Scalable Training of L_1-Regularized Log-Linear Models

Galen Andrew GALENA@MICROSOFT.COM
Jianfeng Gao JFGAO@MICROSOFT.COM
Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA

Abstract

The L-BFGS limited-memory quasi-Newton method is the algorithm of choice for optimizing the parameters of large-scale log-linear models with L_2 regularization, but it cannot be used for an L_1-regularized loss due to its non-differentiability whenever some parameter is zero. Efficient algorithms have been proposed for this task, but they are impractical when the number of parameters is very large. We present an algorithm Orthant-Wise Limited-memory Quasi-Newton (OWL-QN), based on L-BFGS, that can efficiently optimize the L_1-regularized log-likelihood of log-linear models with millions of parameters. In our experiments on a parse reranking task, our algorithm was several orders of magnitude faster than an alternative algorithm, and substantially faster than L-BFGS on the analogous L_2-regularized problem. We also present a proof that OWL-QN is guaranteed to converge to a globally optimal parameter vector.

1. Introduction

Log-linear models, including the special cases of Markov random fields and logistic regression, are used in a variety of forms in machine learning. The parameters of such models are typically trained to minimize an objective function

$$f(x) = \ell(x) + r(x), \tag{1}$$

where ℓ is the negative log-probability of a labelled training set according to the model, and r is a regularization term that favors "simpler" models. It is well-known that the use of regularization is necessary to achieve a model that generalizes well to unseen data, particularly if the number of parameters is very high relative to the amount of training data.

A choice of regularizer that has received increasing attention in recent years is the weighted L_1-norm of the parameters

$$r(x) = C\|x\|_1 = C\sum_i |x_i|$$

for some constant $C > 0$. Introduced in the context of linear regression by Tibshirani (1996) where it is known as the LASSO estimator, the L_1 regularizer enjoys several favorable properties compared to other regularizers such as L_2. It was shown experimentally and theoretically to be capable of learning good models when most features are irrelevant by Ng (2004). It also typically produces *sparse* parameter vectors in which many of the parameters are exactly zero, which makes for models that are more interpretable and computationally manageable.

This latter property of the L_1 regularizer is a consequence of the fact that its first partial derivative with respect to each variable is constant as the variable moves toward zero, "pushing" the value all the way to zero if possible. (The L_2 regularizer, by contrast, "pushes" a value less and less as it moves toward zero, producing parameters that are close to, but not exactly, zero.) Unfortunately, this fact about L_1 also means that it is not differentiable at zero, so the objective function cannot be minimized with general-purpose gradient-based optimization algorithms such as the L-BFGS quasi-Newton method (Nocedal & Wright, 1999), which has been shown to be superior at training large-scale L_2-regularized log-linear models by Malouf (2002) and Minka (2003).

Several special-purpose algorithms have been designed to overcome this difficulty. Perkins and Theiler (2003) propose an algorithm called *grafting*, in which variables are added one-at-a-time, each time reoptimizing the weights with respect to the current set of variables. Goodman (2004) and Kazama and Tsujii (2003) (independently) show how to express the objective as a

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

constrained optimization problem, which they solve with a modification of generalized iterative scaling (GIS) (Darroch & Ratcliff, 1972) and BLMVM (Benson & More, 2001), a quasi-Newton algorithm for problems with bound constraints, respectively. Unfortunately, GIS is generally considered to be dominated by L-BFGS, and both of these algorithms require doubling the number of variables in the general case.

Lee et al. (2006) propose the algorithm IRLS-LARS, inspired by Newton's method, which iteratively minimizes the function's second order Taylor expansion, subject to linear constraints. The quadratic program at each iteration is efficiently solved using the LARS algorithm (LASSO variant) of Efron et al. (2004). They compare the approach to the other aforementioned algorithms (except that of Kazama & Tsujii) on small- to medium-scale logistic regression problems, and show that in most cases it is much faster. Unfortunately IRLS-LARS cannot be used to train very large-scale log-linear models involving millions of variables and training instances, such as are commonly encountered, for example, in natural language processing. Although worst-case bounds are not known, under charitable assumptions the LASSO variant of LARS may require as many as $\mathcal{O}(mn^2)$ operations, where m is the number of variables and n is the number of training instances. Indeed, the only test problems of Lee et al. in which another algorithm approached or surpassed IRLS-LARS were also the largest, with thousands of variables.

In this paper, we propose a new algorithm based on L-BFGS for training large-scale log-linear models using L_1 regularization, *Orthant-Wise Limited-memory Quasi-Newton* (OWL-QN). At each iteration, our algorithm computes a search direction by approximately minimizing a quadratic function that models the objective over an orthant containing the previous point. Our experiments on a parse re-ranking task with over one million features demonstrate the ability of OWL-QN to scale up to very large problems.

