CS 1678/2078 Homework 1

Abstract

This assignment is the first introduction to doing gradient based optimization. Specifically, it focuses on solving regression and classification problems using linear function approximation with and without a basis function. In this assignment, you will solve analytically for the partial derivative of a loss function with respect to the model parameters (weights), and you will begin the basic setup of a program to perform linear regression on a provided data set. To submit this assignment, upload a `.pdf` to Gradescope containing your responses to the questions below. You are required to use LaTeX for your write up. To submit the assignment's coding portion, upload a zip folder to gradescope containing all python files. This code will be used to verify your answers and check for plagiarism. We will be using cheating detection software, so, as a reminder, you are allowed to discuss the homework with other students, but you must write your code on your own. You may also not use ChatGPT, Co-Pilot, or any other AI software to write your answers or code.

1 Written Responses

Consider approximating the function

$$f_*(x) = 6x + 4\cos(3x + 2) - x^2 + 10\ln\left(\frac{|x|}{10} + 1\right) + 7$$

with the function approximator $f(x, w) = \phi(x)^\top w$, where $\phi\colon \mathbb{R} \to \mathbb{R}^n$ and $w \in \mathbb{R}^n$, i.e., $\phi(x)$ creates a feature vector of length n and the weights w are a vector length n.

1. (3 points) What are features that allow f to exactly represent f_*? Specify the features $\phi(x)$.
   Answer:
   $$\phi(x) = [?]^\top$$

   $$\top$$

2. (2 points) What are the optimal weights for this approximation? $\Rightarrow$ $[1, X, X^2, \cos(3X), \sin(3X),$
   Answer:
   $$w^* = [?]^\top \Rightarrow [7, (6\frac{|X|}{10}|+4)\cos(2), -4\sin(2), 10]^\top$$

3. (3 points) What would be the basis function and weights to perfectly represent the function

   $$f_*(x) = 6x \times 4\cos(3x + 2) \times x^2 \times 10\ln\left(\frac{|x|}{10} + 1\right) \times 7?$$

   $$\ln\ln\left(\frac{|X|}{10} + 1\right), 1$$

   Answer:
   $$\phi(x) = [?]^\top \Rightarrow [\ln x, \ln(\cos(3X + 2), \ln(x^2),]$$
   $$w^* = [?]^\top \Rightarrow [1, 1, 1, 1, \ln(6\times4\times10\times7)]^\top$$

4. (3 points) In the programming homework below we will be making predictions for multiple data points at a time. More specifically, we will consider linear function approximation of the form $f\colon \mathbb{R}^{m \times n} \times \mathbb{R}^n \to \mathbb{R}^m$, where f takes as input a matrix $X \in \mathbb{R}^{m \times n}$ that represents a m data points each being a row vector of n features, a vector $w \in \mathbb{R}^n$ that represents the parameters of the function, and maps these to a vector $\hat{y} \in \mathbb{R}^m$ that represents a prediction for m data points, i.e.,
   $$\hat{y} = f(X, w) = Xw.$$

   Let $y \in \mathbb{R}^m$ represent the target (or label) for data point, i.e. y_i and $\hat{y}_i$ are the ith labels and prediction, respectively. Consider the mean squared error loss function $g\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ for each batch of predictions, e.g.,

   $$g(\hat{y}, y) = \frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2.$$

What is the partial derivative of g with respect to the predictions? Note the derivative should be a vector in $\mathbb{R}^m$.

Answer:

$$\frac{\partial}{\partial \hat{y}} g(\hat{y}, y) = \frac{\partial}{\partial \hat{y}} \frac{1}{2} \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

$$= ?$$

This evaluated the expected value of the difference $(y - \hat{y})$

5. (3 points) What is the partial derivative of $f(X, w)$ with respect to w. Write your answer using matrices and/or vectors.

