Linear models for classification (supervised learning)

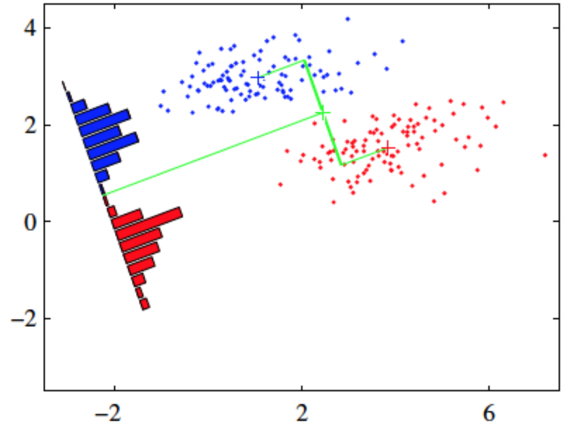
The goal in classification is to take an **input vector x** and to assign it to **one of K discrete classes C_k** (k = 1, . . . , K)

The simplest models for classification are **linear in the input space**, i.e. the decision boundaries that separate the classes are linear functions of the input vector x – they are **(D – 1)-dimensional hyperplanes within the D-dimensional input space x.**

- **Least squares**: $y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$

Tr[$(XW-T)^T(XW-T)$] = Σ_{samples} $(\text{data}_{\text{sample}} * \text{weights} – \text{label}_{\text{sample}})^2$

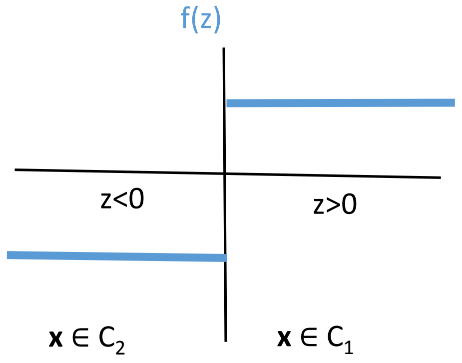
Not robust to outliers and can fail with multiple classes.



- **Fisher linear discriminant**: Minimize the ratio:

$$\frac{(\text{difference of means of projections})^2}{(\text{within class variance of projections})} = \frac{(\text{m2}-\text{m1})^2}{s_1^2+s_2^2} = \frac{\text{between}-\text{class variance}}{\text{total within}-\text{class variance}}$$

Activation function



- **Perceptron algorithm**. y(**x**) = f(**w**^{T}*$\boldsymbol{\varphi}$(**x**)) Can be minimized with stochastic gradient descent: an iterative algorithm where, at iteration τ,

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

You don’t know in advance the number of iterations needed, and it’s hard to distinguish between very-slowly-converging solution and no possible solution.

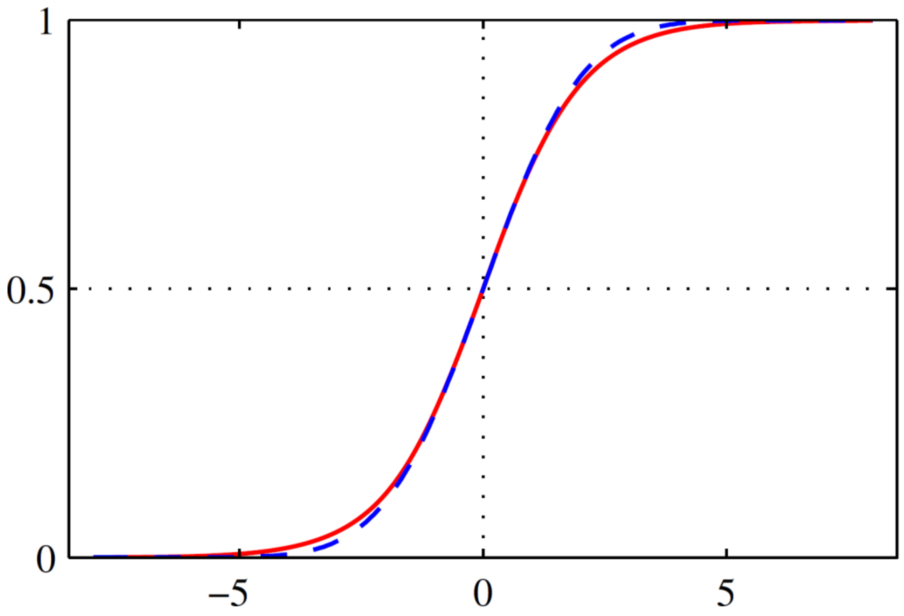
Probabilistic Generative Models

Here the approach is to **find the parameters of a generalized linear model**, by fitting class-conditional densities and class priors separately and then applying Bayes' theorem to get P(Ck|x). This represents an example of **generative modelling**, because we could take such a model and generate synthetic data by drawing values of **x** from the marginal distribution *p*(**x**).

