

# Lecture 5: Evaluation and Training

STATS 202: Data Mining and Analysis

Linh Tran

tranlm@stanford.edu



Department of Statistics  
Stanford University

July 7, 2021



- ▶ HW2 due a 1 week from Friday.
- ▶ HW1 hopefully graded by next Wednesday.
  - ▶ Solutions will be posted next Monday.



- ▶ Linear/Quadratic Discriminant Analysis
- ▶ Evaluating classification models
  - ▶ Confusion matrix
  - ▶ Receiver Operating Characteristic curve
- ▶ Validation sets
  - ▶ Data splitting
  - ▶ Leave one out cross-validation
  - ▶ K-fold cross-validation
  - ▶ Cross-validation theory
  - ▶ Ensembles



Previously, we:

- ▶ Defined Multiple Linear Regression
- ▶ Generalized to Logistic Regression
- ▶ Covered potential issues with linear models

Are there other (types of) linear models?



A linear model (like logistic regression). Unlike logistic regression:

- ▶ Does not become unstable when classes are well separated
- ▶ With small  $n$  and  $\mathbf{X}$  approximately normal, is stable
- ▶ Popular when we have  $> 2$  classes



A linear model (like logistic regression). Unlike logistic regression:

- ▶ Does not become unstable when classes are well separated
- ▶ With small  $n$  and  $\mathbf{X}$  approximately normal, is stable
- ▶ Popular when we have  $> 2$  classes

## High level idea:

Model distribution of  $\mathbf{X}$  given  $Y$ , and apply Bayes' theorem, i.e.

$$\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})} \quad (1)$$

- ▶ A common assumption is  $f_k(\mathbf{x})$  is Gaussian



**Example:**  $K = 2$  with Gaussian  $f_k(\mathbf{x})$  and common  $\sigma^2$

$$\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})} \quad (2)$$

$$= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^2 \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \quad (3)$$



**Example:**  $K = 2$  with Gaussian  $f_k(\mathbf{x})$  and common  $\sigma^2$

$$\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})} \quad (2)$$

$$= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^2 \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \quad (3)$$

Taking the log and rearranging gives:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \quad (4)$$





**Example:**  $K = 2$  with Gaussian  $f_k(\mathbf{x})$  and common  $\sigma^2$

$$\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})} \quad (2)$$

$$= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^2 \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \quad (3)$$

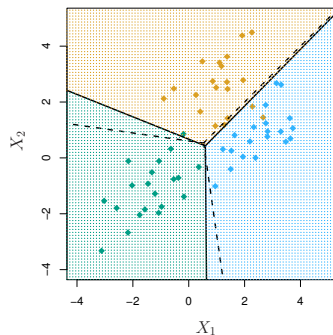
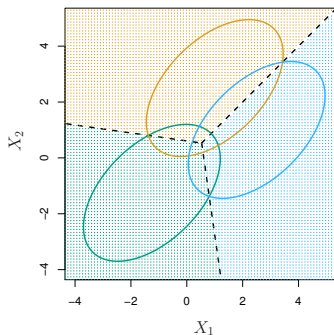
Taking the log and rearranging gives:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \quad (4)$$

If  $\pi_1 = \pi_2$ , our Bayes Classifier is:

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2 \quad (5)$$

## Example of LDA





Similar to LDA

- ▶ Assumes Gaussian  $f_k(\mathbf{x})$
- ▶ Unlike LDA:
  - ▶ Assumes each class has its own covariance matrix ( $\Sigma_k$ )



Similar to LDA

- ▶ Assumes Gaussian  $f_k(\mathbf{x})$
- ▶ Unlike LDA:
  - ▶ Assumes each class has its own covariance matrix ( $\Sigma_k$ )

This results in a quadratic discriminant function:

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \log |\Sigma_k| + \log \pi_k \\ &= -\frac{1}{2}x^\top \Sigma_k^{-1}x + x^\top \Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^\top \Sigma_k^{-1}\mu_k - \frac{1}{2} \log |\Sigma_k| + \log \pi_k\end{aligned}\tag{6}$$



