Deep Generative Models

Durk Kingma

Max Welling

Deep Probabilistic Models Workshop
Wednesday, 1st of Oct, 2014
D.P. Kingma

UNIVERSITEIT VAN AMSTERDAM

Deep generative models

- Transformations between Bayes nets and Neural nets
  "Transformation between Bayes nets and Neural Nets". ***ICML'14***

- Deep generative models of images
  Auto-Encoding Variational Bayes, ***ICLR'14***

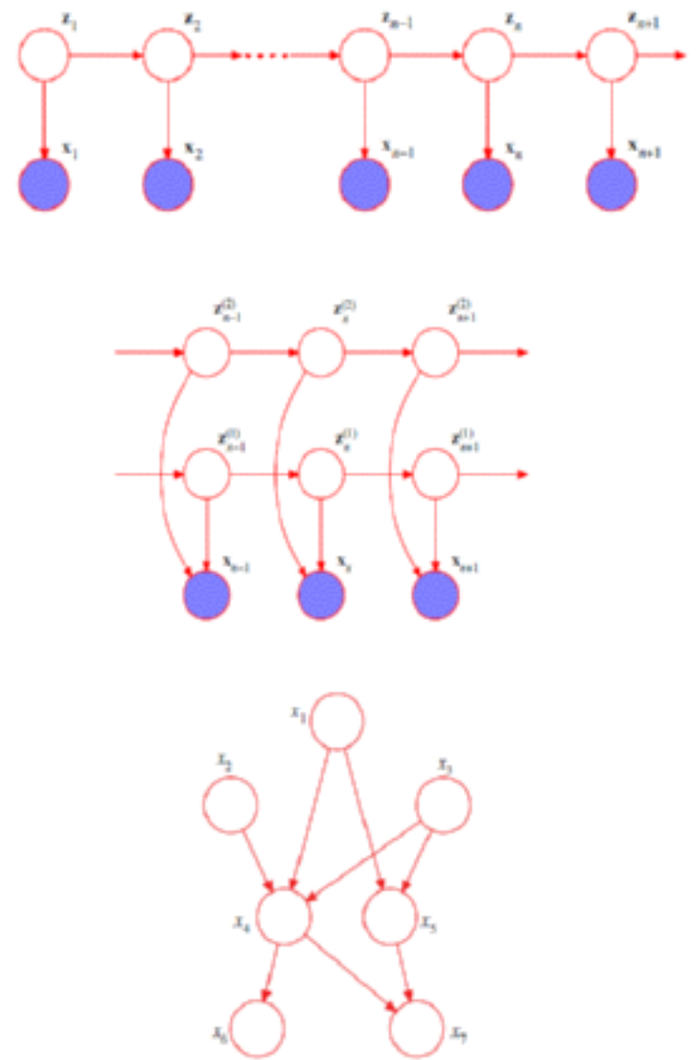
- Semi-Supervised Learning with Deep Generative Models
  NIPS'14; with Shakir & Danilo (Deepmind)

Part 1:

Transformations between Bayes nets and Neural nets

Neural Nets and Deep Generative Models (1)

- Directed latent variables models
  - Can model complex (multimodal) distributions over observed variables
- Feedforward neural nets:
  - Typically used for learning a single conditional: log p(y|x) = f(x,y)
    - e.g. Classification/regression with deep net
- But: not good for modelling complex distributions, e.g. multi-modal distributions



Neural Nets and Deep Generative Models (2)

- Natural idea: neural nets as components of deep latent-variable models
  - Learn complex distributions over data

- Inner loop of learning requires **posterior inference**
  - Exact inference is intractable

- Efficient gradient-based approximate inference methods:
  - Stochastic Variational inference (biased but fast)
    [Hoffman & Blei 2012] [Kingma & Welling 2013] [Deepmind 2014]
  - Sampling-based methods (unbiased but slower), e.g. HMC

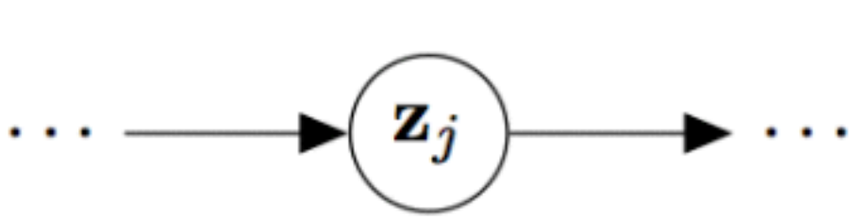
Can we increase the efficiency
of gradient-based posterior inference?

$$p(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{x}, \mathbf{z})$$

$$\nabla_{\mathbf{z}} \log p(\mathbf{z}|\mathbf{x}) = \nabla_{\mathbf{z}} \log p(\mathbf{x}, \mathbf{z})$$

Idea: Reparameterizations (Gaussian example)

Centered form
(CP)

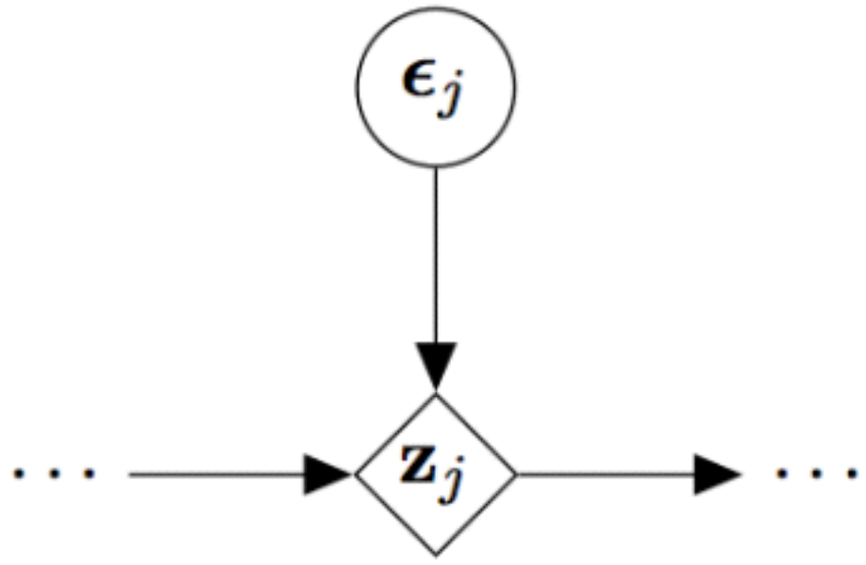


$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{pa}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

$$\boldsymbol{\mu} = f_{\boldsymbol{\theta}}(\mathbf{pa})$$

(e.g. neural net)

Differentiable Non-Centered Form
(DNCP)



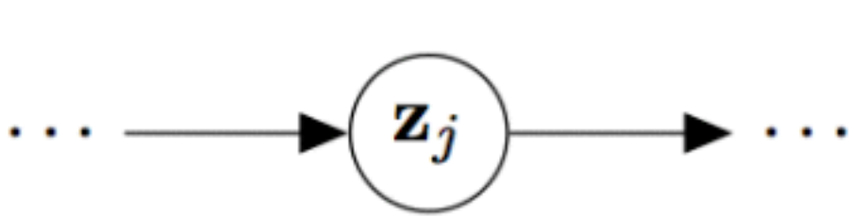
$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

$$\widetilde{\mathbf{z}} = f_{\boldsymbol{\theta}}(\mathbf{pa}) + \sigma \cdot \boldsymbol{\epsilon}$$

Neural net perspective:
hidden unit with injected noise

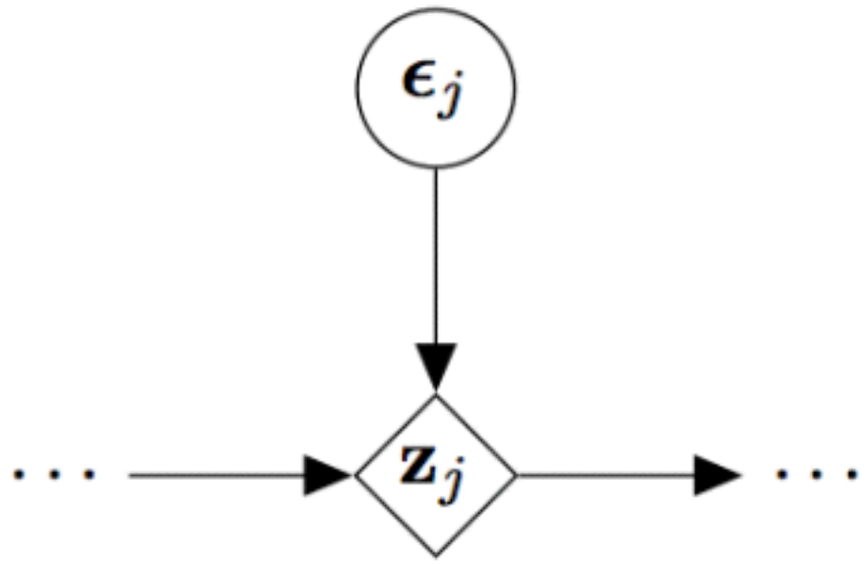
Idea: Reparameterizations

Centered form
(CP)



