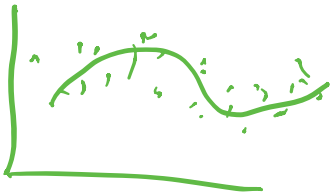
Homework

$y = X\beta + \varepsilon$, $X \in \mathbb{R}^{m \times n}$ $m < n$



$\hat{\beta} = (X^T X)^{-1} X^T y$

$(X^T X)^{-1} X^T \rightarrow \text{pinv}(X)$

$\varepsilon_t(+|X) - +$

$\hat{\beta} = \min_{\beta} \|y - X\beta\|^2_{\Sigma^{-1}}$

$\max_{\beta} \left(\log\left(P(y \mid X, \beta, \Sigma) \right) \right) = \hat{\beta}$

$MSE(\hat{\beta}) = E[\hat{\beta} - \beta] = \sigma^2(\hat{\beta}) + B(\hat{\beta})^2$

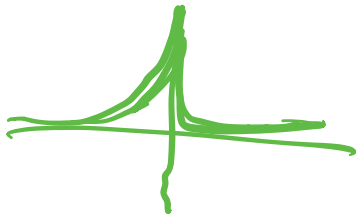
Homework

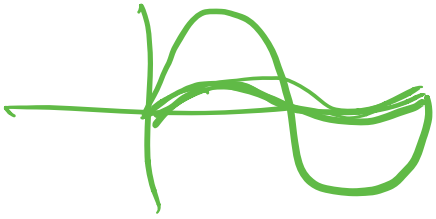
$$\min_{\beta} \{ \|y - X\beta\|^2_{\Sigma^{-1}} + \lambda_2 \beta^T \beta \}$$

$$P(y \mid X, \beta, \Sigma)\, P(\beta) + \lambda \sum_{i=1}^{n} |\beta_i|$$

$\beta \sim N$

$\beta \sim$ Laplace





$y(x) - E_D(y(x))$

Lecture 7: Linear Classification Methods

Final projects?

Groups

Topics

Proposal week 5

Lecture 20 is poster session, Jacobs Hall Lobby, snacks

Final report 15 June.

What is "linear" classification?

Classification is intrinsically non-linear

It puts non-identical things in the same class, so a difference in input vector sometimes causes zero change in the answer

"Linear classification" means that **the part that adapts is linear**

The adaptive part is followed by a fixed non-linearity.

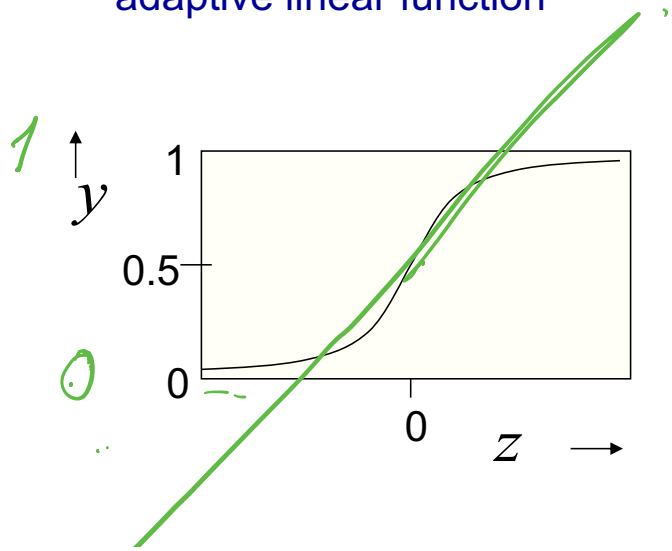
It may be preceded by a fixed non-linearity (e.g. nonlinear basis functions).

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

adaptive linear function

$$Decision = f(y(\mathbf{x}))$$

fixed non-linear function



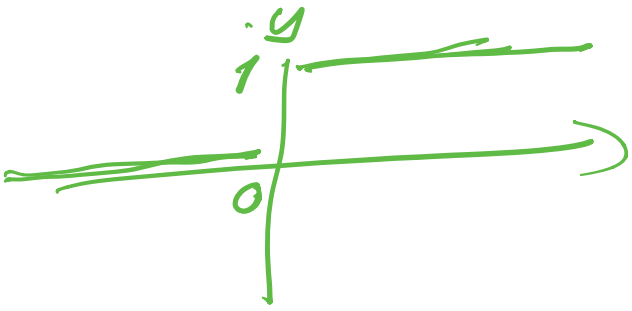
Representing the target values for classification

