Predictability, Stability, and Computability of Locally Learnt SVMs

Florian Dumpert

Abstract We will have a look at the principles predictability, stability, and computability in the field of support vector machines. Support vector machines (SVMs), well-known in machine learning, play a successful role in classification and regression in many areas of science. In the past three decades, much research has been conducted on the statistical and computational properties of support vector machines and related kernel methods. On the one hand, consistency (predictability) and robustness (stability) of the method are of interest. On the other hand, from an applied point of view, there is interest in a method that can deal with many observations and many features (computability). Since SVMs require a lot of computing power and storage capacity, various possibilities for processing large data sets have been proposed. One of them is called regionalization. It divides the space of declaring variables into possibly overlapping domains in a data driven way and defines the function to predict the output by the formation of locally learnt support vector machines. Another advantage of regionalization should be mentioned.

Florian Dumpert
Federal Statistical Office of Germany, Wiesbaden
University of Bayreuth, Department of Mathematics, Bayreuth
✉ florian.dumpert@uni-bayreuth.de

Archives of Data Science, Series A
(Online First) DOI: 10.5445/KSP/1000087327/25
KIT Scientific Publishing ISSN 2363-9881
Vol. 5, No. 1, 2018

If the generating distribution in different regions of the input space has different characteristics, learning only one “global” SVM may lead to an imprecise estimate. Locally trained predictors can overcome this problem. It is possible to show that a locally learnt predictor is consistent and robust under assumptions that can be checked by the user of this method.

1 Introduction

In various talks (for instance during the German Probability and Statistics Days 2018 in Freiburg) and publications Bin Yu (University of California, Berkeley) states and discusses three principles of data science: Predictability, stability, and computability (Yu and Kumbier, 2020). Her *Bernoulli* paper (Yu, 2013), for example, focused on stability. Motivated by this, it is worth to check the statistical method at hand whether it fulfills these three principles. Therefore, as a first step in our context, it might be necessary to have a closer look at support vector machines themselves and the notions of predictability, stability, and computability. Please note that, for this paper, we usually suppress technical details like measurability or properties of the spaces. All mentioned sets are considered as measurable. These details can be looked up in the technical paper Dumpert and Christmann (2018). This article embeds the topic into a bit more applied and commented context instead and offers some numerical investigations. Content of this paper is also included in Dumpert (2020b). Section 2 gives an overview of the basic ideas of support vector machines which are examined in more detail under the aspects predictability, stability, and computability in Section 3. It turns out that, unfortunately, support vector machines are not easy to compute. To overcome this disadvantage, Section 4 lists some approaches and discusses the concept of regionalization in a more detailed way. Section 5 summarizes the paper at hand.

2 Support Vector Machines

Support vector machines (SVMs), initially introduced by Boser et al. (1992) and Cortes and Vapnik (1995), are a well-known and highly accepted machine learning method nowadays. Although there is the so-called *no free lunch theorem*, see Devroye (1982), and, therefore, of course counterexamples for the success of SVMs, they are very successful in practice, see e. g. Bennett and Campbell (2000), Caruana and Niculescu-Mizil (2006), Kotsiantis (2007), Caruana et al. (2008), Fernández-Delgado et al. (2014), and Wainberg et al. (2016). They became popular in a lot of fields of science, see e. g. Ma and Guo (2014). Support vector machines for classification are also known as *maximum margin classifiers*, a name which has its origin in the optimization problem solved by an SVM. Originally, the idea was to separate the data points in a two class problem in a linear way, i. e. by a line, a plane or a hyperplane in higher dimensions. If there is more than one possible linear discriminant the one with the largest distance to the data points of the two classes would be chosen. The data points which are constitutive for the separating hyperplane are called *support vectors*. Extensions have been made for the case that the two classes are not linearly separable (soft margin separating hyperplanes) and, of course, towards the concept to find the linear discriminant in higher dimensions (i. e. in the so-called feature space by using the kernel trick).

Other approaches to support vector machines have a more statistical view. The goal of support vector machines in our context, i. e. in classification (or regression), is to discover the influence of a (generally multivariate) input (or explanatory) variable X (with values in $\mathcal{X}$) on a univariate output (or response) variable Y (with values in $\mathcal{Y}$). $\mathcal{X}$, the so-called input space, is usually assumed as a separable metric space. This could be a set of $\mathbb{R}^d$ for a fixed $d \in \mathbb{N}$ which is, in this situation, the number of features considered in training the SVM. The output space $\mathcal{Y}$ is assumed to be a closed subset of the real line $\mathbb{R}$. If $\mathcal{Y}$ is finite, the goal of supervised learning is classification, otherwise it is a regression. Of course, there are generalizations, see for example Micchelli and Pontil (2005) or Caponnetto and De Vito (2007). In this set-up, we are interested in finding a "good" prediction $f(x)$ of y given a certain realization x of X. A deeper discussion to this point can be found in Section 3.1. Find more details in the textbooks by Cristianini and Shawe-Taylor (2000), Schölkopf and Smola (2001), and Steinwart and Christmann (2008).

The approach via the approximation of the decision boundary (the discriminant) given only the data points is also valid. From this point of view, there is a priori no need for a stochastic component in the analysis of the data. There are, e. g., Wendland (2005) and Cucker and Zhou (2007) for a more detailed look at this approach.

3 Predictability, Stability, and Computability

3.1 Predictability

A statistical method, particularly in the field of machine learning, should be able to make good predictions, i. e., in the case of supervised learning, to predict the right class if the output variable is categorical, or (at least almost) the right value if there is a continuous output variable. Yu and Kumbier (2020, p. 3922) define predictability in the following way:

> *“ Predictive accuracy is a simple, quantitative metric to evaluate how well a model represents relationships in* [*given data*] $\mathcal{D}$*. It is well defined relative to a prediction function, testing data, and an evaluation metric. ”*

While Yu and Kumbier (2020) refer to predictability as a reality check (i.e. as a form of empirical validation), we also focus on theoretical investigations on predictability for localized SVMs. This means that we are interested in finding a good approximation of the true discriminant function between the classes or of the true output value producing function, respectively. In the first case, the problem belongs to the field of classification, in the second one to the field of regression. Of course, there are different ways to measure how good the prediction works. Quite common, and also used by Yu and Kumbier (2020), is the approach of minimizing so-called loss functions. The smaller the (expected) loss, the higher is the predictability of the learnt prediction function. As the focus of this paper is on support vector machines we will recall some common loss functions used in this field for classification and regression. Formally, a loss function in the supervised situation is defined as a (for technical reasons) measurable function $L : \mathcal{Y} \times \mathbb{R} \longrightarrow [0, \infty[$.

For binary classification ($\mathcal{Y} = \{-1, 1\}$) there are for example

(a) $L_{LS}(y, f(x)) = (1 - yf(x))^2$,

(b) $L_{hinge}(y, f(x)) = \max\{0, 1 - yf(x)\}$,

and for (quantile) regression ($\mathcal{Y} = \mathbb{R}$) for example

(c) $L_{LS}(y, f(x)) = (y - f(x))^2$,

(d) $L_{\varepsilon-ins}(y, f(x)) = \max\{0, |y - f(x)| - \varepsilon\}$,

(e) $$L_{\alpha-huber}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{, if } |y - f(x)| \leq \alpha \\ \alpha|y - f(x)| - \frac{\alpha^2}{2} & \text{, otherwise} \end{cases}, \alpha > 0.$$

(f) $$L_{\tau-pinball}(y, f(x)) = \begin{cases} (\tau - 1)(y - f(x)) & \text{, if } y - f(x) < 0 \\ \tau(y - f(x)), & \text{, otherwise} \end{cases}, \tau \in]0, 1[.$$

The choice of the loss function depends on the problem at hand. We are also interested in the so-called shifted version L^* of a loss function L, defined by $L^* : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, $L^*(y, t) := L(y, t) - L(y, 0)$. Note that the shifted loss function is only needed for theoretical investigations; in particular: no new algorithms or implementations are needed. If a prediction is exact, we usually expect the loss function L to be 0, i. e. $L(y, y) = 0$ for all $y \in \mathcal{Y}$. The loss functions (a) to (f) fulfill this property.

