Linear Models for Classification

Sumeet Agarwal, EEL709

(Most figures from Bishop, *PRML*)

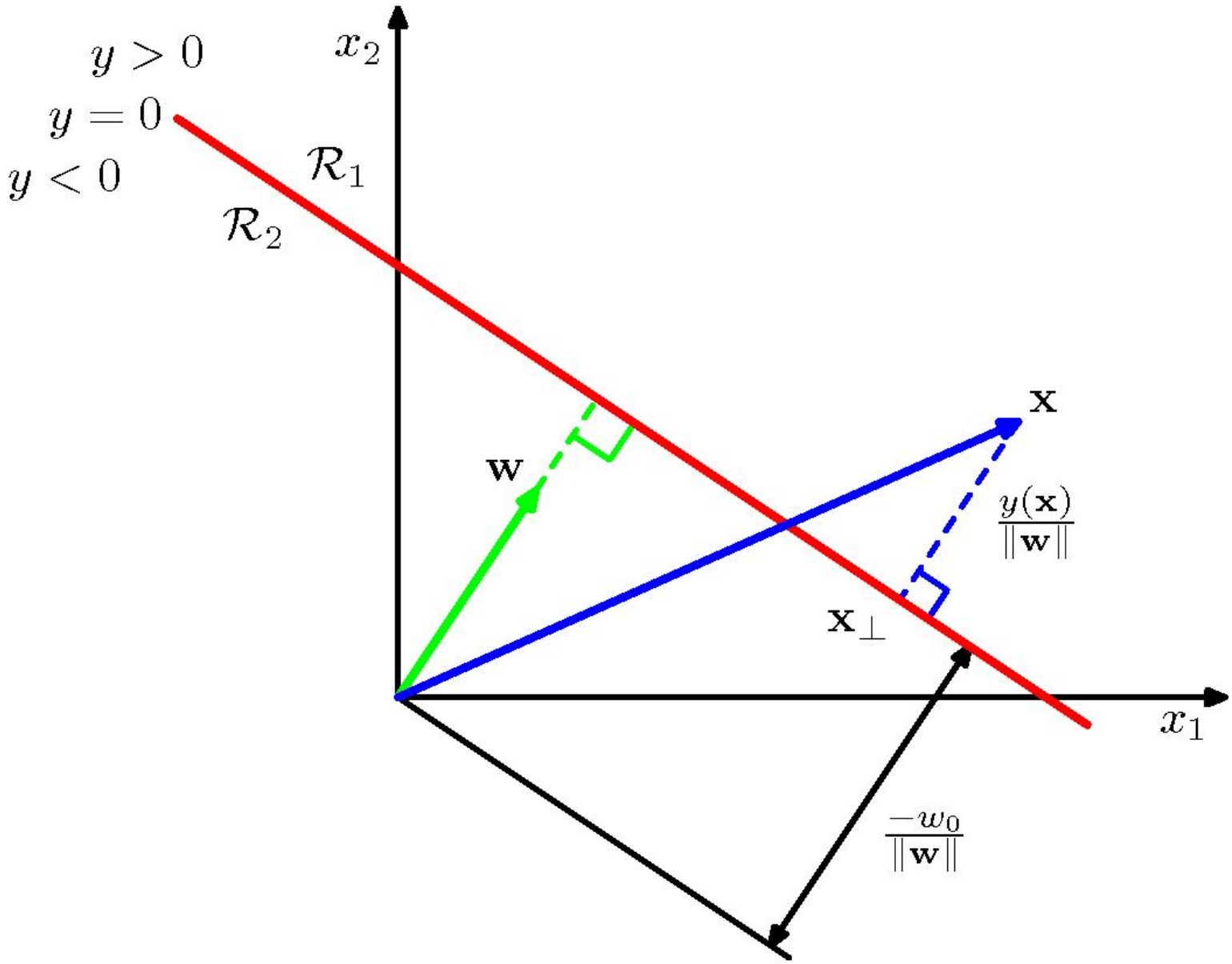
Approaches to classification

- *Discriminant function*: Directly assigns each data point **x** to a particular class C_i
- Model the conditional class distribution $p(C_i|\mathbf{x})$: allows separation of inference and decision
- *Generative approach*: model class likelihoods, $p(\mathbf{x}|C_i)$, and priors, $p(C_i)$; use Bayes' theorem to get posteriors:

$$p(C_i|\mathbf{x}) \sim p(\mathbf{x}|C_i)p(C_i)$$

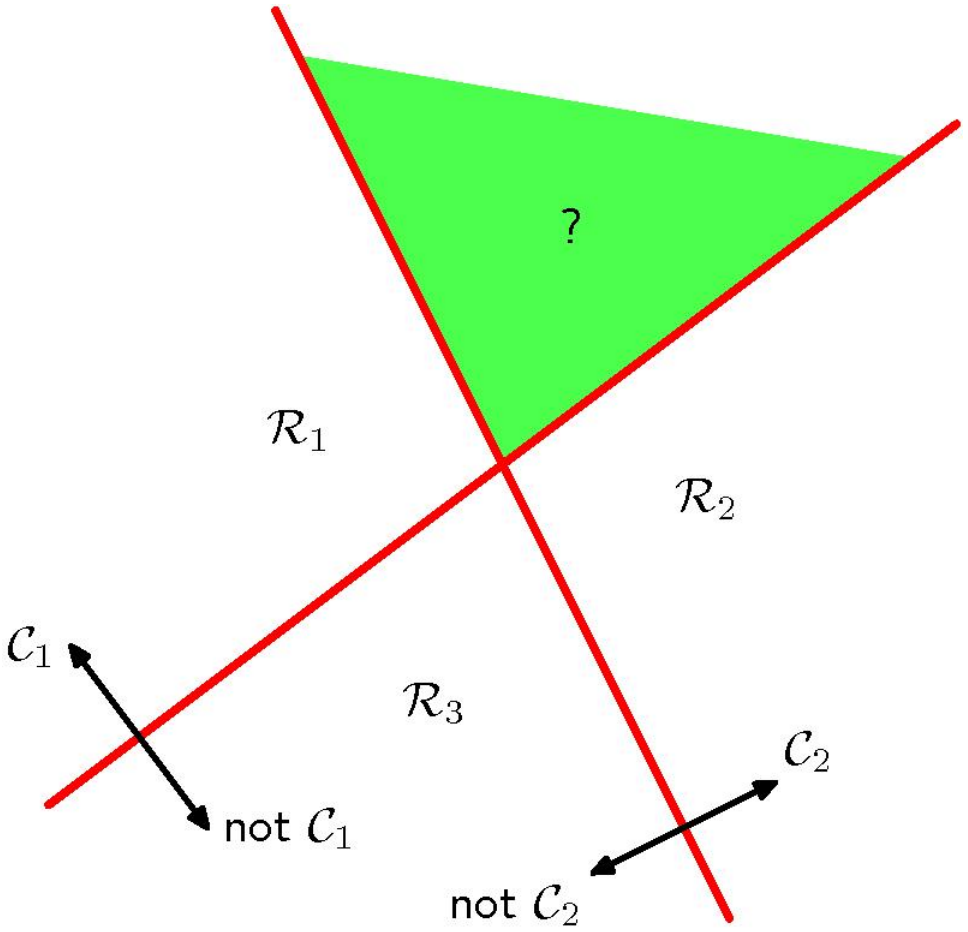
Linear discriminant functions

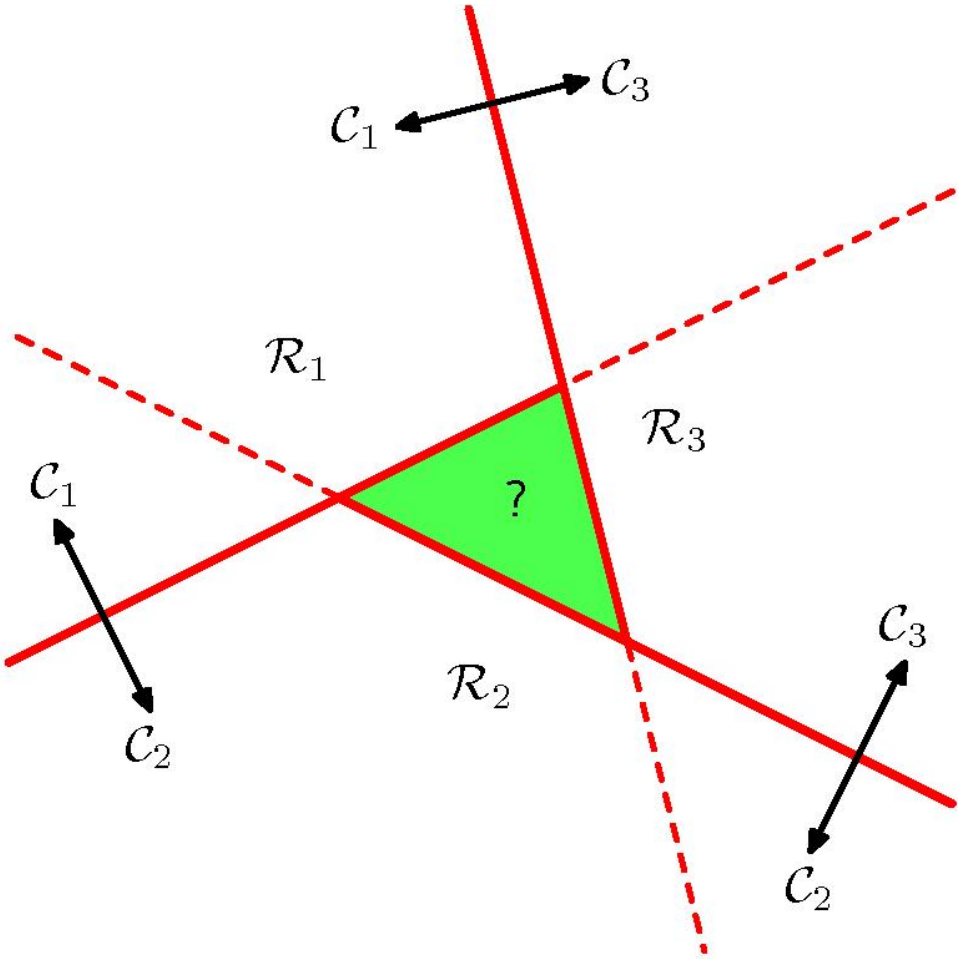
$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$



