

基于视频的电熔镁炉工况识别系统→1.数据准备(data_prepare)

基于视频的电熔镁炉工况识别系统→2.模型建立

基于视频的电熔镁炉工况识别系统→3.模型合成

基于视频的电熔镁炉工况识别系统→4.基于子空间角度的核函数→1.子空间角度

基于视频的电熔镁炉工况识别系统→4.基于子空间角度的核函数→2.四种距离

基于视频的电熔镁炉工况识别系统→5.分类器设计

《基于视频的电熔镁炉工况识别系统→1.数据准备(data_prepare)》

整个项目参考代码是研究生毕业项目→[DTBox-0.5.zip](#)

描述	代码														
<p>1. 将视频00283切割为时长为5s, 帧率为10帧/秒的724个视频 参考见百度网盘→赵磊文集→项目→镁炉可视化系统(研究生毕业)→效果好00283.mp4</p> <p>视频00283的参数</p> <table><tr><td>1</td><td>Video Properties:</td></tr><tr><td>2</td><td>Width: 1920</td></tr><tr><td>3</td><td>Height: 1080</td></tr><tr><td>4</td><td>FrameRate: 25</td></tr><tr><td>5</td><td>Duration: 1449s</td></tr></table> <table><tr><th>原视频帧数</th><th>新视频帧数</th></tr><tr><td>1449×25=36225</td><td>724*50=36200</td></tr></table> <p>图像切割</p>	1	Video Properties:	2	Width: 1920	3	Height: 1080	4	FrameRate: 25	5	Duration: 1449s	原视频帧数	新视频帧数	1449×25=36225	724*50=36200	<pre>video_cut.m input: 效果好00283.mp4 output: video_segment 1 % cut a 10s video from the original video 2 clear ; close all; clc 3 video_name = 'g:\炉口火焰数据库制作\MP4\效果好00283.mp4'; 4 video_len = 5; 5 frameRate = 10; 6 v = VideoReader(video_name); 7 for i = 1:floor(v.Duration*v.FrameRate / (video_len*frameRate)) 8 vcut_name = ['.\video_segment\00283\' , num2str(i), '_00283']; 9 vcut = VideoWriter(vcut_name, 'MPEG-4'); 10 vcut.FrameRate = frameRate; 11 open(vcut); 12 temp = frameRate*video_len; 13 for j = (i-1)*temp+1:i*temp 14 I = read(v, j); 15 % Img=I(200:470, 350:1100, :); % (for 00279) 16 % Img=I(200:470, 350:1100, :); % (for 00282) 17 Img=I(200:470, 400:1450, :); % (for 00283) 271*1051 18 writeVideo(vcut, Img); 19 end 20 close(vcut); 21 fprintf(['the ', num2str(i), ' is generated!\n']); 22 end</pre>
1	Video Properties:														
2	Width: 1920														
3	Height: 1080														
4	FrameRate: 25														
5	Duration: 1449s														
原视频帧数	新视频帧数														
1449×25=36225	724*50=36200														
<p>2. 制作furnace_patches_724.mat 424M</p> <p>1. size=1*724</p> <p>2. 将724个视频数据保存到该mat文件中</p> <p>灰度化 图像缩放</p>	<pre>furnace_patches_724_create.m input: video_segment中的724个视频 output: furnace_patches_724.mat 724个视频图像序列数据 1 clear ; close all; clc 2 3 %% create furnace_patches_724 4 video_num = 724; 5 imgdb = cell(1, video_num); 6 addpath(genpath('.')); 7 for j = 1:video_num 8 dir = ['.\video_segment\00283\']; 9 v = VideoReader([dir, num2str(j), '_00283.mp4']); 10 len = v.FrameRate*v.Duration; 11 pictu_matri = zeros(floor(v.Height/4), floor(v.Width/4), len, 'uint8'); 12 for k = 1:len 13 pictu_matri(:, :, k) = imresize(rgb2gray(read(v, k)), [floor(v.Height/4), floor(v.Width/4)]); 14 end 15 imgdb{j} = pictu_matri; 16 end 17 save('.\furnace_patches_724.mat', 'imgdb');</pre>														
<p>3. 生成724*724 dist矩阵</p> <p>1. 使用calculateMetricLDS 参考见基于视频的电熔镁炉工况识别系统→4.基于子空间角度的核函数→2.四种距离</p>															

4. 对724个数据进行聚类

dist_analysis.m

input: 724*724dist矩阵

output: 聚好类的数据X Y num(每一类的个数)

```
1 clear ; close all; clc
2 %% dist analysis
3 name = '\distMatrix\_n=20_nv=1_724_martin.mat';
4 load(name);
5
6 limit_number = 25;%25: 3类
7 [dist_sort,dist_sort_index] = sort(dist,2);
8 result = dist_sort_index(1,1:limit_number);
9 for i = 2:724
10     if intersect(result(:,2:limit_number),dist_sort_index(i,2:limit_number))
11         continue;
12     else
13         result = [result;dist_sort_index(i,1:limit_number)];
14     end
15 end
16 result2 = result';
17 num=size(result',1);
18 X=result2(:);
19 Y=[zeros(num,1);ones(num,1);ones(num,1)+1];
20 data=[X,Y];
21 save('\cluster_result\data25X3_martin.mat','num','data')
```

$$\begin{cases} x[k+1] = Ax[k] + Bv[k] \\ y[k] = Cx[k] + w[k] \end{cases}$$

1. 该系统是一个高斯白噪声所驱动的一阶ARMA模型很多文献都这么写，且噪声 $Bv[k] \sim N(0, Q)$, $Q = BB^T \in R^{n \times n}$
2. LDS_model.m
 1. input: furnace_patches_724.mat
 2. output: _n=20_nv=1_724.mat 724个视频的 A B C X C0=Ymean=w[k] Q
 3. 调用suboptimalSystemID来求解模型参数

