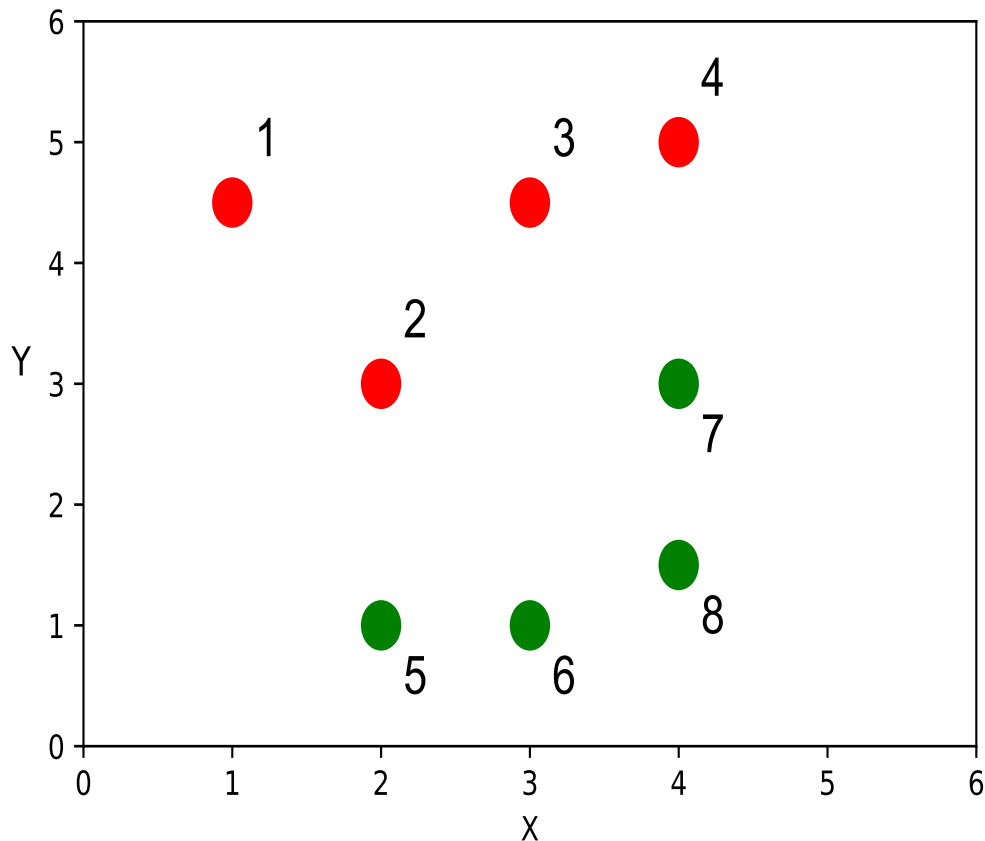


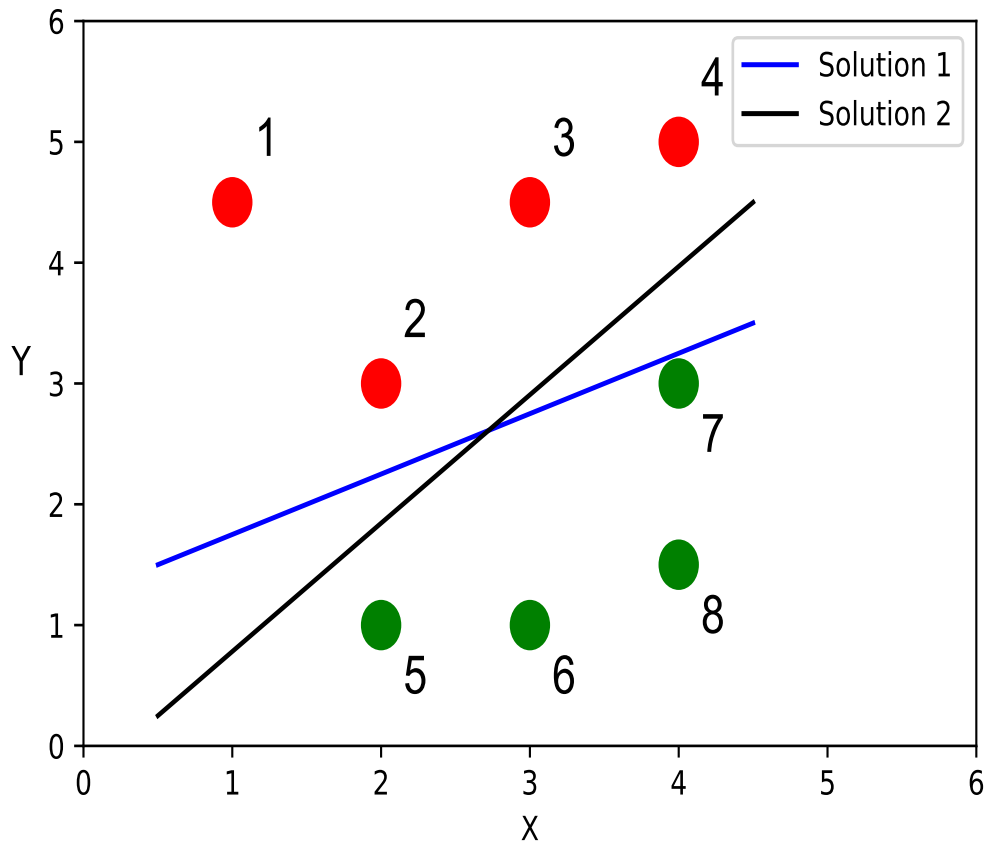
SUPPORT VECTOR MACHINES

Overview



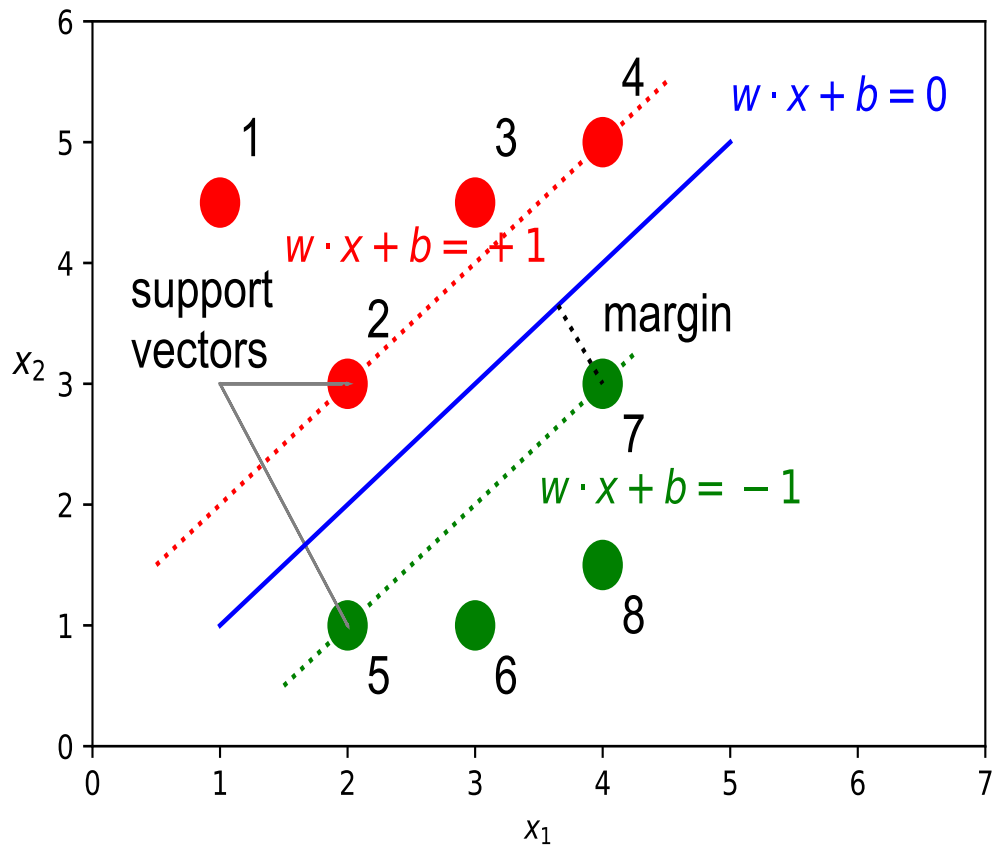
- supervised learning
- want to separate classes

How to Separate?



