

# Support Vector Machines (SVMs) Notes

## Lesson 2-6

- SVMs find a separating line (called a **hyperplane**) between data of 2 classes.
  - It takes data as an input and outputs a line that separates those classes if possible.

SUPPORT VECTOR MACHINE

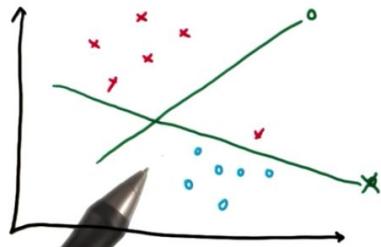


- Maximise the margin so that the hyperplane is more robust, thus less affected by noise.

## Lesson 6

- SVM correctly classifies the labels and then maximises the margin.

SUPPORT VECTOR MACHINES



## Lesson 7

- SVM find a decision boundary that maximises the clearance to both data sets whilst tolerating individual outliers.

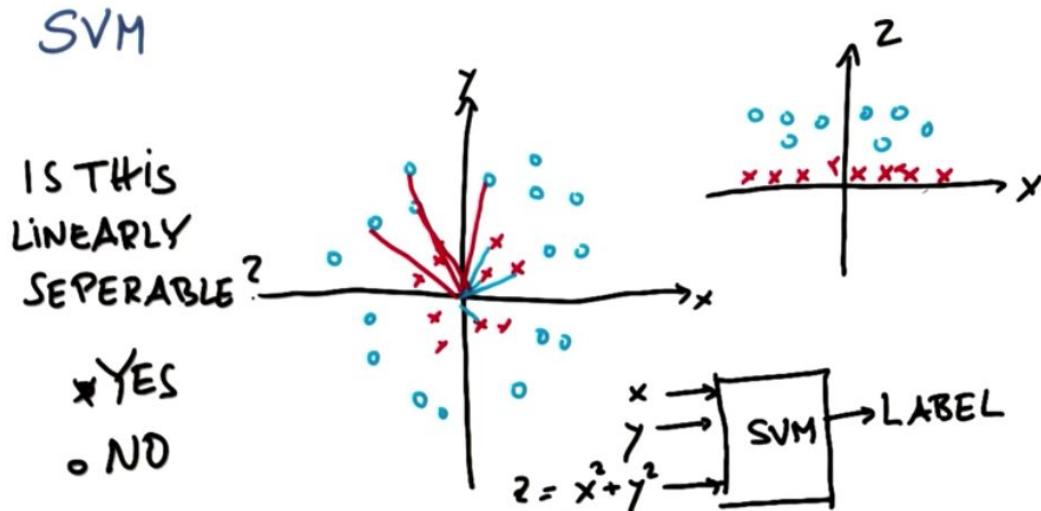
SVMs - OUTLIERS



---

## Lesson 15-16

- In the algorithm, as Z is always non-negative because of the distance to the origin, when plotting x and z you get a hyperplane that is based on the distance to the origin.



### Quiz: A New Feature

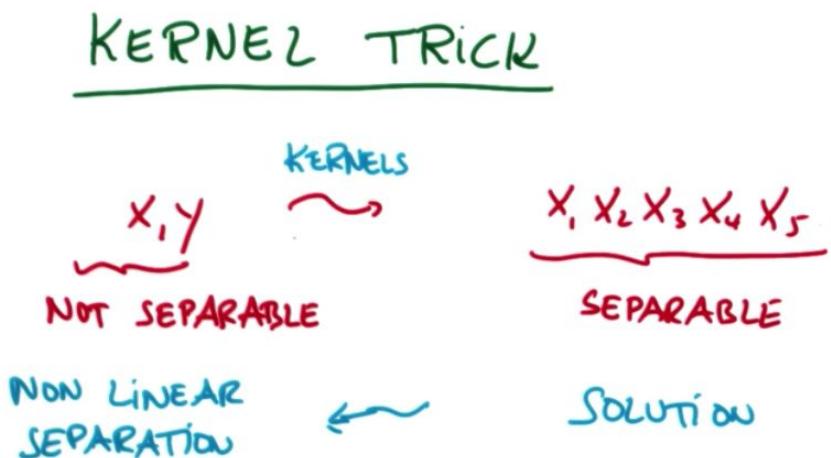
Just to clarify--the labels are outputs when testing, or making predictions. When you're training, of course you need to provide the labels to the algorithm to help it figure out the patterns, which might make the labels sound like inputs in a way.

The main point here is that there's a feature transformation going on in the SVM, which will change the way you think about "linearly separable" data.

---

## Lesson 19

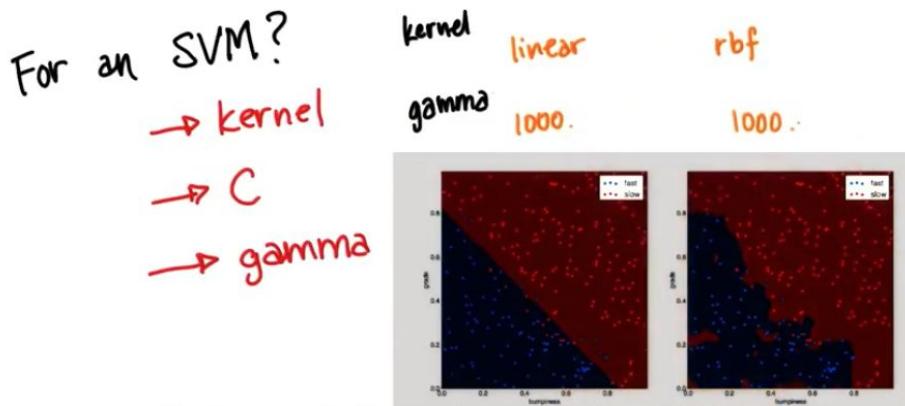
- Kernels** are functions that take a low dimensional input space or feature space, and map it to a very high dimensional space so that what used to be not linearly separable can become separable.



