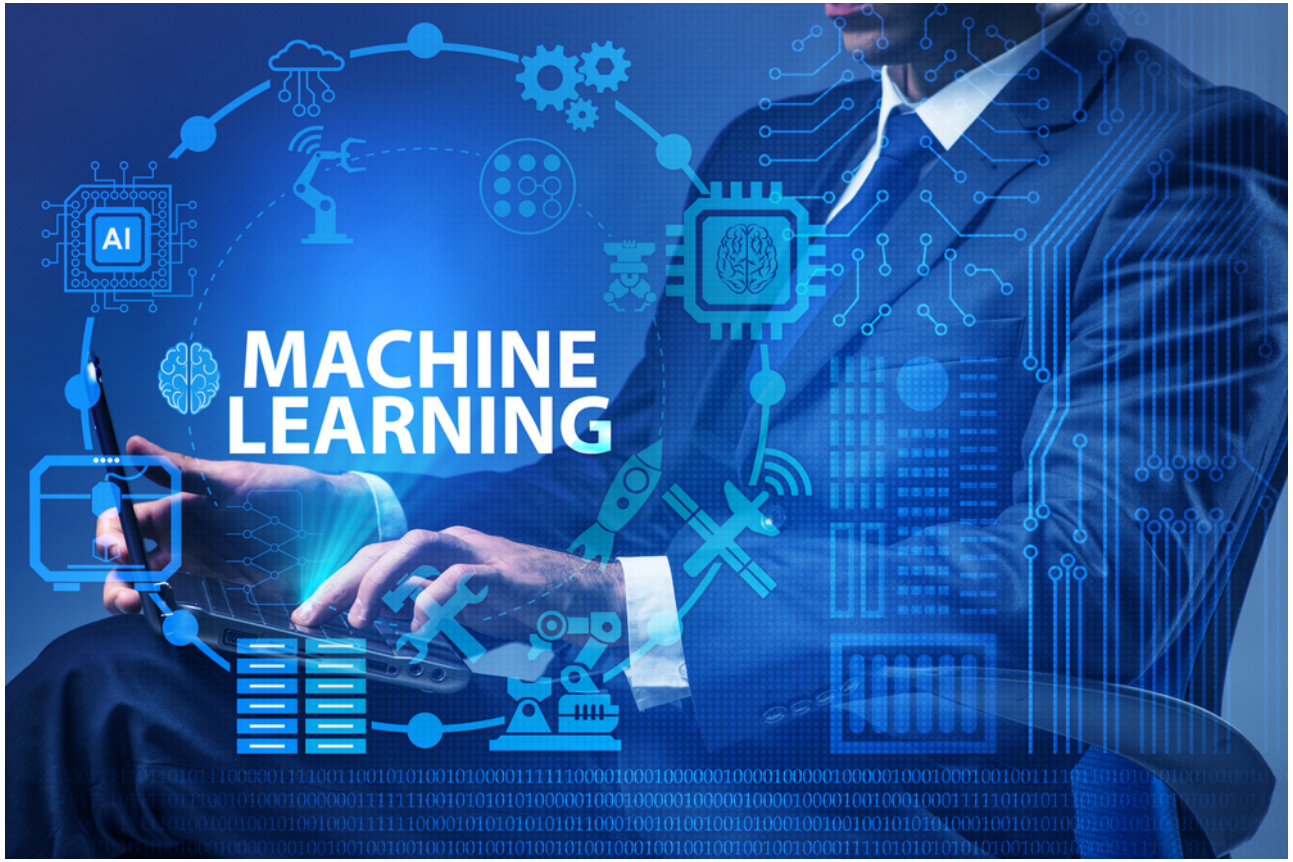
DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 1
Introduction and Review



Definition

- What is machine learning?
  - A set of tools for understanding data by building models *from data*
  - Methods for *automatically* learning and recognizing complex patterns from data
  - Arthur Samuel (1959): *"[a] field of study that gives computers the ability to learn without being explicitly programmed"*

Definition

- These uncovered patterns are then used to predict future data, or to perform other kinds of decision-making under uncertainty.

- The key premise is *learning* from data!!

- predict = guess the value(s) of unknown variable(s)
  - (not necessarily prediction of future… c.f. *forecasting*)
- future data = data you haven't seen before

Big Data is Everywhere

We are in the era of **big data**!

- 40 billion indexed web pages
- 100 hours of video are uploaded to YouTube every minute

The deluge of data calls for automated methods of data analysis, which is what **machine learning** provides!

Driving Forces

- Explosive growth of data in a great variety of fields
  - ❑ Cheaper storage devices with higher capacity
  - ❑ Faster communication
  - ❑ Better database management systems
- Rapidly increasing computing power

- We want to make the data work for us!!

Examples of Applications of Machine Learning

- In automated decision making
  - e.g. email spam filtering

Examples of Applications of Machine Learning

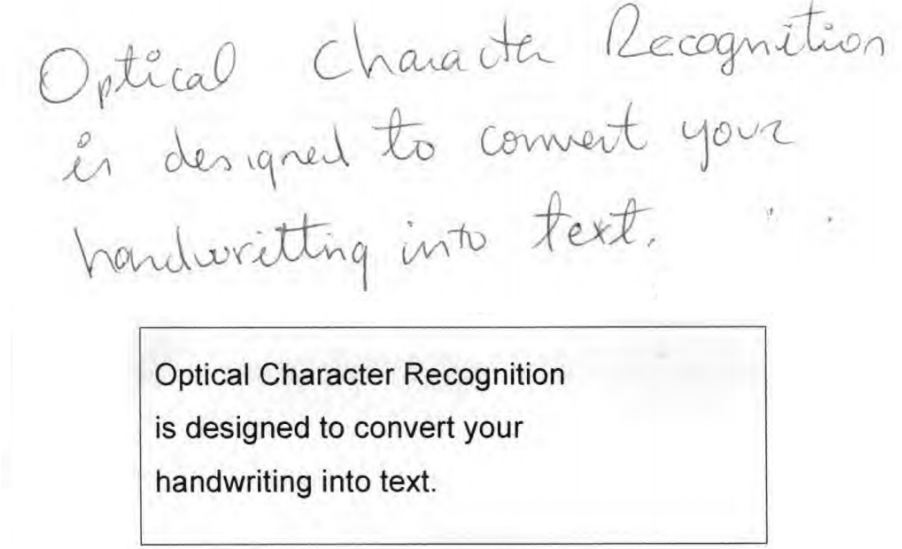
- In automated decision making
  - forensics / fraud detection

Examples of Applications of Machine Learning

- For automating complex programming tasks
  – e.g. driverless cars

Examples of Applications of Machine Learning

- For automating complex programming tasks
  – e.g. optical character recognition (OCR)



Examples of Applications of Machine Learning

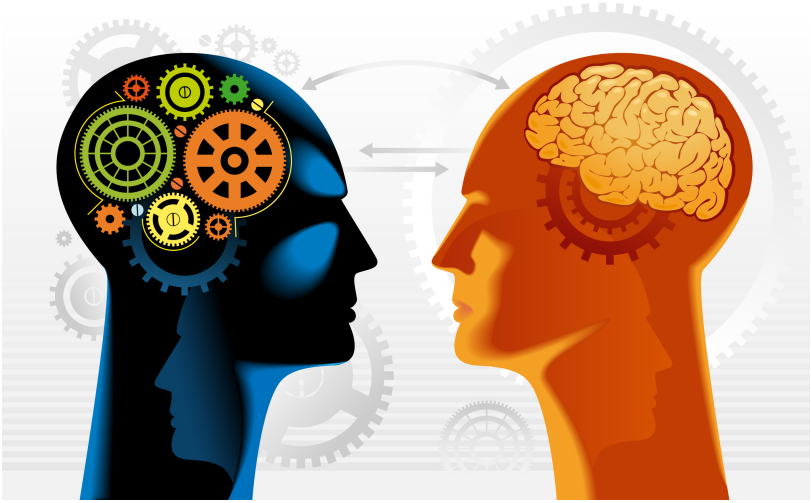
- In self-customizing algorithms
  - e.g. recommendation systems

Examples of Applications of Machine Learning

- In finance
  - To predict movements in markets
  - for risk management

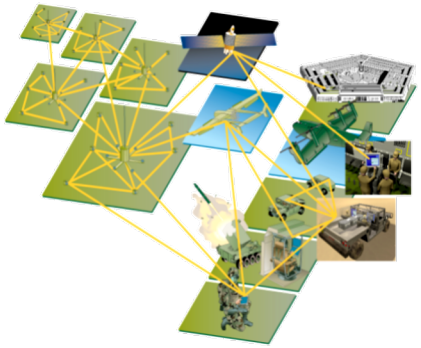
Examples of Applications of Machine Learning

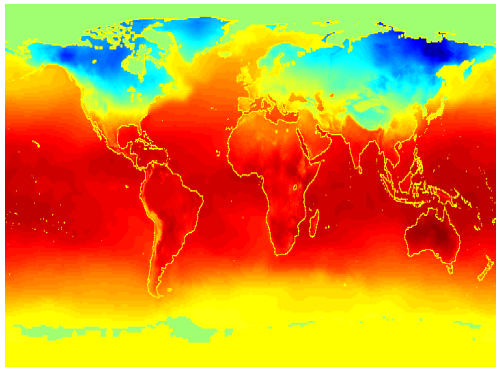
- Natural language processing
  - ❑ Speech recognition
  - ❑ Machine translation
  - ❑ Question Answering
  - ❑ Document classification and retrieval (books, email, web pages)
  - ❑ Sentiment analysis



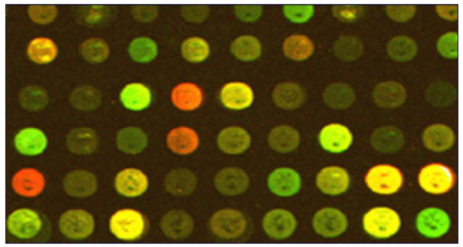
Examples of Applications of Machine Learning

- Scientific
  - Remote sensing networks: atmosphere, ocean, fresh-water, land-based, satellite
    - weather and climate modeling
    - environmental management
    - resource management





  - Biomedical: gene sequencing, gene expression, epidemiology, disease prediction



Types of Learning

- Supervised learning
  - Goal: Prediction
  - Semi-Supervised Learning
- Unsupervised learning
  - Goal: Discovery of Similarity or dissimilarity
  - Goal: Dimensionality Reduction
- Reinforcement learning

Supervised Learning

Starting point:

- Outcome measurement Y (also called dependent variable, response, target).
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables).

Supervised Learning

- In the *regression problem*, *Y* is quantitative (e.g price, blood pressure).
- In the *classification problem*, *Y* takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample).

Supervised Learning

- We have input-output data $(x_1, y_1), \ldots, (x_N, y_N)$ that are called **training data**.
- These are observations (examples, instances) of these measurements.

Objectives

On the basis of the training data we would like to:

- Accurately predict unseen test cases, i.e., **generalize.**
- Understand which inputs affect the outcome, and how.
- Assess the quality of our predictions and inferences.

Classification

- Document classification
  - Is this email spam?
  - Is this tweet positive toward this product?
  - Is this review/article real?
- Image classification
  - Is this a photo of a cat?
  - Which letter or number is written here?
- Object recognition
  - Identify the faces in this image
  - Identify pedestrians in this video

Classification

A classification algorithm is called a **classifier**

Classifiers require examples of inputs paired with outputs

- Called **training data**

Classifiers learn from training examples to map input to output

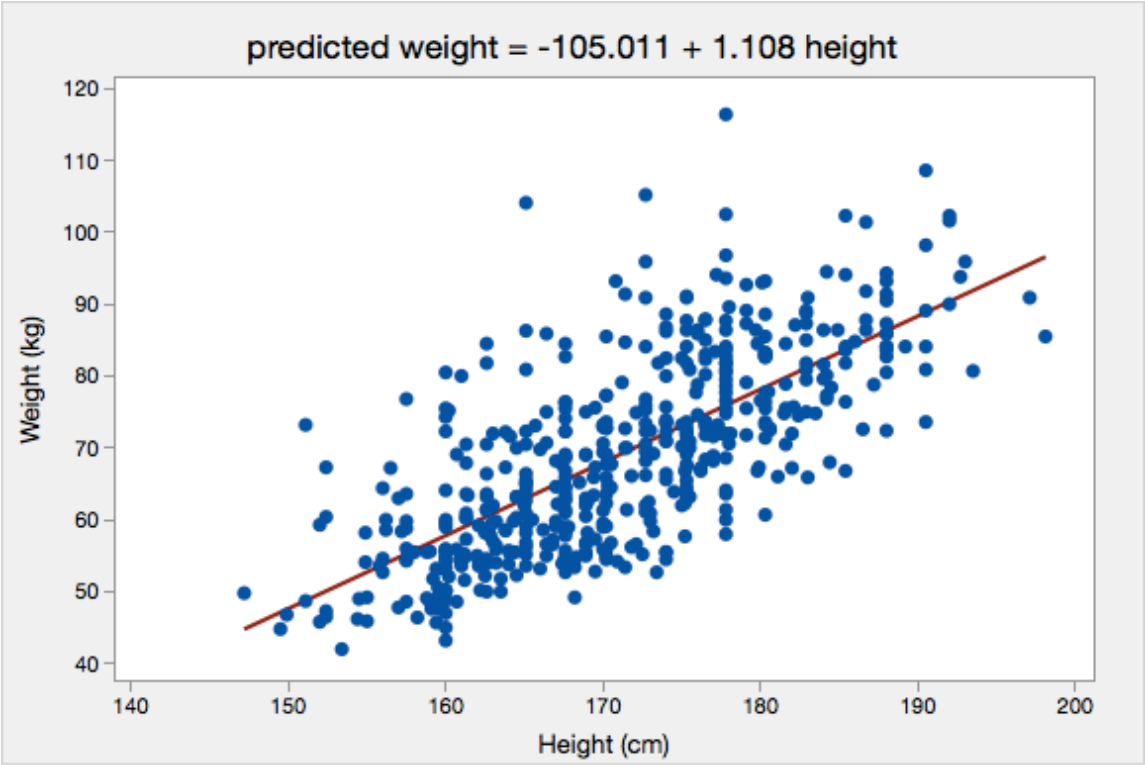
- Then when a classifier encounters new data where the output is unknown, it can make a prediction

Supervised Learning

Two types of prediction:

- Classification
  - Discrete outputs (typically categorical)
- **Regression**
  - **Continuous outputs (usually)**

Regression



Linear regression of weight based on one input variable (height)

Regression

Examples:

- Predict one's weight based on one's height
- Predicting how much money a product will make
- Forecasting a stock price tomorrow
- Estimate someone's age based on their face
- Predict a student's GPA based on their socio-economic conditions

Types of Learning

- Supervised learning
  - Goal: Prediction
  - Semi-Supervised Learning
- **Unsupervised learning**
  - **Goal: Discovery of similarity or dissimilarity**
  - **Goal: Dimensionality Reduction**
- Reinforcement learning

Unsupervised Learning

- No outcome variable, just a set of predictors (features) measured on a set of samples.
- Objective is vague— find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation.

Unsupervised Learning

- Difficult to know how well you are doing.
- Different from supervised learning, but can be useful as a pre-processing step for supervised learning.