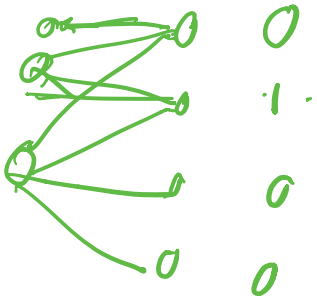
For two classes, we use a single valued output that has target values **1** for the "positive" class and **0** (or **-1**) for the other class

For probabilistic class labels the target value can then be **P(t=1)** and the model output can also represent **P(y=1).**

For **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 input vector into real values. A simple operation (like thresholding) can get the class.

Choose real values to maximize the useable information about the class label that is in the real value.

✓Infer **conditional class probabilities**: $p(class = C_k \mid \mathbf{x})$

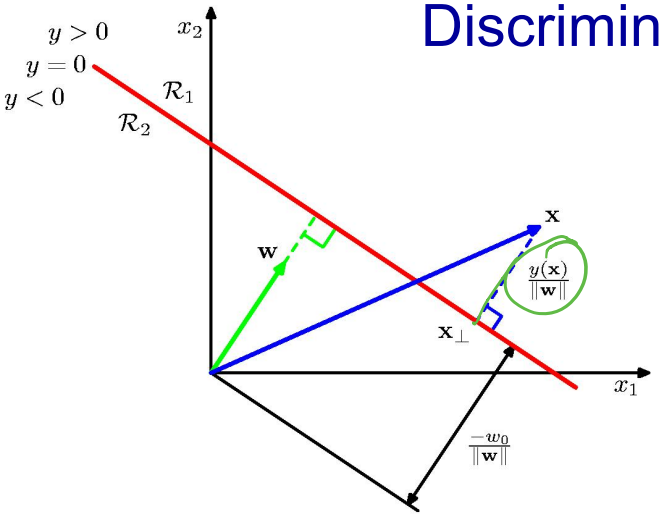
Compute the conditional probability of each class.

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. (Is this the best bet?)

Discriminant functions



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

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

$y = w^T x + w_0$.

$y_\perp = w^T x_\perp + w_0 = 0$ ✓

$\|w\|_2^2 = w^T w$

X on plane => y=0 =>

Distance from plane

$r = \frac{y}{\|w\|_2}$

$x = x_\perp + r\frac{w}{\|w\|_2}$

$w^T x = w^T x_\perp + r\frac{w^T w}{\|w\|_2}$

$= -w_0 + r\|w\|_2$

$x = x_\perp + \frac{y}{\|w\|_2}$

Discriminant functions for N>2 classes

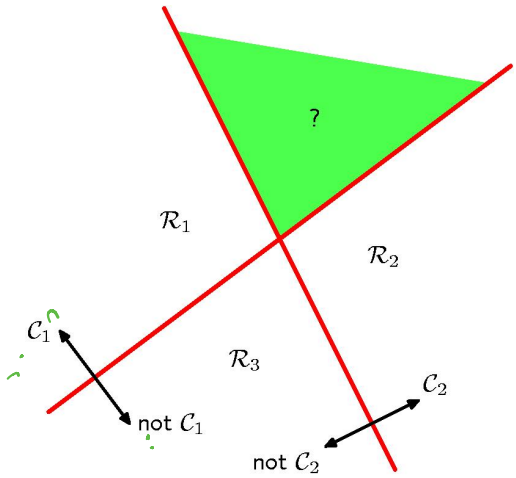
One possibility is to use N two-way discriminant functions.

Each function discriminates one class from the rest.

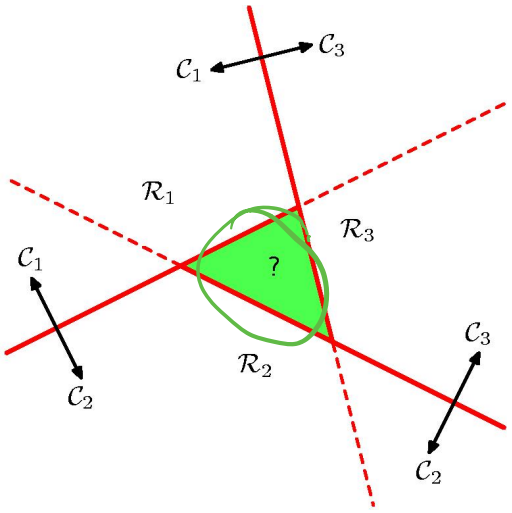
Another possibility is to use N(N-1)/2 two-way discriminant functions

Each function discriminates between two particular classes.

