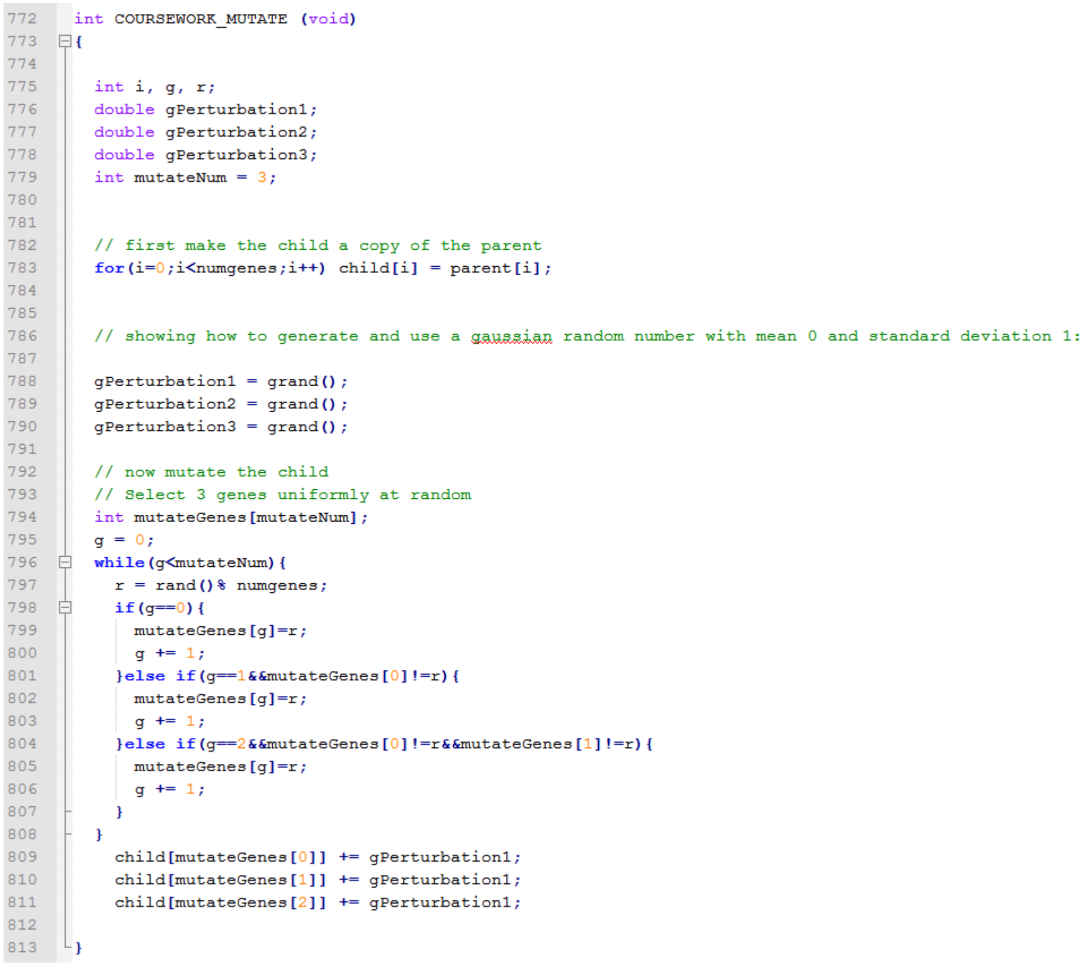
Biological Inspired Computing Coursework 2: Evolving Neural Network to Classify Neuromuscular signals.

