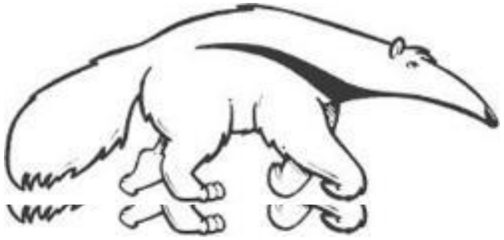
Machine Learning and Data Mining

Introduction

Kalev Kask

273P Spring 2018



BREN:ICS INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

Artificial Intelligence (AI)

- Building “intelligent systems”
- Lots of parts to intelligent behavior

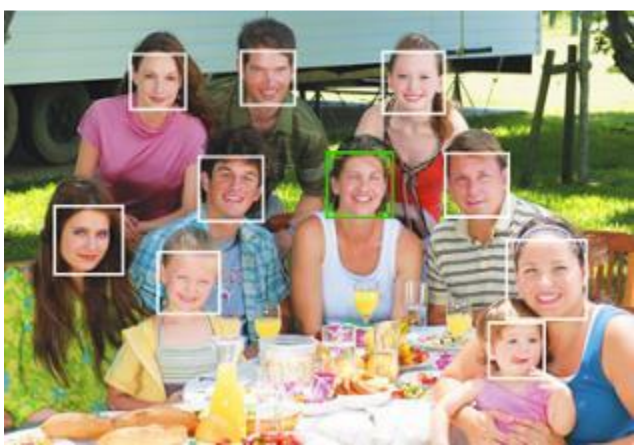
Darpa GC (Stanley)

RoboCup

Chess (Deep Blue v. Kasparov)

Machine learning (ML)

- One (important) part of AI
- Making predictions (or decisions)
- Getting better with experience (data)
- Problems whose solutions are “hard to describe”



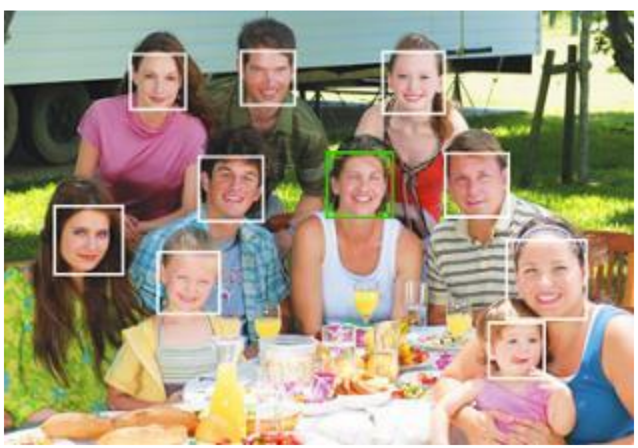
Areas of ML

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Types of prediction problems

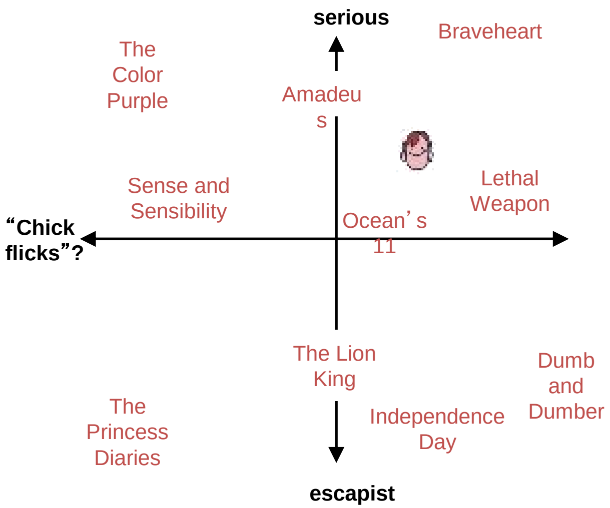
- Supervised learning
  - "Labeled" training data
  - Every example has a desired target value (a "best answer")
  - Reward prediction being close to target

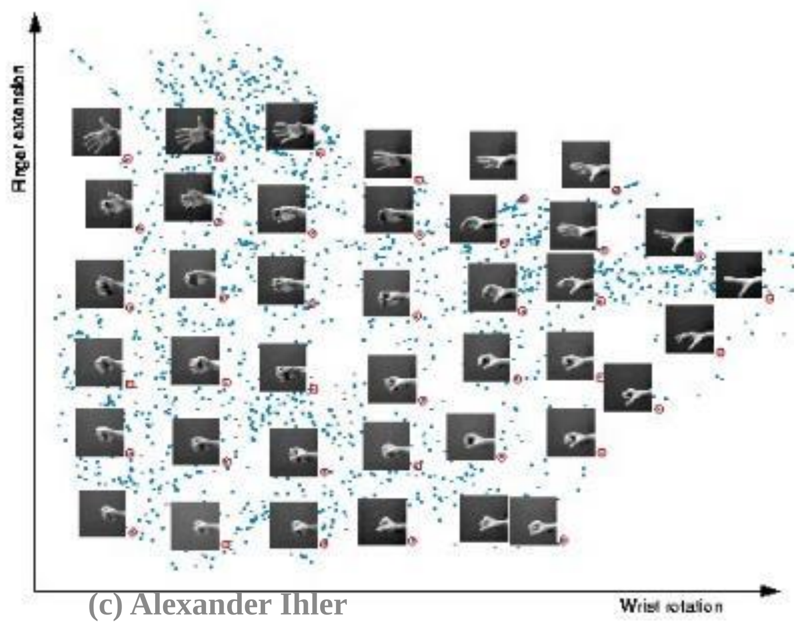
  - Classification: a discrete-valued prediction (often: decision)
  - Regression: a continuous-valued prediction



Types of prediction problems

- Supervised learning
- Unsupervised learning
  – No known target values
  – No targets = nothing to predict?
  – Reward "patterns" or "explaining features"
  – Often, data mining





Types of prediction problems

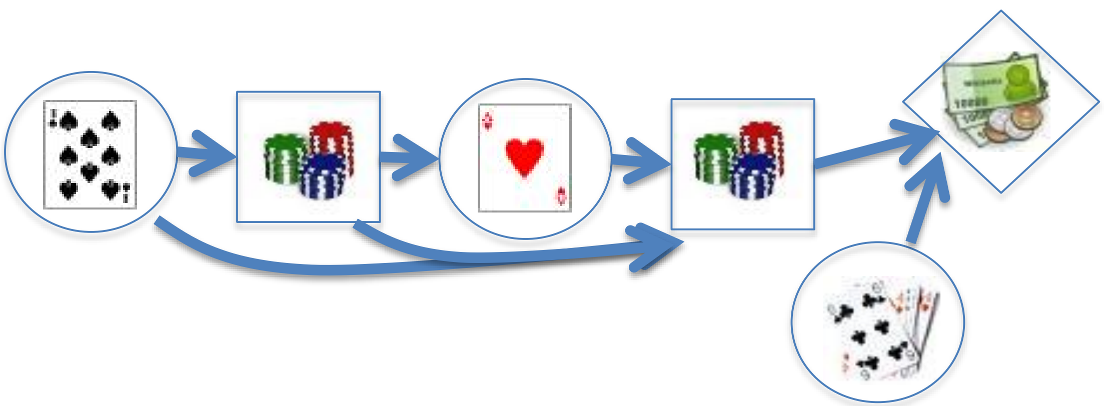
- Supervised learning
- Unsupervised learning
- Semi-supervised learning
  - Similar to supervised
  - some data have unknown target values

- Ex: medical data
  - Lots of patient data, few known outcomes
- Ex: image tagging
  - Lots of images on Flickr, but only some of them tagged

Types of prediction problems

- Supervised learning
- Unsupervised learning
- Semi-supervised learning

- "Indirect" feedback on quality
  - No answers, just "better" or "worse"
  - Feedback may be delayed



Logistics

- 11 weeks
  - 10 weeks of instruction (04/03 – 06/07)
  - Finals week (06/14 4-6pm)
  - Lab Tu 7:00-7:50 SSL 270
- Course webpage for assignments & other info
- gradescope.com for homework submission & return
- Piazza for questions & discussions
  - piazza.com/uci/spring2018/cs273p

Textbook

- No required textbook
  - I’ll try to cover everything needed in lectures and notes
- Recommended reading for reference
  - Duda, Hart, Stork, "Pattern Classification“
  - Daume "A Course in Machine Learning“
  - Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning“
  - Murphy "Machine Learning: A Probabilistic Perspective“
  - Bishop "Pattern Recognition and Machine Learning“
  - Sutton "Reinforcement Learning"

Logistics

- Grading (may be subject to change)
  - 20% homework (5+?   >5: drop 1)
  - 2 projects 20% each
  - 40% final
  - Due 11:59pm listed day, myEEE
  - Late homework:
    - 10% off per day
    - No credit after solutions posted: turn in what you have
- Collaboration
  - Study groups, discussion, assistance encouraged
    - Whiteboards, etc.
  - Any submitted work must be your own
    - Do your homework yourself
    - Don’t exchange solutions or HW code

Projects

- 2 projects:
  - Regression (written report due about week 8/9)
  - Classification (written report due week 11)
- Teams of 3 students
- Will use Kaggle
- Bonus points for winners, but
  - Project evaluated based on report

Scientific software

- **Python**
  - **Numpy, MatPlotLib, SciPy, SciKit …**
- Matlab
  - Octave (free)
- R
  - Used mainly in statistics
- C++
  - For performance, not prototyping
- And other, more specialized languages for modeling…

Lab/Discussion Section

- Tuesday, 7:00-7:50 pm SSL 270
  - Discuss material
  - Get help with Python
  - Discuss projects

Implement own ML program?

- Do I write my own program?
  - Good for understanding how algorithm works
  - Practical difficulties
    - Poor data?
    - Code buggy?
    - Algorithm not suitable?
- Adopt 3rd party library?
  - Good for understanding how ML works
  - Debugged, tested.
  - Fast turnaround.
- Mission-critical deployed system
  - Probably need to have own implementation
  - Good performance; C++; customized to circumstances!
- AI as service

Data exploration

- Machine learning is a data science
  - Look at the data; get a “feel” for what might work

- What types of data do we have?
  - Binary values? (spam; gender; …)
  - Categories? (home state; labels; …)
  - Integer values? (1..5 stars; age brackets; …)
  - (nearly) real values? (pixel intensity; prices; …)

- Are there missing data?

- “Shape” of the data? Outliers?

Representing data

- Example: Fisher's "Iris" data
  http://en.wikipedia.org/wiki/Iris_flower_data_set

- Three different types of iris
  - "Class", y

- Four "features", $x_1,\dots,x_4$
  - Length & width of
    sepals & petals

- 150 examples (data points)

Representing the data

- Have m observations (data points)

$$\left\{ x^{(1)} \ldots , x^{(m)} \right\}$$

- Each observation is a vector consisting of n features

$$x^{(j)} = [x_1^{(j)} x_2^{(j)} \ldots x_n^{(j)}]$$

- Often, represent this as a "data matrix"

$$\underline{X} = \begin{bmatrix} x_1^{(1)} & \ldots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \ldots & x_n^{(m)} \end{bmatrix}$$

```
import numpy as np    # import numpy
iris = np.genfromtxt("data/iris.txt",delimiter=None)
X = iris[:,0:4]          # load data and split into features, targets
Y = iris[:,4]
print X.shape            # 150 data points; 4 features each
   (150, 4)
```

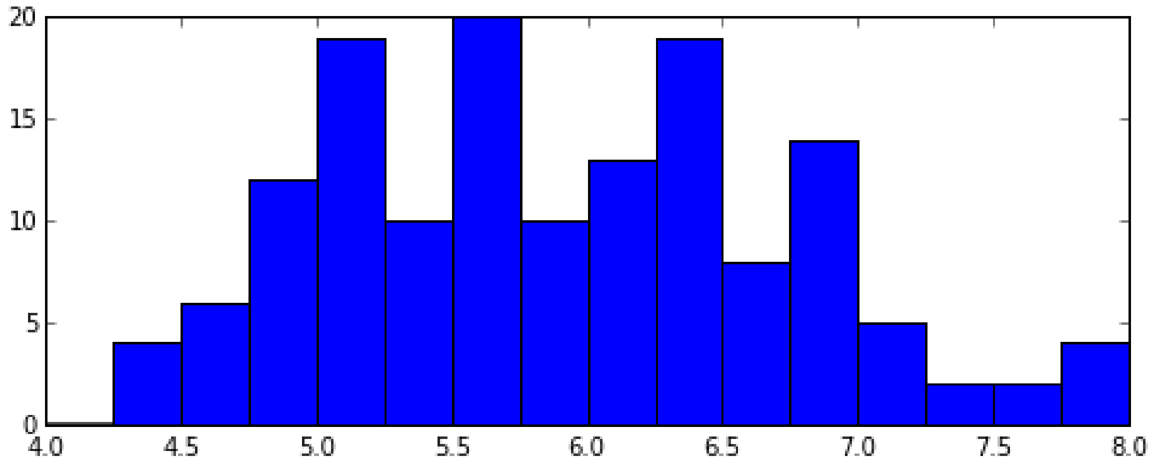
Basic statistics

- Look at basic information about features
  - Average value? (mean, median, etc.)
  - “Spread”? (standard deviation, etc.)
  - Maximum / Minimum values?

```
print np.mean(X, axis=0)          # compute mean of each feature
 [ 5.8433   3.0573   3.7580   1.1993 ]
print np.std(X, axis=0)           #compute standard deviation of each feature
 [ 0.8281   0.4359   1.7653   0.7622 ]
print np.max(X, axis=0)           # largest value per feature
 [  7.9411   4.3632   6.8606   2.5236 ]
print np.min(X, axis=0)           # smallest value per feature
 [  4.2985   1.9708   1.0331   0.0536  ]
```

