Tianxiao Zhao

tzh@kth.se

[redacted]

Assignment 2 – Optional part

(i) Improvements

1) Applying small random jitter:

Here, I add a random jitter with zero mean and 0.1 standard deviation. The number of epoch for training is set to 25. Other parameter settings are kept the same with the unimproved case. The loss changes are depicted in the following figure. And a 46.08% test accuracy is achieved, which is slightly better than that without improvement (45.23%). This indicates that adding jitters to augment training data can improve generalization a little bit.

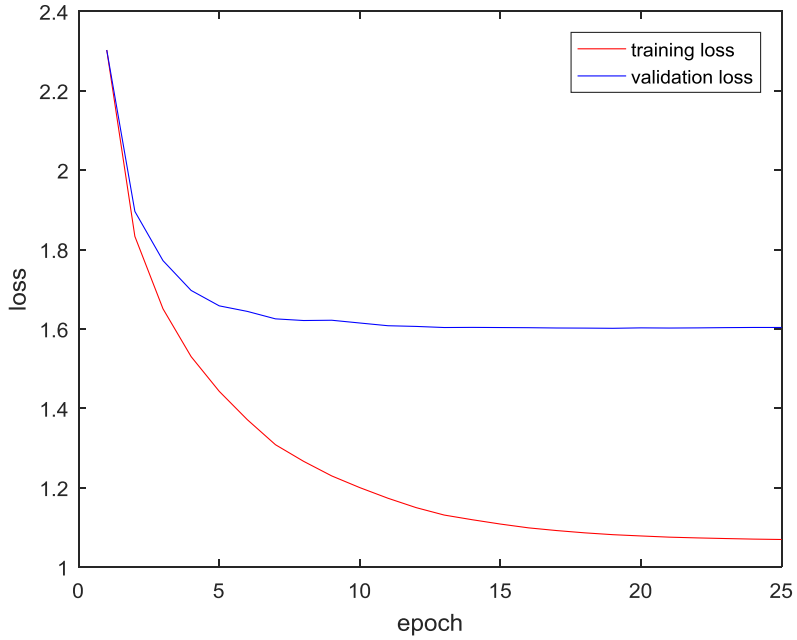

Fig. 1: Training and validation loss with respect to epochs (applying jitters)

2) Adding more hidden nodes:

Another improvement I made is to experiment with more hidden nodes (m = 60). Since more hidden nodes cause a more complex model, the amount of regularization should also increase: lambda = 0.001. Meanwhile, the decay rate is increased to 0.85. The two-layer fully connected network is trained for 20 epochs. The best test accuracy reaches 46.30%.

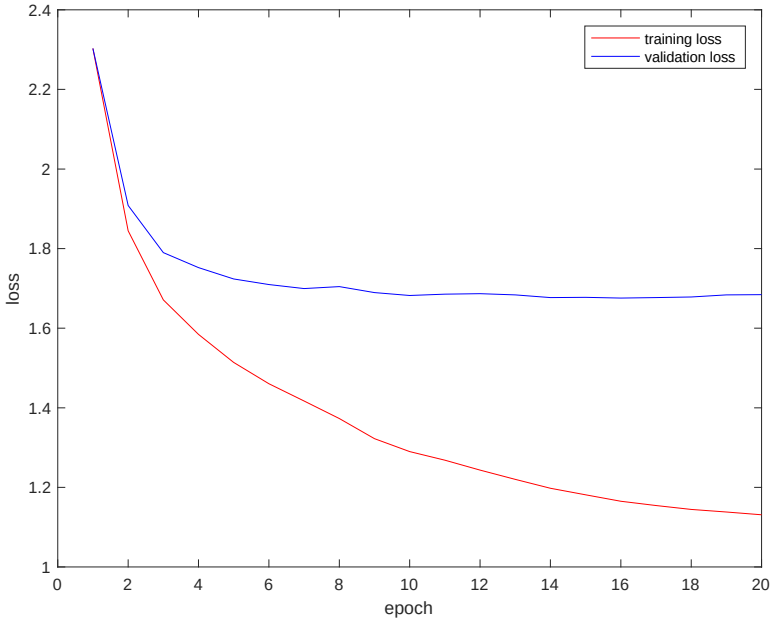

Fig. 2: Training and validation loss with respect to epochs (adding hidden nodes)

3) Annealing the learning rate differently

Instead of decaying the learning rate by a predefined rate (0.8) for each epoch, I choose to reduce the learning rate by a constant (e.g. 0.4) every 4 epochs, and train the network for 20 epochs in total. It gives a 45.67% test accuracy.