Answer:

$$\frac{\partial}{\partial w} f(X, w) = \begin{bmatrix} \frac{\partial f(X,w)_1}{\partial w} & \frac{\partial f(X,w)_2}{\partial w} & \cdots & \frac{\partial f(X,w)_m}{\partial w} \end{bmatrix}$$

$$= ?$$

$$\begin{bmatrix} X_1 & X_2 & \cdots & X_m \end{bmatrix} \leftarrow \text{Add this to code}$$

6. (2 points) Consider the loss function $l(w) = g(f(X, w), y)$. What is the gradient of l? Express your answer using matrices/vectors without summations. Note that we did this derivation in class but used summations.

Answer:

We have discussed the gradient for the loss function in the case of regression. Now ↓ classification

$$\nabla l(w) = \frac{\partial}{\partial w} g(f(X, w), y)$$

$$= ?$$

$$\frac{1}{2m} \frac{\partial}{\partial w} (X_i w_i - Y_i)^2$$

$$= \frac{1}{m} (X_i w_i - Y_i) * X_i$$

$$= \frac{1}{m} X_i \hat{Y}_i - X_i Y_i$$

7. (4 points) Consider the NLL loss function for classification, $g(\hat{y}, y) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \ln \sigma(\hat{y}_i) + (1 - y_i) \ln (1 - \sigma(\hat{y}_i))]$, where σ is the sigmoid (also called the logistic function). What is the gradient of the loss function $l(w) = g(f(X, w), y)$? Express your answer using matrices/vectors without summations. Note that we did this derivation in class but used summations.

Answer:

$$\nabla l(w) = \frac{\partial}{\partial w} g(f(X, w), y)$$

$$= ? \quad \frac{d}{dw} \left[-\frac{1}{m} \sum_{i=1}^{m} \left[y_i \ln \sigma(\hat{y}_i) + (1 - y_i) \ln (1 - \sigma(\hat{y}_i)) \right] \right]$$

$$= -\frac{1}{m} \sum \frac{\partial}{\partial w} \left[y_i \ln \sigma(X_i w_i) + (1 - y_i) \ln (1 - \sigma(X_i w_i)) \right]$$

$$= -\frac{1}{m} \sum \frac{\partial}{\partial w} \left[y_i \ln \sigma(X_i w_i) \right] + \frac{\partial}{\partial w} \left[(1 - y_i) \ln (1 - \sigma(X_i w_i)) \right]$$

$$= -\frac{1}{m} \sum \frac{\partial}{\partial w} \left[y_i \ln \left(\frac{1}{1 + e^{-(X_i w_i)}} \right) \right] + \frac{\partial}{\partial w} \left[(1 - y_i) \ln \left(1 - \frac{1}{1 + e^{-X_i w_i}} \right) \right]$$

$$= \frac{-1}{m} \sum y_i \frac{\partial}{\partial w} \ln \left(\frac{1}{1 + e^{-X_i w_i}} \right) + (1 - y_i) \frac{\partial}{\partial w} \ln \left(1 - \frac{1}{1 + e^{-X_i w_i}} \right)$$

$$= -\frac{1}{m} \sum \left[y_i \frac{\partial}{\partial w} \ln (1 + e^{-X_i w_i}) + (1 - y_i) \frac{\partial}{\partial w} \ln \left(\frac{-1}{1 + e^{X_i w_i}} \right) \right]$$

2

2 Programming

In this section of the assignment, you will be setting up a basic Python scripts to perform linear regression and classification, and run some experiments to understand concepts in gradient based optimization. Unpack the zip file containing the Python code. Each file has TODO comments indicating where you will need to fill in the code. In this assignment, we have given you a lot of template code, but the amount of templating we provide will decrease as the term goes on. This code can serve as examples of how to do things like plotting and basic optimization for future homework.

Do not use AI tools such as ChatGPT or Co-Pilot to complete this assignment. The code you write must be your own. We will check for plagiarism against these AI-generated solutions.

