

1 Ising images

```
1 import numpy as np
2 import numpy.linalg as linalg
3 import matplotlib.pyplot as plt
4 from PIL import Image, ImageFilter, ImageOps
5 import imageio
6 plt.rcParams['image.cmap'] = 'gray'

1 from ipywidgets import IntProgress
2 from IPython.display import display
3 import time
```

1.1 Standard Ising (on a torus)

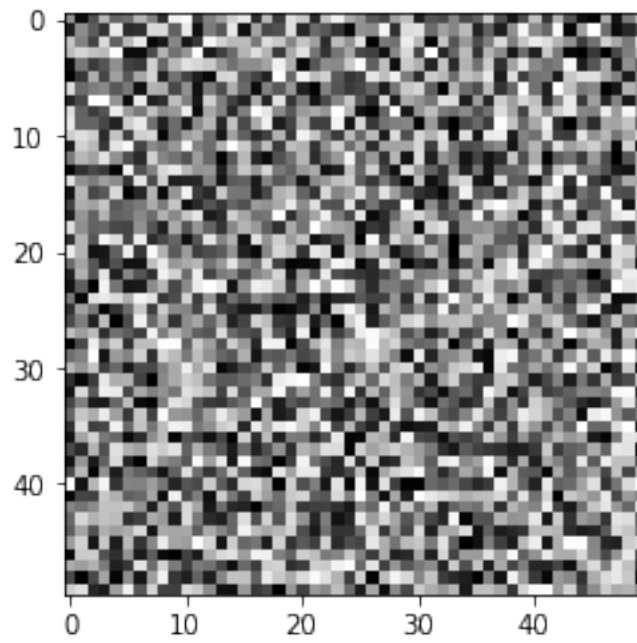
In grayscale for fun.

```
1 def neighbors(a, i, j):
2     return np.hstack([a[:,j].take([i-1,i+1], mode='wrap'),
3                       a[i,:].take([j-1,j+1], mode='wrap')])

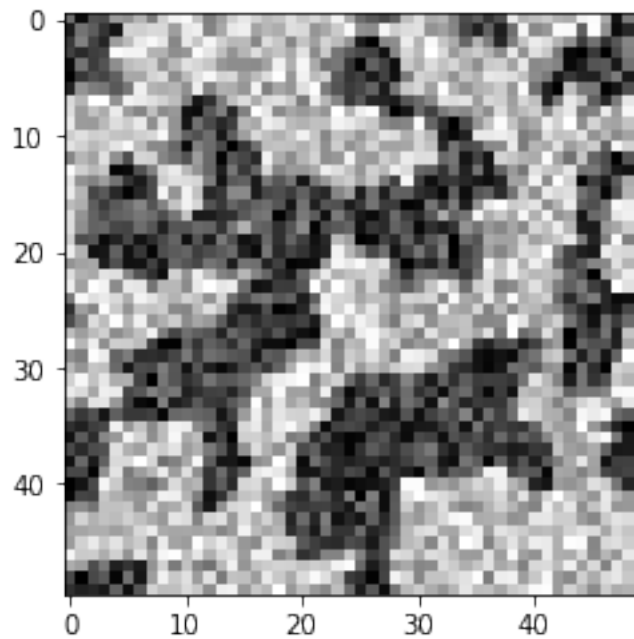
1 def energy(img, i, j):
2     return -1 + np.sum(np.abs(img[i, j] - neighbors(img, i, j)))

4 def isingstep( $\beta$ , img):
5     w, h = np.shape(img)
6     i = np.random.randint(w)
7     j = np.random.randint(h)
8     E0 = energy(img, i, j)
9     img[i, j] *= -1
10    E1 = energy(img, i, j)
11    P = np.exp(- $\beta$ *(E1 - E0)) if E1 > E0 else 1
12    if np.random.rand() > P: # Restore old
13        img[i, j] *= -1
14    return img

1 img = 2*np.random.rand(50, 50) - 1
2 plt.imshow(img);
```



```
1  n = 100000
2  for i in range(n):
3      isingstep(3 * (np.pi / 2) / np.arctan(n - i), img)
4  plt.imshow(img);
```



