Data Analysis and Visualization:

Supervised learning - regression (1/2)

Erik-Jan van Kesteren
Peter van der Heijden

Department of Methodology and Statistics

Universiteit Utrecht

Utrecht Applied Data Science

Important concepts today

- Prediction function
- k-nearest neighbors (KNN)
- Metrics for model evaluation
- Bias and variance (tradeoff)
- Training-validation-test set paradigm (or “Train/dev/test”)
- Cross-validation

Regression

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

There are usually a bunch of x's. We keep notation legible by saying x might be a vector of p predictors.

Regression

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

y: Observed outcome;

x: Observed predictor(s);

$f(x)$: Prediction function, to be estimated;

ϵ: Unobserved residuals, just defined as the “irreducible error”, $\epsilon = y - f(x)$.

The higher the variance of the irreducible error, variance$(\epsilon) = \sigma^2$, the less we can explain.

Different goals of regression

Prediction:

- Given x, work out $f(x)$.

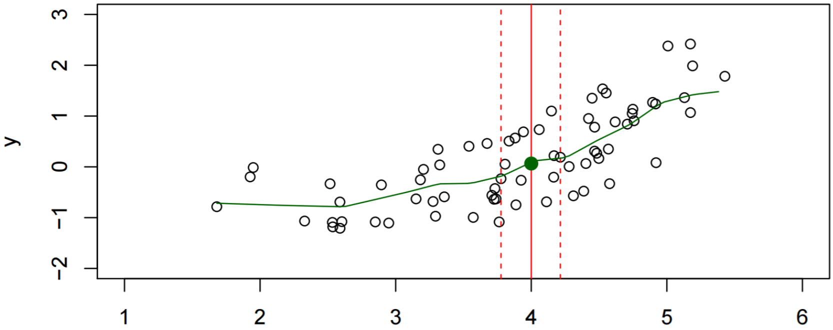
Inference:

- "Is x related to y?"
- "How is x related to y?"
- "How precise are parameters of f(x) estimated from the data?"

Estimating $f(x)$ with k-nearest neighbors (From James et al.)

- Typically we have no data points $x = 4$ exactly.
- Instead, take a "neighborhood" of points around 4 and predict its average:

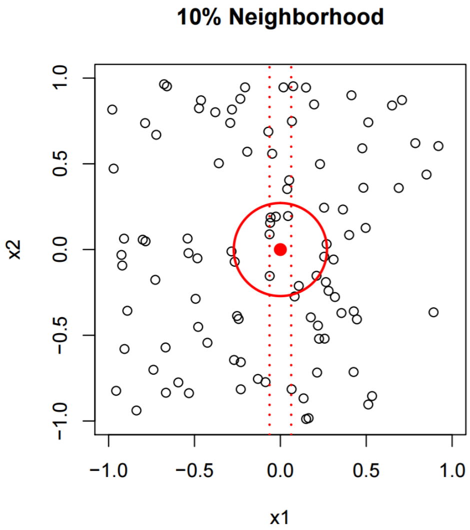
$$\hat{f}(x) = n^{-1} \sum_{i=1}^{n} (y | x \in \text{neighborhood}(x))$$

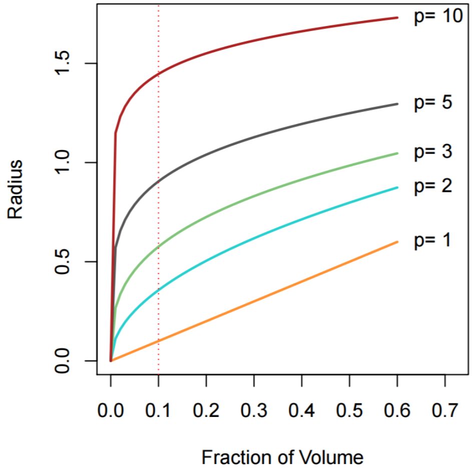


Why not kNN

- kNN is intuitive and can work well with not too many predictors;
- When there are many (say, 5 or more) predictors, kNN breaks down:
- The *closest* points on tens of predictors *simultaneously* may actually be far away.
- “Curse of dimensionality”

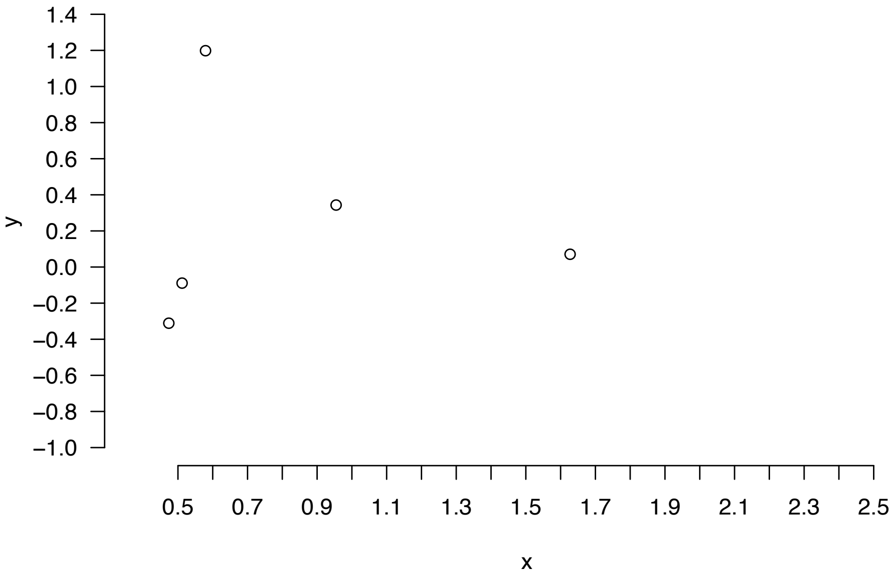
Why kNN does not work with many predictors

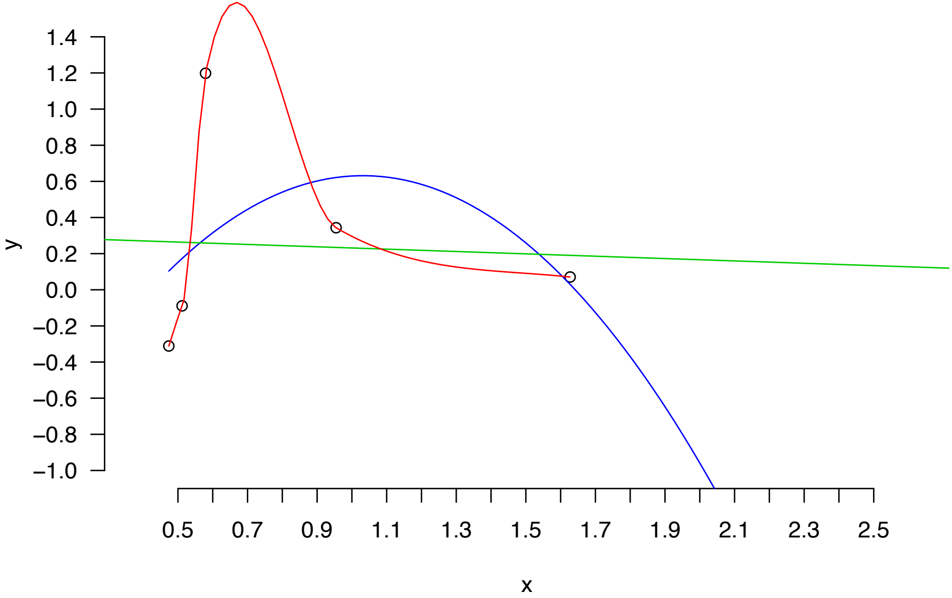




An exercise in prediction

- I am going to show you a data set, and we are going to try to estimate $\hat{f}(x)$ and predict y;
- I generated this data set myself using R, so I know the true $f(x)$ and distribution of ϵ.


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
y
0.5 0.7 0.9 1.1 1.3 1.5 1.7 1.9 2.1 2.3 2.5
x


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
y
0.5 0.7 0.9 1.1 1.3 1.5 1.7 1.9 2.1 2.3 2.5
x

Linear regression:

$$y_i = b_0 + b_1 x_i + \epsilon_i$$

(so $f(x) = b_0 + b_1 x_i$).

