Chapter 7 Neural Networks and Neural Language Models

10/20/20 9:46 PM

Neural Units

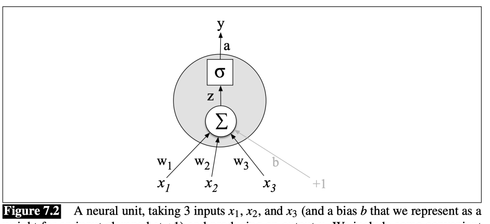

Figure 7.2 A neural unit, taking 3 inputs x_1, x_2, and x_3 (and a bias b that we represent as a weight for an input clamped at +1) and producing an output y. We include some convenient intermediate variables: the output of the summation, z, and the output of the sigmoid, a. In this case the output of the unit y is the same as a, but in deeper networks we'll reserve y to mean the final output of the entire network, leaving a as the activation of an individual node.

Activation function

- Sigmoid
  - $y = \sigma(z) = \dfrac{1}{1+e^{-z}}$
- Tanh

$$y = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

- ReLU

$$y = max(x, 0)$$

Feed Forward Neural Networks (MLP)

Three kinds of nodes: input units, hidden units, and output units

Single layer MLP

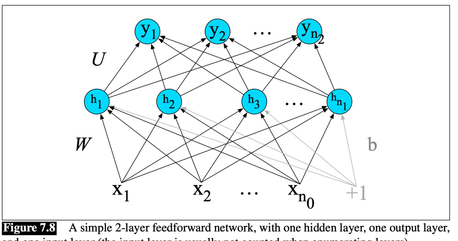

Figure 7.8 A simple 2-layer feedforward network, with one hidden layer, one output layer, and one input layer (the input layer is usually not counted when enumerating layers).

- Each x_n: input scaler value
- *W*: weight matrix, W_ij represents the weight of connecting input unit x_i to hidden unit h_j
- Output of hidden layer (a single vector *h*): sigmoid function of linear
  - $h = \sigma(Wx + b)$
- Output layer: output weight matrix *U*, output vector *z*

$$z = Uh$$

  - And finally apply softmax for z vector:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \quad 1 \le i \le d$$

- Summary: 3 steps for 2-layer MLP (single hidden layer)

$$h = \sigma(Wx + b)$$

  - $z = Uh$
  - $y = \text{softmax}(z)$
  - Note for non-linear activation function (e.g. ReLU), apply activation function g(-) instead of sigma

Training Neural Nets

Loss function

- cross-entropy loss (same as logistic regression)
- Binary classification:
  - $L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$
- Multi-class classification:
  - $L_{CE}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$

Backpropagation (backprop)

- Begin by annotating the final node with partial derivative = 1

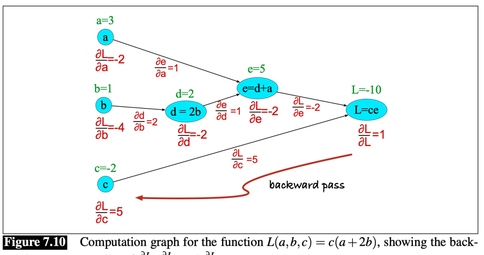

Figure 7.10 Computation graph for the function $L(a,b,c) = c(a+2b)$, showing the backward pass computation of $\frac{\partial L}{\partial a}$, $\frac{\partial L}{\partial b}$, and $\frac{\partial L}{\partial c}$.

Neural Language Models

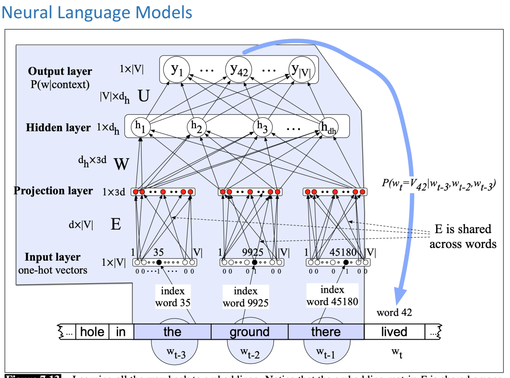

Figure 7.13 Learning all the way back to embeddings. Notice that the embedding matrix E is shared among the 3 context words.

- Input layer * E gives Projection Layer, which should be embeddings of the corresponding words
- W is same weight matrix for passing hidden units
- Output layer: predicted conditional probability of each word in vocabulary (so it's a vector of size |V|)

$$e = (Ex_1, Ex_2, ..., Ex)$$

$$h = \sigma(We + b)$$

$$z = Uh$$

$$y = \text{softmax}(z)$$