# Support Vector Machines — Kernels and the Kernel Trick

An elaboration for the Hauptseminar "Reading Club: Support Vector Machines"

Martin Hofmann

martin.hofmann@stud.uni-bamberg.de

June 26, 2006

# Contents

# 1 Introduction

Pioneered by Vapnik ([Vap95], [Vap98]), Support Vector Machines provide, beside multilayer perceptrons and radial-basis function networks, another approach to machine learning settings as for example pattern classification, object recognition, text classification or regression estimation ([Hay98], [Bur98]). Although this subject can be said to have already started in the late seventies [Vap79], it is only now receiving increasing attention due to sustained success research achieved in this subject.

Ongoing research reveal continuously how Support Vector Machines are able to outperform established machine learning techniques as neural networks, decision trees or k-Nearest Neighbour [Joa98] since they construct models that are complex enough to deal with real-world applications while remaining simple enough to be analysed mathematically [Hea98]. They combine the advantages of linear and non-linear classifiers as time efficient training (polynomial with sample size), high capacity, the prevention of overfitting in high dimensional instance spaces and the application to symbolic data, while simultaneously overcome their disadvantages.

Support Vector Machines belong to the class of Kernel Methods and are rooted in the statistical learning theory. As all kernel-based learning algorithms they are composed of a general purpose learning machine (in the case of SVM a linear machine) and a problem specific kernel function. Since the linear machine can only classify the data in a linear separable feature space, the role of the kernel-function is to induce such a feature space by implicitly mapping the training data into a higher dimensional space where the data is linear separable. Since both, the general purpose learning machine and the kernel function can be used in a modular way, it is possible to construct different learning machines characterized by different nonlinear decision surfaces.

The remainder of this report is organized in two main parts. In Section 2 the general operation of SVMs is described on a selected linear machine and in Section 3 the purpose of the kernel function is described as well as different kernels are introduced and kernel properties are discussed. The report concludes with some final remarks in Section 4.

# 2 Support Vector Machines

As mentioned before, the classifier of a Support Vector Machine can be used in a modular manner (as the kernel function) and therefore, depending on the purpose, domain, and the separability of the feature space different learners are used. There is for example the Maximum Margin Classifier for a linear separable data, the Soft Margin Classifier which allows some noise in the training data or Linear Programming Support Vector Machines for classification purposes, but also different models exist for applying the Support Vector method to regression problems [CST00].

The aim of a Support Vector Machine is to devise a computationally efficient way of learning good separating hyperplanes in a high dimensional feature space. In the following the construction of such a hyperplane is described using the Maximum Margin Classifier as an example of a linear machine. Note that for the sake of simplicity, a linear separable training set is assumed and solely the classifier is explained as the Kernel function is not yet used and explained later.

## 2.1 Optimal Hyperplane for Linearly Separable Patterns

Let $\mathcal{T} = \{(\vec{x_i}, y_i)\};\ i = 1, \ldots, k;\ \vec{x_i} \in \mathbb{R}^n;\ y_i \in \{-1, +1\}$ a linear separable training set. Then there exists a hyperplane of the form

$$\vec{w}^T \vec{x} + b = 0, \tag{1}$$

separating the positive from the negative training examples such that

$$\begin{aligned} \vec{w}^T \vec{x_i} + b \geq 0 \qquad & for\ y_i = +1, \\ \vec{w}^T \vec{x_i} + b < 0 \qquad & for\ y_i = -1, \end{aligned} \tag{2}$$

where $\vec{w}$ is the normal to the hyperplane and $b$ is the perpendicular distance of the hyperplane to the origin. A decision function

$$g(\vec{x}) = \vec{w}^T \vec{x_i} + b \tag{3}$$

therefore can be interpreted as the functional distance of an instance from the hyperplane. For $g(\vec{x}) < 0$ the instance would be classified negative as it lies below the decision surface and it would be classified positive if $g(\vec{x}) \geq 0$ as it lies on or above the surface.

