Introduction to Neural Networks Continued

Hunter Glanz

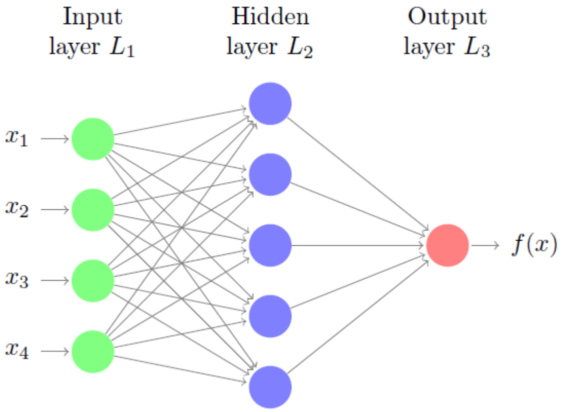
OUTLINE

Overview

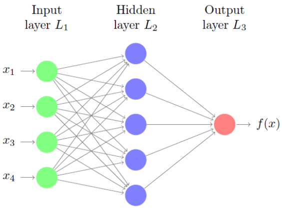
- Neural networks are highly parameterized models inspired by the human brain

Overview

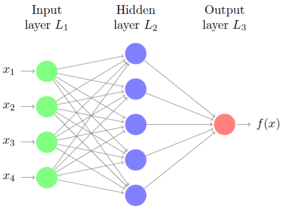
- Neural networks are highly parameterized models inspired by the human brain



More Detail

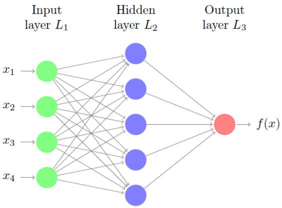


More Detail



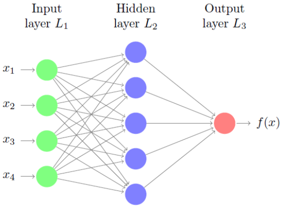
- Four predictors or inputs x_j

More Detail



- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$

More Detail



- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$
- Single output unit $o = h(w_0^{(2)} + \sum_{\ell=1}^{5} w_\ell^{(2)} a_\ell)$

So Many Weights!

- Four predictors or inputs x_j

So Many Weights!

- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$

So Many Weights!

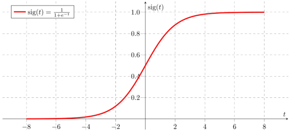
- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$
- Single output unit $o = h(w_0^{(2)} + \sum_{\ell=1}^{5} w_\ell^{(2)} a_\ell)$

So Many Weights!

- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$
- Single output unit $o = h(w_0^{(2)} + \sum_{\ell=1}^{5} w_\ell^{(2)} a_\ell)$
- Typically $g(t) = 1/(1 + e^{-t})$ – a sigmoid

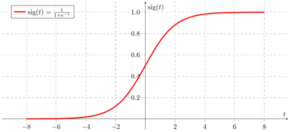
So Many Weights!

- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$
- Single output unit $o = h(w_0^{(2)} + \sum_{\ell=1}^{5} w_\ell^{(2)} a_\ell)$
- Typically $g(t) = 1/(1+e^{-t})$ – a sigmoid



So Many Weights!

- Four predictors or inputs x_j
- Five hidden units $a_\ell = g(w_{\ell 0}^{(1)} + \sum_{j=1}^{4} w_{\ell j}^{(1)} x_j)$
- Single output unit $o = h(w_0^{(2)} + \sum_{\ell=1}^{5} w_\ell^{(2)} a_\ell)$
- Typically $g(t) = 1/(1+e^{-t})$ – a sigmoid



- For quantitative regression, h is typically the identity
- For classification, h is once again the sigmoid

Some Updates on Activation Functions

- Some other popular choices:

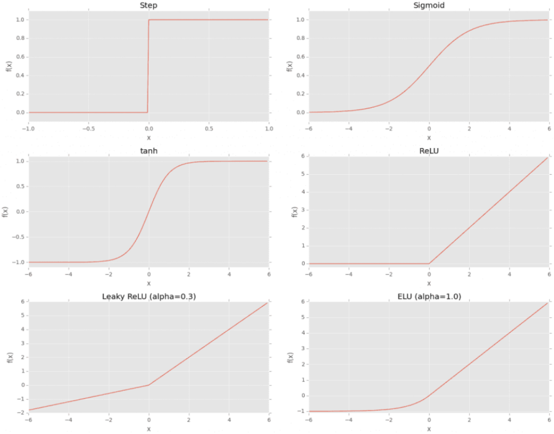


Figure 4: *Top-left:* Step function. *Top-right:* Sigmoid activation function. *Mid-left:* Hyperbolic tangent. *Mid-right:* ReLU activation (most used activation function for deep neural networks). *Bottom-left:* Leaky ReLU, variant of the ReLU that allows for negative values. *Bottom-right:* ELU, another variant of ReLU that can often perform better than Leaky ReLU.

Some Activation Function Notes

- Sigmoid and tanh are better than the simple step function...they're differentiable and smooth

Some Activation Function Notes

- Sigmoid and tanh are better than the simple step function...they're differentiable and smooth
- However, sigmoid and tanh asymptotically approach their saturation values (0 or 1, and -1 or 1 respectively)...

Some Activation Function Notes

- Sigmoid and tanh are better than the simple step function...they're differentiable and smooth
- However, sigmoid and tanh asymptotically approach their saturation values (0 or 1, and -1 or 1 respectively)...
  - A **saturated** neural network is one where most of the hidden nodes have values close to these values
  - If hidden nodes are **saturated** that means their pre-activation sum-of-products is relatively large or small
  - This leads to a situation where a small change in the input-to-hidden weights during training will likely not change the sum-of-products very much....i.e. **training stalls our or moves very slowly**
  - Additionally, saturated models are often overfitted

Some Activation Function Notes

- Sigmoid and tanh are better than the simple step function...they're differentiable and smooth
- However, sigmoid and tanh asymptotically approach their saturation values (0 or 1, and -1 or 1 respectively)...
  - A **saturated** neural network is one where most of the hidden nodes have values close to these values
  - If hidden nodes are **saturated** that means their pre-activation sum-of-products is relatively large or small
  - This leads to a situation where a small change in the input-to-hidden weights during training will likely not change the sum-of-products very much....i.e. **training stalls our or moves very slowly**
  - Additionally, saturated models are often overfitted
- **A solution:**
  - Use other activation functions like ReLU (Rectified Linear Unit) defined as $f(x) = max(0, x)$.