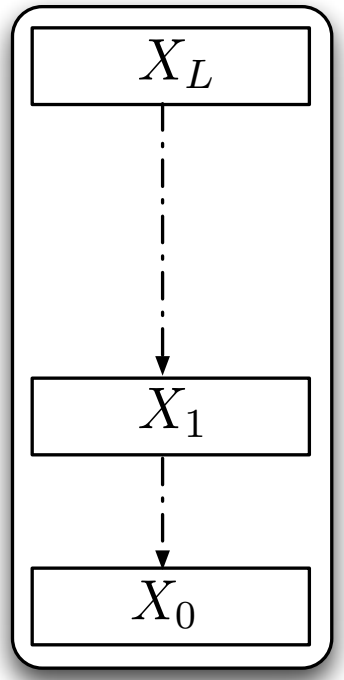
The Helmholtz machine revisited

Danilo Jimenez Rezende
BMI-EPFL

November 8, 2012

Helmholtz machines

Helmholtz machines
[Dayan et al., 1995, Dayan and Hinton, 1996, Dayan, 2000] are directed graphical models with a layered structure:

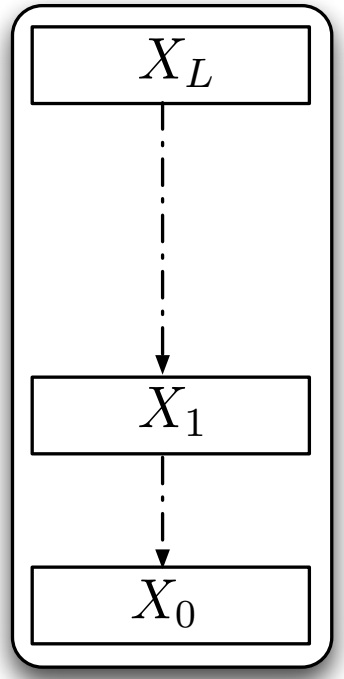


- Complete data likelihood:

$$p(X|\theta^g) = \prod_{l=0}^{L-1} p(X_l|X_{l+1}, \theta^g)p(X_L);$$

Helmholtz machines

Helmholtz machines
[Dayan et al., 1995, Dayan and Hinton, 1996, Dayan, 2000] are directed graphical models with a layered structure:



- Complete data likelihood:

$$p(X|\theta^g) = \prod_{l=0}^{L-1} p(X_l|X_{l+1}, \theta^g)p(X_L);$$

- Only X_0 is observed.

Types of Helmholtz machines

- Binary units:

$$p(X_l|X_{l+1}) = Bern \circ sigmoid(W_l^g X_{l+1} + B_l^g);$$

Parameters $\theta^g = \{W^g, B^g\}$.

Types of Helmholtz machines

- Binary units:

$$p(X_l|X_{l+1}) = Bern \circ sigmoid(W_l^g X_{l+1} + B_l^g);$$

Parameters $\theta^g = \{W^g, B^g\}$.

- Smooth units:

$$p(X_l|X_{l+1}) = N(X_l; \tanh(W_l^g X_{l+1} + B_l^g), \Sigma_l^g).$$

Parameters $\theta^g = \{W^g, B^g, \Sigma^g\}$.

Goal

For a data-set of iid samples $y \in Data$, maximize data log-likelihood w.r.t θ^g

$$\ln p(Data|\theta^g) = \sum_{y \in Data} \ln p(y|\theta^g),$$

where

$$p(y|\theta^g) = \int \prod_{l>0}^{L} dX_l \; p(y|X_1) \prod_{l=1}^{L-1} p(X_l|X_{l+1}, \theta^g) p(X_L)$$

The variational trick

Introduce a parametric family of distributions $q(X_{l>0}|X_0, \theta^r)$

The variational trick

Introduce a parametric family of distributions $q(X_{l>0}|X_0, \theta^r)$
Then

$$\ln p(X_0|\theta^g) = -\overbrace{\langle \ln q(X_{l>0}|X_0, \theta^r) - \ln p(X) \rangle_{q(X_{l>0}|X_0,\theta^r)}}^{\mathcal{F}(X_0,\theta^g,\theta^r)} + KL(q; p),$$

where

The variational trick

Introduce a parametric family of distributions $q(X_{l>0}|X_0, \theta^r)$
Then

$$\ln p(X_0|\theta^g) = -\overbrace{\langle \ln q(X_{l>0}|X_0, \theta^r) - \ln p(X) \rangle_{q(X_{l>0}|X_0,\theta^r)}}^{\mathcal{F}(X_0,\theta^g,\theta^r)} + KL(q;p),$$

where

$$KL(q;p) = \langle \ln \frac{q(X_{l>0}|X_0, \theta^r)}{p(X_{l>0}|X_0, \theta^g)} \rangle_{q(X_{l>0}|X_0,\theta^r)}$$

