Improved Working Set Selection for LaRank

Matthias Tuma[1,⋆] and Christian Igel[2]

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
matthias.tuma@rub.de
[2] Department of Computer Science, University of Copenhagen, Denmark
igel@diku.dk

Abstract. LaRank is a multi-class support vector machine training algorithm for approximate online and batch learning based on sequential minimal optimization. For batch learning, LaRank performs one or more learning epochs over the training set. One epoch sequentially tests all currently excluded training examples for inclusion in the dual optimization problem, with intermittent *reprocess* optimization steps on examples currently included. Working set selection for one reprocess step chooses the *most violating pair* among variables corresponding to a random example. We propose a new working set selection scheme which exploits the gradient update necessarily following an optimization step. This makes it computationally more efficient. Among a set of candidate examples we pick the one yielding *maximum gain* between either of the classes being updated and a randomly chosen third class. Experiments demonstrate faster convergence on three of four benchmark datasets and no significant difference on the fourth.

1 Introduction

Support vector machines (SVMs, e.g. [1]) have attractive theoretical properties and give good classification results in practice. Training times between quadratic and cubic in the number of training examples however impair their applicability to large-scale problems. Over the past years well-performing *online* variants of binary and multi-class SVM solvers have been proposed and refined [2,3,4,5,6]. Online SVMs can be preferable to standard SVMs even for batch learning. On large datasets that prohibit calculating a close to optimal solution to the SVM optimization problem, online SVMs can excel in finding good approximate solutions quickly. The prominent online multi-class SVM LaRank was introduced by Bordes, Bottou, Gallinari, and Weston [4]. It relies on the multi-class SVM formulation proposed by Crammer and Singer (CS, [7]). We refer to LaRank-like solvers [2,4] as *epoch-based* since they complete one or more epochs over a training set, aiming at well-performing hypotheses after as few as a single epoch.

When using universal, non-linear kernels – which are a prerequisite for consistency, cf. [8] – sequential minimal optimization (SMO, [9]) solvers are the method of choice for SVM training. Their time requirements strongly depend

⋆ M.T. acknowledges a scholarship by the German National Academic Foundation.

A. Berciano et al. (Eds.): CAIP 2011, Part I, LNCS 6854, pp. 327–334, 2011.

on stopping criteria, working set selection, and implementation details such as kernel cache organization, shrinking, etc. This paper focuses on an improvement to working set selection for SMO steps in LaRank.

Section 2 formally introduces the CS machine. We give a general definition of epoch-based CS solvers and restate the LaRank algorithm [4]. One of LaRank's building blocks is the random selection of examples for *reprocess*-type optimization steps. Random example selection provides the advantage of a constant time operation, however at the risk of conducting an optimization step yielding little gain for the overall problem. We propose an alternative example selection scheme which is both gain-sensitive and can be carried out fast enough to speed up the overall convergence of dual, primal, and test error. Empirical evaluations are presented in Section 3, followed by our conclusions and outlook in Section 4.

2 Multi-class SVMs

Consider an input set X, an output set $Y = \{1, \ldots, d\}$, a labeled training set $S_N = \{(x_i, y_i)\}_{1 \leq i \leq N} \in (X \times Y)^N$ of cardinality N, and a Mercer kernel [1] function $k : X \times X \to \mathbb{R}$. Then a trained all-in-one multi-class SVM without bias assigns to an example $x \in X$ the output class label

$$h(x) = \arg\max_{y \in Y} \sum_{i=1}^{N} \beta_i^y k(x, x_i) \ . \tag{1}$$

Training the SVM is equivalent to determining the parameter vector $\beta \in \mathbb{R}^{dN}$. Its component β_i^y constitutes the contribution of example i to the kernel expansion associated with class y. Iff $\exists y \ \ \beta_i^y \neq 0$, we say that i or β_i is a support pattern, and iff $\beta_i^y \neq 0$, we say that y or β_i^y is a support class for sample i.

