DSA/ISE 5103 Intelligent Data Analytics

Principles of Modeling

Charles Nicholson, Ph.D.
cnicholson@ou.edu

University of Oklahoma
Gallogly College of Engineering
School of Industrial and Systems Engineering

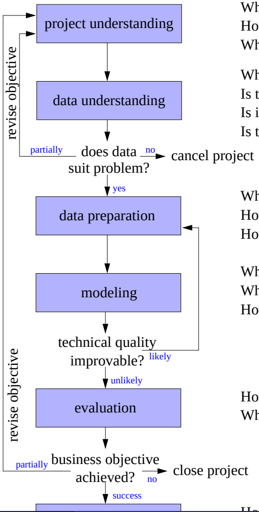
Outline

credits

Credits – some images / excerpts are taken from:

- the textbook “An Introduction to Statistical Learning with Applications in R” by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
- The *useR! 2013 Tutorial* by Max Kuhn: Predictive Modeling with R and the caret Package (one of the authors of the course text)

modeling



What exactly is the problem, the expected benefit?
How would a solution look like?
What is known about the domain?

What data do we have available?
Is the data relevant to the problem?
Is it valid? Does it reflect our expectations?
Is the data quality, quantity, recency sufficient?

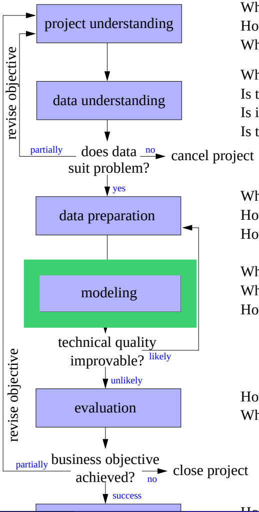
Which data should we concentrate on?
How is the data best transformed for modeling?
How may we increase the data quality?

What kind of model architecture suits the problem best?
What is the best technique/method to get the model?
How good does the model perform technically?

How good is the model in terms of project requirements?
What have we learned from the project?

How is the model best deployed?

modeling



What exactly is the problem, the expected benefit?
How would a solution look like?
What is known about the domain?

What data do we have available?
Is the data relevant to the problem?
Is it valid? Does it reflect our expectations?
Is the data quality, quantity, recency sufficient?

Which data should we concentrate on?
How is the data best transformed for modeling?
How may we increase the data quality?

What kind of model architecture suits the problem best?
What is the best technique/method to get the model?
How good does the model perform technically?

How good is the model in terms of project requirements?
What have we learned from the project?

tasks and methods

The *problem understanding* phase should help identify the analysis task.

tasks and methods

The *problem understanding* phase should help identify the analysis task.

- **Prediction**
  Tomorrow's stock price...
- **Classification**
  Identify numbers in a handwritten ZIP code from a digitized image
- **Clustering**
  Determine distinct customer groups
- **Dependence/Association Analysis**
  Discover interesting buying behaviors

tasks and methods

The *problem understanding* phase should help identify the analysis task.

- **Prediction**
  Tomorrow's stock price...
- **Classification**
  Identify numbers in a handwritten ZIP code from a digitized image
- **Clustering**
  Determine distinct customer groups
- **Dependence/Association Analysis**
  Discover interesting buying behaviors

tasks and methods

The *problem understanding* phase should help identify the analysis task.

- **Prediction**
  Tomorrow's stock price...
- **Classification**
  Identify numbers in a handwritten ZIP code from a digitized image
- **Clustering**
  Determine distinct customer groups
- **Dependence/Association Analysis**
  Discover interesting buying behaviors

tasks and methods

The *problem understanding* phase should help identify the analysis task.

- **Prediction**
  Tomorrow's stock price...
- **Classification**
  Identify numbers in a handwritten ZIP code from a digitized image
- **Clustering**
  Determine distinct customer groups
- **Dependence/Association Analysis**
  Discover interesting buying behaviors

tasks and methods

The *problem understanding* and *data understanding* phase should help narrow the scope of the available methods and model classes.

tasks and methods

The *problem understanding* and *data understanding* phase should help narrow the scope of the available methods and model classes.

The *data preparation* phase should work in concert with the chosen modeling method(s).

- **Regression methods**
  classification, prediction
- **Tree models**
  classification
- **k-nearest Neighbor**
  classification, prediction
- **Support vector machines**
  classification, prediction
- **Cluster Analysis**
  clustering
- **Association rule induction**
  association analysis

- **Regression methods**
  classification, prediction
- **Tree models**
  classification
- **k-nearest Neighbor**
  classification, prediction
- **Support vector machines**
  classification, prediction
- **Cluster Analysis**
  clustering
- **Association rule induction**
  association analysis

- **Regression methods**
  classification, prediction
- **Tree models**
  classification
- **k-nearest Neighbor**
  classification, prediction
- **Support vector machines**
  classification, prediction
- **Cluster Analysis**
  clustering
- **Association rule induction**
  association analysis

- **Regression methods**
  classification, prediction
- **Tree models**
  classification
- **k-nearest Neighbor**
  classification, prediction
- **Support vector machines**
  classification, prediction
- **Cluster Analysis**
  clustering
- **Association rule induction**
  association analysis

- **Regression methods**
  classification, prediction
- **Tree models**
  classification
- **k-nearest Neighbor**
  classification, prediction
- **Support vector machines**
  classification, prediction
- **Cluster Analysis**
  clustering
- **Association rule induction**
  association analysis

- **Regression methods**
  classification, prediction
- **Tree models**
  classification
- **k-nearest Neighbor**
  classification, prediction
- **Support vector machines**
  classification, prediction
- **Cluster Analysis**
  clustering
- **Association rule induction**
  association analysis

two types of statistical learning

Most statistical learning problems fall into one of two categories:

- Supervised learning: inputs *and* outputs are known

two types of statistical learning

Most statistical learning problems fall into one of two categories:

- Supervised learning: inputs *and* outputs are known
- Unsupervised learning: no outputs in the data

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
  - for accurately **predicting** the response of *other* observations
  - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
  - for accurately **predicting** the response of *other* observations
  - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

$$
\begin{array}{ccccc}
Y & X_1 & X_2 & \cdots & X_p \\
\hline
y_1 & x_{1,1} & x_{2,1} & \cdots & x_{p,1} \\
y_2 & x_{1,2} & x_{2,2} & \cdots & x_{p,2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
y_n & x_{1,n} & x_{2,n} & \ldots & x_{p,n} \\
\hline
\end{array}
$$

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
  - for accurately **predicting** the response of *other* observations
  - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
  - for accurately **predicting** the response of *other* observations
  - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
  - for accurately **predicting** the response of *other* observations
  - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
    - for accurately **predicting** the response of *other* observations
    - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

supervised learning

- Model is *trained* to identify/predict a known output data.
- For each of the n cases of predictors X_i, $i = 1, \ldots, p$ there is a response measurement y_i.
- Goal: *fit* a model that relates response to predictors
  - for accurately **predicting** the response of *other* observations
  - or understanding the relationship between the response and the predictors (**inference**).
- Examples: linear and logistic regression, decision trees, SVM
- Evaluation: based on error measures (e.g., MSE, misclassification rate, expected loss).

unsupervised learning

- Unsupervised learning describes the challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but *no associated response* y_i.
  - e.g., it is not possible to fit a linear regression model: there is no response variable to predict.
- We are in some sense working blind
  - the term *unsupervised* is used because we lack a response variable that can supervise (or teach or evaluate) our analysis.
- Examples: PCA, association mining, cluster analysis
- Error measures do not make sense

unsupervised learning

- Unsupervised learning describes the challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but *no associated response* y_i.
  - e.g., it is not possible to fit a linear regression model: there is no response variable to predict.
- We are in some sense working blind
  - the term *unsupervised* is used because we lack a response variable that can supervise (or teach or evaluate) our analysis.
- Examples: PCA, association mining, cluster analysis
- Error measures do not make sense

unsupervised learning

- Unsupervised learning describes the challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but *no associated response* y_i.
  - e.g., it is not possible to fit a linear regression model: there is no response variable to predict.
- We are in some sense working blind
  - the term *unsupervised* is used because we lack a response variable that can supervise (or teach or evaluate) our analysis.
- Examples: PCA, association mining, cluster analysis
- Error measures do not make sense

unsupervised learning

- Unsupervised learning describes the challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but *no associated response* y_i.
  - e.g., it is not possible to fit a linear regression model: there is no response variable to predict.
- We are in some sense working blind
  - the term *unsupervised* is used because we lack a response variable that can supervise (or teach or evaluate) our analysis.
- Examples: PCA, association mining, cluster analysis
- Error measures do not make sense

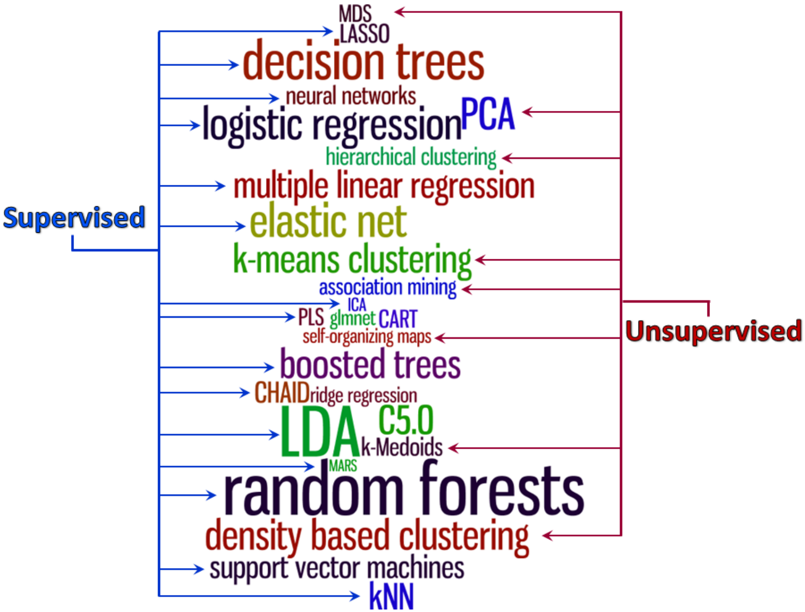
unsupervised learning

- Unsupervised learning describes the challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but *no associated response* y_i.
  - e.g., it is not possible to fit a linear regression model: there is no response variable to predict.
- We are in some sense working blind
  - the term *unsupervised* is used because we lack a response variable that can supervise (or teach or evaluate) our analysis.
- Examples: PCA, association mining, cluster analysis
- Error measures do not make sense

unsupervised learning

- Unsupervised learning describes the challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but *no associated response* y_i.
  - e.g., it is not possible to fit a linear regression model: there is no response variable to predict.
- We are in some sense working blind
  - the term *unsupervised* is used because we lack a response variable that can supervise (or teach or evaluate) our analysis.
- Examples: PCA, association mining, cluster analysis
- Error measures do not make sense


