6.867: Homework 1

Essentially all problems in machine learning involve some form of optimization, whether to fit a model to data (estimation) or to select a prediction based on a model (decision). In a few cases we can find the optimum analytically; in many others we do it numerically. In this assignment we will begin to explore numerical methods for some of the basic estimation problems that we have been learning about.

You will find a zip file with some useful code and data in the Resources section of the Piazza course page. You can do these assignments in any computational system that you are used to. We recommend Matlab or the Pylab/Numpy/Scipy/Matplotlib cluster of packages and we'll try to provide help for those two systems. If you use anything else, you're on your own...

You will be turning in a single, readable "paper" (a single PDF file) with your solutions. We will be emulating the process of submitting papers for publication to a conference. We will be using an actual conference review system (Easy Chair) to have these papers peer reviewed (by the other students). This means that your answers have to be readable and understandable to your peers and, where possible, interesting. Note that when explanations are called for, you will need to convince the reviewers that you understand what you're talking about. The course staff will serve as the Program Committee for the conference and make all final decisions. The details of this process will be posted on Piazza.

Grading process and due dates

- We *strongly* encourage you to do this assignment in groups of two students, though you may do it individually if you wish. Post on Piazza or email a TA or instructor if you'd like help finding a partner.
- You submit your paper via the Easy Chair site by 11:59 PM on Thursday, October 1.
- Each student who was an author or co-author on a submission will be assigned 3 papers to review.
- Reviews must be entered on the Easy Chair site by 11:59PM on Tuesday, October 6.
- All reviews will be made visible shortly thereafter, and students will have a 2–3 day period in which to enter rebuttals of their reviews into EasyChair if they wish.

Grading rubric

Your paper must be anonymous (no identifying information should appear in the PDF file). If it is not, it will automatically receive a 20% deduction, and will be graded by a grumpy staff member.

The paper must be no more than 6 pages long in a font no smaller than 10 point. It should include whatever tables, graphs, plots, etc., are necessary to demonstrate your work and conclusions. *It should not include code.*

Each of the four parts of the assigment will be graded on a scale from 0 to 5 (where 0 is failing and 5 is an A) on two aspects:

- **Content:** Did the solution answer the questions posed? Were the answers correct? Were the experiments well-designed or examples well chosen?
- **Clarity:** Were the results written up clearly? Were the plots labeled appropriately and described well? Did the plots support the points in the paper? Did the discussion in the paper illuminate the plots?

As a reviewer, you will be asked to provide a score for each section, and at at least two paragraphs of feedback, per review, explaining things that were done well and things that could have been improved upon.

Your overall score for this assignment will be:

- **80%**: The average of all 8 scores on your assignment given by all three reviewers.
- **20%**: A score for the quality of your reviews. This will be full credit, by default. But we will skim reviews and examine some carefully and may reduce this grade for review commentary that is sloppy or wrong.

The course staff will spot-check submissions and reviews, paying careful attention to cases where there were rebuttals. The staff will act as the program committee and determine a final score. Our overall goals in this process are:

- To motivate you to work seriously on the problems and learn something about the machine learning material in the process
- To engage you in thinking critically and learning from other students' solutions to the problems

We will arrange to give full credit to anyone who submits a serious and careful solution to the problems and who gives evidence of having read carefully the solutions they were assigned and who writes thoughtful reviews of them.

The following questions are the points that your paper should cover in order to receive full credit. Your presentation should roughly follow the order of these questions so that your reviewers can see what you're doing.

1 Implement Gradient Descent

The simplest and most commonly referred-to method of optimization is gradient descent (see Bishop section 5.2.4, Murphy section 8.3.2, and Wikipedia). It's worth understanding its behavior on actual problems. There are many much more sophisticated optimization methods available in Matlab and Scipy; you should also benchmark one of those.

1. Implement a basic gradient descent procedure to minimize scalar functions of a vector argument. Write it generically, so that you can easily specify the objective function and the function to compute the gradient. You should be able to specify the initial guess, the step size and the convergence criterion (a threshold such that the algorithm terminates when the difference in objective function value on two successive steps is below this threshold).