$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{pa})$$

Differentiable Non-Centered Form
(DNCP)



$$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$
$$\mathbf{z} = g(\mathbf{pa}, \boldsymbol{\epsilon}, \boldsymbol{\theta})$$

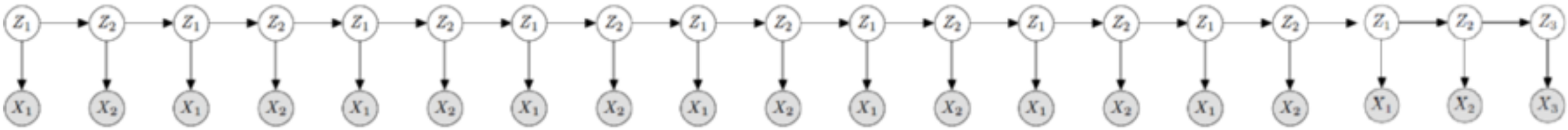
Neural net perspective:
hidden unit with injected noise

Reparameterizations

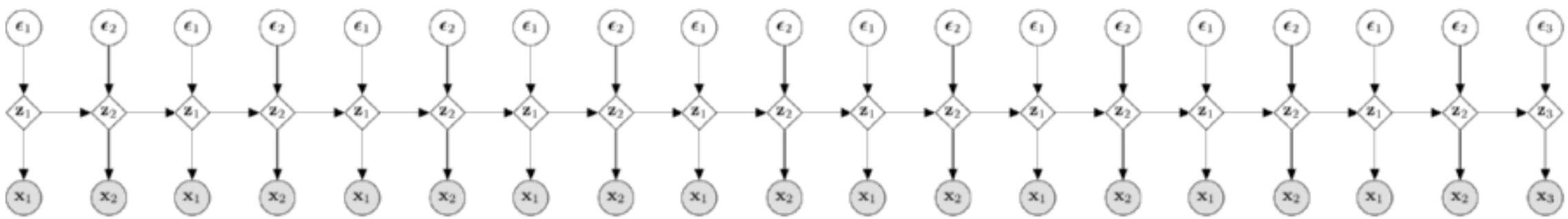
- Can be performed for a broad class of distributions

Example	$q_\varphi(z)$	$p(\varepsilon)$	$g(\varphi, \varepsilon)$	Also...
Normal dist.	$z \sim N(\mu,\sigma)$	$\varepsilon \sim N(0,1)$	$z = \mu + \sigma * \varepsilon$	Location-scale familie: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular, ...
Exponential	$z \sim \exp(\lambda)$	$\varepsilon \sim U(0,1)$	$z = -\log(1 - \varepsilon)/\lambda$	Invertible CDF: Cauchy, Logistic, Rayleigh, Pareto,Weibull, Reciprocal, Gompertz, Gumbel and Erlan, ...
Other	$z \sim \log N(\mu,\sigma)$	$\varepsilon \sim N(0,1)$	$z = \exp(\mu + \sigma * \varepsilon)$	Gamma, Dirichlet, Beta, Chi-Squared, and F distributions

So: we have a choice



Which form to use for learning / inference?



Posterior correlation analysis (1)

Squared posterior correlation ρ^2
second-order metric of posterior dependency
between pair of latent variables A and B

$$C = \begin{pmatrix} \sigma_A^2 & \sigma_{AB}^2 \\ \sigma_{AB}^2 & \sigma_B^2 \end{pmatrix} = -\mathbf{H}^{-1} = \frac{1}{det(\mathbf{H})} \begin{pmatrix} -H_B & H_{AB} \\ H_{AB} & -H_A \end{pmatrix}$$

$$\begin{aligned} \rho^2 &= (\sigma_{AB}^2)^2 / (\sigma_A^2 \sigma_B^2) \\ &= (H_{AB}/det(\mathbf{H}))^2 / ((-H_A/det(\mathbf{H}))(-H_B/det(\mathbf{H}))) \\ &= H_{AB}^2 / (H_A H_B) \end{aligned}$$

Posterior correlation analysis (2)

- Squared correlations for CP:

$$\rho^2_{pa(z)_i,z} = \frac{(H_{pa(z)_i z})^2}{H_{pa(z)_i pa(z)_i} H_{zz}} = \frac{w_i^2/\sigma^4}{(\alpha - w_i^2/\sigma^2)(\sigma^{-2}_{z|ch(z)} - 1/\sigma^2)}$$

- Squared correlations for DNCP:

$$\rho^2_{pa(z)_i,\epsilon} = \frac{(H_{pa(z)_i \epsilon})^2}{H_{pa(z)_i pa(z)_i} H_{\epsilon\epsilon}} = \frac{\sigma^2 w_i^2 (\sigma^{-2}_{z|ch(z)})^2}{(\alpha + w_i^2(\sigma^{-2}_{z|ch(z)}))(\sigma^2 \sigma^{-2}_{z|ch(z)} - 1)}$$

Posterior correlation analysis (3)

Inequality: all terms cancel with very simple result:

$$\rho^2_{pa(z)_i,z} > \rho^2_{pa(z)_i,\epsilon}$$

$$\Leftrightarrow$$

$$\sigma^2_{z|pa(z)} < \sigma^2_{z|ch(z)}$$

i.e., DNCP leads to more efficient inference
when latent variable 'z' is more strongly bound to its parents
then to its children

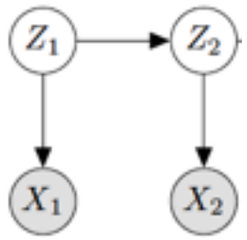
Posterior correlation analysis (4)

	$\rho^2_{y_i,z}$ (CP)	$\rho^2_{y_i,e}$ (DNCP) $\sigma^2_{z\|pa(z)}$
$\lim \sigma_{z\|pa(z)} \to 0$	1	0
$\lim \sigma_{z\|pa(z)} \to \infty$	0	$\frac{\sigma^{-2}_{z\|ch(z)} w_i^2}{\sigma^{-2}_{z\|ch(z)} w_i^2 + \alpha}$
$\lim \sigma_{z\|ch(z)} \to 0$	0	1
$\lim \sigma_{z\|ch(z)} \to \infty$	$\frac{w_i^2}{w_i^2 - \alpha\sigma^2_{z\|pa(z)}}$	0

