Learning in linear systems

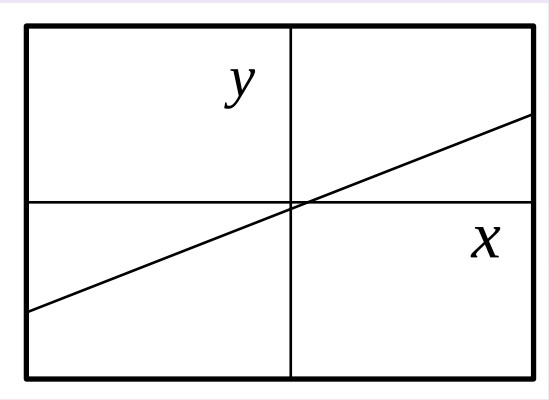
Antanas Verikas
antanas.verikas@hh.se

IDE, Halmstad University

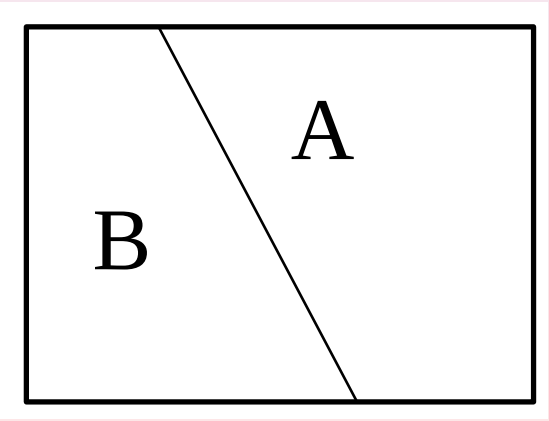
2013

What is a linear system?

In regression:
$y(\mathbf{x})$ is a linear function of $\mathbf{x}$.



In classification:
Separating boundaries are linear.



Assumptions and goal

We assume linear process

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

We use a linear model family Φ

$$\widehat{y}(\mathbf{x}) = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{T}\mathbf{x}$$

and the goal is to find $\mathbf{w} = \mathbf{w}^{*}$.

Training set

We have a training set Z:

$$Z = \{\mathbf{x}(n), y(n)\}_{n=1,\ldots,N}$$

which can be written as a matrix and a vector

$$\mathbf{X} = \begin{pmatrix} 1 & x_1(1) & \cdots & x_d(1) \\ 1 & x_1(2) & \cdots & x_d(2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1(N) & \cdots & x_d(N) \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{pmatrix}$$

Error function

$$\begin{aligned} \texttt{SSE} &= \sum_{n=1}^{N} [y(n) - \widehat{y}(n)]^2 = \dots \\ &= \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{w}^T \mathbf{X}^T \mathbf{y} + \mathbf{y}^T \mathbf{y} \end{aligned}$$

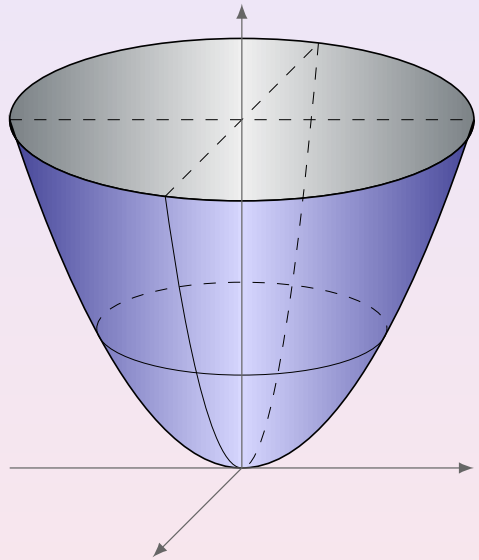
Minimizing SSE gives:

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{X}^\dagger \mathbf{y}$$

where $\mathbf{X}^\dagger$ is the pseudo-inverse of $\mathbf{X}$.

Error surface

The error surface is of the following form



- In batch mode, the optimal and unique weights $\mathbf{w}^*$ can be found by inverting a matrix.
- Batch mode means considering all data in the training set at once.

Ridge regression

If $(\mathbf{X}^T\mathbf{X})$ is almost singular, the following approach is often used:

$$\begin{aligned} \mathbf{X}^T\mathbf{X} &\rightarrow (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}) \\ \Rightarrow \mathbf{w} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

The approach corresponds to using the Bayesian error measure with

$$\ln p(\mathbf{w}) \sim -||\mathbf{w}||^2$$

Using gradient descent

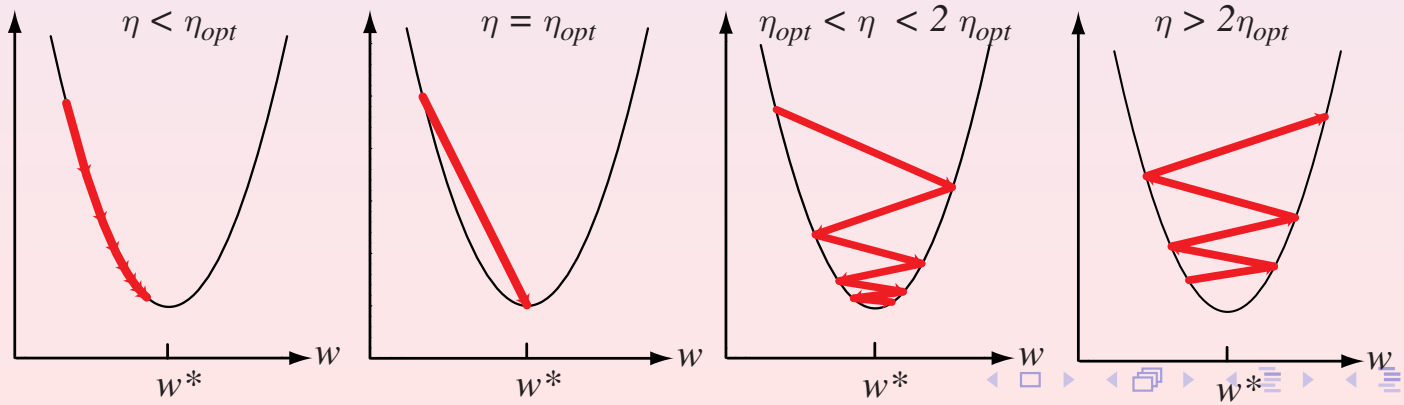
Parameters can be determined by using gradient descent

$$\Delta \mathbf{w} = -\eta \nabla_w E = 2\eta \sum_{n=1}^{N} e(n)\mathbf{x}(n)$$

where

$$e(n) = y(n) - \widehat{y}(n)$$

The influence of η assuming the error function can be approximated by a quadratic:



"On-line" gradient descent

Least Mean Squared (LMS) algorithm

$$\Delta \mathbf{w}(n) = \eta e(n)\mathbf{x}(n)$$

The algorithm is used in adaptive controllers, adaptive filters.
Convergence requires:

$$0 < \eta < \frac{2}{\mathtt{Trace}[\mathbf{X}^T\mathbf{X}]}$$

ADALINE (ADAptive LINear Element)

Designed for classification problems

$$y(\mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{x} \in \texttt{class 1} \\ -1, & \text{if } \mathbf{x} \in \texttt{class 2} \end{cases}$$

