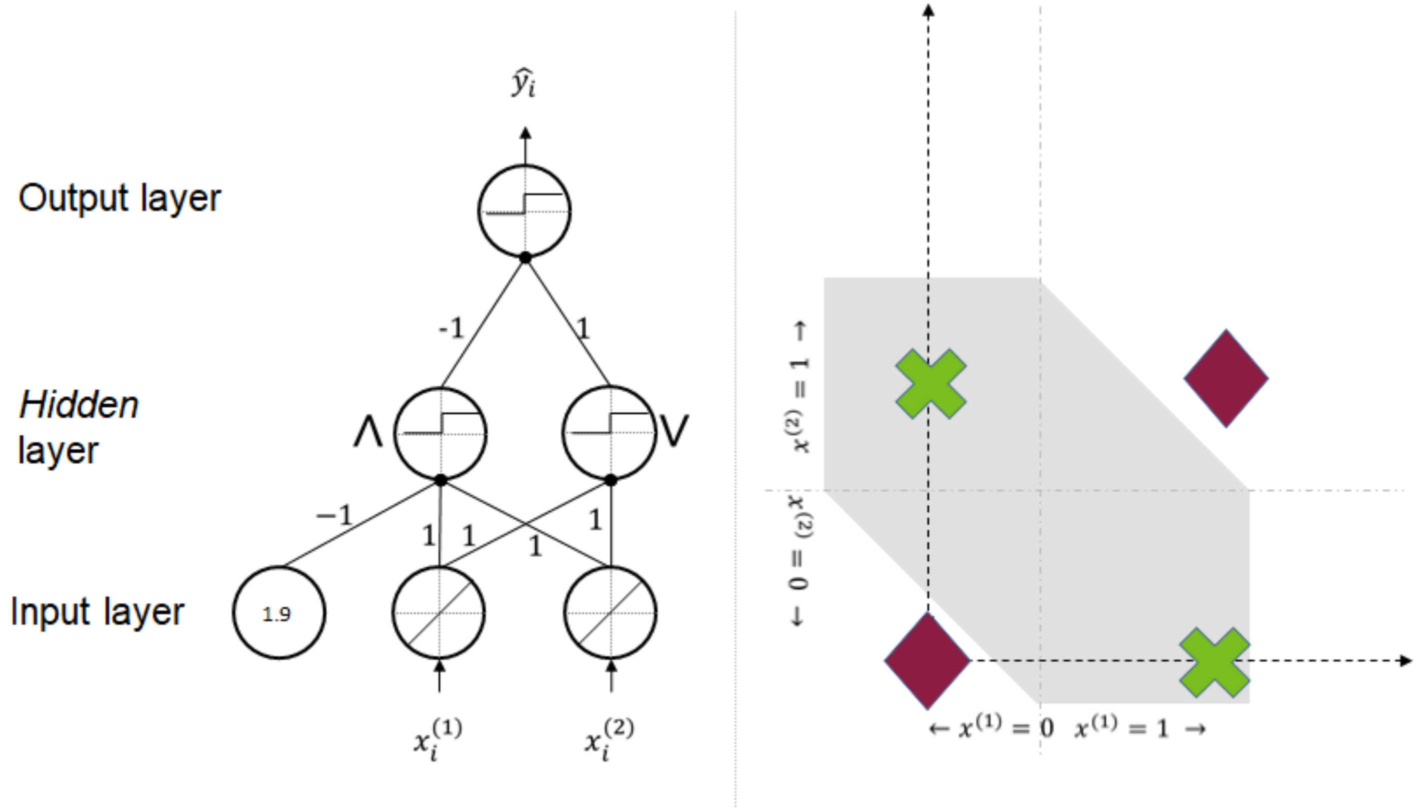
The Question of Learning

- **How can the network learn proper parameters from the given samples?**
  - Can the Perceptron algorithm be used?



Difficulty in Learning for MLP

Perceptron Learning Algorithm

- The weight update of the neuron is proportional to the "error" computed as $y_i - \hat{y}_i$.
  - This requires us to know the target output y_i.

Multi-layer Perceptron

- Except for the neurons on the output layer, other neurons (on the *hidden* layers) do not really have a target output given.

