HW 1

Code

https://github.com/aokblast/NTNU_Programming_Homework/tree/master/Predictive_Learning/hw1

Problem 2.7

Explain

The algorithm is as following:

1. choose on `p` to be a validator
2. collect all other data into PointCloud
3. Predict the answer of `p` and check if it is correct
4. Repeat `1~3` until all points has been choose as `p` once
5. Calculate $correct = \frac{\text{correct number}}{total}$ in step `3` and $\text{validation error} = 1 - correct$
6. Use previous all points as a PointCloud and predict with all data from other year like step `3.`
7. Calculate $correct = \frac{\text{correct number}}{total}$ in step `6` and $\text{test error} = 1 - correct$

Data

k	validation_error	test_error
1	0.46153846153846156	0.4901960784313726
2	0.46153846153846156	0.4509803921568627
3	0.3653846153846154	0.5098039215686274
4	0.32692307692307687	0.5686274509803921

5	0.32692307692307687	0.5490196078431373
6	0.34615384615384615	0.5294117647058824
7	0.2692307692307693	0.47058823529411764
8	0.32692307692307687	0.5294117647058824
9	0.3076923076923077	0.4509803921568627
10	0.3076923076923077	0.5490196078431373
11	0.3076923076923077	0.47058823529411764
12	0.40384615384615385	0.5490196078431373
13	0.3846153846153846	0.5686274509803921
14	0.34615384615384615	0.5686274509803921
15	0.34615384615384615	0.5098039215686274
16	0.28846153846153844	0.5294117647058824
17	0.28846153846153844	0.5294117647058824
18	0.2692307692307693	0.5098039215686274
19	0.3076923076923077	0.5294117647058824
20	0.32692307692307687	0.5294117647058824
21	0.34615384615384615	0.5098039215686274
22	0.32692307692307687	0.5490196078431373
23	0.3653846153846154	0.5294117647058824
24	0.3653846153846154	0.5294117647058824
25	0.42307692307692313	0.5098039215686274
26	0.40384615384615385	0.5098039215686274
27	0.42307692307692313	0.4509803921568627
28	0.42307692307692313	0.4509803921568627
29	0.42307692307692313	0.4509803921568627

30	0.42307692307692313	0.4509803921568627
31	0.42307692307692313	0.4509803921568627
32	0.42307692307692313	0.4509803921568627
33	0.5192307692307692	0.3921568627450981
34	0.5576923076923077	0.3921568627450981
35	0.4423076923076923	0.4117647058823529
36	0.46153846153846156	0.4901960784313726
37	0.3653846153846154	0.4117647058823529
38	0.3653846153846154	0.4117647058823529
39	0.3653846153846154	0.4117647058823529
40	0.3653846153846154	0.4117647058823529
41	0.3653846153846154	0.4117647058823529
42	0.3653846153846154	0.4117647058823529
43	0.3653846153846154	0.4117647058823529
44	0.3653846153846154	0.4117647058823529
45	0.3653846153846154	0.4117647058823529
46	0.3653846153846154	0.4117647058823529
47	0.3653846153846154	0.4117647058823529
48	0.3653846153846154	0.4117647058823529
49	0.3653846153846154	0.4117647058823529
50	0.3653846153846154	0.4117647058823529
51	0.3653846153846154	0.4117647058823529

Result

The best performace (e.g. min(validate_error)) in my experiment is when k == 18:

The validation error is: 0.2692307692307693

The test error is: 0.5098039215686274

Discuss

Thought the best model when considering validation error is when k == 18, however, the best model when considering test error is when k = 0.39. But most of the test error rate when k <= 26 is about 0.5 which is like the algorithm is guessing the result. That means the quality of my model is not quite good as it needs to use half of the data to get a better performance in these dataset.

When I was doing my homework and discuss with classmates, I discover that I have different correctness with them when the k is equal. The difference between our code is that I hand-written most part of the algorithm in rust and they use 3rd package from python. When I check the result from knn I think my result is correct. I don't know what the package do with the knn algorithm.

