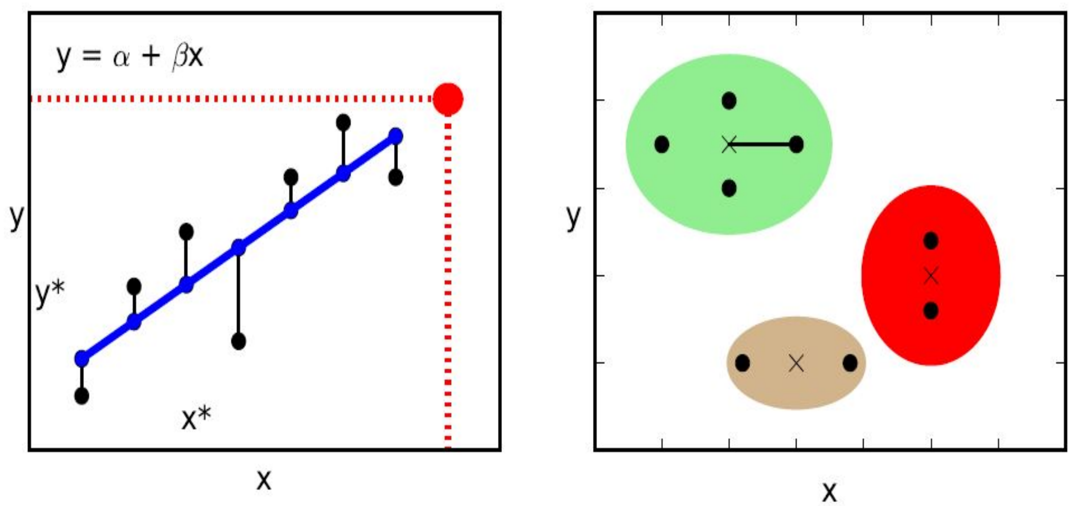
CONCEPTS

AND

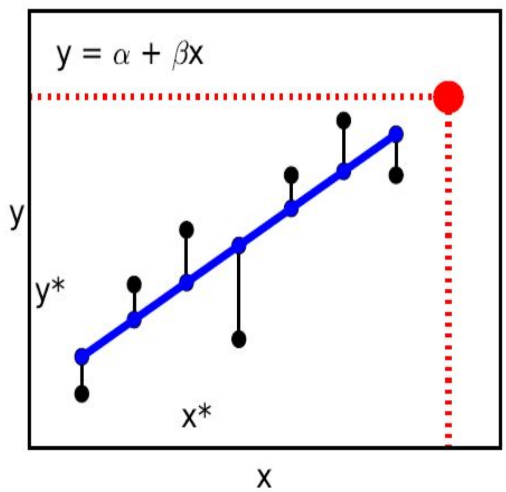
DEFINITIONS

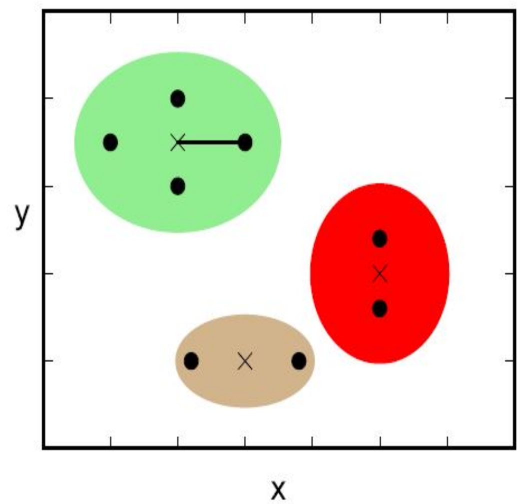
Prediction vs. Classification



- two main goals in machine learning

Examples





- stock price prediction
- spam/no spam email classification

A Numerical Dataset

object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

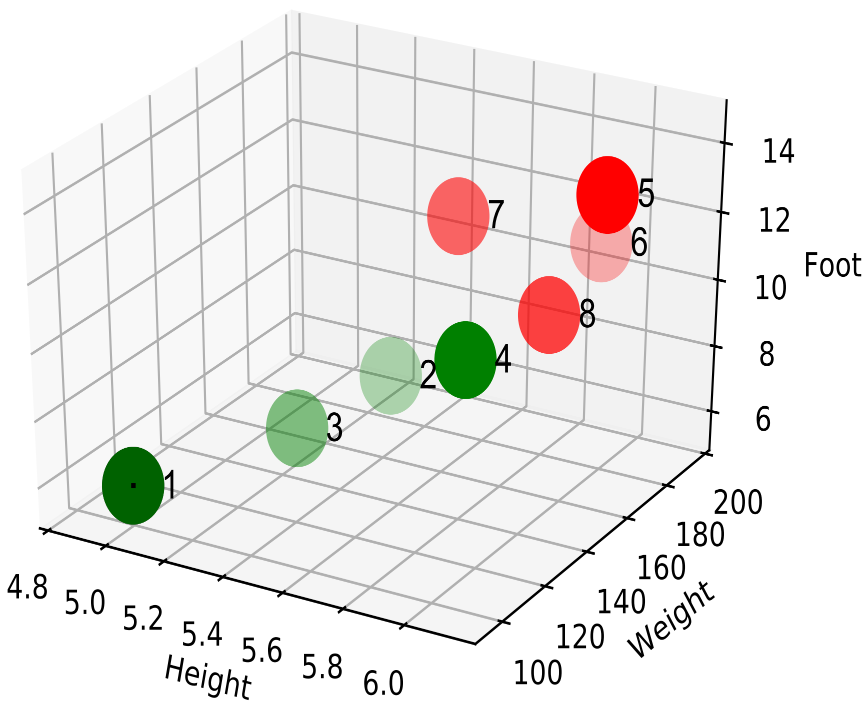
Code for the Dataset

```
import pandas as pd
data = pd.DataFrame(
      {"id": [ 1,2,3,4,5,6,7,8],
       "Label": ["green","green",
                "green","green",
                "red","red",
                "red","red"],
       "Height": [5, 5.5, 5.33,5.75,
