cs584 Assignment 1: Report

Xin Su
Department of Computer Science
Illinois Institute of Technology

February 19, 2016

Abstract

This report states the analysis of the Assignment 1's results. It resolves Problem 1 and Problem 2's totally 9 questions one by one. First I use a linear regression model to fit the datasets of single variable, then apply it to fit to the same datasets again, but using different polynomial models. Second I implement an iterative solution for the model to find the cofactors, that is gradient descent function, stochastic gradient descent function and Newton method. I also implemented a dual regression model with gaussian kernel function. Finally I apply these models to the multivariate datasets and figure out the difference between models and parameters.

1. Problem statement

I explored the following problems:

- How a linear regression model works and compare the results when applying to different datasets of single feature.
- What effect does it have when first applying transforming datasets to polynomial dimension then using linear regression model.
- What will the result be when applying linear regression model to datasets of multiple features?
- What will the result be to apply the linear regression model when adding additional dimensions by combining the features.
- What are the differences between explicit and iterative solutions (stochastic gradient descent function).
- What are the differences between dual regression and primal regression.

2. Proposed solution

- Linear regression model: I use the general explicit equation to calculate the cofactor matrix *theta* and predict the label *h(x^(i))*:
  theta = np.dot(np.linalg.pinv(X), y)
  h(x^(i)) = np.dot(X, theta)
- K Fold: I shuffle the index and cut them into K pieces. Each time I use K-1 pieces as training set and the remaining one piece as test set.

- Raising dimension: I add degree for all features and use them as new features. For example, if the current features are (x1, x2, x3, x4), after adding 2 degrees the all features will be (x1, x2, x3, x4, x1^2, x2^2, x3^2, x4^2, x1^3, x2^3, x3^3, x4^3).
- (Stochastic) Gradient descent: This is the assignment in the loop I use:
  *theta_1 = theta_0 - alpha * np.sum(np.dot(X[i].T, (predicted_0 - y[i])))*
- Gaussian kernel function: This is the equation to use to calculate the gram matrix by using the Gaussian kernel function:
  *G = np.exp((-1) * np.square(cdist(X, X, 'euclidean')) / sigma ** 2)*
- Newton method: This is the equation I calculate the step matrix (*alpha*) using the Newton method:
  alpha = np.linalg.inv((np.dot(X.T, X)))
- Objective *J*: This is the equation I use to calculation the *J* value:
  *J = np.sum(np.square(X - y)) / (2.0 * X.shape[0])*
- *rse*: This is the equation I use to calculate *rse* value:
  *rse = (np.sum(np.square(X - y) / np.square(y))) / (2.0 * X.shape[0])*
- I use sklearn as the ready made Python model to compare with my model

3. Implementation details

I use Python 2.7 as development language, and use IPython Notebook to write the code and run. It is very convenient for use. A user can view it simply using a browser, and the content will contain all output, including text, data and graphs.

When you want to run the code, you should first install the IPython Notebook (Please refer to the Jupyter official website: http://jupyter.readthedocs.org/en/latest/install.html). After this step, you can open the .ipynb file and run. Just run from the top cell to the bottom. This top-down order is required.

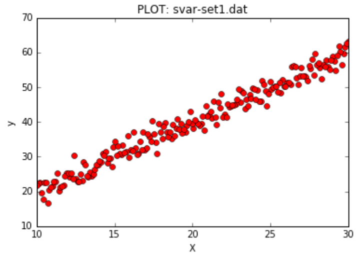
Below are some main functions:

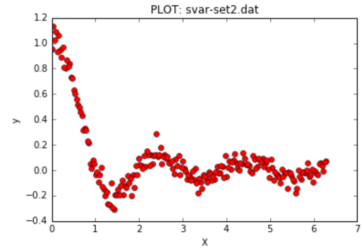
- *fit(X, y, sigma=0.0)*: this is the fit function both for linear regression model's explicit solutions and dual regression model's explicit solution. When the sigma is 0.0 it fits linear regression model to the data; when the sigma is greater than 0.0 it fits dual regression model to the data using the sigma as the parameter in the Gaussian kernel function.
- *run(fname, fit_model, n, n_folds=10, truncate=0.0, sigma=0.0, alpha=0.0, newton=False, epsilon=0.0, stochastic=False, p_verbose=False, r_verbose=False, pl_verbose=False)*: this function runs the model and apply it to the dataset with the file name "*fname*". The *fit_model* has three options:
  - *fit_model* = "explicit":
    - no additional parameter is given: linear regression model explicit solution
    - *alpha* and *epsilon* are given:
      - *newton* is given: using Newton method to calculate *alpha*
        - *stochastic* is given: linear regression model stochastic gradient descent
        - *stochastic* is not given: linear regression model gradient descent
    - *sigma* is given: dual regression with gaussian kernel function
- *batch_run(fnames, fit_model, n, n_folds=10, truncate=0.0, sigma=0.0, alpha=0.0, newton=False, epsilon=0.0, stochastic=False, p_verbose=False, r_verbose=False, pl_verbose=False, bpl_verbose=False)*: this function runs the model in batch mode:
  - *fnames*: dataset file names. The code will run on all the dataset files given.
  - *n*: degree n. The code will run on all degree from 1 to n.

