For Instructor/TA Use Only

Student Name:	
Student Last Name:	
Student Andrew ID:	

Q1 (max 20)	Q2 (max 40)	Q3 (max 10)	Q4 (max 30)	Total (max 100)

24-787 Artificial Intelligence and Machine Learning for Engineering Design

Homework 4

SVM and Optimization

Naming and Folder structure: Name a root folder "andrewid_hw#". Example: lkara_hw1. In that root folder, create subfolders such as q1, q2, q3 etc. each corresponding to a question in the assignment. The files and documents related to each question should go into the corresponding subfolder.

File types: Your folders and subfolders should not contain any files except .pdf, .doc, .docx, .m, .mat, .txt .zip files. If you want to take photos of your assignment, just make sure that combine all the jpg into a single pdf file and make it clear. Unorganized and illegible files will be penalized. Take care in arranging your illustrations, written solutions, photos. If any, do not scan the your hand-written solutions in the highest resolution.

What to include: Only submit the required files that you create or modify. For programming questions, include your source code (.m and/or .mat file rather than text) in your submission. All other files, including the ones we provide as supporting functionality, are unnecessary because we already have copies of them. Unless we tell you otherwise, make sure that you only submit the file you edited/changed. For .m files, we expect to click the "run" button and everything should work (obviously, after we include the additional support files and data that was given to you with the assignment).

Readme files: Provide a readme.txt within each questions' subfolders when necessary to aid grading. This is useful when there are many files of the same type. If your folder structure and your file names are logical, you should not need a readme.txt file.

Submission: Zip the entire assignment by compressing the root folder. Example: lkara_hw1.zip. Do not use .rar**.** Make sure your entire submission file is less than 20MB when zipped. You shall use the online homework submission platform. The submission entry can be found at the top right of homework/assignment page. Make sure your submission is in .zip format, otherwise it cannot be uploaded. Feel free to re-submit your homework before deadline if you find any mistake in your previous submission. Only the last submission will be graded.

1. *(20 points)*

   You will need Matlab 2014 or later to solve this problem. In this programming exercise, you will use support vector machines (SVM) to classify samples from a real-world dataset. Specifically, you will be predicting what activity a user is engaged in (e.g. walking, sitting, standing) based on sensor readings from their smartphone.

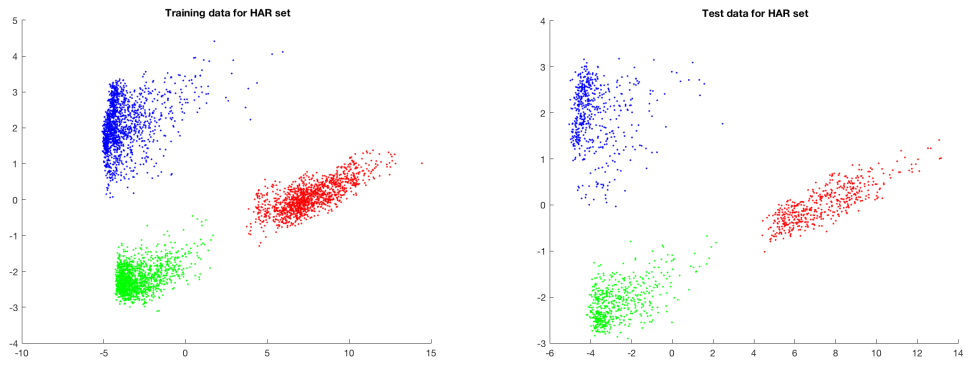
   To get started,

   - **Download or use the zipped** Human Activity Recognition dataset from the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
   - If downloading, extract all files to the directory of your choice.
   - Study the dataset files. The README.txt file is a good starting point.
   - Open and study **q1.m**

   We have provided code to read in the training and testing samples. Each sample contains 561 features extracted from smartphone accelerometer and gyroscope time series data. There are 6 labeled activities (walking, walking upstairs, walking downstairs, sitting, standing, and laying). These correspond to the class categories shown in **activity_labels.txt.**

   For the purposes of this assignment, we will be using a 2D projection of the training and testing data. We have provided the code for this dimensionality reduction, which uses Principal Component Analysis (PCA). Your goal is to create a 3-class SVM on this 2D data. You will apply your algorithm to two different classification problems: (A) Walking-Standing-Laying (labels [1,5,6]) and (B) Walking-Sitting-Laying (labels [1,4,6]).

