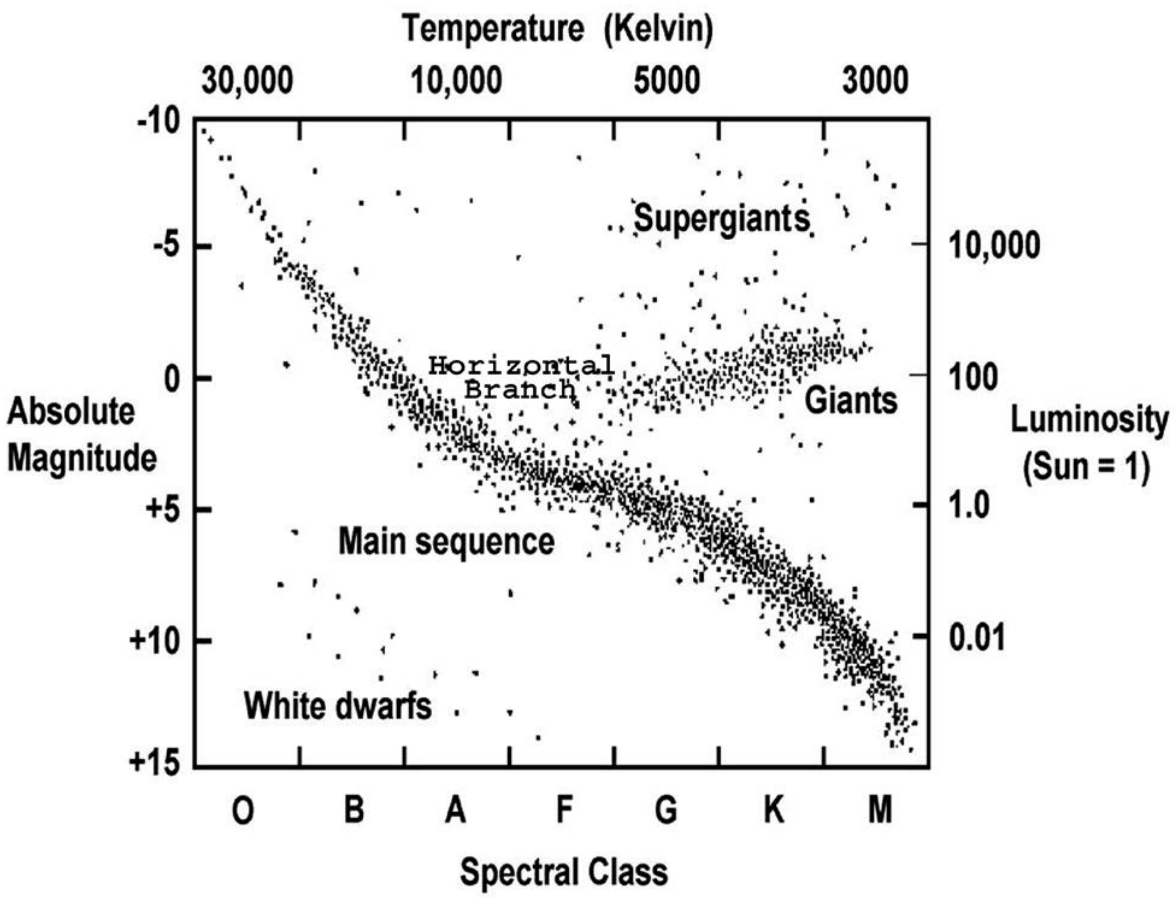
u7280249 ASTR4004 Assignment 4

Olivia Walters | Neural Networks

Introduction

Astrophysical Model

I’m going to create a simple astrophysical model for the classification of stars on a Herzsprung-Russel diagram:



The model will take two parameters: absolute magnitude and temperature ($\log T$), and it should output the probability of it being a:

A. White Dwarf
B. Main Sequence Star
C. Giant

This should allow for a simple training set that I can pull from online. I can also use a small number of nodes.

