HW3

**Part 1**

**Ex 1**

First, we’ll show that for any , if for some constant . Since depends only on the input-output mapping of the network, for any set of parameters which defines a network with the 0 mapping as input-output map (i.e. the mapping ), it holds that (when all parameters are zero the input-output mapping is zero).

Assuming , we have:

Where is a zero matrix, and a zero vector. The equation marked as 1 holds because a zero-matrix multiplied with any vector is a zero vector, and all the equations after it (denoted by … up to 2) are due to the assumption that , and the same fact used for equation 1.

We now assume towards contradiction that is convex. from Jensen’s inequality it holds that for any matrices :

Let be an arbitrary set of parameters, and denote .

Plugging into the above inequality we get:

Using the property we proved previously, we then get:

And since are arbitrary we get that is the global minimum of , but this is a contradiction to the assumption that the global minimum is not attained at . Thus is non-convex.

**Part 2**

**Ex 1**

For simplicity, we will denote most vectors without underline, e.g. .

From the lemma we proved in class we have:

Plugging in the update rule for and with we have:

Applying the triangle inequality to the last term, we get:

We now apply expectation over to both sides, preserving the inequality:

Where the last equation is due to the definition of . Because is a random variable dependent only on and all are time-independent, the expectation of the inner product is equal to the inner product of expectations:

Where the second to last equation is from the definition and the last equation is due to the fact that .

Assume that for steps : . Applying the previous inequality times, since it holds for any , we get:

Since is not dependent on any .

Recall that and so and so:

Meaning the first step at which comes before step .

**Ex 2**

Recall that:

Applying expectation we get:

Assume that for steps : . Choosing we will ensure that for any . So, applying the previous inequality times, since it holds for any , we get:

Therefore:

Setting we have for any , and the inequality we required holds. We then get:

From a lower bound on the harmonic partial sum, and an upper bound on the quadratic harmonic partial sum . We then have:

Meaning the first step at which comes before step .

**Ex 3**

Proof of claim 1:

Recall that we saw in class that for a set of matrices :

Where is the product matrix we defined in class.

Let . Then:

We’ll now show bounds on the Frobenius norm of each of these terms.

Since Frobenius norm is sub-multiplicative. We restrict the domain of to vectors in and so is also in , for any :

Also note that since is a compact set and is twice cont. diff., it’s gradient is continuous and therefore norm-bounded over this set (seen in Calculus 2), i.e.:

Put together we get:

Similarly, for the third term in :

Since is twice cont. diff. and is compact and convex, is -Lipschitz on for some (also seen in calculus 2), and so we can calculate for the second term in :

Plugging these results back into we get:

If we chose to be the argmax of we can write:

We’ll now prove by induction that for any and :

For this is trivial (and actually an equality). Assume it holds for :

From the assumption and the fact that :

Where the last equation is from the properties of Frobenius norm. This proves the desired inequality. Note that this can be applied to , but also to . Setting , we get:

And setting in the same way, it holds that:

Since Frobenius norm is invariant to order of the elements and transposing (we changed the order and transposed both terms together).

In the same manner, setting and accordingly, we have:

And plugging these results back into , we finally get:

Where the second inequality is from the non-negative property of norms, adding the norm of . This shows that is -Lipschitz, meaning is smooth.

Proof of claim 2:

Firstly, it’s clear that if is constant then is also constant, and therefor smooth – it’s gradient is always 0, therefore the norm of the differences between the gradients in any two point is 0, and smaller or equal to the norm of the difference between the two points.

Secondly, if is affine and the depth is , we can write:

Plugging in the terms we calculated before:

Note that the sum of squares is equal to the norm:

Since all terms in the sum are non-negative (norm is non-negative), setting we get:

Then is-Lipschitz and therefore is smooth.

We’ll now assume that is smooth and show one of the mentioned conditions holds. Since it is smooth, there exists s.t. for any :

Since norm is non-negative and each coordinate is independent, for any it holds that:

Denote this inequation by .

Assuming , we choose . Let . Plugging into : we get:

And so: .

Since are constant, we have , and from the squeeze theorem we get . The gradient of is independent of , and therefore we have for any , meaning the gradient must be constant, and is affine.

We now assume , and choose . Let and plug into

And so: .

And once again from the squeeze theorem and the fact that is independent of , we get for any , meaning is constant.

**Ex 4**

We used the California house prices dataset. For the loss and gradient magnitude graphs we ran 100 epochs, but computation of hessian eigenvalues was much heavier, so we ran for 10 epochs only. We used a learning rate of 0.0001. The graphs we got are appended at the end of the answer.

For all network depths, the gradient magnitude stayed more or less constant, and grows with network depth. The second part is due to the growing number of parameters – since the magnitude is equal to the norm, it grows with number of parameters. The first fact is due to the networks being linear – they keep improving in the same “direction”, therefore the gradients stay more or less constant, since they are not dependent on the constants added to the initial weights.

The loss decreases monotonically, and the final values shrink with network depth (i.e. the deeper the network, the smaller the final loss). This makes sense since we saw in class proofs on convergence of linear networks, and also saw that depth can help the network to converge faster, even with the same learning rate.

The biggest and smallest eigenvalues of the hessian stay more or less the same over the epochs, which might be because the slope of change for the direction the network goes during optimization doesn’t change much.

Chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart

Description automatically generated

**Part 3**

**Ex 4**

We used the same dataset as in part 2. With both depths, we trained a network with a hidden width of 20 for 100 epochs, with a learning rate of 0.0001. We plotted the norm of the difference between the 2 end-to-end matrices. The plots are appended below (the plot for depth = 2 is also added separately to better observe it’s changes).

For depth 2, the norm mentioned is monotonically increasing over the epochs, but remains quite small overall. It is possible that with more epochs the 2 end-to-end matrices would converge, but this result may also show that the setup is one where the theoretical results we saw do not apply – perhaps because of the amount of data used.

For depth 3, the norm explodes in several points, but is monotonically decreasing for most epochs. This again may show that the setup is not exactly suitable for the theoretical results we saw, but also strongly suggests that with more epochs the 2 end-to-end matrices will have a more similar trajectory and converge to the same solution.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated