

CS 4824/ECE 4424: Neural Networks II

Acknowledgement:

Many of these slides are derived from Tom Mitchell, Pascal Poupart, Pieter Abbeel, Eric Eaton, Carlos Guestrin, William Cohen, and Andrew Moore.

Optimizing the Weights

- Parameters: $\langle W^{(1)}, W^{(2)}, \dots \rangle$
- Objective:
 - **Error minimization**
 - **Backpropagation (aka backprop)**

Backpropagation Algorithm

- Two phases:
 - Forward phase: compute output z_j for each node j
 - Backward phase: compute output δ_j for each node j

Forward Phase

- Propagate inputs forward through the network to compute the output of each node
- Output z_j at node j
 - $z_j = h(a_j)$ where $a_j = \sum_i w_{ji} z_i$

Backward Phase

- Use chain rule to recursively compute gradient
 - For each weight w_{ji}

$$\delta_{ji} = \frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

- Let $\delta_j = \frac{\partial E_n}{\partial a_j}$

- Then $\delta_j = \begin{cases} h'(a_j)(z_j - y_j) & \text{base case : } j \in \text{output nodes} \\ h'(a_j) \sum_k w_{kj} \delta_k & \text{recursion : } j \in \text{hidden nodes} \end{cases}$

- Since

$$a_j = \sum_k w_{ji} z_i \text{ then } \frac{\partial a_j}{\partial w_{ji}} = z_i$$

- Therefore, $\delta_{ji} = \frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i$

An Example

- A simple two-layer network:
 - Hidden nodes (Tanh): $h(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
 - Note: $\tanh'(a) = (1 - (\tanh(a))^2)$
 - Output nodes (Identity): $h(a) = a$
- Objective function: squared error

The derivation

- Forward phase
 - Hidden nodes: $a_j = z_j =$
 - Output nodes: $a_k = z_k =$
- Backward phase
 - Output nodes: $\delta_k =$
 - Hidden nodes: $\delta_j =$
- Gradients
 - Hidden layer: $\frac{\partial E_n}{\partial w_{ji}} =$
 - Output layer: $\frac{\partial E_n}{\partial w_{kj}} =$

The derivation

- Forward phase

- Hidden nodes: $a_j = \sum_i w_{ji} x_i$ $z_j = \tanh(a_j)$

- Output nodes: $a_k = \sum_j w_{kj} z_j$ $z_k = a_k$

- Backward phase

- Output nodes: $\delta_k = z_k - y_k$

- Hidden nodes: $\delta_j = [1 - (z_j)^2] \sum_k w_{kj} \delta_k$

- Gradients

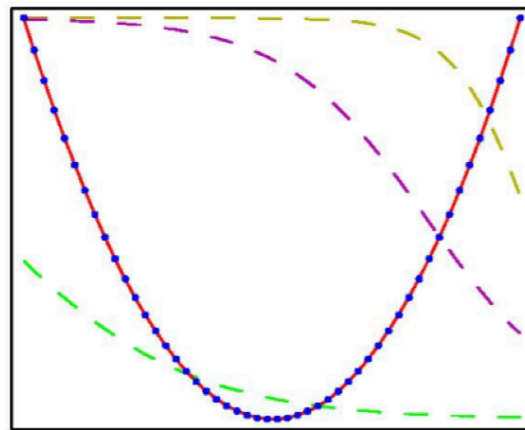
- Hidden layer: $\frac{\partial E_n}{\partial w_{ji}} = \delta_j x_i = [1 - (z_j)^2] \sum_k w_{kj} \delta_k x_i$

- Output layer: $\frac{\partial E_n}{\partial w_{kj}} = \delta_k z_j = (z_k - y_k) z_j$

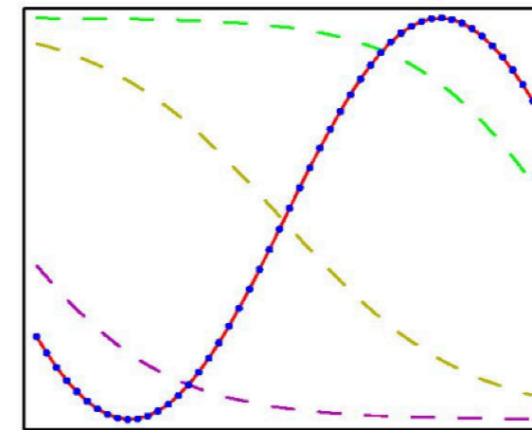
Non-linear regression examples

- Two layer network:
 - 3 tanh hidden units and 1 identity output unit

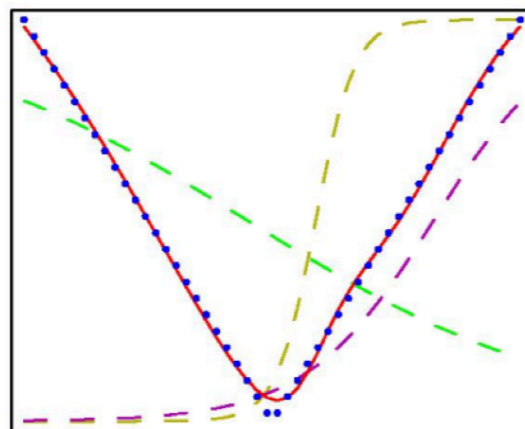
$$y = x^2$$



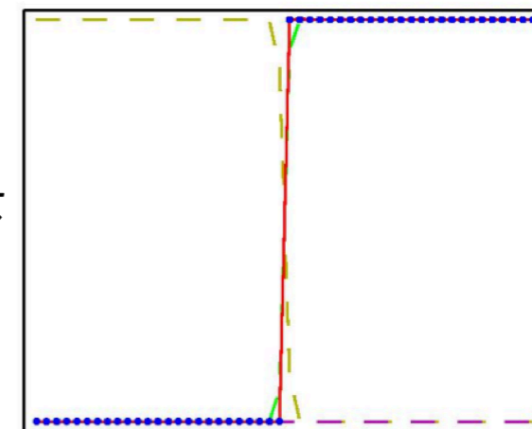
$$y = \sin x$$



$$y = |x|$$



$$y = \int_{-\infty}^x \delta(t) dt$$



Demo Time 😊

<https://playground.tensorflow.org/>