Split Bregman and Linearized Split Bregman for Tuning Neural Network

Ruohan Zhan, Peking University

October 14, 2016

Preliminaries and Reviews

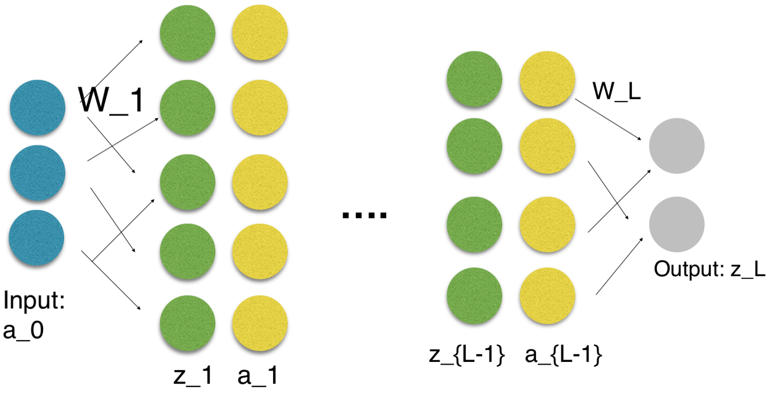
Model and Algorithms

Numerical Results

Open Questions

Architecture of Feedforward Neural Network

Figure: Feedforward Neural Network



For each layer l, input: z_l, output: $a_l = h(z_l)$,
weight W_l and bias b_l: $z_l = W_l a_{l-1} + b_l$.

Backpropagation(BP)

- to use chain rule to calculate all outputs, intermediates and inputs' gradients of loss function backforward.
- loss function: ℓ

$$\frac{\partial \ell}{\partial a_l(i)} = \sum_j \frac{\partial \ell}{\partial z_{l+1}(j)} \frac{\partial z_{l+1}(j)}{\partial a_l(i)}$$

$$= \sum_j \frac{\partial \ell}{\partial z_{l+1}(j)} W_{l+1}(j, i)$$

Problems of BP

- computational burden;
- saturation effects: fails at gradient near 0.

Vanishing Moments:

$$\frac{\partial \ell}{\partial a_1} = \partial h(z_1) W_2 \partial h(z_2) W_3 \frac{\partial \ell}{\partial z_3} \tag{1}$$

Zero Gradient often *can not* be good

- nonconvex
- random initialization

Optimization Perspective

Variable Splitting: denote
$X = (a_1, ..., a_{L-1}, z_1, ..., z_L, W_1, ..., W_L, b_1, ..., b_L)$

$$
\begin{aligned}
\text{minimize}_X \quad & \ell(z_L, Y) + \sum_l \mathcal{R}(W_l) \\
\text{subject to} \quad & \begin{aligned} z_l &= W_l a_{l-1} + b_l, \\ a_l &= h_l(z_l). \end{aligned} \quad l = 1, \ldots, L
\end{aligned} \tag{2}
$$

Normally, the regulation term: $\mathcal{R}(W_l) = \frac{\lambda}{2}\|W_l\|^2$.

Differences from Goldstein's paper[1]

- We try to extend it to multi-class cases, while [1] only with binary cases;
- Our loss function: data fidelity and regulation; Theirs: data fidelity
- Our loss function adds bias terms.
- We give more insights into the potential of SBI not only on the scalable capability but also the ability to avoid saturation.
- We extend SBI into the Linearized SBI.

