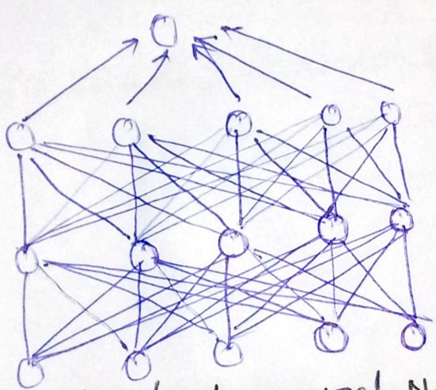
01 Oct 2024

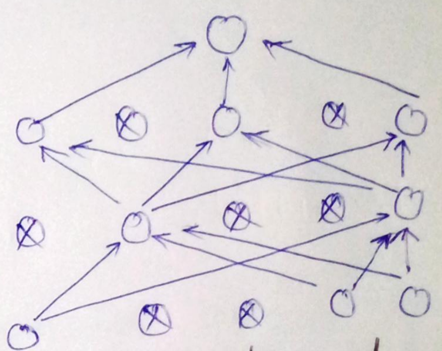
How to Handle Overfitting in Neural Network?

- add more data
- Reduce the complexity (example. reduce number of layer (Hidden layer), number of. neurons in each layer
- early Stopping
- Use Regularization (L1 and L2)
- Dropout

1. Dropout:



standard neural Net



After Applying dropout

- Randomly input and hidden layer's node remove in each epoch (example epoch = 10)
- OR alternatively we can say that in each epoch. our model is train in 10 different neural network

Handle overfitting in NN using Dropout:

(1.) in neural network there is many node possibilities and they are captures the different possibilities and our model overfit,

due to this our model overfit,

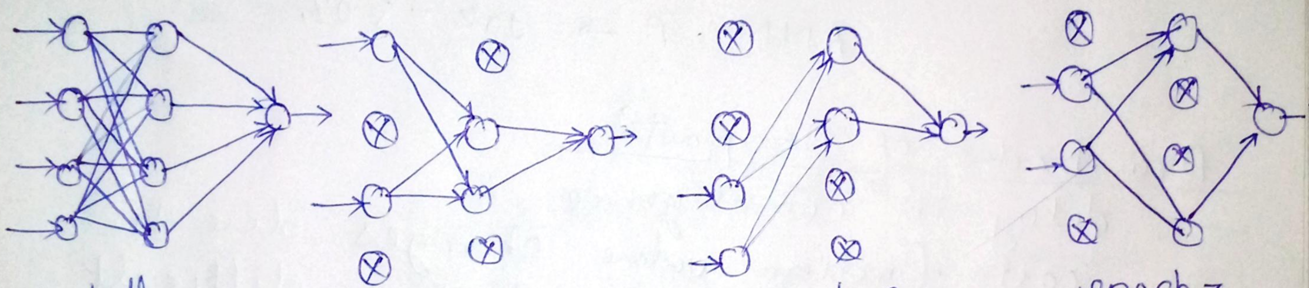
To resolve this issue reduce the number of nodes in neural network.

(2) build the neural network in such a way that where nodes are not biasedly focus on particular feature/pattern.

- Dropout ratio $p = 0.5$ means in each layer 50% of neurons are randomly remove.

How dropout works:

Here the dropout ratio for each layer is 0.5



initially   epoch 1   epoch 2   epoch 3

- Dropout work similar as a Random forest

Ques How Dropout working is similar as a Random forest?

- dropout is only apply in training time.
- After apply the dropout the accuracy may be increase by 2% and also handle the overfitting.
- In the testing time all the weight and neurons are present.

if $p = 0.25$

w w' in testing

in training, $w' = w(1-p)$

practical tips for dropout:

- if overfitting increase p and for underfitting decrease the value of p
- not apply dropout on all layer of neural network first apply only on last layer and then proceed to inner layer according performance.
- for CNN $p \rightarrow 40\% - 50\%$

  RNN $p \rightarrow 20\% - 30\%$

  ANN $p \rightarrow 10\% - 50\%$

Drawbacks of Dropout:

- delay in convergence
- cost function value changes due to this calculation of gradient is difficult

2. Regularization:

- in most of the time for handling the overfitting in neural network. we use the L2 regularization. in L1, L2 and L1+L2.

- in any ANN our aim is to find the weight and / bias for minimizing the loss function/ cost function

$$C = \frac{1}{n} \sum_{i=1}^{n} L(Y_i - \hat{Y}_i) + \text{penalty term}$$

in L2. regularisation

penalty term is $\frac{\lambda}{2n} \sum_{i=1}^{K} \|W_i\|^2$

example $\frac{\lambda}{2n} [W_1^2 + W_2^2 + W_3^2 + --- W_n^2]$

- Here λ is hyperparameter. the higher value of λ means the penalty term's weightage increase. (moving toward overfitting to underfitting)

in L1 regularisation:

$$\text{cost function} = \frac{1}{n}\sum_{i=1}^{n} L(Y_i - \hat{Y}_i) + \frac{\lambda}{2n}\sum \|W\|$$

- for accurate representation of penalty term in NN

$$\sum_{l=1}^{L}\sum_{i=1}\sum_{j=1} \|w_{ij}^{l}\|^2$$

intuition behind the Regularization:

weight update formula in neural network:

$$w_n = w_0 - \eta\frac{\partial L}{\partial w_0} \quad \text{——— (i)}$$

loss function

$$L' = L + \frac{\lambda}{2}\sum \|W_i\|^2$$

$$\frac{\partial L'}{\partial w_0} = \frac{\partial L}{\partial w_0} + \frac{\lambda}{2} 2w_0$$

$$\frac{\partial L'}{\partial w_0} = \frac{\partial L}{\partial w_0} + \lambda w_0$$

Now, $w_n = w_0 - \eta\left(\frac{\partial L}{\partial w_0} + \lambda w_0\right)$

$$= w_0 - \eta\lambda w_0 - \eta\frac{\partial L}{\partial w_0}$$

weight decay

$$w_n = (1-\eta\lambda)w_0 - \eta\frac{\partial L}{\partial w_0} \quad \text{——— (ii)}$$

Here we try to reduce the weight