Note that, as long as the constraints from Eq.(3) hold our decision function can be represented in different ways by simply rescaling $\vec{w}$ and $b$. Although all such decision functions would classify instances equally, the functional distance of an instance would change depending on $\vec{w}$ and $b$. To obtain a distance measure independent from $\vec{w}$ and $b$, the so called geometric distance, we simply normalise $\vec{w}$ and $b$ in Eq.(3) such that $\vec{w}_n = \frac{\vec{w}}{\|\vec{w}\|}$ be the unit vector, $b_n = \frac{|b|}{\|\vec{w}\|}$ the normalised perpendicular distance from the hyperplane to the origin and $\|\vec{w}\|$ the Euclidean norm of $\vec{w}$. Note that in the following both, $\vec{w}$ and $b$ are assumed to be normalised and are therefore not labelled explicitly any more.

Nevertheless, as Figure 1 illustrates, there still exists more than one separating hyperplane. It also follows from the fact that for a given training set $\mathcal{T}$ Eq.(1) has more than one solution.

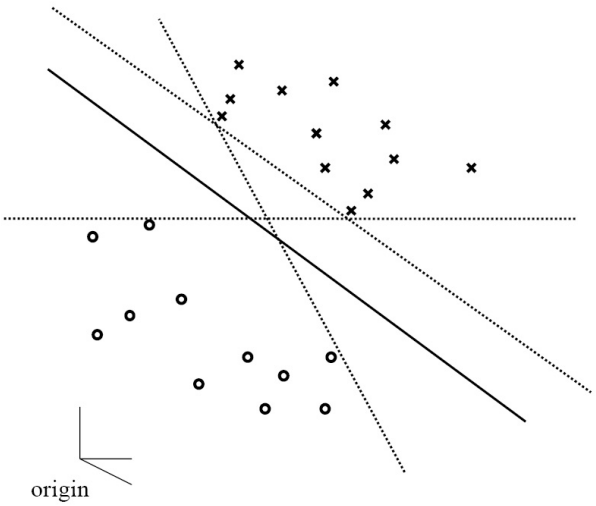


Figure 1: Suboptimal (dashed) and optimal (bold) separating hyperplanes

To solve this, let $d_+$ ($d_-$) be the shortest distance from the separating hyperplane to a positive (negative) training example and be the "margin" of a hyperplane $d_+ + d_-$.

The maximum margin algorithm simply looks for the hyperplane with the largest separating margin. This can be formulated by the following constraints for all $\vec{x}_i \in \mathcal{T}$:

$$\vec{w}^T\vec{x_i} + b \geq +1 \qquad for\ y_i = +1 \tag{4}$$
$$\vec{w}^T\vec{x_i} + b \leq -1 \qquad for\ y_i = -1 \tag{5}$$

Both constraints can be combined into one set of inequalities:

$$y_i(\vec{w}^T\vec{x_i} + b) - 1 \geq 0\ \forall i \tag{6}$$

Thus, we say the distance of every data point from the hyperplane to be greater than a certain value and this value to be +1 in terms of the unit vector.

Now consider all data points $\vec{x_i} \in \mathcal{T}$ for which the equality in Eq.(4) holds. This is equivalent choosing a scale for $\vec{w}$ and $b$ such that this equality holds. Then all these points lie on a hyperplane $H_1 :\ \vec{w}^T\vec{x_i} + b = +1$ with normal $\vec{w}$ and perpendicular distance from the origin $\frac{|1-b|}{\|\vec{w}\|}$. Similarly, all points for which the equality condition in Eq.(5) holds lie on a hyperplane $H_2 :\ \vec{w}^T\vec{x_i} + b = -1$ with normal $\vec{w}$ and perpendicular distance from the origin $\frac{|-1-b|}{\|\vec{w}\|}$. Hence, $d_+ = d_- = \|\vec{w}\|$ implying a margin of $\frac{2}{\|\vec{w}\|}$. Note that $H_1$ and $H_2$ have the same normal and are consequently parallel and due to constraint Eq.(6) no training point lies between them. Figure 2 visualises these findings. Those data points for which the equality condition in Eq.(6) hold would change the solution if removed and are called the support vectors; in Figure 2 they are indicated by extra circels.

Maximising our margin of $\frac{2}{\|\vec{w}\|}$ subject to constraints of (6) would yield the solution for our optimal separating hyperplane and would provide the maximum possible separation between positive and negative training examples.

## 2.2 Quadratic Optimization to Find the Optimal Hyperplane

To solve the maximisation problem derived in the last section we transform it into a minimisation problem of the following quadratic cost function:

$$\Phi(\vec{w}) = \frac{1}{2}\vec{w}^T\vec{w}. \tag{7}$$

Instead of maximising the margin, we minimise the Euclidean norm of the weight vector $\vec{w}$. The reformulation into a quadratic cost function does not

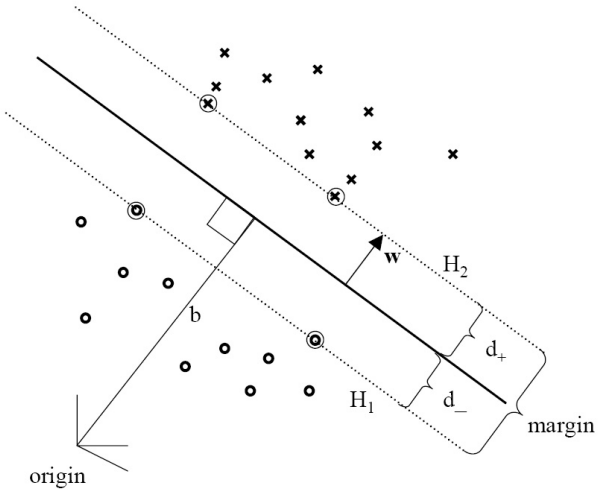


Figure 2: Optimal separating hyperplane with maximum margin

change our optimisation problem but assures that all training data only occur in form of a dot product between vectors. In Section 3 we will take advantage from this crucial property. Since our cost function is quadratic and convex, and the constraints from Eq.(6) are linear this optimisation problem can be dealt by introducing $l$ Lagrange multipliers $\alpha_i \geq 0;\ i = 1, \ldots, l$, one for each inequality constraint (6). The Lagrangian is formed by multiplying the constraints by the positive Lagrange multipliers and subtract them from the cost function. This gives the following Lagrangian:

$$L_P(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\vec{w}^T\vec{w} - \sum_{i=1}^{l} \alpha_i \left[y_i(\vec{w}^T\vec{x}_i + b) - 1\right] \tag{8}$$

The Langragian $L$ has to be minimised with respect to the primal variable $\vec{w}$ and $b$ and maximised with respect to the dual variable $\vec{\alpha}$, i.e. a saddle point has to be found. The Duality Theorem, as formulated in [Hay98], states that in such a constraint optimisation problem (a convex objective function and a linear set of constraints) if the primal problem (minimise with respect to $\vec{w}$ and $b$) has an optimal solution, the dual problem (maximise with respect to $\vec{\alpha}$) has also an optimal solution, and the corresponding optimal values are equal. Note, that from now on we use $L_P$ for the primal Lagrangian problem and $L_D$ for the dual Lagrangian.

Perhaps more intuitively, one can also describe it in the following way. If a constraint (6) is violated ($y_i(\vec{w}^T\vec{x}_i + b) - 1 < 0$) $L$ can be increased by increasing the corresponding $\alpha_i$, but then $\vec{w}$ and $b$ have to change such that

$L$ decreases. To prevent $-\alpha_i(y_i((\vec{w}^T x_i) + b) - 1)$ from becoming arbitrarily large the change in $\vec{w}$ and $b$ will ensure that the constraint will eventually be satisfied. This is the case, when a data point would fall into the margin and then $\vec{w}$ and $b$ have to be changed to adjust the margin again. For all constraints which are not precisely met as equalities, i.e. for which $y_i(\vec{w}^T\vec{x_i} + b) - 1 > 0$ (the data point is more than one unit away from the optimal hyperplane), the corresponding $\alpha_i$ must be 0 to maximize $L$ [Sch00].

