

Classification: LDA, QDA, knn, cross-validation

TMA4300: Computer Intensive Statistical Methods
(Spring 2015)
Andrea Riebler

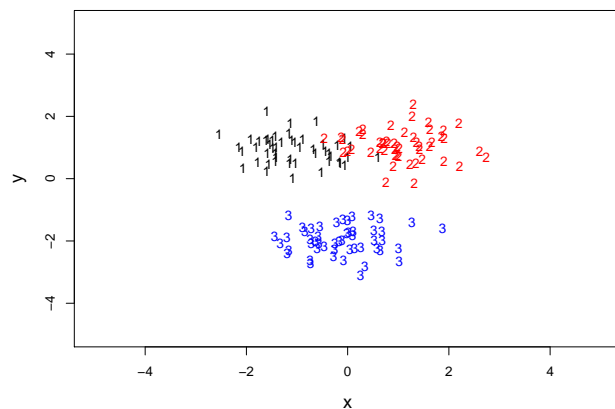
*

*Slides are based on lecture notes kindly provided by Håkon Tjelmeland.

- ⇒ Not closely related to the two first parts
- ⇒ Three topics (not closely related to each other):
 - ▶ Classification problem
 - ▶ Bootstrapping
 - ▶ Expectation-Maximization algorithm

Classification

Situation: Have observations x_1, \dots, x_n and corresponding class labels y_1, \dots, y_n , where $y_i \in \{0, 1, \dots, J-1\}$ (Training data)



New observation: x_0

Classification of the new observation

- Goal: Classify the new observation x_0 to one of the J possible classes.
- Alternatively, want to assign probabilities

$$\pi_j(x_0) = P(y_0 = j | X = x_0)$$

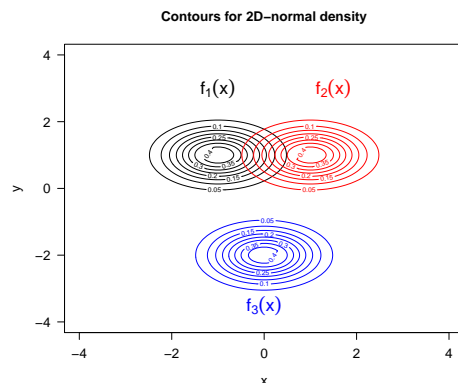
for $j = 0, \dots, J-1$.

Example: x denotes the results of a medical test with $y \in \{0, 1, 2\}$ where ($y = 0$: healthy, $y = 1$: disease 1, $y = 2$: disease 2).

Model

- We assume a distribution for x that depends on the class y :

$$f(x|y = j) = f_j(x)$$



- and prior probabilities for class j : $p_j = P(y = j)$.

Use Bayes rule

$$\begin{aligned}\pi_j(x_0) &= P(y_0 = j | X = x_0) \\ &= \frac{P(Y_0 = j, x_0)}{f(x_0)} \\ &= \frac{P(Y_0 = j, x_0)}{\sum_{i=0}^{J-1} P(Y_0 = i, x_0)} \\ &= \frac{p_j f_j(x_0)}{\sum_{i=0}^{J-1} p_i f_i(x_0)}.\end{aligned}$$

For classification, need to consider the *cost* of misclassification.

Some misclassification may be more *costly* than others.

Example: More *costly* to classify a person as healthy when he/she really has disease, than vice versa.

Model (II)

Thus, we assume a joint distribution for x and y :

$$f(x, y = j) = p_j f_j(x)$$

Comment: Must estimate $f_j(x)$ from training data

$$(x_1, y_1), \dots, (x_n, y_n),$$

perhaps also estimate p_0, \dots, p_{J-1} . For now, assume $p_0, \dots, p_{J-1}, f_0(x), \dots, f_{J-1}(x)$ known.

What is then $\pi_j(x_0)$ and \hat{y}_0 ?

Cost function

Assume a **cost function**:

- $c(i|j)$: cost of classifying a subject to class i when the true class is j .
- In particular, $c(i|i) = 0$, for $i = 0, \dots, J - 1$.

Then, it is natural to make the classification by minimising the expected cost.

See blackboard

Bayes classifier:

$$\hat{y}_0 = \operatorname{argmax}_i \{p_i f_i(x_0)\}$$

Practical challenge

$p_0, \dots, p_{J-1}, f_0(x), \dots, f_{J-1}(x)$ are unknown. Different possibilities exist:

i) estimate the unknown properties from the training data:

$$\hat{p}_j = \frac{\#\{y_i = j\}}{n}$$

- a) Estimate each $f_j(x)$ by density estimation, or
- b) assume parametric form for $f_j(x)$ and estimate its parameters.

ii) Bayesian modelling: Put prior distributions on the unknown quantities. For example

$$\begin{aligned} (p_0, \dots, p_{J-1}) &\sim \text{Dirichlet} \\ x|y = j &\sim \mathcal{N}(\mu_j, \Sigma_j) \\ \text{hyper-priors on } &\mu_j, \Sigma_j \end{aligned}$$

Linear discriminant analysis (LDA)

Consider first $\Sigma_0 = \Sigma_1 = \dots = \Sigma_{J-1} = \Sigma$. Then for given parameters, the decision rule becomes (with "0/1-loss"):

$$\begin{aligned} \hat{y}_0 &= \operatorname{argmax}_i \left\{ p_i \cdot \frac{1}{(2\pi)^{1/2}} \frac{1}{\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x_0 - \mu_i)^\top \Sigma^{-1}(x_0 - \mu_i)\right) \right\} \\ &= \operatorname{argmax}_i \left\{ -\frac{1}{2}(x_0 - \mu_i)^\top \Sigma^{-1}(x_0 - \mu_i) + \log(p_i) \right\} \\ &= \operatorname{argmax}_i \left\{ x_0^\top \Sigma^{-1} \mu_i - \frac{1}{2} \mu_i^\top \Sigma^{-1} \mu_i + \log(p_i) \right\} \\ &= \operatorname{argmax}_i \left\{ \hat{\delta}_i(x_0) \right\} \end{aligned}$$

$$\text{with } \hat{\delta}_i(x_0) = x_0^\top \Sigma^{-1} \mu_i - \frac{1}{2} \mu_i^\top \Sigma^{-1} \mu_i + \log(p_i).$$

Discriminant analysis

We will now only consider alternative i) b).

Assume

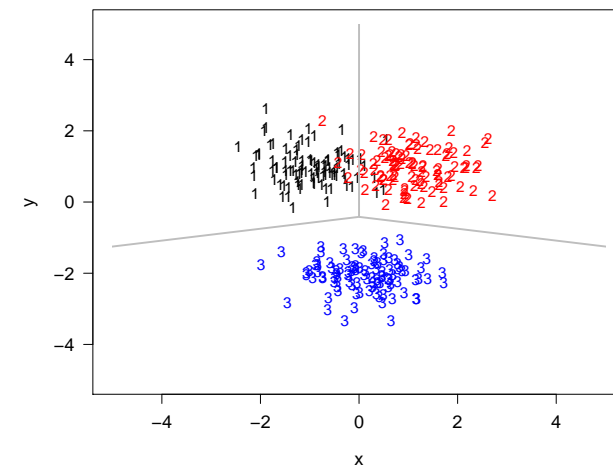
$$x|y = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

Two (extreme) alternatives:

- assume $\Sigma_0 = \Sigma_1 = \dots = \Sigma_{J-1} = \Sigma$
- different covariance matrices (more parameters to estimate)

LDA (cont.)

Note: $\hat{\delta}_i(x_0)$ is linear in x_0 . Thus the borders between the classification regions become lines/hyper-planes.



LDA: Example

Note: $\hat{\delta}_i(x_0)$ is linear in x_0 . Thus the borders between the classification regions become lines/hyper-planes.

