DATA11002 Introduction to Machine Learning

Kai Puolamäki

13 November 2020

Announcements

- Please submit the E1 peer-review reports today
  - you should review a total of 3 answers: 2 random answers and *your own answers* (=give points to yourself)
- Please submit Exercise Set 2 early rather than one minute late!
  - make a preliminary submission already after you have completed *some* problems
  - you can revise your submission in Moodle until the deadline
  - problems 9-12 will be covered this week, problems 13-14 Wednesday next week
- Please contact other members of your term project group as soon as possible
  - take a look at the instructions in Moodle and at least plan your schedule (first DL 6 Dec)
  - feel free to use Slack (incl. private channels)

Generative vs. discriminative learning

Generative vs. discriminative learning

- Logistic regression was an example of a **discriminative** and **probabilistic** classifier that directly models the class distribution $P(y \mid x)$
- Another probabilistic way to approach the problem is to use **generative** learning that builds amodel for the whole joint distribution $P(x, y)$ - often using the decomposition $P(x, y) = P(y)P(x \mid y)$
- Both approaches have their pros and cons:
  - Discriminative learning: only solve the task that you need to solve; may provide better accuracy since focuses on the specific learning task; optimization tends to be harder
  - Generative learning: often more natural to build models for $P(x \mid y)$ than for $P(y \mid x)$; handles missing data more naturally; optimization often easier

Generative vs. discriminative learning covered

- Examples of discriminative classifiers:
  - logistic regression (L5)
  - k-NN (L7)
  - decision trees (L7)
- Examples of generative classifiers (today):
  - naive Bayes (NB)
  - linear discriminant analysis (LDA)
  - quadratic discriminant analysis (QDA)

Generative learning

- Estimating the *class prior* $P(y)$ is usually simple
- Since $P(x, y) = P(x \mid y)P(y)$, what remains is estimating $P(x \mid y)$. In binary classification, we could now, e.g.,
  - use the positive examples to build a model for $P(x \mid Y = 1)$
  - use the negative examples to build a model for $P(x \mid Y = 0)$
- To classify a new data point x, we use the Bayes formula

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)} = \frac{P(x \mid y)P(y)}{\sum_{y'} P(x \mid y')P(y')}$$

Estimating class priors $P(y)$

- Estimating class prior $P(y)$ is usually simple
- Dataset $\{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{0, 1\}$ (binary classification)

We can estimate class priors by class counts easily:

$$\hat{P}(Y = y) = \frac{\sum_{i=1}^n I(y_i = y)}{n}$$

Additive / Laplace smoothing with pseudocount m (e.g., $m = 1$):

$$\hat{P}(Y = y) = \frac{m + \sum_{i=1}^n I(y_i = y)}{2m + n}$$

Indicator function $I(\square) = 1$ if $\square$ is true, 0 otherwise.

Normal distribution

Normal distribution

- For probabilistic models for real-valued features $x_i \in \mathbb{R}$, one basic ingredient is the *normal* or *Gaussian* distribution
- Recall that for a single real-valued random variable, the normal distribution has two parameters μ and σ^2, and density

$$N(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  - If X has this distribution, then $E[X] = \mu$ and $\mathrm{Var}[X] = \sigma^2$
- For multivariate case $x \in \mathbb{R}^p$, we shall first consider the case where individual component x_i has normal distribution with parameters μ_i and σ_i^2 and the components are independent:

$$p(x) = N(x_1 \mid \mu_1, \sigma_1^2), \ldots, (x_p \mid \mu_p, \sigma_p^2)$$

Normal distribution

- We get

$$
\begin{aligned}
p(x) &= N(x_1 \mid \mu_1, \sigma_1^2), \dots, N(x_p \mid \mu_p, \sigma_p^2) \\
&= \prod_{j=1}^{p} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right) \\
&= \frac{1}{(2\pi)^{p/2}\sigma_1 \dots \sigma_p} \exp\left(-\frac{1}{2}\sum_{j=1}^{p} \frac{(x_j-\mu_j)^2}{\sigma_j^2}\right) \\
&= \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)
\end{aligned}
$$

where $\mu = (\mu_1, \dots, \mu_p) \in \mathbb{R}^p$ and $\Sigma \in \mathbb{R}^{p \times p}$ is a diagonal matrix with $\sigma_1^2, \dots, \sigma_p^2$ on the diagonal and $|\Sigma|$ is determinant of Σ

