Spatial Decompositions for Large Scale SVMs

Philipp Thomann
Univeristy of Stuttgart

Ingrid Blaschzyk
University of Stuttgart

Mona Meister
Robert Bosch GmbH

Ingo Steinwart
University of Stuttgart

Abstract

Although support vector machines (SVMs) are theoretically well understood, their underlying optimization problem becomes very expensive if, for example, hundreds of thousands of samples and a non-linear kernel are considered. Several approaches have been proposed in the past to address this serious limitation. In this work we investigate a decomposition strategy that learns on small, spatially defined data chunks. Our contributions are two fold: On the theoretical side we establish an oracle inequality for the overall learning method using the hinge loss, and show that the resulting rates match those known for SVMs solving the complete optimization problem with Gaussian kernels. On the practical side we compare our approach to learning SVMs on small, randomly chosen chunks. Here it turns out that for comparable training times our approach is significantly faster during testing and also reduces the test error in most cases significantly. Furthermore, we show that our approach easily scales up to 10 million training samples: including hyper-parameter selection using cross validation, the entire training only takes a few hours on a single machine. Finally, we report an experiment on 32 million training samples. All experiments used `liquidSVM` (Steinwart and Thomann, 2017).

1 Introduction

Kernel methods are thoroughly understood from a theoretical perspective, used in many settings, and are known to give good performance for small and medium sized training sets. Yet they suffer from their computational complexity that grows quadratically in space and at least quadratically in time. For example, storing the entire kernel matrix to avoid costly recomputations in e.g. 64GB of memory, one can consider at most 100 000 data points. In the last 15 years there has been wide research to circumvent this barrier (cf. Bottou et al. (2007) for an overview): Sequential Minimal Optimization Platt (1999) allows for caching kernel rows so that the memory barrier is lifted, parallel solvers try to leverage from recent advances in hardware design, matrix approximations Williams and Seeger (2001) replace the original $n \times n$ kernel matrix, where n is the number of samples, by a smaller approximation, the random Fourier feature method Rahimi and Recht (2008) approximates the kernel function by another kernel having an explicit, low-dimensional feature map, which, in a second step, makes it possible to solve the primal problem instead of the usually considered dual problem, see Sriperumbudur and Szabo (2015) for a recent theoretical investigation establishing optimal rates for this approximation. Moreover, iterative strategies Rosasco and Villa (2015); Lin et al. (2015) modify the underlying regularization approach, e.g. by controlled early stopping. Finally, various decomposition strategies have been proposed, which, in a nutshell, solve many small rather than one large optimization problem.

In this work we also focus on a data decomposition strategy. Such strategies have been investigated at least since Bottou and Vapnik (1992); Vapnik and Bottou (1993). The most simple such strategy, called *random chunks*, splits the data by random into chunks of some given size, train for each chunk, and finally average the decision functions obtained on every chunk to one final decision function. Recently, this approach has been investigated theoretically in Zhang et al. (2015).

Obviously, the drawback with this approach is that the test sample has to be evaluated on every chunk: for example, if every chunk uses 80% of its training samples as support vectors, then the test sample has to be evaluated using 80% of the entire training set. From this perspective, a more interesting strategy is to decompose the data set into spatially defined cells, since in this case every test sample has only to be evaluated using the cell it belongs to. In our example above, this

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

amounts to 80% of the cell size, instead of 80% of the entire training set size. Clearly, the difference of both costs can be very significant if, for example the cell size ranges in between say a few thousands, but the training set contains millions of examples. The natural next question is of course, whether and by how much one suffers from this approach in terms of test accuracy.

Spatial decompositions for SVMs can be obtained in many different ways, e.g., by clusters Cheng et al. (2007, 2010), decision trees Bennett and Blue (1998); Wu et al. (1999); Chang et al. (2010), and k-nearest neighbors Zhang et al. (2006). Most of these strategies are investigated experimentally, yet only few theoretical results are known: Hable (2013) proves universal consistency of localized versions of SVMs. Oracle inequalities and optimal learning rates have been shown in Meister and Steinwart (2016) for least squares SVMs that are trained on disjunct spatially defined cells.

In the theoretical part of this work we expand the results in Meister and Steinwart (2016) to the hinge loss. Besides the obviously different treatment of the approximation error, the main difference in our work is that the least squares loss allows to use an optimal variance bound, whereas for the hinge loss such an optimal variance bound is only available for distributions satisfying the best version of Tsybakov's noise condition, see (2). In general, however, only a weaker variance bound is possible, which in turn makes the technical treatment harder and, surprisingly, the conditions on the cell radii that guarantee the best known rates, more restrictive.

In the experimental part we use `liquidSVM` (Steinwart and Thomann, 2017) to train local SVMs on some well-known data sets to demonstrate that they provide an efficient way to tackle large-scale problems with millions of samples. As we use full 5-fold cross validation on a 10×10 hyper-parameter grid this shows that SVMs promise to achieve fully automatic machine learning. It takes only 2-6 hours to train on data sets with 10 millions of samples and 28 features, using only 10-30GB of memory and a single computer. Finally, we distributed our software onto 11 machines to attack a data set with 30 million samples and 631 features.

In Section 2 we give a description of local SVMs. In Section 3 we define them for the hinge loss and Gaussian kernels and state the theoretical results. To motivate this we give in Section 4 a toy example. In Section 5 we describe the extensive experiments we performed. The proofs and more details are in the supplement.

