MLweb: A toolkit for machine learning on the web

Fabien Lauer

MLweb: A toolkit for machine learning on the web

Fabien Lauer
Université de Lorraine, CNRS, Inria, LORIA, 54506 Vandoeuvre-lès-Nancy, France

Abstract

This paper describes MLweb, an open source software toolkit for machine learning on the web. The specificity of MLweb is that all computations are performed on the client side without the need to send data to a third-party server. MLweb includes three main components: a JavaScript API for scientific computing (LALOLib), an extension of this library with machine learning tools (ML.js) and an online development environment (LALOLab) with many examples.

1 Introduction

MLweb is an open source software toolkit for machine learning on the web available at `http://mlweb.loria.fr` . A number of web-based commercial solutions exist to enable web applications with machine learning capabilities. These are often server-based: both training and predictions are performed by the server with data and results traveling across the Internet to the client upon request. This model is particularly suitable for the "big data" challenge, where heavy computing resources are needed. However, this also raises issues regarding for instance the privacy of data or the offline use of the prediction models. In addition, such a computing model relies on expensive computing infrastructures and leaves a number of individuals or nonprofit organizations without an easy and free access to machine learning capabilities. On the opposite, MLweb focuses on these issues (privacy, offline use, accessibility to non-experts and affordability at the individual level).

With these goals in mind, MLweb develops machine learning tools for the web in which all computations are performed on the client side, i.e., in the browser. Such a client-side approach has the direct combined benefits of privacy and offline use, since no data, models or predictions need to travel across the network. However, there is currently a lack of machine learning software following that approach, except for a few libraries dedicated to some specific algorithm (see, e.g., [1] for deep learning). Other open-source alternatives with a broader scope like scikit-learn [2] that might satisfy privacy and accessibility constraints do not work on the web and do not offer the possibility to develop applications easily distributed on the Internet without requiring users to

go through a complex installation procedure. On the opposite, MLweb just works on many platforms without requiring any installation step. For teaching or communication purposes, this can be used to easily set up an interactive demonstration of a (possibly novel) learning algorithm while ensuring that it is accessible from anywhere (even tablets and smartphones). Other potential uses include applications for which privacy might be more valuable than efficiency, or where the network or server constitutes the efficiency bottleneck.

2 Software Framework

MLweb includes three main components described below: a client-side web API in JavaScript for scientific computing (LALOLib), an extension of this library with machine learning tools (ML.js) and an online development environment (LALOLab). It also comes with an online help including many examples and an extended documentation in the form of a webbook.

MLweb uses only two third-party libraries, glpk.js for linear programming and JSZip for easy data compression. They are both automatically downloaded when installing from the source code and fetched when necessary when using the online executables.

2.1 LALOLib: a Linear Algebra Online Library

LALOLib is a *Linear ALgebra Online Library* written in JavaScript to enable scientific computing in web pages. JavaScript is the standard language for web applications, which is both very versatile and platform-independent. However, as a non-compiled language, it was not made for computation-intensive tasks and few numerical libraries exist. LALOLib attempts to fill this gap with both ease of use and efficiency in mind. With LALOLib, many common linear algebra, statistics and optimization operations are made available to the web page. A script within the page can for instance compute the singular values of a random matrix with

```
X = rand(200,300);
singularvalues = svd(X);
```

This example directly calls LALOLib functions, providing an easy access to their results. However, such a synchronous approach blocks the page (and freezes the browser) until the functions return. To maintain the responsiveness of the web application, LALOLib also provides an asynchronous mode based on callbacks.

A comparison of LALOLib with other JavaScript libraries [3, 4, 5] shows that LALOLib is both the most comprehensive and the most efficient library for linear algebra.

Indeed, the two JavaScript libraries math.js [3] and numeric.js [4] provide basic scientific computing support. However, they both lack important features such as sparse solvers, eigendecomposition, singular value decomposition (SVD) (for math.js) or statistics (for numeric.js). In terms of performance, they are also

Table 1: Comparison of the computing time (in milliseconds) for various standard operations obtained with Firefox 41 on a laptop with an Intel i7-3540M CPU at 3GHz. The meaning of the operations is as follows with all matrices having random entries. *Matrix creation:* create 10 matrices of size 10000×100. *Matrix multiply:* multiply a 20000×100 matrix by a 100×100 matrix. *Square linear system:* solve a square system with 500 variables. *Least squares:* Solve an overdetermined system with 10000 rows and 200 columns in a least square sense. *Sparse least squares:* same as least squares but with 2000 columns and only 20% of nonzero coefficients. *SVD:* compute the thin SVD of a 300×200 matrix. *Eigendecomposition:* compute the eigendecomposition of a 50×50 real and symmetric matrix. Details on the precise functions used can be found in the source of the page at `http://mlweb.loria.fr/benchmark/index.html` .

