CS 590 Machine Learning
Homework 07
(Ashwini Kulkarni)

1.

A)What is an example data set and problem where you would want to use L1 weight regularization?

B)When would you want to use L2 weight regularization?

C)Explain how the different types of regularization affect a NN's ability to solve the problem given the data set in your example.

Ans:-

A)L1 Regulation:

L1 regularization helps perform feature selection in sparse feature spaces. As->

L1 regularization adds the sum of the |w| to the cost function, the weights shrink by a constant amount towards 0. And the L2 regularization adds the sum of the squares of |w|. As cost function is linear with w, regularization with L1 disables irrelevant features leading to sparse sets of features. Thus, L1 regularization combines efficient feature selection and model generation into one single optimization step. Basically it means, L1 regularization can sometimes have a beneficial side effect of driving one or more weight values to 0.0, which effectively means the associated feature isn't needed.

B)

With using just L1 regulation, sometimes it may not be efficiently approach id doesn't reduce resultant cost. So to minimize the cost function we need to add functionality as derivative of L1 regulation are not defined at vanishing parameter values. So using two steps learning we can use L2 regulation as second stage on selected feature which we got from 1st stage which is L1 regulation.

L2 regularization works with all forms of training, but doesn't give you implicit feature selection

C)

In ans A) and B) we get to know L1,L2 regulation, Sometimes solving NN problems, resultant cost is not efficiently reduced then we can use other Regulation Technique like Drop Out Regulation.

In dropout doesn't rely on modifying the cost function. Instead, in dropout we modify the network itself. In this technique, hidden layer's will randomly choose neurons and only their output will go to next layer and will drop out other neuron.

When we dropout different sets of neurons, it's rather like we're training different neural networks. And so the dropout procedure is like averaging the effects of a very large number of different networks. The different networks will overfit in different ways, and so, hopefully, the net effect of dropout will be to reduce over fitting.

Sometimes given training dataset is not robust for some variation of test data, so make it more robust, create new copy of dataset from existing train dataset and with specific modification to we can add new dataset to Existing dataset, this way by extending training dataset, it can keep up variable test data. This regulation technique is called as artificially expanding dataset.

2. Explain, in your own words, the following problems and ways to avoid/solve them:

A. The Vanishing Gradient problem

In training NN's with gradient based methods such as Back Propagation it makes really hard to learn and tune the parameters of the earlier layers in the network. This problem becomes worse as the number of layers in the architecture increases.

Gradient based methods learn a parameter's value by understanding how a small change in the parameter's value will affect the network's output. If a change in the parameter's value causes very small change in the network's output - the network just can't learn the parameter effectively. This problem is called as Vanishing Gradient Problem.

We can avoid this problem by using activation functions which don't have this property of 'squashing' the input space into a small region. A popular choice is Rectified Linear Unit which maps x to max(0,x).

B. The Exploding Gradient problem:

In training NN's, sometimes the gradient gets much larger in earlier layers, called as exploding gradient problem. This is when gradient get exponentially large from being multiplied by numbers larger than 1. Gradient clipping will clip the gradients between two numbers to prevent them from getting too large.

Gradient Clipping is a technique to prevent exploding gradients in very deep networks, typically Recurrent Neural Networks.

new_gradients = gradients * threshold / l2_norm(gradients).

To avoid exploding gradients we can use L1 or L2 penalty on the recurrent weights.

C. The Unstable Gradient problem

The fundamental problem of the vanishing or exploding gradient problem comes from the fact that shallow layers consist of the multiplication of the gradient of deeper layers. This leads to an unstable configuration which can, due to the random initialization, either swing towards exploding gradients or vanishing gradients and then, as a result, the different layers will learn at vastly different speeds.

According the Xavier Initialization by Glorot et. al. proposes a specific initialization method which can able to solve this problem. They recommend scaling the gradients by dividing it by the square root of the number of inputs for every single neuron. So if a neuron has many inputs, it will end up with lower weights which intuitively makes sense because the inputs that goes into the weighted sum should have less of an interaction.