CSC411 Assignment 1

Yue Guo

October 5, 2017

1 Learning basics of regression in Python

1.1 Describe and summarize the data

Dimension: 13
Target: price
Data points: for each feature, we have 506 data points

1.2 visualization

1.3

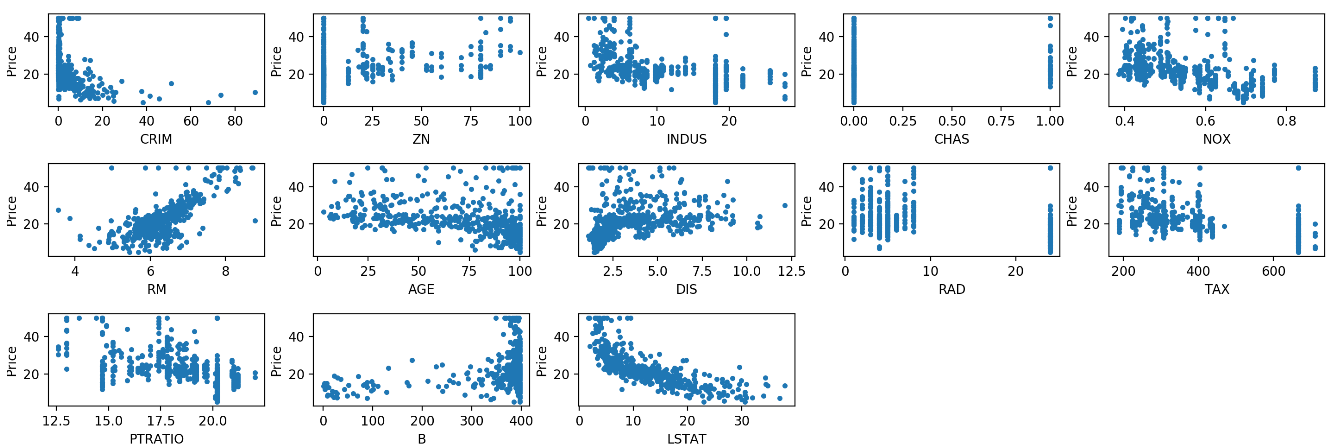

Figure 1:

1.3.1 Feature weights

Weights of each feature:

CRIM	39.3306546864
ZN	-0.105570660997
INDUS	0.033569463917
CHAS	0.0501338462503
NOX	2.44159672082
RM	-18.9563291605
AGE	3.65717113479
DIS	0.00193877592741
RAD	-1.46699325228
TAX	0.349594800713
PTRATIO	-0.0145786907583
B	-0.959592750851
LSTAT	0.008452561222

INDUS matches my expectation. The more business we have, the more prosperous an area is, therefore more expensive housing.

1.3.2 MSE of my model

19.0490487755

1.3.3 Two more error measurement

normal error = 313.546384855
mean square root= 0.281596677525
I suggest these two error measurements because they do not square the differences.

1.3.4 Most significant feature

Based on my results, the most significant feature is RM and CRIM. It has larger weight value among all features.

2 Locally weighted regression

2.1 weighted least square problem and analytic solution proof

Since the matrix A is diagonal and $\hat{y} = A^T x$ and $L(w) = \frac{1}{2}\Sigma a^{(i)}(y^{(i)} - W^T x^{(i)})^2 + \frac{\lambda}{2}\|W\|^2$
$L(w) = \frac{1}{2}A[(y - W^T x)(y - W^T x)] + \frac{\lambda}{2}\|W\|^2$
$L(w) = \frac{1}{2}A(y^T y + W^T X^T XW - 2W^T X^T y) + \frac{\lambda}{2}\|W\|^2$
$\frac{\partial}{\partial w} = \frac{1}{2} \times 2A[X^T XW^* - X^T y] + \lambda\|W\| = 0$
$AX^T XW^* - X^T Ay + \lambda W^* = 0$
$(AX^T X + \lambda)W^* = X^T Ay$
$W^* = X^T Ay(AX^T X + \lambda I)^{-1}$

2.2

x is Tau, y is losses

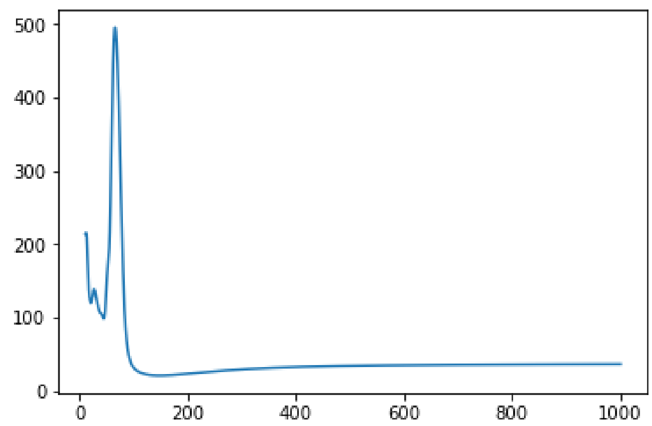

Figure 2:

2.3

The algorithm produces results or weights of each feature with a large variance if τ -> 0, and variance -> constant if τ-> inf

3 Mini-batch

3.1 Proof of expected value of mini batches

$RHS = \frac{1}{n}\Sigma_{i=1}^{n} a_i$ is the average of all samples in the data set
LHS: $\frac{1}{m}\Sigma_{i=1}^{m} a_i$ is the average of each random batch. Since each batch is drown randomly from the dataset, the m elements each have $\frac{1}{n}$ chance to be drawn. The probability of the batch is $\frac{m}{n}$. $E(\frac{1}{m}\Sigma a_i) = \frac{m}{n} \times \frac{1}{m}\Sigma a_i = \frac{1}{n}\Sigma_{i=1}^{n} a_i$
QED

3.2 Proof of gradients

From the result of part 1, substitute l into a_i
$E[\frac{1}{m}\Sigma l(x, y, \theta)] = \frac{1}{n}\Sigma l(x, y, \theta)$
$E[L(x, y, \theta)] = L(x, y, \theta)$
apply gradient, we have
$\nabla E[L(x, y, \theta)] = \nabla L(x, y, \theta) = \frac{1}{n}\Sigma_{i=1}^{n} \nabla l$
$\nabla \frac{1}{n}\Sigma_{i=1}^{n} l = \nabla E(\frac{1}{m}\Sigma a_i)$
$E[\nabla L(x, y, \theta)] = \nabla L(x, y, \theta)$

3.3 Importance of this result

This implies that k random batches of data can approximate the gradient of the complete dataset.

3.4 Gradient

3.4.1 Analytic solution

$\nabla L = 2X^T Xw - 2X^T y$

3.4.2

see q3.py

3.5 Error measurements

square metric = 79165708.6263
cosine similarity = 0.999998432222
I suggest cosine similarity because square distance takes the difference to the power of 2, which punishes certain cases more.

3.6 plot

X axis is weights, y axis is log of M

this is the graph if we average all the weights

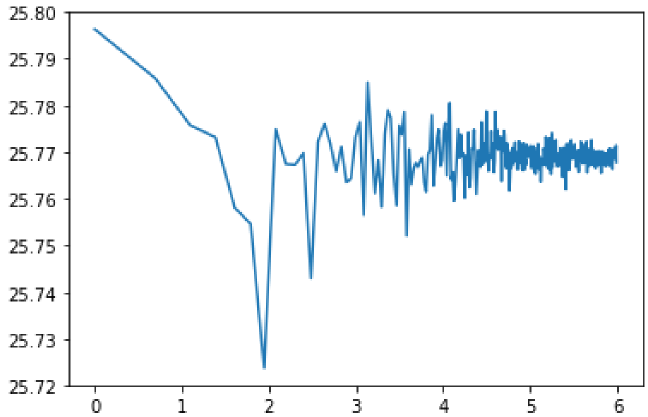

Figure 3:

This is the graph if we average each w_j X axis is weights, y axis is log of M

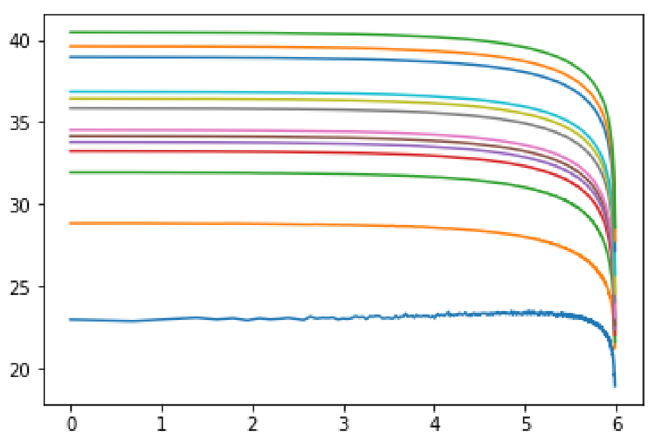

Figure 4: