Statistics 101C - Week 3 Thursday

Shirong Xu

University of California, Los Angeles

shirong@stat.ucla.edu

October 16, 2024

Classification Methods

- We have studied four classification algorithms in the past lectures
    - Logistic Regression
    - Linear discriminant analysis
    - Quadratic discriminant analysis
    - K-nearest neighbors
- **Question**:
    - What is the difference between these four algorithms?
    - How can we choose the most appropriate algorithm in practice?

LDA vs QDA

In both LDA and QDA, we use the conditional probability $\widehat{P}(Y = 1|\boldsymbol{X})$ for classification

$$\widehat{P}(Y = 1|\boldsymbol{X}) > 1/2 \Rightarrow 1$$
$$\widehat{P}(Y = 1|\boldsymbol{X}) < 1/2 \Rightarrow 0$$

LDA vs QDA

In both LDA and QDA, we use the conditional probability $\widehat{P}(Y=1|\boldsymbol{X})$ for classification

$$\widehat{P}(Y=1|\boldsymbol{X}) > 1/2 \Rightarrow 1$$
$$\widehat{P}(Y=1|\boldsymbol{X}) < 1/2 \Rightarrow 0$$

Then, we can derive the decision boundaries as $\widehat{P}(Y=1|\boldsymbol{X}) = 1/2$. In LDA, it yields a linear equation.

$$\boldsymbol{x}^T C_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + C_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = 0,$$

- $C_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$
- $C_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = \log \frac{\pi_1}{\pi_0} - \frac{1}{2}(\boldsymbol{\mu_1} + \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_1} - \boldsymbol{\mu}_0)$

LDA vs QDA

Suppose we have a new sample $\boldsymbol{x}_0$ and we use the conditional probability $\widehat{P}(Y = 1|\boldsymbol{X})$ for prediction.

- Prediction is 1 when

$$\widehat{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0) > 1/2$$

$$\Updownarrow$$

$$\underbrace{\boldsymbol{x}_0}_{\text{new sample}}^T C_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + C_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) > 0,$$

LDA vs QDA

Suppose we have a new sample $\boldsymbol{x}_0$ and we use the conditional probability $\widehat{P}(Y = 1|\boldsymbol{X})$ for prediction.

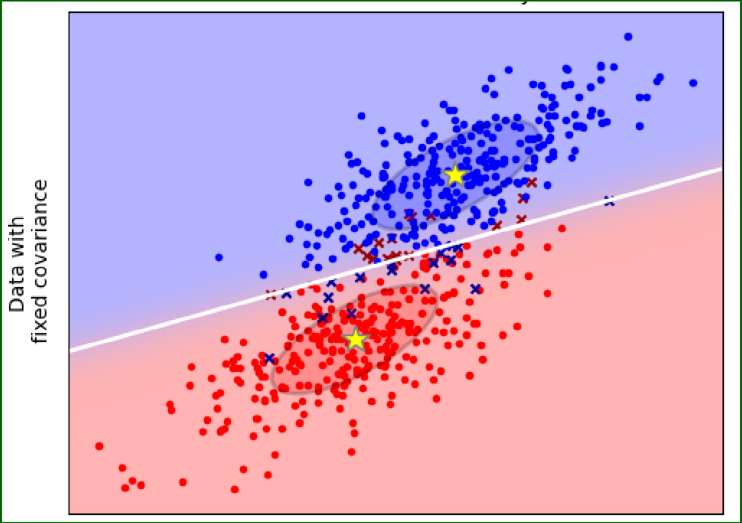
- Prediction is 1 when

$$\widehat{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0) > 1/2$$
$$\Updownarrow$$
$$\underbrace{\boldsymbol{x}_0^T}_{\text{new sample}} C_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + C_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) > 0,$$

