

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

Backprop Code

Summary: the equations of backpropagation

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

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

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (\text{BP3})$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (\text{BP4})$$

```
# 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)
```