- many possibilities

SVM Intuition



- use "thickest" line
- maximize margins

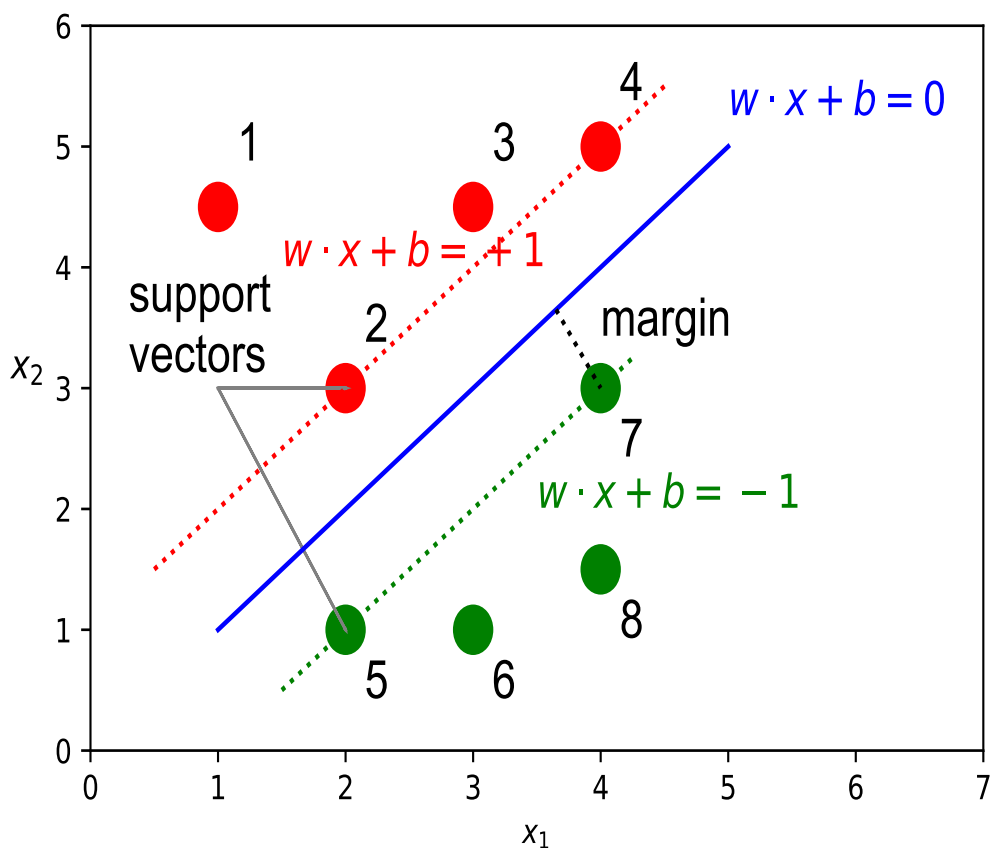
Binary Classification

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

$$H : X \mapsto \{-1, +1\}$$

- low generalization error
- linear classification (based on hyperplanes)

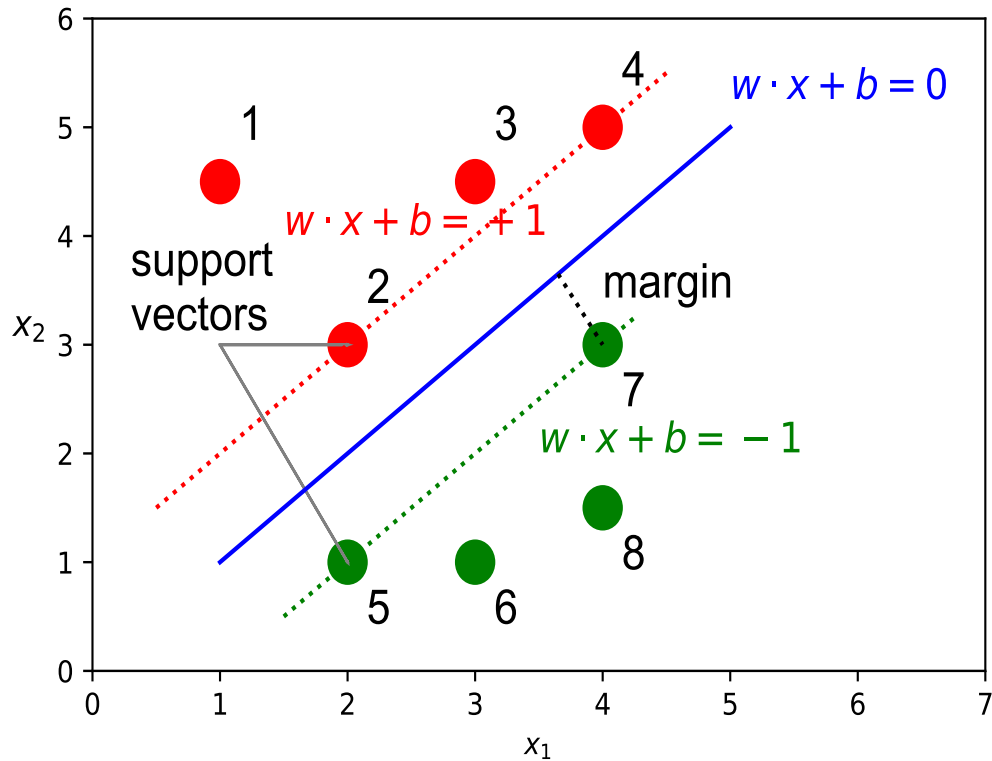
Linear Separation



$$H : X \mapsto \text{sgn}(W \cdot X + b)$$

$$W \in R^N, b \in R$$

Optimal Hyperplane



- $|w \cdot x + b| = 1$ at support vectors
- max margin: $\min_x \frac{|w \cdot x + b|}{\|w\|} = \frac{1}{\|w\|}$

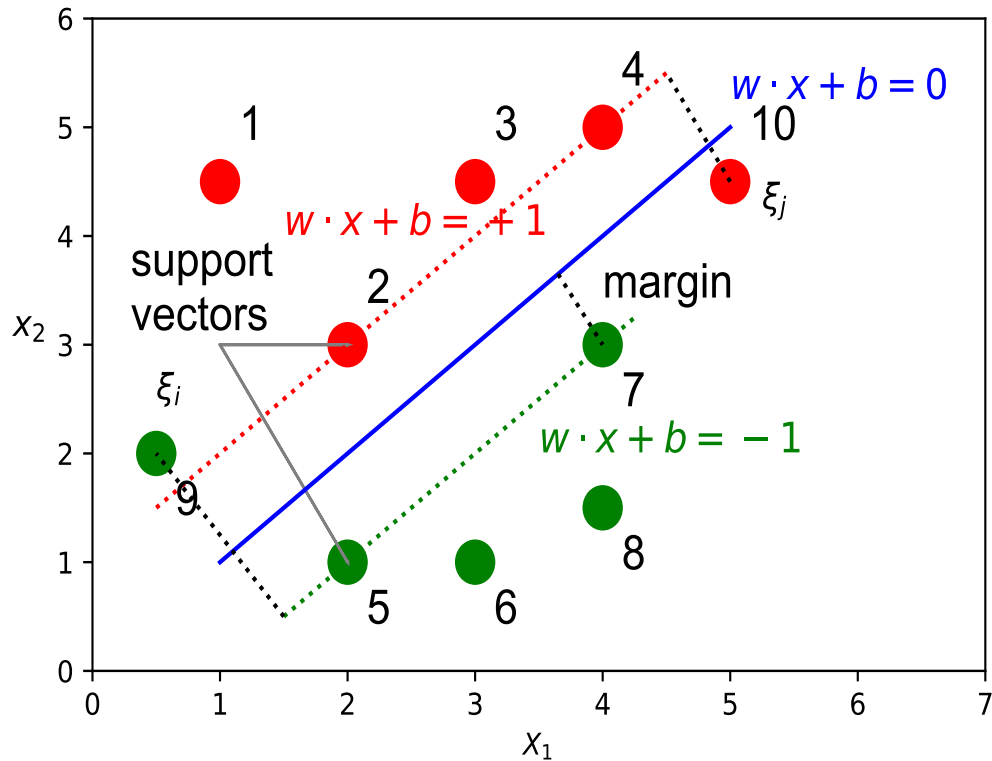
Optimization Problem

- (strictly) convex optimization:

$$\min_{W,b} \frac{1}{2} ||W||^2 \text{ where } y_i(W \cdot X_i + b) \geq 1$$

- unique solution for linearly separable points
- only support vectors for solution

Soft Margin



- slack variables

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

Optimization Problem with Slack Variables

- still (strictly) convex optimization:

$$\min_{W,b} \frac{1}{2} \|W\|^2 + C \sum_{i=1}^m \xi_i$$

where $y_i(W \cdot X_i + b) \geq 1 - \xi_i$

- unique solution
- C is regularization parameter

Optimization Problem

Alternative Formulation

- $y_i(W \cdot X_i + b) \geq 1 - \xi_i$ equivalent to $\xi_i = \max(0, 1 - y_i f(X_i))$
- can re-write optimization as

