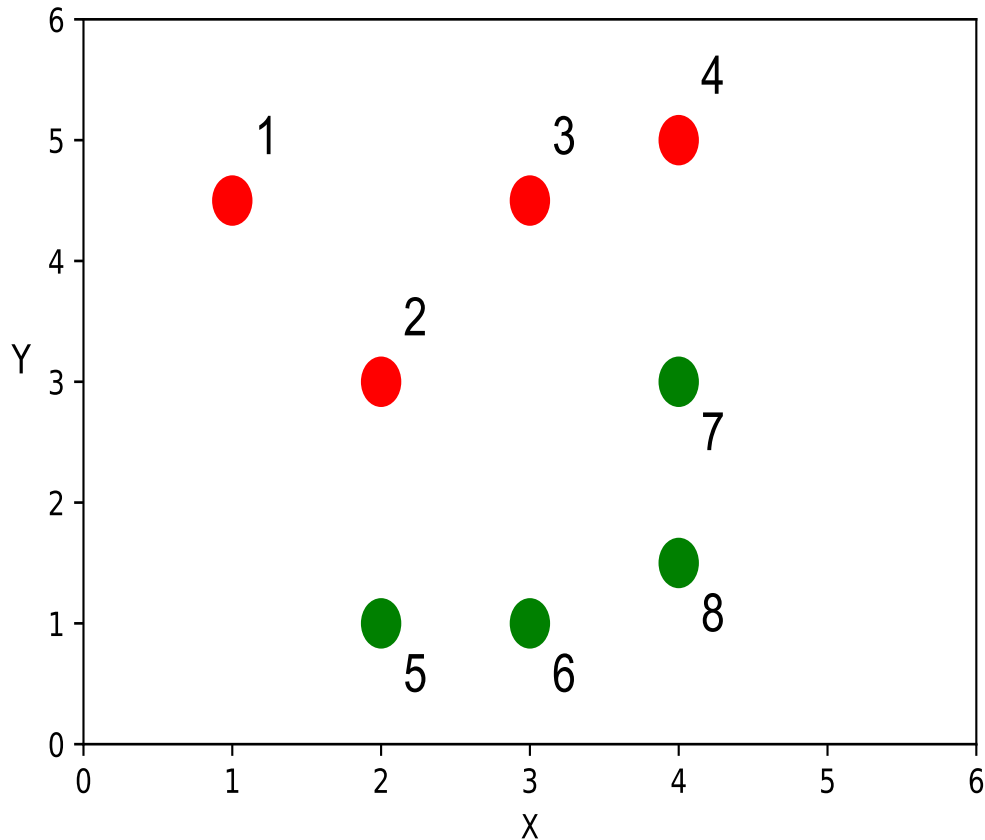


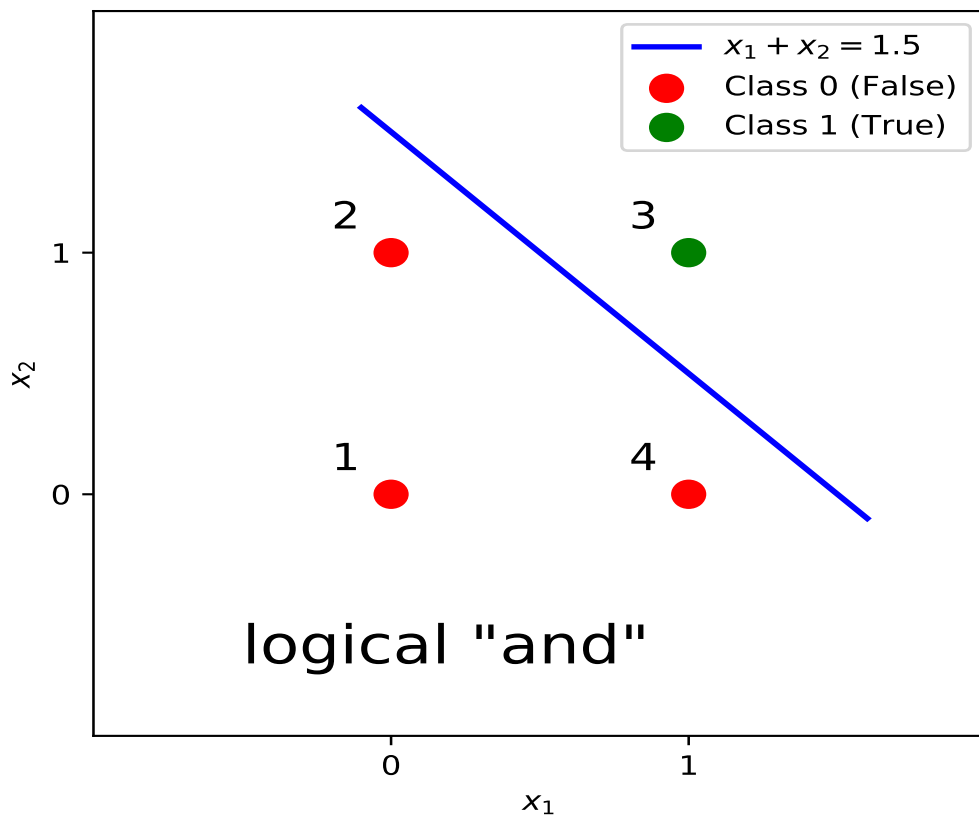
LOGISTIC REGRESSION

Overview



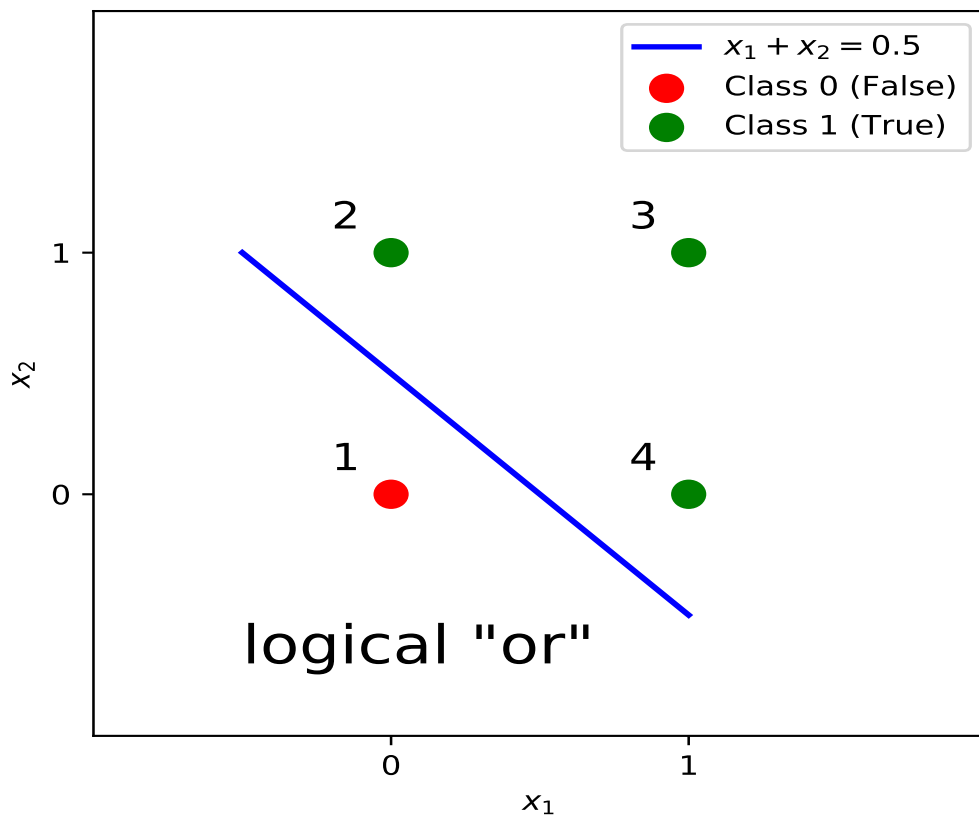
- want to separate classes
- smooth decision function