1.2 Image-edge Ising

```

1 edges = Image.open("ising-edges.png")
2 edata = np.array(edges) > 128
3 edges

```



```

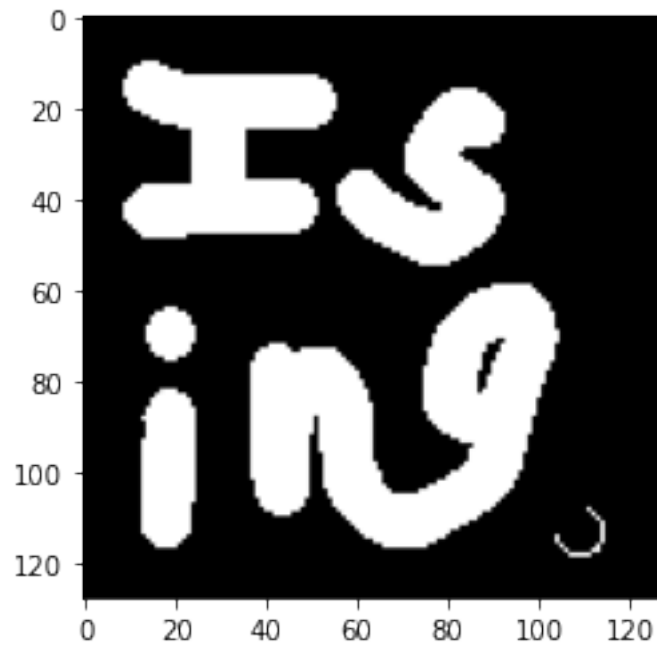
1 def eenergy(img, edges, i, j):
2     """Edge-modified Ising energy: 0 on edge."""
3     if edges[i, j]:
4         return 0
5     w, h = np.shape(img)
6     c = img[i, j]
7     l = img[i-1, j] if i > 0 else img[w-1, j]
8     r = img[i+1, j] if i < w-1 else img[0, j]
9     t = img[i, j-1] if j > 0 else img[i, h-1]

```

```

10     b = img[i, j+1] if j < h-1 else img[i, 0]
11     return -img[i, j] * (l + r + t + b)
12
13 def nenergy(img, edges, i, j):
14     """Neighbor-modified Ising energy: 0 interactions with edges."""
15     if edges[i, j]:
16         return 0
17
18     w, h = np.shape(img)
19     c = img[i, j]
20     l = r = t = b = 0
21     if i > 0:
22         l = img[i-1, j] if not edges[i-1, j] else 0
23     else:
24         l = img[w-1, j] if not edges[w-1, j] else 0
25
26     if i < w - 1:
27         r = img[i+1, j] if not edges[i+1, j] else 0
28     else:
29         r = img[0, j] if not edges[0, j] else 0
30
31     if j > 0:
32         t = img[i, j-1] if not edges[i, j-1] else 0
33     else:
34         t = img[i, h-1] if not edges[i, h-1] else 0
35
36     if j < h - 1:
37         b = img[i, j+1] if not edges[i, j+1] else 0
38     else:
39         b = img[i, 0] if not edges[i, 0] else 0
40
41     return -img[i, j] * (l + r + t + b)
42
43 def eisingstep( $\beta$ , img, edges):
44     w, h = np.shape(img)
45     i = np.random.randint(w)
46     j = np.random.randint(h)
47     E0 = nenergy(img, edges, i, j)
48     img[i, j] *= -1
49     E1 = nenergy(img, edges, i, j)
50     P = np.exp(- $\beta$ *(E1 - E0)) if E1 > E0 else 1
51     if np.random.rand() > P: # Restore old
52         img[i, j] *= -1
53     return img
54
55 def frame(writer, data):
56     writer.append_data((255 * ((eimg + 1) / 2)).astype('uint8'))
57
58 1 img = Image.open("ising-letters.png")
59 2 eimg = -1 + 2 * (np.array(img) / 255)
60 3 plt.imshow(eimg);

