II. Linear Regression

In this problem, you will work on linear regression with multiple features using gradient descent. In your starter code, you will find `input1.csv`, containing a series of data points. Each point is a comma-separated ordered triple, representing **age**, **weight**, and **height** (derived from CDC growth charts data).

Data Preparation and Normalization. Once you load your dataset, explore the content to identify each feature. Remember to add the vector 1 (intercept) ahead of your data matrix. You will notice that the **features** are not on the same scale. They represent age (years), and weight (kilograms). What is the mean and standard deviation of each feature? The last column is the **label**, and represents the height (meters). **Scale** each feature by its standard deviation, and set its mean to zero. You do not need to scale the intercept. For each feature *x* (a column in your data matrix), use the following formula:

$$x_{\text{scaled}} = \frac{x - \mu(x)}{stdev(x)}$$

Gradient Descent. Implement gradient descent to find a regression model. Initialize your β's to zero. Recall the empirical risk and gradient descent rule as follows:

$$R(\beta) = \frac{1}{2n}\sum_{i=0}^{n}(f(x_i) - y_i)^2$$

$$\forall j \qquad \beta_j := \beta_j - \alpha\frac{1}{n}\sum_{i=0}^{n}(f(x_i) - y_i)x_i$$

Run the gradient descent algorithm using the following **learning rates**: $\alpha \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$. For each value of α, run the algorithm for exactly **100 iterations**. Compare the convergence rate when α is small versus large. What is the ideal learning rate to obtain an accurate model? In addition to the nine learning rates above, come up with **your own choice** of value for the learning rate. Then, using this new learning rate, run the algorithm for your own choice of number of iterations.

Implement your gradient descent in a file called `problem2.py`, which will be executed like so:

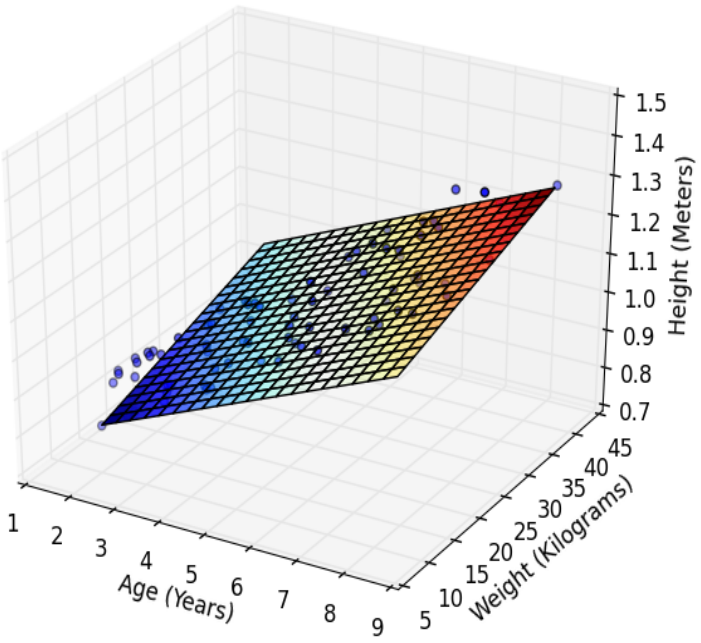
```
$ python problem2.py input2.csv output2.csv
```

If you are using Python3, name your file `problem2_3.py`, which will be executed like so:

```
$ python3 problem2_3.py input2.csv output2.csv
```

This should generate an output file called `output2.csv`. There are **ten cases** in total, nine with the specified learning rates (and 100 iterations), and one with your own choice of learning rate (and your choice of number of iterations). After each of these ten runs, your program must print a new line to the output file, containing a comma-separated list of the coefficients **b_0**, **b_age**, and **b_weight** in that order (see **example**), expressed with as many decimal places as you please. These represent the regression models that your gradient descents have computed for the given dataset.

What To Submit. `problem2.py` or `problem2_3.py`, which should behave as specified above. Before you submit, the **RUN** button on Vocareum should help you determine whether or not your program executes correctly on the platform.



Optional. To visualize the result of each case of gradient descent, you can use **matplotlib** to output an image for each linear regression model in three-dimensional space. For instance, you can plot each feature on the xy-plane, and plot the regression equation as a plane in xyz-space. An example is shown above for reference.