Introduction to Deep Generative Modeling

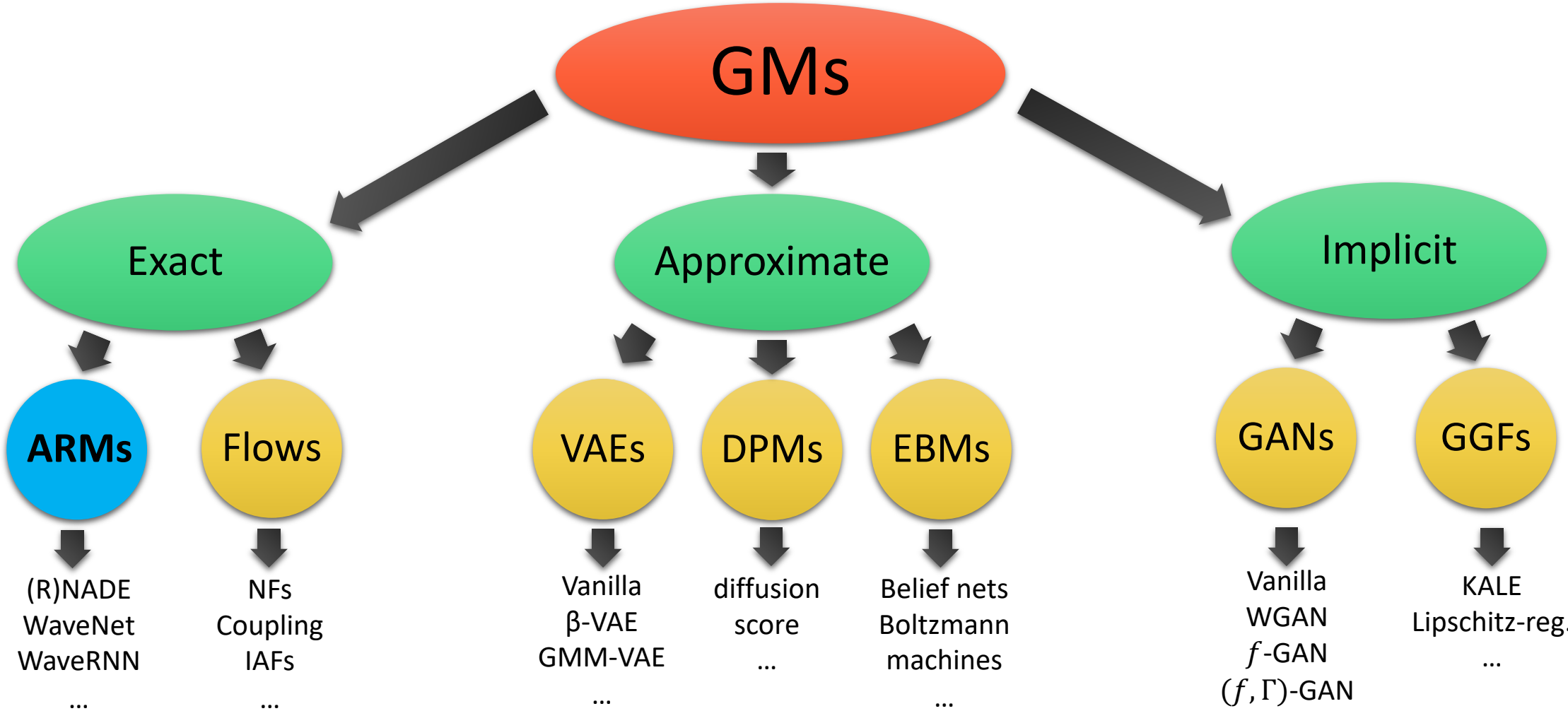
Lecture #5

HY-673 – Computer Science Dep., University of Crete

Professors: Yannis Pantazis & Yannis Stylianou

Teaching Assistant: Michail Raptakis

Taxonomy of GMs



Joint Probability Representations

- Using the chain rule: Fully general, no assumptions needed (exponential size, no free lunch):

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3).$$

- Markov chain: Assumes Markov property:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|\cancel{x_1}, x_2)p(x_4|\cancel{x_1, x_2}, x_3).$$

- Neural Models: Assumes specific functional form for the conditionals. A sufficiently deep neural net can approximate any function:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p_{\text{NN}}(x_2|x_1)p_{\text{NN}}(x_3|x_1, x_2)p_{\text{NN}}(x_4|x_1, x_2, x_3).$$

Neural Models for Classification

- Setting: binary classification of $Y \in \{0, 1\}$ given input features $X \in \{0, 1\}^n$.

- For classification, we care about $p(Y|x)$, and assume that:

$$P(Y = 1|x; \alpha) = f_\alpha(x,).$$

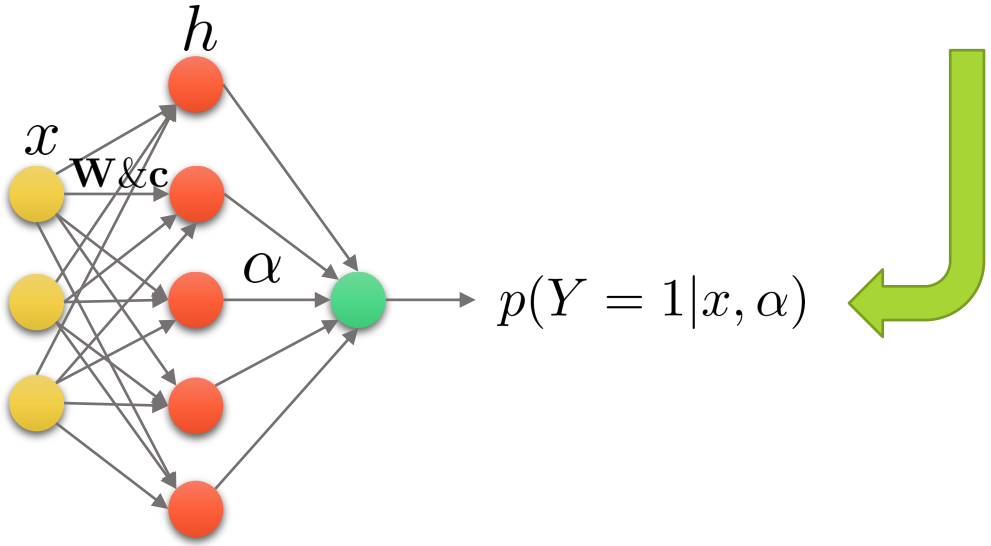
$$\alpha := [\alpha_0, ..., \alpha_n]^T$$

