Linear Models for Classification

Mingmin Chi

Fudan University, Shanghai, China

Outline

Supervised Learning

- Regression: $\mathbf{x} \to y$ or $\mathbf{y}$, y is a real value
- Classification: $\mathbf{x} \to y = \mathcal{C}_k,\ k = 1, \cdots, K$, e.g., for probabilistic models
  - if $K = 2$ the binary case, $y = 1 \to \mathcal{C}_1$ and $y = 0 \to \mathcal{C}_2$
  - if $K > 2$ the multiple cases, we can use 1-of-K coding scheme, $\mathbf{y} \in \mathcal{R}^K$, $y_j = 1,$ if the class is $\mathcal{C}_j$; otherwise 0; e.g., $\mathbf{y} = (0, 1, 0, 0, 0)^\top$

In classification

The input space is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces

Classification Paradigms

- Discriminant function: directly assigns each vector $\mathbf{x}$ to a specific class

Classification Paradigms

- Discriminant function: directly assigns each vector $\mathbf{x}$ to a specific class
- Probability inference: Modeling the conditional probability distribution $P(\mathcal{C}_k|\mathbf{x})$ in an inference stage, then uses this distribution to make optimal decision

Classification Paradigms

- Discriminant function: directly assigns each vector $\mathbf{x}$ to a specific class
- Probability inference: Modeling the conditional probability distribution $P(\mathcal{C}_k|\mathbf{x})$ in an inference stage, then uses this distribution to make optimal decision
  - Discriminative models: modeling $P(\mathcal{C}_k|\mathbf{x})$ directly, e.g.,

Classification Paradigms

- Discriminant function: directly assigns each vector $\mathbf{x}$ to a specific class
- Probability inference: Modeling the conditional probability distribution $P(\mathcal{C}_k|\mathbf{x})$ in an inference stage, then uses this distribution to make optimal decision
  - Discriminative models: modeling $P(\mathcal{C}_k|\mathbf{x})$ directly, e.g., parametric models
  - Generative models: modeling the class-conditional densities given by $p(\mathbf{x}|\mathcal{C}_k)$, together with the prior probabilities $P(\mathcal{C}_k)$, then using Bayer's Theorem to compute the posterior probabilities:

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{P(\mathbf{x})}$$

Generalized Linear Models

- Linear classification models: $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$

Generalized Linear Models

- Linear classification models: $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$
- Generalized linear models: generally we want to predict posterior probabilities $(0, 1)$,

$$y(\mathbf{x}) = f(\mathbf{w}^\top \mathbf{x} + w_0)$$

  These models are called generalized linear models
  - $f(\cdot)$ is known as activation function
  - the decision surfaces correspond to

Generalized Linear Models

- Linear classification models: $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$
- Generalized linear models: generally we want to predict posterior probabilities $(0, 1)$,

$$y(\mathbf{x}) = f(\mathbf{w}^\top \mathbf{x} + w_0)$$

  These models are called generalized linear models
  - $f(\cdot)$ is known as activation function
  - the decision surfaces correspond to $y(\mathbf{x}) = \text{constant}$

Generalized Linear Models

- Linear classification models: $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$
- Generalized linear models: generally we want to predict posterior probabilities $(0, 1)$,

$$y(\mathbf{x}) = f(\mathbf{w}^\top \mathbf{x} + w_0)$$

  These models are called generalized linear models
  - $f(\cdot)$ is known as activation function
  - the decision surfaces correspond to $y(\mathbf{x}) = \text{constant} \rightarrow \mathbf{w}^\top \mathbf{x} + w_0 = \text{constant}$
  - the decision surfaces are linear functions of $\mathbf{x}$, even if $f(\cdot)$ is nonlinear

General Formulation

- The simplest choice of discriminant function is one which is linear in the components of $\mathbf{x}$ and which can therefore be written as

General Formulation

- The simplest choice of discriminant function is one which is linear in the components of $\mathbf{x}$ and which can therefore be written as

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

  where the d-dimensional vector $\mathbf{w}$ is referred to as the weight vector and the parameter w_0 as the bias (or sometimes $-w_0$ as a threshold)
- The input vector $\mathbf{x}$ is assigned according to

$$\mathbf{x} \in \begin{cases} \mathcal{C}_1, & \text{if } y(\mathbf{x}) \leq 0 \\ \mathcal{C}_2, & \text{otherwise} \end{cases}$$

Geometrical interpretation

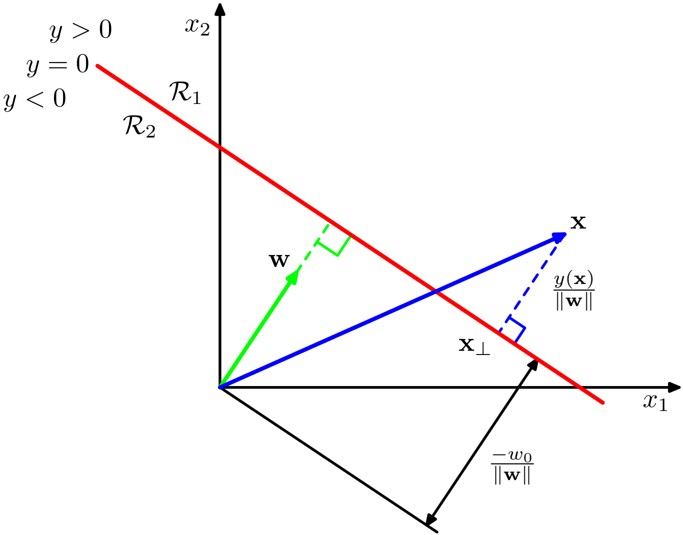
- The decision boundary $y(\mathbf{x}) = 0$ corresponds to a $(d-1)$-dimensional hyperplane in d-dimensional $\mathbf{x}$-space.