1.1. Notation

Let us establish some notation and a few definitions that will be used in the remainder of the paper. Suppose we are given a convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$ and vector $x \in \mathbb{R}^n$. We will let $\partial_i^+ f(x)$ denote the right partial derivative of f at x with respect to x_i:

$$\partial_i^+ f(x) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha e_i) - f(x)}{\alpha} \ ,$$

where e_i is the i^{th} standard basis vector, with the analogous left variant $\partial_i^- f(x)$. The *directional derivative* of f at x in direction $d \in \mathbb{R}^n$ is denoted $f'(x; d)$, and is defined as

$$f'(x; d) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha d) - f(x)}{\alpha} \ .$$

A vector d is referred to as a *descent direction* at x if $f'(x; d) < 0$. We will use $\|\cdot\|$ to represent the L_2 norm of a vector, unless explicitly written $\|\cdot\|_1$.

We will also find it convenient to define a few special functions. The *sign* function σ takes values in $\{-1, 0, 1\}$ according to whether a real value is negative, zero, or positive. The function $\pi : \mathbb{R}^n \mapsto \mathbb{R}^n$ is parameterized by $y \in \mathbb{R}^n$, where

$$\pi_i(x; y) = \begin{cases} x_i & \text{if } \sigma(x_i) = \sigma(y_i), \\ 0 & \text{otherwise,} \end{cases}$$

and can be interpreted as the projection of x onto an orthant defined by y.

2. Quasi-Newton Algorithms and L-BFGS

We begin our discussion of OWL-QN with a description of its parent, the L-BFGS quasi-Newton algorithm for unconstrained optimization of a smooth function.

Like Newton's method, *quasi-Newton* algorithms iteratively construct a local quadratic approximation to a function, and then conduct a line search in the direction of the point that minimizes the approximation. If $\boldsymbol{B_k}$ is the (perhaps approximated) Hessian matrix of a smooth function f at the point x^k, and g^k is the gradient of f at x^k, the function is locally modelled by

$$Q(x) = f(x^k) + (x - x^k)^\top g^k + \frac{1}{2}(x - x^k)^\top \boldsymbol{B_k}(x - x^k). \tag{2}$$

If $\boldsymbol{B_k}$ is positive definite, the value x^* that minimizes Q can be computed analytically according to

$$x^* = x^k - \boldsymbol{H_k} g^k,$$

where $\boldsymbol{H_k} = \boldsymbol{B_k}^{-1}$. A quasi-Newton method then explores along the ray $x^k - \alpha \boldsymbol{H_k} g^k$ for $\alpha \in (0, \infty)$ to obtain the next point x^{k+1}.

While pure Newton's method uses the exact second-order Taylor expansion at each point, quasi-Newton algorithms approximate the Hessian using first-order information gathered from previously explored points. L-BFGS, as a *limited-memory* quasi-Newton algorithm, maintains only curvature information from the most recent m points. Specifically, at step k, it records the displacement $s^k = x^k - x^{k-1}$ and the change in gradient $y^k = g^k - g^{k-1}$, discarding the corresponding

vectors from iteration $k-m$. It then uses $\{s^i\}$ and $\{y^i\}$ to estimate $\boldsymbol{H_k}$, or more precisely, to estimate the search direction $-\boldsymbol{H_k}g^k$, since the full Hessian matrix (which may be unmanageably large) is not explicitly computed or inverted. The time and memory requirements of the computation are linear in the number of variables. The details of this process are not important for the purposes of this paper, and we refer the interested reader to Nocedal and Wright (1999).

3. Orthant-Wise Limited-memory Quasi-Newton

For the remainder of the paper, we assume that the loss function $\ell : \mathbb{R}^n \mapsto \mathbb{R}$ is convex, bounded below, continuously differentiable, and that the gradient $\nabla\ell$ is L-Lipschitz continuous on the set $\aleph = \{x : f(x) \leq f(x^0)\}$ for some L and some initial point x^0. Our objective is to minimize $f(x) = \ell(x) + C\|x\|_1$ for a given constant $C > 0$.

Our algorithm is motivated by the following observation about the L_1 norm: when restricted to any given *orthant*, i.e., a set of points in which each coordinate never changes sign, it is differentiable, and in fact is a linear function of its argument. Hence the second-order behavior of the regularized objective f on a given orthant is determined by the loss component alone. This consideration suggests the following strategy: construct a quadratic approximation that is valid for some orthant containing the current point using the inverse Hessian estimated from the loss component alone, then search in the direction of the minimum of the quadratic, restricting the search to the orthant on which the approximation is valid.

