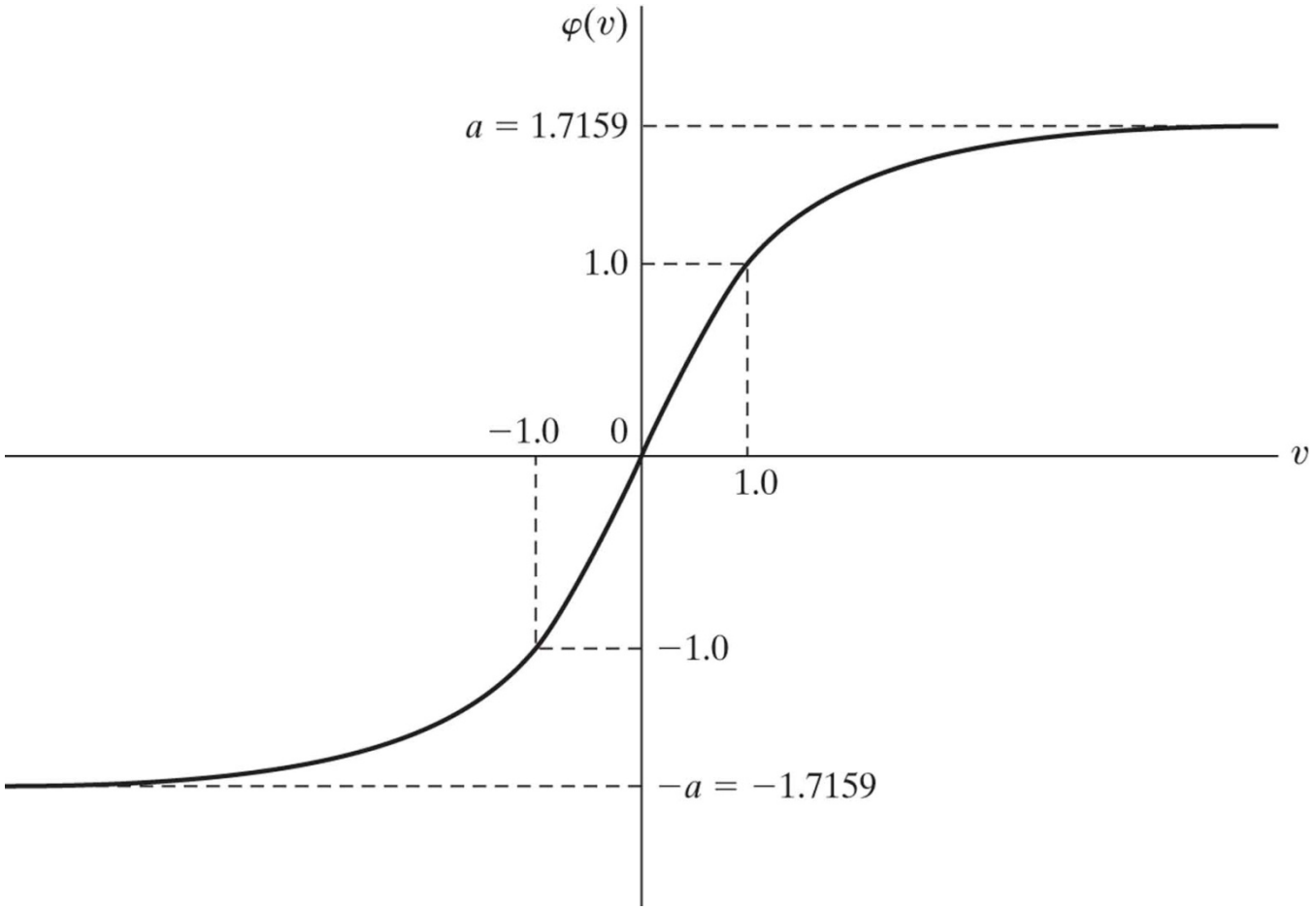
How to use MLP & BP

Applying backpropagation

- NN structure
- Thresholding/activation function
- Initialization of weight (zero mean, small numbers from uniform distribution)
- Training sample preparation
- Training sample presentation
- Select learning rate (or learning algorithm)
- Forward computation of signal flow (x(n), w(n) -> y(n))
- Backward computation of error (d(n)-y(n))
- Iterate until stopping criteria are met

Graph of the hyperbolic tangent function ϕ(*v*) $\alpha \tanh(bv)$ for α 1.7159 and b = 2/3.The recommended target values are +1 and –1.

