DD2421 HT20 Machine Learning

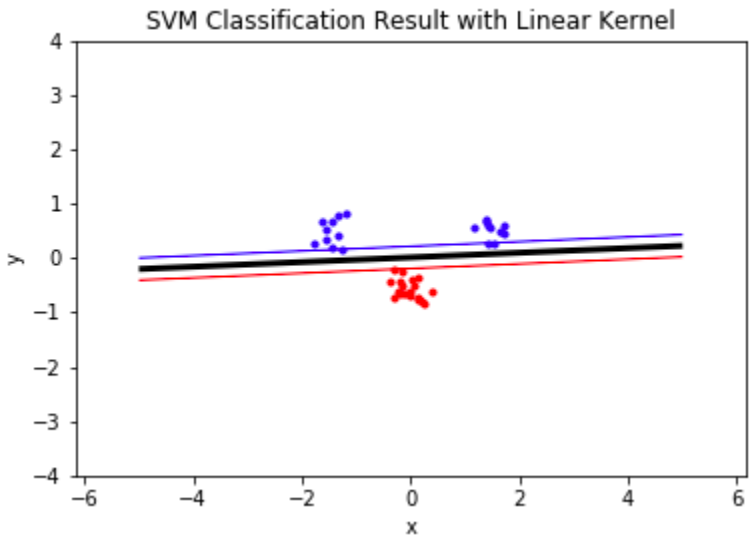
Lab 2: Support Vector Machines

Erfan Wu (erfanw@kth.se)

Yage Hao (yage@kth.se)

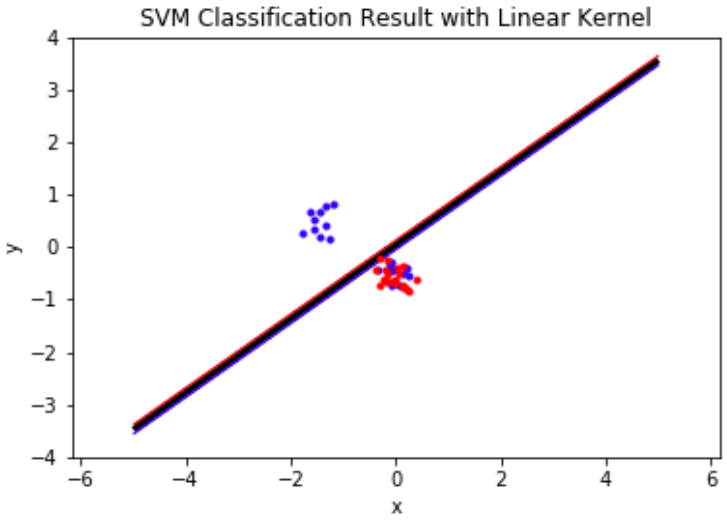
1. **Move the clusters around and change their sizes to make it easier or harder for the classifier to find a decent boundary. Pay attention to when the optimizer (minimize function) is not able to find a solution at all.**

```
# Default Clusters

classA = numpy.concatenate(
    (numpy.random.randn(10,2) * 0.2 + [1.5,0.5],
    numpy.random.randn(10,2) * 0.2 + [-1.5,0.5]))
classB = numpy.random.randn(20,2) * 0.2 + [0.0, -0.5]
```

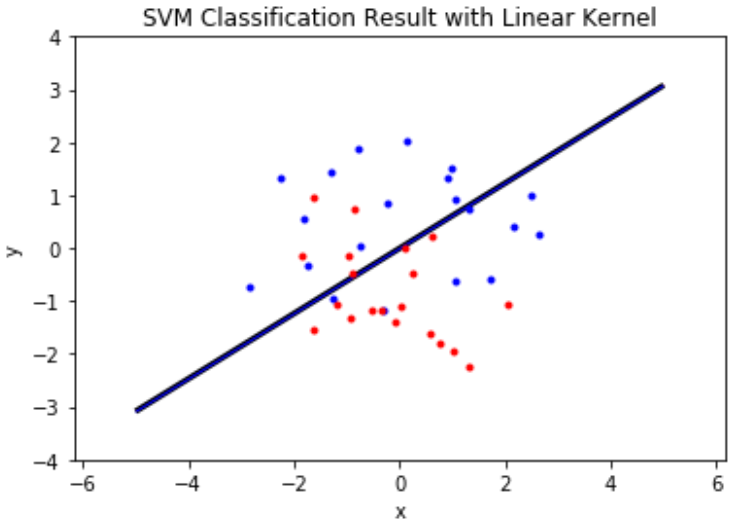



```
# move ClassA by setting the cluster's center overlap with ClassB's
# unable to find a solution
```

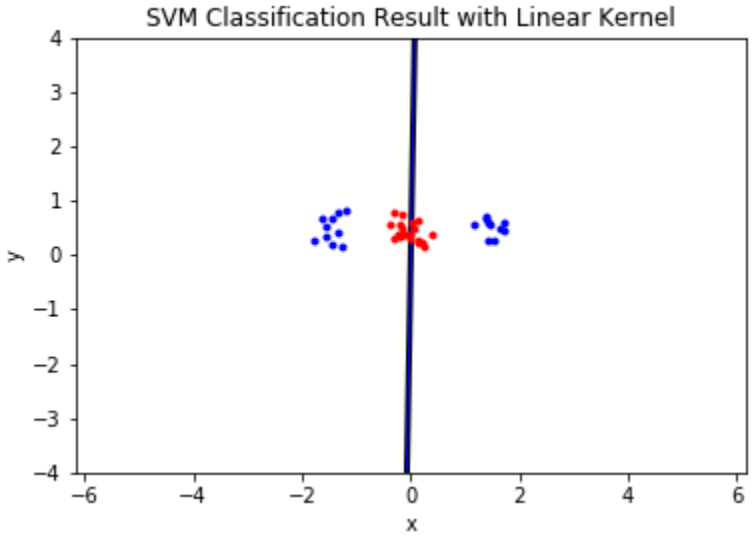
```
classA = numpy.concatenate(
    (numpy.random.randn(10,2) * 0.2 + [0.0,-0.5],
    numpy.random.randn(10,2) * 0.2 + [-1.5,0.5]))
classB = numpy.random.randn(20,2) * 0.2 + [0.0, -0.5]
```




```
# change the size of clusters by increasing the spread factor from 0.2 to 1
# unable to find a solution

classA = numpy.concatenate(
    (numpy.random.randn(10,2) * 1 + [1.5,0.5],
    numpy.random.randn(10,2) * 1 + [-1.5,0.5]))
classB = numpy.random.randn(20,2) * 1 + [0.0, -0.5]
```

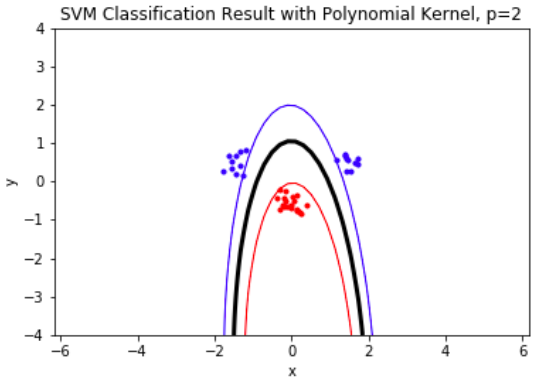


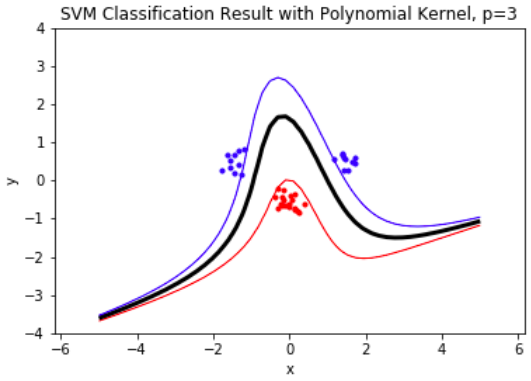
```
# ClassA’s and ClassB’s centers have the same y-coordinate

classA = numpy.concatenate(
    (numpy.random.randn(10,2) * 0.2 + [1.5,0.5],
    numpy.random.randn(10,2) * 0.2 + [-1.5,0.5]))
classB = numpy.random.randn(20,2) * 0.2 + [0.0, 0.5]
```

