Marc Fuentes i Oncins
Víctor Novelle Moriano

REPORT

We will execute our different configurations for the pattern recognition task. We will check the 0-9 values with 3 different algorithms (isd=1,3,7) and parameters (Lambda=0,1,10). The analysis presented in this report comes from the results obtained shown in the table found on the Annex section.

Input parameters:
tr_seed = 801677
te_seed = 945386
Executed on *Linux* version *Ubuntu 18.04.4 LTS*

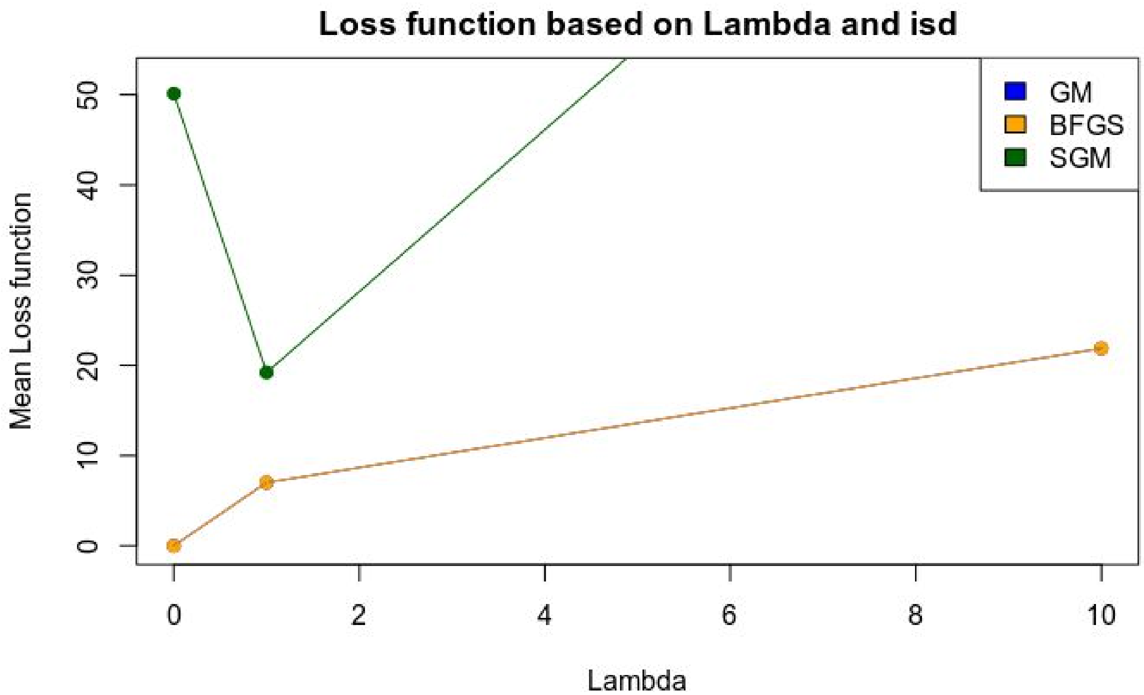
In order to execute the code, you should run *execute.m* with the mentioned seeds.

We indicate the version of the *OS* because, although executing exactly the same code, the results obtained in our respective compute are different. That might be due to the fact that the computers OS are *Ubuntu* and *Mac OS respectively,* and that the random function used in the batch generation, although the same seed has been used, behaves differently.

CONVERGENCE

Global convergence:

First, we are going to study the global convergence of the three algorithms only in terms of the objective function L.



Lambda	isd	Mean of L.
0	1	1.213114e-02
0	3	2.520188e-08
0	7	50.13337
1	1	7.023
1	3	7.023
1	7	19.21
10	1	21.89
10	3	21.89
10	7	Inf

