ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

Solving Constrained Lasso and Elastic Net Using ν–SVMs

Carlos M. Alaíz, Alberto Torres and José R. Dorronsoro *

Universidad Autónoma de Madrid - Departamento de Ingeniería Informática
Tomás y Valiente 11, 28049 Madrid - Spain

Abstract. Many important linear sparse models have at its core the Lasso problem, for which the GLMNet algorithm is often considered as the current state of the art. Recently M. Jaggi has observed that Constrained Lasso (CL) can be reduced to a SVM-like problem, which opens the way to use efficient SVM algorithms to solve CL. We will refine Jaggi's arguments to reduce CL as well as constrained Elastic Net to a Nearest Point Problem and show experimentally that the well known LIBSVM library results in a faster convergence than GLMNet for small problems and also, if properly adapted, for larger ones.

1 Introduction

Big Data problems are putting a strong emphasis in using simple linear sparse models to handle large size and dimensional samples. This has made Lasso [1] and Elastic Net [2] often the methods of choice in large scale Machine Learning. Both are usually stated in their unconstrained version of solving

$$\min_{\beta} U_{\lambda,\mu}(\beta) = \frac{1}{2}\|X\beta - y\|_2^2 + \frac{\mu}{2}\|\beta\|_2^2 + \lambda\|\beta\|_1 \,, \tag{1}$$

where for an N-size sample $S = \{(x^1, y^1), \ldots, (x^N, y^N)\}$, X is an $N \times d$ data matrix containing as its rows the transposes of the d-dimensional features x^n, $y = (y^1, \ldots, y^N)^t$ is the $N \times 1$ target vector, β denotes the Lasso (when $\mu = 0$) or Elastic Net model coefficients, and the subscripts 1, 2 denote the ℓ_1 and ℓ_2 norms respectively. We will refer to problem (1) as λ-Unconstrained Lasso (or λ-UL), writing then simply U_λ, or as (μ, λ)-Unconstrained Elastic Net(or (μ, λ)-UEN). It has an equivalent constrained formulation

$$\min_{\beta} C_{\rho,\mu}(\beta) = \frac{1}{2}\|X\beta - y\|_2^2 + \frac{\mu}{2}\|\beta\|_2^2 \quad \text{s.t. } \|\beta\|_1 \leq \rho \,. \tag{2}$$

Again, we will refer to problem (2) as ρ-CL, and write C_ρ, or as (μ, ρ)-CEN. Two algorithmic approaches stand out when solving (1), the GLMNet algorithm [3] that uses cyclic coordinate descent and the FISTA algorithm [4], based on proximal optimization. Recently M. Jaggi [5] has shown the equivalence between

*With partial support from Spain's grants TIN2013-42351-P and S2013/ICE-2845 CASI-CAM-CM and also of the Cátedra UAM–ADIC in Data Science and Machine Learning. The authors also gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM. The second author is also supported by the FPU–MEC grant AP-2012-5163.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

constrained Lasso and some SVM variants. This is relatively simple when going from Lasso to SVM (the direction we are interested in) but more involved the other way around. In [6] Jaggi's approach is refined to obtain a one way reduction from constrained Elastic Net to a squared hinge loss SVM problem. As mentioned in [5], this opens the way for advances in one problem to be translated in advances into another and, while the application of SVM solvers to Lasso is not addressed in [5], prior work in parallelizing SVMs is leveraged in [6] to obtain a highly optimized and parallel solver for the Elastic Net and Lasso.

In this paper we retake M. Jaggi's approach, first to simplify and refine the reductions in [5] and [6] and, then, to explore the efficiency of the SVM solver LIBSVM, when applied to Lasso. More precisely, our contributions are

- A simple reduction in Sect. 2 of Elastic Net to Lasso and a proof of the equivalence between λ-UL and ρ-CL. This is a known result but proofs are hard to find, so our simple argument may have a value in itself.
- A refinement in Sect. 3 on M. Jaggi's arguments to reduce ρ-CL to Nearest Point and ν-SVC problems solvable using LIBSVM.
- An experimental comparison in Sect. 4 of NPP-LIBSVM with GLMNet and FISTA, showing NPP-LIBSVM to be competitive with GLMNet.

The paper will close with a brief discussion and pointers to further work.