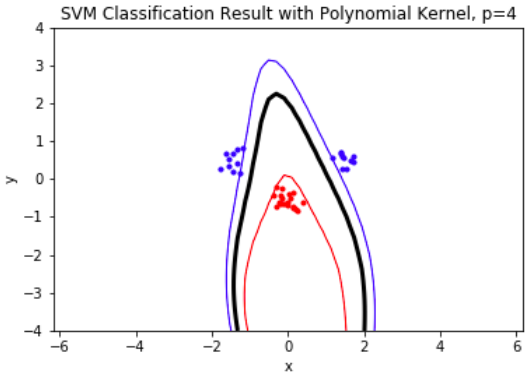


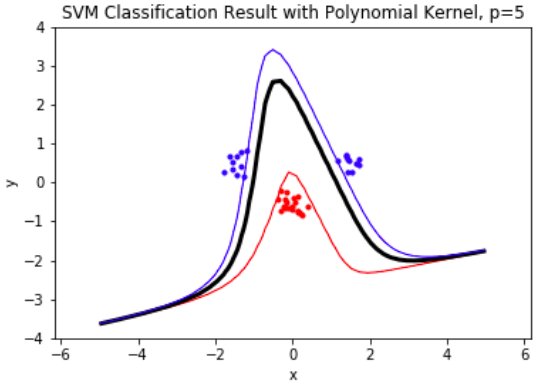
…

2. **Implement the two non-linear kernels. You should be able to classify very hard data sets with these.**

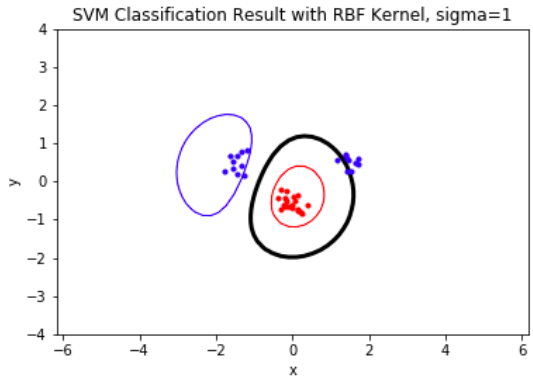


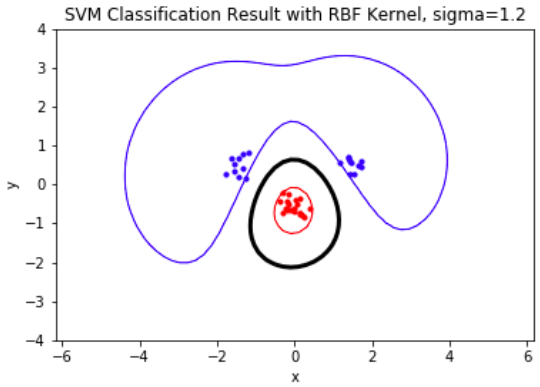


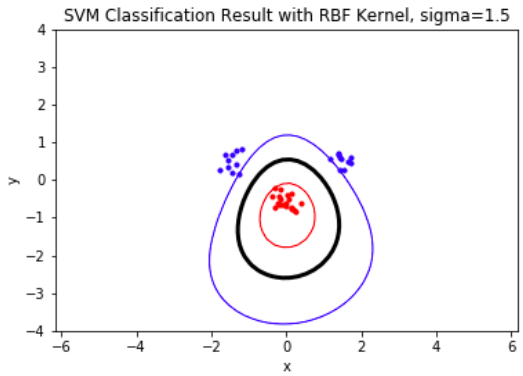


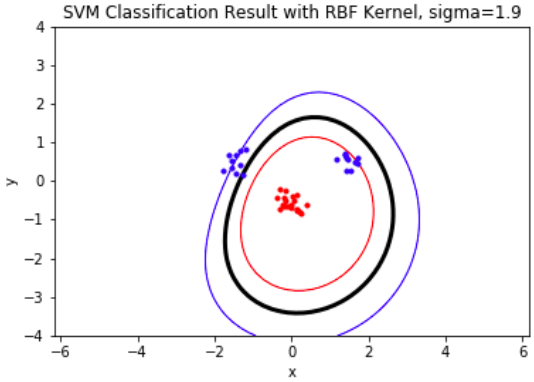


...


SVM Classification Result with RBF Kernel, sigma=1
x
y


SVM Classification Result with RBF Kernel, sigma=1.2
x
y


SVM Classification Result with RBF Kernel, sigma=1.5
x
y


SVM Classification Result with RBF Kernel, sigma=1.9
x
y

...

3. **The non-linear kernels have parameters; explore how they influence the decision boundary. Reason about this in terms of the bias variance trade-off.**

Polynomial Kernel:

There is a parameter, p, representing the degree of the polynomials.

Case p=1 equals a linear kernel. And as the degree of the polynomial increases, the shape of the boundary will be more complex.

As the complexity increases, the classification is more likely to be overfitting, thus bias will decrease and variance will increase.

RBF Kernel:

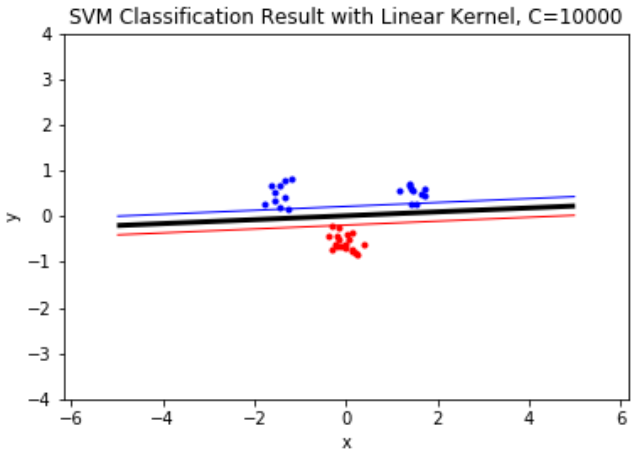
There is a parameter, sigma, controlling the smoothness of the boundary.

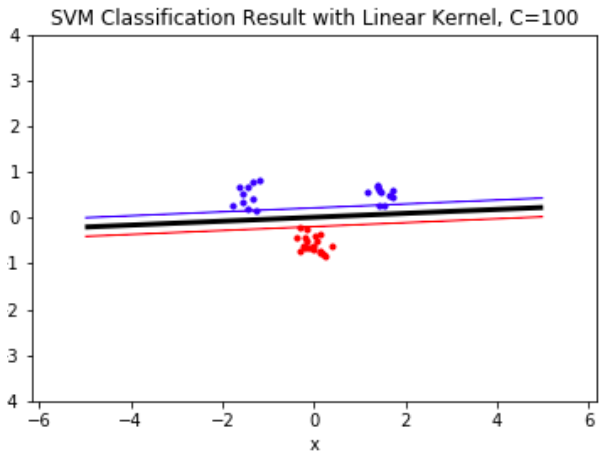
As sigma increases, the boundary will be smoother. Thus when sigma tends to infinity, the RBF kernel tends to be a linear kernel and the complexity decreases.

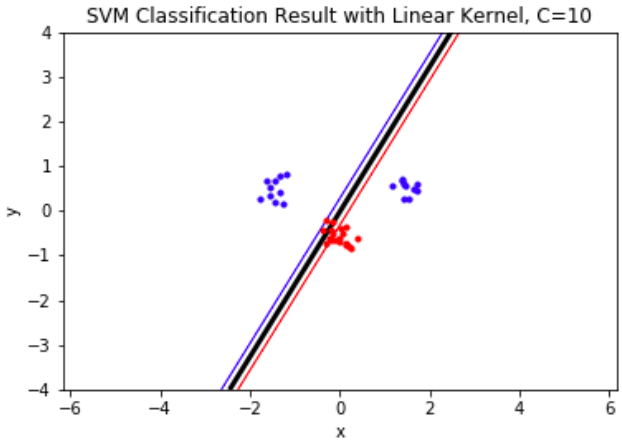
As the complexity decreases, the bias will increase and variance will decrease.

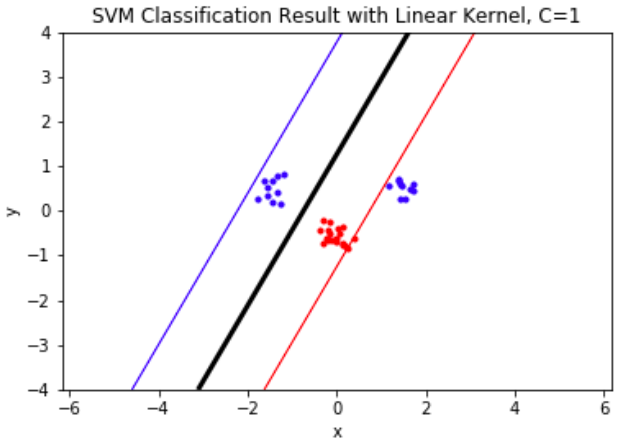
4. Explore the role of the slack parameter C. What happens for very large/small values?

If the data is not hard, i.e. linearly separable, we do not need to set the slack parameter C too small, or we even don't need to set the slack parameter. Because C is an upper bound for α and allows mistakes for separation.

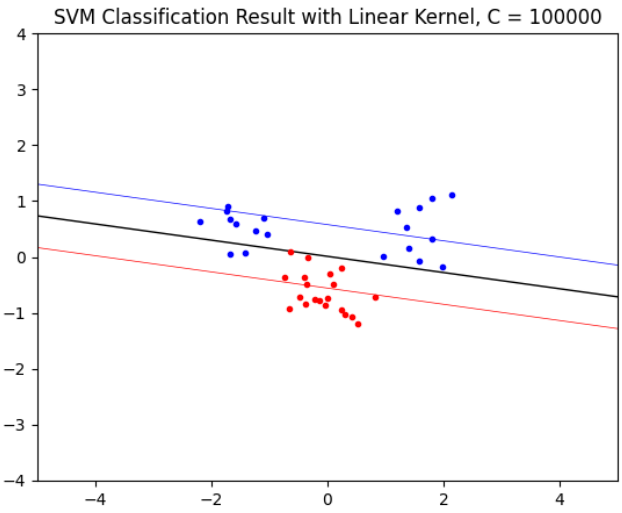


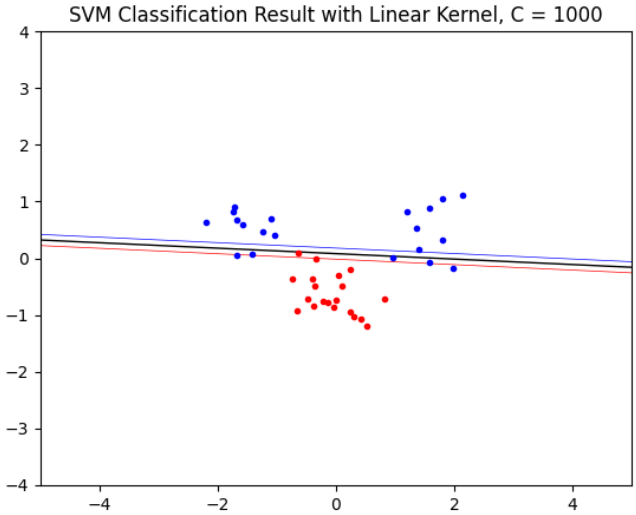


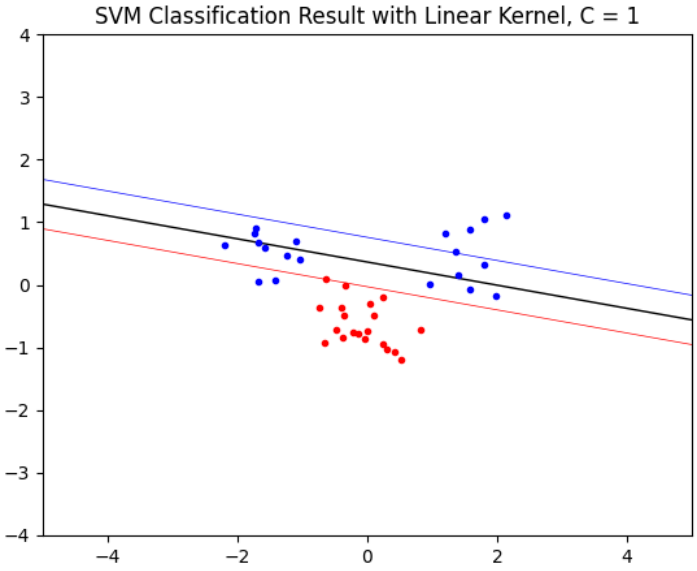




However, if the data contains noise and is hard to separate, we have to find a good C to help separating the data. For example, if we add more noise to the data, either a too large C or a too small C wouldn't work well for the Classification.







5. **Imagine that you are given data that is not easily separable. When should you opt for more slack rather than going for a more complex model (kernel) and vice versa?**

Intuitively, ‘more slack’ means more tolerance on misclassification points.

If a given dataset is mostly linear separable with only a few outliers that may result in misclassification, we can slightly increase the slack variable while still keeping a reasonably high positive true rate.

If a given dataset is not straightforwardly visually separable and distributed complexly, it mostly unlikely can be separated by a linear kernel with slack variable. Thus we may opt for a more complex kernel function, converting the clusters into a higher dimension and separate them there.