Neural nets unit

- Introduction and principles (**Today**)
- Neural net language model lab (Friday)
- RNNs and LTSMs (next week Monday)

Today:

- Why we care about neural nets
- Principles of neural nets
  - The perceptron
  - Multi-layer perceptrons
  - Feed-forward etc
- Neural nets as language models

From the reading assignment on Canvas:

> *This is a reading where I'll have different expectations for those have taken CSCI 381 from those who haven't.*
>
> **If you have taken CSCI 381 Machine Learning**, *read the entire chapter to the level that you pretty much understand all of it. A good portion of it will be review; anything that was fuzzy in CSCI 381 should be clearer after you've read this chapter, having now seen this information a second time.*
>
> **If you have not taken CSCI 381 Machine Learning**, *read 7.(0-5). This will be dense, like some of the recent readings. Since we cover this more intentionally in CSCI 381, I won't expect students in this course to internalize this material at the same level. But you should learn the essential terms and concepts and take note of anything you don't understand. Sections 7.6 and following are optional.*

Machine learning is about training a function $\mathbb{R}^D \to T$ from data. Common machine learning tasks are

- *Regression*, where the target type T is $\mathbb{R}$
- *Classification*, where the target type T is a finite set (that is, a set of classes or labels)
  - *Binary classification*, where the target type T is $\{0, 1\}$
  - *Probabalistic classification*, where the target type is $[0, 1]^{|T|}$; that is, the target values are probability distributions over a finite set of classes T
- *Density estimation*, where the target type T is $[0, 1]$.

A *training algorithm* selects a model (function) from a model family by finding weights (parameters).

A **perceptron** is a function $\mathbb{R}^D \to \mathbb{R}$ defined as

$$p(\boldsymbol{x}) = a(\boldsymbol{w} \cdot \boldsymbol{x} + b) = a\left(b + \sum_{i=0}^{D-1} w_i x_i\right)$$

where

- $\boldsymbol{w}$ is the vector of weights
- b is the bias term
- a is the activation function

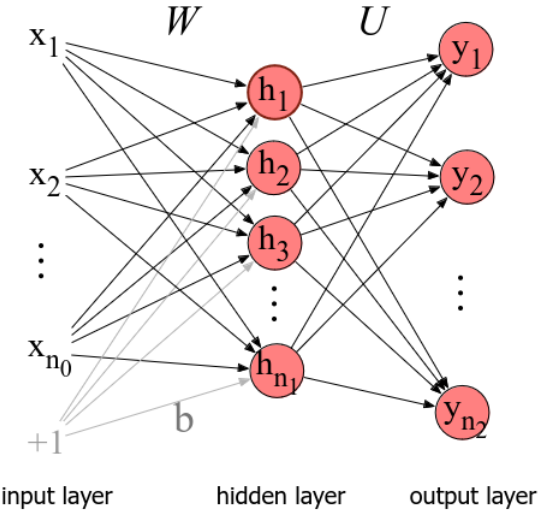

Figure 7.8 A simple 2-layer feedforward network, with one hidden layer, one output layer, and one input layer (the input layer is usually not counted when enumerating layers).

Jurafsky and Martin, Sec 7.3, pg 8

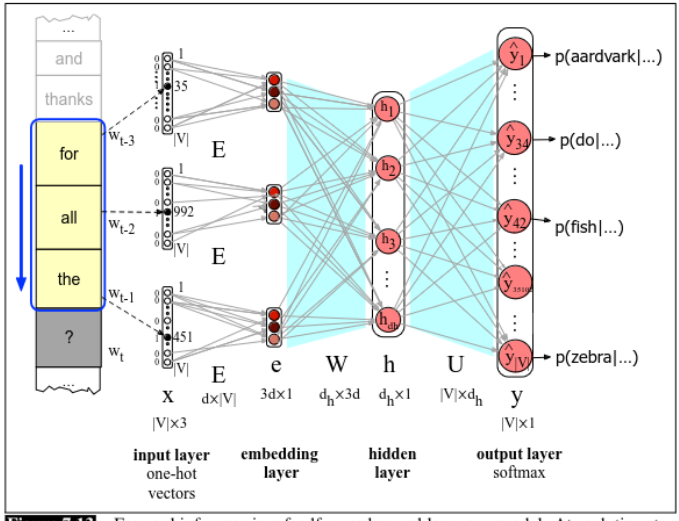

Figure 7.13 Forward inference in a feedforward neural language model. At each timestep t the network computes a d-dimensional embedding for each context word (by multiplying a one-hot vector by the embedding matrix $\mathbf{E}$), and concatenates the 3 resulting embeddings to get the embedding layer $\mathbf{e}$. The embedding vector $\mathbf{e}$ is multiplied by a weight matrix $\mathbf{W}$ and then an activation function is applied element-wise to produce the hidden layer $\mathbf{h}$, which is then multiplied by another weight matrix $\mathbf{U}$. Finally, a softmax output layer predicts at each node i the probability that the next word w_t will be vocabulary word V_i.

Coming up:

- Take word2vec/embeddings quiz (Thurs, Nov 16)
- Read J&M chapter 9 (Mon, Nov 20)

- Work on stylo assignment

Word2Vec assignment coming...