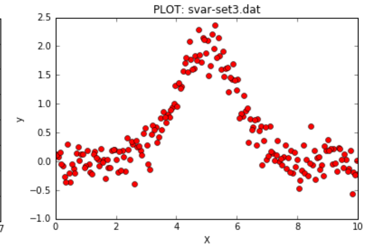
4. Results and discussion

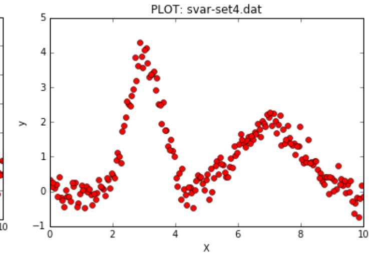
Problem 1

(a) The raw dataset graphs are below:









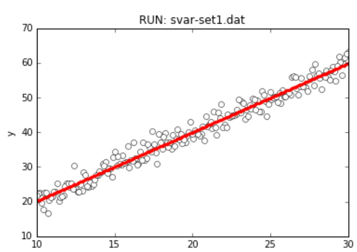
(b) This is the results table. From the data we can tell that the linear model fits the file “svar-set1.dat” very well as it has a very small *rse* both for the cross validation and all data.

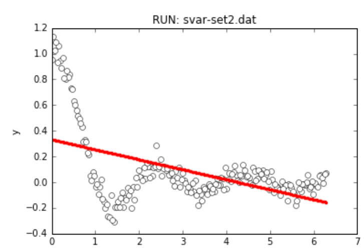
- The reason that the objective *J* of “svar-set1.dat” is greater than other datasets is because the range of the data in “svar-set1.dat” is larger than that of other data files. Therefore, we can tell that from here, the *rse* has more reasonable meaning when compare different datasets.
- The average *J* and *rse* values for training and test in cross validation is almost the same, also the *J* and *rse* values for all data is pretty the same. This means the cross validation does tell if the model runs well or not.

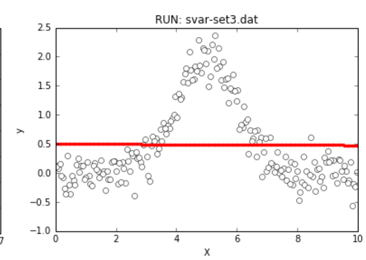
Linear regression result

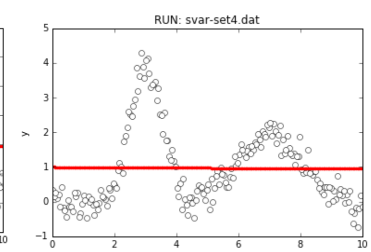
file name	cross validation avg training J	cross validation avg test J	cross validation avg training rse	cross validation avg test rse	All data J	All data rse
svar-set1.dat	2.1137099	2.1655015	0.0018394	0.0018859	2.1162853	0.0018416
svar-set2.dat	0.0297499	0.0304743	50.6271617	49.4665225	0.0297863	50.3872921
svar-set3.dat	0.2490306	0.2556703	48.7271635	51.7232227	0.2493569	48.7861839
svar-set4.dat	0.5999358	0.6093460	856.1220768	854.8711071	0.6004040	855.0362917

We can also see the same result that the linear model fits “svar-set1.dat” well. The graphs are below:

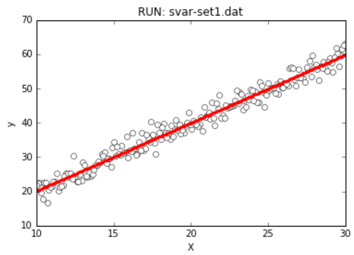


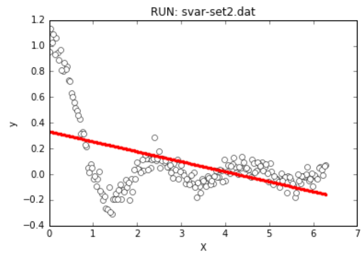


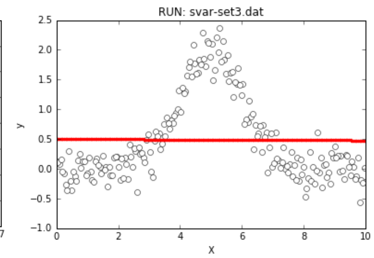


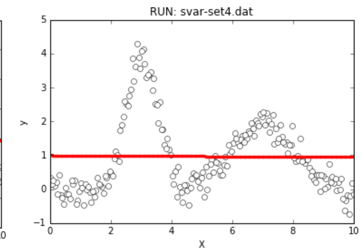


(c) Below is the result and graphs from using sklearn linear regression. I use sklearn to fit the model and predict, but use my own functions to calculate the *J* and *rse* so as to compare my result. The sklearn result is pretty like mine.









Sklearn linear regression result

file name	cross validation avg training J	cross validation avg test J	cross validation avg training rse	cross validation avg test rse	All data J	All data rse
svar-set1.dat	2.1127463	2.1843763	0.0018367	0.0019227	2.1162853	0.0018416
svar-set2.dat	0.0297436	0.0306003	50.1718258	53.9032082	0.0297863	50.3872921
svar-set3.dat	0.2492086	0.2521696	48.7588106	50.0269966	0.2493569	48.7861839
svar-set4.dat	0.6001013	0.6061435	859.7622905	822.2437717	0.6004040	855.0362917

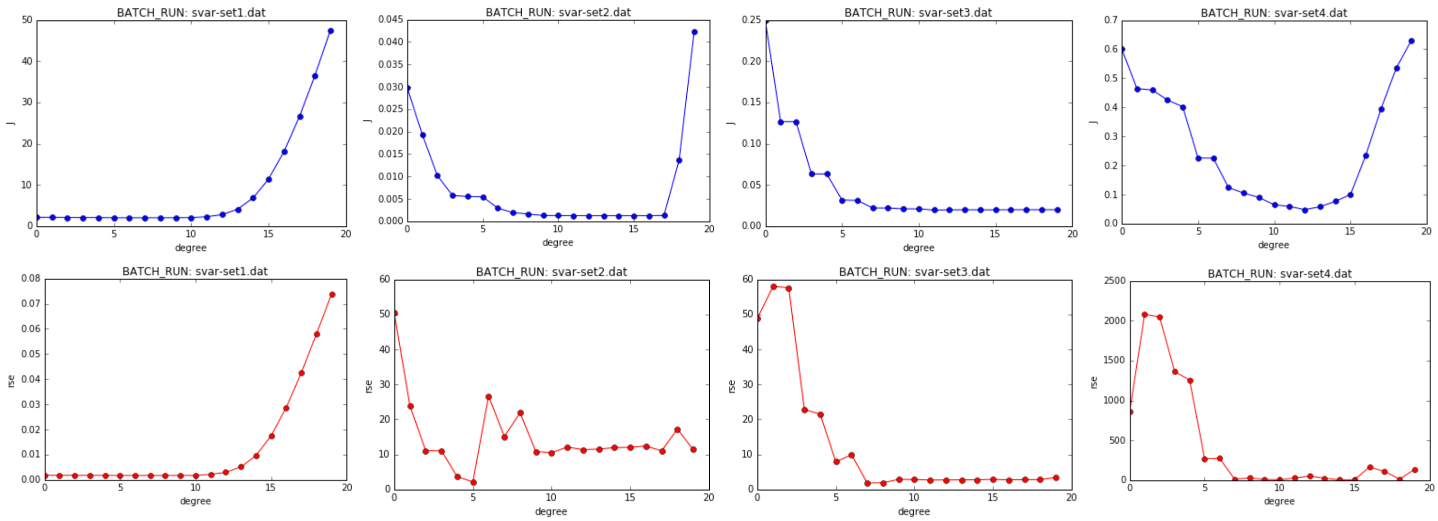
(d) I run the code for all 20 degrees of polynomial model and plot out the changes of *J* and *rse*. The following table shows out the degrees that has minimum *J* and *rse* for all datasets. However, the degree with the minimum value does not imply it's the best choice because of overfitting.

Linear regression result (1 ≤ degree ≤ 20)

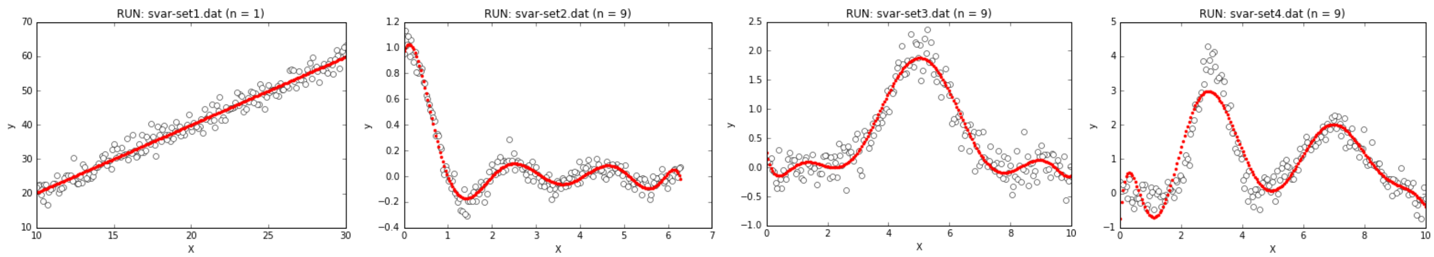
file name	degree with minimum J	minimum J	degree with minimum rse	minimum rse
svar-set1.dat	7	2.0134392	6	0.0017359
svar-set2.dat	14	0.0012537	6	2.1442261
svar-set3.dat	12	0.0193919	8	1.8246026
svar-set4.dat	13	0.0475288	15	4.5023798

I would choose the follow degrees for the four datasets from the following graphs of the change of *J* and *rse* when the *J* dose not change too much:

- svar-set1.dat: degree 1
- svar-set2.dat: degree 9
- svar-set3.dat: degree 9
- svar-set4.dat: degree 9