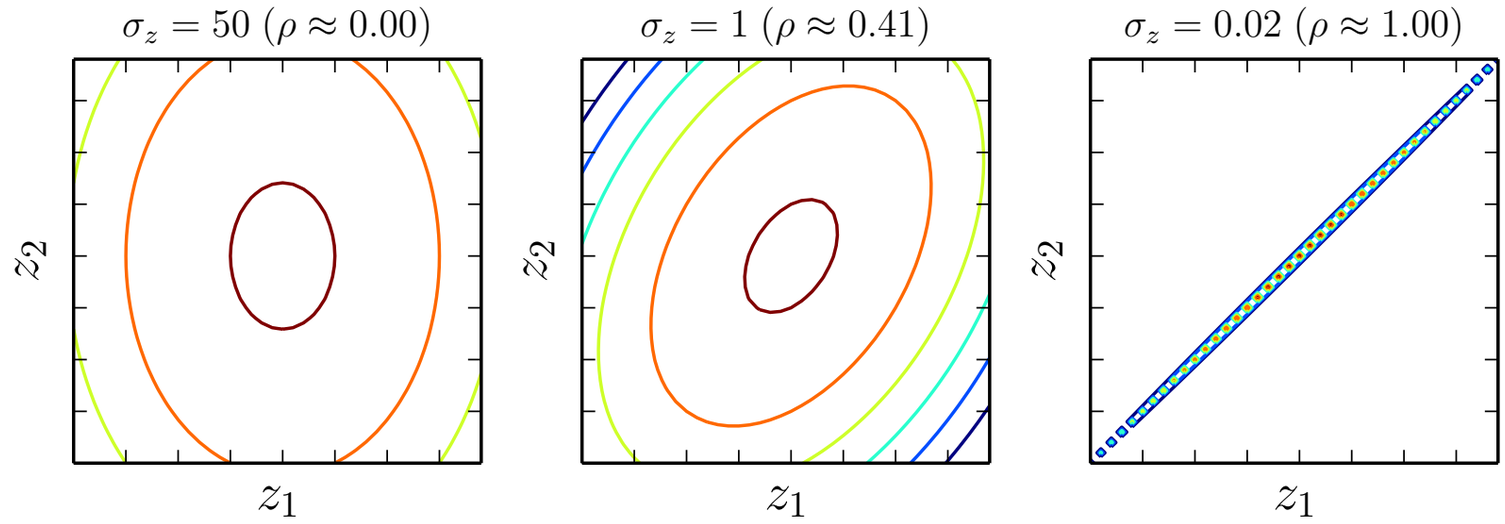
„beauty-and-beast pair“

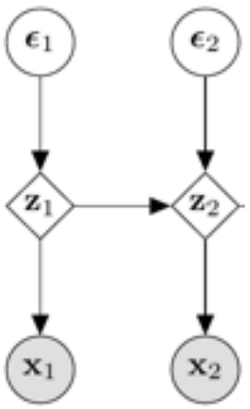
© Disney

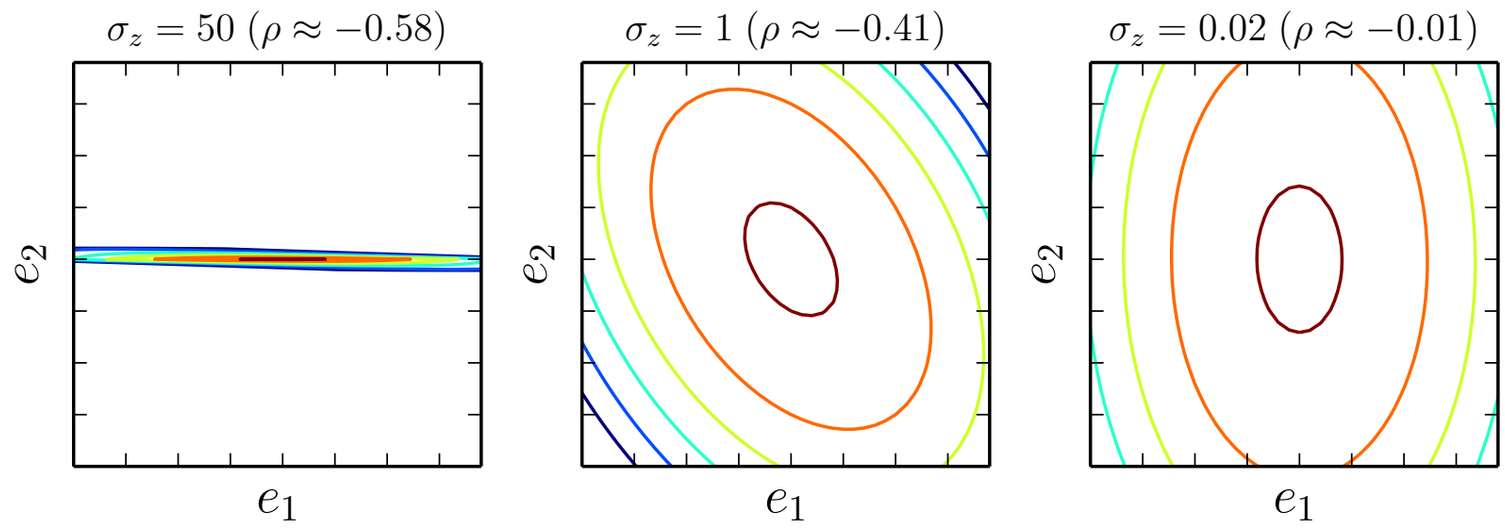
2D linear-Gaussian posterior example



$\sigma_z = 50$ $\sigma_z = 1$ $\sigma_z = 0.02$







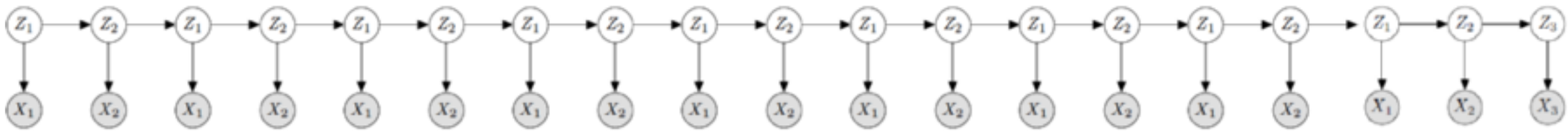
Robust HMC Sampler

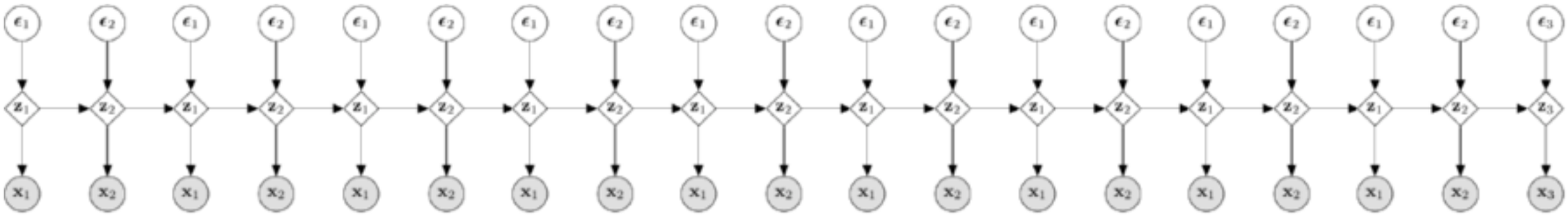
- Proposal distribution is mix between two proposal distributions:

$$Q(\mathbf{z}'|\mathbf{z}) = \rho \cdot Q_{CP}(\mathbf{z}'|\mathbf{z}) + (1-\rho) \cdot Q_{DNCP}(\mathbf{z}'|\mathbf{z})$$

- Where we use ρ = 0.5 in experiments

Experiment: HMC sampling with Dynamic Bayesian network (DBN)

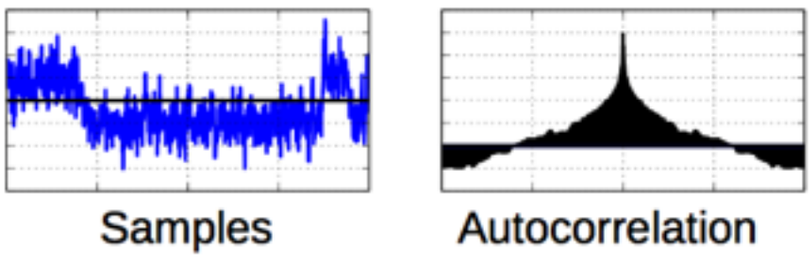




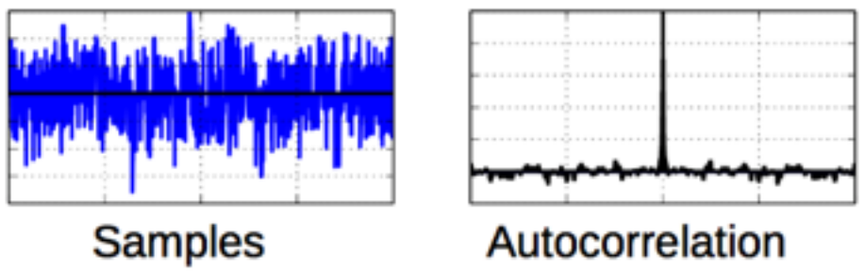
$$p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}; \tanh(W\mathbf{z}_{t-1}), \sigma_z^2 I)$$
$$p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}; \tanh(W\mathbf{z}_t), \sigma_x^2 I)$$

Autocorrelation results on DBN

$\log \sigma_z$	CP	DNCP	ROBUST
-5	2	305	640
-4.5	26	348	498
-4	10	570	686
-3.5	225	417	624
-3	386	569	596
-2.5	542	608	900
-2	406	972	935
-1.5	672	1078	918
-1	1460	1600	1082