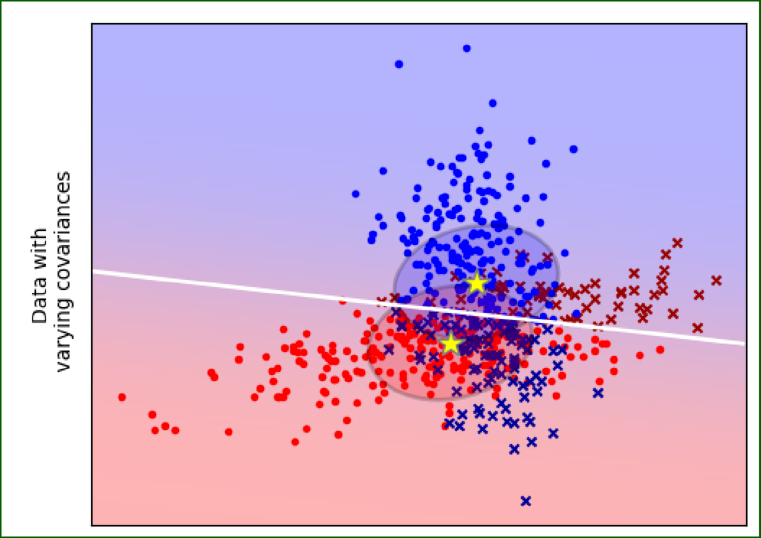
- Prediction is 0 when

$$\widehat{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0) < 1/2$$
$$\Updownarrow$$
$$\underbrace{\boldsymbol{x}_0^T}_{\text{new sample}} C_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + C_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) < 0,$$

LDA: when assumptions holds



LDA: when assumptions does not hold



QDA

In QDA, we also use the conditional probability $\widehat{P}(Y = 1|\boldsymbol{X})$ for classification

$$\widehat{P}(Y = 1|\boldsymbol{X}) > 1/2 \Rightarrow 1$$
$$\widehat{P}(Y = 1|\boldsymbol{X}) < 1/2 \Rightarrow 0$$

Then, we can derive the decision boundaries as $\widehat{P}(Y = 1|\boldsymbol{X}) = 1/2$. In QDA, it yields a quadratic equation.

$$\boldsymbol{x}^T D_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1)\boldsymbol{x} + \boldsymbol{x}^T D_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1) + D_3(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1) = 0,$$

- $D_1(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1) = \boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}$
- $D_2(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1) = \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0$
- $D_3(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1) = \log \frac{\pi_1}{\pi_0} + \frac{1}{2} \log \frac{|\Sigma_0|}{|\Sigma_1|} - \frac{1}{2}\boldsymbol{\mu}_1^T \Sigma_1^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_0^T \Sigma_0^{-1} \boldsymbol{\mu}_0$

LDA vs QDA

- In LDA, we know the decision boundary is a **hyperplane** (linear equation)
- **Question**: What is the decision boundary of QDA?

Quadratic equation

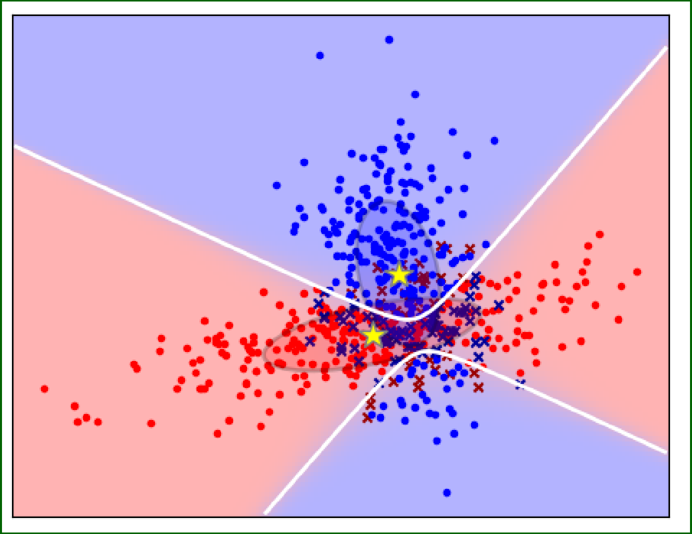
- Ellipsoid:

$$(x_1, x_2)\begin{pmatrix}1, 0\\0, 2\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} = 1 \Leftrightarrow x_1^2 + 2x_2^2 = 1$$

