Support Vector Machine

Mads Møller

LinkedIn: https://www.linkedin.com/in/madsmoeller1/

Mail: mads.moeller@outlook.com

This paper is the third in the series about machine learning algorithms. The Support Vector Machine (SVM) algorithm can both be used for *classification* problems as well as for *regression* problems. In this paper we will focus on how SVM can be used for classification problems. SVM is like linear- and logistic regression also a supervised machine learning algorithm, meaning that our data need to be labelled. There are also a subset of SVM called SVR (regression) which uses the same principles to solve regression problems.

The core objective of SVM is to find an optimal **hyperplane** which linearly separates two classes by maximizing the **margin** (we will come back to this part).

1 Linear Algebra for SVM

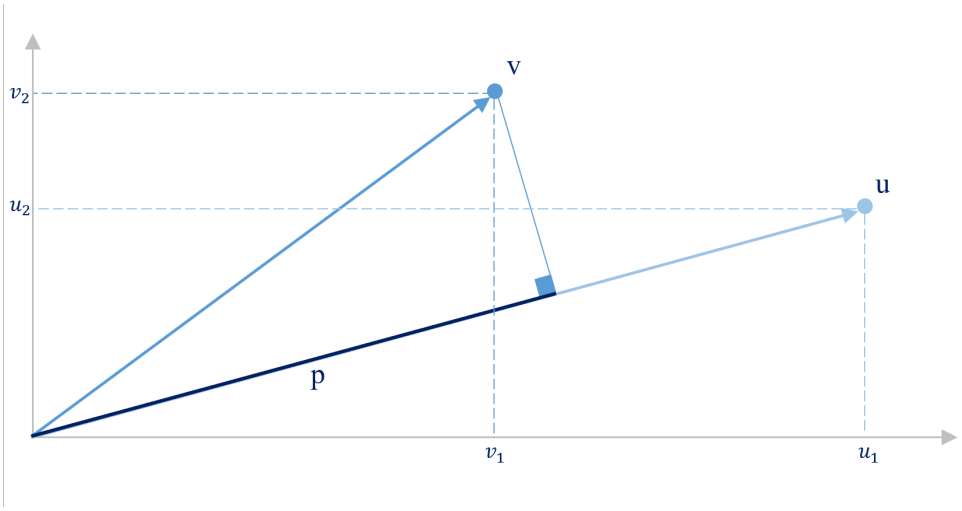

Figure 1: Length of orthogonal projection

In order to understand how SVM works we need to brush up some basic linear algebra. First, the *norm* or *length* of a vector $\underline{x}(x_1, x_2, x_3)$ can be calculated as:

$$||\underline{x}|| = \sqrt{x_1^2 + x_2^2 + x_3^2} \tag{1}$$

The inner product between two vectors $\underline{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $\underline{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ can be calculated as:

$$\underline{u}^T \underline{v} = p \cdot ||\underline{u}|| = u_1 v_1 + u_2 v_2 = \underline{v}^T \underline{u} \tag{2}$$

Where $p \in \mathbb{R}$ is the length of the orthogonal projection of $\underline{v}$ onto $\underline{u}$ as illustrated in figure 1.

2 Intuition

A **hyperplane** is a plane, or imagine a cut, that separates the n-dimensional data-points into two classes. The simplest hyperplane is 2-dimensional, so a linear line. If we have a 3-dimensional hyperplane we can imagine it as a tabletop separating points. When we go to n-dimensions it is hard for our brains to imagine how the hyperplane looks like but the principle is identical with 2- and 3-dimensional data. SVM *linearly* separates data into two components. So let us see how this looks like in figure 2.

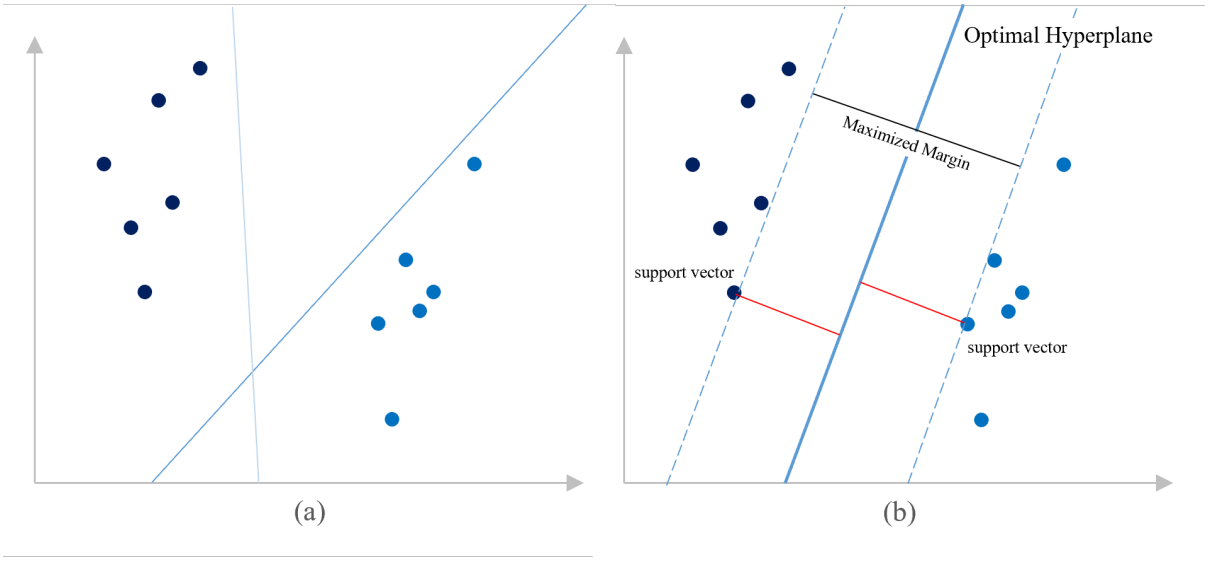

Figure 2: Support Vector Machine Intuition

As we see in figure 2 (a) the data can be linearly separated in an infinite number of ways. The problem of SVM is to find the ideal line. In the figure neither of these two lines are ideal. But how do we find the hyperplane (line) that best separates the two classes? Let us take a look at figure 2 (b). This is the hyperplane that best separates the two classes. Why?, because the margin from the two classes are maximized at exactly this hyperplane.

3 Optimal Hyperplane (Decision boundary)

In order to find the hyperplane which maximizes the margin we need to mathematically formulate our problem, or objective function. SVM is actually a constrained optimization problem. Let us see how the objective of SVM can be formulated:

$$\min_{\omega} = \frac{1}{2} \sum_{j=1}^{n} \omega_j^2 \tag{3}$$

s.t.

$$\omega^T x_{(i)} + b \geq 1 \qquad \text{if } y_{(i)} = 1$$

$$\omega^T x_{(i)} + b \leq -1 \qquad \text{if } y_{(i)} = 0$$

This formulation of SVM is called the *hard margin* (because we demand all observations to be correctly classified). We can also rearrange our constraints in the following form:

$$y_{(i)}(\omega \cdot x_{(i)} + b) \geq 1$$

In words what this optimization problem from equation 4 says is "*maximizing the margin by minimizing the objective function*". In a moment we will dive into why this expression is maximizing the margin. Let us take an example where we have *two explanatory variables*. We can then reformulate the problem using the mathematics we defined in section 1. In this example we assume that the intercept is zero, so $b = 0$:

$$\min_{\omega} = \frac{1}{2}\sum_{j=1}^{n}\omega_j^2 = \frac{1}{2}(\omega_1^2 + \omega_2^2) = \frac{1}{2}(\sqrt{\omega_1^2 + \omega_2^2})^2 = \frac{1}{2}||\underline{\omega}||^2 \tag{4}$$

s.t.

$$\omega^T x_{(i)} = p_{(i)}||\omega|| \geq 1 \qquad \text{if } y_{(i)} = 1$$

$$\omega^T x_{(i)} = p_{(i)}||\omega|| \leq -1 \qquad \text{if } y_{(i)} = 0$$

Where $p_{(i)}$ is the projection for each observation $x_{(i)}$ on the vector ω. So we can see that SVM is really all about maximizing the margin because the constraints can only be fulfilled if the margin is large.

