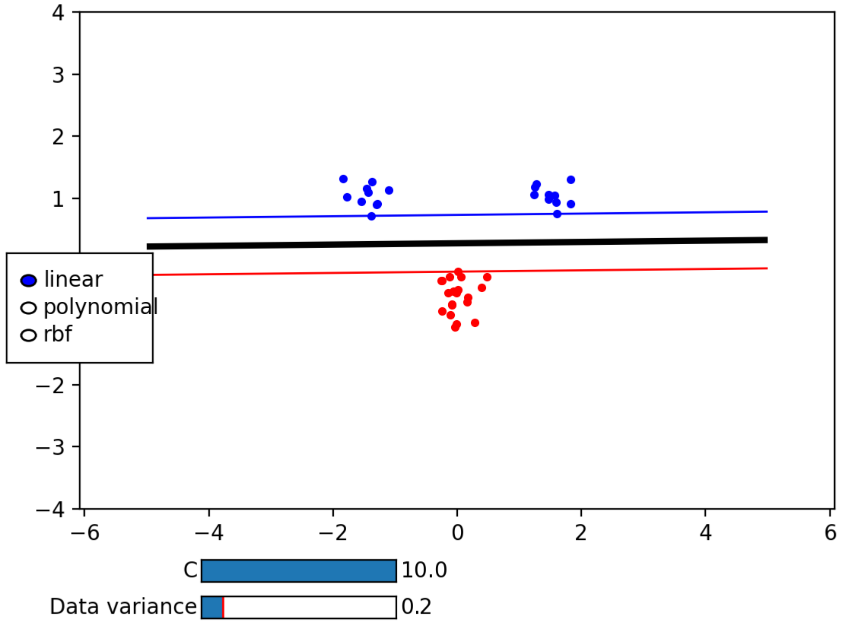
1. Move the clusters around and change their sizes to make it easier or harder for the classifier to find a decent boundary. Pay attention to when the optimizer (minimize function) is not able to find a solution at all.

When data is not linearly separable (because each class is very wide) the **linear kernel** is not able to find a solution

