One-shot generative modelling

Сергей Бартунов

ФКН НИУ ВШЭ

16 сентября 2016 г.

Generative models

Any distribution over object of interest $\mathbf{x}$ is a *generative model*:

$$p(\mathbf{x}|\theta).$$

Generative models

Any distribution over object of interest $\mathbf{x}$ is a *generative model*:

$$p(\mathbf{x}|\theta).$$

Objects can be generated by independent sampling from the distribution:

$$\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots \sim p(\mathbf{x}|\boldsymbol{\theta}).$$

Generative models

Any distribution over object of interest $\mathbf{x}$ is a *generative model*:

$$p(\mathbf{x}|\theta).$$

Objects can be generated by independent sampling from the distribution:

$$\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots \sim p(\mathbf{x}|\boldsymbol{\theta}).$$

Example: bag of words language model

- $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ is a text of length N
- $p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(x_i|\boldsymbol{\theta})$ – bag of words model
- $\boldsymbol{\theta}$ is just word frequencies

Latent-variable models

Assumption: object $\mathbf{x}$ can be summarized or explained by latent variable $\mathbf{z}$:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{z}|\boldsymbol{\theta}) p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) d\mathbf{z}.$$

Latent-variable models

Assumption: object $\mathbf{x}$ can be summarized or explained by latent variable $\mathbf{z}$:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{z}|\boldsymbol{\theta})p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})d\mathbf{z}.$$

Example: Latent Dirichlet Allocation

- $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ is a text of length N
- $\mathbf{z} = \{\gamma, t_1, t_2, \dots, t_N\}$ – latent variable
  - $p(\gamma|\theta) = \text{Dirichlet}(\alpha_1, \dots, \alpha_K)$ – document profile
  - $p(t_i = k|\gamma) = \gamma_k$ – word-topic assignment
- $p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{i=1}^{N} p(x_i|\phi_{t_i})$ – word x_i is explained by the assigned topic t_i
- $\boldsymbol{\theta} = \{\alpha, \phi\}$ – model parameters

Learning in latent-variable models

Given training data $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N$ find the best parameters $\boldsymbol{\theta}$:

$$\sum_{i=1}^{N} \log p(\mathbf{x}_i | \boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}.$$

Learning in latent-variable models

Given training data $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N$ find the best parameters $\boldsymbol{\theta}$:

$$\sum_{i=1}^{N} \log p(\mathbf{x}_i | \boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}} .$$

Exact learning is hard, we resort to *variational learning*.

- For each i introduce $q(\mathbf{z}_i | \lambda_i) \approx p(\mathbf{z}_i | \mathbf{x}_i, \boldsymbol{\theta})$
- Using these approximations derive a *variational lower bound*:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i|\lambda_i)} \left[\log p(\mathbf{x}_i, \mathbf{z}_i | \boldsymbol{\theta}) - \log q(\mathbf{z}_i | \lambda_i)\right]$$

- Optimize the lower bound with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^{N}$.

Deep latent-variable models

Design decision: $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ is modelled by a neural network.

Deep latent-variable models

Design decision: $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ is modelled by a neural network.

Example: neural language model

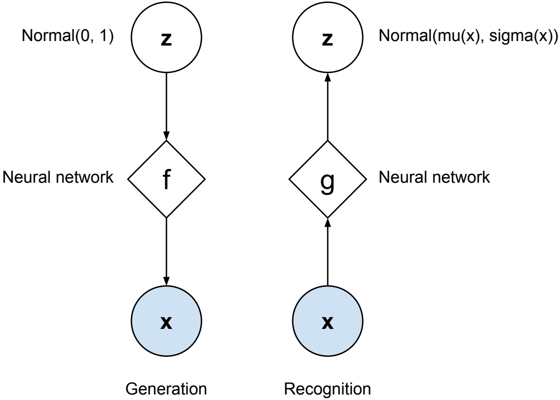
- $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ is a text of length N
- $p(\mathbf{z}) = \mathcal{N}(0, 1)$ – d-dimensional std normal
- Hidden-state dynamics:
  - $h_1 = \mathit{init}(\mathbf{z})$
  - $h_i = \mathit{state}(h_{i-1}, x_{i-1}, \mathbf{z}), \quad i > 1$
- $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(x_i|x_{<i}, \mathbf{z}, \boldsymbol{\theta})$
- $p(x_i = w|x_{<i}, \mathbf{z}, \boldsymbol{\theta}) = \frac{\exp(f(w,h_i))}{\sum_v \exp(f(v,h_i))}$
- $\boldsymbol{\theta}$ controls *init*, *state* and f.

