

# Lecture 2: Autoregressive Models and Flow Models

Scribed by: Shao Yang Hong, Kan Min-Yen, Joni Ngo, Who else??

Lecture presented by: Ang Ming Liang and Eugene Lim

## Toy Autoregressive Models (Recap)

Learn a NN to output a probabilistic density function.

Use conditional dependence to model the joint distribution.

## Recurrent Neural Networks

- RNN can be framed as an AR model

## Autoencoders

- Take original image  $i$ , “squeeze” features into a small feature space, and “decompress” it to match input
- Autoencoder learns a good underlying representation of images  $i_1 - i_N$
- **Why represent the inputs in less no. of neurons when autoencoding:** If the size of the hidden layers is the same as the input, the “squeeze” won’t happen and it’ll learn an identity matrix. The squeezing exists to force the model to generalize and compress the information into a small number of neurons in the hidden layer.
- Slight digression to Colah’s blog:  
<https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>

## MADE: Masked Autoencoder for Distribution Estimation

- Allows calculation of likelihood in a single pass
  - Masked are defined based on Bayesian Beliefs (i.e. “don’t see into the future”)
  - Masked Autoencoder can be implemented simply as a matrix multiplication (autoencoder \* masks)
  - The whole point is to do one forward pass to calculate the probability of each variable; the likelihood of any data point.
- Sampling is slow because you have iteratively do forward passes until you have the entire distribution for all of the variables.
- References and links
  - <https://arxiv.org/abs/1502.03509>
  - [A PyTorch MADE from the great Karpathy](#)

- Mask is usually created as an upper triangular matrix or can just be at random (?) as long as any variable is not dependent on future variable
- Fast for inference, but slow for sampling

## Masked Convolution

### 1D convolution - WaveNet

WaveNet - State of the Art for speech synthesis (eg generating Donald Trump's voice!)

- Based on Temporal (1D) Convolution. Inputs are eg the volume samples of the input audio
- **Receptive Field:** All inputs that the output sees
- **Dilated convolution:** exponentially sample "less and less"

**Question:** how does padding work on TensorFlow (slide 25 of 53)

See <https://www.kaggle.com/alvations/padding-slide-25>

### 2D convolution - PixelCNN (2016)

- Impose an AR ordering on pixels
- Gated PixelCNN - Horizontal stack + Vertical stack
  - Used a modified ResNet Block - which they call a Gated ResNet Block
- PixelCNN++

## Masked Attention

- Originally applied to seq2seq models. See for example: <http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/>
- **Importance weights ( $\alpha$ ):** Importance measures for hidden state
- Self-attention has unlimited receptive field
- Masked Attention + Convolution => Good receptive fields
- Speedups by caching activations

## Second Half: Flow Models

Recap of Gaussian Mixture Models

- Usually trained using expectation maximization  
[https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization\\_algorithm](https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm)

Inference:

- Given a datapoint, tell me the probability of the datapoint

Sampling:

- Hallucinate datapoint, given that you can estimate the probabilities of datapoint

Keywords definitions:

- **Jacobian** = the matrix gradient

$$\mathbf{J}(x_1, \dots, x_n) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \dots & \frac{\partial y_n}{\partial x_n} \end{bmatrix}.$$

- **Determinant** = transformation of function to another
- Jacobian determinant
- **Affine** = fancy word for linear with a bias (sometimes known as “wx + b”, aka simple feedforward)
- **Monotonic function**: Strictly increasing/decreasing function.

**Question:** Must the inverse  $f^{-1}_\theta(x)$  in the elementwise flows be the same function? Or can it be different?

RealNVP: <https://arxiv.org/abs/1605.08803>

This is pretty nice and related: <https://blog.evjang.com/2018/01/nf1.html>

Related Links