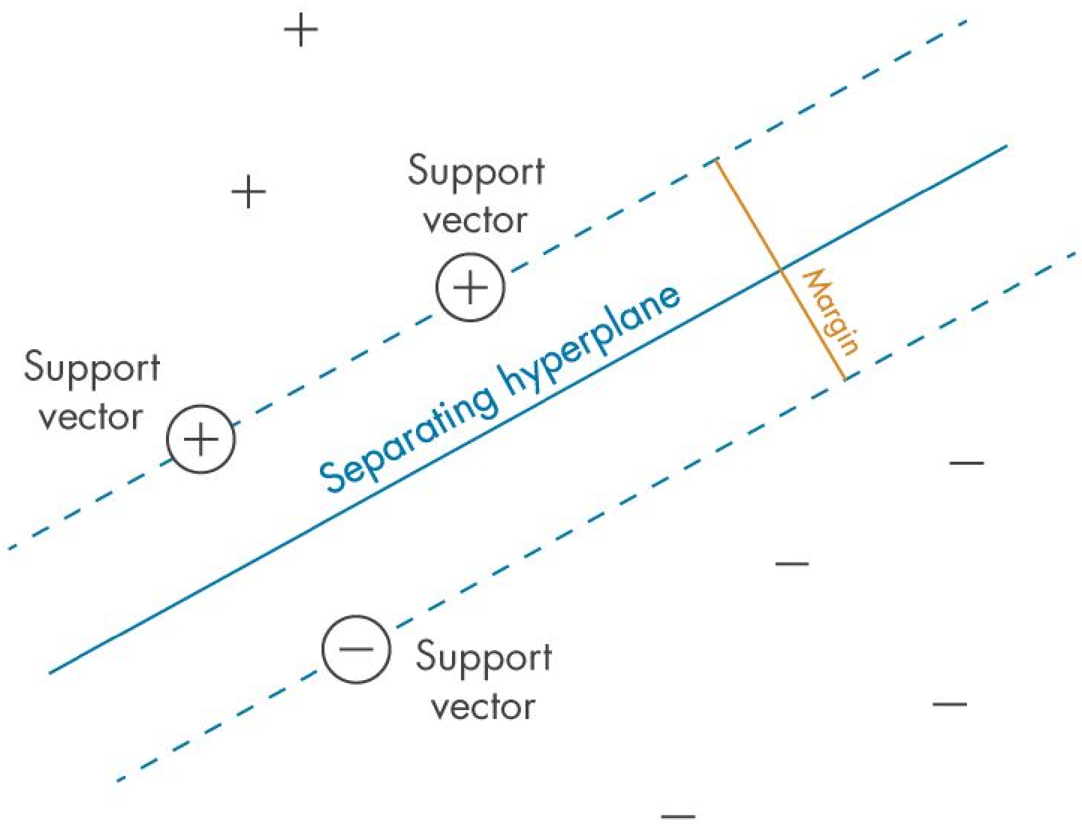
Support Vector Machines

Chapter 5 Machine Learning

- This builds upon concepts in linear discriminant functions, namely the weight vector.
    - It aims to optimise this weight vector such that the boundary it creates ensures maximal distance between itself and points at the extremes of classes:
    - The boundary that comes from maximising the margin is known as, you guessed it, the **maximum margin hyperplane (MMH)** and when used to classify, is called the **maximum margin classifier(MMC)**



- So, back to linear discriminant functions: the link here is that we enforce a weight matrix, but with magnitude 1 such that

$$|\boldsymbol{w}| = 1$$

remember from Linear Discriminant Functions,

$$r = \frac{g(\boldsymbol{x})}{|\boldsymbol{w}|}$$

so setting |w| = 1 allows g(x) to give the signed distance from the hyperplane

- Then we maximise by finding the largest value of M, where y_i = {1,-1} such that for all m training points

$$y_i(\boldsymbol{w}^T x_i + w_0) \geq M$$

this translates to 'distance from hyperplane >/ M' because any negative classifications will leave y_i = -1 which when multiplied with g(x) which will also be negative, just gives a positive distance.

Soft Margin Classifiers

- As you could probably sus out, the crude approach above makes you extremely sensitive to outliers/ extremes within classes - so what if you could reduce overfit a lil?
- Enter the soft margin classifier that adds leeway, **ε**, to the Margin requirement. The modified criteria to satisfy becomes:

$$y_i(\boldsymbol{w}^T x_i + w_0) \geq M(1 - \epsilon_i)$$

- The total amount of leeway is assigned a budget

$$\sum_i \epsilon_i \leq C$$

- The value of **ε** alludes to where the point stands with respect to the margin (if it's an outlier or not)
    - If = 0, then this point is on the right side of the margin and we don't need to give it any leeway
    - if > 0 then this point is on the wrong side of the margin and needs leeway
    - if >1 then this point is a damn outlier and is not only on the wrong side of the margin but also the classification hyperplane and needs hella leeway
- C here is a form of regularisation, akin to Regression > Ridge Regularisation

Mapping using a non-linear boundary

- Remember with [Linear Discriminant Functions > Side note on high order classification using this method] it wasn't typically common to do the whole 'treat high order terms as new parameters' thing due to the curse of dimensionality? Well with SVMs you can take a similar approach to better success.
- It's all based on defining a space that produces the best boundary, e.g.

$$\boldsymbol{y} = (x_1^2, x_2^2, \sqrt{2}x_1, x_2)$$

- Different libaries have some toolboxes to find some good projection spaces for ya.
- So why doesn't this run into the same problem as linear discrims? It's because of the potential to use kernels
    - In the typical workflow you have the following steps:
        1. Training data in original space
        2. Map training data into higher order space
        3. Calculate dot products in high order space
        4. Generate classifier SVM from dese

- Kernels allow you to skip step 2 and do the dot products directly, this speeds up computation