Machine Learning

2. Logistic Regression and LDA

Steffen Rendle

(Slides mainly by Lars Schmidt-Thieme)

Information Systems and Machine Learning Lab (ISMLL)
Institute for Business Economics and Information Systems
& Institute for Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

Classification / Supervised Learning

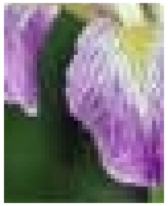
Example: classifying iris plants
(Anderson 1935).

150 iris plants (50 of each species):

- species: setosa, versicolor, virginica
- length and width of sepals (in cm)
- length and width of petals (in cm)

iris setosa

iris versicolor



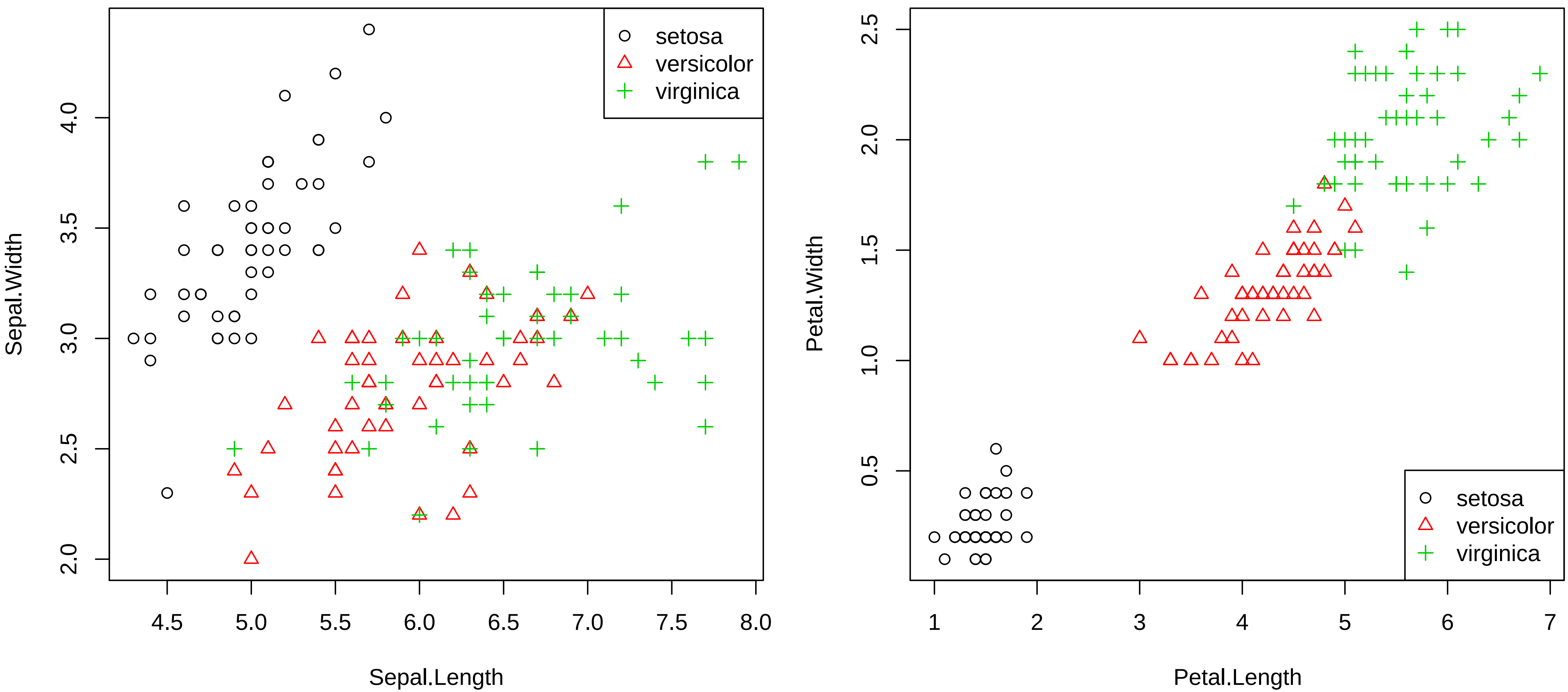
iris virginica

See iris species database
(http://www.badbear.com/signa).

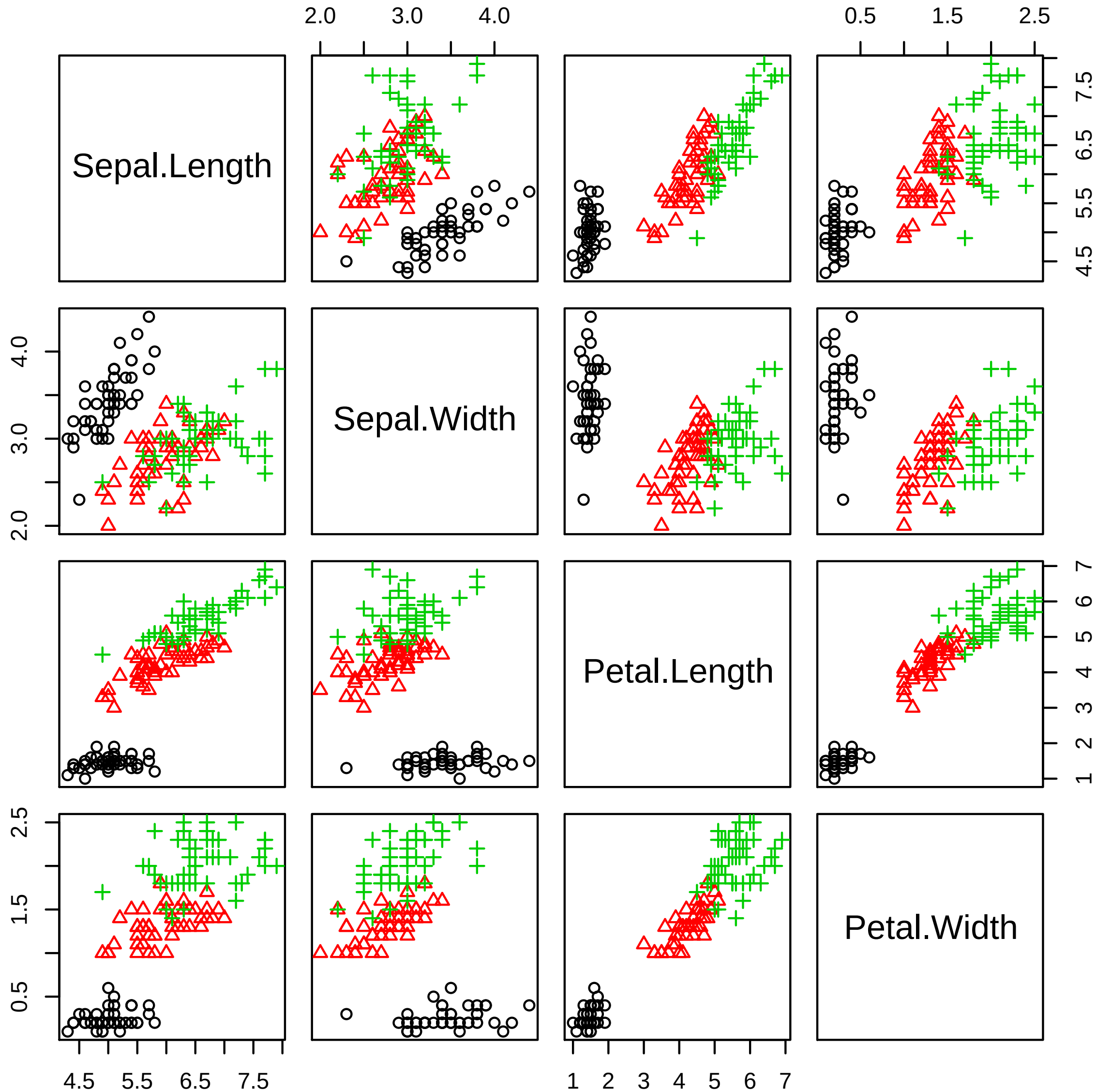
Classification / Supervised Learning

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
4	4.60	3.10	1.50	0.20	setosa
5	5.00	3.60	1.40	0.20	setosa
⋮	⋮	⋮	⋮	⋮	
51	7.00	3.20	4.70	1.40	versicolor
52	6.40	3.20	4.50	1.50	versicolor
53	6.90	3.10	4.90	1.50	versicolor
54	5.50	2.30	4.00	1.30	versicolor
⋮	⋮	⋮	⋮	⋮	
101	6.30	3.30	6.00	2.50	virginica
102	5.80	2.70	5.10	1.90	virginica
103	7.10	3.00	5.90	2.10	virginica
104	6.30	2.90	5.60	1.80	virginica
105	6.50	3.00	5.80	2.20	virginica
⋮	⋮	⋮	⋮	⋮	
150	5.90	3.00	5.10	1.80	virginica

Classification / Supervised Learning



Classification / Supervised Learning



1. The Classification Problem
2. **Logistic Regression**
3. Multi-category Targets
4. Linear Discriminant Analysis

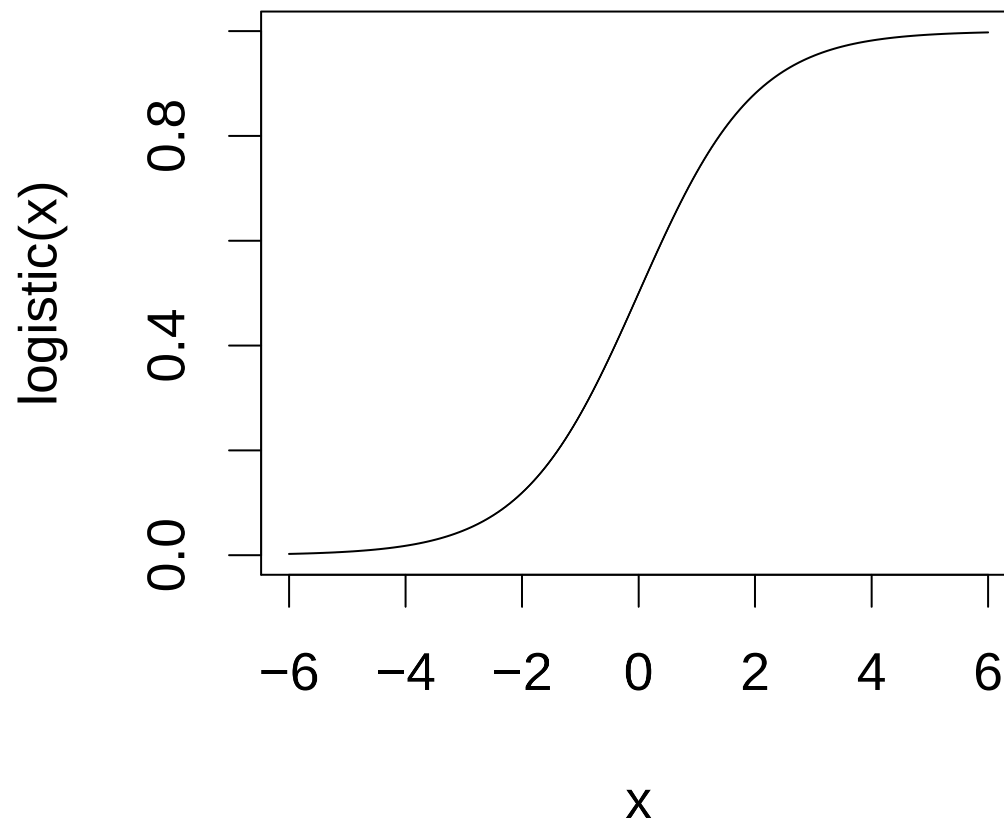
The Logistic Function

Logistic function:

$$\text{logistic}(x) := \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

The logistic function is a function that

- has values between 0 and 1,
- converges to 1 when approaching $+\infty$,
- converges to 0 when approaching $-\infty$,
- is smooth and symmetric at $(0, 0.5)$.



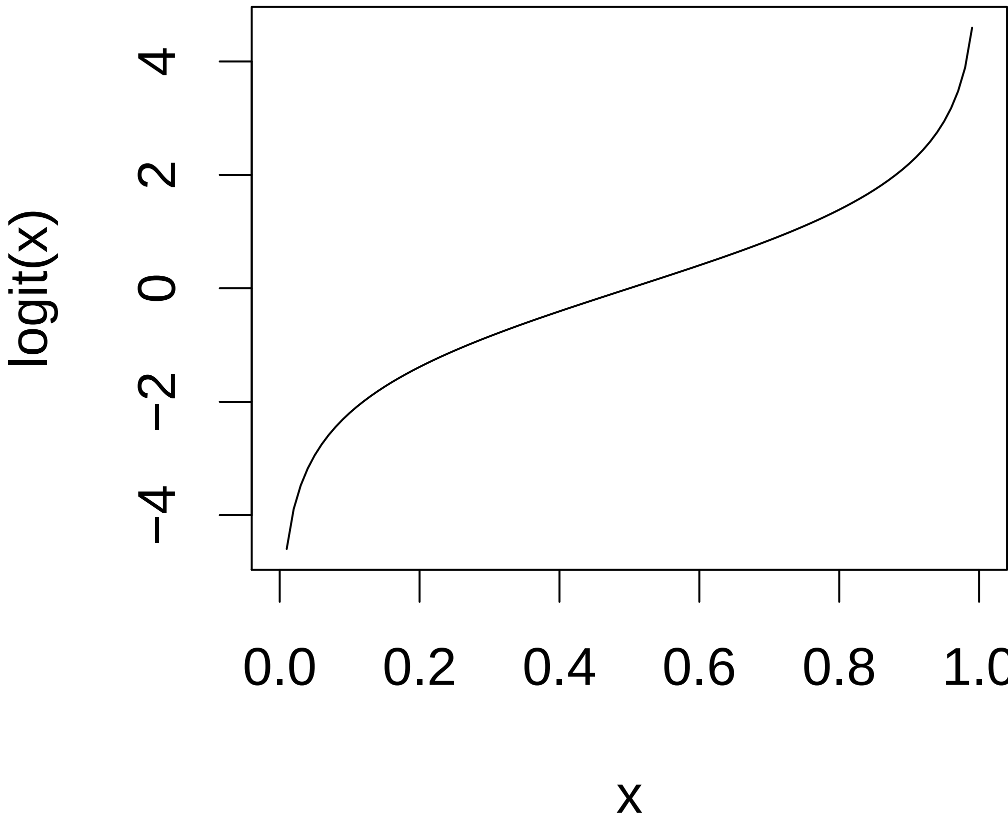
The Logit Function

Logit function:

$$\text{logit}(x) := \log(\frac{x}{1-x})$$

The logit function is a function that

- is defined between 0 and 1,
- converges to $+\infty$ when approaching 1,
- converges to $-\infty$ when approaching 0,
- is smooth and symmetric at $(0.5, 0)$.
- is the inverse of the logistic function.