2 The local SVM approach

In this section we briefly describe the local SVM approach. To this end, let $D := ((x_1, y_1), \ldots, (x_n, y_n))$ be a data set of length n, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Local SVMs construct a function f_D by solving SVMs on many spatially defined small chunks. To be more precise, let $(A_j)_{j=1,\ldots,m}$ be an arbitrary partition of the input space X. We define for every $j \in \{1, \ldots, m\}$ the local data set D_j by

$$D_j := \{(x, y) \in D : x \in A_j\}.$$

Then, one learns an individual SVM on *each* cell by solving for a regularization parameter $\lambda > 0$ the optimization problem

$$f_{D_j,\lambda} = \operatorname*{arg\,min}_{f \in H_j} \lambda \|f\|_{H_j}^2 + \frac{1}{n} \sum_{x_i, y_i \in D_j} L(y_i, f(x_i))$$

for every $j \in \{1, \ldots, m\}$, where H_j is a reproducing kernel Hilbert space over A_j with reproducing kernel $k_j : A_j \times A_j \to \mathbb{R}$, see Steinwart and Christmann (2008, Chapter 4), and where $L : Y \times \mathbb{R} \to [0, \infty)$ is a function describing our learning goal. The final decision function $f_{D,\lambda} : X \to \mathbb{R}$ is then defined by

$$f_{D,\lambda}(x) := \sum_{j=1}^{m} \mathbf{1}_{A_j}(x) f_{D_j,\lambda}(x).$$

To illustrate the advantages of this approach, let us assume that the size of the data sets D_j is nearly the same for all $j \in \{1, \ldots, m\}$, that means $|D_j| \approx n/m$. For example, if the data is uniformly distributed, it is not hard to show that the latter assumption holds, and if the data lies uniformly on a low-dimensional manifold, the same is true, since empty cells can be completely ignored. Now, the calculation of the kernel matrices $K_j^{i,l} = k_j(x_i, x_l)$ for $x_i, y_l \in D_j$ scales as

$$O\left(m \cdot \left(\frac{n}{m}\right)^2 \cdot d\right) = O\left(\frac{n^2 \cdot d}{m}\right).$$

In comparison, for a global SVM the calculation of the kernel matrix $K^{i,l} = k(x_i, x_l)$ scales as

$$O\left(n^2 \cdot d\right),$$

such that splitting and multi-thread improve that scaling by $1/m$. Similarly, it is well-known that the time complexity of the solver is

$$O\left(m \cdot \left(\frac{n}{m}\right)^{1+a}\right) = O\left(n \cdot \left(\frac{n}{m}\right)^{a}\right), \qquad (1)$$

where $a \in [1, 2]$ is a constant, and the test time scales as the kernel calculation (see Table 6 in the supplement for experimental corroboration and Steinwart et al. (2011, Theorem 6) for theoretical bounds). In all three phases we thus see a clear improvement over a globally trained SVM. Moreover, while SVMs trained on random chunks have the same complexities for the kernel matrix and the solver, they only have the bad complexity of the global approach during testing.

3 Oracle inequality and learning rates for local SVMs with hinge loss

The aim of this section is to theoretically investigate the local SVM approach for binary classification. To this end, we define for a measurable function $L : Y \times \mathbb{R} \to [0, \infty)$, called loss function, the L-risk of a measurable function $f : X \to \mathbb{R}$ by

$$\mathcal{R}_{L,\mathrm{P}}(f) = \int_{X\times Y} L(y, f(x))\, d\mathrm{P}(x, y).$$

Moreover, we define the optimal L-risk, called the Bayes risk with respect to P and L, by

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

and call a function $f^*_{L,\mathrm{P}} : X \to \mathbb{R}$, attaining the infimum, Bayes decision function. Given a data set $D := ((x_1, y_1), \ldots, (x_n, y_n))$ sampled i.i.d. from a probability measure P on $X \times Y$, where $X \subset \mathbb{R}^d$ and $Y := \{-1, 1\}$, the learning target in binary classification is to find a decision function $f_D \colon X \to \mathbb{R}$ such that $\operatorname{sign} f_D(x) = y$ for new data (x, y) with high probability. A loss function describing our learning goal is the classification loss $L_{\text{class}} : Y \times \mathbb{R} \to [0, \infty)$, defined by

$$L_{\text{class}}(y, t) := \mathbf{1}_{(0,\infty]}(y \cdot \operatorname{sign} t),$$

where $\operatorname{sign} 0 := 1$. Another possible loss function is the hinge loss $L_{\text{hinge}} : Y \times \mathbb{R} \to [0, \infty)$, defined by

$$L_{\text{hinge}}(y, t) := \max\{0, 1 - yt\}$$

for $y = \pm 1$, $t \in \mathbb{R}$, which is even convex. Since a well-known result by Zhang, e.g. Steinwart and Christmann (2008, Theorem 2.31), shows that

$$\mathcal{R}_{L_{\text{class}},P}(f) - \mathcal{R}^*_{L_{\text{class}},P} \le \mathcal{R}_{L_{\text{hinge}},P}(f) - \mathcal{R}^*_{L_{\text{hinge}},P}$$

for all functions $f : X \to \mathbb{R}$, we consider in the following the hinge loss in our theory and write $L := L_{\text{hinge}}$. We remark that for the hinge loss it suffices to consider the risk for function values restricted to the interval $[-1, 1]$, since this does not worsen the loss and thus the risk. Therefore, we define by

$$\widehat{t} := \max\{-1, \min\{t, 1\}\}$$

for $t \in \mathbb{R}$ the clipping operator, which restricts values of t to $[-1, 1]$, see Steinwart and Christmann (2008, Chapter 2.2). In order to derive a bound on the excess risks $\mathcal{R}_{L,P}(\widehat{f}_{D,\lambda}) - \mathcal{R}^*_{L,P}$, and in order to derive learning rates, we recall some notions from Steinwart and Christmann (2008, Chapter 8), which describe the behaviour of P in the vicinity of the decision boundary. To this end, let $\eta \colon X \to [0, 1]$ be a version of the posterior probability of P, that means that the probability measures $P(\cdot\,|x)$, defined by $P(y = 1|x) =: \eta(x)$, $x \in X$, form a regular conditional probability of P. We write

$$X_1 := \{\, x \in X \colon \eta(x) > 1/2 \,\},$$
$$X_{-1} := \{\, x \in X \colon \eta(x) < 1/2 \,\}.$$

