Pattern Recognition and Machine Learning

Chapter 4: Linear Models for Classification

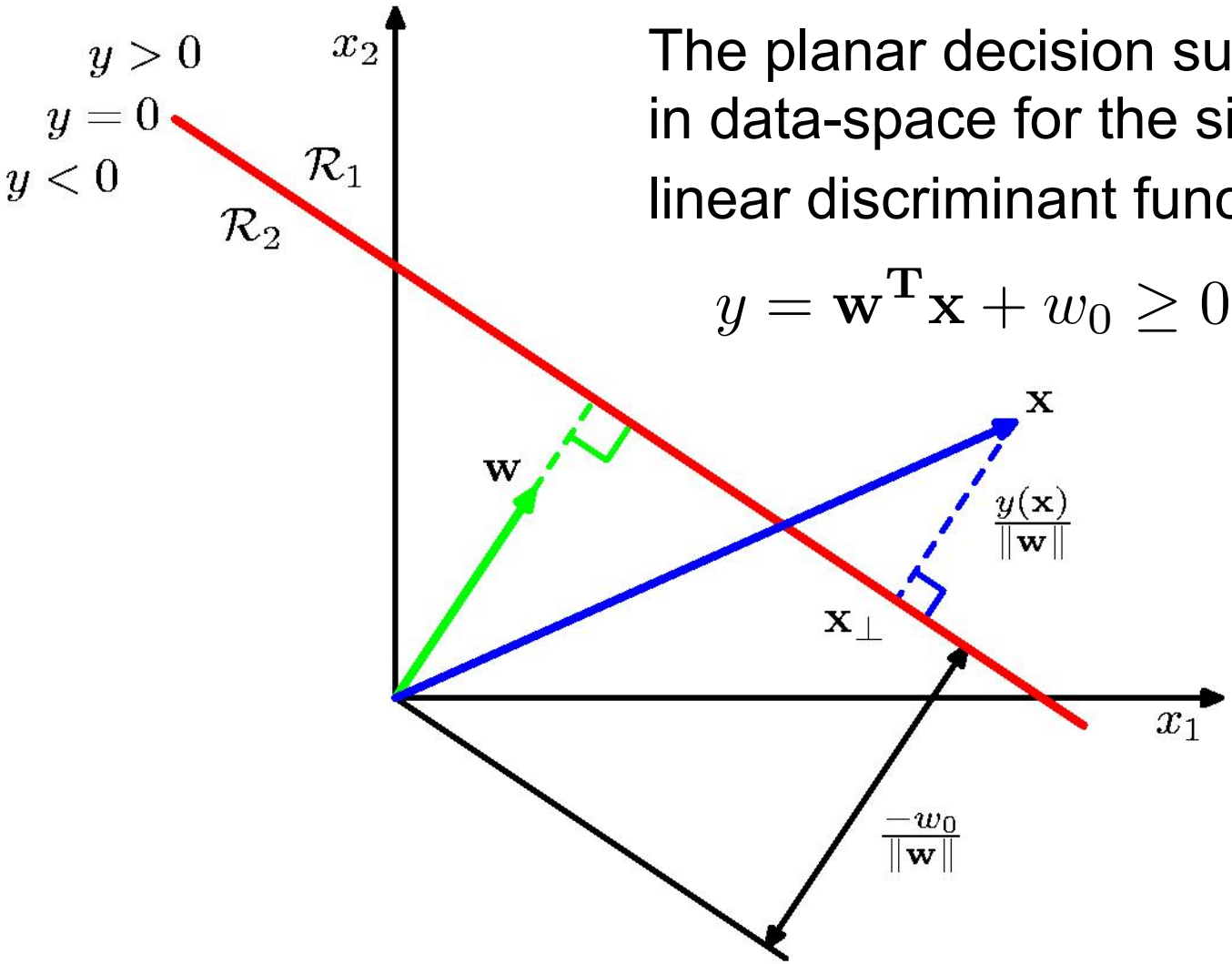
Representing the target values for classification

- If there are only two classes, we typically use a single real valued output that has target values of 1 for the "positive" class and 0 (or sometimes -1) for the other class
- For probabilistic class labels the output of the model can represent the probability the model gives to the positive class.
- If there are N classes we often use a vector of N target values containing a single 1 for the correct class and zeros elsewhere.
- For probabilistic labels we can then use a vector of class probabilities as the target vector.

Three approaches to classification

- Use discriminant functions directly without probabilities:
  - Convert the input vector into one or more real values so that a simple operation (like threshholding) can be applied to get the class.
- Infer conditional class probabilities: $p(class = C_k \mid \mathbf{x})$
  - Then make a decision that minimizes some loss function
- Compare the probability of the input under separate, class-specific, generative models.
  - E.g. fit a multivariate Gaussian to the input vectors of each class and see which Gaussian makes a test data vector most probable.

Discriminant functions



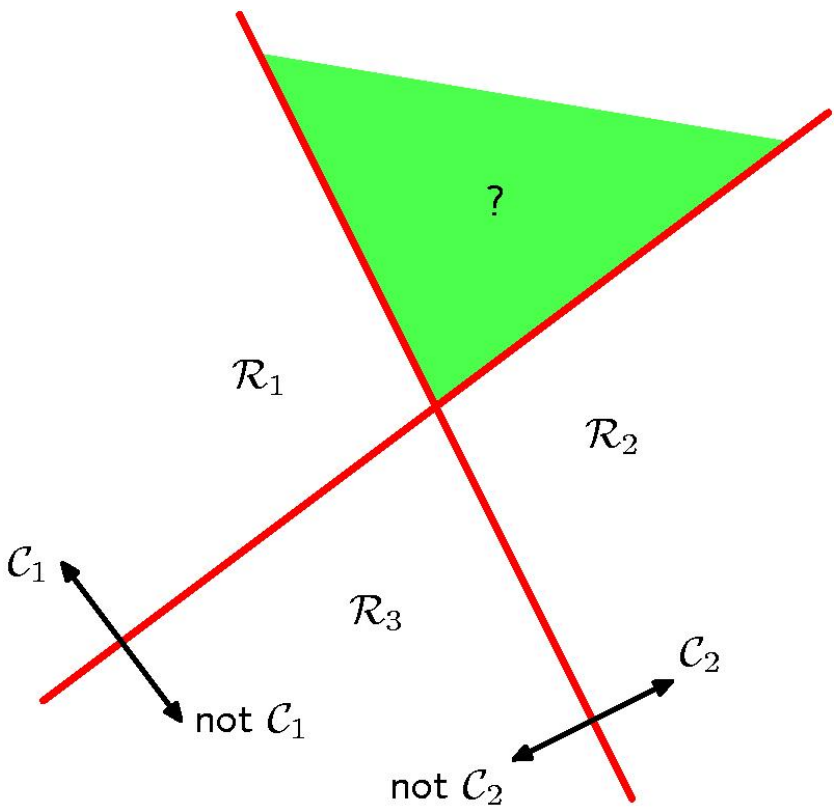
The planar decision surface in data-space for the simple linear discriminant function:

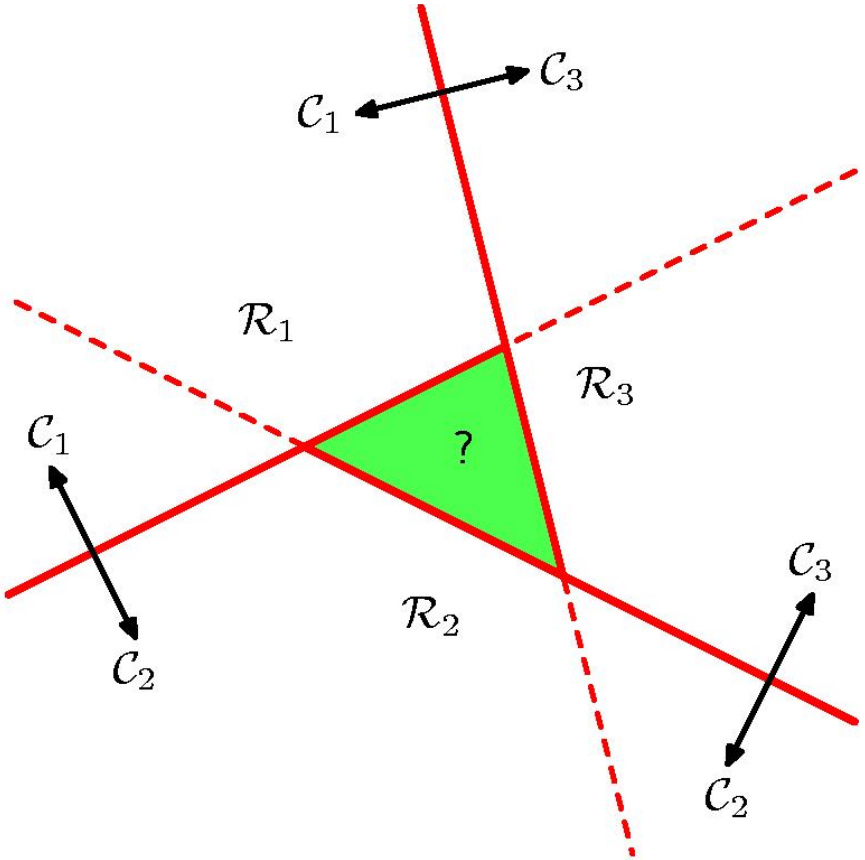
$$y = \mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0 \geq 0$$

Discriminant functions for N>2 classes

- One possibility is to use N two-way discriminant functions.
  - Each function discriminates one class from the rest.
- Another possibility is to use N(N-1)/2 two-way discriminant functions
  - Each function discriminates between two particular classes.
- Both these methods have problems