Similar to LDA

- ▶ Assumes Gaussian  $f_k(\mathbf{x})$
- ▶ Unlike LDA:
  - ▶ Assumes each class has its own covariance matrix ( $\Sigma_k$ )

This results in a quadratic discriminant function:

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \log |\Sigma_k| + \log \pi_k \\ &= -\frac{1}{2}x^\top \Sigma_k^{-1}x + x^\top \Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^\top \Sigma_k^{-1}\mu_k - \frac{1}{2} \log |\Sigma_k| + \log \pi_k\end{aligned}\tag{6}$$

This results in more parameters to fit:

- ▶ LDA:  $Kp$  parameters
- ▶ QDA:  $Kp(p+1)/2$  parameters



Assume a two-class setting with one predictor

## Linear Discriminant Analysis:

$$\log \left[ \frac{p_1(x)}{1 - p_1(x)} \right] = c_0 + c_1 x \quad (7)$$

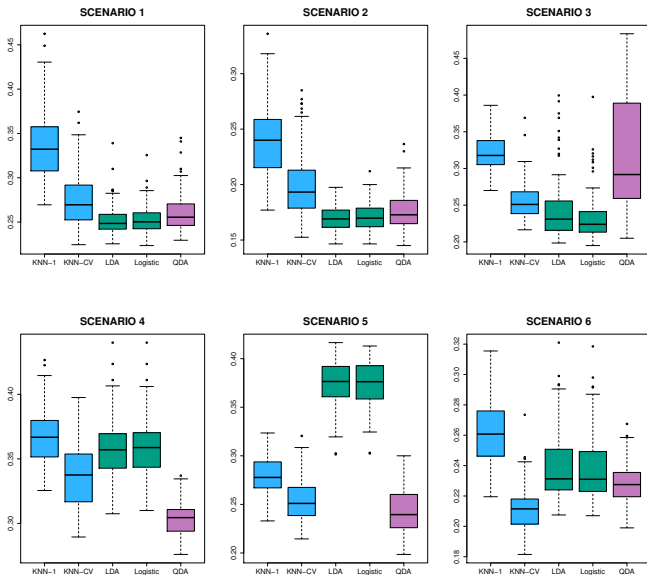
- $c_0$  and  $c_1$  computed using  $\hat{\mu}_0$ ,  $\hat{\mu}_1$ , and  $\hat{\sigma}^2$

## Logistic regression:

$$\log \left[ \frac{\mathbb{P}[Y = 1|x]}{1 - \mathbb{P}[Y = 1|x]} \right] = \beta_0 + \beta_1 x \quad (8)$$

- $\beta_0$  and  $\beta_1$  estimated using MLE

# Comparison of classification methods





**Recall:** Our standard prediction error functions are

- **Classification:** Cross-entropy

$$\hat{CE}(\hat{f}_n) = \mathbb{E}_n[-y \log \hat{f}_n(x)] \quad (9)$$

- **Regression:** Mean squared error

$$\hat{MSE}(\hat{f}_n) = \mathbb{E}_n[y - \hat{f}_n(x)]^2 \quad (10)$$

While MSE has an intuitive interpretation, CE is harder to explain.

- Typically, other losses are used to evaluate classification methods





Many practitioners use the 0-1 loss:

$$\frac{1}{n} \sum_{i=1}^n \mathbb{I}[y_i \neq \hat{y}_i] \quad (11)$$

- Can be thought of as 1 – accuracy



Many practitioners use the 0-1 loss:

$$\frac{1}{n} \sum_{i=1}^n \mathbb{I}[y_i \neq \hat{y}_i] \quad (11)$$

- ▶ Can be thought of as 1 – accuracy
- ▶ Possible to make the wrong prediction for some classes more often than others
  - ▶ The 0-1 loss doesn't tell you anything about this
- ▶ A much more informative error summary is a *confusion matrix*

		<i>Predicted class</i>		
		– or Null	+ or Non-null	Total
<i>True class</i>	– or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
Total		N*	P*	



		<i>Predicted class</i>		
		– or Null	+ or Non-null	Total
<i>True class</i>	– or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
Total		N*	P*	

We can calculate a number of statistics from this table, e.g.