Normal distribution

- More generally, let $\mu \in \mathbb{R}^p$, and let $\Sigma \in \mathbb{R}^{p \times p}$ be
  - symmetric: $\Sigma^T = \Sigma$
  - positive definite: $x^T \Sigma x > 0$ for all $x \in \mathbb{R}^p \setminus \{\mathbf{0}\}$
- We then define p-dimensional Gaussian density with parameter μ and Σ as

$$N(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

- If Σ is diagonal, we get the special case where x_j are independent

Normal distribution

- To understand the multivariate normal distribution, consider a surface of constant density:

$$S = \{x \in \mathbb{R}^p \mid N(x \mid \mu, \Sigma) = a\}$$

  for some a
- By definition of N, this can be written as

$$S = \left\{x \in \mathbb{R}^p \mid (x-\mu)^T \Sigma^{-1} (x-\mu) = b\right\}$$

  for some b
- Because Σ is symmetric and positive definite, so is Σ^{-1}, and this set is an ellipsoid with centre μ

Normal distribution

- More specifically, since Σ is symmetric and positive definite, it has an eigenvalue decomposition

$$\Sigma = U \Lambda U^T$$

where $\Lambda \in \mathbb{R}^{p \times p}$ is diagonal and $U \in \mathbb{R}^{p \times p}$ is orthogonal ($U^T = U^{-1}$), and further

$$\Sigma^{-1} = U \Lambda^{-1} U^T$$

- We then know from analytic geometry that for the ellipsoid

$$S = \left\{ x \in \mathbb{R}^p \mid (x - \mu)^T \Sigma^{-1} (x - \mu) = b \right\}$$

- the directions of the axes are given by the column vectors of U (eigenvectors of Σ)
- the squared lengths of the axes are given by the elements of Λ (eigenvalues of Σ)

Normal distribution

- Let $x = (x_1, \ldots, x_p)$ have normal distribution with parameters μ and Σ
- Then $E[x] = \mu$ and $E[(x_r - \mu_r)(X_s - \mu_s)] = \Sigma_{rs}$
- Hence, we call the parameter μ the *mean* and Σ the *covariance* matrix

Normal distribution

- Let $x_1, \ldots, x_n$, where $x_i = (x_{i,1}, \ldots, x_{i,p})$, be n independent samples from a p-dimensional normal distribution with unknown mean μ and covariance Σ
- The maximum likelihood (ML) estimates

$$(\hat{\mu}, \hat{\Sigma}) = \arg\max_{\mu,\Sigma} \prod_{i=1}^{n} N(x_i \mid \mu, \Sigma)$$

are given by

$$\hat{\mu}_r = \sum_{i=1}^{n} x_{i,r}/n$$

and

$$\hat{\Sigma}_{rs} = \sum_{i=1}^{n} (x_{i,r} - \hat{\mu}_r)(x_{i,s} - \hat{\mu}_s)/n$$

Gaussians in classification

- LDA, QDA, and Gaussian NB are obtained by modeling positive and negative examples both with their own Gaussian:

$$p(x \mid Y = 1) = N(x \mid \mu_1, \Sigma_1)$$
$$p(x \mid Y = 0) = N(x \mid \mu_0, \Sigma_0))$$

  where $\mu_{0/1}$ and $\Sigma_{0/1}$ are obtained for example as maximum likelihood estimates
- Decision boundary is given by

$$N(x \mid \mu_1, \Sigma_1) = N(x \mid \mu_0, \Sigma_0)$$

  or equivalently

$$\log N(x \mid \mu_1, \Sigma_1) = \log N(x \mid \mu_0, \Sigma_0)$$

Gaussians in classification

- By substituting the formula for N into

$$\log N(x \mid \mu_1, \Sigma_1) = \log N(x \mid \mu_0, \Sigma_0)$$

and simplifying we get

$$(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1) - (x-\mu_0)^T \Sigma_0^{-1}(x-\mu_0) + \log \frac{|\Sigma_0|}{|\Sigma_1|} = 0$$

- If $\Sigma_1 = \Sigma_0$ this is a linear equation, so the decision boundary is a hyperplane: **LDA**
- In general case this is a quadratic surface: **QDA**
  - In QDA, decision regions may be non-connected
- If the correlation matrices are diagonal QDA becomes Gaussian Naive Bayes: **NB**

Classification with Bayes

- Given an instance $x = (x_1, \ldots, x_p)$, and any class value $y \in \{1, \ldots, k\}$, Bayes theorem gives us

$$P(Y = y \mid X = x) = \frac{P(X = x \mid Y = y)P(Y = y)}{\sum_{y'} P(X = x \mid Y = y')P(Y = y')}$$

- A Bayes classifier then predicts class c with maximum **posterior probability** (MAP):

$$\hat{y}(x) = \arg\max_{y} P(Y = y \mid X = x)$$

- Probabilistic predictions are obtained directly from $P(Y = y \mid X = x)$

Classification with Bayes

- Since the denominator $\sum_{y'=1}^{k} P(X = x \mid Y = y')P(Y = y')$ does not depend on y, the MAP classification is the same as

$$\hat{y}(x) = \arg\max_{y} P(X = x \mid Y = y)P(Y = y)$$

- If the class prior $P(Y)$ is uniform, this simplifies to maximum likelihood (ML) prediction

$$\hat{y}(x) = \arg\max_{c} P(X = x \mid Y = y)$$

- The question becomes: where do we get $P(X = x \mid Y = y)$ from?