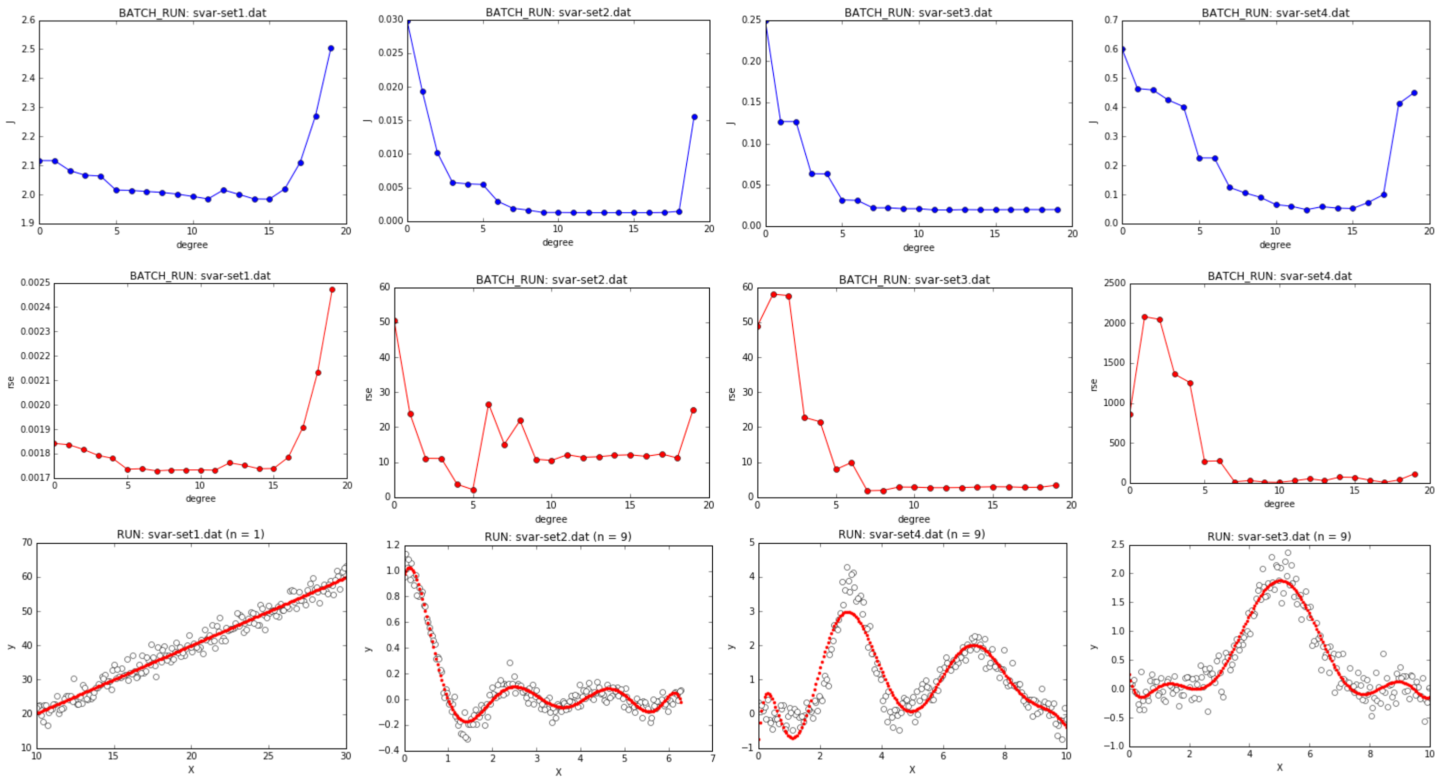


The following graphs are the fit line:



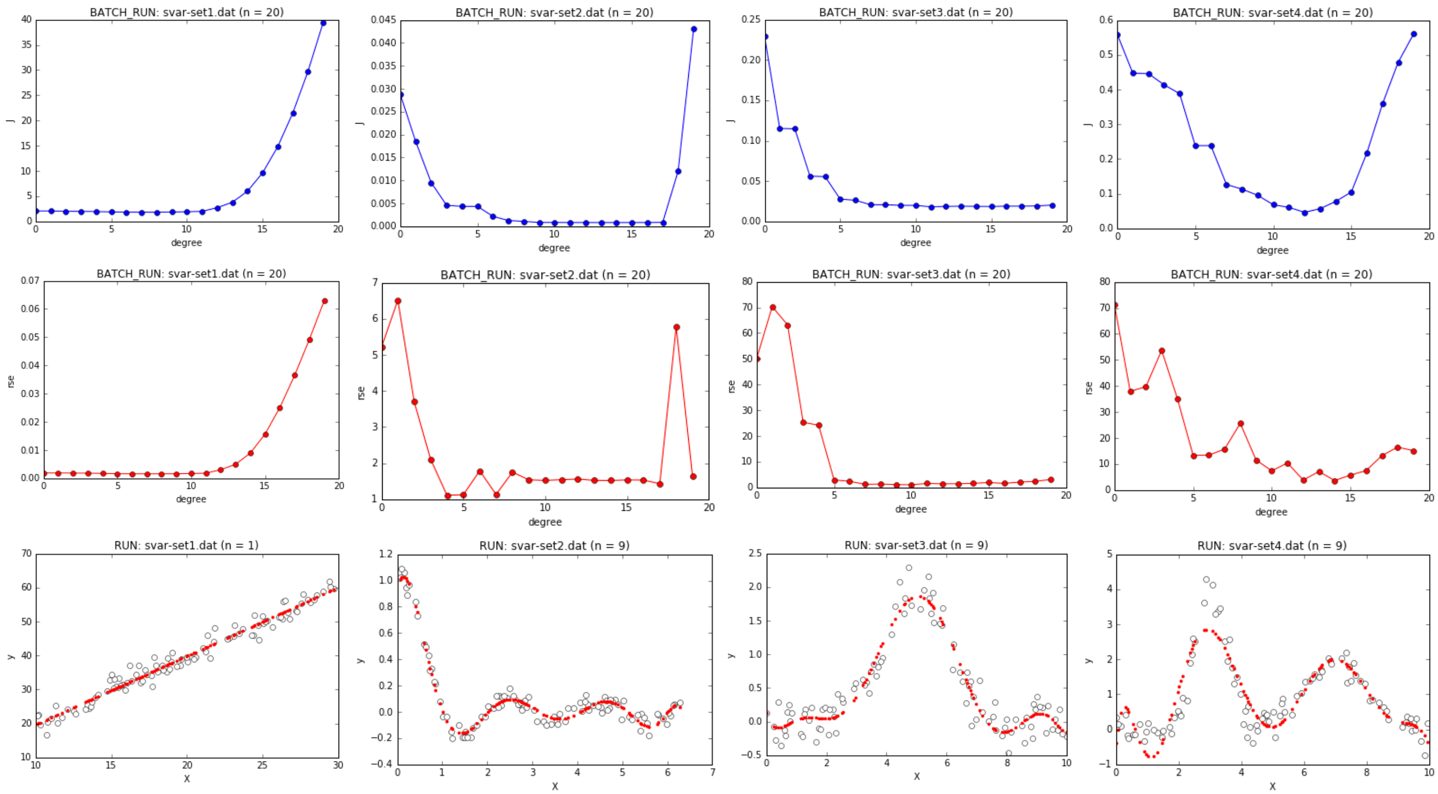
The following table and graphs are the counterparts of sklearn. The result are almost the same. I will choose the same degrees as above for all datasets (1, 9, 9, 9):

