



TECHNICAL UNIVERSITY OF MUNICH

—

Advanced Seminar in Data Science

Mathematics in Data Science

Learning with kernels and SVMs

Author Théomé Borck (MSc. Mathematics in Data Science)

Supervisors Prof. Dr. Massimo Fornasier (Department of Mathematics)
 Prof. Dr. Peter Massopust (Department of Mathematics)

Feb 2021

Abstract

Support Vector Machines and kernels have strong applications in classification and regression, and work well on non-linearly separable data. In this summary paper, we approach both of these concepts through the intuition behind them by providing a strong focus on their interpretation and their application in practice. Despite a meaningful understanding at first sight, both of the concepts can be more rigorously derived from the theory and we will link the interpretation to a more mathematical approach thanks to the *Statistical Learning Theory*.

This summary paper was produced for the purpose of an advanced seminar on *Mathematics in Data Science* at *TUM*. It is based on two resources: the **book *Learning with kernels, Support Vector Machines, Regularization, Optimization and Beyond*** [1] written in 2002 by Bernhard Schölkopf and Alex Smola [1] and a **theoretical paper *The Mathematics of Learning: Dealing with data*** [2] written by Tomaso Poggio and Steve Smale in 2003. In order to fully understand the concepts behind SVMs and kernels, two additional resources have been used: the *C19 Machine Learning Lecture 3: SVM dual, kernels and regression* [3] by Andrew Zissermann from *Oxford University*, Oxford and the *IN2064 Machine Learning Lecture 8: SVM and kernels* [4] by Stephan Günnemann from *TUM*, Munich.

Contents

| | |
|---|-----------|
| References | 1 |
| Abstract | 1 |
| 1 Introduction | 3 |
| 1.1 Binary classification | 3 |
| 1.2 The perceptron algorithm | 3 |
| 2 Support Vector Machines (SVM) | 4 |
| 2.1 The intuition behind the model | 4 |
| 2.2 An optimization problem | 4 |
| 2.3 Soft-SVM and slack variables | 4 |
| 2.4 Gradient descent to solve the SVM problem | 5 |
| 2.5 From primal to dual formulation | 6 |
| 2.6 Support vectors | 7 |
| 3 Kernels | 7 |
| 3.1 Feature map projection | 7 |
| 3.2 The "kernel trick" | 8 |
| 3.3 Examples | 9 |
| 3.4 Kernels for non-numerical data | 10 |
| 4 SVM in practice | 10 |
| 4.1 Test time | 10 |
| 4.2 Multi-classification | 11 |
| 4.3 Applications | 11 |
| 4.4 Regression | 11 |
| 5 A theoretical approach with the mathematics of learning | 12 |
| 5.1 A key algorithm to learning theory | 12 |
| 5.2 Theory behind the bias/variance trade-off | 13 |
| 5.3 Understand SVM and its derivation | 14 |
| 6 Conclusion | 15 |
| Appendices | 17 |
| A Lagrangian multipliers derivation | 17 |
| B RBF kernel and infinite-dimensional feature projection | 17 |
| C Mathematical derivation of the sample and approximation errors | 17 |

1 Introduction

1.1 Binary classification

As stated by Poggio and Smale in *The mathematics of learning: Dealing with Data* [2], one important part of machine learning, and more generally the learning theory, is supervised learning. The latter consists in training systems, instead of programming them, with a set of labeled data examples. Specifically, training means here synthesizing a function - the model - which best represents the relation between inputs and outputs. In parallel, learning stands for a principle method that distills predictive, and hence scientific, theories from the data.

In that respect, the following summary paper is focused on binary classification, which is a supervised learning problem where we learn a model that discriminates between two classes of data. As illustrated in Figure 1, data can either be linearly-separable or not. In the second case, this can be due to noisy data or an underlying non-linear data distribution.

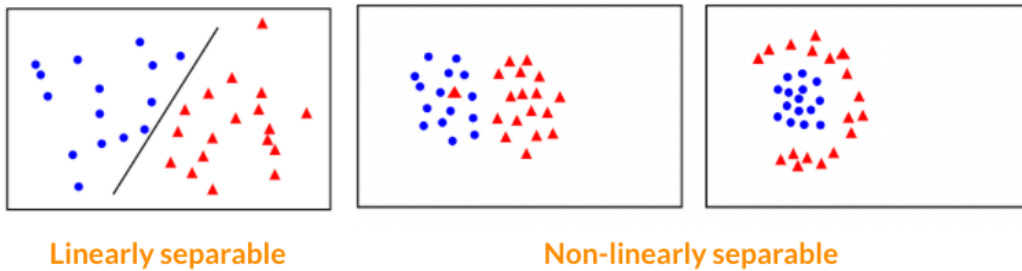


Figure 1: Examples of data distribution in a 2-dimensional space

Here, the following simple problem is considered: we want to find a linear classifier, i.e. a function $f : R^d \rightarrow R$, such that given the labeled data $(\mathbf{x}_i, y_i), i \in [N], y_i \in \{-1, 1\}$, we have $y_i f(\mathbf{x}_i) > 0$ with $f(\mathbf{x}_i) = \mathbf{w}^t \mathbf{x}_i + b$, \mathbf{w}, b being the parameters of our model, d the dimension of the input feature space and N the number of instances. It is worth noting that, compared to an algorithm like *k-nearest neighbors* (*k-NN*) where we need to carry the training data at test time, here we are using the data to learn specific parameters and then the data is discarded.