2.1 Linear Regression and Classification

In this part of the assignment you will implement a linear regression model with and without a basis function. It will be optimized by least squares (inverse matrix solution), gradient descent, and stochastic gradient descent. The basis function you will consider is a Fourier Basis, e.g., there will be features for

$$\phi(x) = \begin{bmatrix} \sin(C_{1,1}x_1 + C_{1,2}x_2 + \cdots + C_{1,k}x_k) \\ \sin(C_{2,1}x_1 + C_{2,2}x_2 + \cdots + C_{2,k}x_k) \\ \vdots \\ \cos(C_{1,1}x_1 + C_{1,2}x_2 + \cdots + C_{1,k}x_k) \\ \cos(C_{2,1}x_1 + C_{2,2}x_2 + \cdots + C_{2,k}x_k) \\ \vdots \end{bmatrix}$$

where $x \in \mathbb{R}^k$, $C \in \mathbb{R}^{o \times k}$ is a matrix containing $\pi \times$ order for each input feature. C considers all possible combinations of orders for the specified order of the fourier basis. For example, if the order is 2 and the their are 2 features, the C matrix is:

$$C = \pi \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 2 & 0 \\ 2 & 1 \\ 2 & 2 \end{bmatrix}.$$

The C has $o = (\text{order} + 1)^k$ rows. The total number of features for the fourier basis is $2o$.

First, complete the TODOs in the `LinearModel.py` file, e.g., computing the mean square error, gradient of the mean squared error, LinearModel class methods, and Fourier basis class construction and methods. See the code for details on implementing each method.

Next complete the TODOs in the `optimization.py` file. This file contains code for performing gradient descent, stochastic gradient descent, and least squares regression. Some of the code has been filled out for you, but make sure to read through and understand each function.

Then complete the TODOs in `hw_linearfa.py`. This file contains code to answer the questions below and help in solving the regression and classification problems. It also contains code to generate data samples from some nonlinear function. By generating our own data we can control the amount of data, noise, and fully analyze how good the function is being estimated. The main functions to run are `q1()`, `q2()`, `q3()`, and `q4()`. They correspond to the questions below.

1. **Approximation a function** (10 points): Use Linear function approximation with and without a Fourier Basis to approximate the function define in `regression_function`. Compute the least squares solutions using no basis function and the Fourier basis. Estimate the optimal parameters of the Fourier basis approximation using gradient descent. You will need specify the hyperparameters for this problem. For the Fourier Basis specify the order of the basis. For gradient descent specify: the step size, maximum number of iterations, and the tolerance for the weight change for terminating gradient descent. Try and set these so as to optimize the fit but not overfit the data. You will have to make a determination of what is an over/underfit.

