session 12
24.01.2018

Exercise 12.1 - Structures and cell arrays

Write a function

`function net=make_ffnet(Nlayers, Nneurons, hasBiasNeuron)`

That creates and initializes a feed-forward neural network with `Nlayers` layers. The number of (non-bias) neurons in each layer is given by the vector `Nneurons` and whether a layer has a bias neuron or not is determined by the logical vector `hasBiasNeuron`. These two vectors must be of size `Nlayers`$\times 1$. The output of the function is a structure with the fields

`Nlayers`	the number of layers
`Nneurons`	the vector with the number of neurons in each layer
`hasBiasNeuron`	the vector determining whether a layer has a bias neuron
`O`	a cell array such that `net.O{l}(j)` is the value of the neuron output for neuron j in layer l. Neuron number `Nneurons(l)+1` is the bias neuron (if there is any)
`I`	a cell array such that `net.I{l}(j)` is the weighted sum of inputs to neuron j in layer l
`factiv`	a cell array such that `net.factiv{l}` is a function handle to the activation function used in layer l. Use tanh for the output layer and ReLU for all other layers
`dfactiv`	$\frac{d}{dx}$`factiv(x)`
`w`	cell array such that `net.w{l}(j,k)` is the weight between neuron j of layer l and neuron k of layer $l+1$. The weights should be initialized to normally distributed random numbers with variance $2/(M+N)$, where M is the number of neurons in layer l, and N is the number of neurons in layer $l+1$
`E`	a function handle to a function $E(y, \hat{y})$, that computes how well the objective was achieved, if the output vector is y and the true result $\hat{y}$. Use by default the least squares function $E(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
`dE`	a function handle to a function that computes the partial derivatives of E with respect to $y_1, \ldots, y_N$

Exercise 12.2 - Evaluating feed-forward networks
Write a function
`function net_out=ffnet_eval(net_in, input_layer)`
that sets the input neurons in `net_in` to the values in the vector `input_layer`, and evaluates the neural network (i.e. computes all values `net_out.I` and `net_out.O`.)

Exercise 12.2 - Text recognition
It is time for an application! We want to use a (already trained) neural network to recognize hand-written digits.

- Create a network with 3 layers. The input layer has 784 neurons, corresponding to the pixel values of a 28×28 image. The values encode shades of gray ranging from white (0) to black (1). The hidden layer shall have 800 neurons and the output layer 10 neurons. The first two layers have an additional bias neuron.

- We will not train the network this week. Instead: set the weights of the network to those from the file `weights.mat`. The weights have been generated by training the network on a set of ~ 60000 handwritten digits. The training was such that $E(y, \hat{y})$ was minimized on the training set. The true result vector $\hat{y}$ had a +1 for the neuron that corresponded to the digit that the training image represented and -1 for all other output neurons. E.g. images from `train{3}` had $\hat{y} =$`[-1 -1 1 -1 -1 -1 -1 -1 -1 -1].'`

- The training `train` set and a second set `test` (not used for training) can be loaded from the file `digits.mat`. Both are cell arrays with ten cells, corresponding to digits 0 to 9. Each cell is a $N \times 784$ matrix that contains N different handwritten digits. For instance `test{1}(3,:)` would be a valid vector for the `input_layer`. It corresponds to the third image of digit 0 from the test set.

- Plot a few of the test digits. Use them as inputs to your neural network and compute the outputs. Are the digits recognized correctly?

- Test your network with the complete test set. What are the recognition rates for each digit?