Activation Functions

Sr. #	Pros	Cons	Use

Sigmoid Activation Functions

1.	This is a Smooth function that facilitates derivation	Vanishing Gradient problem. Sigmoid saturate and kill gradients because it maps value between the range of: $0 \leq \partial(z) \leq 0.25$	Usually, used in output layer of a binary classification where result is either 0 or 1, as value for sigmoid function lies b/w 0 & 1 only so, result can be predicted easily to be 1 if value is greater than 0.5 & 0 otherwise.
2.	It can compress the data to ensure that the data amplitude will not the problematic.	The output isn't zero-centered thus the gradient updates go too far in different equations i.e. $0 <$ output < 1, and it makes optimization harder	
3.	Easy to compute differential	Sigmoid have low convergence	

Tanh Activation Function

1.	It is continuous and differentiable at all points	Gradient disappearance problem.	Usually used in hidden layers of a neural network as it's values lies b/w -1 to 1 hence the mean for the hidden layer comes out be 0 or very close to it, hence helps.
2.	It compresses the input values to a range of -1~1, so, it is 0-mean, which solves the non-zero-centered problem of the sigmoid function.	The gradients are low.	
3.		Power operation problem.	

ReLU Activation Function

Neural Networks

1.	ReLU function is non-linear, which means we can easily back-propagate the errors and have multiple layers of neurons being activated by the ReLU function.	The output is not zero-centered.	ReLU is less computationally expensive than tanh & sigmoid because it involves simpler mathematical operations. At a time, on;ly a few neurons are activated making the network sparse making it efficiently & easy for computation. In simple words, ReLU learn much faster than sigmoid & Tanh function.
2.	It was found to greatly accelerate the convergence of stochastic gradient descent (SGD) compared to the sigmoid and tanh functions.	Dead-Neural Network/ReLU Problem: Some neurons may not be activated. Which ultimately responsible to not updated the new weight means Wnew=Wold No pjenomenon happens just consume the computational function.	
3.	It doesn’t activate all the neurons at the same time. Since the output of some neurons is 0, only a few neurons are activated making the network sparse, efficient and easy for computation.	Non-differentiable at zero.	

Leakly ReLU Activation

1.	No dying ReLU units	It saturates for large negative values, allowing them to be essentially inactive.
2.	It speeds up training. There is evidence that having the “mean activation” be close to 0 makes training faster	

Neural Networks

Softmax Activation Function

It is not a traditional activation function. Other activation functions produce a single output for a single input. In contrast, softmax produces multiple outputs for an input array. Thus softmax can be used to build neural networks models that can classify more than two classes instead of a binary class solution.

1.	It mimics the one hot encoded labels better than the absolute values.	The softmax function should not be used for multiclass functions	Usually used when trying to handle multiple classes.

Loss Functions

1.	**Regression Loss Function**	Mean Squared Error (MSE) Loss
2.	**Binary Classification Loss Functions**	• Binary Cross Entropy • Hinge Loss • Squared Hinge Loss

Binary Cross Entropy

It will work best when the data is normalized (forced b/w 0 to 1) as this will represent it as a probability. This normalization property is common in most cost functions.

Hinge Loss

- This is most popular loss function during pre-deep learning era that is oftenly used for binary classification problems.
- This type of loss is primarily used in SVM classifiers where the target values are in the set {-1, 1}
  Hinge loss not only penalizes the wrong predictions but also the the right predictions that are not confident.

1.	**Multiclass Classification Loss Functions**	• Multi-class cross entropy loss • Sparse Multiclass Cross-Entropy Loss. • Kullback Leibler Divergence (KL-Divergence) Loss

Optimizers

Optimizers are methods or algorithms which is used to update the attributes in neural networks such as Weights during backpropagation to reduce the loss.

Types of Optimizers

	Types of Optimizer	Advantages	Disadvantages
1.	**Gradient Descent**		

	It is a first order optimizer which actually depends on the 1st order derivative of loss function. Through backpropagation, the loss is transferred from one layer to another and weights are updated so, that the loss can be minimized	Easy Computation & Implementation.	• May trap at local minima. • Weights are changed after calculating Gradient on the whole dataset.
2.	**Stochastic Gradient Descent** It tries to update the model parameters more frequently. So, the weights are updated after computation of loss on each training. e.g. if dataset contains 1000 rows SGD update weights 100 times in 1 cycle instead of 1time like gradient descent	• Frequently updates of model parameters hence, convergence in less time. • Requires less memory as there is no need to store values of functions	• High variance of model parameters as model are frequently updated.
3.	**SGD with Momentum** To reducing the high variance in SGD invented the momentum which accelerates the convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction. One more hyperparameter is used in this method.	• Converges faster than gradient descent. • Reduces the high variance.	• Additional parameter is used in this method which needs to be selected manually.
4.	**Adagrad** This optimizer changes the learning rate for each and every step and for every parameter. Whereas, other optimizers do not change the optimizers or having constant learning rate.	• Learning rate changes for each training parameter. • Do not need to manually tune the learning rate. • Able to train on sparse data	• The learning rate is always decreasing result in slow training.