Example: "AND" Gate



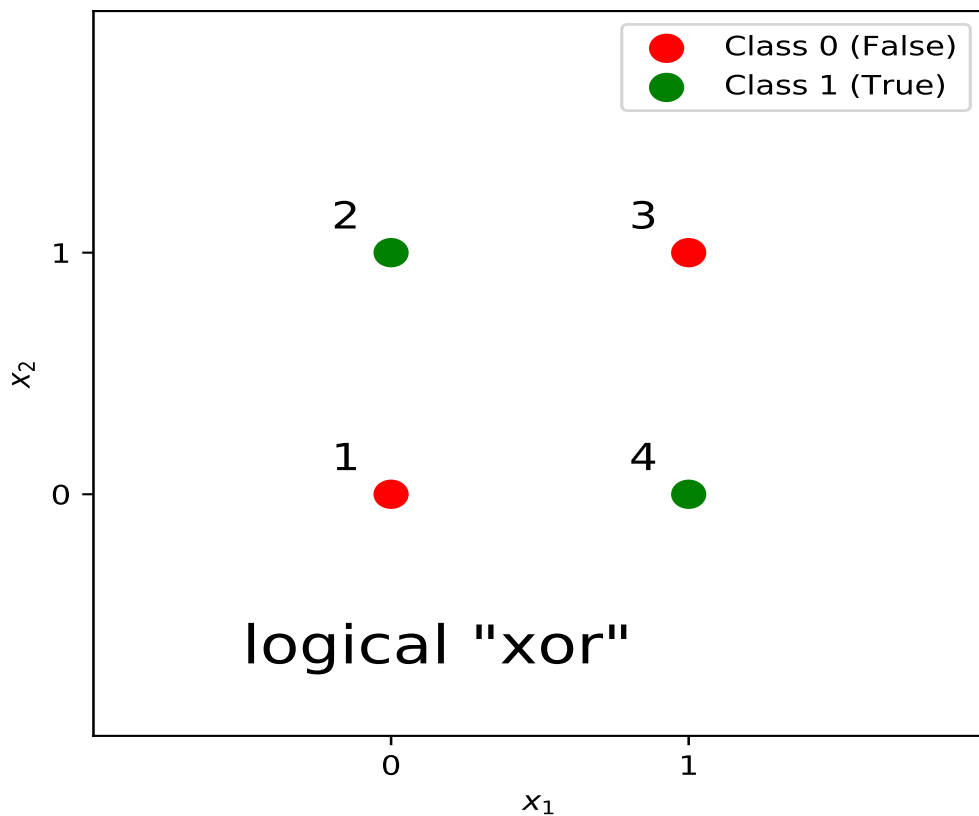
- linearly separable

Example: "OR" Gate



- linearly separable

Example: "XOR" Gate



- not linearly separable

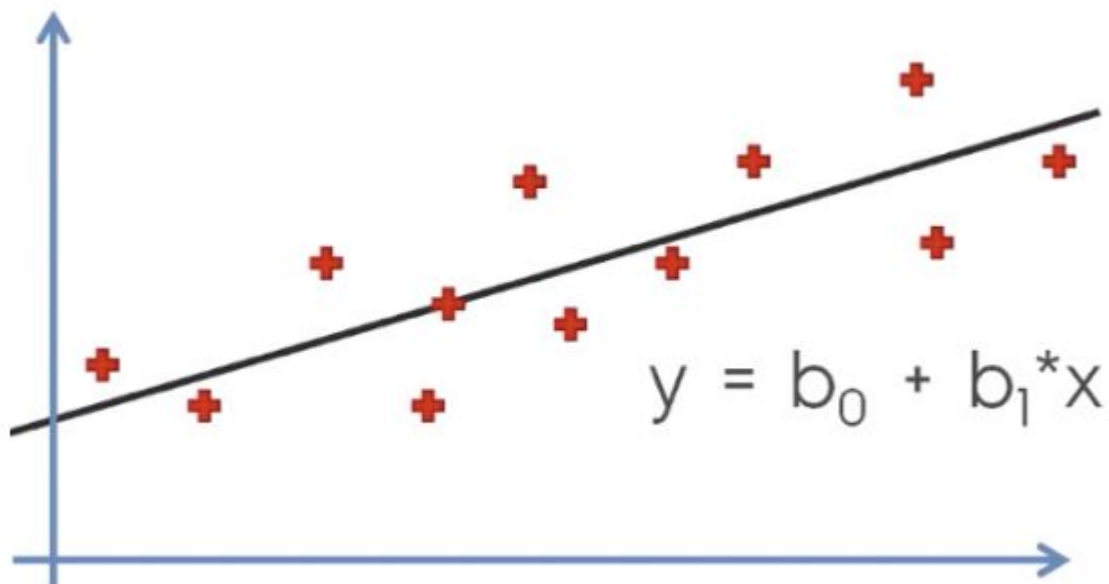
Binary Classification

- training set s with labels $\{0, 1\}$
- find a classifier H

$$H : X \mapsto \{0, 1\}$$

- low generalization error
- linear classification (based on logistic regression)
- dividing (hyper)plane is called linear discriminant

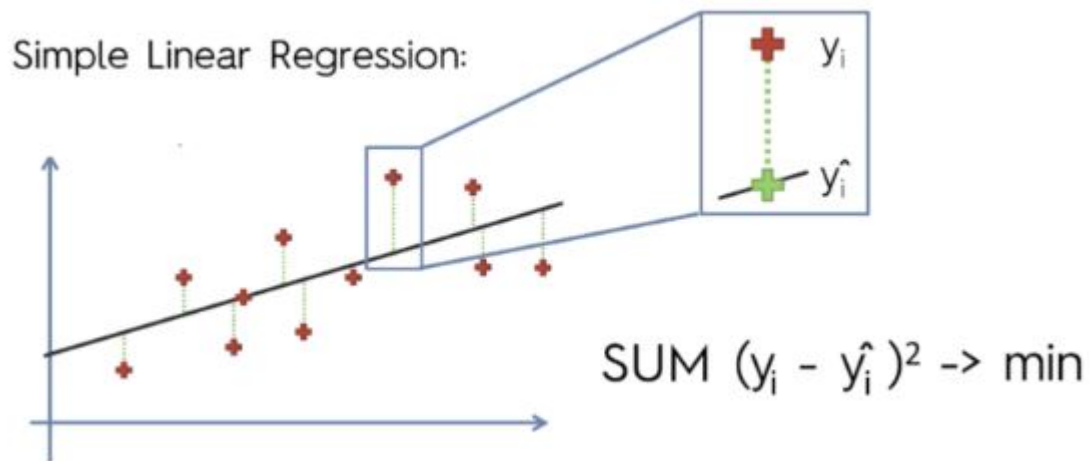
Background: Linear Regression



- example of a Generalized Linear Model (GLM)

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

Simple Linear Regression



- choose line to minimize loss

$$\text{Loss} = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Classification Problem



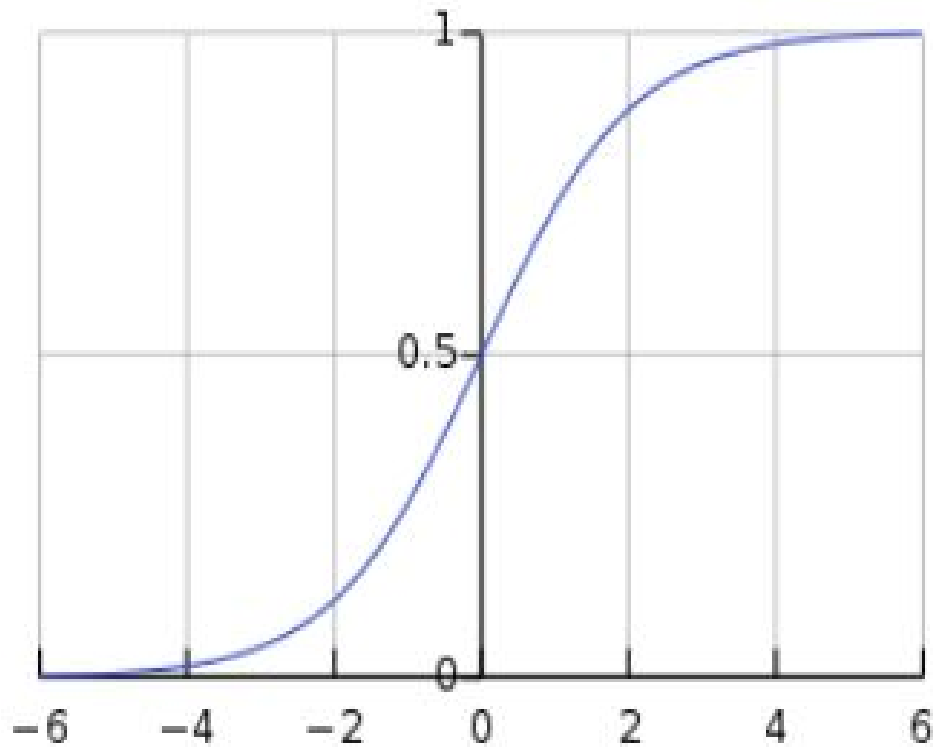
- how do we transform a linear prediction model to classification problem?

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

Issues

- linear regression: continuous variables
- classification: discrete
- probabilities must be in $[0, 1]$
- solution:
linear regression \mapsto classification
- how: use *logit* function

logit Function



$$\frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)}$$

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

Probability and *Odds*

- assume probability P
- define *odds* as

$$\text{odds} = \frac{P}{1 - P}$$

- ex.1: $P = 0.25 \mapsto \text{odds} = 1/3$
- ex.2: $P = 0.50 \mapsto \text{odds} = 1$
- ex.3: $P = 0.75 \mapsto \text{odds} = 3/1$

Main Idea:

- estimate $\text{logit}(\text{odds})$
- use regression

$$\log\left(\frac{P}{1-P}\right) = b_0 + b_1x$$

$$\frac{P}{1-P} = \exp(b_0 + b_1x)$$

$$P = \frac{\exp(b_0 + b_1x)}{1 + \exp(b_0 + b_1x)}$$

- note:

$$\frac{\exp(b_0 + b_1x)}{1 + \exp(b_0 + b_1x)} = \frac{1}{1 + \exp(-(b_0 + b_1x))}$$

Illustration

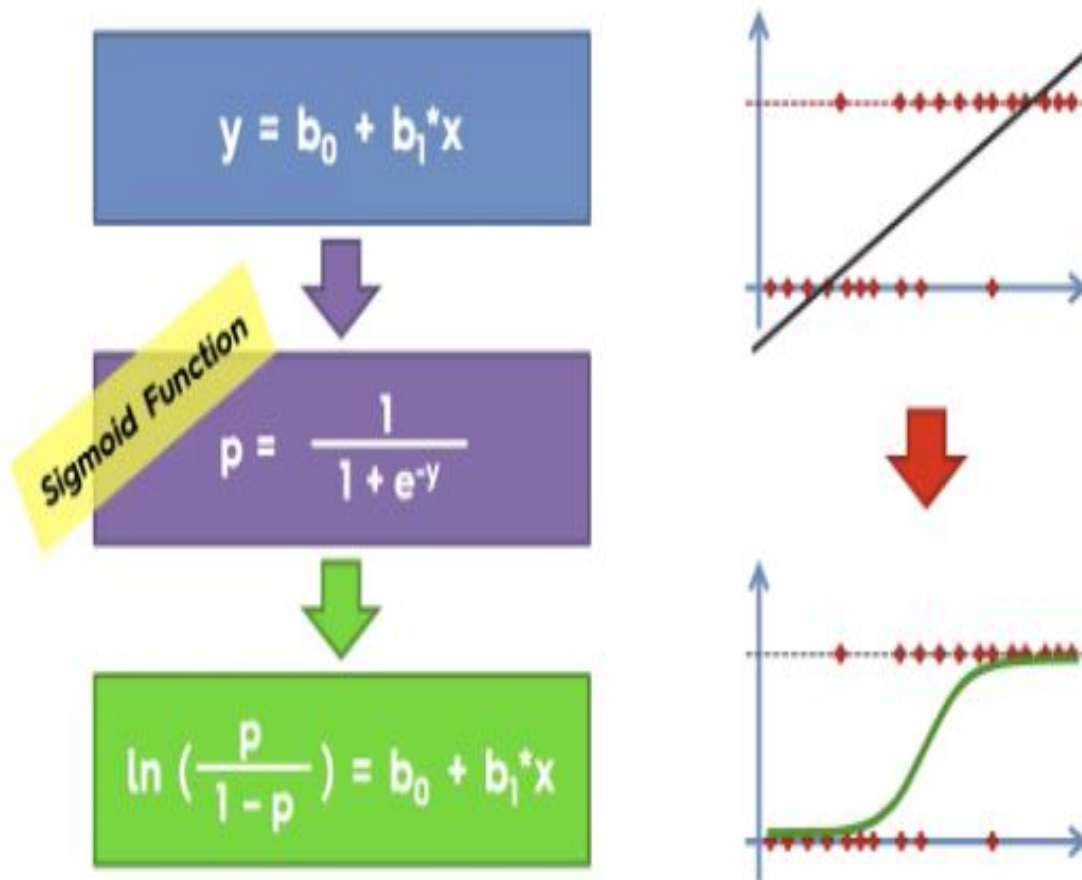
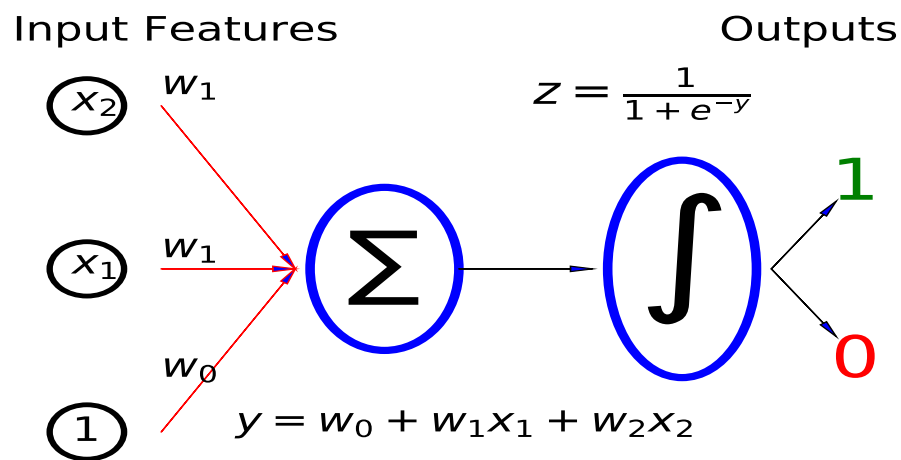


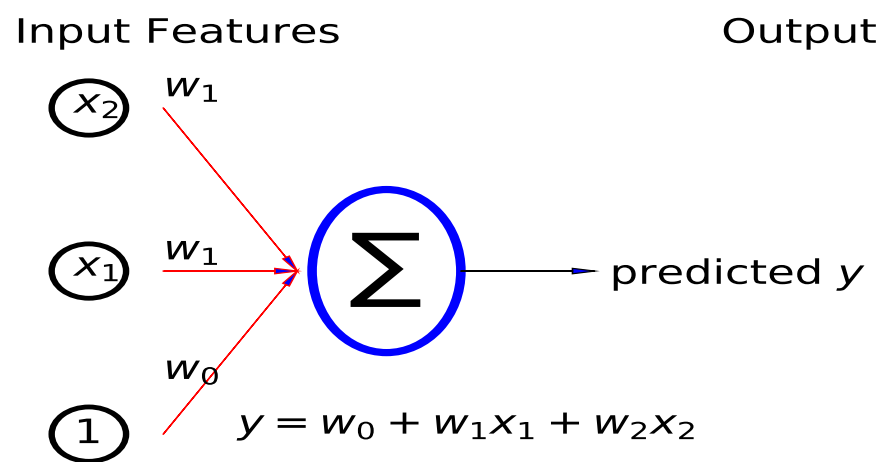
figure reprinted from www.kdnuggets.com with explicit permission of the editor

Logistic Regression



- supervised learning
- estimate label probabilities by sigmoid function

Linear Regression

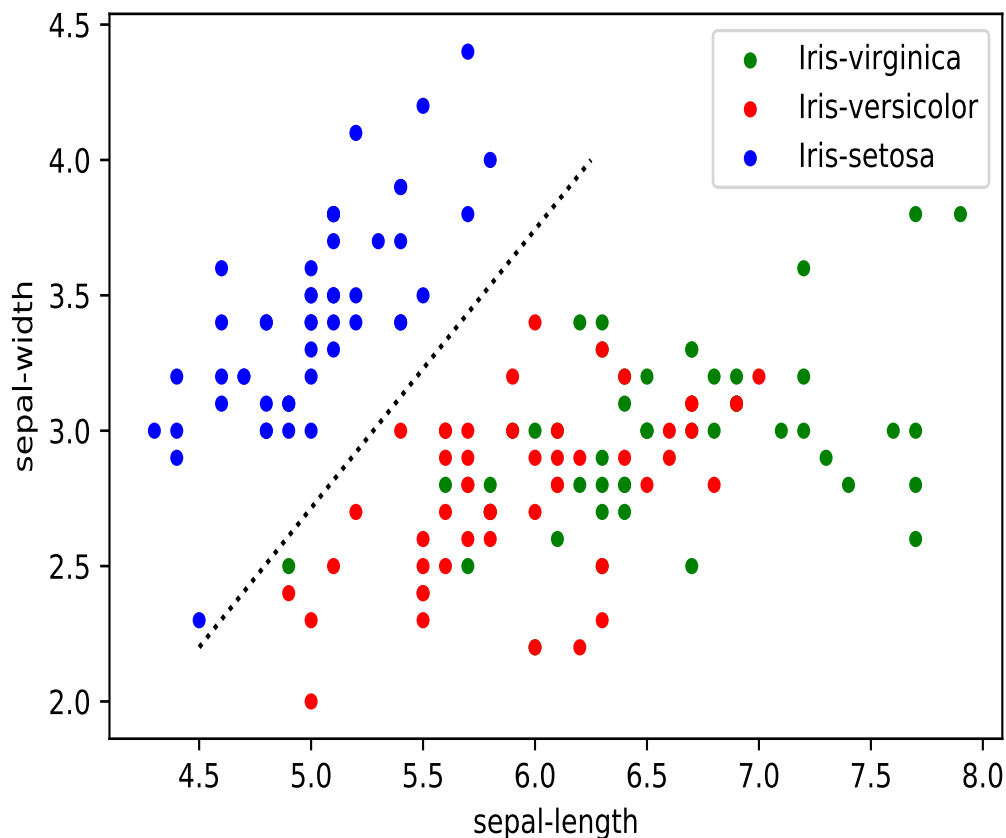


- real-valued output from weighted sum of inputs

Linear vs. Logistic

- linear regression:
 1. estimate w_0, w_1, \dots, w_n using min squared error
 2. predict $y = w_0 + w_1x_x + \dots + w_nx_n$
- logistic regression:
 1. estimate w_0, w_1, \dots, w_n using min squared error
 2. compute $y = w_0 + w_1x_x + \dots + w_nx_n$
 3. apply the sigmoid function $z(y)$ to compute label probabilities

Linear Separability



- draw a hyperplane
- difficult in many cases

Logistic Regression

- dependent variable is class label - categorical
- output is a weighted sum of inputs

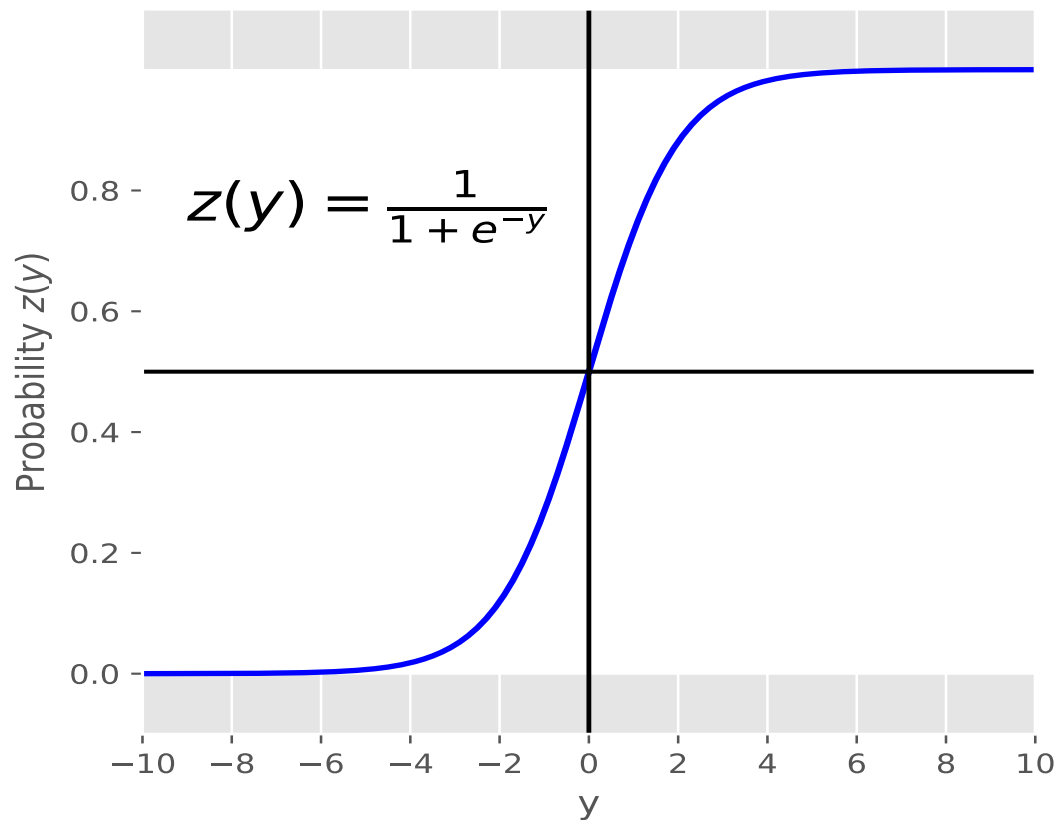
$$y = w_0 + w_1x_1 + \dots + w_mx_m$$

- weighted sum is passed through a sigmoid function

$$z(y) = \frac{1}{1 + e^{-y}}$$

- assign labels based on $z(y)$

Sigmoid Function $z(y)$



- $z(y) > 0.5$ if $y > 0$ (class 1)
- $z(y) < 0.5$ if $y < 0$ (class 0)

