Week 5 meeting notes

I forgot to take photos of the neural networks examples for the description of biases  
also for ReLUs

|  |
| --- |
| Start off by doing hierarchical modelling without the neural net, by using simulated data to demonstrate that it works.  Get good at HBMs, to understand neural nets and figure out how we’re going to get a grid.  There doesn’t seem to be much merit to making our own.  Consult ‘tander lee’? for about their model.  For neural nets people don’t know the optimal way of doing things.  There are too many options to exhaustively test every single one. |
| Tensor Flow  Tensor flow playground  Variables for the neural net:  Epoch = some amount of training  Learning rate = if you make it small, the neural net will train slowly, however making it large, causes large steps to be made and may skip over something important.  Regularization = used to stop overfitting but we are sort of ignoring it for now  Our problem type is “regression”  Batch sizes = we want big batch sizes, but we don’t really care about it.  Algorithms use a gradient descent  Learing rate \* gradient  Gradient sets the direction, Learning rate is the amount you change it by  In general algorithms don’t have a fixed learning rate, instead the learning rate decreases over the course of the training  Adam algorithm = the learning rate is modified at each epoch where it is multiplied by esomething, to make sure it gets smaller overtime.  People “test” the neural net as they go along  They leave validation till later  People butcher these definition but it’s fine as long as we are consistent with our definitions. |
| Bias = allows for translatory movement  It’s the c in mx + c  Each neuron takes a bias, in general. |
| Activation function  ReLU = Rectified linear unit  C:\Users\Harry\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\40E08E86.tmp  The function is 0 up to some threshold  You can use tanh but the important thing is there are regions where nothing happens and a region where lots happens. |
| What we need to talk about in the report:  Need to talk about learning rate, activation function, ?regularization?,  If we did any modifications: we’re putting in mass, age, initial metallicity, helium metallicity,  M-3.2 because the main sequence lifetime is proportional to the mass to the -3.2, we use domain specific knowledge so we can use that to train the neural net better, especially if we’re training log(g) because some ratio of age over main sequence lifetime, will track pretty well to log(g)  When the ratio is < 1 (during MS), it is fairly well approximated by 4.4  When the ratio is > 1 (post MS), then log(g) is going to drop down |
| Guy’s notebook on training on a grid  All the stars in the grid have the same metallicity  Mass runs from 0.8 to 1.4 in steps of 0.1  It goes beyond TAMS (terminal age main sequence)  Trained the neural net by putting mass and age, and trying to get out luminosity and Teff.  So that he can build an HR-diagram  Neural nets hate having inputs and outputs with large dynamical ranges, so he logs the inputs and outputs.  He also does them as single precision floating point numbers  You can do them as doubles  Single precision floating point numbers, make the process faster but less precise.  Though as it’s just referring to the decimal place that can be stored, it’s probably the one you want to choose.  Guy uses the functional form not the sequential form  Karas optimizers:  History = gradient decent (got stuck in local minima) 🡪 stochastic decent (doesn’t have a learning rate which gets smaller over time) 🡪 adam (exponential decay of learning rate) 🡪 nesterov momentum (keeps track of where it is going) 🡪 nesterov momentum + adam = nadam  Epsilon = tolerance value on numerical accuracy = doesn’t take anything below 1E-9 seriously.  Model compiles: by making tensorflow code  The metrics we want to see are MAE and MSE  Tensor board allows you to watch things train, it’s good if you can get it to work.  Build the model 🡪 compile it 🡪 fit it  You can fit multiple times, but you can’t change say the optimizer and then recompiler and keep the same mode, you get a different model so you have to refit.  By training for longer you make neural net outputs more smooth like the real data  “with a small architecture you can only achieve a certain amount of smoothness. With bigger architecture you can achieve more smoothness”  A neural network can better approximate a function if it has a more flexible function (i.e. larger architecture), however it does take longer to train and it takes more computational time to evaluate.  You can make neural network output smoother by:  Increasing architectural size, training for longer, regularizing (if necessary)  It seems that regularizing is designed to smooth, but it isn’t always advisable as it can smooth out actual features of the data.  When we have decided on the architecture for the neural network etc. we can send it to Guy (on github) to train it on his GPU.  Once trained we can save out all the weights and biases, and move to pymc3, so that whenever we want to know what the Teff or luminosity of a star is, given an age and a mass, we can just ask the neural net. From the asteroseismic stuff we only need delnu, you can’t get numax from a stellar model.  We might want to predict radius (maybe log(g)) |
| We want to simultaneously work on Neural net and HBM  Run tests on simulated data so we can tell we are doing sensible things  Then move onto real data but in the minimal viable way e.g. no asteroseismic data  For a cluster, we will have a apparent magnitudes, all the stars will have similar distances because they are in a cluster, using extinction, apparent magnitude and distance we can determine the luminosity (with some bolometric correction).  Temperature we can get from spectroscopy and that is enough to star running, though the results won’t be good as the each star won’t constrain the age to any meaningful value, but it’s enough to show it’s plausible.  Then we can add in delnu and add in the M67 red giant delnus from the stello paper and run it again.  Then find some eclipsing binaries in M67…slowly build the model up, which means as long as we get going quickly we can get results early and then go further and take more risks. |
| To begin with train on a sine wave curve  As we’re making a neural net to do what MESA does we don’t need to add noise to our tests  Evaluate sin(x) for 10,000 points, between 0 and 1 and then input x, output sinx  Then you can make the frequency of the sine wave be fast which confuses the neural net  The relative precision error, goes with the gradient of the function because at the top of the curve when the gradient changes fast the precision error is higher because there is less data in that region. |
| Pymc3 doesn’t have any good start off resources  Pystan does but you can only do it on unix  Tensorflow has its own probability library called Edward2, which is an HMC sampler but doesn’t have a NUTS sampler (we think), tuning the HMC is annoying.  Pymc3 handles the tuning |
| For next week: play around with neural net and fit a sine wave, mess around with tensorflow playground. |
| Train to a grid  Then create special cases from MESA tracks to test form there (because we can create more data by running more MESA tracks) |
| Leaky ReLU should be excellent loss function for what we want  It would be good to have a section in the final report which discussed different set up of the neural network training and architecture backed up by some theory for our choices.  With a description of what set up was best to worst and how that matched the theory.  Do some test training on a single MESA track and then generalize the findings to then figure out what would be best method to train, when it comes to training the neural net on the whole data set. |

Weight number 333?

Get Guy to share the is notebook for training to a grid