17. Support Vector Machines (SVM)

Bruce E. Shapiro

Getting Started in Machine Learning

Last revised: March 23, 2019

Find "Best" Linear Separation of Classes

- Consider $\mathbb{R}^2$: $y = a + bx$

Find "Best" Linear Separation of Classes

- Consider $\mathbb{R}^2$: $y = a + bx$
- Let $x \rightarrow x_0$, $y \rightarrow x_1$, $b \rightarrow w_0$ then $x_1 = a + w_0 x_0$

Find “Best” Linear Separation of Classes

- Consider $\mathbb{R}^2$: $y = a + bx$
- Let $x \to x_0$, $y \to x_1$, $b \to w_0$ then $x_1 = a + w_0 x_0$
- In $\mathbb{R}^d$: $x_{d-1} = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-2} x_{d-2}$

Find “Best” Linear Separation of Classes

- Consider $\mathbb{R}^2$: $y = a + bx$
- Let $x \rightarrow x_0$, $y \rightarrow x_1$, $b \rightarrow w_0$ then $x_1 = a + w_0 x_0$
- In $\mathbb{R}^d : x_{d-1} = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-2} x_{d-2}$
- Rearrange: $0 = a + w_0 x_0 + w_1 x_1$
- In $\mathbb{R}^d$: $0 = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-1} x_{d-1}$

Find "Best" Linear Separation of Classes

- Consider $\mathbb{R}^2$: $y = a + bx$
- Let $x \to x_0$, $y \to x_1$, $b \to w_0$ then $x_1 = a + w_0 x_0$
- In $\mathbb{R}^d$: $x_{d-1} = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-2} x_{d-2}$
- Rearrange: $0 = a + w_0 x_0 + w_1 x_1$
- In $\mathbb{R}^d$: $0 = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-1} x_{d-1}$
- In general: $0 = a + \mathbf{w}^\mathsf{T}\mathbf{x}$

Find "Best" Linear Separation of Classes

- Consider $\mathbb{R}^2$: $y = a + bx$
- Let $x \to x_0$, $y \to x_1$, $b \to w_0$ then $x_1 = a + w_0 x_0$
- In $\mathbb{R}^d : x_{d-1} = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-2} x_{d-2}$
- Rearrange: $0 = a + w_0 x_0 + w_1 x_1$
- In $\mathbb{R}^d$: $0 = a + w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{d-1} x_{d-1}$
- In general: $0 = a + \mathbf{w}^\mathsf{T} \mathbf{x}$
- In $\mathbb{R}^2$ suppose the line $0 = a + \mathbf{w}^\mathsf{T} \mathbf{x}$ separates the classes. Then for any point $\mathbf{x}$ in the plane,

$$a + \mathbf{w}^\mathsf{T} \mathbf{x} \begin{cases} > 0 & \text{points in cluster 1} \\ = 0 & \text{points on the line} \\ < 0 & \text{points in cluster 2} \end{cases}$$

- The vector **w** is perpendicular to the line $0 = a + \mathbf{w}^\mathsf{T}\mathbf{x}$
  Proof in $\mathbb{R}^2$. The hyperplane separating the clusters becomes a line $0 = a + bx - y = a + w_0 x_0 + w_1 x_1$ in $\mathbb{R}^2$. Then Consider

$$\mathbf{w} = (w_0, w_1) = (b, -1)$$

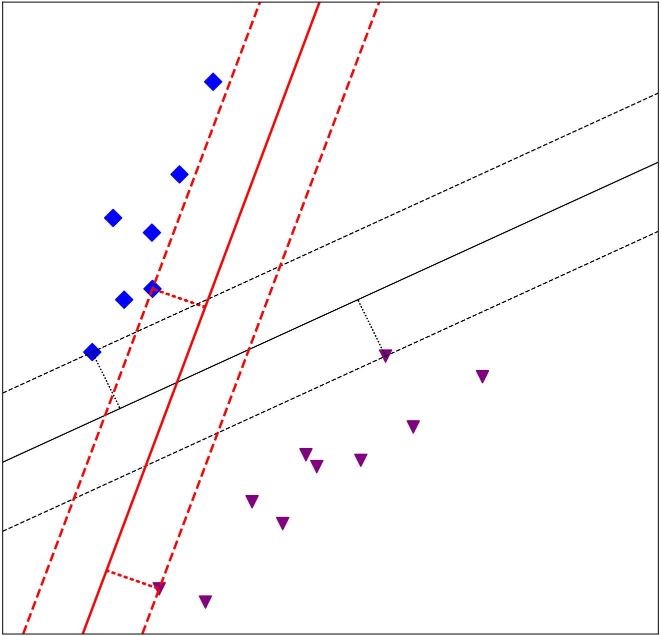
  where b is the slope. A vector parallel to the line is

$$\mathbf{v} = (1, b)$$

  Hence $\mathbf{v} \cdot \mathbf{w} = 0$. $\square$

Support Vectors

- Let η be a real number. Sweep the vector $\eta \mathbf{w}$ along the line of separation.
- At some point(s) $\eta \mathbf{w}$ will point to a data point on either side of the line. Find
$$M = \min_j |\eta \mathbf{w}|$$
- This distance is called the **support** and the vector is the **support vector**
- Find the line with the minimum total support