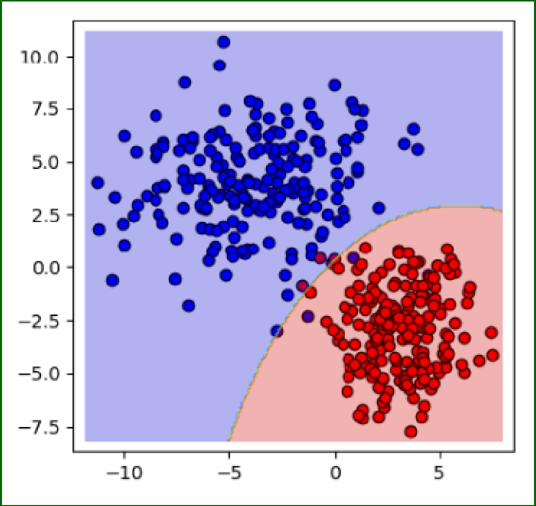
- hyperbolic equation:

$$(x_1, x_2)\begin{pmatrix}1, 0\\0, -2\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} = 1 \Leftrightarrow x_1^2 - 2x_2^2 = 1$$

Decision boundary of QDA: hyperbola



Decision boundary of QDA: ellipsoid



Conclusions from comparing LDA and QDA

- LDA is to find a **hyperplane** to separate different classes.
- QDA is to find a **ellipsoid** or **hyperbola** to separate different classes.

Logistic Regression

- In logistic regression, it is assume that

$$\log\left(\frac{\mathbb{P}(Y=1|\boldsymbol{X})}{\mathbb{P}(Y=0|\boldsymbol{X})}\right)=\beta_0+\boldsymbol{\beta}^T\boldsymbol{x}$$

Logistic Regression

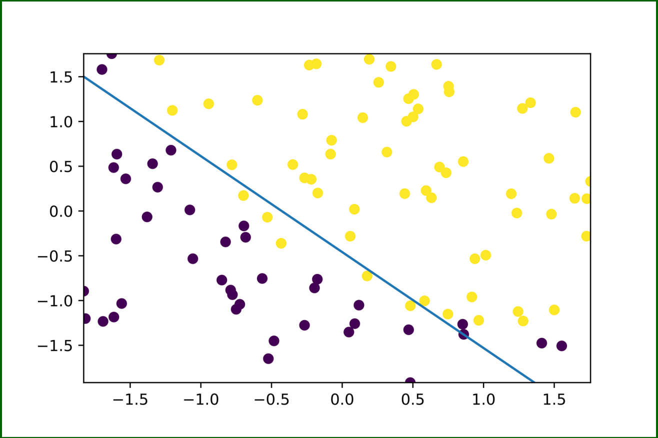
- In logistic regression, it is assume that

$$\log\left(\frac{\mathbb{P}(Y=1|\boldsymbol{X})}{\mathbb{P}(Y=0|\boldsymbol{X})}\right)=\beta_0+\boldsymbol{\beta}^T\boldsymbol{x}$$

- After we estimate β_0 and $\boldsymbol{\beta}$, we can implement prediction for a new sample $\boldsymbol{x}_0$ as

$$\widehat{\beta}_0+\widehat{\boldsymbol{\beta}}^T\boldsymbol{x}>0 \Leftrightarrow \text{Prediction is } 1$$
$$\widehat{\beta}_0+\widehat{\boldsymbol{\beta}}^T\boldsymbol{x}<0 \Leftrightarrow \text{Prediction is } 0$$

