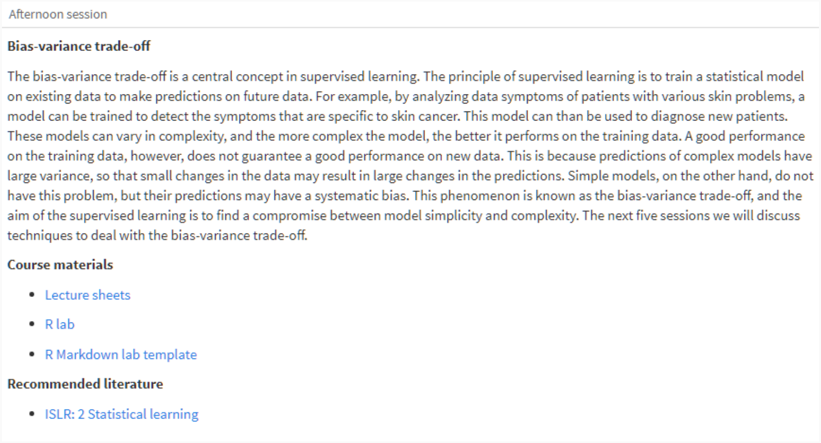
S31: Data Analysis

Bias-variance trade-off

Program



Afternoon session

Bias-variance trade-off

The bias-variance trade-off is a central concept in supervised learning. The principle of supervised learning is to train a statistical model on existing data to make predictions on future data. For example, by analyzing data symptoms of patients with various skin problems, a model can be trained to detect the symptoms that are specific to skin cancer. This model can than be used to diagnose new patients. These models can vary in complexity, and the more complex the model, the better it performs on the training data. A good performance on the training data, however, does not guarantee a good performance on new data. This is because predictions of complex models have large variance, so that small changes in the data may result in large changes in the predictions. Simple models, on the other hand, do not have this problem, but their predictions may have a systematic bias. This phenomenon is known as the bias-variance trade-off, and the aim of the supervised learning is to find a compromise between model simplicity and complexity. The next five sessions we will discuss techniques to deal with the bias-variance trade-off.

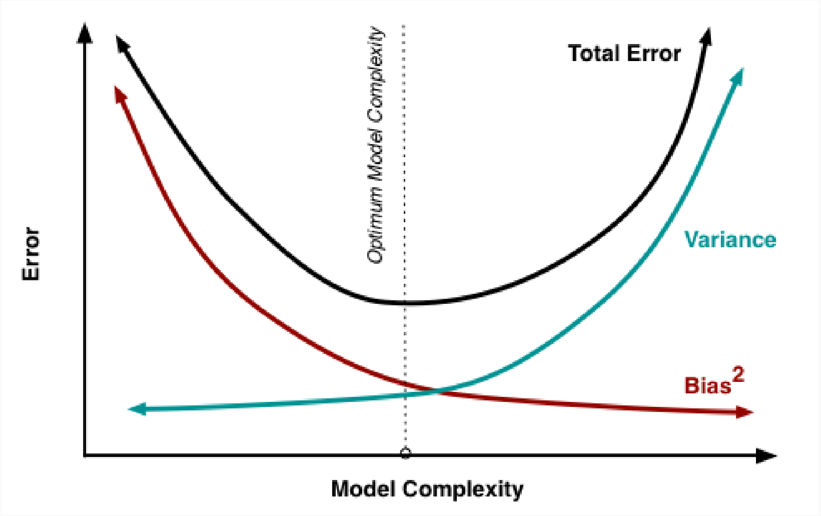
Course materials

- Lecture sheets
- R lab
- R Markdown lab template

Recommended literature

- ISLR: 2 Statistical learning

The trade-off



Content

Supervised learning

Statistics vs supervised learning

Traditional statistics:

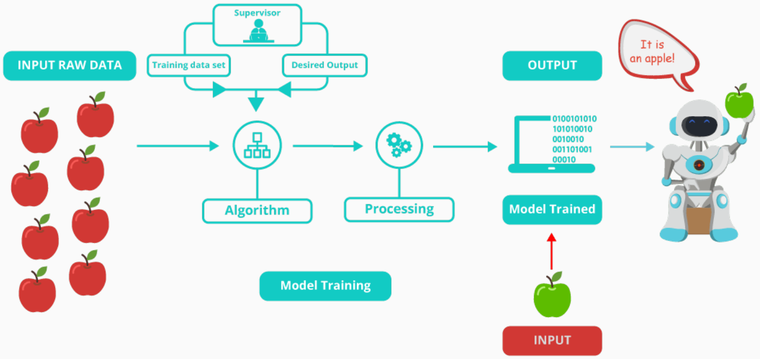
- hypothesis testing
- confirmation of theory
  - intervention X has significant effect on outcome Y

Data science:

- train a model on existing data
- to make predictions on new data
  - medical diagnosis
  - chess or backgammon program
  - face recognition

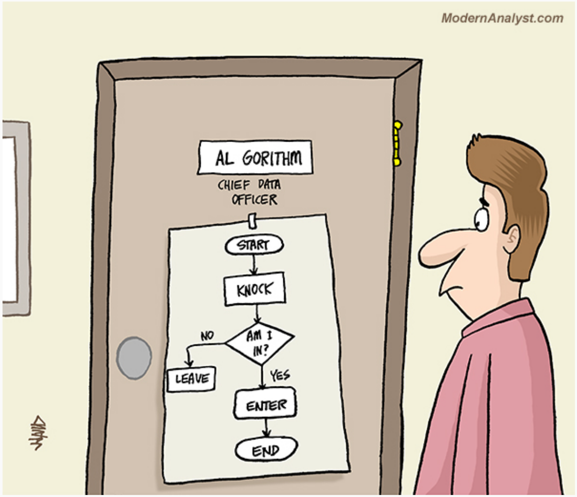
Prediction of an outcome

- outcome variable (supervisor)
- train the model using some algorithm to predict the outcome



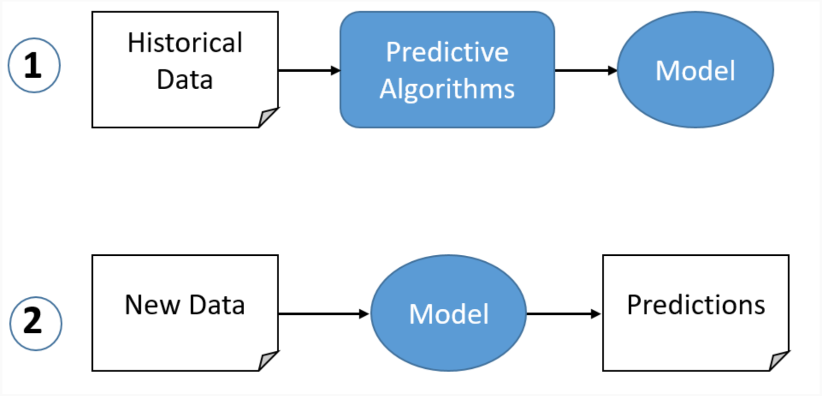
Algorithm

A set of instructions to solve a problem



Model training

- distinction between training data and new data
- aim is to predict outcome in new data set



Learning by training?

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Not really, there is a systematic approach!

