DD2424 Assignment 3 basic

Jacob Malmberg, 198508060558

April 2019

i Analytical gradient check

The analytical gradients were checked against numerical gradients computed by NumericalGradient.m. The relative error was computed following the guidelines laid out on page 7 of Assignment 1:

$$\frac{|g_a - g_n|}{max(eps, |g_a|+|g_n|)} \tag{1}$$

Eps was set to 1e-9. The results are summarized in 1. All checks were done with $n_1 = k_1 = n_2 = k_2 = 5$. Rng was 401, h was 1e-6. According to `http://cs231n.github.io/neural-networks-3/#gradcheck` these are good values which make me conclude that my gradient computations are bug free.

Configuration	Max W relative error	Max F1 relative error	Max F2 relative error
first datapt	7.73e-7	1.20e-8	1.36e-7
first 5 datapts	1.42e-6	9.46e-7	5.19e-7
all datapts	3.84e-4	4.14e-4	7.48e-6

Table 1: Table showing summary of relative errors

ii Compensating for imbalanced dataset

I compensated for the imbalanced dataset by randomly sampling the same number of examples from each class every epoch and making this my training set for the epoch. This number was set to 59, the number of examples belonging to the smallest class in the entire dataset. The training set for every epoch was therefore of size 18*59=1062.

iii Comparing results of using a balanced and imbalanced dataset

Graphs of the loss plots and confusion matrices are included for the balanced and the imbalanced dataset. The loss in both plots was computed every 500th

update step. The networks were trained for 21600 update steps using $k_1 = 5, k_2 = 3, n_1 = 20, n_2 = 20$. The loss plot and the final confusion matrix for the imbalanced dataset is shown in figures 1 & 2. The loss plot and the final confusion matrix for the balanced dataset is shown in figures 3 & 4.

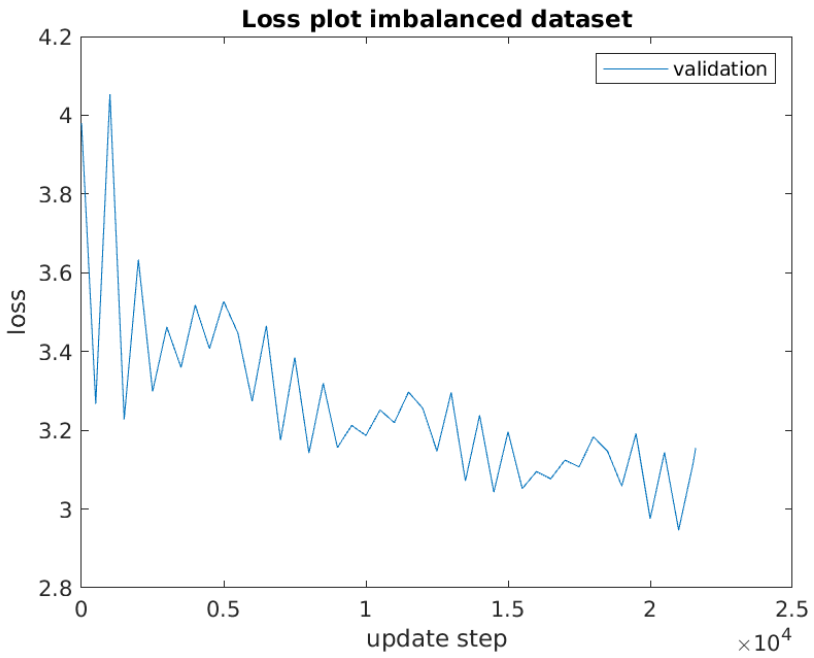

Figure 1: Validation loss plot for imbalanced dataset.

Figure 2: Final confusion matrix for imbalanced dataset.

