T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

CSE4088 Introduction to Machine Learning

Homework #4

Oğuzhan BÖLÜKBAŞ

150114022

- 15.12.2019-

SVM

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the **kernel** trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data is unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The **support vector clustering** algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabelled data, and is one of the most widely used clustering algorithms in industrial applications.

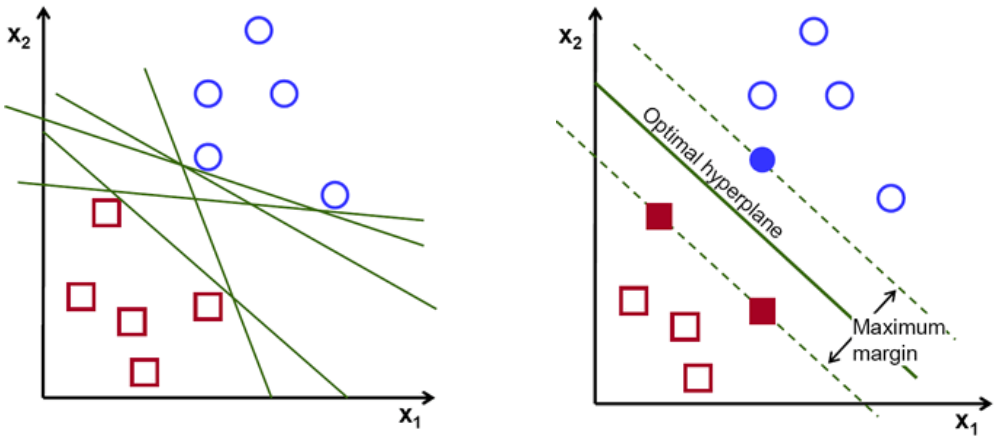

Figure 1: Possible Hyperplanes

SVM with Soft Margins

- **Polynomial Kernels**

$$\text{polynomial kernel } K(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m)^Q$$

Figure 1: Polynomial kernel formula

```
kernel = (1.0 + np.dot(xn[row_idx].T, xn[col_idx])) ** self.Q
```

Figure 2: Implementation polynomial kernel formula statement from my code

Problem 2

The **highest** in-sample error (E_{in}) is "0 vs all" with E_{in} = 0.106%. So, the answer in multiple-choice answers is **a**.

```
Problem 2, 3 and 4:

C = 0.01 and Q = 2, (zero_vs_all) Ein = 0.106%, and # of support vectors = 2179.

C = 0.01 and Q = 2, (one_vs_all) Ein = 0.014%, and # of support vectors = 386.

C = 0.01 and Q = 2, (two_vs_all) Ein = 0.1%, and # of support vectors = 1970.

C = 0.01 and Q = 2, (three_vs_all) Ein = 0.09%, and # of support vectors = 1950.

C = 0.01 and Q = 2, (four_vs_all) Ein = 0.089%, and # of support vectors = 1856.

C = 0.01 and Q = 2, (five_vs_all) Ein = 0.076%, and # of support vectors = 1585.

C = 0.01 and Q = 2, (six_vs_all) Ein = 0.091%, and # of support vectors = 1893.

C = 0.01 and Q = 2, (seven_vs_all) Ein = 0.088%, and # of support vectors = 1704.

C = 0.01 and Q = 2, (eight_vs_all) Ein = 0.074%, and # of support vectors = 1776.

C = 0.01 and Q = 2, (nine_vs_all) Ein = 0.088%, and # of support vectors = 1978.
```

Figure 3: Output of problem 2

Problem 3

The **lowest** in-sample error (E_{in}) is "1 vs all" with E_{in} = 0.014%. So, the answer in multiple-choice answers is **a**.

```
Problem 2, 3 and 4:

C = 0.01 and Q = 2, (zero_vs_all) Ein = 0.106%, and # of support vectors = 2179.

C = 0.01 and Q = 2, (one_vs_all) Ein = 0.014%  and # of support vectors = 386.

C = 0.01 and Q = 2, (two_vs_all) Ein = 0.1%, and # of support vectors = 1970.

C = 0.01 and Q = 2, (three_vs_all) Ein = 0.09%, and # of support vectors = 1950.

C = 0.01 and Q = 2, (four_vs_all) Ein = 0.089%, and # of support vectors = 1856.

C = 0.01 and Q = 2, (five_vs_all) Ein = 0.076%, and # of support vectors = 1585.

C = 0.01 and Q = 2, (six_vs_all) Ein = 0.091%, and # of support vectors = 1893.

C = 0.01 and Q = 2, (seven_vs_all) Ein = 0.088%, and # of support vectors = 1704.

C = 0.01 and Q = 2, (eight_vs_all) Ein = 0.074%, and # of support vectors = 1776.

C = 0.01 and Q = 2, (nine_vs_all) Ein = 0.088%, and # of support vectors = 1978.
```