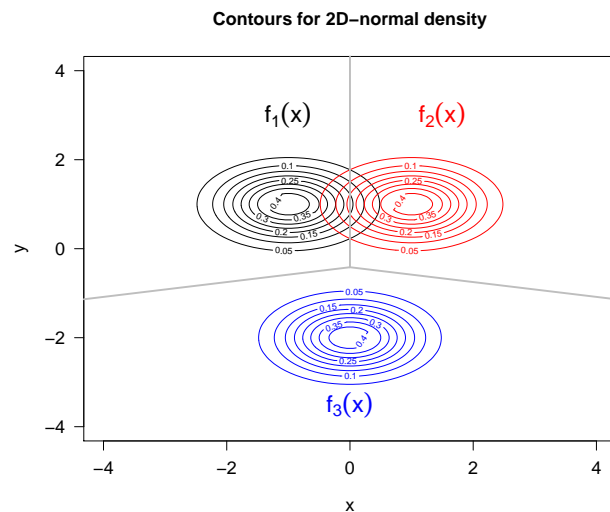
Example

- The center of mass of the individual classes are at:

$$\mu_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mu_3 = \begin{pmatrix} 0 \\ -2 \end{pmatrix}.$$

- The joint covariance matrix has the form $\Sigma = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.25 \end{pmatrix}$.
- The a-priori probabilities are equal: $p_1 = p_2 = p_3 = \frac{1}{3}$.

LDA: Example (III)



LDA: Example (II)

For the *line of separation due to Bayes' rule* we get:

$$\begin{aligned} x^\top \Sigma^{-1} \mu_i - \frac{1}{2} \mu_i^\top \Sigma^{-1} \mu_i + \log(p_i) \\ = x^\top \Sigma^{-1} \mu_j - \frac{1}{2} \mu_j^\top \Sigma^{-1} \mu_j + \log(p_j). \end{aligned}$$

- $-2x_1 + 4x_2 - 3 = 2x_1 + 4x_2 - 3$,
i.e. $x_1 = 0$ is line of separation between classes 1 and 2.
- $-2x_1 + 4x_2 - 3 = -8x_2 - 8$,
i.e. $x_2 = \frac{1}{6}x_1 - \frac{5}{12}$ is line of separation between classes 1 and 3.
- $2x_1 + 4x_2 - 3 = -8x_2 - 8$,
i.e. $x_2 = \frac{-1}{6}x_1 - \frac{5}{12}$ is line of separation between classes 2 and 3.
- All lines meet at point $(0, -\frac{5}{12})$.

Practical comments

In practice we estimate $p_0, \dots, p_{J-1}, \mu_0, \dots, \mu_{J-1}, \Sigma$ by

$$\hat{p}_j = \frac{\sum_{i=1}^n 1(y_i = j)}{n}$$

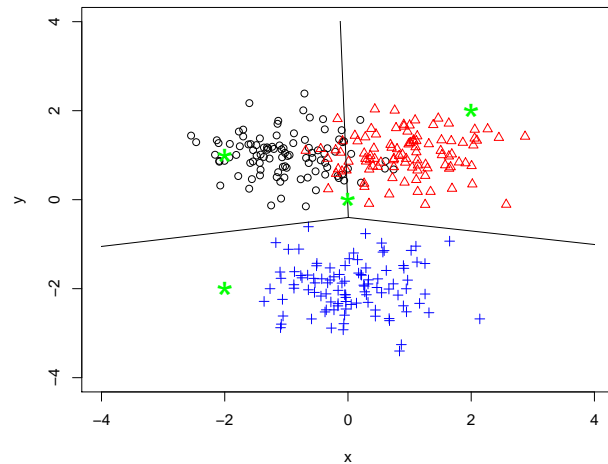
$$\hat{\mu}_j = \frac{\sum_{i=1}^n 1(y_i = j)x_i}{\sum_{i=1}^n 1(y_i = j)}$$

$$\hat{\Sigma} = \frac{1}{n-J} \sum_{i=1}^n (x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^\top$$

where $1(\cdot)$ is the indicator function. Then we use

$$\hat{\delta}_i(x_0) = x_0^\top \hat{\Sigma}^{-1} \hat{\mu}_i - \frac{1}{2} \hat{\mu}_i^\top \hat{\Sigma}^{-1} \hat{\mu}_i + \log(p_i)$$

Result of lda-function in R



Projections are done using the `predict()` function (see R-code).

k-nearest neighbour (KNN) classification

Assume we do not want to do any assumptions about $f_i(x_0)$ except that it is *smooth*. Reasonable to estimate $p_i f_i(x_0)$ by $\widehat{p_i f_i(x_0)} \approx$ number of data points (in the training set) “close to” x_0 that have $y_j = i$.

How to define “close to”?

Not a good idea: “close to” means $\|x_i - x_0\| \leq R$, because the set may be empty for some x_0 , and very large for others.

Quadratic discriminant analysis (QDA)

Consider next the case where the **covariance matrices are different** (still assuming 0/1-loss). Thus,

$$x|y = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

Then

$$\hat{y}_0 = \operatorname{argmax}_i \left\{ p_i \cdot \frac{1}{(2\pi)^{1/2}} \frac{1}{\sqrt{|\Sigma_i|}} \exp \left(-\frac{1}{2} (x_0 - \mu_i)^\top \Sigma_i^{-1} (x_0 - \mu_i) \right) \right\}$$

$$= \operatorname{argmax}_i \left\{ \underbrace{-\frac{1}{2} (x_0 - \mu_i)^\top \Sigma_i^{-1} (x_0 - \mu_i) + \log(p_i) - \frac{1}{2} \log |\Sigma_i|}_{=\hat{\delta}_i(x_0)} \right\}$$

Thus, $\hat{\delta}_i(x_0)$ is quadratic in x_0 and the **borders between the classification regions become quadratic**.

KNN classification

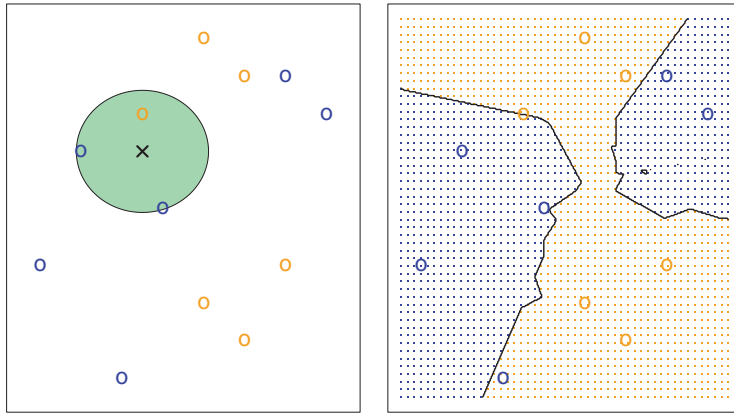
Let R depend on x_0 (and the training set), so that the **number of training values** with $\|x_i - x_0\| \leq R$ is **equal to k** for all x_0 . Then we get the **k-nearest-neighbour classifier**.

Algorithm:

1. find the k x_i 's closest to x_0 (in some norm)
2. choose \hat{y}_0 by majority vote among these k neighbours.

Here, k is a **tuning parameter**.

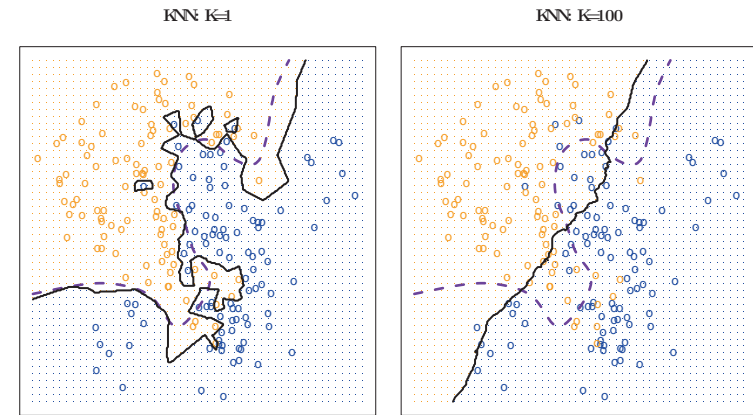
KNN classification - Example K=3



(James, Witten, Tibshirani, Hastie (2014), An Introduction to Statistical Learning, Springer, p.40)

Show animation in R: `knn.ani` in `animation` package.

k-nearest-neighbour classifiers



(James, Witten, Tibshirani, Hastie (2014), An Introduction to Statistical Learning, Springer, p.41)

Cross-validation

Consider a classification problem:

Have observed $(x_1, y_1), \dots, (x_n, y_n) \leftarrow$ training data. Have one (or more) classification rule(s):

$$\hat{y}(x_0; (x_1, y_1), \dots, (x_n, y_n))$$

How can we evaluate how good the rule is? Alternatively, how can we decide which rule is the best?

