Assignment 2: Regularization and Model Evaluation

COMP 551, Fall 2024, McGill University
Contact TAs: Rahma Nouaji and Rafid Saif

Please read this entire document before beginning the assignment.

Preamble

- This assignment is **due on October 21st at 11:59pm (EST, Montreal Time).** There is a penalty of 2^k percent for k days of delay, which means your grade will be scaled to be out of $100-2^k$. No submission will be accepted after 6 days of delay.
- **This assignment is to be completed in groups**. All members of a group will receive the same grade except when a group member is not responding or contributing to the assignment. If this is the case and there are major conflicts, please reach out to the Head TA for help and flag this in the submitted report. Please note that it is not expected that all team members will contribute equally. However every team member should make integral contributions to the assignment, be aware of the content of the submission and learn the full solution submitted.
- You will submit your assignment on MyCourses as a group. You must register your group on MyCourses and any group member can submit. See MyCourses for details.
- We recommend to use **Overleaf** for writing your report and **Google colab** for coding and running the experiments. The latter also gives access to the required computational resources. Both platforms enable remote collaborations.
- You should use Python for this assignment. You are free to use libraries with general utilities, such as numpy, pandas, scipy, and matplotlib for Python unless stated otherwise in the description of the task. **You could use all code provided in code-review**. In particular, in most cases you should implement the models and evaluation functions yourself, which means **you should not use pre-existing implementations of the algorithms or functions as found in scikit-learn, and other packages**. The description will specify this on a per-case basis.
- You should not use large language models (LLMs) to complete this assignment. Relying on them will hinder your learning, the purpose of the work is to practice the underlying skills yourself.

Background

In this assignment you will work with synthetic data to explore the details of linear regression, bias-variance trade-off, regularization with cross-validation and effect of L1 and L2 regularization on loss.

Task 1: Linear Regression with Non-Linear Basis Functions

In this task, you will generate synthetic data and fit a linear regression model using non-linear basis functions. The goal of this task is to explore how model complexity affects performance, helping you understand the transition from underfitting to overfitting.

Step-by-Step Instructions

1. **Data Generation:** Generate 100 data points sampled from the following non-linear function:

$$y(x) = (\log x + 1)\cos(x) + \sin(2x) + \epsilon$$

   where x is sampled uniformly in the range [0, 10]. And ϵ is sampled randomly and independently for each new data point from a Gaussian with mean 0 and variance 1 to simulate real-world noisy data. Plot both the clean data and the noisy data.

2. **Non-Linear Basis Functions:** Fit a linear regression model to the generated data, but instead of using the original features, transform them using Gaussian basis functions. Specifically, use Gaussian basis functions of the form:

$$\phi(x, \mu, \sigma) = \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$$

   where μ is the mean of the Gaussian and $\sigma = 1$ is the standard deviation. You will use basis functions with increasing complexity, ranging from 0 to 45 Gaussians, by varying the number of basis functions D. Please plot the Gaussian basis for range of generated data. **Note:** For μ, use: $\mu_i = x_{min} + \frac{x_{max}-x_{min}}{D-1}i,\ i \in [0, D-1]$. This is the mean for the i^{th} Gaussian.

3. **Model Fitting:** Fit a linear regression model using Gaussian basis functions with the following numbers of bases: 0, 5, ..., 45. For each model, compute the predictions and plot the fitted curves along with the original noisy data and the true function (the output of the given function for each noisy data point). This will help you visualize how the model complexity changes with the number of basis functions.

4. **Model Selection:** Split the data into a training set and a validation set. For each model (with different numbers of basis functions), compute the sum of squared errors (SSE) on both the training and validation sets. Identify the number of Gaussian bases that minimizes the validation SSE. This will help you choose the optimal model complexity by balancing underfitting and overfitting.

5. **Explanation:** After selecting the optimal number of Gaussian bases, explain your observations regarding the model's performance as the number of basis functions increases. Specifically, discuss the transition from underfitting (when the model has too few bases) to overfitting (when the model has too many bases), and how the validation set helps you select the right model.

6. **Additional Tips:** For tasks where you are comparing plots, it is helpful to keep the y-axis limits consistent. Plots can autoscale to your given data, but certain models can produce very large or very small values and this will change the way the entire plot looks and make it much more difficult to compare between them.

Minimum Deliverables

- Plot of the Gaussian basis for range of generated data.
- Plots of the fitted models for 10 different values of Gaussian basis functions (0, 5, ..., 45), showing both the noisy data and the true function.
- A table showing the sum of squared errors (SSE) for each model on both the training and validation sets, or you could show them in your figures.
- A brief report explaining the selection of the optimal model and your observations regarding underfitting and overfitting.