   Running `q1()` will produce two figures in which are saved in the files `all_lines.pdf` and `rbf_learning_curve.pdf`. Upload those plots here and report your chosen hyperparameters.
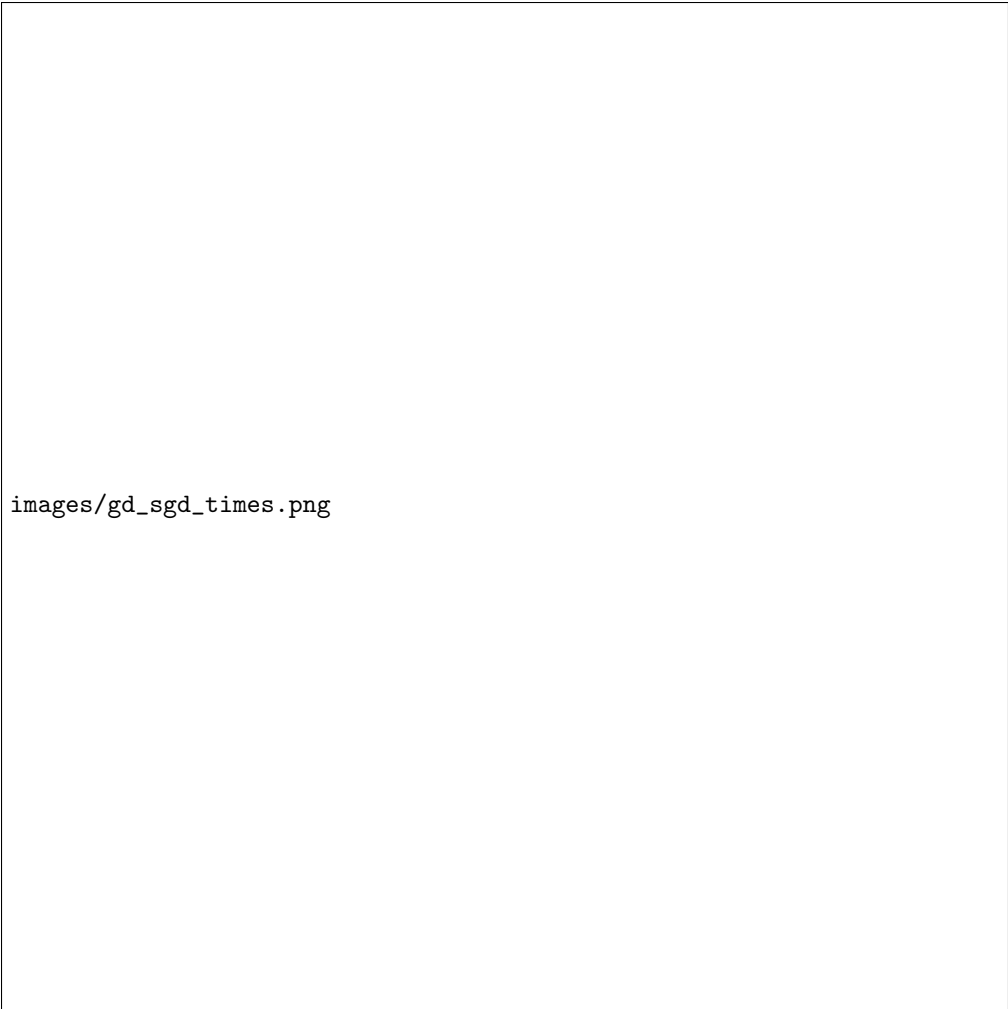   Answer:

images/all_lines.pdf	images/rbf_learning_curve.png

Hyperparameter	Value
order	?
η	?
max iterations	?
weight tolerance	$10^?$

2. **Stochastic Gradient Descent Weight Distribution** (10 points): After completing the missing classification portion of the assignment, complete `q1()`. The goal with this question is to understand how SGD gets close to the optimal weights and the role step size plays in controlling how close the sequence of weights $\{w^k\}$ get to optimal. Recall that we said SGD with a constant step size will lead to weights that "bounce" around the optimal weights but never converge to the optimal weights. For this question you need to specify values of the step size that lets you see that smaller step sizes will lead to a close approximation of the optimal weights and that large step sizes bounce around the optimal weights in a large ball. Find three to five step sizes (in decreasing order) that reveal this trend. Include the Figure saved in `sgd_path.png` below.

images/sgd_path.png

images/gd_sgd_times.png

3. **Speed of Optimization** (10 points): The goal of this question is to understand the computational trade-offs of using least-squares, gradient descent, and stochastic gradient descent. Run `q2()` to produce plots showing how quickly each method finds a solution for various choices of number of data points, number of centers. Based on these plots make an argument about when each method is likely to find a good solution the quickest. You may also argue that one method will not be best choice. Note you should get the run times for each method to at most a few minutes. You do not want to wait hours just to see which is the best.

- Gradient descent is the fastest when:
  Answer:

