

Machine Learning in R

Lecture 1 – Introduction

John Kornak – Epidemiology and Biostatistics

john.kornak@ucsf.edu

Logistics

- Lectures every Thursday, 1.00-2.30 (Mission Hall Room 1407: 4/4, 4/11, 4/18, 5/16, 5/23, 5/30 and 6/6; Genentech North 114: 4/25, 5/2, 5/9)
- Labs every Thursday in same room as lecture, 2.45-3.45
- Structured discussion sessions: 4/11, 4/25, 5/9, and 5/23, 3.45-4.45
- Lectures are to be recorded and posted online
- Grades based on 5 hw assignments (70%) + data analysis project (30%, due 5/30/19)
- Check materials online at CLE (and Biostat 216 discussion forum)

Data Analysis Project

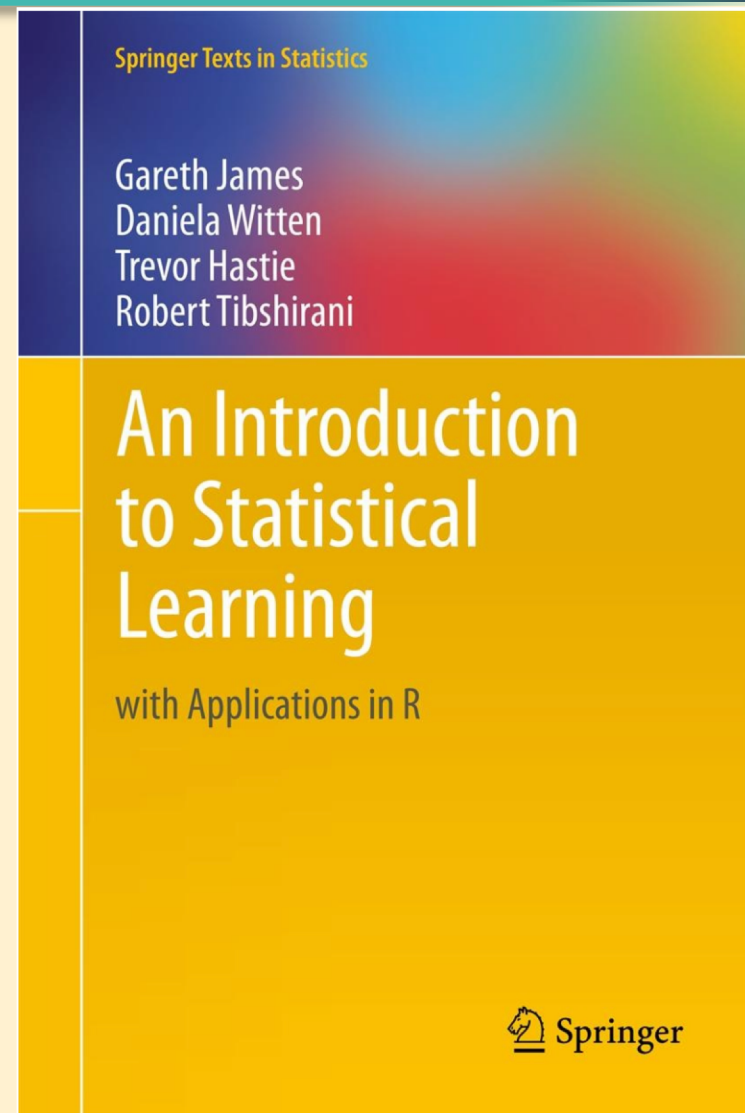
- You will need to perform a data analysis project as part of the course (30% contribution to grade)
- You will be provided with datasets and suggestions as to how to proceed
- However, if you are particularly confident you have the option to work with your own dataset

R, RStudio, and R packages

- You need to have R installed on your laptop: www.cran.org - current version is R 3.5.3.
- I also recommend that you install RStudio and run R from within Rstudio: www.rstudio.com/products/RStudio/
- We will be installing multiple machine learning packages for R and we will do that as we go along. This is easy to do with the `install.packages("package_name")` command in R
- Try to install a package, e.g. the lme4 package with `install.packages("lme4")` to make sure that you don't have any problems

Texts

- Parts of this course borrow from *An Introduction to Statistical Learning*, and associated lectures. James et al., available for free at: <https://www-bcf.usc.edu/~gareth/ISL/>
- Also useful is the more technical *Elements of Statistical Learning*: <http://web.stanford.edu/~hastie/ElemStatLearn/> (again, available free)
- There is a myriad of Machine Learning in R books (of varying quality), many of which are available free via UCSF electronic library.



TICR Professional Conduct Statement – clarifications for this class

- I will maintain the highest standards of academic honesty
- I will not use answer keys from prior years
- Problem Sets and Projects: I am permitted to consult with other classmates, but I will write final answers in my own words away from other classmates

Course objectives

- Generate intuition and mechanistic understanding for a set of important machine learning methods
- Gain insight as to which methods are most appropriate in which situations
- Learn best practices in machine learning
- Discover potential pitfalls to avoid in machine learning
- Enable practical implementation of machine learning methods covered in R

Topics covered in this course

- What is machine learning: supervised vs. unsupervised learning
- Model choice, evaluation and comparison
- Supervised machine learning methods
 - Support vector machines
 - Classification and regression trees / random forests
 - Boosting and gradient boosting
 - Penalized regression
- Unsupervised machine learning methods
 - Clustering algorithms
 - Dimension reduction algorithms
- Applications in biomedical problems
- `rtemis` meta-package

In this lecture:

- What is Machine Learning?
- Supervised vs. Unsupervised learning
- Introduction to prediction techniques for supervised learning with continuous outcomes:
 - Local estimation and nearest neighbors
 - Loss functions and mean square error
 - The training-test paradigm
 - The curse of dimensionality
 - Linear regression modeling (including polynomials)
 - Some trade-offs

Machine Learning Definitions

- *No well accepted definition*
- *Machine learning* is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.
- In 1959, Arthur Samuel defined machine learning as a “Field of study that gives computers the ability to learn without being explicitly programmed”
- Tom Mitchell in 1998 defined Machine Learning via “A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E .”
- *Machine learning* is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.
- Focus on prediction or identifying patterns in data.
- Machine learning problems broadly split into **Supervised** and **Unsupervised** learning problems
- Note that the term “Statistical Learning” is used in ISLR – difficult to define a clear difference

Machine learning problems in biomedicine

- Predict disease-free survival times of patients breast cancer surgery based on diagnostic measures, demographic variables and genetic information
- Determine whether there are distinct subgroups of patients with anxiety and depression that may have different patterns of progression
- Automatically label patients as different types of dementia based on neuropsychological testing scores

Supervised vs. Unsupervised Learning

- **Supervised Learning**

- Have data where you know outcome in order to *train* the model
- Train the model to predict future outcomes from sets of predictors

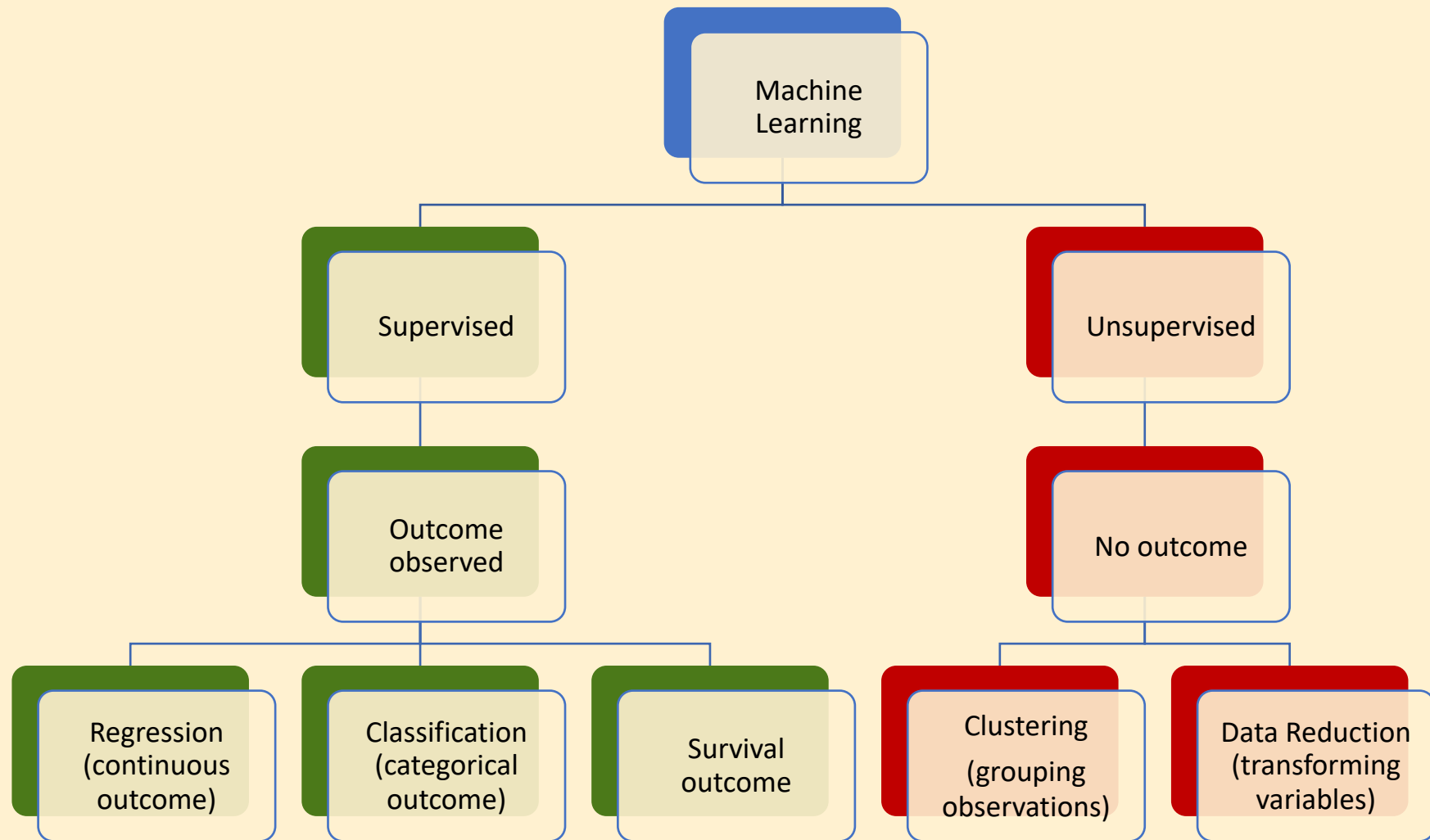
Prediction: Regression (continuous outcomes) and classification (categorical outcomes)

- **Unsupervised Learning**

- No specified outcomes
- Separate data points into groups with “similar” properties (clustering)
- Or reduce data to fewer variables in some way that still captures important structure of the data

Clustering and dimension reduction

Supervised vs. unsupervised



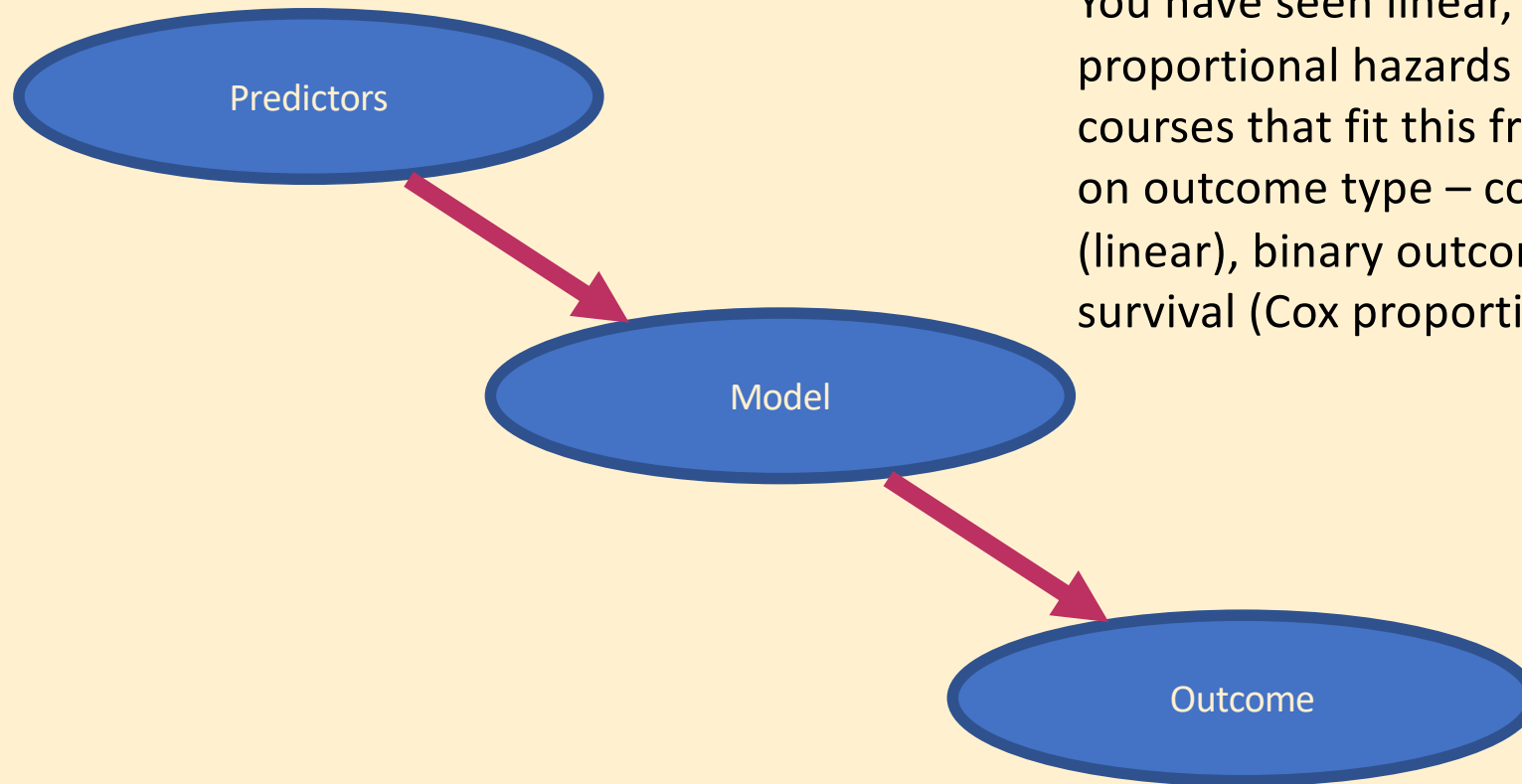
Machine learning problems in biomedicine

- Predict disease-free survival times of patients breast cancer surgery based on diagnostic measures, demographic variables and genetic information
- Determine whether there are distinct subgroups of patients with anxiety and depression that may have different patterns of progression
- Automatically label patients as different types of dementia based on neuropsychological testing scores

Machine learning problems in biomedicine

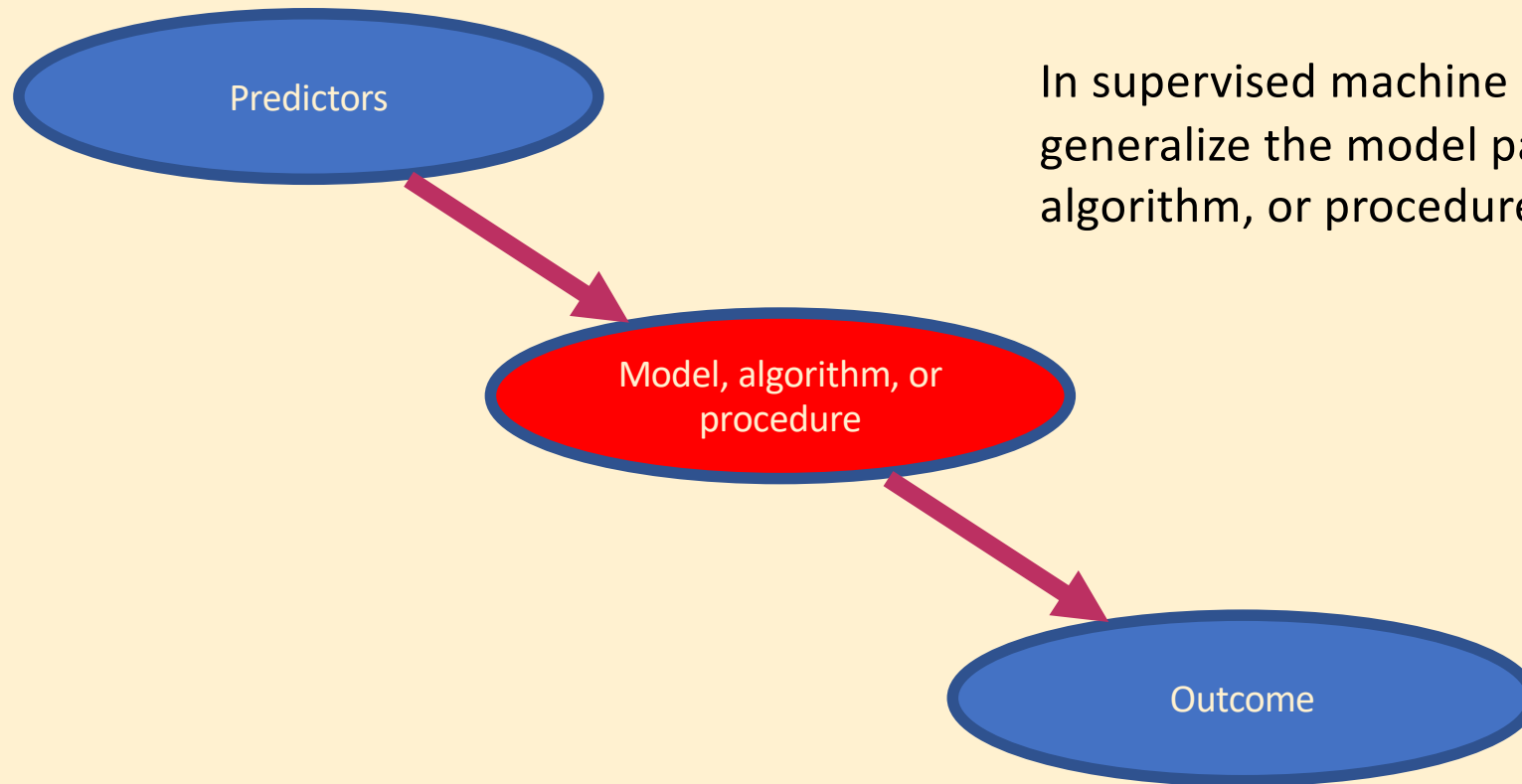
- Predict disease-free survival times of patients breast cancer surgery based on diagnostic measures, demographic variables and genetic information (supervised)
- Determine whether there are distinct subgroups of patients with anxiety and depression that may have different patterns of progression (unsupervised)
- Automatically label patients as different types of dementia based on neuropsychological testing scores (supervised and unsupervised)