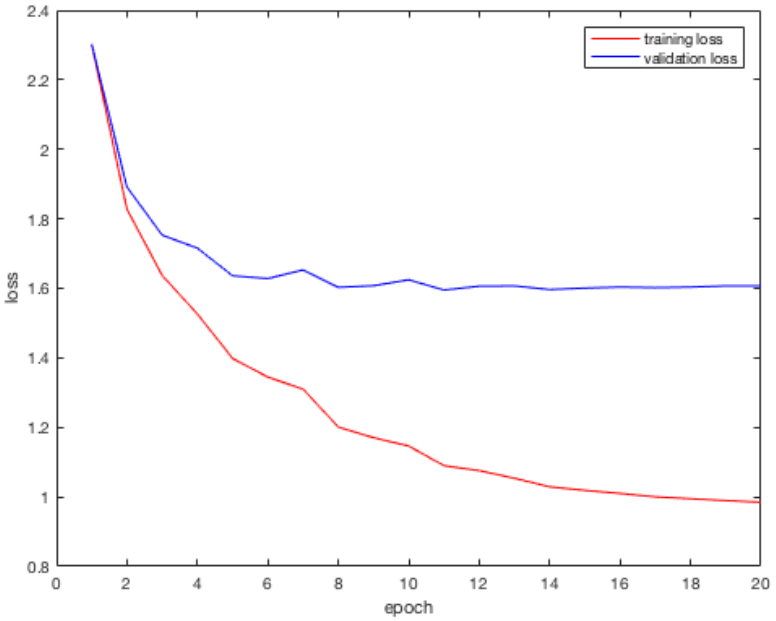

Fig. 3: Training and validation loss with respect to epochs (annealing the learning rate differently)

(ii) Different activation function

I replace the ReLu activation function with a leaky ReLu function. In comparison with the ReLu function, there are two major changes in the code:

1) In the forward pass, the intermediary activation values will be decided by $h = \max(0.1s_1, s_1)$ rather than $h = \max(0, s_1)$.

2) In the backward pass, when computing the derivatives of h with respect to s_1, the equation in the form of vector will become $\frac{\partial h}{\partial s_1} = diag\big(Ind(s_1 > 0)\big) + 0.1 \cdot diag\big(Ind(s_1 < 0)\big)$.

The test accuracy of the network trained with ReLu and leaky ReLu activation function for several sensible parameter settings, which are mentioned in the coarse and fine search, is listed in the table below. The number of epochs is set to 20. It turns out that using leaky ReLu could slightly enhance the test accuracy given the fine parameter settings, and ReLu activation will perform better when hyper-parameters found in coarse search are used. Since we usually use the fine-searched parameters, leaky ReLu should be well considered when training the 2-layer fully connected network.

Tab. 1: Test accuracy for ReLu and leaky ReLu function given different parameter settings

	Fine search			Coarse search		
eta	0.0622	0.0516	0.0306	0.03308	0.02634	0.02279
lambda	1.6424e-6	5.2165e-5	8.2136e-5	2.8311e-3	1.3523e-4	2.7680e-7
ReLu	45.37%	45.22%	44.29%	44.45%	43.95%	43.38%
Leaky ReLu	45.63%	45.44%	44.40%	44.30%	43.63%	42.48%

(iii) Best performance

Combining all the improvements mentioned above, I choose to apply Bagging (Bootstrap Aggregating) to bump up the performance. Here, the training data is enlarged to 50000, and 5 random bootstrap replicates (each with a size of 10000) of the training samples are generated in order to train 5 base classifiers in a parallel way. Besides, leaky ReLu activation function is used as well as the best hyper-parameter settings in fine search. Random jitters are added to the training bootstrap replicates, but the number of hidden nodes is kept to 50. The learning rate is decayed by a rate of 0.8 for every epoch.

The final classification is decided based on the 5 base classifiers with different weights. It achieves a 54.06% test accuracy as my best result.