- stochastic gradient descent is the fastest when:
  Answer:
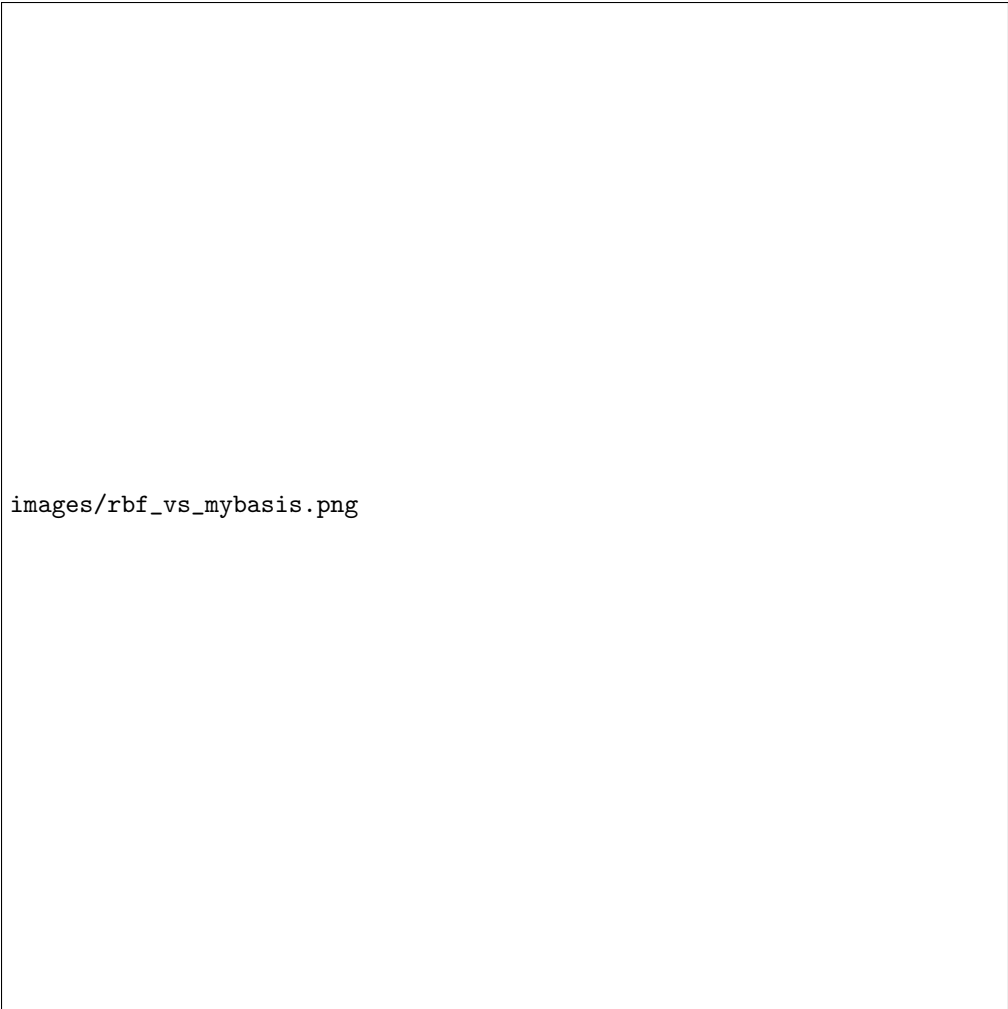
- Least Squares is the best when:
  Answer:

4. **Choosing the right basis function** (15 points): Using Fourier features are flexible, but as you should have seen in the previous questions, if you do not select the right order or the frequencies do not line up in the right spots, the approximation is not capturing underlying function. Due to this mismatch of approximation and true function, it is easy to overfit to the data. So for this question you will create a new basis function that will be able to represent underlying function exactly.

For this question the data is being generated from the function

$$f_*(x) = \sin(2\pi x) + \frac{1}{2}x^2 - x - 1$$

with noise added to the output of function. Design a basis function that will allow for perfect linear approximation of the function. Then you will optimize the fits for this new basis function and an Fourier basis function for differing amounts of data. You will then compare both the MSE on the sampled data as well as the error in approximating $f_*(x)$, e.g., $\frac{1}{1001} \sum_{i=0}^{1,000} |f_*(-2 + i\Delta) - f(-2 + i\Delta, w)|$, where $\Delta = \frac{4}{1000}$. The goal here is to understand how the right features will make minimizing the loss function easier, require less data, and generalize better to points not in the data set.

