An Overview of Statistical Learning

(BOOT CAMP)

Hugh Chipman

Acadia University

January 12, 2015

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Outline

Introduction

Cartoon examples of supervised learning

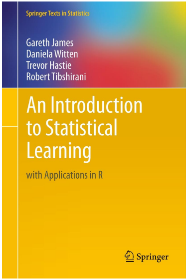
Cross-validation and computational methods for inference

Classification

Big ideas in statistical learning

Some thoughts

2014 AARMS summer school class (Sunny Wang, Statistical Learning co-teacher 2nd from right, first row)



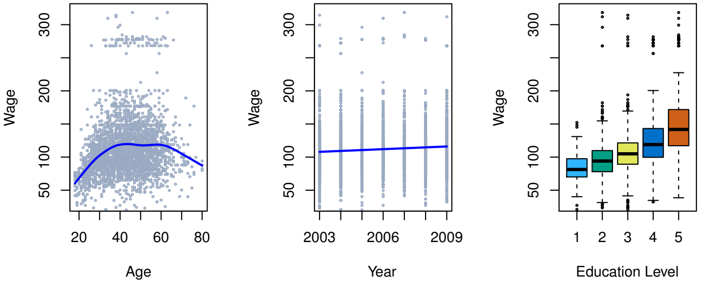
Primary source: *Introduction to Statistical Learning with Applications in R* by James, Witten, Hastie and Tibshirani

Some other resources:

- *Statistical Learning and Data Mining*, Hastie, Tibshirani and Friedman
- *Pattern Recognition and Machine Learning*, Bishop
- *Bayesian Methods for Nonlinear Classification and Regression*, Denison, Holmes, Mallick and Smith.

Some examples of statistical learning

Wage data : Predict salary using demographic variables



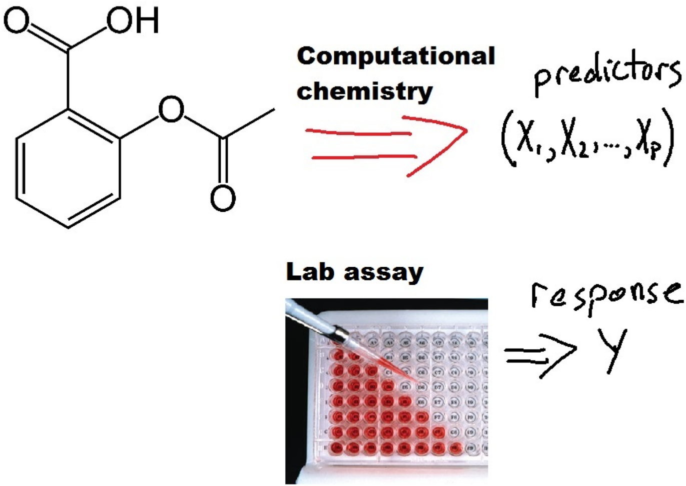
Plots show dependence of wage on individual predictors

Some examples of statistical learning

Drug discovery:

- Identify compounds with desirable effect on biological target
- Response variable: Activity (inactive/active)
- Explanatory variables: Molecular descriptors
- Use high throughput screening to test thousands of compounds, then build a model to predict activity for other compounds.

Some examples of statistical learning



Some examples of statistical learning: Supervised Learning

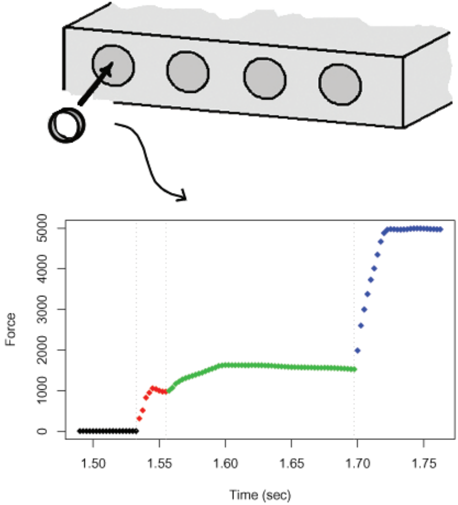
The wage and drug discovery problems are examples of Supervised Learning.

- We seek to predict a response Y using predictors X.
- We have available a training sample of (X, Y) pairs.
- Continuous response (wage) $\Rightarrow$ “regression”
- Categorical response (drug discovery) $\Rightarrow$ “classification”

Although not the focus of this overview, there are also methods for unsupervised learning

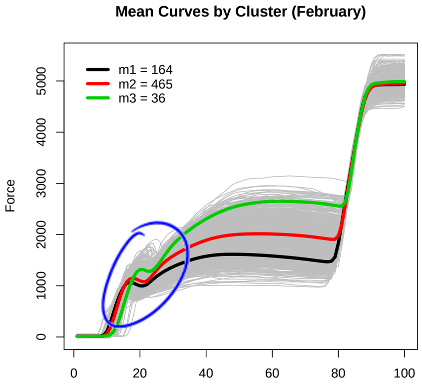
- Discover structure in X without an observed Y.
- Clustering, principal component analysis, graphical models, ...

An unsupervised learning example



- Engine assembly process.
- Steel valve seats force-fitted into cylinder head.
- Data: force profile vs. time for each insertion
- Problem: some insertions bad, but we can't tell which ones.

An unsupervised learning example



- Each observation is a curve
- We have thousands of curves
- Try to group together curves and identify anomalous insertions
- "grouping" = "clustering"

Outline

Introduction

Cartoon examples of supervised learning

Cross-validation and computational methods for inference

Classification

Big ideas in statistical learning

Some thoughts

Regression

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

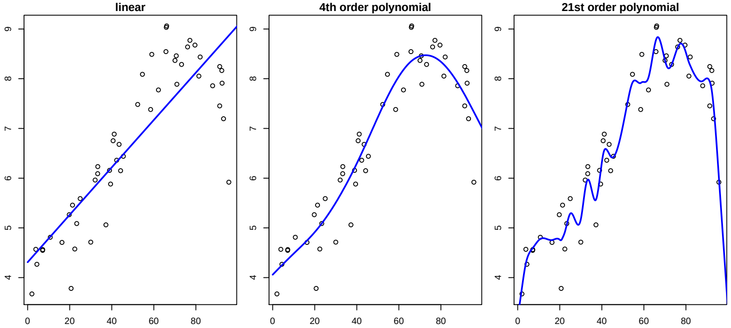
- y = response variable
- x = predictor variable(s)
- $f(x)$ is an unknown function we wish to estimate ("learn")
- ε is a random error

$$y = \text{signal} + \text{noise}$$

Statistical learning typically focuses on estimation of "signal", with minimal attention given to "noise".

A one-dimensional regression example

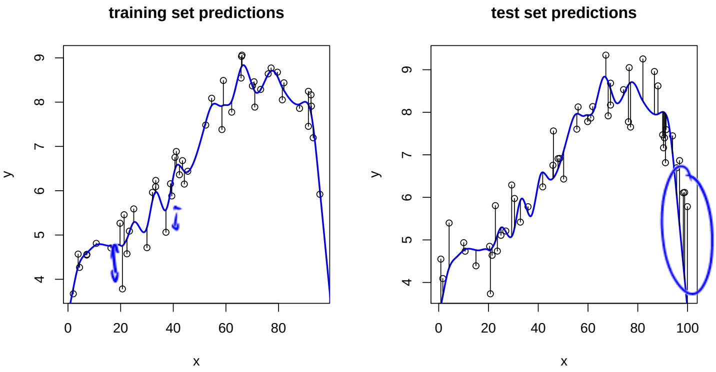
- One dataset ("training data") and 3 different regression models.
- Polynomial regression $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d + \varepsilon$.
- Objectives:
  1) choose flexibility (d) 2) estimate parameters (β's)
- Prediction model: $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \ldots + \hat{\beta}_d x^d$



How to choose a suitable flexibility?

One very general approach: use a **test set**.

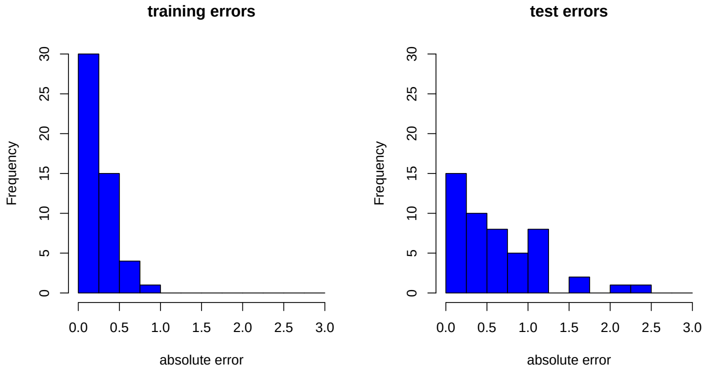
- A set of data points **not** used to estimate the parameters.
- Plot below: errors on training and test sets.



How to choose a suitable flexibility?

In this case (an order $d = 21$ polynomial), test set errors are larger.

This suggests our model may be too flexible and a smaller d should be used.

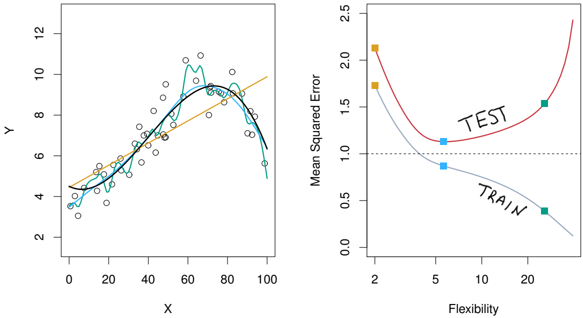


Training and test errors as a function of flexibility

Returning to the 3 different models (left panel), we can compute the mean squared error for a training or a test set.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

MSE will vary as a function of flexibility (right panel):



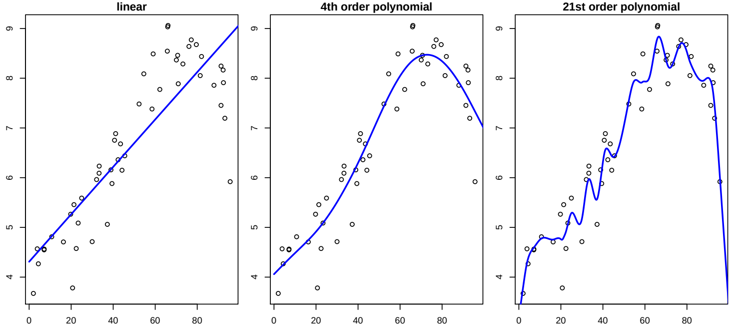
- monotone decreasing shape for training set
- "U" shape for test set

