Week 6 meeting notes

|  |
| --- |
| Project proposals marked by Guy, but more marking is required until we can get the full feedback |
| “The Prior” = knowledge we have before we look at the data  The Prior, is split into:  “The Hyperprior” = parameters that deal with the population level i.e. mean and standard deviation. Additionally information that encompasses our understanding of the population e.g. the age of an open cluster is limited by the age of the universe, they cannot be older than 13.8Gyrs. More specifically if we look at M67, from previous studies we know that its age cannot be less than 2Gyrs or greater than 6Gyrs.  “prior” = parameters that deal with the individual objects of the population    θ = info on individual objects within population, i.e. θ = {θ1, θ2,….θN}, where θ describes the information we have about each star for a population of N stars  φ = info about the overall population    This doesn’t describe the hyperprior or prior, as those are probabilities  But these are parameters and the description of the relation between parameters    There aren’t any population level priors on mass because the population can have essentially any mass.  HOWEVER, I’m wondering about why we still can’t have the unphysical boundaries e.g. cannot be less than 0 (which I guess you could solve by using a half normal distribution) or like a 1000 solar mass star, though I know the population level priors on age limit the possible masses, but then shouldn’t those priors feed into the mass constraint? |
| There are no cluster-to-cluster level priors on mu as you wouldn’t expect there to be any commonalities to the mean, but the standard deviation might depend on some alpha and some beta, that describes the spread of the set of clusters we are studying. |
| Mini report – from preliminary results:  We did this, we tried this, we know it will work because of this, it is limited because of this, but we get that. The interpretation is this  “jot down in notebook”  We will later be able to use it in final report |
| Put a grid into the neural net |
| You can iteratively add layers to neural network using for loop.  According to current wisdom, you don’t want N to get too large, where N is the number of nodes per layer, too large colloquially is N > 1000  In the past people have done N = 512, with 10 hidden layers.  Regularization = start with high value, make it smaller until the output is good  Learning rate = too high is bad, too low doesn’t really matter (Nadam updates learning rate) Beta = determines how the learning rate decays per epoch (should be less than 1 so decay occurs)  = 0.9995 maybe  Validation amount is around 10% to 20%, hold back a random selection from the data, which is why shuffling the data is important, as it prevents some chunk from some part of the data distribution being reserved for validation which prevents the validation being done holistically on the data range. i.e. if the data from 0 and pi was taken from sin data which spread from -3pi to 3pi  Validation set = want to know that the model is generalising well, if the train error is much lower than the validation error it is not generalising well (i.e. over/under fitting)  Guy fit some overfitting by changing the regularization.  Batch sizes:  You split up data into batches, because most data has noise on it and by splitting up the data you essentially average the noisy values to the true value.  However because our data isn’t noisy we want to have a batch size equal to the number of data points so there is only 1 batch. |
| Working with Sin curves:  Do a sin curve or something and then remove part of a couple of the slopes and train the neural network on that  is what overfitting should look like  Figure out how to remove the overfitting |
| Dropout = randomly turning off weights in each layer  You set a fraction (in general between 20% and 50%), at every gradient evaluation, that fraction of the weights are chosen at random and are set to 0.  This prevents any one weight being too useful. Thus spreading the information is more spread out through the neural net. It’s like training multiple neural nets and so the resulting neural net is the average of the ensemble of neural nets.  This gives it the same advantages as random forests.  Dropout is potentially not useful for our purposes because we don’t have noisy data, so we don’t need an ensemble to average out the noise. |
| Gaussian process = multinomial Gaussian distribution (has an infinite number of orthogonal axes)  A neural network with an infinite number of neurons and a prior distribution over the weights is a Gaussian process.  You train the Gaussian process/optimize hyperparameters of the Gaussian process, on the data.  Then you can predict the (conditional) distribution on the infinite number of dimensions.  You can use it to predict what happens between points and also what happens outside the trained region (where of course the errors balloon out because it’s predicting a region where there is no data)  We could make our prior a Gaussian process. |
| Discussion on HBM problem 3 |
| Output layer having a linear activation function  Don’t use sigmoid as that limits between -1 and 1  Don’t use relu as that limits between 0 and infinity  But linear allows you to having unlimited range  Though with a bias you can use a relu to access negative values. |

Students To Do:  
- Check we’re happy with Guy’s annual review comments  
- Write mini report  
- Mess around with NNs e.g. regularization, learning rate, beta(s), batch sizes (mainly: try 1 batch), dropout (but not necessary)  
- We should set a target for next week  
- If we’re confident we have trained enough neural networks; tell Guy and he will share a grid with us (of some variety).

Guy to do:  
- Check Tander Lee’s grid  
- Send “Model Sunlike Star” notebook

Next week:   
- maybe discuss mixture models