A short overview of these loss functions summarized in Table 1 shows their properties needed in the remainder. Please note that the properties only consider the second argument of the loss function $L(\cdot, \cdot)$, i. e. these are in fact properties of $L(y, \cdot)$ for all $y \in \mathcal{Y}$. All of the following loss functions are convex (at least with respect to their second argument).

Table 1: Properties of supervised loss functions.

	application	loss function	Lipschitz continuous	twice Fréchet-differentiable	resulting optimization problem[‡]
(a)	classification	L_{LS}	no	yes	LP
(b)	classification	L_{hinge}	yes	no	boxed QP
(c)	regression	L_{LS}	no	yes	LP
(d)	regression	$L_{\varepsilon-ins}$	yes	no	boxed QP
(e)	regression	$L_{\alpha-huber}$	yes	no	boxed QP
(f)	regression	$L_{\tau-pinball}$	yes	no	boxed QP

[‡] where LP stands for a linear program and boxed QP for a quadratic program with box constraints

Obviously, there is no jack of all trades device, i. e. no “best” loss function for classification or regression. Beyond these, other loss functions have been considered in the literature, in particular:

1. The Lipschitz-continuous and twice Fréchet-differentiable logistic loss function $L_{r-logistic}(y, f(x)) = -\ln\left(\frac{4e^{y-f(x)}}{(1+e^{y-f(x)})^2}\right)$ for regression. For its genesis see Dekel et al. (2005). Hable (2012) used the logistic loss for numerical experiments in the context of the asymptotic normality of SVMs.
2. $L_{c-logistic}(y, f(x)) = \ln(1 + e^{-yf(x)})$ for classification. It was already used by Vapnik (1995) with a convex program as resulting optimization problem.

Further discussions of loss functions are for example given by Rosasco et al. (2004) and Steinwart (2007). From a mathematical point of view, the goal is to find a minimizer of the average loss (over all possible inputs and outputs with respect to the unknown underlying distribution P of the input and output variables). Other approaches measure the difference between the predictors (which are functions) and the true or the best approximating function. The average loss is called the risk over $\mathcal{X}$ of a measurable predictor f with respect to the chosen loss function L (with shifted version L^*) and the unknown underlying distribution P.

It is formally defined by

$$\mathcal{R}_{\mathcal{X},L^*,P} : \{f : \mathcal{X} \to \mathbb{R} \mid f \text{ measurable}\} \to \mathbb{R}, \tag{1}$$

$$\mathcal{R}_{\mathcal{X},L^*,P}(f) := \int_{\mathcal{X}\times\mathcal{Y}} L^*(y, f(x))\, dP(x, y). \tag{2}$$

Naturally, we want to compare our result to the best, i. e. to the minimal risk which we can reach by using a measurable predictor. The best risk is

$$\mathcal{R}^*_{\mathcal{X},L^*,P} := \inf\left\{\mathcal{R}_{\mathcal{X},L^*,P}(f) \mid f : \mathcal{X} \to \mathbb{R} \text{ measurable}\right\}, \tag{3}$$

which is called the Bayes risk on $\mathcal{X}$ with respect to L (with shifted version L^*) and P. Inspired by the law of large numbers, we use the information contained in the (finite and i. i. d.) sample at hand $(x_i, y_i)_{i=1,\dots,n} \in (\mathcal{X} \times \mathcal{Y})^n$ to approximate the above-mentioned risks. Let

$$D_n := \frac{1}{n} \sum_{i=1}^{n} \delta_{(x_i, y_i)} \tag{4}$$

be the empirical distribution based on the data points in the sample where $\delta_{(x_i, y_i)}$ is the Dirac measure at a point $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Of course, this empirical distribution is random because the sample is a realization of random variables. Now, we can define the empirical risk as

$$\mathcal{R}_{\mathcal{X}, L^*, D_n}(f) = \frac{1}{n} \sum_{i=1}^{n} L^*(y_i, f(x_i)). \tag{5}$$

As we usually allow f to be chosen from a very rich class of functions (in order to approximate the true relationship), we have to control the complexity of the predictor to avoid overfitting. Obviously, no one could be interested in interpolating all the data by a polynomial of an extraordinary high degree. Therefore we add a regularization term $\lambda \|f\|_H^2$ that punishes complex functions. The hyperparameter $\lambda > 0$ stands for the influence of the regularization term compared to the risk. Note that λ should depend on the number of observations n: The more data points are given the more complex the function f is allowed to be. H is a so-called reproducing kernel Hilbert space (RKHS; Aronszajn (1950), Berlinet and Thomas-Agnan (2004), Paulsen and Raghupathi (2016)) of measurable functions.

This type of function spaces can be very rich in the sense that they contain enough functions for a very good approximation of any functional relationship between X and Y. On the other hand, these in our context principally infinite dimensional RKHSs allow to solve optimization problems over them by applying numeric methods on finite dimensional (nonlinear) programs. By choosing a suitable kernel one determines which functions are contained in an RKHS. This raises the question which kernel the user should choose. As its RKHS contains enough functions to approximate every measurable function in probability, see Christmann et al. (2016), a default choice can be the Gaussian RBF kernel for some $\gamma > 0$, $k_\gamma(x, x') := \exp\left(-\gamma^{-2} \|x - x'\|_{\mathcal{X}}^2\right)$, $x, x' \in \mathcal{X}$, where $\|\cdot\|_{\mathcal{X}}$ is a suitable norm on $\mathcal{X}$ (e. g., for $\mathcal{X} = \mathbb{R}^d$, the Euclidian norm $\|x\|_2 := \left(\sum_{i=1}^{d} x_i^2\right)^{1/2}$ for $x = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$).

The data influences the kernel only via the norm of differences in a bell shaped function which motivates the denomination *Gaussian radial basis function (RBF) kernel*. But, of course, there are other kernels with special properties a user can choose, see in particular Wendland (1995), Wu (1995), and Wendland (2005). Summarizing these prerequisites, we want to

$$\text{minimize } \mathcal{R}_{\mathcal{X},L^*,D_n,\lambda_n}(f) := \frac{1}{n}\sum_{i=1}^{n} L^*(y_i, f(x_i)) + \lambda_n \|f\|_H^2 \tag{6}$$

over a reproducing kernel Hilbert space of functions (for instance over a suitable subset of continuous or P-integrable functions) based on a sample of observations $(x_i, y_i)_{i=1,\ldots,n}$ created by P. Hence, we want to find the empirical support vector machine

$$f_{L^*,D_n,\lambda_n} := \arg\inf_{f \in H} \frac{1}{n}\sum_{i=1}^{n} L^*(y_i, f(x_i)) + \lambda_n \|f\|_H^2. \tag{7}$$

It is well known that support vector machines exist, are unique and are universal consistent in the i. i. d. case, i. e.

$$\mathcal{R}_{\mathcal{X},L^*,P}(f_{L^*,D_n,\lambda_n}) \xrightarrow[n\to\infty]{} \mathcal{R}^*_{\mathcal{X},L^*,P} \quad \text{in probability w.r.t. } P \tag{8}$$

if the loss function is convex and Lipschitz-continuous (in this situation, the shifted version is convex and Lipschitz-continuous, too) and if we use a bounded and measurable kernel, see Steinwart and Christmann (2008) and Christmann et al. (2009). Universal consistency can also be shown for some non-i. i. d. situations, see Steinwart et al. (2009), Strohriegl (2018). Therefore, the aspect *predictability* is fulfilled under mild conditions the user can easily check or even influence.

