ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Sparse One Hidden Layer MLPs

Alberto Torres, David Díaz and José R. Dorronsoro

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

Abstract. We discuss how to build sparse one hidden layer MLP replacing the standard l_2 weight decay penalty on all weights by an l_1 penalty on the linear output weights. We will propose an iterative two step training procedure where the output weights are found using FISTA proximal optimization algorithm to solve a Lasso-like problem and the hidden weights are computed by unconstrained minimization. As we shall discuss, the procedure has a complexity equivalent to that of standard MLP training, yields MLPs with similar performance and, as a by product, automatically selects the number of hidden units.

1 Introduction

Sparse modelling is attracting an increasing attention in problems with very high dimension, often larger than sample size itself. They are thus ill posed, prone to over-fitting and, if no correcting action is taken, likely to yield poor models. Sparse methods try to remedy this, and two approaches are emerging in the literature. In the first one, sparsity is achieved using sparsity-enforcing priors in a Bayesian setting [1, 2]. The alternative that we consider here is to add a sparsity enforcing regularizer to the problem natural criterion function. This approach leads to the convex optimization problem min $f + g$, where f is a smooth convex function (usually the model criterion) and g is also convex but not differentiable (the sparsity enforcing term). The first example is the Lasso problem [3] where f is the MSE criterion of a linear model and g is the l_1 weight norm. This yields sparse linear models. Recently a general way to solve the previous min $f + g$ problem is through proximal convex optimization [4]. As we discuss in Section 2, we with the observation that for any $\mu > 0$, the optimum $\mathbf{w}^*$ of $f + g$ verifies $\mathbf{w}^* = \text{prox}_{\mu g}(\mathbf{w}^* - \mu \nabla f(\mathbf{w}^*))$ where prox_h denotes the proximal operator associated with h. This fixed point property immediately suggests an iterative procedure. For the Lasso problem this has been done in a number of papers and has led to the Iterative Shrinkage/Thresholding Algorithm (ISTA), improved in [5] to the FISTA (Fast ISTA) algorithm for solving min $f + g$ for a general g. However, in order for it to be efficient, the proximal operator $\text{prox}_{\mu g}$ has to be easily computable.

The iterative structure of ISTA and FISTA suggests that, in principle, they can be applied to other problems. A natural idea is to use it to derive sparse Multilayer Perceptrons (MLPs), either in their standard, one hidden layer form or for deeper, many layer networks. Usual weight decay MLP regularization is performed by adding the l_2 squared norm of the weights $\|\mathbf{w}\|_2^2 = \sum w_j^2$ to the objective function. Looking towards sparse MLPs, the l_2-penalty could be either complemented with the l_1-norm, resulting in an MLP variant of the Elastic Net

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

problem, or simply replaced by the l_1-norm, as we will do here. However, while one could apply the l_1 norm to all weights, some care is needed. First, the FISTA criterion function must be convex but the standard MSE for MLPs is so only locally near minima. Moreover, sparsity at one layer enforces sparsity also at the previous one: if all the output weights from unit j at layer h are zero, all the input weights to that unit from layer $h-1$ can be also set to zero. In other words, enforcing sparsity at one layer might cascade to the previous ones.

To simplify, we will consider in this exploratory work one hidden layer networks with a single output unit. We denote by $\mathbf{W}^H$ the input to hidden layer weights and by $\mathbf{W}^O$ the weights connecting the hidden layer to the single output unit. To ensure FISTA convergence, we will train separately the output and the hidden weights in a two step approach described in Section 3, applying the l_1-penalty only to the output weights $\mathbf{W}^O$. Our overall goal is to analyze whether it is possible to train efficiently sparse MLPs (sMLPs) that yield effective models; sMLP training complexity is considered also in Section 3. In Section 4 we will build and compare standard and sparse MLP models performance in several classification and regression problems. We end the paper with a short discussion in Section 5, where we summarize our findings here and consider ways to further explore the application of sparse proximal optimization to MLPs.