   (a) **Plot the training data:** For case (A), plot the PCA reduced training and test data on two different plots. Your results should look like this:

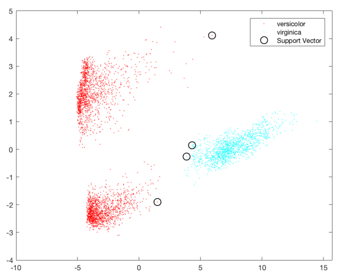


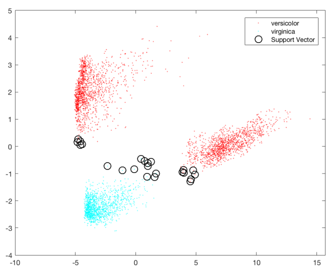
   (b) Output and document the 100th training data and the 100th test data, and their respective class labels (as integer).

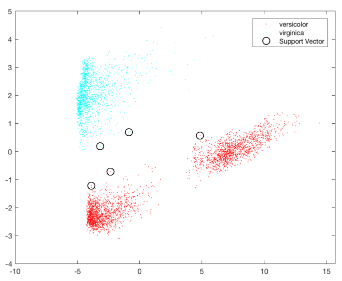
   (c) **Train and visualize your SVMs:** Since support vector machines are binary classifiers, we need a special way to handle multiple classes. Use the one-versus-all strategy on the PCA-reduced data, in which you train a separate SVM for each class (e.g. the first SVM will learn walking versus not walking).

For this, you can use Matlab's **fitcsvm** function with default parameters (i.e., your code should only use the generic form **fitcsvm(X,y),** with no other arguments. Study the documentation of this function. Your approach should yield a total of three decision boundaries; include plots of the training data with the support vectors circled in your report. Matlab's **fitcsvm** structure has functions for these tasks.

Note that in general you can customize the **fitcsvm** function with different parameter settings and fine-tuning, but we will not be exploring these options here. Your results should look like this:





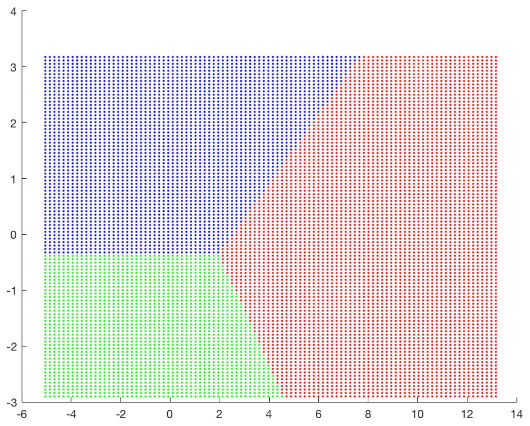


(d) **Introduce a new training sample:** If an extra sample for walking was added to the training dataset that has PCA-reduced coordinates of (4,-1), which (if any) of the decision boundaries would change? Explain your reasoning in a few sentences. Validate your reasoning by adding this data point to the training set, retraining your SVM, and carefully inspecting the new support vectors. Use plots similar to the ones in 'c' (possibly zoomed in) to support your conclusion.

(e) **Report test performance:** Use the SVMs learned in initially (that is, excluding the additional data point in 'd') to classify the test data. Use a winner-takes-all strategy, in which a sample is labeled with the class that has the highest confidence (distance to the decision boundary). Report test accuracy and show the confusion matrix. For this part, study the Matlab function **confusionmat**. To correctly solve this question, you will need to be able to compare the known class labels for each test sample to the corresponding label assigned by your SVM. Show the resulting 3X3 confusion matrix.

