Machine Learning - Exercise 6

SVM

Goal of the exercise

- In this exercise, we will use the SVM implementation of the *scikit-learn* library: http://scikit-learn.org
- *scikit-learn* is a machine learning framework written in Python (with C extensions for performance) allowing to use very easily and efficiently many ML algorithms, including linear classification/regression algorithms, SVM, PCA, clustering techniques and a lot of other state-of-the-art algorithms not seen in the course.
- It has a very good documentation and proposes many tutorials to discover new ML algorithms.
- Install it with:

```
pip2 install scikit-learn --user
```

Note: do not copy and paste, the dashes would not be right.

GUI for simple SVMs

- We will use a graphical interface developped by Peter Prettenhoer to study the influence of the choice of the kernel (and the associated parameters) on the classification performance for simple two-dimensional input data.
- The goal is to get an intuition of how SVM works, but of course SVM is only reasonable for higher-dimensional data.
- Run the provided script `svmgui.py`. It opens a 2D drawing area where you can add training examples with the mouse (left-click: positive class, right-click: negative class). Once the training set is defined, you can apply a binary classification SVM.

GUI for simple SVMs

Below the drawing area are options to the SVM algorithm:

- You can choose the kernel between *linear* (no kernel), *rbf* (gaussian) or *polynomial*.
- You can define the regularization parameter C (for the soft-margin classification, default value is 1).
- You can define the parameters of the kernel: *gamma* is the spatial extent of the kernel for both rbf and polynomial kernels, while *degree* is the degree of the polynome and *coef0* its offset.

With the notations used in the course, this would be:

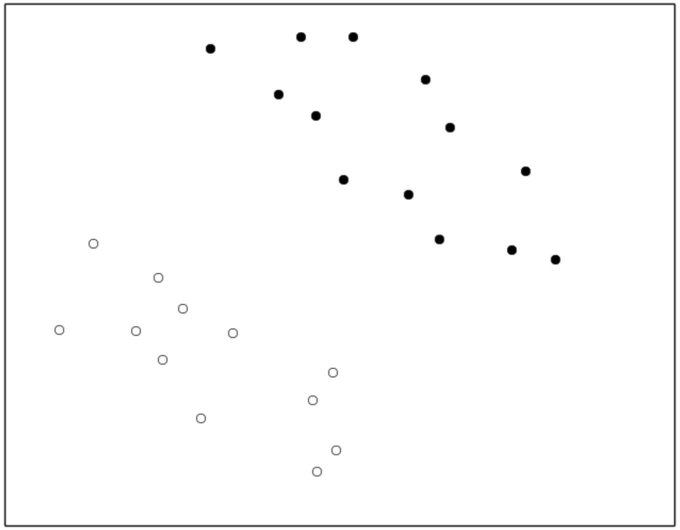
$$\mathrm{linear}(x,z)=\langle x\cdot z\rangle$$

$$\mathrm{rbf}(x,z)=\exp(-\gamma\cdot\langle x\cdot z\rangle^2)$$

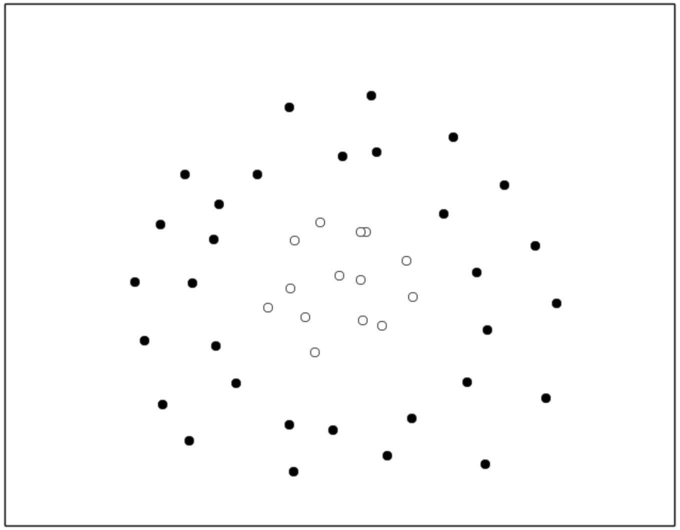
$$\mathrm{polynomial}(x,z)=(\gamma\cdot\langle x\cdot z\rangle+\mathrm{coef0})^d$$

When you change the value of one parameter, you need to press the *Fit* button to apply your changes. The goal of the exercise is to compare the performance of these three kernels on different data sets.

Possible datasets



Linear: $u^T v$ RBF: $\exp(-\gamma|u-v|^2)$ Poly: $(\gamma\, u^T v + r)^d$



Linear: $u^T v$ RBF: $\exp(-\gamma|u-v|^2)$ Poly: $(\gamma\, u^T v + r)^d$

Questions

1. Setup two classes easily and linearly separable. Apply the linear kernel with C = 1.0. Identify the decision function (hyperplane), the geometrical margin and the support vectors.
2. Add a new point far away from the the hyperplane (and correctly classified). Does it change something to the decision function and the number of support vectors.
3. Add a new point inside the functional margin ($\gamma<1$). How does it change the hyperplane and the support vectors? What effect does it have on the geometric margin?
4. Decrease the regularization parameter C until the added point has a slack variable ξ_i different from 0 (in decreasing powers of 10, such as 0.1, 0.01, etc). Which effect does it have on the geometrical margin?
5. Decrease C even further (0.00001 or so). What becomes the geometric margin? How good is the classification? Does it make sense to decrease to much C?
6. With the adequate value for C, add more points inside the functional margin. State the influence of regularization on the number of support vectors.

Questions

7. With C=1.0, apply the rbf and polynomial data on the linearly separable data (redraw it if needed). How does the kernel change the decision function? Test the influence of C.

8. For the gaussian kernel, vary the value of the spatial extent γ(e.g. 1, 0.1, 0.01 etc). What happens? How can you explain this result?

9. By setting coef0 to 1, increase the degree of the polynomial. Also vary the extent γ. What happens? Is a non-linear kernel always good for any problem?

10. Define a non-linear classification problem by surrounding one of the class by the other (figure -right). Apply the three kernels with their default parameters (restart the script if needed). Is it what you expected?

11. Is it possible to find parameters for the linear kernel which correctly classify the data?

12. What do you think of the number of support vectors found by the rbf kernel with the default parameters? Reduce γ to 0.001 and explain why it evolves. Is the classification correct?

Questions

13. Increase C (e.g. to 100) and explain why it improves the classification. Is the "best" value of C independent of the data?

14. For the polynomial kernel, set the degree to 3, C to 1 and coef0 to 1. Compare the found number of support vectors with the rbf kernel. Which one is better?

15. Play around with the interface and design datasets with the most complex decision function you can come out with. Try to find parameter values which allow a correct classification.

16. Summarize what you learned about the role of each parameter.