

1 Maximum-entropy reconstruction

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy import signal, misc
```

Given a measurement D of a system I_0 , we wish to reconstruct I_0 from D . We take the Bayesian approach and find the I that maximizes $P(I|D) \propto P(D|I)P(I)$, where the likelihood is related to the error in D due to I and the prior is $P(I) = \exp(\lambda S(I))$, where S is some notion of the entropy of an image. If $D = T(I_0)$, then $P(D|I) = \exp(-E(T(I), D))$, for some metric E on D 's. From the form of the objective $\ln P(D|I) + \ln(I) = -E(T(I), D) + \lambda S(I)$, we see that the entropy acts as a regularizer.

The astronomers have both I and D as intensity lists $\mathbb{Z}_N \rightarrow \mathbb{R}_{\geq 0}$, where

$$S(I) = \sum_i I_i \ln I_i$$

and

$$E(D', D) = \sum_i \exp\left(-\frac{(D'_i - D_i)^2}{2\sigma_i}\right)$$

(with empirical σ_i). The transformation T is convolution with the point spread function of the telescope.

To perform the optimization, we need not only the functions S and E , but also a procedure to modify candidate images.

```
1 def maxent_objective(D, I, λ): # For minimization
2     return D.E(I.transform()) - λ*I.S

1 # Greedy for now, but can be something like simulated annealing
2 def maxent(D, I, λ = 1, N = 1_000_000, ε = 1e-8):
3     f0 = np.inf
4     f = maxent_objective(D, I, λ)
5     i = 0
6     while ε < f0 - f and i < N:
7         I.propose()
8         fv = maxent_objective(D, I, λ)
9         if fv < f: # Greedy
10             f0, f = f, fv
11             I.accept()
12             i += 1
```

```

13         if i % (N // 20) == 0:
14             print("Maxent: {} / {}".format(i, N))
15
16     print("Maxent: i: {} / {}, Δf: {}".format(i, N, f0 - f))
17     return I, i, f

```

1.1 Example: 1D Point from Gaussian

```

1 class Gaussian:
2     def __init__(self, μ=0, σ=1):
3         self.μ = μ
4         self.σ = σ
5     def E(self, Gv):
6         return (self.μ - Gv.μ)**2 + (self.σ - Gv.σ)**2
7
8 class Point:
9     def __init__(self, x=0):
10        self.x = x
11        self.dx = 0
12        self.S = self.entropy()
13    def propose(self):
14        self.dx = np.random.rand() - 0.5
15        self.S = self.entropy()
16    def accept(self):
17        self.x += self.dx
18        self.dx = 0 # Idempotence
19    def entropy(self):
20        return -(self.x - 10)**2 # Opposite from true max
21    def transform(self):
22        return Gaussian(μ = self.x + self.dx)

```

```

1 D = Gaussian(0, 1)
2 I = Point(9)
3 I0, _, _ = maxent(D, I, λ = 1, N = 1_000_000, ε = 1e-4)
4 I0.x

```

Maxent: i: 1000000 / 1000000, Δf: 0.011299906795997572

4.268883578482481

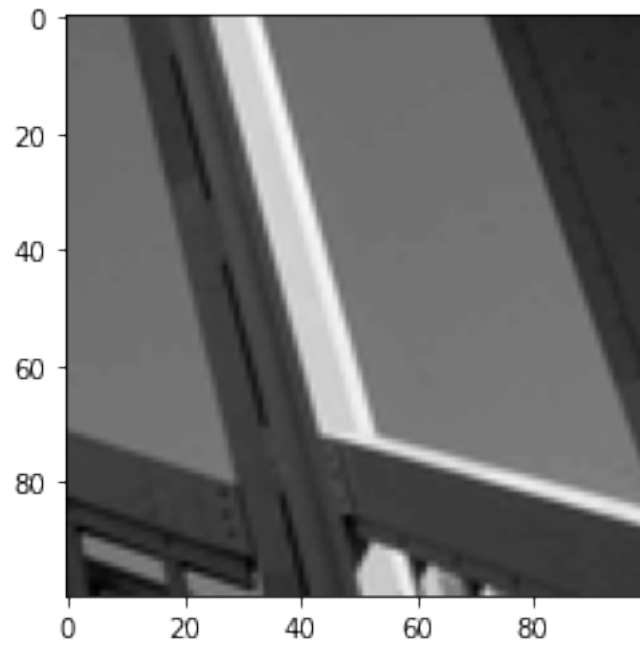
Results are sort of near 5. The optimization is terrible, but you get the idea: the entropy shifts the best point away from zero.

1.2 Example: Image from PSF convolution (measurement)

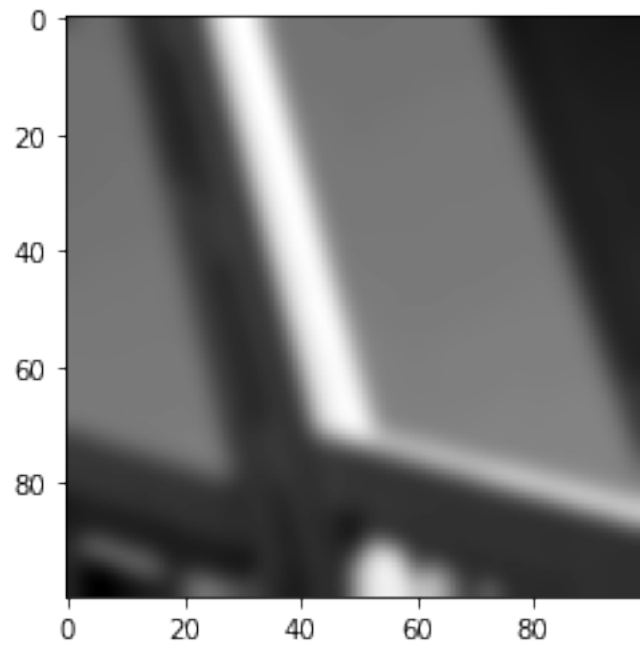
```

1 class DImage:
2     def __init__(self, I):
3         self.I = I
4     def E(self, Dv):
5         return np.sum((self.I - Dv.I)**2)
6
7 class IImage:
8     def __init__(self, I):
9         self.I = I
10        self.w, self.h = np.shape(I)
11        self.i, self.j = 0, 0
12        self.I0 = 0
13        self.Iv = 0
14        n = int(np.sqrt(self.w * self.h))
15        self.G = signal.windows.gaussian(n // 10, n // 50)
16        self.S = self.entropy()
17    def propose(self):
18        self.i, self.j = np.random.randint(self.w), np.random.randint(self.h)
19        I0 = self.I[self.i, self.j]
20        self.I0 = I0
21        self.Iv = np.random.randint(256)
22        self.S = self.entropy()
23    def accept(self):
24        self.I[self.i, self.j] = self.Iv
25    def entropy(self):
26        return -np.sum(np.log(self.I + 1)) + self.I0*np.log(self.I0 + 1) -
27        ↪ self.Iv*np.log(self.Iv + 1)
28    def transform(self):
29        Iv = self.I.copy()
30        Iv[self.i, self.j] = self.Iv
31        return DImage(signal.sepfir2d(Iv, self.G, self.G))
32
33 I = IImage(misc.ascent()[250:350,250:350])
34 plt.imshow(I.I, cmap='gray');

```



```
1 plt.imshow(I.transform().I, cmap='gray');
```



```

1 I = IImage(misc.ascent()[250:350,250:350])
2 Iguess = IImage(128 * np.ones((I.w, I.h), dtype=int))
3 I0, _, _ = maxent(I.transform(), Iguess,  $\lambda = 1e-9$ , N = 1_000_000,  $\epsilon = 1e-20$ ) # Just do the max
    ↪ iterations

Maxent: 50000 / 1000000
Maxent: 100000 / 1000000
Maxent: 150000 / 1000000
Maxent: 200000 / 1000000
Maxent: 250000 / 1000000
Maxent: 300000 / 1000000
Maxent: 350000 / 1000000
Maxent: 400000 / 1000000
Maxent: 450000 / 1000000
Maxent: 500000 / 1000000
Maxent: 550000 / 1000000
Maxent: 600000 / 1000000
Maxent: 650000 / 1000000
Maxent: 700000 / 1000000
Maxent: 750000 / 1000000
Maxent: 800000 / 1000000
Maxent: 850000 / 1000000
Maxent: 900000 / 1000000
Maxent: 950000 / 1000000
Maxent: 1000000 / 1000000
Maxent: i: 1000000 / 1000000,  $\Delta f$ : 4.847227362683043

1 plt.imshow(I0.I, cmap='gray');

```

