Iteratively Refining SVMs using Priors

Enric Junqué de Fortuny*, Theodoros Evgeniou*, David Martens†, Foster Provost‡
*INSEAD, Boulevard de Constance, 77305 Fontainebleau, France
†Faculty of Applied Economics, University of Antwerp, Belgium
‡ Information, Operations & Management Sciences, Stern School of Business, New York University, New York

***Abstract*—Research on scalable machine learning algorithms has gained a considerable amount of traction since the exponential growth in data assets during the past decades. Many Big Data applications resort to somewhat "simple" data modelling techniques due to the computational constraints associated with more complex models. Simple models, while being very efficient to estimate, often fail to capture some of the finer details of more complex datasets. In this manuscript, we explore the idea that complex large scale classification can be tractable using a process of iterative refining. In such a process, we focus on non-linearities of the data only *after* having first found an approximate linear model. This knowledge is then incorporated into the non-linear model implicitly, allowing the non-linear model to focus on important parts of the data after a rough first estimation. This in turn reduces overall training time and allows for a richer model representation, eventually leading to more predictive power.**

***Index Terms*—SVM; iterative refining; big data; prior**

I. Introduction

In a standard classification problem the goal is to discriminate between points of opposing class in a certain input space. Many classification techniques have been developed during the past decades, ranging from simple linear models to very elaborate procedures. One notable example that we will build upon in the centre of this spectrum is the well-known Support Vector Machine [1]. Like many other techniques, the SVM allows for variations in the complexity of feature representations through the introduction of kernels. Applying the richer kernels variants (e.g., RBF) naively to large datasets can become prohibitively expensive as either the number of instances or the number of features increases. In these situations we are therefore often limited to using simple (linear) models, even though richer representations might be more appropriate.

To this end, much effort has been geared towards improving performance of such heavy duty classifiers. For instance, the default approach to optimizing an SVM is to formulate it as a Quadratic Programming problem. While relatively fast off-the-shelve solvers exist for QP problems, they are known to not scale well ($\mathcal{O}(n^3)$) as the amount of input n increases. In consequence, algorithmic variants have been developed to cope with this limitation such as online gradient descent methods (including the well known SMO algorithm [2] and Pegasos [3], the design of sparse variants of the techniques, low-order/high-order information usage and parallelization [4].

We argue that very often, we can relieve the heavy duty kernel of a great part of its work by first making a linear approximation and only using the more complex kernel for those parts in space that are not approximately linearly separable. That is, most of the signal in the dataset can be recovered using a simple representation and it is only after recovering this signal, that one ought to focus on the more difficult parts of the signal. The key insight here will be that we do not need to identify these regions manually, but we can incorporate them implicitly in the modelling procedure using a step-wise classification. The main focus of this manuscript is thus to show how one can include prior knowledge into a more complex model, leading to an iterative procedure where we refine the decision boundary in each step.

II. Iterative refining

A. Notation

We begin by introducing the problem using standard notation: given a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ of n input vectors $\mathbf{x}_i \in \mathbb{R}^m$ and associated target class labels $y_i \in \{0, 1\}$, we want to be able to predict y_i from $\mathbf{x}_i$ using a function $u(\mathbf{x})$ with $u : \mathbb{R}^m \to \mathbb{R}$ (the real value is then afterwards converted to a binary decision). For instance, for a linear SVM the scoring functions would simply be a linear function:

$$u(\mathbf{x}_i) = \mathbf{u} \cdot \mathbf{x}_i - b \tag{1a}$$

$$= \langle \mathbf{u}, \mathbf{x}_i \rangle \tag{1b}$$

Here, $\mathbf{u}$ is a parameter vector that must be sought during training and b is the bias which we can drop without loss of generality (for more details we refer the reader to the relevant literature [5]). The continuous output value from $u(\mathbf{x}_i)$ is then converted to a discrete class label (in the simplest case, based on the sign of the score). If the data is not linearly separable, one can look at more complex relations. These functions usually incorporate some modified form of the inner product $\langle \cdot, \cdot \rangle$ and an associated Hilbert space $\mathcal{U}$ to incorporate a metric of similarity between input vectors.