Problem 2.8

Explain:

See 2.7

Data

k	validation_error	test_error
1	0.5294117647058824	0.3653846153846154
2	0.5490196078431373	0.3653846153846154
3	0.4117647058823529	0.3653846153846154
4	0.47058823529411764	0.3653846153846154
5	0.4901960784313726	0.3846153846153846
6	0.4509803921568627	0.42307692307692313
7	0.43137254901960786	0.3846153846153846

8	0.4901960784313726	0.3846153846153846
9	0.43137254901960786	0.3846153846153846
10	0.3529411764705882	0.34615384615384615
11	0.4509803921568627	0.32692307692307687
12	0.37254901960784315	0.3653846153846154
13	0.43137254901960786	0.3653846153846154
14	0.3529411764705882	0.3653846153846154
15	0.3921568627450981	0.3653846153846154
16	0.3921568627450981	0.3653846153846154
17	0.4509803921568627	0.3653846153846154
18	0.43137254901960786	0.34615384615384615
19	0.3921568627450981	0.34615384615384615
20	0.4117647058823529	0.34615384615384615
21	0.4509803921568627	0.3653846153846154
22	0.3921568627450981	0.3653846153846154
23	0.3921568627450981	0.3653846153846154
24	0.37254901960784315	0.34615384615384615
25	0.37254901960784315	0.34615384615384615
26	0.37254901960784315	0.34615384615384615
27	0.3921568627450981	0.34615384615384615
28	0.37254901960784315	0.34615384615384615
29	0.37254901960784315	0.34615384615384615
30	0.37254901960784315	0.34615384615384615
31	0.43137254901960786	0.34615384615384615
32	0.4117647058823529	0.34615384615384615

33	0.43137254901960786	0.3653846153846154
34	0.43137254901960786	0.3653846153846154
35	0.4901960784313726	0.3846153846153846
36	0.4509803921568627	0.3846153846153846
37	0.5686274509803921	0.3846153846153846
38	0.5098039215686274	0.3846153846153846
39	0.5490196078431373	0.3846153846153846
40	0.607843137254902	0.34615384615384615
41	0.4117647058823529	0.3653846153846154
42	0.4117647058823529	0.3653846153846154
43	0.4117647058823529	0.3653846153846154
44	0.4117647058823529	0.3653846153846154
45	0.4117647058823529	0.3653846153846154
46	0.4117647058823529	0.3653846153846154
47	0.4117647058823529	0.3653846153846154
48	0.4117647058823529	0.3653846153846154
49	0.4117647058823529	0.3653846153846154
50	0.4117647058823529	0.3653846153846154

Result:

The best performace (e.g. min(validate_error)) in my experiment is when k == 10, 14:

The validation error is: 0.3529411764705882

The test error is: 0.34615384615384615, 0.3653846153846154

Discussion:

Compare with 2.7, though the validation error is higher in 2.8, the test error is lower and the performace is much better than 2.7. Most of the k can get 0.3 test error. Maybe it means this model is overfit or the 2.7 model is underfit?

Problem 2.11

Sample:

X	Y
0.9774171505952622	1.1588571517773827
0.4380523297069485	-0.024600862227224185
0.58286698023262	0.677278015593132
0.9090957156485439	0.8965742500346358
0.8485955450006925	1.1681234216821834
0.9735090066919307	0.9225895473090115
0.5842804504562145	0.2283065546608565
0.29553268411834754	0.49888744235465865
0.6999167117596083	0.43153557984825675
0.2855968151774022	0.2745488013963385

Explain

The estimator is as following:

- Trigonometric:

  $f_m(x, w) = \Sigma_{i=1}^{m} w_i \cos(2\pi i x) + w_0$
- Polynomial:

  $f_m(x, w) = \Sigma_{i=1}^{m} w_i x_i + w_0$

To predict the best parameter (w) with fixed m I use gradient descent algorithm, the loss function I used is MSE:

$min(\Sigma_{i=1}^{n}(y_i - f_m(x_i, w))^2)$ where n is the point we have

The gradient is:

- Trigonometric:

  $$\text{Gradient}_i = \frac{d\Sigma_{j=1}^{n}(y_j - f_m(x_j, w))^2}{dw_i} = \Sigma 2(y - f_m(x, w)(-\cos(2\pi i x)))$$

- Polynomial

  $$\text{Gradient}_i = \frac{d\Sigma_{j=1}^{n}(y_j - f_m(x_j, w))^2}{dw_i} = \Sigma 2(y - f_m(x, w)(x^i))$$