1.2 The perceptron algorithm

One approach to binary classification is the *Perceptron algorithm*. The latter is an iteration algorithm invented in 1958 at the *Cornell Aeronautical Laboratory* by Frank Rosenblatt. It converges in the linearly separable case and finds a separating hyperplane. However, this algorithm converges slowly, only works with linearly-separable data and lacks from a margin interpretation, so much so that the separating hyperplane can be too close to the training data and hence the model may be non-robust to new data. This is why SVM has been introduced to solve binary classification.

2 Support Vector Machines (SVM)

2.1 The intuition behind the model

The objective of *Support Vector Machines* is to find a hyperplane which separates both classes with the maximum margin. From Figure 2, one can observe that SVM will use specific vectors, called *support vectors*, to build a classifier with a large margin. Intuitively, a wide margin around the dividing line makes it more likely that new samples will fall on the right side of the boundary. This yields a more stable model given perturbations of the training data and hence a better generalization of the data distribution. The actual rigorous motivation behind SVM comes from the statistical learning theory and was introduced by Vapnik in 1995. This theory will be approached in more details over the last part of this paper.

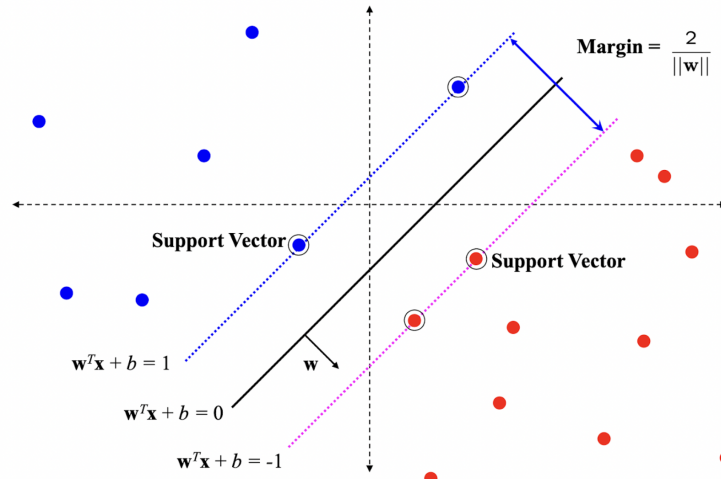


Figure 2: SVM: maximizing a margin [3]

2.2 An optimization problem

Mathematically speaking, maximizing the margin with respect to the classification constraints, we obtain for the SVM the following optimization problem:

$$\begin{aligned} & \text{Minimize } f_0(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \mathbf{w}^t \mathbf{w} \\ & \text{Subject to } f_i(\mathbf{w}, b) = y_i(\mathbf{w}^t \mathbf{x}_i + b) - 1 \geq 0 \text{ for } i = 1, \dots, N \end{aligned} \quad (1)$$

This is a convex and quadratic optimization problem, subject to linear constraints. Hence, a global optimal solution, i.e. a unique minimizer, can be obtained.

2.3 Soft-SVM and slack variables

However, in the real world, data is often noisy with outliers. To manage this problem and still be able to compute a separating hyperplane, we need to relax the hard constraints

and introduce a *trade-off* between the margin and the number of mistakes on the training data, i.e. the constraints that we violate. In that respect, we introduce the slack variables $\xi_i \geq 0$ for $i \in [N]$. Thanks to the latter, the idea is to relax the constraint $y_i(w^t x_i + B) \geq 1 - \xi_i$ for all i but punish as much as necessary the relaxation of this constraint. Practically speaking, for every training sample \mathbf{x}_i , ξ_i gives the distance of how far the margin is violated by this training sample, in units of $\|\mathbf{w}\|$. As you can see in Figure 3, $0 < \xi_i \leq 1$ corresponds to a margin violation for an instance between the margin and the correct side of the hyperplane, and $\xi_i > 1$ corresponds to a misclassified point.