Coding

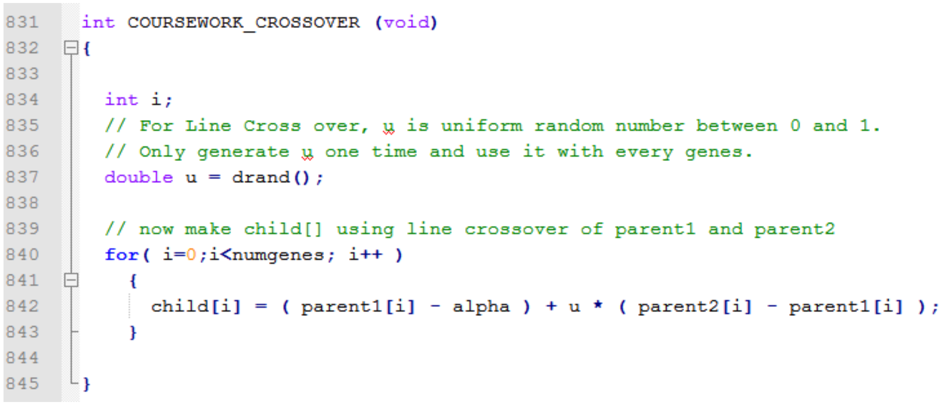
There are three functions that were implemented in order to conduct experiments later. The source codes are provided as follow.

Mutation



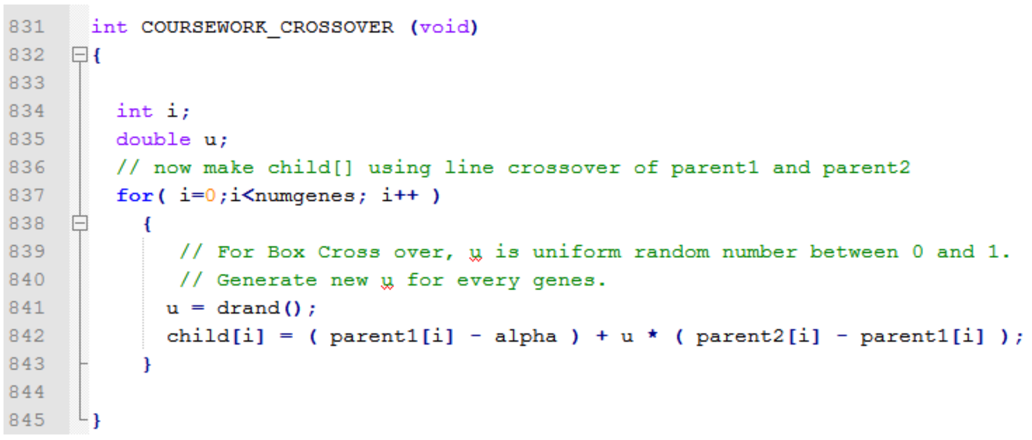
```c
int COURSEWORK_MUTATE (void)
{

  int i, g, r;
  double gPerturbation1;
  double gPerturbation2;
  double gPerturbation3;
  int mutateNum = 3;


  // first make the child a copy of the parent
  for(i=0;i<numgenes;i++) child[i] = parent[i];


  // showing how to generate and use a gaussian random number with mean 0 and standard deviation 1:

  gPerturbation1 = grand();
  gPerturbation2 = grand();
  gPerturbation3 = grand();

  // now mutate the child
  // Select 3 genes uniformly at random
  int mutateGenes[mutateNum];
  g = 0;
  while(g<mutateNum){
    r = rand()% numgenes;
    if(g==0){
      mutateGenes[g]=r;
      g += 1;
    }else if(g==1&&mutateGenes[0]!=r){
      mutateGenes[g]=r;
      g += 1;
    }else if(g==2&&mutateGenes[0]!=r&&mutateGenes[1]!=r){
      mutateGenes[g]=r;
      g += 1;
    }
  }
    child[mutateGenes[0]] += gPerturbation1;
    child[mutateGenes[1]] += gPerturbation1;
    child[mutateGenes[2]] += gPerturbation1;

}
```

