Support Vector Machine

Alex Baier

I. Introduction

In this paper the concept of Support Vector Machines is introduced. SVM is a supervised learning method, which is able to solve the classification problem. The idea on what a classifier is and how the method creates it are explained geometrically as it is more intuitive, while the mathematical formula of SVM is to some degree derived. On a further note the "Kernel Trick" is discussed, which enables SVM to solve non-linear problems as the original method only creates linear classifiers. At last practical applications of SVM are discussed, the main point of this discussion is the effectiveness of SVM in comparison to other methods.

II. Classification Problem

First the notion of *supervised learning* has to be explained. A supervised learning algorithm uses *training data* to create a function, which solves a specific problem. The training data contains multiple samples, which are pairs of input and expected output of the function, which the algorithm has to create. For example assume the problem of gender recognition in pictures. This is actually a binary classification problem as a function, which would want to solve the problem has to classify an entered picture as either male or female. In this example the training data would be pairs of images of persons and their supposed gender. In the case of a classification problem the expected output are called *class labels*. A supervised learning algorithm would take this training data and create function, which is able to assign to each of those pictures the correct gender. To prove that the generated function actually does a good job in classifying, one would need *test data*. The test data has the same form as the training data, but the training and testing set have to be completely disjunct. This is necessary as it can be expected that the function does return perfect results for the training data. The effectiveness of the created function and therefore the algorithm can be measured by comparing the actual results of the function with the expected results of the test data.

Let us now take a closer look at the training data. The input can be described by a feature vector $\vec{u} \in \mathbb{R}^d$, where $d \in \mathbb{N}$ is the number of features of the object. Assuming the number of classes is K we now need to find a function $f : \mathbb{R}^d \to \{0, K-1\}$, which takes a feature vector and assigns the correct class [2]. SVM in its basic form can only solve the binary classification problem ($K = 2$). This means we can simplify the signature of the function, which we want to construct, to $f : \mathbb{R}^d \to \{-1, 1\}$. The training data has the following form: $(\vec{x_i}, y_i), i \in [1, l], \vec{x_i} \in \mathbb{R}^d, y_i \in \{-1, 1\}$ [2].

III. Linear classifier

The classifier, which was defined in the previous section, is called a *(binary) linear classifier*, if its function is composed like this:

$$f(\vec{u}) = \langle \vec{w}, \vec{u} \rangle + b \tag{1}$$

In this case $\vec{w}$ describes the orientation of a plane and b its offset [1]. The training data $(\vec{x_i}, y_i), i \in [1, l], \vec{x_i} \in \mathbb{R}^d, y_i \in \{-1, 1\}$ is called *linear separable*, if a linear classifier exists, so that $y_i * f(\vec{x}) > 0, \forall i = 1, ..., l$ applies.

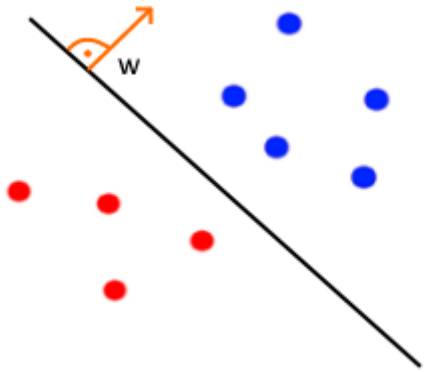

Figure 1. Linear classifier in 2d-space.

In the following it is assumed that the training data is *linear separable*. This means there exists a linear classifier separating the two classes of the training data. Geometrically we can say the classes represents two sets of points in the space, which can be separated by a hyperplane, as it is shown in Figure 1. If the opposite is the case the training data can be called *linear inseparable*. The SVM method can also create linear classifiers with linear inseparable data, but for this a more complex approach is needed, which also depends on the exact structure of the training data. The solution for non-linear classification, which called the "Kernel Trick" will be shown in a later section.

IV. Maximal margin in the linear-separable case

It can be easily seen that there are many different possible linear classifiers for the same set of training data. But the question has to be asked, which of those will fit the unknown feature vectors with the least amount of errors. With just looking at Figure 2, it can be intuitively seen that the black-colored plane is "better" than the other two, because it is the furthest from both sets of points. Would we use the green

plane for classification and we try to classify a vector, which normally would belong to the red color. It could easily be classified into the blue group, because the plane has such close proximity to the set of red points.