2. Test your gradient descent procedure on some functions whose optimal value you know, e.g. a quadratic bowl or (the negative of) a Gaussian probability density function. Make sure that you try functions of more than one variable and that you try at least one convex function and one very non-convex function with multiple minima. Discuss the effect of the choice of starting guess, the step size, and the convergence criterion on the resulting solution.

3. Write code to approximate the gradient of a function numerically at a given point using central differences (see the "Finite difference" article in Wikipedia). Verify its behavior on the functions you used in the question above by comparing the analytic and numerical gradients at various points.

4. Compare the behavior of your gradient descent procedure with one of the more sophisticated optimizers available in Matlab (e.g. `fminunc`) or `scipy.optimize` (e.g. `fmin_bfgs`). A good metric for comparison is the number of function evaluations required to reach convergence. You should make sure your code is able to keep track of function calls and call the Matlab or Scipy optimizers so that they print this.

2 Linear Basis Function Regression

Let's consider the linear combination of basis function class of regression models. We know how to get analytic solutions for the maximum likelihood weight vector (see Bishop equation 3.15, Murphy section 7.2 and equation 7.16). We'll use this problem to "benchmark" the gradient descent solution.

We have provided you with a text file (`curvefitting.txt`) that has the 10 data points that were used to generate the plots in Bishop Figure 1.4 (reproduced below). We have also given you code to read this data and some code illustrating how to generate plots, both in Python and Matlab.

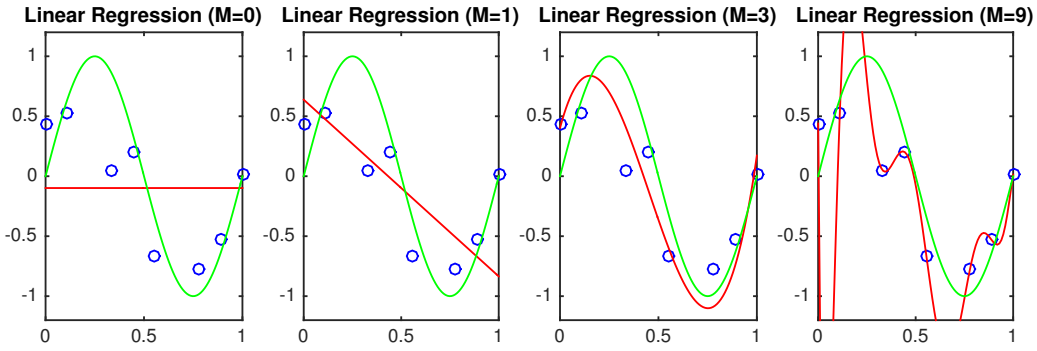

Figure 1: Plots of polynomials having various orders M, shown as red curves, fitted to the data set shown with blue circles. The green curve shows the function $\sin(2\pi x)$ used to generate the data.

1. Write a procedure for computing the maximum likelihood weight vector given (a) an array of 1-dimensional data points X, (b) a vector of Y values, and (c) the value of M, the maximum order of a simple polynomial basis $\phi_1(x) = x, \ldots, \phi_M(x) = x^M$. Test your solution by replicating the plots in Bishop Figure 1.4 and the weight values in Table 1.1. You should be able to get very close agreement.

2. Now, write functions to compute the sum of squares error (SSE) (and its derivative) for a data set and a hypothesis, specified by the list of M polynomial basis functions and a weight vector. Verify your gradient using the numerical derivative code.

3. Use gradient descent on the SSE function to replicate the graphs in Bishop. Describe your experience with initial guesses, step sizes and convergence thresholds. Compare to using one of the more sophisticated optimizers. Explain your results in terms of the properties of the function being optimized and the properties of the algorithms.

4. What would you expect to observe if the basis functions were sine functions rather than polynomials, such that $\phi_1(x) = \sin(2\pi x), \ldots, \phi_M(x) = \sin(2\pi M x)$? (You are encouraged to try using trigonometric basis functions and observing the result, but a brief explanation is sufficient). What are potential disadvantages of this choice if you did not know your data was generated from $\sin(2\pi x)$?