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 |

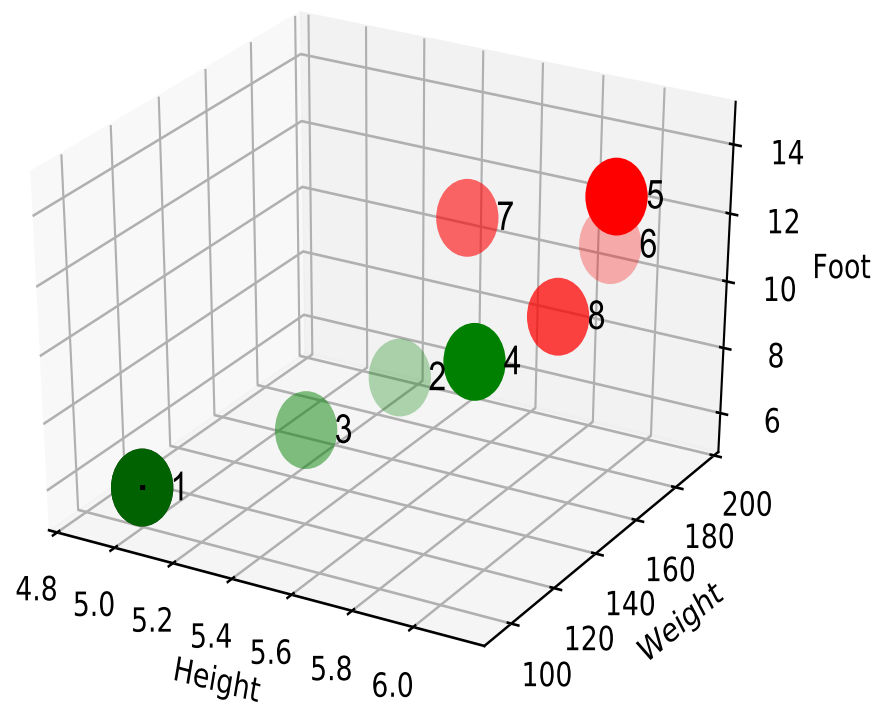
Code for the Dataset

```
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'] )
```

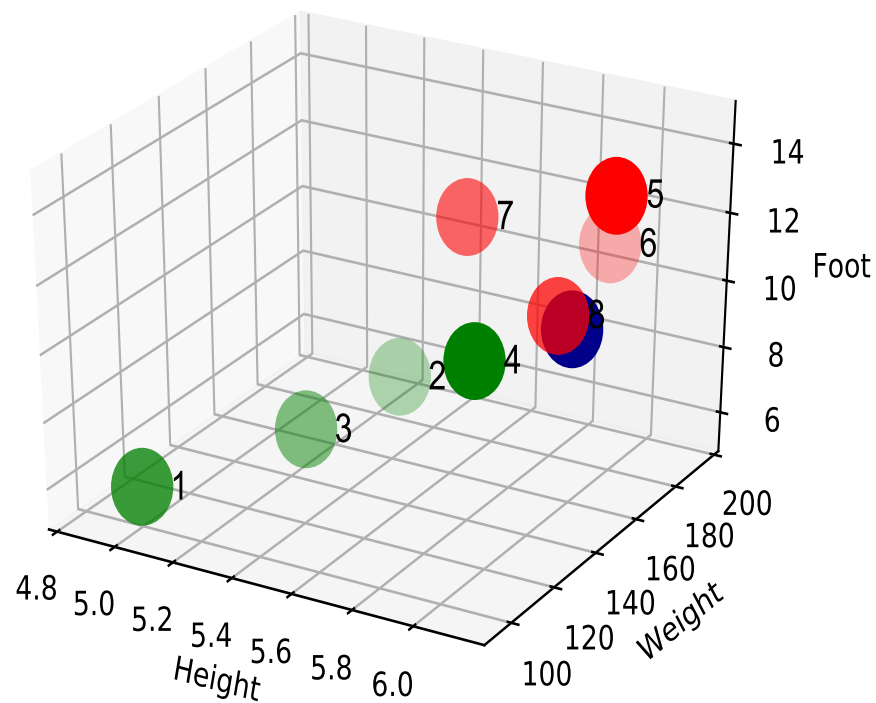
```
ipdb> data
```

| | 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

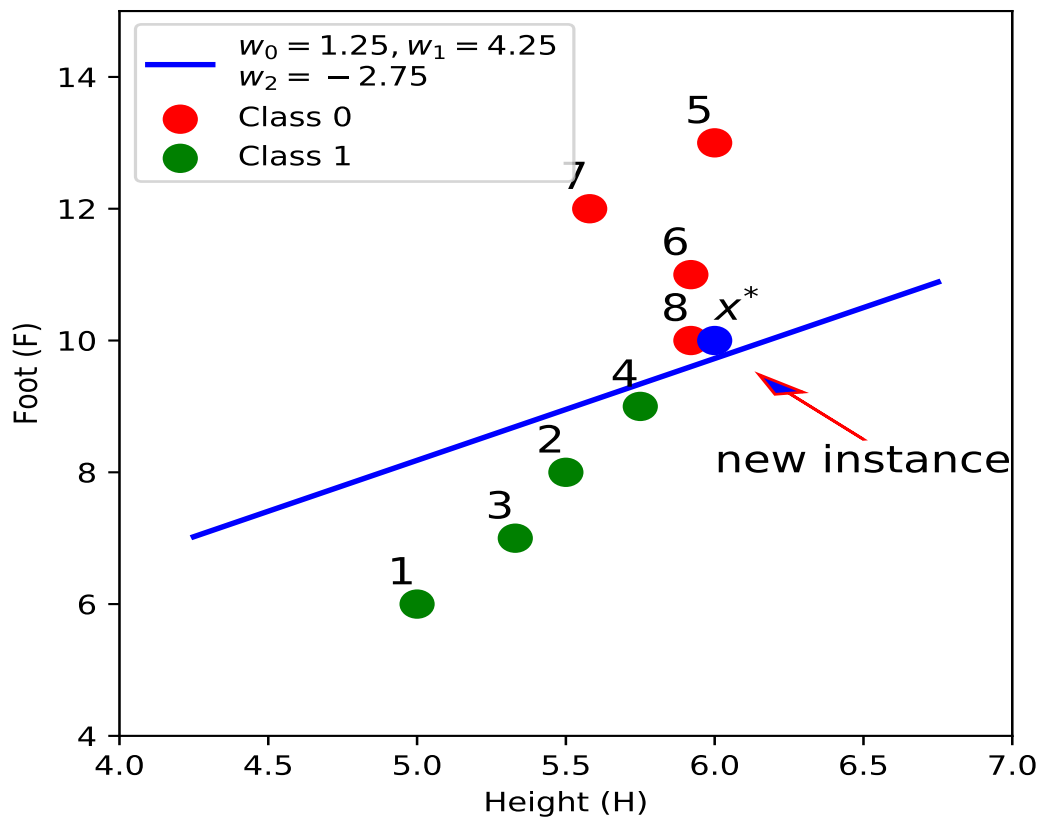


A New Instance

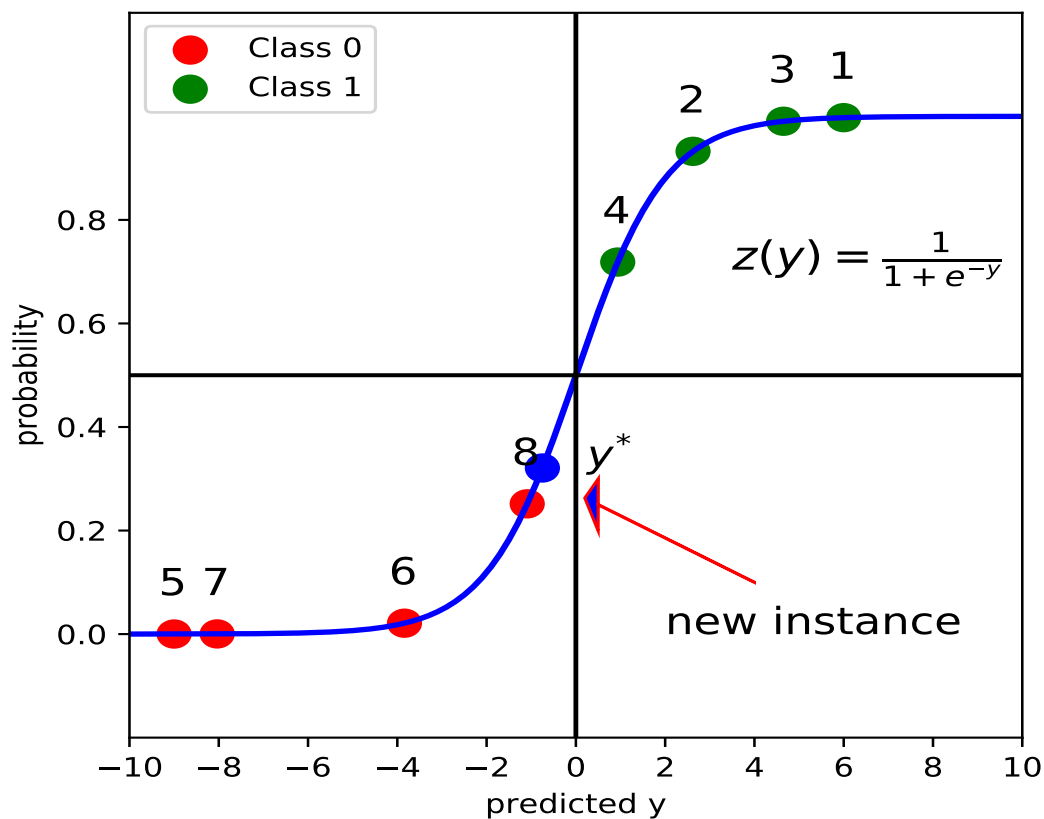


$(H=6, W=160, F=10) \mapsto ?$

Separability in Detail



Computing Class Labels



- $z(y^*) < 0.5$ - "red" (class 0)

Summary of Logistic Regression

- feature vector: $X = (1, x_1, \dots, x_m)$
- weights $W = (w_0, w_1, \dots, w_m)$
- compute

$$y = W \cdot X = w_0 + w_1x_1 + \dots + w_mx_m$$

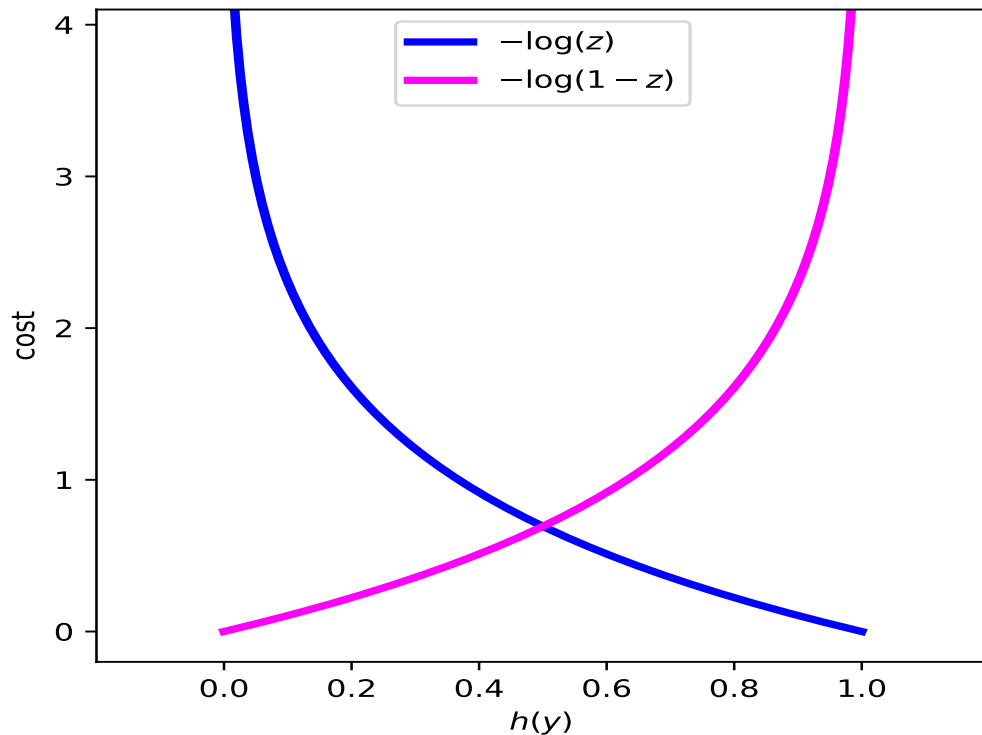
- compute probability $h(x)$:

$$h(x) = \frac{1}{1 + e^{-W \cdot X}}$$

- assign label $C(X)$:

$$C(X) = \begin{cases} 1, & \text{if } h(x) > 0.5 \\ 0, & \text{if } h(x) < 0.5 \end{cases}$$

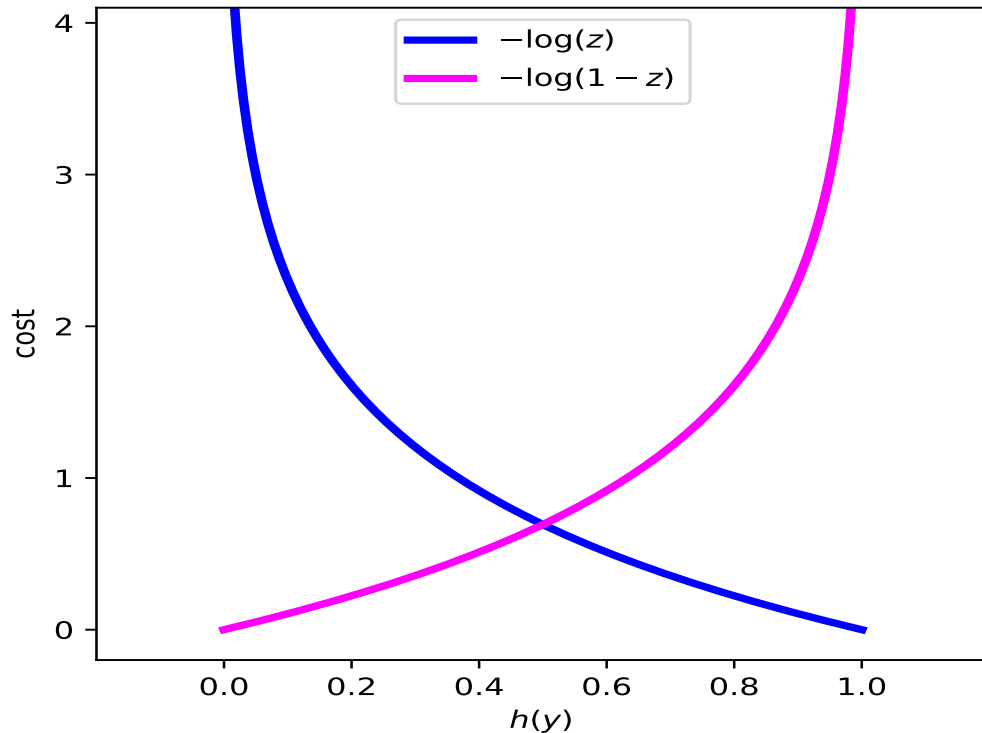
How to Compute w ?



- maximize likelihood

$$L = \prod_X h(X)^{C(X)} \cdot [1 - h(X)]^{1-C(X)}$$

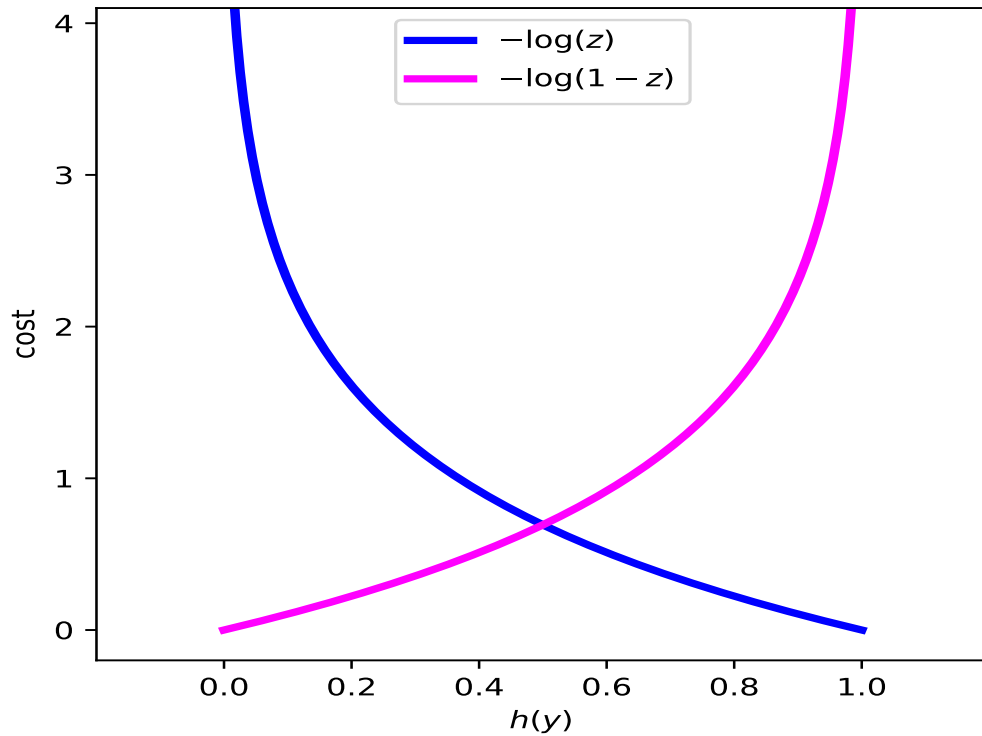
Cost Function



- minimize cost (“loss”):

$$Q = -\sum_X [C(X) \log(h(X)) + (1 - C) \log(1 - h(X))]$$

Cost Intuition



- correct classification cost: 0
- misclassification cost: $\mapsto \infty$

Computing Gradient

- gradient (with respect to w_i):