B. Iterative refining as a convex combination of functions

A toy example to explain the process of iterative refining is shown in Figure 1. The example dataset is an artificial combination of a linear function and a Gaussian, the task is to learn a separating hyperplane that maximizes the margin between the red and green input data points.

a) Basic idea: As explained in the introduction, we will first train a simple linear model that operates on the input vectors to reach an initial solution and refine this solution afterwards:

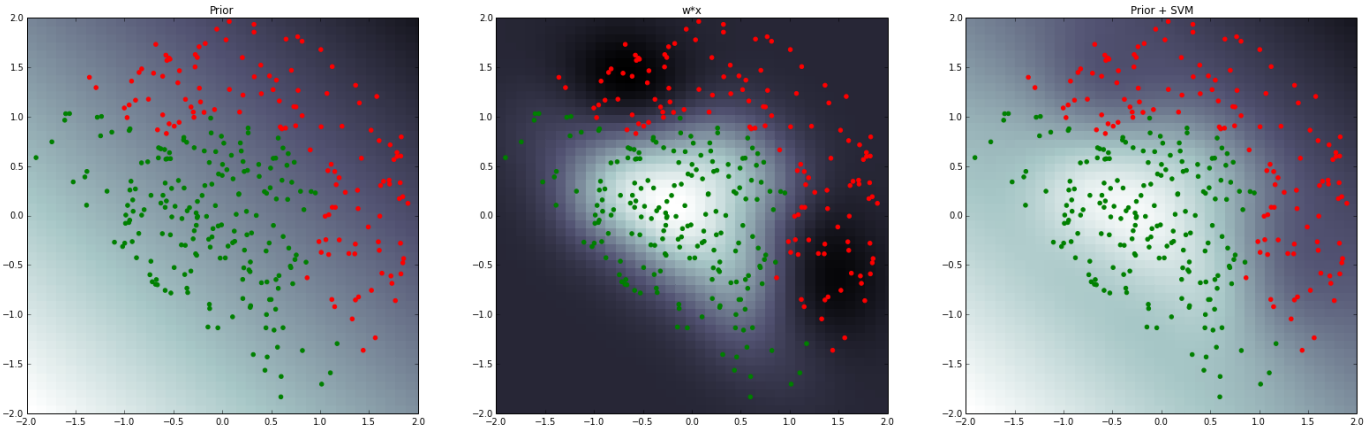

Fig. 1: A toy example in which the combination (right) of the prior model (left) and the complex model (middle) is able to almost perfectly separate between objects of the green and red class. The scoring function is shown by the gradient color in the background.

$$u(\mathbf{x}) = \langle \mathbf{u}, \mathbf{x} \rangle \tag{2}$$

We will call this the *prior*, because $\mathbf{u}$ will not be allowed to change after it has been found and can thus be considered to be a non-subjective fixed prior. In a next step, we refine this initial estimate by including non-linearities using a model that operates in another reproducing kernel Hilbert space $\mathcal{V}$ on top of the prior. The final model then combines both of these as a convex combination:

$$w(\mathbf{x}) = u(\mathbf{x}) + v(\mathbf{x}) \tag{3a}$$
$$= \langle \mathbf{u}, \mathbf{x} \rangle + \langle \mathbf{v}, \mathbf{x} \rangle_{\mathcal{V}} \tag{3b}$$

Note how, very often, one will introduce the Reproducing Kernel Hilbert Space (RKHS) through either a feature map $\Phi : \mathcal{X} \rightarrow \mathcal{V}$ or a kernel function K with $\langle \phi(x_1), \phi(x_2) \rangle = \langle x_1, x_2 \rangle_{\mathcal{V}} = K(\mathbf{x}_1, \mathbf{x}_2)$.