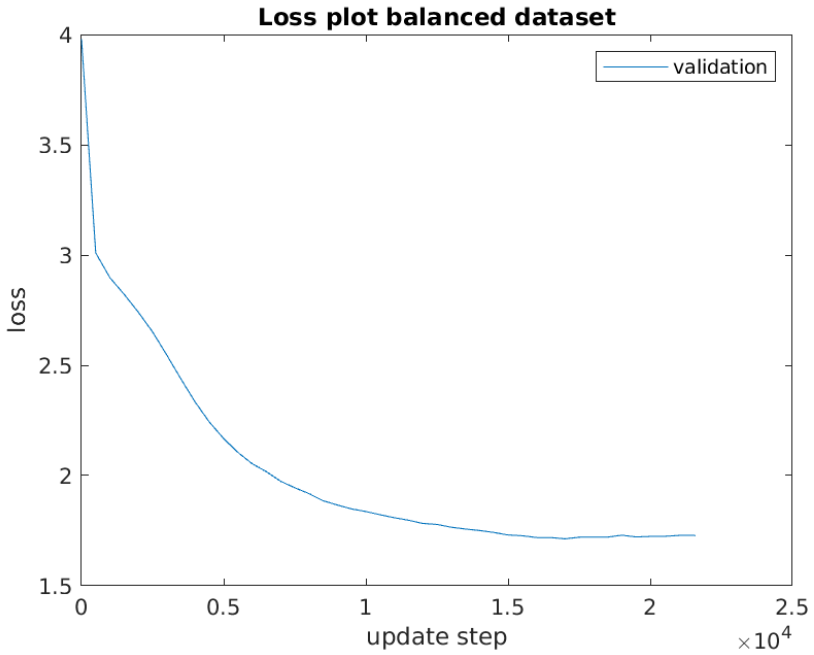

Figure 3: Validation loss plot for balanced dataset.

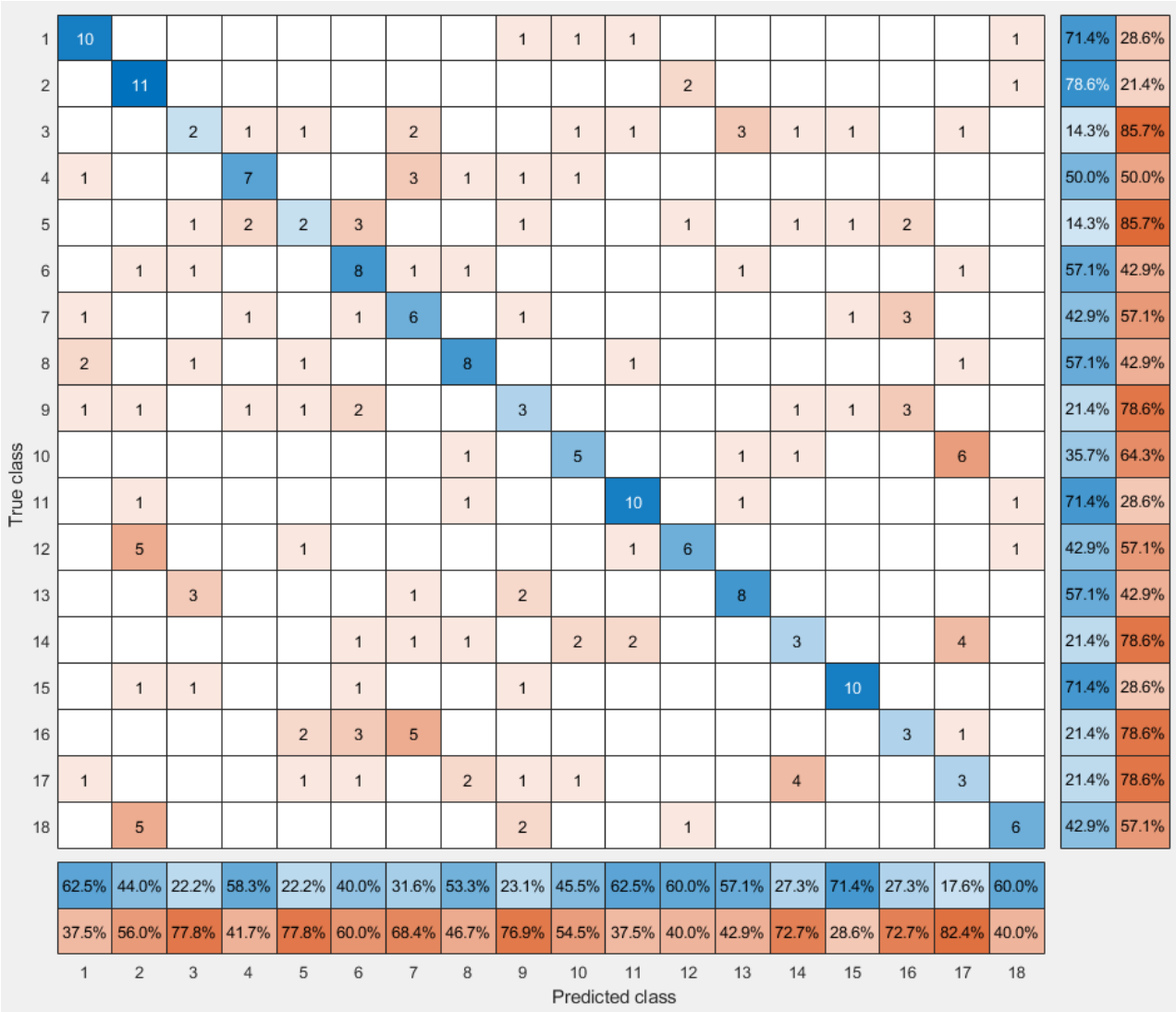

Figure 4: Final confusion matrix for balanced dataset.

The difference between the two training regimes is for the imbalanced dataset, the largest class makes up for about half the datapoints. However, this is only true for the training dataset and not for the validation dataset which is balanced (what about a real life scenario?). Thus, during training the network can easily achieve ∼50% accuracy by just always guessing the largest class. This will make for a pretty nice training loss, which does not translate to the validation set. The class imbalance can clearly be seen in the confusion matrix, fig. 2, where the network makes lots of guesses on class 5 & 15.

The balanced dataset on the other hand has the same ratio of classes in the training & validation set. Thus, it does not pay for the network to always guess on the largest class. This works out well, the guesses are pretty evenly divided among the classes as can be seen in fig. 4.

iv Best performing ConvNet

The best network had a final validation accuracy of 48.02%. The parameter settings are $k_1 = 3, k_2 = 3, n_1 = 40, n_2 = 40$. The network was trained for 21600 update steps with a batch size of 59. Momentum was used with rho being 0.9 and eta was 0.001. The validation loss plot is displayed in fig 5 and the accuracy for each class is displayed in fig fig 6.