$$\frac{\partial Q}{\partial w_i} = \sum_X [h(X) - C(X)] \cdot x_i$$

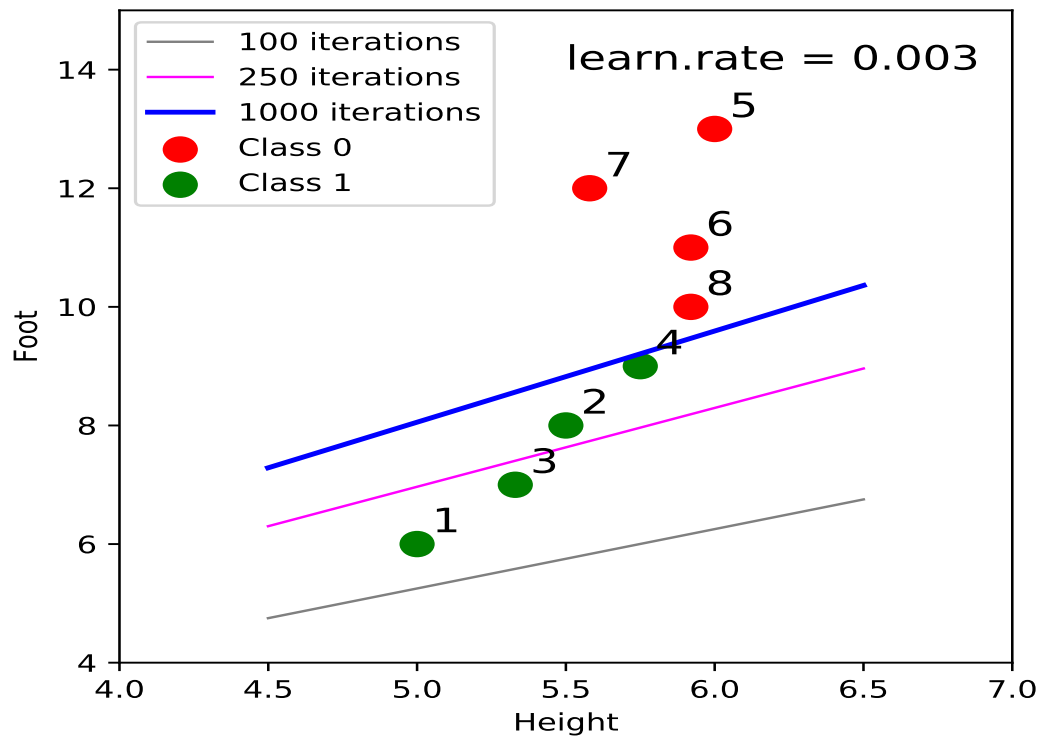
- computation of weights:

1. initialize weights
2. (simultaneously) update

$$w_i = w_i - \alpha \sum_X [h(X) - C(X)] x_i$$

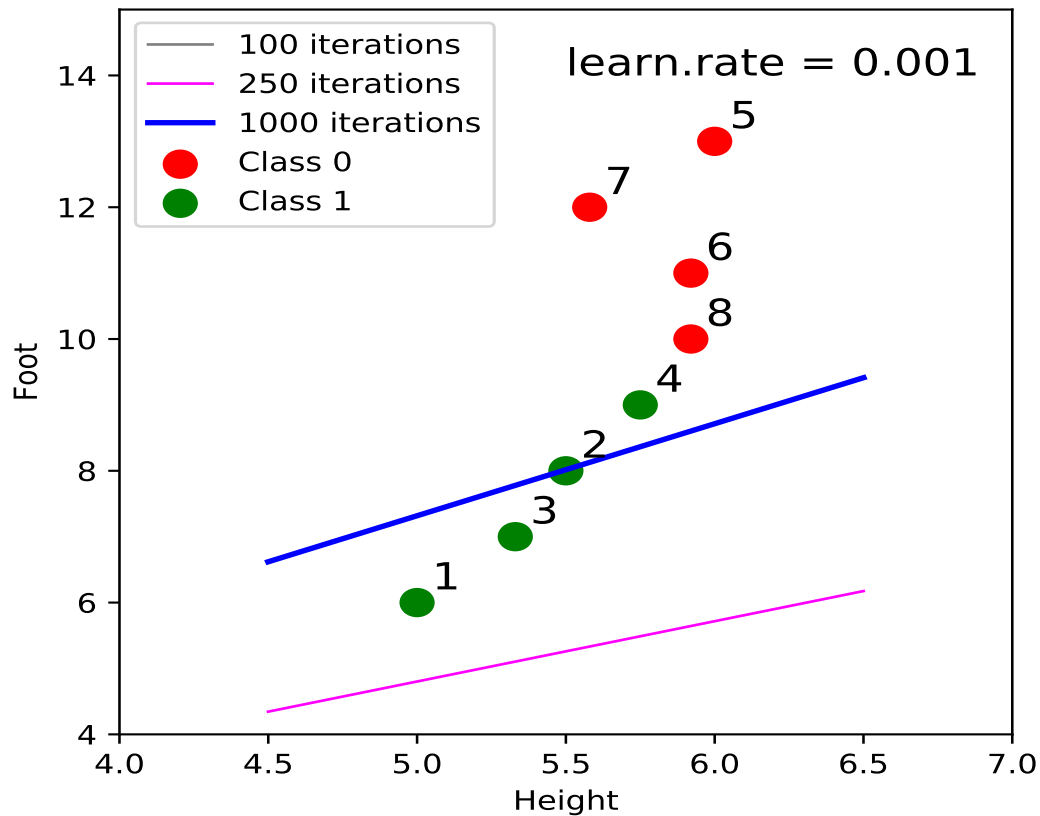
3. α is the learning rate

Computing Weights



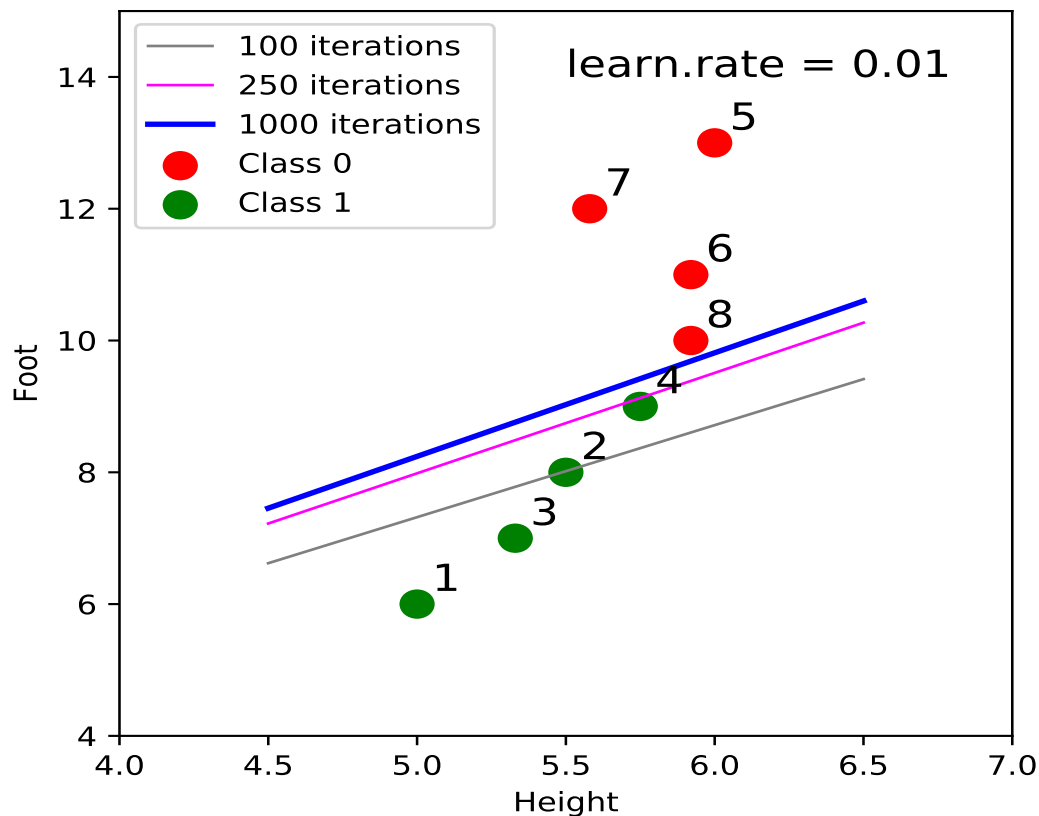
| iterations | w_0 | w_1 | w_2 | accuracy |
|------------|-------|-------|--------|----------|
| 100 | 0.021 | 0.084 | -0.084 | 50% |
| 250 | 0.053 | 0.221 | -0.166 | 75% |
| 1000 | 0.177 | 0.741 | -0.482 | 100% |

Effect of Lower Rate



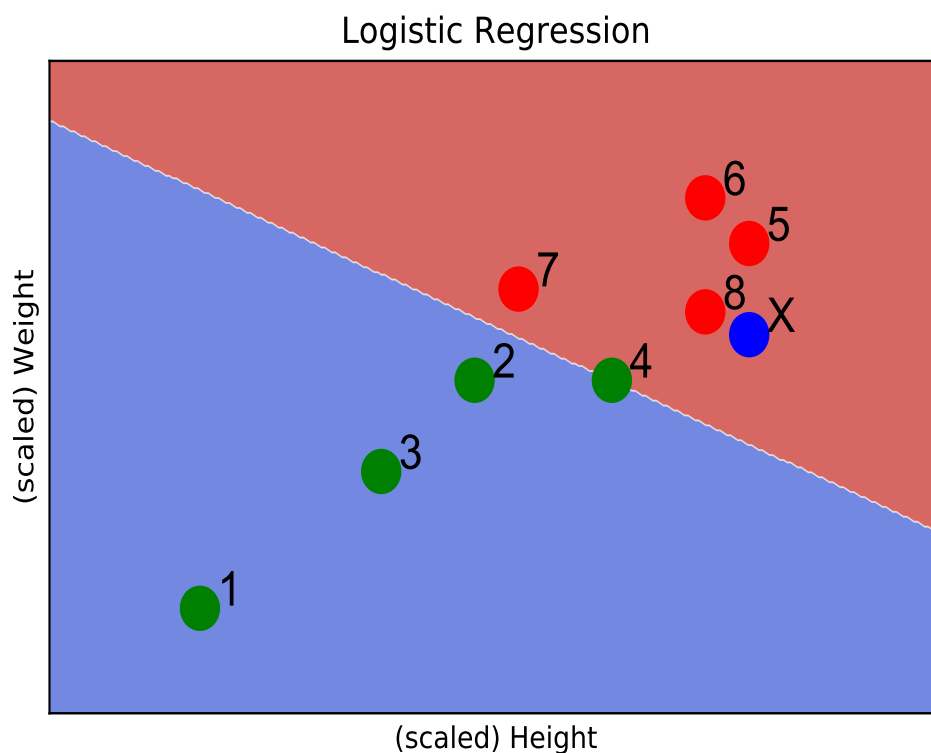
- need more iterations

Effect of Higher Rate



- get higher accuracy for the same number of iterations

Logistic Regression (original dataset)



- $\text{predict}(x^*) = \text{red}$
- $\text{accuracy} = 100\%$

Code: Log. Regression

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder

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'] )

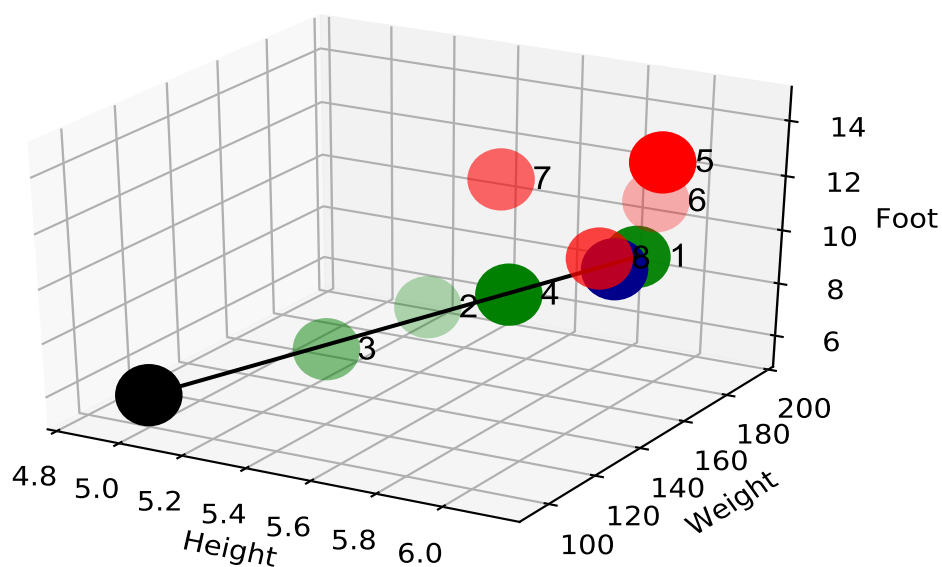
X = data[['Height', 'Weight']].values
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X,Y)

new_x = scaler.transform(np.asmatrix([6, 160]))
predicted = log_reg_classifier.predict(new_x)
accuracy = log_reg_classifier.score(X, Y)

ipdb> predicted[0]
red
ipdb> accuracy
0.875
```

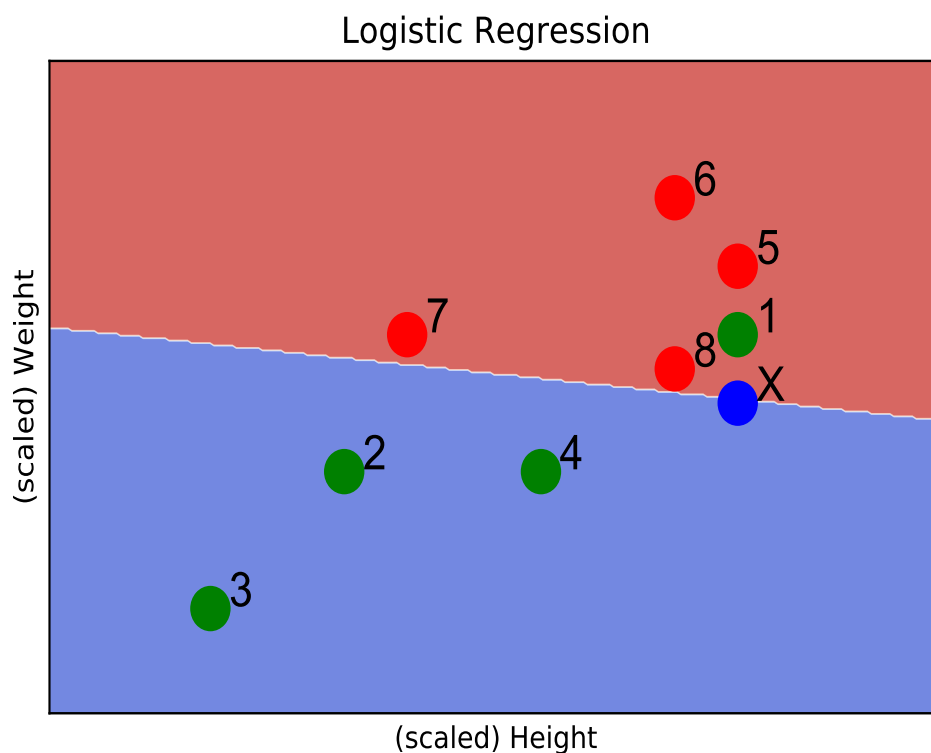
F/W/H Change



| id | Height | Weight | Foot | Label |
|----|---------------|-------------------|----------------|-------|
| 1 | 5 \mapsto 6 | 100 \mapsto 170 | 6 \mapsto 10 | green |

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

Logistic Regression (modified dataset)



- $\text{predict}(x^*) = \text{green}$
- $\text{accuracy} = 87.5\%$

Code: Log. Regression (modified dataset)

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder

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'] )

data['Height'].iloc[0] = 6;
data['Weight'].iloc[0] = 170;
data['Foot'].iloc[0] = 10
X = data[['Height', 'Weight']].values
scaler = StandardScaler().fit(X)

X = scaler.transform(X)
Y = data['Label'].values
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X,Y)
new_x = scaler.transform(np.asmatrix([6, 160]))
predicted = log_reg_classifier.predict(new_x)
accuracy = log_reg_classifier.score(X, Y)

ipdb> predicted[0]
green
ipdb> accuracy
0.875
```

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 |

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

Change to Dummy Variables

| Day | Weather | | | Temp. | | | Wind | |
|-----|----------|-------|-------|-------|-----|------|------|-----|
| | overcast | rainy | sunny | cold | hot | mild | high | low |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 7 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 9 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

Python Code

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder

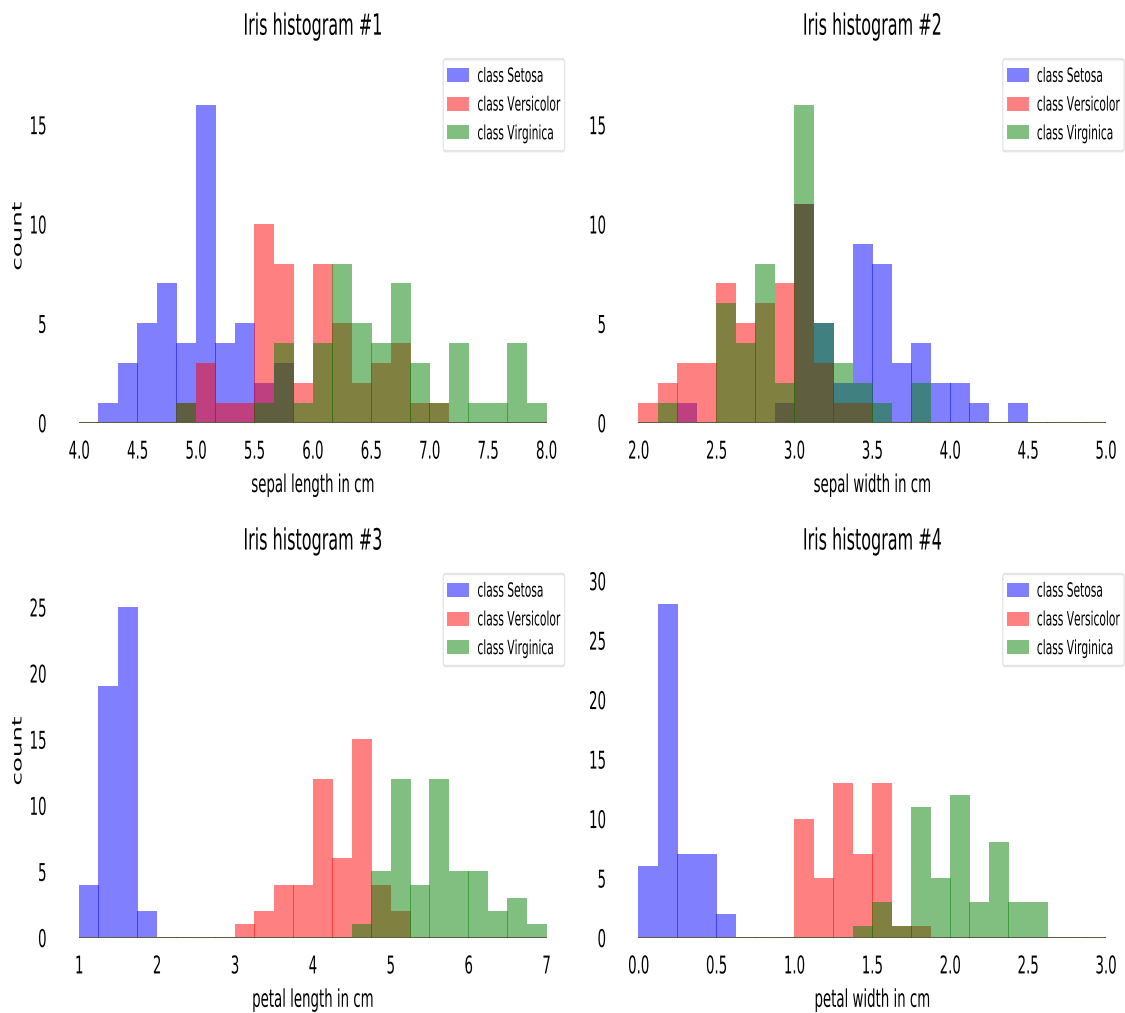
data = pd.DataFrame(
    {'Day': [1,2,3,4,5,6,7,8,9,10],
     'Weather': ['sunny','rainy','sunny','rainy',
                 'sunny','overcast','sunny','overcast',
                 'rainy','rainy'],
     'Temperature': ['hot', 'mild', 'cold','cold','cold',
                    'mild','hot','hot', 'hot','mild'],
     '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'])

input_data = data[['Weather', 'Temperature', 'Wind']]
dummies = [pd.get_dummies(data[c]) for c in input_data.columns]
binary_data = pd.concat(dummies, axis=1)
X = binary_data[0:10].values
le = LabelEncoder()
Y = le.fit_transform(data['Play'].values)
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X,Y)

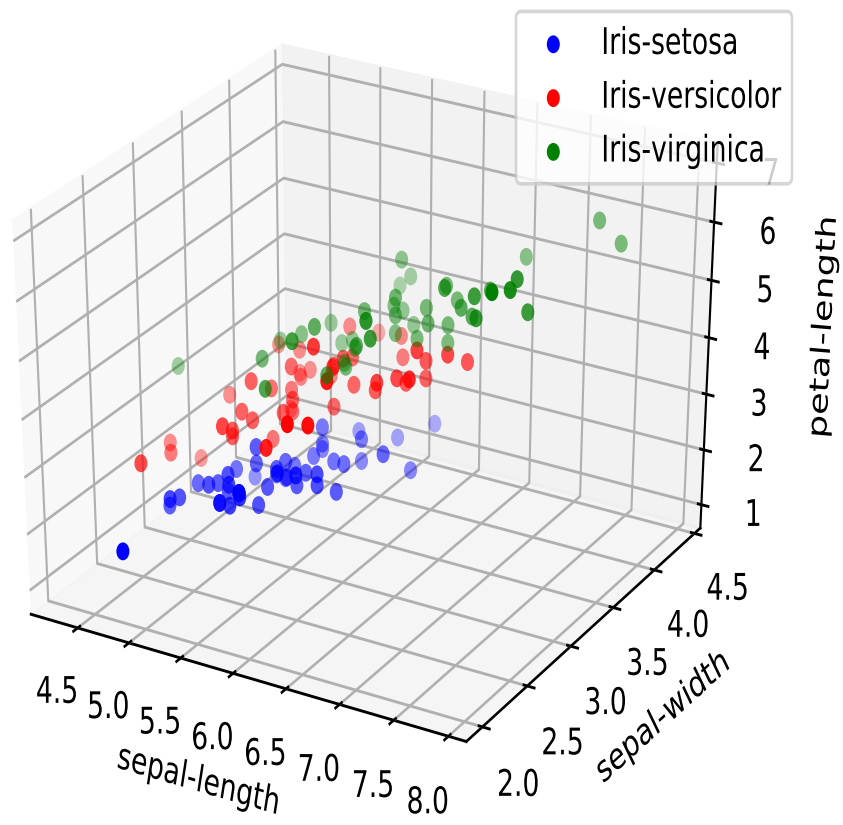
# sunny -> (0,0,1), cold-> (0,1,0), low -> (0,1)
new_instance = np.asmatrix([0,0,1,1,0,0,0,1])
prediction = log_reg_classifier.predict(new_instance)
accuracy = log_reg_classifier.score(X, Y)

ipdb> prediction[0]
1
ipdb> accuracy
0.8
```

Iris Histograms



Iris Dataset:



Iris: Python Code

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

url = r'https://archive.ics.uci.edu/ml/' + \
      r'machine-learning-databases/iris/iris.data'

data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                               'petal-length', 'petal-width', 'Class'])

features = ['sepal-length', 'sepal-width']
class_labels = ['Iris-setosa', 'Iris-versicolor']

X = data[features].values

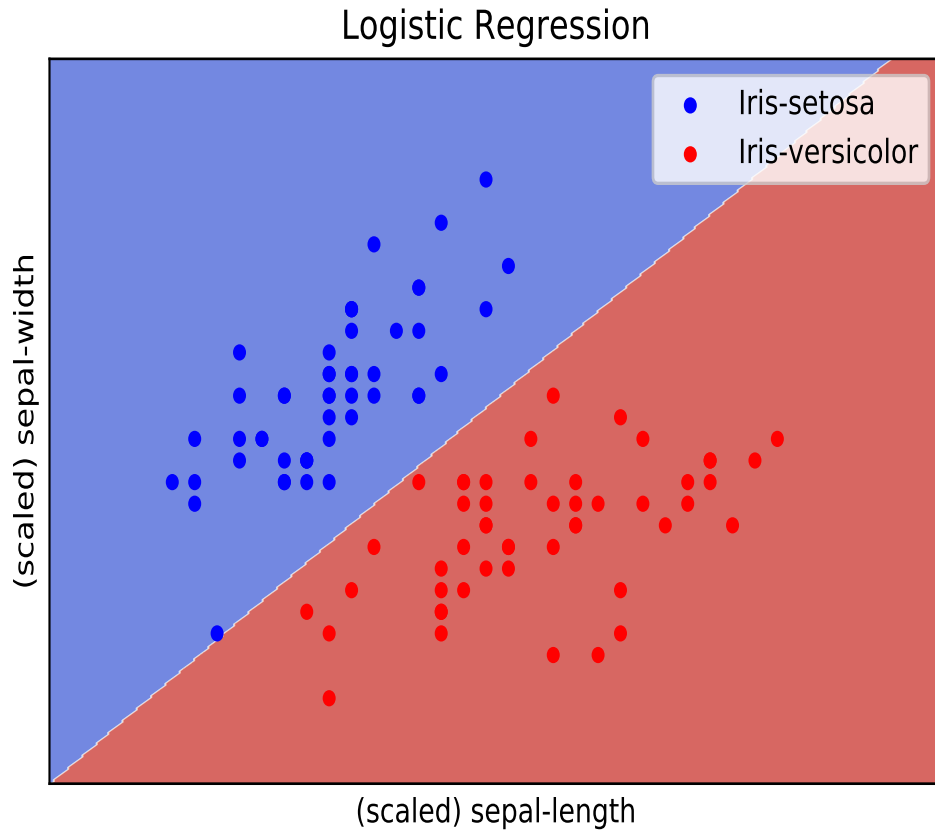
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.5, random_state=3)
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X_train, Y_train)

prediction = log_reg_classifier.predict(X_test)
accuracy = np.mean(prediction == Y_test)

ipdb> accuracy
1.0
```

Iris: Logistic Regression



- accuracy = 100%
- easy to separate

Iris: Python Code

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

url = r'https://archive.ics.uci.edu/ml/' + \
      r'machine-learning-databases/iris/iris.data'

data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                              'petal-length', 'petal-width', 'Class'])

features = ['sepal-length', 'sepal-width']
class_labels = ['Iris-versicolor', 'Iris-virginica']

X = data[features].values

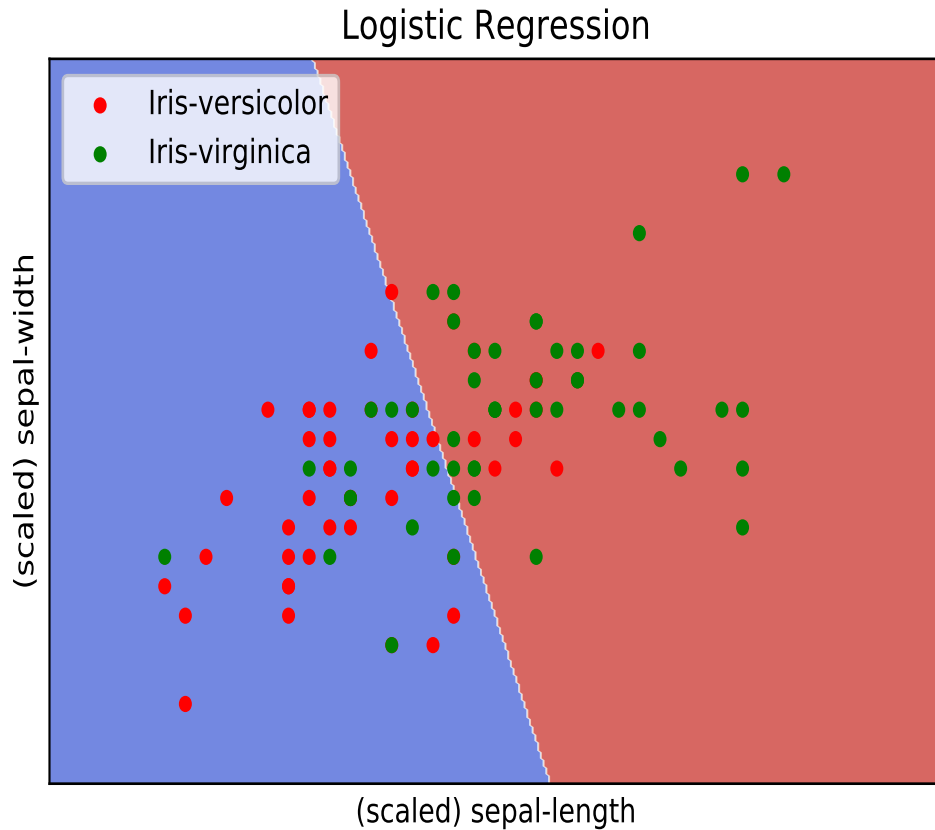
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.5, random_state=3)
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X_train, Y_train)

prediction = log_reg_classifier.predict(X_test)
accuracy = np.mean(prediction == Y_test)

ipdb> accuracy
0.68
```

Iris: Logistic Regression



- accuracy = 68%
- difficult to separate

Concepts Check:

- (a) linear separability
- (b) logistic vs. linear regression
- (c) odds and logit function
- (d) computing weights
- (e) analysis of categorical data