Incomplete quantum tomography using neural networks

Olivia Di Matteo

July 6, 2017

1 Problem formalism

The goal of quantum tomography is, given (many identical copies of) some arbitrary quantum state, take a series of measurements in order to determine its density matrix. When the dimension of the system d is prime (p), or power of prime (p^n), a complete set of $d+1$ bases that are mutually unbiased comprise an optimal set of measurements. These can be written in vector form, but another common way of expressing them is as $d+1$ sets of $d-1$ commuting observables (total of d^2-1) which have as mutual eigenvectors bases that are MU.

A density matrix ρ can be expressed in terms of d^2-1 parameters; while an arbitrary complex $d \times d$ matrix contains $2d^2$ entries, the hermiticity and the condition on the trace of a density matrix, i.e. $\mathrm{Tr}(\rho) = 1$ reduce the number of parameters required. A convenient way of expressing the density matrix is in terms of its expansion coefficients in an operator basis:

$$\rho = \frac{1}{d}\mathbb{1} + \frac{1}{d}\sum_{i=1}^{d^2-1} a_i \Lambda_i \tag{1}$$

The operators in the basis should be traceless, and trace orthogonal $\mathrm{Tr}(\Lambda_i \Lambda_j) = c\delta_{ij}$. Furthermore, if they are Hermitian, the coefficients $a_i = \mathrm{Tr}(\rho\Lambda_i)$ will be real. The canonical example is of course when $d = 2$ and $\Lambda = \{\sigma_x, \sigma_y, \sigma_z\}$, so that the vector $\mathbf{a} = (a_1, a_2, a_3)$ is simply the 2-dimensional Bloch vector.

The goal of a tomographic process would then be, given some measurement data, to determine the values of the coefficients a_i, and consequently ρ. *In what follows, we focus exclusively on the special case where ρ is a unknown random pure state.*

2 Basic version: Incomplete tomography using neural networks

If we consider multi-qubit systems, i.e. $d = 2^n$, the number of measurements required scales poorly with the number of qubits. As larger numbers of qubits become experimentally tractable, we will need effective methods of incomplete tomography to make the reconstruction process feasible. Efforts in this direction have involved compressed sensing, and least-bias maximum-likelihood state estimation (LBMLE). We propose here a method based on neural networks. The

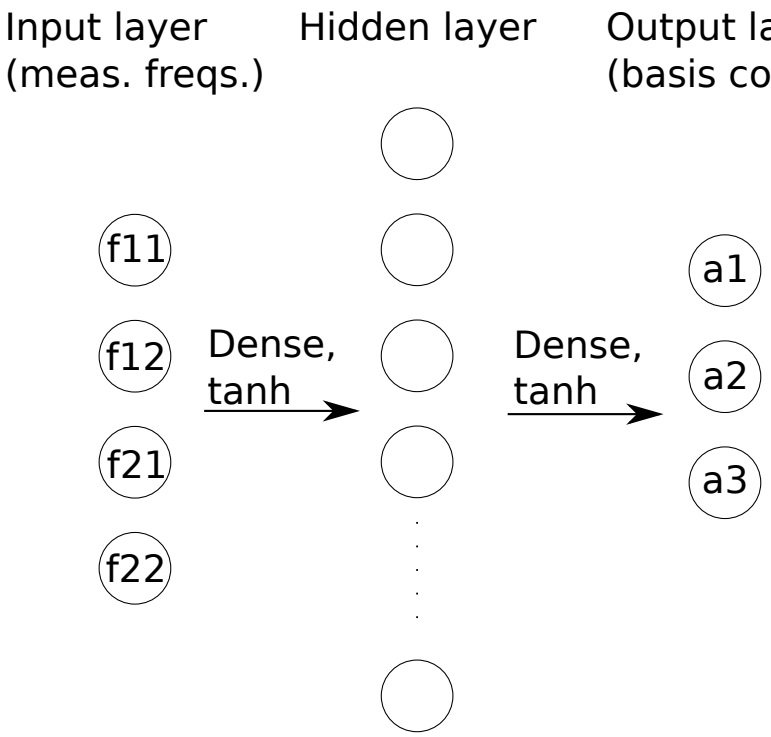

Figure 1: Simple diagram of neural network. The measurement frequencies from each measured basis are passed in as input nodes. The hidden layer seems to work well with a quadratic number of nodes, after which there are diminishing returns. The output layer should have $d^2 - 1$ nodes.

idea here is that the networks can be trained to reconstruct states based on simulated incomplete experimental data; this allows the heavy lifting to be done computationally, as one needs only train the network once, and then feed it the "real" experimental results. There is also a huge amount of versatility as one can simply adapt the process to train using whatever measurements are taken in a particular experimental setup.

We propose a basic feed-forward neural network for this task, shown in figure 1. The input layer consists of measurement frequencies for the measured bases. These are generated 'experimentally' by computing the set of projectors for each basis, and the probability of that outcome via the Born rule. Then we use MC simulation to generate frequencies of each outcome (10000 points seems to give us frequencies pretty close to the original). The output layer consists of the basis coefficients above, the a_i.

Through experimentation we need to determine the properties of the hidden layer(s) that are the most effective for this purpose. So far things that work well are:

- We choose tanh as the activation function, as it takes a range (-1, 1) and the a_i can be positive or negative.
- We choose the cosine proximity as the loss function; this is physically

motivated, as in the single-qubit case it can be thought of as the angle between Bloch vectors, which one should try to minimize to get as close as possible to the true state.

- We use a single hidden layer. Adding more seems to give us no benefits, except for longer coffee breaks while the code runs.
- The networks are trained using data taken from Haar-random pure states.

We also need a way to quantify the performance of the network. We choose the **fidelity**, however there are a few hiccups here:

- The output of the network is not always normalized. As we assume we are working strictly with pure states, before computing the fidelity we normalize the vectors.
- Once normalized, the state is not necessarily a valid quantum state, i.e. is not positive semidefinite (PSD), in particular for $d > 2$. In cases I have looked at, there is some very tiny non-zero eigenvalue, even in the single qubit case. Therefore to get a physical state estimation we should look to the closest PSD matrix.

3 1 qubit

As proof of concept, we train the network to reconstruct a single qubit pure state. We use a single hidden layer of size 64 (input size is 6 for all bases, 4 for two bases, and output size is always 3). We trained our network with 9000 Haar-random pure states, and tested on 1000 additional ones.

Training the network with all 3 MUBs yields an average fidelity of 0.9999, which is a solid sanity check. We then attempt to train it using simulated measurement data from only 2 of the 3 MUBs. The network takes only a second or two to train (data generation and comparison with LBMLE takes far longer!). Our network reconstructs states with a fidelity of 0.944 on average; this can be compared, using the same test data, with the LBMLE algorithm, which reconstructs the states with an average of 0.91. Results are discussed in more detail in Section 8.

To do: plots of dependence of fidelity on hidden layer size, and on number of training data points

4 1 qutrit

We choose as our operator basis the Gell-Mann matrices. The qutrit is an interesting case because density matrices constructed using arbitrary vectors of norm 1 are Hermitian and have trace 1, but are not necessarily PSD. As a consequence there has been a lot of work done in attempt to characterize the regions of the qutrit Bloch sphere that produce 'legal' states (add references).

Regardless of this fact, it seems that the neural network is able to learn quite well how to reconstruct qutrit states. We train again using largely the same parameters as the qubit case: 10000 Haar-random pure states split 9000/1000, and still a hidden layer of size 64. States that are not PSD after reconstruction are kicked over to the closest PSD state.

I put the results in Table 1. Measuring all 4 bases of course reconstructs the states with fidelity close to 1, so I've not included this in the table. With 3 bases, the two algorithms perform comparably with an average fidelity of around 0.958. What is more interesting is that when we measure in only 2 bases, the network begins to perform better.

Num. meas. bases	Fidelity NN	Fidelity LBMLE
3	0.9582	0.9580
2	0.8978	0.8547

Table 1: Fidelity using incomplete measurements. Here I averaged together results of all choices of 2/3 bases, as they weren't really too different.

5 2 qubits

After successfully chasing down some rogue constants and normalization factors, I was able to get the 2 qubit case training. I used again 10000 states, which seems to be sufficient. The hidden layer here has size 128. I changed this randomly because I was messing around with things when they didn't work, and just didn't bother to change it back. The states produced are still quite close to being positive semi-definite, though not perfect.

Training is still extremely fast, in only a couple seconds. The LBMLE process begins to take a long time, however. Results are in table 2. This is quite interesting; LBMLE seems to slightly outperform the network until we are measuring in only 2 bases; then the network becomes significantly more successful.

Num. meas. bases	Fidelity NN	Fidelity LBMLE
5 (All)	0.99988	0.99798
4	0.96154	0.98728
3	0.91237	0.93492
2	0.84865	0.76785
1	0.76184	0.62757

Table 2: Fidelity using incomplete measurements. There are too many combinations, so I chose only one set of 4, 3, and 2 bases each time.

6 3 qubits

For 3 qubits, I had to increase significantly the amount of data as well as the size of the hidden layer (50000 points, 45000 for training, and hidden layer size 512). See Table 3. Unfortunately for this amount of data, comparison with the LBMLE algorithm takes over 1 hour or more, depending on the number of bases measured. Thus for the time being I have stopped doing the comparison and am looking mostly at the parameters of the network itself, as there are some interesting relationships appearing (plotted in figures 2 and 3.

It is interesting too look at the distribution of the fidelities for each case.

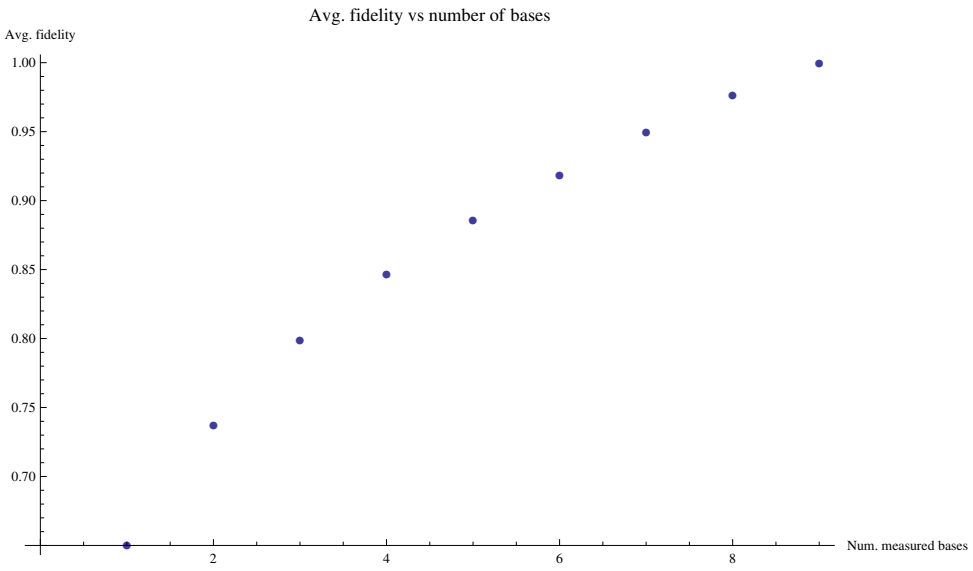

Figure 2: Average fidelity vs number of bases measured in dimension 8 tests.

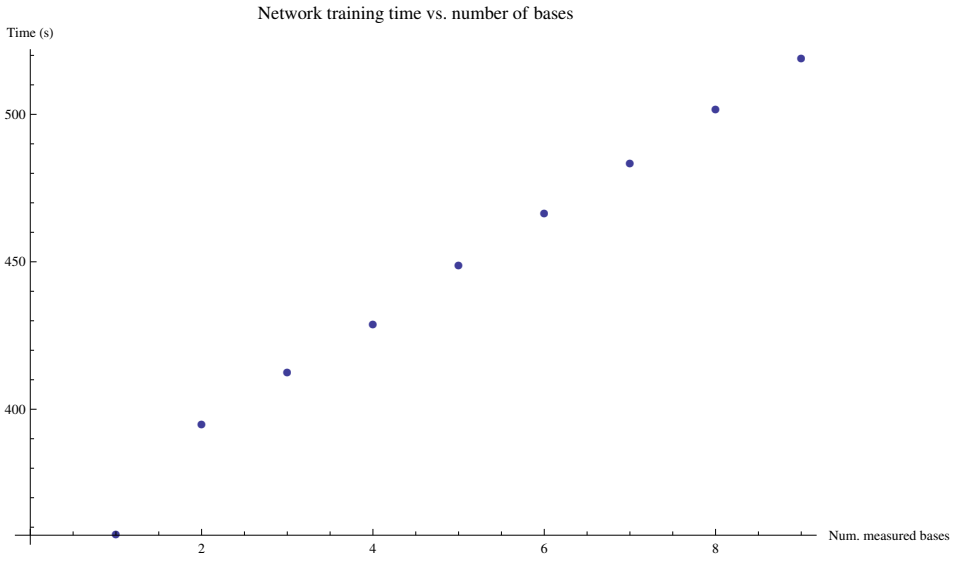

Figure 3: Network training time vs number of bases measured in dimension 8 tests.