Multiple Classes

Problem of *ambiguous regions*





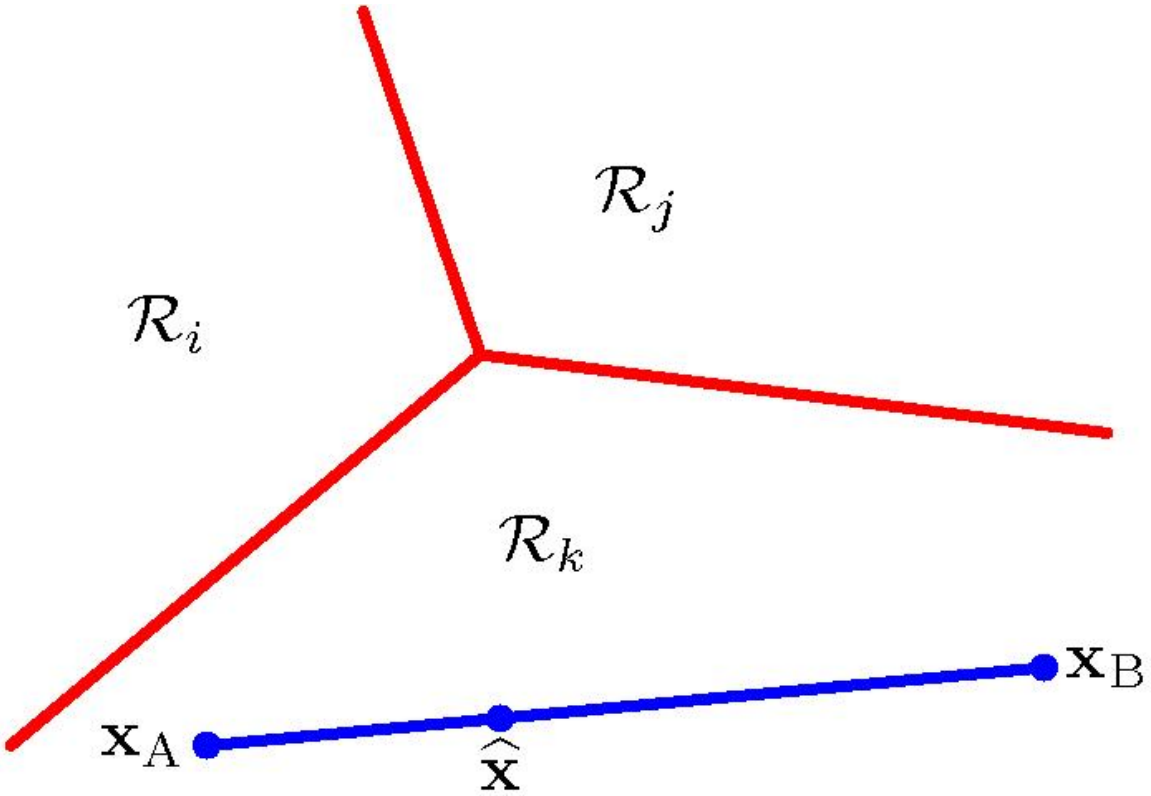
Multiple Classes

Consider a single K-class discriminant, with K linear functions:

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

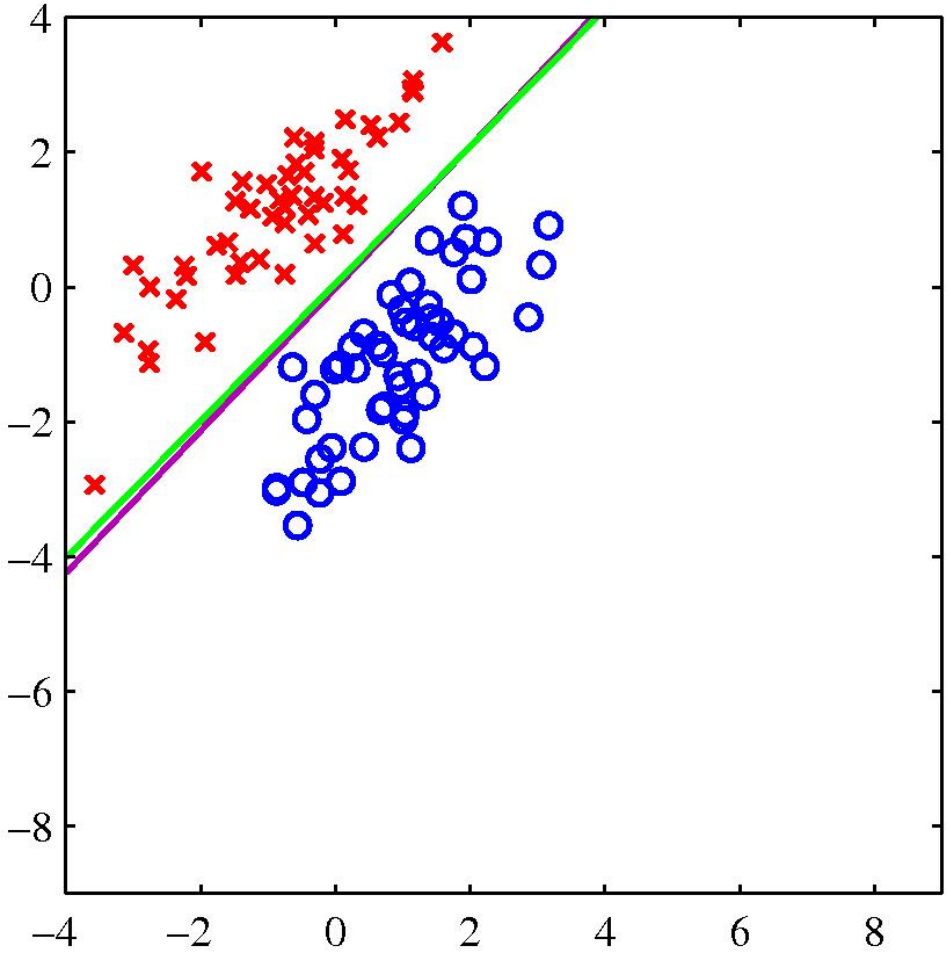
And assign $\mathbf{x}$ to class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$

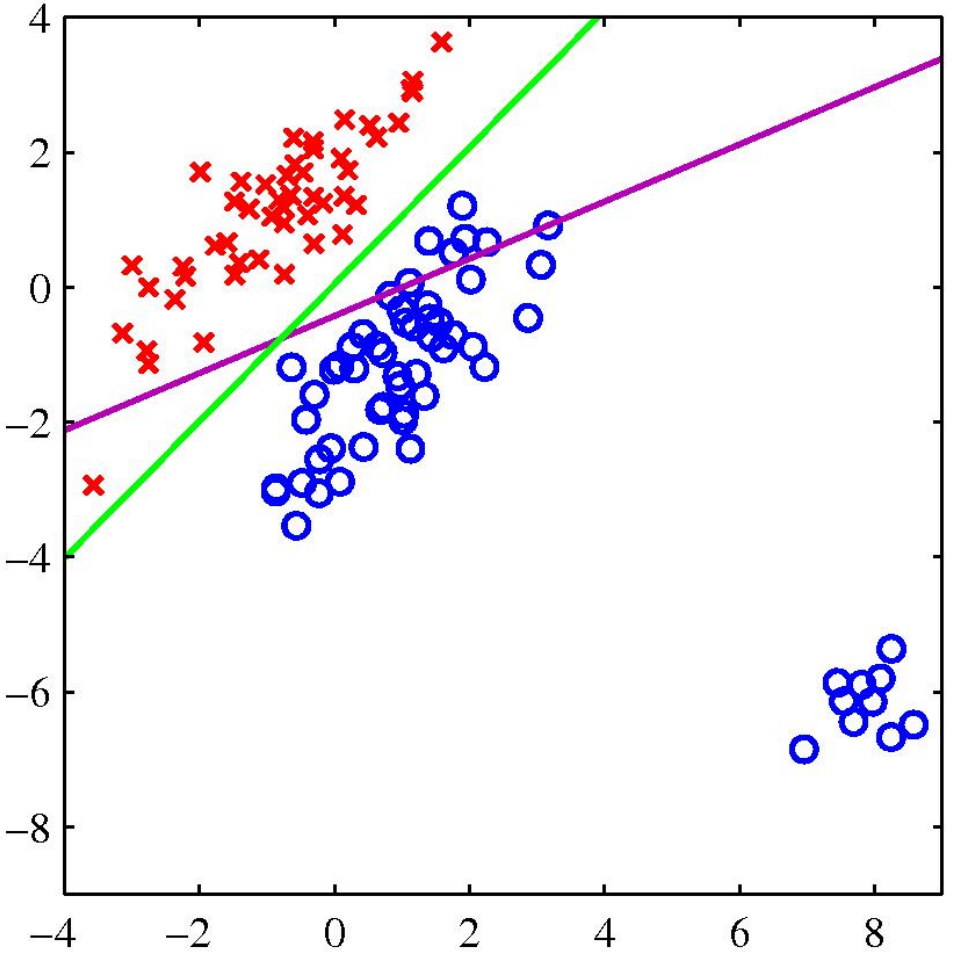
Implies singly connected and convex decision regions:



Least squares for classification

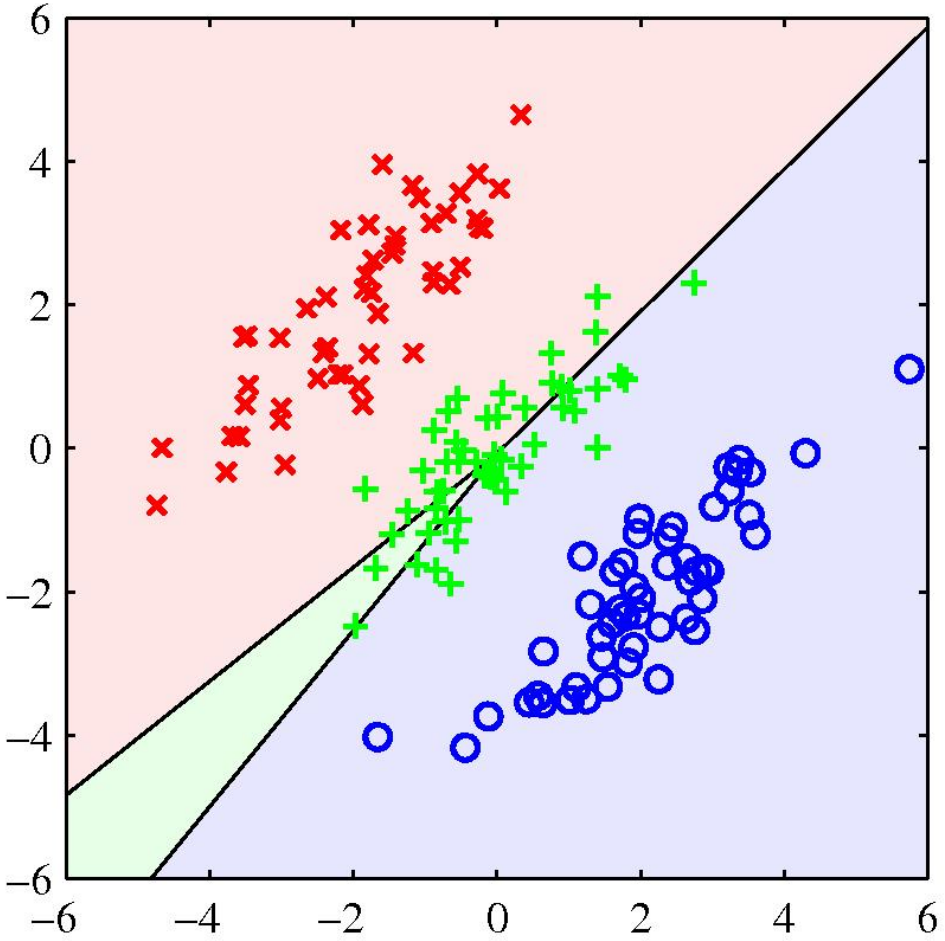
Too sensitive to outliers:

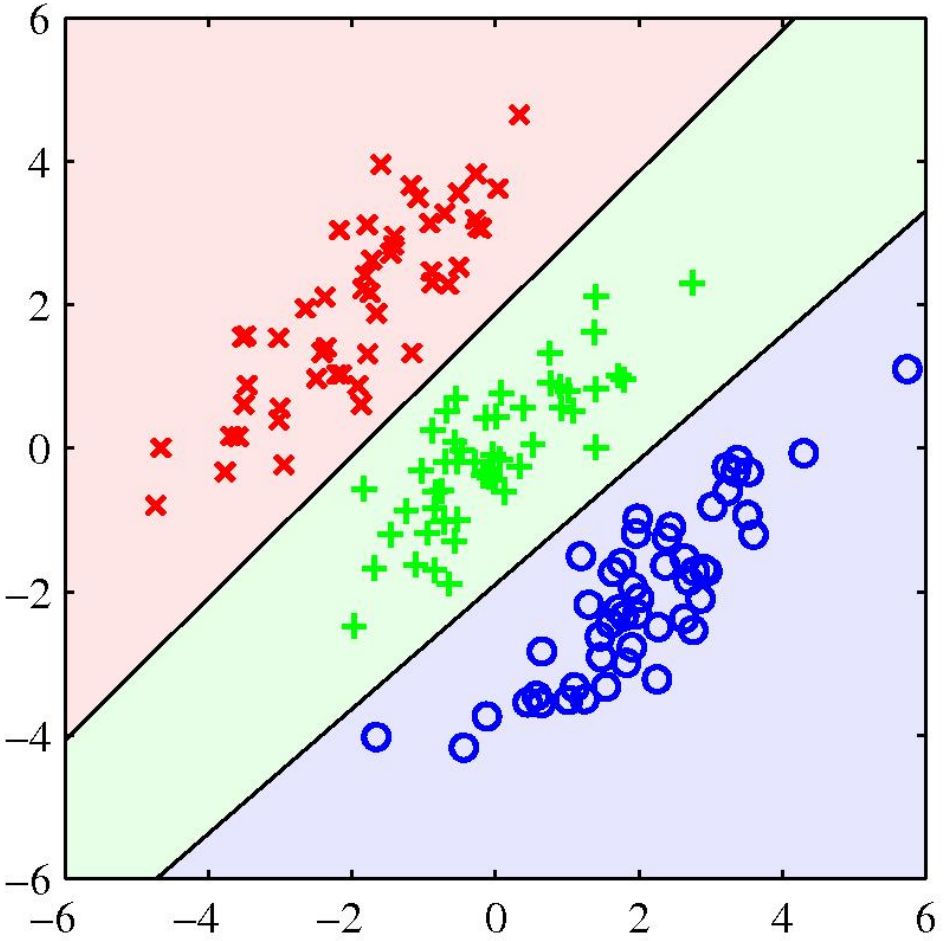




Least squares for classification

Problematic due to evidently non-Gaussian distribution of target values:





Fisher's linear discriminant

Linear classification model is like 1-D projection of data: $y = \mathbf{w}^T\mathbf{x}$. Thus we need to find a decision threshold along this 1-D projection (line). Simplest measure is separation of the class means: $m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$. If classes have nondiagonal covariances, then a better idea is to use the *Fisher criterion*:

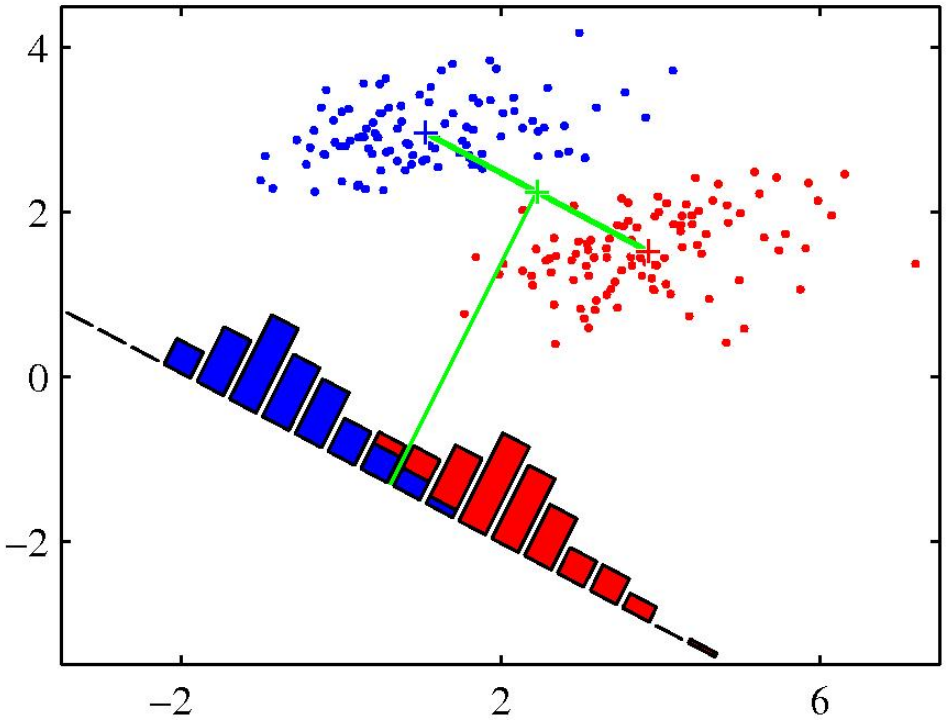
$$J(\mathbf{w}) = (m_2 - m_1)^2 / (s_1^2 + s_2^2)$$

Where s_1^2 denotes the variance of class 1 in the 1-D projection.

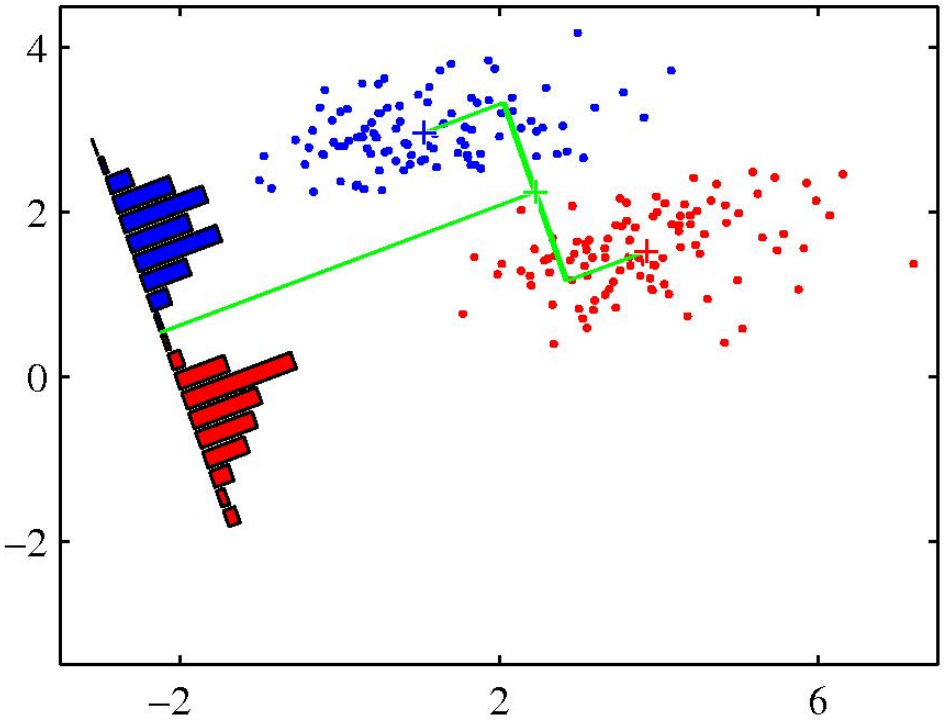
Maximising J() attempts to give a large separation between projected class means, but also a small variance within each class.

