# ML4 – Support Vector Machines

## Introduction

The Perceptron algorithm calculates the decision line between two linearly separable classes clustered in feature space. We note that the solution found by the Perceptron is not unique. Running the algorithm with a different set of initial weights will give you a different but equally valid solution.

What if we want to find the best decision line between the two classes? By "best" we mean that the decision line should be the one that offers the best generalization when tested with novel data. Visually, the best decision line should have the largest margin between the two classes as shown in Figure 1. For linearly separable classes the solution to the best decision line is unique.

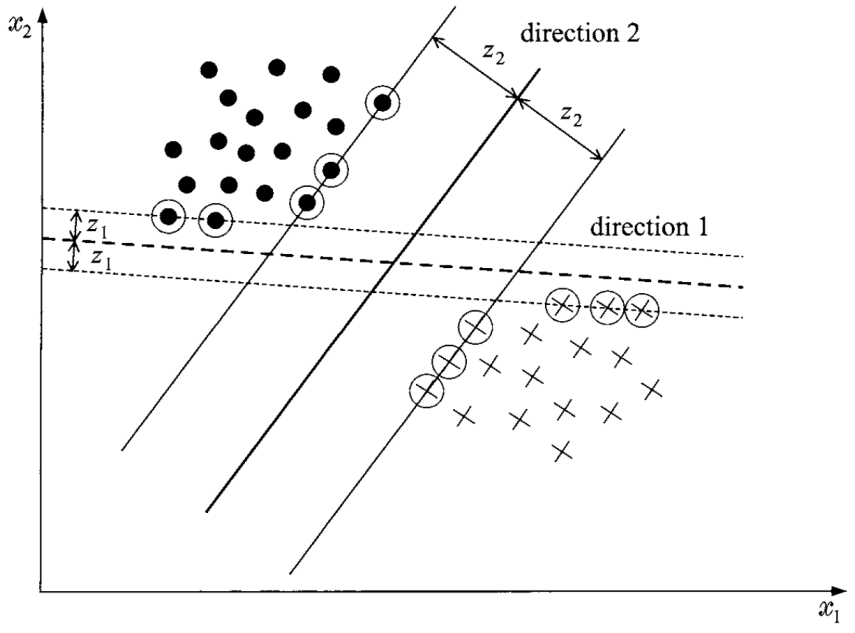


*Figure 1. Decision line along direction 2 has a larger margin (expressed as perpendicular distance z ) than that of direction 1. The feature vectors that are closes to the decision lines are known as the support vectors. Figure from Chapter 3, Theoderidis 'Pattern Recognition'.*

If the best decision line is give by

$$g(\boldsymbol{x}) = w_o + \boldsymbol{x}^T \boldsymbol{w} = 0 \tag{1}$$

the perpendicular distance z is given by

$$z = \frac{|g(\boldsymbol{x})|}{\|\boldsymbol{w}\|} \quad . \tag{2}$$

We can scale **w** and $w_o$ such that the value of g(x) is 1 for $\omega_1$ and -1 for $\omega_2$ . This is equivalent to imposing that

1. The margin given by $\frac{1}{\|\boldsymbol{w}\|} + \frac{1}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$ is maximized;

2. And that $\begin{matrix} w_o + \boldsymbol{x}^T \boldsymbol{w} \geq 1, \forall\, \boldsymbol{x} \in \omega_1 \\ w_o + \boldsymbol{x}^T \boldsymbol{w} \leq -1, \forall\, \boldsymbol{x} \in \omega_2 \end{matrix}$ .

Recall that for each data point $\mathbf{x_i}$ we indicate the class by $y_i$ which is +1 for $\omega_1$ and -1 for $\omega_2$. Our two conditions above can therefore be efficiently rewritten as follows:

minimize :

$$J(\boldsymbol{\omega}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3}$$

subject to:

$$y_i(w_o + \boldsymbol{x}^T \boldsymbol{w}) \geq 1, i = 1, 2, \ldots N \tag{4}$$

Using Lagrange multipliers $\lambda_i$ we can convert the margin maximation problem given by:

$$min\, L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \lambda_i y_i \left(w_o + \boldsymbol{x_i}^T \boldsymbol{w}\right) + \sum_{i=1}^{N} \lambda_i$$

subject to

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i} \quad . \tag{5}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0$$

where N is the number of training points, to a problem involving only dot products given by

 

$$max\, L_D(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j})$$
subject to
$$\sum_{i=1}^{l} \lambda_i y_i = 0 \qquad (6)$$
$$\lambda_i \geq 0.$$

Finding the decision line with the widest margin from the nearest feature vectors using Equation 6 is an optimization problem. Using quadratic problem solver packages we can solve for $\lambda_i$ and use the expression for **w** in Equation 5 to solve for the weights **w** and bias $w_o$.

## Procedure

1. Similar to the Perceptron, SVM is applied to pairs of classes. Read through Reference 2 slides and for each pair of fruits set up the matrices **H**, **B** and **A** , and vectors **f**, **a** and **b** found in slides 19 to 22.
2. Use quadratic programming packages available from Python (numpy and cvxopt) or Matlab (quadprog in Optimization toolbox) using the vectors and matrices setup in Step 1 to solve for the Lagrange multipliers $\lambda_i$ .
3. Solve for the weights **w** and bias $w_o$ and plot the decision lines in feature space with your fruit data.

## Reference

1. Theodoridis, Chapter 3, Pattern Recognition
2. Veksler, O. , CS 434a/541a:Pattern Recognition Lecture 11 slides.

 