Box Crossover



```c
int COURSEWORK_CROSSOVER (void)
{

  int i;
  // For Line Cross over, u is uniform random number between 0 and 1.
  // Only generate u one time and use it with every genes.
  double u = drand();

  // now make child[] using line crossover of parent1 and parent2
  for( i=0;i<numgenes; i++ )
    {
      child[i] = ( parent1[i] - alpha ) + u * ( parent2[i] - parent1[i] );
    }

}
```

Line Crossover



```
int COURSEWORK_CROSSOVER (void)
{

  int i;
  double u;
  // now make child[] using line crossover of parent1 and parent2
  for( i=0;i<numgenes; i++ )
    {
       // For Box Cross over, u is uniform random number between 0 and 1.
       // Generate new u for every genes.
      u = drand();
      child[i] = ( parent1[i] - alpha ) + u * ( parent2[i] - parent1[i] );
    }

}
```

Experiments

The seed values are generated by Microsoft Excel random number generator function (RANDBETWEEN (100, 10000)) between 100 and 10000. Five different seeds are used for each experiments. The results is showed in a table below.

Experiments		Seeds(8725)	Seeds(4023)	Seeds(476)	Seeds(7716)	Seeds(6444)
Line Crossover(α=0)	Training	86.7925%	83.0189%	82.0755%	83.9623%	84.9057%
	Testing	74.5283%	75.4717%	70.7547%	71.6981%	76.4151%
	Fitness	86.6764	82.8562	81.9424	83.8143	84.7434
Extended Line Crossover(α=0.1)	Training	83.9623%	83.0189%	88.6792%	83.9623%	80.1887%
	Testing	64.1509%	71.6981%	70.7547%	70.7547%	67.9245%
	Fitness	83.8236	82.855	88.5451	83.8138	80.0314
Extended Line Crossover(α=0.25)	Training	86.7925%	82.0755%	83.9623%	81.1321%	81.1321%
	Testing	66.9811%	64.1509%	66.0377%	61.3208%	60.3774%
	Fitness	86.6128	81.9274	83.8053	80.9934	80.9963
Box Crossover(α=0)	Training	74.5283%	85.8491%	86.7925%	83.9623%	79.2453%
	Testing	68.8679%	74.5283%	72.6415%	65.0943%	66.9811%
	Fitness	74.35	85.6897	86.6379	83.8422	79.0993
Extended Box Crossover(α=0.1)	Training	79.2453%	91.5094%	83.9623%	81.1321%	84.9057%
	Testing	68.8679%	70.7547%	72.6415%	65.0943%	72.6415%
	Fitness	79.1071	91.3317	83.7994	80.9687	84.7843
Extended Box Crossover(α=0.25)	Training	88.6792%	83.9623%	85.8491%	80.1887%	79.2453%
	Testing	72.6415%	67.9245%	67.9245%	65.0943%	70.7547%
	Fitness	88.5308	83.8084	85.7279	80.0597	79.1144

Discussion

The performance of very results from the table above are mostly acceptable. Overall results of Line Crossover operator seem to show better performance than other type of Line Crossover operators and all of Box Crossover operators. Almost all Crossover operators perform well with training data set, none of them have accuracies below 70%. It's quite obvious since the weights between each node are adjusted by data pattern in training data set. So the performance of Crossover operator should measure from the performance of testing data set. It is clear that Line Crossover have the best performance. From seeds (8725), an accuracy from testing data set is 74.5283% with Line Crossover operator, no operators have better performance than this. With seeds (4023), the highest accuracy is still Line Crossover operator at 75.4717% accuracy. The performance from seeds (7716) is 71.6981% and seeds (6444) is 76.4151% which outperform other operators. Only seeds (476) that have lower accuracy than Box Crossover and Extended Box Crossover. One more thing that have clearly show here is the value of alpha (α). Since each Crossover functions are processed in the same way as you can see in the Coding section, increasing an alpha value tend to lower an accuracy when perform against testing data set.

