Gradient Descent and Linear Regression

In the process of fitting a curve to the data, we utilize the methods of gradient descent and regression since not only the prices are know but also all data points are actually categorized into two groups: successful and failed, denoted by 1 and 0 respectively. The variables that may influence the price (also the final result) of a task are: The average distance (D_{avg}) from the task to the people nearby, the weighted average credit value of these people (C_{wavg}) and the weighted number of people(N). When the price of the tasks is simulated, we generally use the linear regression model with the least square method to minimize the cost function $J(\theta)$, which are given by:

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...\theta_n x_n$$
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

where x_i are different variables and θ_i are parameter that we need to determine by gradient descent. $\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$ is the regularization part in order to avoid overfitting.

Hence, the gradients would be

$$\nabla J_i(\theta) = \frac{\partial J(\theta)}{\partial \theta_i} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)} + \frac{\lambda}{m}\theta_i$$

Therefore, each time we would perform

$$\begin{aligned}\theta_i :&= \theta_i - \alpha(\frac{1}{m}\sum_{j=1}^{m}(h_\theta(x^{(j)}) - y^{(j)})x^{(j)} + \frac{\lambda}{m}\theta_j) \\ &= \theta_i - \frac{\alpha}{m}\sum_{j=1}^{m}(\nabla J_j(\theta))\end{aligned} \tag{1}$$

However, considering that m is very large, we would like to improve the efficiency of the original batch gradient decent by the performing the stochastic gradient decent, which means that instead of summing up all the gradients and take average value, our method is to take one random gradient each time as the average gradient:

$$\theta_i := \theta_i - \alpha\nabla J_j(\theta))$$

where j is a stochastic index number given by uniform distribution among the inters [1,2,3,...m].

Variables

The three things we suggested above that may affect the price of the tasks are D_{avg}, C_{wavg} and N

D_{avg}: The average distance from all people to the task. It should be expected that larger D_avg will result in higher price.

C_{wavg} The weighted average credit value of all the people to the task. The weight of person (i) to the task is given by $w_i = \frac{min(D(i), 1\leq i\leq n)}{D(i)}$, where D is the vector that stores all the distances from the users to the task and P is the number of users. Then $C_{wavg} = \sum_{i=1}^{P} w_i * C(i)$. In the gradient descent procedure, a normalization process is also carried out in order to balance the numerical values of different variables to fasten the speed of descent. To be precise, the new C_{wavg} is given by:

$$C_{wavg} := \frac{C_{wavg} - min(C_{wavg})}{max(C_{wavg}) - min(C_{wavg})}$$

N is the weighted number of users for the task. The weight is also given by the distance weight w_i, which means that if a user is too far away from the task, it would be highly unlikely that he choose to accomplish it. Hence $N = \sum_{i=1}^{P} w_i$. It is worth noticing that although the number of users may affect the price, as it increases, the influence should decrease dramatically since the likelihood that people choose it increases rapidly. Therefore when we run the procedure, we tried $\sqrt{n}$, $\sqrt[3]{n}$, $\sqrt[4]{n}$ and $log(n)$ as one of the variables and finally find that $log(n)$ generates a much better simulation.

Results

Since the number of variables and the variables themselves vary in different regression models, we can only perform trial and error to obtain the best result we can find. However, when we include more higher order terms (such as D_{avg}^3), the performance of the simulation does not appear to be improved (see examples in picture), which makes the process easier. The credit variable (C_{wavg}) shows similar property while the function $log(n)$ takes much more time to determine. After many experiments, we obtain a relatively stable result with understandable parameter values.

$$\begin{aligned} h_\theta(x) = h_\theta(x) &= \theta^T x \\ &= \theta_0 + \theta_1 D_{avg} + \theta_2 C_{wavg} + \theta_3 log(N) + \theta_4 D_{avg}^2 \\ &= 40.9972 + 82.0736 D_{avg} - 8.5672 C_{wavg} - 3.3413 log(N) - 15.3289 D_{avg}^2 \end{aligned} \tag{2}$$

By seeing the picture of the simulation, we can draw the conclusion that it provides a good fitting of the original price. The reason why some differences exist is because the original price vary in a very small interval and frequently arrive at its minimum and maximum. This phenomenon may result from some price boundary settings($Price_{max}$ and $Price_{min}$) in the application.

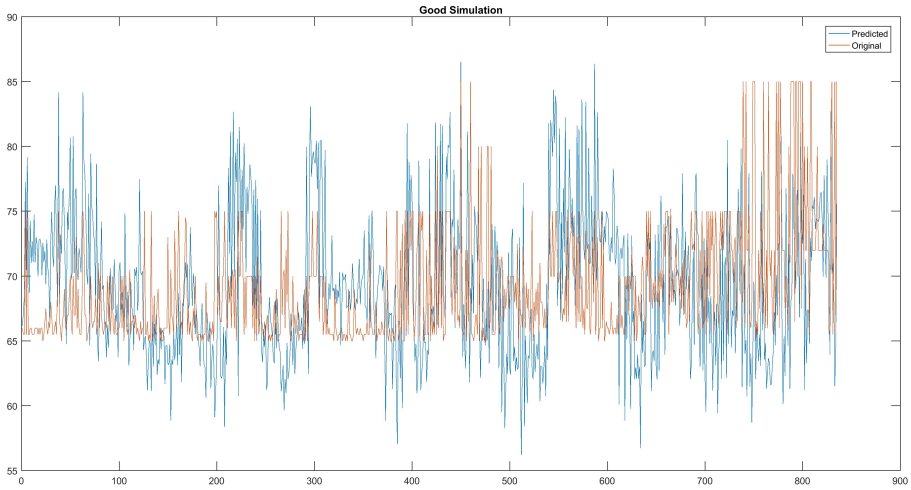

Figure 1: A reasonable simulation

Drawbacks of this pricing scheme

In this pricing scheme, we have taken into consideration the distances, the credit value(which can represent the quota of each person) and the number of people near the task. However, this pricing scheme ignores the detailed situation around each task, i.e. it only takes average on people's quota, distances and the number of people without analyse

them case by case. Therefore, we would have the motivation of building the second model to carefully combine all the information available and do the analysis.

