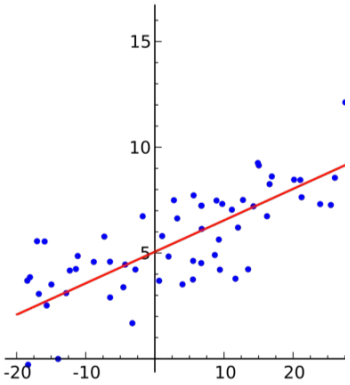
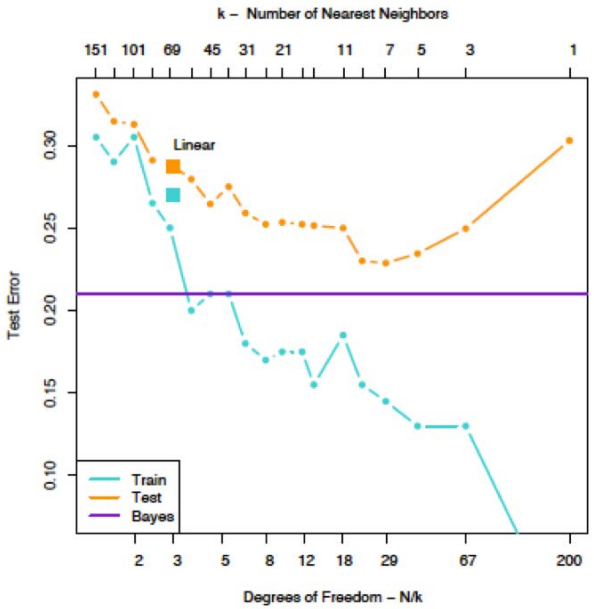
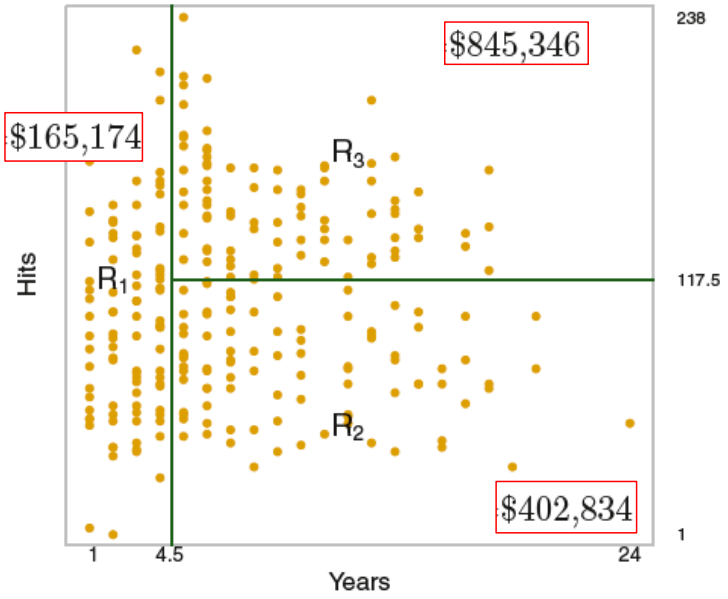
Tufts COMP 135: Introduction to Machine Learning

https://www.cs.tufts.edu/comp/135/2019s/

Regression







Many slides attributable to:
Erik Sudderth (UCI)
Finale Doshi-Velez (Harvard)
James, Witten, Hastie, Tibshirani (ISL/ESL books)

Prof. Mike Hughes

Logistics

- HW0 due TONIGHT (Wed 1/23 at 11:59pm)

- HW1 out later tonight, due a week from today
  - What you submit: PDF and zip

- Next recitation is Mon 1/28
  - Multivariate Calculus review
  - The gory math behind linear regression

Regression Unit Objectives

- 3 steps of a regression task
  - Training
  - Prediction
  - Evaluation
    - Metrics
    - Splitting data into train/valid/test

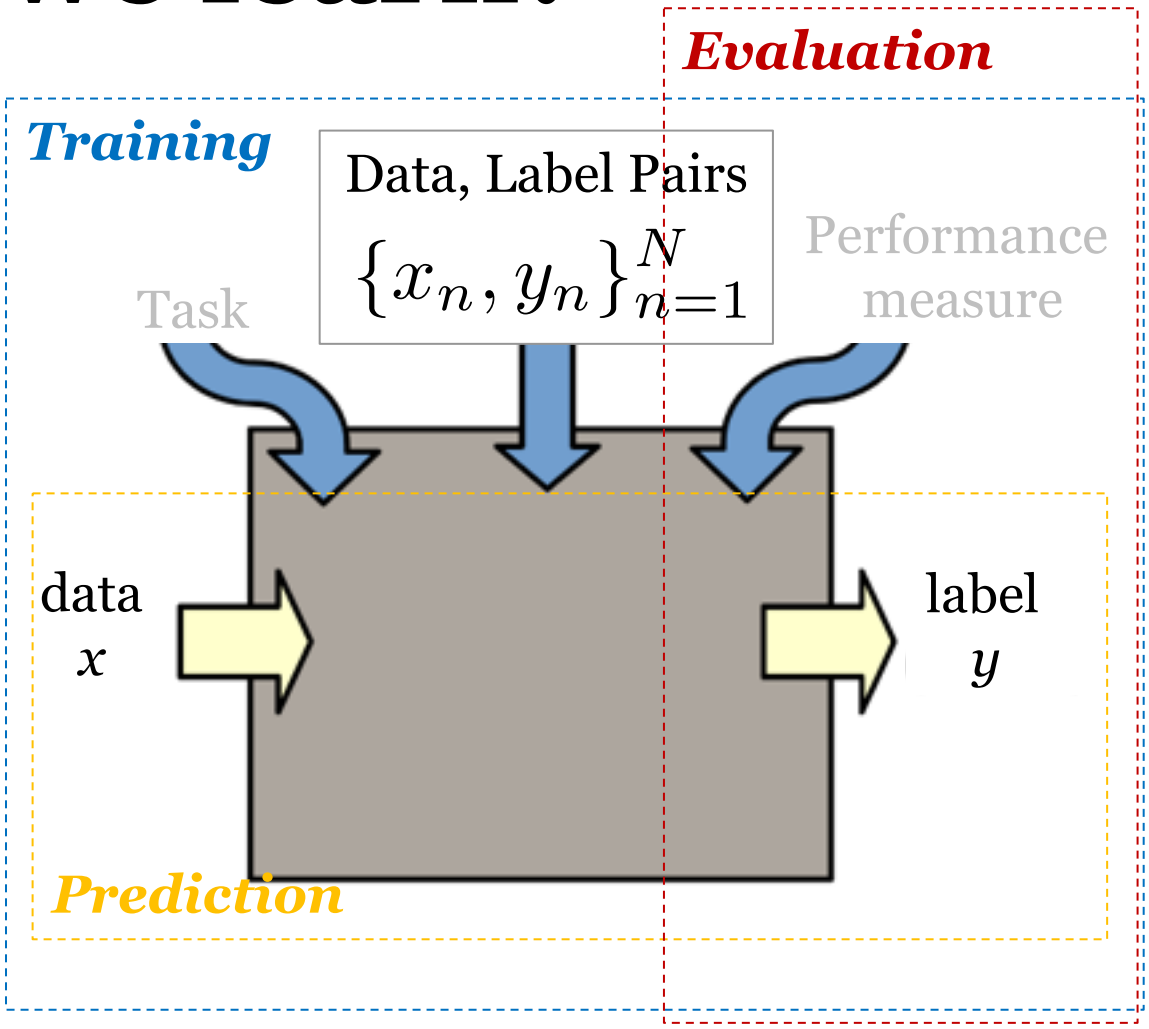
- A "taste" of 3 Methods
  - Linear Regression
  - K-Nearest Neighbors
  - Decision Tree Regression

