Classification

Jitkomut Songsiri

Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University

CUEE

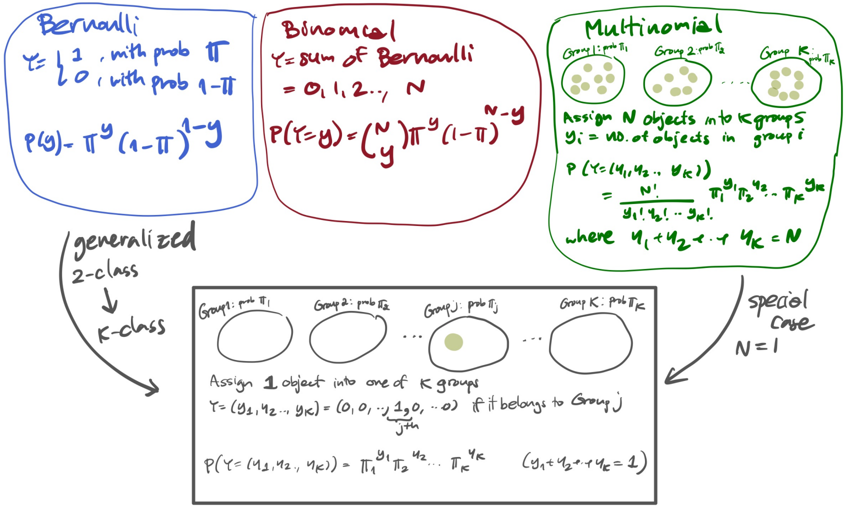
March 6, 2023

Outline

To read this note

- the response variable Y is categorical, and often mapped to $1, 2, \dots, K$ (discrete random variable, RV)
- the predictor/feature vector $X = (X_1, X_2, \dots, X_p)$ can be mixed; some X_j's can be continuous and others can be discrete
- this note does not use a rigorous notation of distribution function (either $f(x)$ or $p(x)$) to distinguish between discrete and continous RV as X can be mixed
- the typically used subscripts x, y of distribution functions as in $p_x(x), p_y(y)$ are omitted for notation simplicity
- the notation $f(x)$ may be referred to as a parametric model, or a density function; please interpret from the local context

Relevant distributions



Bernoulli

$$Y = \begin{cases} 1, & \text{with prob } \pi \\ 0, & \text{with prob } 1-\pi \end{cases}$$

$$P(y) = \pi^y (1-\pi)^{1-y}$$

Binomial

Y = sum of Bernoulli $= 0, 1, 2, \ldots, N$

$$P(Y=y) = \binom{N}{y} \pi^y (1-\pi)^{N-y}$$

Multinomial

Group 1: prob π_1 Group 2: prob π_2 . . . Group K: prob π_K

Assign N objects into K groups

y_i = no. of objects in group i

$$P(Y=(y_1, y_2, \ldots, y_K)) = \frac{N!}{y_1!\, y_2! \cdots y_K!} \pi_1^{y_1} \pi_2^{y_2} \cdots \pi_K^{y_K}$$

where $y_1 + y_2 + \cdots + y_K = N$

generalized 2-class $\downarrow$ K-class

special case $N=1$

Group 1: prob π_1 Group 2: prob π_2 . . . Group j: prob π_j . . . Group K: prob π_K

Assign 1 object into one of K groups

$Y = (y_1, y_2, \ldots, y_K) = (0, 0, \ldots, \underbrace{1}_{j\text{th}}, 0, \ldots, 0)$ if it belongs to Group j

$$P(Y=(y_1, y_2, \ldots, y_K)) = \pi_1^{y_1} \pi_2^{y_2} \cdots \pi_K^{y_K} \qquad (y_1 + y_2 + \cdots + y_K = 1)$$

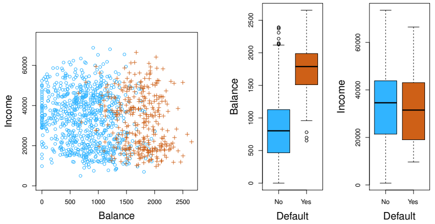
What is a classification? Why NOT a linear regression ?

What is a classification ?

classification is a process that assigns the observation to a class

- often, a method first predicts the probability of each category/class of a qualitative variable; or it provides rules to assign observations to a class
- a classification technique is called classifier

example: the default data set: incomes and monthly credit card balances



- orange shows those who defaulted on credit card payment (failed to pay the debt); and those who did not in blue
- individuals who defaulted tended to have higher balances

action: if the balance of a new individual is given, predict the probability of default example: classify patients heart disease condition: mild, moderate, severe (3 classes) using $x =$(LDL, BMI, alcohol, age) as predictors

- a classifier can provide threshold-based rules on x to predict the class, *e.g.*,
  - $\hat{Y}$=mild if LDL $<$ 40 and BMI $<$ 30
  - $\hat{Y}$=moderate if (LDL in [100,120] and alcohol $>$ 10) OR (LDL in [120,200] and age $>$ 40) OR (BMI·alcohol in [1,100])
  - $\hat{Y}$=severe if (LDL $>$ 200) OR (BMI·alcohol $>$ 200)
- a classifier approximates the probability of each class (given an observed value of X) via mathematical functions

$$\hat{f} : \mathbf{R}^n \rightarrow [0,1]^K, \quad \hat{f}_k(x;\theta) = P(Y = k|X = x)$$

  where θ is the model parameter needed to be identified

training process: involves how to find good thresholds or the best parameter θ, using training data

test process: given a new value of X, we predict $\hat{Y}$ using the estimated model

Why Not linear regression?

suppose we try to predict a medical condition of three cases: stroke, drug overdose, epileptic seizure

we can encode values for the (qualitative) response Y in many ways

choice 1: $Y = \begin{cases} 1, & \text{stroke} \\ 2, & \text{drug overdose} \\ 3, & \text{epileptic seizure} \end{cases}$ choice 2: $Y = \begin{cases} 1, & \text{epileptic seizure} \\ 2, & \text{stroke} \\ 3, & \text{drug overdose} \end{cases}$

- since the values of Y do not have a natural ordering, the two codings would produce different linear models that give different predictions
- but if Y takes a natural ordering such as mild, moderate and severe, and the gaps between (mild,moderatee) and (moderate,severe) are similar, then encoding Y as 1,2,3 would be reasonable

Encoding a binary response

if there are only two possibilities: stroke, drug overdose, only two different codings

$$\text{choice 1: } Y = \begin{cases} 0, & \text{stroke} \\ 1, & \text{drug overdose} \end{cases} \qquad \text{choice 2: } \tilde{Y} = \begin{cases} 0, & \text{drug overdose} \\ 1, & \text{stroke} \end{cases}$$

- we can fit a linear regression using the first choice of encoding and predict drug overdose if $\hat{Y} > 0.5$ (or for choice 2, predict stroke if $\hat{Y} > 0.5$)
- ✎ it can be shown that the two flip codings produce the same predictions

$$\tilde{Y} = \mathbf{1} - Y, \quad Y = X\beta, \ \ \tilde{Y} = X\gamma, \ \ \text{unseen predictor is } \ z^T = \begin{bmatrix} 1 & \tilde{z}^T \end{bmatrix}$$

$$\text{show that } \ z^T \beta_{\text{ls}} > 0.5 \Longleftrightarrow z^T \gamma_{\text{ls}} < 0.5$$

- however, some of $\hat{Y}$ could lie outside $[0, 1]$ making hard to interpret as probabilities

Logistic regression

Binary classification

consider the problem of classifying data into two classes: $Y \in \{0, 1\}$

setting:

- we have data (Y, X) where Y is the response variable and X is the predictor
- example: defaults on credit card payment
  - $X = (X_1, X_2, X_3)$ contains `balance`, `income`, `student status`
  - Y is `default` status; $Y = 1$ is 'yes' and $Y = 0$ is 'no'

goal: find a model that provides $P(Y = 1 \mid X = x)$

$$P(\texttt{default = yes} \mid \texttt{balance} = 10,000\text{baht}, \texttt{income} = 200\text{kbaht}, \texttt{student = no})$$

Logistic model

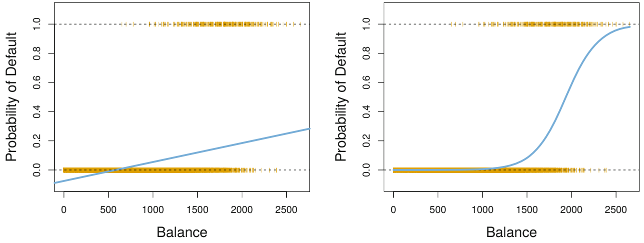
a logistic function is used to gives output between 0 and 1

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} \quad \text{has S-shape}$$

(this is a nominal form of logistic, aka. sigmoid function)

a logistic model uses the logistic function to explain Y from predictors through:

$$P(Y=1|X) = \frac{e^{\beta^T X}}{1+e^{\beta^T X}}, \quad P(Y=0|X) = \frac{1}{1+e^{\beta^T X}}$$



Logistic regression

problem: fitting the logistic model

$$P(Y=1|X) = \frac{e^{\beta^T X}}{1+e^{\beta^T X}}$$

from a training data set $\{(y_i, x_i)\}_{i=1}^N$ to estimate parameters β

- the linear predictor term is $\beta^T X = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$
- if an intercept β_0 is needed, we assume X_k must contain $\mathbf{1}$
- estimation method: maximum likelihood estimation (more on this later)
- for new $X = x$, if $P(Y=1|X) > 0.5$ we classify that this data belong to class '1', and '0' otherwise
- the threshold of 0.5 is up to the user

- the following quantitiy, called odds,

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} = e^{\beta^T X} \qquad \in (0,\infty)$$

  indicates the ratio of the chance that class '1' occurs to class '0'
- the log of odds, called logit

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \beta^T X$$

  provides a *link function* between the probability and the linear regression expression
- if X_k is one-unit changed
  - in linear regression, the average in Y is changed by β_k
  - in logistic regression, the log odds change by β_k

Estimating regression coefficients

denote the logistic function: $p(x) = e^{\beta^T x}/(1 + e^{\beta^T x})$

β_0, β are chosen to maximize the **likelihood function**

$$\begin{aligned}\mathcal{L}(\beta) &= \prod_{i:y_i=1} p(x_i) \prod_{k:y_k=0} (1 - p(x_k)) \\ &= \prod_{i:y_i=1} \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \prod_{k:y_k=0} \frac{1}{1 + e^{\beta^T x_k}}\end{aligned}$$

since $\log(\cdot)$ is increasing, it is the same as maximizing the **log-likelihood**

$$\log \mathcal{L}(\beta) = \sum_{i:y_i=1} \beta^T x_i - \sum_{k=1}^{N} \log(1 + e^{\beta^T x_k})$$

this is a nonlinear unconstrained optimization problem (can be solved by Newton/Quasi-Newton)

Derivation of loglikelihood

suppose $\{(y_i, x_i)\}_{i=1}^n$ are available where $y_i = 0, 1$

- we can write $P(Y = y \mid X = x; \beta) = p(x)^y (1 - p(x))^{1-y}$
- if we have n independent observations, the likelihood function is expressed as

$$\mathcal{L}(y_1, \ldots, y_n \mid x; \beta) = \prod_i P(Y = y_i \mid x_i; \beta) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

$$\begin{aligned} \log \mathcal{L}(y_1, \ldots, y_n \mid x; \beta) &= \sum_{i=1}^n y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i)) \\ &= \sum_{i=1}^n y_i \log \left(\frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \right) + (1 - y_i) \log \left(\frac{1}{1 + e^{\beta^T x_i}} \right) \end{aligned}$$

- substitute $y_i = 1$ for some i and $y_i = 0$ otherwise; this gives $\log \mathcal{L}$ on page 15

Default on credit card payment

example of running logistic regression for the default data on page 11

	Coefficient	Std. error	Z-statistic	P-value
Intercept	−10.8690	0.4923	−22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	−0.6468	0.2362	−2.74	0.0062

prediction: use $\hat{\beta}$ from the table we can make an estimate of Y

- student/non-student with balance of $1,500$ dollars and income of $40,000$

$$\begin{array}{lll} \text{student} & P(Y = 1 \mid X = (1500, 40000, 1)) & = 0.068 \\ \text{non-student} & P(Y = 1 \mid X = (1500, 40000, 0)) & = 0.105 \end{array}$$

- with the same balance and income, a non-student is more likely to default

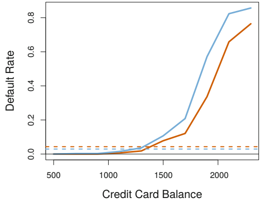
Correlated predictors

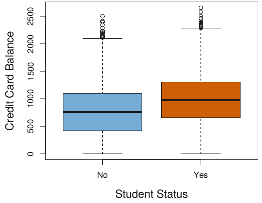
compare the results between one predictor (student status) and three predictors

	Coefficient	Std. error	Z-statistic	P-value
Intercept	−10.8690	0.4923	−22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	−0.6468	0.2362	−2.74	0.0062

	Coefficient	Std. error	Z-statistic	P-value
Intercept	−3.5041	0.0707	−49.55	<0.0001
student[Yes]	0.4049	0.1150	3.52	0.0004

- the coefficient of student status is negative (left) and positive (right)
- negative coefficient of student status indicates that students are less likely to default (than non-students) – here we can have contradictory results ?





students / non-students

regress on balance only

observations:

- in multiple regression (left table), negative coefficient for student indicates that *for a fixed value of balance and income*, a student is less likely to default than a non-student (confirmed by that the orange line is lower than the blue line)
- the horizontal lines show the default rates that are averaged over all values of balance and income – but here the orange line is higher than the blue line
- the box plots suggest that students tend to have higher credit card balance – associated with high default rates

explanations:

- 'student status' and 'balance' are correlated (students tend to have higher debt)
- an *individual* student with a given balance tends to have a lower chance of default, while students *on the whole* tend to have higher credit card balance which further tend to have a higher default rate

conclusions:

- a student is riskier than a non-student if no information about credit card balance is available
- a student is less risky than a non-student with the *same* credit card balance
- a confounding problem: a result obtained from one predictor is different from using multiple predictors when there is correlation among the predictors

K-label classification

the logistic regression can be extended to classify data into K categories

- define the response as indicator variable: $Y = (Y_1, Y_2, \ldots, Y_K)$ where

$$Y_k = 1 \quad \text{if the response fall into } k\text{th category and} \quad Y_j = 0, \quad \forall j \neq k$$

  e.g. three medical conditions:

$$Y = \begin{cases} (1,0,0), & \text{if } \texttt{stroke;} \\ (0,1,0), & \text{if } \texttt{drug overdose ;} \\ (0,0,1), & \text{if } \texttt{epileptic seizure.} \end{cases}$$

- the choice of **generalized Bernoulli** distribution is suitable for the conditional distribution; π_k is the probability of $Y_k = 1$

$$P(Y = (y_1, \ldots, y_K)|X) = \pi_1^{y_1} \pi_2^{y_2} \cdots \pi_K^{y_K}$$

Multinomial logistic model

denote G the variable indicating the group: Y is in kth group iff $G = k$

$$Y = (0, 0, \ldots, \underbrace{1}_{k\text{th}}, 0, \ldots, 0) \quad \Longleftrightarrow \quad G = k$$