The solution for our primal problem we get by differentiating $L_P$ with respect to $\vec{w}$ and $b$. Setting the results equal to zero yields the following two optimum conditions, i.e. minimum of $L_P$ with respect to $\vec{w}$ and $b$:

$$\begin{aligned} &\text{Condition 1:} && \frac{\partial L(\vec{w}, b, \vec{\alpha})}{\delta \vec{w}} = \vec{0}, \\ &\text{Condition 2:} && \frac{\partial L(\vec{w}, b, \vec{\alpha})}{\delta b} = 0. \end{aligned}$$

Application of the optimality condition 1 to the Lagrangian function Eq.(8) and after rearrangement of terms yields:

$$\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i. \tag{9}$$

Application of the optimality condition 2 to the Lagrangian function Eq.(8) and after rearrangement of terms yields:

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{10}$$

Expanding the $L_P$ we get:

$$\begin{aligned} L(\vec{w}, b, \vec{\alpha})_P &= \frac{1}{2}\vec{w}^T\vec{w} - \sum_{i=1}^{l} \alpha_i \left[ y_i(\vec{w}^T\vec{x}_i + b) - 1 \right] \\ &= \frac{1}{2}\vec{w}^T\vec{w} - \sum_{i=1}^{l} \alpha_i y_i \vec{w}^T\vec{x}_i - b\sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \end{aligned} \tag{11}$$

The third term on the right-hand side is zero due to the optimality condition of Eq.(10). Rearranging Eq.(7) yields:

$$\frac{1}{2}\vec{w}^T\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{w}^T\vec{x}_i = \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x}_i^T\vec{x}_j \tag{12}$$

Finally after substitution into Eq.(11) and after rearrangement of terms we get the formalisation of our dual problem:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j \tag{13}$$

Given a training set $\mathcal{T}$, $L_D$ now has to be maximised subject to the constraints:

$$\begin{aligned} &(1) \quad \sum_{i=1}^{l} \alpha_i y_i = 0, \\ &(2) \quad \alpha_i \geq 0 \qquad \text{for } i = 1, \ldots, l, \end{aligned}$$

by finding the optimal Lagrange multipliers $\{\alpha_{i,o}\}_{i=1}^{l}$

In this case, support vector training comprises to find those Lagrange multipliers $\alpha_i$ that maximise $L_D$ in Eq.(13). Simple mathematical measurements are not applicable for this problem, since it requires numerical methods of quadratic optimisation. From now on, the optimal $\alpha_{i,o}$ are assumed to be given and from an explicit derivation is abstained.

Note, that there exists a Lagrange multiplier $\alpha_{i,o}$ for every training point $x_i$. In the solution, the training points for which $\alpha_{i,o} > 0$ are called "support vectors" and lie on the hyperplane $H_1$ or $H_2$. All other data points have $\alpha_{i,o} = 0$ and lie on that side of $H_1$ or $H_2$ such that the strict inequality of Eq.(6) holds. Using the optimum Lagrange multipliers $\alpha_{i,o}$ we may compute the optimal weight vector $\vec{w}_o$ using Eq.(9) and so write:

$$\vec{w}_o = \sum_{i=1}^{l} \alpha_{i,o} y_i \vec{x}_i \tag{14}$$

Now we may formulate our optimal separating hyperplane:

$$\vec{w}_o^T \vec{x} + b_o = \left( \sum_{i=1}^{l} \alpha_{i,o} y_i \vec{x}_i \right)^T \vec{x} + b_o = \sum_{i=1}^{l} \alpha_{i,o} y_i \vec{x}_i^T \vec{x} + b_o = 0 \tag{15}$$

Similarly, the decision function $g(\vec{x})$:

$$g(\vec{x}) = sgn(\vec{w}_o^T \vec{x} + b_o) = sgn \left( \sum_{i=1}^{l} \alpha_{i,o} y_i \vec{x}_i^T \vec{x} + b_o \right) \tag{16}$$

To get the optimal perpendicular distance from the optimal hyperplane to the origin, consider a positive support vector $\vec{x}^{(s)}$. Using the left-hand side of Eq.(15), following equation must hold:

$$\vec{w}_o^T \vec{x}^{(s)} + b_o = +1 \tag{17}$$

This is not surprising since $\vec{x}^{(s)}$ lies on $H_2$. After trivial rearrangement we get:

$$b_o = 1 - \vec{w}_o^T \vec{x}^{(s)} \qquad \text{for } y^{(s)} = +1 \tag{18}$$

# 3 Kernels and the Kernel Trick

Remember, that so far we assumed a linear separable set of training data. Nevertheless, this is only the case in very few real-world applications. Now the kernel function comes to handy as a remedy, as an implicit mapping of the input space into a linear separable feature space, where our linear classifiers are again applicable.

In section 3.1 the mapping from the input space into the feature space is explained as well as the "Kernel Trick", while in Section 3.2 we will concentrate more on different kernels and the properties they must satisfy and finally Section 3.3 focuses on Mercer's Theorem.

## 3.1 Feature Space Mapping

Let us start with an example. Consider a non-linear mapping function $\Phi : I = \mathbb{R}^2 \to F = \mathbb{R}^3$ from the 2-dimensional input space $I$ into the 3-dimensional feature space $F$, which is defined in the following way:

$$\Phi(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T. \tag{19}$$

Taking the equation for a separating hyperplane Eq.(1) into account we get a linear function in $\mathbb{R}^3$:

$$\vec{w}^T \Phi(\vec{x}) = w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 = 0. \tag{20}$$

It is worth mentioning, that Eq.(20) is an elliptic function when set to a constant $c$ and evaluated in $\mathbb{R}^2$. Hence, with an appropriate mapping function we can use our linear classifier in $F$ on a transformed version of the data to get a non-linear classifier in $I$ with no effort. After mapping our non-linear separable data into a higher dimensional space we can find a linear separating

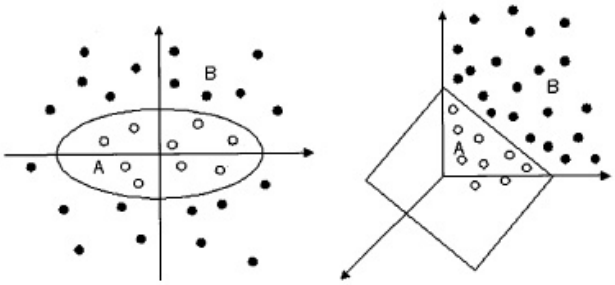


Figure 3: Mapping of non-linear separable training data from $\mathbb{R}^2$ into $\mathbb{R}^3$

hyperplane. For an intuitive understanding, consider Figure 3.

Thus, by simply applying our linear maximum margin classifier to a mapped data set, we can reformulate our dual Lagrangian of our optimisation problem of Eq.(13)

$$L_D(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \Phi(\vec{x}_i)^T \Phi(\vec{x}_j), \tag{21}$$

the optimal weight vector Eq.(14)

$$\vec{w}_o = \sum_{i=1}^{l} \alpha_{i,o} y_i \Phi(\vec{x}_i), \tag{22}$$

the optimal hyperplane Eq.(15)

$$\vec{w}_o^T \vec{x} + b_o = \sum_{i=1}^{l} \alpha_{i,o} y_i \Phi(\vec{x}_i)^T \Phi(\vec{x}) + b_o = 0; \tag{23}$$

and the optimal decision function Eq.(16)

$$g(\vec{x}) = sgn(\vec{w}_o^T \vec{x} + b_o) = sgn\left( \sum_{i=1}^{l} \alpha_{i,o} y_i \Phi(\vec{x}_i)^T \Phi(\vec{x}) + b_o \right). \tag{24}$$

