

Selected files

3 printable files

pnn_io.c

pnn.c

pnn.h

pnn_io.c

```

1  #include "pnn.h"
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  void pnn_fread(pnn * nn, FILE * stream)
7  {
8      fread(&(nn->sigma), sizeof(double), 1, stream);
9
10     int rfv_count;
11     fread(&rfv_count, sizeof(int), 1, stream);
12     nn->rfv_count = rfv_count;
13
14     nn->refs = (v2 *)malloc(sizeof(v2) * rfv_count);
15     nn->f_vals = (double *)malloc(sizeof(double) * rfv_count);
16     fread(nn->refs, sizeof(v2), rfv_count, stream);
17     fread(nn->f_vals, sizeof(double), rfv_count, stream);
18 }
19
20 void pnn_fwrite(pnn * nn, FILE * stream)
21 {
22     fwrite(&(nn->sigma), sizeof(double), 1, stream);
23     fwrite(&(nn->rfv_count), sizeof(int), 1, stream);
24
25     fwrite(nn->refs, sizeof(v2), nn->rfv_count, stream);
26     fwrite(nn->f_vals, sizeof(double), nn->rfv_count, stream);
27 }
28
29 void pnn_fprint(pnn * nn, FILE * stream, int head_count)
30 {
31     fprintf(stream, "sigma = %lf; rfv_count = %d\n", nn->sigma, nn->rfv_count);
32
33     for (int i = 0; (i < nn->rfv_count) && (i < head_count); i++)
34         fprintf(stream, "ref #%d: x = %lf, y = %lf | f value #%d: %lf\n",
35                 i, nn->refs[i].x, nn->refs[i].y, i, nn->f_vals[i]);
36 }
37

```

pnn.c

```

1  #include "pnn.h"
2
3  #include <stdlib.h>

```

```
4  #include <math.h>
5
6  static double act(v2 * x, v2 * ref, double sigma);
7  static double eucl2(v2 * a, v2 * b);
8
9  void pnn_new(double sigma,
10             double a, double b, int segment_count,
11             pnn * nn,
12             double(*f)(double, double))
13  {
14      int rfv_count = segment_count * segment_count;
15
16      nn->sigma = sigma;
17      nn->rfv_count = rfv_count;
18      nn->refs = (v2 *)malloc(sizeof(v2) * rfv_count);
19      nn->f_vals = (double *)malloc(sizeof(double) * rfv_count);
20
21      double delta = (b - a) / (double)segment_count;
22      double x, y;
23      int k;
24      for (int i = 0; i < segment_count; i++)
25      {
26          x = a + (double)i * delta;
27          for (int j = 0; j < segment_count; j++)
28          {
29              y = a + (double)j * delta;
30
31              k = i * segment_count + j;
32
33              nn->refs[k].x = x;
34              nn->refs[k].y = y;
35              nn->f_vals[k] = f(x, y);
36          }
37      }
38  }
39
40  double pnn_run(pnn * nn, v2 * x)
41  {
42      double result = 0;
43
44      for (int i = 0; i < nn->rfv_count; i++)
45          result += act(x, &(nn->refs[i]), nn->sigma) * nn->f_vals[i];
46
47      return result;
48  }
49
50  static double act(v2 * x, v2 * ref, double sigma)
51  {
52      return exp((-1.0 * (eucl2(x, ref)) / (sigma * sigma)));
53  }
54
55  static double eucl2(v2 * a, v2 * b)
56  {
57      double x = b->x - a->x;
58      double y = b->y - a->y;
59      return x * x + y * y;
```

```
60 | }  
61 |
```

pnn.h

```
1  #pragma once  
2  
3  #include <stdio.h>  
4  
5  typedef struct  
6  {  
7      double x;  
8      double y;  
9  } v2;  
10  
11 typedef struct  
12 {  
13     double sigma;  
14     int rfv_count;  
15     v2 * refs;  
16     double * f_vals;  
17 } pnn;  
18  
19 void pnn_new(double sigma,  
20             double a, double b, int segment_count,  
21             pnn * nn,  
22             double(*f)(double, double));  
23  
24 double pnn_run(pnn * nn, v2 * x);  
25  
26 void pnn_fread(pnn * nn, FILE * stream);  
27  
28 void pnn_fwrite(pnn * nn, FILE * stream);  
29  
30 void pnn_fprint(pnn * nn, FILE * stream, int head_count);  
31
```