CP (Terribly inefficient)

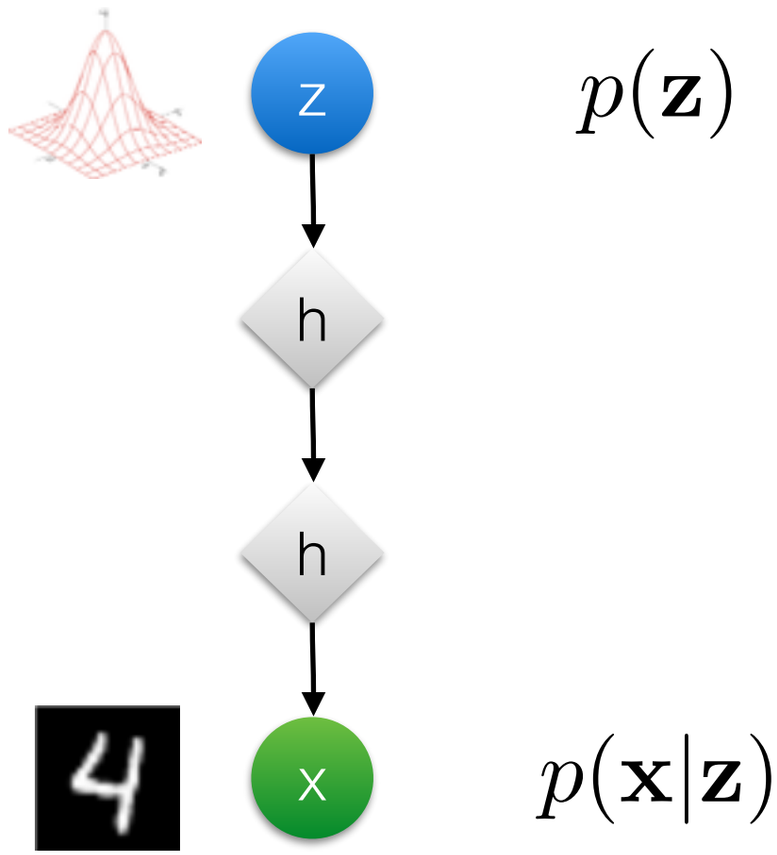


DNCP (Very efficient)

Part 2:

The Variational Auto-Encoder

Deep latent variable model



$p(\mathbf{z})$

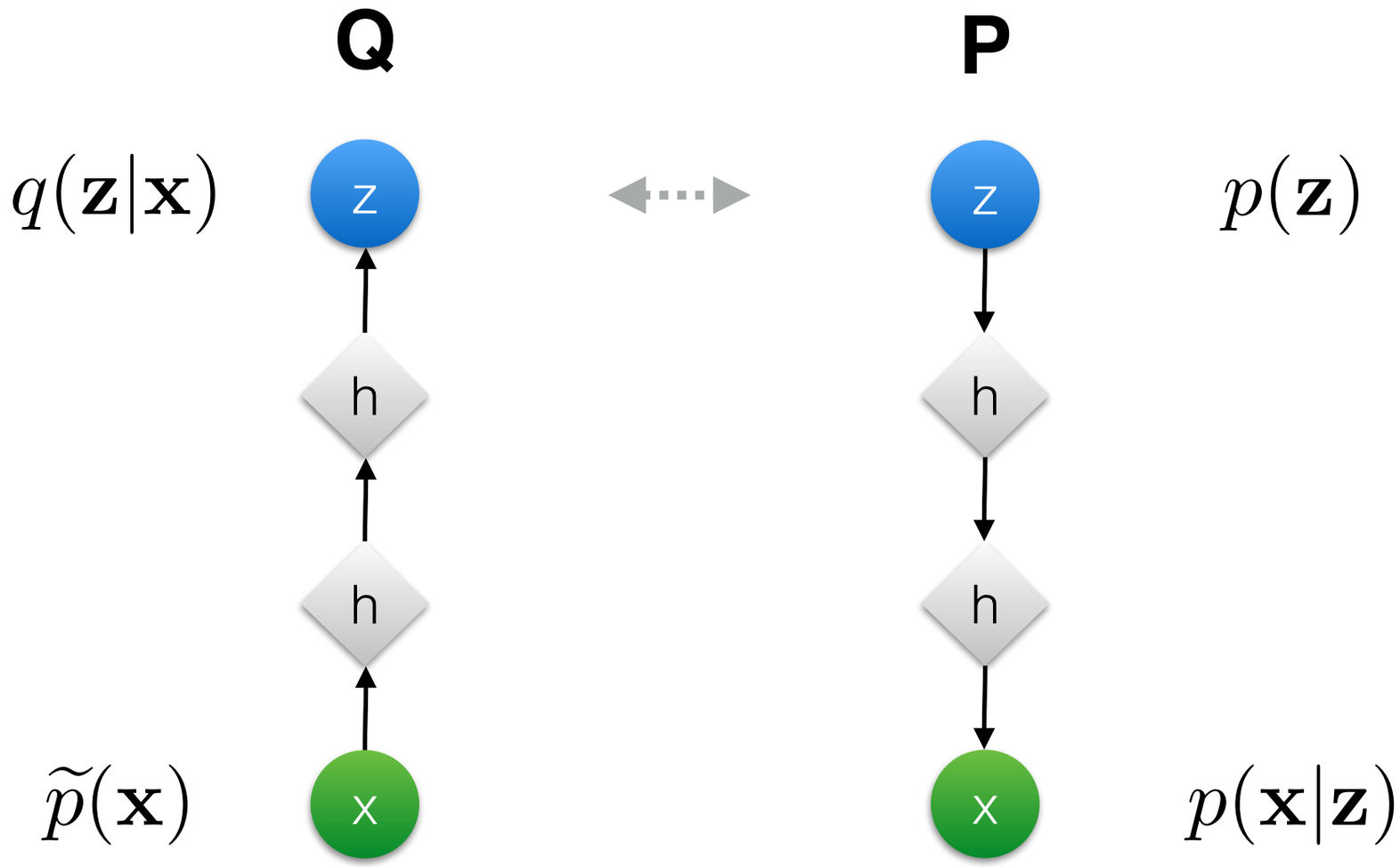
$p(\mathbf{x}|\mathbf{z})$

Monte Carlo EM

Monte Carlo EM - End result

MAP inference with L-BFGS

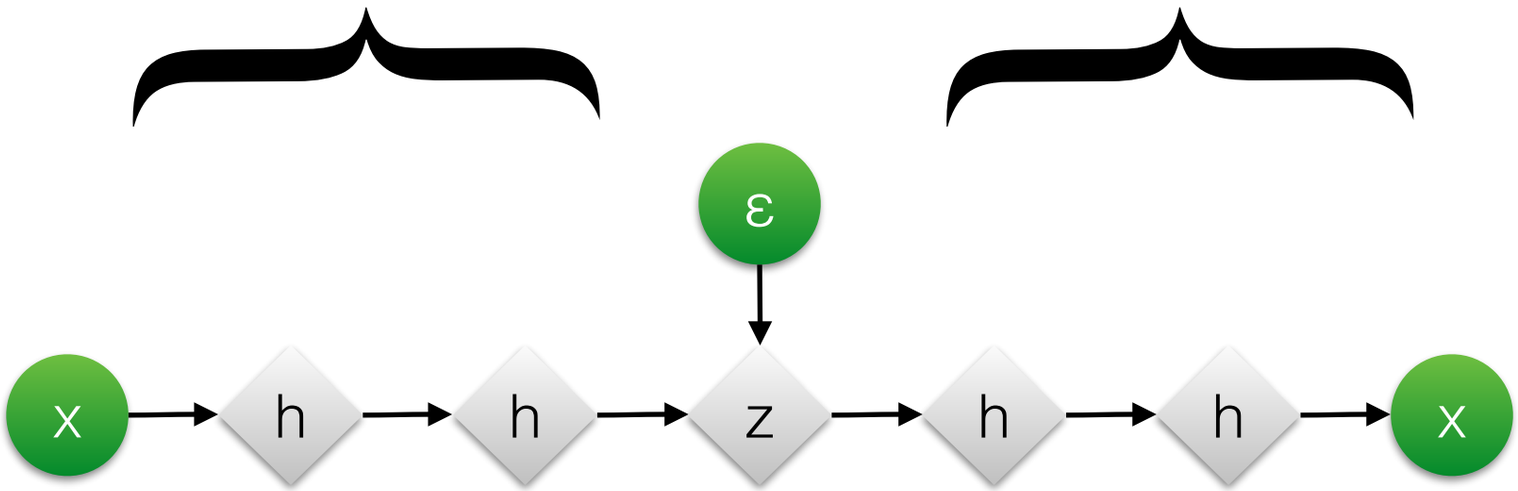
The Variational Auto-Encoder (ICLR'14)



$$L = -D_{KL}(\widetilde{p}(\mathbf{x})q(\mathbf{z}|\mathbf{x})||p(\mathbf{x},\mathbf{z}))$$
$$\leq \log p(\mathbf{x})$$

Why is this an auto-encoder?