- ▶ True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\textit{Predicted} + \mid \textit{True} +], \text{i.e. } TP/P \quad (12)$$



		<i>Predicted class</i>		
		– or Null	+ or Non-null	Total
<i>True class</i>	– or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
Total		N*	P*	

We can calculate a number of statistics from this table, e.g.

- ▶ True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\textit{Predicted} + \mid \textit{True} +], \text{i.e. } TP/P \quad (12)$$

- ▶ True negative rate (aka Specificity)

$$\mathbb{P}[\textit{Predicted} - \mid \textit{True} -], \text{i.e. } TN/N \quad (13)$$



		<i>Predicted class</i>		
		– or Null	+ or Non-null	Total
<i>True class</i>	– or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
Total		N*	P*	

We can calculate a number of statistics from this table, e.g.

- ▶ True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\text{Predicted} + \mid \text{True} +], \text{i.e. } TP/P \quad (12)$$

- ▶ True negative rate (aka Specificity)

$$\mathbb{P}[\text{Predicted} - \mid \text{True} -], \text{i.e. } TN/N \quad (13)$$

- ▶ Positive predicted value (aka Precision)

$$\mathbb{P}[\text{True} + \mid \text{Predicted} +], \text{i.e. } TP/P^* \quad (14)$$



		<i>Predicted class</i>		
		– or Null	+ or Non-null	Total
<i>True class</i>	– or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
Total		N*	P*	

We can calculate a number of statistics from this table, e.g.

- ▶ True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\text{Predicted} + | \text{True} +], \text{i.e. } TP/P \quad (12)$$

- ▶ True negative rate (aka Specificity)

$$\mathbb{P}[\text{Predicted} - | \text{True} -], \text{i.e. } TN/N \quad (13)$$

- ▶ Positive predicted value (aka Precision)

$$\mathbb{P}[\text{True} + | \text{Predicted} +], \text{i.e. } TP/P^* \quad (14)$$

- ▶ Negative predicted value

$$\mathbb{P}[\text{True} - | \text{Predicted} -], \text{i.e. } TN/N^* \quad (15)$$



		<i>Predicted class</i>		
		– or Null	+ or Non-null	Total
<i>True class</i>	– or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

We can also calculate summary statistics, e.g. the *F1*-score

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (16)$$

# Example: Predicting default



Predicting credit card default in a dataset of 10K people

- Predicted “yes” if  $\hat{\mathbb{P}}_n[\text{default} = \text{yes}|\mathbf{X}] > 0.5$

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000



# Example: Predicting default



Predicting credit card default in a dataset of 10K people

- Predicted “yes” if  $\hat{\mathbb{P}}_n[\text{default} = \text{yes}|\mathbf{X}] > 0.5$

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,644	252	9,896
	Yes	23	81	104
Total		9,667	333	10,000

- The error rate among people who do not default is very low (i.e. high specificity)

# Example: Predicting default



Predicting credit card default in a dataset of 10K people

- ▶ Predicted “yes” if  $\hat{\mathbb{P}}_n[\text{default} = \text{yes}|\mathbf{X}] > 0.5$

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,644	252	9,896
	Yes	23	81	104
Total		9,667	333	10,000

- ▶ The error rate among people who do not default is very low (i.e. high specificity)
- ▶ The error rate among people who default (false negative rate) is 76% (i.e. low sensitivity)
  - ▶ i.e.  $FN/P$



Predicting credit card default in a dataset of 10K people

- ▶ Predicted “yes” if  $\hat{\mathbb{P}}_n[\text{default} = \text{yes}|\mathbf{X}] > 0.5$

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,644	252	9,896
	Yes	23	81	104
Total		9,667	333	10,000

- ▶ The error rate among people who do not default is very low (i.e. high specificity)
- ▶ The error rate among people who default (false negative rate) is 76% (i.e. low sensitivity)
  - ▶ i.e.  $FN/P$
- ▶ It's likely that false negatives are a bigger source of concern

# Example: Predicting default



Predicting credit card default in a dataset of 10K people

- ▶ Predicted “yes” if  $\hat{\mathbb{P}}_n[\text{default} = \text{yes}|\mathbf{X}] > 0.5$

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,644	252	9,896
	Yes	23	81	104
Total		9,667	333	10,000

- ▶ The error rate among people who do not default is very low (i.e. high specificity)
- ▶ The error rate among people who default (false negative rate) is 76% (i.e. low sensitivity)
  - ▶ i.e.  $FN/P$
- ▶ It's likely that false negatives are a bigger source of concern
- ▶ **Possible solution:** Change the classifier *threshold*



Predicting credit card default in a dataset of 10K people

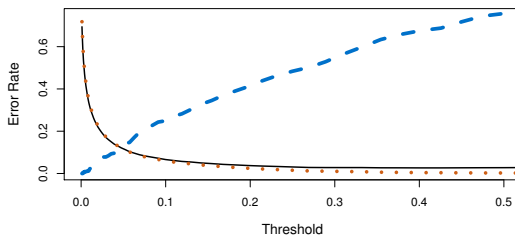
- ▶ Predicted “yes” if  $\hat{\mathbb{P}}_n[\text{default} = \text{yes}|\mathbf{X}] > 0.2$

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,432	138	9,570
	Yes	235	195	430
Total		9,667	333	10,000

- ▶ The false negative rate is now 41%
- ▶ The false positive rate has increased (from  $< 1\%$  to  $2\%$ )
  - ▶ So, we're paying a price for reducing the false negative rate (i.e. there's a trade-off)



Viewing the trade-off over different thresholds

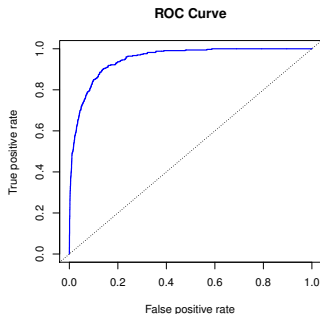


- ▶ — False negative rate (error for defaulting customers)
- ▶ ..... False positive rate (error for non-defaulting customers)
- ▶ — 0-1 loss or total error rate



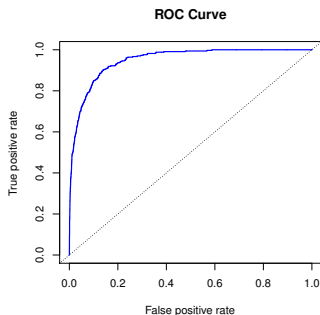
The *Receiver Operating Characteristic (ROC)* curve:

- Displays the performance for every threshold choice.





The *Receiver Operating Characteristic (ROC)* curve:

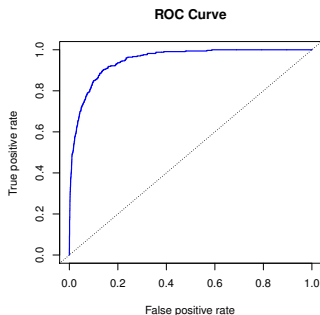


- ▶ Displays the performance for every threshold choice.
- ▶ The *Area Under the Curve (AUC)* summarizes the classifier performance, e.g.
  - ▶ The closer AUC is to 1, the better the performance.





The *Receiver Operating Characteristic (ROC)* curve:



- ▶ Displays the performance for every threshold choice.
- ▶ The *Area Under the Curve (AUC)* summarizes the classifier performance, e.g.
  - ▶ The closer AUC is to 1, the better the performance.
  - ▶  $AUC = 0.5$  is equivalent to a random classifier.
    - ▶ The minimum value (otherwise, you'd just flip the class predictions).
  - ▶  $AUC$  is equivalent to the Mann–Whitney U test.