Geometrical interpretation

- The decision boundary $y(\mathbf{x}) = 0$ corresponds to a $(d-1)$-dimensional hyperplane in d-dimensional $\mathbf{x}$-space.
- If $\mathbf{x}_A$ and $\mathbf{x}_B$ on the decision surface $\Rightarrow$ $\mathbf{w}^\top(\mathbf{x}_A - \mathbf{x}_B) = 0$

  $\Rightarrow$ The weight vector $\mathbf{w}$ is orthogonal to any vector lying in the hyperplane
- $\mathbf{w}$ determines the orientation of the decision surface

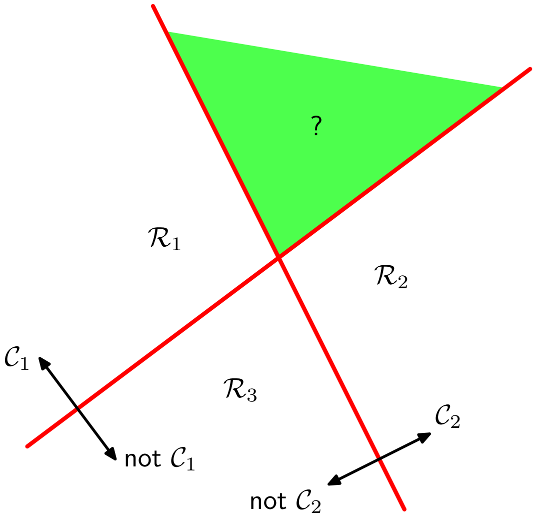
Geometrical interpretation (ctd.)



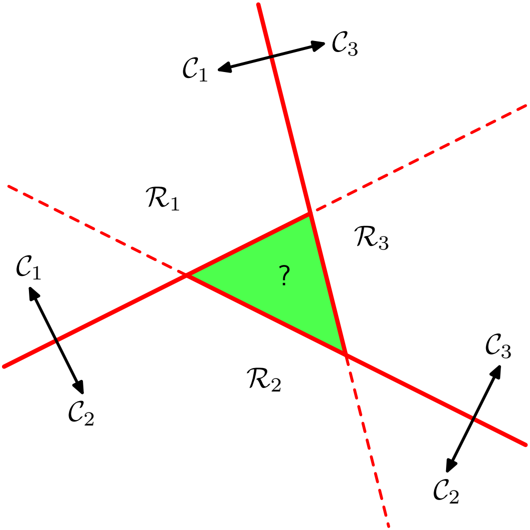
Geometrical interpretation (ctd)

- If $\mathbf{x}$ on decision surface,then $y(\mathbf{x}) = 0$
  $\Rightarrow$ The normal distance from the origin to the decision surface is given by
  $$\frac{\mathbf{w}^\top \mathbf{x}}{||\mathbf{w}||} = -\frac{w_0}{||\mathbf{w}||}$$
- The bias w_0 determines the position of the hyperplane in $\mathbf{x}$-space

One-Versus-the-Rest Strategy



One-Versus-One Strategy



K-Class Discriminant Functions

- We can avoid those ambiguousness by using one discriminant function $y_k(\mathbf{x})$ for each class $\mathcal{C}_k$ of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

K-Class Discriminant Functions

- We can avoid those ambiguousness by using one discriminant function $y_k(\mathbf{x})$ for each class $\mathcal{C}_k$ of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

- A new point $\mathbf{x}$ is then assigned to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$.

K-Class Discriminant Functions

- We can avoid those ambiguousness by using one discriminant function $y_k(\mathbf{x})$ for each class $\mathcal{C}_k$ of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

- A new point $\mathbf{x}$ is then assigned to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$.
- The decision boundary separating class $\mathcal{C}_k$ from class $\mathcal{C}_j$ is given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$, which for linear discriminant, correspond to a hyperplane of the form

$$(\mathbf{w}_k - \mathbf{w}_j)^\top \mathbf{x} + (w_{k0} - w_{j0}) = 0.$$

K-Class Discriminant Functions

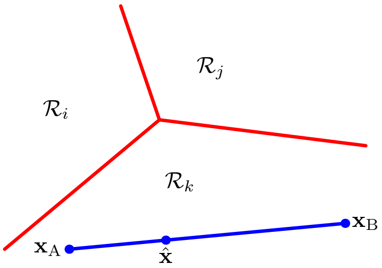
- We can avoid those ambiguousness by using one discriminant function $y_k(\mathbf{x})$ for each class $\mathcal{C}_k$ of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

- A new point $\mathbf{x}$ is then assigned to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$.
- The decision boundary separating class $\mathcal{C}_k$ from class $\mathcal{C}_j$ is given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$, which for linear discriminant, correspond to a hyperplane of the form

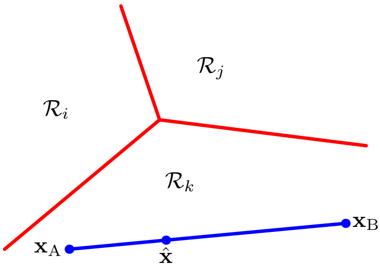
$$(\mathbf{w}_k - \mathbf{w}_j)^\top \mathbf{x} + (w_{k0} - w_{j0}) = 0.$$

Connected and Convex Decision Regions



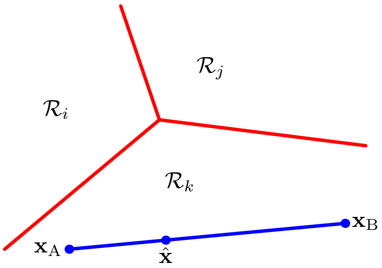
- Any point $\hat{\mathbf{x}}$ that lies on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ can be expressed in the form:

Connected and Convex Decision Regions



- Any point $\hat{\mathbf{x}}$ that lies on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ can be expressed in the form:$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B, \ 0 \leq \lambda \leq 1$
- Due to the linearity of the discriminant functions, it follows that:

Connected and Convex Decision Regions



- Any point $\hat{\mathbf{x}}$ that lies on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ can be expressed in the form: $\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1-\lambda)\mathbf{x}_B,\ 0 \le \lambda \le 1$
- Due to the linearity of the discriminant functions, it follows that: $y(\hat{\mathbf{x}}) = \lambda y(\mathbf{x}_A) + (1-\lambda) y(\mathbf{x}_B)$
- $y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}}) \to \hat{\mathbf{x}}$ lies inside $\mathcal{R}_k$

Geometrical interpretation

By analogy with the two-category case

- The normal to the decision boundary is given by the difference between two weight vectors $\mathbf{w}_k - \mathbf{w}_j$.
- The perpendicular distance of the decision boundary from the origin is given by

$$l = \frac{(w_{k0} - w_{j0})}{\| \mathbf{w}_k - \mathbf{w}_j \|}$$

Error Function

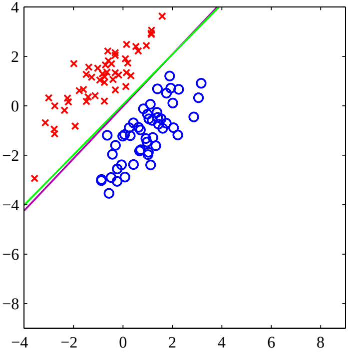
Problem Setting

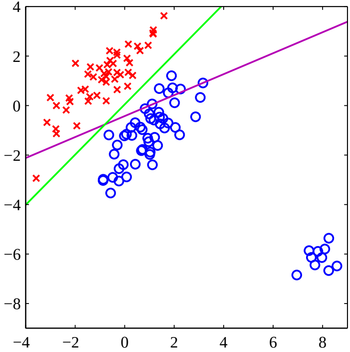
- Each class $\mathcal{C}_k$ is described by its own linear model: $y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_0$
- Using vector notation: $\mathbf{y}(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ by omitting the bias w_0, $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_K]$
- Considering a training dataset $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$ and $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_N]$