Fisher's linear discriminant

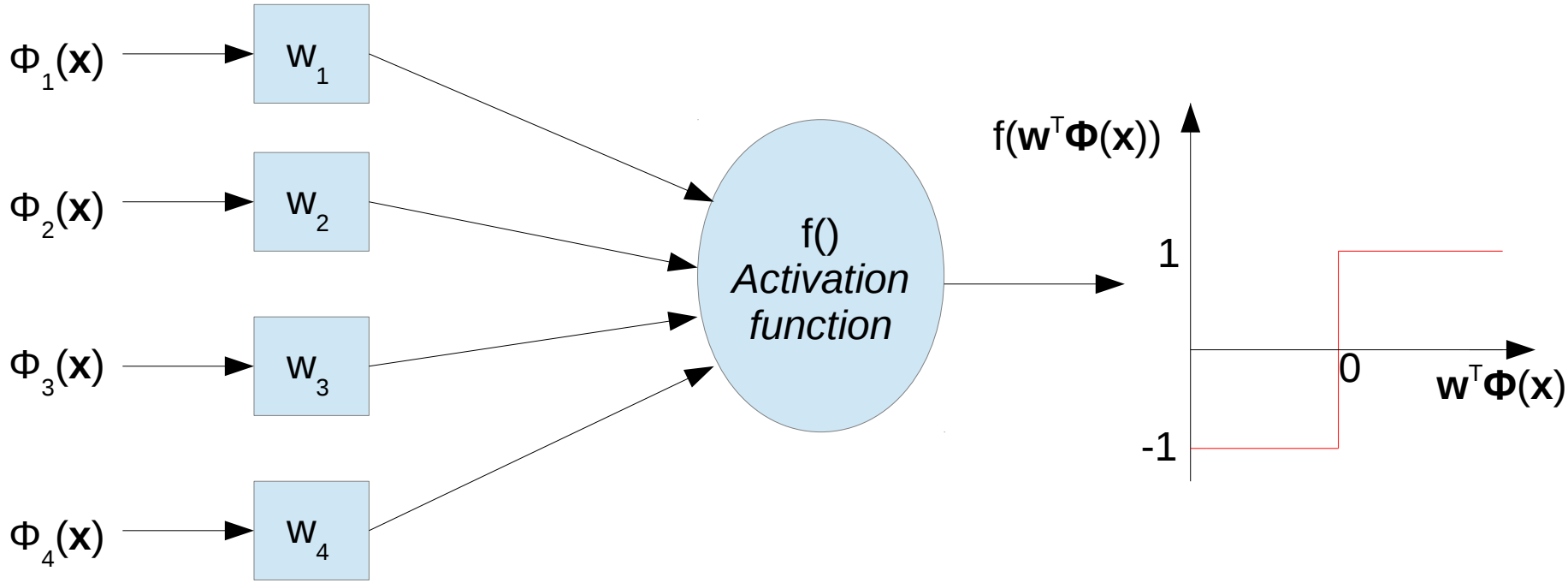
Line joining class means



Fisher discriminant



The Perceptron



A non-linear transformation in the form of a step function is applied to the weighted sum of the input features. This is inspired by the way neurons appear to function, mimicking the *action potential*.

The perceptron criterion

- We'd like a weight vector $\mathbf{w}$ such that $\mathbf{w}^T\mathbf{\Phi}(\mathbf{x}_i) > 0$ for $\mathbf{x}_i \in C_1$ (say, $t_i=1$) and $\mathbf{w}^T\mathbf{\Phi}(\mathbf{x}_i) < 0$ for $\mathbf{x}_i \in C_2$ ($t_i=-1$)
- Thus, we want $\mathbf{w}^T\mathbf{\Phi}(\mathbf{x}_i)t_i > 0 \; \forall$ i; those data points for which this is not true will be misclassified
- The *perceptron criterion* tries to minimise the 'magnitude' of misclassification, i.e., it tries to minimise $-\mathbf{w}^T\mathbf{\Phi}(\mathbf{x}_i)t_i$ for all misclassified points (the set of which is denoted by *M*):

$$E_P(\mathbf{w}) = -\sum_{i \in M} \mathbf{w}^T\mathbf{\Phi}(\mathbf{x}_i)\mathbf{t}_i$$

- Why not just count the number of misclassified points? Because this is a piecewise constant function of **w**, and thus the gradient is zero at most places, making optimisation hard

Learning by gradient descent

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

$$= \mathbf{w}^{(\tau)} + \eta \mathbf{\Phi}(\mathbf{x}_i)\mathbf{t}_i$$

(if $\mathbf{x}_i$ is misclassified)