For any sign vector $\xi \in \{-1, 0, 1\}^n$, let us define

$$\Omega_\xi = \{x \in \mathbb{R}^n : \pi(x; \xi) = x\},$$

which is the intersection of an orthant and a plane constraining some coordinates to be zero. Clearly, for all x in Ω_ξ,

$$f(x) = \ell(x) + C\xi^\top x.$$

Defining f_ξ to be the extension of this function to all of $\mathbb{R}^n$, we have a differentiable function that coincides with f on Ω_ξ. Using $\boldsymbol{H_k}$, the L-BFGS approximation to the inverse Hessian of the loss, and v^k, the negative gradient of f_ξ at x^k projected onto the subspace containing Ω_ξ,[1] we can approximate f_ξ on Ω_ξ with a quadratic function Q_ξ as in (2), and search in the direction of the minimum of Q_ξ. For technical reasons, we constrain the search direction to match the sign pattern of v^k:[2]

$$p^k = \pi(\boldsymbol{H_k}v^k; v^k). \tag{3}$$

3.1. Choosing an orthant

There may be many orthants containing or adjacent to a given point, depending on how many of its coordinates are zero. In order to determine which orthant Ω_ξ to explore, let us define the *pseudo-gradient* of f at x, denoted $\diamond f(x)$, according to

$$\diamond_i f(x) = \begin{cases} \partial_i^- f(x) & \text{if } \partial_i^- f(x) > 0 \\ \partial_i^+ f(x) & \text{if } \partial_i^+ f(x) < 0 \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where the left and right partial derivatives of f are

$$\partial_i^\pm f(x) = \frac{\partial}{\partial x_i}\ell(x) + \begin{cases} C\sigma(x_i) & \text{if } x_i \neq 0 \\ \pm C & \text{if } x_i = 0. \end{cases}$$

Note that $\partial_i^- f(x) \leq \partial_i^+ f(x)$, so $\diamond$ is well-defined. The pseudo-gradient generalizes the gradient in that the directional derivative at x is minimized (the local rate of decrease is maximized) in the direction of $-\diamond f(x)$, and x is a local minimum if and only if $\diamond f(x) = 0$.

A reasonable choice of orthant to explore is the one containing x^k and into which $-\diamond f(x^k)$ leads:

$$\xi_i^k = \begin{cases} \sigma(x_i^k) & \text{if } x_i^k \neq 0 \\ \sigma(-\diamond_i f(x^k)) & \text{if } x_i^k = 0. \end{cases}$$

A consequence of this choice is that $-\diamond f(x^k)$ is equal to v^k, the projection of the negative gradient of f_ξ at x^k onto the subspace containing Ω_ξ. Thus it isn't necessary to determine ξ^k explicitly; one merely computes $-\diamond f(x^k)$, and this is what is multiplied by $\boldsymbol{H_k}$ in (3).

3.2. Constrained line search

During the line search, in order to ensure that we do not leave the region on which Q_ξ is valid, we project each point explored orthogonally back onto Ω_ξ, that is we explore points

$$x^{k+1} = \pi(x^k + \alpha p^k; \xi^k),$$

which amounts to setting to zero any coordinate that moves from positive to negative or vice-versa. Any number of methods could be used to choose α, but we use the following variation of backtracking line search in our experiments and convergence proof. Choose

[1] This projection just means that v_i^k is set to zero whenever ξ_i is zero.

[2] This ensures that the line search does not deviate too far from the direction of steepest descent, and is necessary to guarantee convergence.

Algorithm 1 OWL-QN

choose initial point x^0
$S \Leftarrow \{\}, Y \Leftarrow \{\}$
for $k = 0$ **to** MaxIters **do**
  Compute $v^k = - \diamond f(x^k)$ (1)
  Compute $d^k \Leftarrow \boldsymbol{H_k} v^k$ using S and Y
  $p^k \Leftarrow \pi(d^k; v^k)$ (2)
  Find x^{k+1} with constrained line search (3)
  if termination condition satisfied **then**
    Stop and return x^{k+1}
  end if
  Update S with $s^k = x^{k+1} - x^k$
  Update Y with $y^k = \nabla \ell(x^{k+1}) - \nabla \ell(x^k)$ (4)
end for

constants $\beta, \gamma \in (0, 1)$ and for $n = 0, 1, 2, \ldots$, accept the first step size $\alpha = \beta^n$ such that

$$f(x^{k+1}) \leq f(x^k) - \gamma v^\top (x^{k+1} - x^k).$$

A pseudo-code description of OWL-QN is given in Algorithm 1. In fact, only a few steps of the standard L-BFGS algorithm have been changed. The only differences have been marked in the figure:

1. The pseudo-gradient $\diamond f(x^k)$ of the regularized objective is used in place of the gradient.
2. The resulting search direction is constrained to match the sign pattern of $v^k = - \diamond f(x^k)$. This is the projection step of Equation 3.
3. During the line search, each search point is projected onto the orthant of the previous point.
4. The gradient of the unregularized loss alone is used to construct the vectors y^k used to approximate the inverse Hessian.

