STEP SIZE-ADAPTED ONLINE SUPPORT VECTOR LEARNING

Alexandros Karatzoglou

Department of Statistics
Technische Universität Wien
Wiedner Hauptstraße 8–10, Austria

S.V.N. Vishwanathan, Nicol N. Schraudolph, Alex J. Smola

RSISE, Australian National University
Statistical Machine Learning Program
National ICT Australia, Canberra

ABSTRACT

We present an online Support Vector Machine (SVM) that uses Stochastic Meta-Descent (SMD) to adapt its step size automatically. We formulate the online learning problem as a stochastic gradient descent in Reproducing Kernel Hilbert Space (RKHS) and translate SMD to the nonparametric setting, where its gradient trace parameter is no longer a coefficient vector but an element of the RKHS. We derive efficient updates that allow us to perform the step size adaptation in linear time. We apply the online SVM framework to a variety of loss functions and in particular show how to achieve efficient online multiclass classification. Experimental evidence suggests that our algorithm outperforms existing methods.

1. INTRODUCTION

Stochastic ("online") gradient methods incrementally update their hypothesis by descending a stochastic approximation of the gradient computed from just the current observation. Although they require more iterations to converge than traditional deterministic ("batch") techniques, each iteration is faster as there is no need to go through the entire training set to measure the current gradient. For large, redundant data sets, or continuing (potentially non-stationary) streams of data, stochastic gradient thus outperforms classical optimization methods. Much work in this area centers on the key issue of choosing an appropriate time-dependent gradient step size η_t.

Recent years have seen a growing interest in stochastic gradient algorithms applied to kernel methods [1]. These algorithms typically let η_t simply decay in $O(t^{-\frac{1}{2}})$. Here we adopt the more sophisticated approach of *stochastic meta-descent*: performing a simultaneous stochastic gradient descent on the step size itself. Translating this technique into the kernel framework yields a fast online optimization method for SVMs.

Outline. We begin by providing an overview of SMD in Section 2. We then briefly describe the optimization problems arising from SVMs and Gaussian Processes in Section 3. The application of SMD to these problems is discussed in Section 4. Experiments are presented in Section 5, followed by a discussion.

2. STOCHASTIC META-DESCENT

The SMD online algorithm [2, 3] for gradient step size adaptation can greatly accelerate the convergence of stochastic gradient descent; successful applications to date include independent component analysis [4], turbulent flow modeling [5], and visual hand tracking [6]. SMD updates a system's parameters $\boldsymbol{\alpha}$ by the simple gradient descent

$$\boldsymbol{\alpha}_{t+1} \;=\; \boldsymbol{\alpha}_t \;-\; \boldsymbol{\eta}_t \cdot \boldsymbol{g}_t \,, \tag{1}$$

where $\boldsymbol{g}$ denotes the stochastic gradient, and $\cdot$ the element-wise (Hadamard) product. The vector $\boldsymbol{\eta}$ of individual step sizes is adapted multiplicatively

$$\boldsymbol{\eta}_t \;=\; \boldsymbol{\eta}_{t-1} \cdot \max(\tfrac{1}{2}, 1 + \mu\, \boldsymbol{v}_t \cdot \boldsymbol{g}_t) \tag{2}$$

using a scalar meta-learning rate μ. Finally, the auxiliary vector $\boldsymbol{v}$ used in (2) is itself updated iteratively via

$$\boldsymbol{v}_{t+1} \;=\; \varrho \boldsymbol{v}_t \;+\; \boldsymbol{\eta}_t \cdot (\boldsymbol{g}_t \;-\; \varrho \boldsymbol{G}_t \boldsymbol{v}_t) \,, \tag{3}$$

where $\boldsymbol{G}_t \succeq 0$ is an extended Gauss-Newton approximation [2] of the Hessian at time t, and $0 \leq \varrho \leq 1$ a decay factor. SMD is derived as a dual gradient descent procedure, minimizing the objective with respect to both $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ simultaneously. The $\boldsymbol{G}\boldsymbol{v}_t$ term in (3) is typically computed implicitly [2] using efficient procedures from algorithmic differentiation [7] that do not require explicit — and likely to be computationally expensive — computation of $\boldsymbol{G}$.