The bias-variance trade-off

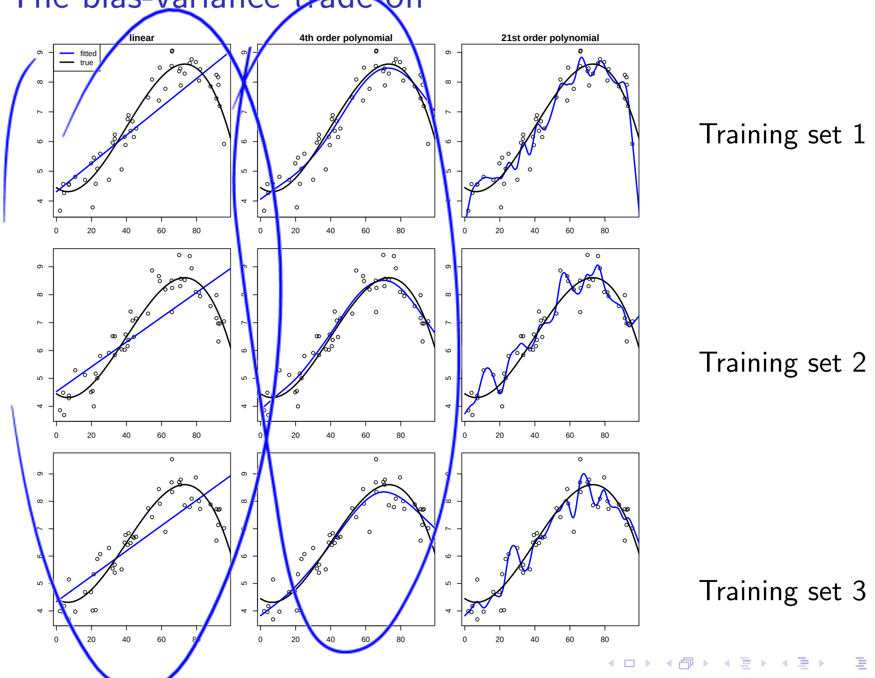
(Robert Bell's "Fundamental Challenge")

- The linear model is not flexible enough: **biased**.
- The order 21 polynomial is too flexible: **variable**.

This is the bias variance trade-off

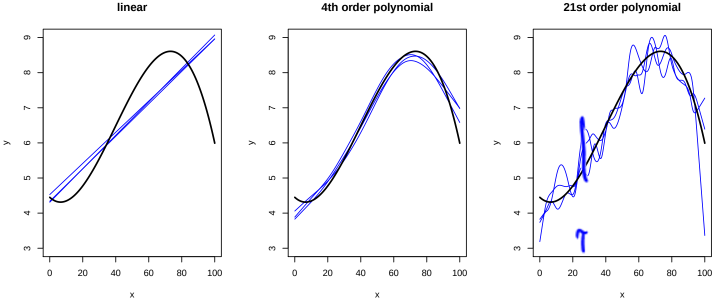


The bias-variance trade-off



The bias-variance trade-off

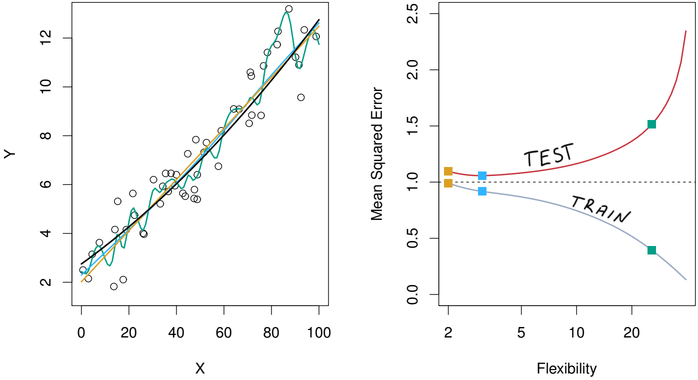
Combine the 3 fits in a single plot:



High bias Low variance	Just right	Low bias High variance

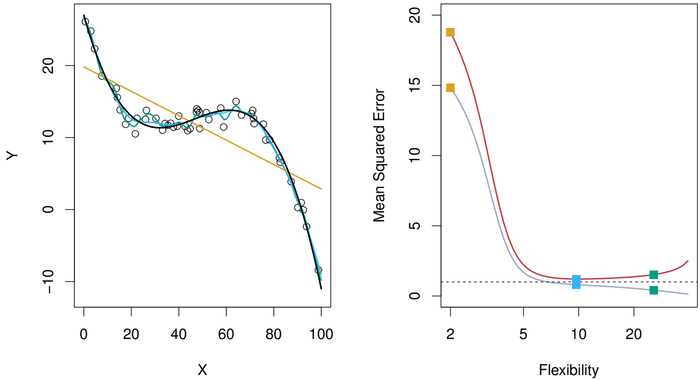
Other one-dimensional examples

True function is nearly linear, noise level is high
(previous example was nonlinear, high noise)



Other one-dimensional examples

True function is nonlinear, noise level is low
What's the best flexibility? **It depends!**