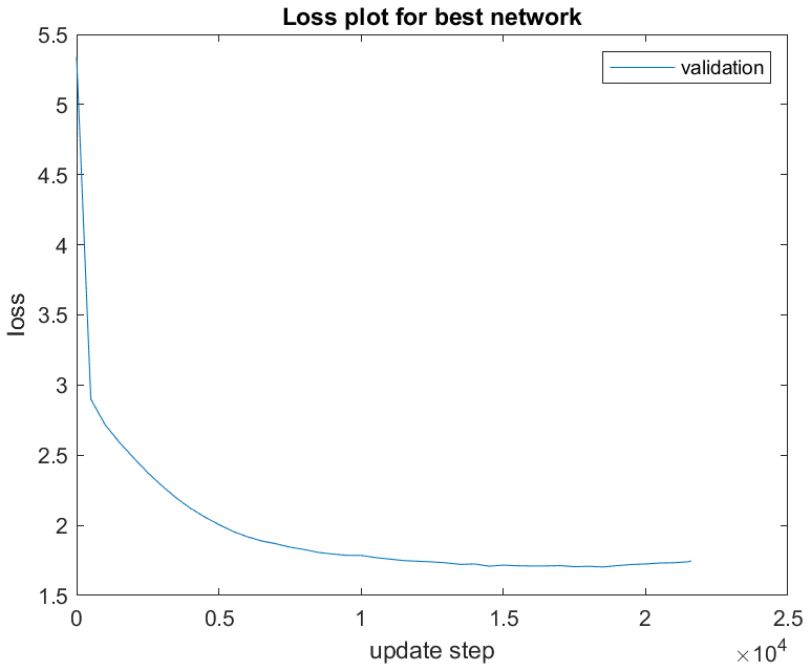

Figure 5: Validation loss plot for best network.

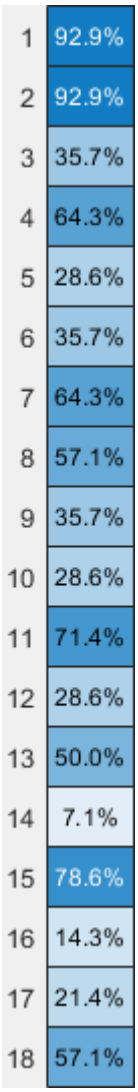

Figure 6: Accuracy for each class. Best network.

v Implemented efficiency gains

I implemented the first listed efficiency trick, pre-computing the first layer MX matrices and making them sparse. The experiment was run for 100 update steps with batch size of 100. Before the upgrade, this experiment took 50.5 seconds. After the upgrade the experiment took 48 seconds, including pre-computing the matrices (which were only used once each so no gain from pre-computing was made). Not including the pre-computing of the matrices the experiment took

17 seconds. Thus, when training my network for 20,000+ update steps where I run multiple epochs and can re-use the MX matrices, much time was saved.

vi Probability vector output of best network on 5 friends

The results are shown in table 2. As for me, Malmberg, mom said we are from Northern Sweden. I am now thinking about a DNA test. Baltatzis is a greek name so 87% greek is good. Karlsson is Swedish, but Dutch and Swedish are reasonably close languages. Zhao is Chinese which the model strongly agrees with. I thought Selmer was Spanish, however the results do not agree with this. I guess german/dutch are both somewhat close to spanish. Singh is not irish/scottish/english but an Indian name. Perhaps the results have something do to with the British colonial efforts in India.

	Malmberg	Baltatzis	Karlsson	Zhao	Selmer	Singh
Arabic	0.0000	0.0000	0.0000	0.0000	0.0010	0.0040
Chinese	0.0000	0.0000	0.0000	0.9601	0.0000	0.0111
Czech	0.0375	0.0002	0.0271	0.0000	0.0088	0.0074
Dutch	0.1334	0.0000	0.5116	0.0000	0.2051	0.0082
English	0.1109	0.0002	0.2065	0.0000	0.0619	0.2002
French	0.0113	0.0005	0.0023	0.0001	0.0180	0.0019
German	0.6860	0.0634	0.0053	0.0000	0.6215	0.0000
Greek	0.0001	0.8776	0.0000	0.0002	0.0000	0.1398
Irish	0.0000	0.0000	0.0178	0.0000	0.0024	0.3080
Italian	0.0000	0.0882	0.0000	0.0000	0.0004	0.0006
Japanese	0.0000	0.00036	0.0000	0.0029	0.0000	0.0018
Korean	0.0000	0.0000	0.0000	0.0041	0.0000	0.0151
Polish	0.0039	0.0000	0.0000	0.0001	0.0200	0.0035
Portuguese	0.0000	0.0000	0.0000	0.0000	0.0001	0.0022
Russian	0.0127	0.0180	0.0041	0.0001	0.0008	0.0443
Scottish	0.0039	0.0029	0.2252	0.0000	0.0546	0.2428
Spanish	0.0002	0.0056	0.0001	0.0003	0.0054	0.0008
Vietnamese	0.0000	0.0000	0.0000	0.0322	0.0000	0.0083

Table 2: Table probabilites of friends/classes