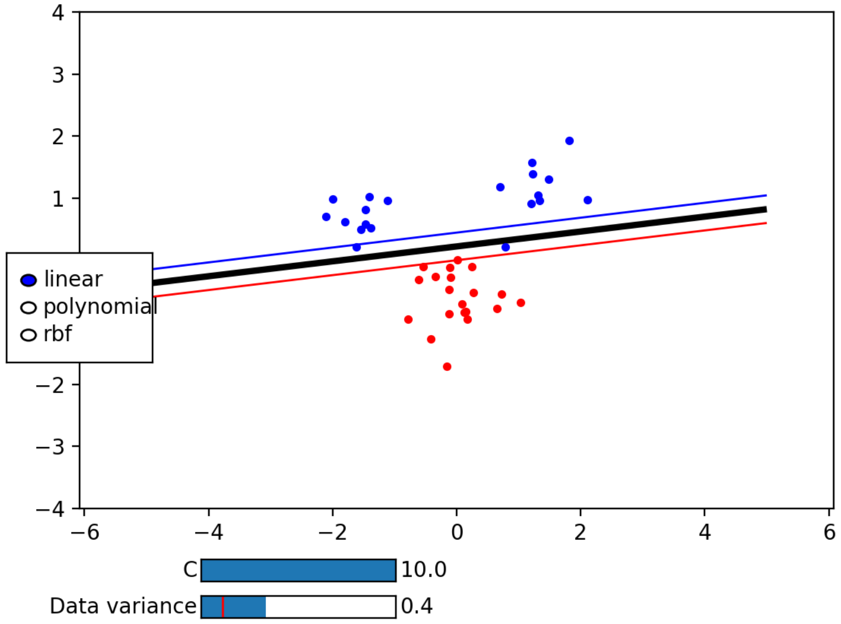
C = 10, Variance = {0.2, 0.4, 0.6, 0.8}



● linear
○ polynomial
○ rbf

C 10.0

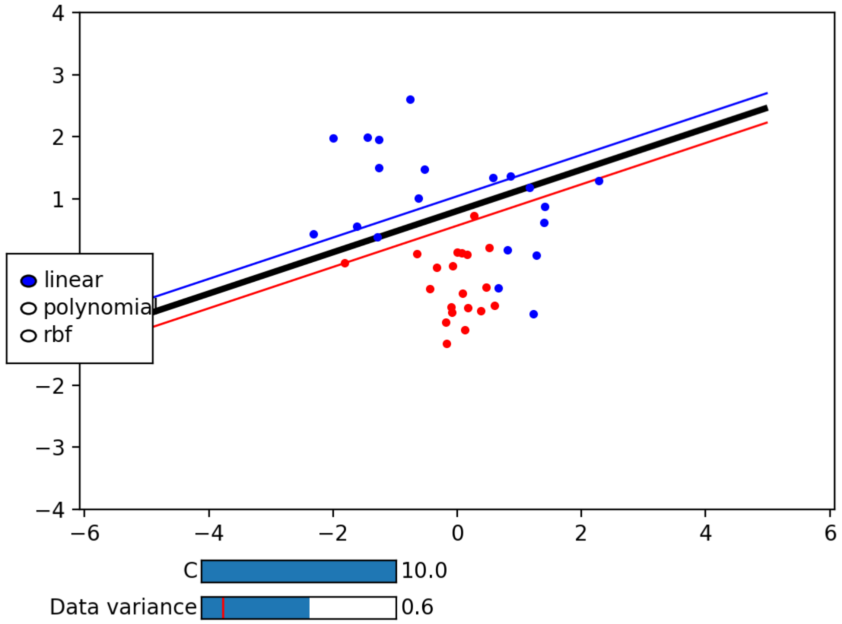
Data variance 0.2



● linear
○ polynomial
○ rbf

C 10.0

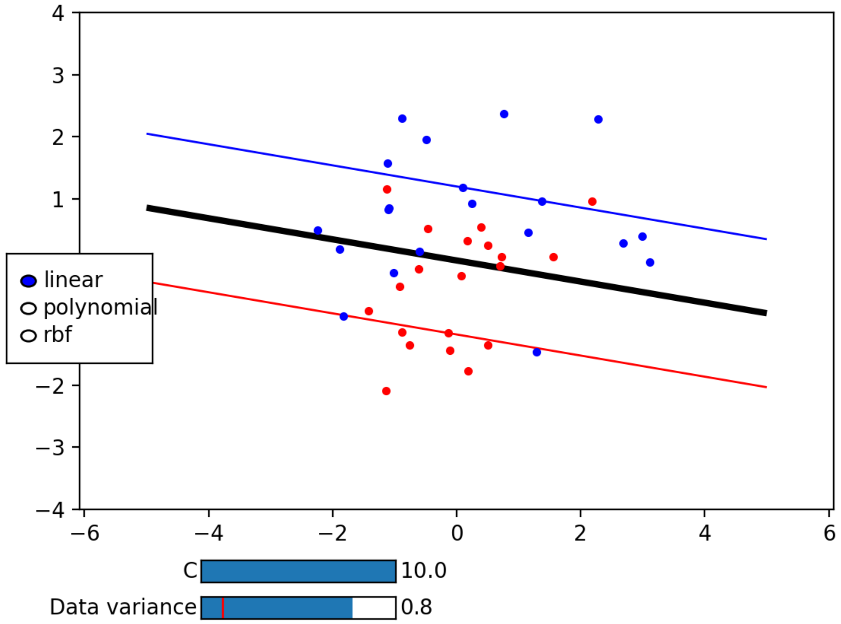
Data variance 0.4



● linear
○ polynomial
○ rbf

C 10.0

Data variance 0.6

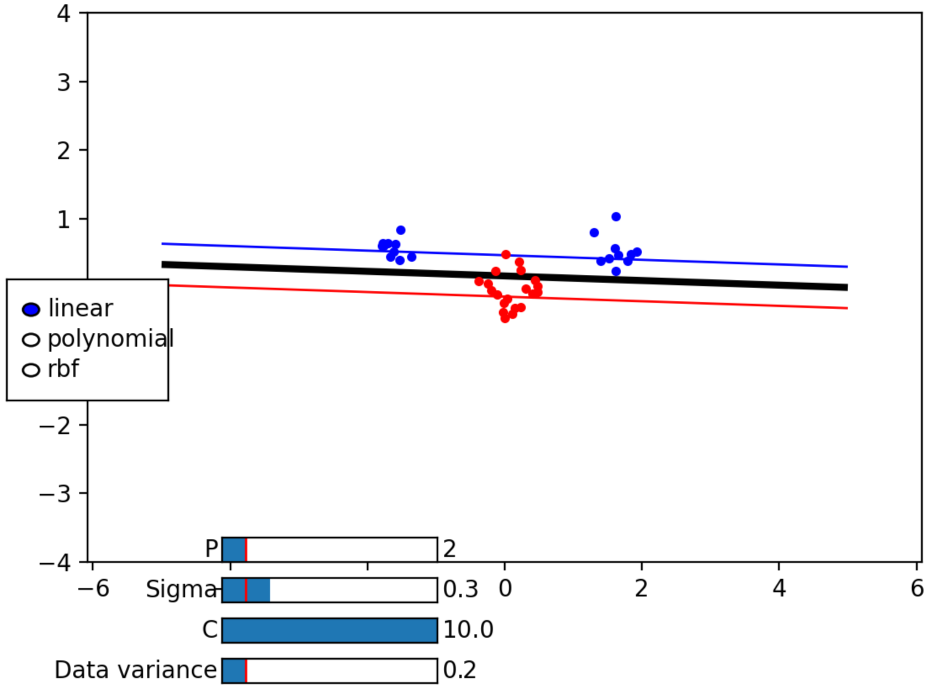


● linear
○ polynomial
○ rbf

C 10.0

Data variance 0.8

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

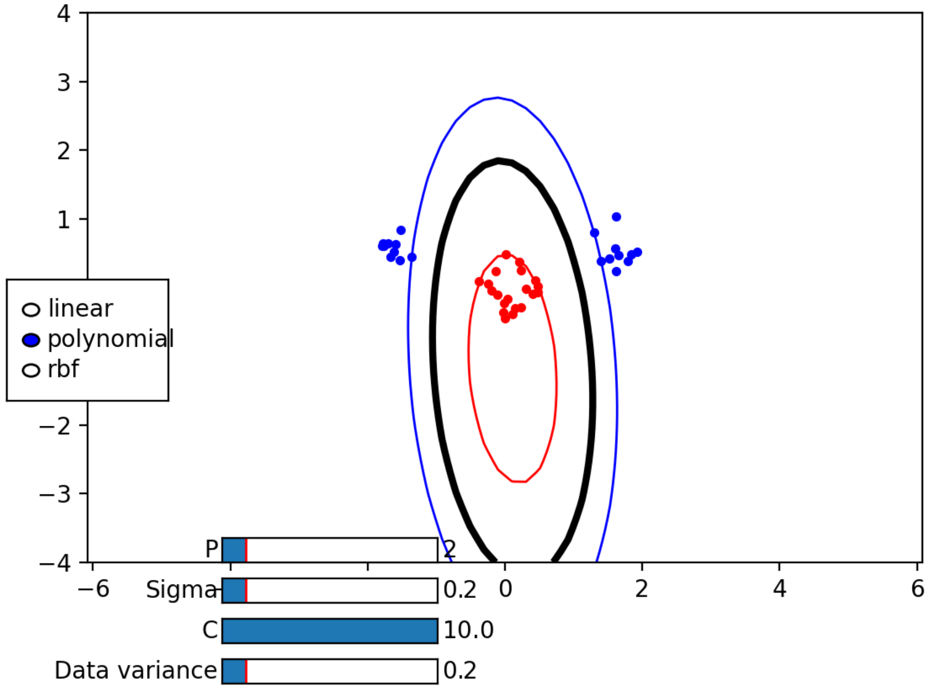


○ linear
○ polynomial
○ rbf

P 2
Sigma 0.3
C 10.0
Data variance 0.2

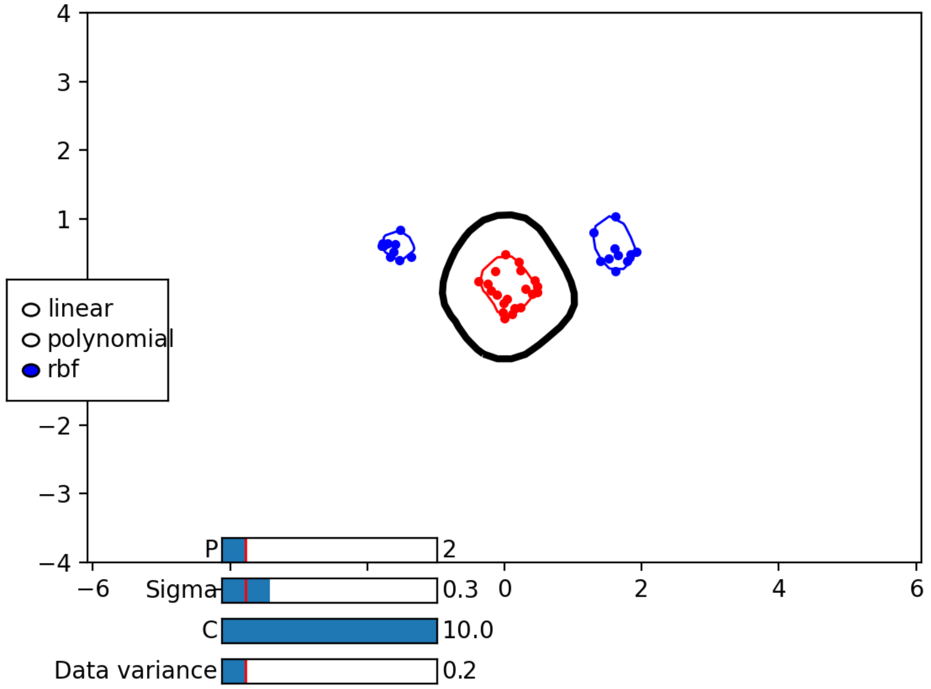
This dataset is not linearly separable.

It is necessary to use more complex kernels to classify the data correctly



○ linear
● polynomial
○ rbf

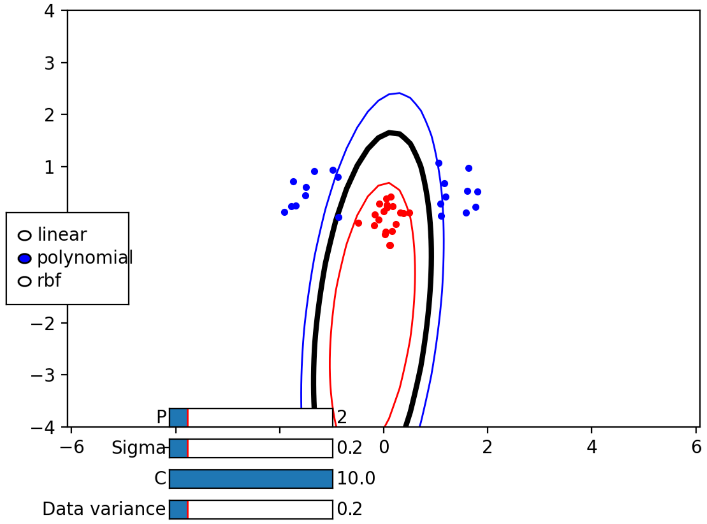
P 2
Sigma 0.2
C 10.0
Data variance 0.2



○ linear
○ polynomial
● rbf

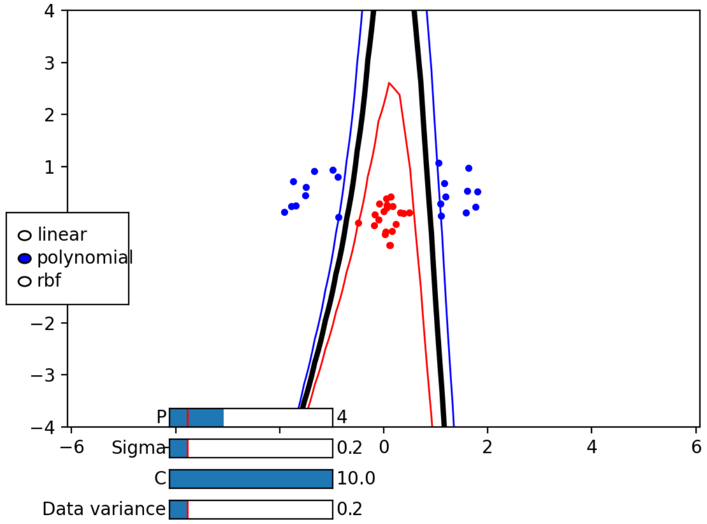
P 2
Sigma 0.3
C 10.0
Data variance 0.2

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.