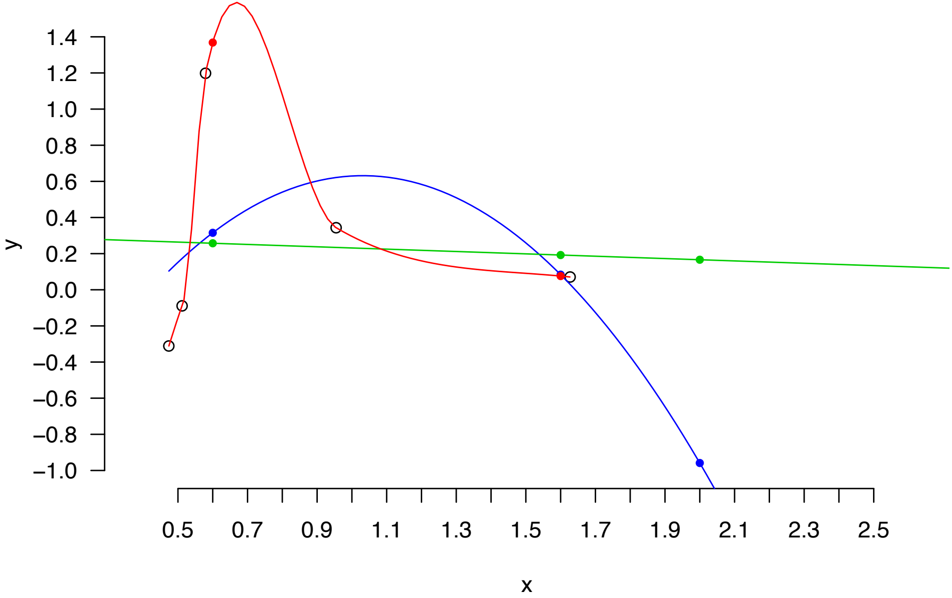
Linear regression with quadratic term:

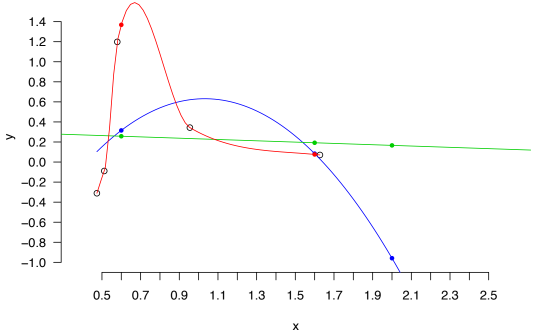
$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + \epsilon_i$$

Nonparametric (loess):

y_i is predicted from a "local" regression within a window defined by its nearest neighbors.

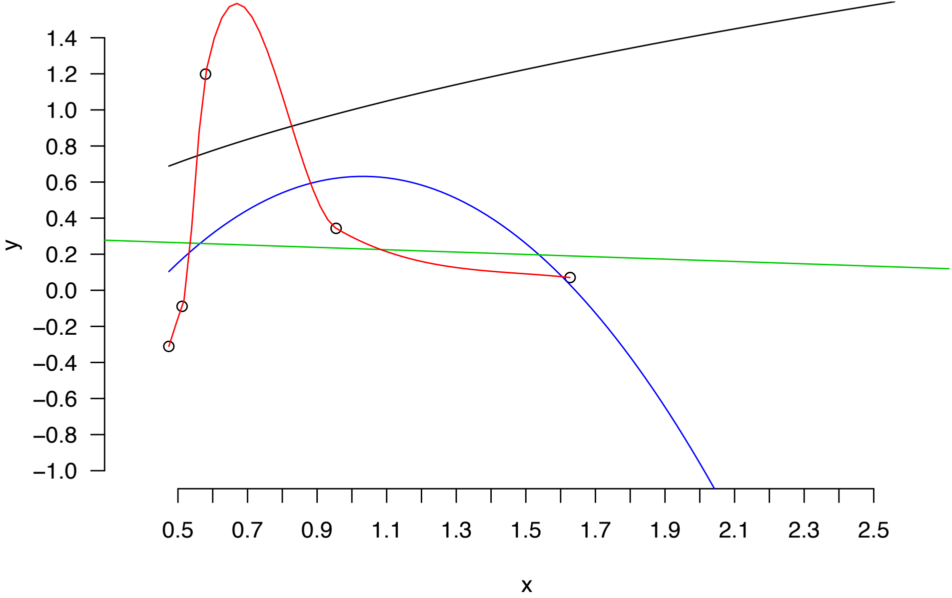
(by default: local quadratic regression and neighbors forming 75% of the data)