Unsupervised Learning

Example: **anomaly detection**

- Trying to identify something unusual (e.g., fraud) but you don't know what it looks like

Unsupervised: Clustering & Dimensionality Reduction

- *Cluster analysis*: partition data into subsets that share common characteristics
  - e.g. group similar patients in a medical database
- *Dimensionality reduction*: create new features from original inputs that retain important information
  - e.g. represent a document as a small set of topics instead of as a large collection of words
- Methods: K--means, PCA, ICA

Unsupervised Learning

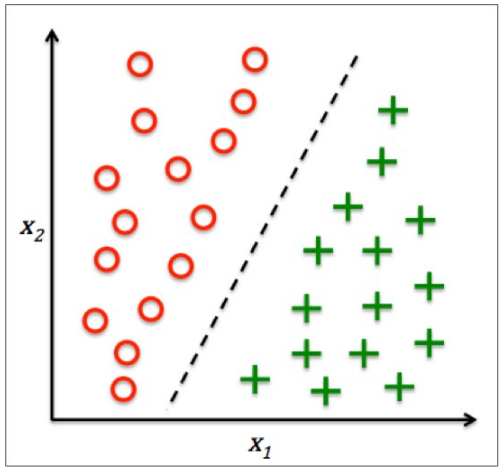
Example: movie recommendation (the Netflix problem)

- Clustering can be used to put people into different groups based on the kinds of movies they like.

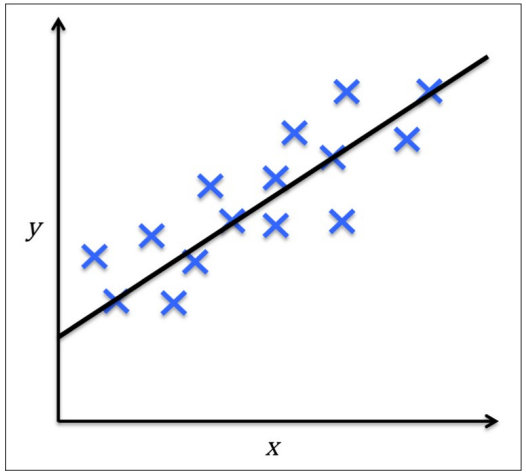
Interest Group 3:	**Interest Group 18:**	**Interest Group 8:**
Trainspotting	Mary Poppins	Pretty Woman
Fargo	Cinderella	Mrs. Doubtfire
Pulp Fiction	The Sound of Music	Ghost
Clerks	Dumbo	Sleepless in Seattle

From Hoffman (2004) "Latent Semantic Models for Collaborative Filtering."

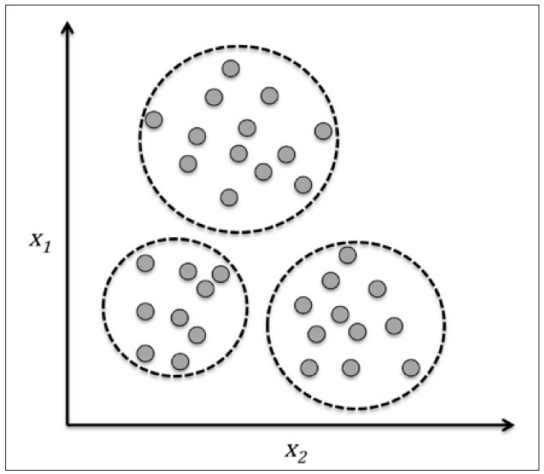
Classification



Regression

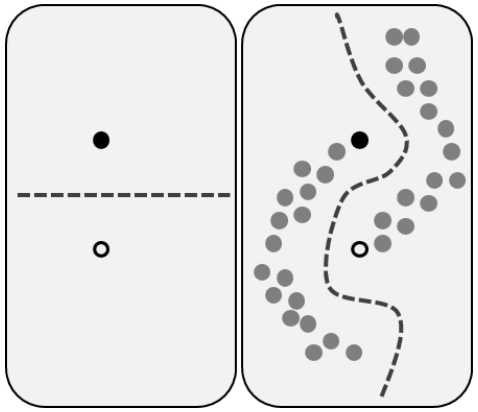


Clustering



Semi-supervised Learning

- A class of **supervised learning** tasks and techniques that also make use of unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data.
- Combines both kinds of learning



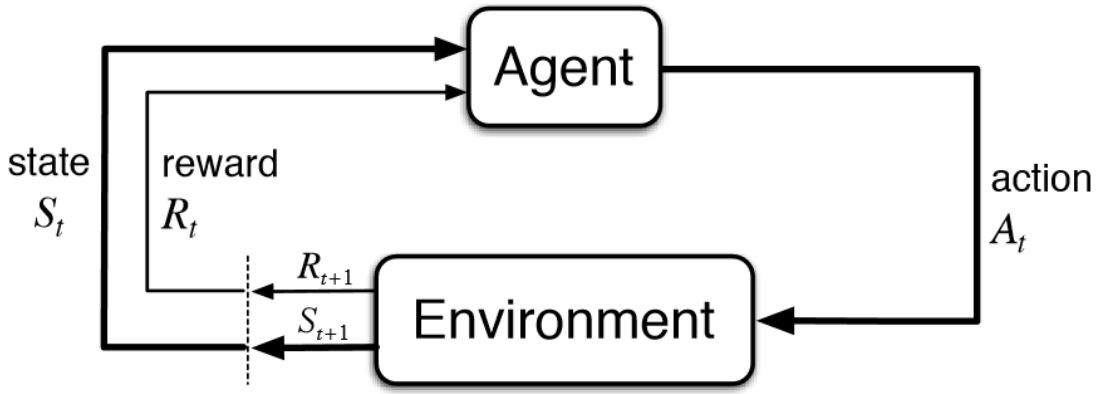
Types of Learning

- Supervised learning
  - Goal: Prediction
  - Semi-Supervised Learning
- Unsupervised learning
  - Goal: Discovery of Similarity or dissimilarity
  - Goal: Dimensionality Reduction
- **Reinforcement learning**
  - categorized as supervised learning as well

Reinforcement Learning

Setting:

- an **agent** interacts with an **environment**
- **actions** by the agent lead to different **states** of the environment
- some states will provide **rewards**

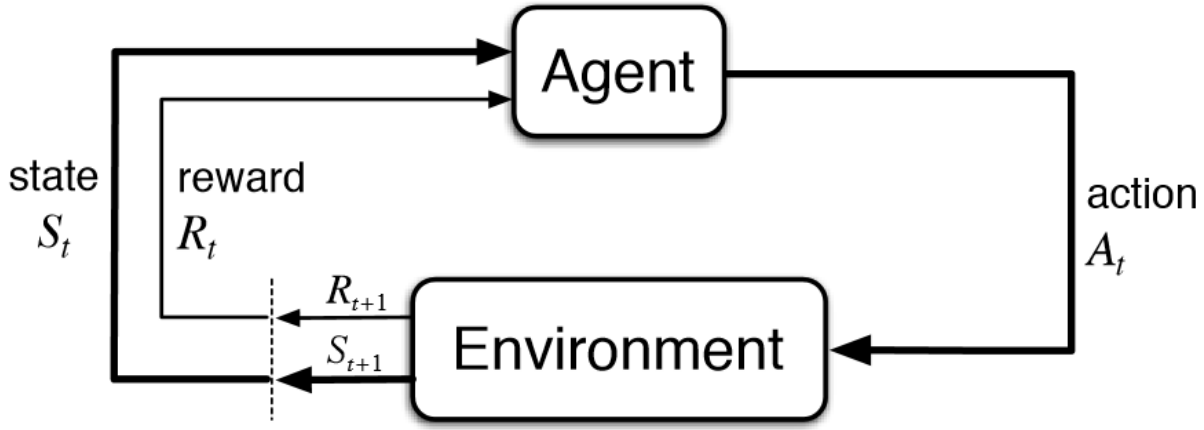


Reinforcement Learning

Setting:

Learning goal is to maximize rewards.

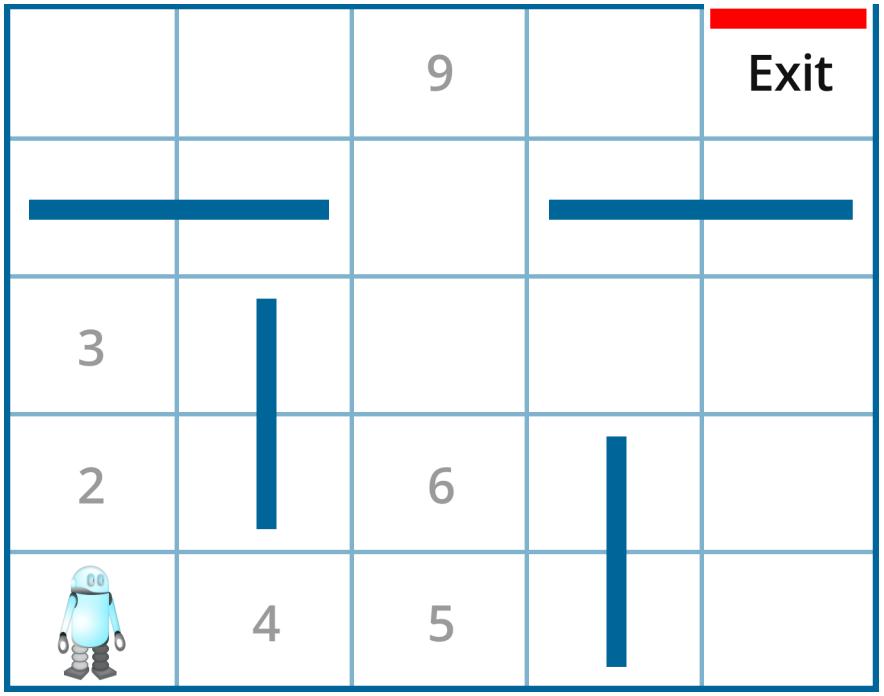
Used to learn models of how to behave, more complex than just input→output

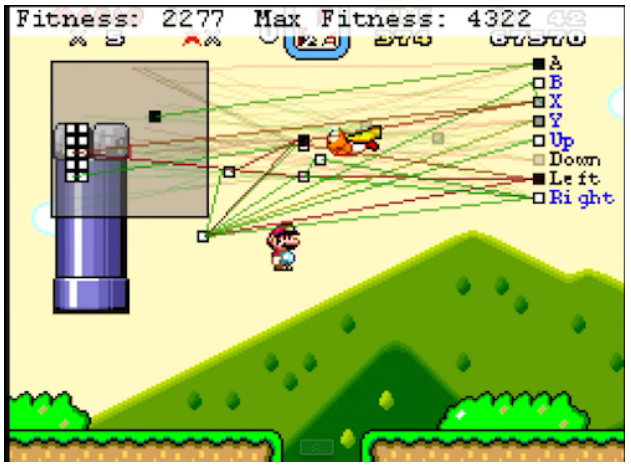


Reinforcement Learning

Most commonly used for designing autonomous robots and automated vehicles

Can also be trained to play games





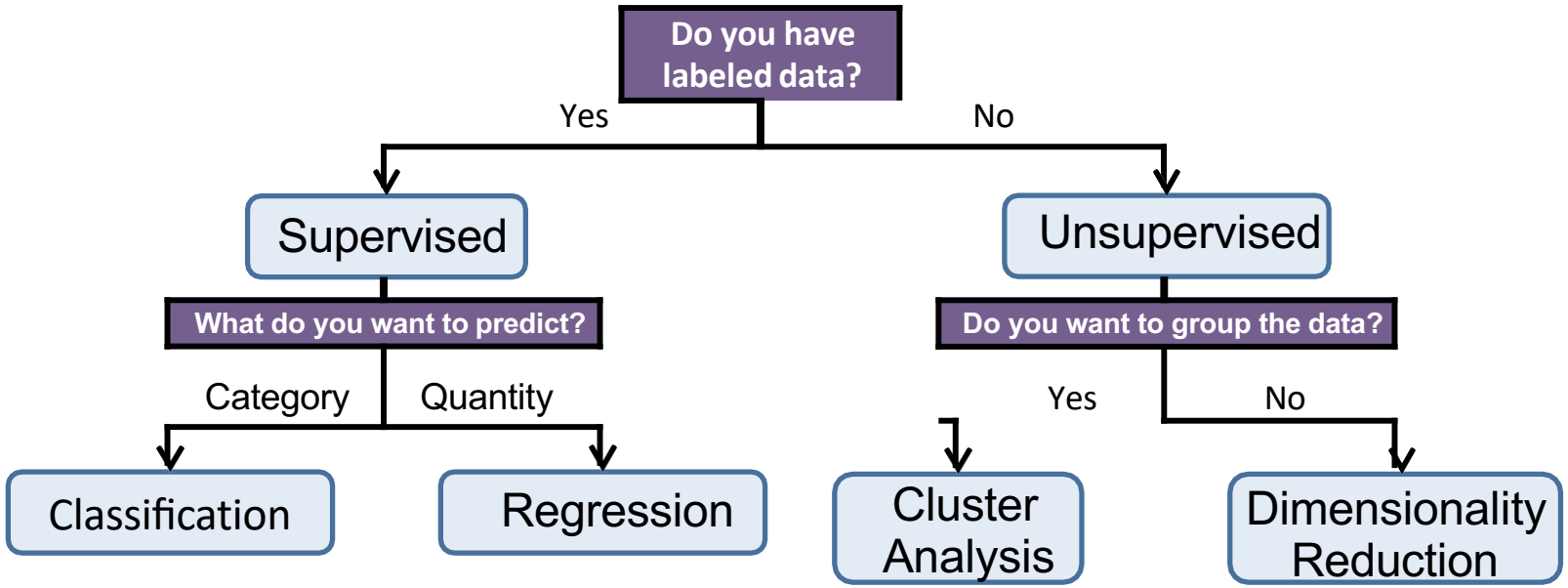
Reinforcement Learning

Some uses in more traditional machine learning tasks by creatively defining what the agent and environment are.

Example: Formulating classification as a reinforcement learning problem!

https://users.cs.duke.edu/~parr/icml03.pdf

Types of Learnimg Algorithms



Terminology

Each data point (i.e., each “thing” you are classifying/regressing/clustering) is called an **instance**

- Alternative name: **observation**
- Also called **examples** or **samples** when used as training data in supervised learning

In a data set, usually each row corresponds to an instance.

Terminology

The “input” variables are called **features**

- Alternative names: **attributes**, **covariates, predictors**
- Also referred to as the **independent** variables

In a data set, each column corresponds to a feature. (Except for the last column, which is the output.)

The list of feature values for an instance is called the instance’s **feature vector**

Terminology

The value of the “output” variable (the “thing” you are trying to predict) is the **label**

- Also called the **dependent** variable

In a data set, this is the final column. (Unless there is more than one label, which is a setting we will consider later in the course.)