Variational autoencoders

Design decision: let not only the generative model, but also the approximate posterior $q(\mathbf{z}_i|\mathbf{x}, \phi) = \mathcal{N}(\mu(\mathbf{x}_i, \phi), \sigma(\mathbf{x}_i, \phi))$ be modelled by a neural network.

Variational autoencoders

Design decision: let not only the generative model, but also the approximate posterior $q(\mathbf{z}_i|\mathbf{x}, \phi) = \mathcal{N}(\mu(\mathbf{x}_i, \phi), \sigma(\mathbf{x}_i, \phi))$ be modelled by a neural network.



Learning in variational autoencoders

Variational lower bound:

$$\mathcal{L}(\boldsymbol{\theta}, \phi) = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i,\phi)} \left[\log p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\theta}) - \log q(\mathbf{z}_i|\mathbf{x}_i, \phi) \right].$$

Learning in variational autoencoders

Variational lower bound:

$$\mathcal{L}(\boldsymbol{\theta}, \phi) = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i,\phi)} \left[\log p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\theta}) - \log q(\mathbf{z}_i|\mathbf{x}_i, \phi)\right].$$

Reparametrization trick:

$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(0,1) \Rightarrow g(\boldsymbol{\epsilon}_i, \phi) = \sigma(\mathbf{x}_i, \phi)\boldsymbol{\epsilon} + \mu(\mathbf{x}_i, \phi) \sim \mathcal{N}(\mu(\mathbf{x}_i, \phi), \sigma(\mathbf{x}_i, \phi)).$$

Learning in variational autoencoders

Variational lower bound:

$$\mathcal{L}(\boldsymbol{\theta}, \phi) = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i,\phi)} \left[\log p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\theta}) - \log q(\mathbf{z}_i|\mathbf{x}_i, \phi)\right].$$

Reparametrization trick:

$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(0,1) \Rightarrow g(\boldsymbol{\epsilon}_i, \boldsymbol{\phi}) = \sigma(\mathbf{x}_i, \boldsymbol{\phi})\boldsymbol{\epsilon} + \mu(\mathbf{x}_i, \boldsymbol{\phi}) \sim \mathcal{N}(\mu(\mathbf{x}_i, \boldsymbol{\phi}), \sigma(\mathbf{x}_i, \boldsymbol{\phi})).$$

Monte-carlo estimate:

$$\begin{aligned}\hat{\mathcal{L}}(\boldsymbol{\theta}, \phi) &= N\left[\log p(\mathbf{x}_i, g(\boldsymbol{\epsilon}_i, \phi)|\boldsymbol{\theta}) - \log q(g(\boldsymbol{\epsilon}_i, \phi)|\mathbf{x}_i, \phi)\right], \\ i &\sim \mathrm{Uniform}(1, \ldots, N), \\ \boldsymbol{\epsilon} &\sim \mathcal{N}(0,1).\end{aligned}$$

Modelling exchangeable data

Can we benefit from relaxation of i.i.d. assumption?

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \neq \prod_{i=1}^{N} p(\mathbf{x}_i).$$

Modelling exchangeable data

Can we benefit from relaxation of i.i.d. assumption?

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \neq \prod_{i=1}^{N} p(\mathbf{x}_i).$$

- By de Finetti's theorem, there exists a *global* latent variable $\boldsymbol{\alpha}$ making data conditionally independent:

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \int p(\boldsymbol{\alpha}) \prod_{i=1}^{N} p(\mathbf{x}_i | \boldsymbol{\alpha}) d\boldsymbol{\alpha}.$$

Modelling exchangeable data

Can we benefit from relaxation of i.i.d. assumption?