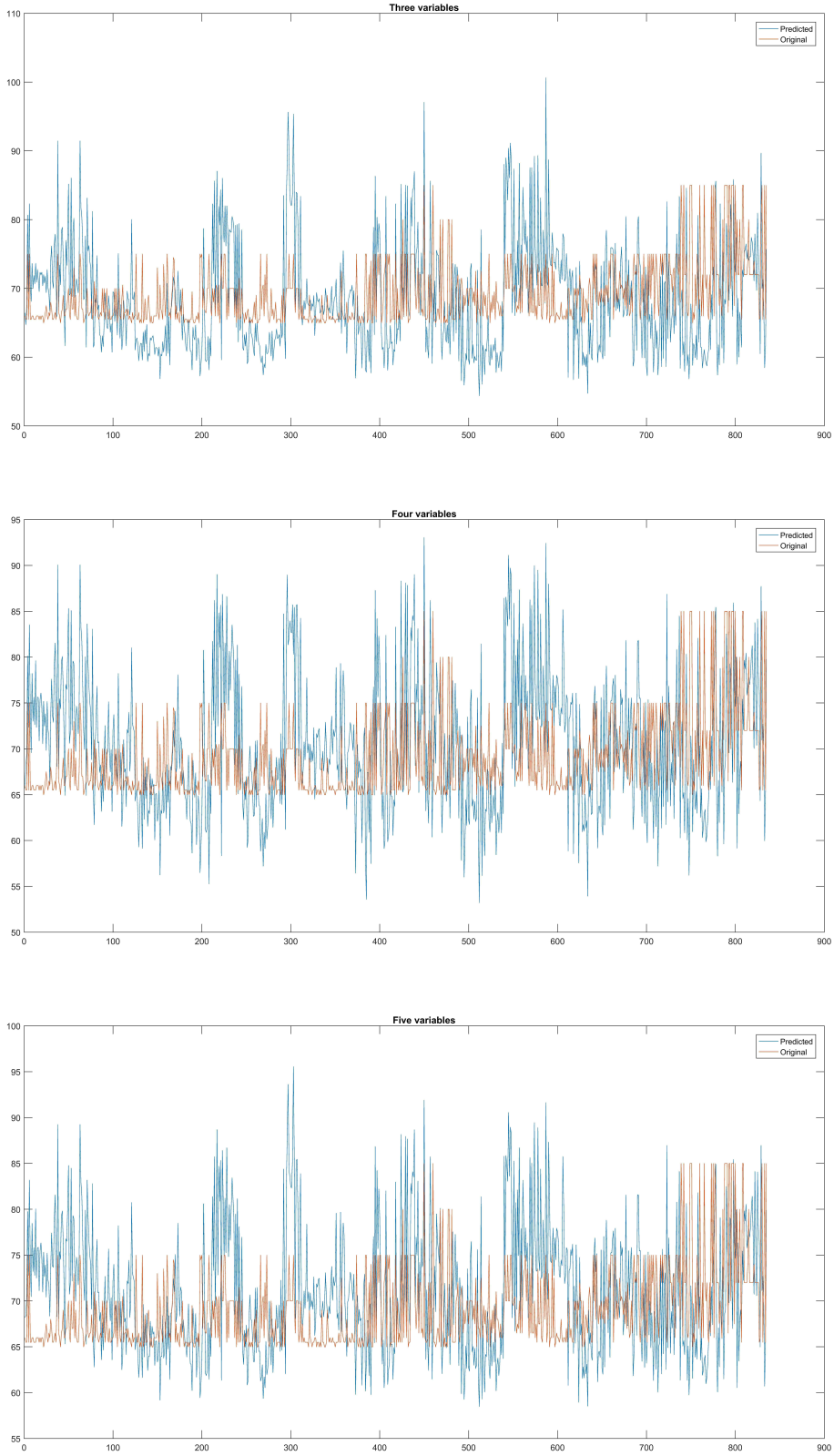

Figure 2: The variables

Packing

The packing of the tasks involves many criteria and we propose a method of packing which utilizes the matrix of distances between two tasks (TD)

$$TD(i,j) = \sqrt{(lat(i) - lat(j))^2 + (lon(i) - lon(j))^2}$$

where $(lat(i), lon(i)), (lat(j), lon(j))$ are the positions of the two tasks.

Upon calculation, it would be easy to observe that TD is a symmetric matrix and we only need to calculate the upper triangular matrix T, where

$$T'(i,j) == \begin{cases} \sqrt{(lat(i) - lat(j))^2 + (lon(i) - lon(j))^2}, & i < j \\ 0, & i \geq j \end{cases}$$

$$TD = T + T'$$

After calculating the matrix TD, we are able to set up the rules for packing, which are:

1. The number of tasks packed should never be larger than a number boundary c
2. For a task t_i in the package, it must be at least very close to another task (t_j), i.e. $T(i,j) < r$ for some distance boundary r.

According to these packing rules, we can construct a packing procedure as follows:

Algorithm of Packing

```
for i=1 : size do
    if t_i is not checked
        Check it
        while the boundary c is not reached
            Find the task t_j that is closest to task t_i
            if T(i,j) < r
                pack t_i with t_j.
                t_i:=t_j and continue.
            else
                break
            end if
        end while
    end if
end for
```

After the packing procedure, we calculate the average location of the packed tasks and regard them as one single task located at the average location of them. Then, we would substitute the position information into our second model and obtain the predicted price.

A possible pair of parameter values are $[c = 5, r = 0.01]$ since they give a moderate packing amount in our experiments while preserve the consistency of the whole system. Therefore, we would continue to use these two values in question 4.