In classification, the possible values the labels can have are called **classes**

Terminology

In supervised learning:

- a **training instance** is a feature vector paired with a label
- the **training data** (sometimes **labeled data**) is the table of all training instances

In unsupervised learning, the data set contains feature vectors but no labels (sometimes called **unlabeled data**)

Problem set-up: Formalization

- X: *input variables* (predictors, independent variables, or features)
- Y: *output variable* (response or dependent variable)
- *Statistical learning*: a set of approaches for estimating function f that describes the relationship between predictors and response:

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

Prediction and Inference

- *Prediction*: predict the output Y given inputs X using model $\hat{f}$ an estimator for f
  - Prediction accuracy depends on
    - *Reducible error*: imperfect estimate for f
    - *Irreducible error*: ϵ, error not predicted by the inputs
- *Inference*: understand relationship between Y and individual predictors, X_i
  - In inference, we do not want our model to be a black box

Advertising Example

- A company can not directly control sales of product Z , but can control advertising strategy
- Data: sales and advertising budgets for three media (TV, radio, newspaper)
- Shown are Sales vs TV, Radio and Newspaper, with a blue linear-regression line fit separately to each.

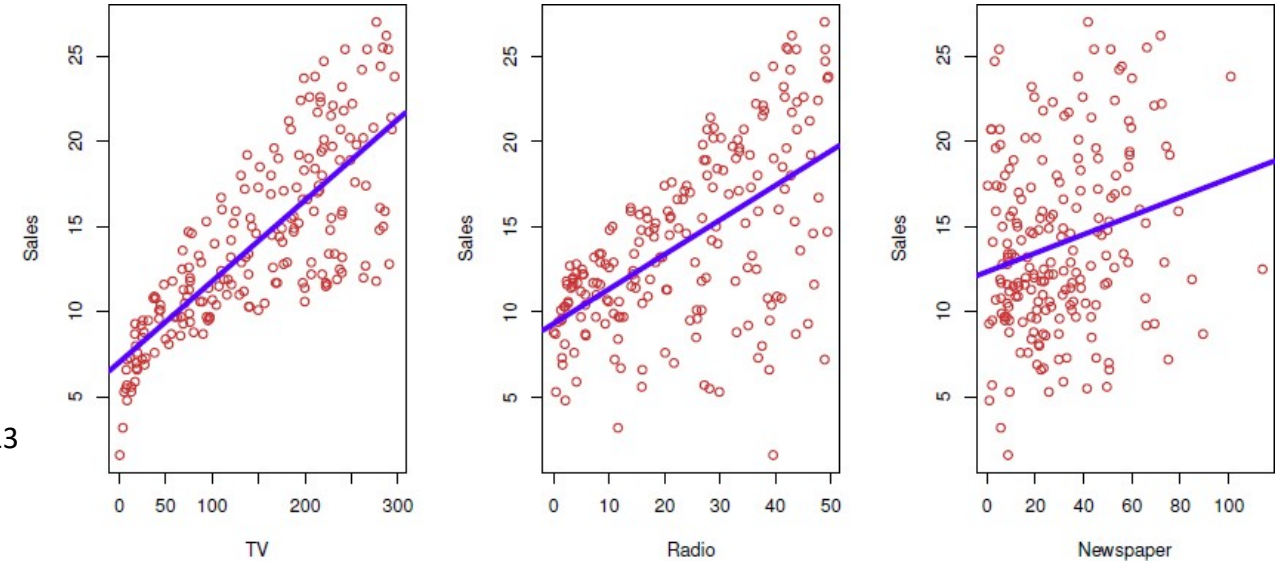


Figure 2.1 , ISL 2013

- In the advertising example, what are the input and output variables?

- Give an example of a prediction question and an inference question that we could attempt to answer with this data.

- In the advertising example, what are the input and output variables?
  - *Output variable*: number of sales
  - *Input variables*: TV advertising budget, Radio advertising budget, Newspaper advertising budget

- Give an example of a prediction question and an inference question that we could attempt to answer with this data.
  - Prediction:
    - What are the expected sales in market Z, given its TV, radio and newspaper budgets?
  - Inference:
    - How much is the increase in sales associated with a 10% increase in the TV advertising budget?
    - Which media (TV, radio, newspaper) generates the largest boost in sales?

Advertising Example

Can we predict Sales using these three? Perhaps we can do better using a model

$$\text{Sales} \approx f(\text{TV}, \text{Radio}, \text{Newspaper})$$

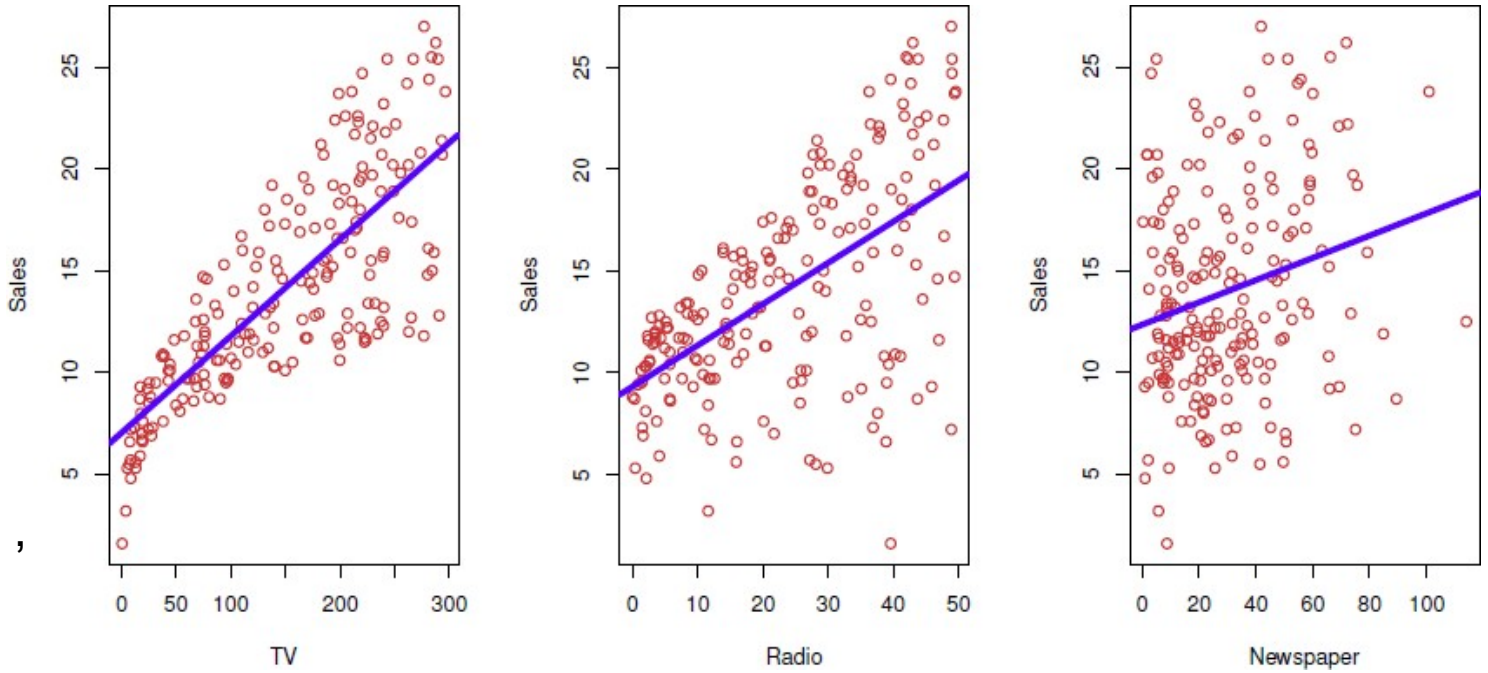


Figure 2.1 , ISL 2013

Notation

Here Sales is a *response* or *target* that we wish to predict. We generically refer to the response as Y. TV is a *feature*, or *input*, or *predictor*; we name it X_1. Likewise name Radio as X_2, and so on. We can refer to the *input vector* collectively as

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

Now we write our model as

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

where ε captures measurement errors and other discrepancies.

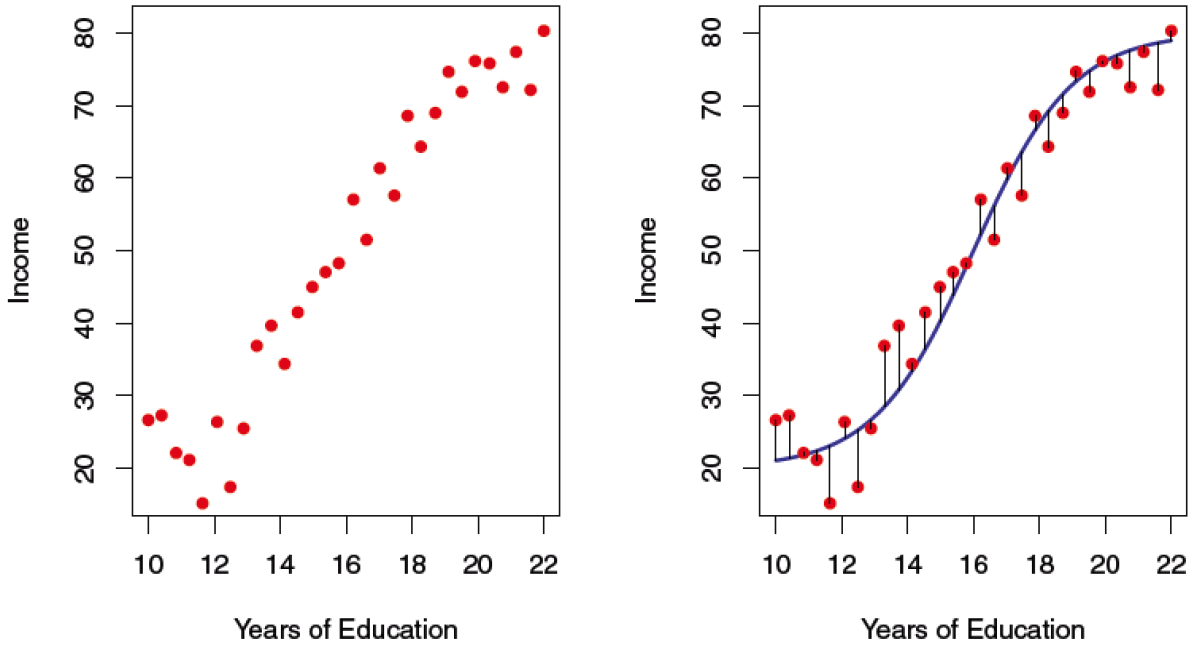
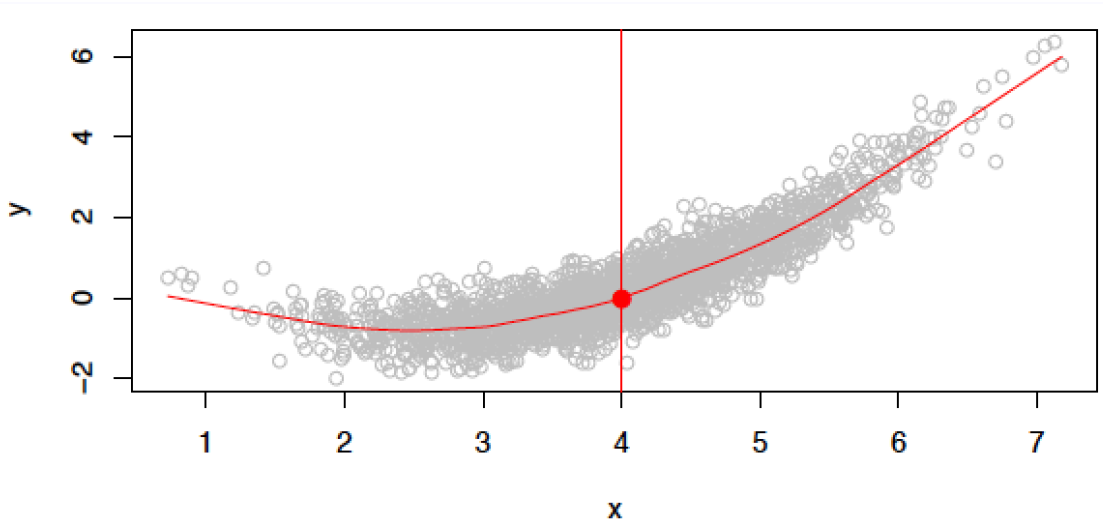


FIGURE 2.2. *The* `Income` *data set.* Left: *The red dots are the observed values of* `income` *(in tens of thousands of dollars) and* `years of education` *for* 30 *individuals.* Right: *The blue curve represents the true underlying relationship between* `income` *and* `years of education`*, which is generally unknown (but is known in this case because the data were simulated). The black lines represent the error associated with each observation. Note that some errors are positive (if an observation lies above the blue curve) and some are negative (if an observation lies below the curve). Overall, these errors have approximately mean zero.*

What is $f(X)$ good for?

- With a good f we can make predictions of Y at new points $X = x$.
- We can understand which components of $X = (X_1, X_2, \ldots, X_p)$ are important in explaining Y, and which are irrelevant. e.g. Seniority and Years of Education have a big impact on Income, but Marital Status typically does not.
- Depending on the complexity of f, we may be able to understand how each component X_j of X affects Y.



Is there an ideal $f(X)$? In particular, what is a good value for $f(X)$ at any selected value of X, say $X = 4$? There can be many Y values at $X = 4$. A good value is

$$f(4) = E(Y \mid X = 4)$$

$E(Y \mid X = 4)$ means *expected value* (average) of Y given $X = 4$.

This ideal $f(x) = E(Y \mid X = x)$ is called the *regression function*.

The regression function $f(x)$

- Is also defined for vector X; e.g.
$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$

- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y|X = x)$ is the function that minimizes $E[(Y - g(X))^2|X = x]$ over all functions g at all points $X = x$.

The regression function $f(x)$

- Is also defined for vector X; e.g. $f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$

- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y|X = x)$ is the function that minimizes $E[(Y - g(X))^2|X = x]$ over all functions g at all points $X = x$.

The regression function $f(x)$

- $\varepsilon = Y - f(x)$ is the *irreducible* error — i.e. even if we knew $f(x)$, we would still make errors in prediction, since at each $X = x$ there is typically a distribution of possible Y values.
- For any estimate $\hat{f}(x)$ of $f(x)$, we have

$$E[(Y-\hat{Y})^2 \mid X=x] = E[(f(X)+\varepsilon-\hat{f}(X))^2 \mid X=x]$$

$$=$$

- So $$E[(Y-\hat{f}(X))^2 | X=x] = \underbrace{[f(x)-\hat{f}(x)]^2}_{Reducible} + \underbrace{\mathrm{Var}(\epsilon)}_{Irreducible}$$

The regression function $f(x)$

- In this course, we study methods to decrease the reducible error $f(x)$-$\hat{f}(x)$.