- model: log-odd of each response is linear function of predictors

$$\begin{aligned}
\log \frac{P(G=1 \mid X)}{P(G=K \mid X)} &= \beta_1^T X = \beta_{10} + \beta_{11} X_1 + \cdots + \beta_{1p} X_p \\
\log \frac{P(G=2 \mid X)}{P(G=K \mid X)} &= \beta_2^T X = \beta_{20} + \beta_{21} X_1 + \cdots + \beta_{2p} X_p \\
&\vdots \\
\log \frac{P(G=K-1 \mid X)}{P(G=K \mid X)} &= \beta_{K-1}^T X = \beta_{K-1,0} + \beta_{K-1,1} X_1 + \cdots + \beta_{K-1,p} X_p
\end{aligned}$$

- the last class $(G = K)$ is chosen to be called the referenced or nominal model
- β_{k0} is the log odds of class k versus nominal given that all $X_1, \ldots, X_p$ are zero
- if X_j increases by one unit, then $P(G = k|X)/P(G = K|X)$ increases by $e^{\beta_{kj}}$

Log-likelihood function

- the conditional probabilities can be expressed as

$$P(G = k \mid X) = \frac{e^{\beta_k^T X}}{1 + \sum_{l=1}^{K-1} e^{\beta_l^T X}}, \quad k = 1, 2, \ldots, K-1,$$

$$P(G = K \mid X) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\beta_l^T X}} \quad \text{(chosen to be the referenced class)}$$

  (the sum of K probabilities is one)
- denote $p_k(x;\beta) = P(G = k \, | x)$ (the conditional pdf of $Y|X$)
- the **log-likelihood** function of $y|x$ is given by replacing π_k with the model

$$\log p(y \mid x; \beta) = \log(\pi_1^{y_1} \pi_2^{y_2} \cdots \pi_K^{y_K}) = \sum_{l=1}^{K} y_l \log p_l(x;\beta)$$

  (1-sample log-likelihood); entries of $y = (y_1, \ldots, y_K)$ are either 0 or 1
- if $y|x$ belongs to class k, it reduces to $\log p(y|x;\beta) = \log p_k(x;\beta)$

Log-likelihood function

model parameters are $\beta \triangleq (\beta_1, \beta_2, \ldots, \beta_{K-1})$ and using

$\log p_l(x;\beta) = \beta_l^T x - \log[1 + \sum_{l=1}^{K-1} e^{\beta_l^T x}]$ for $l = 1, 2, \ldots, K-1$

$$\log \mathcal{L}(\beta) \triangleq \log p(y^{(1)}, \ldots, y^{(N)} | x^{(1)}, \ldots, x^{(N)}; \beta) = \sum_{i=1}^{N} \sum_{l=1}^{K} \log p_l(x^{(i)}; \beta) y_l^{(i)}$$

$$= \sum_{i \in \text{class } 1} \log p_1(x^{(i)}; \beta) + \cdots + \sum_{i \in \text{class } K-1} \log p_{K-1}(x^{(i)}; \beta)$$

$$+ \sum_{i \in \text{class } K} \log p_K(x^{(i)}; \beta)$$

$$= \sum_{i \in \text{class } 1} \beta_1^T x^{(i)} + \cdots + \sum_{i \in \text{class } K-1} \beta_{K-1}^T x^{(i)} - \sum_{i=1}^{N} \log \left[1 + \sum_{l=1}^{K-1} e^{\beta_l^T x^{(i)}} \right]$$

Estimation of multinomial logistic coefficients

suppose data $\{(y^{(i)}, x^{(i)})\}_{i=1}^{n}$ are available (independent samples)

- we aim to **maximize** $\log \mathcal{L}(\beta)$ (hence, minimize the negative log-likelihood)
- β can be solved numerically from optimization methods
- the Newton algorithm can be expressed as iterative reweighted least-square algorithms (see ESL in chapter 4.4)
- softwares: `multinom` in R, `mnrfit` in MATLAB, `scikitlearn:linear_model` in Python
- if a nominal class is changed
  - the estimated coefficients (β_l) would change (and its interpretation is up to the choice of the nominal class)
  - however, the log odds between any pair of classes, and the fitted values (predictions) will remain the same

Softmax coding

instead of estimating coefficients for $K-1$ classes, we estimate for *all* K classes

$$P(G=k|X) = \frac{e^{\beta_k^T x}}{\sum_{l=1}^K e^{\beta_l^T X}} = \frac{e^{\beta_k^T x}}{e^{\beta_1^T X} + e^{\beta_2^T X} + \cdots + e^{\beta_K^T X}}, \quad k = 1, 2, \ldots, K$$

- also known as the **softmax function** used in neural network
- in neural network, the softmax is defined with variable z which is the transformed variable before the output layer
- the log odds ratio between the kth and lth classes is

$$\log \frac{P(G=k \mid X)}{P(G=l \mid X)} = (\beta_k - \beta_l)^T X$$

Bayesian decision theory

Expected loss

setting: Y is categorical variable taking values $\in \mathcal{G} = \{1, 2, \ldots, K\}$

- loss function matrix: $\mathbf{L} \in \mathbf{R}^{K \times K}$ taking zero values on the diagonal where ℓ_{kl} is the penalty for classifying group k as l
- zero-one loss function: $\mathbf{L}$ has all-one entries (except the diagonal) meaning missclasfications are charged with equal weights
- $L(Y = k, \hat{Y} = l) = \ell_{kl}$ is a loss function for classifying Y as $\hat{Y}(X)$

define the expected loss as the expected value of loss function over all X, Y

$$\text{expected loss} = \mathbf{E}_{yx}[L(Y, \hat{Y}(X))] = \mathbf{E}_x \left[\sum_{k=1}^{K} L(Y, \hat{Y}(X)) P(Y = k | X) \right]$$

using conditional expectation

the expected loss is also referred to as expected prediction error or risk function

Bayes classifier

for a given $X = x$, it suffices to minimize the classification error *pointwise*:

$$\hat{Y}(x) = \underset{l \in \mathcal{G}}{\operatorname{argmin}} \sum_{k=1}^{K} L(Y = k, \hat{Y}(x) = l) P(Y = k | X = x)$$

(minimize the sum of weighted penalty; weight = chance of Y when X is observed)

with the zero-one loss function, this reduces to

$$\hat{Y}(x) = \underset{l \in \mathcal{G}}{\operatorname{argmin}} [1 - P(Y = l | X = x)] = \underset{l \in \mathcal{G}}{\operatorname{argmax}}\, P(Y = l | X = x)$$

Bayes classifier: classify to the most probable class using the conditional $P(Y|X)$

Bayes error rate

assumption: using zero-one loss function

the corresponding error rate of the Bayes classifier is called the Bayes error rate:

$$1 - \mathbf{E}_x \left(\max_{l \in \mathcal{G}} P(Y = l | X = x) \right)$$

using $l^\star$ that maximizes the posterior probability, this follows from

$$\text{optimal error} = \mathbf{E}_x \left[\sum_{k \neq l^\star} P(Y = k | X) \right] = \mathbf{E}_x [1 - P(Y = l^\star | X = x)]$$

- the Bayes error rate is irreducible, equivalent to noise variance in regression

Terminology in Bayesian theory

- prior probability, $p(y)$, is the knowledge we have *before* looking at an observed x
- the class likelihood or class-conditional density, $p(x|y)$, gives distribution information of features in each class (once we know the response variable belongs to the class y)
- evidence, $p(x)$ is the marginal probability that a value x is observed (regardless of the class of Y) – can be computed using total probability
- posterior probability, $p(y|x)$ is the likelihood of response *after* x is seen

Bayes' rule: $$\text{posterior} = \frac{\text{class likelihood} \times \text{prior}}{\text{evidence}}, \quad p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

$$P(Y = l|x) = \frac{p(x|Y = l)P(Y = l)}{p(x)}$$

Bayes decision boundary

setting: two-class data $(Y = 1, 2)$ where given $Y = k$, $X|Y \sim \mathcal{N}(\mu_k, \Sigma_k)$

- given: (μ_1, Σ_1) and (μ_2, Σ_2) are estimated using maximum likelihood
- goal: gives the decision rule based on posterior probability to classify Y when X is observed

$$p(y|x) = \frac{f(x|y)p(y)}{f(x)}, \quad \log f(x|k) = -\frac{1}{2}\log\det\Sigma_k - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)$$

$$P(Y=1|X) > P(Y=2|X) \Leftrightarrow \log f(x|1) + \log P(Y=1) > \log f(x|2) + \log P(Y=2)$$

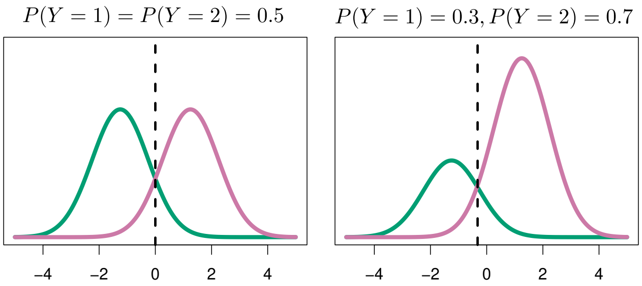
decision rule: classifying to $Y = 1$ if the given x satisfies $g(x) > 0$ where

$$g(x) = \frac{1}{2}\left[(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] + \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \log\frac{P(Y=1)}{P(Y=2)}$$

and classify to $Y = 2$ otherwise

Bayes decision boundary: scalar case

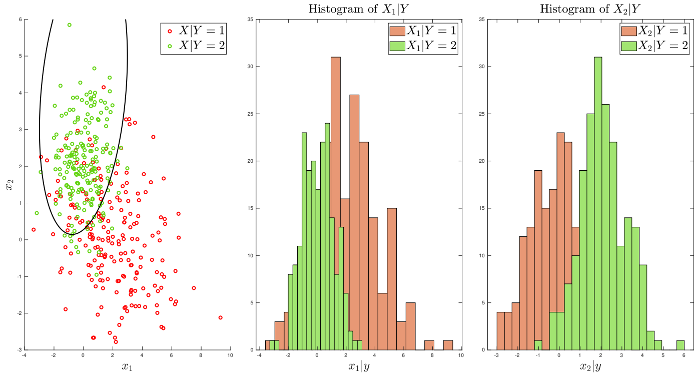
an example of posterior densities for two cases of prior probabilities



- for a given test observation x, assign to the class for which the density is highest; either class 1 or class 2
- when the prior of class 2 is higher, the decision boundary is shifted to the left (more favor to class 2)

example: conditional 2D Gaussians

$$(X|Y=1) \sim \mathcal{N}\left(\begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}4 & -1\\-1 & 2\end{bmatrix}\right) \text{ and } (X|Y=2) \sim \mathcal{N}\left(\begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$



- Bayes decision boundary of Gaussian case leads to a quadratic function
- what happen to the decision boundary as $P(Y=1)$ increases ?

k-nearest neighbor (kNN)

idea sketch:

- the optimal Bayes classifer requires the knowledge of $p(y|x)$ which is typically unknown for real data
- kNN classifies Y to the class with highest estimated conditional probability

algorithm description:

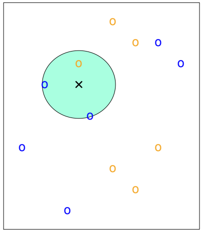
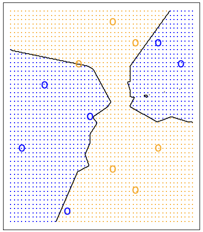
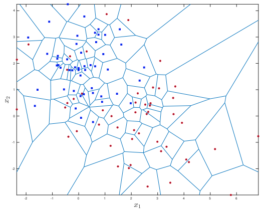
- given a positive integer k and a test observation x_0, find k points in the training data that are closet to x_0 denoted by set $\mathcal{X}$
- estimate the conditional probability:

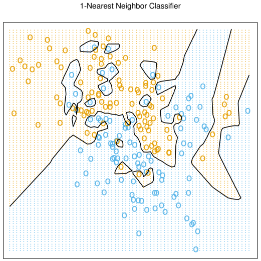
$$P(Y = j|x_0) = \frac{\text{number of points in } \mathcal{X} \text{ that belongs to class } j}{k}$$

- kNN applies Bayes rule, classifying the test point x_0 to the class with the largest conditional probability

Illustration of kNN

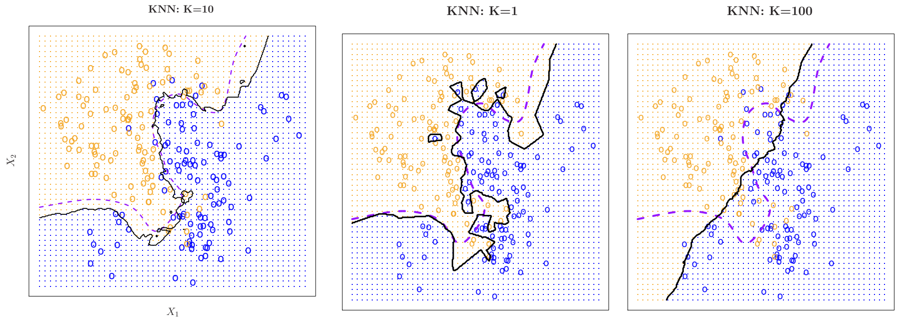
kNN does not assume the *true* distribution structure of the data (non-parametric)



- *1st:* using $k = 3$, locate the three closest points to x_0 (black cross)
- the blue points win the majority vote (2/3), so x_0 is assigned to the blue class
- *2nd:* apply kNN with $k = 3$ at all points, giving kNN decision boundary
- *3rd-4th:* 1NN decision boundary is a Voronoi tessellation; each x_i has a tile bounding region for which it is the closest input point; 1NN has zero training error rate; none of training data are misclassified

effect of k in kNN



- *left:* using $k = 10$, kNN and Bayes decision boundaries are similar
- *middle:* using $k = 1$, kNN boundary finds pattern in the data, and is very flexible with low bias – it is quite different from Bayes boundary
- *right:* using $k = 100$, kNN is less flexible and the boundary is close to linear
- k is generally taken to be an odd number to minimize ties

Further notes of kNN

- the optimal value of k depends on the **bias-variance** tradeoff; small k provides the most flexible fit which has low bias but high variance
- we can choose k from a plot of cross-validated MSE versus $1/k$
- when using kNN in regression problems, the output prediction when $x = x_0$ is the average of all training responses in the neighborhood of x_0

$$\hat{f}(x_0) = \frac{1}{k} \sum_{x_i \in \mathcal{N}(x_0)} y_i$$

- the scale of variable matters because kNN detects the distance between observations (the variable with a larger scale affects more)
- it is advised to *standardize* data to have zero mean and unit variance

Linear discriminant analysis (LDA)

Y takes values in $\mathcal{G} = \{1, 2, \ldots, K\}$

the posterior probability to classify Y into class l is

$$P(Y = l | X = x) = \frac{f(x|Y = l)P(Y = l)}{f(x)}, \quad f(x) = \sum_{k=1}^{K} f(x|Y = k)P(Y = k)$$

to decide the class with highest posterior, it is required to estimate:

- the prior: $\pi_k = P(Y = k)$ – via computing the fraction of each class from the training data (easy)
- the class likelihood: $f(x|Y = k)$ – requires more assumptions about distribution structures

LDA makes a Gaussian assumption with equal covariances, leading to linear decision boundary

Gaussian case with equal covariance

assumptions:

- $f(x|Y=k)$ is Gaussian with mean μ_k and equal covariance Σ for $k=1,2,\ldots,K$
- μ_k and Σ are estimated beforehand; in practice, we can use

$$\hat{\mu}_k = \text{sample mean of } X \text{ in class } k, \quad \hat{\Sigma} = \frac{1}{N-K}\sum_{k=1}^{K}\sum_{i:Y_i=k}(x^{(i)}-\hat{\mu}_k)(x^{(i)}-\hat{\mu}_k)^T$$

the posterior probability $p_l(x) \triangleq P(Y=l|X=x)$ is

$$p_l(x) = \frac{e^{-\frac{1}{2}(x-\mu_l)^T\Sigma^{-1}(x-\mu_l)}\pi_l}{\sum_{k=1}^{K} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)}\pi_k}$$

to decide the highest $p_l(x)$ for $l=1,2,\ldots,K$, the denominators are the same, regardless of the density assumption take the log of $p_l(x)$ and neglect the term $x^T\Sigma^{-1}x$, we predict the class l for which

$$\text{discriminant function:}\quad g_l(x) = x^T\Sigma^{-1}\mu_l - \frac{1}{2}\mu_l^T\Sigma^{-1}\mu_l + \log\pi_l$$

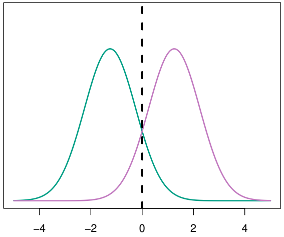
is highest among $l = 1, 2, \ldots, K$ (note that we use estimates of μ_l, Σ)

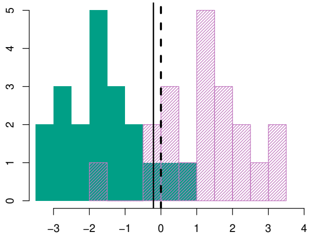
the decision boundary of LDA between class l and k is the set of x that

$$g_l(x) = g_k(x) \iff x^T\hat{\Sigma}^{-1}(\hat{\mu}_l - \hat{\mu}_k) = (1/2)(\hat{\mu}_l^T\hat{\Sigma}^{-1}\hat{\mu}_l - \hat{\mu}_k^T\hat{\Sigma}^{-1}\hat{\mu}_k) + \log(\hat{\pi}_k/\hat{\pi}_l)$$

- the decision boundary is linear in x or a hyperplane in $\mathbf{R}^n$ (so called linear in LDA)
- ✎ for scalar x and when prior densities are equal, the boundary is the midpoint of two sample means: $x = \frac{1}{2}(\hat{\mu}_k + \hat{\mu}_l)$

Decision boundaries for scalar x

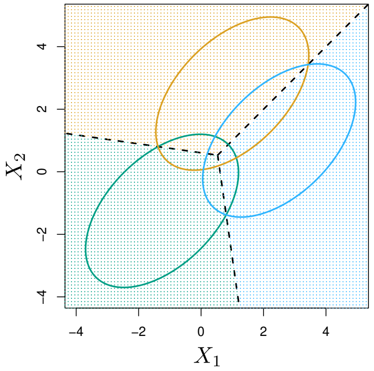


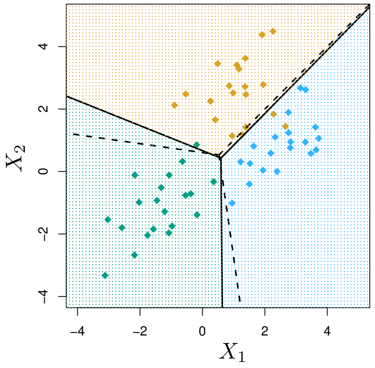


- setting: true densities are Gaussian with $\mu_1 = -1.25, \mu_2 = 1.25$ and unit variance; drawn 20 observations from each class
- Bayes decision (dashed line) is then $x = (\mu_1 + \mu_2)/2 = 0$ (unknown in real-life)
- the area for which the two densities overlapped is the classification error
- LDA decision boundary (solid line) is a little off from Bayes since it uses the sample mean computed from the training data

LDA decision boundaries

setting: 3-class, true distribution is Gaussian with equal Σ, 20 samples for each class





- *left:* Bayes decision boundary and 95 %-confidence ellipsoid (plotted by using the true parameters: μ_k, Σ)
- *right:* solid lines are LDA decision boundaries (using $\hat{\mu}_k, \hat{\Sigma}$); dash lines are Bayes

Quadratic discriminant analysis (QDA)

like LDA, QDA assumes observations in each class are Gaussian but each has its own covariance matrices – the posterior probability $p_l(x) \triangleq P(Y=l|X=x)$ is

$$p_l(x) = \left[\frac{e^{-\frac{1}{2}(x-\mu_l)^T \Sigma_l^{-1}(x-\mu_l)} P(Y=l)}{(2\pi)^{n/2} (\det \Sigma_l)^{1/2}} \right] / f(x)$$

QDA follows Bayes classifer to perform the prediction, choosing the class l for which

discriminant function: $g_l(x) = -\frac{1}{2}(x-\mu_l)^T \Sigma_l^{-1}(x-\mu_l) - \frac{1}{2}\log\det\Sigma_l + \log P(Y=l)$

is highest among $l = 1, 2, \ldots, K$; however, it uses estimates of μ_l, Σ_l

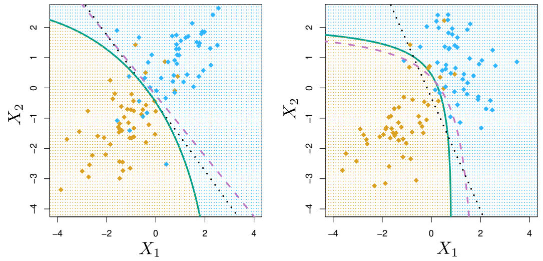
- the decision boundary between class l and k is given by the set of x that $g_l(x) = g_k(x)$ – becoming quadratic function in x
- QDA has more number of parameters to estimate – leading to bias-variance trade-off, if compared to LDA

Comparison between LDA and QDA

assume n predictors and K class

method	number of parameters	model property
LDA	nK	low flexibility, low variance
QDA	$n(n+1)K/2$	high flexibility, high variance

comparison: Gaussian classes have common (left) and different (right) covariances

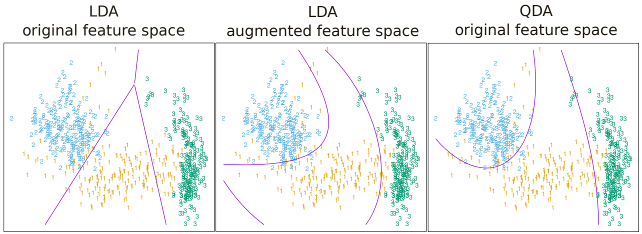


- *left:* Bayes decision boundary is linear and accurately approximated by LDA
- *right:* now Bayes decision boundary is quadratic and QDA is more accurate

Quadratic boundaries

feature spaces: original and its mapping by a quadratic function

original: $\{X_1, X_2, \ldots, X_n\}$, augmented: $\{X_1, X_2, \ldots, X_n, X_1^2, X_2^2, \ldots, X_n^2\}$



- *middle:* linear functions of LDA performed on the augmented space results in quadratic functions in the original space
- while preferring QDA, the differences between QDA and LDA (middle) are small

Further notes on LDA and QDA

we choose a class of interest, assumed to be class K

- we test against the lth classes for $l \neq K$ (hence, there are $K-1$ tests)
- compute the difference between discriminant functions:

$$\text{choose class } K \text{ if } \; g_K(x) - g_l(x) > 0, \quad \text{and choose class } l \text{ otherwise}$$

- reasons for using LDA and QDA are i) simple decision boundaries such as linear or quadratic are sufficient for the data and ii) when the estimates provided via Gaussian models are stable
- extension to regularized discriminant analysis (RDA) where the regularized covariance is a combination of individual and the common covariance: $\hat{\Sigma}_{\mathrm{reg},k} = \alpha \hat{\Sigma}_k + (1-\alpha)\hat{\Sigma}$ – see ELSR section 4.3.1

Kernel density estimation (KDE)

we are often required to have the probability density $f(x)$ at some point x_0

- suppose a random sample $x_1, x_2, \ldots, x_N$ are drawn from $f(x)$
- let $\mathcal{B}(x_0)$ be a small metric neighborhood around x_0 of width h – or a bin
- a natural local estimate of $f(x_0)$ has the form

$$\hat{f}(x_0) = \frac{\text{no. of } \; x_i \in \mathcal{B}(x_0)}{N \cdot h} \qquad \text{(often returns a bumpy estimate)}$$

- we often prefer a smooth estimate using Parzen window of the form

$$\hat{f}(x_0) = \frac{1}{Nh} \sum_{i=1}^{N} K_h(x_0, x_i), \quad (K_h \text{ is a kernel function})$$

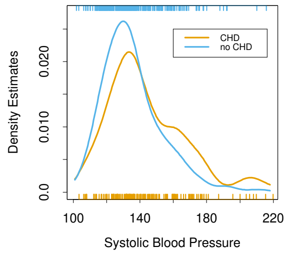
- a kernel function has weights that decreases with distance from x_0, *e.g.*, the Gaussian kernel $K_h(x_0, x) = \phi(|x - x_0|/h)$

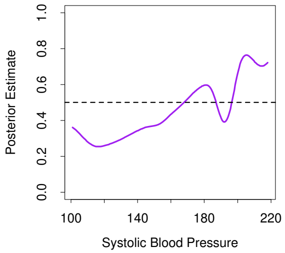
Non-parametric classification

from Bayes theorem, when making a decision based on the posterior probability

$$P(Y=j|X=x) = \frac{f(x|Y=k)P(Y=l)}{f(x)}, \quad j = 1,2,\ldots,K$$

a non-parametric approach uses a non-parametric density estimate of class-conditional and prior densities (replace $f(x|y), P(Y=l)$ above with the estimates)





- classification of coronary heart disease (CHD) where X is systolic blood pressure
- a Gaussian kernel density estimate for each $\hat{f}(x|Y)$

Naïve Bayes

again, consider the posterior probability

$$P(Y=j|X=x) = \frac{f(x|Y=k)\pi(j)}{f(x)}, \quad j=1,2,\ldots,K, \;\; \pi(j) = P(Y=j),$$

when $X = (X_1, \ldots, X_p)$ and p is high (the feature space is high-dimensional)
Naïve Bayes assumes that the inputs X_k's are conditionally independent:

$$f_j(x) \triangleq f(x|Y=j) = \prod_{k=1}^{p} f_{jk}(x_k) \quad \text{(class-conditional is the products of marginals)}$$

- each class-conditional marginal f_{jk} can each be estimated *separately* using
  - one-dimensional Gaussian densities (called Gaussian Naïve Bayes)
  - one-dimensional kernel density estimates
  - multinomial distribution

  depending on the assumption of predictor distribution

- if some x_k is discrete, one can use a histogram estimate; useful when x contain both discrete and continuous variables
- the conditionally independent assumption is usually violated in practice, and Naïve Bayes may yield biased class-conditional density estimates
- even so, the resulting posterior tends to be robust to the biased class-density estimates near the decision boundary
- the logit-transform has a connection with generalized additive model (GAM)

$$\log \frac{P(Y=l|X)}{P(Y=K|X)} = \log \frac{\pi_l f_l(X)}{\pi_J f_K(X)} = \log \frac{\pi_l \prod_{i=1}^p f_{li}(X_i)}{\pi_K \prod_{i=1}^p f_{Ki}(X_i)}$$
$$= \log \frac{\pi_l}{\pi_K} + \sum_{i=1}^p \log \frac{f_{li}(X_i)}{f_{Ki}(X_i)} \triangleq \alpha_l + \sum_{i=1}^p g_{li}(X_i)$$

the latter term $\sum_i g_{li}(X_i)$ is a form of generalized *additive* models

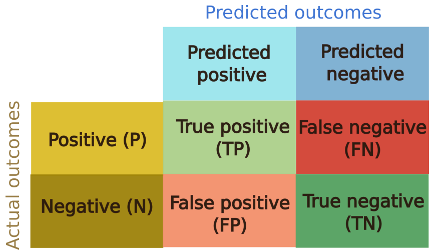
Softwares

Methods	**MATLAB**	**Python (scikit-learn)**
logistic regression	`fitglm`	linear_model.LogisticRegression
multinomial logistic regression	`mnrfit`	linear_model.LogisticRegression
naïve Bayes	`fitcnb`	naive_bayes
kNN	`fitcknn`	neighbors.KNeighborsClassifier
LDA, QDA	`fitcdiscr`	discriminant_analysis

Classification performance evaluation

Confusion matrix

suppose a response variable (y) has two outcomes; either positive or negative



Actual outcomes	Predicted outcomes: Predicted positive	Predicted outcomes: Predicted negative
Positive (P)	True positive (TP)	False negative (FN)
Negative (N)	False positive (FP)	True negative (TN)

- true positive (TP): a correctly identified positive
- true negative (TN): a correctly identified negative
- false positive (FP): an *incorrectly* identified positive – type I error
- false negative (FN): an *incorrectly* identified negative – type II error

- sensitivity or TPR: probability of predicting positive given the truth is positive
- specificity or TNR: probability of predicting negative given the truth is negative
- FPR: probability of predicting positive given the truth is negative

Standard classification indices

$$
\begin{aligned}
\text{sensitivity or recall:}\quad \text{TPR} &= \frac{\text{correctly predicted positive}}{\text{total positive}} = \frac{\text{TP}}{\text{TP}+\text{FN}} \\
\text{FPR} &= \frac{\text{incorrectly predicted positive}}{\text{total negative}} = \frac{\text{FP}}{\text{FP}+\text{TN}} \\
\text{specificity:}\quad \text{TNR} &= \frac{\text{correctly predicted negative}}{\text{total negative}} = \frac{\text{TN}}{\text{TN}+\text{FP}} = 1-\text{FPR} \\
\text{FNR} &= \frac{\text{incorrectly predicted negative}}{\text{total positive}} = \frac{\text{FN}}{\text{TP}+\text{FN}} = 1-\text{TPR} \\
\text{accuracy:}\quad \text{ACC} &= \frac{\text{all correctly predictions}}{\text{total population}} = \frac{\text{TP}+\text{TN}}{\text{P}+\text{N}}
\end{aligned}
$$

- by adjusting a classifier's hyperperameters, if TPR increases, so does FPR
- in medical applications when positive is rare, it's more challenging to obtain a low FPR (aka false alarm) or high specificity

More classification performance indices

when observations contain an imbalance between two classes

$$\begin{aligned}
\text{prevalence} &= \frac{\text{total positive}}{\text{total population}} = \frac{\mathrm{P}}{\mathrm{P}+\mathrm{N}} \text{ (proportion of positive)} \\
\text{false discovery rate:}\quad \mathrm{FDR} &= \frac{\text{incorrectly predicted positive}}{\text{predicted positive}} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TP}} \\
\text{positive predictive value:}\quad \mathrm{PPV} &= \frac{\text{correctly predicted positive}}{\text{predicted positive}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
&= 1-\mathrm{FDR} \;\text{ (or precision)} \\
\text{F1 score} &= \text{harmonic mean of precision and sensitivity} \\
&= 2\times\frac{\mathrm{PPV}\cdot\mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}
\end{aligned}$$

F1 score finds the (harmonic) mean of two rates and the precision rate takes into account the portion of positive in data

Trade-off in classification

sensitivity-specificity trade-off in medical applications

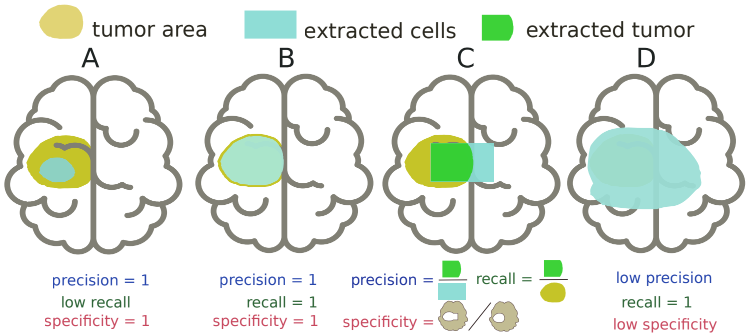
- a classifier is associated with a threshold or some hyperparameters which control the fraction of TP and FP
- if a classifier is aimed to detect a disease condition more correctly, it has a high sensitivity (sensitive to detect such disease)
- however, when it detects more positives, out of those detected positives may be incorrect; it pays a price for FP and a drop in specificity (the prediction is not specific enough to distinguish between the actual and false positives)

precision-recall trade-off in information retrieval

- a user creates a search query (from a universe of data items) and a relevant list of items is retrieved for the user
- the query has high precision if a large fraction of the retrieved results are relevant
- the query has high recall if it retrieves a large fraction of all relevant items in the universe – this application focuses less on TNR because it is typically high

Trade-off in detecting brain tumor

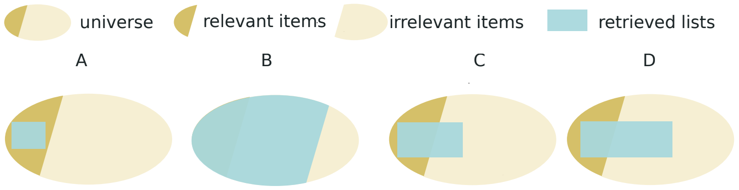
a neuro surgeon makes a decision to remove a brain tumor



- a conservative move (A): avoid to remove healthy cells by extracting only little
- a bold move (D): all tumor must be gone, but unavoidably remove healthy cells
- always consider a combination of (sens,spec) or (precision,recall) – there is a price to pay; when one index increases, the other index would drop

Trade-off in information retrieval

a user creates a search query and a list is retrieved (some are relevent, some are not)



- each case refers to a specific search algorithm, or a parameter value in an algorithm
- explain about classification performance indices for each case

Example: LDA performance on default data

seting: 10,000 training samples, 3.33% of training samples defaulted

		Predicted default		
		Yes	No	Total
Actual default	Yes	81	252	333
	No	23	9,644	9,667
	Total	104	9,896	10,000

performance of LDA on training data:

- FPR $= 23/9{,}667 = 0.238\%$, FNR $= 252/333 = 75.7\%$
- ACC $= (81+9{,}644)/10{,}000 = 97.25\%$, F1 $= 37.07\%$

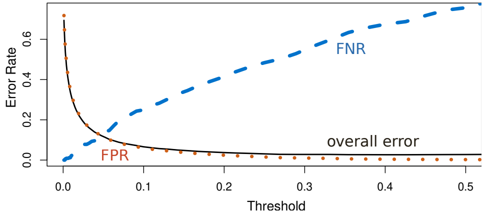
- type I error (FP): incorrectly predicted defaults, type II error (FN): incorrectly predicted non-defaults – a credit card company may wish to avoid FN (more serious) while FP is probably less problematic
- in this example, LDA has a low sensitivity of 24.3% and specificity of 99.76%
- overall accuracy is high while a low sensitivity can tell us the type of error that is more concerned: LDA missed a lot of true defaulters

Adjust a threshold for LDA

the Bayes classifier and LDA uses a threshold of 0.5 for the posterior

$$P(\text{default} = \text{Yes}|X = x) > t, \quad t := 0.5$$

		Predicted default		
		Yes	No	Total
Actual default	Yes	195	138	333
	No	235	9,432	9,667
	Total	430	9,570	10,000

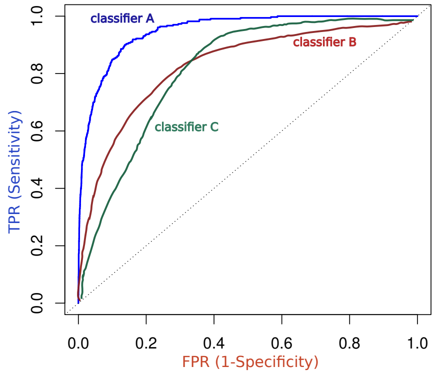


- if t decreases $= 0.2$, more defaults are predicted (more TPR and FPR) – FNR $=$ 41.4%, FPR $=$ 2.43% (higher sens, but slightly lower spec), overall accuracy is 96.27% so overall error rate slightly drops
- as t increases, less predicted defaults, so we have less TPR (hence, higher FNR) and less FPR, shown in the plot

Receiver operating characteristic (ROC)

a plot between FPR and TPR, showing the ability of a binary classifier

example: assign Y to the class '1' if $P(Y = 1|X = x) >$ threshold



- each point on ROC is associated with a threshold/hyperparameter of a classifer
- a classifer performs better than a random guess if ROC lies above the diagonal line
- a better classifier has the ROC curve toward the top left corner
- overall performance is summarized over all possible thresholds is given by area under curve (AUC) (also called the c-statistic)

Matthews correlation coefficient (MCC)

The phi coefficient describes the association of two binary RVs as

$$\phi = \frac{N_{11}N_{00} - N_{10}N_{01}}{\sqrt{N_{:1}N_{:0}N_{1:}N_{0:}}}$$

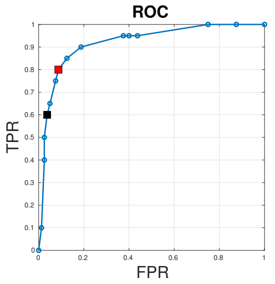
	Y=1	Y = 0	total
X = 1	N11	N10	N1:
X = 0	N01	N00	N0:
total	N:1	N:0	N

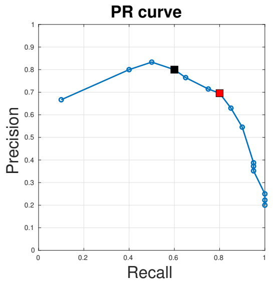
		Predicted	
		positive	negative
Actual	positive	TP	FN
	negative	FP	TN

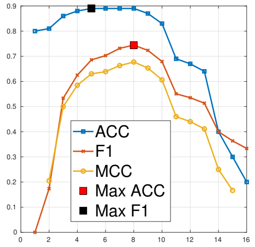
MCC is calculated from the confusion matrix as

$$\text{MCC} = \frac{\text{TP}\cdot\text{TN} - \text{FP}\cdot\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$$

Precision-Recall curve







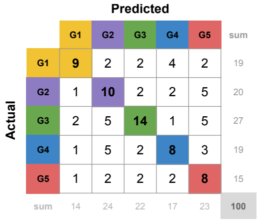
- **black square** is the point for which ACC is maximum
- **red** is the point for which F1 is maximum
- both black and red points lie on the top-left corner of ROC, and on the top-right corner of PR curve, indicating that these points are efficient
- in practice, an operating point can be chosen by selecting the classifer's hyperparameter that maximizes some index (F1,MCC,ACC) on validated data

Confusion matrix of K-class

assume K-class classification labeled as G1,G2,G3,G4,G5 ($K = 5$)

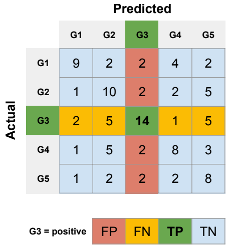
the confusion matrix $\in \mathbf{R}^{K\times K}$ contains the number of samples in each predicted class

calculate according to binary classification indices by picking a class of interest positive, and regard the remaining classes as negative



Predicted

Actual	G1	G2	G3	G4	G5	sum
G1	**9**	2	2	4	2	19
G2	1	**10**	2	2	5	20
G3	2	5	**14**	1	5	27
G4	1	5	2	**8**	3	19
G5	1	2	2	2	**8**	15
sum	14	24	22	17	23	100



Predicted

Actual	G1	G2	G3	G4	G5
G1	9	2	2	4	2
G2	1	10	2	2	5
G3	2	5	14	1	5
G4	1	5	2	8	3
G5	1	2	2	2	8

G3 = positive	FP	FN	TP	TN

- pick G3 as a class of interest
- TP is the diagonal entry (3,3)
- FP is the sum along the 3rd column except (3,3) $= 8$
- FN is the sum along the 3rd row except (3,3) $= 13$

we can calculate precision, recall, and F1 of class G3 as

$$\text{precision} = \frac{14}{22} = 63.6\%, \quad \text{recall} = \frac{14}{27} = 51.9\%, \quad \text{F1} = \frac{2\times 14}{2\times 14 + 8 + 13} = 57.1\%$$

Aggregated F1 score of K-class classsification

given that we have calculated TP, TN, FP, FN for each class as TP_k, $k = 1, 2, \ldots, K$

three common ways to compute F1 score of a K-classifier as one value

1. micro F1: take a sum of all TP_k's and then compute precision/recall
2. macro F1: compute each precision_k and then average
3. weighted F1: weight each F1_k with proportion of samples in each class

$$\text{micro F1} = \text{harmonic mean of precision} = \frac{\sum_i \text{TP}_i}{\sum_i (\text{TP}_i + \text{FP}_i)}, \text{ and recall} = \frac{\sum_i \text{TP}_i}{\sum_i \text{P}_i},$$

$$\text{macro F1} = \text{harmonic mean of precision} = \frac{1}{K} \sum_{i=1}^{K} \left(\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \right), \text{ and recall} = \frac{1}{K} \sum_{i=1}^{K} \frac{\text{TP}_i}{\text{P}_i}$$

$$\text{weighted F1} = \frac{1}{K} \sum_{i=1}^{K} w_i \text{F1}_i$$

Classification methods

Common classifiers

- logistic regression
- k-nearest neighbor (kNN)
- linear/quadratic discrimination analysis
- Naïve Bayes
- support vector machine (SVM)
- tree-based methods
- neural networks

Method comparisons: kNN, logistic regression, LDA, QDA

data generation: 2 predictors; there are 6 scenarios

- Bayes decision boundary are linear in three scenarios, and non-linear in the remaining three
- for each scenario, there are 100 random training data sets
- fit each method and compute the test error rate on a large test set
- kNN uses $K = 1$ and the value selected by cross-validation

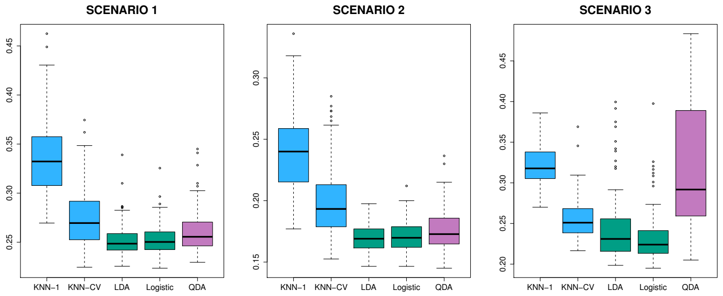
linear cases:

- case 1: 20 training samples, data in each class are uncorrelated normal Gaussian with a different mean
- case 2: as in case 1, but within each class, the two predictors had a correlation of -0.5
- case 3: X_1, X_2 are generated from t-distribution with 50 samples in each class; this set up violates LDA assumption but the decision boundary is still linear

nonlinear cases:

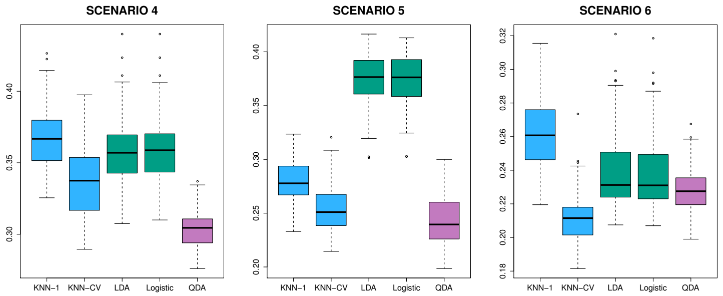
- case 4: data are Gaussian with a correlation of 0.5 between (X_1, X_2) in the first class, and correlation of -0.5 in the second class
- case 5: within each class, the samples are Gaussian with uncorrelated predictors; however, Y is sampled from the logistic function using X_1^2, X_2^2 and $X_1 X_2$
- case 6: as in previous case, but Y are sampled from a more complicated non-linear function

Boxplots of the test error rate: linear case



- case 1,2: LDA performed well (as the setup follows LDA assumptions); kNN performed poorly (too complex); QDA was worse than LDA (more flexible than necessary); as logistic model assumes a linear decision boundary, it is slightly inferior to LDA
- case 3: logistic regression outperformed LDA; QDA performed poorly due to violation of non-normality

Boxplots of the test error rate: nonlinear case



- case 4: fits to QDA assumption, hence QDA outperformed others
- case 5: the setup suggests a quadratic decision boundary; QDA performed best, followed by kNN-CV
- case 6: the setup suggests nonlinear boundaries; kNN-CV had the best result, followed by QDA ; 1NN had the worst result of all methods

Summary of method comparisons

- logistic regression: the model gives the range of Y as probability values for each class; parameters are estimated by maximum likelihood principle
- kNN, LDA, and QDA apply Bayes classifier rule by choosing the class for which the posterior probability is highest
- kNN is a non-parametric method to estimate the posterior probability so no assumptions are made about the shape of decision boundary
- LDA and QDA requires assumption about Gaussian form of the class likelihood: LDA assumes a common covariance, while QDA does not
- for 2-class, both LDA and logistic regression have one thing in common: the log of odds is a linear function of x; they differ by how the linear function coefficients are obtained
- QDA serves as a compromise between non-parametric kNN and linear methods since QDA assumes a quadratic decision boundary
- QDA may perform better than kNN in the presence of low training samples

Generative vs Discriminative models

The posterior probability

$$P(\text{class } k|x) = \frac{f(x|\text{class } k)P(\text{class } k)}{f(x)}$$

three approaches to solve a decision problem

1. generative models: explicitly or implicitly model the joint distribution $f(x, \text{class } k)$ or model the class-conditional density and prior density to form the posterior
   - naive Bayes, LDA, QDA
   - generative model can *generate* samples of target and response from the joint distribution
2. discriminative models: determine the posterior probabilities directly
   - logistic regression, SVM, kNN, decision trees
   - discriminative models *separate* classes directly; they can't be used to generate new data points
3. find a function $g(x)$ called a discriminant function that maps x directly to a class

References

Figures and examples are taken from the first two references (ISLR, ESL)

1. Chapter 1,2,4 in T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, Second edition, 2009
2. Chapter 1-3,4 in G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Application in R*, Springer, 2013
3. Chapter 1, 4 in C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006
4. Chapter 3 in E. Alpaydin, *Introduction to Machine Learning*, 2nd edition, 2010