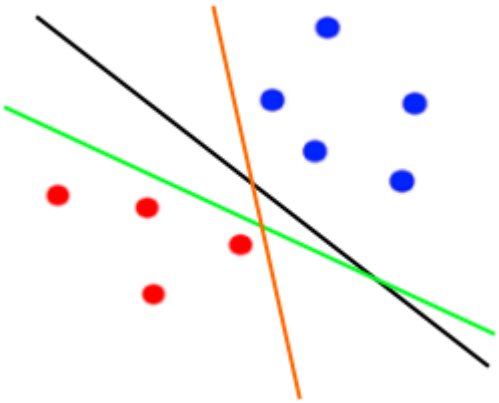

Figure 2. Multiple possible classifiers.

Therefore we come to the intuition that the best linear classifier has the biggest distance, so called margin, to both classes of the training data. This idea is also supported by the mathematical theorems in [2] and by reasoning about the "thickness" of planes in [1].

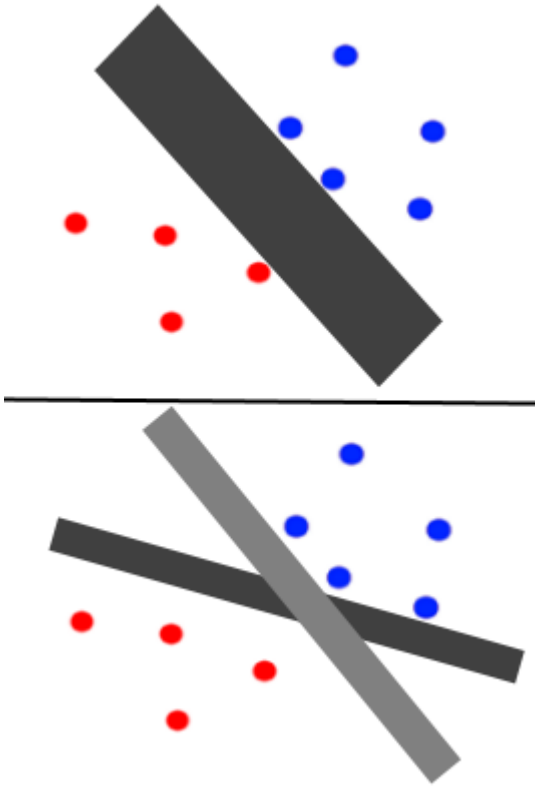

Figure 3. Thick vs. thin margins.

If we compare a separating plane with a "thin" margin to one with a "fat" margin in Figure 3, we can see that the "thin" plane has many possible orientations and is therefore more complex, while on the opposite the "fat" plane has only one possible orientation. Because the complexity of the plane is lower, the possibility of mislabeling new data should be lower.

To find a way of constructing such a plane with maximal margin, we first need to take a look at the structure of the problem. We assume that the two class sets are linear separable and therefore we can points in the sets, which are closest to the points in the other sets. We will call these points *support vectors*, because they will be part of the parallel support planes, of which we try to maximize the margin. The decision rule for the linear classifier is as follows:

$$\text{IF } f(\vec{u}) = \langle \vec{w}, \vec{u} \rangle + b > 0 \text{ THEN } +1 \tag{2}$$

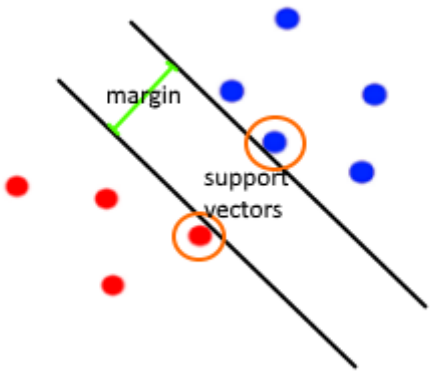

Figure 4. Parallel support planes.

for unknown data $\vec{u} \in \mathbb{R}^d$.

Based on this decision rule, we can create constraints for the training data. Let $\vec{x_+}$ be $+1$-training sample and $\vec{x_-}$ be a -1-training sample.

$$\langle \vec{w}, \vec{x_+} \rangle + b \geq 1$$
$$\langle \vec{w}, \vec{x_-} \rangle + b \leq -1$$

We can introduce the corresponding class label y_i to the constraints, where $y_i = +1$ for $+1$-samples and $y_i = -1$ for -1-samples.

$$y_i * (\langle \vec{w}, \vec{x_+} \rangle + b) \geq 1$$
$$y_i * (\langle \vec{w}, \vec{x_-} \rangle + b) \geq 1$$

As it is easily to see both constraints are now equal and therefore only a single constraints stays for all training samples.