Starting with an implementation of L-BFGS, writing OWL-QN requires changing only about 30 lines of code.

4. Experiments

We evaluated the algorithm OWL-QN on the task of training a conditional log-linear model for re-ranking linguistic parses of sentences of natural language. Following Collins (2000), the setup is as follows. We are given:

- a procedure that produces an N-best list of candidate parses $\texttt{GEN}(x) \subseteq Y$ for each sentence $x \in X$.
- training samples (x_j, y_j) for $j = 1 \ldots M$, where $x_j \in X$ is a sentence, and $y_j \in \texttt{GEN}(x_j)$ is the gold-standard parse for that sentence, and
- a feature mapping $\Phi : X \times Y \mapsto \mathbb{R}^n$, which maps each pair (x, y) to a vector of feature values.

For any weight vector $w \in \mathbb{R}^n$ we define a distribution over parses for each sentence according to

$$P_w(y|x) = \frac{\exp w^\top \Phi(x, y)}{\sum_{y' \in \texttt{GEN}(x)} \exp w^\top \Phi(x, y')}.$$

Our task is to minimize

$$f(w) = \ell(w) + C\|w\|_1,$$

where the loss term $\ell(w)$ is the negative conditional log-likelihood of the training data:

$$\ell(w) = -\sum_{j=1}^{M} \log P_w(y_j|x_j).$$

We followed the experimental paradigm of parse reranking outlined by Charniak and Johnson (2005). We used the same generative baseline model for generating candidate parses, and the nearly the same feature set, which includes the log probability of a parse according to the baseline model plus 1,219,272 additional features. We trained our the model parameters on Sections 2-19 of the Penn Treebank, used Sections 20-21 to select the regularization weight C, and then evaluated the models on Section 22.[3] The training set contains 36K sentences, while the held-out set and the test set have 4K and 1.7K, respectively.

We compared OWL-QN to a fast implementation of the only other special-purpose algorithm for L_1 we are aware of that can feasibly run at this scale: that of Kazama and Tsujii (2003), hereafter called "K&T". In K&T, each weight w_i is represented as the difference of two values: $w_i = w_i^+ - w_i^-$, with $w_i^+ \geq 0, w_i^- \geq 0$. The L_1 penalty term then becomes simply $\|w\|_1 = \sum_i w_i^+ + w_i^-$. Thus, at the cost of doubling the number of parameters, we have a constrained optimization problem with a differentiable objective that can be solved with general-purpose numerical optimization software. In our experiments, we used the AlgLib implementation of the L-BFGS-B algorithm of Byrd et al. (1995), which is a C++ port of the FORTRAN code by Zhu et al. (1997).[4] We also ran two implementations of L-BFGS (AlgLib's and our own, on which OWL-QN is based) on the L_2-regularized problem.

[3] Since we are not interested in parsing performance *per se*, we did not evaluate on the standard test corpus used in the parsing literature (Section 23).

[4] The original FORTRAN implementation can be found at `www.ece.northwestern.edu/~nocedal/lbfgsb.html`, while the AlgLib C++ port is available at `www.alglib.net`.

Table 1: Chosen value of C and F-scores of the models used in the study.

	C	dev	test
Baseline		0.8925	0.8986
Oracle		0.9632	0.9687
L_2 (L-BFGS)	13.0	0.9103	0.9163
L_1 (OWL-QN)	3.2	0.9101	0.9165

We used the memory parameter $m = 5$ for all four algorithms. For the timing results, we first ran each algorithm until the relative change in the function value, averaged over the previous five iterations, dropped below $\tau = 10^{-5}$. We report the CPU time and the number of function evaluations required for each algorithm to reach within 1% of the smallest value found by either of the two algorithms. We also report the number of function evaluations so that the algorithms can be compared in an implementation-independent way.

4.1. Results

Although we are primarily interested in the efficiency of training, we first report the performance of the learned parse models. Performance is measured with the *PARSEVAL* metric, i.e., the F-score over labelled brackets. These results are summarized in Table 1. "Baseline" refers to the generative model used by `GEN`. "Oracle" shows ideal performance if the best parse from `GEN` (according to F-score) were always selected by the re-ranking model. Both types of model performed significantly better than the baseline, and may indeed be considered state-of-the-art. (For comparison, the model of Charniak and Johnson (2005) also achieved 91.6% F-score on the same test set.[5]) Interestingly, the two regularizers performed almost identically: the Wilcoxon paired signed-rank test did not find the difference statistically significant.

The results of CPU timing experiments using the same values of C are shown in Table 2. We stopped K&T after 946 iterations when it had reached the value 7.34×10^4, still 5.7% higher than the best value found by OWL-QN. The difference in both runtime and number of function evaluations between K&T and OWL-QN is quite dramatic. Surprisingly, OWL-QN converges even faster than our implementation of L-BFGS run on the L_2-regularized objective. Note also that the runtime of all algorithms is dominated by evaluations of the objective function, and that otherwise the most expensive step of OWL-QN is the computation of the L-BFGS direction.

