Support Vector Machines Explained

Tristan Fletcher

www.cs.ucl.ac.uk/staff/T.Fletcher/

Introduction

This document has been written in an attempt to make the Support Vector Machines (SVM), initially conceived of by Cortes and Vapnik [1], as simple to understand as possible for those with minimal experience of Machine Learning. It assumes basic mathematical knowledge in areas such as calculus, vector geometry and Lagrange multipliers. The document has been split into *Theory* and *Application* sections so that it is obvious, after the maths has been dealt with, how to actually apply the SVM for the different forms of problem that each section is centred on.

The document's first section details the problem of classification for linearly separable data and introduces the concept of margin and the essence of SVM - margin maximization. The methodology of the SVM is then extended to data which is not fully linearly separable. This *soft margin* SVM introduces the idea of slack variables and the trade-off between maximizing the margin and minimizing the number of misclassified variables in the second section. The third section develops the concept of SVM further so that the technique can be used for regression.

The fourth section explains the other salient feature of SVM - the *Kernel Trick*. It explains how incorporation of this mathematical sleight of hand allows SVM to classify and regress nonlinear data.

Other than Cortes and Vapnik [1], most of this document is based on work by Cristianini and Shawe-Taylor [2], [3], Burges [4] and Bishop [5].

For any comments on or questions about this document, please contact the author through the URL on the title page.

1 Linearly Separable Binary Classification

1.1 Theory

We have L training points, where each input $\mathbf{x}_i$ has D attributes (i.e. is of dimensionality D) and is in one of two classes $y_i = $ -1 or +1, i.e our training data is of the form:

$$\{\mathbf{x}_i, y_i\} \quad \text{where} \quad i = 1 \ldots L, \ y_i \in \{-1, 1\}, \ \mathbf{x} \in \Re^D$$

Here we assume the data is linearly separable, meaning that we can draw a line on a graph of x_1 vs x_2 separating the two classes when $D = 2$ and a hyperplane on graphs of $x_1, x_2 \ldots x_D$ for when $D > 2$.

This hyperplane can be described by $\mathbf{w} \cdot \mathbf{x} + b = 0$ where:

- $\mathbf{w}$ is normal to the hyperplane.

- $\frac{b}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin.

Support Vectors are the examples closest to the separating hyperplane and the aim of Support Vector Machines (SVM) is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes.
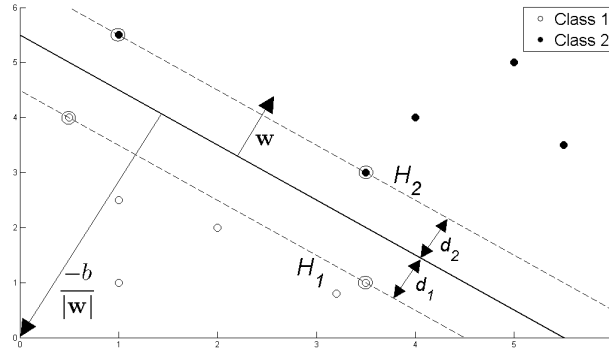

Figure 1: *Hyperplane through two linearly separable classes*

Referring to *Figure 1*, implementing a SVM boils down to selecting the variables $\mathbf{w}$ and b so that our training data can be described by:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1 \tag{1.1}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1 \tag{1.2}$$

These equations can be combined into:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \forall_i \tag{1.3}$$

If we now just consider the points that lie closest to the separating hyperplane, i.e. the Support Vectors (shown in circles in the diagram), then the two planes H_1 and H_2 that these points lie on can be described by:

$$\mathbf{x}_i \cdot \mathbf{w} + b = +1 \quad \text{for } H_1 \tag{1.4}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b = -1 \quad \text{for } H_2 \tag{1.5}$$

Referring to *Figure 1*, we define d_1 as being the distance from H_1 to the hyperplane and d_2 from H_2 to it. The hyperplane's ==equidistance== from H_1 and H_2 means that $d_1 = d_2$ - a quantity known as the ==SVM's *margin.*== In order to orientate the hyperplane to be as far from the Support Vectors as possible, we need to maximize this margin.

