# MAF: Masked Autoregressive Flow for Density Estimation

Lorne Whiteway
lorne.whiteway@star.ucl.ac.uk

Astrophysics Group
Department of Physics and Astronomy
University College London

27 April 2020

Find the presentation at https://tinyurl.com/y72sf93t

# Context

- *Supervised machine learning* i.e. selecting from a highly-parameterised family of non-linear functions to fit some training data.
- Fitting is done to optimise a utility function that rewards a close match to the training data and penalises complexity i.e. enforces *regularization*.
- Regularisation is also provided by any lack-of-flexibility in the fitting functions.
- Bayesian framework: adherence to training data = likelihood; regularisation = prior.

## Density Estimation

- We want to do ML to do density estimation:
- Find the probability density $p$, given a set of training data $\{x_i\}$ that we suppose to be a fair random sample from $p$.

# MAF

- We will discuss the MAF algorithm: Papamakarios et al. 2018; Masked Autoregressive Flow for Density Estimation; https://arxiv.org/abs/1705.07057.
- Based on earlier algorithms of which the most relevant is MADE: Germain et al. 2015; MADE: Masked Autoencoder for Distribution Estimation; https://arxiv.org/abs/1502.03509.
- I will focus first on MADE as all the key ideas are present there.

# Analogy

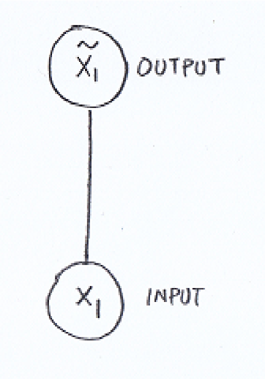
- An *autoencoder* is like a parrot - it learns to mimic.
- An *autoregressive autoencoder* is like a deaf parrot - it learns to play the probabilities (and hence becomes a density estimator).

# Parrot training

- Simple example: single binary outcome. We want the parrot to mimic us when we say 'Yes' or 'No'.
- So give lots of 'Yes', 'No' training data; give the parrot treats but reduce the treats by the non-positive number log(% correctness of the parrot's response). This is called *cross-entropy loss*.

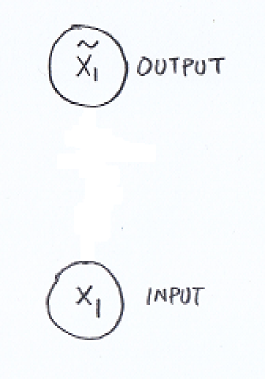
# Parrot training

- The neural network is incentivised to model the identity function $\tilde{x}_1 = x_1$.
- Depending on the flexibility of the neural network the parrot will get this exactly or approximately correct.



# Deaf parrot

- Deaf parrot = no link between input and output.
- Parrot still squawks and gets feedback (more treats if the squawk accidentally matched the unheard training input).
- If say 75% of the training data was 'Yes' then the parrot will learn to say 'Yes' more often. Optimal strategy will be a word that is 75% 'Yes' and 25% 'No'.



# Deaf parrot

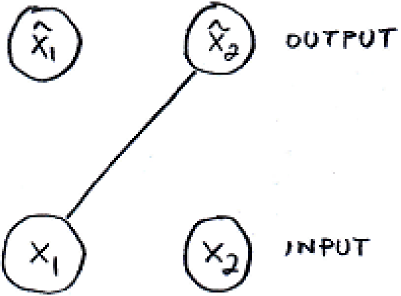
- So by removing links from the neural network but still rewarding the network for mimicry we create a density estimator.

# Two word phrase

- Now consider two word phrases ("Yes, No").
- If there is no correlation between the two words then we just need two deaf parrots.
- But generally $p(x_1 \& x_2) = p(x_1)p(x_2|x_1)$.
- To model this: one deaf parrot for $p(x_1)$ as before. But for the second parrot: unmask its ears, let it hear the training word told to the first parrot, the re-mask its ears for the second word.
- Give treats as before (still encouraging mimicry).
- Easy to show that the second parrot learns $p(x_2|x_1)$.

# Two word phrase

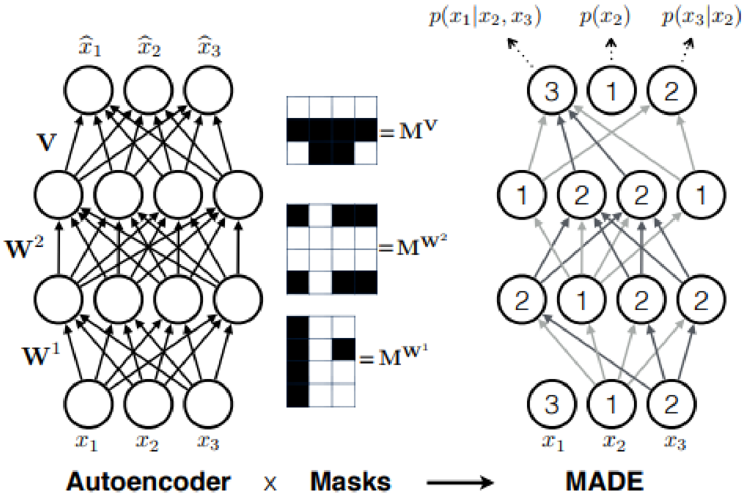
- More generally if $\tilde{x}_i$ is linked only to $x_1, \dots x_{i-1}$ then we learn $p(x_1), \dots, p(x_i | x1 \dots x_{i-1})$, whose product is $p(x)$.



# Autoencoder; Autoregressive autoencoder

- The 'hearing' parrot (that mimics) is an *autoencoder*.
- The network of partially-deaf parrots (that learns the conditional probabilities) is called an *autoregressive autoencoder*.
- Autoregression appears in other contexts e.g. predicting the next element in a causal sequence.

# Figure 1 from MADE paper



# Sampling from the inferred probability density

- Use $(0,0)$ as input to get $(\tilde{x_1}, \tilde{x_2})$ as output;
- Randomly choose $s_1$ in $\{0,1\}$ with probability $\tilde{x_1}$ of being 1;
- Use $(s_1, 0)$ as input to get new $(\tilde{\tilde{x_1}}, \tilde{\tilde{x_2}})$ as output;
- Randomly choose $s_2$ in $\{0,1\}$ with probability $\tilde{\tilde{x_2}}$ of being 1;
- Then $(s_1, s_2)$ is a new sample from the inferred density.
- Takes $D$ evaluations of the net to get one new sample.

# More complicated distributions

- So far we have looked at binary distributions.
- Here the probability density could be described by a single number $(= p(x_i = 1))$. So output $\tilde{x}_i$ has dual meaning as 'best guess at $x_i$' and as '$p(x_i = 1|x_1, \ldots, x_{i-1})$'.

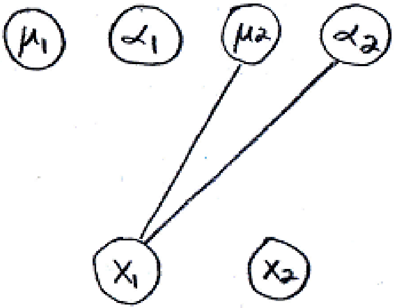
# More complicated distributions

- To handle continuous distributions:
- Model each univariate conditional distribution $p(x_i|x_1, \ldots, x_{i-1})$ as a Gaussian;
- Two output cells for each input: one for mean and one for (log of) standard deviation - from this we easily calculate the conditional probabilities and hence the overall probability of any given input;
- Loss function is now sum of negative logs of probabilities of training data.

# More complicated distributions

$$p(x_1) = N(\mu_1, \exp(\alpha_1))$$

$$p(x_2 | x_1) = N(\mu_2, \exp(\alpha_2))$$



# Sampling from the inferred probability density $\tilde{p}$

- As before we can create new samples with $D$ calls to the neural net.
- At each step we randomly choose $u_i$ from $N(0,1)$ and scale and shift it to get $\tilde{x}_i = \exp(\alpha) * u_i + \mu$.
- At each step the $\alpha$ and $\mu$ come from a call to the net with $\tilde{x}_1, \ldots, \tilde{x}_{i-1}$ as inputs.

# Sampling from the inferred probability density 1

- Thus we get a mapping $f$ from $u \sim N(0, I)$ to $\tilde{x} \sim \tilde{p}$.
- The mapping is easily invertible!
- $u_i = (\tilde{x}_i - \mu)\exp(-\alpha)$
- All the $\mu$ and $\alpha$ that we need are available in **one** call to the net (as here we have all the net input values that we need).
- This gives $f^{-1}$ mapping $\tilde{p}$ to $N(0, I)$.

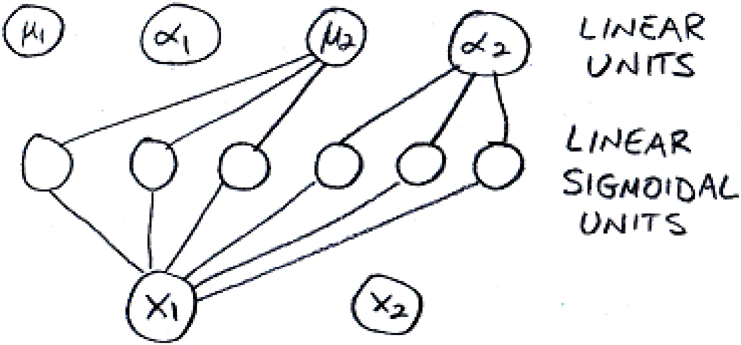
# Sampling from the inferred probability density 2

- Furthermore the autoregressive property means that the Jacobian of $f^{-1}$ is triangular.
- Its diagonal elements are just the $exp(-\alpha_i)$ factors.
- So the determinant of $\mathrm{Jac}(f^{-1})$ is easy to calculate.

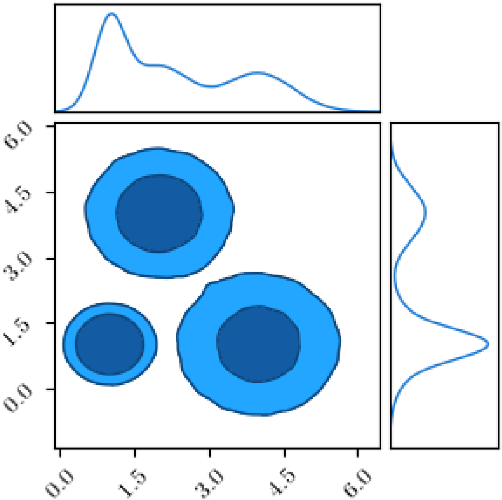
# Normalising flow

- So $f$ maps $N(0, I)$ to $\tilde{p}$.
- Such $f$ is called a normalising flow.
- Recall how probability densities transform:
  $\tilde{p}(x) = \mathcal{N}_{(0,I)}(f^{-1}(x)) \, |\det(\mathrm{Jac}(f^{-1}))|$
- All of these components are easy to calculate, and let us evaluate and sample from $\tilde{p}$.

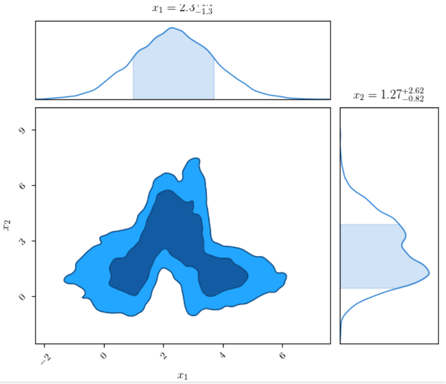
# Example - Neural net



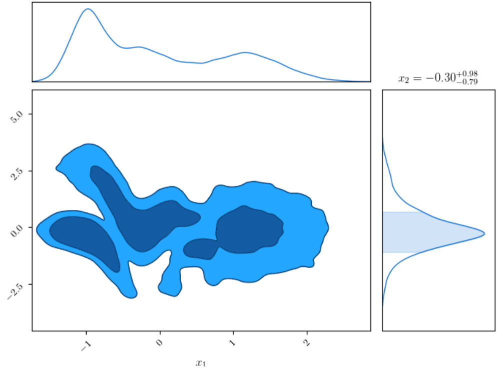
# Example - Target distribution $p$



# Example - inferred distribution $\tilde{p}$



# Inverse image $u = f^{-1}(\tilde{p})$ of inferred distribution - should be $N(0, I)$



# MAF idea

- Map the original training data back through $f^{-1}$ to get $\{u_i\}$ that should be $N(0, I)$ but aren't;
- Repeat the whole process! Treat the $u$ as training data for a new neural net that is trying to model the $u$ as a distorted $N(0, I)$
- Be prepared to do this five or ten iterations.

# MAF - after two iterations $f_2^{-1}(f_1^{-1}(\tilde{p}))$