[5]Mark Johnson, personal communication, May 2007.

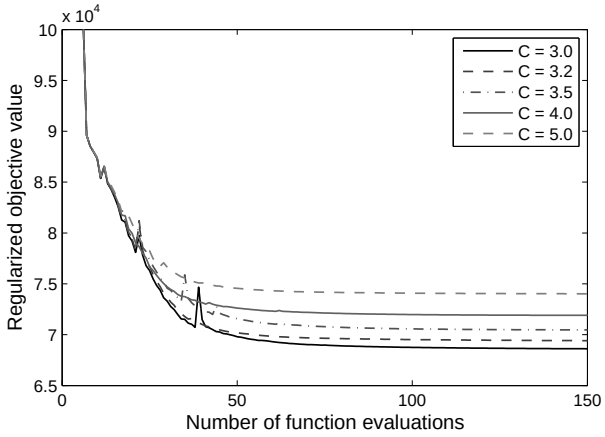

Figure 1: L_1-regularized objective value during the course of optimization with OWL-QN

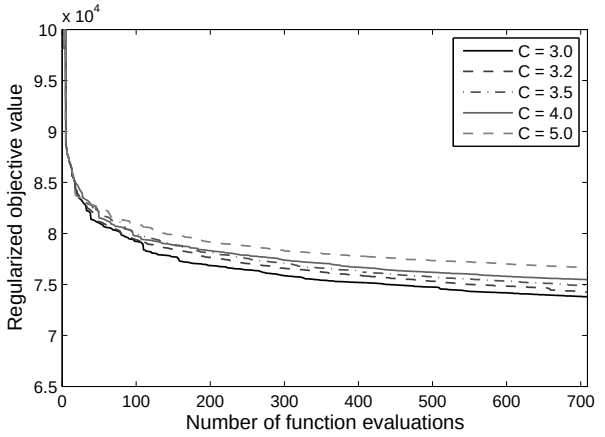

Figure 2: L_1-regularized objective value during the course of optimization with K&T

A more complete picture of the training efficiency of the two models can be gleaned by plotting the value of the objective as a function of the number function calls, shown in Figures 1 through 4. (Note the differences in scale of the x-axis.)

Since its ability to learn a sparse parameter vector is an important advantage of the L_1 regularizer, we examine how the number of non-zero weights changes during the course of optimization in Figures 5 and 6. Both algorithms start with a significant fraction of features (5%-12%) and prune them away as the algorithm progresses, with OWL-QN producing a sparse model rather more quickly. Interestingly, OWL-QN interrupts this pattern with a single sharp valley at the start of the second iteration (which is actually the sixth function evaluation, due to the line search). We believe the cause of this is that the model gives a large weight

Table 2: Time and function evaluations to reach within 1% of the best value found. All times are in seconds. Figures in parentheses show percentages of total time.

	func evals	func eval time	L-BFGS dir time	other time	total time
OWL-QN	54	707 (97.7)	10.4 (1.4)	6.9 (1.0)	724
K&T (AlgLib)	> 946	16043 (91.2)	1555 (8.8)		> 17598
L-BFGS (our impl.)	109	1400 (97.7)	22.4 (1.5)	10 (0.7)	1433
L-BFGS (AlgLib)	107	1384 (83.4)	276 (16.6)		1660

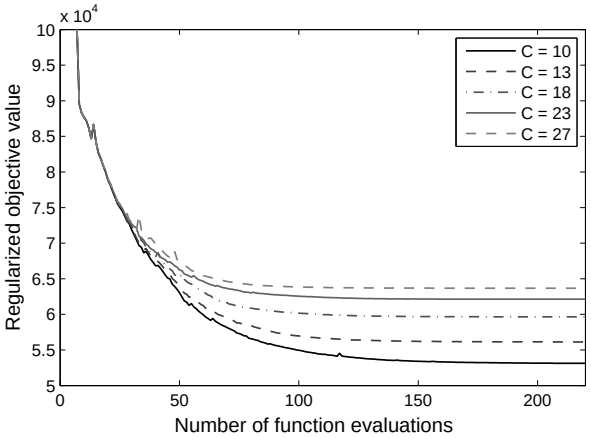

Figure 3: L_2-regularized objective value during the course of optimization with our L-BFGS implementation

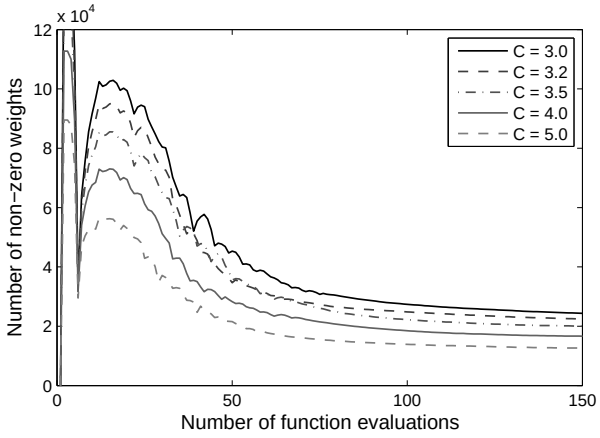

Figure 5: Number of non-zero weights during the course of optimization with OWL-QN

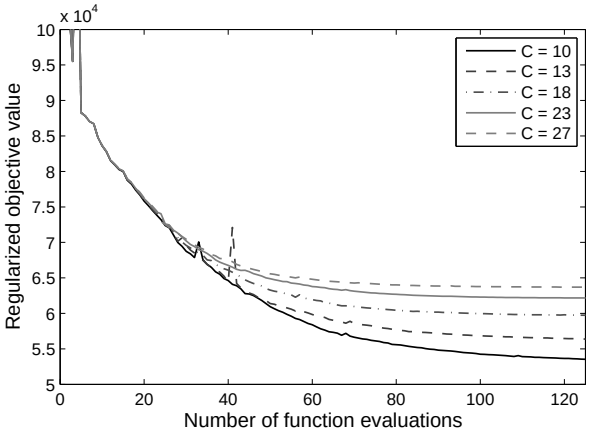

Figure 4: L_2-regularized objective value during the course of optimization with AlgLib's L-BFGS

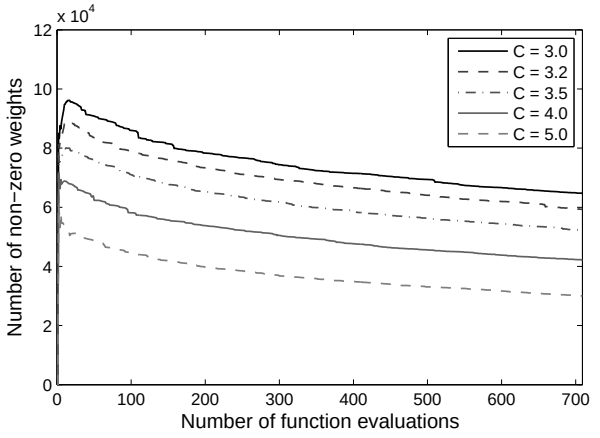

Figure 6: Number of non-zero weights during the course of optimization with K&T

to many features on the first iteration, only to send most of them back toward zero on the second. On the third iteration some of those features receive weights of the opposite sign, and from there the set of non-zero weights is more stable.[6]

[6]Many thanks to Mark Johnson for suggesting this explanation.

Although here we have focused on the runtime behavior of OWL-QN for a single problem, we have also used it to train L_1-regularized models with up to ten million variables for a variety of other problems in NLP, including part-of-speech tagging, Chinese word segmentation and language modeling. This work is described in Gao et al. (2007).

5. Conclusions

We have presented an algorithm OWL-QN for efficiently training L_1-regularized log-linear models with millions of variables. We tested the algorithm on a very large-scale NLP task, and found that it was considerably faster than an alternative algorithm for L_1 regularization, and even somewhat faster than L-BFGS on the analogous L_2-regularized problem. It would be interesting to see whether OWL-QN might be useful on L_1-regularized problems of other forms involving millions of variables, for example LASSO regression. Another direction to explore would be to use similar methods to optimize objectives with different sorts of non-differentiabilities, such as the SVM primal objective.

6. Acknowledgements

We are especially grateful to Kristina Toutanova for discussion of the convergence proof. We also give warm thanks to Mark Johnson for sharing his parser, and to our reviewers for helping us to improve the paper.

7. Appendix: Proof of Convergence

We will use the following fact about L-BFGS:[7]

Theorem 1. *Given positive constants L_1 and L_2, and an integer $m > 0$, there exist positive constants M_1 and M_2 such that, for any set of m vector pairs (s, y) satisfying $L_1\|y\|^2 \leq s^\top y \leq L_2\|s\|^2$, it follows that $\forall x : M_1\|x\|^2 \leq x^\top \boldsymbol{H} x \leq M_2\|x\|^2$, where $\boldsymbol{H}$ is the inverse Hessian approximation (implicitly) generated by* L-BFGS *using those vector pairs.*[8]

Note that this implies that $\boldsymbol{H}$ is positive definite.

We first establish several facts concerning OWL-QN that are not too difficult to verify.

Proposition 1. *If p^k is a descent direction at x^k, the line search will terminate in a finite number of steps.*

Intuitively, this is because backtracking line search will eventually try a value of α small enough that no coordinate changes sign. Then our termination criterion reduces to the Armijo rule, which is always satisfied for small enough α.

