Lectures 5 & 6: Classifiers

Hilary Term 2007 — A. Zisserman

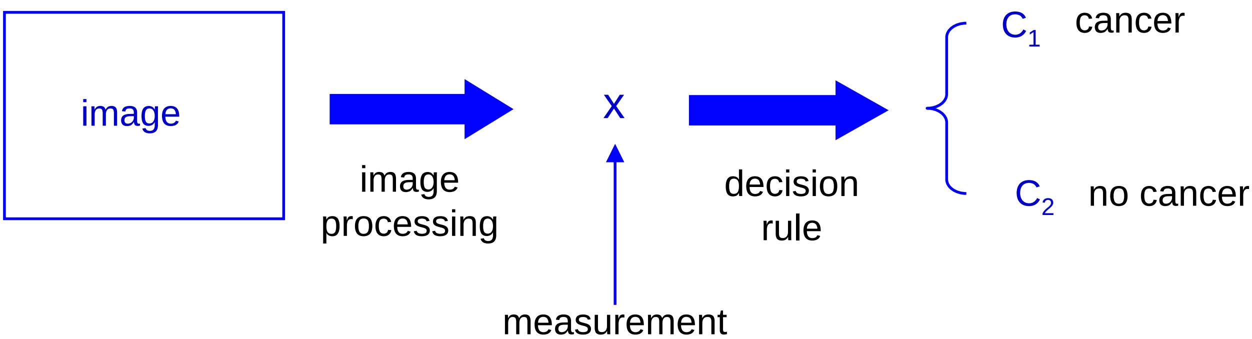
- **Bayesian Decision Theory**
  - Bayes decision rule
  - Loss functions
  - Likelihood ratio test

- **Classifiers and Decision Surfaces**
  - Discriminant function
  - Normal distributions

- **Linear Classifiers**
  - The Perceptron
  - Logistic Regression

Decision Theory

Suppose we wish to make measurements on a medical image and classify it as showing evidence of cancer or not



and we want to base this decision on the learnt joint distribution

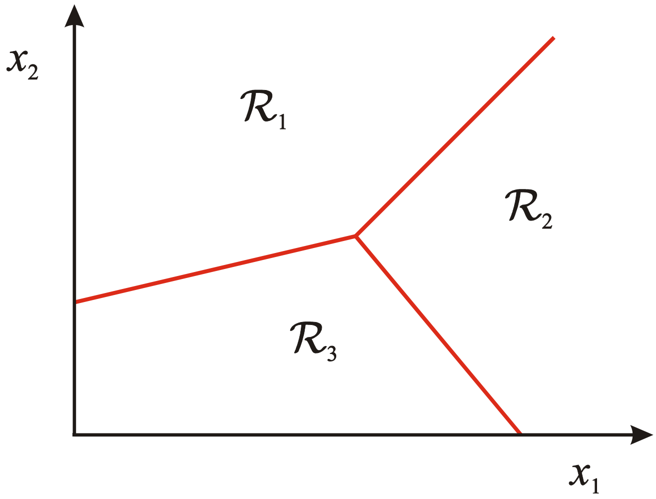
$$p(\mathbf{x}, C_i) = p(\mathbf{x}|C_i)p(C_i)$$

How do we make the "best" decision?

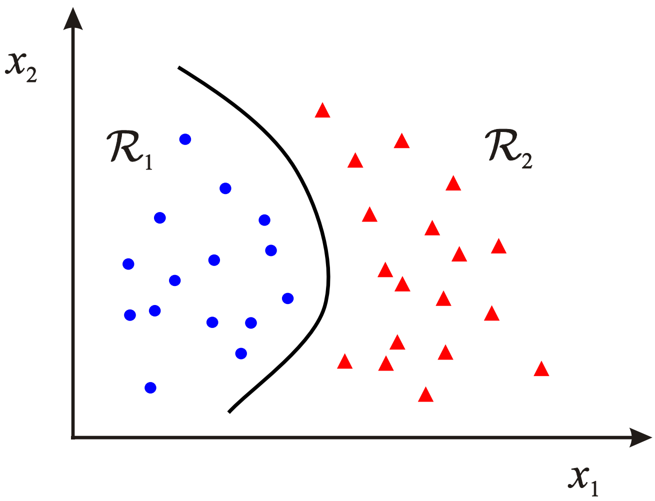
Classification

Assign input vector $\mathbf{x}$ to one of two or more classes C_k

Any decision rule divides input space into *decision regions* separated by *decision boundaries*



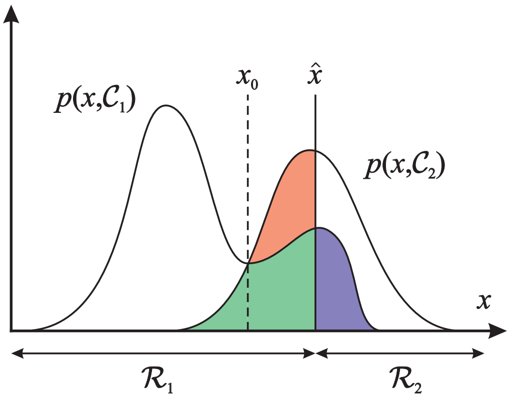
Example: two class decision depending on a 2D vector measurement



Also, would like a confidence measure (how sure are we that the input belongs to the chosen category?)

Decision Boundary for average error

Consider a two class decision depending on a scalar variable x



$$
\begin{aligned}
p(\text{error}) &= \int_{-\infty}^{+\infty} p(\text{error}, x)\, dx \\
&= \int_{\mathcal{R}_1} p(x, C_2)\, dx + \int_{\mathcal{R}_2} p(x, C_1)\, dx
\end{aligned}
$$

minimize number of misclassifications if the decision boundary is at x_0

Bayes Decision rule

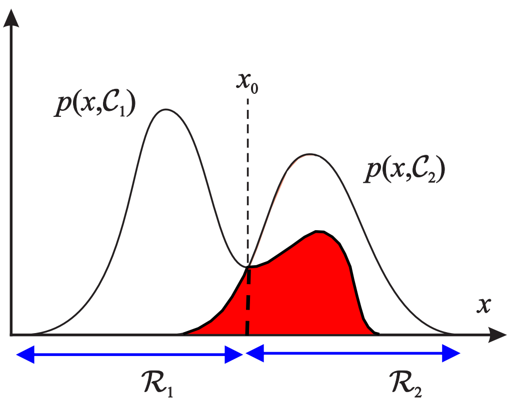
Assign x to the class C_i for which $p(x, C_i)$ is largest

since $p(x, C_i) = p(C_i|x)\, p(x)$ this is equivalent to

Assign x to the class C_i for which $p(C_i | x)$ is largest

Bayes error

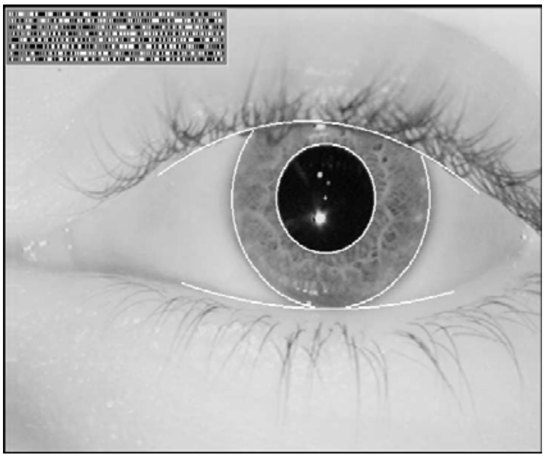
A classifier is a mapping from a vector x to class labels $\{C_1, C_2\}$

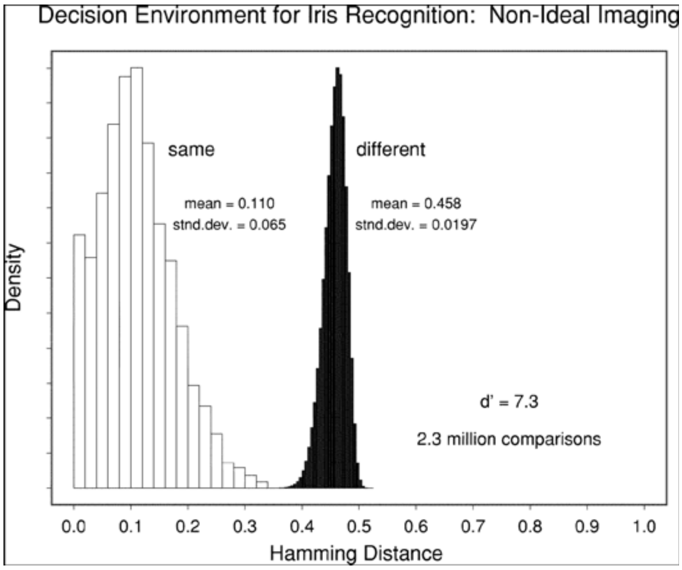


$$
\begin{aligned}
p(\text{error}) &= \int_{-\infty}^{+\infty} p(\text{error}, x)\, dx \\
&= \int_{\mathcal{R}_1} p(x, C_2)\, dx + \int_{\mathcal{R}_2} p(x, C_1)\, dx \\
&= \int_{\mathcal{R}_1} p(C_2|x) p(x)\, dx + \int_{\mathcal{R}_2} p(C_1|x) p(x)\, dx
\end{aligned}
$$

The Bayes error is the probability of misclassification

Example: Iris recognition



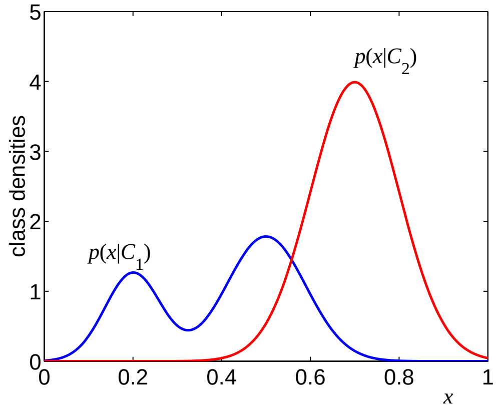


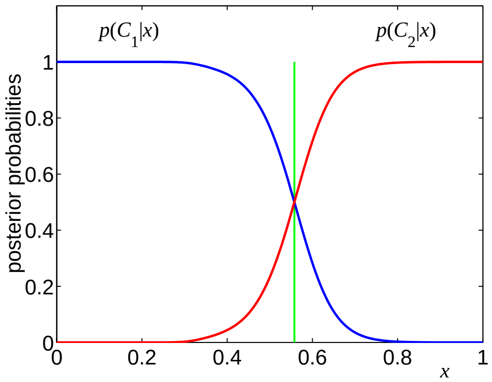
How Iris Recognition Works, John Daugman

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 14, NO. 1, JANUARY 2004

Posteriors

Assign x to the class C_i for which $p(C_i \mid x)$ is largest





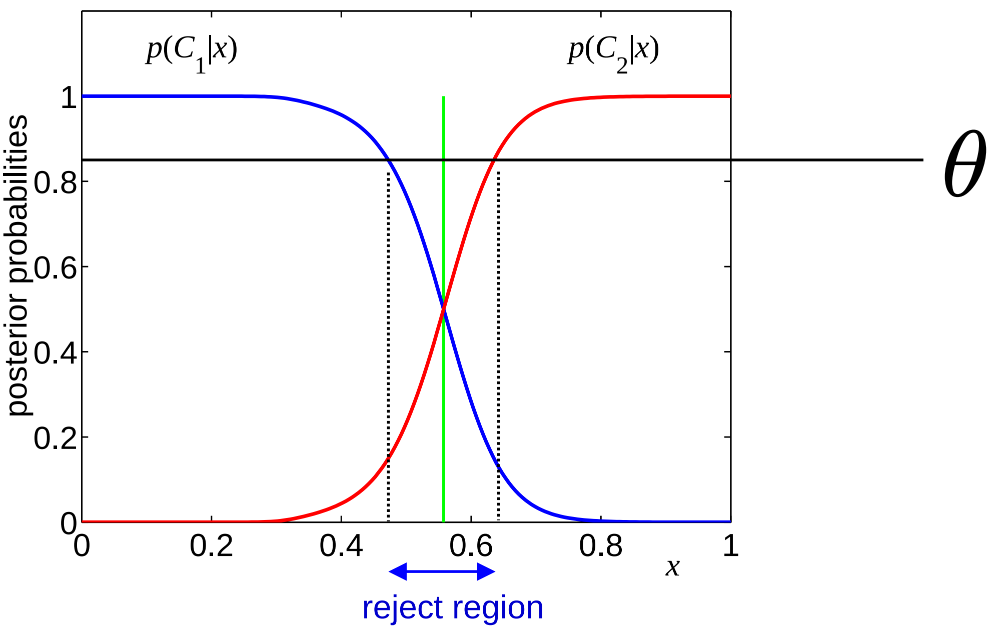
sum to 1

$$p(C_1|x) + p(C_2|x) = 1,$$
$$\text{so } p(C_2|x) = 1 - p(C_1|x)$$

i.e. class i if $p(C_i|x) > 0.5$

Reject option

avoid making decisions if unsure



reject if posterior probability $p(C_i|x) < \theta$

Example – skin detection in video

Objective: label skin pixels (as a means to detect humans)

Two stages:

1. Training: learn likelihood for pixel colour, given skin and non-skin pixels
2. Testing: classify a new image into skin regions

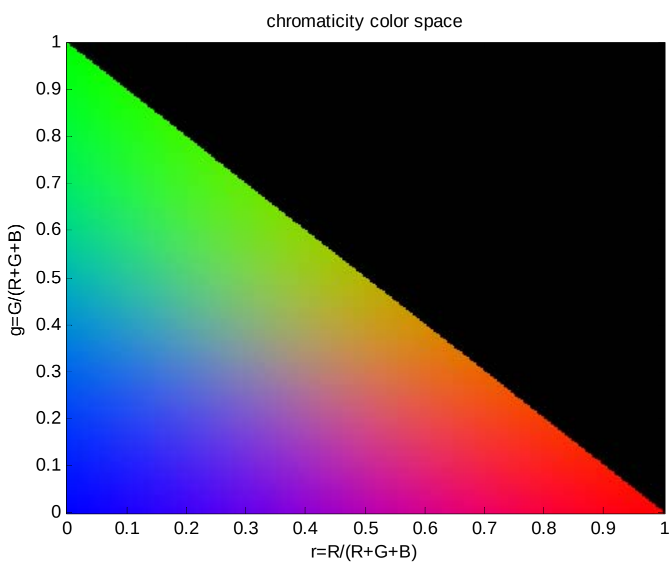
training image

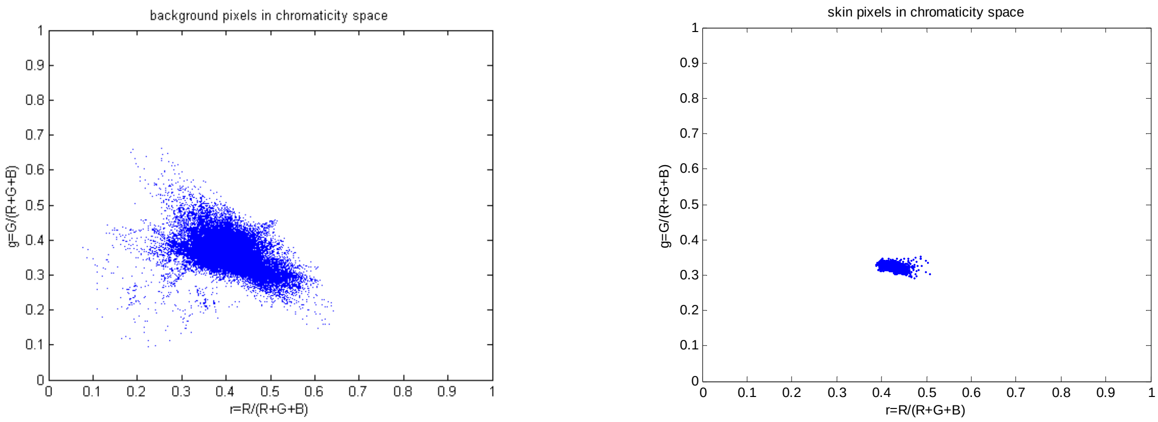
training skin pixel mask

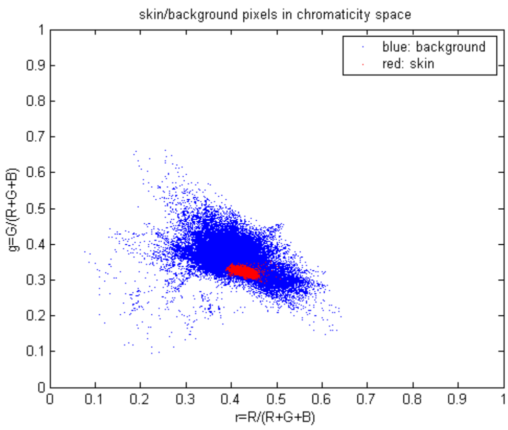
masked pixels

Choice of colour space

- chromaticity color space: r=R/(R+G+B), g=G/(R+G+B)
- invariant to scaling of R,G,B, plus 2D for visualisation

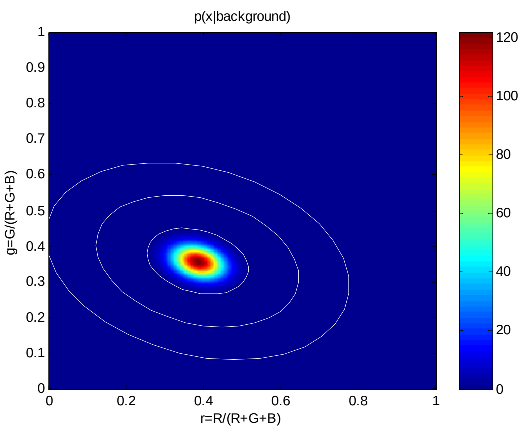




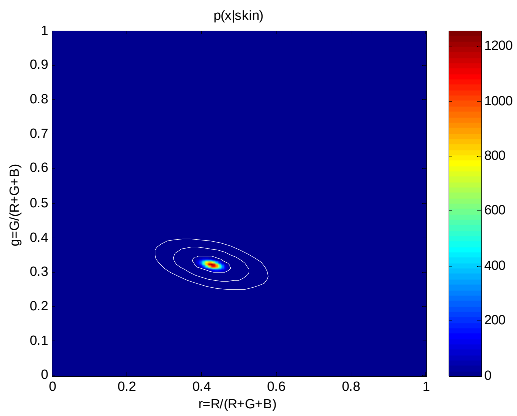


Represent likelihood as Normal Distribution

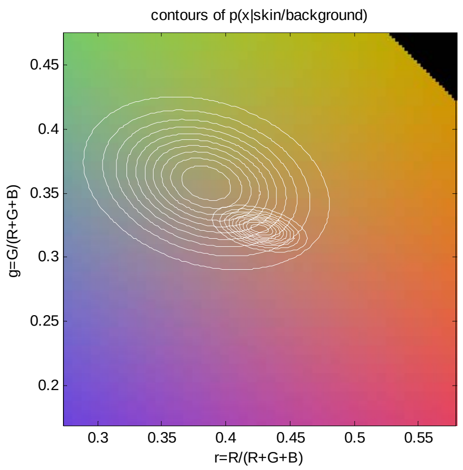
$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^\top\Sigma^{-1}(\mathbf{x}-\mu)\right\}$$



