CME 216, ME 343 - Winter 2020

Eric Darve, ICME

Kernel trick

An important extension of this method allows turning decision surfaces (the hyperplane that divides points with different colors) into more general surfaces.

To understand this connection, we start from the form of our prediction model

$$w^T x + b$$

Given some training points $x^{(i)}$ we can find coefficients α_i such that

$$w^T x + b = \sum_i \alpha_i \left[x^{(i)}\right]^T x + b$$

So far this is just a change in notation.

However, it reveals that we can view $w^T x$ as dot products between $x^{(i)}$ and x.

An important extension is to realize that it is possible to use other coordinates than x.

These are called features.

Imagine now that we have at our disposal a function $\phi(\boldsymbol{x})$ that is vector-valued.

This represents a set of "features" representing the data $\boldsymbol{x}$.

For example, instead of considering

$$x = (x_1, x_2)$$

we could define

$$\phi(x) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

Then we may use as our model to predict a label

$$\sum_i \alpha_i \phi(x^{(i)})^T \phi(x) + b$$

When this function is positive, we predict the label $+1$ and -1 otherwise.

However, in many cases, it may be difficult to find an appropriate $\phi(x)$, and if ϕ is very high-dimensional (a lot of features) it may be expensive to compute the dot product

$$\phi(x^{(i)})^T \phi(x)$$

The kernel trick is an ingenious idea.

It replaces the expression above by

$$\sum_i \alpha_i K(x^{(i)}, x) + b$$

There are several theoretical justifications and explanations for this approach but here we will simply demonstrate this approach through examples.

Please read Kernel methods in machine learning by Hofman, Schölkopf and Smola for mathematical details on this method.

Many different types of kernels can be used. For example in scikit-learn, we have

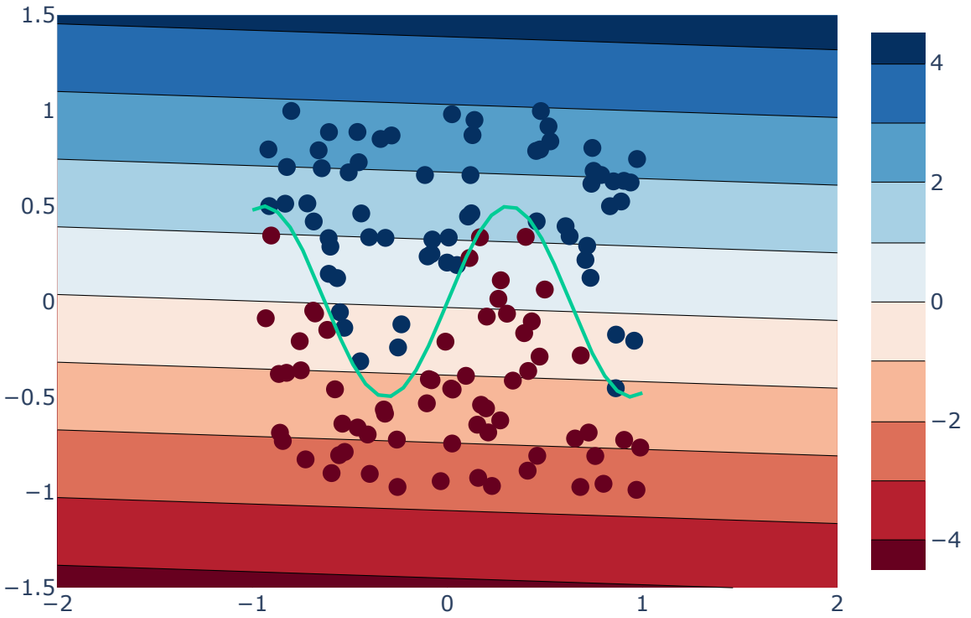
Kernel	Definition
Linear	$x^T x'$
Polynomial	$(\gamma\, x^T x' + r)^d$
Radial basis function (RBF)	$\exp(-\gamma\, \|x - x'\|^2)$
Sigmoid	$\tanh(\gamma\, x^T x' + r)$

We now demonstrate this method on a simple example.

We consider a situation where the blue points on top are separated from the red points on the bottom by a sine function.

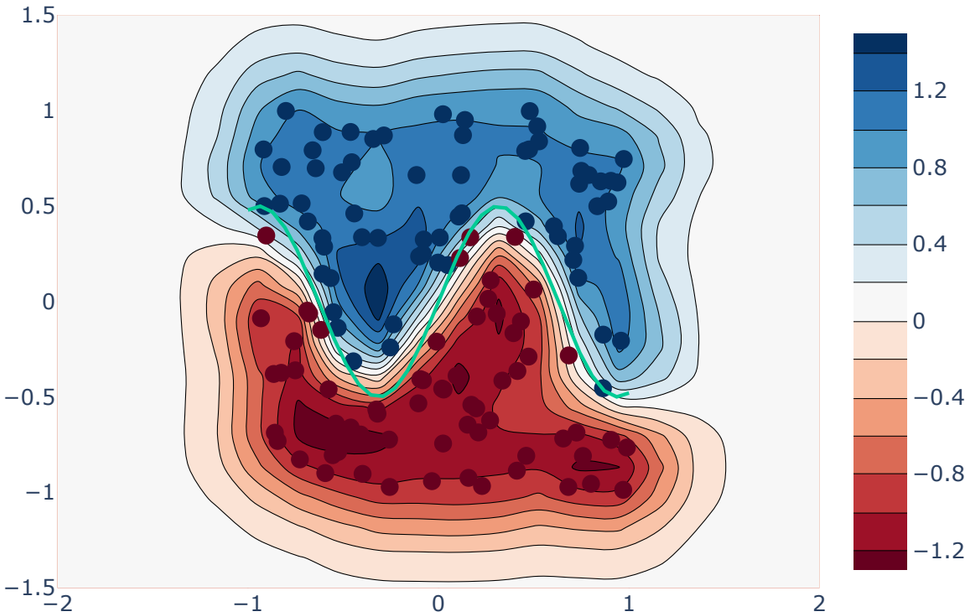
Since a linear SVM uses a line to separate these points it cannot make a good prediction.

Decision function with linear SVM



When we use an RBF, the results are much more accurate.

Decision function with RBF



See Section 5.7.2 in Deep Learning for more details on the kernel trick.

This is done using

```
# fit the model
clf=svm.SVC(kernel="rbf", gamma=10)
clf.fit(X, y)
```

As before, we can tune the regularization parameter `C` to improve the fit

The parameter `gamma` controls the width of the Gaussian function $\mathbf{exp}(-\gamma \left\|x - x'\right\|^2)$.