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

Choice of Activation and Loss Functions

September 02, 2022

The choice of activation function is a critical part of neural network design. In the case of the perceptron, the choice of the sign activation function is motivated by the fact that a binary class label needs to be predicted. However, it is possible to have other types of situations where different target variables may be predicted. For example, if the target variable to be predicted is real, then it makes sense to use the identity activation function, and the resulting algorithm is the same as least-squares regression. If it is desirable to predict a probability of a binary class, it makes sense to use a sigmoid function for activating the output node, so that the prediction $\hat{y}$ indicates the probability that the observed value, y, of the dependent variable is 1. The negative logarithm of $|y/2 - 0.5 + \hat{y}|$ is used as the loss, assuming that y is coded from $\{-1, 1\}$. If $\hat{y}$ is the probability that y is 1, then$|y/2 - 0.5 + \hat{y}|$ is the probability that the correct value is predicted. This assertion is easy to verify by examining the two cases where y is 0 or 1. This loss function can be shown to be representative of the negative log-likelihood of the training data.

The importance of nonlinear activation functions becomes significant when one moves from the single-layered perceptron to the multi-layered architectures discussed later .Different types of nonlinear functions such as the sign, sigmoid, or hyperbolic tangents may be used in various layers.We use the notation Φ to denote the activation function:

$$\hat{y} = \Phi(\overline{W} . \overline{X})$$

Therefore, a neuron really computes two functions within the node, which is why we have incorporated the summation symbol $\sum$ as well as the activation symbol Φ within a neuron.The break-up of the neuron computations into two separate values is shown in Figure below

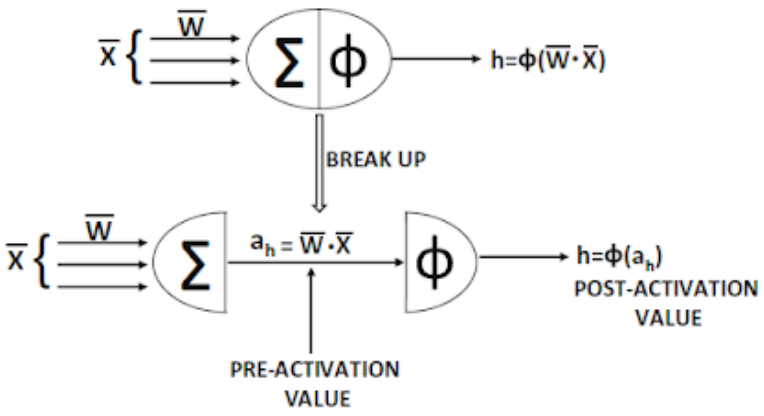


Figure Pre-activation and post-activation values within a neuron

The value computed before applying the activation function $\Phi(\cdot)$ will be referred to as the pre-activation value, whereas the value computed after applying the activation function is referred to as the post-activation value. The output of a neuron is always the post-activation value, although the pre-activation variables are often used in different types of analyses, such as the computations of the backpropagation algorithm discussed later in this chapter. The pre-activation and post-activation values of a neuron are shown in Figure.The most basic activation function $\Phi(.)$ is the identity or linear activation, which provides no nonlinearity:

$$\Phi(v) = v$$

The linear activation function is often used in the output node, when the target is a real value. It is even used for discrete outputs when a smoothed surrogate loss function needs to be set up.The classical activation functions that were used early in the development of neural networks were the **sign, sigmoid, and the hyperbolic tangent functions:**

$$\Phi(v) = sign(v) (signfunction)$$

$$\Phi(v) = \frac{1}{1+e^{-v}} (sigmoidfunction)$$

$$\Phi(v) = \frac{e^{2v}-1}{e^{2v}+1} (tanhfunction)$$

While the sign activation can be used to map to binary outputs at prediction time, its non-differentiability prevents its use for creating the loss function at training time. For example, while the perceptron uses the sign function for prediction, the perceptron criterion in training only requires linear activation. The sigmoid activation outputs a value in (0, 1), which is helpful in performing computations that should be interpreted as

probabilities.

Furthermore, it is also helpful in creating probabilistic outputs and constructing loss functions derived from maximum-likelihood models. The tanh function has a shape similar to that of the sigmoid function, except that it is horizontally re-scaled and vertically translated/re-scaled to [−1, 1]. The tanh and sigmoid functions are related as follows

$$tanh(v) = 2 \cdot sigmoid(2v) - 1$$

The tanh function is preferable to the sigmoid when the outputs of the computations are desired to be both positive and negative. Furthermore, its mean-centering and larger gradient (because of stretching) with respect to sigmoid makes it easier to train. The sigmoid and the tanh functions have been the historical tools of choice for incorporating nonlinearity in the neural network. In recent years, however, a number of piecewise linear activation functions have become more popular:

$$\Phi(v) = max\{v, 0\}(RectifiedLinearUnit[ReLU])$$
$$\Phi(v) = max\{min[v, 1], -1\}(hardtanh)$$

Pictorial representations of all the aforementioned activation functions are illustrated in Figure below. It is noteworthy that all activation functions shown here are monotonic.Furthermore, other than the identity activation function, most of the other activation functions saturate at large absolute values of the argument at which increasing further does not change the activation much.