Reparameterized Q(z|x) **P(x|z)**



$$L = D_{KL}(\widetilde{p}(\mathbf{x})q(\mathbf{z}|\mathbf{x})||p(\mathbf{x},\mathbf{z}))$$
$$= \mathbb{E}_{\widetilde{p}(\mathbf{x})}\left[\mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]\right]$$

Reconstruction error

Regularization terms (dictated by the bound)

Variational Auto-Encoder trained on MNIST (2D Latent space)

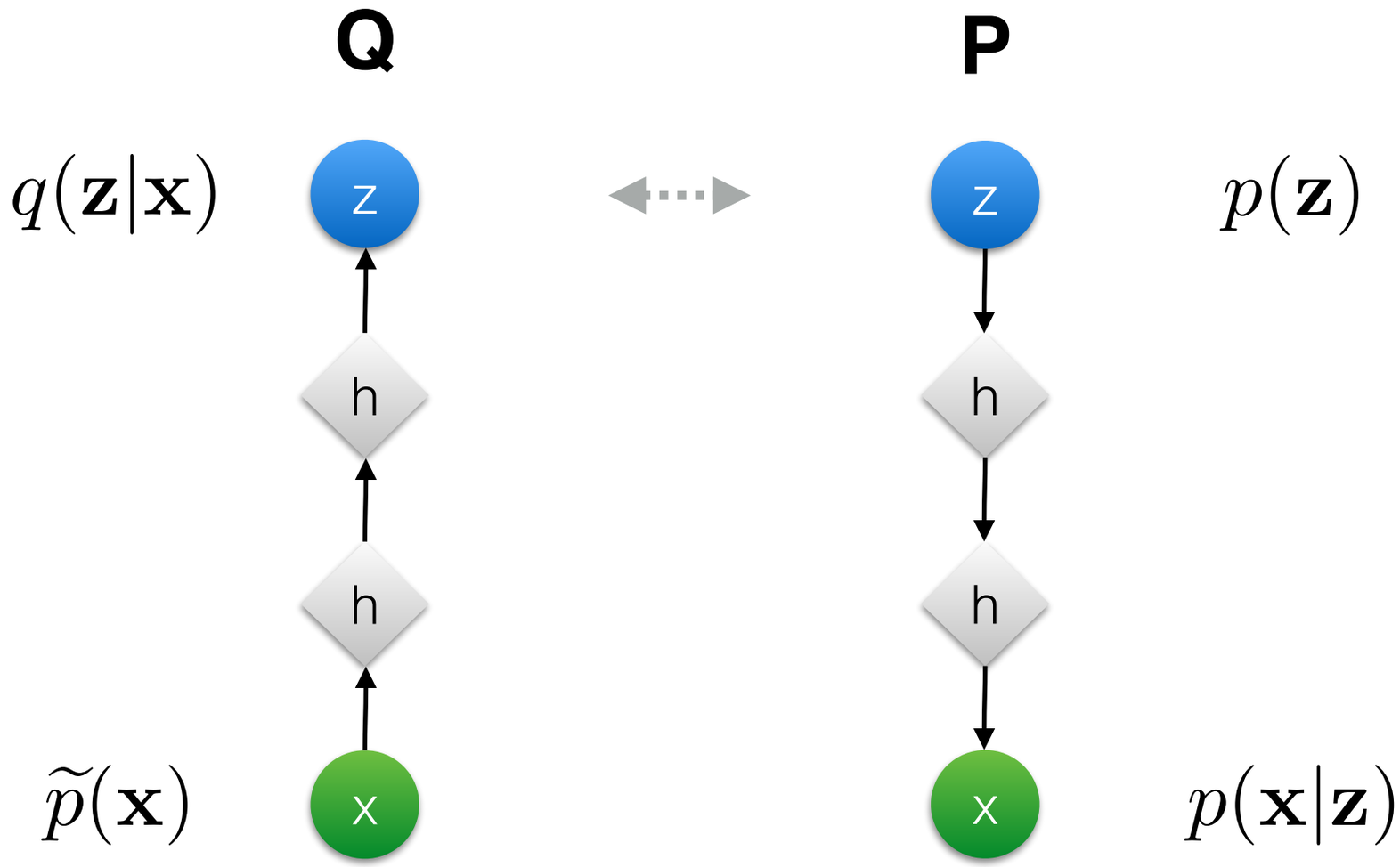
3D latent space

Labeled faces in the wild

Semi-supervised learning with deep generative models (“Model 2.5”)

Approach 1

DLGM / VAE as feature extractor for semi-supervised classifier



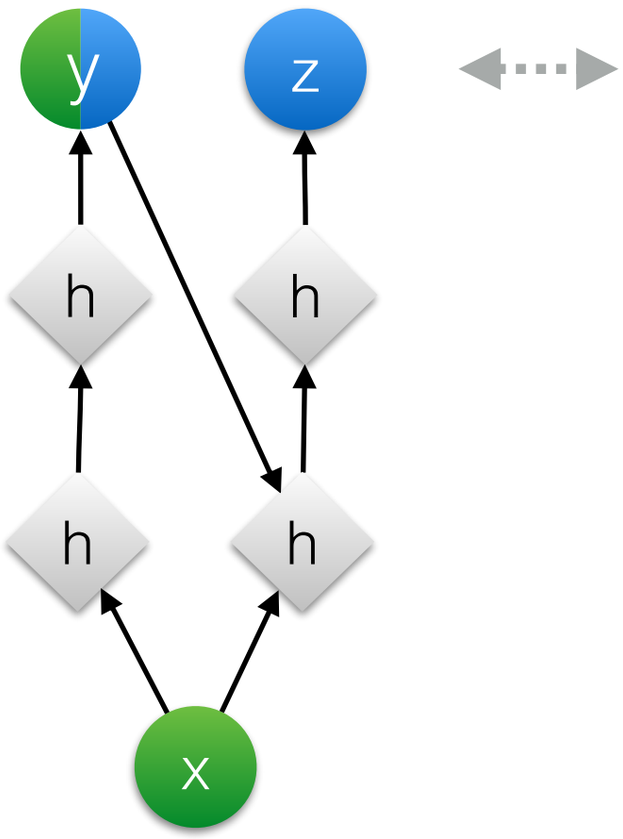
$$L = -D_{KL}(\widetilde{p}(\mathbf{x})q(\mathbf{z}|\mathbf{x})||p(\mathbf{x},\mathbf{z}))$$