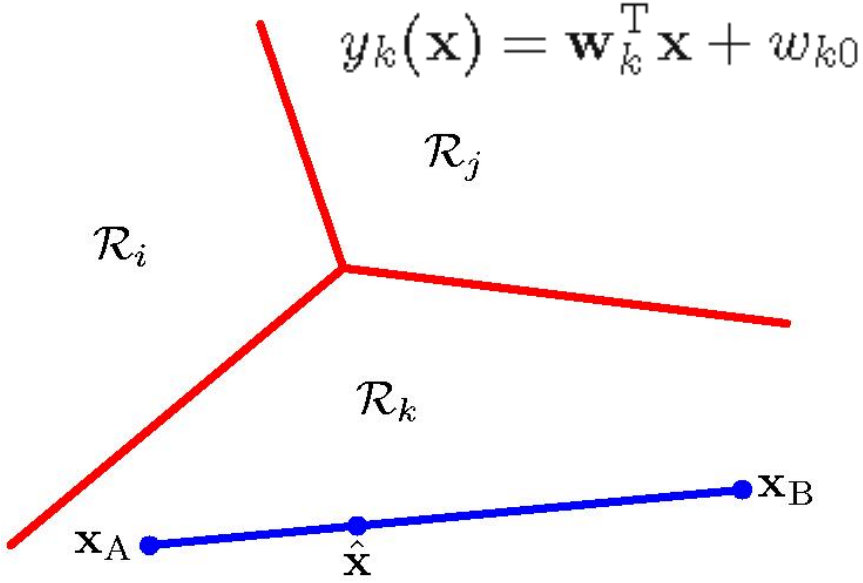
Problems with multi-class discriminant functions





A simple solution

- Use K discriminant functions, $y_i, y_j, y_k \ldots$ and pick the max.
  - This is guaranteed to give consistent and convex decision regions if y is linear.



$$y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A) \;\; and \;\; y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$$

$$implies \; (for \; positive \; \alpha) \; that$$

$$y_k(\alpha \mathbf{x}_A + (1-\alpha)\,\mathbf{x}_B) > y_j(\alpha \mathbf{x}_A + (1-\alpha)\,\mathbf{x}_B)$$

Using "least squares" for classification

This is not the right thing to do and it doesn't work as well as other methods (we will see later), but it is easy.

It reduces classification to least squares regression.

We already know how to do regression. We can just solve for the optimal weights with some matrix algebra.

Least squares for classification

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \quad \Longrightarrow \quad \mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$$

$\{\mathbf{x_n}, \mathbf{t_n}\}$

The sum-of-squares error function:

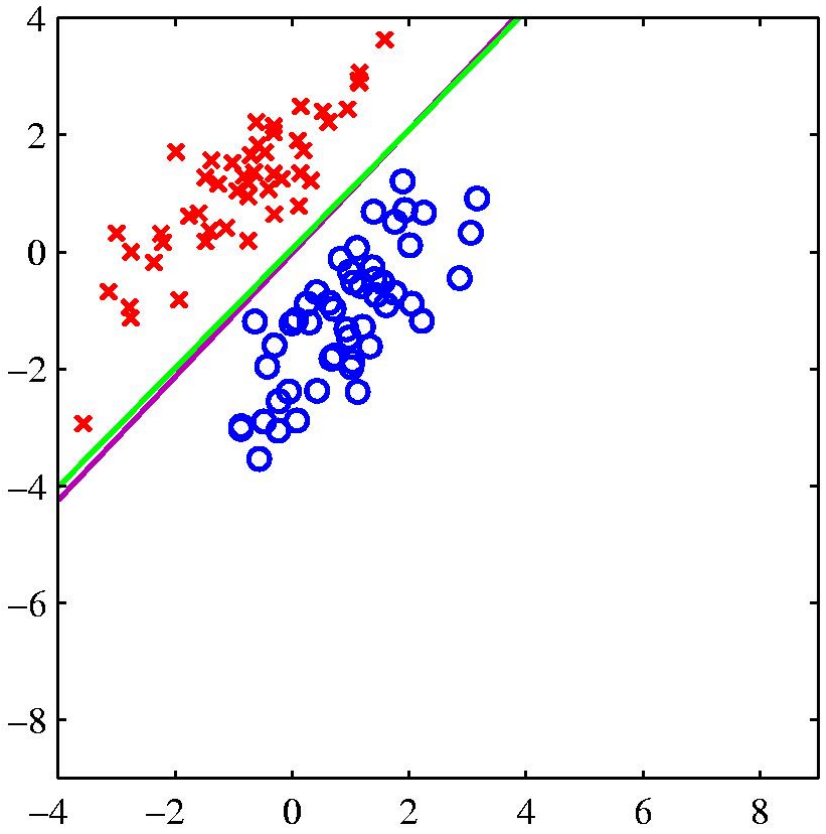
nth row of matrix T is $\mathbf{t}^{\mathrm{T}}_{\mathbf{n}}$