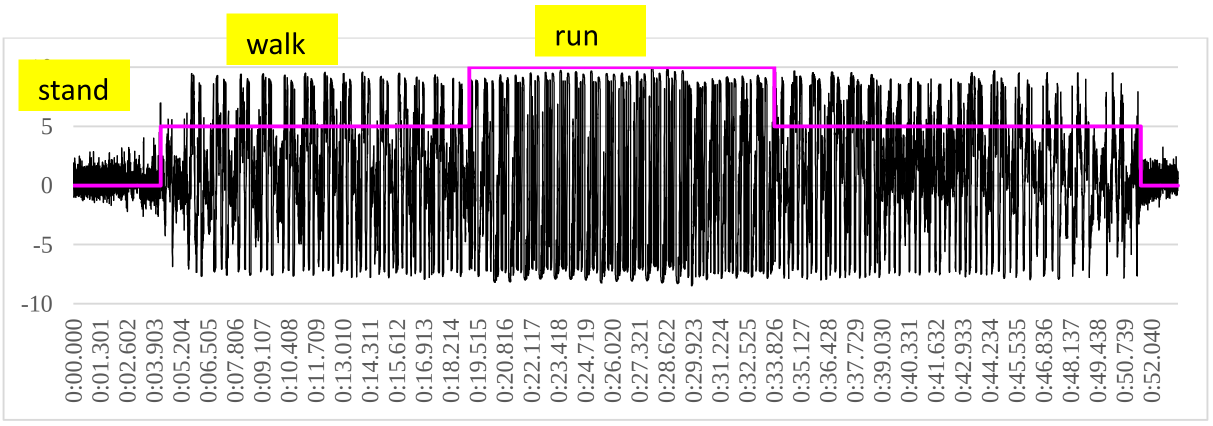
Extra Experiments

The original training data set and testing data set are changed to match up with two scenarios. The three output fields for both scenarios are changed from three to two fields. For secenario1, '1 0' mean walking and '0 1' mean other movement classes. For secenario2, '1 0' mean standing and '0 1' mean other movement classes. The number of output input file is changed from 3 to 2. The Crossover operator for these experiments is Extended Box Crossover operator with α (alpha) = 1.

Experiments		Seeds(8725)	Seeds(4023)	Seeds(476)	Seeds(7716)	Seeds(6444)
Scenario 1	Training	84.9057%	86.7925%	86.7925%	85.8491%	87.7358%
	Testing	51.8868%	54.717%	61.3208%	56.6038%	58.4906%
	Fitness	84.7574	86.6474	86.6537	85.7099	87.6027
Scenario 2	Training	100%	100%	100%	100%	100%
	Testing	94.3396%	94.3396%	90.566%	96.2264%	93.3962%
	Fitness	99.9998	99.9993	99.998	99.9989	99.9996

Discussion

The results of scenario2 from table above clearly showed impressive accuracy for both training data set and testing data set from Extended Box Crossover operator. On another hand, this operator did not perform well with scenario1, most of accuracies of testing set are below 60%. But poor performance and very good performance in this two scenario are actually come from poor classification of data set. As you can see from the figure below, by grouping other classes and run against standing class alone as in scenario2, the range of signal are clearly distinguished between two classes since most of frequencies of standing are very low. When you mixed standing and running to the same class and run against walking like in scenario1, the frequencies become harder to tell the difference. The training set performance is fine since the weights are adjusted by the pattern of the training set. But the range of 'not walking' is quite big, when running against testing data set, it is hard to produce a correct class.


stand
walk
run
0:00.000
0:01.301
0:02.602
0:03.903
0:05.204
0:06.505
0:07.806
0:09.107
0:10.408
0:11.709
0:13.010
0:14.311
0:15.612
0:16.913
0:18.214
0:19.515
0:20.816
0:22.117
0:23.418
0:24.719
0:26.020
0:27.321
0:28.622
0:29.923
0:31.224
0:32.525
0:33.826
0:35.127
0:36.428
0:37.729
0:39.030
0:40.331
0:41.632
0:42.933
0:44.234
0:45.535
0:46.836
0:48.137
0:49.438
0:50.739
0:52.040