Speeding up Support Vector Machines with Random Fourier Features

Yauheni Selivonchyk[1]

University of Bonn, Bonn, NRW, Germany
s6yaseli@uni-bonn.de

Abstract

This paper describes advantages and disadvantages of using Random Fourier Features with Support Vector Machines (SVMs). SVMs are a widely used example of kernel methods that is known to perform well in practice. Unfortunately, running time of a general nonlinear SVM algorithm increases significantly with the size of the training dataset. When an explicit finite-dimensional mapping corresponding to the kernel function can be constructed, a much faster linear SVM algorithm can be used. Rahimi and Recht proposed a technique called "Random Fourier Features" that allows to construct a random mapping into a relatively low-dimensional feature space for a large family of kernel functions. In this paper we show, that linear SVM using Random Fourier Features allow to significantly increase runtime of nonlinear SVM in a predictable manner. We analyze the trade-off between computational complexity and the accuracy of the resulting solutions.

1 Introduction

In this paper we study the qualities of Random Fourier Features (RFF) [16] in conjunction with Support Vector Machines (SVMs). SVMs are a popular tool that generalizes well on yet unseen inputs given sufficient amount of training data [21]. General algorithms for nonlinear SVMs require $\mathcal{O}(N^2)$ training time, where N is the number of training examples, and prediction function uses as many as $\mathcal{O}(N)$ operations [5]. This might require a long time for training on large datasets. Additionally, prediction function might require access to the complete set of the training data. Constructing a random mapping as described by [16] would allow to apply a faster linear SVM algorithm that has significantly better training time complexity and resulting prediction function requires time, independent of the size of the training data [8].

The original algorithm for SVMs was first proposed by V.Vapnik and A.Chervonenkis in 1964 [22, 20]. The algorithm solves a classification problem for the linearly separable data. Algorithm finds a separating hyperplane that has the maximum distance from the closest input points. The hyperplane, if exists, is called "maximum-margin hyperplane". Authors suggested, that such a choice of the hyperplane would likely generalize better on yet unseen data [19]. The problem of selecting maximum-margin hyperplane can be expressed as an empirical risk minimization problem in the following primal form [1]:

$$\min_{\omega,b} \sum_{i=1}^{n} L(y_i, y_i^*) + \lambda ||w||^2 \tag{1}$$

where $L(y, y^*)$ is the hinge loss function, y_i and y_i^* are target and predicted values for input x_i, and regularization parameter $\lambda \in (0, \infty)$. According to Joachims [8] the solution of this classification problem can be found in $\mathcal{O}(Nd)$ time, given accuracy parameter ξ [8], where d is dimensionality of the input space, N is the number of training examples. The evaluation relies on a single product operation $x'w$ and therefore can be performed in $\mathcal{O}(d)$ time. A more

general SVM algorithm for non-linear data was proposed by Boser et al. [4] and relies on the kernel trick.

A kernel is a function $k : x, y \to \mathbb{R} | x, y \epsilon \mathbb{R}^d$ that equals to the inner product of inputs mapped into some Hilbert space [13], i.e.:

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle \tag{2}$$

As long as the learning technique relies only on the linear product of inputs, the underlying mapping $\Phi(x)$ does not need to be explicitly calculated and can, in fact, be unknown.

Boser et al. [4] proposed an algorithm that maximizes the margin between the training pattern and the decision boundary for problems that have a dual kernel representation. In particular, the dual kernel representation of the problem (1) is as follows:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - 1/2 \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{3}$$

where $\alpha_i > 0$ for all i and $\sum_{i=1}^{n} \alpha_i y_i = 0$. Solution of this problem can be found in $\mathcal{O}(N^2)$ time [5]. Additionally, calculation of the decision function $\sum_{i=1}^{n} y_i \alpha_i k(x_i, x) + b | \alpha_i \neq 0$ in the worst case requires $\mathcal{O}(N)$ kernel function calculations, since it is possible that $a_i > 0 | \forall i$.

Given asymptotic worst case complexity as above, we might prefer linear SVMs when working with large datasets. Luckily, approximation techniques have been developed that allow to achieve alike computational time complexity for nonlinear SVMs at a cost of accuracy [23, 9, 16]. In particular, in this paper we are going to analyze the potential of using RFF [16]. Authors of the original publication "Random Features for Large-Scale Kernel Machines" provide a way to construct a random relatively low-dimensional mapping $z(x)$ similar to $\Phi(x)$ for a large family of kernel functions. Resulting mapping can be used with linear SVM.

In section 2 we overview methods, allowing to find an approximate solution of a nonlinear SVM. In the same section we describe existing applications of RFF. Subsection 2.1 provides the necessary theoretical background and restates crucial points of the original publication. In section 3.1 we partially reproduce the results by Rahimi and Recht and argue about qualities of the input data that increase approximation rates above theoretically guarantees. We also argue about potential effects of using RFF on existing heuristics used with kernel methods. In section 3.2 we evaluate the effect of using RFF mappings with linear SVM, we show the benefits and possible limitations of this approach. Finally, Section 4 concludes our findings.

2 Related Work

Several other methods for kernel approximation have been developed. As shown by Williams and Seeger [23] Nystrom method can be used for approximation of Gram matrices used by problem (3) in $\mathcal{O}(Nd)$ time and using $\mathcal{O}(N)$ extra space. As was later shown by Jin et al. [7], this provides theoretical approximation rates of $\mathcal{O}(\sqrt{N})$ or faster when there is a large eigengap in the spectrum of a kernel matrix. However, experimental results by Le et al. [9] did not show a significant difference in approximation rates between RFF and Nystrom method on real world datasets.

Isotropic stationary kernels, that depend only on the quadratic norm of the input vector difference $K(x, y) = K_i(\|x - y\|^2)$, can be approximated at even higher speed $\mathcal{O}(N * log(d))$ comparing to RFF with the FastFood method described by Le et al. [9]. Experiments performed in the paper show that all three methods Nystrom, RFF and FastFood provide comparable accuracy for equivalent dimension of feature space on real-world datasets.

Since time of publication RFF have been studied and applied for several learning techniques. Chitta et al. [6] in their research use RFF to improve performance of kernel clustering algorithm. Authors state, that applying RFF allows to achieve results similar to state-of-the-art kernel clustering algorithms. Authors also provide theoretical justification, that k-means algorithm using RFF provides approximation error of $\mathcal{O}(1/D)$, where D is a number of Fourier components.

Bo and Sminchisescu [2] used RFF to approximate kernel values in context of visual recognition problems. According to experimental data, their approach allowed to achieve results comparable to best known techniques on some difficult computer vision datasets [2] and proved to be efficient. Authors observer significant decrease in learning time using RFF with linear SVM comparing to methods, using nonlinear SVM.

Maillard and Munos [12] applied RFF to solve the least-squares regression problem. Authors show that it allows to achieve an estimation error of order $\mathcal{O}(log(\sqrt{N}))$ for a feature space of size $D = \mathcal{O}(\sqrt{N})$.

Researchers from Duke University [15] used RFF to design compact binary codes in context of the information retrieval task. The paper provides full theoretical analysis of convergence properties and experimental results of using the developed technique along with state-of-the-art methods.

2.1 Random Fourier Features

The technique presented by Rahimi and Recht [16] targets stationary kernels. Stationary or shift-invariant kernels rely only on a vector difference of the inputs:

$$k_s(z) = k(x, y), \text{ where } z = x - y \tag{4}$$

According to Bochner's theorem [3] there always exists a transformation of positive definite function of form:

$$k_s(x - y) = \int_{\mathbb{R}^d} p(\omega) e^{j\omega^T (x-y)} d\omega \tag{5}$$

where $p(\omega)$ is a positive finite measure. Which means that $k_s(z)$ is a Fourier transform of function $p(\omega)$. In turn, finite function $p(\omega)$, when scaled, is a proper probability distribution. For the radial basis function kernel $k_s(z) = \exp -\gamma \left\|z\right\|^2$, where $\gamma = \frac{1}{2\sigma^2}$, that we are going to work with, such a distribution is defined the following way:

$$p(\omega)/k_s(0) = \frac{1}{2\sqrt{\pi\gamma}} e^{-\frac{\|\omega\|^2}{2}} = \frac{1}{\sqrt{2\gamma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{\|\omega\|^2}{2}} = \frac{1}{\sqrt{2\gamma}} f(\omega|0, 1) \tag{6}$$

where $f(\omega|\mu, \sigma^2)$ is a normal distribution.