2 Unconstrained and Constrained Lasso and Elastic Net

We show that λ-CL and ρ-UL and also (μ, λ)-UEN and (μ, ρ)-CEN have the same β solution for appropriate choices of λ and ρ. We will first reduce our discussion to the Lasso case describing how Elastic Net (EN) can be recast as a pure Lasso problem over an enlarged data matrix and target vector. For this let's consider in either problem (1) or (2) the $(N + d) \times d$ matrix $\mathcal{X}$ and $(N+d) \times 1$ vector $\mathcal{Y}$ defined as $\mathcal{X}^t = (X^t, \sqrt{\mu} I_d)$, $\mathcal{Y}^t = (y^t, 0_d^t)$ with I_d the $d \times d$ identity matrix and 0_d the d dimensional 0 vector. It is now easy to check that $\|X\beta - y\|_2^2 + \mu\|\beta\|_2^2 = \|\mathcal{X}\beta - \mathcal{Y}\|_2^2$ and, thus, we can rewrite (1) or (2) as Lasso problems over an extended sample

$$\mathcal{S} = \{(x^1, y^1), \ldots, (x^N, y^N), (\sqrt{\mu} e_1, 0), \ldots, (\sqrt{\mu} e_d, 0)\}$$

of size $N + d$, where e_i denotes the canonical vectors for R^d, for which $\mathcal{X}$ and $\mathcal{Y}$ are the new data matrix and target vector. Because of this, in what follows we limit our discussion to UL and CL, pointing where needed the changes to be made for UEN and CEN, and reverting to the (X, y) notations instead of $(\mathcal{X}, \mathcal{Y})$.

Assuming λ fixed it is obvious that a minimizer β_λ of λ-UL is also a minimizer for ρ_λ-CL with $\rho_\lambda = \|\beta_\lambda\|_1$. Next, if β_{LR} is the solution of Linear Regression and $\rho_{LR} = \|\beta_{LR}\|_1$, the solution of ρ-CL is clearly β_{LR} for $\rho \geq \rho_{LR}$, so we assume $\rho < \rho_{LR}$. Let's denote by β_ρ a minimum of the convex problem ρ-CL; we shall

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

prove next that β_ρ solves λ_ρ-UL with $\lambda_\rho = \beta_\rho^t X^t (X\beta_\rho - y)/\rho$. To do so, let $e_2(\beta) = \frac{1}{2}\|X\beta - y\|^2$ and $g(\beta) = \|\beta\|_1 - \rho$, and define

$$f(\beta) = \max\{e_2(\beta) - e_2^\rho, g(\beta)\} ,$$

where $e_2^\rho = e_2(\beta_\rho)$ is the optimal square error in ρ-CL. Then, since f is convex, the subgradient $\partial f(\beta')$ at any β' is

$$\partial f(\beta') = \{\gamma \nabla e_2(\beta') + (1-\gamma)h \ : \ 0 \leq \gamma \leq 1, \ \ h \in \partial g(\beta') = \partial \|\cdot\|_1(\beta_\rho)\} .$$

Thus, as β_ρ minimizes f (any β' with $\|\beta'\|_1 < \rho$ will have and error greater than e_2^ρ), there is an $h_\rho \in \partial\|\cdot\|_1(\beta_\rho)$ and γ, $0 \leq \gamma \leq 1$, such that $0 = \gamma\nabla e_2(\beta_\rho) + (1-\gamma)h_\rho$. But $\gamma > 0$, otherwise we would have $h_\rho = 0$, i.e., $\beta_\rho = 0$, contradicting that $\|\beta_\rho\|_1 = \rho$. Therefore, taking $\lambda_\rho = (1-\gamma)/\gamma$, we have

$$0 = \nabla e_2(\beta_\rho) + \lambda_\rho h_\rho \in \partial\left[e_2(\cdot) + \lambda_\rho\|\cdot\|_1\right](\beta_\rho) = \partial U_{\lambda_\rho}(\beta_\rho) ,$$

i.e., β_ρ minimizes λ_ρ-UL. We finally derive a value for λ_ρ by observing that $\beta_\rho^t h_\rho = \|\beta_\rho\|_1 = \rho$ and $0 = \nabla e_2(\beta_\rho) + \lambda_\rho h_\rho = -X^t r^{\beta_\rho} + h_\rho\lambda_\rho$, where r^{β_ρ} is just the residual $(X\beta_\rho - y)$. As a result,

$$\lambda_\rho\|\beta_\rho\|_1 = \beta_\rho^t h\lambda_\rho = \beta_\rho^t X^t r^{\beta_\rho} \ \Rightarrow \ \lambda_\rho = \frac{\beta_\rho^t X^t r^{\beta_\rho}}{\|\beta_\rho\|_1} = \frac{\beta_\rho^t X^t r^{\beta_\rho}}{\rho} .$$

For (μ,ρ)-CEN, the corresponding λ_ρ would be

$$\lambda_\rho = \frac{(\mathcal{Y} - \mathcal{X}\beta_\rho)\cdot\mathcal{X}\beta_\rho}{\rho} = \frac{(y - X\beta_\rho)\cdot X\beta_\rho - \mu\|\beta_\rho\|_2^2}{\rho} .$$

3 From Constrained Lasso to NPP

The basic idea in [5] is to re-scale $\tilde{\beta} = \beta/\rho$ and $\tilde{y} = y/\rho$ to normalize ρ-CL into 1-CL and then to replace the ℓ_1 unit ball B_1 with the simplex

$$\Delta_{2d} = \{\alpha = (\alpha_1, \ldots, \alpha_{2d}) \in R^{2d} \ : \ 0 \leq \alpha_i \leq 1, \sum_{i=1}^{2d}\alpha_i = 1\} ,$$

by enlarging the initial data matrix X to an $N \times 2d$ dimensional $\widehat{X}$ with columns $\widehat{X}^c = X^c$ and $\widehat{X}^{d+c} = -X^c$, $c = 1, \ldots, d$. As a consequence, the square error in the re-scaled Lasso $\|X\tilde{\beta} - \tilde{y}\|^2$ becomes $\|\widehat{X}\alpha - \tilde{y}\|^2$. Since $\widehat{X}\alpha$ now lies in the convex hull $\mathcal{C}$ spanned by $\{\widehat{X}^c, 1 \leq c \leq 2d\}$, finding an optimal α^o is equivalent to finding the point in $\mathcal{C}$ closest to $\tilde{y}$, i.e., solving the nearest point problem (NPP) between the N-dimensional convex hull $\mathcal{C}$ and the singleton $\{\tilde{y}\}$. In turn, NPP can be scaled [7] into an equivalent linear ν-SVC problem [8] with $\nu = 2/(2d+1)$, over the sample $\mathcal{S} = \{\mathcal{S}_+, \mathcal{S}_-\}$, where $\mathcal{S}_+ = \{X^1, \ldots, X^d, -X^1, \ldots, -X^d\}$ and $\mathcal{S}_- = \{\tilde{y}\}$. This problem can be solved using the LIBSVM library [9]. The optimal ρ-CL solution β_ρ is recovered as follows: first we obtain the optimal NPP coefficients α^o by scaling the ν-SVM solution γ^o as $\alpha^o = (2d+1)\gamma^o$, then we compute $(\tilde{\beta}_\rho)_i = \alpha_i^o - \alpha_{i+d}^o$ and finally we re-scale again $\beta_\rho = \rho\tilde{\beta}_\rho$.

Finally, changes for (μ,ρ)-CEN are straightforward, since we only have to add the d extra dimensions $\sqrt{\mu}e_c$ and $-\sqrt{\mu}e_c$ to the column vectors X^c.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

4 Numerical Experiments

In this section we will compare the performance of the LIBSVM approach for solving ρ-CL with two well known, state-of-the-art algorithms for λ-UL, FISTA and GLMNet. We first discuss their expected complexity.

