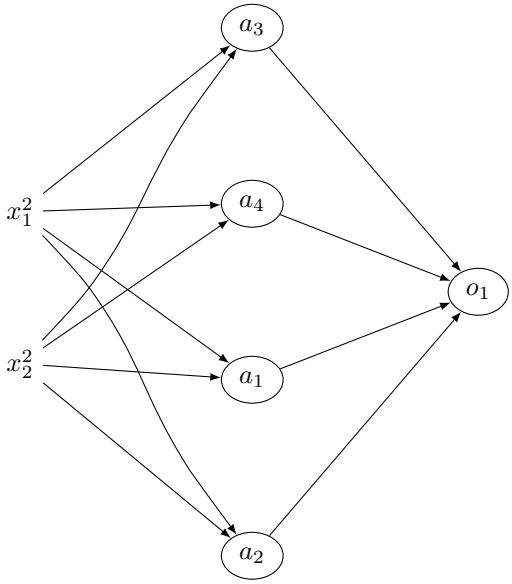

Figure 1: a network

1 random notes

Don't use sigmoid as an activation function. Tanh is generally superior, as it has a mean of zero (sigmoid has a mean of 0.5). Nonzero mean can make training more difficult in subsequent layers. Tanh has a mean of zero. May want a sigmoid as a final activation function (that is, in the output layer) when doing a binary classification problem.

relu is generally good.

2 Notation

2.1 Matrix Dimensions/network layout/notation

A network consists of L layers. The lth layer contains n_l neurons, or nodes. Each layer of the network consists of a number of neurons. Each layer of neurons applies an activation function to a vector of inputs Z^l to produce a vector of outouts, or activations A^l, such that

$$\begin{aligned} A^l &= f(Z^l) \\ &= f(W^l A^{l-1} + b^l) \end{aligned}$$

Where W^l is a matrix of weights for layer l and b^l is a vector of biases. The weight vector acts on the prior layer of activations, or the first layer acts on the raw inputs. for m training examples, the raw inputs X (or A^0) has dimensions (n^0, m), and each the weight matrix for each layer has dimension (n^l, n^{l-1}).

3 Loss and Cost functions

Loss function is the loss/error associated with a single training example Cost function is the loss computed over all examples

3.1 logistic loss

Think of $\hat{y}$ as the conditional probability $\hat{y}(x) = P(y = 1|x)$. Based on our model, the probability $y = 0$ is then $P(y = 0|x) = 1-\hat{y}$. For a single observation, these two outcomes can be summarised as

$$P(y|x) = \hat{y}^y(1 - \hat{y})^{(1-y)} \tag{1}$$

For a set of (y_i, x_i) observations, (Y, X), the likelihood of a given model is given by the product of the conditional probabilities

$$\begin{aligned} L(W|X) &= P(Y|X) \\ &= \prod_i P(y_i|x_i) \\ &= \prod_i \hat{y}_i^{y_i}(1 - \hat{y}_i)^{(1-y_i)} \end{aligned}$$

where $\hat{y}$ is described by our model, and is a function of W and x.

The cost function from logistic loss can then be obtained from the negative log likelihood of $L(W|X)$ above

$$\begin{aligned} J(W, x) &= -\log L(W|X) \\ &= \sum_i (y_i - 1)\log(1 - \hat{y}_i) - y_i \log(\hat{y}_i) \end{aligned}$$

The log of the product reduces to a sum over logs.

The cross entropy is a measure of dissimilarity between two different distributions, p and q.

$$H(p, q) = \sum_i p_i \log q_i \tag{2}$$

The sum here runs over the values that y can take (0 or 1), i.e. the dependant variable of the distributions p and q.

If we interpret p as the distribution of y and q as the distribution of $\hat{y}$, then for our binary classification case we have $y \in \{0, 1\}$, $p \in \{1 - y, y\}$ and $q \in \{1 - \hat{y}, \hat{y}\}$. The cross entropy for a single example is then

$$\begin{aligned} H(p, q) &= -\sum_i p_i \log(q_i) \\ &= -\left((1 - y)\log(1 - \hat{y}) + y \log \hat{y}\right) \end{aligned}$$

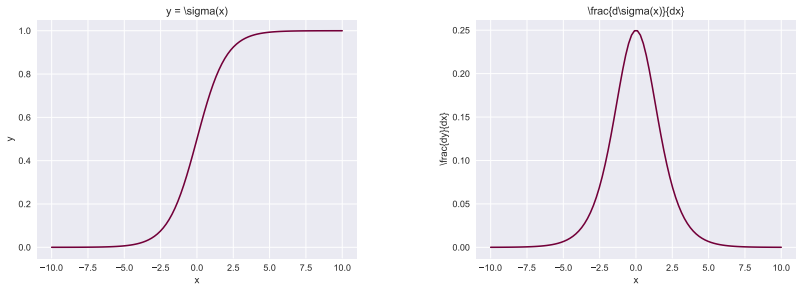

Figure 2: sigmoid activation function and its derivative

The cost function obtained by averaging $H(p, q)$ over all samples is then

$$\begin{aligned} J(W|X) &= \frac{1}{N}\sum_{n}^{N} H(p_n, q_n) \\ &= \frac{-1}{N}\left((1-y_n)\log(1-\hat{y}_n) + y\log\hat{y}_n\right) \end{aligned}$$

Which is equivalent (up to a constant factor) to the cost function obtained from logistic loss above. Minimising cross entropy in this case is equivalent to minimising the logistic loss, which results in the maximum likelihood estimate for the parameters W of the model $\hat{y}$.

4 activation functions and their derivatives

4.1 sigmoid

Sigmoid used to be the default activation function, but in recent times ReLu has proven to be more popular/perform better. Sigmoid is still good for an output function (i.e. activation function in the output layer) in binary classification tasks. The sigmoid function (or logit function) and its derivative are given by

$$\begin{aligned} \sigma(x) &= \frac{1}{1-e^{-x}} \\ \frac{d\sigma(x)}{dx} &= \sigma(x)(1-\sigma(x)) \end{aligned}$$

these are plotted below

4.2 tanh

Tanh is good. It's a nonlinear function, like sigmoid, but for zero input it returns zero output. The mean of this function (given uniform x) is zero, so

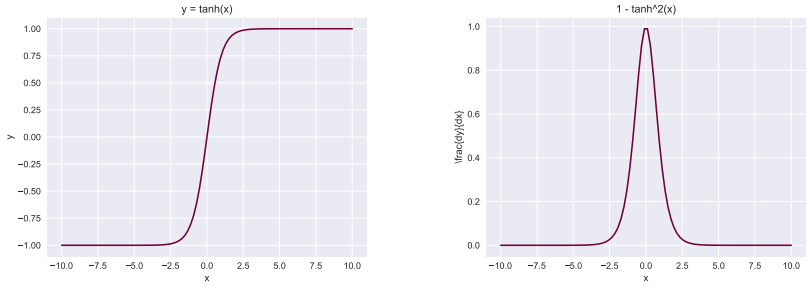

Figure 3: tanh activation function and its derivative

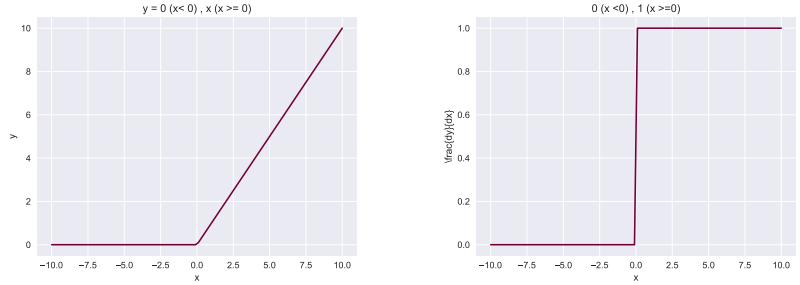

Figure 4: ReLu activation function and its derivative

it handles centred data very well (zero input gives zero output). The tanh function (sinh/cosh) and its derivative are given by

$$\begin{aligned} \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ \frac{d\sigma(x)}{dx} &= 1 - \tanh^2(x) \end{aligned}$$

these are plotted below

4.3 ReLu

rectified linear unit. Max(0,x). This is a very popular activation, as it tends to give good results. Should in general be the first choice for an activation function (rather than sigmoid/tanh, though there may be cases where these perform better).

4.4 leaky ReLu

Gradient saturation and such. If a Relu Network ever gets to a state where the inputs are large and negative, all the gradients will vanish, and the network

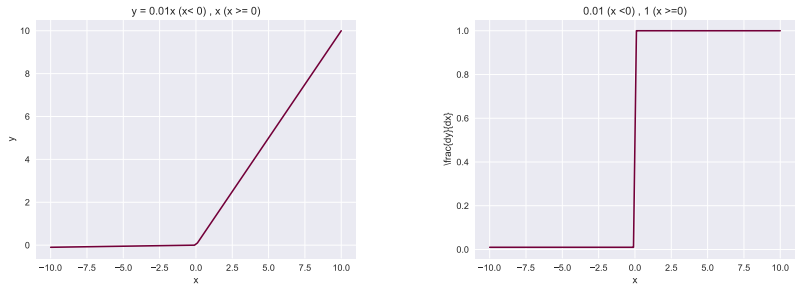

Figure 5: Leaky Relu activation function and its derivative

will stop updating, it gets stuck. Leaky Relu has a very small (1%) output for negative x, so the gradient never fully vanishes. It's a little more robust