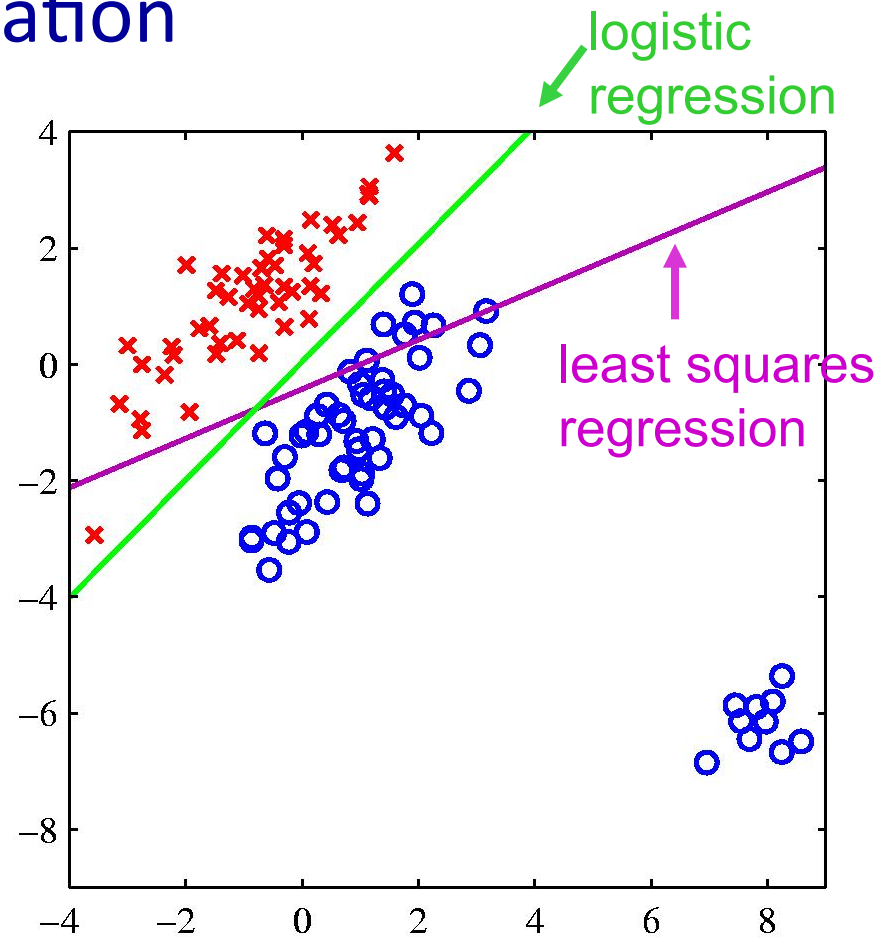
$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}.$$

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

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} = \mathbf{T}^{\mathrm{T}}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\mathrm{T}}\widetilde{\mathbf{x}}.$$

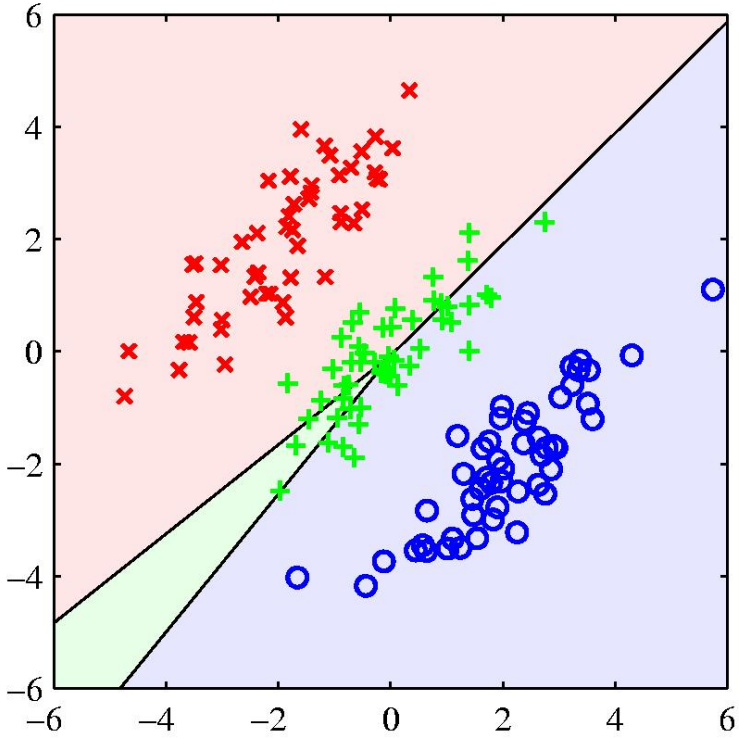
Problems with using least squares for classification

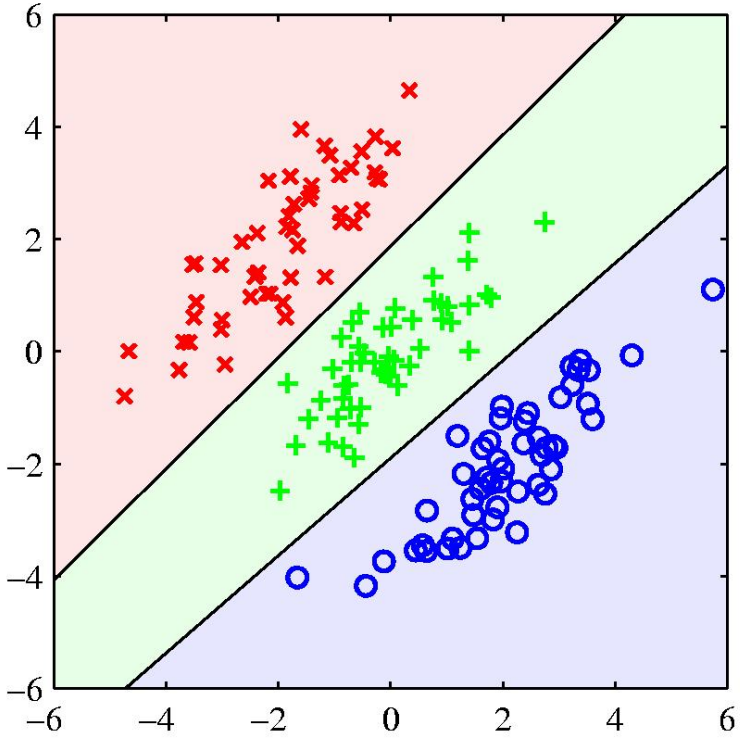




If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being "too correct"

