CPSC 540 Assignment 1 (due January 19)

Linear Models

1. You can work in groups on 1-3 people on the assignments.
2. Place your names and student numbers on the first page.
3. Submit all answers as a single PDF file using the handin program (details on Piazza).
4. The PDF should answer the questions *in order*, and include relevant code in the appropriate place.
5. You will lose marks if answers are unclear or if the TA can't easily find the location of the answers.
6. Corrections/clarifications after the first version of this document is posted will be marked in red.
7. Acknowledge sources of help from outside of class/textbook (classmates, online material, papers, etc.).
8. All numbered questions are equally weighted.

Course Prerequisite Form

If you are a graduate student in CPSC or EECE, then you did not need to submit the prerequisite form to stay enrolled in the course. Please remember to hand this in during class on Thursday January 14 (or any class before this one).

1 Vectors, Matrices, and Quadratic Functions

In this course we will heavily use vectors and matrices, as well as linear and quadratic functions. To refresh the memories of people who haven't used these in a long time, the first part of this question makes you review basic operations on vectors and matrices. If you are rusty on basic vector and matrix operations, see the notes on linear algebra on the course webpage. The second part of the question gives you practice taking the first and second derivatives of linear and quadratic functions, and the third part gives you practice finding the minimizer of (strictly-convex) quadratic functions.

1.1 Basic Operations

Using the definitions below,

$$\alpha = 10, \quad x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix},$$

evaluate the following expressions (show your work):

1. $\alpha(x+y)$ (vector addition and scalar multiplication)
2. $x^T y + \|x\|^2$ (inner product and norm)
3. Ax (matrix-vector multiplication)
4. $A^T A$ (matrix-matrix multiplication)
5. $\text{Tr}(A^T A)$ (matrix trace)

If $\{x, y, z\}$ are real-valued column-vectors of length d and $\{A, B, C\}$ are real-valued d by d matrices, state whether each of the below statements is true or false in general

6. $x^T(y+z) = z^T x + y^T x$
7. $x^T x = x x^T$
8. $x^T A y = y^T A^T x$
9. $x^T(y^T z) = (x^T y)^T z$
10. $AB = BA$
11. $A(B^T C) = (AB^T)C$
12. $A^T(B+C) = A^T B + A^T C$
13. $(A+B)^T = A^T + B^T$
14. $(AB)^T = A^T B^T$

1.2 Gradients and Hessians of Linear and Quadratic Functions

For this question we'll use the convention that all values are real and:

1. α is a scalar.
2. a and b are length-d column-vectors.
3. Element i of b is denoted by b_i.
4. A and B are d by d matrices.
5. Row i row of A is denoted by a_i^T.
6. W is a *symmetric* d by d matrix.

Express the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ of the following linear/quadratic functions in matrix notation, simplifying as much as possible.

1. $f(x) = a^T x + \alpha$ (linear)
2. $f(x) = x^T a + a^T A x + x^T A^T b$ (more linear forms)
3. $f(x) = x^T x + x^T W x + x^T A B x$ (quadratic forms)
4. $f(x) = \frac{1}{2}(Ax - b)^T W (Ax - b)$ (weighted least squares)
5. $f(x) = \frac{\lambda}{2}\|x\|^2 + \frac{1}{2}\sum_{i=1}^n (a_i^T x - b_i)^2$ (L2-regularized least squares)

Hint: You can use the results we showed on the first day of class to quickly compute these quantities for each term (for the fifth example, you'll first need to convert to matrix and vector notation). You can use 0 to represent the zero vector or a matrix of zeroes, and I to denote the identity matrix. As a sanity check, make sure that your results have the right dimensions.

1.3 Minima of Strictly-Convex Quadratic Functions

Using the notation from the previous question, compute the minimizer of the below quadratic functions, simplifying as much as possible. For this question, you can assume that these functions are strictly-convex. This implies that the Hessian is invertible and that any x with $\nabla f(x) = 0$ is the unique minimizer.

1. $f(x) = \frac{1}{2}x^T Ax + b^T x + \alpha$ (general quadratic form)
2. $f(x) = \frac{1}{2}(Ax - b)^T W(Ax - b)$ (weighted least squares)
3. $f(x) = \frac{1}{2}\sum_{j=1}^{d}[\lambda_j x_j^2] + \frac{1}{2}\sum_{i=1}^{n}(a_i^T x - b_i)^2$ (least squares with weighted L2-regularization)

Hint: You can solve problems 2 and 3 by converting to the general form in part 1. For part 3, you can use Λ to denote a diagonal matrix with the λ_j along the diagonal. For part 1, the assumptions imply that A is symmetric and invertibe.

