COMS30035, Machine learning: Kernels 3

James Cussens
james.cussens@bristol.ac.uk

Department of Computer Science, SCEEM
University of Bristol

October 8, 2020

Agenda

- The power of the kernel approach
- Soft margins
- Multi-class classification

Kernelise everything!

- So far, we have implicitly assumed that the original form of the data $\mathbf{x}$ is a vector of real numbers.
- But data can also be: graphs, text documents, images, websites, whatever.
- For any sort of $\mathbf{x}$ as long as we have a kernel function $k(\mathbf{x}, \mathbf{x}')$, measuring the similarity between $\mathbf{x}$ and $\mathbf{x}'$ we can apply kernel-based machine learning such as SVMs.
- This paper [KJM20] has an interesting flowchart which helps one choose an appropriate graph kernel.

Choosing/Constructing kernels

- Suppose you have some, say, classification task and you decide to use an SVM approach.
- How do you decide which kernel to use?

Choosing/Constructing kernels

- Suppose you have some, say, classification task and you decide to use an SVM approach.
- How do you decide which kernel to use?
- In practice, people typically use existing, known kernels.
- RBF is a popular choice.
- One can also construct a new kernel function from existing known kernels. See [Bis06, §6.2].
- If 'rolling your own' kernel, the function you define must be symmetric and also any Gram matrix $\mathbf{K}$ must be a *positive semidefinite* matrix.
- scikit-learn lets you use your own Python functions as kernels, but does not check that your function is a valid kernel!

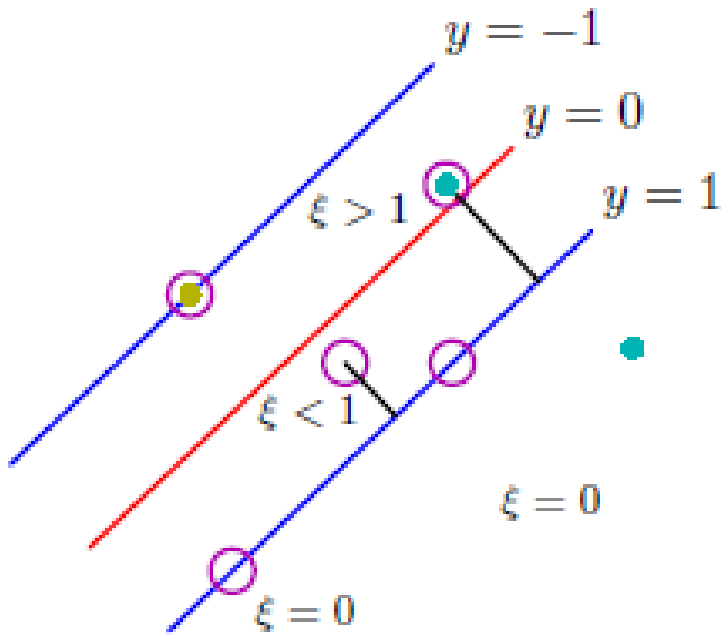
SVMs in practice

- So far we have assumed that:
  1. the training data is linearly separable (in the implicit feature space)
  2. and that we have only two classes.
- Let's now remove these assumptions.

Soft margins

- If we want a nice wide margin then we might have to allow training points to be inside the margin,
- or even on the wrong side of the decicion boundary.
- Figure from [Bis06, p.332].

Illustration of the slack variables $\xi_n \geqslant 0$. Data points with circles around them are support vectors.



Allowing data on the wrong side of the margin

In the Kernels 2 lecture we had the following optimisation problem:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{subject to } t_n(w^T \phi(x_n) + b) \geq 1 \tag{1}$$

scikit learn actually solves the following optimisation (primal version)

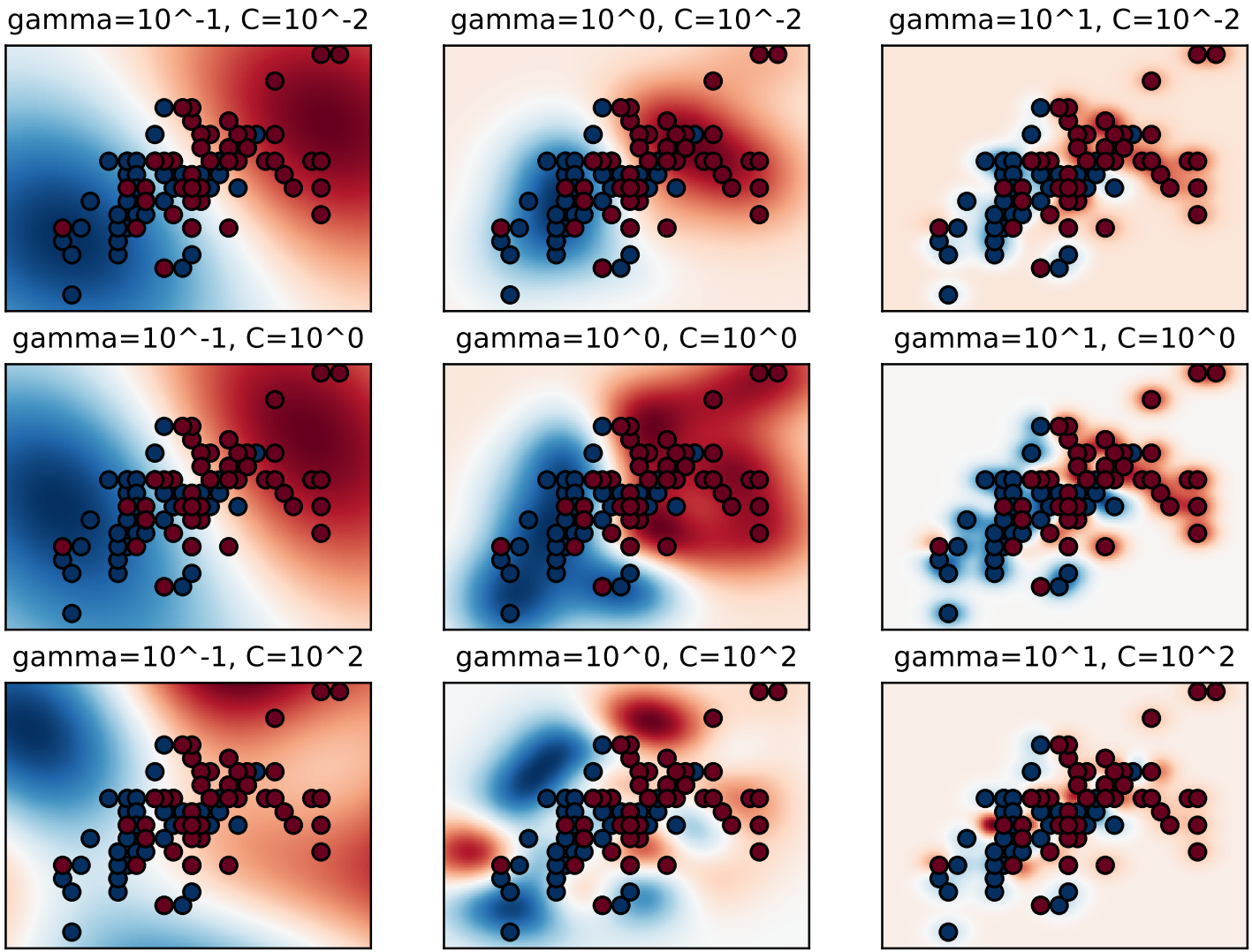
$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \zeta_n \quad \text{subject to } t_n(w^T \phi(x_n) + b) \geq 1 - \zeta_n, \; \zeta_n \geq 0, n = 1, ..., N \tag{2}$$

C as regularisation

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \zeta_n$$
$$\text{subject to } t_n(w^T \phi(x_n) + b) \geq 1 - \zeta_n, \tag{3}$$
$$\zeta_n \geq 0, n = 1, ..., N$$

- "Points for which $0 < \zeta \leq 1$ lie inside the margin but on the correct side of the decision boundary . . . "
- ". . . and those data points for which $\zeta_n > 1$ lie on the wrong side of the decision boundary and are misclassified" [Bis06, p.332]
- We now have a 'soft margin' where being on the wrong side of the margin is merely penalised and C is a regularisation parameter.
- Let's look at part of the output of this Jupypter notebook to understand the effect of varying the value of C.

RBF SVM classification



scikit learn's decision function

The red/blue areas in the plots on the last slide represente values of the *decision function* which is defined in the scikit learn documentation.

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b, \tag{4}$$

RBF SVMs are non-parametric

- Recall that 'the' RBF kernel is: $\exp(-\gamma\|x - x'\|^2)$
- SVMs with RBF kernels are *non-parametric*. The number of support vectors (and thus dual parameters) depends on the data (and value of γ).

Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. (scikit-learn docs)

Multiclass classification

- SVMs are fundamentally two class classifiers.
- If there are k classes, approaches include:

  one-versus-one where we train $k(k-1)/2$ SVM classifiers to distinguish between each pair of classes (and then take a 'vote' for the predicted class)

  one-versus-the-rest where we train k SVM classifiers to distinguish between each class and all other classes.
- In scikit-learn, SVC and NuSVC go for one-versus-one and LinearSVC does one-versus-the-rest.
- This Jupyter notebook provides an example of multi-class learning with various choices of kernel.


Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.


Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris.
A survey on graph kernels.
Applied Network Science, 5, 2020.