Logistic function:

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

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



Softmax function (for more than 2 classes):

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

Example

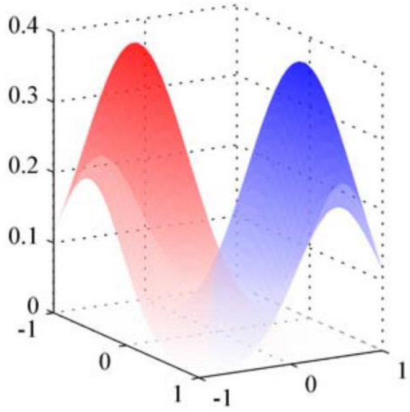
Assume that the **class-conditional densities are Gaussian** and then explore the resulting form for the posterior probabilities.

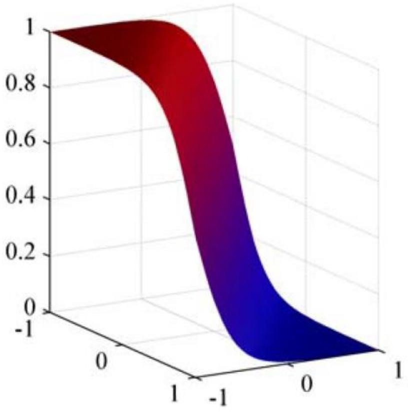
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}_k) \right\}.$$

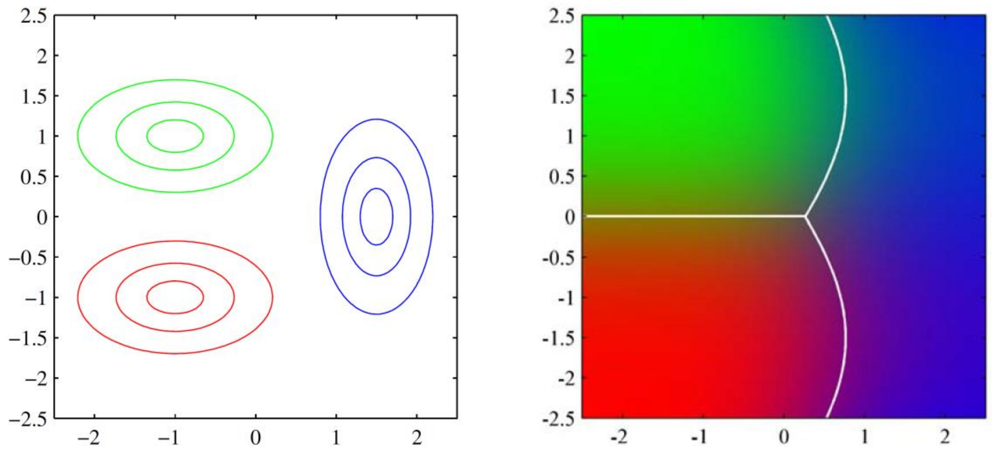
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

With:

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







the boundary between the red and green classes, which have the same covariance matrix, is linear, whereas those between the other pairs of classes are quadratic

Maximum likelihood solution

<u>Continuous case</u>:
Once we have specified a parametric functional form for the class-conditional densities *p*(**x**/*Ck*), we can then **determine the values of the parameters**, together with the prior class probabilities *p*(*Ck*), using maximum likelihood.

Consider first the case of **two classes**, each having a **Gaussian class-conditional** density with a shared covariance matrix. The likelihood function is given by:

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

Then we just need to set the derivative with respect to π, μ_1, μ_2 *and* Σ *to zero to determine the parameters.*

Maximum likelihood solution

<u>**Discrete case**</u>:

Assume values $x_i \in \{0, 1\}$ → If there are D inputs, then a general distribution would correspond to $2^D - 1$ independent variables, which grows exponentially with the number of features. Therefore we make the ***naive Bayes* assumption** in which the feature values are treated as independent, conditioned on the class C_k.