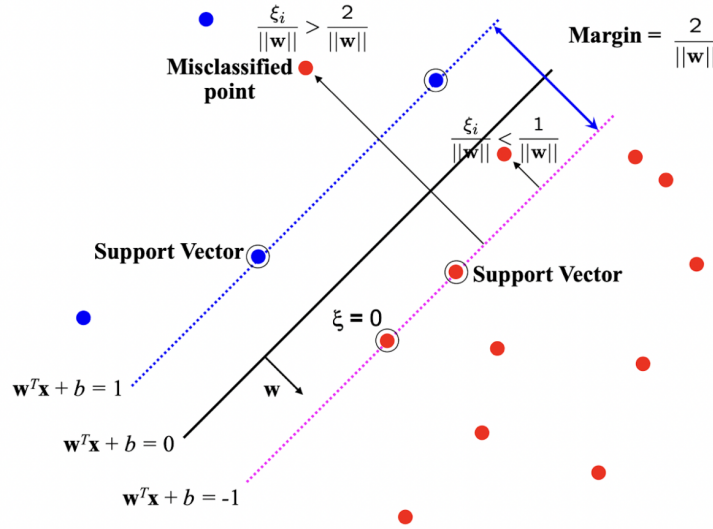


Figure 3: SVM with slack variables [3]

Hence, the optimization problem (1) becomes in this *soft-SVM* setting the following one:

$$\text{Minimize } f_0(\mathbf{w}, b) = \frac{1}{2} \mathbf{w}^t \mathbf{w} + C \sum_{i=1}^N \xi_i \quad (2)$$

$$\begin{aligned} \text{Subject to } f_i(\mathbf{w}, b) &= y_i(\mathbf{w}^t \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \text{ for } i = 1, \dots, N \\ \xi_i &\geq 0 \text{ for } i = 1, \dots, N \end{aligned}$$

$C > 0$ is here a regularization parameter and determines how heavy is the violation punished. A lower C will lead to a larger margin at the cost of misclassifying more examples, but this can build a stronger model with respect to unseen data. In contrary, $C \rightarrow \infty$ gives back the initial SVM with hard constraints.

2.4 Gradient descent to solve the SVM problem

We can easily rewrite the slack variables as the following:

$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^t \mathbf{x}_i) & \text{if } y_i(\mathbf{w}^t \mathbf{x}_i + b) < 1 \\ 0 & \text{else} \end{cases} \quad (3)$$

Hence, the problem (2) becomes the following unconstrained optimization problem:

$$\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^t \mathbf{w} + C \sum_{i=1}^N \max\{0, 1 - y_i(\mathbf{w}^t \mathbf{x}_i + b)\} \quad (4)$$

The second term of (4) is the hinge loss, a convex function which approximates the 0-1 loss function as you can see in Figure 4, whereas the first term corresponds to a regularization term. By sum of two convex functions, the objective function is also convex and the minimum can be found using standard gradient-based methods such as *stochastic gradient descent* (SGD). For the SVM model, SGD is also known as the *Pegasos (Primal Estimated sub-Gradient SOLver) algorithm*. However, this is not the usual way to solve the SVM problem. Instead, a more elegant method in mathematics through the dual formulation is often preferred, especially in order to leverage the help of kernels in non-linearly separable cases.

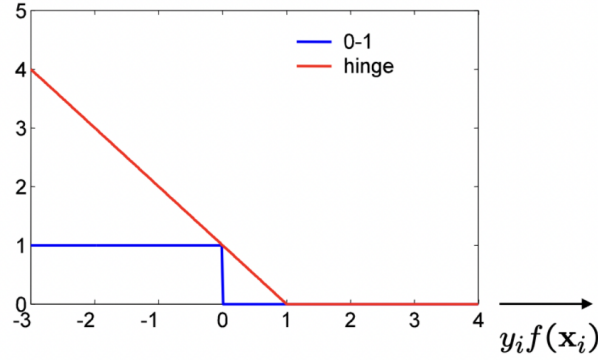


Figure 4: Hinge loss and 0-1 loss functions

2.5 From primal to dual formulation

Using the *Lagrangian multipliers* method (or via the *Representer theorem*), we can convert the primal problem (2) into its following dual formulation (5). The details of such a derivation can be found in Appendix A.

$$\begin{aligned} \text{Maximize } g(\boldsymbol{\alpha}) &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \mathbf{x}_i^t \mathbf{x}_j \\ \text{Subject to } \sum_{i=1}^N \alpha_i y_i &= 0 \text{ for } 0 \leq \alpha_i \leq C \end{aligned} \quad (5)$$

Contrary to the primal problem where we needed to estimate d parameters, d being the dimension of the input vector space, in the dual problem, we need to estimate N parameters. At first sight, the dual form appears to have the disadvantage of a k -NN classifier, where you need all the training data to classify a new instance. But, in practice, we will see that, once the SVM classifier is found, most of the α_i are null, the non-zero

α_i being associated to the support vectors. More generally, if $N \ll d$, such a dual formulation is more efficient. Above all, the greatest advantage of this formulation is that we only manipulate scalar products from the data instances. This will be a huge advantage to use kernels as we will see in the second part.

2.6 Support vectors

We can easily see in Figures 2 and 3 that, once the SVM optimization algorithm has converged, the learnt classifier $f(\mathbf{x}) = \sum_i \alpha_i y_i (\mathbf{x}_i^t \mathbf{x}) + b$ uses only a few vectors that are called *support vectors*. The latter are basically special data points chosen by the optimization. As they are few, classifying a new instance at test time is fast and this is an advantage of the SVM.