To complete this question, complete the `MyBasis` object so that it creates a feature vector that is perfect for approximating f_*. Report that feature vector below. Finish the other TODOs in `q3()`. Running `q3()` will produce a plot `rbf_vs_mybasis.png`. Upload that plot below. Answer the following questions:

images/rbf_vs_mybasis.png

- What is your basis function?

$$\phi(x) = \left[\right]^\top$$

- What is the smallest number of data points required to have a good fit with your basis function?

  Answer:

- How many data points are needed to have a good fit for an Fourier basis with order 10?

  Answer:

- How would the fit change if the order is 5 or 20? You can change the necessary hyperparameters and rerun the code to answer this question.

  Answer:

- How does the generalization of `MyBasis` compare to the Fourier basis? If it is better why would it be better? Think about how many data points would be needed to identify near optimal parameters for each basis function.

  Answer:

2.2 Overfitting and Cross Validation

In this section of the homework, you will implement gradient descent with early stopping and cross validation. You will need to finish the TODOs in the `hw_crossvalidation.py` file. The goal of this section of the assignment are: 1) understand over-training and implement early stopping, and 2) to understand model selection bias and implement cross-validation.

1. **Early Stopping** (10 points): In `q1()` you will be training a classifier using an Fourier basis function using gradient descent with early stopping. The features for each class come from a d-dimensional vector generated from a normal distribution with features limited to being no more than two standard deviations away from the mean. Each class has a different mean and standard deviation. A 2D visualization of the data can be generated by running `plot_data()` and looking at `classification_data_2d.png`. For `q1()` the data will be a 4 dimensional vector.

   You need to optimize the hyperparameters to try and create the best fit to the test data set. See TODOs in the code for hyperparameters to tune. Repeat this process for $n = 50$, $n = 100$, $n = 400$. Report the hyperparameters and figure (`early_stopping_n.png`) for each setting of n. Then answer the questions below.

Hyperparameter	n=50	n=100	n=400
parameter name	value	value	value
parameter name	value	value	value
parameter name	value	value	value

images/early_stopping_50.png	images/early_stopping_100.png	images/early_stopping_400.png

(a) Does early stopping always prevent overfitting during gradient descent optimization? Explain.

   Answer:

(b) How trends did you notice that seemed to help or hurt overfitting with regards to selecting hyperparameters?

   Answer:

2. **Cross Validation** (25 Points): For this question you will be implementing cross validation to find the best hyperparameters and compare the ones found in the previous question. Complete the TODOs in `find_best_model`, `sample_hyperparameters`, `sample_k_folds`, `xval_model_selection`, q3(), and q3(). Also specify the hyperparameters from q1() into a dictionary and save it in the variable `HYPERS_FROM_Q1`.

The purpose of this question is to understand cross validation, how tuning hyperparameters on the test set leads to subpar model creation. For the hyperparameter optimization you will be implementing a simple random hyperparameter search. It works by randomly sampling N hyperparameters choices, training the model, and evaluating how well that model does on the validation set. The best hyperparameters are the ones that do the best on average across the different cross validation data sets. Running q2() will perform cross validation and compare the model found with it to the model you found in q1() using the same data. Running q3() will compare cross validation against the hyperparameters you found in q2() for 50 different attempts to generate data. This way we can see how well cross validation works in general compared to trying to find the best hyperparameters for a single data set. Note this can take a long time if your computer is slow.

To illustrate the performance comparison, we will leverage a quantile plot of the performance difference of $l_{D_{test}}(\phi_{q2}, w_{q2}) - l_{D_{test}}(\phi_{xval}, w_{xval})$, where (ϕ_{q2}, w_{q2}) and (ϕ_{xval}, w_{xval}) are the basis functions and weights found using hyperparameters from q1() and cross validation, respectively. The quantile function specifies a value such that a given proportion of data is less than or equal to that value, i.e., $Q_X(p) \doteq \min\{x : \Pr(X \leq x) \geq p\}$ for a random variable X and $p \in (0, 1)$. For example, $Q_X(0.5)$ is the median value of X (50% of the time X is below it and 50% of the time X is above it). For cross validation to be an effective technique, we would expect the difference of performance to the hyperparameters found in q1() to be mostly greater than zero. That is the plot in the file `loss_distributions.png` should quickly raise above 0.

Run q2() and q3(), upload the quantile plot, and answer the questions.

(a) For q2() did cross validation perform better or worse than your selection of hyperparameters. Why might this be? What would impact if cross validation would find better hyperparameters or not?

Answer:

(b) For `q3()` did cross validation perform better or worse on average than your selection of hyperparameters? If it did better how might it be made to preform worse? Try it and report what you did and the quantile plots for both settings. Similarly, if cross validation performed worse, how can you make it perform better? Report the same.

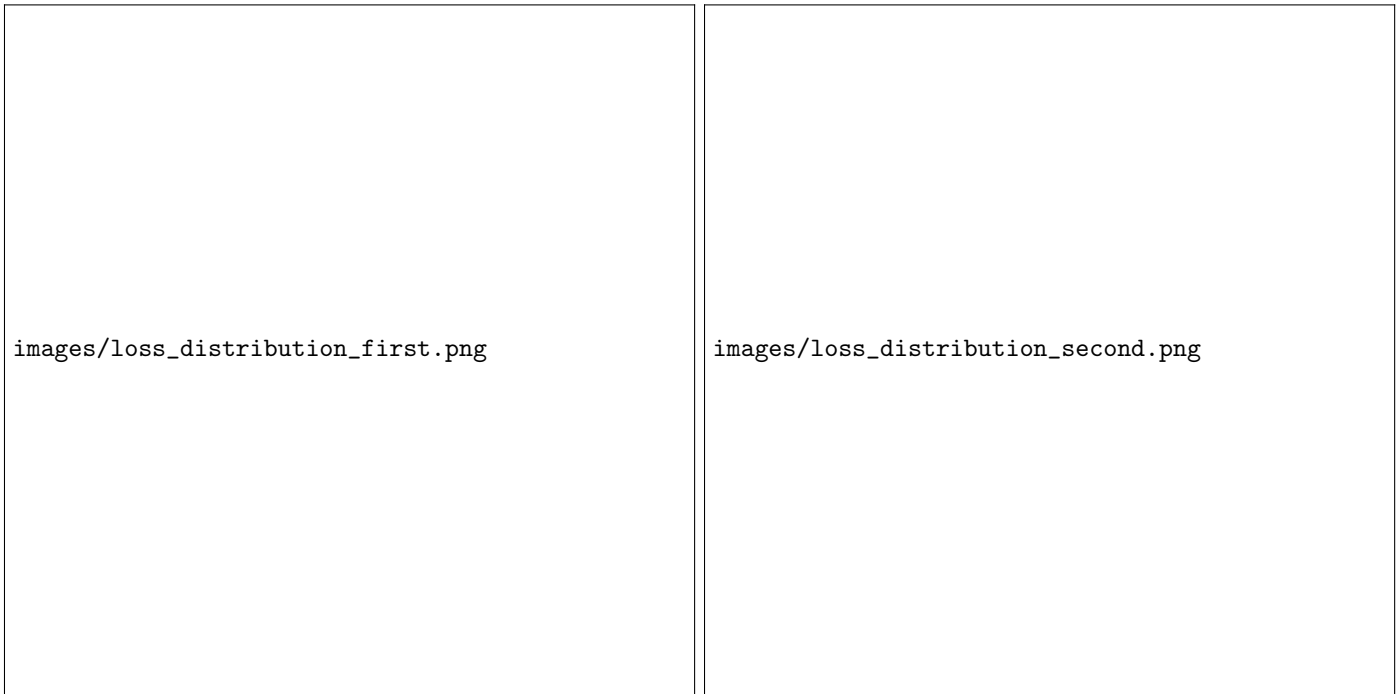|	
images/loss_distribution_first.png	images/loss_distribution_second.png

Figure 1: Specify which plots in the left one and which is the right one. You will need to change the file names to have each plot.

Answer: