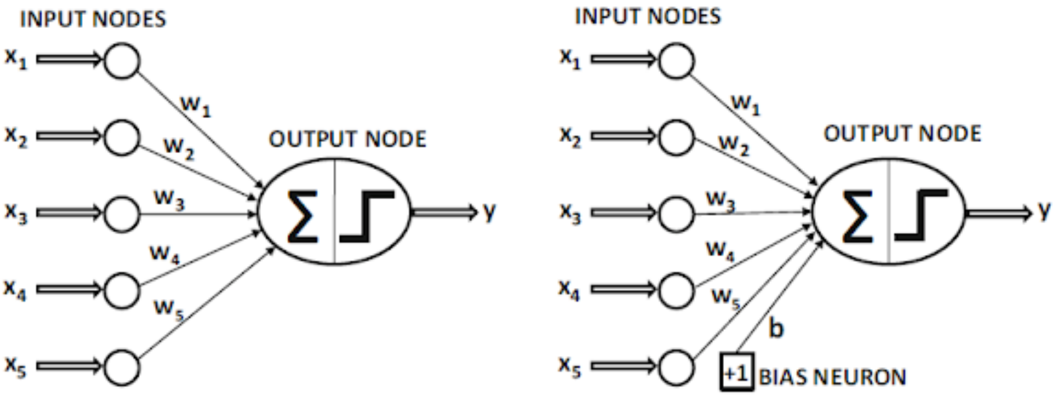
NEURAL NETWORKS AND DEEP LEARNING CST 395 CS 5TH SEMESTER HONORS COURSE- Dr Binu V P, 9847390760

CS 5th Semester Honors course for the Computer Science at KTU- Dr Binu V P

The Basic Architecture of Neural Networks-Single Layer Neural Network

September 07, 2022

In the singlelayer network, a set of inputs is directly mapped to an output by using a generalized variation of a linear function. This simple instantiation of a neural network is also referred to as the **perceptron**. In multi-layer neural networks, the neurons are arranged in layered fashion, in which the input and output layers are separated by a group of hidden layers. This layer-wise architecture of the neural network is also referred to as a **feed-forward network**.



(a) Perceptron without bias

(b) Perceptron with bias

Figure : The basic architecture of the perceptron

The Perceptron

The simplest neural network is referred to as the perceptron. This neural network contains a single input layer and an output node. The basic architecture of the perceptron is shown in Figure (a).

Consider a situation where each training instance is of the form (X, y), where each $X = [x_1, \ldots x_d]$ contains d feature variables, and $y \in \{-1, +1\}$ contains the

observed value of the binary class variable. By "observed value" we refer to the fact that it is given to us as a part of the training data, and our goal is to predict the class variable for cases in which it is not observed. For example, in a credit-card fraud detection application, the features might represent various properties of a set of credit card transactions (e.g.,amount and frequency of transactions), and the class variable might represent whether or not this set of transactions is fraudulent. Clearly, in this type of application, one would have
historical cases in which the class variable is observed, and other (current) cases in which the class variable has not yet been observed but needs to be predicted.

The input layer contains d nodes that transmit the d features $X = [x_1, \ldots, x_d]$ with edges of weight $W = [w_1 \ldots w_d]$ to an output node. The input layer does not perform any computation in its own right. The linear function $W . X = \sum_{i=1}^{d} w_i x_i$ is computed at the output node. Subsequently, the sign of this real value is used in order to predict the dependent variable of X. Therefore, the prediction $\hat{y}$ is computed as follows:

$$\hat{y} = sign\{\overline{W} . \overline{X}\} = sign\{\sum_{j=1}^{d} w_j x_j\}$$

The sign function maps a real value to either +1 or −1, which is appropriate for binary classification.The error of the prediction is therefore $E(X) = y - \hat{y}$, which is one of the values drawn from the set $\{-2, 0, +2\}$.In cases where the error value $E(X)$ is nonzero, the weights in the neural network need to be updated in the (negative) direction of the error gradient. As we will see later, this process is similar to that used in various types of linear models in machine learning. In spite of the similarity of the perceptron with respect to traditional machine learning models, its interpretation as a computational
unit is very useful because it allows us to put together multiple units in order to create far more powerful models than are available in traditional machine learning.