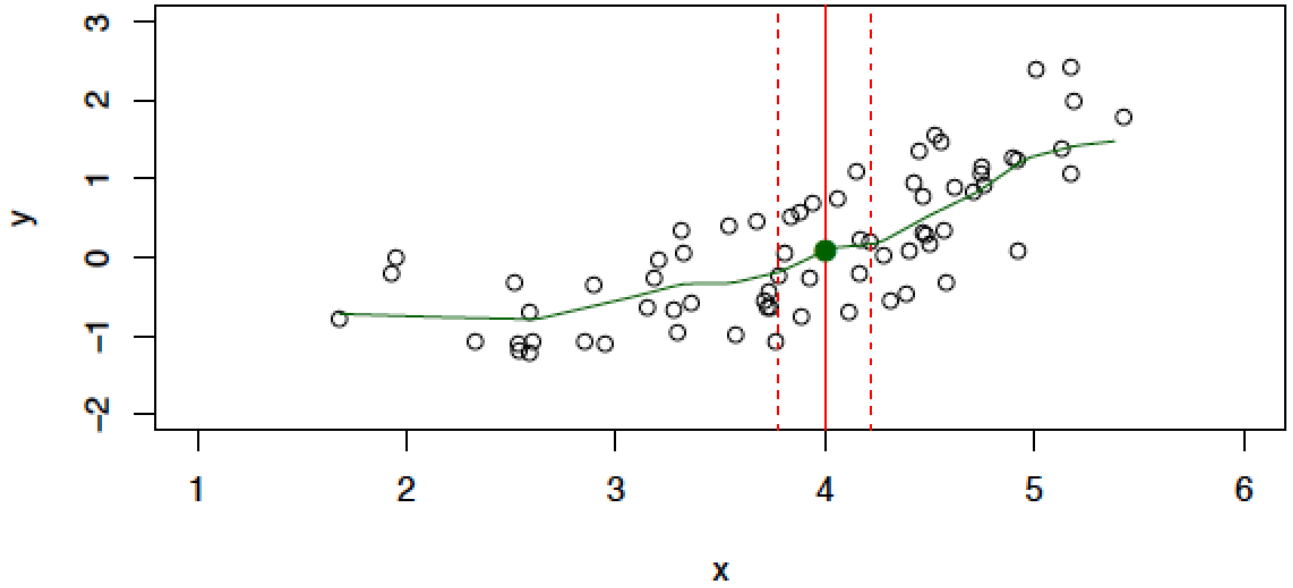
$$E[(Y - \hat{f}(X))^2 | X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{Reducible} + \underbrace{\mathrm{Var}(\epsilon)}_{Irreducible}$$

How to estimate *f*

- Typically we have few if any data points with $X = 4$ exactly.
- So we cannot compute $E(Y|X = x)$.
- Relax the definition and let

$$\hat{f}(x) = \text{Ave}(Y|X \in \text{N}(x))$$

where N(x) is some *neighborhood* of x.



Nearest Neighborhood

- Nearest neighbor averaging can be pretty good for small p — i.e. $p \leq 4$ and large-ish N .
- We will discuss smoother versions, such as kernel and spline smoothing later in the course.

Nearest Neighborhood

- Nearest neighbor methods can be lousy when p is large. Reason: the curse of dimensionality.
- Nearest neighbors tend to be far away in high dimensions.
- NN is a *non-parametric* Method

Nearest Neighborhood

- We need to get a reasonable fraction of the N values of y_i to average to bring the variance down—e.g. 10%.
- A 10% neighborhood in high dimensions need no longer be local, so we lose the spirit of estimating $E(Y|X = x)$ by local averaging.

Parametric and structured models

- The *linear* model is an important example of a parametric model:

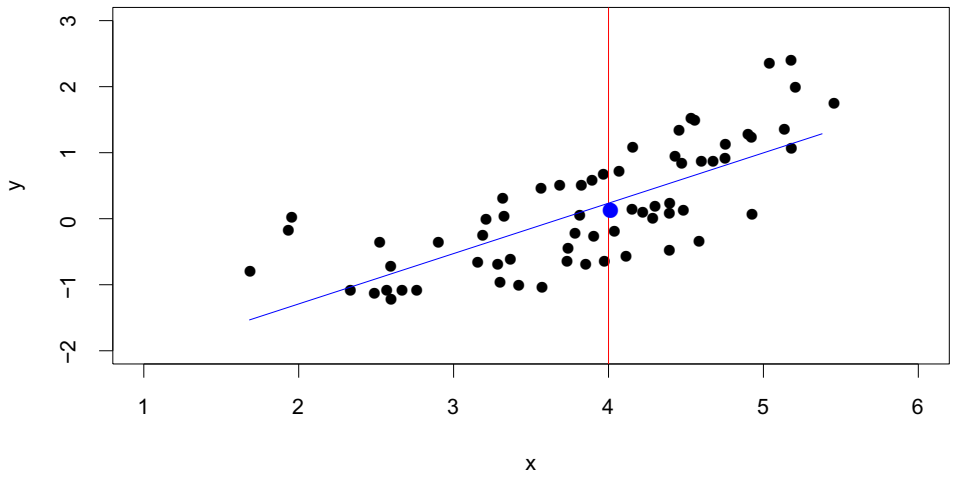
$$f_L(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p.$$

  - A linear model is specified in terms of $p+1$ parameters $\beta_0, \beta_1, \ldots, \beta_p$.

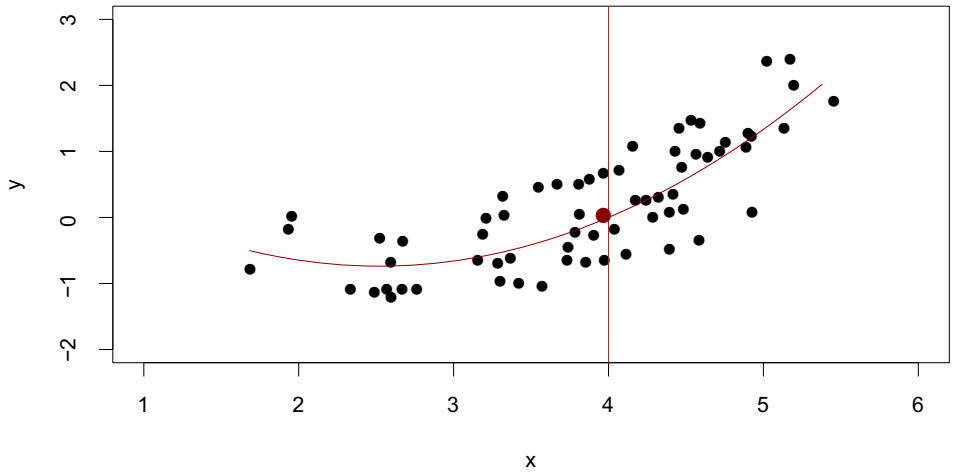
Parametric and structured models

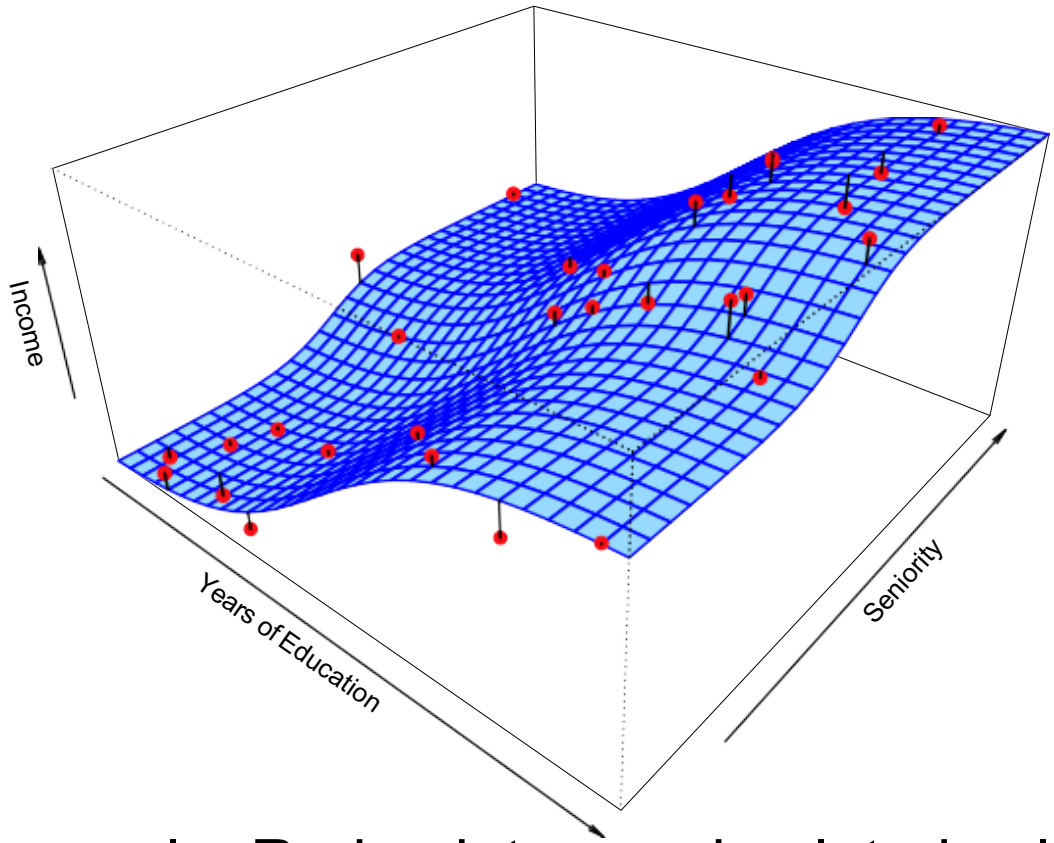
- We estimate the parameters by fitting the model to training data.
- Although it is *almost never correct*, a linear model often serves as a good and interpretable approximation to the unknown true function $f(X)$.

A linear model $\hat{f}_L(X) = \hat{\beta}_0 + \hat{\beta}_1 X$ gives a reasonable fit here



A quadratic model $\hat{f}_Q(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$ fits slightly better.

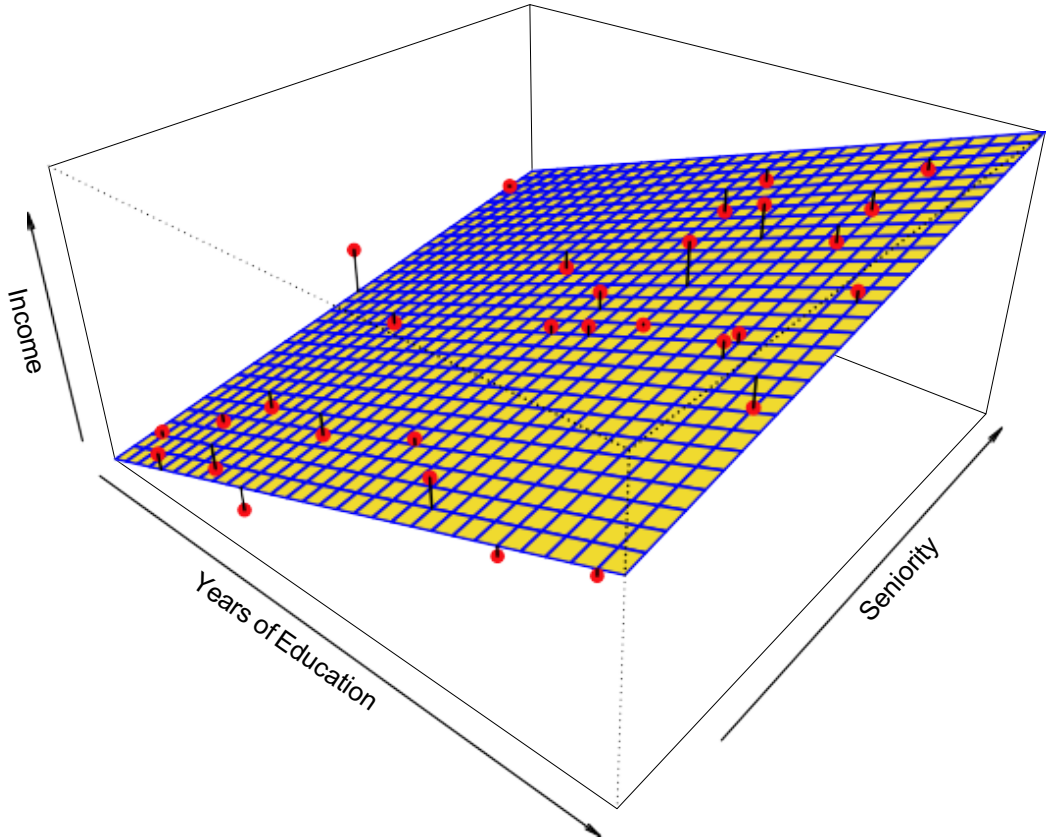




Simulated example. Red points are simulated values for income from the model

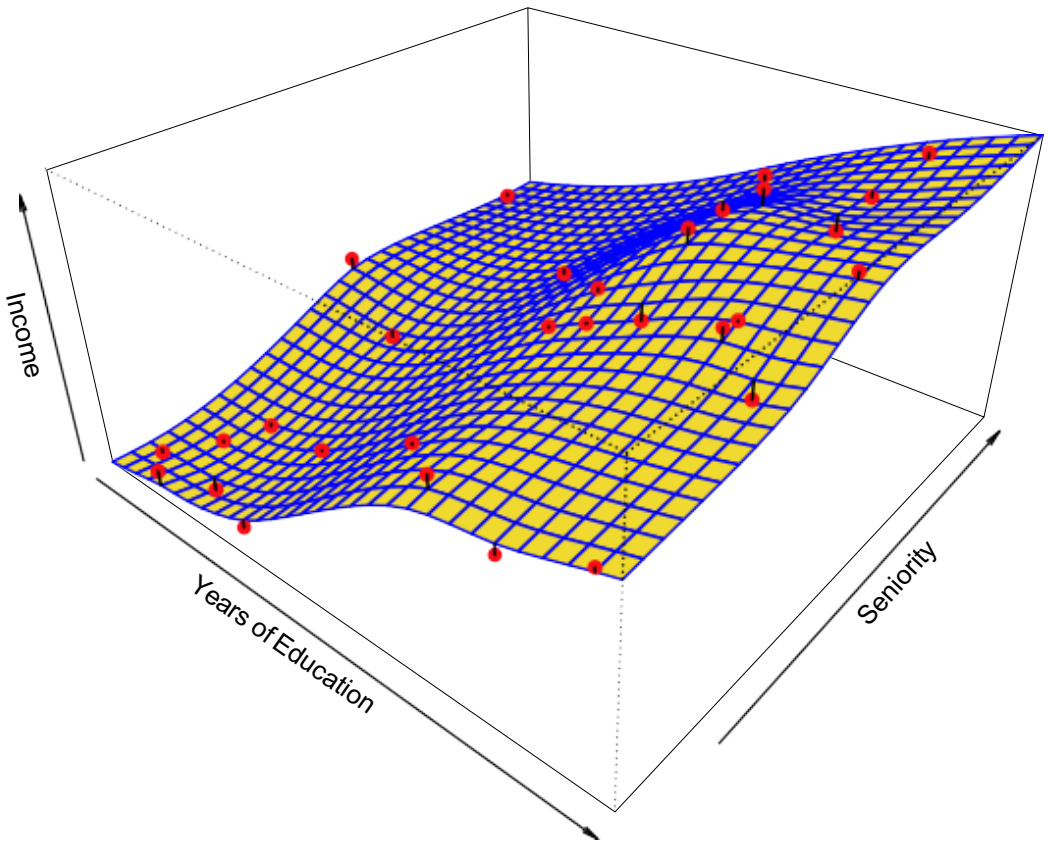
$$\text{income} = f(\text{education}, \text{seniority}) + \epsilon$$

f is the blue surface.