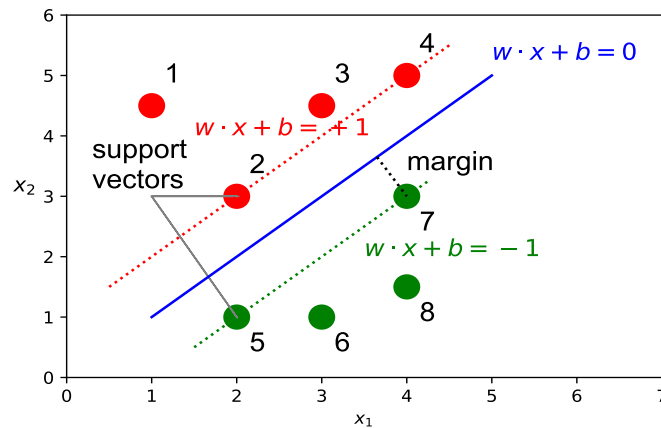
$$\min_W \frac{1}{2} \underbrace{\|W\|^2}_{\text{regularization}} + C \sum_{i=1}^m \underbrace{\max(0, 1 - y_i f(X_i))}_{\text{loss function}}$$

- unique solution
- unconstrained optimization

The Meaning of C

- small C - "soft" margin (ignore constraints)
- C - narrow margin (hard to ignore constraints)
- $C \mapsto \infty$ - hard margin (enforce all constraints)
- for any C still a quadratic optimization
- unique minimum

Loss Function



$$\min_W \frac{1}{2} \|W\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i f(X_i))$$

1. outside margin: $y_i f(X_i) > 1$ - no contribution to loss
2. on the margin: $y_i f(X_i) = 1$ - no contribution to loss
3. violates margin: $y_i f(X_i) < 1$ - contributes to loss

Dual Optimization

- constrained optimization

$$\max_{\alpha} \sum_{i=1}^m \alpha_i = \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j (X_i \cdot X_j)$$

subject to: $\alpha_i \geq 0$ and $\sum_{i=1}^m \alpha_i y_i = 0$

- solution:

$$h(x) = \text{sgn} \left(\sum_{i=1}^m \alpha_i y_i (X_i \cdot X) + b \right)$$

$$\text{with } b = y_i - \sum_{j=1}^m \alpha_j y_j (X_j \cdot X_i)$$

for any support vectors X_i

Kernel Methods

- define $K : X \times X \mapsto R$ so that

$$K(X, X') = \phi(X) \cdot \phi(X')$$

- K is similarity measure
- easier to compute than dot product and $\Phi()$
- example: $K(X, Y) = (X \cdot Y)^2$

$$\Phi(x_1, x_2) = (x_1^2, x_1x_2\sqrt{2}, x_2^2)$$

$$\begin{aligned}\Phi(x_1, x_2) \cdot \Phi(y_1, y_2) &= x_1^2x_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\ &= (x_1y_1)^2 + 2x_1y_1x_2y_2 + (x_2y_2)^2 \\ &= (x_1y_1 + x_2y_2)^2 = (X \cdot Y)^2 = K(X, Y)\end{aligned}$$

Kernels for Non-Linear SVM

- allow separability in higher dimensions
- function like dot product

1. polynomial

$$K(X, X') \mapsto (X \cdot X' + C)^p$$

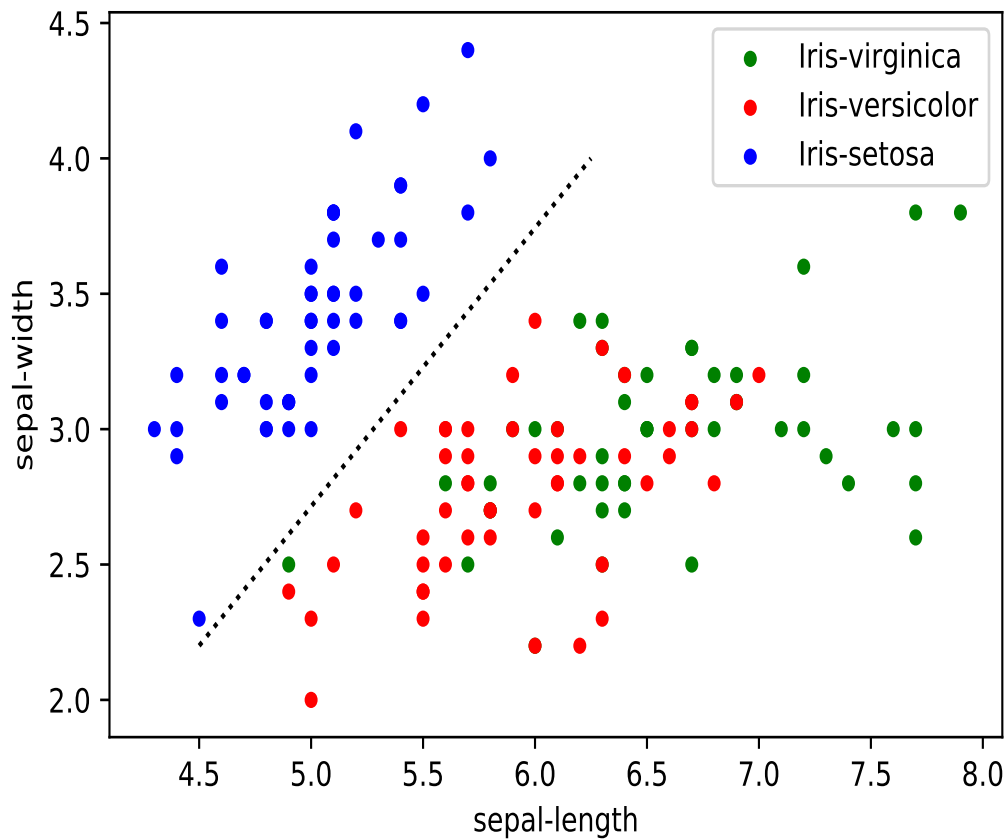
2. sigmoid

$$K(X, X') \mapsto \tanh(k * X \cdot X' - \delta)$$

3. Gauss (radial basis functions)

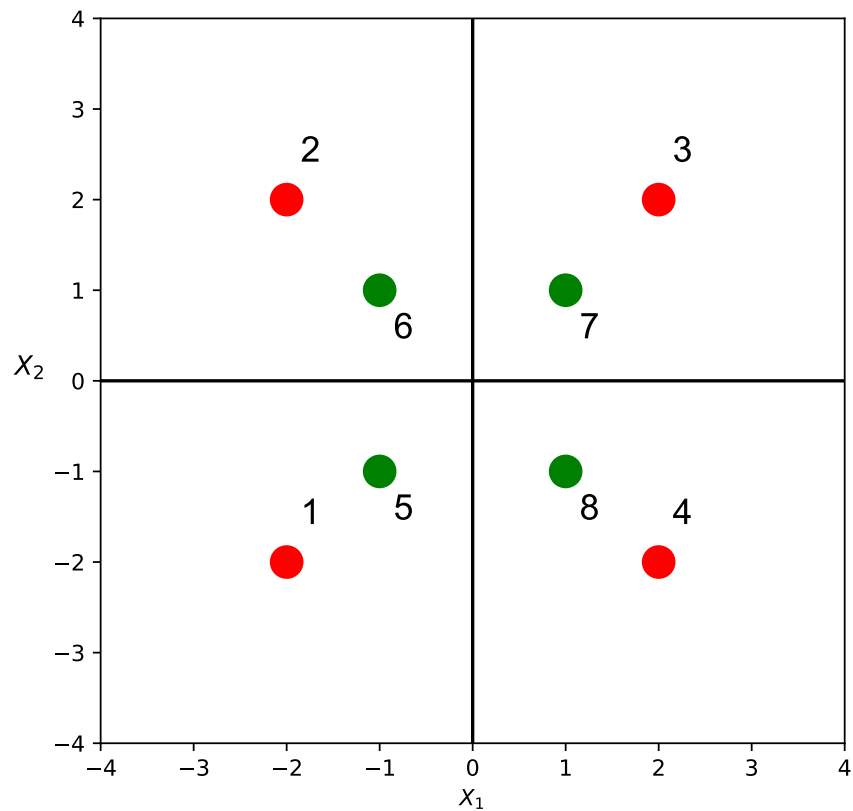
$$K(X, X') \mapsto \exp \left(-\nu(X - X')^2 / 2\sigma^2 \right)$$