Both these methods have problems



More than one good answer

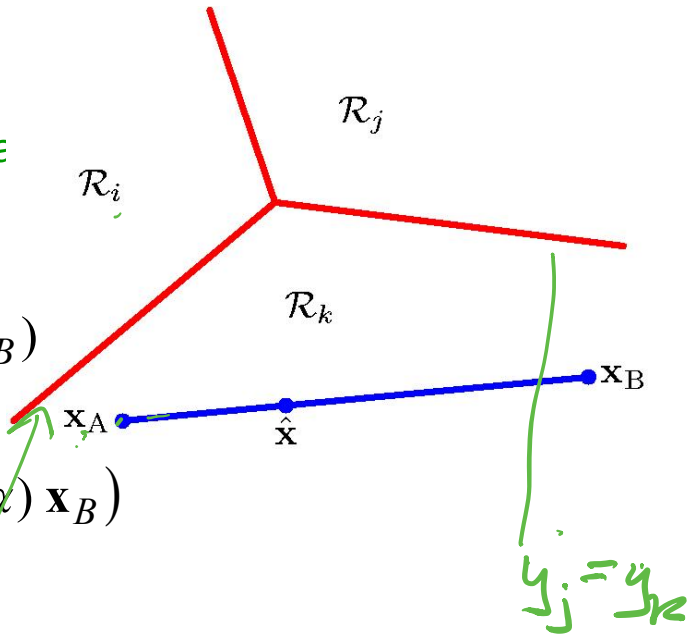


Two-way preferences need not be transitive!

A simple solution (4.1.2)

Use N discriminant functions, and pick the max. $y_i, y_j, y_k \cdots$

This is guaranteed to give consistent a 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 α*) that*

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

$y_j = y_k$

$y_k = y_i$

$y_k = \mathbf{w}_k^T \mathbf{x} + w_0$

Decision boundary?

Maximum Likelihood and Least Squares (from lecture 3)

Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}.$$

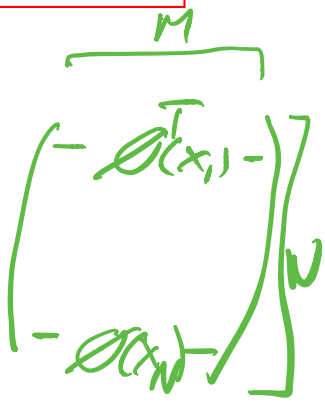
Solving for w,

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

The Moore-Penrose pseudo-inverse, $\boldsymbol{\Phi}^{\dagger}$.

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$



LSQ for classification

Each class $\mathcal{C}_k$ is described by its own linear model so that

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \quad (4.13)$$

where $k = 1, \ldots, K$. We can conveniently group these together using vector notation so that

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} \quad (4.14)$$

Consider a training set $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}, n = 1 \ldots N$

Define **X** and **T**

LSQ solution:

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T} \quad (4.16)$$

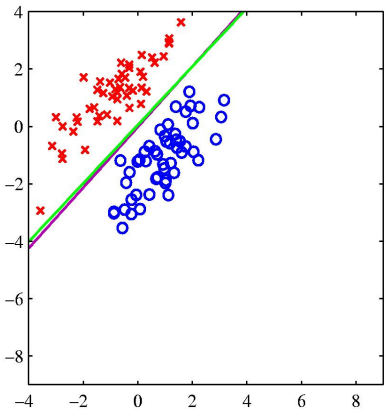
And prediction

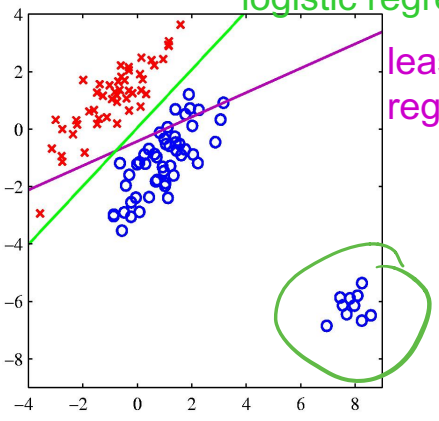
$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} = \mathbf{T}^{\mathrm{T}}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\mathrm{T}}\widetilde{\mathbf{x}}. \quad (4.17)$$

Using “least squares” for classification

It does not work as well as better methods, but it is easy:

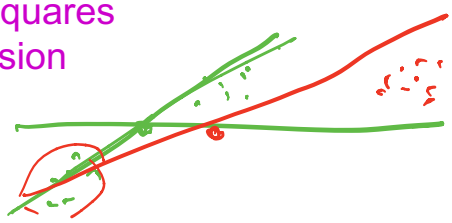
It reduces classification to least squares regression.