From Eq.(22) follows, that our weight vector of the optimal hyperplane in $F$ can be represented only by data points. Note also, that both, Eq.(23) and Eq.(24), only depend on the mapped data through dot products in some feature space $F$. The explicit coordinates in $F$ and even the mapping function $\Phi$ become unnecessary when we define a function $K(\vec{x}_i, \vec{x}) = \Phi(\vec{x}_i)^T \Phi(\vec{x})$, the

so called kernel function, which directly calculates the value of the dot product of the mapped data points in some feature space. The following example of a kernel function $K$ demonstrates the calculation of the dot product in the feature space using $K(\vec{x},\vec{z}) = \left(\vec{x}^T\vec{z}\right)^2$ and inducing the mapping function $\Phi(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$ of Eq.(19):

$$\begin{aligned}
\vec{x} &= (x_1, x_2) \\
\vec{z} &= (z_1, z_2) \\
K(\vec{x},\vec{z}) &= \left(\vec{x}^T\vec{z}\right)^2 \\
&= (x_1z_1 + x_2z_2)^2 \\
&= (x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2) \\
&= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T(z_1^2, \sqrt{2}z_1z_2, z_2^2) \\
&= \phi(\vec{x})^T\phi(\vec{z})
\end{aligned}$$

The advantage of such a kernel function is that the complexity of the optimisation problem remains only dependent on the dimensionality of the input space and not of the feature space. Therefore, it is possible to operate in a theoretical feature space of infinite height.

We can solve our dual Lagrangian of our optimisation problem in Eq.(21) using the kernel function $K$:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{l}\alpha_i - \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(\vec{x}_i,\vec{x}_j) \tag{25}$$

With the dual representation of the optimal weight vector Eq.(22) of the decision surface in the feature space $F$, we can finally also reformulate the equation of our optimal separating hyperplane:

$$\vec{w}_o^T\vec{x} + b_o = \sum_{i=1}^{l}\alpha_{i,o}y_iK(\vec{x}_i,\vec{x}) + b_o = 0, \tag{26}$$

where $\alpha_{i,o}$ are the optimal Lagrange multipliers obtained from maximising Eq. (25) and $b_o$ the optimal perpendicular distance from the origin, calculated according to Eq.(18), but now with $\vec{w}_o$ and $\vec{x}^{(s)}$in $F$.

## 3.2 Kernels and their Properties

We have discussed so far the functionality of kernel functions and their use with support vector machines. Now the question arises how to get an appropriate kernel function. A kernel function can be interpreted as a kind of

similarity measure between the input objects. In practise a couple of kernels (Table 1) turned out to be appropriate for most of the common settings.

| Type of Kernel | Inner product kernel $K(\vec{x}, \vec{x}_i),\ i = 1, 2, \ldots, N$ | Comments |
|---|---|---|
| Polynomial Kernel | $K(\vec{x}, \vec{x}_i) = \left(\vec{x}^T \vec{x}_i + \theta\right)^d$ | Power $p$ and threshold $\theta$ is specified a priori by the user |
| Gaussian Kernel | $K(\vec{x}, \vec{x}_i) = e^{-\frac{1}{2\sigma^2}\|\vec{x}-\vec{x}_i\|^2}$ | Width $\sigma^2$ is specified a priori by the user |
| Sigmoid Kernel | $K(\vec{x}, \vec{x}_i) = tanh(\eta\, \vec{x}\, \vec{x}_i + \theta)$ | Mercer's Theorem is satisfied only for some values of $\eta$ and $\theta$ |
| Kernels for Sets | $K(\chi, \chi') = \sum_{i=1}^{N_\chi} \sum_{j=1}^{N_{\chi'}} k(x_i, x'_j)$ | Where $k(x_i, x'_j)$ is a kernel on elements in the sets $\chi$, $\chi'$ |
| Spectrum Kernel for strings | count number of substrings in common | It is a kernel, since it is a dot product between vectors of indicators of all the substrings. |

Table 1: Summary of Inner-Product Kernels [Hay98]

Although some kernels are domain specific there is in general no best choice. Since each kernel has some degree of variability in practise there is nothing else for it but to experiment with different kernels and adjust their parameters via model search to minimize the error on a test set. Generally, a low polynomial kernel or a Gaussian kernel have shown to be a good initial try and to outperform conventional classifiers ([Joa98], [FU95]).

As already mentioned, a kernel function is a kind of similarity metric between the input objects, and therefore it should be intuitively possible to combine somehow different similarity measures to create new kernels. Following closure properties are defined over kernels, assuming that $K_1$ and $K_2$ are kernels over $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(\cdot)$ a real valued function, $\Phi : X \to \mathbb{R}^m$ with $K_3$ a kernel over $\mathbb{R}^m \times \mathbb{R}^m$, and $\boldsymbol{B}$ a symmetric semi-definite $n \times n$ matrix [CST00]:

$$
\begin{aligned}
1.\ K(\vec{x},\vec{z}) &= c \cdot K_1(\vec{x},\vec{z}), && (27)\\
2.\ K(\vec{x},\vec{z}) &= c + K_1(\vec{x},\vec{z}), && (28)\\
3.\ K(\vec{x},\vec{z}) &= K_1(\vec{x},\vec{z}) + K_2(\vec{x},\vec{z}), && (29)\\
4.\ K(\vec{x},\vec{z}) &= K_1(\vec{x},\vec{z}) \cdot K_2(\vec{x},\vec{z}), && (30)\\
5.\ K(\vec{x},\vec{z}) &= f(\vec{x}) \cdot f(\vec{z}), && (31)\\
6.\ K(\vec{x},\vec{z}) &= K_3(\Phi(\vec{x}),\Phi(\vec{z})), && (32)\\
7.\ K(\vec{x},\vec{z}) &= \vec{x}^T \boldsymbol{B} \vec{z}. && (33)
\end{aligned}
$$

## 3.3 Mercer's Theorem

Up to this point, we only looked on predefined general purpose kernels, but in real world applications it is rather more interesting what properties a similarity function over the input objects has to satisfy to be a kernel function. Clearly, the function must be symmetric,

$$
K(\vec{x},\vec{z}) = \phi(\vec{x})^T\phi(\vec{z}) = \phi(\vec{z})^T\phi(\vec{x}) = K(\vec{z},\vec{x}), \tag{34}
$$

and satisfy the inequalities that follow from the Cauchy-Schwarz inequality,

$$
\begin{aligned}
\left(\phi(\vec{x})^T\phi(\vec{z})\right)^2 &\leq \|\phi(\vec{x})\|^2\|\phi(\ vecz)\|^2\\
&= \phi(\vec{x})^T\phi(\vec{x})\phi(\vec{z})^T\phi(\vec{z})\\
&= K(\vec{x},\vec{x})K(\vec{z},\vec{z}).
\end{aligned} \tag{35}
$$

Furthermore, Mercer's theorem provides a necessary and sufficient characterisation of a function as a kernel function. A kernel as a similarity measure can be represented as a similarity matrix between its input objects as follows:

$$
\mathbf{K} = \begin{pmatrix}
\phi(\vec{v}_1)^T\phi(\vec{v}_1) & \dots & \phi(\vec{v}_1)^T\phi(\vec{v}_n)\\
\vdots & & \\
\phi(\vec{v}_2)^T\phi(\vec{v}_1) & \ddots & \vdots\\
\vdots & & \\
\phi(\vec{v}_n)^T\phi(\vec{v}_1) & \dots & \phi(\vec{v}_n)^T\phi(\vec{v}_n)
\end{pmatrix}, \tag{36}
$$

where $V = \{\vec{v}_1,\dots,\vec{v}_n\}$ is a set of input vectors and $\mathbf{K}$ a matrix, the so called Gram Matrix, containing the inner products between the input vectors. Since $\mathbf{K}$ is symmetric there exists an orthogonal matrix $\mathbf{V}$ such that $\mathbf{K} = \mathbf{V\Lambda V^T}$, where $\Lambda$ is a diagonal matrix containing eigenvalues $\lambda_t$ of $\mathbf{K}$, with

corresponding eigenvectors $\vec{v}_t = (v_{ti})_{i=1}^n$ as the columns of $\mathbf{V}$. Assuming all eigenvalues to be non-negative and assuming that there is a feature mapping

$$\phi : \vec{x}_i \mapsto \left(\sqrt{\lambda_i} v_{ti}\right)_{t=1}^n \in \mathbb{R}^n, i = 1, \ldots, n, \tag{37}$$

then

$$\phi(\vec{x}_i)^T \phi(\vec{x}_j) = \sum_{t=1}^{n} \lambda_t v_{ti} v_{tj} = (\mathbf{V}\Lambda\mathbf{V}')_{ij} = \mathbf{K}_{ij} = K(\vec{x}_i, \vec{x}_j), \tag{38}$$

implying that $K(\vec{x}_i, \vec{x}_j)$ is indeed a kernel function corresponding to the feature mapping $\Phi$. Consequently, it follows from Mercer's theorem, that a matrix is a Gram Matrix, if and only if it is positive and semi-definite, i.e. it is an inner product matrix in some space [CST00]. Hence, a Gram Matrix fuses all information necessary for the learning algorithm, the data points and the mapping function merged into the inner product.

Nevertheless, it is noteworthy, that Mercer's theorem only tells us when a candidate kernel is an inner product kernel, and therefore admissible for use in support vector machines. However it tells nothing about how good such a function is. Consider for example a diagonal matrix, which of course satisfies Mercer's conditions but is not quite good as a Gram Matrix since it represents orthogonal input data and therefore self-similarity dominates between-sample similarity.

# 4 Conclusion

This paper gave an introduction to Support Vector Machines as a machine learning method for classification on the example of a maximum margin classifier. Furthermore, it discussed the importance of the kernel function and introduced general purpose kernels and the necessary properties for inner product kernels.

Support vector machines are able to apply simple linear classifiers on data mapped into a feature space without explicitly carrying out such a mapping and provide a method to compute a non-linear classification function without big effort since the complexity always remains only dependent on the dimension of the input space.

Although using the general purpose kernels with model search and cross validation already achieve sufficient results they don't take peculiarities of the training data into account. Kernel principal components analysis uses the eigenvectors and eigenvalues of the data to draw conclusions from the directions of maximum variance to construct inner product kernels, i.e. inner product of the mapped data points (see Eq.(38)), tailored to the data.

# References


[Bur98] Chris Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CST00] Nello Cristianini and John Shawe-Taylor. *An introduction to Support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[FU95] U. M. Fayyad and R. Uthurusamy, editors. *Extracting support data for a given task*. AAAI Press, 1995.

[Hay98] Simon Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 1998.

[Hea98] Marti A. Hearst. Trends controversies: Support vector machines. *IEEE Intelligent System*, 13(4):18–28, 1998.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Sch00] Bernhard Schölkopf. Statistical learning and kernel methods. In *Proceedings of the Interdisciplinary College 2000, Günne, Germany*, March, 2000.

[Vap79] Vladimir N. Vapnik. *Estimation of Dependencies Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English Translation: Springer-Verlag, New York, 1982).

[Vap95] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. 1998.