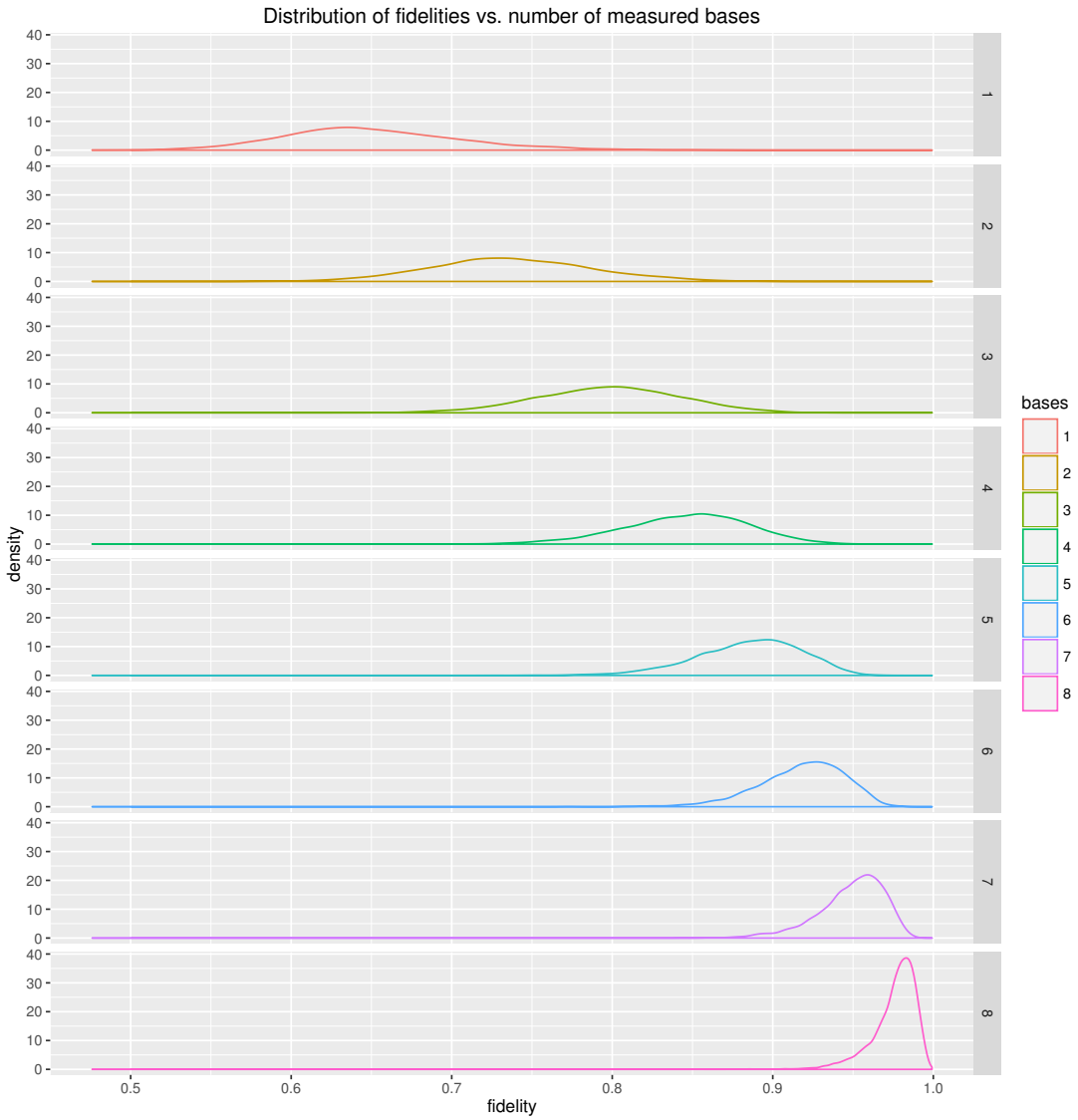

Figure 4: Distributions of the fidelity for varying number of measured bases. As expected the range broadens as we take fewer measurements. The case for all 9 bases is so sharply peaked around 1 that it essentially washes out the others, so it's not plotted here.

Num. meas. bases	Fidelity NN	Training time
9 (All)	0.999	518.870
8	0.976	501.368
7	0.949	483.277
6	0.918	466.364
5	0.885	448.547
4	0.846	428.746
3	0.798	412.304
2	0.737	394.760
1	0.650	357.285

Table 3: Fidelity using incomplete measurements. 50000 data points, 45000 for training, hidden layer size 512. Tried doubling the number of data points for 5 bases case, no change in fidelity, so I suspect 50000 is enough. The relationship between time and number of bases is linear; between fidelity and number of bases is exponential.

7 Next questions

Some obvious questions to consider next:

- What happens for mixed states? Does the neural network still fare well? We cannot normalize it now, it may live inside the Bloch sphere. How do we ensure that the output is always a vector with norm < 1 still close to the original state?

8 Angular dependence of single qubit reconstructions

This is some interesting stuff that goes beyond the scope of doing tomo with neural networks, and focuses more on the specific choice of measurements in an incomplete tomography setting. However it's still quite a neat result, and allows us to see where exactly the neural network is outperforming the existing algorithm.

One important thing to consider is the quality of the data set that goes into the network training. We used Haar-random states (HRS), which are generated by hitting a fiducial state by a Haar-random unitary (HRU). HRUs are distributed in a distinct way; in particular, the "weight" factor in the measure of integration is

$$dU \simeq \sin\beta d\alpha d\beta d\gamma \tag{2}$$

where α, β, γ are the 3 parameters which allow us to construct the matrix

$$U(\alpha,\beta,\gamma) = \begin{pmatrix} e^{i\alpha}\cos(\beta/2) & -e^{i\gamma}\sin(\beta/2) \\ e^{-i\gamma}\sin(\beta/2) & e^{-i\alpha}\cos(\beta/2) \end{pmatrix} \tag{3}$$

(more technically this is SU(2) rather than U(2) I guess). This looks very similar to the integration measure in spherical coordinates, when integrating over a sphere of radius 1. In this sense, β plays the role of the polar angle θ. The extra factor of $\sin\beta$ takes its maximum value at $\beta = \pi/2$, namely at the

equator of the sphere; there is consequently more 'weight' added to these states, meaning that *equatorial states are more likely to occur than, say, polar states, when randomly sampling from the distribution of HRUs.*

To that end, I was initially concerned that using a training dataset based on HRS would mean that there are more states around the equator, and that consequently we would be able to learn these states better than the polar states. So I decided to plot the polar angle of the states on the Bloch sphere vs the fidelity of the reconstruction from the two different methods. I did this for all 3 combinations of measurement bases, ZY, XY, and ZX. Results for ZY and ZX were practically identical; the main differences lie between e.g. ZY and XY.

The first thing to note is that it doesn't seem that the distribution of the data set has much effect; I make this conclusion because when we reconstruct with the ZY bases, the states in the middle (i.e. angle $\pi/2$) are more poorly reconstructed than the polar states, despite there being more of them in the training set (see, for example, the distribution of fidelities of 5. What *does* make a huge difference is the bases we choose to measure in.

The X and Y axes are those defining the equator of the Bloch sphere, and so it makes sense that measuring in those bases will give us better reconstructions simply because there is more 'room to move around' in this portion of the sphere. There is, in a sense, greater chance for error here and so we want to learn as much as we can about how things related to these two axes vs. the polar one.

What's more interesting though is the differences between the two algorithms. The average fidelity for the neural network is pretty much the same whether we use ZY or XY. However for both algorithms, the fidelity for the XY case is far more correlated with angle. The LBMLE looks normally distributed in both cases but for the ZY measurements the normal distribution is filled in, while for XY the points simply follow the outline of a bell curve.

For the ZY case, the neural network looks like it fares worse closer to the middle (which I feel makes sense as we are using the cosine proximity as a loss function). However the average is still superior because the fidelity of the polar states is handled better than in LBMLE. In the XY case again the NN handles the polar states better, and it looks more paraboloid than it does normal.

Figure 5: Fidelity of reconstruction vs angle with measurement data from the Z and Y MUBs. The red points are those of the neural network, and blue for the least-bias estimation.

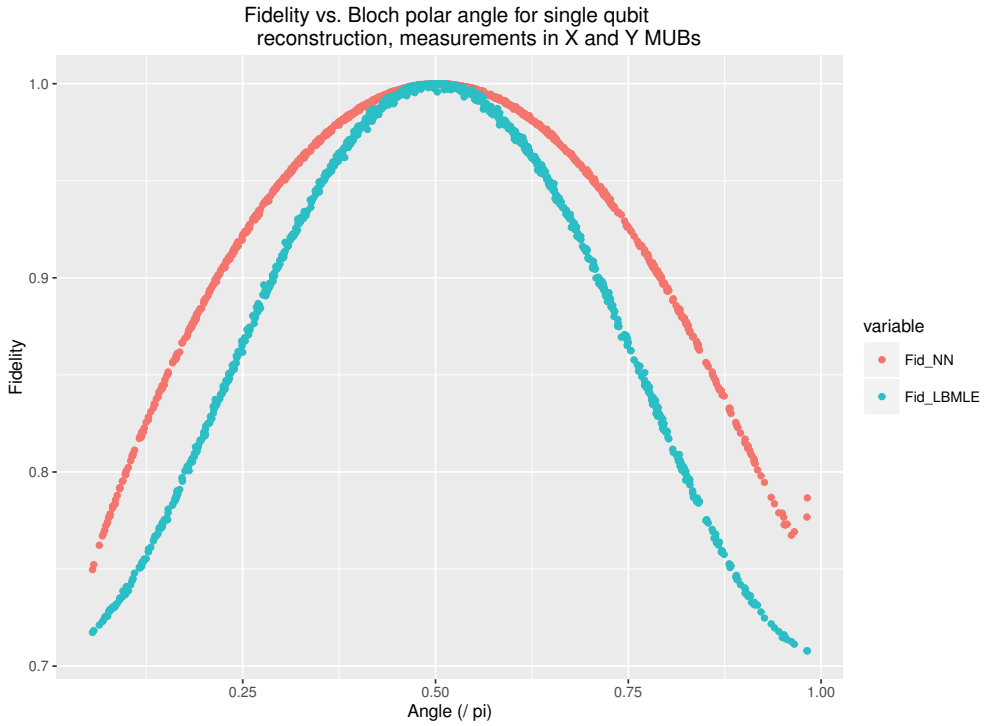

Figure 6: Fidelity of reconstruction vs angle with measurement data from the X and Y MUBs. Seems to be highly correlated with angle. The neural networks average fidelity seems to get a boost from its ability to better reconstruct the states closer to the poles of the Bloch sphere.