```



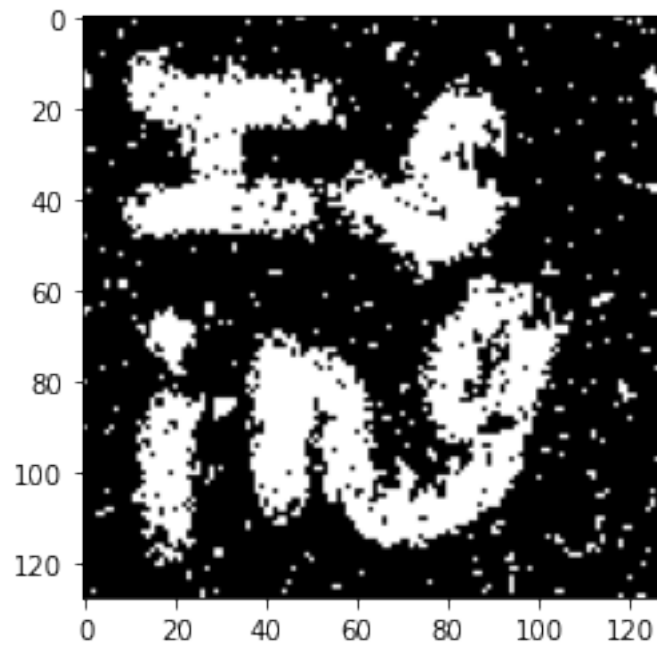
movie.gif: Full neighbor Ising.

```

1  n = 1000000
2  f = IntProgress(min=0, max=1 + (n-1) // 1000) # instantiate the bar
3  display(f)
4  with imageio.get_writer('movie.gif', mode='I') as writer:
5      frame(writer, eimg)
6      for i in range(n):
7          eisingstep(0.5 * (np.pi / 2) / np.arctan(n - i), eimg, edata)
8          if i % 1000 == 0:
9              f.value += 1
10             frame(writer, eimg)
11 plt.imshow(eimg);

```

IntProgress(value=0, max=1000)

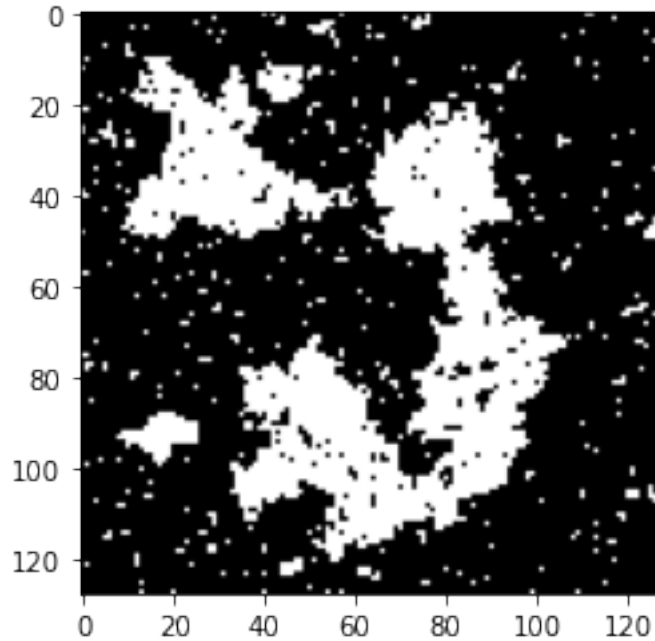


imovie.gif: Normal Ising.

```

1  n = 1000000
2  img = eimg
3  with imageio.get_writer('imovie.gif', mode='I') as writer:
4      frame(writer, img)
5      for i in range(n):
6          isingstep(0.5 * (np.pi / 2) / np.arctan(n - i), img)
7          if i % 1000 == 0:
8              frame(writer, img)
9  plt.imshow(img);

```



1.3 Image-metric Ising

```

1 # def takewrap(a, i, j, xs=np.arange(-1, 2), ys=np.arange(-1, 2)):
2 def takewrap(a, i, j, xs=np.arange(0, 1), ys=np.arange(0, 1)):
3     return np.array([x for v in a.take(xs+i, axis=0, mode='wrap')
4                       for x in v.take(ys+j, mode='wrap')])

```

1.3.1 Unrestricted swapping motion

Swapping preserves the intensity distribution.

```

1 def sienergy(img, init, i, j):
2     """Inversion-symmetric image energy"""
3     eq = takewrap(img, i, j) - takewrap(init, i, j)
4     return -np.abs(np.sum(2*eq - 1))
5
6 def ienergy(img, init, i, j):
7     """Image energy based on 3x3 block deviation"""
8     return np.abs(init[i, j] - img[i, j])
9
10 def swisingstep(β, img, edges):
11     w, h = np.shape(img)
12     i0 = np.random.randint(w)
13     i1 = np.random.randint(w)
14     j0 = np.random.randint(h)
15     j1 = np.random.randint(h)
16     E0 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
17     img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
18     E1 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)

```

```

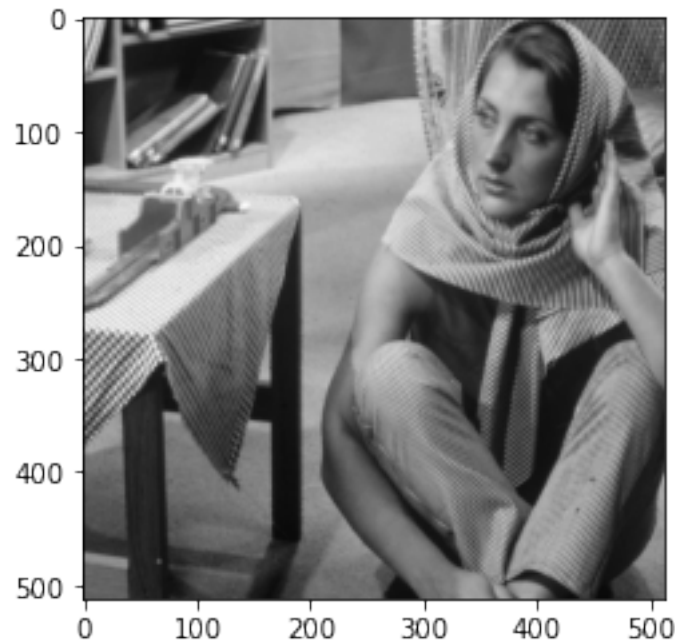
19     P = np.exp(-β*(E1 - E0)) if E1 > E0 else 1
20     if np.random.rand() > P: # Restore old
21         img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
22     return img
23
24 def nnisingstep(β, img, edges):
25     w, h = np.shape(img)
26     i0 = np.random.randint(w)
27     i1 = int((i0 + np.sign(np.random.rand() - 1/2)) % w)
28     j0 = np.random.randint(h)
29     j1 = int((j0 + np.sign(np.random.rand() - 1/2)) % h)
30     E0 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
31     img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
32     E1 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
33     P = np.exp(-β*(E1 - E0)) if E1 > E0 else 1
34     if np.random.rand() > P: # Restore old
35         img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
36     return img

```

```

1  img = Image.open("barbara.png")
2  eimg = -1 + 2 * (np.array(img) / 255)
3  initimg = eimg.copy()
4  plt.imshow(initimg);

```



swmovie.gif: Image metric Ising (arbitrary swaps with ienergy).

```

1  n = 2000000
2  f = IntProgress(min=0, max=(1 + (n-1) // 1000)) # instantiate the bar
3  display(f)
4  with imageio.get_writer('swmovie.gif', mode='I') as writer:

```

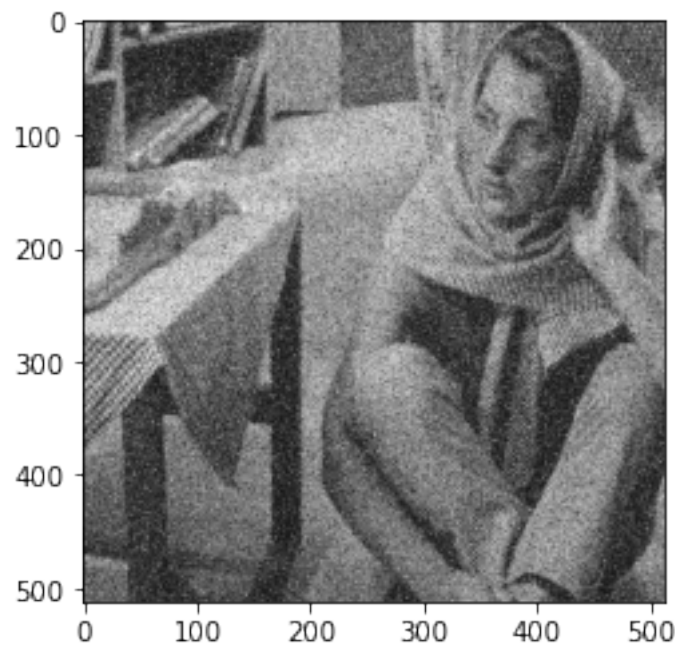


```

5     frame(writer, eimg)
6     for i in range(n):
7         k = i/n
8         swisingstep(3, eimg, initimg)
9         if i % 1000 == 0:
10             f.value += 1
11             frame(writer, eimg)
12     # for i in range(n):
13     #     k = i/n
14     #     swisingstep(4*(1 - k) + 1e-3*k, eimg, initimg)
15     #     if i % 1000 == 0:
16     #         f.value += 1
17     #         frame(writer, eimg)
18     # for i in range(n):
19     #     k = i/n
20     #     swisingstep(1e-3*(1 - k) + 4*k, eimg, initimg)
21     #     if i % 1000 == 0:
22     #         f.value += 1
23     #         frame(writer, eimg)
24
25 plt.imshow(eimg);

IntProgress(value=0, max=2000)

```

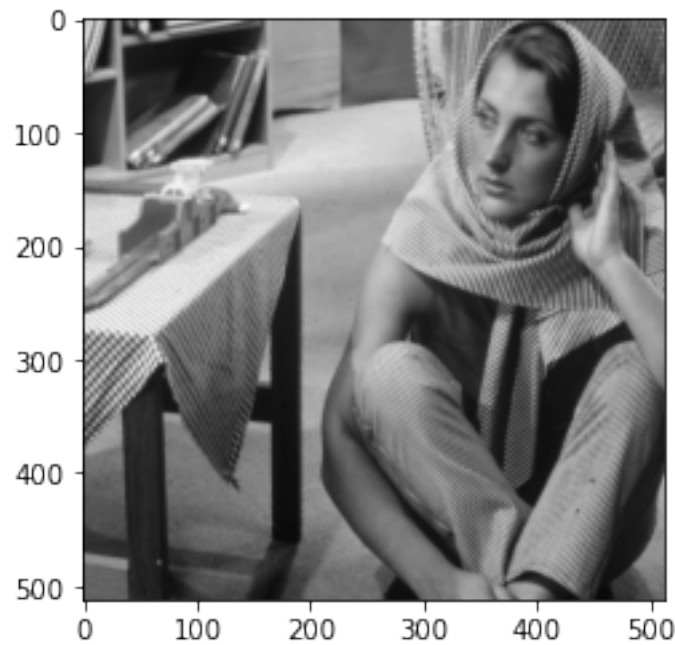


1.3.2 Nearest-neighbor swapping motion

```

1 img = Image.open("barbara.png")
2 eimg = -1 + 2 * (np.array(img) / 255)
3 initimg = eimg.copy()
4 plt.imshow(initimg);

```



nnmovie.gif: Image metric Ising (neighborly swaps with ienergy).

```

1  n = 2000000
2  f = IntProgress(min=0, max=3*(1 + (n-1) // 1000)) # instantiate the bar
3  display(f)
4  with imageio.get_writer('nnmovie.gif', mode='I') as writer:
5      frame(writer, eimg)
6      for i in range(n):
7          k = i/n
8          nnisingstep(5*(1 - k) + 1e-4*k, eimg, initimg)
9          if i % 1000 == 0:
10             f.value += 1
11             frame(writer, eimg)
12     for i in range(n):
13         nnisingstep(1e-4, eimg, initimg)
14         if i % 1000 == 0:
15             f.value += 1
16             frame(writer, eimg)
17     for i in range(n):
18         k = i/n
19         nnisingstep(1e-4*(1 - k) + 5*k, eimg, initimg)
20         if i % 1000 == 0:
21             f.value += 1
22             frame(writer, eimg)
23
24  plt.imshow(eimg);

```

IntProgress(value=0, max=6000)