The variational trick

Introduce a parametric family of distributions $q(X_{l>0}|X_0, \theta^r)$
Then

$$\ln p(X_0|\theta^g) = -\overbrace{\langle \ln q(X_{l>0}|X_0, \theta^r) - \ln p(X) \rangle_{q(X_{l>0}|X_0,\theta^r)}}^{\mathcal{F}(X_0,\theta^g,\theta^r)} + KL(q; p),$$

where

$$KL(q; p) = \langle \ln \frac{q(X_{l>0}|X_0, \theta^r)}{p(X_{l>0}|X_0, \theta^g)} \rangle_{q(X_{l>0}|X_0,\theta^r)}$$

$$KL \geq 0 \Rightarrow \ln p(X_0, \theta^g) \geq -\mathcal{F}(X_0, \theta^g, \theta^r) \quad \forall \theta^r$$

The variational trick

Redefine the learning problem as

$$\{\hat{\theta}^g, \hat{\theta}^r\} = \arg\min_{\theta^g, \theta^r} \sum_{x \in Data} \mathcal{F}(x, \theta^g, \theta^r),$$

The variational trick

Redefine the learning problem as

$$\{\hat{\theta}^g, \hat{\theta}^r\} = \arg\min_{\theta^g, \theta^r} \sum_{x \in Data} \mathcal{F}(x, \theta^g, \theta^r),$$

Why ?

- q can be any distribution with same support as p

The variational trick

Redefine the learning problem as

$$\{\hat{\theta}^g, \hat{\theta}^r\} = \arg\min_{\theta^g, \theta^r} \sum_{x \in Data} \mathcal{F}(x, \theta^g, \theta^r),$$

Why ?

- q can be any distribution with same support as p
- Choose q so that we can calculate expectations in a reasonable time

The variational trick

Redefine the learning problem as

$$\{\hat{\theta}^g, \hat{\theta}^r\} = \arg\min_{\theta^g, \theta^r} \sum_{x \in Data} \mathcal{F}(x, \theta^g, \theta^r),$$

Why ?

- q can be any distribution with same support as p
- Choose q so that we can calculate expectations in a reasonable time
- Allows to solve inference using standard optimization techniques

Fully factorized q

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l>0,i} p(X_l|\theta^r_{l,i}),$$

Fully factorized q

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l>0,i} p(X_l|\theta^r_{l,i}),$$

- Expectations are analytically tractable for sigmoid/tanh nonlinearities [Frey, 1996, Jordan, 1999]

Fully factorized q

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l>0,i} p(X_l|\theta^r_{l,i}),$$

- Expectations are analytically tractable for sigmoid/tanh nonlinearities [Frey, 1996, Jordan, 1999]
- Yields local message-passing type of algorithms

Fully factorized q

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l>0,i} p(X_l|\theta^r_{l,i}),$$

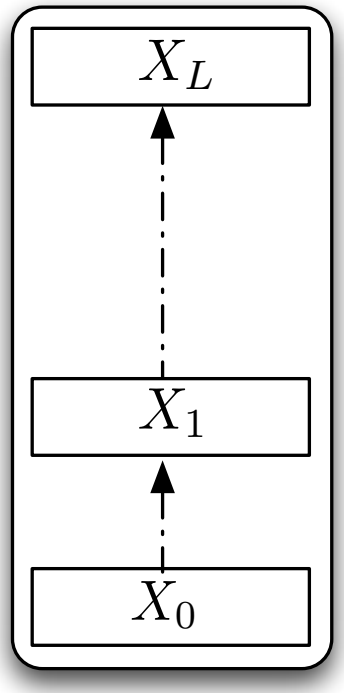
- Expectations are analytically tractable for sigmoid/tanh nonlinearities [Frey, 1996, Jordan, 1999]
- Yields local message-passing type of algorithms
- Fast convergence

Fully factorized q

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l>0,i} p(X_l|\theta^r_{l,i}),$$

- Expectations are analytically tractable for sigmoid/tanh nonlinearities [Frey, 1996, Jordan, 1999]
- Yields local message-passing type of algorithms
- Fast convergence
- Bad approximation to multimodal posteriors

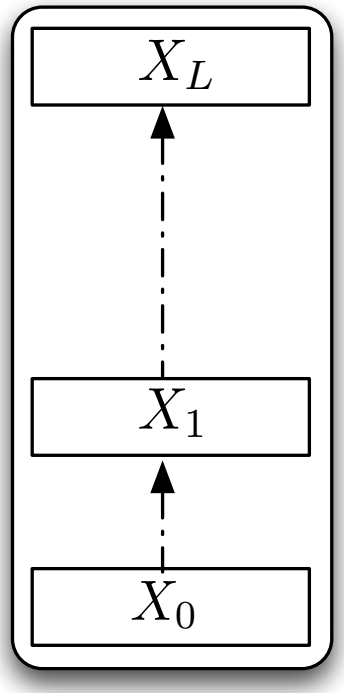
Helmholtz' machine choice of q



- Bottom-up graph [Dayan et al., 1995, Dayan and Hinton, 1996, Dayan, 2000]:

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l=1}^{L} p(X_l|X_{l-1}, \theta^r)$$