## Lesson 21

- Parameters are arguments passed when you create your classifier (done before fitting).
  - They make a huge difference in the decision boundary that your algorithm arrives at.

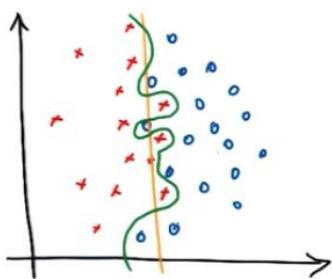
### Parameters in Machine Learning



## Lesson 22

### SVM C Parameter

C — controls tradeoff between smooth decision boundary and classifying training points correctly



#### Quiz

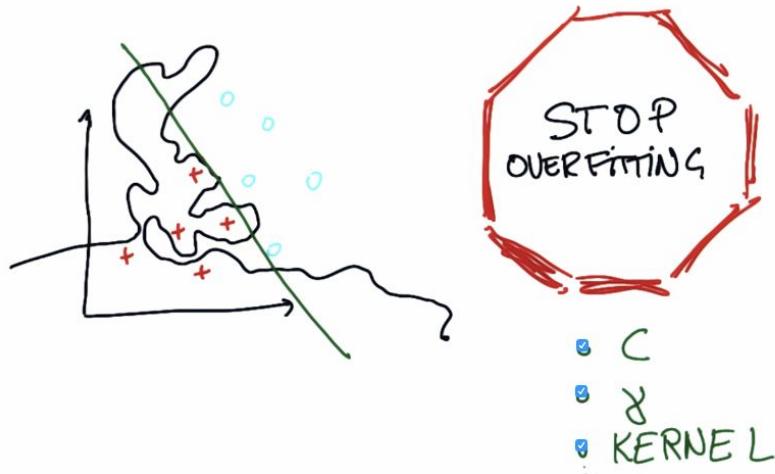
does a large C mean you expect a smooth boundary, or that you will get more training points correct?

- o smooth boundary
- q more training points correct

use rbf kernel!

## Lesson 23

- When the machine learning algorithm produces a complex hyperplane when it could be much simpler, you are **over-fitting**.
- **Control overfitting** by changing the parameter of your algorithm.



---

## Lesson 24

- **SVM**

- **Advantages:**

- Work really well in complicated domains where there is a clear margin of separation.

- **Disadvantages:**

- Don't perform very well in very large datasets because the training time happens to be cubic in the size of the dataset.
    - Don't work well with lots of noise.
      - When the classes are overlapping you have to count independent evidence. **Use NB classifier instead.**

# Mini Project

---

## Lesson 26-27

In this mini-project, we'll tackle the exact same email author ID problem as the Naive Bayes mini-project, but now with an SVM. What we find will help clarify some of the practical differences between the two algorithms. This project also gives us a chance to play around with parameters a lot more than Naive Bayes did, so we will do that too.

Go to the svm directory to find the starter code (svm/svm\_author\_id.py).

Import, create, train and make predictions with the sklearn SVC classifier. When creating the classifier, use a linear kernel (if you forget this step, you will be unpleasantly surprised by how long the classifier takes to train). What is the accuracy of the classifier?

---

## Lesson 28-30

Are the SVM training and predicting times faster or slower than Naive Bayes?

- faster
- slower
- it's about the same

Exact times may vary a bit, but in general, the SVM is MUCH slower to train and use for predicting.

- **One way to speed up an algorithm** is to train it on a smaller training dataset. The tradeoff is that the accuracy almost always goes down when you do this.
- **If speed of algorithm (for many real-time machine applications it is)** is a major consideration then you may want to sacrifice a bit of accuracy to train/predict faster.

Which of these are applications where you can imagine a very quick-running algorithm is especially important?

- predicting the author of an email
- flagging credit card fraud, and blocking a transaction before it goes through
- voice recognition, like Siri

---

## Lesson 32

Keep the training set size and rbf kernel from the last quiz, but try several values of C (say, 10.0, 100., 1000., and 10000.). Which one gives the best accuracy?

Which of these values of C gives the best SVM accuracy?

- 10.
- 1000.
- 100.
- 10000.

---

## Lesson 33

Once you've optimized the C value for your RBF kernel, what accuracy does it give? Does this C value correspond to a simpler or more complex decision boundary?

What's the accuracy of your SVM now? Is the boundary more or less complex than if C had its default value of 1.0?

- more complex
- less complex

0.892491467577

---

## Lesson 34

Now that you've optimized C for the RBF kernel, go back to using the full training set. In general, having a larger training set will improve the performance of your algorithm, so (by tuning C and training on a large dataset) we should get a fairly optimized result. What is the accuracy of the optimized SVM?

---

## Lesson 35

What class does your SVM (0 or 1, corresponding to Sara and Chris respectively) predict for element 10 of the test set? The 26th? The 50th?

- |   |    |
|---|----|
| 1 | 10 |
| 0 | 26 |
| 1 | 50 |

---

## **Lesson 37**

Our general suggestion is to try a few different algorithms for each problem. Tuning the parameters can be a lot of work, but just sit tight for now--toward the end of the class we will introduce you to GridCV, a great sklearn tool that can find an optimal parameter tune almost automatically.