Instructions There are 4 questions on this assignment. The third question involves coding. Do not attach your code to the writeup. Instead, copy your implementation to

/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW3

To write in this directory, you need a kerberos instance for andrew, or you can log into, for example, unix.andrew.cmu.edu. Please submit each problem separately with your name and userid on each problem. Refer to the webpage for policies regarding collaboration, due dates, and extensions.

1 α-Networks and Memory [Babis, 15 points]

Consider the phrase “Yes, we can”. Probably for many of you, it immediately brings to your thought American elections, politics etc. This is a remarkable fact since by a small phrase we can retrieve many different memories. Another example is the phrase “To be or not to be”. It brings to our minds Shakespeare, literature, England etc. In this exercise we will first define a simple model of memory and investigate its properties, step by step.

Our memory model which we are going to call α-network is a weighted, undirected graph $G(V,E,w)$ where $w : E \rightarrow \mathbb{R}$. The edges of our graph are allowed to have negative weights. Each node $v \in V$ will have two states. We model this by assuming that each node can have value either 1 or -1. You can think of each node of the graph as a neuron which can be either “on” or “off”. We will call a configuration $S$ of our network an assignment of $\pm 1$ to our nodes. However, we demand that $S$ respects the following rule:

Rule 1: if an edge $(u,v)$ has negative weight, then $u$ and $v$ must have the same state, and if an edge $(u,v)$ has positive weight, then $u$ and $v$ must have the opposite state.

A natural first question to ask is therefore if given a graph $G(V,E,w)$ we can always find a configuration $S$ such that it respects the above rule.

[3 points] (1) Prove by giving a counterexample that the answer to the above question is negative. In other words, give a network for which you can find no $S$ which respects the aforementioned rule.

Now, let’s define a different concept which will have the property in constrast to the above rule to be satisfiable for any graph $G(V,E,w)$. Consider a node $v$ with state $s(v)$. $v$ has a set of neighbors $v_1, \ldots, v_k$, where $k \geq 1$ and their corresponding states are $s(v_1), \ldots, s(v_k)$. We will say that node $v$ is good if the following condition holds for $v$:

$$\sum_{i=1}^{k} w((v,v_i))s(v)s(v_i) \leq 0 \quad (1)$$

One way to think about equation 1 is the following: each node (neuron) wonders “what should my state be, +1 or -1 in order to satisfy as much as I can for Rule 1?”. We claim the following: Equation 1 provides a rational strategy for each node to decide its state.

It cannot be in two different states at the same time.
[2 points] (2) Justify why the above claim is true, by thinking of each node separately. What happens if it does not behave as described above?

We call a configuration \( S \) for which every node is good \( \alpha \)-configuration. Can we always find an \( \alpha \)-configuration for a network? And if yes, how? Consider the following simple algorithm:

\[
\text{While the current configuration is not an alpha-configuration}
\]
\[
\text{Choose a node } v \text{ which is not good}
\]
\[
\text{Flip its state}
\]
Endwhile

[3 points] (3) Run the above algorithm on the network shown in figure 1.

If you answered correctly question 3, then you should have seen that the algorithm terminates. It is clear that if the algorithm terminates, it outputs an \( \alpha \)-configuration. However it is not clear that the algorithm terminates.

[7 points] (4) Prove that the algorithm terminates.
Hint: Find a quantity which strictly increases at every step of the algorithm and has an absolute upper bound.

\[2 \text{ points} \quad (2) \text{ Justify why the above claim is true, by thinking of each node separately. What happens if it does not behave as described above?}\]

2. [20 points] Feature Maps, Kernels, and SVM (Kate)

You are given a data set \( D \) in Figure 2 with data from a single feature \( X_1 \) in \( \mathbb{R}^1 \) and corresponding label \( Y \in \{+, -\} \). The data set contains four positive examples at \( X_1 = \{-3, -2, 3\} \) and three negative examples at \( X_1 = \{-1, 0, 1\} \).

\[
\text{Figure 2: Dataset for SVM feature map task in Question 2}\]

2.1 Finite Features and SVMs

1. [1pt] Can this data set (in its current feature space) be perfectly separated using a linear separator? Why or why not?

