Trust Region Newton Methods for Large-Scale Logistic Regression

Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science, National Taiwan University, Taipei 106, Taiwan

Ruby C. Weng CHWENG@NCCU.EDU.TW
Department of Statistics, National Chengchi University, Taipei 116, Taiwan

S. Sathiya Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research, California

Abstract

Large-scale logistic regression arises in many applications such as document classification and natural language processing. In this paper, we apply a trust region Newton method to maximize the log-likelihood of the logistic regression model. The proposed method uses only approximate Newton steps in the beginning, but achieves fast convergence in the end. Experiments show that it is faster than the commonly used quasi Newton approach for logistic regression. We also compare it with linear SVM implementations.

1. Introduction

The logistic regression model is useful for two-class classification. Given data $\mathbf{x}$ and weights $(\mathbf{w}, b)$, it assumes the following probability model

$$P(y = \pm 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-y(\mathbf{w}^T\mathbf{x} + b))},$$

where y is the class label. If training instances are $\mathbf{x}_i, i = 1, \ldots, l$ and labels are $y_i \in \{1, -1\}$, one estimates $(\mathbf{w}, b)$ by minimizing the negative log-likelihood:

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{l} \log(1 + e^{-y_i(\mathbf{w}^T\mathbf{x}_i + b)}).$$

There are numerous applications of logistic regression. It can be extended to a multi-class classification model, which is a special case of conditional random fields, and is also called the maximum entropy model in the natural language processing community.

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

To have a simpler derivation without considering the scalar b, one often augments each instance with an additional dimension:

$$\mathbf{x}_i^T \leftarrow [\mathbf{x}_i^T, 1] \qquad \mathbf{w}^T \leftarrow [\mathbf{w}^T, b].$$

Moreover, to obtain good generalization abilities, one adds a regularization term $\mathbf{w}^T\mathbf{w}/2$, so in this paper we consider the following form of regularized logistic regression:

$$\min_{\mathbf{w}} f(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \log(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i}), \quad (1)$$

where $C > 0$ is a parameter decided by users so that the two terms in (1) are balanced.

There are many methods for training logistic regression models. In fact, most unconstrained optimization techniques can be considered. Those which have been used in large-scale scenarios are, for example, iterative scaling (Pietra et al., 1997), nonlinear conjugate gradient, quasi Newton (in particular, limited memory BFGS) (Liu & Nocedal, 1989), and truncated Newton (Komarek & Moore, 2005). All these optimization methods are iterative procedures, which generate a sequence $\{\mathbf{w}^k\}_{k=1}^{\infty}$ converging to the optimal solution of (1). One can distinguish them according to the following two extreme situations of optimization methods:

Low cost per iteration; slow convergence.	$\longleftrightarrow$	High cost per iteration; fast convergence.

For instance, iterative scaling updates one component of $\mathbf{w}$ at a time, so the cost per iteration is low but the number of iterations is high. In contrast, Newton's method, which is expensive at each iteration, has very fast convergence rates. Many have attempted to compare these methods for logistic regression. Malouf (2002) has done an extensive comparison for large-scale sets. Currently, most argue that the limited

memory BFGS method is the most efficient and effective (e.g., Malouf (2002), Sutton and McCallum (2006) and references therein). In this article, we aim at situations for which both l (number of instances) and n (number of features) are very large. In addition, the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_l$ are sparse (i.e., many feature values are zero). Many recent applications from document classification and computational linguistics are of this type.

Truncated Newton methods have been an effective approach for large-scale optimization, but their use for logistic regression has not been fully exploited. Though Komarek and Moore (2005) have considered this type of methods, their implementation does not follow rigorous optimization derivations, and hence may not be guaranteed to obtain the minimum of (1). In Section 2, we discuss an efficient and robust truncated Newton method for logistic regression.

The second term in (1) can be considered as a loss function, so regularized logistic regression is related to other learning approaches such as Support Vector Machines (SVM) (Boser et al., 1992). L1-SVM solves the following optimization problem:

$$\min_{\mathbf{w}} f_1(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right),$$

while L2-SVM solves

$$\min_{\mathbf{w}} f_2(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \left(\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)\right)^2.$$

Figure 1 shows the different shapes of the three loss functions. SVM is often used with a nonlinear kernel, where the data vectors $\mathbf{x}_i$ are mapped to a high dimensional space. However, there are certain applications for which with/without nonlinear mapping give similar performances. For the case of no nonlinear mapping, we have the possibility of directly solving bigger optimization problems. We refer to such cases as *linear* SVM. Considerable efforts have been made on its fast training (e.g., Keerthi and DeCoste (2005); Joachims (2006)). L1-SVM involves the optimization of a non-differentiable function of $\mathbf{w}$, so unconstrained optimization techniques cannot be directly applied. For L2-SVM, the training objective function is differentiable but not twice differentiable (Mangasarian, 2002). A modified Newton method has been proposed in Keerthi and DeCoste (2005) for large problems. Logistic regression and SVM differ only in their loss functions, and they usually give similar performances. In Sections 3 and 4, we discuss existing optimization methods for logistic regression/linear

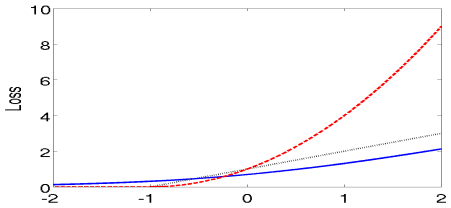


Figure 1: Different loss functions. Dotted (black): $\max(0, 1-y\mathbf{w}^T\mathbf{x})$, dashed (red): $(\max(0, 1-y\mathbf{w}^T\mathbf{x}))^2$, solid (blue): $\log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$. The x-axis is $-y\mathbf{w}^T\mathbf{x}$.

SVM and conduct comparisons. Results show that the proposed Newton's method is consistently better. Finally, Section 5 gives conclusions. A complete report of this work is at `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`

2. Trust Region Newton Methods

In this section, we discuss Newton and truncated Newton methods. For large-scale logistic regression, we then propose a trust region Newton method, which is a type of truncated Newton approach.

2.1. Newton & Truncated Newton Methods

To discuss Newton methods, we need the gradient and Hessian of $f(\mathbf{w})$:

$$\begin{aligned} \nabla f(\mathbf{w}) &= \mathbf{w} + C\sum_{i=1}^{l}(\sigma(y_i\mathbf{w}^T\mathbf{x}_i) - 1)y_i\mathbf{x}_i, \\ \nabla^2 f(\mathbf{w}) &= \mathcal{I} + CX^TDX, \end{aligned}$$

where $\mathcal{I}$ is the identity matrix,

$$\sigma(y_i\mathbf{w}^T\mathbf{x}_i) = (1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i})^{-1},$$

D is a diagonal matrix with

$$D_{ii} = \sigma(y_i\mathbf{w}^T\mathbf{x}_i)(1 - \sigma(y_i\mathbf{w}^T\mathbf{x}_i)), \text{ and } X = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_l^T \end{bmatrix} \quad (2)$$

is an $l \times n$ matrix. The Hessian matrix $\nabla^2 f(\mathbf{w})$ is positive definite. One can easily prove that (1) attains a unique global optimal solution.

Since $\nabla^2 f(\mathbf{w}^k)$ is invertible, the simplest Newton's method updates $\mathbf{w}$ by the following way

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{s}^k, \quad (3)$$

where k is the iteration index and $\mathbf{s}^k$, the Newton direction, is the solution of the following linear system:

$$\nabla^2 f(\mathbf{w}^k)\mathbf{s}^k = -\nabla f(\mathbf{w}^k). \quad (4)$$

However, there are two issues in using this update rule:

1. The sequence $\{\mathbf{w}^k\}$ may not converge to an optimal solution. In fact, even the function value may not be guaranteed to decrease.

2. While we assume that the data matrix X is sparse, $X^T DX$ is much denser. The Hessian matrix is then too large to be stored. Thus, solving the linear system (4) is an issue that needs careful consideration.

Optimization researchers address the first issue by adjusting the length of the Newton direction. Two techniques are often used: line search and trust region.

For the second issue, there are two major types of methods for solving linear systems: direct methods (e.g., Gaussian elimination), and iterative methods (e.g., Jacobi and conjugate gradient). The main operation of certain iterative methods is the product between the Hessian matrix and a vector $\mathbf{s}$:

$$\nabla^2 f(\mathbf{w})\mathbf{s} = (\mathcal{I} + CX^T DX)\mathbf{s} = \mathbf{s} + C \cdot X^T (D(X\mathbf{s})). \quad (5)$$

As we assume sparse X, (5) can be efficiently calculated without storing the Hessian matrix $\nabla^2 f(\mathbf{w}^k)$. Therefore, for large logistic regression, iterative methods are more suitable than direct methods, which require the whole Hessian matrix. Among iterative methods, currently conjugate gradients are the most used ones in Newton's methods. The optimization procedure then has two layers of iterations: at each outer iteration an inner conjugate gradient procedure finds the Newton direction. Unfortunately, conjugate gradient methods may suffer from lengthy iterations in certain situations. To save time, one may use only an "approximate" Newton direction in the early stages of the outer iterations. Such a technique is called truncated Newton method (or inexact Newton method).

Komarek and Moore (2005) are among the first to apply truncated Newton methods for logistic regression*. They approximately solve (4) by conjugate gradient procedures and use (3) to update $\mathbf{w}^k$. They terminate the conjugate gradient procedure if the relative difference of log likelihoods between two consecutive conjugate gradient iterations is smaller than a threshold. However, they do not provide a convergence proof. In fact, when we tried their code, we found that $\|\nabla f(\mathbf{w}^k)\|$ may not approach zero and hence $\{\mathbf{w}^k\}$ may not converge to an optimum†.

Optimization researchers have well addressed the above two issues together. They devise the procedure of outer iterations, and specify stopping conditions for the inner iterations. The overall framework guarantees the convergence to the global minimum. The truncation rule of the inner algorithm is important as one should stop after a sufficiently good direction has been found. A survey of truncated Newton methods is in Nash (2000). Some comparisons between limited memory quasi Newton and truncated Newton are in Nocedal and Nash (1991).

*They minimize only the negative log likelihood without the regularization term $\mathbf{w}^T\mathbf{w}/2$.

†We used the data set citeseer for this purpose. See Section 4 for a description of this dataset.

2.2. A Trust Region Newton Method

We consider the trust region method in Lin and Moré (1999), which is a truncated Newton method for bound-constrained problems (i.e., variables are in certain intervals). We simplify the setting to unconstrained situations, so the method is closer to Steihaug (1983) which is an earlier work meant for such cases.

At each iteration of a trust region Newton method for minimizing $f(\mathbf{w})$, we have an iterate $\mathbf{w}^k$, a size Δ_k of the trust region, and a quadratic model

$$q_k(\mathbf{s}) = \nabla f(\mathbf{w}^k)^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T \nabla^2 f(\mathbf{w}^k)\mathbf{s}$$

as the approximation of the value $f(\mathbf{w}^k + \mathbf{s}) - f(\mathbf{w}^k)$. Next, we find a step $\mathbf{s}^k$ to approximately minimize $q_k(\mathbf{s})$ subject to the constraint $\|\mathbf{s}\| \le \Delta_k$. We then update $\mathbf{w}^k$ and Δ_k by checking the ratio

$$\rho_k = \frac{f(\mathbf{w}^k + \mathbf{s}^k) - f(\mathbf{w}^k)}{q_k(\mathbf{s}^k)} \quad (6)$$

of the actual reduction in the function to the predicted reduction in the quadratic model. The direction is accepted if ρ_k is large enough:

$$\mathbf{w}^{k+1} = \begin{cases} \mathbf{w}^k + \mathbf{s}^k & \text{if } \rho_k > \eta_0, \\ \mathbf{w}^k & \text{if } \rho_k \le \eta_0, \end{cases} \quad (7)$$

where $\eta_0 > 0$ is a pre-specified value.

From Lin and Moré (1999), updating rules for Δ_k depend on positive constants η_1 and η_2 such that $\eta_1 < \eta_2 < 1$, while the rate at which Δ_k is updated relies on positive constants σ_1, σ_2, and σ_3 such that $\sigma_1 < \sigma_2 < 1 < \sigma_3$. The trust region bound Δ_k is updated by the rules

$$\begin{aligned} &\Delta_{k+1} \in [\sigma_1 \min\{\|\mathbf{s}^k\|, \Delta_k\}, \sigma_2 \Delta_k] && \text{if } \rho_k \le \eta_1, \\ &\Delta_{k+1} \in [\sigma_1 \Delta_k, \sigma_3 \Delta_k] && \text{if } \rho_k \in (\eta_1, \eta_2), \\ &\Delta_{k+1} \in [\Delta_k, \sigma_3 \Delta_k] && \text{if } \rho_k \ge \eta_2. \end{aligned} \quad (8)$$

Similar rules are used in other trust region methods. Our trust region method is given in the following table.

Algorithm 1 A trust region algorithm

Given $\mathbf{w}^0$.
For $k = 0, 1, \ldots$ (outer iterations)

- If $\nabla f(\mathbf{w}^k) = \mathbf{0}$, stop.
- Find an approximate solution $\mathbf{s}^k$ of the trust region sub-problem

$$\min_{\mathbf{s}} \ q_k(\mathbf{s}) \quad \text{subject to } \|\mathbf{s}\| \le \Delta_k. \tag{9}$$

- Compute ρ_k via (6).
- Update $\mathbf{w}^k$ to $\mathbf{w}^{k+1}$ according to (7).
- Obtain Δ_{k+1} according to (8).

The conjugate gradient method to approximately solve the sub-problem (9) is given in the following table.