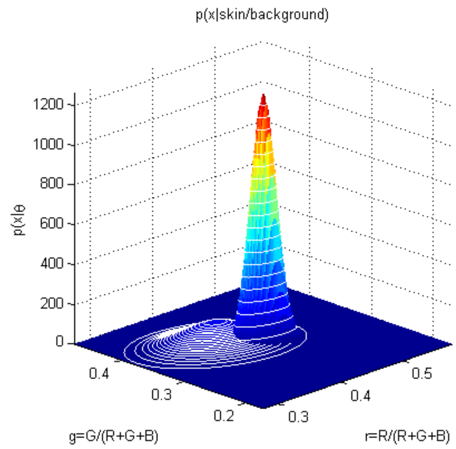
Gaussian fitted to background pixels



Gaussian fitted to skin pixels



contours of two Gaussians



3D view of two Gaussians
vertical axis is likelihood

Posterior probability of skin given pixel colour

Posterior probability of skin is defined by Bayes rule:

$$P(skin|\mathbf{x}) = \frac{p(\mathbf{x}|skin)P(skin)}{p(x)}$$

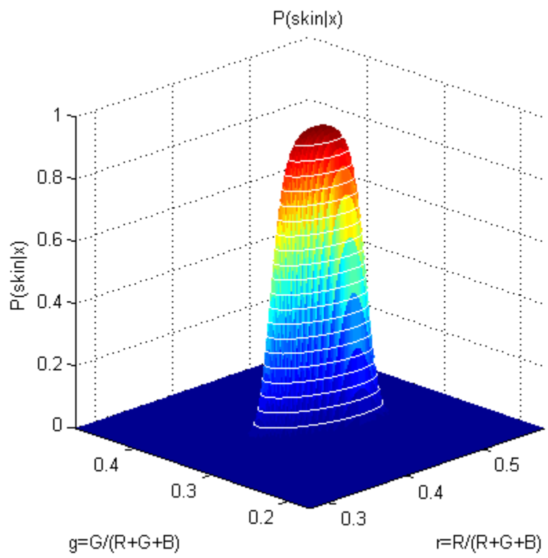
where

$$p(\mathbf{x}) = p(\mathbf{x}|skin)P(skin) + p(\mathbf{x}|background)P(background)$$

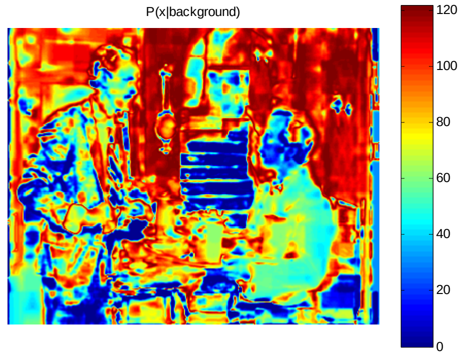
i.e. the marginal pdf of $\mathbf{x}$

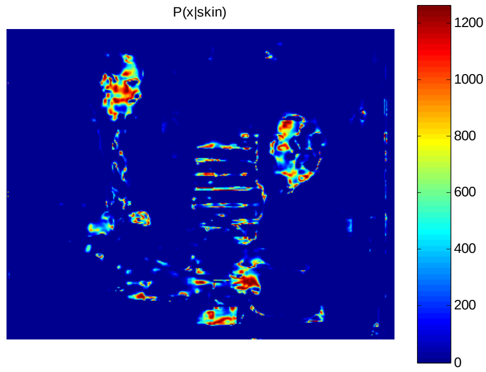
Assume equal prior probabilities, i.e. probability of a pixel being skin is 0.5.

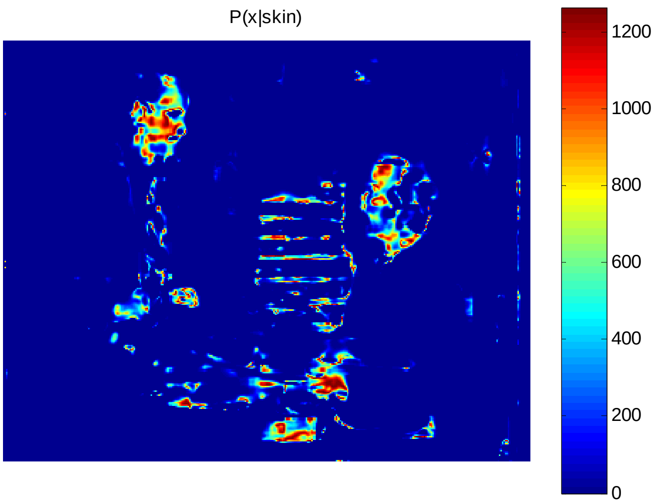
NB: the posterior depends on both foreground and background likelihoods
i.e. it involves both distributions



