法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

Python库

邹博

本次说明

- 本PPT后面仅列举使用Python库的效果截图，详细内容请参考该PPT的配套代码。

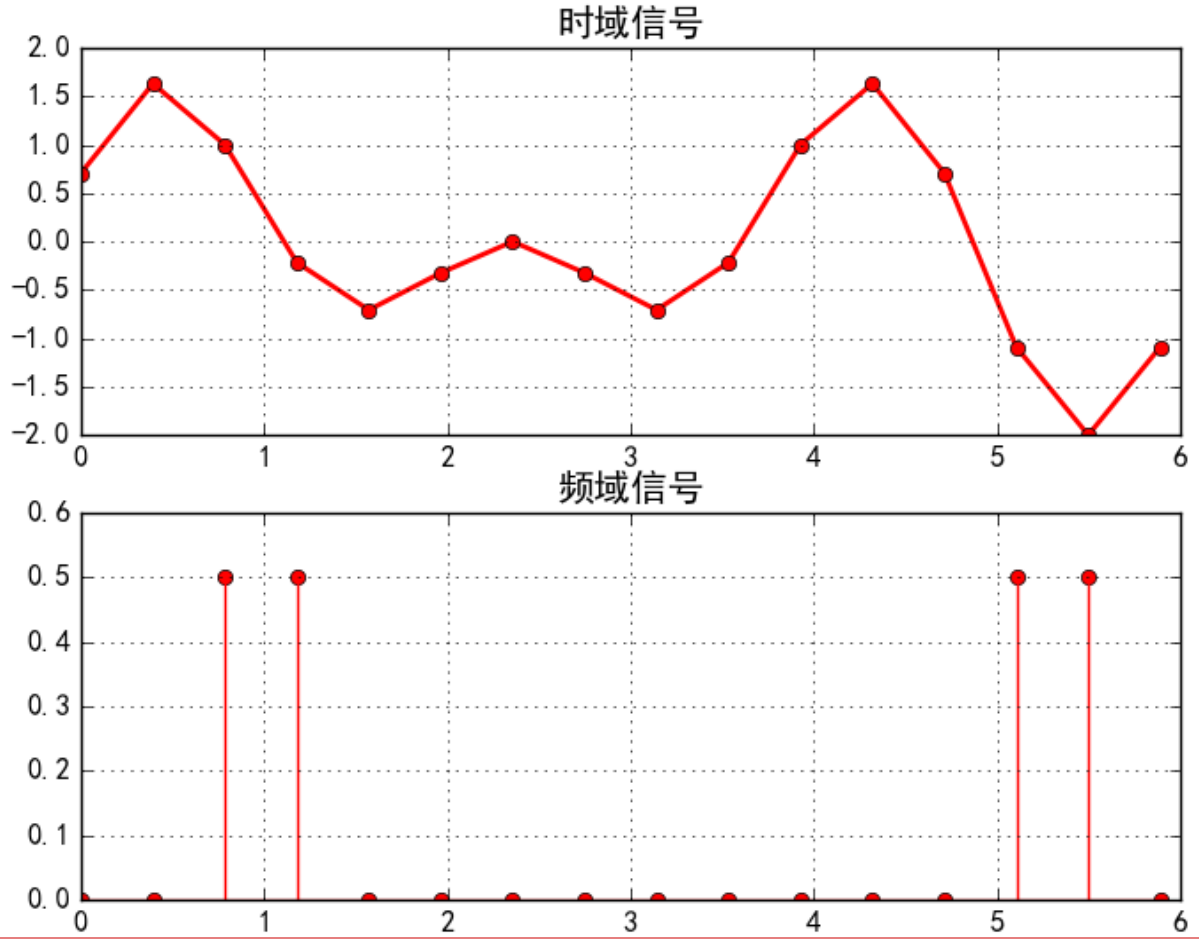
数值计算

- 对于某二分类问题，若构造了10个正确率都是0.6的分类器，采用少数服从多数的原则进行最终分类，则最终分类正确率是多少？
  - 若构造100个分类器呢？

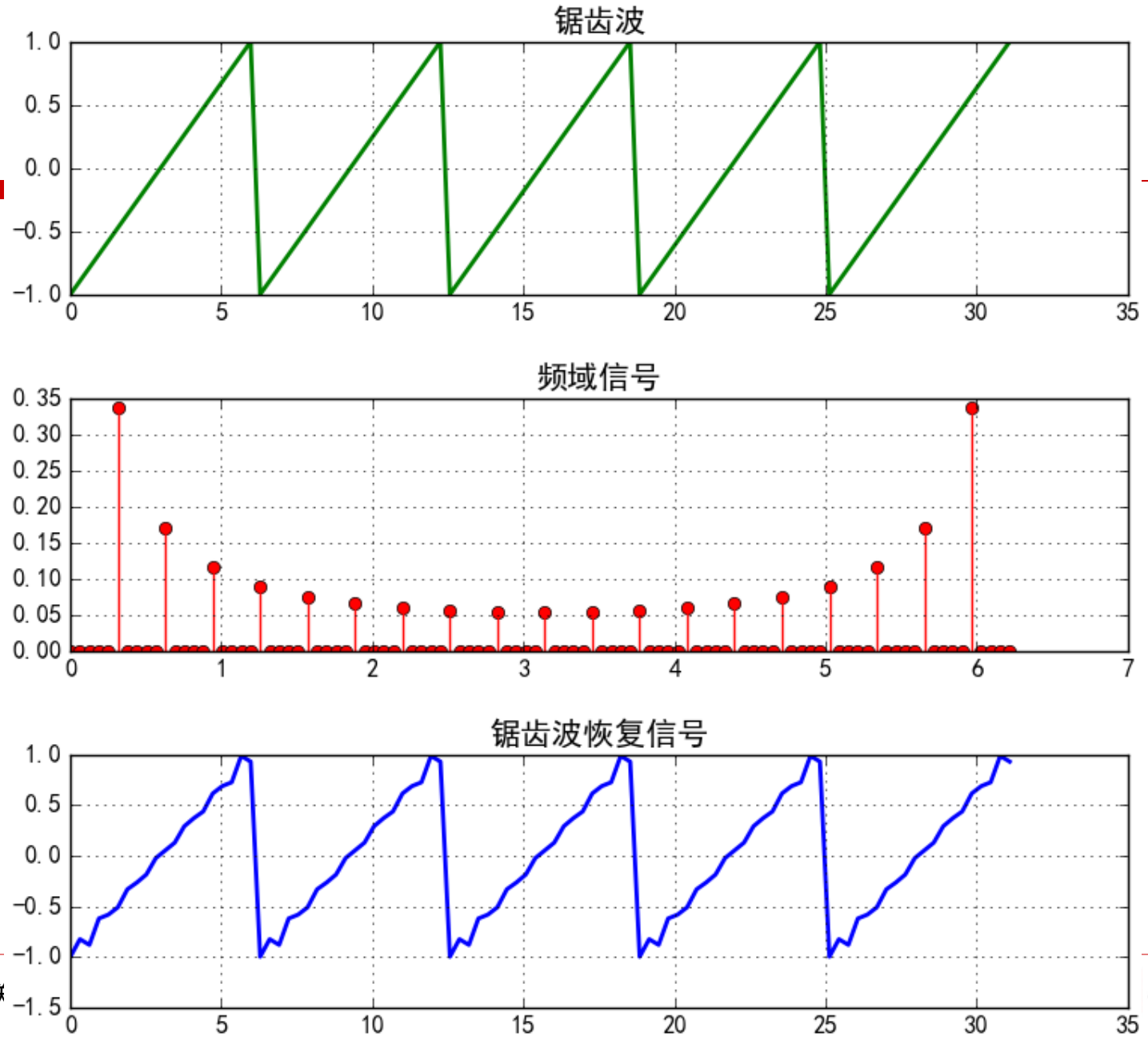
```
def bagging(n, p):
    s = 0
    for i in range(n / 2 + 1, n + 1):
        s += c(n, i) * p ** i * (1 - p) ** (n - i)
    return s


if __name__ == "__main__":
    for t in range(10, 101, 10):
        print t, '次采样正确率：', bagging(t, 0.6)
```

```
Ensumble
C:\Python27\python.exe D:/Python/ML/6.Package/6.1.Ensumble.py
10 次采样正确率： 0.6331032576
20 次采样正确率： 0.755337203316
30 次采样正确率： 0.824630946493
40 次采样正确率： 0.870234294178
50 次采样正确率： 0.902192635847
60 次采样正确率： 0.925376305649
70 次采样正确率： 0.942565538515
80 次采样正确率： 0.955502944118
90 次采样正确率： 0.965347339325
100 次采样正确率： 0.972900802243
```