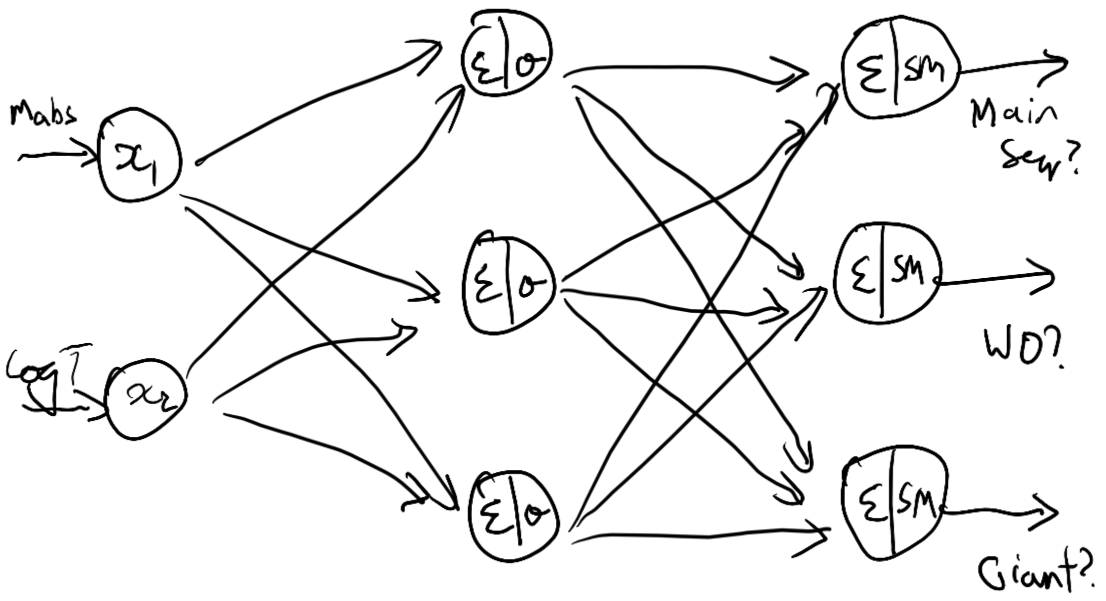
Network Structure

The ideal network would follow what appears to look like a polynomial of degree three, with the model being able to differentiate between stars near, below, and above the main sequence central model line. So there only needs to be 3 parameters at most to have a good fit.

There will be two input neurons, three hidden neurons, and three output neurons.

Like the lecture example, the output layer will have an SoftMax activation function, and the hidden layer will use a sigmoid function.

Below is the diagram of the network structure.



Training Data

I'm only going to include one main sequence star per spectral class, and an additional 5 white dwarfs and 6 red giants/supergiants. Along with this I'm including the Sun. This makes a total of 17 test data points per evaluation.

Both temperature and absolute magnitude will be normalized across the whole data set. Note that we use base 10 logarithmic scales because it linearises the input parameters for the neural network. For example, it's much easier to backpropagate when you're dealing with single digit numbers.

As an example, here's how the input/output would work for the star Sirius B, a white dwarf:

$$x = \begin{bmatrix} 11.18 \\ 4.4014 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Here’s the list of all stars used, with data gathered from Wikipedia:

#	**Name**	**Class**	**Type**	**Abs. Mag.**	**T eff.**	**logT**	**Y-Type**
1	The Sun	G	Main Sequence	4.83	5,778	3.762	1
2	10 Lacertae	O	Main Sequence	−4.17	34,550	4.538	1
3	Alkaid	B	Main Sequence	−0.67	15,540	4.191	1
4	Sirius A	A	Main Sequence	1.43	9,845	3.993	1
5	Sirius B		White Dwarf	11.18	25,000	4.398	2
6	Van Maanen 2		White Dwarf	14.21	6,130	3.787	2
7	WD 0343+247		White Dwarf	16.8	4,197	3.623	2
8	Alnitak Aa	O	Blue Supergiant	-6.0	29,500	4.470	3
9	Aldebaran	K	Red Giant (Branch)	-6.041	3,900	3.591	3
10	Antares	M	Red supergiant	−5.28	3,660	3.563	3
11	78 Ursae Majoris A	F	Main Sequence	2.84	6,908	3.839	1
12	61 Ursae Majoris	G	Main Sequence	5.53	5,488	3.738	1
13	Epsilon Eridani	K	Main Sequence	6.19	5,049	3.703	1
14	Barnard's Star	M	Main Sequence	13.21	3,195	3.504	1
15	Procyon B		White Dwarf	13	7,740	3.889	2
16	LP 658-2		White Dwarf	15.44	4,270	3.630	2
17	IK Pegasi		White Dwarf	7.04	35,500	4.550	2
18	Pollux	K	Red Giant (Clump)	1.08	4,586	3.661	3
19	Rho Persei	M	Red Giant (Asym.)	-1.7	3,479	3.541	3
20	Arcturus	K	Red supergiant	-0.30	4,286	3.632	3

The first 10 rows will be displayed as testing data, and the last 10 will be for validation. I won’t worry about how we’re meant to separate the sets as this is a toy example.

Equations of Gradient

Like what was provided in the lectures, due to the similarity of the network, the same gradient functions are ultimately used (but I will derive them in more detail to show my understanding).

We will use a cross-entropy loss function:

$$L(\hat{y}, \vec{y}_2) = \hat{y}^j \ln {y_2}_j$$

Note that the $\hat{y}$ vector corresponds to the unit vector expected output. I'm also using a more formalised einstein notation, hence the use of superscripts/subscripts without the Σ.

This makes the backpropagated correction written as follows, for completeness:

$${w_{n,\text{new}}}^i{}_j = {w_{n,\text{old}}}^i{}_j - \eta \delta_{jk} \delta^{li} \frac{\partial L}{\partial {w_{n,\text{old}}}^l{}_k} = {w_{n,\text{old}}}^i{}_j - \eta \frac{\partial L}{\partial {w_{n,\text{old}}}_i{}^j}$$

$${b_{n,\text{new}}}^i = {b_{n,\text{old}}}^i - \eta \delta^{ij} \frac{\partial L}{\partial {b_{n,\text{old}}}^j} = {b_{n,\text{old}}}^i - \eta \frac{\partial L}{\partial {b_{n,\text{old}}}_i}$$

For $n = 1{,}2$. The transpose as described is accounted for in lecture code.