What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



Task: Regression

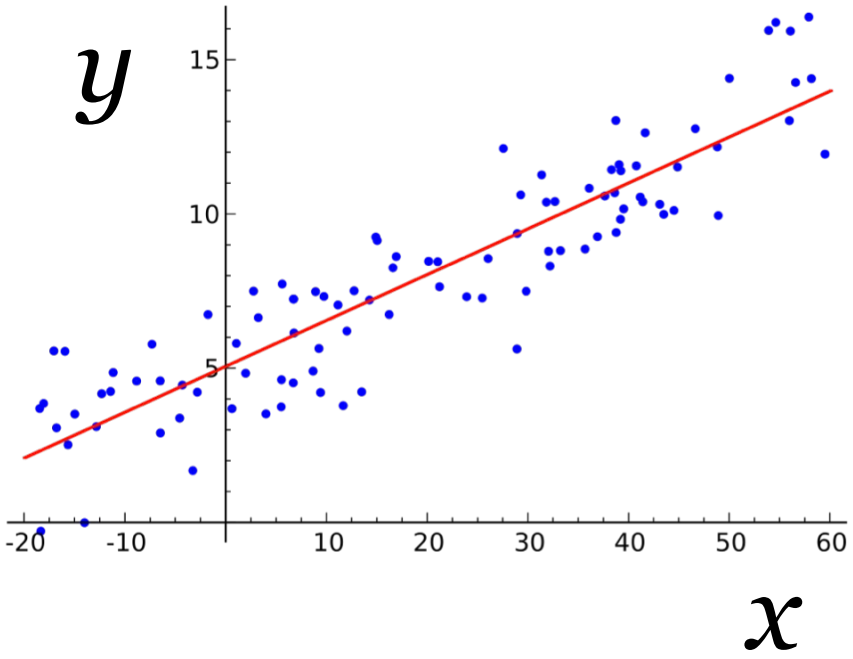
Supervised Learning

regression

Unsupervised Learning

Reinforcement Learning

y is a numeric variable
e.g. sales in $$



Regression Example: Uber

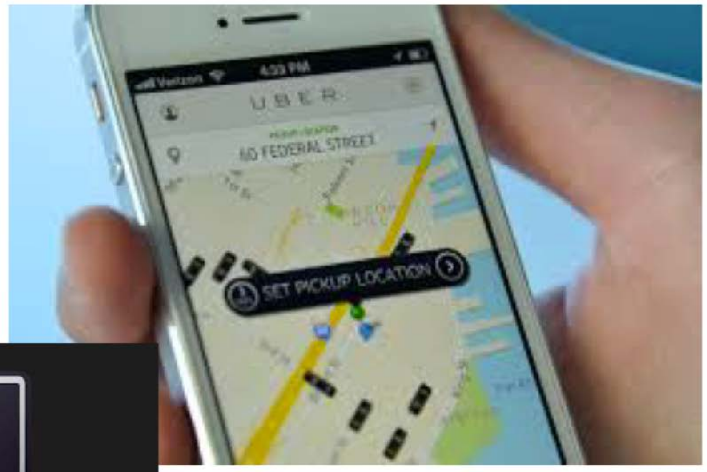
Supervised Learning

regression

Unsupervised Learning

Reinforcement Learning

Predictions of travel time, price, supply, demand

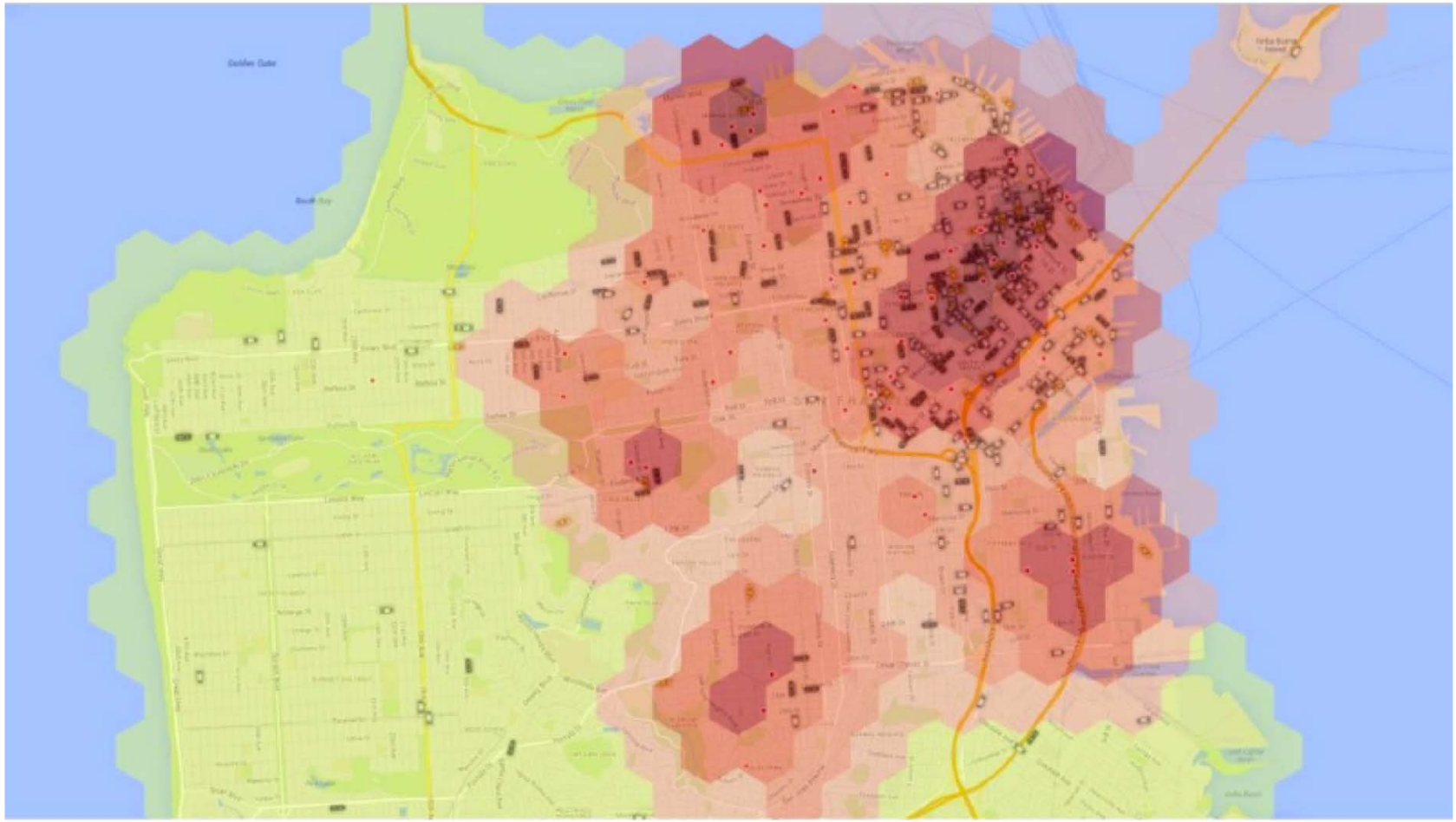


UBER

Regression Example: Uber

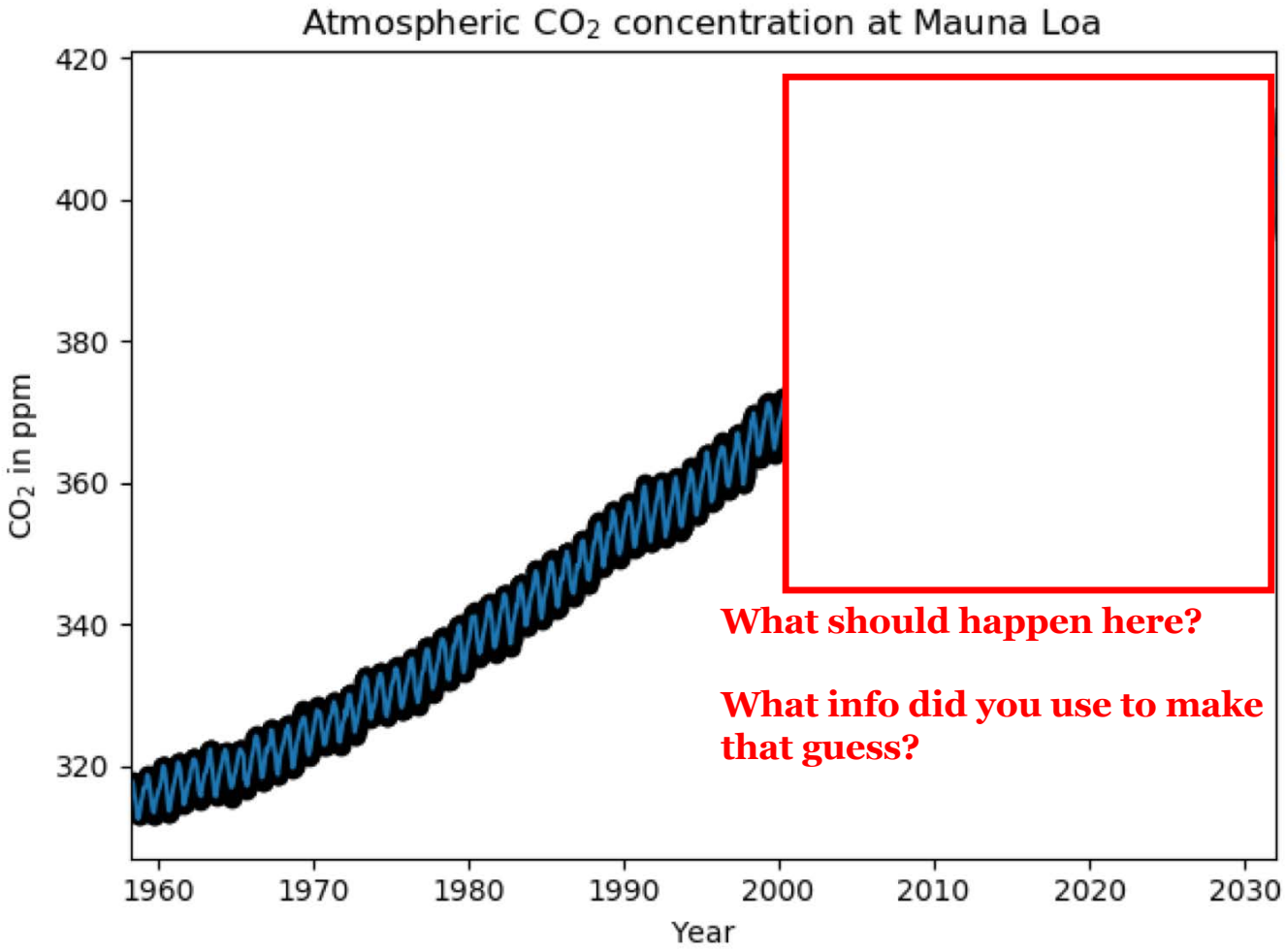
(Keith Chen)

Regression Example: Uber



(Keith Chen)

Try it!


Atmospheric CO_2 concentration at Mauna Loa
CO_2 in ppm
420
400
380
360
340
320
1960
1970
1980
1990
2000
2010
2020
2030
Year
What should happen here?
What info did you use to make that guess?

Regression: **Prediction Step**

Goal: Predict response y well given features x

- Input: $x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$

  "features"
  "covariates"
  "predictors"
  "attributes"

  Entries can be real-valued, or other numeric types (e.g. integer, binary)

- Output: $\hat{y}(x_i) \in \mathbb{R}$

  "responses"
  "labels"

  Scalar value like 3.1 or -133.7

Regression: **Prediction Step**

```
>>> # Given: pretrained regression object model
>>> # Given: 2D array of features x

>>> x_NF.shape
(N, F)

>>> yhat_N = model.predict(x_NF)


>>> yhat_N.shape
(N,)
```

Regression: **Training Step**

Goal: Given a labeled dataset, learn a **function** that can perform prediction well

- Input: Pairs of features and labels/responses

$$\{x_n, y_n\}_{n=1}^{N}$$

- Output: $\hat{y}(\cdot) : \mathbb{R}^F \rightarrow \mathbb{R}$

Regression: **Training Step**

```
>>> # Given: 2D array of features x
>>> # Given: 1D array of responses/labels y

>>> y_N.shape
(N,)
>>> x_NF.shape
(N, F)

>>> model = RegressionModel()
>>> model.fit(x_NF, y_N)
```

Regression: **Evaluation Step**

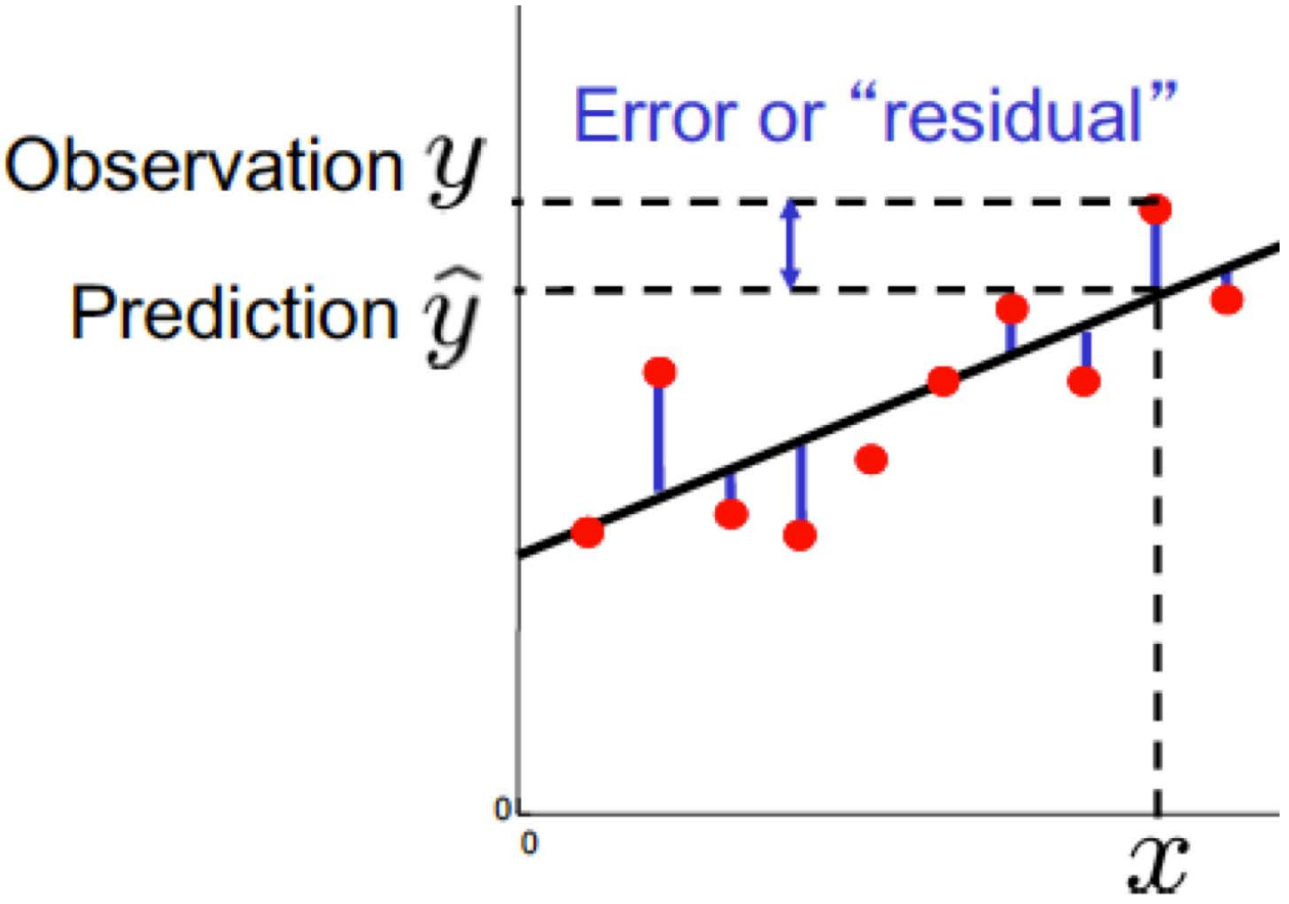
Goal: Assess quality of predictions

- Input: Pairs of predicted and "true" responses

$$\{\hat{y}(x_n), y_n\}_{n=1}^{N}$$

- Output: Scalar measure of error/quality
  - Measuring Error: **lower** is better
  - Measuring Quality: **higher** is better

Visualizing errors



Regression: Evaluation Metrics

- mean squared error $\frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$
- mean absolute error $\frac{1}{N} \sum_{n=1}^{N} |y_n - \hat{y}_n|$

Discuss

- Which error metric is more sensitive to outliers?

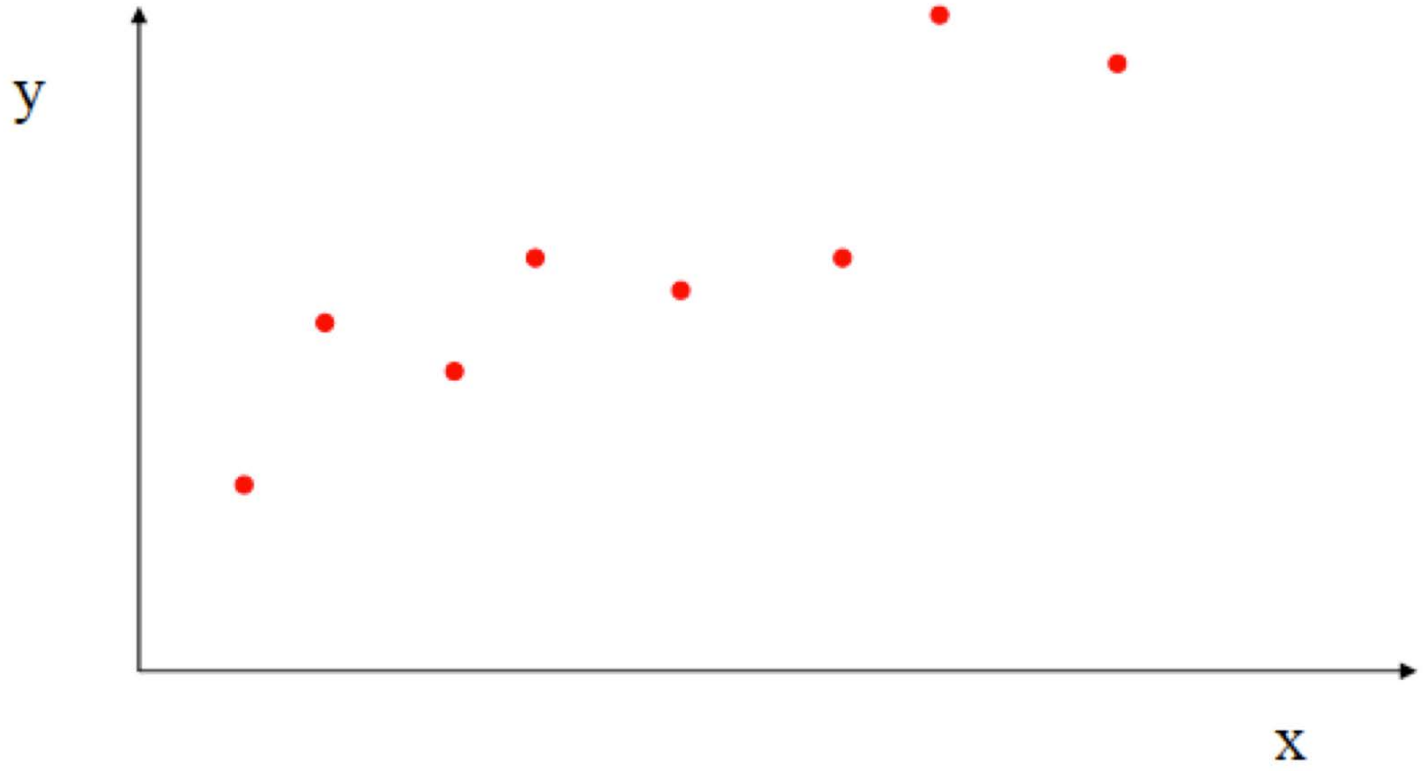
- Which error metric is the easiest to take derivatives of?

Regression: Evaluation Metrics

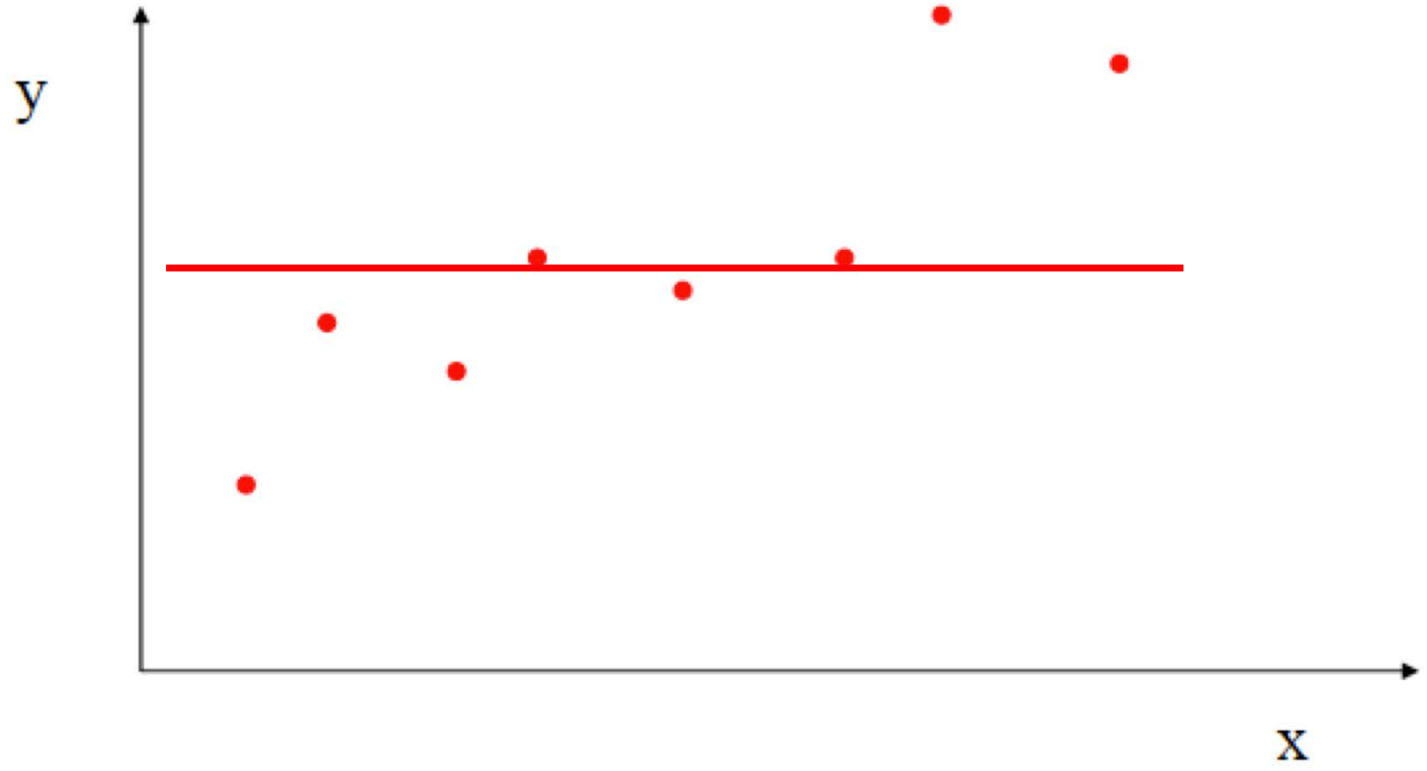
https://scikit-learn.org/stable/modules/model_evaluation.html

Regression	
'explained_variance'	`metrics.explained_variance_score`
'neg_mean_absolute_error'	`metrics.mean_absolute_error`
'neg_mean_squared_error'	`metrics.mean_squared_error`
'neg_mean_squared_log_error'	`metrics.mean_squared_log_error`
'neg_median_absolute_error'	`metrics.median_absolute_error`
'r2'	`metrics.r2_score`

