MIT-CSAIL-TR-2012-003
CBCL-306

January 31, 2012

GURLS: a Toolbox for Regularized Least Squares Learning

Andrea Tacchetti, Pavan S. Mallapragada, Matteo Santoro, and Lorenzo Rosasco

GURLS: a Toolbox for Regularized Least Squares Learning

Andrea Tacchetti$^\diamond$, Pavan K. Mallapragada$^\diamond$,Matteo Santoro$^\diamond$,§, Lorenzo Rosasco$^\diamond$,§
$^\diamond$*CBCL, McGovern Institute for Brain Research, MIT, Cambridge (MA) USA*
§ *IIT@MIT Lab, Istituto Italiano di Tecnologia, Genova, Italy*
`atacchet,pavanm msantoro, lrosasco@mit.edu`

January 22, 2012

Abstract

We present GURLS, a toolbox for supervised learning based on the regularized least squares algorithm. The toolbox takes advantage of all the favorable properties of least squares and is tailored to deal in particular with multi-category/multi-label problems. One of the main advantages of GURLS is that it allows training and tuning a multi-category classifier at essentially the same cost of one single binary classifier.

The toolbox provides a set of basic functionalities including different training strategies and routines to handle computations with very large matrices by means of both memory-mapped storage and distributed task execution. The system is modular and can serve as a basis for easily prototyping new algorithms. The toolbox is available for download, easy to set-up and use.

1 Introduction

The Regularized Least Squares (RLS) algorithm is one of the simplest algorithms for supervised learning and yet it has been reported to consistently achieve state of the art performances in a large number of learning tasks [8]. Based on solid theoretical foundations, RLS has connections with estimation, Gaussian processes [6] and Fisher discriminant analysis [2]. From a computational point of view it reduces to solving a linear system (in fact several, if parameter tuning is needed), and this fact allows to exploit the latest mathematical, as well as software, tools in numerical linear algebra. In particular, this allows to handle large, and potentially massive, data-sets. The above observations make RLS classification an interesting alternative to other learning methods such as SVM. The software we present in this paper is an easy to use toolbox that fully exploits all the favorable properties of least squares.

RLS can naturally handle both scalar and multi-output supervised learning problems. The GURLS toolbox, whose name is an acronym for Grand Unified Regularized Least Squares, has been specifically tailored to solve multi-category classification problems and related extensions, such as multi-label classification, where one input can belong to more than one category. In this work we consider a one vs all (OVA) strategy. Even though more complex approaches have been proposed, the OVA strategy is often reported to perform as well as more sophisticated methods [7]. While the computational complexity of training OVA classifiers with other regularization algorithms, such as support vector machines (SVM) or regularized logistic classification, typically grows linearly in the number of categories, the training complexity of OVA with RLS is *independent* on the number of classes [7]. While RLS does not produce sparse solutions like SVMs, its properties make it competitive to SVM. Computing the classifiers corresponding to different regularization parameter values does not increase the computational complexity with respect to training a single classifier, so that parameter selection is essentially free. The output of a RLS classifier on a test point is proportional to the conditional probability of the sample belonging to a certain class. Like SVMs, RLS allows for easy implementation of primal and dual formulation of the optimization problem. The linear system induced by RLS can be solved with different batch strategies, as well as incremental and online approaches. Above all, simplicity often makes RLS an ideal benchmark in a variety of learning applications.

GURLS is a toolbox that takes full advantage of all the above properties. The package is available for download at `http://cbcl.mit.edu/gurls` and is developed entirely in MATLAB, providing an easy to use tool to quickly prototype solutions for supervised learning applications. A basic C++ version is also available and will be further developed. The package includes utilities that allow for automatic optimal parameter selection, fast training of multiclass classifiers with a variety of coding schemes, testing of the predictor function on a new test set as well as routines to compute and display various performance measures. All these functionalities are wrapped in a common and intuitive interface that makes GURLS a modular package which is fast, easy to use and trivial to expand. The rest of the paper is organized as follows: in Section (2) we briefly recall some facts and definitions about RLS Classification, in the following section we provide some details on the implementation and software engineering of the toolbox itself and in the last Section we present some experiments to show the performance of RLS classification on publicly available datasets.

