

# Lecture 14

## Cost Functions and Regularization

07 March 2016

Taylor B. Arnold  
Yale Statistics  
STAT 365/665

The Yale logo, consisting of the word "Yale" in a blue, serif font.

Notes:

- ▶ Problem set 5 is due a week from today

## Outline for this week:

- ▶ Monday
  - ▶ walk through R reference implementation of back-propagation
  - ▶ address issues of slow learning
  - ▶ over-fitting / regularization
  - ▶ initializing weights
- ▶ Wednesday
  - ▶ quasi second order SGD
  - ▶ learning rate schedule
  - ▶ hyper-parameter selection heuristics

## Today's Notes

For today, rather than re-writing someone else's notes, I am simply going to go through the notes on this page:

*<http://neuralnetworksanddeeplearning.com/chap3.html>*

The primary reason for this is that there are some great visualizations embedded on the page; also, as I have used the exact same notation as presented here, there should be no great confusion in doing so.

There are three summary slides here for reference purposes, which summarize the main points I am covering today.

## Cross-entropy and soft-max

Due to the shape of the sigmoid neuron, weights that are very far from their optimal values learn slowly in a plain, vanilla network. Two ways to fix this are to use the cross-entropy cost-function, defined as:

$$C = - \sum_j [y_j \log(a_j^L) + (1 - y_j) \log(1 - a_j^L)]$$

For a single sample, and similarly for an entire mini-batch.

Another common approach is to define what is termed a softmax layer. This redefines the activations of the output layer,  $a^L$ , as follows:

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$$

This has the additional benefit that the last layer is easily interpreted as a sequence of probabilities.

## Regularization in Neural Networks

As the size of neural networks grow, the number of weights and biases can quickly become quite large. State of the art neural networks today often have billions of weight values. In order to avoid over-fitting, one common approach is to add a penalty term to the cost function. Common choices are the  $\ell_2$ -norm, given as:

$$C = C_0 + \lambda \sum_i w_i^2$$

Where  $C_0$  is the unregularized cost, and the  $\ell_1$ -norm:

$$C = C_0 + \lambda \sum_i |w_i|.$$

The distinction between these is similar to the differences between lasso and ridge regression.

## Dropout

A very different approach to avoiding over-fitting is to use an approach called *dropout*. Here, the output of a randomly chosen subset of the neurons are temporarily set to zero during the training of a given mini-batch. This makes it so that the neurons cannot overly adapt to the output from prior layers as these are not always present. It has enjoyed wide-spread adoption and massive empirical evidence as to its usefulness.