DLM Labor: Backpropagation

Wintersemester 2017

Aufgaben:

3_1:

Ersetzen Sie die Aktivierungsfunktion (tanh) an allen benötigten Stellen durch ReLU.

```
27 #Feedfoward pass
28 def feedforward(X, model):
29     W1 = model['W1']
30     b1 = model['b1']
31     W2 = model['W2']
32     b2 = model['b2']
33
34     #W.x+b
35     z1 = X.dot(W1) + b1
36
37     #Activation function
38     a1 = np.tanh(z1)
39
40     #W.a1 + b
41     z2 = a1.dot(W2) + b2
42
```

3_2:

Erweitern Sie das Modell um ein weiteren Hidden Layer

```
121
122 # Train Model
123 # This function learns parameters for the neural network and returns the model.
124 # - neurons: Number of neurons in the hidden layer
125 # - epochs: Number of passes through the training data for gradient descent
126 def train_model(X, y, nn_hdim, epochs=200):
127
128     # Initialize the parameters to random values. We need to learn these.
129     np.random.seed(0)
130     W1 = np.random.randn(nn_input_dim, nn_hdim) / np.sqrt(nn_input_dim)
131     b1 = np.zeros((1, nn_hdim))
132     W2 = np.random.randn(nn_hdim, nn_output_dim) / np.sqrt(nn_hdim)
133     b2 = np.zeros((1, nn_output_dim))
134
135     # This is what we return at the end
136     model = { 'W1': W1, 'b1': b1, 'W2': W2, 'b2': b2}
137
138     # Gradient descent. For the complete training data...
139     for i in xrange(0, epochs):
140
141         # Forward propagation
142         probs, model = feedforward(X, model)
143
144         # Backpropagation
145         deltas = backprop(X, y, probs, model)
146
147         # Gradient descent parameter update
148         model = parameter_update(model, deltas, learning_rate/len(X))
149
150     #Accuracy
151     y_pred = np.argmax(probs, axis=1)
152     accuracy = np.mean(np.array(y_pred == y, dtype=np.uint))
153     print 'Training accuracy for epoch %d : %f ' %(i, accuracy)
```