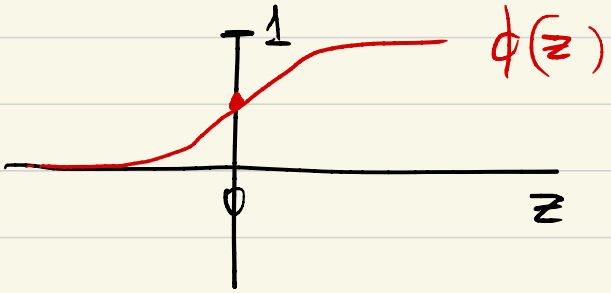
C. Logistic regression (LR)

In LR a sigmoid activation function is used, i.e.

$$\phi(z) = \frac{1}{1+e^{-z}}$$



To see where this comes from, let p be the probability of a certain (binary) event (such as a coin), then define $\left(\text{logit} : [0,1] \to \mathbb{R}\right)$

$$\text{logit}(p) = \ln \frac{p}{1-p}$$

where $p/(1-p)$ is the odds ratio.

Let p now indicate the conditional probability that a particular sample belongs to a class $\left(\text{e.g. } y = 1\right)$, under input data $\underline{x}$, i.e.

$$p = P\left(y = 1 \mid \underline{x}\right)$$

then

$$\text{logit}\left(p\right) = z = \sum_{j=0}^{n} w_j x_j$$

defines the activitation function in LR. Since

$$z = \ln \frac{p}{1-p} \rightarrow -z = \ln\left(\frac{1}{p} - 1\right)$$

$$\rightarrow \quad e^{-z} = \frac{1}{p} - 1 \rightarrow p = \frac{1}{1+e^{-z}}.$$

Assuming that the samples $(i = 1 \dots N)$ in the training data are independent, then the cost function $L(\underline{w})$ is defined as

$$L(\underline{w}) = P(\underline{y} \mid \underline{x}; \underline{w}) =$$

$$= \prod_{i=1}^{N} P(y^i \mid x^i, \underline{w}) =$$

$$= \prod_{i=1}^{N} \left(\phi(z^i)\right)^{y^i} \left(1 - \phi(z^i)\right)^{1-y^i}$$

probability that feature gives y^i

not.

In practise, we use $\mathcal{E}(\underline{w}) = -\ln L(\underline{w})$ given by

$$\mathcal{E}(\underline{w}) = -\sum_{i=1}^{N} \{ y^i \ln \phi(z^i) + (1 - y^i) \ln(1 - \phi(z^i)) \}$$

Consider $N=1$ (single sample),

then

$$J(\underline{w}) = -\left(y \ln \phi(z) + (1-y) \ln\left(1-\phi(z)\right)\right)$$

Note that the quantizer in LR is given by

$$q(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

hence $J(\underline{w}) = -\ln\left(1-\phi(z)\right)$ $y=0$

and $J(\underline{w}) = -\ln \phi(z)$ $y=1$

In summary, we penalize wrong predictions ($\phi(z)=0$ when $y=1$ and $\phi(z)=1$ when $y=0$) by very large costs.

Using gradient descend, we again update

$$\Delta \underline{w}_{k+1} = -\eta \nabla E(\underline{w}_k)$$

where $(\nabla E)_j$ is now computed as

$$-(\nabla E)_j = \frac{\partial}{\partial w_j}\left(y \ln \phi(z) + (1-y)\ln(1-\phi(z))\right)$$

(assume for simplicity that $N=1$)

$$= \left(\frac{y}{\phi(z)} - \frac{(1-y)}{(1-\phi(z))} \right) \frac{\partial}{\partial w_j} \phi(z)$$

use: $\phi'(z) = \frac{d\phi}{dz} = \left(\frac{1}{1+e^{-z}}\right)' =$

$$= \frac{+e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$= \phi(z)\left(1-\phi(z)\right)$$

hence $\frac{\partial}{\partial w_j} \phi(z) = \phi'(z) \frac{\partial}{\partial w_j} z$

$$\left(-\nabla \mathcal{J}\right)_j = \left(\frac{y}{\phi(z)} - \frac{1-y}{1-\phi(z)} \right) \phi(z)\left(1-\phi(z)\right) * \frac{\partial}{\partial w_j} z \quad , \; z = \sum_j w_j x_j$$

$$= \left(y\left[1-\phi(z)\right) - \left(1-y\right)\phi(z) \right) x_j$$

$$= \left(y - \phi(z) \right) x_j$$

So the update of the weights is

$$\Delta \underline{w}_{k+1} = -\eta \, \nabla \mathcal{J}(\underline{w})$$

$$\left(\Delta \underline{w}_{k+1}\right)_j = \eta \sum_{i=1}^{N} \left(y^i - \phi(z^i) \right) x_j^i$$

similar to the gradient descend rule of the Adaline.

D. Overfitting

Overfitting is the problem that a model (Perceptron, Adaline or LR) perform well on the training data, but not on the test data. A solution to over fitting is regularization by adding terms to the cost functn to avoid large weights. For example, in LR the cost function is written as

$$E(\underline{w}) = -\sum_{i=1}^{N} \left(y^i \ln \phi(z^i) + (1 - y^i) \ln \left(1 - \phi(z^i)\right) \right) + \frac{\lambda}{2} \| \underline{w} \|^2$$

which adds a term

$$-\left(\nabla E\right)_j = -\lambda w_j$$

to the gradient of the cost function.