In the Name of God

Machine Learning

Homework IV

Feature Maps, Kernels, and Regularization Factors in Linear Regression

Submission Deadline: **One week after your final exams are over**

Contents

Precautions

- Any programming language is allowed, preferably python.
- A report is required with the following format:

  Ch.1 Problem Definition

  Ch.2 Solution methodology and implementation details.

  Ch.3 Results and discussion. (answer the homework questions here)
- Your submission will be a zip file containing your code files and your report. Naming format is "your names.zip" for example "Achraf_Hakimi_Mohamed_Salah.zip".
- Late submission strategy is: 70 percent score for one day delay and 50 percent for 2 days delay. Submissions after 2 days will not be graded.
- Do not copy other works, write your own code. Good luck!

1 Feature Maps

In this section we are going to familiarize ourselves with feature maps in linear regression, see whether they improve model performance or not and why (feature maps are not just used in regression tasks, so you may get the idea here and use it in your own problem).

1.1 Data: Generate your Dataset

Generate your dataset running the notebook generate_synthetic_data.ipynb shared with you. This will give you a training and test set of features and labels. Then answer the following questions about your dataset:

- What's the feature dimension?
- What's the mapping (function) between input features and output label? (the goal in regression is to estimate this mapping as close to the true mapping as possible)
- Does this dataset need to be normalized? or is it already?
- * Note that you won't be able to know the true mappings in any real task.

1.2 Model: Linear Regression

Implement the following functions. You could implement either the closed form regression (refer to the Linear Regression lecture thought in class: slide 21) as your backbone algorithm or a gradient-decent-based one (refer to the Linear Regression lecture thought in class: slide 36), its up to you.

find_regression_coefficients(X, y) Gets your training samples (features and labels) and returns the coefficients of your estimated linear regression line.

predict_target_values(X_test, coefficients) Gets test samples (only features) and the regression line coefficients you found, and returns predicted labels for those test samples.

calculate_mse(y_actual, y_predicted) Gets predicted and actual labels and returns the Mean Squared Error (MSE) between them.

1.3 Tasks and Questions

Walk through these four steps for each of the specified feature maps below:

1. Fit a linear regression line to your training features and print out the estimated line coefficients.
2. Using the line coefficients you just found, predict target values for the test set.
3. Calculate and report MSE between predicted and actual target values of your test set.
4. Plot the predicted and actual target values of your test set in one figure and use different colors for each (simply try pyplot.plot(y)).

Feature Map 1 $\boldsymbol{\phi_1(x_1, x_2, x_3) = [x_1, x_2, x_3]^T}$
Or no feature mapping.

Feature Map 2 $\boldsymbol{\phi_2(x_1, x_2, x_3) = [1, x_1, x_2, x_3]^T}$
See how adding the bias term is a type of feature mapping.

Feature Map 3 $\boldsymbol{\phi_3(x_1, x_2, x_3) = [1, \sin(2\pi x_1), \sin(2\pi x_2), \sin(2\pi x_3), \cos(2\pi x_1), \cos(2\pi x_2), \cos(2\pi x_3)]^T}$
Because we love Fourier features.

Feature Map 4 Use equation 4 from the paper shared with you.
Here is a question of how many Fourier features to use. Apply this mapping over all of your features $\boldsymbol{x_1, x_2, x_3}$. Find the optimum L value which is when you reach the minimum MSE on test set. To do so plot MSE of your test set in case of using different L values ranging from 0 to 30.

Feature Map 5 $\boldsymbol{\phi_5(x_1, x_2, x_3) = [1, \log x_1^2, \log x_2^2, \log x_3^2]^T}$
Is this feature mapping familiar?

Note that once you apply a features mapping, your feature matrix will look something like this, where $\boldsymbol{x^i}$ for $\mathbf{1} < \boldsymbol{i} < \boldsymbol{n}$ is the i-th sample in the dataset:

$$\boldsymbol{X} = \begin{bmatrix} \phi(\boldsymbol{x^1})^T \\ \phi(\boldsymbol{x^2})^T \\ \cdots \\ \phi(\boldsymbol{x^n})^T \end{bmatrix}$$

Finally answer the following questions:

- Compare the plots of predicted and actual target values, for the first and second feature maps. Can you see the effect of a bias term?
- As we plot the target values we see a nonlinear function, Why do we still call the algorithm a **linear** regression? linear in terms of what?
- Which feature map resulted in a less MSE on test set? why?
- What is the optimal value for L (or the number of Fourier features)? Can you interpret why higher L values lead to an increase in MSE?
- The reference paper for feature map 4 is also doing a regression task, how did this feature map help this paper? Can you see what is the regression task of this paper (what is the input and output)?
- Why can a feature map, if chosen properly, boost the performance of a linear regression model?

2 Kernels and Regularization

You derived a kernel version of linear regression in class using regularized least squares criterion. In this section we are going to see what a kernel is and whether it improves performance of a linear regression model. To see the resulting dual from you can refer to one of the following sources:

- Kernel Regression lecture thought in class: slide 15
- Pattern Recognition and Machine Learning, C. M. Bishop: section 6.1, equations 6.8 and 6.9

2.1 Data: Generate your Dataset

Your dataset for this section is the same as that of the feature map section.

2.2 Model: Linear Regression (Dual Form)

Implement the following functions for the dual form of linear regression.

rbf_kernel(x1, x2, beta) Gets two samples (x^1, x^2) and a parameter β to calculate the Radial Basis Function kernel between them (also referred to as Gaussian kernel) and return the kernel value. Here is the RBF kernel, use $\beta = 20$:

$$K(x^1, x^2) = \exp(-\frac{\beta}{2}||x^1 - x^2||^2), \beta > 0$$

gram_matrix(X, kernel_parameter) Gets a feature matrix and kernel parameters (in case of a RBF kernel this would be β) and returns the Gram matrix.

find_regression_coefficients(X_train, y_train, regularization_coeff, kernel_parameter) Gets your training samples (features and labels), the regularization term, and the kernel parameter. Returns the coefficients of your estimated linear regression line.

predict_target_values(X_train, X_to_predict_for, coefficients, kernel_parameter) Gets training samples (only features), the test samples for which you want to predict labels, the regression line coefficients you found, and the kernel parameters. Returns predicted labels for those test samples.

calculate_mse(y_actual, y_predicted) Gets predicted and actual labels and returns the Mean Squared Error (MSE) between them.

2.3 Tasks and Questions

You've got three main tasks here. Once done with the task, don't forget to answer the questions as well:

- First we want to find the optimum value for the regularization term. Therefor, try different regularization values in a range of 0 to 1, fit a linear regression model for each value, and predict labels for training and test sets. Then in one figure, plot MSE over training and test sets in different colors versus the regularization term. This plot will tell you what value to choose for regularization.
- Now fit a regression model using the regularization value you found and predict labels for test set. Then plot predicted and actual labels for test set in one figure. Also report the MSE on test set. How good are the predictions?
- We tried the Gaussian kernel so far, try 3 other kernel and see if they also improve model performance.

Now here are the questions:

- Why did we use kernels in this homework? What does a kernel function do really?
- Were all the kernels effective? Which ones were effective? How can we choose the right kernel for an application?
- Compare all the experiments in this homework (including different feature map and kernel settings) in terms of time and memory complexity. Which one has the highest memory/time complexity?

- Can we both apply a feature map and a kernel at the same time? design an experiment and implement it to prove whether these two techniques can be applied at the same time to improve performance of a linear regression model or not.