Part II

1. **Suppose we perform a linear regression with a Gaussian Kernel**

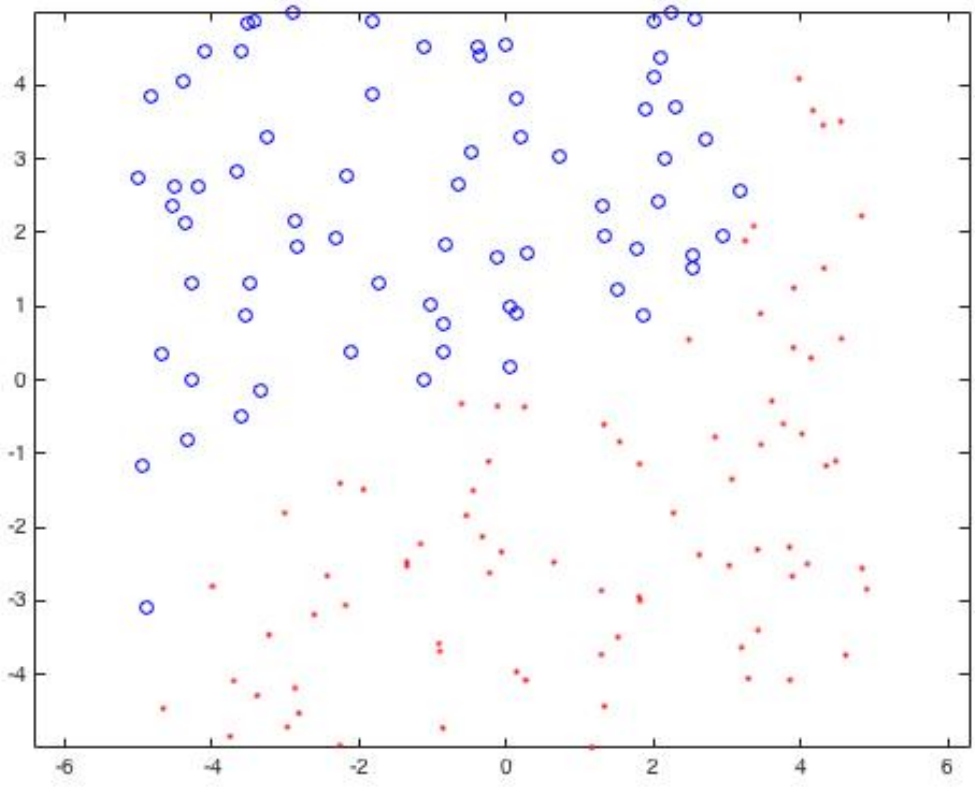
$$K_\beta(\boldsymbol{x}, \boldsymbol{t}) = \boldsymbol{e}^{-\boldsymbol{\beta}\|x-t\|^2}$$

To train a classifier for a two class data set. How should one choose β so that the learned classifier simulates a 1-nearest neighbour classifier.

