Machine Learning Introduction

Supervisor
PD Dr. Friedhelm Schwenker
Institute of Neural Information Processing
Ulm University
Students
Abdelrahman Mahmoud And Mohamed Ashry

April 1, 2020

Support Vector Machines (SVM)?

- (SVM) is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression, and even outlier detection. It is one of the most popular models in Machine Learning
- SVMs are particularly well suited for classification of complex but small- or medium-sized datasets .
- This chapter will explain the core concepts of SVMs, how to use them, and how they work.

the core concepts of SVMs

- Linear SVM Classification
- Nonlinear SVM Classification
- SVM Regression

1) Linear SVM Classification

The fundamental idea behind SVMs is best explained with some pictures. Figure 5-1

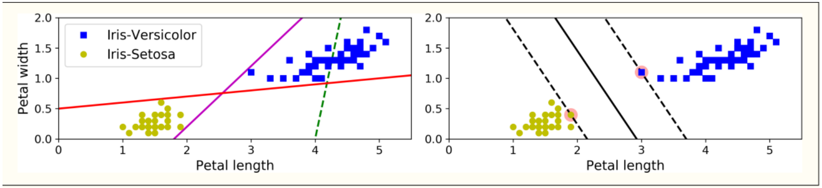


Figure 5-1. Large margin classification

You can think of an SVM classifier as fitting the widest possible street (represented by the parallel dashed lines) between the classes. This is called large margin classification.

Hard Margin Classification

- If we strictly impose that all instances be off the street and on the right side, this is called hard margin classification
- here are two main issues with hard margin classification. First, it only works if the data is linearly separable, and second it is quite sensitive to outliers .

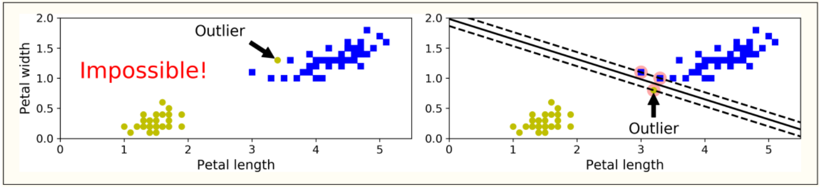


Figure 5-3. Hard margin sensitivity to outliers

Soft Margin Classification

The objective is to find a good balance between keeping the street as large as possible and limiting the margin violations (i.e., instances that end up in the middle of the street or even on the wrong side). This is called so margin classification.

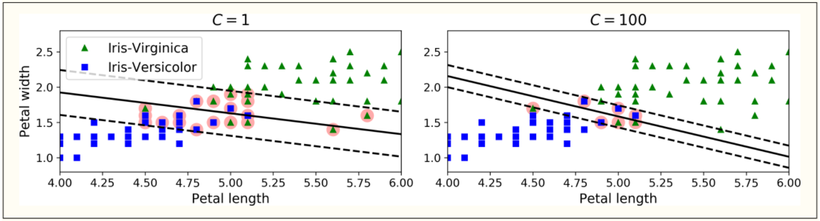


Figure 5-4. Large margin (left) versus fewer margin violations (right)

2) Nonlinear SVM Classification

- Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable.
- One approach to handling nonlinear datasets is to add more features, such as polynomial features (as you did in Chapter 4)in some cases this can result in a linearly separable dataset. Consider the left plot in Figure 5-5:

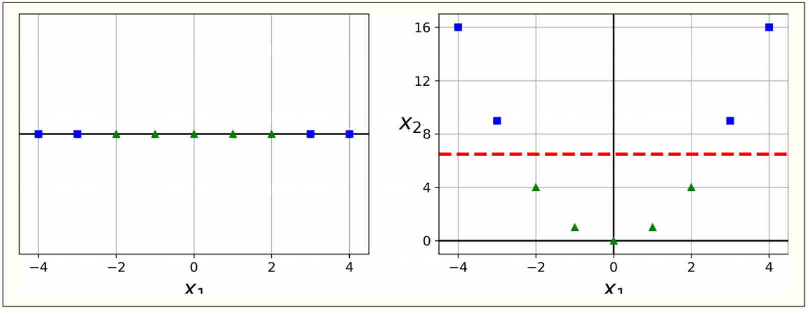


Figure 5-5. Adding features to make a dataset linearly separable

Solutions !!!!

- Polynomial Kernels
- Adding Similarity Features
- Gaussian RBF Kernel
- Computational Complexity