MDS
LASSO
decision trees
neural networks
logistic regression
PCA
hierarchical clustering
multiple linear regression
elastic net
k-means clustering
association mining
ICA
PLS
glmnet
CART
self-organizing maps
boosted trees
CHAID
ridge regression
LDA
C5.0
k-Medoids
MARS
random forests
density based clustering
support vector machines
kNN



Supervised
Unsupervised
MDS
LASSO
decision trees
neural networks
logistic regression
PCA
hierarchical clustering
multiple linear regression
elastic net
k-means clustering
association mining
ICA
PLS
glmnet
CART
self-organizing maps
boosted trees
CHAID
ridge regression
LDA
C5.0
k-Medoids
MARS
random forests
density based clustering
support vector machines
kNN

a few notes

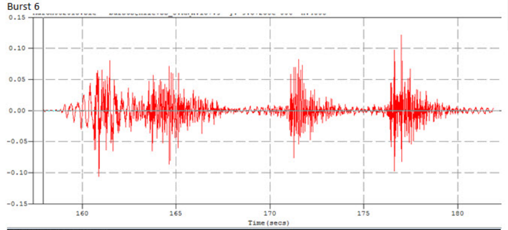
- unlabeled data is “cheap”
- labeled data can be hard to get
    - human annotation is boring
    - labels may require experts
    - labels may require special devices
    - your graduate student may quit

a few notes

- unlabeled data is “cheap”
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student may quit

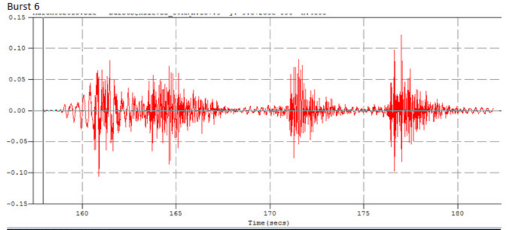
a few notes

- unlabeled data is “cheap”
- labeled data can be hard to get
    - human annotation is boring
    - labels may require experts
    - labels may require special devices
    - your graduate student may quit



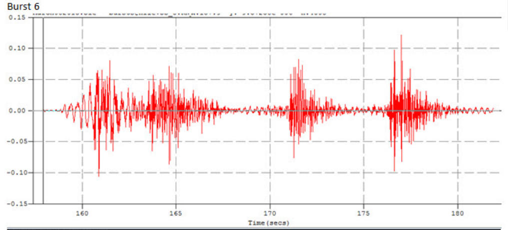
a few notes

- unlabeled data is “cheap”
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student may quit



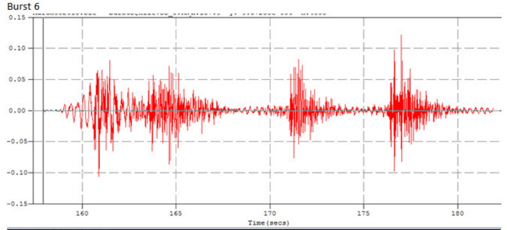
a few notes

- unlabeled data is “cheap”
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student may quit



a few notes

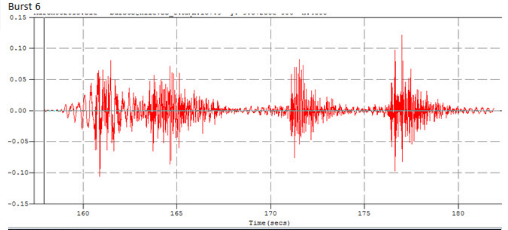
- unlabeled data is “cheap”
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student may quit



a few notes

- unlabeled data is “cheap”
- labeled data can be hard to get
    - human annotation is boring
    - labels may require experts
    - labels may require special devices
    - your graduate student may quit

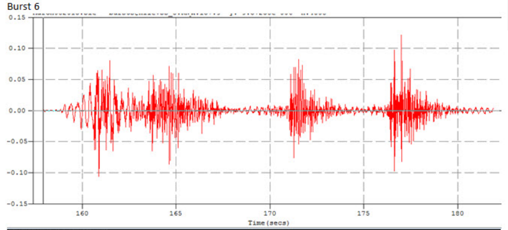
Note: we will spend the majority of the next several lectures focuses on supervised learning



a few notes

- unlabeled data is “cheap”
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student may quit

Note: we will spend the majority of the next several lectures focuses on supervised learning



Semi-supervised learning is a third option which attempts to blend supervised and unsupervised learning. The above motivation is provided by Xiaojin Zhu (2007) Semi-Supervised Learning Tutorial. ICML.
The truck-bridge sensor problem, well, that one is mine...

what and why?

In supervised learning, we believe

$$Y = f(X) + \epsilon$$

what and why?

In supervised learning, we believe

$$Y = f(X) + \epsilon$$

and we are trying to learn f to predict Y:

$$\hat{Y} = \hat{f}(X)$$

what and why?

In supervised learning, we believe

$$Y = f(X) + \epsilon$$

and we are trying to learn f to predict Y:

$$\hat{Y} = \hat{f}(X)$$

where X is our predictor inputs, $\hat{f}$ is our estimate of f, and $\hat{Y}$ is the resulting prediction for Y.

what and why?

The two possible motivations:

- Inference
- Prediction

inference

This study examined the relationship of age to sexual recidivism using data from 10 follow-up studies of adult male sexual offenders (combined sample of 4,673). Rapists were younger than child molesters, and the recidivism risk of rapists steadily decreased with age. In contrast, extrafamilial child molesters showed relatively little reduction in recidivism risk until after the age of 50. The recidivism rate of intrafamilial child molesters was generally low (less than 10%), except for the intrafamilial offenders in the 18- to 24-year-old age group, whose recidivism risk was comparable to that of rapists and extrafamilial child molesters. The results are discussed in terms of developmental changes in sexual drive, self-control, and opportunities to offend.

Recidivism and Age

Follow-Up Data From 4,673 Sexual Offenders

R. KARL HANSON
Department of the Solicitor General of Canada

The public is justifiably concerned about the risk posed by sexual offenders. Although the observed sexual recidivism rates are only 10% to 15% after 5 years (Hanson & Bussière, 1998), the rates continue to increase gradually with extended follow-up periods (Hanson, Steffy, & Gauthier, 1993a). Do sexual offenders remain at risk throughout their lives, or is there some age

inference

TABLE 2: The Relationship Between Age (years) and Sexual Recidivism (1 = yes, 0 = no)

			Logistic Regression Coefficients					
Sample	*Sample Size*	*Step*	*Intercept*		*Linear*		*Curvilinear*	
Rapists	1,133	1	–0.334	(0.319)	–0.040	(0.010)	—	—
		2	–0.585	(0.995)	–0.024	(0.060)	0.00023	(0.00088)
Extrafamilial child molesters	1,411	1	–0.411	(0.232)	–0.028	(0.006)	—	—
		2	–2.344	(0.778)	0.082	(0.043)	–0.00144	(0.00056)
Incest offenders	1,207	1	–0.069	(0.448)	–0.064	(0.013)	—	—
		2	1.359	(1.154)	–0.144	(0.061)	0.00108	(0.00079)
Total	4,673	1	–0.324	(0.140)	–0.035	(0.004)	—	—
		2	–0.489	(0.410)	–0.026	(0.023)	0.00013	(0.00030)

NOTE: Standard deviations in parentheses.
$*p < .05$. $**p < .01$. $***p < .001$.

what and why?

The two possible motivations:

- Inference
- Prediction

what and why?

The two possible motivations:

- Inference
- Prediction

Many real-world applications will require a combination or these two approaches.

types of error

“All models are wrong; some are useful.”

— George E. P. Box

types of error

> **"All models are wrong; some are useful."**
>
> — George E. P. Box

There are multiple types of *error* in supervised learning:

- experimental error
- sample error
- model error
- train vs. test error

types of error

"All models are wrong; some are useful."

— George E. P. Box

There are multiple types of *error* in supervised learning:

- experimental error
- sample error
- model error
- train vs. test error

types of error

> **“All models are wrong; some are useful.”**
>
> — George E. P. Box

There are multiple types of *error* in supervised learning:

- experimental error
- sample error
- model error
- train vs. test error

types of error

“All models are wrong; some are useful.”

— George E. P. Box

There are multiple types of *error* in supervised learning:

- experimental error
- sample error
- model error
- train vs. test error

experimental error

experimental error

- The **pure error** or **experimental error** is inherent in the data and due to noise, random variations, imprecise measurements or the influence of hidden variables that cannot be observed.

experimental error

- The **pure error** or **experimental error** is inherent in the data and due to noise, random variations, imprecise measurements or the influence of hidden variables that cannot be observed.
- Also called **intrinsic error** or **irreducible error**.

experimental error

- The **pure error** or **experimental error** is inherent in the data and due to noise, random variations, imprecise measurements or the influence of hidden variables that cannot be observed.
- Also called **intrinsic error** or **irreducible error**.
- It is impossible to overcome this error by the choice of a suitable model.

