MACHINE LEARNING
BARUCH COLLEGE
SPRING 2015

Homework Assignment 5 Solutions

1. Putting $\epsilon = 0.001$ and $\delta = 0.01$ in the expression for the Chernoff Bound, we get a sample size of $m_S \geq 2{,}649{,}200$ records. The number of partitions is: $N \simeq 100{,}000{,}000/2{,}650{,}000 = 38$. Given a sample size of $m_S = 2{,}650{,}000$, the new support threshold would need to be changed by the one-sided Chernoff Bound correction of $\sqrt{\frac{1}{m_S} ln\left(\frac{1}{\delta}\right)} = 0.000932$, which is insignificant compared to s and can be safely ignored.

2. An RBFNN with a 2-dimensional input has good coverage and would train faster, so it would be preferable. Yes, the answer would change because of the poor coverage. We should consider the Linear Separability because this is what affects coverage most.

3. The Linear Separability Index can be approximated by the minimum number of hidden nodes in an Auto-Associative network that would reproduce the input reasonably well. We can use this estimate to determine the dimensions and required coverage of the RBF.

4. Since the Exclusion Zone is defined by $|\boldsymbol{w} \cdot \boldsymbol{x} + b| \leq R$, with $R = 1/\left(2\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}\right)$, we can see that the Exclusion Zone Radius is maximized whenever the square of the weights (the weight "energy") is minimized (with the exception of the excluded trivial case of zero weights). We can simply add a term in the error function that trades off a penalty for the square of the weight norm vs. the misclassification error (as we saw earlier in the context of regularization).

5. (a) Assuming that both n and n_B are sufficiently large and that the errors ε_i and ε_i^B are uncorrelated with each other and with all other variables, we can see that $\sigma_{Y^B}^2 = Var\left(\hat{Y}_i + \varepsilon_i^B\right) = \sigma_Y^2(1 + n_B^2)$, so that $r = \lambda\sqrt{(1 + n_B^2)}\sigma_Y$. Likewise, $Var(Y_i^*) = Var\left(\hat{Y}_i^B + \varepsilon_i\right) = \sigma_Y^2(1 + n_B^2 + n^2)$. We can now write:

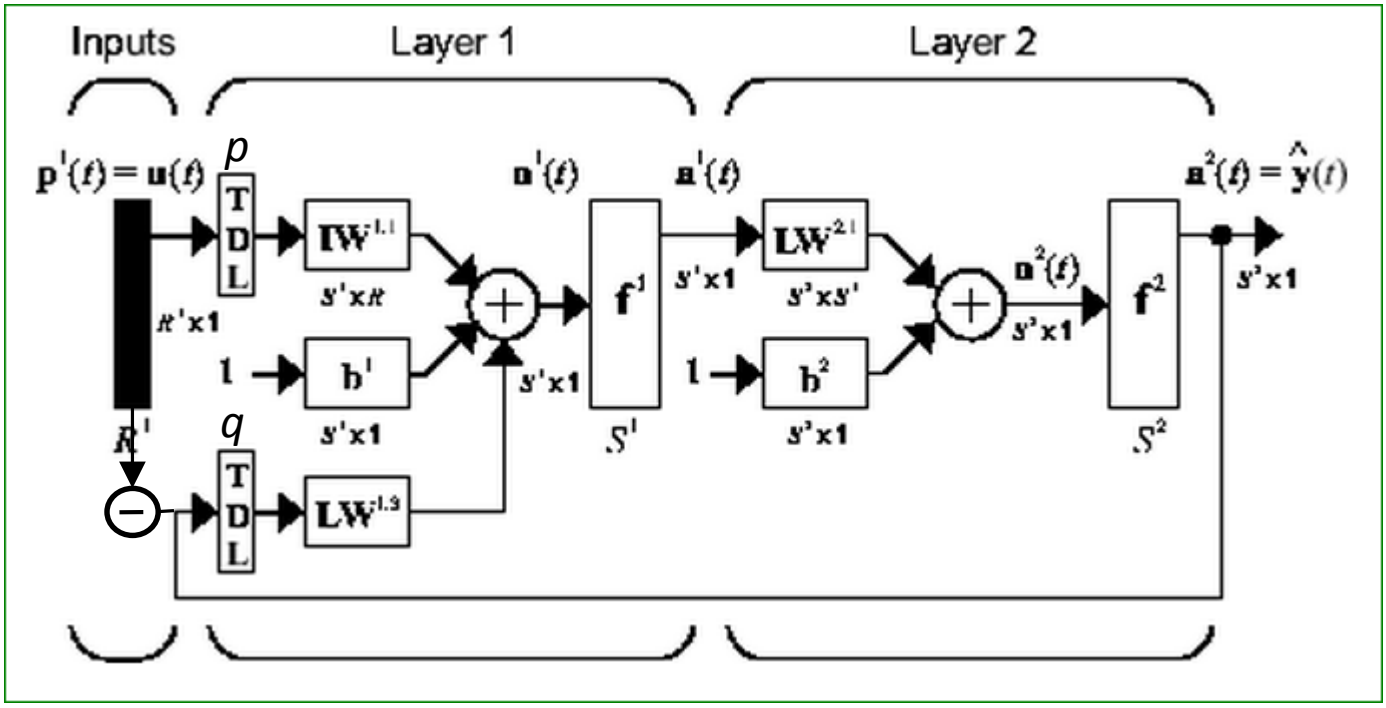
$$SR = \frac{\lambda\sqrt{(1 + n_B^2)}\sigma_Y}{\sqrt{(1 + n_B^2 + n^2)}\sigma_Y} = \lambda\sqrt{\frac{1 + n_B^2}{1 + n_B^2 + n^2}} \cong \lambda\sqrt{\frac{1 + n_B^2}{1 + n^2}} \cong \frac{\left(1 + \frac{1}{2}n_B\right)}{\sqrt{1 + n^2}}\lambda\,.$$

(b) We have:

$$maxR^2 = 1 - \frac{Var(\varepsilon_i)}{Var(Y_i^*)} = 1 - \frac{n^2}{1 + n_B^2 + n^2} \cong \frac{1 + n_B^2}{1 + n^2}.$$

Comparing this with the $maxR^2$ found in HW#2, $maxR^2 = 1 - \frac{n^2}{1+n^2} = \frac{1}{1+n^2}$, we see that the bootstrapping procedure has increased the $maxR^2$ by $\frac{n_B^2}{1+n^2}$. Comparing the unadjusted *SR* found in (a) above with what we found in HW#3, $SR = \frac{1}{\sqrt{1+n^2}}\lambda$, we see that we have increased the SR by $\frac{\frac{1}{2}n_B\lambda}{\sqrt{1+n^2}}$.

6. See diagram below.



The input $\mathbf{p^1}(t)$ is just the series target itself, $\mathbf{y}(t)$, but lagged *p* time steps by the input TDL labeled with a *p*, before going into the hidden layer via a set of weights $\mathbf{LW}^{1,1}$ (this is the *AR* term). The prediction $\hat{\mathbf{y}}(t)$ is subsequently fed back through the recursion loop, but it is combined *by subtraction* with the input (denoted by the circle with a minus sign inside), and both target and prediction are lagged by *q* time steps by the TDL labeled with a *q* before going into the hidden layer with its own set of weights $\mathbf{LW}^{1,2}$ (this is the *MA* term). To implement an $ARMA(p, q)$, both $\mathbf{f_1}$ and $\mathbf{f_2}$ are linear. Note that an output will be generated after the smaller of *p* or *q* delays. Bootstrapping is problematic in time series prediction to the extent that random sampling does not respect the ordering of the time series. To alleviate this problem, the time series can be split into contiguous sub-segments of the overall series, and models trained on those sub-segments.