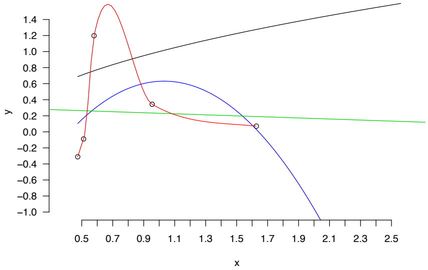

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
y
0.5 0.7 0.9 1.1 1.3 1.5 1.7 1.9 2.1 2.3 2.5
x



Model	$\hat{f}(0.6)$	$\hat{f}(1.6)$	$\hat{f}(2.0)$
Eyeballing	?	?	?
Linear regression	0.257	0.192	0.166
Linear regression w/ quadratic	0.315	0.084	-0.959
Nonparametric	1.368	0.076	–

The truth (normally we don't know this)


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
y
0.5
0.7
0.9
1.1
1.3
1.5
1.7
1.9
2.1
2.3
2.5
x



Model	$\hat{f}(0.6)$	$\hat{f}(1.6)$	$\hat{f}(2.0)$
Eyeballing	?	?	?
Linear regression	0.257	0.192	0.166
Linear regression w/ quadratic	0.315	0.084	-0.959
Nonparametric	1.368	0.076	–
Truth	0.775	1.265	1.414

Model accuracy

- The predictions $\hat{y}$ differ from the true y;
- We can evaluate how much this happens “on average”.

A few model evaluation metrics

- Mean squared error (MSE):

$$\text{MSE} = n^{-1} \sum_{i=1}^{n} (y - \hat{y})^2$$

- Root mean squared error $\text{RMSE} = \sqrt{\text{MSE}}$
- Mean absolute error (MAE:)

$$\text{MAE} = n^{-1} \sum_{i=1}^{n} |y - \hat{y}|$$

- Median absolute error (mAE):

$$\text{mAE} = \text{median}|y - \hat{y}|$$

- Proportion of variance explained: (R^2)

$$R^2 = \text{correlation}(y, \hat{y})^2$$

- Which model appears to fit best to the training data?
- Calculate MSE for each model, relative to truth.
- Which is the best model in terms of MSE?

...And the winner is...

Model	MSE	MSE (interpolation only)
Eyeballing	?	?
Linear regression	0.992	0.709
Linear regression w/ quadratic	2.410	0.883
Nonparametric	–	0.883

True $f(x)$:

$$y = \sqrt{x} + \epsilon$$

with $\epsilon \sim \text{Normal}(0, 1)$.

What happened?

- There were few observations, relative to the complexity of most models (except linear regression);
- The observed data were a random sample from the true "data-generating process"

$$f(x) + \epsilon,$$

BUT

- By chance, some patterns appeared that are not in the true $f(x)$;
- The more flexible models $\hat{f}(x)$ **overfitted** these patterns.

Thought experiment

Imagine we had sampled a different 5 observations, re-fitted all of the models, and predicted again. Each time we remember the predictions given. We do this a large number of times, and then take the average for the predictions over all samples.

Questions:

- Which model(s) would, on average, give the prediction corresponding exactly to $f(x) = \sqrt(x)$?
- Which models' predictions would vary the most?
- Which model would you guess (!) to have the lowest MSE, on average?

Unbiased:

Model that gives the correct prediction, on average over samples from the target population

- Unbiased in this case: nonparametric, square-root
- Biased in this case: all others

High variance:

Model that easily overfits accidental patterns.

- High variance in this case: nonparametric, quadratic
- Low variance in this case: linear regression

Bias-variance tradeoff

- Flexibility $\rightarrow$ lower bias
- Flexibility $\rightarrow$ higher variance

Bias and variance are implicitly linked because they are both affected by model complexity.

