EML'24 – Lecture 5

Classification II

ISLR 4, ESL 4

Prof. Isabel Valera

14 November 2024

Saarland Informatics Campus

Fitting Classification Models

Fitting Logistic Regression Models

Recall that the multivariate logistic regression model is defined as

- $\log\left(\frac{p(Y=1|X)}{1-p(Y=1|X)}\right) = \beta_0 + \beta_1 X + \cdots \beta_p X_p$ with $p(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X+\cdots+\beta_p X_p}}$

We usually fit a logistic regression model by maximum likelihood

- log-likelihood function $\ell(\theta) = \sum_{i=1}^{n} \log p_{g_i}(x_i;\theta)$ and density function $p_k(x_i,\theta) = \Pr(G = k \mid X = x_i;\theta)$
- for a binary problem, class coding $y_i = \begin{cases} 1 \mid g_i = 1 \\ 0 \mid g_i = 0 \end{cases}$ gives us $p_1(x;\theta) = p(x;\theta)$ and $p_0(x;\theta) = 1 - p(x;\theta)$

The log-likelihood then becomes

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \{y_i \log p(x_i;\boldsymbol{\beta}) + (1-y_i)\log(1-p(x_i;\boldsymbol{\beta}))\} = \sum_{i=1}^{n} \left\{y_i \boldsymbol{\beta}^T x_i - \log\left(1 + e^{\boldsymbol{\beta}^T x_i}\right)\right\}$$

- where $\boldsymbol{\beta} = \{\beta_0, \beta_1, \ldots\}$ and $\boldsymbol{x_i}$ a vector of the input values padded with a constant term $X_0 = 1$

Side calculation

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \{y_i \log p(x_i; \boldsymbol{\beta}) + (1 - y_i) \log(1 - p(x_i; \boldsymbol{\beta}))\}$$

$$= \sum_{i=1}^{n} \left\{ y_i \log \frac{e^{\beta_0 + \beta_1^T x_i}}{1 + e^{\beta_0 + \beta_1^T x_i}} + (1 - y_i) \log \frac{1}{1 + e^{\beta_0 + \beta_1^T x_i}} \right\}$$ (definition of $p(x_i; \boldsymbol{\beta})$)

$$= \sum_{i=1}^{n} \left\{ y_i \left[(\beta_0 + \beta_1^T x_i) - \log\left(1 + e^{\beta_0 + \beta_1^T x_i}\right) \right] - (1 - y_i) \log(1 + e^{\beta_0 + \beta_1^T x_i}) \right\}$$ ($\log a/b = \log a - \log b$)

$$= \sum_{i=1}^{n} \left\{ y_i \boldsymbol{\beta}^T x_i - \log\left(1 + e^{\boldsymbol{\beta}^T x_i}\right) \right\}$$ (simplify)

Fitting Logistic Regression Models

We find the $\boldsymbol{\beta}$ that achieves maximum likelihood by setting the derivative to zero

- this yields the score equations

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} x_i \big(y_i - p(x_i; \boldsymbol{\beta})\big) = 0$$

- these can be broken down to $\boldsymbol{p+1}$ equations that are nonlinear in $\boldsymbol{\beta}$
- because the first value of $\boldsymbol{x_i}$ is 1, the first equation takes the shape

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} p(x_i; \boldsymbol{\beta})$$

- the expected number of class-1 assignments is the number class-1 we observed

Fitting Logistic Regression Models

We can solve the score equations numerically using Newton-Raphson

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \left(\frac{\partial^2 \ell(\boldsymbol{\beta}^{old})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}\right)^{-1} \frac{\partial \ell(\boldsymbol{\beta}^{old})}{\partial \boldsymbol{\beta}}$$

- i.e. adjust coefficients proportionally to second derivative in the opposite direction of first derivative
- repeat until convergence
- note that $\frac{\partial^2 \ell(\beta^{old})}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{n} x_i x_i^T p(x_i;\beta)\big(1 - p(x_i;\beta)\big)$ is our old friend, the Hessian matrix!