Histograms

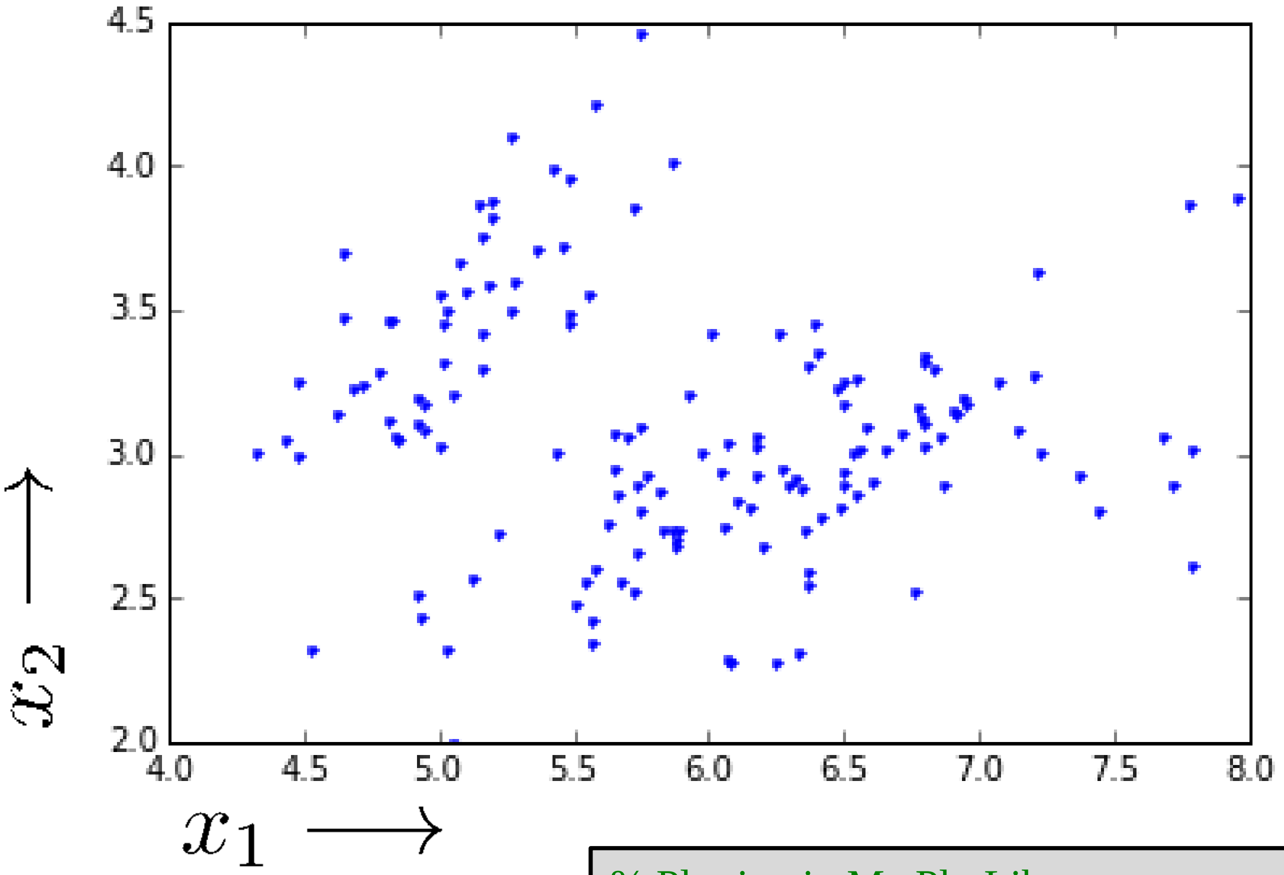
- Count the data falling in each of K bins
  - "Summarize" data as a length-K vector of counts (& plot)
  - Value of K determines "summarization"; depends on # of data
    - K too big: every data point falls in its own bin; just "memorizes"
    - K too small: all data in one or two bins; oversimplifies




```
% Histograms in MatPlotLib
import matplotlib.pyplot as plt
X1 = X[:,0]                   # extract first feature
Bins = np.linspace(4,8,17)    # use explicit bin locations
plt.hist( X1, bins=Bins )     # generate the plot
```

Scatterplots

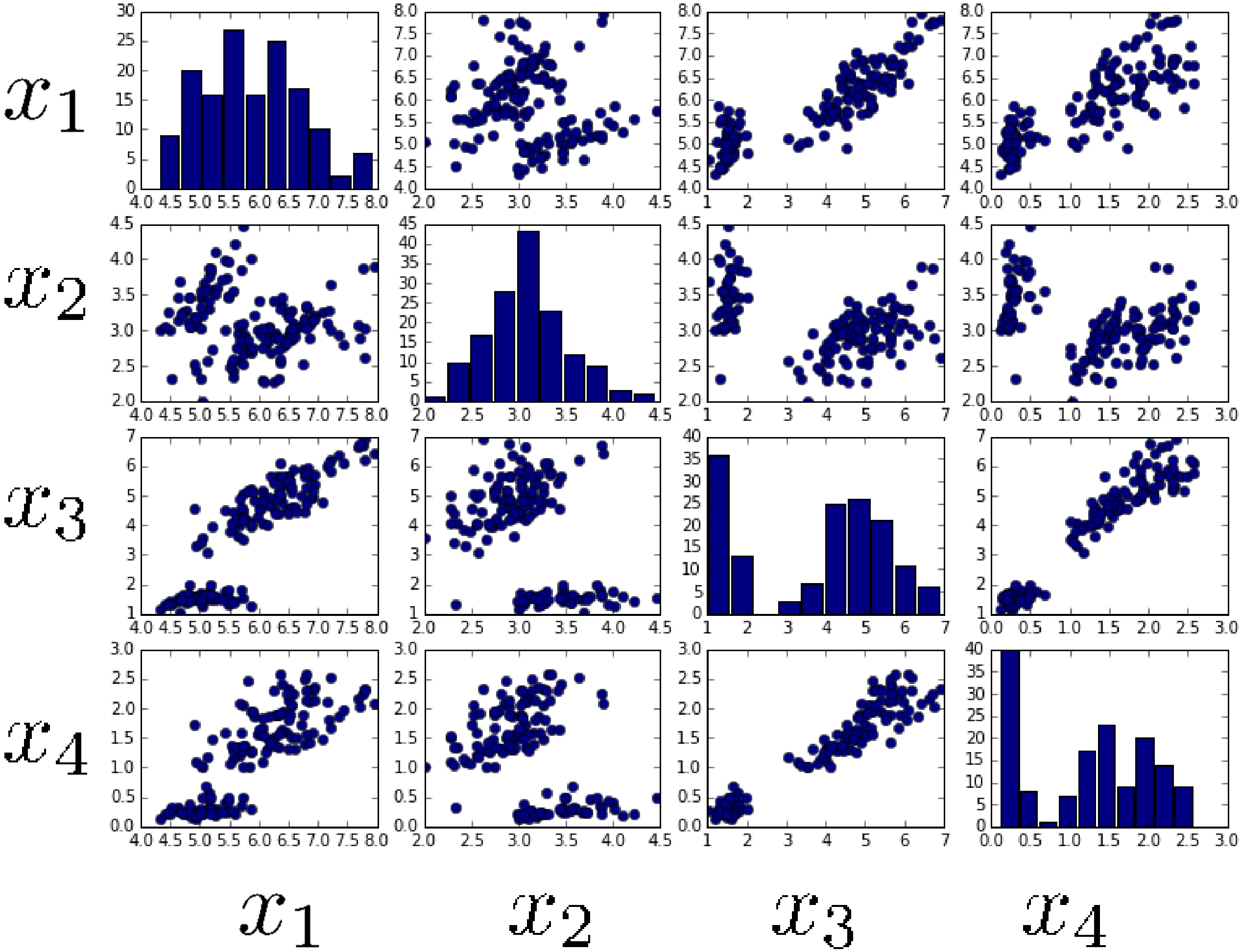
- Illustrate the relationship between two features




```
% Plotting in MatPlotLib
plt.plot(X[:,0], X[:,1], 'b.');    % plot data points as blue dots
```

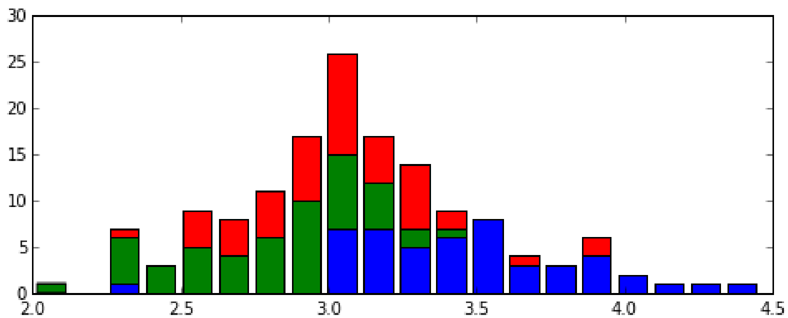
Scatterplots

- For more than two features we can use a pair plot:

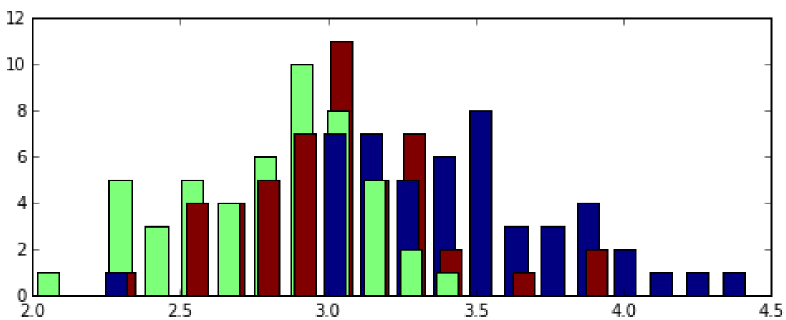


Supervised learning and targets

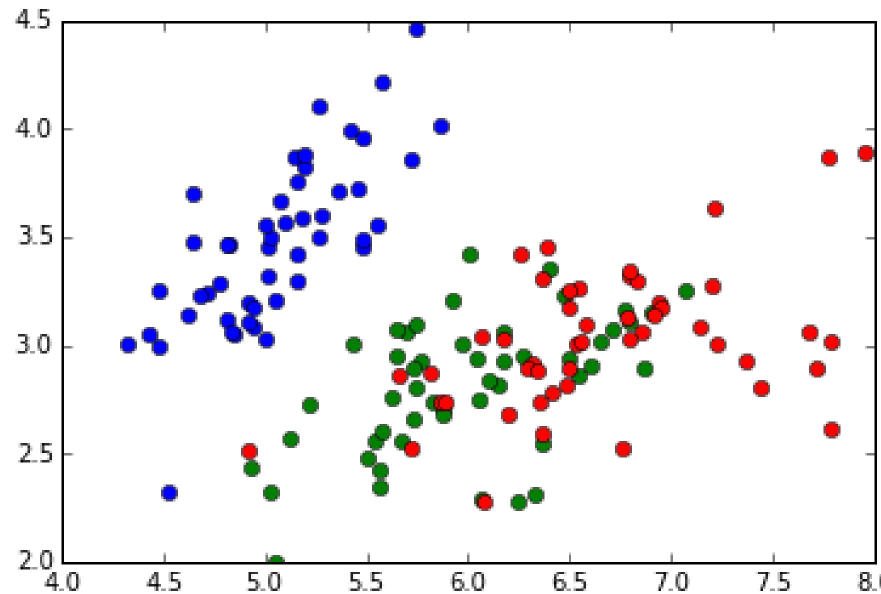
- Supervised learning: predict target values
- For discrete targets, often visualize with color




```
plt.hist( [X[Y==c,1] for c in np.unique(Y)] ,
            bins=20, histtype='barstacked’)
```