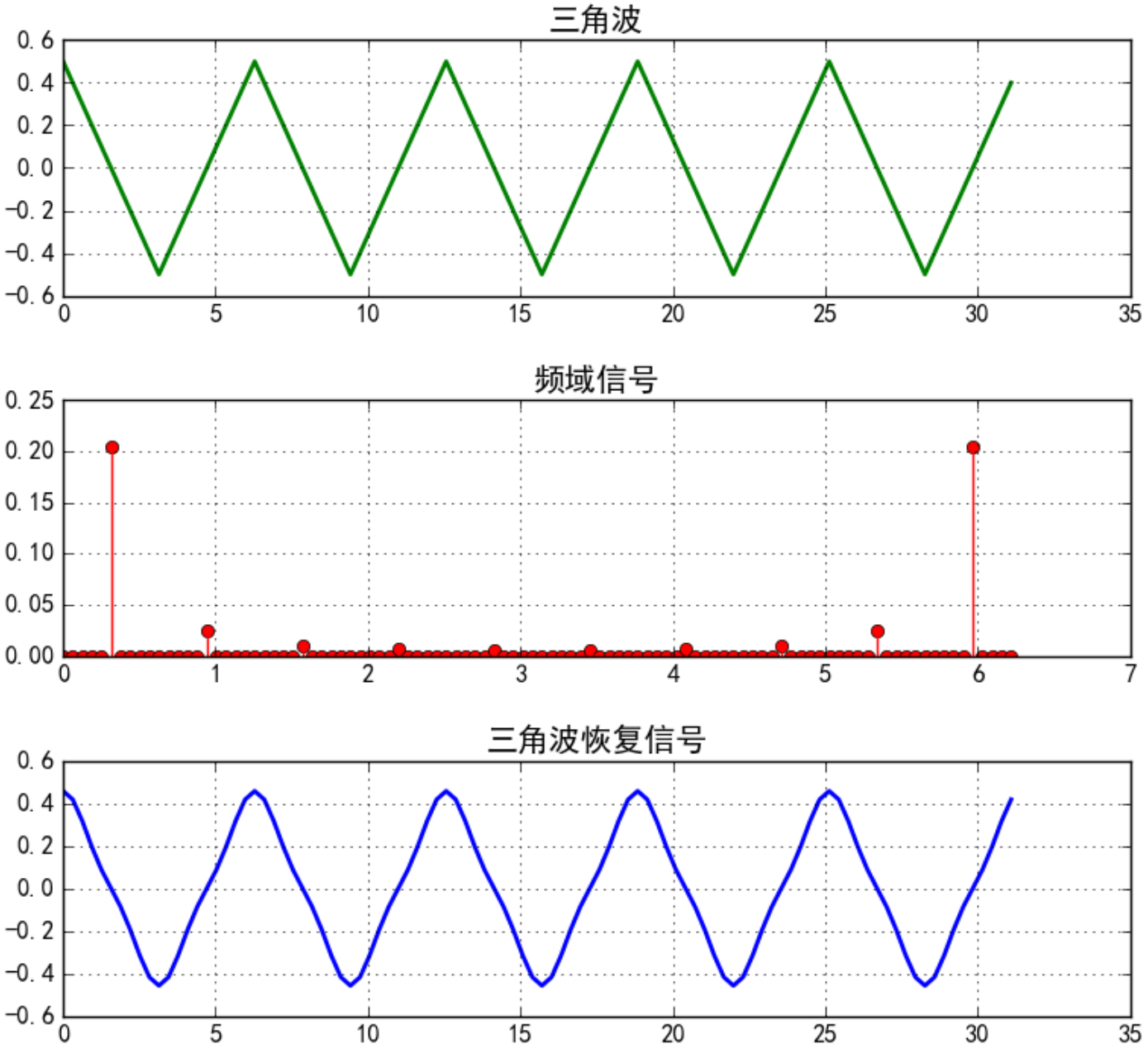
时域与频域信号


时域信号
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5
6
频域信号
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
6