Algorithm 2 Conjugate gradient procedure for approximately solving the trust region sub-problem (9)

Given $\xi_k < 1, \Delta_k > 0$. Let $\bar{\mathbf{s}}^0 = \mathbf{0}, \mathbf{r}^0 = -\nabla f(\mathbf{w}^k)$, and $\mathbf{d}^0 = \mathbf{r}^0$.
For $i = 0, 1, \ldots$ (inner iterations)

- If

$$\|\mathbf{r}^i\| = \|\nabla f(\mathbf{w}^k) + \nabla^2 f(\mathbf{w}^k)\bar{\mathbf{s}}^i\| \le \xi_k \|\nabla f(\mathbf{w}^k)\|, \tag{10}$$

  then output $\mathbf{s}^k = \bar{\mathbf{s}}^i$ and stop.
- $\alpha_i = \|\mathbf{r}^i\|^2 / ((\mathbf{d}^i)^T \nabla^2 f(\mathbf{w}^k)\mathbf{d}^i)$.
- $\bar{\mathbf{s}}^{i+1} = \bar{\mathbf{s}}^i + \alpha_i \mathbf{d}^i$.
- If $\|\bar{\mathbf{s}}^{i+1}\| \ge \Delta_k$, compute τ such that

$$\|\bar{\mathbf{s}}^i + \tau \mathbf{d}^i\| = \Delta_k. \tag{11}$$

  Output $\mathbf{s}^k = \bar{\mathbf{s}}^i + \tau \mathbf{d}^i$ and stop.
- $\mathbf{r}^{i+1} = \mathbf{r}^i - \alpha_i \nabla^2 f(\mathbf{w}^k)\mathbf{d}^i$.
- $\beta_i = \|\mathbf{r}^{i+1}\|^2 / \|\mathbf{r}^i\|^2$.
- $\mathbf{d}^{i+1} = \mathbf{r}^{i+1} + \beta_i \mathbf{d}^i$.

The main operation is $\nabla^2 f(\mathbf{w}^k)\mathbf{d}^i$, which is implemented using the idea in Eq. (5). Algorithm 2 is different from standard conjugate gradient methods for linear systems as the constraint $\|\mathbf{s}\| \le \Delta$ must be taken care of. If the conjugate gradient iterate $\bar{\mathbf{s}}^{i+1}$ violates the trust region constraint, (11) finds a point on the trust region boundary. This is a careful design to make sure that the approximate Newton direction is good enough and the trust region method converges. Next, we discuss convergence properties. For the sequence $\{\mathbf{w}^k\}$ to have at least one limit point (i.e, $\{\mathbf{w}^k\}$ has at least one convergent sub-sequence), since $f(\mathbf{w}^k)$ is decreasing, it suffices to show that the level set $\{\mathbf{w} \mid f(\mathbf{w}) \le f(\mathbf{w}^0)\}$ is closed and bounded. The proof of this for our problem is simple, but is omitted due to space limitation. To have the limit point to be the minimum, Theorem 2.1 of Lin and Moré (1999) requires that $\nabla^2 f(\mathbf{w}^k)$ is uniformly bounded. We have this property as $\nabla^2 f(\mathbf{w})$ is continuous in this bounded level set.

Eq. (10) is a relative stopping condition in solving a linear system. The parameter ξ_k effectively controls the computational effort associated with the inner iterations. By the explanation after the proof of Theorem 5.4 of Lin and Moré (1999), if $\xi_k < 1$, the trust region method Q-linearly converges to the global minimum of (1). That is,

$$\lim_{k\to\infty} \frac{\|\mathbf{w}^{k+1} - \mathbf{w}^*\|}{\|\mathbf{w}^k - \mathbf{w}^*\|} < 1, \tag{12}$$

where $\mathbf{w}^*$ is the unique optimal solution of (1). If $\xi_k \to 0$ as $k \to \infty$, then the limit in (12) becomes zero, so we have Q-superlinear convergence.

3. Related Methods

In this section, we discuss two related techniques, which will be compared in Section 4. The first is a modified Newton method for L2-SVM (Keerthi & DeCoste, 2005). It is now one of the fastest implementations in training linear SVMs. The second is Liu and Nocedal (1989), which is a general limited memory quasi Newton implementation. Many consider it as very efficient for training logistic regression. We also discuss implementation issues of the proposed trust region Newton method.

3.1. Modified Newton Method for L2-SVM

The key idea of Keerthi and DeCoste (2005) to solve L2-SVM is that for any given index set $I \subset \{1, \ldots, l\}$, if the optimal solution $\mathbf{w}^*$ of the following problem

$$\min_{\mathbf{w}} f_2(\mathbf{w}) \equiv \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i\in I}(1 - y_i\mathbf{w}^T\mathbf{x}_i)^2 \tag{13}$$

satisfies

$$1 - y_i(\mathbf{w}^*)^T\mathbf{x}_i \begin{cases} > 0 & \text{if } i \in I, \\ \le 0 & \text{if } i \notin I, \end{cases}$$

then $\mathbf{w}^*$ is an optimal solution of the L2-SVM problem. Once I is fixed, (13) is a simple regularized least square problem and can be solved by the following linear system:

$$(\mathcal{I} + 2CX_{I,:}^T X_{I,:})\mathbf{w} = 2CX_{I,:}^T \mathbf{y}_I, \tag{14}$$

where $X_{I,:}$ includes X's rows corresponding to the set I. Their algorithm is described in the following table.

