

How do people learn multiple behaviors at the same time? To a person this seems trivial, but for machine learning neural networks this is a real problem. In general, machine learning algorithms are learned by optimizing a cost function, in doing so, minimizing error on a task. However, cost functions for different tasks are very different. This makes it **difficult for one static neural network to be trained at two tasks at the same time**.

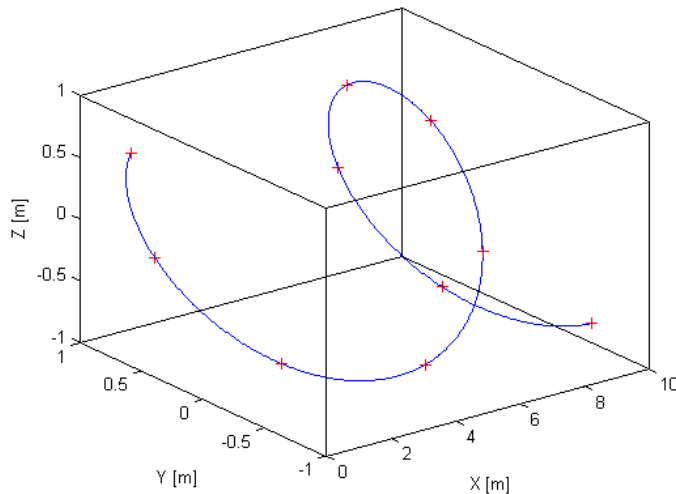
Humans solve the problem of having one system perform multiple tasks by way of focus. When focusing on one task, the brain creates situation specific memories of how performance in their current environment leads to either success or failure. Additionally, the human brain has better recall of past memories more similar to their current environment. Thus, we can say that **humans train a task within a certain brain state**, the closer they are to that brain state, the better they will be able to leverage past experiences to perform successfully.

In general, a brain state can be defined as how the neurons in the brain are firing. In specific, we could define the brain state by the voltage difference along all neuronal membranes. However, due to bounds on the complexity of the model (Want to quickly simulate firing of all nodes via a single program), I will redefine brain state as the frequency of firing of all neurons. Note that this does not take into account key features of synaptic plasticity, such as time dependent learning (model does not keep track of when neurons are firing relative to each other).

In a static network (connections and weights are unchanging), the brain state is greatly impacted by the external environment. Thus, it makes sense to use external stimulus (ie the firing frequency of sensory neurons) as a focuser for a specific task.

Based on our definition of brain state as the firing frequency of all neurons, we can **map the brain state of a neural network onto n-dimensional space**, where n is the number of neurons in the network. This is accomplished by making each axis represent the firing frequency of a distinct neuron.

For example, consider a 3 neuron brain. We can represent any brain state as a point in 3 dimensional space. To do this, let each axis represent the firing frequency of a distinct neuron. As neurons fire and affect each other, the brain state, or point in this 3-d space changes. If we plot multiple brain states over time, we get a path like that shown in the figure. This path represents the brain's 'thought process' over time. Note that if the network has some external influence such as sensory neurons, changes in sensory stimuli could lead to discontinuities in the thought process path. Additionally, there is no guarantee that a closed neural network has a continuous brain state path, to even start to prove you would need to get into the specifics of network implementation.



While imagining a  $n$ -dimensional path of brain states might be an interesting thought experiment, how can this idea be utilized to help us create neural networks able to dynamically learn multiple different tasks? The trick is to influence the neural network differently when it is in different brain states. To explain this better, we need to make some assumptions about our network:

1. The network operates iteratively, with neuronal frequencies, connections, and weights affecting the neuronal frequencies of the next time step
2. The network has some sensory neurons, which are affected by an external environment
3. The network has some output neurons, whose change in frequency causes changes in an external environment
4. The network has some level of feedback/interconnectivity (if purely feedforward, there is no stability in brain state, brain state is completely determined by stimulus)
5. The network has some purpose/goal/motivation/reward system/cost function

We can code a neural network with  $n$  neurons which functions like this relatively simply, and then we can represent this neural network's brain state in  $n$ -dimensional space. Now, I am going to mention some key insights about our brain state map:

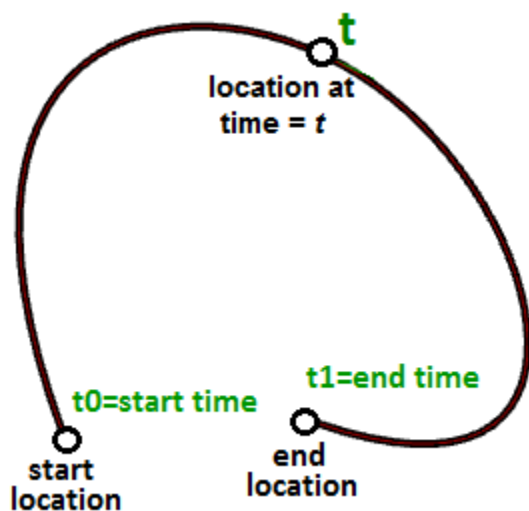
Let  $s_1$  be a point in our  $n$ -dimensional map representing a brain state during which the corresponding output behavior is causing our reward system to give the brain some level of reward  $r_1$ .

