EE 559 **Homework 5** Posted: Fri., 2/24/2023.
Jenkins/Chugg Due: Fri., 3/3/2023, 11:59 PM*.
*No submissions will be accepted after Sun., 3/5/2023, 11:59 PM.

Notes for the computer problem (Pr. 2):

(1) For this homework assignment, you are **allowed** to use selected functions from the **Scikit-learn library.** Scikit-learn is a high-level machine learning library in Python built on top on NumPy. You may use other Python built-in functions, NumPy, matplotlib like previous homework; the PlotNonlinear.py function from homework 4 would be useful for Problem 2, and you may use pandas only for reading and/or writing csv files.

(2) Also see the piazza post on *Using Sample Code from Lecture/Discussion* (pinned near the top of your Q&A feed) that describes using code from lecture, discussion, or Prof. Chugg's GitHub postings. This applies to all homeworks and the class project.

1. (a) Write the criterion function J_{MSE} for a 2-class MSE classifier. Then add an L2 regularizer term (similar to the L2 regularizer term for Ridge Regression). Give any restrictions on $\underline{b}$.

   (b) Solve for the MSE solution using a pseudoinverse technique.

2. **2-class MSE Classifier.** In this problem you will code the MSE Classifier for binary classification on a synthetic dataset that has two features. You will implement it by using Linear Regression with and without L2 regularization. You will also transform the features to higher dimensional space using polynomials of different order and compare results.

   In order to turn the class labels into target values for regression, use the following convention. The data points of the first class are assigned target value +1 and data points of the second class are assigned target value -1. After the regression model is trained, you may use simple decision rule to assign class labels – if predicted output is positive, it is assigned to the first class, else if the predicted output is negative, it is assigned to the second class.

   Do **not** use reflected data points **nor** perform any standardization or any data preprocessing (other than generating polynomial features). Do not use augmented feature space as the polynomial generator function will do it for you.

   In order to generate the polynomial features you may use the **PolynomialFeatures** class from **sklearn.preprocessing.** Set the ***degree*** parameter to the desired degree of polynomial (p). Make sure that the ***include_bias*** parameter is to **True** (which is the default value). This ensures the transformed features are in the augmented space. Create an object of **PolynomialFeatures** and call the **fit_transform()** function to generate polynomial features of the specified degree. As a sanity check, make sure for ***degree*=*2***, you get **6** features. They are $[1, x_1, x_2, x_1^2, x_1 x_2, x_2^2]$.

   (a) Code a MSE classifier with 7 different degrees of polynomial ($p = 1,2,3,4,5,6,7$). As a baseline, do not use any regularization.

Use the **LinearRegression** or **Ridge** class from **sklearn.linear_model** for this problem. It is important to set the ***fit_intercept*** parameter to **False** (default value is **True**). This is because your features are already in the augmented space and will have an additional weight component (w_0) associated. If you use **Ridge** regression class, make sure you set ***alpha=0.*** This will default to Linear Regression without regularization. You will need the ***fit()*** method to train the regressor and ***predict()*** to obtain prediction. Explore other parameters/methods of these classes for your understanding. No need to report anything.

i. Report the classification accuracy on the **training and test** sets for each value of p.

ii. Plot the **training** data points and the decision regions for $p = 1,2,4,7$**.** Observe the same plots for other values of p (no need to include in your submission). Make sure that the plot is in the original two-dimensional feature space. The decision boundary would be linear in the transformed space and would appear to be non-linear on the 2D plot. The PlotNonlinear.py function would be useful here.

iii. Plot the train and test accuracy vs. p on a single plot for all values of p.

iv. Plot J_{mse} vs. p for all values of p.

Note: In order to calculate J_{mse} make sure you use the target values (-1/1) set for regression and not the original class labels. J_{mse} is calculated based on the training data.

(b) Compare and comment on the results in part (a). In particular, how does the train and test accuracy vary as p increases. Also, how do the decision boundaries appear? Do you see any effect of overfitting?

(c) How many d.o.f and constraints are there (for each p) and how do they relate to the obtained results?

(d) Repeat part (a) but with regularization. Use six different values of λ, $\lambda = 0.3, 1, 3, 10, 30, 100$. The **sklearn Ridge** class uses parameter alpha instead, however it is the same as λ. Note that you can no longer use the **LinearRegression** class for this part. For the plots in (a)ii. of the decision regions, report only the plots for $p = 1,2,4,7$ for this sub problem, each with $\lambda = 1,10,100$ (12 plots in all). However, observe all plots for your understanding. Repeat everything for (a)i., (a)iii., (a)iv. as it is (for all seven values of p and six values of λ).

(e) Additionally, plot test accuracy vs. log (λ), for $p = 1,2,4,7$ on a single plot. Use log base 10. (Also consider $\lambda = 0$ case, which will be your results from part (a).)

(f) Compare and comment on the results in part (d) and how they relate to results from part (a). Do you see any effect of regularization? Explain briefly.

(g) Study the sklearn **LinearRegression / Ridge** class and explain how to obtain the trained weight vector. The weight vector can be used to write the equation of the decision boundary / the decision rule. Give the decision rule for $p = 2$ in part (a).

It will be a quadratic equation in terms of the two features. For instance, if your algorithm returns a circular decision boundary of radius 2 centered at origin, then the equation of decision boundary will be $x_1^2 + x_2^2 - 4 = 0$. Your decision rule will very likely also involve x_1, x_2, and x_1x_2 terms.

3. (a) You are given the following criterion function for a 2-class classification problem:

$$J(\underline{w}) = -\sum_{n=1}^{N} [\![\underline{w}^T z_n \underline{x}_n \leq 0]\!] \, \underline{w}^T z_n \underline{x}_n$$

in which augmented notation is used.