Helmholtz' machine choice of q

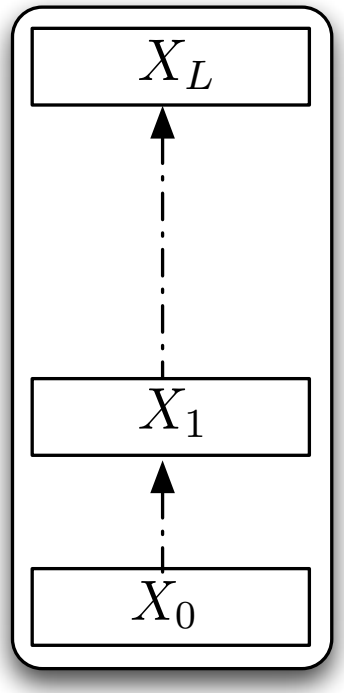


- Bottom-up graph [Dayan et al., 1995, Dayan and Hinton, 1996, Dayan, 2000]:

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l=1}^{L} p(X_l|X_{l-1}, \theta^r)$$

- Conditioned on X_0

Helmholtz' machine choice of q

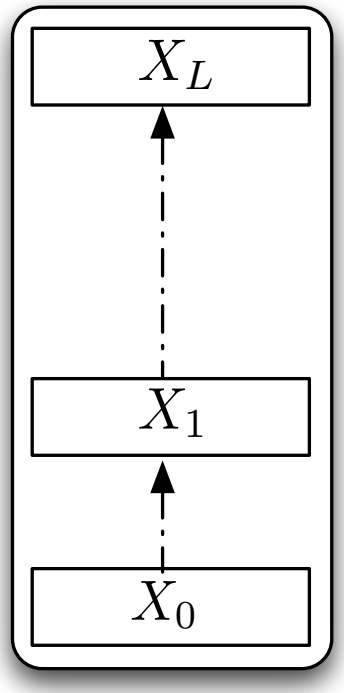


- Bottom-up graph [Dayan et al., 1995, Dayan and Hinton, 1996, Dayan, 2000]:

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l=1}^{L} p(X_l|X_{l-1}, \theta^r)$$

- Conditioned on X_0
- Cannot solve any expectation analytically

Helmholtz' machine choice of q



- Bottom-up graph [Dayan et al., 1995, Dayan and Hinton, 1996, Dayan, 2000]:

$$q(X_{l>0}|X_0, \theta^r) = \prod_{l=1}^{L} p(X_l|X_{l-1}, \theta^r)$$

- Conditioned on X_0
- Cannot solve any expectation analytically
- Resort to Monte Carlo approximations

Point-estimate of the free energy

$$\begin{aligned}\mathcal{F}(X_0, \theta^g, \theta^r) &= \langle \ln q(X_{l>0}|X_0, \theta^r) - \ln p(X) \rangle_{q(X_{l>0}|X_0,\theta^r)} \\ &= \langle \hat{\mathcal{F}} \rangle_{q(X_{l>0}|X_0,\theta^r)},\end{aligned}$$

where

$$\hat{\mathcal{F}} = \ln q(X_{l>0}|X_0, \theta^r) - \ln p(X)$$

REINFORCE

Simple, unbiased, stochastic gradient estimator

- Update for θ^g :

$$\delta\theta^g \propto -\nabla_{\theta^g}\mathcal{F}(X_0, \theta^g, \theta^r) = \langle \nabla_{\theta^g} \ln p \rangle_q$$

REINFORCE

Simple, unbiased, stochastic gradient estimator

- Update for θ^g :

$$\delta\theta^g \propto -\nabla_{\theta^g}\mathcal{F}(X_0, \theta^g, \theta^r) = \langle \nabla_{\theta^g} \ln p \rangle_q$$

- Update for θ^r :

$$\delta\theta^r \propto -\mathcal{F}(X_0, \theta^g, \theta^r) = \langle (\hat{\mathcal{F}} - b)\nabla_{\theta^r} \ln q \rangle_q,$$

where

$$b \approx \langle \hat{\mathcal{F}} \rangle_q$$

REINFORCE

Simple, unbiased, stochastic gradient estimator

- Update for θ^g :

$$\delta\theta^g \propto -\nabla_{\theta^g}\mathcal{F}(X_0, \theta^g, \theta^r) = \langle \nabla_{\theta^g} \ln p \rangle_q$$

- Update for θ^r :

$$\delta\theta^r \propto -\mathcal{F}(X_0, \theta^g, \theta^r) = \langle (\hat{\mathcal{F}} - b)\nabla_{\theta^r} \ln q \rangle_q,$$

where

$$b \approx \langle \hat{\mathcal{F}} \rangle_q$$

REINFORCE

Simple, unbiased, stochastic gradient estimator

- Update for θ^g :

$$\delta\theta^g \propto -\nabla_{\theta^g}\mathcal{F}(X_0, \theta^g, \theta^r) = \langle \nabla_{\theta^g} \ln p \rangle_q$$

