Parallel Trust Region Newton Method for Logistic Regression

Chih Lee

chih.lee@uconn.edu

Computer Science and Engineering Department, University of Connecticut
A classification algorithm

Given the number of classes $K$, it models the posterior probability as follows.

$$\log \frac{\Pr(G = 1|X = x)}{\Pr(G = K|X = x)} = w_1^T x,$$

$$\log \frac{\Pr(G = 2|X = x)}{\Pr(G = K|X = x)} = w_2^T x,$$

$$\vdots$$

$$\log \frac{\Pr(G = K - 1|X = x)}{\Pr(G = K|X = x)} = w_{K-1}^T x,$$

where $G$ is the random variable for class label, $X$ is the random vector representing the features of a sample and $w_i$'s are the model parameters to be estimated.
Consider the case when $K = 2$ for simplicity.

The posterior probability function

$$\Pr(G = y | X = x) = \frac{1}{1 + e^{-y w^T x}},$$

where $y \in \{1, -1\}$ is the class label.

Given a set of $l$ labeled samples, logistic regression finds the coefficients $w^*$ by maximizing the logarithm of the posterior probability of the data.
\[ w^* = \arg \max_w \sum_{i=1}^l \log(1 + e^{-y_i w^T x_i}) \]

\[ = \arg \min_w \sum_{i=1}^l \log(1 + e^{-y_i w^T x_i}), \]

where \( x_i \) and \( y_i \) are the feature vector and label of sample \( i \).

**Regularized cost function**

\[ f(w) = \frac{1}{2} w^T w + C \sum_{i=1}^l \log(1 + e^{-y_i w^T x_i}), \]  

(1)

where \( C > 0 \) is a parameter.
The gradient and the Hessian of the cost function

\[ \nabla f(w) = w + C \sum_{i=1}^{l} (\sigma(y_i w^T x_i) - 1)y_i x_i, \quad (2) \]

\[ \nabla^2 f(w) = I + CX^T DX, \quad (3) \]

- \( I \) is the identity matrix
- \( \sigma(y_i w^T x_i) = \frac{1}{1+e^{-y_i w^T x_i}} \)
- \( D \) is a diagonal matrix such that
  \[ D_{ii} = \sigma(y_i w^T x_i)(1 - \sigma(y_i w^T x_i)) \]

\[ X = \begin{bmatrix}
  x_1^T \\
  \vdots \\
  x_l^T
\end{bmatrix} \text{ is an } l \times n \text{ matrix.} \]
Trust Region Newton Method (TRON)
The state-of-the-art algorithm TRON by Lin et al. [1] for solving large-scale logistic regression problems. Outperforms the limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) method, which was argued to be the most efficient and effective one.
TRON is a Newton method using an approximate direction $s^k$ to $-\left(\nabla^2 f(w^k)\right)^{-1}\nabla f(w^k)$ at iteration $k$.

$$s^k = \arg\min_s q^k(s) \text{ subject to } ||s|| \leq \Delta_k,$$  \hspace{1cm} (4)

where

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

$w^k$ and $\Delta_k$ is updated according to the ratio

$$\rho_k = \frac{f(w^k + s^k) - f(w^k)}{q^k(s^k)}$$ \hspace{1cm} (5)

of the actual reduction in the cost function to the predicted reduction in the quadratic approximation.
\[ w^{k+1} = \begin{cases} 
  w^k + s^k & \text{if } \rho_k > \eta_0, \\
  w^k & \text{if } \rho_k \leq \eta_0,
\end{cases} \]

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

\[ \Delta^{k+1} \in [\sigma_1 \min\{||s^k||, \Delta_k\}, \sigma_2 \Delta_k] \quad \text{if } \rho_k \leq \eta_1, \]
\[ \Delta^{k+1} \in [\sigma_1 \Delta_k, \sigma_3 \Delta_k] \quad \text{if } \rho_k \in (\eta_1, \eta_2), \]
\[ \Delta^{k+1} \in [\Delta_k, \sigma_3 \Delta_k] \quad \text{if } \rho_k \geq \eta_2, \]

where \( 0 < \eta_1 < \eta_2 < 1 \) and \( 0 < \sigma_1 < \sigma_2 < 1 < \sigma_3 \).
Procedure TRON

1: Given $w^0$.
2: for $k = 0, 1, \ldots$ do {outer iterations}
3: \hspace{1em} if $\nabla f(w^k) = 0$ then
4: \hspace{2em} return $w^k$.
5: \hspace{1em} end if
6: \hspace{1em} Find an approximate solution $s^k$ to the trust region sub-problem
7: \hspace{2em} $\min_{s} q_k(s)$ subject to $||s|| \leq \Delta_k$
8: \hspace{2em} using TRCG.
9: \hspace{2em} $\rho_k \leftarrow \frac{f(w^k + s^k) - f(w^k)}{q_k(s^k)}$.
10: \hspace{1em} Update $w^k$ to $w^k$ according to (6).
11: Obtain $\Delta_{k+1}$ with (7).
12: end for
Procedure TRCG

1: Given $\xi_k < 1$ and $\Delta_k > 0$. $\bar{s}^0 \leftarrow 0$, $r^0 \leftarrow -\nabla f(w^k)$ and $d^0 \leftarrow r^0$.
2: for $i = 0, 1, \ldots$ do {inner iterations}
3: if $||r^i|| \leq \xi_k ||\nabla f(w^k)||$ then
4: return $\bar{s}^i$.
5: end if
6: $\alpha_i \leftarrow \frac{||r^i||^2}{(d^i)^T \nabla^2 f(w^k) d^i}$.
7: $\bar{s}^{i+1} \leftarrow \bar{s}^i + \alpha_i d^i$.
8: if $||\bar{s}^{i+1}|| \geq \Delta_k$ then
9: Compute $\tau \geq 0$ such that $||\bar{s}^i + \tau d^i|| = \Delta_k$.
10: return $\bar{s}^i + \tau d^i$.
11: end if
12: $r^{i+1} \leftarrow r^i - \alpha_i \nabla^2 f(w^k) d^i$.
13: $\beta_i \leftarrow ||r^{i+1}||^2 / ||r^i||^2$.
14: $d^{i+1} \leftarrow r^{i+1} + \beta_i d^i$.
15: end for
Distribution of Data

- By sample: $X$ is stored a sample a line in the data file. The non-zeros in a sample are represented as (index, value) pairs.
  - Perfect shuffling
  - By number of non-zeros (did not investigate)

- By variable: $X$ is stored a variable a line in the data file. The non-zeros in a variable are represented as (index, value) pairs.
  - Perfect shuffling
  - By number of non-zeros
## Amount of Communication

<table>
<thead>
<tr>
<th>Function</th>
<th>Distribution of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>By Variable</td>
</tr>
<tr>
<td>$f(w)$</td>
<td>$(1 + l)$ Allreduce($1^a$)</td>
</tr>
<tr>
<td>$\nabla f(w)$</td>
<td>0</td>
</tr>
<tr>
<td>$\nabla^2 f(w)s$</td>
<td>$l$ Allreduce($1$)</td>
</tr>
<tr>
<td>DDOT, DNRM2</td>
<td>Allreduce($1$)</td>
</tr>
<tr>
<td>DAXPY, DSCAL</td>
<td>0</td>
</tr>
</tbody>
</table>

$^a$Message size in **double** (8 bytes).

$^b$Provided that $f(w)$ is evaluated right before it.

$^c$Provided that $\nabla f(w)$ is evaluated right before it.
Locality

- \( X \) is stored as a sparse matrix (compressed columns/rows).

- Computation of \( Xv \) and \( X^Tv \) is required by \( f(w) \), \( \nabla f(w) \) and \( \nabla^2 f(w)s \), where \( v \) and \( s \) are compatible vectors.

- By sample: poor spatial locality for large \( n \).

- By variable: poor spatial locality for large \( l \).
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>l</th>
<th># Positive</th>
<th># Negative</th>
<th>n</th>
<th># nonzeros</th>
<th>l/n</th>
</tr>
</thead>
<tbody>
<tr>
<td>a9a</td>
<td>32,561</td>
<td>7,841</td>
<td>24,720</td>
<td>123</td>
<td>451,592</td>
<td>254.7</td>
</tr>
<tr>
<td>real-sim</td>
<td>72,309</td>
<td>22,238</td>
<td>50,071</td>
<td>20,958</td>
<td>3,709,083</td>
<td>3.45</td>
</tr>
<tr>
<td>news20</td>
<td>19,996</td>
<td>9,999</td>
<td>9,997</td>
<td>1,355,191</td>
<td>9,097,916</td>
<td>0.015</td>
</tr>
<tr>
<td>rcv1</td>
<td>677,399</td>
<td>355,460</td>
<td>321,939</td>
<td>47,236</td>
<td>49,556,258</td>
<td>14.34</td>
</tr>
</tbody>
</table>

- **Binary-class datasets**

- $l \gg n$ or $l/n \gg 1$: Distributing data by sample should give better performance.

- $l \ll n$ or $l/n \ll 1$: Distributing data by variable should give better performance.
Perfect Shuffling v.s. By NNZs

![Graph for a9a dataset showing standard deviation against the number of processes.](image1)

![Graph for real-sim dataset showing standard deviation against the number of processes.](image2)

![Graph for news20 dataset showing standard deviation against the number of processes.](image3)

![Graph for rcv1 dataset showing standard deviation against the number of processes.](image4)
Results
rcev1

Parallel Trust Region Newton Method for Logistic Regression – p. 21/26
LibNBC

- Not stable
- Produces “unaligned access” warning messages when a call to MPI_Bcast() is replaced by NBC_Ibcast() followed by NBC_Wait().
- After substituting the nonblocking collective operations offered by LibNBC for the blocking counterparts, the resulting program does not yield the desired/correct output.
- Nonblocking collective operations will be included in MPI-3.
Decomposition of a dataset depends on its nature ($l$ and $n$).

- $l \gg n$: distribute data by sample.
- $l \ll n$: distribute data by variable.

Balancing the NNZs among processors does not lower execution/training time as expected.

Consider $X_1$, $X_2$: $l \times n_1$, $l \times n_2$ matrices with the same NNZs and $v$: $l \times 1$ dense vector.

- $X_1^Tv$ has much lower cache miss (mostly incurred by accesses to elements of $v$) rate than $X_2^Tv$ when $l$ is large and $X_1$ is much denser than $X_2$ ($n_1 \ll n_2$).

The same applies to $X_1u_1$ and $X_2u_2$, where $u_i$’s are compatible dense vectors.
Conclusions

- Scalable parallel TRON for logistic regression problems. Linear speedup for the two larger datasets.
- Distribution of a dataset depends on its nature.
- Without communication latency hiding in the training process, the observed speedup is likely due to improved cache performance and the architecture of SGI Altix 3700.


Questions?

Thanks for your attention.