Implementing backpropagation

Now we've seen that the error term for the output layer is

$\delta_k = (y_k - \hat{y}_k) f'(a_k)$

and the error term for the hidden layer is

$$\delta_j = \sum [w_{jk} \delta_k] f'(h_j)$$

For now we'll only consider a simple network with one hidden layer and one output unit. Here's the general algorithm for updating the weights with backpropagation:

- Set the weight steps for each layer to zero
  - The input to hidden weights $\Delta w_{ij} = 0$
  - The hidden to output weights $\Delta W_j = 0$
- For each record in the training data:
  - Make a forward pass through the network, calculating the output $\hat{y}$
  - Calculate the error gradient in the output unit, $\delta^o = (y - \hat{y}) f'(z)$ where $z = \sum_j W_j a_j$, the input to the output unit.
  - Propagate the errors to the hidden layer $\delta_j^h = \delta^o W_j f'(h_j)$
  - Update the weight steps:
    - $\Delta W_j = \Delta W_j + \delta^o a_j$
    - $\Delta w_{ij} = \Delta w_{ij} + \delta_j^h a_i$
- Update the weights, where η is the learning rate and m is the number of records:
  - $W_j = W_j + \eta \Delta W_j / m$
  - $w_{ij} = w_{ij} + \eta \Delta w_{ij} / m$
- Repeat for e epochs.

Backpropagation exercise

Now you're going to implement the backprop algorithm for a network trained on the graduate school admission data. You should have everything you need from the previous exercises to complete this one.

Your goals here:

- Implement the forward pass.
- Implement the backpropagation algorithm.
- Update the weights.

backprop.py | data_prep.py | binary.csv | solution.py

```
        hidden_output = sigmoid(hidden_input)

        output = sigmoid(np.dot(hidden_output,
                                weights_hidden_output))

        ## Backward pass ##
        # TODO: Calculate the network's prediction error
        error = y - output

        # TODO: Calculate error term for the output unit
        output_error_term = error * output * (1 - output)

        ## propagate errors to hidden layer

        # TODO: Calculate the hidden layer's contribution to the error
        hidden_error = np.dot(output_error_term, weights_hidden_output)

        # TODO: Calculate the error term for the hidden layer
        hidden_error_term = hidden_error * hidden_output * (1 - hidden_output)

        # TODO: Update the change in weights
        del_w_hidden_output += output_error_term * hidden_output
        del_w_input_hidden += hidden_error_term * x[:, None]

    # TODO: Update weights
    weights_input_hidden += learnrate * del_w_input_hidden / n_records
```

RESET QUIZ TEST RUN SUBMIT ANSWER

Note: This code takes a while to execute, so Udacity's servers sometimes return with an error saying it took too long. If that happens, it usually works if you try again.

NEXT