	Linear regression result				Sklearn linear regression result			
file name	**degree with minimum J**	**minimum J**	**degree with minimum rse**	**minimum rse**	**degree with minimum J**	**minimum J**	**degree with minimum rse**	**minimum rse**
svar-set1.dat	7	2.0134392	6	0.0017359	16	1.9835461	8	0.0017296
svar-set2.dat	14	0.0012537	6	2.1442261	17	0.0012535	6	2.1442261
svar-set3.dat	12	0.0193919	8	1.8246026	13	0.0193883	8	1.8246027
svar-set4.dat	13	0.0475288	15	4.5023798	13	0.0472079	18	4.9942140



(e) I truncate the data to leave half of the total amount (*truncate*=0.5) and then fit my linear model to the truncated datasets. This is the following result table and graphs. From this table and the graphs compared to the previous non-truncate result, we can see the neither *J* value nor *rse* change too much. I also choose the same degrees as before (1, 9, 9, 9):

	Linear regression result				Linear regression result (truncated data)			
file name	**degree with minimum J**	**minimum J**	**degree with minimum rse**	**minimum rse**	**degree with minimum J**	**minimum J**	**degree with minimum rse**	**minimum rse**
svar-set1.dat	7	2.0134392	6	0.0017359	9	1.8214318	8	0.0015829
svar-set2.dat	14	0.0012537	6	2.1442261	14	0.0007938	5	1.1099932
svar-set3.dat	12	0.0193919	8	1.8246026	12	0.0180816	11	1.0274905
svar-set4.dat	13	0.0475288	15	4.5023798	13	0.0461803	15	3.6592190



Problem 2

(a) I use previously stated method to raise dimension. As the data amount in the third and fourth files are too large (10^5), for some parts I will truncate it to the same order of magnitude of the data (5000) as in the first two files (2500). I will still run the last two files but for some part it took too much time to get the result when raising the dimension to too high. Therefore, I will give a general tendency of comparison results for different orders of magnitude in a relatively low degree (≤ 5).

(b) I apply the linear regression model (explicit solution) on all four datasets with degree from 1 to 5 and record training error, test error, all-data's *J, rse* and running time.

- The training error and test error are both almost the same the all-data error, which means the cross validation does have a verification on the model.
- The running time goes up along with degree, but not changing rapidly.

Linear regression with explicit solution (mvar-set1.dat)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	0.1293361	0.1296423	360.4082295	375.2359958	0.1293514	361.8288696	0.0073189735
degree = 2 (4 features)	0.1291610	0.1298879	331.5389082	355.3820723	0.1291973	333.7110219	0.0088319778
degree = 3 (6 features)	0.1291316	0.1301966	325.3745843	342.8463671	0.1291847	327.0355973	0.0108468532

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 4 (8 features)	0.1287080	0.1297006	361.0206686	359.2957687	0.1287575	360.7932859	0.0141408443
degree = 5 (10 features)	0.1286415	0.1299275	372.7706804	352.7279726	0.1287054	370.5086936	0.0196931362

Linear regression with explicit solution (mvar-set2.dat)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	0.0099544	0.0099844	2068.3546415	2107.1489057	0.0099559	2070.6570848	0.0043740272
degree = 2 (4 features)	0.0099551	0.0099712	2075.8166669	2035.1350778	0.0099559	2069.9818947	0.0055711269
degree = 3 (6 features)	0.0066955	0.0067341	7019.8767267	6946.6595219	0.0066974	7006.8962265	0.0087549686
degree = 4 (8 features)	0.0066928	0.0067447	7181.2342468	7326.8092894	0.0066954	7192.8918859	0.0125007629
degree = 5 (10 features)	0.0058409	0.0058896	9438.2811470	9983.8238323	0.0058433	9481.9599217	0.0155291557

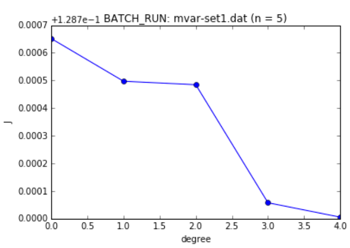
Linear regression with explicit solution (mvar-set3.dat)

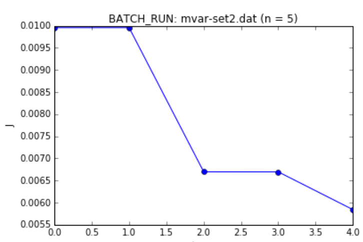
degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	0.1253708	0.1253855	1767.5635246	1781.9418229	0.1253716	1767.6242337	0.2418889999
degree = 2 (4 features)	0.1253664	0.1253963	1604.5079904	1638.3303119	0.1253679	1607.0286321	0.4380970001
degree = 3 (6 features)	0.1253555	0.1254056	1530.7531346	1610.1606114	0.1253580	1536.5441926	0.8650441169
degree = 4 (8 features)	0.1253530	0.1254195	1481.0563946	1506.7719835	0.1253563	1481.2682357	1.3560309410
degree = 5 (10 features)	0.1253478	0.1254146	1414.9467686	1519.5466925	0.1253511	1424.1506508	1.8996341228

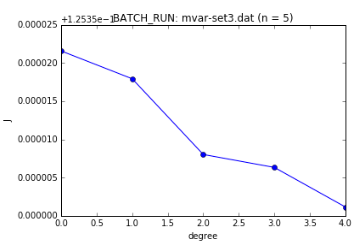
Linear regression with explicit solution (mvar-set4.dat)

