} t_k \log y_k$ of a data point $(\boldsymbol{t}, \boldsymbol{x})$.

$$\begin{aligned}
\nabla_{\boldsymbol{w}_j} \left(-\sum_{k=1}^{K} t_k \log y_k \right) &= -\sum_{k=1}^{K} t_k \left(\frac{1}{y_k} \nabla_{\boldsymbol{w}_j} y_k \right) \\
&= -\sum_{k=1}^{K} t_k \left((\delta_{kj} - y_j) \boldsymbol{\phi} \right) \\
&= -t_j \boldsymbol{\phi} + \sum_{k=1}^{K} t_k y_j \boldsymbol{\phi} \\
&= -t_j \boldsymbol{\phi} + y_j \boldsymbol{\phi} \\
&= (y_j - t_j) \boldsymbol{\phi}
\end{aligned}$$

Thus, the gradient of $E(\boldsymbol{w})$ of data set $\mathcal{D}$ is

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}) = \nabla_{\boldsymbol{w}_j} \left(\sum_{n=1}^{N} E_n(\boldsymbol{w}) \right) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \boldsymbol{\phi}_n = \boldsymbol{\Phi}^T (\mathbf{y}_j - \mathbf{t}_j)$$

Consider the second-order derivatives of the error $E = -\sum_{k=1}^{K} t_k \log y_k$ of a data point $(\boldsymbol{t}, \boldsymbol{x})$.

$$\begin{aligned}\nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} \left(-\sum_{k=1}^{K} t_k \log y_k \right) &= \nabla_{\boldsymbol{w}_k} (y_j - t_j) \boldsymbol{\phi} \\ &= y_k (\delta_{kj} - y_j) \boldsymbol{\phi} \boldsymbol{\phi}^T\end{aligned}$$

Thus, the Hessian of $E(\boldsymbol{w})$ of data set $\mathcal{D}$ is

$$\begin{aligned}\nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}) &= \nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} \left(\sum_{n=1}^{N} E_n(\boldsymbol{w}) \right) = \sum_{n=1}^{N} y_{nk} (\delta_{kj} - y_{nj}) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T \\ &= \boldsymbol{\Phi}^T \boldsymbol{R}' \boldsymbol{\Phi}\end{aligned}$$

The Newton-Raphson method (via iterative re-weighted least squares) can be used to learn parameter iteratively.

Other than the logistic sigmoid function, there are many choices for the activation function in a binary classification.