Another example where least squares regression gives poor decision surfaces





Fisher's linear discriminant

- A simple linear discriminant function is a projection of the data down to 1-D.
- So choose the projection that gives the best separation of the classes. What do we mean by "best separation"?
- An obvious direction to choose is the direction of the line joining the class means.
- But if the main direction of variance in each class is not orthogonal to this line, this will not give good separation.
- Fisher's method chooses the direction that maximizes the ratio of between class variance to within class variance.
- This is the direction in which the projected points contain the most information about class membership (under Gaussian assumptions)

Fisher's linear discriminant

The two-class linear discriminant acts as a projection:

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

followed by a threshold $y \geq -w_0$

In which direction **w** should we project?

One which separates classes "well"

Intuition: We want the (projected) centers of the classes to be far apart, and each class (projection) to be clustered around its centre.

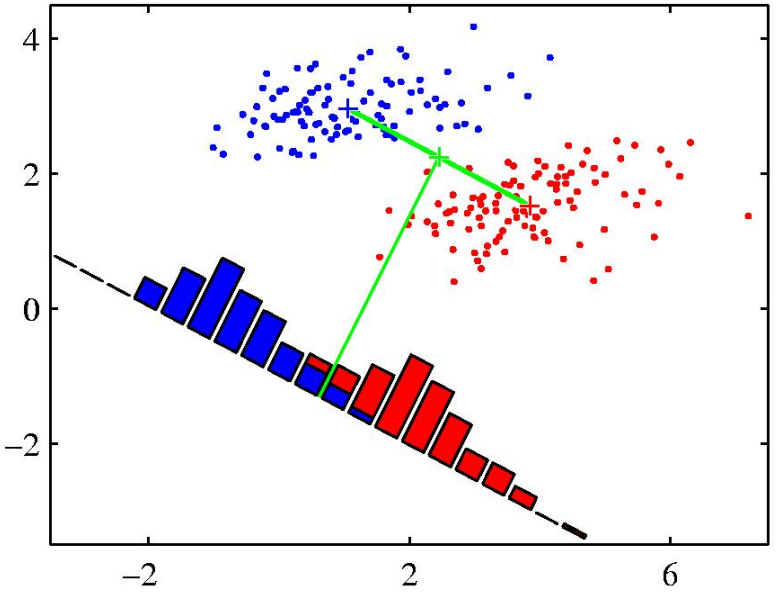
Fisher's linear discriminant

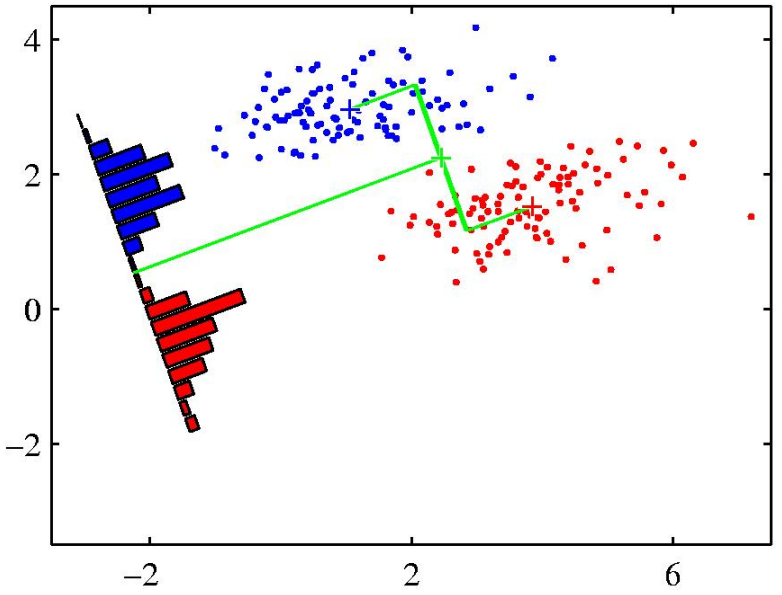
A natural idea would be to project in the direction of the line connecting class means

However, problematic if classes have variance in this direction (ie are not clustered around the mean)

Fisher criterion: maximize ratio of inter-class separation (between) to intra-class variance (inside)

A picture showing the advantage of Fisher's linear discriminant.





When projected onto the line joining the class means, the classes are not well separated.

Fisher chooses a direction that makes the projected classes much tighter, even though their projected means are less far apart.