Overall, SVM consists in a linear binary classifier which maximizes the margin between two classes. It can be optimized through gradient descent or thanks to the use of standard *quadratic programming* libraries. We have seen that a soft version allows us to classify noisy data like in the first plot of Figure 5. However, how can we handle data for which the distribution is non-linearly separable in the input feature space, as illustrated in the second plot? In that purpose, we need to approach another concept: the kernels.

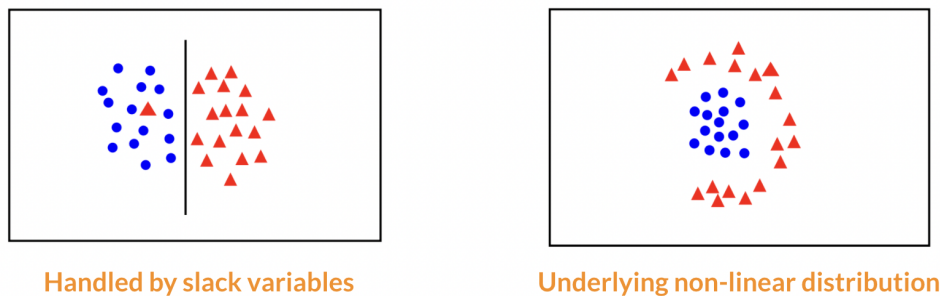


Figure 5: Non-linearly separable data in a 2-dimensional space

3 Kernels

3.1 Feature map projection

In order to use a linear classifier, the motivation behind kernels is to project the data from the input vector space into a higher-dimensional space. This is done through a feature map transformation $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$ which maps data from R^d into R^D with $D > d$. This new feature space is hence chosen to represent data in a linearly-separable way, such that a SVM can be applied and a classifier $f(x) = \mathbf{w}^t \phi(\mathbf{x}) + b$ computed. An illustration of such a feature transformation from a 2-dimensional space into a 3-dimensional space can be found in Figure 6. We could think here that as $D > d$, we should learn more parameters and our problem could become far more complex if $D \gg d$ but that is where the dual formulation provides a great framework to use the *kernel trick*.

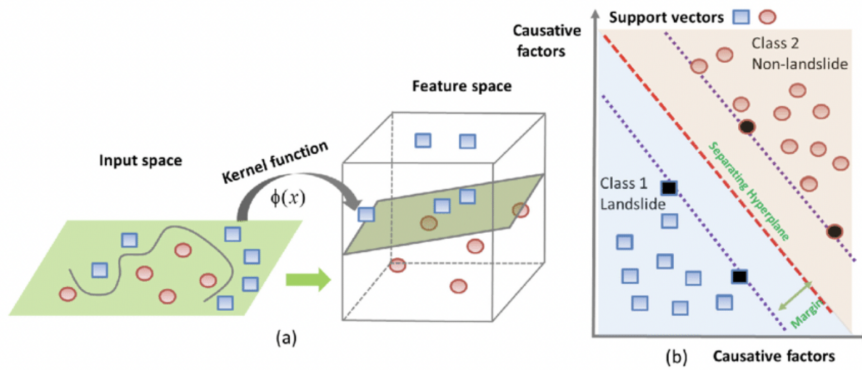


Figure 6: Feature transformation [1]

3.2 The "kernel trick"

As seen in subsection 2.5, the objective function in the dual formulation (5) only manipulates inner products from data instances and hence becomes the following function with the feature transformation:

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \phi(\mathbf{x}_i)^t \phi(\mathbf{x}_j) \quad (6)$$

By defining a *kernel function* $k : R^d \times R^d \rightarrow R$ with $k(\mathbf{x}_i, \mathbf{x}_j) := \phi(\mathbf{x}_i)^t \phi(\mathbf{x}_j)$, we can rewrite the dual (6) as:

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (7)$$

All we need to know is the kernel function in order to compare all training instances with each other. As we do not compute in the implicit feature map projection space, the latter can be of huge dimension, even infinite. The SVM classifier obtained by solving the convex Lagrange dual of the primal max-margin SVM formulation is the following:

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(x, x_i) + b, \text{ where } n \text{ is the number of support vectors.} \quad (8)$$

If you recall the intuition behind a linear discriminant function, the non-parametric SVM classifier given above is very easy to understand. Instead of visualizing the original features of each data point, consider a transformation to a new feature space where the data point has n features, one for each support vector. The value of the i^{th} feature is equal to the value of the kernel between the i^{th} support vector and the data point being classified. In this space, the original (possibly non-linear) SVM classifier is just like any other linear discriminant. Note that after the transformation, the original features of the data point are irrelevant. It is represented only in terms of its dot products with support

vectors as seen in equation (8). This approach is called the *kernel trick* and this makes our model non-linear with respect to the initial input feature space, which means SVM can now be applied to non-linearly separable data.

It is worth adding that the *kernel trick* can be actually applied to other algorithms. In all algorithms where instances are compared between each other through an inner product, such as *linear regression* or *k-NN*, you can therefore use the kernels as well.