[7] For our problem, we know that $s^\top y/\|s\|^2$ is bounded because the gradient of ℓ is Lipschitz continuous. Technically, since ℓ is convex but perhaps not strictly convex, it is possible for $y^\top s/\|y\|^2$ to be arbitrarily close to zero. One can still make sure the conditions of the theorem are satisfied by choosing a small, positive constant ω and skipping L-BFGS updates whenever $y^\top s < \omega$. This is the strategy used by Byrd et al. (1995).

[8] An equivalent conclusion would be that the condition number of $\boldsymbol{H}$ is bounded.

Proposition 2. *For all $v \in \mathbb{R}^n$, if $v \neq 0$ and $\boldsymbol{H}$ is positive definite, then $p = \pi(\boldsymbol{H}v; v) \neq 0$.*

Proof. It follows immediately from the definition of π that $\pi_i(\boldsymbol{H}v; v) = 0$ only if $v_i(\boldsymbol{H}v)_i \leq 0$. If $p = 0$, we would have $v^\top \boldsymbol{H} v = \sum_i v_i(\boldsymbol{H}v)_i \leq 0$, which contradicts that $\boldsymbol{H}$ is positive definite. □

Proposition 3. *If $\{x^k\} \to \bar{x}$ and $\diamond f(\bar{x}) \neq 0$, then $\liminf_{k\to\infty} \| \diamond f(x^k)\| > 0$.*

Proof. Since $\diamond f(\bar{x}) \neq 0$, we can take i such that $\diamond_i f(\bar{x}) \neq 0$, so either $\partial_i^- f(\bar{x}) > 0$ or $\partial_i^+ f(\bar{x}) < 0$. From (4), we have that $\forall k : \diamond_i f(x^k) \in [\partial_i^- f(x^k), \partial_i^+ f(x^k)]$. Therefore all limit points of $\{\diamond_i f(x^k)\}$ must be in the interval $[\partial_i^- f(\bar{x}), \partial_i^+ f(\bar{x})]$, which does not include zero.[9] □

Proposition 4. *Define $q_\alpha^k = \frac{1}{\alpha}(\pi(x^k + \alpha p^k; \xi^k) - x^k)$. Then for all $\alpha \in (0, \infty)$ and all i,*

$$d_i^k v_i^k \leq p_i^k v_i^k \leq (q_\alpha^k)_i v_i^k \leq 0,$$

and therefore

$$(v^k)^\top q_{\alpha^k}^k \geq (v^k)^\top p^k \geq (v^k)^\top d^k.$$

Proposition 5. *At any point x with steepest descent vector $v = -\diamond f(x)$, if p is a non-zero direction vector such that $\sigma(v_i) = \sigma(p_i)$ whenever $\sigma(p_i) \neq 0$, then $f'(x; p) = -v^\top p$, and $f'(x; p) < 0$.*

Note that d^k, p^k and q_α^k as previously defined all satisfy the conditions of Prop. 5.

Theorem 2. *The sequence of values $\{f(x^k)\}$ explored by the algorithm* OWL-QN *during the course of optimization converges to the global minimum of f.*

Proof. The sequence $\{x^k\}$ must have a limit point $\bar{x}$ since every x^k is in the bounded set $\aleph$. It will be sufficient to show that $\bar{x}$ minimizes f, since $\{f(x^k)\}$ is decreasing. To simplify notation, let us assume without loss of generality that $\{x^k\}$ converges to $\bar{x}$, by replacing $\{x^k\}$ with a convergent subsequence if necessary. Let $\bar{v} = -\diamond f(\bar{x})$. We will show that $\|\bar{v}\| = 0$, and therefore $\{\bar{x}\}$ attains the globally minimal function value. To this end, assume for a contradiction that $\|\bar{v}\| > 0$.

Since $\{f(x^k)\}$ is decreasing and bounded, we know that $\lim_{k\to\infty} f(x^k) - f(x^{k+1}) = 0$. If we define q_α^k as in Prop. 4, the line search criterion can be written:

$$f(x^{k+1}) = f(x^k + \alpha q_\alpha^k) \leq f(x^k) - \gamma\alpha(v^k)^\top q_\alpha^k.$$

[9] We use property B.24(c) of (Bertsekas, 1999).

Therefore,

$$\begin{aligned}\frac{f(x^k)-f(x^{k+1})}{\gamma\alpha^k} &\geq (v^k)^\top q^k_{\alpha^k}\\ &\geq (v^k)^\top d^k\\ &= (v^k)^\top \boldsymbol{H_k} v^k\\ &\geq M_1\|v^k\|^2, \qquad (5)\end{aligned}$$

and as Prop. 3 gives us that $\liminf_{k\to\infty}\|v^k\| > 0$ we conclude that $\{\alpha^k\}\to 0$.

Thus there must exist a $\bar{k}$ such that $k > \bar{k} \Rightarrow \alpha^k < \beta$. Due to the form of the line search, if for some k, $\alpha^k < \beta$, it must be the case that the previously tried value of α on that iteration, $\alpha^k\beta^{-1}$, did not meet the criterion, i.e., for $k > \bar{k}$,

$$f(x^k+(\alpha^k\beta^{-1})q^k_{\alpha^k\beta^{-1}}) > f(x^k)-\gamma\alpha^k\beta^{-1}(v^k)^\top q^k_{\alpha^k\beta^{-1}}$$

which we can rewrite as

$$\frac{f(x^k+\hat{\alpha}^k\hat{q}^k)-f(x^k)}{\hat{\alpha}^k} > -\gamma(v^k)^\top\hat{q}^k, \qquad (6)$$

defining

$$\hat{q}^k = \tfrac{q^k_{\alpha^k\beta^{-1}}}{\|q^k_{\alpha^k\beta^{-1}}\|}, \quad \hat{\alpha}^k = \alpha^k\beta^{-1}\|q^k_{\alpha^k\beta^{-1}}\|.$$

Since $\aleph$ is a bounded set, $\{\|v^k\|\}$ is bounded, so it follows from Theorem 1 that $\{\|p^k\|\}$, hence $\{\|q^k_{\alpha^k\beta^{-1}}\|\}$, is bounded. Therefore $\{\hat{\alpha}^k\}\to 0$. Also, since $\|\hat{q}^k\| = 1$ for all $k > \bar{k}$, there exists a subsequence $\{\hat{q}^k\}_\kappa$ of $\{\hat{q}^k\}_{k>\bar{k}}$ and a vector $\bar{q}$ with $\|\bar{q}\| = 1$ such that $\{\hat{q}^k\}_\kappa \to \bar{q}$.

Applying the mean value theorem to (6) we have that, for each $k\in\kappa$, there exists some $\tilde{\alpha}^k\in[0,\hat{\alpha}^k]$ such that

$$f'(x^k+\tilde{\alpha}^k\hat{q}^k;\hat{q}^k) > -\gamma(v^k)^\top\hat{q}^k = \gamma f'(x^k;\hat{q}^k).$$

Passing to the limit for $k\in\kappa$, we see that $f'(\bar{x};\bar{q})\geq\gamma f'(\bar{x};\bar{q})$ and as $\gamma<1$, we conclude that

$$f'(\bar{x};\bar{q})\geq 0. \qquad (7)$$

On the other hand,

$$f'(x^k;\hat{q}^k) = \frac{f'(x^k;q^k_{\alpha^k\beta^{-1}})}{\|q^k_{\alpha^k\beta^{-1}}\|} = \frac{-(v^k)^\top q^k_{\alpha^k\beta^{-1}}}{\|q^k_{\alpha^k\beta^{-1}}\|},$$

and if we take the limit for $k\in\kappa$, we obtain

$$f'(\bar{x};\bar{q}) = \frac{\limsup -(v^k)^\top q^k_{\alpha^k\beta^{-1}}}{\limsup\|q^k_{\alpha^k\beta^{-1}}\|}.$$

It follows from (5) that the numerator is strictly negative, and also that the denominator is strictly positive (because if $\{\|q^k_{\alpha^k\beta^{-1}}\|\}\to 0$, then $\{\|v^k\|\}\to 0$). Therefore $f'(\bar{x};\bar{q})$ is negative which contradicts (7). □

References

Benson, J. S., & More, J. J. (2001). A limited memory variable metric method for bound constraint minimization.

Bertsekas, D. P. (1999). *Nonlinear programming.* Athena Scientific.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. Y. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing, 16*, 1190–1208.

Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. *ACL.*

Collins, M. (2000). Discriminative reranking for natural language parsing. *ICML* (pp. 175–182).

Darroch, J., & Ratcliff, D. (1972). Generalised iterative scaling for log-linear models. *Annals of Mathematical Statistics.*

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics.*

Gao, J., Andrew, G., Johnson, M., & Toutanova, K. (2007). A comparative study of parameter estimation methods for statistical NLP. *ACL.*

Goodman, J. (2004). Exponential priors for maximum entropy models. *ACL.*

Kazama, J., & Tsujii, J. (2003). Evaluation and extension of maximum entropy models with inequality constraints. *EMNLP.*

Lee, S.-I., Lee, H., Abbeel, P., & Ng, A. (2006). Efficient L1 regularized logistic regression. *AAAI-06.*

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. *CONLL.*

Minka, T. P. (2003). *A comparison of numerical optimizers for logistic regression* (Technical Report). Microsoft Research.

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. *ICML.*

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization.* Springer.

Perkins, S., & Theiler, J. (2003). Online feature selection using grafting. *ICML.*

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B.*

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw., 23*, 550–560.