CIS 520, Machine Learning, Fall 2018: Assignment 4
Due: Friday, October 12th, 11:59pm
PDF to Gradescope, Data Output to Autograder
No Late Submissions Allowed

For Problems 1-3

- **Instructions.** Please write up your responses to the following problems clearly and concisely. We encourage you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**
- **Collaboration Policy.** You are allowed and encouraged to work together. You may discuss the homework to understand the problem and reach a solution in groups up to size **two students.** However, *each student must write down the solution independently, and without referring to written notes from the joint session.* **In addition, each student must write on the problem set the names of the people with whom you collaborated.** You must understand the solution well enough in order to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

For Problem 4

- **Instructions.** This is a MATLAB programming problem. You can submit your prediction results to be automatically evaluated to receive feedback ahead of time.

  We are providing you with codebase / templates / dataset that you will require for this problem. Download the file `hw4_kit.zip` from Canvas **before** beginning the assignment. **Please read through the documentation provided in ALL Matlab files before starting the assignment.** The instructions for submitting your homeworks and receiving automatic feedback are online on the wiki:

  http://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Resources.HomeworkSubmission

  If you are not familiar with Matlab or how Matlab functions work, you can refer to Matlab online documentation for help:

  http://www.mathworks.com/help/matlab/
- **Collaboration Policy.** You are allowed and encouraged to work together. You may discuss the homework to understand the problem and reach a solution in groups up to size **two students.** You can write **one copy** of solution code per group. However, each group member needs to submit plots, and codes separately (can be the same as your collaborator's) in the PDF submitted to gradescope. For the autograder you need to submit a copy of your data predictions **separately** as well.

Collaborators:

Type Collaborator Name Here

1 Convolutional Neural Network [20 points]

Convolutional Neural Network (CNN) is considered superior to regular neural networks when dealing with image classification problems. In this question, we will work through the details of a CNN model. For simplicity, we will assume there is no bias in this model.

1. Regular neural networks use fully connected layers. If we use a regular neural net to classify a set of 108×162 RBG images, how many weights do we need for each single neuron in the first hidden layer?

2. CNNs use a convolutional layer to help reduce the number of parameters in the model. Each neuron only connects to a small local spatial region of the images. If we use a filter (weights on the local region) of size $18 \times 6 \times 3$, how many weights do we need now for a single neuron in this convolutional layer?

3. We can adjust the size of the filters as well as the stride to limit the number of neurons we need in a convolutional layer. In the previous example of 108×162 RBG images, if we choose to use a stride of 6 in both x and y dimensions, how many neurons will be there in this layer? Note that we greatly reduce the dimension of the original image parameter space after this convolutional layer.

4. Using the following toy example, let's compute by hand exactly how convolutional layer works.

$$\text{input image} = \begin{bmatrix} 4 & 4 & 1 & 3 & 2 \\ 2 & 2 & 4 & 1 & 2 \\ 5 & 1 & 2 & 5 & 1 \\ 2 & 1 & 5 & 2 & 4 \\ 4 & 3 & 4 & 5 & 1 \end{bmatrix} \quad \text{filter 1} = \begin{bmatrix} -1 & 0 & 1 \\ -3 & 0 & 2 \\ 1 & 1 & 2 \end{bmatrix} \quad \text{filter 2} = \begin{bmatrix} 2 & -2 & 1 \\ -1 & 0 & 2 \\ 3 & -2 & 0 \end{bmatrix}$$

   Here we have a $5 \times 5 \times 1$ input image, and we are going to use 2 different filters with size $3 \times 3 \times 1$ and stride 1 as our first convolutional layer. Compute and write the exact output from this simple convolutional layer (Hint: the output dimension is $3 \times 3 \times 2$).

2 Optimization and Lagrangian Duality [10 points]

As a simple example of a constrained optimization problem that can be solved easily through duality, consider the following optimization problem where the goal is to minimize a quadratic function of two variables (x_1, x_2) subject to a given affine space:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(x_1^2 + x_2^2) \\ \text{subject to} & 3x_1 + 2x_2 = 1\,. \end{array}$$

This is a convex optimization problem that satisfies Slater's condition, so the strong duality holds. You will find the solution by solving the (Lagrange) dual problem.

1. Since there is just one equality constraint, there will be one dual variable ν associated with it. Write down the Lagrangian function $\mathcal{L}(x_1, x_2, \nu)$.

2. Write down the dual function $\phi(\nu)$. (To obtain this, you will need to minimize $\mathcal{L}(x_1, x_2, \nu)$ over x_1, x_2; you can do this by setting the gradient to zero, solving for x_1 and x_2 in terms of ν, and then plugging back these values of x_1 and x_2 into $\mathcal{L}(x_1, x_2, \nu)$).