The sum-of-squares error function can be written as

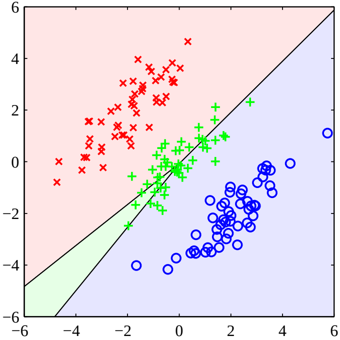
$$E_D(\mathbf{W}) = \frac{1}{2}(\mathbf{XW} - \mathbf{T})^\top(\mathbf{XW} - \mathbf{T})$$
$$\Rightarrow \mathbf{W} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{T} = \mathbf{X}^\dagger\mathbf{T}$$
$$\Rightarrow \mathbf{y}(\mathbf{x}) = \mathbf{W}^\top\mathbf{x} = \mathbf{T}^\top(\mathbf{X}^\dagger)^\top\mathbf{x}$$

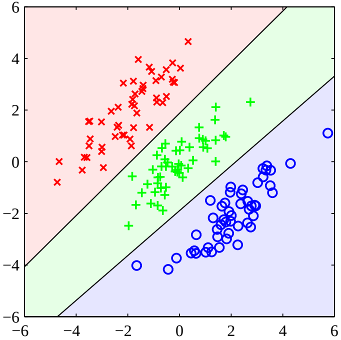
Drawbacks of LSC: Sensitive to Outlies





Drawbacks of LSC





Least squares corresponds to maximum likelihood under the assumption of a Gaussian conditional distribution.

Introduction

In terms of dimensionality reduction, an alternative linear classification model can project the data onto a lower dimensional space, e.g., one-dimensional projection $\mathbf{x} \in \mathcal{R}^d \to y \in \mathcal{R}$ given by

$$y = \mathbf{w}^\top \mathbf{x}$$

by one-dimensional projection

- leading to a considerable loss of information
- classes which are well separated in the original d-dimensional space may become strongly overlapping in one dimension

Solution: by adjusting the components of the weight vector **w**, we can select a projection which maximizes the class separation

Means in Original and Projection Space

Consider a two-class problem in which there are N_1 points of class $\mathcal{C}_1$ and N_2 points of class $\mathcal{C}_2$

- The mean vectors in the original space:

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \mathbf{x}_i$$

- The means vectors in the projection space with some projected direction **w**:

$$\mu_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \mathbf{w}^\top \mathbf{x}_i$$

Binary Classes

To separate the classes, we can separate the projected class means, i.e.,

Binary Classes

To separate the classes, we can separate the projected class means, i.e., by choosing a projected direction **w** so as to maximize

$$\mu_2 - \mu_1 = \mathbf{w}^\top(\mathbf{m}_2 - \mathbf{m}_1)$$

- Problem: arbitrarily big of **w**
- Solution: constraining **w** to have unit length, so that

$$\sum_i w_i^2 = 1$$

Binary Classes

To separate the classes, we can separate the projected class means, i.e., by choosing a projected direction **w** so as to maximize

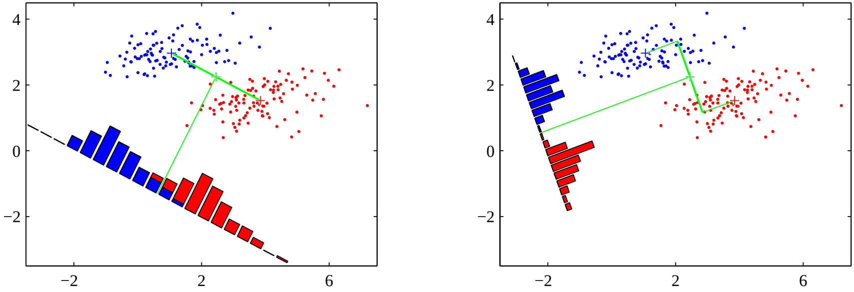
$$\mu_2 - \mu_1 = \mathbf{w}^\top(\mathbf{m}_2 - \mathbf{m}_1)$$

- Problem: arbitrarily big of **w**
- Solution: constraining **w** to have unit length, so that

$$\sum_i w_i^2 = 1$$

Using a Lagrange multiplier to perform the constrained maximization, we then find that $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$

Problem?



The goal is to find a direction that maximizes the between class variance while minimizing the within class variance at the same time

Main Idea

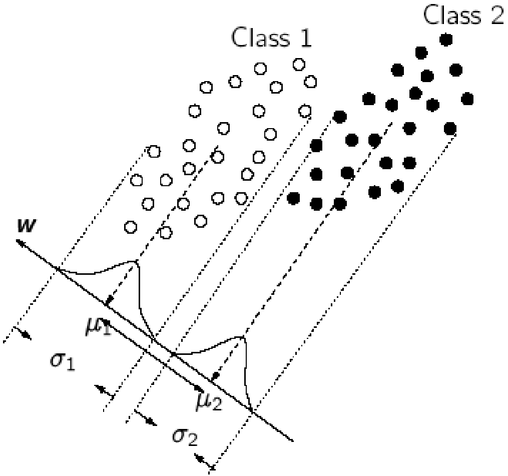
- Fisher's idea was to look for a direction **w** that separates the class means well (when projected onto the found direction) while achieving a small variance around these means
- The quantity measuring the difference between the means is called between-class variance:

$$\mu_2 - \mu_1$$

and the quantity measuring the variance around these class means is called within-class variance defined by:

$$\underbrace{\sum_{i \in \mathcal{C}_1} (\mathbf{w}^\top \mathbf{x}_i - \mu_1)(\mathbf{w}^\top \mathbf{x}_i - \mu_1)^\top}_{\sigma_1} + \underbrace{\sum_{i \in \mathcal{C}_2} (\mathbf{w}^\top \mathbf{x}_i - \mu_2)(\mathbf{w}^\top \mathbf{x}_i - \mu_2)^\top}_{\sigma_2}$$

Illustration



Main Idea (ctd.)

- Then maximizing the between class variance and minimizing the within class variance is given by

$$J(\mathbf{w}) = \frac{(\mu_2 - \mu_1)^2}{\sigma_1 + \sigma_2}$$

- This will yield a direction **w** such that the ratio of between-class variance (i.e. separation) and within-class variance (i.e. overlap) is maximal
- The quantity measuring the difference between the means is called between class covariance, so we have

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}$$

Main Idea (ctd.)

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}$$

This is usually referred to a Rayleigh coefficient, where

- Define between-class scatter matrix:

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top$$

- Define within-class scatter matrix:

$$\mathbf{S}_W = \sum_k \sum_{i \in \mathcal{C}_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^\top$$

Differentiating w.r.t **w**, $\max J(\mathbf{w}) \Rightarrow$

$$(\mathbf{w}^\top \mathbf{S}_B \mathbf{w})\mathbf{S}_W \mathbf{w} = (\mathbf{w}^\top \mathbf{S}_W \mathbf{w})\mathbf{S}_B \mathbf{w}$$

Finding **w**

One particularly nice property of Fisher's discriminant is that

- The criterion function has a global solution (although not necessarily unique)
- Such a globally optimal **w** maximizing the criterion function can be found by solving an eigenvalue problem

It is well known, that the **w** maximizing the criterion function is the leading eigenvector of the generalized eigenproblem

$$\begin{aligned} \mathbf{S}_B\mathbf{w} &= \lambda\mathbf{S}_W\mathbf{w} \\ &\Rightarrow \mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1) \end{aligned}$$

Finding **w** (ctd.)

Examining the eigenproblem closer one finds an even simpler way of obtaining the optimal **w** and remember we have

$$(\mathbf{w}^\top \mathbf{S}_B \mathbf{w})\mathbf{S}_W \mathbf{w} = (\mathbf{w}^\top \mathbf{S}_W \mathbf{w})\mathbf{S}_B \mathbf{w}$$

- Since $\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top$, $\mathbf{S}_B\mathbf{w}$ will always point in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
- We can also see that only the direction of **w** matters, not its length
- We can drop all scalar factors and multiply both sides of $\mathbf{S}_B\mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}$ by $\mathbf{S}_W^{-1}$ and we can also get the solution of **w**

If $\mathbf{S}_W$ is isotropic, what's the **w**?

Excises

- Implementation of Fisher's linear discriminant (Linear Discriminant Analysis, LDA)
- Relation to least squares
- The shortcoming of Fisher's linear discriminant
- Fisher's discriminant for multiple classes

Introduction

- Rosenblatt (1962) called single-layer networks with threshold activation functions as perceptrons

Introduction

- Rosenblatt (1962) called single-layer networks with threshold activation functions as perceptrons
- Due to the limited capabilities of a network with a single layer of weights, Rosenblatt used a layer of fixed processing elements to transform the raw input data. These processing elements can be regarded as the basis functions of a generalized linear discriminant:

Introduction

- Rosenblatt (1962) called single-layer networks with threshold activation functions as perceptrons
- Due to the limited capabilities of a network with a single layer of weights, Rosenblatt used a layer of fixed processing elements to transform the raw input data. These processing elements can be regarded as the basis functions of a generalized linear discriminant:

$$y(\mathbf{x}) = f(\mathbf{w}^\top \Phi(\mathbf{x}))$$

- The nonlinear activation function $f(\cdot)$ is given by a step function of the form

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

The Criterion

- The total number of misclassified patterns - piecewise constant function, discontinuities
- the Perceptron criterion
  - probabilistic models, $t \in \{0, 1\}$?

The Criterion

- The total number of misclassified patterns - piecewise constant function, discontinuities
- the Perceptron criterion
  - probabilistic models, $t \in \{0, 1\}? \Rightarrow t \in \{-1, +1\}$
  - $\mathcal{C}_1 : \mathbf{w}^\top \Phi_i > 0; \mathcal{C}_2 : \mathbf{w}^\top \Phi(\mathbf{x}_i) < 0$

The Criterion

- The total number of misclassified patterns - piecewise constant function, discontinuities
- the Perceptron criterion
  - probabilistic models, $t \in \{0, 1\}? \Rightarrow t \in \{-1, +1\}$
  - $\mathcal{C}_1 : \mathbf{w}^\top \Phi_i > 0; \mathcal{C}_2 : \mathbf{w}^\top \Phi(\mathbf{x}_i) < 0$
  - therefore, using the $t \in \{-1, +1\}$ target coding scheme, all patterns satisfy

$$\mathbf{w}^\top \Phi(\mathbf{x}_i) t_i > 0$$

Perceptron Criterion

- $\mathbf{x}_i \Rightarrow \Phi(\mathbf{x}_i) = \Phi_i$

Perceptron Criterion

- $\mathbf{x}_i \Rightarrow \Phi(\mathbf{x}_i) = \Phi_i$, $\mathbf{x}_i \Rightarrow t_i$

Perceptron Criterion

- $\mathbf{x}_i \Rightarrow \Phi(\mathbf{x}_i) = \Phi_i$, $\mathbf{x}_i \Rightarrow t_i$
- $\mathcal{C}_1 : \ \mathbf{w}^\top \Phi_i > 0; \mathcal{C}_2 : \ \mathbf{w}^\top \Phi_i < 0$

Perceptron Criterion

- $\mathbf{x}_i \Rightarrow \Phi(\mathbf{x}_i) = \Phi_i$, $\mathbf{x}_i \Rightarrow t_i$
- $\mathcal{C}_1 : \ \mathbf{w}^\top \Phi_i > 0; \mathcal{C}_2 : \ \mathbf{w}^\top \Phi_i < 0$
- $\mathcal{C}_1 : \ t_i = +1; \mathcal{C}_2 : \ t_i = -1$

Perceptron Criterion

- $\mathbf{x}_i \Rightarrow \Phi(\mathbf{x}_i) = \Phi_i$, $\mathbf{x}_i \Rightarrow t_i$
- $\mathcal{C}_1 : \mathbf{w}^\top \Phi_i > 0; \mathcal{C}_2 : \mathbf{w}^\top \Phi_i < 0$
- $\mathcal{C}_1 : t_i = +1; \mathcal{C}_2 : t_i = -1$

The error function of the perceptrons

$$E_P(\mathbf{w}) = -\sum_{\Phi_i \in \mathcal{M}} \mathbf{w}^\top \Phi_i t_i$$

where $\mathcal{M}$ is the set of vectors Φ_i which are misclassified by the current weight vector **w**

The perceptron criterion is continuous and piecewise-linear

Perceptron Learning

Stochastic gradient descent

- If we apply the pattern-by-pattern gradient descent rule to the perceptron criterion we obtain

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi(\mathbf{x}_i) t_i$$

Perceptron Learning

Stochastic gradient descent

- If we apply the pattern-by-pattern gradient descent rule to the perceptron criterion we obtain

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi(\mathbf{x}_i) t_i$$

- If the pattern $\mathbf{x}_i$ is correctly classified, do nothing, i.e., $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)}$

Perceptron Learning

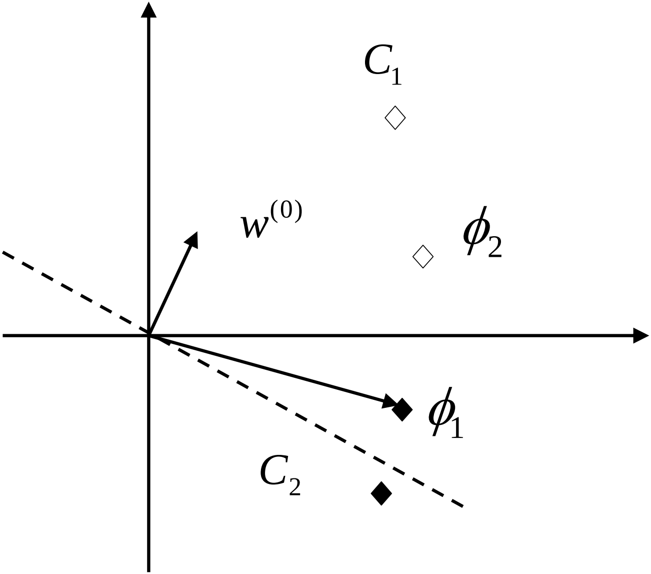
Stochastic gradient descent

- If we apply the pattern-by-pattern gradient descent rule to the perceptron criterion we obtain

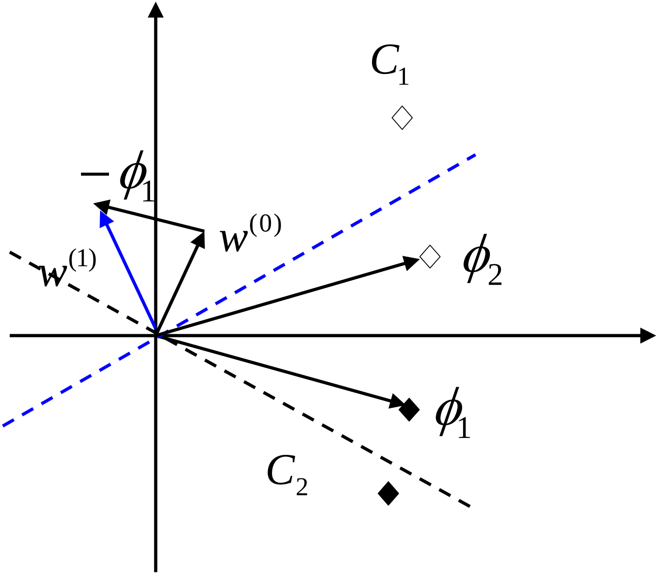
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi(\mathbf{x}_i) t_i$$

- If the pattern $\mathbf{x}_i$ is correctly classified, do nothing, i.e., $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)}$
- Otherwise the pattern is misclassified, updating the weight vector in the new iteration.

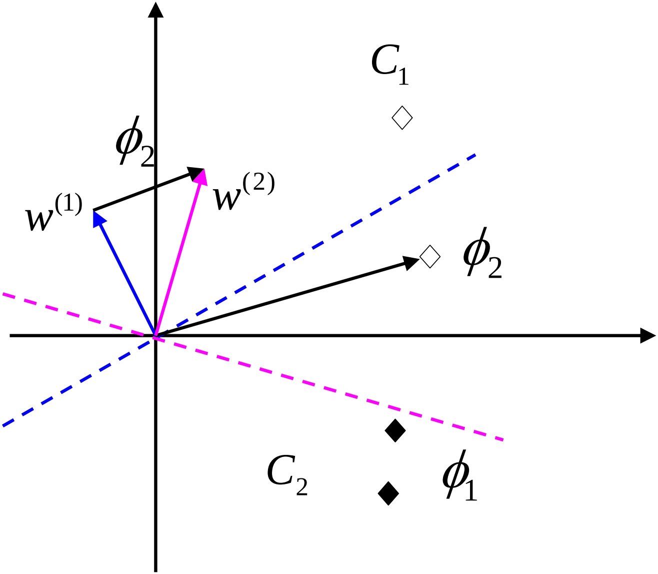
Perceptron Learning



Perceptron Learning



Perceptron Learning



Perceptron Convergence Theorem

- In the single update of the perceptron learning, we can see the contribution of error from a misclassified pattern, will be reduced:

$$-\mathbf{w}^{(\tau+1)}\Phi(\mathbf{x}_i)\mathbf{y}_i = -\mathbf{w}^{(\tau)}\Phi(\mathbf{x}_i)\mathbf{y}_i - (\Phi(\mathbf{x}_i)\mathbf{y}_i)^\top\Phi(\mathbf{x}_i)\mathbf{t}_i$$

Perceptron Convergence Theorem

- In the single update of the perceptron learning, we can see the contribution of error from a misclassified pattern, will be reduced:

$$-\mathbf{w}^{(\tau+1)}\Phi(\mathbf{x}_i)\mathbf{y}_i = -\mathbf{w}^{(\tau)}\Phi(\mathbf{x}_i)\mathbf{y}_i - (\Phi(\mathbf{x}_i)\mathbf{y}_i)^\top\Phi(\mathbf{x}_i)\mathbf{t}_i < -\mathbf{w}^{(\tau)}\Phi(\mathbf{x}_i)\mathbf{y}_i$$

- Does the contribution of the error from other misclassified patterns will be reduced?
- Is the perceptron learning rule guaranteed to reduce the total error function at each stage?

Perceptron Convergence Theorem

- In the single update of the perceptron learning, we can see the contribution of error from a misclassified pattern, will be reduced:

$$-\mathbf{w}^{(\tau+1)}\Phi(\mathbf{x}_i)\mathbf{y}_i = -\mathbf{w}^{(\tau)}\Phi(\mathbf{x}_i)\mathbf{y}_i-(\Phi(\mathbf{x}_i)\mathbf{y}_i)^\top\Phi(\mathbf{x}_i)\mathbf{t}_i < -\mathbf{w}^{(\tau)}\Phi(\mathbf{x}_i)\mathbf{y}_i$$

- Does the contribution of the error from other misclassified patterns will be reduced?
- Is the perceptron learning rule guaranteed to reduce the total error function at each stage?
- Perceptron convergence theorem states that if there exists an exact solution, then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps

Logistic Sigmoid Function

Considering a binary classification problem, the posterior probability for class $\mathcal{C}_1$ can be written as

Logistic Sigmoid Function

Considering a binary classification problem, the posterior probability for class $\mathcal{C}_1$ can be written as

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

where

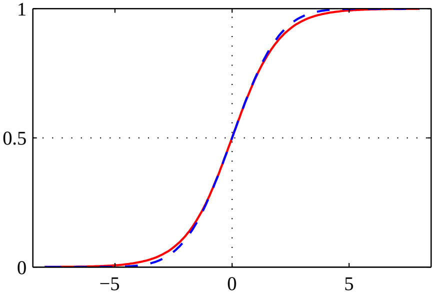
$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)}$$

The $\sigma(a)$ is the logistic sigmoid activation function defined by:

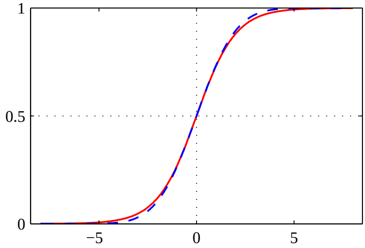
$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

Logistic Sigmoid Function (ctd.)

- Symmetry property: $\sigma(-a) = 1 - \sigma(a)$
- The inverse of the logistic sigmoid is given by $a = \ln\left(\frac{\sigma}{1-\sigma}\right)$ -logit function as $a = \ln[p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})]$ - log odds

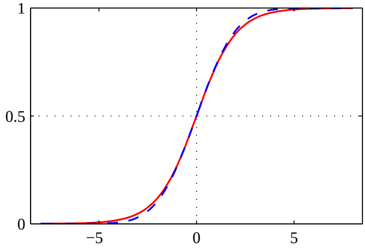


Logistic Sigmoid Function (ctd.)



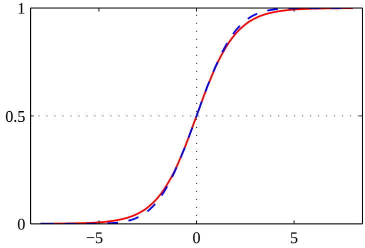
- The logistic form of the sigmoid maps the interval $(-\infty, +\infty)$ onto $(0, 1)$

Logistic Sigmoid Function (ctd.)



- The logistic form of the sigmoid maps the interval $(-\infty, +\infty)$ onto $(0, 1)$
- If $|a|$ is small, then the logistic sigmoid function $\sigma(a)$ can be approximated by a linear function

Logistic Sigmoid Function (ctd.)



- The logistic form of the sigmoid maps the interval $(-\infty, +\infty)$ onto $(0, 1)$
- If $|a|$ is small, then the logistic sigmoid function $\sigma(a)$ can be approximated by a linear function
- The use of the logistic sigmoid activation function allows the outputs of the discriminant to be interpreted as *a posteriori*

Logistic Sigmoid in Multi-class Case

If there are more than two classes then an extension of the previous analysis leads to a generalization of the logistic sigmoid called a normalized exponential or softmax:

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)}$$

Logistic Sigmoid in Multi-class Case

If there are more than two classes then an extension of the previous analysis leads to a generalization of the logistic sigmoid called a normalized exponential or softmax:

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)$

Continuous Inputs - Binary Case

Assume that $p(\mathbf{x}|\mathcal{C}_k)$ are Gaussian:

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{d/2}} \frac{1}{(\Sigma)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

Remember we have

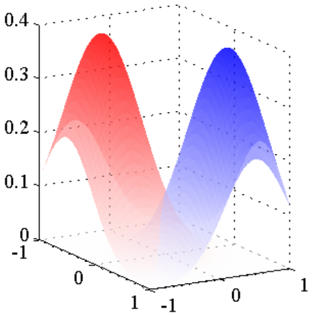
$$P(\mathcal{C}_1|\mathbf{x}) \quad = \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{\sum_k p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)} = \frac{1}{1+\exp(-a)} = \sigma(a)$$

where $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)}$. With common covariance matrices, we have

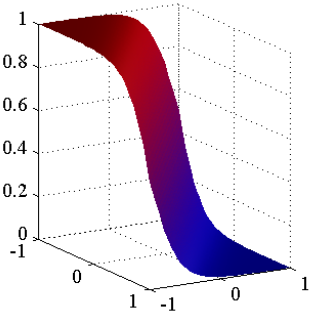
$$P(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$

where
$$\begin{cases} \mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{P(\mathcal{C}_1)}{P(\mathcal{C}_2)} \end{cases}$$

Continuous Inputs - Binary Case(ctd.)



$p(\mathbf{x}|\mathcal{C}_1)$ and $p(\mathbf{x}|\mathcal{C}_2)$



$P(\mathcal{C}_1|\mathbf{x})$

Continuous Inputs - Multiple Class Problem

For the general case of K classes,

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)}$$

Continuous Inputs - Multiple Class Problem

For the general case of K classes,

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

and

$$a_k = -\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x} + (\Sigma^{-1}\boldsymbol{\mu}_k)^\top \mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\mathcal{C}_k)$$

With common covariance matrices, we have

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

$$\text{where } \begin{cases} \mathbf{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \\ w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\mathcal{C}_k) \end{cases}$$

Continuous Inputs - Multiple Class Problem

For the general case of K classes,

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

and

$$a_k = -\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x} + (\Sigma^{-1}\boldsymbol{\mu}_k)^\top \mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\mathcal{C}_k)$$

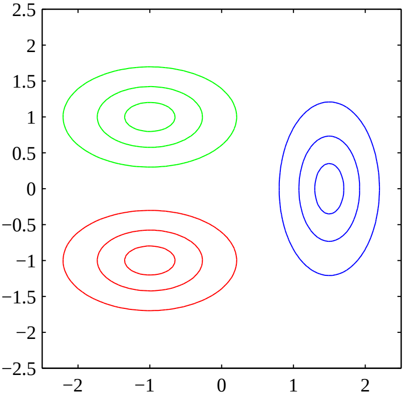
With common covariance matrices, we have

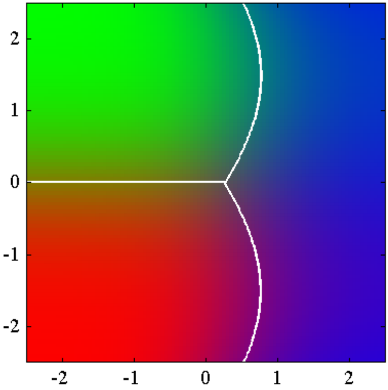
$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

$$\text{where } \begin{cases} \mathbf{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \\ w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\mathcal{C}_k) \end{cases}$$

Continuous Inputs - Quadratic Discriminant

If allowing different class-conditional density has own covariance matrices, we will obtain quadratic functions $\mathbf{x} \Rightarrow$ a quadratic discriminant





Excises

- Parameter estimation using Maximum Likelihood for binary class problems
- Parameter estimation using Maximum Likelihood for multiple class problems

Discrete features

For simplicity, we consider a binary feature values $x_i \in \{0, 1\}$ and d-feature values are independent (naive Bayes assumption)

The class-conditional distribution of $\mathcal{C}_k$:

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{d} \mu_{ki}^{x_i}(1 - \mu_{ki})^{1-x_i}$$

The posterior probability

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)}$$

Discrete features

For simplicity, we consider a binary feature values $x_i \in \{0, 1\}$ and d-feature values are independent (naive Bayes assumption)

The class-conditional distribution of $\mathcal{C}_k$:

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{d} \mu_{ki}^{x_i}(1 - \mu_{ki})^{1-x_i}$$

The posterior probability

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) = \sum_{i=1}^{d}\{x_i \ln \mu_{ki} + (1 - x_i)\ln(1 - \mu_{ki}) + \ln P(\mathcal{C}_k)\}$$

Exponential Family

- Assume the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are members of the exponential family of distribution
- The corresponding posterior probabilities are given by generalized linear models
  - For $K = 2$, logistic sigmoid activation functions
  - For $K > 2$, softmax activation functions

Exponential Family (ctd.)

$$p(\mathbf{x}|\boldsymbol{\lambda}_k) = h(\mathbf{x})g(\boldsymbol{\lambda}_k)\exp\{\boldsymbol{\lambda}_k^\top \mathbf{u}(\mathbf{x})\}$$

Assume $\mathbf{u}(\mathbf{x}) = \mathbf{x}$ and introduce a scaling parameter s and let all the classes share the same s,

$$p(\mathbf{x}|\boldsymbol{\lambda}_k, s) = \frac{1}{s}h(\frac{1}{s}\mathbf{x})g(\boldsymbol{\lambda}_k)\exp\{\frac{1}{s}\boldsymbol{\lambda}_k^\top \mathbf{x}\}$$

For $K = 2$:

$$a(\mathbf{x}) = \ln\frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)} = \frac{1}{s}(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)^\top \mathbf{x} + \ln\frac{g(\boldsymbol{\lambda}_1)}{g(\boldsymbol{\lambda}_2)} + \ln\frac{P(\mathcal{C}_1)}{P(\mathcal{C}_2)}$$

For $K > 2$

$$a_k(\mathbf{x}) = \ln p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) = \frac{1}{s}\boldsymbol{\lambda}_k^\top \mathbf{x} + \ln g(\boldsymbol{\lambda}_k) + \ln P(\mathcal{C}_k)$$

Introduction

Summary for Generative Models

- Choice of class conditional densities $p(\mathbf{x}|\mathcal{C}_k)$
- Using ML for the parameter estimation
- Together with prior probability
- Using Bayes' theorem, the posterior probabilities $P(\mathcal{C}_k|\mathbf{x})$ are generalized linear function of $\mathbf{x}$

Introduction

Summary for Generative Models

- Choice of class conditional densities $p(\mathbf{x}|\mathcal{C}_k)$
- Using ML for the parameter estimation
- Together with prior probability
- Using Bayes' theorem, the posterior probabilities $P(\mathcal{C}_k|\mathbf{x})$ are generalized linear function of $\mathbf{x}$ $\Rightarrow$ implicitly finding the parameters of a generalized linear model

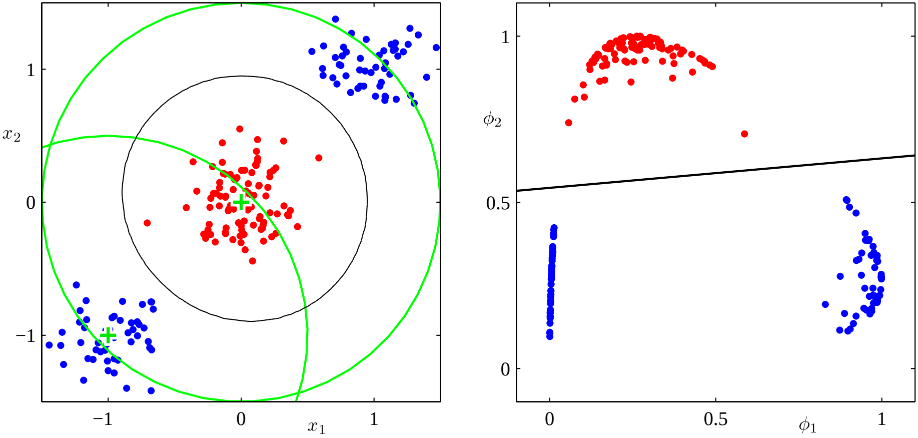
Disadvantages:

1. poor $p(\mathbf{x}|\mathcal{C}_k)$ approximation to the true distribution
2. more adaptive parameters for computing

$\Rightarrow$ Explicitly using the functional form of the generalized linear model and then determining its parameters directly by ML algorithm

Fixed basis functions

The role of nonlinear basis functions in linear classification models



Logistic Regression

We now consider generalized linear models. For simplicity, we consider a binary class problem again

$$P(\mathcal{C}_1|\Phi) = y(\Phi) = \sigma(\mathbf{w}^\top \Phi)$$

where $\sigma(\cdot)$ is the logistic sigmoid function. In statistic terminology, this model is known as logistic regression [a]

[a] This is a model for classification not regression.

Note that

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

ML for Logistic Regression

Using ML to determine the parameters of the logistic regression model

For a dataset $\{\Phi_n, t_n\}$, where $t_n \in \{0, 1\}$, the likelihood function:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}, \text{ where } y_n = P(\mathcal{C}_1|\Phi_n)$$

We can define an error function:

$$E(\mathbf{w}) = -\ln P(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\}$$

which is the cross-entropy error function, where $y_n = \sigma(a_n)$ and $a_n = \mathbf{w}^\top \Phi_n$

ML for Logistic Regression (ctd.)

