Neural networks notes

1. **Notes on cross\_val\_score**
   1. Use this as a performance evaluation tool of the model we make and mitigate over-fitting.
   2. The main idea is to-
      1. We split our initial dataset, in most case the X\_test from the train\_test\_split.
      2. For each epoch, we partition the data into k folds.
      3. Hold out a set at a time and train the model the rest of the k-1 folds.
      4. Test model on hold out set and retain the evaluation score.

A screenshot of a cell phone

Description automatically generated

* 1. **Types of Cross Validation**
     1. **K-Fold Cross Validation**
     2. **Stratified K-fold Cross Validation**- Splitting of data into folds may be governed by criteria such as ensuring that each fold has the same proportion of observations with a given categorical value. Example- Male or female
     3. **Leave One Out Cross Validation-** Train on N-1 points and test on the Nth point.
  2. **Deciding on a value for k**- k is chosen such that each train/test group of data samples large enough to be statistically representative of the broader dataset. Usually a value of k=10 is common. It is preferable to split the data evenly.

1. To use cross\_val\_score which sklearn and we want to apply a keras model to cross\_val\_score, use kerasClassifier
2. **Dropout Regularization-** To counter overfitting. The idea behind Dropout Regularization is to disable a fraction of the neurons/set their weight to 0 at any given layer. In terms of coding it, we add a dropout layer in between the layers and define the percentage of neurons to disable.
3. **Parameter Tuning-** The idea is to find better values for hyperparameters like number of epochs, batch size, training optimizer (like ‘adam’, etc), activation functions, dropout regularizion, learn rate, momentum. We can achieve this using GridSearchCV which tries all possible combinations of the hyperparameter we want to optimize for. GridSearch does a k\_fold validation on each of the combination of parameter to optimize for.
4. **Great reading for CNN working-** <https://missinglink.ai/guides/keras/keras-conv2d-working-cnn-2d-convolutions-keras/>
5. **Fitting test data for images in CNN- technique- Image augmentation**
   1. The main idea behind image augmentation is to divide the test data in batches and apply random transformations like rotate, flip, etc in order to develop more correlations and hence prevent overfitting by enriching the test dataset in a way
   2. Overfitting usually occurs due to lack of test data and the neural net makes limited number of correlations based on the test data.
   3. Check the code in the keras documentation -> preprocessing -> image processing
6. **Notes on loss functions and choosing loss functions**

The problem of leaning in when training a neural network can be viewed as a search or optimization problem in terms of finding a set of weights the model can use in order to make good enough predictions. Concept of gradient- In each epoch we are moving down the gradient of error to reduce error.

The function we are trying to minimize or maximize is the OBJECTIVE FUNCTION. When we are minimizing it, we may also call it the loss function. Choosing a loss function is important as the function has to capture proper properties of the problem.

* 1. **Regression Loss Functions**
     1. **Mean Squared Error Loss**
     2. **Mean Squared Logarithmic Error Loss**
     3. **Mean Absolute Error Loss**
  2. **Binary Classification Loss Function**
     1. **Binary Cross-Entropy**
     2. **Hinge Loss**
     3. **Squared Hinge Loss**
  3. **Multi Class Classification Loss Function**
     1. **Multi Class Cross Entropy Loss**
     2. **Sparse Multiclass Cross Entropy Loss**
     3. **Kullback Leibler Divergence Loss**

1. **Notes on optimizers**
2. **Notes on choosing activation functions**

The main idea behind an activation function is to decide whether or not the neuron should activate based on the value generated by the neuron- + bias

1. Step Function
   1. Activate the neuron if the value is above a certain threshold
   2. Good for binary classification
   3. Does not work for multi class classification, since all neuron might activate if the threshold requirement is met
   4. In such cases we need function that work in proportion to the input -🡪 which brings in point number 2.
2. Linear Function
   1. A = cx
   2. Here neurons are activated in proportion to the input.
   3. Does not work well for multiple layers, if all layers have a linear activation function, that is equivalent to having just one layer.
   4. Gradient descent does not work well, since the derivative of the linear function is a constant, so the changes made by the back propagation is constant and does not depend on the change in input.
3. Sigmoid Function----main problem- Vanishing Gradient problem
   1. A= 1/(1+e^-x)
   2. Smoother step function
   3. Works well with binary and non binary classification
   4. Analyzing the curve further, a small change in X brings about a large change in Y which works well for binary information.
   5. Range of values are always between 0 and 1 rather than -inf and inf
   6. A minor disadvantage is the flattening of the curve at the extremes, gradient is small, learning for the network becomes very small.
4. Tanh function
   1. A variation of the sigmoid
5. ReLu
   1. A(x) = max(x,0)
   2. Non linear in nature
   3. Good approximator and any combinations of ReLu is also Non Linear, using multiple layers is possible
   4. This function is great in terms of activating only a proportion of neuron, this is not offered by the sigmoid or tanh to a large extent since they are essentially smoother step functions. 🡪sparse activation
   5. Dying ReLu problem- for the flat portion the gradient is 0 so for any negative values the gradient is 0 so no response in change of weights, potentially making a considerable part of the network to be passive.

Workaround- make the flat portion slight inclined, for example y = 0.01x for x<0 -> main idea- make the gradient any value other than 0.

* 1. Computationally less expensive.