Powerplay

Evolutionary powerplay for Nascence materials

- This example contains a search for classifiers

Jan Koutnik, IDSIA, E−mail: hkou.idsia@ch

```
SetDirectory[NotebookDirectory[]]
```

```
/home/koutnij/git/NASCENCE/alg/mathematica/powerplay
```

```
LaunchKernels[2]
```

```
{KernelObject[1, local], KernelObject[2, local]}
```

Imports

```
Import["../libMLP/bp.m"]
```

```
Import["../libNES/nes10.m"]
```

```
Import["../libCoSyNE/libCosyne9.m"]
```

```
Import["../vm/vmWrapper1.m"]
```

VM

In Mathematica

Vitual material is a simple MLP here :

```
nInputs = 20;
```

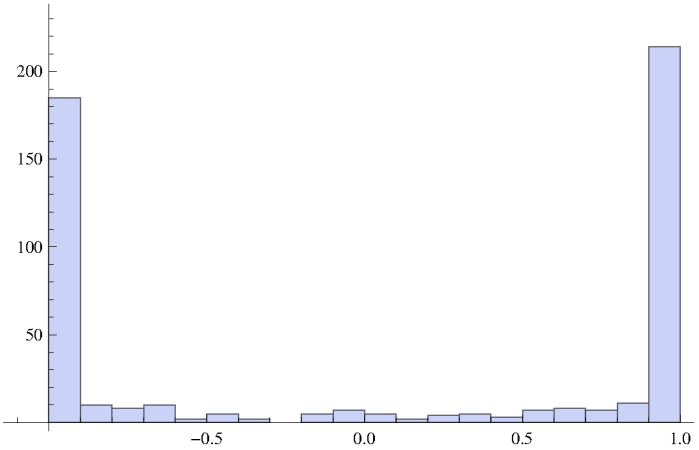
```
actF = Tanh;
```

```
vm = randomNet[{nInputs, 500, 1}];
```

```
evalVM[vm_, x_] := Last@fwdPass[vm, x]
```

```
res = Sort@Flatten[evalVM[vm, #] & /@ RandomReal[{-1, 1}, {5, nInputs}]];
Min@res
Max@res
```

```
Histogram[res, 20]
```




```
evalVM[vm_, config_, input_] := evalVM[vm, Join[config, input]]
```

```
plotBounds[vm_, config_, set_] :=
 With[{points = (First /@ #) & /@ SortBy[GatherBy[set, Last], #[[1, 2, 1]] &]}, Show[{
    ContourPlot[Sign@First@evalVM[vm, config, {x, y}], {x, -1, 1},
     {y, -1, 1}, Contours → {0}, ContourShading → {LightRed, LightGreen},
     ContourStyle → Directive[Black, Dashed], FrameTicks → False, ImageSize → 100],
    ListPlot[points, PlotStyle → {Darker@Red, Darker@Green}]
   }]]
```

Nascence API VM

```
vmHost = "localhost";
vmPort = 9090;
```

```
vmPath = "git/NASCENCE/vm";
```

```
vmClient = vmConnect[vmHost, vmPort, vmPath]
```

```
« JavaObject[nascence.vm.io.MathClient] »
```

```
vmClient@programmeVarElmanRandom[20, 1000, 20, 3.0, 0.5, False, False]
```

This function evaluates the VM via Nascence API :

```
evalVMN[vmClient_, config_, input_] :=
 First[vmClient@evaluateArray[{Join[config, input]}, 255, Range[0, nPins - 2]]]
```

```
AbsoluteTiming@evalVMN[vmClient, RandomReal[{-1, 1}, 17], RandomReal[{-1, 1}, 2]]
```

```
{0.007705, {1.}}
```

```
evalVM = evalVMN
```

```
evalVMN
```

```
vm = vmClient
```

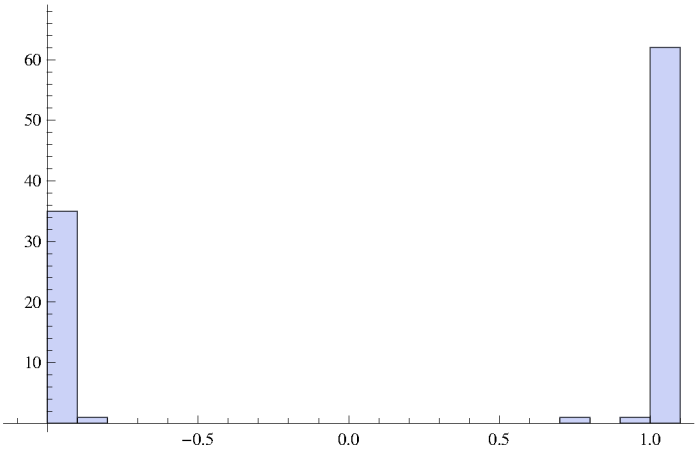
```
« JavaObject[nascence.vm.io.MathClient] »
```

```
res = Sort@Flatten[MapThread[evalVM[vm, #1, #2] &,
      {RandomReal[{-1, 1}, {100, dim}], RandomReal[{-1, 1}, {100, 2}]}]];
Min@res
Max@res
```

```
-1.
```

```
1.
```

```
Histogram[res, 20]
```