Logistic Regression

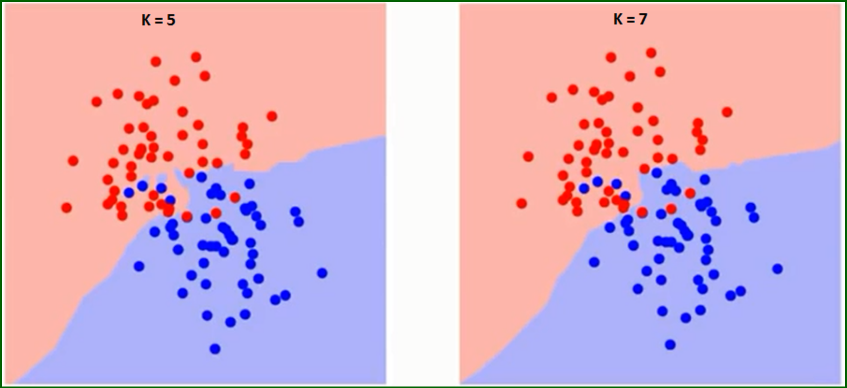


Question: Both LDA and logistic regression intend to find a **hyperplan** to separate two classes. What is the difference between these two algorithms?

LDA vs Logistic Regression

1. Logistic regression and LDA differ only in their fitting procedures, one might expect the two approaches to give similar results.
2. LDA assumes that the observations are drawn from a Gaussian distribution with a common covariance matrix in each class, and so can provide some improvements over logistic regression when this assumption approximately holds
3. logistic regression can outperform LDA if these Gaussian assumptions are not met

Decision boundary of KNN

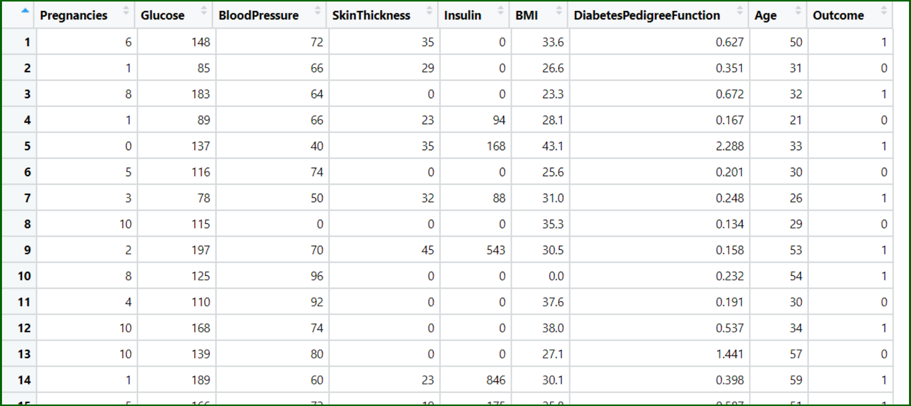


- KNN is a completely **non-parametric approach**: no assumptions are made about the shape of the decision boundary.
- KNN will dominate LDA and logistic regression when the decision boundary is highly non-linear.

Comparison: KNN vs (LDA, QDA, Logistic regression)

- KNN is **non-parametric** models
- LDA, QDA, Logistic regression are all **parametric** models
- KNN is more powerful when your training dataset is super large

Experiment: Diabetes Dataset



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31.0	0.248	26	1
8	10	115	0	0	0	35.3	0.134	29	0
9	2	197	70	45	543	30.5	0.158	53	1
10	8	125	96	0	0	0.0	0.232	54	1
11	4	110	92	0	0	37.6	0.191	30	0
12	10	168	74	0	0	38.0	0.537	34	1
13	10	139	80	0	0	27.1	1.441	57	0
14	1	189	60	23	846	30.1	0.398	59	1

Cross Validation: Background

- In the past, we always assume that we have a training dataset $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and a testing dataset $D_e = \{(\boldsymbol{x}_i', y_i')\}_{i=1}^m$. The test error can be easily calculated if a testing dataset is available. Unfortunately, this is usually not the case.
- **Motivation**: In practice, we only have a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and we need to figure a way to estimate the **testing error** based on D.
- **Question:** Why do we need to know testing error?