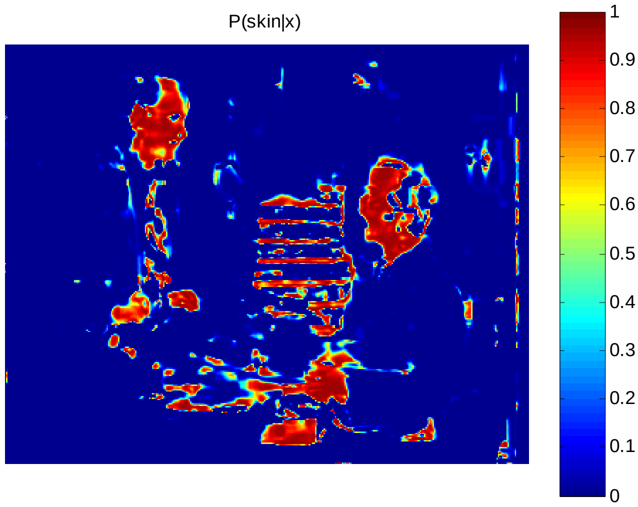
Assess performance on training image





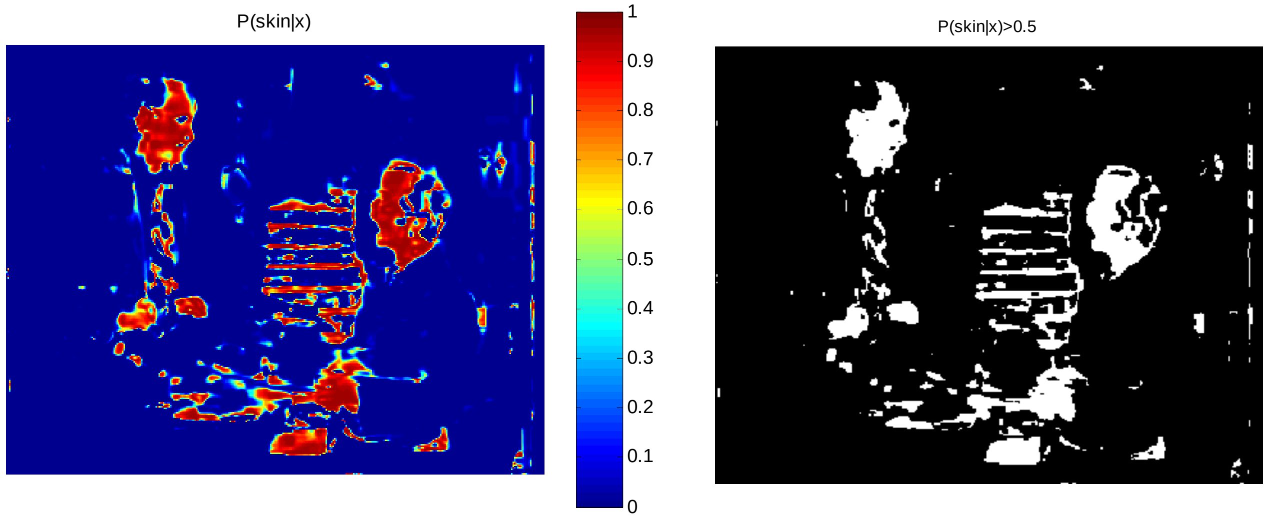


likelihood



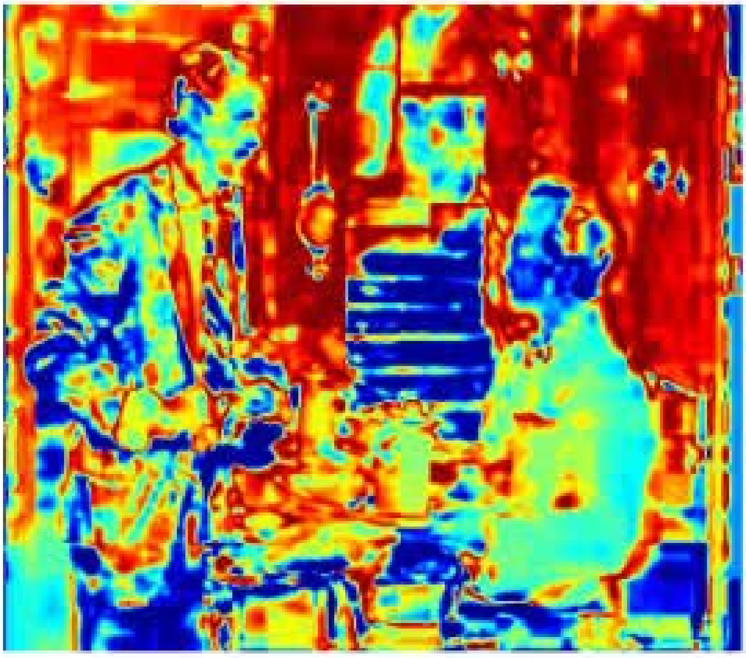
posterior

posterior depends on likelihoods (Gaussians) of both classes

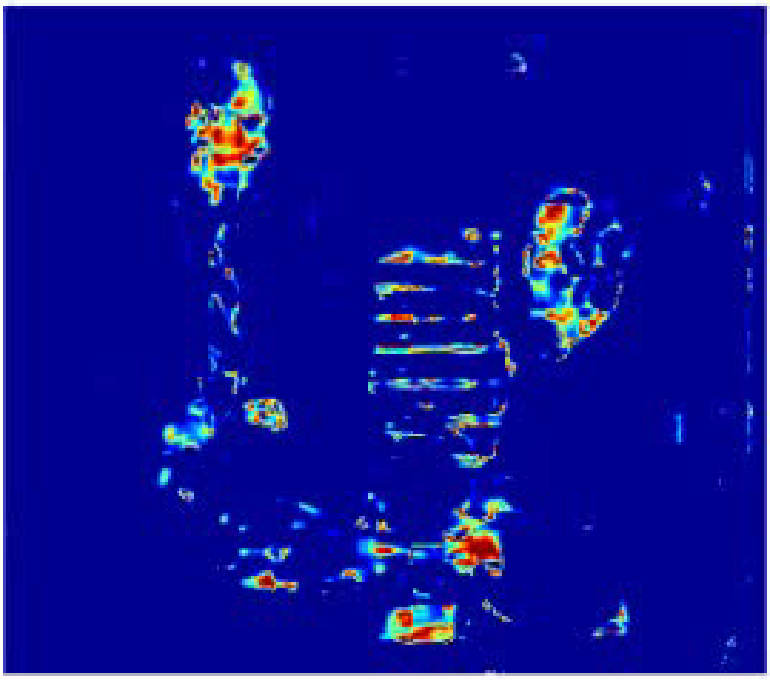


Test data

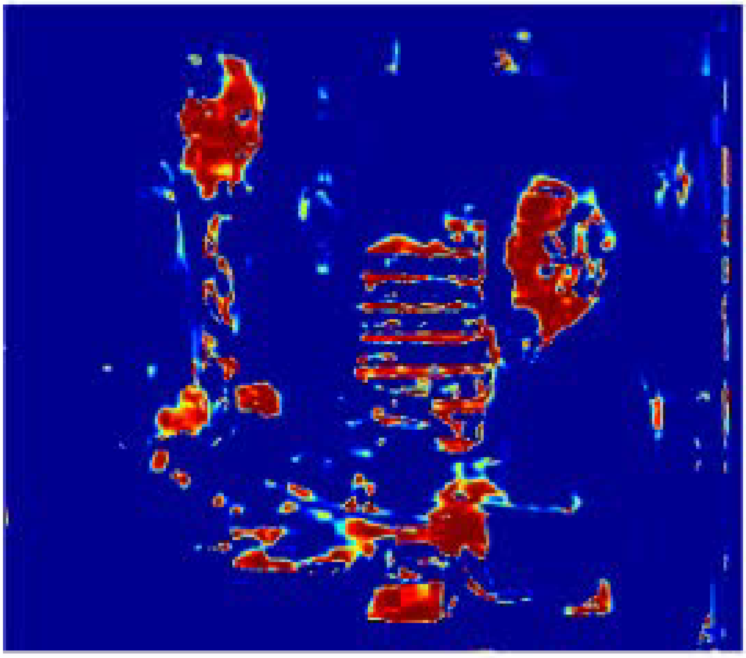
p(x|background)



p(x|skin)

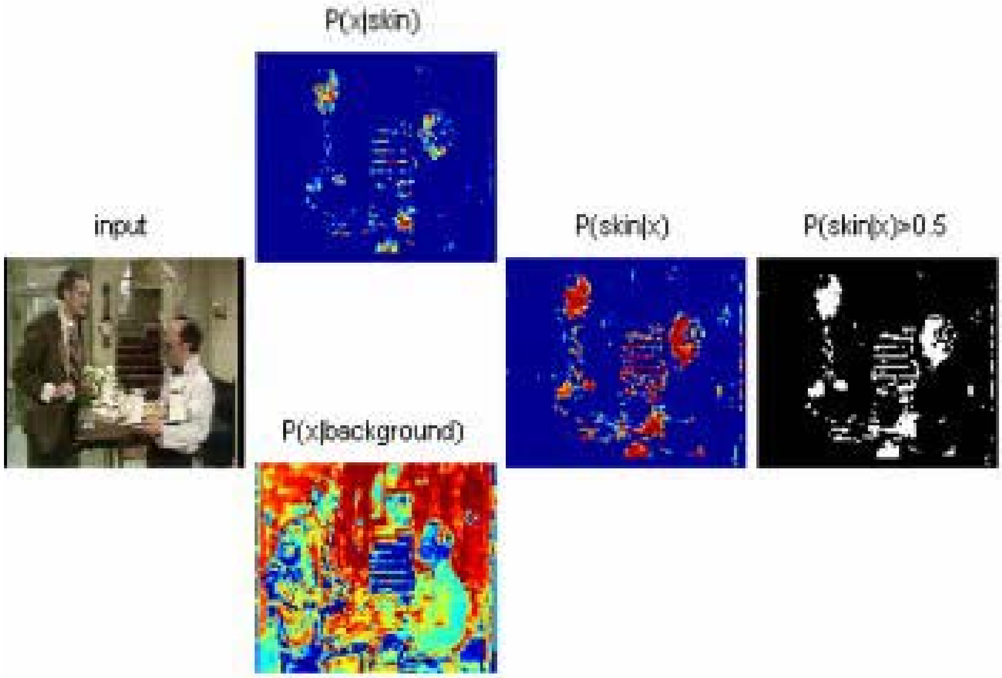


p(skin|x)



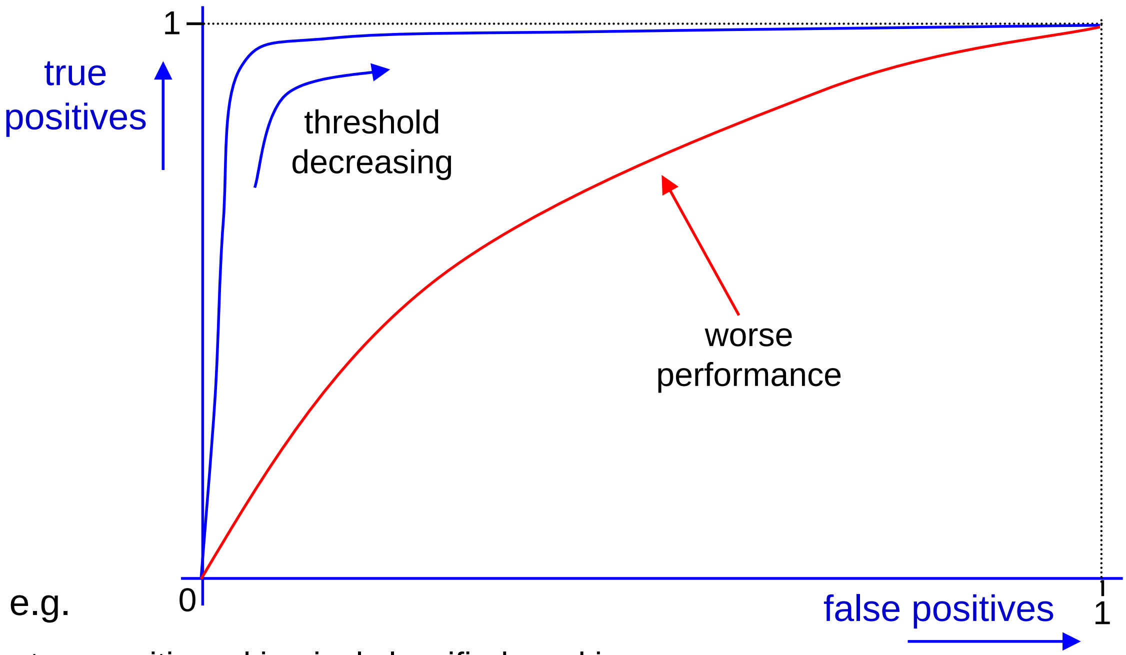
p(skin|x)>0.5

Test performance on other frames



Receiver Operator Characteristic (ROC) Curve

In many algorithms there is a threshold that affects performance



e.g.

• true positive: skin pixel classified as skin

• false positive: background pixel classified as skin

Loss function revisited

Consider again the cancer diagnosis example. The consequences for an incorrect classification vary for the following cases:

- False positive: does not have cancer, but is classified as having it
  > distress, plus unnecessary further investigation

- False negative: does have cancer, but is classified as not having it
  > no treatment, premature death

The two other cases are true positive and true negative.

Because the consequences of a false negative far outweigh the others, rather than simply minimize the number of mistakes, a loss function can be minimized.