Back-propagation (BP) Learning for MLP

- The key: Properly distribute error computed from output layer back to earlier layers to allow their weights to be updated in a way that reduce the error
  - The basic philosophy of the BP algorithm

- Differentiable activation functions
  - We can use – e.g.,
    - the logistic neurons
    - neurons with sigmoid activation
    - or its variants

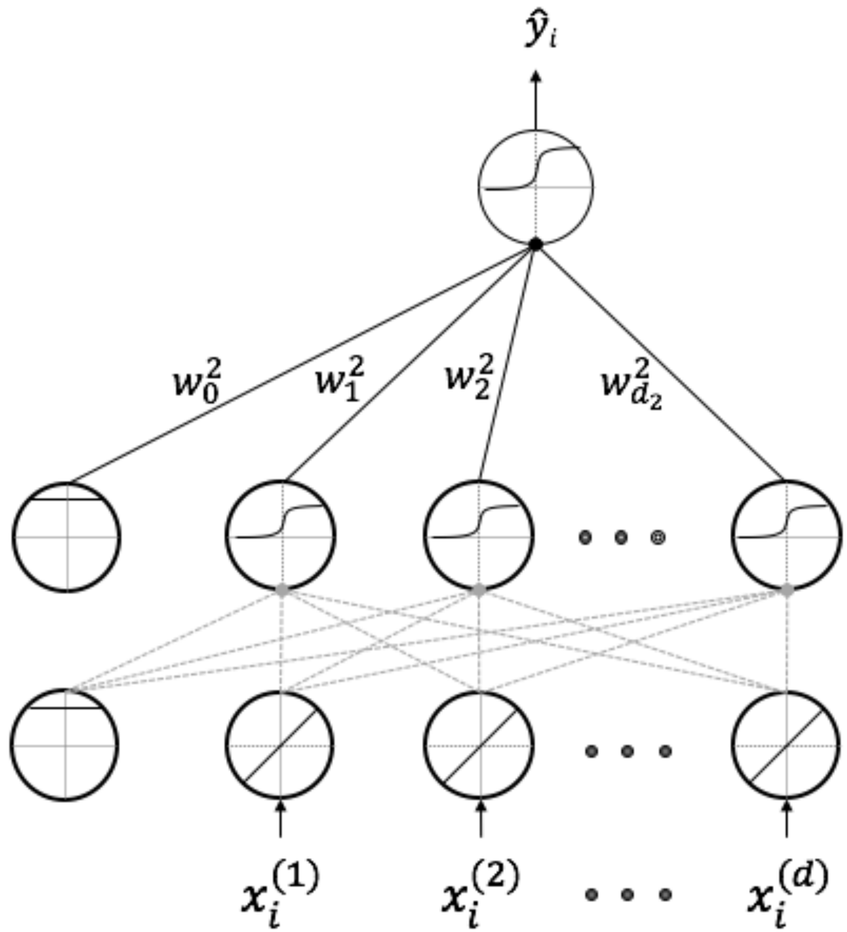
A Multi-Layer Neural Network

Using Logistic Neurons

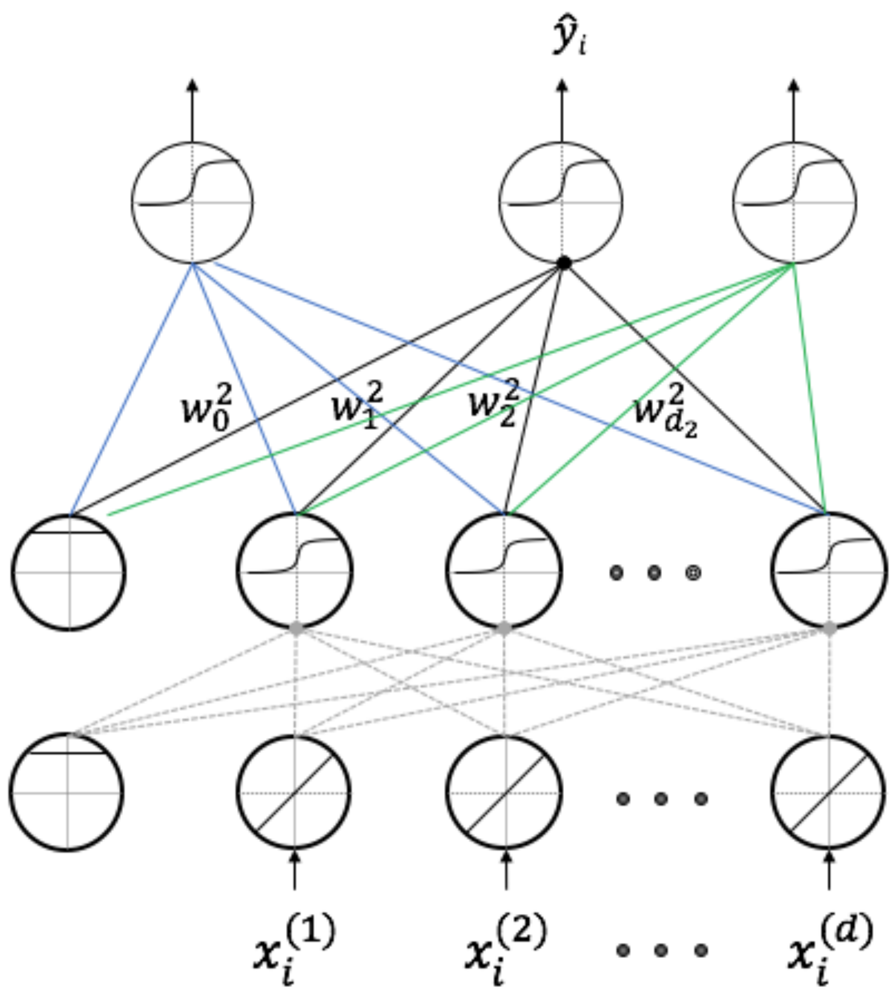