- Update for θ^r :

$$\delta\theta^r \propto -\mathcal{F}(X_0, \theta^g, \theta^r) = \langle (\hat{\mathcal{F}} - b)\nabla_{\theta^r} \ln q \rangle_q,$$

where

$$b \approx \langle \hat{\mathcal{F}} \rangle_q$$

Scales badly

$$Var[(\hat{\mathcal{F}} - b)\nabla_{\theta^r} \ln q] \approx O(\text{Nr of hidden nodes})$$

Wake-Sleep: The wrong gradient that works

- Update for θ^g (or the "wake" phase) : $\delta\theta^g \propto \langle \nabla_{\theta^g} \ln p \rangle_q$

Wake-Sleep: The wrong gradient that works

- Update for θ^g (or the "wake" phase) : $\delta\theta^g \propto \langle \nabla_{\theta^g} \ln p \rangle_q$
- Update for θ^r (or the "sleep" phase) : $\delta\theta^r \propto \langle \nabla_{\theta^r} \ln q \rangle_p$,

Wake-Sleep: The wrong gradient that works

- Update for θ^g (or the "wake" phase) : $\delta\theta^g \propto \langle \nabla_{\theta^g} \ln p \rangle_q$
- Update for θ^r (or the "sleep" phase) : $\delta\theta^r \propto \langle \nabla_{\theta^r} \ln q \rangle_p$,

Wake-Sleep: The wrong gradient that works

- Update for θ^g (or the "wake" phase) : $\delta\theta^g \propto \langle \nabla_{\theta^g} \ln p \rangle_q$
- Update for θ^r (or the "sleep" phase) : $\delta\theta^r \propto \langle \nabla_{\theta^r} \ln q \rangle_p$,

Why it is wrong ?

Minimizing $KL(p;q)$ instead of $KL(q;p)$

Wake-Sleep: The wrong gradient that works

- Update for θ^g (or the "wake" phase) : $\delta\theta^g \propto \langle \nabla_{\theta^g} \ln p \rangle_q$
- Update for θ^r (or the "sleep" phase) : $\delta\theta^r \propto \langle \nabla_{\theta^r} \ln q \rangle_p$,

Why it is wrong ?

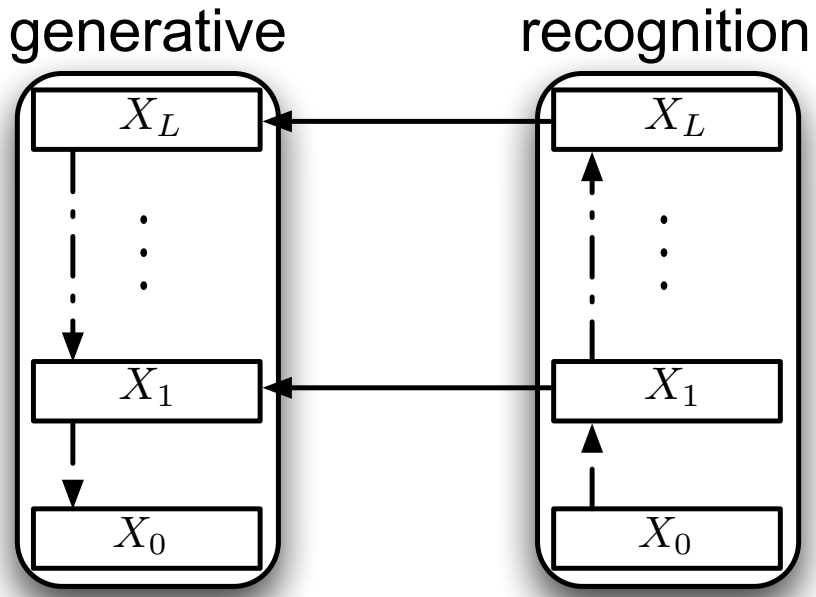
Minimizing $KL(p;q)$ instead of $KL(q;p)$

Why and when it works?

If for any θ^g, $\exists\theta^r$ such that $q(X_{l>0}|X_0,\theta^r) = p(X_{l>0}|X_0,\theta^g)$ then [Ikeda et al., 1999]

$$\arg\min_q KL(p;q) = \arg\min_q KL(q;p)$$

Wake-Sleep: The wrong gradient that works



Summary

- REINFORCE is unbiased but scales badly

Summary

- REINFORCE is unbiased but scales badly
- Wake-sleep scales nicely, but it is wrong

Summary

- REINFORCE is unbiased but scales badly
- Wake-sleep scales nicely, but it is wrong
- None of them exploits specific properties of exponential family pdfs

Why REINFORCE is so bad?

REINFORCE is based on Likelihood-Ratio:

$$\nabla_\theta \langle f(x) \rangle_{p(x)} = \langle f(x) \nabla_\theta \ln p(x) \rangle_{p(x)} = \langle (f(x) - b) \nabla_\theta \ln p(x) \rangle_{p(x)}$$

Why REINFORCE is so bad?

REINFORCE is based on Likelihood-Ratio:

$$\nabla_\theta \langle f(x) \rangle_{p(x)} = \langle f(x) \nabla_\theta \ln p(x) \rangle_{p(x)} = \langle (f(x) - b) \nabla_\theta \ln p(x) \rangle_{p(x)}$$

If $p(x) = N(x; \mu, \Sigma)$ ($\theta = \{\mu, \Sigma\}$) we can do much better:

- Bonnet's theorem:

$$\nabla_{\mu_i} \langle f(x) \rangle_{p(x)} = \langle \frac{\partial}{\partial x_i} f(x) \rangle_{p(x)}$$

Why REINFORCE is so bad?

REINFORCE is based on Likelihood-Ratio:

$$\nabla_\theta \langle f(x) \rangle_{p(x)} = \langle f(x) \nabla_\theta \ln p(x) \rangle_{p(x)} = \langle (f(x) - b) \nabla_\theta \ln p(x) \rangle_{p(x)}$$

If $p(x) = N(x; \mu, \Sigma)$ ($\theta = \{\mu, \Sigma\}$) we can do much better:

- Bonnet's theorem:

$$\nabla_{\mu_i} \langle f(x) \rangle_{p(x)} = \langle \frac{\partial}{\partial x_i} f(x) \rangle_{p(x)}$$

- Price's theorem:

$$\nabla_{\Sigma_{i,j}} \langle f(x) \rangle_{p(x)} = (\frac{1}{2})^{\delta_{i,j}} \langle \frac{\partial^2}{\partial x_i \partial x_j} f(x) \rangle_{p(x)}$$

Rules for stochastic backpropagation (SBP)

Let $V(y)$ be some cost function on y and $\varepsilon_y = \frac{\partial V(y)}{\partial y}$ and $\lambda_y = \frac{\partial^2 V(y)}{\partial y^2}$ be its first and second derivatives.

- If $y = \mu(x)$ (deterministic case) then

$$\varepsilon_x = \varepsilon_y|_{y=\mu(x)} \frac{\partial \mu(x)}{\partial x}$$

Rules for stochastic backpropagation (SBP)

Let $V(y)$ be some cost function on y and $\varepsilon_y = \frac{\partial V(y)}{\partial y}$ and $\lambda_y = \frac{\partial^2 V(y)}{\partial y^2}$ be its first and second derivatives.

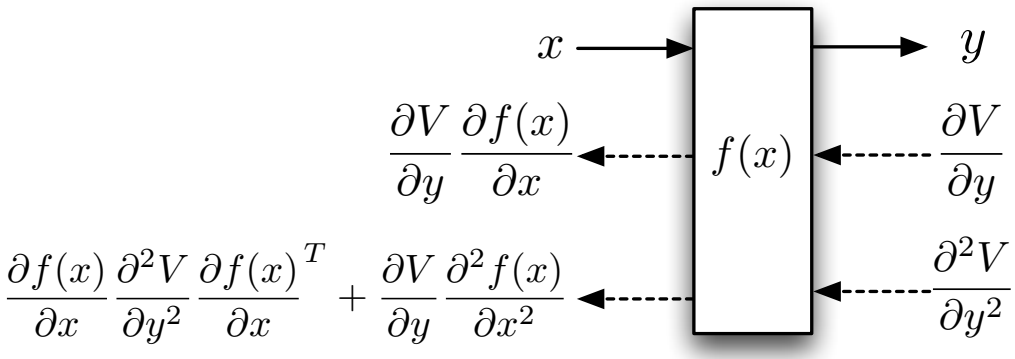
- If $y = \mu(x)$ (deterministic case) then

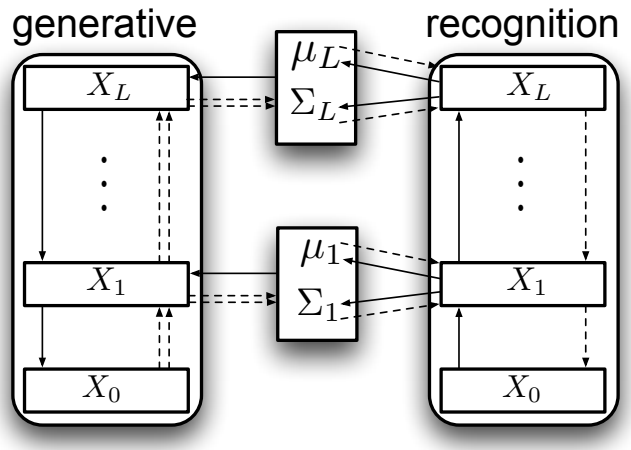
$$\varepsilon_x = \varepsilon_y|_{y=\mu(x)} \frac{\partial \mu(x)}{\partial x}$$

- If $y \sim N(y|\mu(x), \Sigma(x))$ (stochastic case) then

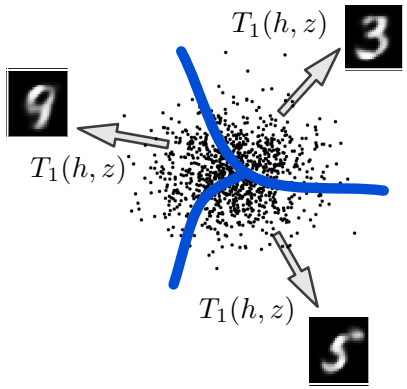
$$\varepsilon_x = \langle \overbrace{\varepsilon_y \frac{\partial \mu(x)}{\partial x}}^{\text{"Drift"}} + \underbrace{\frac{1}{2}\lambda_y \frac{\partial \Sigma(x)}{\partial x}}_{\text{"Diffusion"}} \rangle_{N(y|\mu(x),\Sigma(x))}$$

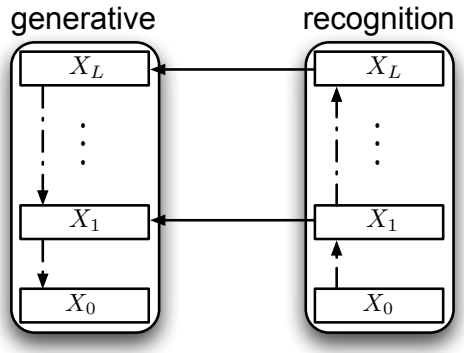
Proposed smooth recognition model

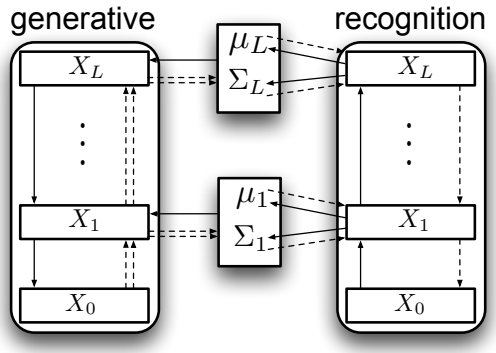




Proposed smooth recognition model







Learning procedure: algorithm 1

while isLearning() **do**
    $v \leftarrow$ getNewDataSample()
    bottomUp()

Learning procedure: algorithm 1

while isLearning() **do**
$v \leftarrow$ getNewDataSample()
bottomUp()
backpropagateThroughGenerativeModel()
$\varepsilon_l^{\mu} \leftarrow \nabla_{h_l} \mathcal{F}(v, h, z)$
$\varepsilon_l^{\Sigma} \leftarrow \left(\frac{1}{2}\right)^{\delta_{i,j}} \nabla^2_{h_{l,i}, h_{l,j}} \mathcal{F}(v, h, z)$

Learning procedure: algorithm 1

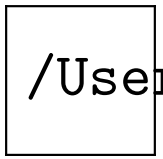
while isLearning() **do**
$v \leftarrow$ getNewDataSample()
bottomUp()
backpropagateThroughGenerativeModel()
$\varepsilon_l^{\mu} \leftarrow \nabla_{h_l} \mathcal{F}(v, h, z)$
$\varepsilon_l^{\Sigma} \leftarrow \left(\frac{1}{2}\right)^{\delta_{i,j}} \nabla^2_{h_{l,i}, h_{l,j}} \mathcal{F}(v, h, z)$
backpropagateThroughRecognitionModel()

Learning procedure: algorithm 1

while isLearning() **do**
$v \leftarrow$ getNewDataSample()
bottomUp()
backpropagateThroughGenerativeModel()
$\varepsilon_l^{\mu} \leftarrow \nabla_{h_l} \mathcal{F}(v, h, z)$
$\varepsilon_l^{\Sigma} \leftarrow \left(\frac{1}{2}\right)^{\delta_{i,j}} \nabla^2_{h_{l,i}, h_{l,j}} \mathcal{F}(v, h, z)$
backpropagateThroughRecognitionModel()
applyGradients()

Performance comparison on MNIST

Training set 60000 28x28 grey images; Test set 10000 images



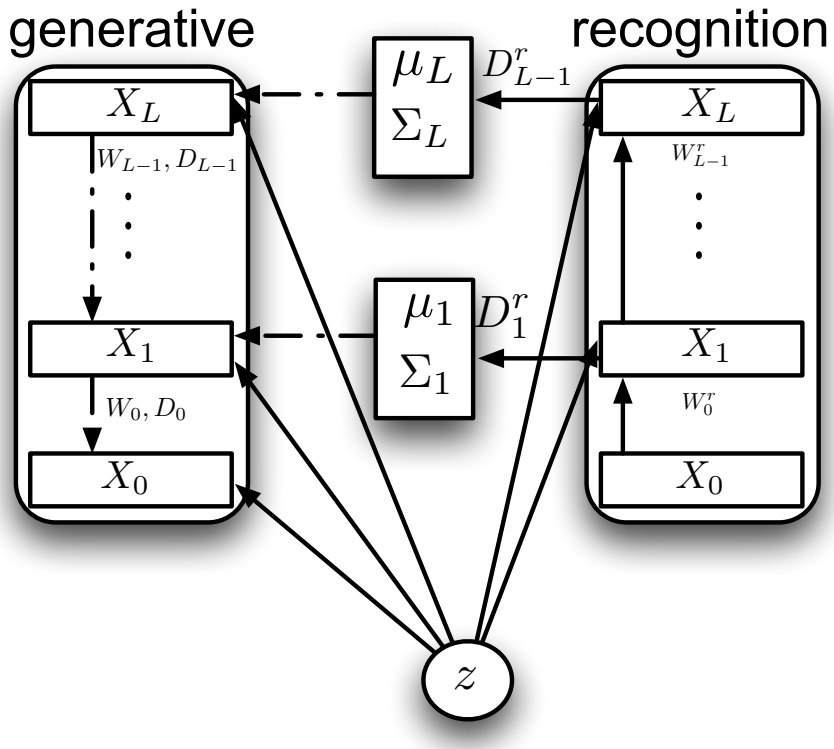
/Users/danilo/workspace/RCHMII/sim

Performance comparison on MNIST

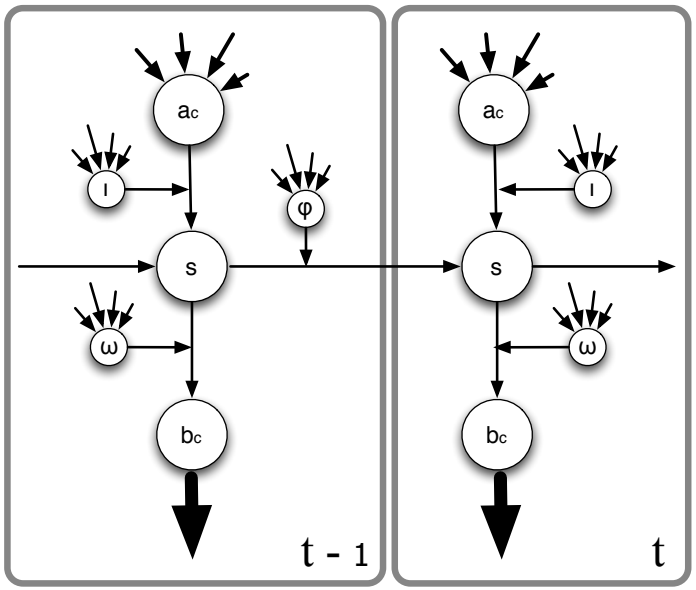
Model	Parameters	Log-Likelihood
HM	$Nh = 200 + 50$, Wake-Sleep	-157
HM	$Nh = 200 + 50$, REINFORCE	-145
Mix. Bernoulli	$N = 500$	-137.64
RBM	$Nh = 500$, CD_1	-125.53
HM	$Nh = 200 + 50$, BP-SBP	-108.35
RBM	$Nh = 500$, CD_3	-105.53
NADE	$N = 500$	-88.86
RBM	$Nh = 500$, CD_{25}	-86.86

RBM, Mix. Bernoulli and NADE log-likelihoods from [Larochelle and Murray, 2011].

Making the model conditional

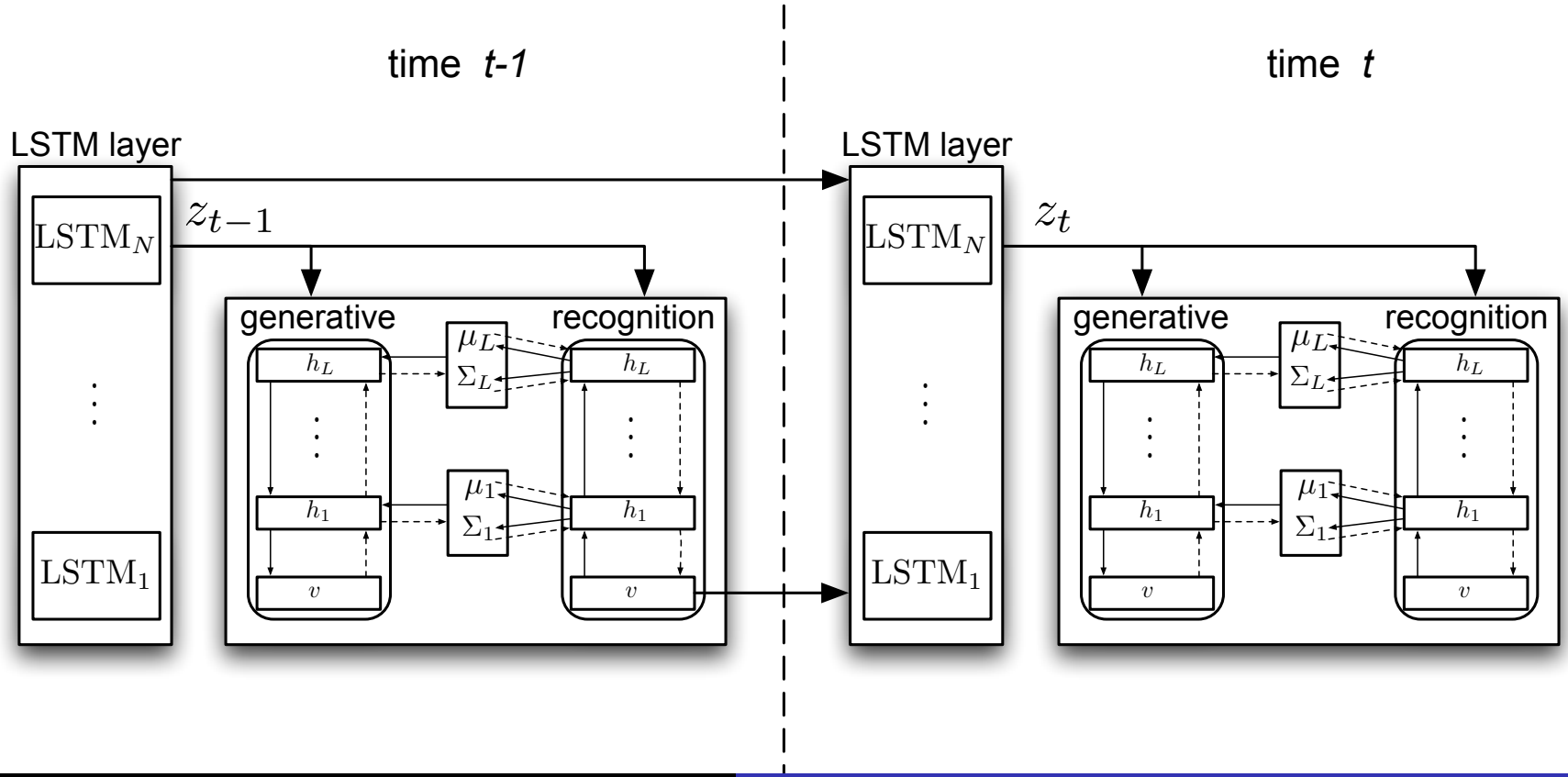


Conditioning on a RNN of LSTMs (Differentiable Memory cells)



Idea suggested (but not implemented) by [Sutskever and Hinton, 2008].

Full model


time t-1
time t
LSTM layer
LSTM layer
LSTM_N
LSTM_1
z_{t-1}
z_t
generative
recognition
h_L
h_1
v
μ_L
Σ_L
μ_1
Σ_1

Learning procedure: algorithm 2

while isLearning() **do**
  resetGradients()
  for $t = 1 \rightarrow T$ **do**
    forwardLSTM(t)
    $v_t \leftarrow$ getDataFrame(t)
    bottomUp()
    backpropagateThroughGenerativeModel()
    backpropagateThroughRecognitionModel()
    $\varepsilon_t^{\text{LSTM}} \leftarrow \nabla_{z_t} \mathcal{F}(v_t, h_t, z_t)$
    incrementGradientsOfGenerativeAndRecognitionModels()
  for $t = T \rightarrow 1$ **do**
    LSTM.BPTT(t)
  applyGradients()

Benchmarks: temporal sequences

Final remarks

- Performance is similar to RBM + CD_3

Final remarks

- Performance is similar to RBM + CD_3
- Not yet competitive with RBM + CD_{25}

Final remarks

- Performance is similar to RBM + CD_3
- Not yet competitive with RBM + CD_{25}
- Exploiting specific properties of the noise and graph is substantially better than Wake-Sleep and REINFORCE

Final remarks

- Performance is similar to RBM + CD_3
- Not yet competitive with RBM + CD_{25}
- Exploiting specific properties of the noise and graph is substantially better than Wake-Sleep and REINFORCE
- How much would we benefit from having less constrained covariance matrices?

THANK YOU!

Bibliography

Dayan, P. (2000).

Helmholtz machines and wake-sleep learning.
Handbook of Brain Theory and Neural Network. MIT Press, Cambridge, MA, 44(0).

Dayan, P. and Hinton, G. E. (1996).

Varieties of Helmholtz Machine.
Neural Networks, 9(8):1385–1403.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995).

The Helmholtz machine.
Neural computation, 7(5):889–904.

Frey, B. J. (1996).

Variational inference for continuous sigmoidal Bayesian networks.
IN SIXTH INTERNATIONAL WORKSHOP ON ARTIFICIAL INTELLIGENCE AND STATISTICS.

Ikeda, S., Nakahara, H., and Amari, S.-i. (1999).

Convergence of The Wake-Sleep Algorithm.
ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 11, pages 239 – 245.

Jordan, M. I. (1999).

An introduction to variational methods for graphical models.
MACHINE LEARNING, 37:183 – 233.

Larochelle, H. and Murray, I. (2011).

The Neural Autoregressive Distribution Estimator.

Estimating the data log-likelihood

- Naive:

$$\ln p(X_0) \approx \ln \frac{1}{K} \sum_k p(X_0|X_1^k),$$

where X_1^k, $k = 1 \ldots K$, are samples from p.

Estimating the data log-likelihood

- Naive:

$$\ln p(X_0) \approx \ln \frac{1}{K} \sum_k p(X_0|X_1^k),$$

where X_1^k, $k = 1 \ldots K$, are samples from p.

- Importance sampling:

$$\ln p(X_0) \approx \ln \frac{1}{K} \sum_k \exp -\hat{\mathcal{F}}_k,$$

where $\hat{\mathcal{F}}_k$ is a point-estimate of $\mathcal{F}$ using a sample from q.