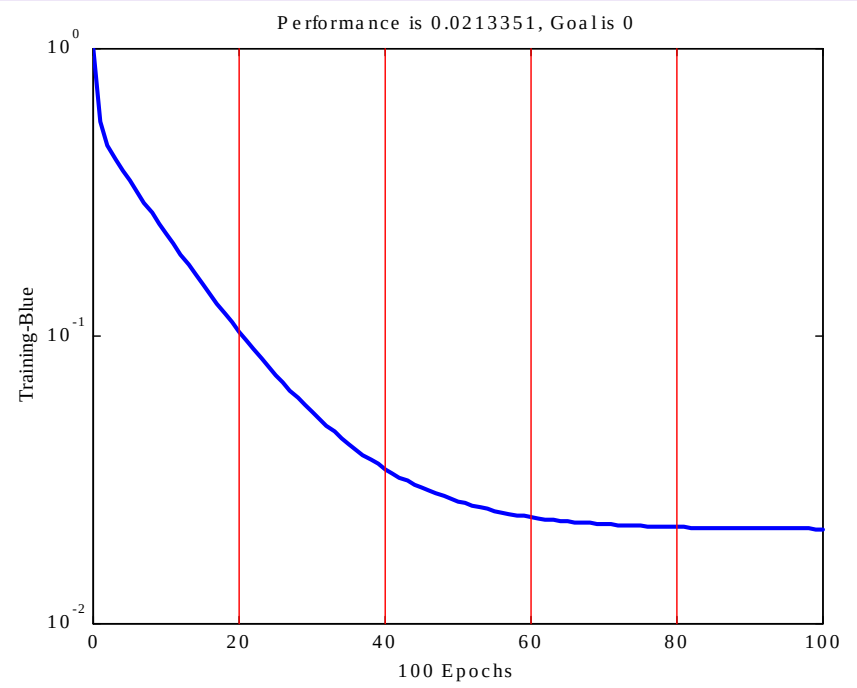
ADALINE output

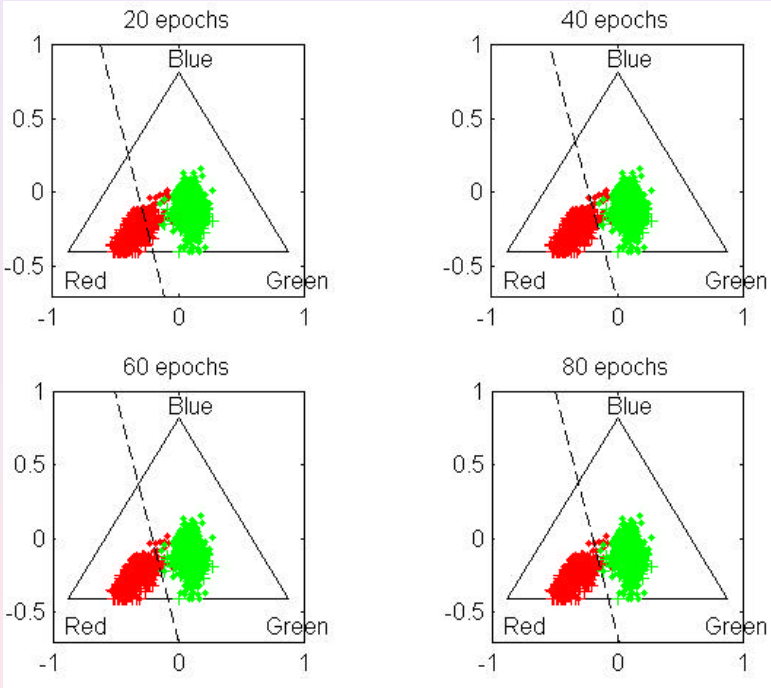
$$\widehat{y}(\mathbf{x}) = \texttt{sign}[\mathbf{w}^T \mathbf{x}]$$

ADALINE is trained using the LMS algorithm.

ADALINE training

Training ADALINE to separate red LEGO pieces from green carpet.

