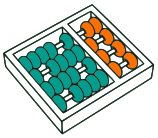



# Autoregressive Models

Lesson No. 02

Samuel Felipe Chenatti

## 1 Unsupervised Learning

Both in industry and in academy, labeling data for developing Machine Learning algorithms is an expensive and time consuming process. In some cases, it can be infeasible to have the amount of data required for achieving a trustable model. In parts this ineficiency is due to the fact that we are not directly training these models to learn better representations of the data domain.

This is where Unsupervised Models comes into play: this family of models are designed to directly learn the peculiarities of the domain by *econding* and *decoding* (or reconstructing) the data. After trained, this models can be fine-tuned in a supervised way with less labeled data. Even better, since some Unsupervised Models are modeled to explicitly learn the underlying data distribution, we can use them to generate new data such as image, text and audio.

## 2 Likelihood models

The basic idea of a likelihood model is to learn the underlying distribution of the data; or, mathematically: learn $p(.)$ from $z(\mathbf{x})$, where $z(.)$ is the true data distribution, $p(.)$ is the probability that data comes from $z(.)$, and $\mathbf{x}$ is a feature vector such as an image or an audio. Then, for generating new data we can simply sample from $\mathbf{x} \sim p(\mathbf{x})$.

Review from basic probabilities: since $p(.)$ takes a point from the data domain and returns a probability between 0 and 1, we can then "simply" sample a number from a uniform distribution $z \sim U(0,1)$ and plug it into the learned probaiblity to generate a new sample: $x \sim p(z)^{-1}$.

Since we are dealing with high dimensional data, we take our distribution to be $p_\theta$ where in most of the cases $\theta$ is a Deep Neural Network, which is a model well known for learning data generalization with a finite number of parameters (this is important since we does not have access to the real data distribution, only to the data it generates). The architecture of these DNNs heavily depends on the structure of the data being learned (such as temporal or spacial dependency).

The obvious framework for learning likelihood models is the *Maximum Likelihood* for which the loss function is showed in equation 1.

$$\text{argmin}_\theta \, p(\mathbf{x_0}, \mathbf{x_1}, ..., \mathbf{x_n}) = \frac{1}{N} \sum_i^N -\log p_\theta(\mathbf{x}_i) \tag{1}$$

## 3 Autoregressive models

The core idea of *Deep Autregressive Models* is to take advantage of *Bayes Nets* while designing DNNs.

Review from basic probabilities: Bayes Nets are Directed Acyclic Graphs that defines a dependency between random variables. For example, given the distribution $p(x_1, x_2, x_3) = p(x_1) * p(x_2|x_1)p(x_3|x_2, x_1)$, we can represent as the DAG in figure **??**. It is important to note that *any prob distribution can be modeled as a Bayes Net.*

Since the Bayes Net follows from the basic rules of probability, designing our DNNs following this scheme constrains its output to be a valid probability distribution. We can see from equation 2 that the loss for such model is also tractable.

$$\log p_\theta(\mathbf{x}) = \sum_i^d p_\theta(\mathbf{x}_i|\mathbf{x}_{1:i}) \tag{2}$$

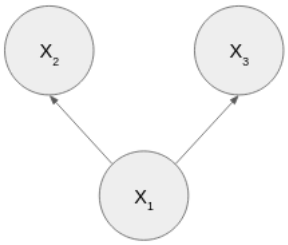


**Figure 1.** A representation of a Bayesian Network composed of three random variables.

| | | |
|---|---|---|
| $x_0$ | $x_1$ | $x_2$ |
| $x_3$ | $x_4$ | $x_5$ |
| $x_6$ | $x_7$ | $x_8$ |

**Table 1.** A computational representation of an image where each pixel is a random variable.

So, for example, for an image feature, we can assume that each of its pixels are random variables $x_0, ..., x_n$ such as in the table **??**. We can also assume that each of the pixels is dependent of the other in some order, so the problem can be modeled by a Bayes Net. This class of models is what we call *Autregressive Models*.

## Sharing parameters

It is obvious that developing one DNN for each conditional distribution is not a scalable option, so instead we try to design a single Neural Network that outputs the conditional probs given the data. The main issue is that we cannot allow an output to "see" data it is not related to; that is, $p(x_1|x_2)$ cannot have access to $x_3$ since this probability is not conditioned on it.