3.2 Stability

A learnt predictor should not change "too much" if the sample contains observations with errors, an experiment is repeated, a new sample is drawn, or the underlying distribution has changed only slightly. Ideally, a predictor is, furthermore, not vulnerable to outliers in the sample. The following quote highlights the view of Yu (2013) and Yu and Kumbier (2020, pp. 3923 f.) on the notion of stability.

> "*At the modeling stage, stability measures how a data result changes when the data and/or model are perturbed* [...]. *Stability extends the concept of sampling variability in statistics, which is a measure of instability relative to other data that could be generated from the same distribution.* [...] *To evaluate the stability of a data result, we measure the change in target $\mathcal{T}$* [*the stability target $\mathcal{T}(\mathcal{D}, \lambda)$ corresponds to the data result or estimand of interest. It depends on input data $\mathcal{D}$ and a specific model/algorithm λ used to analyze the data.*] *that results from a perturbation to the input data or learning algorithm.*"

In statistics, these properties are sometimes also known as *robustness*. Shawe-Taylor and Cristianini (2004, p. 13) use the two terms *robustness* and *statistical stability*:

> "*Robustness*: [...] [*A*]*n effective pattern analysis algorithm must address is the fact that in real-life applications data is often corrupted by noise. By noise we mean that the values of the features for individual data items may be affected by measurement inaccuracies or even miscodings, for example through human error.* [...] [*The algorithms*] *should therefore tolerate a small amount of noise in the sense that it will not affect their output too much.* [...] *Statistical stability*: [...] [*T*]*he patterns the algorithm identifies really are genuine patterns of the data source and not just an accidental relation occurring in the finite training set. We can view this property as the statistical robustness of the output in the sense that if we rerun the algorithm on a new sample from the same source it should identify a similar pattern. Hence, the output of the algorithm should not be sensitive to the particular dataset, just to the underlying source of the data.* [...] *A relation identified by such an algorithm as a pattern of the underlying source is also referred to as* stable, significant *or* invariant*.*"

Shawe-Taylor and Cristianini (2004, p. 13) also add that

> "*there is some overlap between robustness and statistical stability in that they both measure sensitivity of the pattern function to the sampling process. The difference is that robustness emphasise the effect of the sampling on the pattern function itself, while statistical stability measures how reliably the particular pattern function will process unseen examples.*"

Note that, by this, there is a direct link between *statistical stability* and *predictability*. The latter one deals in the setting of Section 3.1 also with the ability of a method to generalize from the training data to unseen examples.

There are a lot of concrete notions of stability and robustness of a statistical method. Support vector machines fulfill some of them, in particular quantitative robustness in terms of the maximal difference (in some metric) between two SVMs (based on different samples from the same population or based on samples from two slightly different populations) or in terms of the so-called influence function. Let $\mathcal{M}_1(\mathcal{X}\times\mathcal{Y})$ be the set of all distributions on $\mathcal{X}\times\mathcal{Y}$. If one interprets the support vector machine as the result of a mapping $S : \mathcal{M}_1(\mathcal{X}\times\mathcal{Y}) \to H$, i. e. $S(D_n) = f_{L^*,D_n,\lambda_n}$ is the empirical SVM, the influence function can be thought of as a derivative of S at the point P in the direction of another measure if it exists. Note that, in this case, there are usually stronger differentiability properties of the loss function needed: L has to be twice Fréchet-differentiable with bounded first and second derivative (with respect to the second argument, of course), see Christmann et al. (2009). This assumption can be weakened by using Bouligand-derivatives, see Christmann and Van Messem (2008), Van Messem and Christmann (2010). Another notion of robustness also fulfilled by support vector machines is qualitative robustness, see Hampel (1971), but we will not consider this within this paper. Qualitative robustness is defined in terms of the equicontinuity (uniformly with respect to the sample size n) of the distribution of the estimator S.

For this paper, we restrict ourselves to the so-called "maxbias". Investigations on other robustness properties can be found in Christmann and Van Messem (2008), Christmann et al. (2009), Hable and Christmann (2011), Dumpert (2020a) and, for some non-i. i. d. situations, in Strohriegl and Hable (2016) and Strohriegl (2018). In particular, total stability (i. e. with respect to the underlying distribution, the sample, the regularization parameter and the choice of the kernel) of support vector machines in the i. i. d. case has been proved in Christmann et al. (2018).

Formally, by using $S : \mathcal{M}_1(\mathcal{X}\times\mathcal{Y}) \to H$, the maxbias can be defined by observing the behaviour of S in a *contamination neighbourhood* $N_\varepsilon := N_\varepsilon(P)$ of a distribution $P \in \mathcal{M}_1(\mathcal{X}\times\mathcal{Y})$, i. e. in

$$N_\varepsilon = N_\varepsilon(P) := \{(1-\varepsilon)P + \varepsilon Q \mid Q \in \mathcal{M}_1(\mathcal{X}\times\mathcal{Y})\} \tag{9}$$

for $\varepsilon \in [0, \frac{1}{2}[$. This neighbourhood N_ε contains the original distribution P but also distributions which are more or less P but a bit (ε) disturbed by another distribution Q. Q can represent outliers, gross errors and so on. The corresponding maxbias of S (i. e. of the method "SVM") is then defined by

$$\begin{aligned}\text{maxbias}(\varepsilon, S, P) &:= \sup_{Q \in N_\varepsilon} \|S(Q) - S(P)\|_H \\ &= \sup_{Q \in N_\varepsilon} \|f_{L^*,Q,\lambda} - f_{L^*,P,\lambda}\|_H . \end{aligned} \tag{10}$$

A kernel k is called bounded if $\|k\|_\infty := \sup_{x \in \mathcal{X}} \sqrt{k(x,x)} < \infty$. If and only if the reproducing kernel k of an RKHS H is bounded, every $f \in H$ is bounded. In this situation, the inequality $|f(x)| = |\langle f, k(\cdot, x)\rangle_H| \leq \|f\|_H \|k\|_\infty$ holds true for all $f \in H, x \in \mathcal{X}$. Particularly: $\|f\|_\infty \leq \|f\|_H \|k\|_\infty$. Therefore it is also useful to have a look at

$$\sup_{Q \in N_\varepsilon} \|S(Q) - S(P)\|_\infty = \sup_{Q \in N_\varepsilon} \|f_{L^*,Q,\lambda} - f_{L^*,P,\lambda}\|_\infty . \tag{11}$$

There is an upper bound for Lipschitz-continuous loss functions and the Gaussian RBF kernel, see Steinwart and Christmann (2008):

$$\sup_{Q \in N_\varepsilon} \|f_{L^*,Q,\lambda} - f_{L^*,P,\lambda}\|_\infty \leq \frac{\varepsilon}{\lambda} |L|_1 \|k\|_\infty^2 \|Q - P\|_{TV} \leq 2 \frac{\varepsilon}{\lambda} |L|_1 \tag{12}$$

where $\|\cdot\|_{TV}$ is the total variation norm on $\mathcal{M}_1(\mathcal{X} \times \mathcal{Y})$ (which can itself get upperbounded by 2) and $|L|_1$ the Lipschitz constant of the Lipschitz-continuous loss function (and its shifted version).

This is a finite upper bound for the maximum difference between two SVMs based on two slightly different samples.

Hence, we see that support vector machines fulfill the property *stability* (in the sense of the maxbias) under mild assumptions the user of the method can check.

3.3 Computability

The following quote from Yu and Kumbier (2020, p. 3923) shows the importance (and a definition) of computability.