- **Logistic regression:** Let $z(\alpha, x) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$.

$$p_{\text{logit}}(Y = 1|x; \alpha) = \sigma(z(\alpha, x)), \text{ where } \sigma(z) = 1/(1 + e^{-z}).$$

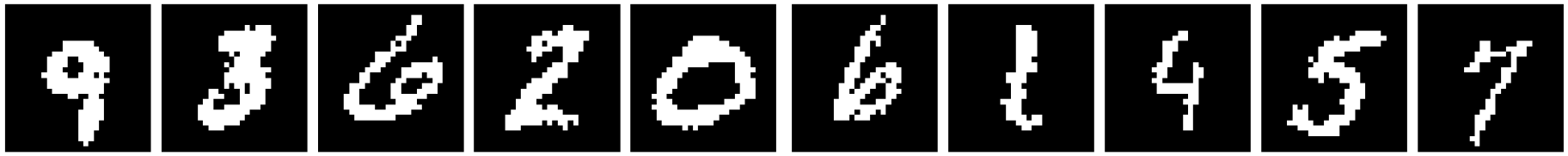
Neural Models for Classification

- **Non-linear** dependence: Let $h(x;\theta)$ be a non-linear transformation of the input features $p_{\text{NN}}(Y=1|x;\boldsymbol{\alpha},\theta) = \sigma\left(\alpha_0 + \sum_{i=1}^{d} \alpha_i h_i\right)$.
  - More flexible since $h(x;\theta) = \tanh(Wx+c)$ with $\theta = \{W, c\}$.
  - More parameters: W, c, α.
  - Repeat multiple times to get a multilayer perceptron (dense-layered NN).



Motivating Example: MNIST

- **Given:** a dataset $\mathcal{D}$ of handwritten digit (binarized MNIST).



- Each image $n = 28 \times 28 = 784$ pixels. Each pixel can either be black (0) or white (1).
- **Goal:** Learn a probability distribution $p(x) = p(x_1, \ldots, x_{784})$ over $x \in \{0, 1\}^{784}$ such that when $x \sim p(x)$, x looks like a digit.
- Two step process:
  1. Parametrize a model family $\{p_\theta(x), \theta \in \Theta\}$.
  2. Search for model parameters θ based on training data $\mathcal{D}$.

Autoregressive (AR) Models

- We can pick an ordering of all random variables, i.e., raster scan ordering of pixels from top-left (x_1) to bottom-right (x_{784}).

- Without loss of generality, we can use chain rule for factorization:

$$p(x_1, \ldots, x_{784}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_n|x_1, \ldots, x_{n-1}).$$

- Lets approximate the above conditionals with:

$$p(x_1, \ldots, x_{784}) = p(x_1; \alpha^1) \times p_{\text{logit}}(x_2|x_1; \alpha^2) \times p_{\text{logit}}(x_3|x_1, x_2; \alpha^3) \times \cdots \times p_{\text{logit}}(x_n|x_1, \ldots, x_{n-1}; \alpha^n).$$

$$\alpha^i := [\alpha_0^i, ..., \alpha_{i-1}^i]^T \in \mathbb{R}^i$$

Autoregressive Models

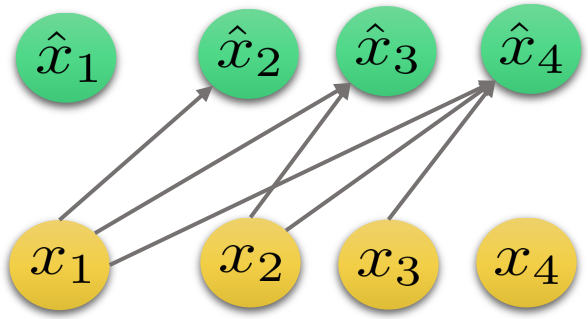
- More explicitly:

  1. $p(X_1 = 1; \alpha^1) = \alpha^1, \; p(X_1 = 0) = 1 - \alpha^1.$
  2. $p_{\text{logit}}(X_2 = 1 | x_1; \alpha^2) = \sigma(\alpha_0^2 + \alpha_1^2 x_1).$
  3. $p_{\text{logit}}(X_3 = 1 | x_1, x_2, \alpha^3) = \sigma(\alpha_0^3 + \alpha_1^3 x_1 + \alpha_2^3 x_2).$

- *Note:* This is a **modeling assumption**.
  We are using parametrized functions (e.g., logistic regression) to predict the next pixel given all the previous ones.
- This is called an **autoregressive** (AR) model.

Fully Visible Sigmoid Belief Network (FVSBN)

FVSBN:



- The conditional variables $X_i|X_1, \ldots, X_{i-1}$ are Bernoulli with parameters:

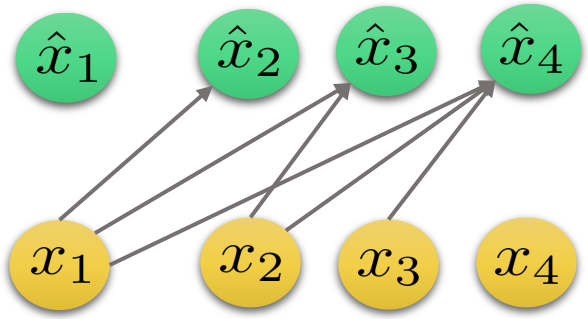
$$\hat{x}_i = p(X_i = 1|x_1, \ldots, x_{i-1}; \boldsymbol{\alpha}^i) = p(X_i = 1|x_{<i}; \boldsymbol{\alpha}^i) = \sigma\left(\alpha_0^i + \sum_{j=1}^{i-1} \alpha_j^i x_j\right).$$

- How to evaluate $p(x_1, \ldots, x_{784})$? Multiply all the conditionals (factors).
  - In the above example:

$$p(X_1 = 0, X_2 = 1, X_3 = 1, X_4 = 0) = (1 - \hat{x}_1) \times \hat{x}_2 \times \hat{x}_3 \times (1 - \hat{x}_4)$$
$$= (1 - \hat{x}_1) \times \hat{x}_2(X_1 = 0) \times \hat{x}_3(X_1 = 0, X_2 = 1) \times (1 - \hat{x}_4(X_1 = 0, X_2 = 1, X_3 = 1)).$$

Fully Visible Sigmoid Belief Network (FVSBN)

FVSBN:



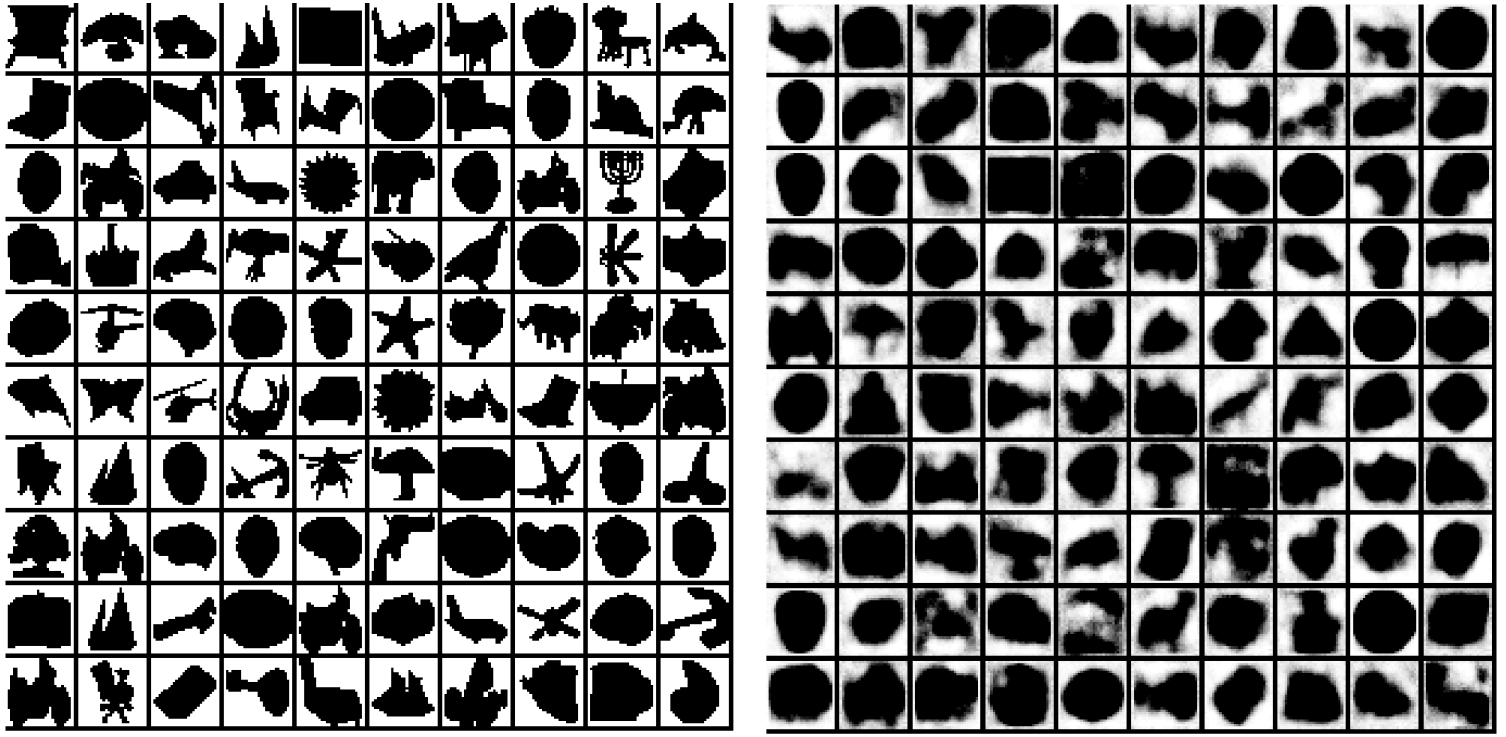
- How to sample from $p(x_1, \ldots, x_{784})$?

1. Sample $\bar{x}_1 \sim p(x_1)$ (`np.random.choice([1,0],p=[x1hat, 1-x1hat])`).
2. Sample $\bar{x}_2 \sim p(x_2|x_1 = \bar{x}_1)$.
3. Sample $\bar{x}_3 \sim p(x_3|x_1 = \bar{x}_1, x_2 = \bar{x}_2) \cdots$

- How many parameters? $1 + 2 + \cdots + n \approx n^2/2$.

FVSBN Results

Training Data
(Caltech 101 Silhouettes)

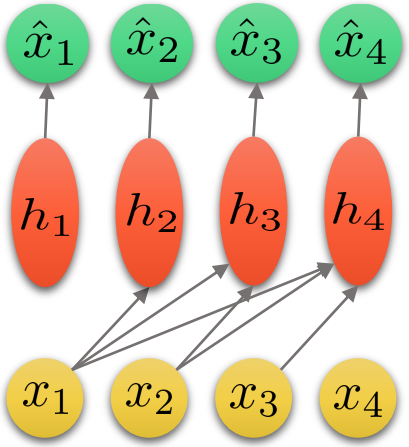


Model Samples

Figure from *Learning Deep Sigmoid Belief Networks with Data Augmentation, 2015.*

NADE: Neural Autoregressive Density Estimation

NADE:



$$h_i = \sigma(W_i x_{<i} + c_i).$$

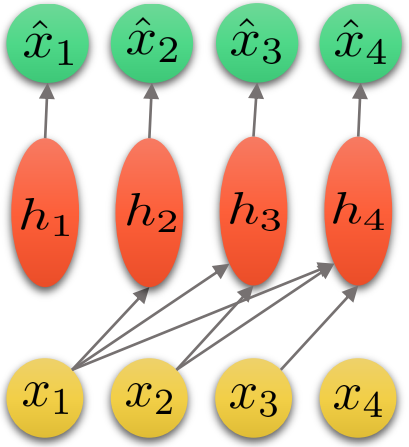
- To improve the model: use one NN layer instead of logistic regression:

$$\hat{x}_i \sim p(x_i | x_1, \ldots, x_{i-1}; \underbrace{W_i, c_i, \alpha_i, b_i}_{\text{parameters}}) = \sigma(\alpha_i^T h_i + b_i).$$

- For example: $h_2 = \sigma\left(\underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{W_2} x_1 + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c_2}\right)$, $h_3 = \sigma\left(\underbrace{\begin{pmatrix}\vdots\\ \vdots\end{pmatrix}}_{W_3} (x_1, x_2)^\intercal + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c_3}\right)$.

