**1. What’s the difference between logistic regression and linear regression? Is logistic regression a linear model?**

**2. What’s LOSS function with single example or total examples? And try to give the ‘Logistic LOSS function’.**

**3. What does a neuron compute?4. Given J(w) is model loss, try to give the gradient descent process to update the model parameters ‘w’.**

1.The objective of linear regression is to minimize the regression loss. Logistic regression limit the output values range from 0 to 1 to minimize the classification loss. Linear regression requires that the dependent variable be a continuous numerical variables, while logistic regression needs a discrete variables.

If the interface(or decision boundary) is linear, we say the model is linear model, thus logistic regression is a linear model.

1. Loss function ：Computes the error (discrepancy between output and ground truth) of single sample or total samples.

Cost function：Average Loss of entire training set(or a single mini-batch).



1. A neuron computes a linear function (z =Wx+b) followed by a activation function(sigmoid, tanh, ReLU…)
2. The gradient descent process is defined as:

 Where learning rate is determined manually.

1.3

**1. Why we need activation function in neural network?**

**2. You initialize the weights to relative large values in a neural network which using ‘tanh’ activation function for all hidden units. What will happen?**

**3. Why we should initialize weights randomly rather than to all zeros in neural network? Is it stay true for logistic regression?**

1. Neural network aims to implement complex mapping functions hence they need non-linear activation functions to bring in the needed non-linearity property that enables them to approximate any function. It doesn’t make sense to build a multi-layer network without activation function because it just equals to a single layer network.



Thus ‘tanh’ becomes flat for large values, this leads its gradient to be close to zero and slows down the optimization algorithm.

1. If we initialize it all with zeros, all the neuron output would be the same for any example. The weight update are also the same so that all the neuron would perform a same function no matter how many iterations. It’s not helpful because we want the different unit calculate the different function.

The derivatives of the logistic regression directly depend on the input (because there’s no hidden layer) which is not zero, so it doesn’t have the problem of “symmetry” and can be initialized with zeros.

2.1

**1. What does the training set, the validation set and the test set used for?**

**2. Must the training set, the validation(dev) set and the test set all come from the same distribution?**

**3. What does high bias and high variance means? Can they happen simultaneously?**

**4. why should we make a careful choice of initialization weight in a very deep NN?**

**5. What is regularization? Why it can prevent overfitting?**

**6. What’s drop-out? Why it helps reducing overfitting?**

1. Training set is used for models training, validation set is used to judge a best model, and test set shows us how well our model’s doing.

The main difference between validation set and test set is that the validation set is used to further determine the hyper-parameters in the model in train phase.

The test set is only used to evaluate the accuracy of the model in test phase.

1. It’s common issue that the training set and test set have mismatched distribution because the data may come from different sources.

But we must make sure the validation set and test set come from the same distribution, so that validation set can help us find the best test model.

1. High bias means the train set error is high, the model is not doing very well on a problem i.e. underfitting.

High variance means that the validation set error is much higher than train set error i.e. overfitting.

They can happen at the same time (overfitting in a part of data, and underfitting in another part).

1. Very deep NN would be more susceptible to gradient vanishing/exploding if all layer of weights are small/large.

The larger number of units, the smaller variance of weights should be chose.

1. Regularization is the addition of constraints to minimize the empirical error function. Such constraints can be interpreted as prior knowledge. The constraint has a guiding effect.

When optimizing the error function, it tends to choose the direction of the gradient reduction that satisfies the constraint, so that the final solution tends to conform to the prior knowledge (such as the general l1-norm, indicating that the original problem is more likely related to less parameters.).

1. In the training epoch, drop-out makes some neuron output go away (set to zero) with a certain probability.

in the test epoch, we do not perform drop-out but rescale the outputs by divided by keep-probs to maintain the same scale as training epoch.

It makes a neuron can’t rely on any one feature, so the network have to spread out weights.( Reducing the interaction between feature detectors, which means that some detectors rely on specific detectors in function.)

2.2

**1. Does mini-batch gradient descent’s performance always lower than full-batch gradient descent?**

**2. What is exponential weighted average? What is bias correction of exponential weighted average?**

**3. Why should we decay the learning rate during training epochs?**

**4. What is “local optima” problem at high-dimensional inputs? What can be done to this problem?**

**5. Try to write down the Adam’s formulas.**

1. No.

Full-batch gradient descent may fall into the saddle point or worse local minimal.

Mini-batch gradient descent can jump out these points because each iteration the training samples are different.