Output layer

Hidden layer

Input layer

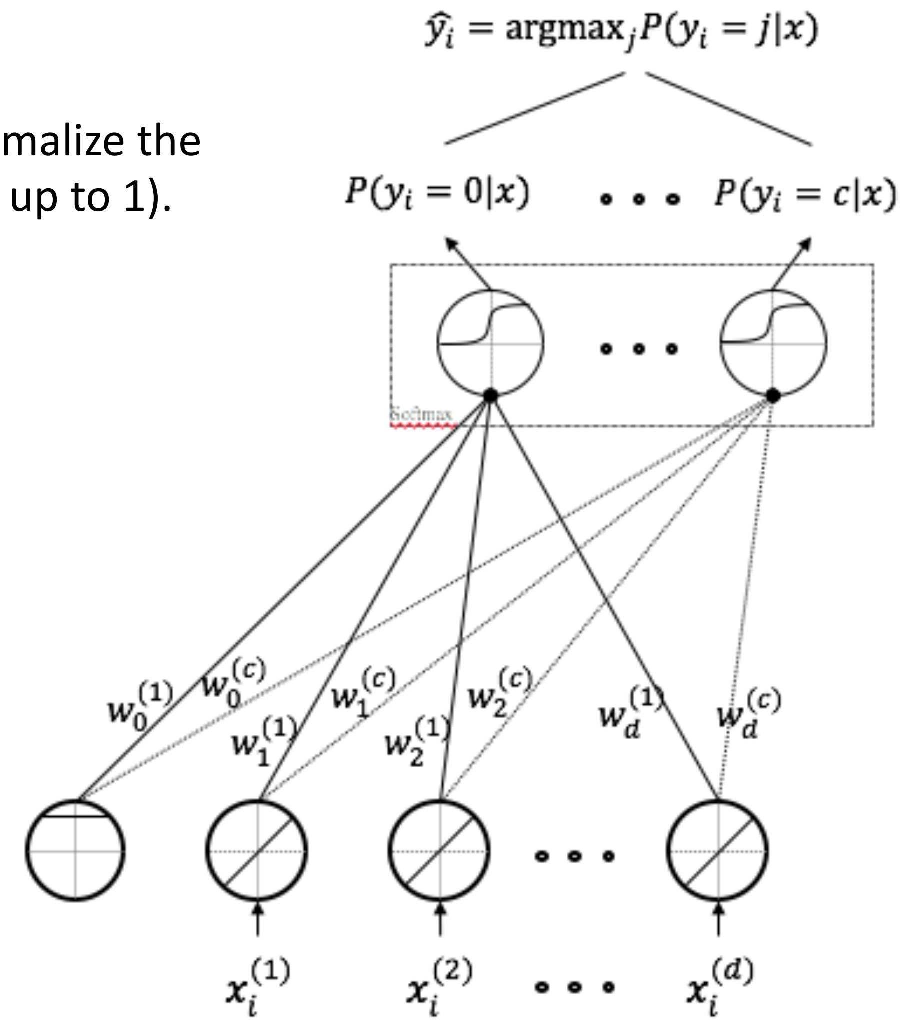


Handling Multiple (>2) Classes



Softmax for Handling Multiple Classes

Using *softmax* to normalize the outputs (so they add up to 1).



How to compute "errors" in this case?

- Consider the cross-entropy as a loss function:

$$l(\boldsymbol{W}) = \sum_{i=1}^{n} \sum_{j=1}^{c} \mathbb{I}_j(y_i) \log P(y_i = j | x_i) \ ,$$

$$\mathbb{I}_j(y_i) = \begin{cases} 1, & if \ y_i = j \\ 0, & otherwise \end{cases}$$

Neural Networks and Deep Learning