

Introduction to neural networks

- In lecture 03 we outlined a general framework by which we can conduct discriminative learning using the following computational units:

$$\mathbf{x} \rightarrow f(\mathbf{x}; \boldsymbol{\theta}) \rightarrow \mathbf{z} \rightarrow \sigma_{softmax}(\mathbf{z}) \rightarrow P(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta})$$

- We then outlined a procedure called maximum likelihood estimation (MLE) to learn the parameters $\boldsymbol{\theta}$ from a set of labeled examples $D = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(M)}, \mathbf{y}^{(M)})\}$:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{i=1}^M \mathbf{y}^{(i)} \cdot \log P(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{i=1}^M \sum_{j=1}^K \mathbf{y}_j^{(i)} \log \left[\frac{\exp[f(\mathbf{x}^{(i)}; \boldsymbol{\theta})_j]}{\sum_{j'} \exp[f(\mathbf{x}^{(i)}; \boldsymbol{\theta})_{j'}]} \right]$$

- And showed that gradient descent is an excellent means by which to find $\hat{\boldsymbol{\theta}}$ when $f(\mathbf{x}; \boldsymbol{\theta})$ is at least once differentiable with respect to $\boldsymbol{\theta}$; it's an especially good/necessary choice when the likelihood function is not convex.
- Neural language modeling fits directly into this framework, making the following selections:
 - \mathbf{y} is a one-hot encoded representation of our language tokens (e.g., words)
 - $f(\mathbf{x}; \boldsymbol{\theta})$ is a neural network (there are many different flavors, or *architectures*)

Neural language modeling: feature representation

