Neural Networks cheat-sheet

Andrea Jemmett

October 21, 2015

1 Perceptron

1.1 Perceptron Convergence Algorithm

1. Variables & parameters:
   $\mathbf{x}(n) = [1, x_1(n), \ldots, x_m(n)]^T$
   $\mathbf{w}(n) = [b, w_1(n), \ldots, w_m(n)]^T$
   $y(n) =$ net out
   $d(n) =$ target
   $\eta =$ learning rate
2. *Initialization* $\mathbf{w}(0) = \mathbf{0}$ then for $n = 1, 2, \ldots$ do the following
3. *Activation* Feed input $\mathbf{x}(n)$ to network
4. *Compute actual response* as $y(n) = (\mathbf{w}^T(n)\mathbf{x}(n))$
5. *Adaptation of weight vector* update weights using:
   $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$
   where:
   $$d(n) = \begin{cases} +1 & \text{if } x(n) \text{ belongs to class } C_1 \\ -1 & \text{if } x(n) \text{ belongs to class } C_2 \end{cases}$$

2 Statistic Based Methods

1. *Observation density / class-conditional / likelihood*
   $P(X = x|C_i) = \frac{\text{\# x samples}}{\text{\# of samples in } C_i}$
2. *Prior* $P(C_i) = \frac{\text{\# samples in } C_i}{\text{\# all samples}}$
3. *Posterior* $P(C_i|X = x) = \frac{\text{likelihood x prior}}{\text{evidence}}$
4. *Evidence* $P(X = x)$ is normalization / scaling factor

Maximum A Posteriori estimate $\mathbf{w_{MAP}} = \text{argmax}_{\mathbf{w}}\, \pi(\mathbf{w}|d, \mathbf{x})$

3 Linear Models

Gradient Descent Algorithm

1. Start from arbitrary point $\mathbf{w}(0)$
2. find a direction by means of a gradient: $\nabla\xi = [\frac{\partial f}{\partial w_1}, \dots, \frac{\partial f}{\partial w_n}]$
3. make a small step in that direction: $\Delta\mathbf{w} = -\eta\nabla\xi$
4. repeat the whole process

ADALINE uses an identity activation function and update rule is

$$\Delta\mathbf{w} = +\eta\mathbf{x}(d - y)$$

4 Multi-Layer Perceptrons

Generalized Backprop delta rule

$$\Delta w_{ji} = \eta\delta_j y_i$$

$$\delta_j = \begin{cases} \varphi'(v_j)(d - y_j) & \text{if } j \text{ is output node} \\ \varphi'(v_j)\sum_k \delta_k w_{kj} & \text{if } j \text{ is hidden node} \end{cases}$$

5 Self-Organizing Maps

Three processes:

1. Competition : find the winning neuron: $i(\mathbf{x}) = \text{argmin}_j \|\mathbf{x} - \mathbf{w_j}\|$

2. Cooperation : determine neighbourhood function: $h_{j,i} = \exp(-\frac{d_{j,i}^2}{2\sigma^2})$

3. Adaptation : adapt weights with:
$\mathbf{w_j}(n+1) = \mathbf{w_j}(n) + \eta(n)h_{j,i}(n)(\mathbf{x}(n) - \mathbf{w_j}(n))$