4 Soft Margin Formulation and Dual Form

Until now we have assumed that the data **is** linearly separable. This is often not the case. So let us formulate SVM where not all observations needs to be correctly classified:

$$\min_{\omega,\xi} = \frac{1}{2}\sum_{j=1}^{n}\omega_j^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{5}$$

s.t.

$$y_i(\omega^T x_i) \geq 1 - \xi_i \qquad i = 1, \ldots, n$$

$$\omega = \begin{bmatrix} \omega_0 \\ \vdots \\ \omega_k \end{bmatrix} \in \mathbb{R}^{k+1}$$

$$\xi_i \geq \qquad\qquad i = 1, ..., n$$

In the soft margin formulation we allow observations to be inside the margin and allow miss-classifications. If $\xi_i = 0$ the individual observation $x_{(i)}$ are correctly classified. Else if $0 \leq \xi_i \leq 1$, the observation $x_{(i)}$ is inside the margin but at the correct side of the hyperplane. Here imagine a point inside the margin on the correctly classified side on figure 2 (b). Else $\xi_i > 1$, the observation is miss-classified. We call the hyperparameter C the **trade-off parameter** between the margin and correct classifications. This is exactly what the parameter controls which is obvious when we look at equation 5.

However, when later introducing kernels in SVM we need to write the optimization problem of SVM on what is called **the dual problem**. This requires a strong mathematical background to fully understand, knowledge of lagrange multipliers and KKT. However, I will try to explain the dual formulation of SVM as easy as possible. The dual formulation looks like this:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{6}$$

s.t.

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq \frac{C}{n} \qquad i = 1, \ldots, n$$

Here $\alpha > 0$ are the support vectors. So, only the support vectors matters. Support vectors are observations which intercepts with the margin of the decision boundary, so the observations closest to the margin. I have illustrated the two support vectors on figure 2 (b). If a point is a support vector then $\alpha > 0$. If a point is not a support vector then $\alpha = 0$. Lets expect equation 6. The last part of the objective function $(x_i^T x_j)$ is the dot product between one point and the rest of the data points. This dot-product measures the similarity between one observation and the rest of the observations. So, in the dual formulation only the support vectors and observations that are not correctly classified matter. When a solution for the dual problem is found a solution to the primal problem can easily be found:

$$\omega^* = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{7}$$

5 Kernel Trick for SVM

In general, the best way to separate data into two classes is not linear. However, SVM support **kernels**. Kernels are functions we use to transform data. We will see an example graphically later on how this could look like. Until now we have worked with what is called a linear kernel. When we use a linear kernel we therefore work directly on the dataset. However, when using another kernel than linear, SVM works on a transformed dataset. Like neural networks SVM often becomes a black-box because we cannot fully explain what the model does. To formulate SVM with kernels we need to formulate it as a dual problem as well:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{8}$$

s.t.

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq \frac{C}{n} \qquad i = 1, \ldots, n$$

As you can see the dual problem is very similar with the soft margin dual formulation we saw in equation 6. Now the dot product has been switched out with a kernel, $K(\cdot, \cdot)$. As mentioned earlier a kernel transforms the data. So now the similarity between x_i and x_j is measured in a transformed space. Know,

why does this make sense? So, the kernel trick can also be to add another dimension to the data. Let us take an example. If we inspect figure 3 below:

