Assignment 2

- Published: *Feb 11*
- Deadline: *March 4*
- Implement backpropagation for a neural network
- Find hyperparameters for optimization and generalization
\[ L = \text{cross-entropy}(y_1, \ldots, y_K) \]

\[ y_k = \text{softmax}(o_1, o_2, \ldots, o_K) \]

\[ o_k = \sum_{j=1}^{J} v_{jk} h_j \]

\[ h_j = \text{logistic}(z_j) \]

\[ z_j = \sum_{i=1}^{I} w_{ij} x_i \]

I: # input units
J: # hidden units
K: # output classes
Use chain rule to backpropagate gradients

Hidden to output weights:

\[ \frac{\partial L}{\partial v_{jk}} = \frac{\partial L}{\partial o_k} \frac{\partial o_k}{\partial v_{jk}} \]

Input to hidden weights:

\[ \frac{\partial L}{\partial w_{ij}} = \sum_{k=1}^{K} \frac{\partial L}{\partial o_k} \frac{\partial o_k}{\partial h_j} \frac{\partial h_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}} \]
To Do, possible snags, and tips

- derive specific expressions for gradients
- derive expressions on paper before implementing them
- compute gradients for multiple training cases
- check specific expression for loss function, average gradients over minibatch
- vectorize expressions
- make use of gradient checker (not a guarantee for correct gradients but helpful)

Helpful material:
- Backprop video (Lecture 3d)
- Backprop example
- Code from Jan 23 tutorial
- Read assignment carefully