Homework #5

TA in charge: Hanhsing Tu, Room 536

RELEASE DATE: 11/27/2008

DUE DATE: 12/04/2008, 4:00 pm IN CLASS

TA SESSION: 12/03/2008, noon to 2:00 pm IN R106

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

5.1 Powerfulness of Neural Networks

In this problem, we will work with Neural Networks using sign$(\cdot)$ instead of tanh$(\cdot)$ as the activation functions. In addition, we will take $+1$ to mean logic TRUE, and -1 to mean logic FALSE. Then, we will show that Neural Networks can implement any Boolean function.

(1) (5%) Assume that we want to compute

$$f_A(x) = AND\left((x)_1, (x)_2, \ldots, (x)_d\right).$$

Show a perceptron

$$g_A(x) = \text{sign}\left(\sum_{i=1}^{d} w_i (x)_i - \theta\right).$$

that implements $f_A(x)$.
(*Note: By* ***SHOW****, you should write down your choice of* $(\theta, w_1, w_2, \cdots, w_d)$ *and prove that* $g_A(x) = f_A(x)$ *for all* $x \in \{-1, +1\}^d$)

(2) (5%) Similar to (1), show a perceptron $g_O(x)$ that implements

$$f_O(x) = OR\left((x)_1, (x)_2, \ldots, (x)_d\right).$$

(3) (5%) Similar to (1), show a perceptron $g_N(x)$ that implements

$$f_N(x) = NOT\left((x)_1\right).$$

(4) (10%) Show a neural network that implements

$$AND\Big((x)_1, NOT\big((x)_2\big), OR\Big((x)_3, (x)_4, NOT\big((x)_1\big)\Big)\Big).$$

(5) (5%) Argue how you can systematically implement any Boolean function with a neural network.

(6) (10%) Show a 3-4-1 neural network that implements $XOR\big((x)_1, (x)_2, (x)_3\big)$.

(7) (10%) Show a 3-3-1 neural network that implements $XOR\big((x)_1, (x)_2, (x)_3\big)$.

(8) (Bonus 10%) Show a d-d-1 neural network that implements $XOR\big((x)_1, (x)_2, (x)_3, \ldots, (x)_d\big)$.

(*Note: No partial credits will be given to the bonus question! You need to give a complete and convincing proof to get the bonus.*)

5.2 Limitations of Neural Networks

(1) (10%) Prove that $(N^D + 1)^3 \leq N^{3D+1} + 1$ for $D \geq 1$ and $N \geq 3$.

(2) (10%) Prove that if $N \geq 3\Delta \log_2 \Delta$, $N^\Delta + 1 < 2^N$.

(3) (20%) Consider a learning model G_{3A} that consists of all d-3-1 neural networks with sign$(\cdot)$ as all the activation functions, and with $w = (+1, +1, +1)$, $\theta = 2.5$ for the output neuron ***only***. Use the facts above (or not) to prove that

$$VCD(G_{3A}) < 3 \cdot \big(3(d+1)+1\big) \log_2\big(3(d+1)+1\big).$$

(4) (10%) In Problem 5.1 we proved that Neural Networks can implement any Boolean function. In this problem (including the bonus one below) we proved that $VCD(G_M)$ is bounded. Are those facts contradicting each other? Explain your opinions.

(5) (Bonus 10%) Consider a learning model G_M that consists of all d-M-1 neural networks with sign$(\cdot)$ as the activation functions. Prove that

$$VCD(G_M) = O\big(M \cdot d \cdot \log(M \cdot d)\big).$$

(*Note: No partial credits will be given to the bonus question! You need to give a complete and convincing proof to get the bonus.*)

5.3 Experiment with Weight Decay in Neural Network (*)

Modify your backpropagation algorithm for d-M-1 neural network from Problem 4.4 by changing the activation function of the output neuron ***only*** into a linear function

$$x^{\text{out}} = \left(\sum_{i=0}^{d^{\text{in}}} w_i \cdot x_i^{\text{in}} \right).$$

In other neurons, keep using

$$x^{\text{out}} = \tanh \left(\sum_{i=0}^{d^{\text{in}}} w_i \cdot x_i^{\text{in}} \right).$$

Implement the weight decay algorithm with stochastic gradient descent on this neural network. That is, update $\mathbf{w}$ by

$$\mathbf{w}^{(t)} \longleftarrow (1 - \eta\lambda) \cdot \mathbf{w}^{(t-1)} - \eta \cdot \nabla E_n\big(\mathbf{w}^{(t-1)}\big).$$

Run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml08fall/data/hw5_3_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml08fall/data/hw5_3_test.dat`

Take $M = 8$, $\eta = 0.05$, and $T = 5000$. Initialize the elements of $\mathbf{w}^{(0)}$ randomly from the range $(-0.02, 0.02)$. Now, consider $\lambda = 0$ (no weight decay), 0.001, 0.01, 0.1, or 1.

(1) (50%) Let

$$g^{(t)}(x) \quad = \quad \text{sign}\Big(\text{NN}\big(x_n, \mathbf{w}^{(t)}\big)\Big).$$

For the five λ values above, plot $\nu\big(g^{(t)}\big)$, and $\hat{\pi}\big(g^{(t)}\big)$ as functions of t on the same figure. Compare the curves across different λ, and briefly state your findings.

(*Note: You should have five separate plots with scales, legends, labels, and captions clearly marked. You need to implement your own weight decay algorithm instead of using sophisticated packages*)

5.4 Experiment with Naïve Bayes (*)

(1) (50%) Implement the Naïve Bayes algorithm below.

(a) estimate $P(+)$ and $P(-)$ from Z by $P(+) = N_+/N$, and $P(-) = N_-/N$.

(b) discretize the region $[0, 1)$ uniformly into K bins $\Big\{[L_k, R_k]\Big\}_{k=1}^{K}$.

(c) for each combination of $s \in \{+, -\}$, $i \in \{1, 2, \cdots, d\}$, and $k \in \{1, 2, \cdots, K\}$

- Estimate $P\Big((x)_i \,|s\Big)$ by

$$\begin{aligned} P\Big((x)_i \,|s\Big) &= I\Big[(x)_i \in [L_k, R_k)\Big] P_{ik}^{(s)}, \\ P_{ik}^{(s)} &= \frac{1}{N_s} \cdot \sum_{n=1}^{N} I\Big[(x_n)_i \in [L_k, R_k) \text{ and } y_n = s\Big]. \end{aligned}$$

(d) Return

$$\begin{aligned} g(x) &= \underset{s \in \{+,-\}}{\text{argmax}}\, P(s \,|x), \\ \text{using } P(x \,|s) &= \prod_{i=1}^{d} P\Big((x)_i \,|s\Big) \text{ and the Bayes theorem.} \end{aligned}$$

Run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml08fall/data/hw5_4_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml08fall/data/hw5_4_test.dat`

Assume that g_K is the decision function returned when using parameter K. For $K = 2, 5, 10, 20, 50$, report $\nu(g_K)$ and $\hat{\pi}(g_K)$. Briefly state your findings.