Python Example - Car Cylinder Classification (1/5)

Read auto-mpg data

```
import pandas as pd

data=pd.read_fwf("https://archive.ics.uci.edu/ml/
   machine-learning-databases/auto-mpg/auto-mpg.data",
   header=None,na_values="?")
data.columns=("mpg","cyl","displ","hp","weight","accel",
  "model","origin","carname")
data = data.dropna(axis=0)
```

Python Example - Car Cylinder Classification (1/5)

Read auto-mpg data

```
import pandas as pd

data=pd.read_fwf("https://archive.ics.uci.edu/ml/
   machine-learning-databases/auto-mpg/auto-mpg.data",
   header=None,na_values="?")
data.columns=("mpg","cyl","displ","hp","weight","accel",
  "model","origin","carname")
data = data.dropna(axis=0)
```

5 features, 3 classes

```
import numpy as np
cars=np.array(data[["cyl","mpg","displ","hp","weight",
  "accel"]])
cars=np.array([line for line in cars
   if line[0] in [4,6,8]])
Y=cars[:,0]/2-2
X=cars[:,1:]
```

Python Example - Car Cylinder Classification (2/5)

Find principal components

```
from sklearn.decomposition import PCA
pca=PCA(n_components=5)
pca.fit(X)
comps=pca.components_
explain=pca.explained_variance_ratio_
for comp, frac in zip(comps,explain):
    print(round(100*frac,4),"percent:", np.round(comp,3))
```

```
99.7651 percent: [-0.008  0.114  0.039  0.993 -0.001]
0.1985 percent: [-0.018  0.943  0.307 -0.121 -0.036]
0.0337 percent: [-0.027 -0.312  0.947 -0.002 -0.074]
0.0023 percent: [ 0.999  0.009  0.03   0.005 -0.02 ]
0.0004 percent: [ 0.018  0.012  0.082 -0.003  0.996]
```

Two components are sufficient.

Python Example (3/5) -Transform and Scale

Transform, 2 components $\mathbf{X} \to \mathbf{P}$

```
pca=PCA(n_components=2)
pca.fit(X)
P=pca.transform(X)
```

Scale, $\mathbf{P} \to \mathbf{Q}$

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
Q=scaler.fit_transform(P)
```

Python Example (4/5) - Function to Evaluate

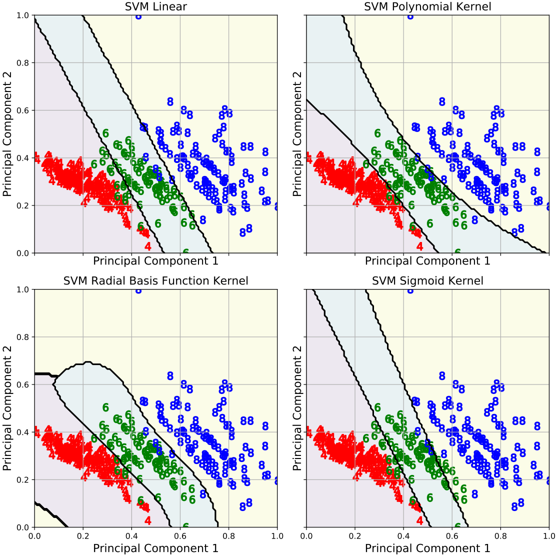
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

def svm_evaluate(classifier, X, Y, N):
    errs=[]
    for j in range(N):
        XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
        classifier.fit(XTRAIN,YTRAIN)
        YP=classifier.predict(XTEST)
        errs.append(1-accuracy_score(YTEST,YP))
    return(np.mean(errs),np.std(errs))
```

Python Example - Results (5/5)

```
print("Linear Kernel     mean error=%7.5f sd=%7.5f" \
      %svm_evaluate(SVC(kernel="linear",gamma=10),
      Q, Y,100))
print("Polynomial Kernel mean error=%7.5f sd=%7.5f" \
      %svm_evaluate(SVC(kernel="poly", degree=3, gamma=10,
      coef0=3), Q, Y,100))
print("RBF Kernel        mean error=%7.5f sd=%7.5f" \
      %svm_evaluate(SVC(kernel="rbf",gamma=20), Q, Y,100))
print("Sigmoid Kernel    mean error=%7.5f sd=%7.5f" \
      %svm_evaluate(SVC(kernel="sigmoid", gamma=1,
      coef0=0), Q, Y,100))
```

```
Linear Kernel     mean error=0.05474 sd=0.02003
Polynomial Kernel mean error=0.02835 sd=0.01626
RBF Kernel        mean error=0.02804 sd=0.01624
Sigmoid Kernel    mean error=0.07021 sd=0.02703
```

SVM Decision Boundary - Different Kernels



References

1. MPG data from: Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.