> *“ In a broad sense, computability is the gatekeeper of data science. If data cannot be generated, stored, managed, and analyzed efficiently and scalably, there is no data science. ”*

Another formulation of computability (computational efficience) is written down in Shawe-Taylor and Cristianini (2004, p. 12):

> *“ Since we are interested in practical solutions to real-world problems, pattern analysis algorithms must be able to handle very large datasets. Hence, it is not sufficient for an algorithm to work well on small toy examples*; *we require that its performance should scale to large datasets. ”*

Although there are no theoretical issues in situations with a lot of observations (large n) and/or a lot of explanatory variables (large d), and even not for the situation $d > n$, i. e. SVMs can theoretically deal with these situations, difficulties from a practical point of view arise. To compute the predictor, i. e. the support vector machine, the sample has to get stored, the kernel has to be evaluated at all 2-tuples (x_j, x_l) for x_j, x_l appearing in the observations $(x_i, y_i)_{i=1,\dots,n}$ in the sample (and this kernel matrix has to get stored, too) and a nonlinear optimization problem has to be solved. (Note that the discussion of efficient solvers for nonlinear optimization problems is out of the scope of this paper.) Hence, a lot of storage capacity (ideally RAM) and computational power is needed to find an SVM given a sample of n observations. The complexity of learning an SVM is $O(n^3)$ for time and $O(n^2)$ for memory. Some basic discussions of this can also be found in Zhou et al. (2014) or Steinwart and Christmann (2008, Chapter 11). Obviously, computability is a problem for SVMs when n and/or d is large.
Among decomposition methods and feature or subset selection techniques, one might think about other approaches to address the computational problems. Section 4 considers these approaches.

4 Addressing the Problem of Computability

4.1 A Short Overview

There are different approaches to address the problem of computability (or scalability). Some of them should be mentioned in this paper. Recall that n denotes the number of observations, and d stands for the number of features for every observation, i. e. for the number of input variables.

1. *Feature selection* to reduce d. A general overview is given by Guyon and Elisseeff (2003) and the references therein (concerning the special issue 3/2003 of the Journal of Machine Learning Research). Early approaches for feature selection for SVMs are, among others, Hermes and Buhmann (2000), Weston et al. (2001), and, e. g., Claeskens et al. (2008). A recent overview and further theoretical investigations of this topic are given by Zhang et al. (2016).

2. *Low-rank approximations* of the kernel matrix to reduce n and d (based on the idea that suitable chosen parts of the sample already contain enough information) and approximations of the kernel function. There are a lot of different ways to do this, like single value decomposition, CUR matrix decomposition or different Nystrom methods. Bach (2013) or Si et al. (2017) provide (beside their own advances) overviews of the different approaches in this field.

3. *Sequential* or *online learning* to reduce n per time unit, see e.g. Smale and Yao (2006), Ying and Zhou (2006), Ying and Pontil (2008), and Guo et al. (2017b). In this case, the data is not completely available or at least not completely used at the beginning of learning but in a sequential way. Therefore, the learnt predictor becomes updated when new data points "arrive" or are considered.

4. *Distributed learning* to reduce n per processing unit (but the whole sample can be used), see e. g. Christmann et al. (2007), Duchi et al. (2014), Guo et al. (2017a), Mücke (2017), and Guo et al. (2017a). The big advantage of this approach is the high scalability in the sense that more and more processing units can be used to calculate the predictors on

the subsamples. Nevertheless, statistical properties of the data in different parts of the sample are probably not preserved.

5. *Local learning* in the specification that if one needs a prediction for a new data point, the training data is observed only locally around the new data point, see e. g. Zakai and Ritov (2009), Blanzieri and Bryl (2007), Blanzieri and Melgani (2008), or Hable (2013). Of course, this approach does not need any training time on the whole sample, but little training time whenever a new data point needs to get a prediction. If, for example, there are not many new data points to classify, this is obviously very advantageous.
6. *Local learning* in the specification that the input space gets regionalized in advance. If one needs a prediction for a new data point, this only depends on the region(s) where the data point belongs to. See Subsection 4.2 for details.

Note that the local learning versions mentioned in 5 and 6 are not identical and offer actually two different ways to localize the learning process. Of course, combinations or successive use of approaches are possible, too, see for example Mücke (2019). Note that the six mentioned classes do not exploit all possibilities to deal with the problem of computability (or scalability). Other approaches are, e. g., gradient descent with early stopping regularization/iterative regularization, see Guo et al. (2018) and Lin et al. (2016) and the references therein.

4.2 The Idea of Regionalization

The idea of localized statistical learning has already been thought in the beginning of support vector machines. Bottou and Vapnik (1992) and Vapnik and Bottou (1993) give some theoretical investigations. An early overview of different ideas of localizing and combining can be found in Collobert et al. (2002).
A big advantage of regionalization is the parallelization of the calculations for different regions. For numerical experiences see, e. g., Bennett and Blue (1998), Wu et al. (1999), or Chang et al. (2010).

But there is another motivation to consider localized (particularly instead of distributed learning) approaches. For example, there might be regions that require a simple function as predictor for the class or the regression value; another region might need a more volatile function. While a global machine learning approach has to find a global predictor (with, in the case of SVMs, only one global regularization parameter λ and only one possible kernel k), a local approach would overcome this disadvantage.

Thus, the idea of localized learning contains, as a first step, a regionalization method which divides the input space $\mathcal{X}$ into a finite number of (possibly overlapping) regions **(R1)**. In formulae: $\mathcal{X} = \bigcup_{b=1}^{B} \mathcal{X}_b$. Of course, there are regionalization methods which satisfy these requirements. Among these are, e. g., trees in the sense of Breiman et al. (1984), K-means clustering, or Voronoi partitions, see for example Aurenhammer (1991).

The regionalization method has to guarantee that the resulting regions are measurable and complete for all distributions **(R2)**.

In particular, that if the sample gets larger and larger, every region contains more and more data points, i. e.,

$$\lim_{n\to\infty} \min_{b\in\{1,\dots,B\}} |\mathcal{D}_n \cap \mathcal{X}_b| = \infty\,, \tag{13}$$

where $\mathcal{D}_n = (x_i, y_i)_{i=1,\dots,n}$ is the sample and $|M|$ denotes the number of data points in a set M **(R3)**.

Then, it is, as a second step, possible to learn local (one for each region) SVMs $f_{b,L^*,D_{n,b},\lambda_b}$, $b = 1,\dots,B$, on $\mathcal{X}_1 \times \mathcal{Y}, \dots, \mathcal{X}_B \times \mathcal{Y}$ with kernels $k_1,\dots,k_B$ and corresponding reproducing kernel Hilbert spaces $H_1,\dots,H_B$ and combine them in a weighted way to a predictor on the whole input space:

$$f^{comp}_{L^*,D_n,\lambda} : \mathcal{X} \to \mathbb{R}, \qquad f^{comp}_{L^*,D_n,\lambda}(x) := \sum_{b=1}^{B} w_b(x) f_{b,L^*,D_{n,b},\lambda_b}(x) \tag{14}$$

where $D_{n,b}$ denotes the empirical measure on $\mathcal{X}_b \times \mathcal{Y}$ and the weights w_b, $b = 1,\dots,B$, only have to fulfill $\sum_{b=1}^{B} w_b(x) = 1$ for all $x \in \mathcal{X}$, and $w_b(x) = 0$ for all $x \notin \mathcal{X}_b$ and for all $b \in \{1,\dots,B\}$.

4.3 Properties of Locally Learnt SVMs

In order to show that locally learnt SVMs fullfil the desired properties predictability and stability in theory and practise, we cite and specialize two results from Dumpert and Christmann (2018) where these results have been proved rigorously and in a more general form by restricting to a special kernel, the widely used Gaussian RBF kernel, the standard input space in practice $\mathcal{X} = \mathbb{R}^d$ for a $d \in \mathbb{N}$, and the standard output space in practice $\mathcal{Y} = \{1, ..., C\}$ for a $C \in \mathbb{N}$, $C \geq 2$, (for classification) or $\mathcal{Y} = \mathbb{R}$ (for regression).

