

Homework 2 – Deep Learning (CS/DS 541, Whitehill, Spring 2023)

You may complete this homework assignment either individually or in teams up to 2 people.

1. **Double Cross-Validation** [10 points]: Below is a Python/numpy implementation of *single* cross-validation (with k folds over the dataset \mathcal{D}) that estimates the accuracy of a model trained with hyperparameter configuration h .

```
def doCrossValidation (D, k, h):
    allIdxs = np.arange(len(D))
    # Randomly split dataset into k folds
    idxs = np.random.permutation(allIdxs)
    idxs = idxs.reshape(k, -1)

    accuracies = []
    for fold in range(k):
        # Get all indexes for this fold
        testIdxs = idxs[fold,:]
        # Get all the other indexes
        trainIdxs = idxs[list(set(allIdxs) - set(testIdxs)),:].flatten()
        # Train the model on the training data
        model = trainModel(D[trainIdxs], h)
        # Test the model on the testing data
        accuracies.append(testModel(model, D[testIdxs]))

    return np.mean(accuracies)
```

For instance, h might represent a particular value for, say, the learning rate used to train a neural network. Note that the code above assumes the existence of two methods, `trainModel` and `testModel`. It also assumes (for simplicity) that k divides the length of \mathcal{D} .

Your task: Implement a method called `doDoubleCrossValidation` that takes a dataset \mathcal{D} , the number of folds k , and a list of hyperparameter configurations \mathcal{H} . The model should follow the logic shown in the `Class3.pdf` slides, i.e., for each of the k “outer” folds, conduct k “inner” folds to decide what is the best hyperparameter configuration *for that outer fold*. For simplicity, you can assume that k^2 divides the length of \mathcal{D} .

```
def doDoubleCrossValidation (D, k, H):
    ...
```

2. **Convexity:**

- (a) Consider the function $f(x, y) = x^4 + xy + x^2$. Either prove that f is convex by showing that the Hessian matrix is positive semi-definite (PSD) everywhere in the domain of f , or identify a point in the domain of f where the Hessian is not PSD. [4 pts]
- (b) Prove that the (half-)MSE loss is convex for a 2-layer linear neural network w.r.t. weight vector \mathbf{w} , i.e.: $f_{\text{MSE}}(\mathbf{w}) = \frac{1}{2n}(\mathbf{X}^\top \mathbf{w} - \mathbf{y})^\top (\mathbf{X}^\top \mathbf{w} - \mathbf{y})$, where \mathbf{y} is the n -dimensional vector of ground-truth labels and \mathbf{X} is the design matrix. Hint: find the matrix expression for the Hessian \mathbf{H} of f (which does not depend on \mathbf{w}); then, show that, for any real vector \mathbf{v} , it is always true that $\mathbf{v}^\top \mathbf{H} \mathbf{v} \geq 0$. [8 pts]

3. **L_2 -regularized Linear Regression via Stochastic Gradient Descent** [20 points, in Python]: Train a 2-layer neural network (i.e., linear regression) for age regression using the same data as in homework 1. Your prediction model should be $\hat{y} = \mathbf{x}^\top \mathbf{w} + b$. You should regularize \mathbf{w} but not b .

Instead of optimizing the weights of the network with the closed formula, use stochastic gradient descent (SGD). There are several different hyperparameters that you will need to choose:

- Mini-batch size \tilde{n} .
- Learning rate ϵ .
- Number of epochs.
- L_2 Regularization strength α .

In order not to cheat (in the machine learning sense) – and thus overestimate the performance of the network – it is crucial to optimize the hyperparameters **only** on a *validation set*. (The training set would also be acceptable but typically leads to worse performance.) To create a validation set, simply set aside a fraction (e.g., 20%) of the `age_regression_Xtr.npy` and `age_regression_ytr.npy` to be the validation set; the remainder (80%) of these data files will constitute the “actual” training data. While there are fancier strategies (e.g., Bayesian optimization – another probabilistic method, by the way!) that can be used for hyperparameter optimization, it’s common to just use a grid search over a few values for each hyperparameter. In this problem, you are required to explore systematically (e.g., using nested `for` loops) at least 4 different parameters for each hyperparameter.

Performance evaluation: Once you have tuned the hyperparameters and optimized the weights so as to minimize the cost on the validation set, then: (1) **stop** training the network and (2) evaluate the network on the **test** set. Report the performance in terms of *unregularized* MSE; put this number into the PDF document you submit.

4. **Logistic Sigmoid Identities** [8 points, on paper]: The logistic sigmoid function is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$.

- (a) Prove that $\sigma(-x) = 1 - \sigma(x) \quad \forall x$.
- (b) Prove that $\sigma'(x) = \frac{\partial \sigma}{\partial x}(x) = \sigma(x)(1 - \sigma(x)) \quad \forall x$.

Put your code in a Python file called `homework2-WPIUSERNAMES.py`. For the proofs, please create a PDF called `homework2-WPIUSERNAMES.pdf`. Create a Zip file containing both your Python and PDF files, and then submit on Canvas.