HOMEWORK 6

>>NAME HERE<<
>>ID HERE<<

Instructions:

- Please submit your answers in a single pdf file. Preferably made using latex. No need to submit latex code.
- See piazza post @114 for some recommendations on how to write the answers.
- Submit code for programming exercises. You can use any programming language you like, but we highly recommend python and pytorch.
- Submit all the material on time.

1 Kernel SVM [30 points]

Consider the following kernel function defined over $z, z' \in Z$:

$$k(z, z') = \begin{cases} 1 & \text{if } z = z', \\ 0 & \text{otherwise.} \end{cases}$$

1. (10 pts) Prove that for any integer $m > 0$, any $z_1, \ldots, z_m \in Z$, the $m \times m$ kernel matrix $K = [K_{ij}]$ is positive semi-definite, where $K_{ij} = k(z_i, z_j)$ for $i, j = 1 \ldots m$. (Let us assume that for $i \neq j$, we have $z_i \neq z_j$.) Hint: An $m \times m$ matrix K is positive semi-definite if $\forall v \in \mathbb{R}^m, v^\top K v \geq 0$.
2. (10 pts) Given a training set $(z_1, y_1), \ldots, (z_n, y_n)$ with binary labels, the dual SVM problem with the above kernel k will have parameters $\alpha_1, \ldots, \alpha_n, b \in \mathbb{R}$. (Assume that for $i \neq j$, we have $z_i \neq z_j$.) The predictor for input z takes the form
$$f(z) = \sum_{i=1}^{n} \alpha_i y_i k(z_i, z) + b.$$
Recall the label prediction is $\text{sgn}(f(z))$. Prove that there exists $\alpha_1, \ldots, \alpha_n, b$ such that f correctly separates the training set. In other words, k induces a feature space rich enough such that in it any training set can be linearly separated.
3. (10 pts) How does that f predict input z that is not in the training set?

Comment: One useful property of kernel functions is that the input space Z does not need to be a vector space; in other words, z does not need to be a feature vector. For all we know, Z can be turkeys in the world. As long as we can compute $k(z, z')$, kernel SVM works on turkeys.

2 Game of Classifiers (70 pts)

2.1 Implementation (10 pts)

Implement the following models in choice of your programming language. Include slack variables in SVM implementation if needed. You can use autograd features of pytorch, tensorflow etc. or derive gradients on your own.(But don't use inbuilt models for SVM, Kernel SVM and Logistic Regression from libraries)

- Implement Linear SVM (without kernels).
- Implement Kernel SVM, with options for linear, rbf and polynomial kernels. You should keep the kernel parameters tunable (e.g. don't fix the degree of polynomial kernels but keep it as a variable and play with different values of it. Is Linear SVM a special case of Kernel SVMs?
- Implement Logistic Regression with and without kernels (use same kernels as above).

2.2 Evaluation on Synthetic Dataset

2.2.1 Synthetic Dataset-1 (20 pts)

Generate 2-D dataset as following,
Let $\mu = 2.5$ and I_2 be the 2×2 identity matrix. Generate points for the positive and negative classes respectively from $\mathcal{N}([\mu, 0], I_2)$, and $\mathcal{N}([-\mu, 0], I_2)$. For each class generate 750 points, (1500 in total). Randomly create train, validation and test splits of size 1000, 250, 250 points respectively. Do the following with this dataset:

- (5 pts) Train your Linear SVM, Logistic Regression models and report decision boundaries, test accuracies.
- (5 pts) Show decision boundaries with K-NN and Naive Bayes Classifiers. (You can use library implementations or implement from scratch. Figure out the hyper-parameters using the validation set.)
- (5 pts) Repeat the process by varying μ from 1.0 to 2.4 with step size of 0.2, for each value of μ obtain test accuracies of the models and plot (μ on x-axis and test accuracy on y-axis). (You will have a curve for each of the 4-classifiers mentioned above)
- (5 pts) What are your conclusions from this exercise?

2.2.2 Synthetic Dataset-2 (20 pts)

Generate 1500 data points from the 2-D circles dataset of sklearn (`sklearn.datasets.make_circles`). Randomly create train, validation and test splits of size 1000, 250, 250 points respectively. Evaluate the above classifiers on this setting.

- (5 pts) Show decision boundaries for Linear SVM and Logistic Regression classifiers.
- (5 pts) Show decision boundaries for Kernel SVM and Kernel Logistic Regression (use rbf, polynomial kernels). Try different values of hyperparameters, report results with whichever works the best.
- (5 pts) Train Neural Network from HW4, and K-NN classifiers on this dataset and show decision boundaries. (You can use library implementation for these classifiers).
- (5 pts) What are your conclusions from this exercise?

2.3 Evaluation on Real Dataset (20 pts)

Lets put all this to some real use. For this problem use the the Wisconsin Breast Cancer dataset. You can download it from sklearn library(`sklearn.datasets.load_breast_cancer`)

- (10 pts) Do all the points of section 2.2.2 on this dataset. Since this is high-dimensional data, so you don't have to show the decision boundaries. Just report test accuracies for these classifiers and discuss your findings.
- (10 pts) In addition, you also want to figure out the important features which determine the class. Which regularization will you use for this? Upgrade your SVM, Kernel SVM implementation to include this regularization. Discuss what are the important features that you obtain by running your regularized SVM on this dataset. (You might have to normalize this dataset before training any classifier).