Possible definitions of "complexity"

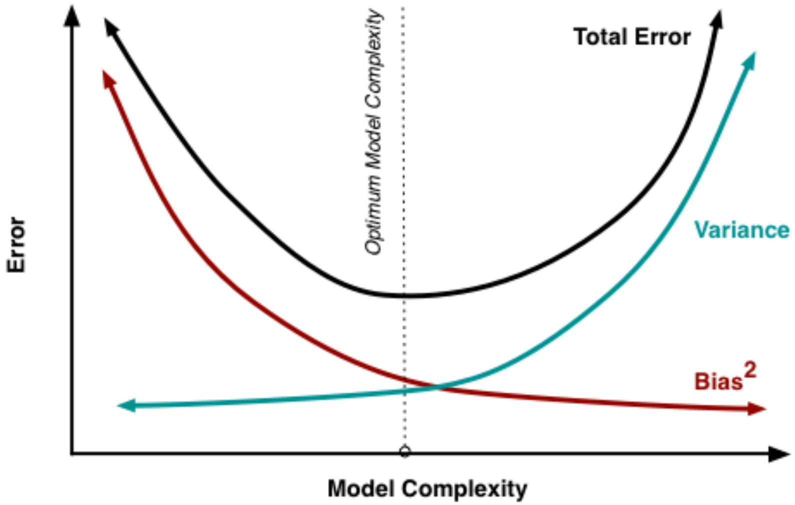
- Amount of information in data absorbed into model;
- Amount of compression performed on data by model;
- Number of effective parameters, relative to effective degrees of freedom in data.

For example:

- More predictors, more complexity;
- Higher-order polynomial, more complexity (x, x^2, x^3, $x_1 \times x_2$, etc.);
- Smaller "neighborhood" in KNN, more complexity
- ...

Note: bias variance tradeoff occurs just as much with $n = 1,000,000,000$ as it does with $n = 5$!

MSE contains both bias and variance (picture)



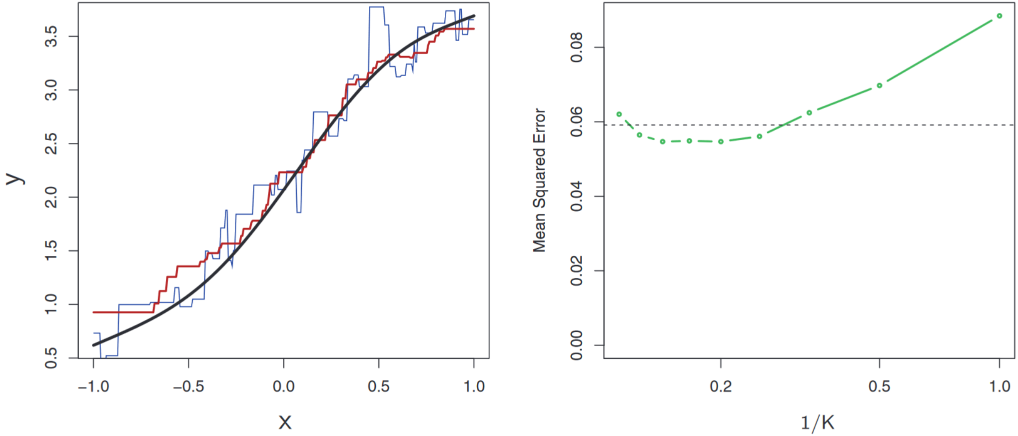
MSE contains both bias and variance (equation)

$$E(\text{MSE}) = \text{Bias}^2 + \text{Variance} + \sigma^2$$

Population mean squared error is squared bias PLUS model variance PLUS irreducible variance.

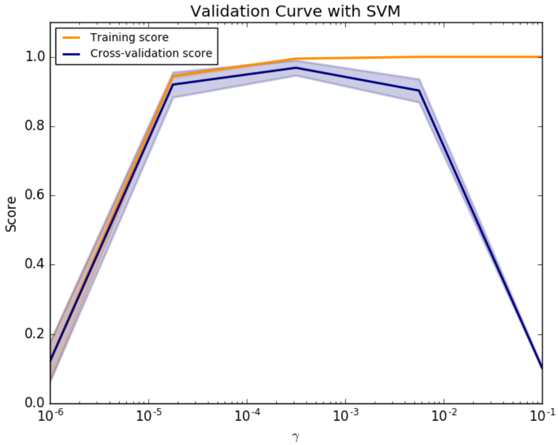
(The $E(.)$ means "on average over samples from the target population").