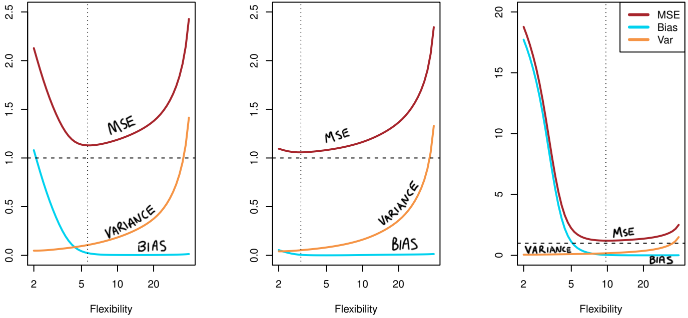


Supervised learning

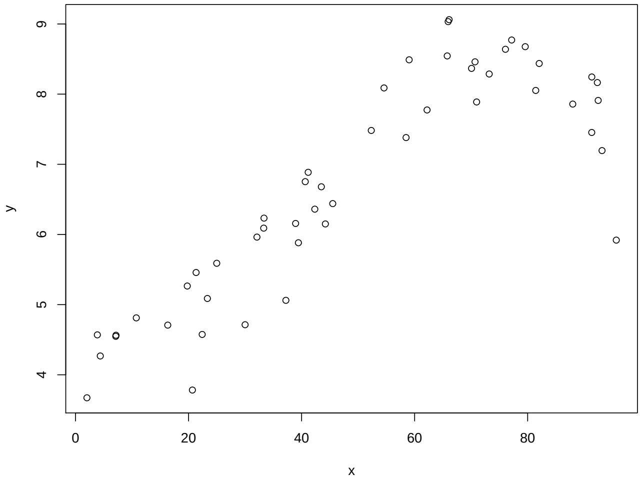
$$E\left[(y - \hat{f}(x))^2\right] = E\left((y - f(x))^2\right) + Var(\hat{f}(x))$$

In the three examples, we can break down the MSE into bias and variance:

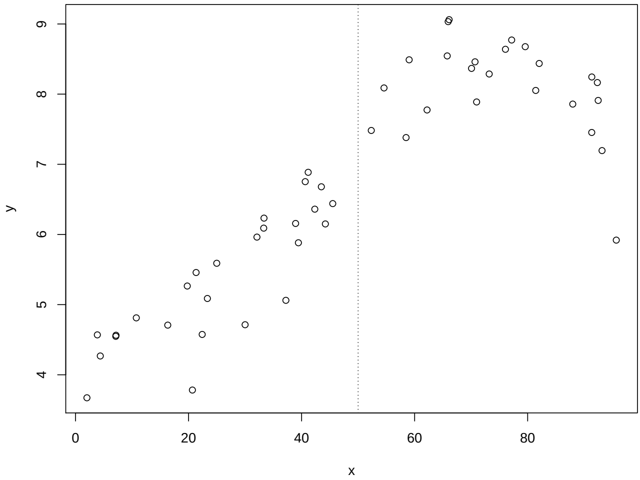
nonlinear high noise	linear high noise	nonlinear low noise



K-nearest neighbours with K=10

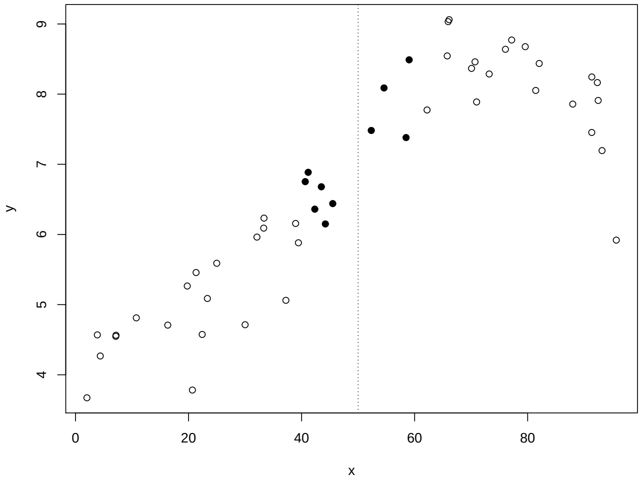


K-nearest neighbours with K=10



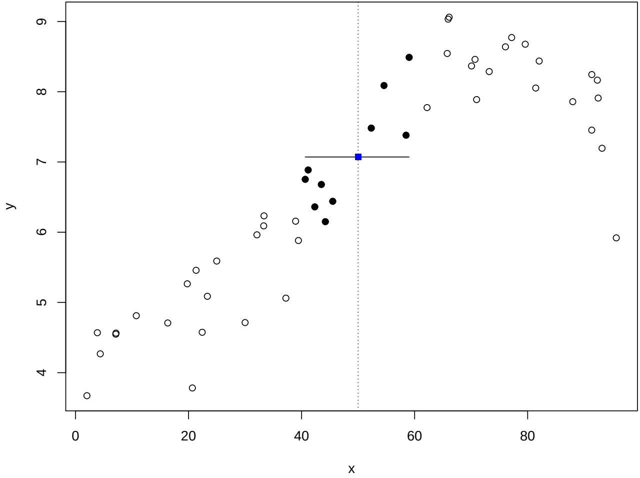
We want to predict y at $x = 50$.