3.3 Examples

The simplest example is the **linear kernel** $k(a, b) = a^t b$ which we use within the linear dual SVM formulation. Another frequently used kernel is the **polynomial** one $k(a, b) = (a^t b)^p$ or $k(a, b) = (a^t b + 1)^p$. It corresponds to a feature map projection which contains all polynomials terms up to degree p . The **gaussian kernel (or Radial Basis Function)** defined in (9) is often used to separate more complex data distribution. In certain cases, defining the transformed features in terms of the original features of a data point leads to an infinite-dimensional representation. This is namely the case with the RBF kernel, and the mathematical derivation of this equivalence can be found in Appendix B.

$$\text{Gaussian kernel (or RBF): } k(a, b) = \exp\left(-\frac{\|a - b\|^2}{2\sigma^2}\right) \quad (9)$$

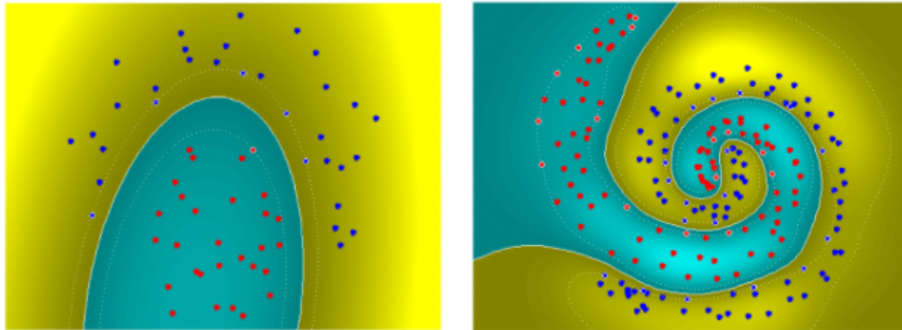


Figure 7: Classifiers built with a linear kernel (left) and a gaussian kernel (right)

Like any other kernel, the RBF kernel can also be understood in terms of the feature transformation via dot products. However, we best analyze the RBF kernel with the concept of the Gaussian distribution. The Gaussian kernel computed with a support vector is an exponentially decaying function in the input feature space, the maximum value of which is attained at the support vector and which decays uniformly in all directions around the support vector, leading to hyper-spherical contours of the kernel function. The SVM classifier with the Gaussian kernel is simply a weighted linear combination of the kernel function computed between a data point and each of the support vectors. The role of a support vector in the classification of a data point is tempered with α , the global prediction usefulness of the support vector, and $K(x, x_i)$, the local influence of a support vector in prediction at a particular data point.

3.4 Kernels for non-numerical data

The *Mercer's theorem* states that a kernel is valid if gives rise to a symmetric, positive semidefinite kernel matrix (or *Gram matrix*) for any input data \mathbf{X} , the kernel matrix being defined as the following:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots \\ \vdots & \ddots & \\ k(\mathbf{x}_N, \mathbf{x}_1) & & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \quad (10)$$

If the kernel does not fulfill the Mercer's condition, our SVM problem might become non-convex, so we may not get a globally optimal solution. This theorem gives us the freedom to arbitrarily create kernels to implicitly compute features map transformations that are useful to linearly split the data. We can even use kernels to encode similarity with non-numerical data such as strings or graphs. Suppose you want to deal with "Coca", "Pepsi" and "Wine". One way to encode similarity between these objects would be to define the kernel function on the support created by the combinations of these three drinks, such as $k(\text{"Coca"}, \text{"Pepsi"}) = -5$, $k(\text{"Coca"}, \text{"Wine"}) = 8$ and $k(\text{"Pepsi"}, \text{"Wine"}) = 7$.

4 SVM in practice

4.1 Test time

At test time, as illustrated in Figure 8 with handwriting digits data, the input data that must be classified will be compared to the support vectors. Then, the class of the input data is inferred thanks to the sign of the weighted sum of these comparisons and the bias.

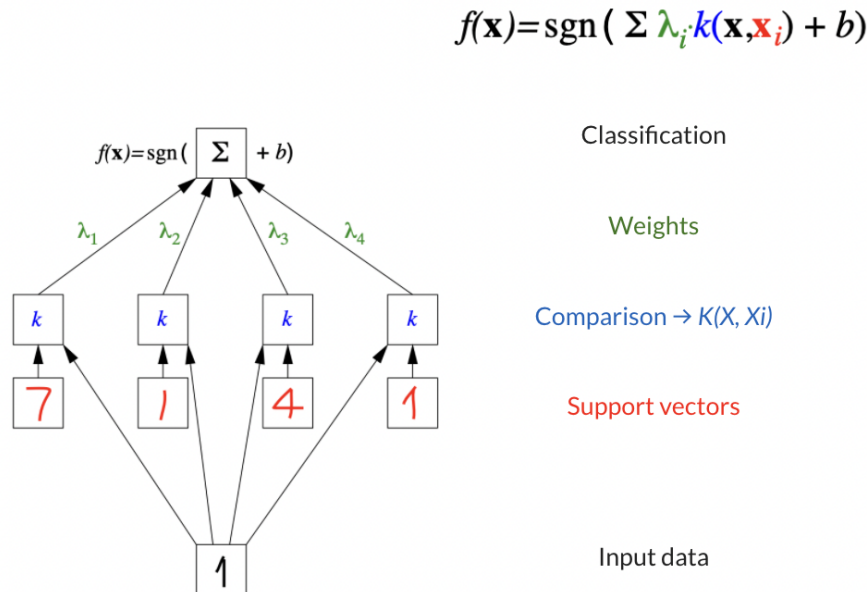


Figure 8: SVM at test time [3]