Cross Validation: Background

- In the past, we always assume that we have a training dataset $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and a testing dataset $D_e = \{(\boldsymbol{x}_i', y_i')\}_{i=1}^m$. The test error can be easily calculated if a testing dataset is available. Unfortunately, this is usually not the case.

- **Motivation**: In practice, we only have a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and we need to figure a way to estimate the **testing error** based on D.

- **Question:** Why do we need to know testing error?
- **Answer**: Because we need to know the true performance of model before we deploy it (economic cost).

Training - Validation Split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validation. The error on validation set can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.

Training - Validation Split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validation. The error on validation set can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.
- Disadvantages of this approach:

Training - Validation Split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validation. The error on validation set can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.
- Disadvantages of this approach:
  (1) the **validation estimate** of the test error rate can be **highly variable**, depending on how you split the dataset.
  (2) In the validation approach, only a subset of training set are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

Cross-Validation

- **Definition**: Cross-validation is a re-sampling method that uses different portions of the data to test and train a model on different iterations.
- **Objective**: Estimate the test error associated with a given statistical learning method in order to
  - evaluate its performance (model assessment)
  - select the appropriate level of flexibility (select the best model or parameters)

Cross-validation

- **Test error rate**: A measurement based on data that is not used in training a statistical learning method.
- In the absence of a very large designated test set, one way to estimate the test error rate is **Cross-validation**.
- **Mechanism**: holding out a subset of the training observations from the fitting process and then applying the statistical learning method to those held out observations

Cross-validation for regression

Background

- Suppose we observe a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ is a p-dimensional covariate vector and $y_i \in \mathbb{R}$ is a scalar.
- We use the training data to construct $\widehat{f} : \mathbb{R}^p \to \mathbb{R}$, which aims to mimics $f(\boldsymbol{X}) = \mathbb{E}(Y|\boldsymbol{X})$

$$\widehat{f} = \arg\min_f \frac{1}{n} \sum_{i=1}^n (y_i - f(\boldsymbol{x}_i))^2.$$

- We assess the quality of $\widehat{f}$ for predicting Y using the square error.

$$R(\widehat{f}) = \mathbb{E}\Big(Y - \widehat{f}(\boldsymbol{X})\Big)^2$$

Background

- Obtain $R(\widehat{f})$ is impossible in practice, which requires the population of $(\boldsymbol{X}, Y)$. So we need to estimate it.

- A very good estimator is the Test MSE, which is computed using test data. Specifically, let $\{(\boldsymbol{x}_i', y_i')\}_{i=1}^m$, the testing MSE is computed as

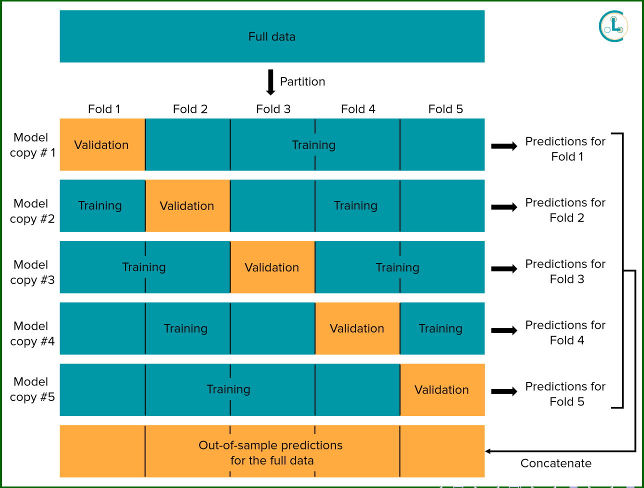
$$\text{Testing MSE} : R_m(\widehat{f}) = \frac{1}{m}\sum_{i=1}^{m}(\widehat{f}(\boldsymbol{x}_i') - y_i')^2$$

- By law of large number, we have

$$R_m(\widehat{f}) \to R(\widehat{f}), \text{ as } m \to \infty.$$

K-fold Cross-Validation

- **Basic Idea**: Divide the training data into K equallysized divisions or folds