Log-likelihood is concave

- single starting point suffices, $\boldsymbol{\beta} = \mathbf{0}$ is fine
- typically converges, but overshooting can occur
- diagonal of the Hessian matrix contains the squared standard deviations of outputs in the training set

Fitting Logistic Regression Models

In matrix notation we have

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{X}^T(\mathbf{y}-\mathbf{p}) \qquad \frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X}$$

- where $\mathbf{W}$ is a diagonal matrix with elements $w_{ii} = -p(x_i; \boldsymbol{\beta}^{old})\left(1 - p(x_i; \boldsymbol{\beta}^{old})\right)$

A single Newton-Raphson step is

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}-\mathbf{p}) = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\left(\mathbf{X}\boldsymbol{\beta}^{old} - \mathbf{W}^{-1}(\mathbf{y}-\mathbf{p})\right) = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}$$

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta}^{old} - \mathbf{W}^{-1}(\mathbf{y}-\mathbf{p})$$

- a linear least-squares problem with output $\mathbf{z}$ weighted by diagonal matrix $\mathbf{W}$

$$\boldsymbol{\beta}^{new} = \arg\min_{\boldsymbol{\beta}} \; (\mathbf{z} - \mathbf{X}\boldsymbol{\beta})^T\mathbf{W}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})$$

Linear Discriminant Analysis

The Bayes-optimal choice is to classify $\boldsymbol{x}$ to the class with the largest discriminant

- the discriminant of a class $\boldsymbol{k}$ is the log-probability that cancels in the log-odds

$$\log\left(\frac{p_k(x)}{p_l(x)}\right) = \delta_k(x) - \delta_l(x)$$

- Where we assume the class-conditional densities to be Gaussian, yielding:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k$$

  is the log-numerator from previous slide with the class-independent terms removed

Fitting Univariate LDA Models

In general, we do not know the underlying class densities but assume they are Gaussians, so that

- we estimate these using the finite training sample

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

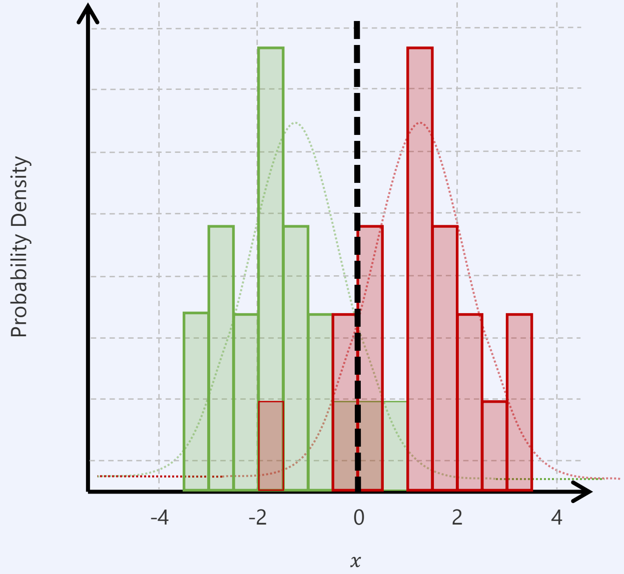
$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

$$\pi = n_k / n$$

- we assign x to the class with the largest fitted discriminant

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log \hat{\pi}_k$$

- note that the discriminants are linear (!)



LDA fit over 20 samples per class, fitted decision boundary in dashed black. Bayes error 10.6%, LDA test error 11.1%

Fitting LDA and QDA Models

Again, we use sample estimates

- $\hat{\mu}_k = \frac{1}{n_k}\sum_{i:y_i=k} x_i$
- $\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$
- $\widehat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k-K}\sum_{i:y_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$
- $\pi_k = n_k/n$

To simplify calculation we use the eigenvalue decomposition of the covariance matrices

$$\widehat{\boldsymbol{\Sigma}}_k = \boldsymbol{U}_k \boldsymbol{D}_k \boldsymbol{U}_k^T$$

- $\boldsymbol{U}_k$ is a $p \times p$ orthonormal matrix
- $\boldsymbol{D}_k$ is a diagonal matrix of decreasing positive eigenvalues $\boldsymbol{d_{kl}}$

The main terms in the discriminants,

$$\delta_k(x) = -\frac{1}{2}\log\left|\widehat{\boldsymbol{\Sigma}}_k\right| - \frac{1}{2}(x-\mu_k)^T\widehat{\boldsymbol{\Sigma}}_k^{-1}(x-\mu_k) + \log\pi_k$$

then turn into

$$\log\left|\widehat{\boldsymbol{\Sigma}}_k\right| = \sum_l \log d_{kl}$$

$$(x-\hat{\mu}_k)^T\hat{\Sigma}_k^{-1}(x-\hat{\mu}_k) = \left[\boldsymbol{U}_k^T(x-\hat{\mu}_k)\right]^T D_k^{-1}\left[U_k^T(x-\hat{\mu}_k)\right]$$