Then, we call the function $\Delta_\eta \colon X \to [0, \infty)$, defined by

$$\Delta_\eta(x) := \begin{cases} \operatorname{dist}(x, X_1) & \text{if } x \in X_{-1}, \\ \operatorname{dist}(x, X_{-1}) & \text{if } x \in X_1, \\ 0 & \text{otherwise,} \end{cases}$$

distance to the decision boundary, where $\operatorname{dist}(x, A) := \inf_{x' \in A} \|x - x'\|_2$. Thus we can describe the mass of the marginal distribution P_X of P around the decision boundary by the following exponents: We say that P has margin-noise exponent (MNE) $\beta \in (0, \infty)$ if there exists a constant $c_{\text{MNE}} \ge 1$ such that

$$\int_{\{\Delta_\eta(x) < t\}} |2\eta(x) - 1|\, dP_X(x) \le (c_{\text{MNE}} t)^\beta$$

for all $t > 0$. That is, the MNE β measures the mass and the noise, i.e. points $x \in X$ with $\eta(x) \approx 1/2$, around the decision boundary. Hence, we have a large MNE if we have low mass and/or high noise around the decision boundary. Furthermore, we say that P has noise exponent (NE) $q \in [0, \infty]$ if there exists a constant $c_{\text{NE}} > 0$ such that

$$P_X(\{x \in X : |2\eta(x) - 1| < \varepsilon\}) \le (c\varepsilon)^q \qquad (2)$$

for all $\varepsilon > 0$. Thus, the NE q, which corresponds to Tsybakov's noise condition, introduced in Tsybakov (2004), measures the amount of noise, but does not locate it. For examples of typical values of these exponents and relations between them we refer the reader to Steinwart and Christmann (2008, Chapter 8).

For the theory of local SVMs it is necessary to specify the partition. Hence, we assume in the following that $X \subset [-1, 1]^d$ and define for a set $T \subset X$ its radius by

$$r_T = \inf\{\varepsilon > 0 : \exists\, s \in T \text{ such that } T \subset B_2(s, \varepsilon)\},$$

where $B_2(s, \varepsilon) := \{t \in T : \|t - s\|_2 \le \varepsilon\}$ with Euclidean norm $\|\cdot\|_2$ in $\mathbb{R}^d$. In the following let $(A_j)_{j=1,\ldots,m}$ be a partition of X such that all its cells have non-empty interior, that is $\mathring{A}_j \neq \emptyset$ for every $j \in \{1, \ldots, m\}$, and such that for $r > 0$ we have

$$r_{A_j} < r \le 16 m^{-\frac{1}{d}}. \qquad (3)$$

A simple partition that fulfills the latter condition is for example the partition by cubes with a specific side length. We refer the reader to Section 4 for the computation of another type of partition satisfying (3).

In the following we restrict ourselves to local SVMs with Gaussian kernels. For this purpose, we denote for every $j \in \{1,\dots,m\}$ by $H_{\gamma_j}(A_j)$ the RKHS over A_j with Gaussian kernel $k_{\gamma_j} : A_j \times A_j \to \mathbb{R}$, defined by

$$k_{\gamma_j}(x,x') := \exp\left(-\gamma_j^{-2}\|x-x'\|_2^2\right)$$

for some width $\gamma_j > 0$. Furthermore, we define for $f \in H_{\gamma_j}(A_j)$ the function $\hat{f} : X \to \mathbb{R}$ by

$$\hat{f}(x) := \begin{cases} f(x)\,, & x \in A_j\,, \\ 0\,, & x \notin A_j\,. \end{cases}$$

Then, according to Eberts and Steinwart (2015, Lemma 2) the space $\hat{H}_{\gamma_j} := \{\hat{f} : f \in H_{\gamma_j}(A_j)\}$ equipped with the norm

$$\|\hat{f}\|_{\hat{H}_{\gamma_j}} := \|f\|_{H_{\gamma_j}(A_j)}$$

is an RKHS over X. Thus, local SVMs for Gaussian kernels solve the optimization problem

$$f_{D_j,\lambda_j,\gamma_j} = \operatorname*{arg\,min}_{\hat{f}\in\hat{H}_{\gamma_j}} \lambda_j\|\hat{f}\|_{\hat{H}_{\gamma_j}}^2 + \frac{1}{n}\sum_{x_i,y_i\in D_j} L(y_i,f(x_i)),$$

for every $j \in \{1,\dots,m\}$, where $\lambda_j,\gamma_j > 0$. Then, for the vectors $\boldsymbol{\gamma} := (\gamma_1,\dots,\gamma_m)$ and $\boldsymbol{\lambda} := (\lambda_1,\dots,\lambda_m)$ the decision function $f_{D,\boldsymbol{\lambda},\boldsymbol{\gamma}} : X \to \mathbb{R}$ is defined by

$$f_{D,\boldsymbol{\lambda},\boldsymbol{\gamma}}(x) := \sum_{j=1}^{m} f_{D_j,\lambda_j,\gamma_j}(x). \tag{4}$$