b) Generalization: More generally, one can see this process as learning in increasingly more complex (independent) subspaces of a larger Hilbert space at different iterations. Let us call these individual Hilbert subspaces $\mathcal{U}$ (for the prior, $\mathcal{U}$ is $\mathbb{R}^m$) and $\mathcal{V}$ (for the non-linear model, possibly infinitely dimensional). It is then possible to represent the iterative process as happening in a larger space $\mathcal{W}$ which is the Hilbert space direct sum of the subspaces (defined by the $\oplus$ operator):

$$\mathcal{W} : \mathcal{U} \oplus \mathcal{V} = \{u \oplus v : u \in \mathcal{U}, v \in \mathcal{V}\} \tag{4}$$

Note how this leads to the Equation 3b in our case because the inner product associated with the resulting Hilbert space can be distributed accordingly:

$$\langle (\mathbf{u} \oplus \mathbf{v}), (\mathbf{x} \oplus \mathbf{x}) \rangle_{\mathcal{U} \oplus \mathcal{V}} = \langle \mathbf{u}, \mathbf{x} \rangle_{\mathcal{U}} + \langle \mathbf{v}, \mathbf{x} \rangle_{\mathcal{V}} \tag{5}$$

It is of course possible to generalize this to the sum of a finite set of J Hilbert spaces $\{(h_j) : h_j \in \mathcal{H}_j\}$ as follows:

$$\mathcal{W} = \bigoplus_{j=1}^{J} \mathcal{H}_j$$

$$\langle h, g \rangle_{\mathcal{W}} = \sum_{j=1}^{J} \langle h_j, g_j \rangle_{\mathcal{H}_i}$$

$$||h||_{\mathcal{W}} = \left[\sum_{j=1}^{J} ||h_n||^2 \right]^{-1/2}$$

For our specific purpose, it will suffice to stick to the combination of two spaces for the remainder of this manuscript (i.e., a linear prior and subsequent RBF-based model). We envision the development of more complex combinations; e.g., in text mining many distance metrics exist and one could iteratively add kernels based on these different prior metrics (Hamming, Euclidian, ...).

C. The iterative refining optimization

We will formulate the learning mechanism as a generic regularization functional, where we aim to minimize the usual trade-off between complexity and accuracy:

$$\underset{w \in \mathcal{W}}{\text{minimize}} \quad Q(w) + \mu\, \Omega(w)$$

Here, Q is a *loss function* which measures the empirical loss of the previously defined w-function (Eq. 3a), Ω is the *smoothness* of the solution and μ controls the trade-off between both (and can be found using cross-validation).

The main particularity of the way in which iterative refining works is that after building a prior part of the model ($\mathbf{u}$), only the remainder of the model may be changed (this is of key importance). In the formulation of the previous section, this means that $\mathbf{u}$ is considered to be constant in the optimization procedure. Since we cannot update $\mathbf{u}$ at this stage, we only

need to control for the smoothness of the new part ($\mathbf{v}$), the functional to minimize then becomes:

$$\underset{\mathbf{v}}{\text{minimize}} \quad Q(w) + \mu \, ||\mathbf{v}||^2_{\mathcal{V}} \tag{6}$$

Choosing the SVM loss for Q leads us to the following optimization functional for the second step of our iteration:

$$\underset{\mathbf{v}}{\text{minimize}} \quad \sum_{i=1}^{n} \max\{0, 1 - y_i w(\mathbf{x})\} + \mu \, ||\mathbf{v}||^2_{\mathcal{V}} \tag{7}$$

Of course, one might choose other loss function where appropriate.