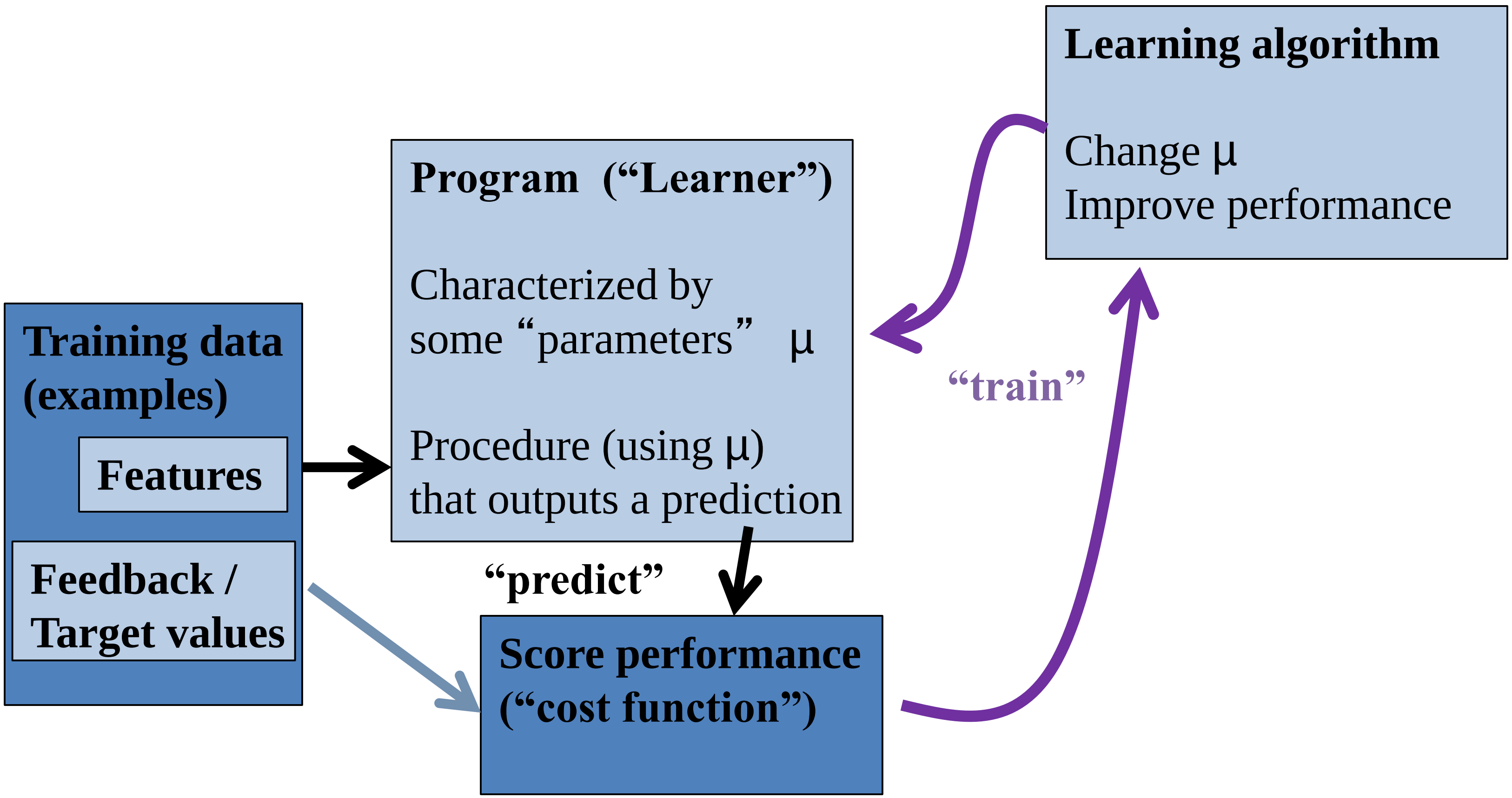

```
ml.histy(X[:,1], Y, bins=20)
```




```
colors = ['b','g','r']
for c in np.unique(Y):
    plt.plot( X[Y==c,0], X[Y==c,1], 'o',
              color=colors[int(c)] )
```

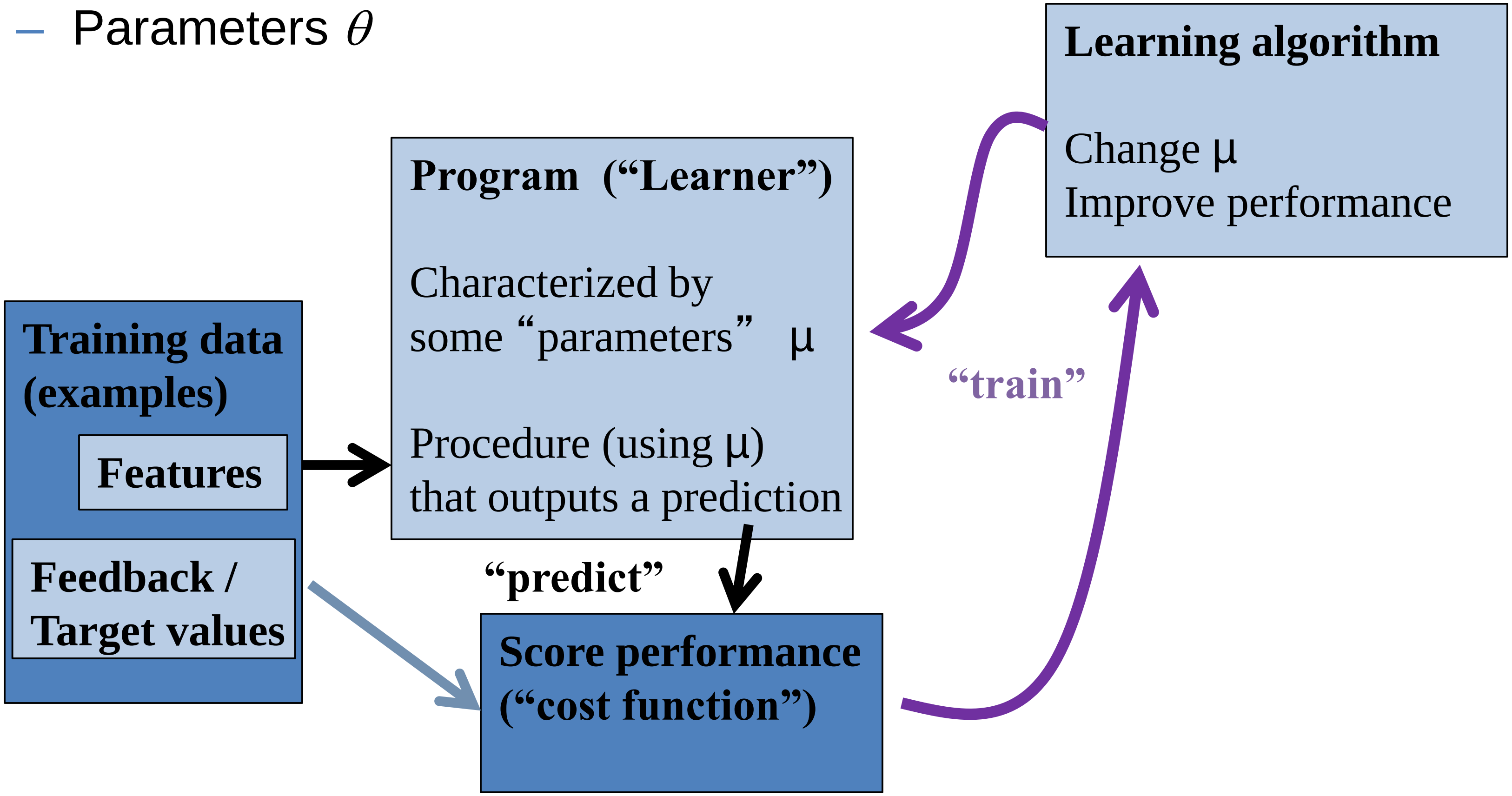
How does machine learning work?

- "Meta-programming"
  - Predict – apply rules to examples
  - Score – get feedback on performance
  - Learn – change predictor to do better

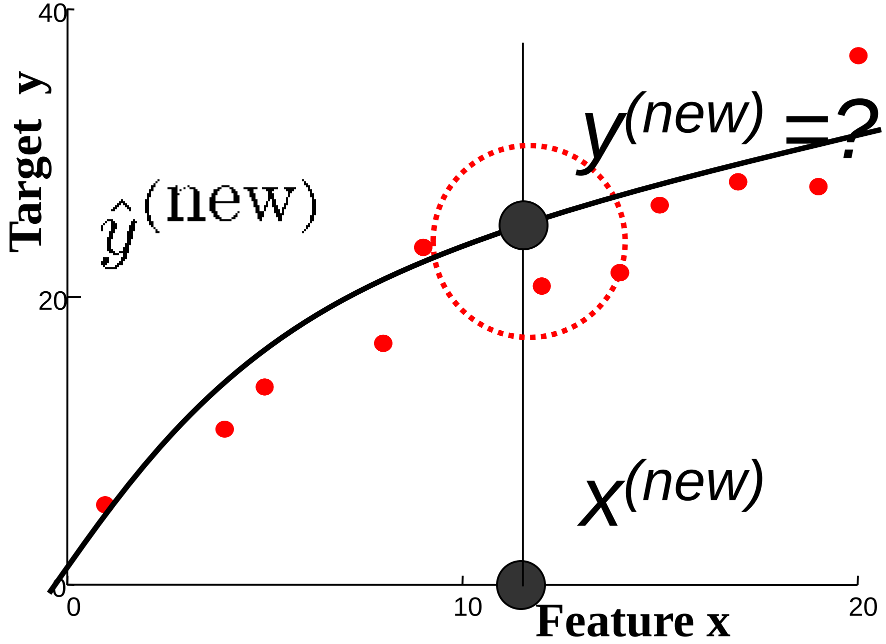


Supervised learning

- Notation
  - Features x
  - Targets y
  - Predictions $\hat{y} = f(x ; \theta)$
  - Parameters θ

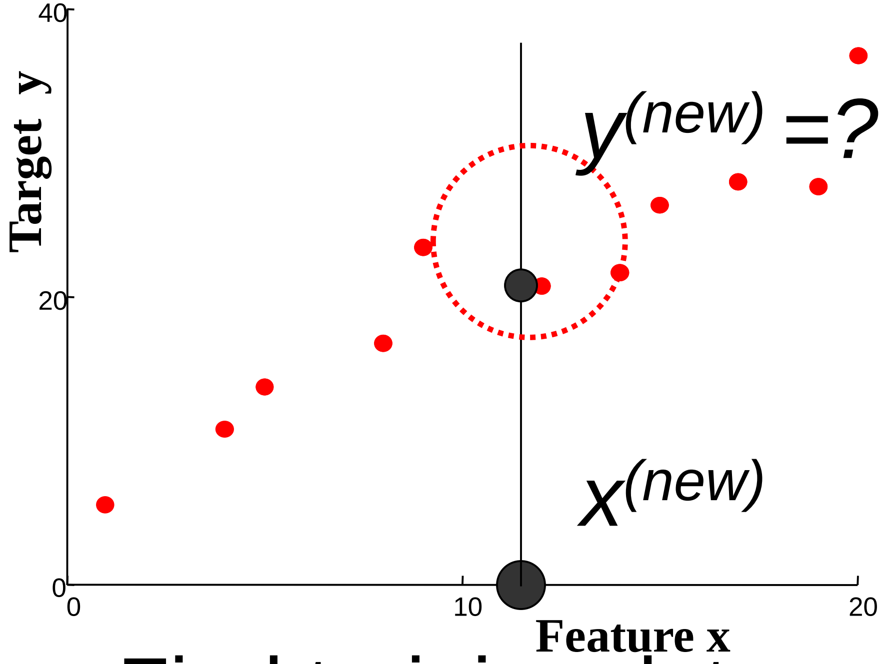


Regression; Scatter plots



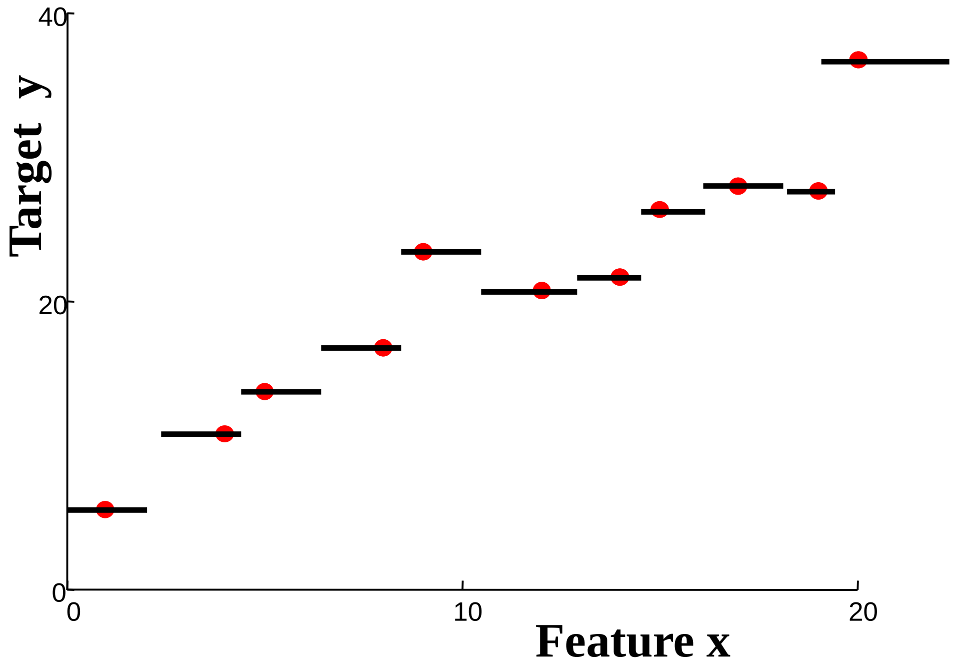
- Suggests a relationship between x and y
- *Prediction*: new x, what is y?