Simple vector geometry shows that the margin is equal to $\frac{1}{\|\mathbf{w}\|}$ and maximizing it subject to the constraint in (1.3) is equivalent to finding:

$$\min \|\mathbf{w}\| \quad \text{such that} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \ \forall_i$$

Minimizing $\|\mathbf{w}\|$ is equivalent to minimizing $\frac{1}{2}\|\mathbf{w}\|^2$ and the use of this term makes it possible to perform Quadratic Programming (QP) optimization later on. We therefore need to find:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \ \forall_i \tag{1.6}$$

In order to cater for the constraints in this minimization, we need to allocate them Lagrange multipliers $\boldsymbol{\alpha}$, where $\alpha_i \geq 0 \ \forall_i$:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \boldsymbol{\alpha}\left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \ \forall_i\right] \tag{1.7}$$

$$\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i\left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right] \tag{1.8}$$

$$\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{L} \alpha_i \tag{1.9}$$

We wish to find the $\mathbf{w}$ and b which minimizes, and the $\boldsymbol{\alpha}$ which maximizes (1.9) (whilst keeping $\alpha_i \geq 0 \ \forall_i$). We can do this by differentiating L_P with respect to $\mathbf{w}$ and b and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i \tag{1.10}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.11}$$

3

Substituting (1.10) and (1.11) into (1.9) gives a new formulation which, being dependent on $\boldsymbol{\alpha}$, we need to maximize:

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \text{ s.t. } \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.12}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j \text{ where } H_{ij} \equiv y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{1.13}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \text{ s.t. } \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.14}$$

This new formulation L_D is referred to as the *Dual* form of the *Primary* L_P. It is worth noting that the Dual form requires only the dot product of each input vector x_i to be calculated, this is important for the *Kernel Trick* described in the fourth section.

Having moved from minimizing L_P to maximizing L_D, we need to find:

$$\max_{\alpha} \left[\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \right] \quad \text{s.t.} \quad \alpha_i \geq 0 \;\; \forall_i \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.15}$$

This is a convex quadratic optimization problem, and we run a QP solver which will return $\boldsymbol{\alpha}$ and from (1.10) will give us $\mathbf{w}$. What remains is to calculate b.

Any data point satisfying (1.11) which is a Support Vector x_s will have the form:

$$y_s(\mathbf{x}_s \cdot \mathbf{w} + b) = 1$$

Substituting in (1.10):

$$y_s \left(\sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s + b \right) = 1$$

Where S denotes the set of indices of the Support Vectors. S is determined by finding the indices i where $\alpha_i > 0$. Multiplying through by y_s and then using $y_s^2 = 1$ from (1.1) and (1.2):

$$y_s^2 \left(\sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s + b \right) = y_s$$

$$b = y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s$$

Instead of using an arbitrary Support Vector $\mathbf{x}_s$, it is better to take an average over all of the Support Vectors in S:

$$b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s) \tag{1.16}$$

We now have the variables $\mathbf{w}$ and b that define our separating hyperplane's optimal orientation and hence our Support Vector Machine.

1.2 Application

In order to use an SVM to solve a linearly separable, binary classification problem we need to:

- Create $\mathbf{H}$, where $H_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$.

- Find $\boldsymbol{\alpha}$ so that

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}$$

  is maximized, subject to the constraints

$$\alpha_i \geq 0 \ \ \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

  This is done using a QP solver.

- Calculate $\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i$.

- Determine the set of Support Vectors S by finding the indices such that $\alpha_i > 0$.

- Calculate $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s)$.

- Each new point $\mathbf{x}'$ is classified by evaluating $y' = sgn(\mathbf{w} \cdot \mathbf{x}' + b)$.

