Study of
Line Search of Learning Rate in RBM

Xing Zhang

SUNY, UAlbany

Motivation

- Many RBM learning algorithms:
  - focus on the gradient update
  - lack of attention on the learning rate update
- Our work:
  - try to pick well-grounded values for learning rate, therefore speed up RBM learning

Restricted Boltzmann Machine

- Hidden layer
- Visible layer

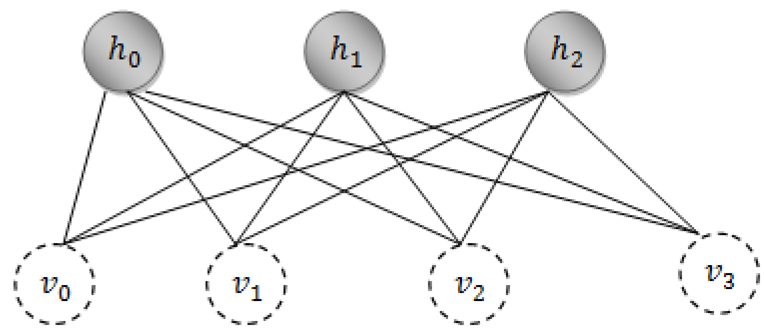

Fig 1: The structure of RBM

- Assume no bias
- Energy function: $E(V,H) = -\sum_{i,j} v_i h_j w_{ij}$
- Probability: $P(V,H) = \frac{exp(\frac{1}{2}\sum_{i,j} v_i h_j w_{ij})}{Z(W)}$
- Activity rule: $P(h_i = 1|V) = sigmoid(W_i^T V)$

  $P(v_j = 1|H) = sigmoid(W_j H)$

RBM learning

- A set of data $X^{(1)}, \dots, X^{(N)}$
- Target density $P^{0}(X)$
- parametric model density $P^{\infty}(X, W)$
- Goal: find weights W so that $P^{\infty}(X, W)$ is close to $P^{0}(X)$
- Solution: gradient descent/ascent algorithm

$$W^{(k+1)} = W^{(k)} + \eta^{(k)} \Delta W^{(k)}$$

Maximum Likelihood

- Objective function

$$L(W) = \frac{1}{N}\sum_{n=1}^{N} log\, P(X^{(n)}|W)$$

$$W_{ML} = argmax\, L(W) = argmin\, KL(P^0||P^\infty)$$

- Gradient of $L(W)$ w.r.t. W (optimal direction):

$$\frac{\partial L}{\partial W_{st}} = < v_s h_t >_{P^0} - < v_s h_t >_{P^\infty}$$

positive phase   negative phase

- MCMC (Gibbs sampling) → bottleneck

Contrastive Divergence

- Gradient (right direction):

$$\frac{\partial CD_n}{\partial W_{st}} = < v_s h_t >_{p^0} - < v_s h_t >_{p^n}$$

where p^n is the distribution which starts at p^0 and run Markov Chain for n steps.

- n=1 works well

Contrastive Divergence

- Toy example

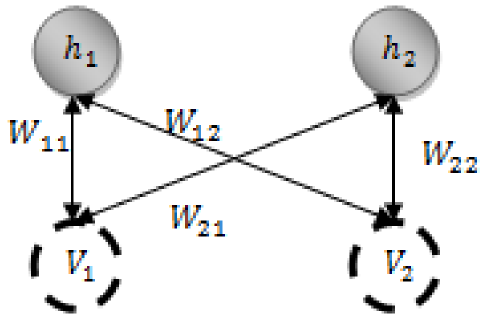

Fig. 2 The structure of RBM in the example

$$W_{11} = W_{12} = W_{21} = W_{22} = 0.5$$

$$SampleNo = 1000$$

During each loop, Gibbs sampler runs 1000 steps for ML and only one step for CD.

Log likelihood is chosen as the measurement.