Supervised Regression -- Predictive Models



You have seen linear, logistic and Cox proportional hazards models in previous courses that fit this framework depending on outcome type – continuous outcome (linear), binary outcome (logistic), survival (Cox proportional hazards)

General supervised learning



In supervised machine learning we generalize the model part to “model, algorithm, or procedure”

Supervised prediction

- *Predictors, X* (also called *inputs, regressors, covariates, features, or independent variables*)
- *Response, Y* that we want to predict (also called *outcome, target, or dependent variable*)
- Model/algorithm:

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

- where ε is an error term
- In the regression problem, Y is quantitative
- In the classification problem, Y is a class variable/unordered categories
- In the survival problem, Y is a survival outcome (survival time and status censor/uncensored)
- Train the model with a training dataset: $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n) = \{(x_i, y_i)\}, i = 1, 2, \dots, n$

Supervised prediction objectives

Build a model/algorithm that can:

- Predict unknown future responses/outcomes
- Determine which predictors/inputs affect the outcome, and how
- Inform as to the quality of predictions and inferences

Unsupervised learning

- No responses, just a set of variables/predictors/features measured on a set of samples
- Objective can be:
 - Clustering: find groups for which predictors follow a more similar pattern (correlated) than between groups
 - Dimension reduction: find ways to reduce the information in a large set of predictors to something more manageable – ideally, such that you get new features that are interpretable

Back to supervised learning: notation

- Consider an outcome Y (e.g. blood pressure, or disease status)
- Consider predictors, e.g. X_1 (heart rate), X_2 (exercise yes/no), etc.
- Refer to the set of predictors as:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix}$$

p is the number
of predictors

- Then model is:

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

where ε is the “error” term

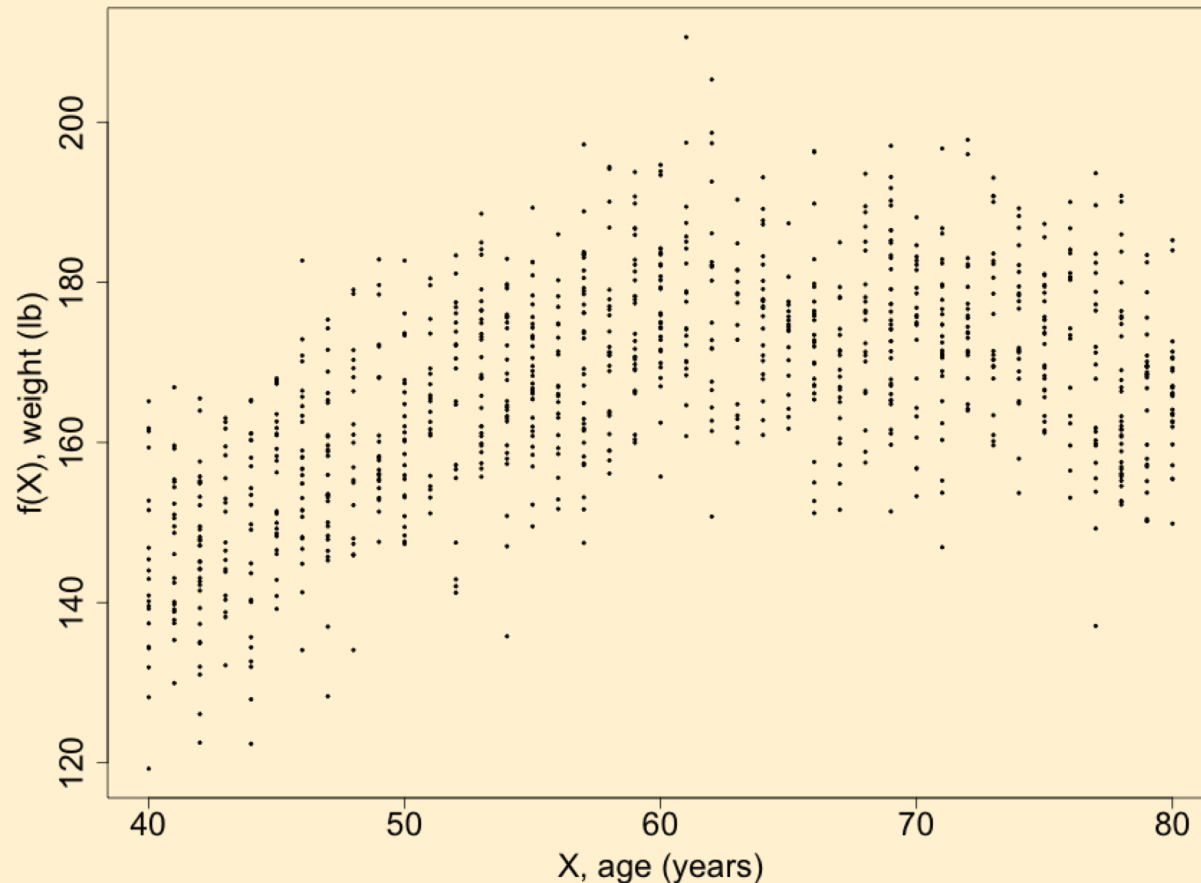
What do we do with $f(X)$?

- Make predictions of Y at new points $X = x$
- Determine which predictors in $X = (X_1, X_2, \dots, X_p)$ are important – e.g. recent exercise is important for explaining blood pressure, but number of years of education is not
- Depending on the complexity of $f(X)$ we can try to understand the relationship between each X_i of X and Y

What makes a good estimate of $f(X)$?

Consider supervised learning with a continuous outcome and a single predictor

**Training
data**

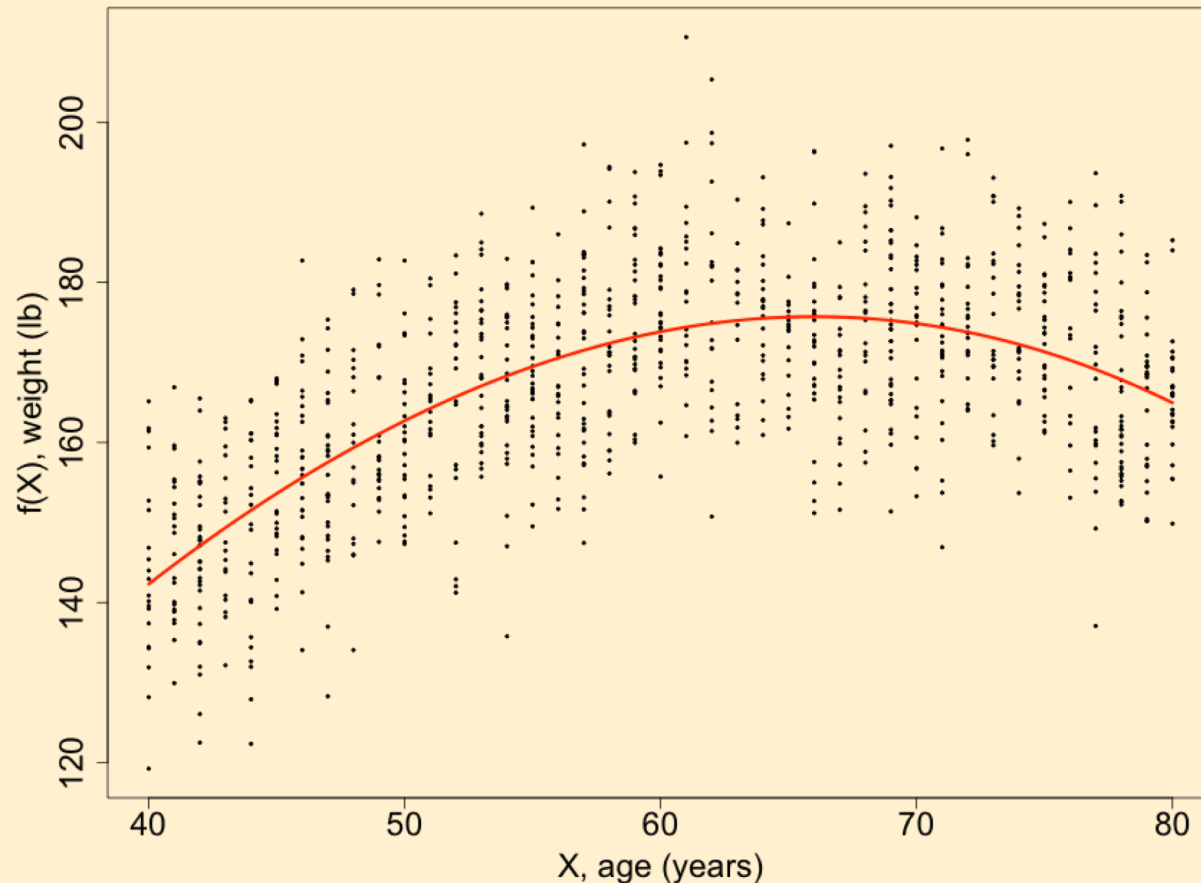


What makes a good estimate of $f(X)$?

Consider supervised learning with a continuous outcome and a single predictor

A good $f(X)$
represents mean
trend in the data well

Training
data



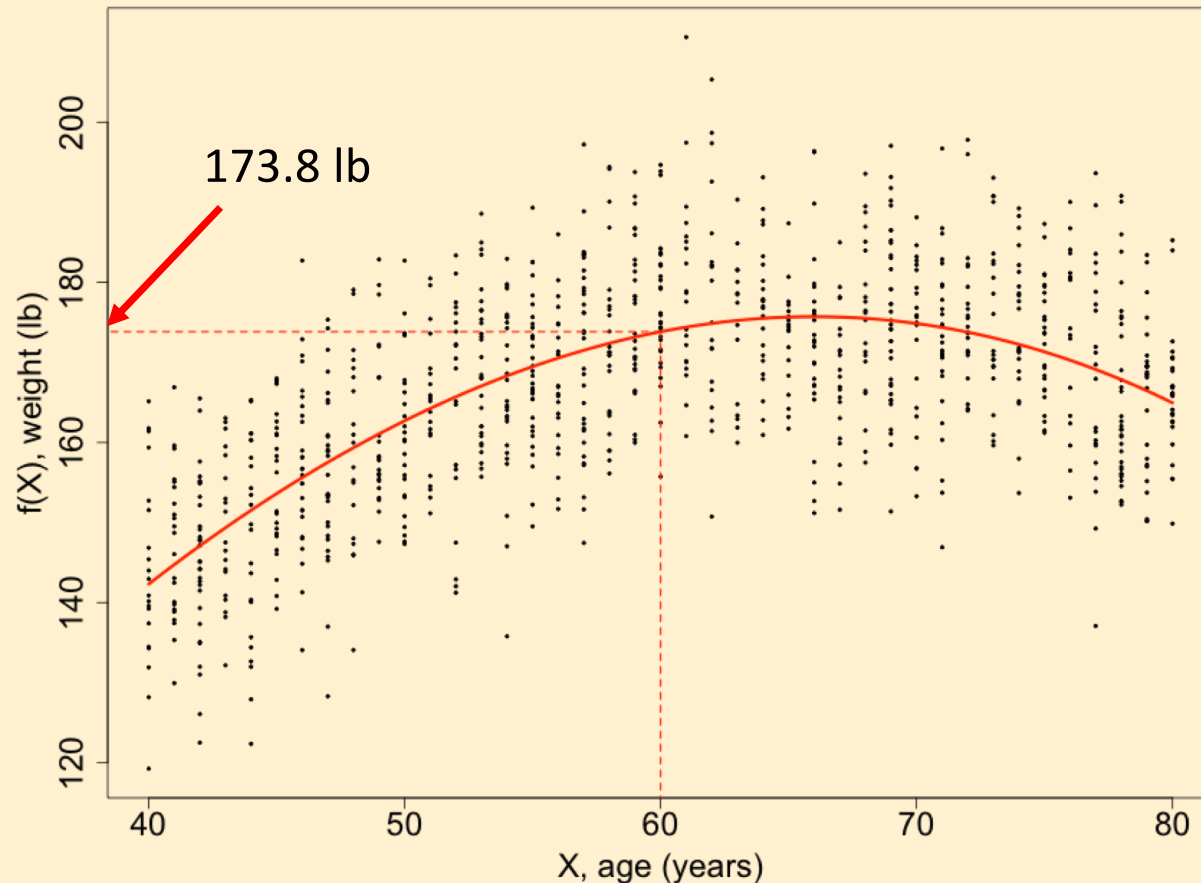
What makes a good estimate of $f(X)$?

Consider supervised learning with a continuous outcome and a single predictor

A good $f(X)$
represents mean
trend in the data well

**Training
data**

Does a good job
(intuitively) of
making a prediction



What makes a good estimate of $f(X)$?

- The ideal predictor for a continuous outcome is most often considered to be the function that minimizes mean-squared prediction error
- $f(X) = E(Y|X = x)$ is the *loss function* that minimizes the mean square error (MSE): $E[(Y - g(X))^2 | X = x]$ over all possible functions g over all points x
- $\varepsilon = Y - f(x)$ is the *irreducible* error – even if we have the best possible predictor, we cannot predict the random variation
- And so we have

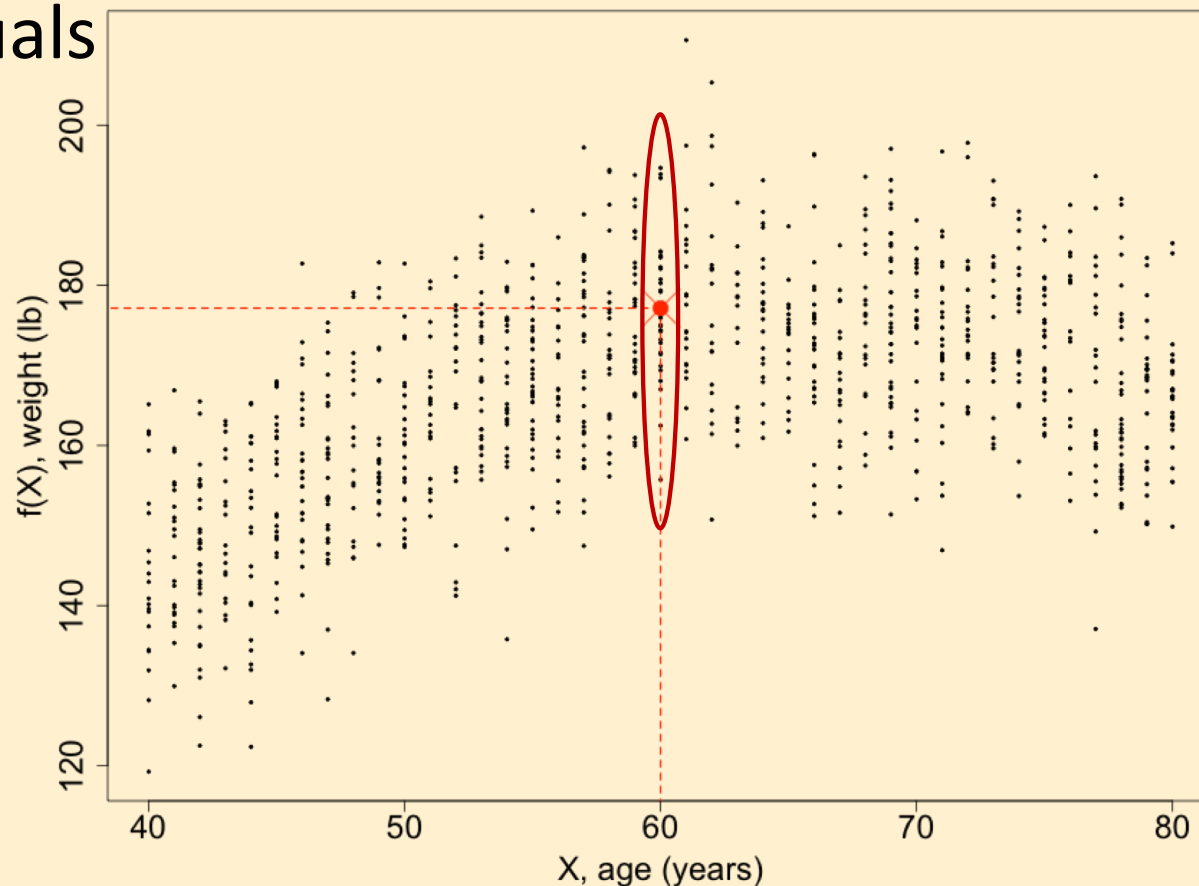
$$E[(Y - \hat{f}(X))^2 | X = x] = \underbrace{\left(f(x) - \hat{f}(x)\right)^2}_{\text{Reducible}} + \underbrace{\text{Var}(\varepsilon)}_{\text{Irreducible}}$$

$\hat{} = \text{estimate}$

What can we choose for $\hat{f}(X)$?

We could choose $\hat{f}(X) = E(Y|X = x)$ calculated at each age, e.g. estimate $\hat{f}(60)$ as the mean weight of age 60 individuals

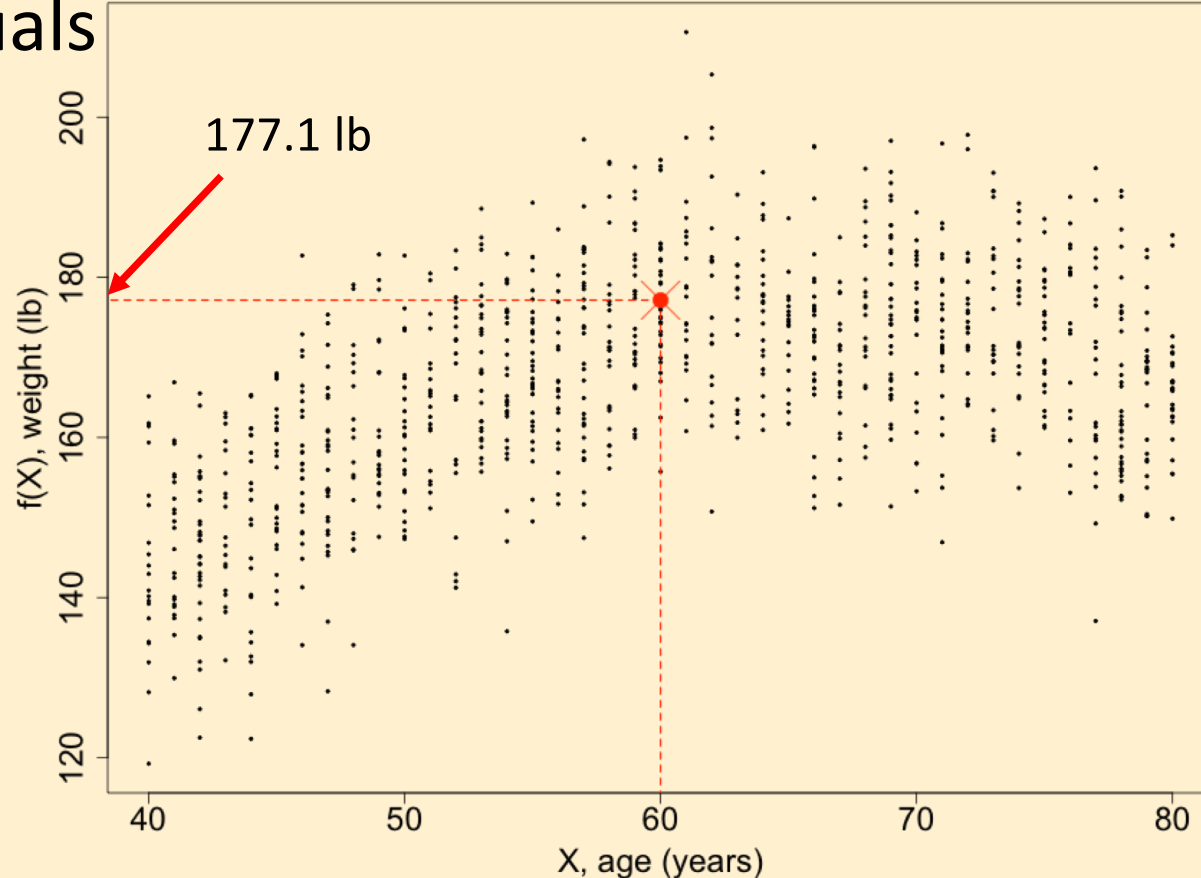
**Training
data**



What can we choose for $\hat{f}(X)$?

