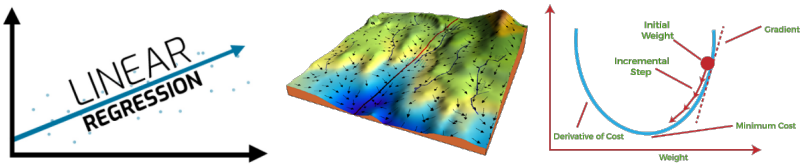
Visual Exploration of Cost Functions and Gradient Descent

DS4400: Machine Learning I

Prof. Rachlin



Introduction

In this assignment, we'll practice working with and numpy arrays and various ways to visualize cost functions. In addition, you will implement and compare several algorithms for finding the linear best. Your implementation should be in python. You may use any libraries you wish for visualizing the cost functions, but your algorithms for gradient descent should be hand-crafted and not simply call scikit-learn or other libraries (unless instructed to do so.) You may submit your solutions as python programs + well-named image files (png, jpg, or pdf) or you may submit a Jupyter notebook. If using Jupyter, please *also* upload the notebook saved as a .pdf.

Instructions

1. Read the advertising data set (3 attributes, one response variable) into table using a pandas data frame.

2. In machine learning, feature scaling refers to putting the feature values into the same range. Feature scaling can be done using *standardization* or *normalization.* Add two columns to your table:

   TV_{std}: $x' = (x - \mu) / \sigma$, where μ is the mean and σ is the standard deviation.

   TV_{norm}: $x' = (x - x_{min}) / (x_{max} - x_{min})$

   Here, the *x* values are taken from the original TV column.

3. Use the seaborn library to produce a joint plot of sales (y axis) vs. TV_{std} advertising (x-axis). Specifying the parameter *kind='reg'* will produce a linear regression line. The shaded area around the line represents a 95% confidence interval.

4. Implement functions for MSE (Mean Squared Error) and MAE (Mean Absolute Error). Both functions should accept two parameters (actual response values and model predictions as either a data series or a vector).

5. Suppose that our predicted response is: $y_{pred} = \beta_0 + \beta_1 x$. Create two surface plots depicting *MSE* as a function of β_0 and β_1. One plot should use x-values taken from TV_{std}, the other from TV_{norm}. Allow both β_0 and β_1 to range from -200 to +200 in steps of 5. Create two more surface plots for MAE. Do these visualizations suggest why scikit learn will standardize the feature values by default?

6. Implement a "random step" search algorithm (as a function) to find the best-fit linear model for X = $TV_{std,}$ y = sales. Here is how the random step algorithm works: Start your search at a random value for β_0 and β_1 in the range [-200…+200]. Measure the current MSE. Now randomly adjust values for both coefficients by some random amount [-1…+1] and recompute the error. If the MSE is reduced, keep the new values for β_0 and β_1. Otherwise revert to the original values. Halt the algorithm when no improvements to MSE can be found after *k = 1000* random updates. Report your final β_0 and β_1. Are your results consistent with your Seaborn joint plot from step 3?

7. Modify your random step algorithm to instead implement gradient descent. Recompute and report your optimal best fit coefficients.

8. Determine a linear best-fit using the scikit-learn linear regression model. The values obtained from your two algorithms should be very close to the scikit-learn result!

9. Plot the progress of your two algorithms on a single *contour plot* of the MSE error function.

Submit

Submit code (python files or Jupyter notebooks) and clearly labeled outputs to gradescope.