Hint: Your classification accuracy should be 100%. That is, your confusion matrix should be a diagonal matrix.

(f) **Show decision boundaries:** While there are different ways of visualizing the decision boundary, we ask you to do this in the following way. Create a 100X100 grid structure in the max-min range of the training data (excluding the extra point you added in 'd'). Create a set of test points by uniformly sampling this space. You may find **meshgrid** or **linspace** functions in Matlab to be useful. Compute the class label assigned by your SVM to each of the 10,000 test samples and plot the results. Your result should look like this:



(g) Now, change your original class labels to reflect case (B). This should involve commenting out one line and uncommenting another line in the provided .m file, and rerunning your training and testing algorithms.

For this case, report:

- Your new 3X3 confusion matrix (*hint:* the classification performance should no longer be 100%)
- The new decision boundary similar to the way you did in 'f'

2. *(40 points)*

You will use support vector machines (SVM) to classify samples from two synthetic datasets. In this problem, you cannot use Matlab's SVM functions. In class we have seen how formulating support vector machines (SVMs) involves implementing and solving a constrained quadratic optimization problem. In this problem, you will implement linear SVMs in the primal form, first for the linearly separable case, and later for the non-separable case. Conveniently, Matlab has a solver for constrained quadratic optimization problems built in (quadprog), which does essentially all of the work for you. This solver can solve arbitrary constrained quadratic optimization problems of the type:

$$\begin{aligned} \arg\min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^\top \mathbf{H}\mathbf{z} + \mathbf{f}^\top \mathbf{z} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{z} \leq \mathbf{b}. \end{aligned}$$

Your task it to convert the SVM formulation (see below) into this generic input format of the quadprog function, and then turn its output into a linear discriminant function.

Task 1: Linearly separable case *(15 out of 40 points)*

a) Open the file **q2_lin_clean.m** and load the training data **clean_lin.txt**. Make a scatter plot that allows you to see which data point belongs to which class.

b) Implement the following quadratic optimization problem:

$$\arg\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \qquad \forall i.$$

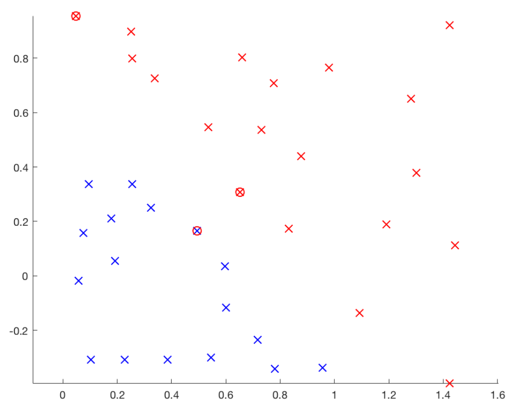
To do this, you have to find how to turn this constrained optimization problem into the generic form from above. Things you should ask yourself are: Which variable(s) does **z** need to represent? What do you need to set **H,f,A**, and **b** to? Note that the **b** in the SVM formulation is not the same as the **b** in the quadprog formulation. Show your derivation of the formulas on paper (hand-written or typed). Be clear about the size of these matrices and vectors.

Hint: **z** should be 3X1. **A** should be nX3 where n is the number of training samples. You may find the following Matlab functions useful: ones, bsxfun. Your code must have a line that reads:

[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b)

Train the linear SVM on the training data. Output the final optimized values of **w** and **b**.

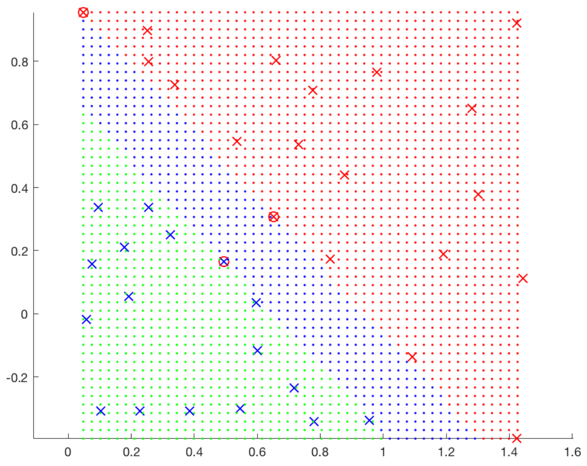
c) Output the **lambda** vector. To do that, inspect the Lagrange multipliers that quadprog returns. Show the corresponding support vectors on your plot. Your output should now look like this:



d) Create a test set in the form of a uniform grid of approximately **50X50** using the bounding box of the training data. Here is a sample code you may use:

```
d = min((max(X(:,1)) - min(X(:,1))), ((max(X(:,2)) - min(X(:,2)))))/50;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1)
```

Where X is the nX2 data matrix. xGrid is your test data. Classify each point in xGrid using your SVM. Provide a scatter plot of these points and identify the partitions of positive labels, negative labels and the margin zone. Note that the margin zone is where the classification is in the range [-1.0 , +1.0]. Your plot should look like this:



e) Now, using the same code, load the file **dirty_nonlin.txt** (just comment one line, uncomment another). Show that your quadprog does not converge (cannot find a solution). You can see this by removing the comma at the end of the line (in case you added a comma):

    [x,fval,exitflag,output,lambda] = quadprog(H,f,A,b)

In your report, include the text produced by Matlab that confirms that a solution has not been found.

Task 2: Linearly non-separable case *(25 out of 40 points)*

a) Open the file **q2_lin_dirty.m** and load the training data **dirty_nonlin.txt**. Make a scatter plot that allows you to see which data point belongs to which class.

b) Implement the quadratic optimization problem for this case, which now contains slack variables:

$$\begin{aligned} &\underset{\mathbf{w},b,\boldsymbol{\xi}}{\arg\min} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ &\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \forall i \\ &\qquad \xi_i \geq 0, \qquad \forall i. \end{aligned}$$

At this point, use C=1 in your code, but keep that as a variable because you will be asked to vary C in the subsequent questions.

Again, ask yourself what the various terms in the generic formulation need to represent now. Show your derivation of the formulas on paper (hand-written or typed). Be clear about the size of these matrices and vectors.

Hint: The problem is still quadratic in the design variables. So your solution, if found, will be the global minimum. **z** should be (n+3)X1. **A** should be 2nX(n+3) where n is the number of training samples. If you construct your design vector as [w1; w2; b; psi_1,…; psi_n], you shall see that **A** can be constructed by putting together four sub matrices (upper left, lower right etc.): **A_ul, A_ur**, **A_ll, A_lr**. **A_ul** should be nX3. You may find the following Matlab functions useful: zeros, ones, eye, bsxfun. Your code must have a line that reads:

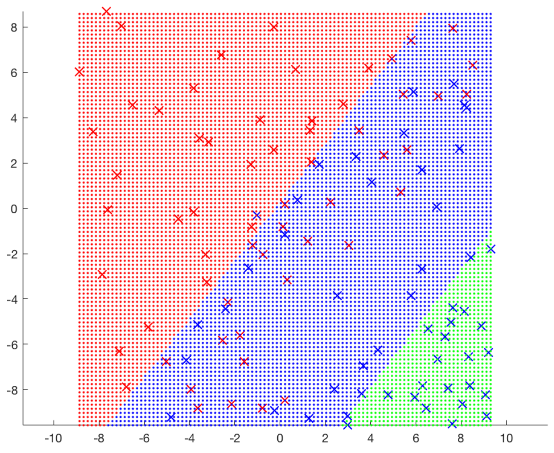
[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b)

Train the linear SVM on the training data. Output the final optimized values of **w** and **b** for C=1.