6-fold Cross-Validation: Procedure

1. Suppose a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{6n}$ is given. We split it into 6 parts as

$$D_1 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=n+1}^{2n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=2n+1}^{3n}$$
$$D_4 = \{(\boldsymbol{x}_i, y_i)\}_{i=3n+1}^{4n}, D_5 = \{(\boldsymbol{x}_i, y_i)\}_{i=4n+1}^{5n}, D_6 = \{(\boldsymbol{x}_i, y_i)\}_{i=5n+1}^{6n}$$

2. We repeat the following step from $j = 1, 2, 3, 4, 5, 6$: (1) Construct a dataset $D_{-j} = \cup_{i \neq j} D_i$; (2) Train a function via

$$\widehat{f}_{-j} = \arg\min_{f \in \mathcal{F}} \frac{1}{5n} \sum_{i \in D_{-j}} (\widehat{f}(\boldsymbol{x}_i) - y_i)^2$$

3. Compute the validation error of $\widehat{f}_{-j}, j = 1, \ldots, 6$

$$VE_j(\widehat{f}_{-j}) = \frac{1}{n} \sum_{i=(j-1)n+1}^{jn} (\widehat{f}(\boldsymbol{x}_i) - y_i)^2$$

6-fold Cross-Validation: Procedure

4. Use the averaged validation errors as an estimate of testing error

$$\text{Testing Error Estimate} : \frac{1}{6}\sum_{j=1}^{6} VE_j(\widehat{f}_{-j})$$

Leave-one-out cross-validation (LOOCV)

- **Definition**: Special case of K-fold validation when K = n, called leave-one-out cross-validation.
- **Procedure**: Repeat the following steps for $j = 1, \ldots, n$
  - For a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we split it into two parts $D_{train} = \{(\boldsymbol{x}_i, y_i)\}_{i \neq j}$ and $D_{test} = \{(\boldsymbol{x}_j, y_j)\}$
  - Train a model with the dataset D_{train}. Let $\widehat{f}$ denote the model trained from D_{train}, we then test it on D_{test}
  - Compute the averaged testing errors.

Discussion of LOOCV

- Computationally expensive (or even infeasible) when the number of observations in the training data is large. Except if you are using linear regression.

Discussion of LOOCV

- Computationally expensive (or even infeasible) when the number of observations in the training data is large. Except if you are using linear regression.
- The validation MSE from LOOCV is based on averaging n individual fold-based error estimates. Each of these individual estimates is based on almost the same data. Therefore, these estimates are highly correlated with each other

Bias-Variance Trade-Off for k-Fold Cross-Validation

- **Recall**: The validation set approach can lead to overestimates of the test error rate.
- LOOCV approximately unbiased estimates of the test error, since each training set contains $n-1$ observations, which is almost as many as the number of observations in the full data set.

Bias-Variance Trade-Off for k-Fold Cross-Validation

- **Conclusions**:
  - k-fold CV (Medium K) often gives accurate estimates of the **test error rate** than LOOCV.

Bias-Variance Trade-Off for k-Fold Cross-Validation

- **Conclusions**:
  - k-fold CV (Medium K) often gives accurate estimates of the **test error rate** than LOOCV.
  - There is a **bias-variance trade-off** associated with the choice of k in k-fold cross-validation.

Bias-Variance Trade-Off for k-Fold Cross-Validation

- **Conclusions**:
  - k-fold CV (Medium K) often gives accurate estimates of the **test error rate** than LOOCV.
  - There is a **bias-variance trade-off** associated with the choice of k in k-fold cross-validation.
  - The test error estimate resulting from LOOCV tends to have **higher variance** than k-fold CV.

Bias-Variance Trade-Off for k-Fold Cross-Validation

- **Conclusions**:
  - k-fold CV (Medium K) often gives accurate estimates of the **test error rate** than LOOCV.
  - There is a **bias-variance trade-off** associated with the choice of k in k-fold cross-validation.
  - The test error estimate resulting from LOOCV tends to have **higher variance** than k-fold CV.
  - LOOCV is **less biased** than k-fold CV.