Math of Fisher's linear discriminants

What linear transformation is best for discrimination?

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

The projection onto the vector separating the class means seems sensible.

But we also want small variance within each class:

$$s_1^2 = \sum_{n \varepsilon C_1} (y_n - m_1)$$

$$s_2^2 = \sum_{n \varepsilon C_2} (y_n - m_2)$$

Fisher's objective function is:

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

← between

← within

More math of Fisher's linear discriminants

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

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

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

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

Probabilistic Generative Models

Model the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ and class prior $p(\mathcal{C}_k)$

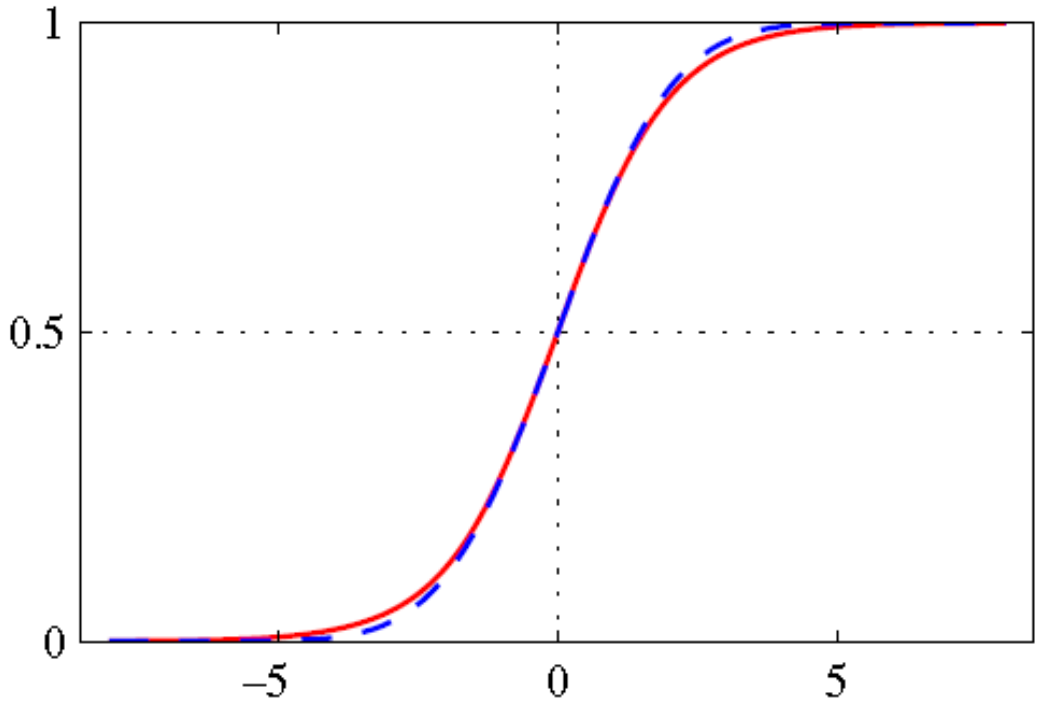
Use Bayes' theorem to compute posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$

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

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

Logistic Sigmoid Function

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



Probabilistic Generative Models (k class)

- Beyes' theorem

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

$$a_k = \ln p(\mathbf{x}|C_k)p(C_k).$$

- if $a_k \gg a_j$ *for all* $j \neq k$, *then* $p(C_k|x) \approx 1$, *and* $p(C_j|x) \approx 0$.

Continuous inputs

Model the class-conditional densities

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

Here, all classes share the same covariance matrix.

Two-classes Problem

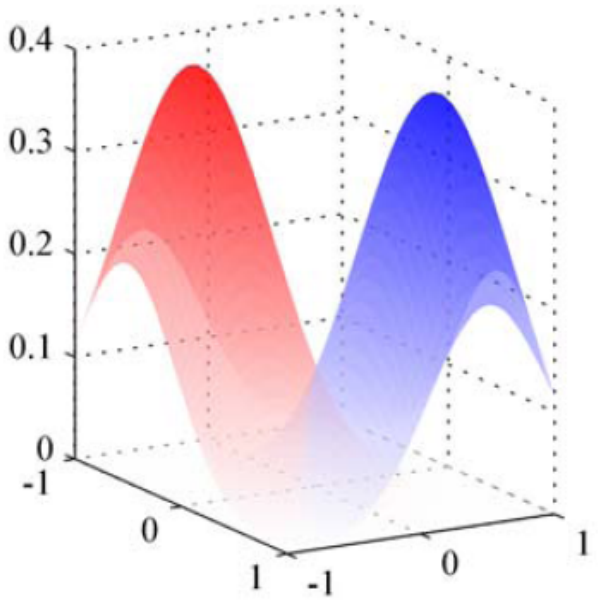
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

$$\begin{aligned} \mathbf{w} &= \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \end{aligned}$$