2.1 Crammer-Singer Type Multi-class SVMs

Following the notation in [4] and given a regularization parameter $C \in \mathbb{R}^+$, Crammer-Singer type multi-class SVMs [7] determine the parameter vector β by solving the dual optimization problem

$$\max_{\beta} \qquad \sum_i \beta_i^{y_i} - \frac{1}{2} \sum_{i,j} \sum_y \beta_i^y \beta_j^y k(x_i, x_j) \tag{2}$$

$$\text{s.t.} \qquad \forall i \ \ \forall y \ \ \beta_i^y \leq C\delta(y, y_i) \tag{3}$$

$$\forall i \ \ \sum_y \beta_i^y = 0 \ , \tag{4}$$

with Kronecker delta δ. The derivative of (2) w.r.t. the variable β_i^y is given by

$$g_i^y = \delta(y, y_i) - \sum_j \beta_j^y k(x_i, x_j) \ . \tag{5}$$

Two notable consequences arise from the specific form of problem (2). Constraint (4) in practice restricts SMO solvers to working sets of size two with both working variables corresponding to the same training example. This closely links working set selection to example selection. Second, because the quadratic part of problem (2) is entirely composed of diagonal sub-matrices, altering a variable β_i^c only propagates through to gradients g_j^c involving the same class label.

2.2 Epoch-Based Crammer-Singer Solvers

We understand an epoch-based Crammer-Singer (EBCS) solver to carry out optimization epochs over a training set according to Alg. 1. Specific variants of EBCS solvers are then realized through different implementations of the sub-algorithms $\mathrm{WSS}_{\mathrm{new}}$, $\mathrm{WSS}_{\mathrm{rep}}$, and R. These control working set selection for (i) non-support patterns and (ii) support patterns, as well as (iii) the relative ratio between optimization steps on non-support and support patterns, respectively. All three will in practice depend on the joint state of both solver and solution. Note that we understand the SMO steps in lines 7 and 11 of Alg. 1 to potentially leave β unaltered, for example if both variables are actively constrained.

Algorithm 1. Epoch-based Crammer-Singer solver

Input: training set S_N, epoch limit e_{max}, working set selection algorithms $\mathrm{WSS}_{\mathrm{new}}$ and $\mathrm{WSS}_{\mathrm{rep}}$, step selection algorithm R

```
1  β ← 0
2  for e ← 1 to e_max do                                   // 1 loop = 1 epoch
3  |  Shuffle training set S_N jointly with β
4  |  for i ← 1 to N do                                    // 1 loop = 1 sample
5  |  |  if ∀ y  β_i^y = 0 then                           // process new sample
6  |  |  |  Choose (c, e) ∈ Y^2 according to WSS_new
7  |  |  |  SMO-step on (β_i^c, β_i^e) and gradient update
8  |  |  |  while not R do                          // reprocess old samples
9  |  |  |  |  Choose (j, c, e) ∈ {1, ..., N} × Y^2 according to WSS_rep
10 |  |  |  |  if ∃y  β_j^y ≠ 0 then
11 |  |  |  |  |  SMO-step on (β_j^c, β_j^e) and gradient update
```

LaRank. The popular EBCS solver LaRank [4] in its query to $\mathrm{WSS}_{\mathrm{rep}}$ (Line 9 of Alg. 1) chooses the example index j randomly. For both $\mathrm{WSS}_{\mathrm{rep}}$ and $\mathrm{WSS}_{\mathrm{new}}$, the class indices (c, e) are selected according to the *most violating pair* heuristic (MVP, [10]) on the given example. In addition, $\mathrm{WSS}_{\mathrm{rep}}$ operates in two different modes, $\mathrm{WSS}_{\mathrm{old}}$ and $\mathrm{WSS}_{\mathrm{opt}}$, which perform MVP among all classes or all support classes of one example, respectively. The resulting three step variants *processNew*, *processOld*, and *processOpt* are chosen from in a stochastic manner. Their probabilistic weights are adapted through three slowly relaxing linear dynamical systems with attractors to the current dual gain rate of each step variant. For alternative, deterministic step selection schemes also see [2,5,6].

Algorithm 2. Gain-sensitive working set selection for LaRank

```
2  SMO-step on (β_i^c, β_i^e)
4  (w_max, v) ← (0, ∅)
6  Pick random class t,   t ∉ {c, e}
8  for (j, y) : β_j^y ≠ 0,  y ∈ {c, e} do          // loop through support classes
9  |  Update (g_j^y)
10 |  if β_j^t ≠ 0 then
11 |  |  w ← clipped SMO-Gain(β_j^y, β_j^t)
12 |  |  if w > w_max then                          // found new best candidate
13 |  |  |  (w_max, v) ← (w, j)
14 if w_max = 0 then                                // fallback to random
15 |  v ← index of random support pattern
16 Provide example v upon next call to WSS_rep
```

Gain-sensitive working set selection. LaRank and its binary predecessor LaSVM [2] are inspired by perceptron-like kernel machines. As such, the random traversal of hitherto excluded training samples around line 4 of Alg. 1 is conceptually well-founded. Since first and second order working set selection coincide for CS, MVP can further be seen as a viable approximation to clipped gain working set selection [11,3]. Another relevant building block of EBCS solvers is the example selection procedure for $\mathrm{WSS}_{\mathrm{rep}}$. A naive deterministic alternative to LaRank's random selection scheme would be to compute the full $\arg\max_i \arg\max_{(c,e)}$ of the clipped or unclipped gain. Yet, the computational effort outburdens the potential gain, especially if, as for original LaRank, not all gradients are being cached. The LaRank algorithm with minimal cost and random gain can thus be seen as lying on one end of all possible example selection methods and the argmax-scheme with maximum cost and maximum gain on the other. This paper explores the question whether the already well-performing LaRank algorithm can be further improved by an example selection scheme for which the added cost (relative to instant example selection) is outweighed by the gain advantage received in turn (relative to the average gain of MVP on random examples).

We propose to exploit the gradient update necessarily following each SMO step to select the next "old" example. Similar to [11], reusing information recently computed promises efficient working set selection. Let (β_i^c, β_i^e) be the pair of variables altered by the last SMO step. Then, according to (5), the subset of all gradients $\{g_j^c, g_j^e\}_{1 \le j \le N}$ currently stored by the solver must be updated. For LaRank these are all $g_j^y,\ y \in \{c, e\}$, for which $\beta_j^y \neq 0$. As the solver looks at this subset in any case, it suggests itself to select the next old example according to some property of all gradients being updated. We propose as such a property the clipped gain achievable by a SMO step between the variable β_j^y the gradient g_j^y of which is being updated and, fixed within each update loop, a random third class t. If β_j^t is not a support class, it is not considered. Alg. 2 summarizes the resulting example selection procedure following both SMO steps in lines 7 and 11 of Alg. 1. If no feasible pair can be identified, a fallback to a random sample is guaranteed. In practice, this only occurs in the first few iterations. After Alg. 2,

Table 1. Datasets, SVM hyperparameters, and average reprocess step rates

	Train Ex.	Test Ex.	Classes	Features	C	$k(x,z)$	s_{old}	s_{opt}
USPS	7291	2007	10	256	10	$e^{-0.025(x-z)^2}$	1.94	36.7
LETTER	16000	4000	26	16	10	$e^{-0.025(x-z)^2}$	1.75	82.6
INEX	6053	6054	18	167295	100	$x \cdot z$	3.67	35.1
MNIST	60000	10000	10	780	1000	$e^{-0.02(x-z)^2}$	1.65	53.4

a call to WSS_{opt} will directly return (β_v^y, β_v^t), while a call to WSS_{old} returns the MVP within the candidate example v. In the rare case that the latter does not yield a feasible variable pair, we also choose a random example in the next step.

Compared to the original version, Alg. 2 adds the computational burden of checking whether $\beta_j^t = 0$ for all examples for which $\beta_j^y \neq 0$, $y \in \{c, e\}$. For those examples for which $\beta_j^t \neq 0$ we say that we have a *hit* between class y and t. For every hit the potential gain of a SMO step on (β_j^y, β_j^t) has to be calculated and compared to the current maximum candidate. Because for each class the support classes lie sparse in the total set of support patterns, the gain calculation is only conducted in a fraction of update steps. Yet still, experiments not documented here indicate that Alg. 2 does not typically improve upon LaRank. Since Alg. 2 is carried out after each SMO step, the resulting constant time cost propagates through to all three average gain rates which steer the stochastic step selection procedure. The added time is negligible for the more costly step types *processNew* and *processOld*, but large enough to make the selection of *processOpt* significantly more unlikely. This in turn impedes the removal of useless support patterns, which again makes update steps more costly.

We reduce the computational cost by entering candidate examination at line 10 of Alg. 2 only for a subset of all variables being updated. In detail, we introduce a parameter D representing the desired number of hits within the entire update loop. Starting from a random index we only enter candidate examination at line 10 while less than D hits have occurred. Note that the best of D hits with probability $1 - x^D$ is better or equal to the best in a fraction x of all possible hits (e.g., theorem 6.33 in [1]). We choose $D = 10$, for which the probability of the best of D random hits being in the highest quintile of all possible hits is $\sim 90\%$, and divide these ten hits evenly between the two classes being updated. We further provide an incentive towards sparser solutions and hence shorter gradient update times by slightly modifying line 12 of Alg. 2. If a SMO step on a candidate hit would eliminate at least one of the two support classes, that step is given preference over a non-eliminating candidate step. Between candidates of identical priority the resulting gain remains the selection criterion, just as stated in line 12 of Alg. 2. For brevity we refer to this final algorithm employing gain sensitive example selection in LaRank reprocess steps as "GaLa".