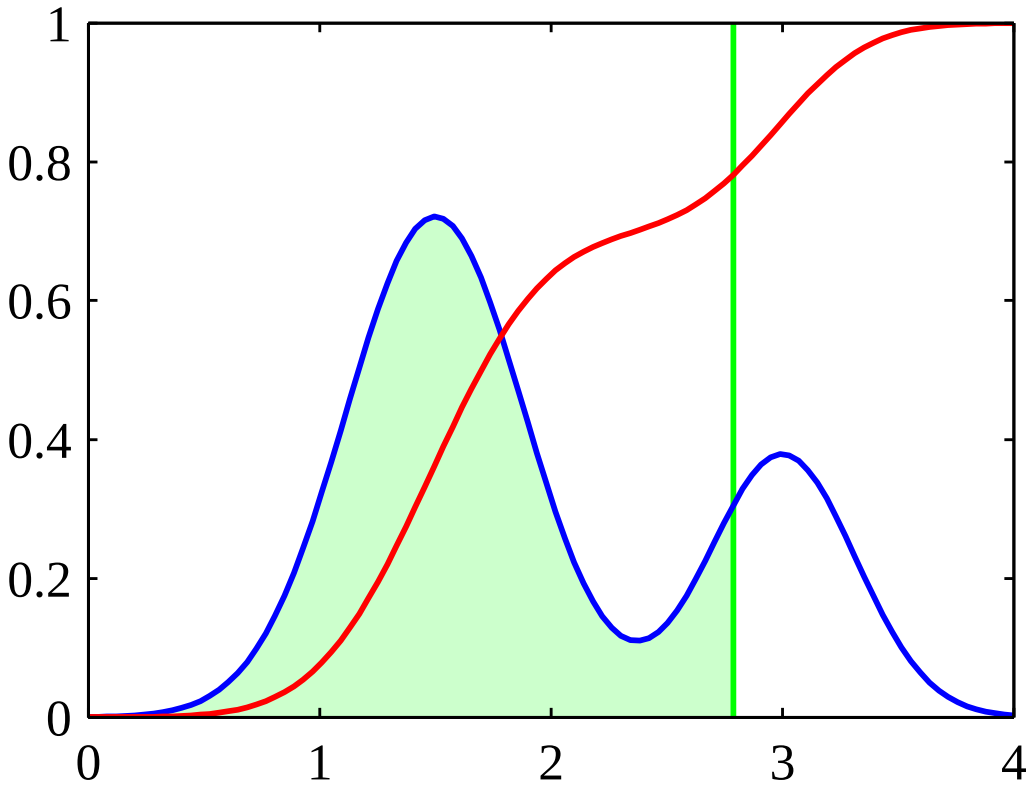
In fact, any CDF

$$f(a) = \int_{-\infty}^{a} p(t)dt$$

where $p(\cdot)$ is a PDF, is a valid activation function for binary classification since

$$0 \leq f(a) \leq 1$$

CDF and PDF



Inverse Probit Function

Definition. The CDF of a standard Gaussian

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(t|0,1)dt$$

is the **inverse probit function**. A generalized linear model with the inverse probit function as the activation function is a **probit regression**.

The data labels may be subject to errors.

For binary classification, the conditional likelihood of $(t, \boldsymbol{x})$ is

$$
\begin{aligned}
p(t|\boldsymbol{x}) &= \sum_{e=0}^{1} p(t, e|\boldsymbol{x}) \\
&= \sum_{e=0}^{1} p(e)p(t|e, \boldsymbol{x}) \\
&= \begin{cases} (1-\epsilon)\sigma(\boldsymbol{x}) + \epsilon(1-\sigma(\boldsymbol{x})), & t = 1 \\ \epsilon\sigma(\boldsymbol{x}) + (1-\epsilon)(1-\sigma(\boldsymbol{x})), & t = 0 \end{cases} \\
&= [(1-\epsilon)\sigma(\boldsymbol{x}) + \epsilon(1-\sigma(\boldsymbol{x}))]^t [\epsilon\sigma(\boldsymbol{x}) + (1-\epsilon)(1-\sigma(\boldsymbol{x}))]^{1-t}
\end{aligned}
$$

where $\epsilon = p(e)$ is the probability of error in labeling.

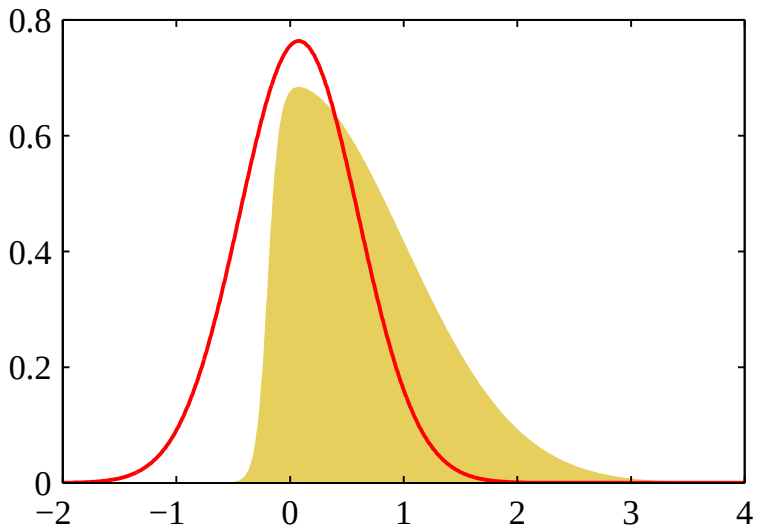
Laplace Approximation

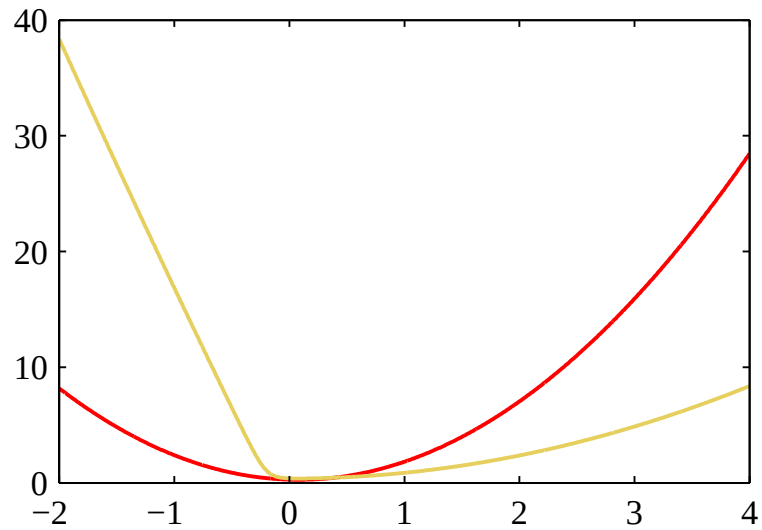
Approximate PDF by Gaussian

Let $p(\boldsymbol{x})$ be a PDF defined by unnormalized $f(\boldsymbol{x}) \geq 0$

$$p(\boldsymbol{x}) = \frac{1}{Z} f(\boldsymbol{x})$$

In Laplace approximation, $p(\boldsymbol{x})$ is approximated by a Gaussian PDF centered on a mode x_0 of $f(\boldsymbol{x})$ (or $p(\boldsymbol{x})$). The precision of the Gaussian is the negative Hessian of $\log f(\boldsymbol{x})$ at $\boldsymbol{x}_0$.





- Given $f(z)$, find a **mode** (local maximum) z_0 of $f(z)$ with

$$\left.\frac{df(z)}{dz}\right|_{z=z_0} = 0$$

- Find the negative second derivative of $\log f(z)$ at z_0

$$A = -\left.\frac{d^2}{dz^2}\log f(z)\right|_{z=z_0}$$

- The second-order approximation of $\log f(z)$ around z_0 is

$$\log f(z) \approx \log f(z_0) - \frac{1}{2}A(z-z_0)^2$$

- The Laplace approximation of $p(z)$ is

$$q(z) \propto \exp\left\{-\frac{A}{2}(z-z_0)^2\right\}$$

$$= \left(\frac{A}{2\pi}\right)^{1/2}\exp\left\{-\frac{A}{2}(z-z_0)^2\right\}$$

- Given $f(\boldsymbol{z})$, find a **mode** (local maximum) $\boldsymbol{z}_0$ of $f(\boldsymbol{z})$ with

$$\boldsymbol{\nabla} f(\boldsymbol{z})\Big|_{\boldsymbol{z}=\boldsymbol{z}_0} = \mathbf{0}$$

- Find the negative Hessian of $\log f(\boldsymbol{z})$ at $\boldsymbol{z}_0$

$$\boldsymbol{A} = -\boldsymbol{\nabla}\boldsymbol{\nabla} \log f(\boldsymbol{z})\big|_{\boldsymbol{z}=\boldsymbol{z}_0}$$

- The second-order approximation of $\log f(\boldsymbol{z})$ around $\boldsymbol{z}_0$ is

$$\log f(\boldsymbol{z}) \approx \log f(\boldsymbol{z}_0) - \frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)$$

- The Laplace approximation of $p(\boldsymbol{z})$ is

$$\begin{aligned} q(\boldsymbol{z}) &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)\right\} \\ &= \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}_0, \boldsymbol{A}^{-1}) \end{aligned}$$