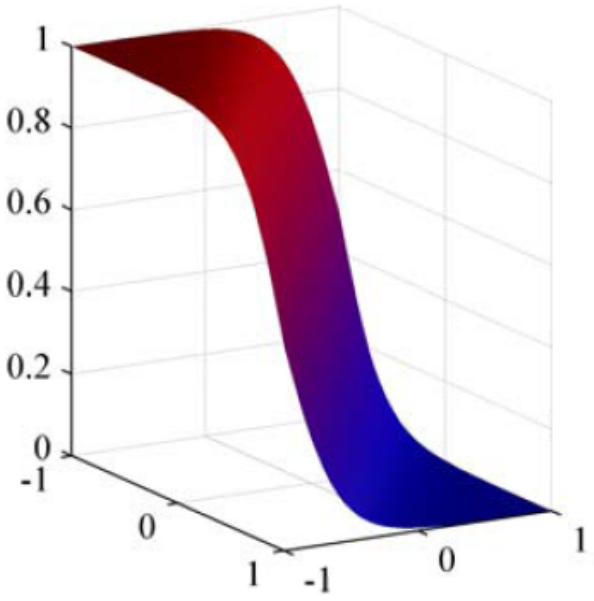
Two-Classes Problem

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

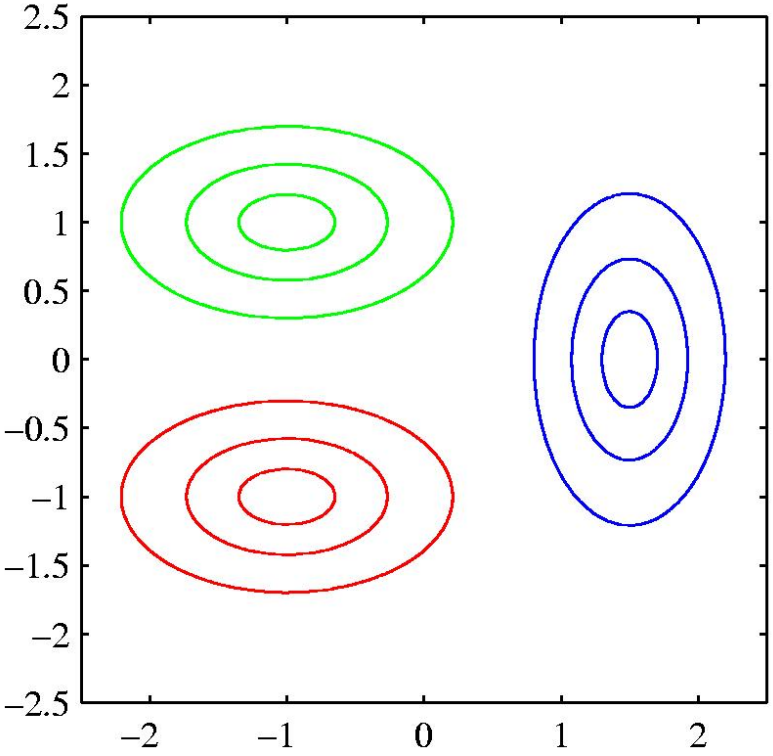
$p(\mathbf{x}|\mathcal{C}_k)$

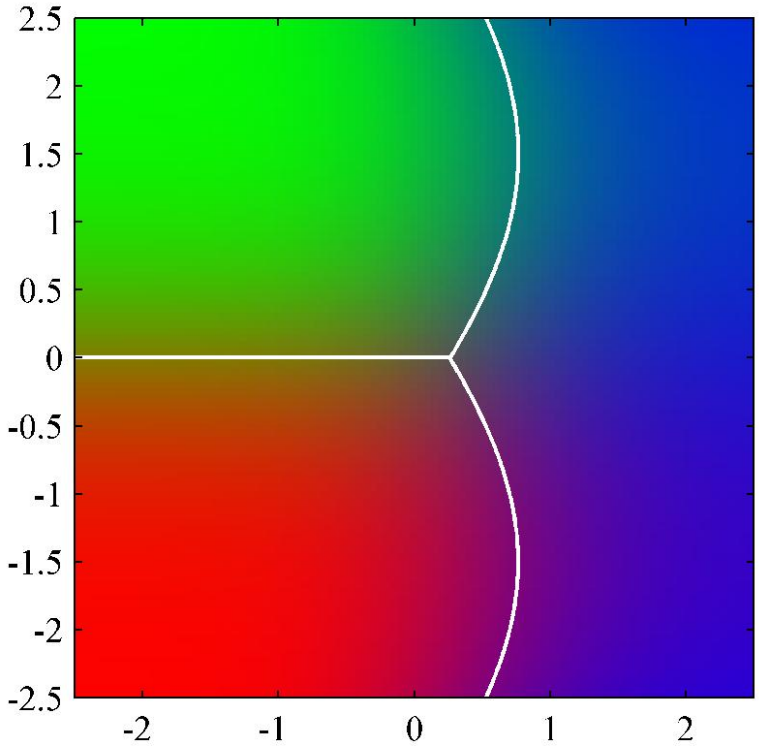


$p(\mathcal{C}_1|\mathbf{x})$



The posterior when the covariance matrices are different for different classes.





The decision surface is planar when the covariance matrices are the same and quadratic when they are not.