K-nearest neighbours with K=10



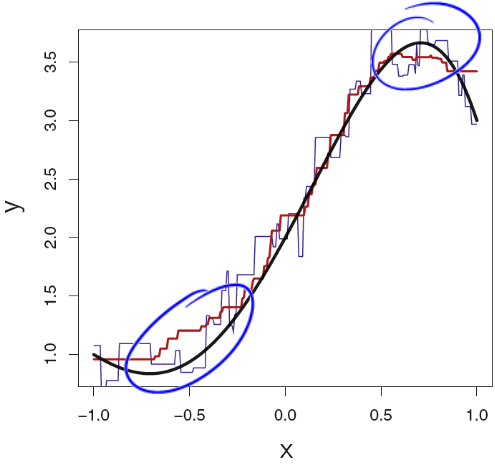
Use 10 nearest neighbours.

K-nearest neighbours with K=10



Prediction is average y of the 10 nearest neighbours

KNN results for K=1 and K=9

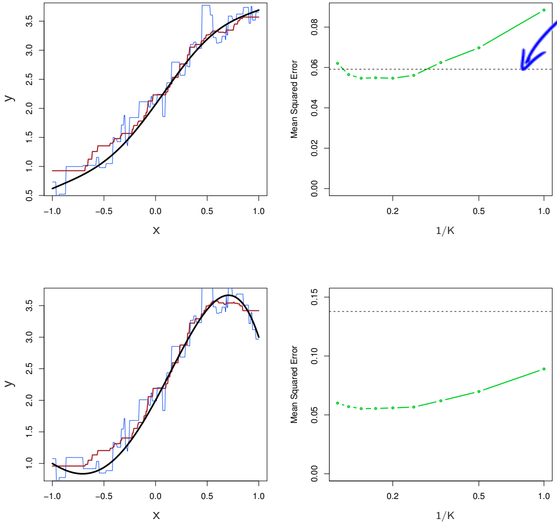


- Predictions are piecewise constant
- $K = 1$ high variance, low bias
- $K = 9$ higher bias, lower variance

KNN vs. linear regression: Rounds 1 and 2

True function and KNN fit — test error

lin. reg



Near-linear: KNN can do as well as regression

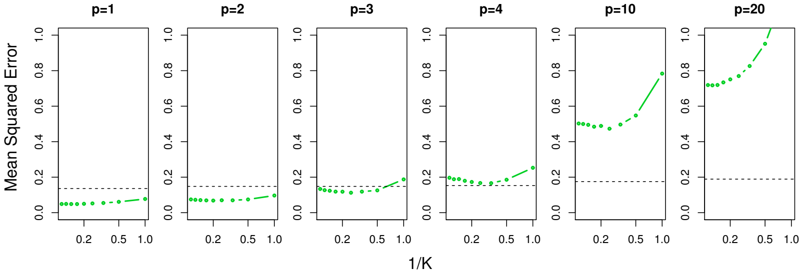
Non-linear: KNN beats regression

Note use of $1/K$ as "flexibility" axis: small $K \Rightarrow$ more flexibility

KNN vs. linear regression: Round 3

True function = function of x_1 only, with additional irrelevant predictors.

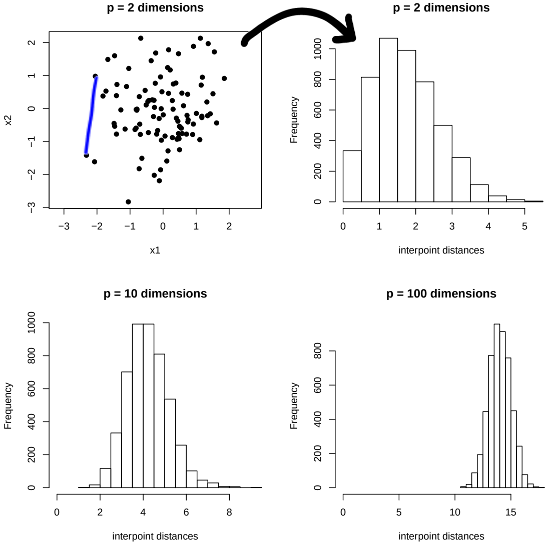
Below: MSE vs. flexibility (1/k) as dimension p increases.



KNN fails with many irrelevant predictors.

... This is the **curse of dimensionality**.

The curse of dimensionality



- Simulate independent N(0,1) data in p dimensions.
- Calculate all interpoint distances.
- In high dimensions, all points are far apart.

KNN vs. linear regression

Remember the basic model

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

Linear regression:

- Makes strong assumptions about $f(x)$: linearity, additivity
- Also assumes a probability model for error ε.
- Has "flexibility parameter(s)" (e.g., polynomial degree)

KNN:

- Makes no assumptions about $f(x)$ or error ε.
- Has a "flexibility parameter" (k neighbours).

Choosing model flexibility

What model is best? What flexibility parameter to choose? It depends on...

- True function $f(x)$
- Noise level
- Training set sample size
- Dimensionality of the input space
- ...

How do you choose?

- Our "test set" in examples was stylized
  - Shouldn't extra observations be used to train the model?
- Related and more realistic approach: Cross-validation.
- For models that make stronger assumptions, inferential methods are available.

Interlude

Before discussing cross-validation, I'll answer the unasked question:

Hugh, have you no shame? 50 points with a single predictor is not "big data" or "statistical learning"! And I think I learned KNN in preschool!

Maybe not, but:

- The bias-variance trade-off is central to statistical learning
- Most models use some combination of strong assumptions (linear model) and local modelling (knn)
- Can I send my kids to your preschool?
- By the way, I lied about using "polynomial regression". Smoothing splines were actually used.

