SVM

Federico Gianno

***Abstract*—This homework consists into the application of SVM: the purpose is to plot data item as a point in n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, I perform classification by finding the hyper-plane that differentiate the two classes very well.**

I. Introduction

Support Vector Machines (SVMs) are **supervised learning models** with associated learning algorithms that analyze 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 [1].

II. Theory

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in.

- Linear Classification
  A data point is viewed as a *p-dimensional* vector, and we want to know whether we can separate such points with a *(p-1)-dimensional* hyper-plane.
  There are many hyper-planes that might classify the data. One reasonable choice as the best hyper-plane is the one that represents the largest separation, or margin, between the two classes.
- Non-Linear Classification
  Nonlinear classifiers consist applying the kernel trick to maximum-margin hyper-planes.
  The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyper plane in a transformed feature space.

III. Procedures

This section briefly describes a python code implementation reporting several **snippet code** with the corresponding explanations and motivations.

1) Requirements
   - numpy (imported as np)
   - scikit-learn
   - matplotlib (imported as plt)
2) Data preparation
   First of all I need a data-set, *Iris* can provide me a good one. Then I have to create two arrays: one for values and one for labels.

```
iris = datasets.load_iris()
# take the first two features.
x = iris.data[:, :2]
y = iris.target  # label
```

3) Split data into train, test and validation set
   I cannot choose the optimal values based on my test set because of in real problems I won't have access to it. A good way to evaluate a method is to split the data into 3 parts:
   - Train set (50%)
   - Validation set (20%)
   - Test set (30%)

   I'll choose some parameters and I'll train my model on the training set; then I'll evaluate the performances on the validation set; finally, once discovered the parameters which work best on the validation set, I just apply the same model on the test set.
4) SVC and fit model
   At this point I can choose the parameters of the SVM:
   - **Kernel**: the function on which the SVM is build on
   - **C**: it controls the cost of missclassification on the training data
   - **Gamma**: it controls the tradeoff between error due to bias and variance in the model

```
# for example
clf = svm.SVC(kernel='rbf', C=c, gamma=g)
clf.fit(x_train, y_train)
```

   The *effectiveness* of SVM will depend on these parameters
5) Decision boundaries
   Now it's time to plot the data and the decision boundaries. The next step is to evaluate the method on the validation set and check the corresponding success rate. The role of the validation set is to find the optimum for the parameters of the algorithm.

```
clf_predictions = clf.predict(x_validate)
clf_sc = clf.score(x_validate, y_validate)
```

   After doing it many times with different pair of *C* and *gamma*, I chose the best pair and I evaluated the test set.

IV. Linear SVM

For different values of C, as anticipated in section III, I trained a *linear* SVM on the training set and plotted the data and the decision boundaries. For each step I plot the data and the decision boundaries.

As you can see on Fig. 1 boundaries change and it's due to the changing of *C*: it tells to the SVM optimization how much you want to avoid missclassifying at each training example. For large values of C, the optimization will choose a smaller-margin hyper-plane if that hyper-plane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyper-plane, even if that hyper-plane missclassifies more points. So:

- Small C makes the cost of missclassificaiton low (”soft margin”), thus allowing more of them for the sake of wider ”cushion”
- Large C makes the cost of missclassification high (”hard margin”), thus forcing the algorithm to explain the input data stricter and potentially overfit

The goal is to find the balance between ”not too strict” and ”not too loose”. Cross-validation (accuracy on validation set) and resampling, along with grid search, are good ways to finding the best C.

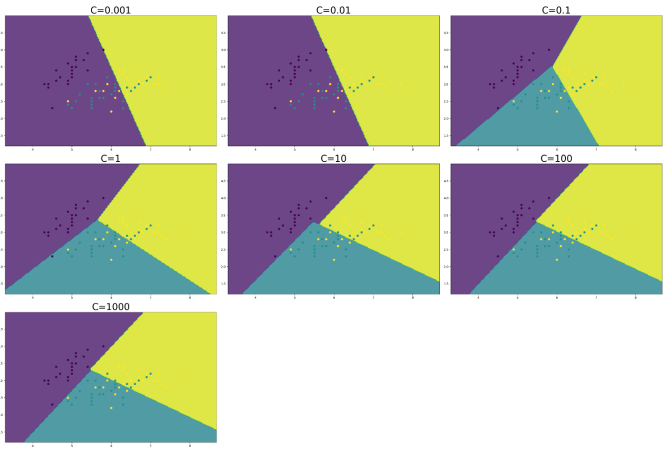

Fig. 1. *Linear kernel*. C values are, respectively, from left to right: 10^-3, 10^-2, 10^-1, 10, 10^2, 10^3

Then, I plot in a graph that shows how the accuracy varies, on the validation set, when changing C.

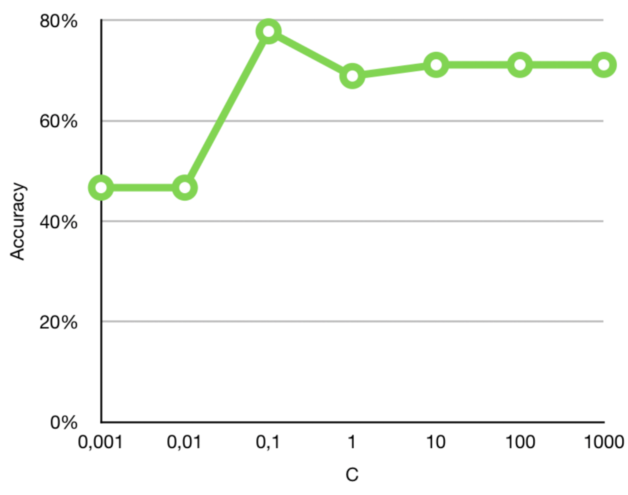

Fig. 2. *Linear kernel*. Correlation between C and Accuracy

As you can notice the best accuracy is given for $C = 0.1$ and it is equal to 77.778%. So, finally, I chose the best value of C and evaluated the model on the test set. Fig. 3 shows the data and decision boundaries.

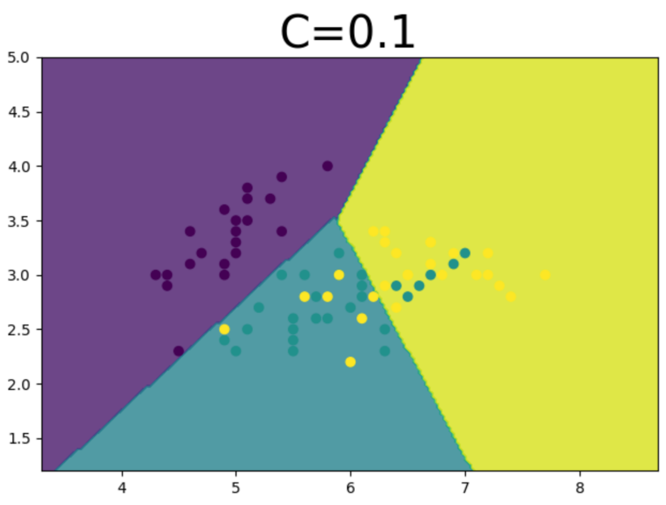

Fig. 3. *Linear kernel*. Best value of $C = 10^-1$ evaluated on test set

On this attempt I found a test accuracy that is equal to 83.333% and it the highest one. Consideration in section VII.

V. RBF Kernel

I did the same steps describes on IV but, this time, using a **Radial Basis Function** kernel (*RBF*).

This time both C and $gamma$ will change.

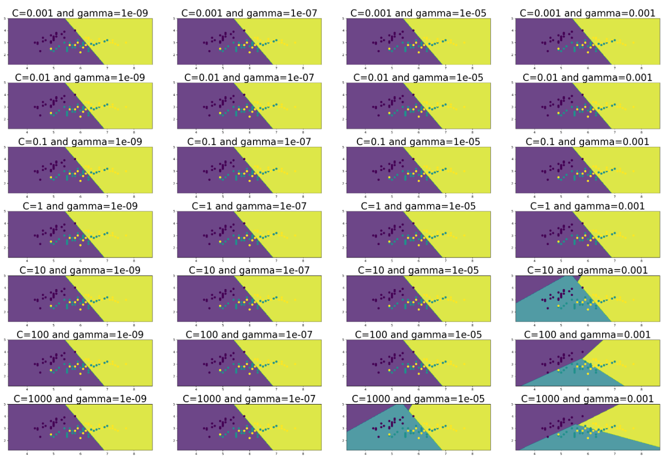

Fig. 4. *RBF kernel*. In each row, from left to right, gamma takes the following values: 10^-9, 10^-7, 10^-5 and 10^-3. C value is set for each row and grow row after row, from top to bottom: 10^-3, 10^-2, 10^-1, 10^0, 10, 10^2, 10^3

I found that the best value of C and gamma, in order to obtain the best accuracy on the validation set, is: $C = 100$ and $gamma = 0.001$, with an accuracy of 80.000%.

So, as done before, with the best value of C and $gamma$, I evaluated the parameters on the test set. Fig. 5 shows the relative data and decision boundaries.

I found an accuracy that is equal to 83.333%. Consideration in section VII.

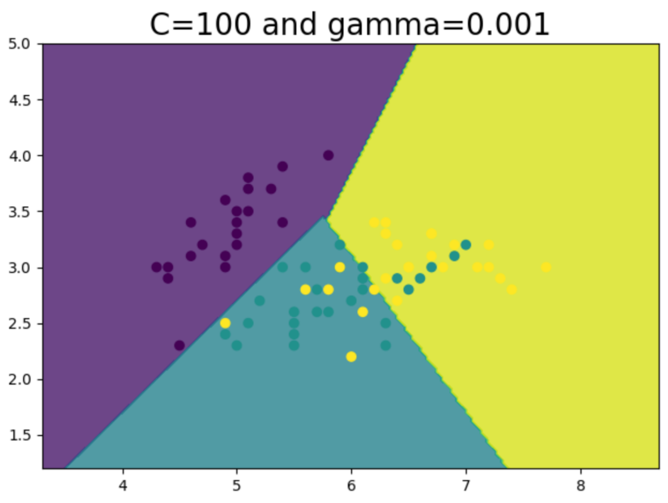

Fig. 5. *RBF kernel*. Best value of $C = 10^2$ and $gamma = 10^-3$ evaluated on test set

VI. K-Fold

K-Fold Cross Validation is useful when the training set is small. In this case the validation set is a subset of the training data and is data we can no longer use to train our model.

Now, I have not yet a training, validation and test set but only a training (70%) and test (30%) set.

I repeated the same operation of III, but this time performing 5-fold validation.

Th K-Fold split the data into K consecutive folds and at each cycle one fold is used as validation set while the other remaining folds are the training set.

This time I evaluated the method on the training set (test set no more exist) and I looked for the best couples of C and $gamma$ in order to maximize the accuracy. So, I found that the best value are: $C = 100$ and $gamma = 0.001$.

At this point I evaluate the parameters on the test set and I got 80% accuracy. Consideration in section VII.

VII. Accuracy consideration

Sometimes the test accuracy is lower than the validation accuracy, but in others some cases I got a test accuracy that is lower than the validation accuracy by $10\% - 15\%$.

Typically I should have validation accuracy lower than the test accuracy.

In order to estimate how well the model have to be trained and to estimate model properties, a good approach is to split (randomly) the data into three parts (as I did) in which each part have a different role:

- Training set: it is used to build up the prediction algorithm. The algorithm tries to tune itself to the quirks of the training data sets.
- Validation set: it is used to compare the performances of the prediction algorithms that were created based on the training set. I'll choose the parameters of the algorithm that has the best performance.
- Test set: I applied the chosen prediction algorithm on the test set.

Finally, the accuracy of the model on the test data gives a realistic estimate of the performance of the model on completely unseen data and in order to confirm the actual predictive power of the model.

So, the risk is to **overfit** the validation set during the loop which has the role to chose the best model based on best accuracy using the validation set. Then, in this case, the test set accuracy is lower than the validation accuracy.

On the other hand, sometimes could be possible that the accuracy of the test accuracy is higher than the validation accuracy. This is the expected result because the *validation set* is usually used for parameter selection and to avoid overfitting.

VIII. More

I also plotted the **validation accuracy** for the RBF kernel: I increased the value of both C and $gamma$ in order to obtain a better plot that shows a how the validation accuracy change at the changing of both C and gamma.

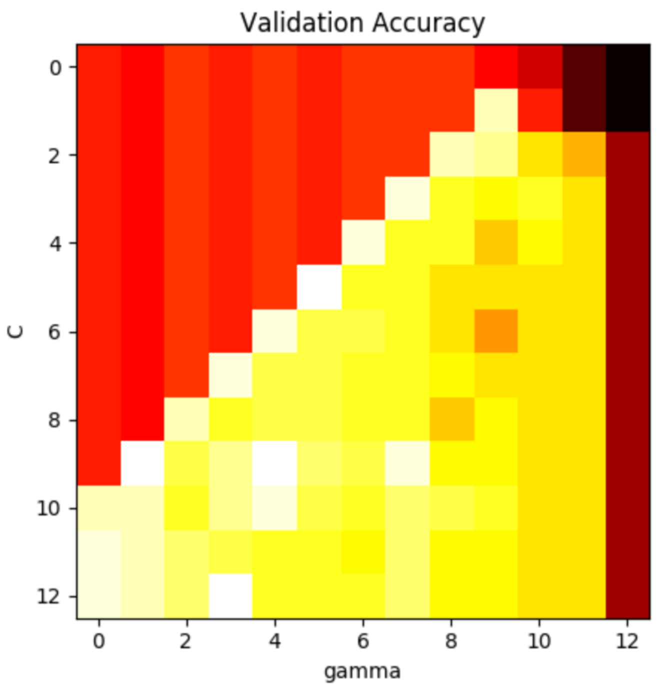

Fig. 6. *RBF kernel*. Validation accuracy: C values go from 10^-2 to 10^9 and gamma values go from 10^-9 to 10^3

IX. Conclusions

SVM is used for text classification tasks such as category assignment, detecting spam and sentiment analysis. It is also commonly used for image recognition challenges, performing particularly well in aspect-based recognition and color-based classification. SVM also plays a vital role in many areas of handwritten digit recognition, such as postal automation services [2].

- Pros
  - It works well on smaller cleaner data-sets
  - It is effective in high dimensional spaces

  - It uses a subset of training points in the decision function (support vectors), so it is also memory efficient
- Cons
  - It does not perform well, when we have large data set because the required training time is higher
  - It also does not perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library

REFERENCES

[1] From Wikipedia, Support Vector Machine. https://en.wikipedia.org/wiki/Support_vector_machine

[2] Support Vector Machines - What are they? By Noel Bambrick, AYLIEN. https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html