Smart random initialisation of the FFNN variational parameters

Francesco Calcavecchia

February 21, 2018

When using a randomly initialised FFNN, we found that the final result of a fit or a VMC optimisation heavily depends on the initial random betas. In fact, for some betas the results were extremely poor, demonstrating that the minimisation process must have been stuck somehow. This article tries to make the point about it.

From Jan's investigation it seems that there are two possible sources of troubles:

1. the activation function input is such that the result is always the same. For example: in the logistic function (Fig. 1), if the input is constantly below -5 or above 5 then the output will be flat, which does not share much with our expectations from an activation function;

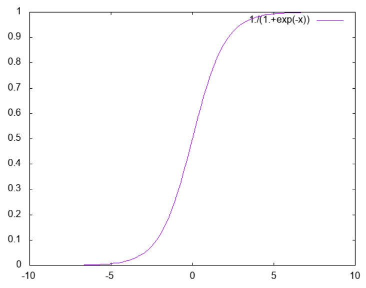


Figure 1: Logistic activation function

2. if the beta are such that two units will result to be the same, there will be a sort of degeneracy that might cause problems.

In the following I suggest how to deal with these two problems.

1 Activation Function Range

Each activation function has an active region, as it is $[-5, 5]$ for the logistic function. Let us denote as Σ the activation function input, which will be the result of a sum $\Sigma = \beta_i x^i$ (Einstein sum's notation). It is simpler to characterise the active region with a mean value μ_Σ and standard deviation σ_Σ. In the case of a normalised flat distribution with range $[a, b]$, this corresponds to:

$$\begin{aligned} \mu_\Sigma &= \frac{a+b}{2} \qquad (1) \\ \sigma_\Sigma &= \frac{b-a}{2\sqrt{3}} \qquad (2) \end{aligned}$$

Since Σ is generated from β and x, we can write:

$$\begin{aligned} \mu_\Sigma &= \langle \beta_i x^i \rangle = \langle \beta_0 + \beta_\alpha x^\alpha \rangle = \langle \beta \rangle + (n-1)\langle \beta x \rangle \qquad (3) \\ \sigma_\Sigma &= \sigma_\beta + (n-1)\frac{\sigma_\beta}{\langle \beta \rangle}\frac{\sigma_x}{\langle x \rangle}\langle \beta x \rangle \qquad (4) \end{aligned}$$

where n is the number of units (including the offset unit) in the layer.

Given that we have knowledge of the distribution of the x^i (mean and standard deviation), we can extrapolate some informations about the ideal distribution for the betas. Assuming that β and x are stochastically independent and therefore $\langle \beta x \rangle = \langle \beta \rangle \langle x \rangle$, we can bring forward Eq. 3 and 4:

$$\begin{aligned} \mu_\Sigma &= \langle \beta \rangle + (n-1)\langle \beta \rangle \langle x \rangle = \mu_\beta(1 + (n-1)\mu_x) \qquad (5) \\ \sigma_\Sigma &= \sigma_\beta(1 + (n-1)\sigma_x) \qquad (6) \end{aligned}$$

and therefore

$$\begin{aligned} \mu_\beta &= \frac{\mu_\Sigma}{1 + (n-1)\mu_x} \qquad (7) \\ \sigma_\beta &= \frac{\sigma_\Sigma}{1 + (n-1)\sigma_x} \qquad (8) \end{aligned}$$

Generalising beyond the first layer, x will represent the input for the next layer, and could therefore represent the output region of the activation functions used. In general, however, the activation functions of the input layer will not be always the same. If this is the case, it is sufficient to do the replacement

$$\mu_x \to \frac{1}{n-1}\sum_\alpha \mu_{x^\alpha} \qquad (9)$$

and

$$\sigma_x \to \frac{1}{n-1}\sum_\alpha \sigma_{x^\alpha} \qquad (10)$$

To summarise up, if we need to generate the betas for connecting an input layer (whether it is the actual input layer or a hidden layer) with $n-1$ values

x^α to a unit with an activation function with an active region characterised by μ_Σ and σ_Σ, we can draw them from a normal distribution with mean

$$\mu_\beta = \frac{\mu_\Sigma}{1 + \sum_\alpha \mu_{x^\alpha}} \tag{11}$$

and standard deviation

$$\sigma_\beta = \frac{\sigma_\Sigma}{1 + \sum_\alpha \sigma_{x^\alpha}}. \tag{12}$$

2 Linear independence

Since the betas can be interpreted as a matrix, it is clear that if the rank of this matrix is lower than it could be, then we are reducing the dimensionality of the output. This might results in two units having the same values, and could lead to numerical instabilities when making a fit or a wave function optimisation within VMC.

For avoiding this problem there are mainly two approaches that one could think of. To illustrate them, we will make show some results obtained with a python notebook (stored in `NNVMC_Share/Beta initialisation/Code`).

2.1 Strict linear independence

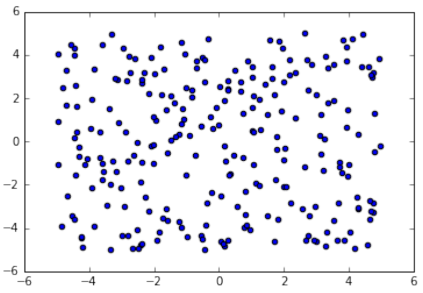
This is the preferable choice, i.e. take l.i. betas. However, if the output has an higher dimensionality than the input, it is clear that not all the betas can be l.i..

Nevertheless, we show some practical examples. Let us take a NN with a two-dimensional input and output, and let use the logistic function as activation function. From Figure 2 we find confirmation to what was to be expected: with l.i. betas the output still spans the whole 2D space, with the l.d. betas the output space is reduced to a line, and with fully random betas the output space is a somewhat reduced 2D space.

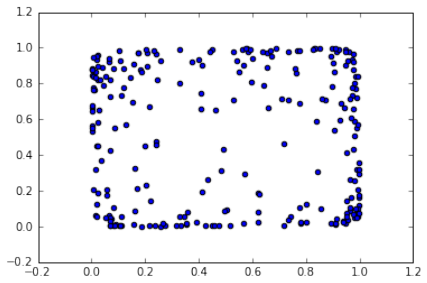
It is worth mentioning that including or not the offset into the betas (for l.i.) does not change the result.

2.2 Alternating activation functions

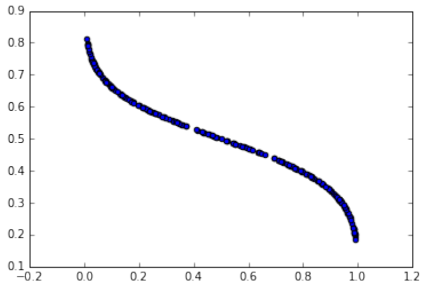
It might be beneficial to use different activation function, instead of always the same. In the following we report the results obtained with the same settings as for subsection 2.1, but alternating a gaussian activation function and a logistic one. Results are shown in Figure 3 and demonstrate that this technique is not very beneficial from the space-spanning point of view. However, this does not imply that it cannot help when minimising, since it could still help breaking some degeneracies.



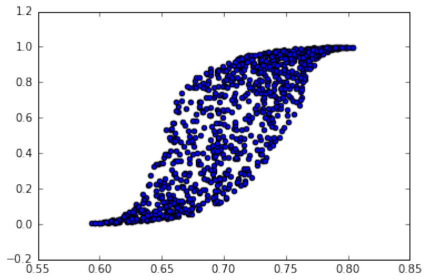
(a) Input



(b) Output with l.i. betas

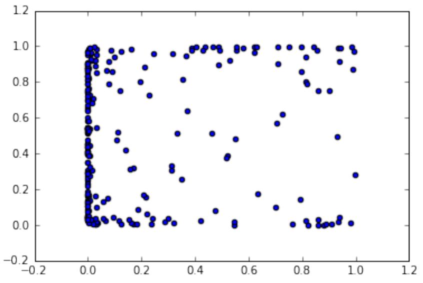


(c) Output with l.d. betas

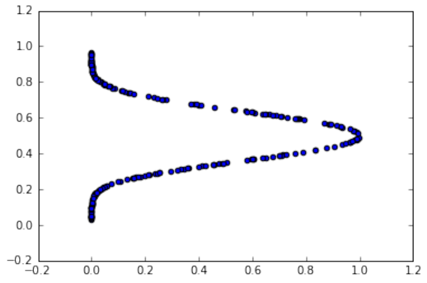


(d) Output with fully random betas

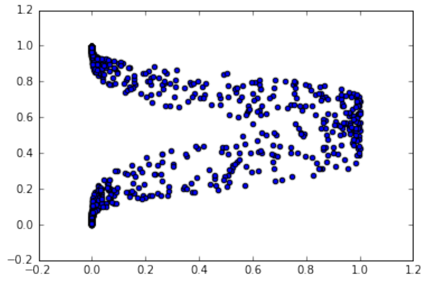
Figure 2: Results obtained using alternating activation functions



(a) L.i. betas



(b) L.d. betas



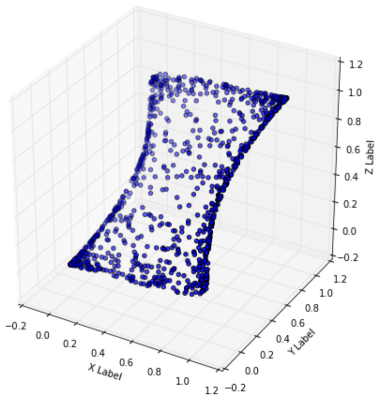
(c) Fully random betas

Figure 3: Results obtained using alternating activation functions

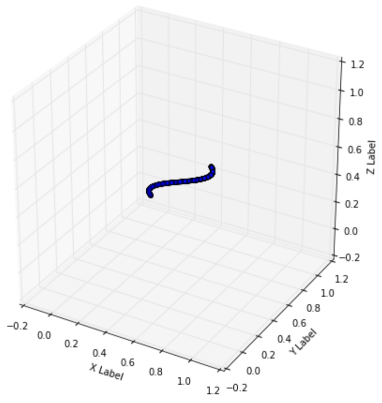
2.3 Quasi-l.i. betas

When the dimensionality of the output is higher than the dimensionality of the input, it is impossible to have l.i. betas. In this case we then generate the remaining betas as linear combination of the random basis set we have built.

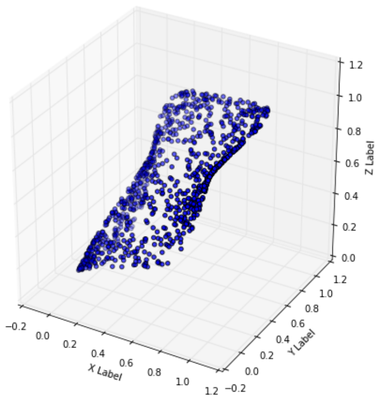
To graphically represent the results, we have considered the case of a two-dimensional input, and a three-dimensional output. Results are reported in Figure 4.



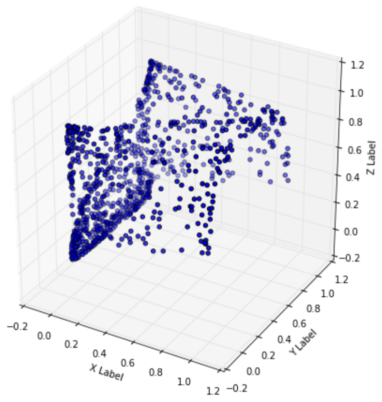
(a) Quasi-l.i. betas



(b) L.d. betas



(c) Fully random betas



(d) Quasi-l.i. betas with alternating activation functions

Figure 4: Results obtained using alternating activation functions

From this example not only we can have confirmation that using quasi-l.i. betas is in general better than fully random betas, as it was expected, but also that the use of alternating activation functions might after all be beneficial. In fact, with the two combined techniques we have the most "scattered" output.