Bregman Formulation

$$
\begin{aligned}
X^{(k+1)} = \operatorname{argmin}_X \quad & \ell(z_L, Y) - \langle z_L - z_L^{(k)}, p^{(k)} \rangle \\
& + \sum_l \left[\mathcal{R}(W_l) - \langle W_l - W_l^{(k)}, q_l^{(k)} \rangle \right] \\
& + \sum_l \left[\frac{\gamma}{2} \| a_l - h_l(z_l) \|^2 + \frac{\beta}{2} \| z_l - W_l a_{l-1} \|^2 \right]
\end{aligned}
$$

where $p^{(k)} \in \partial \ell(z_L^{(k)}, Y)$ and $q_l^{(k)} \in \partial \mathcal{R}(W_l^{(k)})$

Another Interpretation, **Augmented Lagrangian**(similar to formulation in [3])

$$
\begin{aligned}
X^{(k+1)} = \operatorname{argmin}_X \quad & \ell(z_L, Y) + \sum_l \mathcal{R}(W_l) \\
& + \sum_l \left[\frac{\gamma}{2} \| a_l - h_l(z_l) + \xi_l^{(k)} \|^2 + \frac{\beta}{2} \| z_l - W_l a_{l-1} + \eta_l^{(k)} \|^2 \right]
\end{aligned}
$$

Split Bregman Iteration(SBI)

$$
\begin{aligned}
z_L^{(k+1)} &= \mathrm{argmin}_{z_L} \ell_1(z_L, Y) - \langle p^{(k)}, z_L - z_L^{(k)} \rangle + \frac{\beta}{2} \| W_L^{(k)} a_{L-1}^{(k)} + b_l^{(k)} - z_L \|^2 \\
p^{(k+1)} &= p^{(k)} + \beta (W_L^{(k)} a_{L-1}^{(k)} - z_L^{(k+1)}) \\
b_L^{(k+1)} &= \mathrm{argmin}_{b_L} \frac{\beta}{2} \| W_L^{(k)} a_{L-1}^{(k)} + b_L - z_L^{(k+1)} \|^2 \\
W_L^{(k+1)} &= \mathrm{argmin}_{W_L} \mathcal{R}(W_L) - \langle W_L - W_L^{(k)}, q_L^{(k)} \rangle + \frac{\beta}{2} \| W_L a_{L-1}^{(k)} + b_L^{(k+1)} - z_L^{(k+1)} \|^2 \\
q_L^{(k+1)} &= q_L^{(k)} + \beta (z_L^{(k+1)} - (W_L^{(k+1)} a_{L-1}^{(k)} + b_L^{(k+1)}))(a_{L-1}^{(k)})^T
\end{aligned}
$$

For $l = L-1, \ldots, 2, 1,$ **updating order matters**

$$
\begin{cases}
a_l^{(k+1)} = \mathrm{argmin}_{a_l} \frac{\gamma}{2} \| a_l - h(z_l^{(k)}) \|^2 + \frac{\beta}{2} \| W_{l+1}^{(k+1)} a_l + b_{l+1}^{(k+1)} - z_{l+1}^{(k+1)} \|^2 \\
z_l^{(k+1)} = \mathrm{argmin}_{z_l} \frac{\gamma}{2} \| a_l^{(k+1)} - h(z_l) \|^2 + \frac{\beta}{2} \| W_l^{(k)} a_{l-1}^{(k)} + b_l^{(k)} - z_l \|^2 \\
b_l^{(k+1)} = \mathrm{argmin}_{b_l} \frac{\beta}{2} \| W_l^{(k)} a_{l-1}^{(k)} + b_l - z_l^{(k+1)} \|^2 \\
W_l^{(k+1)} = \mathrm{argmin}_{W_l} \mathcal{R}(W_l) - \langle W_l - W_l^{(k)}, q_l^{(k)} \rangle + \frac{\beta}{2} \| W_l a_{l-1}^{(k)} + b_l^{(k+1)} - z_l^{(k+1)} \|^2 \\
q_l^{(k+1)} = q_l^{(k)} + \beta (z_l^{(k+1)} - (W_l^{(k+1)} a_{l-1}^{(k)} + b_l^{(k+1)}))(a_{l-1}^{(k)})^T
\end{cases}
\tag{3}
$$

Updating Order

Numerical experiments show that, for multi-class cases, backward updating converges faster than forward updating. Therefore, in our later experiments, we use backward updating.