$$y_i * (\langle \vec{w}, \vec{x_i} \rangle + b) \geq 1 \tag{3}$$

Because the supporting planes contain exactly those points, which return 1 or -1 for the decision rule, they can be described the following equations [1]:

$$\langle \vec{w}, \vec{x} \rangle + b = 1$$
$$\langle \vec{w}, \vec{x} \rangle + b = -1$$

Therefore the margin between the parallel planes is $\gamma = \frac{2}{||\vec{w}||}$. To find the maximum margin plane, we need to maximize γ and therefore we need to minimize $||\vec{w}||$. This leads to the following quadratic program [1]:

$$\min \frac{1}{2} * ||\vec{w}||_2^2 = \min \frac{1}{2} * \langle \vec{w}, \vec{w} \rangle \quad (4)$$
$$\text{constrained by } y_i * (\langle \vec{w}, \vec{x_i} \rangle + b) \geq 1 \; \forall i = 1...l$$

Optimization problems with constraints like the one we encountered now can be solved by using Lagrangian multipliers [2]. We essentially create a single function out of the quadratic problem by introducing the mentioned l Lagrangian multipliers $\alpha_i \; \forall i = 1...l$. The resulting Lagrangian function looks like this:

$$L = \frac{1}{2} * \langle \vec{w}, \vec{w} \rangle - \sum_{i=1}^{l} \alpha_i * (y_i * (\langle \vec{w}, \vec{x_i} \rangle + b) - 1) \quad (5)$$

Because we want to find the extrema of this function L, we create its derivate regarding $\vec{v}$ and b. Then we solve them for 0 [2].

$$\frac{dL}{dw} = \vec{w} - \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i} = 0 \quad (6)$$
$$\Leftrightarrow \vec{w} = \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i}$$
$$\frac{dL}{db} = \sum_{i=1}^{l} \alpha_i * y_i = 0 \quad (7)$$

Now we can substitute $\vec{w} = \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i}$ into the Lagrangian function L.

$$\begin{aligned}
L &= \frac{1}{2} \langle \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i}, \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j} \rangle \\
&- \sum_{i=1}^{l} \alpha_i * (y_i * (\langle \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j}, \vec{x_i} \rangle + b) - 1) \\
&= \frac{1}{2} \langle \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i}, \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j} \rangle \\
&+ \sum_{i=1}^{l} \alpha_i - \sum_{i=1}^{l} \alpha_i * y_i * b \\
&- \sum_{i=1}^{l} \alpha_i * y_i * \langle \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j}, \vec{x_i} \rangle \\
&= \frac{1}{2} \langle \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i}, \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j} \rangle \\
&+ \sum_{i=1}^{l} \alpha_i - b * \sum_{i=1}^{l} \alpha_i * y_i \\
&- \langle \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j}, \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i} \rangle \\
&= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \langle \sum_{i=1}^{l} \alpha_i * y_i * \vec{x_i}, \sum_{j=1}^{l} \alpha_j * y_j * \vec{x_j} \rangle \\
&= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \vec{x_i}, \vec{x_j} \rangle
\end{aligned}$$

This leads to the following dual formulation:

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \vec{x_i}, \vec{x_j} \rangle \quad (8)$$
$$\text{constrained by } \sum_{i=1}^{l} \alpha_i y_i = 0, \alpha_i \geq 0 \; \forall i = 1...l$$

Notable about this formulation is that the solution of the problem is only dependent on the training data $(\vec{x_i}, y_i)$. After solving this problem $\vec{w}$ is given by $\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x_i}$, while b has to be calculated from the constraints of the quadratic program (4) [2]. Inserting the equation for $\vec{w}$ into the the linear classification function (1) gives the following classifier [2]:

$$\begin{aligned}
f(\vec{x}) &= \langle \sum_{i=1}^{l} \alpha_i y_i \vec{x_i}, \vec{x} \rangle + b \quad (9) \\
&= \sum_{i=1}^{l} y_i \alpha_i \langle \vec{x_i}, \vec{x} \rangle + b
\end{aligned}$$

If $\alpha_i = 0$ the corresponding feature vector and class label are ignored in the calculation of the linear classifier. All other feature vectors have an influence on the solution. These vectors are actually the support vectors, which were already mentioned in the geometric consideration [2].

V. Non-linear classification via kernels

As we could see in the previous section SVM solves the binary linear classification problem. But using the right mathematical tools it is possible to also classify non-linear problems using SVM. For example in the following figure the two classes of feature vectors can be separated by a quadratic function [1].

Figure 5. Non-linear classification problem.

