LINEAR

AND

QUADRATIC

DISCRIMINANT

Overview

- statistical technique to classify categorical outcomes
- assumes data is normally distributed
- two variations:
  1. same covariances - "linear" discriminant
  2. different covariances - quadratic discriminant

Problem Statement

- training set X and classes
- want to assign class y to x^*

$$y = \text{argmax}_y P\big(Y = y | X = x^*\big)$$

- instead of estimating $P\big(Y|X\big)$ estimate $P\big(X|Y\big)$ and $P\big(Y\big)$
- by Bayes formula

$$P(Y = k|X = x^*) = \frac{P(X = x^*|Y = k)P(Y = k)}{P(X = x^*)}$$

Analysis

- estimate $P\big(Y = k\big) = \pi_k$ by the fraction of training samples of class k
- $f_k\big(x\big) = P\big(X = x^*|Y = k\big)$ assume normally distributed

$$f_k(x) = \frac{1}{(2\pi)^{p/2}\sqrt{|\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right]$$

- assume different μ but same covariance Σ
- by Bayes rule:

$$P\big(Y = k|X = x^*\big) = \frac{f_k\big(x\big)\pi_k}{P\big(X = x^*\big)}$$

Analysis (cont'd)

- take logarithms and ignoring constants, we want to maximize

$$\delta_k(x) = \log \pi_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + x^T \Sigma^{-1} \mu_k$$

- decision boundary $\delta_l(x) = \delta_k(x)$:

$$\log \pi_l - \frac{1}{2}\mu_l^T \Sigma^{-1} \mu_l + x^T \Sigma^{-1} \mu_l$$
$$= \log \pi_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + x^T \Sigma^{-1} \mu_k$$

- linear in x

Quadratic Discriminant

- assume each class has its own covariance matrix Σ_k
- want to maximize

$$\delta_k(x) = \log \pi_k - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}x^T\Sigma_k^{-1}x$$

- objective is quadratic in x

A Numerical Dataset

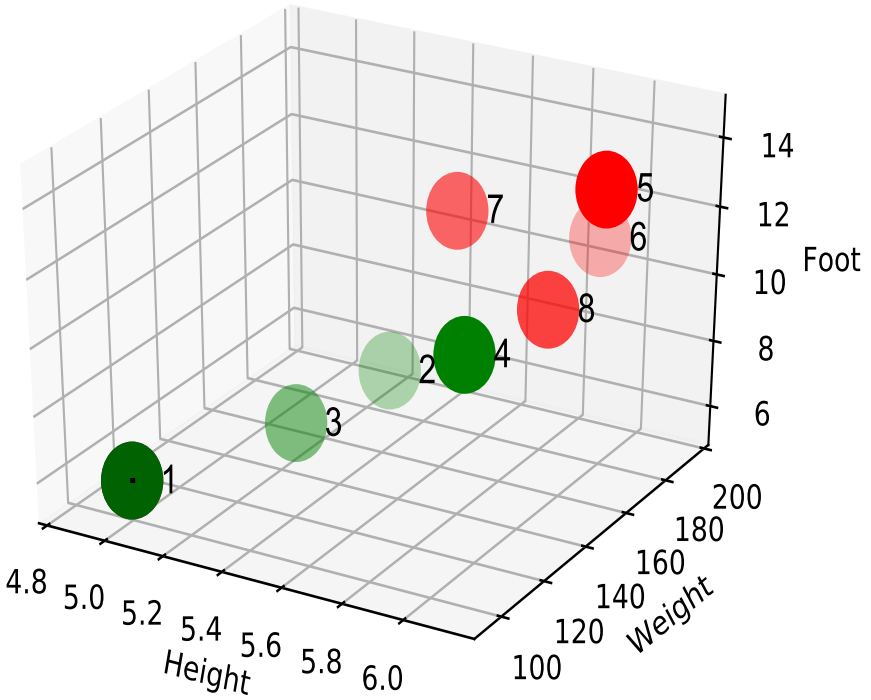
object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