○ linear
● polynomial
○ rbf

P 2
Sigma 0.2
C 10.0
Data variance 0.2



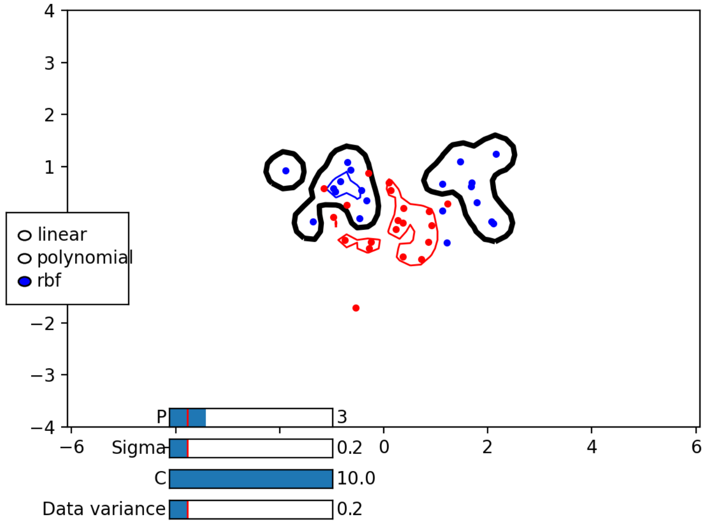
○ linear
● polynomial
○ rbf

P 4
Sigma 0.2
C 10.0
Data variance 0.2

Polynomial Kernel

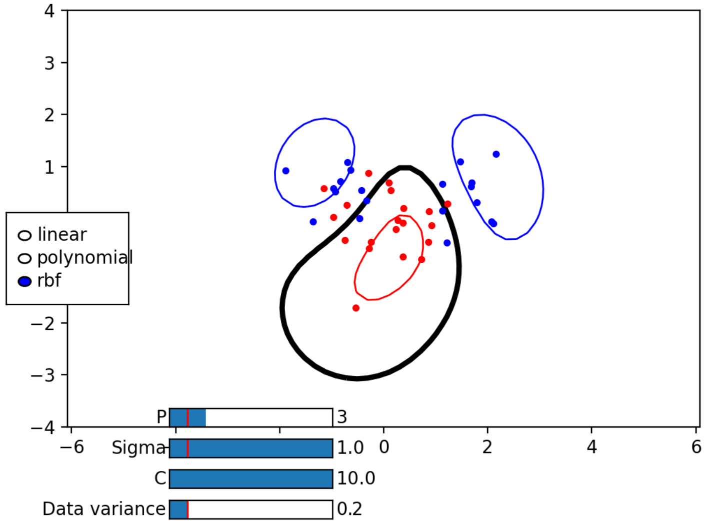
<u>High p:</u> Overfitting, rough decision boundary

Low P: Underfitting, smooth decision boundary



○ linear
○ polynomial
● rbf

P 3
Sigma 0.2
C 10.0
Data variance 0.2



○ linear
○ polynomial
● rbf

P 3
Sigma 1.0
C 10.0
Data variance 0.2

RBF Kernel

Low Sigma: Overfitting, rough decision boundary.
model specialised over the training data
high variance and low bias

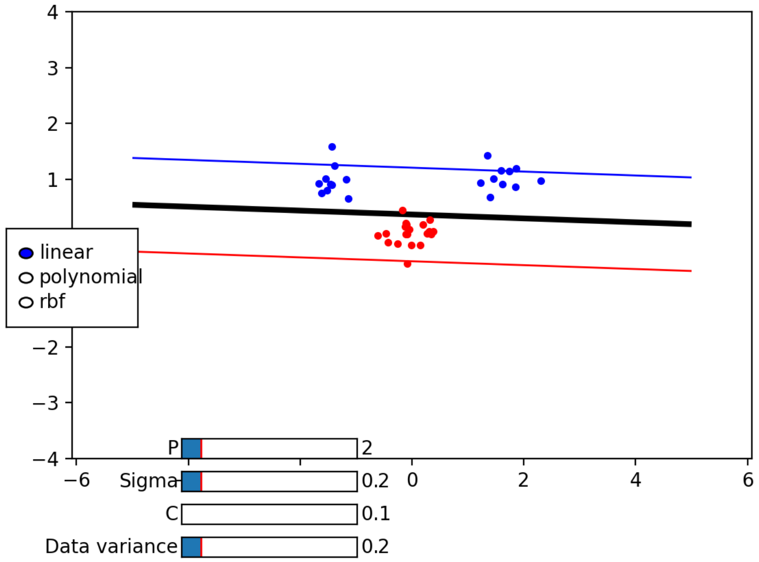
High Sigma: Underfitting, smooth decision boundary
Too generic model
low variance and high bias

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

Slack parameter C sets the importance of avoiding misclassification versus getting a wider margin. C is a penalty term for the error ε in the cost function. The highest the more more penalised the error is.

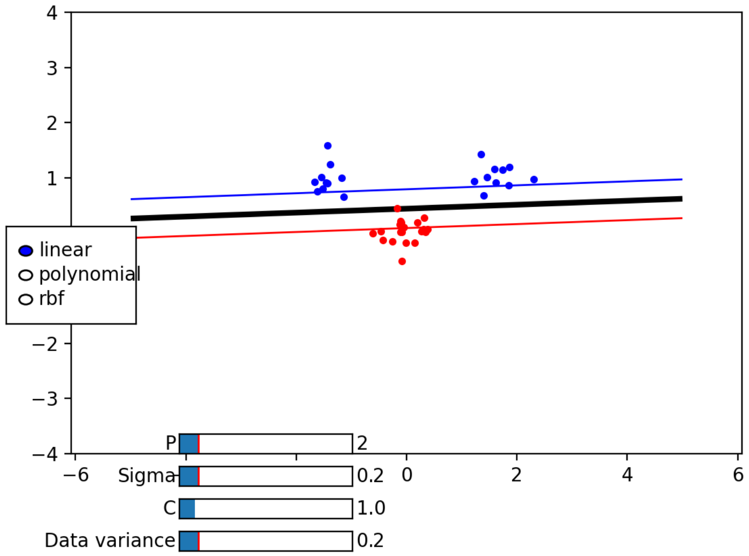
Small C: more slack allowed, ε is not penalised, more misclassification, wider margin

