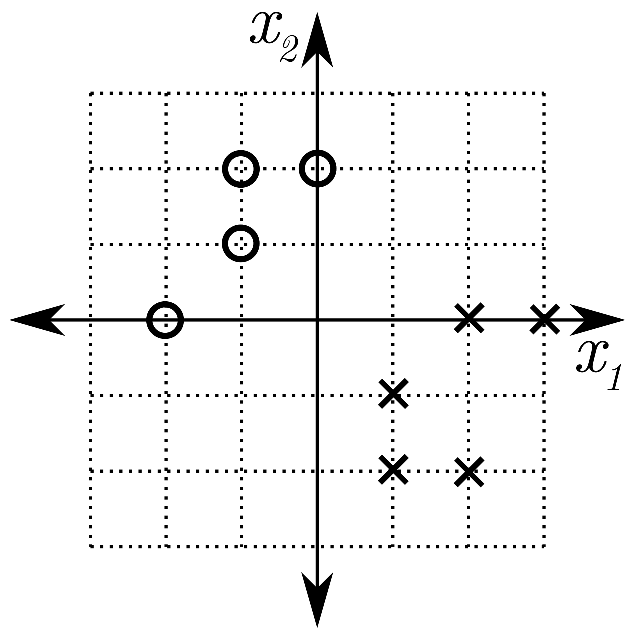
ECE/CS 559 - Fall 2016 - Final Exam.

Full Name: **ID Number:**

- **Q0 (5 pts):** Attach the e-mail/webpage confirming that you completed the instructor/TA evaluations.

- **Q1 (30 pts):** This problem will be on SVMs. **Warning: Neither of the two parts of this question require solving complicated optimization problems**.

  (a) **(15 pts):** Consider the figure below. Each small dotted square is 1×1. Members of C^+ are represented by crosses while members of C^- are represented by hollow disks. According to this description, we have $C^+ = \{\left[\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 3 \\ 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 \\ -2 \end{smallmatrix}\right], \left[\begin{smallmatrix} 2 \\ -2 \end{smallmatrix}\right]\}$, and $C^- = \{\left[\begin{smallmatrix} -2 \\ 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} -1 \\ 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} -1 \\ 2 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 \\ 2 \end{smallmatrix}\right]\}$.



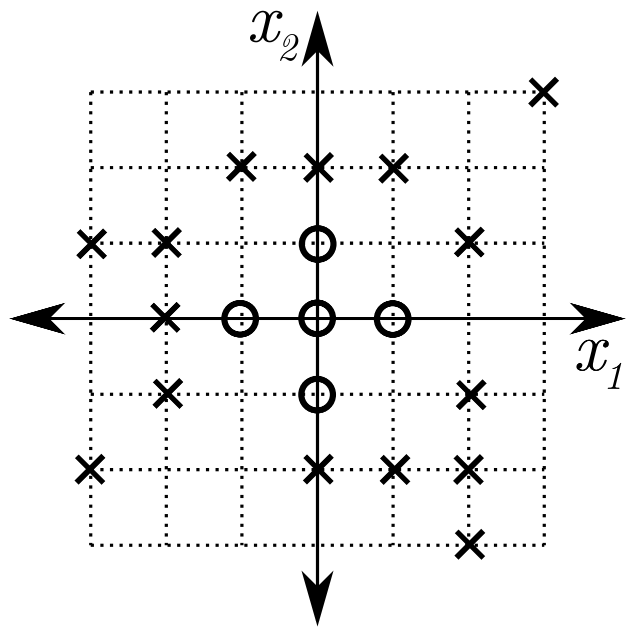
  Design a **linear SVM** that separates the classes C^+ and C^-.

  Recall that the result of the SVM will be a discriminant function $g(\mathbf{x}) = g(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right])$ with the property that $g(\mathbf{x}) \geq 1$ for every $\mathbf{x} \in C^+$, and $g(\mathbf{x}) \leq -1$ for every $\mathbf{x} \in \mathcal{C}^-$. A pattern $\mathbf{x} \in C^+$ with $g(\mathbf{x}) = 1$ is called a support vector for class C^+, while a pattern $\mathbf{x} \in C^-$ and $g(\mathbf{x}) = -1$ is called a support vector for class C^-. In your solution, you should clearly indicate

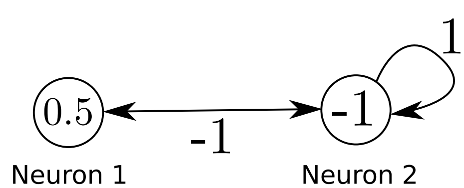
  - The steps and justifications of your solution. In particular, you should formally prove why the hyperplane separator of your SVM is the best.
  - The resulting discriminant function $g(\mathbf{x})$.
  - The support vectors for class C^+, and the support vectors for class C^-.
  - A sketch of the decision boundaries

  $$\mathcal{H} = \{\mathbf{x} : g(\mathbf{x}) = 0\},\ \mathcal{H}^+ = \{\mathbf{x} : g(\mathbf{x}) = 1\},\ \text{and } \mathcal{H}^- = \{\mathbf{x} : g(\mathbf{x}) = -1\}.$$

(b) **(15 pts):** Repeat (a) for classes C^+ and C^- illustrated in the figure below. This time, instead of a linear SVM, you will have to design **a non-linear SVM** by picking an appropriate feature mapping/kernel. **Your feature mapping $\phi(\mathbf{x})$ should depend only on the norm $\|\mathbf{x}\|$ of $\mathbf{x}$.** For example, $\phi(\mathbf{x}) = \begin{bmatrix} \|\mathbf{x}\|+1 \\ \|\mathbf{x}\|^2 \end{bmatrix}$ would be a valid feature mapping for this question.



- **Q2 (31 pts):** Consider the Hopfield network below. The activation function is $\phi(x) = 1$ if $x \geq 0$, and $\phi(x) = -1$ if $x < 0$.



(a) **(9 pts):** Draw the state transition diagram together with state energy levels for the asynchronous update rule. Indicate the steady state(s) of the network.

(b) **(7 pts):** Does the network always converge to a steady state under the asynchronous update rule? Justify your answer.

(c) **(15 pts):** Repeat (a) and (b) for the synchronous update rule.

- **Q3 (34 pts):** Let $\mathbf{x}_1 = \begin{bmatrix}1 & 1\end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix}2 & 1\end{bmatrix}$, $\mathbf{x}_3 = \begin{bmatrix}-1 & -1\end{bmatrix}$. We wish to design a one-dimensional SOM with 2 neurons. Let $\mathbf{w}_{1,0} = \begin{bmatrix}-1 & 0\end{bmatrix}$ and $\mathbf{w}_{2,0} = \begin{bmatrix}0 & 1\end{bmatrix}$ be the initial weights of the first neuron and the second neuron, respectively.

  We recall the usual online learning procedure. The patterns are shown sequentially as

  $$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$$

  resulting in the sequence of weights

  $$\mathbf{w}_{1,0} = [-1 \quad 0], \mathbf{w}_{1,1}, \mathbf{w}_{1,2}, \mathbf{w}_{1,3}, \mathbf{w}_{1,4}, \mathbf{w}_{1,5}, \ldots$$

  for the first neuron, and the sequence of weights

  $$\mathbf{w}_{2,0} = [0 \quad 1], \mathbf{w}_{2,1}, \mathbf{w}_{2,2}, \mathbf{w}_{2,3}, \mathbf{w}_{2,4}, \mathbf{w}_{2,5}, \ldots$$

  for the second neuron. Thus, for $n \in \{1, 2, \ldots\}$, the vectors $\mathbf{w}_{1,n}$ and $\mathbf{w}_{2,n}$ denote the updated weights after n patterns are shown to the network. For notational convenience, we let $\mathbf{x}_n$ denote the nth pattern shown to the network. For example, $\mathbf{x}_n = \mathbf{x}_3$ whenever n is a multiple of 3.

  When the nth pattern $\mathbf{x}_n$ is shown, we define the winning neuron

  $$i_n = \arg \min_{i \in \{1,2\}} \|\mathbf{x}_n - \mathbf{w}_{i,n-1}\|, \; n \in \{1, 2, \ldots\}$$

  as the neuron whose weight vector is closest to $\mathbf{x}_n$ in terms of the Euclidean distance.

  (a) **(8 pts):** For any real number x, let $\lfloor x \rfloor$ denote the "floor function," i.e. the largest integer that is less than or equal to x. For example, $\lfloor 2.1 \rfloor = 2$, $\lfloor 3 \rfloor = 3$, $\lfloor -1.1 \rfloor = -2$, $\lfloor -5 \rfloor = -5$. For a vector $\mathbf{x} = \begin{bmatrix}x_1 & x_2\end{bmatrix}$, we let $\lfloor \mathbf{x} \rfloor = \begin{bmatrix}\lfloor x_1 \rfloor & \lfloor x_2 \rfloor\end{bmatrix}$. Given $n \in \{1, 2, \ldots\}$, when the nth pattern $\mathbf{x}_n$ is shown, suppose that we update the winning neuron as

  $$\mathbf{w}_{i_n,n} = \left\lfloor \mathbf{w}_{i_n,n-1} + \tfrac{1}{2}(\mathbf{x}_n - \mathbf{w}_{i_n,n-1}) \right\rfloor,$$

  while keeping the loser neuron weights the same (i.e. $\mathbf{w}_{i,n} = \mathbf{w}_{i,n-1}$ if $i \neq i_n$). Find the weights of both neurons after one epoch of training.

  (b) **(8 pts):** Do the limits $\lim_{n\to\infty} \mathbf{w}_{1,n}$ and $\lim_{n\to\infty} \mathbf{w}_{2,n}$ exist? In other words, do the weight vectors converge? Justify your answer.

  (c) **(8 pts):** Given $n \in \{1, 2, \ldots\}$, when $\mathbf{x}_n$ is shown, suppose that we instead update the winning neuron weights as

  $$\mathbf{w}_{i_n,n} = \mathbf{w}_{i_n,n-1} + \tfrac{1}{2}(\mathbf{x}_n - \mathbf{w}_{i_n,n-1}),$$

  while again keeping the loser neuron weights the same (i.e. $\mathbf{w}_{i,n} = \mathbf{w}_{i,n-1}$ if $i \neq i_n$). Find the weights of both neurons after one epoch of training.

  (d) **(10 pts):** Repeat (b) for the update rule in (c).