Orthogonal Matching Pursuit

Noiseless case

Fig.1 shows the normalized errors and the probabilities of exact support recovery. The parameters of *smax* are 20, 30 and 40 for N = 20, 50, and 100, respectively. The parameters of *Mmax* are 20, 30, and 40 for N = 20, 50, and 100, respectively. All figures have sharp phase transitions. The critical lines are monotonic convex-upward curves. When s is small and M is large, OMP succeeds. The curves also depend on N.

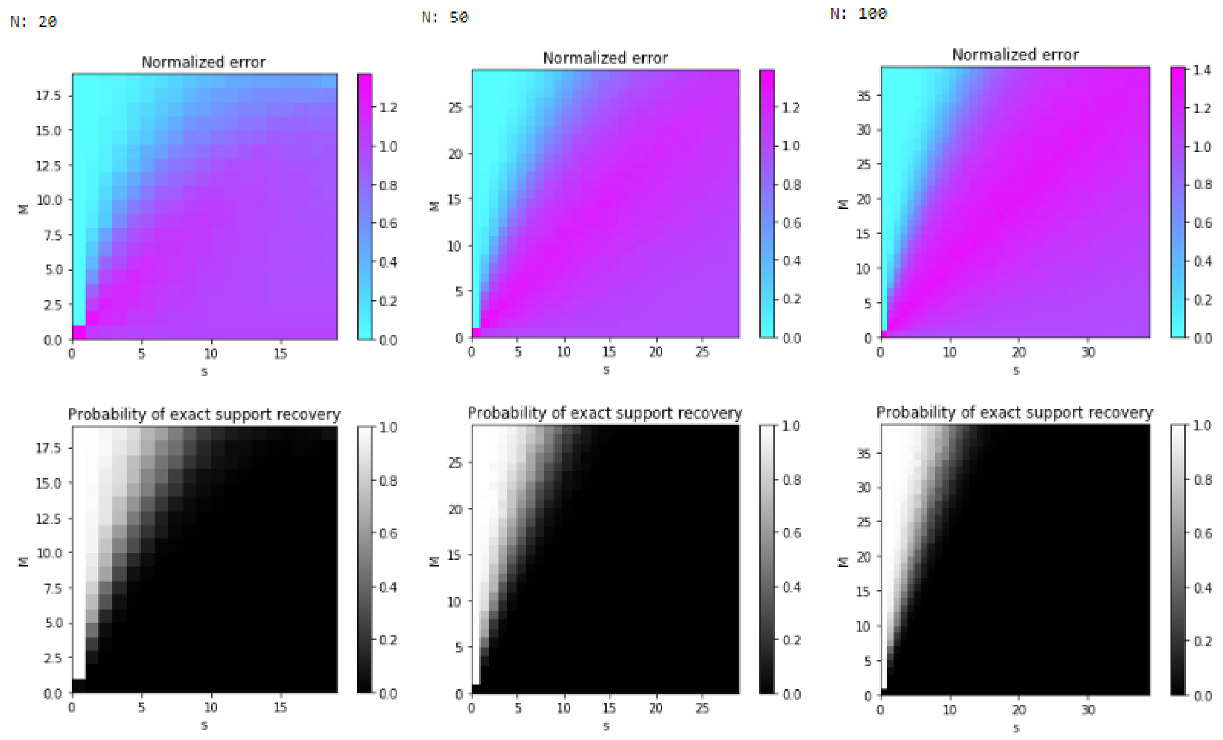


Fig.1 The normalized errors and the probabilities of exact support recovery (N=20, 50, 100)

Noisy case 1

Fig.2 shows the normalized errors and the probabilities of success in noisy cases. Success is defined as the event that the normalized error is less than 0.001. In the case of small noise (sigma=0.1), OMP succeeds when s is small and M is large. On the other hand, in the case of large noise (sigma=3.0), it fails with rare exceptions. This is because it fails in choosing n_i in OMP algorithm. When noise is large, it is more likely to choose wrong n_i because of the noise. In this case, OMP tries to provide x_0 s.t. $Ax_0=Ax+n$.

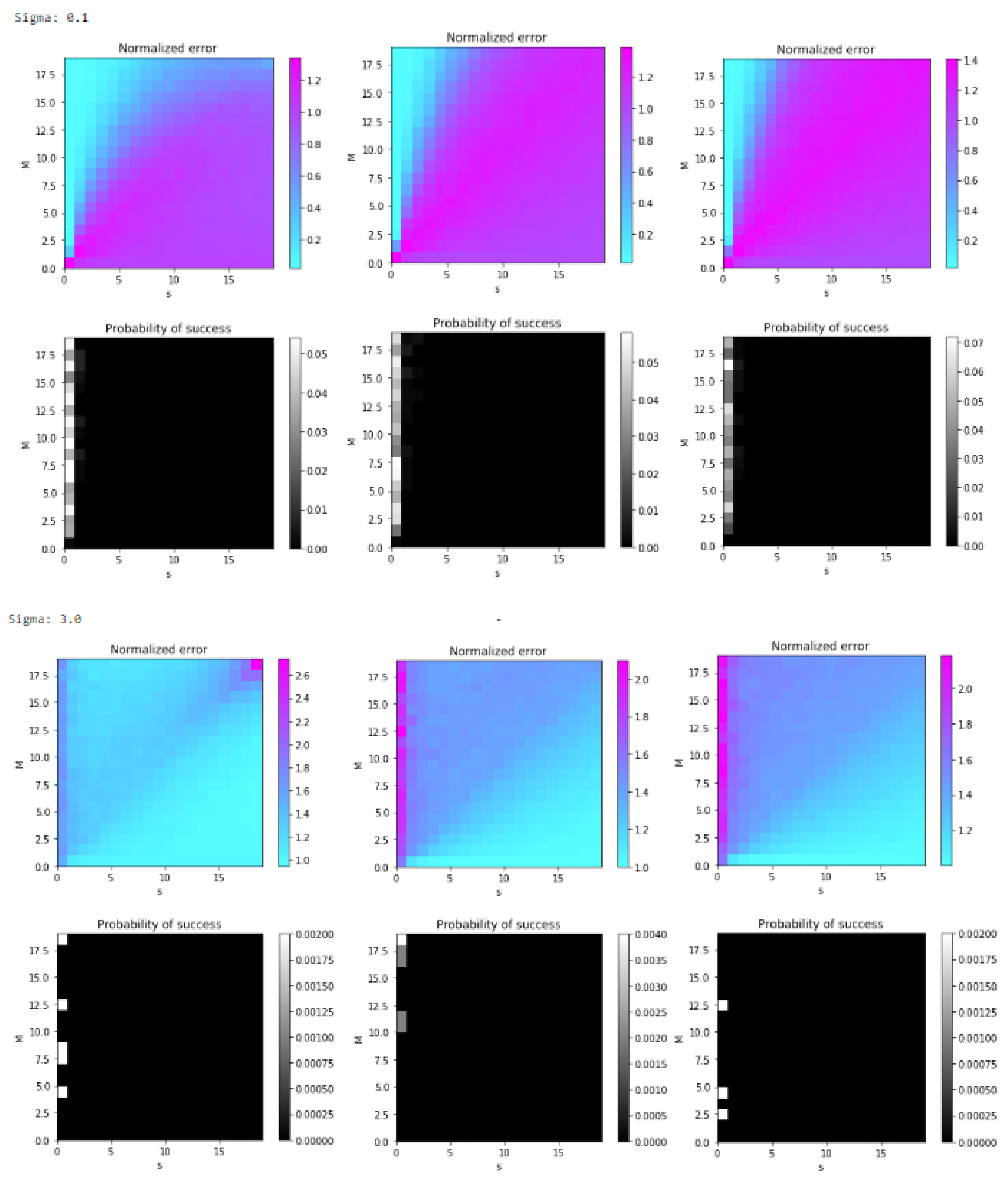


Fig.2 The normalized errors and probabilities of success in noisy cases (sigma=0.1, 3.0, N=20, 50, 100)

Noisy case 2 (unknown sparsity)

Fig.3 shows the normalized errors and the probabilities of success in noisy cases. In these simulations, the loop process in OMP is stopped when $|y-Ax|$ is smaller than the norm of *n*. Success is defined as the event that the normalized error is less than 0.001. The result is almost the same as that of the previous problem. However, this algorithm is faster than the previous one. This is because it removes useless computations. Since OMP can fail in choosing proper n_i, the following steps can be useless. By stopping the loop process in OMP when $|y-Ax|$ is small enough considering the intensity of noise, it removes unproductive process. These results indicate the algorithm does not need to know the sparsity.

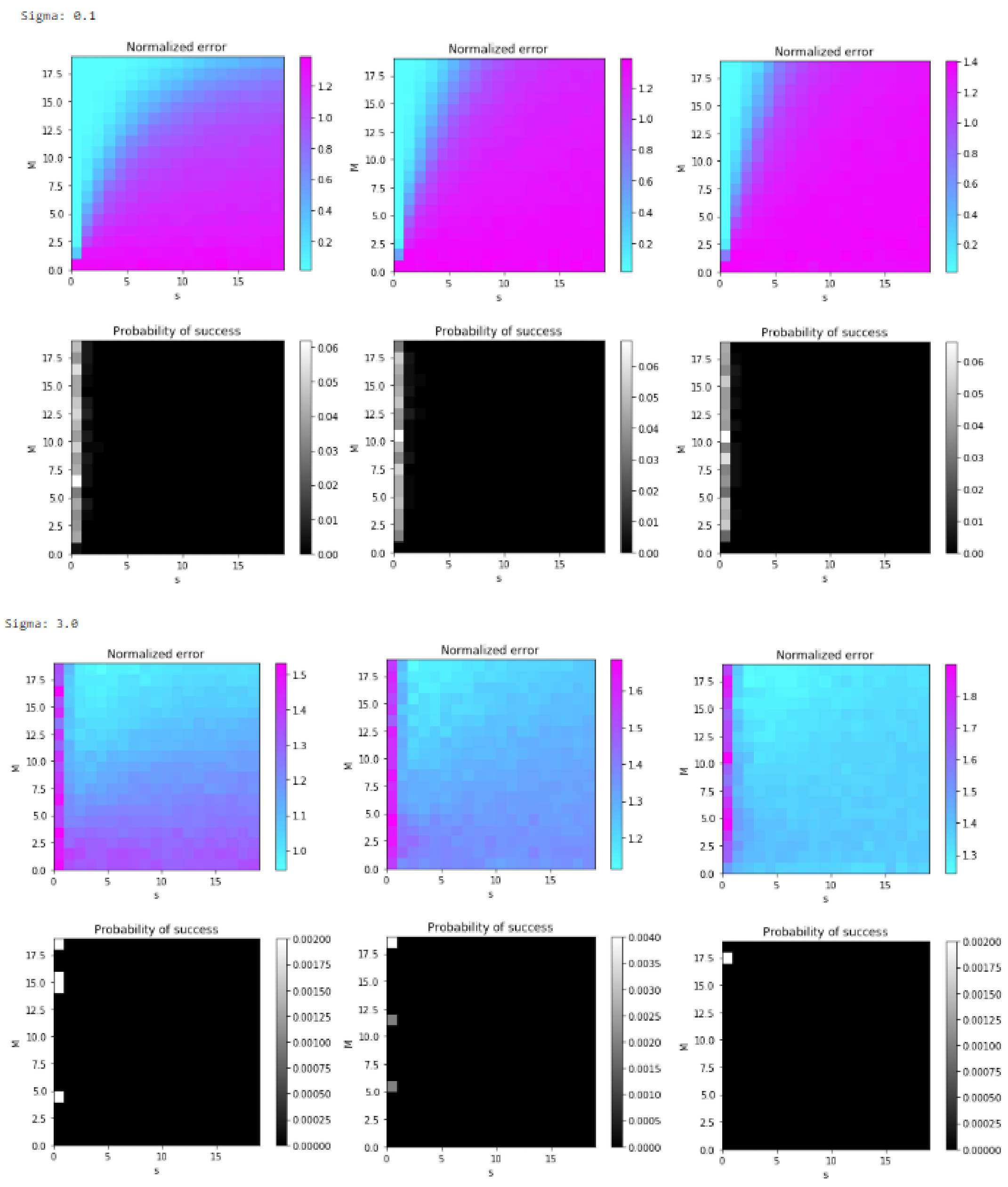


Fig.3 The normalized errors and probabilities of success in noisy cases (sigma=0.1, 3.0, N=20, 50, 100)

Experiments on a real image

The experiments are conducted following this approach:

Approach

1. Read the original image and reshape it to 100x100 gray-scale data
2. Transform it by Discrete Cosine Transformation
3. Make it sparse by picking the s significant(largest in absolute value) coefficients
4. Compress the sparse array with a random matrix
5. Decompress the data with the OMP method.
6. Output the data transformed by Inverse Discrete Cosine Transformation

Fig.4 shows the original image and the sparse-data image. The original image is slightly blurred due to rescaling process. The sparse-data image is created from the *s* significant coefficients of discreet cosine transformation directly by inverse discreet cosine transformation without compression/decompression process. Therefore, the ideal compression/decompression should reconstruct the identical image of the sparse-data image.

Fig.5 shows the reconstructed images for M/N = 0.1, 0.2, 0.4, 0.6, and 0.8. In the experiments, N = 10000, and s = 1250. The sparsity is 12.5%.

As stated in the previous comments, the smaller M is, the more likely orthogonal matching pursuit is to succeed. When M/N >= 0.4, the shape of the dinosaur is recognizable, but When M/N is 0.2 or 0.1, the shape of the dinosaur is too blur to be recognized. As we saw in the previous simulations, the threshold is sharp. This means the algorithm works as well when M/N = 0.4 as when M/N = 0.8. The maximum compression M/N is 0.4 (the maximum compression rate N/M = 2.5).

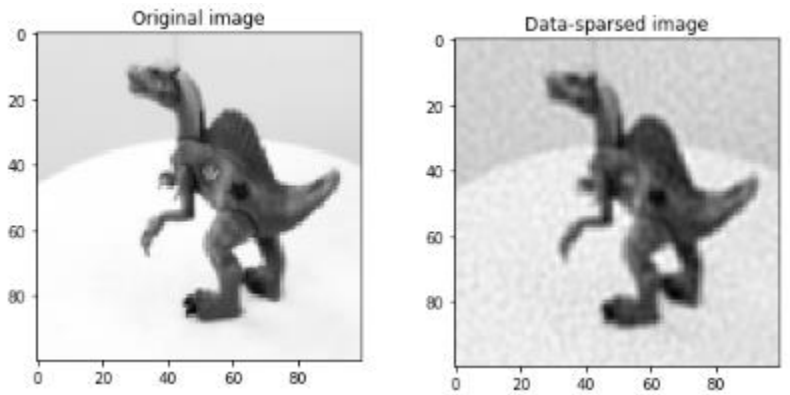
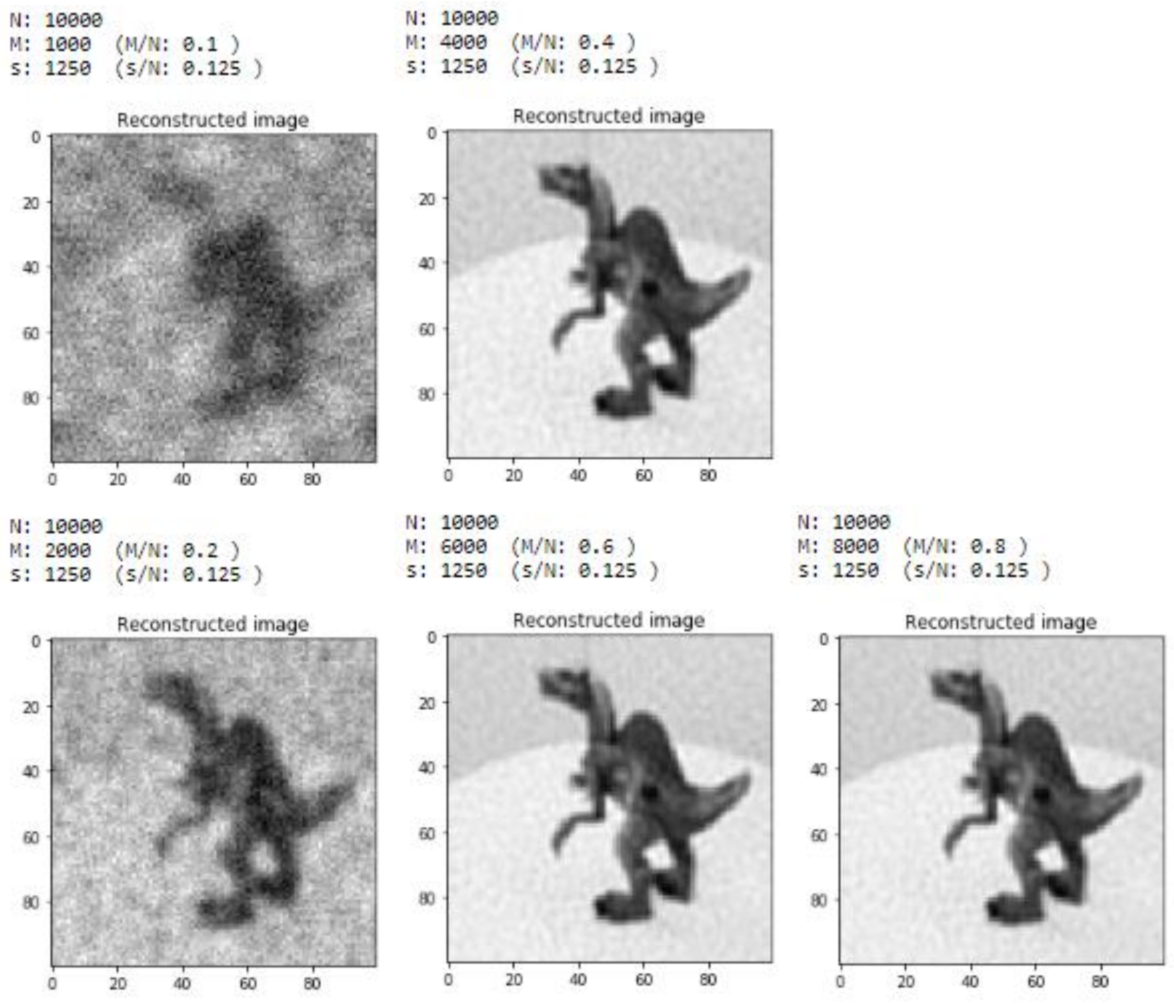


Fig.4 The original image (left) and the sparse-data image (right). The ideal compression/decompression should reconstruct the identical image of the sparse-data image.



```
N: 10000
M: 1000  (M/N: 0.1 )
s: 1250  (s/N: 0.125 )
```

```
N: 10000
M: 4000  (M/N: 0.4 )
s: 1250  (s/N: 0.125 )
```

```
N: 10000
M: 2000  (M/N: 0.2 )
s: 1250  (s/N: 0.125 )
```

```
N: 10000
M: 6000  (M/N: 0.6 )
s: 1250  (s/N: 0.125 )
```

```
N: 10000
M: 8000  (M/N: 0.8 )
s: 1250  (s/N: 0.125 )
```

Fig.5 The reconstructed images for M/N = 0.1, 0.2, 0.4, 0.6, and 0.8

Appendix

Fig.6 shows the source codes of orthogonal matching pursuit used in the experiments.

```
def orthogonal_matching_pursuit(y, A, s):
    N = A.shape[1]
    ri = y
    S = []
    for i in range(s):
        tmp = ri.T * A
        ni = abs(tmp).argmax()
        S.append(ni)
        a = np.linalg.pinv(A[:, S]) * y
        ri = y - A[:, S] * a
    x_hat = np.matrix(np.zeros([N, 1]))
    x_hat[S] = a
    return x_hat
```

```
def orthogonal_matching_pursuit_unknownsparsity(y, A, norm_n):
    N = A.shape[1]
    ri = y
    S = []
    x_hat = np.matrix(np.zeros([N, 1]))
    for _ in range(N):
        if (np.linalg.norm(y - A*x_hat) <= norm_n):
            break
        tmp = ri.T * A
        ni = abs(tmp).argmax()
        S.append(ni)
        a = np.linalg.pinv(A[:, S]) * y
        ri = y - A[:, S] * a
        x_hat[S] = a
    return x_hat
```

Fig.6 The source codes of orthogonal matching pursuit