Code for the Dataset

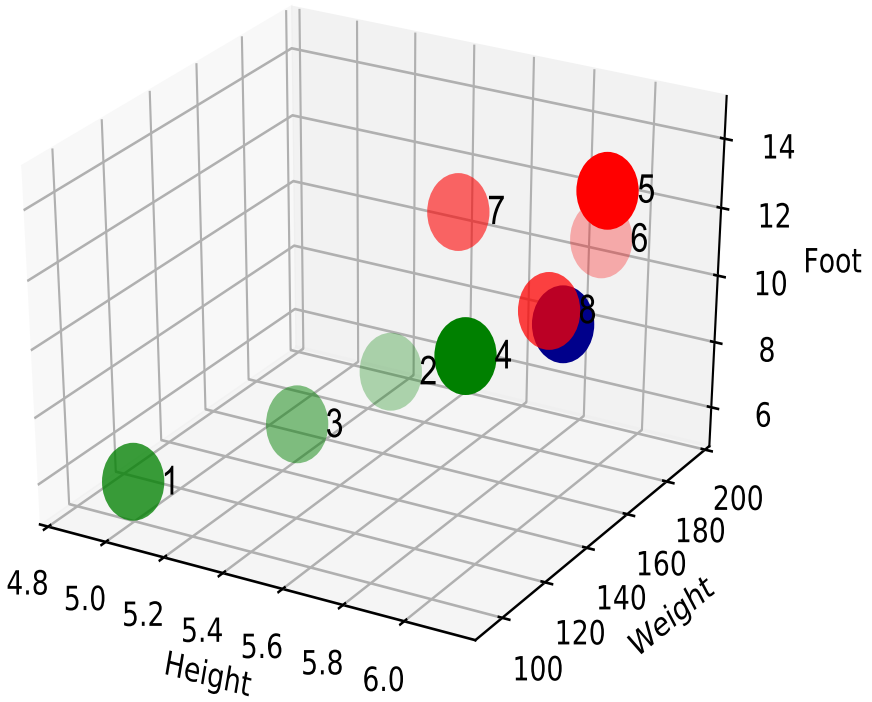
```
import pandas as pd
data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                  'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   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

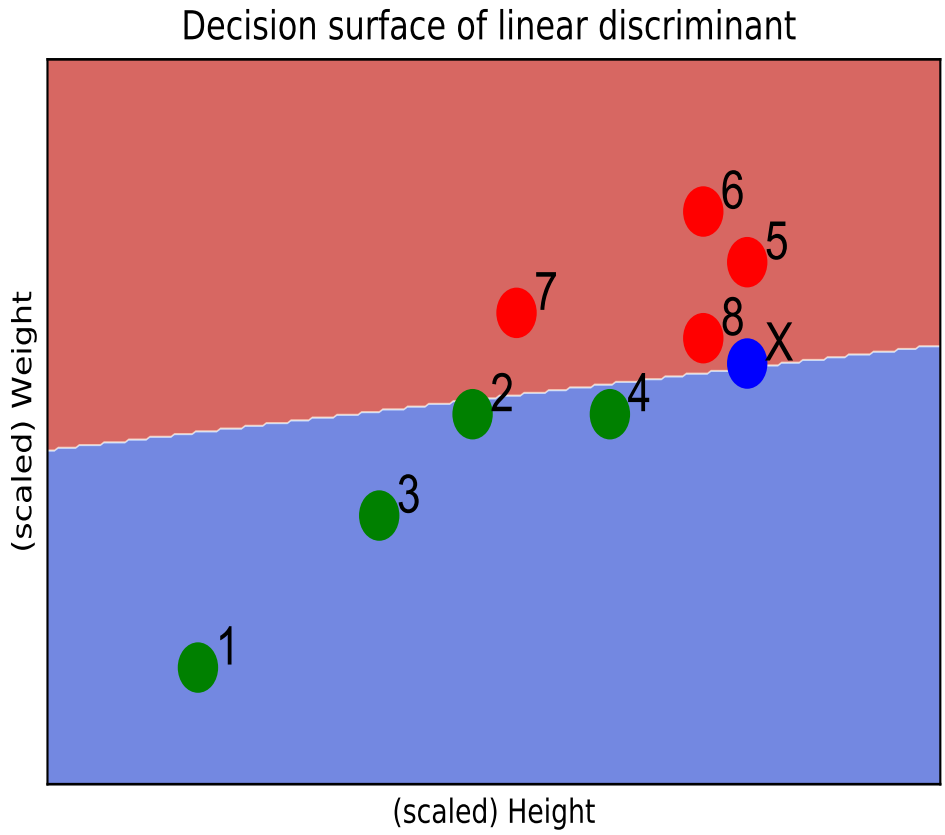


A New Instance



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

A Linear Discriminant



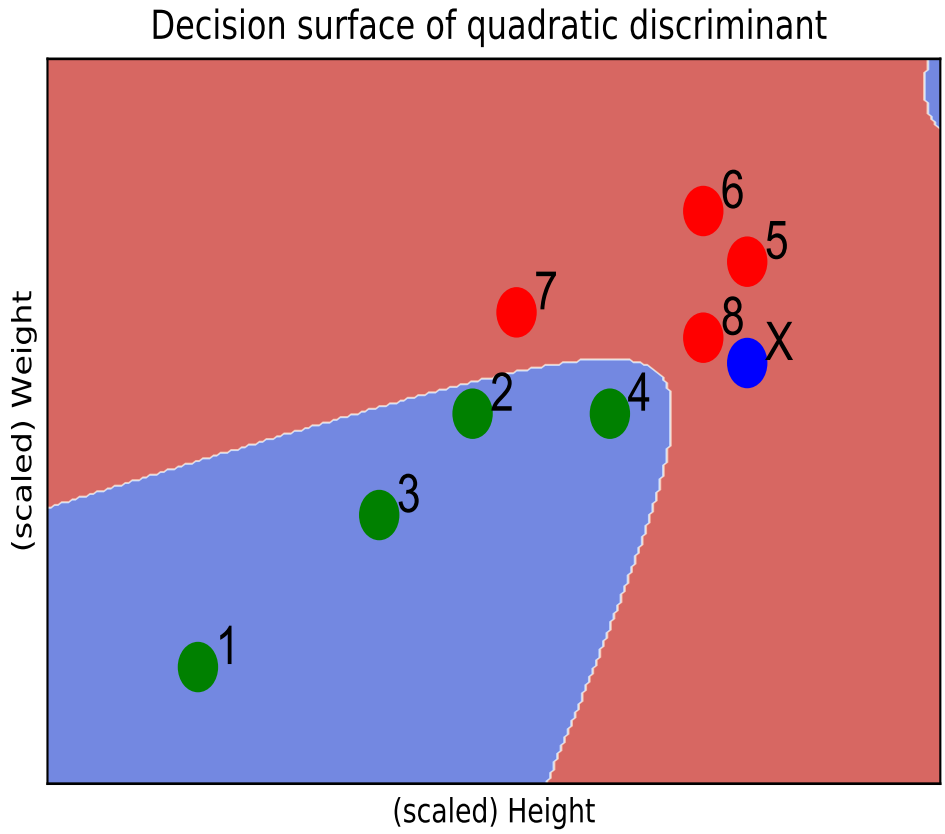
- predict(x^*)=red
- accuracy = 100%

Python Code: LDA

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as
from sklearn.preprocessing import StandardScaler

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

lda_classifier =  LDA(n_components=2)
lda_classifier.fit(X,Y)

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