The LDA estimator

- Step 1: Normalize $\boldsymbol{X}$ to spherical covariance

$$X^* \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{U}^T X$$

- Step 2: Classify to the closest class centroid in the transformed space, where distance is weighted by the class prior probabilities $\boldsymbol{\pi_k}$

Types of Errors – a handy guide

Type I error
(false positive)

Type II error
(false negative)

Example on Multivariate LDA

Example **default** with **balance** and **student** as inputs

- training error for LDA is 2.75%
- data is highly unbalanced, we have only 3,33% positives
- the No-only classifier has an error of already only 3,33%

Sensitivity $\text{Sens} = TP/(TP+FN) = TP/P^*$

- fraction of correctly predicted positives

Specificity $\text{Spec} = TN/(TN+FP) = TN/N^*$

- fraction of correctly predicted negatives

- No-only $\text{Sens} = \frac{0}{333} = 0\%$, $\text{Spec}= \frac{9{,}667}{9{,}667} = 100\%$
- LDA $\text{Sens}= \frac{81}{333} = 24.3\%$, $\text{Spec}= \frac{9{,}644}{9{,}667} = 99.8\%$
- LDA approximates the Bayes classifier,
  it minimizes error on all observations

LDA Model Results

Prediction	True Default Status No (−)	Yes (+)	Total
No (−)	9,644	252	9,896
Yes (+)	23	81	104
Total	**9,667**	**333**	**10,000**

Prediction	True Default Status No (−)	Yes (+)	Total
No (−)	TN	FN	N
Yes (+)	FP	TP	P
Total	N^*	P^*	n

Confusion matrix

Type-1 error
False positive

Type-2 error
False negative

Example on Multivariate LDA

Biasing the classifier trades sensitivity for specificity

$$\log((p_k(x))/(p_l(x))) = \delta_k(x) - \delta_l(x)$$

- move the decision threshold between class **no** or **yes** away from
  $\Pr(\textbf{default} = \text{yes} \mid X = x) = P(Y = 1 \mid X) = 0.5$
- we can increase sensitivity by choosing
  $$threhold < 0.5$$
  as this assigns more points to positive class **yes**
- for $\Pr(\textbf{default} = \text{yes} \mid X = x) > 0.2$
  - Sens = $195/333 = 58.6\%$
  - Spec = $9{,}432/9{,}667 = 97.6\%$
  - Error = $373/10{,}000 = 3.73\%$

For a threshold of 0.5 we get
Sens = 24.3%, Spec = 99.8%, Error=2.75%

Prediction	True Default Status No (−)	Yes (+)	Total
No (−)	9,644	252	9,896
Yes (+)	23	81	104
Total	**9,667**	**333**	**10,000**

Prediction	True Default Status No (−)	Yes (+)	Total
No (−)	9,432	138	9,570
Yes (+)	235	195	430
Total	**9,667**	**333**	**10,000**

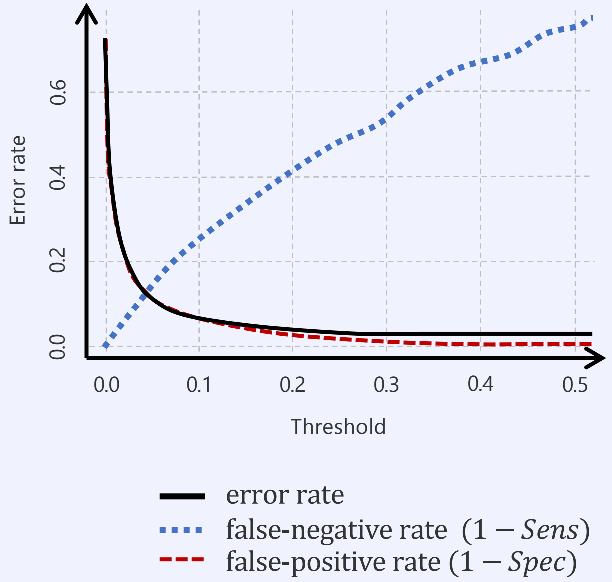
While for a threshold of 0.2 we have
Sens = 58.6%, Spec = 97.6%, Error=3.73%