Due to the use of SoftMax, this makes conveniently makes the pre-activation derivative of the output neurons as follows:

$$\frac{\partial L}{\partial {z_2}^j} = {y_2}_j - \hat{y}_j$$

Using the chain rule, and ${z_2}^j = {w_2}^j{}_i {y_1}^i + {b_2}^j$, we can get the equations for the weight matrix and bias vector (using einstein summation notation):

$$\frac{\partial {z_2}^k}{\partial {w_2}^j{}_i} = \frac{\partial}{\partial {w_2}^j{}_i} ({w_2}^k{}_l) {y_1}^l = \delta^k{}_j \delta^i{}_l {y_1}^l = \delta^k{}_j {y_1}^i$$

$$\frac{\partial {z_2}^k}{\partial {b_2}^j} = \frac{\partial}{\partial {b_2}^j} ({b_2}^k) = \delta^k{}_j$$

$$\frac{\partial L}{\partial {w_2}^j{}_i} = \frac{\partial L}{\partial {z_2}^k} \frac{\partial {z_2}^k}{\partial {w_2}^j{}_i} = ({y_2}_k - \hat{y}_k) \delta^k{}_j {y_1}^i$$
$$= \left({y_2}_j - \hat{y}_j\right) {y_1}^i$$

$$\frac{\partial L}{\partial {b_2}^j} = \frac{\partial L}{\partial {z_2}^k} \frac{\partial {z_2}^k}{\partial {b_2}^j} = ({y_2}_k - \hat{y}_k) \delta^k{}_j$$
$$= \left({y_2}_j - \hat{y}_j\right)$$

For the hidden layer nodes, the activation function being a sigmoid make calculating the derivative easy as well:

$$\frac{\partial {y_1}^k}{\partial {z_1}^j} = \delta^k{}_j(1 - {y_1}_j){y_1}_j$$

We can use the chain rule again:

$$\frac{\partial {z_2}^i}{\partial {y_1}^k} = {w_2}^i{}_l \frac{\partial}{\partial {y_1}^k}({y_1}^l) = {w_2}^i{}_l \delta^l{}_k = {w_2}^i{}_k$$

$$\begin{aligned}\frac{\partial L}{\partial {z_1}^j} &= \frac{\partial L}{\partial {z_2}^i}\frac{\partial {z_2}^i}{\partial {y_1}^k}\frac{\partial {y_1}^k}{\partial {z_1}^j} \\ &= ({y_2}_i - \hat{y}_i){w_2}^i{}_k \delta^k{}_j \left(1 - {y_1}_j\right){y_1}_j \\ &= ({y_2}_i - \hat{y}_i){w_2}^i{}_j \left(1 - {y_1}_j\right){y_1}_j\end{aligned}$$

Then, with ${z_1}^j = {w_1}^j{}_i x^i + {b_1}^j$, we can find our hidden layer gradient parameters:

$$\frac{\partial {z_1}^k}{\partial {w_1}^j{}_i} = \delta^k{}_j x^i$$

$$\frac{\partial {z_2}^k}{\partial {b_2}^j} = \delta^k{}_j$$

$$\begin{aligned}\frac{\partial L}{\partial {w_1}^j{}_i} &= \frac{\partial L}{\partial {z_1}^k}\frac{\partial {z_1}^k}{\partial {w_1}^j{}_i} = \frac{\partial L}{\partial {z_1}^k}\delta^k{}_j x^i = \frac{\partial L}{\partial {z_1}^j}x^i \\ &= ({y_2}_l - \hat{y}_l){w_2}^l{}_j \left(1 - {y_1}_j\right){y_1}_j x^i\end{aligned}$$

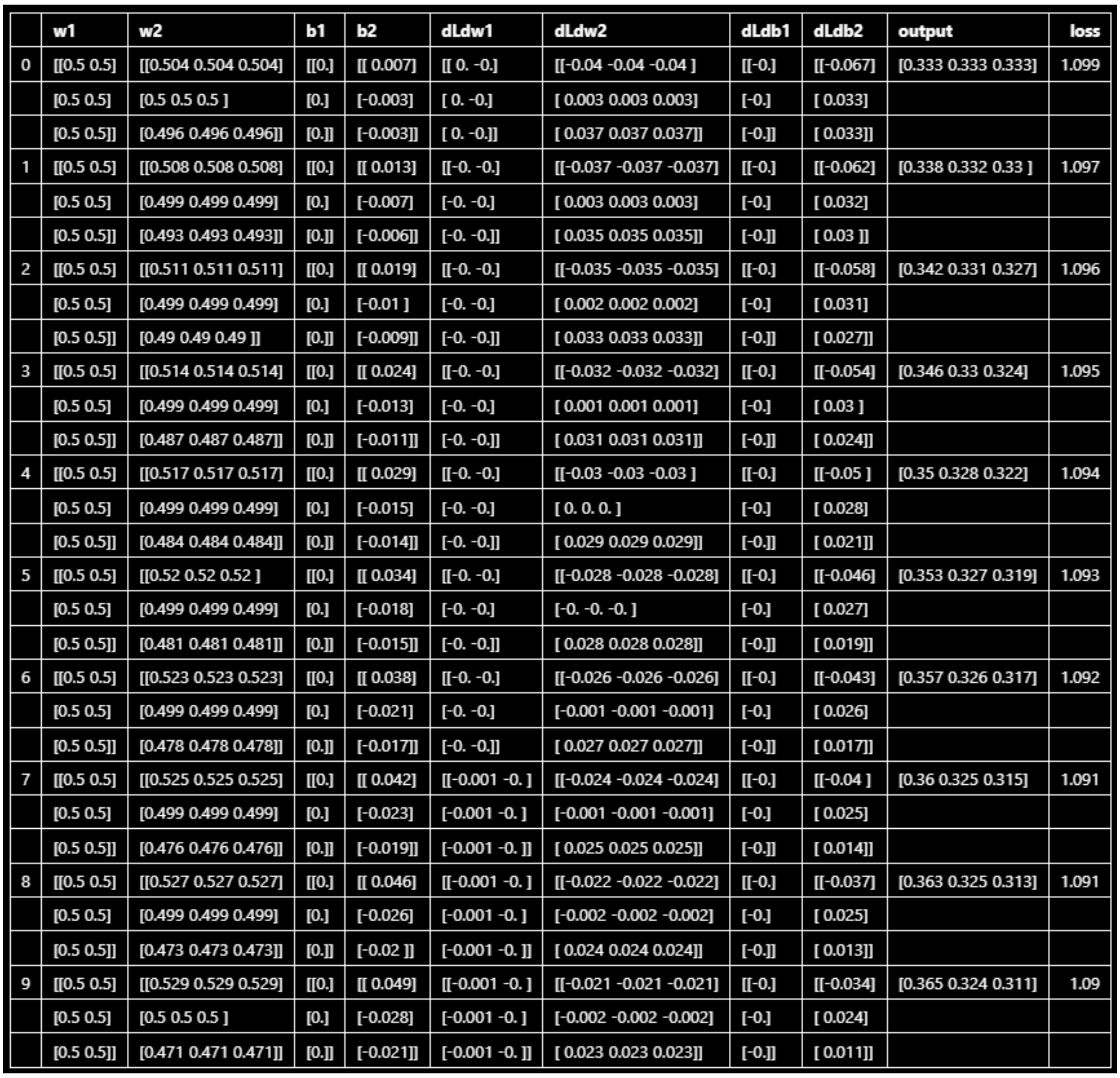
$$\begin{aligned}\frac{\partial L}{\partial {b_1}^j} &= \frac{\partial L}{\partial {z_1}^k}\frac{\partial {z_1}^k}{\partial {b_1}^j} = \frac{\partial L}{\partial {z_1}^k}\delta^k{}_j = \frac{\partial L}{\partial {z_1}^j} \\ &= ({y_2}_l - \hat{y}_l){w_2}^l{}_j \left(1 - {y_1}_j\right){y_1}_j\end{aligned}$$

And thus, we have the gradient values for each of the parameters we want to adjust. The einstein notation makes it look more complicated, but it is effectively a mathematical formalisation of what was provided in the lecture code.

Results of the Training

Learning Rate =0.1

Here is the table as displayed in a markdown-converted format of the Pandas python output:



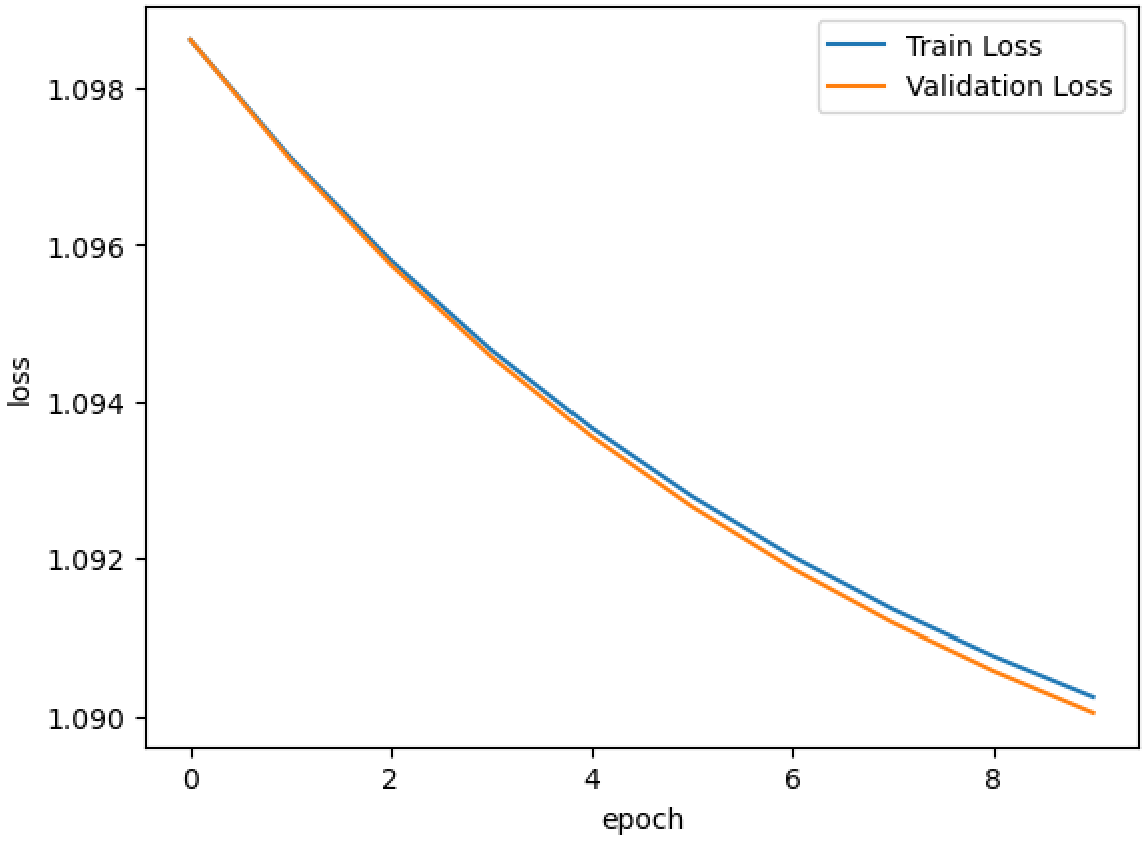
	w1	w2	b1	b2	dLdw1	dLdw2	dLdb1	dLdb2	output	loss
0	[[0.5 0.5]	[[0.504 0.504 0.504]	[[0.]	[[0.007]	[[0. -0.]	[[-0.04 -0.04 -0.04]	[[-0.]	[[-0.067]	[0.333 0.333 0.333]	1.099
	[0.5 0.5]	[0.5 0.5 0.5]	[0.]	[-0.003]	[0. -0.]	[0.003 0.003 0.003]	[-0.]	[0.033]		
	[0.5 0.5]]	[0.496 0.496 0.496]]	[0.]]	[-0.003]]	[0. -0.]]	[0.037 0.037 0.037]]	[-0.]]	[0.033]]		
1	[[0.5 0.5]	[[0.508 0.508 0.508]	[[0.]	[[0.013]	[[-0. -0.]	[[-0.037 -0.037 -0.037]	[[-0.]	[[-0.062]	[0.338 0.332 0.33]	1.097
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.007]	[-0. -0.]	[0.003 0.003 0.003]	[-0.]	[0.032]		
	[0.5 0.5]]	[0.493 0.493 0.493]]	[0.]]	[-0.006]]	[-0. -0.]]	[0.035 0.035 0.035]]	[-0.]]	[0.03]]		
2	[[0.5 0.5]	[[0.511 0.511 0.511]	[[0.]	[[0.019]	[[-0. -0.]	[[-0.035 -0.035 -0.035]	[[-0.]	[[-0.058]	[0.342 0.331 0.327]	1.096
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.01]	[-0. -0.]	[0.002 0.002 0.002]	[-0.]	[0.031]		
	[0.5 0.5]]	[0.49 0.49 0.49]]	[0.]]	[-0.009]]	[-0. -0.]]	[0.033 0.033 0.033]]	[-0.]]	[0.027]]		
3	[[0.5 0.5]	[[0.514 0.514 0.514]	[[0.]	[[0.024]	[[-0. -0.]	[[-0.032 -0.032 -0.032]	[[-0.]	[[-0.054]	[0.346 0.33 0.324]	1.095
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.013]	[-0. -0.]	[0.001 0.001 0.001]	[-0.]	[0.03]		
	[0.5 0.5]]	[0.487 0.487 0.487]]	[0.]]	[-0.011]]	[-0. -0.]]	[0.031 0.031 0.031]]	[-0.]]	[0.024]]		
4	[[0.5 0.5]	[[0.517 0.517 0.517]	[[0.]	[[0.029]	[[-0. -0.]	[[-0.03 -0.03 -0.03]	[[-0.]	[[-0.05]	[0.35 0.328 0.322]	1.094
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.015]	[-0. -0.]	[0. 0. 0.]	[-0.]	[0.028]		
	[0.5 0.5]]	[0.484 0.484 0.484]]	[0.]]	[-0.014]]	[-0. -0.]]	[0.029 0.029 0.029]]	[-0.]]	[0.021]]		
5	[[0.5 0.5]	[[0.52 0.52 0.52]	[[0.]	[[0.034]	[[-0. -0.]	[[-0.028 -0.028 -0.028]	[[-0.]	[[-0.046]	[0.353 0.327 0.319]	1.093
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.018]	[-0. -0.]	[-0. -0. -0.]	[-0.]	[0.027]		
	[0.5 0.5]]	[0.481 0.481 0.481]]	[0.]]	[-0.015]]	[-0. -0.]]	[0.028 0.028 0.028]]	[-0.]]	[0.019]]		
6	[[0.5 0.5]	[[0.523 0.523 0.523]	[[0.]	[[0.038]	[[-0. -0.]	[[-0.026 -0.026 -0.026]	[[-0.]	[[-0.043]	[0.357 0.326 0.317]	1.092
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.021]	[-0. -0.]	[-0.001 -0.001 -0.001]	[-0.]	[0.026]		
	[0.5 0.5]]	[0.478 0.478 0.478]]	[0.]]	[-0.017]]	[-0. -0.]]	[0.027 0.027 0.027]]	[-0.]]	[0.017]]		
7	[[0.5 0.5]	[[0.525 0.525 0.525]	[[0.]	[[0.042]	[[-0.001 -0.]	[[-0.024 -0.024 -0.024]	[[-0.]	[[-0.04]	[0.36 0.325 0.315]	1.091
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.023]	[-0.001 -0.]	[-0.001 -0.001 -0.001]	[-0.]	[0.025]		
	[0.5 0.5]]	[0.476 0.476 0.476]]	[0.]]	[-0.019]]	[-0.001 -0.]]	[0.025 0.025 0.025]]	[-0.]]	[0.014]]		
8	[[0.5 0.5]	[[0.527 0.527 0.527]	[[0.]	[[0.046]	[[-0.001 -0.]	[[-0.022 -0.022 -0.022]	[[-0.]	[[-0.037]	[0.363 0.325 0.313]	1.091
	[0.5 0.5]	[0.499 0.499 0.499]	[0.]	[-0.026]	[-0.001 -0.]	[-0.002 -0.002 -0.002]	[-0.]	[0.025]		
	[0.5 0.5]]	[0.473 0.473 0.473]]	[0.]]	[-0.02]]	[-0.001 -0.]]	[0.024 0.024 0.024]]	[-0.]]	[0.013]]		
9	[[0.5 0.5]	[[0.529 0.529 0.529]	[[0.]	[[0.049]	[[-0.001 -0.]	[[-0.021 -0.021 -0.021]	[[-0.]	[[-0.034]	[0.365 0.324 0.311]	1.09
	[0.5 0.5]	[0.5 0.5 0.5]	[0.]	[-0.028]	[-0.001 -0.]	[-0.002 -0.002 -0.002]	[-0.]	[0.024]		
	[0.5 0.5]]	[0.471 0.471 0.471]]	[0.]]	[-0.021]]	[-0.001 -0.]]	[0.023 0.023 0.023]]	[-0.]]	[0.011]]		