We could choose $\hat{f}(X) = E(Y|X = x)$ calculated at each age, e.g. estimate $\hat{f}(60)$ as the mean weight of age 60 individuals

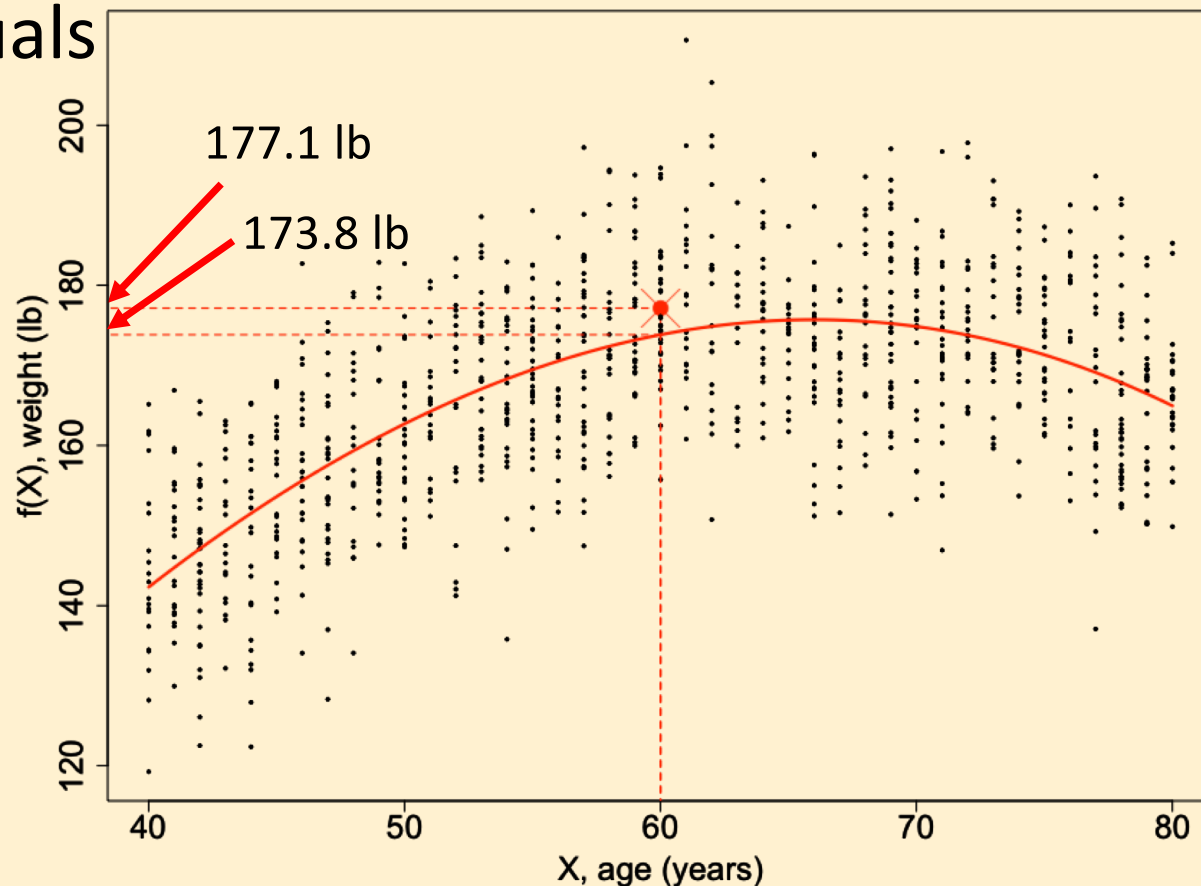
**Training
data**



What can we choose for $\hat{f}(X)$?

We could choose $\hat{f}(X) = E(Y|X = x)$ calculated at each age, e.g. estimate $\hat{f}(60)$ as the mean weight of age 60 individuals

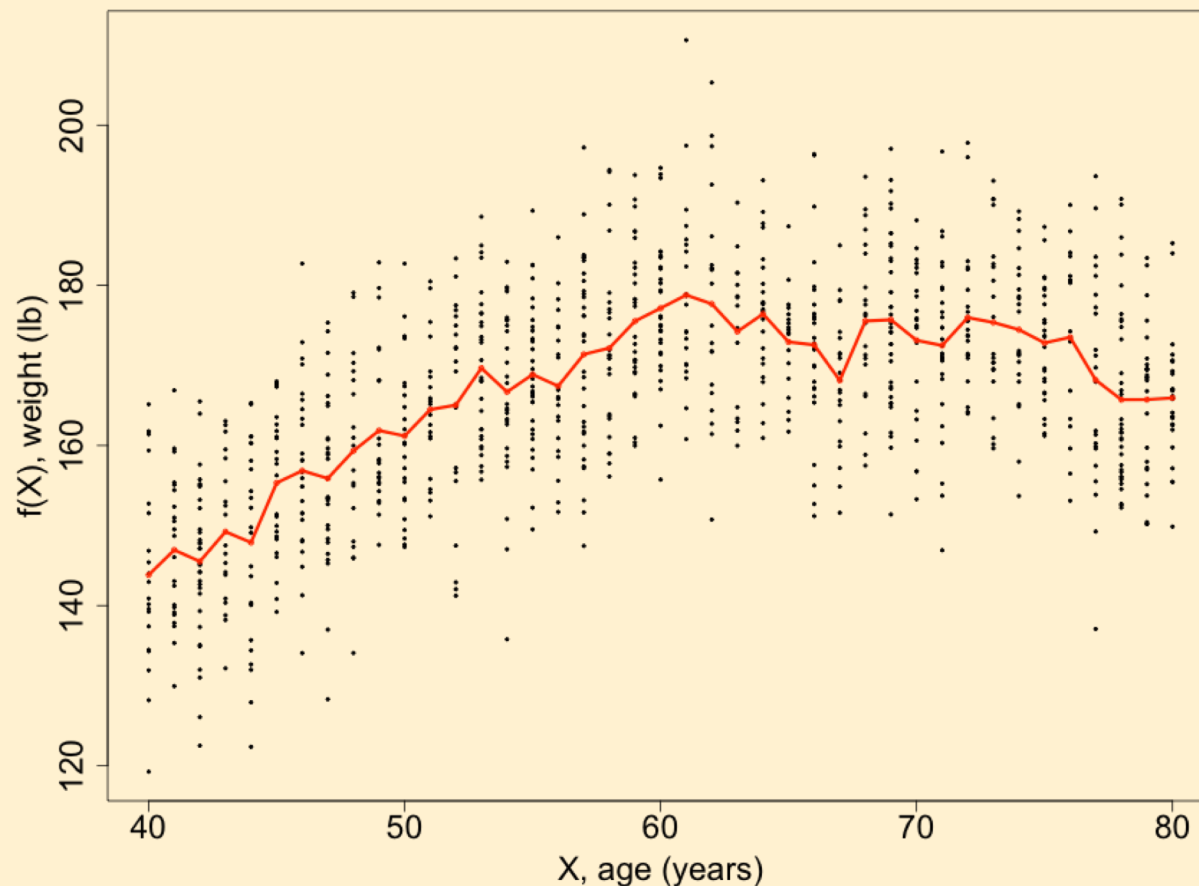
Training
data



Local expectation

So try $\hat{f}(X) = E(Y|X = x)$ at all ages...

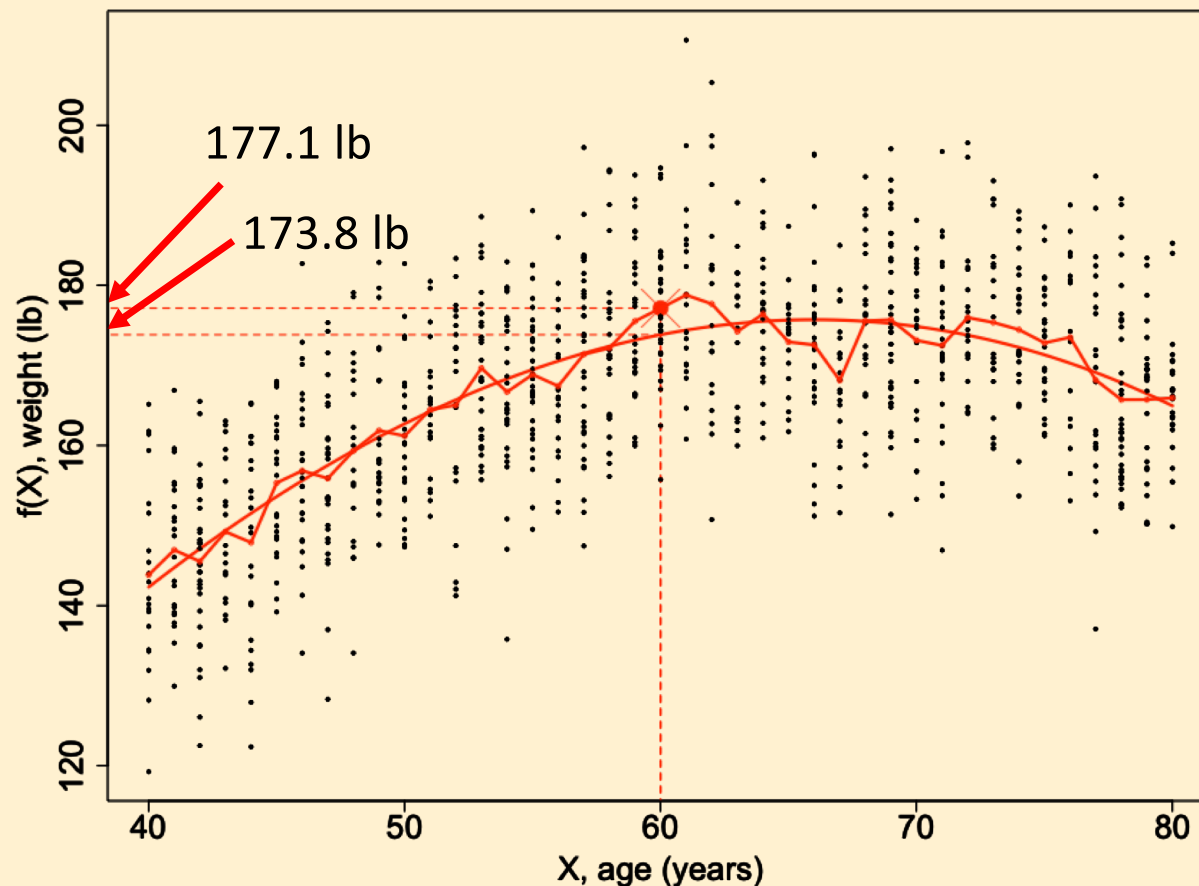
Training
data



What can we choose for $\hat{f}(X)$?

Tracks true curve pretty well, but “noisy”

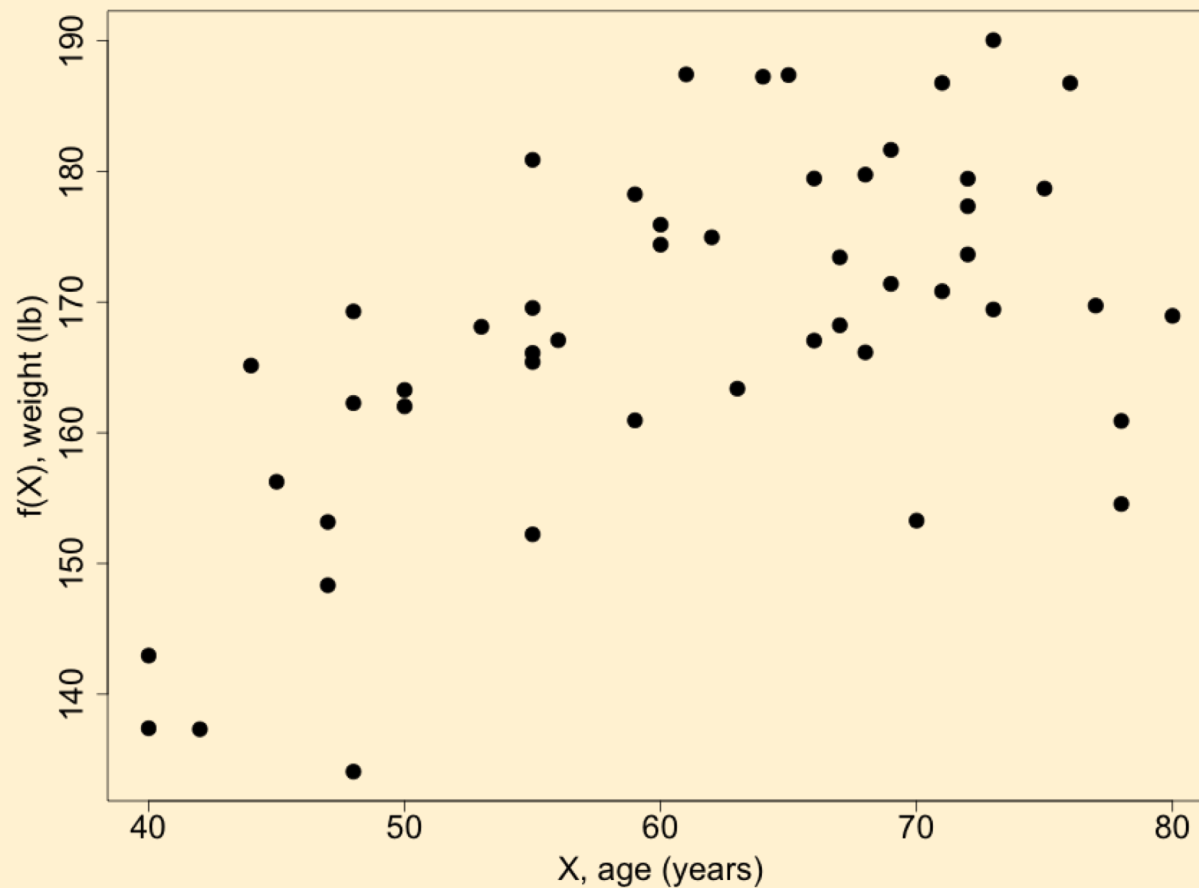
Training
data



Local expectation

And what if we have less data?

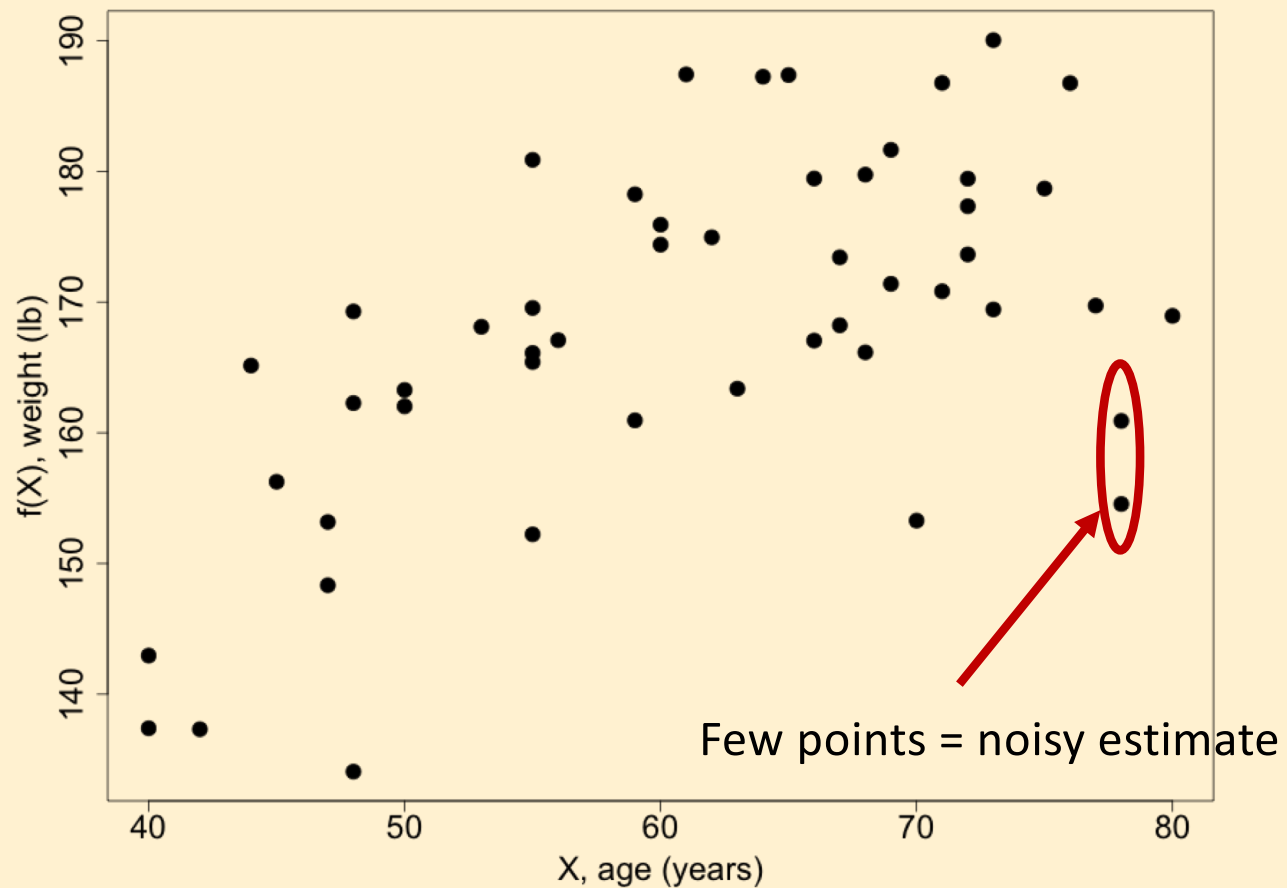
Training
data



Local expectation

But what if we have less data?

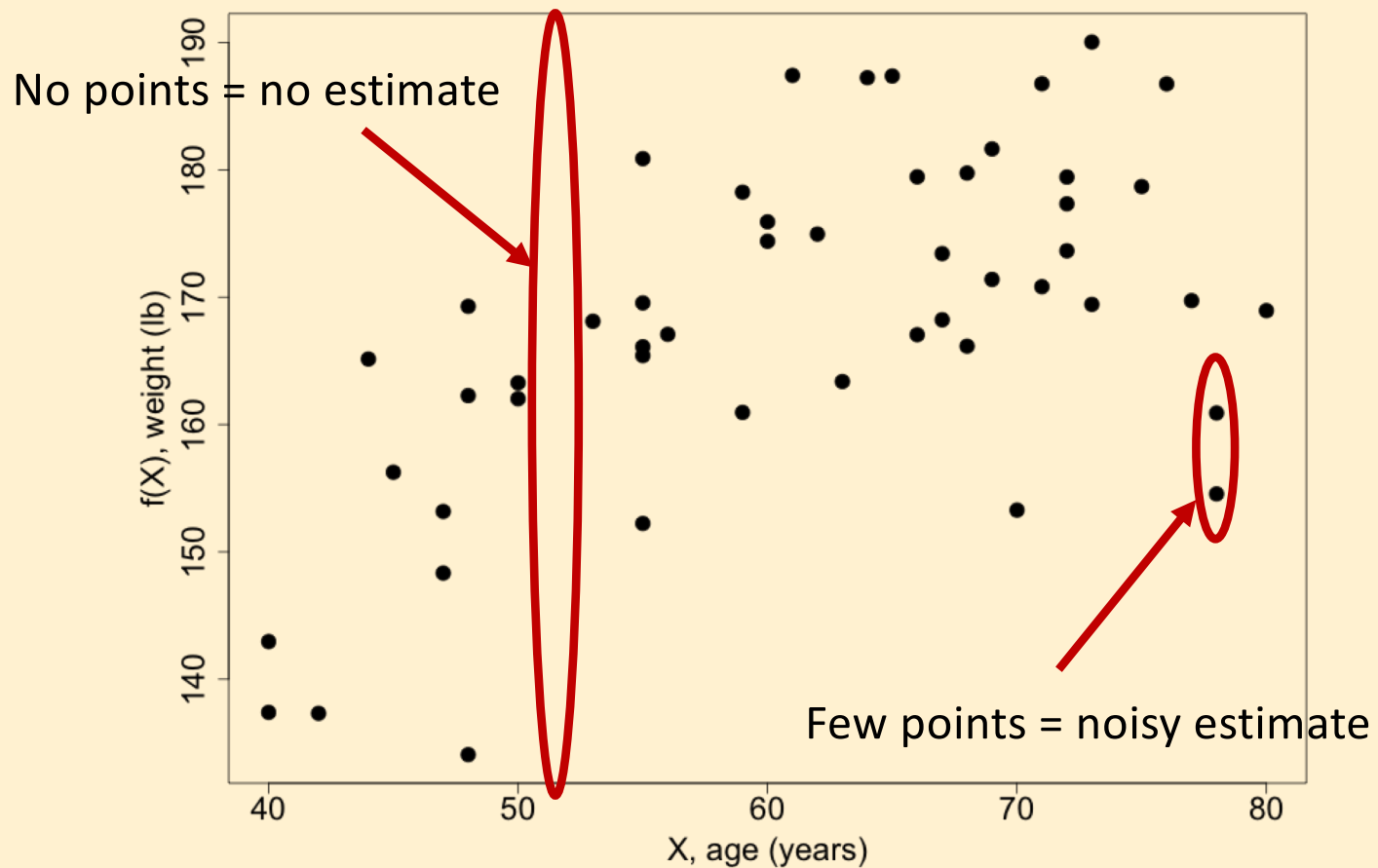
Training
data



Local expectation

But what if we have less data?

Training
data

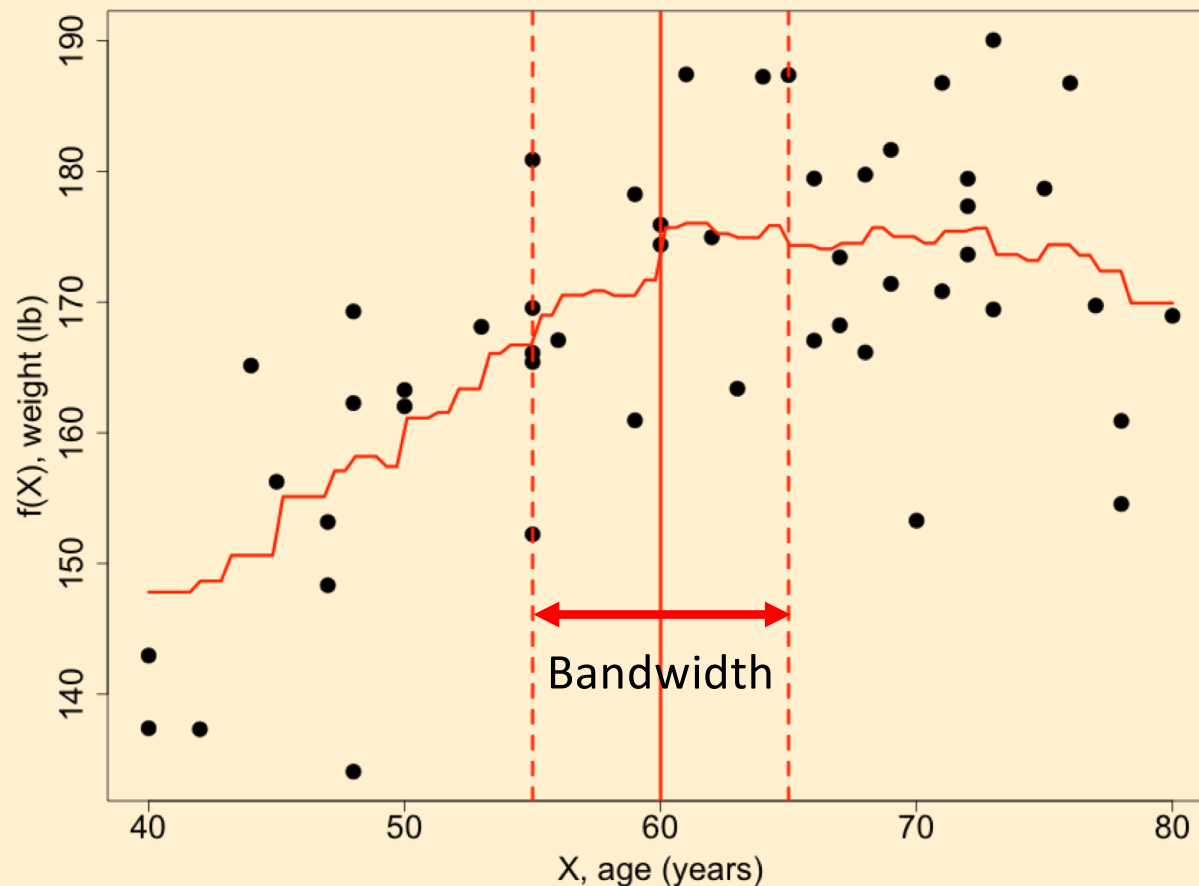


Kernel estimation/smoothing

We can “average” over a *neighborhood* of X ,...

```
ksmooth(x, y, kernel = "box", bandwidth = 10)
```

Training
data

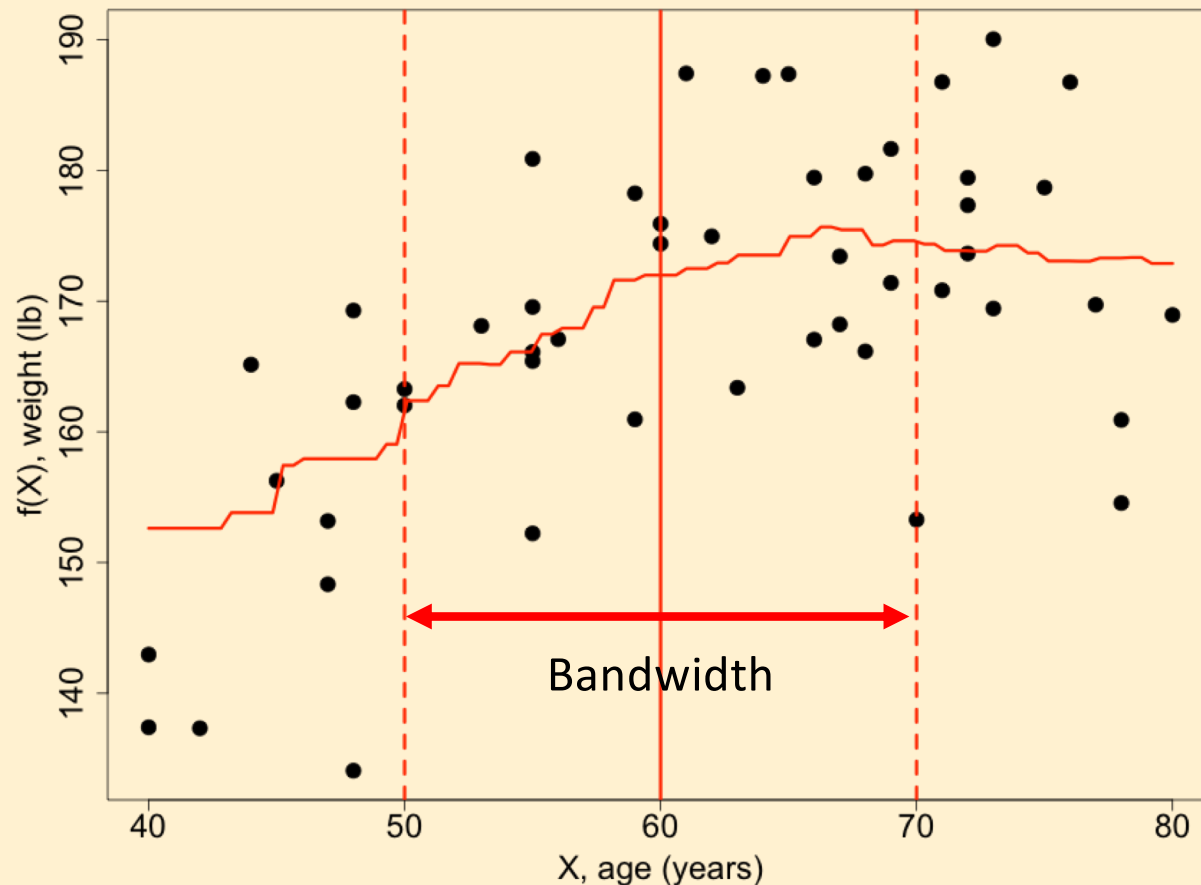


Kernel estimation

We can average over a *neighborhood* of X ,...

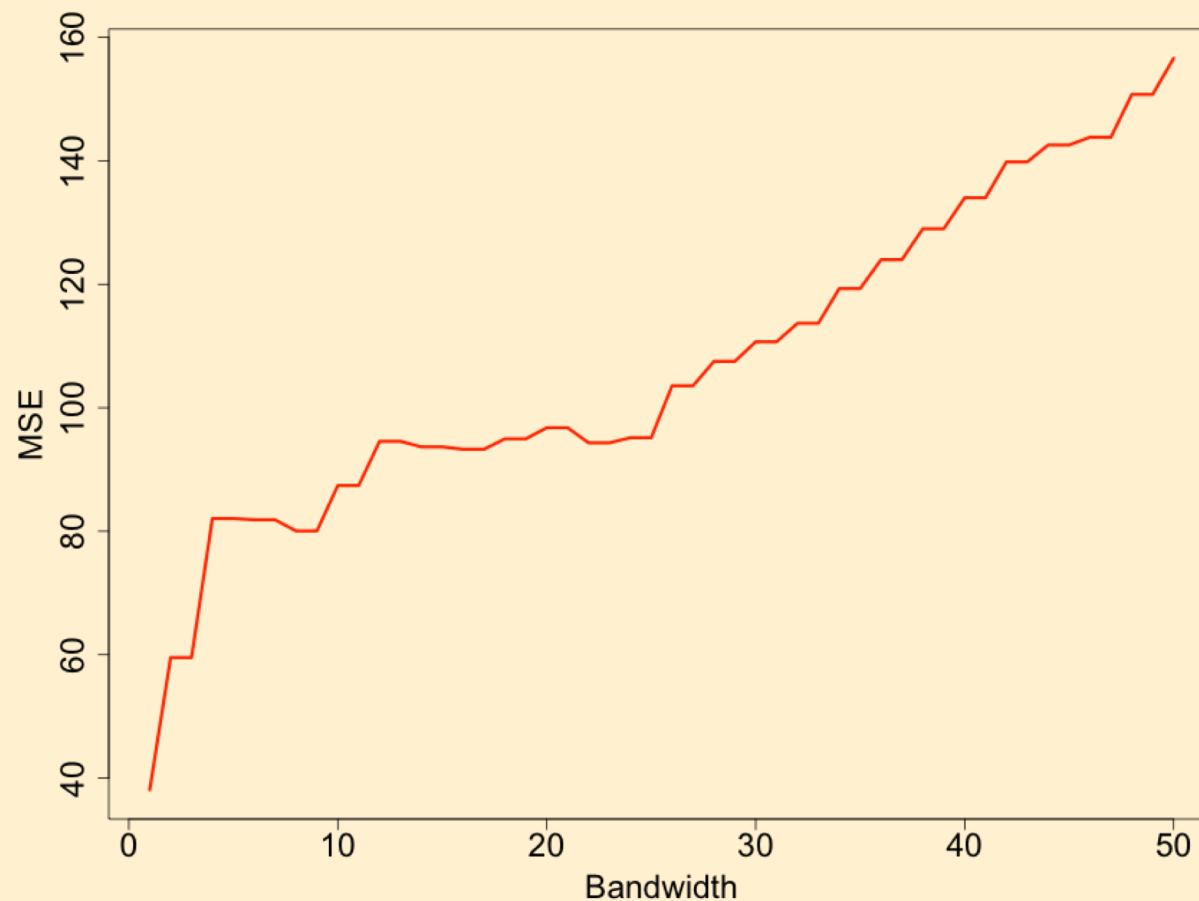
```
Ksmooth(x, y, kernel = "box", bandwidth = 20)
```

Training
data



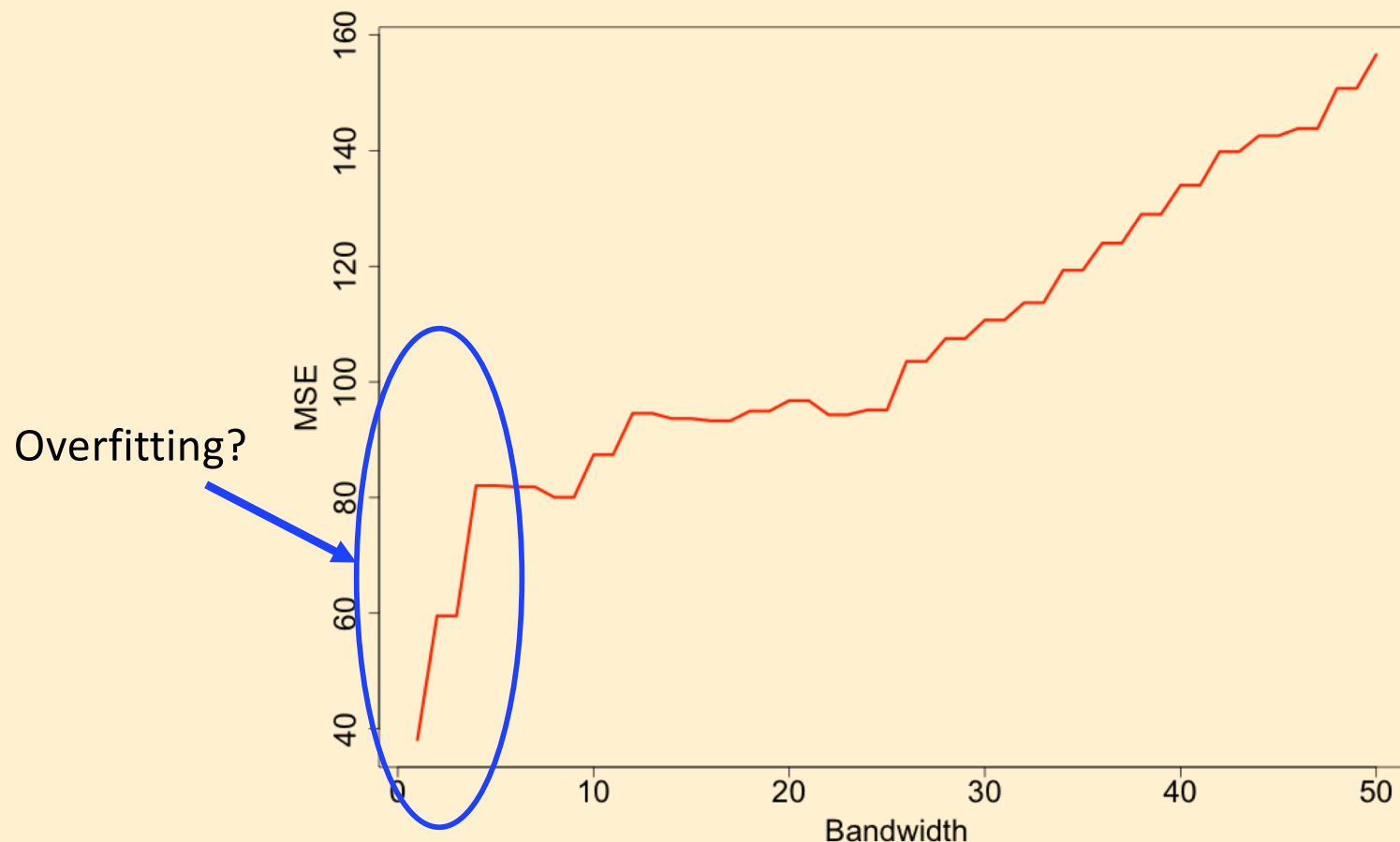
How to choose the bandwidth?

- We can try different values of the bandwidth and figure out which value gives us the smallest MSE: $E[(Y - g(X))^2 | X = x]$



How to choose the bandwidth?

- We can try different values of the bandwidth and figure out which value gives us the smallest MSE: $E[(Y - g(X))^2 | X = x]$

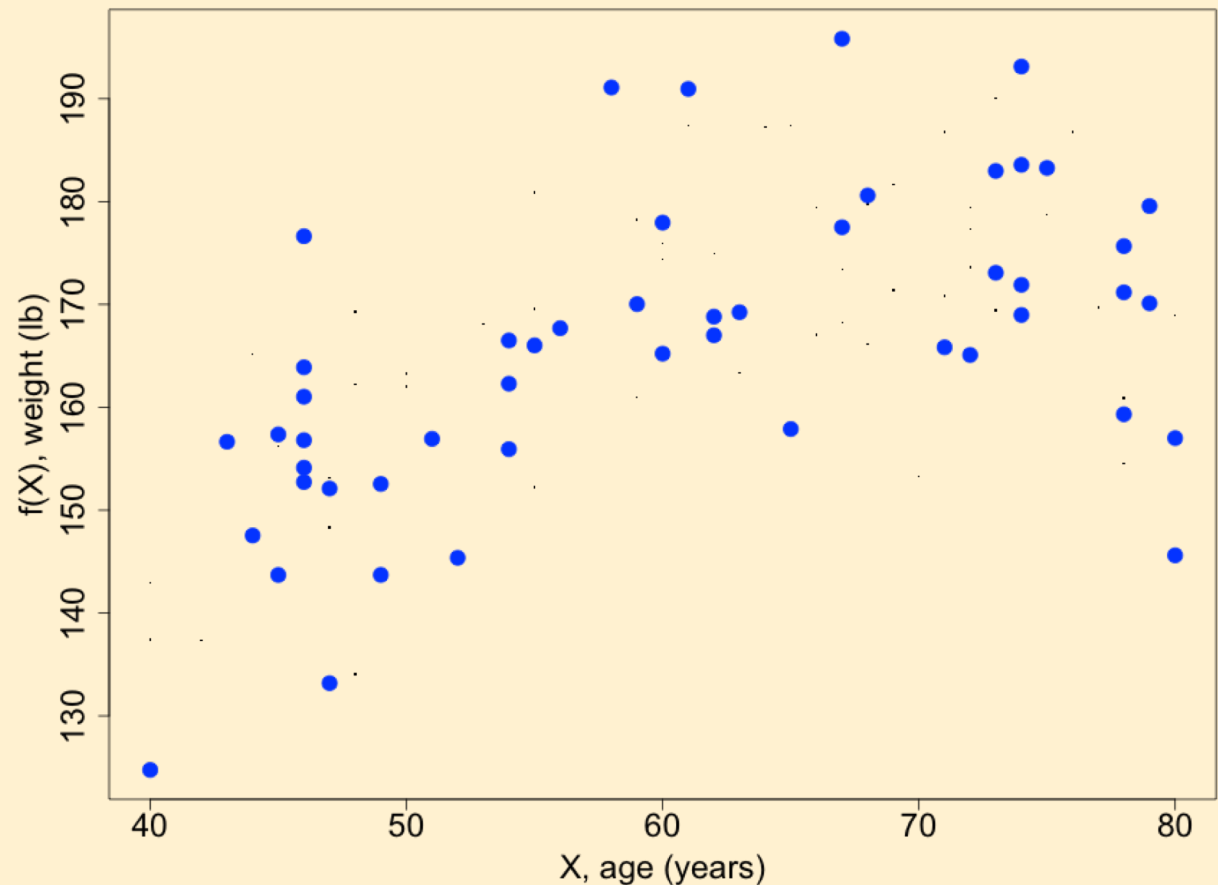


Evaluate on independent test set

- Overfitting occurs when examining the model with data used to fit the model – fits the *signal, $f(X)$* and also potentially the *noise, ϵ*
- Over-optimism due to overfitting is avoided when evaluating on a separate test dataset
- The test data were not involved in the fitting so it is not possible that the model/algorithm can overfit to their irreducible error

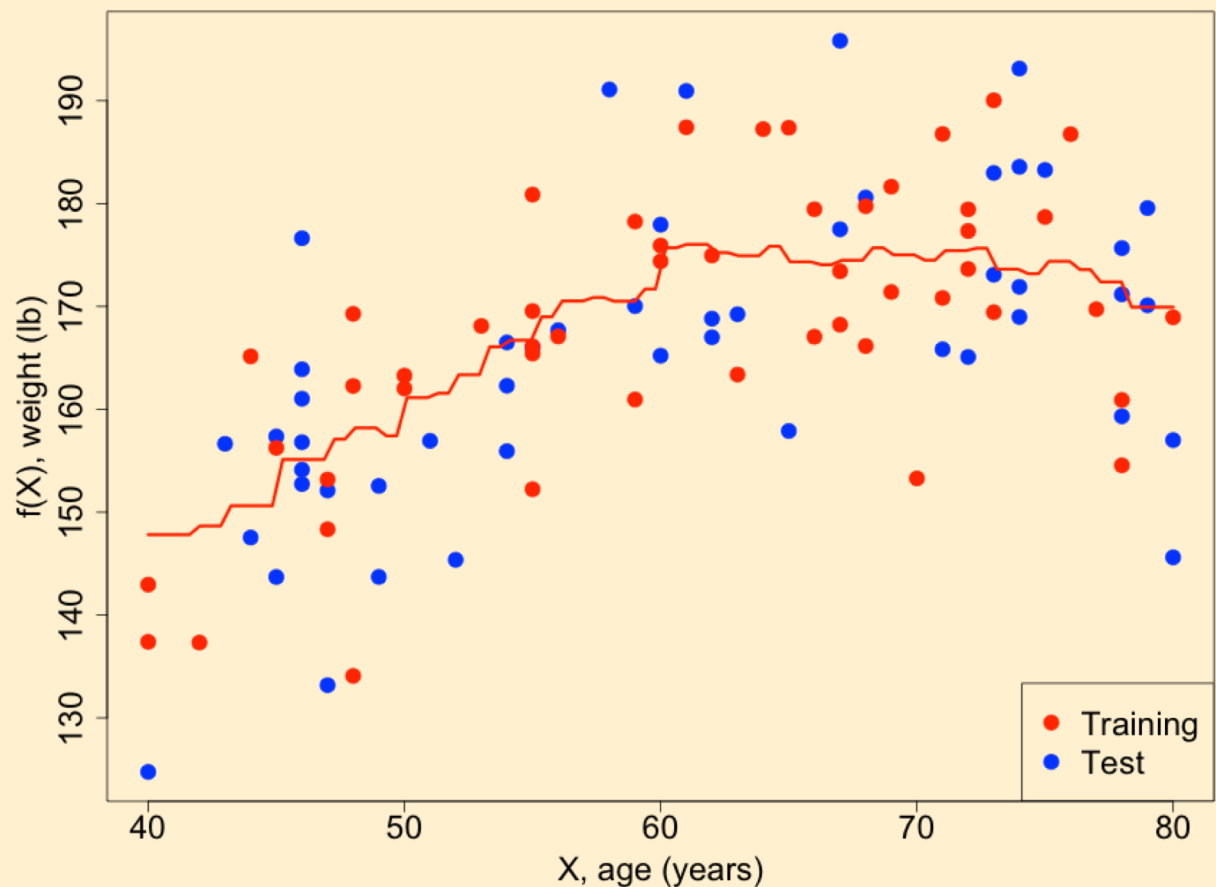
Evaluate on independent test set

- Test dataset independently generated from the same model as the training data (similar to if I went out and collected more data of the same type)



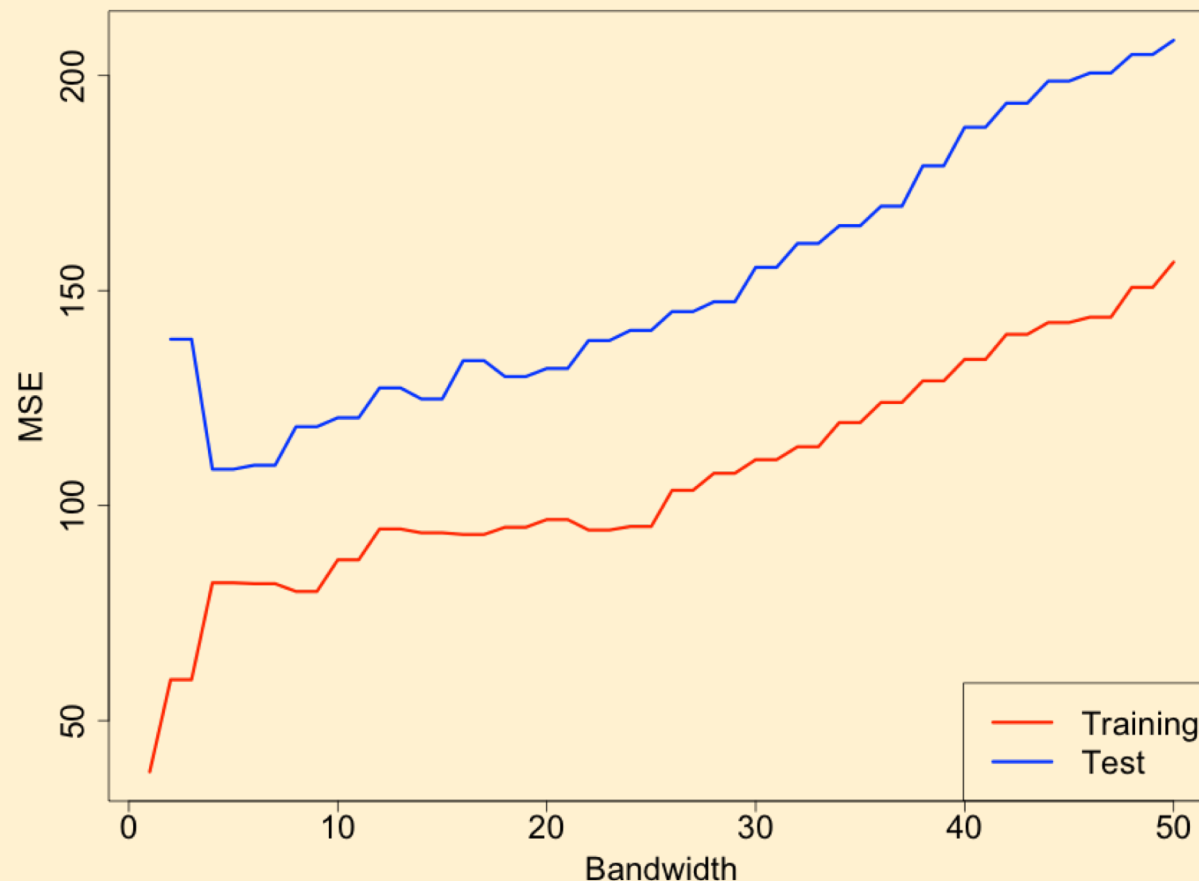
Evaluate on independent test set

- Test dataset independently generated from the same model as the training data (similar to if I went out and collected more data of the same type)
- Take kernel fit to training data and evaluate on test data (w.r.t. MSE)



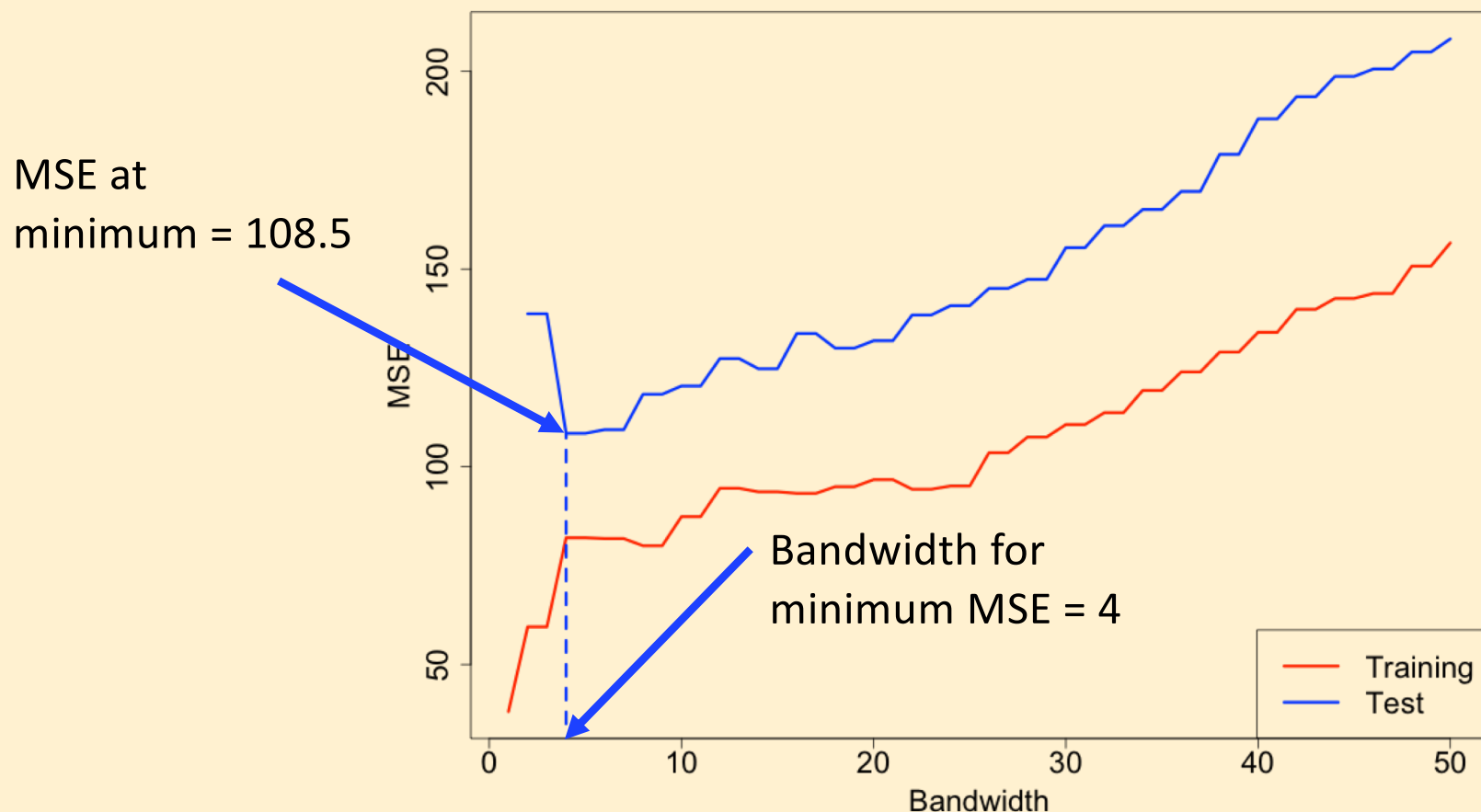
How to choose the bandwidth?

- We can try different values of the bandwidth and figure out which value gives us the smallest MSE on the test set: $E[(Y - g(X))^2 | X = x]$



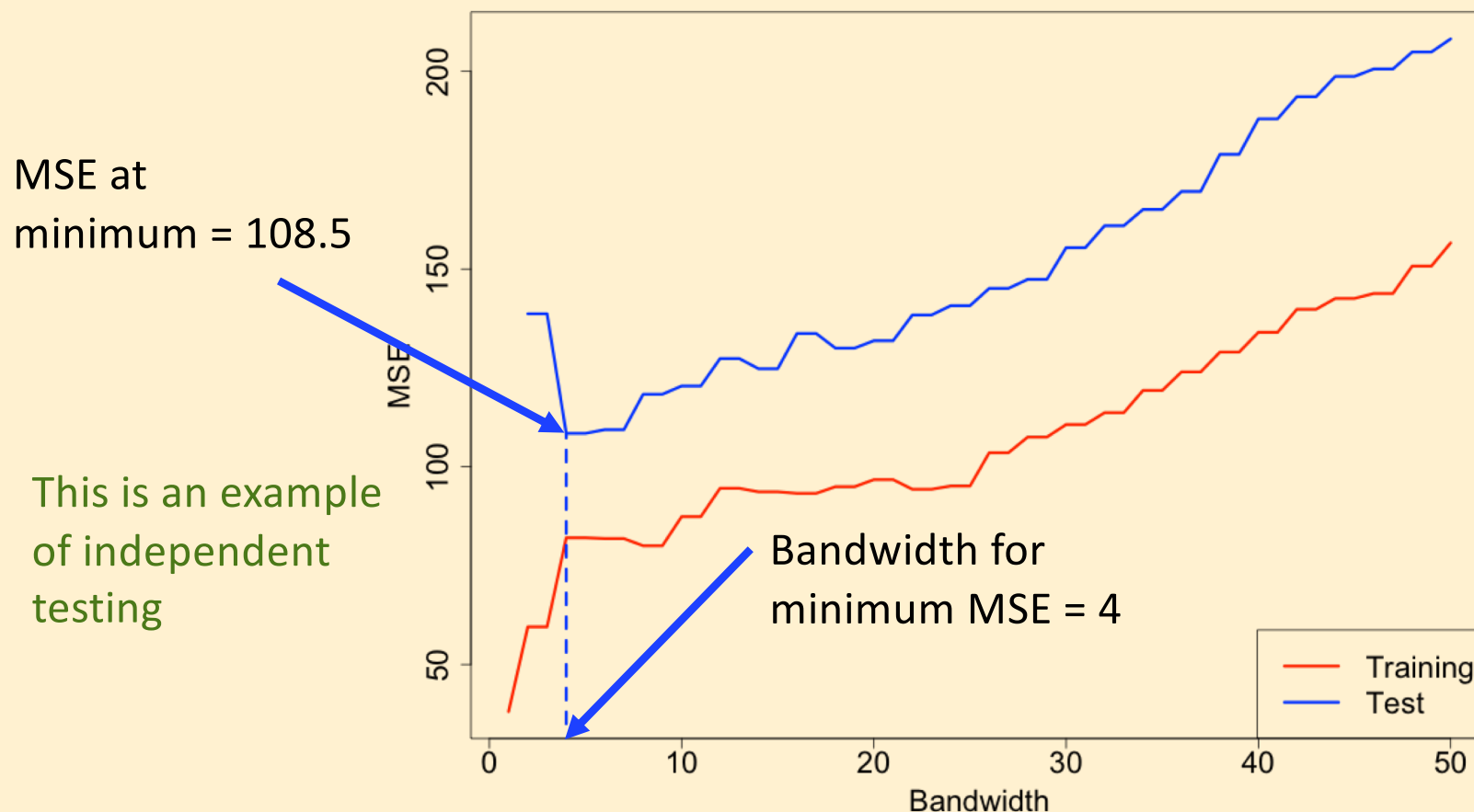
How to choose the bandwidth?