Figure 4: Output of problem 3

Problem 4

The difference of the number of support vectors between "0 vs all" and "1 vs all" is 1793 (2179 - 386). So, the answer in multiple-choice answers is **c**.

```
Problem 2, 3 and 4:

C = 0.01 and Q = 2, (zero_vs_all) Ein = 0.106%, and # of support vectors = 2179.

C = 0.01 and Q = 2, (one_vs_all) Ein = 0.014%, and # of support vectors = 386

C = 0.01 and Q = 2, (two_vs_all) Ein = 0.1%, and # of support vectors = 1970.

C = 0.01 and Q = 2, (three_vs_all) Ein = 0.09%, and # of support vectors = 1950.

C = 0.01 and Q = 2, (four_vs_all) Ein = 0.089%, and # of support vectors = 1856.

C = 0.01 and Q = 2, (five_vs_all) Ein = 0.076%, and # of support vectors = 1585.

C = 0.01 and Q = 2, (six_vs_all) Ein = 0.091%, and # of support vectors = 1893.

C = 0.01 and Q = 2, (seven_vs_all) Ein = 0.088%, and # of support vectors = 1704.

C = 0.01 and Q = 2, (eight_vs_all) Ein = 0.074%, and # of support vectors = 1776.

C = 0.01 and Q = 2, (nine_vs_all) Ein = 0.088%, and # of support vectors = 1978.
```

Figure 5: Output of problem 4

Problem 5

The statement **"The number of support vectors goes down when C goes up**." is **false** because the number of support vector does not always go down.

The statement **"The number of support vectors goes up when C goes up."** is **false** because the number of support vector does not go up.

The statement **"E_{out} goes down when C goes up."** is **false** because E_{out} does not always go down when C goes up.

The statement **"Maximum C achieves the lowest E_{in}."** is **false** because for C = 1 we see the minimum E_{in}. So, the answer in multiple-choice answers is **d**.

```
Problem 5:
C = 0.001 and Q = 2, "one_vs_five" Ein = 0.004%,  Eout = 0.017% and # of support vectors = 76.

C = 0.01 and Q = 2, "one_vs_five" Ein = 0.004%,  Eout = 0.019% and # of support vectors = 34.

C = 0.1 and Q = 2, "one_vs_five" Ein = 0.004%,  Eout = 0.019% and # of support vectors = 24.

C = 1.0 and Q = 2, "one_vs_five" Ein = 0.003%,  Eout = 0.019% and # of support vectors = 24.
```

Figure 6: Output of problem 5

Problem 6

The statement **"When C = 0.0001, E_{in} is higher at Q = 5." is false** because E_{in} is lower at Q = 5.

The statement **"When C = 0.01, E_{in} is higher at Q = 5." Is false** because both E_{in} is equal (0.004%).

The statement **"When C = 1, E_{out} is lower at Q = 5."** is **false** because E_{out} is higher at Q = 5.

The statement **"When C = 0.001, the number of support vectors is lower at Q = 5"** is **true.** So, the answer in multiple-choice answers is **b**.

```
Problem 6:

C = 0.0001 and Q = 2, "one_vs_five" Ein = 0.009%,  Eout = 0.017% and # of support vectors = 236.

C = 0.0001 and Q = 5, "one_vs_five" Ein = 0.004%,  Eout = 0.019% and # of support vectors = 26.

C = 0.001 and Q = 2, "one_vs_five" Ein = 0.004%,  Eout = 0.017% and # of support vectors = 76.

C = 0.001 and Q = 5, "one_vs_five" Ein = 0.004%,  Eout = 0.021% and # of support vectors = 25.

C = 0.01 and Q = 2, "one_vs_five" Ein = 0.004%,  Eout = 0.019% and # of support vectors = 34.

C = 0.01 and Q = 5, "one_vs_five" Ein = 0.004%,  Eout = 0.021% and # of support vectors = 23.

C = 0.1 and Q = 2, "one_vs_five" Ein = 0.004%,  Eout = 0.019% and # of support vectors = 24.

C = 0.1 and Q = 5, "one_vs_five" Ein = 0.003%,  Eout = 0.019% and # of support vectors = 25.

C = 1.0 and Q = 2, "one_vs_five" Ein = 0.003%,  Eout = 0.019% and # of support vectors = 24.

C = 1.0 and Q = 5, "one_vs_five" Ein = 0.003%,  Eout = 0.021% and # of support vectors = 21.
```

Figure 7: Output of problem 6

- **Cross Validation**

Problem 7 and 8

C = 0.001 is the most selected often 30 times out of 100 runs. So, the answer in multiple-choice answers is **b** for question 7.

Average of cross validation error $\mathbf{E_{cv}}$ is **0.005**. So, the answer in multiple-choice answers is **c** for question 8.

```
Problem 7 and problem 8:

C = 0.001 is selected the most often, (30/100 runs)
Average  Ecv = 0.005
```

Figure 8: Output of problem 7 and 8

- **RBF Kernel**

$$\text{radial basis function (RBF) kernel } K(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(-||\mathbf{x}_n - \mathbf{x}_m||^2\right)$$

Figure 9: RBF kernel formula

```
if self.rbf:
    kernel = np.exp(-1.0 * (np.linalg.norm(xn[row_idx] - xn[col_idx]) ** 2))
```

Figure 10: Implementation RBF kernel formula statement from my code

Problem 9 and 10

The lowest in-sample error E_{in} (0.001%) is when C = 10^6. So, the answer in multiple-choice answers is **e** for question 9.

The lowest in-sample error E_{out} (0.019%) is when C = 100. So, the answer in multiple-choice answers is **c** for question 10.

```
Problem 9 and problem 10:

C = 0.01: "one_vs_five"  Ein = 0.004% and  Eout = 0.024%

C = 1.0: "one_vs_five"  Ein = 0.004% and  Eout = 0.021%

C = 100.0: "one_vs_five"  Ein = 0.003% and  Eout = 0.019%

C = 10000.0: "one_vs_five"  Ein = 0.003% and  Eout = 0.024%

C = 1000000.0: "one_vs_five"  Ein = 0.001% and  Eout = 0.024%
```

Figure 11: Output of problem 9 and 10