CMPSCI 390A Homework 4

YOUR NAME HERE

Assigned: Feb 19 2018; Due: Feb 25 2018 @ 11:00 am ET

Abstract

This assignment will cover two important aspects of fitting linear models to data: normalization and basis functions. To submit this assignment, upload a `.pdf` to Gradescope containing your responses to the written response questions below. You are required to use LaTeX for your write-up. When submitting your answers, use the template LaTeX code provided and put your answers below the question they are answering. Do not forget to put your name on the top of the .pdf. To submit the assignment's coding portion, upload a single python file called `my_hw4.py`, to the Gradescope programming assignment for Homework 4. An auto-grader will check your code for the correct output. As such, your program must meet the requirements specified below. We will also be using cheating detection software, so, as a reminder, you are allowed to discuss the homework with other students, but you must write the code on your own.

1 Feature Normalization (40 points)

In the last assignment, you were asked to find the optimal step-size α to make gradient descent converge quickly. In general, this a challenging problem, but it can be even more difficult when features are scaled differently, e.g., $x_{i,j} \in [0, 1]$ and $x_{i,k} \in [100, 1{,}000]$. This makes choosing a learning rate more difficult because a gradient step for parameter w_j is proportional $x_{i,j}$, which is much less than the change to w_k, which is proportional to $x_{i,k}$. A simple method to alleviate this issue is to modify the features, so they are all on the same scale. In the previous assignment, we gave you a data set after we rescaled the features. In this part of the homework, you will write code to normalize the features using several standard normalization techniques and then perform gradient descent using features created with different normalization techniques.

We have provided template code for normalizing the data and performing gradient descent in `my_hw4.py`. First, you should implement the following functions to normalize the data:

- `normalize_gaussian(X)` returns a normalized set of features X' and makes it, so each column of X' has a mean of 0 and standard deviation of 1. The per feature normalization operation is:

$$x'_{i,j} = \frac{x_{i,j} - \bar{x}_j}{\sigma_j},$$

  where $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}$ and $\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{x}_j - x_{i,j})^2}$

- `normalize_01(X, low, high)` maps each feature to be in the range $[0, 1]$ using each feature's min and max values. This normalization operation is

$$x'_{i,j} = \frac{x_{i,j} - a_j}{b_j - a_j},$$

  where a_j and b_j are the upper and lower bounds on $x_{i,j}$.

- `normalize_posneg(X, low, high)` maps each feature to be in the range $[-1, 1]$ using each feature's min and max values. This normalization operation is:

$$x'_{i,j} = \frac{x_{i,j}}{b_j - a_j}.$$

In each of the above functions, the normalization technique should be applied to the entire data set. See the comments in the template file for exact return types and inputs. For the low and high values, you should use the values we provided in the template code. Additionally, in the last assignment we added a bias feature, a feature that is always 1, to every row in the data set, i.e., $\forall i\ x_{i,1} = 1$. For this assignment, you will implement the function `add_constant(X)`, which will return a new set of feature vectors X' with $m + 1$ columns and where the first column is all ones. An autograder will check each of the above four functions and make the grade visible before the deadline.

After implementing these functions, you will then compare gradient descent's performance using each of these methods and not using any normalization method. To do this you can run the functions `original_fit`, `meanzero_fit`, `zeroone_fit`, and `posneg_fit`. We have already provided implementations of these functions for you. However, you must tune the step-sizes for each method. These functions have the same minimum loss value, but each takes different amounts of time to reach

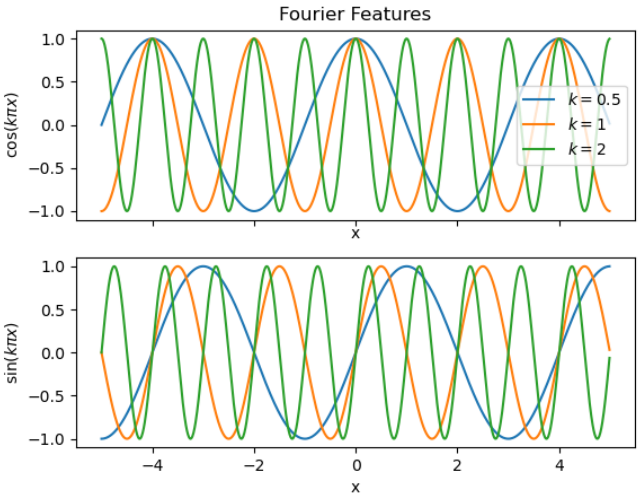

Figure 1: Example Fourier basis functions.

it. You should be able to find a step-size so that for each normalization method, except for the original fit, gradient descent converges to have the same loss value (approximately 13,000) by the 10,000$^{\text{th}}$ iteration. For the `original_fit` function the loss should be below 14,000. When reporting the results, you need to use $N = 10{,}000$ iterations for each method, but you can lower it to tune the step-sizes quickly. You will report the step-sizes and learning curves for each method in a written report. This is discussed in detail in Section 3

2 Basis Expansion (40 points)

Linear models are often useful tools, but their prediction power is limited to the linear trends in each feature. To make the model able to represent more complicated functions, new features can be created that use nonlinear transformations of the original features. These transformation functions are often referred to as basis functions. For this assignment, you will implement two common basis functions: the polynomial basis and Fourier basis.