Taking the gradient of the error function w.r.t. **w**, we obtain

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\Phi_n$$

- contribution to gradient from data is given by the 'error' $y_n - t_n$ times Φ_n
- taking the same form as the gradient of the-sum-of-squares error function for the linear regression model
- given a sequential algorithm, we can update the weight vectors by the stochastic gradient descent, i.e.,

ML for Logistic Regression (ctd.)

Taking the gradient of the error function w.r.t. **w**, we obtain

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\Phi_n$$

- contribution to gradient from data is given by the 'error' $y_n - t_n$ times Φ_n
- taking the same form as the gradient of the-sum-of-squares error function for the linear regression model
- given a sequential algorithm, we can update the weight vectors by the stochastic gradient descent, i.e.,

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E(\mathbf{w})$$

ML for Logistic Regression (ctd.)

Note that ML can exhibit severe over-fitting for datasets that are linearly separable

E.g., when the hyperplane corresponding to

$$\sigma = 0.5 \Rightarrow$$

ML for Logistic Regression (ctd.)

Note that ML can exhibit severe over-fitting for datasets that are linearly separable

E.g., when the hyperplane corresponding to

$$\sigma = 0.5 \Rightarrow \mathbf{w}^\top \Phi = 0$$

ML for Logistic Regression (ctd.)

Note that ML can exhibit severe over-fitting for datasets that are linearly separable

E.g., when the hyperplane corresponding to

$$\sigma = 0.5 \Rightarrow \mathbf{w}^\top \Phi = 0$$

and so all the training data belonging to class k is assigned a posterior probability $p(C_k|\mathbf{x}) = 1$.

Iterative Reweighted Least Squares (IRLS)

For logistic regression, the ML solution is or is not a closed-form solution?

Iterative Reweighted Least Squares (IRLS)

For logistic regression, the ML solution is or is not a closed-form solution? Reason?

Iterative Reweighted Least Squares (IRLS)

For logistic regression, the ML solution is or is not a closed-form solution? Reason? due to the nonlinearity of the logistic sigmoid function

Newton-Raphson Method

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(old)} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

where $\mathbf{H}$ is the Hessian matrix whose elements comprise the second derivatives of $E(\mathbf{w})$ wrt $\mathbf{w}$

IRLS for Linear Regression

Recall

$$y = \mathbf{w}^\top \phi(\mathbf{x})$$

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^\top \phi(\mathbf{x}) - t_n\}^2$$

The gradient and Hessian of the sum-of-squares error function are given by

$$\nabla E_D(\mathbf{w}) = \sum_{n=1}^{N}\{\mathbf{w}^\top \phi_n - t_n\}\phi_n = \mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{w} - \mathbf{\Phi}^\top\mathbf{t}$$

$$\mathbf{H} = \nabla\nabla E_D(\mathbf{w}) = \sum_{n=1}^{N}\phi_n\phi_n^\top = \mathbf{\Phi}^\top\mathbf{\Phi}$$

where $\mathbf{\Phi}$ is the $N \times M$ design matrix, whose n^{th} row is given by $\phi_n^\top$

IRLS for Linear Regression (ctd.)

The Newton-Raphson update then takes the form

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\{\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{w} - \mathbf{\Phi}^\top\mathbf{t}\} \\ &= (\mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{t}\end{aligned}$$

IRLS for Logistic Regression

$$E(\mathbf{w}) = -\ln P(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\Phi_n = \mathbf{\Phi}^\top(\mathbf{y}-\mathbf{t})$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\Phi_n\Phi_n^\top = \mathbf{\Phi}^\top\mathbf{R}\mathbf{\Phi}$$

where $\mathbf{R}$ is an $N \times N$ diagonal matrix with elements $R_{nn} = y_n(1-y_n)$

IRLS for Logistic Regression

$$E(\mathbf{w}) = -\ln P(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\Phi_n = \boldsymbol{\Phi}^\top(\mathbf{y}-\mathbf{t})$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\Phi_n\Phi_n^\top = \boldsymbol{\Phi}^\top\mathbf{R}\boldsymbol{\Phi}$$

where $\mathbf{R}$ is an $N \times N$ diagonal matrix with elements $R_{nn} = y_n(1-y_n)$

- No longer quadratic error function
- $\mathbf{H}$ is positive definite
- The convex function error function of $\mathbf{w}$, a unique minimum

Newton-Raphson Update Formula

$$
\begin{aligned}
\mathbf{w}^{(new)} &= \mathbf{w}^{(old)} - (\mathbf{\Phi}^\top \mathbf{R} \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top (\mathbf{y} - \mathbf{t}) \\
&= (\mathbf{\Phi}^\top \mathbf{R} \mathbf{\Phi})^{-1} \{ \mathbf{\Phi}^\top \mathbf{R} \mathbf{\Phi} \mathbf{w}^{(old)} - \mathbf{\Phi}^\top (\mathbf{y} - \mathbf{t}) \} \\
&= (\mathbf{\Phi}^\top \mathbf{R} \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{R} \mathbf{z}
\end{aligned}
$$

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

- $\mathbf{R}$ can be interpreted as variances between the mean and variance of in the logistic regression model as $\mathrm{var}[t] = y(1-y)$ and $E[t] = y$
- $\mathbf{z}$ can be interpreted as an effective target value by making a local linear approximation to the logistic sigmoid function around the current operating point $\mathbf{w}^{(old)}$ as

$$
a_n(\mathbf{w}) \simeq a_n(\mathbf{w}^{(old)}) + \left. \frac{da_n}{dy_n} \right|_{\mathbf{w}^{(old)}} (t_n - y_n)
$$

Multiclass Logistic Regression

- recall the posterior probabilities in generative models for multiclass classification

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(C_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(C_j)}$$

Multiclass Logistic Regression

- recall the posterior probabilities in generative models for multiclass classification

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(C_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

  where $a_k = \ln p(\mathbf{x}|C_k)P(C_k)$
- we can rewrite it as

$$P(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

  where the 'activations' $a_k = \mathbf{w}^\top \phi$ which is the posterior probabilities in discriminative models for multiclass classification

Multiclass Logistic Regression (cont.)

- in generative models, implicitly determine the parameters $\{\mathbf{w}_k\}$ by determining separately $p(\mathbf{x}|C_j)$ and $P(C_j)$
- in discriminative models, explicitly determine the parameters $\{\mathbf{w}_k\}$ by the use of maximum likelihood

The likelihood function

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} P(C_k|\phi_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$