Large C: less slack allowed, ε is highly penalised, less misclassification, thiner margin



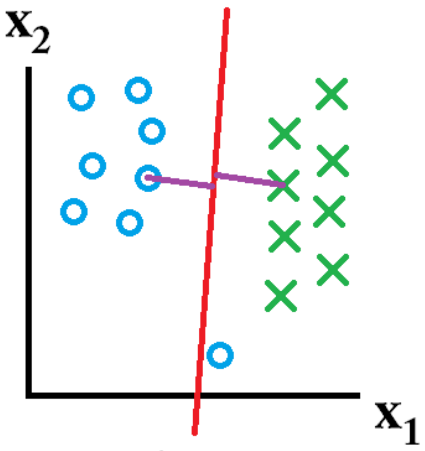
● linear
○ polynomial
○ rbf

P 2
Sigma 0.2
C 0.1
Data variance 0.2

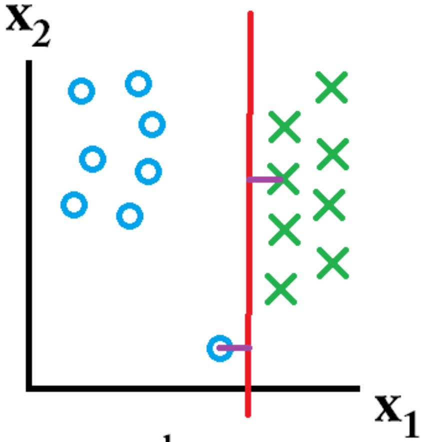


● linear
○ polynomial
○ rbf

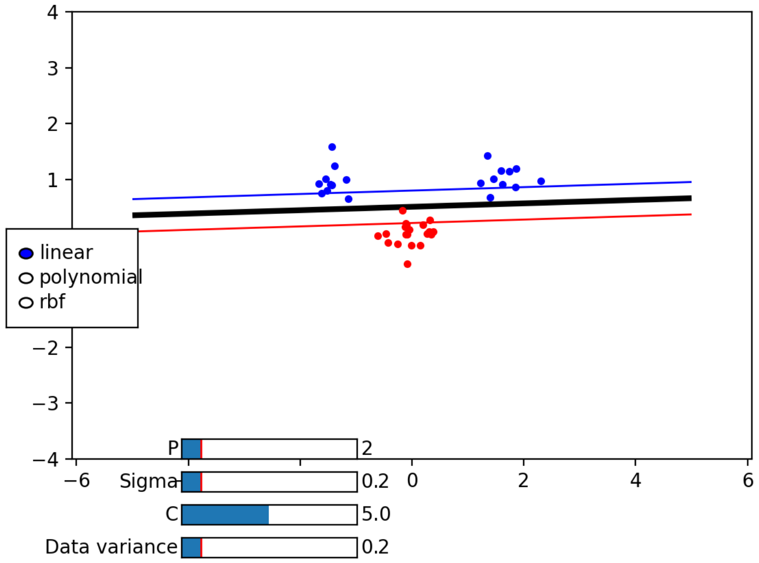
P 2
Sigma 0.2
C 1.0
Data variance 0.2



low c

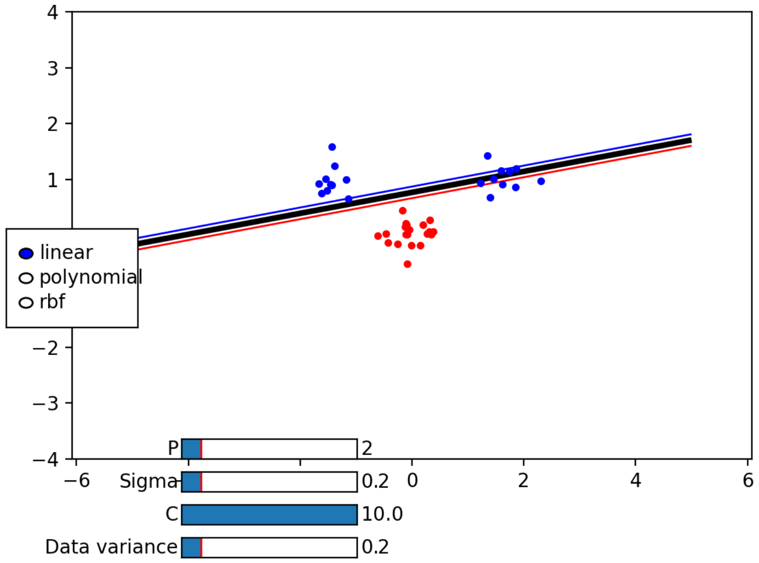


large c



● linear
○ polynomial
○ rbf

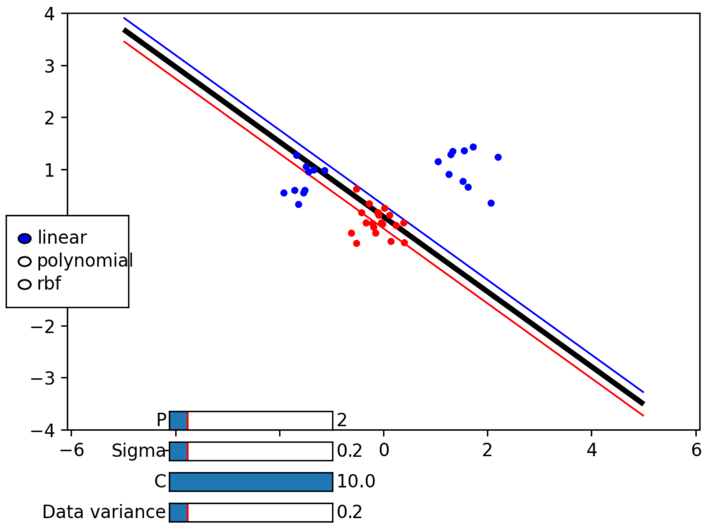
P 2
Sigma 0.2
C 5.0
Data variance 0.2



● linear
○ polynomial
○ rbf

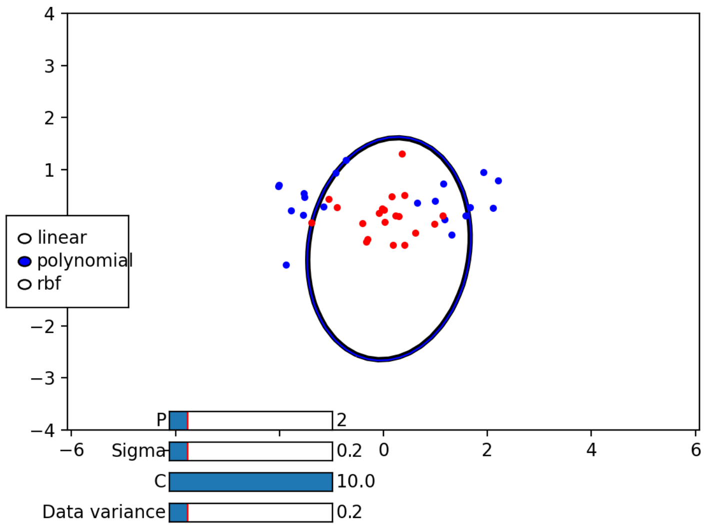
P 2
Sigma 0.2
C 10.0
Data variance 0.2

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?



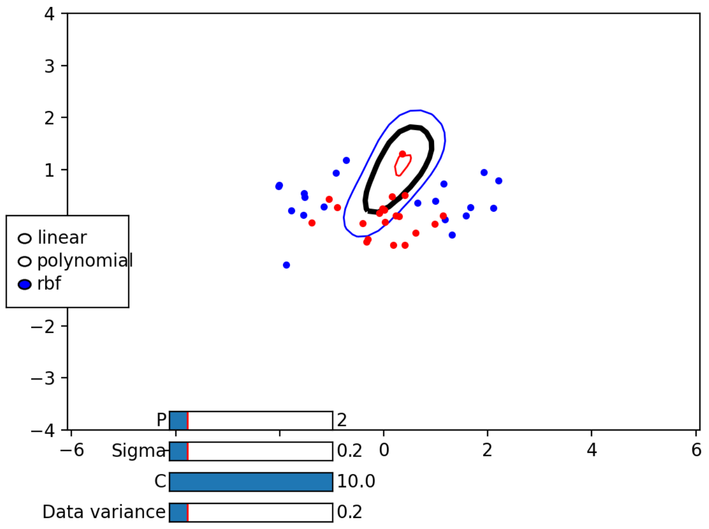
● linear
○ polynomial
○ rbf

P 2
Sigma 0.2
C 10.0
Data variance 0.2



○ linear
● polynomial
○ rbf

P 2
Sigma 0.2
C 10.0
Data variance 0.2



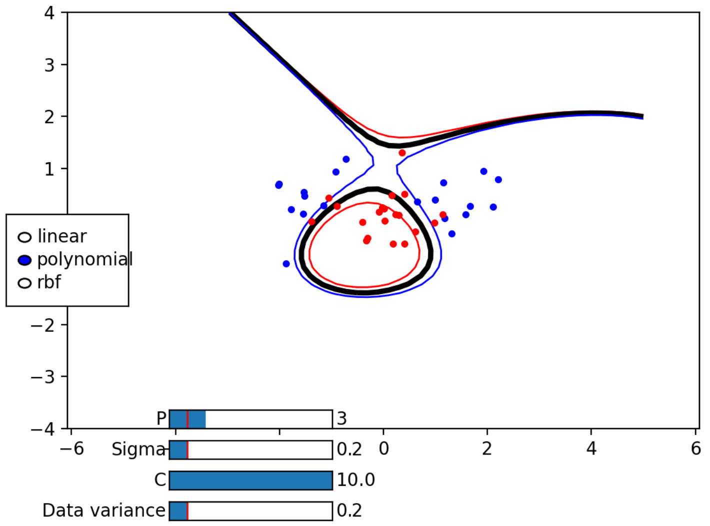
○ linear
○ polynomial
● rbf

P 2
Sigma 0.2
C 10.0
Data variance 0.2

First, try models with more complex kernels and try different setups for those kernels.

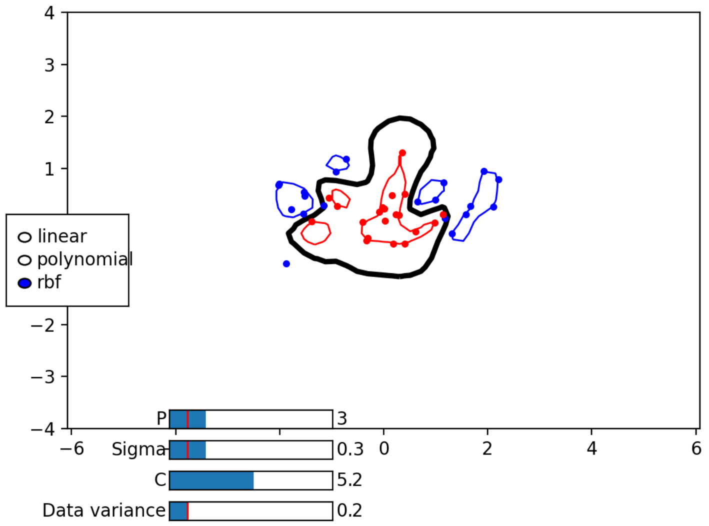
Degree P for polynomial

Sigma for RBF



○ linear
● polynomial
○ rbf

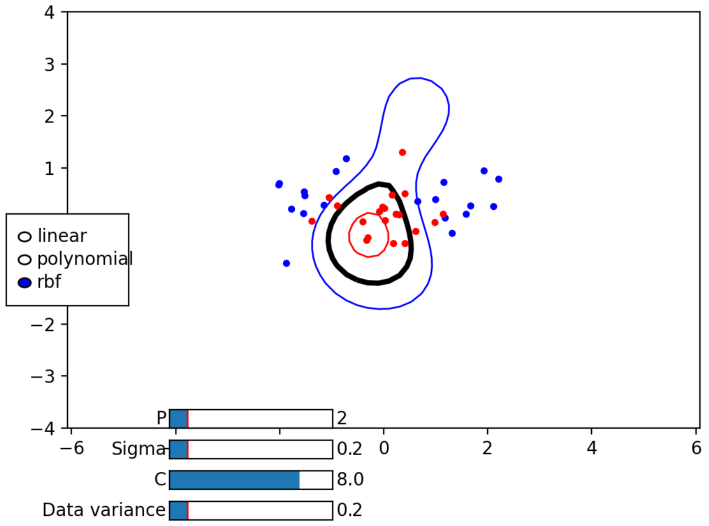
P 3
Sigma 0.2
C 10.0
Data variance 0.2



○ linear
○ polynomial
● rbf

P 3
Sigma 0.3
C 5.2
Data variance 0.2

Then, try smaller values of C until the classification is satisfying



○ linear
○ polynomial
● rbf

P 2
Sigma 0.2
C 8.0
Data variance 0.2

Because slacking consist in a trade-off between maximising the margin and separating correctly all the samples