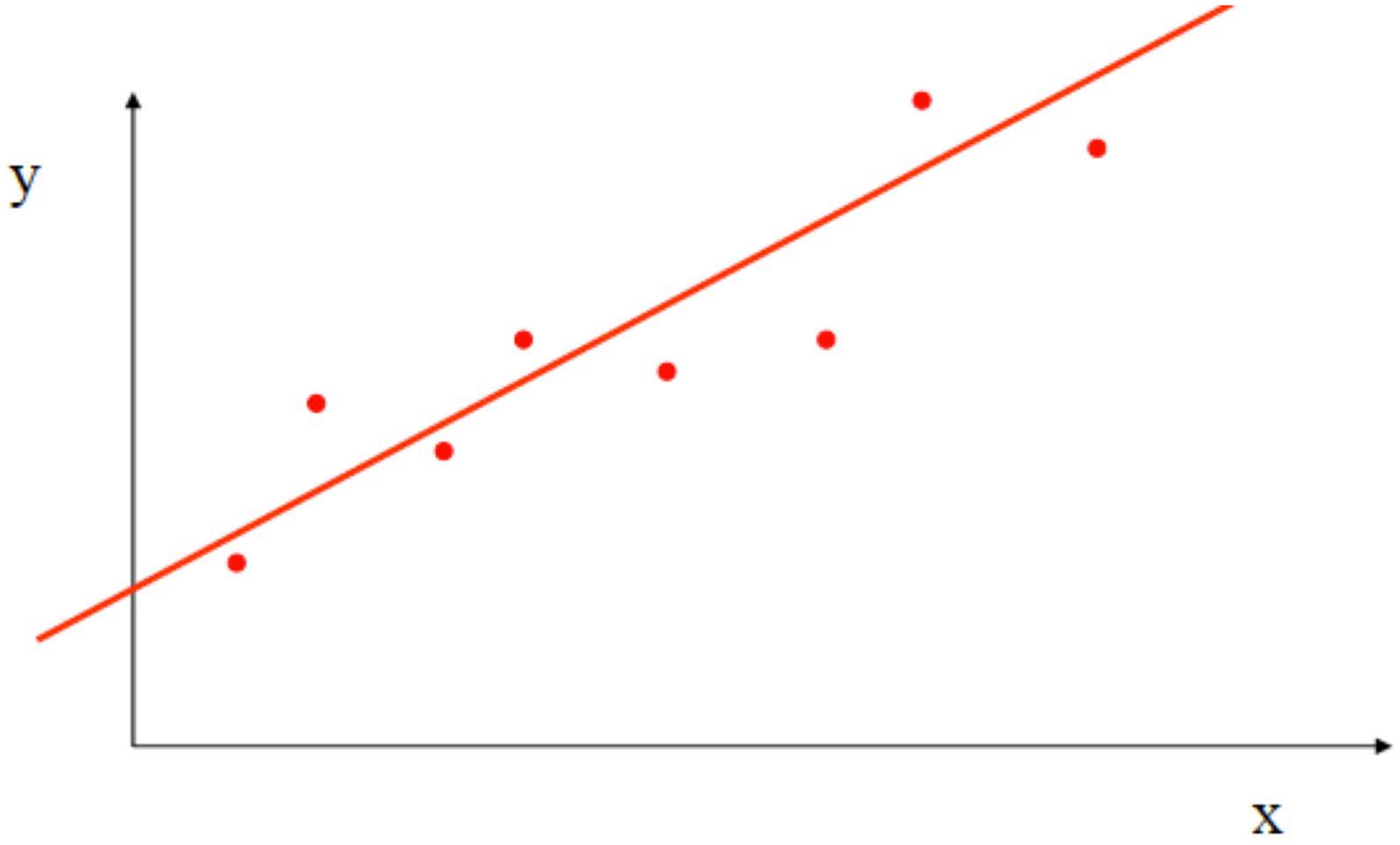
How to model y given x?



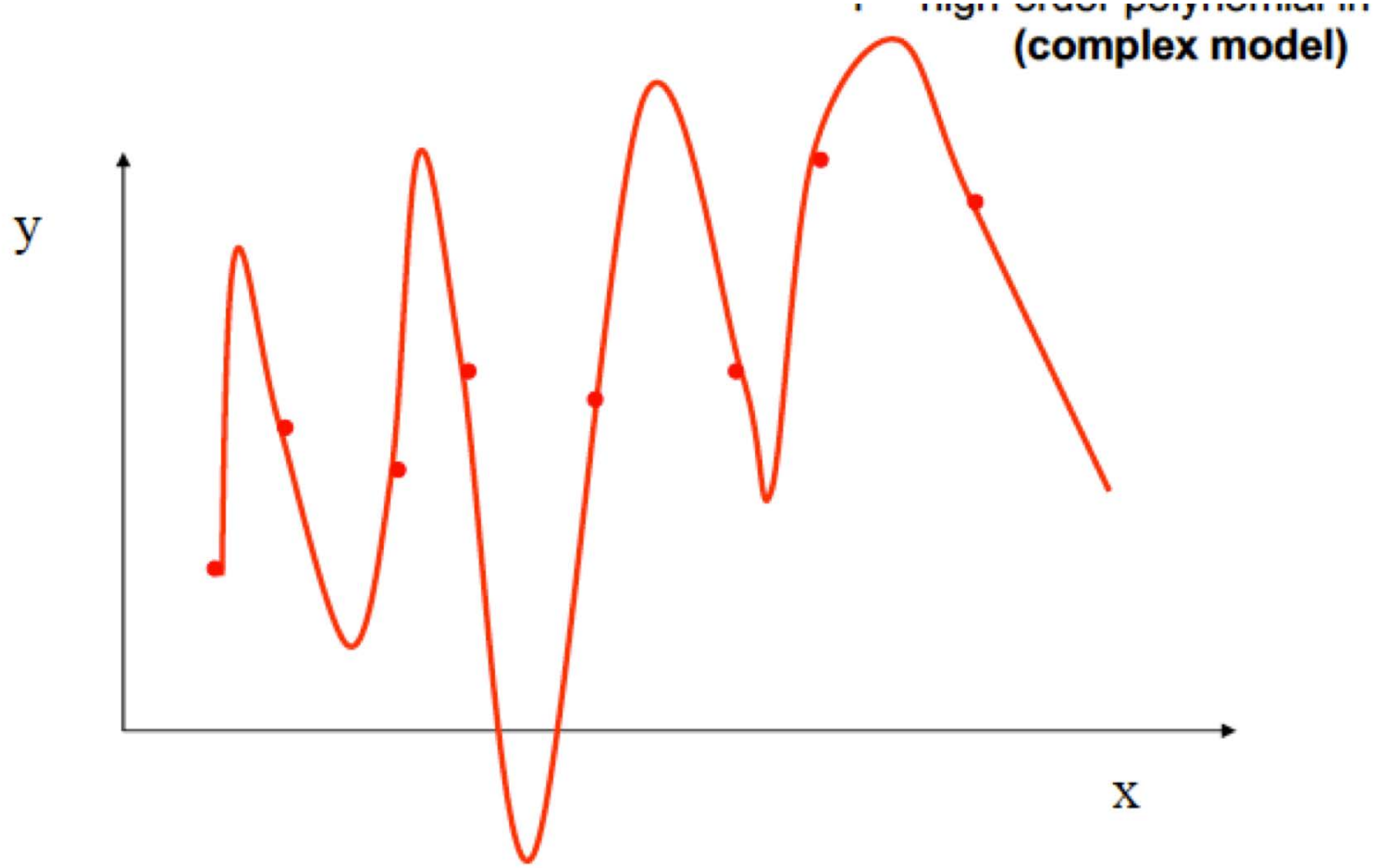
Is the model constant?



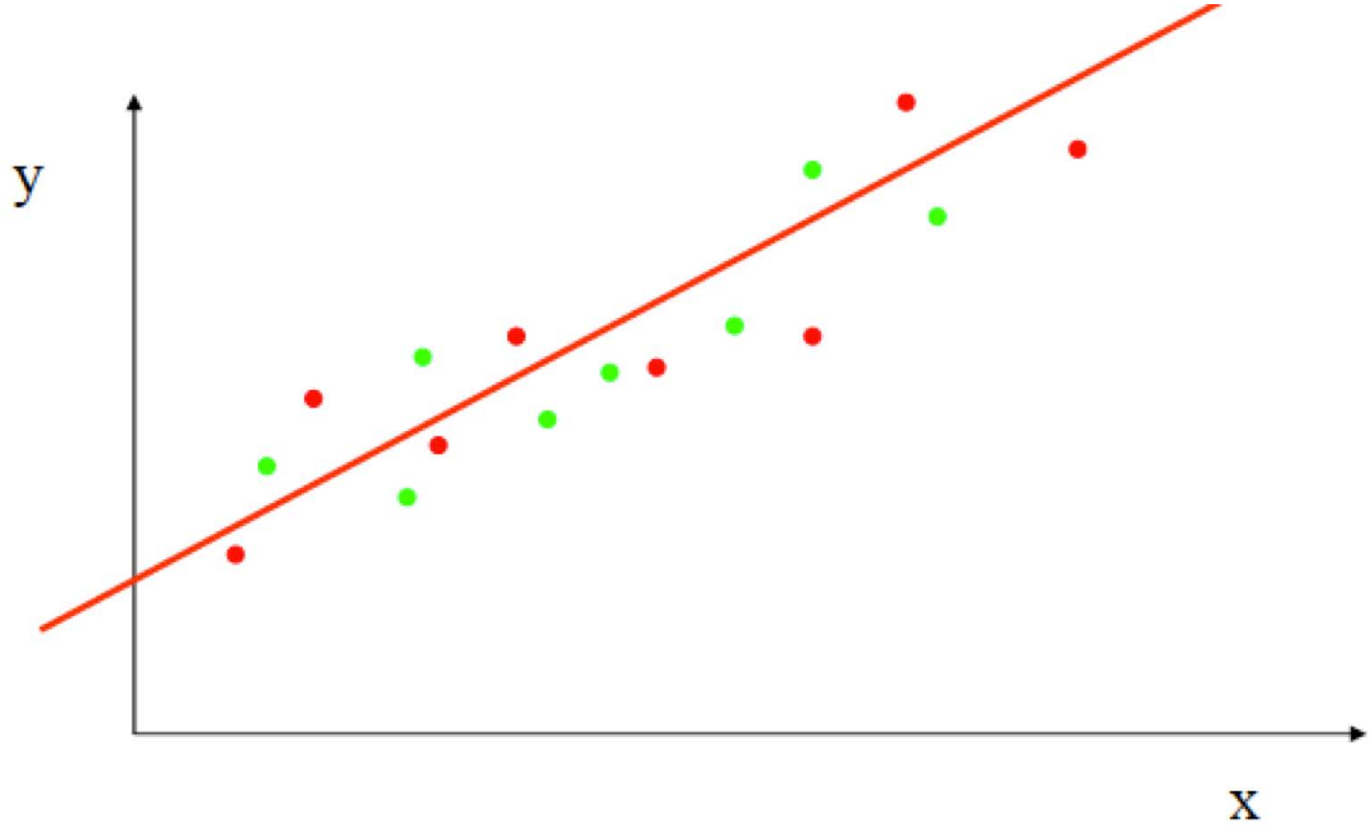
Is the model linear?