4.2 Multi-classification

Until now, we have only considered the binary classification problem, but SVMs can easily be extended to multi-classification. Two methods are possible. The first one is called **one-vs-rest** and consists in training C SVM models for C classes, where each SVM is being trained for classification of one class against the remaining ones. The predicted class is then the one, where the distance of the test instance from the hyperplane is maximal. The second version is **one-vs-one** and with this method, we train a classifier for all possible pairings of classes and then evaluate all. The predicted class is the one with the majority vote, the votes being weighted according to the distance from the margin. This second method can be more precise in certain cases but is also more expensive since for C classes, $\binom{C}{2}$ classifiers are trained.

4.3 Applications

Early applications of SVM have been in the field of image classification and object detection such as face detection or handwriting recognition. An example of the latter is given in Figure 9 with the MNIST Benchmark dataset which consists in $60k$ training examples and $10k$ test examples from handwritten digits data of size 28×28 . The results date from 2003 and we can observe that a *translation-invariant SVM*, i.e. using RBF kernels, was able to outperform *LeNet*, the first convolutional neural network developed by Yann Lecun and used by US banks in the 90's.


| | | |
|---|---------------------------|------------|
|  | Classifier | test error |
| | linear classifier | 8.4% |
| | 3-nearest-neighbour | 2.4% |
| | SVM | 1.4% |
| | Tangent distance | 1.1% |
| | LeNet4 | 1.1% |
| | Boosted LeNet4 | 0.7% |
| | Translation invariant SVM | 0.56% |

Figure 9: MNIST dataset and benchmark results, 2003 [1]

4.4 Regression

The SVM problem can also be adapted to regression purpose. In that respect, we formulate the problem as finding a prediction line (or an hyperspace in a high-dimensional feature space, in the non-linear case) which deviates less than ϵ with respect to the ground truth \mathbf{y} . Only the points that lie outside the margin will hence contribute to the final loss which is once again a hinge loss, as you can see with the reformulated optimization problem for the SVM regression (11). An illustration with different kernels can be found

in Figure 10.

$$\begin{aligned}
 & \text{Minimize } \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N \xi_i && \text{w.r.t } \mathbf{w}, \xi_i \\
 & \text{Subject to } |y_i - (\mathbf{w}^t \mathbf{x}_i + b)| \leq \epsilon + |\xi_i| && \text{for } i = 1, \dots, N
 \end{aligned} \tag{11}$$

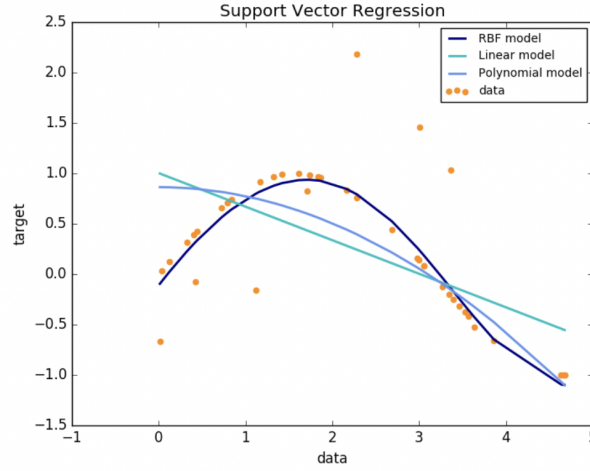


Figure 10: SVM regression with multiple kernels - scikit-learn.org

5 A theoretical approach with the mathematics of learning

5.1 A key algorithm to learning theory

In [2], Tomaso Poggio and Steve Smale, from *MIT* and *Berkeley* universities, outlines the mathematical foundations of the *learning theory* and the link to a key algorithm provided in Figure 11. The main idea from this paper is to provide a theoretical approach to supervised learning, for which they predict, back in 2003, it would become a key technology to extract information from the vast amount of data.

The equation (2) from the algorithm 11 presented in the paper approximates the unknown function by a weighted superposition of gaussian *blobs*, each centered at the location of one the m examples. The algorithm performs well in a number of applications involving regression as well as binary classification, and we can find a clear similarity with a RBF-kernelized SVM.

Mathematically, they show in the paper that this algorithm can be derived from the *Empirical Risk Minimization* described in equation (12), through *Tikhonov regularization*

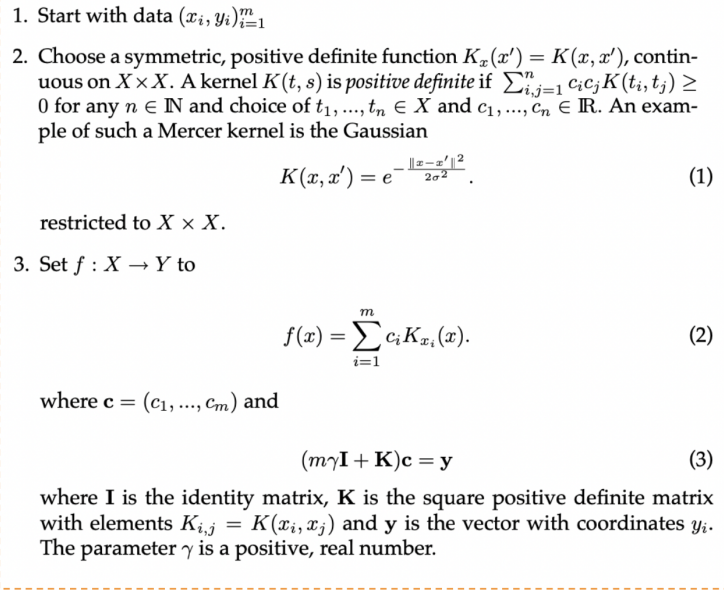


Figure 11: A key algorithm to learning theory [2]

by defining the norm $\|f\|_K$ in the *Reproducing Kernel Hilbert Space (RKHS)* induced by the kernel K , within the hypothesis space \mathcal{H}_K .

$$\frac{1}{m} \sum_{i=1}^m (y_i - f(\mathbf{x}_i))^2 + \gamma \|f\|_K^2, \text{ where } m = N \text{ is the number of training data.} \quad (12)$$

Such a derivation leads to *proximal vector machines*. To retrieve the SVM formulation initially derived in this summary paper, we can replace the quadratic loss by a more adapted non-quadratic loss: the hinge-loss. Coefficients c_i from the proposed key algorithm 11 are then retrieved by solving the quadratic programming problem. With the SVM, the RKHS norm of f can be translated into the norm of the parameters vector \mathbf{w} since our function is parameterized by \mathbf{w} . It is also worth noting here that (12) has a Bayesian interpretation: the left-hand side, related to data, is a model of noise while the RKHS norm corresponds to a prior probability on the hypothesis space \mathcal{H} .

5.2 Theory behind the bias/variance trade-off

One interesting work in the paper is the derivation of the *sample* (or *estimation*) and *approximation* errors. Indeed, one can split the quadratic error (13) into the sum of these two terms. In this equation, f_ρ can be said to be the true input-output function reflecting the environment which produces the data, and f_z is the empirical optimum for the model. S is the **sample error** and must be estimated in probability over z , and hence studied with the theory of probability. On the other hand, A is the **approximation error** and is dealt with the approximation theory. This split is related to the bias/variance trade-off in statistics, which is, in simple terms, the usual balance between a too simple model (biased) and a too complicated one, over-fitting the data and hence being not robust with

respect to new data.

$$\int_X (f_z - f_\rho)^2 = S(z, \mathcal{H}) + A(\mathcal{H}) \quad (13)$$

Here, the approximation error refers to the error incurred by limiting the space of classification models over which search is performed, and the sample error refers to error in estimation of the model parameters. Increasing the number of data points m will decrease the sample error. In contrary, the effect of increasing the complexity of the hypothesis space have two consequences: usually, the approximation error decreases but the sample error increases. This means that there is an optimal complexity of the hypothesis space for a given number of training data. In the case of the key algorithm presented in the paper, this trade-off corresponds to an optimal value γ . Practically, the optimum value is often found through cross-validation. The specific mathematical derivation of the sample and approximation errors for the key algorithm 11 of the paper [2] can be found in Appendix C.

5.3 Understand SVM and its derivation

We have seen that the basis concept of SVM in the linearly-separable case focuses on the concept of separating data with an hyperplane maximizing the margin. In the non-separable case, this interpretation loses most of its meaning. Hence, a more general and simpler framework for deriving SVM algorithms for classification and regression is to regard them as special cases of regularization and follow the proposed key algorithm 11. In this approach, maximizing the margin comes down to minimizing $\|\mathbf{w}\|^2$ which is equivalent to minimizing the RKHS norm. This is the rigorous theory of SVM brought by Vapnik in 1995.

Similarly, we can see with the approach of the *learning theory* that gaussian kernels are universal kernels, i.e. their use with appropriate regularization guarantees a globally optimal predictor which minimizes both the sample and approximation errors of a classifier.