With gradient and loss function:

1. Use gradient descent algorithm to find best w when m fixed.
2. Calculate $\Sigma_{i=1}^{n}(y_i - f_m(x_i, w))^2$ with predicted w to get current MSE.
3. Schwartz criterion is $r(p, n) = 1 + p(1 - p)^{-1} \ln n$ where $p = \frac{m}{n}$ that $m = \text{degree of function}, n = \text{number of sample}$
4. $R = \text{MSE} * r(p, n)$
5. Repeat the previous steps with $m = 1 \rightarrow n$

Data

- Trigonometric

m	MSE	Schwartz	R
1	1.5004954426427666	1.2558427881104497	1.8843863802355154
2	0.4352945089531259	1.5756462732485115	0.6858701707975337
3	0.4308606253587933	1.986822182711734	0.8560434481199005
4	0.38759064501267054	2.5350567286626973	0.9825642726060854
5	0.3245248841069851	3.302585092994046	1.0717710445573494
6	0.16085293523146885	4.453877639491068	0.7164192914739442
7	0.14851668677512325	6.3726985503194395	0.9464520745100742
8	0.11680741785176631	10.210340371976185	1.1926434942381814
9	0.11827834600032126	21.723265836946418	2.569391952919307

- Polynomial

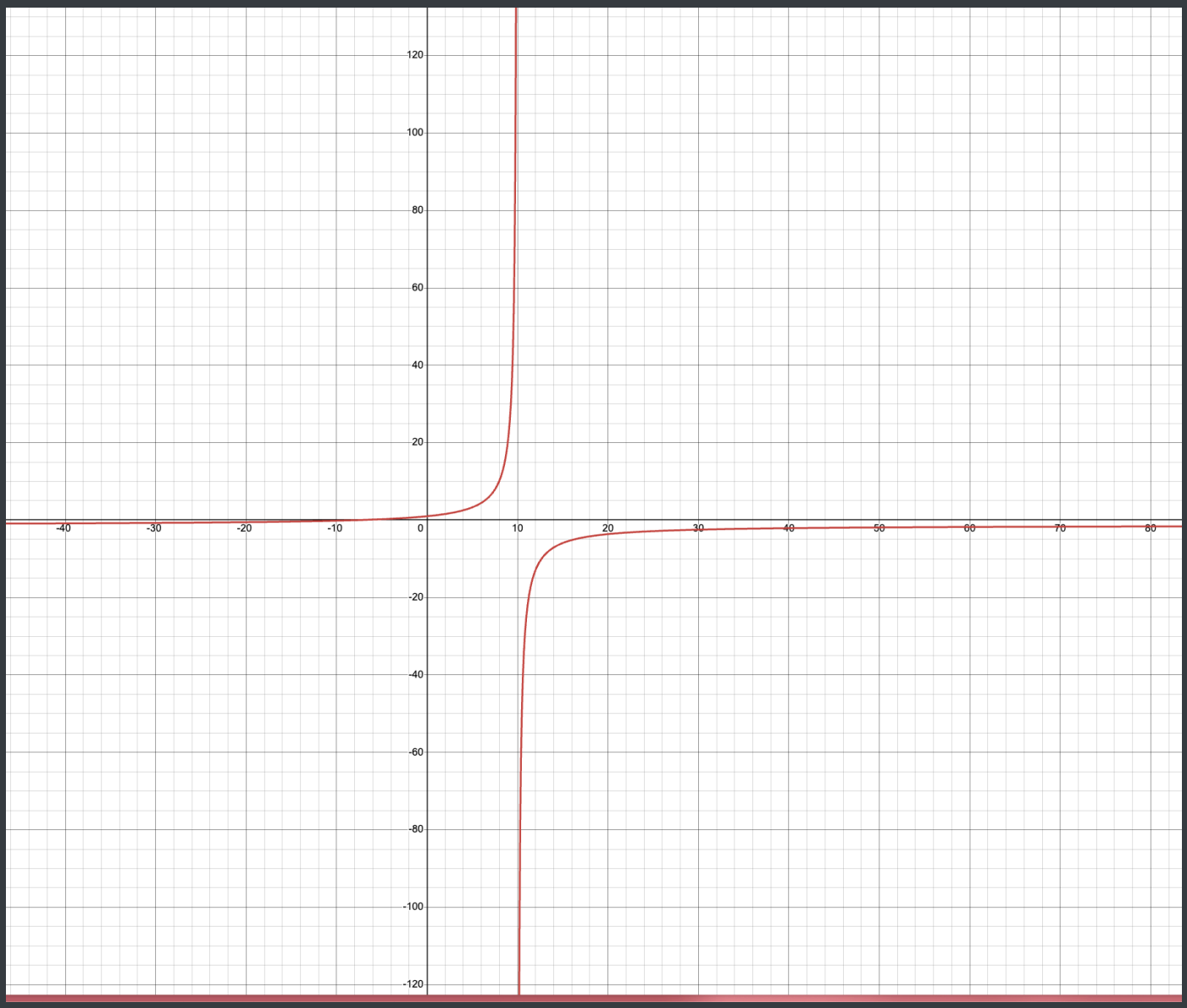
m	Loss	Schwartz	R
1	1.5004954426427666	1.2558427881104497	1.8843863802355154
2	0.556678101174504	1.5756462732485115	0.877127775514665
3	0.44343219131889194	1.986822182711734	0.8810209142408482
4	0.448981913450957	2.5350567286626973	1.1381946207417013
5	0.42811087262843034	3.302585092994046	1.4138725860913268
6	0.4396019922717735	4.453877639491068	1.9579334836549773
7	0.40374408648143717	6.3726985503194395	2.572939354620301
8	0.3736641856230035	10.210340371976185	3.815238520028156
9	0.36120705914733103	21.723265836946418	7.8465969680391