Algorithm 3 Modified Newton Method for L2-SVM

Given $\mathbf{w}^0$.
For $k = 0, 1, \ldots$

- If $\nabla f(\mathbf{w}^k) = \mathbf{0}$, stop.
- Set up (14) using

$$I_k = \{i \mid 1 - y_i(\mathbf{w}^k)^T\mathbf{x}_i > 0\}.$$

  Solve (14) by conjugate gradients and obtain $\bar{\mathbf{w}}^k$.
- Let $\mathbf{s}^k = \bar{\mathbf{w}}^k - \mathbf{w}^k$. Find

$$\alpha_k = \arg\min_{\alpha \geq 0} f(\mathbf{w}^k + \alpha\mathbf{s}^k),$$

  and set $\mathbf{w}^{k+1} = \mathbf{w}^k + \alpha_k\mathbf{s}^k$.

Keerthi and DeCoste (2005) prove that Algorithm 3 converges to the optimal solution of in a finite number of iterations. Compared to our approach, this method has the following advantages and disadvantages.

Advantages: The final set I corresponds to the set of support vectors. Thus, the sets I_k may be small at final iterations, and solving (14) by conjugate gradient methods is fast.
Disadvantages:
1. Their convergence result assumes that at each iteration, (14) is exactly solved. However, they use a relative stopping condition in practical implementations, so the convergence remains an issue. In contrast, the convergence of trust region Newton method has been well studied under various early stopping conditions of the conjugate gradient procedure.
2. Using $X_{I,:}$ in (14) requires an easy access of X's rows. This restricts the way how X is stored. Our experiments show that sometimes an easy access of X's columns leads to shorter training time. For fairness in our later comparisons, all methods use the same way to store X, so that X's rows are easily accessed.

More discussions on the stopping conditions of Algorithm 3 are given in Section 4.2. To avoid exactly solving every instance of equation (14), one may also apply the trust region framework to L2-SVM. However, as L2-SVM is not twice differentiable, the convergence and the numerical behavior need to be investigated.

3.2. Limited Memory quasi Newton Method

We briefly introduce the approach in Liu and Nocedal (1989). Quasi Newton methods use approximate inverse Hessian H_k and can easily update it to H_{k+1}. One of the most popular updates is BFGS. The approach by Liu and Nocedal (1989) is almost the same as BFGS, but restricts the update to use only m vectors from the previous iterations. The matrix H_k is not formed explicitly and there is an efficient way to compute $H_k\nabla f(\mathbf{w}^k)$. This property is useful as we cannot afford to store H_k. Since only an approximate Hessian is used, this method sometimes suffers from slow convergence. The algorithm is in the following table.

Algorithm 4 Limited memory BFGS

Given $\mathbf{w}^0, H^0$ and a small integer m.
For $k = 0, 1, \ldots$

- If $\nabla f(\mathbf{w}^k) = \mathbf{0}$, stop.
- Use m vectors from previous iterations to calculate $H_k\nabla f(\mathbf{w}^k)$.
- Search α_k so that $f(\mathbf{w}^k - \alpha H_k\nabla f(\mathbf{w}^k))$ satisfies some sufficient decrease conditions. Obtain H_{k+1}.

Regarding convergence, problem (1) satisfies Assumption 6.1 of Liu and Nocedal (1989), so Algorithm 4 converges to the optimum of (1). For experiments, we use $m = 5$, which is the default choice in the LBFGS software (Liu & Nocedal, 1989).

3.3. Implementation Issues of Trust Region Newton Method

We give details of parameters in the proposed Algorithms 1 and 2. All settings are almost the same as the TRON software (Lin & Moré, 1999).

We set the initial $\Delta_0 = \|\nabla f(\mathbf{w}^0)\|$ and take $\eta_0 = 10^{-4}$ in (7) to update $\mathbf{w}^k$. For changing Δ_k to Δ_{k+1}, we use

$$\eta_1 = 0.25, \eta_2 = 0.75, \sigma_1 = 0.25, \sigma_2 = 0.5, \sigma_3 = 4.0.$$

As (8) specifies only the interval in which Δ_{k+1} should lie, there are many possible choices for the update rules. We use the same rules as given in Lin and Moré (1999). In the conjugate gradient procedure, we use $\xi_k = 0.1$ in the stopping condition (10). One may wonder how the above numbers are chosen. These choices are considered appropriate following the research on trust region methods in the past several decades. It is unclear yet if they are the best for logistic regression problems, but certainly we would like to try custom settings first.