6 Conclusion

We have explored in this summary paper the mathematical roots and the intuition behind SVMs and kernels. The former consist in classification or regression algorithms which learn from labeled data. SVM can be interpreted through the concepts of hyperplanes and margin but also more theoretically from the learning theory. The latter provide a *trick* to help SVMs, and other algorithms, handle non-linearly separable data. Namely, they leverage the dual formulation of SVM and they also take root from the learning theory. Overall, kernelized SVMs are good at high-dimensional data, can work on small data-sets and can solve non-linear problems. Despite their mathematical meaningful interpretation, SVMs can become inefficient on large data and are nowadays outclassed by more sophisticated deep neural networks methods, especially in perceptual tasks such as vision and speech. On structured data, they can also be outperformed by gradient boosted trees. Other disadvantages include not giving class probabilities and being rather cumbersome for multi-class problems, but they still remain a strong baseline for many classification problems, thanks to their interpretability and their speed at test time. Eventually, one last important thing on the kernels must be outlined. The choice of kernels corresponds to either choosing a similarity measure for the data, or choosing a representation for the data distribution, or eventually choosing a hypothesis space for learning. This, in the end, should reflect some prior knowledge about the problem at hand. This highlights once again the *no free lunch theorem* in machine learning: a specific kernel cannot be the solution for all problems.

References

- [1] B. Schölkopf and A. Smola, “Learning with kernels, support vector machines, regularization, optimization and beyond.” *MIT Press, Cambridge, MA*, 2002. [Online]. Available: <http://agbs.kyb.tuebingen.mpg.de/lwk/>
- [2] T. Poggio and S. Smale, “The mathematics of learning: Dealing with data.” *Notices of the American Mathematical Society (AMS)*, vol. 50, no. 5, pp. 537–544, 2003. [Online]. Available: <https://www.dima.unige.it/~devito/Fmas/poggiosmale.pdf>
- [3] A. Zissermann, “Lecture 3: Svm dual, kernels and regression.” *C19 Machine Learning, Oxford University, Oxford*, 2015.
- [4] S. ”Günnemann, “Lecture 8: Svm and kernels.” *IN2064 Machine Learning WS2019, TUM, Munich.*, 2019.

Appendices

A Lagrangian multipliers derivation

We want to transform the primal problem stated in equation (2). In that purpose, using the Lagrange *recipe*, we first calculate the Lagrangian:

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2} \mathbf{w}^t \mathbf{w} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i (\mathbf{w}^t \mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^N \mu_i \xi_i \quad (14)$$

Then, we minimize $L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$ with respect to \mathbf{w} , b , $\boldsymbol{\xi}$:

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) &= \mathbf{w} - \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i = 0; \\ \frac{\partial L}{\partial b} &= \sum_{i=1}^N \alpha_i y_i = 0, \quad \frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \text{ for } i = 1, \dots, N \end{aligned} \quad (15)$$

From $\alpha_i = C - \mu_i$ and dual feasibility $\mu_i \geq 0, \alpha_i \geq 0$, we get $0 \leq \alpha_i \leq C$. This leads to the dual problem defined in equation (5).

B RBF kernel and infinite-dimensional feature projection

Without losing generality, we can set $\sigma = 1$ and consider the following derivation of the gaussian kernel defined in equation (9):

$$\exp\left(-\frac{1}{2} \|a-b\|_2^2\right) = \underbrace{\exp\left(-\frac{1}{2} \|a\|_2^2\right)}_{=c(a)} \underbrace{\exp\left(-\frac{1}{2} \|b\|_2^2\right)}_{=c(b)} \sum_{k=0}^{\infty} \frac{\langle a, b \rangle^k}{k!} = \sum_{k=0}^{\infty} \left\langle \sqrt{\frac{c(a)}{k!}} a, \sqrt{\frac{c(b)}{k!}} b \right\rangle^k \quad (16)$$

We hence see that the gaussian/RBF kernel is equivalent to a feature map projection into an infinite feature space.

C Mathematical derivation of the sample and approximation errors

In subsection 5.2, the discussion depends upon a compact hypothesis space \mathcal{H} from which the experimental optimum f_z and the true optimum $f_{\mathcal{H}}$ are taken. In the proposed key algorithm 11, the optimization runs over all $f \in \mathcal{H}_K$ with a regularized error function. Thanks to two theorems presented in [2] over the sections 3.1 and 3.2, we can explicitly

compute $A(\gamma)$ and $S(\gamma)$. In that respect, let $f_{\gamma,z}$ be the empirical optimum for the regularized problem in equation (12), then $\int (f_{\gamma,z} - f_\rho)^2 \leq S(\gamma) + A(\gamma)$ with:

$$\begin{aligned} A(\gamma) &= \gamma^{1/2} \|L_K^{-\frac{1}{4}} f_\rho\|^2 \\ S(\gamma) &= \frac{32M^2(\gamma + C)^2}{\gamma^2} v^*(m, \gamma) \end{aligned} \tag{17}$$

where $v^*(m, \delta)$ is the unique solution of $\frac{m}{4}v^3 - \ln(\frac{4m}{\delta})v - c_1 = 0$

and all other variables defined as in the paper [2].

The variables $C, c_1 \geq 0$ depend only on X and K . Thus, in this case, the bias-variance problem is to minimize $S(\gamma) + A(\gamma)$ over $\gamma > 0$ for a given m . In that respect there is a unique solution γ for the equation (12).