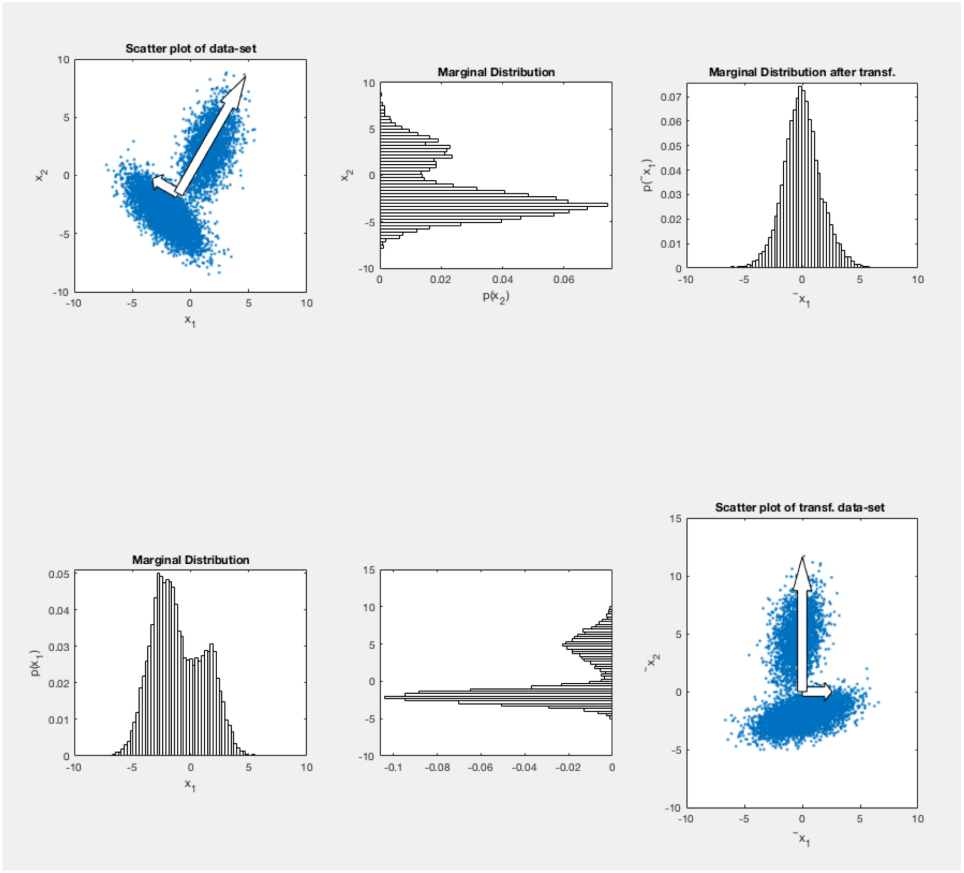


logistic regression

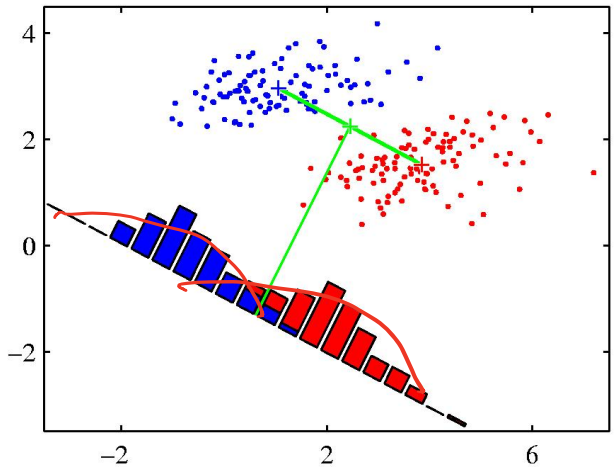
least squares regression



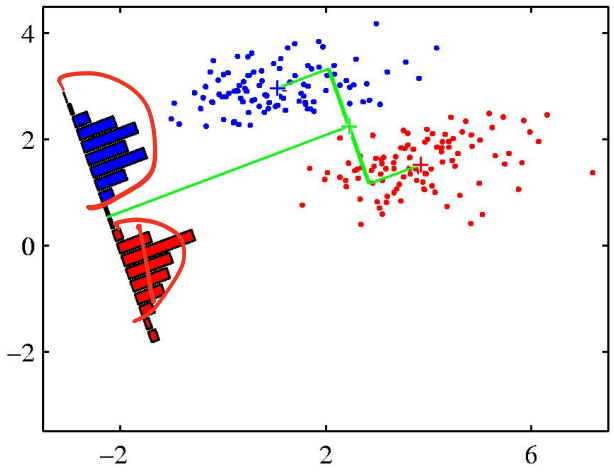
PCA don’t work well



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:

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

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

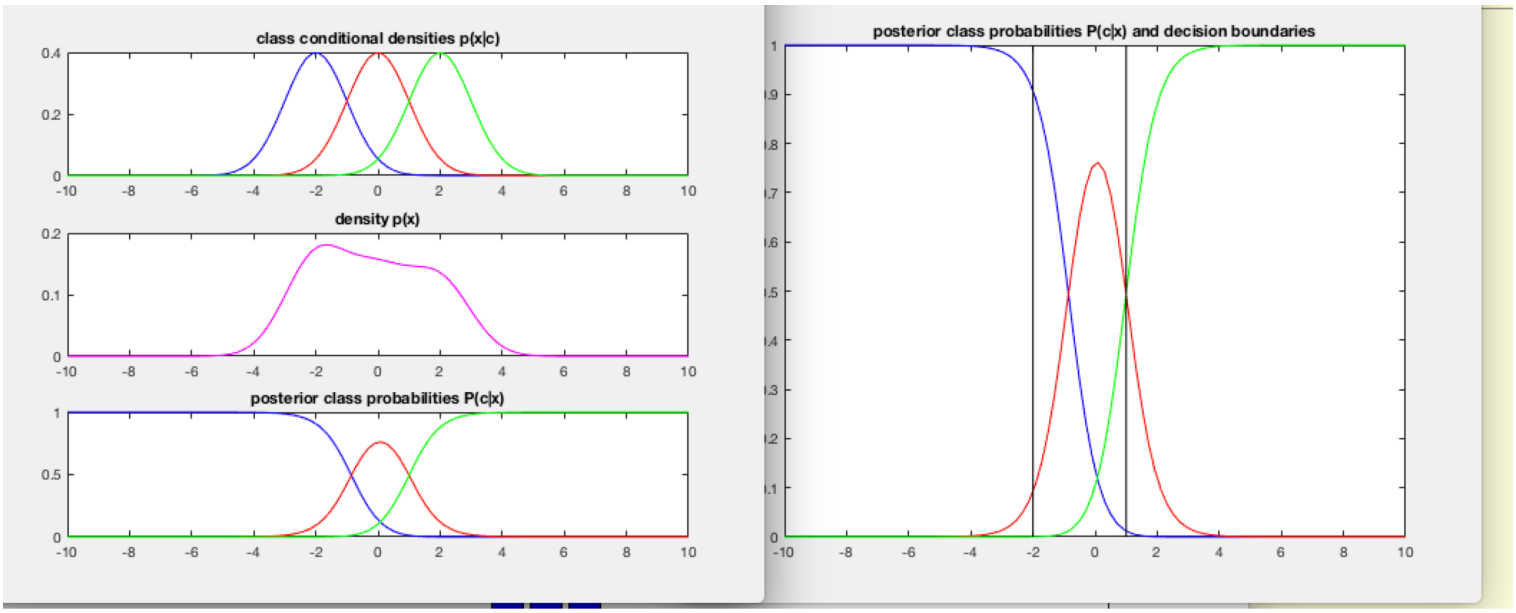
$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

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

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

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

We have probalistic classification!



Probabilistic Generative Models for Discrimination (Bishop p 196)

Use a generative model of the input vectors for each class, and see which model makes a test input vector most probable.