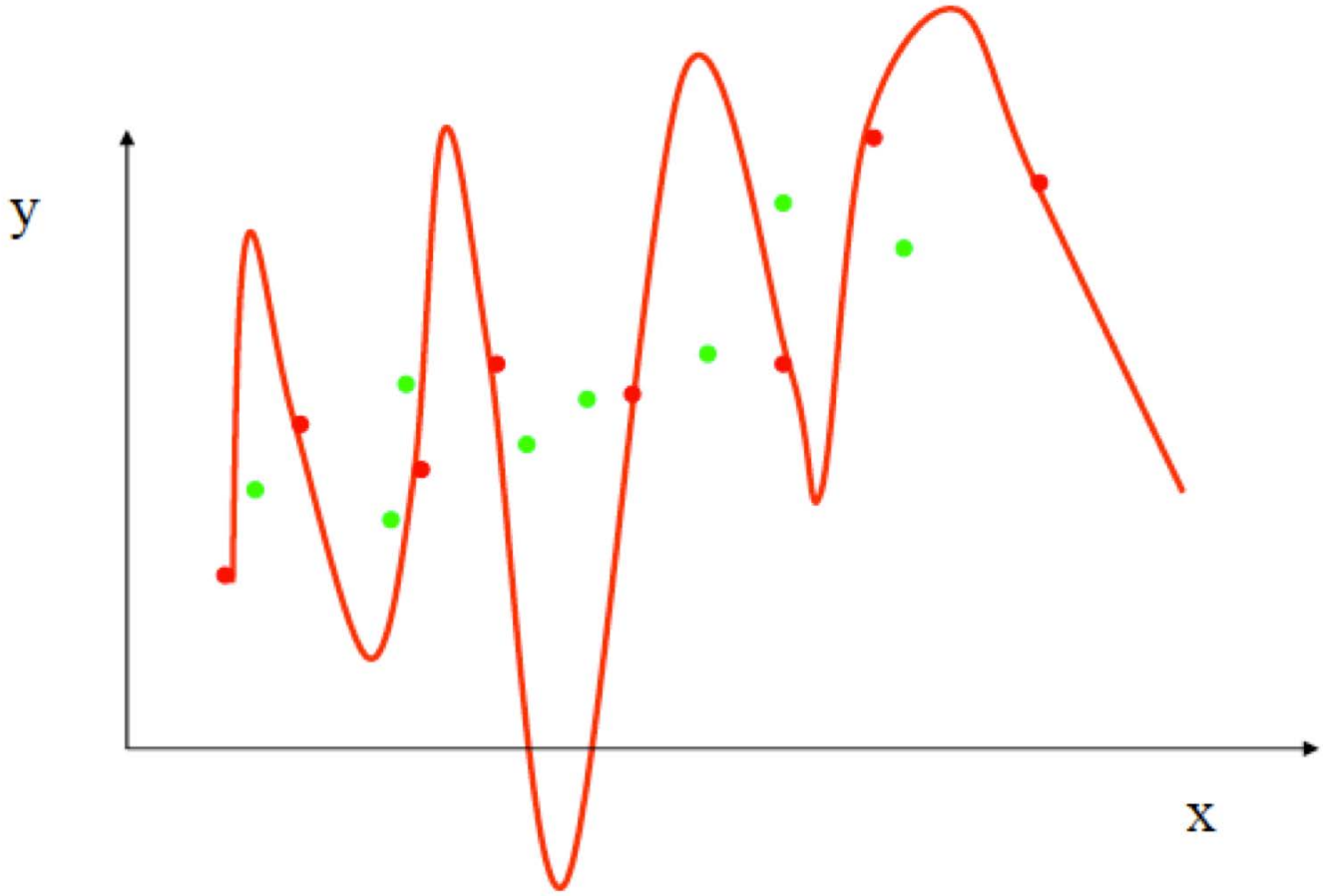
Is the model polynomial?



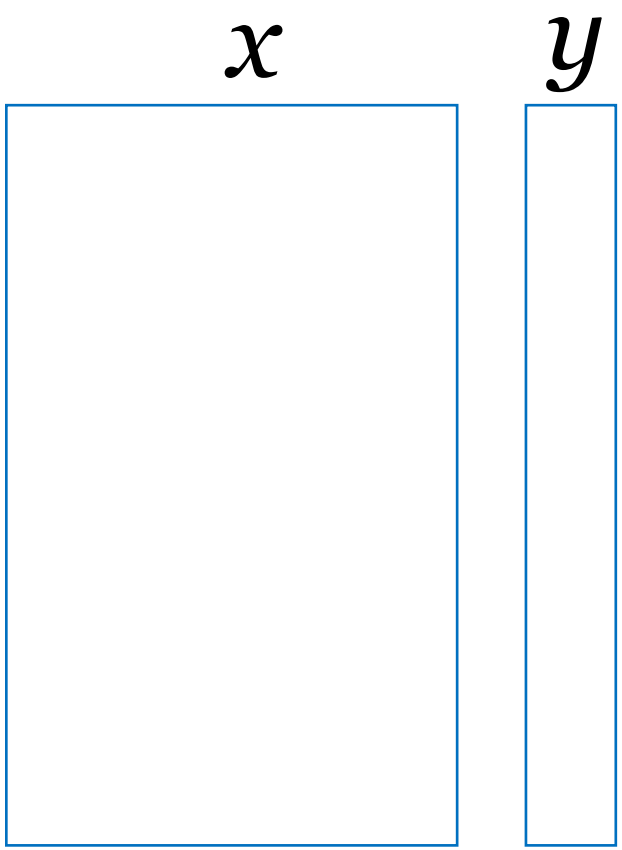
Generalize: sample to population



Generalize: sample to population



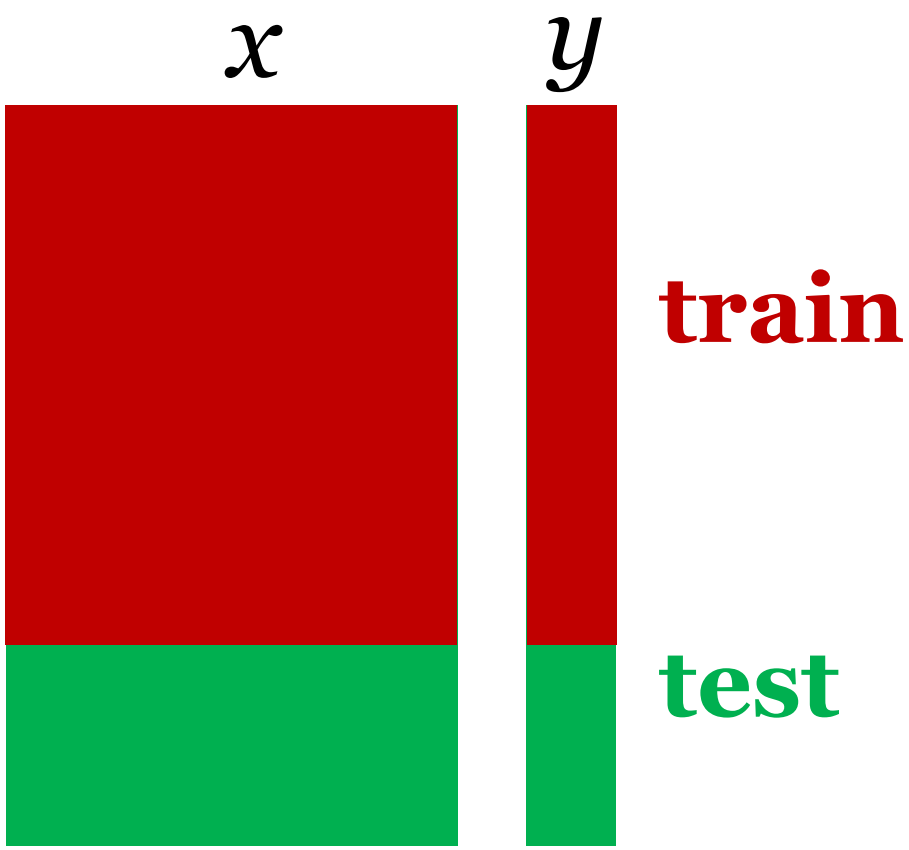
Labeled dataset



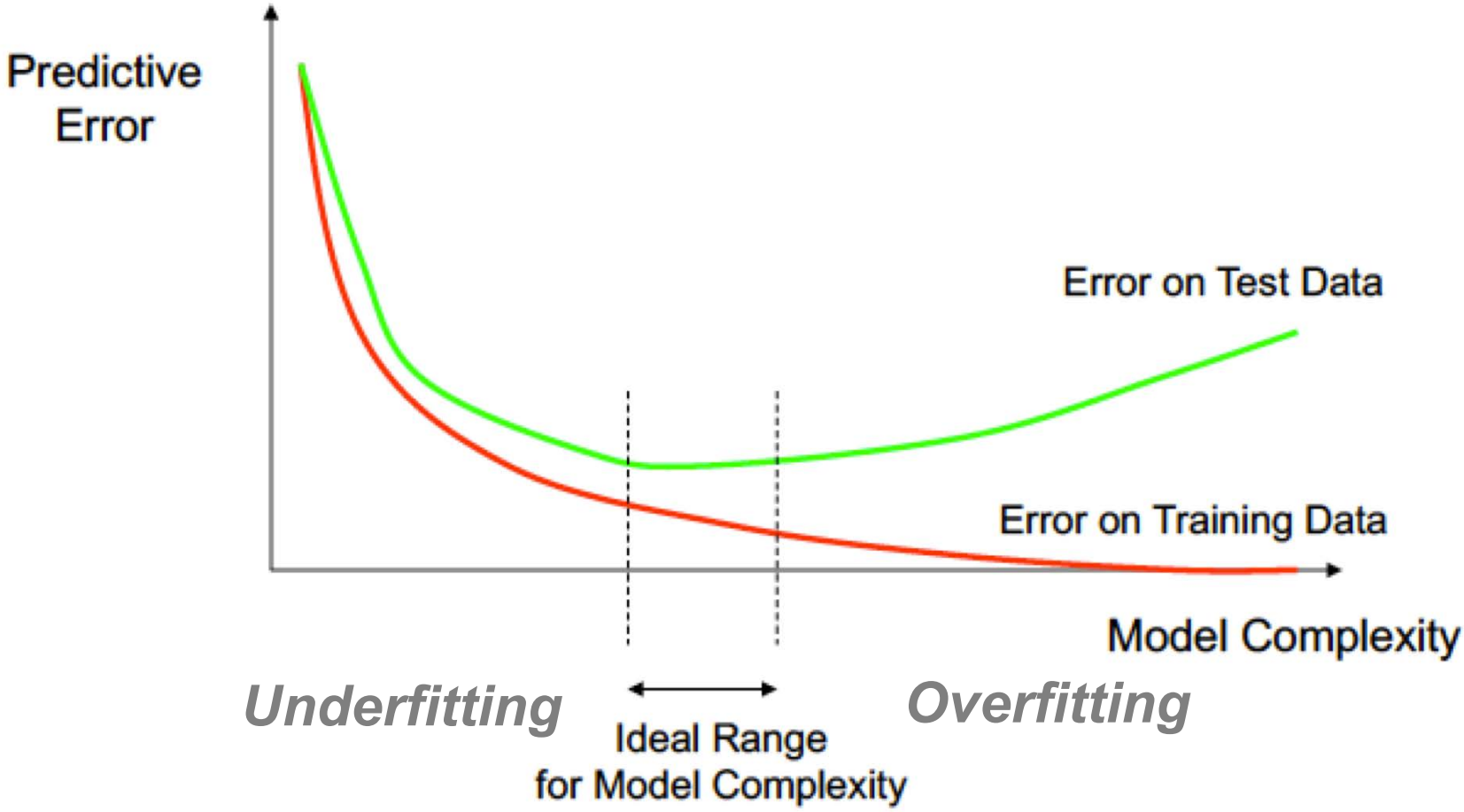
Each row represents one example

Assume rows are arranged "uniformly at random" (order doesn't matter)

Split into train and test



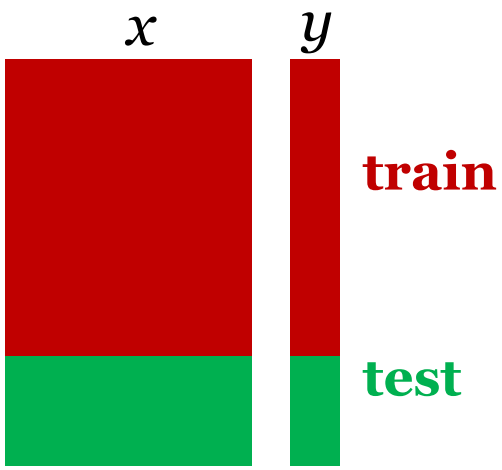
Model Complexity vs Error



How to fit best model?

Option 1: Fit on train, select on test

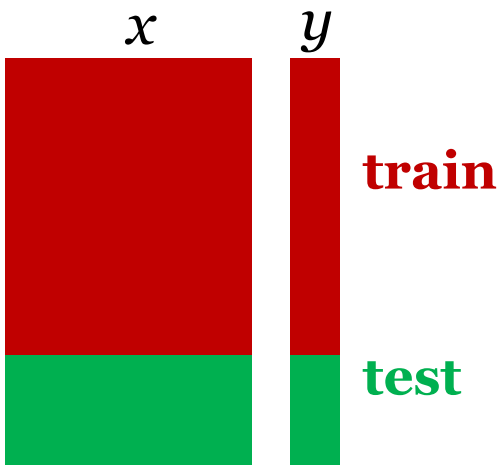
1) Fit each model to training data
2) Evaluate each model on test data
3) Select model with lowest test error



How to fit best model?

~~Option 1: Fit on train, select on test~~ **Avoid!**

1) Fit each model to training data
2) Evaluate each model on test data
3) Select model with lowest test error



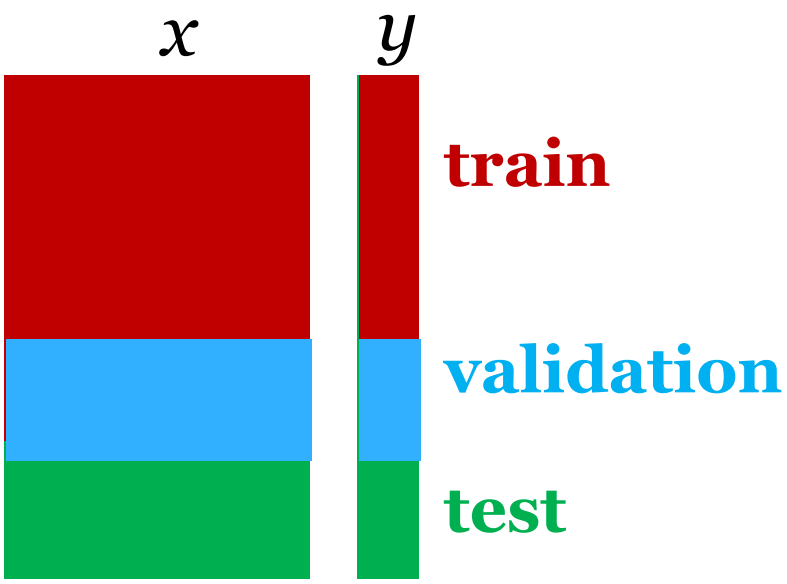
Problems

- Fitting procedure used test data
- Not fair assessment of how will do on unseen data

How to fit best model?

Option: Fit on train, select on **validation**

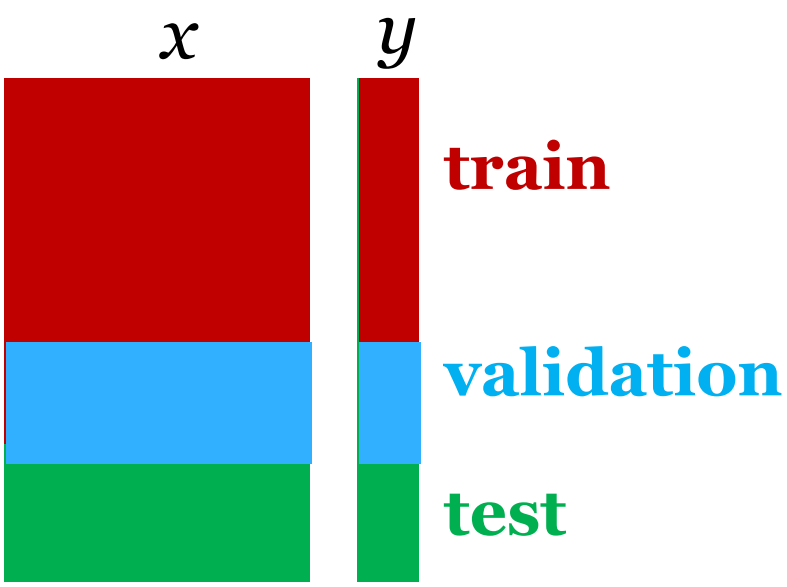
1) Fit each model to **training** data
2) Evaluate each model on **validation** data
3) Select model with lowest **validation** error
4) Report error on **test** set



How to fit best model?

Option: Fit on train, select on **validation**

1) Fit each model to **training** data
2) Evaluate each model on **validation** data
3) Select model with lowest **validation** error
4) Report error on **test** set



Concerns

- Will train be too small?
- Make better use of data?

Linear Regression

Parameters:

weight vector $w = [w_1, w_2, \ldots w_f \ldots w_F]$

bias scalar b

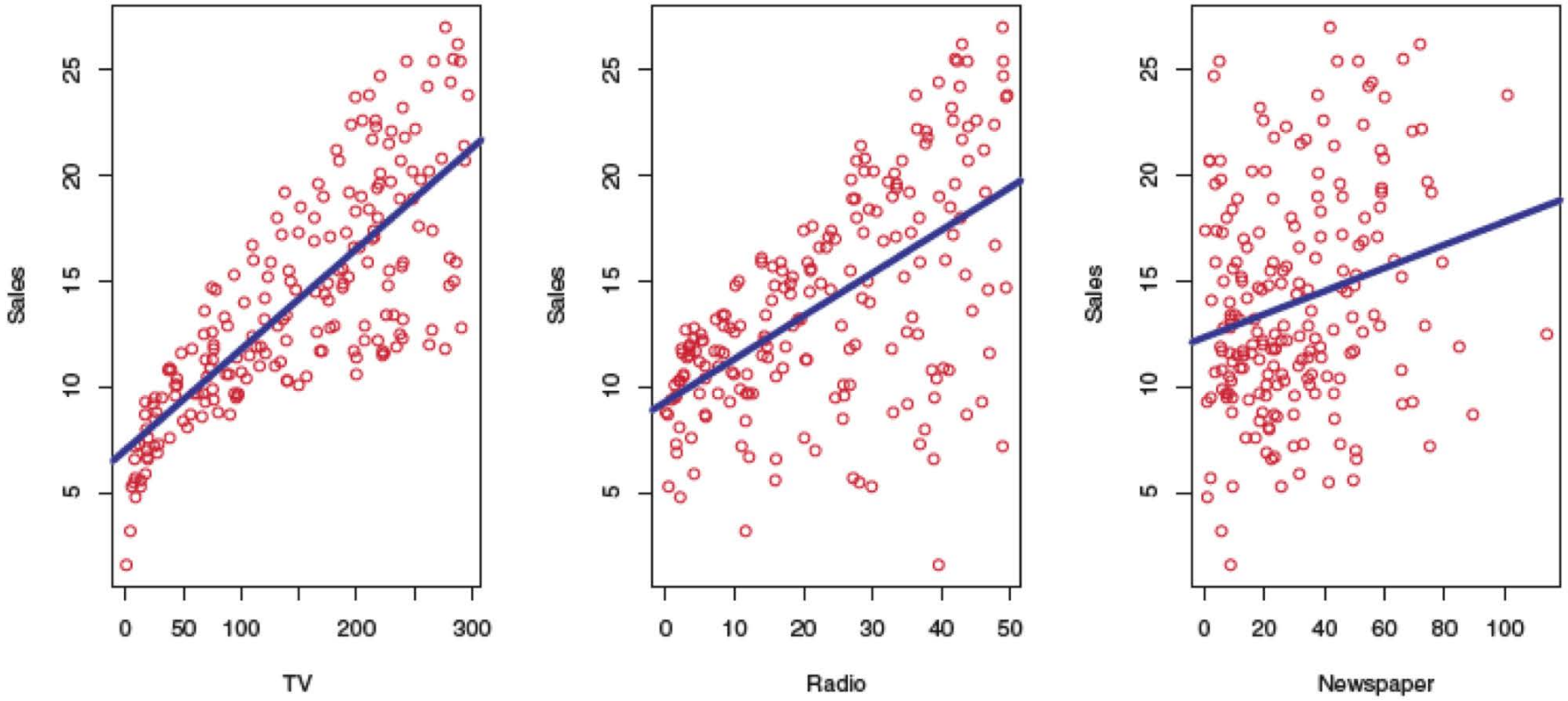
Prediction:

$$\hat{y}(x_i) \triangleq \sum_{f=1}^{F} w_f x_{if} + b$$

Training:

find weights and bias that minimize error

Sales vs. Ad Budgets



Linear Regression: Training

Optimization problem: “Least Squares”

$$\min_{w,b} \sum_{n=1}^{N} \Big(y_n - \hat{y}(x_n, w, b) \Big)^2$$

Linear Regression: Training

Optimization problem: "Least Squares"

$$\min_{w,b} \sum_{n=1}^{N} \Big(y_n - \hat{y}(x_n, w, b) \Big)^2$$

Exact formula for optimal values of w, b exist!

With only one feature (F=1):

$$w = \frac{\sum_{n=1}^{N} (x_n - \bar{x})(y_n - \bar{y})}{\sum_{n=1}^{N} (x_n - \bar{x})^2}$$

$$\bar{x} = \text{mean}(x_1, \ldots x_N)$$

$$\bar{y} = \text{mean}(y_1, \ldots y_N)$$

$$b = \bar{y} - w\bar{x}$$

We will derive these in next class

Linear Regression: Training

Optimization problem: "Least Squares"

$$\min_{w,b} \sum_{n=1}^{N} \Big(y_n - \hat{y}(x_n, w, b) \Big)^2$$

Exact formula for optimal values of w, b exist!

$$\tilde{X} = \begin{bmatrix} x_{11} & \dots & x_{1F} & 1 \\ x_{21} & \dots & x_{2F} & 1 \\ & & \dots & \\ x_{N1} & \dots & x_{NF} & 1 \end{bmatrix}$$

With many features (F >= **1**):

$$[w_1 \dots w_F \; b]^T = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

We will derive these in next class

Nearest Neighbor Regression

Parameters:

none

Prediction:

- find "nearest" training vector to given input x
- predict y value of this neighbor

Training:

none needed (use training data as lookup table)

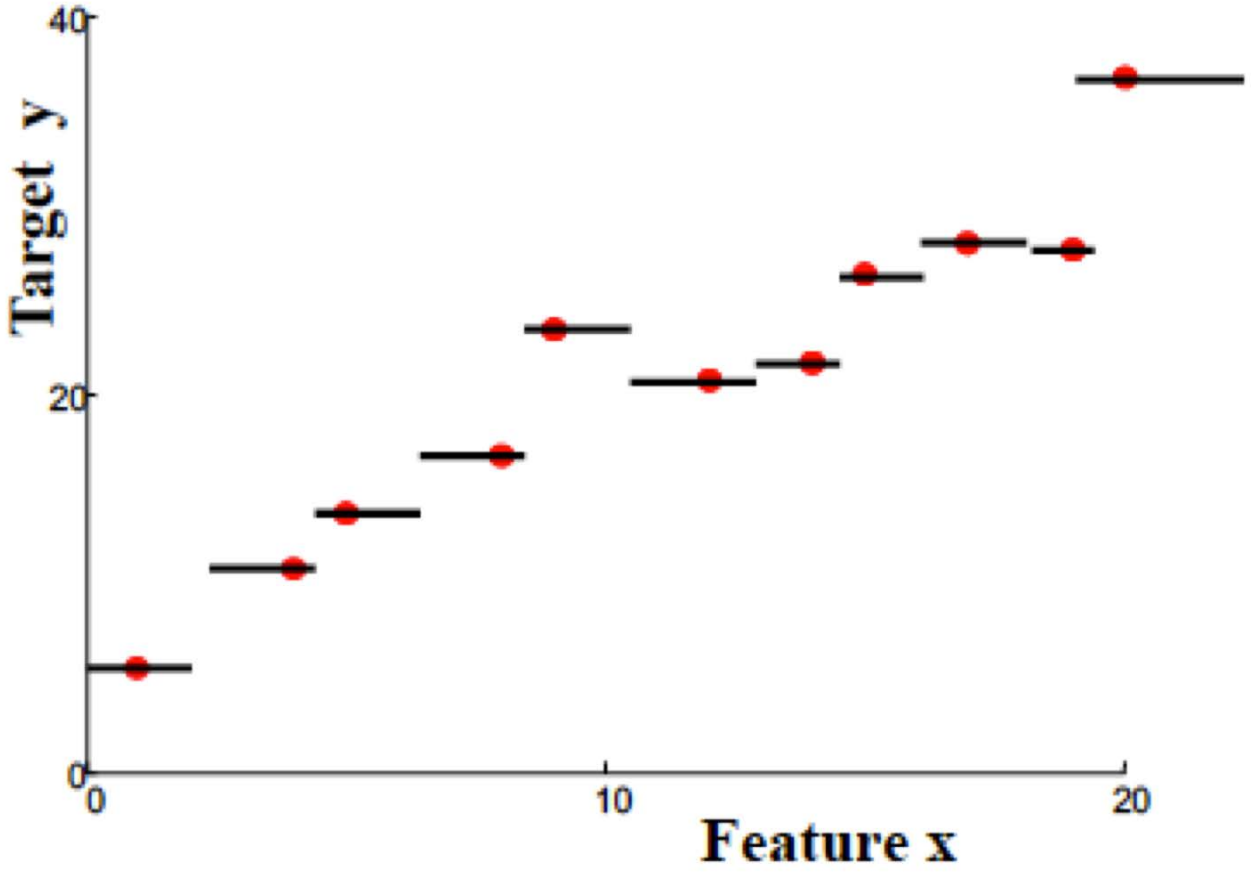
Distance metrics

- Euclidean $\text{dist}(x, x') = \sqrt{\sum_{f=1}^{F} (x_f - x'_f)^2}$

- Manhattan $\text{dist}(x, x') = \sum_{f=1}^{F} |x_f - x'_f|$

- Many others are possible

Nearest Neighbor "Prediction functions" are piecewise constant



K nearest neighbor regression

Parameters:

K : number of neighbors

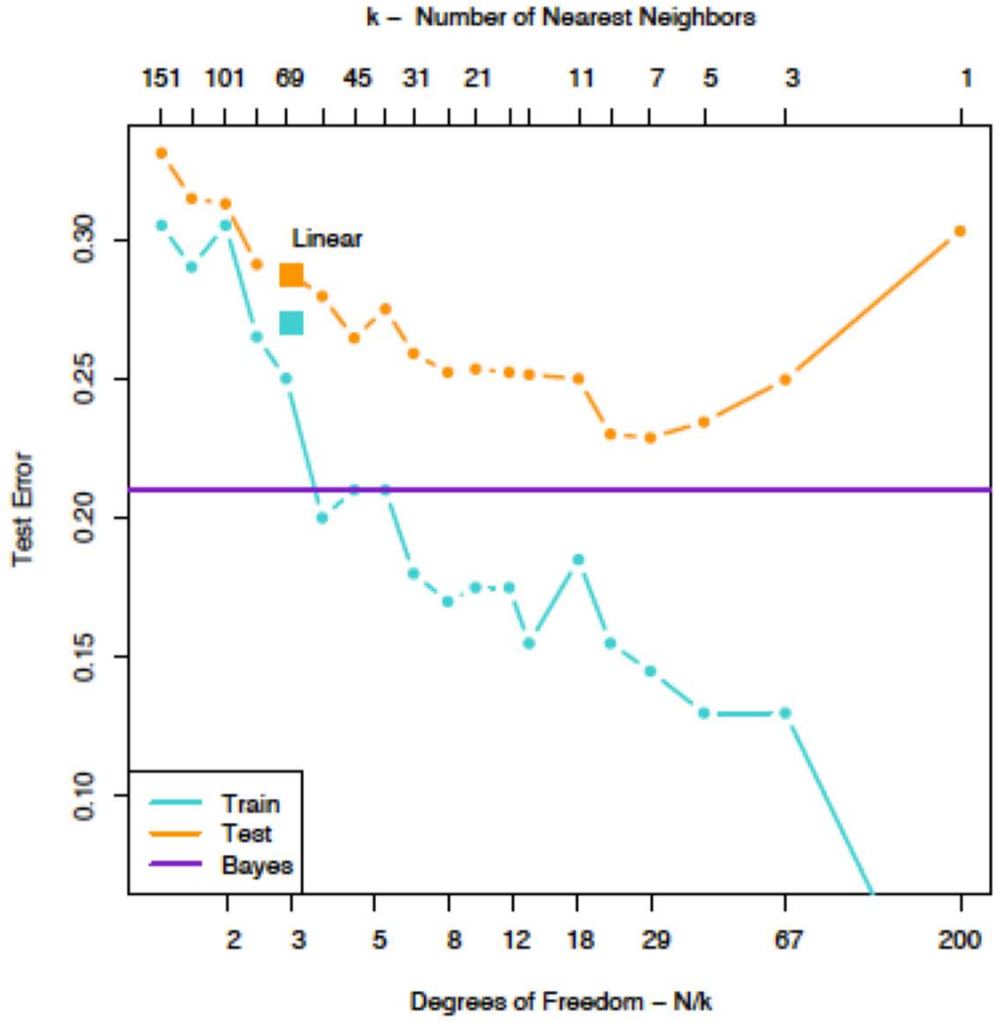
Prediction:

- find K "nearest" training vectors to input x
- predict **average** y of this neighborhood

Training:

none needed (use training data as lookup table)

Error vs Model Complexity



Credit:
Fig 2.4
ESL textbook

Salary prediction for Hitters data

A data frame with 322 observations of major league players on the following 20 variables.

