Machine Learning

A. Supervised Learning: Linear Models & Fundamentals
A.2. Linear Classification

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

Fri. 27.10.	(1)	0. Introduction
		A. Supervised Learning: Linear Models & Fundamentals
Fri. 3.11.	(2)	A.1 Linear Regression
Fri. 10.11.	(3)	A.2 Linear Classification
Fri. 17.11.	(4)	A.3 Regularization
Fri. 24.11.	(5)	A.4 High-dimensional Data
		B. Supervised Learning: Nonlinear Models
Fri. 1.12.	(6)	B.1 Nearest-Neighbor Models
Fri. 8.12.	(7)	B.2 Neural Networks
Fri. 15.12.	(8)	B.3 Decision Trees
Fri. 12.1.	(9)	B.4 Support Vector Machines
Fri. 19.1.	(10)	B.5 A First Look at Bayesian and Markov Networks
		C. Unsupervised Learning
Fri. 26.1.	(11)	C.1 Clustering
Fri. 2.2.	(12)	C.2 Dimensionality Reduction
Fri. 9.2.	(13)	C.3 Frequent Pattern Mining

Outline

1. The Classification Problem
2. Logistic Regression
3. Logistic Regression via Gradient Ascent
4. Logistic Regression via Newton
5. Multi-category Targets
6. Linear Discriminant Analysis

Machine Learning

Outline

1. The Classification Problem
2. Logistic Regression
3. Logistic Regression via Gradient Ascent
4. Logistic Regression via Newton
5. Multi-category Targets
6. Linear Discriminant Analysis

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany
1 / 48

The Classification Problem

Example: classifying iris plants (Anderson 1935).

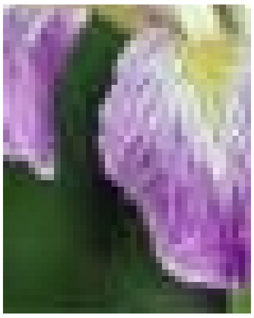
150 iris plants (50 of each species):

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

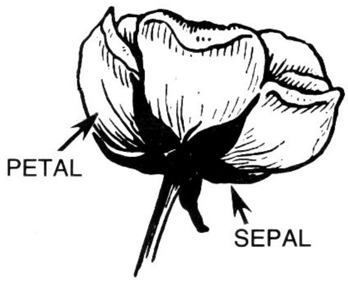
Given the lengths and widths of sepals and petals of an instance, which iris species does it belong to?

iris setosa

iris versicolor



iris virginica

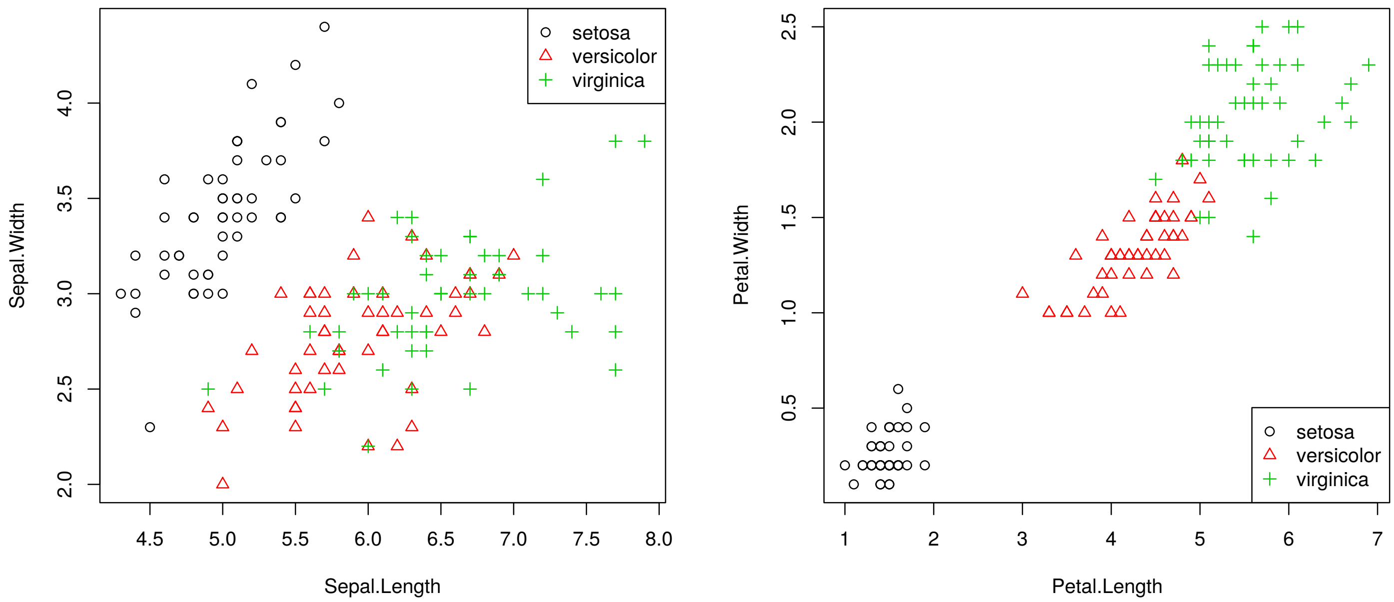


[source: iris species database, http://www.badbear.com/signa]

