Batch-normalized Deep Boltzmann Machines

Hung Vu[1], Tu Dinh Nguyen[2], Trung Le[2], Wei Luo[1] and Dinh Phung[2]

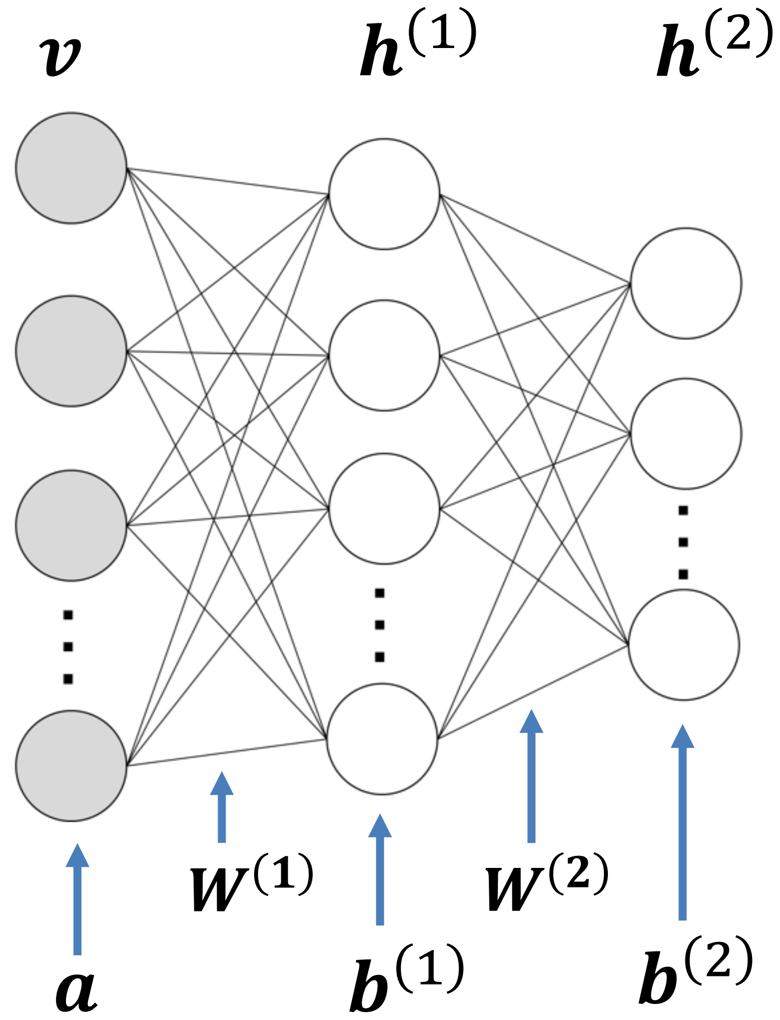
[1]Centre for Pattern Recognition and Data Analytics (PRaDA), Deakin University, Australia
[2]Monash University Clayton, VIC 3800, Australia

Outline

- Deep Boltzmann Machines (DBMs)
- Problem statement
- Batch normalized DBMs (BNDBMs)
- Experiments
- Conclusion

Deep Boltzmann Machines (DBM)

- Energy-based generative model
  - probabilistic modeling *(Salakhudinov and Hinton, 2009)*
  - feature extraction *(Montavon et al., 2012).*
- Consider a **binary** DBM:
  - 1 visible layer $\boldsymbol{v} \in \{0,1\}^M$
  - 2 hidden layers $\boldsymbol{h} = \{\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}\}$
    - $h^{(1)} \in \{0,1\}^{K_1}$
    - $h^{(2)} \in \{0,1\}^{K_2}$



Deep Boltzmann Machines (DBM)