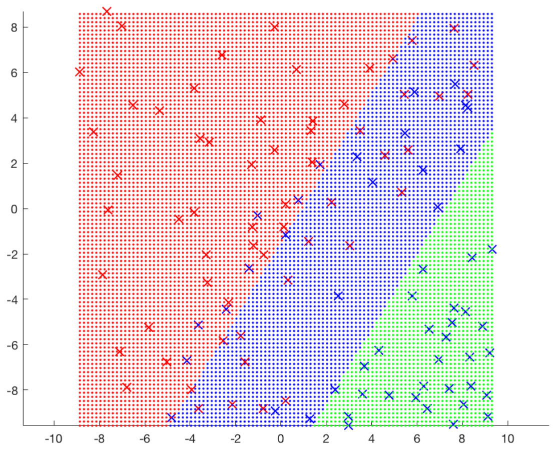
c) Create a test set in the form of a uniform grid of approximately **100X100** using the bounding box of the training data. Here is a sample code you may use:

```
d = min((max(X(:,1)) - min(X(:,1))), ((max(X(:,2)) - min(X(:,2)))))/100;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1)
```

Provide 5 plots, each corresponding to different values of C= [0.05, 0.1, 1, 100, 1000000]. The figures below show the classification for C=0.05 (left) and C=1000000 (right).



C=0.05



C=1000000

3. *(10 points)*

Given the objective function $0.5x^2 + 3.2x + 4.5y^2 - 6y + 7$

Subject to:

$x, y \geq 0$

$x + 3y \geq 15$

$2x + 5y \leq 100$

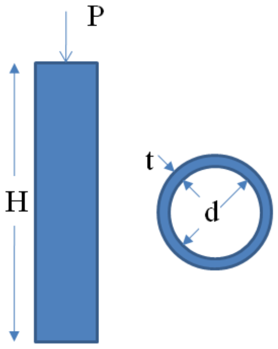
$3x + 4y \leq 80$

(a) Formulate the problem using Matlab's `quagprog`. Write the above optimization problem in the standard form by identifying the matrices and vectors `H,f,A,b,Aeq,beq,lb,ub`.

(b) Solve the problem using Matlab's quadprog "`'interior-point-convex' (default)`" option.

(c) Repeat the previous step using Matlab's `fmincon` function

4. *(30 points)*



You are asked to design a column to support a compressive load P as shown in the figure. The column has a cross-section shaped as a thin-walled pipe. The design variables are the mean pipe diameter *d* and the wall thickness *t*. The cost of the pipe is given by:

$$\text{Cost} = f(t,d) = c_1 W + c_2 d$$

Where $c_1 = 4$ and $c_2 = 2$ are the cost factors and W = weight of the pipe. The column must support the load under compressive stress and not buckle.

The compressive stress is given by: $\sigma = \frac{P}{A} = \frac{P}{\pi d t}$

The buckling stress can be shown to be: $\sigma_b = \frac{\pi E I}{H^2 d t}$

where E is the modulus of elasticity and I is the second moment of the area of the cross section given by:

$$I = \frac{\pi}{8} d t (d^2 + t^2)$$

Finally, the diameters of available pipes are between d_1 and d_2, and thicknesses between t_1 and t_2.

(a) Develop the optimization problem on paper. Clearly show your mathematical model (e.g. objective function, design variables, design parameters you find useful, and necessary constraints).

(b) Using Matlab's Active Set algorithm, solve this problem by determining the values of *d* and *t* that minimize the cost. Note that H=275cm, P=2000kg, E=900,000kg/cm^2, d_1=1cm,

d_2=10cm, t_1=0.1cm, and t_2=1cm. The yield stress of the material is 550kg/cm^2. The density of the material is 0.0025 kg/cm^3. Submit both your mathematical model and your software solution file. What are the resulting thickness and diameter values? Provide all m-files including a main script as well as the m-files you may have written for the objective functions and constraints. You may have to study the following document and related examples: http://www.mathworks.com/help/toolbox/optim/ug/brnoxzl.html

(c) Solve the same problem using MATLAB's SQP algorithm. Use the same initial values you used in part (b). Report your results similar to the way you did in part (b).

(d) Compare the solutions you obtained using Active-set and SQP methods. Do the results agree? Keeping everything else the same, choose a different set of initial values and re-run the two algorithms. Report whether the final solution is consistent between the two algorithms.