Linear Separability



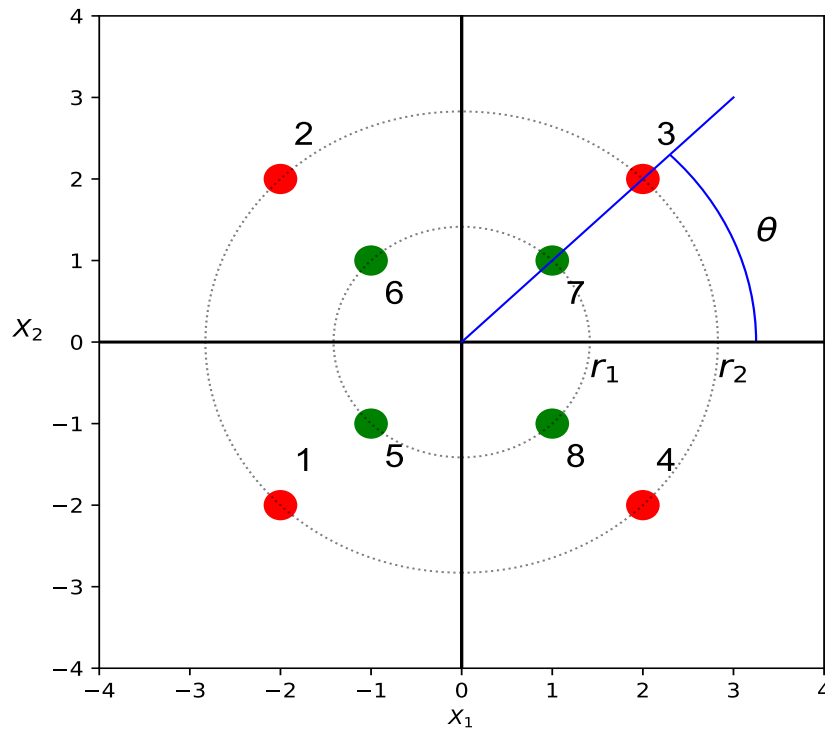
- draw a hyperplane
- difficult in many cases

Example of Difficulty



- non-separable in 2 dimensions

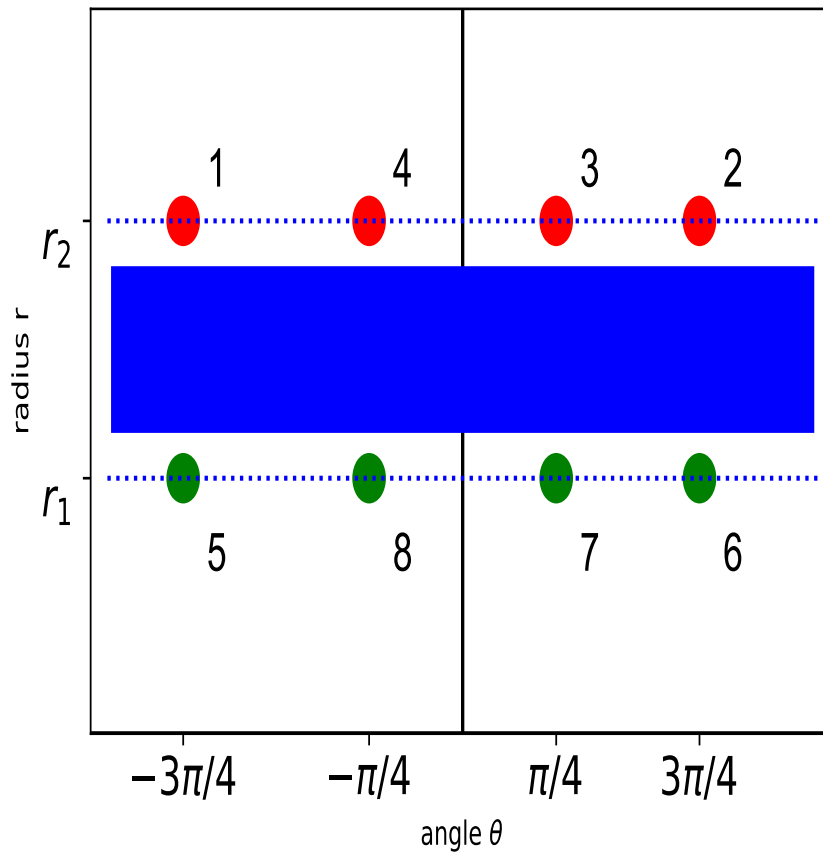
Mapping to Polar



$$\Phi : (x_1, x_2) \mapsto (\sqrt{x_1^2 + x_2^2}, \arccos \frac{x_1}{\sqrt{x_1^2 + x_2^2}})$$

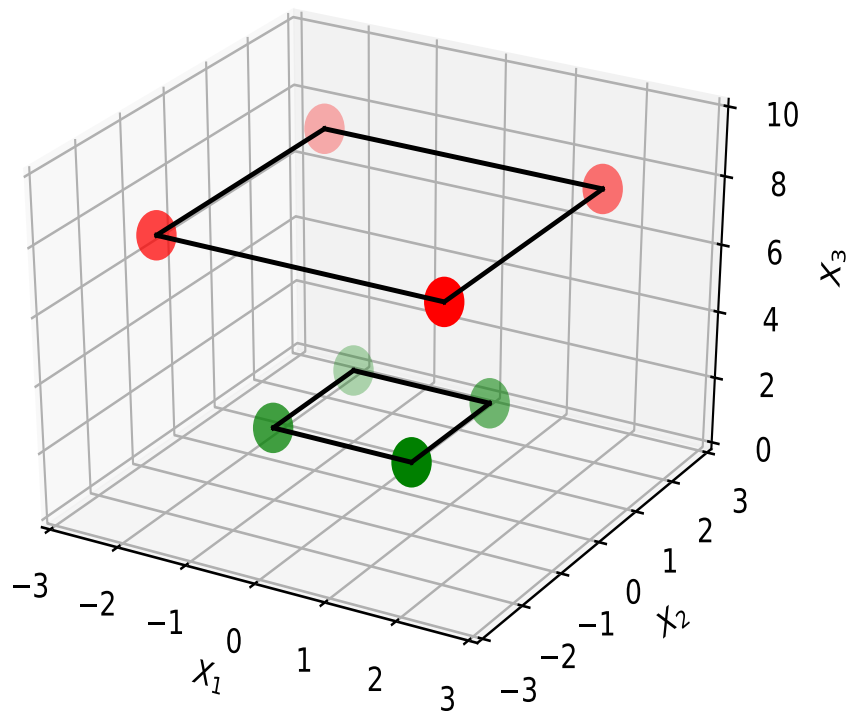
$$\Phi(1, 1) \mapsto (\sqrt{2}, \pi/4)$$

Linear Separation in Polar Coordinates



- separation by radius

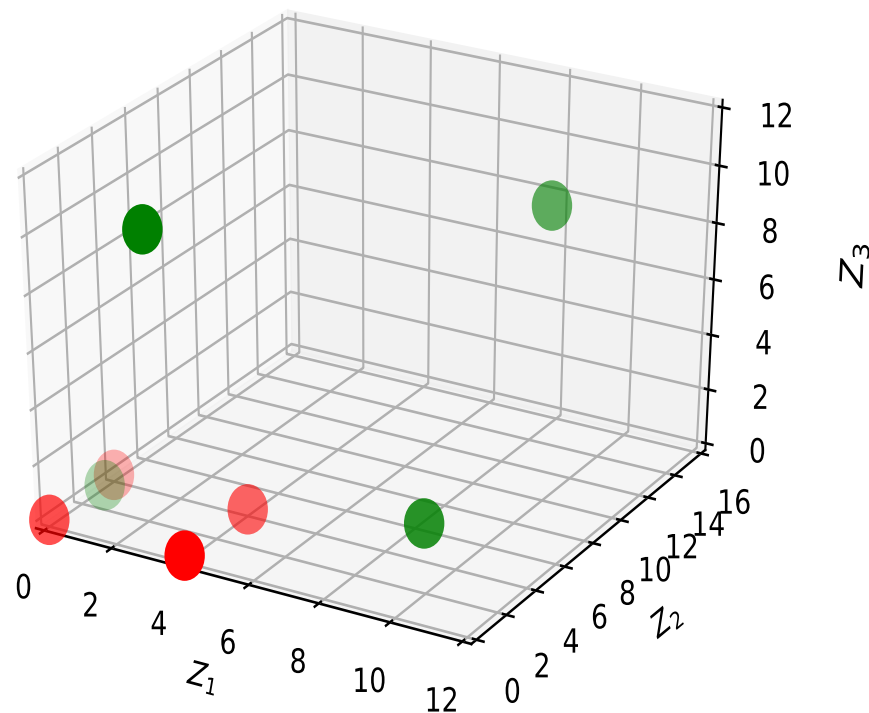
Alternative Solution



$$\Phi(x_1, x_2) \mapsto (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$

- separable by z

Another Solution



$$\Phi(x_1, x_2) \mapsto ((x_1 + 2)^2, \sqrt{2}(x_1 + 2)(x_2 + 2), (x_2 + 2)^2)$$

- separable in 3 dimensions

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

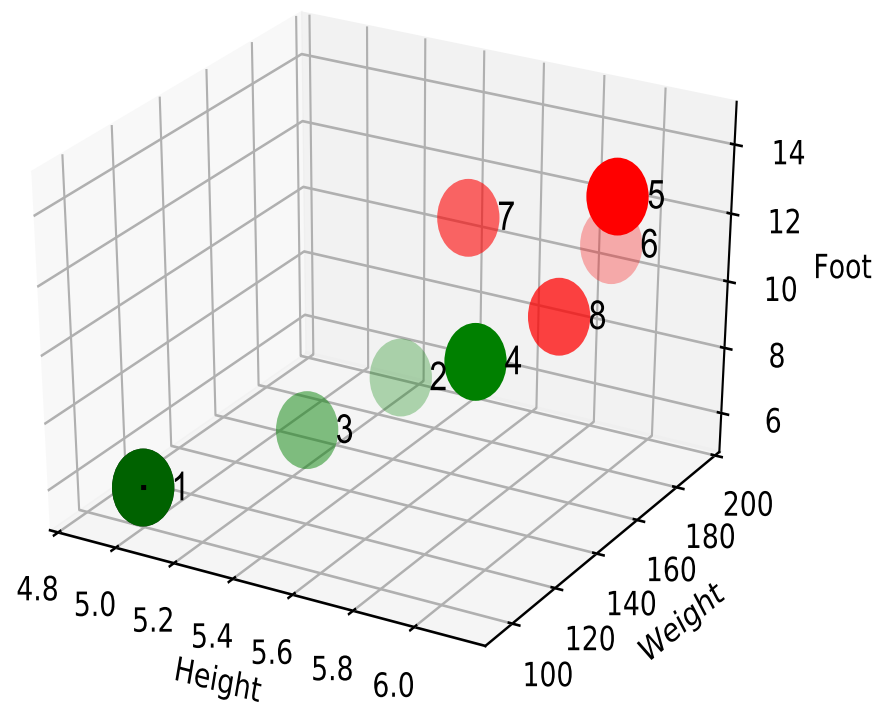
Code for the Dataset

```
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'] )
```

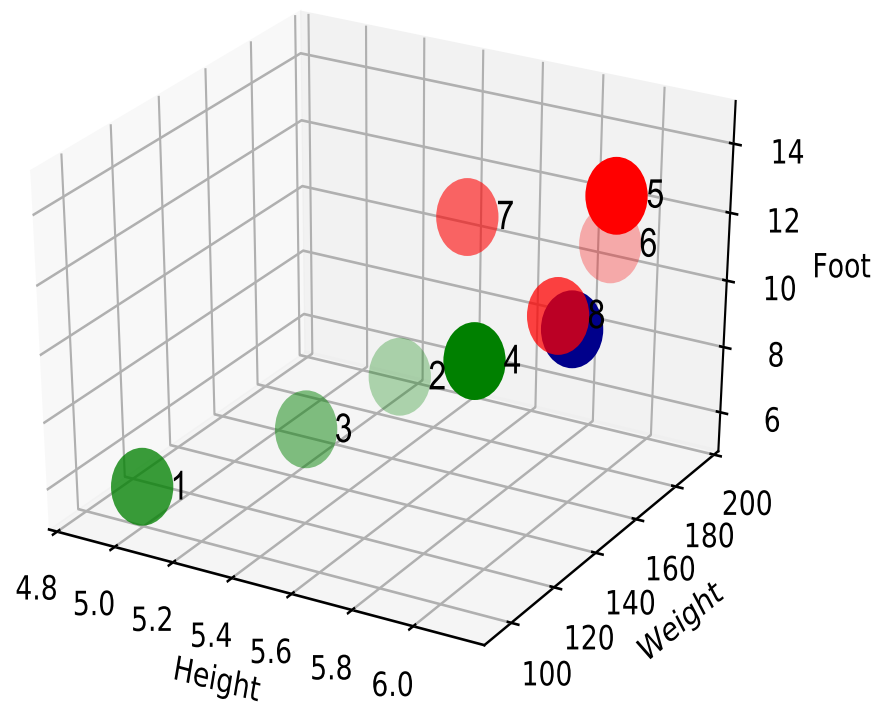
```
ipdb> data
```

	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

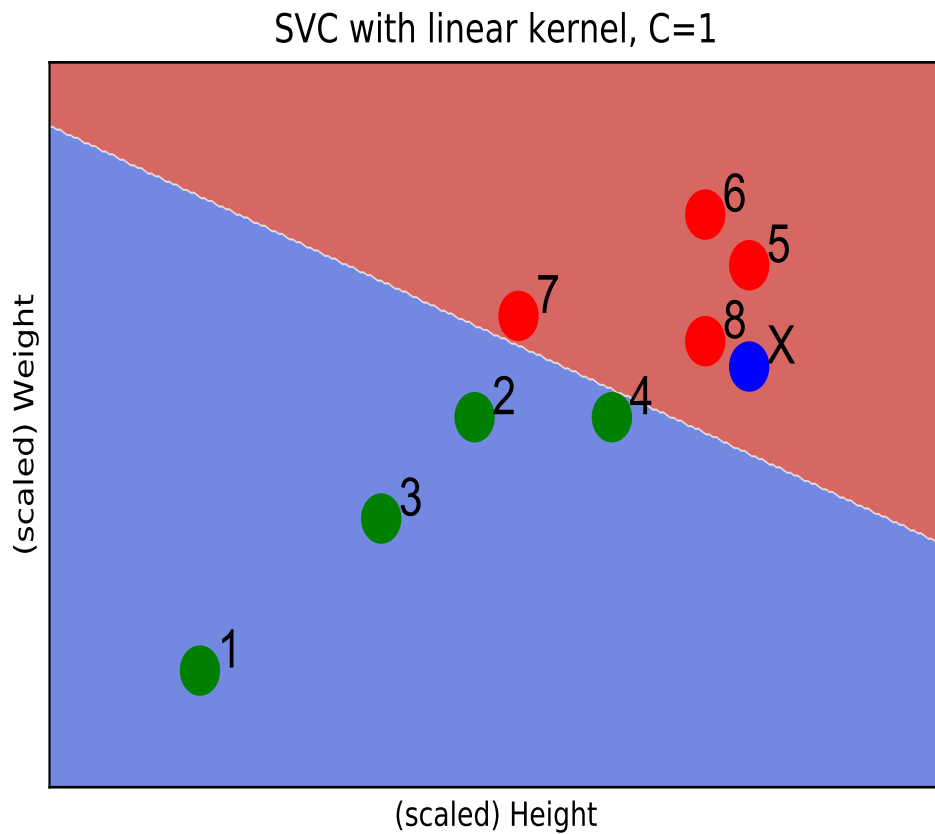


A New Instance



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

A Linear SVM



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

Python Code: Linear

```
import pandas as pd
import numpy as np
from sklearn import svm
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'] )

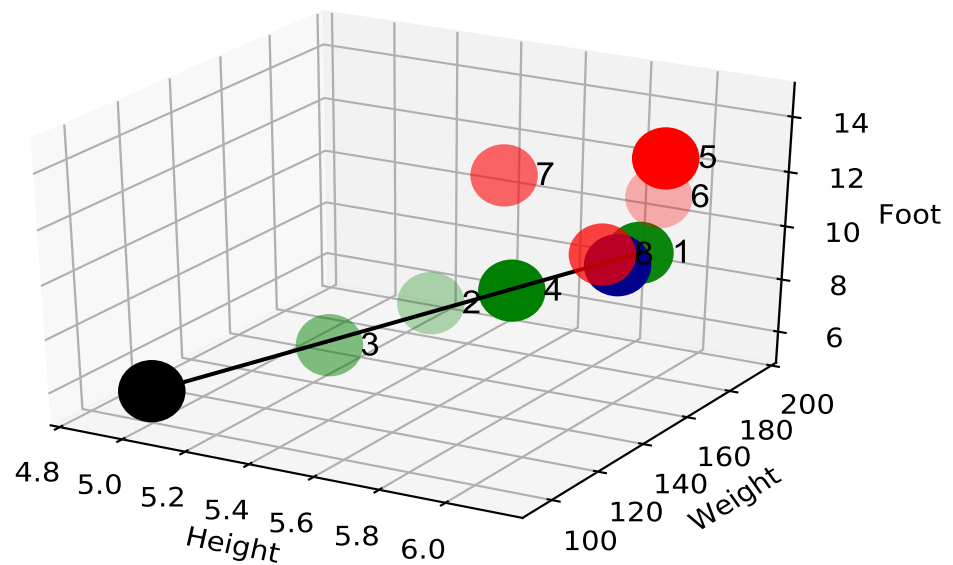
X = data[['Height', 'Weight']].values
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

svm_classifier = svm.SVC(kernel='linear')
svm_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]
red
ipdb> accuracy
1.0
```

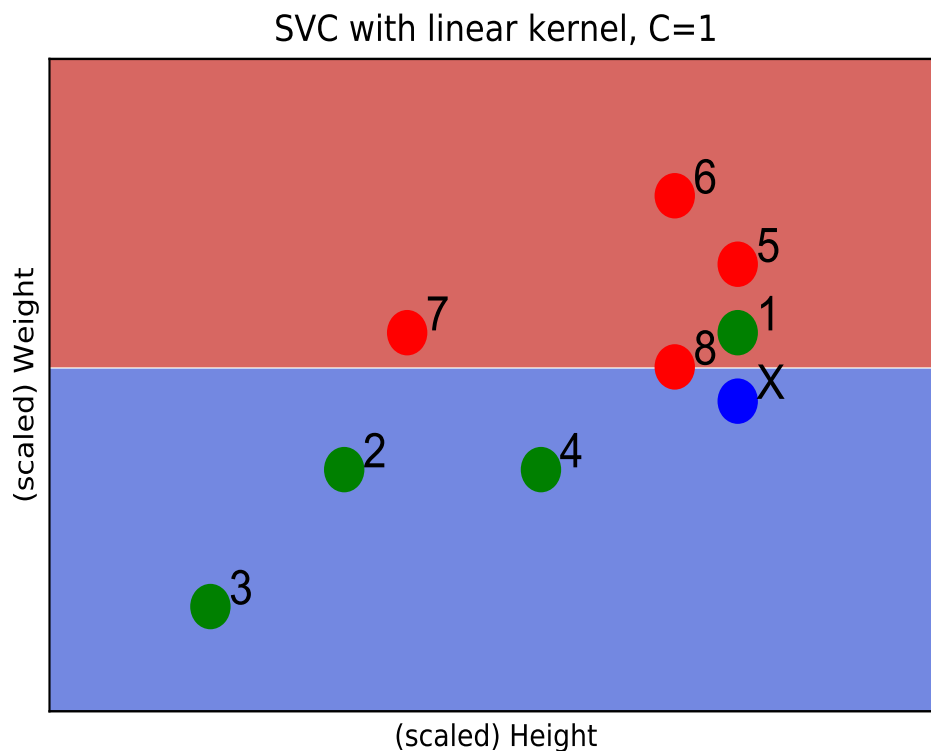
F/W/H Change



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

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

A Linear SVM (modified dataset)



- $\text{predict}(x^*) = \text{green}$
- $\text{accuracy} = 75\%$

Python Code: Linear (modified dataset)

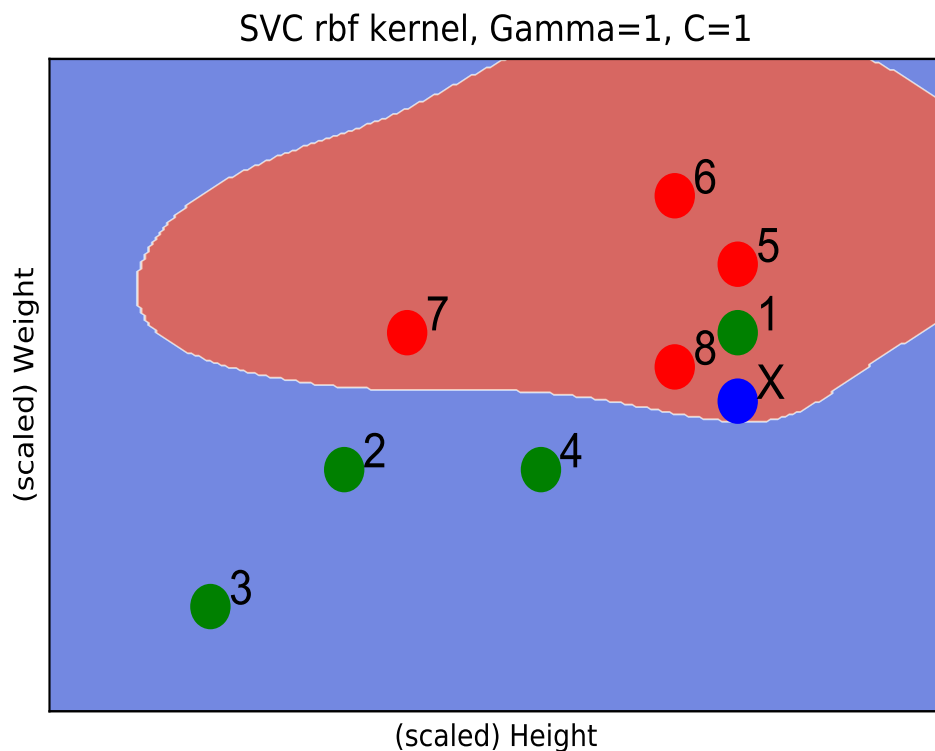
```
import pandas as pd
import numpy as np
from sklearn import svm
from sklearn.preprocessing import StandardScaler

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()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

svm_classifier = svm.SVC(kernel='linear')
svm_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
```

A Gaussian SVM (modified dataset)



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

Python Code: Gaussian (modified dataset)

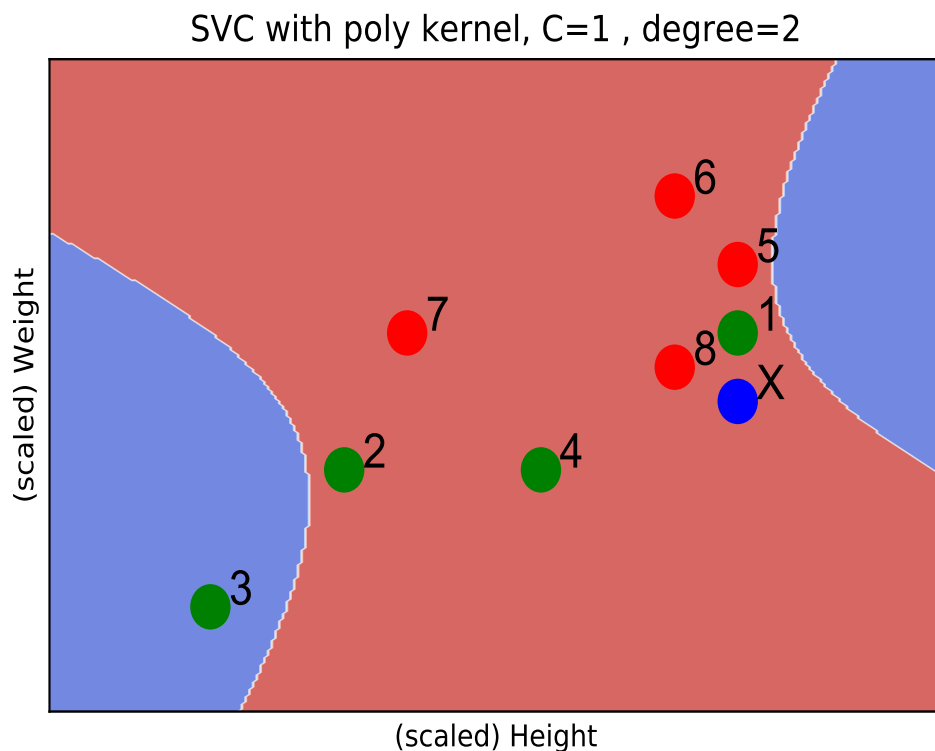
```
import pandas as pd
import numpy as np
from sklearn import svm
from sklearn.preprocessing import StandardScaler

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()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

svm_classifier = svm.SVC(kernel='rbf')
svm_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]
red
ipdb> accuracy
0.875
```

Polynomial (d=2) SVM (modified dataset)



- $\text{predict}(x^*) = \text{'red'}$
- $\text{accuracy} = 62.5\%$

Python Code: Poly (d=2, modified dataset)

