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.

- Implement the backpropagation algorithm.
- Update the weights.

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

```
import numpy as np
from data_prep import features, targets, features_test, targets_test

np.random.seed(21)

def sigmoid(x):
    """
    Calculate sigmoid
    """
    return 1 / (1 + np.exp(-x))


# Hyperparameters
n_hidden = 2  # number of hidden units
epochs = 900
learnrate = 0.005

n_records, n_features = features.shape
last_loss = None
# Initialize weights
weights_input_hidden = np.random.normal(scale=1 / n_features ** .5,
                                        size=(n_features, n_hidden))
weights_hidden_output = np.random.normal(scale=1 / n_features ** .5,
                                         size=n_hidden)

for e in range(epochs):
```

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