Definition: Energy function

$$E\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \Psi\right) = -\boldsymbol{a}^{\top}\boldsymbol{v} - \boldsymbol{b}^{(1)\top}\boldsymbol{h}^{(1)} - \boldsymbol{b}^{(2)\top}\boldsymbol{h}^{(2)} - \boldsymbol{v}^{\top}\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)} - \boldsymbol{h}^{(1)\top}\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)}$$

Joint distribution:

$$p\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \Psi\right) = \frac{e^{-E\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \Psi\right)}}{\mathcal{Z}(\Psi)}$$

Marginal distributions:

$$p\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}; \Psi\right)$$

$$p\left(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \Psi\right)$$

$$p\left(\boldsymbol{v}, \boldsymbol{h}^{(2)}; \Psi\right)$$

Conditional distributions:

$$p\left(\boldsymbol{v} | \boldsymbol{h}^{(1)}; \Psi\right)$$

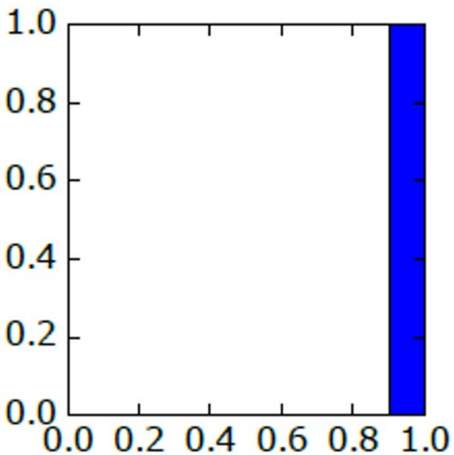
$$p\left(\boldsymbol{h}^{(1)} | \boldsymbol{v}, \boldsymbol{h}^{(2)}; \Psi\right)$$

$$p\left(\boldsymbol{h}^{(2)} | \boldsymbol{h}^{(1)}; \Psi\right)$$

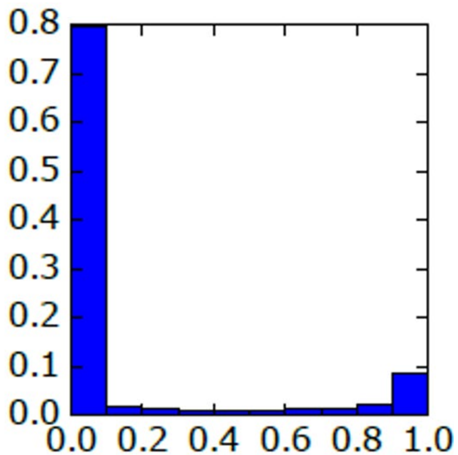
Problem: Internal Covariance Shift

- Hidden states:
  - Degenerate into 0 or 1
  - Not diverse

Hidden Layer 1's states

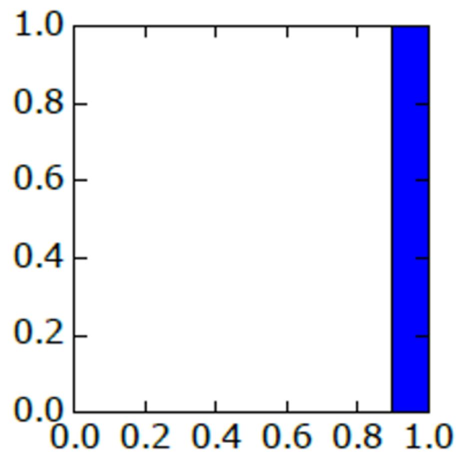


Unit 11

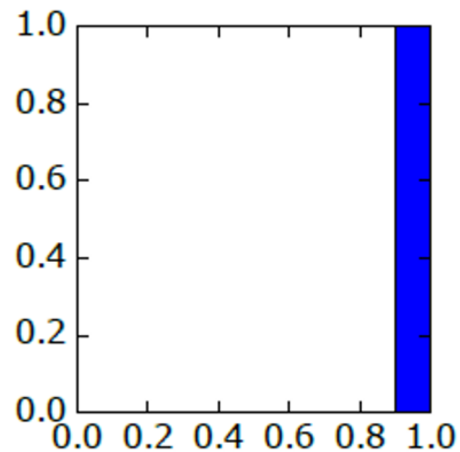


Unit 79

Hidden Layer 2 's states



Unit 17



Unit 100

Idea

- DBM training is also affected by Internal Covariance Shift
- Internal Covariance Shift is successfully solved for CNNs using Batch Normalization [1]
- Not investigated in DBMs

