Code for Neural Networks

Matt Olson

October 13, 2017

Tricks of the Trade

- Initialization and scaling: He initialization, batch normalization
  - He, et al 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification
  - Ioffe, et al 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Regularization: Dropout, early stopping, penalties
  - Srivastava, et al 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting
- Activation functions: relu, elu, selu, leaky relu, ...
- Reusing pre-trained layers from other models

Mini batch SGD

Algorithm 1: Mini batch SGD

Fix: n_{epochs} (number passes over data), K (number batches)

Minimize: $\frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$

For $epoch = 1, \ldots, n_{epochs}$:

Choose a random partition $\mathcal{P} = \{S_1, \ldots, S_K\}$ of $\{1, \ldots, n\}$

For $k = 1, \ldots, K$:

$$g = \frac{1}{|S_k|}\sum_{i \in S_k} \nabla f_i(\theta)$$

$$\theta = \theta - \eta g$$

Backprop Code

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

```
# start with the last layer: L = cross entropy; dL/d0_k = p_k - I(y=k)
delta = probs
delta[np.argmax(y, axis=0), range(n_examples)] -= 1
delta /= n_examples

dw[-1] = np.dot(delta, activations[-1].T)
db[-1] = np.sum(delta, axis=1, keepdims=True)

for l in xrange(2, self.n_layers):
    z = zs[-l + 1]
    g_prime = self.activation_prime(z)
    delta = np.dot(self.weights[-l + 1].T, delta) * g_prime

    dw[-l] = np.dot(delta, activations[-l].T)
    db[-l] = np.sum(delta, axis=1, keepdims=True)
```