CM146 HW#2

Zhaoxing Deng, 005024802

February 6, 2018

1 Perceptron

(a) Let two inputs to be x_1, x_2. To remove the bias term b, let $\mathbf{x} = (1, x_1, x_2)$.

x$_1$	x$_2$	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Let perceptron $\theta = (\theta_1, \theta_2, \theta_3)$, y = sgn($\theta^T \boldsymbol{x}$). Then $\theta = (-1, 1, 1)$ and $\theta = (-2, 1, 1.5)$ are different valid perceptron.

(b) The XOR table is like the following.

x$_1$	x$_2$	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	1

No perceptron exists. The data is not linearly seperable.

2 Logistic Regression

(a)

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial(-\sum_{n=1}^{N}(y_n \log h_\theta(\mathbf{x}_n) + (1-y_n)\log(1-h_\theta(\mathbf{x}_n))))}{\partial \theta_j} = -\sum(y_n \frac{\partial \log h_\theta(\mathbf{x}_n)}{\partial \theta_j} + (1-y_n)\frac{\partial \log(1-h_\theta(\mathbf{x}_n))}{\partial \theta_j})$$

$$\because h_\theta(\mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$\therefore \frac{\partial h_\theta(\mathbf{x})}{\partial \theta_j} = \frac{\partial \sigma(\theta^T \mathbf{x})}{\partial \theta_j} = \frac{\partial \sigma(\theta^T \mathbf{x})}{\partial(\theta^T \mathbf{x})}\frac{\partial(\theta^T \mathbf{x})}{\partial \theta_j} = \sigma(\theta^T \mathbf{x})(1-\sigma(\theta^T \mathbf{x}))\frac{\partial(\sum \theta_i x_i)}{\partial \theta_j} = h_\theta(\mathbf{x}_n)(1-h_\theta(\mathbf{x}_n))x_j$$

$$\frac{\partial \log h_\theta(\mathbf{x}_n)}{\partial \theta_j} = \frac{\partial \log h_\theta(\mathbf{x}_n)}{\partial h_\theta(\mathbf{x}_n)}\frac{\partial h_\theta(\mathbf{x}_n)}{\partial \theta_j} = \frac{1}{h_\theta(\mathbf{x}_n)}h_\theta(\mathbf{x}_n)(1-h_\theta(\mathbf{x}_n))x_j = (1-h_\theta(\mathbf{x}_n))x_j$$

$$\frac{\partial(\log(1-h_\theta(\mathbf{x}_n)))}{\partial \theta_j} = \frac{\partial \log(1-h_\theta(\mathbf{x}_n))}{\partial(1-h_\theta(\mathbf{x}_n))}\frac{\partial(1-h_\theta(\mathbf{x}_n))}{\partial \theta_j} = -\frac{1}{1-h_\theta(\mathbf{x}_n)}h_\theta(\mathbf{x}_n)(1-h_\theta(\mathbf{x}_n))x_j = -h_\theta(\mathbf{x}_n)x_j$$

$\therefore \frac{\partial J}{\partial \theta_j} = -\sum(y_n(1-h_\theta(\mathbf{x}_n))x_j - (1-y_n)h_\theta(\mathbf{x}_n)x_j) = \sum_{n=1}^{N}(h_\theta(\mathbf{x}_n)-y_n)x_{n,j}$

3 Locally Weighted Linear Regression

(a) $\frac{\partial J}{\partial \theta_0} = \sum_{n=1}^{N} 2w_n(\theta_0 + \theta_1 x_{n,1} - y_n)$

$\frac{\partial J}{\partial \theta_1} = \sum_{n=1}^{N} 2w_n(\theta_0 + \theta_1 x_{n,1} - y_n)x_{n,1}$

(b) Let $\frac{\partial J}{\partial \theta_0} = 0$ Then $\sum_{n=1}^{N} 2w_n(\theta_0 + \theta_1 x_{n,1} - y_n) = 0$

$\sum w_n\theta_0 + \sum w_n\theta_1 x_{n,1} - \sum w_n y_n = 0$

$(\sum w_n)\theta_0 + (\sum w_n x_{n,1})\theta_1 = \sum w_n y_n$ (1)

Let $\frac{\partial J}{\partial \theta_1} = 0$ Then $\sum_{n=1}^{N} 2w_n(\theta_0 + \theta_1 x_{n,1} - y_n)x_{n,1} = 0$

$(\sum w_n x_{n,1})\theta_0 + (\sum w_n x_{n,1}^2)\theta_1 = \sum w_n x_{n,1} y_n$ (2)

From(1)and(2), we can get the solution for these linear equations.

$\hat{\theta}_0 = \frac{\sum_{n=1}^{N} w_n x_{n,1}^2 \sum_{n=1}^{N} w_n y_n - \sum_{n=1}^{N} w_n y_n x_{n,1} \sum_{n=1}^{N} w_n x_{n,1}}{\sum_{n=1}^{N} w_n \sum_{n=1}^{N} w_n x_{n,1}^2 - (\sum_{n=1}^{N} w_n x_{n,1})^2}$

$\hat{\theta}_1 = \frac{\sum_{n=1}^{N} w_n \sum_{n=1}^{N} w_n y_n x_{n,1} - \sum_{n=1}^{N} w_n x_{n,1} \sum_{n=1}^{N} w_n y_n}{\sum_{n=1}^{N} w_n \sum_{n=1}^{N} w_n x_{n,1}^2 - (\sum_{n=1}^{N} w_n x_{n,1})^2}$

4 Understanding Linear Separability

(a) Because the data set is linear seperable, there exists a $\mathbf{u}$and θsuch that $y_i(\mathbf{u}^T\mathbf{x}_i + \theta) > l$, where l is the margin of the data set.

$\because l > 0$ divide the both sides of the inequality, we get $y_i(\frac{\mathbf{u}}{l}^T\mathbf{x}_i + \frac{\theta}{l}) > 1$, which shows that there is an optimal solution to the linear program

(b) If there is an optimal solution . Let $y_i(\mathbf{w}^T\mathbf{x}_i + \theta) > 1$ for all $(\mathbf{x}_i, y_i) \in D$. This means that $y_i(\mathbf{w}^T\mathbf{x}_i + \theta) > 0$, that is $y_i$has the same sign as $\mathbf{w}^T\mathbf{x}_i + \theta$

$\therefore$Define $y_i = \begin{cases} 1 & \mathbf{w}^T\mathbf{x}_i + \theta \geqslant 0 \\ -1 & \mathbf{w}^T\mathbf{x}_i + \theta < 0 \end{cases}$, we can say the data set is linear seperable.

(c) If $\delta \in (0, 1)$, the data set is seperable, using the same argument in (b). Otherwise, we can say whether it is linear seperable or not.

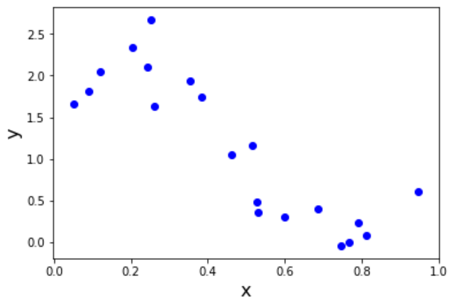
(d) Let $\mathbf{w} = \mathbf{0}$, $\theta = 0$, then it is an optimal solution to this formulation, because $y_i(\mathbf{w}^T\mathbf{x}_i + \theta) = 0$ for any D. The problem is this solution won't give us any information about the data set.

(e) $\mathbf{w} = (\mathrm{a}, \mathrm{b}, \mathrm{c})$, $\theta = 0$, where$a + b + c = 1$ are optimal solutions.

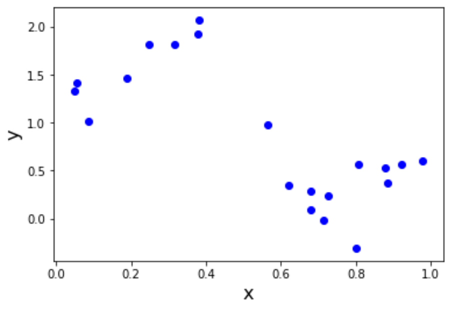
5 Implementation: Polynomial Regression

Visualization

(a)



This graph shows the plot of the training data.



This graph shows the plot of the test data. It seems like the data is not well linear seperable. So I suppose polynomial regression may have better results.

Linear Regression

(b) Modify PolynomialRegression.generate_polynomial_features(...)

(c) Complete PolynomialRegression.predict(...)

(d)

Returned 40.2338474097

η	coefficients	number of iterations	cost	time
0.0407	[-9.40470931e+18 -4.65229095e+18]	10000	2.71091652001e+39	1.636999
0.01	[2.44640703 -2.81635347]	765	3.91257640579	0.1299998
0.001	[2.4464068 -2.816353]	7012	3.91257640579	1.177999
0.0001	[2.27044798 -2.46064834]	10000	4.0863970368	1.703999

The coefficients for the converge situation is [2.44640703 -2.81635347].The larger step size, the quicker the algorithm converge. When $\eta = 0.0407$, the step size is too large that the algorithm does not converge.

(e)

η	coefficients	number of iterations	cost	time
Closed form	[2.44640676 -2.81635292]		3.91257640579	0.000999927520752

The coefficients and cost match up with those obtained by GD, but it's much more quicker.

(f)

η	coefficients	number of iterations	cost	time
1/(1+k)	[2.44640676 -2.81635292]	1357	3.91257640579	0.230999946594

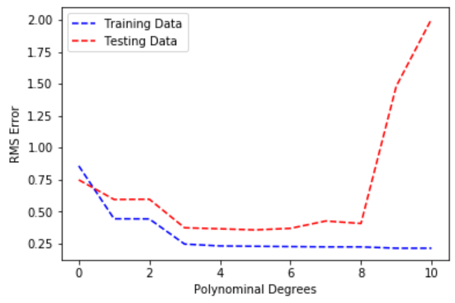
It took 1357 iterations and about 0.231s to converge.

Polynomial Regression

(g) Update PolynomialRegression.generate_polynomial_features(...)

(h) I think RMS error is better than cost function, because RMS error provide the square root of average, while cost function just add the error's squares. Also, cost function will grow whenever we add new data points, but RMS may decrease based on the data distribution.

(i)



Five degree polynominal best fits the data. When the degree is less than 2, the data is unfitted, since the error is so large and with the degree goes up the error decreases to a lower level. When the degree is larger than 8, overfitting happens, because the testing error climbs up quickly and trainning error remains small.