

Pattern Recognition



Linear Models for Classification

林彥宇 教授
Yen-Yu Lin, Professor

國立陽明交通大學 資訊工程學系
Computer Science, National Yang Ming Chiao Tung University



Some slides are modified from Prof. Sheng-Jyh Wang
and Prof. Hwang-Tzong Chen

Classification problem

- The goal of classification is to take an input vector \mathbf{x} and to assign it to one of the K discrete classes C_k where $k = 1, 2, \dots, K$.
- The input space is divided into decision regions whose boundaries are called decision boundaries or decision surfaces

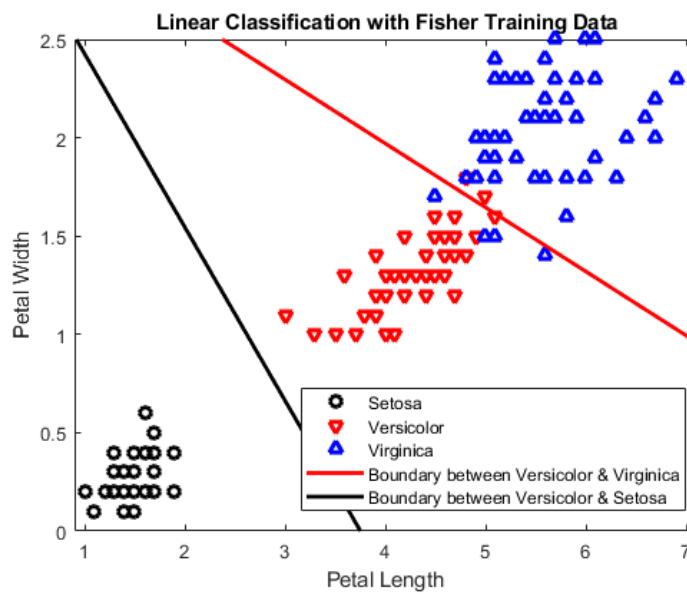
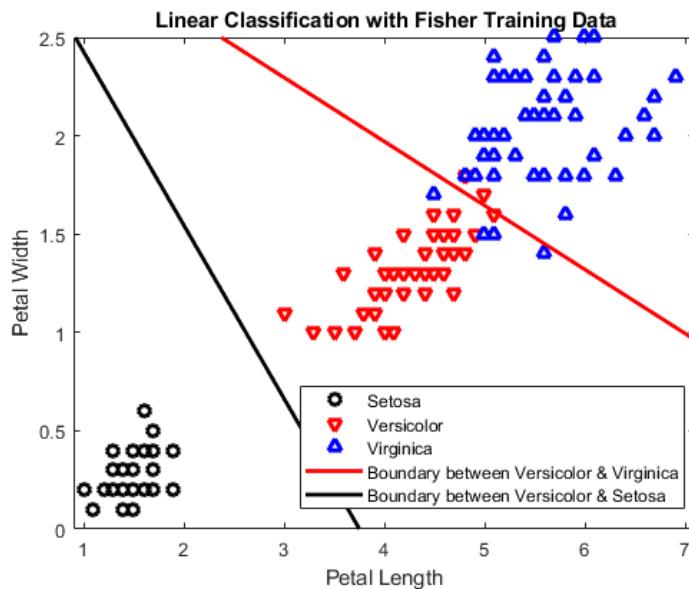


Photo: <https://ww2.mathworks.cn/help/stats/create-and-visualize-discriminant-analysis-classifier.html>

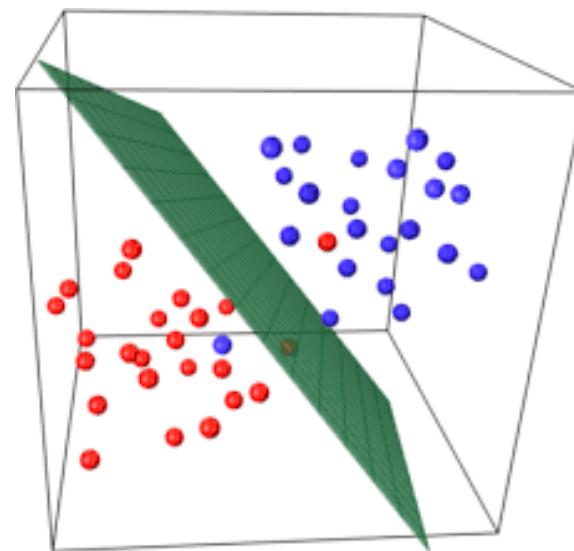


Classification problem

- **Linear models for classification:** the decision surfaces are linear functions of the input vector \mathbf{x}



2D case

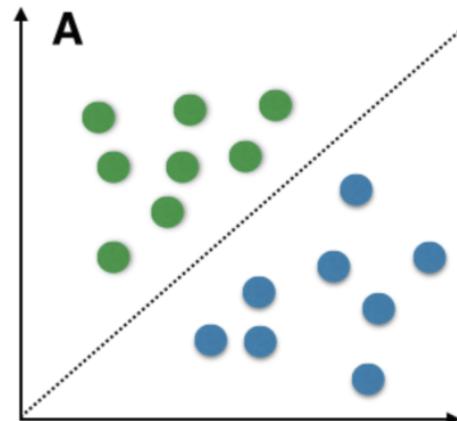


3D case

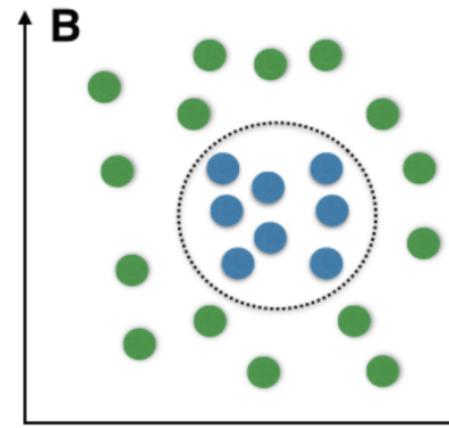


Classification problem

- **Linear models for classification:** the decision surfaces are linear functions of the input vector \mathbf{x}
- The decision surfaces are defined by $(D - 1)$ -dimensional hyperplanes within the D -dimensional input space
- Data whose classes can be separated by linear decision surfaces are said to be **linearly separable**



linearly
separable



not linearly
separable



Classification problem

- Given a training data set comprising N observations $\{\mathbf{x}_n\}_{n=1}^N$ and the corresponding target labels $\{t_n\}_{n=1}^N$, the goal of classification is to predict the label of t for a new data sample of \mathbf{x}
 - Categorical outputs, e.g., yes/no, dog/cat/other, called labels
 - A classifier assigns each input vector to one of these labels
- Binary classification: two possible labels
- Multi-class classification: multiple possible labels
- Label representation
 - Two classes: $t \in \{1, 0\}$ or $t \in \{+1, -1\}$
 - Multiple classes, e.g., $K = 5$: $t = (0, 1, 0, 0, 0)^T$ (1-of- K scheme)



Three representative linear classifiers

- Different linear models for classification
 - Discriminant functions
 - Generative approach using Bayes' theorem: $p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$
 - Discriminative approach to directly model the class-conditional density: $p(\mathcal{C}_k|\mathbf{x})$

Linear discriminant for two-class classification

- A **linear discriminant** is a linear function that takes an input vector \mathbf{x} and assigns it to one of K classes
- A linear discriminant function for two-class classification

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

where \mathbf{w} is the weight vector and w_0 is the bias

- The **decision boundary** is $y(\mathbf{x}) = 0$, i.e., classification result
$$\begin{cases} C_1, & \text{if } y(\mathbf{x}) \geq 0, \\ C_2, & \text{otherwise.} \end{cases}$$
- $y(\mathbf{x}) = 0$ is a $(D - 1)$ -dimensional hyperplane within the D -dimensional input space

Properties of A Linear discriminant

- Weight vector w is orthogonal to every vector lying within the decision boundary
 - Consider two points \mathbf{x}_A and \mathbf{x}_B , which lie on the decision boundary
 - We have $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$, leading to

$$\mathbf{w}^T (\mathbf{x}_A - \mathbf{x}_B) = 0$$

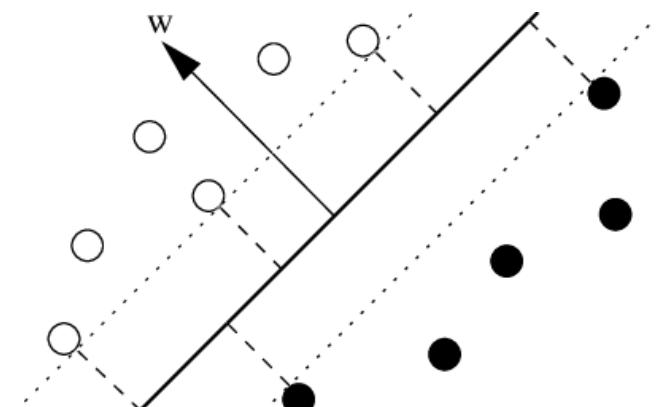


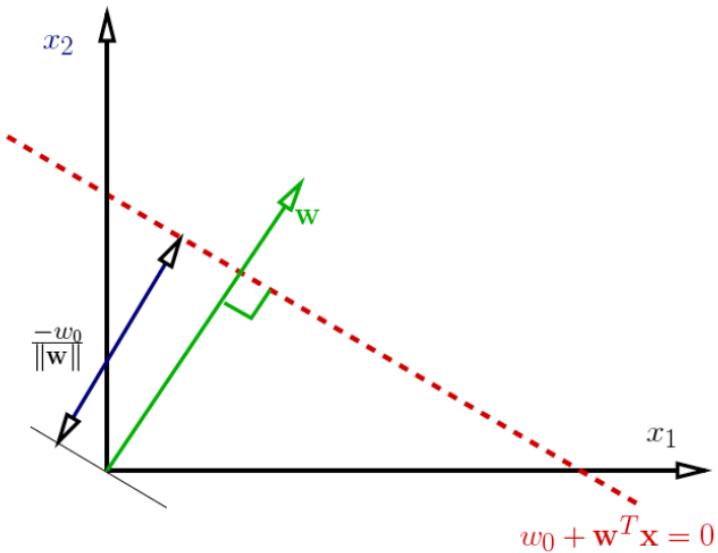
Photo: K.-R. Muller



Properties of A Linear discriminant

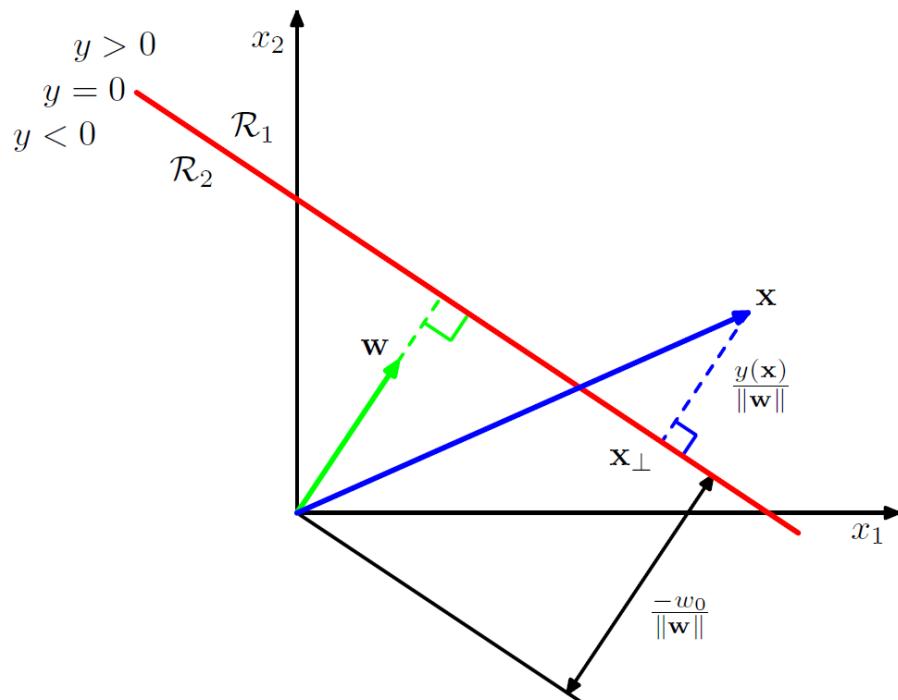
- The distance from the origin to the decision boundary is

$$\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$



Properties of A Linear discriminant

- How to compute the distance between an arbitrary point \mathbf{x} to the decision boundary $y(\mathbf{x}) = 0$?



$$\mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\begin{cases} y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \\ y(\mathbf{x}_\perp) = \mathbf{w}^T \mathbf{x}_\perp + w_0 = 0 \end{cases}$$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = \mathbf{w}^T \mathbf{x}_\perp + w_0 + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|}$$

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

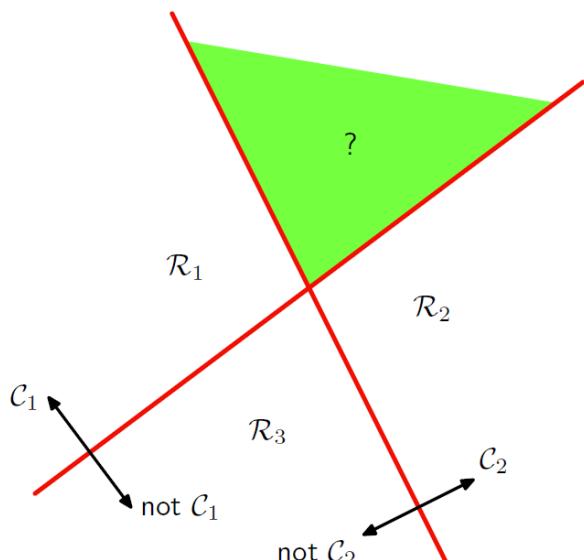


Linear discriminant for multi-class classification

- Consider the extension of linear discriminants to $K > 2$ classes
- Many classifiers cannot directly extend to multi-class classification
 - Build a K -class discriminant by combining a number of two-class discriminant functions
 - One-versus-the-rest strategy
 - One-versus-one strategy

One-versus-the-rest

- One-vs-the-rest
 - Learn $K - 1$ two-class classifiers (linear discriminants)
 - Classifier 1 is derived to separate data of class 1 from the rest
 - Classifier 2 is derived to separate data of class 1 from the rest
 - ...
 - Classifier $K - 1$ is derived to separate data of class $K - 1$ from the rest



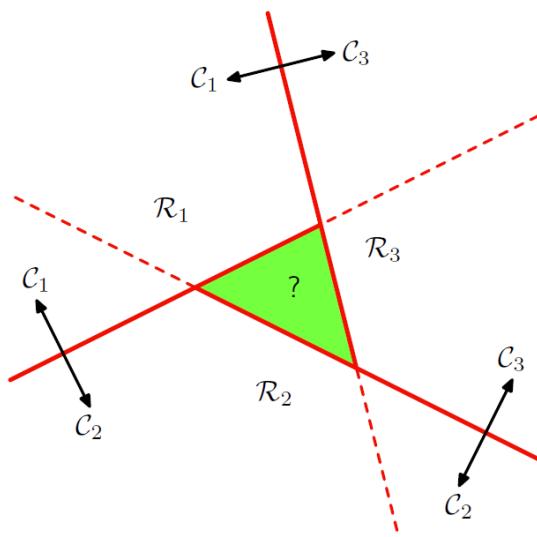
3-class classification

2 one-vs-the-rest linear
discriminants



One-vs-one

- One-vs-one
 - Learn $K(K - 1)/2$ two-class classifiers, one for each class pair
 - For classes i and j , a binary classifier is learned to separate data of class i to those of class j
 - Classification is done by majority vote
- The problem of ambiguous regions



3-class classification

3 one-vs-one linear
discriminants



K-class discriminant

- A single K -class discriminant can avoid the problem of ambiguous regions
 - It is composed of K linear functions of the form

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

for $k = 1, 2, \dots, K$

- It assigns a point \mathbf{x} to class k if

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

- The decision boundary between class k and class j is

$$y_k(\mathbf{x}) = y_j(\mathbf{x})$$

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

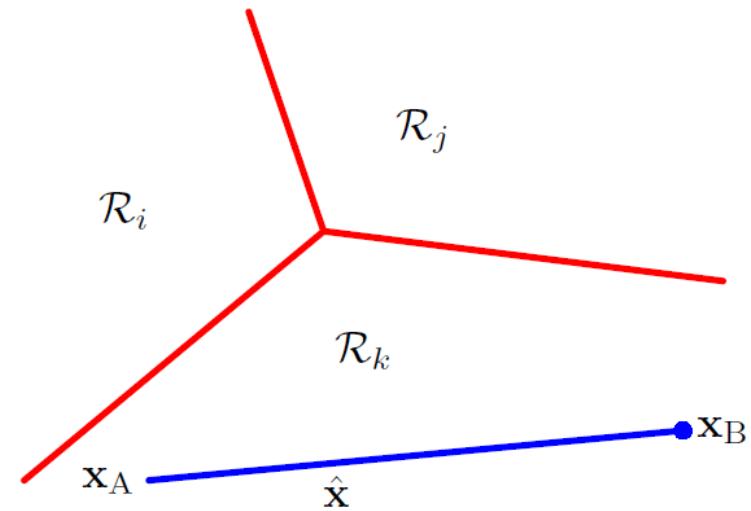


K-class discriminant

- An example: 3-class discriminant
- The decision regions of such a discriminant are **convex**
- Consider two points \mathbf{x}_A and \mathbf{x}_B , which lie inside region \mathcal{R}_k
- For any point $\hat{\mathbf{x}}$ that lies on the line connecting \mathbf{x}_A and \mathbf{x}_B , it can be expressed as

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

- It can be proved that $\hat{\mathbf{x}}$ also lies inside \mathcal{R}_k
 - Linear function of class k : $y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B)$
 - Proof: $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A), y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$, for all $j \neq k$
 $\Rightarrow y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$



Linear discriminant learning

- Learning focuses on estimating a **good** decision boundary
- We need to optimize parameters w and w_0 of the boundary
- What does good mean here?
- Is this boundary good

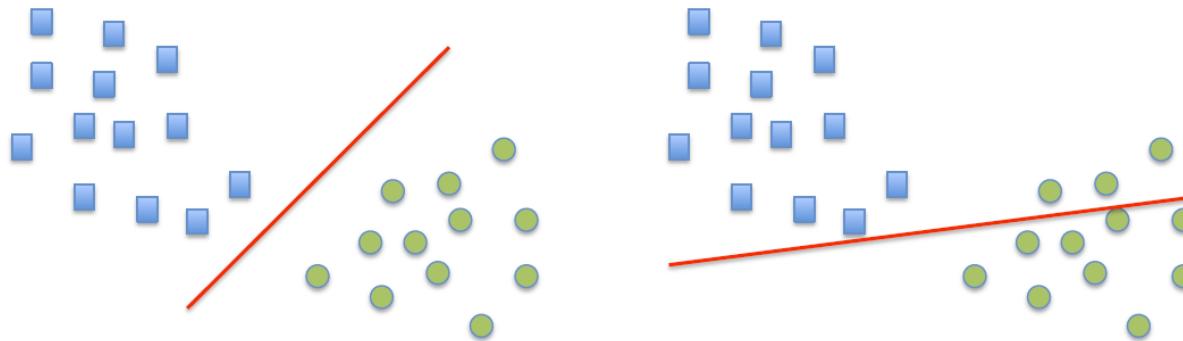


Photo: R. Zemel et al.

- We need a criterion to tell how to optimize these parameters

Least squares for classification

- Use least squares technique to solve a K -class discriminant
- Each class k is described by its own linear model

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \text{ where } k = 1, \dots, K$$

- A point \mathbf{x} is assigned to class k if

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

Least squares for classification

- Some notations
 - A data point \mathbf{x} : $\tilde{\mathbf{x}} = (1, \mathbf{x}^T)^T$
 - 1-of- K binary coding for the label vector of \mathbf{x} : $\mathbf{t} = [0, 1, 0, 0, 0]^T$
 - The linear model for class k : $\tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^T)^T$
 - Apply the linear model for class k to a point \mathbf{x} : $y_k = \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}$
- All data points: $\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_1^T \\ \vdots \\ \tilde{\mathbf{x}}_N^T \end{bmatrix}$ All data label vectors: $\tilde{\mathbf{T}} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}$
- All linear models: $\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \dots, \tilde{\mathbf{w}}_K]$
- Apply all linear models to a point \mathbf{x} : $\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$

Least squares for classification

- The squared difference between t and $y(x) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$
- Sum-of-squares error

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2} \text{Tr} \left\{ (\tilde{\mathbf{X}} \tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}} \tilde{\mathbf{W}} - \mathbf{T}) \right\}$$

- Proof sketch
 - $\text{Tr}(AB) = \text{Tr}(BA)$
 - $\text{Tr}(BA)$ is the sum of the diagonal elements of square matrix BA
 - The n th diagonal element is the squared error of point \mathbf{x}_n
- Setting the derivative w.r.t. $\tilde{\mathbf{W}}$ to 0, we obtain

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

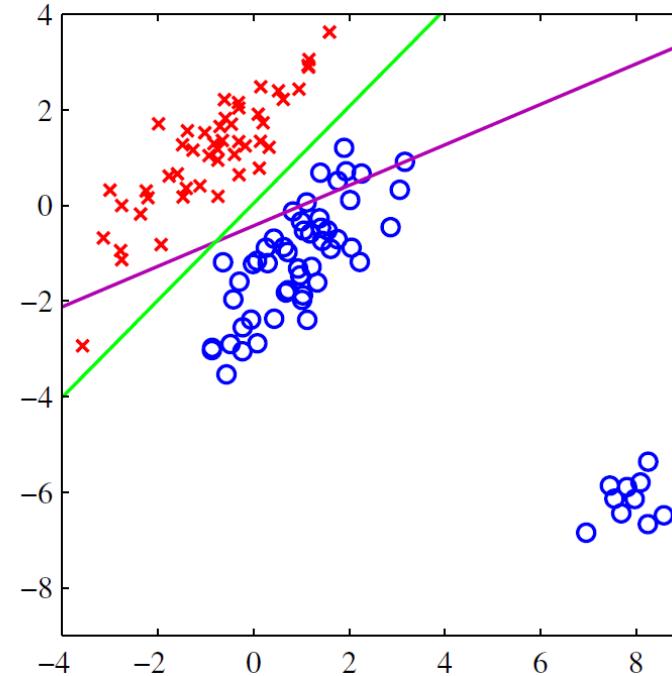
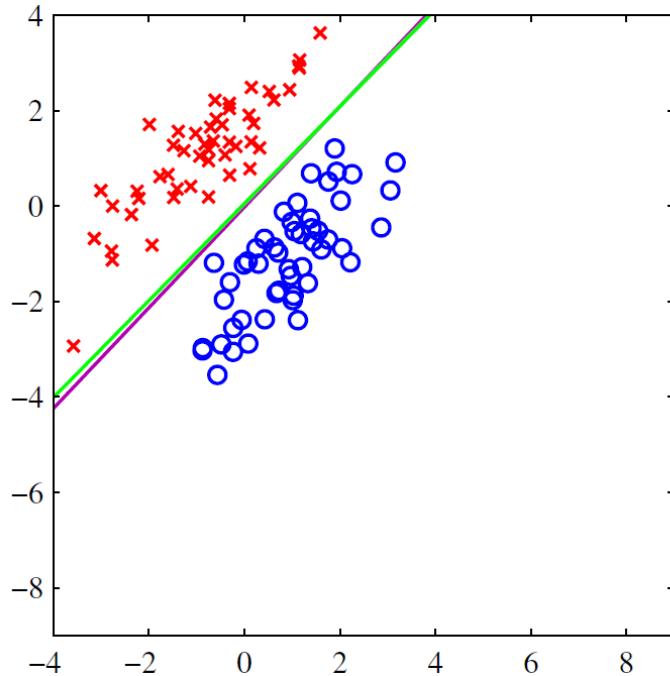
Least squares for classification

- After getting $\tilde{\mathbf{W}}$, we classify a new data point via

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = \mathbf{T}^T (\tilde{\mathbf{X}}^\dagger)^T \tilde{\mathbf{x}}$$

Least squares for classification

- The least-squares solutions are sensitive to outliers

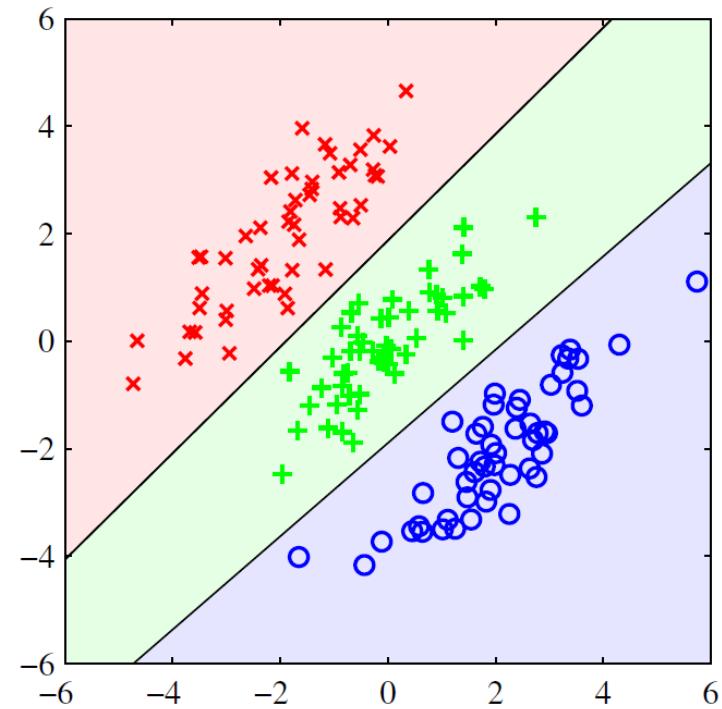
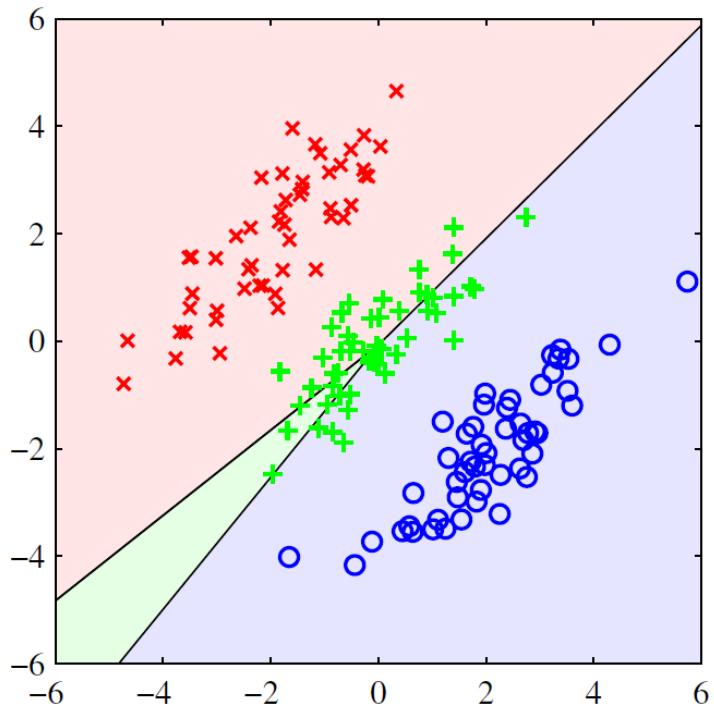


Magenta: least squares
Green: logistic regression



Least squares for classification

- The least squares method sometimes gives poor results



Left: least squares

Right: logistic regression



Fisher's linear discriminant: 2 classes

- Fisher's linear discriminant (FLD): a non-probabilistic method for dimensionality reduction
- Consider the case of two classes, and suppose we take a D -dimensional input vector \mathbf{x} and project it onto one dimension by

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

- If we place a threshold on y , and classify it as class C_1 if $y \geq -w_0$, and otherwise class C_2 , we get the linear classifier discussed previously
- In general, dimensionality reduction leads to information loss, but we can select a projection maximizing data separation

Linear discriminant with maximum separation

- A two-class problem where there are N_1 points of class C_1 and N_2 points of class C_2
- The mean vectors of the two classes are

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathbf{x}_n \quad \text{and} \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathbf{x}_n$$

- An intuitive choice of \mathbf{w} that maximizes the distance between the projected mean vectors, i.e.,

$$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$$

$$\text{where } m_k = \mathbf{w}^T \mathbf{m}_k$$

- However, the distance can be arbitrarily large by increasing the magnitude of \mathbf{w}

Linear discriminant with maximum separation

- We constrain \mathbf{w} to have unit length, i.e., $\sum_i w_i^2 = 1$
- The constrained optimization problem:

$$\begin{aligned} & \text{Maximize } \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1), \\ & \text{subject to } \mathbf{w}^T \mathbf{w} = 1. \end{aligned}$$

- The optimal \mathbf{w} in the optimization problem above

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

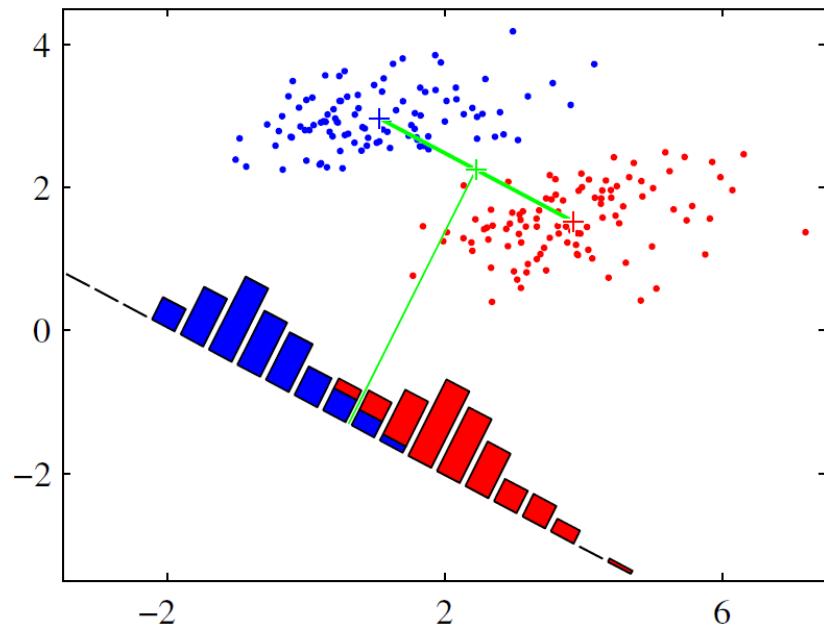
- How to prove?
 - Use **Lagrange multiplier** to solve it
 - By setting the gradient of Lagrange function w.r.t. optimization variables to 0, we get

$$(\mathbf{m}_2 - \mathbf{m}_1) + 2\lambda \mathbf{w} = 0$$



Linear discriminant with maximum separation

- Is the obtained $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$ good?



- +: \mathbf{m}_1 , +: \mathbf{m}_2 , +: threshold
- Histograms of the two classes overlap
- Right plot: The projection learned by FLD



Fisher's linear discriminant: 2 classes

- FLD seeks the projection \mathbf{w} that gives **a large distance between the projected data means** while giving **a small variance within each class**
- Maximize the between-class variance

$$(m_2 - m_1)^2 \text{ where } m_k = \mathbf{w}^T \mathbf{m}_k$$

- Minimize the within-class variance

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \text{ where } y_n = \mathbf{w}^T \mathbf{x}_n$$

- The objective (Fisher criterion):

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



Fisher's linear discriminant: 2 classes

- The objective (Fisher criterion):

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

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

where \mathbf{S}_B is the **between-class covariance matrix**

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

\mathbf{S}_W is the **within-class covariance matrix**

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



Fisher's linear discriminant: 2 classes

- Differentiate Fisher criterion w.r.t. \mathbf{w}

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

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}) = 0$$

$$\frac{2\mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} + \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{(\mathbf{w}^T \mathbf{S}_W \mathbf{w})^2} (-2\mathbf{S}_W \mathbf{w}) = 0$$

- We find that

$$\underbrace{(\mathbf{w}^T \mathbf{S}_B \mathbf{w})}_{\text{scalar}} \mathbf{S}_W \mathbf{w} = \underbrace{(\mathbf{w}^T \mathbf{S}_W \mathbf{w})}_{\text{scalar}} \mathbf{S}_B \mathbf{w}$$

- As $\mathbf{S}_B \mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}$, $\mathbf{S}_B \mathbf{w}$ is in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$
- We have

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

Fisher's linear discriminant: 2 classes

- The optimized \mathbf{w} is called Fisher's linear discriminant
- Project training data into a one-dimensional space via \mathbf{w}
 - Classification can be carried out by several methods
 - Determine a threshold y_0
 - ◆ Predict a point \mathbf{x} as C_1 if $y(\mathbf{x}) \geq -y_0$, and otherwise class C_2
 - Use the nearest-neighbor rule
 - ◆ Project all training data into the one-dimensional space via \mathbf{w}
 - ◆ Project a testing point \mathbf{x} to the same space
 - ◆ Retrieve the nearest training sample of \mathbf{x} in the projected space
 - ◆ Predict \mathbf{x} as the class that the retrieved sample belongs to

Fisher's linear discriminant: Multiple classes

- Fisher's linear discriminant (FLD) for $K > 2$ classes
- Assume the dimension of the input space is D , which is greater than K
- FLD introduces $D' \geq 1$ linear weight vectors $y_k = \mathbf{w}_k^T \mathbf{x}$ for $k = 1, \dots, D'$
- Gathering the weight vectors together projects each data point \mathbf{x} to a D' -dimensional space

$$\mathbf{y} = \mathbf{W}^T \mathbf{x}$$

where weight vectors $\{\mathbf{w}_k\}$ are the columns of \mathbf{W}

Fisher's linear discriminant: Multiple classes

- Generalize the within-class covariance matrix to K classes
- Recall the within-class covariance matrix when $K = 2$

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

- The **within-class covariance matrix** when $K \geq 2$

$$\mathbf{S}_W = \sum_{k=1}^K \mathbf{S}_k$$

where

$$\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \quad \text{and} \quad \mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

Fisher's linear discriminant: Multiple classes

- Recall the between-class covariance matrix when $K = 2$

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

- The extended between-class covariance matrix for $K > 2$

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

where

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

Fisher's linear discriminant: Multiple classes

- Consider the case where FLD projects data to a one-dimensional space, i.e., $D' = 1$

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

- An equivalent objective

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\frac{1}{2} \mathbf{w}^T S_B \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

- Lagrangian function

$$\mathcal{L}_P = -\frac{1}{2} \mathbf{w}^T S_B \mathbf{w} + \frac{1}{2} \lambda (\mathbf{w}^T S_W \mathbf{w} - 1)$$

- We have $S_B \mathbf{w} = \lambda S_W \mathbf{w} \Rightarrow S_W^{-1} S_B \mathbf{w} = \lambda \mathbf{w}$
- The optimal \mathbf{w} is the eigenvector of $S_W^{-1} S_B$ that corresponds to the largest eigenvalue



Fisher's linear discriminant: Multiple classes

- Consider the case where FLD projects data to a multi-dimensional space, i.e., $D' > 1$
- Can we directly extend the objective to learn a multi-dimensional projection? No

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \quad \longrightarrow \quad J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- A choice for the objective is

$$J(\mathbf{w}) = \text{Tr} \left\{ (\mathbf{W} \mathbf{S}_W \mathbf{W}^T)^{-1} (\mathbf{W} \mathbf{S}_B \mathbf{W}^T) \right\}$$

- The columns of the optimal \mathbf{W} are the eigenvectors of $\mathbf{S}_W^{-1} \mathbf{S}_B$ that correspond to the D' largest eigenvalues

Fisher's linear discriminant: Multiple classes

- About the value D' , the dimension of the projected space
- Note that the rank of the between-class covariance matrix is at most $K - 1$

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

- In other words, the dimension of the projected space by FLD is at most $K - 1$

Perceptron algorithm

- Perceptron by Rosenblatt is an important two-class classification algorithm in pattern recognition
- Each data point \mathbf{x} is nonlinearly transformed to $\phi(\mathbf{x})$
- Perceptron is a generalized linear model of the form

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

where f is a nonlinear activation function

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

- In training data, we use the target values $t = +1$ for class C_1 and $t = -1$ for class C_2

Perceptron algorithm

- We have $\mathbf{w}^T \phi(\mathbf{x}_n) t_n > 0$ if data point \mathbf{x}_n is correctly classified
- Goal of the perceptron algorithm is to find \mathbf{w} such that
 $\mathbf{w}^T \phi(\mathbf{x}_n) t_n > 0$, for $n = 1, \dots, N$
- The criterion of perceptron is

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi_n t_n$$

where \mathcal{M} is the set of training data that are misclassified



Perceptron algorithm

- Stochastic gradient descent is used for optimization
- At timestamp τ , the weight vector is $\mathbf{w}^{(\tau)}$ and a newly given data point \mathbf{x}_n is misclassified
- The weight vector is updated by