If  $s_2$  is a point such that  $s_2$  is spatially close to  $s_1$ , then  $r_2$  must be similar to  $r_1$

Essentially, **brain states which are close together on our n-dimensional map will correspond to similar behaviors and thus a similar reward level.** Additionally, **being in similar environmental situations will lead to spatially close brain states.** (that is, spatially close in the n-dimensional space of our brain state map)

Now, assume our network is interacting with its environment, changing brain states, and receiving reward and punishment. Over time, the network is going to traverse our n-dimensional map, and when it is in a situation it has faced in the past, it will be in a similar position. **Our goal is to use our map to enable the network to recall past experiences which are similar to its current situation and use them to influence the brain state.** However, in order to do this, we need to add yet another dimension.

In order to add an axis but stay in 3 dimensions, we must start in two dimensions. Let us imagine a 2-neuron brain. We can represent the brain state with two dimensions: let  $x$  represent the firing frequency of neuron 1, and  $y$  represent the firing frequency of neuron 2. Thus, we can represent this brain's thought process as a curve in two dimensional space. Note that by thought process, we mean the brain state over time.



Axes not shown, but clearly 2d with two neurons.

Now, assume neuron 1 is an input neuron, and neuron 2 is a behavior neuron. The points in this 2D environment represent every possible environment and every possible behavioral response. The path from  $t_0$  to  $t_1$  represents the behavioral response to input stimuli over time.

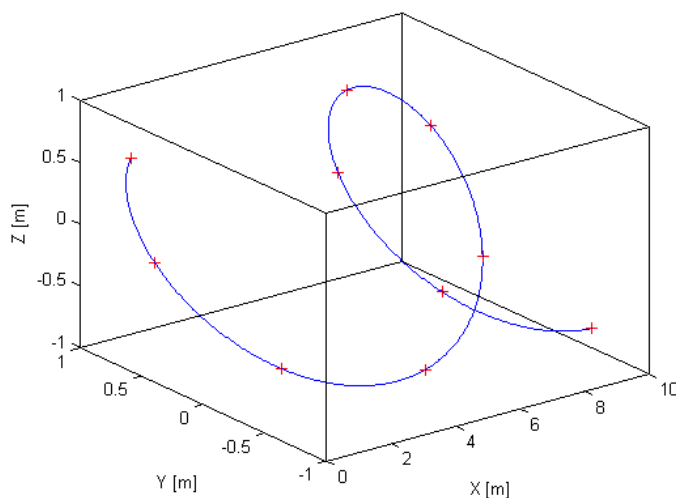
Assume that as the brain is acting and moving through brain states, it is connected to an environment. For example, a human brain is connected to an environment if it is in a physical body and the actions it makes affect the world around it in an observable way. For a computer simulated brain, we can connect it to an environment as well. For example, we could make the

inputs correspond to the location of pieces on the chessboard, and have its output cause different moves in the game.

I bring up the concept of an environment because when you are in an environment, your position in that environment changes over time, and this change is affected by your behavior. Additionally, **some environmental positions are better than others.**

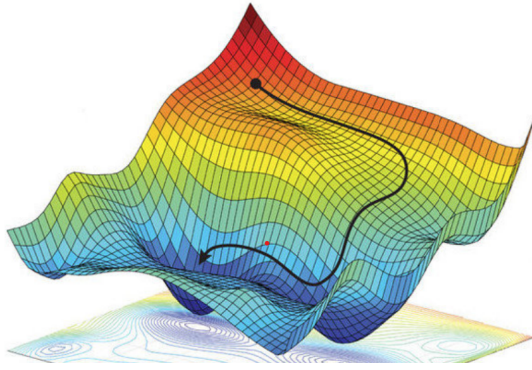
Back to our two dimensional brain. At every time  $t$ , we are in a certain position in the environment, and our brain state (point in the 2D map) reflects this position. Now, let us assume that every moment in time, we are getting an additional piece of information, the amount of reward we are receiving at the moment.

Thus we can reinterpret our 3d image differently:



Now,  $x$  and  $y$  axis represent the firing frequency of neurons, while the  $z$ -axis represents the reward the net is getting. As the net steps (traversing the path), we can see that the brain state moves back and forth between environmentally favorable and unfavorable positions (as the  $z$ -axis changes). Now that we have positive and negative experiences, we define memory.

**Memory is the prediction of reward level at a certain brain state.** For our two neuron case,  $z = M(x_1, x_2)$  is a surface in 3 dimensional space. Here is an example of what a memory function would look like.