Outline

Introduction

Cartoon examples of supervised learning

Cross-validation and computational methods for inference

Classification

Big ideas in statistical learning

Some thoughts

Cross-Validation

A problem with the "test set" idea described earlier: It's wasteful to not use all your data to train a model.

Idea #1: Train on 80%, test on 20%

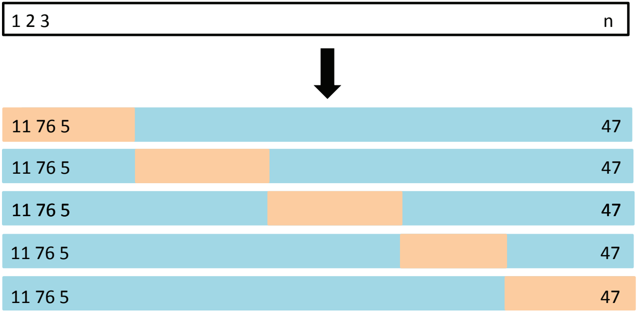
- 80% of the data will resemble the full dataset.

Another problem: Randomness of data splitting and small test set leads to noisy results.

Idea #2: Repeat idea #1, for different splits of the data.

- Repetition reduces variation due to random splitting.
- This is 5-fold cross-validation.

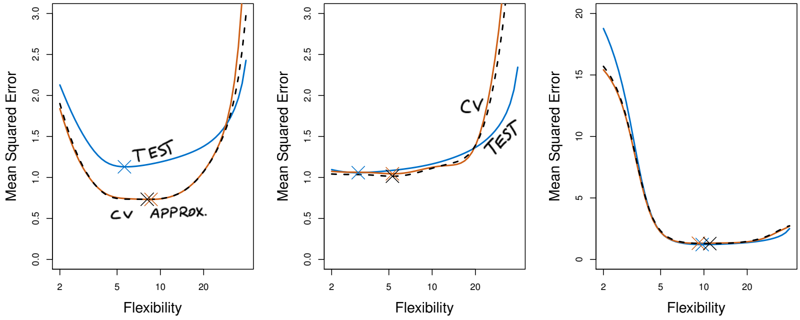
Picture of 5-fold CV



- White box = (sideways) data matrix with n observations.
- In each of 5 folds (coloured rows) ...
  - Train on blue 80%
  - Test on beige 20%
- ... Then average the results over the 5 "folds".
- ... Once you've chosen your flexibility parameter (e.g. k in KNN), use 100% of the data to retrain and make predictions.

Cross-Validation approximates the test error

- The actual test error can only be known with an infinite number of test observations.
- CV approximates this.
- For the 1-dimensional polynomial regression problems, the CV curve is a decent approximation to the true (blue) curve.



But what about statistical inference?

Remember the basic model

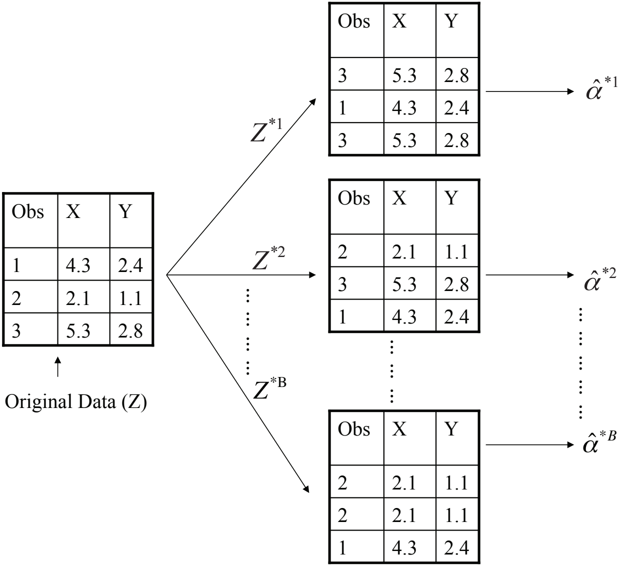
$$y = f(x) + \varepsilon$$

- CV helps us find a good estimate of $f(x)$.
- But all we get is a **point estimate**. We don't get uncertainty (e.g. prediction intervals).
- Inferential methods in Statistics can effectively provide uncertainty quantification.
- Easiest for simple models, in which parameter estimates are linear functions of the data (e.g. linear regression).

Inference for complex models: Bootstrap

- (Freqentist) Inference: Under repeated sampling of training sets from the population, how does my estimator behave?
- If we could sample multiple training sets, we could directly calculate an estimator's distribution.
- But we can't.
- **Bootstrap:** Pretend the training sample is the population. Resample with replacement a pseudo-training-sample of the same size, and apply your estimator to it. Repeat.

Inference for complex models: Bootstrap



Inference for complex models: Bootstrap

Big data: If we can't analyze the full data, how can we analyze hundreds of similar-sized bootstrap resamplings?

- "Bag of little bootstraps" by Kleiner, Talwalkar, Sarkar and Jordan (JRSS-B 2014)
- Approximates the bootstrap using faster computation (subsampling and reweighting).

Inference for complex models: Bayes

- Bayesian methods treat all unknown parameters as random variables.
- Convenient mechanism to quantify uncertainty for "tuning parameters", such as order of polynomial, k in KNN, etc.
- Posterior distributions combine data (likelihood) and prior belief, giving full inference.
- Computation typically carried out by simulation (Markov chain Monte Carlo, MCMC).
- MCMC makes it easy to compute inferential statements for *arbitrary* functions of parameters.
- As with the Bootstrap, big data is challenging (see "Consensus Bayes" talk by Steve Scott).