2 Linear Regression and Nonlinear Bases

In class we discuss fitting a linear regression model by minimizing the squared error. This classic model is the simplest version of many of the more complicated models we will discuss in the course. However, it typically performs very poorly in practice. One of the reasons it performs poorly is that it assumes that the target y_i is a linear function of the features x_i with an intercept of zero. This drawback can be addressed by adding a bias variable and using nonlinear bases (although nonlinear bases may increase to over-fitting).

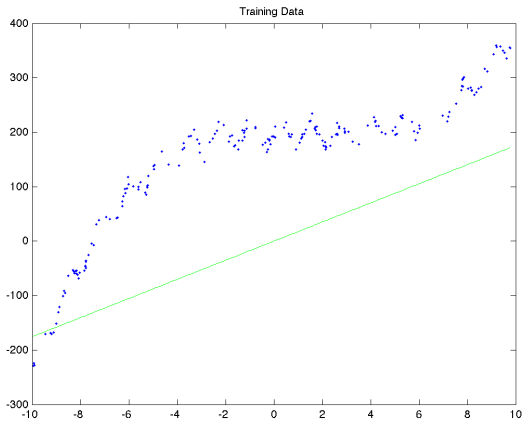
In this question, you will start with a data set where least squares performs poorly. You will then explore how adding a bias variable and using nonlinear (polynomial) bases can drastically improve the performance. You will also explore how the complexity of a basis affects both the training error and the test error. In the final part of the question, it will be up to you to design a basis with better performance than polynomial bases. If you are not familiar with Matlab, to get you started please see the notes on Matlab commands on the course webpage.

2.1 Adding a Bias Variable

Download *a1.zip* from the course webpage, and start Matlab in a directory containing the extracted files. If you run the script *example_basis*, it will:

1. Load a one-dimensional regression dataset.
2. Fit a least-squares linear regression model.
3. Report the training error.
4. Report the test error (on a dataset not used for training).
5. Draw a figure showing the training data and what the linear model looks like.

Unfortunately, this is an awful model of the data. The average squared training error on the data set is over 28000 (as is the test error), and the figure produced by the demo confirms that the predictions are usually nowhere near the training data:



The y-intercept of this data is clearly not zero (it looks like it's closer to 200), so we should expect to improve performance by adding a *bias* variable, so that our model is

$$y_i = w^T x_i + \beta.$$

instead of

$$y_i = w^T x_i.$$

Write a new function, *leastSquaresBias*, that has the same input/model/predict format as the *leastSquares* function, but that includes a *bias* variable w_0. Hand in your new function, the updated plot, and the updated training/test error.

Hint: recall that adding a bias β is equivalent to adding a column of ones to the matrix X. Don't forget that you need to do the same transformation in the *predict* function.

2.2 Polynomial Basis

Adding a bias variable improves the prediction substantially, but the model is still problematic because the target seems to be is a *non-linear* function of the input. Write a new function, *leastSquaresBasis(x,y,deg)*, that takes a data vector x (i.e., assuming we only have one feature) and the polynomial order *deg*. The function should perform a least squares fit based on a matrix $Xpoly$ where each of its rows contains the values $(x_i)^j$ for $j = 0$ up to *deg*. E.g., *leastSquaresBasis(x,y,3)* should form the matrix

$$Xpoly = \begin{bmatrix} 1 & x_1 & (x_1)^2 & (x_1)^3 \\ 1 & x_2 & (x_2)^2 & (x_2)^3 \\ \vdots & & & \\ 1 & x_n & (x_n)^2 & (x_N)^3 \end{bmatrix},$$

and fit a least squares model based on it. Hand in the new function, and report the training and test error for $deg = 0$ through $deg = 10$. Explain the effect of *deg* on the training error and on the test error..

Hints: To keep the code simple and reduce the chance of having errors, you may want to write a new function *polyBasis* that you can use for both the training and testing data.

2.3 Manual Search for Optimal Basis

Polynomials are a flexible class of functions, but there is structure in this data that is not well-modelled by polynomials. Try to find a nonlinear basis that gives the best performance on this dataset in terms of test

error. Report the basis that you use and the training/test score that you achieve.

Hint: the data seems to have periodic behaviour, and it's possible to obtain training and test errors below 60.

3 Non-Parametric Bases and Cross-Validation

Unfortunately, in practice we often don't know what basis to use. However, if we have enough data then we can make up for this by using a basis that is flexible enough to model any reasonable function. These may perform poorly if we don't have much data, but can perform almost as well as the optimal basis as the size of the dataset grows. In this question you will explore using Gaussian radial basis functions (RBFs), which have this property. These RBFs depend on a parameter σ, which (like *deg* in the polynomial basis) can be chosen using a validation set. In this question, you will also see how cross-validation allows you to tune parameters of the model on a larger dataset than a strict training/validation split would allow.

3.1 Proper Training and Validation Sets

If you run the demo *example_rbf*, it will load a dataset and split the training examples into a "train" and a "validation" set. It will then search for the best value of σ for the RBF basis (it also uses regularization since $X_{\text{rbf}}^T X_{\text{rbf}}$ tends to be very close to singular). Once it has the "best" value of σ, it re-trains on the entire dataset and reports the training error on the full training set as well as the error on the test set.

Unfortunately, there is a problem with this function. Because of this problem, the RBF basis doesn't perform much better than a linear model. What is the problem with this training/validation/testing procedure? How can you fix this problem?

Hint: you may want to plot the models that are considered during the search for σ.

3.2 Cross-Validation

Using the standard solution to the above problem, a strange behaviour appears: if you run the script more than once it might choose different values of σ. It rarely performs too badly, but it's clear that the randomization has an effect on the value of σ that we choose. This variability would be reduced if we had a larger "train" and "validation" set, and one way to simulate this is with *cross-validation*. Modify the training/validation procedure to use 10-fold cross-validation to select σ, and hand in your code. What value of σ does this procedure typically select?

3.3 Cost of Non-Parametric Bases

When dealing with larger datasets, an important issue is the dependence of the computational cost on the number of training examples n and the number of features d. What is the cost in big-O notation of training the model on n training examples with d features under (a) the linear basis, and (b) Gaussian RBFs (for a fixed σ)? What is the cost of classifying t new examples under these 3 bases? When are RBFs cheaper to train? When are RBFs cheaper to test

Hint: if you are not familiar with big-O notation, see the slides on big-O notation on the course webpage.

3.4 Non-Parametric Bases with Uneven Data

There are reasons why this dataset is particularly well-suited to Gaussian RBFs:

1. The period of the oscillations stays constant.
2. We have evenly sampled the training data across its domain.

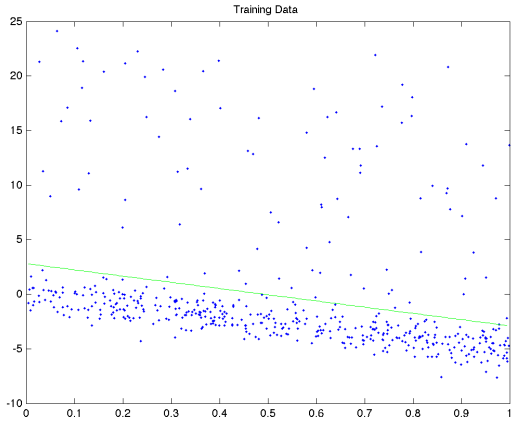
If either of these assumptions are violated, the performance with our Gaussian RBFs might be much worse. Consider a scenario where either 1 or 2 above is violated, and describe a way that you could address this problem.

4 Robust and Brittle Loss Functions

Using the squared error is a convenient choice because we can compute the solution using linear algebra. However, it is rarely the best choice. In this question, you'll analyze a dataset containing a larger number of points that look like outliers. You'll see that this leads to a non-intuitive least squares estimate, and that using a "robust" loss function like the absolute error gives a more intuitive result. Then we'll consider the case where we want to optimize the *worst-case* performance, and how in this case we want to go in the opposite direction and consider a "brittle" loss function that focuses on not performing too badly on the "outliers". This question will also give you some practice with non-smooth optimization, which will arise frequently in this course.

4.1 Least Absolute Error

The script *example_outliers* loads a one-dimensional regression dataset that has a non-trivial number of 'outlier' data points (note that this script calls the *leastSquaresBias* function you need to write for Question 2.1). These points do not fit the general trend of the rest of the data, and pull the least squares model away from the main cluster of points:



One way to improve the performance in this setting is simply to remove or downweight the outliers. However, in high-dimensions it may be difficult to determine whether points are indeed outliers. In such cases, it is preferable to replace the squared error with an error that is more robust to outliers. Write a new function, *robustRegression(X,y)*, that adds a bias variable and fits a linear regression model by minimizing the absolute

error instead of the square error,

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{N} |w^T x_i - y_i|.$$

You should turn this into a *linear program* as shown in class, and you can solve this linear program using Matlab's *linprog* function. Hand in the new function and report the minimum absolute training error that is possible on this dataset.

4.2 Maximum Absolute Error

The previous question assumes that the "outliers" are points that we don't want to model. But what if obtaining good performance on "outliers" is crucial for our application? In this setting we may want to consider a "brittle" regression method that chases outliers in order to improve the worst-case error. For example, consider minimizing the absolute error in the worst-case,

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \max_i \{|w^T x_i - y_i|\}.$$

This objective function is non-smooth because of the absolute value function as well as the max function. Show how to formulate this non-smooth optimization problem as a linear program. Write and hand in a function, *brittleRegression* that fits this model using *linprog* (after adding a bias variable). Report the minimum maximum absolute training error that is possible on this dataset.

5 Regularization and MAP Estimation

Regularization is a core idea in machine learning: regularization can significantly reduce overfitting when we fit very complicated models. In this question, you'll first implement a simple L2-regularized least squares model and see the dramatic performance improvement it can bring on test data.

Regularization is also connected to the idea of using a *prior* on model parameters and performing MAP estimation. This connection is what will allow us to develop more advanced Bayesian methods later in the course that further reduce overfitting. After the first part of this question, you will express MAP estimators as the solutions of regularized optimization problems. If you are not used to manipulating expressions that involve max and argmax, see the max and argmax notes on the course webpage.

5.1 Effect on Test Error of L2-Regularization

If you run the function *example_regularization*, it will:

1. Load a regression dataset containing a training and a validation set.
2. 'Standardize' the columns of X (subtract the mean and divide by the standard deviation).
3. Apply the same transformation to $Xvalidate$.
4. Fit a least squares model.
5. Report the training and validation set errors.
6. Fit a least squares model with a bias term (assuming you've finished Question 2.1).
7. Report the training and validation set errors.

In this dataset, most of the variables are not relevant to the prediction. In particular, after you add a bias variable only the variables j where j is a prime number are relevant. In this demo, the validation error is huge compared to the (tiny) training error. This model is clearly over-fitting.

To reduce overfitting and improve performance on the validation set, write a function leastSquaresReg that adds a bias variable and then fits an L2-regularized least squares model. Hand in your code. With $\lambda = 1$, what is the effect of the regularization on the training error and on the validation error?

5.2 Weighted L2-Regularization and MAP Estimation

In class, we showed that computing a MAP estimate under the assumptions

$$y_i|x_i, w \sim \mathcal{N}(w^T x_i, 1), \quad w_j \sim \mathcal{N}(0, \lambda^{-1}),$$

is equivalent to finding the w that minimizes the L2-regularized squared error,

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2, \tag{1}$$

and the solution to this problem is

$$w = (X^T X + \lambda I)^{-1}(X^T y).$$

If we have prior knowledge about the scale of the variables (for example, some w_j are expected to be small while some are expected to be large), we might consider using a different λ_j for each variable w_j. In other words, we would assume

$$y_i|x_i, w \sim \mathcal{N}(w^T x_i, 1), \quad w_j \sim \mathcal{N}(0, \lambda_j^{-1}).$$

Derive the MAP estimate for w under these assumptions (you can assume that $\lambda_j > 0$ for all j).

Hint: see part 3 of Question 1.3.

5.3 Laplacian Errors

The normal distribution has "light" tails, meaning that the deviations from its mean are expected to be very small. If we want to allow for the possibility that y_i is sometimes very different from $w^T x_i$, we should use a distribution with "heavy" tails. For example, we could assume that y_i follows a Laplace distribution,

$$y_i|x_i, w \sim \mathcal{L}(w^T x_i, 1), \quad w_j \sim \mathcal{N}(0, \lambda^{-1}),$$

where the probability density function of the Laplace distribution $\mathcal{L}$ is given by

$$p(\theta|\mu, \sigma) = \frac{1}{2\sigma}\exp\left(-\frac{|\theta - \mu|}{\sigma}\right).$$

Derive an optimization problem, similar to (1), that we could solve to compute the MAP estimator under these assumptions. Simplify the problem as much as possible. What kind of mathematical programming technique could we use to solve this non-smooth problem?

5.4 Student's t Errors

A distribution that can be even more heavy-tailed than the Laplace distribution is Student's t-distribution, where the density is given by

$$p(\theta|\nu) = \frac{1}{\sqrt{\nu}B\left(\frac{1}{2}, \frac{\nu}{2}\right)}\left(1 + \frac{\theta^2}{\nu}\right)^{-\frac{\nu+1}{2}},$$

where B is the "Beta" function and the parameter ν is called the "degrees of freedom". Again assuming a $\mathcal{N}(0, \lambda^{-1})$ prior on the weights w_j, use Student's t-distribution to derive an optimization problem, similar to (1), that could be even less sensitive to outliers than the model from the previous question. With this loss function, what is the effect of the "degree of freedom" parameter ν on the robustness of this model?.