4. Comparisons

4.1. Data Sets

We consider six data sets from various sources. Table 1 lists the numbers of instances, features, and nonzero feature values. Except citeseer, all other problems are quite balanced. It is known that unbalanced sets usu-

Table 1: Data statistics.

Problem	# instances	# features	# nonzeros
real-sim	72,309	20,958	3,709,083
news20	19,996	1,355,191	9,097,916
citeseer	181,395	105,354	512,267
yahoo-japan	176,203	832,026	23,506,415
rcv1	677,399	47,236	49,556,258
yahoo-korea	460,554	3,052,939	156,436,656

ally lead to shorter training time. Therefore, problems used in this article are more challenging in terms of training time. More details of the data sets are described below.

real-sim: This set is from the web site `http://people.cs.uchicago.edu/~vikass/datasets/lskm/svml/`.
news20: This is a collection of news documents. We use the data processed in Keerthi and DeCoste (2005). They consider binary term frequencies and normalize each instance to unit length.
citeseer: This set, obtained from `http://komarix.org/ac/ds/#spardat`, is a collection of research papers. Positive ones include those authored by "J. Lee."
yahoo-japan: This set, from Yahoo!, includes documents in hundreds of classes. We consider the class with the largest number of instances as positive and all remaining instances as negative. We use binary term frequencies and normalize each vector to unit length.
rcv1: This set (Lewis et al., 2004) contains newswire stories from Reuters Ltd. Each vector is a cosine normalization of a log transformed TF-IDF feature vector. The news documents are in a hierarchical structure of classes. We use the set at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary`. It splits the data to positive/negative by using the two branches in the first layer of the class hierarchy. Data which are multi-labeled are not considered.
yahoo-korea: This set, obtained from Yahoo!, includes documents in a hierarchy of classes. We consider the largest branch from the root node (i.e., the branch including the largest number of classes) as positive, and all others as negative.

Clearly, all sets are from document classification. Past experiences have shown that for such data, linear classifiers are often as good as kernalized ones. We find that normalizations are usually needed so that the length of each instance is not too large. Otherwise, when the number of features is large, $\mathbf{w}^T\mathbf{x}_i$ may be huge and cause difficulties in solving optimization problems. (For good performance also, such normalizations are usually needed.)

4.2. Comparisons

We compare four methods. Two solve logistic regression, one solves L2-SVM, and one solves L1-SVM.

TRON: the trust region Newton method discussed in Section 2.2.

LBFGS: the limited memory quasi Newton implementation (Liu & Nocedal, 1989). See the discussion in Section 3.2. The source code is available online at `http://www.ece.northwestern.edu/~nocedal`.

SVMLIN: modified Newton method for L2-SVM discussed in Section 3.1. The code (by Vikas Sindhwani) is at `http://people.cs.uchicago.edu/~vikass`.

SVM^{perf} (Joachims, 2006): it is an approximation algorithm for L1-SVM. A threshold controls how well the approximation is. Since this approach is very different from the other three, we conduct a separate comparison at the end of this sub-section.

We do not consider the code in Komarek and Moore (2005) because of two reasons. First, we have mentioned its convergence problems in Section 2.1. Second, for sparse data, it handles only those with 0/1 features, but most our data have real-numbered features. All sources used for our comparisons are available at `http://www.csie.ntu.edu.tw/~cjlin/liblr`.

For the initial $\mathbf{w}^0$, all the methods use $\mathbf{0}$. Regarding the stopping conditions, for TRON and LBFGS, we use

$$\|\nabla f(\mathbf{w}^k)\|_\infty \le 10^{-3}. \tag{15}$$

Though SVMLIN minimizes a different function $f_2(\mathbf{w})$, a similar condition should be $\|\nabla f_2(\mathbf{w}^k)\|_\infty \le 10^{-3}$. Note that

$$\nabla f_2(\mathbf{w}) = (\mathcal{I} + 2CX_{I,:}^T X_{I,:})\mathbf{w} - 2CX_{I,:}^T \mathbf{y}_I,$$

where $I = \{i \mid 1 - y_i\mathbf{w}^T\mathbf{x}_i > 0\}$ is an index set depending on $\mathbf{w}$. Recall in Algorithm 3 that we sequentially obtain the following items:

$$\mathbf{w}^k \to I_k \to \bar{\mathbf{w}}^k.$$

In practice we use

$$\|(\mathcal{I} + 2CX_{I_k,:}^T X_{I_k,:})\bar{\mathbf{w}}^k - 2CX_{I_k,:}^T \mathbf{y}_{I_k}\|_\infty \le 10^{-3}, \tag{16}$$

and

$$1 - y_i\bar{\mathbf{w}}^k\mathbf{x}_i \begin{cases} \ge -10^{-3} & \text{if } i \in I_k, \\ \le 10^{-3} & \text{if } i \notin I_k, \end{cases} \tag{17}$$

as the stopping condition. We also use (16) as the stopping rule when solving (14) by conjugate gradients. The default setting of SVMLIN is similar, but a relative stopping condition is used for (16).