The Classification Problem / Data

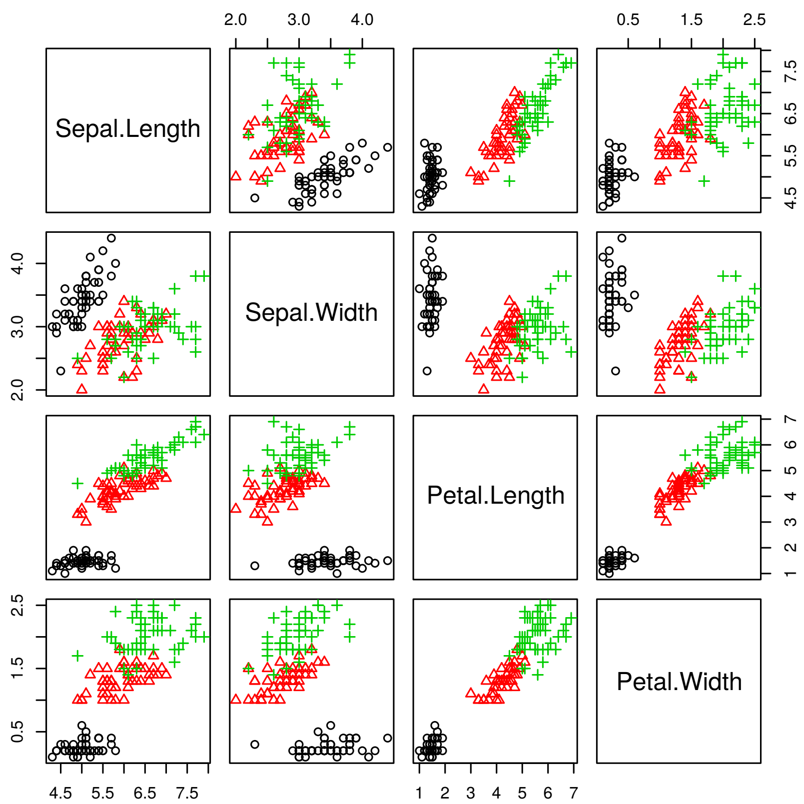
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
⋮	⋮	⋮	⋮	⋮	
51	7.00	3.20	4.70	1.40	versicolor
52	6.40	3.20	4.50	1.50	versicolor
53	6.90	3.10	4.90	1.50	versicolor
⋮	⋮	⋮	⋮	⋮	
101	6.30	3.30	6.00	2.50	virginica
⋮	⋮	⋮	⋮	⋮	
150	5.90	3.00	5.10	1.80	virginica

The Classification Problem



Machine Learning

The Classification Problem



Binary Classification

Let us start simple and consider two classes only,
e.g., target space $\mathcal{Y} := \{0, 1\}$.

Given

- a set $\mathcal{D}^{\text{train}} := \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \subseteq \mathbb{R}^M \times \mathcal{Y}$ called **training data**,

we want to estimate a model $\hat{y}(x)$ s.t. for a set $\mathcal{D}^{\text{test}} \subseteq \mathbb{R}^M \times \mathcal{Y}$ called **test set** the **test error** (here: **misclassification rate**)

$$\text{err}(\hat{y}; \mathcal{D}^{\text{test}}) := \text{mcr}(\hat{y}; \mathcal{D}^{\text{test}}) := \frac{1}{|D^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} I(y \neq \hat{y}(x))$$

is minimal.

Note: $I(A) := 1$ if statement A is true, $I(A) := 0$ otherwise (**indicator function**).
$\mathcal{D}^{\text{test}}$ has (i) to be from the same data generating process and (ii) not to be available during training.

Binary Classification / Data

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species setosa
1	5.10	3.50	1.40	0.20	1
2	4.90	3.00	1.40	0.20	1
3	4.70	3.20	1.30	0.20	1
⋮	⋮	⋮	⋮	⋮	
51	7.00	3.20	4.70	1.40	0
52	6.40	3.20	4.50	1.50	0
53	6.90	3.10	4.90	1.50	0
⋮	⋮	⋮	⋮	⋮	
101	6.30	3.30	6.00	2.50	0
⋮	⋮	⋮	⋮	⋮	
150	5.90	3.00	5.10	1.80	0

Outline

Binary Classification with Linear Regression

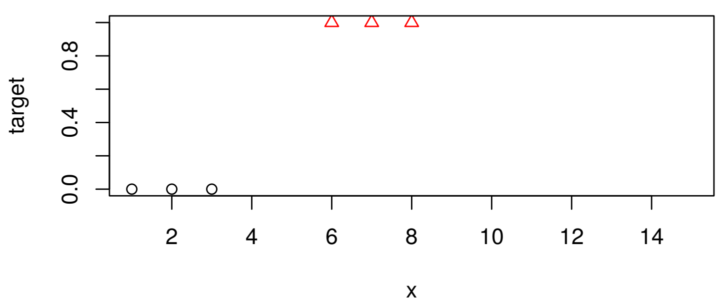
One idea could be to optimize the linear regression model

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

for RSS.

This has several problems

- It is not suited for predicting y as it can assume all kinds of intermediate values.
- It is optimizing for the wrong loss.



Binary Classification with Linear Regression

Instead of predicting Y directly, we predict

$$p(Y = 1|X; \beta) \text{ — the probability of } Y \text{ being 1 knowing } X.$$

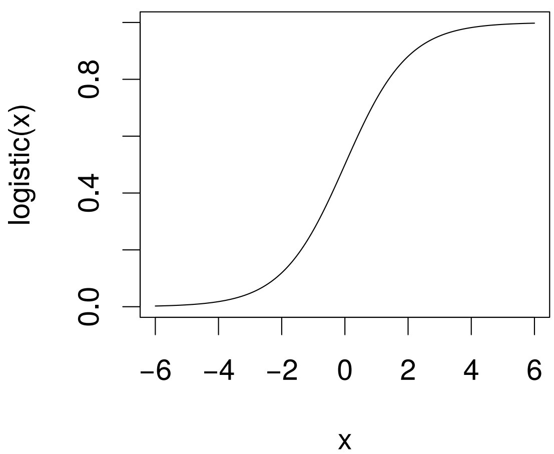
But linear regression is also not suited for predicting probabilities, as its predicted values are principally unbounded.

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

Logistic Function

Logistic function:

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



Basic properties:

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

Logistic Regression Model

$$p(Y = 1 \mid X;\ \beta) = \text{logistic}(\langle X, \beta \rangle) + \epsilon = \frac{e^{\sum_{m=1}^{M} \beta_m X_m}}{1 + e^{\sum_{m=1}^{M} \beta_m X_m}} + \epsilon$$

- observed targets are converted to probabilities $0, 1$
  - probability 1 for targets $Y = 1$,
    probability 0 for targets $Y = 0$
  - ϵ is a random variable called **noise**

- predicted targets are probabilities $[0, 1]$

$$\hat{y}(x;\ \hat{\beta}) := \text{logistic}(\langle x, \hat{\beta} \rangle) = \frac{e^{\sum_{m=1}^{M} \hat{\beta}_m x_m}}{1 + e^{\sum_{m=1}^{M} \hat{\beta}_m x_m}}$$

- remember: a logistic regression model is a **classification model**
  - despite its name

Loss Function

Misclassification rate

$$\begin{aligned} \text{mcr}(\hat{\beta}; \mathcal{D}^{\text{test}}) &:= \text{mcr}(\hat{y}(.; \hat{\beta}); \mathcal{D}^{\text{test}}) \\ &= \frac{1}{|D^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} I(y \neq \hat{y}(x; \hat{\beta})) \\ &= \frac{1}{|D^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} I(y \neq I(\text{logistic}(\hat{\beta}^T x) \geq 0.5)) \end{aligned}$$

Loss Function

Misclassification rate

$$\begin{aligned}
\text{mcr}(\hat\beta; \mathcal{D}^{\text{test}}) &:= \text{mcr}(\hat y(.; \hat\beta); \mathcal{D}^{\text{test}}) \\
&= \frac{1}{|D^{\text{test}}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}} I(y \neq \hat y(x; \hat\beta)) \\
&= \frac{1}{|D^{\text{test}}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}} I(y \neq I(\text{logistic}(\hat\beta^T x) \geq 0.5))
\end{aligned}$$

is unsuited as loss function for minimization as it is not continuous.

Use a continuous **proxy loss** instead, e.g., adhoc

$$\begin{aligned}
\ell(\hat y; \mathcal{D}^{\text{test}})) = \frac{1}{|D^{\text{test}}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}} & I(y=0)\ \text{logistic}(\hat\beta^T x) \\
& + I(y=1)\ (1-\text{logistic}(\hat\beta^T x))
\end{aligned}$$

Maximum Likelihood Estimator

As fit criterium, 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:

$$\begin{aligned}
L^{\text{cond}}_{\mathcal{D}}(\hat\beta) &= \prod_{n=1}^{N} p(Y=y_n\,|\,X=x_n; \hat\beta) \\
&= \prod_{n=1}^{N} p(Y=1\,|\,X=x_n; \hat\beta)^{y_n}(1-p(Y=1\,|\,X=x_n; \hat\beta))^{1-y_n}
\end{aligned}$$

Estimating Model Parameters

The last step is to estimate the model parameters $\hat{\beta}$.

This will be done by

- maximizing the **conditional likelihood function** $L_{\mathcal{D}}^{\textbf{cond}}$
  which is equivalent to
- maximizing the **log likelihood** $log(L_{\mathcal{D}}^{\textbf{cond}})$

This can be done with any optimization technique. We will have a closer look at

- Gradient Ascent
  - = Gradient Descent, but for maximization:
    update direction is just the gradient.
- Newton Method

Outline

Gradient Ascent

maximize-GA$(f : \mathbb{R}^N \rightarrow \mathbb{R}, x_0 \in \mathbb{R}^N, \alpha, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:
  for $t := 1, \ldots, t_{\max}$:
    $x^{(t)} := x^{(t-1)} + \alpha \cdot \frac{\partial f}{\partial x}(x^{(t-1)})$
    if $f(x^{(t)}) - f(x^{(t-1)}) < \epsilon$:
      return $x^{(t)}$
  raise exception "not converged in $t_{\max}$ iterations"

For maximizing function f instead of minimizing it go to the positive direction of the gradient.

Gradient Ascent for the Loglikelihood

$$\begin{aligned}
\log L_{\mathcal{D}}^{\text{cond}}(\hat{\beta}) &= \sum_{n=1}^{N} y_n \log p_n + (1 - y_n) \log(1 - p_n) \\
&= \sum_{n=1}^{N} y_n \log(\frac{e^{\langle x_n, \hat{\beta} \rangle}}{1 + e^{\langle x_n, \hat{\beta} \rangle}}) + (1 - y_n) \log(1 - \frac{e^{\langle x_n, \hat{\beta} \rangle}}{1 + e^{\langle x_n, \hat{\beta} \rangle}}) \\
&= \sum_{n=1}^{N} y_n(\langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})) + (1 - y_n) \log(\frac{1}{1 + e^{\langle x_n, \hat{\beta} \rangle}}) \\
&= \sum_{n=1}^{N} y_n(\langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})) + (1 - y_n)(-\log(1 + e^{\langle x_n, \hat{\beta} \rangle})) \\
&= \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})
\end{aligned}$$

Gradient Ascent for the Loglikelihood

$$\log L^{\text{cond}}_{\mathcal{D}}(\hat\beta) = \sum_{n=1}^{N} y_n \langle x_n, \hat\beta\rangle - \log(1 + e^{\langle x_n, \hat\beta\rangle})$$

$$\begin{aligned}
\nabla_\beta \log L^{\text{cond}}_{\mathcal{D}} = \frac{\partial \log L^{\text{cond}}_{\mathcal{D}}(\hat\beta)}{\partial \hat\beta} &= \sum_{n=1}^{N} y_n x_n - \frac{1}{1 + e^{\langle x_n, \hat\beta\rangle}} e^{\langle x_n, \hat\beta\rangle} x_n \\
&= \sum_{n=1}^{N} x_n (y_n - p(Y = 1 \mid X = x_n; \hat\beta)) \\
&= \mathbf{X}^T(\mathbf{y} - \mathbf{p})
\end{aligned}$$

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

Gradient Ascent for the Loglikelihood

```
learn-logreg-GA(D^train := {(x_1, y_1), ..., (x_N, y_N)}, α, t_max ∈ ℕ, ε ∈ ℝ^+):
  ℓ := log L^cond_D(β̂) := Σ_{n=1}^N y_n⟨x_n, β̂⟩ − log(1 + e^{⟨x_n, β̂⟩})
  β̂ := maximize-GA(ℓ, 0_M, α, t_max, ε)
  return β̂
```

Gradient Ascent for the Loglikelihood

learn-logreg-GA$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \alpha, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

  $X := (x_1, x_2, \ldots, x_N)^T$

  $y := (y_1, y_2, \ldots, y_N)^T$

  $\hat{\beta} := 0_M$

  $\ell := \sum_{n=1}^N y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

  for $t = 1, \ldots, t_{\max}$:

    $p := (1/(1 + e^{-\hat{\beta}^T x_n}))_{n \in 1:N}$

    $\hat{\beta} := \hat{\beta} + \alpha \cdot X^T (y - p)$

    $\ell^{\text{old}} := \ell$

    $\ell := \sum_{n=1}^N y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

    if $\ell - \ell^{\text{old}} < \epsilon$:

      return $\hat{\beta}$

  raise exception "not converged in $t_{\max}$ iterations"

Outline

Newton Algorithm

Given a function $f : \mathbb{R}^p \rightarrow \mathbb{R}$, 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:

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

Newton Algorithm

Newton Algorithm

minimize-Newton$(f : \mathbb{R}^N \rightarrow \mathbb{R}, x^{(0)} \in \mathbb{R}^N, \alpha, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:
  for $t := 1, \ldots, t_{\max}$:
    $g := \nabla f(x^{(t-1)})$
    $H := \nabla^2 f(x^{(t-1)})$
    $x^{(t)} := x^{(t-1)} - \alpha H^{-1} g$
    if $f(x^{(t-1)}) - f(x^{(t)}) < \epsilon$:
      return $x^{(t)}$
  raise exception "not converged in $t_{\max}$ iterations"

$x^{(0)}$ start value
α (fixed) step length / learning rate
$t_{\max}$ maximal number of iterations
ϵ minimum stepwise improvement

$\nabla f(x) \in \mathbb{R}^N$: gradient, $(\nabla f(x))_n = \frac{\partial}{\partial x_n} f(x)$

$\nabla^2 f(x) \in \mathbb{R}^{N \times N}$: Hessian matrix, $\nabla^2 f(x)_{n,m} = \frac{\partial^2 f}{\partial x_n \partial x_m}(x)$

Newton Algorithm for the Loglikelihood

$$\frac{\partial \log L^{\text{cond}}_{\mathcal{D}}(\hat{\beta})}{\partial \hat{\beta}} = \mathbf{X}^T(\mathbf{y} - \mathbf{p})$$

$$\frac{\partial^2 \log L^{\text{cond}}_{\mathcal{D}}(\hat{\beta})}{\partial \hat{\beta} \partial \hat{\beta}^T} = \mathbf{X}^T \mathbf{W} \mathbf{X}$$

with

$$W := diag\,(\langle p, \mathbf{1} - p \rangle)$$

and $p_n := P(Y = 1 \mid X = x_n; \hat{\beta})$.

Update rule for the Logistic Regression with Newton optimization:

$$\hat{\beta}^{(t)} := \hat{\beta}^{(t-1)} + \alpha (X^T W X)^{-1} X^T (y - p)$$

Learning Logistic Regression via Newton

learn-logreg-Newton$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \alpha, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

$\quad \ell := -\log L_{\mathcal{D}}^{\text{cond}}(\hat{\beta}) := \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

$\quad \hat{\beta} := \text{minimize-Newton}(\ell, 0_M, \alpha, t_{\max}, \epsilon)$

$\quad$ return $\hat{\beta}$

Newton Algorithm for the Loglikelihood

$x1$	$x2$	y
1	1	+
3	2	+
2	2	−
0	3	−

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

$$p^{(0)} = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}, \quad W^{(0)} = diag \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad X^T(y - p) = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

$$\left(X^T W^{(0)} X\right)^{-1} = \begin{pmatrix} 14.55 & -2.22 & -5.11 \\ -2.22 & 0.88 & 0.44 \\ -5.11 & 0.44 & 2.22 \end{pmatrix}, \quad \hat{\beta}^{(1)} = \begin{pmatrix} 2.88 \\ 0.44 \\ -1.77 \end{pmatrix}$$