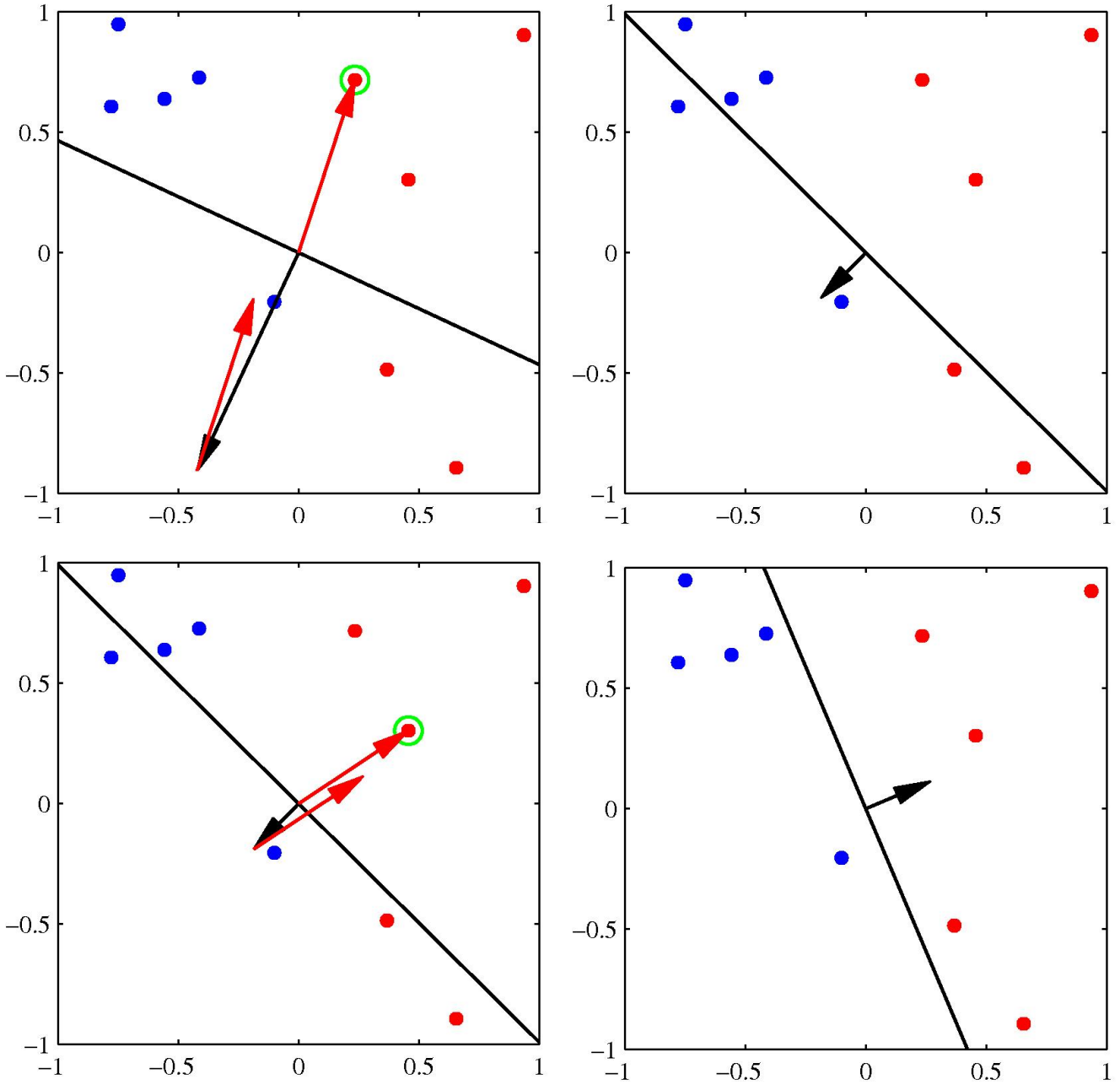
We can show that after this update, the error due to $\mathbf{x}_i$ will be reduced:

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

$$< -\mathbf{w}^{(\tau)T}\mathbf{\Phi}(\mathbf{x}_i)\mathbf{t}_i$$

(having set $\eta=1$, which can be done without loss of generality)

Perceptron convergence



Perceptron convergence theorem guarantees exact solution in finite steps for linearly separable data; but no convergence for nonseparable data

Gaussian Discriminant Analysis

- Generative approach, with class-conditional densities (likelihoods) modelled as Gaussians

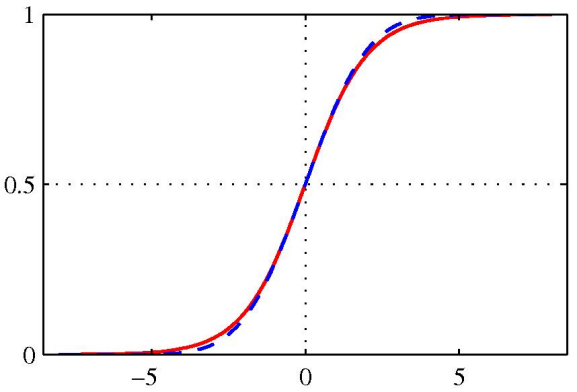
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}.$$

For the case of two classes, we have:

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

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

Logistic sigmoid



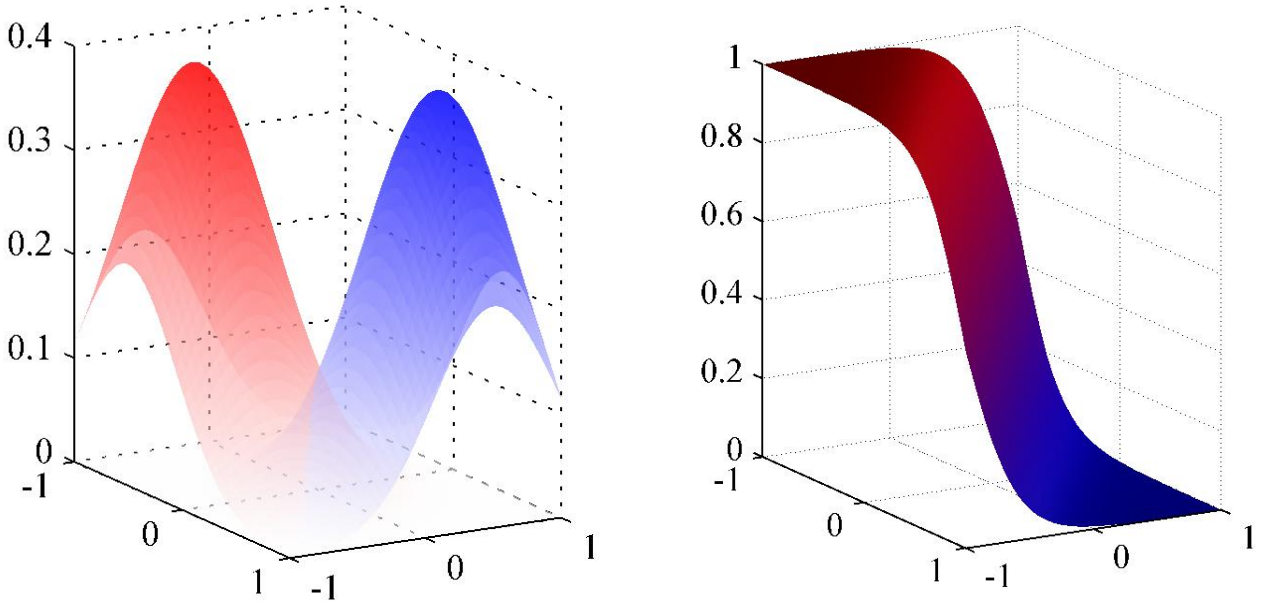
Gaussian Discriminant Analysis

- In the Gaussian case, we get

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

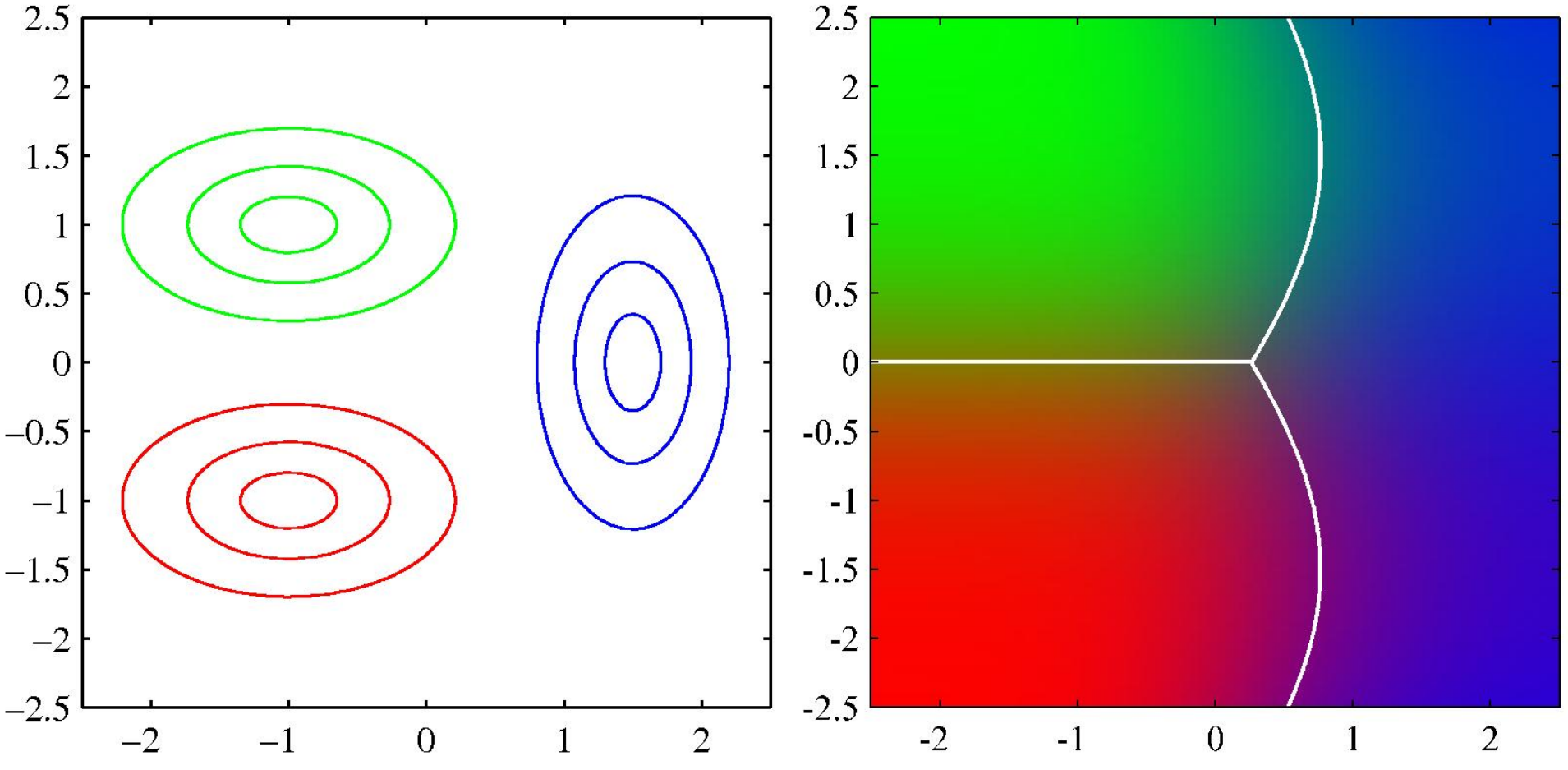
$$\begin{aligned} \mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}.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

The assumption of equal covariance matrices leads to linear decision boundaries



Gaussian Discriminant Analysis

Allowing for unequal covariance matrices for different classes leads to quadratic decision boundaries



Parameter estimation for GDA

Likelihood:
(assuming equal covariance matrices)

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

Maximum Likelihood Estimators

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

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

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

$$\boldsymbol{\Sigma} = \mathbf{S}$$

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

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

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

Logistic Regression

- An example of a *probabilistic discriminative model*
- Rather than learning $P(\boldsymbol{x}|C_i)$ and $P(C_i)$, attempts to directly learn $P(C_i|\boldsymbol{x})$
- Advantages: fewer parameters, better if assumptions in class-conditional density formulation are inaccurate
- We have seen how the class posterior for a two-class setting can be written as a logistic sigmoid acting on a linear function of the feature vector $\boldsymbol{\Phi}$:

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

- This model is called *logistic regression*, even though it is a model for classification, not regression!

Parameter learning

- If we let

$$y_n = p(\mathcal{C}_1|\boldsymbol{\phi}_n)$$

then the likelihood function is

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

and we can define a corresponding error, known as *cross-entropy*:

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

Parameter learning

- The derivative of the sigmoid function is given by:

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

- Using this, we can obtain the gradient of the error function with respect to **w**:

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

- Thus the contribution to the gradient from point *n* is given by the 'error' between model prediction and actual class label ($y_n - t_n$) times the basis function vector for that point, $\boldsymbol{\Phi}_n$
- Could use this for sequential learning by gradient descent, exactly as for least-squares linear regression

Nonlinear basis functions