FISTA (Fast ISTA; [4]) combines the basic iterations in Iterative Shrinkage–Thresholding Algorithm (ISTA) with a Nesterov acceleration step. Assuming that the covariance X^tX is precomputed at a fixed initial cost $O(Nd^2)$, the cost per iteration of FISTA is $O(d^2)$, i.e., that of computing $X^tX\beta$. If only m components of β are non zero, the iteration cost is $O(dm)$. On the other hand, GLMNet performs cyclic coordinate subgradient descent on the λ-UL cost function. GLMNet carefully manages the covariance computations $X^j \cdot X^k$ ensuring that there is a cost $O(Nd)$ only the first time they are performed at a given coordinate k and that afterwards their cost is just $O(d)$. Note that an iteration in GLMNet just changes one coordinate while in FISTA it changes d components. Finally, the ν-SVC solver in LIBSVM performs SMO-like iterations that change two coordinates, the pair that most violates the ν-SVC KKT conditions. The cost per iteration is also $O(d)$ plus the time needed to compute the required dot products, i.e., linear kernel operations. LIBSVM builds the kernel matrix making no assumptions on the particular structure of the data. This may lead to eventually compute $4d^2$ dot products (without considering the last column) even though only d^2 are actually needed, since the $2d \times 2d$ dimensional linear kernel matrix $\hat{X}\hat{X}^t$ is made of four $d \times d$ blocks with the covariance matrix $C = XX^t$ in the two diagonal blocks and $-C$ in the other two. This can be certainly controlled but in our experiments we have directly used LIBSVM, without any attempt to optimize running times exploiting the structure of the kernel matrix.

We compare next the number of iterations and running times that FISTA, GLMNet and LIBSVM need to solve equivalent λ-UL and ρ-CL problems. We will do so over four datasets from the UCI collection, namely the relatively small `prostate` and `housing`, and the larger `year` and `ctscan`. As it is well known, training complexity greatly depends on λ. We will consider three possible λ values for UL: an optimal λ^* obtained according to the regularization path procedure of [3], a smaller $\lambda^*/2$ value (that should result in longer training) and a stricter penalty $2\lambda^*$ value. The optimal λ^* values for the problems considered are $2.04\ 10^{-3}$, $8.19\ 10^{-3}$, $6.87\ 10^{-3}$ and $3.75\,10^{-3}$ respectively. The corresponding ρ parameters are computed as $\rho = \|\beta_\lambda\|_1$, with β_λ the optimal solution for λ-UL.

To make a balanced comparison, for each λ and dataset we first make a long run of each method M so that it converges to a β_M^* that we take as the optimum. We then compare for each M the evolution of $f_M(\beta_M^k) - f_M(\beta_M^*)$, with β_M^k the coefficients at the k-th iteration of method M, until it is smaller than a threshold ϵ that we take as 10^{-12} for `prostate` and `housing` and 10^{-6} for `year` and `ctscan`. Table 1 shows in columns 2 to 4 the number of iterations that each method requires to arrive at the ϵ threshold. As it can be seen, LIBSVM is the fastest on this account, with GLMNet in second place and FISTA a more distant third (for a proper comparison a FISTA iterate is made to correspond to

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

	Iterations			Time (ms)	
Dataset	LIBSVM	GLMNet	FISTA	LIBSVM	GLMNet
`prostate` ($2\lambda^*$)	86	181	1,232	0.051	0.053
`prostate` (λ^*)	75	174	1,336	0.049	0.057
`prostate` ($\lambda^*/2$)	52	173	904	0.036	0.075
`housing` ($2\lambda^*$)	143	1,010	5,538	0.266	0.832
`housing` (λ^*)	112	933	5,538	0.205	0.749
`housing` ($\lambda^*/2$)	88	666	3,913	0.190	0.520
`year` ($2\lambda^*$)	760	5,584	8,550	1,528.9	558.2
`year` (λ^*)	576	6,123	8,460	1,371.8	590.7
`year` ($\lambda^*/2$)	392	6,393	8,640	1,140.6	626.0
`ctscan` ($2\lambda^*$)	972	92,390	190,080	23,935.4	8,823.9
`ctscan` (λ^*)	670	78,456	159,744	15,868.2	6,935.8
`ctscan` ($\lambda^*/2$)	418	53,630	132,480	14,999.5	4,173.7

Table 1: Iterations and running times.

d iterates of LIBSVM and GLMNet). Columns 5 and 6 give the times required by LIBSVM and GLMNet; we omit FISTA's times as they are not competitive (at least under the implementation we used). We use the LIBSVM and GLMNet implementations in the Scikit Python library, which both have a compiled C core, so we may expect time comparisons to be broadly homogeneous.

Even without considering LIBSVM's overhead when computing a $2d \times 2d$ kernel matrix, it is clearly faster on `prostate` and `housing` but not so on the other two datasets. However, it turns out that LIBSVM indeed computes about 4 times more dot products than needed, which suggests that the running time of a LIBSVM version properly adapted for ρ-CL should have running times about a fourth of those reported here, outperforming then GLMNet. This behavior is further illustrated in Fig. 1 that depicts, for `housing` and `ctscan` with λ^*, the number of iterations and running times required to reach the ϵ threshold.