Gaussians in classification

- Choose $\mathrm{nonevent} = 0$ and $\mathrm{event} = 1$.
- Let I_0 and I_1 be the row indices for nonevents and events, respectively.
- We try to estimate the joint distribution $P(X = x, Y = y) = P(X = x \mid Y = y)P(Y = y)$.
- For this data $P(Y)$ is easy: $P(Y = 0) = |I_0|/(|I_0| + |I_1|) = 1/2$ and $P(Y = 1) = |I_1|/(|I_0| + |I_1|) = 1/2$.

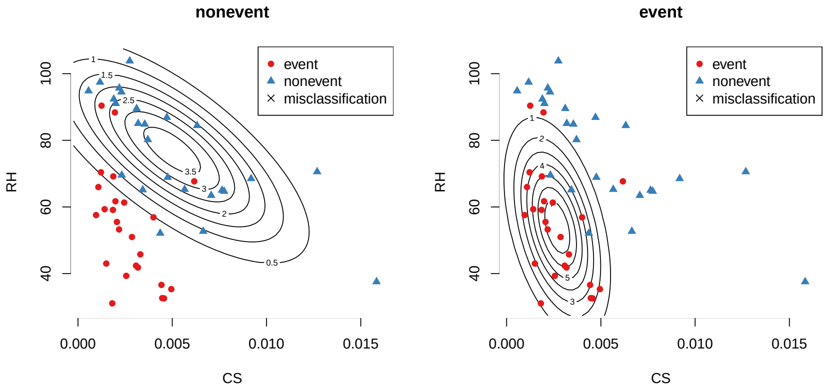
Gaussians in classification (cont.)

- ML estimates for the parameters:
  - data mean: $\hat{\mu} = \sum_{i=1}^{n} x_i / n$.
  - means for a class: $\hat{\mu}_0 = \sum_{i \in I_0} x_i / |I_0|$ and $\hat{\mu}_1 = \sum_{i \in I_1} x_i / |I_1|$.
  - class-centered covariance for all data: $\hat{\Sigma} = \sum_{i=1}^{n} (x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^T / n$.
  - class-specific covariances $\hat{\Sigma}_0 = \sum_{i \in I_0} (x_i - \hat{\mu}_0)(x_i - \hat{\mu}_0)^T / |I_0|$ and $\hat{\Sigma}_1 = \sum_{i \in I_1} (x_i - \hat{\mu}_1)(x_i - \hat{\mu}_1)^T / |I_1|$.

Gaussians in classification

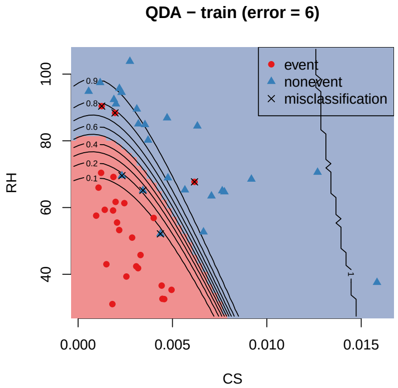
```
mu1 <- apply(npf$dtr[npf$dtr[,3]=="event",1:2],2,
             mean)
mu0 <- apply(npf$dtr[npf$dtr[,3]=="nonevent",1:2],2,mean)
mu <- apply(npf$dtr[,1:2],2,mean)
sigma1 <- var(npf$dtr[npf$dtr[,3]=="event",1:2])
sigma0 <- var(npf$dtr[npf$dtr[,3]=="nonevent",1:2])
sigma <- var(npf$dtr[,1:2]-1*(npf$dtr[,3]=="nonevent")
             %o% mu0-1*(npf$dtr[,3]=="event") %o% mu1)
```

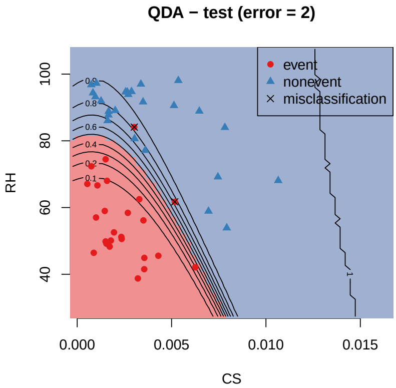
Gaussians in classification: QDA



- Left: $P(X \mid Y = 0) \sim N(X \mid \hat{\mu}_0, \hat{\Sigma}_0)$.
- Right: $P(X \mid Y = 1) \sim N(X \mid \hat{\mu}_1, \hat{\Sigma}_1)$.

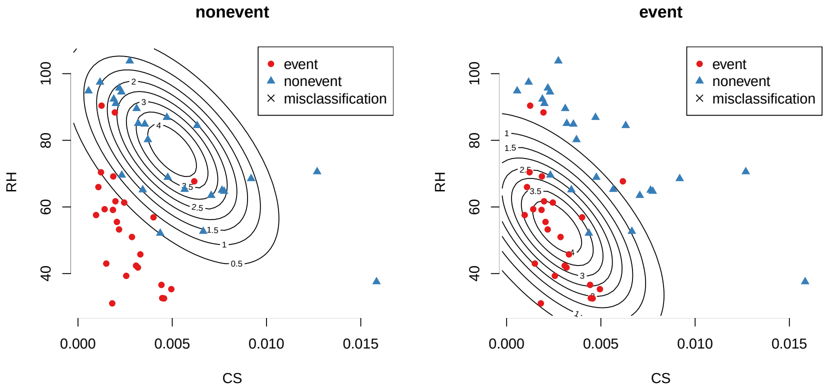
Gaussians in classification: QDA





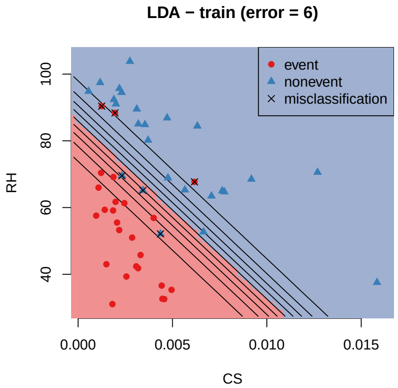
$$P(Y=1 \mid X) = \frac{P(X \mid Y=1)P(Y=1)}{P(X \mid Y=0)P(Y=0) + P(X \mid Y=1)P(Y=1)}$$

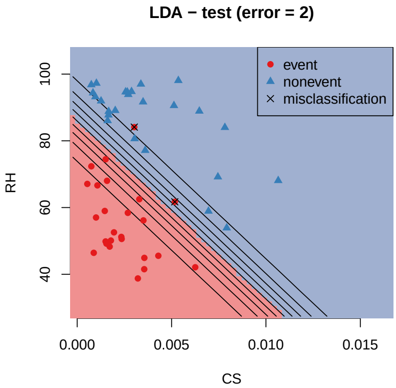
Gaussians in classification: LDA



- Left: $P(X \mid Y = 0) \sim N(X \mid \hat{\mu}_0, \hat{\Sigma})$.
- Right: $P(X \mid Y = 1) \sim N(X \mid \hat{\mu}_1, \hat{\Sigma})$.

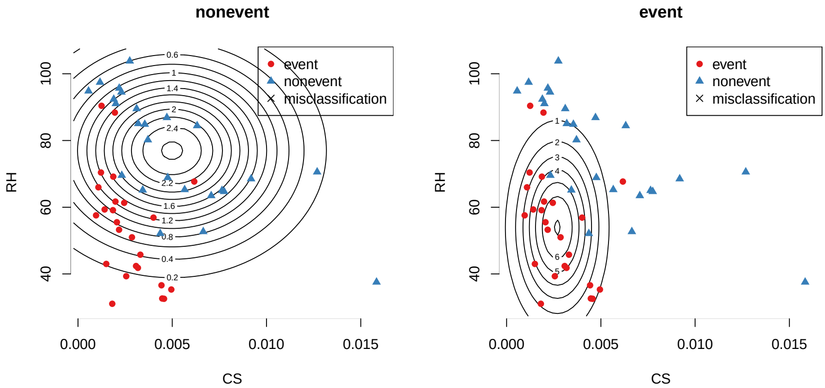
Gaussians in classification: LDA





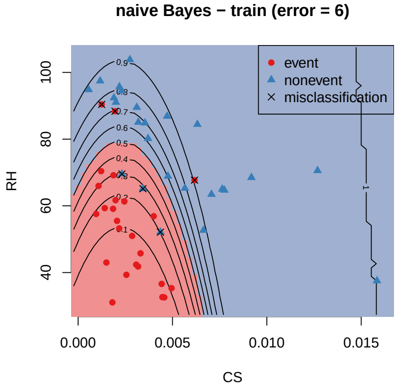
$$P(Y = 1 \mid X) = \frac{P(X \mid Y = 1)P(Y = 1)}{P(X \mid Y = 0)P(Y = 0) + P(X \mid Y = 1)P(Y = 1)}$$

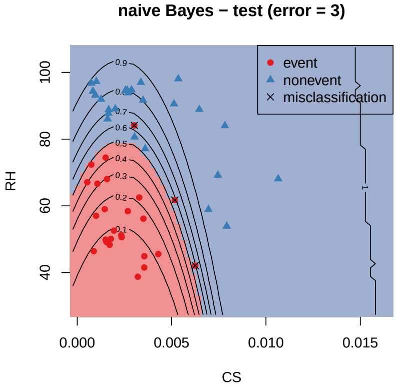
Gaussians in classification: Naive Bayes



- Left: $P(X \mid Y = 0) = P(CS \mid Y = 0)P(RH \mid Y = 0)$.
- Right: $P(X \mid Y = 1) = P(CS \mid Y = 1)P(RH \mid Y = 1)$.

Gaussians in classification: Naive Bayes





$$P(Y=1 \mid X) = \frac{P(X \mid Y=1)P(Y=1)}{P(X \mid Y=0)P(Y=0) + P(X \mid Y=1)P(Y=1)}$$

Gaussian Naive Bayes

- Assume that we have p input features, $X_1, \ldots, X_p$.
- The **naive Bayes** assumption is that input features are conditionally independent given class:

$$P(X_1, \ldots, X_p \mid Y) = P(X_1 \mid Y) \ldots P(X_p \mid Y)$$

- Thus for a feature vector x, we have

$$P(X \mid Y = 1) = P(X_1 = x_1 \mid Y = 1) \ldots P(X_p = x_p \mid Y = 1)$$
$$P(X \mid Y = 0) = P(X_1 = x_1 \mid Y = 0) \ldots P(X_p = x_p \mid Y = 0)$$

- In the Gaussian naive Bayes model, we let $P(X_j = x \mid Y = y)$ be independent univariate Gaussians for each feature X_j and class y

Number of parameters in the models

- Exercise: count the number of numbers needed to parametrize each of the models
- QDA: $kp + kp(p+1)/2 = O(kp^2)$
- LDA: $kp + p(p+1)/2 = O(kp + p^2)$
- Gaussian NB: $kp + kp = O(kp)$
- Questions:
  - How does the flexibility of different models compare?
  - Are the inductive biases (distributional assumptions) reasonable?

Naive Bayes (NB)

Conditional independence

- Classical example used to illustrate conditional independence (and also difference between correlation and causation) is correlation between ice cream sales and drowning deaths
- During sunny and warm weather people tend to both eat ice cream and go boating, swimming etc. which increases chances of drowning
- Hence, there is positive correlation between ice cream sales and number of drownings on a given day
- However, if we already know what the weather actually was, then knowing how much ice cream was sold does not help us predict drowning
- Hence, ice cream sales and drownings are *conditionally* independent given weather

About naive Bayes assumption

- The assumption that features are independent conditioned on class is
  - very strong
  - often quite untrue
- Therefore in particular the probabilities produced by a naive Bayes model should not be trusted too much
- However the classification performance (zero–one loss) of naive Bayes is often quite hard to beat in practice
- An informal justification for using naive Bayes is that often the data are collected in a way that aims to ensure (approximate) conditional independence
  - for example, in medical diagnosis, obtaining each feature requires that we carry out a test: it makes no sense to measure temperature from both armpits, or other redundant variables that we know to be strongly dependent (given the class)

. . . and what about discrete data

- For real data we can define correlations and rotations
- For discrete data these are not so obvious
- NB is the only "obvious" model to generalise to discrete data here (of QDA/LDA/NB)!

Discrete Naive Bayes

- Assume now that we have p **categorical** input features $X_1, \ldots, X_p$ where the possible values for X_j are $\{1, \ldots, q_j\}$ for some (small) number q_j of distinct values
- There are $|X| = \prod_{j=1}^{p} q_j$ possible inputs we may need to classify
- Without the naive Bayes assumption, in order to determine an arbitrary distribution over X, or an arbitrary conditional distribution $P(Y \mid X)$, we would need $|X| - 1$ parameters (since probabilities sum to one but can otherwise be chosen freely to each $x \in X$)
- In many realistic scenarios, $|X|$ is much more than the sample size, so learning such a distribution is out of the question

Naive Bayes classifier

- Let's again make the naive Bayes assumption that input features are conditionally independent given class:

$$P(X_1, \dots, X_p \mid Y) = P(X_1 \mid Y) \dots P(X_p \mid Y)$$

- Each $P(X_i \mid Y)$ is determined by $q_i - 1$ (free) parameters
- For k classes, the number of parameters is $k \sum_{j=1}^{p}(q_i - 1) \ll k(\prod_{j=1}^{p} q_i - 1)$

Learning a naive Bayes model

- Assume there are k classes $1, \ldots, k$ and p input features where for $j = 1, \ldots, p$ feature X_j has range $\{1, \ldots, q_j\}$
- We model $P(X \mid Y = y)$ separately for each class y and feature $X \in \{X_1, \ldots, X_p\}$:
  - For each $y \leq k$, $j \leq d$, and $x \leq q_j$, let $n_{y,j,x}$ be the number of examples in the training data in class y with feature value $X_j = x$, and $n_y = \sum_{x=1}^{q_j} n_{y,j,x}$
  - We estimate

$$P(X_j = x \mid Y = y) = \frac{n_{y,j,x} + m_{y,j,x}}{n_y + m_{y,j}}$$

    where $m_{y,j,x}$ is a prior pseudocount and $m_{y,j} = \sum_{x=1}^{q_j} m_{y,j,x}$
  - Usual choices for pseudocounts are $m_{y,j,x} = 0$ (maximum likelihood), $m_{y,j,x} = 1$ (Laplace smoothing), $m_{y,j,x} = 1/2$ (Krichevsky-Trofimov) etc.

From probabilistic to discrete classifier & evaluating classifiers

How to move from probabilistic to “discrete” classifier

- Assume you have a probabilistic classifier outputting $\hat{P}(Y = 1 \mid X)$.
- Choose a threshold $\theta \in [0, 1]$ and make a new classifier

$$\hat{f}(x) = \begin{cases} 1 & , \quad \hat{P}(Y = 1 \mid X = x) \geq \theta \\ 0 & , \quad \hat{P}(Y = 1 \mid X = x) < \theta \end{cases}$$

- A good choice of θ depends of a cost of false positive (classifying 0 as 1) and false negative (classifying 1 as 0).
- The “default choice”: $\theta = 1/2$.

Discrete classifiers: performance measures

	predicted class = 0	predicted class = 1	total
true class = 0	true negative (TN)	false positive (FP)	N
true class = 1	false negative (FN)	true positive (TP)	P
total	N^*	P^*	n

name	definition
false positive rate (FPR)	FP/N
true positive rate (TPR)	TP/P
positive predicted value	TP/P^*
negative predicted value	TN/N^*
accuracy	$(TN + TP)/n$

FPR = Type I error, 1-specificity; TPR = 1-Type II error, power, sensitivity, recall; TP/P^* = precision, 1-false discovery proportion

ROC curves: *FPR* vs. *TPF* as a function of threshold θ

```
makeroc <- function(score,class,main="ROC curve") {
  i <- order(score,decreasing=TRUE)
  score <- score[i]
  class <- class[i]
  tpr <- c(0,cumsum(class)/sum(class))
  fpr <- c(0,cumsum(!class)/sum(!class))
  acc <- sapply(0:length(class),function(j) mean(c(if(j<1) c() else class[1:j],
                                                   if(j<length(class))
                                                     !class[(j+1):length(class)]
                                                   else
                                                     c())))
  auc <- sum((fpr[-1]-fpr[-length(fpr)])*tpr[-1])
  plot(c(-0.1,1),0:1,type="n",xlab="false positive rate",
       ylab="true positives rate",main=main)
  abline(a=0,b=1,lty="dotted")
  lines(fpr,tpr)
  j <- floor(seq(from=1,to=length(tpr)-1,
                 length.out=min(21,length(tpr)-1)))
  points(c(fpr[j],0.6),c(tpr[j],0.2))
  text(fpr[j],tpr[j],sapply(score[j],function(x) sprintf(" %.3f",x)),
       adj=c(0,1),col="blue")
  text(fpr[j],tpr[j],sapply(acc[j],function(x) sprintf("(%.3f) ",x)),
       adj=c(1,0),col="red")
  text(0.6,0.4,sprintf("AUC = %.3f",auc),cex=1.5,pos=4)
  text(0.6,0.2,expression(paste(" threshold ",theta)),col="blue",adj=c(0,1))
  text(0.6,0.2,"(accuracy) ",col="red",adj=c(1,0))
}
```

A good tutorial: Fawcett (2006) An introduction to ROC analysis. Pattern Recognition Letters.

Training logistic regression, LDA, QDA, and NB on banknote authentication data

```r
## UCI banknote authentication dataset
## https://archive.ics.uci.edu/ml/datasets/banknote+authentication
set.seed(42)
bank <- read.csv("data_banknote_authentication.txt",header=FALSE)
colnames(bank) <- c("variance","skewness","curtosis","entropy","class")
bank$class <- factor(bank$class)
i.tr <- sample.int(nrow(bank),50)
i.te <- setdiff(1:nrow(bank),i.tr)
data_tr <- bank[i.tr,]
data_te <- bank[i.te,]
```

https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Training logistic regression, LDA, QDA, and NB on banknote authentication data

```
library(MASS)
library(e1071)

m.lr  <- glm(class ~ .,data_tr,family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

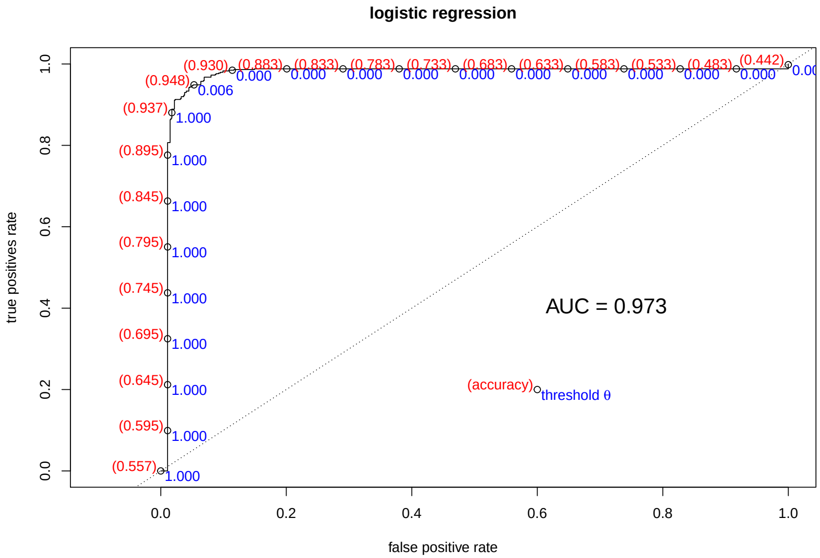
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occ
```

```
m.lda <- lda(class ~ .,data_tr)
m.qda <- qda(class ~ .,data_tr)
m.nb  <- naiveBayes(class ~ .,data_tr)

phat.lr  <- predict(m.lr,data_te,type="response")
phat.lda <- predict(m.lda,data_te)$posterior[,"1"]
phat.qda <- predict(m.lda,data_te)$posterior[,"1"]
phat.nb  <- predict(m.nb,data_te,type="raw")[,"1"]
```

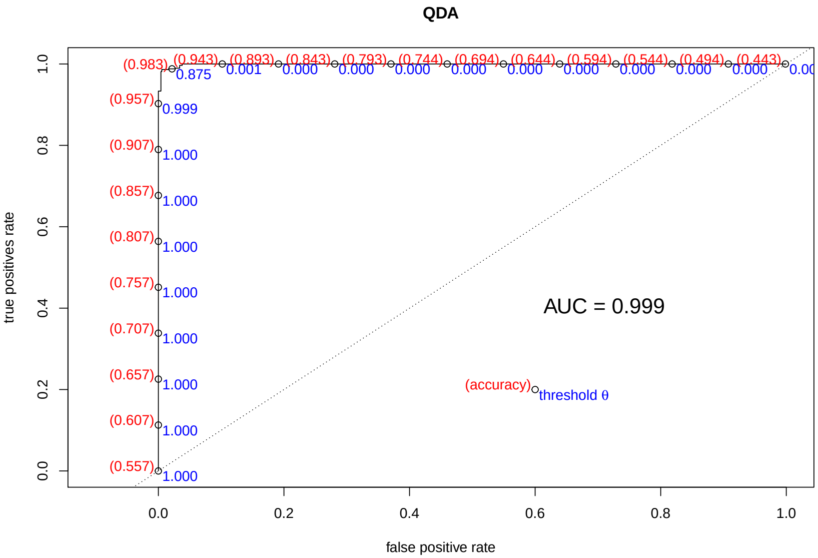
ROC curve: logistic regression

```
makeroc(score=phat.lr,class=data_te$class=="1",main="logistic regression")
```



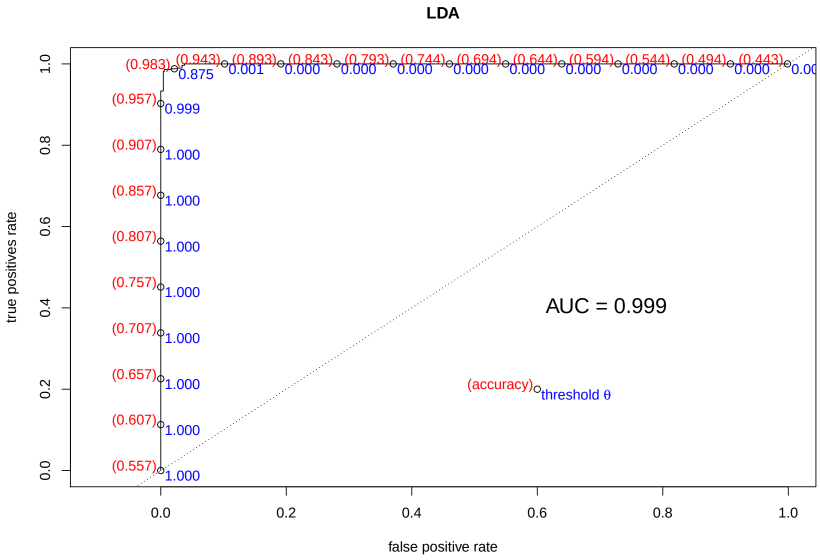
ROC curve: QDA

```
makeroc(score=phat.qda,class=data_te$class=="1",main="QDA")
```



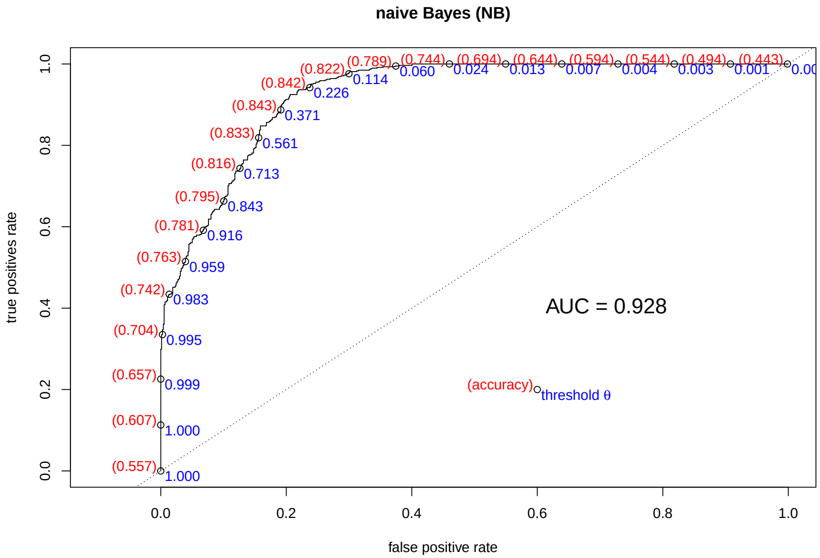
ROC curve: LDA

```
makeroc(score=phat.lda,class=data_te$class=="1",main="LDA")
```



ROC curve: naive Bayes (NB)

```
makeroc(score=phat.nb,class=data_te$class=="1",main="naive Bayes (NB)")
```



How to compare classifiers?

- counts of items classified incorrectly and correctly by Algorithms 1 & 2

counts of items...	incorr. by Alg. 2	corr. by Alg. 2
incorr. by Alg. 1	e_{00}	e_{01}
corr. by Alg. 1	e_{10}	e_{11}

- *McNemar's test*: if Algs. 1 & 2 have the same error rate we expect $e_{01} \approx e_{10} \approx (e_{01} + e_{10})/2$. Chi-square statistic with one degree of freedom:

$$\chi_1^2 \sim (|e_{01} - e_{10}| - 1)^2/(e_{01} + e_{10})$$

- We can reject the null (with the p-value of $p = 0.05$) if this value is larger than 3.84!

How to compare classifiers?

Is NB really worse than logistic regression, with threshold $\theta = 0.5$?

```
e <- table(ifelse(phat.lr>=0.5,"1","0")==data_te$class,
           ifelse(phat.nb>=0.5,"1","0")==data_te$class)
print(e)
```

```
##
##         FALSE TRUE
##   FALSE    36   34
##   TRUE    172 1080
```

```
cat(sprintf("accuracy of LR = %f, accuracy of NB = %f\n",
            mean(ifelse(phat.lr>=0.5,"1","0")==data_te$class),
            mean(ifelse(phat.nb>=0.5,"1","0")==data_te$class)))
```

```
## accuracy of LR = 0.947050, accuracy of NB = 0.842663
```

```
(abs(e[1,2]-e[2,1])-1)^2/(e[1,2]+e[2,1])
```

```
## [1] 91.11165
```

```
mcnemar.test(e)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  e
## McNemar's chi-squared = 91.112, df = 1, p-value < 2.2e-16
```

Answer: Yes. :)

Summary

Probabilistic models: summary

- Generative probabilistic models involve modeling both $P(X \mid Y = y)$ and $P(Y = y)$ for different classes y
- Important tools for this include
  - multivariate Gaussians (LDA, QDA): very important overall in statistics and machine learning, important to be familiar with them
  - Naive Bayes: especially discrete NB commonly used in practice, important to understand its uses and limitations
- Discriminative probabilistic learning aims directly at $P(Y = y \mid X)$.
  - Logistic regression is a good example

Probabilistic models in textbook

- We have more or less covered Sec. 4 (“Classification”), including **logistic regression**, **LDA**, and **QDA**
- In addition, we discussed **Naive Bayes** (NB)
- Next up: **k-NN** and **decision trees**.