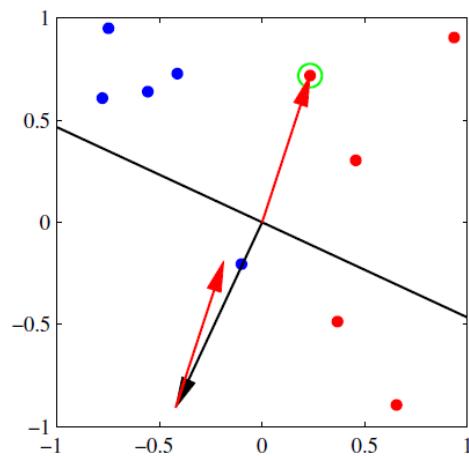
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

where η is the learning rate

- Suppose that the value of η is set to 1
 - If \mathbf{x}_n is correctly classified, the weight vector remains unchanged
 - If \mathbf{x}_n belongs to C_1 and is misclassified, ϕ_n is added to \mathbf{w}
 - If \mathbf{x}_n belongs to C_2 and is misclassified, ϕ_n is subtracted from \mathbf{w}

Perceptron algorithm

- The algorithm **converges** in a finite number of steps if the data are linearly separable



Probabilistic generative models: Two-class case

- In a **generative model**, we model the **class-conditional densities** $p(\mathbf{x}|\mathcal{C}_k)$ and **class priors** $p(\mathcal{C}_k)$, and then use them to compute posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ via Bayes' theorem
- Consider two-class cases. The **posterior probability** for class \mathcal{C}_1 is defined by

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

where $\sigma(a)$ is the **logistic sigmoid function**

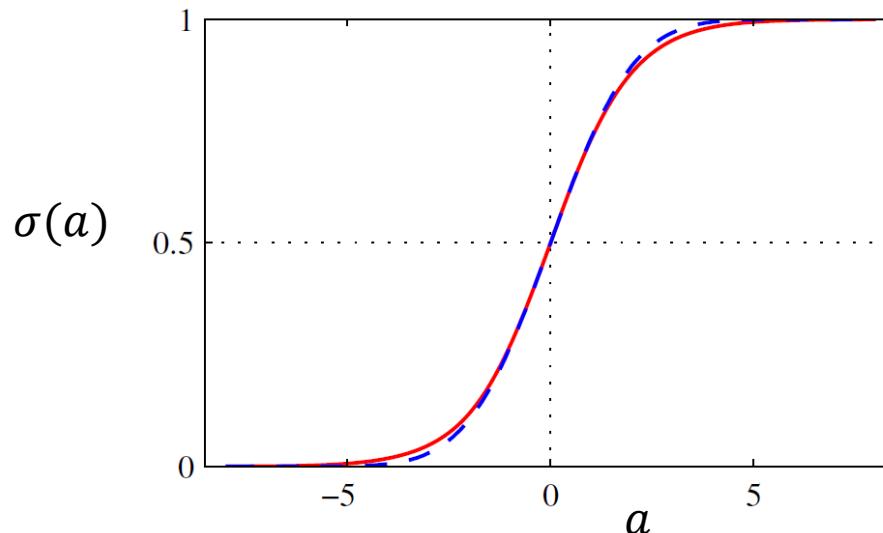
$$\sigma(a) = \frac{1}{1 + \exp(-a)} \qquad a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

Logistic sigmoid function

- Logistic sigmoid function maps the whole real axis into [0,1]
 - Symmetric property: $\sigma(-a) = 1 - \sigma(a)$
- The variable a here represents the log of the ratio of probabilities

$$\ln [p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})]$$

- Logistic sigmoid function (red curve)



Probabilistic generative models: Multi-class case

- For the case of $K > 2$ classes, the posterior probability for class \mathcal{C}_k is defined by

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

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

- Multi-class generalization of logistic sigmoid function, or **softmax function**
 - If $a_k \gg a_j$ for all $j \neq k$, we have $p(\mathcal{C}_k | \mathbf{x}) \simeq 1$ and $p(\mathcal{C}_j | \mathbf{x}) \simeq 0$

Continuous inputs: Two-class case

- Assume that the class-conditional densities are Gaussian and all classes share the same covariance matrix:

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

- Recall $p(\mathcal{C}_1|\mathbf{x}) = \sigma(a)$ where $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$

- We have $p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$

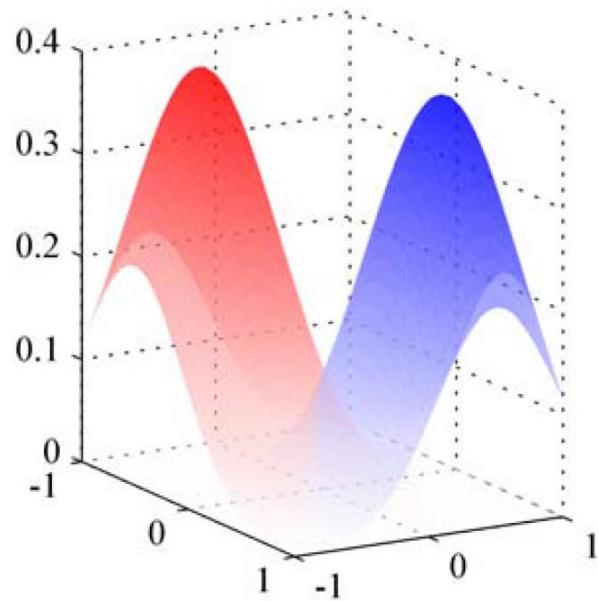
where $\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$

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

- Note that the quadratic terms in \mathbf{x} are canceled



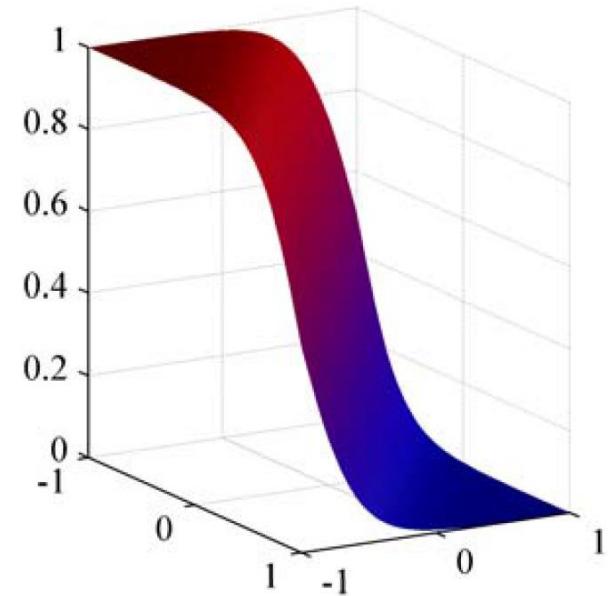
Class-conditional and posterior probabilities



class-conditional densities

$p(\mathbf{x}|\mathcal{C}_1)$: red

$p(\mathbf{x}|\mathcal{C}_2)$: blue



posterior probability

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

Note that $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$

Continuous inputs: Multi-class case

- Assume that the class-conditional densities are Gaussian and all classes share the same covariance matrix:

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

- Recall $p(\mathcal{C}_k|\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$ where $a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$
- We have $a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$

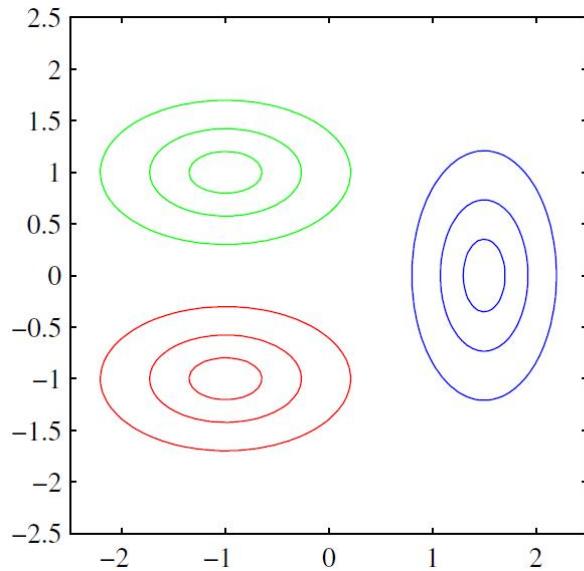
where $\mathbf{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k$