D. Interpretation as a prior

So far we have not really used the fact that the previous stage model ($\mathbf{u}$) is a prior, besides noting that it does not change during optimization. Interestingly, Equation 6 can also be interpreted as a regularization procedure 'towards' the linear prior. This is easily seen after substituting $\mathbf{v} = \mathbf{w} - \mathbf{u}$ (cfr. 3b) for two linear models defined by parameter vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\underset{\mathbf{w}}{\text{minimize}} \quad Q(w) + \mu \, ||\mathbf{w} - \mathbf{u}||^2 \tag{8}$$

For the case of two linear models, the regularization component is essentially penalizing for excessive dissimilarity from $\mathbf{u}$. In the special case of $\mathbf{u} = \mathbf{0}$, we recover the usual maximal margin classification functional:

$$\underset{\mathbf{v}}{\text{minimize}} \quad Q(v) + \mu \, ||\mathbf{v}||^2 \tag{9}$$

In this light, our procedure can thus be seen as a generalization of the typical SVM optimization procedure, but with inclusion of a prior.

E. Implementation

We will use the Pegasos [3] online stochastic gradient descent method to optimize the functional in Equation 7. In an online stochastic gradient descent method, at each iteration t, a parameter vector $\mathbf{w}_t$ is updated according to the loss with respect to a randomly chosen input sample. Taking into account the fact that $\mathbf{u}$ is constant, we can rewrite w_t as follows:

$$w_t(\mathbf{x}) = \langle \mathbf{w}_t, \mathbf{x} \rangle \tag{10a}$$

$$= \langle \mathbf{u}, \mathbf{x} \rangle_{\mathcal{U}} + \langle \mathbf{v}_t, \mathbf{x} \rangle_{\mathcal{V}} \tag{10b}$$

Our goal functional then becomes:

$$\underset{\mathbf{v}}{\text{minimize}} \quad G(\mathbf{v}_t) = \frac{\lambda}{2} ||\mathbf{v}_t||^2_{\mathcal{V}} + \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i w_t(\mathbf{x}_i)\} \tag{11}$$

At each step t we follow the gradient with respect to this functional after seeing one example $\{\mathbf{x}_{i_t}, y_{i_t}\}$:

$$\nabla_t G(\mathbf{v}_t) = \lambda(\mathbf{v}_t) + \begin{cases} 0 & \text{if} \quad y_{i_t} w_t(\mathbf{x}_{i_t}) \geq 1 \\ -y_{i_t} \mathbf{x}_{i_t} & \text{otherwise} \end{cases} \tag{12}$$

The update rule for $\mathbf{v}$ then becomes:

$$\mathbf{v}_{t+1} \leftarrow \mathbf{v}_t - \eta_t \cdot \nabla_t G(\mathbf{v}_t) \tag{13a}$$

$$\leftarrow \eta \sum_{i=1}^{n} \gamma_i y_i \mathbf{x}_i \tag{13b}$$

$$\gamma_i = \sum_{k<t+1} I\left[i_k = i \wedge y_i w_k(\mathbf{x}_i) < 1\right] \tag{13c}$$

With $\eta = 1/(\lambda t)$ and γ_i a counter which counts how often the prediction for an example was wrong ($y_i w_k(\mathbf{x}_i) < 1$) when it was picked ($i_k = i$) at a prior time ($k < t + 1$). Leading to our final model after convergence:

$$w_t(\mathbf{x}) = u(\mathbf{x}) + v_t(\mathbf{x}) \tag{14}$$

The latter function is of course the same as the linear combination stated in the beginning of this manuscript (Equation 3b). Thus this method indeed optimizes the linear combination, albeit in an iterative manner where prior information is taken into account yet remains unchanged.

III. Experiments

A. An illustrated example

Recall that the main idea behind the iterative inclusion of priors within the SVM is that a great part of the necessary modelling can be learned rather fast and one needs to focus on subtleties only at a later stage.

To illustrate this problem experimentally, consider the dataset displayed in Figure 1, where the objective is to discriminate objects of two classes (green, red). The ideal decision-boundary contains both a linear and a non-linear part by design. As can be seen from Figure 1, the linear model (left) was able to distinguish the two zones quite well, but it misclassified some of the object in the central zone at the same time. We thus proceeded by iterating on this model using the algorithm from Section II-E (where we used an RBF kernel SVM for the second iteration). In the middle of the figure we show only the RBF-SVM part. Note how it has focussed on the non-linear regions and did not have to bother about some of the already correctly identified regions. This has two advantages (a) by not having to focus on the other regions, the model converged faster and (b) the bandwidth of the RBF kernel was tuned to fit only the non-linear region, avoiding a bandwidth that is too wide. The combination of both (rightmost figure) shows the final, combined model which is superior to either one individually (Eq. 3b).

