Implementing backpropagation

Now we've seen that the error in the output layer is

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

and the error in 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.

Implementing Backpro...

Your goals here:

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

backprop.py | data_prep.py | binary.csv | soultion.py

```
25
26  for e in range(epochs):
27      del_w_input_hidden = np.zeros(weights_input_hidden.shape)
28      del_w_hidden_output = np.zeros(weights_hidden_output.shape)
29      for x, y in zip(features.values, targets):
30          ## Forward pass ##
31          # TODO: Calculate the output
32          hidden_input = np.dot(x, weights_input_hidden)
33          hidden_output = sigmoid(hidden_input)
34
35          output = sigmoid(np.dot(hidden_output,
36                                  weights_hidden_output))
37
38          ## Backward pass ##
39          # TODO: Calculate the error
40          error = y - output
41
42          # TODO: Calculate error gradient in output unit
43          output_error = error * output * (1 - output)
44
45          # TODO: propagate errors to hidden layer
46          hidden_error = np.dot(output_error, weights_hidden_output) * \
47                         hidden_output * (1 - hidden_output)
48
49          # TODO: Update the change in weights
50          del_w_hidden_output += output_error * hidden_output
51          del_w_input_hidden += hidden_error * x[:, None]
52
53      # TODO: Update weights
54      weights_input_hidden += learnrate * del_w_input_hidden / n_records
55      weights_hidden_output += learnrate * del_w_hidden_output / n_records
56
```

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.

Implementing Backpro...

NEXT