- ▶ In general, we want to optimize for loss functions that are specific to our (applied) problem
  - ▶ e.g. When predicting credit default, we'll likely care more about the precision or recall
  - ▶ i.e. Natural loss function is not the cross-entropy or 0-1 loss
- ▶ Even if we use one method which minimizes a certain kind of training error, we can tune it to optimize our true loss function
  - ▶ e.g. Find the threshold that brings the recall above an acceptable level



**Recall:** Training our model on a dataset and evaluating on the same dataset leads to (overly) optimistic results

- ▶ We care about the error on  $P_0$ , not  $P_n$



**Recall:** Training our model on a dataset and evaluating on the same dataset leads to (overly) optimistic results

- ▶ We care about the error on  $P_0$ , not  $P_n$

**Additionally:** We have *hyper-parameters* that we have to tune, e.g.

- ▶ The number of neighbors  $k$  in  $k$ -nearest neighbors
- ▶ The number of variables in forward/backward step selection
- ▶ The order of a polynomial in polynomial regression



**Typically:** Datasets are divided into three parts

1. *Training set*: data you use to train your model parameters
2. *Development set*: data you use to tune your model hyper-parameters
3. *Test set*: data you use to evaluate your model after it's been trained

Keeping separate development/test sets prevents the model from over-fitting (providing overly optimistic prediction errors)



**Typically:** Datasets are divided into three parts

1. *Training set*: data you use to train your model parameters
2. *Development set*: data you use to tune your model hyper-parameters
3. *Test set*: data you use to evaluate your model after it's been trained

Keeping separate development/test sets prevents the model from over-fitting (providing overly optimistic prediction errors)

**n.b.** The terminology will vary across fields, e.g.

- ▶ Sometimes, people refer to the “*development*” set as the “*validation*” set
- ▶ The course textbook refers to the “*test*” set as the “*validation*” set



In practice, may not need a development set. Consequently, you'll

1. Split the data into two parts (i.e. a train and test set)
2. Train on the first part
3. Compute the error on the second part



Visualization of data splitting



In practice, may not need a development set. Consequently, you'll

1. Split the data into two parts (i.e. a train and test set)
2. Train on the first part
3. Compute the error on the second part



Visualization of data splitting

**n.b.** You'll want to split the data randomly

- Avoids possible correlation (e.g. between households)





## Comparing scenarios:

Assume  $n = 1000$ , and consider the three splits

1.  $n_{train} = 500$  and  $n_{test} = 500$
2.  $n_{train} = 50$  and  $n_{test} = 950$
3.  $n_{train} = 950$  and  $n_{test} = 50$

What happens to the model fit and prediction errors?



## Comparing scenarios:

Assume  $n = 1000$ , and consider the three splits

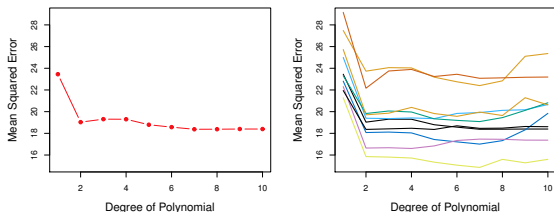
1.  $n_{train} = 500$  and  $n_{test} = 500$
2.  $n_{train} = 50$  and  $n_{test} = 950$
3.  $n_{train} = 950$  and  $n_{test} = 50$

What happens to the model fit and prediction errors?

- ▶ Scenario 2 has high model variance, but lower variance in estimates of prediction error
- ▶ Scenario 3 has low model variance, but higher variance in estimates of prediction error
- ▶ Scenario 1 provides a trade-off between the two



**Example:** Polynomial regression to estimate mpg from horsepower in the Auto data



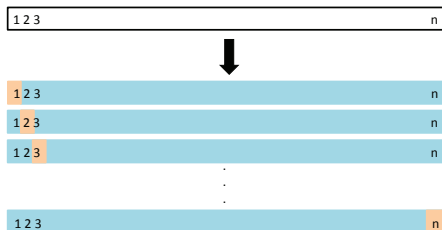
MSE under different samples for the test split

**Problem:** Every split yields a different estimate of the error



Allows us to use every observation as the test split

1. For  $i = 1, 2, \dots, n$ :
  - ▶ Train the model on every point except  $i$
  - ▶ Compute the test error on point  $i$
2. Average the test errors





Allows us to use every observation as the test split

1. For  $i = 1, 2, \dots, n$ :
  - ▶ Train the model on every point except  $i$
  - ▶ Compute the test error on point  $i$
2. Average the test errors

For *regression*:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i^{(-i)})^2 \quad (17)$$

For *classification*:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n \mathbb{I}[y_i \neq \hat{y}_i^{(-i)}] \quad (18)$$



Computing  $CV_{(n)}$  can be computationally expensive, since it involves fitting the model  $n$  times.