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

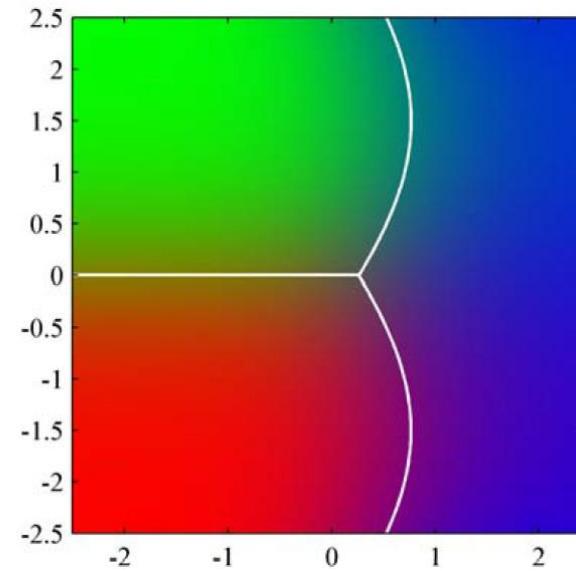


Continuous inputs: Multi-class case

- If each class-conditional density $p(\mathbf{x}|\mathcal{C}_k)$ has its own covariance matrix Σ_k , it leads to a quadratic discriminant



three class-conditional densities



posterior probability

Decision boundary between red and green classes is linear, while those between other pairs are quadratic



Determine parameter values via maximum likelihood

- We specify the functional form of **class-conditional density**

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

- We assume the **prior class probability** takes the form

$$p(\mathcal{C}_1) = \pi \quad \text{and} \quad p(\mathcal{C}_2) = 1 - \pi$$

- Suppose a set of N data points $\{\mathbf{x}_n, t_n\}$ is provided, where $t_n = 1$ denotes class C_1 and $t_n = 0$ denotes class C_2
- Our goal is to determine the values of parameters $\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}$ to complete classification

Determine parameter values via maximum likelihood

- For a data point \mathbf{x}_n from class C_1 , we have

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

- Similarly for class C_2 , we have

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi) \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- Suppose data are i.i.d. The **likelihood function** is given by

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi) \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

where $\mathbf{t} = (t_1, \dots, t_N)^T$

Maximum likelihood solution for π

- It is convenient to maximize the log of the likelihood function
- Consider first the maximization w.r.t. π
- The terms in the log likelihood function that depend on π are

$$\sum_{n=1}^N \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\}$$

- Setting the derivative w.r.t. π to zero, we obtain

$$\pi = \frac{1}{N} \sum_{n=1}^N t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

where N_1 is the number of data in class C_1 and N_2 is the number of data in class C_2

- The ML estimate for π is simply the fraction of points in class C_1

Maximum likelihood solution for μ_1

- Consider the maximization w.r.t. μ_1
- We pick those terms that depend on μ_1 in the log likelihood function

$$\sum_{n=1}^N t_n \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const}$$

- Setting the derivative w.r.t. $\boldsymbol{\mu}_1$ to zero leads to

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n$$

- It is simply the mean of all data points belonging to class C_1

Maximum likelihood solution for μ_2

- Similarly, the corresponding result for μ_2 is

$$\mu_2 = \frac{1}{N_2} \sum_{n=1}^{N_2} (1 - t_n) \mathbf{x}_n$$

- It is simply the mean of all data points belonging to class C_2

Maximum likelihood solution for Σ

- Finally, consider the maximum likelihood solution for the shared covariance matrix Σ
- Picking out the terms in the log likelihood function that depend on Σ , we get

$$\begin{aligned} & -\frac{1}{2} \sum_{n=1}^N t_n \ln |\Sigma| - \frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \\ & - \frac{1}{2} \sum_{n=1}^N (1 - t_n) \ln |\Sigma| - \frac{1}{2} \sum_{n=1}^N (1 - t_n) (\mathbf{x}_n - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2) \\ & = -\frac{N}{2} \ln |\Sigma| - \frac{N}{2} \text{Tr} \{ \Sigma^{-1} \mathbf{S} \} \end{aligned}$$

Maximum likelihood solution for Σ

$$\text{where } \mathbf{S} = \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2$$

$$\mathbf{S}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T$$

$$\mathbf{S}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T.$$

- Setting the derivative to zero, we have $\Sigma = \mathbf{S}$
- The derivation

$$\left. \begin{aligned} \frac{\partial}{\partial \Sigma} \ln |\Sigma| &= (\Sigma^{-1})^T \\ \frac{\partial}{\partial \Sigma} \text{Tr} \Sigma^{-1} \mathbf{S} &= -(\Sigma^{-1})^T \mathbf{S}^T (\Sigma^{-1})^T \end{aligned} \right\} \Rightarrow (\Sigma^{-1})^T = (\Sigma^{-1})^T \mathbf{S}^T (\Sigma^{-1})^T$$



Generative approach summary

- Class-conditional densities

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

- Class prior probabilities

$$p(\mathcal{C}_1) = \pi \quad \text{and} \quad p(\mathcal{C}_2) = 1 - \pi$$

- Determine the parameter values of $\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}$

- Classification is carried out via

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(a) \quad \text{where} \quad a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$$

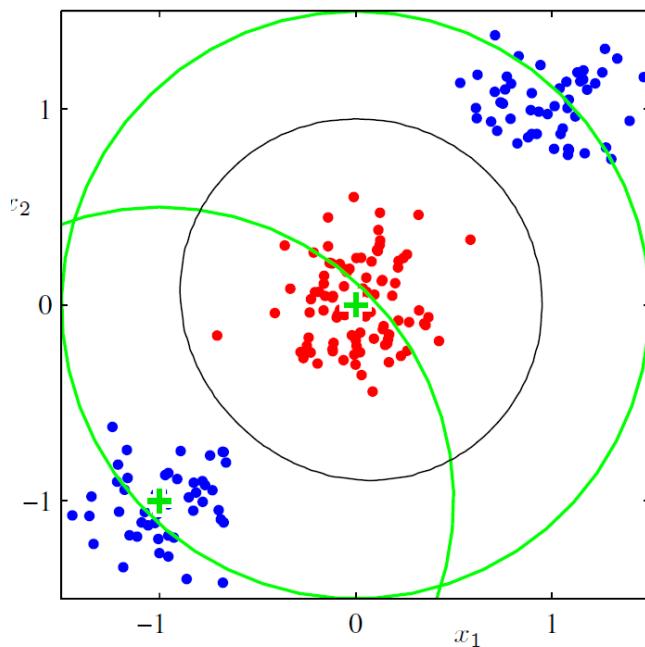
- ML solution can be directly extended to multi-class cases

Generative vs. Discriminative models

- **Probabilistic generative model:** Indirect approach that finds the parameters of class-conditional densities and class priors, and applies Bayes' theorem to get posterior probabilities
- **Probabilistic discriminative models:** Direct approach that uses the generalized linear model to represent posterior probabilities, and determines its parameters directly.
- Advantages of discriminative models:
 - Better performance in most cases, especially when the class-conditional density assumption give a poor approximation to the true distribution
 - Less parameters