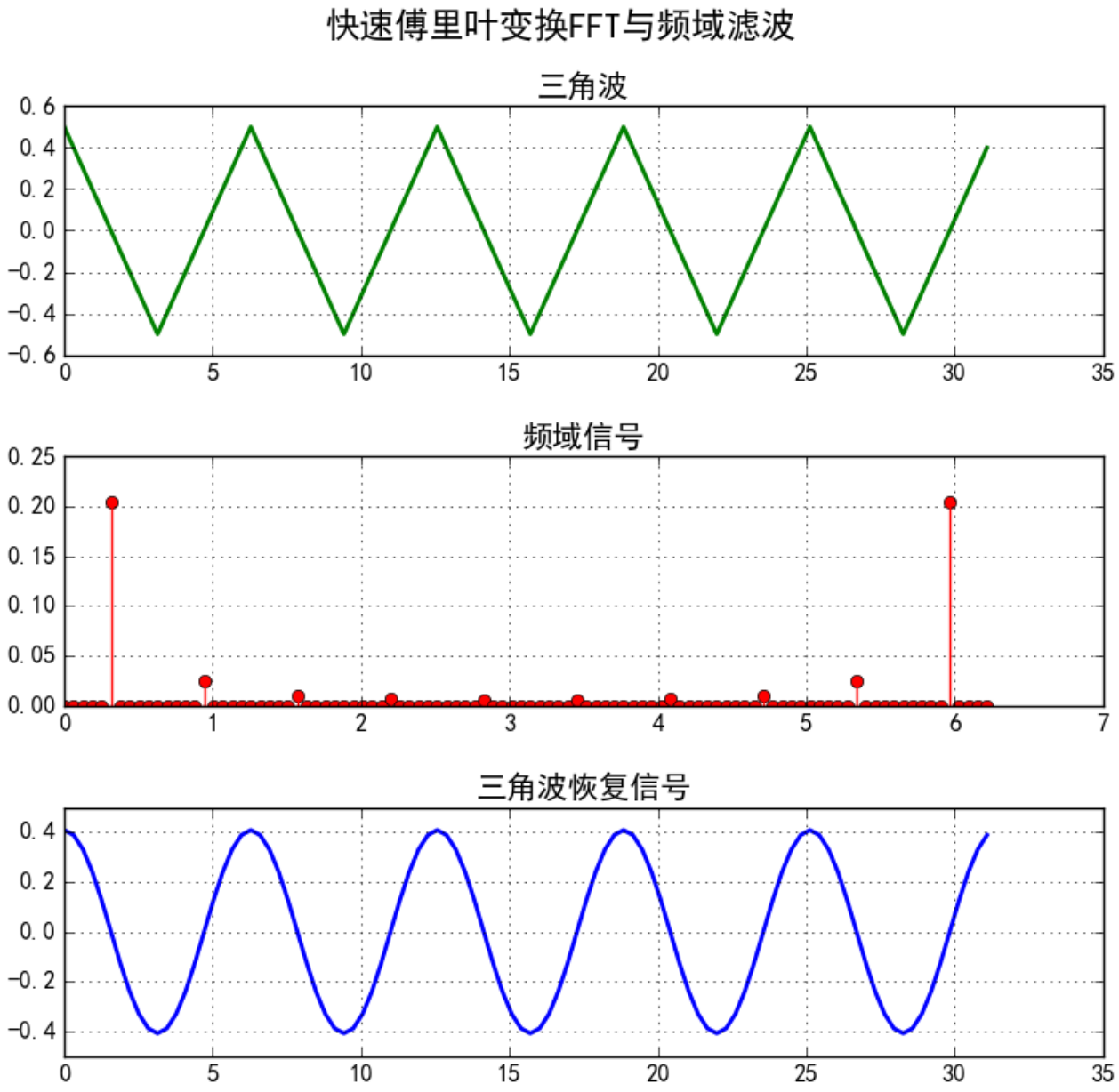
快速傅里叶变换FFT与频域滤波



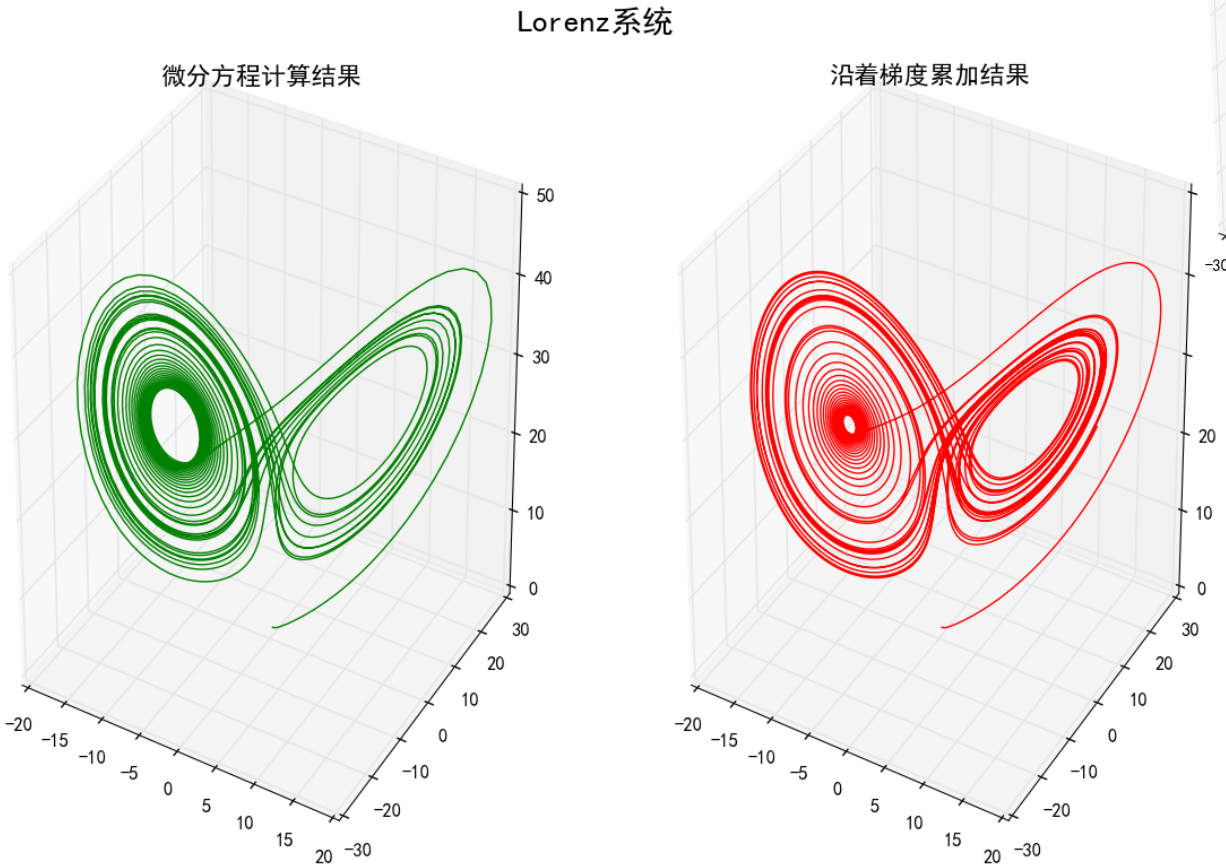
快速傅里叶变换FFT与频域滤波

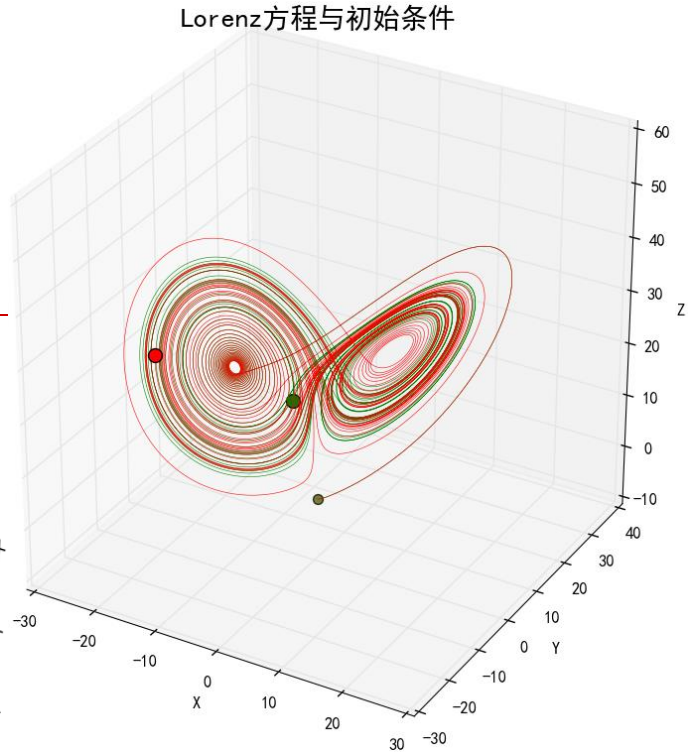


不同的阈值


快速傅里叶变换FFT与频域滤波
三角波
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
5
10
15
20
25
30
35
频域信号
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
6
7
三角波恢复信号
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20
25
30
35