2 Basic Background on RLS

We recall some basic facts on RLS and refer to [8] for further details and references. We focus our discussion on classification.

Given a training set of input-output pairs $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, T\}$ for $i = 1, \ldots, n$, and a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ induced by a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$, RLS corresponds to Tikhonov regularization with the square loss. Let $\mathbf{K}$ the $n \times n$ matrix with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{Y}$ the $n \times T$ output matrix whose entries Y_{ij} are equal to 1 if the i^{th} training example belongs to the j^{th} class and -1 otherwise. The associated optimization problem is

$$\min_{\mathbf{C} \in \mathbb{R}^{n \times T}} \left\{ \frac{1}{n} \|\mathbf{Y} - \mathbf{K}\mathbf{C}\|_F^2 + \lambda \mathbf{C}^T \mathbf{K} \mathbf{C} \right\}, \tag{1}$$

where $\| \cdot \|_F$ if the Frobenius norm. The optimal solution $\mathbf{C}^*$ to the above problem can be computed in closed form, $\mathbf{C}^* = (\mathbf{K} + \lambda n I)^{-1}\mathbf{Y}$. If we consider a linear model and let $\mathbf{X}$ be the $n \times d$ matrix whose i^{th} row is $\mathbf{x}_i{}^T$, the problem becomes,

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times T}} \left\{ \frac{1}{n} \|\mathbf{Y} - \mathbf{X}\mathbf{W}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 \right\} \tag{2}$$

and its minimizer is

$$\mathbf{W}^* = (\mathbf{X}^T\mathbf{X} + \lambda n I)^{-1}\mathbf{X}^T\mathbf{Y}. \tag{3}$$

These two formulations are equivalent when we use a linear kernel and therefore the problem's complexity depends on the smallest between n and d.

Algorithm 1: Stochastic gradient descent algorithm

```
Initialization: $\lambda > 0; \tau_0 = \lceil \|\mathbf{x}_0\|_\infty / \lambda \rceil$
for $\tau = 1, 2, \dots, t$ do
    $\eta = 1/[\lambda(\tau + \tau_0)]$;
    $\mathbf{W}_\tau = (1 - \lambda\eta)\mathbf{W}_{\tau-1} + \eta \mathbf{x}_\tau (\mathbf{Y}_\tau - \mathbf{x}_\tau^T \mathbf{W}_{\tau-1})$;
    if $\|\mathbf{W}_\tau\| > \sqrt{T/\lambda}$ then
        $\mathbf{W}_\tau = \frac{\mathbf{W}_\tau}{\|\mathbf{W}_\tau\|}\sqrt{T/\lambda}$;
    end
end
$\mathbf{W} = \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{W}_\tau$
```