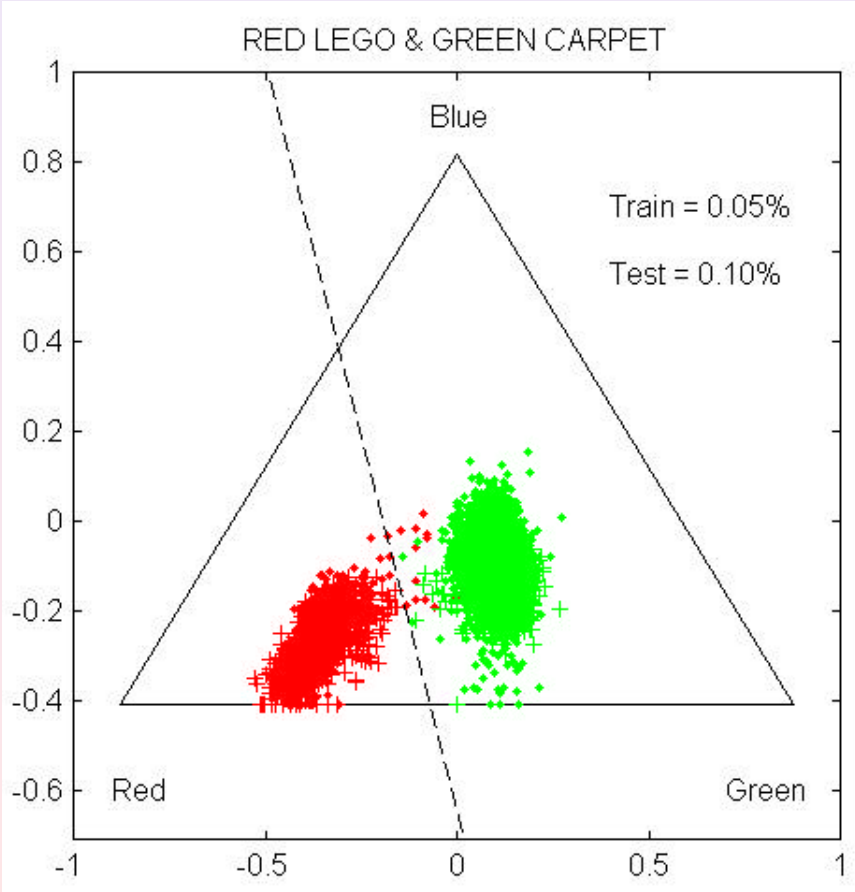




1 *epoch* = one presentation of all the training data.
Classification error after 80 epochs: Train = 0.05%, Test = 0.09%

ADALINE training

Final decision boundary after 100 epochs; Train = 0.05%, Test = 0.10%



Definition

Uses $\{-1, +1\}$ representation:

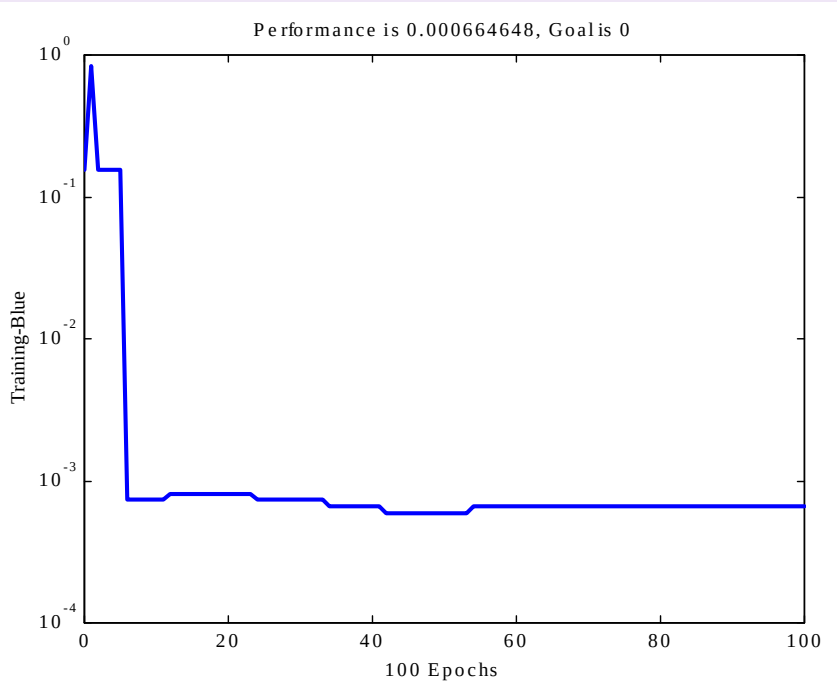
$$\widehat{y}(\mathbf{x}) = \phi[\mathbf{w}^T\mathbf{x}] = \begin{cases} +1, & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1, & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

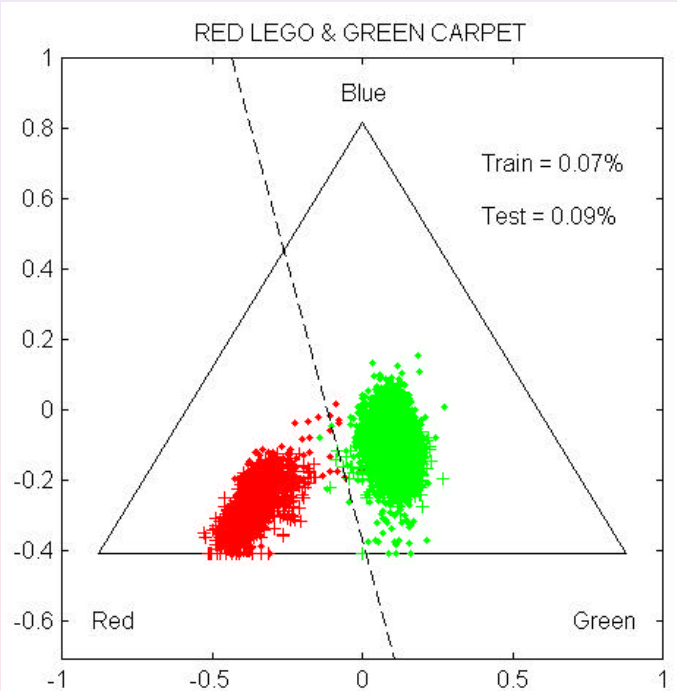
where $\phi(z)$ is called the *Heaviside* function.

