CPSC 636 Midterm Exam (3/4/2008, Tuesday)[1]

Last name: _______________, First name: _______________

Time: **12:45pm–2:00pm (65 minutes)**

Subject	Score
Neural networks in general	/20
Learning in general	/20
Single layer perceptron	/20
Multilayer perceptron	/30
RBF networks	/40
Total	min(100, /130)

- Be as **succinct** as possible.
- Read the questions carefully to see what kind of answer is expected (*explain blah* **in terms of ...** *blah*).
- Some problems have multiple subproblems, indicated by (1), (2), etc. Solve all subproblems for full credit.
- Your final score will be $\min(\text{raw score}, 100)$. There can be various strategies:
  - Solve all problems (to a varying degree of completeness) hoping that for some you'll get partial credit and they all add up to be over 100 points.
  - Solve a subset of problems that adds up to 100 points exactly, and hope you get perfect scores on all those problems.
- This is a closed-book exam. You may use one sheet of note (US letter).

Double-sided print: Total of 8 pages including cover sheet.

[1] Instructor: Yoonsuck Choe.

1 Neural networks in general

Question 1 (20 pts): It is often required to measure the *similarity* between vectors (points in high dimensional space). Two common measures are the Euclidean distance and the dot product (inner product). The two are related in interesting ways.

Given two unit vectors $\mathbf{x}$ and $\mathbf{y}$ (i.e., $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$), (1) express the Euclidean distance $d(\mathbf{x}, \mathbf{y})$ (or $d^2(\mathbf{x}, \mathbf{y})$) between $\mathbf{x}$ and $\mathbf{y}$ in terms of the dot product $\mathbf{x}^T\mathbf{y}$, and (2) discuss their relationship (e.g., the smaller/larger the $d(\mathbf{x}, \mathbf{y})$, the smaller/larger the dot product, etc.).

Hint: $\mathbf{x}^T\mathbf{x} = \sum_{i=1}^{m} x_i^2$, where $\mathbf{x} = [x_1, x_2, ..., x_m]^T$.

2 Learning in general

Question 2 (20 pts): Correlation matrix memory is defined as:

$$\hat{\mathbf{M}} = \sum_{j=1}^{n} \mathbf{y}_j \mathbf{x}_j^T$$

for n input–output vector pairs $(\mathbf{x}_k, \mathbf{y}_k)$ for $k = 1, 2, ..., n$. (The vectors are all column vectors.)

Explain why the memory matrix gives perfect mapping from $\mathbf{x}_k$ to $\mathbf{y}_k$ when all the input vectors are *mutually orthogonal* ($\mathbf{x}_k^T \mathbf{x}_j = 0$ for $k \neq j$) and have *unit length* ($\|\mathbf{x}_k\| = 1$ for all k).

That is, prove that, for all k

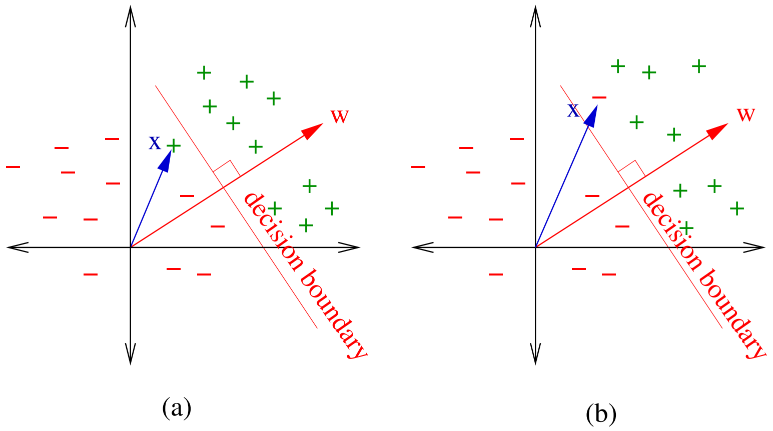
$$\mathbf{y}_k = \hat{\mathbf{M}} \mathbf{x}_k.$$

Hint: Start with the formula below and show that it reduces to $\mathbf{y}_k$.

$$\hat{\mathbf{M}} \mathbf{x}_k = \left(\sum_{j=1}^{n} \mathbf{y}_j \mathbf{x}_j^T \right) \mathbf{x}_k.$$

3 Single-layer perceptrons

Question 3 (20 pts): In perceptron learning, no learning occurs when the given input is correctly classified by the network with the current set of connection weights. Learning takes a slightly different form depending on the type of error made: (a) positive input falsely classified as negative, or (b) negative input falsely classified as positive. (See figure below.)



(1) Based on your intuition on how the weight vector **w** needs to be tilted, write down the perceptron learning rule, below: Fill in the blank with a proper sign (+ or -).

- (a) Positive input **x** is falsely classified as negative:

$$\mathbf{w}(n+1) = \mathbf{w}(n)\_\_\_\_\_\_\_\eta\mathbf{x}(n).$$

- (b) Negative input **x** is falsely classified as positive:

$$\mathbf{w}(n+1) = \mathbf{w}(n)\_\_\_\_\_\_\_\eta\mathbf{x}(n).$$

Note: η is a small constant learning rate factor.

(2) Explain why while referencing the figure above.

4 Multi-layer perceptrons

Question 4 (20 pts): Explain how multi-layer perceptrons can effectively handle classification tasks that are *not* linearly separable. Explain in terms of the role of the hidden units.

Question 5 (10 pts): What is the merit of the conjugate gradient method compared to gradient descent (backpropagation), *besides* it potentially being more accurate and less oscillatory/zig-zaggy?

5 RBF networks

Question 6 (20 pts): In instance-based learning (such as nearest neighbor [NN] or k nearest neighbor [KNN]), *all input–target pairs* $(\mathbf{x}_i, d_i)$ *are stored in memory*, and whenever a new input is presented, the predicted target value is calculated by referencing directly the stored information. Radial-basis function networks (but *not* the generalized RBF) are similar to NN or KNN in some sense. Explain why they are similar.

Question 7 (20 pts): Explain what are the main differences between regularization networks (radial basis function networks) and *generalized* radial basis function networks, in terms of (1) number of hidden units relative to the number of inputs, (2) hidden unit centers, and (3) hidden-to-output weight learning.