Backpropagation

Backpropagation Algorithm

Step 1:

Inputs X, arrive through the preconnected path.

Step 2:

The input is modeled using true weights W. Weights are usually chosen randomly.

Step 3:

Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

Backpropagation Algorithm

Step 4:

Calculate the error in the outputs

`Backpropagation Error= Actual Output – Desired Output`

Step 5:

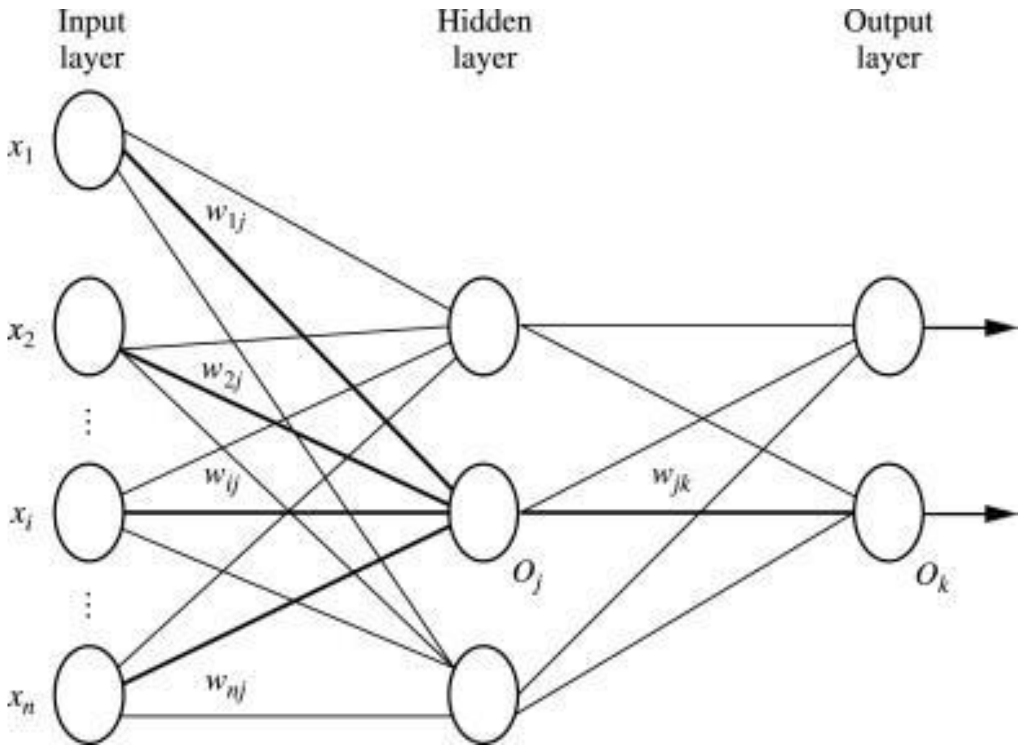
From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 6:

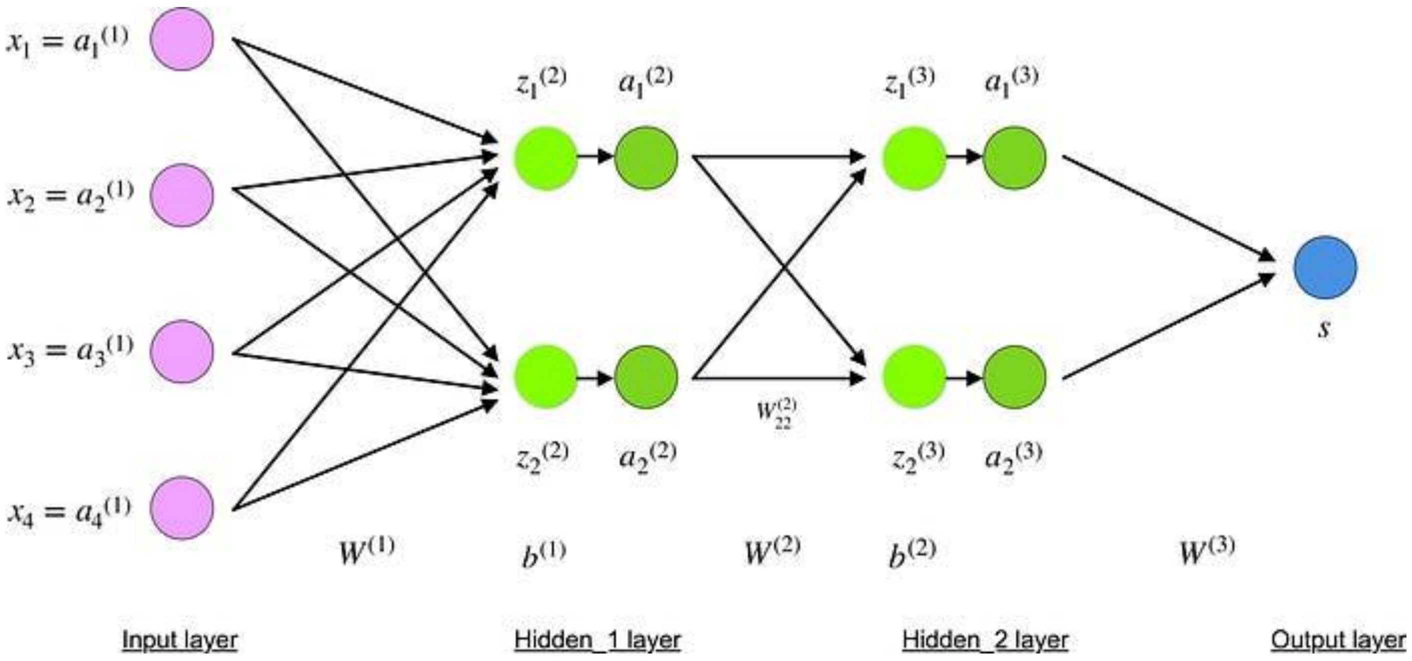
Repeat the process until the desired output is achieved.