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \neq \prod_{i=1}^{N} p(\mathbf{x}_i).$$

- By de Finetti's theorem, there exists a *global* latent variable $\boldsymbol{\alpha}$ making data conditionally independent:

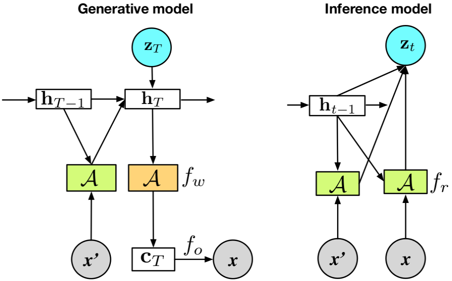
$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \int p(\boldsymbol{\alpha}) \prod_{i=1}^{N} p(\mathbf{x}_i | \boldsymbol{\alpha}) d\boldsymbol{\alpha}.$$

- We may consider the following conditional dependence:

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \prod_{i=1}^{N} p(\mathbf{x}_i | \mathbf{x}_{<i}).$$

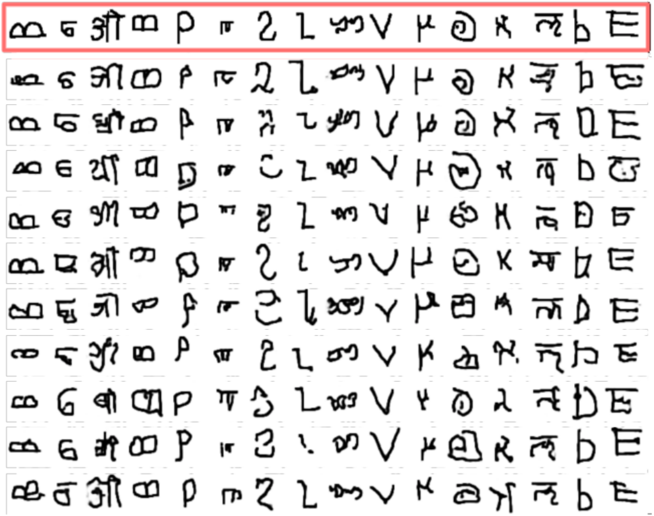
Sequential generative model (Rezende et al, 2016)

A modification of the DRAW (Gregor et al, 2015), can explicitly condition on another image $\mathbf{x}'$.



- $p(\mathbf{z}) = \prod_{t=1}^{T} p(z_t)$
- $z_t \sim \mathcal{N}(z_t|0, 1)$
- $v_t = f_v(h_{t-1}, \mathbf{x}'; \theta_v)$
- $h_t = f_h(h_{t-1}, z_t, v_t; \theta_h)$
- $c_t = f_c(c_{t-1}, h_t; \theta_c)$
- $\mathbf{x} \sim p(\mathbf{x}|f_o(c_t; \theta_o))$

One-shot generalization



Neural statistical (Edwards & Storkey, 2016)

Latent-variable model implementing de Finetti's theorem:

$$p(\mathbf{x}_1, \ldots \mathbf{x}_N | \boldsymbol{\theta}) = \int p(\mathbf{c}|\boldsymbol{\theta}) \prod_{i=1}^{N} \int \left(p(\mathbf{z}_i|\mathbf{c};\boldsymbol{\theta}) p(\mathbf{x}_i|\mathbf{z}_i,\mathbf{c};\boldsymbol{\theta}) \right) d\mathbf{z}_i d\mathbf{c}$$

Neural statistical (Edwards & Storkey, 2016)

Latent-variable model implementing de Finetti's theorem:

$$p(\mathbf{x}_1, \ldots \mathbf{x}_N | \boldsymbol{\theta}) = \int p(\mathbf{c}|\boldsymbol{\theta}) \prod_{i=1}^{N} \int \left(p(\mathbf{z}_i|\mathbf{c}; \boldsymbol{\theta}) p(\mathbf{x}_i|\mathbf{z}_i, \mathbf{c}; \boldsymbol{\theta}) \right) d\mathbf{z}_i d\mathbf{c}$$

Recognition model:

$$q(\mathbf{z}_1, \ldots, \mathbf{z}_N, \mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) = q(\mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) \prod_{i=1}^{N} q(\mathbf{z}_i|\mathbf{x}_i, \mathbf{c}; \phi)$$

Neural statistical (Edwards & Storkey, 2016)

Latent-variable model implementing de Finetti's theorem:

$$p(\mathbf{x}_1, \ldots \mathbf{x}_N|\boldsymbol{\theta}) = \int p(\mathbf{c}|\boldsymbol{\theta}) \prod_{i=1}^{N} \int \left(p(\mathbf{z}_i|\mathbf{c};\boldsymbol{\theta})p(\mathbf{x}_i|\mathbf{z}_i,\mathbf{c};\boldsymbol{\theta})\right) d\mathbf{z}_i d\mathbf{c}$$