4.3.1 Predictability

Theorem 4.1 (Universal consistency)**.** *Let L be a convex, Lipschitz-continuous (with Lipschitz-constant $|L|_1 \neq 0$) loss function and L^* its shifted version. For all $b \in \{1, \dots, B\}$ let k_b be a Gaussian RBF kernel on $\mathcal{X}_b$. Let the regionalization method fulfill the technical assumptions (R1), (R2), and (R3) from Subsection 4.2.*

Then for all distributions P on $\mathcal{X} \times \mathcal{Y}$ and every collection of sequences $\lambda_{(n_1,1)}, \dots, \lambda_{(n_B,B)}$ with $\lambda_{(n_b,b)} \to 0$ and $\lambda^2_{(n_b,b)} n_b \to \infty$ when $n_b \to \infty$, $b \in \{1, \dots, B\}$, it holds true that

$$\mathcal{R}_{\mathcal{X},L^*,P}(f^{comp}_{L^*,D_n,\lambda_n}) \xrightarrow[n\to\infty]{} \mathcal{R}^*_{\mathcal{X},L^*,P} \quad \textit{in probability with respect to } P.$$

This result expresses that the average loss — over all $(x, y) \in \mathbb{R}^d \times \{1, ..., C\}$ (for classification) or all $(x, y) \in \mathbb{R}^d \times \mathbb{R}$ (for regression) which are distributed according to the unknown distribution P — we can achieve by locally learnt support vector machines (learnt based on n data points) converges for data sets that become larger ($n \to \infty$) to the best (i. e. the minimal) possible average loss.

Note that, in this situation, there is no learning rate shown, i. e. we cannot say how fast the average loss converges to the best loss. This might be a disadvantage of this theorem because it is not possible to say how many observations are needed to be sufficiently close to the Bayes risk in this situation. On the other hand, the theorem gives universal consistency for all distributions P under conditions the user can check: The choice of a loss function which is Lipschitz-continuous is possible (see Subsection 3.1). It should also be possible to use a Gaussian RBF kernel. And the user can choose the regionalization method.

Of course, the value of the regularization parameter λ_b in the regions $\mathcal{X}_b, b \in \{1, \dots, B\}$, which corresponds to the cost parameter C_b often used in implementations usually via $C_b = (2\lambda_b)^{-n_b}$, has to be set carefully by the user. No further (uncheckable) assumptions on the existence of moments, of densities, or of symmetry aspects of the underlying distribution or the sample have to be made. This might be very important in statistics where we usually do not know properties of P. If more information about P is available and if the regions are disjoint, results like Eberts (2015), Meister and Steinwart (2016), and Thomann et al. (2017) could be of interest.

4.3.2 Stability

As a second aspect, we have to consider stability. Let $\mathcal{M}_1(M)$ again denote the set of probability measures on a set M, and let $P_{|Z}$ denote the restriction of a measure P on a set Z, i. e. $P_{|Z}(A) := P(A \cap Z)$ for all sets A. For all $b \in \{1, \dots, B\}$ and $\varepsilon_b \in [0, \frac{1}{2}[$ let $P_b := P(\mathcal{X}_b \times \mathcal{Y})^{-1} P_{|\mathcal{X}_b \times \mathcal{Y}}$ if $P(\mathcal{X}_b \times \mathcal{Y}) \neq 0$ and the null measure otherwise, i. e. the local distribution of (X, Y) on $\mathcal{X}_b \times \mathcal{Y}$. Define for a distribution P on $\mathcal{X} \times \mathcal{Y}$ the ε_b-contamination neighbourhood of P (or P_b) on $\mathcal{X}_b \times \mathcal{Y}$

$$N_{b,\varepsilon_b}(P) := N_{b,\varepsilon_b}(P_b) := \left\{ (1 - \varepsilon_b) P_b + \varepsilon_b \tilde{P}_b \;\middle|\; \tilde{P}_b \in \mathcal{M}_1(\mathcal{X}_b \times \mathcal{Y}) \right\} .$$

Furthermore, let $\varepsilon := (\varepsilon_1, \dots, \varepsilon_B) \in [0, \frac{1}{2}[^B$. By using these notations, we can define

$$N_\varepsilon^{comp}(P) := \left\{ \tilde{P} \in \mathcal{M}_1(\mathcal{X} \times \mathcal{Y}) \;\middle|\; \tilde{P}_b \in N_{b,\varepsilon_b}(P) \text{ for all } b \in \{1, \dots, B\} \right\}$$

as a composed ε-contamination neighbourhood of P on $\mathcal{X} \times \mathcal{Y}$.

Theorem 4.2 (Robustness). *Let L be a convex, Lipschitz-continuous (with Lipschitz-constant $|L|_1 \neq 0$) loss function and L^* its shifted version. For all $b \in \{1, \dots, B\}$ let k_b be a Gaussian RBF kernel on $\mathcal{X}_b$. Let the regionalization method fulfill the technical assumptions from Subsection 4.2. Then, for all distributions P on $\mathcal{X} \times \mathcal{Y}$ and all $\lambda := (\lambda_1, \dots, \lambda_B) \in]0, \infty[^B$, it holds that*

$$\sup_{\tilde{P} \in N_\varepsilon^{comp}(P)} \left\| f_{L^*,\tilde{P},\lambda}^{comp} - f_{L^*,P,\lambda}^{comp} \right\|_\infty \leq 2 \, |L|_1 \sum_{b=1}^{B} \|w_b\|_{\mathcal{X}_b \infty} \frac{\varepsilon_b}{\lambda_b} ,$$

where $\|\cdot\|_{\mathcal{X}_b \infty}$ denotes the supremum norm on $\mathcal{X}_b$.

To comment this a bit, we should have a look at the question whether we get worse in terms of robustness when we use the regionalization approach to gain computability. Let us, therefore, compare the shown upper bound for the maxbias with the one without regionalization (see Subsection 3.2). It is obvious that if we set the number of regions B to 1 (such that there is no regionalization) the upper bounds (12) and (15) coincide. This means that we do not lose robustness properties by using regionalization. Note that Theorem 4.1 requires $\lambda_{(n_b,b)}$ to converge to 0 when $n_b \to \infty$. Theorem 4.2, however, states for fixed $n_1, \ldots, n_B$ that the smaller λ_b is, $b \in \{1, \ldots, B\}$, the higher is the upper bound. This shows, roughly spoken, that the more precise the local SVMs are, the less robustly they behave, and vice versa. Note that this phenomenon already appears in the case of one global SVM (thus, it is not a disadvantage introduced by regionalization) and can be interpreted, to some extent, as a bias-variance-trade-off of the mapping S. This has been discussed in more detail in Steinwart and Christmann (2008, Chapter 10) and for the notion of qualitative robustness in Hable and Christmann (2013).

4.3.3 Computability

Investigations on computability can be done by using the R package liquidSVM, Steinwart and Thomann (2017), R Core Team (2018), which builds its regions by using Voronoi-diagrams, i. e. by using non-overlapping regions. Therefore, five scenarios are shown for a short toy binary classification example to visualize aspects considered in this paper:

1. One global SVM based on `liquidSVM` without creating regions;
2. Three local SVMs based on `liquidSVM` on manually defined regions;
3. As much as local SVMs as `liquidSVM` finds by its internal regionalization (`partition_choice=5`);
4. As much as local SVMs as `liquidSVM` finds by its internal regionalization (`partition_choice=6`);
5. Local SVMs based on `liquidSVM` on regions defined by a tree based on `rpart` in a first step; for `rpart` see Therneau and Atkinson (2018).

The sample contains the available information on the distribution of the blue and the red class. The true distribution of the two classes is shown in Figure 2. This distribution can be reproduced by the R-code shown in Figure 1. x stands for the first input dimension, y for the second input dimension. k denotes the class (0 or 1; blue or red) a data point belongs to.

```
x = runif(n, 0, 1)
y = runif(n, 0, 1)
k = rep(0, n)

for  (i in 1:n)
{
 if (x[i]<0.16
      && (y[i]-0.5)< 0.25*sin(50*x[i])-0.15)
        k[i]=1

 if (x[i]<0.16
      && (y[i]-0.5)> 0.25*sin(50*x[i])+0.15)
        k[i]=1

 if (x[i]>=0.16
       && x[i]<0.83
       && y[i]<0.6
       && y[i]>0.4)
        k[i]=1

 if (x[i]>=0.83
       && y[i]>=0.5-20*(x[i]-0.83)^2-0.05
       && y[i]<=0.5-20*(x[i]-0.83)^2+0.05)
        k[i]=1
```

Figure 1: R-code.