Misclassification rate

It is reasonable to focus on

- the misclassification rate

$$P(y_0 \neq \hat{y}(x_0; (x_1, y_1), \dots, (x_n, y_n)))$$

, or

- expected cost (from misclassification)

$$E[c(\hat{y}(x; (x_1, y_1), \dots, (x_n, y_n)) | y)]$$

Apparent error rate

The **apparent misclassification rate** becomes

$$\frac{1}{n} \sum_{i=1}^n \mathbf{1}(y_i \neq \hat{y}(x_i; (x_1, y_1), \dots, (x_n, y_n)))$$

This estimate becomes clearly **too optimistic** because we use the same data to “train” the classifier and to estimate the misclassification rate.

We have to take into account:

- the assumed (parametric) model may be wrong.
- uncertainty in the parameter estimates
- inherent randomness

Leave-one-out cross validation (CV)

Let $\hat{y}(x) = \hat{y}(x; (x_1, y_1), \dots, (x_n, y_n))$ denote our classifier based on all training data. Let

$$\hat{y}_{-i}(x) = \hat{y}(x; (x_1, y_1), \dots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \dots, (x_n, y_n))$$

be our classifier based on all training data except (x_i, y_i) .

Estimate the misclassification rate by:

$$\frac{1}{n} \sum_{i=1}^n \mathbf{1}(y_i \neq \hat{y}_{-i}(x_i))$$

Leave-one-out CV is computationally expensive. A cheaper variant is K-fold CV

If we have a lot of training data ...

... the effect of parameter uncertainty is negligible and we can do the following:

1. divide the (training) data in two parts: **training and test set**
2. **establish classifier from training set data**
3. do **classification for data in test data set**, and estimate misclassification rate by the fraction of misclassification in test set.

Note: If we do not have so many training data this procedure will overestimate the misclassification rate.

K-fold CV

Divide at random training data into K sets A_1, \dots, A_K of equal size (or as close as possible). Let

$$\hat{y}_{-A_k}(x) = \hat{y}(x; (x_i, y_i), i \in \bigcup_{j \neq k} A_j)$$

and estimate the misclassification rate by

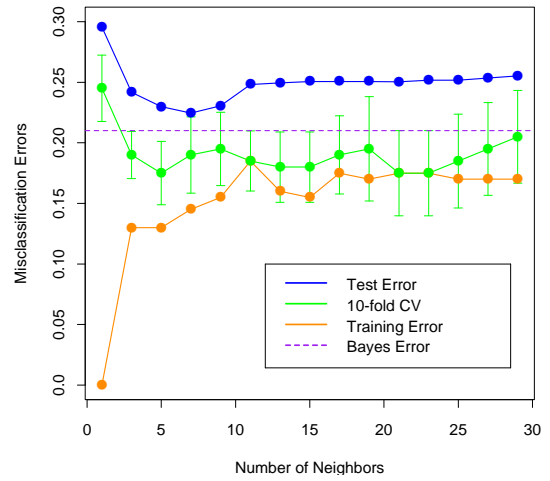
$$\frac{1}{n} \sum_{k=1}^K \left[\sum_{i \in A_k} \mathbf{1}(y_i \neq \hat{y}_{-A_k}(x_i)) \right].$$

Often, $K = 5$ or $K = 10$ is used.

Note: The tuning parameter k in the knn-classifier can be chosen using CV.

Show animation in R: `cv.anim` in `animation` package.

Misclassification as function of k



Hastie, Tibshirani, Friedman, "The elements of statistical learning", 2nd ed., p. 467