Traditionally is trained with the *Perceptron learning* algorithm, which is *not a very stable* algorithm.

Perceptron training and final decision boundary

Training perceptron (using Perceptron learning) to separate red LEGO pieces from green carpet.





Works well here, since the problem is almost linearly separable.
Classification error after 100 epochs: Train = 0.07%, Test = 0.09%

Training by gradient descent

Uses a differentiable output function:

$$\widehat{y}(\mathbf{x}) = \phi(\mathbf{w}^T\mathbf{x})$$

for example:

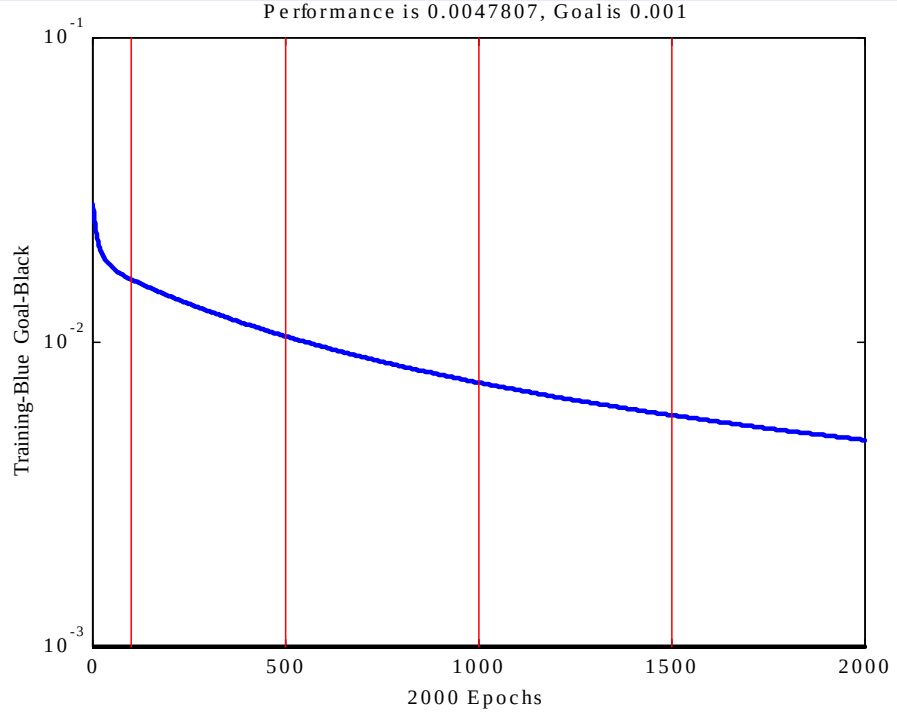
$$\phi(z) = \frac{1}{1+\exp(-z)}$$

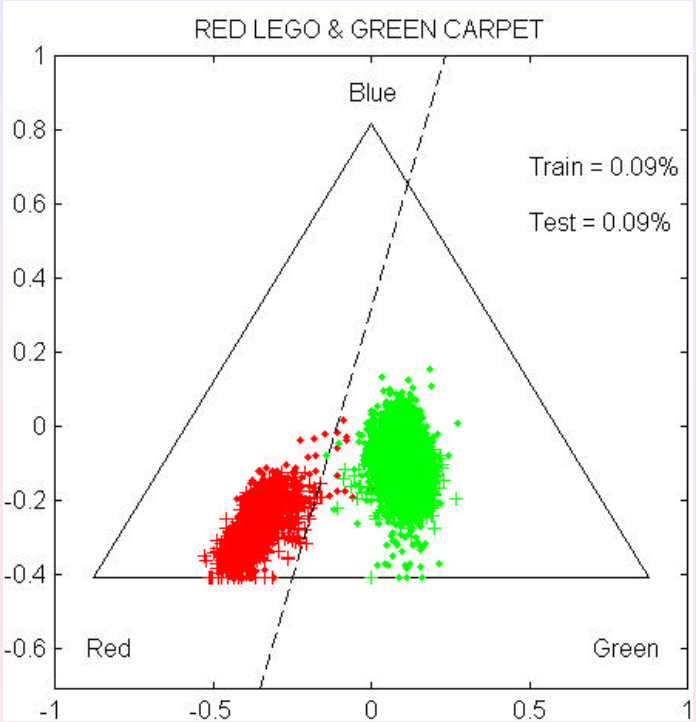
and update

$$\Delta w = -\eta\nabla_w E = \eta 2\sum_{n=1}^{N} e(n)\phi'(\mathbf{w}^T\mathbf{x}(n))\mathbf{x}(n)$$

Gradient descent training and decision boundary

Training to separate red LEGO pieces from green carpet.





After 2000 epochs (goes on forever).
Classification error: Train = 0.09%, Test = 0.09%