To give comparable values for computational times, parallel computing has been switched off. We compare the five scenarios in terms of the time needed for computations (measured by the differences in `proc.time()` (user)) and accuracy (number of correct classifications divided by the length of the test set) on a test set of the length 25 000. There have been 10 runs per scenario and training sample length (750, 10 000, 50 000). The calculations have been done on a Intel Xeon CPU (E5-2640 v4), 2.4 GHz, 16 GB RAM, a 64-bit Windows 7 and using R 3.5.1. In order to visualize the result of the proposed method, Figure 3 shows for Seed 1000 the classification of 25 000 test data points based on a model learnt on 10 000 training data points.

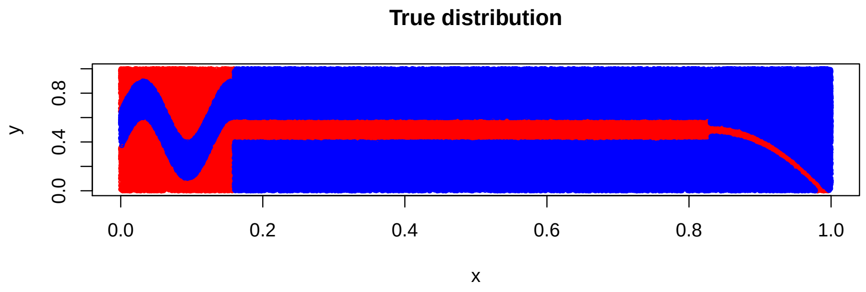


Figure 2: True distribution of the two classes (red and blue).

The results are summarized by six boxplots (Figures 4, 5, and 6). Figure 4 shows that, as long as the training sample is small, the local SVMs on manually defined regions perform best in terms of accuracy; the other approaches have not been able to discover the different regions of the data set. As this is, however, not the situation which the new approach is made for, a closer look to Figures 5 and 6 is necessary. In Figure 5, we observe with respect to time the very efficient implementation of liquidSVM as well as the fact that the regionalization that uses a tree compares to liquidSVM. The global solution is (as it was expected) the one that needs the most time to get learnt. While the accuracy results of liquidSVM are comparable to the global solution, the proposed regionalization approach achieved higher accuracy. In Figure 6, these findings are confirmed although it is not really clear, why the global SVM (learnt by liquidSVM) can be so fast at the scenario with 50 000 training points.

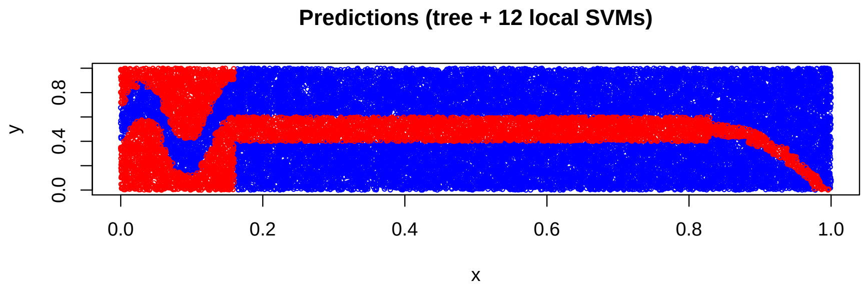
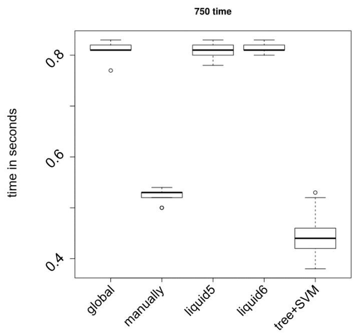
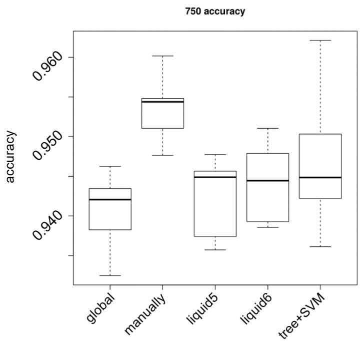


Figure 3: Result for local SVMs on regions found by a tree based on 10 000 training points.

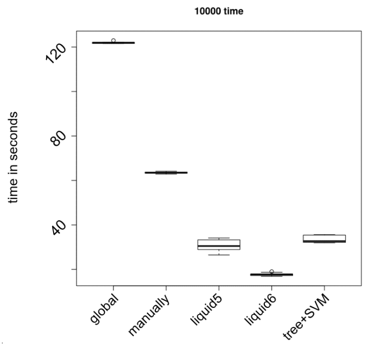


(a) time in seconds.

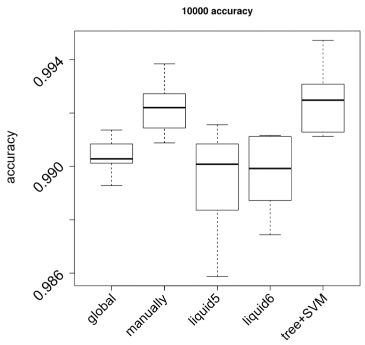


(b) accuracy (0.99 = 99 %).

Figure 4: Summarized results for 750 training points.

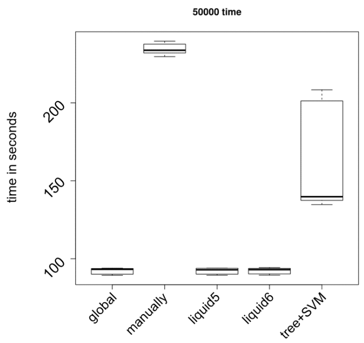


(a) time in seconds.

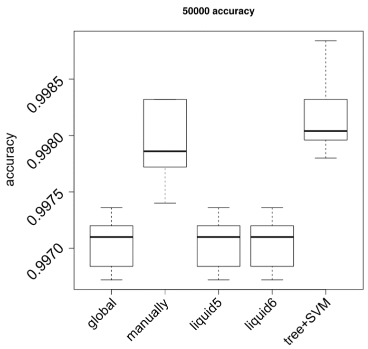


(b) accuracy (0.99 = 99 %).

Figure 5: Summarized results for 10 000 training points.



(a) time in seconds.



(b) accuracy (0.99 = 99 %).

Figure 6: Summarized results for 50 000 training points.

5 Conclusion

Numerical investigations showed that the regionalization approach described in Subsection 4.2 leads to at least the same accuracy as the global approach. Whether there is also an advantage in time depends on how fast the tree finds regions and how fast the training data can get regionalized. Further investigations on these aspects have to be done in the future. Recall that this approach is not new, the combination of tree and SVM, e. g., has already been proposed by Chang et al. (2010). Our contribution was the theoretical investigation of this approach, see Dumpert and Christmann (2018) for additional details, and to show that regional aspects of the data are covered, too. However, it is not clear, whether the approach of liquidSVM via Voronoi diagrams or the approach via trees leads to better results.

The described regionalization approach offers the chance to reduce the time needed to compute the SVM, it is risk-consistent and robust. Therefore, the three principles predictability, stability, and computability are fulfilled by locally learnt SVMs.

References

Aronszajn N (1950) Theory of Reproducing Kernels. Transactions of the American Mathematical Society 68:337–404, American Mathematical Society. DOI: 10.2307/1990404.

Aurenhammer F (1991) Voronoi Diagrams – a Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys 23(3):345–405, New York. DOI: 10.1145/116873.116880.

Bach F (2013) Sharp Analysis of Low-Rank Kernel Matrix Approximations. In: Shalev-Shwartz S, Steinwart I (eds.), Proceedings of the 26th Annual Conference on Learning Theory, PMLR, Proceedings of Machine Learning Research, Vol. 30, pp. 185–209.