For linear regression, there is a shortcut:

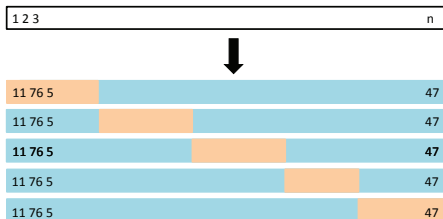
$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \hat{y}_i}{1 - h_{ii}} \right)^2 \quad (19)$$

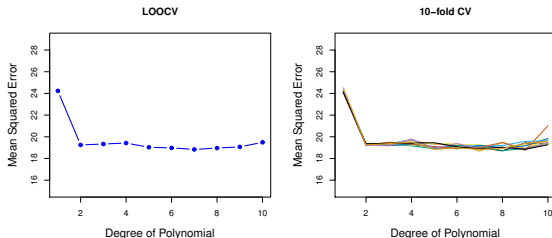
where  $h_{ii}$  is the leverage statistic (Chapter 3).



Rather than LOOCV, we can just divide the data into  $k$  splits (aka folds)

1. Divide the data into  $k$  splits (aka folds)
2. For  $i = 1, \dots, k$ :
  - a Train your model on all the data excluding the  $i^{\text{th}}$  fold
  - b Compute the prediction error on the  $i^{\text{th}}$  fold
3. Average the errors over the  $k$  splits



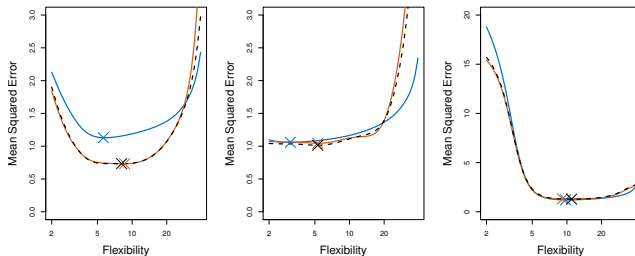


- ▶  $k$ -fold CV depends on the chosen split
- ▶ In  $k$ -fold CV, we train the model on less data than what is available.
  - ▶ Introduces bias into estimates of the test error
- ▶ In LOOCV, the fitted models are very correlated
  - ▶ Increases variance of the estimates of the test error



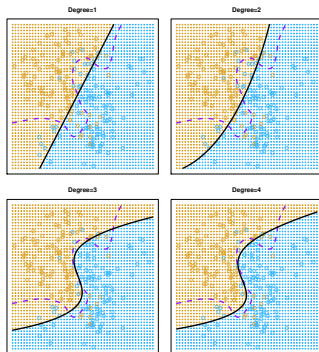


## Choosing an optimal model



- ▶ Despite the bias-variance tradeoff, the error estimates still (generally) tend to be pretty similar
- ▶ Choosing the model with the minimum cross validation error often leads to the method with minimum test error

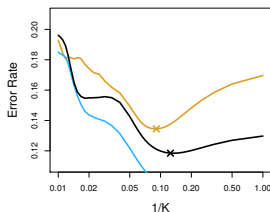
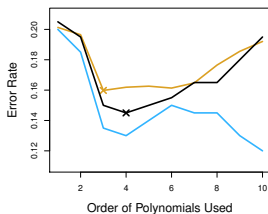
In classification, we can take the same approach, e.g.



- ▶ - - - Bayes boundary
- ▶ — Logistic regression with polynomial predictors of increasing degree.

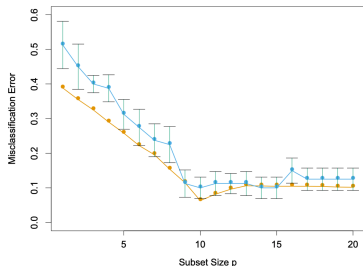


n.b. We don't know Bayes boundary in practice, but can choose the fit with the lowest error rate, e.g.





## Forward stepwise selection



Blue: 10-fold cross validation

Yellow: True test error

- ▶ A number of models with  $9 \leq p \leq 15$  have the same CV error.
- ▶ The vertical bars represent 1 standard error in the test error from the 10 folds.
- ▶ **Rule of thumb:** Choose the simplest model whose CV error is no more than one standard error above the model with the lowest CV error.



Suppose we want to classify 200 individuals according to whether they have cancer or not.

- ▶ We use logistic regression onto 1,000 measurements of gene expression

## **Our proposed strategy**

- ▶ Using all our data, select the 20 most significant genes using z-tests
- ▶ Estimate the test error of logistic regression with these 20 predictors via 10-fold cross validation



Suppose we want to classify 200 individuals according to whether they have cancer or not.

Let's simulate some data so that we know the true distribution  $P_0$

- ▶ Each gene expression (of the 1,000) is standard normal and independent of all others
- ▶ The response (cancer or not) is sampled from a coin flip — no correlation to any of the “genes”



Suppose we want to classify 200 individuals according to whether they have cancer or not.

Let's simulate some data so that we know the true distribution  $P_0$

- ▶ Each gene expression (of the 1,000) is standard normal and independent of all others
- ▶ The response (cancer or not) is sampled from a coin flip — no correlation to any of the “genes”

**Under these settings, the misclassification rate for any classification method using these predictors should be 50%**



Runing this simulation gives us CV error rate of 3%!

Why is this?

- ▶ Since we only have 200 individuals in total, among 1,000 variables, at least some will be correlated with the response
- ▶ Doing variable selection using **all** the data, means that the variables we select will have some correlation with the response in every subset or fold in the cross validation



# The **right** way to do cross validation



- ▶ Divide the data into 10 folds
- ▶ For  $i = 1, \dots, 10$ :
  - a Using every fold except  $i$ , perform **both** the variable selection and model fit with the selected variables
  - b Compute the prediction error on the  $i^{th}$  fold
- ▶ Average the errors over the 10 splits

The simulation produces an error estimate of close to 50%



- ▶ Divide the data into 10 folds
- ▶ For  $i = 1, \dots, 10$ :
  - a Using every fold except  $i$ , perform **both** the variable selection and model fit with the selected variables
  - b Compute the prediction error on the  $i^{th}$  fold
- ▶ Average the errors over the 10 splits

The simulation produces an error estimate of close to 50%

**Moral of the story:** Every aspect of the learning method that involves using the data (variable selection, for example) must be cross-validated



**Recall:** We're interested in estimating some function  $f_0$  within a space of probability functions.

- ▶ How do we define '*best*' within this space?



**Recall:** We're interested in estimating some function  $f_0$  within a space of probability functions.

- How do we define '*best*' within this space?

Define  $O = (X_1, \dots, X_p, Y) \sim P_0$

$$f_0 = \arg \min_{f \in \mathcal{F}} \int L(o, f) \partial P_0(o) \quad (20)$$

where  $L(O, f)$  is a bounded loss function with finite variance.



**Recall:** We're interested in estimating some function  $f_0$  within a space of probability functions.

- How do we define '*best*' within this space?

Define  $O = (X_1, \dots, X_p, Y) \sim P_0$

$$f_0 = \arg \min_{f \in \mathcal{F}} \int L(o, f) \partial P_0(o) \quad (20)$$

where  $L(O, f)$  is a bounded loss function with finite variance.