Risk $$R(C_i|x) = \sum_j L_{ij} p(C_j|x)$$

Loss matrix
$$L_{ij} = \begin{array}{cc} \text{cancer} & \text{normal} \\ \left[\begin{array}{cc} 0 & 1 \\ 1000 & 0 \end{array}\right] & \end{array} \begin{array}{l} \\ \text{cancer} \\ \text{normal} \end{array}$$

truth

classification

$$\left[\begin{array}{cc} \text{true +ve} & \text{false +ve} \\ \text{false -ve} & \text{true -ve} \end{array}\right]$$

Bayes Risk

The class conditional risk of an action is

measurement

$$R(a_i|x) = \sum_j \underbrace{L(a_i|C_j)}_{\text{loss incurred if action i taken and true state is j}} p(C_j|x)$$

action

Bayes decision rule: select the action for which $R(a_i \mid x)$ is minimum

Mininimize Bayes risk $\hat{a}_i = \arg\min_{a_i} R(a_i|x)$

This decision minimizes the expected loss

Likelihood ratio

Two category classification with loss function

Conditional risk

$$\begin{aligned} R(a_1|x) &= L_{11}p(C_1|x) + L_{12}p(C_2|x) \\ R(a_2|x) &= L_{21}p(C_1|x) + L_{22}p(C_2|x) \end{aligned}$$

Thus for minimum risk, decide C_1 if

$$\begin{aligned} L_{11}p(C_1|x) + L_{12}p(C_2|x) &< L_{21}p(C_1|x) + L_{22}p(C_2|x) \\ p(C_2|x)(L_{12} - L_{22}) &< p(C_1|x)(L_{21} - L_{11}) \\ p(x|C_2)p(C_2)(L_{12} - L_{22}) &< p(x|C_1)p(C_1)(L_{21} - L_{11}) \quad \text{Bayes} \end{aligned}$$

Assuming $L_{21} - L_{11} > 0$, then decide C_1 if

$$\frac{p(x|C_1)}{p(x|C_2)} > \frac{p(C_2)(L_{22} - L_{12})}{p(C_1)(L_{11} - L_{21})}$$

i.e. likelihood ratio exceeds a threshold that is independent of x

Discriminant functions

A two category classifier can often be written in the form

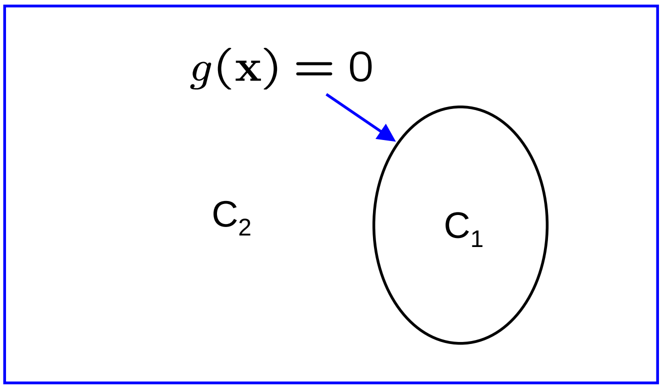
$$g(\mathbf{x}) \begin{cases} > 0 & \text{assign } \mathbf{x} \text{ to } C_1 \\ < 0 & \text{assign } \mathbf{x} \text{ to } C_2 \end{cases}$$

where $g(\mathbf{x})$ is a discriminant function, and

$$g(\mathbf{x}) = 0$$

is a discriminant surface.

In 2D $g(\mathbf{x}) = 0$ is a set of curves.



Posterior probability of skin given pixel colour

Posterior probability of skin is defined by Bayes rule:

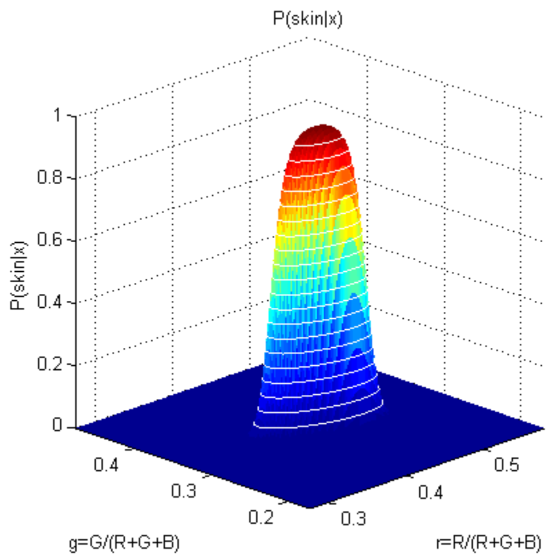
$$P(skin|\mathbf{x}) = \frac{p(\mathbf{x}|skin)P(skin)}{p(x)}$$

where

$$p(\mathbf{x}) = p(\mathbf{x}|skin)P(skin) + p(\mathbf{x}|background)P(background)$$

i.e. the marginal pdf of $\mathbf{x}$

Assume equal prior probabilities, i.e. probability of a pixel being skin is 0.5.



Example

In the minimum average error classifier, the assignment rule is: decide C_1 if the posterior $p(C_1|\mathbf{x}) > p(C_2|\mathbf{x})$.

The equivalent discriminant function is

$$g(\mathbf{x}) = p(C_1|\mathbf{x}) - p(C_2|\mathbf{x})$$

or

$$g(\mathbf{x}) = \ln \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}$$

Note, these two functions are not equal, but the decision boundaries are the same.

Developing this further

$$\begin{aligned} g(\mathbf{x}) &= \ln \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} \\ &= \ln \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \ln \frac{p(C_1)}{p(C_2)} \end{aligned}$$

Decision surfaces for Normal distributions

Suppose that the likelihoods are Normal:

$$p(\mathbf{x}|C_1) \sim N(\boldsymbol{\mu}_1, \Sigma_1) \qquad p(\mathbf{x}|C_2) \sim N(\boldsymbol{\mu}_2, \Sigma_2)$$

Then

$$\begin{aligned} g(\mathbf{x}) &= \ln \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \ln \frac{p(C_1)}{p(C_2)} \\ &= \ln p(\mathbf{x}|C_1) - \ln p(\mathbf{x}|C_2) + \ln \frac{p(C_1)}{p(C_2)} \\ &\sim -(\mathbf{x}-\boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + (\mathbf{x}-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + c' \end{aligned}$$

where $c' = \ln \frac{p(C_1)}{p(C_2)} - \frac{1}{2}\ln|\Sigma_1| + \frac{1}{2}\ln|\Sigma_2|$.

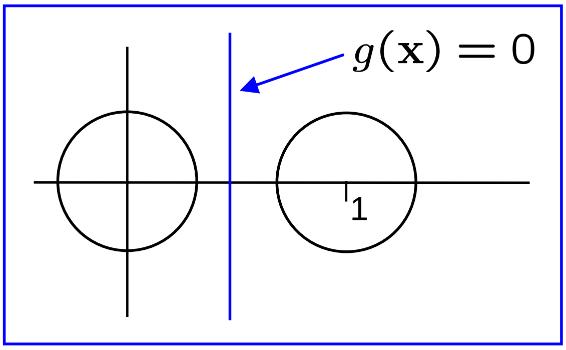
Case 1: $\Sigma_i = \sigma^2 I$

$$g(\mathbf{x}) \;=\; -(\mathbf{x}-\mu_1)^\top(\mathbf{x}-\mu_1) + (\mathbf{x}-\mu_2)^\top(\mathbf{x}-\mu_2) + 2\sigma^2 c$$

Example in 2D

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{\Sigma}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

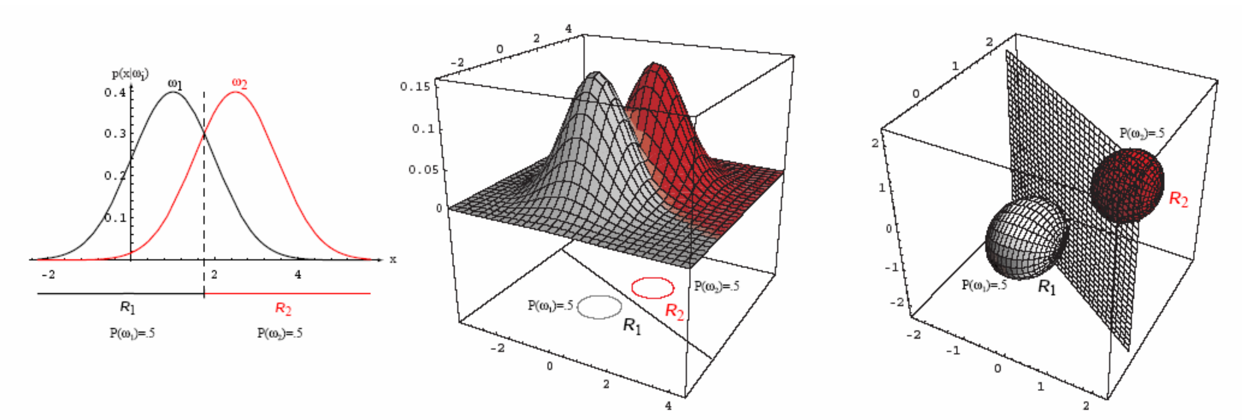
$$\begin{aligned} g(\mathbf{x}) &= -(x^2+y^2) + (x-1)^2 + y^2 + c \\ &= -2x + c + 1 \end{aligned}$$



This is a line at x = (c+1)/2

- if the priors are equal then c = 0
- in nD the discriminant surface is a hyperplane

$$(\mu_2 - \mu_1)\,.\mathbf{x} = c''$$

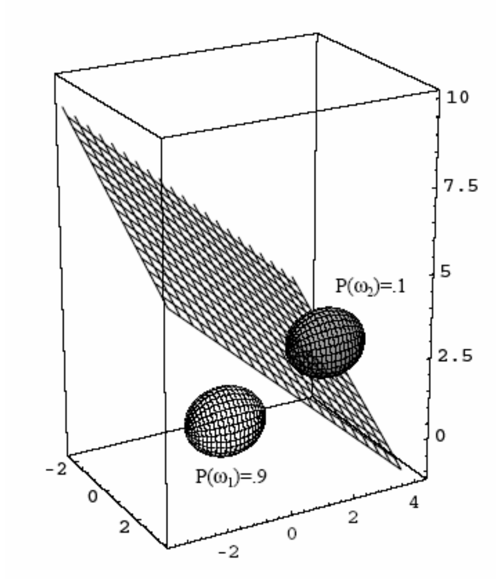


Case 2: $\Sigma_i = \Sigma$ (covariance matrices are equal)

The discriminant surface

$$g(\mathbf{x}) = -(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1) + (\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2) + c'$$

is also a hyperplane. Why?