In the architecture of the perceptron , in which a single input layer transmits the features to the output node.The edges from the input to the output contain he weights $w_1 \ldots w_d$ with which the features are multiplied and added at the output node Subsequently, the sign function is applied in order to convert the aggregated value into a class label. The sign function serves the role of an **activation function**. Different choices of activation functions can be used to simulate different types of models used in machine learning, like **least-squares regression** with numeric targets, the **support vector machine, or a logistic regression classifier.** Most of the basic machine learning models can be easily represented as simple neural network architectures. It is a useful exercise to model traditional machine learning techniques as neural architectures, because it provides a clearer picture of how deep learning generalizes traditional machine learning.

It is noteworthy that the perceptron contains two layers, although the input layer does not perform any computation and only transmits the feature values. The input layer is not included in the count of the number of layers in a neural network. Since the perceptron contains a single computational layer, it is considered a single-layer network.

In many settings, there is an invariant part of the prediction, which is referred to as the bias b. For example, consider a setting in which the feature variables are mean centered, but the mean of the binary class prediction from $\{-1, +1\}$ is not 0. This will tend to occur in situations in which the binary class distribution is highly imbalanced. In such a case, the aforementioned approach is not sufficient for prediction. We need to incorporate an additional bias variable b that captures this invariant part of the prediction:

$$\hat{y} = sign\{\overline{W}.\overline{X} + b\} = sign\{\sum_{j=1}^{d} w_j x_j + b\}$$

The bias can be incorporated as the weight of an edge by using a bias neuron. This is achieved by adding a neuron that always transmits a value of 1 to the output node. The weight of the edge connecting the bias neuron to the output node provides the bias variable.An example of a bias neuron is shown in Figure (b) above .
Another approach that works well with single-layer architectures is to use a feature engineering trick in which an additional feature is created with a constant value of 1. The coefficient of this feature provides the bias,and one can then work with first Equation . So biases will not be explicitly used (for simplicity in architectural representations) because they can be incorporated with bias neurons. The details of the training algorithms remain the same by simply treating the bias neurons like any other neuron with a fixed activation value of 1. Therefore, the following will work with the predictive assumption of first Equation , which does not explicitly uses biases.

At the time that the perceptron algorithm was proposed by Rosenblatt , these optimizations were performed in a heuristic way with actual hardware circuits, and it was not presented in terms of a formal notion of optimization in machine learning (as is common today). However, the goal was always to minimize the error in prediction

The perceptron algorithm was, therefore,heuristically designed to minimize the number of misclassifications.Therefore, the goal of the perceptron algorithm in least-squares form with respect to all training instances in a data set D containing feature label pairs can be written as

$$Minimize_{\bar{W}} L = \sum_{(\overline{X},y)\in D}(y-\hat{y})^2 = \sum_{(\overline{X},y)\in D}(y - sign\{\overline{W}.\overline{X}\})^2$$

This type of minimization objective function is also referred to as a **loss function**. As we will see later, almost all neural network learning algorithms are formulated with the use of a loss function.This loss function looks a lot like leastsquares regression. However, the latter is defined for continuous-valued target variables, and the corresponding loss is a smooth and continuous function of the variables. On the

other hand, for the least-squares form of the objective function, the sign function is nondifferentiable, with step-like jumps at specific points. Furthermore, the sign function takes on constant values over large portions of the domain, and therefore the exact gradient takes on zero values at differentiable points. This results in a staircase-like loss surface, which is not suitable for gradient-descent. The perceptron algorithm (implicitly) uses a smooth approximation of the gradient of this objective function with respect to each example:

$$\bigtriangledown L_{smooth} = \sum_{(\bar{X},y)\in D}(y - \hat{y}).\, X$$

Perceptron algorithm optimizes some unknown smooth function with the use of gradient descent.Although the above objective function is defined over the entire training data, the training algorithm of neural networks works by feeding each input data instance $\overline{X}$ into the network one by one (or in small batches) to create the prediction $\hat{y}$. The weights are then updated, based on the error value $E(X) = (y - \hat{y})$. Specifically, when the data point $\overline{X}$ is fed into the network, the weight vector $\overline{W}$ is updated as follows:

$$\overline{W} \Leftarrow \overline{W} + \alpha(y - \overline{y})\overline{X}$$

The parameter αregulates the learning rate of the neural network. The perceptron algorithm repeatedly cycles through all the training examples in random order and iteratively adjuststhe weights until convergence is reached. A single training data point may be cycled through many times. Each such cycle is referred to as an **epoch**. One can also write the gradientdescent update in terms of the error $E(\overline{X}) = (y - \hat{y})$ as follows:

$$\overline{W} \Leftarrow \overline{W} + \alpha E(\overline{X})\overline{X}$$

The basic perceptron algorithm can be considered a **stochastic gradient-descent** method, which implicitly minimizes the squared error of prediction by performing gradient-descent updates with respect to randomly chosen training points. The assumption is that the neural network cycles through the points in random order during training and changes the weights with the goal of reducing the prediction error on that point. It is easy to see from Equation above that non-zero updates are made to the weights only when $y \neq \hat{y}$, which occurs only.when errors are made in prediction. In **mini-batch stochastic gradient descent,** the aforementioned updates of Equation above are implemented over a randomly chosen subset of training points S:

$$\overline{W} \Leftarrow \overline{W} + \alpha \sum_{\overline{X} \in S} E(\overline{X})\overline{X}$$

An interesting quirk of the perceptron is that it is possible to set the learning rate α to 1, because the learning rate only scales the weights.The type of model proposed in the perceptron is a linear model, in which the equation $\overline{W} \cdot \overline{X} = 0$ defines a linear hyperplane. Here, $\overline{W} = (w_1 \ldots w_d)$ is a d-dimensional vector that is normal to the hyperplane. Furthermore, the value of $\overline{W} \cdot \overline{X}$ is positive for values of $\overline{X}$ on one side of the hyperplane, and it is negative for values of $\overline{X}$ on the other side. This type of model performs particularly well when the data is linearly separable. Examples of linearly separable and inseparable data are shown in Figure.

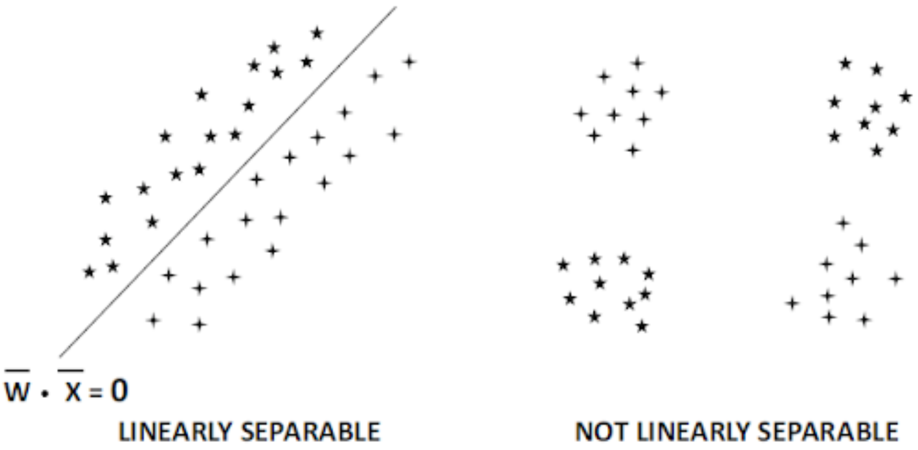


Figure : Examples of linearly separable and inseparable data in two classes

The perceptron algorithm is good at classifying data sets like the one shown on the left-hand side of Figure above, when the data is linearly separable. On the other hand, it tends to perform poorly on data sets like the one shown on the right-hand side of Figure above. This example shows the inherent modeling limitation of a perceptron, which necessitates the use of more complex neural architectures.

Since the original perceptron algorithm was proposed as a heuristic minimization of classification errors, it was particularly important to show that the algorithm converges to reasonable solutions in some special cases. In this context, it was shown that the perceptron algorithm always converges to provide zero error on the training data when the data are linearly separable. However, the perceptron algorithm is not guaranteed to converge in instances where the data are not linearly separable. The perceptron might sometimes arrive at a very poor solution with data that are not linearly separable (in comparison with many other learning algorithms).

