EECS/EEAP 484 Computational Intelligence, Fall 2018
Problem Set 3: 2-D Convolutional Neural Networks with Error Backpropagation
Assigned: 10/6/18
Due: 10/15/18

In this problem set, we explore convolutional neural networks applied to 2-D images. In this simple example, the training data contains a "feature" comprised of 0's and 1's in a 2x3 rectangle. The location of the feature is expressed in terms of the (i,j) coordinates of the upper-left corner of this rectangle within a larger image. The training data has targets that specify a "1" in this location for each feature.

In the example code, there is a single hidden layer. The output layer has its synapses (W2) hard coded. Synapses in layer W1 have repeating patterns of weights (comprising the convolution). These patterns are described in terms of 6 parameters, comprising a kernel. The kernel is also 2x3, or it can be strung out as a 6x1 (or 1x6) vector.

Starter code is provided. You need to edit:
  cnn_main() (to define W1)
and edit the functions:
  compute_bias_sensitivity_vecs(...)
  compute_dW_from_sensitivities(...)
  compute_dE_dkvec(..)

You can check your results by comparing numerical values vs. computed numerical estimates. (Functions to compute numerical estimates are provided).

You should find that, for the very small default problem size, your network converges quickly to good values for a convolution kernel.

Once your code is debugged, do some experiments. Try different features to recognize. Try different (larger) images sizes (by editing "gen_training_data()").

For your submission, include:
*your matlab code for the modified functions
*evidence that your derivatives are correct (relative to numerical estimates)
*evidence that your synapse matrix performs equivalent to 2-D Matlab convolution (conv2() function)
*comments on your training results
*description of your experiments and observations