Example on Multivariate LDA

Biasing the classifier trades sensitivity for specificity

$$\log((p_k(x))/(p_l(x))) = \delta_k(x) - \delta_l(x)$$

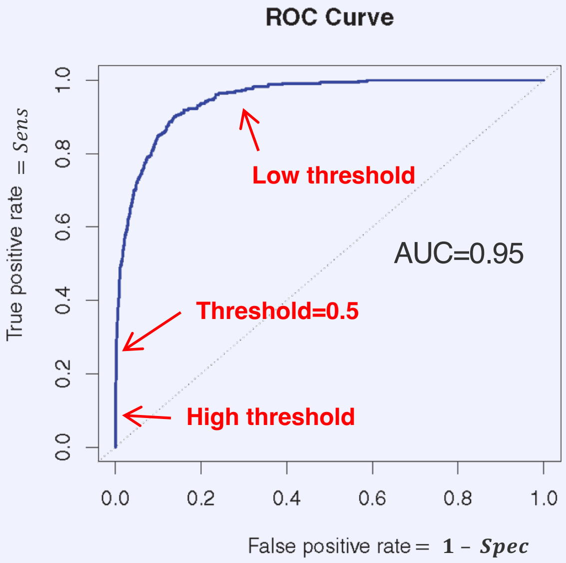
- move the decision threshold between class **no** or **yes** away from
  $\Pr(\textbf{default} = \text{yes} \mid X = x) = P(Y = 1 \mid X) = 0.5$
- we can increase sensitivity by choosing
  $$threhold < 0.5$$
  as this assigns more points to positive class **yes**
- for $\Pr(\textbf{default} = \text{yes} \mid X = x) > 0.2$
  - Sens = $195/333 = 58.6\%$
  - Spec = $9{,}432/9{,}667 = 97.6\%$
  - Error = $373/10{,}000 = 3.73\%$
- error rates change smoothly when we move the threshold



ROC Curves

Receiver-Operating Characteristic (ROC) curves plot $Sens$ against $1 - Spec$ for all thresholds

- Area Under the ROC-Curve (AUC) measures the quality of a classifier independent of the choice of that threshold
- optimally $Spec = Sens = 1$ for any threshold ($AUC = 1$)
- random classifier performs on the diagonal ($AUC = 0.5$)
- if the ROC curve goes below the diagonal, we can improve accuracy by inverting the classifier



ROC curves are not influenced by imbalance of the data

- balance only affects locations of a threshold along the curve

Comparing Different Classifiers

Comparison of the Classification Methods

We now know four classifiers: $\boldsymbol{k}$-NN (L01), LDA, QDA and logistic regression

- when should we use which?

Logistic regression and LDA are surprisingly closely related

- univariate binary setting $p_2(x) = 1 - p_1(x)$
- log-odds for LDA are $\log \frac{p_1(x)}{1-p_1(x)} = c_0 + c_1 x$

  (difference of two linear discriminants)
- while for logistic regression $\log \frac{p_1(x)}{1-p_1(x)} = \beta_0 + \beta_1 x$

Similar, but different

- $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ are maximum likelihood estimates
- $\boldsymbol{c_0}$ and $\boldsymbol{c_1}$ are estimated from sample mean and variance of Gaussian distribution
- relationship extends to multivariate data: LR and LDA often give similar results – but not always!
- LDA makes stronger assumptions (i.e., Gaussian class-conditional density)

Comparison of the Classification Methods

We now know four classifiers: k-NN, LDA, QDA and logistic regression

- when should we use which?

k-NN is nonparametric and tends to work better for strongly nonlinear settings

- it does not allow for inference, i.e. we do not get a model that we can learn from

QDA is a compromise between LDA and k-NN

Logistic regression very often works great in practice, and one can transform the features to have non-linear classification with respect to original features. Often used as baseline!

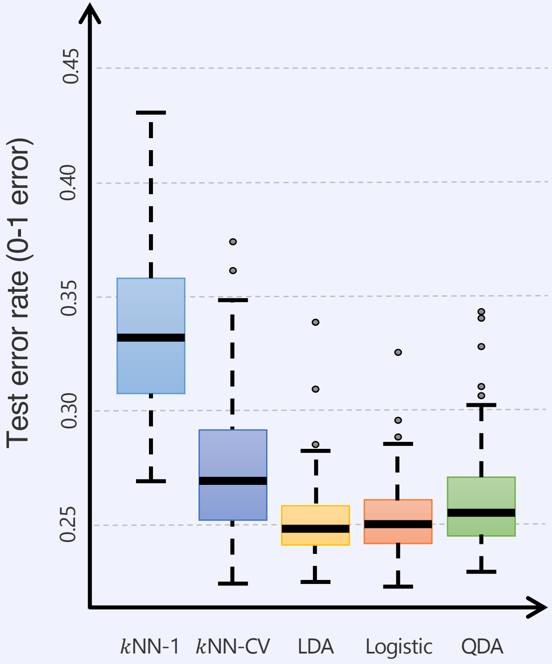
Comparing the Classification Methods

Scenario 1

- 100 random training data sets, $p = 2$ predictors, $K = 2$ classes
- 20 observations per class
- observations in different classes uncorrelated normal variables with different means and the same variance (spherical Gaussian)
- this matches the LDA assumptions of LDA