B. Big Data

To show the practical usefulness on Big Data, we consider a few datasets from different domains containing up to a few million input entries (see Table I). The resulting set-up is as follows:

1) **Prior learner:** A linear SVM model that has access to a large sample of the dataset.
2) **Pegasos:** An RBF-SVM model that has access to a limited (smaller) subset of the original training data for training purposes.

Dataset	n	m	Source	Description
a9a	48,842	123	[6]	predict income based on census data
w8a	64,700	300	[6]	web pattern detection
gisette	13,500	5,000	[7]	handwritten digit recognition problem
MiniBooNE	130,065	50	[8]	distinguish electron neutrinos (signal) from muon neutrinos (background)
ijcnn	141,691	22	[9]	IJCNN 2001 challenge
cod-rna	488,565	8	[10]	detection of non-coding RNAs
protein-homology	145,752	75	[6]	protein homology detection
SUSY	5,000,000	18	[11]	supersymmetric particle detection

TABLE I: Overview of large-scale datasets used in the experiments, most come from real problem domains and were acquired from the UCI learning repository [6].

3) **Pegasos with prior:** A RBF-SVM model that has access to the same limited (smaller) subset of the training data, but that also has access to the prior model.

By increasing the amount of available data, we can generate so called learning curves which demonstrate the characteristics of the classifier under varying data conditions. The results are displayed in Figure 2 and demonstrate how the Pegasos with prior performs better in most regions of the learning curves than a traditional Pegasos model. This was to be expected for the leftmost part of the learning curve because we intentionally added knowledge through the prior [1]. More interestingly, this advantage continues after the complex model has crossed the prior model's performance for many datasets.

C. Discussion

The empirical results confirm that our primary goal of reaching better performance with less data is being achieved, given that we have a good prior. We use a non-subjective prior trained on the data here for reasons of consistency in comparison, but it is reasonable to assume that other sources of priors could be used instead. While not shown explicitly here, the models that included prior information took less time to reach convergence as well. When interpreting the results from our empirical section, we must of course also keep in mind that if bigger and more complex datasets were to be used, the variance and scaling problems of the RBF Pegasos model would most likely worsen. We thus expect even better results in the future when applying these methods to even larger datasets.

IV. Comparison with previous work

The easiest way to compare the work presented in this manuscript to the existing literature is to consider the special case of two models as stated in Section II-D. We will perform the comparison along two equally important dimensions. First, we can study the differences with various theoretical frameworks that use their own formulations to include prior information. Second, we can contrast our application context to other's by looking at studies that end up using a model that is very close to ours in definition, but not in application.

[1]While not a necessary condition, it might be plausible that for some problems one is able to train a linear classifier only on a larger portion of the data.

A. Prior inclusion in the QP

In order to position our research in the broad body of work on inclusion of priors in SVMs, we will use an existing comparison table from a recent survey [12]. The survey identifies three types of ways to include prior information into SVM:

- Sample methods: incorporate prior knowledge by generating new data or reweighing existing data
- Kernel methods: adapt the kernel function to include prior information
- Optimization methods: include prior information into the optimization problem itself which then optimizes with respect to this prior knowledge

Our model falls in the optimization-based category. The distinction with prior literature can then be made as to how these methods alter the optimization problem. The general form of prior inclusion is given below:

$$\begin{aligned} \text{minimize} \quad & J_C = J_{SVM} + J_{Prior} \\ s.t. \quad & \mathbf{c}_{SVM} \leq \mathbf{0} \\ & \mathbf{c}_{Prior} \leq \mathbf{0}, \end{aligned}$$

where J_{SVM} and $\mathbf{c}_{SVM}$ corresponds to the usual SVM constraints and regularization and J_{Prior} and $\mathbf{c}_{Prior}$ corresponds to the new components based on the prior. An overview of the most closely related existing literature is given in Table II (based on [12]).

B. Applications of prior inclusion

1) Incremental OLR: In a previous research on credit scoring, [13] studied the inclusion of non-linear components on top of a linear model, in order to achieve better performance while keeping transparency. Their logistic model is defined as follows (using their notation):

$$\begin{aligned} P(y \leq i) &= \frac{1}{1 + \exp(-\theta_i - z)} \\ z &= (-\beta_1 x_1 - \ldots - \beta_m x_m) \\ &\quad + (-\beta_{m+1} f(x_{m+1}) - \ldots - \beta_n f(x_n)) \\ &\quad + (-w_1 \phi_1(\mathbf{x}) - \ldots - w_p \phi_p(\mathbf{x})) \end{aligned}$$

The first term group of z is a pure linear model, the second is still intrinsically linear because after the (possibly non-linear)

Method	J_C	$\mathbf{c}_{PK} \leq \mathbf{0}$
Virtual samples	$J_{SVM} + C\sum_{i=1}^{n_V} \hat{\xi}_i$	$\hat{y}_i(\langle \mathbf{w}, \hat{\mathbf{x}}_i \rangle + b) \geq 1 - \hat{\xi}_i$
Asymetric margin	$J_{SVM} + (C^+ - C)\sum_{i\in\mathcal{P}} \xi_i + (C^- - C)\sum_{i\in\mathcal{N}} \xi_i$	
WSVM	$J_{SVM} + \sum_{i=1}^{n}(C_i - C)\xi_i$	
Unlabeled samples	$J_{SVM} + \sum_{i=1}^{n_U}(C_i - C)\xi_i$	$\hat{y}_i(\langle \mathbf{w}, \hat{\mathbf{x}}_i \rangle) \geq 1 - \hat{\xi}_i$
Iterative refining	$\mu\|\|\mathbf{w} - \mathbf{u}\|\|^2$	

TABLE II: Overview of most closely-related research on incorporating prior knowledge with an optimization or sampling-based approach. Sampling methods generate a new number of data instances (n_v using known labels, n_u if the label is not known). Reweighing methods include some form of class prior constants C^+ and C^-.

transformation f is applied, a linear model is still being used to fit the terms. The third term group contains the truly non-linear SVM terms. If we remove the second term group and use the Nyrström approximation for the SVM component, we can rewrite the equation as follows:

$$z_{IL+SVM} = -\boldsymbol{\beta} \cdot \mathbf{x} - \sum_{i=1}^{n} w_i \phi_i(\mathbf{x})$$

$$\phi_i(\mathbf{x}) = \frac{\sqrt{n}}{\sqrt{v_i}} \sum_{k=1}^{n} u_{ki} K(\mathbf{x}_k, \mathbf{x})$$

$$z_{IL+SVM} = \langle \hat{\mathbf{v}}, \mathbf{x} \rangle + \langle \hat{\mathbf{w}}, \mathbf{x} \rangle_{\mathcal{H}}$$

with eigenvectors $\mathbf{u}_k$, eigenvalues v_i and $\phi_i(\mathbf{x}) = 0$ whenever $v_i = 0$. This last form similar to our Equation 3b, so one might conclude our method to be a generalization of the OLR-based method. Due to the difference in application, there are some important design characteristics that separate our method from the latter though. The main advantage of this form over our formulation is transparency: we can still inspect the weights and determine the importance of each factor. Unfortunately, the fact that both the eigenvector decomposition and Nyrström sampling need to be computed greatly increases computational expensiveness. In fact, our set-up is to be used in situations in which such operations can no longer be applied (and thus neither the decomposition or the sampling can be computed in reasonable amounts of time).