                 6.00,5.92,5.58,5.92],
       "Weight": [100, 150, 130, 150,
                   180, 190, 170, 165],
       "Foot": [6, 8, 7, 9,
                13, 11, 12, 10]},
        columns = ["id", "Height",
                   "Weight", "Foot",
                   "Label"] )
```

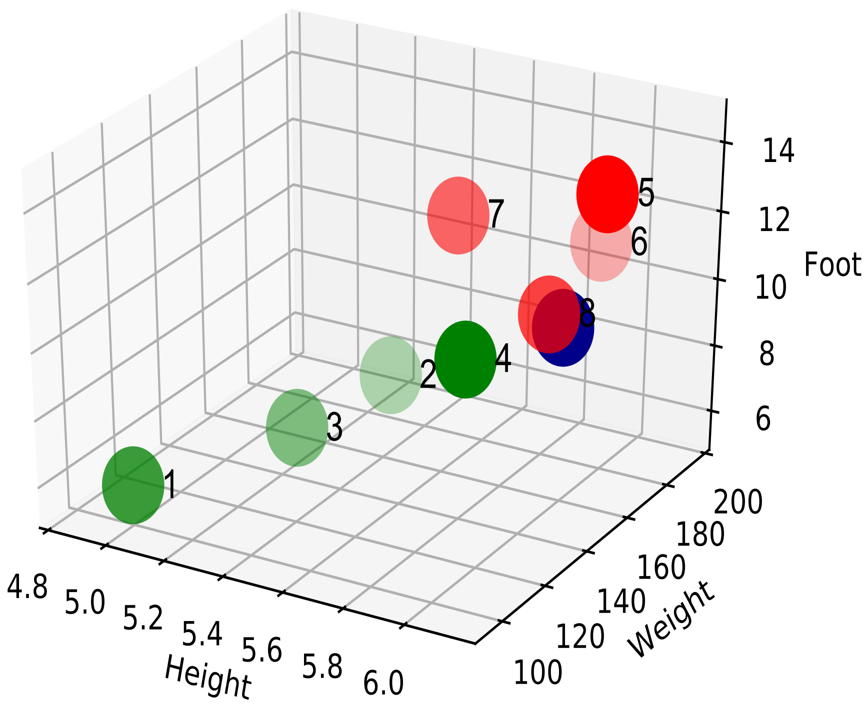
Code for the Dataset (cont'd)

```
ipdb> data
  id Height Weight Foot Label
0  1  5.00    100    6  green
1  2  5.50    150    8  green
2  3  5.33    130    7  green
3  4  5.75    150    9  green
4  5  6.00    180   13    red
5  6  5.92    190   11    red
6  7  5.58    170   12    red
7  8  5.92    165   10    red
```

A Dataset Illustration



A New Instance



$$(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto ?$$

Categorical Dataset

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
2	rainy	mild	high	**yes**
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- $\text{x}^* = (\text{sunny, cold, low}) \mapsto ?$
- need numeric values for attributes

Python Code

```
import pandas as pd

data = pd.DataFrame(
   {"Day":[1,2,3,4,5,6,7,8,9,10],
    "Weather": ["sunny","rainy","sunny",
        "rainy","sunny","overcast",
        "sunny","overcast",
        "rainy","rainy"],
    "Temperature":["hot", "mild",
        "cold","cold","cold",
        "mild","hot","hot",
        "hot","mild"],
    "Wind":["low","high","low",
        "high","high","low",
        "low","high","high","low"],
    "Play":["no","yes","yes","no","yes",
        "yes","yes","yes","no","yes"]},
    columns=["Day","Weather",
            "Temperature","Wind","Play"])
```

Basic Definitions

- eight objects: $x_1, \ldots, x_8$
- three features: height H, weight W, foot F
- for input x, its feature vector $\phi(x) = [H(x), W(x), F(x)]$
- $\phi(x)$ is a point in 3-dimensional space R^3

Examples of Objectives

1. binary classification: compute label (green or red) on new instance

$$x \mapsto \phi(\cdot) \mapsto y \in \{0, 1\}$$

2. regression: predict foot size given height and weight

$$x \mapsto \phi(\cdot) \mapsto y \in R$$

- how do we compute $\phi(\cdot)$?

Computing Prediction

- choose a class of functions (e.g. linear, polynomial)
- choose hyper-parameters to:
  1. minimize training loss
  2. generalize to unseen data
- example: in linear regression we choose slope and intersect (hyper-parameters) to minimize sum of squared residuals

Loss Minimization

- compute parameters to minimize loss
- learning is optimization
- solution depends on the loss function
- examples of loss functions:

1. squared loss $\left(y - \phi(x)\right)^2$
2. absolute loss $|y - \phi(x)|$

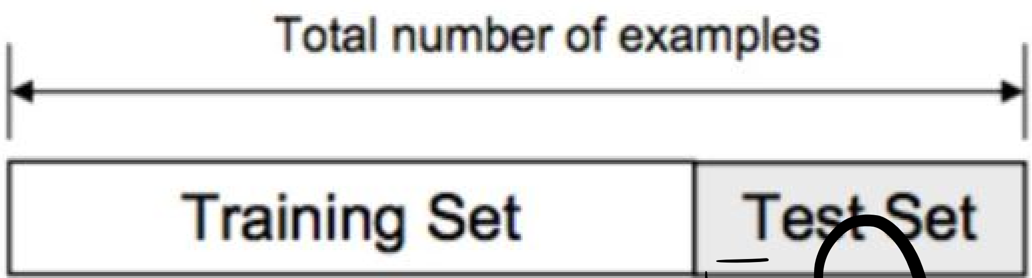
- use gradient descent to compute hyper-parameters

Testing/Training Sets

- Q: how do we compute model parameters?
- Q: how do we estimate accuracy?
- A: split known objects into two subsets

1. X_train: for parameters
2. X_testing: for accuracy

Testing/Training Sets (cont'd)



50 50

- need to minimize any biases in the data

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Bias in Training Data

```
import pandas as pd
data = pd.DataFrame(
      {"id": [ 1,2,3,4,5,6,7,8],
       "Label": ["green","green",
                "green","green",
                "red","red",
                "red","red"],
       "Height": [5, 5.5, 5.33,5.75,
                 6.00,5.92,5.58,5.92],
       "Weight": [100, 150, 130, 150,
                   180, 190, 170, 165],
       "Foot": [6, 8, 7, 9,
                13, 11, 12, 10]},
        columns = ["id", "Height",
                   "Weight", "Foot",
                   "Label"] )
X = data[["Height","Weight","Foot"]]
X_train = X[ : 4]
X_test  = X[ 4 : ]
```

Bias in Training Data (cont'd)

```
ipdb> X_train
0    5.00     100     6  green
1    5.50     150     8  green
2    5.33     130     7  green
3    5.75     150     9  green
ipdb> X_test
   Height  Weight  Foot Label
4    6.00     180    13   red
5    5.92     190    11   red
6    5.58     170    12   red
7    5.92     165    10   red
ipdb> X_test
```

- need to randomize data

Eliminating Bias

```
import pandas as pd
from sklearn.model_selection \
      import train_test_split

data = pd.DataFrame(
      {"id": [ 1,2,3,4,5,6,7,8],
       "Label":["green","green","green","green",
                "red","red","red","red"],
       "Height":[5, 5.5, 5.33,5.75,
                 6.00,5.92,5.58,5.92],
       "Weight":[100, 150, 130, 150,
                 180, 190, 170, 165],
       "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
        columns = ["id", "Height",
                   "Weight", "Foot","Label"] )

X = data[["Height","Weight","Foot"]]
y = data["Label"]

X_train,X_test,y_train,y_test=\
      train_test_split(X, y, train_size=0.5)
```

Eliminating Bias

```
ipdb> X_train
   Height  Weight  Foot  Label
3    5.75     150     9  green
1    5.50     150     8  green
4    6.00     180    13    red
5    5.92     190    11    red
ipdb> X_test
   Height  Weight  Foot  Label
2    5.33     130     7  green
0    5.00     100     6  green
7    5.92     165    10    red
6    5.58     170    12    red
ipdb> X_test
```

- data is now "shuffled"

Stratifying Classes

```
import pandas as pd
from sklearn.model_selection \
      import train_test_split

data = pd.DataFrame(
      {"id": [ 1,2,3,4,5,6,7,8],
       "Label":["green","green","green","green",
                "red","red","red","red"],
       "Height":[5, 5.5, 5.33,5.75,
                 6.00,5.92,5.58,5.92],
       "Weight":[100, 150, 130, 150,
                   180, 190, 170, 165],
       "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
        columns = ["id", "Height",
                   "Weight", "Foot","Label"] )

X = data[["Height","Weight","Foot"]]
y = data["Label"]

X_train,X_test,y_train,y_test=\
      train_test_split(X, y, train_size=0.75)
```

Stratifying Classes

```
ipdb> X_train
   Height  Weight  Foot  Label
6    5.58     170    12    red
2    5.33     130     7  green
3    5.75     150     9  green
0    5.00     100     6  green
1    5.50     150     8  green
4    6.00     180    13    red
ipdb> X_test
   Height  Weight  Foot Label
7    5.92     165    10   red
5    5.92     190    11   red
```

- label counts are not evenly distributed

Split and Stratify

```
import pandas as pd
from sklearn.model_selection \
      import train_test_split

data = pd.DataFrame(
      {"id": [ 1,2,3,4,5,6,7,8],
       "Label":["green","green","green","green",
                "red","red","red","red"],
       "Height":[5, 5.5, 5.33,5.75,
                 6.00,5.92,5.58,5.92],
       "Weight":[100, 150, 130, 150,
                  180, 190, 170, 165],
       "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
        columns = ["id", "Height",
                   "Weight", "Foot","Label"] )

X = data[["Height","Weight","Foot"]]
y = data["Label"]

X_train,X_test,y_train,y_test=\
      train_test_split(X, y,
          train_size=0.75,stratify=y)
```

Stratifying Classes

```
ipdb> X_train
1     5.50      150      8  green
6     5.58      170     12    red
3     5.75      150      9  green
5     5.92      190     11    red
2     5.33      130      7  green
7     5.92      165     10    red
ipdb> X_test
   Height  Weight  Foot  Label
0     5.0     100     6  green
4     6.0     180    13    red
```

- label counts are now evenly distributed

Cross Validation

- we use only a portion of data for testing and training
- can use more with n-fold cross validation

1. split data randomly into n parts
2. use $n - 1$ for training
3. use 1 part for testing
4. repeat n times using different part for testing
5. average results

k-fold Validation

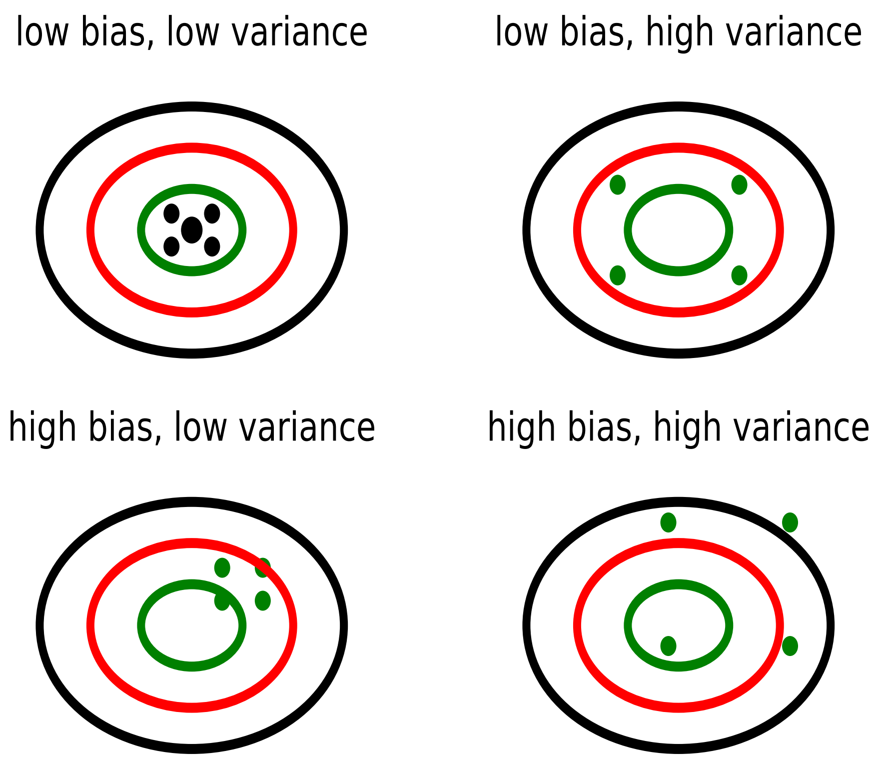
- expensive for large n

Bias-Variance Decomposition

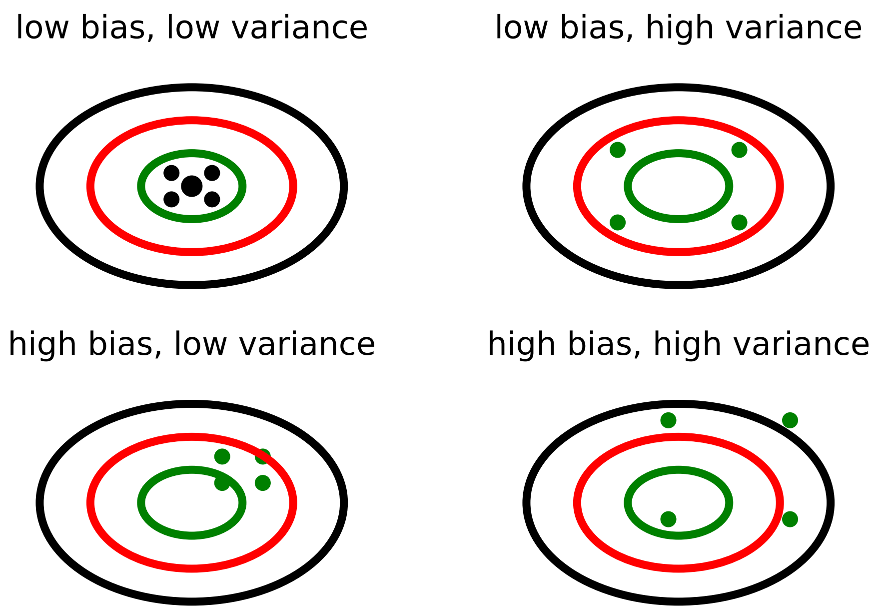
- bias - average difference between prediction and correct value
- variance - variability of prediction for a given point

$$\begin{aligned}\textbf{Error} = &\ \textbf{Bias}^2 \\ &+ \textbf{Variance} \\ &+ \textbf{Irreducible Error}\end{aligned}$$

Bias-Variance Trade-Off

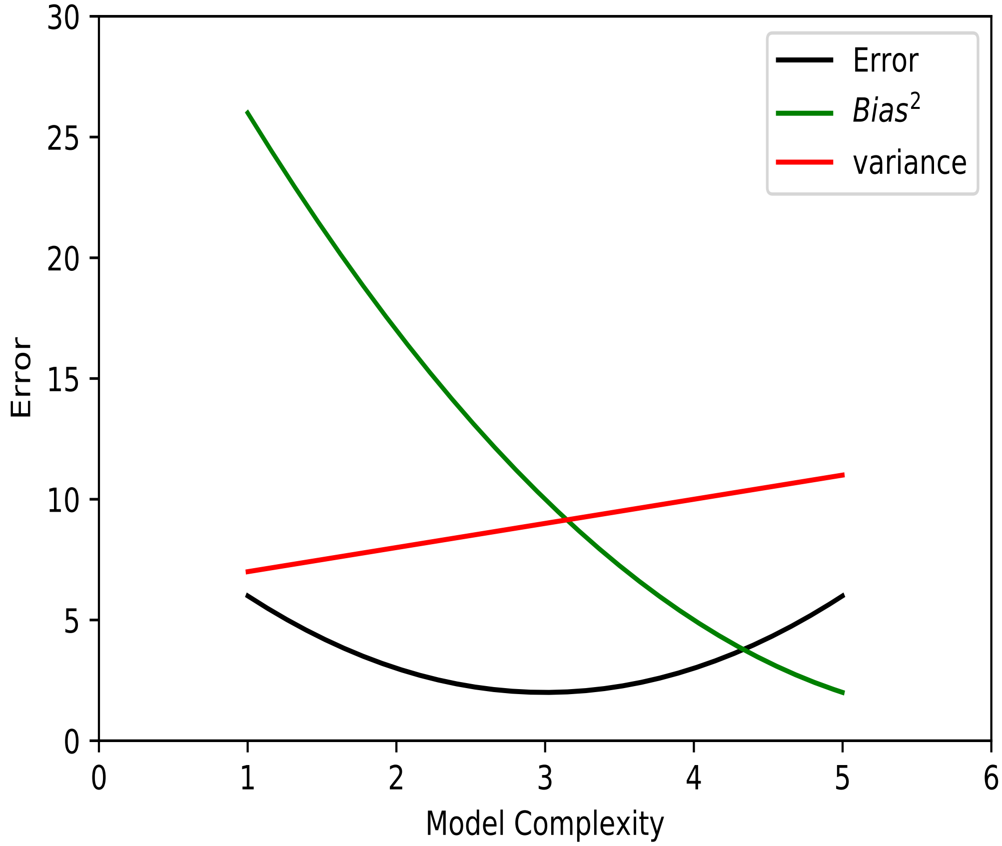


Bias-Variance Trade-Off



- ”overfitting” - low bias, high variance
- ’underfitting” - low variance, high bias

Bias-Variance: Ideal Model



Concepts Check:

(a) prediction vs. classification

(b) numerical vs. categorical data

(c) loss function

(d) testing and training data

(e) bias elimination and stratification

(f) cross and k-fold validation

(g) bias vs. variance trade-offs