Machine Learning PS4: Neural Net

Simon Seo

Some important functions in this implementation

- $\mathbf{g}(\mathrm{x1}, \mathrm{x2})$ sigmoid function of dot product of x1 and x2
- $\mathbf{neuron}(\mathrm{a\_pre}, \mathrm{w\_intra}, \mathrm{i})$ calculate an activation from previous layer and weight matrix
- $\mathbf{forward\_propagate}(\mathrm{input\_layer}, *\ \mathrm{weight\_matrix\_args})$ forward propagation algorithm from input and weight(s)
- $\mathbf{classifier\_idx}(\mathrm{input\_layer}, *\ \mathrm{weight\_matrix\_args})$ classifies input data as {index of maximum value in output layer}
- $\mathbf{error}(\mathrm{data}, \mathrm{label}, \mathrm{classifier}, *\ \mathrm{weight\_matrix\_args})$ calculates error (decimal) for given data, label, classifier, and weights
- $\mathbf{\_cost\_MLE}(\mathrm{prediction}, \mathrm{label})$ maximum likelihood error: -(y*log(h) + (1-y)log(1-h))
- $\mathbf{cost}(\mathrm{output\_matrix}, \mathrm{label\_list}, \mathrm{cost\_model})$ calculates cost for given outputs, labels, and cost model

Error rate for the 5000 digits

The error calculated from the 5000 datasets was 2.48%

Loss function $\mathrm{J}(\Theta)$

$$J(\Theta) = -\frac{1}{m}\sum_{\mathrm{i}=1}^{\mathrm{m}}\left[y^{(i)}\log h_\Theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\Theta\left(x^{(i)}\right)\right)\right]$$

Using the above MLE with natural log, the cost J calculated from the 5000 datasets was 0.28762951217843125.

Back Propagation algorithm for calculating partial derivatives

The pseudocode for computing the gradient for a given Θ is as follows:

computeGradient(x, y, Θ):
$\Delta_{st}^{(l)} = 0$ for all l, s, t
for each training data $x^{(i)}$ and label $y^{(i)}$ $(i \in 1{\sim}m)$:
    $a = \text{forwardPropagate}\left(x^{(i)}, \Theta\right)$ //returns activations $a^{(l)}$ for all $l \in 1{\sim}L$
    $\delta = \text{backPropagate}\left(a, \Theta, y^{(i)}\right)$ //returns node error $\delta^{(l)}$ for all $l \in 2{\sim}L$
    for $l \in 1{\sim}L$:
        for $t \in 1{\sim}\, dim\left(a^{(l)}\right)$; $s \in 1{\sim}\, dim\left(a^{(l+1)}\right)$:
            $\Delta_{st}^{(l)} \mathrel{+}= a_t^{(l)} \delta_s^{(l+1)}$
            //each $\Delta_{st}^{(l)}$ is the gradient of the cost function $\left(\frac{\partial}{\partial \Theta_{st}^{(l)}} J(\Theta)\right)$
return Δ

For each training data, the $\Delta_{st}^{(l)}$ term is the gradient for each $\Theta_{st}^{(l)}$.

backPropagate(a, Θ, y):
$\delta_{st}^{(l)} = 0$ for all l, s, t
$\delta^{(L)} = a^{(L)} - y$
for $l \in (L-1){\sim}2$:
    $\delta^{(l)} = \left({\Theta^{(l)}}^T \delta^{(l+1)}\right) .* \left(a^{(l)}\right) .* \left([1] - a^{(l)}\right)$
    //$\delta_j^{(l)}$ is the error contributed by node j of layer l
return δ