Note that the clipped decision function $\widehat{f}_{D,\boldsymbol{\lambda},\boldsymbol{\gamma}} : X \to [-1,1]$ is then defined by the sum of the clipped empirical solutions $\widehat{f}_{D_j,\lambda_j,\gamma_j}$ since for all $x \in X$ there is exactly one $f_{D_j,\lambda_j,\gamma_j}$ with $f_{D_j,\lambda_j,\gamma_j}(x) \neq 0$. We finally introduce the following set of assumptions:

(A) Let $(A_j)_{j=1,\dots,m}$ be a partition of X and $r > 0$, such that for every $j \in \{1,\dots,m\}$ we have $\mathring{A}_j \neq \emptyset$ and (3). In addition, for every $j \in \{1,\dots,m\}$ let $H_{\gamma_j}(A_j)$ be the RKHS of the Gaussian kernel k_{γ_j} over A_j with width $\gamma_j > 0$. Furthermore, we use the notation $\gamma_{\max} := \max\{\gamma_1,\dots,\gamma_m\}$.

Now, we present an upper bound on the excess risk for the hinge loss and Gaussian kernels.

Theorem 3.1. *Let $Y = \{-1,1\}$ and let $L : Y \times \mathbb{R} \to [0,\infty)$ be the hinge loss. Let P be a distribution on $X \times Y$ with MNE $\beta \in (0,\infty)$ and NE $q \in [0,\infty]$. Moreover, let (A) be satisfied. Then, for all $p \in (0,1)$, $n \geq 1$, $\tau \geq 1$ with $\tau \leq n$, $\boldsymbol{\lambda} = (\lambda_1,\dots,\lambda_m) \in (0,\infty)^m$, $\boldsymbol{\gamma} = (\gamma_1,\dots,\gamma_m) \in (0,r]^m$ the SVM given by (4) satisfies*

$$\begin{aligned}
&\mathcal{R}_{L,P}(\widehat{f}_{D,\boldsymbol{\lambda},\boldsymbol{\gamma}}) - \mathcal{R}_{L,P}^* \\
&\leq C_{\beta,d,p,q}\left(\sum_{j=1}^{m}\lambda_j\gamma_j^{-d} + \gamma_{max}^{\beta} + \left(\frac{\tau}{n}\right)^{\frac{q+1}{q+2}}\right. \\
&\quad \left. + \left(r^{2p}\left(\sum_{j=1}^{m}\lambda_j^{-1}\gamma_j^{-\frac{d+2p}{p}}P_X(A_j)\right)^{p} n^{-1}\right)^{\frac{q+1}{q+2-p}}\right)
\end{aligned}$$

with probability P^n not less than $1-3e^{-\tau}$, where the constant $C_{\beta,d,p,q} > 0$ depends only on d,β,p and q.

The main idea for the proof is that $f_{D,\boldsymbol{\lambda},\boldsymbol{\gamma}}$ is an SVM solution for a particular RKHS. To be more precise, it is easy to show with a generalization of Eberts and Steinwart (2015, Lemma 3) that for RKHSs $\hat{H}_{\gamma_j}$ on X and a vector $\boldsymbol{\lambda} := (\lambda_1,\dots,\lambda_m) > 0$ the direct sum

$$H := \bigoplus_{j=1}^{m}\hat{H}_{\gamma_j} = \left\{f = \sum_{j=1}^{m} f_j : f_j \in \hat{H}_{\gamma_j} \text{ for all } j \in J\right\}$$

is again an RKHS if it is equipped with the norm

$$\|f\|_H^2 = \sum_{j=1}^{m}\lambda_j\|f_j\|_{\hat{H}_{\gamma_j}}^2\,.$$

Obviously, $f_{D,\boldsymbol{\lambda},\boldsymbol{\gamma}} \in H$. We remark that the latter construction actually holds for RKHSs with arbitrary kernels. For the proof it is necessary on the one hand to chose suitable RKHSs in order to bound the approximation error and on the other hand to bound the entropy numbers of the operator $\mathrm{id} : H_{\gamma_j}(A_j) \mapsto L_2(P_{X_{|A_j}})$, which describe in some sense the size of the RKHS $H_{\gamma_j}(A_j)$. Unfortunately, such bounds contain constants which depend on the cells A_j and which are in general hard to control. However, for Gaussian kernels we obtain such bounds if we restrict γ_j by r for every $j \in \{1,\dots,m\}$, which explains our restriction to Gaussian kernels. Moreover, the proof actually shows that $\gamma_j \leq r$ can be replaced by $\gamma_j \leq cr$ for a constant c, which is independent of p,m,τ,λ and γ. Furthermore, using this entropy number bound leads to a dependence of a parameter p on the right-hand side of the oracle inequality, where small p lead to an unknown behaviour in the constant $C_{\beta,d,p,q}$. This problem is well-known in the Gaussian kernel case, see Steinwart and Scovel (2007) or Eberts and Steinwart (2011), and there is not given a solution for this problem yet.

Let us assume that all cells have the same kernel parameter γ and regularization parameter λ. Then, the oracle inequality stated in Theorem 3.1 coincides up to constants and to the parameter p, which can be chosen arbitrary small, the oracle inequality for global

SVMs stated in Steinwart and Christmann (2008, Theorem 8.25) for $\tau = \infty$. Furthermore, we remark that our oracle inequality is formally similar to the inequality for local SVMs for the least square loss, see Eberts and Steinwart (2015, Theorem 7) if we assume that the Bayes decision function is contained in a Besov space with smoothness $\alpha = \beta/2$.

