Lecture07

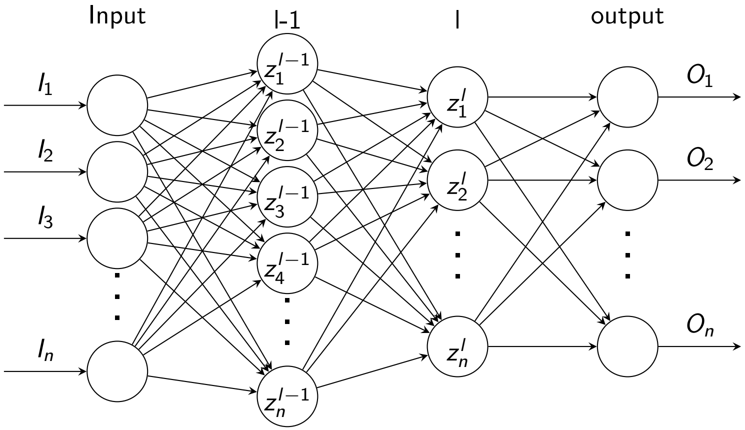
Dihui Lai

dlai@wustl.edu

March 17, 2020

Introduction to Neural Network

Neural Network: Topology



Neural Network: Forward

Each neuron at layer l recieves inputs from all neuron from the previous layer $l-1$

$$z_k^l = \sum_j w_{kj}^{l-1} a_j^{l-1}$$

The neuron tansfer the input signal z_k^l via a transfer function σ and send as input to to the next layer

$$a_k^l = \sigma(z_k^l)$$

The cost function of the neural network is dependeng on all the zs of neurons in all layers

$$C\left(z_1^l, z_2^l, ...z_k^l(z_1^{l-1}, z_2^{l-1}, z_3^{l-1}, ...), ...\right)$$

Neural Network: Activation Functions

Step Function: $\sigma(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{for } z \geq 0 \end{cases}$

Logistic/Sigmoid: $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

hyperbolic tangent: $\sigma(z) = \dfrac{(e^z - e^{-z})}{(e^z + e^{-z})}$

ReLU: $\sigma(z) = \begin{cases} 0 & \text{for } z \leq 0 \\ z & \text{for } z > 0 \end{cases}$

Reference `https://en.wikipedia.org/wiki/Activation_function`

Optimization: Stochastic Gradient Descent Method

1. Update the weights by changing it along the gradient to reduce the cost function
2. do one data point at a time

$$w_j \leftarrow w_j - \eta \frac{\partial C^i(w_j)}{\partial w_j}, i = 1, 2, 3, \ldots n$$

Gradient Descent Method for Linear Regression

Cost function $C(\beta) = \sum_i \frac{1}{2}(y^i - \vec{x}^i \cdot \vec{\beta})^2$

$$\frac{\partial C}{\partial \beta_j} = \sum_j (\hat{y}^i - y^i) x_j^i$$

The corresponding stochastic gradient methods is

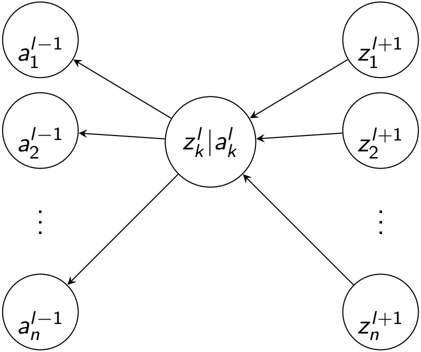
$$\beta_j \leftarrow \beta_j + \epsilon (y^i - \hat{y}^i) x_j^i$$

The update method is quite intuitive considering that β_j is adjusted higher if estimated $\hat{y}^i$ is less than y^i; adjusted lower if $\hat{y}^i$ is more than y^i

mini-Batch Gradient Descent

Between use the full data set or use 1 single data point to update w, one can choose to update w by calculating gradient using m data points or called a mini-batch.
Dividing the data set to k mini-batch so that $km = n$. Iterating through k mini-batches is called an epoch

Neural Network: Backpropogation



Neural Network: Backpropogation

The contribution to the cost function from a neuron in layer l can be cauculated iteratively as

$$\delta_k^l = \frac{\partial C}{\partial z_k^l} = \sum_m \frac{\partial C}{\partial z_m^{l+1}} \frac{\partial z_m^{l+1}}{\partial z_k^l}$$
$$= \left(\sum_m \frac{\partial C}{\partial z_m^{l+1}} \frac{\partial z_m^{l+1}}{\partial a_k^l} \right) \frac{\partial a_k^l}{\partial z_k^l}$$
$$= \sum_m \delta_m^{l+1} w_{mk}^l \sigma'(z_k^l)$$

The partial derivative of a cost function w.r.t the weight w_{kj}^{l-1} is

$$\frac{\partial C}{\partial w_{kj}^{l-1}} = \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial w_{kj}^{l-1}} = \delta_k^l a_j^{l-1}$$

Training Neural Network

1. Define the topology of your neural network: number of layers, number of units in each layer
2. initialize the weights w of the network
3. calculate the gradient of cost function by calculating δ_l^k against all neurons, backpropogate iteratively
4. updates weights of the network along with the gradient
   - batch gradient descent
   - mini-batch gradient descent
   - stochastic gradient descent

Multinomial distribution and multi-class classification

Multinomial distribution: there could be c outcome of an experiment, each of probability p_1, p_2, p_3, ..., p_c and $\sum_{j=1}^{c} p_j = 1$. If one perform N experiments, the probability of getting x_1, x_2, x_3, ... x_c of each out come can be described as

$$f(x_1, x_2, x_3, ...x_c) = \frac{N!}{x_1!x_2!...x_c!} p_1^{x_1} p_2^{x_2} ... p_c^{x_c}$$

The likelihood function can be accordingly written as $L = \prod_{i=1}^{n} \prod_{j=1}^{c} p_j^{x_j^i}$

The log-likelihood function (a.k.a log-loss) is

$$\ell = \log(L) = \sum_{i=1}^{n} \sum_{j=1}^{c} x_j^i \log(p_j)$$