2 Binary Classification for Data that is not Fully Linearly Separable

2.1 Theory

In order to extend the SVM methodology to handle data that is not fully linearly separable, we relax the constraints for (1.1) and (1.2) slightly to allow for misclassified points. This is done by introducing a positive slack variable $\xi_i, \ i = 1, \ldots L$:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \tag{2.1}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \quad \text{for } y_i = -1 \tag{2.2}$$

$$\xi_i \geq 0 \ \forall_i \tag{2.3}$$

Which can be combined into:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0 \quad \text{where} \quad \xi_i \geq 0 \ \forall_i \tag{2.4}$$
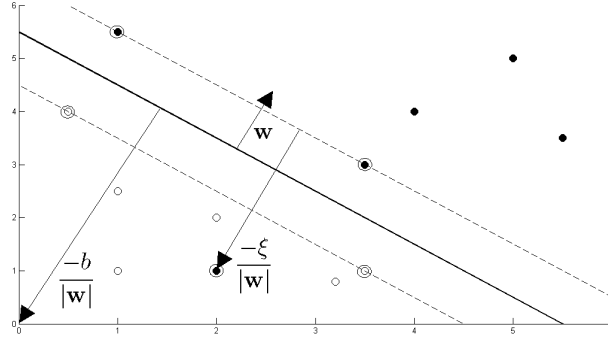

Figure 2: *Hyperplane through two non-linearly separable classes*

In this *soft margin* SVM, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. As we are trying to reduce the number of misclassifications, a sensible way to adapt our objective function (1.6) from previously, is to find:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{L} \xi_i \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0 \ \ \forall_i \tag{2.5}$$

Where the parameter C controls the trade-off between the slack variable penalty and the size of the margin. Reformulating as a Lagrangian, which as before we need to minimize with respect to $\mathbf{w}$, b and ξ_i and maximize

with respect to $\boldsymbol{\alpha}$ (where $\alpha_i \geq 0$, $\mu_i \geq 0 \; \forall_i$):

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\alpha_i\left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\right] - \sum_{i=1}^{L}\mu_i\xi_i \quad (2.6)$$

Differentiating with respect to $\mathbf{w}$, b and ξ_i and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L}\alpha_i y_i \mathbf{x}_i \qquad (2.7)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L}\alpha_i y_i = 0 \qquad (2.8)$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \mu_i \qquad (2.9)$$

Substituting these in, L_D has the same form as (1.14) before. However (2.9) together with $\mu_i \geq 0 \; \forall_i$, implies that $\alpha \geq C$. We therefore need to find:

$$\max_{\alpha}\left[\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H}\boldsymbol{\alpha}\right] \quad \text{s.t.} \quad 0 \leq \alpha_i \leq C \; \forall_i \quad \text{and} \quad \sum_{i=1}^{L}\alpha_i y_i = 0$$

$$(2.10)$$

b is then calculated in the same way as in (1.6) before, though in this instance the set of Support Vectors used to calculate b is determined by finding the indices i where $0 < \alpha_i < C$.

2.2 Application

In order to use an SVM to solve a binary classification for data that is not fully linearly separable we need to:

- Create $\mathbf{H}$, where $H_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$.

- Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C.

- Find $\boldsymbol{\alpha}$ so that

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}$$

  is maximized, subject to the constraints

$$0 \le \alpha_i \le C \quad \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

  This is done using a QP solver.

- Calculate $\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i$.

- Determine the set of Support Vectors S by finding the indices such that $0 < \alpha_i < C$.

- Calculate $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s)$.

- Each new point $\mathbf{x}'$ is classified by evaluating $y' = sgn(\mathbf{w} \cdot \mathbf{x}' + b)$.

3 Support Vector Machines for Regression

3.1 Theory

Instead of attempting to classify new unseen variables $\mathbf{x}'$ into one of two categories $y' = \pm 1$, we now wish to predict a real-valued output for y' so that our training data is of the form:

$\{\mathbf{x}_i, y_i\}$ where $i = 1 \ldots L$, $y_i \in \Re$, $\mathbf{x} \in \Re^D$
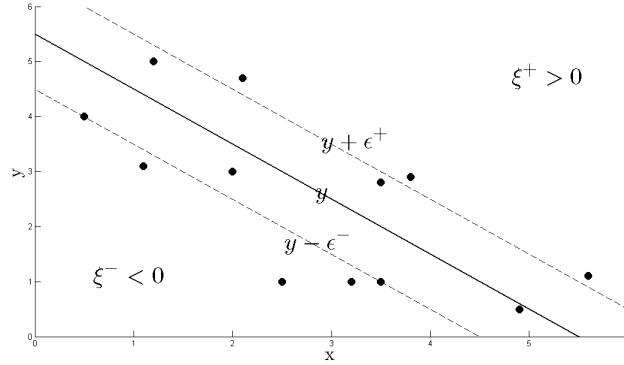
$$y_i = \mathbf{w} \cdot \mathbf{x}_i + b \tag{3.1}$$

Figure 3: *Regression with ϵ-insensitive tube*

The regression SVM will use a more sophisticated penalty function than before, not allocating a penalty if the predicted value y_i is less than a distance ϵ away from the actual value t_i, i.e. if $|t_i - y_i| < \epsilon$. Referring to *Figure 3*, the region bound by $y_i \pm \epsilon$ $\forall_i$ is called an ϵ-insensitive tube. The other modification to the penalty function is that output variables which are outside the tube are given one of two slack variable penalties depending on whether they lie above (ξ^+) or below (ξ^-) the tube (where $\xi^+ > 0$, $\xi^- > 0$ $\forall_i$):

$$t_i \leq y_i + \epsilon + \xi^+ \tag{3.2}$$
$$t_i \geq y_i - \epsilon - \xi^- \tag{3.3}$$

The error function for SVM regression can then be written as:

$$C \sum_{i=1}^{L} (\xi_i^+ + \xi_i^-) + \frac{1}{2} \|\mathbf{w}\|^2 \tag{3.4}$$

This needs to be minimized subject to the constraints $\xi^+ \geq 0$, $\xi^- \geq 0$ $\forall_i$ and (3.2) and (3.3). In order to do this we introduce Lagrange multipliers

$\alpha_i^+ \geq 0, \ \alpha_i^- \geq 0, \mu_i^+ \geq 0 \ \mu_i^- \geq 0 \ \forall_i$:

$$L_P = C\sum_{i=1}^{L}(\xi_i^+ + \xi_i^-) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L}(\mu_i^+\xi_i^+ + \mu_i^-\xi_i^-) - \sum_{i=1}^{L}\alpha_i^+(\epsilon + \xi_i^+ + y_i - t_i) - \sum_{i=1}^{L}\alpha_i^-(\epsilon + \xi_i^- - y_i + t_i) \tag{3.5}$$

Substituting for y_i, differentiating with respect to $\mathbf{w}$, b, ξ^+ and ξ^- and setting the derivatives to 0:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i \tag{3.6}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) = 0 \tag{3.7}$$

$$\frac{\partial L_P}{\partial \xi_i^+} = 0 \Rightarrow C = \alpha_i^+ + \mu_i^+ \tag{3.8}$$

$$\frac{\partial L_P}{\partial \xi_i^-} = 0 \Rightarrow C = \alpha_i^- + \mu_i^- \tag{3.9}$$

Substituting (3.6) and (3.7) in, we now need to maximize L_D with respect to α_i^+ and α_i^- ($\alpha_i^+ \geq 0, \ \alpha_i^- \geq 0 \ \forall_i$) where:

$$L_D = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)t_i - \epsilon\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\mathbf{x}_i \cdot \mathbf{x}_j \tag{3.10}$$

Using $\mu_i^+ \geq 0$ and $\mu_i^- \geq 0$ together with (3.8) and (3.9) means that $\alpha_i^+ \leq C$ and $\alpha_i^- \leq C$. We therefore need to find:

$$\max_{\alpha^+,\alpha^-}\left[\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)t_i - \epsilon\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\mathbf{x}_i \cdot \mathbf{x}_j\right] \tag{3.11}$$

such that $0 \leq \alpha_i^+ \leq C$, $0 \leq \alpha_i^- \leq C$ and $\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) = 0 \ \forall_i$.