NADE: Neural Autoregressive Density Estimation

NADE:



$$h_i = \sigma(W_i x_{<i} + c_i).$$

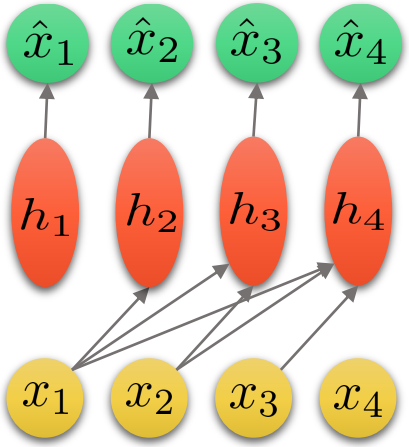
$$W := [w_1, ..., w_{n-1}] \in \mathbb{R}^{d \times (n-1)}$$

- Tie the weights to reduce the number of parameters and speed up computation: $\hat{x}_i \sim p(x_i | x_1, \ldots, x_{i-1}) = \sigma(\alpha_i h_i + b_i)$. For example:

$$h_2 = \sigma\left(\underbrace{\begin{pmatrix}\vdots \\ w_1 \\ \vdots\end{pmatrix}}_{W_{\cdot,<2}} x_1 + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c}\right), \quad h_3 = \sigma\left(\underbrace{\begin{pmatrix}\vdots & \vdots \\ w_1 & w_2 \\ \vdots & \vdots\end{pmatrix}}_{W_{\cdot,<3}} \begin{pmatrix}x_1 \\ x_2\end{pmatrix} + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c}\right), \quad h_4 = \sigma\left(\underbrace{\begin{pmatrix}\vdots & \vdots & \vdots \\ w_1 & w_2 & w_3 \\ \vdots & \vdots & \vdots\end{pmatrix}}_{W_{\cdot,<4}} \begin{pmatrix}x_1 \\ x_2 \\ x_3\end{pmatrix} + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c}\right), \ldots$$

NADE: Neural Autoregressive Density Estimation

NADE:



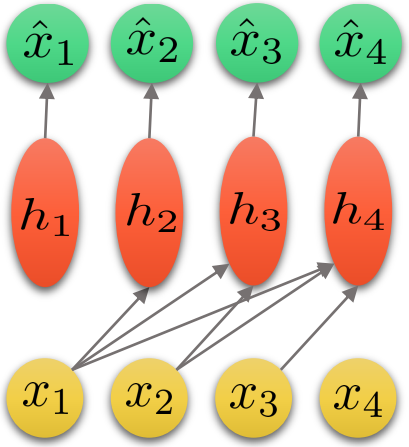
$$h_i = \sigma(W_i x_{<i} + c_i).$$

- If $h_i \in \mathbb{R}^d$, how many total parameters are there?
  - Linear in n: weights $W \in \mathbb{R}^{d \times (n-1)}$ and bias $c \in \mathbb{R}^d$,
  - n logistic regression coefficient vectors with $\alpha_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$.
  - Probability is evaluated in $O(nd)$.
- Computational trick (due to shared weights):

$$(W_{\cdot,<i+1} x_{<i+1} + c) - (W_{\cdot,<i} x_{<i} + c) = w_i x_i.$$

How to train a NADE?

NADE:



$$h_i = \sigma(W_i x_{<i} + c_i).$$

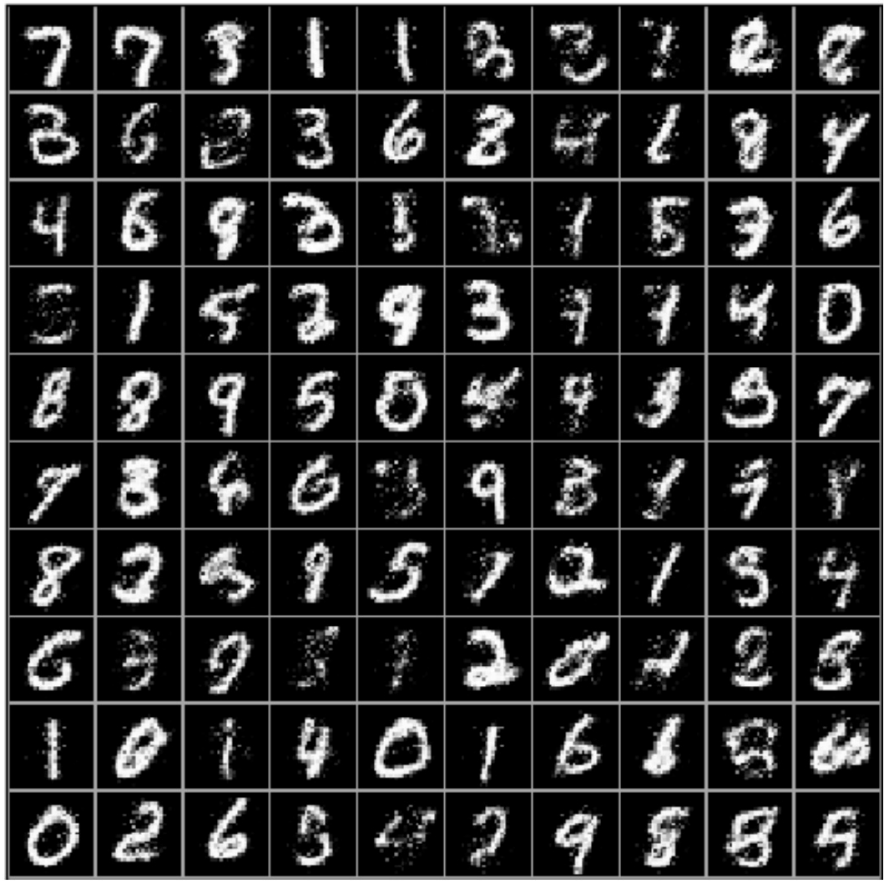
- Maximum log-likelihood estimation:

$$\sum_{x \in \mathcal{D}} \sum_{i=1}^{n} \log p_\theta(x_i | x_{<i}),$$

where $\theta := \{W, c, \underbrace{\alpha_1, ..., \alpha_n}_{A}, b\}$.