Common covariance matrix

Assume Gaussian $p(\mathbf{x}|c)$ with different means $\boldsymbol{\mu}$ and common $\boldsymbol{\Sigma}$.
A new observation $\mathbf{x}$ is classified by computing the probability.

$$p(c_k|\mathbf{x}) = \frac{p(c_k)p(\mathbf{x}|c_k)}{p(\mathbf{x})} \quad p(c_k) = \frac{N_k}{N} = \text{a priori probability}$$

The decision boundary between c_k and c_j is given by

$$\begin{aligned} p(c_k|\mathbf{x}) &= p(c_j|\mathbf{x}) \Rightarrow \\ \ln\left[\frac{p(c_k)}{p(c_j)}\right] &= \frac{(\mathbf{x}-\widehat{\boldsymbol{\mu}}_k)^T\widehat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\widehat{\boldsymbol{\mu}}_k)}{2} - \frac{(\mathbf{x}-\widehat{\boldsymbol{\mu}}_j)^T\widehat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\widehat{\boldsymbol{\mu}}_j)}{2} \end{aligned}$$

and defines a hyperplane, i.e. the classifier is linear.

Maximum likelihood estimates

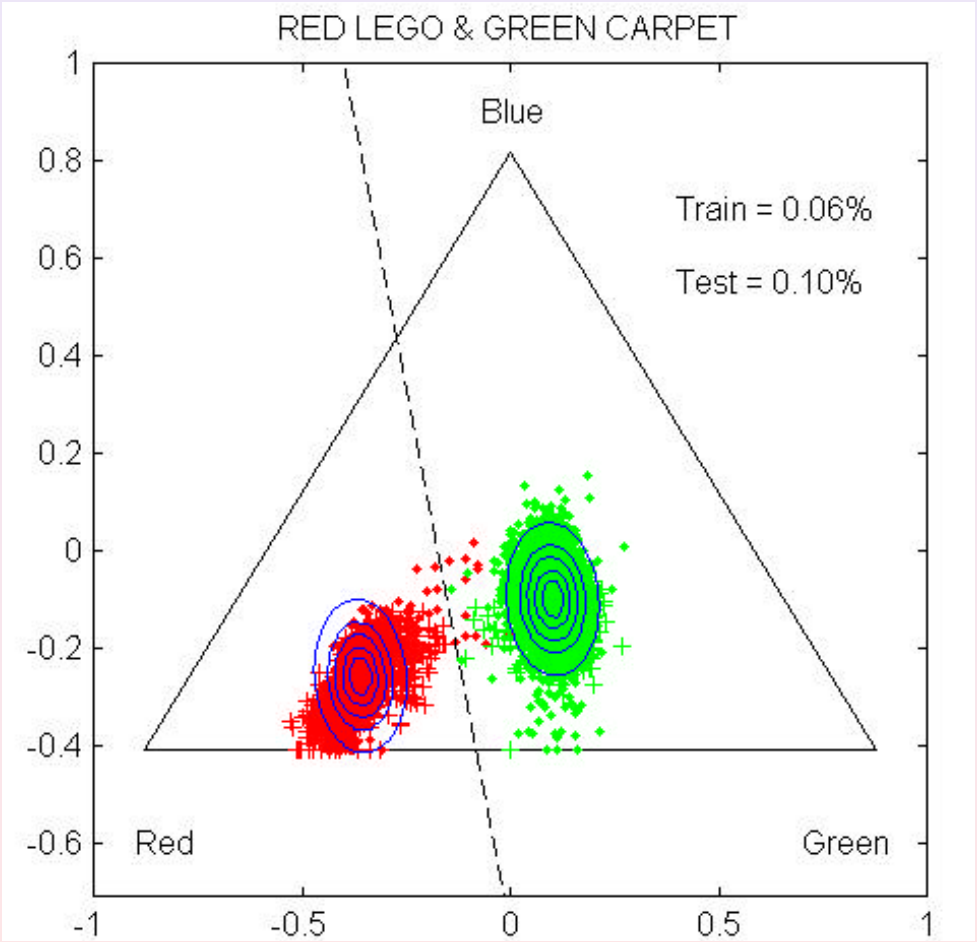
Maximizing data likelihood gives:

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{\mathbf{x}(n) \in C_k} \mathbf{x}(n)$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N-1} \sum_{n=1}^{N} [\mathbf{x}(n) - \widehat{\boldsymbol{\mu}}][\mathbf{x}(n) - \widehat{\boldsymbol{\mu}}]^T$$

Linear Gaussian decision boundary

Training error = 0.06%, Test error = 0.10%



Equal a priori probabilities and variances

If $p(c_k) = p(c_j)$, the decision boundary becomes:

$$(\mathbf{x} - \widehat{\boldsymbol{\mu}}_k)^T \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \widehat{\boldsymbol{\mu}}_k) = (\mathbf{x} - \widehat{\boldsymbol{\mu}}_j)^T \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \widehat{\boldsymbol{\mu}}_j)$$

where

$$(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) = ||\mathbf{x} - \boldsymbol{\mu}_k||_{\Sigma}^2$$

is called the *Mahalanobis distance* between $\mathbf{x}$ and $\boldsymbol{\mu}_k$.
If $\boldsymbol{\Sigma}$ is the identity matrix, the decision boundary becomes:

$$(\mathbf{x} - \widehat{\boldsymbol{\mu}}_k)^T (\mathbf{x} - \widehat{\boldsymbol{\mu}}_k) = (\mathbf{x} - \widehat{\boldsymbol{\mu}}_j)^T (\mathbf{x} - \widehat{\boldsymbol{\mu}}_j)$$

the classifier is the *Euclidean distance* classifier.

Logistic regression

Assuming two classes with Gaussian $p(\mathbf{x}|c)$ and equal covariance matrices leads to

$$p(c|\mathbf{x}) = \frac{1}{1 + \exp(-w_0 - \mathbf{w}^T\mathbf{x})}$$

where

$$-w_0 - \mathbf{w}^T\mathbf{x}$$

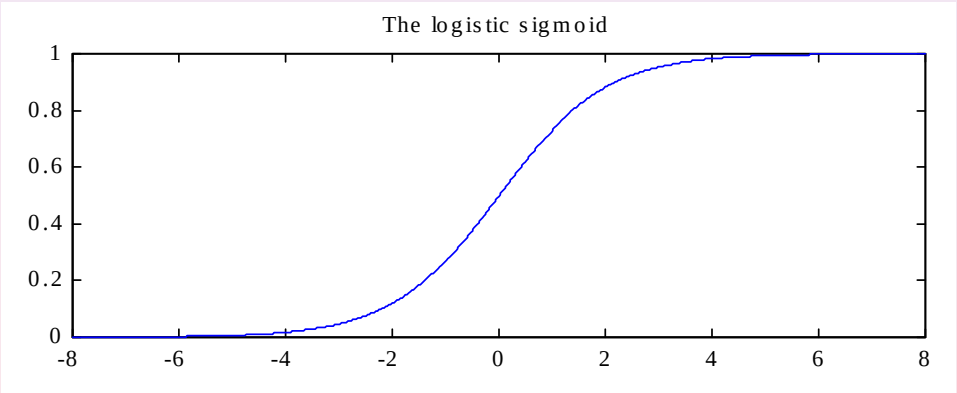
is linear in $\mathbf{x}$.

Logistic regression

The function

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

is usually called the *logistic function* or *logistic sigmoid*.



Logistic regression has less free parameters than the full Gaussian $p(\mathbf{x}|c)$ method; $[D + 1$ vs. $D(D + 5)/2]$.

Logistic regression decision boundary

Training error = 0.03%, Test error = 0.09%