Contrastive Divergence

- The performace of ML and CD

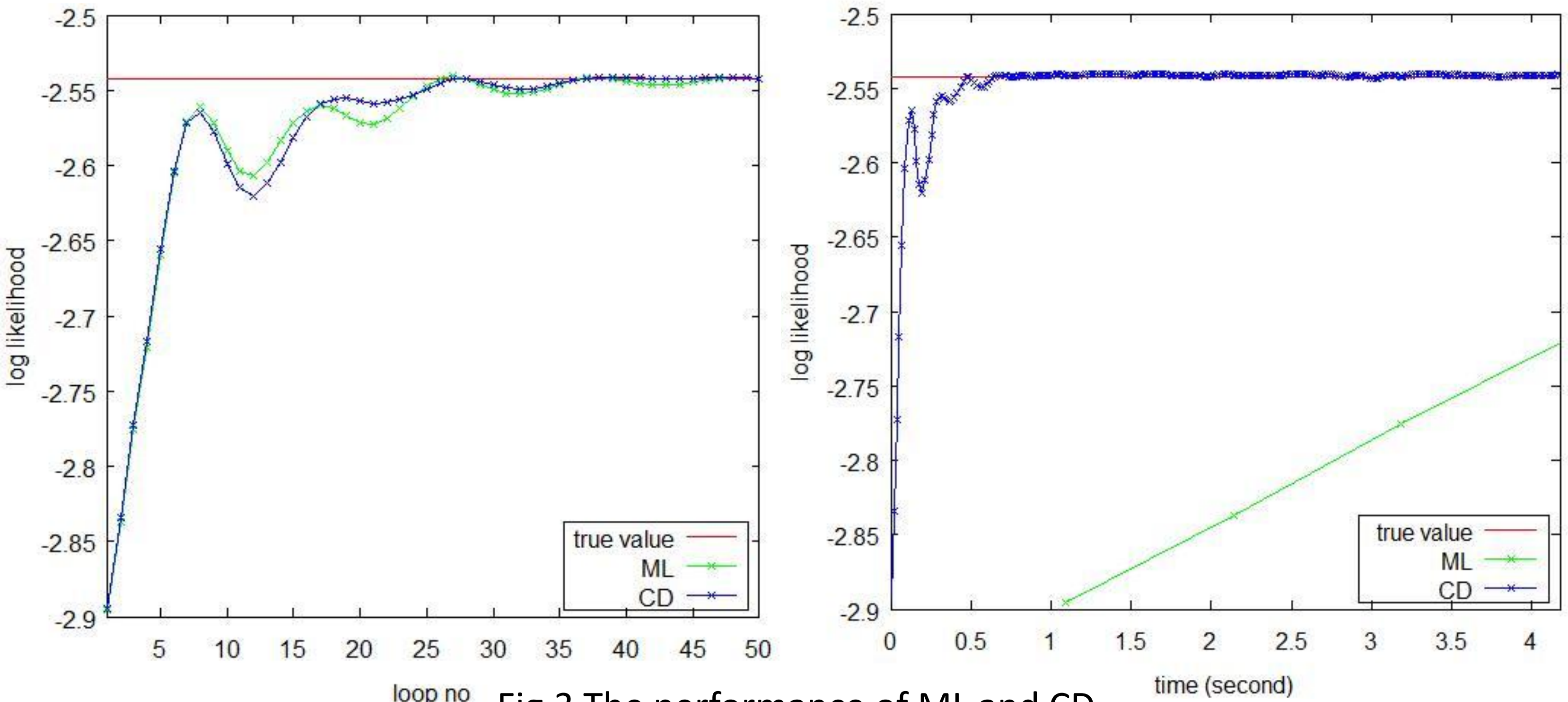

Fig.3 The performance of ML and CD

1-D line search of learning rate

- $$W^{(k+1)} = W^{(k)} + \eta^{(k)} \Delta W^{(k)}$$

  optimal/right step size? ($\eta^{(k)}$) optimal/right direction ($\Delta W^{(k)}$)

- How to determine the learning rate?
  - Decrease the learning rate with the increase of step number
  - Experience
  - Experiments

  Why not pick a well-gounded value for learning rate?

1-D line search of learning rate

- Fact: the optimal learning rate η is also required to maximize/minimize the objective function L, i.e.,

$$\eta^{(k)*} = argmax\ L(W^{(k)} + \eta^{(k)} \Delta W^{(k)})$$

- Idea: during each step, after updating the gradient, append a line search of learning rate which gives an value close to the optimal one.

1-D line search of learning rate

- ML with 1-D line search of learning rate
  - $\eta = argmax\ L(W + \eta \Delta W)$

$$\frac{\partial L(W + \eta \Delta W)}{\partial \eta} = \sum < v_s h_t \Delta W_{st} >_{P^0} - \sum < v_s h_t \Delta W_{st} >_{P^\infty}$$

Gibbs sampling

  - use CD instead
  - the gradient of $CD_1(W + \eta \Delta W)$ w.r.t. η (right step size):

$$\frac{\partial CD_1(W + \eta \Delta W)}{\partial \eta} = \sum < v_s h_t \Delta W_{st} >_{P^0} - \sum < v_s h_t \Delta W_{st} >_{P^1}$$

1-D line search of learning rate

- ML with 1-D line search of learning rate
  - Algorithm

while (W_Loop<=maxLoopNo)

1) compute $P^{+}_{W_{st}} = < v_s h_t >_{P^0}$ for all (s,t);

2) run Gibbs sampler m steps to get samples;

3) compute $P^{-}_{W_{st}} = < v_s h_t >_{P^\infty}$ for all (s,t);

4) while(eta_Loop<=n)

4.1) compute $P^{+}_{\eta} = \sum < v_s h_t \Delta W_{st} >_{P^0}$;

4.2) run Gibbs sampler one step to get samples;

4.3) compute $P^{-}_{\eta} = \sum < v_s h_t \Delta W_{st} >_{P^1}$;

4.4) update η;

5) update W;

1-D line search of learning rate

- ML with 1-D line search of learning rate

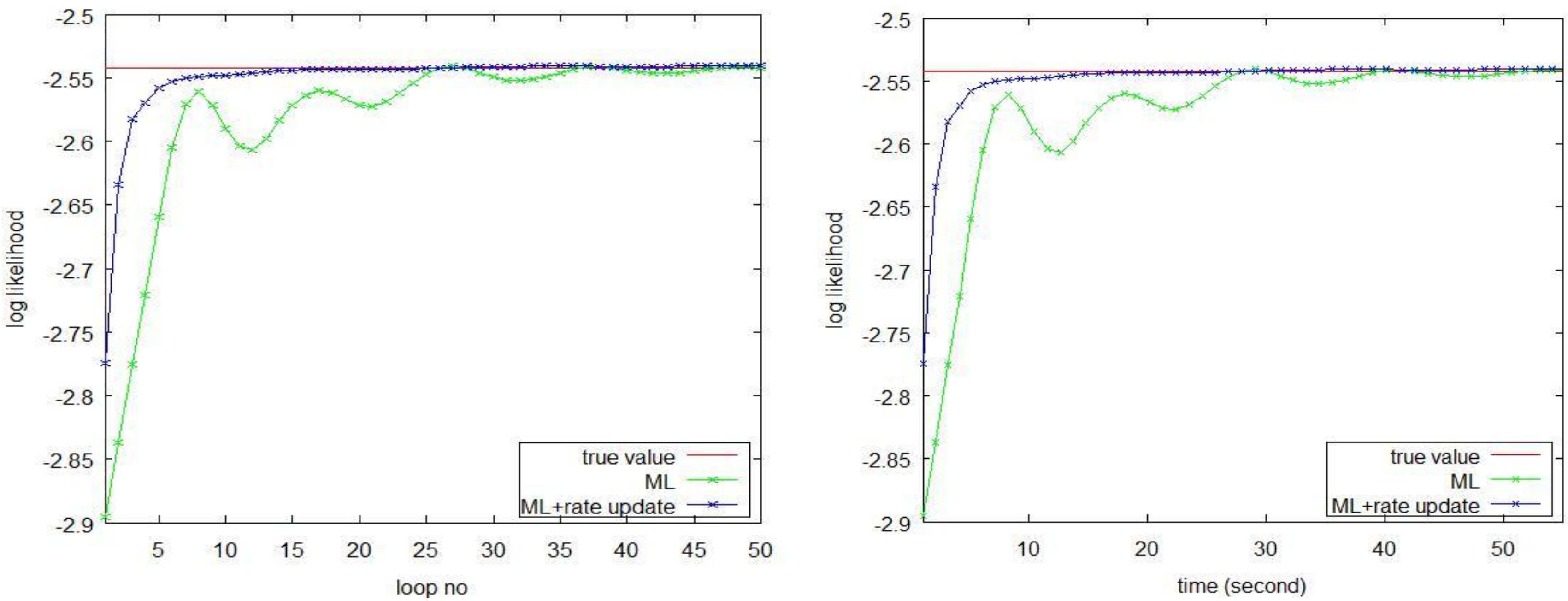

Fig. 4 The performance of original ML and ML with 1-D line search of learning rate (During each outer loop, the original ML runs 1000 steps Gibbs Sampling and the inner one runs 10 loops of 1-step CD. The learning rate of original ML is 0.1.)

1-D line search of learning rate

- ML with 1-D line search of learning rate
  - Why it is more efficient than original ML?

    (1) Most of the running time is consumed by Gibbs Sampling.

    (2) During each outer loop,

    the original ML runs m steps Gibbs Sampling $\longrightarrow O(m)$;

    the inner one runs n loops of 1-step CD $\longrightarrow O(n)$.

    Therefore, $O(T_{new}/T_{orig}) = O(1+n/m)$.

    For ML, $n \ll m \longrightarrow O(1+n/m) \approx O(1)$.

    (3) $loop_{new} < loop_{orig}$.

1-D line search of learning rate

- Negative result: CD with 1-D line search of learning rate

  For 1-step CD, $m = 1$ ➡ $O(1 + n/m) = O(1 + n)$.

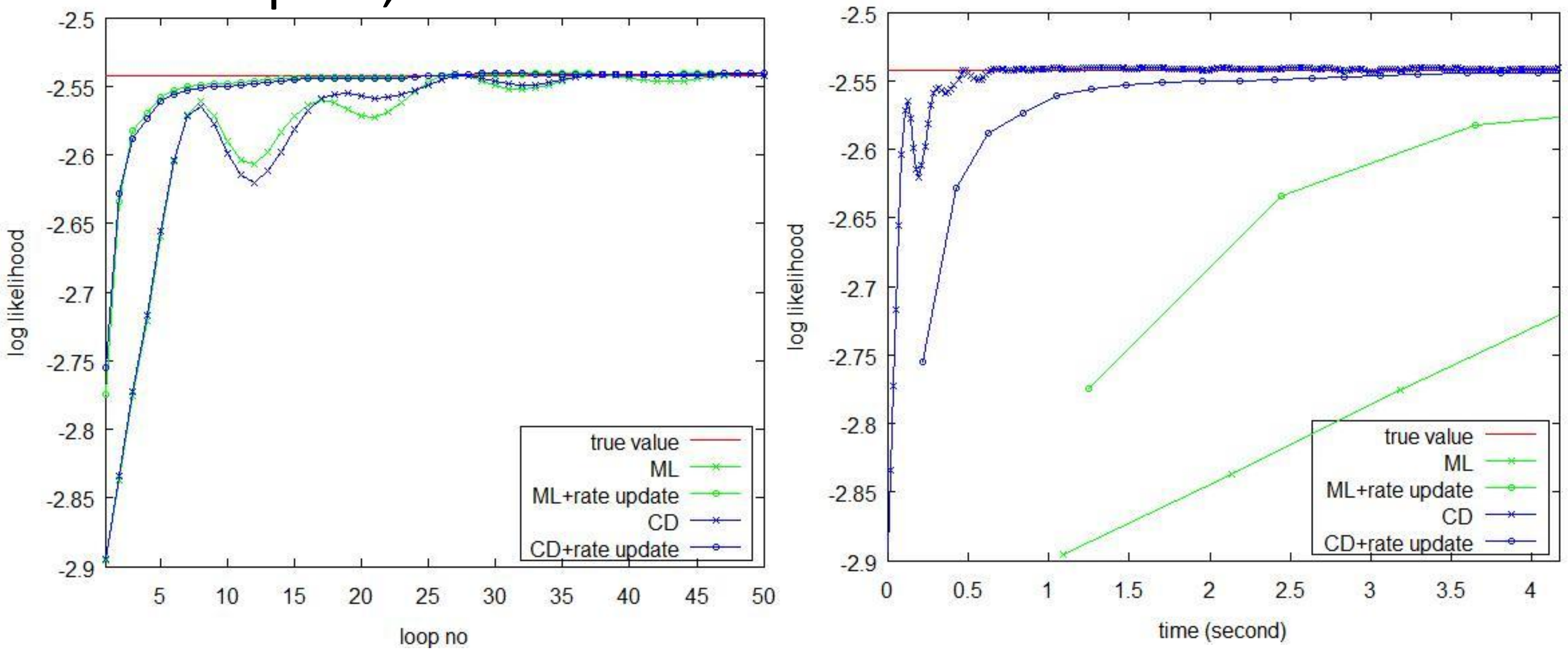

Fig. 5 The performance of original CD and CD 1-D line search of learning rate (During each outer loop, the original CD runs one step Gibbs Sampling and the inner one runs 10 loops of 1-step CD. The learning rate of original ML/CD is 0.1.)

Conclusion

- The efficiency of the gradient algorithm depends on the gradient update and the learning rate update.
- ML gives the optimal direction to update the weight. It guarantees convergence, but runs slowly.
- CD runs fast, and uses a non-optimal gradient update rule.
- To improve its efficiency, ML can be combined with 1-D line search of learning rate which gives a right step size. However, this trick is not worthwhile for CD.

Thank you!