Operation	LALOLib	math.js	numeric.js	Sushi
Matrix creation	59	151	1923	73
Matrix multiply	429	3854	1420	545
Square linear system	98	1179	158	577
Least squares	667	> 12000	10905	1751
Sparse least squares	235	N/A	N/A	N/A
SVD	272	N/A	484	388
Eigen-decomposition	24	N/A	6661	N/A

behind LALOLib, in particular for large matrices, as can be seen from Table 1. This drop in performance is partly due to the choice of matrix representation based on an `Array` of `Array`s. While this provides easy access to the entries of a matrix `A` with `A[i][j]`, it has two major issues. First, operations on `Array`s are not easily optimized by the browser, since the size of the `Array` can change and its entries are not typed.[1] Second, for each row, this creates an `Array`, which is a complex object that must be handled by the garbage collector. As a result, a matrix with many rows implies significant slow downs at every garbage collection cycle.

To avoid these issues, LALOLib stores matrix entries in a single `Float64Array`, i.e., a typed array with fixed size for which browsers offer a much more interesting level of optimization. Sushi [5] also implements a more efficient matrix representation and multiplication. But it can be less accurate for more complex tasks, since it uses 32-bits floats (`Float32Array`) instead of 64 bits. In addition, default tolerance constants are set to large values, which results in large errors when solving linear systems or computing an SVD. Also note that Sushi does not provide functions for solving linear systems, and we had to explicitly compute the matrix inverse in the experiments of Table 1.

[1] Note that some browsers like Chrome are less sensitive to this issue.

2.2 ML.js: a JavaScript Library for Machine Learning

ML.js extends LALOLib with easy to use machine learning functions for classification, regression, clustering and dimensionality reduction. For instance, the code

```
model = new Classifier(SVM, {kernel: "rbf", kernelpar: 2.5, C: 10});
model.train(X,Y);
label = model.predict(x);
```

creates a new support vector machine classifier and trains it on examples stacked as rows in `X` and labeled by `Y`. Then, the classifier is used to predict the label of `x`. All classification and regression methods enjoy a unified interface with functions for training, prediction, test, tuning and cross-validation.

Despite the fact that MLweb targets small to moderate scale data sets, it has been developed with efficiency in mind, both at the coding level and in the choice of techniques implemented for the learning algorithms. While a JavaScript application cannot compete in terms of speed with a compiled code written for instance in C, Table 2 shows that MLweb still yields reasonable computing times for moderate size data sets. Note that these performances could not have been obtained with straightforward re-implementations of the algorithms due to the specificities of the JavaScript engines running (and optimizing) the code in the browsers.

In addition to standard training algorithms, MLweb includes automatic tuning procedures based on cross-validation and grid search or more advanced model selection techniques involving for instance regularization paths or fast kernel matrix updates[2]. Finally, models trained on large data sets with, e.g., LibSVM [6] or MSVMpack [7], can also be easily loaded to make online predictions.

2.3 LALOLab: a Linear Algebra Online Lab

LALOLab[3] is an online development environment in the flavor of Matlab based on the scientific computing library LALOLib. It also includes the functionalities of ML.js by default and can easily be extended with more. LALOLab accepts commands or scripts written in JavaScript or an easy to grasp Matlab-like syntax, e.g.,

```
x = randn(10)            // generate a random vector
A = randn(100,10)        // generate a random matrix
y = A*x + 0.01*randn(100)
xhat = A \ y       // solve the linear system Ax = y (min_x ||Ax - y||)
```

[2]Given a value of the Gaussian kernel, $K_\sigma(x,x') = \exp\left(-\|x-x'\|^2/2\sigma^2\right)$, for some bandwidth σ, we can efficiently compute the value for another bandwidth σ' as $K_{\sigma'}(x,x') = (K_\sigma(x,x'))^{(\sigma/\sigma')^2}$. In addition, by choosing a grid of bandwidths $\{\sigma_k\}_{k=1}^n$ such that $\sigma_k = 2^{-k/2}\beta$ for some constant $\beta > 0$, we have $K_{\sigma_{k+1}}(x,x') = K_{\sigma_k}(x,x')^2$ and we can update each value in the kernel matrix with a simple multiplication.

[3]LALOLab can be used simply by opening a browser at `http://mlweb.loria.fr/lalolab/` or from the source code of MLweb.

Table 2: Training time (in seconds) on random (or well-known) data sets of size #examples×dimension obtained with Firefox 53 on a laptop with an Intel i7-3540M at 3GHz. These experiments can be reproduced by running examples in LALOLab.

Method	Data set size	Time
Least squares regression	100000×1000	8.5
	100000×1000 (sparse)	3.0
	100000×2000 (sparse)	6.9
LASSO	442×10 (diabetes)	0.3
	10000×100	5.1
	1000×500	14.5
LARS entire regularization path	442×10 (diabetes)	0.01
	1000×500	3.2
	5000×500	13.4
Kernel ridge regression	2000×1000	2.8
	5000×1000	17.4
SVM classification (rbf) (tuning C and σ over a 5×10 grid)	1300×18 (image seg.)	12.2
SVM (one-vs-all, rbf)	500×256, 10 classes (usps)	0.5
Crammer&Singer multiclass-SVM (rbf)	500×256, 10 classes (usps)	61.4
K-nearest neighbors (tuning K by leave-one-out cross-validation)	500×256, 10 classes (usps)	4.9
Spectral clustering	500×100	3.5
	1000×100	10.8

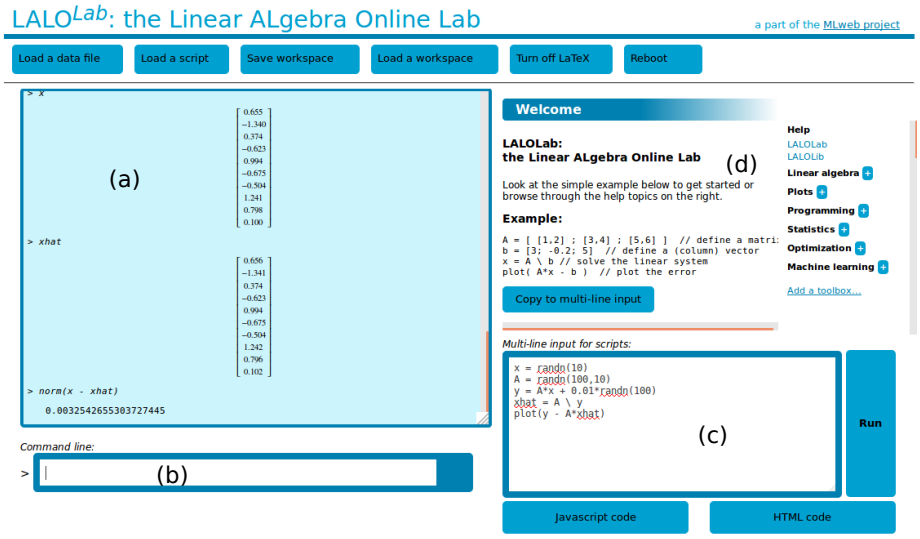

Figure 1: LALOLab web interface.

```
plot(y - A*xhat) // plot the error
norm(x - xhat)   // compute the estimation error ||x - xhat||
```

LALOLab divides the screen in four parts (see Fig. 1): (a) a main panel showing the results, (b) a command line, (c) a multi-line input block for writing scripts and (d) the online help including a list of all implemented functions. Illustrative examples are provided with the help of most functions and links to a webbook explaining machine learning topics in more details are also included. This provides a convenient platform to experiment with MLweb.

2.4 Extending MLweb

MLweb is meant to provide the basic building blocks for machine learning on the web together with a solid implementation of a few baseline methods rather than an exhaustive library of algorithms. Therefore, it was made to be easily extended with additional toolboxes. This can be done with different levels of expertise depending on the desired efficiency and the nature of the new features. For instance, a toolbox can be written with the Matlab-like syntax of LALOLab and easily converted into JavaScript by clicking a button in LALOLab. If more efficiency is needed, the documentation provides a set of guidelines to avoid common pitfalls and write faster code. Storing the resulting source file in the `toolboxes` folder and running `make` finishes to include the toolbox in LALOLab.

3 Conclusions

MLweb provides an efficient JavaScript implementation of basic functions for numerical computing together with a solid basis of machine learning tools for the web. MLweb will continuously be extended with new features and learning algorithms. We are also considering the now experimental features of the JavaScript language regarding shared-memory parallel processing in order to benefit from multi-cores architectures in a transparent manner for the user.

Acknowledgements

The author thanks Pedro Ernesto Garcia Rodriguez for his contribution to the software.

References

[1] A. Karpathy, Convnetjs: Deep learning in your browser (2014).
URL `http://cs.stanford.edu/people/karpathy/convnetjs/`

[2] 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, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[3] J. de Jong, math.js (2013).
URL `http://mathjs.org/`

[4] S. Loisel, Numeric javascript (2012).
URL `http://numericjs.com/`

[5] K. Miura, T. Mano, A. Kanehira, Y. Tsuchiya, T. Harada, Miljs : Brand new javascript libraries for matrix calculation and machine learning, arXiv:1502.06064 (2015).

[6] C. Chang, C. Lin, LibSVM: a Library for Support Vector Machines, http://www.csie.ntu.edu.tw/∼cjlin/libsvm/ (2001).

[7] F. Lauer, Y. Guermeur, MSVMpack: a multi-class support vector machine package, Journal of Machine Learning Research 12 (2011) 2269–2272, `http://www.loria.fr/~lauer/MSVMpack`.