Lecture 6: Non-Linear Regression and SVM

Gonzalo De La Torre Parra, Ph.D.

Fall 2021

Nonlinear Regression

- Suppose you are given training data $(x_1, y_1), \ldots, (x_n, y_n)$. Assume both variables x and y are real numbers. Then, the linear regression fit between x and y is obtained by solving the following optimization problem:

$$\min_{a,b} \sum_{i=1}^{n} (y_i - ax_i - b)^2.$$

- **Bold assumption or approximation**: By seeking a linear fit between the variables x and y, we are either assuming that the true relationship between x and y is linear. Or we are assuming that the true relationship may be nonlinear, but we are only looking for a linear approximation to that relationship.
- **Elementary nonlinear regression**: If the linear fit leads to terrible predictions, then we can look for possibly nonlinear fit. The framework of linear regression is actually more general than it seems: it can allow us to obtain nonlinear fit as well. For example, the following two problems will give us a nonlinear fit:

$$\min_{a,b} \sum_{i=1}^{n} (y_i - a\frac{1}{x_i} - b)^2$$

$$\min_{a,b,c} \sum_{i=1}^{n} (y_i - ax_i - bx_i^2 - c)^2.$$

- **Nonlinear regression using linear regression**: More generally, we can transform the data using maps $\phi_1, \phi_2, \ldots, \phi_d$, etc and solve

$$\min_{\beta=\beta_1,\ldots,\beta_d} \sum_{i=1}^{n} (y_i - \beta_1\phi_1(x_i) - \beta_2\phi_2(x_i) - \cdots - \beta_d\phi_d(x_i))^2.$$

Here out $\mathbb{X}$ matrix will be

$$\mathbb{X} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_d(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_d(x_2) \\ \vdots & & & \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_d(x_n) \end{bmatrix}.$$

- **Example 1**: For example, for the problem

$$\min_{a,b} \sum_{i=1}^{n} (y_i - a\frac{1}{x_i} - b)^2,$$

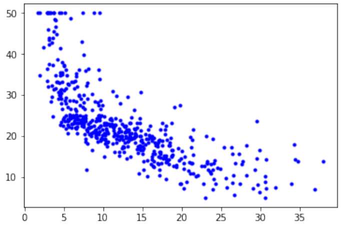
$d = 2$, $\phi_1(x) = \frac{1}{x}$, $\phi_2(x) = 1$.

- **Example 2**: For example, for the problem

$$\min_{a,b,c} \sum_{i=1}^{n} (y_i - ax_i - bx_i^2 - c)^2,$$

$d = 3$, $\phi_1(x) = x$, $\phi_2(x) = x^2$, and $\phi_3(x) = 1$.

- **Boston Housing Data**:



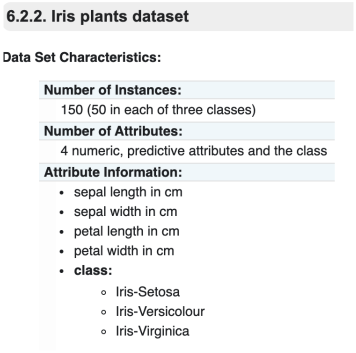
Support Vector Machines

- **Supervised Classification**: Recall the problem of classification in which we are given n pairs of data points:

$$(x_1, y_1), \ldots, (x_n, y_n).$$

  We have to learn a relationship between x and y so as to predict y from x. In the problem of classification, y is a **discrete** variable.

- **Example: Iris classification**



6.2.2. Iris plants dataset

Data Set Characteristics:

Number of Instances:
150 (50 in each of three classes)

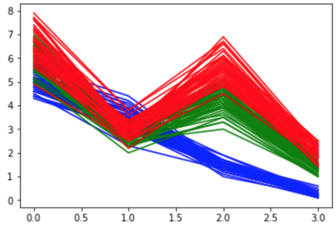
Number of Attributes:
4 numeric, predictive attributes and the class

Attribute Information:
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- **class:**
  - Iris-Setosa
  - Iris-Versicolour
  - Iris-Virginica

IRIS Data from Scikit-learn Package

- **Plots of the** 150 **features each of length** 4**:**

```
(150, 4) (150,)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

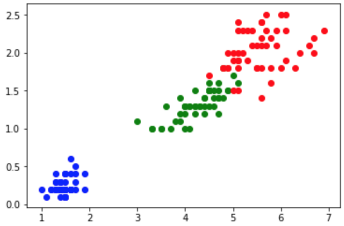


- **2-D Feature extraction**: We want to pick a low-dimensional representation so that we can plot the learned SVM model and appreciate the theory.
- **The third and fourth components have clear red, blue and green non-overlapping bands.**

- **Extracting the third and fourth components for each of the** 150 **training points and creating a scatter plot**:

```
X0 = iris.data[0:50, 2:4]
X1 = iris.data[50:100, 2:4]
X2 = iris.data[100:150, 2:4]
y = iris.target[0:100]
print(X0.shape, X1.shape, X2.shape)
plt.scatter(X0[:,0], X0[:,1], color='b')
plt.scatter(X1[:,0], X1[:,1], color='g')
plt.scatter(X2[:,0], X2[:,1], color='r')
plt.show()
```

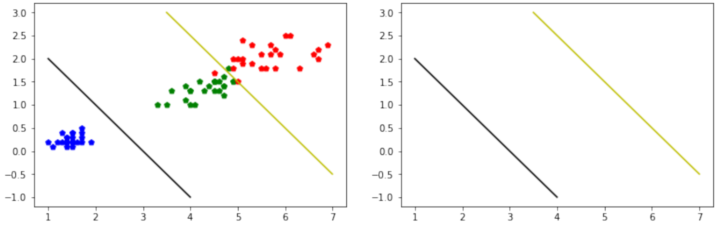
```
(50, 2) (50, 2) (50, 2)
```



- We want to understand how classification, especially SVM, works using 2-D data. Note that the actual classification will be performed using all the four features. The 2-D representation is used only for understanding.
- **Let us imagine we only have a 2D dataset.**

How to Train a Classifier Using 2D Data?

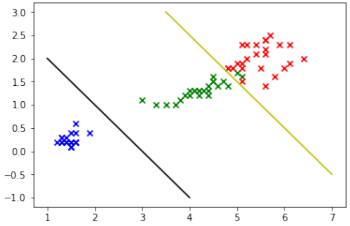
- **Drawing Lines**: One option for training is to 'draw' black and yellow lines between training points of different classes as shown in the figure. **We have used only the first** 25 **points from each class**.



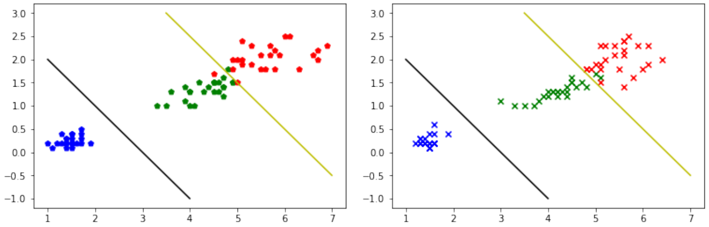
- **Inference or Classification**: Once we have drawn the lines the inference or classification can be done on new data as
  - If point below black line: choose class 0
  - If point above yellow line: choose class 2
  - If point between yellow and black line: choose class 1
- **The central problem in classification is to draw such lines or decision boundaries between data from different classes.**

Why Would We Succeed In Such an Approach?

- **Why do we think drawing such lines between points from different classes is such a great idea or will lead to good prediction?**
- Recall that we drew the lines by looking at the first 25 training points. Now, let us keep the black and yellow lines that we learned and now superimpose the next 25 points from each classes. **Note that the points in the figure are the next** 25 **points from each of the classes**.



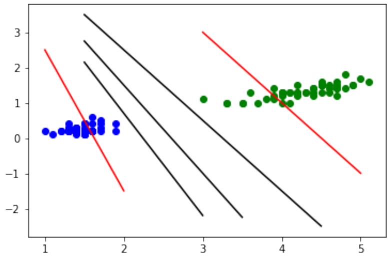
- A comparison reveals that
  - the blue crosses are close to the blue pentagons.
  - the green crosses are close to the green pentagons.
  - the red crosses are close to the red pentagons.



- If test data for a class is close to the old data or training data from the same class, the decision boundaries should successfully separate the points in the test data well.
- **How to draw the 'best' line that will lead to the best prediction?** The Support Vector Machine (SVM) method is one way to draw a good decision boundary.

Two-Class Problem

- **Two-Class Problem**: In order to understand the SVM method better, we just consider the first two classes: Class 0 and Class 1.
- **If we wish to draw lines between points from the two classes, which line should we draw? We are now thinking about an automatic ways of doing things.**



- In the figure above, black lines are better than red lines as black lines completely separate the data, but the red lines do not.
- **Conclusion: Black lines are better**.