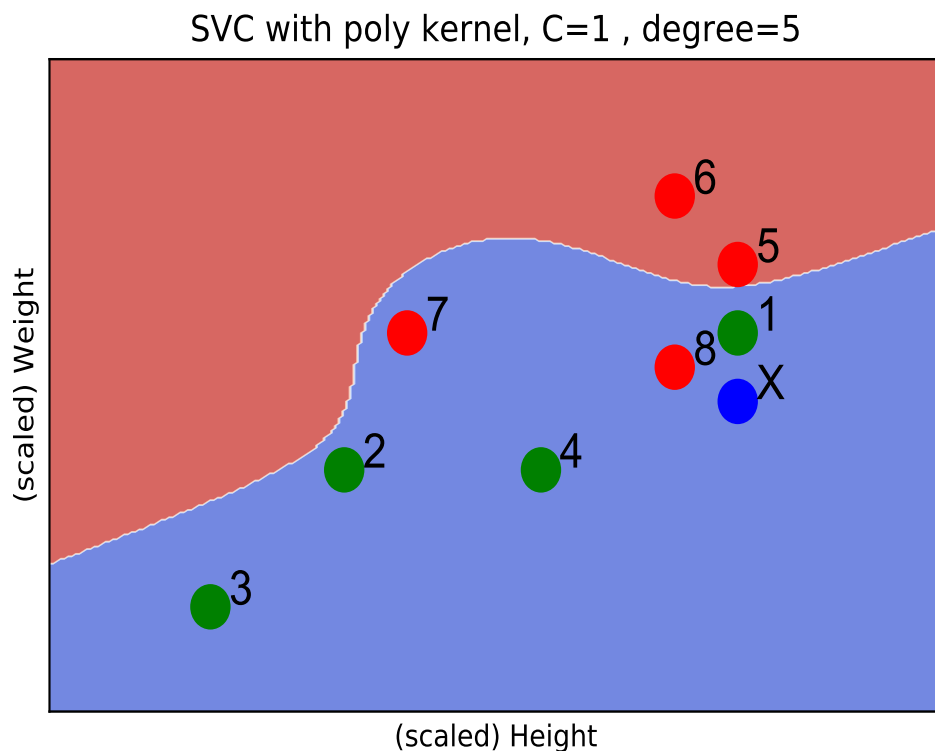
```
import pandas as pd
import numpy as np
from sklearn import svm
from sklearn.preprocessing import StandardScaler

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()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

svm_classifier = svm.SVC(kernel='poly', degree=2)
svm_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]
red
ipdb> accuracy
0.625
```

Polynomial (d=5) SVM (modified dataset)



- $\text{predict}(x^*) = \text{'green'}$
- $\text{accuracy} = 75\%$

Python Code: Poly (d=5, modified dataset)

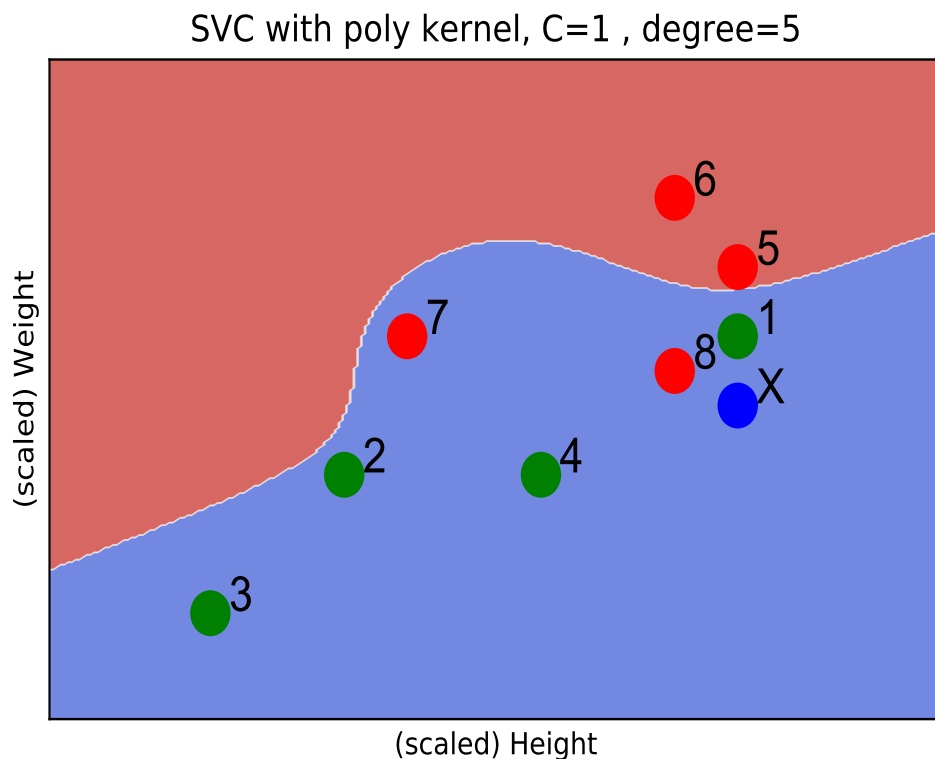
```
import pandas as pd
import numpy as np
from sklearn import svm
from sklearn.preprocessing import StandardScaler

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()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

svm_classifier = svm.SVC(kernel='poly', degree=3)
svm_classifier.fit(X,Y)
new_x = scaler.transform(np.asmatrix([6, 160]))
predicted = svm_classifier.predict(new_x)
accuracy = svm_classifier.score(X, Y)

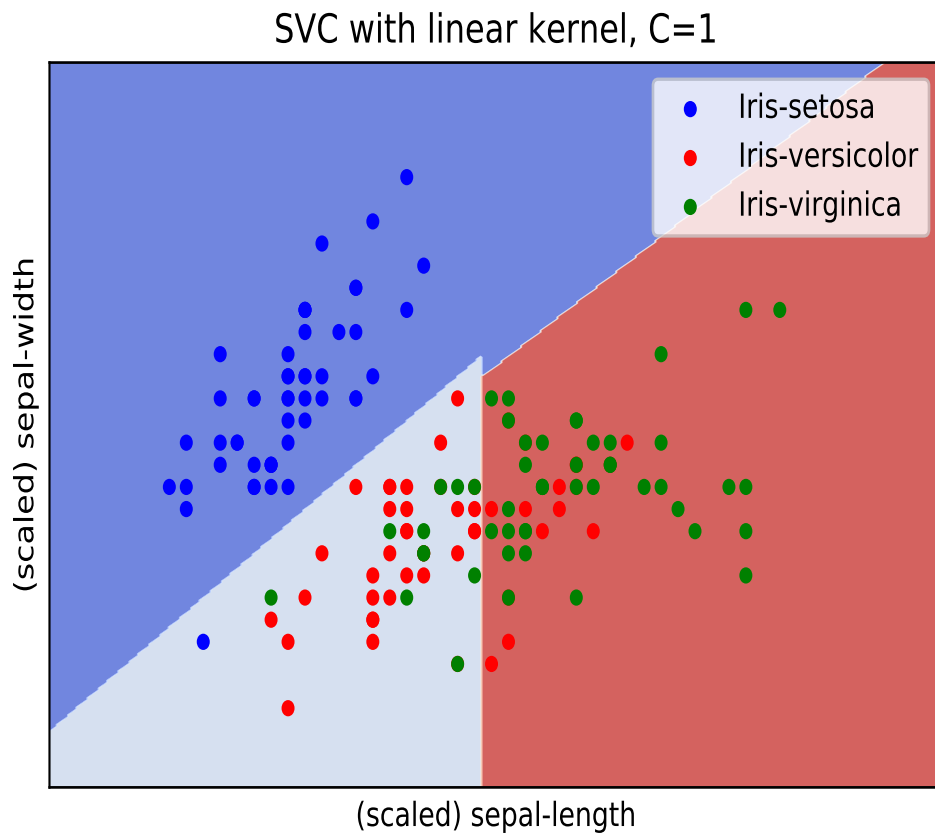
ipdb> predicted[0]
green
ipdb> accuracy
0.75
```

Polynomial (d=9) SVM (modified dataset)



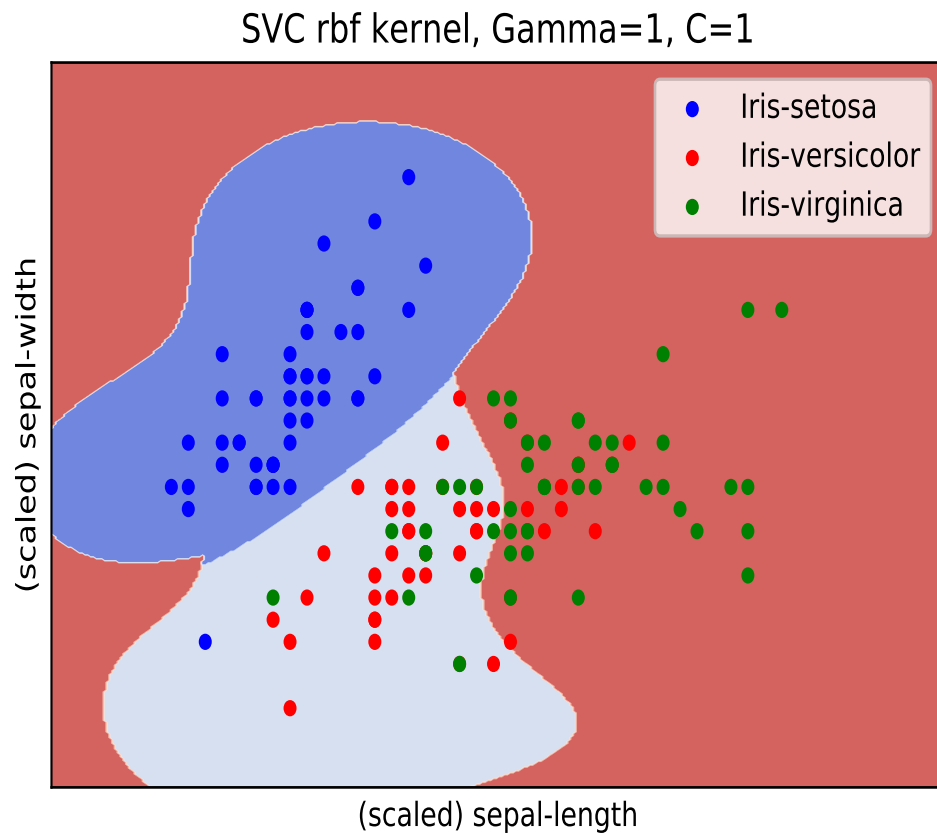
- $\text{predict}(x^*) = \text{'green'}$
- accuracy = 87.5% (high d)

Iris: Linear SVM



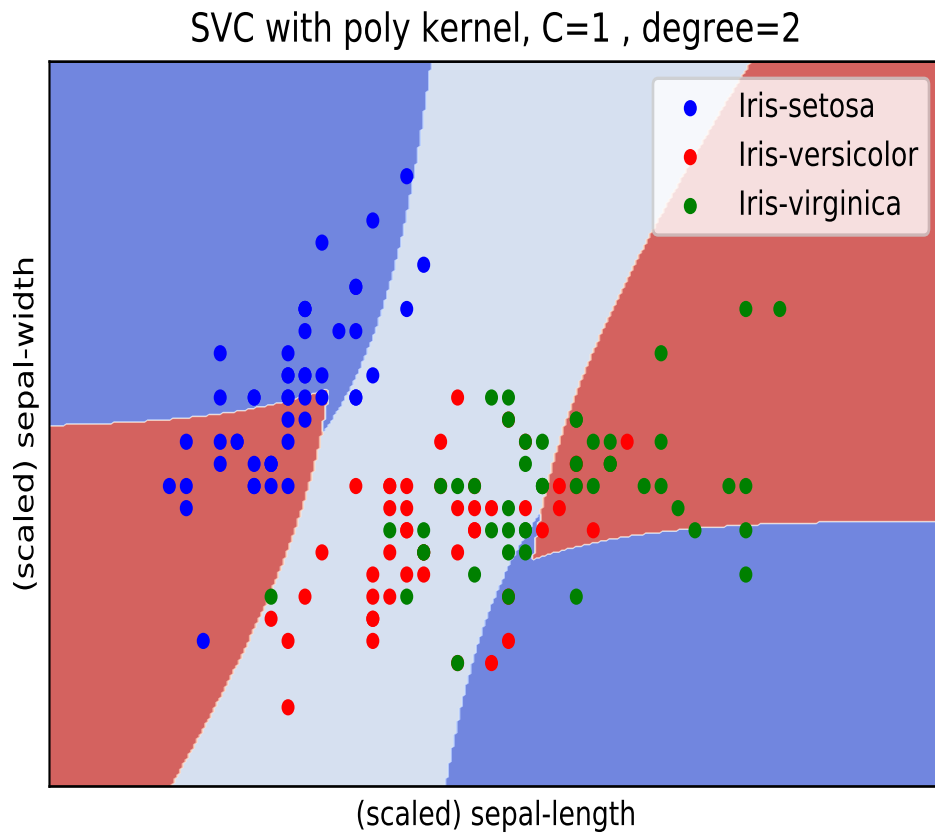
- accuracy = 80%

Iris: Gaussian SVM



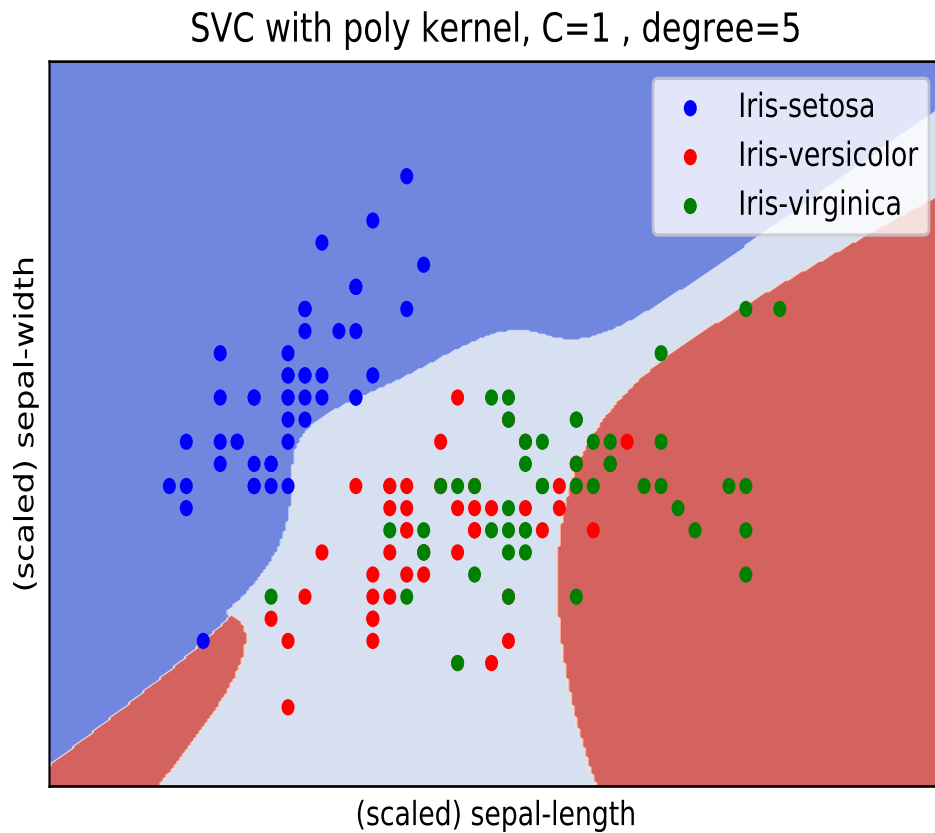
- accuracy = 80%

Iris: Poly SVM (d=2)



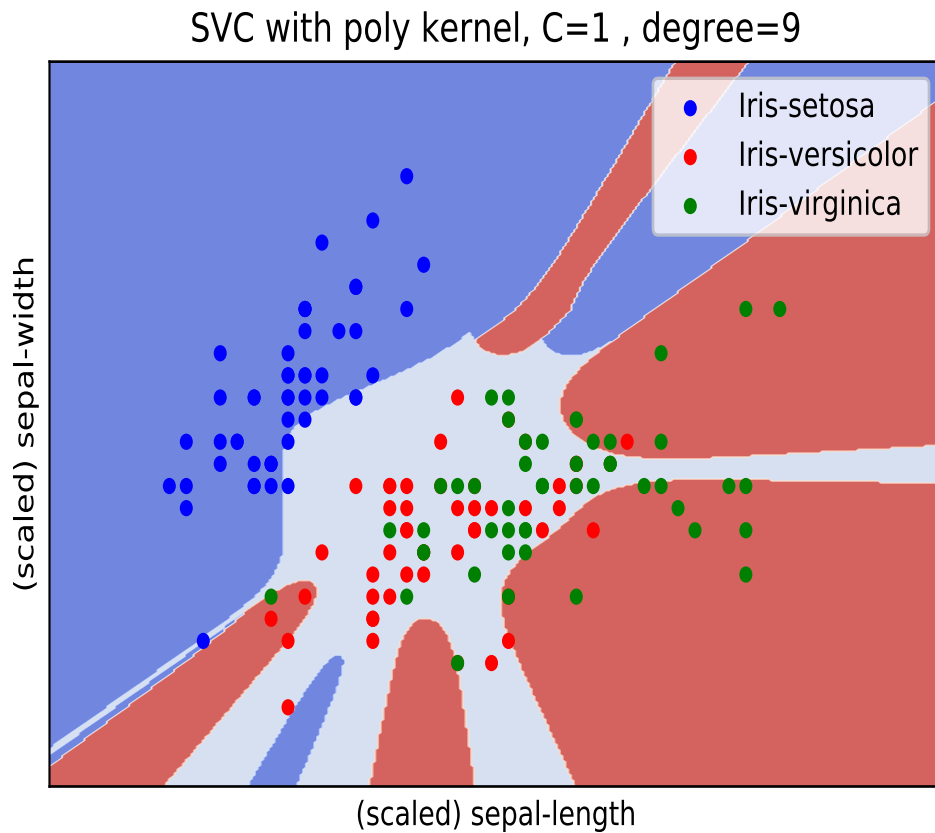
- accuracy = 47%

Iris: Poly SVM (d=5)



- accuracy = 75%

Iris: Poly SVM (d=9)



- accuracy = 64%