`AtBat`

Number of times at bat in 1986

`Hits`

Number of hits in 1986

`HmRun`

Number of home runs in 1986

`Runs`

Number of runs in 1986

`RBI`

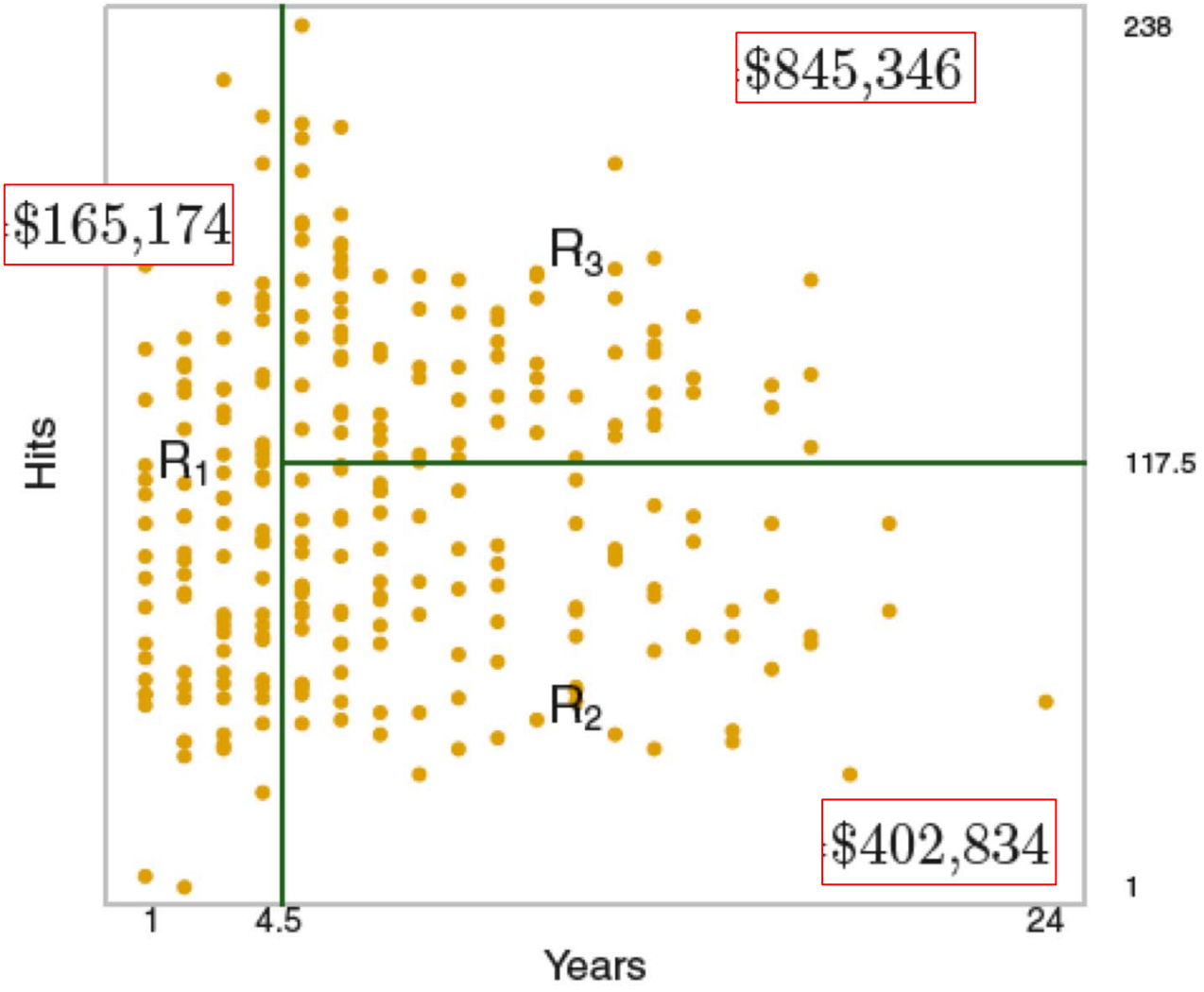
Number of runs batted in in 1986

`Walks`

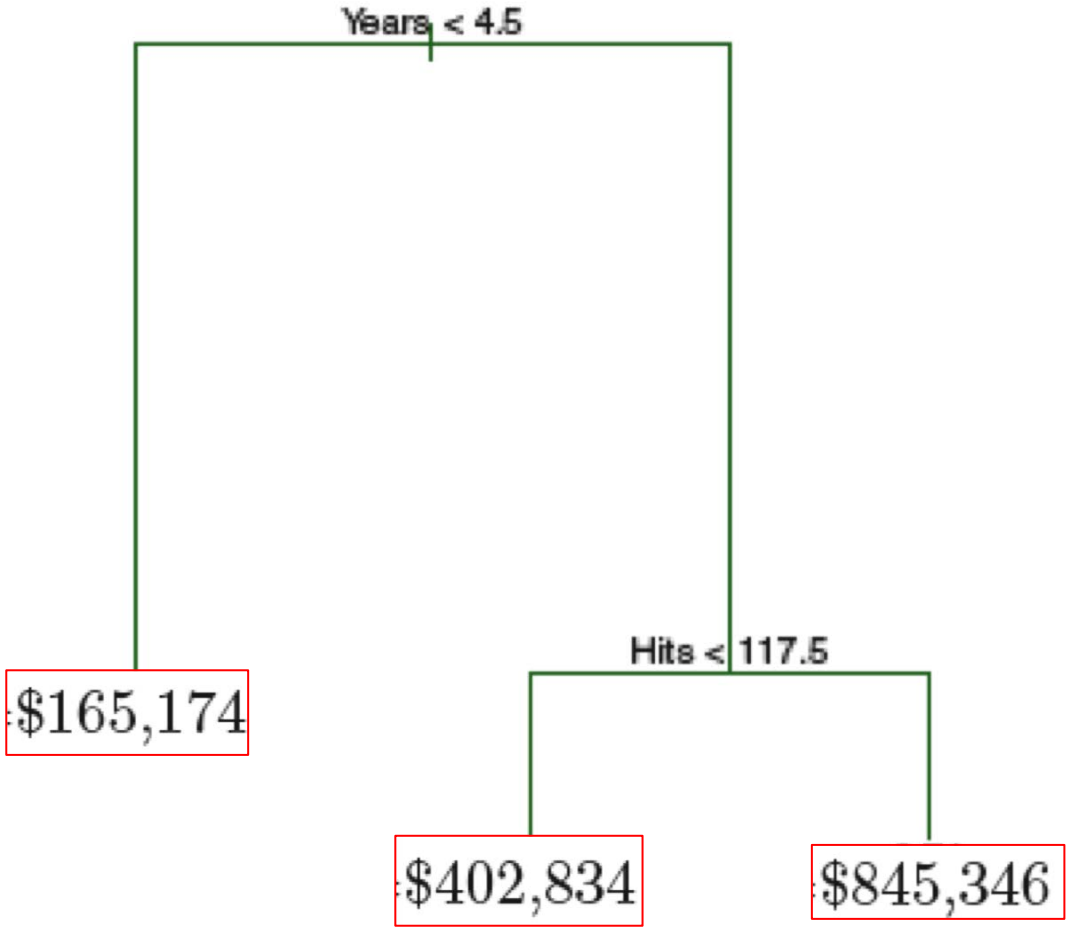
Number of walks in 1986

`Years`

Number of years in the major leagues


$845,346
$165,174
$402,834
R1
R2
R3
Hits
Years
238
117.5
1
1
4.5
24

Decision Tree Regression



Decision tree regression

Parameters:

- *at each internal node: x* variable id and threshold
- *at each leaf:* scalar *y* value to predict

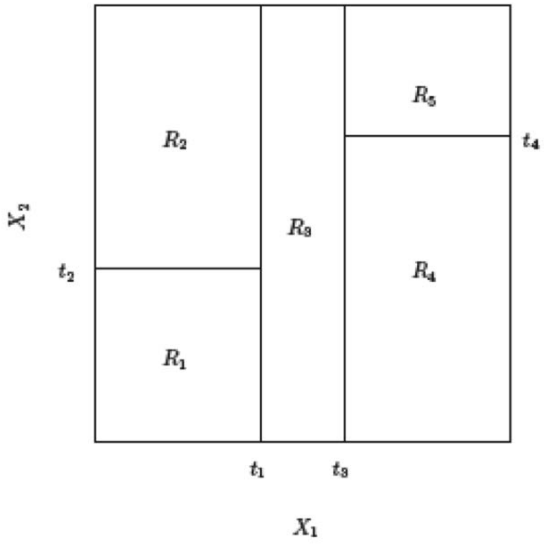
Prediction assumption:

- x space is divided into rectangular regions
- y is similar within “region”

Training assumption:

- minimize error on training set
- often, use greedy heuristics

Ideal Training for Decision Tree



$$\min_{R_1, \ldots R_J} \sum_{j=1} \sum_{n: x_n \in R_j} (y_n - \hat{y}_{R_j})^2$$

Search space is too big! Hard to solve exactly...

Greedy Top-Down Training

Given a big region, find best binary split into two subregions

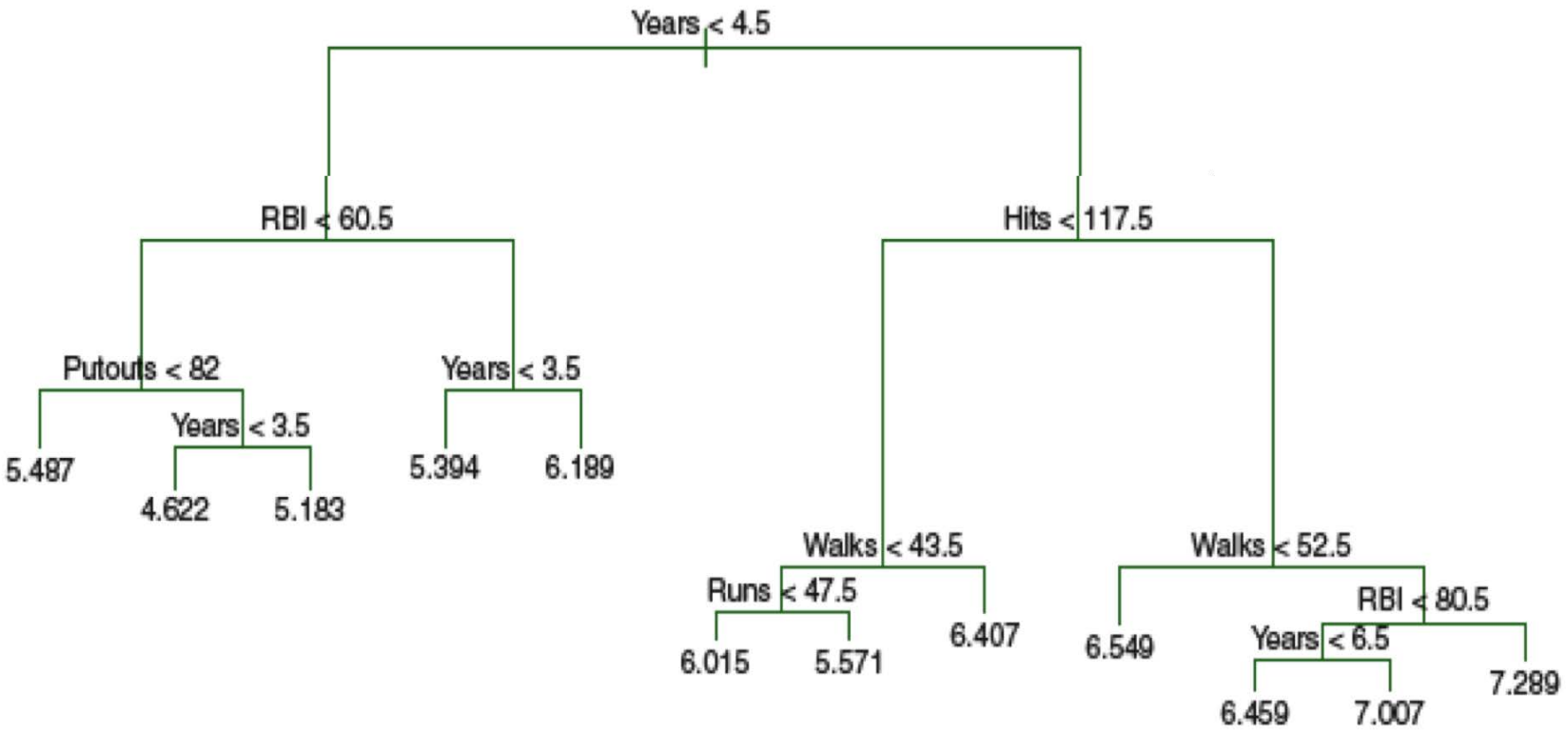
$$R_1(j, s) = \{X | X_j < s\} \text{ and } R_2(j, s) = \{X | X_j \geq s\}$$

$$\min_{j, s, \hat{y}_{R_1}, \hat{y}_{R_2}} \sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Stop when:

- number of examples assigned to a leaf is too small
- Maximum depth is exceeded

Greedy Tree for Hitters Data



Summary of Methods

	Function class flexibility	**Knobs to tune**	**Interpret?**
Linear Regression	Linear	Include bias? *Penalize weights* (*more next week*)	Inspect weights
Decision Tree Regression	Axis-aligned Piecewise constant	Max. depth Min. leaf size Goal criteria	Inspect tree
K Nearest Neighbors Regression	Piecewise constant	Number of Neighbors Distance metric How neighbors vote	Inspect neighbors

Discuss:

We are studying data with ~5 features

Which method's predictions change most across versions of feature representation?

- Version A: $[x_1 \ x_2 \ x_3 \ x_4 \ x_5]$
- Version B: $[10x_1 \ x_2 \ x_3 \ x_4 \ x_5]$

Discuss:

We are studying data with ~3 features

Which method's predictions change most across versions of feature representation?

- Version A: $[x_1 \ x_2]$
- Version B: $[x_1 \ x_2 \ \text{noise} \ \text{noise} \ \text{noise}]$

Regression Unit Objectives

- 3 steps of a regression task
  - Training
  - Prediction
  - Evaluation
    - Metrics
    - Splitting into train/valid/test
- Chosen performance metric should be integrated at training
- Mean squared error is "easy", but not always the right thing to do

- "Taste" of 3 Methods
  - Linear Regression
  - K-Nearest Neighbors
  - Decision Tree Regression