ML2016_hw1_report

姓名: 徐有慶 學號: r05922162

1.Linear regression function by Gradient Descent.

```
double LinearRegression::lossFunction(vector<vector<double> > train_set,
vector<double*>    &past_gradients)
{
    double error_value = 0;
    int parameters_num = parameters.size();
    double *gradients = new double [parameters_num];
    for(int i = 0; i < parameters_num; i++)
    {
        gradients[i] = 0;
    }
    for(int m = 0; m < train_set.size(); m++)
    {
        double y_head = y_heads[m];
        double features[parameters_num] = {0};

        features[0] = 1;
        for(int i = 1; i < parameters_num; i++)
```

```
        {
            features[i] = train_set[m][i-1];
        }

        double y = 0;
        for(int i =0; i < parameters_num; i++)
        {
            y = y + parameters[i]*features[i];
        }

        for(int i = 0; i < parameters_num; i++)
        {
            gradients[i] = gradients[i] + (-2)*(y_head - y) * features[i];    //compute
every feature's gradient
        }

        error_value = error_value + (y_head - y)*(y_head - y);
    }
     /***** regularization *****/
    double lambda = 100; //100
    if(lambda!=0)
```

```
	{
		double sigma_w_square = regularization(gradients, lambda);
		error_value = error_value + lambda*sigma_w_square;
	}
	past_gradients.push_back(gradients);
	gradientDescent(gradients, past_gradients);

	return error_value;
}

void LinearRegression::gradientDescent(double gradients[], vector<double*>
past_gradients)
{
	int gradients_num = parameters.size();
	double learning_rate = 1;
	double sigma_past[gradients_num] = {0};

	for(int i = 0; i < past_gradients.size(); i++)
	{
		for(int j = 0; j < gradients_num; j++)
		{
```

```
                sigma_past[j] = sigma_past[j] + past_gradients[i][j] *
past_gradients[i][j];      // Agagrad, compute the past gradient's sum
            }
        }

        for(int i = 0; i < parameters.size(); i++)
        {
            parameters[i] = parameters[i] - learning_rate * gradients[i] /
sqrt(sigma_past[i]);    // Adjust the weight
        }
}
```

2.Describe your method

取每項的前 5 個小時來預測第 6 個小時 PM2.5 的值，每天有 18 項，所以會取 90 筆資料當作 features 來訓練，而要跑 test set 時就只取最後的 5 個小時來當作 features。初始 weight 則在 0~0.01 當中隨機產生出來，會這樣選的原因在於這樣子算出來的 y 值會比較符合實際上 PM2.5 的值。在本次作業中使用 Adagrad 來動態調整 learning rate，並加上了 regularization 且利用控制迭代次數來訓練 model。

3. Discussion on regularization

λ	iteration	Training	Testing
0	10000	4637.68	1094.312
10	10000	4743.52	1091.142
100	10000	6719.98	1057.485
1000	10000	12088.7	1329.134
10000	10000	12088.7	1329.0161
0	15000	4490.52	1111.204
10	15000	4618.78	1104.338
100	15000	5020.77	1086.408
1000	15000	6711.31	1054.990
10000	15000	12088.7	1329.062

利用 data 資料夾內的 own_train 及 own_test_X 做出 Training 及 Testing 的結果。由此可知，在相同迭代次數下雖然 lambda 值較大，訓練出的 model 錯誤率也較高，在 test set 的結果當中卻可能會較佳，但 lambda 值過大也不太好

4.Discussion on learning rate

Learning rate 的部份，在沒有使用 Adagrad 時，必須設很小(ex. 0.00000001)才能成功的將錯誤率給降低，但也因為固定的 learning rate 會導致收斂速度變慢許多，所以之後便加上了 Adagrad 來增加收斂的速度。