Statistical models

Notation

Mathematical representation of a theory

$$y = f(x) + \epsilon$$

- y outcome/output/response/dependent variable
- x input/predictors/features
- $f(x)$ prediction function
- ϵ (irreducible) prediction error

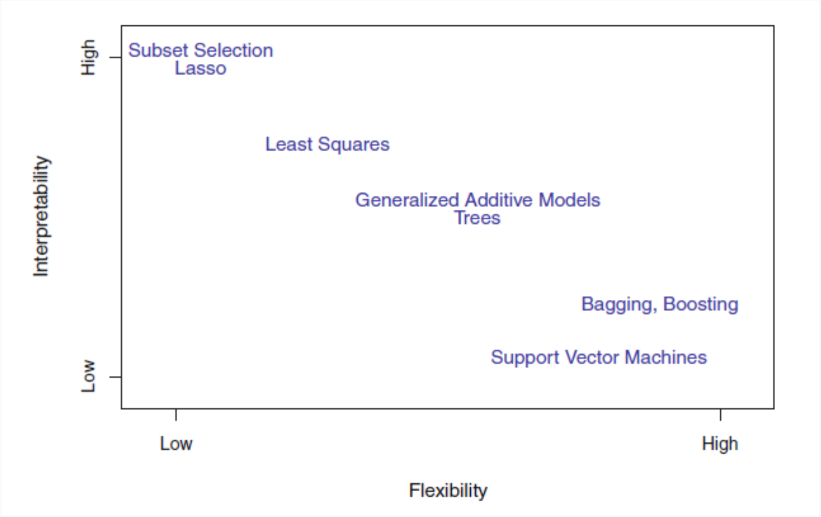
Competing $f(x)$

Many competing models

- linear regression
- nonlinear regression
- tree-based methods
- support vector machines
- neural networks
- etc.

Models differ in complexity and interpretability

Model complexity and interpretability



Model choice

“All models are wrong, but some are useful” (George Box)

What is the best model choice?

- no a priori best model
- depends on the data
- try out alternatives, and select the best performing

But how to determine which one performs best???

Bias-variance trade-off

Mean Squared Error

Measurement of model performance:

- sum of squared differences between observations and predictions

$$MSE = \frac{1}{n}\sum(y - f(x))^2 = \frac{1}{n}\sum \varepsilon^2$$

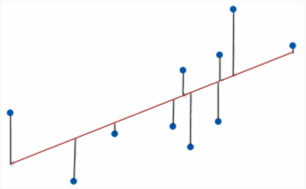


Figure 1: Linear regression example

Bias vs variance

The expected MSE is composed of bias, variance and irreducible error

$$E(MSE) = bias^2 + variance + \sigma^2$$

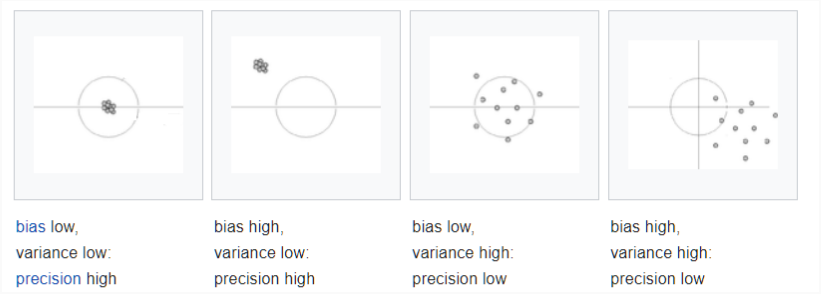


Figure 2: Estimates of multiple samples from same population

MSE vs model complexity

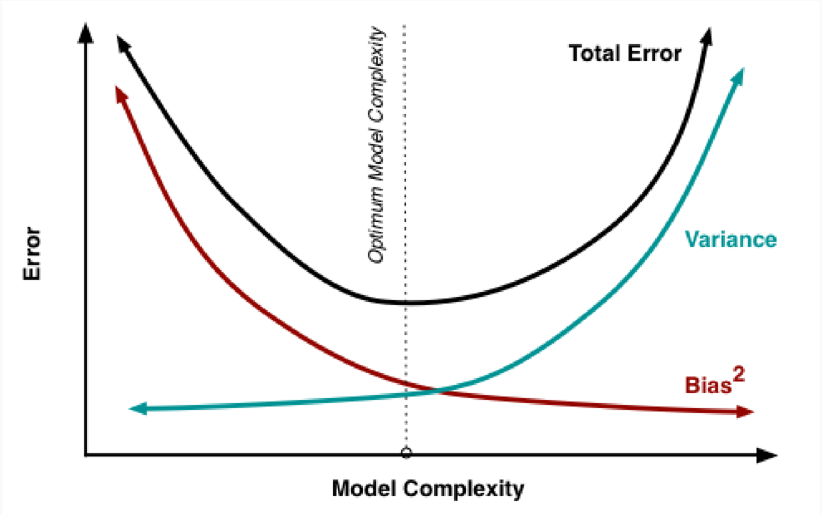


Figure 3: The bias-variance trade-off

Example

Polynomial regression models

Increasing model complexity with polynomials

- Linear model (2 parameters)

$$f(x) = \beta_0 + \beta x, \quad \epsilon \sim N(0, \sigma_\epsilon^2)$$

- Quadratic model (3 parameters)

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$$

- Cubic model (4 parameters)

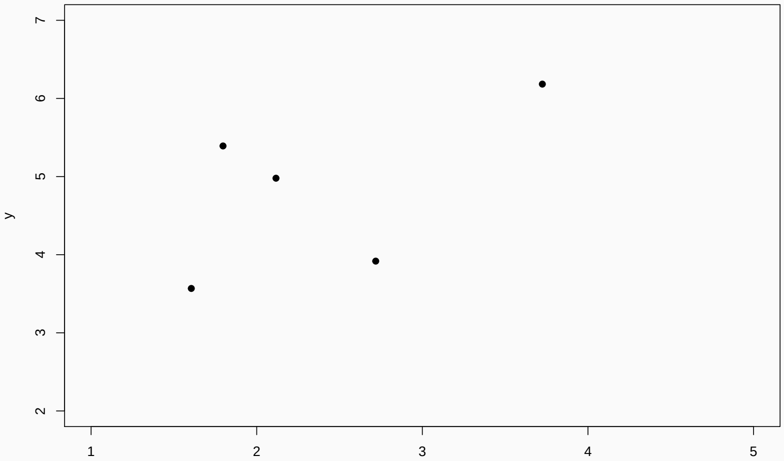
$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

etc.

Data points generated from $f(x)$

What is $f(x)$? Is it . . .

- linear, quadratic, cubic, etc.?



Strategy for finding $f(x)$

Fit all polynomial models to the data

- compute the MSE for each model
- select the model with the smallest MSE

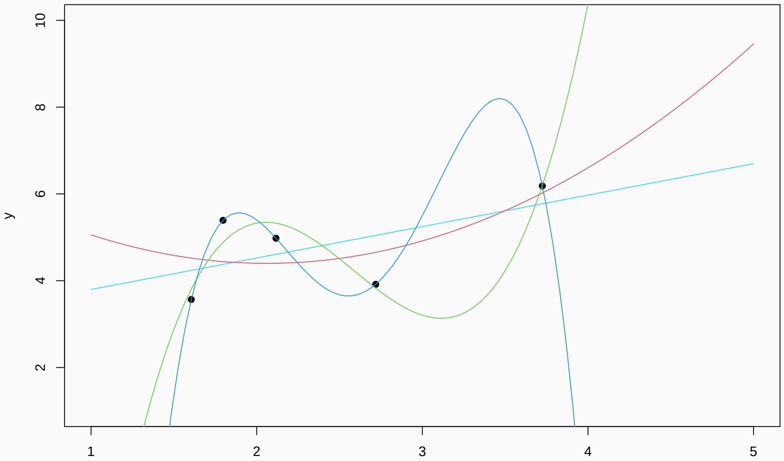
The model with the smallest MSE most resembles the true $f(x)$.

TRUE?

Predicted values for y

Regression lines linear, quadratic, cubic and quartic models

- quartic model fits data perfectly



MSE

Comparison of the MSE of the fitted models

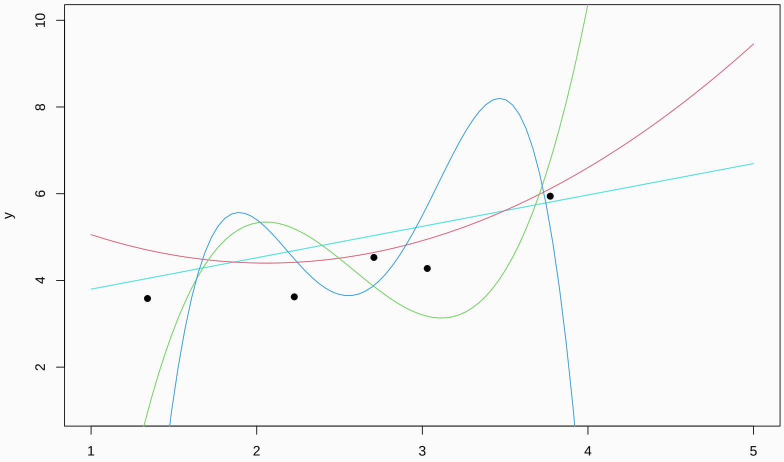
Fitted models	MSE
linear	0.6109
quadratic	0.5428
cubic	0.0881
quartic	0

So the quartic model is the best????

Five new data points from $f(x)$

Regression lines for original five data points

- How well do the models fit to these new five data points?



MSE for new data points

MSE of the new data points show different picture!

Fitted models	MSE
linear	0.523
quadratic	0.477
cubic	2.34
quartic	13.09

- quadratic model performs best
- quartic model performs worst

True $f(x)$

Data generated from from quadratic model:

$$f(x) = 2 + x + 0.1x^2 + \epsilon, \qquad \epsilon \sim N(0, 5)$$

How to find the true quadratic model when only the original sample is available?

Train/dev/test paradigm

Data partitioning

Split the sample in a training and test:

Training set:

- to estimate the model parameters (train the model)
- and cross-validate hyperparameter(s)

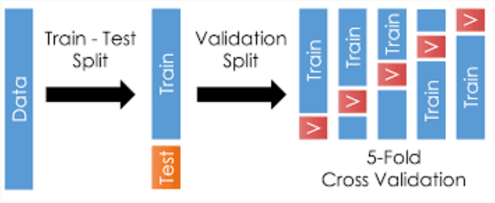
Test set:

- evaluate model performance
- test set not involved in parameter estimation
- high variance models will perform poorly on test set

Train/test paradigm

Models without hyperparameters

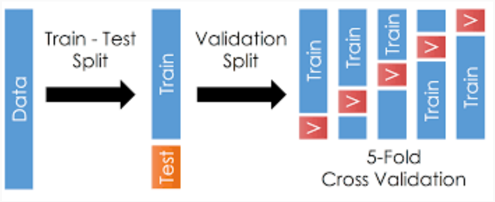
1. Train the model on the traing set
2. Evaluate the model on the test set



Train/dev/test paradigm

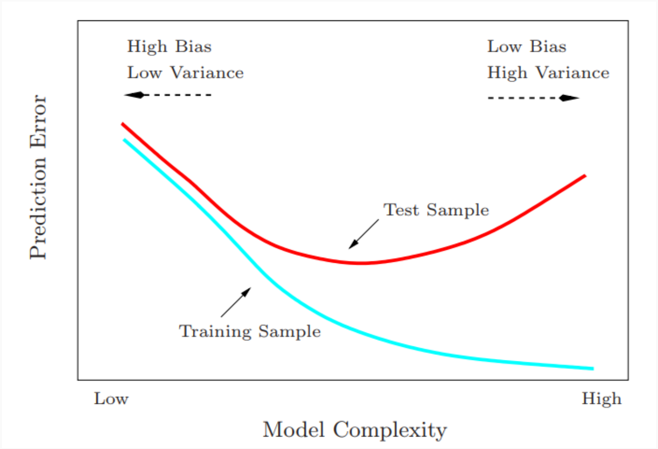
Model with hyperparameters (parameter to control learning process)