Bayesian Logistic Regression

Consider a Gaussian prior for the parameters

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, \boldsymbol{S}_0)$$

For the posterior of the parameters, we have

$$p(\boldsymbol{w}|\mathcal{D}) \propto p(\boldsymbol{w})p(\mathcal{D}|\boldsymbol{w})$$

Taking the logarithm, we have

$$\begin{aligned}\log p(\boldsymbol{w}|\mathcal{D}) = &-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{m}_0)^T\boldsymbol{S}_0^{-1}(\boldsymbol{w}-\boldsymbol{m}_0) \\ &+\sum_{n=1}^{N} t_n \log y_n + (1-t_n)\log(1-y_n) + \text{const}\end{aligned}$$

To proceed, we apply the Laplace approximation to $p(\boldsymbol{w}|\mathcal{D})$.

- The mode of $p(\boldsymbol{w}|\mathcal{D})$ is $\boldsymbol{w}_{\text{MAP}}$
- The negative Hessian of $\log p(\boldsymbol{w}|\mathcal{D})$ at $\boldsymbol{w}_{\text{MAP}}$ is

$$-\boldsymbol{\nabla}\boldsymbol{\nabla}\log p(\boldsymbol{w}|\mathcal{D}) = \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T$$

Hence, the Laplace approximation to $p(\boldsymbol{w}|\mathcal{D})$ is

$$p(\boldsymbol{w}|\mathcal{D}) \approx q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{\text{MAP}}, \boldsymbol{S}_N)$$

where

$$\boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T$$

The prediction is the integration of the prediction over the posterior distribution of the parameters.

With the Laplace approximation, we have

$$\begin{aligned} p(\mathcal{C}_1|\boldsymbol{x}, \mathcal{D}) &= \int p(\mathcal{C}_1|\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w} \\ &\approx \int \sigma(\boldsymbol{w}^T \boldsymbol{\phi}) q(\boldsymbol{w}) d\boldsymbol{w} \\ &= \int \sigma(a) p(a) da \end{aligned}$$

where $a = \boldsymbol{w}^T \boldsymbol{\phi}$.

Note that $a = \boldsymbol{w}^T\boldsymbol{\phi}$ is Gaussian as $\boldsymbol{w}$ is Gaussian. In particular, the mean and variance of $a = \boldsymbol{w}^T\boldsymbol{\phi}$ is

$$\mu_a = \int a(\boldsymbol{w})\, q(\boldsymbol{w}) d\boldsymbol{w} = \int \boldsymbol{w}^T\boldsymbol{\phi}\, q(\boldsymbol{w}) d\boldsymbol{w} = \boldsymbol{w}_{\text{MAP}}^T\boldsymbol{\phi}$$

$$\sigma_a^2 = \int (\boldsymbol{w}^T\boldsymbol{\phi} - \mu_a)^2\, q(\boldsymbol{w}) d\boldsymbol{w} = \boldsymbol{\phi}^T\boldsymbol{S}_N\boldsymbol{\phi}$$

Thus

$$\int \sigma(a)p(a)da = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2)da$$

A logistic sigmoid function can be approximated by an inverse probit function

$$\sigma(a) \approx \Phi(\lambda a), \ \lambda^2 = \frac{\pi}{8}$$

It can be shown that

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)da = \Phi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}\right)$$

Thus we have

$$\int \sigma(a)\mathcal{N}(a|\mu, \sigma^2)da \approx \sigma\left(\kappa(\sigma^2)\mu\right), \ \kappa(\sigma^2) = (1 + \lambda^2\sigma^2)^{-1/2}$$

So

$$p(\mathcal{C}_1|\boldsymbol{x}, \mathcal{D}) \approx \sigma\left(\kappa(\sigma_a^2)\mu_a\right)$$

Note that the decision boundary is given by $\mu_a = \boldsymbol{w}_{\text{MAP}}^T\boldsymbol{\phi} = 0$.