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 curves**

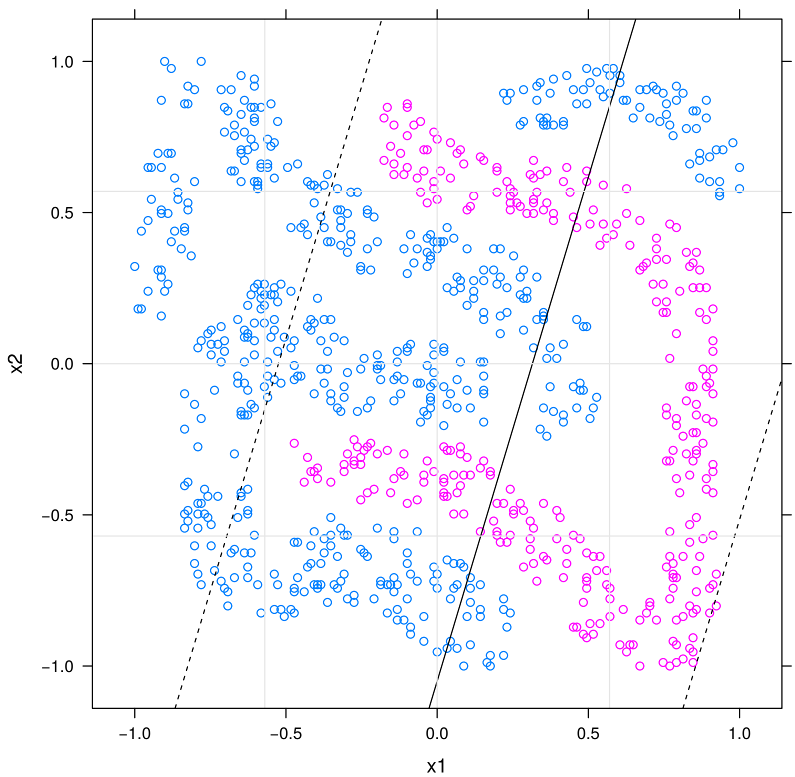
$$\hat{p}(Y = 1 \mid X) = p_0$$

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

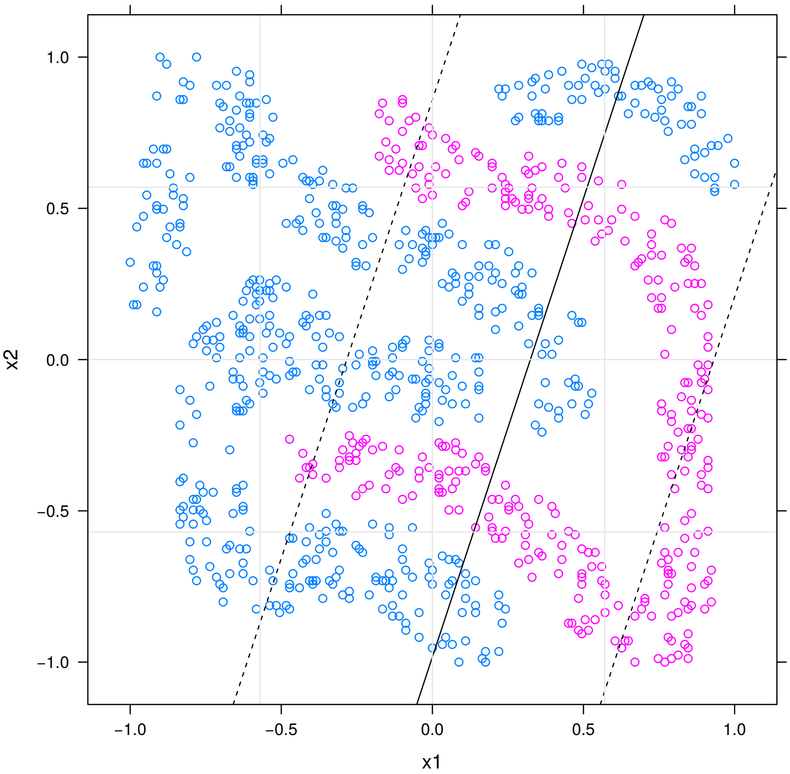
$$\langle \hat{\beta}, X \rangle = \log\left(\frac{p_0}{1 - p_0}\right)$$

For logistic regression: decision boundary and level curves are straight lines!

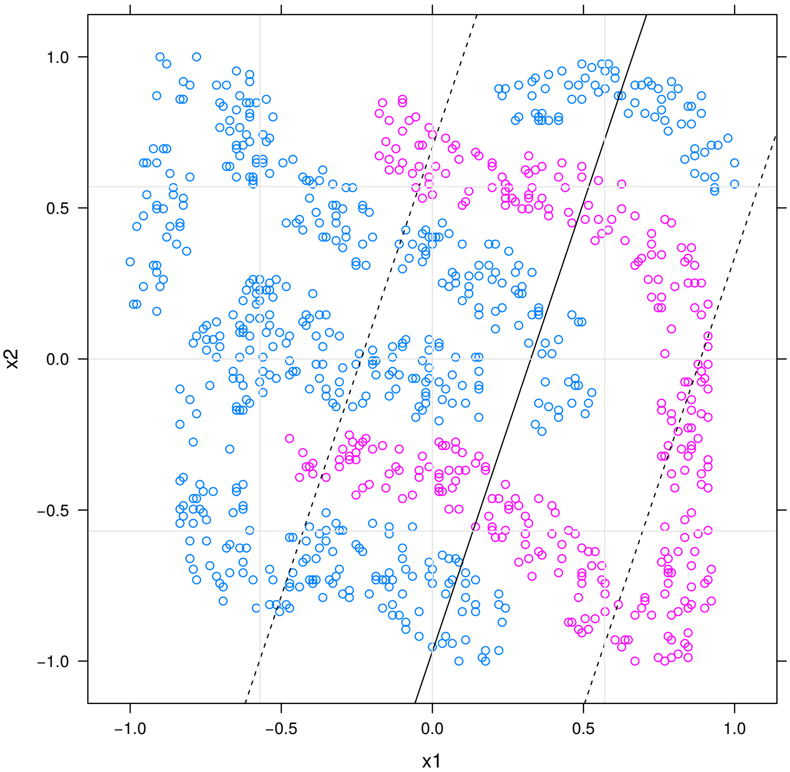
Visualization Logistic Regression Models ($t = 1$)



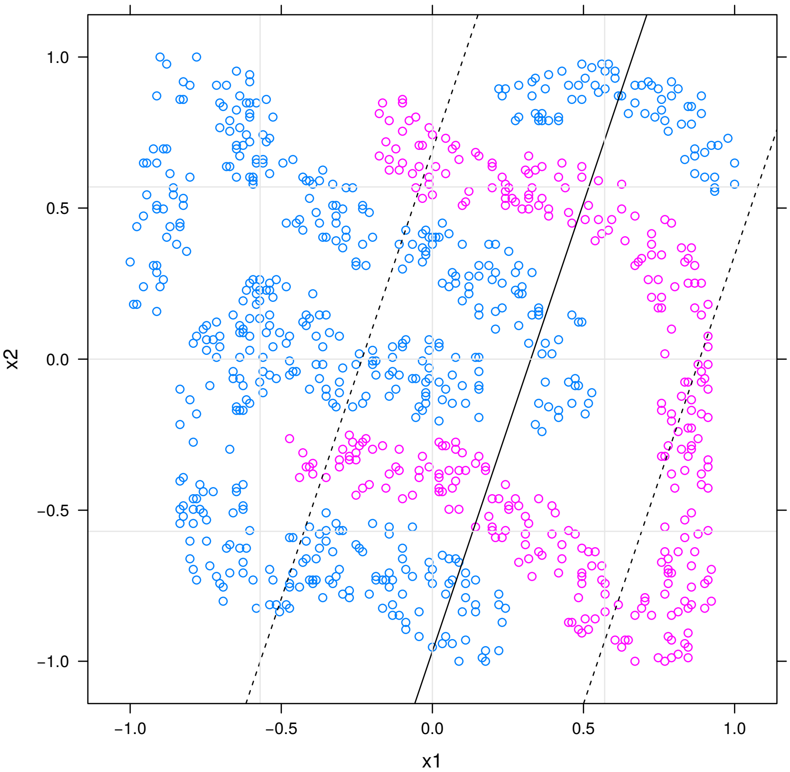
Visualization Logistic Regression Models ($t = 2$)



Visualization Logistic Regression Models ($t = 3$)



Visualization Logistic Regression Models ($t = 4$)



Outline

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)

Types of Compound Models

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

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

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

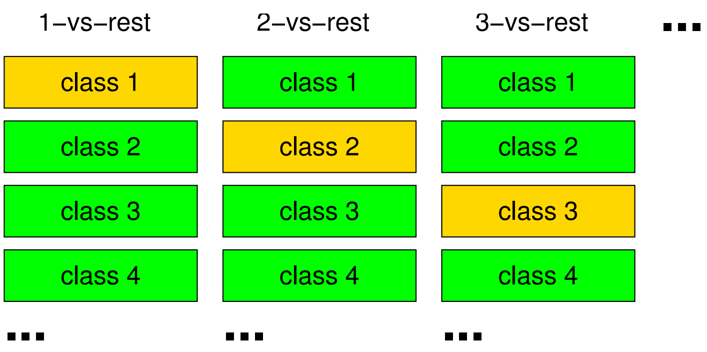
1-vs-last: one binary classifier per class (but last y_k):

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

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

Polychotomous Discrimination, k target categories

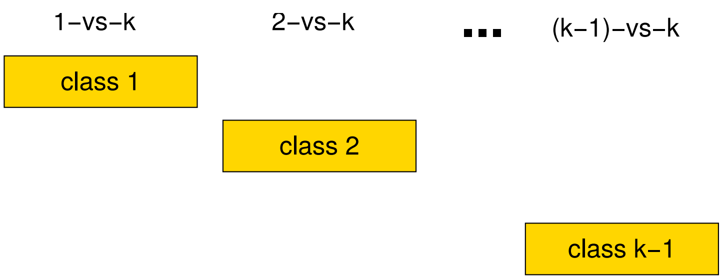
1-vs-rest construction:



k classifiers trained on N cases

kN cases in total

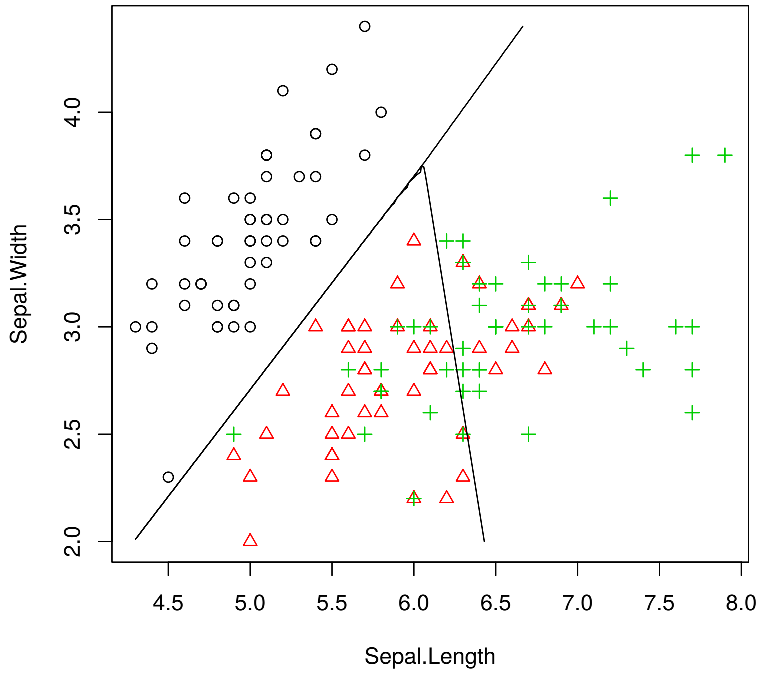
1-vs-last construction:



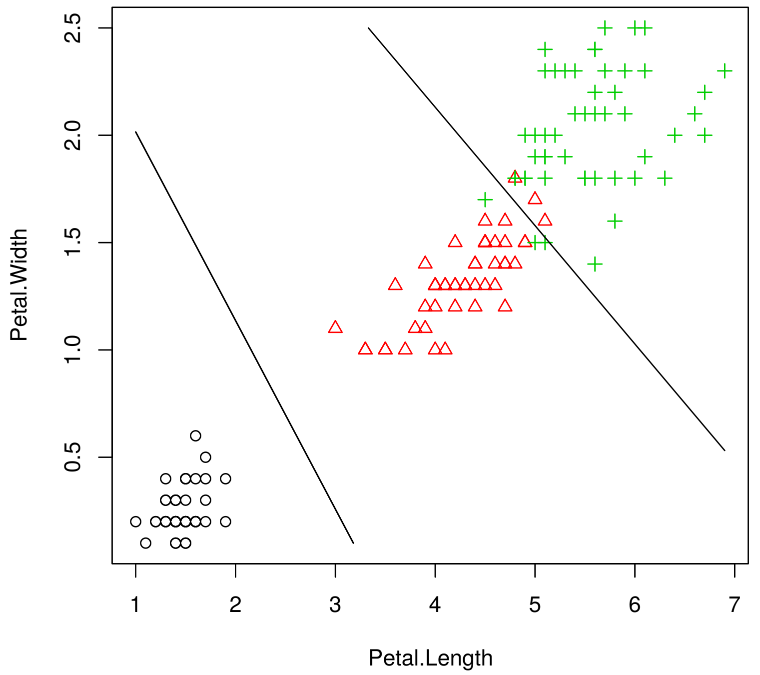
$k-1$ classifiers trained on approx. 2 N/k on average.

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

Example / Iris data / Logistic Regression



Example / Iris data / Logistic Regression



Outline

Machine Learning

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

$$X \mid Y = k \sim \mathcal{N}(X \in \mathbb{R}^M \mid \mu_k, \Sigma_k)$$

  i.e.

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

$$\mu_k \in \mathbb{R}^M, \Sigma_k \in \mathbb{R}^{M\times M}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

36 / 48

Decision Rule

Discriminant analysis predicts as follows:

$$\hat{Y} \mid X = x := \arg\max_k \pi_k p_k(x) = \arg\max_k \delta_k(x)$$

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 to the class of the nearest prototype.

Maximum Likelihood Parameter Estimates

The maximum likelihood parameter estimates are as follows:

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

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

$$\hat{\mu}_k := \frac{1}{\hat{n}_k} \sum_{n:y_n=k} x_n$$

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

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.,

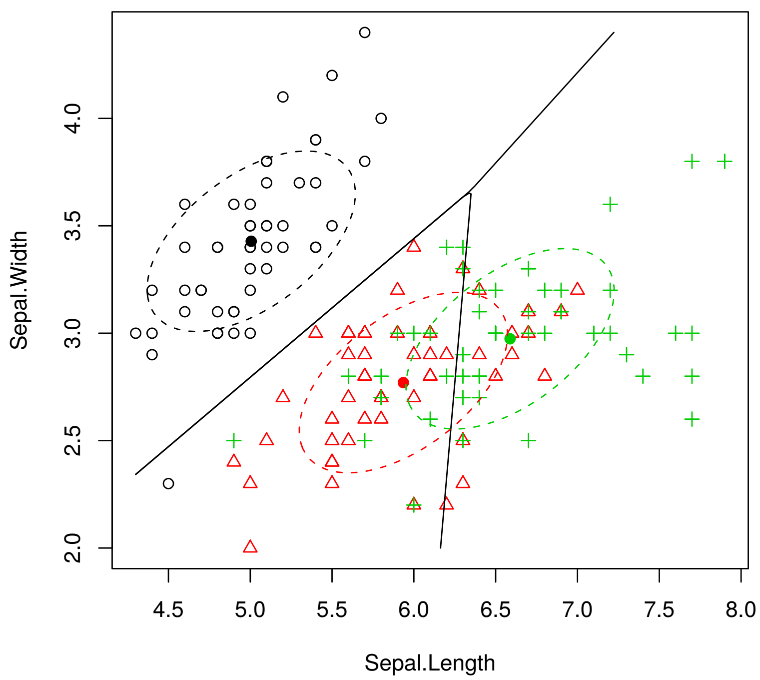
$$\Sigma_k = \Sigma_{k'} \quad \forall k, k'$$

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

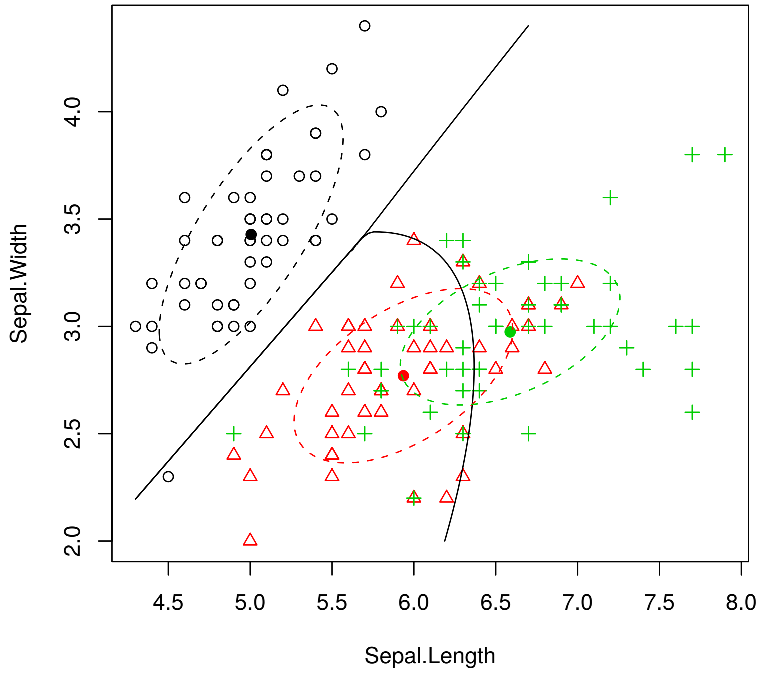
The maximum likelihood estimator for the common covariance matrix in LDA is

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

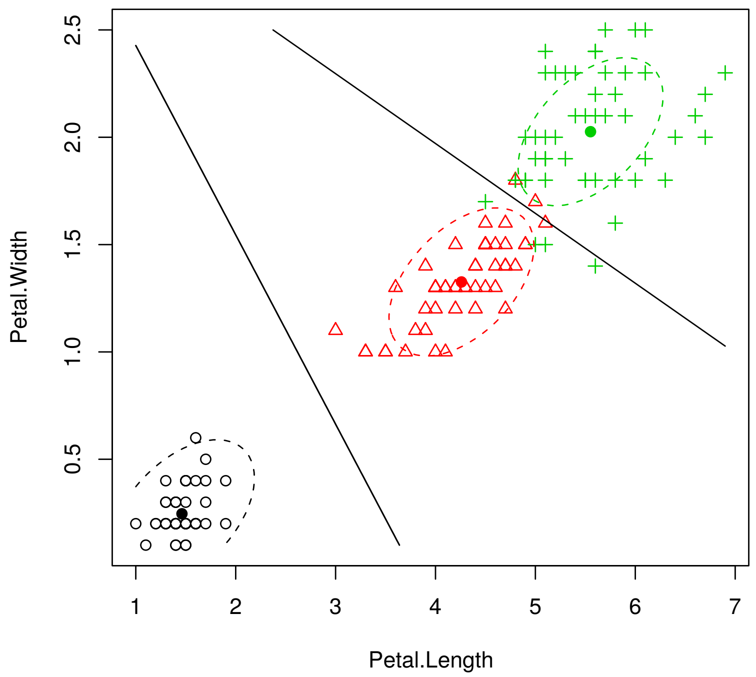
Example / Iris data / LDA



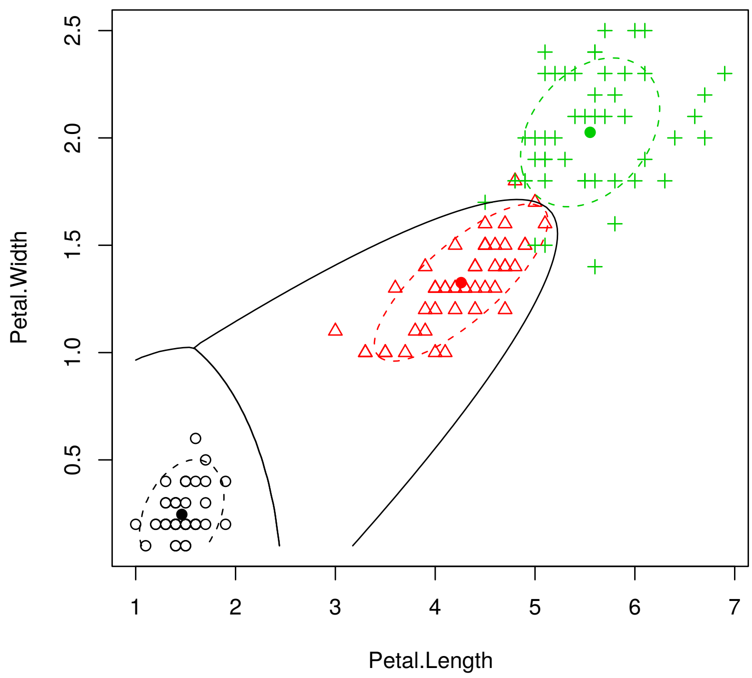
Example / Iris data / QDA



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 the **singular value decomposition** (SVD) of $\hat{\Sigma}$

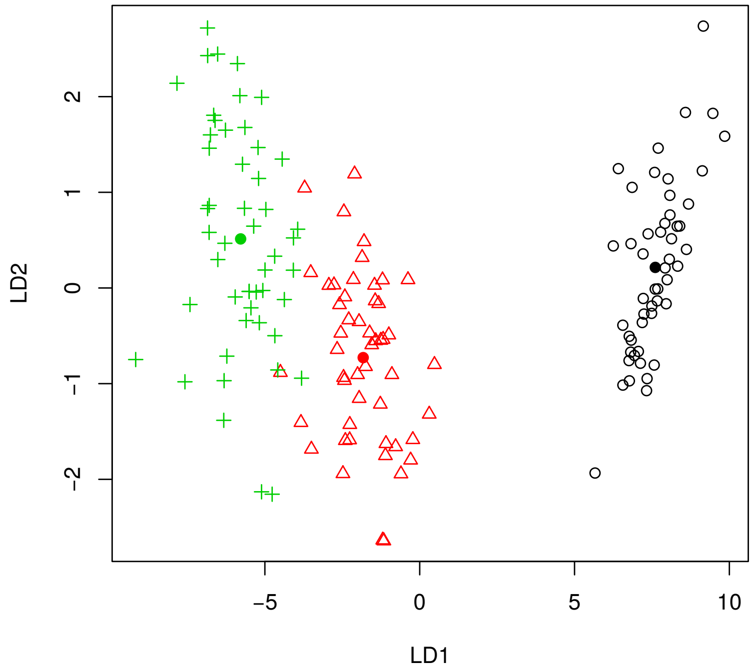
$$\hat{\Sigma} = UDU^T$$

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:

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

For logistic regression by definition we have:

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

LDA vs. Logistic Regression

Both models differ in the way they estimate the parameters.

LDA maximizes the **complete likelihood**:

$$\prod_n p(x_n, y_n) = \underbrace{\prod_n p(x_n \mid y_n)}_{\text{normal } p_k} \; \underbrace{\prod_n p(y_n)}_{\text{bernoulli } \pi_k}$$

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

$$\prod_n p(x_n, y_n) = \underbrace{\prod_n p(y_n \mid x_n)}_{\text{logistic}} \; \underbrace{\prod_n f(x_n)}_{\text{ignored}}$$

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)** can be computed using Gradient Ascent or Newton's algorithm on the loglikelihood.
- 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**.

Further Readings

- [James et al., 2013, chapter 3], [Murphy, 2012, chapter 7], [Hastie et al., 2005, chapter 3].

References

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, volume 27. Springer, 2005.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.