Outline

Introduction

Cartoon examples of supervised learning

Cross-validation and computational methods for inference

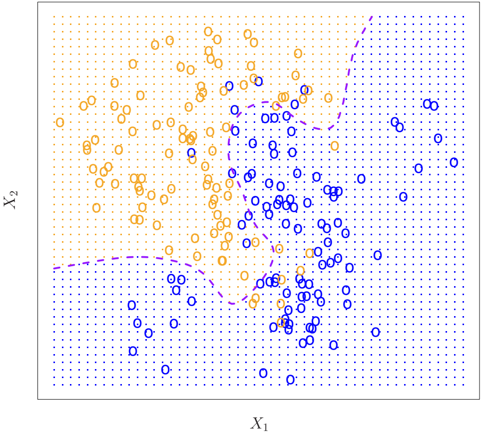
Classification

Big ideas in statistical learning

Some thoughts

Classification

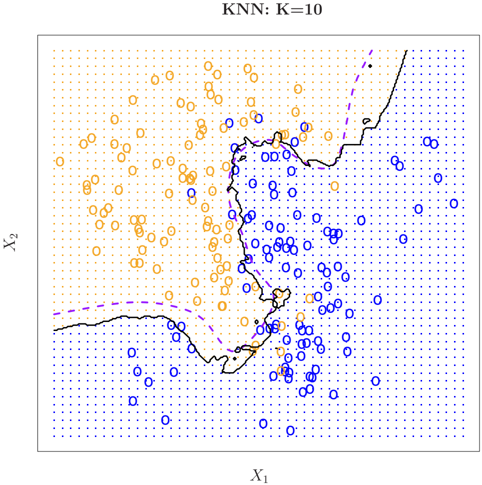
Y is a category (e.g. 2 categories - orange / purple).
Example with two-dimensional input $x = (x_1, x_2)$:



$\Pr(Y = \text{orange} \mid X)$ is a function like $f(x)$, and includes a random error model.

Classification

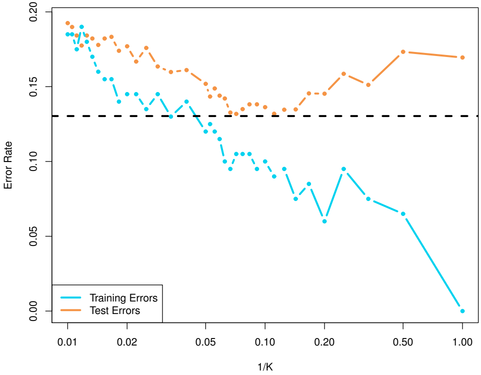
KNN with $K = 10$ does quite well:



Classification

A test set or CV can be used to choose flexibility (e.g. K).

- Similar bias/variance issues.



Outline

Introduction

Cartoon examples of supervised learning

Cross-validation and computational methods for inference

Classification

Big ideas in statistical learning

Some thoughts

Big ideas: additive models

Strong assumption of linear regression: Effect of varying x_1 does not depend on value of other x's.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_p x_p + \varepsilon$$

Generalize to have additive model with univariate functions:

$$Y = \beta_0 + g_1(x_1) + g_2(x_2) + \ldots g_p(x_p) + \varepsilon$$

- Retains ease of interpretation.
- Estimation of p separate univariate functions much easier than estimation of a single $f(x_1, x_2, \ldots, x_p)$.
- Extension: allow some low-order interactions

Big ideas: variable selection

With many predictors, we may expect many $\beta_j = 0$. But which ones?

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_p x_p + \varepsilon$$

Replace usual least squares criterion

$$\text{minimize } \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \text{ over } \beta_0 \ldots, \beta_p$$

with a penalized version (Lasso, Tibshirani 1996)

regularization

$$\text{minimize } \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \text{ over } \beta_0 \ldots, \beta_p$$

Second term constrains β's to be small or zero.
See Richard Lockhart's talk on inference....

Big ideas: dimension reduction

(Adam Kalai: "Choose a representation")

$$y = f(g(x)) + \varepsilon$$

- The function g maps a high-dimensional input vector x to a lower-dimensional space.
- What's the point? Isn't $f(g(x))$ just another function $h(x)$?
  - Idea is to estimate g without over-training.
- Principal component analysis seeks projections $\alpha_1^T x, \alpha_2^T x, \ldots$ with maximal variance. These are estimated without using Y (i.e. **unsupervised learning**).
- Example: digit recognition x_1 = intensity of (1,1) pixel of image, etc. Functions $g(x)$ of the pixels should capture structure of the handwritten digits.

- Similar approach in "deep learning": estimate functions of inputs without using the response until the final learning step.

Big ideas: neural nets

Nonlinear models with linear regressions at their core...

They have the functional form

$$f(x) = \Psi\left[\alpha_0 + \sum_i \alpha_i \Phi(\beta_{i0} + \sum_j \beta_{ij} x_j)\right]$$

with Ψ, Φ known, nonlinear functions.

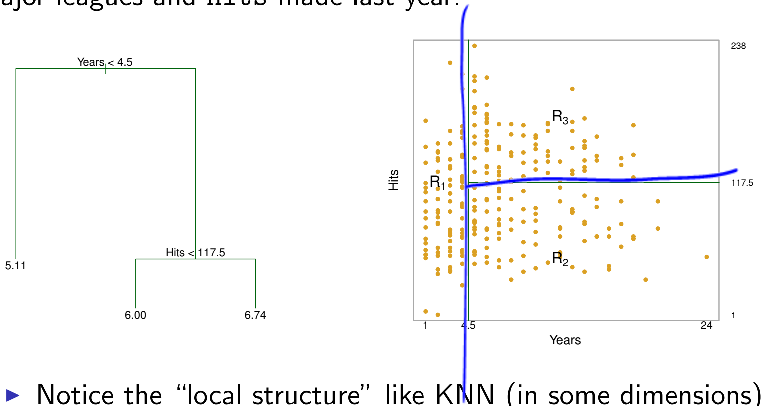
- We seek to estimate the coefficients (β's and α's).
- Nonlinear regression with many parameters.

A linear combination of...
A nonlinear transformation of ...
A linear combination of ...
the original variables

Big ideas: decision trees

Recursively partition the X space into rectangular regions.

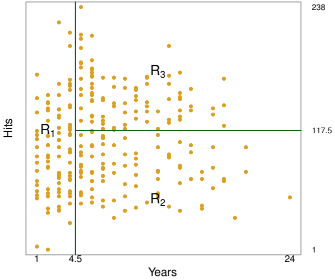
Example: Predict (log) `Salary` of baseball player, given `Years` in major leagues and `Hits` made last year.



- Notice the "local structure" like KNN (in some dimensions).
- We must learn the tree topology (variables used, split values, etc) and outputs from training data.

Big ideas: decision trees

Decision trees are interpretable, flexible, good at detecting interactions and automatically select variables.



But they're sensitive to noise and terrible at representing additive structure (try fitting $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ with a tree).

Big ideas: ensemble models

Not just trees (Robert Bell - blending)

Overcome the limitations of a single tree by fitting a "sum of trees" model.

- Let $(T_1, M_1), \ldots, (T_m, M_m)$ identify a set of m trees and their terminal node μ's.

$$Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \varepsilon$$

- For an input value x, each $g(x; T_i, M_i)$ outputs a corresponding μ
- The prediction is the sum of the μ's
- Random Forests (Breiman 2001) and Boosting (Freund & Schapire 1997) are two algorithms for building this model.

Big ideas: ensemble models

Breiman's **random forests** (2001) use randomized search and the bootstrap to perturb individual trees.

- Uses noise sensitivity of trees to build a stable model.

Freund and Schapire's **boosting algorithm** (1997) encourages each tree to fit structure not captured by the other trees.

- Enables an additive model to be fit.
- Friedman (2001) presents a more statistically motivated boosting algorithm.

The model

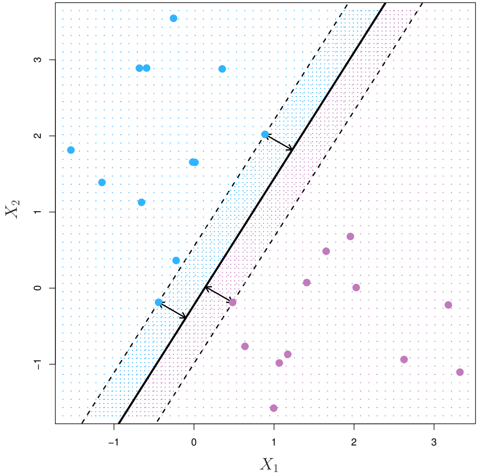
$$Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \varepsilon$$

also forms the basis for Bayesian Additive Regression Trees (BART; Chipman George and McCulloch 2010).

- Full Bayesian inference + extensible error models.

Big ideas: support vector machines

Originated as a 2-class classification problem (Vapnik, 1996).
Approach: find a hyperplane that separates the input space into two regions, maximally separating two classes.



Big ideas: support vector machines

Two other key ideas:

1. Allow some misclassifications (amount is a tuning parameter).
2. Transform input vector X into a higher-dimensional space where a hyperplane is more likely to separate classes (often a parametrized transformation).

Comments on point 2:

- A "kernel trick" avoids the need to actually compute the high-dimensional mapping.
- Expensive algorithm - $O(n^2)$ for n observations.

SVM is one of many **Kernel methods** for learning.

Outline

Introduction

Cartoon examples of supervised learning

Cross-validation and computational methods for inference

Classification

Big ideas in statistical learning

Some thoughts

Some thoughts

Rich error distributions: A soon-to-be big idea?

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

We've focused mostly on estimating $f(x)$.
"Traditional" statistics puts more into the error model:

- time series and spatial data have correlated errors
- mixed models have multilevel error structure, including longitudinal data
- survey sampling has variances induced by the sampling plan

Some thoughts

Uncertainty quantification
Michael Jordan: "We have to have error bars around all our predictions. That is something that's missing in much of the current machine learning literature. "

Huh? With big data, won't all your error bars be 0?

Not necessarily:

- Complexity often grows with sample size: with thousands of variables, there will still be uncertainty.
- As large samples drive down sampling variation, other source of sample error gain prominence: biased sampling, correlated errors, etc.

Some thoughts: Summary

Key ideas:

- Bias/variance trade-off
- Cross-validation to choose flexibility
- Inference is possible (and under-appreciated)
- Fancy methods try to introduce assumptions in a way that they're flexible:
  - variable selection / dimension reduction
  - additivity and low-dimensional functions
  - transformations
- There's a lot of room to insert statistical thinking into statistical and machine learning.