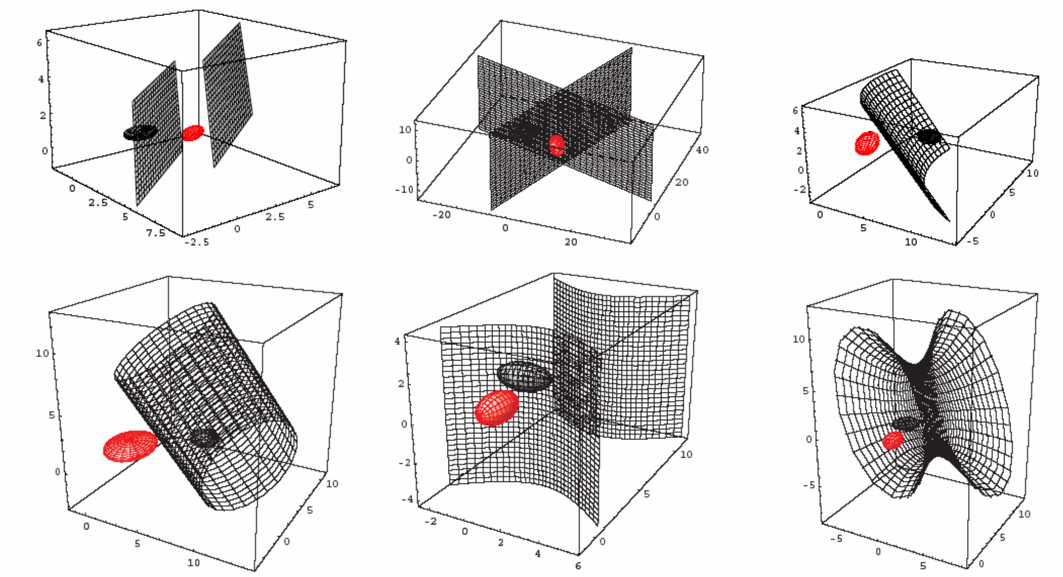
Case 3: $\Sigma_i =$ arbitrary

The discriminant surface

$$g(\mathbf{x}) = -(\mathbf{x}-\mu_1)^\top \Sigma_1^{-1}(\mathbf{x}-\mu_1) + (\mathbf{x}-\mu_2)^\top \Sigma_2^{-1}(\mathbf{x}-\mu_2) + c'$$

is a conic (2D) or quadric (nD).

e.g. in 3D



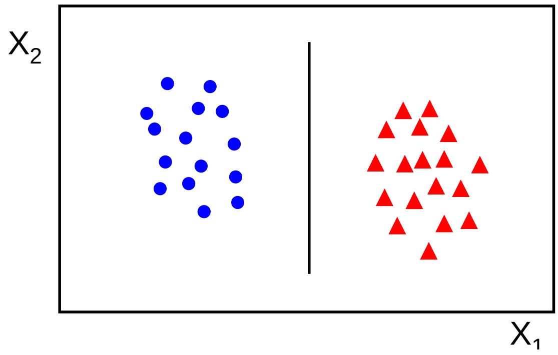
The surface can be a hyperboloid, i.e. it need not be closed

Discriminative Methods

So far, we have carried out the following steps in order to compute a discriminant surface:

1. Measure feature vectors (e.g. in 2D for skin colour) for each class from training data
2. Learn likelihood pdfs for each class (and priors)
3. Represent likelihoods by fitting Gaussians
4. Compute the posteriors $p(C_i \mid x)$
5. Compute the discriminant surface (from the likelihood Gaussians)
6. In 2D the curve is a conic …

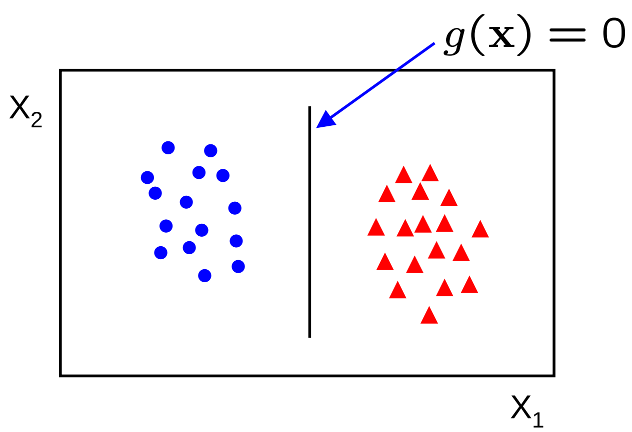
Why not fit the discriminant curve to the data directly?



Linear classifiers

A linear discriminant has the form

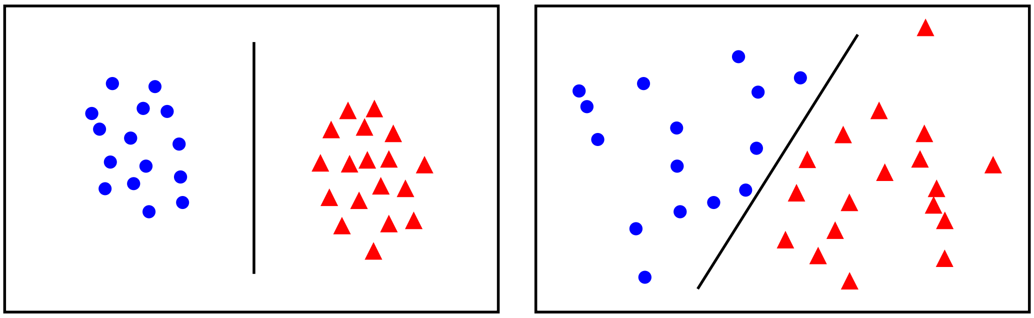
$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$



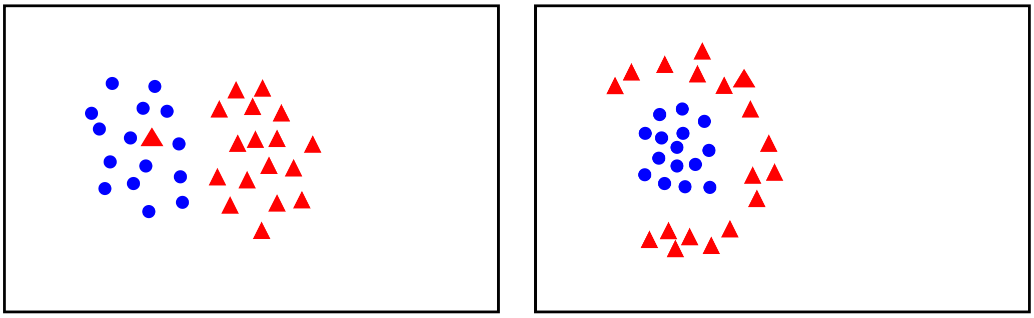
- in 2D a linear discriminant is a line, in nD it is a hyperplane
- $\mathbf{w}$ is the normal to the plane, and w_0 the bias
- $\mathbf{w}$ is known as the weight vector

Linear separability

linearly separable



not linearly separable



Learning separating hyperplanes

Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{0,1\}$, find a weight vector **w** such that the discriminant function

$$g(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + w_0$$

separates the categories for i = 1,n

- how can we find this separating hyperplane ?

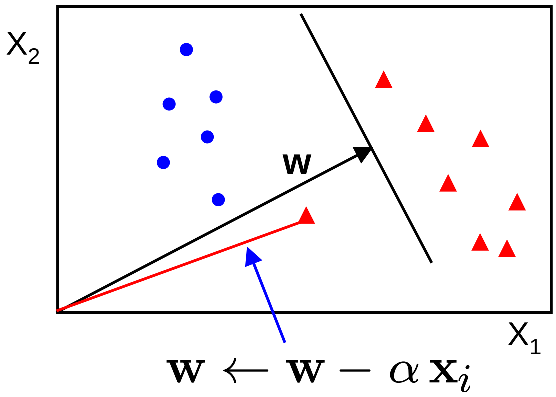
The Perceptron Algorithm

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha\, \mathrm{sign}(g(\mathbf{x}_i))\, \mathbf{x}_i$
- Until all the data is correctly classified