What Objective Function Is the Perceptron Optimizing?

The original perceptron paper by Rosenblatt did not formally propose a loss function. In those years, these implementations were achieved using actual hardware circuits. The original Mark I perceptron was intended to be a machine rather than an algorithm, and custom-built hardware was used to create it The general goal was to minimize the number of classification errors with a heuristic update process (in hardware) that changed weights in the “correct” direction whenever errors were made. This heuristic update strongly resembled gradient descent but it was not derived as a gradient-descent method. Gradient descent is defined only for smooth loss functions in algorithmic settings, whereas the hardware-centric approach was designed in a more heuristic way with binary outputs.

Unfortunately, binary signals are not prone to continuous optimization.Can we find a smooth loss function, whose gradient turns out to be the perceptron update? The number of classification errors in a binary classification problem can be written in the form of a 0/1 loss function for training data point $(\overline{X}_i, y_i)$ as follows:

$$L_i^{(0/1)} = \frac{1}{2}(y_i - sign\{\overline{W}\,.\,\overline{X_i}\}^2) = 1 - y_i.\,sign\{\overline{W}\,.\,\overline{X_i}\}$$

The simplification to the right-hand side of the above objective function is obtained by setting both y_i^2 and $sign\{\overline{W}\,.\,\overline{X_i}\}^2\}$ to 1, since they are obtained by squaring a value drawn from $\{-1, +1\}$. However, this objective function is not differentiable, because it has a staircase like shape, especially when it is added over multiple points. Note that the 0/1 loss above is dominated by the term $-y_i sign\{\overline{W}\cdot\overline{X_i}\}$, in which the sign function causes most of the problems associated with non-differentiability. Since neural networks are defined by gradient-based optimization, we need to define a smooth objective function that is responsible for the perceptron updates. It can be shown that the updates of the perceptron implicitly optimize the perceptron criterion. This objective function is defined by dropping the sign function in the above 0/1 loss and setting negative values to 0 in order to treat all correct predictions in a uniform and lossless way:

$$L_i = max\{-y_i(\overline{W}\,.\,\overline{X_i}), 0\}$$

The gradient of this smoothed objective function leads to the perceptron update, and the update of the perceptron is essentially

$$\overline{W} \Leftarrow \overline{W} - \alpha \bigtriangledown_w L_i$$

The modified loss function to enable gradient computation of a non differentiable function is also referred to as a **smoothed surrogate loss function**. Almost all continuous optimization-based learning methods (such as neural networks) with discrete outputs (such as class labels) use some type of smoothed surrogate loss function.

To leave a comment, click the button below to sign in with Google.

SIGN IN WITH GOOGLE

Popular posts from this blog

NEURAL NETWORKS AND DEEP LEARNING CST 395 CS 5TH SEMESTER HONORS COURSE NOTES - Dr Binu V P, 9847390760

October 03, 2022

About Me Syllabus Question Paper Dec 2022 Module 1 (Basics of Machine Learning) Overview of Machine Learning Machine Learning Algorithm Linear Regression Capacity, Overfitting and Underfitting Regularization Hyperparameters and Validatioı …

READ MORE

Machine Learning Algorithm

October 01, 2022

A machine learning algorithm is an algorithm that is able to learn from data. But what do we mean by learning? Mitchell (1997) provides the definition "A computer program is said to learn from experience E with respect to some class of tasks T and …

READ MORE

Syllabus CST 395 Neural Network and Deep Learning

October 01, 2022

Syllabus Module - 1 (Basics of Machine Learning) Machine Learning basics - Learning algorithms - Supervised, Unsupervised, Reinforcement, overfitting, Underfitting, Hyper parameters and Validation sets, Estimators -Bias and Variance. Challenges …

READ MORE

Powered by Blogger

Theme images by Michael Elkan

DR.BINU V P-9847390760

Associate Professor and Head Dept of Computer Science-IHRD Karunagappally.Love to share the thoughts and views .I play lot of Games and basically an Athlet in my school and college days.I never keep my studies as a second option.Stood first In MTech and BTech.I did my resea …

VISIT PROFILE

Archive

Report Abuse