COMP.4220 - Machine Learning

Homework 3

student name:

---

1. In this question, we investigate the relationship between the sigmoid function and `tanh` that are used as basis functions.

   (a) Show that the `tanh` function and the logistic sigmoid function (i.e. $\sigma(a) = 1/(1+\exp(-a))$) have the following relationship:

   $$\tanh(a) = 2\sigma(2a) - 1.$$

   (b) Show that a general linear combination of logistic sigmoid functions of the form

   $$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \sigma(\frac{x-\mu_j}{s}),$$

   is equivalent to a linear combination of `tanh` functions of the form

   $$y(x, \mathbf{u}) = u_0 + \sum_{j=1}^{M} u_j \tanh(\frac{x-\mu_j}{2s}).$$

   (c) Show the relationship between the new parameters $\{u_i, \ldots, u_M\}$ to the original parameters $\{w_i, \ldots, w_M\}$.

(10 marks)

2. (Extra Credit) Given a data set where each data point t_n is associated with a weighting factor $r_n > 0$, so that the sum-of-squares error function becomes

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

   (a) Find the expression for the solution $\mathbf{w}^*$ that minimizes this error function (**Hint:** solve using maximizing the derivative of the weighted error function).

   (b) Give two alternative interpretation of the weighted sum-of-squares error function in terms of

      I. data dependent noise variance (**Hint:** compare with the unweighted error functions), and

      II. replicated data points

(4 marks)

3. (Programming) In this section, we investigate linear regression models on a publicly available data set.

   (a) Install the `Scikit-learn` library by following instructions form this link `https://scikit-learn.org/stable/install.html`. In most cases, you should be able to install using

   ```
   pip install scikit-learn
   ```

   After installing the library, you can fetch the data set by adding this line in your code

   ```
   from sklearn.datasets import fetch_california_housing
   ```

   Now you can fetch the data set and check a few of its properties. Let's start by checking the size of the observations and targets. This can be done as follows:

   ```
       housing = fetch_california_housing()
       print(housing.data.shape, housing.target.shape)
       print(housing.feature_names[0:6])
   ```

   You should be able to see the size (and shape) of the observations and target points as $(20640, 8)$ and $(20640,)$, respectively. In other words, the data set includes 20640 observations, each of which represented with 8 features. The last line of code prints the name of the features. Finally, print out and read the description of the data set using the following line of code:

   ```
       print(housing.DESCR)
   ```

   Include the previous four lines of code in the `main` function in the `hw3.py` file. Then assign observations to a variable X and targets to t.

   (b) Now that we have the observations and targets, we need to standardize our data using **mean removal and variance scaling**. Standardization of datasets is a common requirement for many machine learning estimators since they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance. In the `hw3.py` file, complete the `standradscalar` function by subtracting the mean of the input array and dividing the result by the standard deviation as

   $$x_{\text{scaled}} = \frac{x - \mu}{\sigma}$$

   After completing the function, call it in the `main` function and standardize the observations. Make sure that the standardization is being applied along the intended axis.

   (c) The standardization process has been implemented in the `scikit-learn` library as well. In the `main` function, use the `StandardScaler` utility class from the `Preprocessing` module (import with a different name to avoid conflicts) and standardazie your raw data again, but this time assign the results to a different variable. Write a simple test to show that results from both functions (your implementation of `standradscalar` vs. the `StandardScaler` class) are identical. This can be done by using `numpy` functions such as `all, allclose` and `array_equal` as well as comparing their `mean` and `std` values.

(d) In this step, we would like to split our data set into a training and a test set. To do so, first you should complete the `train_test_split` function. The first step is to shuffle the data. To do this, you can apply the `numpy.shuffle` function to the indices of the data set. Then use the shuffled indices to assign the points to new arrays for both the observations and the targets. The second step is to split the data by assigning the first portion of the data points (e.g., the first 80% of the points) to the training set and the rest to the test set. Finally, call the completed function in the `main` function and return $X_{\text{train}}, X_{\text{test}}, t_{\text{train}}, t_{\text{test}}$. Note that the same process can be done using the `train_test_split` function from the `model_selection` module in `scikit-learn`. Verify this but make sure you avoid naming conflict.

(e) Complete implementation of Ridge Regression class in the `regression.py` file. The file already includes the Least Squares that we used in Homework 1. First complete the `fit` function using the solution provided in (3.28) in the textbook. Hint: this can be solved as an $Ax = b$ equation without inverting A directly.

(f) Finish implementation of Ridge Regression class in the `regression.py` file by completing the `predict` function.

(g) In the `main` function in `hw3.py`, use the `RidgeRegression` and `LinearRegression` classes to train two regression models on the training set. Then fit the trained model to the test set.

(h) In a previous homework you implemented the root-mean-squared-error. The `Scikit-learn` library also include many evaluation metrics. Import the R^2 and `root_mean_squared_error` metrics and use them to evaluate the models you trained. Print the results.

(i) In this step, you are comparing the results of your regression models with the regression models from `Scikit-learn`. Import the `LinearRegression` and `Ridge` classes from the library and train two models using the same data you used for training the previous models. Then evaluate the models and print the results.

(j) In a 2×2 plot, use the `scatter` function to plot targets t_{test} vs. $y(x_{\text{test}})$ for each model. On each subplot add a line (using the `plot` function) starting from minimum target value to the maximum target value. This line gives us a view of how predictions are distributed around the 'mean' of the data.

(k) If you implement the regression correctly, you should get identical results from all four models. If your models are predicting worse than the `Scikit-learn` model, can you find out why and how to fix it? Explain your fix and show identical results.

(l) Since all models are preforming identically, what does this mean for the Ridge regression performance. Can you find a regularization factor λ that improves the behavior of the model? Explain your answer.

(10 marks)