1. Select hyperparameter with smallest cross-validation error
2. Train the model on the training set
3. Evaluate the model on the test set



Train vs test MSE

Select the model with the lowest prediction error (MSE) on the test set



The `caret` package

R package for training models

Short for **C**lassification **A**nd **RE**gression **T**raining

https://cran.r-project.org/web/packages/caret/vignettes/caret.html

Model training for over 40 different models

http://topepo.github.io/caret/available-models.html

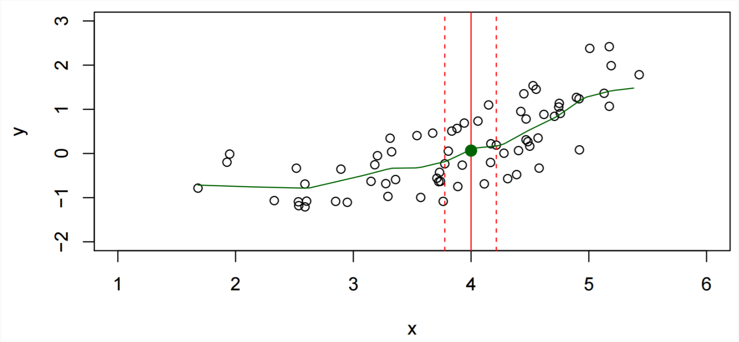
- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

Example K-nearest neighbors (KNN)

Nonparametric model with hyperparameter K

$$f(x) = \frac{1}{K}\sum_{i=1}^{K}(y|x_i \in \text{neighborhood of 4})$$

Predicted value is mean of nearest neighbors



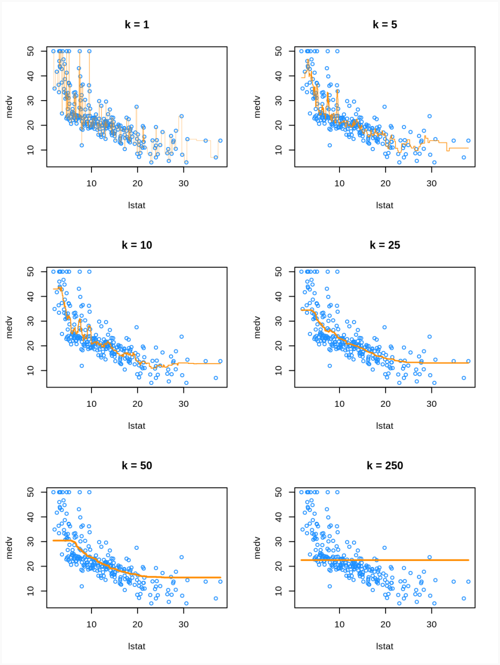


Figure 4: Predictions with varying values of K

Cross-validation procedure

Determine optimal value of K

```
# inTrain contains the rownumbers for the training set
inTrain <- createDataPartition(y    = data$y,   # y is outcome
                               p    = .8,       # 80% training cases
                               list = FALSE)
Train   <- data[ inTrain, ]                     # training set
Test    <- data[-inTrain, ]                     # test set

# train KNN model on training set using 5-fold cross-validation
knn_train <- train(formula,
                   data      = Train,
                   method    = "knn"
                   tuneGrid  = expand.grid(<range tuning parameters>),
                   trControl = trainControl(method = "cv",
                                            number = 5)
                   )

# get predictions for test set with cross-validated K
knn_test <- predict(knn_train, newdata = Test)
```

Recap

Bias-variance trade-off:

- simple models are biased, complex models have high variance
- train/dev/test procedure for optimal bias-variance trade-off

The `caret` package

- splits the data in train and test
- performs train/dev/test operation (including cross-validation)
- wide variety of models

Lab 1B

1. Fit linear, quadratic and KNN model
2. Compare test MSE to select the best model
3. Use `caret` data partition and cross-validation