3 Ridge Regression

1. Implement ridge regression; the closed form solution for the optimal solution of ridge regression is easy to calculate so do not use gradient descent for this exercise. Experiment with

different values of λ on the simple data from Bishop's Figure 1.4, for various values of M. Describe your observations.

2. We have given you three additional data sets: one training data set, one test and one validation set. In general, we want to use the training data to optimize parameters and then use the performance of those parameters on the validation data to choose among models (e.g. values of M and values of λ), this is called "model selection". Show some values of M and λ, and show the effect on the fit in the test and validation data. Which values work best? Explain.

3. The BlogFeedback dataset was compiled last year by Krisztian Buza at the Budapest University of Technology and Economics. The original paper can be found at: `http://link.springer.com/chapter/10.1007/978-3-319-01595-8_16`. An accurate description of the dataset is here: `https://archive.ics.uci.edu/ml/datasets/BlogFeedback`. In this dataset a blog post is described by 280 numerical values like the day of the week it was posted on, the number of parent pages and the number of comments it received in the first 24 hours. The target value is the number of comments it will receive in the next 24 hours. This is a real world dataset; the output of your prediction function, for example, could be used as input to an ad pricing algorithm (e.g. charge more to post ads on popular posts). We provide you again with a training set, a validation set, and a test set.

   Use the closed form solution of Ridge Regression for a simple linear model (i.e. $\hat{y} = w^T x$) on the BlogFeedback Dataset and perform model selection. What do you observe? How did you come up with the values of λ you tested?

   This is a *real* dataset, which means that, though it has already gone through some preprocessing, each of the features may have rather significant variations in terms of amplitude. Take a look at the Wikipedia article on "Feature Scaling" to see if you can perform better on the test data after trying some of the different strategies. While not required here, this step is common in practice.

4 Generalizations

So far, we have been working with problems where the analytic solution is very easy and thus gradient descent might be used only for computational reasons, e.g. big data sets. Let's now consider some problems where analytic solutions don't exist. This question is divided into two parts. **You can choose to work on either one of them**.

Table 1: Variations of Least Squares algorithm

	Data	Regularization
OLS	L2	L2
LAD	L1	L2
LASSO	L2	L1

4.1 Outlier and LAD

Only use the `regress` data for this exercise.

1. We have seen that OLS regression solves the "least squares (LS)" fitting problem; it minimizes the sum of the squares of the prediction errors (the L_2 norm). Alternatively, we could minimize the sum of the absolute values of the errors (the L_1 norm). This is called "least absolute deviations (LAD)". Just as using least squares is equivalent to assuming that the outputs ($\Pr(Y \mid X)$) have a Gaussian distribution, LAD is equivalent to assuming that the outputs have a Laplace distribution. Use gradient descent to find the weights that minimize LAD, using the quadratic regularization term as in ridge regression. Repeat your experiments from the ridge regression question using this criterion; illustrate and summarize your salient results.

2. Based on your observation, explain when would you want to use LAD and when would you want to use ridge regression.

4.2 Sparsity and LASSO

For this section, use the dataset provided in `regress-highdim.mat`. In Matlab, the data can be loaded directly with the `load` function. In Python, the function `scipy.io.loadmat` can be used to load it. The dataset contains the following variables:

- `X_train`: a 12×5 matrix, 5 training points.
- `Y_train`: a 1×5 vector, the labels corresponding to the training points.
- `X_test`: a 12×500 matrix, 500 testing points.
- `Y_test`: a 1×500 vector, the labels corresponding to the test points.
- `W_true`: a 1×12 vector, the ground-truth weights.

