Split Bregman and Linearized Split Bregman for Tuning Neural Network

Ruohan Zhan

August 21, 2016

Abstract I I

This work compares the Split Bregman Iteration(SBI) and Linearized Split Bregman Iteration(Linearized SBI)'s behavior when tuning neural network for classification. SVM loss(piecewise linear for fidelity term) is used. Warm start(without subgradient updating) can achieve a significant error reduction for either SBI(exactly solve each subproblem in warm start) or Linearized SBI(gradient descend solve each subproblem in warm start).

Abstract II

Linearized Split Bregman iteration:
pros: computational efficiency due to no inverse calculation, experimentally consumes nearly half the time of SBI at one iteration. This advantage will be more obvious when more hidden nodes are used, since the inverse calculation will be more difficult.
cons:

- at the cost of error decay rate(now Nesterov acceleration is taken into account, but still can not be comparable to SBI loss decaying rate).
- learning rate can be an issue to ensure the algorithm running stably and decaying error fast. The result can be oscillating due to large learning rate, while small learning rate results in little error decaying.

Split Bregman Iteration:
pros:

- no learning rate issue and can ensure the loss function's value decrease at each iteration.
- faster error reducing.

cons: computational burden due to inverse calculation.

Abstract III

Things Presented in the Slides

- Algorithms
  1. warm up for split bregman iteration and split bregman iteration;
  2. warm up for linearized split bregman iteration and linearized split bregman iteration;
- Numerical Results
  1. simple data's binary classification based on an average result of 10 experiments;
  2. binary classification for MNIST subdataset including 0s and 1s;
  3. MNIST dataset including 10 classes tested on SBI on an average result of 8 experiments.
- Conclusions and Open Questions.

Feedforward Neural Network

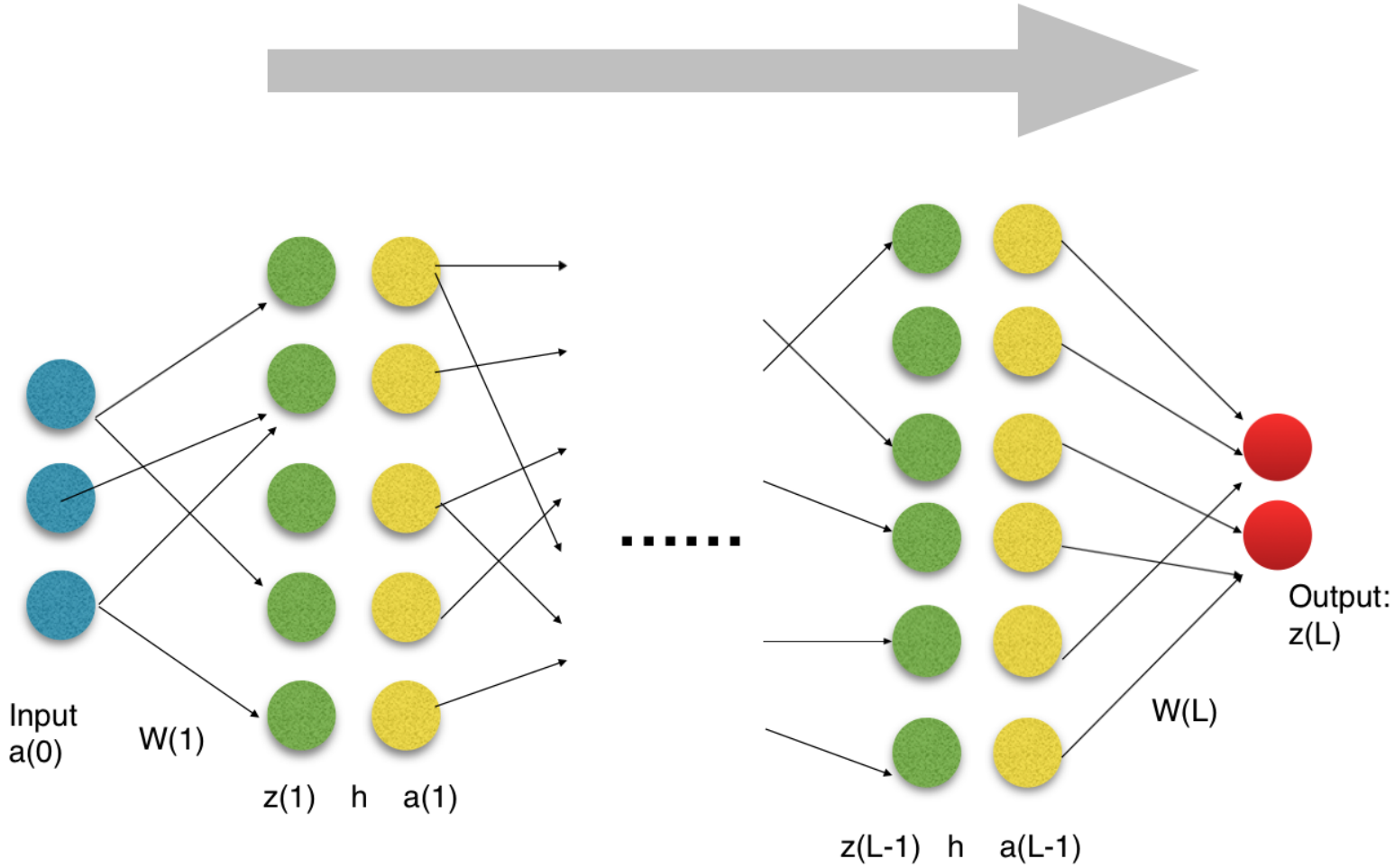


Figure: Feedforward Neural Network

SVM Loss Function

Denote $X = (a_1, \ldots, a_{L-1}, z_1, \ldots, z_{L-1}, W_1, \ldots, W_L)$. Y is the ground truth output. For multi-class classification, the SVM Loss Function is:

$$
\begin{aligned}
\min_{z_L, X} \quad & L(z_L, X) = \ell_1(z_L, Y) + \ell_2(W) \\
& = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, z_L(i)_j - z_L(i)_i + 1) + \frac{\lambda}{2} \sum_{l} \sum_{l} \|W_l\|^2 \\
\text{s.t.} \quad & z_l = W_l a_{l-1}, \quad a_l = h_l(z_l), \quad l = 1, \ldots, L.
\end{aligned}
\tag{1}
$$

Split Bregman Iteration

Denote $F(z_L, X) =$
$\frac{\beta}{2}\|z_L - W_L a_{L-1}\|_2^2 + \sum_{l=1}^{L-1}[\frac{\gamma}{2}\|a_l - h_l(z_l)\|_2^2 + \frac{\beta}{2}\|z_l - W_l a_{l-1}\|_2^2]$..
The corresponding split bregman iteration is:

For $l = 1, 2, \ldots, L-1$,

$$\begin{cases} W_l^{(k+1)} = \min\limits_{W_l} \dfrac{\lambda}{2}\|W_l\|^2 + \dfrac{\beta}{2}\|z_l^{(k)} - W_l a_{l-1}^{(k)}\|_2^2] \\ z_l^{(k+1)} = \min\limits_{z_l} \dfrac{\gamma}{2}\|a_l^{(k)} - h_l(z_l)\|_2^2 + \dfrac{\beta}{2}\|z_l - W_l^{(k+1)} a_{l-1}^{(k)}\|_2^2 \\ a_l^{(k+1)} = \dfrac{\gamma}{2}\|a_l - h_l(z_l^{(k+1)})\|_2^2 + \dfrac{\beta}{2}\|z_{l+1}^{(k+1)} - W_{l+1}^{(k+1)} a_l\|_2^2 \end{cases}$$

$$z_L^{(k+1)} = \min_{z_L} \ell_1(z_L, Y) - \langle p^{(k)}, z_L - z_L^{(k)}\rangle + \frac{\beta}{2}\|z_L - W_L^{(k+1)} a_{L-1}^{(k+1)}\|_2^2$$

$$p^{(k+1)} = p^{(k)} + \beta(W_L^{(k+1)} a_{L-1}^{(k+1)} - z_L^{(k+1)}) \tag{2}$$

while p is the subgradient of $\ell_1(z_L, Y)$.

Warm Start for Split Bregman Algorithm

Algorithm 1: Warm Start for Split Bregman

Input: training feature $\{a_0\}$, and labels $\{y\}$,
Initialize: allocate $\{W_l\}_{l=1}^{L}$, calculate
$z_l = W_l a_{l-1}, a_l = h_l(z_l), l = 1, 2, \ldots, L.$
repeat
for *l=1,2,. . . ,L-1* **do**

$$W_l^{(k+1)} = (\beta z_l^{(k)})(\beta a_{l-1}^{(k)} + \lambda I)^{-1}$$
$$z_l^{(k+1)} : \gamma(h(z_l^{k+1}) - a_l^{(k)})\nabla h(z_l) + \beta(z_l - W_l^{(k+1)} a_{l-1}^{k}) = 0$$
$$a_l^{k+1} = (\gamma I + \beta(W_{l+1}^{(k+1)})^T W_{l+1}^{(k+1)})^{-1}(\gamma h(z_l^{k+1}) + \beta W_{l+1}^{(k+1)} z_{l+1}^{(k+1)}) \tag{3}$$

$$W_L^{k+1} = (\beta z_L^{(k)})(\beta a_{L-1}^{(k)} + \lambda I)^{-1}$$
$$z_L^{(k+1)} : \nabla \ell_1(z_L, Y) + \beta(z_L - W_l^{(k+1)} a_{l-1}^{k}) = 0 \tag{4}$$

Until converged

Split Bregman Algorithm

Algorithm 2: Split Bregman for Neural Nets

Input: training feature $\{a_0\}$, and labels $\{y\}$,
Initialize: allocate $\{W_l\}_{l=1}^{L}$, calculate
$z_l = W_l a_{l-1}, a_l = h_l(z_l), l = 1, 2, \ldots, L.$
repeat
for *l=1,2,. . . ,L-1* **do**

$$
\begin{aligned}
W_l^{(k+1)} &= (\beta z_l^{(k)})(\beta a_{l-1}^{(k)} + \lambda I)^{-1} \\
z_l^{(k+1)} &: \gamma(h(z_l^{k+1}) - a_l^{(k)}) \nabla h(z_l) + \beta(z_l - W_l^{(k+1)} a_{l-1}^{k}) = 0 \\
a_l^{k+1} &= (\gamma I + \beta(W_{l+1}^{(k+1)})^T W_{l+1}^{(k+1)})^{-1}(\gamma h(z_l^{k+1}) + \beta W_{l+1}^{(k+1)} z_{l+1}^{(k+1)})
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
W_L^{k+1} &= (\beta z_L^{(k)})(\beta a_{L-1}^{(k)} + \lambda I)^{-1} \\
z_L^{(k+1)} &: \nabla \ell_1(z_L, Y) + \beta z_L = \beta W_l^{(k+1)} a_{l-1}^{k} + p^{(k)} \\
p^{(k+1)} &= p^{(k)} + \beta(W_L^{(k+1)} a_{L-1}^{(k+1)} - z_L^{(k+1)})
\end{aligned} \tag{6}
$$

Until converged

Warm Start for Linearized Split Bregman Iteration

Linearize Split Bregman Iteration *or* use ISS, we can get the linearized split bregman iteration based on gradient descend updating

Warm Start for Linearized Split Bregman Iteration

Algorithm 3: Warm Start for Linearized Split Bregman

Input: training feature $\{a_0\}$, and labels $\{y\}$,
Initialize: allocate $\{W_l\}_{l=1}^{L}$, calculate $z_l = W_l a_{l-1}, a_l = h_l(z_l), l = 1, 2, \ldots, L$.
repeat
for *l=1,2,. . . ,L-1* **do**

$$
\begin{aligned}
W_l^{(k+1)} &= W_l^{(k)} - \kappa(\beta_l(W_l^{(k)} a_{l-1}^{(k)}(a_{l-1}^{(k)})^T - z_l^{(k)}(a_{l-1}^{(k)})^T)) \\
z_l^{(k+1)} &= z_l^{(k)} - \kappa(\gamma_l \nabla h(z_l^{(k)})(h(z_l^{(k)}) - a_l^{(k)}) + \beta_l(z_l^{(k)} - W_l^{(k)} a_{l-1}^{(k)})) \\
a_l^{(k+1)} &= a_l^{(k)} - \kappa(\gamma_l(a_l^{(k)} - h_l(z_l^{(k)})) + \beta_{l+1}((W_{l+1}^{(k)})^T W_{l+1}^{(k)} a_l^{(k)} - (W_{l+1}^{(k)})^T z_{l+1}^{(k)}))
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
W_L^{(k+1)} &= W_L^{(k)} - \kappa(\beta_L(W_L^{(k)} a_{L-1}^{(k)}(a_{L-1}^{(k)})^T - z_L^{(k)}(a_{L-1}^{(k)})^T)) \\
z_L^{(k+1)} &= \mathrm{Prox}_{\kappa L}(z_L^{(k)} - \kappa\beta(z_L^{(k)} - W_L^{(k+1)} a_{L-1}^{(k+1)}))
\end{aligned}
\tag{8}
$$

Until converged

where solving z_L is equivalent to solve the following equation:

$$
p^{(k+1)} + \frac{1}{\kappa} z_L^{(k+1)} = \frac{1}{\kappa}(z_L^{(k)} - \kappa\beta(z_L^{(k)} - W_L^{(k+1)} a_{L-1}^{(k+1)})).
$$

Linearized Split Bregman Iteration

Algorithm 4: Linearized Split Bregman for Neural Nets

Input: training feature $\{a_0\}$, and labels $\{y\}$,
Initialize: allocate $\{W_l\}_{l=1}^{L}$, calculate $z_l = W_l a_{l-1}, a_l = h_l(z_l), l = 1, 2, \ldots, L$.
repeat
for *l=1,2,. . . ,L-1* **do**

$$
\begin{aligned}
W_l^{(k+1)} &= W_l^{(k)} - \kappa(\beta_l(W_l^{(k)} a_{l-1}^{(k)}(a_{l-1}^{(k)})^T - z_l^{(k)}(a_{l-1}^{(k)})^T)) \\
z_l^{(k+1)} &= z_l^{(k)} - \kappa(\gamma_l \nabla h(z_l^{(k)})(h(z_l^{(k)}) - a_l^{(k)}) + \beta_l(z_l^{(k)} - W_l^{(k)} a_{l-1}^{(k)})) \\
a_l^{(k+1)} &= a_l^{(k)} - \kappa(\gamma_l(a_l^{(k)} - h_l(z_l^{(k)})) + \beta_{l+1}((W_{l+1}^{(k)})^T W_{l+1}^{(k)} a_l^{(k)} - (W_{l+1}^{(k)})^T z_{l+1}^{(k)}))
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
W_L^{(k+1)} &= W_L^{(k)} - \kappa(\beta_L(W_L^{(k)} a_{L-1}^{(k)}(a_{L-1}^{(k)})^T - z_L^{(k)}(a_{L-1}^{(k)})^T)) \\
v^{(k+1)} &= v^{(k)} - \beta_L(z_L^{(k)} - W_L^{(k)} a_{L-1}^{(k)}) \\
z_L^{(k+1)} &: p^{(k+1)} + \frac{1}{\kappa} z_L^{(k+1)} = v^{(k+1)}
\end{aligned} \tag{10}
$$

Until converged

Numerical Details

It's clear that in all four algorithms above, one common nontrivial step is to solve the following PDE:

$$\text{find } z \text{ s.t. } p + \frac{1}{\kappa} z = \frac{1}{\kappa} v \tag{11}$$

where κ, v are given, $p \in \partial L(z)$.
This problem 11 can be formulated into a first order optimal condition of a strongly convex problem, so my notes *Multi-class SVM Loss Subproblem* for numerical implementation details.

Numerical Results on Split Bregman and Linearized Split Bregman for Binary Classification

The iteration includes two parts:

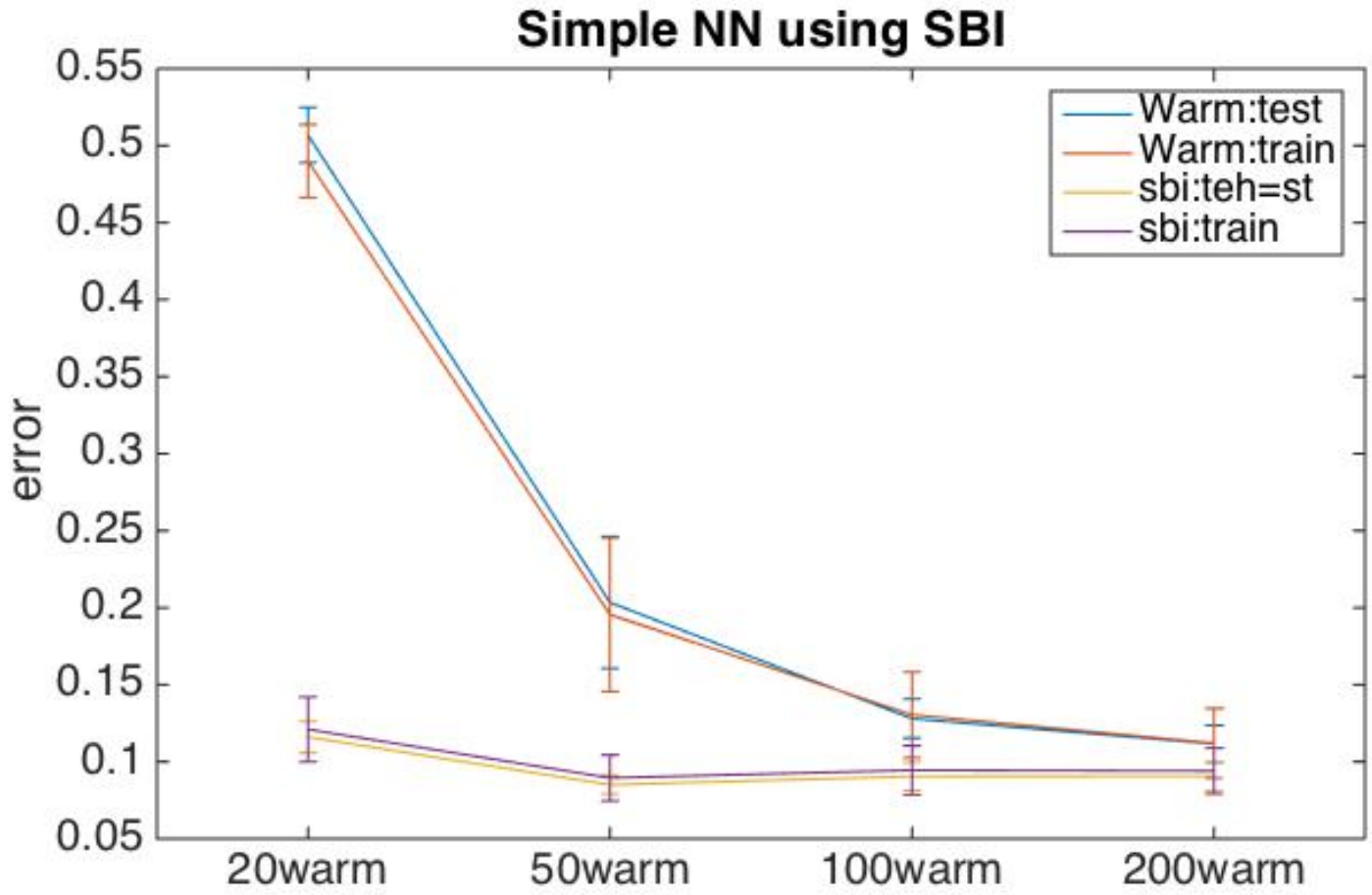
- Warm start: coordinately minimize $\ell_1(z_L, Y) + \ell_2(W) + F(z_L, X)$, that is, update coordinates without subgradient p updating.
- Split Bregman Iteration or Linearized Split bregman Iteration.

Experiment One: Simple data binary classification I

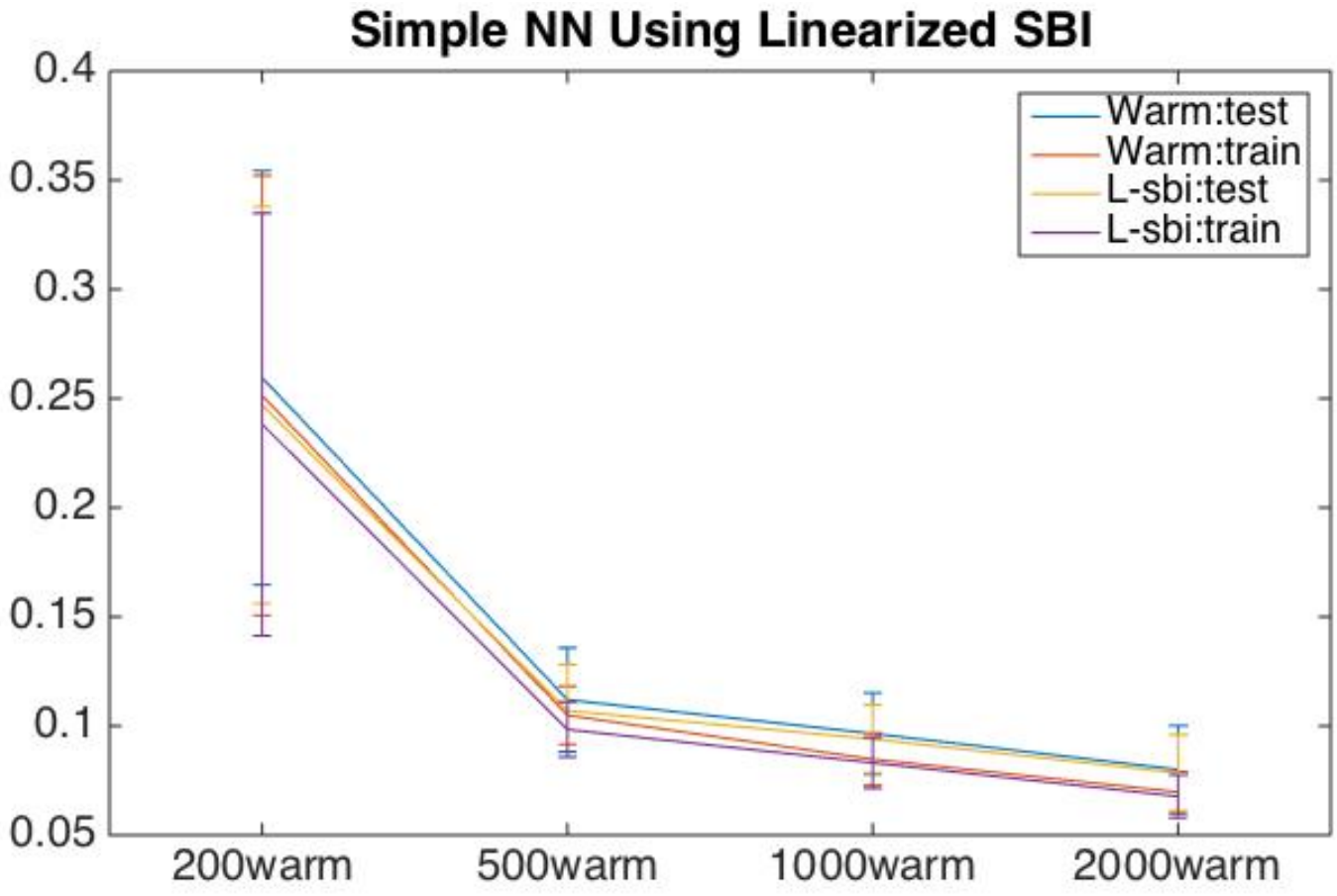
Figure: Figures from up to down are respectively results from split bregman(SBI) and linearized split bregman(Linearized SBI). In each

Experiment One: Simple data binary classification II

figure, errorbars of training data and testing data in both warm start period and split bregman iteration(or linearized) period are shown.



Experiment One: Simple data binary classification III



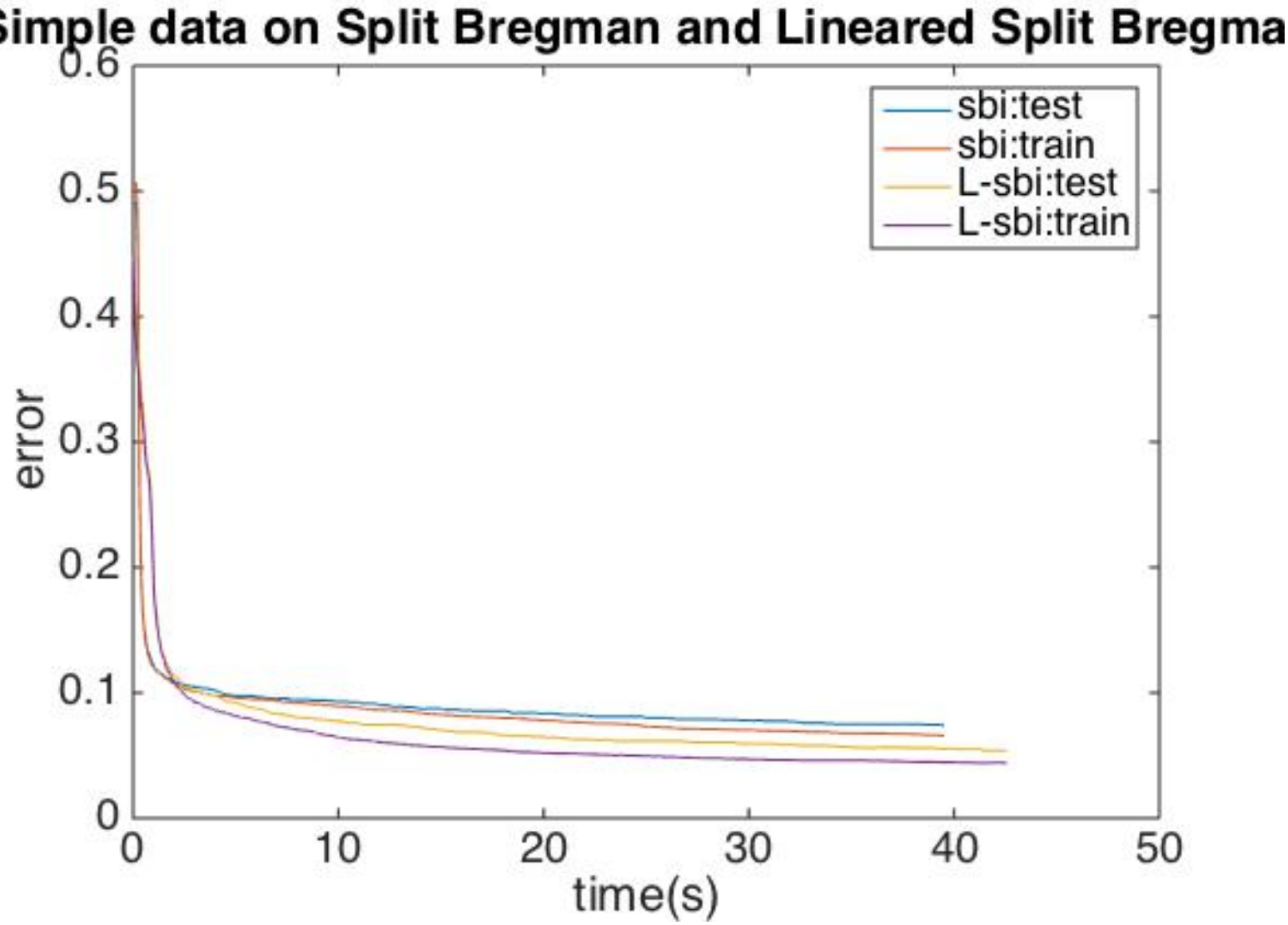
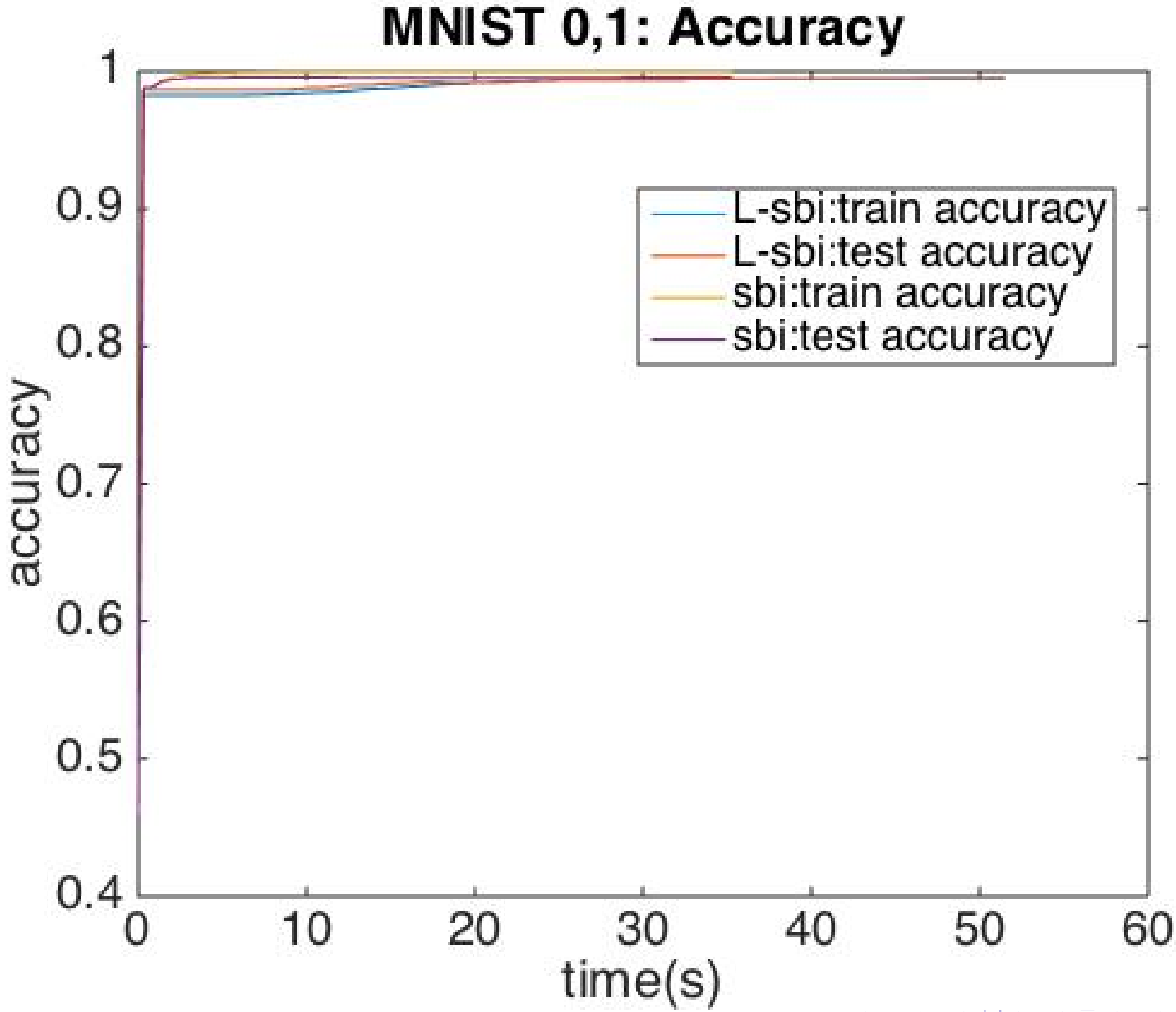


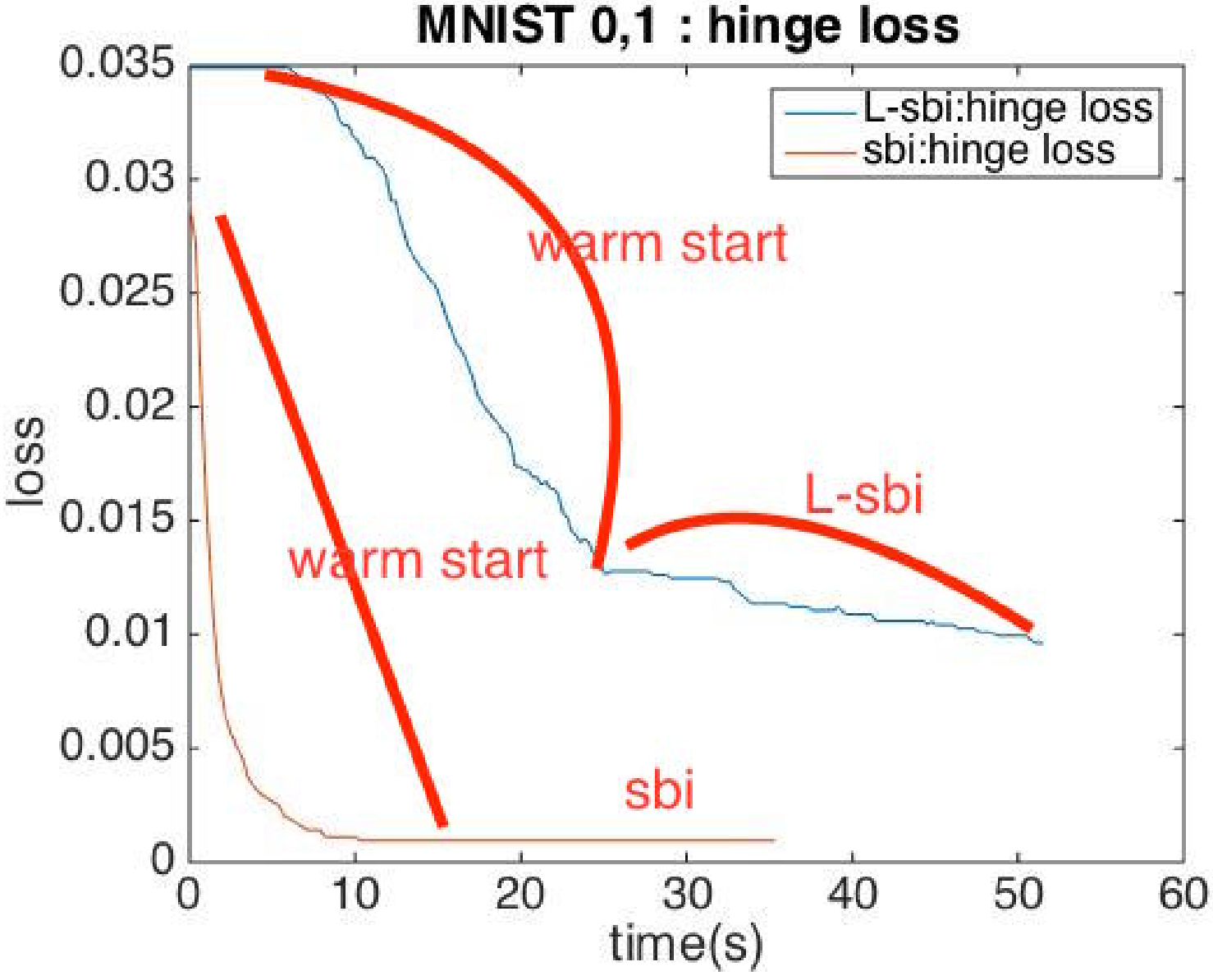
Figure: Error with respect to time in seconds for split bregman and linearized split bregman

Experiment Two: Mnist data binary classification between 0,1 I

Test two algorithms on MNIST sub-datasets, differentiating 0s from 1s. Use hinge loss(SVM classification). Use Nesterov Acceleration for Linearized split bregman.

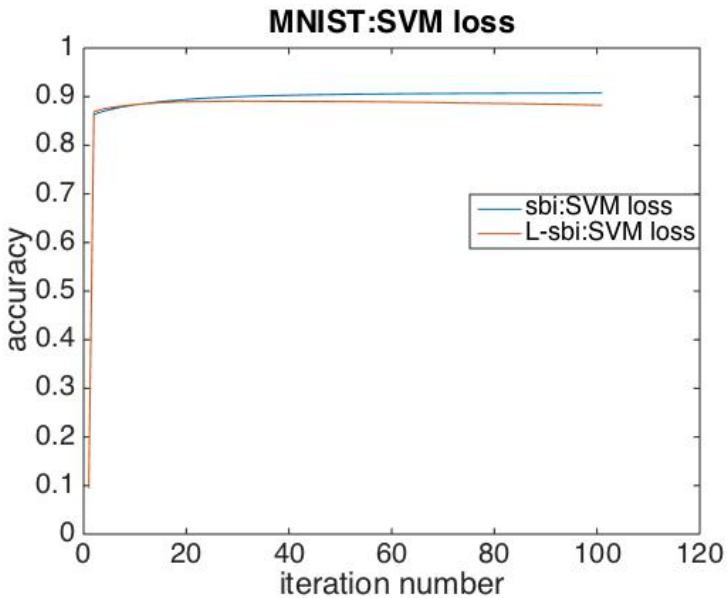
Experiment Two: Mnist data binary classification between 0,1 II

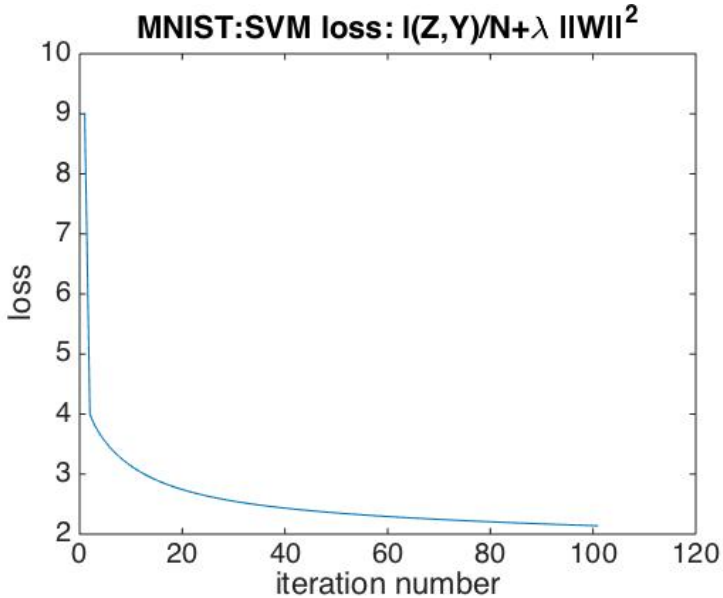



MNIST 0,1 : hinge loss
L-sbi:hinge loss
sbi:hinge loss
warm start
warm start
L-sbi
sbi
loss
time(s)
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
10
20
30
40
50
60

Experiment Three: Mnist data classification–Ten classes

Results of testing SBI on MNIST dataset based on 8 experiments average.





Conclusions

- Tests on simple data shows that L-SBI and SBI have comparable error reducing rate. SBI reduces error faster at the very beginning, but soon L-SBI catches up with SBI and finally has less error than SBI;
- Tests on binary MNIST sub-dataset shows that SBI has a notably faster error reducing rate and is easier to be implemented due to no parameter as learning rate. According to the algorithm, SBI can ensure the relaxed loss function decreasing at each iteration;
- For tests on MNIST, I only shows the results from SBI, since the parameters for L-SBI are remained to be adjusted, and results by now is not acceptable. However, the accuracy from SBI is still not comparable with BP tuning.

Open Questions I

- By now, we can clearly see the computational efficiency for L-SBI due to no inverse calculation, but we may lose the rate of error reducing, and the algorithm can be unstable or too slow for error decaying due to inappropriate choice of parameters. These problems may hopefully be solved by finding a better way of choosing parameters. We'll trying to figure it out.
- We still have no idea whether it is worthwhile to replace SBI with L-SBI, since both these algorithms' computation is comparable except the matrix inverse calculation, while the matrix size is the number of hidden nodes. Also, since SBI has already been a relaxed version of solving original loss function, even if it solves the subproblems of relaxed version accurately, it's still at little risk of overfitting. However, the L-SBI solves each subproblems approximately, which may cause error reducing slower.

Open Questions II

- Now, we have figured out one way to solve the multi-class classification problem for SBI and L-SBI. This method gets an approximated solution, and the results till now show that that approximation can be rather accurate. But it's still remains to determine its effectiveness, and an exact solution is always what we want, although we have no idea whether it exists or not.
- Tests on MNIST dataset shows that SBI can not beat BP. Other datasets should be tested on, especially those on which BP does not work well.