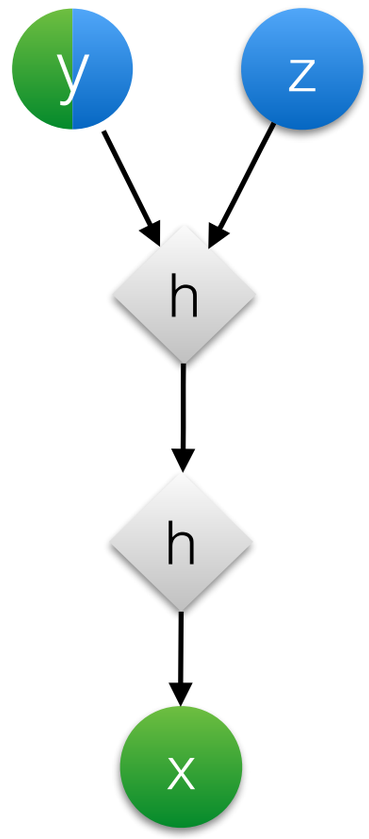
Approach 2

DLGM / VAE as regularizer of neural net classifier

Q **P**

$q(\mathbf{y}|\mathbf{x})\,q(\mathbf{z}|\mathbf{x},\mathbf{y})$





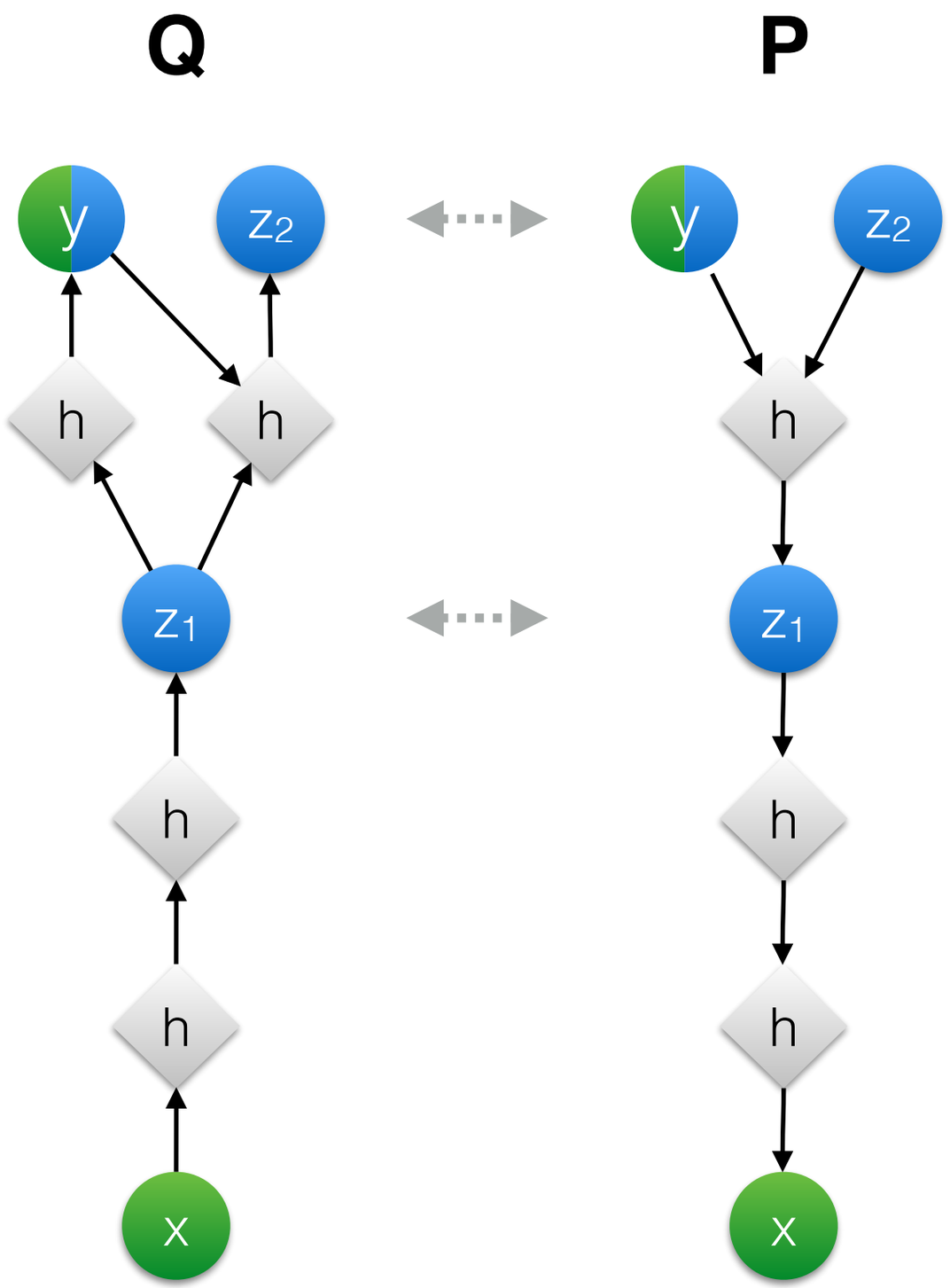
$p(\mathbf{y})\,p(\mathbf{z})$

$\widetilde{p}(\mathbf{x})$

$p(\mathbf{x}|\mathbf{y},\mathbf{z})$

$$
\begin{aligned}
L &= \mathbb{E}_{\widetilde{p}_l(\mathbf{x},\mathbf{y})}\left[\log q(\mathbf{y}|\mathbf{x})\right] + \beta L^{reg} \\
L^{reg} &= -D_{KL}\left(\widetilde{p}_l(\mathbf{x},\mathbf{y})q(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{x},\mathbf{y},\mathbf{z})\right) \\
&\quad -D_{KL}\left(\widetilde{p}_u(\mathbf{x})q(\mathbf{y}|\mathbf{x})q(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{x},\mathbf{y},\mathbf{z})\right)
\end{aligned}
$$

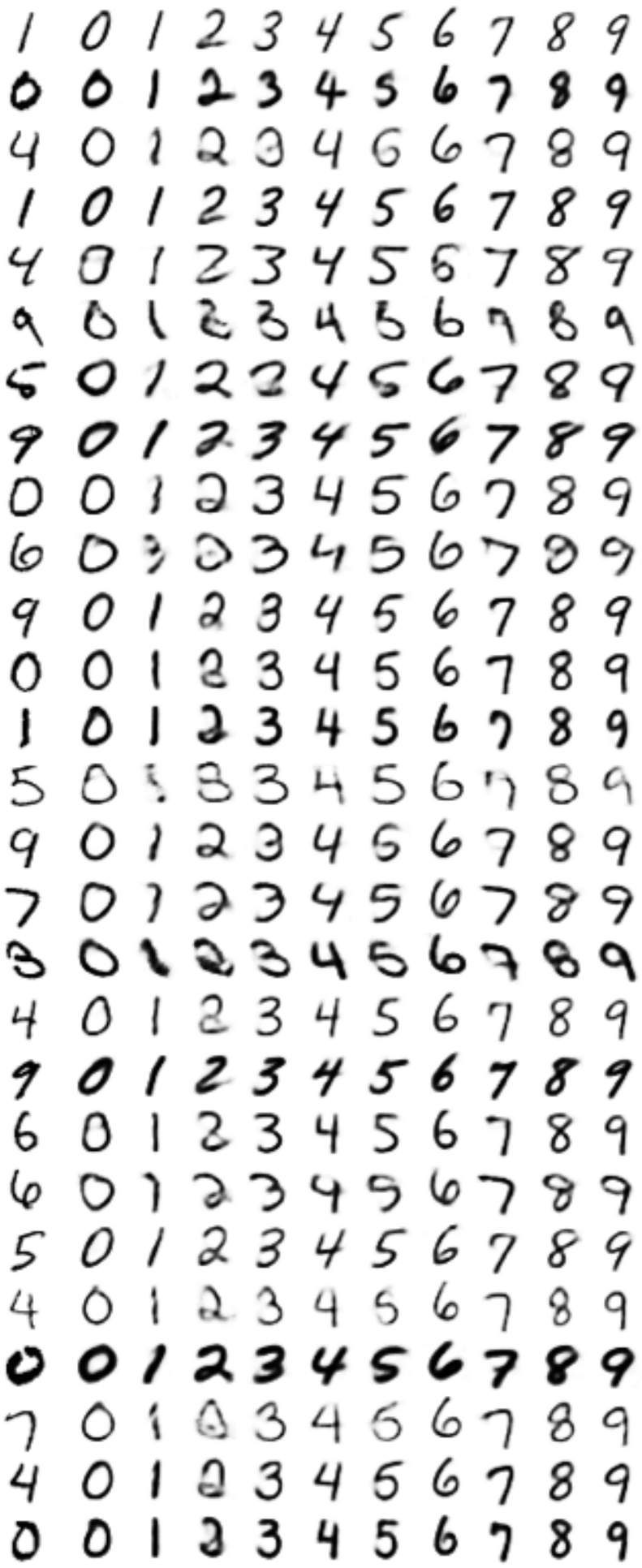
Approach 3



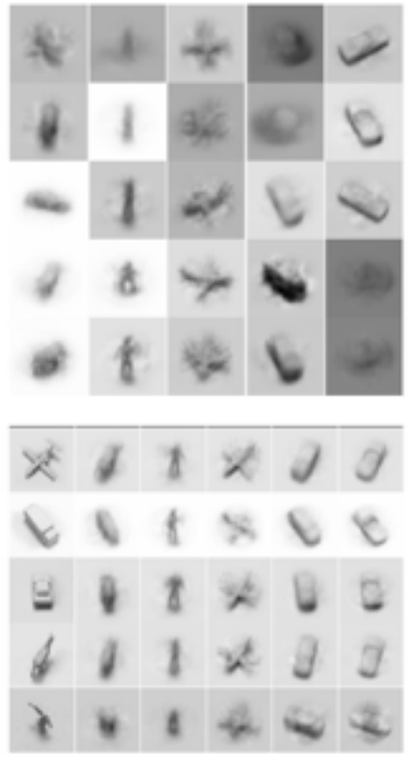
Results

Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

N_{labeled}	NN	CNN	TSVM	EmbNN	CAE	MTC	M1	M2	M3
100	25.81	22.98	16.81	16.86	13.47	12.03*	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.54** ($\pm$ 0.03)
600	11.44	7.68	6.16	5.97	6.3	5.13*	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.85** ($\pm$ 0.1)
1000	10.7	6.45	5.38	5.73	4.77	3.64*	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.76** ($\pm$ 0.30)
3000	6.04	3.35	3.45	3.59	3.22	2.57*	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.63** ($\pm$ 0.24)