常微分方程





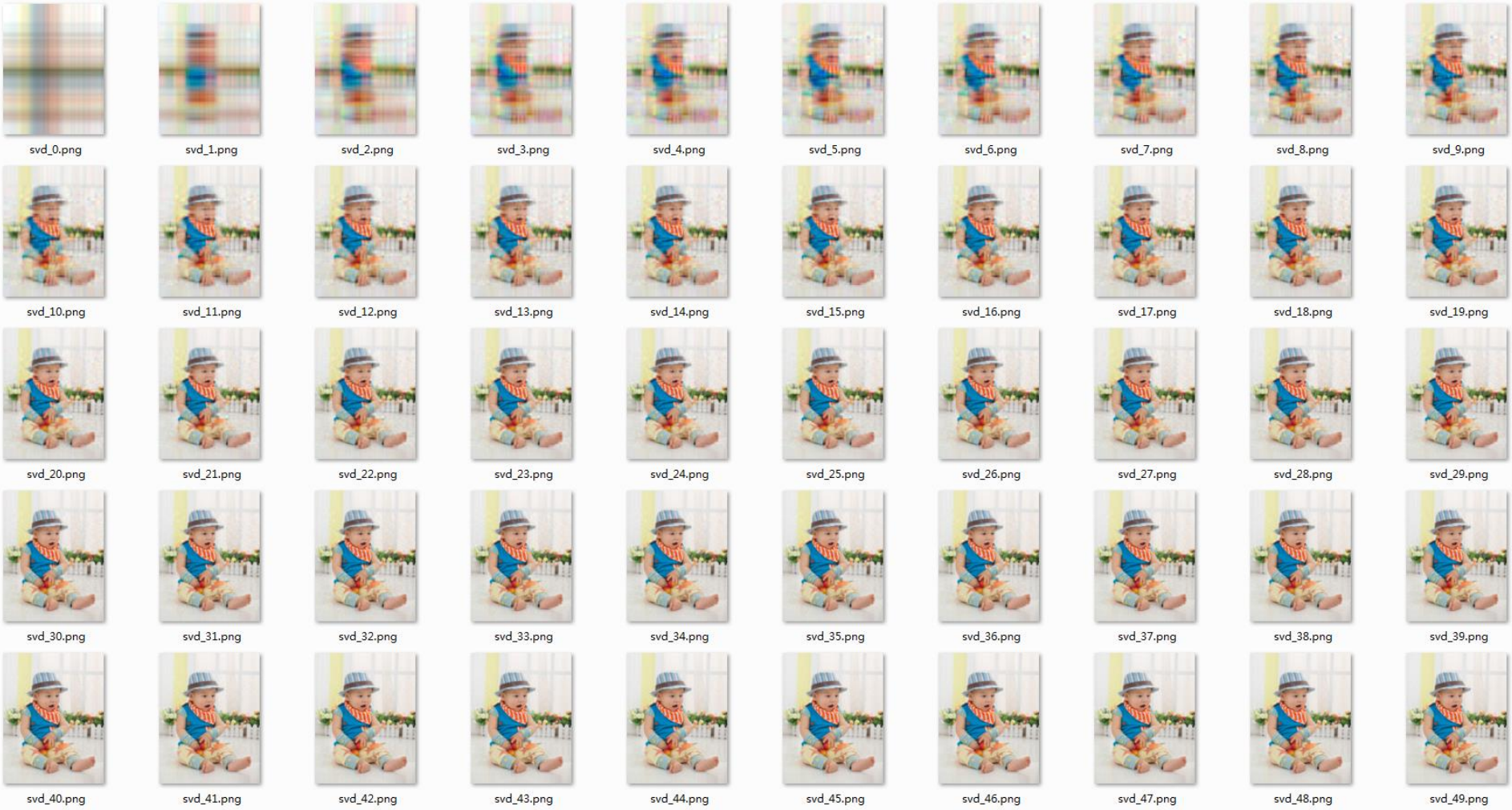
Code

```
s0 = (0., 1., 0.)
t = np.arange(0, 30, 0.01)
s = odeint(lorenz, s0, t)
plt.figure(figsize=(12, 8), facecolor='w')
plt.subplot(121, projection='3d')
plt.plot(s[:, 0], s[:, 1], s[:, 2], c='g')
plt.title(u'微分方程计算结果', fontsize=16)

s = lorenz_trajectory(s0, 40000)
plt.subplot(122, projection='3d')
plt.plot(s[:, 0], s[:, 1], s[:, 2], c='r')
plt.title(u'沿着梯度累加结果', fontsize=16)

plt.tight_layout(1, rect=(0,0,1,0.98))
plt.suptitle(u'Lorenz系统', fontsize=20)
plt.show()

ax = Axes3D(plt.figure(figsize=(8, 8)))
s0 = (0., 1., 0.)
s1 = lorenz_trajectory(s0, 50000)
s0 = (0., 1.0001, 0.)
s2 = lorenz_trajectory(s0, 50000)
# 曲线
ax.plot(s1[:, 0], s1[:, 1], s1[:, 2], c='g', lw=0.4)
ax.plot(s2[:, 0], s2[:, 1], s2[:, 2], c='r', lw=0.4)
# 起点
ax.scatter(s1[0, 0], s1[0, 1], s1[0, 2], c='g', s=50, alpha=0.5)
ax.scatter(s2[0, 0], s2[0, 1], s2[0, 2], c='r', s=50, alpha=0.5)
# 终点
ax.scatter(s1[-1, 0], s1[-1, 1], s1[-1, 2], c='g', s=100)
ax.scatter(s2[-1, 0], s2[-1, 1], s2[-1, 2], c='r', s=100)
ax.set_title(u'Lorenz方程与初始条件', fontsize=20)
ax.set_xlabel(u'X')
ax.set_ylabel(u'Y')
ax.set_zlabel(u'Z')
plt.show()
```

奇异值分解-效果

```
def restore(sigma, u, v, K): # 奇异值、左特征向量、右特征向量
    print K
    m = len(u)
    n = len(v[0])
    a = np.zeros((m, n))
    for k in range(K+1):
        for i in range(m):
            a[i] += sigma[k] * u[i][k] * v[k]
    b = a.astype('uint8')
    Image.fromarray(b).save("svd_" + str(K) + ".png")
```

SVD


svd_0.png
svd_1.png
svd_2.png
svd_3.png
svd_4.png
svd_5.png
svd_6.png
svd_7.png
svd_8.png
svd_9.png
svd_10.png
svd_11.png
svd_12.png
svd_13.png
svd_14.png
svd_15.png
svd_16.png
svd_17.png
svd_18.png
svd_19.png
svd_20.png
svd_21.png
svd_22.png
svd_23.png
svd_24.png
svd_25.png
svd_26.png
svd_27.png
svd_28.png
svd_29.png
svd_30.png
svd_31.png
svd_32.png
svd_33.png
svd_34.png
svd_35.png
svd_36.png
svd_37.png
svd_38.png
svd_39.png
svd_40.png
svd_41.png
svd_42.png
svd_43.png
svd_44.png
svd_45.png
svd_46.png
svd_47.png
svd_48.png
svd_49.png