Task 2: Bias-Variance Tradeoff with Multiple Fits

In this task, you will explore the bias-variance tradeoff by repeating the fitting process from Task 1 multiple times and analyzing the variation in the fitted models.

Step-by-Step Instructions

1. **Repeating the Process:** Repeat the process from Task 1 a total of 10 times. Meaning, for each of the number of basis functions given, repeat the fitting process for 10 repitition. For each repetition, re-sample the 100 data points from the same distribution:

   $$y(x) = (\log x + 1)\cos(x) + \sin(2x) + \epsilon$$

   where ϵ is Gaussian noise. Each time, apply the same procedure of fitting linear regression models using Gaussian basis functions with different numbers of bases.

2. **Plotting Multiple Fits:** For each of the 10 repetitions, plot the fitted models on the same plot. Specifically:

   - Plot all 10 model fits using green lines.
   - Plot the true underlying distribution (the ground truth) using a blue line.
   - Plot the average of the 10 fitted models using a red line.

   These plots will allow you to visualize the bias and variance of the model as the number of Gaussian bases increases.

3. **Plotting Train and Test Errors:** For each of the 10 repetitions, compute both the training error and test error, and plot their average.

4. **Explaining Bias and Variance:** Based on the plots, explain your observations regarding bias and variance.

Minimum Deliverables

- Plots of bias-variance tradeoff figure and your explanation.

Task 3: Regularization with Cross-Validation

In this task, you will implement and apply both L1 (Lasso) and L2 (Ridge) regularization to a linear regression model and use 10-fold cross-validation to determine the optimal regularization strength (λ).

Step-by-Step Instructions

1. **Adding Regularization:** Modify the existing linear regression model with Gaussion basis number 45, to include both L1 and L2 regularization. You should allow the model to switch between L1 and L2 regularization based on input parameters. Hints: In order to plot Bias-Variance decomposition graph properly, you need to sample many datasets from the ground truth distribution and then train a model on each dataset.

2. **Cross-Validation:** Use 10-fold cross-validation to evaluate a range of regularization strengths (λ) for both L1 and L2 regularization. For each λ, you will:

   - Split the training data into 10 subsets (folds).
   - Train the model on 9 folds and validate on the remaining fold.
   - Repeat this for each fold and compute the mean squared error (MSE) on both training and validation sets.

3. **Plotting Train and Validation Errors:** For each regularization method (L1 and L2), plot the train error and test error as a function of λ. You could use the parameters settings similar to $\lambda = np.logspace(-3, 1, 10)$, num_datasets=50, N=20 (sampling number), D=45 (basis number), noise_variance=1, num_folds=10

4. **Plotting Bias-Variance Decomposition:** In addition to plotting train and test errors, you will also calculate and plot the bias squared ($(\text{Bias})^2$), variance, and their sum $(\text{Bias})^2 + \text{Variance}$, $(\text{Bias})^2 + \text{Variance} + \text{noise\_variance}$,

test errors for both L1 and L2 regularization. You could use a log scale for the λ axis to visualize the changes in these metrics more clearly. You could use the similar parameters settings as above. If you find it too difficult to include k-fold into this plot, you could draw the the Bias-Variance Decomposition without k-fold logic, and the plot with k-fold logic is an optional task which could be used as a part of creativity work.

5. **Selecting the Optimal λ:** Based on the validation error plot, choose the λ value that minimizes the validation error. Discuss how different values of λ affect the model's performance in terms of bias and variance, using the bias-variance decomposition plot for insights.

Minimum Deliverables

- Plots showing train and validation errors vs. λ for both L1 and L2 regularization.
- Plots showing bias squared, variance, and $(\text{Bias})^2 + \text{Variance}$, $(\text{Bias})^2 + \text{Variance} + \text{noise\_variance}$, as a function of λ, test error for both L1 and L2 regularization.
- Analysis of the results, explaining how the choice of λ affects the model performance in terms of bias and variance.
- Selection of the optimal λ for both L1 and L2 regularization.

Task 4: Effect of L1 and L2 Regularization on Loss

In this task, you will explore the effects of L1 and L2 regularization on linear regression models by fitting models with varying regularization strengths, plotting the corresponding loss contours, and visualizing the paths taken by gradient descent.

Step-by-Step Instructions

1. **Generating Synthetic Data:** Use the following procedure to generate synthetic data that follows a linear relationship with Gaussian noise:

$$y = -3x + 8 + 2\epsilon$$

   where ϵ is Gaussian noise with a mean of 0 and variance of 1. Generate 30 data points with x uniformly distributed between 0 and 10.

