Machine Learning

A. Supervised Learning
A.2. Linear Classification

Lars Schmidt-Thieme

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

Outline

Outline

The Classification Problem

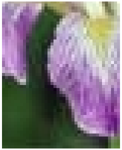
Example: classifying iris plants (Anderson 1935).

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

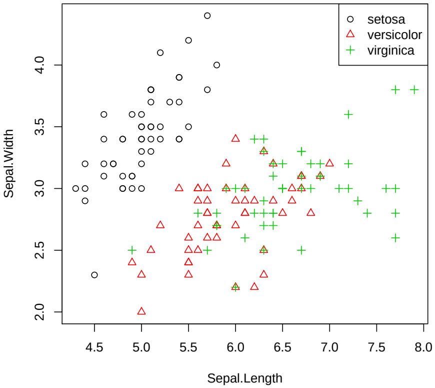


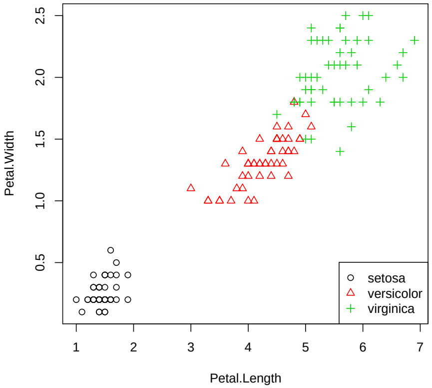
iris virginica

The Classification Problem

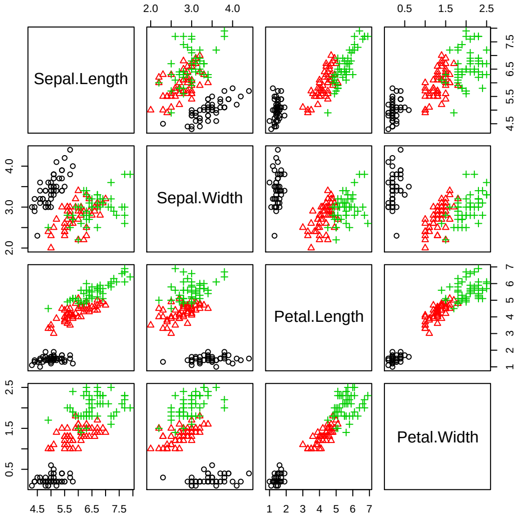
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
⋮	⋮	⋮	⋮	⋮	
51	7.00	3.20	4.70	1.40	versicolor
52	6.40	3.20	4.50	1.50	versicolor
53	6.90	3.10	4.90	1.50	versicolor
⋮	⋮	⋮	⋮	⋮	
101	6.30	3.30	6.00	2.50	virginica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

The Classification Problem





The Classification Problem



Binary Classification

Lets start simple and consider two classes only. Lets say our target Y is $\mathcal{Y} := \{0, 1\}$.
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**

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

is minimal.

Note: $\mathcal{D}^{\text{test}}$ has (i) to be from the same data generating process and (ii) not to be available during training.

Outline

Binary Classification with Linear Regression

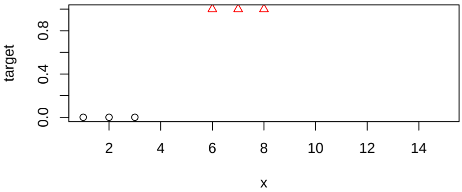
One idea could be to optimize the linear regression model

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

for RSS.

This has several problems

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



Binary Classification with Linear Regression

Instead of predicting Y directly, we predict

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

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

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

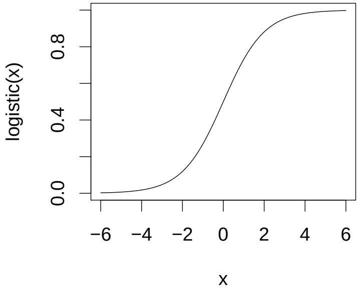
Logistic Function

Logistic function:

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

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 $(0, 0.5)$.



Maximum Likelihood Estimator

Logistic regression model:

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

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:

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

Estimating Model Parameters

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

This will be done by maximizing the conditional likelihood function $L_{\mathcal{D}}^{\text{cond}}$ which is in this case equivalent to maximizing the log likelihood $log(L_{\mathcal{D}}^{\text{cond}})$.

This can be done with any optimization technique, we will have a closer look at

- Gradient Ascent
- Newton

Gradient Ascent

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

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

Gradient Ascent for the Loglikelihood

$$
\begin{aligned}
\log L^{\text{cond}}_{\mathcal{D}}(\hat\beta) &= \sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i) \\
&= \sum_{i=1}^{n} y_i \log\left(\frac{e^{\langle x_i, \hat\beta\rangle}}{1+e^{\langle x_i, \hat\beta\rangle}}\right) + (1-y_i)\log\left(1-\frac{e^{\langle x_i, \hat\beta\rangle}}{1+e^{\langle x_i, \hat\beta\rangle}}\right) \\
&= \sum_{i=1}^{n} y_i(\langle x_i, \hat\beta\rangle - \log(1+e^{\langle x_i, \hat\beta\rangle})) + (1-y_i)\log\left(\frac{1}{1+e^{\langle x_i, \hat\beta\rangle}}\right) \\
&= \sum_{i=1}^{n} y_i(\langle x_i, \hat\beta\rangle - \log(1+e^{\langle x_i, \hat\beta\rangle})) + (1-y_i)(-\log(1+e^{\langle x_i, \hat\beta\rangle})) \\
&= \sum_{i=1}^{n} y_i\langle x_i, \hat\beta\rangle - \log(1+e^{\langle x_i, \hat\beta\rangle})
\end{aligned}
$$

Gradient Ascent for the Loglikelihood

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

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

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

Gradient Ascent for the Loglikelihood

procedure $\text{Log-Regr-GA}(L^{\text{cond}}_{\mathcal{D}} : \mathbb{R}^{P+1} \to \mathbb{R}, \hat{\beta}^{(0)} \in \mathbb{R}^{P+1}, \alpha, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$

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

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

    if $L^{\text{cond}}_{\mathcal{D}}(\hat{\beta}^{(t-1)}) - L^{\text{cond}}_{\mathcal{D}}(\hat{\beta}^{(t)})) < \epsilon$ **then**

      return $\hat{\beta}^{(t)}$

  error "not converged in $t_{\max}$ iterations"

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

procedure $\textsc{minimize-Newton}(f : \mathbb{R}^N \rightarrow \mathbb{R}, x^{(0)} \in \mathbb{R}^N, \alpha, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$
  for $t = 1, \ldots, t_{\max}$ **do**
    $x^{(t)} := x^{(t-1)} - \alpha H^{-1} \nabla_x f$
    if $f(x^{(t-1)}) - f(x^{(t)}) < \epsilon$ **then**
      return $x^{(t)}$
  error "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
$H \in \mathbb{R}^{N \times N}$	Hessian matrix, $H_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$
$\nabla_x f \in \mathbb{R}^N$	$(\nabla_x f)_i = \frac{\partial}{\partial x_i} f$

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

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

and $p_i := P(Y = 1 \,|\, X = x_i; \hat{\beta})$.

Update rule for the Logistic Regression with Newton optimization:

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

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

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

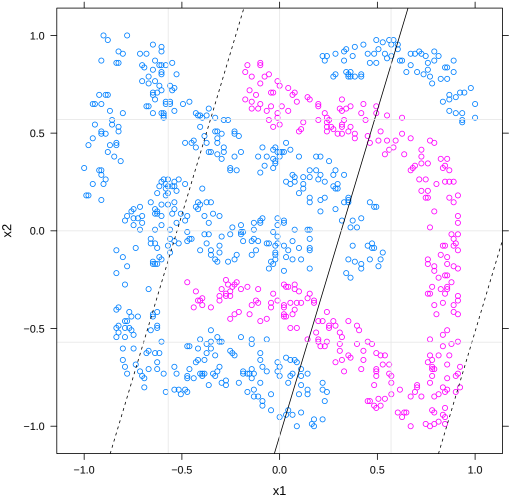
and more detailed some level lines

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

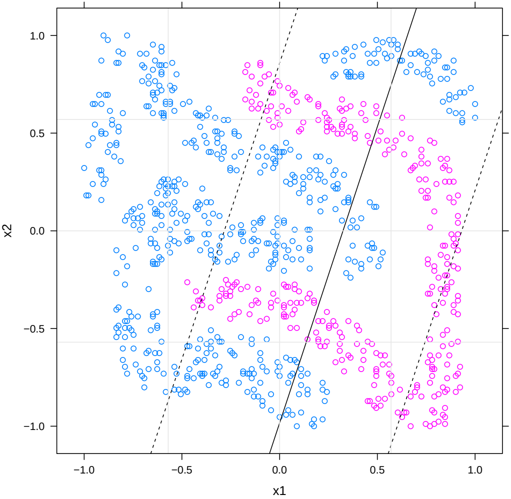
e.g., for $p_0 = 0.25$ and $p_0 = 0.75$:

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

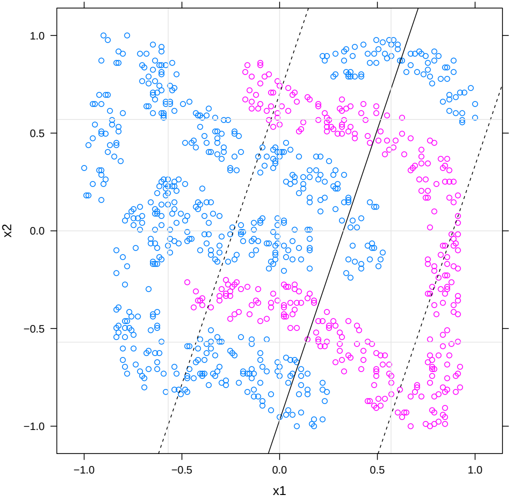
Visualization Logistic Regression Models



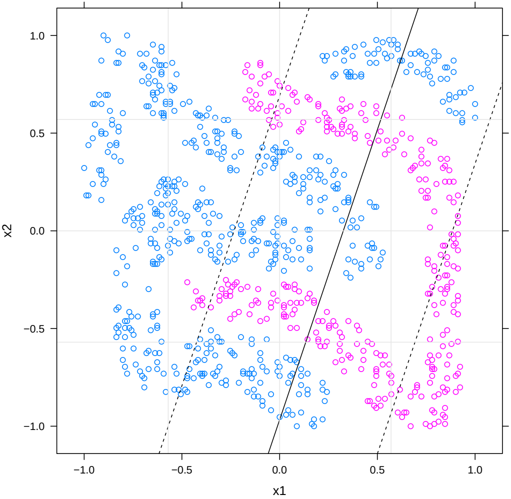
Visualization Logistic Regression Models



Visualization Logistic Regression Models



Visualization Logistic Regression Models



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:

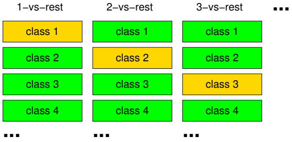
$$f_y : X \rightarrow [0,1], \quad y \in Y$$
$$f(x) := \left(\frac{f_1(x)}{\sum_{y \in Y} f_y(x)}, \ldots, \frac{f_k(x)}{\sum_{y \in Y} f_y(x)}\right)$$

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

$$f_y : X \rightarrow [0,1], \quad y \in Y, y \neq y_k$$
$$f(x) := \left(\frac{f_1(x)}{1 + \sum_{y \in Y} f_y(x)}, \ldots, \frac{f_{k-1}(x)}{1 + \sum_{y \in Y} f_y(x)}, \frac{1}{1 + \sum_{y \in Y} f_y(x)}\right)$$

Polychotomous Discrimination, k target categories

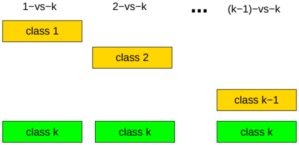
1-vs-rest construction:



k classifiers trained on N cases

kN cases in total

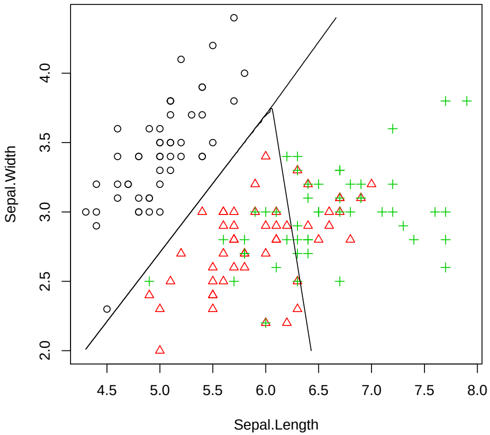
1-vs-last construction:



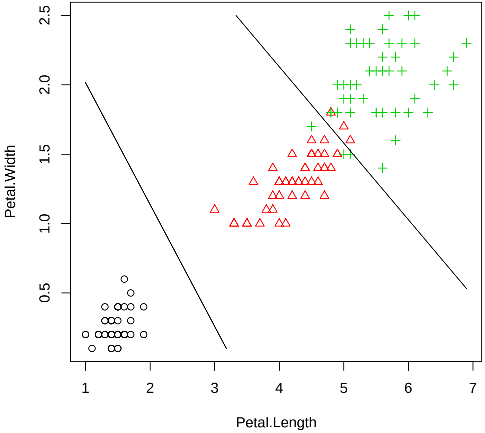
$k-1$ classifiers trained on approx. 2 N/k on average.

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

Example / Iris data / Logistic Regression



Example / Iris data / Logistic Regression



Outline

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.

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

Decision Rule

Discriminant analysis predicts as follows:

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

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 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.,

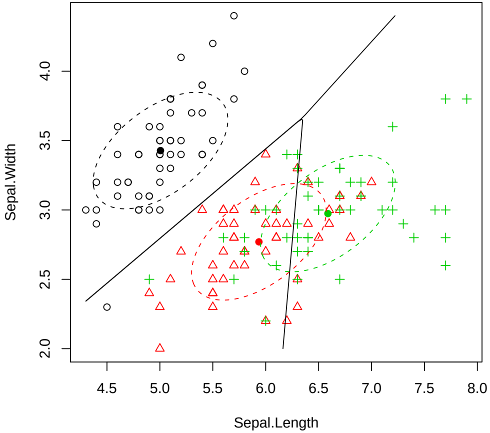
$$\Sigma_k = \Sigma_{k'} \quad \forall k, k'$$

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

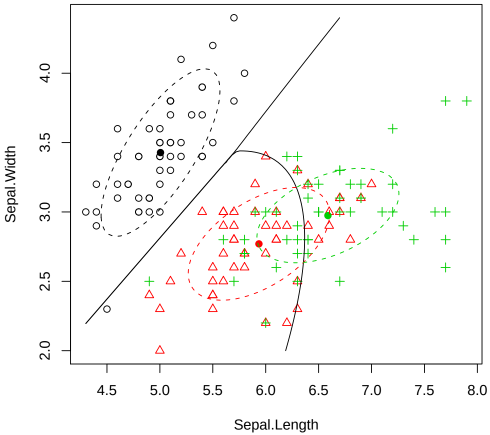
The maximum likelihood estimator for the common covariance matrix in LDA is

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

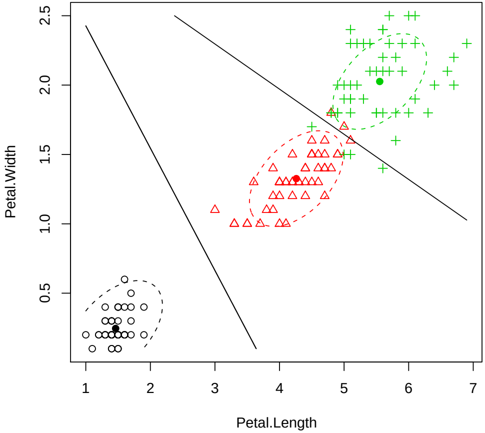
Example / Iris data / LDA



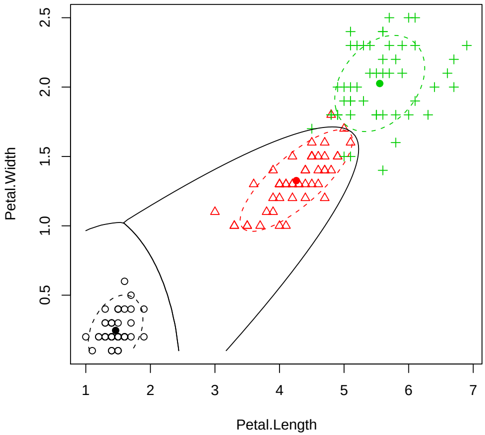
Example / Iris data / QDA



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

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

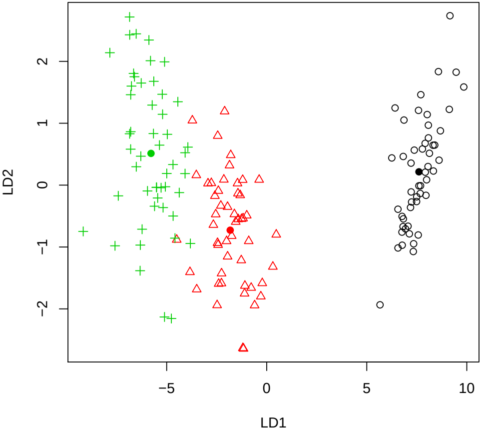
This is accomplished by decomposing $\hat{\Sigma}$ as

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

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**:

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

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

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

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)** 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

- [JWHT13, chapter 3], [Mur12, chapter 7], [HTFF05, chapter 3].

References

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin.
The elements of statistical learning: data mining, inference and prediction, volume 27.
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.