2) Iterative model updating: Also closely related, is the work on iterative model updating by Chapelle [14]. The problem they tackle, is the iterative updating of a model used for display advertising that goes out of sync over time. Their solution is to iteratively update the model using new data, each time updating $\mathbf{w}$ as the minimizer of:

$$\frac{1}{2}\sum_{i=1}^{d} q_i(w_i - m_i)^2 + \sum_{j=1}^{n} \log(1 + \exp(-y_j \mathbf{w}^T \mathbf{x}_j))$$

Here, each weight w_i has a prior $\mathcal{N}(m_i, q_i^{-1})$ initially. In next iterations, m_i is set to be equal to w_{i-1} and q_i is updated accordingly as well. Unlike in our work, their hypothesis space stays the same in each iteration.

3) Other related literature: The applications of such types of techniques are vast and the difference subtle, but important. We cannot cover all of the related literature here, but want to point out that our work shares links with multi-task learning [15], where - different from our scenario - the goals is to optimize multiple tasks at the same time (taking away the iterative aspect). Also related, is some of the literature in transfer learning [16] in which knowledge from a source domain is "transferred" to the target domain using a similar approach as Chapelle's [14]. Here again the main difference is that the hypothesis space remains static over all learning iterations whereas ours changes.

V. Conclusion

We started by positing a general way of combining Hilbert spaces iteratively, after which we used this framework in order to improve classification performance. More precisely, we iteratively scaled up complexity by incorporating prior information implicitly in the algorithm design. We then showed the relationship of this new framework to more traditional learning functionals and existing literature. The proposed implementation using Pegasos scales to datasets of up to billions of input samples while still achieving state-of-the-art performance in mere minutes. In future work we would like to investigate the interplay of more exotic function families. Furthermore, it could be interesting to investigate theoretical convergence limits of the proposed algorithms as well.

References

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, 1995.

[2] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft Research, Tech. Rep., 1998.

[3] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos," in *Proceedings of the 24th international conference on Machine learning - ICML '07*. New York, New York, USA: ACM Press, Jun. 2007, pp. 807–814.

[4] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent Advances of Large-Scale Linear Classification," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2584–2603, Sep. 2012.

[5] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, 2000.

[6] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository," 2007.

[7] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge." in *Advances in Neural Information Processing Systems 17*, L. S. and L.Bottou and Y. Weiss, Eds. MIT Press, 2005, pp. 545—-552.

[8] B. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor, "Boosted decision trees as an alternative to artificial neural networks for particle identification," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 543, no. 2-3, pp. 577–584, May 2005.

[9] D. Prokhorov, "IJCNN 2001 neural network competition," *Slide presentation in IJCNN*, 2001.

[10] A. Uzilov, J. Keegan, and D. Mathews, "Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change," *BMC bioinformatics*, 2006.

[11] P. Baldi, P. Sadowski, and D. Whiteson, "Deep Learning in High-Energy Physics: Improving the Search for Exotic Particles," Feb. 2014.
[12] F. Lauer and G. Bloch, "Incorporating prior knowledge in support vector machines for classification: A review," *Neurocomputing*, 2008.
[13] T. Van Gestel, D. Martens, B. Baesens, D. Feremans, J. Huysmans, and J. Vanthienen, "Forecasting and analyzing insurance companies' ratings," *International Journal of Forecasting*, vol. 23, no. 3, pp. 513–529, 2007.
[14] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM Transactions on Intelligent Systems and Technology*, 2014.
[15] T. Evgeniou and M. Pontil, "Regularized multi–task learning," *Proceedings of the tenth ACM SIGKDD ...*, 2004.
[16] B. Dalessandro, D. Chen, and T. Raeder, "Scalable hands-free transfer learning for online advertising," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

Fig. 2: Experiments on big datasets to study the convergence behaviour of Pegasos with prior. The green line represents the prior knowledge from the heuristic (in this case a linear Pegasos trained in less than 5 seconds on the full training set). For each dataset we show a full learning curve (top of panel) and a zoomed-in version (zoom on the y-axis; bottom of panel). The model with prior (RBF Pegasos+prior) performs better than the original model (RBF Pegasos) in most regions of the learning curve.