The dataset is generated according to $y = f(x) + \varepsilon$, where $x, y \in \mathbb{R}$, and ε us a zero-mean Gaussian noise. The function $f(\cdot)$ is given by $f(x) = w^T \phi(x)$, for the weight vector $w \in \mathbb{R}^{12}$ and the features are given by

$$\phi(x) = (x, \sin(0.4\pi x \times 1), \ldots, \sin(0.4\pi x \times 11))$$

The goal is to estimate the weights w from the training dataset. If you look at `W_true`, you will see that only two entries of w are non-zero. So this is essentially a regression problem in 2D, instead of 12D. But without knowing `W_true`, we do not know which components are non-zero. This problem provides a guide for estimating sparse weights with the LASSO estimator.

1. Assume `W_true` is unknown, estimate it using ridge regression. Compute the estimation with gradient descent instead of using the closed-form solution. Compute the mean square error (MSE) of your estimation on the test data. What happens if you disable the L_2 regularizer? Compute the MSE.

2. In ridge regression we used a quadratic regularizer (the sum of squares of the weights (L_2 norm)). Bishop's equation (3.29) (reproduced below with minor notation changes for consistency with the lectures) shows a generalization to other exponents. In particular, using $q = 1$ (the sum of the absolute values of the weights (L_1 norm)) is called the "LASSO". Specifically, we minimize the following objective function for LASSO:

$$\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - w^T\phi(x^{(i)})\right)^2 + \lambda\sum_{j=1}^{M}|w_j| \tag{1}$$

   Use gradient descent to compute the estimated weights of LASSO on the provided training dataset. Compute the MSE of the LASSO estimation.

3. Plot the `W_true`, the estimated w by the ridge regression (L_2 regularizer) and the LASSO (L_1 regularizer). Do you get sparse estimation from LASSO? Can you explain or develop intuition for some simple cases why LASSO provides sparse estimation while ridge regression does not?

Figure 2 and Figure 3 provides some reference illustrations of this dataset, but you are free to choose your own way of illustration in the report. The results will be different depending on different optimization algorithm, regularization coefficients, random initialization, etc. If you want to plot the estimated function as in Figure 2, you can use the following Python code snippet:

```
import numpy as np
import matplotlib.pyplot as plt

X0 = np.arange(-1, 1, 0.01)
X0 = X0.reshape(1, len(X0))

Ks = np.arange(1,12).reshape(12-1,1)
X  = np.sin(2*np.pi*X0 * Ks / 5)
X  = np.vstack((X0, X))
```

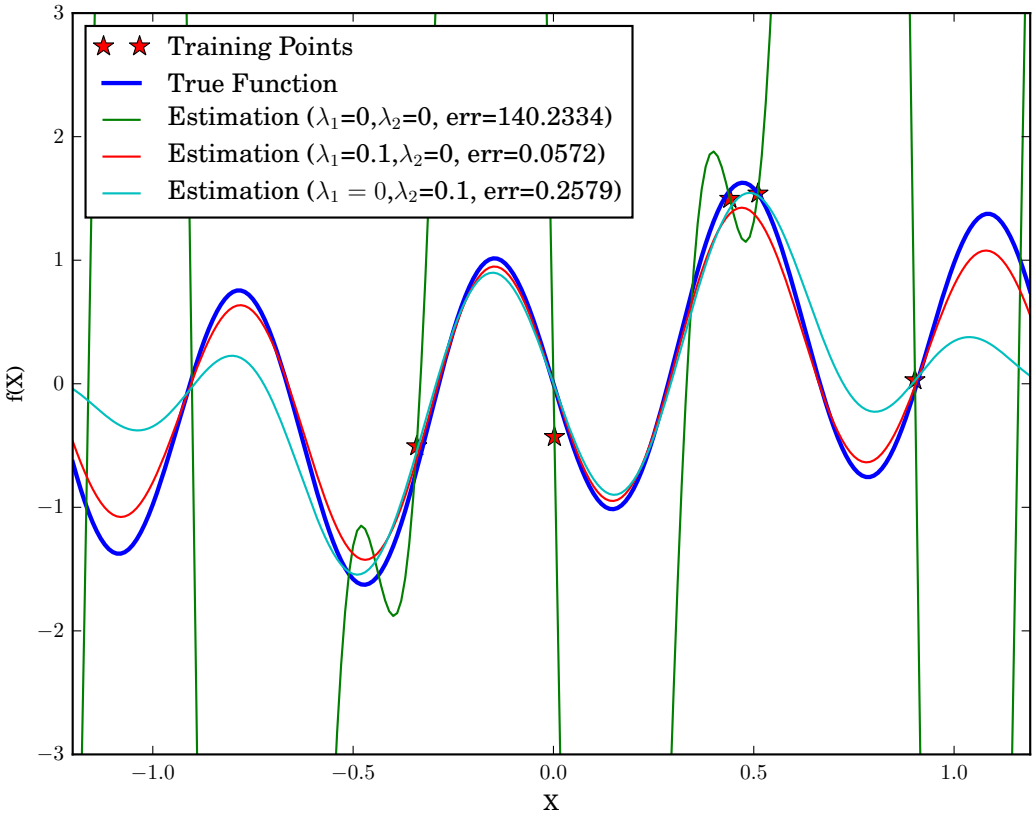

Figure 2: Plots of estimated function with different regularizers.

```
Y  = np.dot(X.T, W)  % W is your estimation

plt.plot(X0, Y, label="my estimation")
```

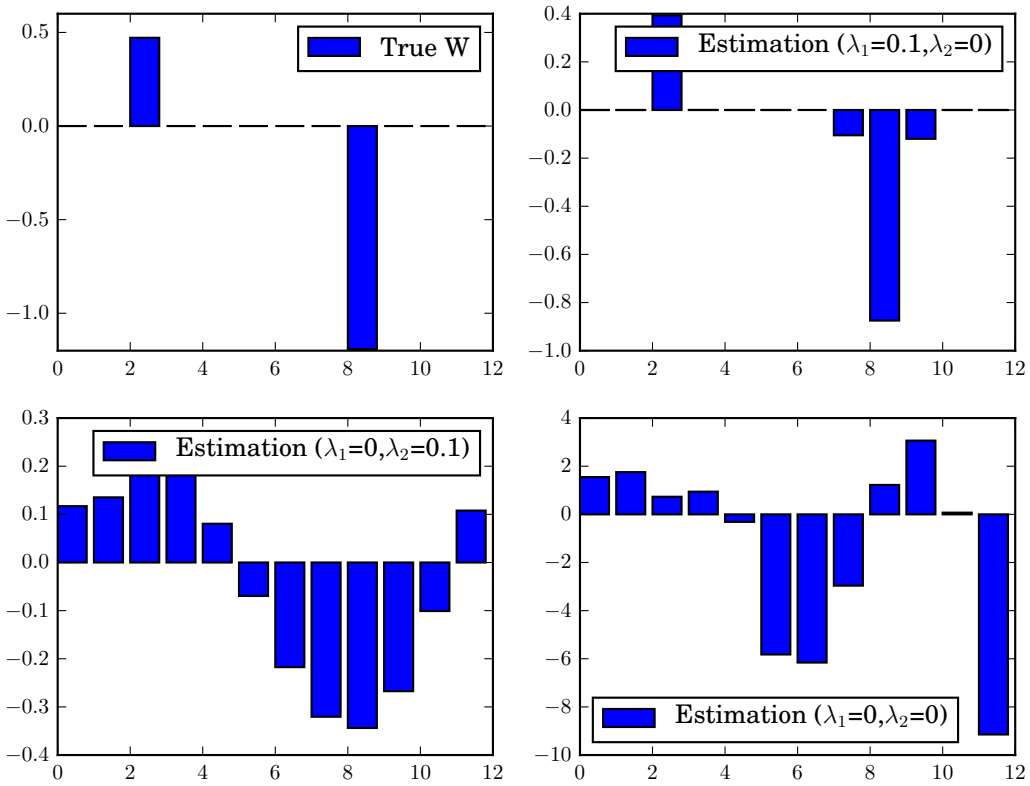

Figure 3: Estimated weights with different regularizers.