By transformation (5), given $\zeta_\omega(x) = e^{j\omega' x}$, we can construct an unbiased estimate of $k_s(x - y)$:

$$k_s(x - y) = E_\omega[\zeta_\omega(x)\zeta_\omega(y)^*], \text{ where } \omega \text{ are drawn from } p(\omega). \tag{7}$$

Kernel value $k_s(x - y)$ can be approximated with any target variance by taking average result of D such mappings. Since kernel function $k_s(z)$ is real-valued $e^{j\omega^T (x-y)}$ can be replaced with $\cos \omega(x - y)$ under the integral sign. Authors of the paper propose next 2 mappings of $\zeta_\omega(x)$ that allow to estimate function $\cos \omega(x - y)$:

$$\begin{aligned} z_\omega(x) =& [\cos(\omega x) \sin(\omega x)]' \\ z_\omega(x) =& [\sqrt{2} \cos(\omega' x + b)], \text{ where } b \text{ is drawn uniformly from } [0, 2\pi] \end{aligned} \tag{8}$$

since $z_\omega(x)' z_\omega(y) = \cos \omega'(x - y)$.

Altogether, to produce a mapping of a d-dimensional input vector x into a random feature space of size D we construct a vector of form:

$$z(x) = \{z_{w_0}(x), z_{w_1}(x), \ldots, z_{w_{D-1}}(x)\} \tag{9}$$

where $w_i | \forall i \in [0, D-1]$ are drawn uniformly at random from distribution $p(\omega)/k_s(0)$ and z_w is one of the form (8). Mapping a single input requires time proportional to $\mathcal{O}(dD)$.

Finally, by Hoeffding's inequality:

$$Pr\big[|z(x)'z(y) - k(x,y)| \geq \epsilon\big] \leq 2\exp(-D\epsilon^2/2) \tag{10}$$

From which we can derive next guideline for selection of projection size D:

$$D(\epsilon, \delta) = -\frac{2ln\frac{\delta}{2}}{\epsilon^2}, \text{ where } \epsilon > 0, \delta\epsilon(0,1) \tag{11}$$

where resulting value indicates the size of projection D to limit probability $\delta = Pr\big[|z(x)'z(y) - k(x,y)| \geq \in \big]$ for a any pair of inputs.

We constructed a mapping of input vectors $z : X \to \mathbb{R}^D$, or more specifically $z(x) = [z_{\omega_1}(x), z_{\omega_2}(x), \ldots, z_{\omega_D}(x)]$, where ω_i are chosen uniformly at random from $p(\omega)$. This allows to estimate kernel function (2) for a family of functions (4) arbitrary well with a linear product:

$$k(x,y) = \langle \Phi(x), \Phi(y) \rangle \approx z(x)'z(y) \tag{12}$$

Resulting mapping $z(x) = [z_{\omega_1}(x), z_{\omega_2}(x), \ldots, z_{\omega_D}(x)]$ represents a projection into low-dimensional feature space of size D and can be used with linear SVM.

3 Evaluation

In this section we take a closer look at the properties of the radial basis function kernel approximation. In the first subsection we analyse qualities of kernel function approximation itself and make assumption about preferable qualities of the input data. Second subsection analyses differences between non-linear SVM and approximating linear SVM based on RFF.