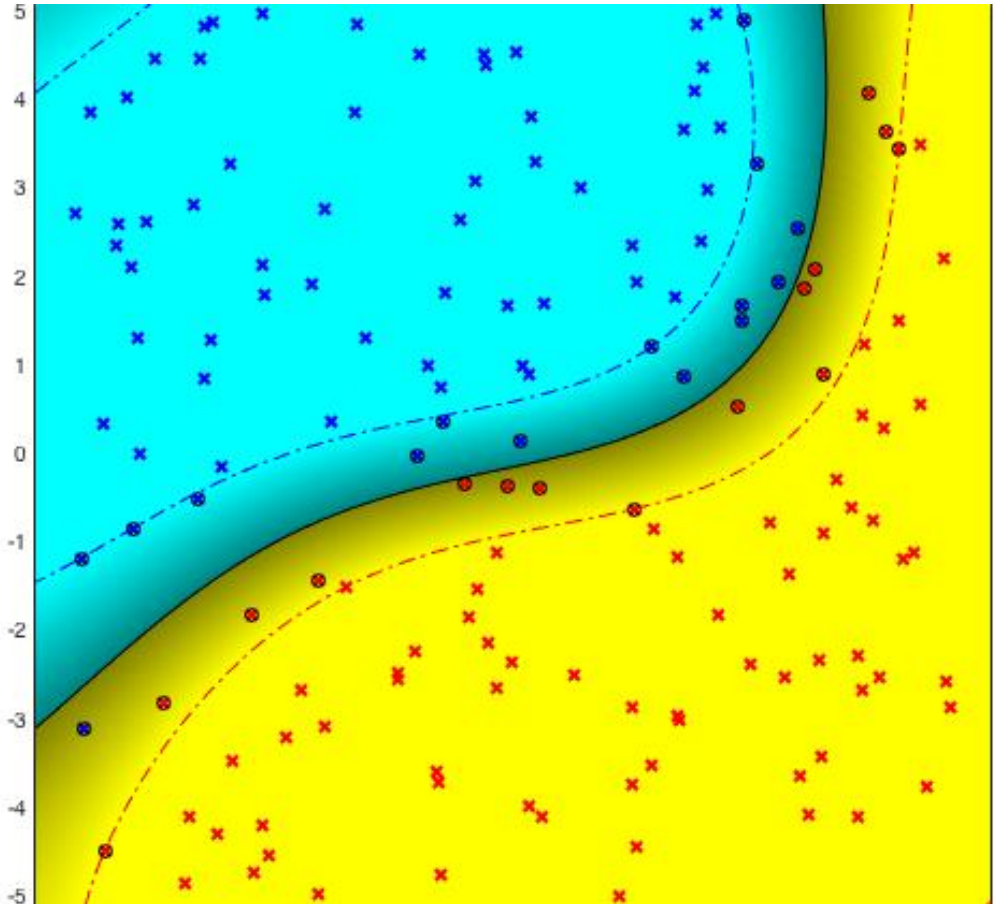
The larger the parameter beta then the more flexibility the classifier will have to fit any label. In fact, at the extreme the Kernel matrix simply becomes close to the identity matrix. One can use this to simulate 1-nearest neighbour.

For illustrative purposes, I have closely followed an example from the book. 'Introduction to pattern recognition using Matlab: A Matlab approach' – Theordoritis et al. The software for the radial basis function kernel for this example and the graphics was not written by ourselves but used from software available with the book.

a) First we generated 150 data points for training and 150 data points for testing that were clearly non-linearly separable in 2-D space.

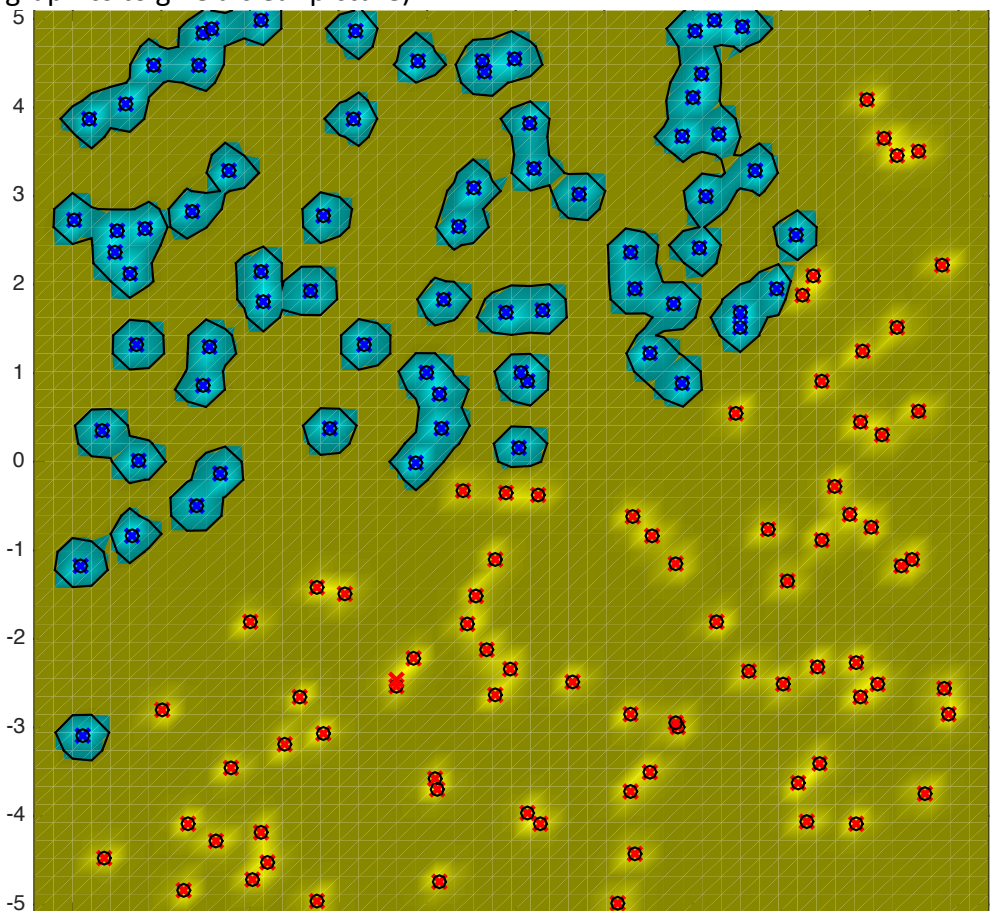


b) The we applied a Gaussian kernel to separate the data – our beta here is a reasonably low value of 0.1.