Prove that $J(\underline{w})$ is convex.

Hints:

(i) How can $[[a \leq 0]]\, a$ be written using the max[.] function? ($[[..]]$ denotes the indicator function.)

(ii) You may find Discussion 4 notes on convexity helpful.

(b) Suppose you make the classifier nonlinear by using a nonlinear transformation of the feature space first:

$$\underline{x} \rightarrow \underline{\varphi}(\underline{x})$$

so that the criterion function is:

$$J(\underline{w}') = -\sum_{n=1}^{N} \left[\left[\underline{w}'^T z_n \underline{\phi}(\underline{x}_n) \leq 0\right]\right] \underline{w}'^T z_n \underline{\phi}(\underline{x}_n).$$

Is $J(\underline{w}')$ convex? Prove your answer.

4. In this problem you will solve a constrained optimization problem by hand. You will minimize a function $f(\underline{w})$ with respect to $\underline{w}$, subject to a set of inequality constraints that depend on a set of data points $\underline{u}_i,\ i = 1,2,\cdots,N$. (In part (c) you will use $N = 2$; before then keep N as a variable.) You will then use this result to find a decision boundary and regions for a 2-class classifier based on the 2 given data points.

You will use Lagrangian optimization. Nonaugmented notation will be used throughout this problem.

Consider that we want to minimize the function $f(\underline{w}) = \frac{1}{2}\|\underline{w}\|_2^2$ subject to the constraints $z_i\left(\underline{w}^T \underline{u}_i + w_0\right) \geq 1$ for all $i = 1, 2, \cdots, N$, in which $\underline{u}_i$ is the i^{th} training sample in expanded feature space, and as usual $z_i = \begin{cases} +1, & \underline{u}_i \in S_1 \\ -1, & \underline{u}_i \in S_2 \end{cases}$.

Tip: Write clearly so that μ_i and $\underline{u}_i$ are always distinguishable.

(a) If the above constraints are satisfied, will the training samples be correctly classified? (**Hint**: remember $\underline{u}_i$ is not augmented; if you're not sure of the answer, you can temporarily convert the inequality to augmented notation. Please continue to use non-augmented notation for the rest of this problem.)

(b) (i) Set up the Lagrangian function $L\left(\underline{w}, w_0, \underline{\mu}\right)$ for minimization of f subject to the inequality constraints. Use $\mu_i,\, i = 1, 2, \cdots, N$ for the Lagrange multipliers. State the KKT conditions (should be 2 sets of restrictions that involve the μ_i, plus the original set of inequality constraints).

(ii) First minimize L w.r.t. the weights, by solving $\nabla_{\underline{w}} L = \underline{0}$ for the optimal weight $\underline{w}^*$ (in terms of μ_i and other variables). Also set $\frac{\partial L}{\partial w_0} = 0$ and simplify. Show your work as a way of proving your result.

(iii) Use the two expressions you obtained in (ii), to substitute into your expression for $L\left(\underline{w}, w_0, \underline{\mu}\right)$ in (i). Call your result L_D. Again, show your work. Thus, you should now have derived:

$$L_D\left(\underline{\mu}\right) = -\tfrac{1}{2}\left[\sum_{i=1}^{N}\sum_{j=1}^{N} \mu_i \mu_j z_i z_j \underline{u}_i^{\,T} \underline{u}_j\right] + \sum_{i=1}^{N} \mu_i$$

subject to the (new) constraints $\sum_{i=1}^{N} \mu_i z_i = 0$. Also write the equation for $\underline{w}^*$ from (ii) above, and the other KKT conditions from (i), for your complete result. This is the dual form of the original Lagrangian function.

You have solved for the optimal weight value in terms of $\underline{\mu}$. But the optimization isn't finished until you also optimize over $\underline{\mu}$, which you will do in part (c).

(c) You are now given a training set consisting of 2 samples, in a 2-class problem. In the expanded feature space, the data points are:

$$\underline{u}_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \in S_1; \quad \underline{u}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \in S_2.$$

Use Lagrangian techniques to optimize L_D w.r.t. $\underline{\mu}$ subject to the new constraint from (b)(iii), in order to find the numerical optimal weight vector $\underline{w}^*$.

Plot the training samples and the decision boundary in 2D (nonaugmented) $\underline{u}$-space using $g\left(\underline{x}\right) = \underline{w}^{*T}\underline{u} + w_0^* = 0$, and show which side of the boundary is positive (S_1).

Hints for (c): Start from:

$$L_D'\left(\underline{\mu}, \lambda\right) = \sum_{i=1}^{N} \mu_i - \tfrac{1}{2}\left[\sum_{i=1}^{N}\sum_{j=1}^{N} \mu_i \mu_j z_i z_j \underline{u}_i^{\,T} \underline{u}_j\right] + \lambda\left(\sum_{i=1}^{N} z_i \mu_i\right)$$

in which the last term has been added to incorporate the new equality constraint. Once finished, then check that the resulting $\underline{\mu}$ satisfies the restrictions you stated in (b)(i).

You can use one of the KKT conditions to help you find $w_0{}^*$.

(d) Use the data points and solution you found in part (c), and call its solution decision boundary $\mathbf{H}$. Calculate in 2D $\underline{u}-$space the (projected) distances $\left|d\left(H \rightarrow \underline{u}_1\right)\right|$ between $\mathbf{H}$ and $\underline{u}_1$, and $\left|d\left(H \rightarrow \underline{u}_2\right)\right|$ between $\mathbf{H}$ and $\underline{u}_2$. Is there any other possible linear boundary in $\underline{u}-$space that would give larger values for both distances (margins) than $\mathbf{H}$ gives? (No proof required.)