**Compute derivative of following activation functions: (3 points)**

**Sigmoid:**

1. ad

**tanh:**

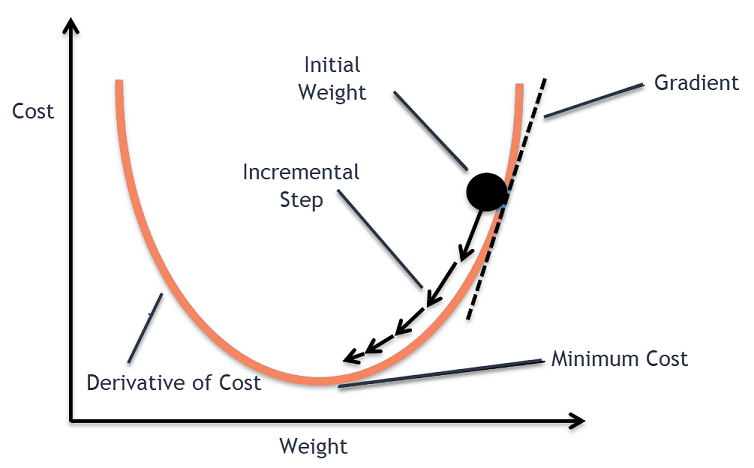
1. s

**ReLU:**

1. dd

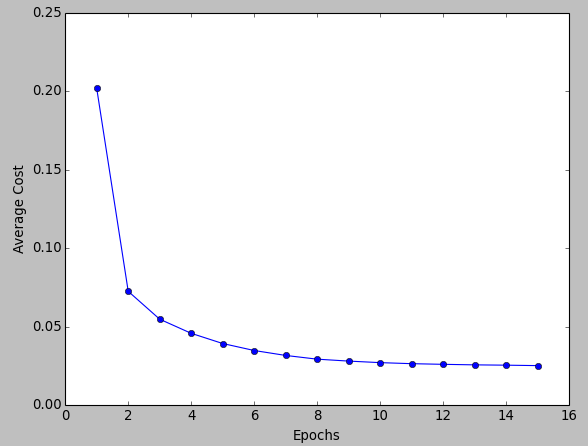
**What are the strategies you will follow to avoid over-fitting in a neural network (2 point)**

**Variants of gradient descent’s (10 points).**

1. Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function
2. Gradient descent is an optimization algorithm that follows the negative gradient of an objective function in order to locate the minimum of the function.
3. The goal of the gradient descent is to minimise a given function which, in our case, is the loss function of the neural network
4. Compute the slope (gradient) that is the first-order derivative of the function at the current point
5. Move-in the opposite direction of the slope increase from the current point by the computed amount
6. 
7. So, the idea is to pass the training set through the hidden layers of the neural network and then update the parameters of the layers by computing the gradients using the training samples from the training dataset.

**Batch Gradient Descent**

1. In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that’s just one step of gradient descent in one epoch.
2. Batch Gradient Descent is great for convex or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution.



1. The graph of cost vs epochs is also quite smooth because we are averaging over all the gradients of training data for a single step. The cost keeps on decreasing over the epochs.

**Stochastic gradient descent**

1. In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in one epoch for SGD:
   1. Take an example
   2. Feed it to Neural Network
   3. Calculate it’s gradient
   4. Use the gradient we calculated in step 3 to update the weights
   5. Repeat steps 1–4 for all the examples in training dataset
2. Since we are considering just one example at a time the cost will fluctuate over the training examples and it will not necessarily decrease. But in the long run, you will see the cost decreasing with fluctuations.



**Mini batch gradient descent**

1. since in SGD we use only one example at a time, we cannot implement the vectorized implementation on it. This can slow down the computations. To tackle this problem, a mixture of Batch Gradient Descent and SGD is used.
2. We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch
3. after creating the mini-batches of fixed size, we do the following steps in one epoch:
   1. Pick a mini-batch
   2. Feed it to Neural Network
   3. Calculate the mean gradient of the mini-batch
   4. Use the mean gradient we calculated in step 3 to update the weights
   5. Repeat steps 1–4 for the mini-batches we created

**Momentum**

1. Momentum is an extension to the gradient descent optimization algorithm that allows the search to build inertia in a direction in the search space and overcome the oscillations of noisy gradients and coast across flat spots of the search space.
2. It is designed to accelerate the optimization process, e.g. decrease the number of function evaluations required to reach the optima, or to improve the capability of the optimization algorithm, e.g. result in a better final result.
3. Momentum involves adding an additional hyperparameter that controls the amount of history (momentum) to include in the update equation

**Adam**

1. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.
2. Adaptive Gradient Algorithm maintains a per-parameter learning rate that improves performance on problems with sparse gradients

**Loss of Functions (3 points)**

**MeanSquareError(MSE)**

**MeanSquareError used in gradient descent**

**Binary CrossEntroy(BCE)**

**Binary CrossEntroy used in neural networks**

**Categorical Cross Entropy (CC)**

**Categorical Cross Entropy used in neural networks**

Ref:

Variants for gradient descent

[Gradient descent - Wikipedia](https://en.wikipedia.org/wiki/Gradient_descent#:~:text=Gradient%20descent%20is%20a%20first,the%20direction%20of%20steepest%20descent.)

[Batch, Mini Batch & Stochastic Gradient Descent | by Sushant Patrikar | Towards Data Science](https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a)

[Gradient Descent With Momentum from Scratch (machinelearningmastery.com)](https://machinelearningmastery.com/gradient-descent-with-momentum-from-scratch/)

[Gentle Introduction to the Adam Optimization Algorithm for Deep Learning (machinelearningmastery.com)](https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/)

Loss functions derivation

[In machine learning how can I calculate the gradient of a function? (Ex: sigmoid, tanh, etc.) - Quora](https://www.quora.com/In-machine-learning-how-can-I-calculate-the-gradient-of-a-function-Ex-sigmoid-tanh-etc)

Avoid Overfitting

[5 Techniques to Prevent Overfitting in Neural Networks - KDnuggets](https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html)