# Statistical Machine Learning

**Pier Francesco Palamara**
Department of Statistics
University of Oxford

Slide credits and other course material can be found at:
http://www.stats.ox.ac.uk/~palamara/SML19_BDI.html

# Plug-in Classification

- Consider the 0-1 loss and the risk:

$$\mathbb{E}\Big[L(Y, f(X))\Big|X = x\Big] = \sum_{k=1}^{K} L(k, f(x))\mathbb{P}(Y = k|X = x)$$

  The Bayes classifier provides a solution that minimizes the risk:

$$f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\arg\max}\, \pi_k g_k(x).$$

- We know neither the conditional density $g_k$ nor the class probability $\pi_k$!
- The **plug-in classifier** chooses the class

$$f(x) = \underset{k=1,\ldots,K}{\arg\max}\, \widehat{\pi}_k \widehat{g}_k(x),$$

- where we plugged in
  - estimates $\widehat{\pi}_k$ of $\pi_k$ and $k = 1, \ldots, K$ and
  - estimates $\widehat{g}_k(x)$ of conditional densities,
- **Linear Discriminant Analysis** is an example of plug-in classification.

# Summary: **Linear Discriminant Analysis**

- **LDA**: a plug-in classifier assuming multivariate normal conditional density $g_k(x) = g_k(x|\mu_k, \Sigma)$ for each class $k$ sharing the **same covariance** $\Sigma$:

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$
$$g_k(x|\mu_k, \Sigma) = (2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left(-\frac{1}{2}(x-\mu_k)^\top\Sigma^{-1}(x-\mu_k)\right).$$

- LDA minimizes the squared **Mahalanobis distance** between $x$ and $\widehat{\mu}_k$, offset by a term depending on the estimated class proportion $\widehat{\pi}_k$:

$$\begin{aligned} f_{\mathsf{LDA}}(x) &= \operatorname*{argmax}_{k\in\{1,\ldots,K\}} \log \widehat{\pi}_k g_k(x|\widehat{\mu}_k, \widehat{\Sigma}) \\ &= \operatorname*{argmax}_{k\in\{1,\ldots,K\}} \underbrace{\left(\log\widehat{\pi}_k - \frac{1}{2}\widehat{\mu}_k^\top\widehat{\Sigma}^{-1}\widehat{\mu}_k\right) + \left(\widehat{\Sigma}^{-1}\widehat{\mu}_k\right)^\top x}_{\text{terms depending on } k \text{ linear in } x} \\ &= \operatorname*{argmin}_{k\in\{1,\ldots,K\}} \frac{1}{2}\underbrace{(x-\widehat{\mu}_k)^\top\widehat{\Sigma}^{-1}(x-\widehat{\mu}_k)}_{\text{squared Mahalanobis distance}} - \log\widehat{\pi}_k. \end{aligned}$$

# LDA projections

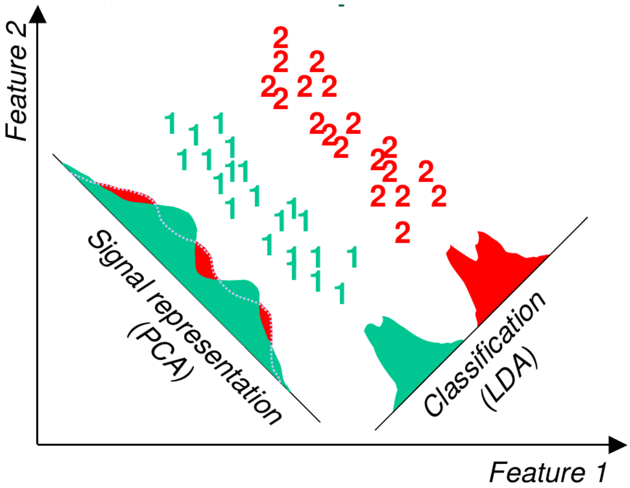
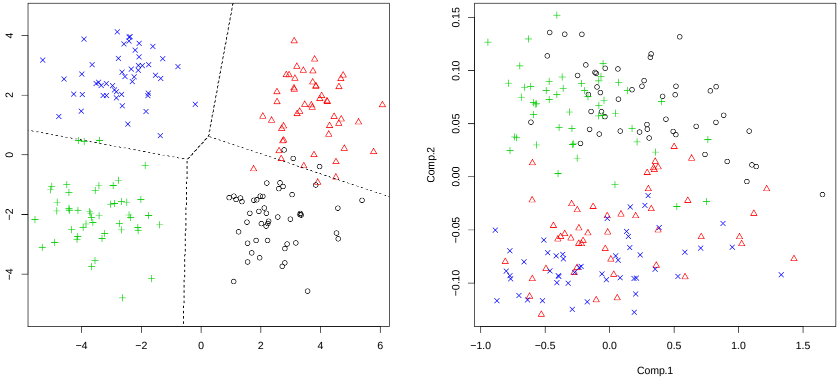


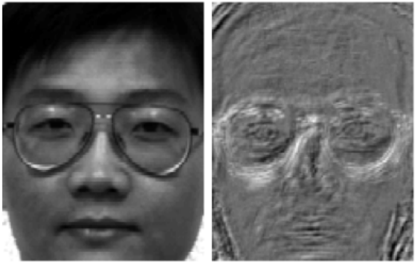

Figure by R. Gutierrez-Osuna

# LDA vs PCA projections




LDA separates the groups better.

# Fisherfaces



Eigenfaces vs. Fisherfaces, Belhumeur et al. 1997

# Conditional densities with different covariances

Given training data with $K$ classes, assume a parametric form for conditional density $g_k(x)$, where for each class

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

i.e., instead of assuming that every class has a different mean $\mu_k$ with the **same** covariance matrix $\Sigma$ (LDA), we now allow each class to have its own covariance matrix.

Considering $\log \pi_k g_k(x)$ as before,

$$\begin{aligned}
\log \pi_k g_k(x) &= \text{const} + \log(\pi_k) - \frac{1}{2}\left(\log|\Sigma_k| + (x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right) \\
&= \text{const} + \log(\pi_k) - \frac{1}{2}\left(\log|\Sigma_k| + \mu_k^T \Sigma_k^{-1}\mu_k\right) \\
&\qquad + \mu_k^T \Sigma_k^{-1} x - \frac{1}{2} x^T \Sigma_k^{-1} x \\
&= a_k + b_k^T x + x^T c_k x.
\end{aligned}$$

A **quadratic** discriminant function instead of linear.

# Quadratic decision boundaries

Again, by considering that we choose class $k$ over $k'$,

$$\begin{aligned} a_k + b_k^T x + x^T c_k x - (a_{k'} + b_{k'}^T x + x^T c_{k'} x) \\ = a_\star + b_\star^T x + x^T c_\star x > 0 \end{aligned}$$

we see that the decision boundaries of the Bayes Classifier are quadratic surfaces.

- The plug-in Bayes Classifer under these assumptions is known as the **Quadratic Discriminant Analysis** (QDA) Classifier.

# QDA

LDA classifier:

$$f_{\mathsf{LDA}}(x) = \underset{k\in\{1,\ldots,K\}}{\arg\min} \left\{ (x-\widehat{\mu}_k)^T \widehat{\Sigma}^{-1}(x-\widehat{\mu}_k) - 2\log(\widehat{\pi}_k) \right\}$$

QDA classifier:

$$f_{\mathsf{QDA}}(x) = \underset{k\in\{1,\ldots,K\}}{\arg\min} \left\{ (x-\widehat{\mu}_k)^T \widehat{\Sigma}_k^{-1}(x-\widehat{\mu}_k) - 2\log(\widehat{\pi}_k) + \log(|\widehat{\Sigma}_k|) \right\}$$

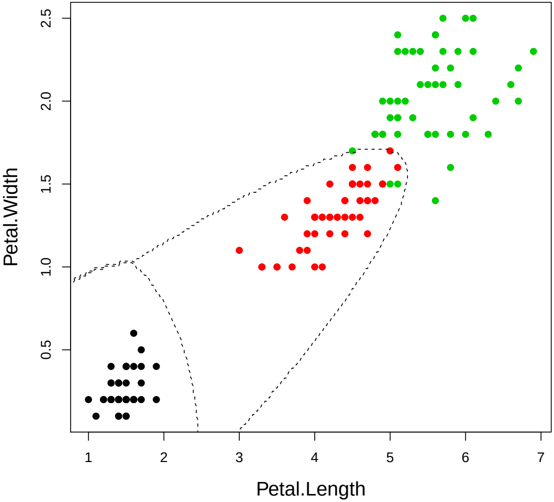
for each point $x \in \mathcal{X}$ where the plug-in estimate $\widehat{\mu}_k$ is as before and $\widehat{\Sigma}_k$ is (in contrast to LDA) estimated for each class $k = 1, \ldots, K$ separately:

$$\widehat{\Sigma}_k = \frac{1}{n_k} \sum_{j:y_j=k} (x_j - \widehat{\mu}_k)(x_j - \widehat{\mu}_k)^T.$$

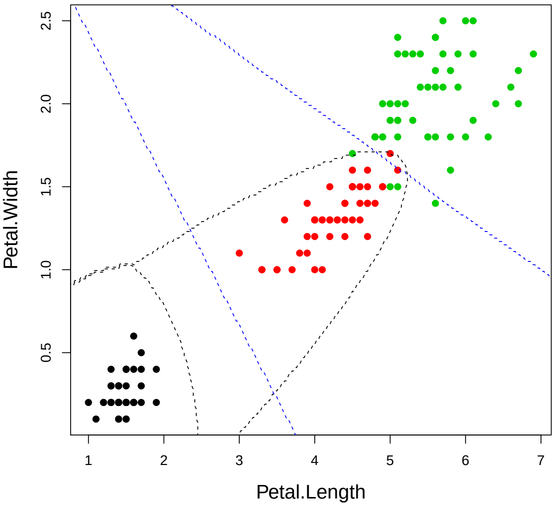
## Computing and plotting the QDA boundaries.

```
##fit QDA
iris.qda <- qda(x=iris.data,grouping=ct)

##create a grid for our plotting surface
x <- seq(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y),0)
m <- length(x)
n <- length(y)


iris.qdp <- predict(iris.qda,z)$class
contour(x,y,matrix(iris.qdp,m,n),
        levels=c(1.5,2.5), add=TRUE, d=FALSE, lty=2)
```

# Iris example: QDA boundaries



# Iris example: QDA boundaries



# LDA or QDA?

- Having seen both LDA and QDA in action, it is natural to ask which is the "better" classifier.
- If the covariances of different classes are very distinct, QDA will probably have an advantage over LDA.
- Parametric models are only ever approximations to the real world, allowing **more flexible decision boundaries** (QDA) may seem like a good idea. However, there is a price to pay in terms of increased variance and potential **overfitting**.

# Regularized Discriminant Analysis

In the case where data is scarce, to fit

- LDA, need to estimate $K \times p + p \times p$ parameters
- QDA, need to estimate $K \times p + K \times p \times p$ parameters.

Using LDA allows us to better estimate the covariance matrix $\Sigma$. Though QDA allows more flexible decision boundaries, the estimates of the $K$ covariance matrices $\Sigma_k$ are more variable.
RDA combines the strengths of both classifiers by regularizing each covariance matrix $\Sigma_k$ in QDA to the single one $\Sigma$ in LDA

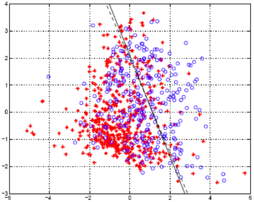
$$\Sigma_k(\alpha) = \alpha\Sigma_k + (1-\alpha)\Sigma \qquad \text{for some } \alpha \in [0,1].$$

This introduces a new parameter $\alpha$ and allows for a continuum of models between LDA and QDA to be used. Can be selected by Cross-Validation for example.

# Logistic regression

# Review

- In LDA and QDA, we estimate $p(x|y)$, but for classification we are mainly interested in $p(y|x)$
- Why not estimate that directly? Logistic regression[1] is a popular way of doing this.



[1] Despite the name "regression", we are using it for classification!

# Logistic regression

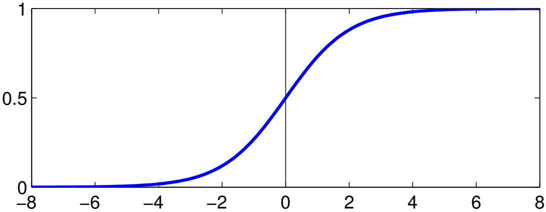
- One of the most popular methods for classification
- Linear model on the probabilities
- Dates back to work on population growth curves by Verhulst [1838, 1845, 1847]
- Statistical use for classification dates to Cox [1960s]
- Independently discovered as the perceptron in machine learning [Rosenblatt 1957]
- Main example of “discriminative” as opposed to “generative” learning
- Naïve approach to classification: we could do linear regression assigning specific values to each class. Logistic regression refines this idea and provides a more suitable model.

# Logistic regression

- Statistical perspective: consider $\mathcal{Y} = \{0, 1\}$. Generalised linear model with Bernoulli likelihood and logit link:

$$Y|X = x, a, b \sim \text{Bernoulli}\left(s(a + b^\top x)\right)$$

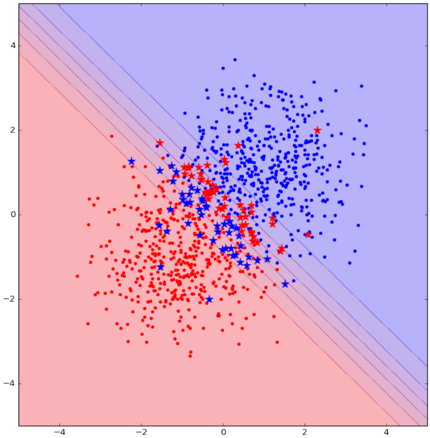
$$s(a + b^\top x) = \frac{1}{1+\exp(-(a+b^\top x))}.$$

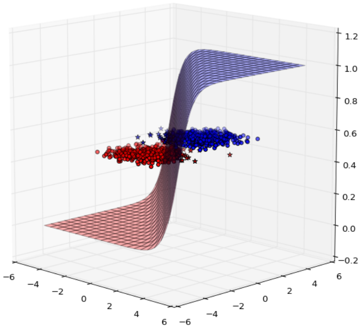



- ML perspective: a **discriminative classifier**. Consider binary classification with $\mathcal{Y} = \{+1, -1\}$. Logistic regression uses a parametric model on the conditional $Y|X$, not the joint distribution of $(X, Y)$:

$$p(Y = y|X = x; a, b) = \frac{1}{1 + \exp(-y(a + b^\top x))}.$$

# Prediction Using Logistic Regression



# Hard vs Soft classification rules

- Consider using LDA for binary classification with $\mathcal{Y} = \{+1, -1\}$. Predictions are based on linear decision boundary:

$$\begin{aligned}\widehat{y}_{\text{LDA}}(x) &= \text{sign}\left\{\log \widehat{\pi}_{+1} g_{+1}(x|\widehat{\mu}_{+1}, \widehat{\Sigma}) - \log \widehat{\pi}_{-1} g_{-1}(x|\widehat{\mu}_{-1}, \widehat{\Sigma})\right\} \\ &= \text{sign}\left\{a + b^\top x\right\}\end{aligned}$$

  for $a$ and $b$ depending on fitted parameters $\widehat{\theta} = (\widehat{\pi}_{+1}, \widehat{\pi}_{-1}, \widehat{\mu}_{+1}, \widehat{\mu}_{-1}, \Sigma)$.
- Quantity $a + b^\top x$ can be viewed as a soft classification rule. Indeed, it is modelling the difference between the log-discriminant functions, or equivalently, the **log-odds ratio**:

$$a + b^\top x = \log \frac{p(Y = +1|X = x; \widehat{\theta})}{p(Y = -1|X = x; \widehat{\theta})}.$$

- $f(x) = a + b^\top x$ corresponds to the "confidence of predictions" and loss can be measured as a function of this confidence:
  - exponential loss: $L(y, f(x)) = e^{-yf(x)}$,
  - log-loss: $L(y, f(x)) = \log(1 + e^{-yf(x)})$,
  - hinge loss: $L(y, f(x)) = \max\{1 - yf(x), 0\}$.

# Linearity of log-odds and logistic function

- $a + b^\top x$ models the **log-odds ratio**:

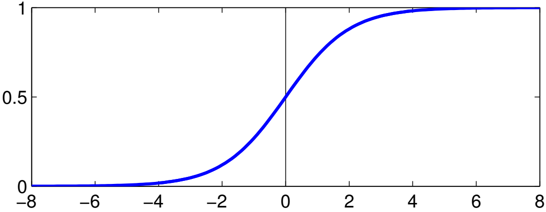
$$\log \frac{p(Y = +1|X = x; a, b)}{p(Y = -1|X = x; a, b)} = a + b^\top x.$$

- Solve explicitly for conditional class probabilities (using $p(Y = +1|X = x; a, b) + p(Y = -1|X = x; a, b) = 1$):

$$p(Y = +1|X = x; a, b) = \frac{1}{1 + \exp(-(a + b^\top x))} =: s(a + b^\top x)$$

$$p(Y = -1|X = x; a, b) = \frac{1}{1 + \exp(+(a + b^\top x))} = s(-a - b^\top x)$$

where $s(z) = 1/(1 + \exp(-z))$ is the **logistic function**.



# Fitting the parameters of the hyperplane

How to learn $a$ and $b$ given a training data set $(x_i, y_i)_{i=1}^n$?

- Consider maximizing the **conditional log likelihood** for $\mathcal{Y} = \{+1, -1\}$:

$$p(Y = y_i | X = x_i; a, b) = p(y_i | x_i) = \begin{cases} s(a + b^\top x_i) & \text{if} \quad Y = +1 \\ 1 - s(a + b^\top x_i) & \text{if} \quad Y = -1 \end{cases}$$

- Noting that $1 - s(z) = s(-z)$, we can write the log-likelihood using the compact expression:

$$\log p(y_i | x_i) = \log s(y_i(a + b^\top x_i)).$$

- And the log-likelihood over the whole i.i.d. data set is:

$$\ell(a, b) = \sum_{i=1}^n \log p(y_i | x_i) = \sum_{i=1}^n \log s(y_i(a + b^\top x_i)).$$

# Fitting the parameters of the hyperplane

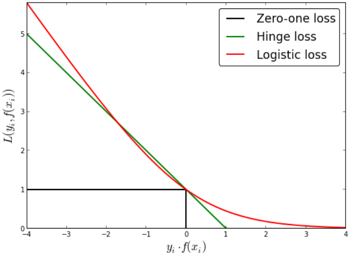
How to learn $a$ and $b$ given a training data set $(x_i, y_i)_{i=1}^n$?

- Consider maximizing the **conditional log likelihood**:

$$\ell(a,b) = \sum_{i=1}^n \log p(y_i|x_i) = \sum_{i=1}^n \log s(y_i(a + b^\top x_i)).$$

- Equivalent to minimizing the empirical risk associated with the **log loss**:

$$\widehat{R}_{\log}(f_{a,b}) = \frac{1}{n}\sum_{i=1}^n -\log s(y_i(a+b^\top x_i)) = \frac{1}{n}\sum_{i=1}^n \log(1+\exp(-y_i(a+b^\top x_i)))$$



# Could we use the 0-1 loss?

- With the 0-1 loss, the risk becomes:

$$\widehat{R}(f_{a,b}) = \frac{1}{n}\sum_{i=1}^{n} \text{step}(-y_i(a + b^\top x_i))$$

- But what is the gradient? ...

# Logistic Regression

- Log-loss is differentiable, but it is not possible to find optimal $a, b$ analytically.
- For simplicity, absorb $a$ as an entry in $b$ by appending '1' into $x$ vector, as we did before.
- Objective function:

$$\widehat{R}_{\text{log}} = \frac{1}{n}\sum_{i=1}^{n} -\log s(y_i x_i^\top b)$$

**Logistic Function**

$$s(-z) = 1 - s(z)$$
$$\nabla_z s(z) = s(z)s(-z)$$
$$\nabla_z \log s(z) = s(-z)$$
$$\nabla_z^2 \log s(z) = -s(z)s(-z)$$

- Differentiate wrt $b$:

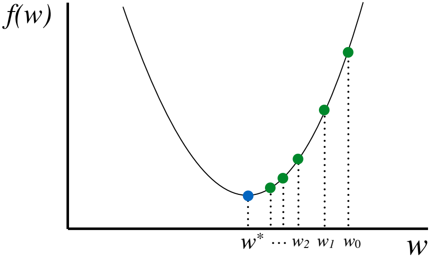
$$\nabla_b \widehat{R}_{\text{log}} = \frac{1}{n}\sum_{i=1}^{n} -s(-y_i x_i^\top b) y_i x_i$$

$$\nabla_b^2 \widehat{R}_{\text{log}} = \frac{1}{n}\sum_{i=1}^{n} s(y_i x_i^\top b) s(-y_i x_i^\top b) x_i x_i^\top \succeq 0.$$

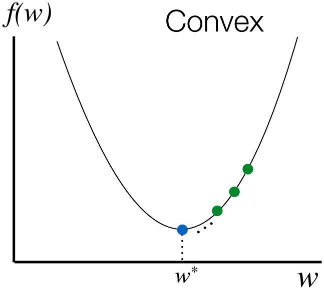
- We cannot set $\nabla_b \widehat{R}_{\text{log}} = 0$ and solve: no closed form solution. We'll use numerical methods.

# Gradient Descent

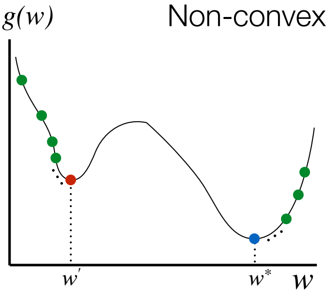
Start at a random point
**Repeat**
    Determine a descent direction
    Choose a step size
    Update
**Until** stopping criterion is satisfied



# Where Will We Converge?




Any local minimum is a global minimum




Multiple local minima may exist

**Least Squares, Ridge Regression and Logistic Regression are all convex!**

# Convexity

**How to determine convexity?** $f(x)$ is convex if

$$f^{''}(x) \geq 0$$

Examples:

$$f(x) = x^2, f^{''}(x) = 2 > 0$$

**How to determine convexity in this case?**

Matrix of second-order derivatives (Hessian)

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial {x_1}^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \dots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} & \dots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_D^2} \end{pmatrix}$$

**How to determine convexity in the multivariate case?**

If the Hessian is positive semi-definite $\mathbf{H} \succeq 0$ , then $f$ is convex.

A matrix $\mathbf{H}$ is positive semi-definite if and only if, $\forall \boldsymbol{z}$,

$$\boldsymbol{z}^T \mathbf{H} \boldsymbol{z} = \sum_{j,k} H_{j,k} z_j z_k \geq 0$$

# Logistic Regression

- Hessian is positive-definite: objective function is **convex** and there is **a single unique global minimum**.
- Many different algorithms can find optimal $b$, e.g.:
  - Gradient descent:

$$b^{\text{new}} = b + \epsilon \frac{1}{n} \sum_{i=1}^{n} s(-y_i x_i^\top b) y_i x_i$$

  - Stochastic gradient descent:

$$b^{\text{new}} = b + \epsilon_t \frac{1}{|I(t)|} \sum_{i \in I(t)} s(-y_i x_i^\top b) y_i x_i$$

    where $I(t)$ is a subset of the data at iteration $t$, and $\epsilon_t \to 0$ slowly ($\sum_t \epsilon_t = \infty, \sum_t \epsilon_t^2 < \infty$).
  - Conjugate gradient, LBFGS and other methods from numerical analysis.
  - Newton-Raphson:

$$b^{\text{new}} = b - (\nabla_b^2 \widehat{R}_{\text{log}})^{-1} \nabla_b \widehat{R}_{\text{log}}$$

    This is also called **iterative reweighted least squares**.

# Iterative reweighted least squares (IRLS)

- We can write gradient and Hessian in a more compact form. Define $\mu_i = s(x_i^\top b)$, and the diagonal matrix $\mathbf{S}$ with $\mu_i(1-\mu_i)$ on its diagonal. Also define the vector $\mathbf{c}$ where $c_i = \mathbb{1}(y_i = +1)$. Then

$$
\begin{aligned}
\nabla_b \widehat{R}_{\text{log}} &= \frac{1}{n}\sum_{i=1}^{n} -s(-y_i x_i^\top b) y_i x_i \\
&= \frac{1}{n}\sum_{i=1}^{n} x_i(\mu_i - c_i) \\
&= \mathbf{X}^\top(\mu - \mathbf{c}) \\
\nabla_b^2 \widehat{R}_{\text{log}} &= \frac{1}{n}\sum_{i=1}^{n} s(y_i x_i^\top b) s(-y_i x_i^\top b) x_i x_i^\top \\
&= \mathbf{X}^\top \mathbf{S} \mathbf{X}
\end{aligned}
$$

# Iterative reweighted least squares (IRLS)

Let $\mathbf{b}_t$ be the parameters after $t$ "Newton steps".
The gradient and Hessian at step $t$ are given by:

$$\mathbf{g}_t = \mathbf{X}^\mathsf{T}(\boldsymbol{\mu}_t - \mathbf{c}) = -\mathbf{X}^\mathsf{T}(\mathbf{c} - \boldsymbol{\mu}_t)$$
$$\mathbf{H}_t = \mathbf{X}^\mathsf{T}\mathbf{S}_t\mathbf{X}$$

The Newton Update Rule is:

$$\begin{aligned}
\mathbf{b}_{t+1} &= \mathbf{b}_t - \mathbf{H}_t^{-1}\mathbf{g}_t \\
&= \mathbf{b}_t + (\mathbf{X}^\mathsf{T}\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}(\mathbf{c} - \boldsymbol{\mu}_t) \\
&= (\mathbf{X}^\mathsf{T}\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{S}_t(\mathbf{X}\mathbf{b}_t + \mathbf{S}_t^{-1}(\mathbf{c} - \boldsymbol{\mu}_t)) \\
&= (\mathbf{X}^\mathsf{T}\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{S}_t\mathbf{z}_t
\end{aligned}$$

Where $\mathbf{z}_t = \mathbf{X}\mathbf{b}_t + \mathbf{S}_t^{-1}(\mathbf{c} - \boldsymbol{\mu}_t)$. Then $\mathbf{b}_{t+1}$ is a solution of the "weighted least squares" problem:

$$\text{minimise} \quad \sum_{i=1}^{N} S_{t,ii}(z_{t,i} - \mathbf{b}^\mathsf{T}\mathbf{x}_i)^2$$

# Linearly separable data

Assume that the data is linearly separable, i.e. there is a scalar $\alpha$ and a vector $\beta$ such that $y_i(\alpha + \beta^\top x_i) > 0$, $i = 1, \ldots, n$. Let $c > 0$. The empirical risk for $a = c\alpha$, $b = c\beta$ is

$$\widehat{R}_{\mathsf{log}}(f_{a,b}) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-cy_i(\alpha + \beta^\top x_i)))$$

which can be made arbitrarily close to zero as $c \to \infty$, i.e. soft classification rule becomes $\pm\infty$ (overconfidence) $\to$ overfitting.

**Regularization** provides a solution to this problem.

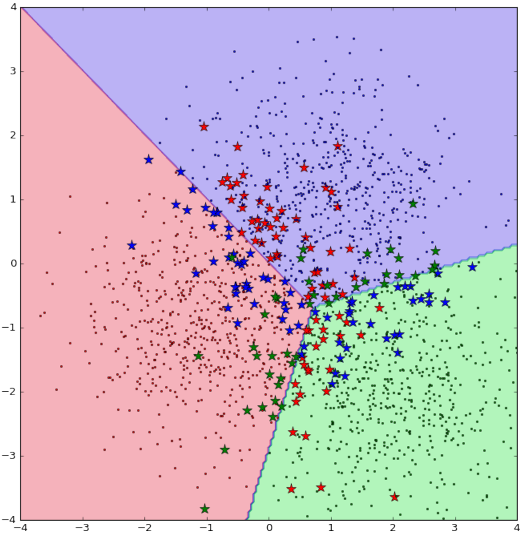
# Multi-class logistic regression

The **multi-class/multinomial** logistic regression uses the **softmax** function to model the conditional class probabilities $p\left(Y=k|X=x;\theta\right)$, for $K$ classes $k=1,\ldots,K$, i.e.,

$$p\left(Y=k|X=x;\theta\right)=\frac{\exp\left(w_k^\top x+b_k\right)}{\sum_{\ell=1}^{K}\exp\left(w_\ell^\top x+b_\ell\right)}.$$

Parameters are $\theta=(b,W)$ where $W=(w_{kj})$ is a $K\times p$ matrix of weights and $b\in\mathbb{R}^K$ is a vector of bias terms.

# Multi-class logistic regression



# Crab Dataset

```
library(MASS)
## load crabs data
data(crabs)
ct <- as.numeric(crabs[,1])-1+2*(as.numeric(crabs[,2])-1)
## project into first two LD
cb.lda <- lda(log(crabs[,4:8]),ct)
cb.ldp <- predict(cb.lda)
x <- cb.ldp$x[,1:2]
y <- as.numeric(ct==0)
eqscplot(x,pch=2*y+1,col=y+1)
```

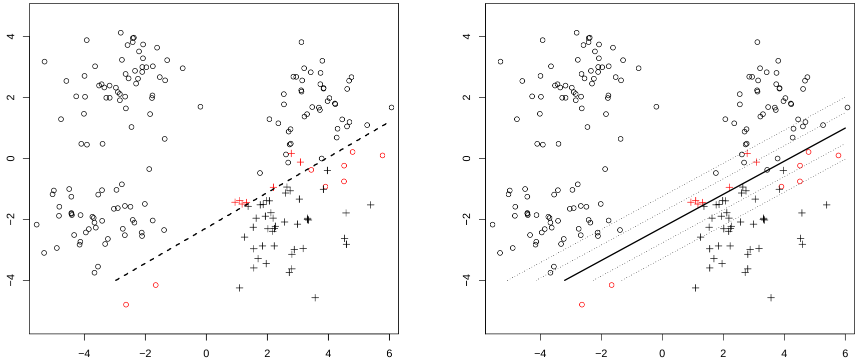
# Crab Dataset

```
## visualize decision boundary
gx1 <- seq(-6,6,.02)
gx2 <- seq(-4,4,.02)
gx <- as.matrix(expand.grid(gx1,gx2))
gm <- length(gx1)
gn <- length(gx2)
gdf <- data.frame(LD1=gx[,1],LD2=gx[,2])

lda <- lda(x,y)
y.lda <- predict(lda,x)$class
eqscplot(x,pch=2*y+1,col=2-as.numeric(y==y.lda))
y.lda.grid <- predict(lda,gdf)$class
contour(gx1,gx2,matrix(y.lda.grid,gm,gn),
   levels=c(0.5), add=TRUE,d=FALSE,lty=2,lwd=2)
```

# Crab Dataset

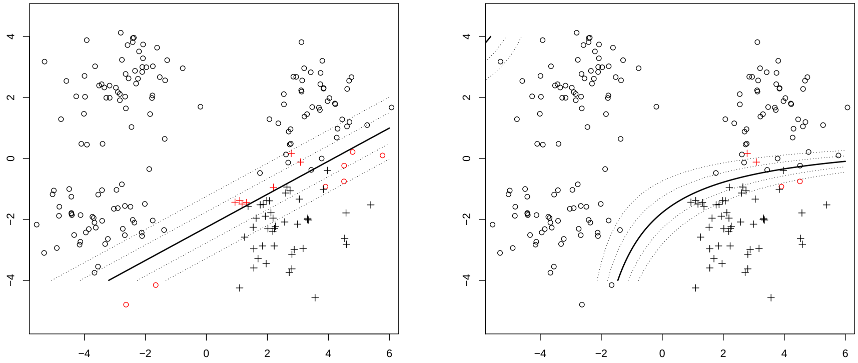
```
## logistic regression
xdf <- data.frame(x)
logreg <- glm(y ~ LD1 + LD2, data=xdf, family=binomial)
y.lr <- predict(logreg,type="response")
eqscplot(x,pch=2*y+1,col=2-as.numeric(y==(y.lr>.5)))
y.lr.grid <- predict(logreg,newdata=gdf,type="response")
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.1,.25,.75,.9), add=TRUE,d=FALSE,lty=3,lwd=1)
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.5), add=TRUE,d=FALSE,lty=1,lwd=2)

## logistic regression with quadratic interactions
logreg <- glm(y ~ (LD1 + LD2)^2, data=xdf, family=binomial)
y.lr <- predict(logreg,type="response")
eqscplot(x,pch=2*y+1,col=2-as.numeric(y==(y.lr>.5)))
y.lr.grid <- predict(logreg,newdata=gdf,type="response")
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.1,.25,.75,.9), add=TRUE,d=FALSE,lty=3,lwd=1)
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.5), add=TRUE,d=FALSE,lty=1,lwd=2)
```

# Crab Dataset : Blue Female vs. rest




Comparing LDA and logistic regression.

# Crab Dataset




Comparing logistic regression with and without quadratic interactions.

# Logistic regression Python demo

Single-class: https://github.com/vkanade/mlmt2017/blob/master/lecture11/Logistic%20Regression.ipynb

Multi-class: https://github.com/vkanade/mlmt2017/blob/master/lecture11/Multiclass%20Logistic%20Regression.ipynb