In the next theorem we show learning rates by choosing appropriate sequences of $r_n, \lambda_{n,j}$ and $\gamma_{n,j}$.

Theorem 3.2. *Let $\tau \geq 1$ be fixed and $\nu \in \left(0, \frac{q+1}{\beta(q+2)+d(q+1)}\right]$. Under the assumptions of Theorem 3.1 and with*

$$\begin{aligned} r_n &= c_1 n^{-\nu}, \\ \lambda_{n,j} &= c_2 r_n^d n^{-\frac{(\beta+d)(q+1)}{\beta(q+2)+d(q+1)}}, \\ \gamma_{n,j} &= c_3 n^{-\frac{(q+1)}{\beta(q+2)+d(q+1)}}, \end{aligned}$$

for every $j \in \{1, \ldots, m_n\}$, we have for all $n \geq 1$ and $\xi > 0$ that

$$\begin{aligned} &\mathcal{R}_{L,P}(\widehat{f}_{D,\boldsymbol{\lambda}_n,\boldsymbol{\gamma}_n}) - \mathcal{R}^*_{L,P} \\ &\leq C_{\beta,\nu,\xi,d,q} \tau^{\frac{q+1}{q+2}} \cdot n^{-\frac{\beta(q+1)}{\beta(q+2)+d(q+1)}+\xi} \end{aligned}$$

holds with probability P^n not less than $1 - 3e^{-\tau}$, where $\boldsymbol{\lambda}_n := (\lambda_{n,i})_{i=1,\ldots,m_n}$ as well as $\boldsymbol{\gamma}_n := (\gamma_{n,i})_{i=1,\ldots,m_n}$ and where $C_{\beta,\nu,\xi,d,q}, c_1, c_2, c_3$ are positive constants.

Note that the restriction of the parameter ν in Theorem 3.2 is set to ensure that $\sup_{n\geq 1} \gamma_{n,j}/r_n < \infty$, as we mentioned after Theorem 3.1. The learning rate stated in Theorem 3.2 coincides always with the fastest known rate which can be achieved by a global SVM, cf. Steinwart and Christmann (2008, Theorem 8.26 and (8.18)). Let us consider some special cases for the parameters β and q. In the case of "benign noise", that is $q = \infty$, our learning rate reduces to

$$n^{-\frac{\beta}{\beta+d}+\xi},$$

and is only less sensitive to the dimension d if β is large. Next, let us assume that $q = 1$ such that P has a rather moderate noise concentration and that $\beta = 2$, which means that we have additionally much mass around the decision boundary. For examples of distributions having these parameters we refer to the examples in Steinwart and Christmann (2008, Chapter 8). The chosen parameters q and β yield the rate

$$n^{-\frac{2}{3+d}+\xi}$$

and we observe that the dimension d has a high impact, which means that the rate gets worse the higher d. We refer the reader to Section 4, where we created a toy example for a distribution having these parameters β and q and where we compare this rate with the experimental one. Since the class of considered distributions contains the ones, whose marginal distribution P_X is the uniform distribution, a bad dependence on d is not surprisingly as these distributions usually among those which cause the curse of dimensionality. However, if the data lies for example on a $d_{\mathcal{M}}$-(low)-dimensional rectifiable manifold $\mathcal{M}$, we believe that one is able to improve the rate since one would learn the local SVM only on a few cells—instead of learning on $m = O(r^d)$ cells, one only has to learn on $m = O(r^{d_{\mathcal{M}}})$ cells.

Clearly, to obtain the rates in Theorem 3.2 we need to know the parameters β and q. However, such rates can also be obtained by a data-dependent parameter selection strategy without knowing the parameters. For example, Eberts and Steinwart (2015) presented for the least-square loss the training validation Voronoi partition support vector machine (TV-VP-SVM) and showed that this learning method achieves adaptively the same rates as the rates for local SVM for the least-square loss. We remark that this method can be adapted to our case, that is for the hinge loss, since a key ingredient is an oracle inequality having the structure of Eberts and Steinwart (2015, Theorem 7), which is given in our case, as we mentioned in the discussion before Theorem 3.2. For more details to parameter selection methods we refer the reader to Eberts and Steinwart (2015, Section 5) and Steinwart and Christmann (2008, Chapter 6.5 and Chapter 8.2). At this point we finally remark that the mentioned adaptive strategies obtain the rate

$$n^{-\frac{\beta(q+1)}{\beta(q+2)+d(q+1)}+\xi}$$

for all β and q satisfying $\nu \leq \frac{\beta(q+1)}{\beta(q+2)+d(q+1)}$. Consequently, for larger ν, which lead to faster training times, the range of adaptivity becomes smaller, and vice-versa.