2. [1pt] Let define the simple feature map \( \phi(u) = (u, u^2) \) which transforms points in \( \mathbb{R}^1 \) to points in \( \mathbb{R}^2 \). Apply \( \phi \) to the data and plot the points in the new \( \mathbb{R}^2 \) feature space.
3. [1pt] Can a linear separator perfectly separate the points in the new \( \mathbb{R}^2 \) features space induced by \( \phi \)? Why or why not?

4. [1pt] Give the analytic form of the kernel that corresponds to the feature map \( \phi \) in terms of only \( X_1 \) and \( X'_1 \). Specifically define \( k(X_1, X'_1) \).

5. [3pt] Construct a maximum-margin separating hyperplane. This hyperplane will be a line in \( \mathbb{R}^2 \), which can be parameterized by its normal equation, i.e. \( w_1 Y_1 + w_2 Y_2 + c = 0 \) for appropriate choices of \( w_1, w_2, c \). Here, \((Y_1, Y_2) = \phi(X_1)\) is the result of applying the feature map \( \phi \) to the original feature \( X_1 \). Give the values for \( w_1, w_2, c \). Also, explicitly compute the margin for your hyperplane. You do not need to solve a quadratic program to find the maximum margin hyperplane. Note that the line must pass somewhere between \((-2,4)\) and \((-1,1)\) (why?), and that the hyperplane must be perpendicular to the line connecting these two points. Use only two support vectors.

6. [1pt] On the plot of the transformed points (from part 3), plot the separating hyperplane and the margin, and circle the support vectors.

7. [1pt] Draw the decision boundary separating of the separating hyperplane, in the original \( \mathbb{R}^3 \) feature space.

8. [3pt] Compute the coefficients \( \alpha \) and the constant \( b \) in Equation 2 for the kernel \( k \) and the support vectors \( SV = \{u_1, u_2\} \) you chose in part 6. Be sure to explain how you obtained these coefficients.

\[
y(x) = \text{sign} \left( \sum_{n=1}^{\vert SV \vert} \alpha_n y_n k(x, u_n) + b \right)
\]  

Equation 2

Think about the dual form of the quadratic program and the constraints placed on the \( \alpha \) values.

9. [1pt] If we add another positive (\( Y = + \)) point to the training set at \( X_1 = 5 \) would the hyperplane or margin change? Why or why not?

10. [3pts] Three different support vector machines have been trained on a 2D data set using:

(a) a linear kernel, \( k(x, y) = x^T y \) (Figure 3a)

(b) a quadratic polynomial kernel, \( k(x, y) = (x^T y + 1)^2 \) (Figure 3b)

(c) an RBF kernel, \( k(x, y) = \exp\left(-\frac{1}{2\sigma^2}(\|x - y\|^2)\right) \) (Figure 3c)

Figure 3: Three different kernels are trained on a 2D data set used in problem 2.1.10

Assume we now translate the data by adding a large constant value (i.e. 10) to the vertical coordinate of each of the data points, i.e. a point \((x,y)\) becomes \((x,y+10)\).

If we retrain the above SVMs on this new data, do the resulting SVM boundary change relative to the data points? Say if it does change or not for case (a), case (b) and case (c). Briefly explain why or why not it changes for all 3 cases (a), (b) and (c) and suggest or draw what happens to the resulting new boundaries when appropriate.
2.2 Infinite Features Spaces and Kernel Magic

Let's define a new (infinitely) more complicated feature transformation $\phi_n : \mathbb{R}^1 \to \mathbb{R}^n$ given in Equation 3.

$$\phi_n(x) = \left\{ e^{-x^2/2}, e^{-x^2/2}x, \frac{e^{-x^2/2}x^2}{\sqrt{2}}, \ldots, \frac{e^{-x^2/2}x^n}{\sqrt{n!}} \right\}$$  (3)

Suppose we let $n \to \infty$ and define new feature transformation in Equation 4. You can think of this feature transformation as taking some finite feature vector and producing an infinite dimensional feature vector rather than the simple two dimensional feature vector used in the earlier part of this problem.

$$\phi_\infty(x) = \left\{ e^{-x^2/2}, e^{-x^2/2}x, \frac{e^{-x^2/2}x^2}{\sqrt{2}}, \ldots, \frac{e^{-x^2/2}x^n}{\sqrt{n!}} \ldots \right\}$$  (4)