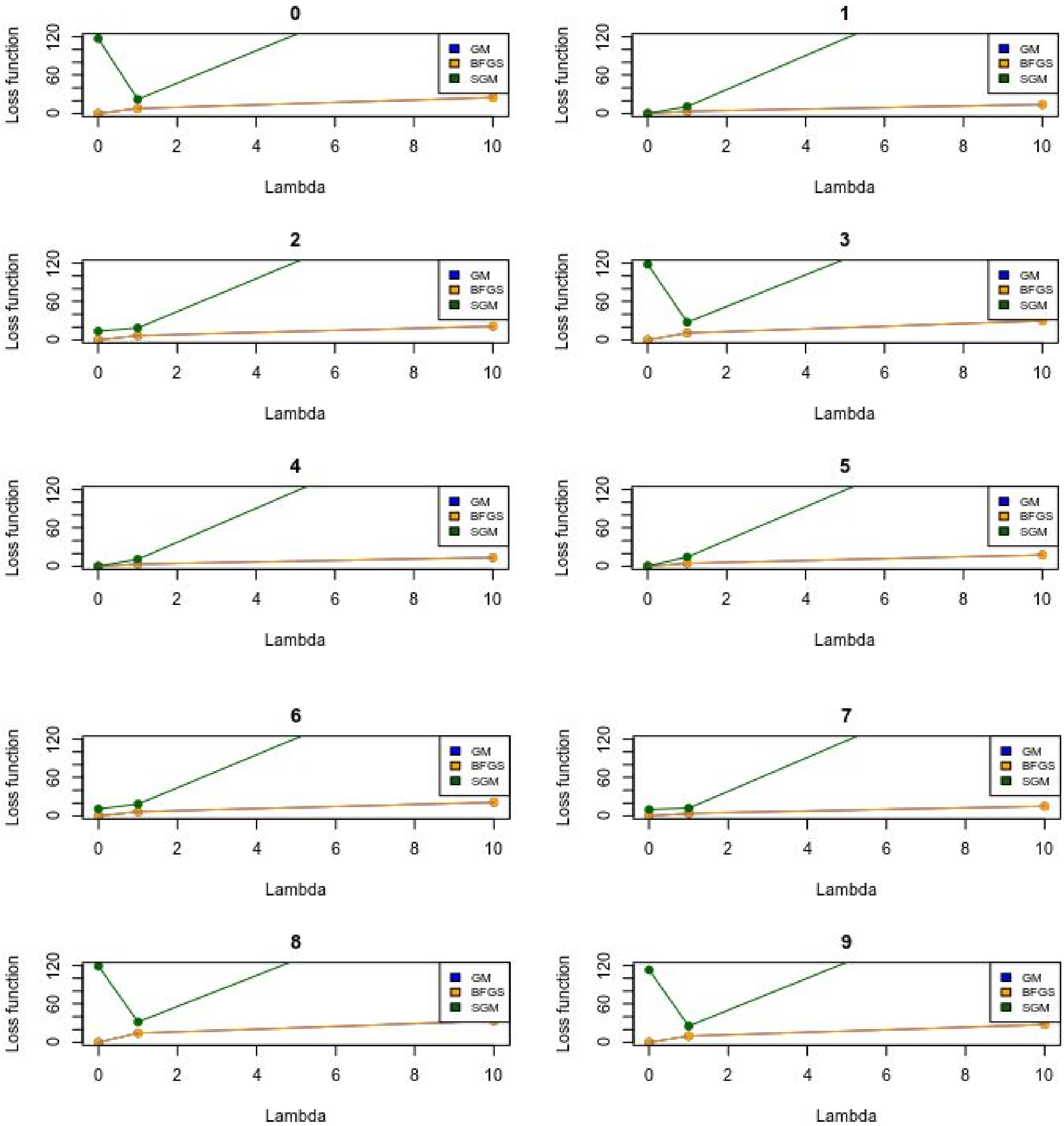
Looking at the data we can see that the executions with *isd*=1 and *isd*=3 do always converge and the ones with *isd*=7 sometimes do not. The cases were it doesn't converge are specifically the ones with *lambda*=10.

This is due to the fact that, according to the Zoutendijk's theorem for global convergence, *GM* and *BFGS* satisfy the necessary conditions. In all the cases, the function is continuously differentiable and bounded below. For the *GM*, the selected direction is always descendent and the learning rate satisfies the WC. Moreover, the *CAC* is obviously accomplished, as its value is always 1. Regarding BFGS, the approximation of the true hessian is positive definite, assuring that the selected direction is descending, and its condition number is uniformly bounded, satisfying the *CAC.*

However, the *SGM*, as we have seen above, does not converge, due to the fact that the step length found may not satisfy the *WC*. The computation of this parameter is realized using a pre-established formula, instead of applying *BLS* like the other methods. As a result, despite having a descent direction, the step length does not satisfy the *WC*, specifically *WC1*, and we cannot guarantee its convergence.

Indicate that the *Loss* function is convex and, as result, all the stationary points found are global minimizers of the function. We also must emphasize that the optimal points we find may not be exactly the same, as we finish the execution when the value of the gradient norm is inferior to a certain *epsilon.*

We have also analyzed the value of the loss function for the 9 combinations of parameters/methods for each one of the target values, observing the following results.