4 Toy Example

To illustrate the theoretical results we now give a toy example by mixing two multivariate Gaussians, one for $y = 1$ and one for $y = -1$, see Figure 1. We define $X := [-2, 2]^d$ and $x_1 := (1, 0, \ldots, 0) \in X$ and $\vartheta := 0.6$. Moreover, we set for all events $A \subset X \times Y$:

$$P(A) := \vartheta \int_{A^+} \phi(x)\, dx + (1-\vartheta) \int_{A^-} \psi(x)\, dx,$$

where $A^+ := \{x \mid (x, 1) \in A\}$ and $A^- := \{x \mid (x, -1) \in A\}$. Here ϕ and ψ are the densities of the multivariate normal distribution around the origin, and x_1 resp. and of variance 1 and $\frac{1}{8}$ resp. on X. In this case $\eta(x)$ can be calculated and it is easy to see that the decision boundary is an ellipsoid, the NE is $q = 1$, and the MNE

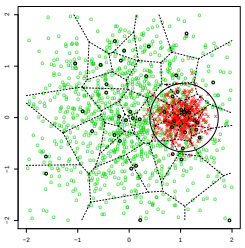

Figure 1: Toy model: Mixture of two Gaussians. The decision boundary is depicted as almost a circle and the dotted lines mark a Voronoi partition induced by the black center samples.

is $\beta = 2$. Hence, we can use Theorem 3.2 by using

$$\nu = \frac{1}{3+d}, \qquad r_n = c_1 n^{-\frac{1}{3+d}},$$
$$\lambda_n = c_2 n^{-\frac{2+d}{3+d}}, \qquad \gamma_n = c_3 n^{-\frac{1}{3+d}}.$$

to achieve the rate $n^{-\frac{2}{3+d}+\varepsilon}$. A (even faster) rate than the theoretical from Theorem 3.2.

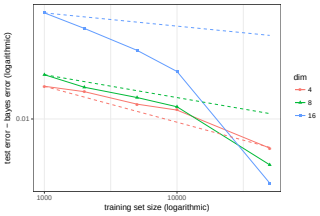

Figure 2: Obtained excess risks for the toy models at dimensions $4, 8, 16$. We decrease the maximal data radius per cell as in Theorem 3.2 (average of 20 runs). The rates from Theorem 3.2 are marked in dashed lines with normalization to match the first data point.

5 Experiments

We now use the developed methods for large scale data sets in order to understand whether the theoretical and synthetic results also transfer to real world data. We intend to demonstrate that partitioning allows for large n to give efficiently good results, but only if one uses spatial decomposition. As discussed above, global kernel methods become unfeasible for $n \geq 100\,000$, but there is already known the decomposition method of *random chunks*: split the data into samples and train on these smaller sets first, then average predictions over these samples. We will see that spatial decompositions often outperform random chunks in terms of test error—while testing is much faster. We did not consider any other method for speeding up SVMs, e.g. random fourier features, since these can be naturally combined with our spatial approach. Indeed, if such a method is faster than a global SVM on data sets of, say 20.000 and more samples, then one could also use this method on each cell of that size, which in turn would further decrease the overall training time of our approach. In this sense, our reported training times are a kind of worst-case scenario.

In Section 5.1 we will demonstrate that the spatial decomposition approach even can be used for problems with dozens of millions of samples and hundreds of dimensions, that is too big for a single machine.

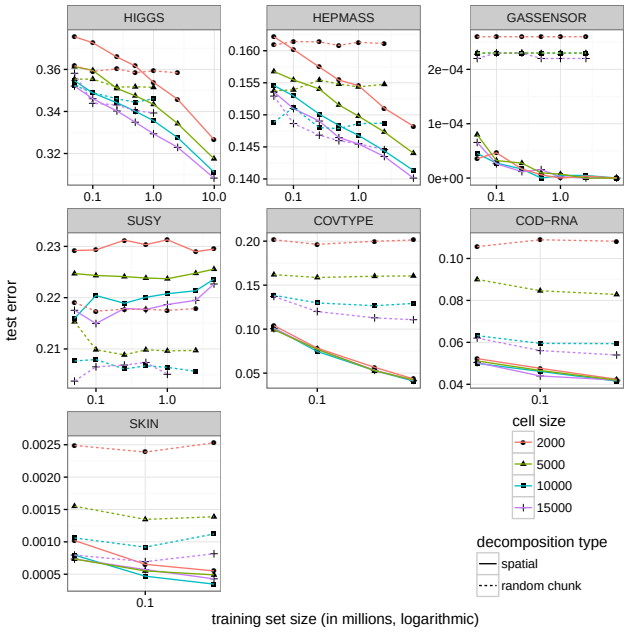

Figure 3: Test errors for large-scale data sets. The training set size is the most influential factor for spatial decompositions. Bigger cell sizes give only sometimes earlier better results. Random chunk decomposition only profits from bigger and not from more chunks.

Table 1: Overview of the data sets we used.

Name	train size	test size	dims	Source
HIGGS	9 899 999	1 100 001	28	UCI
HEPMASS	7 000 000	3 500 000	28	UCI
GASSENSOR	7 548 087	838 678	18	UCI
SUSY	4 499 999	50 0001	18	UCI
COVTYPE	464 429	116 583	54	UCI
COD-RNA	231 805	99 347	9	LIBSVM
SKIN	196 045	49 012	3	UCI

For the data sets, we looked on the UCI and LIBSVM repositories for the biggest examples, which did have firstly a suitable learning target, secondly only numerical features, and thirdly not too many dimensions since for high dimensions spatial decompositions be-

come at least debatable. Hence, we did not consider any image data sets. See Table 1 for an overview of the data sets and the train/test splits. We trained on one hand using the full training set, and on the other hand using a sample of the full training set of the 6 sizes $n = 50\,000, \ldots, 1\,000\,000, 2\,500\,000$.

Even though our theoretical results are by control of radii, in applications it is more important to control the computational cost, namely the maximal number of samples in the cells as we discussed in Section 2. Hence, for each cell size $2000, 5000, 10000, 15000$, each of the 7 training sets was first split spatially into cells of that size and a SVM was trained on each cell. This means that on each cell five-fold cross validation was performed for a 10×10 geometrically spaced hyper-parameter grid where the endpoints were scaled to accommodate the number of samples in every fold, the cell size, and the dimension. Then, for the full testing set the test error was computed by assigning each test sample to the cell it spatially belongs to and the cell's SVM was used to predict a label for it. For comparison we also performed these experiments using random chunks: Each training set was split uniformly into chunks of size $2000, 5000, 10000, 15000$. There, due to time constraints, for the bigger data sets we calculated the testing error using a test set of size 100 000.

We used `liquidSVM` (Steinwart and Thomann, 2017), our own SMO-type implementation in `C++`. As architecture, we used Intel® Xeon® CPUs (E5-2640 0 at 2.50GHz, May 2013) with Ubuntu Linux. There were two NUMA-sockets each with a CPU having 6 physical cores, but multi-threading was only used to compute the kernel matrix in training and the test error. The data-partitioning and the solver are single-threaded. Even though there were 128GB RAM per NUMA-socket available, the processes were limited to use at most 64GB to give results which could be compared on other workstations. The smaller cell sizes use considerably less memory. But this way, we restricted us to only use cell sizes up to 15 000. For random chunks in the biggest cases even that was impossible.

Table 2: Time to train the local SVM including 5-fold cross-validation on a 10×10 hyper-parameter grid.

	2000	5000	10000	15000
	training time (in min.)			
HIGGS	308	679	1358	1992
HEPMASS	145	316	624	964
GASSENSOR	90	214	421	636
SUSY	121	261	513	779
COVTYPE	9	18	39	54
COD-RNA	4	9	16	25
SKIN	2	6	10	16

Table 3: Time to predict using random chunks divided by time to predict using spatial decomposition. Since RC becomes to expensive this is using 100 000 training and test samples.

	2000	5000	10000	15000
HIGGS	74	34	15	10
HEPMASS	84	28	14	6
GASSENSOR	667	513	254	414
SUSY	88	35	19	11
COVTYPE	153	61	26	19
COD-RNA	139	52	18	6
SKIN	25	10	8	4

The results on real world data sets give a clear picture. It can be seen in Figure 3 that increasing the training set size enhances the test error in most cases dramatically. All data sets but SUSY show that by using spatial partitioning one is able to attack large-scale problems nicely. Remark that all cell sizes give almost the same error. Yet smaller cell sizes give it faster (see Table 2), only for HIGGS and HEPMASS bigger cells achieve the same test error already using fewer training data. Bigger cell sizes can give better test error, although in the trade-off time vs. error, they play not too big of a role, cf. Figure 4 in the supplement. In contrast, for the random chunks strategy the cell size plays the most crucial role for the test error and the error quickly saturates at some value. The error can only be made smaller by using bigger cell sizes, and hence much more expensive training and even more testing time, see Table 3. Finally, our spatial decomposition seems overwhelmed with SUSY and exposes the same saturation effect as in random chunk and does not achieve to beat that.

5.1 Big-data: ECBDL 2014

One of the most important aspects of spatial decompositions is that this makes the process cloud scalable, since the training and testing on the cells trivially can be assigned to different workers, once the data is split. To demonstrate this we used Apache Spark, an in-memory map/reduce framework. The data set was saved on a Hadoop distributed file system on one master and up to 11 worker machines of the above type. In a first step, the data was split into coarse cells by the following procedure. A subset of the training data was sampled and sent to the master machine where 1000-2000 centers were found and these centers were sent back to the worker machines. Now each worker machine could assign locally to every of it's samples the coarse cell in the Voronoi sense. Finally a Spark-shuffle was performed: Every cell was assigned to one of the workers and all its samples were sent to that worker. This procedure is quite standard for Spark. It had to be performed only once for the data set.

Table 4: More detailed times at fixed cell size 2000 (in seconds). The testing phase is given per 100 000 test samples. The last column gives the maximal resident size over all of training and testing (in GB).

name	time (in seconds) used in phase						RAM
	part.	kernel calc.	solver	valid.	sel	test	
HIGGS	40	2887	11862	994	1030	322	31
HEPMASS	26	2022	4235	682	560	208	22
GASSENSOR	21	2354	729	633	541	166	21
SUSY	18	1296	4419	448	374	141	13
COVTYPE	3	138	206	44	47	13	3
COD-RNA	1	69	76	20	21	7	1
SKIN	1	53	21	18	11	3	1

In the second step every such coarse cell–now being on one physical machine–was used for training by our `C++` implementation discussed above: this in particular means that each coarse cell was again split into fine cells of some specified size and then the TV-VP-SVM method was used. For these experiments we used a 20×20 hyper-parameter grid. Obviously this now was done in parallel on all worker nodes. The test set was also split into the coarse cells and then by our implementation further into fine cells for prediction.

To give an experimental evaluation we used the classification data set introduced in the *Evolutionary Computation for Big Data and Big Learning Workshop*[1] (ECBDL). The data set considers contact points of polymers. It has 631 dimensions and 32 million training (60GB disk space) and 2.9 million test samples (5.1GB disk space). One of the major challenges with this is a rather strong imbalance in the labels–there are 98% of negative samples. Therefore the competition used a scoring of "TruePositiveRate · TrueNegativeRate".

Certainly this data set is not ideally suited for local SVMs: on one hand it certainly has higher dimensions than we would hope. On the other hand the imbalance in the data has to be treated by hand. For this we used the hinge loss with weight 0.987 after trying out different weights on a validation sample of size 100 000. For the training calculations, the use of the full 128GB memory per socket allowed us to use (fine) cell sizes up to 100 000. The splitting was not optimized and took about an hour. Training and testing took 32.2 hours on our 11 worker machines, see Table 5.