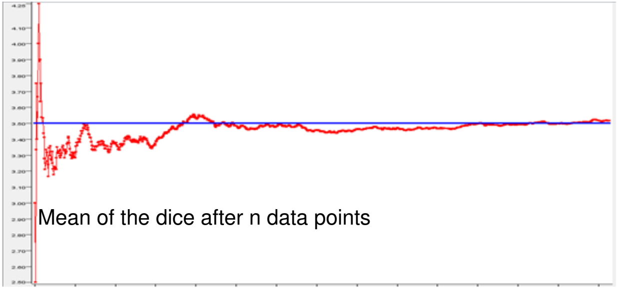
sample error

- Data is not a good representation of the underlying data
- Smaller the sample, smaller the probability of a good model

sample error

- Data is not a good representation of the underlying data
- Smaller the sample, smaller the probability of a good model

e.g., throw 6-sided dice and compute mean of pips



model error

There are different models for the data:

- simpler model $\Longrightarrow$ bigger error

model error

There are different models for the data:

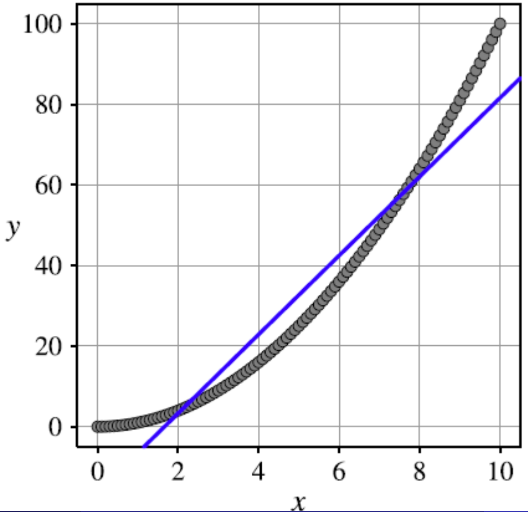
- simpler model $\Longrightarrow$ bigger error
- more complex model $\Longrightarrow$ overfitting and larger error on new data

model error

There are different models for the data:

- simpler model $\Longrightarrow$ bigger error
- more complex model $\Longrightarrow$ overfitting and larger error on new data
- type of model $\Longrightarrow$ different “fit” to data

model error



test-train error

- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

test-train error

- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

test-train error

- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

test-train error

- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

test-train error

- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

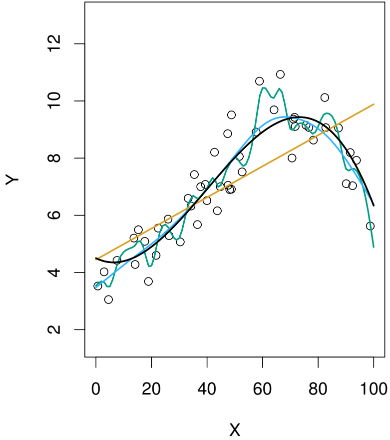
test-train error

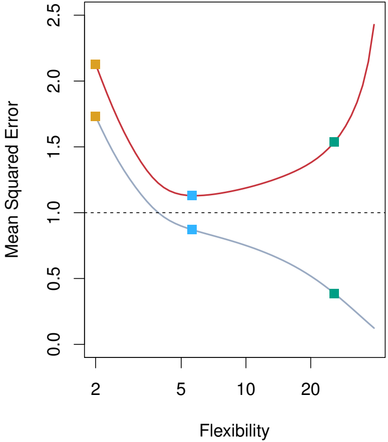
- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

test-train error

- training minimizes error on the *training* data
- model performance on different or new data may be different
- to estimate this, we evaluate models on *test* data (data is not used in training the model)
- the resulting error is known as the *test error*
- no guarantee that the model with the smallest training error will have the smallest test error
- usually, more flexible/complex models have less training error
- test error may be higher for more flexible/complex models

example: test vs. train





Black: Truth
Orange: Linear Estimate
Blue: smoothing spline
Green: smoothing spline (more flexible)

RED: Test MSE
Grey: Training MSE
Dashed: Minimum possible test MSE (irreducible)

bias and variance tradeoff

bias and variance tradeoff

- *Bias* refers to the difference between the expected prediction of our model and the correct value

http://scott.fortmann-roe.com/docs/BiasVariance.html

bias and variance tradeoff

- *Bias* refers to the difference between the expected prediction of our model and the correct value

  more flexible/complex models tend to have less bias

http://scott.fortmann-roe.com/docs/BiasVariance.html

bias and variance tradeoff

- *Bias* refers to the difference between the expected prediction of our model and the correct value

    more flexible/complex models tend to have less bias

http://scott.fortmann-roe.com/docs/BiasVariance.html

bias and variance tradeoff

- *Bias* refers to the difference between the expected prediction of our model and the correct value

    more flexible/complex models tend to have less bias

- *Variance* refers to how different your predictions would be if you had different training data

http://scott.fortmann-roe.com/docs/BiasVariance.html

bias and variance tradeoff

- *Bias* refers to the difference between the expected prediction of our model and the correct value

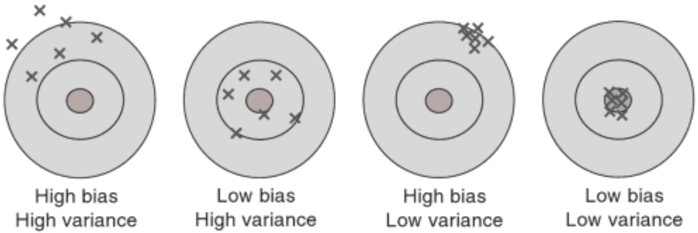
  more flexible/complex models tend to have less bias

- *Variance* refers to how different your predictions would be if you had different training data

  more flexible models tend to have more variance

http://scott.fortmann-roe.com/docs/BiasVariance.html

bias and variance



bias and variance decomposition

Assume the true model is: $Y = f(X) + \epsilon$ and we develop the model $\hat{f}(X)$ to predict Y. At the point $X = x$, the test error is:

$$\text{Err}(x) = E\left(Y - \hat{f}(x)\right)^2$$

bias and variance decomposition

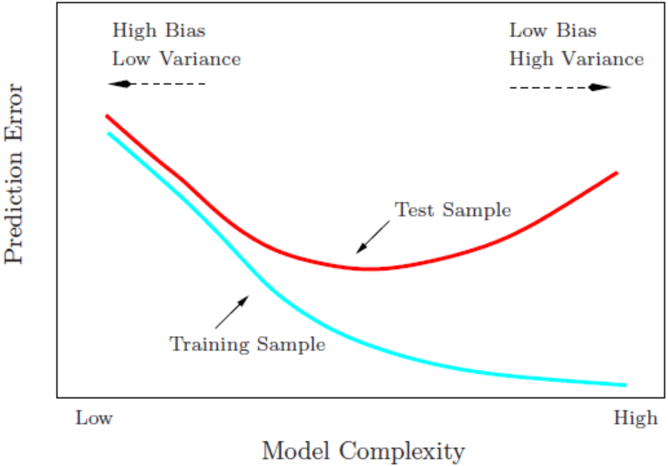
Assume the true model is: $Y = f(X) + \epsilon$ and we develop the model $\hat{f}(X)$ to predict Y. At the point $X = x$, the test error is:

$$\text{Err}(x) = E\left(Y - \hat{f}(x)\right)^2$$

which can be decomposed as,

$$\begin{aligned}\text{Err}(x) &= \left(E\left[\hat{f}(x)\right] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E\left[\hat{f}(x)\right]\right)^2\right] + \sigma_e^2 \\ &= \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}\end{aligned}$$

a fundamental picture



modeling steps

After

1. selecting a model class

the common modeling steps in *supervised learning* are:

modeling steps

After

1. selecting a model class

the common modeling steps in *supervised learning* are:

2. estimate model parameters ("training the model")
3. determine technical assessment mechanism
4. determine tuning parameters (parameters which cannot be estimated from data)
5. assess "generalizable" technical performance

modeling steps

After

1. selecting a model class

the common modeling steps in *supervised learning* are:

2. estimate model parameters (“training the model”)
3. determine technical assessment mechanism
4. determine tuning parameters (parameters which cannot be estimated from data)
5. assess “generalizable” technical performance

modeling steps

After

1. selecting a model class

the common modeling steps in *supervised learning* are:

2. estimate model parameters ("training the model")
3. determine technical assessment mechanism
4. determine tuning parameters (parameters which cannot be estimated from data)
5. assess "generalizable" technical performance

modeling steps

After

1. selecting a model class

the common modeling steps in *supervised learning* are:

2. estimate model parameters (“training the model”)
3. determine technical assessment mechanism
4. determine tuning parameters (parameters which cannot be estimated from data)
5. assess “generalizable” technical performance

fitting function and assessment function

There are two functions necessary to build and assess models; sometimes they are the same; sometimes they are different.

- “fitting” objective function
- assessment function

fitting function and assessment function

There are two functions necessary to build and assess models; sometimes they are the same; sometimes they are different.

- "fitting" objective function
- assessment function

In either case, we need to:

- define function $g : \mathcal{M} \rightarrow \mathbb{R}$
- which, evaluates the quality of the model
- in order to fit or detect the "best" model

Example

Simple linear regression model $\mathcal{M} : \hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

Example

Simple linear regression model $\mathcal{M} : \hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

Mean squared error:

$$g(\hat{\beta}_0, \hat{\beta}_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \mathcal{M}(\hat{\beta}_0, \hat{\beta}_1) \right)^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \left(\hat{\beta}_0 + \hat{\beta}_1 x_i \right) \right)^2$$

prediction and classification assessment

Assessment measures for:

- prediction (e.g., regression-based)
- classification (we will discuss later)

regression assessment

regression assessment

- **MSE** (mean-squared error)

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

regression assessment

- **MSE** (mean-squared error)

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

- **RMSE** (root mean squared error)

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

regression assessment

regression assessment

- **MAE** (mean-absolute error)

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

regression assessment

- **MAE** (mean-absolute error)

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **MAPE** (mean-absolute percentage error)

$$\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

- **R^2** is very popular statistical measure of how close the data are to the fitted regression line

- **R^2** is very popular statistical measure of how close the data are to the fitted regression line
  - it is the percentage of the response variable variation that is explained by a linear model, i.e. explained variation / total variation

- **R^2** is very popular statistical measure of how close the data are to the fitted regression line
  - it is the percentage of the response variable variation that is explained by a linear model, i.e. explained variation / total variation
  - $R^2 \in [0, 1]$

- **R^2** is very popular statistical measure of how close the data are to the fitted regression line
  - it is the percentage of the response variable variation that is explained by a linear model, i.e. explained variation / total variation
  - $R^2 \in [0, 1]$
  - always decreases a you add more variables $\rightarrow$ does not adjust for model complexity

- **R^2** is very popular statistical measure of how close the data are to the fitted regression line
  - it is the percentage of the response variable variation that is explained by a linear model, i.e. explained variation / total variation
  - $R^2 \in [0, 1]$
  - always decreases a you add more variables $\rightarrow$ does not adjust for model complexity
- **Adjusted R^2** accounts for model complexity:

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(n - 1)}{N - p - 1}$$

- **R^2** is very popular statistical measure of how close the data are to the fitted regression line
  - it is the percentage of the response variable variation that is explained by a linear model, i.e. explained variation / total variation
  - $R^2 \in [0, 1]$
  - always decreases a you add more variables $\rightarrow$ does not adjust for model complexity
- **Adjusted R^2** accounts for model complexity:

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(n - 1)}{N - p - 1}$$

where n is number of observations; p is number of predictors

regression assessment

regression assessment

- **AIC** (Akaike information criterion): $2k - 2\ln(L)$

- L is the "log likelihood"; k is number of estimated parameters: $p + 1$
- lower scores are better
- penalize model complexity

regression assessment

- **AIC** (Akaike information criterion): $2k - 2\ln(L)$
    - AIC tends to choose more complex models as $n \to \infty$

- L is the "log likelihood"; k is number of estimated parameters: $p + 1$
- lower scores are better
- penalize model complexity

regression assessment

- **AIC** (Akaike information criterion): $2k - 2\ln(L)$
  - AIC tends to choose more complex models as $n \to \infty$
- **BIC** (Bayesian information criterion): $k\ln(n) - 2\ln(L)$

- L is the "log likelihood"; k is number of estimated parameters: $p + 1$
- lower scores are better
- penalize model complexity

regression assessment

- **AIC** (Akaike information criterion): $2k - 2\ln(L)$
  - AIC tends to choose more complex models as $n \to \infty$
- **BIC** (Bayesian information criterion): $k\ln(n) - 2\ln(L)$
  - BIC is asymptotically consistent as a selection criterion

- L is the “log likelihood”; k is number of estimated parameters: $p + 1$
- lower scores are better
- penalize model complexity

regression assessment

- **AIC** (Akaike information criterion): $2k - 2\ln(L)$
  - AIC tends to choose more complex models as $n \to \infty$
- **BIC** (Bayesian information criterion): $k\ln(n) - 2\ln(L)$
  - BIC is asymptotically consistent as a selection criterion
  - For small or moderate samples, BIC often chooses models that are too simple, because of its heavy penalty on complexity
- L is the “log likelihood”; k is number of estimated parameters: $p + 1$
- lower scores are better
- penalize model complexity

likelihood and log likelihood

A quick digression on "log likelihood"

If

$$x_i \sim F(\Theta), i = 1, \ldots, n$$

then the likelihood function is

$$L\left(\{x_i\}_{i=1}^n, \Theta\right) = \prod_{i=1}^n F\left(x_i; \Theta\right)$$

The likelihood function L can be maximized w.r.t. model parameters Θ

likelihood and log likelihood

An important trick! $\rightarrow$ We usually maximize the log of the likelihood function instead of the likelihood directly:

$$\begin{aligned}\log\left(L\left(\{x_i\}_{i=1}^{n}, \Theta\right)\right) &= \log\left(\prod_{i=1}^{n} F\left(x_i;\Theta\right)\right) \\ &= \sum_{i=1}^{n} \log\left(F\left(x_i;\Theta\right)\right)\end{aligned}$$

log likelihood for normal regression

What does L look like for a normal error simple linear regression?

log likelihood for normal regression

What does L look like for a normal error simple linear regression?

$$y_i = \beta_0 + \beta_1 + \epsilon_i$$

- y_i is the value of the response in the i^{th} observation
- β_0 and β_1 are regression coefficients
- x_i is a known constant
- $\epsilon_i \sim i.i.d. N(0, \sigma^2)$
- $i = 1, \ldots, n$

log likelihood for normal regression

What does L look like for a normal error simple linear regression?

$$\begin{aligned} L\left(\beta_0, \beta_1, \sigma^2\right) &= \prod_{i=1}^{n} \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2\sigma^2}(y_i-\beta_0-\beta_1 x_i)^2} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n} r_i^2} \end{aligned}$$

where $r_i = y_i - \beta_0 - \beta_1 x_i$

$$
\begin{aligned}
\log L\left(\beta_0, \beta_1, \sigma^2\right) &= \log\left(\frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^n r_i^2}\right) \\
&= \log 1 - \frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma^2 - \frac{1}{2}\sum_{i=1}^{n}\frac{r_i^2}{\sigma^2} \\
&= -\frac{n}{2}\left(\log 2\pi + \log\sigma^2 + 1\right) \\
&= -\frac{n}{2}\left(\log 2\pi + \log\frac{\sum_{i=1}^n r_i^2}{n} + 1\right) \\
&= -\frac{n}{2}\left(\log 2\pi + \log\sum_{i=1}^{n} r_i^2 - \log n + 1\right)
\end{aligned}
$$

log likelihood for normal regression

Example with the `mtcars` data set (n=32):

```
> fit<-lm(data=mtcars, mpg~disp+wt)
```

log likelihood for normal regression

Example with the `mtcars` data set (n=32):

```
> fit<-lm(data=mtcars, mpg~disp+wt)

> sum(fit$residuals^2)
[1] 246.6825
```

log likelihood for normal regression

Example with the mtcars data set (n=32):

```
> fit<-lm(data=mtcars, mpg~disp+wt)

> sum(fit$residuals^2)
[1] 246.6825

> -16*( log(2*pi) + log(246.6825) - log(32) + 1)
[1] -78.08389
```

log likelihood for normal regression

Example with the `mtcars` data set (n=32):

```
> fit<-lm(data=mtcars, mpg~disp+wt)

> sum(fit$residuals^2)
[1] 246.6825

> -16*( log(2*pi) + log(246.6825) - log(32) + 1)
[1] -78.08389
```

And using the `logLik` function in R:

```
> logLik(fit)
'log Lik.' -78.08389 (df=4)
```

log likelihood for normal regression

And finally,

$$\begin{aligned}
\textbf{AIC}: \quad & 2k - 2\ln(L) & = 2(4) - 2(-78.08389) = 164.1678 \\
\textbf{BIC}: \quad & k\ln(n) - 2\ln(L) & = 4\ln(32) - 2(-78.08389) = 170.0307
\end{aligned}$$

log likelihood for normal regression

And finally,

AIC: $2k - 2\ln(L) = 2(4) - 2(-78.08389) = 164.1678$

BIC: $k\ln(n) - 2\ln(L) = 4\ln(32) - 2(-78.08389) = 170.0307$

Commands available in R:

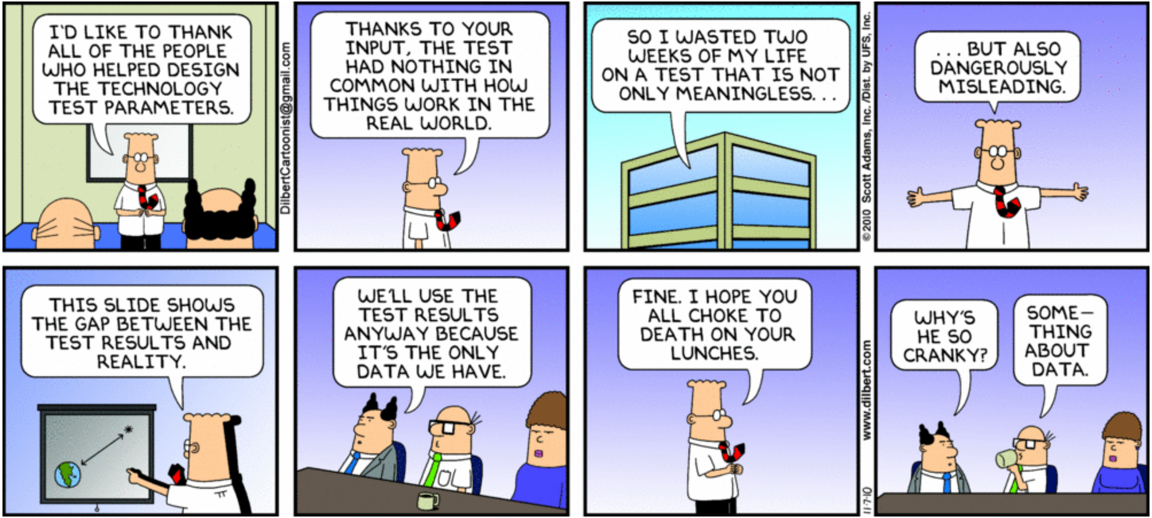
```
> AIC(fit)
[1] 164.1678

> BIC(fit)
[1] 170.0307
```

regression assessment

Common assessment for predictive (regression) models:

- R^2, R^2_{adjusted}
- MSE, RMSE, MAE, MAPE
- AIC, BIC

There are many more tools available to help you measure, diagnose, and improve regression models. We discuss these in detail in another lecture.


I'D LIKE TO THANK ALL OF THE PEOPLE WHO HELPED DESIGN THE TECHNOLOGY TEST PARAMETERS.
DilbertCartoonist@gmail.com
THANKS TO YOUR INPUT, THE TEST HAD NOTHING IN COMMON WITH HOW THINGS WORK IN THE REAL WORLD.
SO I WASTED TWO WEEKS OF MY LIFE ON A TEST THAT IS NOT ONLY MEANINGLESS. . .
© 2010 Scott Adams, Inc. /Dist. by UFS, Inc.
. . . BUT ALSO DANGEROUSLY MISLEADING.
THIS SLIDE SHOWS THE GAP BETWEEN THE TEST RESULTS AND REALITY.
WE'LL USE THE TEST RESULTS ANYWAY BECAUSE IT'S THE ONLY DATA WE HAVE.
FINE. I HOPE YOU ALL CHOKE TO DEATH ON YOUR LUNCHES.
www.dilbert.com
11-7-10
WHY'S HE SO CRANKY?
SOME-THING ABOUT DATA.

overfitting a.k.a. overtraining

Overfitting

Fitting the **noise** rather than fitting the *underlying relationship*.

overfitting a.k.a. overtraining

Overfitting

Fitting the **noise** rather than fitting the *underlying relationship*.

When we fit the model, it cannot tell which is the “signal” and which is “noise”. **So it fits both.** If the model is very flexible it can model the irreducible error well.

overfitting a.k.a. overtraining

Overfitting

Fitting the **noise** rather than fitting the *underlying relationship*.

When we fit the model, it cannot tell which is the “signal” and which is “noise”. **So it fits both.** If the model is very flexible it can model the irreducible error well. **This is a disaster!**

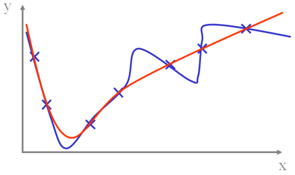
overfitting a.k.a. overtraining

Overfitting

Fitting the **noise** rather than fitting the *underlying relationship*.

When we fit the model, it cannot tell which is the “signal” and which is “noise”. **So it fits both.** If the model is very flexible it can model the irreducible error well. **This is a disaster!**

Typical indicator for overfitting: “Perfect fit”



model validation

The error for unseen data will most probably always be bigger than for the data used for training.

model validation

The error for unseen data will most probably always be bigger than for the data used for training.

This leads to a couple of questions:

model validation

The error for unseen data will most probably always be bigger than for the data used for training.

This leads to a couple of questions:

- **How do we estimate model performance on new data?**
- Some models have specific *tuning parameters* (a.k.a. hyperparameters) to control overfitting, e.g.,
  - the penalty associated with the number of predictors in LASSO
  - the number if splits in a tree model
- **How do we determine the best hyperparameters?**

model validation

The error for unseen data will most probably always be bigger than for the data used for training.

This leads to a couple of questions:

- **How do we estimate model performance on new data?**
- Some models have specific *tuning parameters* (a.k.a. hyperparameters) to control overfitting, e.g.,
  - the penalty associated with the number of predictors in LASSO
  - the number if splits in a tree model
- **How do we determine the best hyperparameters?**

model validation

The error for unseen data will most probably always be bigger than for the data used for training.

This leads to a couple of questions:

- **How do we estimate model performance on new data?**
- Some models have specific *tuning parameters* (a.k.a. hyperparameters) to control overfitting, e.g.,
  - the penalty associated with the number of predictors in LASSO
  - the number if splits in a tree model
- **How do we determine the best hyperparameters?**

model validation

The error for unseen data will most probably always be bigger than for the data used for training.

This leads to a couple of questions:

- **How do we estimate model performance on new data?**
- Some models have specific *tuning parameters* (a.k.a. hyperparameters) to control overfitting, e.g.,
  - the penalty associated with the number of predictors in LASSO
  - the number if splits in a tree model
- **How do we determine the best hyperparameters?**

model validation

The error for unseen data will most probably always be bigger than for the data used for training.

This leads to a couple of questions:

- **How do we estimate model performance on new data?**
- Some models have specific *tuning parameters* (a.k.a. hyperparameters) to control overfitting, e.g.,
  - the penalty associated with the number of predictors in LASSO
  - the number if splits in a tree model
- **How do we determine the best hyperparameters?**

Let's look at an example of *overfitting* and *underfitting* using k-nearest neighbors as a classifier.

Let's look at an example of *overfitting* and *underfitting* using k-nearest neighbors as a classifier.

But first... *how does kNN work as a classifier?*

Let's look at an example of *overfitting* and *underfitting* using k-nearest neighbors as a classifier.

But first... *how does kNN work as a classifier?*

Consider a simple example problem – two predictors X_1 and X_2 and a binary outcome Y with outcomes "blue" and "orange"

Let's look at an example of *overfitting* and *underfitting* using k-nearest neighbors as a classifier.

But first... *how does kNN work as a classifier?*

Consider a simple example problem – two predictors X_1 and X_2 and a binary outcome Y with outcomes "blue" and "orange"

Questions:

1. How would a kNN be used to classify data?
2. What is the "tuning parameter" to control model complexity?

lazy and eager leaners

kNN is different than most of the classifiers that we will work with. It is known as a lazy learner; most of the methods we deal with are eager learners.

lazy and eager leaners

kNN is different than most of the classifiers that we will work with. It is known as a lazy learner; most of the methods we deal with are eager learners.

Eager learning: construct general, explicit description of target function based on training samples – this is used for predictions/classification of new data

lazy and eager leaners

kNN is different than most of the classifiers that we will work with. It is known as a lazy learner; most of the methods we deal with are eager learners.

Eager learning: construct general, explicit description of target function based on training samples – this is used for predictions/classification of new data

Lazy learning: store the training data – generalizing beyond these data is deferred until an explicit request is made (e.g. by new data)

classifying using kNN

Given a positive integer k and a *test* observation x_0, the kNN classifier identifies the k points in the training data that are closest to x_0, represented by $\mathcal{N}_0$.

It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$P(Y = j | X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

classifying using kNN

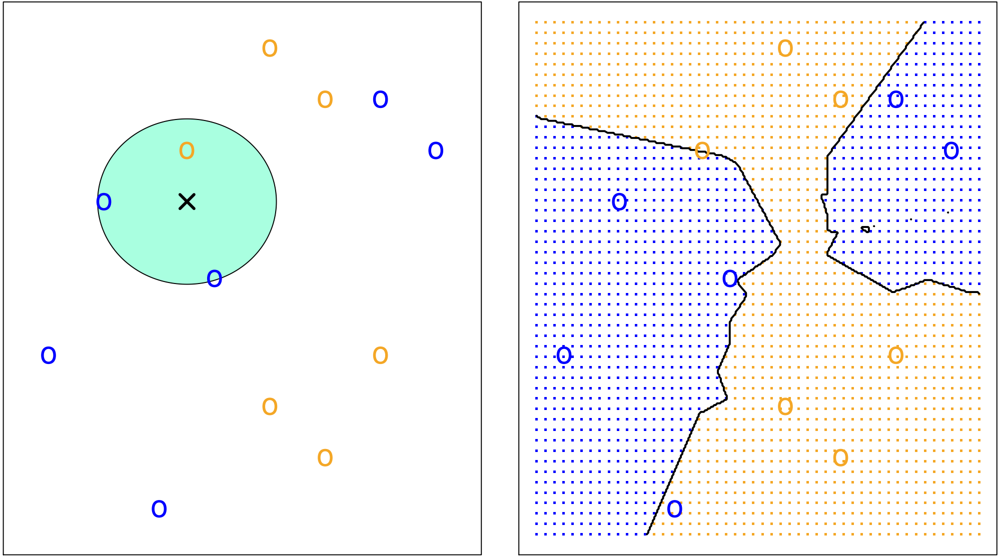
Given a positive integer k and a *test* observation x_0, the kNN classifier identifies the k points in the training data that are closest to x_0, represented by $\mathcal{N}_0$.

It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

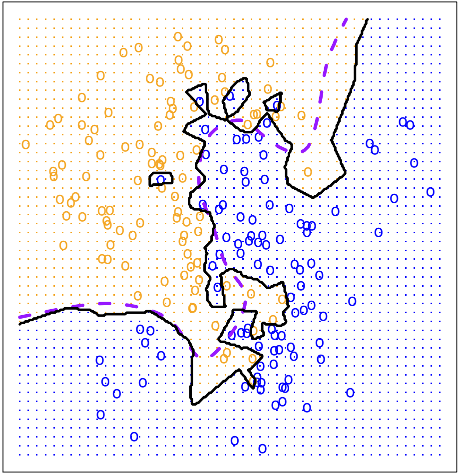
$$P(Y = j | X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

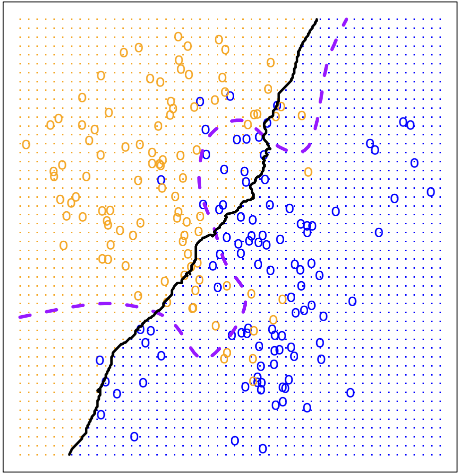
The tuning parameter is the value for k.

classification example: KNN k=3

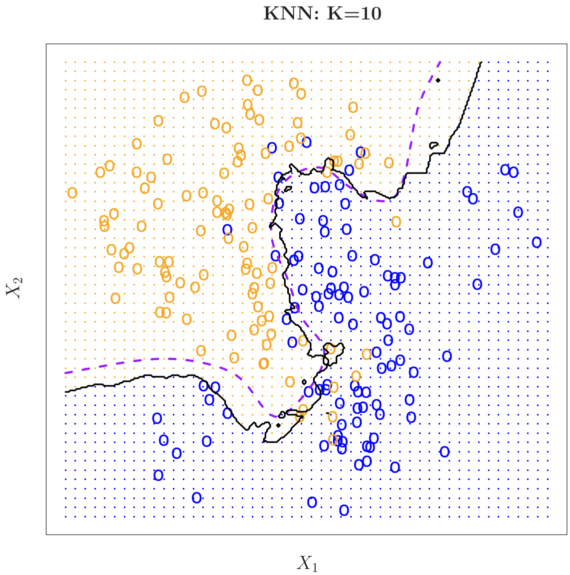


classification example: KNN

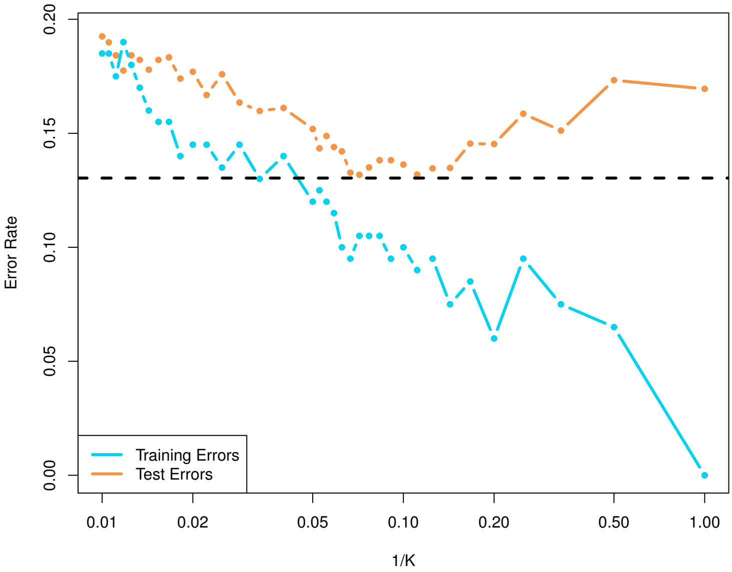




classification example: KNN k = 10



classification example: kNN – train vs. test



data strategies for testing

There are multiple strategies useful in:

- model selection (i.e., determine tuning parameters)
- estimate error rates on new data

data strategies for testing

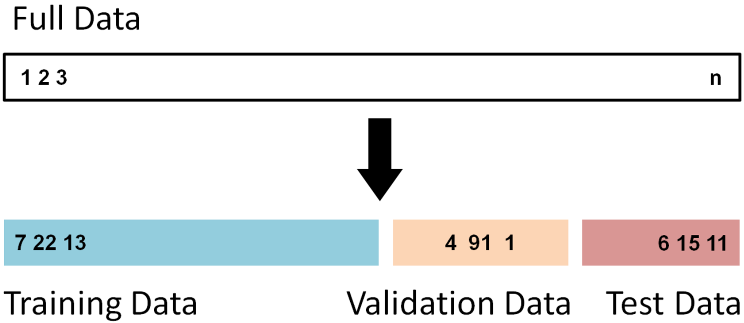
There are multiple strategies useful in:

- model selection (i.e., determine tuning parameters)
- estimate error rates on new data

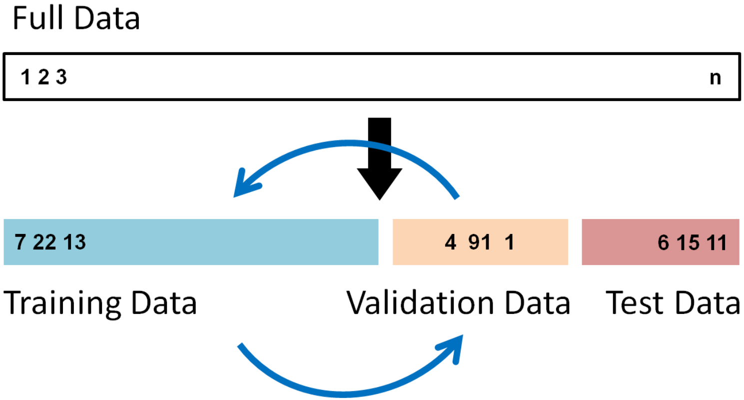
They are two basic approaches:

- **Holdout validation**
- **Resampling**
  - cross-validation (and nested cross-validation)
  - bootstrap sampling

data strategies: holdout validation



data strategies: holdout validation



data strategies: holdout validation

Split data into different subsets:

data strategies: holdout validation

Split data into different subsets:

- **Training set**: data used for learning; to fit the model

data strategies: holdout validation

Split data into different subsets:

- **Training set**: data used for learning; to fit the model
- **Validation set**: data used to tune the parameters; select the model

data strategies: holdout validation

Split data into different subsets:

- **Training set**: data used for learning; to fit the model
- **Validation set**: data used to tune the parameters; select the model
- **Test set**: used to assess the generalization error for the final chosen model. After assessing the final model on the test set, you must *not* tune the model any further!

kaggle.com competition site

Springleaf
Lending made personal

$100,000 • 1,681 teams

Springleaf Marketing Response

Merger and 1st Submission Deadline

Fri 14 Aug 2015

Mon 19 Oct 2015 (24 days to go)

Dashboard

- Home
  - Data
  - Make a submission
- Information
  - Description
  - Evaluation
  - Rules
  - Prizes
  - Timeline
- Forum
- Scripts
  - New Script
- Leaderboard
- My Submissions

Competition Details » Get the Data » Make a submission

Determine whether to send a direct mail piece to a customer

Springleaf puts the humanity back into lending by offering their customers personal and auto loans that help them take control of their lives and their finances. Direct mail is one important way Springleaf's team can connect with customers whom may be in need of a loan.

kaggle.com competition site

Springleaf
Lending made personal

$100,000 • 1,679 teams

Springleaf Marketing Response

Merger and 1st Submission Deadline

Fri 14 Aug 2015 — Mon 19 Oct 2015 (24 days to go)

Dashboard

- Home
  - Data
  - Make a submission
- Information
  - Description
  - Evaluation
  - Rules
  - Prizes
  - Timeline
- Forum
- Scripts
  - New Script
- Leaderboard
- My Submissions

Competition Details » Get the Data » Make a submission

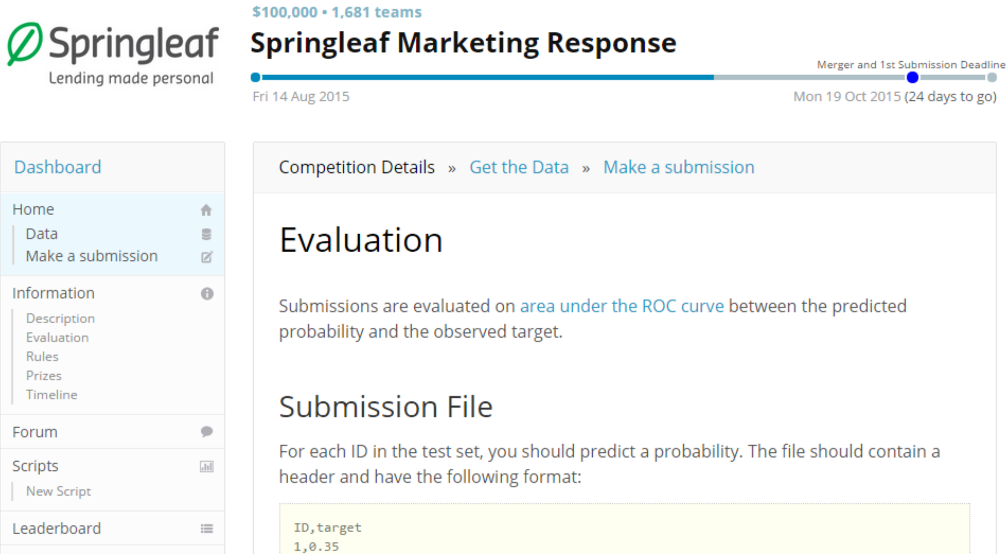
Data Files

File Name	Available Formats
test.csv	.zip (149.94 mb)
train.csv	.zip (149.83 mb)
sample_submission.csv	.zip (205.45 kb)

See this example R Script that trains an XGBoost model and creates a submission

You are provided a high-dimensional dataset of anonymized customer information. Each row corresponds to one customer. The response variable is binary and labeled "target". You must predict the target variable for every row in the test set.

kaggle.com competition site



data strategies: holdout validation

Holdout validation is tempting, BUT typically WRONG – especially when data is scarce.

data strategies: holdout validation

Holdout validation is tempting, BUT typically WRONG – especially when data is scarce.

In many cases, you will have data rich environments. In fact, to perform in-memory computations even on modern machines you may only use a small sample of your overall data.

data strategies: holdout validation

Holdout validation is tempting, BUT typically WRONG – especially when data is scarce.

In many cases, you will have data rich environments. In fact, to perform in-memory computations even on modern machines you may only use a small sample of your overall data.

For example: if you have 1,000,000,000 observations – do you really need *all* of them for modeling? if not, maybe a random sampling of $100,000$ is sufficient...

*In such a case, you could create **many** INDEPENDENT training, validation, and test samples... this could work well.*

Very basic “guideline” for data splitting:

Very basic "guideline" for data splitting:

- Splitting strategies

Very basic “guideline” for data splitting:

- Splitting strategies
  - Random

Very basic "guideline" for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)

Very basic “guideline” for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)
    – applies only to training data; not validation or test data

Very basic “guideline” for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)
    – applies only to training data; not validation or test data
- Split into training, validation, and test

Very basic "guideline" for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)
    – applies only to training data; not validation or test data
- Split into training, validation, and test
  - Select and tune model based on validation performance

Very basic “guideline” for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)
    – applies only to training data; not validation or test data
- Split into training, validation, and test
  - Select and tune model based on validation performance (possibly several iterations; several models)

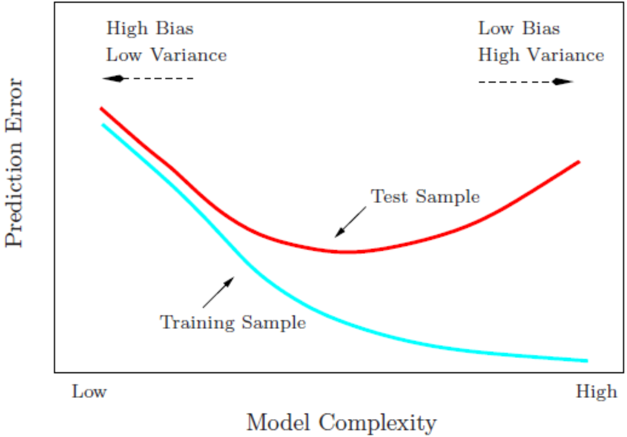
Very basic "guideline" for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)
    – applies only to training data; not validation or test data
- Split into training, validation, and test
  - Select and tune model based on validation performance (possibly several iterations; several models)
  - After selecting a final model complexity – re-train on all of train and validation data to produce the final model

Very basic “guideline” for data splitting:

- Splitting strategies
  - Random
  - Stratification (e.g., *under* or *over* sampling)
    – applies only to training data; not validation or test data
- Split into training, validation, and test
  - Select and tune model based on validation performance (possibly several iterations; several models)
  - After selecting a final model complexity – re-train on all of train and validation data to produce the final model
  - Evaluate final model on untouched, unseen, test data set

a fundamental picture



test error

What can we see from the preceding graph?

- There is an optimal model complexity that gives minimum test error.

test error

What can we see from the preceding graph?

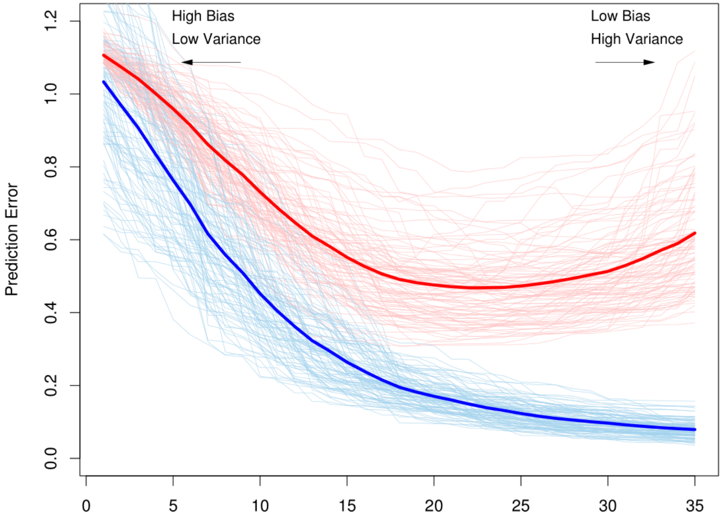
- There is an optimal model complexity that gives minimum test error.
- Training error is not a good estimate of the test error.

test error

What can we see from the preceding graph?

- There is an optimal model complexity that gives minimum test error.
- Training error is not a good estimate of the test error.
- There is a bias-variance trade-off in choosing the appropriate complexity of the model.

another important picture...



test error

What can we see from the preceding graph?

- There is a *distribution* of both train and test error
- If you do have smaller datasets – partitioning off a large chunk for testing might *hurt* your model more than it helps your model!
- A single assessment of test error may not be representative – especially for smaller data
- **We must have multiple assessments of test error to understand the distribution of test error!**

test error

What can we see from the preceding graph?

- There is a *distribution* of both train and test error
- If you do have smaller datasets – partitioning off a large chunk for testing might *hurt* your model more than it helps your model!
- A single assessment of test error may not be representative – especially for smaller data
- **We must have multiple assessments of test error to understand the distribution of test error!**

test error

What can we see from the preceding graph?

- There is a *distribution* of both train and test error
- If you do have smaller datasets – partitioning off a large chunk for testing might *hurt* your model more than it helps your model!
- A single assessment of test error may not be representative – especially for smaller data
- **We must have multiple assessments of test error to understand the distribution of test error!**

test error

What can we see from the preceding graph?

- There is a *distribution* of both train and test error
- If you do have smaller datasets – partitioning off a large chunk for testing might *hurt* your model more than it helps your model!
- A single assessment of test error may not be representative – especially for smaller data
- **We must have multiple assessments of test error to understand the distribution of test error!**

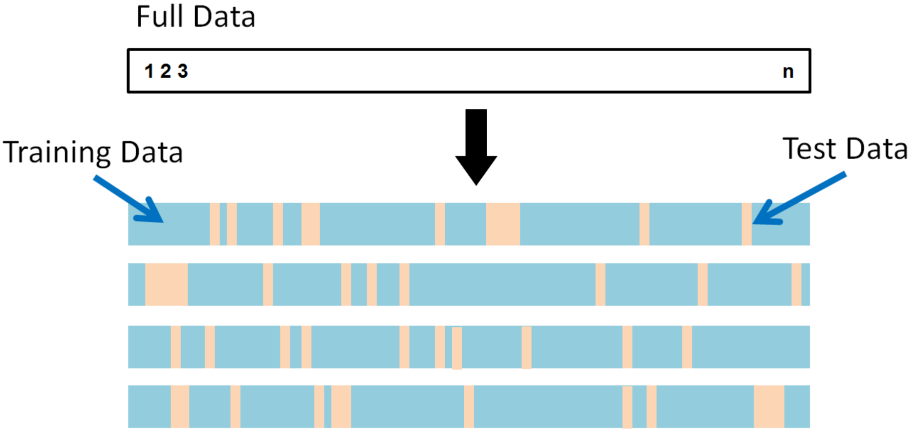
resampling methods

Resampling methods: Tools that involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain more information about the fitted model.

Types of resampling:

- cross-validation (CV)
  - random subset sampling (a.k.a. leave-group-out CV)
  - k-fold CV
  - leave-one-out CV
- bootstrap sampling

random subset sampling a.k.a. LGOCV



Same percentage withheld each time.

k-fold cross-validation

1. split the data into k distinct blocks of roughly equal size
2. leave out a block of data and fit the model on the rest
3. the model is used to predict the held-out block
4. repeat (go back to step 2) until we've predicted all k held-out blocks
5. final performance is based on hold-out predictions and error (averaged across all k predicted blocks)

Note: unlike LGOCV this ensures all data is used

k-fold cross-validation

1. split the data into k distinct blocks of roughly equal size
2. leave out a block of data and fit the model on the rest
3. the model is used to predict the held-out block
4. repeat (go back to step 2) until we've predicted all k held-out blocks
5. final performance is based on hold-out predictions and error (averaged across all k predicted blocks)

Note: unlike LGOCV this ensures all data is used

k-fold cross-validation

1. split the data into k distinct blocks of roughly equal size
2. leave out a block of data and fit the model on the rest
3. the model is used to predict the held-out block
4. repeat (go back to step 2) until we've predicted all k held-out blocks
5. final performance is based on hold-out predictions and error (averaged across all k predicted blocks)

Note: unlike LGOCV this ensures all data is used

k-fold cross-validation

1. split the data into *k* distinct blocks of roughly equal size
2. leave out a block of data and fit the model on the rest
3. the model is used to predict the held-out block
4. repeat (go back to step 2) until we've predicted all *k* held-out blocks
5. final performance is based on hold-out predictions and error (averaged across all *k* predicted blocks)

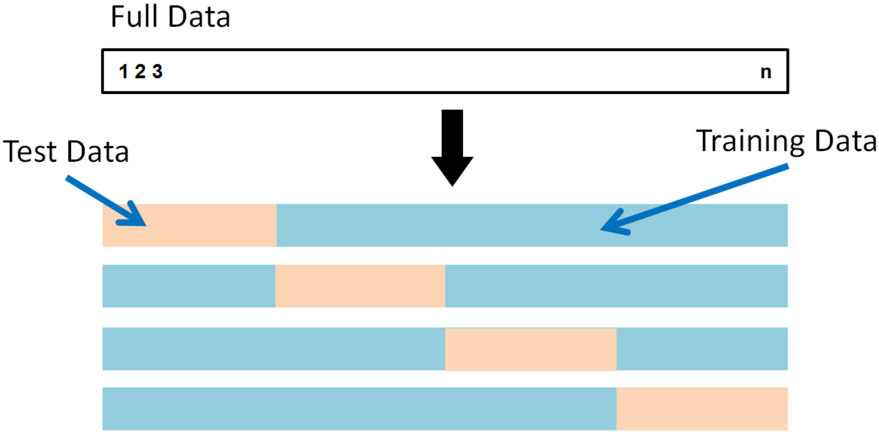
Note: unlike LGOCV this ensures all data is used

k-fold cross-validation

1. split the data into *k* distinct blocks of roughly equal size
2. leave out a block of data and fit the model on the rest
3. the model is used to predict the held-out block
4. repeat (go back to step 2) until we've predicted all *k* held-out blocks
5. final performance is based on hold-out predictions and error (averaged across all *k* predicted blocks)

Note: unlike LGOCV this ensures all data is used

k-fold cross-validation



Note: k is usually 5 or 10

k-fold cross-validation

A possible improvement on this is **repeated k-fold CV** which just means doing the whole thing multiple times. e.g., 5 repeats of 10-fold CV would give 50 total re-samples that are averaged.

Note: This is not the same as 50-fold CV.

leave-one-out cross-validation

- Use everything except of one data point for training
- This single data point is used for testing
- Identical to k-fold CV where $k = n$

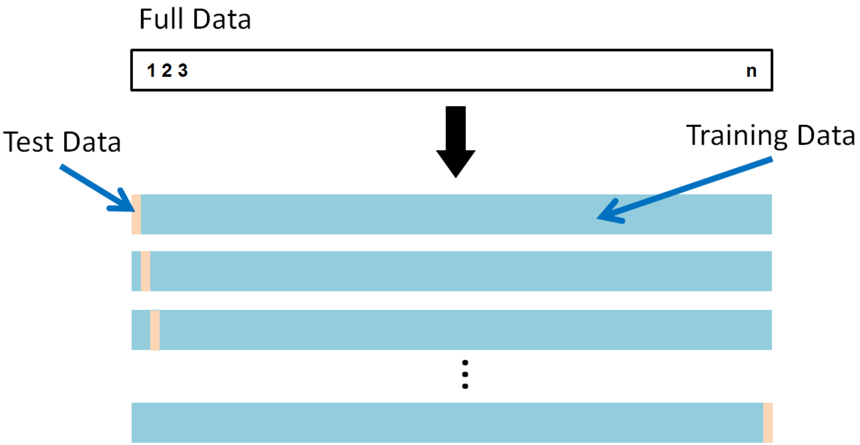
leave-one-out cross-validation

- Use everything except of one data point for training
- This single data point is used for testing
- Identical to k-fold CV where $k = n$

leave-one-out cross-validation

- Use everything except of one data point for training
- This single data point is used for testing
- Identical to k-fold CV where $k = n$

LOOCV



bootstrap sampling

“pull yourself up by your bootstraps”

This refers to the physically impossible task of actually lifting yourself up by grabbing your own bootstraps and pulling.

It is common saying which basically means: improve your situation by your own efforts, without anyone else’s help, and usually in some difficult or seemingly impossible way.

bootstrap sampling

- **Bootstrap sample**: random sample taken *with replacement*
- Bootstrapping has wide application, e.g., bootstrap aggregation ("bagging")
- Observations not selected in the bootstrap are called *out-of-bag* samples.
- Here: each of the B bootstraps are same size as original data
- Model is built on the bootstrap and tested against the out-of-bag data.

bootstrap sampling

- **Bootstrap sample**: random sample taken *with replacement*
- Bootstrapping has wide application, e.g., bootstrap aggregation ("bagging")
- Observations not selected in the bootstrap are called *out-of-bag* samples.
- Here: each of the B bootstraps are same size as original data
- Model is built on the bootstrap and tested against the out-of-bag data.

bootstrap sampling

- **Bootstrap sample**: random sample taken *with replacement*
- Bootstrapping has wide application, e.g., bootstrap aggregation ("bagging")
- Observations not selected in the bootstrap are called *out-of-bag* samples.
- Here: each of the B bootstraps are same size as original data
- Model is built on the bootstrap and tested against the out-of-bag data.

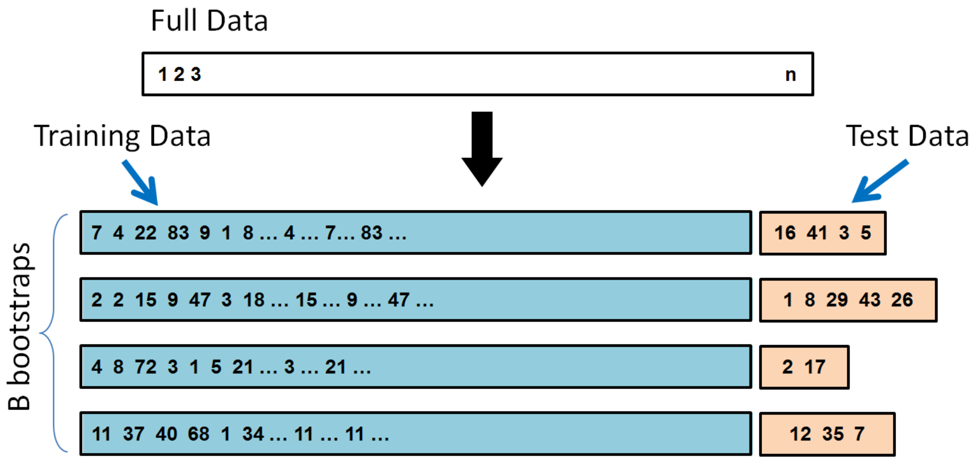
bootstrap sampling

- **Bootstrap sample**: random sample taken *with replacement*
- Bootstrapping has wide application, e.g., bootstrap aggregation (“bagging”)
- Observations not selected in the bootstrap are called *out-of-bag* samples.
- Here: each of the B bootstraps are same size as original data
- Model is built on the bootstrap and tested against the out-of-bag data.

bootstrap sampling

- **Bootstrap sample**: random sample taken *with replacement*
- Bootstrapping has wide application, e.g., bootstrap aggregation (“bagging”)
- Observations not selected in the bootstrap are called *out-of-bag* samples.
- Here: each of the B bootstraps are same size as original data
- Model is built on the bootstrap and tested against the out-of-bag data.

bootstrap sampling



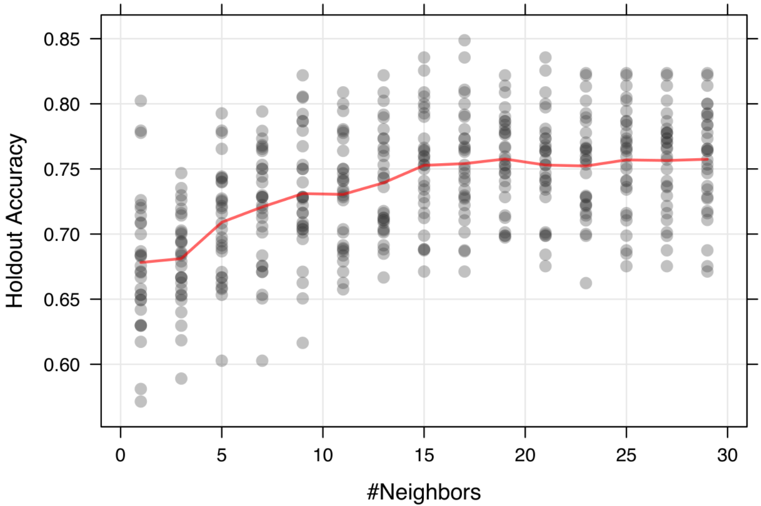
model tuning

The resampling techniques can give us good estimates of model performance on new data, but there is still the issue of selecting the right model complexity – that is, determining the best hyperparameters for a model class.

model tuning algorithm

define sets of model parameter values to evaluate;
for *each parameter set* **do**
    for *each resampling iteration* **do**
        hold-out specific samples;
        fit the model on the remainder;
        predict the hold-out samples;
    end
    calculate the average performance across hold-out predictions;
end
determine the optimal parameter set

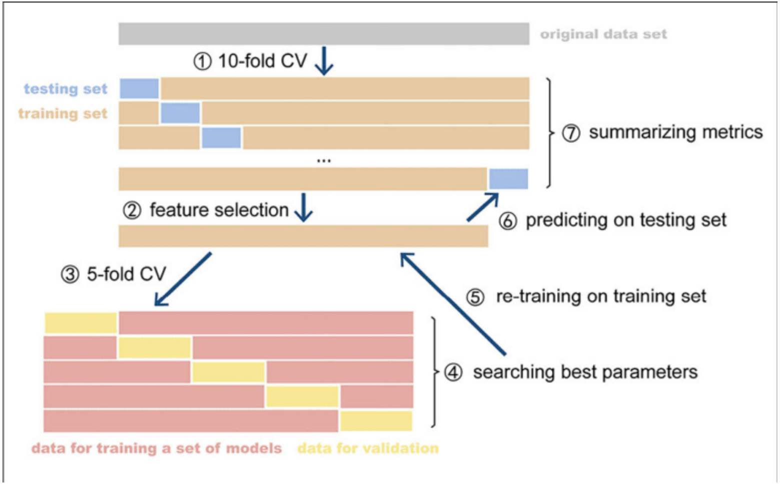
example: tuning kNN



to summarize

- A single evaluation of a model (e.g. using one test set) has limited ability to characterize the uncertainty in the results
- Resampling methods can help with tuning and performance estimates on new data.
  - error is average of error across all resampled predictions
  - k-fold is very common technique; $k = 5$ or 10
  - LOOCV less biased, but more variance than k-fold CV
  - for large data, the bias and variance difference in 5-fold, 10-fold, and bootstrapping becomes insignificant
  - *repeated k-fold less bias and less variance than single k-fold*
- Final model for resampled techniques: after selecting best complexity level, build model on full data

there are other approaches, e.g., nested cv is probably the best



model specifications

Two main conventions for specifying models in R:

- the formula interface
- the non-formula (or “matrix”) interface

formula interface

The formula interfaces explicitly lists the predictors:

$$\texttt{outcome} \sim \texttt{var1} + \texttt{var2} + \ldots$$

formula interface

The formula interfaces explicitly lists the predictors:

$$\texttt{outcome} \sim \texttt{var1} + \texttt{var2} + \ldots$$

for example,

$$\texttt{modelFunction(price} \sim \texttt{numBed} + \texttt{numBath} + \texttt{acres, data=} \texttt{housingData)}$$

would predict house price based on 3 quantitative characteristics: number of bedrooms, bathrooms, and acreage

where `modelFunction` is a command such as `lm`, `kNN`, `rpart`, etc.

formula interface

The shortcut `y ~ .` indicates that all columns in the data (except y) should be used as a predictor.

The formula interface has many conveniences, e.g.

- some transformations can be specified in-line, e.g. `log(acres)`
- automatically converts factor predictors into dummy variables (for some model functions)

matrix interface

The matrix interface specifies predictors using a matrix (or data frame). All the predictors in the object are used in the model.

Outcome data are specified as a vector object.

for example,

```
modelFunction(x = housePredictors, y = price)
```

Any transformations or dummy coding must be performed prior to being passed to the function.

Note that not all R functions have both interfaces.

building predictive models

Modeling in R generally follows a workflow:

building predictive models

Modeling in R generally follows a workflow:

1. Create the model using the appropriate function, e.g.
   `fit <- knn(trainingData, outcome, k =5)`

building predictive models

Modeling in R generally follows a workflow:

1. Create the model using the appropriate function, e.g. `fit <- knn(trainingData, outcome, k =5)`
2. Assess the properties of the model using `print`, `plot`, `summary` or other methods

building predictive models

Modeling in R generally follows a workflow:

1. Create the model using the appropriate function, e.g.
   `fit <- knn(trainingData, outcome, k =5)`
2. Assess the properties of the model using `print`, `plot`, `summary` or other methods
3. Predict outcomes for samples using the predict method:
   `predict(fit, newSamples)`

Several modeling packages in R, written by different people.
Some inconsistencies in model specification/prediction syntax.

For example,

- many models have only one method of specifying the model (e.g. formula method only)
- generating class probabilities may be different,

command	**package**	`predict` **function syntax**
`lda`	`MASS`	`predict(obj)` (no options needed)
`glm`	`stats`	`predict(obj, type =“response”)`
`rpart`	`rpart`	`predict(obj, type ="prob")`