Maximum likelihood solution

- The likelihood function:

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

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

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n \qquad \boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1-t_n)\mathbf{x}_n$$

- Shared covariance matrix:

$$\boldsymbol{\Sigma} = \frac{N_1}{N}\frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^T + \frac{N_2}{N}\frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^T$$

Maximum likelihood solution

Note that the approach of fitting Gaussian distributions to the classes is not robust to outliers, because the maximum *likelihood estimation of a Gaussian is not robust.*

Discrete features

When $x_i \in \{0, 1\}$

The class-conditional distributions:

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

Linear functions of the input values:

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

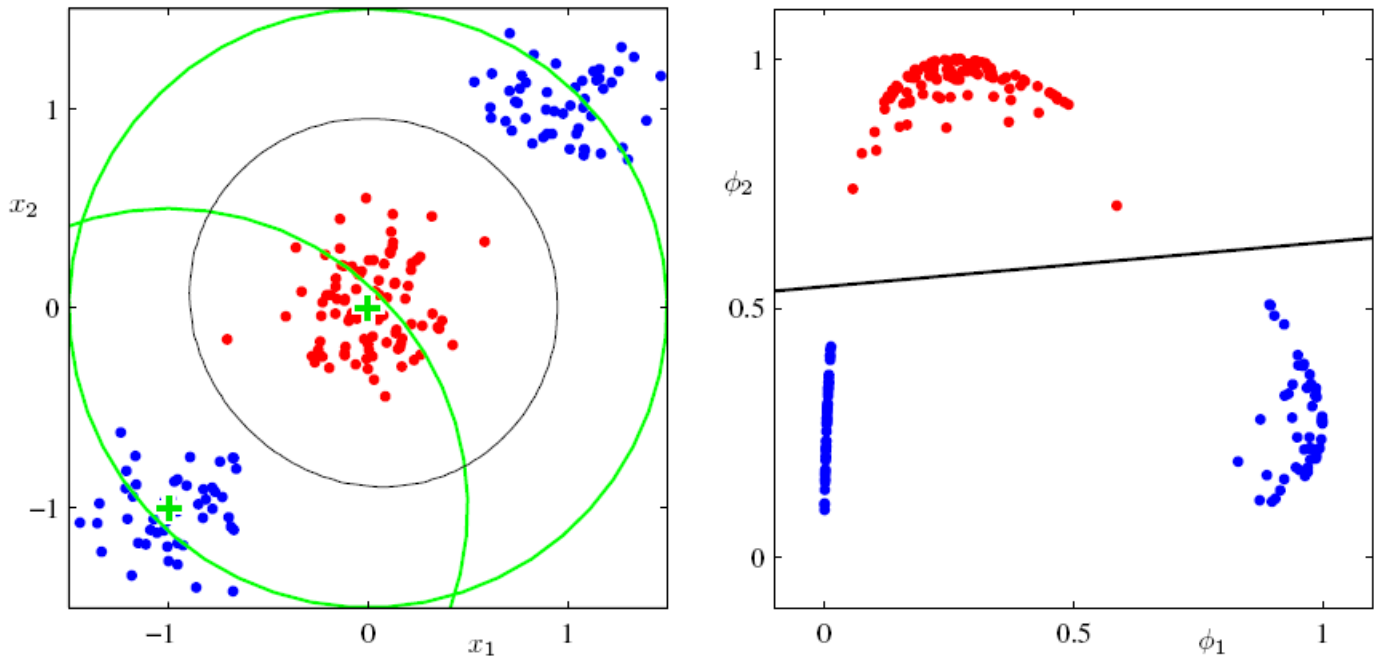
Probabilistic Discriminative Models

What is it?

It means to use the functional form of the generalized linear probabilistic model explicitly, and to determine its parameters directly $p(\mathcal{C}_k|\mathbf{x})$

The difference from Discriminant Functions, and Probabilistic Generative Models

Fixed Basis Functions



Space transformation by nonlinear function $\phi(\mathbf{x})$

Logistic Regression

- To reduce the complexity of probabilistic models, the *logistic sigmoid* $\sigma(\cdot)$ is introduced

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

  - For Gaussian class conditional densities:

    $M(M+5)/2+1$ parameters

  - Logistic Regression:

    M parameters

The Likelihood

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

$$y_n = p(C_1|\phi_n)$$

The Gradient of the Log Likelihood

- Error function:

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

- Gradient of the Error function

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

Iterative Reweighted Least Squares

- *Newton-Raphson*

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

- **H** is the Hessian Matrix

$$\mathbf{H} = \nabla\nabla E(\mathbf{w})$$

$$\mathbf{H} = \mathbf{\Phi}^{\mathbf{T}}\mathbf{R}\mathbf{\Phi}$$

- **R** is an NxN diagonal matrix with elements:

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

Iterative Reweighted Least Squares

- The Newton-Raphson update takes this form:

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t}) \\ &= (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\left\{\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t})\right\} \\ &= (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{z}\end{aligned}$$

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

Multiclass Logistic Regression

- Generative models (discussed in section 4.2)

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

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

- Using the maximum likelihood to determine the parameters directly

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j)$$

Multiclass Logistic Regression

- Maximum likelihood method

$$p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

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

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1,\ldots,\mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\,\boldsymbol{\phi}_n$$

- Using IRLS algorithm to do the w updating work