2. **Applying L1 and L2 Regularization:** Implement Ridge (L2) and Lasso (L1) regression using the `GradientDescent`.
3. **Plotting the Loss Function:** For each regularization type (L1 and L2) and different regularization strengths, plot the loss contours to visualize the optimization landscape. Use the function `plot_contour` to generate these plots:
4. **Visualizing Gradient Descent:** For each regularization strength, plot the trajectory of the gradient descent optimizer on top of the contour plot. This will help you visualize how the regularization affects the optimization process.
5. **Analyzing the Effects of L1 and L2 Regularization:** Based on the plots, observe and compare the following:
   - How L1 regularization encourages sparsity (setting some weights to zero).
   - How L2 regularization penalizes large weights but does not promote sparsity as strongly as L1.
   - How different values of λ affect the optimization paths and loss landscapes.

Minimum Deliverables

- **Contour Plots:** Contour plots of the loss function for L1 and L2 regularization with varying strengths.
- **Optimization Paths:** Plots showing the trajectory of gradient descent for different regularization strengths.
- **Analysis:** Based on the plots, provide observations about the impact of L1 and L2 regularization on the model's weights and performance.

Deliverables

You must submit two separate files to MyCourses.

1. **code.ipynb or code.zip**: Your code. Please keep all your running results in the code.ipynb file or in your code folder. Submit your code results separately along with your final report. Ensure that the **results in your Colab/.ipynb file match those in your report.**
2. **writeup.pdf**: Your (max 7-page, excluding references, appendix) write-up as a pdf.

Assignment write-up

Your team must submit a assignment write-up that is a **maximum of** 7 **pages** avaibale template could be find at **Overleaf Assignment Template** (single-spaced, 11pt font or larger; minimum 0.5 inch margins, excluding references (if any). We highly recommend that students use LaTeX to complete your write-ups and you could share it with your groupmates via share project on overleaf. You have some flexibility in how you report your results, and you could follow the structure and minimum requirements listed below:

Abstract ($100 - 250$ **words)** Summarize the assignment task and your most important findings.

Task 1-4 Your deliverables results as described in task 1-4.

Originality/creativity ($3+$ **sentences, possibly with figures or tables)** Describe what you have done to go beyond the bare minimum requirements and your findings). Such as trying other synthetic functions, other noise levels, use other non-linear basis, add k-fold in your Bias-Variance Decomposition plot.

Discussion and Conclusion ($5+$ **sentences)** Summarize the key takeaways from the assignment and possibly directions for future investigation.

Statement of Contributions ($1-3$ **sentences)** State the breakdown of the workload across the team members.

Evaluation

This assignment is out of 100 points, and the evaluation breakdown is as follows:

- Task 1: 25 points
  - Did you correctly generate the required data and produce accurate plots? (3 points)
  - Did you generate a proper Gaussian basis plot? (2 points)
  - Is your model fitting approach correct, and do the resulting curves match expectations? (10 points)
  - Did you provide a complete and accurate Sum of Squared Errors (SSE) table or plot? (5 points)

  - Did you clearly explain the concepts of underfitting and overfitting with appropriate examples from your results? (5 points)

- Task 2: 20 points
  - Did you successfully repeat the fitting process with the required variations? (5 points)
  - Did you plot multiple fits correctly and ensure accuracy in visual presentation? (5 points)
  - Are the train and test errors plotted correctly, and do they support your analysis? (5 points)
  - Did you clearly explain the concepts of bias and variance based on your experimental results? (5 points)
- Task 3: 30 points
  - Did you correctly implement L1 and L2 regularization, integrate them effectively into your model, and show the result? (10 points)
  - Did you use cross-validation correctly to assess your model's performance? (5 points)
  - Are the train and test errors plotted accurately, and do they demonstrate the effects of regularization? (5 points)
  - Did you successfully plot the bias-variance decomposition and explain its significance? (5 points)
  - Did you correctly select the optimal λ and justify your choice based on your results? (5 points)
- Task 4: 15 points
  - Did you generate contour plots of the loss function that are accurate and informative? (5 points)
  - Did you successfully trace and analyze optimization paths in your model? (5 points)
  - Did you provide a thoughtful analysis of the effects of regularization on model performance? (5 points)
- Writing, code quality, and novelty: 10 points
  - Is your code clean, efficient, and well-organized? (4 points)
  - Is your report clear, well-written, and free from grammatical errors? (3 points)
  - Does your work demonstrate creativity or novelty in approach, beyond the basic requirements? (3 points)

Final remarks

You are expected to display initiative, creativity, scientific rigour, critical thinking, and good communication skills. You don't need to restrict yourself to the requirements listed above. Feel free to go beyond, and explore further. You can discuss methods and technical issues with members of other teams, but **you cannot share any code or data with other teams.**