3. The dual problem is now

$$\text{maximize} \quad \phi(\nu) \quad \text{(no constraint on } \nu\text{)}$$

This dual problem is always a convex optimization problem since $\phi(\nu)$ is concave and maximizing a concave function is equivalent to minimizing a convex function. Since the strong duality holds in this case, the optimal solution to the dual problem corresponds to the optimal solution to the primal problem. Please solve the dual problem first to find optimal ν^*, and then obtain a solution (x_1^*, x_2^*) to the primal problem by plugging in the value of ν^* into the expressions for x_1 and x_2 you derived in part 2.

3 Kernel Functions [10 points]

Let $K_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be two symmetric, positive definite kernel functions, and for simplicity, assume that each implements dot products in some finite-dimensional space, so that there are vector mappings $\phi_1 : \mathcal{X} \to \mathbb{R}^{d_1}$ and $\phi_2 : \mathcal{X} \to \mathbb{R}^{d_2}$ for some $d_1, d_2 \in \mathbb{Z}_+$ such that

$$K_1(x, x') = \phi_1(x)^\top \phi_1(x'), \qquad K_2(x, x') = \phi_2(x)^\top \phi_2(x') \qquad \forall x, x' \in \mathcal{X}.$$

For each of the following functions $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, either find a vector mapping $\phi : \mathcal{X} \to \mathbb{R}^d$ for some suitable $d \in \mathbb{Z}_+$ such that $K(x, x') = \phi(x)^\top \phi(x')$ $\forall x, x'$ (a valid kernel), or explain why such a mapping cannot exist (not a valid kernel).

1. $K(x, x') = c \cdot K_2(x, x')$, where $c > 0$
2. $K(x, x') = K_1(x, x') - c \cdot K_2(x, x')$, where $c > 0$
3. $K(x, x') = K_1(x, x') + K_2(x, x')$
4. $K(x, x') = K_1(x, x') \cdot K_2(x, x')$
5. $K(x, x') = K_1(f(x), f(x'))$, where $f : \mathcal{X} \to \mathcal{X}$ is any function

4 SVM and Neural Nets: Programming Exercise [60 points]

4.1 SVM on synthetic data [30 points]

In this problem you will write a piece of MATLAB code to experiment with SVMs on both synthetic and real-world data. You will use MATLAB fitcsvm for training your SVMs. You will be given binary classification datasets $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is an $m \times d$ matrix (m instances, each of dimension d) and y is an m-dimensional vector (with $y_i \in \{\pm 1\}$ being a binary label associated with the i-th instance in $\mathbf{X}$). You will be experimenting with the linear kernel and the RBF kernel K_{rbf}, which is defined as $K_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{\sigma^2}}$ with width parameter σ.

1. **Linear kernel:** You are provided a 2-dimensional synthetic dataset (See folder \`Synthetic`\ in hw4-kit.zip). You will be training SVMs on the training set using linear kernel.

In order to select the SVM parameter C, do 5-fold cross-validation on the training data using the cross-validation folds provided to you. Note that the training data is already divided into 5 folds, and the resulting datasets for cross-validation are provided in the folder `\Synthetic\Cross-Validation`; you should not create your own folds. Use cross-validation to select C from the range $\{1, 10, 10^2, 10^3, 10^4, 10^5\}$.

Task: For SVM with a linear kernel, after selecting C via cross-validation as above, learn a classier from the entire training data. Plot the decision boundary of this classifier learned on top of test data using the code provided `decision_boundary_SVM.m, use test data for scatter plot`. Also, report the cross-validation error, training error and test error achieved by the learned model (the train and test data are provided in the folder `\Synthetic`).

2. **RBF kernel:** You will now repeat part (1) for the RBF kernel. This time, you will be training SVMs with different values of the RBF parameter σ in $\{0.1, 1, 10, 100, 1000\}$. Again, for each σ, select C from the range $\{1, 10, 10^2, 10^3, 10^4, 10^5\}$ using cross-validation on the folds provided. For each σ, keep the cross-validation error you obtained for the chosen C this will be used in the error plot.

   Task: For each $\sigma \in \{0.1, 1, 10, 100, 1000\}$, after selecting C via cross-validation as above, learn a classifier from the entire training data set. Plot the decision boundary of this classifier learned on top of test data using the code provided `decision_boundary_SVM.m, use test data for scatter plot` Also, plot the cross-validation error, training error and test error achieved by each value of $\sigma \in \{0.1, 1, 10, 100, 1000\}$ ($\log(\sigma)$ on the x-axis and the classification error on the y-axis). Which value of σ achieves the lowest test error? Which value of σ achieves the lowest cross-validaion error?

3. **Cross-validation over kernels:** Here you will again experiment with RBF kernels over the same synthetic dataset as in parts (1) and (2), but now you will see if you can select a good *kernel* using cross-validation. In order to select a good RBF kernel via cross-validation, you will be doing cross-validation over 30 parameter combinations (5 kernel parameters $\times$ 6 values of C parameter). Note that finding the smallest cross-validation error among these 30 parameter combinations is the same as finding smallest cross-validation error among the 5 combinations of (σ, C) in part 2, thus you do not need to calculate cross-validation errors again.

   Task: For the RBF kernel report the absolute difference between the test error you got for the best value of σ in part(2) (i.e. σ that achieves lowest test error), and the test error you got after selecting σ using cross-validation (i.e. σ that achieves lowest cross-validation error). Is this difference in test errors large? Can you conclude that selecting these parameters by cross-validation is a good approach?

4.2 SVM and Neural Nets on Breast Cancer Dataset [30 points]

1. **SVM:** For this problem you will be training SVMs on the breast cancer dataset provided in the folder `\Breast-Cancer` (which was also used in HW2), and will be applying the learned classifiers to some test data **provided to you**. You will **upload the predicted labels** in a .mat file to the autograder, instead of your code.

   Task: Your task is to learn a SVM model with RBF kernel or linear kernel, with same kernel parameters σ and SVM parameters C as for synthetic data above. You should choose the parameters using the 5-fold cross-validation data provided in `\Breast-Cancer\Cross-Validation`. Use the training data provided in `\Breast-Cancer`. After you have chosen best parameters, learn a model (think about how you will go about choosing the model) from the entire training data and use it to predict +1/-1 labels for the test set. Save the labels in a file named SVMlabels.mat and upload it to autograder. Summarize and report your findings, such as cross-validation results, any training or test errors, as well as performance comparisons.

   Please do not submit your code to autograder. Instead enter your code in the LaTex documents using the following command:

```
\ begin{verbatim}
    paste your code here
\ end{verbatim}
```

2. **Neural Nets:** For this problem you will be writing a piece of MATLAB code to experiment with Neural Nets on breast cancer dataset provided in the folder `\Breast-Cancer` (which was also used in HW2), and will be applying the learned classifiers to some test data **provided to you**. **Upload the predicted labels** in a .mat file to the autograder, instead of your code. Use the MATLAB Deep Learning Toolbox*. For this assignment, use Neural Net architecture with one hidden layer and 10 neurons. For both sigmoid and ReLU activation functions, use cross validation to select L_2 regularizer C from the set $\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ refer to MSE and Cross Entropy documentation.

   References:

   - MATLAB Deep Learning Toolbox: https://www.mathworks.com/help/deeplearning/getting-started-with-deep-learning-toolbox.html
   - Multilayer Shallow Neural Networks and Backpropagation Training: https://www.mathworks.com/help/deeplearning/ug/multilayer-neural-networks-and-backpropagation-training.html

   Task: For each activation unit, after selecting regularization coefficient C via cross-validation as above, learn a classier from the entire training data. After you have chosen best parameters, learn a model (think about how you will go about choosing the model) from the entire training data and use it to predict +1/-1 labels for the test set. Save the labels in a file named NNlabels.mat and upload it to autograder. Summarize and report your findings, such as cross-validation results, any training or test errors, as well as performance comparisons.

   Please do not submit your code to autograder. Instead enter your code in the LaTex documents using the following command:

```
\ begin{verbatim}
    paste your code here
\ end{verbatim}
```

*Here is a small example of how we can use the deep learning toolbox:

```
load crab_dataset       % load MATLAB built in dataset
[x,t] = crab_dataset;       % x is the feature space, t is the true target
net = feedforwardnet([20 10 10]);   % 3-layer neural network of 20,10,10 neurons in each layer
net.layers{1}.transferFcn = 'logsig';   % 1st layer activation function is logistic sigmoid
net.layers{2}.transferFcn = 'tansig';   % 2nd layer activation function is tanh
net.layers{3}.transferFcn = 'poslin';   % 3rd layer activation function is ReLU
net.performFcn = 'crossentropy'   %specify loss function appropriate for classification
net.performParam.regularization = 0.5   % regularization parameter
net = trainlm(net,x,t); % train the neural network (you can also use a different
                        training function that implements gradient descent in a different way)
view(net)               % view the network structure
y = net(x);             % predict scores for each class
plotconfusion(t,y)      % plot confusion matrix
```