➔ This work:

$\boxed{\text{Investigate Batch Normalization for DBMs}}$

[1] *(Ioffe and Szegedy, 2015)*

Batch-normalization

- Batch Normalization ($\mathcal{B}$)
  - normalizes every dimension i of an input vector $\boldsymbol{x}$
  - normalized vector with zero-mean and unit-variance
  - scaled by γ_i and shifted by β_i

$$\mathcal{B}(x_i) = \gamma_i \frac{x_i - \mathbb{E}[x_i]}{\sqrt{\mathrm{Var}(x_i) + \epsilon}} + \beta_i$$

γ_i: scale parameter
β_i: shift parameter

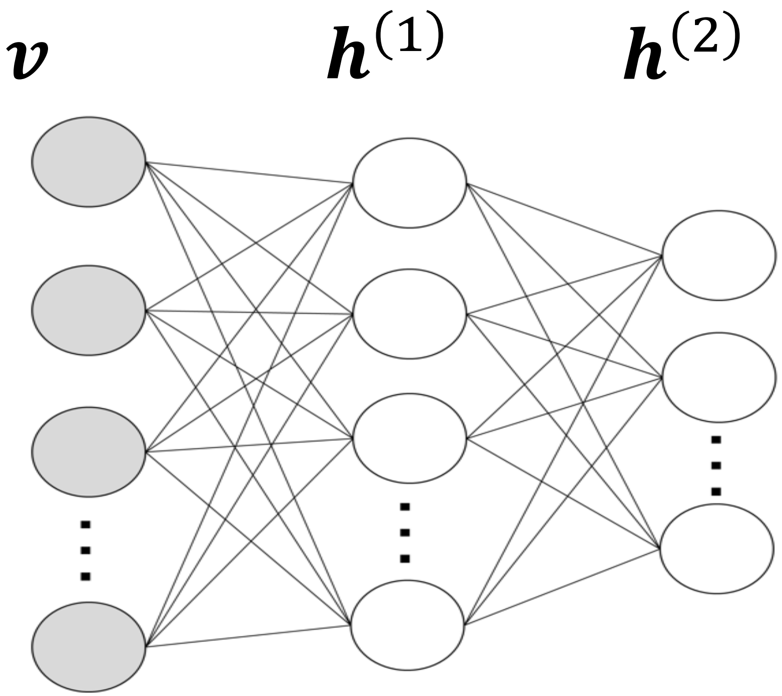
BN in DBMs: challenges

- BN in CNNs (deterministic networks)
  - Add one line of code

```
# code snippet using TensorFlow BN
training = tf.placeholder(tf.bool)
x = tf.layers.dense(input_x, units=100)
x = tf.layers.batch_normalization(x, training=training)
x = tf.nn.relu(x)
```

- BN in DBMs (probabilistic networks)
  - Add one line of code → incorrect
  - Why?