Powerplay Experiment, 2-space Classifiers

```
SetDirectory[NotebookDirectory[]]
```

```
/home/koutnij/git/NASCENCE/alg/mathematica/powerplay
```

In this example a space of classifiers is explored.

```
maxTasks = 1000;
```

```
SeedRandom[30];
taskSet = RandomReal[{-1, 1}, {maxTasks, 2}];
```

```
fitFn[g_, set_] := With[
  {diff = Flatten[(evalVM[vm, g, #] & /@ set[[All, 1]]) - set[[All, 2]]]}, Total[diff^2]]
```

This adds a number of non - solved tasks :

```
fitFn[g_, set_] := With[{eval = Flatten[(evalVM[vm, g, #] & /@ set[[All, 1]])]},
  Mean[(eval - Flatten@set[[All, 2]])^2] + Length[set] -
   Count[MapThread[Equal, {Sign@eval, Flatten@set[[All, 2]]}], True]]
```

```
fitFn[g_] := fitFn[g, trainingSet]
```

```
solvesAllTasksQ[g_, set_] :=
 Equal[Sign@Flatten[(evalVM[vm, g, #] & /@ set[[All, 1]])], Flatten@set[[All, 2]]]
```

```
solvesAllTasksQ[g_] := solvesAllTasksQ[g, trainingSet]
```

```
optimize[pop_, fitFn_, trainingSet_, nGen_] :=
 Module[{popTmp}, NestWhile[(popTmp = coSyNEstep[#, fitFn[#, trainingSet] &,
        minimize → True, mutate → 0.8, permuteAll → True, verbose → False, elite → 1];
      Print[{popTmp[[1, 1]], solvesAllTasksQ[popTmp[[1, 2]], trainingSet]}]; popTmp) &,
   pop, Not[solvesAllTasksQ[#[[1, 2]], trainingSet]] &, 1,
   nGen]]

reevaluate[pop_, fitFn_] := SortBy[{fitFn[#], #} & /@ pop[[All, 2]], First]

appendTask[set_, config_] := With[{point = taskSet[[Length[set] + 1]]},
  Append[set, {point, -Sign[evalVM[vm, config, point]]}]]

powerplay[fitFn_, popSize_, nGen_, nTasks_] := Module[{pop, set},
  set = {{taskSet[[1]], {1}}}; (*first task*)
  pop = SortBy[
    newRandomPop[popSize, dim, NormalDistribution[0, 1], fitFn[#, set] &], First];
  Print["Generated population"];
  Nest[
   (Print["# of tasks : " <> ToString@Length[set]];
     pop = optimize[pop, fitFn, set, nGen]; (*optimize*)
     Print@plotBounds[vm, pop[[1, 2]], set];
     set = appendTask[set, pop[[1, 2]]]; (*add next task*)
     pop = reevaluate[pop, fitFn[#, set] &];
     pop) &
   , pop, nTasks]
 ]
```

Experiment

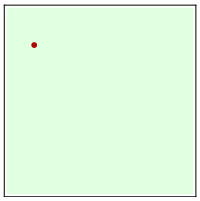
Powerplay with population size of 8, 30 generations of CoSyNE and 10 consecutive classification tasks to solve :

```
nPins = 20; nOutputs = 1; nInputs = nPins - nOutputs; dim = nInputs - 2;

popSize = 20;

fitFn[RandomReal[{-1, 1}, 17], set]
```

```
2.
```

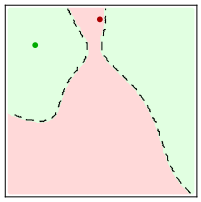
```
pop = powerplay[fitFn, 32, 300, 10];
```

```
Generated population

# of tasks : 1
```

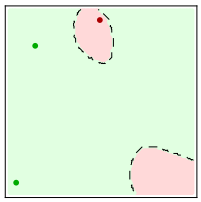


```
# of tasks : 2

{3., False}
```

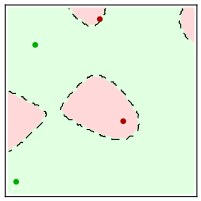
```
{0.000768935, True}
```



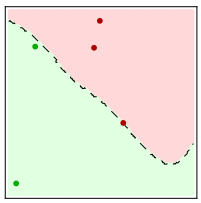
```
# of tasks : 3
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{2.33333, False}
{0.00100474, True}
```



```
# of tasks : 4
{0.0553633, True}
```

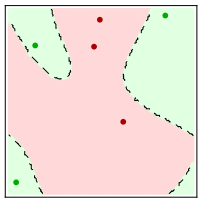


```
# of tasks : 5
```

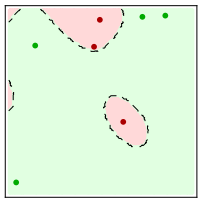


```
# of tasks : 6
{1.43508, False}
```

```
{1.43508, False}

{0.0000922722, True}
```



```
# of tasks : 7

{0.00211128, True}
```



```
# of tasks : 8

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.46886, False}

{1.42285, False}

{1.42285, False}
```

```
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
{1.42285, False}
$Aborted
```