Backpropagation

Backpropagation: A **neural network** learning algorithm

Backpropagation is an algorithm that back propagates the errors from output nodes to the input nodes. Therefore, it is simply referred to as backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.

A neural network:

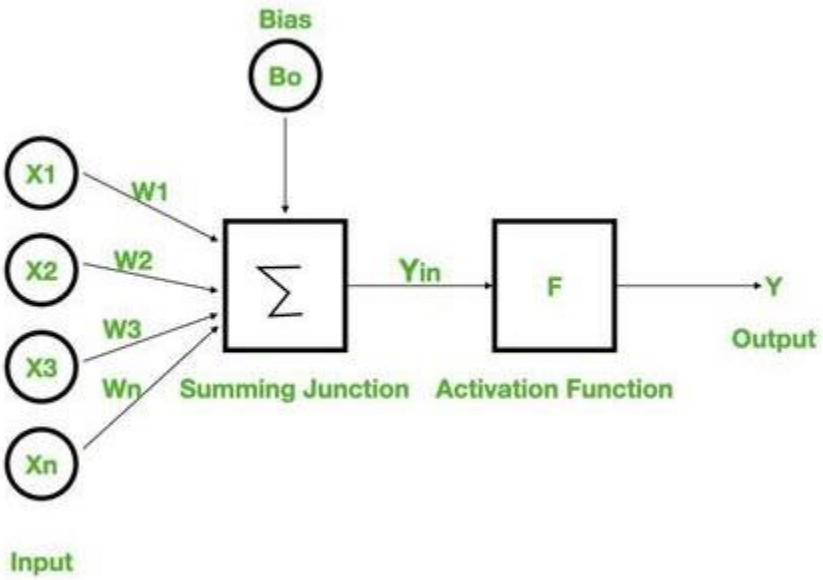
A set of connected input/output units where each connection has a **weight** associated with it

During the learning phase, the **network learns by adjusting the weights** to be able to predict the correct class label of the input tuples.

Also referred to as **connectionist learning** due to the connections between units.

Neural Network:

Neural networks are an information processing paradigm inspired by the human nervous system. Just like in the human nervous system, we have biological neurons in the same way in neural networks we have artificial neurons, artificial neurons are mathematical functions derived from biological neurons. The human brain is estimated to have about 10 billion neurons, each connected to an average of 10,000 other neurons. Each neuron receives a signal through a synapse, which controls the effect of the signal on the neuron.



Backpropagation:

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights and is very efficient, rather than naively directly computing the gradient with respect to each individual weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

Working of Backpropagation:

Neural networks use supervised learning to generate output vectors from input vectors that the network operates on. It Compares generated output to the desired output and generates an error report if the result does not match the generated output vector. Then it adjusts the weights according to the bug report to get your desired output.

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.

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.

Need for Backpropagation:

Backpropagation is “backpropagation of errors” and is very useful for training neural networks. It’s fast, easy to implement, and simple. Backpropagation does not require any parameters to be set, except the number of inputs. Backpropagation is a flexible method because no prior knowledge of the network is required.

Types of Backpropagation

There are two types of backpropagation networks.

- **Static backpropagation:** Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
- **Recurrent backpropagation:** Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached. Static backpropagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

Advantages:

- It is simple, fast, and easy to program.
- Only numbers of the input are tuned, not any other parameter.
- It is Flexible and efficient.
- No need for users to learn any special functions.

Disadvantages:

- It is sensitive to noisy data and irregularities. Noisy data can lead to inaccurate results.
- Performance is highly dependent on input data.
- Spending too much time training.
- The matrix-based approach is preferred over a mini-batch.

SVM:

https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

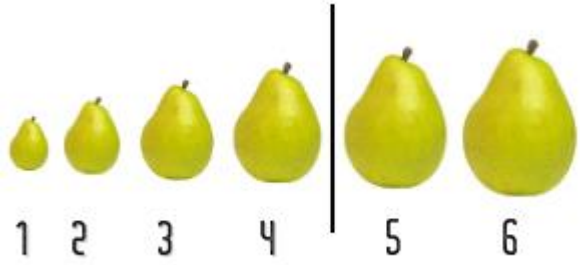
What is Active Learning?
Active Learning is a special case of Supervised Machine Learning. This approach is used to construct a high-performance classifier while keeping the size of the training dataset to a minimum by actively selecting the valuable data points.

Where should we apply active learning?

1. We have a very small amount or a huge amount of dataset.
2. Annotation of the unlabeled dataset cost human effort, time, and money.
3. We have access to limited processing power.

Example

On a certain planet, there are various fruits of different size(1-5), some of them are poisonous and others don't. The only criteria to decide a fruit is poisonous or not is its size. our task is to train a classifier that predicts the given fruit is poisonous or not. The only information we have, is a fruit with size 1 is not poisonous, the fruit of size 5 is poisonous and after a particular size, all fruits are poisonous.



The first approach is to check each and every size of the fruit, which consumes time and resources.

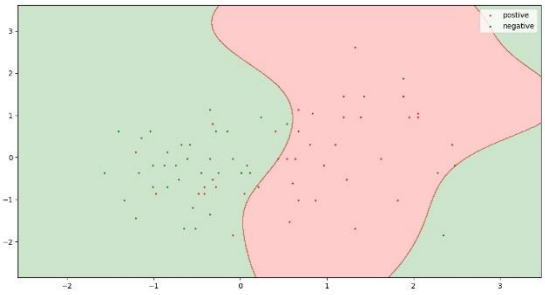
The second approach is to apply the binary search and find the transition point (decision boundary). This approach uses fewer data and gives the same results as of linear search.

General Algorithm :

```
1. train classifier with the initial training dataset
2. calculate the accuracy
3. while(accuracy < desired accuracy):
4.    select the most valuable data points (in general points close to decision
boundary)
5.    query that data point/s (ask for a label) from human oracle
6.    add that data point/s to our initial training dataset
7.    re-train the model
8.    re-calculate the accuracy
```

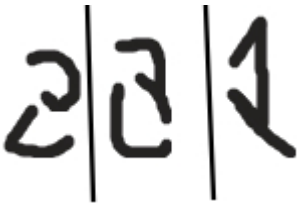
Approaches Active Learning Algorithm
1. Query Synthesis

- Generally, this approach is used when we have a very small dataset.
- This approach we choose any uncertain point from given n-dimensional space. we don't care about the existence of that point.



In this query, synthesis can pick any point(valuable) from 3*3 2-D plane.

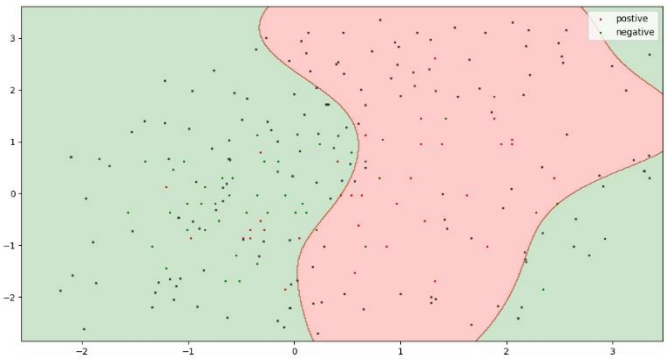
- Sometimes it would be difficult for human oracle to annotate the queried data point.



These are some queries generated by the Query Synthesis approach for a model trained for handwritten recognition. It is very difficult to annotate these queries.

2. Sampling

- This approach is used when we have a large dataset.
- In this approach, we split our dataset into three parts: Training Set; Test Set; Unlabeled Pool(ironical) [5%; 25%, 70%].
- This training dataset is our initial dataset and is used to initially train our model.
- This approach selects valuable/uncertain points from this unlabeled pool, this ensures that all the query can be recognized by human oracle



What is transfer learning?

Transfer learning for machine learning is when existing models are reused to solve a new challenge or problem. Transfer learning is not a distinct type of machine learning algorithm, instead it's a technique or method used whilst training models. The knowledge developed from previous training is recycled to help perform a new task. The new task will be related in some way to the previously trained task, which could be to categorise objects in a specific file type. The original trained model usually requires a high level of generalisation to adapt to the new unseen data.

Transfer learning means that training won't need to be restarted from scratch for every new task. Training new machine learning models can be resource-intensive, so transfer learning saves both resources and time. The accurate labelling of large datasets also takes a huge amount of time. The majority of data encountered by organisations can often be unlabelled, especially with the extensive datasets required to train a machine learning algorithm. With transfer learning, a model can be trained on an available labelled dataset, then be applied to a similar task that may involve unlabelled data.

What is transfer learning used for?

Transfer learning for machine learning is often used when the training of a system to solve a new task would take a huge amount of resources. The process takes relevant parts of an existing machine learning model and applies it to solve a new but similar problem. A key part of transfer learning is generalisation. This means that only knowledge that can be used by another model in different scenarios or conditions is transferred. Instead of models being rigidly tied to a training dataset, models used in transfer learning will be more generalised. Models developed in this way can be utilised in changing conditions and with different datasets.

An example is the use of transfer learning with the categorisation of images. A machine learning model can be trained with labelled data to identify and categorise the subject of images. The model can then be adapted and reused to identify another specific subject within a set of images through transfer learning. The general elements of the model will stay the same, saving resources. This could be the parts of the model that identifies the edge of an object in an image. The transfer of this knowledge saves retraining a new model to achieve the same result.

Transfer learning is generally used:

- To save time and resources from having to train multiple machine learning models from scratch to complete similar tasks.
- As an efficiency saving in areas of machine learning that require high amounts of resources such as image categorisation or natural language processing.
- To negate a lack of labelled training data held by an organisation, by using pre-trained models.

How does transfer learning work?

Transfer learning means taking the relevant parts of a pre-trained machine learning model and applying it to a new but similar problem. This will usually be the core information for the model to function, with new aspects added to the model to solve a specific task. Programmers will need to identify which areas of the model are relevant to the new task, and which parts will need to be retrained. For example, a new model may keep the processes that allow the machine to identify objects or data, but retrain the model to identify a different specific object.

A machine learning model which identifies a certain subject within a set of images is a prime candidate for transfer learning. The bulk of the model which deals with how to recognise different subjects can be kept.

The part of the algorithm which highlights a specific subject to categorise is the element that will be retrained. In this case, there's no need to rebuild and retrain a machine learning algorithm from scratch.

In supervised machine learning, models are trained to complete specific tasks from labelled data during the development process. Input and desired output are clearly mapped and fed to the algorithm. The model can then apply the learned trends and pattern recognition to new data. Models developed in this way will be highly accurate when solving tasks in the same environment as its training data. It will become much less accurate if the conditions or environment changes in real-world application beyond the training data. The need for a new model based on new training data may be required, even if the tasks are similar.

Transfer learning is a technique to help solve this problem. As a concept, it works by transferring as much knowledge as possible from an existing model to a new model designed for a similar task. For example, transferring the more general aspects of a model which make up the main processes for completing a task. This could be the process behind how objects or images are being identified or categorized. Extra layers of more specific knowledge can then be added to the new model, allowing it to perform its task in new environments.

Benefits of transfer learning for machine learning

Transfer learning brings a range of benefits to the development process of machine learning models. The main benefits of transfer learning include the saving of resources and improved efficiency when training new models. It can also help with training models when only unlabelled datasets are available, as the bulk of the model will be pre-trained.

The main benefits of transfer learning for machine learning include:

- Removing the need for a large set of labelled training data for every new model.
- Improving the efficiency of machine learning development and deployment for multiple models.
- A more generalised approach to machine problem solving, leveraging different algorithms to solve new challenges.
- Models can be trained within simulations instead of real-world environments.

Saving on training data

A huge range of data is usually required to accurately train a machine learning algorithm. Labelled training data takes time, effort and expertise to create. Transfer learning cuts down on the training data required for new machine learning models, as most of the model is already previously trained.

In many cases, large sets of labelled data are unavailable to organisations. Transfer learning means models can be trained on available labelled datasets, then applied to similar data that's unlabelled.

Efficiently train multiple models

Machine learning models designed to complete complex tasks can take a long time to properly train. Transfer learning means organisations don't have to start from scratch each time a similar model is required. The resources and time put into training a machine learning algorithm can be shared across different models. The whole training process is made more efficient by reusing elements of an algorithm and transferring the knowledge already held by a model.

Leverage knowledge to solve new challenges

Supervised machine learning is one of the most popular types of machine learning today. The approach creates highly accurate algorithms that are trained to complete specific tasks using labeled training data. However once deployed, performance may suffer if the data or environment stray from the training data. Transfer learning means knowledge can be leveraged from existing models instead of starting from scratch each time.

Transfer learning helps developers take a blended approach from different models to fine-tune a solution to a specific problem. The sharing of knowledge between two different models can result in a much more accurate and powerful model. The approach allows for the building models in an iterative way.

Simulated training to prepare for real-world tasks

Transfer learning is a key element of any machine learning process which includes simulated training. For models that need to be trained in real-world environments and scenarios, digital simulations are a less expensive or time-consuming option. Simulations can be created to mirror real-life environments and actions. Models can be trained to interact with objects in a simulated environment.

Simulated environments are increasingly used for reinforcement machine learning models. In this case, models are being trained to perform tasks in different scenarios, interacting with objects and environments. For example, simulation is already a key step in the development of self-driving systems for cars. Initial training of a model in a real-world environment could prove dangerous and time-consuming. The more generalised parts of the model can be built using simulations before being transferred to a model for real-world training.

Bayesian Belief Network

https://www.javatpoint.com/bayesian-belief-network-in-artificial-intelligence

https://www.geeksforgeeks.org/basic-understanding-of-bayesian-belief-networks/

https://pandio.com/the-bayesian-belief-network-in-machine-learning/

Frequent Pattern

https://www.geeksforgeeks.org/ml-frequent-pattern-growth-algorithm/