BN in DBMs: challenges



- Conditional distributions:
  - show how the information is passed between layers

$p(\boldsymbol{v}|\boldsymbol{h}^{(1)})$

$p(\boldsymbol{h}^{(1)}|\boldsymbol{v},\boldsymbol{h}^{(2)})$

$p(\boldsymbol{h}^{(2)}|\boldsymbol{h}^{(1)})$

$$\begin{cases} \mu_i^{(2)} = \sigma(b_i^{(2)} + \boldsymbol{h}^{(1)\top}\boldsymbol{w}_{.i}^{(1)}) \\ h_i^{(2)} \sim \text{Bernoulli}(\mu_i^{(2)}) \end{cases}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

BN in DBMs: challenges

- Conditional distribution:
  - show how the information is passed between layers
  - BN in DBM: new conditional distribution

$$\mu_i^{(2)} = \sigma(b_i^{(2)} + \boldsymbol{h}^{(1)\top}\boldsymbol{w}_{.i}^{(1)})$$

$$\downarrow$$

$$\mu_i^{(2)} = \sigma(\mathcal{B}(b_i^{(2)} + \boldsymbol{h}^{(1)\top}\boldsymbol{w}_{.i}^{(1)}))$$

$$p(\boldsymbol{h}^{(2)}|\boldsymbol{h}^{(1)}) \begin{cases} \mu_i^{(2)} = \sigma(b_i^{(2)} + \boldsymbol{h}^{(1)\top}\boldsymbol{w}_{.i}^{(1)}) \\ h_i^{(2)} \sim \mathrm{Bernoulli}(\mu_i^{(2)}) \end{cases} \qquad \sigma(x) = \frac{1}{1+e^{-x}}$$

BN in DBMs: challenges

Definition: Energy function E

↓

Joint distribution

↓

Marginal distributions

↓

$\boxed{\text{Conditional distributions}}$

- Conditional distribution:
  - show how the information is passed between layers
  - BN in DBM: **new** conditional distribution

➔ New energy function
  - change the whole model

Batch-normalized DBMs (BNDBMs)

- Standard energy function E (DBMs):

$$E\left(\boldsymbol{v},\boldsymbol{h}^{(1)},\boldsymbol{h}^{(2)};\boldsymbol{\Psi}\right)=-\boldsymbol{a}^{\top}\boldsymbol{v}-\boldsymbol{b}^{(1)\top}\boldsymbol{h}^{(1)}-\boldsymbol{b}^{(2)\top}\boldsymbol{h}^{(2)}-\boldsymbol{v}^{\top}\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)}-\boldsymbol{h}^{(1)\top}\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)}$$

- Proposed energy function E_{BN} (BNDBMS)

$$E_{BN}\left(v,\boldsymbol{h}^{(1)},\boldsymbol{h}^{(1)};\boldsymbol{\Gamma}\right)=-\sum_{i=1}^{M}a_i v_i-\sum_{l=1}^{2}\sum_{j=1}^{K_l}\left(\bar{\gamma}_j^{(l)}\bar{b}_j^{(l)}+\bar{\beta}_j^{(l)}\right)h_j^{(l)}$$
$$-\sum_{i=1}^{M}\sum_{j=1}^{K_1}\bar{\gamma}_j^{(1)}v_i w_{ij}^{(1)}h_j^{(1)}-\sum_{i=1}^{K_1}\sum_{j=1}^{K_2}\bar{\gamma}_i^{(1)}\,\bar{\gamma}_j^{(2)}h_i^{(1)}w_{ij}^{(2)}h_j^{(2)}$$

$\bar{\gamma}_i^{(l)}$ and $\bar{\beta}_i^{(l)}$: BN coefficients of the i^{th} neuron at the l^{th} layer

Batch-normalized DBMs (BNDBMs)

- Batch-normalized conditional distributions

$$p\left(v_m = 1 \middle| \boldsymbol{h}^{(1)}\right) = \sigma\left(a_m + \sum_{j=1}^{K_1} \bar{\gamma}_j^{(1)} w_{mj}^{(1)} h_j^{(1)}\right)$$