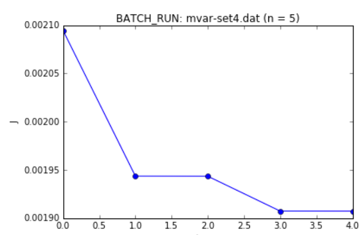
degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	0.0020946	0.0020948	28604.076303	28009.261400	0.0020946	28540.876188	0.2284450531
degree = 2 (4 features)	0.0019436	0.0019440	398692.05629	392473.72566	0.0019436	398054.03751	0.4390969276
degree = 3 (6 features)	0.0019435	0.0019440	402334.68287	392825.44649	0.0019435	401364.98025	0.8619291782
degree = 4 (8 features)	0.0019074	0.0019081	478066.69861	477327.79818	0.0019074	477960.99967	1.3528089523
degree = 5 (10 features)	0.0019073	0.0019083	487511.81500	483601.51367	0.0019073	487089.24529	1.9798588752

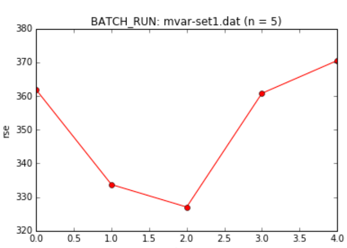
The following graphs are the tendency of the *J* and *rse* value. From the graphs we can tell that the *J's* values become smaller as the degree goes up, which means the linear model fits better. The reason for this may be as we use this particular way to raise the dimension, the linear regression model treated it as a datasets that seems like a polynomial model, therefore fits it better.

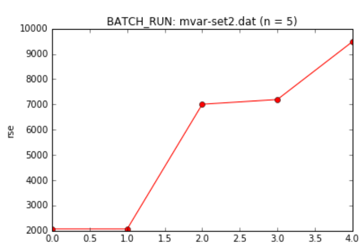


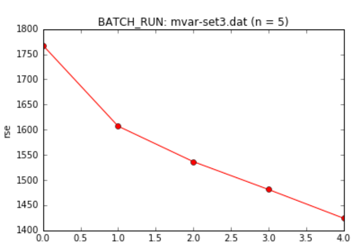


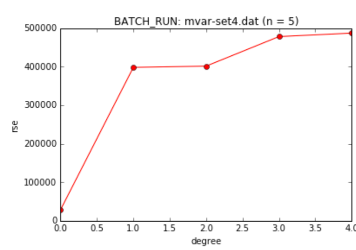












(c) I apply the linear regression model with iterative solution (stochastic gradient descent with *alpha=0.1**5, epsilon=0.1**4* as parameters) on the four datasets with degree from 1 to 5 and record training error, test error, all-data's *J, rse* and running time. The following is the result and graphs.

- The training error and test error are both quite different from the all-data error, which means the cross validation are not able to provide a verification on the model. The reason for this may be that the step alpha and the threshold epsilon are not chosen quite well which leads to it does not always behave with the same process when reaching close to the threshold.

- For the first two datasets the running time does not have a linear relationship with degree, but itself does not changing rapidly; for the last two datasets it grows with the degree. The reason may be that the order of magnitude of the data amount quite differ (2500 vs. 10^5).
- The *J's* values become larger as the degree goes up, which means this linear model with stochastic gradient descent model fits worse. The reason for this may be as we use this particular way to raise the dimension, the linear regression model treated it as a datasets that seems like a polynomial model, therefore fits it better.
- The accuracy is pretty bad compared to the explicit solution, and the running time also runs longer. I may not choose a proper combination of *alpha* and *epsilon* (I run some tests on how different combinations works but didn't get any obvious tendency).
- The issue of how to choose the parameters (*alpha* and *epsilon*). There is one way to choose the *alpha*. We can draw the graph with minimum *J* value (*min_J*) regarding to the number of iteration for a certain *alpha.* If the *min_J* goes down along with the iteration time and does not goes up this means the *alpha* will make the iteration converge. If the *min_J* goes up or up and down repeatedly, that means the *alpha* we choose is too large. Then we should choose a smaller one.

Linear regression with iterative solution (mvar-set1.dat)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	202.8286713	205.9570663	572422.35597	1361024.4978	22.8881674	23956.382620	0.0709998607
degree = 2 (4 features)	394.2638381	392.0790274	1130193.2410	2857234.6983	1029.1894221	2415429.6948	0.1500830650
degree = 3 (6 features)	1909.0281614	1935.2356749	6549678.3843	16272213.840	4277.3637483	10419433.381	0.0796971321
degree = 4 (8 features)	4599.0943461	4558.4468724	23897547.236	812810.51183	1011.1714094	1190091.4429	0.1028251647
degree = 5 (10 features)	16304.845161	15556.409105	53898731.699	55167541.774	6932.1246411	110145271.5 647999	0.0966839790

Linear regression with iterative solution (mvar-set2.dat)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	112.8245631	104.6048386	190213071.8 920090	53728940.875	44.6899099	32183081.589	0.0649518966
degree = 2 (4 features)	545.4377277	571.3959987	778321032.4 986173	347517955.3 663061	667.8971428	2548598151. 4122791	0.1079969406

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 3 (6 features)	1347.9727378	1332.0776649	3362454547.2077131	614305770.1418971	3447.8084775	12737568424.9286537	0.0715069770
degree = 4 (8 features)	4589.5133338	4749.4796285	13612011359.8991680	13216051800.1316681	1397.3187972	944049037.9111441	0.0729851722
degree = 5 (10 features)	11258.952495	11242.624633	51336933619.5838165	35182858612.5818863	559.8605834	1042836585.7586036	0.0733079910