- We can try different values of the bandwidth and figure out which value gives us the smallest MSE on the test set: $E[(Y - g(X))^2 | X = x]$



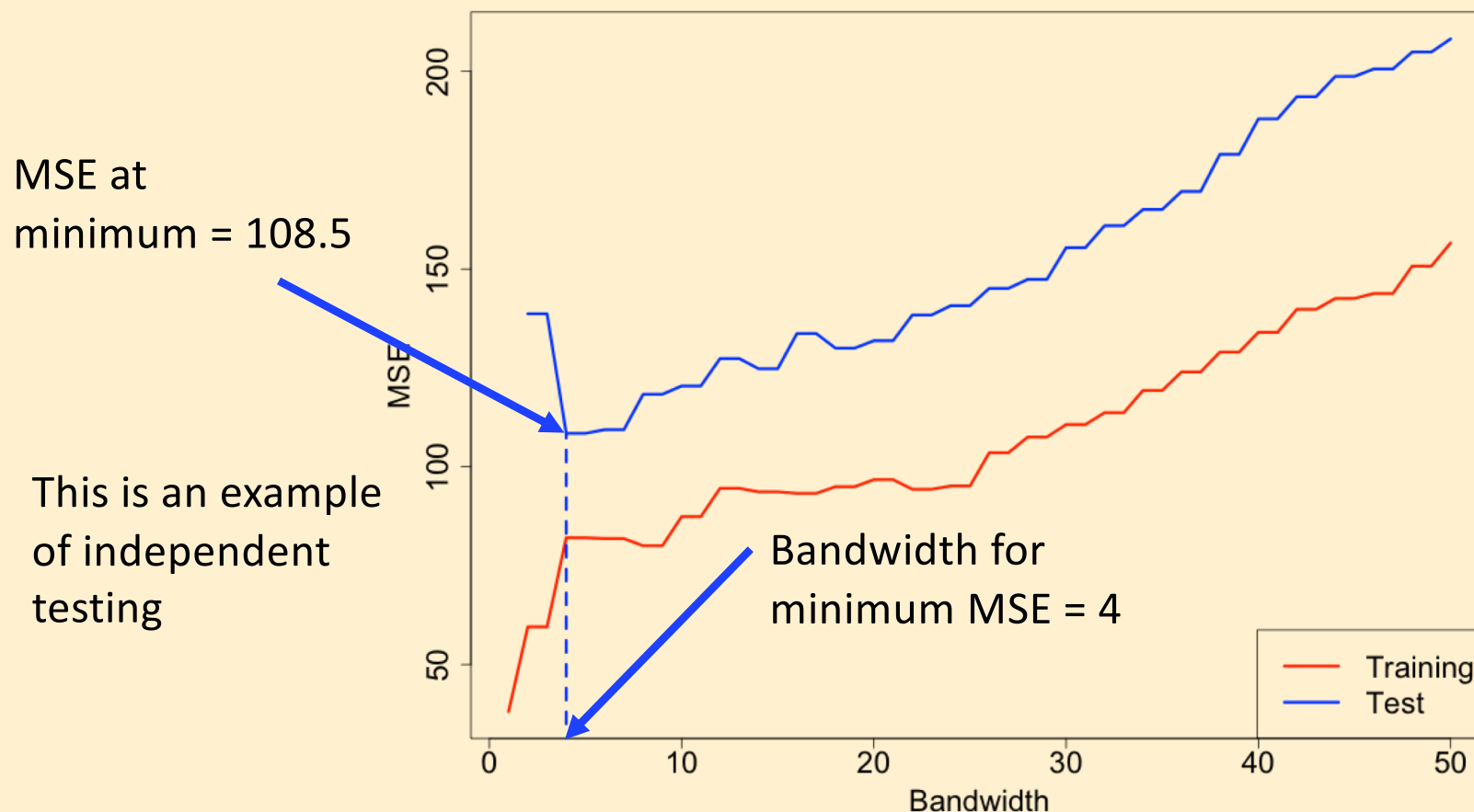
How to choose the bandwidth?

- We can try different values of the bandwidth and figure out which value gives us the smallest MSE on the test set: $E[(Y - g(X))^2 | X = x]$



How to choose the bandwidth?

- We can try different values of the bandwidth and figure out which value gives us the smallest MSE on the test set: $E[(Y - g(X))^2 | X = x]$



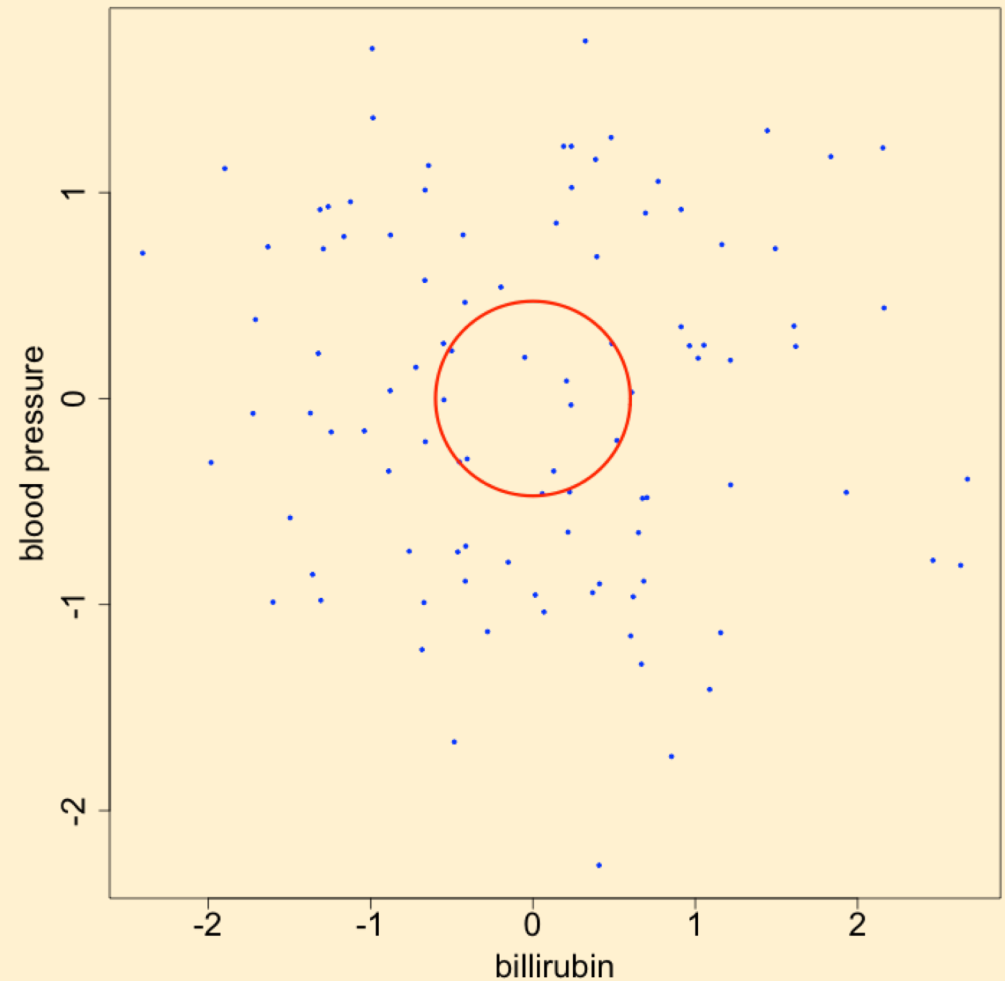
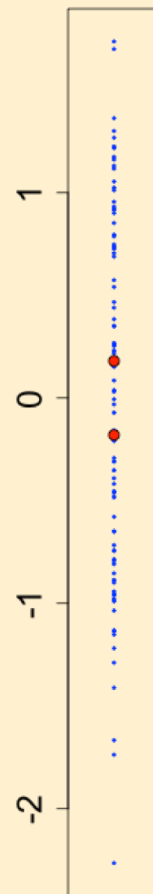
Finding the minimum of a loss function such as MSE is a common computational problem in machine learning

What if we want to do smoothing with 2 or more predictors?

- Nearest neighbor methods can be extended to higher dimensions (i.e. more than one predictor)
- Works well for large N , and relatively small p , i.e. $p \leq 4$
- Need to *standardize/normalize* predictors
- We used a specified bandwidth for 1D example, but more common to use the k -nearest neighbors (kNN)
- Often have different weighting kernels – observations further away from the point contribute more to the local estimate than those further away in the neighborhood – normal kernel example in lab

Curse of dimensionality

- 100 data points
- Standardized on each axis
- Compare size of kernel needed to capture 10 points in each of 1D and 2D cases

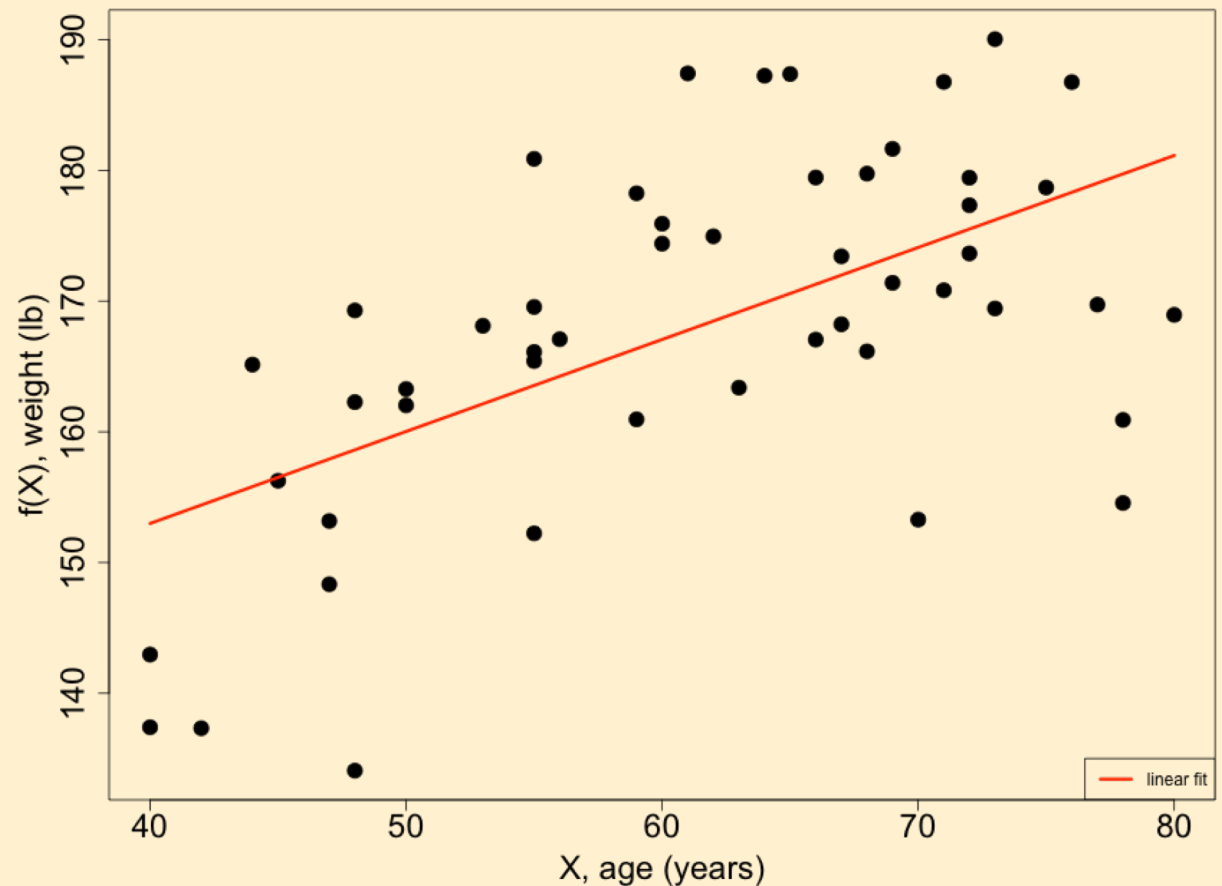


Linear model

- So far we have dealt with estimating $f(X)$ in a non-parametric way – i.e. using neighborhoods
- But that leads to the curse of dimensionality
- We can get around the curse of dimensionality by adding structure or defining a model
- You have seen the *linear model* before, but we will look at it slightly differently to motivate the machine learning paradigm
- Linear model has form: $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$
- Specified in terms of $p + 1$ parameters $\beta_0, \beta_1, \beta_2, \dots, \beta_p$
- Estimate parameters by fitting linear model to training data
- Although the linear model is almost always wrong, it can serve as a useful and interpretable approximation to the true function $f(X)$

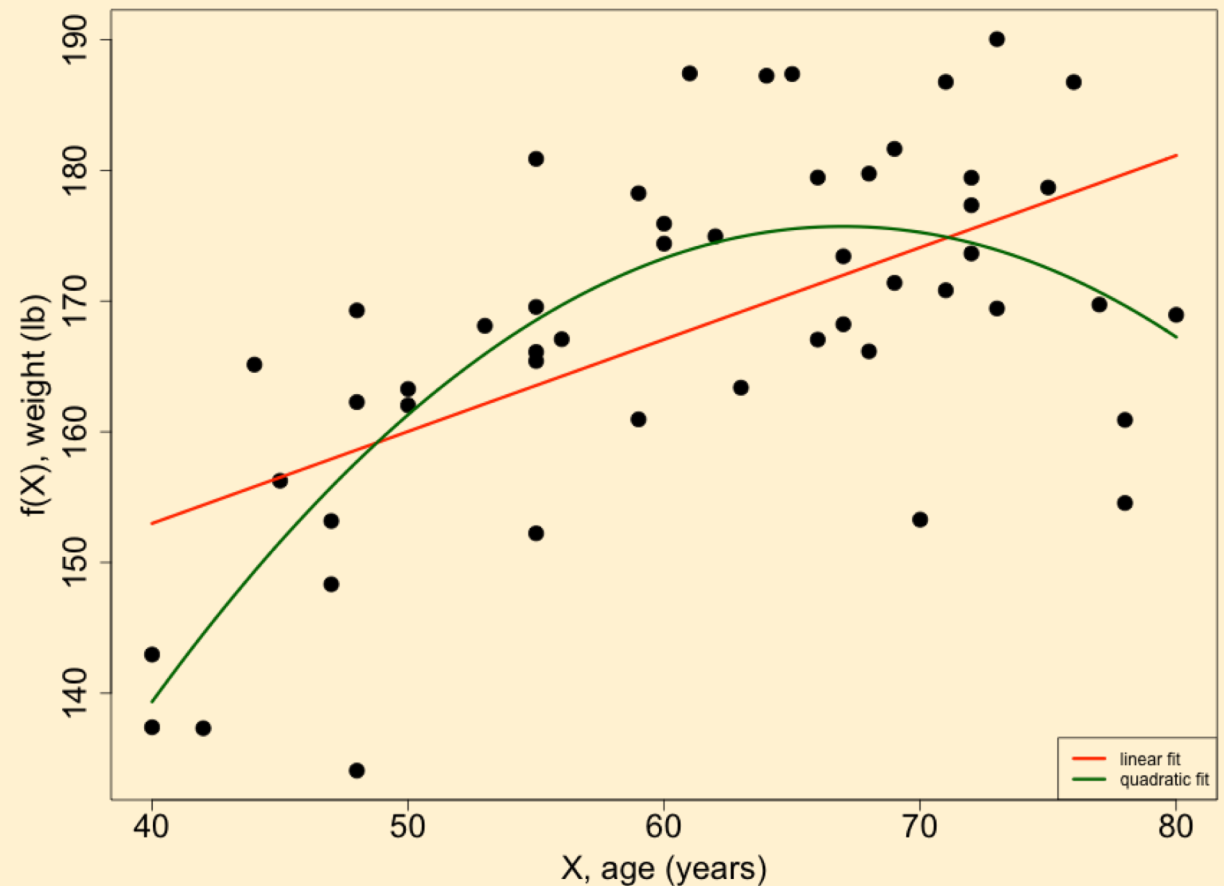
Linear model – linear

- Fit the model
$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1$$
- Finds $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize MSE
- Straight forward for linear models – analytic solution



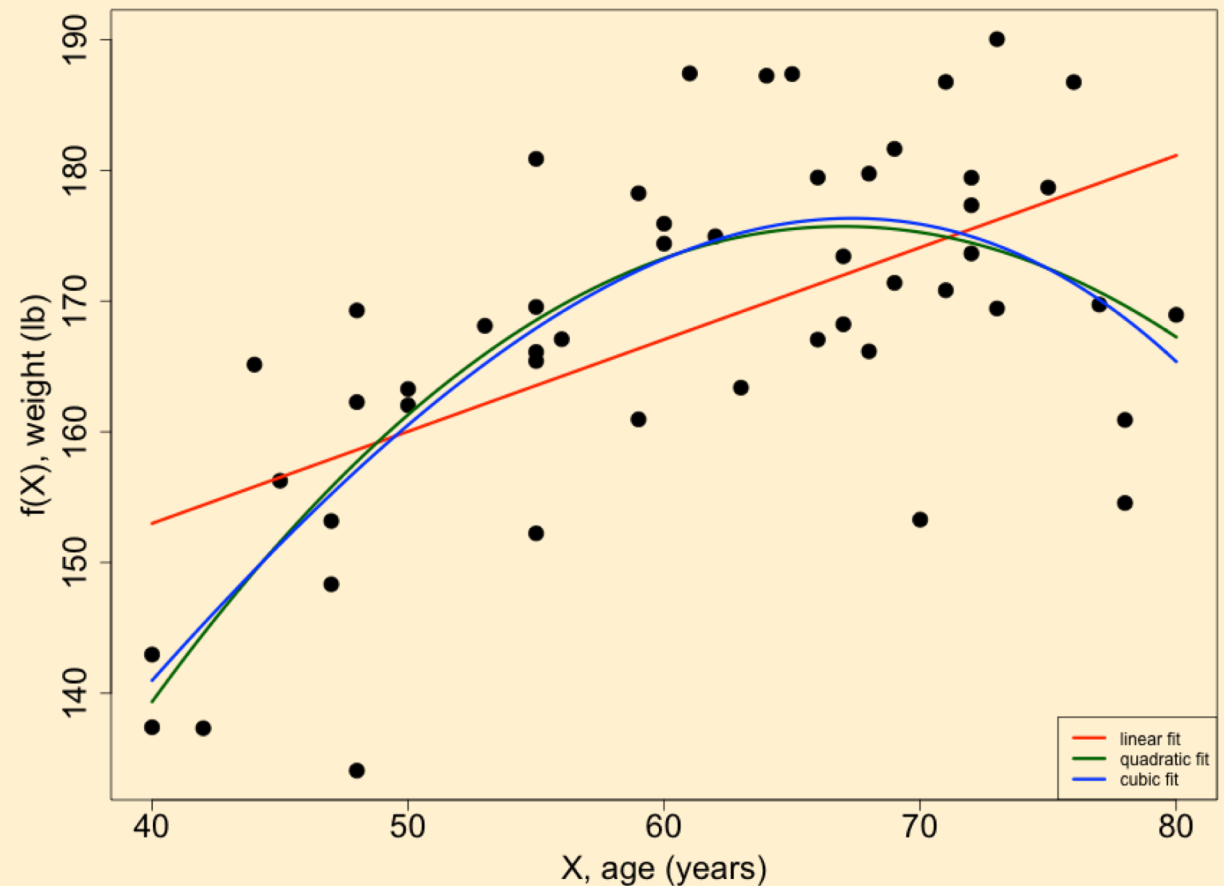
Linear model – quadratic

- $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_1^2$



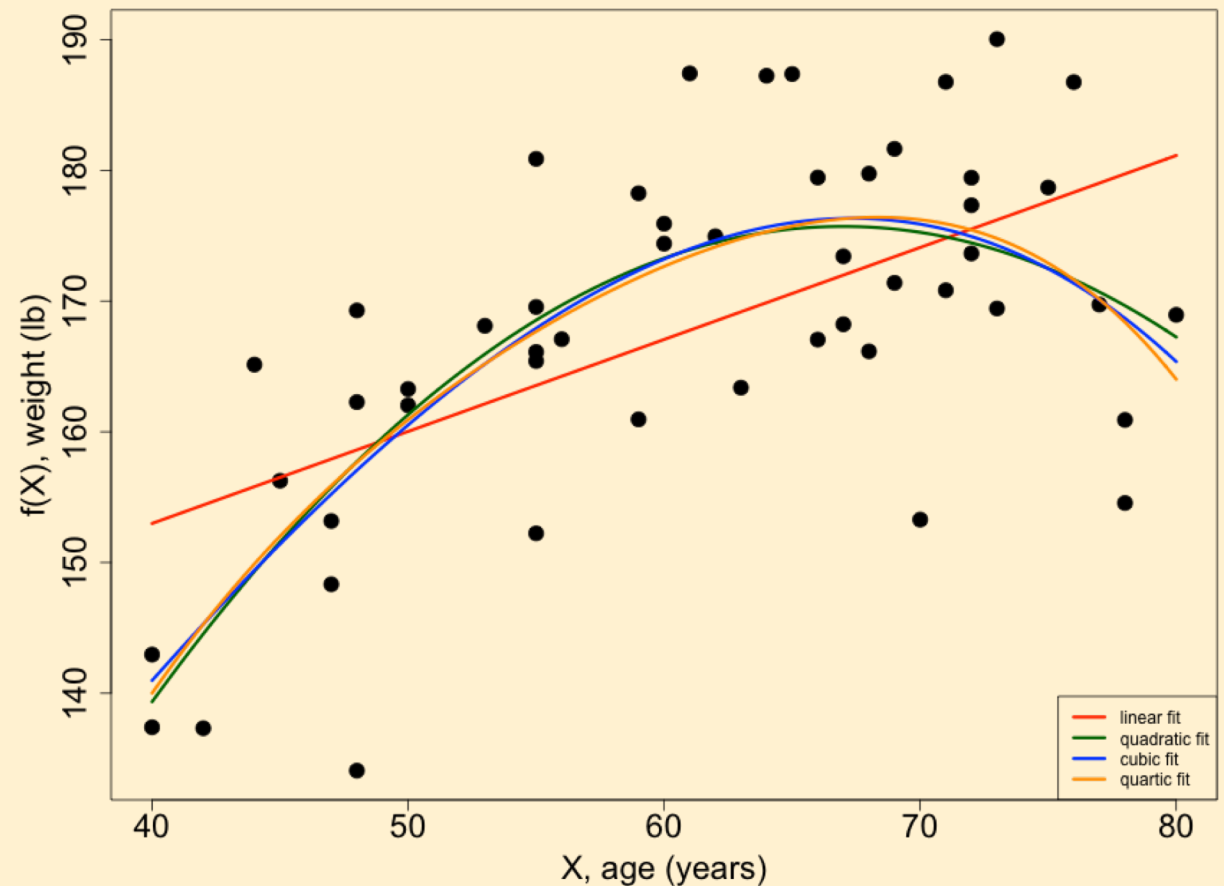
Linear model – cubic

- $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_1^2 + \hat{\beta}_3 X_1^3$



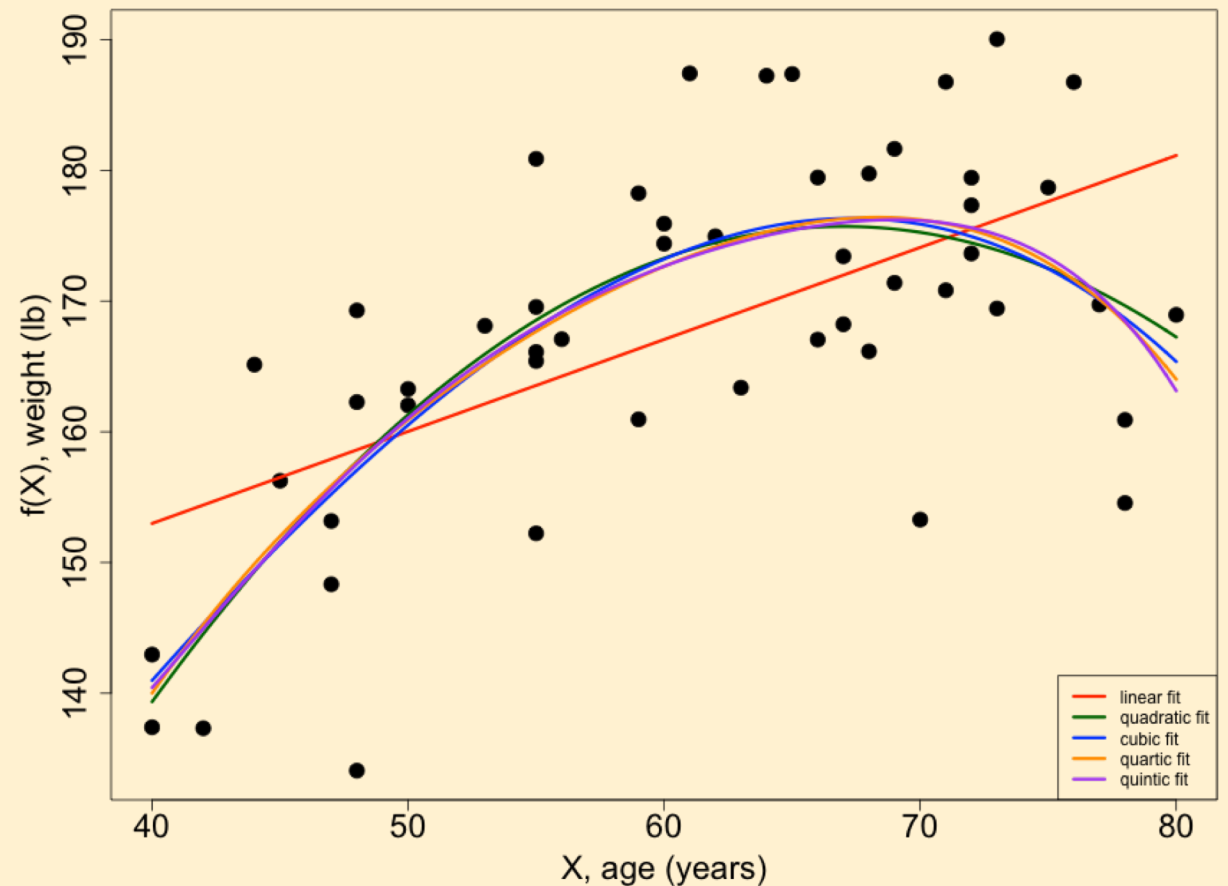
Linear model – quartic

- $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_1^2 + \hat{\beta}_3 X_1^3 + \hat{\beta}_4 X_1^4$

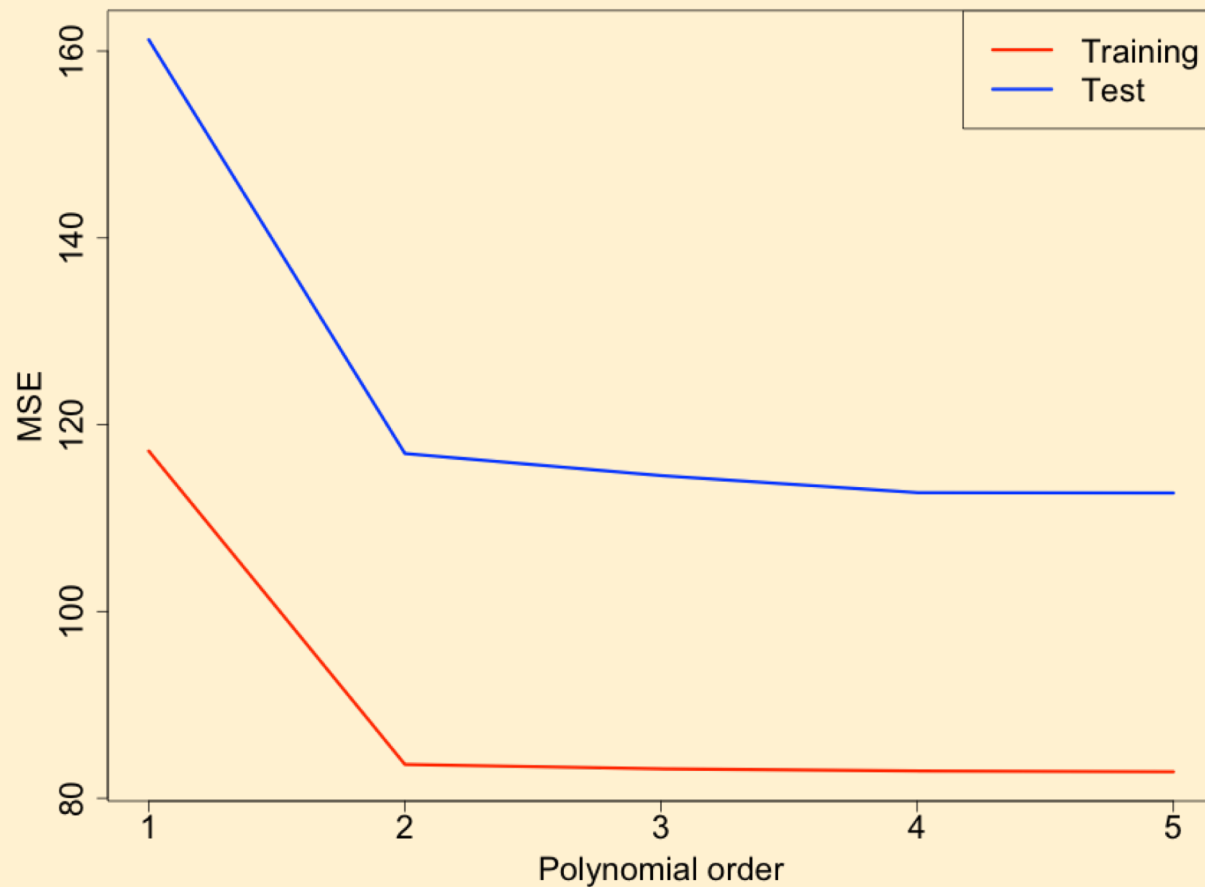


Linear model – quintic

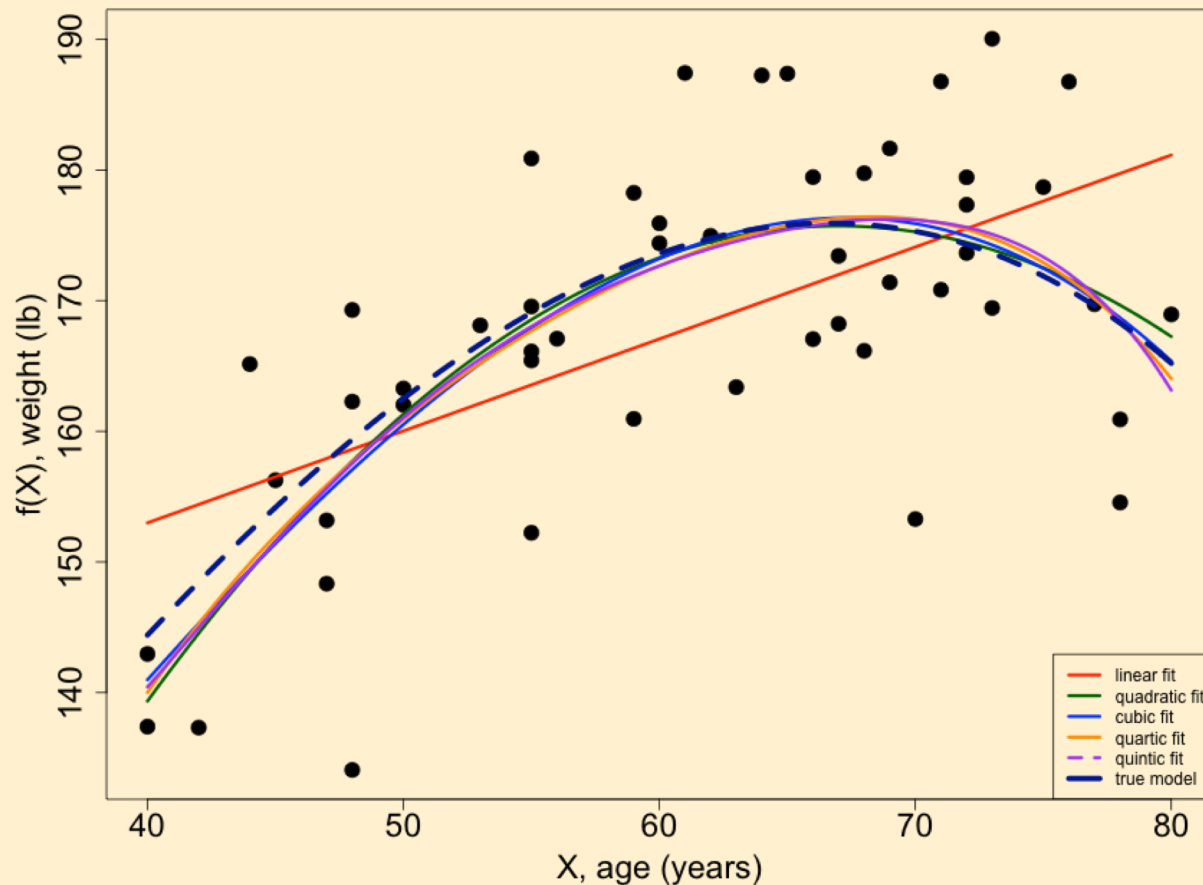
- $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_1^2 + \hat{\beta}_3 X_1^3 + \hat{\beta}_4 X_1^4 + \hat{\beta}_5 X_1^5$



Which order polynomial to pick?

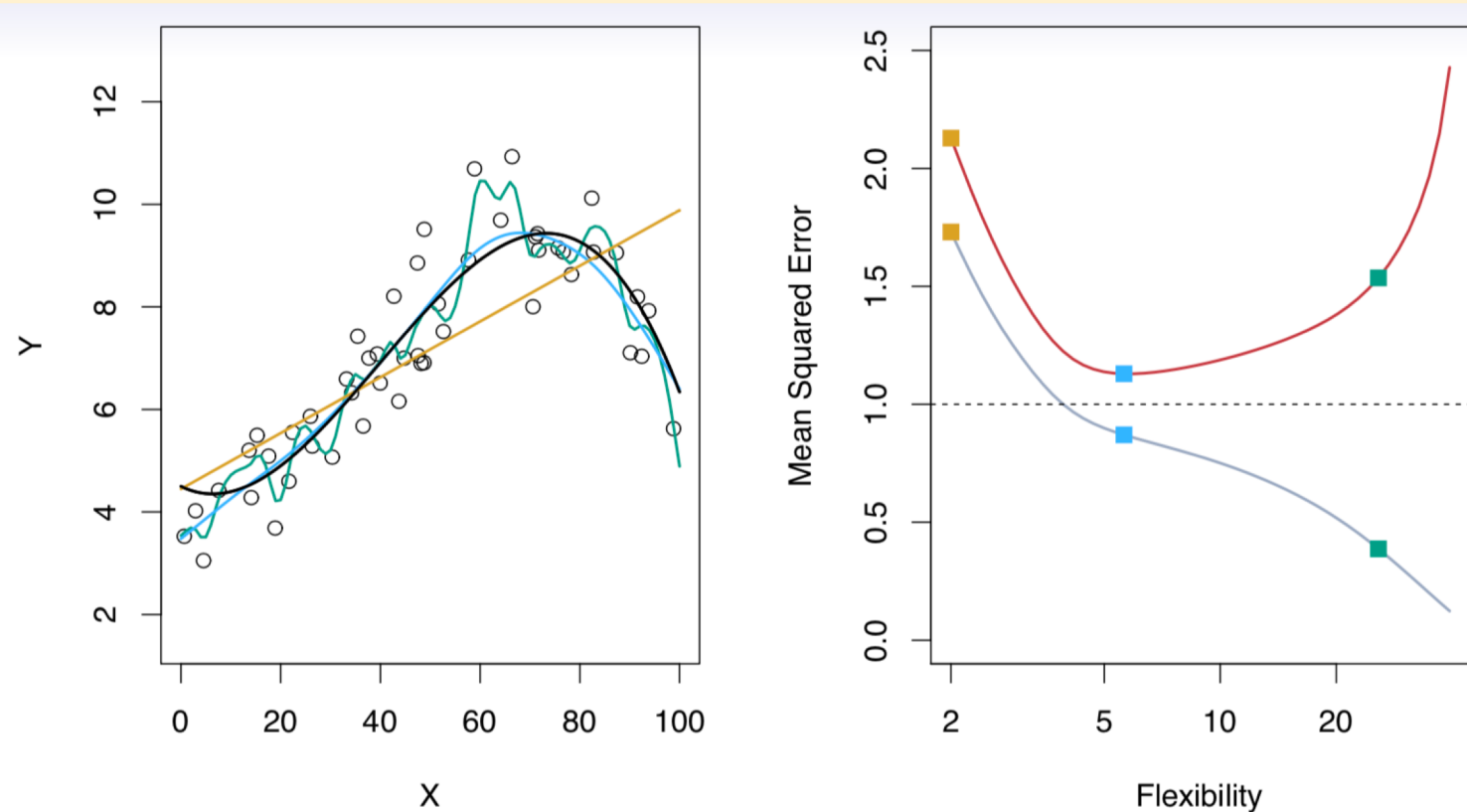


The truth was cubic



- Trade off between complexity and fit in the test data
- Would like a simple model
- Would like good fit to test data
- For this data quadratic looked good enough

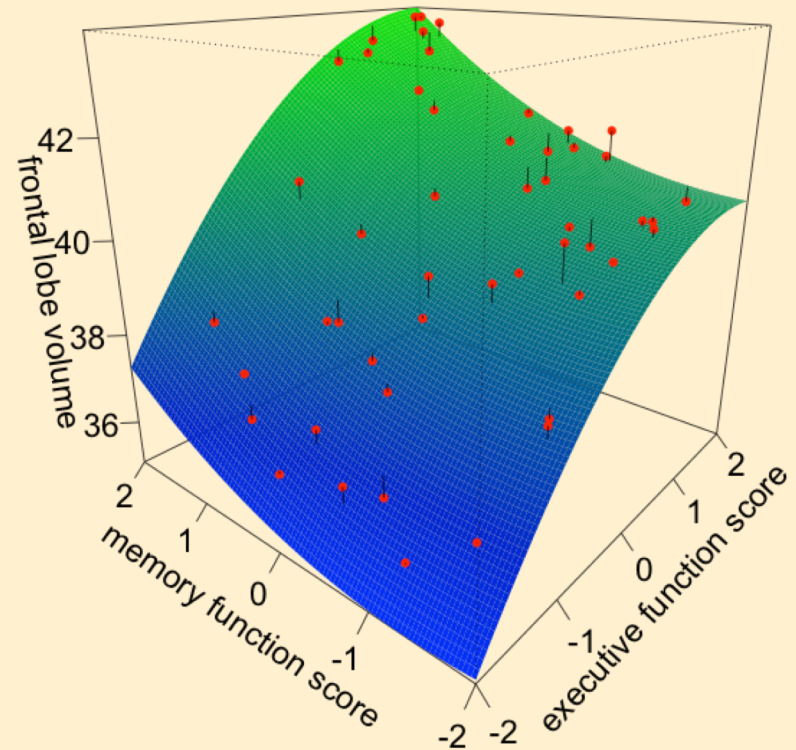
Another example (from ISLR)



Black curve is truth. Red curve on right is MSE_{Te} , grey curve is MSE_{Tr} . Orange, blue and green curves/squares correspond to fits of different flexibility.

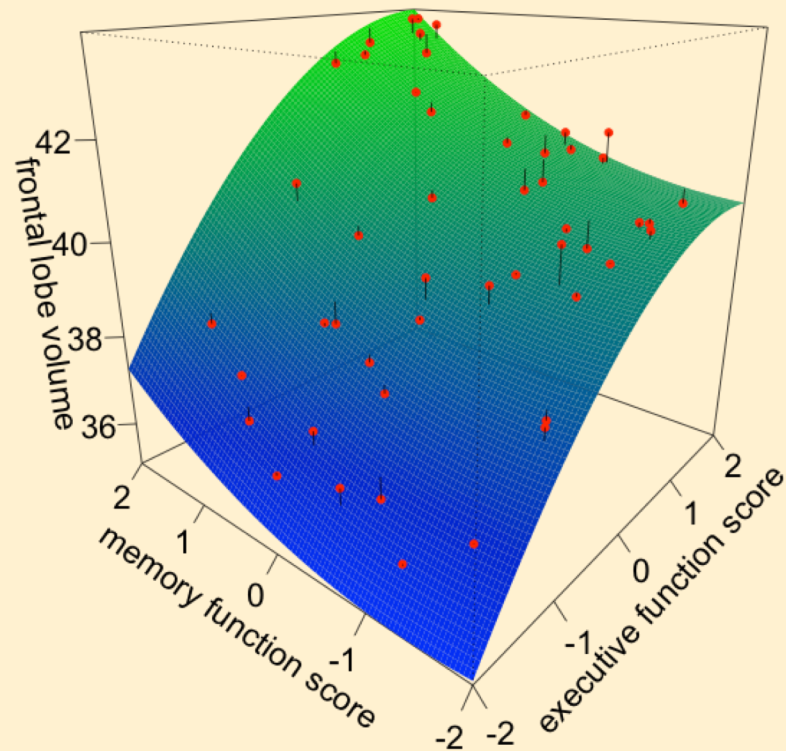
Linear models with 2 or more predictors

- Simulated data relating cognitive scores to frontal region brain volume loss
- Frontal lobe volume is simulated as a quadratic function of memory and executive scores + normal noise, i.e., $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 + \beta_5 X_2^2 + \varepsilon$

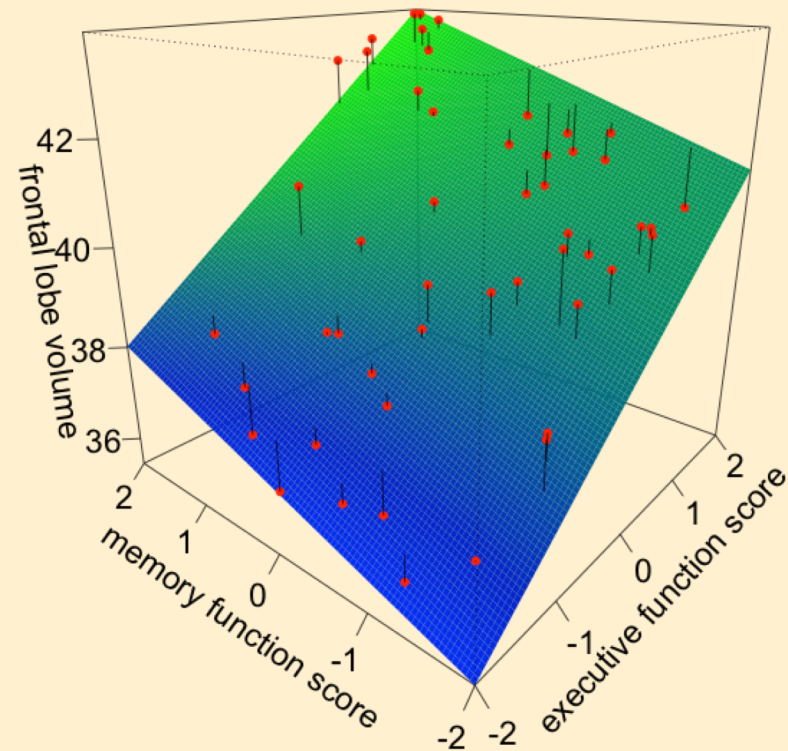


Linear terms only

TRUTH

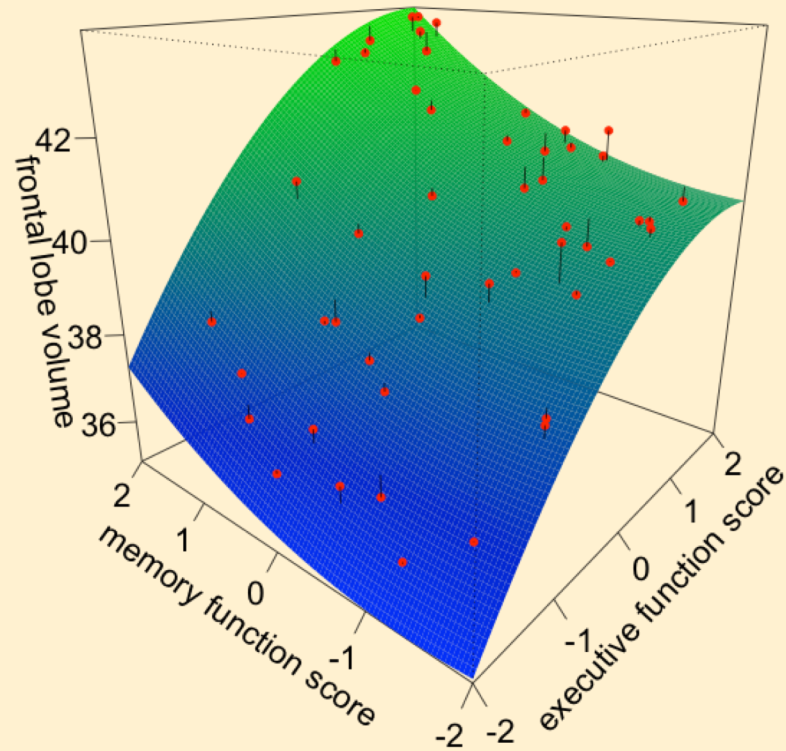


$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$

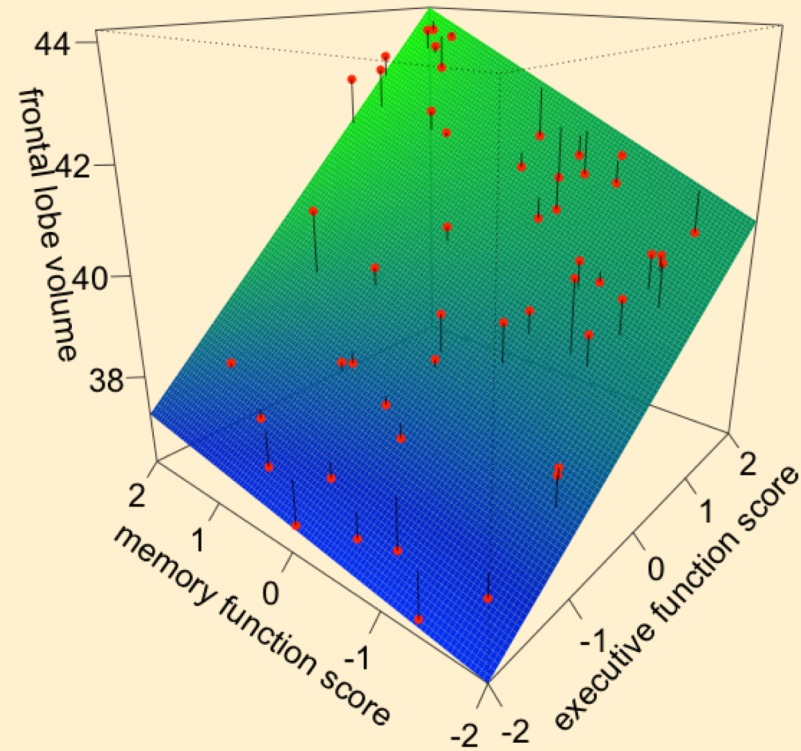


Adding an interaction

TRUTH



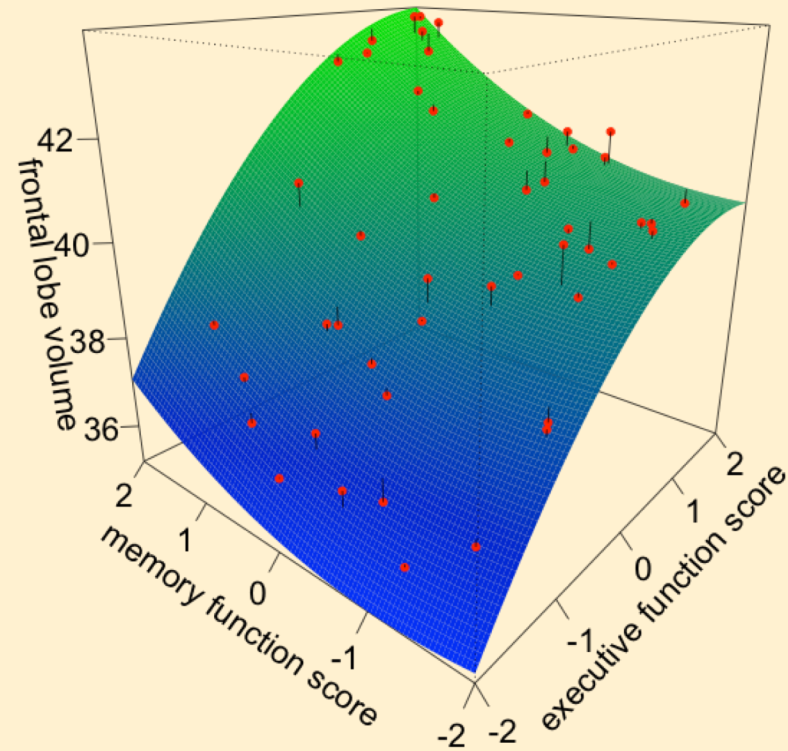
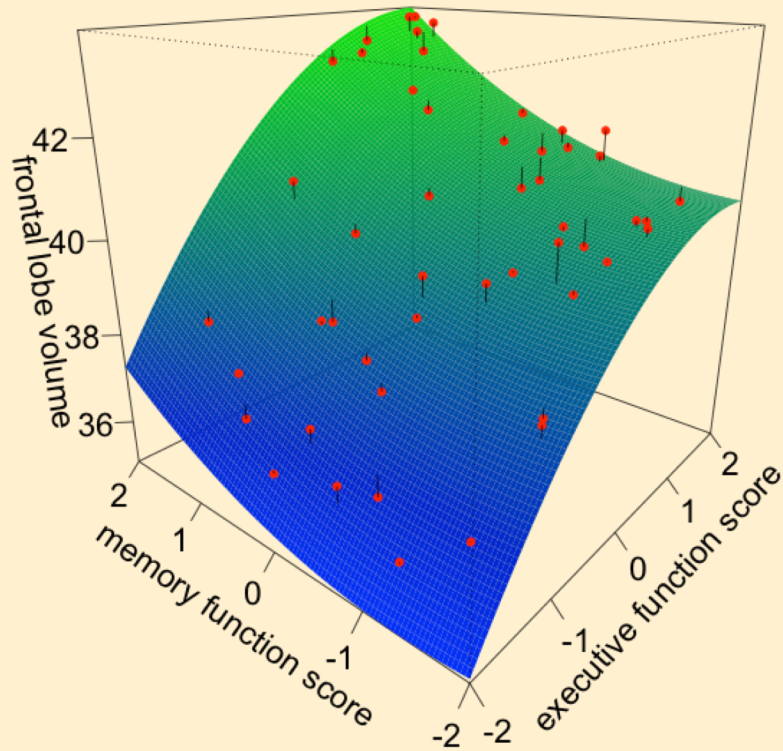
$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_1 X_2$$



And finally quadratic terms

TRUTH

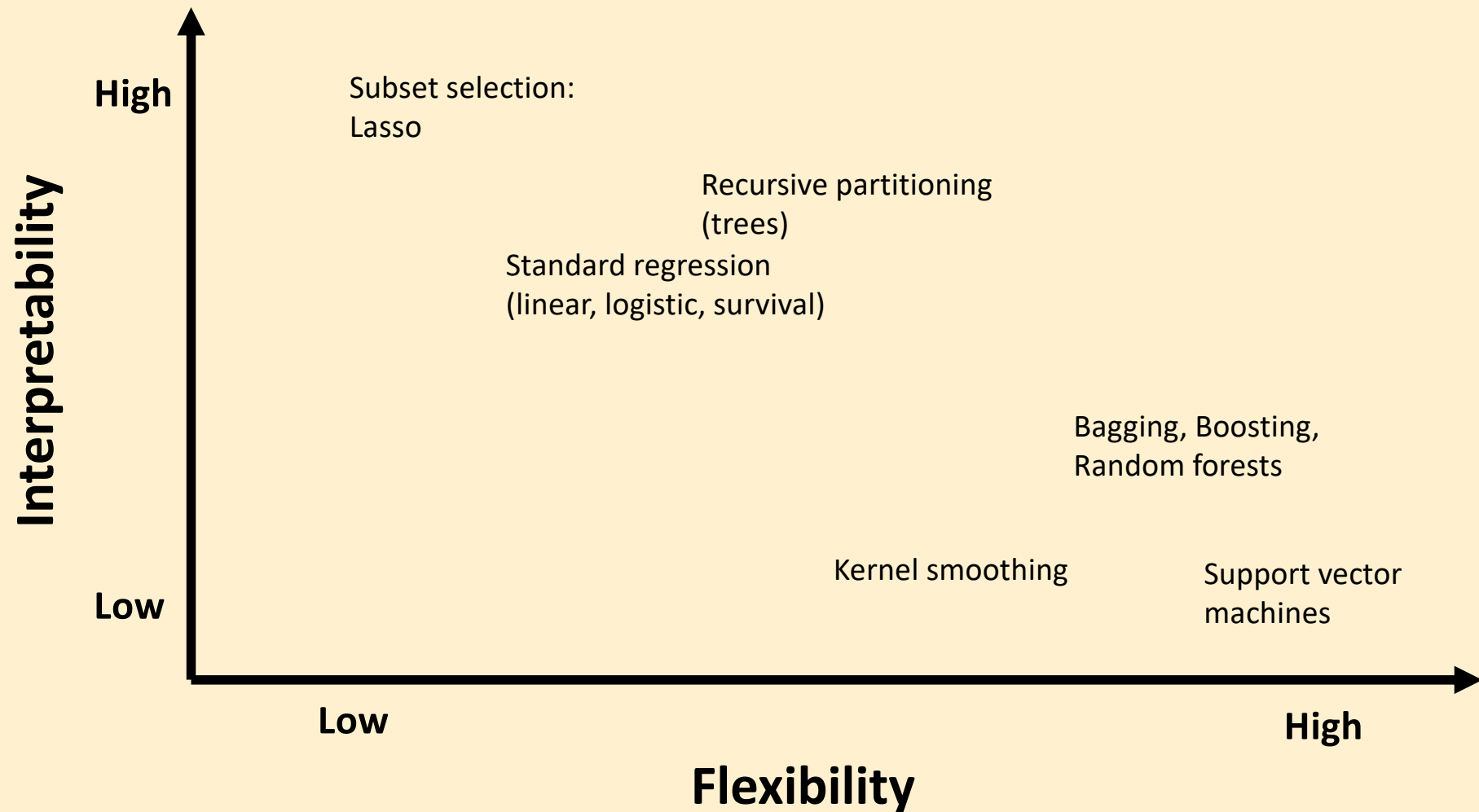
$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_1 X_2 + \hat{\beta}_4 X_1^2 + \hat{\beta}_5 X_2^2$$



Trade-offs

- Prediction accuracy vs. interpretability
 - Linear models easy to interpret; nearest-neighbors not so much
- Good fit vs. under or over-fitting
 - How do we find the fit which is “just right”
- Parsimony vs. black box
 - Usually prefer a simpler model involving fewer variables over a black-box predictor involving them all

Trade-offs



What we covered in this lecture:

- What is Machine Learning?
- Supervised vs. Unsupervised learning
- Lead into prediction techniques for supervised learning with continuous outcomes:
 - Local estimation and nearest neighbors
 - Loss functions and mean square error
 - The training-test paradigm
 - The curse of dimensionality
 - Linear regression modeling (including polynomials)
 - Trade-offs
 - Assessing model accuracy

Next lecture(s)

- Classification
- Loss functions for binary data
- Matrix algebra and calculus
- Minimizing loss functions

Extra materials

- Linear algebra tutorials – Andrew Ng, Stanford Machine Learning Course on YouTube or Coursera
Lecture 3.1 to 3.6

https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcIltqhlRJLN

- And read pages 9-11 of ISLR