Analogic reasoning



(will improve)

Next steps

- SSL on larger images
  - SVHN, CIFAR-10 (this NIPS paper)
  - Imagenet (Future papers)
- SSL on video's
  - Youtube
- Applications of anagogic reasoning

Thanks!

RMSProp

Initialization:

$\mathbf{m} \leftarrow 0$

$\mathbf{v} \leftarrow 0$

Update_parameters$(\mathbf{w}, \mathbf{g})$:

$\mathbf{m} \leftarrow \beta_1 \cdot \mathbf{g} + (1 - \beta_1) \cdot \mathbf{m}$

$\mathbf{v} \leftarrow \beta_2 \cdot \mathbf{g}^2 + (1 - \beta_2) \cdot \mathbf{v}$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \mathbf{m} / \sqrt{\mathbf{v}}$

Estimation based on exp. mov. avg.

Assume: g is a random variable
Goal: from exponential moving averages of i.i.d. draws g_i, estimate moments E[g] and $E[g^2]$

$$\mathbf{m}_t = \beta_1 \cdot \mathbf{g}_t + (1 - \beta_1) \cdot \mathbf{m}_{t-1}$$

$$= \beta_1 \sum_{i=1}^{t} (1 - \beta_1)^{t-i} \cdot \mathbf{g}_i \qquad \text{(As function of past samples)}$$

$$\mathbb{E}\left[\mathbf{m}_t\right] = \mathbb{E}\left[\beta_1 \sum_{i=1}^{t} (1 - \beta_1)^{t-i} \cdot \mathbf{g}_i\right] \qquad \text{(Taking expectations)}$$

$$= \mathbb{E}\left[\mathbf{g}\right] \cdot \beta_1 \sum_{i=1}^{t} (1 - \beta_1)^{t-i} \qquad \text{(Because i.i.d.)}$$

$$= \mathbb{E}\left[\mathbf{g}\right] \cdot (1 - (1 - \beta_1)^t) \qquad \text{(Further simplification)}$$

Fixed RMSProp (AdaM)

Initialization:

$\mathbf{m} \leftarrow 0$

$\mathbf{v} \leftarrow 0$

$\text{Update\_parameters}(\mathbf{w}, \mathbf{g}) :$

$\mathbf{m} \leftarrow \beta_1 \cdot \mathbf{g} + (1 - \beta_1) \cdot \mathbf{m}$

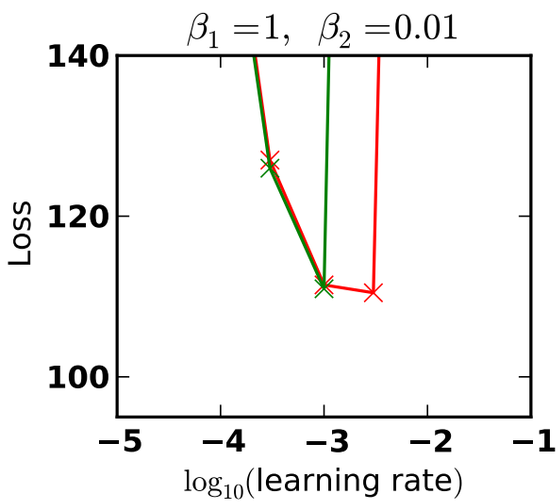
$\mathbf{v} \leftarrow \beta_2 \cdot \mathbf{g}^2 + (1 - \beta_2) \cdot \mathbf{v}$

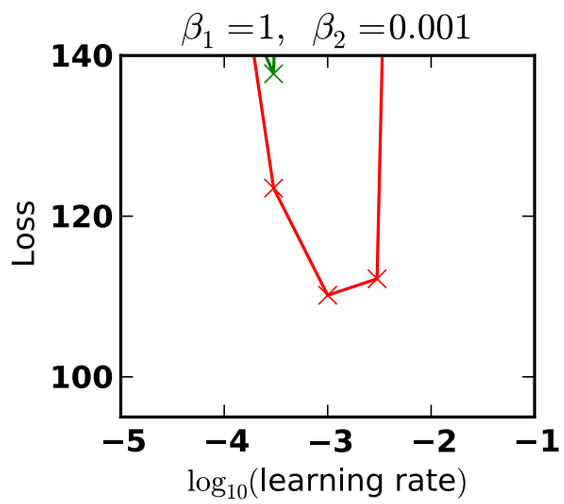
$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \gamma_t \cdot \mathbf{m}/\sqrt{\mathbf{v}}$

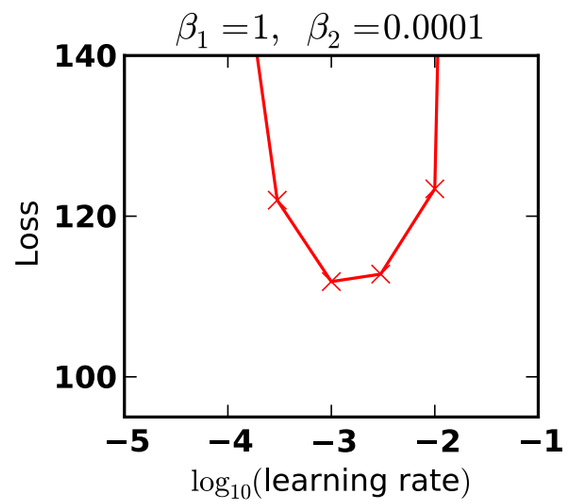
where $\gamma_t = \sqrt{1 - (1 - \beta_2)^t}/(1 - (1 - \beta_1)^t)$