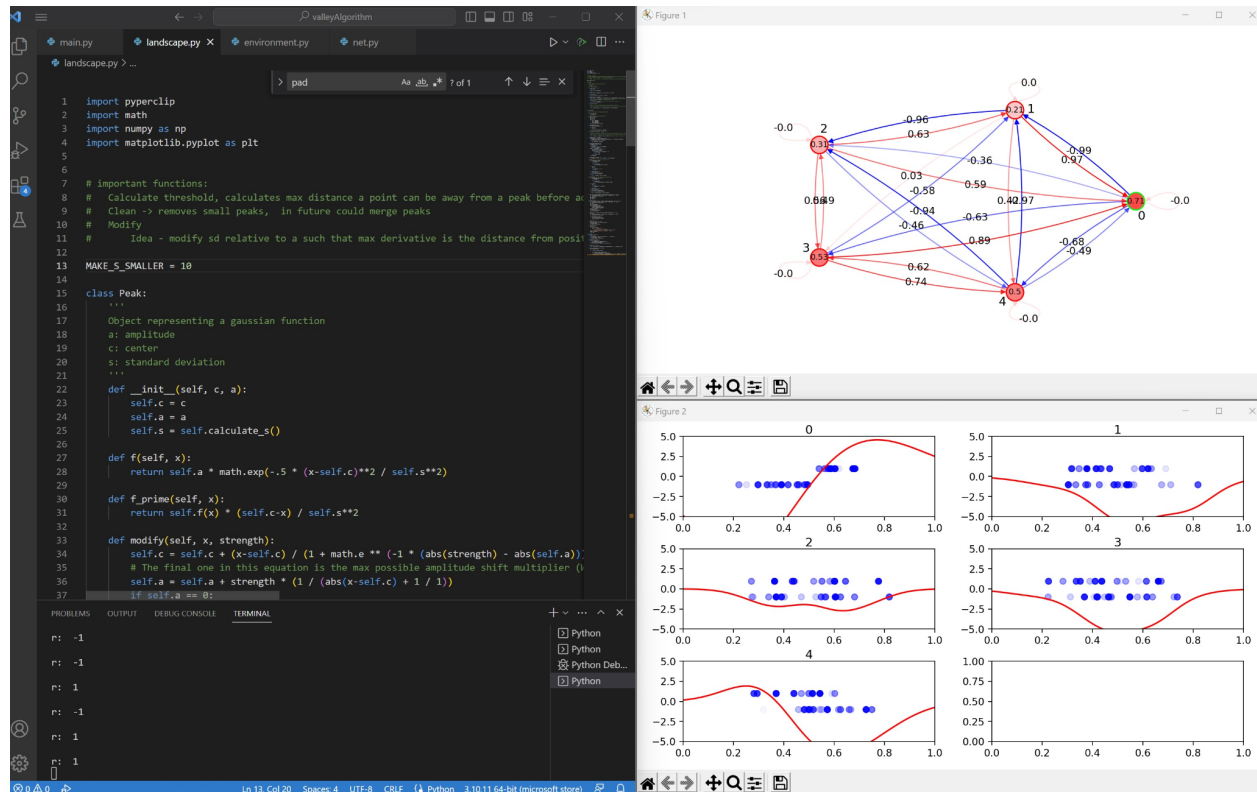
The path in this picture shows gradient descent towards a local minimum (useful when minimizing a cost function). Since our Memory surface is a reward predictor (not a cost predictor), we instead want to move up the gradient towards the highest point, this would be the brain state with the highest predicted reward.

**If we had a continuous, differentiable memory function, we could use the derivative to make small incremental changes to the weights in the right direction to move the brain state to one of higher predicted reward.**

But how to create such a memory function. 2

Valley in progress:

No learning occurring, weights assigned random value every time step



8/10, Valley in progress

No learning occurring, weights assigned random value every time step

Changed so a value moves towards r value instead of just increasing/decreasing by r

Ossification value added, displayed on landscape graph

Standard deviation no longer dynamically chosen (curr .1)

```
environment.py - valleyAlgorithm - Visual Studio Code
main.py environment.py x net.py
environment.py > BasicEnv > _init_
> [E] As ab... No results ↑ ↓ ×
70
71
72
73 def save_env(self, name:str):
74     """
75     Saves environment as pickle file in environent
76     name: name of file
77     type: str
78     """
79
80
81
82 class BasicEnv(Environment):
83     def __init__(self):
84         super().__init__(1, 0)
85         self.state = {'age': 0, 'r': 0}
86
87     def step(self, net_output):
88         """
89         Steps the basic environment
90         Increases 'age' attribute
91         """
92         self.state['age'] += 1
93         if net_output[0] > .5:
94             self.state['r'] = 1
95         else:
96             self.state['r'] = -1
97
98     def get_net_input(self):
99         """
100         Converts the environment state into values of ti
101         Returns a numpy array of length equal to net's
102         First value of net input corresponds to the val
103         """
104         return []
105
106     def get_net_reward(self):
107         """
108         Returns the net reward level based on the curre
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