Result

The best model according to R is when m = 2.

Discuss

The graph of Schwartz criterion is as follows (n = 10):



It shows when m appears n. The r will takes more responsibility on the score of R. It is just like a type of panelty to balance the result from MSE. With panelty, we can use the result of Schwartz criterion to get the best model in some case. Also, the result from Trigonometric estimator and Polynomial estimator have the very close RME, which means these two models have a quite close performance.

Problem 2.12

Explain

Same as 2.11 but the data will be 8 only because we use 5 folds ($\text{validation number} = \frac{10}{5} = 2$). And in step `3.` , we don't use the training data to calculate the MSE, instead, we use the extracted 2 data to calculate MSE. We choose the best model with minimium R in Schwartz criterion on each round of fold-5. The validation data is in sample. We choose the first 2 data in each iteration and step the iteration with 2 (e.g. 0 (0, 1 as validation), 2 (2, 3 as validation),4, 6, 8).

Data

- Trigonometric:

k	mse	rme
2	0.14662934075234452	0.24826505488219558
1	0.10780471715234635	0.1398295181860083
2	0.11174197787331741	0.18919561478639915
2	0.0009101583428534974	0.0015410320320655112
2	0.10039803538275358	0.16998865054206683

- Linear:

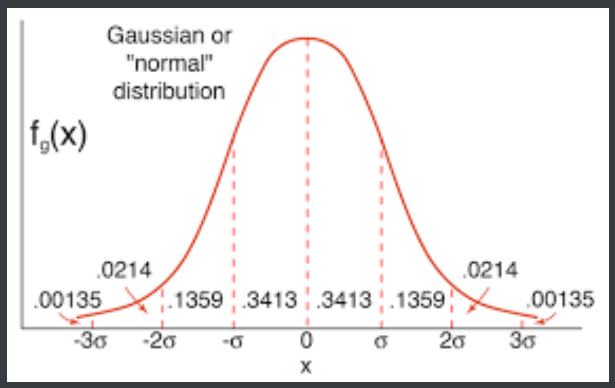
k	mse	rme
3	0.1778765211500136	0.39980681757131176
2	0.0274755991870545	0.046520233297756446
2	0.13581491670447166	0.22995464329615997
1	0.247228588010343	0.3206710731817082
3	0.03009431150717867	0.06764192841694718

Discussion

The lowest RME on Trigonometric is when data (6,7) are validation data. For linear case, the best model is when data (2, 3) is validataion data.

The results shows that the performance of Trigonometric is better than the Linear case. The reason maybe the noise is Gaussian distribution that has a peak. The cosine function from [0, 1] -> [0, π] is also a function with peak. That means the train result is more comply with the RME.

- Gaussian Distribution:



- Cosine function: