CS 335 Homework 2, Fall 2014

Dan Sheldon

Revision History

- Friday 9/26: added some extra credit problems
- Tuesday 9/23: due dates revised
- Monday 9/22: clarification in Problem 3 and added extra credit problem
- Sunday 9/21: first posted

Instructions

The exercises are due on Friday 9/26 at 11:55pm.

What to submit:

- Written work
- `exercise_1.m`

The problems are due on Wed 10/1 at 11:55pm.

What to submit:

- `problem_1.m`
- `problem_2.m`
- `problem_3.m`
- Written or typed solutions to non-coding questions

Digital files should be submitted via moodle. Your answers to the other questions can be submitted as a pdf by moodle or as a hard copy in the dropbox outside Clapp 222B.

Exercises (2 points unless otherwise indicated)

The purpose of these exercises is to practice manipulating matrices and vectors in MATLAB and the basics of matrix multiplication, matrix-vector multiplication, and algebraic rules using matrices and vectors. These are foundational skills for developing machine learning algorithms.

1. (10 points) Create a file `exercise_1.m` and write code to do the following.
    - Enter the following matrices and vectors

$$A = \begin{bmatrix} -2 & -3 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

- Compute $C = A^{-1}$
- Check that $AC = I$ and $CA = I$
- Compute $A\mathbf{x}$
- Compute $A^T A$
- Compute $A\mathbf{x} - B\mathbf{x}$
- Compute $\|\mathbf{x}\|$ (use the dot product)
- Compute $\|A\mathbf{x} - B\mathbf{x}\|$
- Print the first column of A (do not use a loop)
- Assign the vector $\mathbf{x}$ to the first column of B (do not use a loop)
- Compute the element-wise product between the first column of A and the second column of A

2. Write the result of the following matrix-matrix multiplication. Your answer should be written in terms of u, v, a, and b.

$$\begin{bmatrix} 1 & -1 \\ 2 & 3 \\ 4 & -1 \end{bmatrix} \cdot \begin{bmatrix} u & a \\ v & b \end{bmatrix}$$

3. Suppose $A \in \mathbb{R}^{2\times 3}, B \in \mathbb{R}^{4\times 2}$. Does the product AB exist? If so, what size is it?

4. Suppose $A \in \mathbb{R}^{3\times 2}, B \in \mathbb{R}^{2\times 4}$. Does the product AB exist? If so, what size is it?

5. Suppose $A \in \mathbb{R}^{3\times 2}, \mathbf{x} \in \mathbb{R}^2$. Is $A\mathbf{x}$ a row vector or a column vector?

6. Suppose $A \in \mathbb{R}^{3\times 2}, \mathbf{y} \in \mathbb{R}^3$. Is $\mathbf{y}^T A$ a row vector or a column vector?

7. (5 points) Suppose $(A\mathbf{u} - \mathbf{v})^T B^T = \mathbf{0}$, where A and B are both invertible $n \times n$ matrices, $\mathbf{u}$ and $\mathbf{v}$ are vectors in $\mathbb{R}^n$, and $\mathbf{0}$ is a vector of all zeros. Use the properties of multiplication, transpose, and inverse (slides 17, 31, and 36 from the linear algebra review) to show that $\mathbf{u} = A^{-1}\mathbf{v}$. Show your work.

Problems (10 points each)

The purpose of these problems is: (1) to practice writing "vectorized" versions of algorithms in MATLAB (normal equations and gradient descent for linear regression), (2) to understand how feature expansion can be used to fit non-linear hypotheses using linear methods, and (3) to understand feature normalization and its impact on numerical optimization for machine learning.

1. (Normal equations, polynomial regression) In this problem you will implement the normal equations for multivariate linear regression in MATLAB and use them to perform polynomial regression. Read and run the code in `problem_1.m`. This generates data from a fourth-degree polynomial and then uses feature expansion to set up the problem of learning the polynomial as multivariate linear regression:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 = \boldsymbol{\theta}^T \mathbf{x}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{bmatrix}$$

When you run the script, you will see a figure showing the training data and predictions from the learned model—but these are just a flat line because the functions `cost_function.m` and `normal_equations.m` are not yet implemented properly. Complete the code in these two functions. When you are done, you should see that the predicted values from the learned model are a good match to the training data.

2. (Gradient descent) In this problem you will write a vectorized version of gradient descent for multivariate linear regression. Create a file `problem_2.m` that is just like `problem_1.m` except it calls the function `gradient_descent()` instead of `normal_equations()` to learn $\boldsymbol{\theta}$. Complete the code in `gradient_descent.m` and call this function from your script so `problem_2.m` works properly.

   Experiment with different step sizes and number of iterations until your implementation of gradient descent is able to find a good hypothesis. Try to match the cost function value you got in Problem 1 to two decimal places. How many iterations does this take?

3. (Feature normalization) You should have observed that it takes many iterations of gradient descent to match the cost function value achieved by the normal equations. In this problem you will implement feature normalization to improve the convergence of gradient descent. Do this in a new file called `problem_3.m`, which will look just like `problem_2.m` except it normalizes the features before calling gradient descent. Remember that the formula for feature normalization is:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

   Here are some guidelines for the implementation:

   - The same transformation should be applied to train and test data.
   - The values μ_j and σ_j are the mean and standard deviation of the jth column (i.e., feature) in the **training data**. (Hint: there are MATLAB functions to compute these.)
   - Do not normalize the column of all ones. (Optional question: why?)
   - Learn the parameter vector $\boldsymbol{\theta}$ by gradient descent using the normalized training data.
   - Use $\boldsymbol{\theta}$ with the normalized test data to make predictions for the test data.

   Now, tune the step size and number of iterations again to make gradient descent converge as quickly as possible. How many iterations does it take to match the cost function from Problem 1 to two decimal places?

4. (Extra credit) Show that the hypothesis found by the normal equations will be the same regardless of whether or not you normalize features.[1] (**Hint**: show that there is a bijection between hypotheses using the original features and hypotheses using the normalized features. Use the fact that the normal equations find a hypothesis of minimum cost.)

Additional Problems (for fun or extra credit)

Note: max 10 points extra credit on the assignment.

1. The data in Problem 1 was generated from a 4th degree polynomial. Try fitting higher degree polynomials to the same data. Observe what happens as you increase the degree of the fitted polynomial.

[1] In this model, normalization helps gradient descent find the best hypothesis more easily, but it doesn't change what the best hypothesis is. In later algorithms we will study, normalization also helps improve the quality of the best hypothesis.

2. There are some cases where X^TX is *not* invertible. In this case, the normal equations still do the right thing if they are modified to use the Moore-Penrose *pseudoinverse*, which is often denoted $(X^TX)^+$. In MATLAB, this looks like:

   ```
   theta = pinv(X'*X)*X'*y;
   ```

   (This also does the right thing if X^TX is invertible, so it is the preferred way of coding the normal equations.) Here are some things for you to explore:

   - Read about the Moore-Penrose pseudoinverse on Wikipedia.
   - Describe two different properties of a training set that would lead to X^TX not being invertible.

3. Find your own data where a straight line does not give a good fit, and fit a polynomial to it.