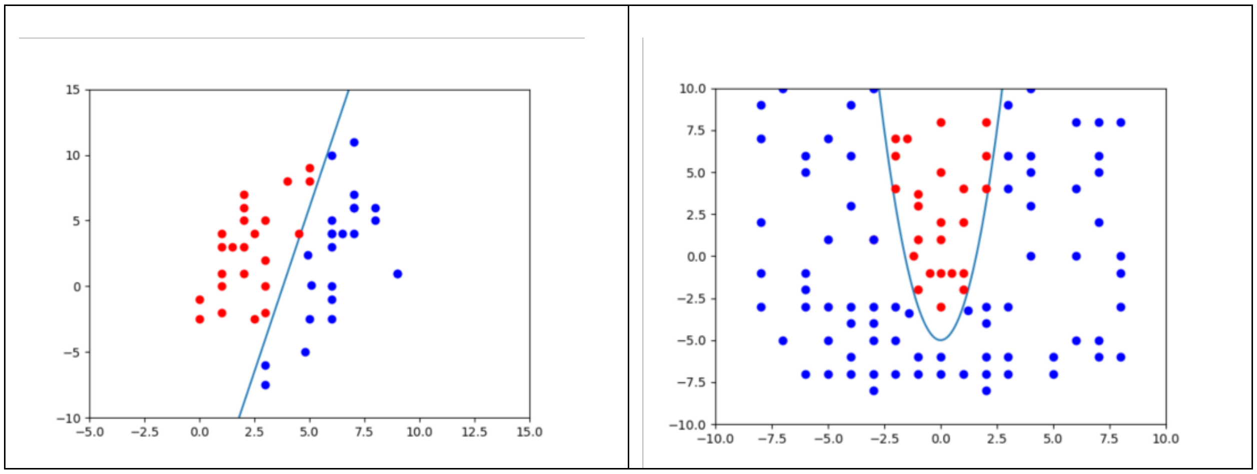
General results:

2nd degree polynomial

Problem with not overlapping set of data and 2nd degree polynomial is solved by my algorithm in average of 14 generations, best result is 100% and average result is 97%. Average execution time is 2 sec

Set of lines has length of 100 lines. Set of points has length of 50 (25 positive, 25 negative).

Probability of mutation is set to 50%. Crossover coefficient is 0.5 which means that numbers are crossed by the half of their length.



3rd degree polynomial

Problem with not overlapping set of data and 3rd degree polynomial is solved by my algorithm in average of 31 generations, best result is 100% and average result is 98%. Average execution time is 8 sec

Set of lines has length of 100 lines. Set of points has length of 100 (25 positive, 75 negative).

Probability of mutation is set to 50%. Crossover coefficient is 0.5 which means that numbers are crossed by the half of their length.

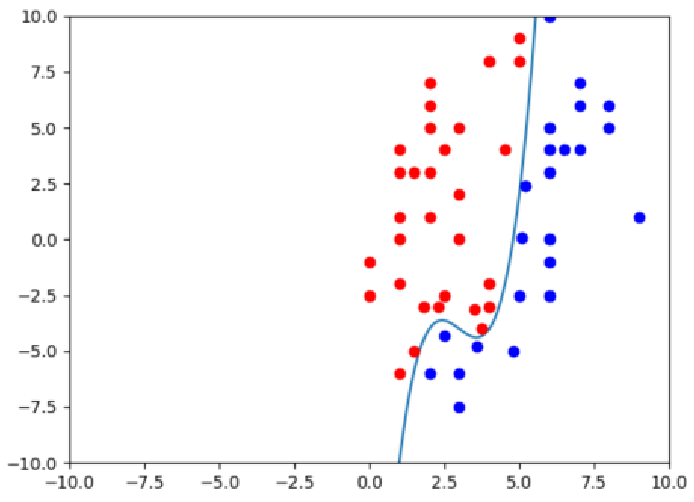
4rd degree polynomial

Problem with overlapping set of data and 4th degree polynomial is solved by my algorithm in average of 82 generations, best result is 100% and average result is 97%. Average execution time is 11 sec

It was not enough for algorithm to solve this problem with 100% if number of generations is limited to 100 but if it is limited to 200 it is possible, so I changed limit to 200.