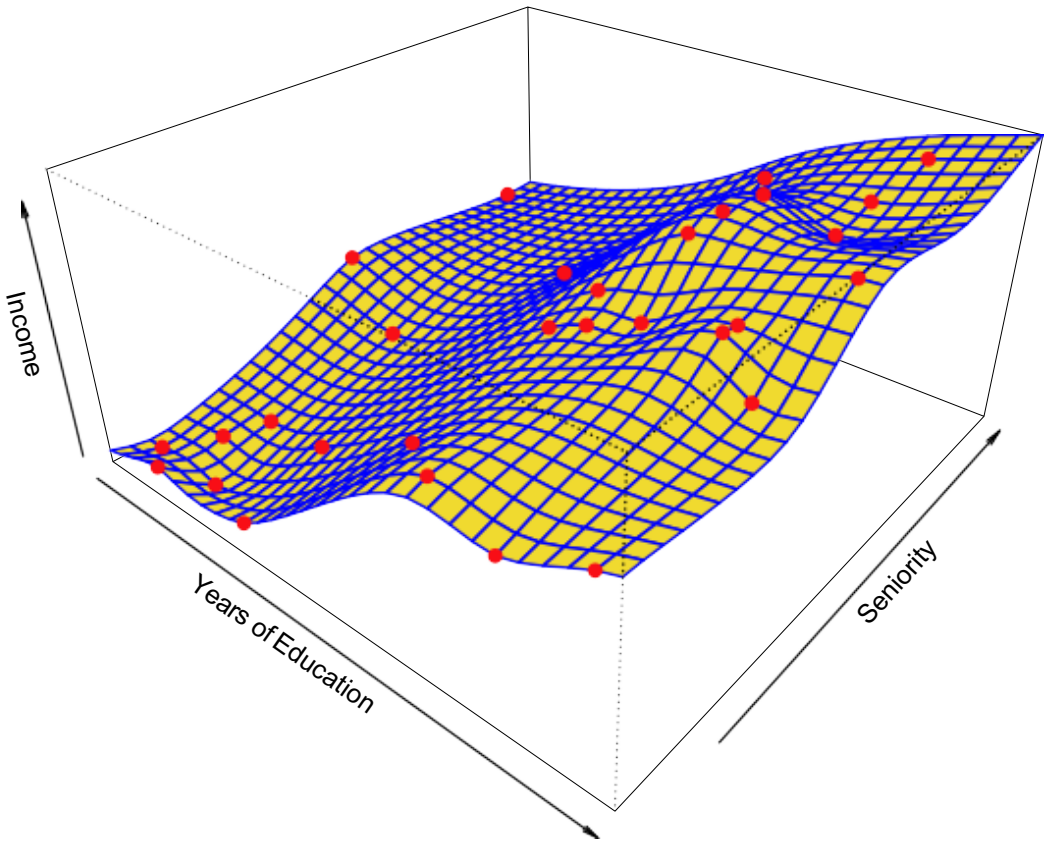


Linear regression model fit to the simulated data.

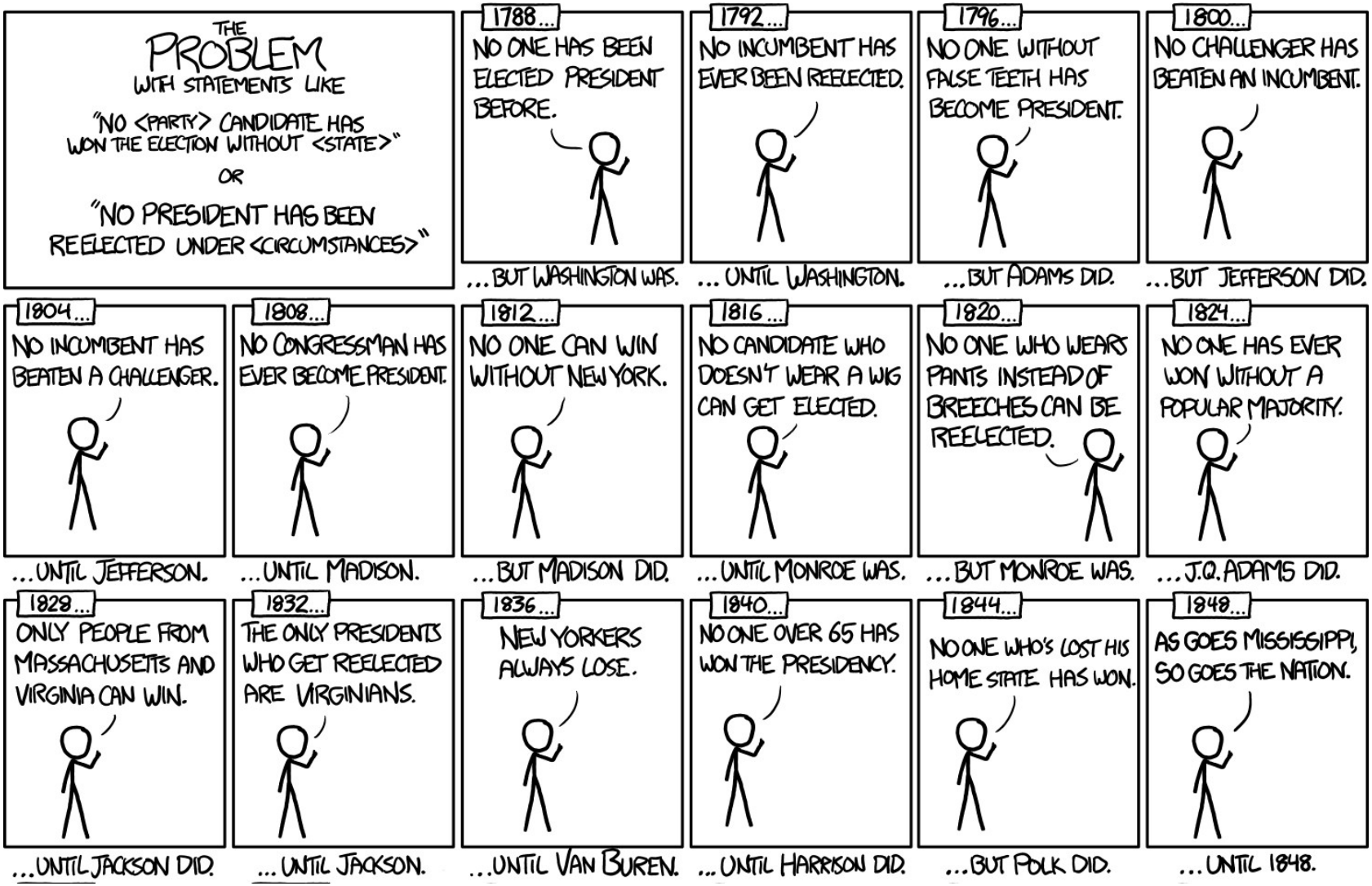
$\hat{f}_L(\text{education}, \text{seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \text{education} + \hat{\beta}_2 \times \text{seniority}$



More flexible regression model $\hat{f}_S$(education, seniority) fit to the simulated data. Here we use a technique called a *thin-plate spline* to fit a flexible surface. We control the roughness of the fit (chapter 7).



Even more flexible spline regression model $\hat{f}_S$ (education, seniority) fit to the simulated data. Here the fitted model makes no errors on the training data! Also known as *overfitting*.



... ...

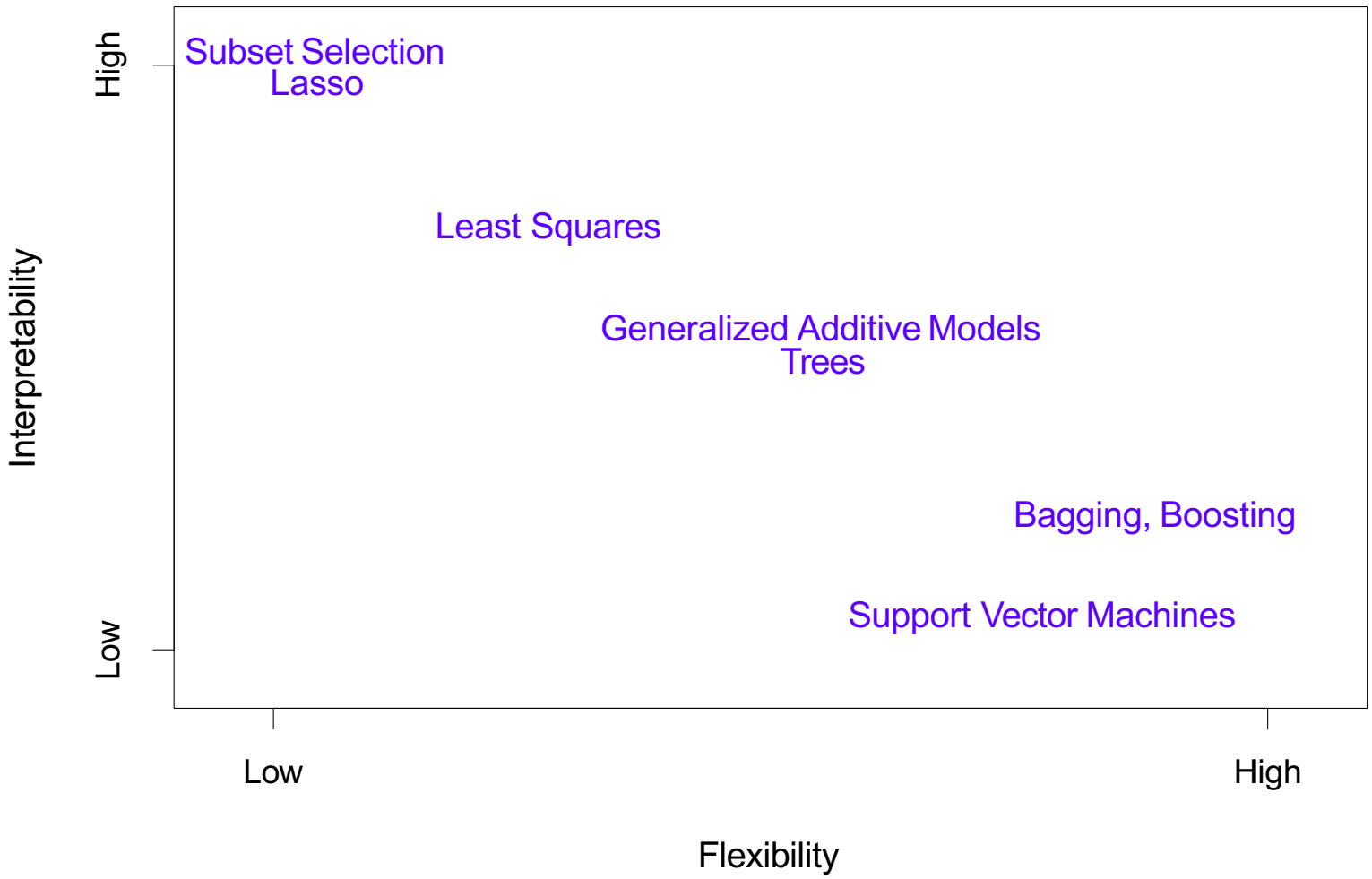
From: https://xkcd.com/1122/

Some trade-offs

- Prediction accuracy versus interpretability.
  —Linear models are easy to interpret; thin-plate splines are not.
- Good fit versus over-fit or under-fit.
  —How do we know when the fit is just right?

Some trade-offs

- Parsimony versus black-box.
  —We often prefer a simpler model involving fewer variables over a black-box predictor involving them all.


Subset Selection
Lasso
Least Squares
Generalized Additive Models
Trees
Bagging, Boosting
Support Vector Machines
High
Low
Interpretability
Low
High
Flexibility

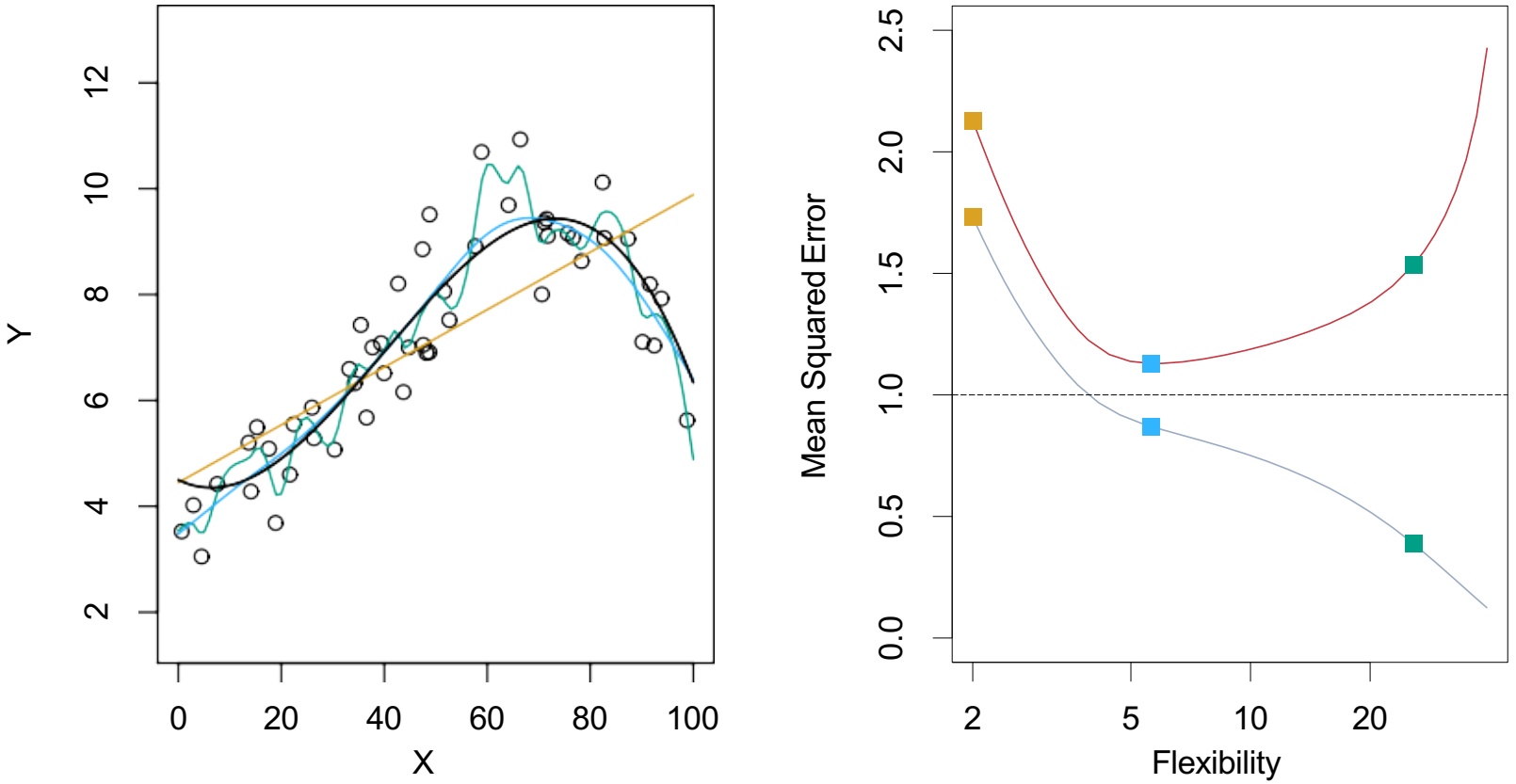
Assessing Model Accuracy

- Suppose we fit a model $\hat{f}(x)$ to some training data $\text{Tr} = \{x_i, y_i\}_1^N$, and we wish to see how well it performs.
- We could compute the average squared prediction error over Tr:

