**CNN HW2-**

1. c. let’s set activation function to be ReLU and the layer size as 2, then and .

Now we will define the empirical loss as a function of as .

If we find with (no loss) then we have   
 and if for some the loss then we are done.

Notice that if we set the first d-2 layers to be the identity transformation:  
then after applying ReLU on and is positive we still have the identity function.  
from the above claim we get that as long as we can choose as we like and generalize the claim to every by setting the first layers to be the identity (if the claim is incorrect as is just a linear transformation of and the logistic loss is convex in w) transformation.

Set

Now let’s look at the following counter example, we choose the dataset and the classifiers

and and show that the loss is not convex for these examples:

we have so

Output from classifier:

Loss:

Now we define a new classifier like this - and choose   
the output for the classifier is:

And the loss is:

Hence the empirical loss is non convex with respect to

2. we will compute the gradient of step by step.  
mark the dimensions:

first let’s define and we get:

Let’s write the analytical derivatives we will use:

We’ll mark as the -th row of matrix and compute the gradient row wise

When x is a scalar we can use the following identity:  
and when x is a vector of length n we get:

All of the above involves no computation.

Now we start computing the gradients, we make a forward pass and save the intermediate results of the form . (no need to save as we saw earlier that and we have that from the net state).  
This takes time.  
saving the intermediate results will take space

for comfort we will mark the output of the -th sigmoid layer as

Now we will compute the gradients backward using the chain rule and save intermediate matrix multiplication that we will use in the future from each calculation

Gradients w.r.t :

For every calculation we multiply a vector by a sparse matrix where only the  
-th row is non zero, basically we multiply the -th row by the -th index of the vector this takes, time

We will do this time so overall time

Gradients w.r.t :

We need to compute once and save it ( space) for later use, this is done in as the last multiplication is vector by a diagonal matrix.  
Than we multiply the result by the final part for every times in  
 as is mostly zeros except row r , overall we have for this part

Gradients w.r.t :

We already calculated so in order to calculate we only need 2 more martrix multiplications where one is diagonal. So similarly to last step (with different dimensions) we need to perform calculations and then for a total of .  
we saved one vector of length so space.

Let’s sum It all up: space, )