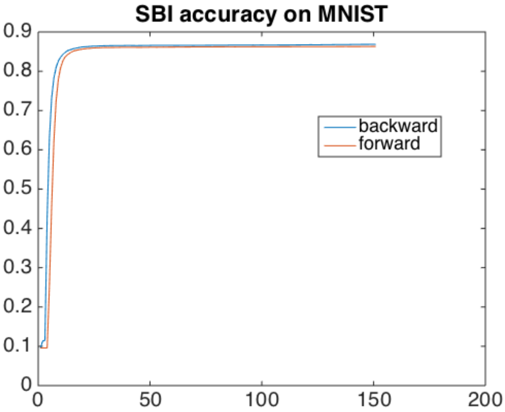


Figure: Results shown on a 5-test average

Split Bregman Algorithm

Algorithm 1: Split Bregman for Neural Nets

Input: training feature $\{a_0\}$, and labels $\{y\}$,
Initialize: initialize a_l, z_l with Gaussian distribution, set $b_l^{(0)} = 0$ and solve out the W_l explicitly.
repeat

$$\begin{aligned} z_L^{(k+1)} &: \nabla \ell_1(z_L, Y) + \beta z_L = \beta(W_L^{(k)} a_{L-1}^{(k)} + b_L^{(k)}) + p^{(k)} \\ p^{(k+1)} &= p^{(k)} + \beta(W_L^{(k)} a_{L-1}^{(k)} + b_L^{(k)} - z_L^{(k+1)}) \\ b_L^{(k+1)} &= \mathrm{mean}_i(z_L^{(k+1)}(i) - W_L^{(k)} a_{L-1}^{(k)}(i)) \\ W_L^{k+1} &= (\beta(z_L^{(k+1)} - b_L^{(k+1)}) + \lambda W_L^{(k)})(\beta a_{L-1}^{(k)} + \lambda I)^{-1} \end{aligned} \tag{4}$$

for *l=L-1...,2,1* **do**

Inverse Calculation

$$\begin{aligned} a_l^{k+1} &= \boxed{(\gamma I + \beta(W_{l+1}^{(k+1)})^T W_{l+1}^{(k+1)})^{-1}} (\gamma h(z_l^k) + \beta(W_{l+1}^{(k+1)})^T (z_{l+1}^{(k+1)} - b_{l+1}^{(k+1)})) \\ z_l^{(k+1)} &: \gamma(h(z_l^{k+1}) - a_l^{(k+1)})\partial h(z_l) + \beta(z_l - W_l^{(k)} a_{l-1}^{(k)} - b_l^{(k)}) = 0 \\ b_l^{(k+1)} &= \mathrm{mean}_i(z_l^{(k+1)}(i) - W_l^{(k)} a_{l-1}^{(k)}(i)) \\ W_l^{(k+1)} &= (\beta(z_l^{(k+1)} - b_l^{(k+1)}) + \lambda W_l^{(k)})\boxed{(\beta a_{l-1}^{(k)} + \lambda I)^{-1}} \end{aligned} \tag{5}$$

Until converged

Linearized Split Bregman Iteration(L-SBI)

At each updating, linearize the relaxation term with **first Taylor expansion** plus a **quadratic regulation**.

$$
\begin{cases}
SBI : z_L^{(k+1)} = \mathrm{argmin}_{z_L} \ell_1(z_L, Y) - \langle p^{(k)}, z_L - z_L^{(k)} \rangle + \frac{\beta}{2} \| W_L^{(k)} a_{L-1}^{(k)} + b_l^{(k)} - z_L \|^2 \\
L-SBI : z_L^{(k+1)} = \mathrm{argmin}_{z_L} \ell_1(z_L, Y) - \langle p^{(k)}, z_L - z_L^{(k)} \rangle \\
\qquad + \beta \langle z_L - z_L^{(k)}, z_L^{(k)} - W_L^{(k)} a_{L-1}^{(k)} \rangle + \frac{\beta}{2} \| z_L^{(k)} - W_L^{(k)} a_{L-1}^{(k)} - b_L^{(k)} \|^2 \\
\qquad + \frac{1}{2\kappa} \| z_L - z_L^{(k)} \|^2
\end{cases}
$$

The same idea for updating other variables.

