What are common cost functions used in evaluating the performance of neural networks?

Details

(feel free to skip the rest of this question, my intent here is simply to provide clarification on notation that answers may use to help them be more understandable to the general reader)

I think it would be useful to have a list of common cost functions, alongside a few ways that they have been used in practice. So if others are interested in this I think a community wiki is probably the best approach, or we can take it down if it's off topic.

Notation

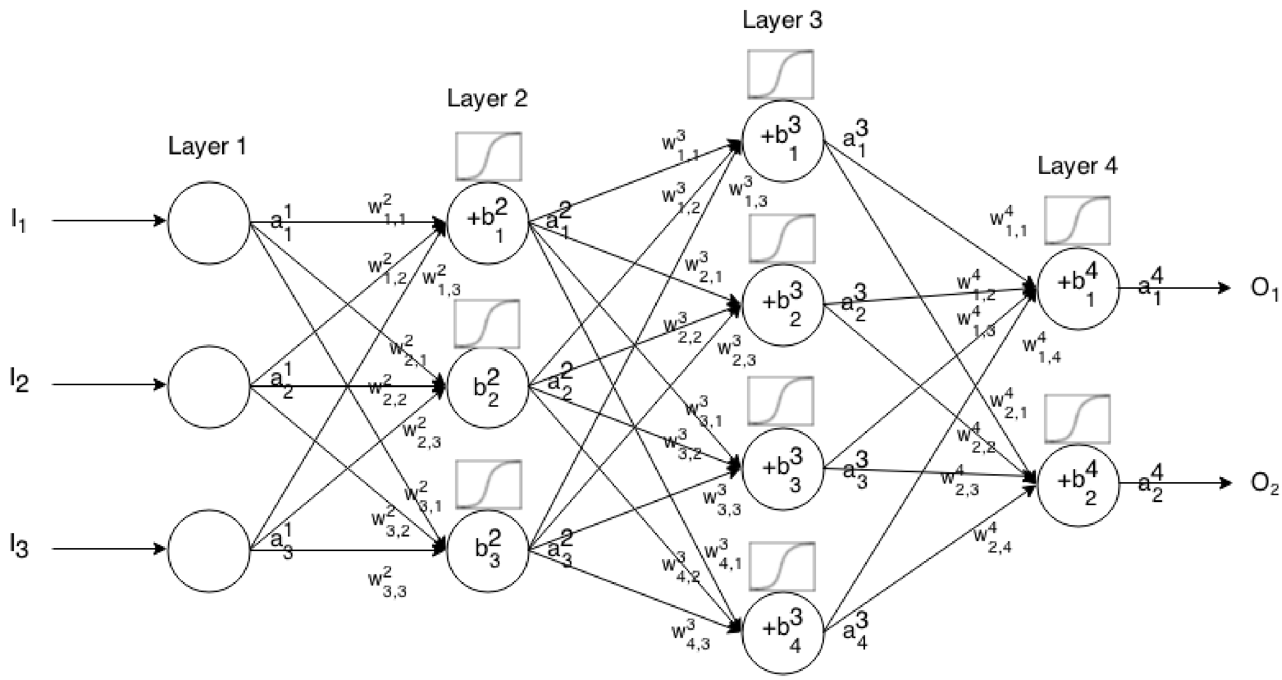
So to start, I'd like to define a notation that we all use when describing these, so the answers fit well with each other.

This notation is from Neilsen's book.
A Feedforward Neural Network is a many layers of neurons connected together. Then it takes in an input, that input "trickles" through the network and then the neural network returns an output vector.

More formally, call a^i_j the activation (aka output) of the j^{th} neuron in the i^{th} layer, where a^1_j is the j^{th} element in the input vector.
Then we can relate the next layer's input to it's previous via the following relation:

$$a^i_j = \sigma\left(\sum_k (w^i_{jk} \cdot a^{i-1}_k) + b^i_j\right)$$
where

σ is the activation function,
w^i_{jk} is the weight from the k^{th} neuron in the $(i-1)^{th}$ layer to the j^{th} neuron in the i^{th} layer,
b^i_j is the bias of the j^{th} neuron in the i^{th} layer, and
a^i_j represents the activation value of the j^{th} neuron in the i^{th} layer.
Sometimes we write z^i_j to represent $\sum_k (w^i_{jk} \cdot a^{i-1}_k) + b^i_j$, in other words, the activation value of a neuron before applying the activation function.



For more concise notation we can write

$a_i=\sigma(w_i\times a_{i-1}+b_i)$

To use this formula to compute the output of a feedforward network for some input $I\in\mathbb{R}^n$, set $a_1=I$, then compute a_2, a_3, ...,a_m, where m is the number of layers.

Introduction

A cost function is a measure of "how good" a neural network did with respect to it's given training sample and the expected output. It also may depend on variables such as weights and biases.

A cost function is a single value, not a vector, because it rates how good the neural network did as a whole.

Specifically, a cost function is of the form

$$C(W,B,S^r,E^r)$$

where W is our neural network's weights, B is our neural network's biases, S^r is the input of a single training sample, and E^r is the desired output of that training sample. Note this function can also potentially be dependent on y^i_j and z^i_j for any neuron j in layer i, because those values are dependent on W, B, and S^r.

In backpropagation, the cost function is used to compute the error of our output layer, δ^L, via

$$\delta^L_j=\frac{\partial C}{\partial a^L_j}\sigma'(z^i_j)$$

.

Which can also be written as a vector via

$$\delta^L = \nabla_a C \odot \sigma'(z^i)$$

.

We will provide the gradient of the cost functions in terms of the second equation, but if one wants to prove these results themselves, using the first equation is recommended because it's easier to work with.

Cost function requirements

To be used in backpropagation, a cost function must satisfy two properties:

1: *The cost function* C *must be able to be written as an average*

$$C = \frac{1}{n}\sum_x C_x$$

over cost functions C_x for individual training examples, x.

This is so it allows us to compute the gradient (with respect to weights and biases) for a single training example, and run Gradient Descent.

2: *The cost function* C *must not be dependent on any activation values of a neural network besides the output values* a^L*.*

Technically a cost function can be dependent on any a^i_j or z^i_j. We just make this restriction so we can backpropagte, because the equation for finding the gradient of the last layer is the only one that is dependent on the cost function (the rest are dependent on the next layer). If the cost function is dependent on other activation layers besides the output one, backpropagation will be invalid because the idea of "trickling backwards" no longer works.

Also, activation functions are required to have an output $0 \le a^L_j \le 1$ for all j. Thus these cost functions need to only be defined within that range (for example, $\sqrt{a^L_j}$ is valid since we are guaranteed $a^L_j \ge 0$).