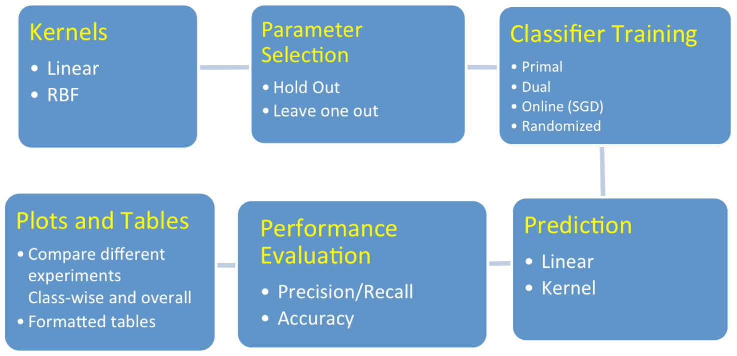
2.1 Training and Tuning RLS

A fast and numerically stable method to compute the solution to (1) and (2) is to solve the linear system after doing a Cholesky decomposition of the regularized kernel matrix. Such decomposition requires $O(\min(n, d)^3)$ operations and the subsequent matrix multiplication has complexity $O(Tn^2)$ for OVA and $O(n^2)$ for the binary case (for which the $\mathbf{Y}$ matrix only has one column). Since $n > T$, (i.e. several examples per class) the leading term remains unchanged: training a OVA T-category classifier has essentially the same complexity as training a single binary classifier. The generalization ability of the classifier depends on the choice of the regularizer λ. For RLS, training several classifiers by changing λ, has roughly the same order of complexity as computing a single solution to (1). To see why this is true, it is sufficient to reformulate the inversion of $\mathbf{G} = (\mathbf{K} + \lambda n I)$ in terms of the eigen-decomposition of $\mathbf{K} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. We can write: $(\mathbf{K} + \lambda n I)^{-1}\mathbf{Y} = \mathbf{Q}(\mathbf{\Lambda} + \lambda n I)^{-1}\mathbf{Q}^T\mathbf{Y}$. Hence, the solutions corresponding to different regularization parameter values can be easily computed once the eigen-decomposition of the kernel matrix is available.

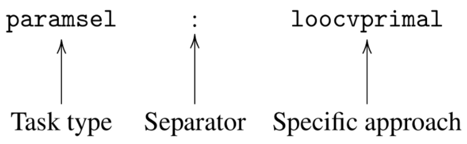
Both the Cholesky decomposition of a matrix and its eigen-decomposition require $O(n^3)$ operations, thus the leading term of the problem complexity remains unchanged [8]. In fact, even though the constant for the eigen-decomposition is worse, we have observed in practice that peforming single eigen-decomposition may be preferable to computing the Cholesky decomposition many times. Further, computations can be sped up considering a randomized singular value decomposition of the data matrix[1]. The above observation allows us to perform training and tuning of RLS at essentially the same computational cost of a binary classifier with fixed regularization parameter. This fact is one of the main advantages of RLS over other learning algorithms such as SVM for which the regularization path is hard to obtain [4].

Besides the above numerical routines the linear system associated with the RLS problem can be solved considering other methods. GURLS features a recursive training scheme, based on the rank-one update of the Cholesky decomposition [9], that allows to update the solution when a new point becomes available (without re-training on the whole data-set). Moreover GURLS is provided with an implementation of a stochastic gradient descent algorithm (pseudo-code in Algorithm 1), related to the PEGASOS algorithm [10] and can therefore deal with scenarios where n and d are so large that one cannot store in memory neither an n-by-n nor a d-by-d matrix.

[1] `http://cims.nyu.edu/~tygert/software.html`



(a) *A schematic view of the most important components in the GURLS pipeline.*



(b) *Task specification format.*

3 The GURLS Toolbox

In this section we illustrate some implementation details and design patterns that were employed while building the GURLS Toolbox. This information is targeted to the end user as well as to a potential developer, interested in expanding the GURLS package. We further refer to the documentation on the website for details and examples.

The GURLS Toolbox is primarily a collection of functions which share a common I/O signature. A standard computational learning pipeline can be broken down into a number of simple tasks which can in turn be completed using different methods. While designing the GURLS toolbox we singled out eight task categories (pre-processing, data-splitting, computation of the kernel matrix, model-selection, classifier training, classifier prediction on a test set, performance assessment, visualization and comparison) and organized the toolbox accordingly (see Fig. 1(a)). The function names are composed of two parts, separated by an underscore, the first part indicates the type of task the function is meant to carry out, while the second specifies which method is used to complete the computation. All functions that belong to a given category can be interchanged with each other so that the user can easily decide how each task is performed. All tasks are executed, in the order specified by the user, via an interface function (using the convention presented in Fig. 1(b)) , options and results are automatically stored in a single structure; this allows the user to easily skip some steps and include the desired results as options.

3.1 Large datasets: bGURLS and GDM

The GURLS package features a number of routines and a separate interface function to deal with large datasets. These methods, which constitute the bGURLS Toolbox, use an ad-hoc implementation of a memory-mapped file as core data structure to handle large data sets. This data structure, called *bigarray*, extends a MATLAB array and allows to handle data matrices as large as a machine's hard drive (instead of its central memory). Furthermore, *bigarrays* allow for parallel and distributed computations via the internally developed simple *GURLS' distributed manager* (GDM) which provides a very simple interface to distribute matrix-matrix multiplications.

4 Experiments

In this section we present experiments to compare the performance of the GURLS package with popular implementations of SVM based learning solutions on publicly available datasets. While a thorough comparison of RLS classification and SVMs remains well beyond the scope of this paper, we wish to show how the favourable properties of RLS often lead, in practice, to comparable performance with significantly less computational burden. We considered two datasets. The MNIST dataset[2], composed of 60000 training images of digits 1 to 10 and 10000 corresponding test images. For this dataset we considered a linear kernel using raw pixel values as features. The second is the *PubFig83* dataset[3]. This dataset contains images of real-world faces. For this dataset we considered a linear kernel computed on a set of biologically inspired features [5] as well as the Gaussian kernel on the same features.

Package	Kernel	Accuracy	Time
MNIST			
GURLS	Linear	84.8%	1m31s
LIBLNEAR	Linear	84.5%	3m55s
PubFig83			
GURLS	Linear	86%	00h53m
GURLS	Gaussian	88%	24h40m
LIBSVM	Linear	76%	05h20m
LIBSVM	Gaussian	76%	05h36m

Table 1: Benchmark of classification accuracy and training time on two publicly available datasets.

We compared RLS to SVM for which we used the MATLAB interface to LIBLINEAR [3] and the SHOGUN [11] python modular interface to LIBSVM [1]. Results are reported in Table 4 and refer to the average classification accuracy over classes.

For SVM, all parameters were set to default values while for RLS we performed parameter tuning on a validation set (80% training and 20% validation for both MNIST and PubFig83). Note that this was done to keep training times comparable. As expected RLS remains faster even though it performs model selection. When using the Gaussian kernel the complexity grows linearly in the number of considered Gaussian widths σ. It is worth noting that a simple heuristic choice of σ (e.g. the 25th percentile of the non-zero pairwise distances between training points) followed by a fine tuning of λ gives only slightly worse results.

Acknowledgements

We would like to thank Tomaso Poggio and the rest of CBCL for useful comments and insights. This report describes research done at the Center for Biological & Computational Learning, which is in the McGovern Institute for Brain Research at MIT, as well as in the Dept. of Brain & Cognitive Sciences, and which is affiliated with the Computer Sciences & Artificial Intelligence Laboratory (CSAIL). This research was sponsored by grants from DARPA (IPTO and DSO), National Science Foundation (NSF-0640097,

[2]`http://yann.lecun.com/exdb/mnist/`

[3]`http://www.eecs.harvard.edu/~zak/pubfig83/`

NSF-0827427), AFSOR-THRL (FA8650-05- C-7262) Additional support was provided by: Adobe, Honda Research Institute USA, King Abdullah University Science and Technology grant to B. DeVore, NEC, Sony and especially by the Eugene McDermott Foundation.

References

[1] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[3] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Theory*. Springer, 2009.

[5] N. Pinto, Z. Stone, T. Zickler, and D.D. Cox. Scaling-up biologically-inspired computer vision: A case-study on facebook. In *Proc. of Workshop on Biologically Consistent Vision*. CVPR, 2011.

[6] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning. MIT Press, 2006.

[7] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[8] R.M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

[9] Matthias Seeger. Low rank updates for the Cholesky decomposition. Technical report, Department of EECS - University of California at Berkeley, 2008.

[10] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.

[11] S. Sonnenburg, G. Raetsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The shogun machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, 2010.