No Inverse Calculation!!

A More Reasonable Linearization

The linearization motivates from dropping inverse calculation terms and thus speeding up. Note that in SBI, we only need to calculate inverse for updating a_l, W_l, thus we can only linearize these two. Take a_l as an example:

$$
\begin{aligned}
a_l^{(k+1)} = \operatorname{argmin}_{a_l} \langle \gamma(a_l - h(z_l^{(k)})) + \beta(W_{l+1}^{(k+1)})^T(W_{l+1}^{(k+1)} a_l^{(k)} \\
+ b_{l+1}^{(k+1)} - z_{l+1}^{(k+1)}), a_l - a_l^{(k)} \rangle + \frac{1}{2\kappa} \|a_l - a_l^{(k)}\|^2
\end{aligned}
\tag{6}
$$

Warm Start

Any algorithm needs a good initialization...

In the warm start period, we do not update the subgradient, or in other words, the dual Lagrangian, **which allows the algorithm to search on a broader field not only on the manifold with equality constraints**.

Warm Start for Split Bregman Algorithm

Algorithm 2: Warm Start for Split Bregman

Input: training feature $\{a_0\}$, and labels $\{y\}$,

Initialize: initialize a_l, z_l with Gaussian distribution, set $b_l^{(0)} = 0$ and solve out the W_l explicitly.

repeat

$$
\begin{aligned}
z_L^{(k+1)} &: \nabla \ell_1(z_L, Y) + \beta z_L = \beta(W_L^{(k)} a_{L-1}^{(k)} + b_L^{(k)}) \\
b_L^{(k+1)} &= \text{mean}_i(z_L^{(k+1)}(i) - W_L^{(k)} a_{L-1}^{(k)}(i)) \\
W_L^{k+1} &= \beta(z_L^{(k+1)} - b_L^{(k+1)})(\beta a_{L-1}^{(k)} + \lambda I)^{-1}
\end{aligned} \tag{7}
$$

for *l=L-1,...,2,1* **do**

$$
\begin{aligned}
a_l^{k+1} &= (\gamma I + \beta (W_{l+1}^{(k+1)})^T W_{l+1}^{(k+1)})^{-1}(\gamma h(z_l^k) + \beta (W_{l+1}^{(k+1)})^T (z_{l+1}^{(k+1)} - b_{l+1}^{(k+1)})) \\
z_l^{(k+1)} &: \gamma(h(z_l^{k+1}) - a_l^{(k+1)}) \nabla h(z_l) + \beta(z_l - W_l^{(k)} a_{l-1}^{(k)} - b_l^{(k)}) = 0 \\
b_l^{(k+1)} &= \text{mean}_i(z_l^{(k+1)}(i) - W_l^{(k)} a_{l-1}^{(k)}(i)) \\
W_l^{(k+1)} &= \beta(z_l^{(k+1)} - b_l^{(k+1)})(\beta a_{l-1}^{(k)} + \lambda I)^{-1}
\end{aligned} \tag{8}
$$

Until converged

Warm Start for Linearized Split Bregman Iteration

Just Omit the subgradient terms.

At each updating, linearize the relaxation term with **first Taylor expansion** plus a **quadratic regulation**.

$$
\begin{aligned}
z_L^{(k+1)} =& \operatorname{argmin}_{z_L} \ell_1(z_L, Y) + \beta \langle z_L - z_L^{(k)}, z_L^{(k)} - W_L^{(k)} a_{L-1}^{(k)} \rangle \\
&+ \frac{\beta}{2} \| z_L - W_L^{(k)} a_{L-1}^{(k)} - b_L^{(k)} \|^2 + \frac{1}{2\kappa} \| z_L - z_L^{(k)} \|^2
\end{aligned}
$$

The same idea for updating other variables.

No Inverse Calculation!!