NADE Results

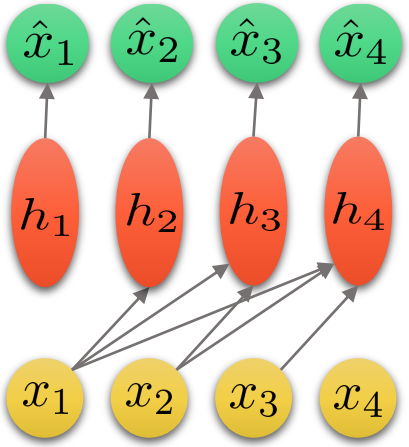
Samples (binary)



Conditional probabilities (in $[0, 1]$)

Figure from *The Neural Autoregressive Distribution Estimator, 2011.*

General Discrete Distributions



$$h_i = \sigma(W_i x_{<i} + c_i).$$

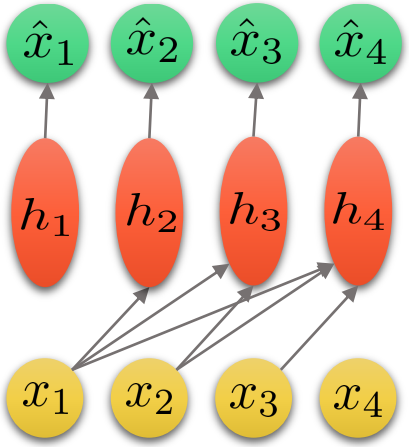
- How to model non-binary discrete random variables $x_i \in \{1, \ldots, K\}$? E.g., pixel intensities varying from 0 to 255 (i.e., greyscale).

- One solution is to let each x_i be parametrized by a categorical distribution:

$$p(x_i | x_1, \ldots, x_{i-1}) = \text{Cat}(p_i^1, \ldots, p_i^K).$$

$$\hat{x}_i \sim \text{Cat}(p_i^1, \ldots, p_i^K) = \text{softmax}(A_i h_i + b_i).$$

General Discrete Distributions



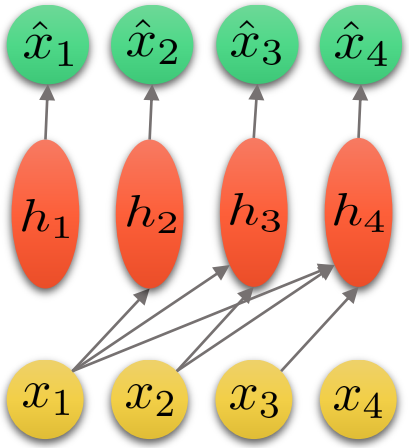
$$h_i = \sigma(W_i x_{<i} + c_i).$$

- Softmax generalizes the sigmoid/logistic function $\sigma(\cdot)$ and transforms a vector of K numbers into a vector of K probabilities (i.e., non-negative & sum to 1):

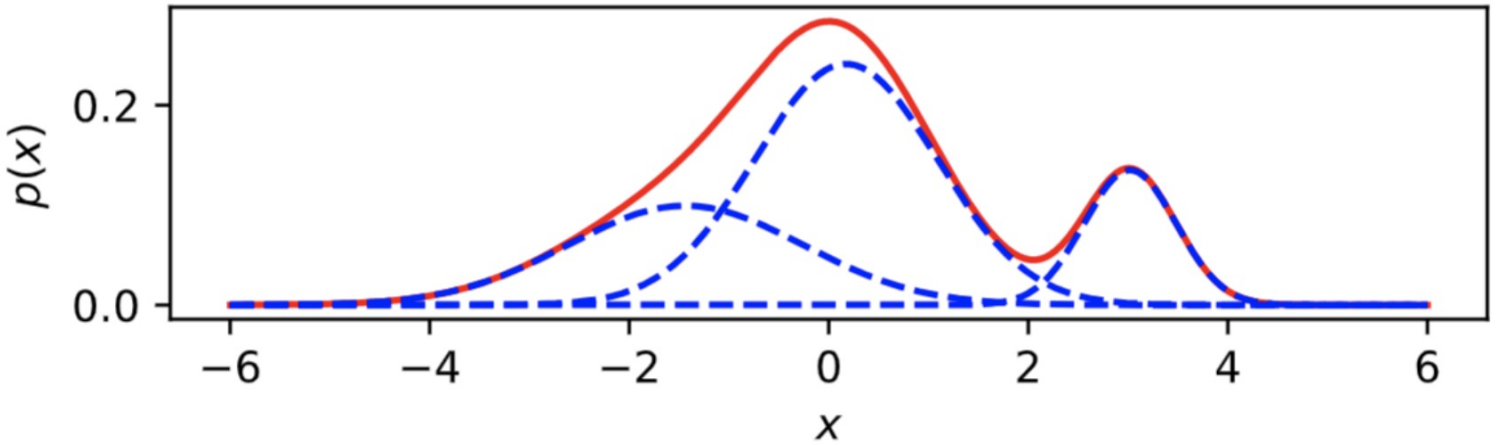
$$\text{softmax}(a) = \text{softmax}(a^1, \ldots, a^K) = \left(\frac{\exp(a^1)}{\sum_i \exp(a^i)}, \cdots, \frac{\exp(a^K)}{\sum_i \exp(a^i)} \right).$$