Recognition model:

$$q(\mathbf{z}_1, \ldots, \mathbf{z}_N, \mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) = q(\mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) \prod_{i=1}^{N} q(\mathbf{z}_i|\mathbf{x}_i, \mathbf{c}; \phi)$$

- Context variable $\mathbf{c}$ represents *global* statistics of the dataset (how the character looks like).
- Local variable $\mathbf{z}$ controls variations applied to the global image of a character.

Training protocol

- Assume there is a distribution $p(D)$ on *datasets* $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.

Training protocol

- Assume there is a distribution $p(D)$ on *datasets* $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$.
- Maximize the expected likelihood of a dataset:

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Training protocol

- Assume there is a distribution $p(D)$ on *datasets* $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$.
- Maximize the expected likelihood of a dataset:

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

- Since it's intractable, use variational inference

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \geq$$
$$\mathbb{E}_{D \sim p(D)} \Big[\log p(\boldsymbol{\alpha}|\boldsymbol{\theta}) - \log q(\boldsymbol{\alpha}|D; \phi) +$$
$$\sum_{i=1}^{N} \log p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\alpha}; \boldsymbol{\theta}) - \log q(\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\alpha}; \phi) \Big]$$

Training protocol

- Assume there is a distribution $p(D)$ on *datasets* $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.
- Maximize the expected likelihood of a dataset:

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

- Since it's intractable, use variational inference

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \geq$$
$$\mathbb{E}_{D \sim p(D)} \Big[\log p(\boldsymbol{\alpha}|\boldsymbol{\theta}) - \log q(\boldsymbol{\alpha}|D; \phi) +$$
$$\sum_{i=1}^{N} \log p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\alpha}; \boldsymbol{\theta}) - \log q(\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\alpha}; \phi) \Big]$$

- $p(D)$ is constrained in a way that all objects belong to the same class

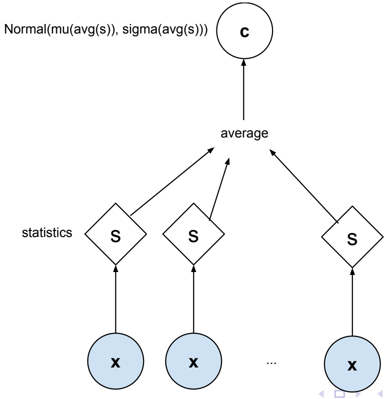
One-shot generalization

$$p(\mathbf{x}|\mathbf{x}_1,\ldots,\mathbf{x}_N;\boldsymbol{\theta}) = \int p(\mathbf{c}|\mathbf{x}_1,\ldots,\mathbf{x}_N;\boldsymbol{\theta})p(\mathbf{x}|\mathbf{c};\boldsymbol{\theta})d\mathbf{c}$$
$$\approx \int q(\mathbf{c}|\mathbf{x}_1,\ldots,\mathbf{x}_N;\phi)p(\mathbf{x}|\mathbf{c};\boldsymbol{\theta})d\mathbf{c}$$

One-shot generalization

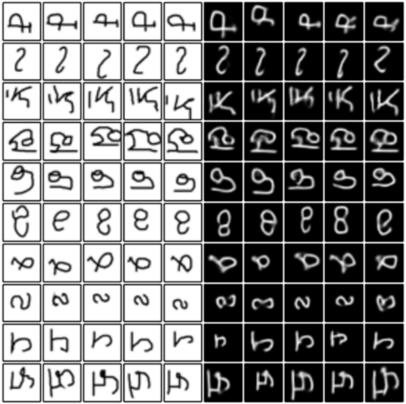
$$p(\mathbf{x}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \boldsymbol{\theta}) = \int p(\mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \boldsymbol{\theta}) p(\mathbf{x}|\mathbf{c}; \boldsymbol{\theta}) d\mathbf{c}$$

$$\approx \int q(\mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) p(\mathbf{x}|\mathbf{c}; \boldsymbol{\theta}) d\mathbf{c}$$