Set of lines has length of 100 lines. Set of points has length of 54 (30 positive, 24 negative).

Probability of mutation is set to 50%. Crossover coefficient is 0.5 which means that numbers are crossed by the half of their length.



Size of data

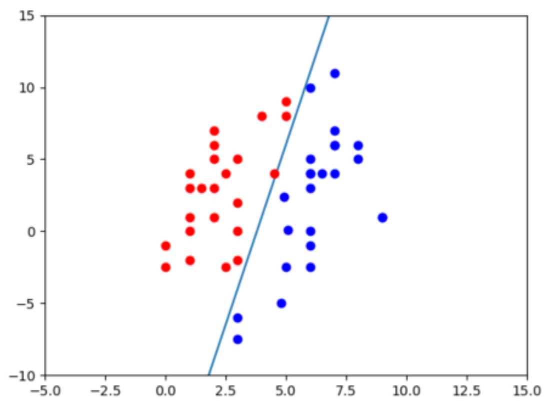
Size of data has influence on results. Percentage of final solution differs a lot. For example if there are 100 points and line which is found has 98 points distributed correctly(i.e. above or beyond) it would be 98% of accuracy, for 10 points and 9 correct it would be 90% of accuracy and etc. The biggest difference is shown when amount of border points is different. If "border" is tight - it is hard to find perfect solution, but if the amount of point is the same but border points are distributed in different manner. 100% solution could be found easily.

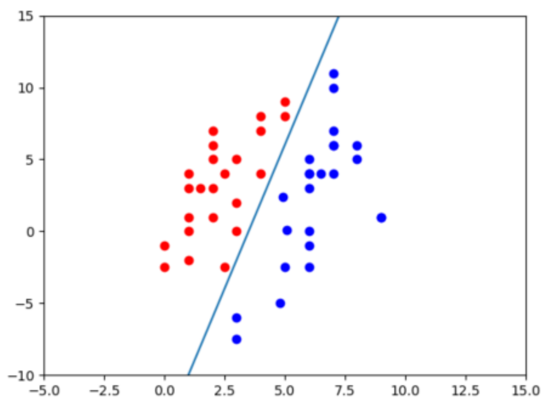
For test the same algorithm was run 10 times. Each for the set of points which contains 47 points and 100 lines. Algorithm is doing 100 generations, has mutation variance equal to 50% and crossover is distributed randomly.

For set of point on the second graph it took algorithm in average 8 generations to find 100% solutions and for the first graph, where border is narrow, it took in average 12 generations to find 97% solutions and only 1 of the solutions was 100%.

The same situation will appear for polynomial of higher degreases.

Size of data not affect time of execution in major way. It affects only fitness function where simple mathematical operation is done, so global time of execution is the mostly the same for all sizes of data.





All other tests I will do using 3rd degree polynomial because it is middle representation of complexity of algorithms

Population size

10 lines in start population:

Average generation of solution: 16, Average percent of solution: 92 %, Best result 93 %

Execution time = 0.8 sec

50 lines in start population:

Average generation of solution: 31, Average percent of solution: 97 %, Best result 100 %

Execution time = 3.9 sec

100 lines in start population:

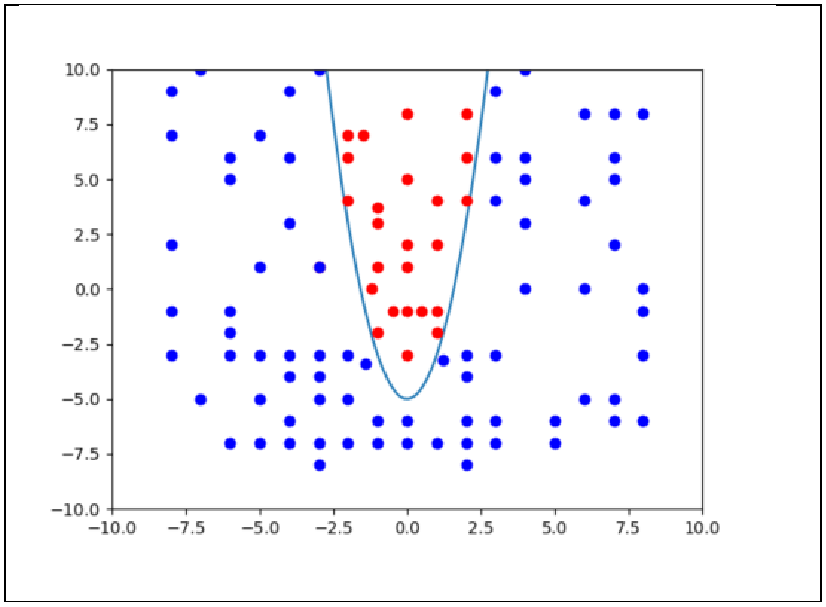
Average generation of solution: 30, Average percent of solution: 98 %, Best result 100 %

Execution time = 8 sec

Execution time differs respectively to the difference of population size: i.e. if population is bigger in to times, than execution time is also bigger in two times

On the diagram graph with percentage 100 % is presented.

Time difference



Amount of generations

10 generations:

Average generation of solution: 5, Average percent of solution: 94 %, Best result 96 %

Execution time = 1 sec

50 generations:

Average generation of solution: 27, Average percent of solution: 97 %, Best result 100 %

Execution time = 4 sec

100 generations:

Average generation of solution: 35, Average percent of solution: 98 %, Best result 100 %

Execution time = 8 sec

300 generations:

Average generation of solution: 42, Average percent of solution: 98 %, Best result 100 %

Execution time = 23 sec

As we can see the difference between 10, 50 and 100 generations exists. Solution is found later than 50 generation, so when 100 of them is used, Average percent of solution is the highest. When amount is increased to 300 there is no difference with previous step, so 100 generations is optimal solution.

Time difference is significant and respective to difference in generations: 3 times more generations, 3 times more time and etc. It is obvious, because for 100 generations code is run 100 times for 300 generations 300 times and etc.

Mutation probability

1 %:

Average generation of solution: 54, Average percent of solution: 97 %, Best result 100 %

Execution time = 7.7 sec

50 %:

Average generation of solution: 35, Average percent of solution: 98 %, Best result 100 %

Execution time = 7.8 sec

100 %:

Average generation of solution: 55, Average percent of solution: 98, Best result 99.0

Execution time = 7.9 sec

As we can see Average generation of solution is the best when mutation probability is around 50%

Time difference is not so much, because only one operation depends on mutation probability.

Crossover coefficient

0.5:

Average generation of solution: 35, Average percent of solution: 98 %, Best result 100 %

0.6:

Average generation of solution: 36, Average percent of solution: 98 %, Best result 100 %

0.9:

Average generation of solution: 39, Average percent of solution: 98 %, Best result 100 %

1:

Average generation of solution: 55, Average percent of solution: 98 %, Best result 100 %

As we can see, crossover coefficient does not influence result that much. Main work is done by mutation, but if we simply interchange coefficients between each gene we have done nothing, so only mutation doing something which leads our result to become worse.

Execution time is not changing because the same number of operations is done no matter how many bits of number is interchanged.

Selection method

I have chosen method which looks like that we randomly chose 2 lines from set of lines, from which 2 best are already removed. If line is already chosen it couldn’t be chosen again. This method is “random” one.

It is possible to use some other method, for example make crossover of 1 and 51 or 2 and 52 but in that situation each generation will be made with the same genes which is not good. So random method is the best.