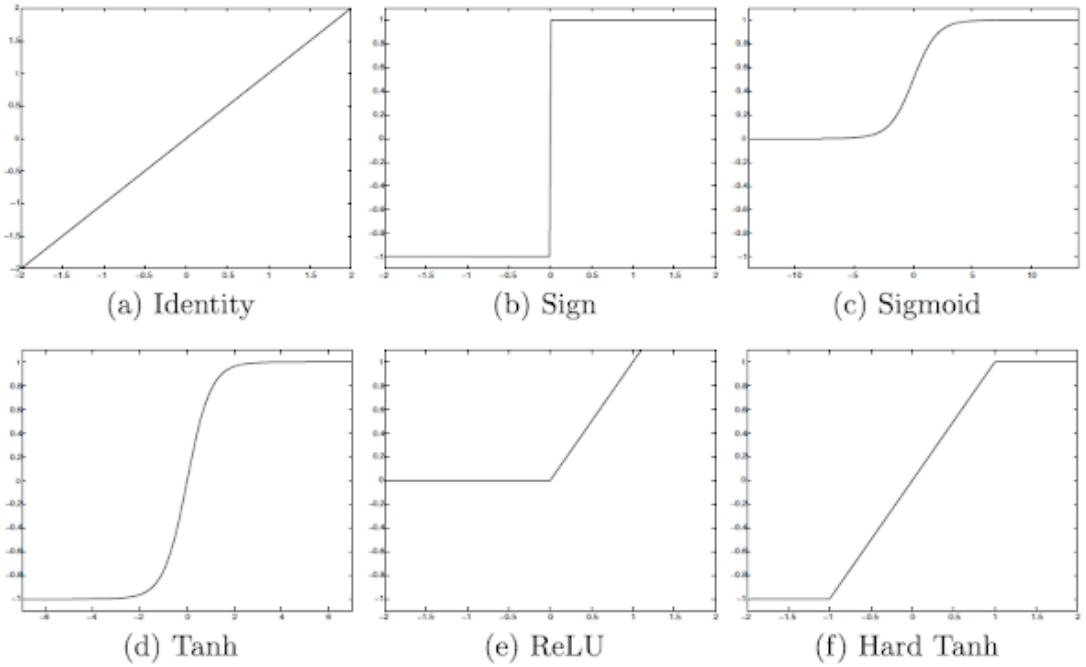


Figure : Various activation functions

As we will see later, such nonlinear activation functions are also very useful in multilayer networks, because they help in creating more powerful compositions of different types offunctions. Many of these functions are referred to as squashing functions, as they map the outputs from an arbitrary range to bounded outputs. The use of a nonlinear activation plays a fundamental role in increasing the modeling power of a network. If a network used only linear activations, it would not provide better modeling power than a single-layer linear network.

Choice and Number of Output Nodes

The choice and number of output nodes is also tied to the activation function, which in turn depends on the application at hand. For example, if $k-$way classification is intended, k output values can be used, with a softmax activation function with respect to outputs $\bar{v} = [v_1, \ldots, v_k]$ at the nodes in a given layer. Specifically, the activation function for the ith output is defined as follows:

It is helpful to think of these k values as the values output by k nodes, in which the inputs are $v_1 \ldots v_k$. An example of the softmax function with three outputs is illustrated in Figure , and the values v_1, v_2, and v_3 are also shown in the same figure. Note that the three outputs correspond to the probabilities of the three classes, and they convert the three outputs of the final hidden layer into probabilities with the softmax function. The final hidden layer often uses linear (identity) activations, when it is input into the softmax layer.Furthermore, there are no weights associated with the softmax layer, since it is only converting real-valued outputs into probabilities.

The use of softmax with a single hidden layer of linear activations exactly implements a model, which is referred to as **multinomial logistic regression** . Similarly, many variations like multi-class SVMs can be easily implemented with neural networks. Another example of a case in which multiple output nodes are used is the **autoencoder**, in which each input data point is fully reconstructed by the output layer.The autoencoder can be used to implement matrix factorization methods like singular value decomposition.The simplest neural networks that simulate basic machine learning algorithms are instructive because they lie on the continuum between traditional machine learning and deep networks. By exploring these architectures, one gets a better idea of the relationship between traditional machine learning and neural networks, and also the advantages provided by the latter.

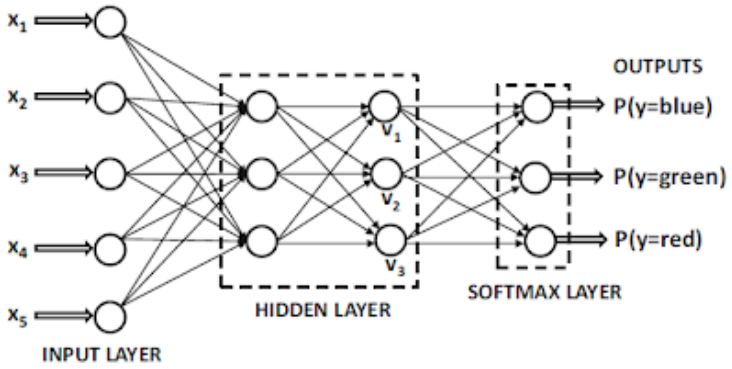


Figure . An example of multiple outputs for categorical classification with the use of a softmax layer

Choice of Loss Function

The choice of the loss function is critical in defining the outputs in a way that is sensitive to the application at hand. For example, least-squares regression with numeric outputs requires a simple squared loss of the form $(y - \hat{y})^2$ for a single training instance with target y and prediction $\hat{y}$. One can also use other types of loss like hinge loss for$y \in \{-1, +1\}$ and real-valued prediction $\hat{y}$ (with identity activation):

$$L = max\{0, 1 - y.\hat{y}\}$$

The hinge loss can be used to implement a learning method, which is referred to as a support vector machine.For multiway predictions (like predicting word identifiers or one of multiple classes), the softmax output is particularly useful. However, a softmax output is probabilistic, and therefore it requires a different type of loss function. In fact, for probabilistic predictions, two different types of loss functions are used, depending on whether the prediction is binary or whether it is multiway:

1. Binary targets (logistic regression): In this case, it is assumed that the observed value y is drawn from $\{-1, +1\}$, and the prediction $\hat{y}$ is a an arbitrary numerical value on using the identity activation function. In such a case, the loss function for a single instance with observed value y and real-valued prediction $\hat{y}$ (with identity activation) is defined as follows:

$$L = log(1 + exp(-y \cdot \hat{y}))$$

This type of loss function implements a fundamental machine learning method, referred to as logistic regression. Alternatively, one can use a sigmoid activation function to output $\hat{y} \in (0, 1)$, which indicates the probability that the observed value y is 1.Then, the negative logarithm of $|y/2 - 0.5 + \hat{y}|$ provides the loss, assuming that y is coded from $\{-1, 1\}$. This is because $|y/2 - 0.5 + \hat{y}|$ indicates the probability that the prediction is correct. This observation illustrates that one can use various combinations of activation and loss functions to achieve the same result.

2. Categorical targets: In this case, if $\hat{y_1} \ldots \hat{y}_k$ are the probabilities of the k classes (using the softmax activation), and the rth class is the ground-truth class, then the loss function for a single instance is defined as follows:

$L = -log(\hat{y}_r)$

This type of loss function implements multinomial logistic regression, and it is referred to as the cross-entropy loss. Note that binary logistic regression is identical to multinomial logistic regression, when the value of k is set to 2 in the latter.

The key point to remember is that the nature of the output nodes, the activation function, and the loss function depend on the application at hand. Furthermore, these choices also depend on one another. Even though the perceptron is often presented as the quintessential representative of single-layer networks, it is only a single representative out of a very large universe of possibilities. In practice, one rarely uses the perceptron criterion as the loss function. For discrete-valued outputs, it is common to use softmax activation with crossentropy loss. For real-valued outputs, it is common to use linear activation with squared loss. Generally, cross-entropy loss is easier to optimize than squared loss.

Some Useful Derivatives of Activation Functions

Most neural network learning is primarily related to gradient-descent with activation functions.For this reason, the derivatives of these activation functions are important.This section provides details on the derivatives of these loss functions. Later we will extensively refer to these results.

1. Linear and sign activations: The derivative of the linear activation function is 1 at all places. The derivative of $sign(v)$ is 0 at all values of v other than at $v = 0$,where it is discontinuous and non-differentiable. Because of the zero gradient and non-differentiability of this activation function, it is rarely used in the loss function even when it is used for prediction at testing time. The derivatives of the linear and sign activations are illustrated in Figure (a) and (b), respectively.

2. Sigmoid activation: The derivative of sigmoid activation is particularly simple, when it is expressed in terms of the output of the sigmoid, rather than the input. Let o be the output of the sigmoid function with argument v **:**

$o = \frac{1}{1+exp(-v)}$

Then, one can write the derivative of the activation as follows:

$\frac{\partial o}{\partial v} = \frac{exp(-v)}{(1-exp(v))^2}$

The key point is that this sigmoid can be written more conveniently in terms of the outputs:

$$\frac{\partial o}{\partial v} = o(1 - o)$$

The derivative of the sigmoid is often used as a function of the output rather than the input. The derivative of the sigmoid activation function is illustrated in Figure (c).

3. Tanh activation: As in the case of the sigmoid activation, the tanh activation is often used as a function of the output o rather than the input v:

$$o = \frac{exp(2v)-1}{exp(2v)+1}$$

One can then compute the gradient as follows:

$$\frac{\partial o}{\partial v} = \frac{4.exp(2v)}{(exp(2v)+1)^2}$$

One can also write this derivative in terms of the output o:

$$\frac{\partial o}{\partial v} = 1 - o^2$$

The derivative of the tanh activation is illustrated in Figure (d).

4. ReLU and hard tanh activations: The ReLU takes on a partial derivative value of 1 for non-negative values of its argument, and 0, otherwise. The hard tanh function takes on a partial derivative value of 1 for values of the argument in [−1, +1] and 0,otherwise. The derivatives of the ReLU and hard tanh activations are illustrated in Figure (e) and (f), respectively.

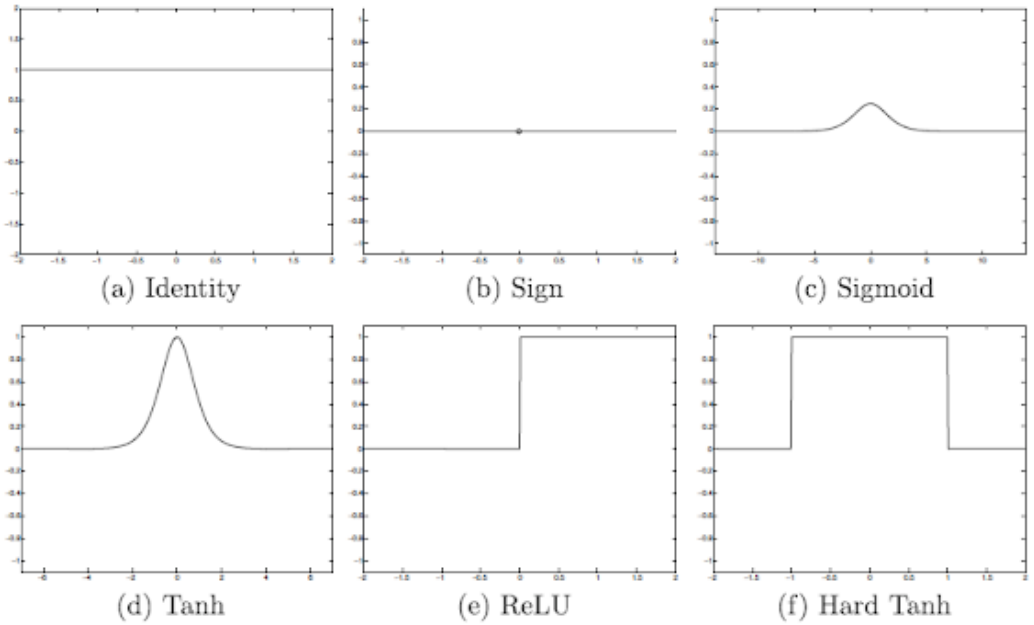
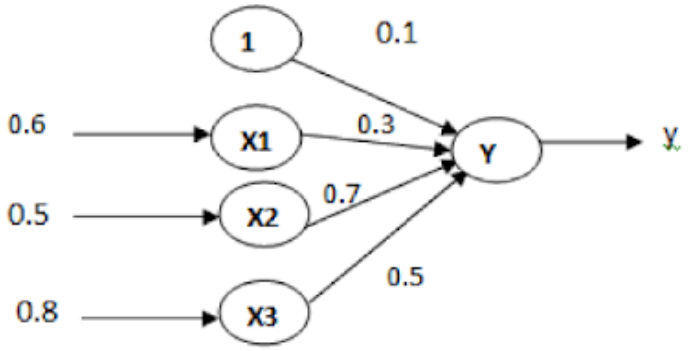


Figure : The derivatives of various activation functions

Example:

Calculate the output of the following neuron Y with the activation function as a) binary sigmoid b) tanh c)ReLU



a) Sigmoid

$Y = \frac{1}{1+e^{-(w^T x+b)}}$

$w^T x + b = 1.03$

$Y = 0.7369158958334202$

b) Tanh

$Y = \frac{exp(2(w^T x+b)-1}{exp(2(w^T x+b)+1}$

$Y = 0.7739083398558421$

c)ReLU

$Y = max\{0, w^T x + b\}$

$Y = 1.03$

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 Validatio ...

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

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