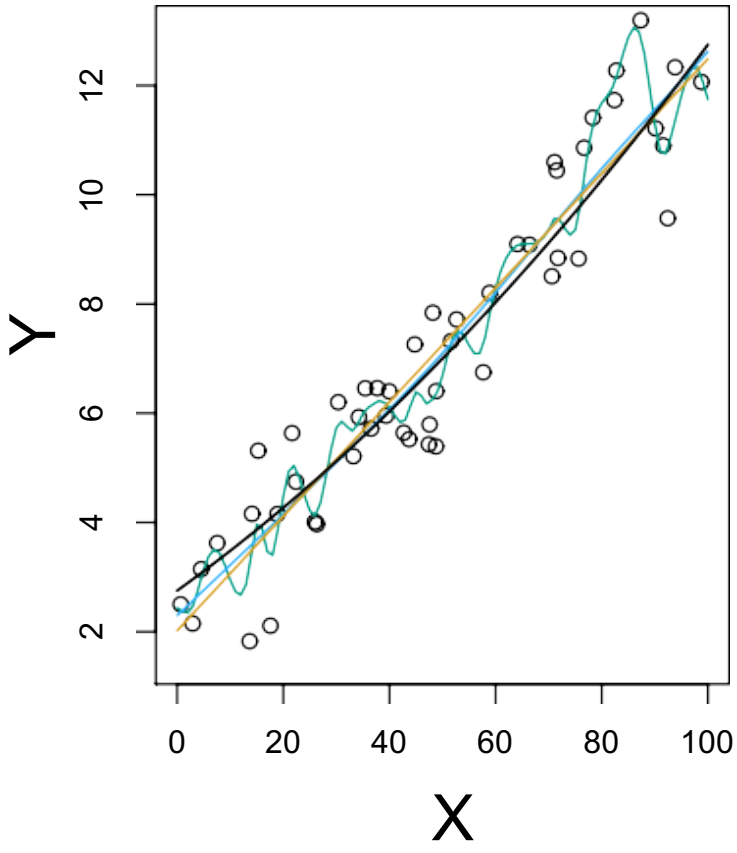
$$\text{MSE}_{\text{Tr}} = \text{Ave}_{i \in \text{Tr}}[y_i - \hat{f}(x_i)]^2$$

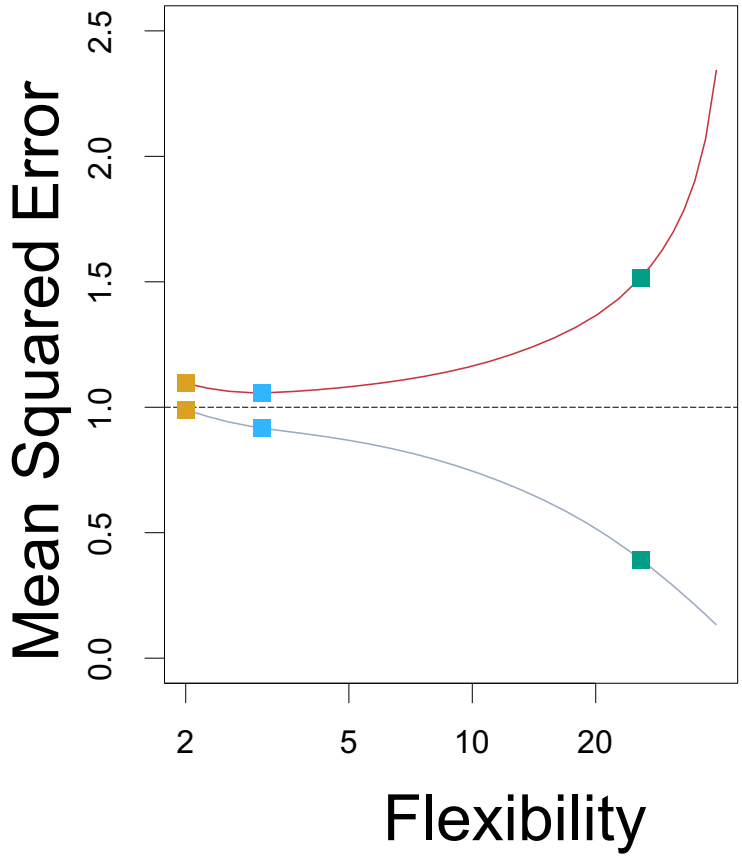
- This may be biased toward more overfit models.
- Instead we should, if possible, compute it using fresh *test* data $\text{Te} = \{x_i, y_i\}_1^M$:

$$\text{MSE}_{\text{Te}} = \text{Ave}_{i \in \text{Te}}[y_i - \hat{f}(x_i)]^2$$

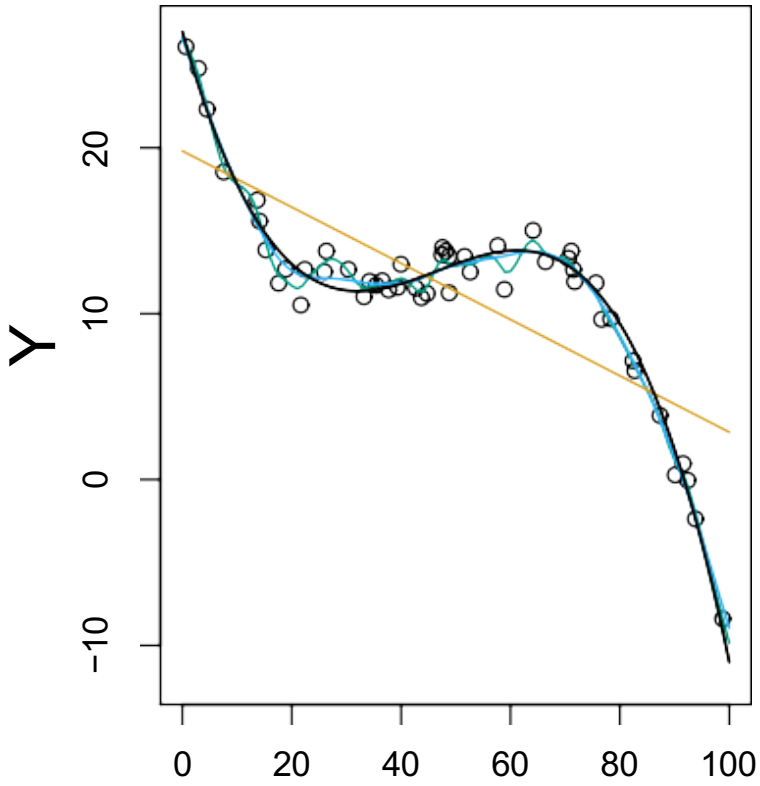


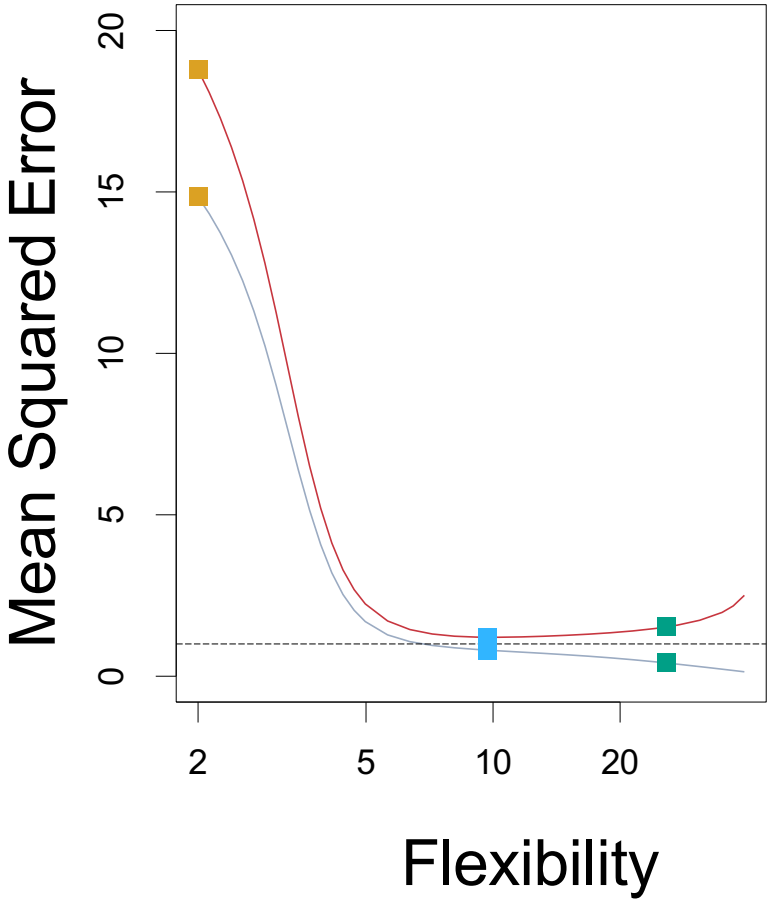
Black curve is truth. Red curve on right is MSE_{Te}, grey curve is MSE_{Tr}. Orange, blue and green curves/squares correspond to fits of different flexibility.





Here the truth is smoother, so the smoother fit and linear model do really well.





Here the truth is wiggly and the noise is low, so the more flexible fits do the best.

Bias-Variance Trade-off

Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \varepsilon$ (with $f(x) = E(Y|X = x)$), then

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$

Bias-Variance Trade-off

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$

The expectation averages over the variability of y_0 as well as the variability in Tr. Note that

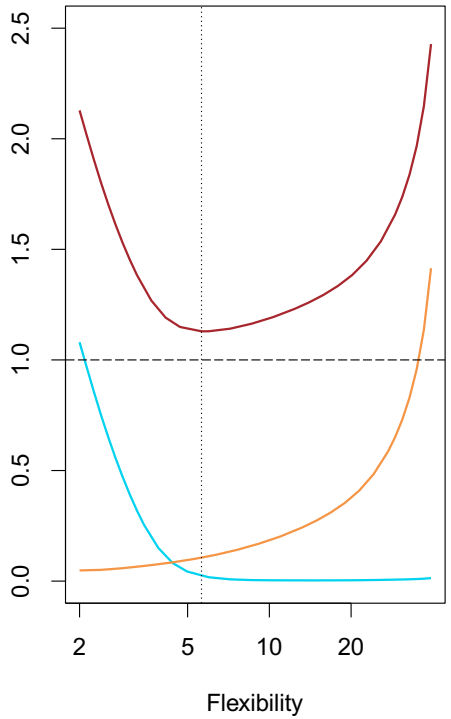
Bias[$\hat{f}(x_0)$)] = $E[\hat{f}(x_0)] - f(x_0)$.

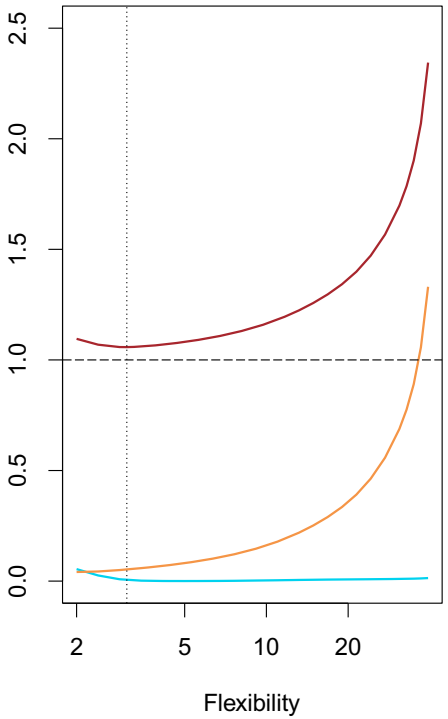
Typically as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off*.

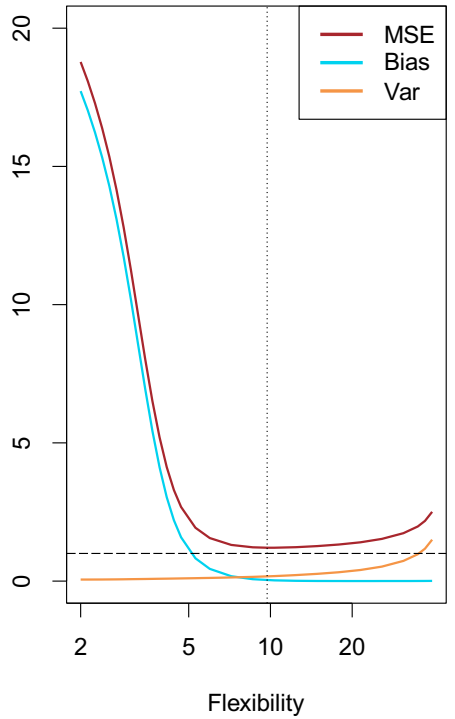
Bias-Variance Trade-off

Bias-Variance Trade-off

Bias-variance trade-off for the three examples



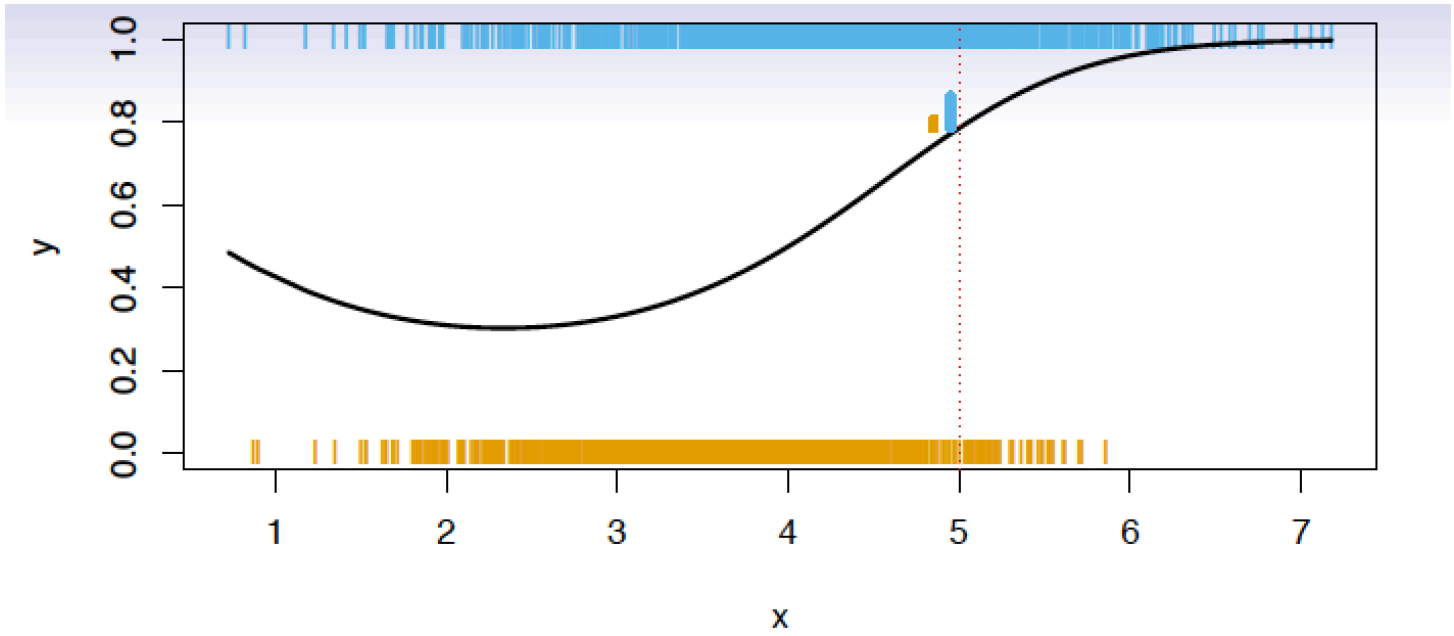




Classification Problems

Here the response variable Y is *qualitative* — e.g. email is one of C = (spam, ham) (ham=good email), digit class is one of C = {0, 1, . . . , 9}. Our goals are to:

- Build a classifier $C(X)$ that assigns a class label from C to a future unlabeled observation X.
- Assess the uncertainty in each classification
- Understand the roles of the different predictors among $X = (X_1, X_2, \ldots, X_p)$.

