# CS6140 Assignment2 Guanglei “Garrett” Wu

2.1

# Housing:

RMSE for training data:

[4.674250141206967, 4.725606283576037, 4.832539703752558, 4.733448286013331, 4.8486644099027325, 4.596485145842515, 4.866209075613999, 4.6978100603824675, 4.715356726005594, 4.885759754107733]

Mean:

4.75761295864

RMSE for testing data:

[5.774980290155129, 5.284103306215706, 3.9631107091566893, 5.192152100619355, 4.01108843461127, 6.2382822735983385, 3.827224455640817, 5.466517138890734, 5.220961743322239, 3.465990152650557]

Mean:

4.84444106049

SSE for training data:

Mean:

10325.2450094

Standard Deviation:

392.654454453

SSE for testing data:

Mean:

1213.71908557

Standard Deviation:

430.421881241

RMSE trend for fold-0:



# Yacht:

RMSE for training data:

[8.817569832886193, 8.823200920283846, 8.837891441303913, 9.096531381143118, 8.880133196472467, 8.742699926269658, 8.849852717901845, 9.021582963392325, 8.589493093504787, 9.038433342556148]

Mean:

8.86973888157

RMSE for testing data:

[9.383337556007792, 9.278400264660954, 9.122134398610617, 6.330221903694199, 8.775297831431228, 10.03544794787197, 9.049295710688428, 7.244267702790189, 11.397793819042516, 7.0980903826821224]

Mean:

8.77142875175

SSE for training data:

Mean:

21876.5625187

Standard Deviation:

703.605722581

SSE for testing data:

Mean:

2369.39581942

Standard Deviation:

750.779459384

RMSE trend for fold-0:



# Concrete:

RMSE for training data:

[10.267028918352064, 10.380994362789213, 10.324921385933735, 10.413467239237676, 10.382721885018885, 10.342588533209176, 10.322318834355045, 10.443533421822064, 10.320939955834254, 10.288138999369258]

Mean:

10.3486653536

RMSE for testing data:

[11.288436265172287, 10.134023978527381, 10.679773219158413, 9.88339063321222, 10.170428747470753, 10.660194548649764, 10.695300128353631, 9.643269180672835, 10.692180286772738, 10.994934849873697]

Mean:

10.4841931838

SSE for training data:

Mean:

99279.5254194

Standard Deviation:

1012.2120265

SSE for testing data:

Mean:

11345.8307644

Standard Deviation:

1043.87753539

RMSE trend for fold-0:



2.2

2.2.1,

As we set the tolerance value as the stop point, different selection of w does not matter much on the fitness of the model. However, initial w do influence time of iterations to converge. Housing for example, it seems choosing initial w near actual w will get smaller number of iterations.

Housing iterations mean test RMSE

w -1 23 4.85

-0.5 24 4.87

0 25 4.85

0.5 29 4.89

1 31 4.93

2.2.2

Smaller tolerance makes model slower to converge, but with better training and testing fitness. While larger tolerance makes model converge faster, but worse in fitness.

Housing example:

Housing iterations mean test RMSE

tolerance 0.5e-5 268 4.82

0.5e-4 163 4.79

0.5e-3 74 4.82

0.5e-2 25 4.94

0.5e-1 15 4.99

0.5 9 5.71

2.2.3

Larger learning rate do accelerate the speed of gradient descent. However, too large learning rate causes the model fail to converge and fail to produce any results.

Housing iterations

LR 0.5e-4 138

0.1e-3 78

0.2e-3 44

0.4e-3 26

3

# Housing:

RMSE for training data:

[4.649895496764158, 4.544808850225311, 4.58841863288941, 4.628884892282834, 4.690926688899657, 4.798668989496345, 4.743048324479664, 4.74321563962085, 4.659845334045983, 4.599713944332414]

Mean:

4.6647426793

RMSE for testing data:

[5.111095045048797, 5.864088143604326, 5.572662724895493, 5.3108958650541, 4.620893417358014, 3.4898819158375467, 4.190682162638951, 4.206776611048698, 4.942435568863504, 5.390291257645013]

Mean:

4.8699702712

Comparing with gradient descent method, normal equation provides slightly better performance for both training and testing data. This may because gradient descent stops by meeting the tolerance instead of finding the real min.

# Yacht:

RMSE for training data:

[8.604452449595964, 9.055656044492213, 8.748517384097445, 9.04959845410558, 8.91204137296486, 8.754915439371272, 8.771044140018406, 8.819938274959162, 8.92830951338483, 8.79272747375524]

Mean:

8.84372005467

RMSE for testing data:

[11.17411734502676, 6.978188319638096, 9.955581728764745, 7.017955991205116, 8.405169323583735, 9.77697251533944, 9.764218719088879, 9.21091148972869, 8.392123047630033, 9.517042627555726]

Mean:

9.01922811076

Comparing with gradient descent method, normal equation provides slightly better performance for training data but slightly worse performance for testing data. This may caused by the noises of the testing data.

4

y = fw(x) = w0 + w1x

J = ½ \* Σ ( w0 + w1x – y )2

∂J/∂w0 = Σ ( w0 + w1x – y )

∂J/∂w1 = Σ ( w0 + w1x – y ) x1

In minimizing SSE, set both derivatives to 0, we have:

Nw0 + Σx w1 - Σy = 0 => w0 = (Σy - Σx w1) / N = E(y) – E(x) w1

Σx w0 + Σx2 w1 - Σxy = 0 , substitute w0 into, we have:

Σx (Σy - Σx w1) / N + Σx2 w1 = Σxy, =>

w1 = (NΣxy - ΣxΣy) / (NΣx2 – (Σx)2) = ( E(xy) – E(x)E(y) ) / ( E(x2) – E(x)2 )

w0 = E(y) – E(x) w1 = ( E(y)E(x2) – E(x)E(xy) ) / ( E(x2) – E(x)2 )

5

5.1

# Sinusoid:

training:

[4.884782489157089, 4.199967464106566, 2.13817079990268, 1.8659482331638377, 1.6587351464351052, 1.6293022379843642, 1.6131098619249205, 1.613094055775583, 1.6093155618704995, 1.6074239376652188, 1.6007956578522335, 1.6017840572626534, 1.5984020765948415, 1.597876297136764, 1.5974645545939237]

testing:

[4.182116031703597, 3.5952201531073866, 1.180783826008107, 0.9361581788765945, 0.6080862208769404, 0.6394299711916872, 0.7998052975161498, 0.8012414747866585, 0.7849316749826861, 0.7848007067050877, 0.851603410004051, 0.8577933471865238, 0.8059433686119598, 0.811799126504297, 0.8234360099783155]



Surprisingly, the model performs better on testing data, this may because the bias of the original data set. The graph shows the cost is minimized at around max(p) = 5, before that point, it shows under fitting, after shows overfitting.

# Yacht:

training:

[8.8373219392056228, 4.0707424108182533, 1.8116159020329796, 1.4257889785738695, 1.4279902328159815, 1.4076194365631847, 1.407245146718106]

testing:

[9.0583057269988423, 4.2627134132018485, 1.9336347757610823, 1.5333788466999683, 1.5090146559475222, 1.5303936733689185, 1.4620451783817512]



Adding polynomial features improves the performance of Yacht dataset, especially when p is small. When p goes large, adding higher order polynomial features does not help the performance.

5.2

1)

When p is small, adding new features does improve the performance of both dataset. But when p is large, continue adding new features does not seem help.

The 2 datasets show slightly different impacts.

For yacht, adding new features is effective for p < 4. After that, adding new features does no impact on the outcome.

For sinusoid, adding new features is effective for p < 5. The test result is even worse when adding new features after that. This may because of over fitting.

2)

Obviously, when current number of features is relatively small or the dataset is large and complex, where current features are not enough to explain the data, adding higher order polynomial features or cross-terms is a good idea. However, if the features are enough to illustrate the dataset, continuing adding new features may not help, even causes over fitting problems. To determine how many features do we need, we may use a validation set in addition to training and testing sets to choose the appropriate model.

6

H = X (XTX)-1 XT

1,

HT = (XT)T (X (XTX)-1)T = X ( (XTX)-1 )T XT = X ((XTX)T)-1 XT = X (XTX)-1 XT = H

2,

HH = X (XTX)-1 XT X (XTX)-1 XT = X (XTX)-1 [(XT X) (XTX)-1 ]XT = X (XTX)-1 I XT = X (XTX)-1 XT = H

7

RMSE when p = 5



RMSE when p = 10



7.1

The 2 graphs look similar, smaller lambda makes both models fit the data better, as shown when lambda = 0. And when lambda increases, both models are less fitting to the data, because larger lambda constraints the model to fit, shown as under fitting. The 2 graphs do not show evidence of over fitting, may because of the size or characteristic of the dataset.

The difference between the 2 models is that as lambda grows, p = 10 grows slower than p = 5 in RMSE. This may because p10 has more features to explain the data, comparing to p5. So when implies constraint, p10 has more freedom to adjust its features, especially higher order features to the data than p5.

8

1,

pN = Π p(x; μ, σ) =

L = ln(pN) =

∂L/∂μ = , set to 0, we get , =>

2,

∂L/∂σ = , set to 0, we get ,

=