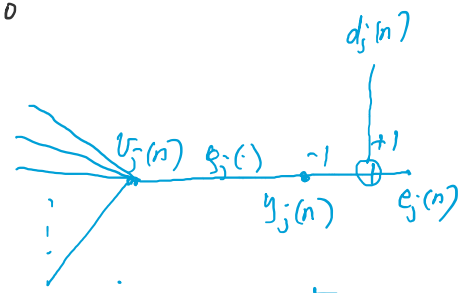


Thresholding Function $\varphi(\cdot)$ and local gradient

- Logistic function

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-a v_j(n))} \qquad a > 0$$

$$\varphi_j'(v_j(n)) = \frac{a\exp(-a v_j(n))}{[1+\exp(-a v_j(n))]^2}$$



j : an output neuron

use $y_j(n) = \varphi_j(v_j(n))$

$$\varphi_j'(v_j(n)) = a\, y_j(n)\,[1-y_j(n)]$$

local gradient $\delta_j(n) = e_j(n)\cdot\varphi_j'(v_j(n))$

$$= a\,[d_j(n)-y_j(n)]\,y_j(n)\,[1-y_j(n)]$$

if j is a hidden neuron

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n)\, w_{kj}(n)$$

$$= a\,y_j(n)(1-y_j(n)) \sum_k \delta_k(n)\, w_{kj}(n)$$

- hyperbolic tangent

$$\varphi_j(v_j(n)) = a\tanh(b\,v_j(n)) = \frac{a(e^{bx} - e^{-bx})}{(e^{bx} + e^{-bx})}$$

$$\varphi_j'(v_j(n)) = \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right]$$

if j is an output layer neuron

$$\delta_j(n) = e_j(n)\,\varphi_j'(v_j(n)) = \frac{b}{a}\left[d_j(n) - y_j(n)\right]\left[a - y_j(n)\right]\left[a + y_j(n)\right]$$

if j is a hidden layer neuron

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n)\,w_{kj}(n)$$

$$= \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right] \sum_k \delta_k(n)\,w_{kj}(n)$$

Learning rate

- The smaller the learning rate, the smaller the changes to the weight during each iteration, the smoother the weight trajectory in the weight space. However, this results in slower rate of learning
- Large learning rate may make the network become unstable
- The idea of adding momentum to the weight update during each iteration -> generalized delta rule
- Adaptive learning rate

Variants of Back-propagation

Back propagation with momentum

motivation: speed up w/o losing stability

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

α: momentum constant

To generalize a bit:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t)$$

$$= -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial \mathcal{E}(t)}{\partial w_{ji}(t)}$$

① $\frac{\partial \mathcal{E}(t)}{\partial w_{ji}(t)}$

② $\frac{\partial \mathcal{E}(t-1)}{\partial w_{ji}(t-1)}$

if ① + ② same sign → momentum accelerates $\Delta w_{ji}(n)$

if ① + ② different sign → momentum slows down $\Delta w_{ji}(n)$

Stopping criteria 1

- No well-defined rules
- No special consideration of local vs. global minimum in the error surface
- Necessary condition -> $g(w)=0$, i.e., gradient of the error surface wrt the weight vector w no longer changes
- May result in slow learning

Stopping criteria 2

- Use the fact that the error function become stationary at w^*
- A reduction of the averaged squared error is considered sufficiently small if it is in the range of 0.1%~1%

Stopping criteria 3

- Inspecting training error and generalization error, idea based in statistical learning theory

Heuristics to make backpropagation work better

- Choose stochastic over batch for redundant and large training data set. Batch may result in rank deficient Jacobian matrix
- Maximize info content – at each iteration, 1) use an example that leads to large training error, 2) use one that is much different from previous. Practically, shuffle training samples
- Use a symmetric activation function about the origin with reasonable slopes, e.g., see next page
- Set the desired target value to prior to the saturation level to avoid misleading the learning process

Heuristics to make backpropagation work better

- Preprocess training data - Normalizing training sample inputs: to un-correlate the training inputs, use PCA, to scale the covariances to similar level
- Rule of thumb – the previous pre-processing should make the weight sum of inputs at each approximately at the linear region, prior to saturation
- Observations – for non-zero mean inputs, the larger the mean of the inputs, the poorer the Jacobian condition number, and thus the slower the learning process

Illustrating the operation of mean removal, decorrelation, and covariance equalization a two-dimensional input space.