Nearest neighbor regression



- Find training datum $x^{(i)}$ closest to $x^{(new)}$
  Predict $y^{(i)}$

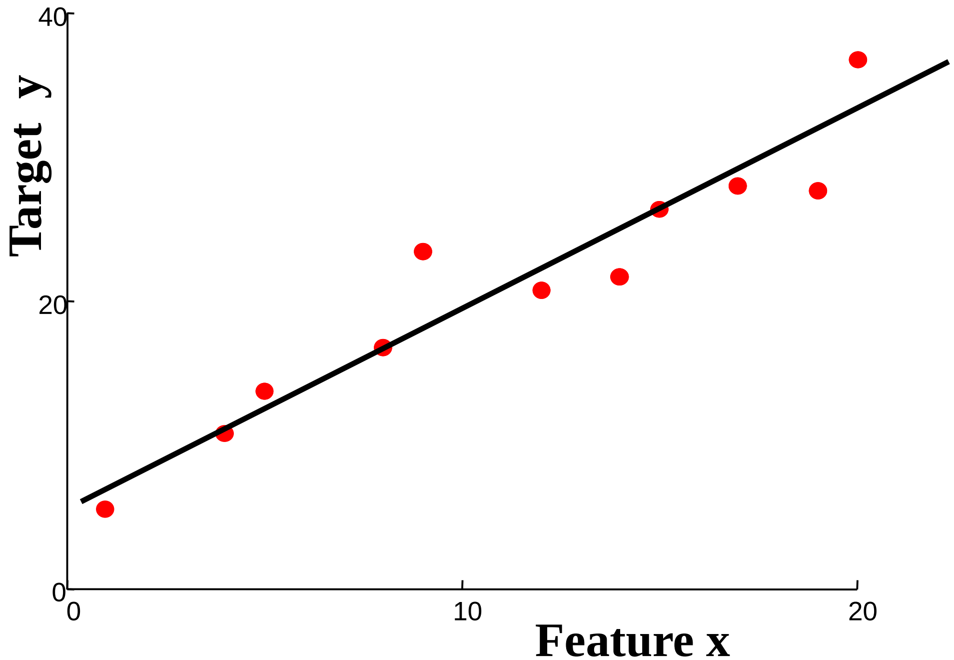
Nearest neighbor regression



"Predictor":
Given new features:
 Find nearest example
 Return its value

- Defines a function f(x) implicitly
- "Form" is piecewise constant

Linear regression



"Predictor":

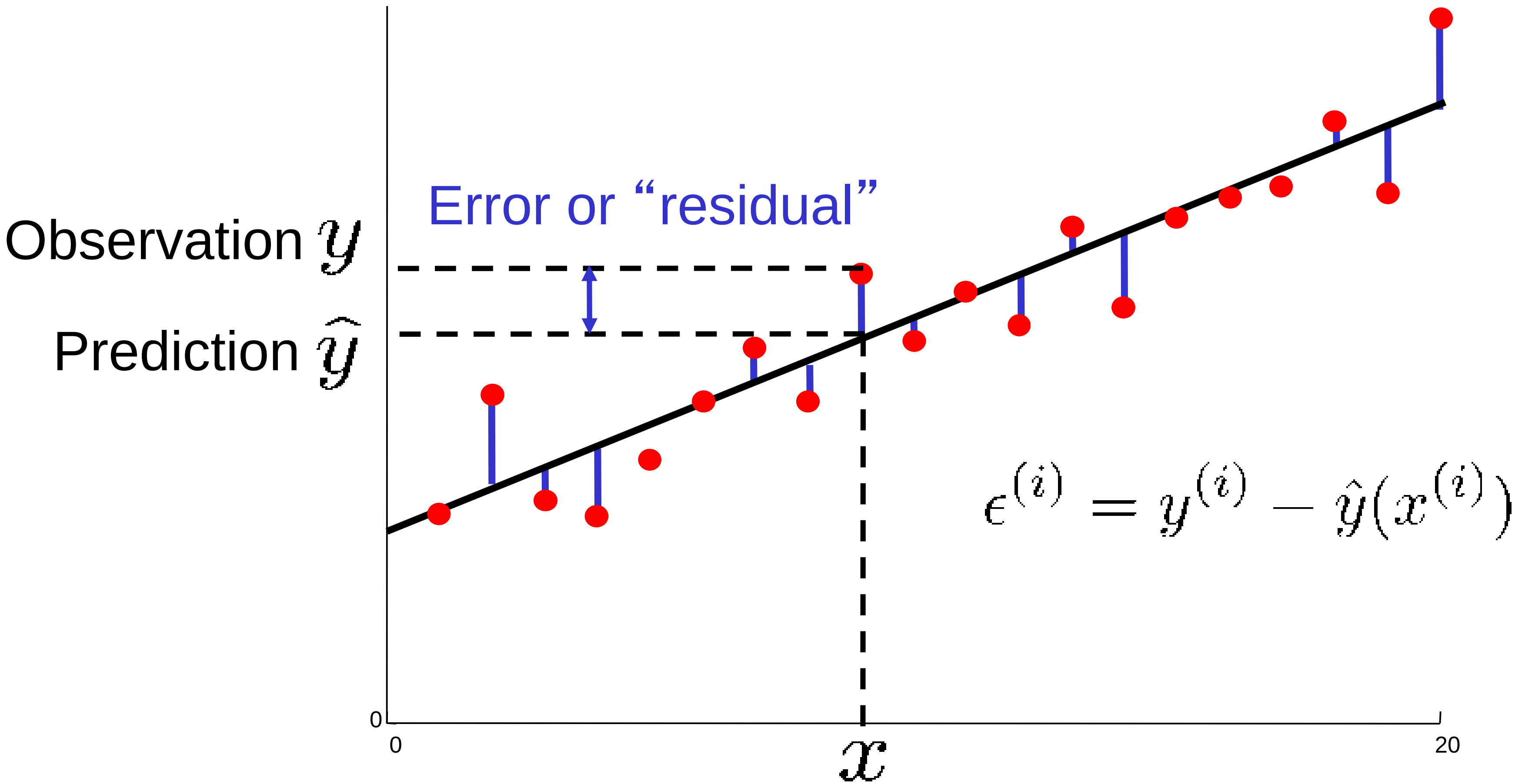
Evaluate line:

$$r = \theta_0 + \theta_1 x_1$$

return r

- Define form of function f(x) explicitly
- Find a good f(x) within that family

Measuring error

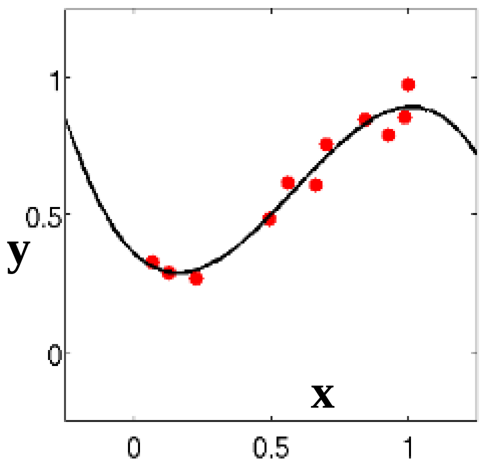


$$\epsilon^{(i)} = y^{(i)} - \hat{y}(x^{(i)})$$

$$\mathrm{MSE} = \frac{1}{m} \sum_i \left(y^{(i)} - \hat{y}(x^{(i)}) \right)^2$$

Regression vs. Classification

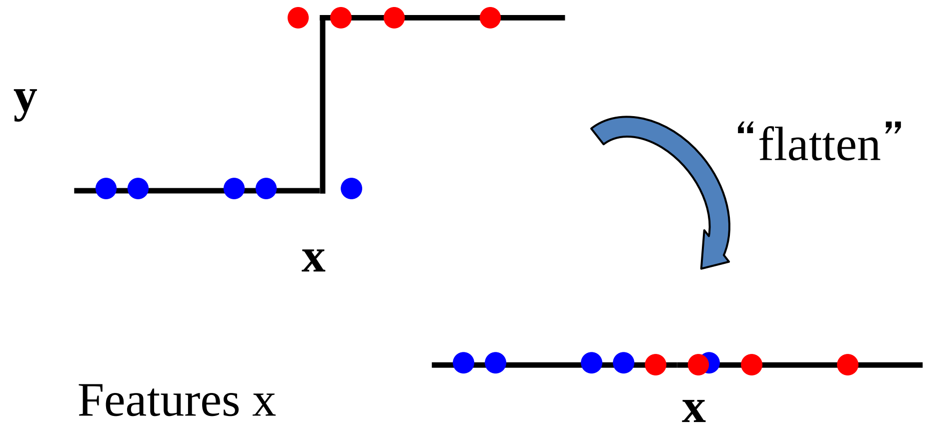
Regression



Features x
Real-valued target y

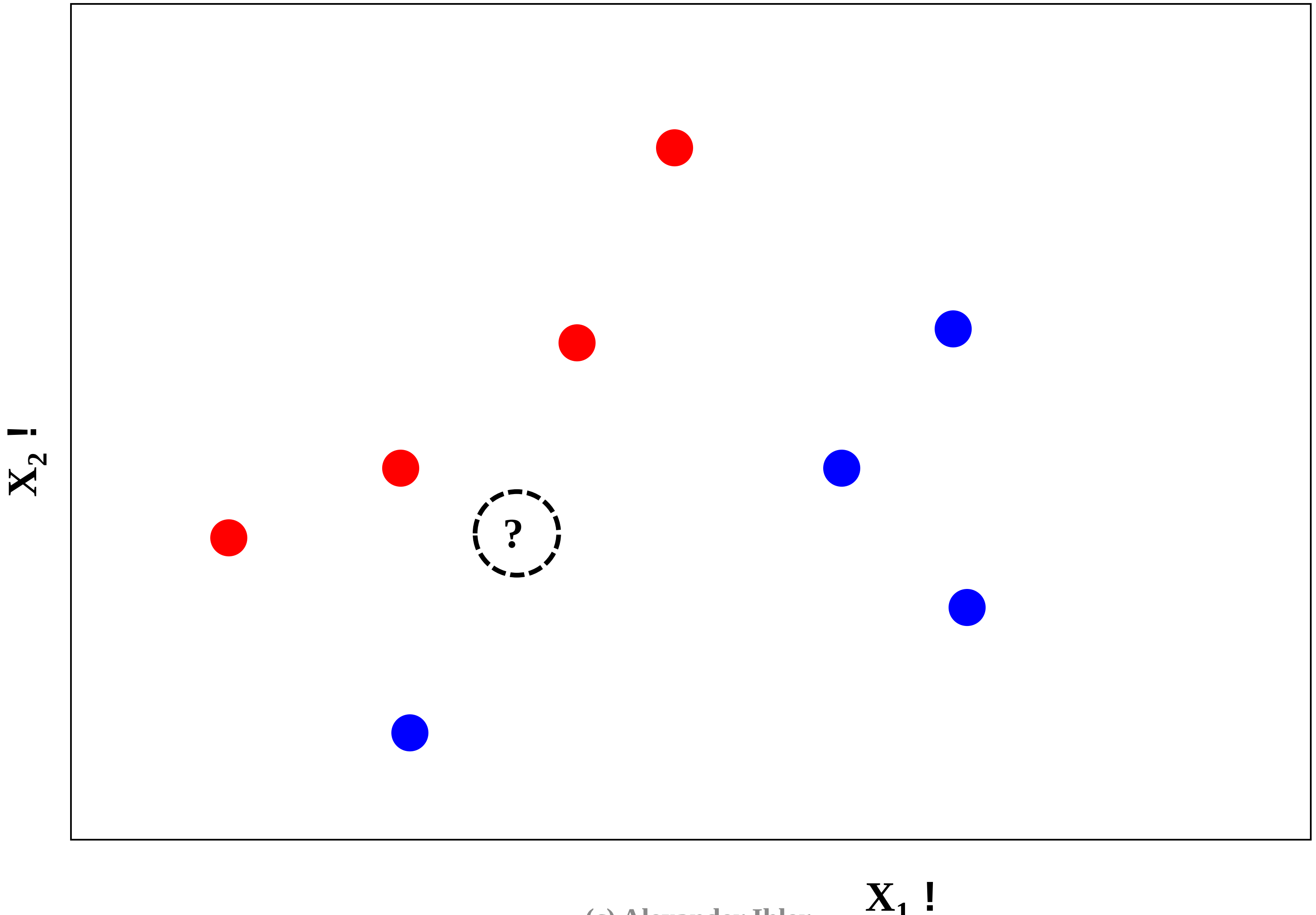
Predict continuous function ŷ(x)

Classification

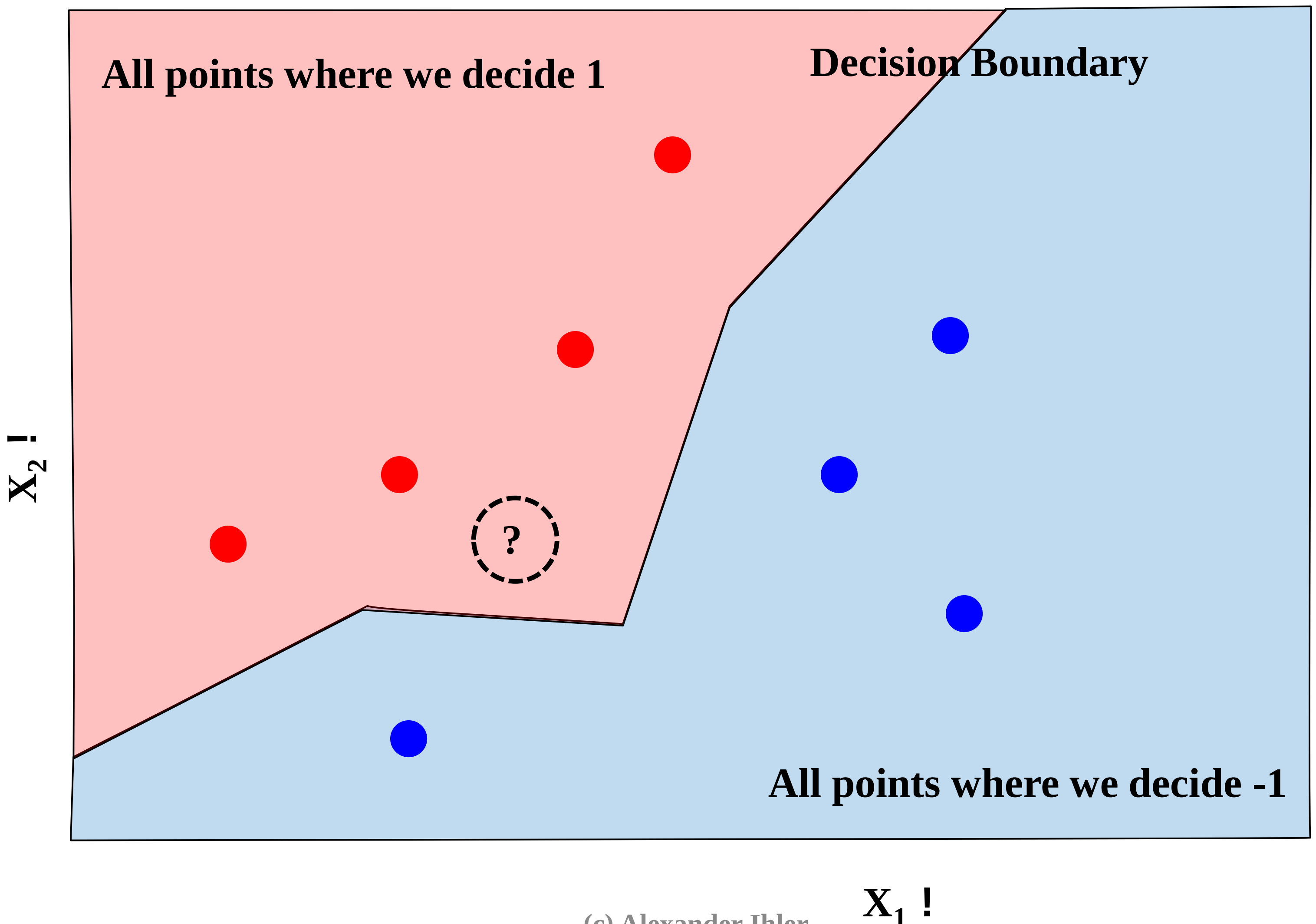


Features x
Discrete class c
(usually 0/1 or +1/-1)
Predict discrete function ŷ(x)

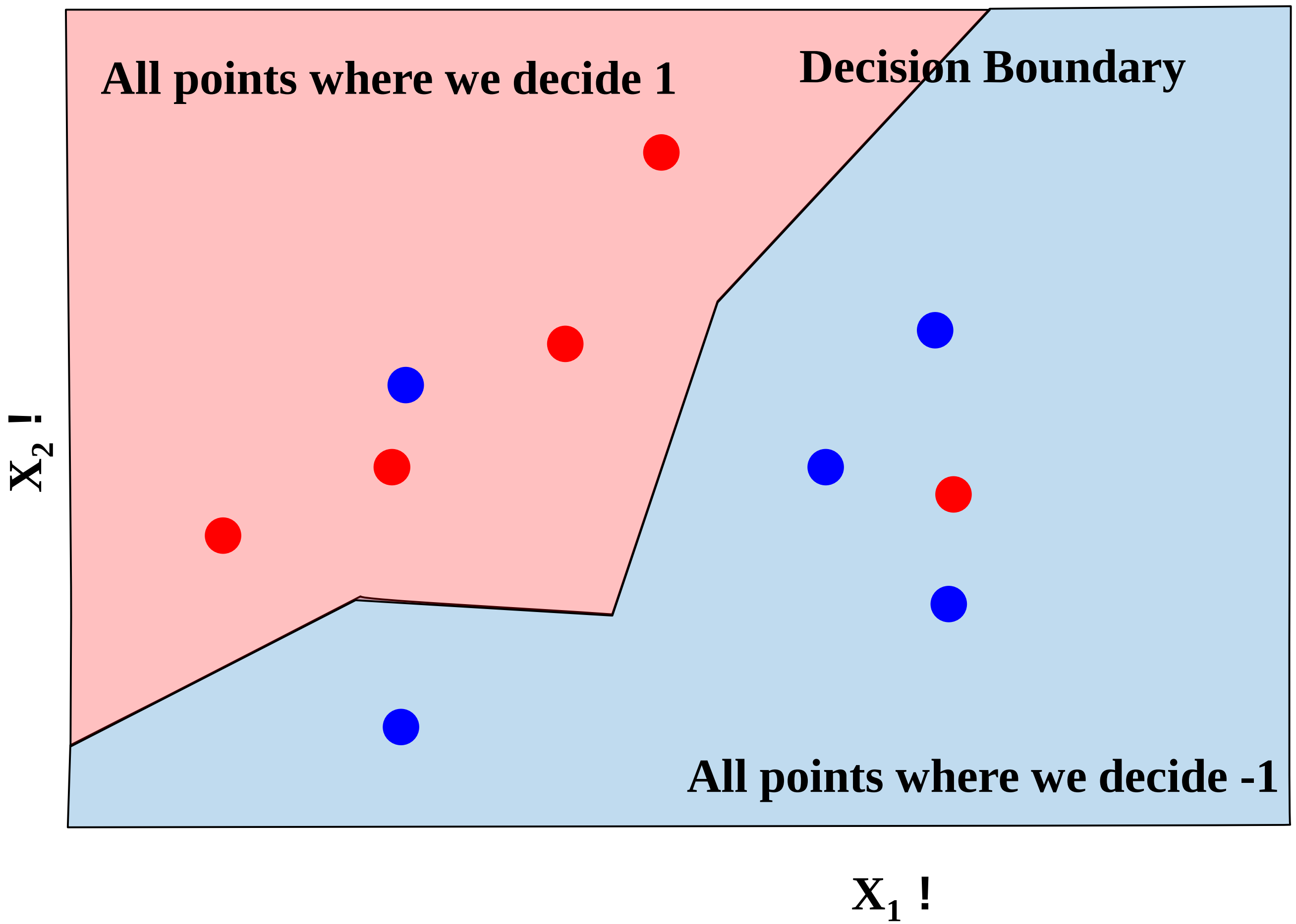
Classification



Classification



Measuring error



$$\mathrm{ERR} = \frac{1}{m} \sum_i \left[\, y^{(i)} \neq \hat{y}(x^{(i)}) \,\right]$$

A simple, optimal classifier

- Classifier f(x ; μ)
  - maps observations x to predicted target values
- Simple example
  - Discrete feature x: f(x ; μ) is a contingency table
  - Ex: spam filtering: observe just X_1 = in contact list?
- Suppose we knew the true conditional probabilities:
- Best prediction is the most likely target!

Feature	spam	keep
X=0	0.6	0.4
X=1	0.1	0.9

"Bayes error rate"

Pr[X=0] * Pr[wrong | X=0]   + Pr[X=1] * Pr[wrong | X=1]

= Pr[X=0] * (1- Pr[Y=S | X=0]) + Pr[X=1] * (1-Pr[Y=K | X=1])

Optimal least-squares regression

- Suppose that we know true p(X,Y)
- Prediction f(x): *arbitrary* function
  - Focus on some specific x: f(x) = v
- Expected squared error loss is

$$\mathbb{E}_{Y|X=x}\left[(Y-v)^2\right] = \int p(Y|X=x)(Y-v)^2 \, dY$$

- Minimum: take derivative & set to zero

$$\frac{\partial}{\partial v} \int p(Y|X=x)(Y-v)^2 \, dY = \int p(Y|X=x) 2(Y-v) = 0$$

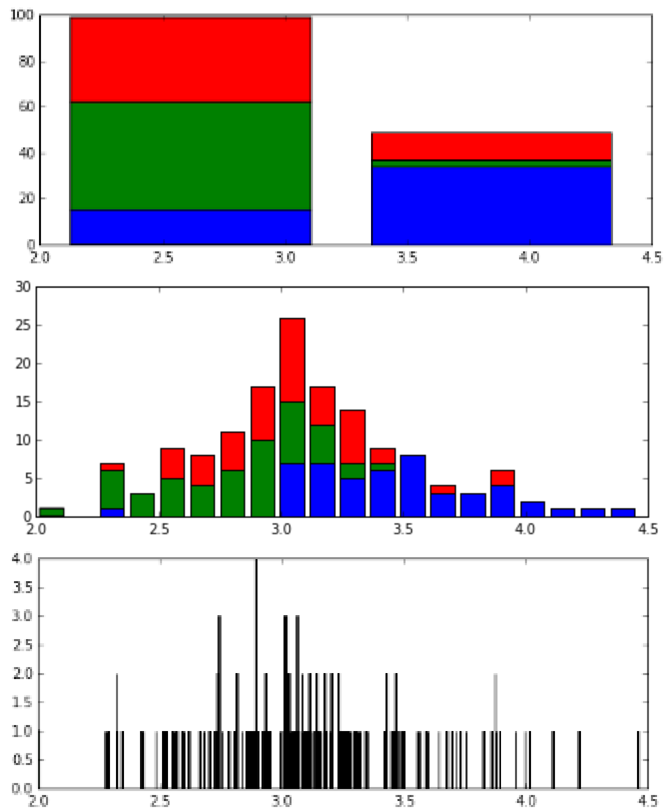
$$\Rightarrow \qquad 2 \int p(Y|X=x) Y = 2 \Big(\int p(Y|X=x) \Big) v$$

$$\Rightarrow \qquad v = \int p(Y|X=x) Y = \mathbb{E}_{Y|X=x}[Y]$$

Optimal estimate of Y: conditional expectation given X

Bayes classifier, estimated

- Now, let's see what happens with "real" data
  - Use empirically estimated probability **model** for p(x,y)
- Iris data set, first feature only (real-valued)
  - We can estimate the probabilities (e.g., with a histogram)



2 Bins:

Predict "green" if X < 3.25, else "blue"

Model is "too simple"

20 Bins:

Predict by majority color in each bin

500 Bins:

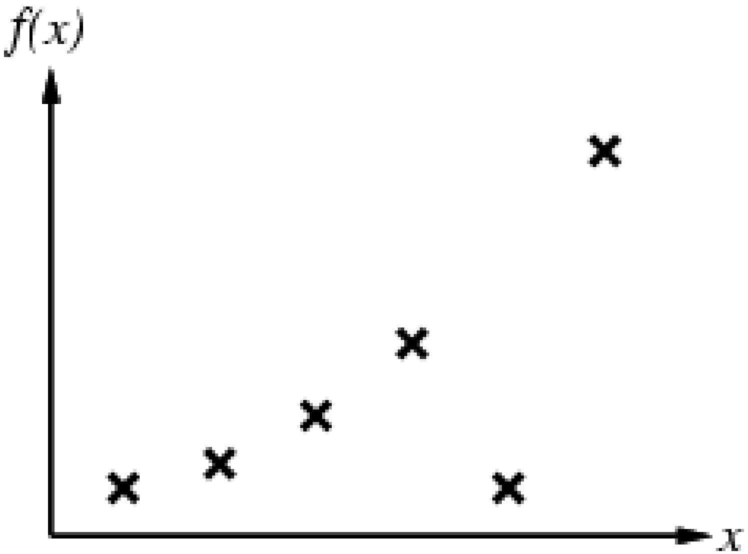
Each bin has ~ 1 data point!

