Notes on NADE

Alvin Chua

August 11, 2015

1 Introduction

The NADE (neural autoregressive distribution estimator) model acts on binarized ordered inputs $v = (v_1, v_2, \cdots, v_n)$, where $v_i \in \{0, 1\}$; and gives an estimate for its probability $p(v)$ after it is trained on examples. The model is autoregressive, which means that we estimate $p(v)$ from a series of conditional probabilities $p(v_i|v_{<i})$, where $i \in \{1, \cdots, n\}$ and $v_{<i} = (v_1, v_2, \cdots, v_{i-1})$ are all inputs before v_i. It is also useful to specify that $p(v_1|v_{<1}) = p(v_1)$. The value of $p(v)$ is related to $p(v_i|v_{<i})$ by probability chain rule

$$p(v) = \prod_{i=1}^{n} p(v_i|v_{<i}), \tag{1}$$

or similarly in terms of log probabilities

$$-\ln p(v) = -\sum_{i=1}^{n} \ln p(v_i|v_{<i}). \tag{2}$$

2 A Fully Visible Model

To understand NADE, it is useful to first discuss the fully visible sigmoid belief network. Here, we treat each each $p(v_i|v_{<i})$ as a sigmoid unit such that

$$\begin{aligned} p_i &= \sigma\Big(c_i + \sum_{j=1}^{n} W_{ij} v_j\Big) \\ &= \sigma\Big(c_i + \sum_{j<i} W_{ij} v_j\Big), \end{aligned}$$

where $\sigma(x)$ is an activation function, and the autoregressivity is enforced by requiring that W_{ij} is a lower triangular $n \times n$ matrix such that $W_{ij} = 0$ whenever $j \geq i$. As such, c_i contributes n parameters and W_{ij} contributes $n(n-1)/2$ to the model. The value p_i represents the probability $p(v_i = 1|v_{<i})$, and conversely, $p(v_i = 0|v_{<i}) = (1 - p_i)$. We can write this more succinctly as

$$p(v_i|v_{<i}) = p_i^{v_i}(1 - p_i)^{1-v_i}, \tag{3}$$

or similarly in terms of log probabilities

$$\ln p(v_i|v_{<i}) = v_i \ln p_i + (1 - v_i) \ln(1 - p_i), \tag{4}$$

where we have used the property that v_i is a Bernoulli random variable that only admits $v_i \in \{0, 1\}$.

3 NADE

The NADE model estimates conditional probabilities $p(v_i|v_{<i})$ by introducing hidden units. We will only consider the simplest case where each term $p(v_i|v_{<i})$ is associated with a single sub-block comprising m hidden units. We require that the number of hidden units must be at most the same as the number of visible units, $m \leq n$. Allowing more hidden units introduces redundant structure to the network.

The hidden units are denoted by $h^{(i)} = \left(h_1^{(i)}, \cdots, h_m^{(i)}\right)$, where the superscript i refers to a particular hidden layer sub-block that we use to estimate p_i. We will first discuss the unconstrained network and thereafter invoke the autoregressivity property to inform us of how we can constrain the network to insert structure and reduce the dimensionality of the network. The unconstrained equations with a one hidden layer are

$$p_i = \sigma\Big(b^{(i)} + \sum_{j=1}^{m} V_j^{(i)} h_j^{(i)}\Big) \text{ and} \tag{5}$$

$$h_j^{(i)} = \sigma\Big(c_j^{(i)} + \sum_{k=1}^{n} W_{jk}^{(i)} v_k\Big), \tag{6}$$

where $c_j^{(i)}$ are biases associated with each of the hidden units j and model i and $W_{jk}^{(i)}$ are weights, each comprising an $m \times n$ matrix. Similarly, the hidden units are linked to the output by the weights $V_j^{(i)}$ and biases $b^{(i)}$. These are in turn used as inputs to estimate p_i.

We constrain the network by requiring autoregressivity. This manifests in tied weights that link the inputs v_i to the hidden units in sub-blocks $h_j^{(i)}$. In particular, the autoregressive property suggests that all values with the superscript i are dependent only on the inputs $v_{<i}$. As such, we can set $W_{jk}^{(i)} = 0$ whenever $k \geq i$. Furthermore, we require that weights linking input nodes to equivalent structural nodes in each sub-block of hidden units are tied. More precisely, we can express this by requiring $W_{jk}^{(i)} = W_{jk}^{(l)}$ for all j, $i < l$ and $k < i$. In doing this, we can now construct all weight matrices $W_{jk}^{(i)}$, where $i < n$ and n is the number of visible units by setting

$$W_{jk}^{(i)} = \begin{cases} W_{jk} & k < i \text{ and} \\ 0 & \text{otherwise} \end{cases},$$

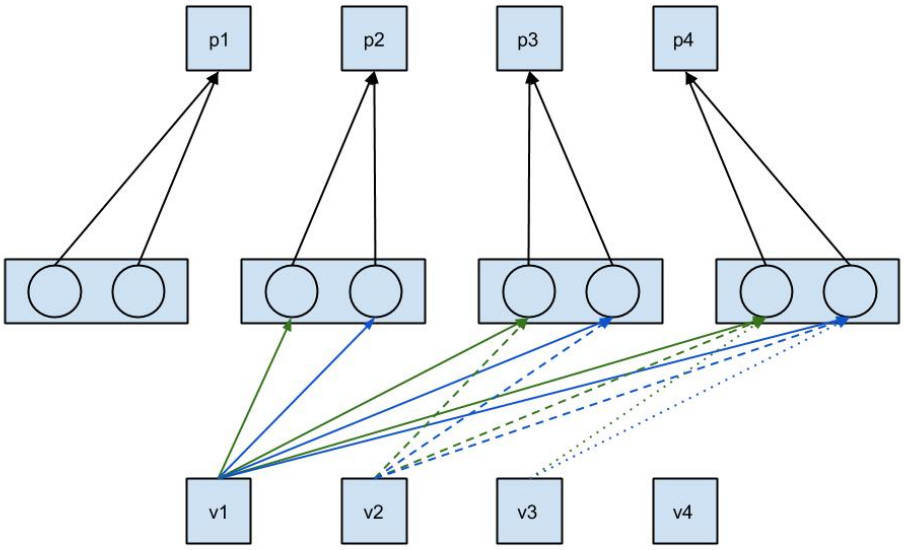

Figure 1: A NADE network with 4 visible and 2 hidden units. Common colours and line styles linking the input to hidden units indicate tied weights.

where drop the superscript and write $W_{jk} = W_{jk}^{(n)}$. In the same spirit, we will also share biases across hidden units by requiring $c_j^{(i)} = c_j$ for all i. The NADE model has these adjustments that introduce autoregressivity. The equation linking the inputs to the hidden units is now

$$h_j^{(i)} = \sigma\Big(c_j + \sum_{k=1}^{i-1} W_{jk} v_k\Big), \tag{7}$$

where equation (7) replaces the unconstrained equation (6). Figure (1) illustrates the NADE model graphically.

We can now estimate the density of $p(v)$ by first calculating $h_j^{(i)}$ and subsequently p_i, and folding this into $p(v)$ using equations (3) and (1). Alternatively, we often find it more convenient to calculate log probability $\ln p(v)$. To do this, we would use equations (4) and (2) instead.

Gradients of the loss function with respect to each of the parameters are used in the back propagation algorithm. We choose the negative log probability $-\ln p(v)$ as the loss function. We can first break the calculation up using the

chain rule with

$$\begin{aligned} -\frac{\partial \ln p(v)}{\partial x} &= -\sum_{i=1}^{n} \frac{\partial}{\partial x}\Big(v_i \ln p_i(x) + (1-v_i)\ln(1-p_i(x))\Big) \\ &= \sum_{i=1}^{n} \frac{p_i - v_i}{p_i(1-p_i)} \frac{\partial p_i(x)}{\partial x}, \end{aligned}$$

where x is a variable associated with the top layer of the network and δ_{ij} is the dirac-δ function

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

We note that to back propagate derivites to subsequent layers we will have to expand the $\partial p/\partial x$ term further.

We require gradients for each of the parameters in the network and denote these by $\Delta b^{(i)}$, $\Delta V_j^{(i)}$, Δc_i and ΔW_{ij}. As an intermediate step, it is also useful to compute gradients associated with the hidden unit $\Delta h_j^{(i)}$. Finally we also specify that the activation function σ for this model is the sigmoid function, for which we can use the identity $d\sigma(y)/dx = \sigma(1-\sigma)dy/dx$.

We begin the calculation with

$$\begin{aligned} \Delta b^{(i)} &= -\frac{\partial \ln p(v)}{\partial b^{(i)}} \\ &= \sum_{j=1}^{n} \frac{p_j - v_j}{p_j(1-p_j)} \frac{\partial p_j}{\partial b^{(i)}} \\ &= \sum_{j=1}^{n} (p_j - v_j)\delta_{ij} \\ &= p_i - v_i, \end{aligned}$$

and

$$\begin{aligned} \Delta V_j^{(i)} &= -\frac{\partial \ln p(v)}{\partial V_j^{(i)}} \\ &= \sum_{k=1}^{n} \frac{p_k - v_k}{p_k(1-p_k)} \frac{\partial p_k}{\partial V_j^{(i)}} \\ &= \sum_{k=1}^{n} \sum_{l=1}^{m} (p_k - v_k) h_l^{(k)} \delta_{ik}\delta_{jl} \\ &= (p_i - v_i) h_j^{(i)} = \Delta b^{(i)} h_j^{(i)}. \end{aligned}$$

The calculation of $\Delta h_j^{(i)}$ is similar to that for $\Delta V_j^{(i)}$ with

$$\Delta h_j^{(i)} = -\frac{\partial \ln p(v)}{\partial h_j^{(i)}} = \Delta b^{(i)} V_j^{(i)}.$$

We can now calculate gradients associated with the hidden layer

$$\begin{aligned}
\Delta c_i &= -\frac{\partial \ln p(v)}{\partial c_i} \\
&= -\sum_{j=1}^{n}\sum_{k=1}^{m} \frac{\partial \ln p(v)}{\partial h_k^{(j)}} \frac{\partial h_k^{(j)}}{\partial c_i} \\
&= \sum_{j=1}^{n}\sum_{k=1}^{m} \Delta h_k^{(j)} \frac{\partial h_k^{(j)}}{\partial c_i} \\
&= \sum_{j=1}^{n}\sum_{k=1}^{m} \Delta h_k^{(j)} h_k^{(j)}(1-h_k^{(j)})\delta_{ik} \\
&= \sum_{j=1}^{n} \Delta h_i^{(j)} h_i^{(j)}(1-h_i^{(j)}),
\end{aligned}$$

and

$$\begin{aligned}
\Delta W_{ij} &= -\frac{\partial \ln p(v)}{\partial W_{ij}} \\
&= \sum_{k=1}^{n}\sum_{l=1}^{m} \Delta h_l^{(k)} \frac{\partial h_l^{(k)}}{\partial W_{ij}} \\
&= \sum_{k=1}^{n}\sum_{l=1}^{m} \Delta h_l^{(k)} h_l^{(k)}(1-h_l^{(k)}) \frac{\partial}{\partial W_{ij}} \sum_{q=1}^{k-1} W_{lq} v_q \\
&= \sum_{k=1}^{n}\sum_{l=1}^{m}\sum_{q=1}^{k-1} \Delta h_l^{(k)} h_l^{(k)}(1-h_l^{(k)}) v_q \delta_{il}\delta_{jq} \\
&= \sum_{k=1}^{n}\sum_{q=1}^{k-1} \Delta h_i^{(k)}(1-h_i^{(k)}) h_i^{(k)} v_q \delta_{jq} \\
&= \sum_{q=1}^{n}\sum_{k=q+1}^{n} \Delta h_i^{(k)}(1-h_i^{(k)}) h_i^{(k)} v_q \delta_{jq} \\
&= \sum_{k=j+1}^{n} \Delta h_i^{(k)}(1-h_i^{(k)}) h_i^{(k)} v_j,
\end{aligned}$$

where we have used the trick

$$\sum_{i=1}^{n}\sum_{j=1}^{i-1} a_i b_j = \sum_{j=1}^{n}\sum_{i=j+1}^{n} a_i b_j$$

to exchange summations. As seen in these equations, to calculate gradients we must first do a forward pass with v_i to compute values for both conditional probabilities p_i as well as values for the the hidden layer $h_j^{(i)}$.

Errors and improvements to the Larochelle and Murray paper: The Neural Autoregressive Distribution Estimator

In the Algorithm 1,

- $p(v) \leftarrow 0$ should read $p(v) \leftarrow 1$.
- $p(v) \leftarrow p(v)\big(p(v_i = 1|\mathbf{v}_{<i})^{v_i} + (1 - p(v_i = 1|\mathbf{v}_{<i}))^{1-v_i}\big)$ should read $p(v) \leftarrow p(v)\big(p(v_i = 1|\mathbf{v}_{<i})^{v_i}(1 - p(v_i = 1|\mathbf{v}_{<i}))^{1-v_i}\big)$.
- Calculating $\delta\mathbf{c}$ is redundant. Just calculate $\delta\mathbf{a}$ within the loop as specified and after that set $\delta\mathbf{c} \leftarrow \delta\mathbf{a}$.
- The term $(\delta\mathbf{h}_i)\mathbf{h}_i(1 - \mathbf{h}_i)$ refers to a element by element product given the index i.

4 kNade

The NADE model is a fully autoregressive model. We now consider the case where we impose a k-Markov condition in place of full autoregressivity. That is, we will assume that $p(v_i|v_{<i}) \approx p(v_i|v_{i-1}v_{i-2}\cdots v_{i-k})$. The joint probability distribution is now given by

$$-\ln p(v) = -\sum_{i=1}^{n} \ln p(v_i|v_{i-1}v_{i-2}\cdots v_{i-k}), \tag{8}$$

where we ignore the variable v_i whenever $i \leq 0$. The construction of the kNADE model proceeds similarly and the only difference is in the lowest layer, linking the visible units to the first layer of hidden units. The gradient calculation for the bias Δc_i is identical, but the calculation for the weight now reads

$$\Delta W_{ij} = \sum_{q=j+1}^{j+1+k} \Delta h_i^{(q)}(1 - h_i^{(q)})h_i^{(q)}v_j.$$

Acknowledgements

Thanks to Sébastien Racaniere for his helpful comments.