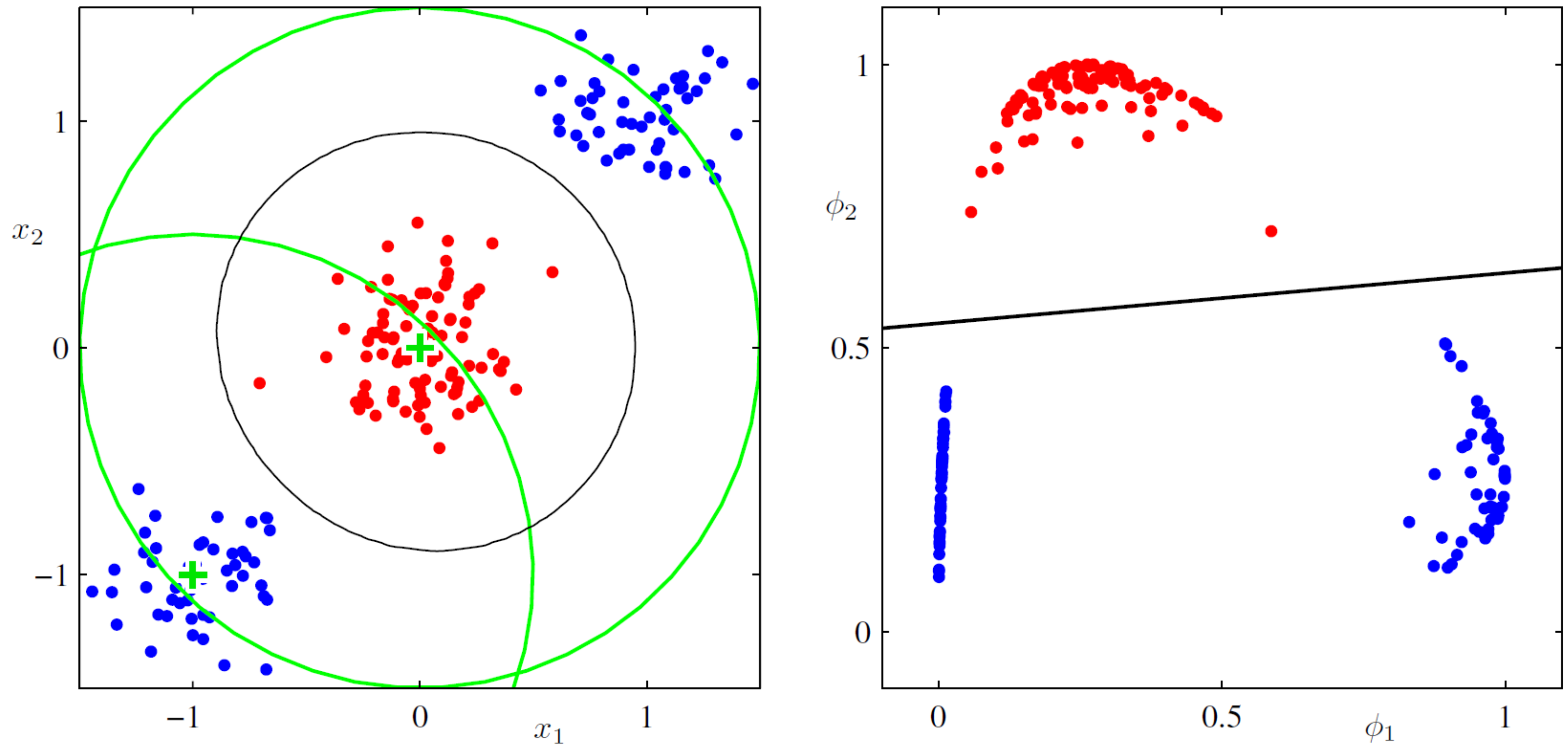
$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i}(1-\mu_{ki})^{1-x_i}$$

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

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

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

Basis Function



$$\mathbf{x} \longrightarrow \phi(\mathbf{x})$$

Probabilistic Discriminative Models

An alternative approach is to **use the functional form of the generalized linear model explicitly** and to determine its parameters directly by using maximum likelihood.
In this more direct approach, we are maximizing a likelihood function defined through the conditional distribution p(Ck|x), which represents a form of **discriminative training**. One advantage of this discriminative approach is that there will typically be **fewer adaptive parameters to be determined**, as we shall see shortly.

Generative approach

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

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

We need to model these

Discriminative model approach

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

Logistic regression

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

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

$$\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}} \text{ and } y_n = p(\mathcal{C}_1|\boldsymbol{\phi}_n)$$

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

We can define an error function by taking the negative logarithm of the likelihood, which gives the cross-entropy error function in the form:

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

Taking the gradient of the error function with respect to w, we obtain:

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

Iterative reweighted least squares

For logistic regression, there is no closed-form solution as in Linear Regression, due to the nonlinearity of the logistic sigmoid function. However, the error function can be minimized by an efficient iterative technique based on the *Newton-Raphson* iterative optimization scheme, that uses a **local quadratic approximation** to the log likelihood function:

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

where **H** is the Hessian matrix whose elements comprise the second derivatives of $E(\mathbf{w})$ with respect to the components of **w**. The update formula is (see section 4.3.3 in Bishop's book):

$$\boxed{\mathbf{w}^{(\text{new})} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{z}}$$

Where

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

And **R** is a $N \times N$ diagonal matrix with elements:

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

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

Multiclass logistic regression

In the case of more than 2 classes, the posterior probabilities can be often modeled by a **softmax transformation** of linear functions of the feature variables, so that

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

where the 'activations' a_k are given by

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

The *cross-entropy* error function for the multiclass classification problem is given by

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

And its gradient (as in the logistic regression case) is

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