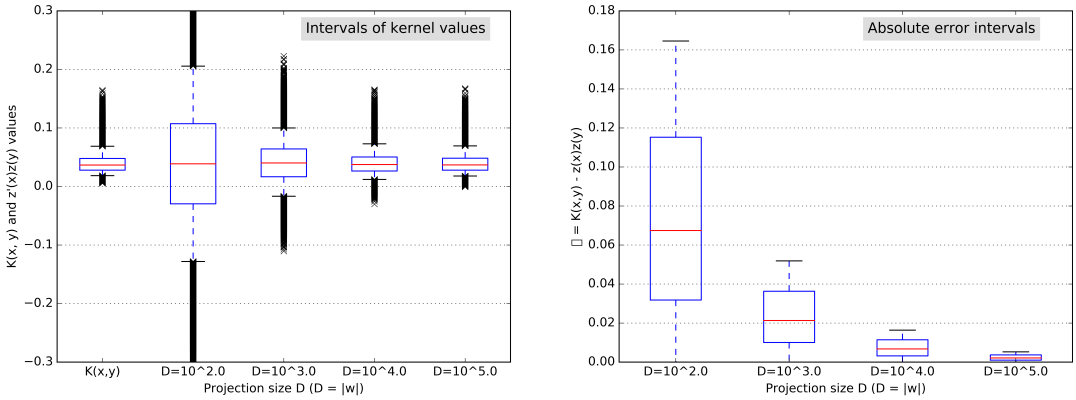
Experiments were performed on a regular computer (Intel Core i7 2.3GHz, 8GB RAM) running `Mac OS X 10.11` and `python 2.7`. SVM algorithms implementations from `sklearn 0.17` [14] and `numpy 1.10` [18] were used. Every test run was designed in a way that complete memory footprint would feet into the RAM and swap file would not be used to allow direct comparison of the running time between different learners and sizes of training sets.

3.1 RBF function approximation

For this section we implemented RFF sampling and projection logic. Both mappings (8) were used in implementation. Experiments with quality of kernel values showed no statistically significant difference between those two mappings for equivalent projection size. Further experiments were conducted for mapping $z_\omega(x) = cos(\omega x + b)$, where b is selected uniformly at random from $[0; 2\pi)$ [16].

For validation of theoretical result synthetic dataset has been designed. Every input is sampled uniformly at random from d-dimensional spherical Gaussian distribution with zero mean and 1-variance along the axis:

$$x_{ij} \in G(m = 0, \sigma = 1) | \forall i \in [0, N), j \in [0, d) \tag{13}$$



(a) RBF kernel value approximation depending on number of random features. Actual kernel values shown in the left column.

(b) Absolute approximation error depending on the projection size D.

Figure 1: RBF kernel approximation with RFF. Median value (red), 25 through 75 percentile (box), 5th through 95th percentile (dashed), outliers.

where N is the number of generated examples and d is the dimensionality of input data. For example, for 3-dimensional data a single input vector might look like $x_0 = \{0.729, 1.921, -0.503\}$.

For the first experiment we generate a dataset of size $N = 1000$ according to (13) in R^{100} ($d = 100$) space. In the experiment we perform pairwise comparison of kernel values (total of approx. $N^2/2 = 500000$). Result is shown on Figure 1.

Results of the test confirms inverse quadratic relation between approximation error ϵ and projection size D expressed by (11) and (10). Comparison of predicted and actual values in Table 1 suggests that accuracy expressed by inequality (10) can be achieved with smaller projection sizes at least for some inputs. Experiment depicted in the Figure 2a confirms this finding.

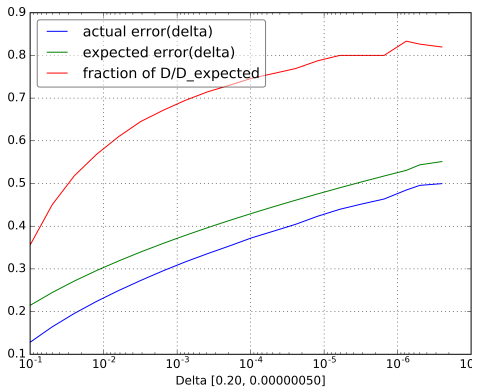
Experiment on Figure 2a uses single input of size $N = 20000, x \in \mathbb{R}^{50}$ (over 2 millions kernel values are analyzed) to verify theoretical results expressed by inequality (10). Inequality (10) provides a guarantee of absolute approximation error for a fraction $(1 - \delta)$ of all kernel values. Experiment shows, that inequality for a particular input holds even for smaller projection size D_{actual} than predicted $D_{required}$ for any delta $\delta = Pr\left[|z(x)'z(y) - k(x, y)| \geq \epsilon\right]$ arbitrary close to zero.

Altering the dimensionality of the input in experiment on Figure 2a showed, that for input design (13) when dimensionality d increases, fraction of $(D_{required}/D_{actual})$ increases and approaches 1. In other words, difference between guaranteed and sufficient projection size disappears. Unfortunately, we can not directly compare results for datasets that have different dimensionality d. Mainly, because adding dimensions to the input data might effect output of the stationary kernel. In case of RBF kernels, difference in kernel outputs most probably would require to readjust kernel parameters, depending on the heuristic used to select the parameters [10].

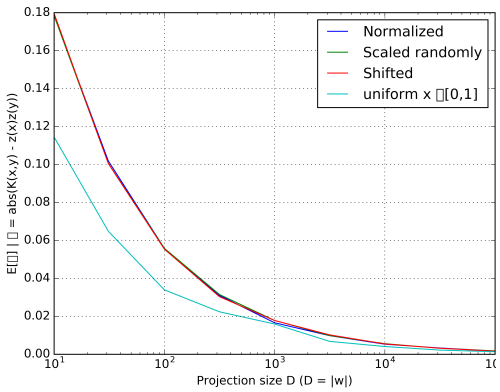
Since both absolute approximation error and kernel parameters might effect qualities of linear SVM solution, heuristic for choosing kernel parameters, as for RBF kernels, might be effected. Or more specifically, when using linear SVM with RFF as a standalone learner,

Delta δ	Max.error for δ	Predicted max.error of for δ	Predicted D for actual error
$\delta = 0.25$	0.082	0.204	615
$\delta = 0.05$	0.116	0.271	549
$\delta = 0.0002$	0.287	0.429	221

Table 1: Predicted and actual kernel approximation results on a synthetic dataset for $D = 100$



(a) Predicted (green) and actual (blue) errors depending on input fracture (delta). Fraction of the projection size $D = D_{actual}/D_{req}$ (red) sufficient to achieve actual error.



(b) Effects of data irregularities. Expected approximation error depending on projection size D.

Figure 2: Kernel approximation qualities.

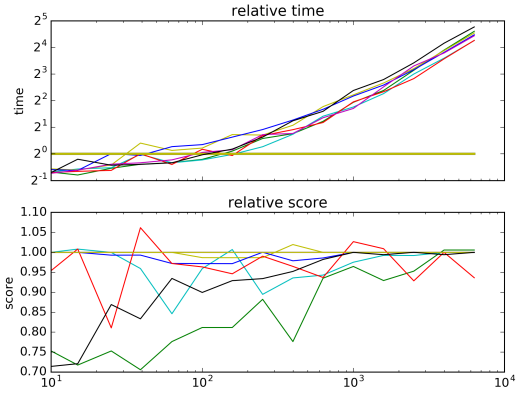
heuristic for choosing best parameters to generate RFF might be different from heuristic used for kernel of the nonlinear SVM learner. For example, from two sets of parameter values that perform equally well with nonlinear SVM we would prefer one, that would be less affected by approximation error when used to generate RFF for linear SVM.

At last, we studied the effect of qualities of the input space on the quality of kernel approximation. For this experiment we used input data from experiment 1 and randomly denormalized the data i.e. shifted and scaled. Additionally, we used a different distribution to peek input components. As shown on Figure 2b, for multidimensional gaussian inputs normalization or denormalization have no significant effect on expected approximation error. On the contrary, form of distribution of the inputs might effect the approximation speed. This difference can be explained by the difference in qualities of the $\epsilon - net$ produced inputs.

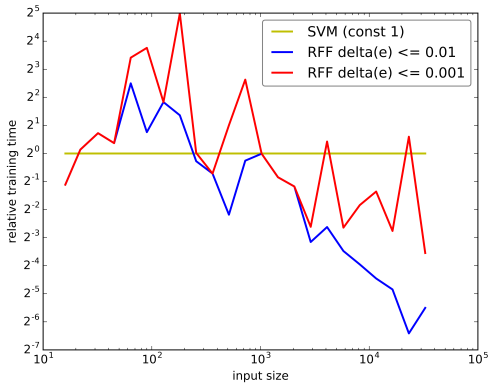
3.2 Approximate solution of SVM using Random Fourier Features

In this subsection we look at qualities of the solutions produced by linear SVM algorithm using RFF mapping and compare them with a solution of a reference SVM solving the empirical risk optimization problem in dual kernel form (3).

First experiments were conducted on a number of small classification datasets (Horse Colic, Heart Statlog, Breast Cancer Wisconsin, Hepatitis, Credit Approval, German Credit Data [11]). Results are shown on Figure 3a. They suggest, that for small dataset sizes approximation



(a) Training time (top) and score (bottom) relative to the nonlinear learner (straight line on both plots) depending on the projection size for several small datasets [11].



(b) Relative training time of linear SVM (red, blue) using RFF comparing to reference SVM (yellow). Red and blue lines correspond to different approximation errors: 0.01 (blue) and 0.001 (red).

Figure 3: Early SVM approximation experiments.

algorithm provides limited improvement in running time. Quality of approximation varies significantly between datasets. This experiments led to the conclusion, that synthetic dataset must allow to choose arbitrary number of inputs with unavoidable small error i.e. it must be unlikely to be able to classify the inputs perfectly.

First results for synthetic dataset are shown on Figure 3b. Input space in represented by classification problem. Inputs of each class are picked from 2-dimensional gaussian cluster. Clusters have non empty intersection. First experiments with described dataset showed, that approximation of a SVM is unlikely to outperform reference learner and all further tests were conducted for approximation absolute errors $e = \{0.01, 0.001\}$.

As was shown in subsection 3.1, inputs picked according to multidimensional gaussian distribution tend to show higher approximation speed than stated in the original publication. Additionally, relation between kernel value approximation and relative error of a linear learner using RFF has not been analyzed. In absence of direct theoretical justification for projection size a fixed set of projection sizes is generated for each experiment:

$$\mathcal{D} = \{D_{min}, D_{min}c^1, D_{min}c^2, \ldots, D_{max}\}$$

where $c \in \mathbb{R}$ (typically $\sqrt[3]{10}$ or $\sqrt[4]{10}$) and D_{max} has proven to be sufficient to reach required approximation accuracy. Every approximating learner runs multiple times for every setting combination.

Recommendation above were implemented in experiment depicted on Figure 4. Inputs represent two intersecting gaussian clusters $X, Y \in \mathbb{R}^d$. Each input component is chosen according

to gaussian distribution:

$$
\begin{aligned}
&x_i \in G(m=0,\sigma=1)\cdot scale_{xi}\\
&y_i \in G(m=0,\sigma=1)\cdot scale_{yi} + shift_i\\
&scale_{xi}, scale_{yi} \text{ chosen uniformly at random } [1, scale_{max}]\\
&shift = \frac{\frac{x}{\|x\|}\circ scale_x + \frac{y}{\|y\|}\circ scale_y}{\sqrt{d}} | x,y \in N(0,1)\\
&(x\circ y) \text{ - Hadamard product. } \circ : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n
\end{aligned}
\tag{14}
$$

Clusters constructed this way always have non-empty intersection, yet validation error remains relatively small $e \approx 0.1$. First two principal components of input $X \in \mathbb{R}^{10}$ constructed this way for the experiment are shown on Figure 4 (top-left).

To assess the score of each learner we use cross-validation [17]. Training set is not involved in the validation process. We have chosen to generate validation set of the same size as training set. Larger sets would necessary decrease amount of RAM memory accessible to the learners.

Experiment conducted for next input sizes and projection sizes:

$$\mathcal{N} = [1024, 1448, 2048, 2896, 4096, 5792, 8192, 11585, 16384, 23170, 32768, 46340, 65536]$$
$$\mathcal{D} = [10, 21, 46, 100, 215, 464, 1000, 2154, 4641, 14641, 10000]$$

Experiment on Figure 4 shows time complexity of the training and evaluation of linear learner using RFF. Two absolute approximation errors [0.01, 0.001] were used for comparison. Results of a reference SVM are shown alongside. Results for linear SVM are averaged among $k = 10$ distinct runs. Experimental data suggests, that linear learner maintains linear complexity in input size while using RFF mappings with fixed allowed approximation error. To emphasize this suggestion we fitted second degree polynomial to emulate resulting time complexity functions of two learners. Resulting parameters of the least-square fitting clearly suggest quadratic relationship for nonlinear SVM and linear relationship for linear learner. Resulting slopes are $k = 2.01 + -0.04$ for nonlinear SVM and $k = 0.98 + -3$ for linear SVM. As follows, while projection size to achieve equivalent accuracy for different input sizes stays constant, prediction time also stays constant regardless of the size of the training set.

At last, we designed an experiment to show tradeoff between the accuracy and the running time. Training and test datasets were generated according to rules (14) for $d = 6$. Learning performed for $N = 10000$ inputs and projection sizes $100, 320, 1000, 3200, 10000$. In this experiment we want to pay special attention to the accuracy of the approximation linear learner in situation, when the it's learning time is greater or equal than learning time of reference learner. For that reason largest projection size is chosen equal to number of input examples. We also interested in the deviation of the resulting error. Learning is repeated $k = 100$ times for every projection size. Best and worst 5 runs by score are ignored, rest are depicted on the plot along with expected values of validation error and running time.

Results are shown on Figure 5. Regardless of the projection size, expected validation error of the approximation is greater than error of the reference learner. SVM using RFF showed predictable running time for every projection size. Variance of the approximation error is order of approximation error itself and must be taken into account. Approximations that require even greater running time than the reference learner might seem an unreasonable choice, but they can still have better evaluation time and might show more flexibility in context of distributed systems or online learning.

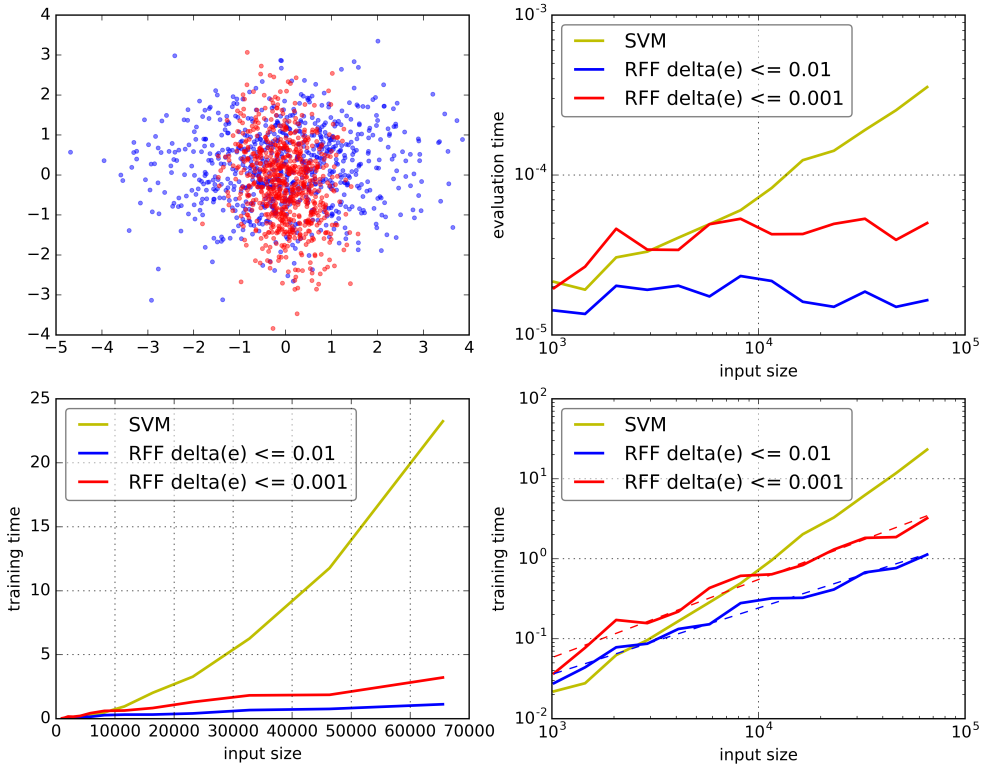


Figure 4: Training time of approximating linear SVM. Low dimensional input. Top-left: 2d PCA of the 2000 inputs. Top-right: average evaluation time per input. Bottom-right: training time. Bottom-left: training time on a LogLog scale with second degree polynomial (dashed) fitted using least-squares.

4 Conclusion

In this work we conduct a extensive evaluation comparing qualities of the approximating linear SVM using RFF to the reference nonlinear SVM. In the first part of our study we confirm theoretical results of the original publication by Rahimi and Recht [16]. We show that some qualities of the input data and required approximation accuracy allow higher approximation rates than theoretically guaranteed. We also analyze the properties of the solution produced by the linear algorithm for SVMs when using RFF. Our experimental data confirms, that required dimensionality of the random feature space remains constant regardless of the size of the training set for a fixed absolute approximation error. Therefore, approximate solution can indeed be found in linear time. Approximating nonlinear concept with linear SVMs provides a possibility of a trade off between solution accuracy and running time of learning and prediction. We also showed, that approximation technique is unlikely to produce equal or superior solutions to the reference nonlinear SVM algorithm even while using equivalent amount of computational time.

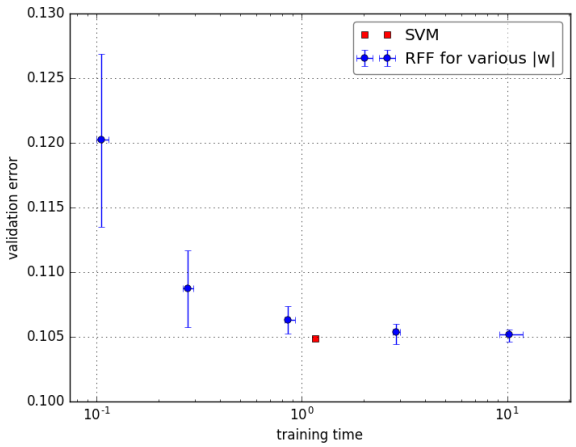


Figure 5: Expected training time and training error of approximation linear SVM. Blue lines cover 5th through 95th percentile. Projection sizes D=[100, 320, 1000, 3200, 10000], N=10000, n=6.

References

[1] R. C. Williamson B. Schoelkopf, A. J. Smola and P. L. Bartlett. New support vector algorithms, 2000.

[2] L. Bo and C. Sminchisescu. Efficient match kernel between sets of features for visual recognition, 2009.

[3] Salomon Bochner. Harmonic analysis and the theory of probability, 1955.

[4] Boser, B. E., Guyon, I. M., and V. N. Vapnik. A training algorithm for optimal margin classifiers, 1992.

[5] Léon Bottou, Jason Weston, and Gökhan H. Bakir. Breaking svm complexity with cross-training, 2005. URL `http://papers.nips.cc/paper/2695-breaking-svm-complexity-with-cross-training.pdf`.

[6] Radha Chitta, Rong Jin, and Anil K. Jain. Efficient kernel clustering using random fourier features, 2012.

[7] R. Jin, T. Yang, M. Mahdavi, Y.F. Li, and Z.H. Zhou. Improved bound for the nystroms method and its application to kernel classification, 2011.

[8] Thorsten Joachims. Training linear svms in linear time, 2006.

[9] Quoc Le, Tamas Sarlos, and Alex Smola. Fastfood - approximating kernel expansions in loglinear time, 2013.

[10] Cheng-Hsuan Li, Chin-Teng Lin, Bor-Chen Kuo, Hui-Shan Chu, and . An automatic method for selecting the parameter of the normalized kernel function to support vector machines, 2012.

[11] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[12] O. Maillard and R. Munos. Compressed least-squares regression, 2009.

[13] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations, 1909.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python, 2011.

[15] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels, 2009.

[16] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines, 2007.

[17] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Roy. Stat. Soc.*, 36:111–147, 1974.

[18] S. Chris Colbert Stfan van der Walt and Gal Varoquaux. The numpy array: A structure for efficient numerical computation, 2011.

[19] V. Vapnik. Statistical learning theory, 1998.

[20] V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. automation and remote control, 1964.

[21] V. Vapnik and A. Chervonenkis. Pattern recognition theory, statistical learning problems, 1974.

[22] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method, 1963.

[23] C. K. I. Williams and M. Seeger. Using the nystrom method to speed up kernel machines, 2001.