Parameters

- x = inputs training vector $x=(x_1,x_2,\ldots\ldots\ldots\ldots x_n)$.
- t = target vector $t=(t_1,t_{2\ldots\ldots\ldots\ldots\ldots}t_{n)}$.
- δ_k = error at output unit.
- δ_j = error at hidden layer.
- α = learning rate.
- V_{0j} = bias of hidden unit j.



Backpropagation Algorithm



Training Algorithm :

- **Step 1:** Initialize weight to small random values.
- **Step 2:** While the steps stopping condition is to be false do step 3 to 10.
- **Step 3:** For each training pair do step 4 to 9 (Feed-Forward).
- **Step 4:** Each input unit receives the signal unit and transmits the signal x_i signal to all the units.

Training Algorithm

- **Step 5 :** Each hidden unit Zj (z=1 to a) sums its weighted input signal to calculate its net input

$$z_{inj} = v_{0j} + \Sigma x_i v_{ij} \quad (\text{ i=1 to n})$$

Applying activation function $z_j = f(z_{inj})$ and sends this signals to all units in the layer about i.e output units

For each output l=unit y_k = (k=1 to m) sums its weighted input signals.

$$y_{ink} = w_{0k} + \Sigma\, z_i w_{jk} \quad (\text{j=1 to a})$$

and applies its activation function to calculate the output signals.

$$y_k = f(y_{ink})$$

Backpropagation Error

- **Step 6:** Each output unit y_k (k=1 to n) receives a target pattern corresponding to an input pattern then error is calculated as:

  $$\delta_k = (t_k - y_k) + y_{ink}$$

- **Step 7:** Each hidden unit Z_j (j=1 to a) sums its input from all units in the layer above

  $$\delta_{inj} = \Sigma \delta_j w_{jk}$$

  The error information term is calculated as :

- $$\delta_j = \delta_{inj} + z_{inj}$$

Updation of weight and bias

- **Step 8:** Each output unit y_k (k=1 to m) updates its bias and weight (j=1 to a). The weight correction term is given by :

$$\Delta w_{jk} = \alpha \, \delta_k \, z_j$$

and the bias correction term is given by $\Delta w_k = \alpha \, \delta_k$.

therefore $w_{jk(new)} = w_{jk(old)} + \Delta w_{jk}$

$$w_{0k(new)} = w_{ok(old)} + \Delta w_{ok}$$

for each hidden unit z_j (j=1 to a) update its bias and weights (i=0 to n) the weight connection term

$$\Delta v_{ij} = \alpha \, \delta_j \, x_i$$

and the bias connection on term

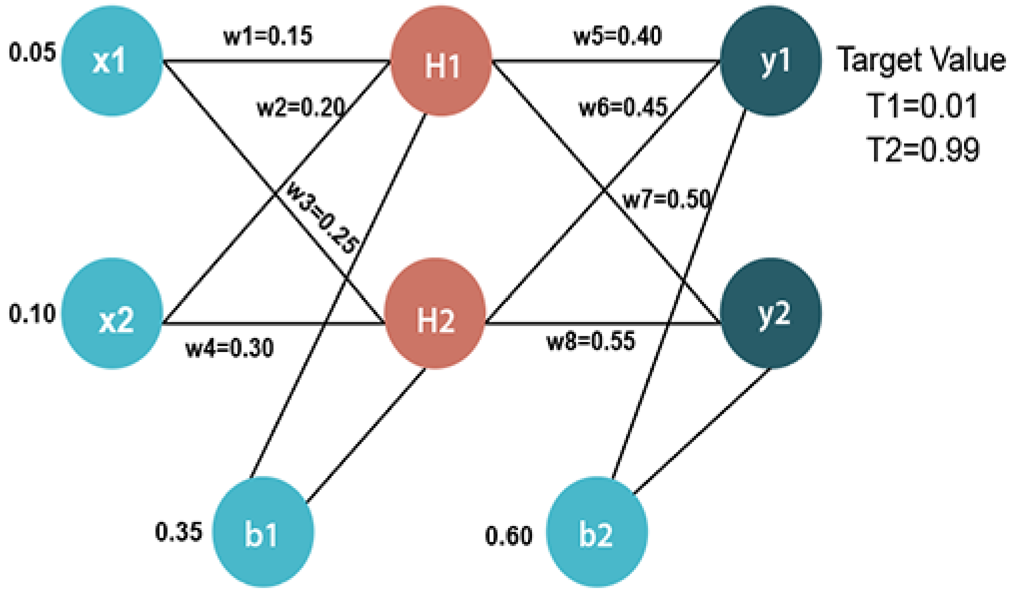
$$\Delta v_{0j} = \alpha \, \delta_j$$

Therefore $v_{ij(new)} = v_{ij(old)} + \Delta v_{ij}$

$$v_{0j(new)} = v_{0j(old)} + \Delta v_{0j}$$

- **Step 9:** Test the stopping condition. The stopping condition can be the minimization of error, number of epochs.

Backpropagation- Sample model



Solution in separate pdf

What is overfitting?

- It is a common pitfall in deep learning algorithms in which a model tries to fit the training data entirely and ends up memorizing the data patterns and the noise and random fluctuations.
- These models fail to generalize and perform well in the case of unseen data scenarios, defeating the model's purpose.
- When can overfitting occur?
- The high variance of the model performance is an indicator of an overfitting problem.
- The training time of the model or its architectural complexity may cause the model to overfit.
- If the model trains for too long on the training data or is too complex, it learns the noise or irrelevant information within the dataset.

Bias-variance

- **Bias:**
- Bias measures the difference between the model's prediction and the target value.
- If the model is oversimplified, then the predicted value would be far from the ground truth resulting in more bias.

- **Variance:**
- Variance is the measure of the inconsistency of different predictions over varied datasets.
- If the model's performance is tested on different datasets, the closer the prediction, the lesser the variance.
- Higher variance is an indication of overfitting in which the model loses the ability to generalize.

Bias-variance

- **Bias-variance tradeoff:**
- A simple linear model is expected to have a high bias and low variance due to less complexity of the model and fewer trainable parameters.
- On the other hand, complex non-linear models tend to observe an opposite behavior. In an ideal scenario, the model would have an optimal balance of bias and variance.

Overfitting happens when?

- The data used for training is not cleaned and contains garbage values. The model captures the noise in the training data and fails to generalize the model's learning.
- The model has a high variance.
- The training data size is not enough, and the model trains on the limited training data for several epochs.
- The architecture of the model has several neural layers stacked together. Deep neural networks are complex and require a significant amount of time to train, and often lead to overfitting the training set.

How to detect overfit models?

- Detecting overfitting is *technically* not possible unless we test the data.
  - *K-fold cross-validation* is one of the most popular techniques commonly used to detect overfitting.
  - We split the data points into k equally sized subsets in K-folds cross-validation, called "folds." One split subsets act as the testing set, and the remaining folds will train the model.
  - After all the iterations, we average the scores to assess the performance of the overall model.

How to detect overfit models?

Techniques to avoid overfitting

- **Train with more cleaned data**
  - assumption in this method is that the data to be fed into the model should be clean; otherwise, it would worsen the problem of overfitting
- **Data augmentation**
  - makes a sample data look slightly different every time the model processes it.
- **Feature selection**
  - model can detect many redundant features or features determinable from other features leading to unnecessary complexity
- **Simplify data**
  - by decreasing the complexity of the model to make it simple enough that it does not overfit - pruning a decision tree, reducing the number of parameters in a neural network
- **Early stopping**
  - Stopping the training of deep learning models where the number of epochs is set high
- **Dropout techniques**
  - randomly selecting nodes and removing them from training

Train, Validation, and Test Datasets

- To reiterate the findings from researching the experts above, this section provides unambiguous definitions of the three terms.
- **Training Dataset**: The sample of data used to fit the model.
- **Validation Dataset**: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.
- **Test Dataset**: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.