Numerical Results

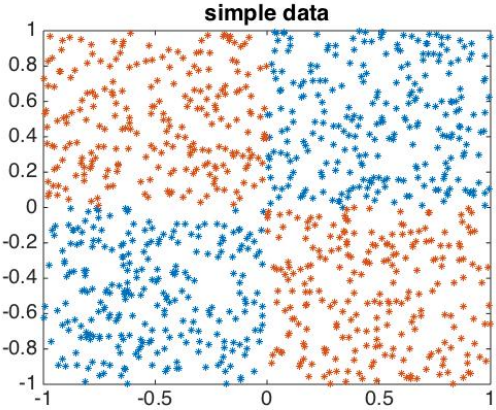
$X = (a_1, ..., a_{L-1}, z_1, ..., z_L, W_1, ..., W_L, b_1, ..., b_L)$.
Denote $F(X) = \sum_i \frac{\beta}{2}\|W_l a_{l-1} + b_l - z_l\|^2 + \frac{\gamma}{2}\|a_l - h(z_l)\|^2$.

The learning includes two parts:

- Warm start: coordinately minimize $\ell_1(z_L, Y) + \sum_l \mathcal{R}(W_l) + F(X)$.
- Bregman Iteration: Split Bregman or Linearized Split Bregman

Experiment One: Simple data binary classification I

Training set: 800; Testing set: 200.
$y \in \{1, -1\}$: *target*, z : *output*.



All results are shown based on an average of 10 experiments.

Experiment One: Simple data binary classification II

Run warm start for different times, then run SBI for 1000 times.

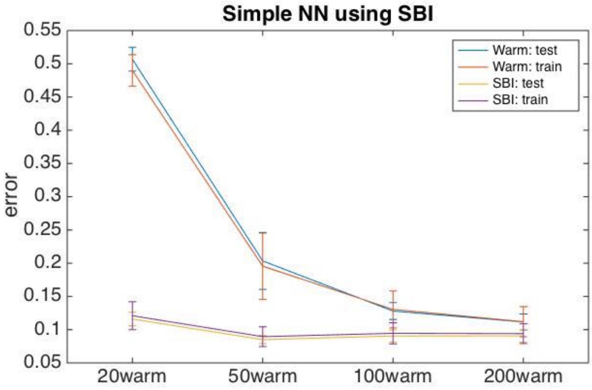
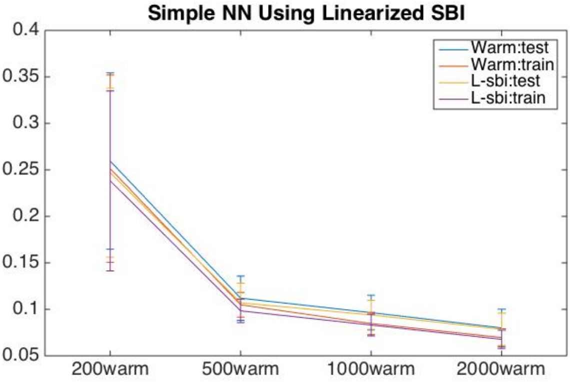


Figure: Split Bregman Iteration: errorbars of training data and testing data in both warm start period and SBI period.

Experiment One: Simple data binary classification III

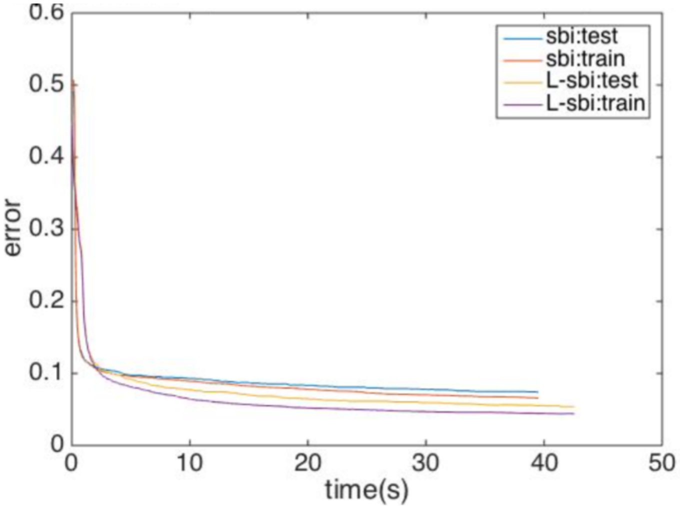
Run warm start for different times, then run L-SBI for 1000 times.

Figure: Linearized Split Bregman Iteration: errorbars of training data and testing data in both warm start period and L-SBI period.



Experiment One: Simple data binary classification IV

Figure: A comparison of Error with respect to time in seconds for SBI and L-SBI



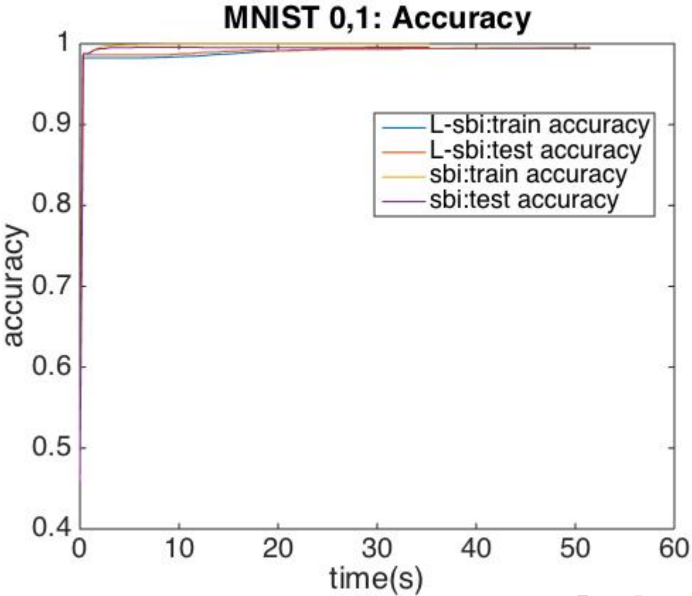
Experiment Two: Mnist data binary classification between 0,1 I

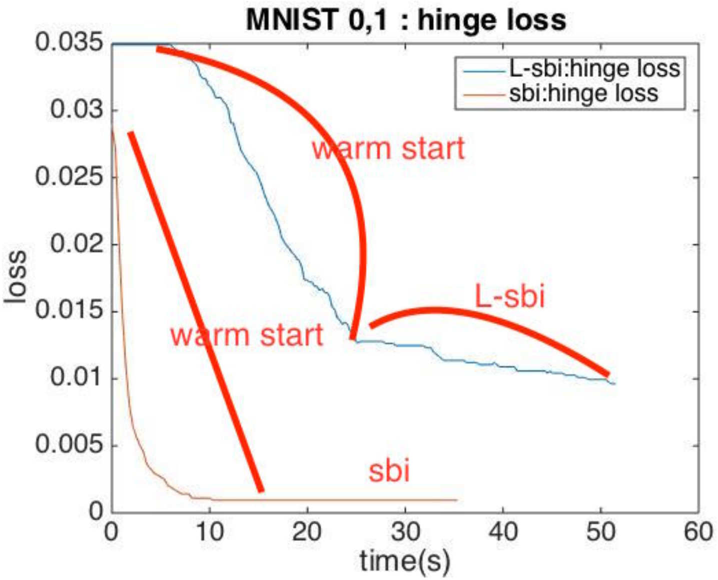
Test two algorithms on MNIST sub-datasets(0, 1).
Use hinge loss(SVM classification).
Use Nesterov Acceleration for Linearized split bregman.