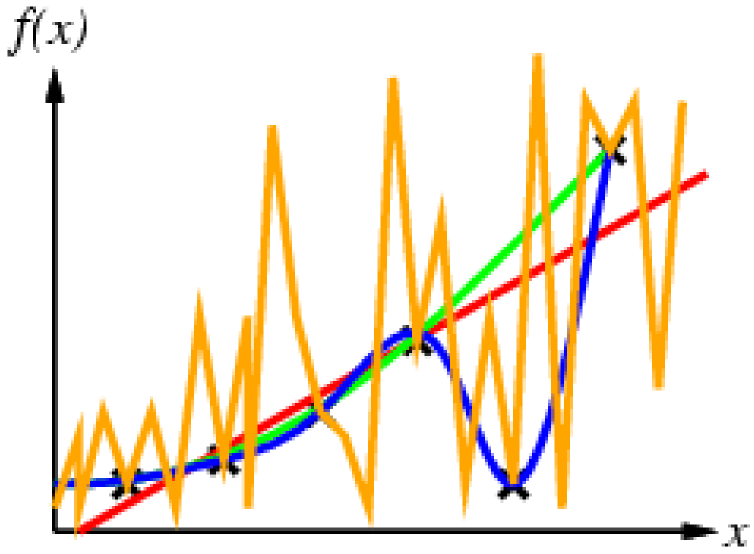
What about bins with 0 data?

Model is "too complex"

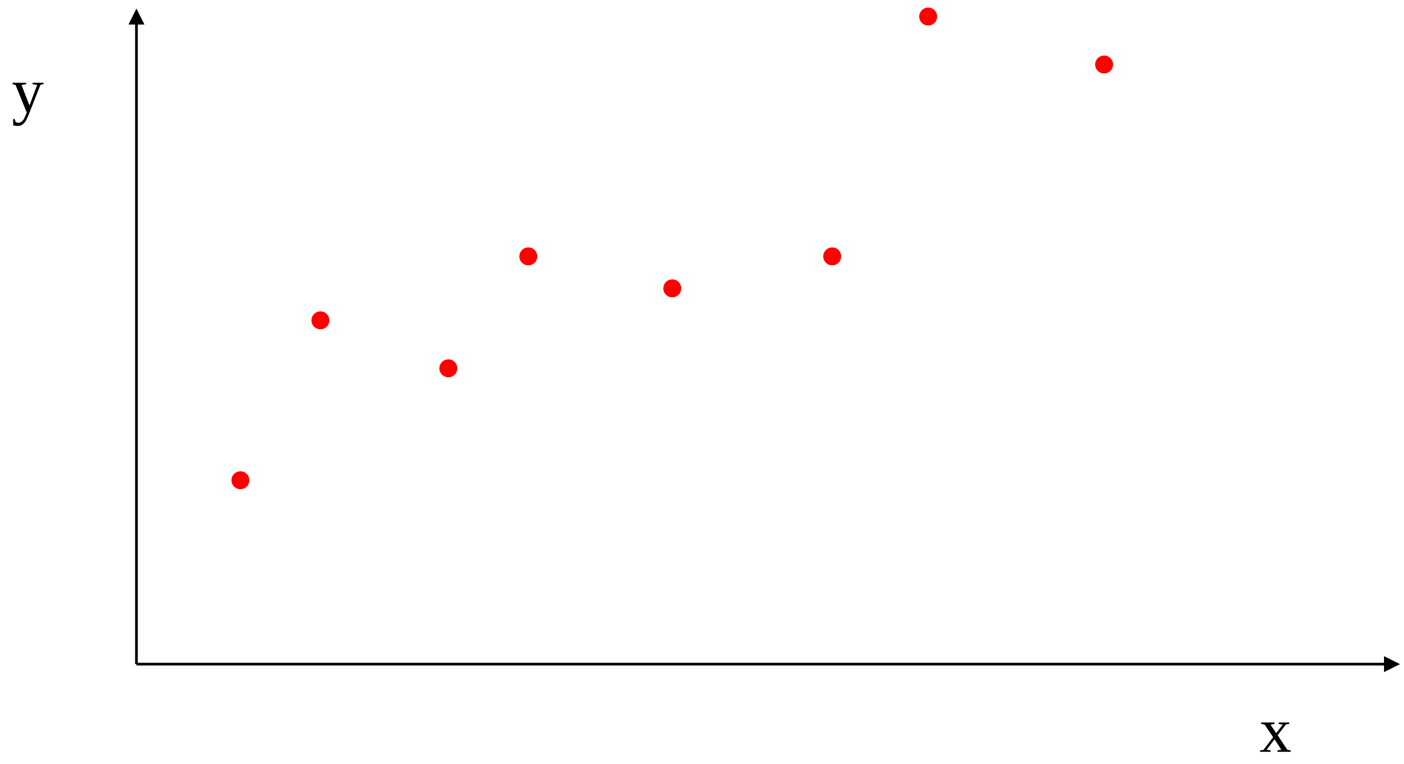
Inductive bias

- "Extend" observed data to unobserved examples
  - "Interpolate" / "extrapolate"
- What kinds of functions to expect? Prefer these ("bias")
  - Usually, let data pull us away from assumptions only with evidence!



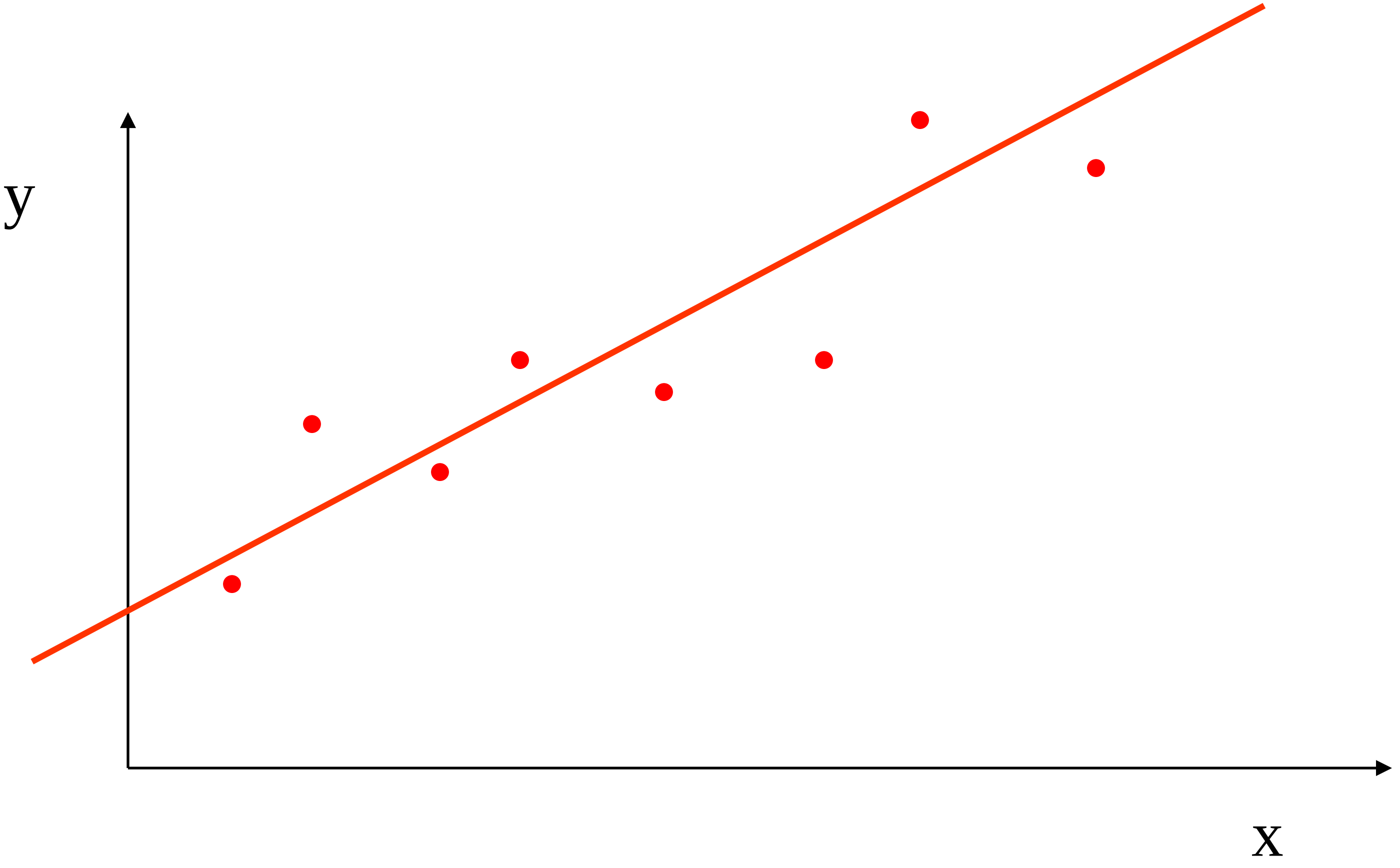


Overfitting and complexity

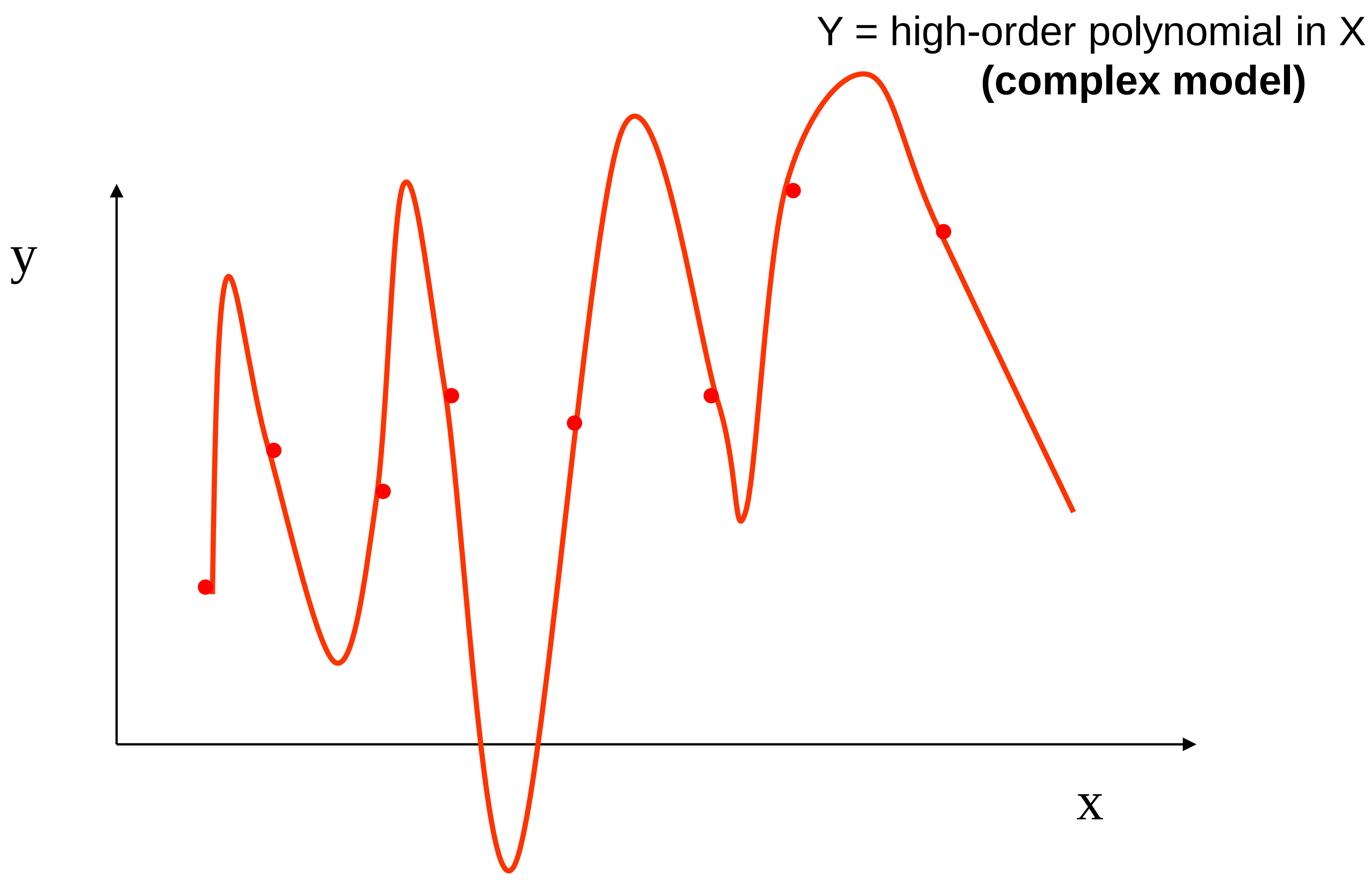


Overfitting and complexity

Simple model: Y= aX + b + e

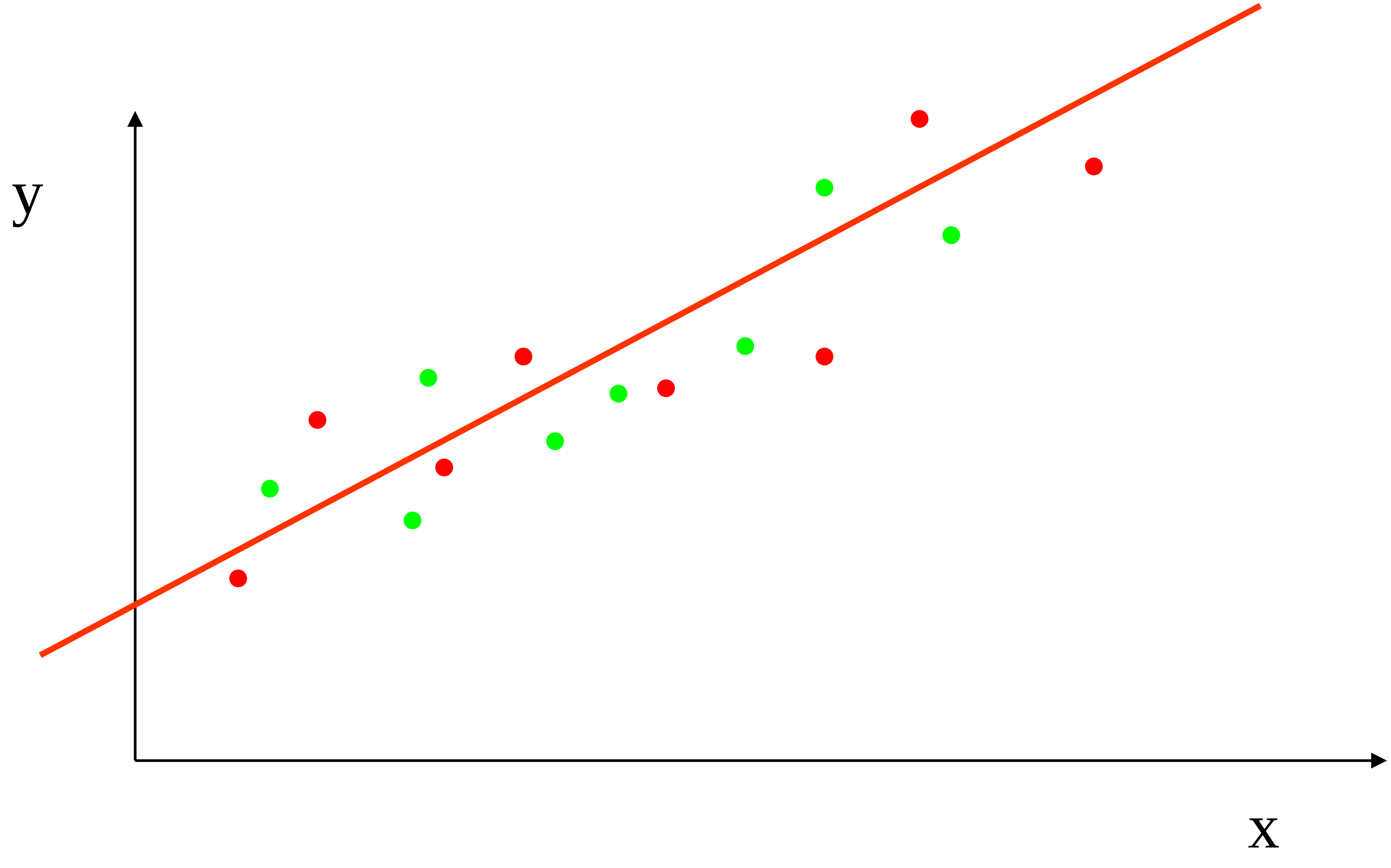


Overfitting and complexity

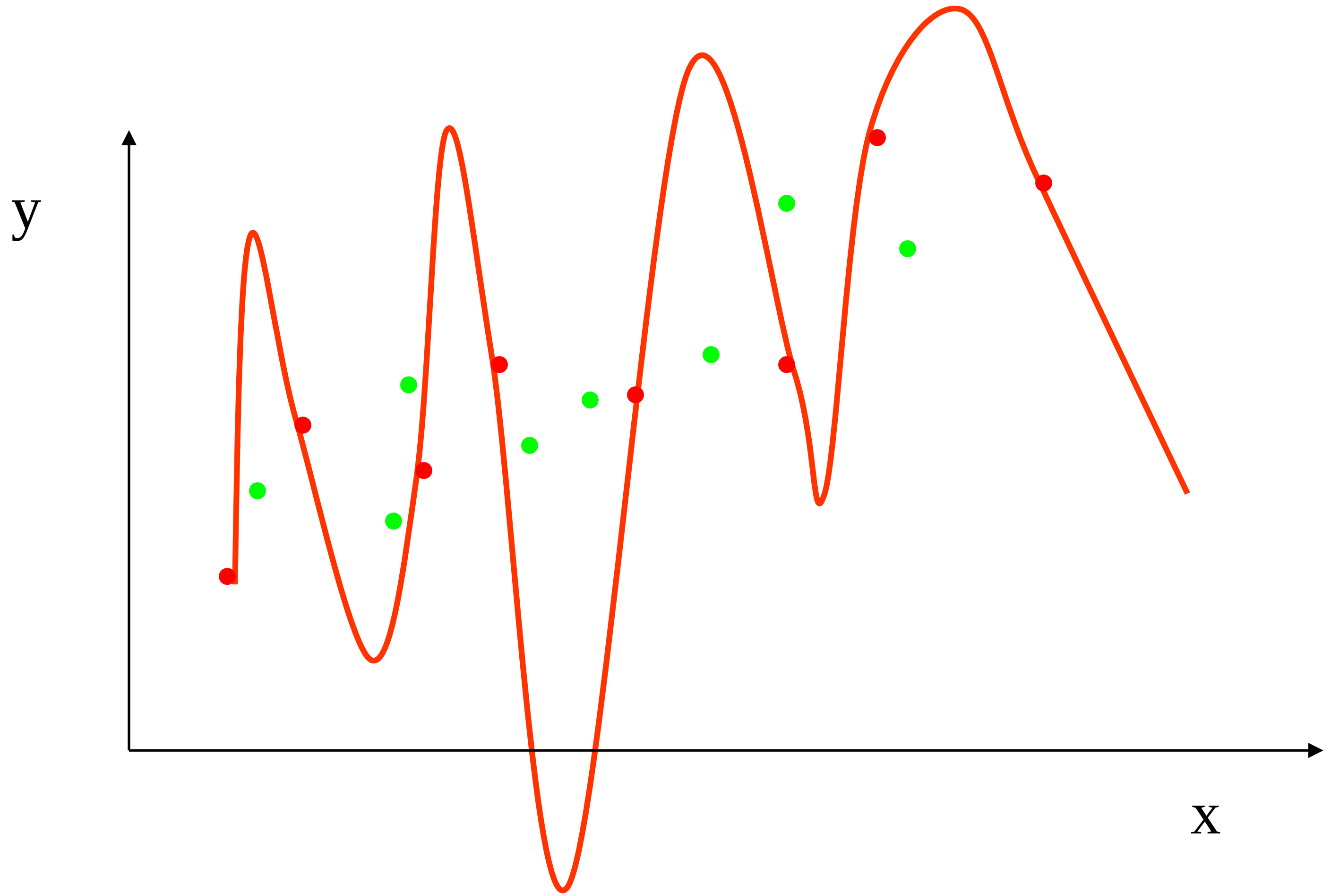


Overfitting and complexity

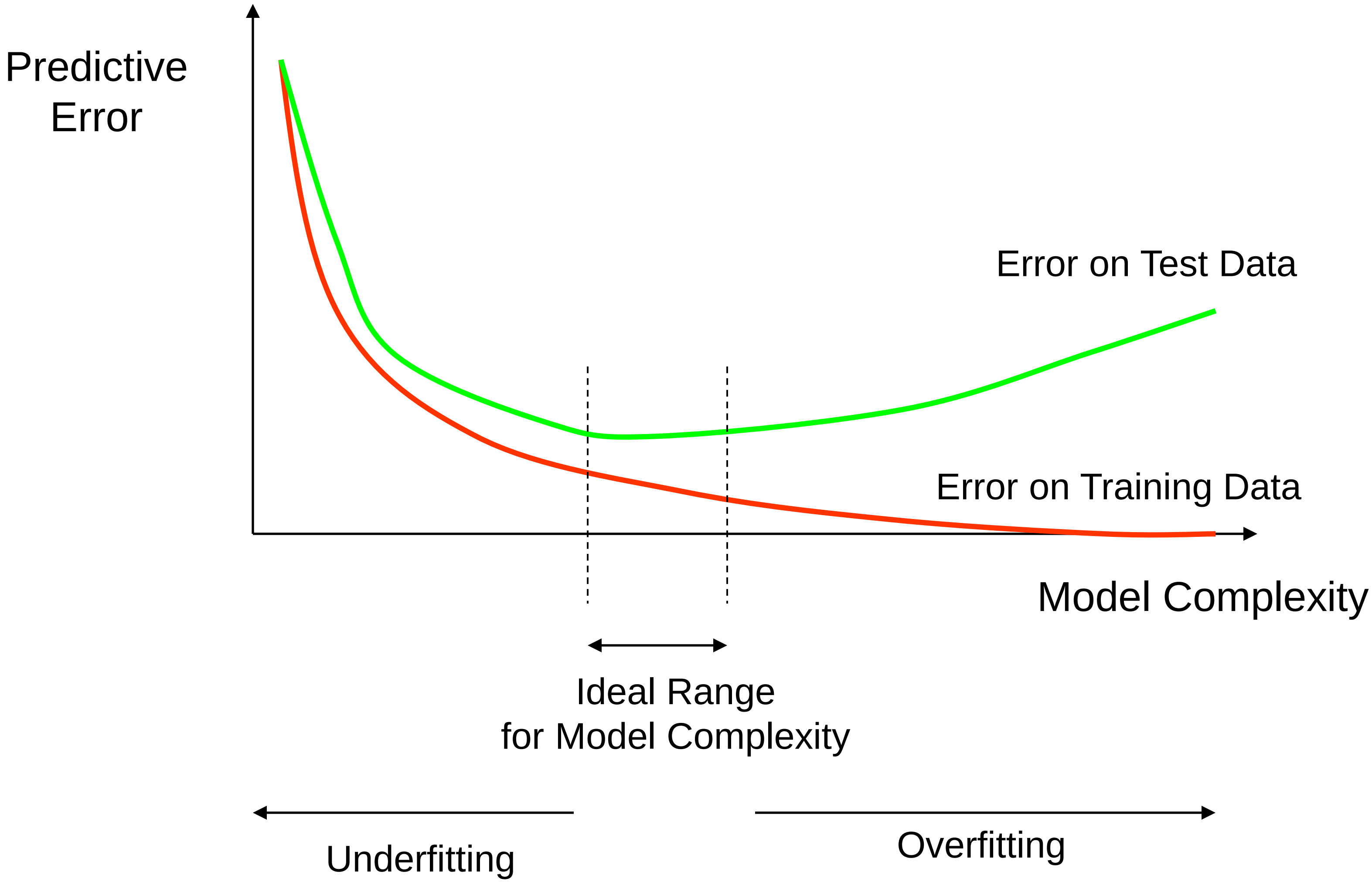
Simple model: Y= aX + b + e



Overfitting and complexity



How Overfitting affects Prediction



Bias vs Variance

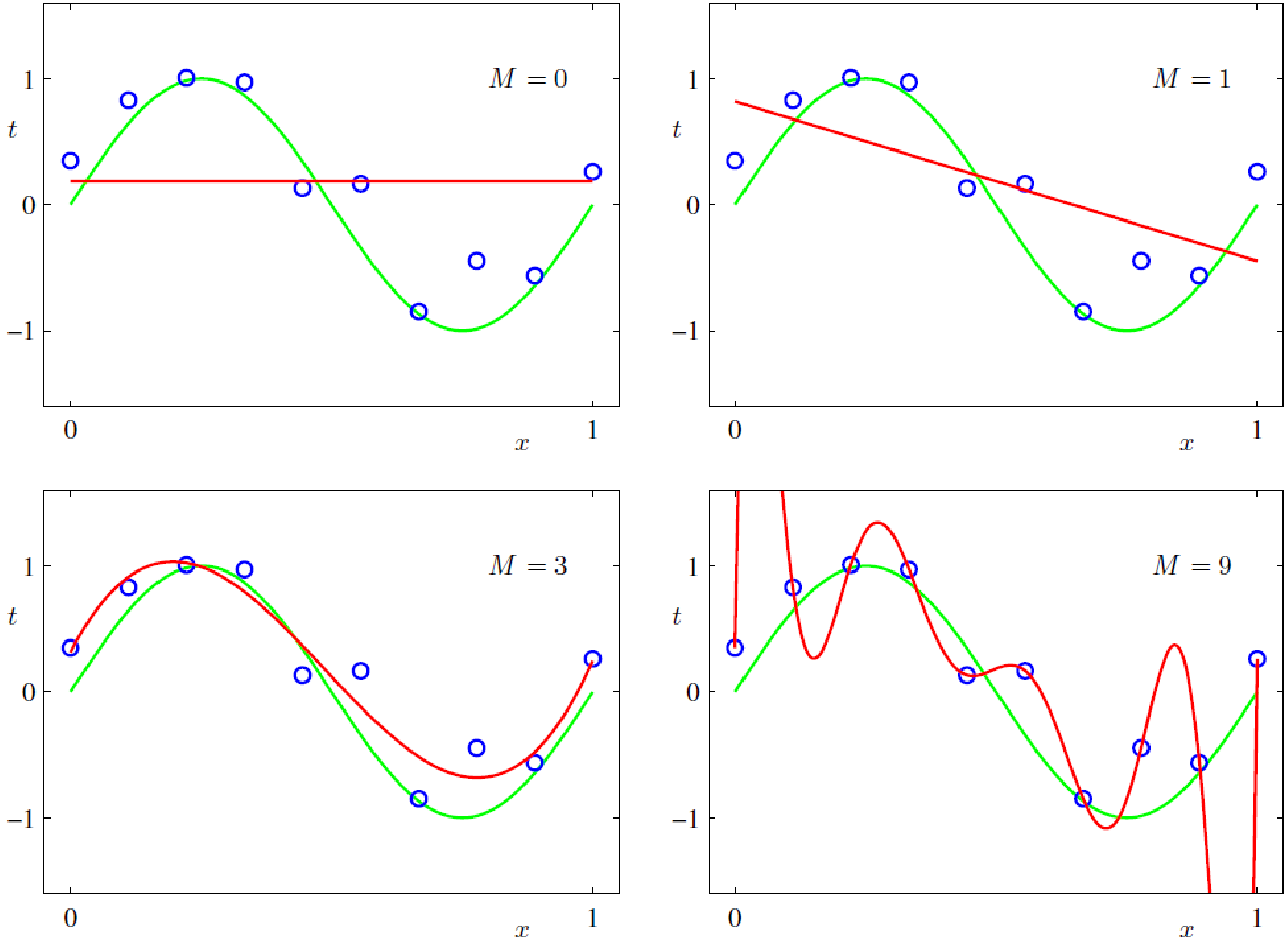
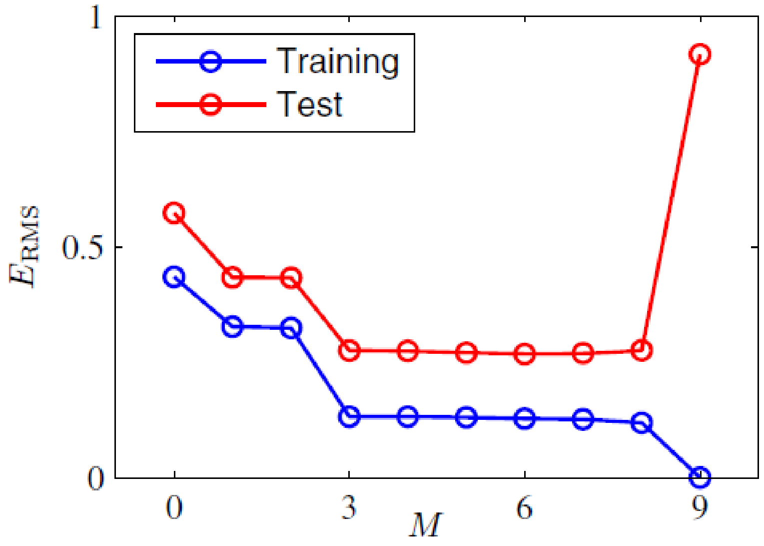


Figure 1.4 Plots of polynomials having various orders M, shown as red curves, fitted to the data set shown in Figure 1.2.

Bias vs Variance



Bias vs Variance

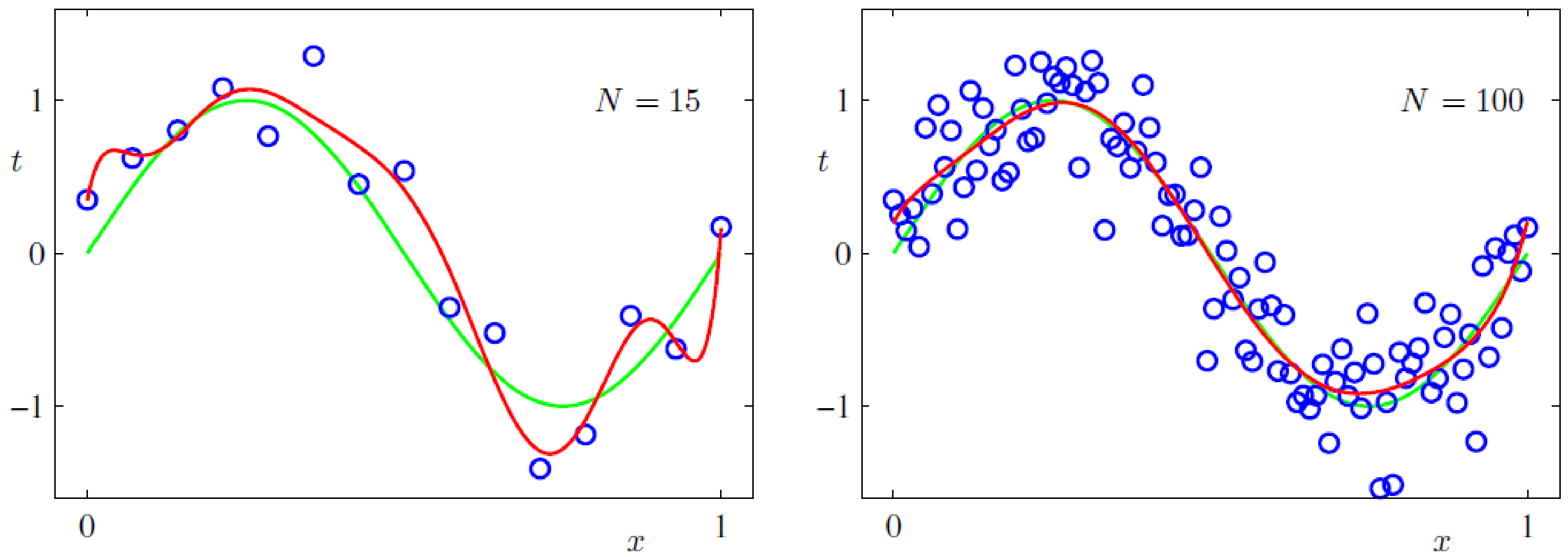


Figure 1.6 Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

Bias vs Variance

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

	$\ln\lambda=-\infty$	$\ln\lambda=-18$	$\ln\lambda=0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

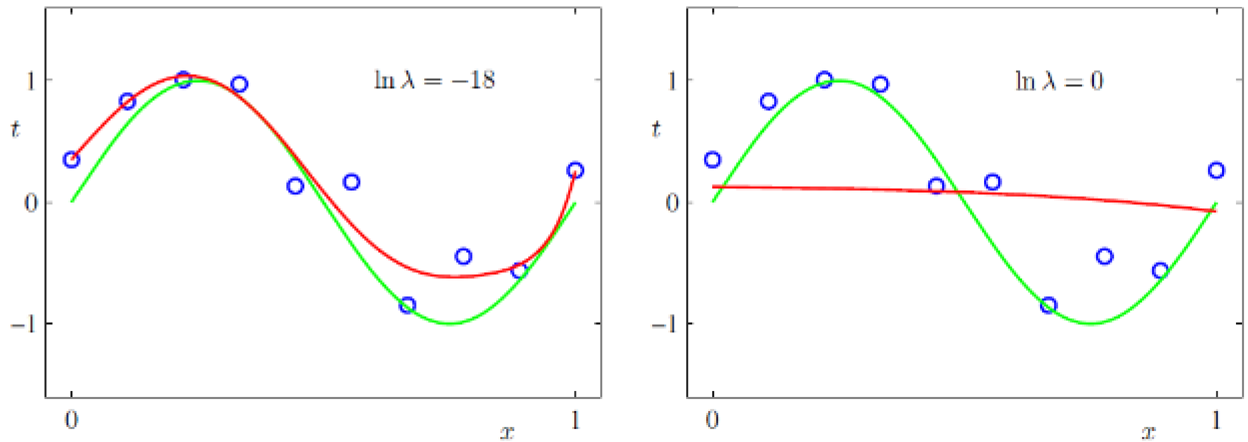


Figure 1.7 Plots of $M = 9$ polynomials fitted to the data set shown in Figure 1.2 using the regularized error function (1.4) for two values of the regularization parameter λ corresponding to $\ln\lambda = -18$ and $\ln\lambda = 0$. The case of no regularizer, i.e., $\lambda = 0$, corresponding to $\ln\lambda = -\infty$, is shown at the bottom right of Figure 1.4.

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Bias vs Variance

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

Where:

$$\mathrm{Bias}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x) - f(x)\right]$$

and

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

The expectation ranges over different choices of the training set $x_1, \ldots, x_n, y_1, \ldots, y_n$, all sampled from the same joint distribution $P(x, y)$. The three terms represent:

- the square of the *bias* of the learning method, which can be thought of as the error caused by the simplifying assumptions built into the method. E.g., when approximating a non-linear function $f(x)$ using a learning method for linear models, there will be error in the estimates $\hat{f}(x)$ due to this assumption;
- the *variance* of the learning method, or, intuitively, how much the learning method $\hat{f}(x)$ will move around its mean;
- the irreducible error σ^2. Since all three terms are non-negative, this forms a lower bound on the expected error on unseen samples.[4]:34

Learner Validation & Testing

- Training data
  - Used to build your model(s)
- Validation data
  - Used to assess, select among, or combine models
  - Personal validation; leaderboard; …
- Test data
  - Used to estimate "real world" performance

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UT
1	-	BrickMover *	1.21251	40	Sat, 31 Aug 2013 23:
2	new	vsu *	1.21552	13	Sat, 31 Aug 2013 20:
3	↑2	Merlion	1.22724	29	Sat, 31 Aug 2013 23:
4	↓2	Sergey	1.22856	15	Sat, 31 Aug 2013 23:
5	new	liuyongqi	1.22980	13	Sat, 31 Aug 2013 13:

Summary

- What is machine learning?
  - Types of machine learning
  - How machine learning works
- Supervised learning
  - Training data: features x, targets y
- Regression
  - (x,y) scatterplots; predictor outputs f(x); optimal MSE predictor
- Classification
  - (x,x) scatterplots
  - Decision boundaries, colors & symbols; Bayes optimal classifier
- Complexity
  - Training vs test error
  - Under- & over-fitting