One of the most basic methods to allow this kind of dependency are RNN. For a given sentence, if we assume each charactere as a random variable and we condition each letter on all of its predecessors, we can then pass the data into a Recurrent Neural Network where each hidden unit will output the probability of the next char. The recurency ensures the conditional constraint.

## Masking-based Autoregressive Models

In these models we still use an DNN, but we make sure that a conditional probability does not see the data it does not depends on. If we think of a classical Multilayer Perceptron as a graph, one of its core ideas is that there is always a path between any input node and output node. In *Masked-based* approaches, the connections (or synapses in Deep Learning terminology) that could lead from an output node to the output of the conditional distribution it does not depends on is masked. Since the connections between any two nodes are represented by matrices of learnable parameters $\mathbf{W}$, the masking is accomplished by setting the correct parameters $w \in \mathbf{W}$ to zero.

MADE [1] makes use of this approach to model an autoencoder capable of learning distribution estimations. Figure **??** ilustrates the Masked connections implemented by MADE. We can see from the resulting Neural Network on the right that, following the Bayes Network presented by the output conditional probabilities, the input $x_1$ has no path to any of the output neurons whereas $p(x_2)$ has no path to any input.

### Masked Temporal 1D Convolution

This models also relies in masking some of the parameters to achieve the conditional probabilities, but instead of making use of Multilayer Perceptrons we use masked Convolution kernels so that each output only has access to the previous inputs. However, since the kernel is limited in size, the receptive field (or the nodes to which the kernel has access) is limited to few inputs so the length of the input sequence cannot be very long.

To mitigate this issue, Wavenet has introduced masked *Dilated Convolutions* wich increases the receptive field exponentially through the sequence.

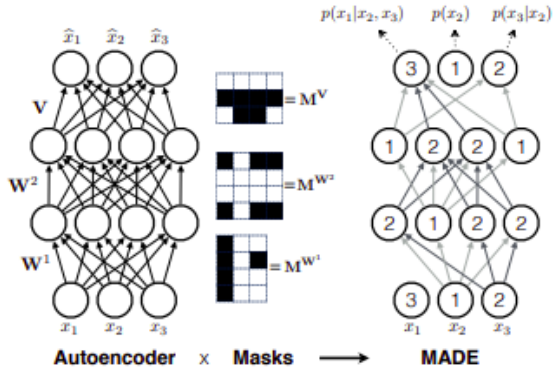


**Figure 2.** Adapted from MADE [1]. On the left we have an conventional three hidden layer autoencoder. MADE uses masked activations, as showed in the middle column to achieve the Network on the right, which obeys to the Bayes Net structure.

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

**Table 2.** Masked 2d convolution. In this case we are conditioning the probability of the pixel that matches the center of the kernel on all the pixels that comes before it.

### Masked Spatial 2D Convolution

To make use of these methods along with image data, we could simply flatten a 2d image matrix into a 1d vector and use the methods of the last subsection. But this would imply in losing the spatial relation between some of the pixels. To keep this kind of relation, models such PixelCNN keep the depedency of the random variables in a raster way (in table **??**, for example, $x_4$ depends on $x_0, x_1, x_2, x_3$.

An example of masked 2d convolution as proposed by the PixelCNN[2] can be seem in table **??**. The issue with this kernel is that it will create a blind-spot in the receptive field, ie, some of the pixels won't be reachble by the kernel. This issue can be overcomed by adopting stacked convolutions that covers the blind regions.

## 4 Conclusion

Unsupervised learning is an exciting field that enables the adoption of Deep Learning by the industry, since it reduces the need for large amounts of labeled data for training models. In fact, self-supervised models such as BERT[3] - a language encoder pretrained in tons of Wikipedia articles is allowing the development of NLP products for a large range of domains where automation is highly needed. In this sense, unsupervised learning will soon become a essential tool in any Machine Learning pipeline.

## References

[1] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked Autoencoder for Distribution Estimation," *arXiv e-prints*, arXiv:1502.03509, arXiv:1502.03509, Feb. 2015. arXiv: 1502.03509 [cs.LG].

[2] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications," in *ICLR*, 2017.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints*, arXiv:1810.04805, arXiv:1810.04805, Oct. 2018. arXiv: 1810.04805 [cs.CL].