Nesterov Acceleration:

$$\begin{cases} y_k = x_{k-1} - \nu \nabla f(x_{k-1}) \\ x_k = y_k + \dfrac{k-1}{k+2}(y_k - y_{k-1}). \end{cases} \tag{9}$$

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



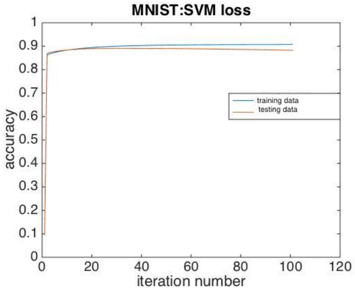


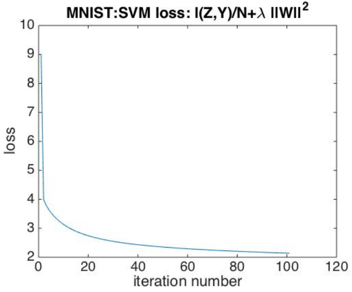
MNIST 0,1 : hinge loss
L-sbi:hinge loss
sbi:hinge loss
warm start
warm start
L-sbi
sbi
loss
time(s)
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
10
20
30
40
50
60

Experiment Three: Mnist data classification–Ten classes I

$784 \to 30 \to 10$

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





Experiment Four: Saturation? I

$784 \to 30 \to 10$

One hidden layer: even if the output changes little, the hidden nodes change greatly: potential for a jump in accuracy.

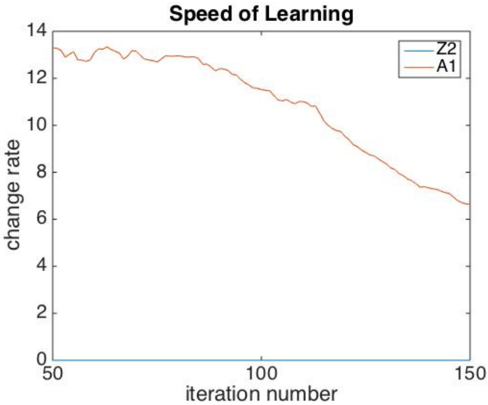


Figure: SBI on MNIST: One hidden layer, speed of learning

Experiment Four: Saturation? II

$784 \to 30 \to 30 \to 10$

Two hidden layers: hidden layers change rate are comparable

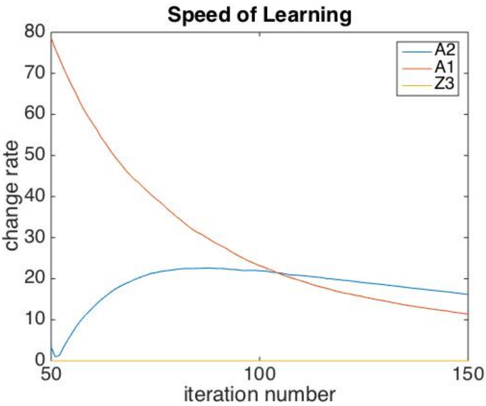


Figure: SBI on MNIST: Two hidden layers, speed of learning

Experiment Four: Saturation? III

$784 \to 30 \to 30 \to 10$

Let's look at the learning speed of BP[2]: early hidden layers learn much more slowly than later hidden layers

y-coordinate has been taken logarithm on.

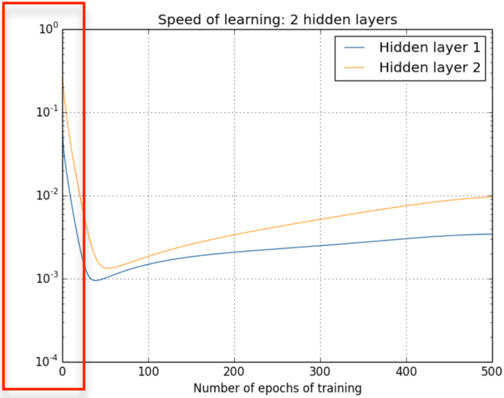


Figure: BP on MNIST[2]: Two hidden layers, speed of learning

Technical issue: more discussion on loss function I

In split bregman iteration, one important step is to solve the following subproblem:

$$
\begin{aligned}
&solve \quad z, \\
&s.t. \quad p + \frac{1}{\kappa} z = v, \quad p \in \partial \ell(z).
\end{aligned}
$$

Three common data fidelity terms in loss function:
square loss, hinge loss and cross entropy loss.

Technical issue: more discussion on loss function II

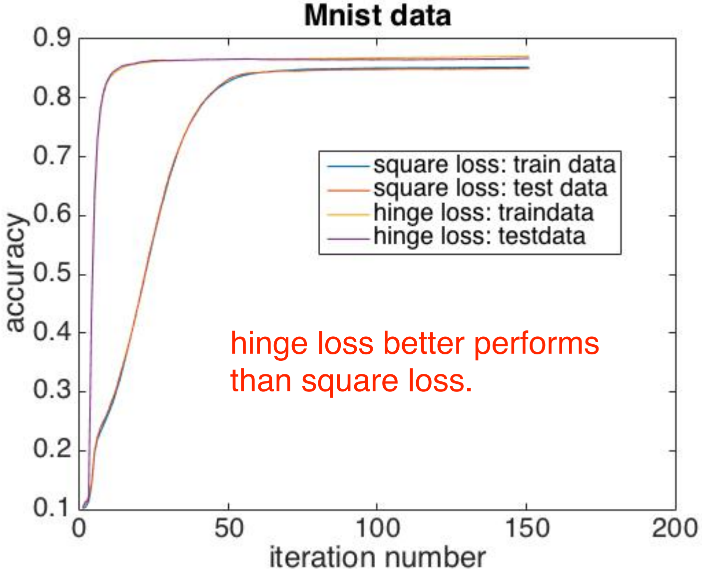


Figure: A comparison of square loss and hinge loss.

Open Questions I

- L-SBI: computational efficiency, but hard for parameter choosing.
- the potential of L-SBI to replace SBI?
- Saturation: DNN? RNN?

Bibliography

[1]Taylor G, Burmeister R, Xu Z, et al. Training Neural Networks Without Gradients: A Scalable ADMM Approach[J]. arXiv preprint arXiv:1605.02026, 2016.
[2]Deep Learning, draft book in preparation, Yoshua Bengio, Ian Goodfellow, and Aaron Courville, 2016.01
[3]Zhang Z, Chen Y, Saligrama V. Efficient Training of Very Deep Neural Networks for Supervised Hashing[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 1487-1495.

Numerical Details for SVM Loss I

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

$$\text{find } z \text{ s.t. } p + \frac{1}{\kappa} z = \frac{1}{\kappa} v \tag{10}$$

where κ, v are given, $p \in \partial L(z)$.

Numerical Details for SVM Loss II

This problem 10 can be formulated into a first order optimal condition of a strongly convex problem:

$$fix \quad i, \quad \min_{z \in \mathbf{R}^n} \frac{1}{2\kappa}\|z-v\|^2 + \sum_{j \neq i} \max(0, 1 - z_i + z_j), \tag{11}$$

which is equivalent to the following problem with constraints:

$$\min_{z, \xi \in \mathbf{R}^n} \frac{1}{2\kappa}\|z-v\|^2 + \mathbf{1}^T \xi, \quad s.t. \xi \succeq 0, \xi \succeq Az + b, \tag{12}$$

where $A = I - \mathbf{1}e_i^T$ and $b = \mathbf{1} - e_i = -Ae_i$.

Numerical Details for SVM Loss III

The Lagrangian is as follows:

$$L(z,\xi,\lambda,\mu) = \frac{1}{2\kappa}\|z-v\|^2+\mathbf{1}^T\xi-\mu^T\xi+\lambda^T(Az+b-\xi), \quad \mu \succeq 0, \lambda \succeq 0. \tag{13}$$

Therefore, the dual problem of problem 12 is

$$\min_{\lambda\in\mathbf{R}^n} \quad \frac{\kappa}{2}\lambda^T AA^T\lambda - \lambda^T(b+Av), \text{ s.t. } 1 \succeq \lambda, \lambda \succeq 0. \tag{14}$$

which can be roughly solved by penalty method.