5 Discussion and Conclusions

GLMNet can be considered the current state of the art to solve the Lasso problem. M. Jaggi's recent observation that ρ-CL can be reduced to ν-SVC opens the way to the application of SVM methods. In this work we have shown using four examples how the ν-SVC option of LIBSVM is faster than GLMNet for small problems and pointed out how adapted versions could also beat it on larger problems. Devising such an adaptation is thus a first further line of work.

Moreover, Lasso is at the core of many other problems in convex regularization, such as Fused Lasso, wavelet smoothing or trend filtering, currently solved using specialized algorithms. A ν-SVC approach could provide for them the same faster convergence that we have illustrated here for standard Lasso. Furthermore, the ν-SVC training could be speed up, as suggested in [5], using the screening rules available for Lasso [10] in order to remove columns of the data matrix. We are working on these and other related questions.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

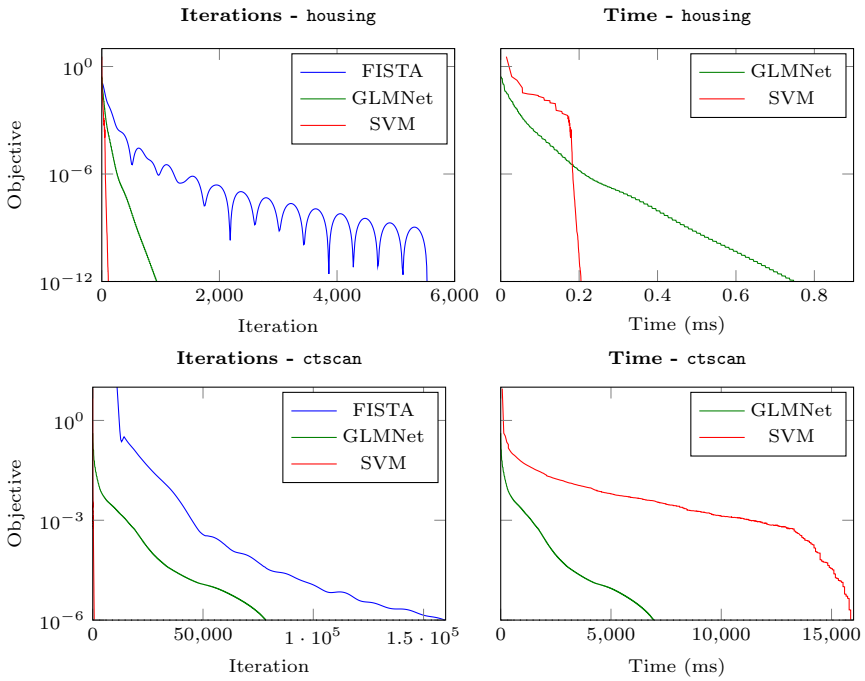

Fig. 1: Results for `housing` and `ctscan`.

References

[1] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society and Series B*, 58:267–288, 1994.

[2] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[3] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, 2009.

[5] M. Jaggi. An Equivalence between the Lasso and Support Vector Machines. In *Regularization, Optimization, Kernels, and Support Vector Machines*. CRC Press, 2014.

[6] Q. Zhou, W. Chen, S. Song, J. R. Gardner, K. Q. Weinberger, and Y. Chen. A Reduction of the Elastic Net to Support Vector Machines with an Application to GPU Computing. Technical Report arXiv:1409.1976 [stat.ML], 2014.

[7] J. López, Á. Barbero, and J.R. Dorronsoro. Clipping Algorithms for Solving the Nearest Point Problem over Reduced Convex Hulls. *Pattern Recognition*, 44(3):607–614, 2011.

[8] C.-C. Chang and C.-J. Lin. Training ν-Support Vector Classifiers: Theory and Algorithms. *Neural Computation*, 13(9):2119–2147, 2001.

[9] C.-C. Chang and C.-J. Lin. *LIBSVM: a Library for Support Vector Machines*. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[10] J. Wang, J. Zhou, P. Wonka, and J. Ye. Lasso screening rules via dual polytope projection. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1070–1078. Curran Associates, Inc., 2013.