Lecture Ten

March 31, 2016

1 Homework

1. Train a neral netowrk for the cancer data. Test the network.
2. Train a netowrk for the XOR-Porblem

The number of layers is up to the user.

1.1 Deep learning

Deep learning is a regular nerual netowrk with many layers in it.

1.2 Recursive Network

This is when a netowork will be resused on the output to improve the result. Deep learning is apart of the deep learning mechanism.

2 Neural Nets

$$\vec{o}^{(1)} = s(\vec{\bar{o}}^{(0)} \bar{W}_1) \tag{1}$$

$$\vec{o}^{(2)} = s(\vec{\bar{o}}^{(1)} \bar{W}_2) \tag{2}$$

$$E = \frac{1}{2} ||\vec{t} - \vec{o}^{(2)}||^2 \tag{3}$$

$$D_2 = \begin{bmatrix} 0_1^{(2)}(1-0_1^{(2)}) & 0 & \dots & 0 \\ 0 & 0_2^{(2)}(1-0_2^{(2)}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0_m^{(2)}(1-0_m^{(2)}) \end{bmatrix} \tag{4}$$

$$D_1 = \begin{bmatrix} 0_1^{(1)}(1-0_1^{(1)}) & 0 & \dots & 0 \\ 0 & 0_2^{(1)}(1-0_2^{(1)}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0_k^{(1)}(1-0_k^{(1)}) \end{bmatrix} \tag{5}$$

$$\vec{e} = \begin{bmatrix} t_1 - o_1^{(2)} \\ t_1 - o_2^{(2)} \\ \vdots \\ t_1 - o_m^{(2)} \end{bmatrix} \tag{6}$$

backprop error:

$$\vec{S^{(2)}} = D_2 \vec{e} \tag{7}$$

$$\vec{S}^{(1)} = D_1 W_2 \vec{S}^{(2)} \tag{8}$$

$$\nabla \bar{W}_2^T = -\gamma \vec{S}^{(2)} \hat{o}^{(1)} \tag{9}$$

$$\nabla \bar{W}_1^T = -\gamma \vec{S}^{(1)} \hat{o}^{(0)} \tag{10}$$

Error for all samples offline backprop:

$$\nabla \bar{W_2}^T = \nabla \bar{W_2}_1^T + \nabla \bar{W_2}_2^T + \dots \nabla \bar{W_2}_n^T \tag{11}$$

3 Speeding up the back prop

This method is only for the Batch back prop method. It is too risky for the online method.

$$\gamma_i^{k+1} = \begin{cases} \gamma_i^k u & \text{if } \nabla_i E^k \nabla_i E^{k-1} \geq 0 \\ \gamma_i^k d & \text{if } \nabla_i E^k \nabla_i E^{k-1} < 0 \end{cases} \tag{12}$$

Update step:

$$\nabla^k \omega_i = -\gamma_i^k \nabla_i E^k \tag{13}$$

This computation will hopefully accelerate the learning rate as to get you to the minimum faster. This is based on the idea of optimizing a quadratic function. u and d are free constants that can be played with.

4 RPROP

Only used for batch propagation.

$$\gamma_i^{k+1} = \begin{cases} min(\gamma_i^k u, \gamma_{max}) & \text{if } \nabla_i^k E \nabla_i^{k+1} E > 0 \\ max(\gamma_i^k d, \gamma_{min}) & \text{if } \nabla_i^k E \nabla_i^{k+1} E < 0 \\ \gamma_i^k & \text{otherwise} \end{cases} \tag{14}$$

Update Step:

$$\nabla^k \omega_i = -\gamma_i^k sgn(\nabla_i E^k) \tag{15}$$

This prevents the use of the value of the partial derivative so that you don't slow too much.