3. ONLINE KERNEL METHODS

We now present various kernel methods from a loss function and regularization point of view. Our notation closely follows [1] with minor modifications and extensions.

3.1. Optimization Problem

Denote by $\mathcal{X}$ the space of observations and $\mathcal{Y}$ be the space of labels (wherever appropriate). We use $|\,\mathcal{Y}\,|$ to denote the size of $\mathcal{Y}$. Given a sequence $\{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ of examples and a loss function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}$ the goal is to minimize the regularized risk

$$J(f) = \frac{1}{m} \sum_{i=1}^{m} l(\boldsymbol{x}_i, y_i, f) + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}}, \tag{4}$$

where $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) of functions on $\mathcal{X} \times \mathcal{Y}$. Its defining kernel is denoted by $k : (\mathcal{X} \times \mathcal{Y})^2 \to \mathbb{R}$ and it satisfies $\langle f, k((\boldsymbol{x}, y), \cdot)\rangle_{\mathcal{H}} = f(\boldsymbol{x}, y)$. In a departure from tradition we let our kernel depend on the labels as well as the observations. Finally, we make the assumption that l only depends on f via its evaluations at $f(\boldsymbol{x}_i, y)$ and that l is piecewise differentiable.

By the reproducing property of $\mathcal{H}$ we can compute derivatives of the evaluation functional. That is,

$$\partial_f f(\boldsymbol{x}, y) = \partial_f \langle f, k((\boldsymbol{x}, y), \cdot)\rangle_{\mathcal{H}} = k((\boldsymbol{x}, y), \cdot). \quad (5)$$

Since l depends on f only via its evaluations we can see that $\partial_f l(\boldsymbol{x}, y, f) \in \mathcal{H}$, and more specifically

$$\partial_f l(\boldsymbol{x}, y, f) \in \operatorname{span}\{k((\boldsymbol{x}, \tilde{y}), \cdot) \text{ where } \tilde{y} \in \mathcal{Y}\}. \quad (6)$$

Using the stochastic approximation of $J(f)$:

$$J_t(f) := l(\boldsymbol{x}_t, y_t, f) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}} \quad (7)$$

and setting $g_t := \partial_f J_t(f_t)$, we can write the following online learning algorithm:

Algorithm 1 Online learning (adaptive step size)

1. Initialize $f_0 = 0$
2. **Repeat**
   (a) Draw data sample $(\boldsymbol{x}_t, y_t)$
   (b) Adapt step size η_t
   (c) Update $f_{t+1} \leftarrow f_t - \eta_t g_t$

Practical considerations are how to implement steps 2.b and 2.c efficiently. We will discuss 2.c below. Step 2.b, which distinguishes the present paper from the update rules of previous algorithms, is discussed in Section 4.

3.2. Loss Functions

We now give specific details for two loss functions used in kernel methods. Similar derivations can be found for binary classification, logistic regression, ε-insensitive regression, Huber's robust regression, LMS problems, and graph-structured output domains [8].

Multiclass Classification Here we employ a definition of the margin arising from log-likelihood ratios. This leads to

$$l(\boldsymbol{x}, y, f) = \max(0, 1 + \max_{\tilde{y} \neq y} f(\boldsymbol{x}, \tilde{y}) - f(\boldsymbol{x}, y))$$

$$\partial_f l(\boldsymbol{x}, y, f) = \begin{cases} 0 \text{ if } f(\boldsymbol{x}, y) \geq 1 + f(\boldsymbol{x}, y^*) \\ k((\boldsymbol{x}, y^*), \cdot) - k((\boldsymbol{x}, y), \cdot) \text{ otherwise} \end{cases} \quad (8)$$

Here y^* is the maximizer of the $\max_{\tilde{y} \neq y}$ operation. If several y^* exist we pick one arbitrarily, *e.g.* by dictionary order. Note that when the number of classes is exactly two (binary classification) and $k((x, y), (x', y')) = \frac{yy'}{2}k(x, x')$ the loss function reduces to the well-known hinge loss.

Novelty Detection uses a trimmed version of the log-likelihood as a loss function. This means that labels are ignored and the one-class margin needs to exceed 1, leading to

$$l(\boldsymbol{x}, y, f) = \max(0, 1 - f(\boldsymbol{x}))$$

$$\partial_f l(\boldsymbol{x}, y, f) = \begin{cases} 0 \text{ if } f(\boldsymbol{x}) \geq 1 \\ -k(\boldsymbol{x}, \cdot) \text{ otherwise} \end{cases} \quad (9)$$

Table 1 summarizes the expansion coefficient(s) $\boldsymbol{\xi}_t$ arising from the derivative of the loss at time t.

Table 1. Gradient expansion coefficients.

task	expansion coefficient
Multiclass Classification	$\boldsymbol{\xi}_t = \mathbf{0}$ if $f_t(\boldsymbol{x}_t, y_t) \geq 1 + f_t(\boldsymbol{x}_t, y^*)$ $\xi_{t,y_t} = -1, \xi_{t,y^*} = 1$ otherwise
Novelty Detection	$\xi_t = \begin{cases} 0 & \text{if } f_t(\boldsymbol{x}_t) \geq 1 \\ -1 & \text{otherwise} \end{cases}$

3.3. Coefficient Updates

Since the update step 2.c in Algorithm 1 is not particularly useful in Hilbert space, we now rephrase it in terms of kernel function expansions. From (7) it follows that $g_t = \partial_f l(\boldsymbol{x}_t, y_t, f_t) + \lambda f_t$ and consequently

$$\begin{aligned} f_{t+1} &= f_t - \eta_t \left[\partial_f l(\boldsymbol{x}_t, y_t, f_t) + \lambda f_t\right] \\ &= (1 - \lambda\eta_t) f_t - \eta_t \partial_f l(\boldsymbol{x}_t, y_t, f_t). \end{aligned} \quad (10)$$

Using the initialization $f_1 = 0$ this implies that

$$f_{t+1}(\cdot) = \sum_{i=1}^{t} \sum_{y} \alpha_{tiy} k((\boldsymbol{x}_i, y), \cdot). \quad (11)$$

With some abuse of notation we will use the same expression for the cases where $\mathcal{H}$ is defined on $\mathcal{X}$ rather than $\mathcal{X} \times \mathcal{Y}$. In this setting we replace (11) by the sum over i only (with corresponding coefficients α_{ti}). Whenever necessary we will use $\boldsymbol{\alpha}_t$ to refer to the entire coefficient vector (or matrix) and α_{ti} (or α_{tiy}) will refer to the specific coefficients. Observe that we can write

$$g_t(\cdot) = \sum_{i=1}^{t} \sum_{y} \gamma_{tiy} k((\boldsymbol{x}_i, y), \cdot), \quad (12)$$

$$\text{where } \boldsymbol{\gamma}_t := \begin{bmatrix} \lambda \boldsymbol{\alpha}_{t-1} \\ \boldsymbol{\xi}_t^\top \end{bmatrix}. \quad (13)$$

We can now rewrite (10) using the expansion coefficients as

$$\boldsymbol{\alpha}_t = \begin{bmatrix} (1 - \lambda\eta_t)\boldsymbol{\alpha}_{t-1} \\ -\eta_t \boldsymbol{\xi}_t^\top \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_{t-1} \\ 0 \end{bmatrix} - \eta_t \boldsymbol{\gamma}_t. \quad (14)$$

Note that conceptually $\boldsymbol{\alpha}$ grows indefinitely as it acquires an additional row with each new data sample. Practical implementations will of course retain only a buffer of past examples with nonzero coefficients. If the loss function has a bounded gradient (as in all cases of Table 1) then the quality of the approximation increases exponentially with the number of terms retained [1], so good solutions can be obtained with limited buffer size.