(No BN at visible layer)

$$p(h_n^{(1)} = 1|\boldsymbol{h}^{(2)}) = \sigma\left(\mathcal{B}_1(t_n^{(1)})\right)$$

$$t_n^{(1)} = b_n^{(1)} + \sum_{j=1}^{M} v_j \; w_{jn}^{(1)} + \sum_{j=1}^{K_2} \bar{\gamma}_j^{(2)} w_{nj}^{(2)} h_j^{(2)}$$

layer's input

$$p(h_n^{(2)} = 1|\boldsymbol{h}^{(1)}) = \sigma\left(\mathcal{B}_2(t_n^{(2)})\right)$$

$$t_n^{(2)} = b_n^{(2)} + \sum_{j=1}^{K_1} \bar{\gamma}_j^{(1)} h_j^{(1)} w_{jn}^{(2)}$$

layer's input

BNDBM – Model learning

- BNDBM training
  - same as vanilla DBMs
  - maximizing data log-likelihood

- Gradient of the log-likelihood

$$\boldsymbol{\nabla_{\Gamma} \log \mathcal{L}} = \mathbb{E}_{\text{data}}\left[-\frac{\partial E_{BN}}{\partial \Gamma}\right] - \mathbb{E}_{\text{model}}\left[-\frac{\partial E_{BN}}{\partial \Gamma}\right]$$

- Parameters Γ update

$$\Gamma = \Gamma + \eta \Delta \Gamma$$

Experiments

- **Datasets**: MNIST[1], Fashion-MNIST[2] and Caltech 101 Silhouette[3]
- **Network structures**: 784-500-100 and 784-500-500[4]
- **Training methods:**
  - Persistent Contrastive Divergence[5] (PCD)
  - Contrastive Divergence[6] (CD) without sampling step
- **Training settings**:
  - Batch size: 100, #Gibbs chains: 100, learning rate: 0.01, epochs: 500
  - BN parameters: learnable γ and $\beta = 0$

[1](Lecun et al., 1998), [2] (Xiao et al., 2017) , [3] (Marlin et al., 2010), [4](Montavon and Muller, 2012; Melchior et al., 2016), [5](Tieleman, 2008), [6](Hinton, 2002)

Experiments: Classification

- Classification evaluation:
  - train DBMs in unsupervised manner (without label information)
  - use the last hidden layer as features
  - train a Logistic Regression classifier
- Run 10 times

Experiments: Classification

Classification ↑	PCD (500-100)			PCD (500-500)			CD^{prob}(500-100)		
Pretraining	**DBM**	**cDBM**	**BNDBM**	**DBM**	**cDBM**	**BNDBM**	**DBM**	**cDBM**	**BNDBM**
MNIST	94.81 ±0.58	93.89 ±0.19	**95.37** ±0.19	96.68 ±0.36	96.63 ±0.15	**96.80** ±0.11	11.35 ±0.00	85.56 ±3.25	**90.31** ±0.002
Fashion-MNIST	81.27 ±1.07	73.60 ±2.7	**81.66** ±0.76	69.88 ±8.79	83.07 ±0.40	**84.70** ±0.38	15.92 ±6.60	71.97 ±2.12	**72.67** ±2.59
Caltech 101 Silhouette	65.27 ±0.24	58.39 ±0.58	**65.63** ±0.25	66.54 ±0.65	65.96 ±0.42	**69.20** ±0.45	28.08 ±3.13	54.40 ±0.99	**62.94** ±0.19
No pretraining	**DBM***	**cDBM***	**BNDBM***	**DBM***	**cDBM***	**BNDBM***	**DBM***	**cDBM***	**BNDBM***
MNIST	11.35 ±0.00	84.35 ±2.13	**93.81** ±0.58	12.22 ±2.74	94.05 ±0.08	**96.50** ±0.11	25.32 ±11.84	61.84 ±3.42	**85.73** ±0.81
Fashion-MNIST	16.14 ±9.91	71.64 ±1.15	**76.26** ±1.20	35.66 ±8.39	**82.27** ±0.27	78.75 ±0.99	12.31 ±0.66	68.25 ±3.19	**71.92** ±1.09
Caltech 101 Silhouette	19.77 ±4.78	54.70 ±0.78	**59.45** ±1.20	23.65 ±1.20	64.63 ±8.12	**66.80** ±0.69	35.34 ±3.98	55.95 ±0.79	**60.99** ±0.31
Our average improvement	**vs DBM**	**vs cDBM**		**vs DBM**	**vs cDBM**		**vs DBM**	**vs cDBM**	
Pretraining	0.44	5.60		5.86	1.67		56.95	4.76	
No pretraining	60.75	6.28		56.86	0.38		48.56	10.87	

➔ Better classification accuracy

*: no pretraining
Bold: best number
Underlined: next best

positive signs: better
negative signs: worse

Experiments: Reconstruction

- Reconstruction improvement evaluation
  - DBM/cDBM's reconstruction error - BNDBM's reconstruction error

Reconstruction	PCD (500-100)		PCD (500-500)		CD^{prob}(500-100)	
Average improvement	**vs DBM**	**vs cDBM**	**vs DBM**	**vs cDBM**	**vs DBM**	**vs cDBM**
Pretraining	-1.13	-1.50	-0.59	-0.64	**0.29**	**1.55**
No pretraining	-0.98	-1.86	**1.02**	-1.34	**0.06**	**3.57**

Bold/positive values: better

➔ Comparable reconstruction

1) Training facilitation

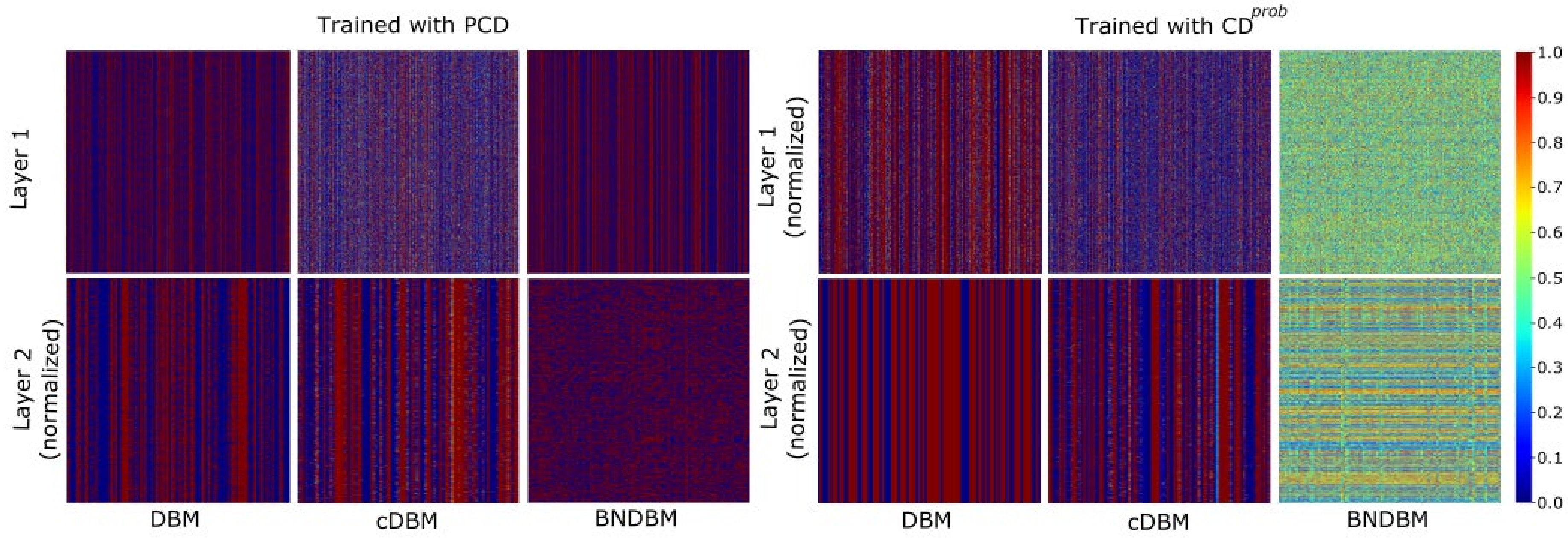
- DBMs and cDBMs
  - require pretraining
  - unsuccessfully train DBMs without pretraining
- BNDBMs
  - train without pretraining
  - 5.84% better than no-pretraining cDBMs

→ easy to train

Classification ↑	PCD (500-100)		
Pretraining	**DBM**	**cDBM**	**BNDBM**
MNIST	94.81 ±0.58	93.89 ±0.19	**95.37** ±0.19
Fashion-MNIST	81.27 ±1.07	73.60 ±2.7	**81.66** ±0.76
Caltech 101 Silhouette	65.27 ±0.24	58.39 ±0.58	**65.63** ±0.25
No pretraining	**DBM***	**cDBM***	**BNDBM**
MNIST	11.35 ±0.00	84.35 ±2.13	**93.81** ±0.58
Fashion-MNIST	16.14 ±9.91	71.64 ±1.15	**76.26** ±1.20
Caltech 101 Silhouette	19.77 ±4.78	54.70 ±0.78	**59.45** ±1.20
Our average improvement	**vs DBM**	**vs cDBM**	
Pretraining	0.44	5.60	
No pretraining	60.75	6.28	

2) Feature representation improvement

- 10,000 MNIST test samples vs hidden units (network 500-100)



- DBMs and cDBMs
  - vertical homogeneous strips → neurons respond to all data in a similar way
- BNDBMs
  - random/heterogeneous patterns → neurons are distinguishing data samples

→richer and more distinctive representation

Conclusion

- **Finding**
  - Internal Covariance Shift in DBM training
- **Solution**
  - Derive Batch Normalization for DBM
    - Via new energy function E_{BN}
- **Experiments**
  - Better classification and comparable reconstruction
  - Meaningful learning representations
  - No pretraining required
- **Analysis**: How to use BN in DBM efficiently (*more details in paper*)

References

Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann Machines. In AISTATS, volume 5, pages 448-455, 2009.

Gregoire Montavon, Mikio Braun, and Klaus-Robert Muller. Deep Boltzmann Machines as feed-forward hierarchies. In AISTATS, 21-23 Apr 2012.

KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Gaussian-Bernoulli Deep Boltzmann Machine. In IJCNN, pages 1{7, Dallas, TX, USA, August 4-9 2013a.

KyungHyun Cho, Tapani Raiko, Alexander Ilin, and Juha Karhunen. A Two-Stage Pretraining Algorithm for Deep Boltzmann Machines. In ICANN, September 10-13 2013b.

Jan Melchior, Asja Fischer, and Laurenz Wiskott. How to center Deep Boltzmann Machines. Journal of Machine Learning Research, 17(99):1-61, 2016.

Sergey Ioe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

Yann LeCun, L´ eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, number 11, 1998.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. 2017.

Benjamin M. Marlin, Kevin Swersky, Bo Chen, and Nando de Freitas. Inductive principles for Restricted Boltzmann Machine learning. In AISTATS, Italy, May 13-15 2010. PMLR.

Gregoire Montavon and Klaus Robert Muller. Deep Boltzmann Machines and the centering trick. In Neural Networks: Tricks of the Trade - Second Edition, pages 621–637. 2012.

Tijmen Tieleman. Training Restricted Boltzmann Machines using approximations to the likelihood gradient. In ICML, pages 1064–1071, New York, NY, USA, 2008.

References

G.E. Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation, 14(8):1771–1800, 2002.

Ruslan Salakhutdinov and Geoffrey Hinton. An efficient learning procedure for Deep Boltzmann Machines. Neural Computation, 24(8):1967–2006, Aug 2012.

Question

THANK YOU