Mean square error of KNN with different levels of K

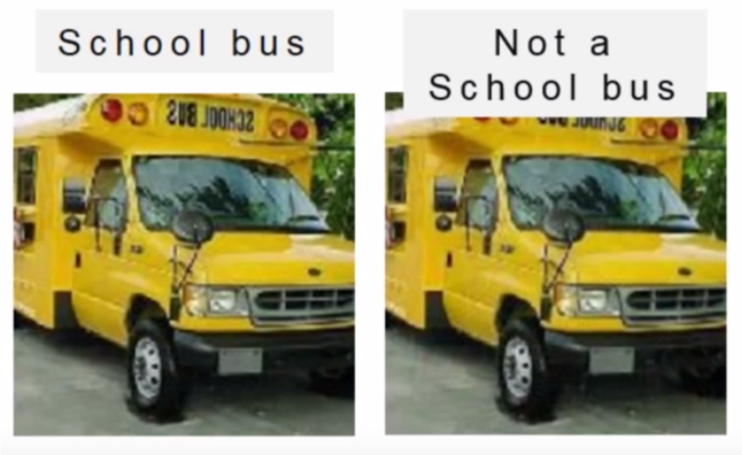


ISLR, Figure 3.19

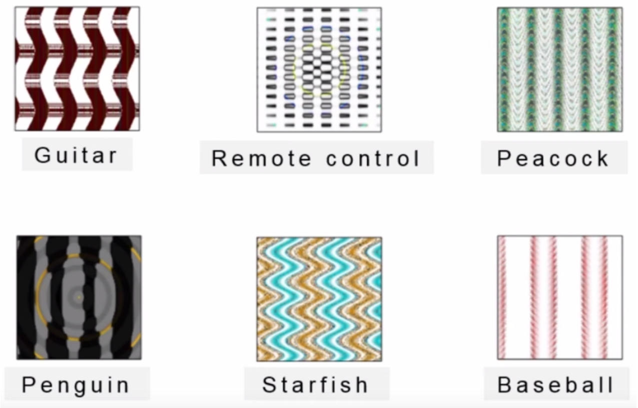
Observed training and test error in a flexible model



“Neural networks are easily fooled”



“Neural networks are easily fooled”



https://www.youtube.com/watch?v=M2IebCN9Ht4

What this means in practice

- Sometimes a wrong model is better than a true model (on average etc);
- If you do not believe in true models: sometimes a simple model is better than a more complex one.
- These factors **together** determine what works best:
  - How close the functional form of $\hat{f}(x)$ is to the true $f(x)$;
  - The amount of irreducible variance (σ^2);
  - The sample size (n);
  - The complexity of the model (p/df or equivalent).

Training-validation-test paradigm

- So far, I have **cheated**;
- I **knew** the true $f(x)$ so you could calculate exactly what $E(\text{MSE})$ was;
- This is sometimes called the “Bayes error”.

- In practice we do not know the truth;

$\rightarrow$ **How can we estimate** $E(\text{MSE})$?

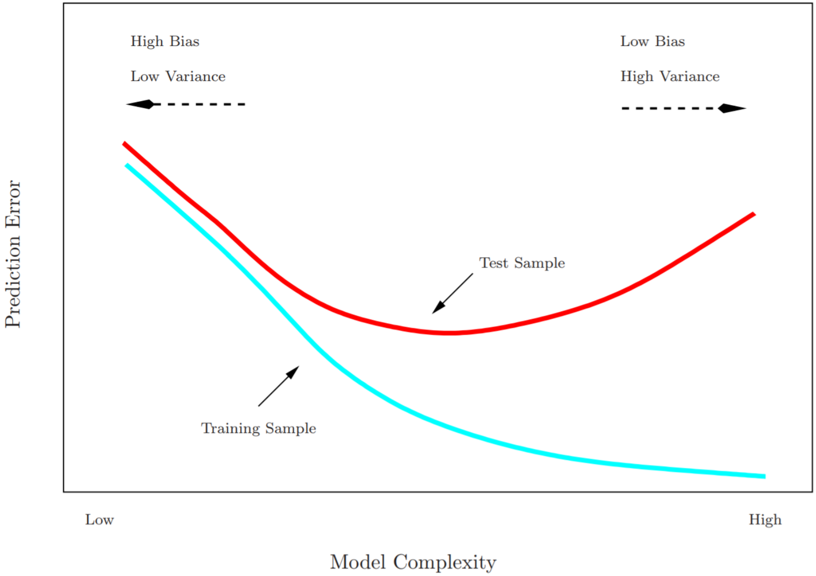
Train/dev/test

Training data:
Observations used to fit $\hat{f}(x)$

Validation data (or “dev” data):
New observations from the same source as training data
(Used several times to select model complexity)