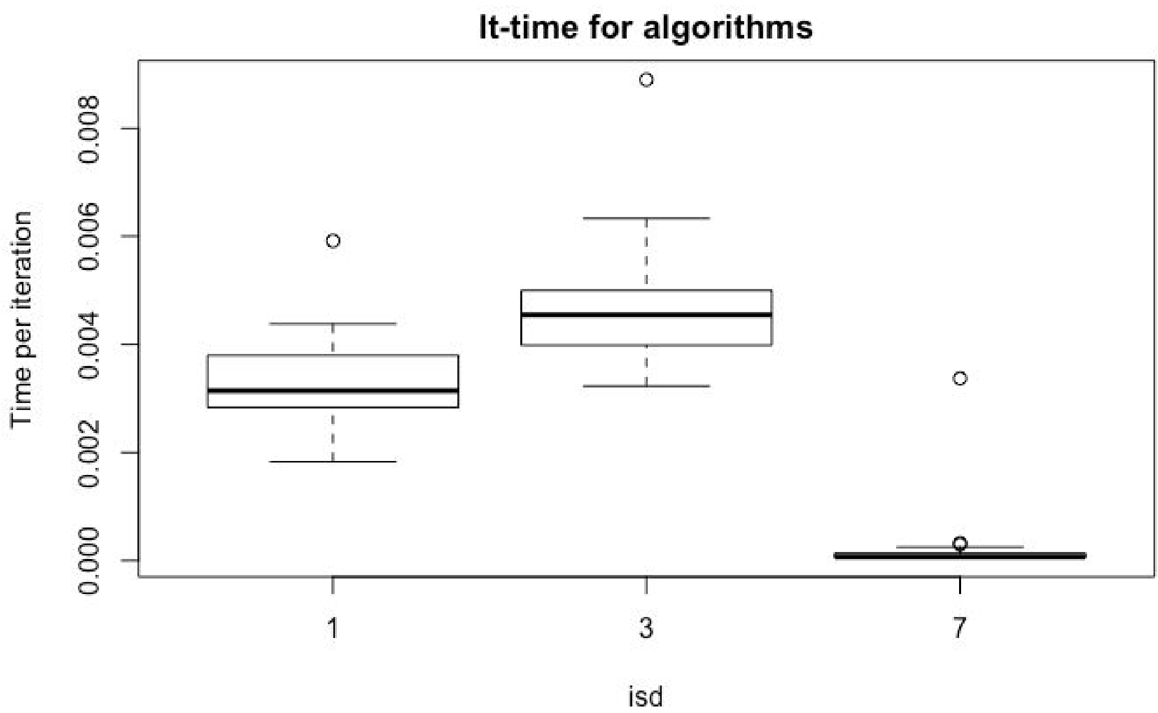


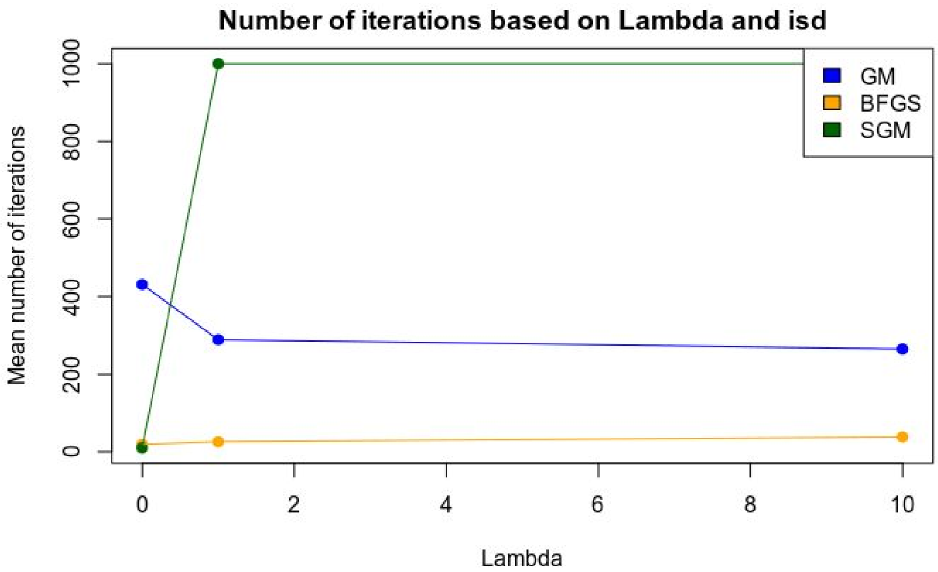
Local convergence:

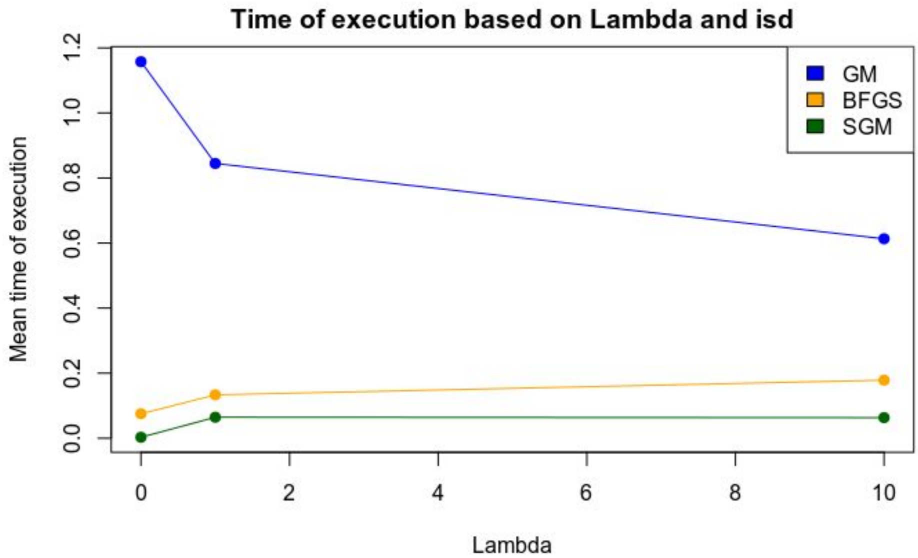
To analyze the local convergence of the algorithms we will use the number of iterations *niter*, and the execution time *tex*. To compare the configurations we will use the time per iteration, which is equal to *tex/niter*.