The graphics show the non-linear separation of the data. The training error was 0.0067, and when tested the test error was 0.02. The data required only 31 support vectors. Given the non-linear data was generated using a polynomial separation then we don't need to increase the flexibility of the kernel with a high beta in reality.

c) Now we simulate the effect of 1-nearest neighbour by increasing our beta. For this small data set and for illustrative purposes I only used a beta of 100. (To enable the graphics to give a clear picture).



Here the training data has been classified correctly 100% of the time. However, when tested the test error was worse at 0.33. Also the number of support vectors equals the number of the data set. For this particular data set which is non-linearly separable by a polynomial this is gross-overfitting. However, this also illustrates how one can simulate 1-nearest neighbour with arbitrary precision by increasing beta.

2. Using a perceptron to learn a linear classifier with a bias.

a) To move from using a perceptron which separates through the origin to one which separates with a bias we observe the following.

$$\boldsymbol{w^T x + b} = (\boldsymbol{w^T}, \boldsymbol{b}) \begin{pmatrix} \boldsymbol{x} \\ \mathbf{1} \end{pmatrix} = \boldsymbol{w'^T x'}$$

So we add ones to our **x** values.

b) How does incorporating the bias term change the Novikoff guarantee on the number of mistakes incurred by the perceptron algorithm.

The Novikoff conditions for the perceptron specify that all of our **x** data points must lie within a ball of radius R (clearly this can be of high dimension). In addition the separation between the data should be at least γ either side of the perceptron plane.

The Novikoff condition on the number of mistakes at time t, $M_t$should be as follows.

$$M_t \leq \binom{R}{\gamma}^2$$

The wording of the question we are not 100% clear on, but we are imagining a situation where we are attempting to separate the same data but with and without a bias term. And then answering as to the impact on the Novikov condition by adding a bias term.

We believe the answer is that this is specific to the data itself. We will illustrate using two examples.

Example 1
Imagine date points in the (x,y) plane with the following labels.

Negative labels (-1/n,0), (-1/n,1).
Positive labels (2,0), (2,1).

A perceptron with no bias would have $\gamma = 1/n$ and R=$\sqrt{5}$. In this case a very high n, or data points very close to the separating line would have a very large Novikov bound i.e $M \leq n^2\sqrt{5}$.

However, if we add a bias term then the data can clearly be separated by a plane with γ=1. Yet R has only increased slightly to 3. (The norm of (2,2,1)). Thus the Novikov bound has decreased.

Example 2
Imagine however the following data points in the (x,y) plane.

Negative labels (-1/2,0),(-1/2,1)
Positive labels (1,0),(1,1)

Here γ=1/2 and $R = \sqrt{2}$ and $M = 2\sqrt{2}$

However, if we introduce a bias term by adding one dimension then our new γ will increase to $\sqrt{\frac{5}{4}}$ and R will increase to $\sqrt{3}$, thus the Novikov bound has increased. In this simple case because $\gamma R < 1$ then increasing the dimensionality by adding a bias increased M (using Pythagoras to calculate the new value γ).

Thus, it seems the effect is data dependent.

3. Kernel Modification

a) For what values of c is K_c a positive semi-definite Kernel?

$$K_c(\boldsymbol{x}, \boldsymbol{z}) = c + \sum_{i=1}^{n} x_i \, y_i$$

$$c \in \mathbb{R}, \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n$$

For this to be a valid Kernel then this must define an inner product in the feature space and satisfy the mercer condition.

c>=0, enables this.

If c<0, then this wouldn't define a valid norm, for example ||x|| <0.

However, in this case we can show the explicit feature map i.e.

$$f(\boldsymbol{x}) = (x1, x2, x3, \ldots., xn, \sqrt{c})$$

so

$$K_c(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{z})'$$

for this to be valid we require c>=0.

b) If we use K_c as a kernel function with linear least squares regression. Explain the effect of c on the solution.