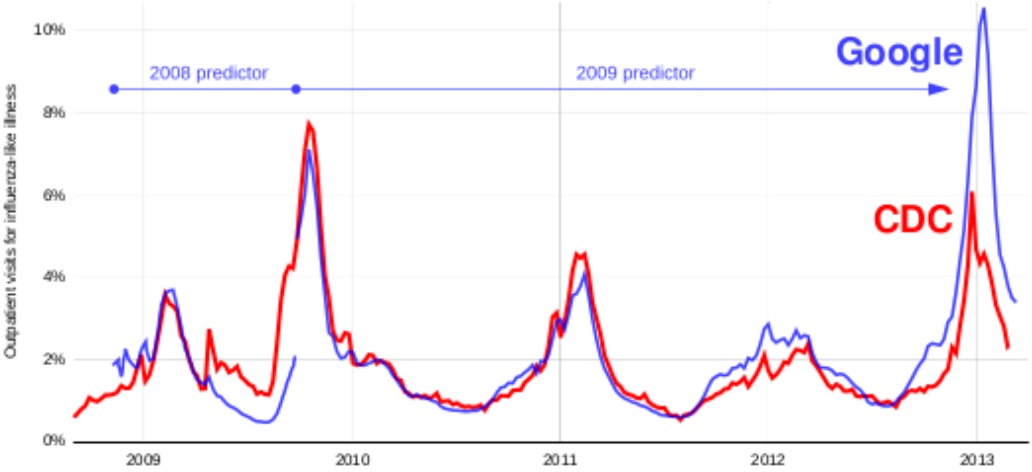
Test data:
New observations from the intended prediction situation

Question: Why don’t these give the same average MSE?


High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Prediction Error
Low
High
Model Complexity

Train/dev/test

- The idea is that the average squared error in the test set MSE_{test} is a good estimate of the Bayes error $E(\text{MSE})$
- This only holds when the test set is "like" the intended prediction situation!


Google
CDC
2008 predictor
2009 predictor
Outpatient visits for influenza-like illness
10%
8%
6%
4%
2%
0%
2009
2010
2011
2012
2013

Drawbacks of train/validation split

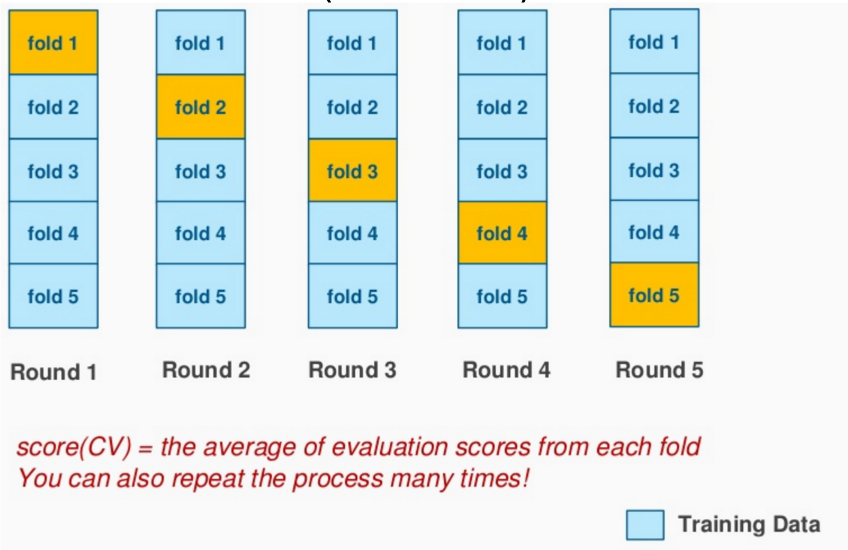
- the validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to overestimate the test error for the model fit on the entire data set.

From `https://www.edx.org/course/statistical-learning`

K-fold crossvalidation

- "Cross-validation" often used to replace single dev set approach;
- Perform the train/dev split several times, and average the model accuracy.
- Usually $K = 5$ or $K = 10$.
- When $K = N$, "leave-one-out";

K-fold crossvalidation ($K = 5$ here)



Correctly using cross-validation

Consider a simple regression used to predict an outcome:

1. Starting with 5000 predictors and 500 cases, find the 100 predictors having the largest correlation with the outcome;
2. We then fit a linear regression, using only these 100 predictors.

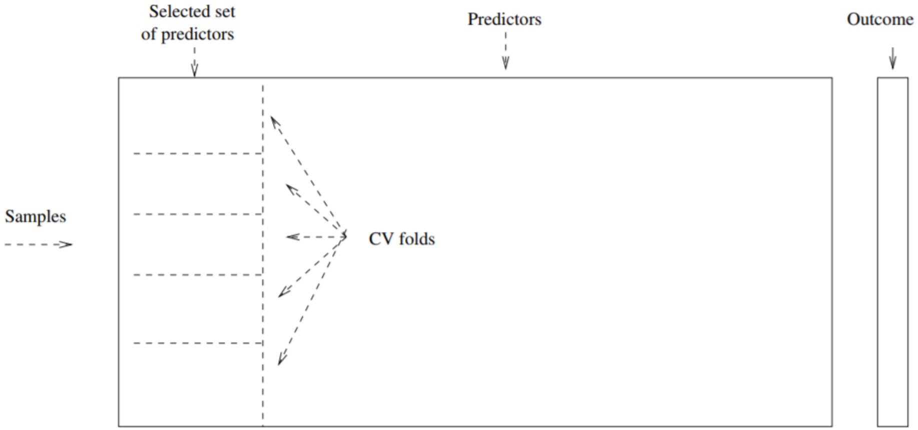
Question:
How do we estimate the test set performance of this method?
Can we apply cross-validation in step 2, forgetting about step 1?

From `https://youtu.be/r64tRyHFAJ8`

Answer: In Step 1, the procedure has already seen the labels of the training data, and made use of them. This is a form of training and must be included in the validation process!

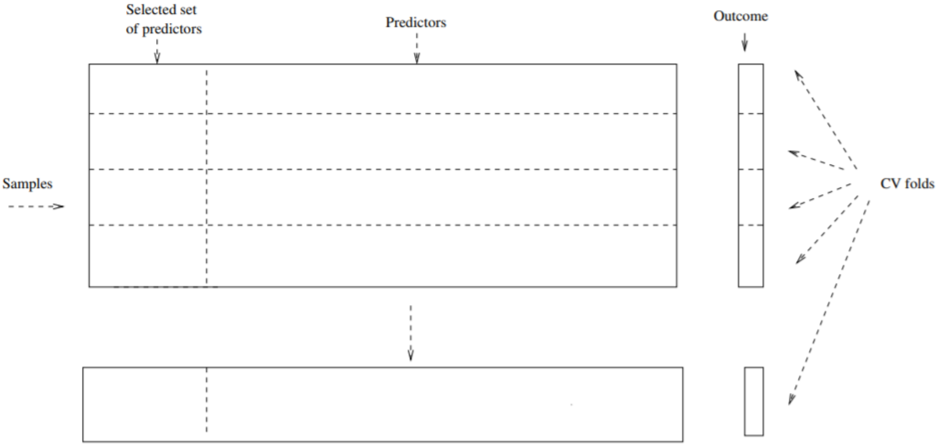
From `https://youtu.be/r64tRyHFAJ8`

Wrong way



From `https://youtu.be/r64tRyHFAJ8`

Right way



From https://youtu.be/r64tRyHFAJ8

Conclusion

Conclusion

- Bias and variance trade off, in theory;
- Bias and variance trade off, in practice;
- We try to estimate error using train/dev/test paradigm;
- Cross-validation is a useful alternative to separate dev set;
- It is important to be precise when applying this setup;
- Beware that **any procedure that makes decisions based on the data** requires validation!
- Getting good test data is **difficult problem**;