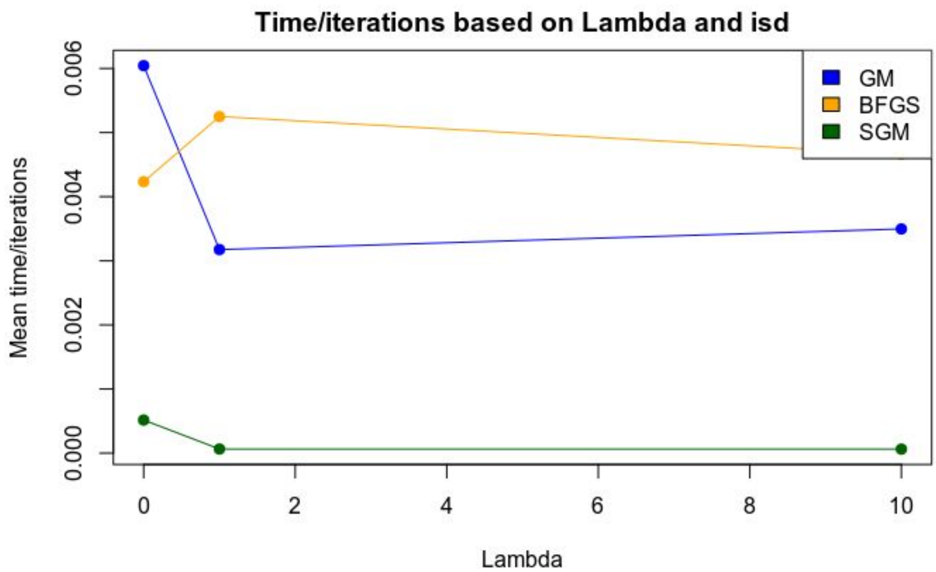
Lambda	isd	nº iterations	time of execution	time/iterations
0	1	431.1	1.15757	0.0060430888
0	3	19.1	0.07513	0.0042328502
0	7	10.1	0.00284	0.0005159492
1	1	288.9	0.84465	0.0031747911
1	3	25.9	0.13329	0.0052496031
1	7	1000.0	0.06434	0.0000643400
10	1	264.8	0.61305	0.0034966555
10	3	38.2	0.17839	0.0046617139
10	7	1000.0	0.06306	0.0000630600

Focusing now on the algorithm used we can plot the results:




Number of iterations based on Lambda and isd
Mean number of iterations
Lambda
GM
BFGS
SGM
0
200
400
600
800
1000
0
2
4
6
8
10


Time of execution based on Lambda and isd
Mean time of execution
Lambda
GM
BFGS
SGM
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
2
4
6
8
10


Time/iterations based on Lambda and isd
Mean time/iterations
Lambda
GM
BFGS
SGM
0.000
0.002
0.004
0.006
0
2
4
6
8
10

If we analyze the 3 graphs provided above we can state that the worst algorithm in terms of iterations is *SGM*. This is due that, as we have seen previously it does not converge for all cases. Between *BFG*S and *GM*, we can observe that the *gradient method* performs a notably larger number of iterations, due to the fact that it has linear order of convergence against the superlinear convergence of *BFGS* (when the conditions are satisfied).

The slowest algorithm in terms of time per iterations is *isd*=3, followed by *isd*=1 and the fastest being *isd*=7. The stochastic gradient method is the fastest due to it only uses a random fraction of our data. Regarding time per iteration, it is logical to think that the *gradient method* is going to be faster than the *BFGS*. As shown in the data, the iterations based on the *gradient method* take about 30% less time than the ones with *BFGS*. As *BFGS* takes the direction from the product of the approximation of the true hessian and the gradient, it adds extra computations compared to the *gradient method* and therefore it becomes slower.

Conclusion:

After observing the impact of the algorithms and lambda values in terms of the loss and timer per iteration we can see that the best algorithms are:

1. If we are looking to minimize the loss function independently of the used time, *BFGS* and *GM* both perform well. Compared to SGM, their respective times of execution are higher but they have a smaller loss value than it.

2. The SGM produces the highest loss but it is the fastest one by a big difference. Despite not achieving the global minimum,we will see later on that it has a high percentage of accuracy (especially with *Landa = 1),* making it the best method if we want a quite high success rate and a prioritization of time.

ACCURACY

We will now take a look at the performance of the algorithms in terms of the accuracy of the result to check how well does the prediction fit the data.

