An Improved Training Procedure for Neural Autoregressive Data Completion

Maxime Voisin
Stanford University
maximev@cs.stanford.edu

Daniel Ritchie
Brown University
daniel_ritchie@brown.edu

1 Introduction

Neural autoregressive models are explicit density estimators that achieve state-of-the-art likelihoods for generative modeling [3, 4, 5, 7]. The D-dimensional data distribution $p(x)$ is factorized into an autoregressive product of one-dimensional conditional distributions according to the chain rule. Each conditional distribution is parametrized by a shared neural network.

Data completion is a more involved task than data generation: the model must infer missing variables for any partially observed input vector. Previous work [5] introduced an order-agnostic training procedure for data completion with autoregressive models. It maximizes the average likelihood of the model over all $D!$ orderings of the data dimensions. As a result, all possible one-dimensional conditionals $p(x_{o_d}|x_{o_1}, ..., x_{o_{d-1}})$ are trained, for any $d \in \{1..D\}$ and any ordering o of $\{1...n\}$. Thus, missing variables in any partially observed input vector can be imputed efficiently by choosing an ordering where observed dimensions precede unobserved ones and by computing the autoregressive product in this order. This training procedure can be made efficient: [6] estimates the order-agnostic loss with an unbiased estimator that reuses most computations.

In this paper, we provide evidence that the order-agnostic (OA) training procedure is suboptimal for data completion. We propose an alternative procedure (OA++) that reaches better performance in fewer computations. It can handle all data completion queries while training fewer one-dimensional conditional distributions than the OA procedure. In addition, these one-dimensional conditional distributions are trained proportionally to their expected usage at inference time, reducing overfitting. Finally, our OA++ procedure can exploit prior knowledge about the distribution of inference completion queries, as opposed to OA. We support these claims with quantitative experiments on standard datasets used to evaluate autoregressive generative models.

2 Improving the order-agnostic loss for data completion

The OA procedure trains autoregressive models for data completion by optimizing the loss $\mathscr{I}_{\text{OA}}(\theta)$ in Equation 1. In practice, the exact loss has too many terms to be computationally tractable. It is estimated by sampling a training vector x uniformly at random, a number $d \in \{1..D\}$ and a set $o_{<d}$ of conditioned variables uniformly at random, and by computing $\widehat{\mathscr{I}_{\text{OA}}}(\theta)$ as in Equation 2. The sum in $\widehat{\mathscr{I}_{\text{OA}}}(\theta)$ is computed over all possible choices of o_d, and the neural network computations involved in $p(x_{o_d}|x_{o<d}; \theta, o_d)$ can be reused across the terms of the sum.

$$\mathscr{I}_{\text{OA}}(\theta) = \mathbb{E}_{x \in X} \mathbb{E}_{o \in D!} - \log p(x; \theta, o) = \mathbb{E}_{x \in X} \mathbb{E}_{o \in D!} - \sum_{d=1}^{D} \log p(x_{o_d}|x_{o<d}; \theta, o) \qquad (1)$$

$$\approx \widehat{\mathscr{I}_{\text{OA}}}(\theta) = \frac{D}{D-d+1} \sum_{o_d} - \log p(x_{o_d}|x_{o<d}; \theta, o_{<d}, o_d) \qquad (2)$$

In this section, we present and motivate step-by-step the modifications we propose to the OA procedure (Equation 1), leading to the OA++ procedure (Equation 5).

2.1 Equal use of 1D conditional distributions at training and inference

Ideally, the usage of each 1D conditional distribution at inference time should be proportional to its usage at training time. Otherwise, some 1D conditional distributions frequently used at inference time might be under-trained (undermining performance), and some 1D conditional distributions rarely used at inference time might be over-trained (undermining efficiency). For $d \in \{1..D\}$, we refer to 1D conditional distributions with $d - 1$ conditioned variables, $p(x_{o_d}|x_{o<d}; \theta)$, as 1D conditional distributions of size d.

Result: Under the OA procedure, at any training iteration, each 1D conditional distribution of size d has a fixed probability $\frac{1}{\binom{D}{d-1}(D-d+1)}$ of being trained.

Proof: At any training iteration, exactly one 1D conditional distribution of size d is trained. It is chosen uniformly at random among the $\binom{D}{d-1}(D - d + 1)$ 1D conditional distributions of size d.

Result: Under the OA procedure, each 1D conditional distribution of size d has probability $\frac{1}{\binom{D}{d-1}(D-d+1)}$ of being involved in any *generation* query.

Proof: Exactly one 1D conditional distribution of size d is involved in any *generation* query, as each of the D variables must be sampled in sequence. There are $\binom{D}{d-1}(D - d + 1)$ such distributions. The OA procedure chooses a variable ordering uniformly at random. Thus, any 1D conditional distribution of size d has an equal probability $\frac{1}{\binom{D}{d-1}(D-d+1)}$ of being used in any *generation* query.

Therefore, under the OA procedure, the expected usage of any 1D conditional distribution at inference time is proportional to its expected usage at training time for *generation* queries. We are interested in data *completion* queries, however. In a completion query, the values of some variables are known, and thus conditional distributions for those variables do not need to be computed. This changes the expected usage patterns for each size d of conditional distribution. Using the OA procedure leads to a discrepancy between the usage of 1D conditional distributions during training and inference. Further, if we know something in advance about the distribution of inference queries (e.g. which variables will be known, or how many will be known), the OA procedure has no way to exploit such prior knowledge.

The OA++ procedure we propose does not suffer from these limitations. It assumes that inference queries follow a distribution $\mathscr{D}$, and it trains all 1D conditional distributions proportionally to their expected usage during inference under this distribution. If we have prior knowledge about the expected structure of inference queries, this can be encoded in $\mathscr{D}$. If we have no such prior knowledge, then OA++ sets $\mathscr{D}$ to be a uniform distribution over inference queries, i.e. the set of observed variables in a completion query is drawn uniformly at random.

Instead of optimizing $\log p(x; \theta, o)$ over all D! orderings of $\{1..n\}$ as in the OA procedure, OA++ samples $\mathbf{obs} \subset \{1..D\}$ from the expected distribution of inference queries $\mathscr{D}$, it samples an ordering o of the unobserved input $x^{\mathbf{missing}}$ uniform at random, and it optimizes $\log p(x^{\mathbf{missing}}|x^{\mathbf{obs}}; \theta, o)$. Since $\mathbf{missing} \cup \mathbf{obs} = \{1..D\}$, there are $(D - |\mathbf{obs}|)!$ possible orderings o of the unobserved input. The corresponding loss is expressed in Equation 3:

$$\mathbb{E}_{x \in X} \mathbb{E}_{\mathbf{obs} \sim \mathscr{D}} \mathbb{E}_{o \in (D-|\mathbf{obs}|)!} - \log p(x^{\mathbf{missing}}|x^{\mathbf{obs}}; \theta, o) \tag{3}$$

2.2 Training fewer 1D conditional distributions

The OA procedure maximizes the average log-likelihood of the model over all orderings: in Equation 1, the ordering o is treated as a uniform random variable over its D! values. Consequently, all 1D conditional distributions are trained. However, these conditional distributions will not all be used. To handle any given query, the model must either fix an order of the unobserved variables, or use an ensemble of $K > 1$ orderings of the unobserved variables as in [5]. In most settings, $K << D!$, thus far fewer 1D conditionals will be used than were trained. Since the parameters of all conditionals are determined by a single shared neural network, the model is wasting its representational capacity on 1D conditionals that will not be used at inference time.

Model	Adult	DNA	Mushrooms	NIPS-0-12	Connect4	OCR-letters	RCV1	Web
OA	9.8 / 13.6	87.2 / 90.7	5.2 / 9.5	277.0 / 280.7	9.4 / 14.7	31.7 37.8	47.0 / 48.1	28.8 / 30.1
OA++	**7.8 / 11.9**	**78.2 / 83.3**	**4.2 / 7.8**	**272.0 / 276.4**	**4.7 / 9.5**	**23.0 / 27.7**	**46.0 / 47.2**	**27.9 / 29.1**

Table 1: Average negative log-likelihood of NADE models on 2 test sets of inference queries. In each cell, the left metric corresponds to uniform random queries, the right metric corresponds to random queries of size D/2 known in advance

The OA++ procedure we propose instead trains at most $K \sum_{d=1}^{D} \binom{D}{d-1}$ conditional distributions, much fewer than the total number $\sum_{d=1}^{D} \binom{D}{d-1}(D-d+1)$ of 1D conditional distributions, and it can still handle any completion query by using an ensemble of K orderings. To do so, we fix in advance K orderings $O_K = \{o_1, ..., o_K\}$ of $\{1..D\}$. For any completion query $x^{\textbf{obs}}$, an ordering o is sampled uniform at random from O_K. The autoregressive sum is computed over the unobserved input $x^{\textbf{missing}}$ (data dimensions are ordered according to o). Fundamentally, OA++ treats the ordering o as a uniform random variable over K values, instead of $D!$ values for OA.

The OA++ loss $\mathscr{I}_{\text{OA++}}(\theta)$ for data completion is expressed as in Equation 5. $\mathscr{D}$ is the distribution of inference queries if we have prior knowledge of it; otherwise $\mathscr{D}$ is the uniform random distribution of completion queries. K is the maximum number of inference queries that we expect to average by ensembling at inference time. For a given completion query $\textbf{obs} \in \mathscr{D}$, the $D - |\textbf{obs}|$ data dimensions of $x^{\textbf{missing}}$ are reordered according to ordering o.

$$\mathscr{I}_{\text{OA++}}(\theta) = \mathbb{E}_{x \in X} \mathbb{E}_{o \in O_K} \mathbb{E}_{\textbf{obs} \sim \mathscr{D}} - \log p(x^{\textbf{missing}} | x^{\textbf{obs}}; \theta, o) \tag{4}$$

$$= \mathbb{E}_{x \in X} \mathbb{E}_{o \in O_K} \mathbb{E}_{\textbf{obs} \sim \mathscr{D}} \sum_{d=1}^{D-|\textbf{obs}|} - \log p(x_d^{\textbf{missing}} | x^{\textbf{obs}}, x_{<d}^{\textbf{missing}}; \theta, o) \tag{5}$$

Unbiased Estimator The OA++ loss in 5 has a large number of terms. It can be estimated by sampling a training example x uniformly at random, a completion query $\textbf{obs} \sim \mathscr{D}$, a number $d \in \{1..D - |obs|\}$, and by computing $\widehat{\mathscr{I}_{\text{OA++}}}(\theta)$ as in Equation 6.

$$\widehat{\mathscr{I}_{\text{OA++}}}(\theta) = -(D - |\textbf{obs}|) \log p(x_d^{\textbf{missing}} | x^{\textbf{obs}}, x_{<d}^{\textbf{missing}}; \theta, o) \tag{6}$$

It is possible to provide an unbiased estimator of the loss, that consists of a sum with computations shared among its different terms, like in Equation 2. However, under the assumption $K << D!$, the sum would likely consist of a single term, effectively reducing to Equation 6.

Unifying data generation and data completion
By choosing $K = 1$, OA++ reduces to the original training procedure for NADEs introduced in [2]. By choosing $K = D!$, OA++ reduces to the order-agnostic training procedure for data *generation* introduced in [6]. In other words, OA++ unifies the training procedures of data generation and data completion under a single framework.

3 Results

In order to compare OA and OA++, we train the same autoregressive model (two-layer NADE) with both procedures. We conduct experiments on eight multivariate binary datasets commonly used by previous works on autoregressive models [1, 2, 5]. Table 1 reports the performance of all models on 2 test sets of inference queries: one consists of uniform random completion queries; the other consists of completion queries of size $D/2$ picked at random. On the second test set, models were provided prior knowledge of the distribution of inference queries. Results are computed for $K = 1$ (no ensemble learning). The comparison of OA and OA++ for $K > 1$ is left as future work.

Performance at convergence OA++ outperforms OA on all experiments and does especially well when given prior knowledge about the distribution of inference queries.

Speed of convergence OA++ converges in fewer computations than OA. OA++ also does not suffer from overfitting, while OA sometimes does. Figure 1 reports the evolution of the training and validation log-likelihoods with the number of computations, for some experiments.
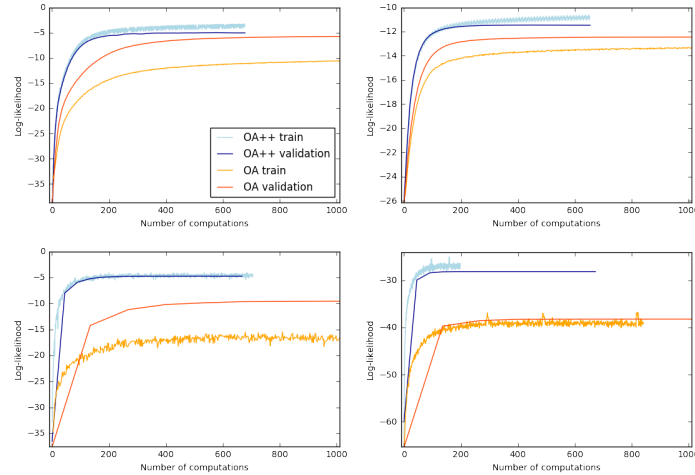
Figure 1: Evolution of the train and validation loglikelihoods with the number of computations. Computations are reported as the number of neural network inferences performed, divided by the dimensionality D of the data. From left to right and from top to bottom: Mushrooms, Adult, Connect4, OCRletters.

References

[1] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 881–889, 2015.

[2] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[3] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[4] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[5] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

[6] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pages 467–475, 2014.

[7] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.