If the sample size is very small, the batch gradient descent algorithm is adopted, otherwise the small batch gradient descent algorithm should be used.

1. Exponential weighted average use the actual value and the estimated value of the previous period to make a prediction for the next estimated value, which is calculated as:



is bias correction which is used to correct the results in initial phase of training and helps getting better estimation early.

1. Setting the size of the learning rate is to find a balance between precision and speed.

If the learning rate is high, the training speed will be improved, but the accuracy of the result is not enough.

If the learning rate is small and the accuracy is improved, the training will consume too much time.

The degraded learning rate is also called the decay of learning rate. When the training starts, the high learning rate is used to accelerate the speed, and the small learning rate is used to improve the accuracy after the training reaches a certain level.

1. It is unlikely that there will be in local optimality in a high-dimensional space, because that means the gradient of all dimensions be zero, which is almost impossible.

The real problem of “local optima” is the stagnation region near the saddle point. The saddle point is not the best point, but the standard gradient descent algorithm will cost lots of time to across this point.

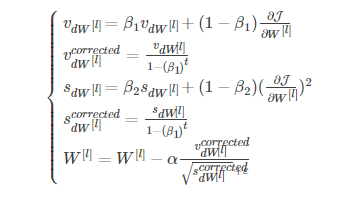
Momentum, Adam, RMSprop can be adopted to jump out the “local optima” and speed up training.

1. L is the number of layers

β1 and β2 are hyperparameters that control the two exponentially weighted averages. In general,β1=0.9 β2 =0.999

α is the learning rate

ε is a very small number to avoid dividing by zero ,usually set 10^−8



2.3

**1. Is grid search a good hyper-parameter searching algorithm for deep learning? Why?**

**2. If we have a very large training set but not have enough computational resources, how can we find good hyper-parameters?**

**3. How to use batch normalization in training phase and test phase?**

**4. Why does batch normalization work?**

**5. Try to write down the softmax function and it’s derivative.**

1. No.

Grid search is a simple way to try all possible configurations. This method can be used when the number of hyper-parameter is small. But deep learning network lots of hyper-parameters. This parallel strategy can be inefficient because it doesn't take hyper-parameters’ importance into account.

Even if it guarantees the ultimate optimal solution, it is still not desirable. Instead, it's best to use a random search.

1. If we don’t have much computational resources, construct a babysitting model may be a good choice. It allows you to change the hyper parameters during training progress by manual.

If we don't have enough manpower to adjust the hyper-parameters. Bayesian optimization can be used to learn a acceptable result automatically. It use a prior knowledge to approximate the posterior distribution of an unknown objective function to adjust the hyper-parameters.

1. Batch normalization scales the input of each layer (the output of previous one, before activation) with by their mean/variance. And add two parameters to learn a suitable mean/variance of input data.

In the training process, for each iteration BN layer use exponentially weighted average to save the mean/variance values of layer inputs.

In the test phase, use the stored mean/variance value to scale input.

4 It scales the input of each layer to close range and solve the gradient vanish/explosion problem in the process of back propagation.

It can reduce the problem of the input values changing (or “Covariate Shift”), and make training stable.

It add some noise to the input of each layer because each batch is scaled by the mean/variance computed on this current batch. So it has a slight regularization effect and avoid overfitting.

5 Softmax regression generalizes logistic regression to multiple classes.

If we have C classes and the inputs of softmax are a1,a2..ac, then softmax compute the probability of predicted sample belongs to the i-th input:



Each output of softmax function contains all inputs, so the derivative of softmax is:



**4.1**

1. **Why should we use convolutional layer in large images computer vision task?**
2. **Why should we use pooling layer in computer vision task?**
3. **How to compute the gradient of max/average pooling layer?**

1. If we use fully connect layer in a large images, the number of parameters would be too large and cause time consuming and overfitting.

The convolutional layer avoid these problem by using parameter sharing (Small feature filters are used in all parts of the image) and sparsity of connections (each output value is only depends on small number of inputs).

2. Pooling layer is used to further reduce parameters and maintain invariance of images (like rotation, translation, expansion). The pooling layer also can reduce the mean shift caused by convolution layers.

(Max pooling: take the maximum value in pooled visual field Average pooling: take the Average of the pooled visual field)

3. For max pooling, the position of the maximum value in the input data is recorded in forward propagation and transmit the gradient value to the corresponding position of the maximum value in back propagation.

For average pooling, we don’t need to save the position like max pooling. The gradient value would evenly transmit to all position in back propagation.