Bennett KP, Blue JA (1998) A Support Vector Machine Approach to Decision Trees. In: Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence, Vol. 3, pp. 2396–2401. DOI: 10.1109/IJCNN.1998.687237.

Bennett KP, Campbell C (2000) Support Vector Machines: Hype or Hallelujah? ACM SIGKDD Explorations Newsletter 2(2):1–13, ACM, New York, NY, USA. DOI: 10.1145/380995.380999.

Berlinet A, Thomas-Agnan C (2004) Reproducing Kernel Hilbert Spaces in Probability and Statistics. Springer, New York. DOI: 10.1007/978-1-4419-9096-9.

Blanzieri E, Bryl A (2007) Instance-Based Spam Filtering Using SVM Nearest Neighbor Classifier. In: Wilson D, Sutcliffe G (eds.), Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, pp. 441–442.

Blanzieri E, Melgani F (2008) Nearest Neighbor Classification of Remote Sensing Images With the Maximal Margin Principle. IEEE Transactions on Geoscience and Remote Sensing 46(6):1804–1811. DOI: 10.1109/TGRS.2008.916090.

Boser BE, Guyon IM, Vapnik VN (1992) A Training Algorithm for Optimal Margin Classifiers. In: Proceedings of the fifth annual workshop on computational learning theory, pp. 144–152. DOI: 10.1145/130385.130401.

Bottou L, Vapnik V (1992) Local Learning Algorithms. Neural Computation 4(6):888–900. DOI: 10.1162/neco.1992.4.6.888.

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and Regression Trees. Chapman & Hall / CRC, Boca Raton. DOI: 10.1201/9781315139470.

Caponnetto A, De Vito E (2007) Optimal Rates for the Regularized Least-Squares Algorithm. Foundations of Computational Mathematics 7(3):331–368. DOI: 10.1007/s10208-006-0196-8.

Caruana R, Niculescu-Mizil A (2006) An Empirical Comparison of Supervised Learning Algorithms. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 161–168. DOI: 10.1145/1143844.1143865.

Caruana R, Karampatziakis N, Yessenalina A (2008) An Empirical Evaluation of Supervised Learning in High Dimensions. In: Proceedings of the 25th International Conference on Machine Learning, pp. 96–103. DOI: 10.1145/1390156.1390169.

Chang F, Guo CY, Lin XR, Lu CJ (2010) Tree Decomposition for Large-Scale Svm Problems. Journal of Machine Learning Research 11:2935–2972. URL: `http://jmlr.org/papers/v11/chang10b.html`.

Christmann A, Van Messem A (2008) Bouligand Derivatives and Robustness of Support Vector Machines for Regression. Journal of Machine Learning Research 9:915–936. DOI: 10.1145/1390681.1390711.

Christmann A, Steinwart I, Hubert M (2007) Robust Learning From Bites for Data Mining. Computational Statistics & Data Analysis 52(1):347–361. DOI: 10.1016/j.csda.2006.12.009.

Christmann A, Van Messem A, Steinwart I (2009) On Consistency and Robustness Properties of Support Vector Machines for Heavy-Tailed Distributions. Statistics and Its Interface 2(3):311–327. DOI: 10.4310/SII.2009.v2.n3.a5.

Christmann A, Dumpert F, Xiang DH (2016) On Extension Theorems and Their Connection to Universal Consistency in Machine Learning. Analysis and Applications 14(6):795–808. DOI: 10.1142/S0219530516400029.

Christmann A, Xiang D, Zhou DX (2018) Total Stability of Kernel Methods. Neurocomputing 289:101–118. ISSN: 0925-2312, DOI: 10.1016/j.neucom.2018.02.009.

Claeskens G, Croux C, Kerckhoven JV (2008) An Information Criterion for Variable Selection in Support Vector Machines. Journal of Machine Learning Research 9:541–558. DOI: 1390681.1390699.

Collobert R, Bengio S, Bengio Y (2002) A Parallel Mixture of SVMs for Very Large Scale Problems. In: Dietterich TG, Becker S, Ghahramani Z (eds.), Advances in Neural Information Processing Systems 14. pp. 633–640. DOI: 10.1162/089976602753633402.

Cortes C, Vapnik V (1995) Support-Vector Networks. Machine Learning 20(3):273–297, Springer. DOI: 10.1023/A:1022627411411.

Cristianini N, Shawe-Taylor J (2000) An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge.

Cucker F, Zhou DX (2007) Learning Theory: An Approximation Theory Viewpoint. Cambridge University Press, Cambridge.

Dekel O, Shalev-Shwartz S, Singer Y (2005) Smooth ε-Insensitive Regression by Loss Symmetrization. Journal of Machine Learning Research 6:711–741. URL: `https://www.jmlr.org/papers/volume6/dekel05a/dekel05a.pdf`.

Devroye L (1982) Any Discrimination Rule Can Have an Arbitrarily Bad Probability of Error for Finite Sample Size. IEEE Transactions on Pattern Analysis and Machine Intelligence 4(2):154–157. DOI: 10.1109/TPAMI.1982.4767222.

Duchi JC, Jordan MI, Wainwright MJ, Zhang Y (2014) Optimality Guarantees for Distributed Statistical Estimation. URL: `https://arxiv.org/abs/1405.0782`.

Dumpert F (2020a) Quantitative Robustness of Localized Support Vector Machines. Communications on Pure & Applied Analysis 19(8):3947–3956. DOI: 10.3934/cpaa.2020174.

Dumpert F (2020b) Statistische Eigenschaften Lokalisierter Maschineller Lernverfahren. PhD thesis, Bayreuth (Germany). URL: `https://epub.uni-bayreuth.de/4600/`.

Dumpert F, Christmann A (2018) Universal Consistency and Robustness of Localized Support Vector Machines. Neurocomputing 315:96–106. ISSN: 0925-2312, DOI: 10.1016/j.neucom.2018.06.061.

Eberts M (2015) Adaptive Rates for Support Vector Machines. Shaker, Aachen. ISBN: 978-3-844035-80-3.

Fernández-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Journal of Machine Learning Research 15:3133–3181. URL: `http://jmlr.org/papers/v15/delgado14a.html`.

Guo ZC, Lin SB, Zhou DX (2017a) Learning Theory of Distributed Spectral Algorithms. Inverse Problems 33(7):1–29, IOP Publishing, Bristol. ISSN: 0266-5611, DOI: 10.1088/1361-6420/aa72b2.

Guo ZC, Ying Y, Zhou DX (2017b) Online Regularized Learning With Pairwise Loss Functions. Advances in Computational Mathematics 43(1):127–150, Springer US, New York, NY. ISSN: 101-9-7168; -15-7, DOI: 10.1007/s10444-016-9479-7.

Guo ZC, Hu T, Shi L (2018) Gradient Descent for Robust Kernel-Based Regression. Inverse Problems 34(6):1–29, IOP Publishing. DOI: 10.1088/1361-6420/aabe55.

Guyon I, Elisseeff A (2003) An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3:1157–1182. URL: `https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf`.

Hable R (2012) Asymptotic Normality of Support Vector Machine Variants and Other Regularized Kernel Methods. Journal of Multivariate Analysis 106:92–117. DOI: 10.1016/j.jmva.2011.11.004.

Hable R (2013) Universal Consistency of Localized Versions of Regularized Kernel Methods. Journal of Machine Learning Research 14:153–186. URL: `https://www.jmlr.org/papers/volume14/hable13a/hable13a.pdf`.

Hable R, Christmann A (2011) On Qualitative Robustness of Support Vector Machines. Journal of Multivariate Analysis 102(6):993–1007. DOI: 10.1016/j.jmva.2011.01.009.

Hable R, Christmann A (2013) Robustness Versus Consistency in Ill-Posed Classification and Regression Problems. In: Giusti A, Ritter G, Vichi M (eds.), Classification and Data Mining. Springer, Berlin, pp. 27–35. DOI: 10.1007/978-3-642-28894-4_4.

