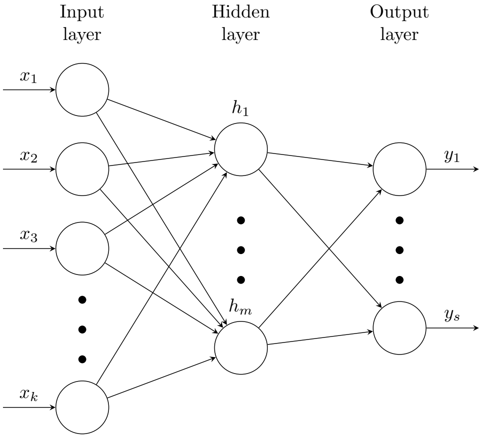
Neural Networks

- Neural Networks: mimicking the operation of neurons in the brain.



- Form a graph with input nodes and output nodes,
  and with hidden layers of extra nodes between them.
- The output of each layer is the input to the next layer.
  The forwarded input goes through an activation at each hidden layer.
- Each node is meant to play the role of a neuron in the brain.
- The neural network is also called the multi-layer perceptron or MLP, in short.

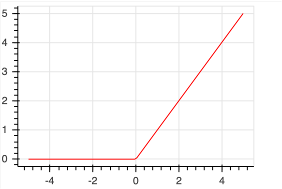
- In each layer of a typical neural network, the process is similar to a logistic regression.

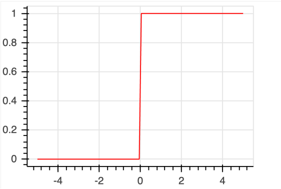
  Neural networks $\approx$ multiple layers of logistic regressions

- Activation functions are given

  by a sigmoid or a rectified linear unit (relu).

$$r(x) = \max(x, 0), \quad r'(x) = u_0(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x < 0. \end{cases}$$





Set-up

- Multi-class Classification:

$$\mathcal{D} = \{(x_1, \dots, x_k; t)\}, \quad t \in \{1, 2, \dots, s\}$$

- Build a network with k input nodes and s output nodes.

  Put one hidden layer with m nodes.

- Each output node will produce probability for the input to be in the corresponding class.

  (Multi-class logistic regression = one-layer neural network)

For simplicity, we take $N = 1$ in what follows.

$$
\begin{aligned}
\text{input:} \quad & \boldsymbol{x} = [x_1, \ldots, x_k, 1] \\
\text{hidden layer:} \quad & \boldsymbol{x}\boldsymbol{w}^{(1)} = [z_1, \ldots, z_m] \\
& \qquad\qquad \text{with } \boldsymbol{w}^{(1)} \text{ of size } (k+1) \times m \\
& \boldsymbol{h} = [h_1, \cdots, h_m, 1] \\
& \quad = [\sigma(z_1), \ldots, \sigma(z_m), 1] \quad (\sigma : \text{sigmoid}) \\
\text{output:} \quad & \boldsymbol{y} = [y_1, \ldots, y_s] = \boldsymbol{\sigma}(\boldsymbol{h}\boldsymbol{w}^{(2)}) \quad (\boldsymbol{\sigma} : \text{softmax}) \\
& \qquad\qquad \text{with } \boldsymbol{w}^{(2)} \text{ of size } (m+1) \times s
\end{aligned}
$$

As in logistic regression, we want to learn
the best values of $\boldsymbol{w}^{(\ell)}$ ($\ell = 1, 2$) using the training data.

- We have

$$\boxed{\boldsymbol{x} \rightsquigarrow \boldsymbol{h} \rightsquigarrow \boldsymbol{y}}.$$

  This process is considered as forward propagation.

- We can develop more general neural networks by considering more complex directed graphs with many layers and by adopting activation functions different from σ.

- However, there should be no oriented cycles in the directed graph to ensure that the outputs are deterministic functions of the inputs. In other words, a network must be feed-forward.

- Cross-entropy

$$\boxed{E(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}) = -\sum_{i=1}^{s} t_i \ln y_i}$$

  where t is identified with a binary vector $\mathbf{t} = [t_1, \ldots, t_n]$.

- We will use gradient descent and need to take derivatives. In particular, we need to use the chain rule inductively.

$$\nabla_{\boldsymbol{w}^{(2)}} E \rightsquigarrow \nabla_{\boldsymbol{w}^{(1)}} E$$

- This process is called backpropagation.

- Write $\boldsymbol{xw}^{(1)} = [z_1^{(1)}, \ldots, z_m^{(1)}]$ and $\boldsymbol{hw}^{(2)} = [z_1^{(2)}, \ldots, z_s^{(2)}]$.
- As in multi-class logistic regression,

$$\frac{\partial E}{\partial z_i^{(2)}} = -\sum_{j=1}^{s} t_j \frac{1}{y_j} \frac{\partial y_j}{\partial z_i^{(2)}} = -\sum_{j=1}^{s} t_j(\delta_{i,j} - y_i) = y_i - t_i,$$

$$\frac{\partial E}{\partial w_{p,q}^{(2)}} = \sum_{i=1}^{s} \frac{\partial E}{\partial z_i^{(2)}} \frac{\partial z_i^{(2)}}{\partial w_{p,q}^{(2)}} = h_p(y_q - t_q),$$

and

$$\nabla_{\boldsymbol{w}^{(2)}} E = \boldsymbol{h}^\top (\boldsymbol{y} - \mathbf{t}).$$

Next we have

$$
\begin{aligned}
\frac{\partial E}{\partial z_i^{(1)}} &= \sum_{j=1}^{s} \frac{\partial E}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial z_i^{(1)}} = \sum_{j=1}^{s} (y_j - t_j) \sum_{\ell=1}^{m} \frac{\partial z_j^{(2)}}{\partial h_\ell} \frac{\partial h_\ell}{\partial z_i^{(1)}} \\
&= \sum_{j=1}^{s} (y_j - t_j) \sum_{\ell=1}^{m} w_{\ell j}^{(2)} \delta_{i,\ell} h_i (1 - h_i) = h_i (1 - h_i) \sum_{j=1}^{s} (y_j - t_j) w_{ij}^{(2)}, \\
\frac{\partial E}{\partial w_{p,q}^{(1)}} &= \sum_{i=1}^{m} \frac{\partial E}{\partial z_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial w_{p,q}^{(1)}} = x_p h_q (1 - h_q) \sum_{j=1}^{s} (y_j - t_j) w_{qj}^{(2)},
\end{aligned}
$$

and

$$
\nabla E_{\boldsymbol{w}^{(1)}} = \boldsymbol{x}^T \left[h_q (1 - h_q) \sum_{j=1}^{s} (y_j - t_j) w_{qj}^{(2)} \right]_{q=1,\ldots,m} .
$$

- Write

$$\delta_i^{(a)} = \frac{\partial E}{\partial z_i^{(a)}} \quad \text{for } a = 1, 2.$$

The formula

$$\boxed{\delta_i^{(1)} = \sigma'(z_i^{(1)}) \sum_{j=1}^{s} \delta_j^{(2)} w_{ij}^{(2)}}$$

is called the backpropagation formula.

Warnings

- The error function E is non-convex (and non-concave).
- Initialization – don't take the zero vector for $\boldsymbol{w}^{(1)}$
- More susceptible to over-fitting – a lot more of parameters

Remarks

- Complex neural networks with multiple layers are usually called deep neural networks.
- Most deep learning models are based on convolutional neural networks.
- TensorFlow is an open-source software library for machine learning. It has a particular focus on deep neural networks.
- Keras is an open-source library that provides a Python interface for neural networks. It acts as an interface for the TensorFlow library.