In numpy: `np.exp(a)/np.sum(np.exp(a))`

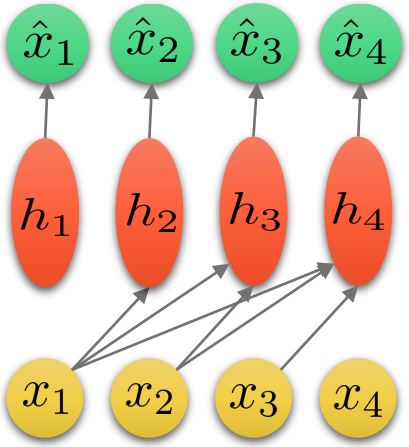
Real-Valued NADE



- How to model continuous random variables $x_i \in \mathbb{R}$?
  E.g., speech signals.
- One solution is to parametrize a continuous distribution.
  E.g., uniform mixture of K Gaussians:



Real-Valued NADE



$$h_i = \sigma(W_i x_{<i} + c_i).$$

- E.g., in a uniform mixture of K Gaussians:

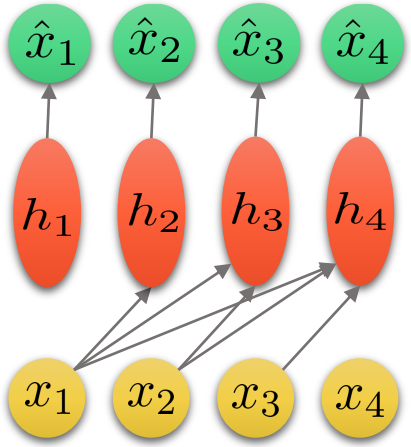
$$\hat{x}_i \sim p(x_i|x_1, \ldots, x_{i-1}) = \sum_{k=1}^{K} \frac{1}{K} \mathcal{N}\left(x_i; \mu_i^k, (\sigma_i^k)^2\right),$$

with $\left[\mu_i^1, \ldots, \mu_i^K, \sigma_i^1, \ldots, \sigma_i^K\right] = f_\theta(h_i)$

$$= \left[m^1(h_i), \ldots, m^K(h_i), \exp(\alpha^1(h_i)), \ldots, \exp(\alpha^K(h_i))\right].$$

- $f_\theta(\cdot)$ defines the mean and std of each Gaussian $\mathcal{N}\left(\mu_i^k, (\sigma_i^k)^2\right)$.

Real-Valued NADE



$$h_i = \text{ReLU}(\rho_i(W_{\cdot,<i}x_{<i} + c)).$$

(more appropriate with real-valued outputs)

- E.g., in a uniform mixture of K Gaussians:

$$\hat{x}_i \sim p(x_i|x_1, \ldots, x_{i-1}) = \sum_{k=1}^{K} \frac{1}{K} \mathcal{N}\left(x_i; \mu_i^k, (\sigma_i^k)^2\right),$$

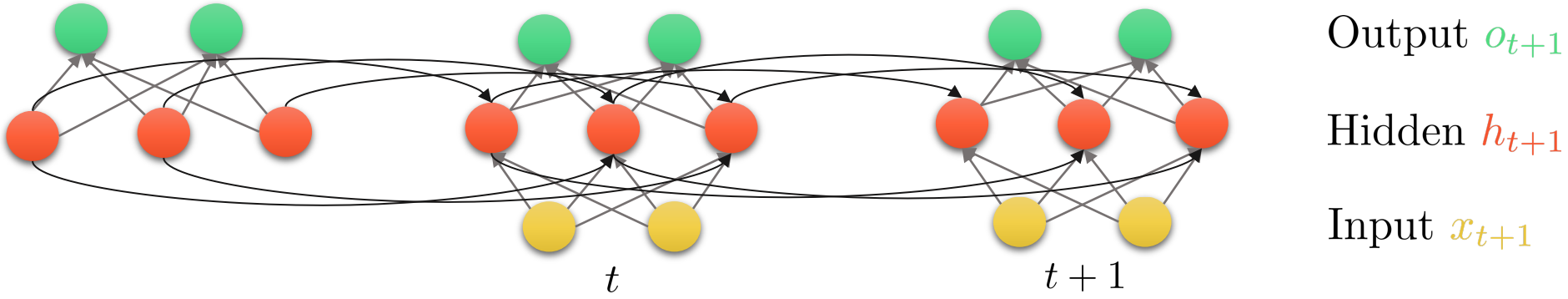
with $\left[\mu_i^1, \ldots, \mu_i^K, \sigma_i^1, \ldots, \sigma_i^K\right] = f_\theta(h_i)$

$$= \left[m^1(h_i), \ldots, m^K(h_i), \exp(\alpha^1(h_i)), \ldots, \exp(\alpha^K(h_i))\right].$$

- $f_\theta(\cdot)$ defines the mean and std of each Gaussian $\mathcal{N}\left(\mu_i^k, (\sigma_i^k)^2\right)$.

RNNs: Recurrent Neural Networks

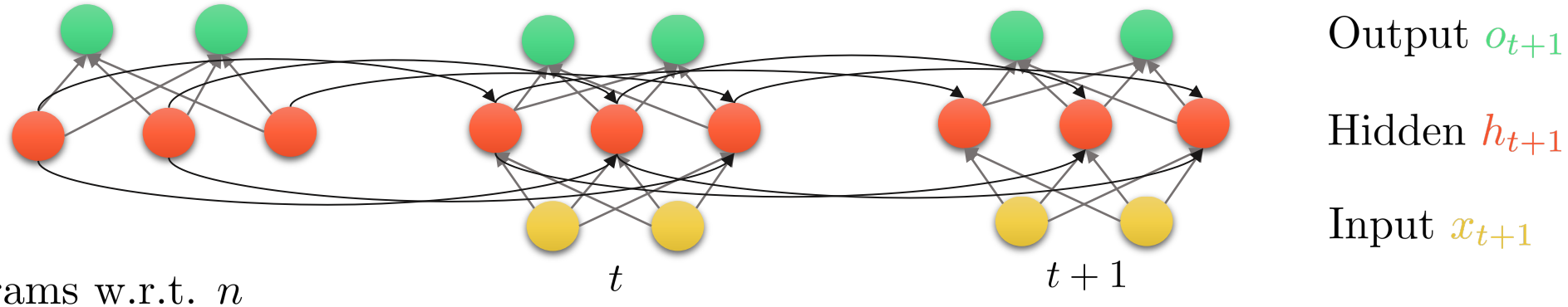
- **Challenge**: model $p\left(x_t | x_{1:t-1}; \boldsymbol{\alpha}^t\right)$. "History" $x_{1:t-1}$ keeps getting longer.
- **Idea**: keep a summary and recursively update it:



1. Summary update rule: $h_{t+1} = \tanh\left(W_{hh} h_t + W_{xh} x_{t+1}\right)$.
2. Prediction: $o_{t+1} = W_{hy} h_{t+1}$.
3. Summary initialization: $h_0 = \mathbf{b}_0$.