Logistic Regression Model

Make it simple:

- target Y is binary: $\mathcal{Y} := \{0, 1\}$.

The linear regression model

$$Y = \langle X, \beta \rangle + \epsilon$$

is not suited for predicting y as it can assume all kinds of intermediate values.

Instead of predicting Y directly, we predict

$p(Y = 1|X)$, the probability of Y being 1 knowing X.

Logistic Regression Model

But linear regression is also not suited for predicting probabilities, as its predicted values are principially unbounded.

Use a trick and transform the unbounded target by a function that forces it into the unit interval $[0, 1]$, e.g., the logistic function.

Logistic regression model:

$$p(Y = 1 \,|\, X) = \text{logistic}(\langle X, \beta \rangle) + \epsilon = \frac{e^{\sum_{i=1}^{n} \beta_i X_i}}{1 + e^{\sum_{i=1}^{n} \beta_i X_i}} + \epsilon$$

A Naive Estimator

A naive estimator could fit the linear regression model to Y (treated as continuous target) directly, i.e.,

$$Y = \langle X, \beta \rangle + \epsilon$$

and then post-process the linear prediction via

$$\hat{p}(Y = 1 \,|\, X) = \text{logistic}(\hat{Y}) = \text{logistic}(\langle X, \hat{\beta} \rangle) = \frac{e^{\sum_{i=1}^n \hat{\beta}_i X_i}}{1 + e^{\sum_{i=1}^n \hat{\beta}_i X_i}}$$

But

- $\hat{\beta}$ have the property to give minimal RSS for $\hat{Y}$, but what properties do the $\hat{p}(Y = 1 \,|\, X)$ have?
- A probabilistic interpretation requires normal errors for Y, which is not adequate as Y is bounded to $[0, 1]$.

Maximum Likelihood Estimator

As fit criterium, again the likelihood is used.

As Y is binary, it has a Bernoulli distribution:

$$Y|X = \text{Bernoulli}(p(Y = 1 \,|\, X))$$

Thus, the conditional likelihood function is:

$$\begin{aligned} L_{\mathcal{D}}^{\text{cond}}(\hat{\beta}) &= \prod_{i=1}^{n} p(Y = y_i \,|\, X = x_i; \hat{\beta}) \\ &= \prod_{i=1}^{n} p(Y = 1 \,|\, X = x_i; \hat{\beta})^{y_i} (1 - p(Y = 1 \,|\, X = x_i; \hat{\beta}))^{1-y_i} \end{aligned}$$

Background: Gradient Descent

Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, find x with minimal $f(x)$.

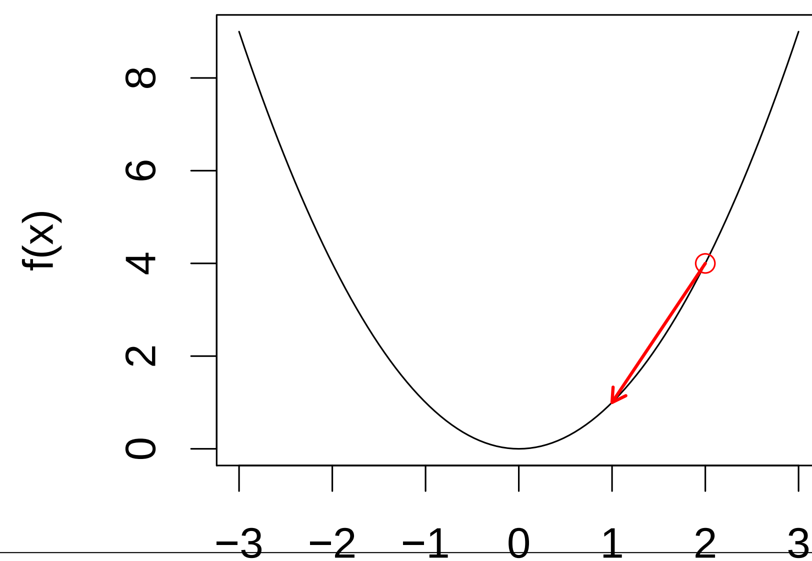
Idea: start from a random x_0 and then improve step by step, i.e., choose x_{n+1} with

$$f(x_{n+1}) \leq f(x_n)$$

Choose the negative gradient $-\frac{\partial f}{\partial x}(x_n)$ as direction for descent, i.e.,

$$x_{n+1} - x_n = -\alpha_n \cdot \frac{\partial f}{\partial x}(x_n)$$

with a suitable step length $\alpha_n > 0$.



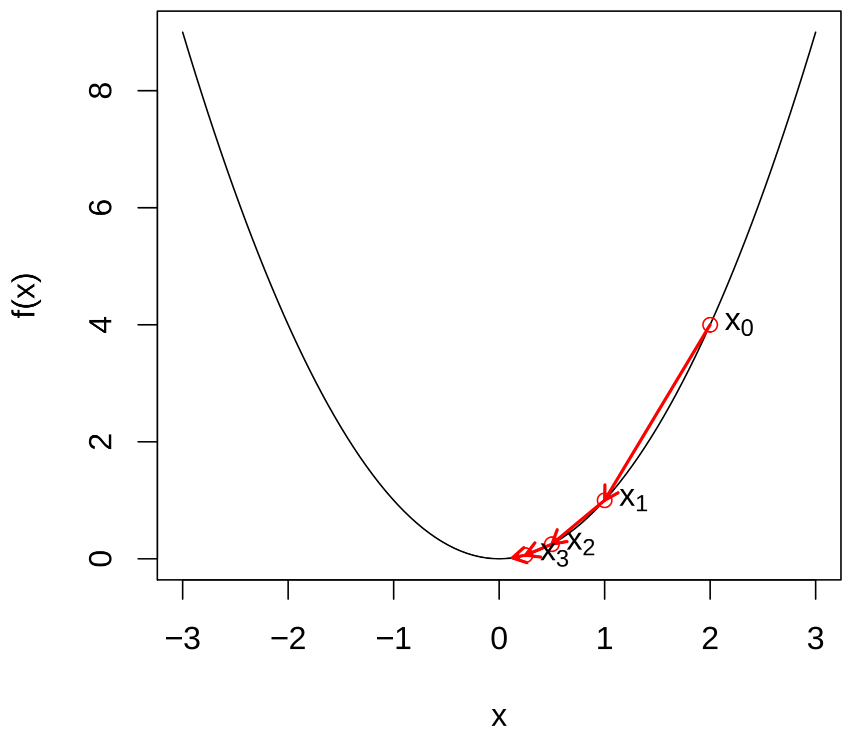
Background: Gradient Descent / Example

Example:

$$f(x) := x^2, \quad \frac{\partial f}{\partial x}(x) = 2x, \quad x_0 := 2, \quad \alpha_n :\equiv 0.25$$

Then we compute iteratively:

n	x_n	$\frac{\partial f}{\partial x}(x_n)$	x_{n+1}
0	2	4	1
1	1	2	0.5
2	0.5	1	0.25
3	0.25	⋮	⋮
⋮	⋮	⋮	⋮



using

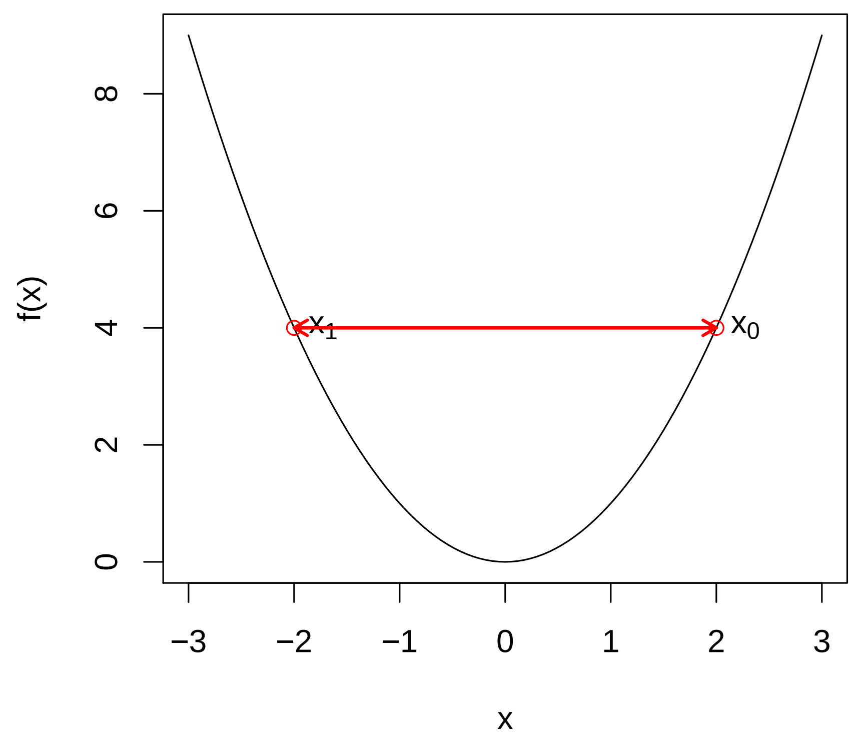
$$x_{n+1} = x_n - \alpha_n \cdot \frac{\partial f}{\partial x}(x_n)$$

Background: Gradient Descent / Step Length

Why do we need a step length? Can we set $\alpha_n \equiv 1$?

The negative gradient gives a direction of descent only in an infinitesimal neighborhood of x_n.

Thus, the step length may be too large, and the function value of the next point does not decrease.



Background: Gradient Descent / Step Length

There are many different strategies to adapt the step length s.t.

1. the function value actually decreases and
2. the step length becomes not too small (and thus convergence slow)

Armijo-Principle:

$$\alpha_n := \max\{\alpha \in \{2^{-j} \mid j \in \mathbb{N}_0\} \mid f(x_n - \alpha \frac{\partial f}{\partial x}(x_n)) \leq f(x_n) - \alpha\delta \langle \frac{\partial f}{\partial x}(x_n), \frac{\partial f}{\partial x}(x_n) \rangle \}$$

with $\delta \in (0, 1)$.

Background: Newton Algorithm

Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, find x with minimal $f(x)$.

The Newton algorithm is based on a quadratic Taylor expansion of f around x_n:

$$F_n(x) := f(x_n) + \langle \frac{\partial f}{\partial x}(x_n), x - x_n \rangle + \frac{1}{2} \langle x - x_n, \frac{\partial^2 f}{\partial x \partial x^T}(x_n)(x - x_n) \rangle$$

and minimizes this approximation in each step, i.e.,

$$\frac{\partial F_n}{\partial x}(x_{n+1}) \stackrel{!}{=} 0$$

with

$$\frac{\partial F_n}{\partial x}(x) = \frac{\partial f}{\partial x}(x_n) + \frac{\partial^2 f}{\partial x \partial x^T}(x_n)(x - x_n)$$

which leads to the Newton algorithm:

$$\frac{\partial^2 f}{\partial x \partial x^T}(x_n)(x_{n+1} - x_n) = -\frac{\partial f}{\partial x}(x_n)$$

starting with a random x_0.