Where the output is the specific prediction for the sun, but the loss is aggregated over all 20 stars.

It's a bit messy, as to be expected when printing multiple lists of matrices. But hopefully you can see both the decreasing loss rate and the evolution of the table data over epoch. Notice that the values of the first weights don't do much. I believe that this is just because the model doesn't really need it in the initial stages.

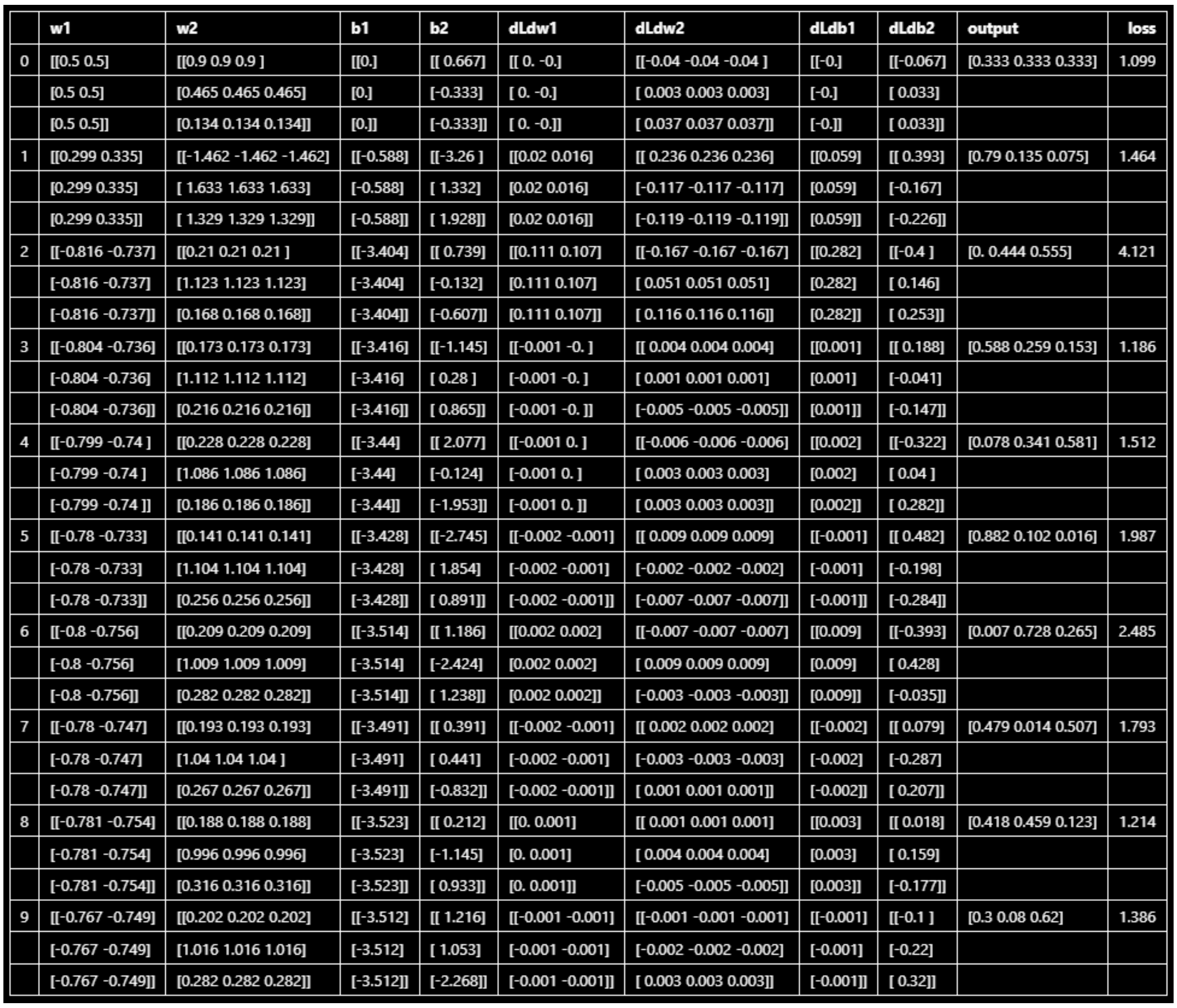
The learning rate is so small that there is only a small difference after each epoch. This is also seen in how w1 doesn’t change much.

