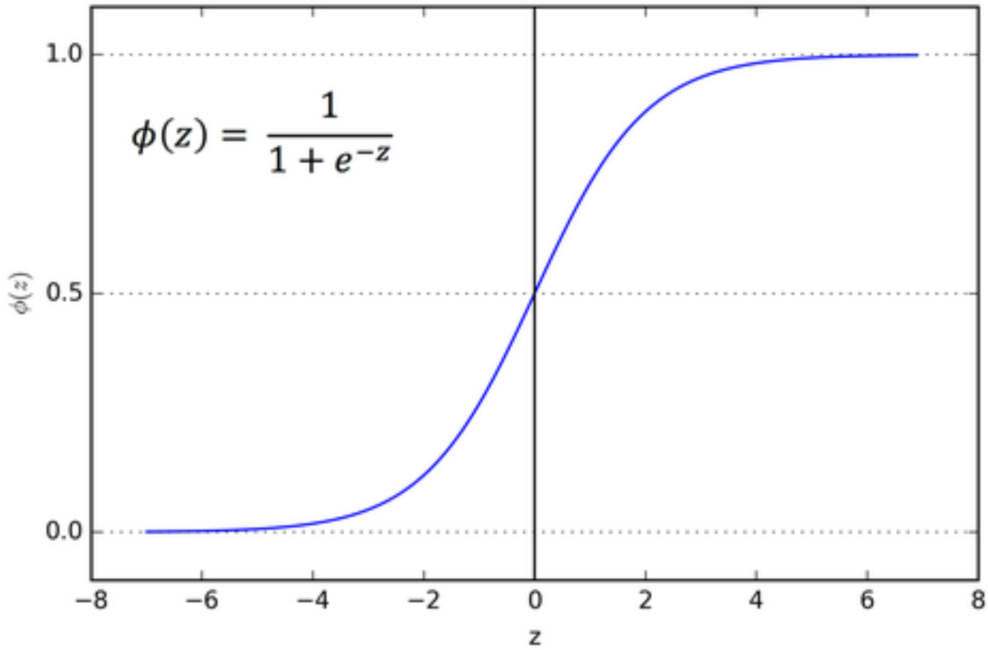
Logistic Regression

Although the name might confuse, please note that this is a *classification* algorithm.

Considering a problem of two-class classification, in logistic regression the posterior probability of class C_1 can be written as a logistic sigmoid function:

$$p(C_1|\phi) = \frac{1}{1+e^{-\mathbf{w}^T\phi}} = \sigma(\mathbf{w}^T\phi) \tag{1}$$



and $p(C_2|\phi) = 1 - p(C_1|\phi)$

Applying the *Maximum Likelihood* approach...

Given a dataset $\mathcal{D} = \{\mathbf{x}_n, t_n\}$, $t_n \in \{0,1\}$, we have to maximize the probability of getting the right label:

$$P(\mathbf{t}|\mathbf{X},\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}, \quad y_n = \sigma(\mathbf{w}^T\phi_n) \tag{2}$$

Taking the negative log of the likelihood, the *cross-entropy* error function can be defined and it has to be minimized:

$$L(\mathbf{w}) = -\ln P(\mathbf{t}|\mathbf{X},\mathbf{w}) = -\sum_{n=1}^{N}(t_n \ln y_n + (1-t_n)\ln(1-y_n)) = \sum_{n}^{N} L_n \tag{3}$$

Differentiating and using the chain rule:

$$\frac{\partial L_n}{\partial y_n} = \frac{y_n - t_n}{y_n(1-y_n)}, \quad \frac{\partial y_n}{\partial \mathbf{w}} = y_n(1-y_n)\phi_n \tag{4}$$
$$\frac{\partial L_n}{\partial \mathbf{w}} = \frac{\partial L_n}{\partial y_n}\frac{\partial y_n}{\partial \mathbf{w}} = (y_n - t_n)\phi$$

The gradient of the loss function is

$$\nabla L(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n \tag{5}$$

It has the same form as the gradient of the sum-of-squares error function for linear regression. But in this case y is not a linear function of $\mathbf{w}$ and so, there is no closed form solution. The error function is *convex* (only one optimum) and can be optimized by standard *gradient-based* optimization techniques. It is, hence, easy to adapt to the online learning setting.

Talking about *Multiclass Logistic Regression*...

For the multiclass case, the posterior probabilities can be represented by a *softmax* transformation of linear functions of feature variables:

$$p(C_k|\phi) = y_k(\phi) = \frac{e^{\mathbf{w}_k^T\phi}}{\sum_j e^{\mathbf{w}_j^T\phi}} \tag{6}$$

$\phi(\mathbf{x})$ has been abbreviated with ϕ for simplicity.

Maximum Likelihood is used to directly determine the parameters

$$p(\mathbf{T}|\Phi, \mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N}\underbrace{\left(\prod_{k=1}^{K} p(C_k|\phi_n)^{t_{nk}}\right)}_{\text{Term for correct class}} = \prod_{n=1}^{N}\left(\prod_{k=1}^{K} y_{nk}^{t_{nk}}\right) \tag{7}$$

where $y_{nk} = p(C_k|\phi_n) = \frac{e^{\mathbf{w}_k^T\phi_n}}{\sum_j e^{\mathbf{w}_j^T\phi_n}}$

The *cross-entropy* function is:

$$L(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\Phi, \mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\left(\sum_{k=1}^{K} t_{nk}\ln y_{nk}\right) \tag{8}$$

Taking the gradient

$$\nabla L_{\mathbf{w}_j}(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N}(y_{nj} - t_{nj})\phi_n \tag{10}$$