Newton Algorithm for the Loglikelihood

$$L_{\mathcal{D}}^{\text{cond}}(\hat\beta) = \prod_{i=1}^{n} p(Y=1 \mid X=x_i;\hat\beta)^{y_i}(1-p(Y=1 \mid X=x_i;\hat\beta))^{1-y_i}$$

$$\begin{aligned}
\log L_{\mathcal{D}}^{\text{cond}}(\hat\beta) &= \sum_{i=1}^{n} y_i \log p(Y=1 \mid X=x_i;\hat\beta) + (1-y_i)\log(1-p(Y=1 \mid X=x_i;\hat\beta)) \\
&= \sum_{i=1}^{n} y_i \log(\frac{e^{\langle x_i,\hat\beta\rangle}}{1+e^{\langle x_i,\hat\beta\rangle}}) + (1-y_i)\log(1-\frac{e^{\langle x_i,\hat\beta\rangle}}{1+e^{\langle x_i,\hat\beta\rangle}}) \\
&= \sum_{i=1}^{n} y_i(\langle x_i,\hat\beta\rangle - \log(1+e^{\langle x_i,\hat\beta\rangle})) + (1-y_i)\log(\frac{1}{1+e^{\langle x_i,\hat\beta\rangle}}) \\
&= \sum_{i=1}^{n} y_i(\langle x_i,\hat\beta\rangle - \log(1+e^{\langle x_i,\hat\beta\rangle})) + (1-y_i)(-\log(1+e^{\langle x_i,\hat\beta\rangle})) \\
&= \sum_{i=1}^{n} y_i\langle x_i,\hat\beta\rangle - \log(1+e^{\langle x_i,\hat\beta\rangle})
\end{aligned}$$

Newton Algorithm for the Loglikelihood

$$\begin{aligned}
\log L_{\mathcal{D}}^{\text{cond}}(\hat\beta) &= \sum_{i=1}^{n} y_i \langle x_i, \hat\beta\rangle - \log(1 + e^{\langle x_i, \hat\beta\rangle}) \\
\frac{\partial L_{\mathcal{D}}^{\text{cond}}(\hat\beta)}{\partial \hat\beta} &= \sum_{i=1}^{n} y_i x_i - \frac{1}{1 + e^{\langle x_i, \hat\beta\rangle}} e^{\langle x_i, \hat\beta\rangle} x_i \\
&= \sum_{i=1}^{n} x_i (y_i - p(Y = 1 \mid X = x_i; \hat\beta)) \\
&= \mathbf{X}^T (\mathbf{y} - \mathbf{p})
\end{aligned}$$

with

$$\mathbf{p} := \begin{pmatrix} p(Y = 1 \mid X = x_1; \hat\beta)) \\ \vdots \\ p(Y = 1 \mid X = x_n; \hat\beta)) \end{pmatrix}$$

Newton Algorithm for the Loglikelihood

$$
\begin{aligned}
\frac{\partial L^{\text{cond}}_{\mathcal{D}}(\hat\beta)}{\partial \hat\beta} =& \mathbf{X}^T(\mathbf{y}-\mathbf{p}) \\
\frac{\partial^2 L^{\text{cond}}_{\mathcal{D}}(\hat\beta)}{\partial \hat\beta \partial \hat\beta^T} =& \sum_{i=1}^n -x_i p(Y=1 \,|\, X=x_i;\hat\beta)(1-p(Y=1\,|\,X=x_i;\hat\beta))x_i^T \\
=& -\sum_{i=1}^n x_i x_i^T p(Y=1 \,|\, X=x_i;\hat\beta)(1-p(Y=1\,|\,X=x_i;\hat\beta)) \\
=& -\mathbf{X}^T\mathbf{W}\mathbf{X}
\end{aligned}
$$

with

$$
\mathbf{W} := \begin{pmatrix}
q(x_1;\hat\beta)(1-q(x_1;\hat\beta)) & 0 & \dots & 0 \\
0 & \ddots & & 0 \\
\vdots & & \ddots & \vdots \\
0 & \dots & \dots & q(x_n;\hat\beta)(1-q(x_n;\hat\beta))
\end{pmatrix}
$$

and $q(x;\hat\beta) := P(Y=1\,|\,X=x;\hat\beta)$.

Newton Algorithm for the Loglikelihood

Newton algorithm:

$$\frac{\partial^2 \log L}{\partial \hat\beta \partial \hat\beta^T}(\hat\beta_n)(\hat\beta_{n+1} - \hat\beta_n) = -\frac{\partial \log L}{\partial \hat\beta}(\hat\beta_n)$$

$$-\mathbf{X}^T\mathbf{W}\mathbf{X}(\hat\beta_{n+1} - \hat\beta_n) = -\mathbf{X}^T(\mathbf{y} - \mathbf{p})$$

$$\mathbf{X}^T\mathbf{W}\mathbf{X}\hat\beta_{n+1} = \mathbf{X}^T\mathbf{W}(\mathbf{X}\hat\beta_n + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}))$$

Equivalent to a weighted least squares of the “adjusted response”

$$z := \mathbf{X}\hat\beta_n + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$$

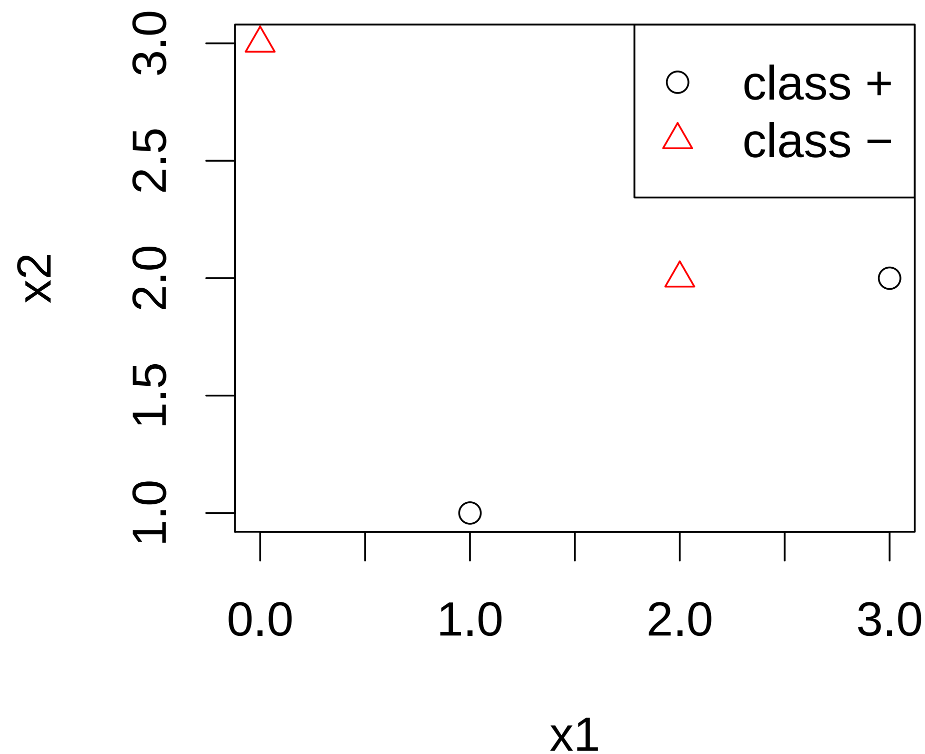
on X known as **iteratively reweighted least squares (IRLS)**.

IRLS typically is started at $\hat\beta^{(0)} := 0$.

Example

Learn a classification function for the following data:

x1	x2	y
1	1	+
3	2	+
2	2	-
0	3	-



Example

$x1$	$x2$	y
1	1	+
3	2	+
2	2	−
0	3	−

$$\mathbf{X} := \begin{pmatrix} 1 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 2 \\ 1 & 0 & 3 \end{pmatrix}, \quad \mathbf{y} := \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\beta}^{(0)} := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{p}^{(0)} := \left(\frac{e^{\langle \beta, x_i \rangle}}{1 + e^{\langle \beta, x_i \rangle}}\right)_i = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}, \quad w^{(0)} := \mathbf{p}^{(0)}(1 - \mathbf{p}^{(0)}) = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix},$$

$$z^{(0)} := \mathbf{X}\hat{\beta}^{(0)} + \mathbf{W}^{(\mathbf{0})^{-1}}(\mathbf{y} - \mathbf{p}^{(\mathbf{0})}) = \begin{pmatrix} 2 \\ 2 \\ -2 \\ -2 \end{pmatrix}$$

Visualization Logistic Regression Models

To visualize a logistic regression model, we can plot the decision boundary

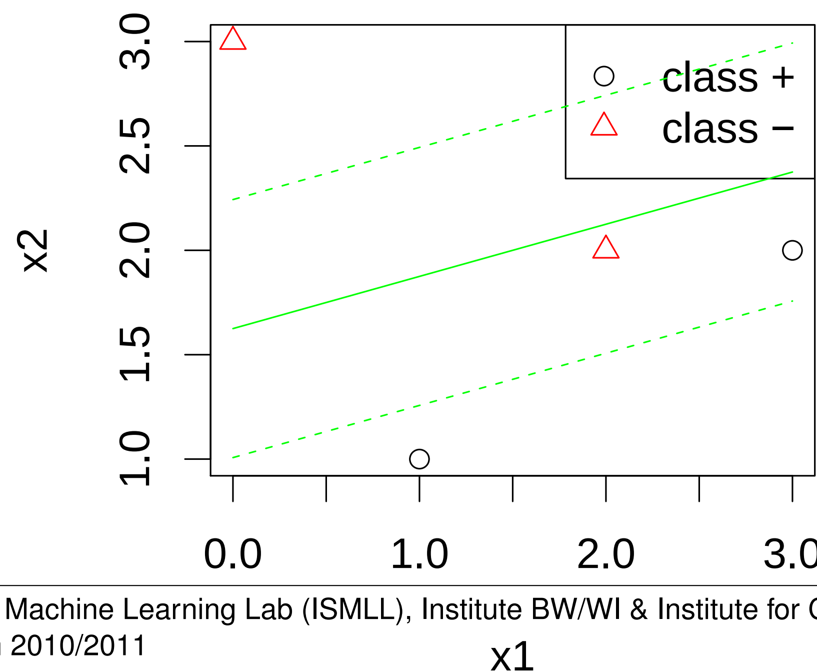
$$\hat{p}(Y = 1 \mid X) = \frac{1}{2}$$

and more detailed some level lines

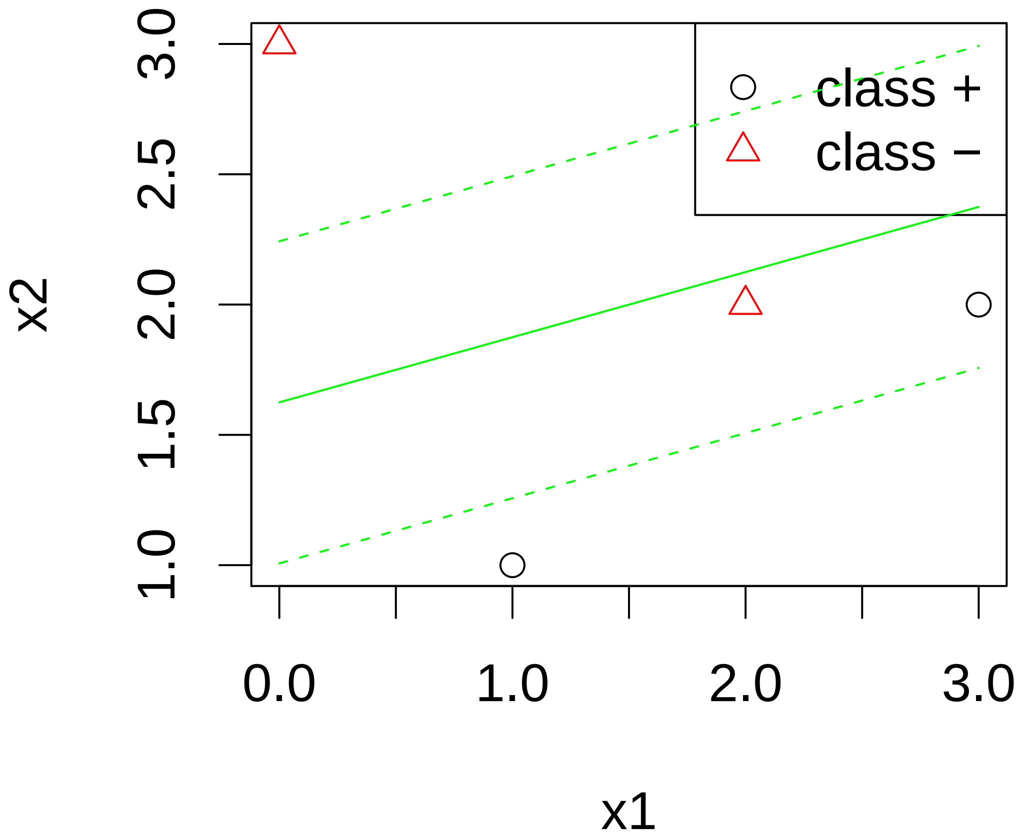
$$\hat{p}(Y = 1 \mid X) = p_0$$

e.g., for $p_0 = 0.25$ and $p_0 = 0.75$:

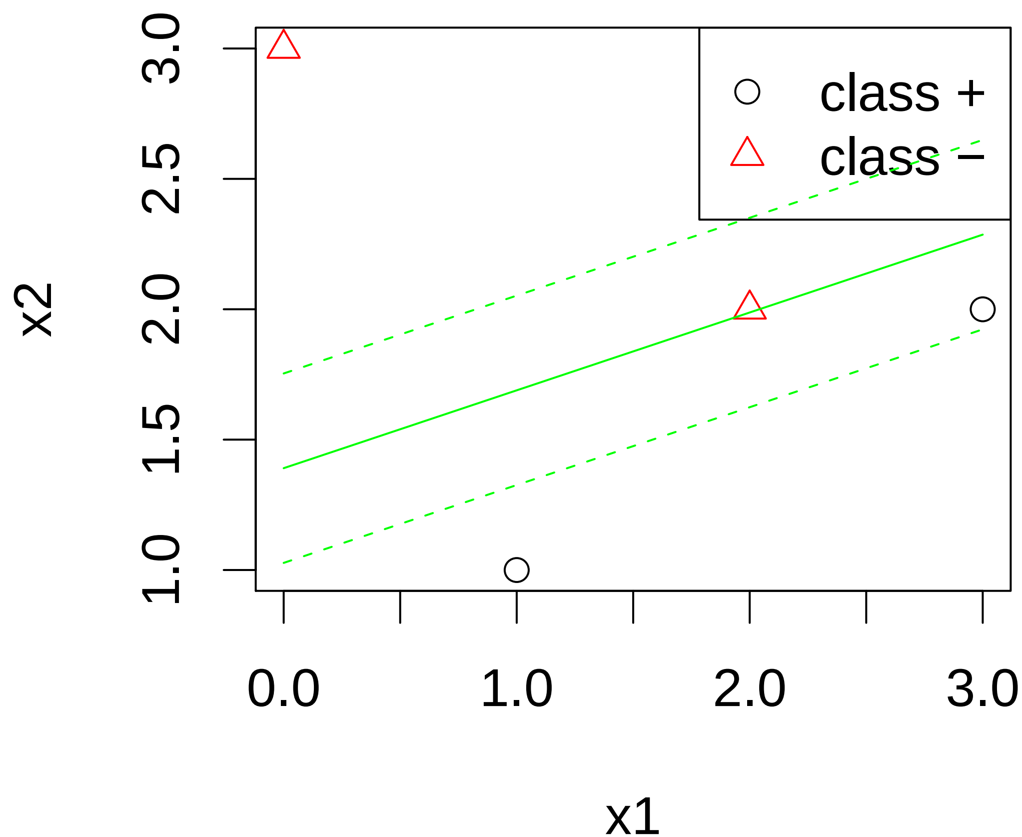
$$\langle \hat{\beta}, X \rangle = \log(\frac{p_0}{1 - p_0})$$



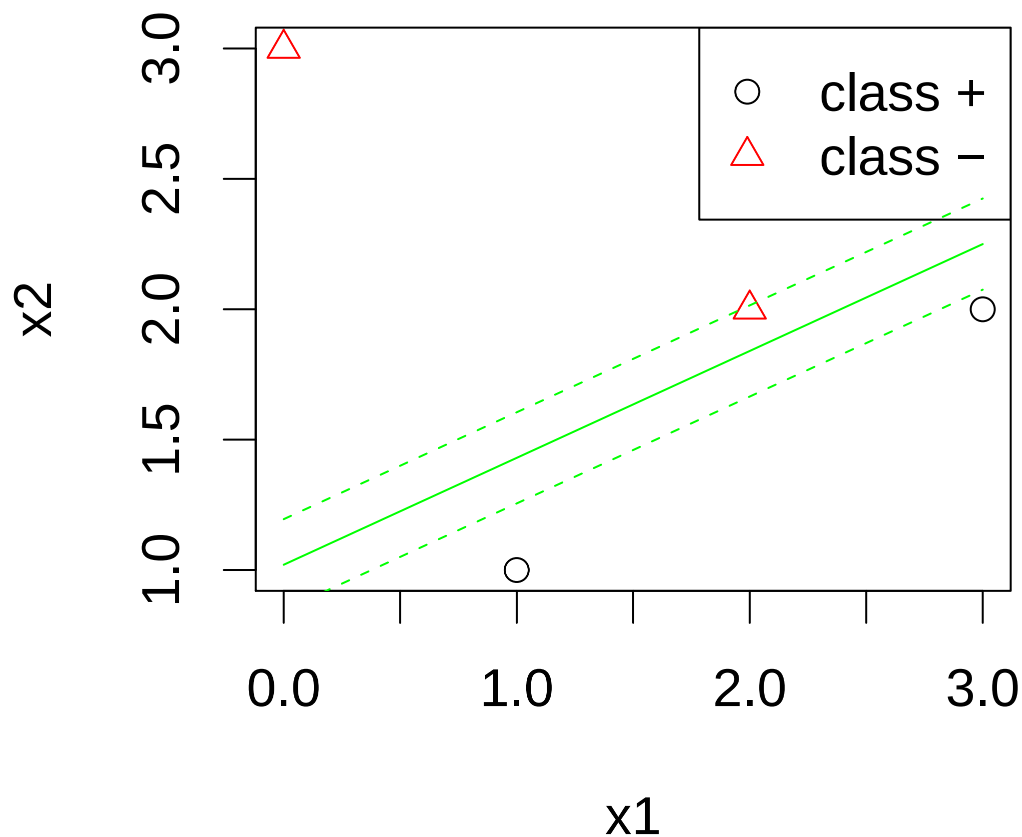
Example



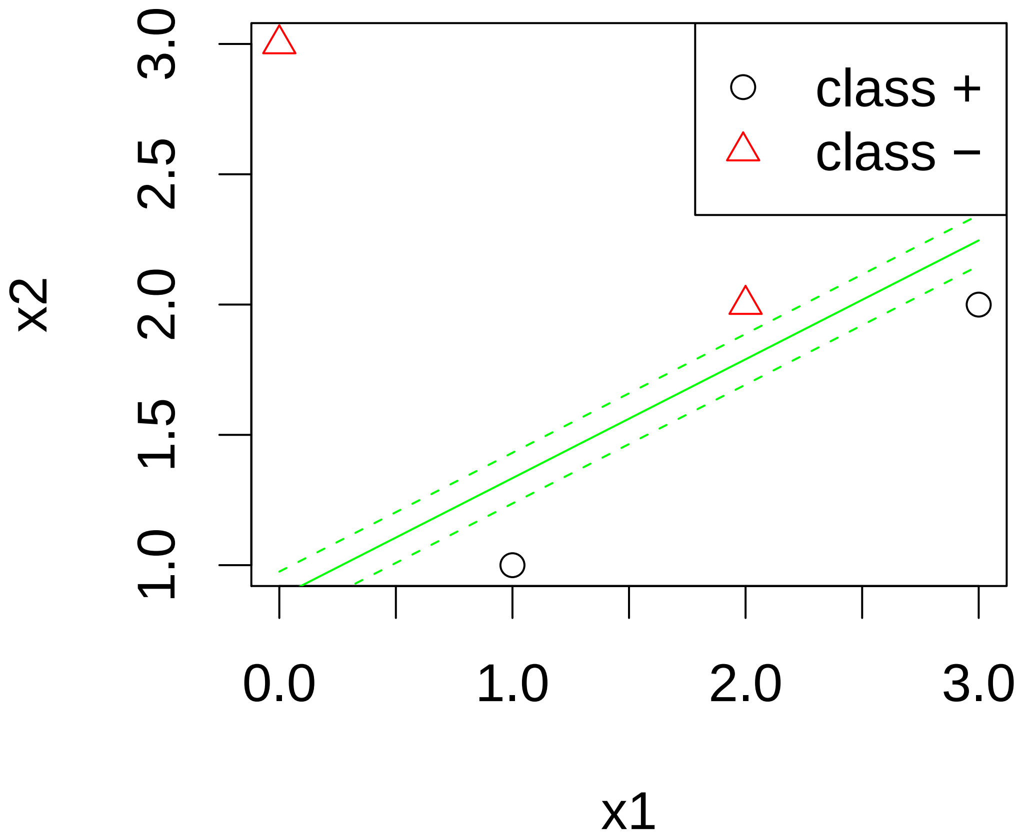
Example



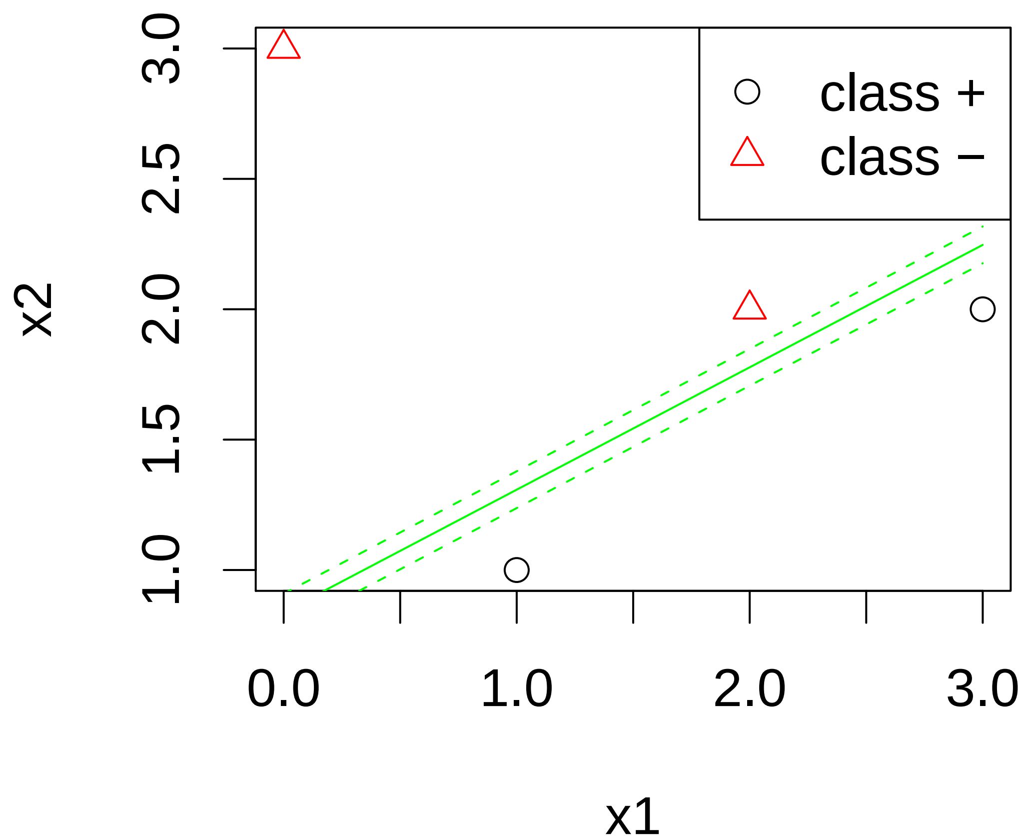
Example



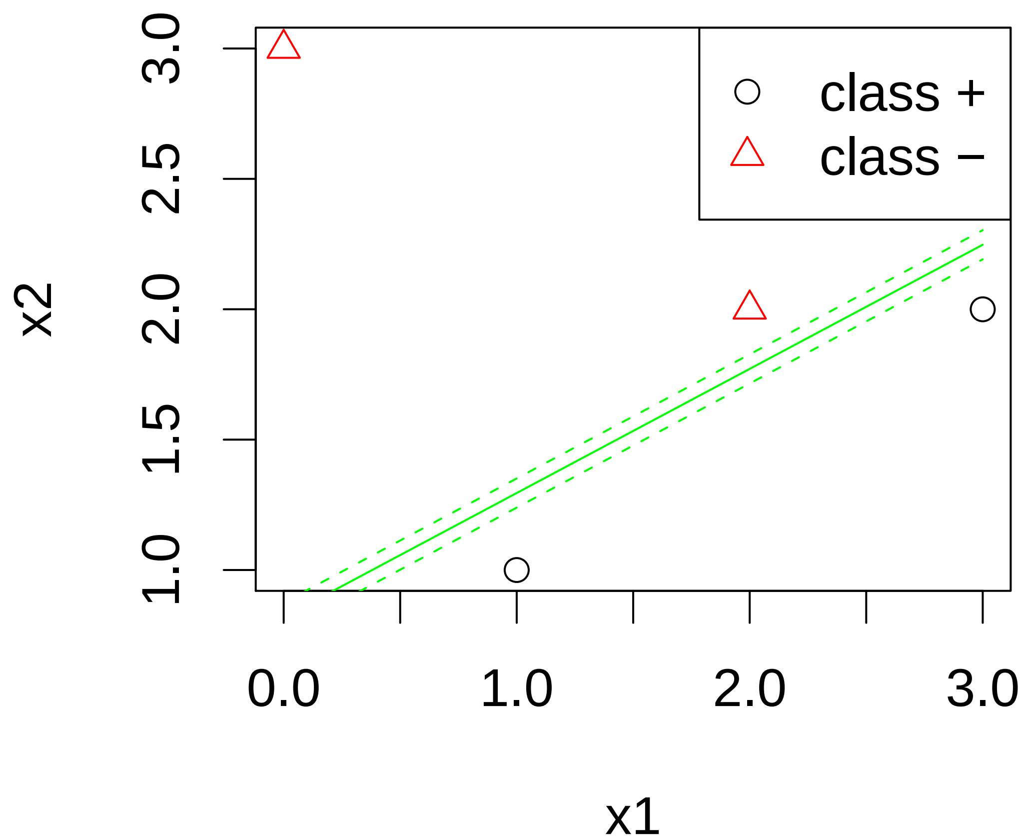
Example



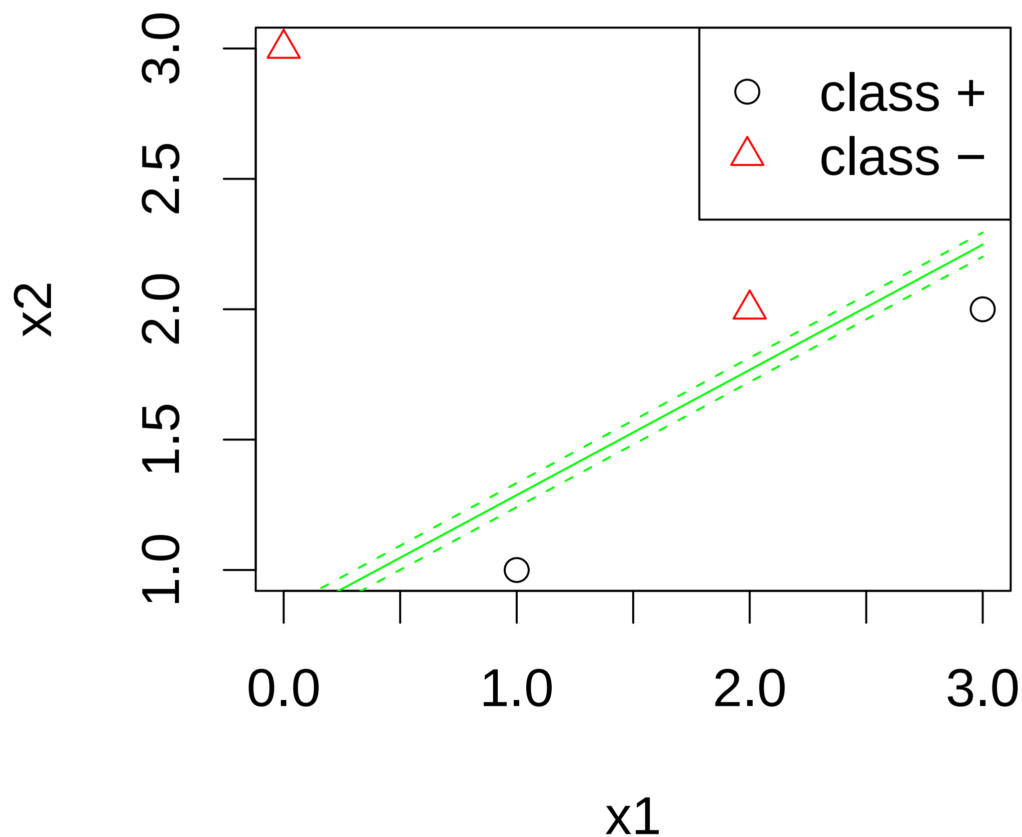
Example



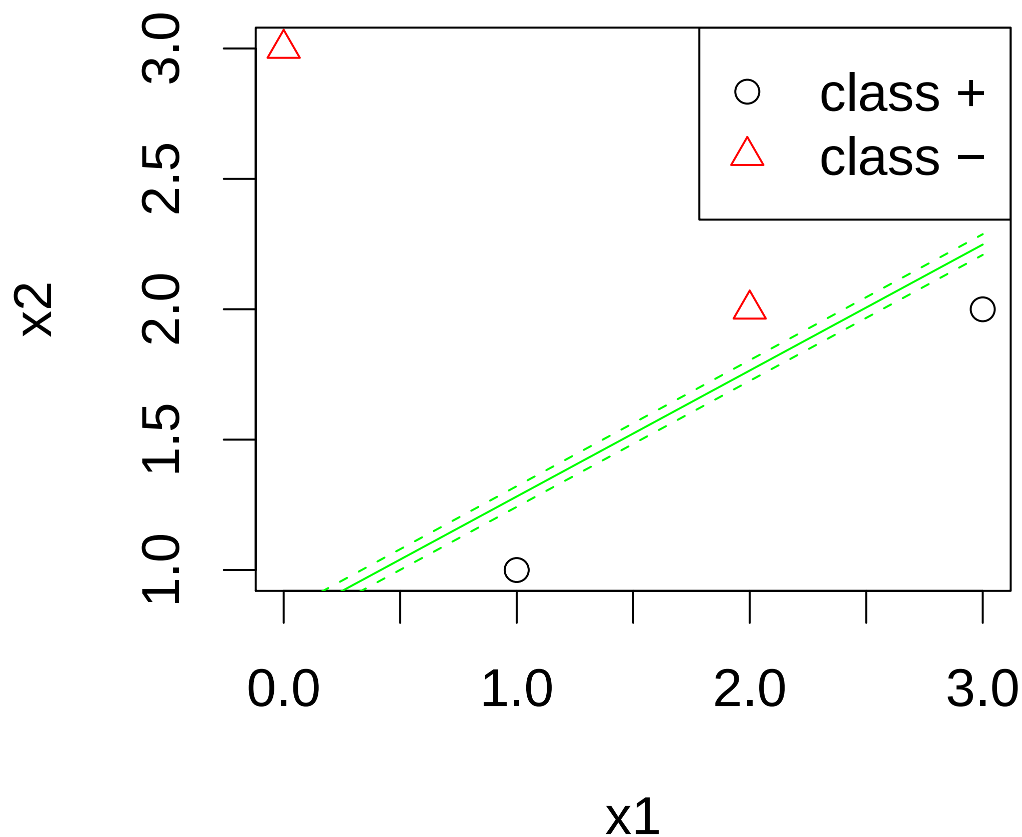
Example