VAE, 10 epochs

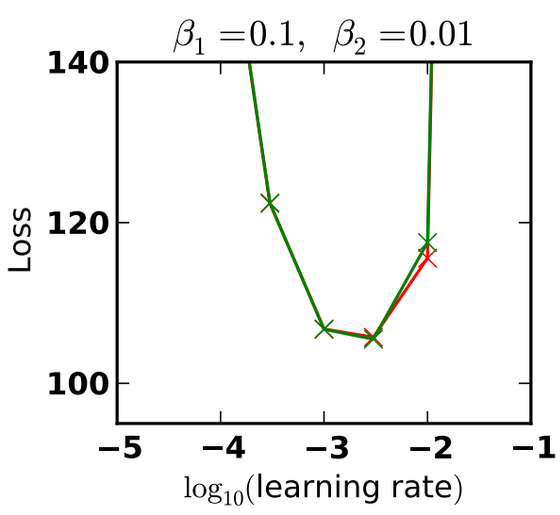
$\beta_1 = 1$

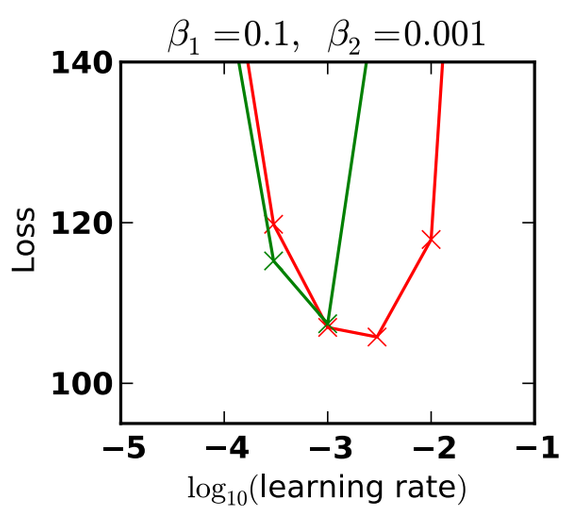


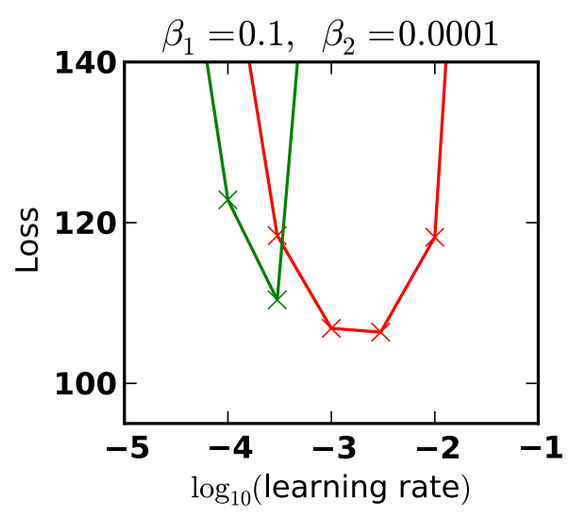




$\beta_1 = 0.1$





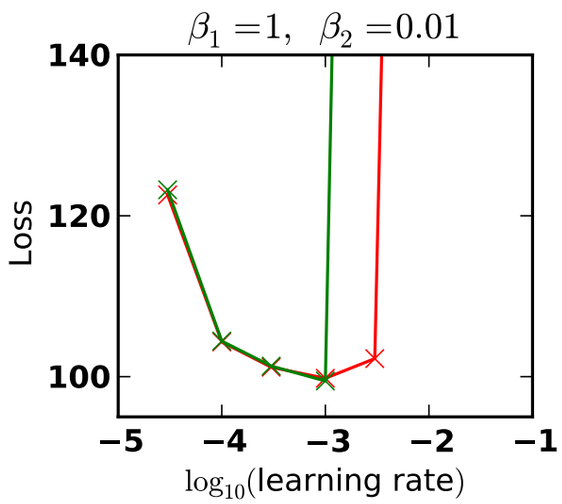


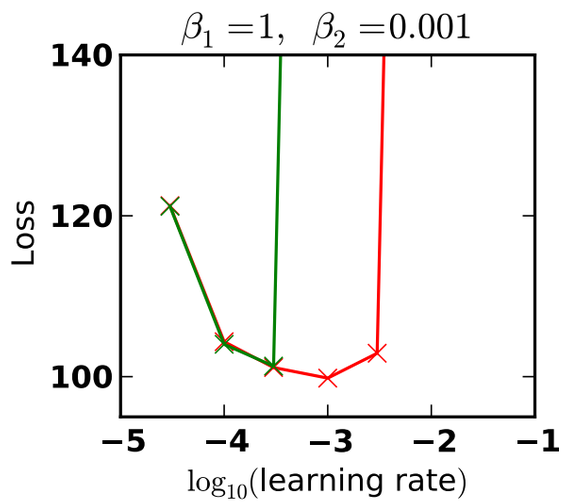
$\beta_2 = 0.01$ $\beta_2 = 0.001$ $\beta_2 = 0.0001$

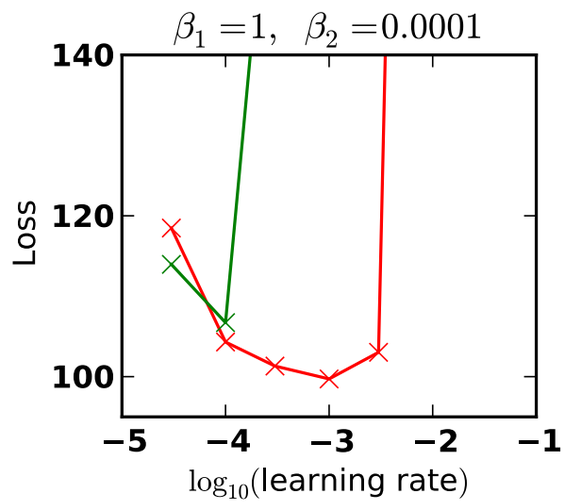
More like AdaGrad (infinite memory)

VAE, 100 epochs

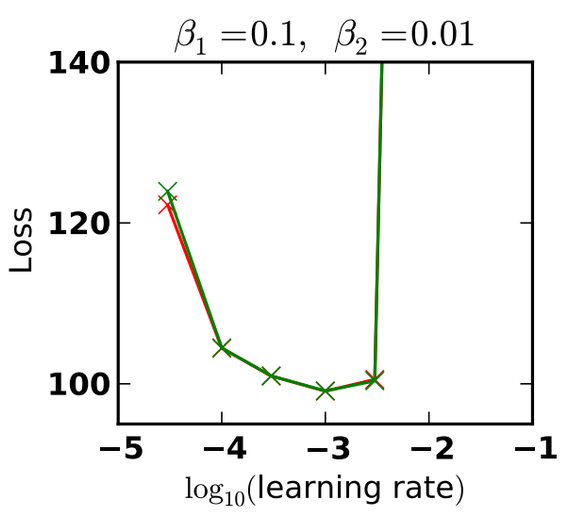
$\beta_1 = 1$

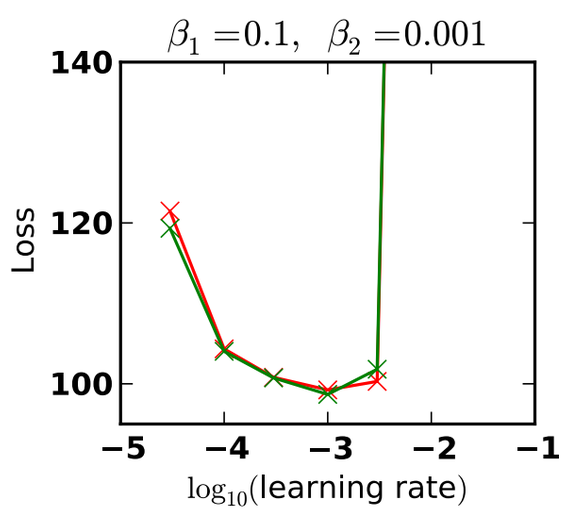


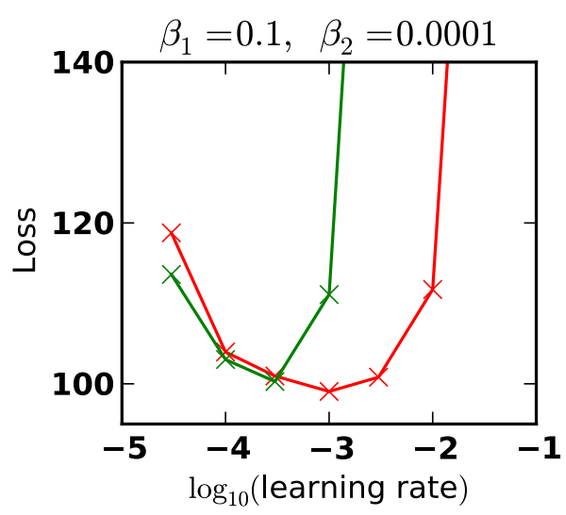




$\beta_1 = 0.1$







$\beta_2 = 0.01$ $\beta_2 = 0.001$ $\beta_2 = 0.0001$

More like AdaGrad (infinite memory)

Now possible:
Interpolation between Adagrad and RMSProp

$$\text{Initialization:}$$
$$\mathbf{m} \leftarrow 0$$
$$\mathbf{v} \leftarrow 0$$

$$\text{Update\_parameters}(\mathbf{w}, \mathbf{g}) :$$
$$\mathbf{m} \leftarrow \beta_1 \cdot \mathbf{g} + (1 - \beta_1) \cdot \mathbf{m}$$
$$\mathbf{v} \leftarrow \beta_2 \cdot \mathbf{g}^2 + (1 - \beta_2) \cdot \mathbf{v}$$
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \gamma_t \cdot \mathbf{m} / \sqrt{t \cdot \mathbf{v}}$$

$$\text{where } \gamma_t = \sqrt{1 - (1 - \beta_2)^t} / (1 - (1 - \beta_1)^t)$$

This algorithm AdaGrad when $\beta_1 = 1$ and $\beta_2 = \varepsilon$ (infinitesimal)