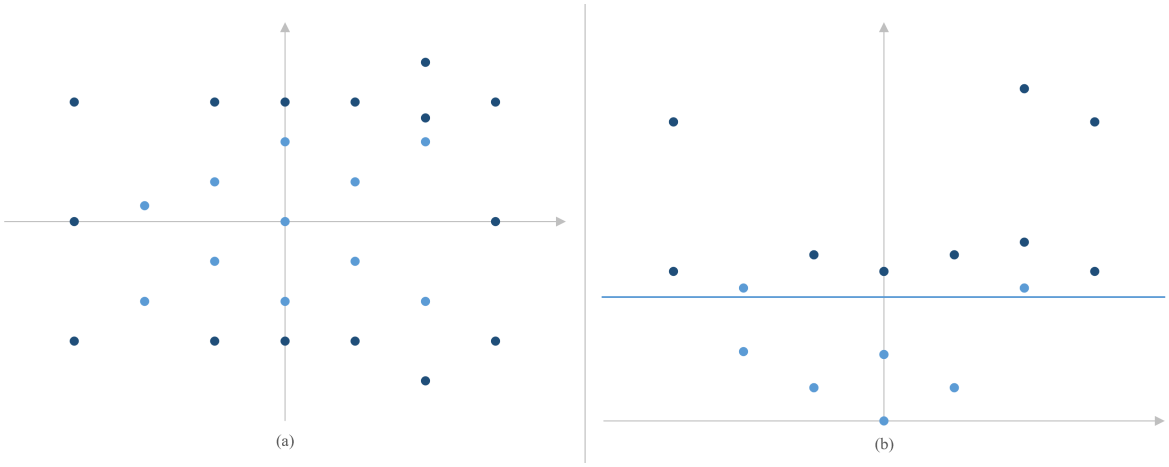

Figure 3: Transformation to higher dimension

When we look a figure 3(a) it is clear that this dataset is not linear separable. However, we can actually just add another dimension to the dataset in order to make it linear separable. To make this transformation we have added another dimension $z = x^2 + y^2$ to our data. So figure 3 (b) actually has a third dimension z, just imagine that this how you see the data points in this three dimensional space from one side. By simply adding one more dimension our data is now linearly separable again! In real life the SVM algorithm is not this simple often you get a lot more dimensions or the SVM uses advanced kernels. Hopefully the idea behind adding more dimensions and kernels are now in place. The last thing we need to cover in this paper about SVM is **kernels**. Because I have not really introduced any kernels, so let us take a look at the default kernel in most ML libraries.

6 Kernels

As mentioned earlier kernels are ways we can transform our data in order to make it more linear separable. The default kernel in most ML libraries are called **Radial Basis Function (RBF)**. The RBF kernel for two points x_i and x_j computes the similarity or how close the two observations are on each other. The mathematical representation of this kernel is:

$$K(x_i, x_j) = exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \tag{9}$$

Where $||x_i - x_j||$ is the euclidean norm between our two points x_i and x_j. This norm measures the distance between two points. The maximum value of the RBF kernel is 1. This only happens when $||x_i - x_j||$ is zero, meaning that the points are the same (similar). If the value of the RBF kernel is near zero the points are very dissimilar, meaning that the points are very far from each other. Distance can be thought of as an equivalent to dissimilarity. When inspecting equation 9 we also see the parameter σ. This is a hyperparameter and it decides when a point should be considered similar. Let us take some examples to illustrate the value of σ:

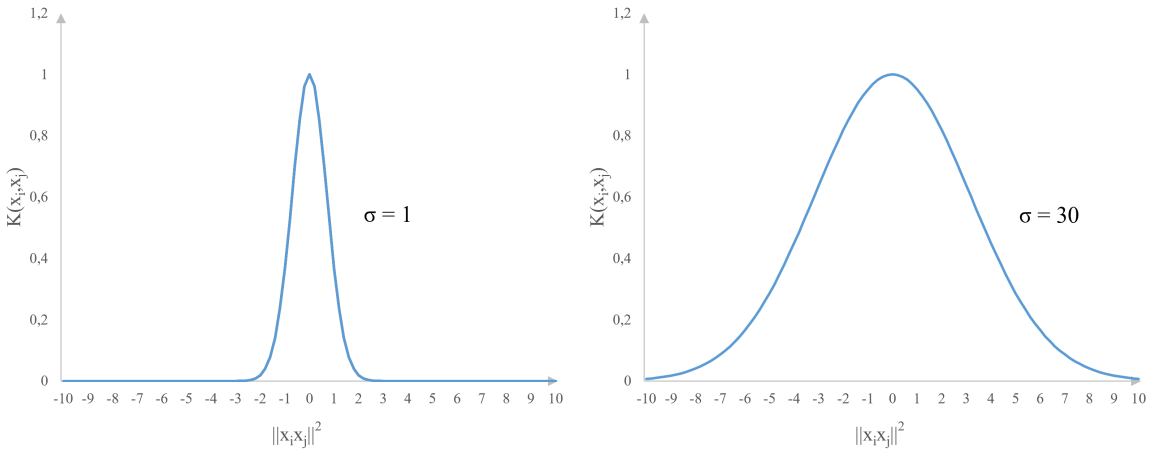

Figure 4: RBF illustration of σ

As we can see the value of σ has a huge effect of when data points are considered similar or not. Normally the σ parameter will be chosen with hyperparameter tuning in a software that can handle SVM. Of course there are lots of other kernels that we will not cover in this paper. The reason of why RBF is the default kernel is because it often performs best when you don't know how your data looks like.