Nonlinear basis functions for linear classification

- Nonlinear basis functions help when dealing with data that are not linearly separable: $\mathbf{x} \rightarrow \phi(\mathbf{x})$



data points of class C_1

data points of class C_2

+: mean of Gaussian basis function



Logistic regression for two-class classification

- Recall the posterior probability for two-class classification

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

- 1. Ignore the class-conditional probabilities and class priors
- 2. Apply basis functions for nonlinear transform
- 3. Assume the posterior probability can be written as a logistic sigmoid acting on a linear function of the feature vector
- We get

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi) \text{ and } p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$

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

Logistic regression model

- This model is called **logistic regression**, though it is used for classification
$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi)$$
- Suppose the dimension of ϕ is M . There are M parameters to learn in logistic regression
- Cf. For the generative model, we use $2M$ parameters for the means of two classes and $M(M + 1)/2$ parameters for the shared covariance matrix



Determine parameters of logistic regression

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi)$$

- The derivative of the logistic sigmoid function

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

- Derivation:

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

$$\frac{\partial \sigma}{\partial a} = \frac{1}{(1+e^{-a})^2} \cdot (e^{-a}) = \frac{1}{1+e^{-a}} \cdot \frac{e^{-a}}{1+e^{-a}} = \sigma(1 - \sigma)$$

- Given training data $\{\phi_n, t_n\}$, where $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$ for $n = 1, \dots, N$, the likelihood function of logistic regression is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

where $\mathbf{t} = (t_1, \dots, t_N)^T$ and $y_n = p(\mathcal{C}_1|\phi_n)$



Determine parameters of logistic regression

- The negative log likelihood, called cross entropy error, is

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

where $y_n = \sigma(a_n)$ and $a_n = \mathbf{w}^T \phi_n$

- The gradient of the error function w.r.t. \mathbf{w} is

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

- Derivation:

Using $\frac{\partial \sigma}{\partial a} = (1 - \sigma)\sigma$

$$\begin{aligned}\nabla E(\mathbf{w}) &= -\sum_{n=1}^N \left\{ \frac{t_n}{y_n} (y_n(1 - y_n)) \phi_n - \frac{1-t_n}{1-y_n} (1 - y_n) y_n \phi_n \right\} \\ &= -\sum_{n=1}^N \{t_n(1 - y_n) \phi_n - (1 - t_n) y_n \phi_n\}\end{aligned}$$



Determine parameters of logistic regression

- Optimize the parameters by stochastic gradient descent

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

- Optimize the parameters by iterative reweighted least squares (IRLS), i.e., Newton-Raphson iterative optimization scheme

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

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

- 1D Newton's method

$$x^{(\text{new})} = x^{(\text{old})} - g(x^{(\text{old})})/g'(x^{(\text{old})}) \quad \text{finding } g = 0$$
$$x^{(\text{new})} = x^{(\text{old})} - f'(x^{(\text{old})})/f''(x^{(\text{old})}) \quad \text{finding } f' = 0$$



Newton-Raphson iterative optimization

- Negative log likelihood function

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

- Gradient

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

- Hessian

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

where \mathbf{R} is a diagonal matrix with

$$R_{nn} = y_n(1 - y_n)$$



Newton-Raphson iterative optimization

- The Newton-Raphson update formula

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - \mathbf{H}^{-1} \nabla E(\mathbf{w}) \\ &= \mathbf{w}^{(\text{old})} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t})\end{aligned}$$

- Hessian matrix is positive definite
- Proof:

$\mathbf{R} : N \times N$ diagonal

$$\mathbf{R}_{nn} = y_n(1 - y_n)$$

$$0 < y_n < 1$$

$\Rightarrow \mathbf{v}^T \mathbf{H} \mathbf{v} > 0$ for an arbitrary \mathbf{v}

$\Rightarrow \mathbf{H}$ is positive definite



Multiclass logistic regression

- Recall two-class logistic regression

$$p(\mathcal{C}_1|\phi) = \sigma(a) = \sigma(\mathbf{w}^T \phi) \quad \text{and} \quad p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$

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

- We deal with multiclass cases

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where the activation a_k is given by

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



Likelihood function of multiclass logistic regression

- Two-class case: Given training data $\{\phi_n, t_n\}$, where $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$ for $1, \dots, N$, the likelihood function of two-class logistic regression is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

where $\mathbf{t} = (t_1, \dots, t_N)^T$ and $y_n = p(C_1|\phi_n)$

- For a multiclass case, we use 1-of-K coding $(00 \cdots 1 \cdots 0)^T$ to represent each target label vector \mathbf{t}_n , let $y_{nk} = y_k(\phi_n)$, we have the likelihood function

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



Newton-Raphson iterative optimization

- Negative log likelihood function is

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}$$

- When using Newton-Raphson iterative optimization, we need to have the following gradient and Hessian

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K)$$

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K)$$

Background

- Variable dependence: $\mathbf{w} \rightarrow a \rightarrow y \rightarrow E$
- According to

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}$$

we have

$$\frac{\partial E}{\partial y_{nk}} = -\frac{t_{nk}}{y_{nk}}$$



Background

- According to

$$p(\mathcal{C}_k | \phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

we have

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j) \quad \text{where} \quad I_{kj} = \begin{cases} 1, & j = k, \\ 0, & \text{otherwise.} \end{cases}$$

- Proof

$$\frac{\partial y_k}{\partial a_k} = \frac{e^{a_k}}{\sum_i e^{a_i}} - \left(\frac{e^{a_k}}{\sum_i e^{a_i}} \right)^2 = y_k(1 - y_k)$$

$$\frac{\partial y_k}{\partial a_j} = \frac{-e^{a_k} e^{a_j}}{(\sum_i e^{a_i})^2} = -y_k y_j \text{ for } j \neq k$$

Background

- According to

$$a_{nj} = \mathbf{w}_j^T \phi_n$$

we have

$$\nabla_{\mathbf{w}_j} a_{nj} = \phi_n$$

- Given $\frac{\partial E}{\partial y_{nk}} = -\frac{t_{nk}}{y_{nk}}$ and $\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j)$, we can compute

$$\begin{aligned}\frac{\partial E}{\partial a_{nj}} &= \sum_{k=1}^K \frac{\partial E}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nj}} = - \sum_{k=1}^K \frac{t_{nk}}{y_{nk}} y_{nk} (I_{kj} - y_{nj}) \\ &= - \sum_{k=1}^K t_{nk} (I_{kj} - y_{nj}) = -t_{nj} + \sum_{k=1}^K t_{nk} y_{nj} = y_{nj} - t_{nj}\end{aligned}$$

(1-of- K coding scheme: $\forall n, \sum_k t_{nk} = 1$)

Newton-Raphson iterative optimization

- Gradient

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N \frac{\partial E}{\partial a_{nj}} \nabla_{\mathbf{w}_j} a_{nj} = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n$$

- Hessian

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N y_{nk} (I_{kj} - y_{nj}) \phi_n \phi_n^T$$

- With gradient and Hessian, Newton-Raphson method works

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

Discriminative approach summary

- Posterior probability and logistic regression

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi) \text{ and } p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$

- The negative log likelihood (cross entropy error)

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

- Newton-Raphson method for iterative optimization

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

- It can be extended to multiclass classification

Summary

- Linear discriminant
 - Two-class discriminant
 - K-class discriminant
 - Fisher's linear discriminant
 - Perceptron algorithm
- Probabilistic generative model
 - Class-conditional probability and class prior probability
 - ML solution
- Probabilistic discriminative model: Logistic regression
 - Posterior probability
 - Newton-Raphson iterative optimization



References

- Chapters 4.1, 4.2, and 4.3 in the PRML textbook

Thank You for Your Attention!

感谢您们的聆听！

Yen-Yu Lin (林彥宇)

Email: lin@cs.nctu.edu.tw

URL: <https://www.cs.nctu.edu.tw/members/detail/lin>