RNNs: Recurrent Neural Networks

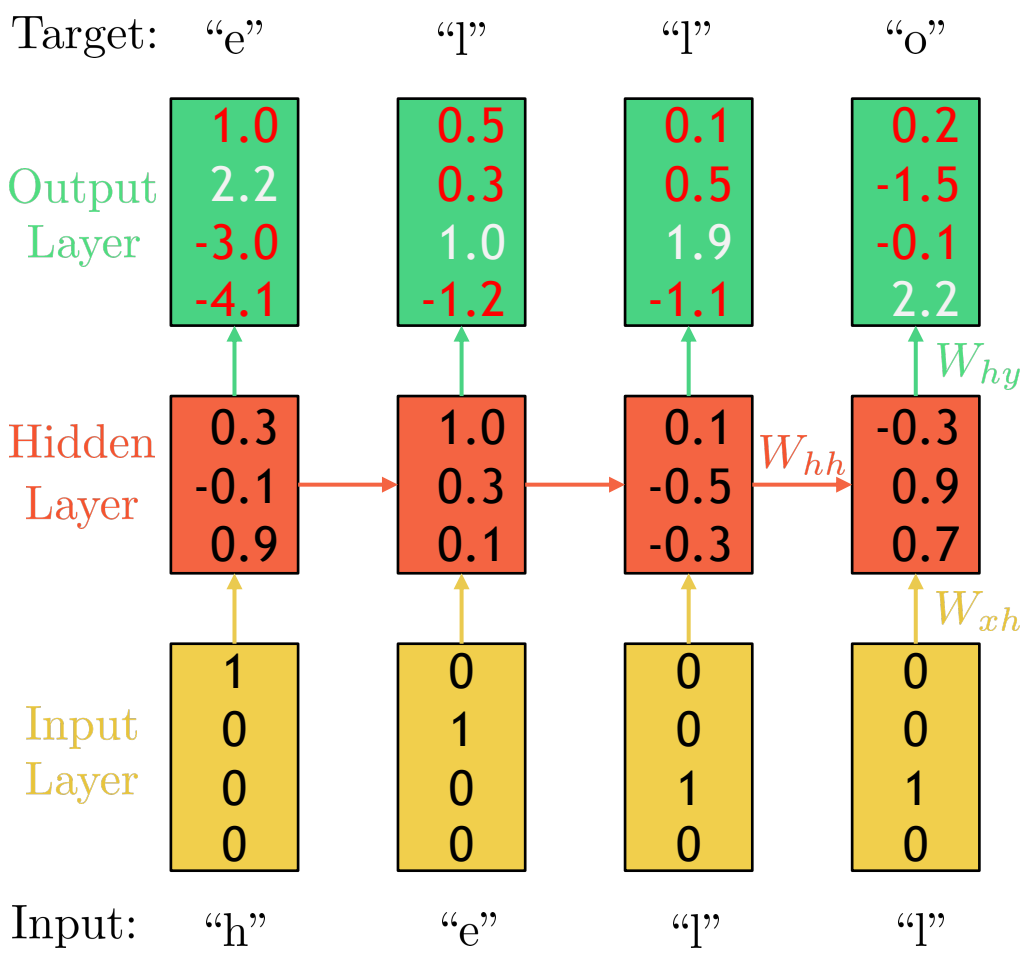
- **Challenge**: model $p\left(x_t | x_{1:t-1}; \boldsymbol{\alpha}^t\right)$. "History" $x_{1:t-1}$ keeps getting longer.
- **Idea**: keep a summary and recursively update it:



constant #params w.r.t. n

1. Hidden layer h_t is a summary of the inputs seen till time t.
2. Output layer o_{t-1} specifies parameters for conditional $p(x_t | x_{1:t-1})$.
3. Parametrized $\mathbf{b}_0$ (initialization), and matrices W_{hh}, W_{xh}, W_{hy}.

Example: Character RNN (Andrej Karpathy)



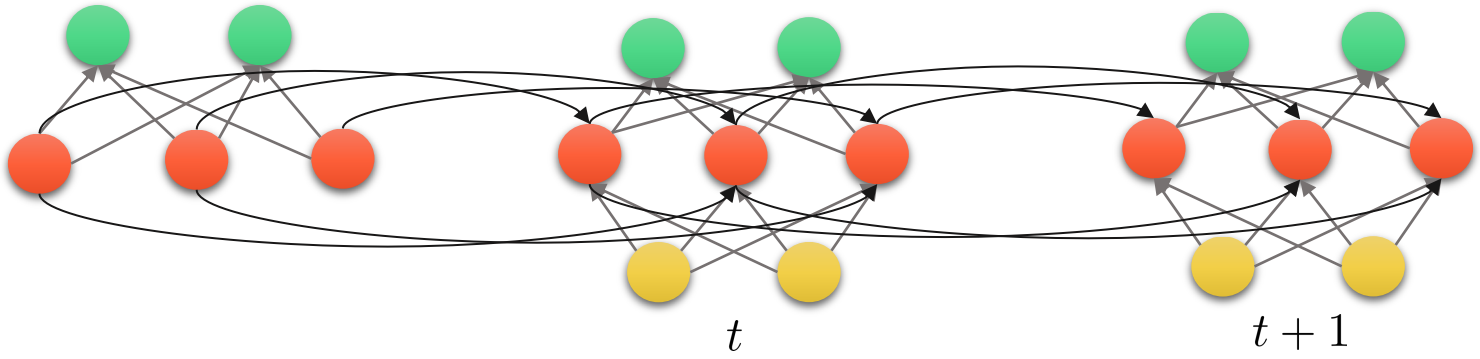
1. Suppose $x_i \in \{\text{“h”}, \text{“e”}, \text{“l”}, \text{“o”}\}$ & one-hot:

- “h” is encoded as $[1, 0, 0, 0]$, “e” as $[0, 1, 0, 0]$, etc.

2. **Autoregression**: $p(x = \text{“hello”}) =$
$p(x_1 = \text{“h”})p(x_2 = \text{“e”}|x_1 = \text{“h”})$
$p(x_3 = \text{“l”}|x_1 = \text{“h”}, x_2 = \text{“e”}) \cdots$
$p(x_5 = \text{“o”}|x_1 = \text{“h”}, \ldots, x_4 = \text{“l”}).$

3. For example: $p(x_2 = \text{“e”}|x_1 = \text{“h”}) =$
$\text{softmax}(o_1) = e^{2.2} / \left(e^{1.0} + \cdots + e^{4.1}\right).$
$o_1 = W_{hy} h_1, \;\; h_1 = \tanh\left(W_{hh} h_0 + W_{xh} x_1\right).$

RNN Pros



Output o_{t+1}

Hidden h_{t+1}

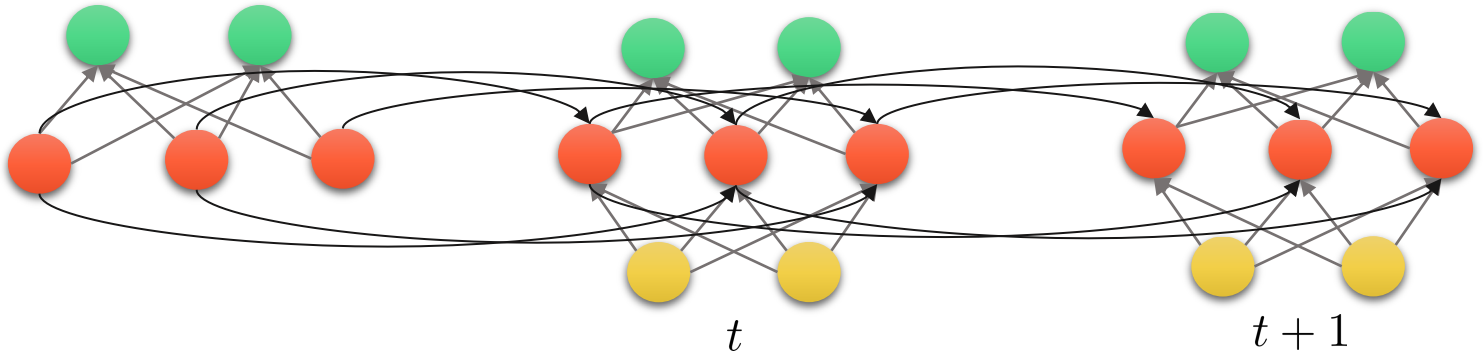
Input x_{t+1}

Pros:

1. Can be applied to sequencies of arbitrary length.

2. Very general: For every computable function, there exists a finite RNN that can compute it.

RNN Cons



Output o_{t+1}

Hidden h_{t+1}

Input x_{t+1}

Cons:

1. Still requires an ordering.
2. Sequential likelihood evaluation (very slow for training).
3. Sequential generation (unavoidable in an AR model).
4. Can be difficult to train (vanishing/exploding gradient issues).

Example: Character RNN (Andrej Karpathy)

- Train a 3-layer RNN with 512 hidden nodes on all the works of Shakespeare, then sample from the model:

> KING LEAR: O, if you were a feeble sight, the courtesy of your law,
> Your sight and several breath, will wear the gods
> With his heads, and my hands are wonder'd at the deeds,
> So drop upon your lordship's head, and your opinion
> Shall be against your honour.

- Note: The model's generation happens **character by character**, so, it needs to learn valid words, grammar, punctuation, etc.

Example: Character RNN (Andrej Karpathy)

- Train on Wikipedia, then sample from the model:

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25—21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict.

- Note: Correct Markdown syntax, opening and closing of brackets [[·]], etc.

Example: Character RNN (Andrej Karpathy)

- Train on dataset of baby names, then sample from the model:

Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacrie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa Wallon Martine Merus Jelen Candica Wallin Tel Rachene Tarine Ozila Ketia Shanne Arnande Karella Roselina Alessia Chasty Deland Berther Geamar Jackein Mellisand Sagdy Nenc Lessie Rasemy Guen Gavi Milea Anneda Margoris Janin Rodelin Zeanna Elyne Janah Ferzina Susta Pey Castina

Summary of ARMs

1. Easy to sample from.
   (a) Sample $\bar{x}_0 \sim p(x_0)$.
   (b) Sample $\bar{x}_1 \sim p(x_1|x_0 = \bar{x}_0)$.
   (c) $\cdots$
2. Easy to extend to continuous variables. For example, can choose Gaussian conditionals $p(x_t|x_{<t}) = \mathcal{N}\left(\mu_\theta\left(x_{<t}\right), \Sigma_\theta\left(x_{<t}\right)\right)$, or mixture of logistics.
3. No natural way to get features, cluster points, do unsupervised training.

Summary of ARMs

4. Easy to compute probability $p(x = \bar{x})$.
   (a) Compute $p(x_0 = \bar{x}_0)$.
   (b) Compute $p(x_1 = \bar{x}_1 | x_0 = \bar{x}_0)$.
   (c) Multiply together (sum of the logarithms).
   (d) $\cdots$
   (e) Ideally, can compute all these terms in parallel for fast training.

References

1. `https://deepgenerativemodels.github.io`
2. The Neural Autoregressive Distribution Estimation, Larochelle and Murray
3. RNADE: Neural Autoregressive Distribution Estimation, Uria et al.

Introduction to Deep Generative Modeling

Lecture #5

HY-673 – Computer Science Dep., University of Crete

Professors: Yannis Pantazis & Yannis Stylianou

Teaching Assistant: Michail Raptakis