3 Experiments

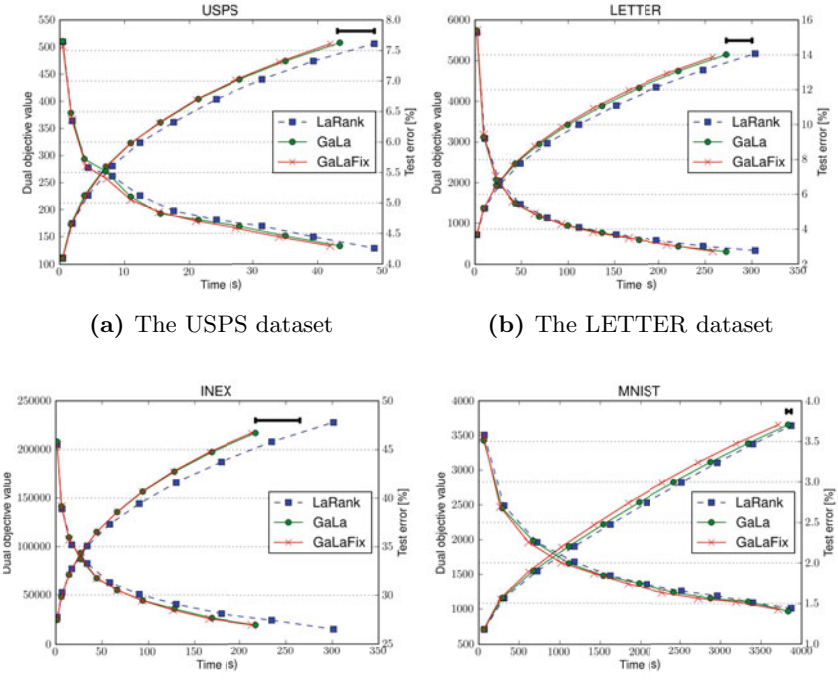
We incorporated GaLa into the original LaRank implementation and conducted our comparison on the original benchmark datasets, both obtainable at the software page of [4]. Tb. 1 lists the corresponding dataset characteristics and SVM hyperparameters.[1] We further wish to rule out that our results are merely an artifact of GaLa nudging the stochastic step selection mechanism to for some reason more suitable relative step rates. We therefore besides LaRank and GaLa considered a third variant for comparison, GaLaFix, in which we fixed the average number of *processOld* and *processOpt* steps per *processNew* in GaLa to those exhibited by LaRank. In detail, we for each dataset simultaneously let one LaRank and two dummy GaLa runs perform ten independent single-epoch trials and noted the average relative step rates $(s_{\text{old}}, s_{\text{opt}})$ of LaRank in Tb. 1. In the actual experiments we compare GaLaFix, clamped to these empirical step rates, to LaRank and GaLa, afterwards verifying that LaRank approximately reached the same step rates again. Fig. 1 shows the results obtained as mean averages over ten independent single-epoch trials on differently shuffled training sets. We for clarity excluded the primal training curves, which qualitatively follow those of the test errors. The horizontal black bar in the upper right of each plot illustrates the factual time advantage of GaLa over LaRank. It extends from the finish time of that method with lower final dual value to the linearly extrapolated time at which the respective other method reached the same dual value. Dividing the length of the line by the time of its later endpoint we note a speed-up of 12, 9, 18, and 2 percent for USPS, LETTER, INEX, and MNIST, respectively. Experiments were carried out on an eight-core 2.9 GHz machine with 8 MB CPU cache, 3.5 GB memory, using 500 MB of which as kernel cache, and no other avoidable tasks running besides all three methods in parallel.[2]

[1] As SVM hyperparameters were selected on the basis of "past experience" [4], the dual curve should probably be seen as most significant performance measure. Further, MNIST data and hyperparameters slightly vary between the printed and website version of [4], which we used and where the relevant differences are listed. We also slightly modified the LaRank implementation for training set shuffling, serialization, etc. The entire source code underlying our experiments can be obtained at `http://image.diku.dk/igel/downloads.php`. Besides the implementation described above, we added a complete re-implementation of an EBCS solver to the Shark machine learning library [12]. In that implementation one epoch of GaLa on MNIST on average reaches a dual of 3656 in 987 seconds, despite not speeding up sparse radial basis function kernel evaluations through precomputed norms as in the original.