图像的卷积

laplacian.png

laplacian2.png

prewitt.png

prewitt_x.png

prewitt_y.png

soble.png

soble_x.png

soble_y.png

图像的卷积

laplacian.png

laplacian2.png

prewitt.png

prewitt_x.png

prewitt_y.png

soble.png

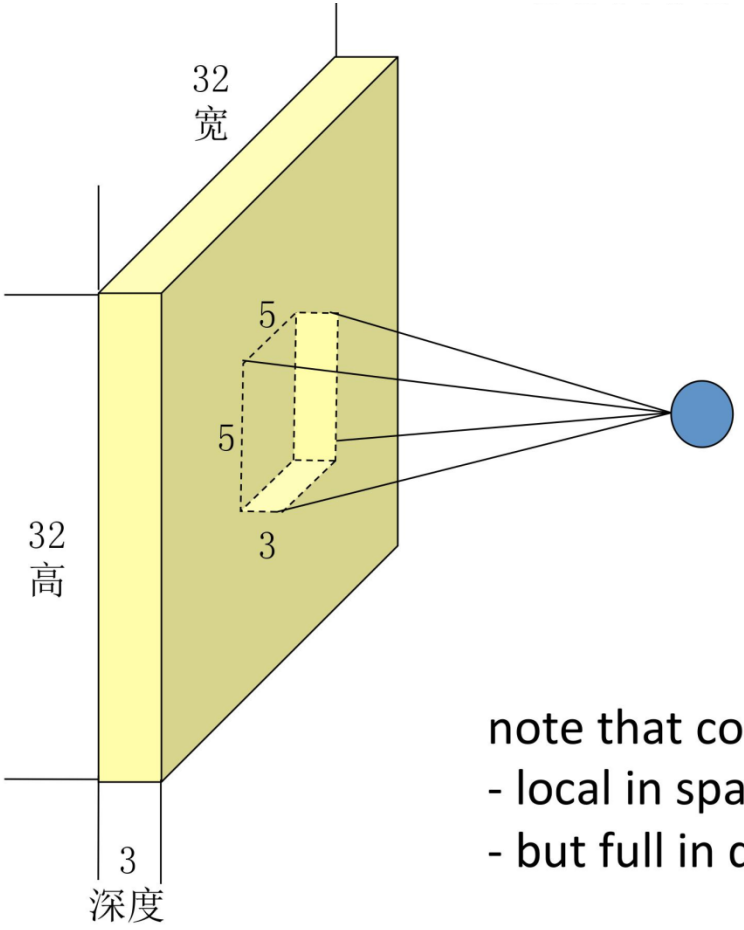
soble_x.png

soble_y.png

Code

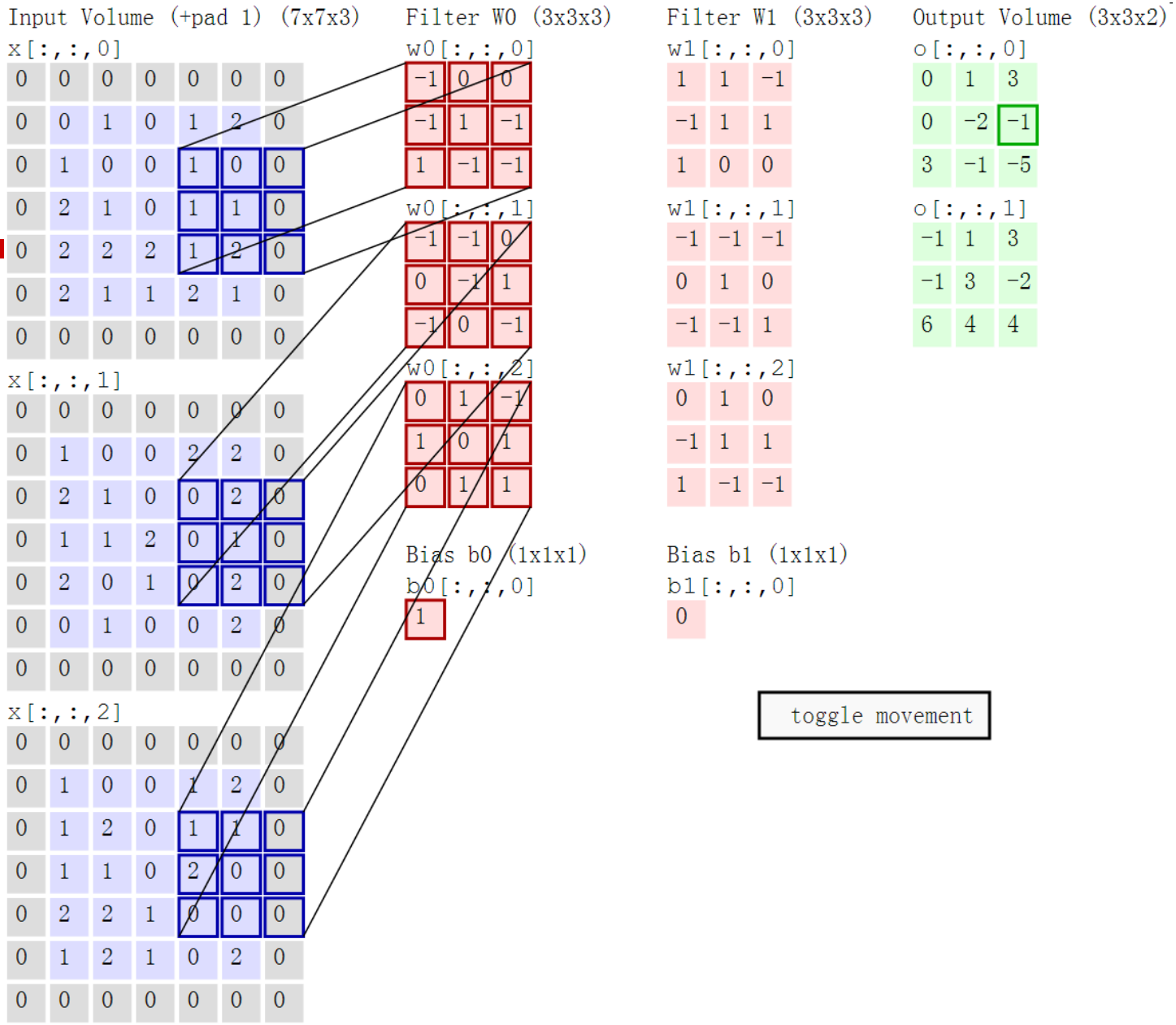
```
def convolve(image, weight):
    height, width = image.shape
    h, w = weight.shape
    height_new = height - h + 1
    width_new = width - w + 1
    image_new = np.zeros((height_new, width_new), dtype=np.float)
    for i in range(height_new):
        for j in range(width_new):
            image_new[i,j] = np.sum(image[i:i+h, j:j+w] * weight)
    image_new = image_new.clip(0, 255)
    image_new = np.rint(image_new).astype('uint8')
    return image_new
```

卷积网络

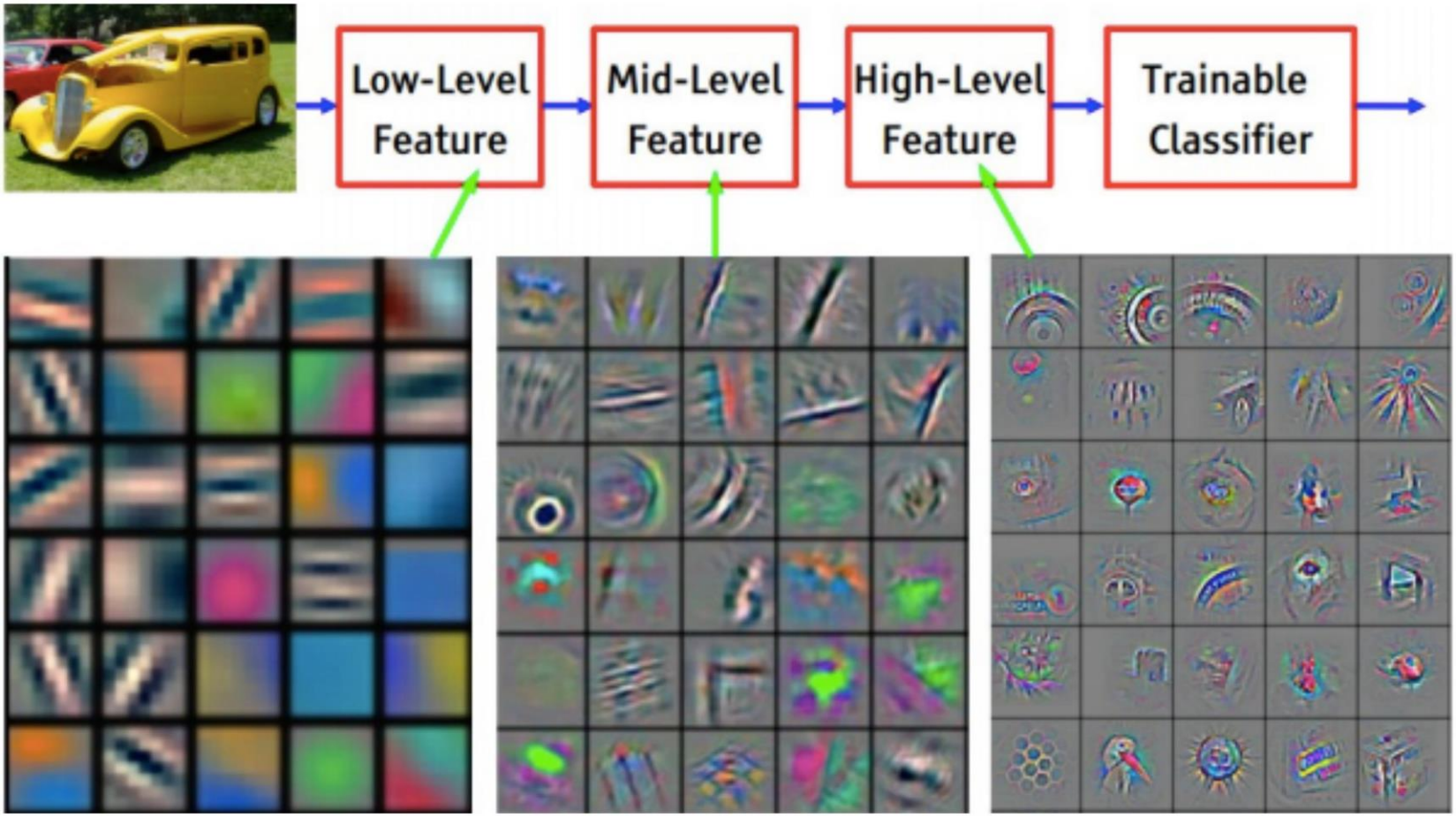


note that connectivity is:
- local in space (5x5 inside 32x32)
- but full in depth (all 3 depth channels)

卷积


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 1 0 1 2 0
0 1 0 0 1 0 0
0 2 1 0 1 1 0
0 2 2 2 1 2 0
0 2 1 1 2 1 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 2 2 0
0 2 1 0 0 2 0
0 1 1 2 0 1 0
0 2 0 1 0 2 0
0 0 1 0 0 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 1 0 0 1 2 0
0 1 2 0 1 1 0
0 1 1 0 2 0 0
0 2 2 1 0 0 0
0 1 2 1 0 2 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 0
-1 1 -1
1 -1 -1
w0[:,:,1]
-1 -1 0
0 -1 1
-1 0 -1
w0[:,:,2]
0 1 -1
1 0 1
0 1 1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
1 1 -1
-1 1 1
1 0 0
w1[:,:,1]
-1 -1 -1
0 1 0
-1 -1 1
w1[:,:,2]
0 1 0
-1 1 1
1 -1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
0 1 3
0 -2 -1
3 -1 -5
o[:,:,1]
-1 1 3
-1 3 -2
6 4 4
toggle movement

深度网络


Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier

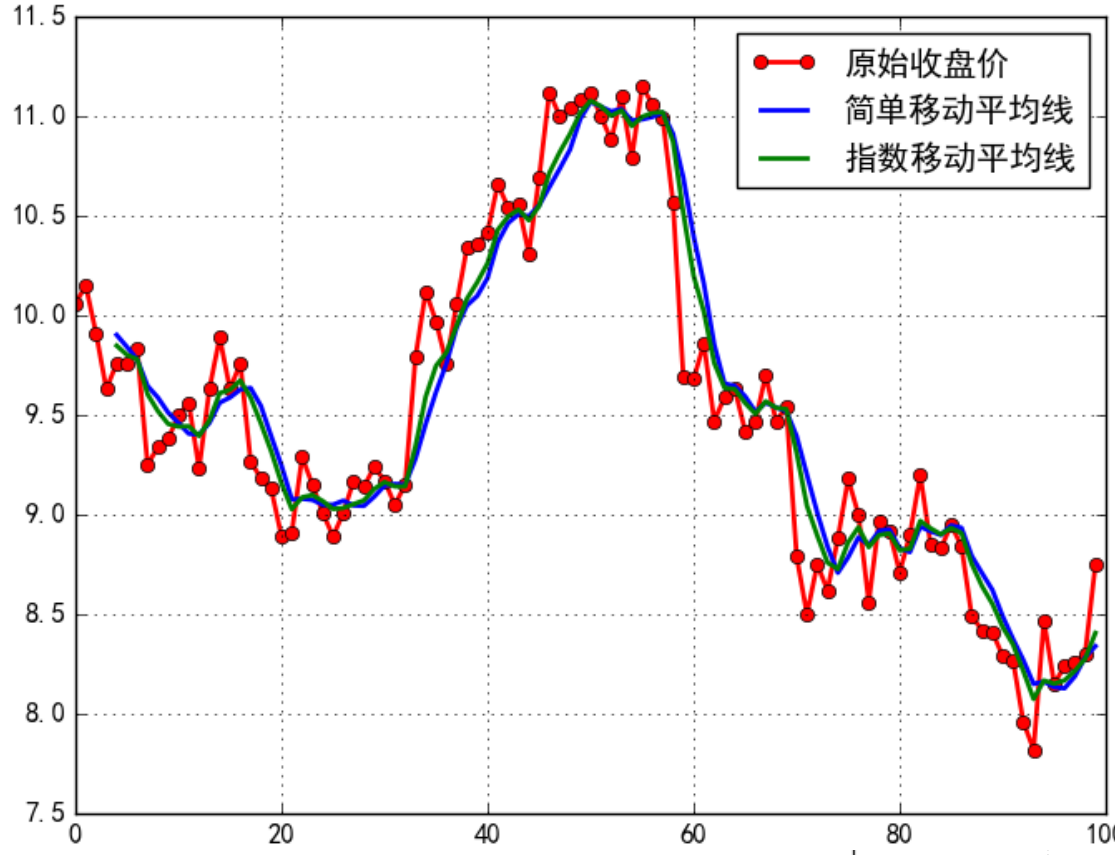
VGGNet

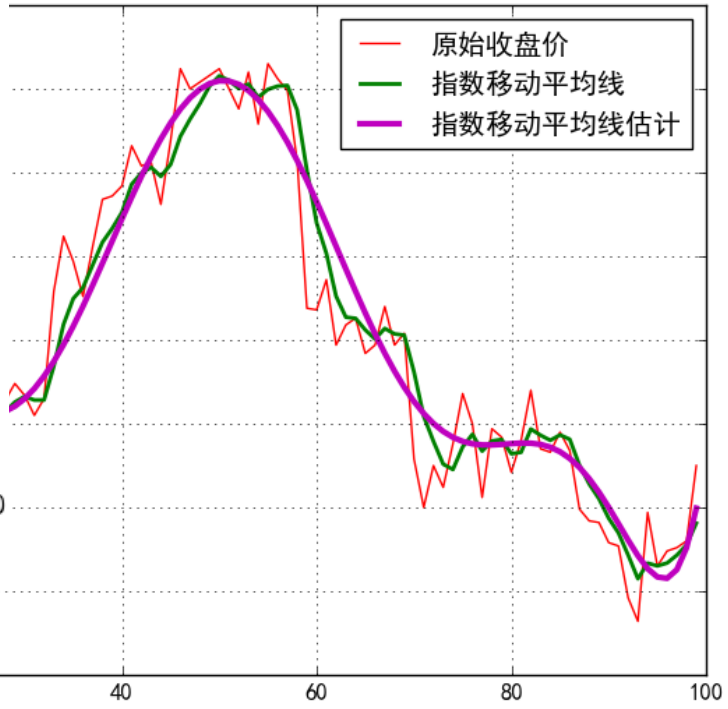
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

某股票收盘价数据处理


原始收盘价
简单移动平均线
指数移动平均线
11.5
11.0
10.5
10.0
9.5
9.0
8.5
8.0
7.5
0
20
40
60
80
100


原始收盘价
指数移动平均线
指数移动平均线估计
8.0
7.5
0
20
40
60
80
100

Demo

作业

- 实现任何一个函数曲线/曲面的Python显示。
  - Matplotlib
- 尝试使用SVD实现图像处理和特征提取。
- 熟悉Python的Numpy/Scipy数值计算数学库。

我们在这里

- http://wenda.ChinaHadoop.cn
  - 视频/课程/社区
- 微博
  - @ChinaHadoop
  - @邹博_机器学习
- 微信公众号
  - 小象
  - 大数据分析挖掘

感谢大家！

恳请大家批评指正！