2 Proximal Optimization Review

Assume we want to minimize a sum $f+g$ of a convex differentiable function f and a convex non differentiable one g. By the Moreau–Rockafellar theorem [6], $\mathbf{w}^*$ will be a minimum of $f+g$ iff $0 \in \nabla f(\mathbf{w}^*)+\partial g(\mathbf{w}^*)$ where $\partial h(\mathbf{w})$ denotes the subdifferential operator. Thus, for any $\gamma > 0$, we have $-\gamma\nabla f(\mathbf{w}^*) \in \gamma\partial g(\mathbf{w}^*)$ and, also, $\mathbf{w}^* - \gamma\nabla f(\mathbf{w}^*) \in (I+\gamma\partial g)(\mathbf{w}^*)$. This implies that the set-valued function $(I+\gamma\partial g)^{-1}$ verifies $\mathbf{w}^* \in (I+\gamma\partial g)^{-1}(\mathbf{w}^* - \gamma\nabla f(\mathbf{w}^*))$. Now, the proximal operator at $\mathbf{w}$ of a convex, lower semicontinuous function F with step $\gamma > 0$ is defined as

$$\mathbf{y} = \text{prox}_{\gamma F}(\mathbf{w}) = \underset{\mathbf{z}}{\text{argmin}} \left\{ \frac{1}{2}\|\mathbf{z}-\mathbf{w}\|_2^2 + \gamma F(\mathbf{z}) \right\}.$$

It follows that $0 \in \mathbf{y} - \mathbf{w} + \gamma\partial F(\mathbf{y})$ and, hence, $\mathbf{y} \in (I+\partial F)^{-1}(\mathbf{w})$. It can be shown [6] that ∂F is a monotone operator and this implies that the value of $(I+\partial F)^{-1}$ is unique and it defines a function that verifies $\text{prox}_{\gamma F}(\mathbf{w}) = \mathbf{y} = (I+\partial F)^{-1}(\mathbf{w})$. As a consequence, $\mathbf{w}^* = \text{prox}_{\gamma g}(\mathbf{w}^* - \gamma\nabla f(\mathbf{w}^*))$, which immediately suggests an iterative algorithm of the form

$$\mathbf{w}_{k+1} = \text{prox}_{\gamma g}(\mathbf{w}_k - \gamma\nabla f(\mathbf{w}_k)). \tag{1}$$

The previous equation is the basic principle behind the well known proximal gradient method [4] and also ISTA/FISTA. The latter combines equation (1) with $\gamma = 1/L$ and the momentum step

$$\mathbf{y}_{k+1} = \mathbf{w}_k + \frac{t_k - 1}{t_{k+1}}(\mathbf{w}_k - \mathbf{w}_{k-1}),$$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Algorithm 1: Batch Conjugate Gradient and FISTA MLP training

Input: Sample $(\mathbf{X}, \mathbf{y})$ and FISTA parameter λ_1
Initialize $k = 0$, $\mathbf{W}_0 = (\mathbf{W}_0^H, \mathbf{W}_0^O)$
while *stopping condition* $== false$ **do**
  $\mathbf{W}_{k+1}^H = \text{Conjugate Gradient}\left(E\left[\left\|\mathbf{y} - F\left(\mathbf{X}; \mathbf{W}_k^H, \mathbf{W}_k^O\right)\right\|_2^2\right]\right)$
  $\mathbf{H} = \sigma(\mathbf{W}_{k+1}^H \mathbf{X}^t)$
  $\mathbf{W}_{k+1}^O = \text{FISTA}\left(E\left[\left\|\mathbf{y} - \mathbf{H}^t \mathbf{W}_k^O\right\|_2^2\right] + \lambda_1 \|\mathbf{W}_k^O\|_1\right)$
end

where $t_{k+1} = \frac{1}{2}(1 + \sqrt{1 + 4t_k^2})$ and L a Lipschitz constant for ∇f. Observe that for FISTA to be efficient we need a simple computation of the proximal operator at the current $\mathbf{w}_k$. In the case of the l_1-norm, we have $g(\mathbf{w}) = \|\mathbf{w}\|_1$ and $[\text{prox}_{\gamma g}(\mathbf{w})]_i = \text{sign}(w_i)(|w_i| - \gamma)_+$. For a D-dimensional $\mathbf{w}$, this simply adds an $O(D)$ cost to the ∇f computation.

3 Sparse One Hidden Layer MLPs

Recall that we are considering here one hidden layer MLPs with linear output weights. The overall criterion function is therefore

$$J(\mathbf{W}) = J\left(\mathbf{W}^H, \mathbf{W}^O\right) = E\left[\left\|\mathbf{y} - F\left(\mathbf{X}; \mathbf{W}^H, \mathbf{W}^O\right)\right\|_2^2\right] + \lambda_1 \|\mathbf{W}^O\|_1, \quad (2)$$

where F is the MLP transfer function. This choice forces us to split the training of the $\mathbf{W}^O$ and $\mathbf{W}^H$ weights in a two step procedure, where we alternate FISTA optimization on the $\mathbf{W}^O$ weights and unconstrained optimization on the $\mathbf{W}^H$.

More precisely, we consider alternatively $J(\mathbf{W})$ first as a function $J_1\left(\mathbf{W}^O\right)$ of $\mathbf{W}^O$, with $\mathbf{W}^H$ fixed at a given $\overline{\mathbf{W}}^H$, namely,

$$J_1\left(\mathbf{W}^O\right) = J\left(\overline{\mathbf{W}}^H, \mathbf{W}^O\right) = E\left[\left\|\mathbf{y} - F\left(\mathbf{X}; \overline{\mathbf{W}}^H, \mathbf{W}^O\right)\right\|_2^2\right] + \lambda_1 \|\mathbf{W}^O\|_1, \quad (3)$$

that we optimize by FISTA. Then we fix $\mathbf{W}^O$ at a given $\overline{\mathbf{W}}^O$ and consider now the J_2 function $J_2\left(\mathbf{W}^H\right)$

$$J_2\left(\mathbf{W}^H\right) = E\left[\left\|\mathbf{y} - F\left(\mathbf{X}; \mathbf{W}^H, \overline{\mathbf{W}}^O\right)\right\|_2^2\right] = J\left(\mathbf{W}^H, \overline{\mathbf{W}}^O\right) - \lambda_1 \|\overline{\mathbf{W}}^O\|_1, \quad (4)$$

which we will optimize on $\mathbf{W}^H$ using the Conjugate Gradient (CG) method [7]. Now, if $(\mathbf{W}_k^H, \mathbf{W}_k^O)$ are the weights at step k and we apply successively the FISTA and CG steps to the functions $J_1^k(\mathbf{W}^O)$ and $J_2^k(\mathbf{W}^H)$, where we fix $\mathbf{W}_k^H$ and $\mathbf{W}_{k+1}^O$ respectively, it follows that

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

$$\begin{aligned} J(\mathbf{W}_{k+1}^H, \mathbf{W}_{k+1}^O) &= J_2(\mathbf{W}_{k+1}^H) + \lambda_1 \|\mathbf{W}_{k+1}^O\| \\ &\leq J_2(\mathbf{W}_k^H) + \lambda_1 \|\mathbf{W}_{k+1}^O\| = J_1(\mathbf{W}_{k+1}^O) \\ &\leq J_1(\mathbf{W}_k^O) = J(\mathbf{W}_k^H, \mathbf{W}_k^O); \end{aligned}$$

i.e., the overall criterion J decreases monotonically. The previous procedure will minimize the error no matter the order in which FISTA and CG are applied, so we can start with any of them. Notice also that the individual FISTA steps are not monotonic; thus FISTA has to converge to ensure minimization of J_1 and, hence, the overall monotonicity. An outline of the procedure is given in Algorithm 1.

We briefly discuss the algorithm's complexity. Let N be the number of sample patterns, H the number of hidden units and D pattern dimension. First, standard MLP complexity is dominated by the ∇J computation, which has a cost $O(NH)$ for the $\mathbf{W}^O$ gradient and $O(NHD)$ for the $\mathbf{W}^H$ one. We will train standard MLPs by the Conjugate Gradient (CG) method, that adds a $O(DH + H)$ cost to the previous one and, more importantly, several computations of J, with a cost of $O(NDH+NH)$ each, since the Numerical Recipes implementation we use computes several line minimizations of J. We can thus summarize the cost of CG over J as $nI_{CG} \times O(NDH + NH)$, where nI_{CG} is the number of global CG iterations needed. A similar analysis applies to the CG minimization of J_2 in Algorithm 1 that has a cost $nI_{CG}^s \times O(NDH)$, where nI_{CG}^s is now the total number of iterations performed by the outer loop in Algorithm 1. To this, the FISTA optimization of J_1 adds the cost $O(NH)$ of the $\mathbf{W}^O$ gradient plus the cost of the J_1 computations required by its backtracking version, with a cost of $O(NH)$ per iteration. Thus, FISTA total cost is $nI_F^s \times O(NH)$, where nI_F^s is the total number of FISTA iterations. We should expect $nI_{CG} \simeq nI_{CG}^s \simeq nI_F^s$ and, hence, the cost of our sMLP algorithm to be comparable to that of standard CG MLP training.

4 Numerical Experiments

In this section we will illustrate sparse MLP training and compare their accuracy with those of similar l_2-regularized MLPs. To do so we will work with five datasets taken from the UCI [8] and Delve[1] repositories: housing (HOU), computer hardware (CPU), car fuel consumption (MPG), Pima Indians diabetes (PIM) and Australian credit card approval (AUS). The number of patterns and attributes are given in table 1. The optimal weight decay parameter λ_2 and the optimal number of hidden units for the l_2-MLP models are determined by a search over a bidimensional grid. For each grid point the generalization error is estimated by 5-fold cross-validation and the parameters with the lowest error are selected. We find the optimal λ_1 for the sparse MLP in a similar way. Although the number of values tested for the parameters λ_1 and λ_2 is the same, the

[1] `http://www.cs.toronto.edu/~delve/data/datasets.html`

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

first and last values are 10 times bigger in the λ_1 case. This is done to roughly balance the sparse MLP penalty, that only applies to output weights, with the l_2 standard MLP penalty, that applies to both hidden and output weights. As mentioned before, standard MLP training is performed using the CG implementation in [7], iterating until convergence. Sparse MLPs are trained using Algorithm 1. At each step of the main loop the CG and FISTA components are iterated until convergence and loop iterations stop when the change in the objective function $F = f + g$ is less than a certain tolerance, that we set to 10^{-9}.

Table 1: Number of patterns and attributes.

Dataset	Patterns	Attributes	Type
HOU	506	14	Regression
CPU	8192	12	Regression
MPG	392	8	Regression
PIM	768	8	Classification
AUS	690	14	Classification

Training results are summarized in Table 2. For each dataset we compute the mean absolute errors (MAE) in the regression datasets and the accuracies (Acc) in the classification datasets. Figures shown in the table are the average of 100 independent train/test runs, together with their respective standard deviations. For sparse MLPs (l_1-MLP) we also give the number of active hidden units (nAHU), that is, the number of hidden-to-output weights different from 0 that determine the l_1–MLP sparsity.

Table 2: l_2 and l_1 MLP results for batch training.

Dataset	l_2-MLP MAE/Acc		l_1-MLP MAE/Acc		l_1-MLP nAHU
	Mean	Sd	Mean	Sd	
HOU	0.2456	0.0242	0.2575	0.0257	15
CPU	0.1170	0.0018	0.1086	0.0019	19
MPG	0.2494	0.0203	0.2740	0.0239	11
PIM	74.73	3.33	74.90	2.78	3
AUS	85.37	2.53	85.49	2.50	3

These results show that it is possible to train sparse MLPs with competitive results since its performance is similar to those of standard MLPs. However, we do not aim to find the best possible models or for sparse MLPs to outperform standard ones.

5 Discussion and Conclusions

In this paper we show one possible way to train l_1-regularized MLPs with the same theoretical complexity of weight decay MLPs. In addition, experiments demonstrate that both perform similar in terms of the MAE/accuracy. This suggests that sparse MLPs may deserve further attention. For instance, they

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

perform a type of automatic architecture selection, since the choice of the l_1 penalty parameter λ_1 ultimately determines the number of non-zero output weights and, hence, the effective number of hidden units. By using them we may avoid having to explore different hidden unit values when looking for optimal MLPs. In addition, this could be exploited in a more systematic way to derive a new approach to select MLP architecture, by applying the regularization path for the sparse $\mathbf{W}^O$ weights, similarly to what is done for the Lasso in [9]. Feature selection for MLPs is another natural application of sparsity enforcing proximal optimization, which can be done either for individual input variables or by feature groups using group variants of the Lasso. These variants can also be solved using proximal optimization. Note that this would require to penalize input to hidden unit weights with an sparse norm and, hence, to extend the approach here to that setting. If successful, this would open the possibility to apply these ideas to automatically compute sparse architectures in deeper, many layered perceptrons. We are currently studying these and similar issues.

References

[1] Ian Goodfellow, Aaron Courville, and Yoshua Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *ICML*, 2012.

[2] Daniel Hernández-Lobato, José Miguel Hernández-Lobato, and Pierre Dupont. Generalized spike-and-slab priors for bayesian group feature selection using expectation propagation. *Journal of Machine Learning Research*, 14:1891–1945, 2013.

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

[4] Patrick L. Combettes and Jean-Christophe Pesquet. Proximal Splitting Methods in Signal Processing. *ArXiv e-prints*, December 2009.

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

[6] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, September 2007.

[8] Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013.

[9] Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2 2010.