1. [3pt] We know that we can express a linear classifier using only inner products of support vectors in the transformed feature space as seen in Equation 2. It would be great if we could somehow use the feature space obtained by the feature transformation $\phi_\infty$. However, to do this we must be able to compute the inner product of examples in this infinite vector space. Let's define the inner product between two infinite vectors $a = \{a_1, \ldots, a_i, \ldots\}$ and $b = \{b_1, \ldots, b_i, \ldots\}$ as the infinite sum given in Equation 5.

$$k(a,b) = a \cdot b = \sum_{i=1}^{\infty} a_i b_i$$  (5)

Can we explicitly compute $k(a,b)$? What is the explicit form of $k(a,b)$? Hint you may want to use the Taylor series expansion of $e^x$ which is given in Equation 6.

$$e^x = \lim_{n \to \infty} \sum_{i=0}^{n} \frac{x^i}{i!}$$  (6)

2. [1pt] With such a high dimensional feature space should we be concerned about overfitting?

3 [45 points] k-NN, SVM, and Cross-Validation (Yucheng)

In this question, you will explore how cross-validation can be used to fit “magic parameters.” More specifically, you’ll fit the constant $k$ in the $k$-Nearest Neighbor algorithm, and the slack penalty $C$ in the case of Support Vector Machines. For all implementation questions, please electronically submit your source code to

/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW3/

and supply pseudo-code in your writeup where requested.

3.1 Dataset

1. Download the file hw3_matlab.zip and unpack it. The file faces.mat contains the Matlab variables traindata (training data), trainlabels (training labels), testdata (test data), testlabels (test labels) and evaldata (evaluation data, needed later).

This is a facial attractiveness classification task: given a picture of a face, you need to predict whether the average rating of the face is hot or not. So, each row corresponds to a data point (a picture). Each column is a feature, a pixel. The value of the feature is the value of the pixel in a grayscale image. (This is an “easier” version of the dataset on the project website.) For fun, try showface(evaldata(1,:)), showface(evaldata(2,:)), . . . .

---

cosineDistance.m implements the cosine distance, a simple distance function. It takes two feature vectors \( x \) and \( y \), and computes a nonnegative, symmetric distance between \( x \) and \( y \). To check your data, compute the distance between the first training example from each class. (It should be 0.2617)

### 3.2 \( k \)-NN (20 pts)

1. Implement the \( k \)-Nearest Neighbor (\( k \)-NN) algorithm in Matlab. Hand in pseudo-code. **Hint:** You might want to precompute the distances between all pairs of points, to speed up the cross-validation later.

2. Implement \( n \)-fold cross validation for \( k \)-NN. Your implementation should partition the training data and labels into \( n \) parts of approximately equal size. Hand in the pseudo-code.

3. For \( k = 1, 2, \ldots, 100 \), compute and plot the 10-fold (i.e., \( n = 10 \)) cross-validation error for the training data, the training error, and the test error. Don’t forget to hand in the plot!

   - How do you interpret these plots? Does the value of \( k \) which minimizes the cross-validation error also minimize the test set error? Does it minimize the training set error? Either way, can you explain why?
   - Also, what does this tell us about using the training error to pick the value of \( k \)?)

### 3.3 SVM (20pts)

1. Now download libsvm using the link from the course website and unpack it to your working directory. It has a Matlab interface which includes binaries for Windows. It can be used on OS X or Unix but has to be compiled (requires g++ and make) — see the README file from the libsvm zip package and/or the instructions on the course homework page.

   hw3 matlab.zip, which you downloaded earlier, contains files testSVM.m (an example demonstration script), trainSVM.m (for training) and classifySVM.m (for classification), which will show you how to use libsvm for training and classifying using an SVM. Run testSVM. This should report a test error of 0.4333.

   In order to train an SVM with slack penalty \( C \) on training set data with labels labels, call
   ```
   svmModel = trainSVM(data, labels, C)
   ```

   In order to classify examples test, call
   ```
   testLabels = classifySVM(svmModel, test)
   ```

   Train an SVM on the training data with \( C = 500 \), and report the error on the test set.

2. Now implement \( n \)-fold cross-validation for SVMs. Similarly to \( k \)-NN, split your training data into \( n \) roughly equal parts. Hand in the pseudo-code.

3. For \( C = 10, 10^2, 10^3, 10^4, 5 \cdot 10^4, 10^5, 5 \cdot 10^5, 10^6 \), compute and plot the 10-fold (i.e., \( n = 10 \)) cross-validation error for the training data, the training error, and the test error, with the axis for \( C \) in log-scale (try semilogx). Don’t forget to hand in the plot!

   - How do you interpret these plots? Does the value of \( C \) which minimizes the cross-validation error also minimize the test set error? Does it minimize the training set error? Either way, can you explain why?
   - Also, what does this tell us about using the training error to pick the value of \( C \)?

### 3.4 DIY (5 pts + 5 pts Extra Credit)

1. Design your favorite classifier: You have to use either \( k \)-NN or SVM, but you are allowed to use arbitrary values for \( k \) or for \( C \). For \( k \)-NN, you can invent different distance functions than the one we gave you or you can try to weigh the influence of training examples by their distance from the test point. If you want, you can do arbitrary feature selection, e.g. you can ignore some columns. You can also perform any linear transformation of the features if you want. Whatever you do, please document it, and apply your algorithm to the evaldata data set. Output your class labels for this evaluation set, one label per line, in the order of the examples from the evaluation set. Submit your labels as file evallabels.txt where yourid is your Andrew ID.
4 Learning Theory [Shay, 20 points]

4.1 VC Dimension

In this section you will calculate the VC-dimension of some hypothesis classes. Remember that in order to prove that $H$ has VC-dimension $d$ you need to show that

- There exists a set of $d$ points which can be shattered by $H$. (This step is often easy).
- There exists no set of $d+1$ points that can be shattered by $H$. (This step is hard).

1. (5 points) Find the VC-dimension of the hypothesis class that consists of the union of $k$ intervals on the real line. In other words each hypothesis $h \in H$ is associated with $k$ closed intervals $[a_i, b_i], i \in \{1, 2, \ldots, k\}$; and $h(x) = 1$ iff $x \in \bigcup_{i \in \{1, 2, \ldots, k\}} [a_i, b_i]$.

2. (5 points) Find the VC-dimension of the hypothesis class that consists of the set of axis aligned rectangles in the $n$-dimensional reals $\mathbb{R}^n$. That is, any $h \in H$ is characterized by $n$ closed intervals $[a_i, b_i]$ for $i \in \{1, 2, \ldots, n\}$, and for any $x \in \mathbb{R}^n$,

$$h(x) = 1 \text{ iff } \forall i \in \{1, 2, \ldots, n\}, \ x_i \in [a_i, b_i].$$

Hint: Can you always find a subset of examples such that labeling those points 1 forces all the other examples to be labeled 1?

3. (5 points) Let $C$ be a finite hypothesis class such that (meaning, $|C| < \infty$). Show that the VC-dimension of $C$ is bound: $\text{VCDim}(C) \leq \log_2 |C|$.

4.2 Sample Complexity

In this part, you will use sample complexity bounds to determine how many training examples are needed to find a good classifier.

- (5 points) Let $H$ be the hypothesis class of linear separators. Recall that the VC dimension of linear separators in $\mathbb{R}^n$ is $n+1$. Suppose we sample a number $m$ of training examples i.i.d. according to some unknown distribution $D$ over $\mathbb{R}^2 \times \{-1, 1\}$.

$$(X_1, Y_1), (X_2, Y_2), \ldots, (X_m, Y_m) \sim D$$

Prove that if $m \geq 14619$, then with probability at least .99 over the draw of the training examples, the linear separator with smallest training error $\hat{h}_{\text{ERM}} = \arg \min_{h \in H} \text{error}_{\text{train}}(h)$ has

$$\text{error}_{\text{true}}(\hat{h}_{\text{ERM}}) - \text{error}_{\text{train}}(\hat{h}_{\text{ERM}}) \leq .05$$

You may not assume $\text{error}_{\text{train}}(\hat{h}_{\text{ERM}}) = 0$. You may use any formulas from the lecture slides, textbook, or readings from the website, but please tell us where you found the formula(s) you use.