The posterior probability of class 1 is:

$$p(C_1 \mid \mathbf{x}) = \frac{p(C_1)p(\mathbf{x} \mid C_1)}{p(C_1)p(\mathbf{x} \mid C_1) + p(C_0)p(\mathbf{x} \mid C_0)} = \frac{1}{1+e^{-z}}$$

sigmoid

$$\textit{where} \quad z = \ln \frac{p(C_1)p(\mathbf{x} \mid C_1)}{p(C_0)p(\mathbf{x} \mid C_0)} = \ln \frac{p(C_1 \mid \mathbf{x})}{1 - p(C_1 \mid \mathbf{x})}$$

z is called the **logit** and is given by the **log odds**

An example for continuous inputs

Assume input vectors for each class are Gaussian, all classes have the same covariance matrix.

normalizing constant

inverse covariance matrix

$$p(\mathbf{x} \mid C_k) = a \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

For two classes, C_1 and C_0, the posterior is a logistic:

$$p(C_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 + \ln\frac{p(C_1)}{p(C_0)}$$

$$z = ln\left(\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}\right)$$

$= \ln P(x|C_1) - \ln(x|C_2) + \ln\frac{P(C_1)}{P(C_2)}$

$= -\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^T\Sigma(x-\mu_2) + \text{const}$

$= \mu_1^T\Sigma^{-1}x - \mu_2^T\Sigma^{-1}x + \underbrace{\mu_1^T\Sigma^{-1}\mu_1 + \mu_2\Sigma^{-1}\mu_2 + \text{const}}_{w_0}$

$= w^Tx + w_0$

$w^T = \mu_1^T\Sigma^{-1} - \mu_2^T\Sigma^{-1}$

$w = \Sigma^{-1}(\mu_1 - \mu_2)$

The role of the inverse covariance matrix

If the Gaussian is spherical no need to worry about the covariance matrix.

So, start by transforming the data space to make the Gaussian spherical

- This is called "**whitening**" the data.
- It pre-multiplies by the matrix **square root of the inverse covariance matrix**.

In transformed space, the weight vector is the difference between transformed means.

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

gives the same value for $\mathbf{w}^T\mathbf{x}$ *as:*

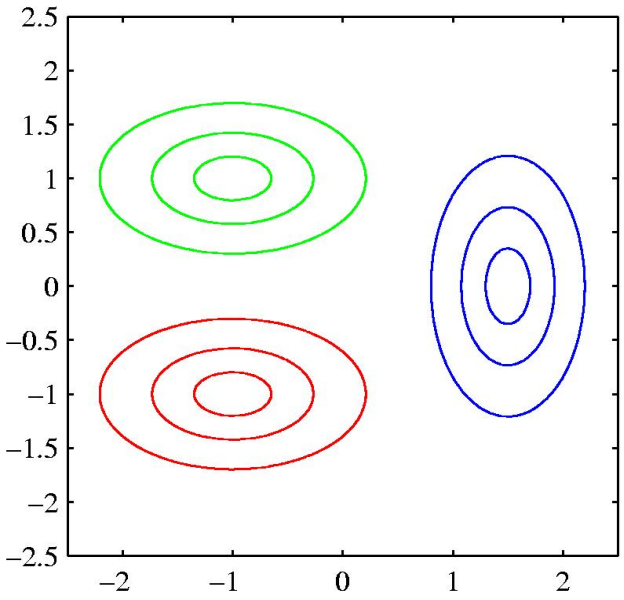
$$\mathbf{w}_{aff} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}_0$$

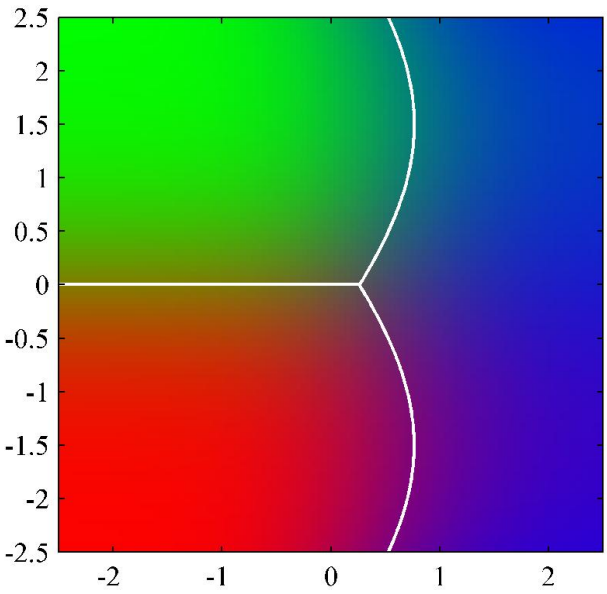
and $\mathbf{x}_{aff} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{x}$

gives for $\mathbf{w}_{aff}^T\mathbf{x}_{aff}$

aff = affine

The posterior when the covariance matrices are different for different classes (Bishop Fig)