Our off-the-cuff scores range from 0.422 to 0.456. That would have landed us in the middle of the scores at the beginning of the competition. Of the seven teams three had their first submissions between 0.3 and 0.42 and at some point found a way to boost it to 0.45 or more. One of these, the team HyperEns, used *standard and budgeted SVMs with bayesian optimisation of parameters* and started with submissions scoring around 0.34–0.38 and after ten days found a way to score over 0.45 and achieving in the end 0.489 using *4.7 days of parameter optimisation in a 16-core machine*. The three best teams scored from start over 0.47. The winning team Efdamis used feature weighting and random forests and its best model took *39h of Wall-clock time in a 144-core cluster (not used exclusively)* which is comparable to the training time of our best model.

Generally, we suppose that in such competitions the best results are achieved by careful hand-optimisation of the features and the hyper-parameters. We on the other hand aspire to realize fully automatic learning and hence are not aiming to beat such results.

Table 5: Results for the ECBDL'14 data set. Increasing the cell size achieved to increase the score = TPR·TNR.

cell size	time	Score	TPR	TNR	work nodes
10 000	4.8h	0.422	0.656	0.643	7
15 000	7.2h	0.433	0.664	0.652	7
20 000	9.3h	0.438	0.664	0.660	7
50 000	14.0h	0.453	0.666	0.679	11
100 000	32.2h	0.456	0.667	0.680	11

6 Conclusion

The experiments show that on commodity hardware SVMs can be used to train with millions of samples achieving at least decent errors in a few hours–even including 5-fold cross validation on a 10×10 hyper-parameter grid. The results for ECBDL demonstrate that local SVMs scale trivially across clusters. Hence, cloud scaling of fully automatic state-of the art SVMs is possible. Together with the statistical guarantees in Theorem 3.2 it is clear that local SVMs provide a reliable and broadly usable machine learning system for large scale data.

Acknowledgments

We thank the Institute of Mathematics at the University of Zurich for the computational resources.

[1] See `http://cruncher.ncl.ac.uk/bdcomp/` and `http://cruncher.ncl.ac.uk/bdcomp/BDCOMP-final.pdf`.

References

K. P. Bennett and J. A. Blue. A support vector machine approach to decision trees. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 3, pages 2396–2401, 1998.

L. Bottou and V. Vapnik. Local Learning Algorithms. *Neural Computation*, 4:888–900, 1992.

L. Bottou, O. Chapelle, D. DeCoste, and J. Weston. *Large-Scale Kernel Machines*. MIT Press, Cambridge, MA, 2007.

B. Carl and I. Stephani. *Entropy, Compactness and the Approximation of Operators*. Cambridge University Press, Cambridge, 1990.

F. Chang, C.-Y. Guo, X.-R. Lin, and C.-J. Lu. Tree Decomposition for Large-Scale SVM Problems. *J. Mach. Learn. Res.*, 11:2935–2972, 2010.

H. Cheng, P.-N. Tan, and R. Jin. Localized Support Vector Machine and Its Efficient Algorithm. In *SIAM International Conference on Data Mining*, 2007.

H. Cheng, P.-N. Tan, and R. Jin. Efficient Algorithm for Localized Support Vector Machine. *IEEE Transactions on Knowledge and Data Engineering*, 22: 537–549, 2010.

M. Eberts and I. Steinwart. Optimal learning rates for least squares SVMs using Gaussian kernels. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1539–1547, 2011.

M. Eberts and I. Steinwart. Optimal learning rates for localized SVMs. Technical report, `http://arxiv.org/pdf/1507.06615v1`, 2015.

R. Hable. Universal consistency of localized versions of regularized kernel methods. *J. Mach. Learn. Res.*, 14:153–186, 2013.

J. Lin, L. Rosasco, and D.-X. Zhou. Iterative regularization for learning with convex loss functions. *arXiv:1503.08985*, 2015.

M. Meister and I. Steinwart. Optimal learning rates for localized SVMs. *J. Mach. Learn. Res.*, 17:1–44, 2016.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods–Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. 2008.

L. Rosasco and S. Villa. Learning with incremental iterative regularization. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1621–1629. 2015.

B. Sriperumbudur and Z. Szabo. Optimal rates for random fourier features. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1144–1152. 2015.

I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, New York, 2008.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. *Ann. Statist.*, 35: 575–607, 2007.

I. Steinwart and P. Thomann. liquidSVM: A fast and versatile SVM package. *ArXiv e-prints 1702.06899*, feb 2017. URL `http://www.isa.uni-stuttgart.de/software`.

I. Steinwart, D. Hush, and C. Scovel. Training SVMs without offset. *J. Mach. Learn. Res.*, 12:141–202, 2011.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Ann. Statist.*, 32:135–166, 2004.

V. Vapnik and L. Bottou. Local Algorithms for Pattern Recognition and Dependencies Estimation. *Neural Computation*, 5:893–909, 1993.

C K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

D. Wu, K. P. Bennett, N. Cristianini, and J. Shawe-Taylor. Large Margin Trees for Induction and Transduction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 474–483, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2126–2136, 2006.

Y. Zhang, J. Duchi, and M. Wainwright. Divide and Conquer Kernel Ridge Regression: A Distributed Algorithm with Minimax Optimal Rates. *J. Mach. Learn. Res.*, 16:3299–3340, 2015.