We define the distance between  $f$  and  $f_0$  as

$$d_0(f, f_0) \triangleq \mathbb{E}_{P_0}[L(O, f) - L(O, f_0)] \quad (21)$$

**The best estimator is the one that minimizes  $d_0(f, f_0)$**



Let  $\hat{f}_n^k = \hat{\Psi}_k(P_n) : k = 1, 2, \dots, K$  be the  $K$  estimators we're evaluating. Our cross-validation selector is

$$\hat{K}(P_n) \triangleq \arg \min_k \mathbb{E}_{B_n} \int L(o, \hat{\Psi}_k(P_{n,B_n}^0)) \partial P_{n,B_n}^1(o) \quad (22)$$



Let  $\hat{f}_n^k = \hat{\Psi}_k(P_n) : k = 1, 2, \dots, K$  be the  $K$  estimators we're evaluating. Our cross-validation selector is

$$\hat{K}(P_n) \triangleq \arg \min_k \mathbb{E}_{B_n} \int L(o, \hat{\Psi}_k(P_{n,B_n}^0)) \partial P_{n,B_n}^1(o) \quad (22)$$

Consider an *oracle selector*:

$$\tilde{K}(P_n) \triangleq \arg \min_k \mathbb{E}_{B_n} \int L(o, \hat{\Psi}_k(P_{n,B_n}^0)) \partial P_0(o) \quad (23)$$



Let  $\hat{f}_n^k = \hat{\Psi}_k(P_n) : k = 1, 2, \dots, K$  be the  $K$  estimators we're evaluating. Our cross-validation selector is

$$\hat{K}(P_n) \triangleq \arg \min_k \mathbb{E}_{B_n} \int L(o, \hat{\Psi}_k(P_{n,B_n}^0)) dP_{n,B_n}^1(o) \quad (22)$$

Consider an *oracle selector*:

$$\tilde{K}(P_n) \triangleq \arg \min_k \mathbb{E}_{B_n} \int L(o, \hat{\Psi}_k(P_{n,B_n}^0)) dP_0(o) \quad (23)$$

Then for any  $\delta > 0$ ,

$$\begin{aligned} \mathbb{E}_0 d_0(\hat{\Psi}_{\hat{K}(P_n)}(P_{n,B_n}^0), f_0) &\leq (1 + 2\delta) \mathbb{E}_0 d_0(\hat{\Psi}_{\tilde{K}(P_n)}(P_{n,B_n}^0), f_0) \\ &\quad + C(\delta) \frac{\log(K)}{n} \end{aligned} \quad (24)$$

where  $C(\delta)$  is a constant.





What this means:

- ▶ If none of the candidate learners converge at a parametric rate, the cross validated selector performs asymptotically as well (in the risk difference sense) as the oracle selector.



What this means:

- ▶ If none of the candidate learners converge at a parametric rate, the cross validated selector performs asymptotically as well (in the risk difference sense) as the oracle selector.
- ▶ If one of the candidate learners searches within a parametric model (that contains  $f_0$ ), and thus achieves a parametric rate of convergence, then the cross validated selector achieves the almost parametric rate of convergence  $\log n/n$ .



**Question:** Rather than just picking one estimator, can we combine them? e.g.

$$\bar{K}(P_n) \triangleq \alpha_1 \hat{\Psi}_1(P_{n,B_n}^0) + \cdots + \alpha_K \hat{\Psi}_K(P_{n,B_n}^0) : \sum_{k=1}^K \alpha_k = 1 \quad (25)$$

Each weighted average is a unique candidate algorithm in our '*augmented*' library. e.g.

- ▶ Taking the average across the estimators corresponds to  $\alpha_k = 1/K$ .



**Question:** Rather than just picking one estimator, can we combine them? e.g.

$$\bar{K}(P_n) \triangleq \alpha_1 \hat{\Psi}_1(P_{n,B_n}^0) + \cdots + \alpha_K \hat{\Psi}_K(P_{n,B_n}^0) : \sum_{k=1}^K \alpha_k = 1 \quad (25)$$

Each weighted average is a unique candidate algorithm in our '*augmented*' library. e.g.

- ▶ Taking the average across the estimators corresponds to  $\alpha_k = 1/K$ .

We could then apply the cross validated selector to this augmented library.

- ▶ Alternatively: could just estimate the  $\alpha_k$ 's directly.
- ▶ Could also make  $\alpha_k$ 's dependent on our inputs  $(X_1, \dots, X_p)$ .



- [1] ISL. Chapters 4-5.
- [2] ESL. Chapters 7, 8.8.
- [3] Super Learner. M.J. van der Laan, E.C. Polley, A.E. Hubbard (2007).