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

Bernoulli distribution

Random variable $x \in \{0,1\}$

Coin flipping: heads=1, tails=0

$$p(x=1|\mu) = \mu$$

Bernoulli Distribution

$$\begin{aligned} \text{Bern}(x|\mu) &= \mu^x(1-\mu)^{1-x} \\ \mathbb{E}[x] &= \mu \\ \text{var}[x] &= \mu(1-\mu) \end{aligned}$$

ML for Bernoulli

Given: $\mathcal{D} = \{x_1, \ldots, x_N\}$, m heads (1), $N-m$ tails (0)

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} \ln p(x_n|\mu) = \sum_{n=1}^{N} \{x_n \ln \mu + (1-x_n)\ln(1-\mu)\}$$

head tail

$$\mu_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{m}{N}$$

The logistic function

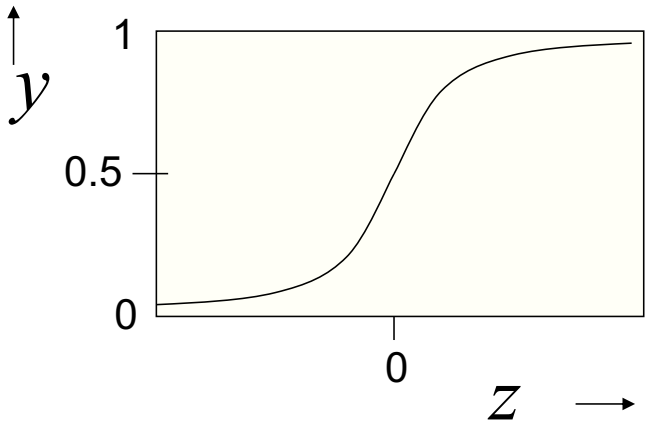
The output is a smooth function of the inputs and the weights.

$$z = \mathbf{w}^T \mathbf{x} + w_0$$

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

$$\frac{\partial z}{\partial w_i} = x_i \qquad \frac{\partial z}{\partial x_i} = w_i$$

$$\frac{dy}{dz} = y(1 - y)$$



Its odd to express it in terms of y.

$$\frac{dy}{dz} = \frac{+e^{-z}}{(1+e^{-z})^2} = y(1-y)$$

$$\frac{e^{-z}}{1+e^{-z}} = 1 - y$$

Logistic regression (Bishop 205)

$p(C_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$

Observations $\{x_n, t_n\}$ $t_n \in [0, 1]$

Likelihood

$y = \sigma(\boldsymbol{w}^T\boldsymbol{x})$

$p(y|\boldsymbol{x}, \boldsymbol{w}) =$ $\text{Bern}(y, \mu) = \mu^y(1-\mu)^{1-\mu}$

$p(T|\boldsymbol{x}, \boldsymbol{w}) =$ $\prod_n^N y_n^{t_n}(1-y_n)^{1-t_n}$

Log-likelihood

$E_{\boldsymbol{w}} = -ln(p(T|\boldsymbol{x}, \boldsymbol{w}))=$ $-\sum_n^N (t_n \ln y_n + (1-t_n)\ln(1-y_n))$

Minimize –log like

Derivative

$\nabla_{\boldsymbol{w}} E_{\boldsymbol{w}} = -\sum_n^N \left[t_n \frac{1}{y_n} + \frac{1-t_n}{1-y_n}\cdot(-1)\right] y_n(1-y_n)\, x_n$

$= \sum_n^N (t_n - y_n)\, x_n$ $\qquad w_{i+1} = w_i - \eta \nabla E_w$

Logistic regression (page 205)

When there are only two classes we can model the conditional probability of the positive class as

$$p(C_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0) \qquad where \quad \sigma(z) = \frac{1}{1 + \exp(-z)}$$

If we use the right error function, something nice happens: The gradient of the logistic and the gradient of the error function cancel each other:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}), \qquad \nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\, \mathbf{x}_n$$