First of all the analysis will focus on the *lambda-isd* combinations.
Mean value of accuracy for all lambda-isd combinations:

Lambda	isd	tr_acc	te_acc
0	1	100.00	98.80
0	3	100.00	97.28
0	7	79.84	60.16
1	1	99.80	99.44
1	3	99.80	99.44
1	7	97.48	95.44
10	1	99.16	98.36
10	3	99.16	98.36
10	7	65.80	32.96

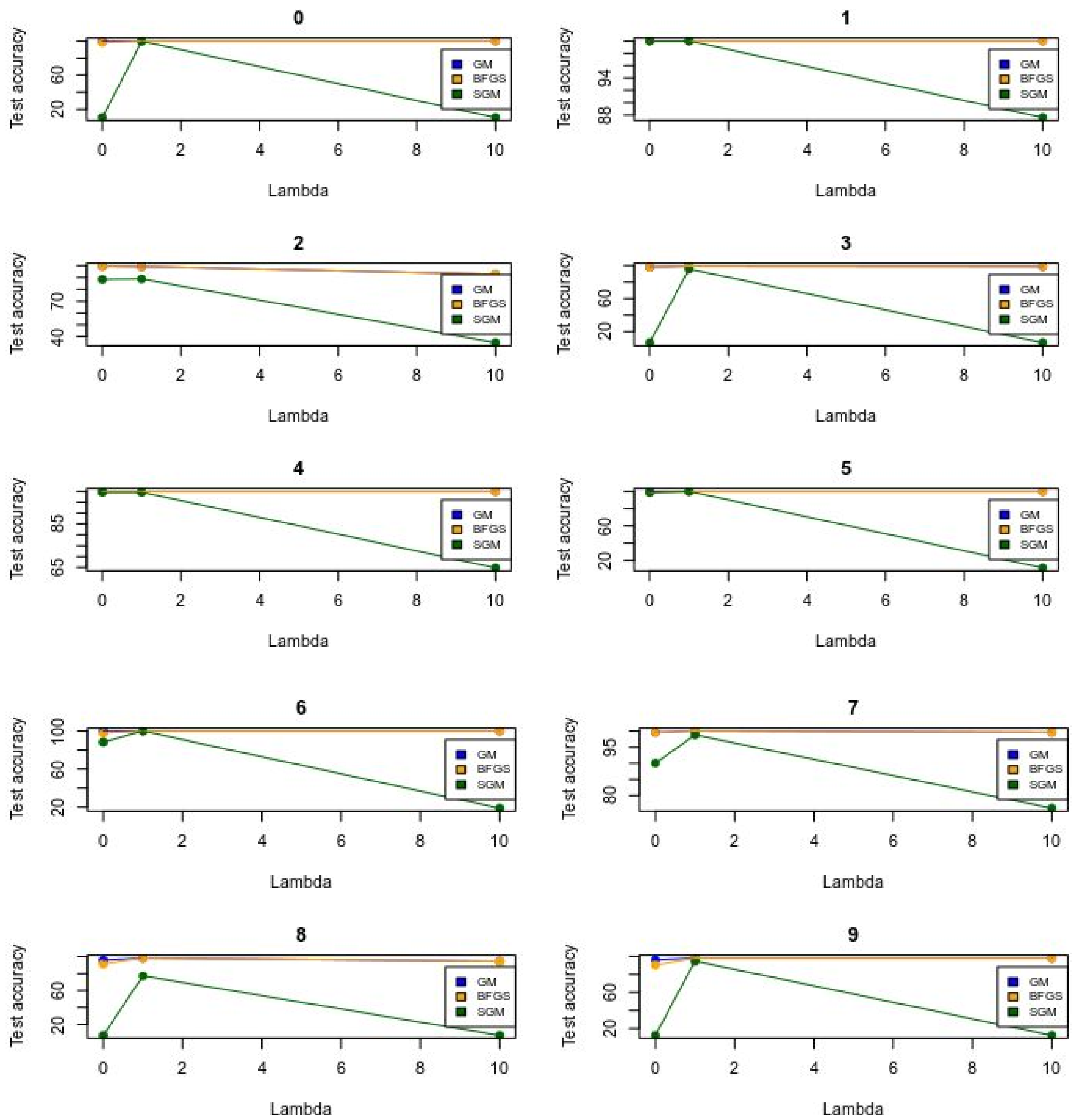
Looking at the results we can see that in general the test accuracy is at least 95% in all cases but two.

Looking at the table we could say that the best combination is *Lambda*=1 and *isd*=1 or 3.
If we take an overall look to the analysis made based on the *lambda-isd* executions we can see that:

1. *isd*=7 is the most problematic of the 3, in the earlier study and now returning sometimes low test accuracy values.
2. *isd=1,3* and *lambda*=1 gave the best loss and convergence results and behave greatly when looking at the accuracy.

If we were to choose a *lambda-isd* combination for our executions to have the best results it would be *isd*=3 and *lambda*=1.

Now we will take another approach focusing on the effect of every target number on the accuracy. Looking at the mean values of the accuracy for each number we can be able to identify special cases.



Mean value of accuracy for all numbers.

num_target	tr_acc	te_acc
0	89.51	79.78
1	99.51	98.62
2	94.62	88.36
3	88.84	77.82
4	98.09	96.00
5	94.80	89.82
6	94.62	89.38
7	98.44	95.91
8	86.84	73.69
9	89.20	77.56

We can clearly see that the values 8, 9 and 3 are harder to find than the others with the worst case being 8.

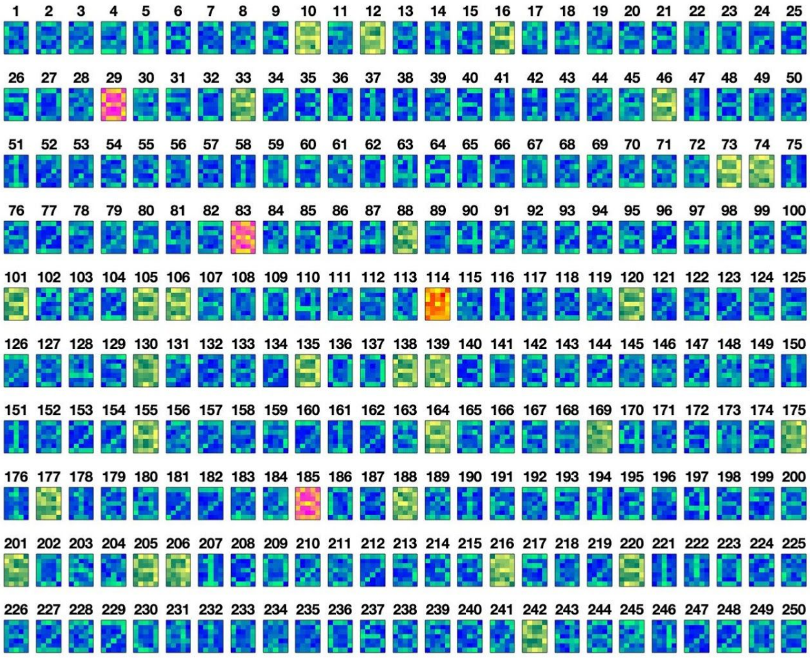
As seen in the graphs above, the configuration *Landa = 10, isd = 7* always perform bad, as *Landa = 0, isd = 7* performs wrongly for certain numbers. If we remove *L = 10, isd = 7*, the percentages of accuracy will be increased, as it is showed below:

num_target	**tr_acc**	**te_acc**
0	94.05	88.45
1	100.00	100.00
2	98.20	95.05
3	93.35	86.75
4	100.00	99.90
5	100.00	99.65
6	99.40	98.20
7	99.40	98.40
8	91.15	82.00
9	93.50	85.70

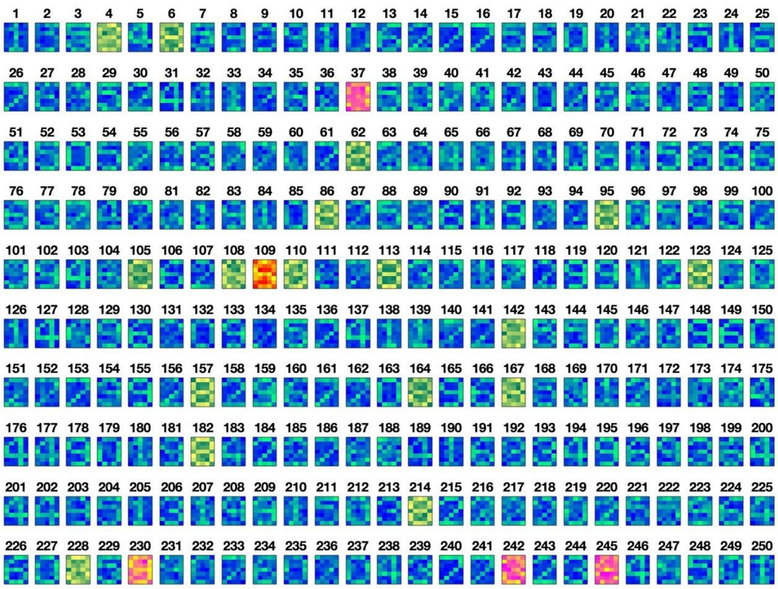
As we can see, the performance of the classification has increased notably, and the numbers with more error are still 8,9 and 3.

We will now take a look at the plot result for both this numbers. For this execution we are going to use the best *lambda-isd* combination we found for accuracy which was *isd*=3 and *lambda*=1.

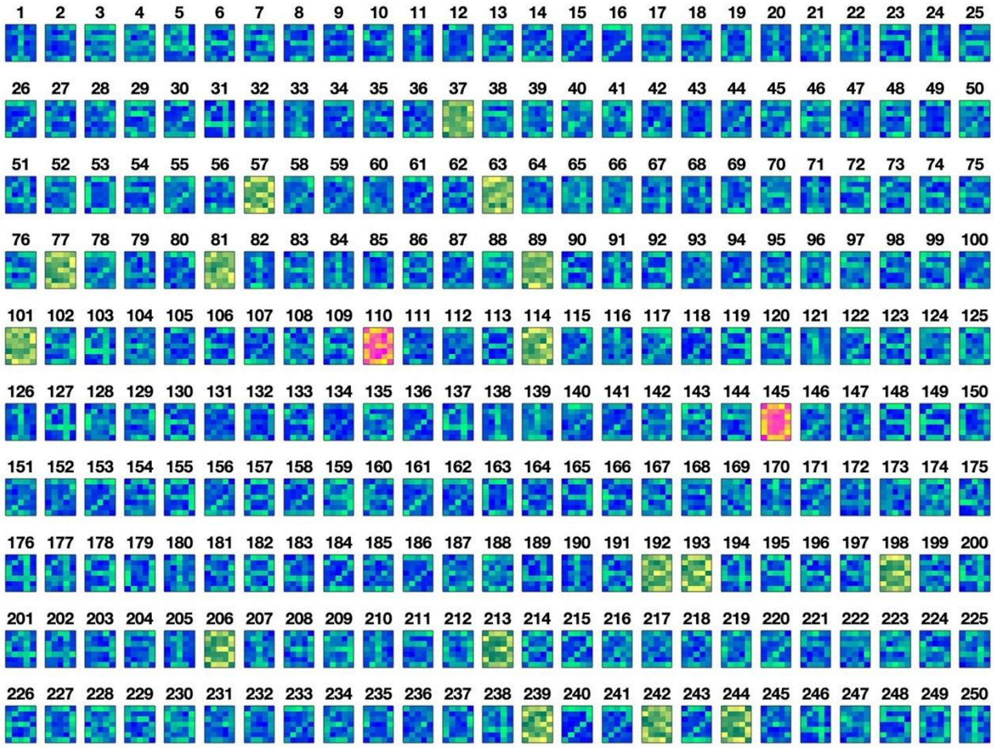
For the number 9:



For the number 8:



For the number 3:



For the number 8 the result is the worse, the errors are mostly false positives. The number 8 is very similar to four of the possible numbers (0,3,6,9) and it is harder to identify the 8's correctly.

The number 9 is also similar to 8 and 3 but not that much to 6 or 0.

The number 3 looks alike to 8 and the others but in a more subtle way.

ANNEX

num_target	Lambda	isd	niter	tex	tr_acc	te_acc	L*	tex/niter
1	0	1	3	0.0957	100	100	6.31e-16	0.0319
1	0	3	3	0.0267	100	100	5.58e-16	0.0089
1	0	7	4	0.0135	100	100	1.00e-02	0.003375
1	1	1	91	0.3626	100	100	3.40e+00	0.003984615385
1	1	3	20	0.1266	100	100	3.40e+00	0.00633
1	1	7	1000	0.0855	100	100	1.09e+01	0.0000855
1	10	1	49	0.29	100	100	1.41e+01	0.005918367347
1	10	3	35	0.1678	100	100	1.41e+01	0.004794285714
1	10	7	1000	0.0697	95.6	87.6	Inf	0.0000697
2	0	1	59	0.2339	100	99.6	5.49e-07	0.00396440678
2	0	3	83	0.3446	100	99.6	2.52e-07	0.004151807229
2	0	7	20	0.0027	94.4	88.4	1.35e+01	0.000135
2	1	1	217	0.6961	100	99.2	6.61e+00	0.003207834101
2	1	3	24	0.1347	100	99.2	6.61e+00	0.0056125
2	1	7	1000	0.0578	95.2	88.8	1.86e+01	0.0000578
2	10	1	84	0.3184	98	92.8	2.13e+01	0.00379047619
2	10	3	41	0.2322	98	92.8	2.13e+01	0.005663414634
2	10	7	1000	0.0713	66	34.8	Inf	0.0000713
3	0	1	1000	2.6894	100	98	8.21e-05	0.0026894
3	0	3	15	0.0518	100	98	1.96e-34	0.003453333333
3	0	7	13	0.0018	52.8	6.4	1.18e+02	0.0001384615385
3	1	1	430	1.2203	100	99.2	1.07e+01	0.002837906977
3	1	3	31	0.1515	100	99.2	1.07e+01	0.004887096774
3	1	7	1000	0.0563	96.4	95.6	2.75e+01	0.0000563
3	10	1	1000	1.9149	98.8	98.8	2.99e+01	0.0019149
3	10	3	40	0.1582	98.8	98.8	2.99e+01	0.003955
3	10	7	1000	0.0582	52.8	6.4	Inf	0.0000582
4	0	1	3	0.0131	100	100	8.19e-14	0.004366666667
4	0	3	3	0.0115	100	100	3.77e-14	0.003833333333
4	0	7	11	0.0015	100	99.6	6.98e-04	0.0001363636364
4	1	1	97	0.3687	100	100	3.33e+00	0.003801030928
4	1	3	20	0.1232	100	100	3.33e+00	0.00616
4	1	7	1000	0.0553	100	99.6	1.09e+01	0.0000553
4	10	1	39	0.1709	100	100	1.36e+01	0.004382051282

4	10	3	34	0.1552	100	100	1.36e+01	0.004564705882
4	10	7	1000	0.055	82.8	64.8	Inf	0.000055
5	0	1	109	0.3369	100	100	5.60e-07	0.003090825688
5	0	3	15	0.0648	100	98.4	1.30e-25	0.00432
5	0	7	12	0.0017	100	98.8	2.33e-01	0.0001416666667
5	1	1	176	0.6278	100	100	4.51e+00	0.003567045455
5	1	3	29	0.1314	100	100	4.51e+00	0.004531034483
5	1	7	1000	0.0675	100	100	1.45e+01	0.0000675
5	10	1	78	0.2472	100	100	1.77e+01	0.003169230769
5	10	3	35	0.14	100	100	1.77e+01	0.004
5	10	7	1000	0.0669	53.2	11.2	Inf	0.0000669
6	0	1	491	1.2299	100	99.6	9.56e-07	0.002504887984
6	0	3	17	0.0602	100	97.6	1.05e-12	0.003541176471
6	0	7	25	0.0027	95.6	88.4	1.08e+01	0.000108
6	1	1	268	0.7827	100	100	6.39e+00	0.002920522388
6	1	3	27	0.1331	100	100	6.39e+00	0.00492962963
6	1	7	1000	0.0696	99.6	100	1.84e+01	0.0000696
6	10	1	84	0.3025	100	100	2.12e+01	0.003601190476
6	10	3	43	0.2098	100	100	2.12e+01	0.004879069767
6	10	7	1000	0.0551	56.4	18.8	Inf	0.0000551
7	0	1	3	0.0118	100	99.6	1.75e-14	0.003933333333
7	0	3	3	0.0106	100	99.6	2.18e-14	0.003533333333
7	0	7	4	0.0013	95.2	90	9.79e+00	0.000325
7	1	1	121	0.3642	100	100	3.81e+00	0.003009917355
7	1	3	19	0.1113	100	100	3.81e+00	0.005857894737
7	1	7	1000	0.0564	100	98.8	1.20e+01	0.0000564
7	10	1	56	0.2127	100	99.6	1.50e+01	0.003798214286
7	10	3	39	0.1863	100	99.6	1.50e+01	0.004776923077
7	10	7	1000	0.0576	90.8	76	Inf	0.0000576
8	0	1	1000	2.4774	100	95.6	1.21e-01	0.0024774
8	0	3	21	0.0679	100	91.2	1.77e-11	0.003233333333
8	0	7	4	0.0012	52.4	7.2	1.19e+02	0.0003
8	1	1	712	1.6766	98.8	98	1.41e+01	0.002354775281
8	1	3	31	0.1496	98.8	98	1.41e+01	0.004825806452
8	1	7	1000	0.0664	86.4	77.2	3.18e+01	0.0000664
8	10	1	1000	1.8282	96.4	94.4	3.35e+01	0.0018282
8	10	3	39	0.1742	96.4	94.4	3.35e+01	0.004466666667

8	10	7	1000	0.0698	52.4	7.2	Inf	0.0000698
9	0	1	1000	2.657	100	96	2.26e-04	0.002657
9	0	3	17	0.0574	100	90.4	7.83e-28	0.003376470588
9	0	7	4	0.001	54.8	12.4	1.13e+02	0.00025
9	1	1	426	1.2436	99.6	98	9.61e+00	0.002919248826
9	1	3	29	0.1244	99.6	98	9.61e+00	0.004289655172
9	1	7	1000	0.0691	97.2	94.8	2.53e+01	0.0000691
9	10	1	132	0.4102	98.4	98	2.76e+01	0.003107575758
9	10	3	41	0.1852	98.4	98	2.76e+01	0.004517073171
9	10	7	1000	0.0696	54.8	12.4	Inf	0.0000696
0	0	1	643	1.8306	100	99.6	1.25e-06	0.002846967341
0	0	3	14	0.0558	100	98	4.34e-29	0.003985714286
0	0	7	4	0.001	53.2	10.4	1.17e+02	0.00025
0	1	1	351	1.1039	99.6	100	7.77e+00	0.003145014245
0	1	3	29	0.1471	99.6	100	7.77e+00	0.005072413793
0	1	7	1000	0.0595	100	99.6	2.22e+01	0.0000595
0	10	1	126	0.4355	100	100	2.50e+01	0.003456349206
0	10	3	35	0.175	100	100	2.50e+01	0.005
0	10	7	1000	0.0574	53.2	10.4	Inf	0.0000574