Substituting (3.6) into (3.1), new predictions y' can be found using:

$$y' = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i \cdot \mathbf{x}' + b \tag{3.12}$$

A set S of Support Vectors $\mathbf{x}_s$ can be created by finding the indices i where $0 < \alpha < C$ and $\xi_i^+ = 0$ (or $\xi_i^- = 0$).

11

This gives us:

$$b = t_s - \epsilon - \sum_{m\in=S}^{L} (\alpha_m^+ - \alpha_m^-)\mathbf{x}_m \cdot \mathbf{x}_s \tag{3.13}$$

As before it is better to average over all the indices i in S:

$$b = \frac{1}{N_s} \sum_{s\in S} \left[t_s - \epsilon - \sum_{m\in=S}^{L} (\alpha_m^+ - \alpha_m^-)\mathbf{x}_m \cdot \mathbf{x}_s \right] \tag{3.14}$$

12

3.2 Application

In order to use an SVM to solve a regression problem we need to:

- Choose how significantly misclassifications should be treated and how large the insensitive loss region should be, by selecting suitable values for the parameters C and ϵ.

- Find $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$ so that:

$$\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)t_i - \epsilon\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\mathbf{x}_i \cdot \mathbf{x}_j$$

  is maximized, subject to the constraints

$$0 \le \alpha_i^+ \le C,\, 0 \le \alpha_i^- \le C \text{ and } \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) = 0 \; \forall_i.$$

  This is done using a QP solver.

- Calculate $\mathbf{w} = \displaystyle\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i$.

- Determine the set of Support Vectors S by finding the indices i where $0 < \alpha < C$ and $\xi_i = 0$.

- Calculate

$$b = \frac{1}{N_s}\sum_{s \in S}\left[t_i - \epsilon - \sum_{m=1}^{L}(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i \cdot \mathbf{x}_m\right].$$

- Each new point $\mathbf{x}'$ is determined by evaluating

$$y' = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i \cdot \mathbf{x}' + b.$$

4 Nonlinear Support Vector Machines

4.1 Theory

When applying our SVM to linearly separable data we have started by creating a matrix $\mathbf{H}$ from the dot product of our input variables:

$$H_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i^T \mathbf{x}_j \tag{4.1}$$

$k(\mathbf{x}_i, \mathbf{x}_j)$ is an example of a family of functions called *Kernel Functions* ($k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ being known as a Linear Kernel). The set of kernel functions is composed of variants of (4.2) in that they are all based on calculating inner products of two vectors. This means that if the functions can be recast into a higher dimensionality space by some potentially non-linear feature mapping function $\mathbf{x} \longmapsto \phi(\mathbf{x})$, only inner products of the mapped inputs in the feature space need be determined without us needing to explicitly calculate ϕ.

The reason that this *Kernel Trick* is useful is that there are many classification/regression problems that are not linearly separable/regressable in the space of the inputs $\mathbf{x}$, which might be in a higher dimensionality feature space given a suitable mapping $\mathbf{x} \longmapsto \phi(\mathbf{x})$.

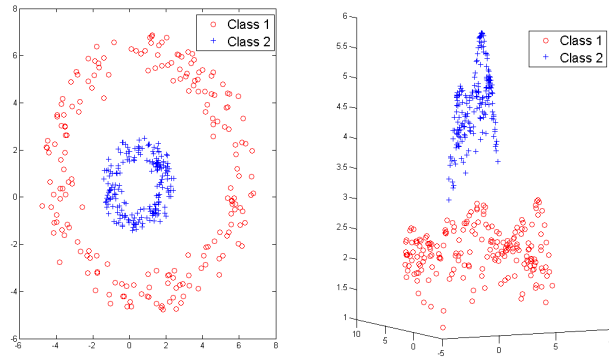
Figure 4: *Dichotomous data re-mapped using Radial Basis Kernel*

Refering to *Figure 4*, if we define our kernel to be:

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right)} \tag{4.2}$$

then a data set that is not linearly separable in the two dimensional data space $\mathbf{x}$ (as in the left hand side of *Figure 4*) is separable in the nonlinear

feature space (right hand side of *Figure 4*) defined implicitly by this non-linear kernel function - known as a *Radial Basis Kernel*.

Other popular kernels for classification and regression are the *Polynomial Kernel*

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + a)^b$$

and the *Sigmoidal Kernel*

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a\mathbf{x}_i \cdot \mathbf{x}_j - b)$$

where a and b are parameters defining the kernel's behaviour.

There are many kernel functions, including ones that act upon sets, strings and even music. There are requirements for a function to be applicable as a kernel function that lie beyond the scope of this very brief introduction to the area. The author therefore recomends sticking with the three mentioned above to start with.

4.2 Application

In order to use an SVM to solve a classification or regression problem on data that is not linearly separable, we need to first choose a kernel and relevant parameters which you expect might map the non-linearly separable data into a feature space where it is linearly separable. This is more of an art than an exact science and can be achieved empirically - e.g. by trial and error. Sensible kernels to start with are the Radial Basis, Polynomial and Sigmoidal kernels.

The first step, therefore, consists of choosing our kernel and hence the mapping $\mathbf{x} \longmapsto \phi(\mathbf{x})$.

For classification, we would then need to:

- Create $\mathbf{H}$, where $H_{ij} = y_i y_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$.

- Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C.

- Find $\boldsymbol{\alpha}$ so that

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}$$

is maximized, subject to the constraints

$$0 \leq \alpha_i \leq C \ \ \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

This is done using a QP solver.

- Calculate $\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \phi(\mathbf{x}_i)$.

- Determine the set of Support Vectors S by finding the indices such that $0 < \alpha_i < C$.

- Calculate $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \phi(\mathbf{x}_m) \cdot \phi(\mathbf{x}_s))$.

- Each new point $\mathbf{x}'$ is classified by evaluating $y' = sgn(\mathbf{w} \cdot \phi(\mathbf{x}') + b)$.

For regression, we would then need to:

- Choose how significantly misclassifications should be treated and how large the insensitive loss region should be, by selecting suitable values for the parameters C and ϵ.

- Find $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$ so that:

$$\sum_{i=1}^{L}(\alpha_i^+-\alpha_i^-)t_i-\epsilon\sum_{i=1}^{L}(\alpha_i^+-\alpha_i^-)-\frac{1}{2}\sum_{i,j}(\alpha_i^+-\alpha_i^-)(\alpha_j^+-\alpha_j^-)\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$$

  is maximized, subject to the constraints

$$0\leq\alpha_i^+\leq C,\ 0\leq\alpha_i^-\leq C\ \text{and}\ \sum_{i=1}^{L}(\alpha_i^+-\alpha_i^-)=0\ \forall_i.$$

  This is done using a QP solver.

- Calculate $\mathbf{w}=\sum_{i=1}^{L}(\alpha_i^+-\alpha_i^-)\phi(\mathbf{x}_i)$.

- Determine the set of Support Vectors S by finding the indices i where $0<\alpha<C$ and $\xi_i=0$.

- Calculate

$$b=\tfrac{1}{N_s}\sum_{s\in S}\left[t_i-\epsilon-\sum_{m=1}^{L}(\alpha_i^+-\alpha_i^-)\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_m)\right].$$

- Each new point $\mathbf{x}'$ is determined by evaluating

$$y'=\sum_{i=1}^{L}(\alpha_i^+-\alpha_i^-)\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}')+b.$$

References

[1] C. Cortes, V. Vapnik, in *Machine Learning*, pp. 273–297 (1995).

[2] N. Cristianini, J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, New York, NY, USA (2000).

[3] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge University Press, New York, NY, USA (2004).

[4] C. J. C. Burges, *Data Mining and Knowledge Discovery* **2**, 121 (1998).

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer (2006).