```
ipdb> predicted[0]
red
ipdb> accuracy
1.0
```

Quadratic Discriminant



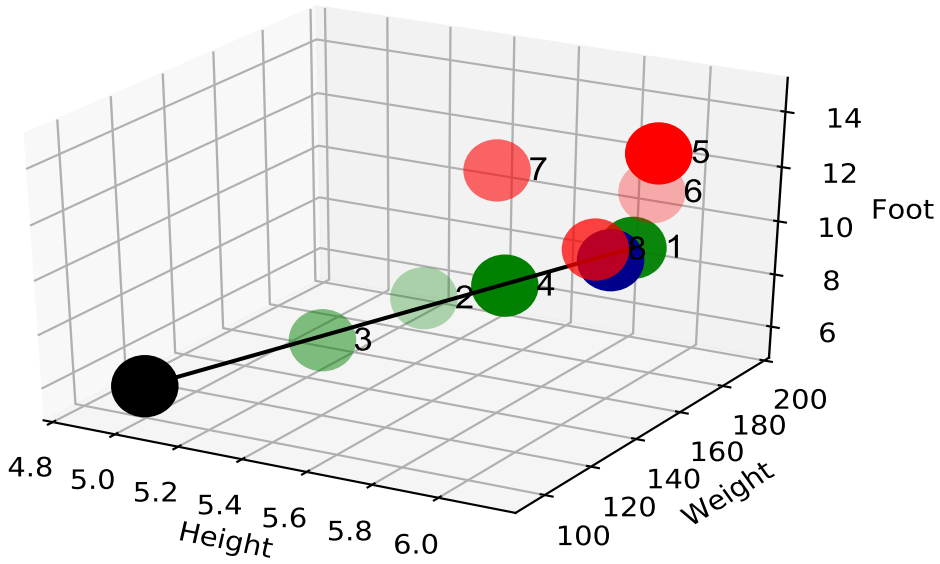
- predict(x^*)=red
- accuracy = 100%

Python Code: QDA

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler

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

qda_classifier =  QDA()
qda_classifier.fit(X,Y)

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

```
ipdb> predicted[0]
red
ipdb> accuracy
1.0
```

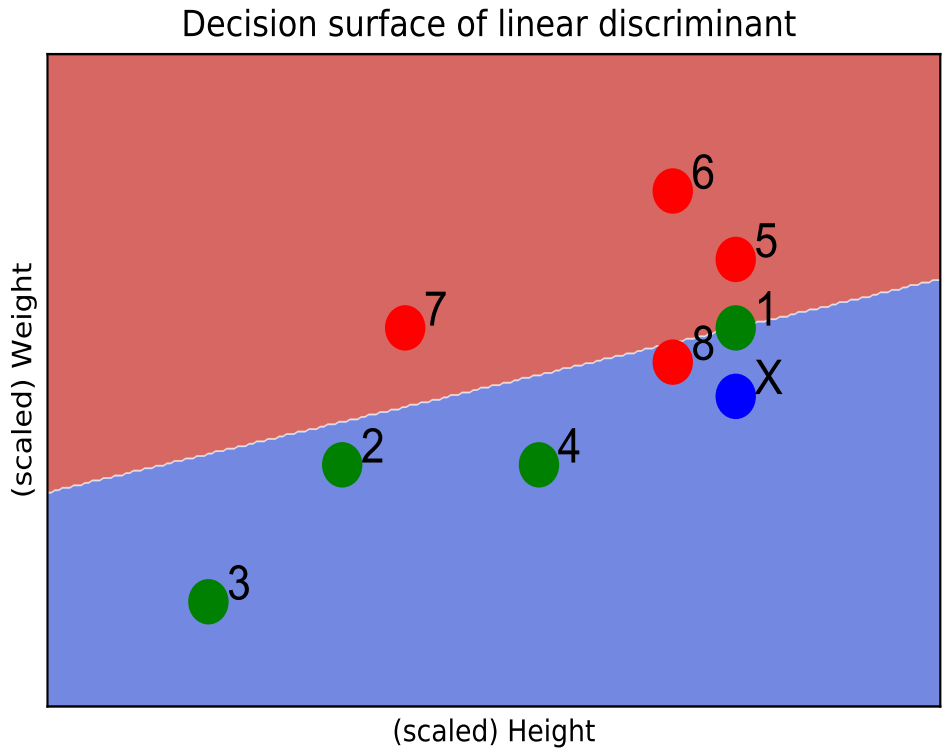
F/W/H Change



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

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

LDA (modified dataset)



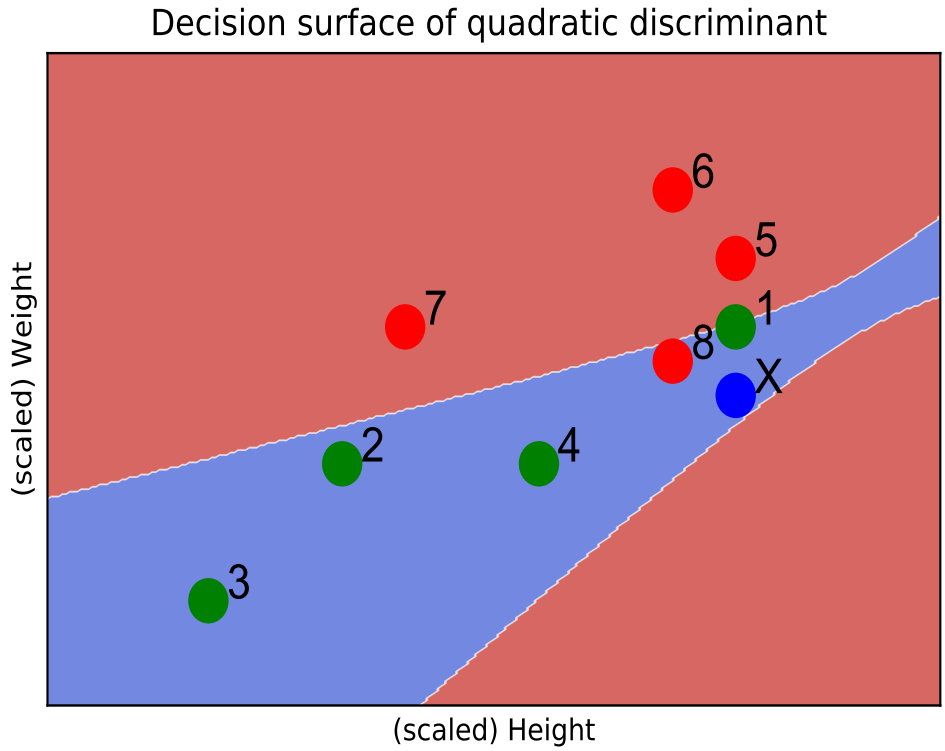
- predict(x^*)=green
- accuracy = 75%

Python Code: LDA (modified dataset)

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as
from sklearn.preprocessing import StandardScaler

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

lda_classifier =  LDA(n_components=2)
lda_classifier.fit(X,Y)
new_x = scaler.transform(np.asmatrix([6, 160]))
predicted = svm_classifier.predict(new_x)
accuracy = svm_classifier.score(X, Y)
```

```
ipdb> predicted[0]
green
ipdb> accuracy
0.75
```

QDA (modified dataset)



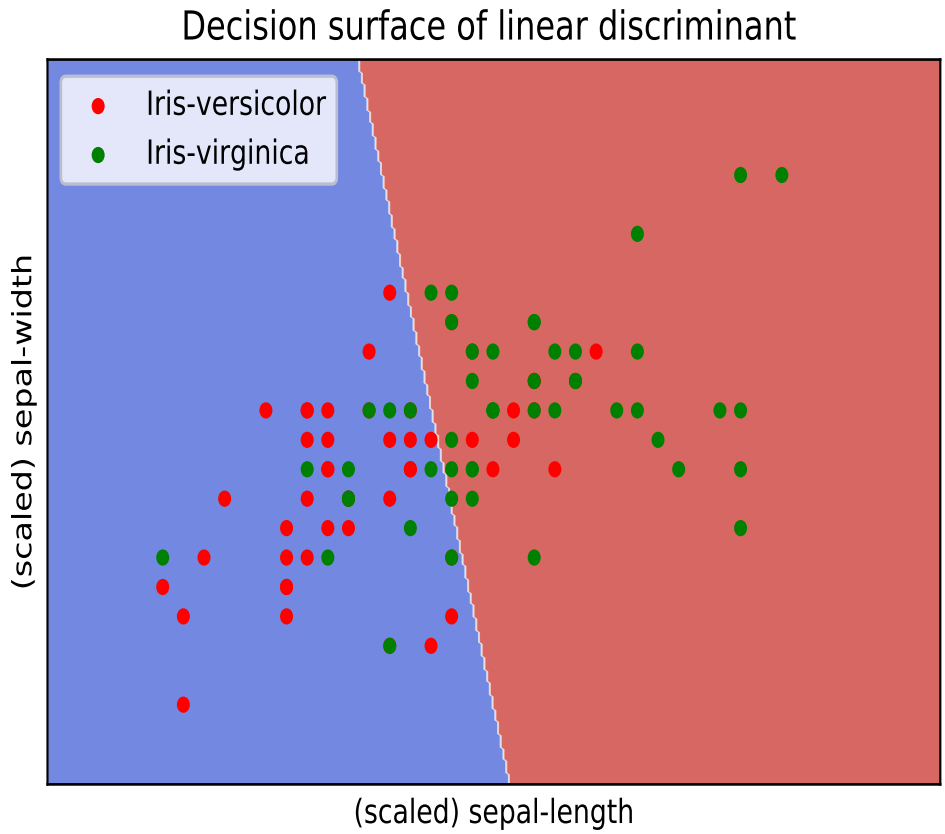
- predict(x^*)=green
- accuracy = 87.5%

Python Code: QDA (modified dataset)

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler

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

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

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

Iris: LDA

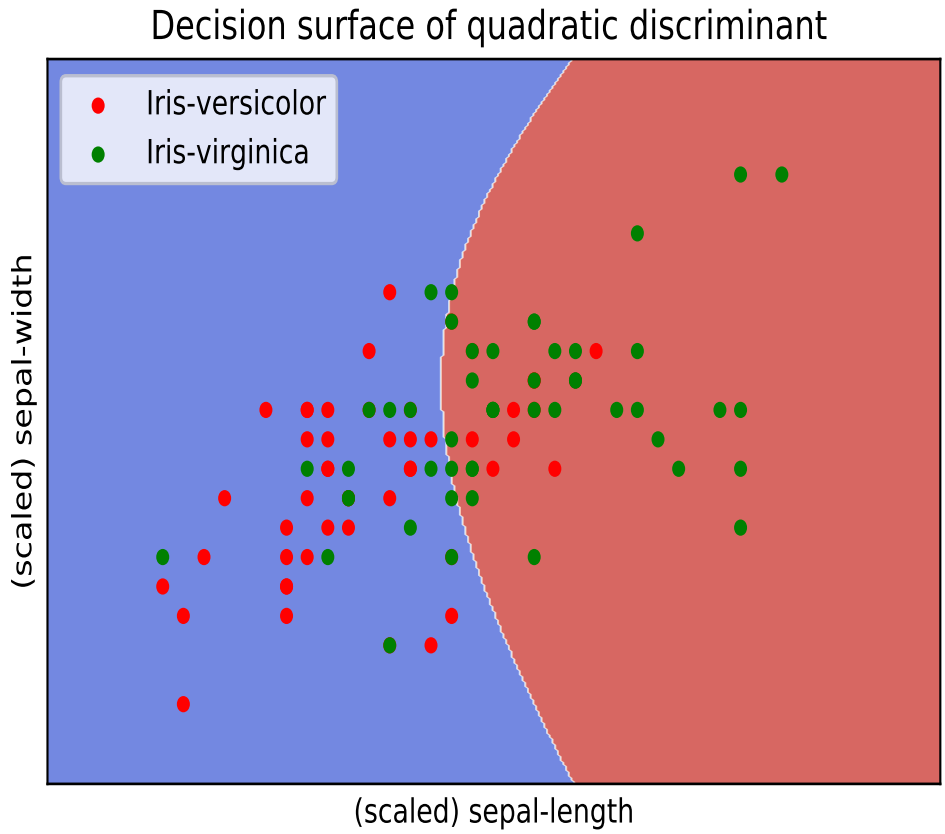


- accuracy = 72%

Python: Iris LDA

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as
from sklearn.preprocessing import StandardScaler

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

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

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

x_label = 'sepal-length'
y_label = 'sepal-width'
data_feature_names = [x_label, y_label]

X = data[data_feature_names].values
scaler = StandardScaler().fit(X)
X = scaler.transform(X)

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)

lda_classifier = LDA()
lda_classifier.fit(X_train,Y_train)

prediction = lda_classifier.predict(X_test)
accuracy = np.mean(prediction == Y_test)
print('prediction accuracy: ' + str(round(accuracy,2)))
```

Iris: QDA



- accuracy = 66%

Python: Iris QDA

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler

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

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

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

x_label = 'sepal-length'
y_label = 'sepal-width'
data_feature_names = [x_label, y_label]

X = data[data_feature_names].values
scaler = StandardScaler().fit(X)
X = scaler.transform(X)

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)

qda_classifier = QDA()
qda_classifier.fit(X_train,Y_train)

prediction = lda_classifier.predict(X_test)
accuracy = np.mean(prediction == Y_test)
print('prediction accuracy: ' + str(round(accuracy,2)))
```