The loss, however, clearly decreases:



Learning Rate >0.1

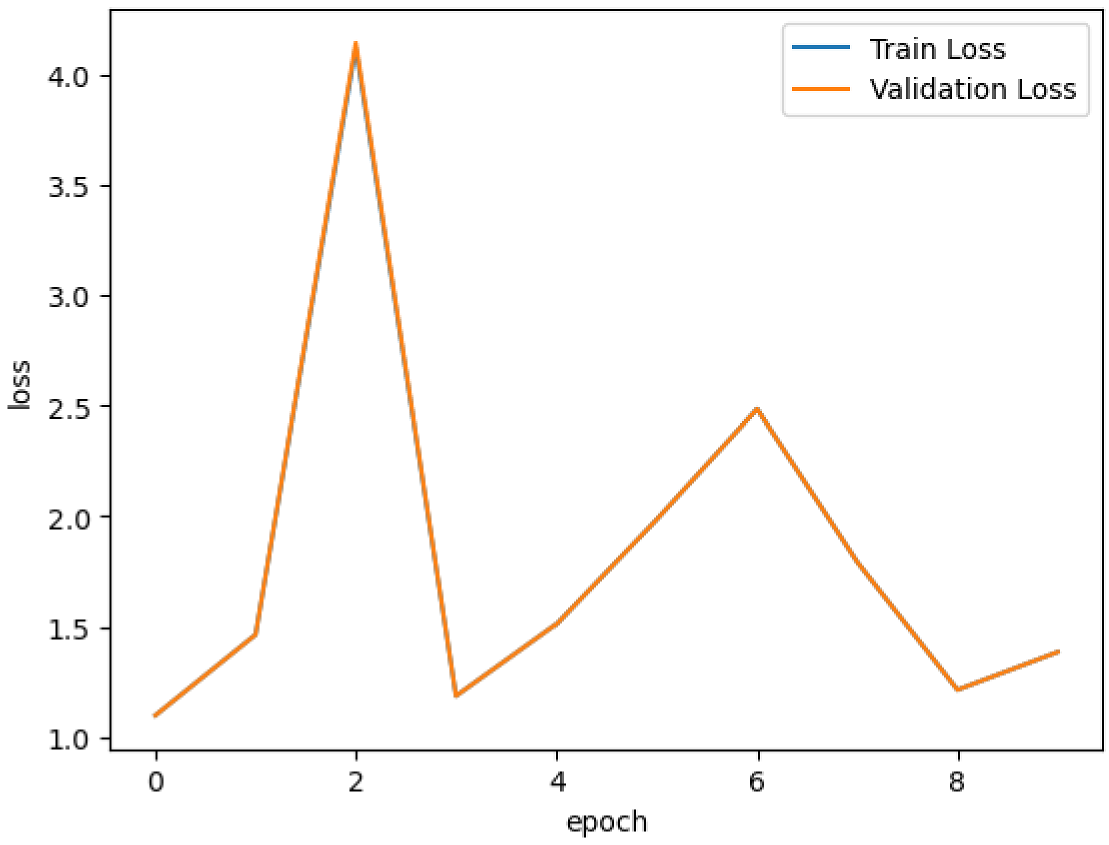
For this task, I chose a learning rate of 10, which is 100x that of above:



	w1	w2	b1	b2	dLdw1	dLdw2	dLdb1	dLdb2	output	loss
0	[[0.5 0.5]	[[0.9 0.9 0.9]	[[0.]	[[0.667]	[[0. -0.]	[[-0.04 -0.04 -0.04]	[[-0.]	[[-0.067]	[0.333 0.333 0.333]	1.099
	[0.5 0.5]	[0.465 0.465 0.465]	[0.]	[-0.333]	[0. -0.]	[0.003 0.003 0.003]	[-0.]	[0.033]		
	[0.5 0.5]]	[0.134 0.134 0.134]]	[0.]]	[-0.333]]	[0. -0.]]	[0.037 0.037 0.037]]	[-0.]]	[0.033]]		
1	[[0.299 0.335]	[[-1.462 -1.462 -1.462]	[[-0.588]	[[-3.26]	[[0.02 0.016]	[[0.236 0.236 0.236]	[[0.059]	[[0.393]	[0.79 0.135 0.075]	1.464
	[0.299 0.335]	[1.633 1.633 1.633]	[-0.588]	[1.332]	[0.02 0.016]	[-0.117 -0.117 -0.117]	[0.059]	[-0.167]		
	[0.299 0.335]]	[1.329 1.329 1.329]]	[-0.588]]	[1.928]]	[0.02 0.016]]	[-0.119 -0.119 -0.119]]	[0.059]]	[-0.226]]		
2	[[-0.816 -0.737]	[[0.21 0.21 0.21]	[[-3.404]	[[0.739]	[[0.111 0.107]	[[-0.167 -0.167 -0.167]	[[0.282]	[[-0.4]	[0. 0.444 0.555]	4.121
	[-0.816 -0.737]	[1.123 1.123 1.123]	[-3.404]	[-0.132]	[0.111 0.107]	[0.051 0.051 0.051]	[0.282]	[0.146]		
	[-0.816 -0.737]]	[0.168 0.168 0.168]]	[-3.404]]	[-0.607]]	[0.111 0.107]]	[0.116 0.116 0.116]]	[0.282]]	[0.253]]		
3	[[-0.804 -0.736]	[[0.173 0.173 0.173]	[[-3.416]	[[-1.145]	[[-0.001 -0.]	[[0.004 0.004 0.004]	[[0.001]	[[0.188]	[0.588 0.259 0.153]	1.186
	[-0.804 -0.736]	[1.112 1.112 1.112]	[-3.416]	[0.28]	[-0.001 -0.]	[0.001 0.001 0.001]	[0.001]	[-0.041]		
	[-0.804 -0.736]]	[0.216 0.216 0.216]]	[-3.416]]	[0.865]]	[-0.001 -0.]]	[-0.005 -0.005 -0.005]]	[0.001]]	[-0.147]]		
4	[[-0.799 -0.74]	[[0.228 0.228 0.228]	[[-3.44]	[[2.077]	[[-0.001 0.]	[[-0.006 -0.006 -0.006]	[[0.002]	[[-0.322]	[0.078 0.341 0.581]	1.512
	[-0.799 -0.74]	[1.086 1.086 1.086]	[-3.44]	[-0.124]	[-0.001 0.]	[0.003 0.003 0.003]	[0.002]	[0.04]		
	[-0.799 -0.74]]	[0.186 0.186 0.186]]	[-3.44]]	[-1.953]]	[-0.001 0.]]	[0.003 0.003 0.003]]	[0.002]]	[0.282]]		
5	[[-0.78 -0.733]	[[0.141 0.141 0.141]	[[-3.428]	[[-2.745]	[[-0.002 -0.001]	[[0.009 0.009 0.009]	[[-0.001]	[[0.482]	[0.882 0.102 0.016]	1.987
	[-0.78 -0.733]	[1.104 1.104 1.104]	[-3.428]	[1.854]	[-0.002 -0.001]	[-0.002 -0.002 -0.002]	[-0.001]	[-0.198]		
	[-0.78 -0.733]]	[0.256 0.256 0.256]]	[-3.428]]	[0.891]]	[-0.002 -0.001]]	[-0.007 -0.007 -0.007]]	[-0.001]]	[-0.284]]		
6	[[-0.8 -0.756]	[[0.209 0.209 0.209]	[[-3.514]	[[1.186]	[[0.002 0.002]	[[-0.007 -0.007 -0.007]	[[0.009]	[[-0.393]	[0.007 0.728 0.265]	2.485
	[-0.8 -0.756]	[1.009 1.009 1.009]	[-3.514]	[-2.424]	[0.002 0.002]	[0.009 0.009 0.009]	[0.009]	[0.428]		
	[-0.8 -0.756]]	[0.282 0.282 0.282]]	[-3.514]]	[1.238]]	[0.002 0.002]]	[-0.003 -0.003 -0.003]]	[0.009]]	[-0.035]]		
7	[[-0.78 -0.747]	[[0.193 0.193 0.193]	[[-3.491]	[[0.391]	[[-0.002 -0.001]	[[0.002 0.002 0.002]	[[-0.002]	[[0.079]	[0.479 0.014 0.507]	1.793
	[-0.78 -0.747]	[1.04 1.04 1.04]	[-3.491]	[0.441]	[-0.002 -0.001]	[-0.003 -0.003 -0.003]	[-0.002]	[-0.287]		
	[-0.78 -0.747]]	[0.267 0.267 0.267]]	[-3.491]]	[-0.832]]	[-0.002 -0.001]]	[0.001 0.001 0.001]]	[-0.002]]	[0.207]]		
8	[[-0.781 -0.754]	[[0.188 0.188 0.188]	[[-3.523]	[[0.212]	[[0. 0.001]	[[0.001 0.001 0.001]	[[0.003]	[[0.018]	[0.418 0.459 0.123]	1.214
	[-0.781 -0.754]	[0.996 0.996 0.996]	[-3.523]	[-1.145]	[0. 0.001]	[0.004 0.004 0.004]	[0.003]	[0.159]		
	[-0.781 -0.754]]	[0.316 0.316 0.316]]	[-3.523]]	[0.933]]	[0. 0.001]]	[-0.005 -0.005 -0.005]]	[0.003]]	[-0.177]]		
9	[[-0.767 -0.749]	[[0.202 0.202 0.202]	[[-3.512]	[[1.216]	[[-0.001 -0.001]	[[-0.001 -0.001 -0.001]	[[-0.001]	[[-0.1]	[0.3 0.08 0.62]	1.386
	[-0.767 -0.749]	[1.016 1.016 1.016]	[-3.512]	[1.053]	[-0.001 -0.001]	[-0.002 -0.002 -0.002]	[-0.001]	[-0.22]		
	[-0.767 -0.749]]	[0.282 0.282 0.282]]	[-3.512]]	[-2.268]]	[-0.001 -0.001]]	[0.003 0.003 0.003]]	[-0.001]]	[0.32]]		

The loss is quite clearly going all over the place. At the same time, it is quite clear that w1 is being affected properly (just that the previous learning rate was too small to affect it).

And at last, here's the loss function, which is clearly also not converging:


Train Loss
Validation Loss
loss
epoch
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
2
4
6
8