Neural Autoregressive Distribution Estimation

John Thickstun

Recall that our goal is to learn a collection of conditional probability distributions $p(x_t|x_{<t})$. We previously saw linear autoregressive models, where $x_t \sim \mathcal{N}(f(x_{<t}), \sigma^2)$, where $f(x_{<t})$ is a linear function of the past, for example $f(x_{<t}) = \rho(x_{t-1} - \mu) + \mu$. The next step we will take is to replace linear functions with more expressive non-linear functions, parameterized by neural networks. While the history of this modeling strategy is older than most of the ideas we discuss in this course, specialized and computationally-intensive neural parameterizations of autoregressive models are directly responsible for the latest generative modeling results in NLP [Radford et al., 2019, Brown et al., 2020] as well as modern work on audio modeling [Oord et al., 2016a]

Fully-Visible Sigmoid Belief Networks

Non-linear parameterizations f allow us to bridge the gap between continuous and discrete modeling. The fully-visible sigmoid belief network [Neal, 1992] is a simple example of a non-linear parameterization of models over binary sequences $x \in \{0,1\}^T$, which can be loosely interpreted as an extension of the linear autoregressive model. The idea is to predict $x_t \in \{0,1\}$ using a log-linear combination of $x_{<t}$. For now, we assume all our sequences have constant length T. Recall that $\text{sigmoid}(u) = 1/(1+e^{-u})$.

Definition 1. *Let $P \in \mathbb{R}^T$ and define $f_t : \mathbb{R}^{(t-1)} \to \mathbb{R}$ by $f_t(x_{<t}) = \langle P_{<t}, x_{<t}\rangle$. The **fully-visible sigmoid belief network** is given by*

$$\begin{aligned} x_t &\sim \text{Bernoulli}\left(\text{sigmoid}(f_t(x_{<t}))\right), \\ x_1 &\sim \text{Bernoulli}(P_0). \end{aligned}$$

This setup is a little more sophisticated than the simplest possible parameterization: we share the weights at each location in the history, versus simply parameterizing each conditional distribution $p(x_t|x_{<t})$ with its own set of $t-1$ weights. Another way to think about this model (and a convenient way to implement it) is that we mask the future $x_{\geq t}$ in the input sequence for our prediction of x_t. Formally, we introduce a mask $\mathbf{m} \in \{0,1\}^{T\times T}$ defined by $\mathbf{m}_{t,s} = \mathbf{1}_{s<t}$, and it follows (where $\odot$ denotes element-wise multiplication) that

$$f_t(x_{<t}) \equiv \langle P_{<t}, x_{<t}\rangle = \langle P, \mathbf{m}_t \odot x\rangle.$$

For discrete time series $\mathbf{w} \in \mathcal{V}^T$, we can replace the Bernoulli distribution with a Categorical distribution and parameterize the model with a softmax activation

$$\text{softmax}(\mathbf{u}) = \frac{e^{\mathbf{u}}}{\sum_{k=1}^d e^{\mathbf{u}_k}}.$$

We also need to encode the history $\mathbf{w}_{<t}$ by embedding it in Euclidean space, in order to define a log-linear model. A popular way to do this is one-hot encoding: we assign each item in $\mathcal{V}$ an index $0 \leq w < d = |\mathcal{V}|$ and represent w as one-hot vector $x \in \mathbb{R}^d$ such that $x_k = \mathbf{1}_{k=w}$.

Definition 2. *Let* $\boldsymbol{P} \in \mathbb{R}^{T \times d \times d}$ *and define* $f_t : \mathbb{R}^{(t-1) \times d} \to \mathbb{R}^d$ *by*

$$f_t(\boldsymbol{x}_{<t}) = \sum_{s<t} \boldsymbol{P}_s \boldsymbol{x}_s.$$

The ***fully-visible softmax belief network*** *is given by*

$$\begin{aligned} \boldsymbol{w}_t &\sim \text{Categorical}\left(\text{softmax}\left(f_t(\boldsymbol{x}_{<t})\right)\right), \\ \boldsymbol{w}_1 &\sim \text{Categorical}(\boldsymbol{P}_{0,0}). \end{aligned}$$

These fully-visible sigmoid and softmax belief networks can be interpreted as sequential variants of logistic and multinomial logistic regression respectively. We can learn the weights $\mathbf{P}$ by approximating the maximum likelihood estimator via optimization of the cross-entropy loss

$$\frac{1}{n} \sum_{i=1}^{n} -\log p(\mathbf{w}^i) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T} -\log p(\mathbf{w}_t^i | \mathbf{w}_{<t}^i).$$

Neural Autoregressive Distribution Estimation

Written this way, it becomes clear how to generalize to neural parameterizations: replace the log-linear model with $\text{softmax}(f_{\theta,t}(\mathbf{x}_{<t}))$, where the function $f_{\theta,t} : \mathbb{R}^{(t-1)d} \to \mathbb{R}^d$ is defined by a neural network with weights θ. This possibility was observed at least as early as Bengio and Bengio [2000], and applied to language modeling in Bengio et al. [2003]. The modern version of this idea is called the Neural Autoregressive Distribution Estimator (NADE) [Larochelle and Murray, 2011]. Specifically, Larochelle and Murray [2011] propose a fully connected parameterization with k hidden nodes defined by

$$\begin{aligned} \mathbf{h}_t &= \text{sigmoid}\left(\mathbf{c}_t + \sum_{s<t} \mathbf{V}_s \mathbf{x}_s\right), & \mathbf{V} \in \mathbb{R}^{T \times k \times d}, \mathbf{c} \in \mathbb{R}^{T \times k}, \\ f_{\theta,t}(\mathbf{x}_{<t}) &= \mathbf{b}_t + \sum_{s \leq t} \mathbf{W}_s \mathbf{h}_s, & \mathbf{W} \in \mathbb{R}^{T \times d \times k}, \mathbf{b} \in \mathbb{R}^{T \times d}. \end{aligned}$$

The weights θ in this case consist of the quantities listed on the right-hand side of the equations.

Although we introduced this parameterization for discrete time-series, it can be directly extended to continuous time-series. All we need to do is, instead of using the outputs $f_{\theta,t}$ to parameterize a categorical distribution, use them to parameterize a continuous distribution over the conditionals $p(\mathbf{x}_t | \mathbf{x}_{<t})$, as we did using a Gaussian distribution in the linear autoregressive setting. For more expressive models, we could use $f_{\theta,t}$ to parameterize e.g. a mixture of Gaussians. This idea is developed under the name real-valued NADE (RNADE) [Uria et al., 2013].

Parameterizing Your NADE

One potential modification of the NADE architecture weight-tying. Observe that the weight tensors $\mathbf{V}$ and $\mathbf{W}$ have the same shape, simply transposing the second and third axes. Weight-tying uses

the same weights for both layers of the NADE, replacing $\mathbf{V}_s$ with $(\mathbf{W}_s)^T$. This weight-tying idea was proposed in the original NADE paper [Larochelle and Murray, 2011], and a generalization of the idea has proven highly effective in NLP domain [Press and Wolf, 2017]. For a modern implementation of NADE, we might also consider replace the hidden-state sigmoid activation with a ReLU [Krizhevsky et al., 2012] or more sophisticated non-linearity.

Beyond these minor tweaks, more interesting extensions of NADE exploit the machinery of deep learning to better fit the data distribution. One way to do this is to stack multiple hidden layers between the inputs and outputs of the NADE. For example, an L-layer NADE with hidden layers h_ℓ ($\ell \in \{1, \ldots, L\}$) using weight-tying and ReLU non-linearities could look like

$$\begin{aligned}
\mathbf{h}_{0,t} &= \text{ReLU}\left(\mathbf{c}_{0,t} + \sum_{s<t} (\mathbf{W}_s)^T \mathbf{x}_s\right), & \mathbf{W} \in \mathbb{R}^{T\times d\times k}, \mathbf{c}_0 \in \mathbb{R}^{T\times k},\\
\mathbf{h}_{\ell,t} &= \text{ReLU}\left(\mathbf{c}_{\ell,t} + \sum_{s\leq t} \mathbf{V}_{\ell,s}\mathbf{h}_{\ell-1,s}\right), & \mathbf{V} \in \mathbb{R}^{(L-1)\times T\times k\times k}, \mathbf{c} \in \mathbb{R}^{(L-1)\times T\times k},\\
f_{\theta,t}(\mathbf{x}_{<t}) &= \mathbf{b}_t + \sum_{s\leq t} \mathbf{W}_s \mathbf{h}_{L,s}, & \mathbf{b} \in \mathbb{R}^{T\times d}.
\end{aligned}$$

This deep NADE is developed in detail by Uria et al. [2016]. Both Uria et al. [2016] and Oord et al. [2016b] develop the idea of a convolutional NADE, which exploits the convolutional neural network architecture [LeCun et al., 1989]. An efficient implementation of the partial sums $s \leq t$ using masks is developed in detail under the name MADE (masked autoregressive distribution estimation) [Germain et al., 2015].

One downside of NADE (and deep NADE) is that the number of parameters in the model is a (linear) function of the length T of the sequences we are modeling. If our sequences have a variable length, then the number of parameters will grow linearly in the length of the longest sequence. This is undesirable from a learning perspective, because these models are likely to overfit. The recurrent neural network (RNN) [Rumelhart et al., 1986] is a clever way to construct a neural architecture with parameter counts that don't scale with the length of the sequence. A simple variant of the RNN is defined by

$$\begin{aligned}
\mathbf{h}_0 &= \mathbf{W}_{\text{init}}, & \mathbf{W}_{\text{init}} \in \mathbb{R}^k,\\
\mathbf{h}_t &= \text{sigmoid}(\mathbf{c} + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t), & \mathbf{W}_h \in \mathbb{R}^{k\times k}, \mathbf{W}_x \in \mathbb{R}^{k\times d}, \mathbf{c} \in \mathbb{R}^k,\\
f_{\theta,t}(\mathbf{x}_{<t}) &= \mathbf{b} + \mathbf{W}_o \mathbf{h}_{t-1}, & \mathbf{W}_o \in \mathbb{R}^{d\times k}, \mathbf{b} \in \mathbb{R}^d.
\end{aligned}$$

The idea is to maintain a running state $\mathbf{h}_t$; at every time step t, we construct the new state $\mathbf{h}$ as a non-linear function of the previous state $\mathbf{h}_{t-1}$ and the token $\mathbf{x}_t$. The particular non-linearity is non-essential, and can be replaced with e.g. a ReLU (although if we use sigmoid, there is an interesting interpretation of the hidden state as "bits"). And of course, analogous to the deep NADE, we can stack multiple recurrent layers together to construct a deep RNN with hidden nodes $\mathbf{h}_{\ell,t}$. These ideas are explored in depth by Boulanger-Lewandowski et al. [2012], using the popular long short-term memory (LSTM) variant [Hochreiter and Schmidhuber, 1997, Gers et al., 2002] of the RNN architecture.

Exposure Bias

One prevalent concern about neural autoregressive models like NADE is a phenomenon known as exposure bias. This refers to the fact that we train our models to fit the conditionals $p(x_t|x_{<t})$, but we sample from $p(x_t|\tilde{x}_{<t})$ where $\tilde{x}_{<t}$ are samples from our estimated conditionals $\tilde{x}_s \sim \hat{p}(\cdot|x_{<s})$, not the true conditionals $x_s \sim p(\cdot|x_{<s})$. There is a line of work called scheduled sampling that attempts to adjust for this discrepancy by replacing some of the tokens in the history with outputs from the model during learning [Bengio et al., 2015, Ranzato et al., 2016, Zhang et al., 2019]. This work borrows from intuitions developed in the reinforcement learning setting for robotics, notably dataset aggregation (DAGGER) [Ross et al., 2011] and data as demonstrator (DaD) [Venkatraman et al., 2015].

It remains unclear whether these techniques are promising. Exposure bias corrections typically lead to inconsistent estimators, and rely on careful tuning in the optimization process to avoid degeneracy. One way to think about these techniques pragmatically is that they provide another regularization knob (or data augmentation) that can be tuned via hyper-parameter search to improve generalization, at increased computational cost. To some extent, it seems that these problems can be ignored simply by building better generative models; for a sufficiently strong generative model, the mismatch between training sequences and generated sequences will become negligible and can be ignored. These techniques are not used, for example, to train state of the art language models [Radford et al., 2019, Brown et al., 2020]. Nevertheless, even these powerful models may still be susceptible to exposure bias. Diagnostic work for modern language models that quantifies a form of exposure bias using calibration statistics and suggests a way to correct it is presented in Braverman et al. [2020]

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015. (document)

Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems*, 2000. (document)

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003. (document)

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. 2012. (document)

Mark Braverman, Xinyi Chen, Sham M Kakade, Karthik Narasimhan, Cyril Zhang, and Yi Zhang. Calibration, entropy rates, and memory in language models. *International Conference on Machine Learning*, 2020. (document)

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. (document)

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015. (document)

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 2002. (document)

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. (document)

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. (document)

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics*, 2011. (document)

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989. (document)

Radford M Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 1992. (document)

Aaron van den Oord, 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 preprint arXiv:1609.03499*, 2016a. (document)

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning*, 2016b. (document)

Ofir Press and Lior Wolf. Using the output embedding to improve language models. *Conference of the European Chapter of the Association for Computational Linguistics*, 2017. (document)

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. (document)

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *International Conference in Learning Representations*, 2016. (document)

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011. (document)

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 1986. (document)

Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, 2013. (document)

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 2016. (document)

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI Conference on Artificial Intelligence*, 2015. (document)

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. *Annual Meeting of the Association for Computational Linguistics*, 2019. (document)