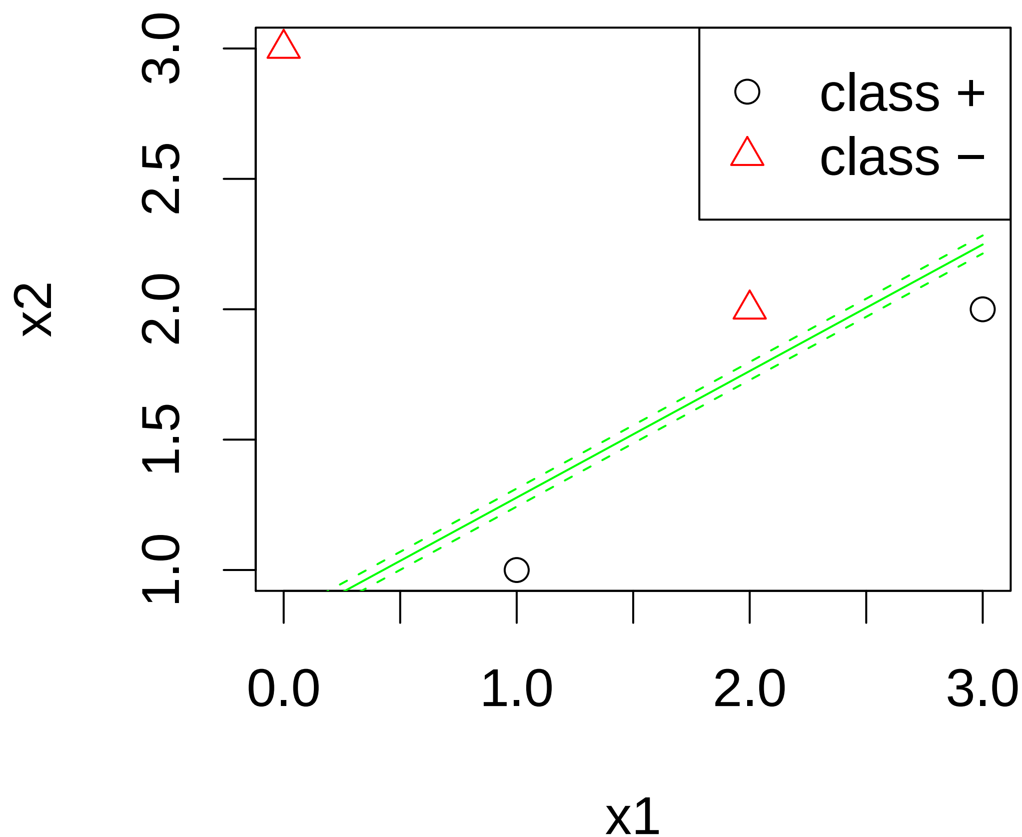
Example



Example



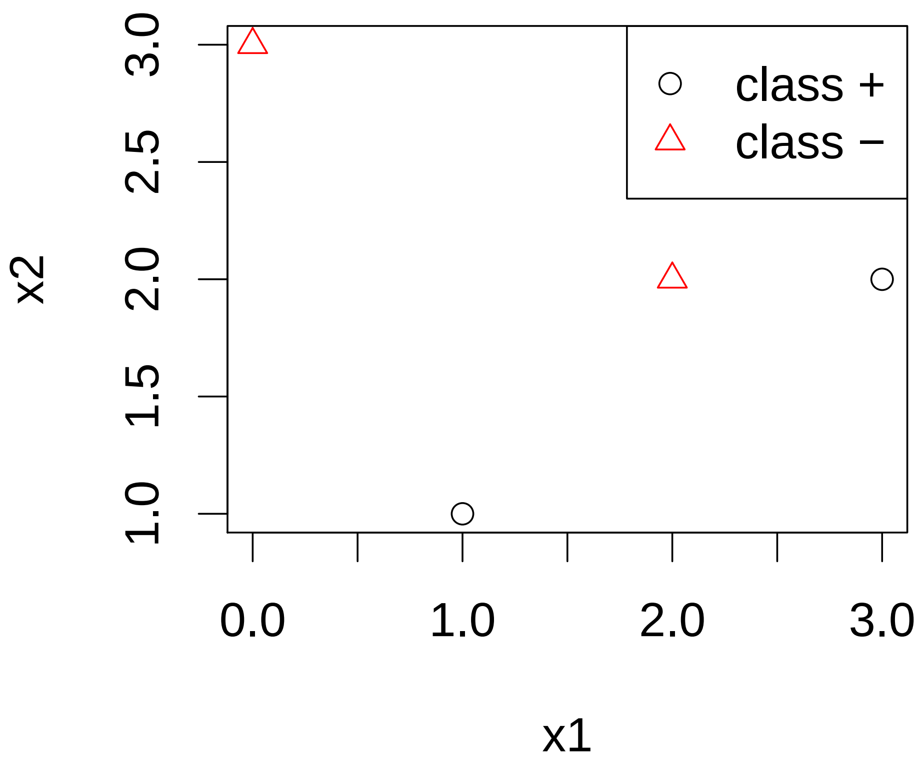
Example



Linear separable vs. linear non-separable

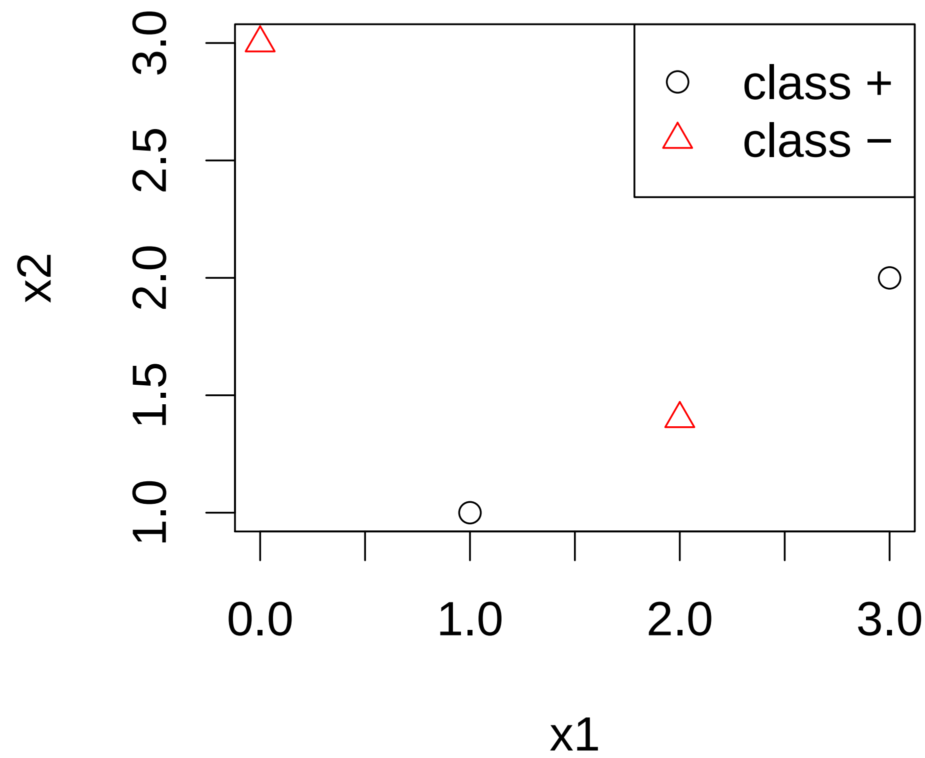
Example 1: Linear separable.

x1	x2	y
1	1	+
3	2	+
2	2	-
0	3	-

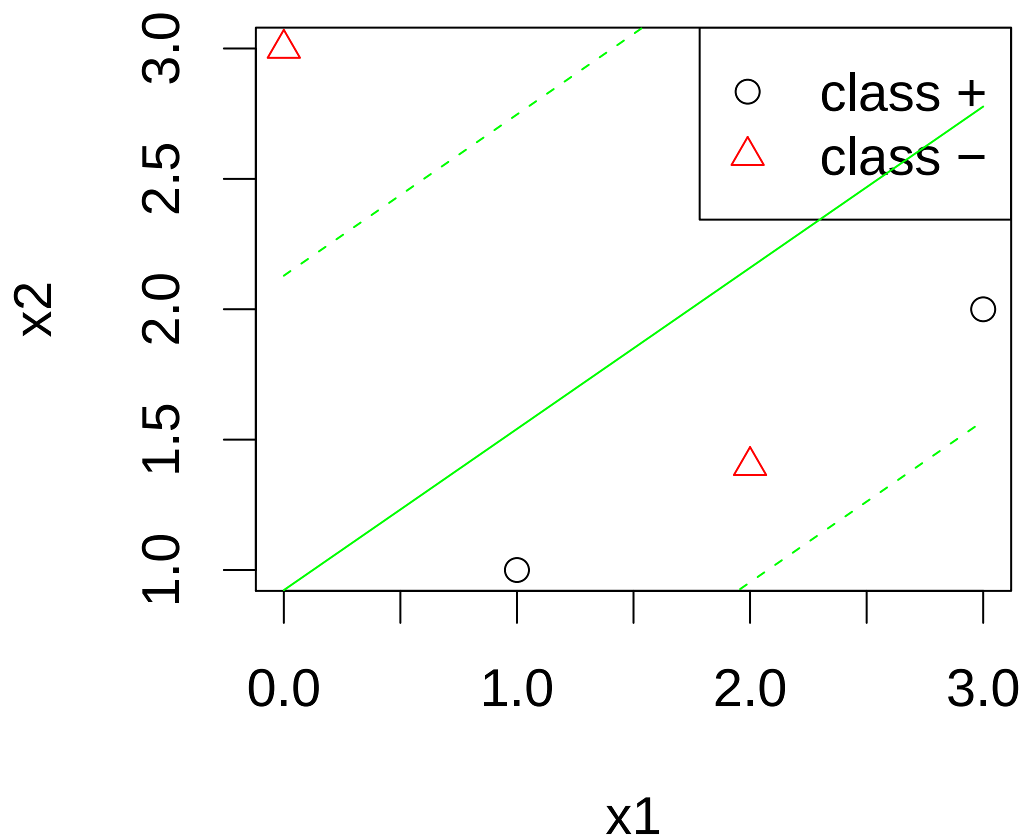


Example 2: Linear non-separable.