Observations

- LDA works very well
- logistic regression assumes a linear decision boundary, performs only slightly worse than LDA
- k-NN overfits, as does QDA



Scenario 1

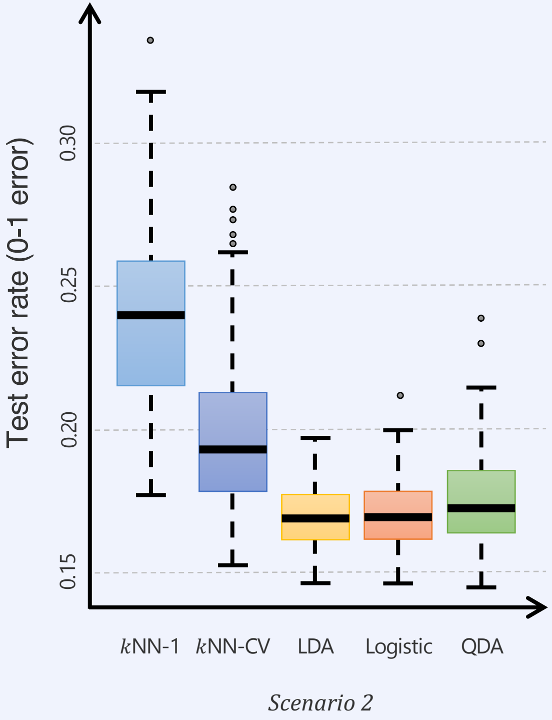
Comparing the Classification Methods

Scenario 2

- 100 random training data sets, $p = 2$ predictors, $K = 2$ classes
- like scenario 1, but predictors in each class now have a correlation of -0.5 (elliptical multivariate Gaussian)

Observations

- relative performances are similar to scenario 1 but with QDA competing as we match its assumptions.



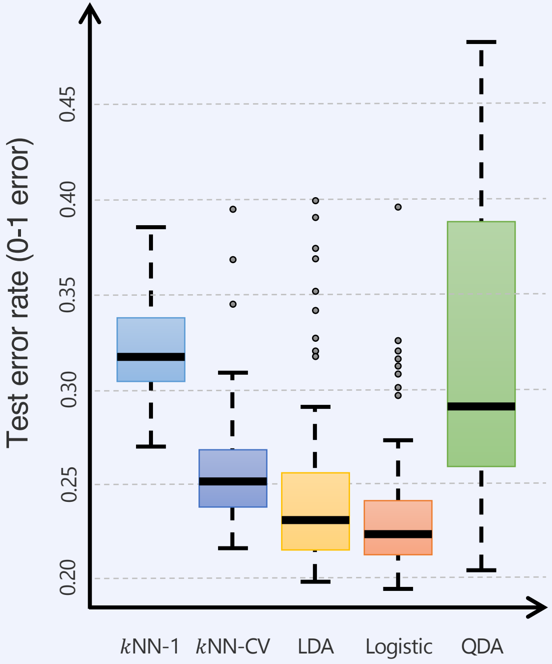
Comparing the Classification Methods

Scenario 3

- 100 random training data sets, $p = 2$ predictors, $K = 2$ classes
- X_1 and X_2 are generated using a $\boldsymbol{t}$-distribution
- more extreme points than with a Gaussian
- decision boundary is linear, but, setup violates LDA assumption

Observations

- logistic regression performs best
- QDA deteriorates because of non-normality of the data