The polynmoial basis function, ϕ_p, transforms features by taking them to a higher power, e.g., $x_{i,j}^2, x_{i,j}^3$. A k^{th} order polynomial should create the following features,

$$\phi_p(x_i) = [x_{i,1}, x_{i,2}, \ldots, x_{i,m}, x_{i,1}^2, x_{i,2}^2, \ldots, x_{i,m}^2, \ldots, x_{i,1}^k, x_{i,2}^k, \ldots, x_{i,m}^k].$$

The Fourier basis function, ϕ_f, transforms features by applying trigonometric functions to make features that spin around the unit circle at different frequencies, e.g., $\cos(k\pi x_{i,j}), \sin(k\pi x_{i,j})$. The coefficient, k applied to π determines how fast the features go around the unit circle as $x_{i,j}$ changes. These features are visualized in Figure 1. For this assignment, you will use the following for a k^{th} order Fourier basis:

$$\phi_f(x_i) = [\cos(\pi x_{i,1}), \cos(\pi x_{i,2}), \ldots, \cos(\pi x_{i,m}), \cos(2\pi x_{i,1}), \ldots, \cos(2\pi x_{i,m}), \ldots, \cos(k\pi x_{i,1}), \ldots, \cos(k\pi x_{i,m})].$$

In the template code we define the functions `fourier_basis(X, order)` and `polynomial_basis(X, order)`, which you will use to implement the basis functions. Each function should return the corresponding set of features defined above for each feature vector in X. See the comments in the code for output specifications. An autograder will check these functions, but not display the results until after grades are published.

After implementing these functions, you will compare the loss using normalized features to the loss using these basis functions. To conduct this comparison, you should implement the functions `fourier_fit` and `polynomial_fit`. Each function should normalize the data set, create new features using the basis function on the normalized data, add then run gradient descent using the expanded feature set X''. The expanded features set X'' should include the bias term, normalized features, and features from the basis function, e.g.,

$$x_i'' = [1, \phi_p(x_i')] \text{ and}$$
$$x_i'' = [1, x_{i,1}', \ldots, x_{i,m}', \phi_f(x_i')].$$

Note that in the expanded feature vectors for the polynomial fit, the features $x_{i,j}'$ are already include in $\phi_p(x_i')$. For each method, you need to determine which normalization method to use, the order of the basis function, and the step-size. The Fourier fit and polynomial fit's final loss should be below 10,000 and 11,000 respectively.

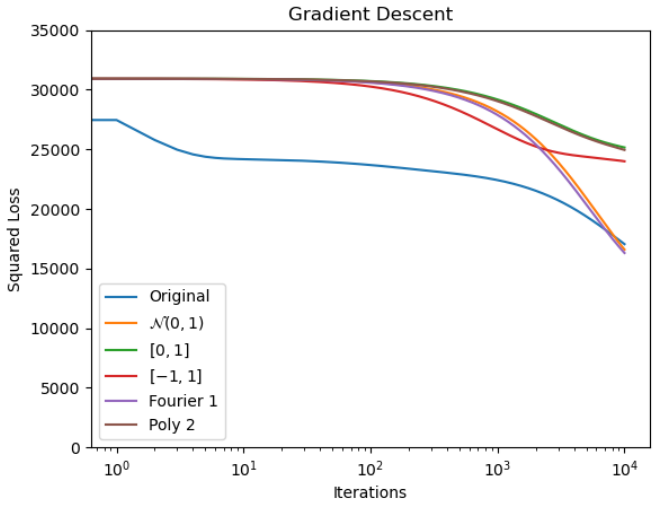

Figure 2: Learning curve plot with default hyperparameters.

3 Written Report (20 points)

In addition to submitting your code, you should submit a pdf to gradescope containing a table of the final performance values for each method, corresponding hyperparameters, and the learning curve plot output by main. When your code is correct, but hyperparameters are left as defaults, the learning curve should look like the plot shown in Figure 2. Like the previous homework, we have provided a table and figure placeholders in `main.tex` that you should use. For the results of Fourier fit and polynomial fit, in addition to the step-size, you need to specify the normalization method and order you used.

After conducting the experiments, you should respond to the following questions. Your responses will not be graded based on whether the answers are "correct". Instead, they will be graded based on whether you present reasonable arguments and show that you thought about the questions.

1. (5 points) Which normalization learns the quickest? Why might this method be superior in this problem? Your response should be less than approximately 250 words.

2. (5 points) The Fourier basis and polynomial basis cannot achieve the level of performance that k-NN did in the previous assignment. What features can be added to improve the fit of these models further? (less than approximately 250 words).

4 Extra Credit (15 points)

In your experiments, you should have noticed that each normalization method has the same minimum loss. However, this observation only indicates that it is true on this data set. To confirm that it is always true, we will ask you to prove the following Theorem.

Consider the sets of feature vectors X and X', where $\forall i$ and $x_{i,1} = x'_{i,1} = 1$, and $\forall j \in \{2, 3, \ldots, m\}$, $x'_{i,j} = (x_{i,j} - a_j)/b_j$, where $a_j \in \mathbb{R}$ and $b_j \in \mathbb{R}^+$, i.e., $b_j > 0$. Let w be the weights of a linear model f_w.

Theorem 1. *There exists a weight vector w' such that $\forall i$ $f_w(x_i) = f_{w'}(x'_i)$.*

Proof. **Hint: Construct a vector w' such that each weight w'_j only contains terms of w, a, and/or b.**
Let $w'_1 =?$, $w'_2 =?, \ldots$

$$
\begin{aligned}
f_{w'}(x'_i) &= \\
&\vdots \\
&= f_w(x_i)
\end{aligned}
$$

□