Polynomial Kernel

- Adding polynomial features is simple to implement and can work great with all sorts of Machine Learning algorithms (not just SVMs), but at a low polynomial degree it cannot deal with very complex datasets, and with a high polynomial degree it creates a huge number of features, making the model too slow.
- Fortunately, when using SVMs you can apply an almost miraculous mathematical technique called the kernel trick (it is explained in a moment). It makes it possible to get the same result as if you added many polynomial features, even with very highdegree polynomials, without actually having to add them. So there is no combinato- rial explosion of the number of features since you don't actually add any features.

```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

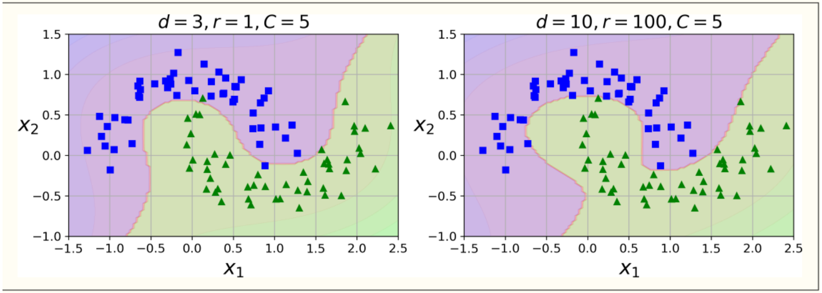


Figure 5-7. SVM classifiers with a polynomial kernel

Adding Similarity Features

- Another technique to tackle nonlinear problems is to add features computed using a similarity function that measures how much each instance resembles a particular landmark.
- The simplest approach is to create a landmark at the location of each and every instance in the dataset . This creates many dimensions and thus increases the chances that the transformed training set will be linearly separable. The downside is that a training set with m instances and n features gets transformed into a training set with m instances and m features (assuming you drop the original features).
- If your training set is very large, you end up with an equally large number of features.

Gaussian RBF Kernel

- Just like the polynomial features method, the similarity features method can be useful with any Machine Learning algorithm, but it may be computationally expensive to compute all the additional features, especially on large training sets. However, once again the kernel trick does its SVM magic: it makes it possible to obtain a similar result as if you had added many similarity features, without actually having to add them.

```
rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ])
rbf_kernel_svm_clf.fit(X, y)
```

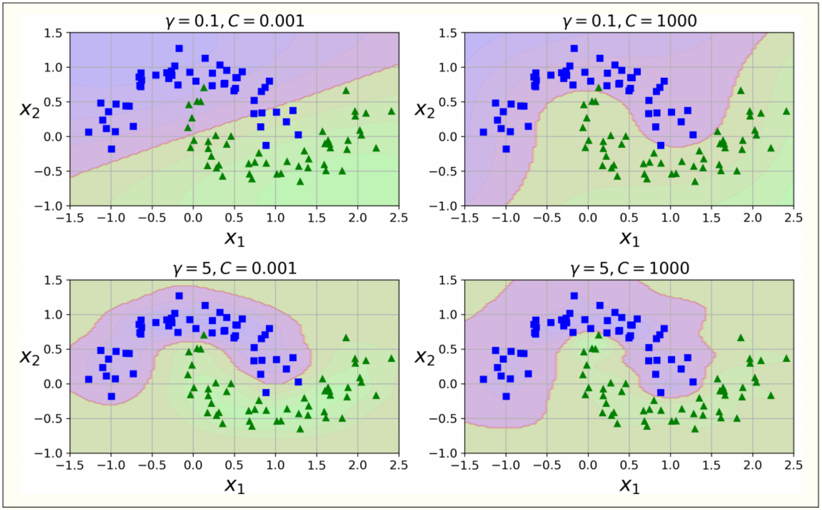
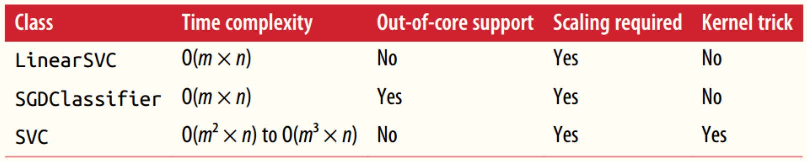


Figure 5-9. SVM classifiers using an RBF kernel

Computational Complexity

- The LinearSVC class is based on the liblinear library, which implements an optimized algorithm for linear SVMs.1 It does not support the kernel trick, but it scales almost linearly with the number of training instances and the number of features: its training time complexity is roughly O(m x n).
- The algorithm takes longer if you require a very high precision. This is controlled by the tolerance hyperparameter (called tol in Scikit-Learn). In most classification tasks, the default tolerance is fine.

Table 5-1. Comparison of Scikit-Learn classes for SVM classification



Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes

3) SVM Regression

- As we mentioned earlier, the SVM algorithm is quite versatile: not only does it sup- port linear and nonlinear classification, but it also supports linear and nonlinear regression. The trick is to reverse the objective: instead of trying to fit the largest pos- sible street between two classes while limiting margin violations.
- SVM Regression tries to fit as many instances as possible on the street while limiting margin violations (i.e., instances o the street). The width of the street is controlled by a hyperparame- ter .
- Figure 5-10 shows two linear SVM Regression models trained on some random linear data, one with a large margin ($= 1.5$) and the other with a small margin ($= 0.5$)

```
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5)
svm_reg.fit(X, y)
```

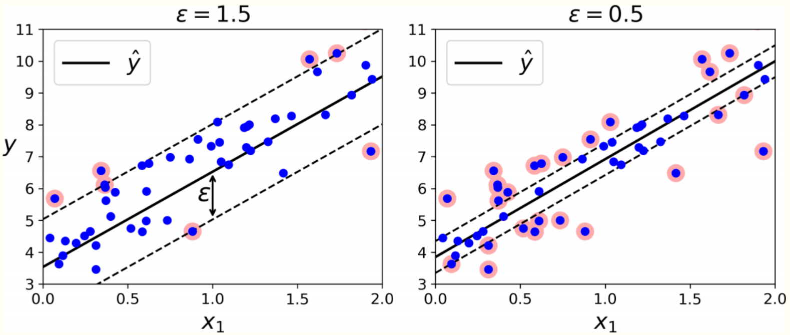


Figure 5-10. SVM Regression

Thank You !