3.4. Handling Offsets

In many situations, for instance in binary classification, it is advantageous to predict with $f(\cdot, y) + b_y$ where $f \in \mathcal{H}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{Y}|}$ is an offset parameter. While the update equations described above remain unchanged, the offset $\mathbf{b}$ is now adapted as well:

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \eta_t \partial_{\mathbf{b}} J_t(f_t + \mathbf{b}_t) = \mathbf{b}_t - \eta_t \boldsymbol{\xi}_t. \tag{15}$$

4. ONLINE SVMD

We now show how the SMD framework described in Section 2 can be used to adapt the step size for online SVMs. The updates given in Section 3 remain as before, the only difference being that the step size η_t is adapted before its value is used to update $\boldsymbol{\alpha}$.

4.1. Scalar Representation

When η is a scalar, (2) becomes

$$\eta_{t+1} = \eta_t \max(\tfrac{1}{2}, 1 - \mu \langle g_{t+1}, v_{t+1} \rangle), \tag{16}$$

where μ is the meta-step size described in Section 2. The update for v is now given by

$$v_{t+1} = \varrho v_t - \eta_t (g_t + \varrho \boldsymbol{H}_t v_t), \tag{17}$$

where $\boldsymbol{H}_t$ is the Hessian of the objective function. Note that now $\boldsymbol{H}_t$ is an operator in Hilbert space. For piecewise linear loss functions, such as (8), and (9), we have $\boldsymbol{H}_t = \lambda \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity operator, and obtain the simple update

$$v_{t+1} = (1 - \eta_t \lambda) \varrho v_t - \eta_t g_t. \tag{18}$$

4.2. Expansion in Hilbert Space

The above discussion implies that v can be expressed as a linear combination of kernel functions, and consequently is also a member of the RKHS defined by $k(\cdot, \cdot)$. Thus v cannot be updated explicitly, as is done in the normal SMD algorithm (Section 2). Instead we write

$$v_{t+1}(\cdot) = \sum_{i=1}^{t} \sum_{y} \beta_{tiy} k((\boldsymbol{x}_i, y), \cdot) \tag{19}$$

and update the coefficients $\boldsymbol{\beta}$. This is sufficient for our purpose because we only need to be able to compute the inner products $\langle g, v \rangle_{\mathcal{H}}$ in order to update η.

Analogous to the update on $\boldsymbol{\alpha}$ we can determine the updates on $\boldsymbol{\beta}$ via

$$\boldsymbol{\beta}_t = \begin{bmatrix} (1 - \eta_t \lambda) \varrho \boldsymbol{\beta}_{t-1} \\ 0 \end{bmatrix} - \eta_t \boldsymbol{\gamma}_t. \tag{20}$$

Although (20) suffices in principle to implement the overall algorithm, a naive implementation of the inner product $\langle g_t, v_t \rangle$ takes $O(t^2)$ time. In the next section we show how these updates can be performed in linear time.

4.3. Linear-Time Incremental Updates

We now turn to computing $\langle g_{t+1}, v_{t+1} \rangle$ in linear time by bringing it into an incremental form. We use the notation $f(\boldsymbol{x}_t, \cdot)$ to denote the vector of $f(\boldsymbol{x}_t, \tilde{y})$ for $\tilde{y} \in \mathcal{Y}$. Expanding g_{t+1} into $\lambda f_{t+1} + \boldsymbol{\xi}_{t+1}$ we can write

$$\langle g_{t+1}, v_{t+1} \rangle = \lambda \pi_{t+1} + \boldsymbol{\xi}_{t+1}^\top v_{t+1}(\boldsymbol{x}_{t+1}, \cdot), \tag{21}$$

where $\pi_t := \langle f_t, v_t \rangle$. The function update (10) yields

$$\pi_{t+1} = (1 - \lambda \eta_t) \langle f_t, v_{t+1} \rangle - \eta_t \boldsymbol{\xi}_t^\top v_{t+1}(\boldsymbol{x}_t, \cdot). \tag{22}$$

The v update (17) then gives us

$$\langle f_t, v_{t+1} \rangle = \varrho(1 - \lambda \eta_t) \pi_t - \eta_t \langle f_t, g_t \rangle, \tag{23}$$

and using $g_t = \lambda f_t + \boldsymbol{\xi}_t$ again we have

$$\langle f_t, g_t \rangle = \lambda \|f_t\|^2 + \boldsymbol{\xi}_t^\top f_t(\boldsymbol{x}_t, \cdot). \tag{24}$$

Finally, the squared norm of f can be maintained via:

$$\begin{aligned} \|f_{t+1}\|^2 = (1 - \lambda \eta_t)^2 \|f_t\|^2 \\ - 2\eta_t (1 - \lambda \eta_t) \boldsymbol{\xi}_t^\top f_t(\boldsymbol{x}_t, \cdot) \\ + \eta_t^2 \boldsymbol{\xi}_t^\top k((\boldsymbol{x}_t, \cdot), (\boldsymbol{x}_t, \cdot)) \boldsymbol{\xi}_t. \end{aligned} \tag{25}$$

The above sequence (21)–(25) of equations, including the evaluation of the associated functionals, can be performed in $O(t)$ time.

5. EXPERIMENTS

To show the utility of our approach we performed experiments on the the USPS data set following [1]. Since our approach is particularly useful when the data is non-stationary we create a data set where we use 4 letters from the data set (0-3) and split them in two classes. We create a data set of 1000 data points where the task at the first 500 is to classify between the 0 and 1 classes and at the second part of the data between the 2 and 3 classes. This sudden change of the distribution in the data although not usual illustrates the usefulness of the SVMD algorithm.

We compare SVMD to the conventional online SVM algorithm on binary classification, using a standard decay for the learning rate $\eta_t = \sqrt{\tau/(\tau + t)}$, where τ was tuned appropriately to obtain good performance. A Gaussian kernel width $\sigma = 10$ was used and the SVMD parameters where set to $\mu = 0.1$ and $\rho = 1$ for this experiment.

We plot the value of the average error rate during the iterations in Figure 1 as well as the value of η_t in Figure 2. In Figure 1 we can see that SVMD outperforms the online SVM algorithm during the initial 500 stationary iterations and adapts well to the sudden change in the distribution of the data outperforming the online SVM.

SVMD adapts the values of the learning rate η_t quite well to the changes in the data set. Although both algorithms start with an initial value of $\eta_t = 1$ in the case of SVMD η is raised at the initial iterations providing slightly better performance and then decays until halfway through the data where in response to the change of the distribution η incrises again. This shows that our method is well suited for learning non-stationary distributions.

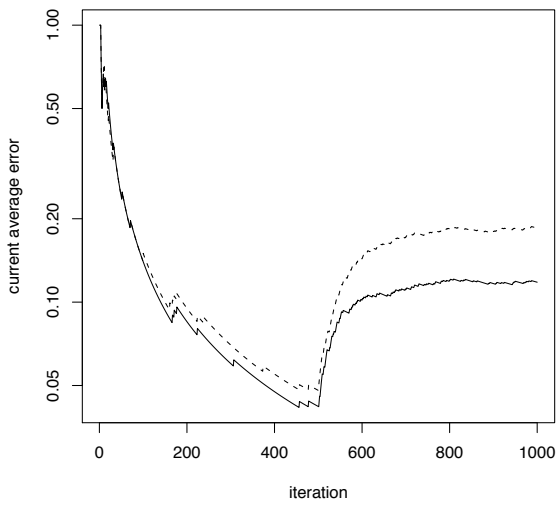

Fig. 1. Average error rate (on a log scale) incurred over a single run through the digits 0 and 1 until iteration 500 and 2, 3 from 500 onwards of the USPS data set, for SVMD (solid) *vs.* online SVM with scheduled step size decay (dashed). SVMD performs better throughout the data.

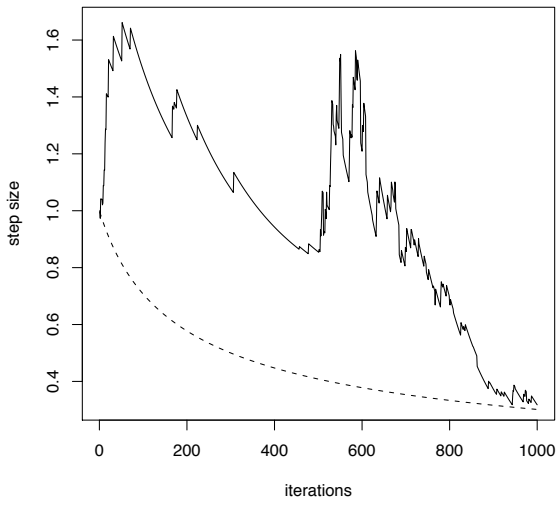

Fig. 2. Step size for classification over a single run of the USPS data set where the first half of the data set are zeros and ones and the second half twos and threes. The dased line is the sheduled step size of the online SVM. Observe how the SVMD values (solid) of η_t respond to the change in the dataset.

6. DISCUSSION

We presented online SVMD, an extension of the SMD step size adaptation method to the kernel framework. Using an incremental approach to reduce a naive $O(t^2)$ computation to $O(t)$, we showed how the SMD parameters can be updated efficiently even though they now reside in an RKHS. In experiments online SVMD outperformed the online SVM with scheduled step size decay.

Adaptive step size control is clearly most useful when faced with a data stream exhibiting irregular nonstationarities that thwart less reactive approaches, such as fixed decay schedules. We expect SVMD to become the first viable online kernel algorithm for such data.

Future research will focus on the use of online learning techniques to prove worst case loss bounds for SVMD.

Acknowledgments

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts, and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work is supported by the grants of the ARC as well as the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778.

References

[1] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, Aug 2004.

[2] Nicol N. Schraudolph, "Fast curvature matrix-vector products for second-order gradient descent," *Neural Computation*, vol. 14, no. 7, pp. 1723–1738, 2002.

[3] Nicol N. Schraudolph, "Local gain adaptation in stochastic gradient descent," in *Proceedings of the International Conference on Artificial Neural Networks*, Edinburgh, Scotland, 1999, pp. 569–574, IEE, London.

[4] Nicol N. Schraudolph and Xavier Giannakopoulos, "Online independent component analysis with local learning rate adaptation," in *Neural Information Processing Systems*, S. A. Solla, T. K. Leen, and K.-R. Müller, Eds., Vancouver, Canada, 2000, vol. 12, pp. 789–795, MIT Press.

[5] Michele Milano, *Machine Learning Techniques for Flow Modeling and Control*, Ph.D. thesis, Eidgenössische Technische Hochschule (ETH), Zürich, Switzerland, 2002.

[6] Matthieu Bray, Esther Koller-Meier, Pascal Müller, Luc Van Gool, and Nicol N. Schraudolph, "Stochastic optimization for high-dimensional tracking in dense range maps," *IEE Proceedings Vision, Image & Signal Processing*, to appear 2005.

[7] Andreas Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

[8] S. V. N. Vishwanathan, N. N. Schraudolph, and A. J. Smola, "Online SVM with multiclass classification and SMD step size adaptation," Tech. Rep., National ICT Australia, Canberra, Australia, June 2005, `http://users.rsise.anu.edu.au/~vishy/papers/VisSchSmo05.pdf`.