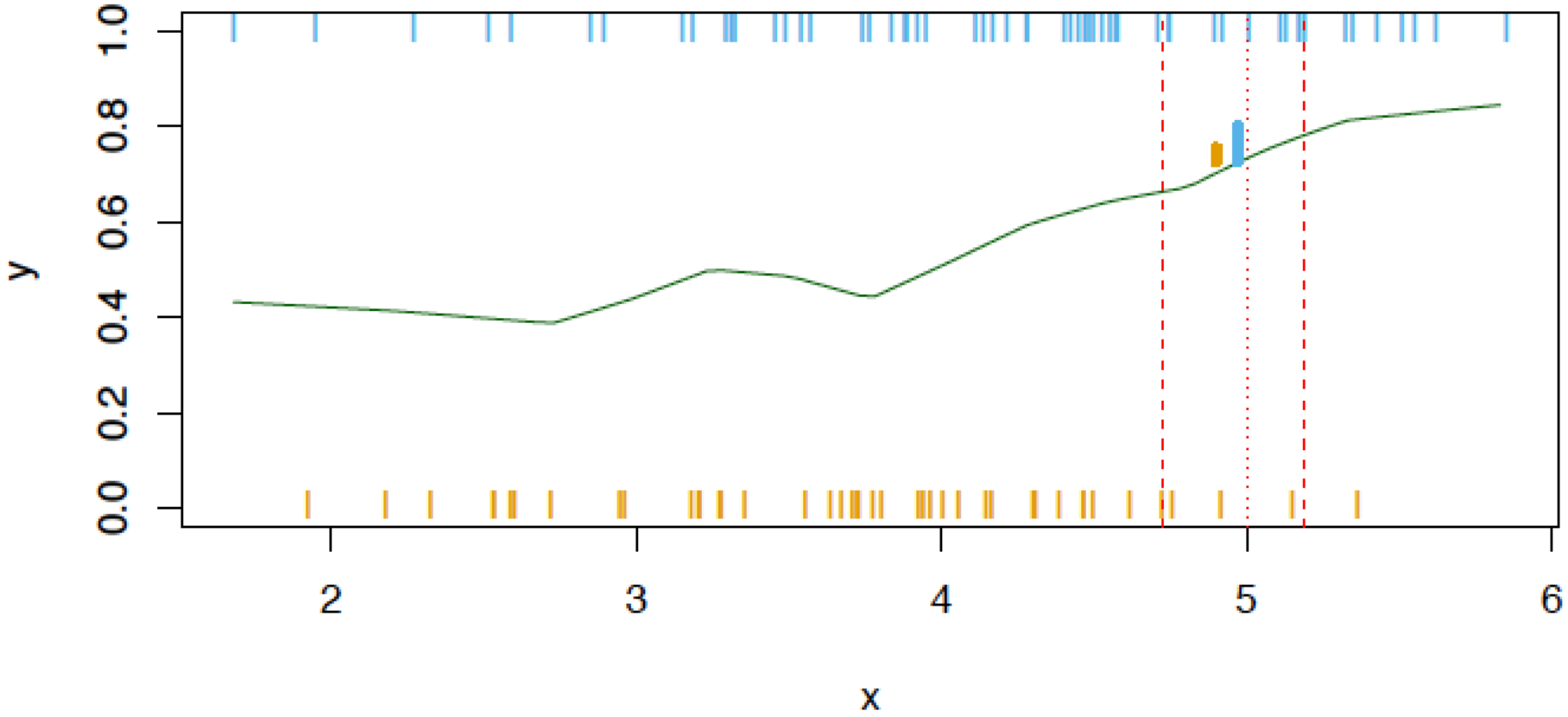


Is there an ideal $C(X)$? Suppose the K elements in C are numbered 1, 2, . . . , K. Let

$p_k(x) = \Pr(Y = k|X = x)$, $k = 1, 2, \ldots, K$.

These are the *conditional class probabilities* at x; e.g. see little barplot at $x = 5$. Then the *Bayes optimal* classifier at x is

$C(x) = j$ if $p_j(x) = \max\{p_1(x), p_2(x), \ldots, p_K(x)\}$



Nearest-neighbor averaging can be used as before.

Also breaks down as dimension grows.

However, the impact on $\hat{C}(x)$ is less than on $\hat{p}_k(x)$, $k = 1, \ldots, K$.

Classification: some details

- Typically we measure the performance of $\hat{C}(x)$ using the misclassification error rate:

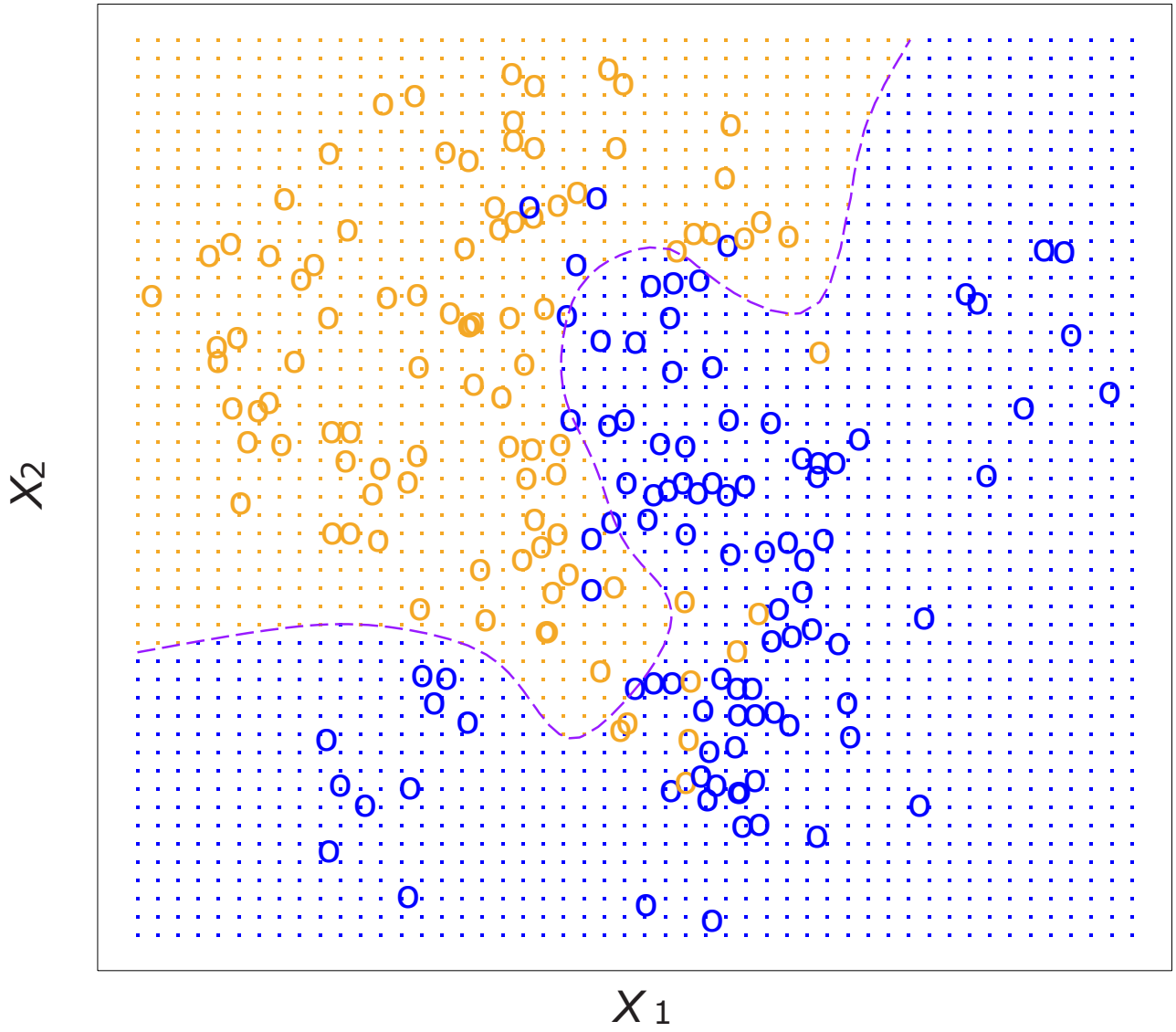
$$\text{Err}_{\text{Te}} = \text{Ave}_{i \in \text{Te}} \boldsymbol{I}[y_i \neq \hat{C}(x_i)]$$

- The Bayes classifier (using the true $p_k(x)$) has smallest error (in the population).

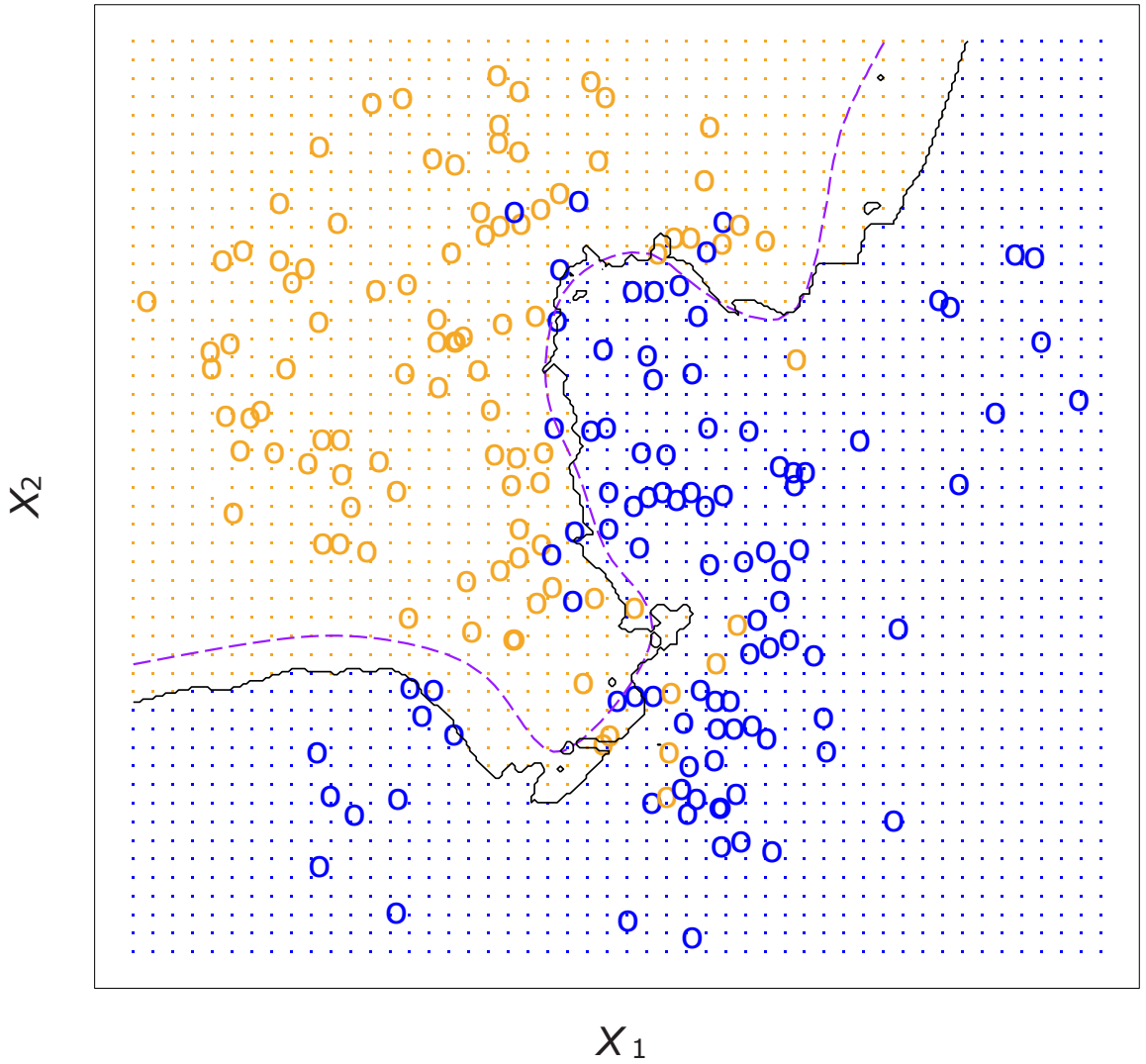
Classification: some details

- Support-vector machines build structured models for $C(x)$.
- We will also build structured models for representing the$p_k(x)$. e.g. Logistic regression, generalized additive models.

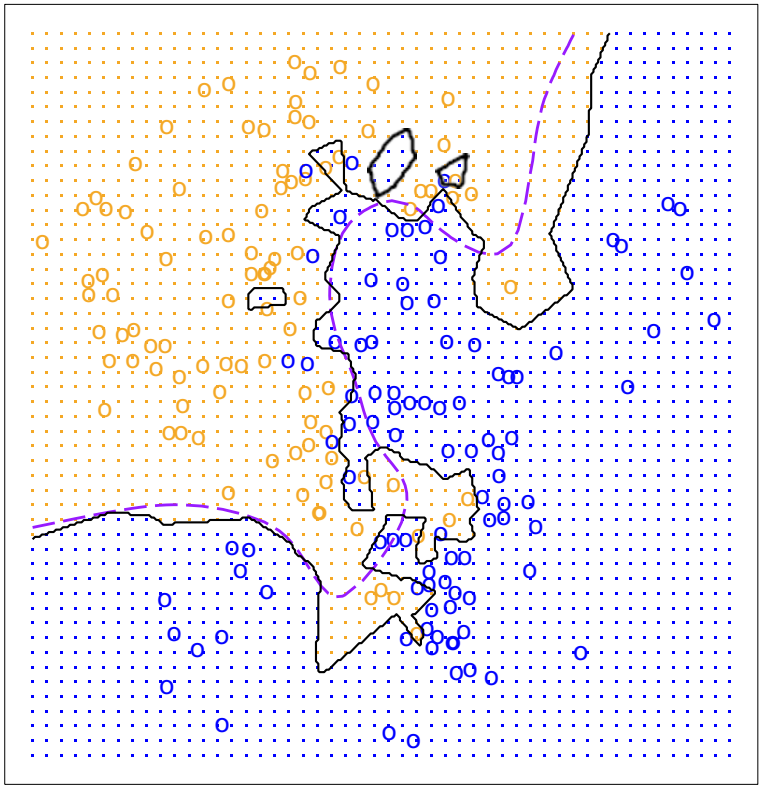
Example: K-nearest neighbors in two dimensions



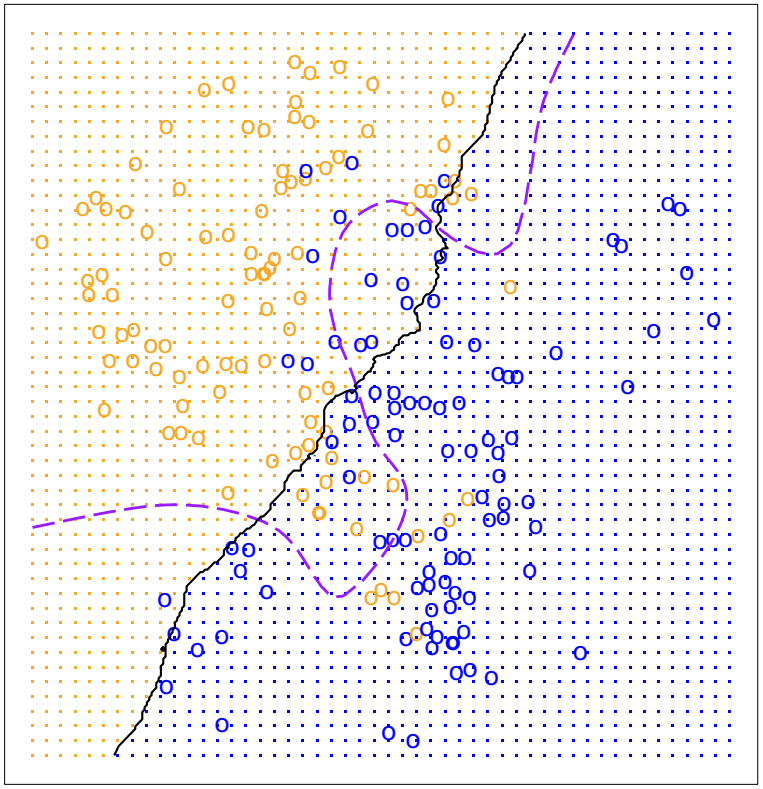
KNN: K=10



KNN: K=1

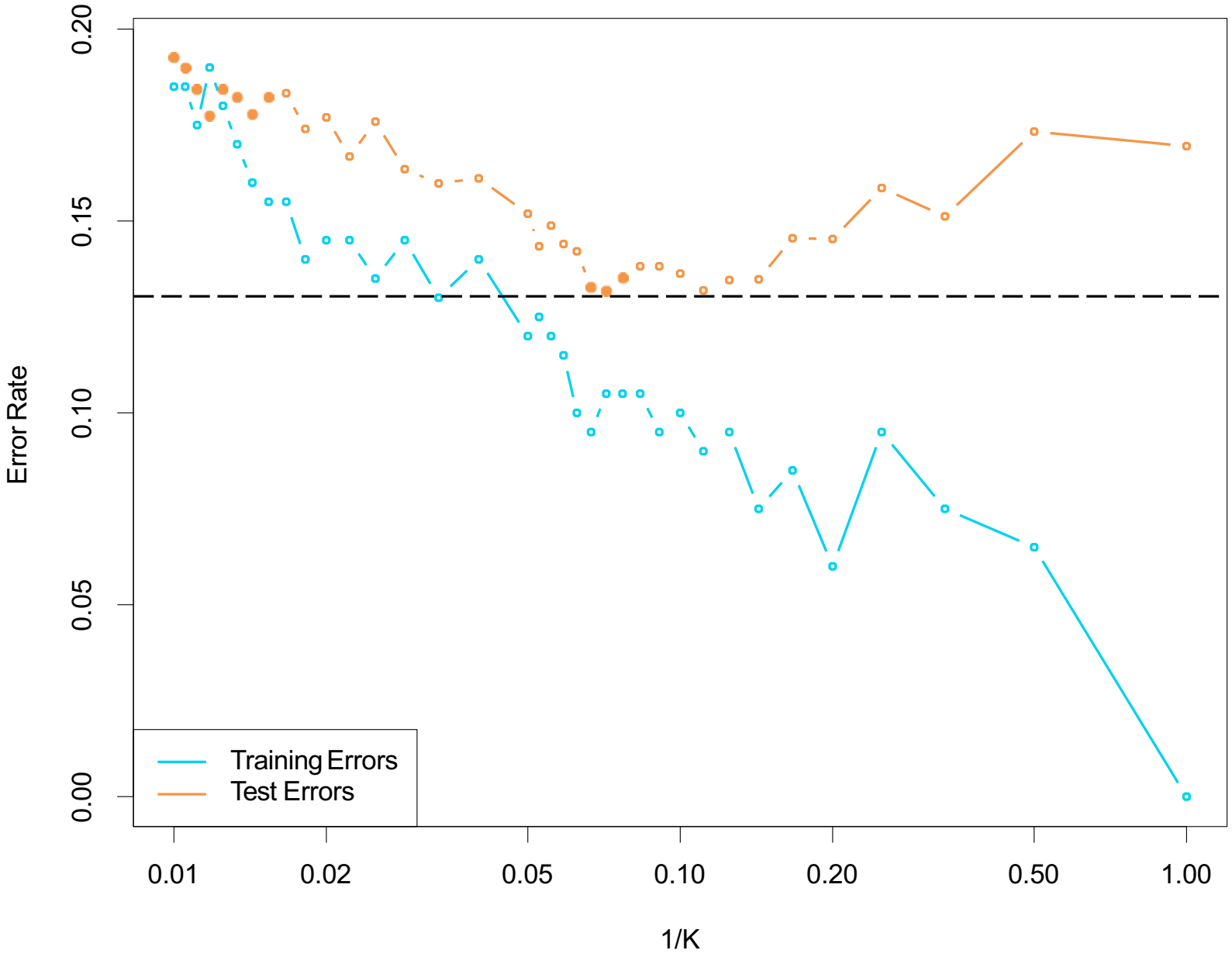


KNN: K=100



K-Nearest Neighbor classifier (KNN)

- Choice of K
  - K small → high variance and low bias
  - Large K → low variance and high bias


Error Rate
0.00
0.05
0.10
0.15
0.20
0.01
0.02
0.05
0.10
0.20
0.50
1.00
1/K
Training Errors
Test Errors

K-Nearest Neighbor classifier (KNN)

- Advantages:
  - Simple to implement
  - Few tuning parameters (K, distance metric)
  - Flexible, classes do not have to be linearly separable

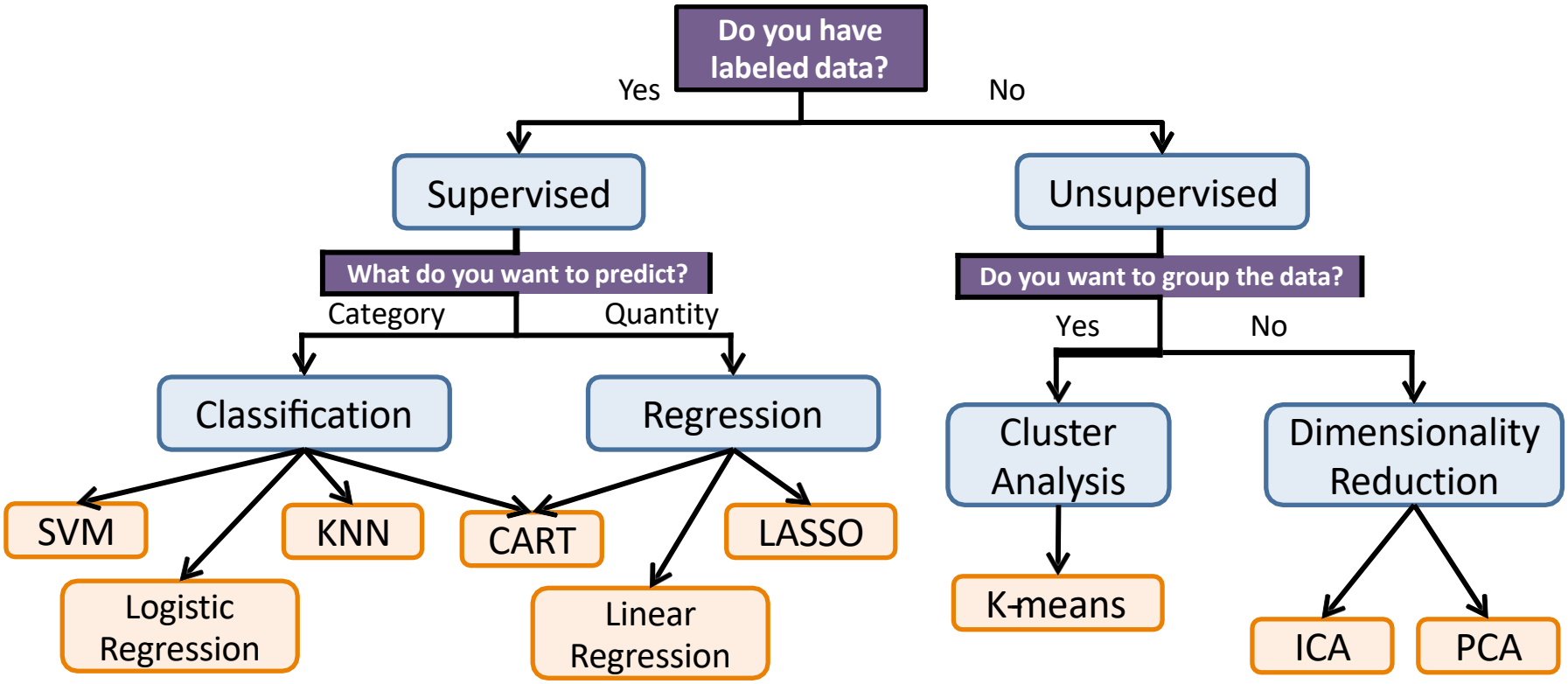
K-Nearest Neighbor classifier (KNN)

- Disadvantages:
  - Computationally expensive – must compute distance from new observation to all known samples
  - Sensitive to imbalanced datasets – may get poor results for infrequent classes
  - Sensitive to irrelevant inputs – these make distances less meaningful for identifying similar neighbors

K--Nearest Neighbor classifier (KNN)

- It is a very well-known example of "**Lazy Learning Algorithms**" in which generalization beyond the training data is delayed until a query is made to the system, as opposed to in **Eager Learning**, where the system tries to generalize the training data before receiving queries.

Types of Algorithms



“Best” Machine Learning Algorithm

- Bad news: no algorithm is the best
  - No machine learning algorithm will perform well on every task / data set
- Good news: all of them are the best
  - Each machine learning algorithm will perform well on some task / dataset

“Best” Machine Learning Algorithm

- “No free lunch” theorem
  - Wolpert (1996): all algorithms perform equally when averaged over all possible problems
  - Alternatively: Generalization can only be obtained by assumptions

“Best” Machine Learning Algorithm

- “No free lunch” theorem

Trade--offs in Machine Learning

- Accuracy vs. interpretability
- Bias vs. variance
- Complexity vs. scalability
  - Some models / algorithms for computing them may not scale to large data sets
- Domain-knowledge vs. data-driven
- More data vs. better algorithm

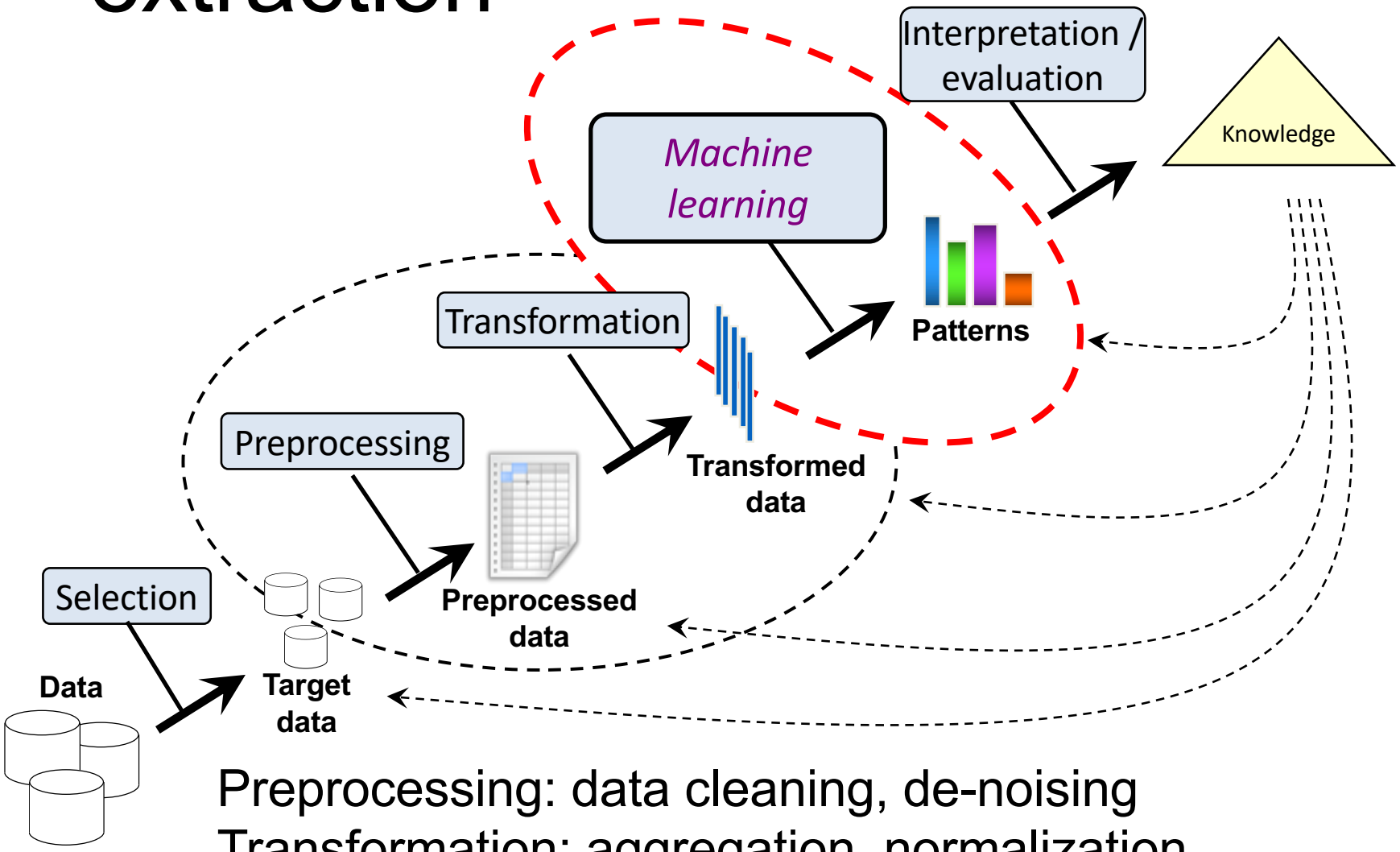
Preparing Data

- Machine learning algorithms require data!
- Preprocessing is often necessary to transform the data prior to applying a learning algorithm
  - Sampling: selecting a subset of observations
  - *Feature extraction*: selecting input variables
  - *Normalization* (standardization, scaling, binarization)
  - Handling missing data and outliers

Preparing Data

- Decision trees can handle missing data / outliers
- PCA requires data to be standardized (zero--mean)

Stages of knowledge extraction



Preprocessing: data cleaning, de-noising
Transformation: aggregation, normalization, feature creation, feature selection

Supervised Machine Learning