The typical way to solve a non-linear problem with a linear method is to increase the dimension of the feature space by adding addtional attributes to the feature vectors. SVM can be used on this higher dimensional data creating a linear classifier in the extended feature space and therefore creating a non-linear classifier in the original feature space [1].
It is important to note, that in the dual formulation (8) and the classification function (1) the feature vectors are only appear in a inner product. Because of this the results of Mercer's theorem can be used to achieve the mapping into the higher-dimensional feature space. Mercer's theorem says that "any symmetric positive semi-definite function $K(\vec{x}, \vec{z})$ is an inner product in some space"[2]. This basically means we can replace the inner product $\langle \vec{x_i}, \vec{x_j} \rangle$ in the dual formulation (8) with a so called kernel function $K(\vec{x_i}, \vec{x_j})$. There is an ever increasing amount of usable kernel functions. Different kernels are developed and used for different purposes like sequence matching in bioinformatics[2]. The following two functions are just examples of the most basic kernels. They are both mentioned in [1] and [2].

	$K(\vec{v}, \vec{u})$
p-dimensional polynomial kernel	$(\langle \vec{u}, \vec{v} \rangle + 1)^p$
Gaussian kernel	$e^{-\frac{\|\vec{u}-\vec{v}\|^2}{2\sigma^2}}$

These kernels can be just used in place of the inner products of the dual formulation. The dual formula with kernel looks like this:

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\vec{x_i}, \vec{x_j}) \qquad (10)$$

$$\text{constrained by } \sum_{i=1}^{l} \alpha_i y_i = 0, \alpha_i \geq 0 \, \forall i = 1...l$$

Besides this minor change in the dual formula and the equivant change in the classifier function the algorithm stays the same. The "Kernel Trick" turns the linear SVM algorithm into a general non-linear method [1].

VI. Application of SVMs in real life

Until now we have only seen that SVM should in theory provide good results. Even though the theory provides some kind of proof for the quality of SVM, it is still advisable to take a look at the error percentage of SVM in real life applications.
The first example is an experiment of SVM in machine vision. In this experiment SVM was used for facial recognition. The original data used were 10 images per person of 40 people. Half of the data was used as training data 5 pictures per person, while the other half was used for testing and computing the generalization error. A direct SVM classifier used on this data had a generalization error of 5.5%. Preprocessing the pictures improved the error to 3.7%. An improved invariant SVM classifier, which used 1400 instead of 200 pictures as training data, generated a even better result with an error of 1.5%. The additional amount of training data was gained by translated and zoomed examples. Another improvement was the use of an RBF (radial basis function) kernel, which is also called Gaussian kernel. The best result of other methods achieved a generalization error of 2.7%. Therefore this experiment shows that in machine vision the SVM method definitely has its uses as it is able to generate classifiers, which can compete with other approaches.
The second example compares SVM with RBF network, which is a machine learning method based on neural networks, in handwritten digit recognition. Both methods were applied on the United States Postal Service data set, which contains 9298 entries of handwritten digits. SVM achieved better results on all digits, using a training set of 7291 and a test set of 2007. A similar experiment was done on the NIST (National Institute of Standards and Technology) dataset, which contains 60000 training and 10000 test samples. This experiment proved a similar result, as it turned out that SVM performs better than all other methods on this data.
The third and last example is in the classification of ovary cancer in gene expression data. Such data has a high dimensionality and therefore SVM is a good method as it its performance is not depended on the dimension of the feature space [1]. SVM achieved perfect results in the classification of ovarian cancer. Other tasks in the bioinformatics such as classification of colon tumors etc. showed similar results.
The data of all above examples is obtained from [1].
All this experiments showed further proof to the statement that SVM generates "good" classifier. But it is important to recognize that in most of these experiment not the raw SVM method was used, but rather specialized variants as well as task-specific kernels..

VII. Conclusion

The concept behind SVM holds multiple advantages in comparison to other methods like artificial neural networks.

For one the idea of pushing two parallel planes against opposing support vectors to create a maximum margin hyperplane as linear classifier is geometrically intuitive. But at the same time it is both with mathematics and experiments proven that the resulting classifier provides good results. Another advantage with SVM is that it does not have any problems with multiple extremes [1], which means for one set of training data there is only best solution. Because the formulation of SVM is only dependent on inner products, as we have seen, kernels can be used, which is another benefit as it allows non-linear classification and adaption of the classifier to certain applications. In this paper we only observed how SVM can solve the classification problem, in reality SVM can with the proper extension also help in novelty detection, regression analysis and others. For further reading in this topic [1] provides a starting point.

REFERENCES

[1] Kristin P. Bennett and Colin Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13, December 2000.

[2] Dmitriy Fradkin and Ilya Muchnik. Support vector machines for classification. *Discrete methods in epidemiology*, 70:13–20, 2006.