Hampel FR (1971) A General Qualitative Definition of Robustness. The Annals of Mathematical Statistics 42:1887–1896. DOI: 10.1214/aoms/1177693054.

Hermes L, Buhmann JM (2000) Feature Selection for Support Vector Machines. In: Sanfeliu A, Villanueva J, Vanrell M, Alqukzar R, Crowley J, Shirai Y (eds.), Proceedings of the 15th International Conference on Pattern Recognition 2000, Vol. 2, pp. 712–715. DOI: 10.1109/ICPR.2000.906174.

Kotsiantis S (2007) Supervised Machine Learning: A Review of Classification Techniques. Informatica (Ljubljana) 31.

Lin J, Rosasco L, Zhou DX (2016) Iterative Regularization for Learning with Convex Loss Functions. Journal of Machine Learning Research 17(77):1–38. URL: `http://jmlr.org/papers/v17/15-115.html`.

Ma Y, Guo G (2014) Support Vector Machines Applications. Springer, New York.

Meister M, Steinwart I (2016) Optimal Learning Rates for Localized SVMs. Journal of Machine Learning Research 17(194):1–44. URL: `http://jmlr.org/papers/v17/14-023.html`.

Micchelli CA, Pontil M (2005) On Learning Vector-Valued Functions. Neural computation 17(1):177–204. DOI: 10.1162/0899766052530802.

Mücke N (2017) Direct and Inverse Problems in Machine Learning. PhD thesis, Universität Potsdam.

Mücke N (2019) Reducing Training Time by Efficient Localized Kernel Regression. In: Chaudhuri K, Sugiyama M (eds.), Proceedings of Machine Learning Research, Proceedings of Machine Learning Research, Vol. 89, pp. 2603–2610. URL: `http://proceedings.mlr.press/v89/muecke19a/muecke19a.pdf`.

Paulsen VI, Raghupathi M (2016) An Introduction to the Theory of Reproducing Kernel Hilbert Spaces. Cambridge University Press, Cambridge.

R Core Team (2018) R: A Language and Environment for Statistical Computing. Vienna, Austria. R Foundation for Statistical Computing, Vienna, Austria.

Rosasco L, Vito ED, Caponnetto A, Piana M, Verri A (2004) Are Loss Functions All the Same? Neural Computation 16(5):1063–1076, MIT Press. DOI: 10.1162/089976604773135104.

Schölkopf B, Smola AJ (2001) Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT press. ISBN: 02-6219-475-9.

Shawe-Taylor J, Cristianini N (2004) Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge.

Si S, Hsieh CJ, Dhillon IS (2017) Memory Efficient Kernel Approximation. Journal of Machine Learning Research 18(20):1–32. URL: `http://jmlr.org/papers/v18/15-025.html`.

Smale S, Yao Y (2006) Online Learning Algorithms. Foundations of Computational Mathematics 6(2):145–170. ISSN: 161-5-3375; -16-1, DOI: 10.1007/s10208-004-0160-z.

Steinwart I (2007) How to Compare Different Loss Functions and Their Risks. Constructive Approximation 26(2):225–287, Springer. DOI: 10.1007/s00365-006-0662-3.

Steinwart I, Christmann A (2008) Support Vector Machines. Springer, New York. DOI: 10.1007/978-0-387-77242-4.

Steinwart I, Thomann P (2017) liquidSVM: A Fast and Versatile SVM package. URL: `https://arxiv.org/abs/1702.06899`.

Steinwart I, Hush D, Scovel C (2009) Learning From Dependent Observations. Journal of Multivariate Analysis 100(1):175–194. ISSN: 0047-259X, DOI: 10.1016/j.jmva.2008.04.001.

Strohriegl K (2018) On Robustness and Consistency of Support Vector Machines for Non-I.I.D. Observations. PhD thesis, Bayreuth. URL: `https://epub.uni-bayreuth.de/3773/`.

Strohriegl K, Hable R (2016) Qualitative Robustness of Estimators on Stochastic Processes. Metrika 79(8):895–917. DOI: 10.1007/s00184-016-0582-z.

Therneau T, Atkinson B (2018) Rpart: Recursive Partitioning and Regression Trees. R package version 4.1-13.

Thomann P, Blaschzyk I, Meister M, Steinwart I (2017) Spatial Decompositions for Large Scale SVMs. In: Singh A, Zhu J (eds.), Proceedings of Machine Learning Research: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics 2017, Vol. 54, pp. 1329–1337. URL: `http://proceedings.mlr.press/v54/thomann17a/thomann17a.pdf`.

Van Messem A, Christmann A (2010) A Review on Consistency and Robustness Properties of Support Vector Machines for Heavy-Tailed Distributions. Advances in Data Analysis and Classification 4(2):199–220. ISSN: 1862-5355, DOI: 10.1007/s11634-010-0067-2.

Vapnik V, Bottou L (1993) Local Algorithms for Pattern Recognition and Dependencies Estimation. Neural Computation 5(6):893–909. DOI: 10.1162/neco.1993.5.6.893.

Vapnik VN (1995) The Nature of Statistical Learning Theory. Springer, New York.

Wainberg M, Alipanahi B, Frey BJ (2016) Are Random Forests Truly the Best Classifiers? Journal of Machine Learning Research 17(110):1–5. URL: `http://jmlr.org/papers/v17/15-374.html`.

Wendland H (1995) Piecewise Polynomial, Positive Definite and Compactly Supported Radial Basis Functions of Minimal Degree. Adv. Comput. Math. 4(1):389–396. DOI: 10.1007/BF02123482.

Wendland H (2005) Scattered Data Approximation. Cambridge University Press, Cambridge.

Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V (2001) Feature Selection for SvmS. In: Leen T, Dietterich T, Tresp. V (eds.), Advances in Neural Information Processing Systems 13, pp. 668–674. ISBN: 978-0-262122-41-2.

Wu D, Bennett KP, Cristianini N, Shawe-Taylor J (1999) Large Margin Trees for Induction and Transduction. In: Bratko I, Dzeroski S (eds.), Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '99, pp. 474–483. ISBN: 15-5860-612-2, DOI: 10.5555/645528.

Wu Z (1995) Compactly Supported Positive Definite Radial Functions. Advances in Computational Mathematics 4(1):283–292. DOI: 10.1007/BF03177517.

Ying Y, Pontil M (2008) Online Radient Descent Learning Algorithms. Foundations of Computational Mathematics 8(5):561–596. ISSN: 161-5-3375; -16-1, DOI: 10.1007/s10208-006-0237-y.

Ying Y, Zhou DX (2006) Online Regularized Classification Algorithms. IEEE Transactions on Information Theory 52(11):4775–4788. ISSN: 0018-9448, DOI: 10.1109/TIT.2006.883632.

Yu B (2013) Stability. Bernoulli 19(4):1484–1500, Bernoulli Society for Mathematical Statistics and Probability. DOI: 10.3150/13-BEJSP14.

Yu B, Kumbier K (2020) Veridical Data Science. Proceedings of the National Academy of Sciences 117(8):3920–3929. ISSN: 0027-8424, DOI: 10.1073/pnas.1901326117.

Zakai A, Ritov Y (2009) Consistency and Localizability. Journal of Machine Learning Research 10(30):827–856. URL: `http://jmlr.org/papers/v10/zakai09a.html`.

Zhang X, Wu Y, Wang L, Li R (2016) Variable Selection for Support Vector Machines in Moderately High Dimensions. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 78(1):53–76. DOI: 10.1111/rssb.12100.

Zhou Z, Chawla NV, Jin Y, Williams GJ (2014) Big Data Opportunities and Challenges: Discussions from Data Analytics Perspectives. IEEE Computational Intelligence Magazine 9(4):62–74. ISSN: 1556-603X, DOI: 10.1109/MCI.2014.2350953.