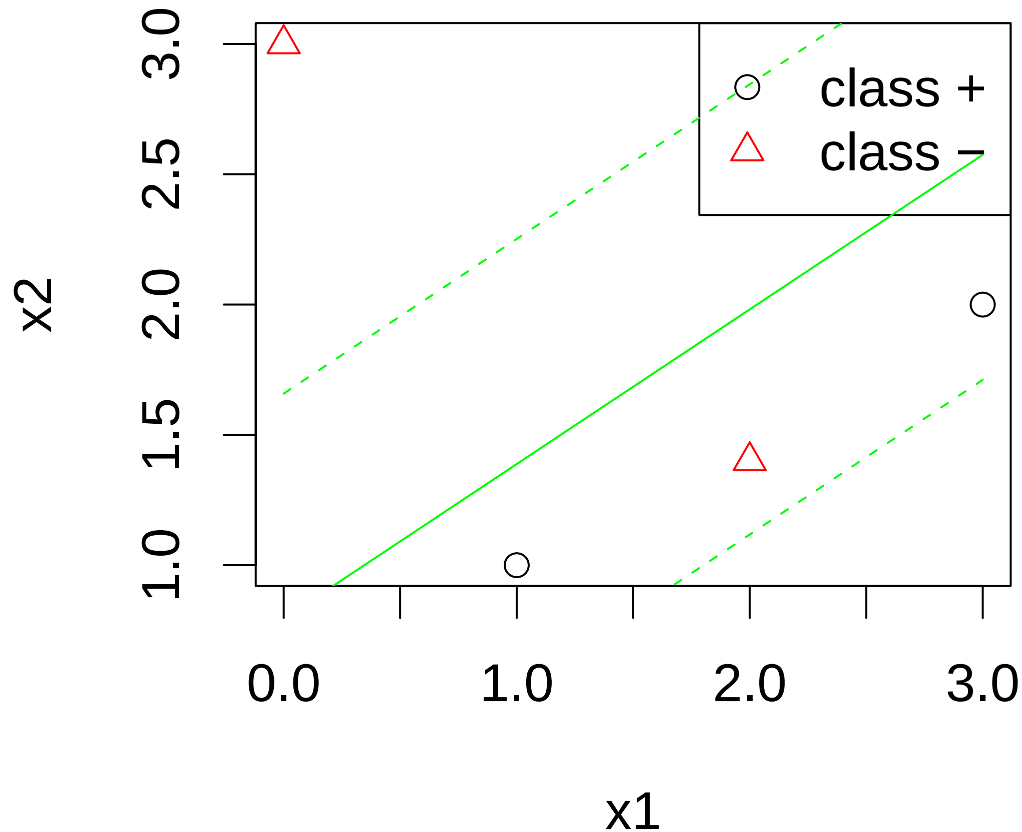
x1	x2	y
1	1	+
3	2	+
2	1.4	-
0	3	-



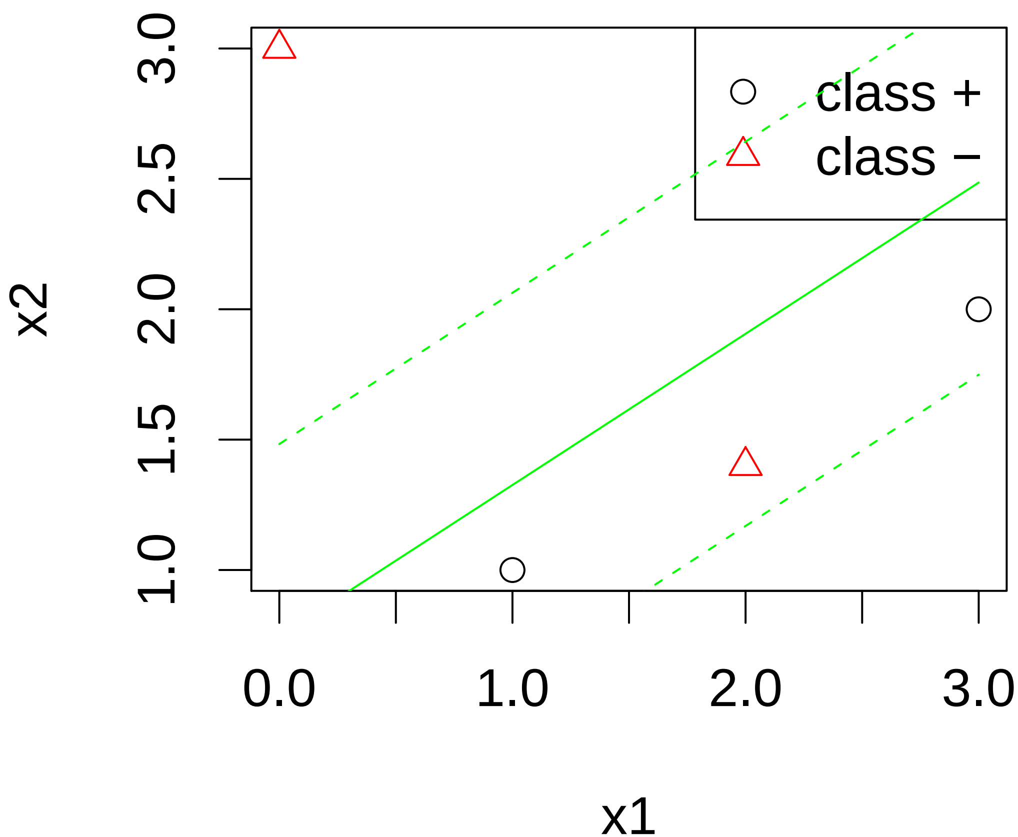
Example 2



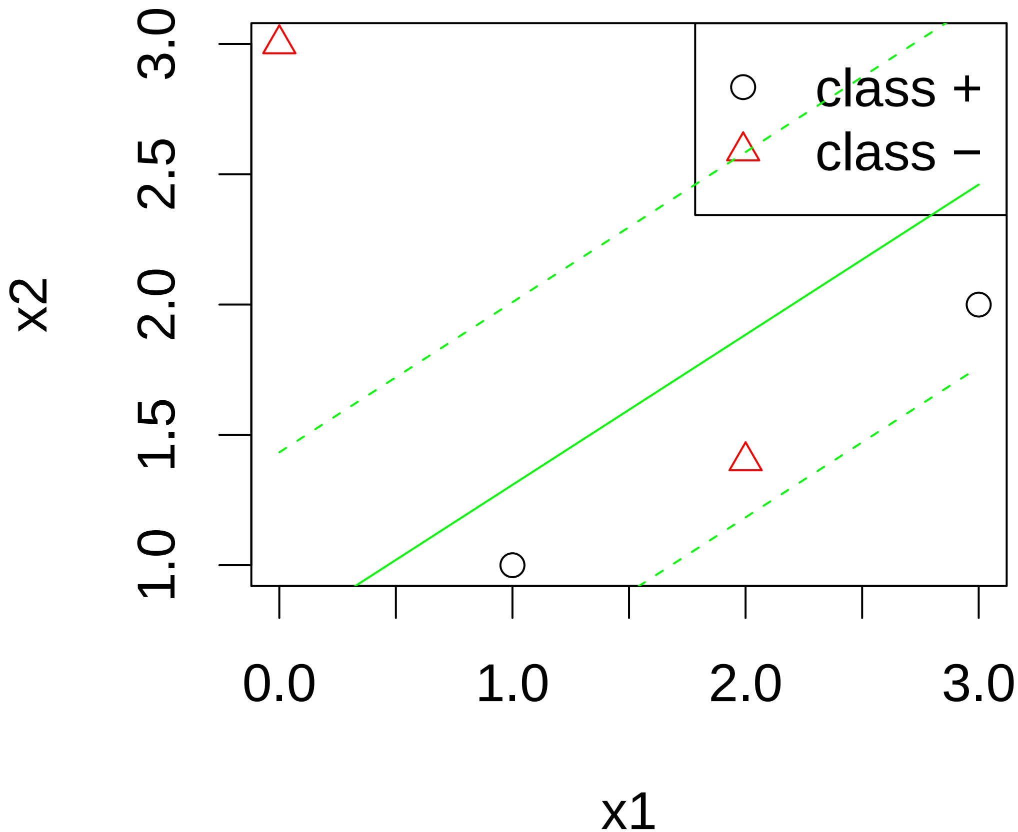
Example 2



Example 2



Example 2



Binary vs. Multi-category Targets

Binary Targets / Binary Classification:
prediction of a nominal target variable with 2 levels/values.

Example: spam vs. non-spam.

Multi-category Targets / Multi-class Targets / Polychotomous Classification:
prediction of a nominal target variable with more than 2 levels/values.

Example: three iris species; 10 digits; 26 letters etc.

Compound vs. Monolithic Classifiers

Compound models

- built from binary submodels,

- different types of compound models employ different sets of submodels:
  1-vs-rest (aka 1-vs-all)
  1-vs-last

  1-vs-1 (Dietterich and Bakiri 1995; aka pairwise classification)
  DAG

- using error-correcting codes to combine component models.

- also ensembles of compound models are used (Frank and Kramer 2004).

Monolithic models (aka "‘one machine"’ (Rifkin and Klautau 2004))

- trying to solve the multi-class target problem intrinsically

- examples: decision trees, special SVMs, etc.

Types of Compound Models

1-vs-rest: one binary classifier per class:

$$f_y : X \to [0,1], \quad y \in Y$$

$$f(x) := (\frac{f_1(x)}{\sum_{y \in Y} f_y(x)}, \ldots, \frac{f_k(x)}{\sum_{y \in Y} f_y(x)})$$

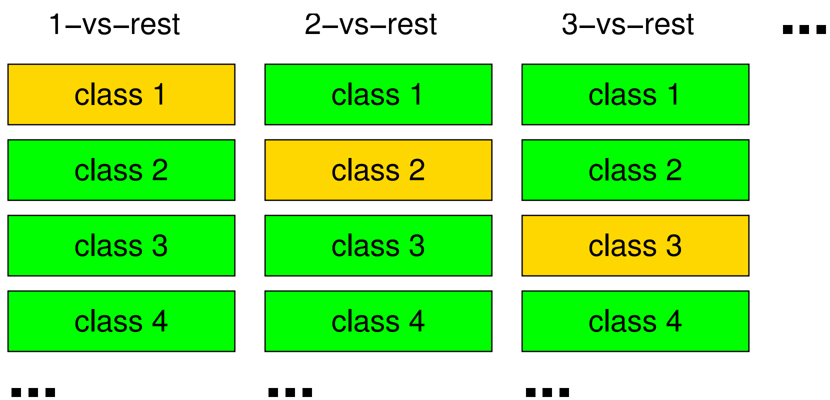
1-vs-last: one binary classifier per class (but last y_k):

$$f_y : X \to [0,1], \quad y \in Y, y \neq y_k$$

$$f(x) := (\frac{f_1(x)}{1 + \sum_{y \in Y} f_y(x)}, \ldots, \frac{f_{k-1}(x)}{1 + \sum_{y \in Y} f_y(x)}, \frac{1}{1 + \sum_{y \in Y} f_y(x)})$$

Polychotomous Discrimination, k target categories

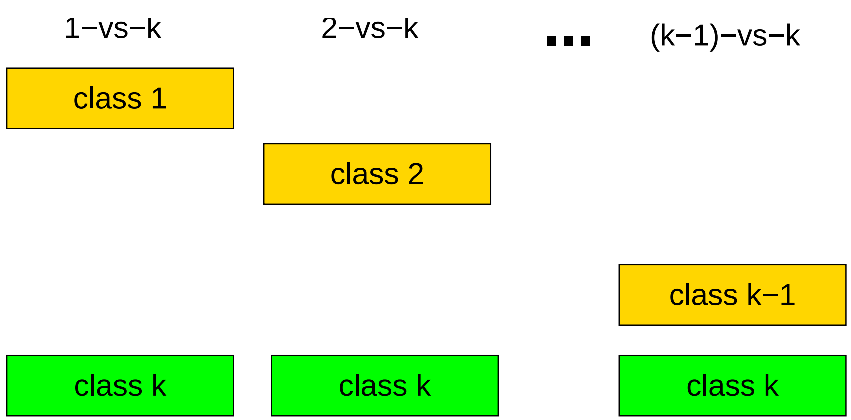
1-vs-rest construction:



k classifiers trained on N cases

kN cases in total

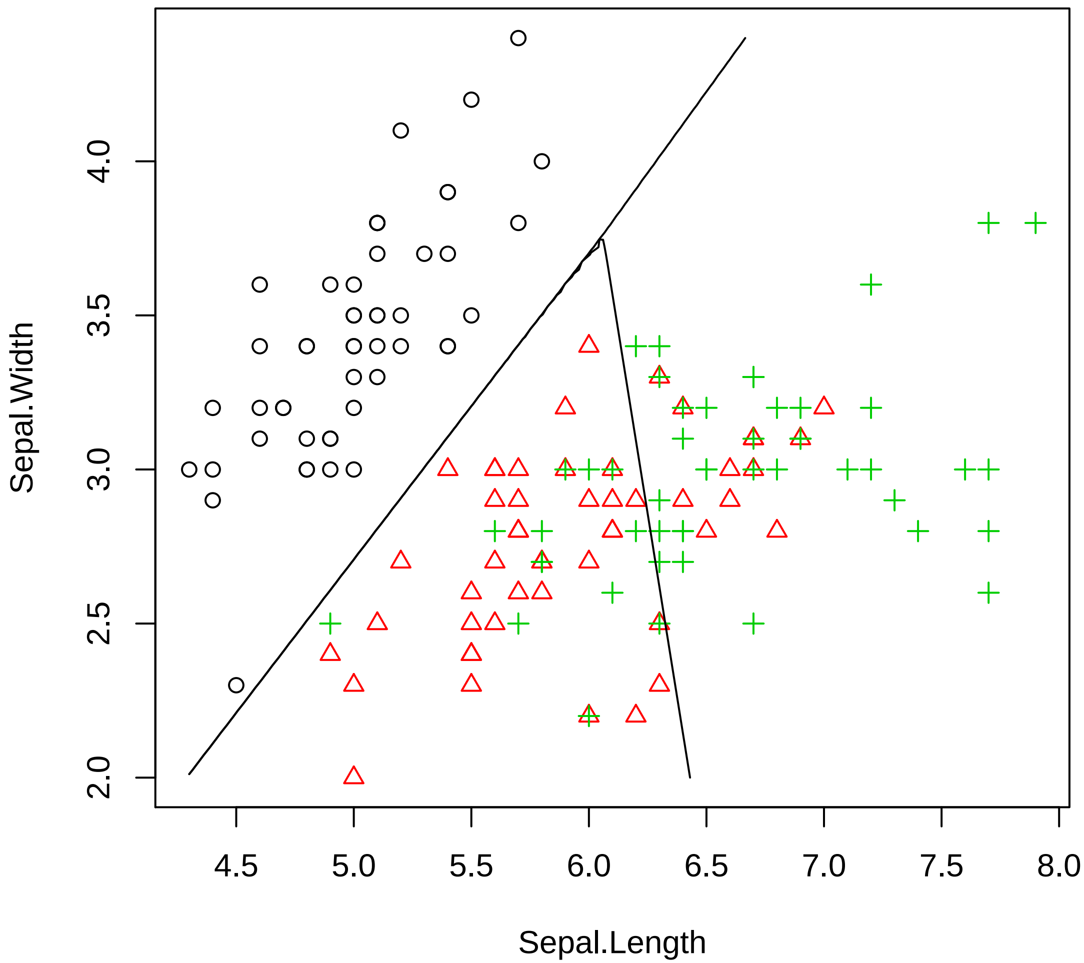
1-vs-last construction:



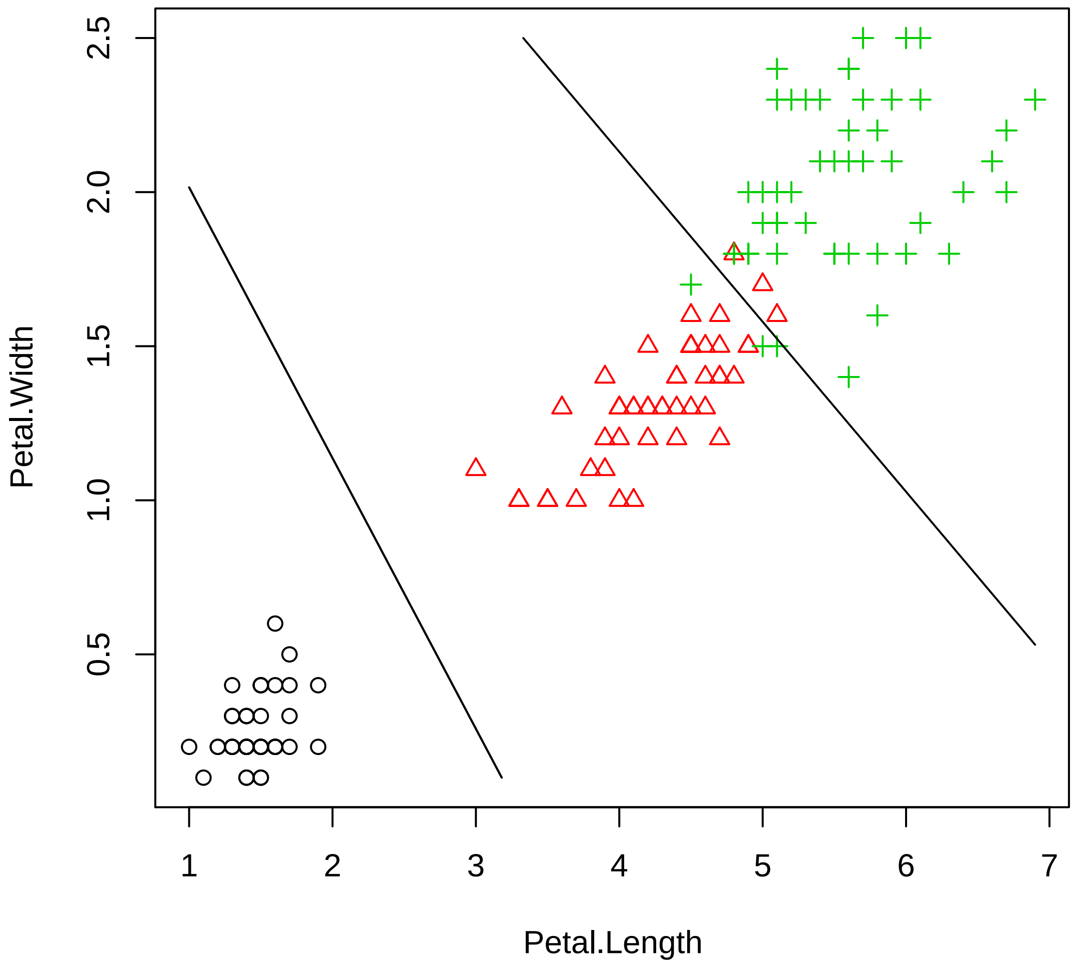
$k-1$ classifiers trained on approx. 2 N/k on average.

$N+(k-2)N_k$ cases in total

Example / Iris data / Logistic Regression



Example / Iris data / Logistic Regression



1. The Classification Problem

2. Logistic Regression

3. Multi-category Targets

4. Linear Discriminant Analysis

Assumptions

In discriminant analysis, it is assumed that

- cases of a each class k are generated according to some probabilities

$$\pi_k = p(Y = k)$$

  and

- its predictor variables are generated by a class-specific multivariate normal distribution

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

  i.e.

$$p_k(x) := \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}\langle x-\mu_k, \Sigma^{-1}(x-\mu_k)\rangle}$$

Decision Rule

Discriminant analysis predicts as follows:

$$\hat{Y}|X = x := \text{argmax}_k \, \pi_k p_k(x) = \text{argmax}_k \, \delta_k(x)$$

with the **discriminant functions**

$$\delta_k(x) := -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}\langle x - \mu_k, \Sigma_k^{-1}(x - \mu_k)\rangle + \log \pi_k$$

Here,

$$\langle x - \mu_k, \Sigma_k^{-1}(x - \mu_k)\rangle$$

is called the **Mahalanobis distance of x and μ_k.**

Thus, discriminant analysis can be described as **prototype method**, where

- each class k is represented by a prototype μ_k and
- cases are assigned the class with the nearest prototype.

Maximum Likelihood Parameter Estimates

The maximum likelihood parameter estimates are as follows:

$$\hat{n}_k := \sum_{i=1}^{n} I(y_i = k), \quad \text{with } I(x = y) := \begin{cases} 1, & \text{if } x = y \\ 0, & \text{else} \end{cases}$$

$$\hat{\pi}_k := \frac{\hat{n}_k}{n}$$

$$\hat{\mu}_k := \frac{1}{\hat{n}_k} \sum_{i:y_i=k} x_i$$

$$\hat{\Sigma}_k := \frac{1}{\hat{n}_k} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

QDA vs. LDA

In the general case, decision boundaries are quadratic due to the quadratic occurrence of x in the Mahalanobis distance. This is called **quadratic discriminant analysis (QDA)**.

If we assume that all classes share the same covariance matrix, i.e.,

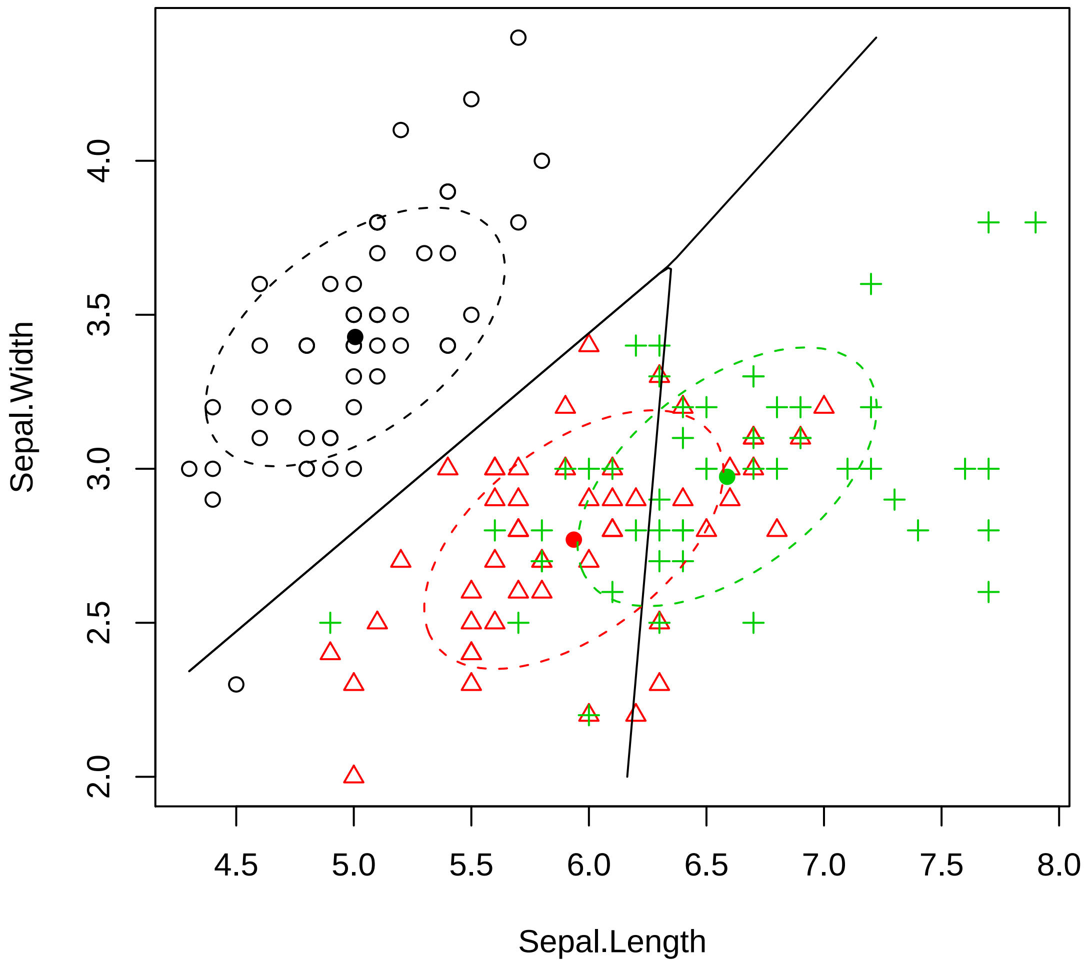
$$\Sigma_k = \Sigma_{k'} \quad \forall k, k'$$

then this quadratic term is canceled and the decision boundaries become linear. This model is called **linear discriminant analysis (LDA)**.

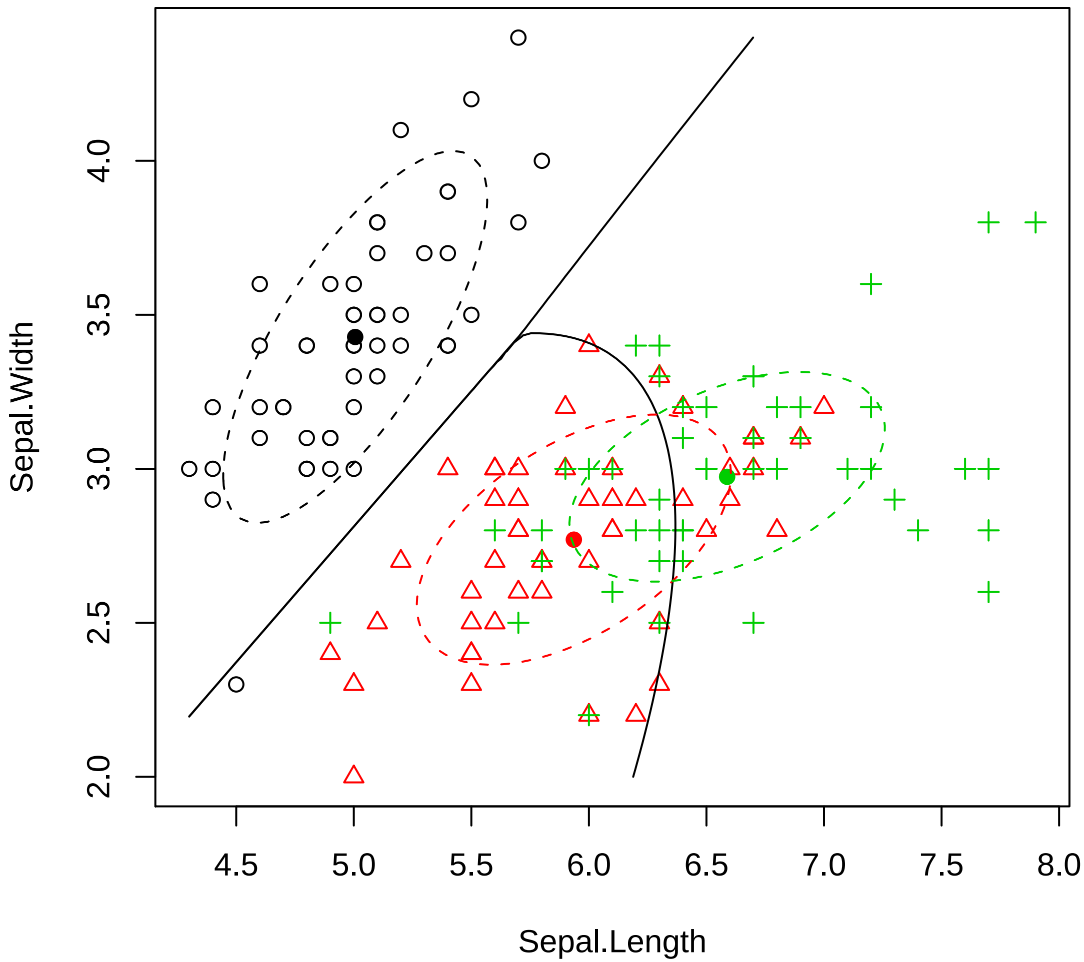
The maximum likelihood estimator for the common covariance matrix in LDA is