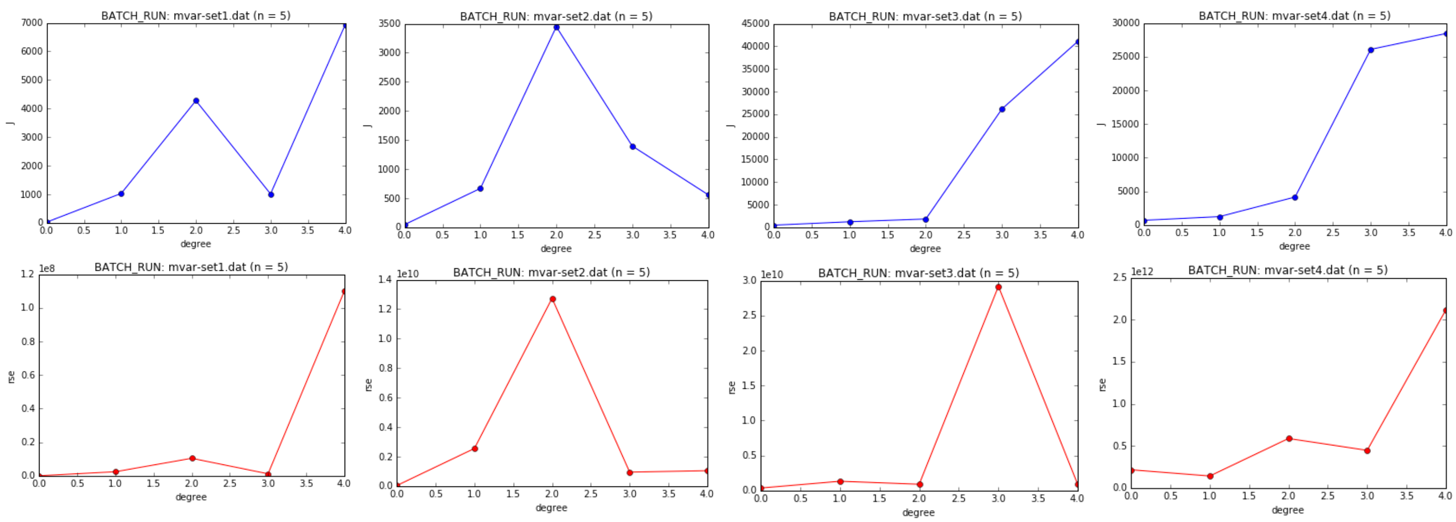
Linear regression with iterative solution (mvar-set3.dat)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	477.6168061	474.3067988	401866174.7810403	50304964.303	455.5522243	343735547.1871337	2.5786299705
degree = 2 (4 features)	1727.7887226	1716.0605707	2554611509.8350320	20920960.303	1212.1543571	1317761104.9344943	2.7108860015
degree = 3 (6 features)	6094.4414379	6122.7608316	11304507464.8149738	3427882605.2362738	1825.6090319	889457247.0449075	2.9876351356
degree = 4 (8 features)	12746.573494	12747.341843	13378178784.0820236	7132894137.7534008	26157.827506	29166746806.7113800	3.3172180652
degree = 5 (10 features)	121464.20605	120367.80225	39888751064.2796783	87032726478.5726166	41150.191453	891979494.1386679	3.6453669071

Linear regression with iterative solution (mvar-set4.dat)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	456.5979647	459.3192153	116983299049.8673859	510657199752.0978394	700.2639543	215523990002.5590515	2.5794279575
degree = 2 (4 features)	1346.5216402	1352.0719613	181292141856.3600159	253460255818.0941467	1244.1309442	141254247120.8588867	2.6913459301
degree = 3 (6 features)	6551.0266322	6506.1879179	392705898896.1884155	425172203514.5186768	4156.7163249	587256552511.1613770	3.0147910118
degree = 4 (8 features)	17163.406981	17203.377887	789936178436.2829590	491471648667.7125854	26089.734484	446410391002.9657593	3.3639090061

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 5 (10 features)	104767.29361	104379.97438	3313839875 248.8061523	6266770166 990.4062500	28456.284660	2116377785 399.1228027	3.6426429748



(d) I use a Gaussian kernel function with *sigma*=0.1 as parameter for the first two datasets and *sigma*=1.0 for the last two datasets (For less error). All running have 1 to 4 degrees of dimension expansion. I compared the result with different *sigma*. Details can be found in the code file, under Problem 2 (d) section. For the last two datasets I truncated the data to 5000.

- The training error and test error are both not far from the all-data error, which means the cross validation does have a verification on the model.
- The running time goes up along with degree generally, except for the first two datasets which time drops when raising to degree 4. For the last two datasets there is a large increment when raising to degree 4. I consider the reason for this may be that when raising to degree 4, as the amount of data in the last two datasets is very large, the calculation of the matrix grows exponentially.
- The errors for training, test and all-data are all much larger than those got from the primal regression problem, and the time consuming becomes very large as well. I only use 5% part of the entire datasets but get a much worse runtime than using the entire datasets in the primal regression problem.

Dual linear regression with Gaussian kernel function (mvar-set1.dat sigma=0.1)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	1003809.892(	1003352.199(	78182131.216	78618858.026	1020617.959(	79757336.569	5.3065590858
degree = 2 (4 features)	55727828.41(	55575249.733	2004927381 99.5550842	1991689151 27.5007324	66036217.519	2376691027 94.0010376	11.516115903
degree = 3 (6 features)	671728961.4 658854	668186689.2 618682	1164152224 3.2386436	1108039664 2.6692810	809215530.1 192247	1364634904 9.7663212	48.226015090
degree = 4 (8 features)	3439645853. 1653852	3427369895. 7334776	1313798752 0658.041015 6	1390382525 1502.371093 8	4149249796. 6732640	1594722883 0254.246093 8	18.187160968