[2] Similar to the note on the LaSVM software page [2] we observed performance variability across platforms also for LaRank. We ascribe this effect to the volatility of the step selection algorithm. E.g., if kernel computations are slightly *faster* on one machine, this will make *processNew* and *processOld* steps more likely, but might lead to an actual *decrease* in accuracy if the relative advantage for *processNew* is higher. Further, if the operation underlying gradient updates takes longer on one machine, this constant cost on all three step types will regularize the original step selection mechanism. In [5] the adaptive step selection mechanism is discarded for a deterministic one, at the cost of introducing an additional SVM hyperparameter.

3.1 Results and Discussion

For the first three datasets the proposed method arrives at the same dual values between 9% and 18% faster than the original approach. For the fourth dataset, MNIST, it only yields a marginal advantage of 2%. Possible reasons for this comparatively weak performance may be that the distribution of gradients is such that randomly picking an example holds no real disadvantage as compared to a gain-sensitive selection method. We also conducted minor experiments not documented here towards the role of the parameter D, but did not find qualitatively different results for reasonable changes in D. Third, it is notable that until around 2300 seconds, GaLaFix persistently sustains an advantage of 10 to 15% over LaRank. It might be enlightening to relate the subsequent decline to the onset of the kernel cache overflow, since that would most likely significantly perturb the target attractor for the probabilistic weight of *processOld* steps. This however is not straightforward as LaRank uses d class-wise kernel caches.



(a) The USPS dataset **(b)** The LETTER dataset

(c) The INEX dataset **(d)** The MNIST dataset

Fig. 1. Development of dual objective (left axis) and test error (right axis) of LaRank, GaLa, and GaLaFix over one epoch on four benchmark datasets

4 Conclusions

We proposed a gain-sensitive working set selection algorithm for LaRank by Bordes et al. [4], which is an epoch-based solver for the Crammer-Singer multi-class SVM [7]. Our new working set selection scheme improves learning speed and is conceptually compatible with a wide range of conceivable step selection procedures. While several approaches to step selection have been presented [2,4,5,6], a robust canonical solution has yet to be developed. We further believe that the method suggested here is a promising basis for parallelizing *processOpt* steps in LaRank. Since SMO steps and subsequent gradient updates are independent for disjunct class pairs, $d/3$ parallel SMO steps should with slight modifications be possible while still benefiting from gain-sensitive example selection.

References

1. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
2. Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. Journal of Machine Learning Research 6, 1579–1619 (2005), http://leon.bottou.org/papers/bordes-ertekin-weston-bottou-2005
3. Glasmachers, T., Igel, C.: Second order SMO improves SVM online and active learning. Neural Computation 20(2), 374–382 (2008)
4. Bordes, A., Bottou, L., Gallinari, P., Weston, J.: Solving multiclass support vector machines with LaRank. In: Proceedings of the 24th International Conference on Machine Learning, pp. 89–96. OmniPress (2007), http://www-etud.iro.umontreal.ca/~bordesa/mywiki/doku.php?id=larank
5. Bordes, A., Usunier, N., Bottou, L.: Sequence labelling SVMs trained in one pass. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 146–161. Springer, Heidelberg (2008)
6. Ertekin, S., Bottou, L., Giles, C.: Non-convex online support vector machines. IEEE Transactions on Pattern Recognition and Machine Intelligence 33(2), 368–381 (2011)
7. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research 2, 265–292 (2002)
8. Steinwart, I.: Support vector machines are universally consistent. Journal of Complexity 18, 768–791 (2002)
9. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods: Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
10. Joachims, T.: Making large-scale SVM learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods: Support Vector Learning, pp. 169–184. MIT Press, Cambridge (1999)
11. Glasmachers, T., Igel, C.: Maximum-gain working set selection for support vector machines. Journal of Machine Learning Research 7, 1437–1466 (2006)
12. Igel, C., Glasmachers, T., Heidrich-Meisner, V.: Shark. Journal of Machine Learning Research 9, 993–996 (2008), http://shark-project.sourceforge.net