One-shot generalization

$$p(\mathbf{x}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \boldsymbol{\theta}) = \int p(\mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \boldsymbol{\theta}) p(\mathbf{x}|\mathbf{c}; \boldsymbol{\theta}) d\mathbf{c}$$
$$\approx \int q(\mathbf{c}|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) p(\mathbf{x}|\mathbf{c}; \boldsymbol{\theta}) d\mathbf{c}$$



Generative Matching Network

Explicit conditioning (no global latent variable):

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N|\boldsymbol{\theta}) = \prod_{i=1}^{N} \int p(\mathbf{z}_i|\mathbf{x}_{<i}; \boldsymbol{\theta}) p(\mathbf{x}_i|\mathbf{z}_{<i}, \mathbf{x}_{<i}; \boldsymbol{\theta}) d\mathbf{z}_i.$$

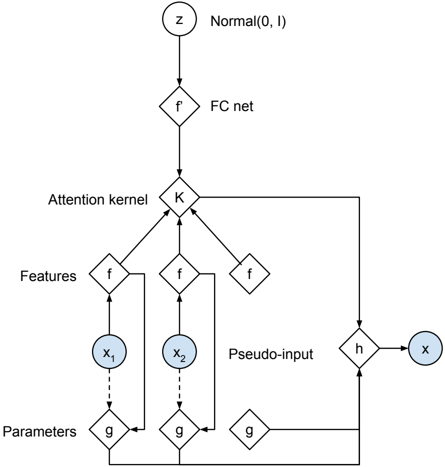
Recognition model:

$$q(\mathbf{z}_1, \ldots, \mathbf{z}_N|\mathbf{x}_1, \ldots, \mathbf{x}_N; \phi) = \prod_{i=1}^{N} q(\mathbf{z}_i|\mathbf{x}_{<i}; \phi)$$

No assumptions on the class structure of data! Inspired by (Vinyals et al, 2016).

Generative part

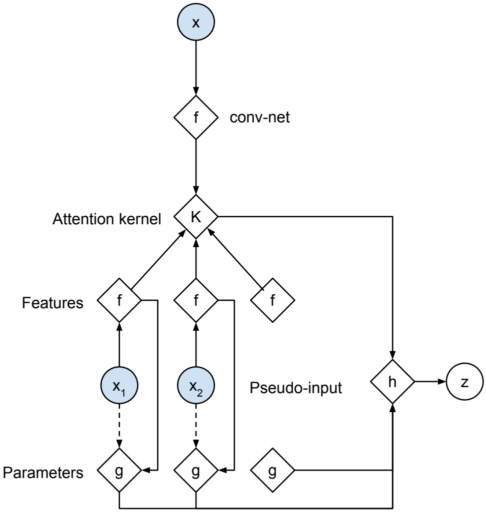
$$p(\mathbf{x}|\mathbf{x}_1,\ldots,\mathbf{x}_k;\boldsymbol{\theta}) = \int p(\mathbf{z}|\mathbf{x}_1,\ldots,\mathbf{x}_k;\boldsymbol{\theta})p(\mathbf{x}|\mathbf{z},\mathbf{x}_1,\ldots,\mathbf{x}_k;\boldsymbol{\theta})dz$$



- $\mathbf{z} \sim \mathcal{N}(\mathbf{z}|0, I)$
- $K(\mathbf{x}_i, \mathbf{z}) = \frac{\exp(cos(f'(\mathbf{z}), f(\mathbf{x}_i)))}{\sum_{j=1}^{k} \exp(cos(f'(\mathbf{z}), f(\mathbf{x}_j)))}$
- $h = \sum_{i=1}^{k} K(\mathbf{x}_i, \mathbf{z})g(\mathbf{x}_i)$
- $\mathbf{x} \sim p(\mathbf{x}|h)$

Recognition part

$$q(\mathbf{z}|\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_k; \phi)$$



- $K(\mathbf{x}_i, \mathbf{x}) = \frac{\exp(cos(f(\mathbf{x}), f(\mathbf{x}_i)))}{\sum_{j=1}^{k} \exp(cos(f(\mathbf{x}), f(\mathbf{x}_j)))}$
- $h = \sum_{i=1}^{k} K(\mathbf{x}_i, \mathbf{x}) g(\mathbf{x}_i)$
- $\mathbf{z} \sim \mathcal{N}(\mathbf{z}|\mu(h), \Sigma(h))$

Training protocol

- Assume there is a distribution $p(D)$ on *datasets* $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.
- Maximize the expected likelihood of a dataset:

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

- Since it's intractable, use variational inference

$$\mathbb{E}_{D \sim p(D)} \log p(D|\boldsymbol{\theta}) \geq$$

$$\mathbb{E}_{D \sim p(D)} \Big[\sum_{i=1}^{N} \log p(\mathbf{x}_i, \mathbf{z}_i | \mathbf{x}_{<i}; \boldsymbol{\theta}) - \log q(\mathbf{z}_i | \mathbf{x}_i, \mathbf{x}_{<i}; \phi) \Big]$$

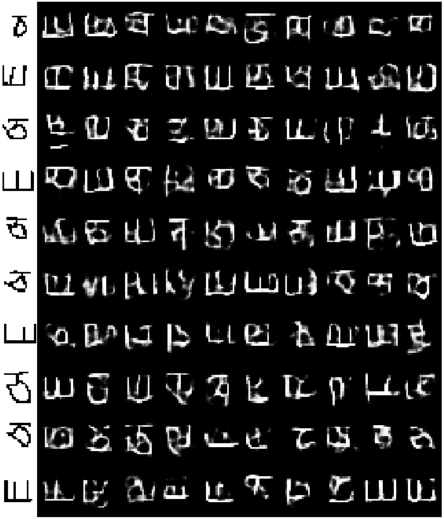
- $p(D)$ is constrained in a way that all objects belong up to C classes

Does it work?

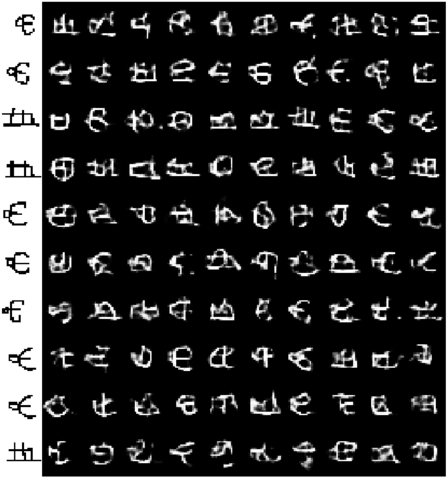
Model	**1st**	**2nd**	**3rd**	**4rd**	**5th**	**10th**
GMN FC	-101.00	-99.67	-98.06	-97.48	-95.89	-92.11
GMN conv	-94.35	-92.26	-90.39	-89.75	-88.65	-84.14
$\sim$ C=3	-95.05	-93.40	-91.87	-91.06	-91.02	-87.55
IWAE K=50, FC	-103.38					
Seq Gen steps=80	≥ -95.5					

Рис. : Results on test Omniglot data. Each column is $\mathbb{E}_D \log p(\mathbf{x}_i|\mathbf{x}_{<i})$.

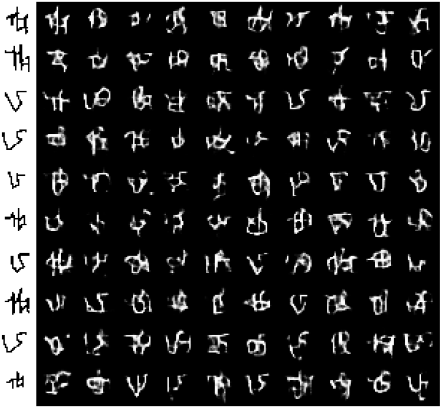
One-shot generalization



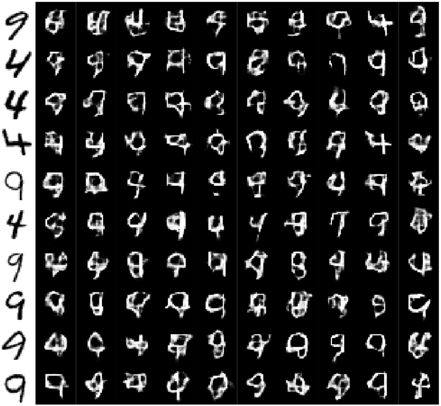
One-shot generalization



One-shot generalization



One-shot generalization



One-shot generalization

Future work

- Architecture design
- Direct parameter prediction
  - e.g. filters of convolution
- Curriculum learning
  - Gradual increasing max. number of classes C during the training