Dual linear regression with Gaussian kernel function (mvar-set2.dat sigma=0.1)

degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	1575.9020651	1575.3227983	1160116443. 6679690	1154745786. 8762374	1593.0793671	1174526269. 7126796	5.9769029617
degree = 2 (4 features)	11418.061943	11385.293283	8507929637. 2965603	8534255136. 1887951	12911.921194	9626560015. 2140369	11.603729963
degree = 3 (6 features)	330353.94945	330582.10014	1604638769 30.7758484	1590682438 07.3128967	384820.96968	1869404337 07.5044861	47.725635051
degree = 4 (8 features)	345382.63299	343163.59244	1846365262 30.0689392	1572890203 23.6152954	403067.41721	2117746210 63.7340393	18.141860961

Dual linear regression with Gaussian kernel function (mvar-set3.dat sigma=1.0)

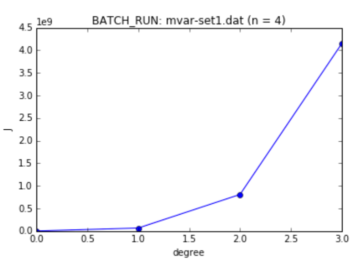
degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	5774927.1317	5769510.8175	1253557276. 3427930	1181508937. 1457267	6236736.2438	1415652947. 9357581	28.799829006
degree = 2 (4 features)	1549176982. 2342811	1545924934. 8674951	9809088724 84.8951416	1000491253 844.8474121	1818832587. 2162952	1151708796 127.1916504	29.778747081
degree = 3 (6 features)	4728706815 8.0624466	4713424202 1.0766220	2205599902 6538.148437 5	2150209674 9512.859375 0	5789488751 8.9523392	2695164641 1787.765625 0	30.893382072

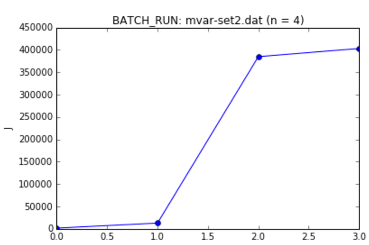
degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 4 (8 features)	3535645010 92.0264282	3514402848 14.3825684	9240857327 9868.062500 0	1060860938 82700.46875 00	4349063568 42.6094971	1151057783 39368.54687 50	193.28428101

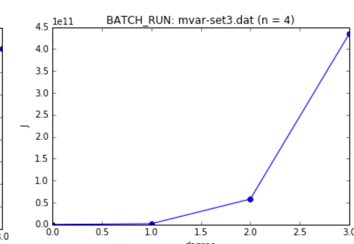
Dual linear regression with Gaussian kernel function (mvar-set4.dat sigma=1.0)

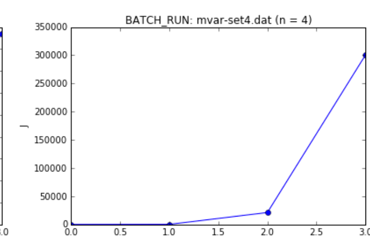
degree	cross validation training average J	cross validation test average J	cross validation training average rse	cross validation test average rse	all-data J	all-data rse	running time (second)
degree = 1 (2 features)	81.5188111	80.7848015	293788657.5 852719	148950563.2 192642	113.4556707	352057434.0 254115	31.248461961
degree = 2 (4 features)	325.7297527	328.5601603	1524938148. 7975826	2956624221. 7996564	200.4387224	1077291404. 0104530	32.813752889
degree = 3 (6 features)	19441.178862	19174.744531	1392982290 49.9418640	2322460732 84.8623047	21244.095456	1663628533 88.1490784	34.650149822
degree = 4 (8 features)	282838.92826	281864.55869	1461668184 821.7299805	1753759332 182.8117676	300739.04857	1649569682 669.2973633	199.02502417

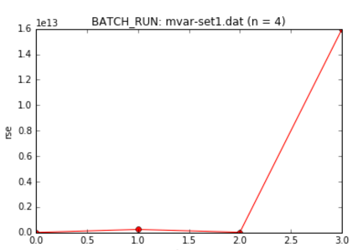
The following graphs describes how are the *J* and *rse* value tendency. As the degree goes up, the *J* and *rse* value both goes up as well. However, I did some additional tests on choosing *sigma*. When I choose to use a larger *sigma* (*sigma*=0.3/0.5 for the first two datasets, *sigma*=10.0 for the last two datasets), I found the *J* and *rse* value go down as the degree goes up, which means when applying the Gaussian kernel function to a multivariate problem it fits better when we raise the dimension to a higher degree. But the error becomes worse. This is the trade-off we should deal with. Details can be found in the code file, under Problem 2 (d) section.

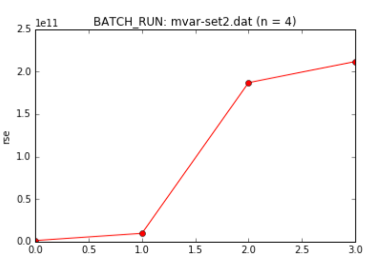


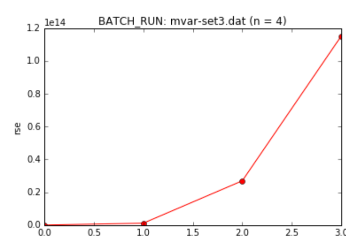


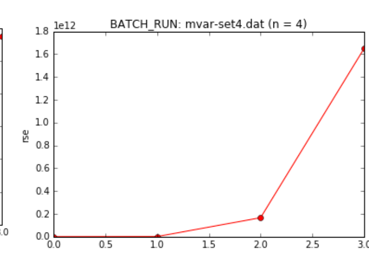












5. References

[1] https://www.coursera.org/learn/machine-learning
[2] http://www.numpy.org/
[3] https://www.python.org/
[4] http://scikit-learn.org/stable/