Scenario 3

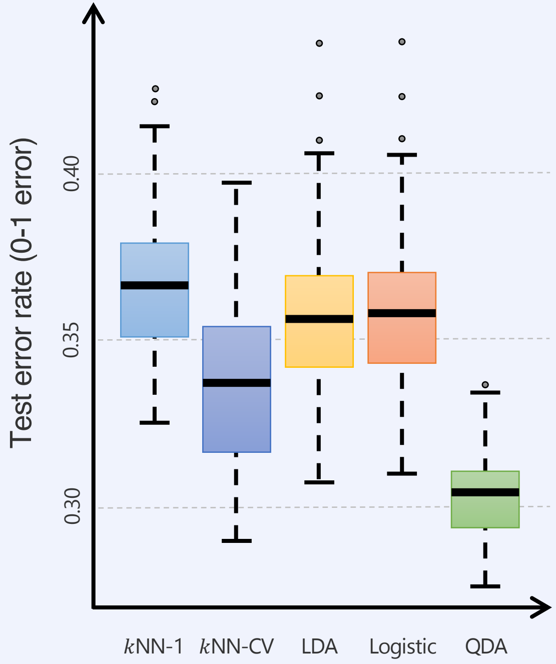
Comparing the Classification Methods

Scenario 4

- 100 random training data sets, $p = 2$ predictors, $K = 2$ classes
- class 1: normal distribution with correlation $\mathbf{0.5}$ to predictors
- class 2: normal distribution with correlation $\mathbf{-0.5}$ to predictors
- assumptions of QDA are met (but not LDA!)

Observations

- QDA outperforms all other methods



Scenario 4

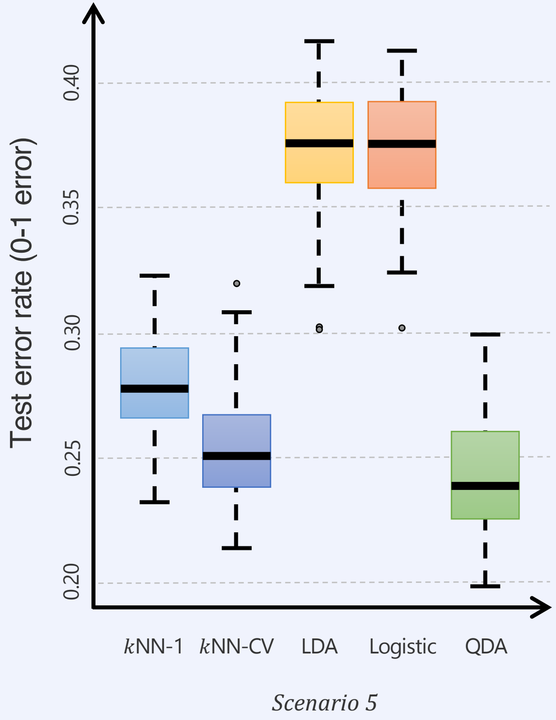
Comparing the Classification Methods

Scenario 5

- 100 random training data sets, $p = 2$ predictors, $K = 2$ classes
- two normal distributions with uncorrelated predictors
- inputs X_1^2, X_2^2 and $X_1 X_2$, not X_1 and X_2
- the decision boundary is quadratic

Observations

- QDA performs best
- kNN (CV) follows closely
- the linear methods all perform poorly



V

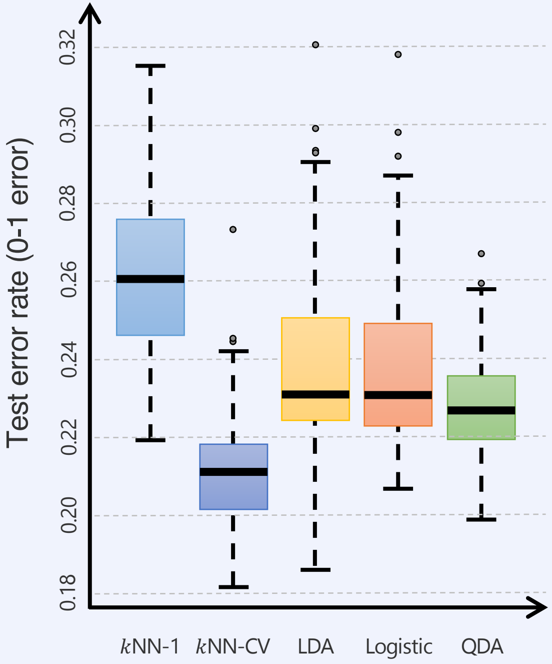
Comparing the Classification Methods

Scenario 6

- 100 random training data sets, $p = 2$ predictors, $K = 2$ classes
- like scenario 5, but responses sampled from a complicated linear function

Observations

- even QDA cannot model data well
- k-NN-1 overfits
- k-NN (cross-validated) outperforms all parametric approaches
- smoothness must be chosen carefully



Scenario 6