It is important to check the prediction ability of logistic regression and L2-SVM, so we first report cross-validation (CV) accuracy. For the larger sets (yahoo-japan, rcv1, and yahoo-korea), we use two-fold CV. For

Table 2: Comparison of three methods. Here time (in seconds) is the total training time in the CV procedure. As TRON and LBFGS minimize the same formulation and their CV accuracy values are almost the same, we present only the result of TRON. The number of CV folds is five for small problems, and is two for larger ones (yahoo-japan, rcv1, yahoo-korea). Note for both models, the CV values do not increase using $C > 16$.

	Logistic regression			L2-SVM	
	TRON		LBFGS	SVMLIN	
C	CV	Time	Time	CV	Time
0.25	95.85%	10	32	97.43%	20
1	96.97%	16	63	97.53%	29
4	97.41%	23	96	97.24%	47
16	97.51%	35	175	96.86%	92

(a) real-sim

	Logistic regression			L2-SVM	
	TRON		LBFGS	SVMLIN	
C	CV	Time	Time	CV	Time
0.25	89.74%	62	187	95.78%	79
1	93.36%	96	434	96.72%	147
4	95.49%	144	806	96.90%	270
16	96.30%	198	1424	96.86%	495

(b) news20

	Logistic regression			L2-SVM	
	TRON		LBFGS	SVMLIN	
C	CV	Time	Time	CV	Time
0.25	99.84%	9	15	99.85%	19
1	99.84%	14	28	99.85%	21
4	99.84%	20	50	99.80%	100
16	99.85%	32	83	99.72%	492

(c) citeseer

	Logistic regression			L2-SVM	
	TRON		LBFGS	SVMLIN	
C	CV	Time	Time	CV	Time
0.25	91.91%	60	176	92.87%	144
1	92.50%	96	337	93.00%	290
4	92.81%	163	702	92.71%	651
16	92.86%	283	1184	92.14%	1396

(d) yahoo-japan

	Logistic regression			L2-SVM	
	TRON		LBFGS	SVMLIN	
C	CV	Time	Time	CV	Time
0.25	97.18%	78	241	97.76%	131
1	97.56%	122	988	97.74%	275
4	97.72%	181	946	97.53%	595
16	97.69%	261	1422	97.07%	1370

(e) rcv1

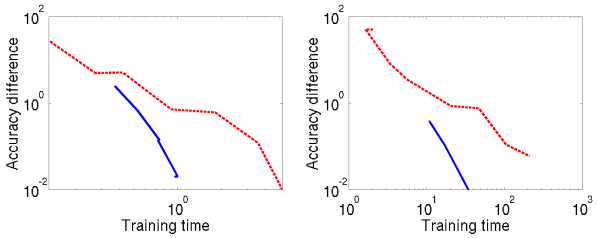
	Logistic regression			L2-SVM	
	TRON		LBFGS	SVMLIN	
C	CV	Time	Time	CV	Time
0.25	81.34%	528	2512	85.43%	1734
1	84.03%	912	4368	86.48%	3813
4	85.75%	1771	8937	86.38%	8896
16	86.40%	4475	14074	85.66%	19751

(f) yahoo-korea

others, five-fold CV is conducted. We do not consider other measurements such as Area Under Curve or F-measure as all problems except citeseer are rather balanced, and CV accuracy is suitable. Moreover, different values of the regularization parameter C may affect the performance as well as training time. So we check different values and present results of $C = 0.25$, 1, 4, and 16 in Table 2. Both logistic regression and L2-SVM achieve their best CV in this range of C values. In the table we show CV accuracy and the total training time in the CV procedure.

Regarding CV values, for small C, L2-SVM is slightly better than logistic regression. However, as C increases, the two give similar CV values. Overall, the two learning models give comparable generalization performance. On training time TRON is better than LBFGS, so truncated Newton methods are effective for training logistic regression. If we take TRON and SVMLIN, though it may not be right to compare the training speed using the same C, TRON has the advantage of fast convergence. Note that as C increases, the stopping condition (15) is more strict. But TRON's training time does not increase as much as that of SVMLIN. Thus, TRON does not easily get into a situation of long training time. Both approaches spend most of their time on the operation (5) in the conjugate gradient procedure. Since SVMLIN accurately solves Newton directions in early iterations, many conjugate gradient iterations are wasted.

In the above comparisons, we do not include SVM^{perf} because it significantly differs from the other three codes, which follow more traditional optimization



(a) real-sim (b) rcv1

Figure 2: A comparison between TRON (blue solid line) and SVM^{perf} (red dotted line). The y-axis shows the accuracy differences between current models and that of using the stopping tolerance 0.1. The x-axis (training time) is in seconds. We use $C = 1$. One must be careful that SVM^{perf}'s C parameter is different. We take our C value and set SVM^{perf}'s C parameter as $Cl/100$.

derivations. SVM^{perf} uses a threshold to control how well the obtained solution approximates the optimal one. This threshold is different from the tolerance used in the stopping condition (15). A suitable comparison is to check the relationship between training time and testing accuracy. In other words, how quickly the algorithm can achieve an accuracy close to that of using the minimum of the optimization problem. We conduct the following experiment. A data set is equally split into training and testing sets. We then run SVM^{perf} with its default threshold 0.1, and TRON with the stopping condition $\|\nabla f(\mathbf{w}^k)\|_\infty \le 0.1$ to obtain their respective models and testing accuracy. Since accuracy stabilizes for tolerances smaller than 0.1, the accuracies obtained from these solutions can be taken to be very close to those of the true minima. So we check against these accuracy values as training time increases

from zero. Figure 2 shows the results for real-sim and rcv1. We do not present results for other data sets as for unknown reasons SVM^{perf} needs lengthy time for predicting data with a large number of features (e.g., news20 and yahoo-japan).

From Figure 2, we see that SVM^{perf} quickly obtains models with some accuracy values, but the convergence is slow. For TRON, it needs at least a certain amount of time to have the first model, but the accuracy of this model is already close to that of the optimal one. The final convergence is also faster. The experiment indicates that unless the problem is extremely huge so that limits on computational time force one to use very early sub-optimal stopping, (e.g., the accuracy is still 10% away from that of the final model), TRON is a more effective method. We use $C = 1$ for this experiment. Results of using other C values show similar observations.

4.3. Stopping Tolerance

The stopping tolerance used in Section 4.2 is 10^{-3}, but one may wonder whether it is too strict. We thus compare the three methods (TRON, LBFGS, and SVMLIN) using two larger stopping tolerances: 0.1 and 10. We summarize some observations in the following:

Stopping tolerance 0.1: All CV accuracy values are almost the same, so for these data sets, a tolerance value of 0.1 is usually small enough. The training time of all three approaches is less. Overall, TRON is the fastest among the three methods.

Stopping tolerance 10: For some cases, SVMLIN gives an erroneous CV accuracy. The reason is that the index set I is wrongly identified under the loose stopping condition (16)-(17). Thus, it is possible that in (17), one should use a smaller tolerance than in (16). In contrast, TRON and LBFGS yield CV values close to those by using tolerances 0.1, since their gradient function is calculated in a more straightforward way.

5. Discussion and Conclusions

The experiments indicate that because our method (TRON) only has to approximately find the Newton direction, it is faster than the Newton method for L2-SVM (Keerthi & DeCoste, 2005). We may also apply the same trust region strategy to their formulation. However, as L2-SVM is not twice differentiable, theoretical convergence and implementation issues must be investigated. This is a useful future research topic.

In summary, we have shown that a trust region Newton method is effective for training large-scale logistic regression problems. The method has nice optimization properties following past developments for large-scale unconstrained optimization. It is interesting that we do not need many special settings for logistic regression; a rather direct use of modern trust region techniques already yields excellent performances.

References

Boser, B. E., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *COLT*.

Joachims, T. (2006). Training linear SVMs in linear time. *ACM KDD*.

Keerthi, S. S., & DeCoste, D. (2005). A modified finite Newton method for fast solution of large scale linear SVMs. *JMLR*, *6*, 341–361.

Komarek, P., & Moore, A. W. (2005). *Making logistic regression a core data mining tool* (Technical Report). Carnegie Mellon University.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *JMLR*, *5*, 361–397.

Lin, C.-J., & Moré, J. J. (1999). Newton's method for large-scale bound constrained problems. *SIAM Journal on Optimization*, *9*, 1100–1127.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*, 503–528.

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

Mangasarian, O. L. (2002). A finite Newton method for classification. *Optimization Methods and Software*, *17*, 913–929.

Nash, S. G. (2000). A survey of truncated-Newton methods. *Journal of CAM*, *124*, 45–59.

Nocedal, J., & Nash, S. G. (1991). A numerical study of the limited memory BFGS method and the truncated-newton method for large scale optimization. *SIAM Journal on Optimization*, *1*, 358–372.

Pietra, S. D., Pietra, V. D., & Lafferty, J. (1997). Inducing features of random fields. *IEEE PAMI*, *19*, 380–393.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, *20*, 626–637.

Sutton, C., & McCallum, A. (2006). An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar (Eds.), *Introduction to statistical relational learning*. MIT Press.