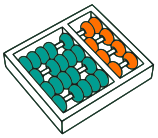

Autoregressive Models

Lesson No. 2

José Dorivaldo Nascimento Souza Júnior - RA 170862

1 Introduction

In the context of *deep unsupervised learning*, it is introduced herein the **autoregressive models**, that is a subfield of the Likelihood-based models. The general field is explained first, and then the features that distinguish the autoregressive models are pointed. Finally, some works that used Autoregressive models are cited.

2 Likelihood-based Models

2.1 Overview

Given a dataset $\mathscr{X}$, the models from *Likelihood-based Models*, by definition, aims to estimate the **probability distribution** from the samples $x_1, x_2, ..., x_n \sim \mathscr{X}$. It means that, given a *trained* model $p_\theta(x)$ (where p represents a neural network architecture, and θ represents the weights), inputting x into the model will output the probability for x being part of the "group" represented by $\mathscr{X}$.

2.2 Negative Log-Likelihood

Likelihood-based models use **Negative Log-Likelihood** as Loss Function. The definition of Negative Log-Likelihood is based on the concept of *Maximum Likelihood Estimation*, and its definition is:

$$\underset{\theta}{\operatorname{argmin}}\, loss(\theta, x_1, ..., x_n) = \frac{1}{n}\sum_{i=1}^{n} -log\, p_\theta(x_i) \tag{1}$$

We can observe in this loss function that it aims to find a set of weights θ such as it *increases* the outputted probability $p_\theta(x)$ for each item x from the dataset $\mathscr{X}$.

2.3 Applications

Likelihood-based models have some interesting application, such as:

- **Sampling**: We can generate new data similar to the items from the dataset, since we can estimate what is a high-probability item.
- **Anomaly Detection**: Given that we know the distribution, a possible outlier may be straighforwardly detected by inputting this outlier into that distribution and then checking the output, as it is expected to have a low value.

3 Autoregressive Models

One of the requirements for a likelihood-based model is that, by the definition of distribution, *the sum of the outputted probability for the entire set of possible inputs should be equal to 1.* As a consequence, the architecture of the neural network p should be designed carefully (otherwise the output will not be a distribution). This design is what distinguish Autoregressive models from other likelihood-based models.

The concepts that the design of an autoregressive model relies on will be explained in the next subsections.

Table 1. Value for a vertex X in a Bayes Network, in which its parents are W and Z.

	x=0	x=1
w=0; z=0	0.01	0.99
w=0; z=1	1	0
w=1; z=0	0.1	0.9
w=1; z=1	0.5	0.5

3.1 Conditional Distribution

Given two random variables x and y, we define the conditional distribution $P(x|y)$ as the probability of x given y.

3.2 Joint Distribution

Given two random variables x and y, we define the joint distribution of both variables as:

$$P(x, y) = P(x)P(y|x) \tag{2}$$

3.3 Bayes Network

A Bayes Network uses the concept of conditional distribution in a Directed Acyclic Graph. In a Bayes Network, each vertex corresponds to a random variable, and each edge corresponds to a conditional relation. For instance, given a vertex x, which parents are the vertices w and z, an possible configuration for the random variable represented by x is described in Table 1. It is important to point that the sum of each probability of x is 1 for every combination of w and z.

3.4 Summing Up: Modelling an Autoregressive Model

Given the concepts cited above, we may define the basis of an autoregressive model architecture as a *Bayes network that output some distribution as a product of conditional distributions*. It means that, given a dataset $\mathscr{X}$ that contains items of D dimensions, we should define an ordering for the dimensions, and after we train the model, we obtain the distribution by generalizing the equation **2** :

$$P(x) = P(\bigcap_{i=1}^{D} x^{(i)}) = P(x^{(1)}) \prod_{i=2}^{D} P(x^{(i)} | \bigcap_{j=1}^{i-1} x^{(j)}) \tag{3}$$

Where $P(x^{(1)})$ and each item $P(x^{(i)}|\bigcap_{j=1}^{i-1} x^{(j)})$ from the produtory will be outputted at the final layer of the network.

4 Implementation and Some Relevant Works

4.1 Recurring Neural Networks

Given the specification above, one possible way to implement an autoregressive model is by a RNN, since RNNs are suitable to work with sequences. In this case, we first input $x^{(1)}$, and them when we input $x^{(d)}$, the result will be influenced by $x^{(<d)}$ through the hidden layer (therefore we will have the conditional probability). A relevant work in this sense is **Pixel Recurrent Neural Network** [1].

4.2 Masking

On the other hand, other way to implement is using the concept of *masking*. In masking, instead of using a RNN, we desing the network wisely, so *we can calculate all conditionals in parallel* - something that is not possible in a RNN. Some relevant works in this sense are:

- **MADE** [2]: Given a MLP Neural Network, we mask some weights (i.e, turn them into 0), so the conditional for dimension $d^{(i)}$ will depend only on the inputs of $d^{(<i)}$.

- **Wavenet** [3]: It generates audio in a architecture with 1D convolution by (i) masking part of the receptive field (so $d^{(i)}$ will depend only on its predecessors), and (ii) applying the concept of *dilated convolution*, so the conditionals will not be limited by the size of the receptive field (making it possible for conditioning in every predecessor).
- **PixelCNN** [1]: It applies the concept in a convolutional layer, by masking half of the receptive field (all pixel in the line below the center plus all pixels on the right), creating a conditional relationship.
- **Gated PixelCNN** [4]: It enhances PixelCNN by using two stacks (horizontal and vertical) as receptive field, in order to avoid some *black holes* in the image that happens in PixelCNN during the convolution procedure.
- **PixelCNN++** [5]: Further improvement of PixelCNN, it uses the concept of *attention* [6] to conditioning on the entire range of pixels.

5 Conclusion

Autoregressive models are reported to have a good performance and stable training. Also, it is possible to calculate the probability of an input straighforwardly. On the other hand, one of the main drawbacks is that, even though by masked convolutions it is possible to paralellize the conditioning, the sampling procedure still cannot be paralellized, therefore it is quite expensive to sample from autoregressive models. To work around this issue it was developed the *Flow models*.

References

[1] Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel Recurrent Neural Networks.” ArXiv:1601.06759 [Cs], August 19, 2016. http://arxiv.org/abs/1601.06759.

[2] Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle. “MADE: Masked Autoencoder for Distribution Estimation.” ArXiv:1502.03509 [Cs, Stat], June 5, 2015. http://arxiv.org/abs/1502.03509.

[3] Oord, Aaron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio.” ArXiv:1609.03499 [Cs], September 19, 2016. http://arxiv.org/abs/1609.03499.

[4] Oord, Aaron van den, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. “Conditional Image Generation with PixelCNN Decoders.” ArXiv:1606.05328 [Cs], June 18, 2016. http://arxiv.org/abs/1606.05328.

[5] Salimans, Tim, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. “PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications.” ArXiv:1701.05517 [Cs, Stat], January 19, 2017. http://arxiv.org/abs/1701.05517.

[6] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need.” ArXiv:1706.03762 [Cs], December 5, 2017. http://arxiv.org/abs/1706.03762.