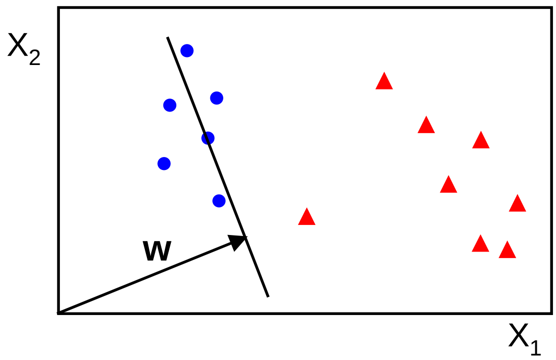
For example in 2D

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha\, \mathsf{sign}(g(\mathbf{x}_i))\, \mathbf{x}_i$
- Until all the data is correctly classified

before update

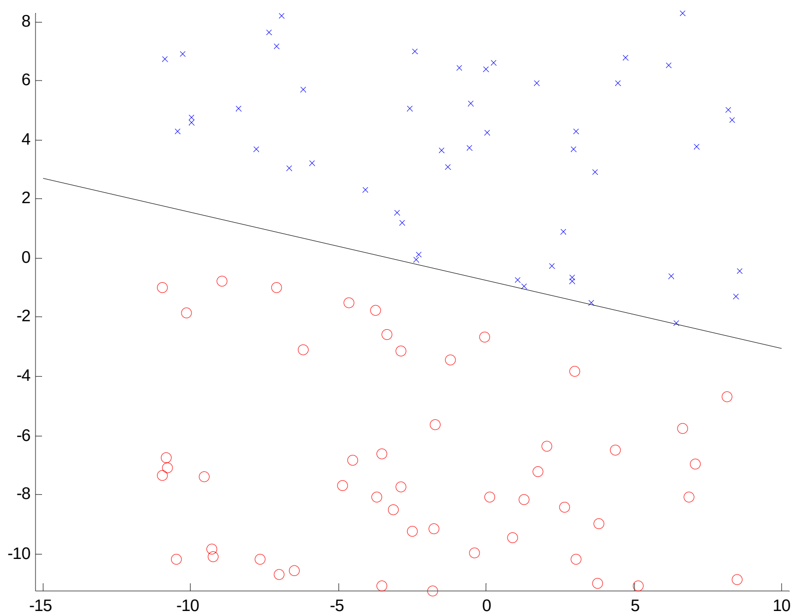


after update



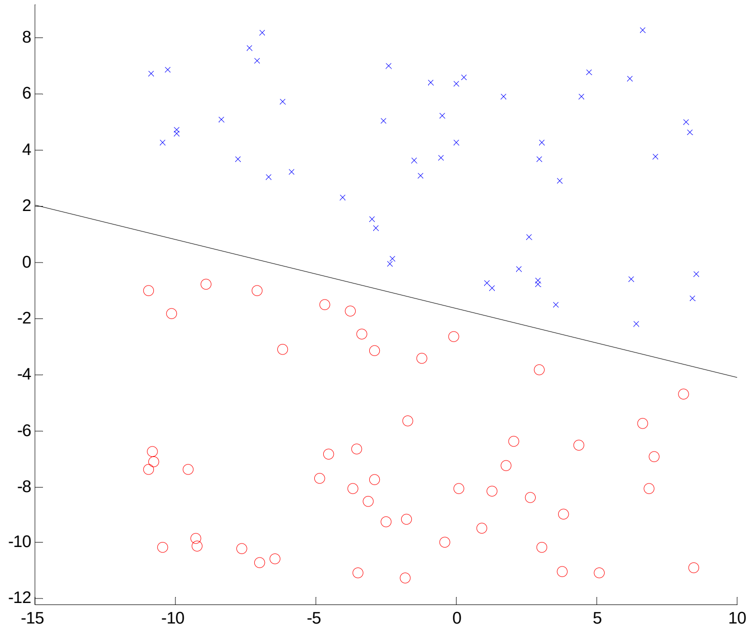
NB after convergence $\mathbf{w} = \sum_i^n \alpha_i \mathbf{x}_i$

Perceptron example



- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization

wider margin
classifier



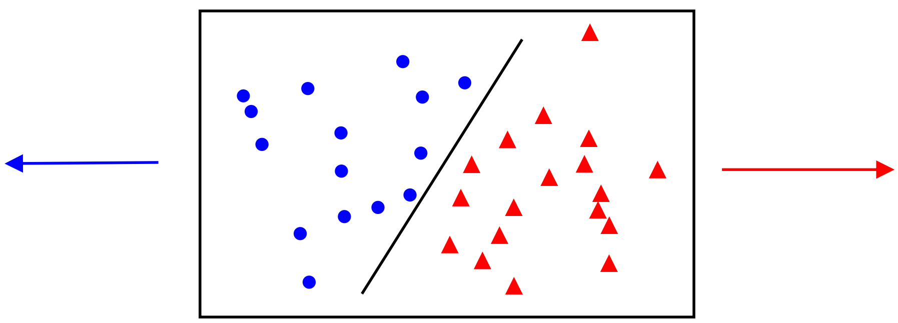
• how to achieve a wider margin automatically in high dimensions ?

Logistic Regression

• ideally we would like to fit a discriminant function using regression methods similar to those developed for ML and MAP parameter estimation

• but there is not the equivalent of model + noise here, since we wish to map all the spread out features in the same class to one label

to $-\infty$



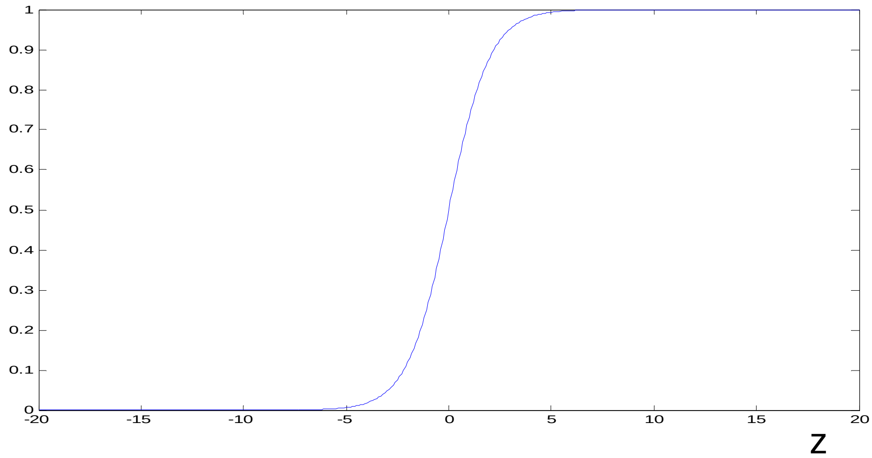
to $+\infty$

• the solution is to transform the parameter space so that

$$(-\infty, \infty) \to (0, 1)$$

The logistic function or sigmoid function

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



Notation: write the equation $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$

more compactly as $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$

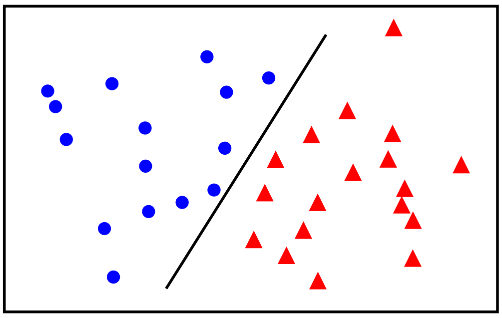
- e.g. in 2D

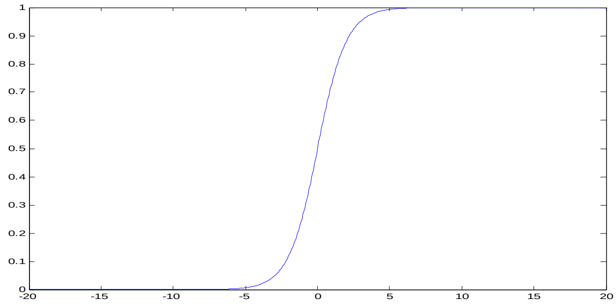
$$g(\mathbf{x}) = \begin{pmatrix} w_2 & w_1 & w_0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

In logistic regression fit a sigmoid function

$$\sigma(\mathbf{w}^t \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^t \mathbf{x}}}$$

to the data { $\mathbf{x}_i$, y_i } by minimizing the classification errors $y_i - \sigma(\mathbf{w}^t \mathbf{x}_i)$





Maximum Likelihood Estimation

Assume

$$\begin{aligned} p(y=1|\mathbf{x};\mathbf{w}) &= \sigma(\mathbf{w}^\top \mathbf{x}) \\ p(y=0|\mathbf{x};\mathbf{w}) &= 1-\sigma(\mathbf{w}^\top \mathbf{x}) \end{aligned}$$

write this more compactly as

$$p(y|\mathbf{x};\mathbf{w}) = \left(\sigma(\mathbf{w}^\top \mathbf{x})\right)^y \left(1-\sigma(\mathbf{w}^\top \mathbf{x})\right)^{(1-y)}$$

Then the likelihood (assuming independence) is

$$p(\mathbf{y}|\mathbf{x};\mathbf{w}) \sim \prod_i^n \left(\sigma(\mathbf{w}^\top \mathbf{x}_i)\right)^{y_i} \left(1-\sigma(\mathbf{w}^\top \mathbf{x}_i)\right)^{(1-y_i)}$$

and the negative log likelihood is

$$L(\mathbf{w}) = -\sum_i^n y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1-y_i)\log(1-\sigma(\mathbf{w}^\top \mathbf{x}_i))$$

Minimize $L(\mathbf{w})$ using gradient descent [exercise]

$$\frac{\partial}{\partial w_j} L(\mathbf{w}) = -\sum_i \left(y_i - \sigma(\mathbf{w}^\top \mathbf{x}_i)\right) x_j$$

which gives the update rule

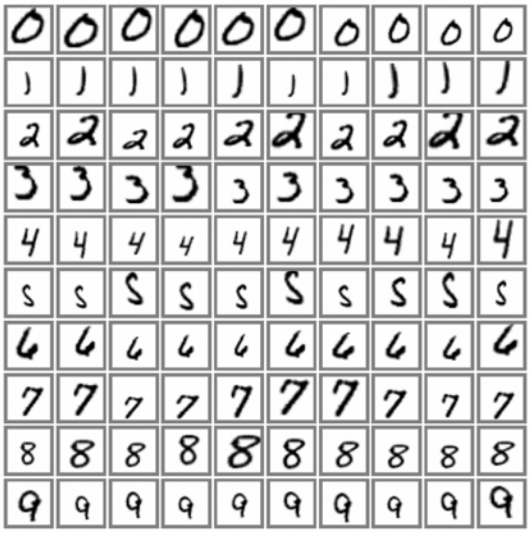
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(\sigma(\mathbf{w}^\top \mathbf{x}_i) - y_i)\mathbf{x}_i$$

Note

- this is similar, but not identical, to the perceptron update rule.
- there is a unique solution for $\mathbf{w}$
- in n-dimensions it is only necessary to learn n+1 parameters. Compare this with learning normal distributions where learning involves 2n parameters for the means and n(n+1)/2 for a common covariance matrix

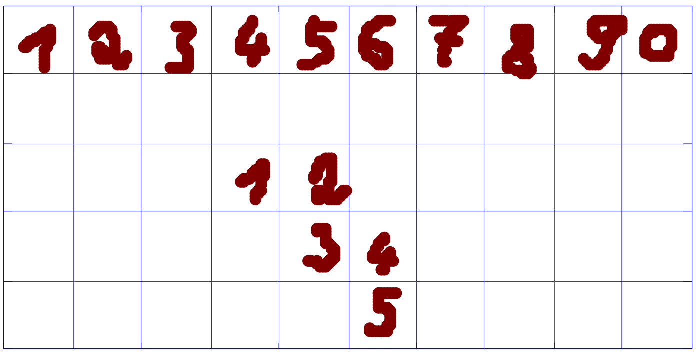
Application: hand written digit recognition

- Feature vectors: each image is 28 x 28 pixels. Rearrange as a 784-vector
- Training: learn a set of two-class linear classifiers using logistic regression, e.g.
    - 1 against the rest, or
    - (0-4) vs (5-9) etc
- An alternative is to learn a multi-class classifier, e.g. using k-nearest neighbours



Example

hand drawn



classification

1	2	3	4	5	6	7	8	9	0
			1	2					
				3	4				
					5				

Comparison of discriminant and generative approaches

Discriminant

+ don't have to learn parameters which aren't used (e.g. covariance)
+ easy to learn
- no confidence measure
- have to retrain if dimension of feature vectors changed

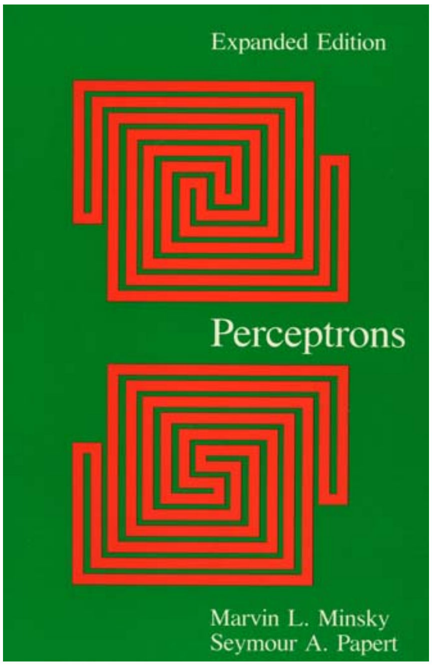
Generative

+ have confidence measure
+ can use 'reject option'
+ easy to add independent measurements

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}_\mathsf{A},\mathbf{x}_\mathsf{B}) &\propto p(\mathbf{x}_\mathsf{A},\mathbf{x}_\mathsf{B}|\mathcal{C}_k)\,p(\mathcal{C}_k) \\ &\propto p(\mathbf{x}_\mathsf{A}|\mathcal{C}_k)\,p(\mathbf{x}_\mathsf{B}|\mathcal{C}_k)\,p(\mathcal{C}_k) \\ &\propto \frac{p(\mathcal{C}_k|\mathbf{x}_\mathsf{A})\,p(\mathcal{C}_k|\mathbf{x}_\mathsf{B})}{p(\mathcal{C}_k)} \end{aligned}$$

- expensive to train (because many parameters)

Perceptrons (1969)



Recent progress in Machine Learning

Perceptron Non-examinable

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \text{ where } \mathbf{w} = \textstyle\sum_i^n \alpha_i \mathbf{x}_i$$

$$g(\mathbf{x}) = \sum_i \alpha_i {\mathbf{x}_i}^\top \mathbf{x}$$

Generalize to

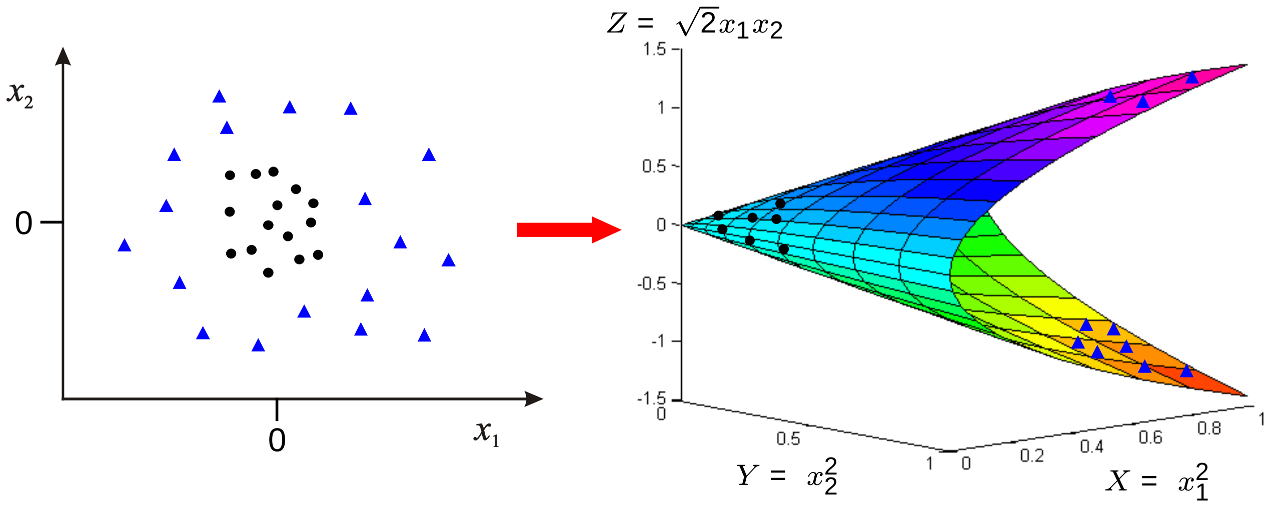
$$g(\mathbf{x}) = \sum_i \alpha_i \phi(\mathbf{x}_i)^t \phi(\mathbf{x})$$

where $\phi(\mathbf{x})$ is a map from $\mathbf{x}$ to a higher dimension.
For example, for $\mathbf{x} = (x_1, x_2)^t$

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)^t$$

Example

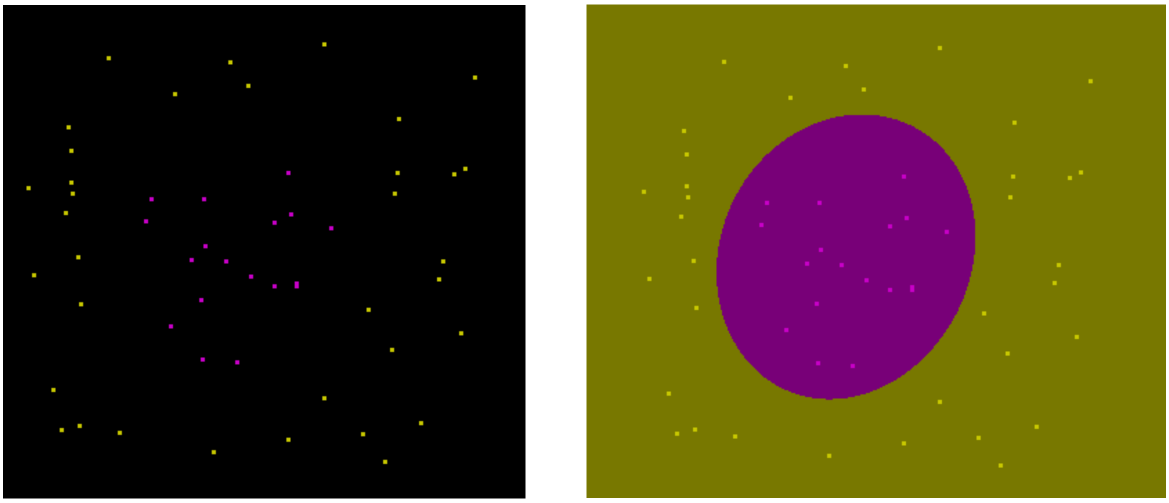
$$\phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)^t$$



Data is linearly separable in 3D

This means that the problem can still be solved by a linear classifier

Example



Kernels

Generalize further to

$$g(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

where $K(\mathbf{x}, \mathbf{z})$ is a (non-linear) Kernel function.
For example

$$K(\mathbf{x}, \mathbf{z}) \sim \exp - \left\{ (\mathbf{x} - \mathbf{z})^2 / (2\sigma^2) \right\}$$

is a radial basis function kernel, and

$$K(\mathbf{x}, \mathbf{z}) \sim (\mathbf{x}.\mathbf{z})^n$$

is a polynomial kernel.

Exercise

If $n = 2$ show that

$$K(\mathbf{x}, \mathbf{z}) \sim (\mathbf{x}.\mathbf{z})^2 = \phi(\mathbf{x})^t \phi(\mathbf{z})$$