$$\hat{\Sigma} := \sum_k \frac{\hat{n}_k}{n} \hat{\Sigma}_k$$

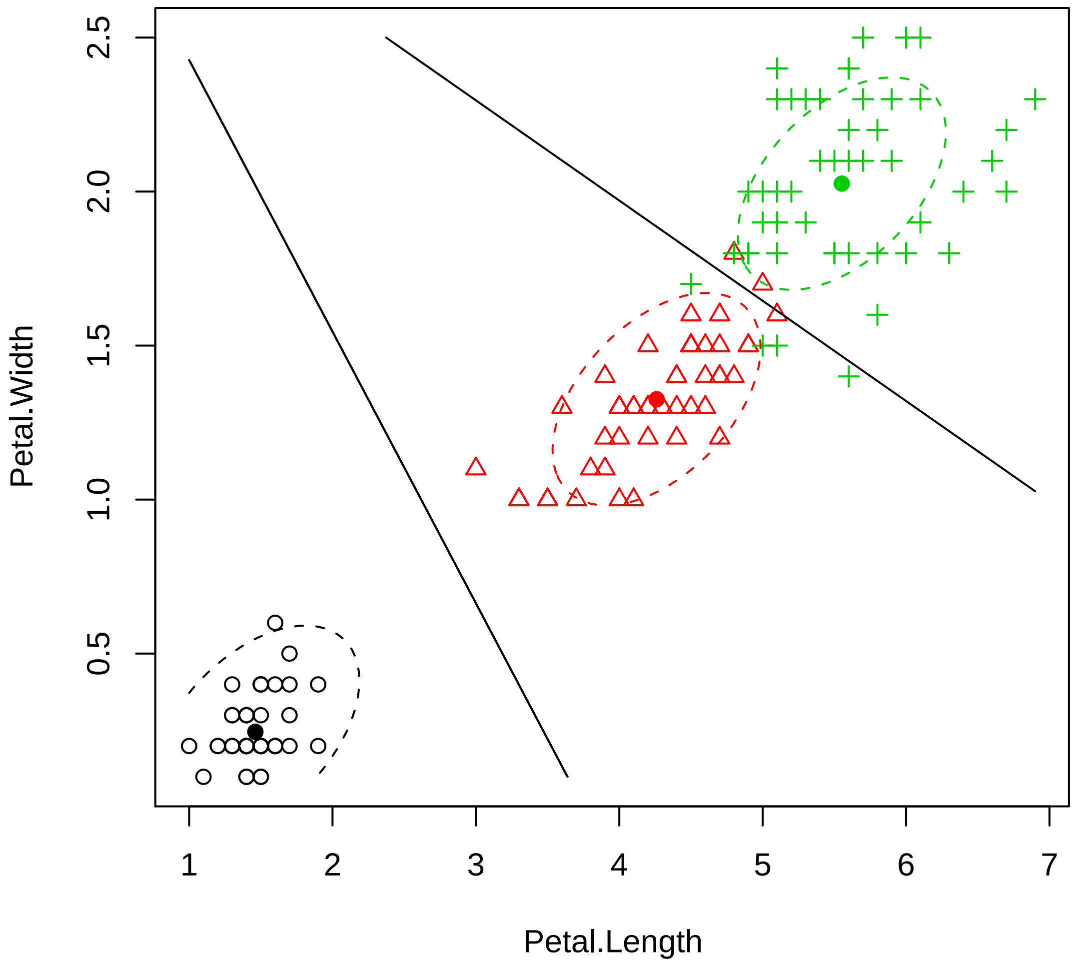
Example / Iris data / LDA



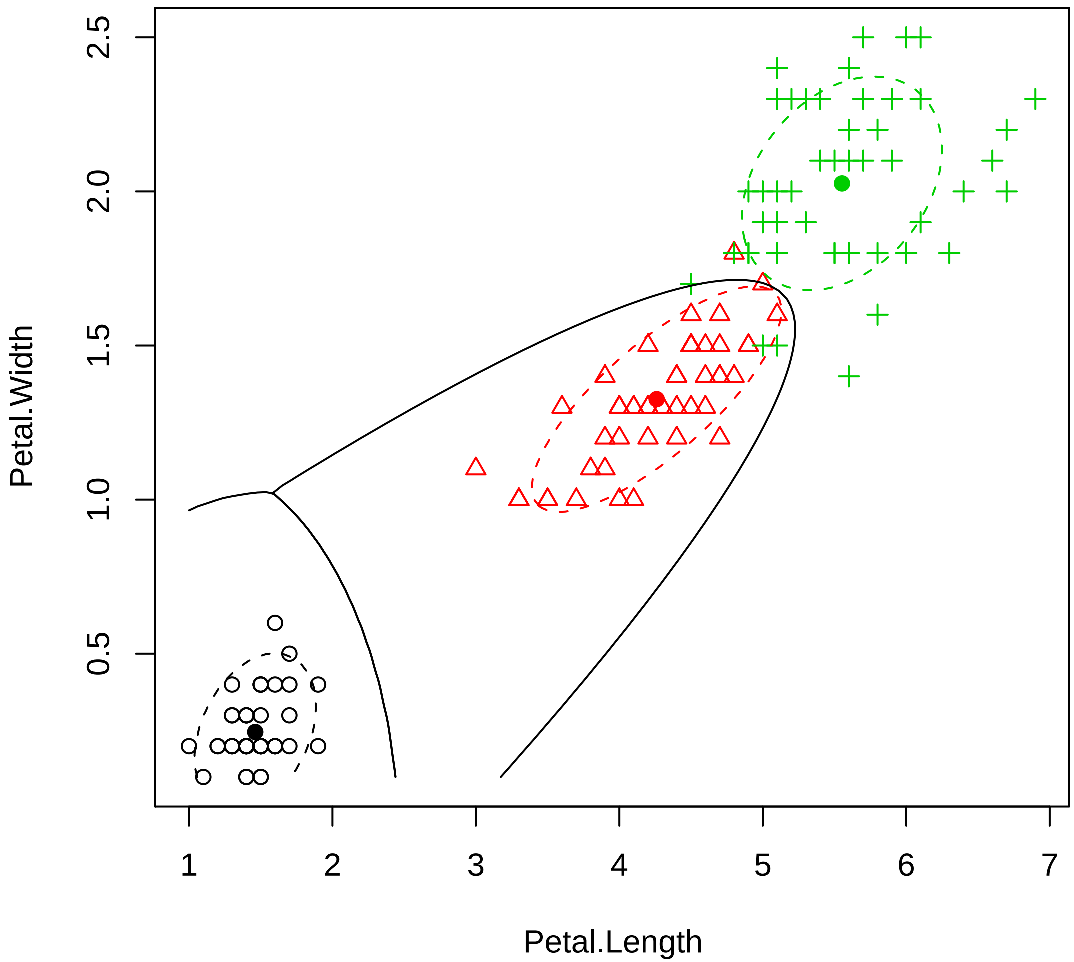
Example / Iris data / QDA



Example / Iris data / LDA



Example / Iris data / QDA



LDA coordinates

The variance matrix estimated by LDA can be used to linearly transform the data s.t. the Mahalanobis distance

$$\langle x, \hat{\Sigma}^{-1} y \rangle = x^T \hat{\Sigma}^{-1} y$$

becomes the standard euclidean distance in the transformed coordinates

$$\langle x', y' \rangle = x'^T y'$$

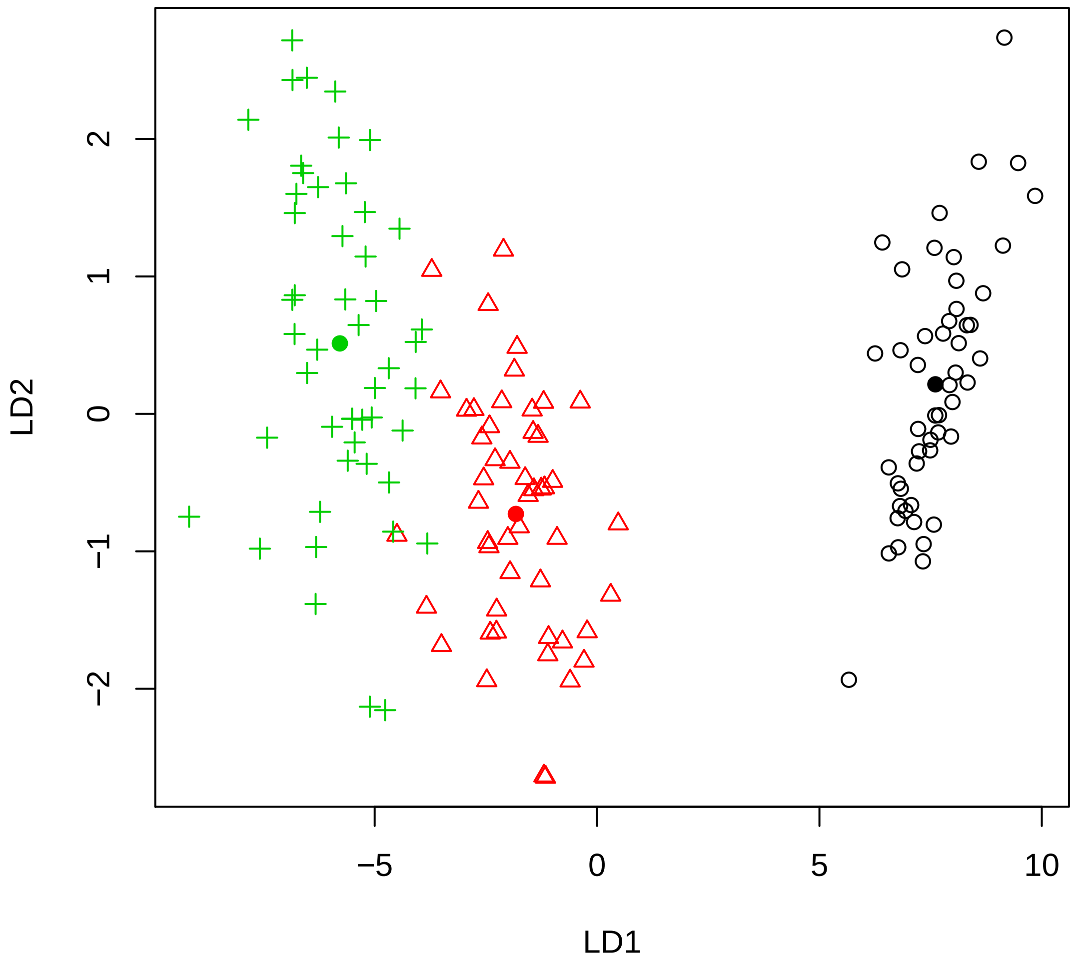
This is accomplished by decomposing $\hat{\Sigma}$ as

$$\hat{\Sigma} = U D U^T$$

with an orthonormal matrix U (i.e., $U^T = U^{-1}$) and a diagonal matrix D and setting

$$x' := D^{-\frac{1}{2}} U^T x$$

Example / Iris data / LDA coordinates



LDA vs. Logistic Regression

LDA and logistic regression use the same underlying linear model.

For LDA:

$$\log(\frac{P(Y=1|X=x)}{P(Y=0|X=x)}) = \log(\frac{\pi_1}{\pi_0}) - \frac{1}{2}\langle \mu_0 + \mu_1, \Sigma^{-1}(\mu_1 - \mu_0)\rangle + \langle x, \Sigma^{-1}(\mu_1 - \mu_0)\rangle$$
$$= \alpha_0 + \langle \alpha, x\rangle$$

For logistic regression by definition we have:

$$\log(\frac{P(Y=1|X=x)}{P(Y=0|X=x)}) = \beta_0 + \langle \beta, x\rangle$$

LDA vs. Logistic Regression

Both models differ in the way they estimate the parameters.

LDA maximizes the **complete likelihood**:

$$\prod_i p(x_i, y_i) = \underbrace{\prod_i p(x_i \mid y_i)}_{\text{normal } p_k} \underbrace{\prod_i p(y_i)}_{\text{bernoulli } \pi_k}$$

While logistic regression maximizes the **conditional likelihood** only:

$$\prod_i p(x_i, y_i) = \underbrace{\prod_i p(y_i \mid x_i)}_{\text{logistic}} \underbrace{\prod_i f(x_i)}_{\text{ignored}}$$

Summary

- For classification, **logistic regression models** of type $Y = \frac{e^{\langle X,\beta\rangle}}{1+e^{\langle X,\beta\rangle}} + \epsilon$ can be used to predict a binary Y based on several (quantitative) X.

- The **maximum likelihood estimates (MLE)** have to be computed using Newton's algorithm on the loglikelihood. The resulting procedure can be reinterpreted as **iteratively reweighted least squares (IRLS)**.

- Another simple classification model is **linear discriminant analysis (LDA)** that assumes that the cases of each class have been generated by a multivariate normal distribution with class-specific means μ_k (the class prototype) and a common covariance matrix Σ.

- The maximum likelihood parameter estimates $\hat{\pi}_k, \hat{\mu}_k, \hat{\Sigma}$ for LDA are just the sample estimates.

- Logistic regression and LDA share the same underlying linear model, but logistic regression optimizes the **conditional likelihood**, LDA the **complete likelihood**.