参数求解过程

代码

1. 构建输入序列矩阵、去均值

1. $Y^\tau = [y(1), \dots, y(\tau)] \in \mathbb{R}^{m \times \tau}$ 每一列是每帧的外观特征，反映外观信息
 - o m: 视频一帧的特征数 17884
 - o τ : 视频帧数 50

input: dataMatrix 68*263*50

1. 将dataMatrix变为17884*50维

```
1 F = size(dataMatrix, 3);
2
3 if (size(dataMatrix, 3)~=1)
4     I = double(reshape(dataMatrix, [], F));
5 else
6     F = size(dataMatrix, 2);
7     I = double(dataMatrix);
8 end
```

2. 去均值

```
1 C0 = mean(I, 2);
2 Y = I - repmat(C0, 1, F);
```

2. 选择状态维数，估计观测矩阵C。

1. $Y^\tau = U \Sigma V^T m * \tau = (m * \tau) * (\tau * \tau) * (\tau * \tau)$
2. C 为 $\hat{C} = U \in R^{m \times n}$
3. $\hat{X}^\tau = \Sigma V^T \in R^{n \times \tau}$ 每一列是每帧的状态特征，反映动态信息:

```
1 [U, S, V] = svd(Y, 0);
2 C = U(:, 1:n);
3 X = S(1:n, 1:n)*V(:, 1:n)';
```

3. 估计状态转移矩阵A并进行正则化

1. $[x(2), \dots, x(\tau)] = A[x(1), \dots, x(\tau-1)]$
2. $\hat{A} = [\hat{x}(2), \dots, \hat{x}(\tau)][\hat{x}(1), \dots, \hat{x}(\tau-1)]^{-1} \in R^{n \times n}$ 这里
3. A除以A的谱半径

```
1 A = X(:, 2:F)*pinv(X(:, 1:(F-1)));
2 e = eig(A);
3 e = abs(e);
4 target = 0.9999;
5 if any(e>=target)
6     spectral_radius = max(e);
7     A = A*target/spectral_radius;
8 end
```

4. 估计输入矩阵B

1. 定义
- BV
- 为
- S
- ,

$$V = [v(1), \dots, v(\tau-1)] \in R^{n_v \times (\tau-1)}$$

所以说B是S的一部分

- 2.
- $\hat{S} = [\hat{x}(2), \dots, \hat{x}(\tau)] - \hat{A}[\hat{x}(1), \dots, \hat{x}(\tau-1)] \in R^{n \times (\tau-1)}$

$$\hat{Q} = \frac{1}{\tau-1} \hat{S} \hat{S}^T$$

3. 1.S的每一列代表1个样本，且
- $\sim N(0, Q)$
- ，即均值为0 2. 因此协方差矩阵这样写

4. 由于
- $Q = BB^T \in R^{n \times n}$
- ，所以可以估计

$$\hat{B} = \frac{1}{\sqrt{\tau-1}} \hat{S} \in R^{n \times n_v}$$

- 5.
- $\hat{S} = U_S \Sigma_S V_S^T$
- $n \times (\tau-1) = n \times n \times n \times (\tau-1) \times (\tau-1) \times (\tau-1)$
- 因为
- $n < \tau-1$

$$\hat{B} = \frac{1}{\sqrt{\tau-1}} U_S \Sigma_S \in R^{n \times n_v}$$

- 6.
- $n \times n_v \times n_v \times n_v$

- 7.
- $Q = BB^T \in R^{n \times n}$

```

1 S      = X(:, 2:F) - A*X(:, 1:(F-1));
2 [Uv, Sv, Vv] = svd(S, 0);
3 B      = Uv(:, 1:nv)*Sv(1:nv, 1:nv)./sqrt(F-1);
4 Q      = B*B';

```

自回归AR(p)模型 [\[编辑 \]](#)

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t.$$

自回归模型描述的是当前值与历史值之间的关系。

移动平均MA(q)模型 [\[编辑 \]](#)

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

移动平均模型描述的是自回归部分的误差累计。

ARMA(p,q)模型 [\[编辑 \]](#)

ARMA (p, q) 模型中包含了p个自回归项和q个移动平均项，ARMA (p, q) 模型可以表示为：

$$X_t = c + \varepsilon_t + \sum_{i=1}^p \varphi_i X_{t-i} + \sum_{j=1}^q \theta_j \varepsilon_{t-j}$$

$$\begin{cases} x[k+1] = Ax[k] + Bv[k] \\ y[k] = Cx[k] + w[k] \end{cases}$$

generateFromLDS.m

input: 一个LDS模型的**A B C X0 C0=Ymean=w[k]**

output: I:图像序列 X:状态变量

描述header2

1. 参数设置	<pre>1 %% parameter settings 2 name = './LDS_Model/furnace/_n=20_nv=1_724.mat'; 3 load(name); 4 ith = 1; 5 data = imgpara(ith); 6 [A,B,C,X0,C0] = deal(data.A,data.B,data.C,data.X0,data.C0); 7 [n,nv] = size(B); 8 F = 400; %Synthetic frame number 9 [r,c] = deal(68,263); 10 11 X = zeros(n,F); 12 I = zeros(r,c,F);</pre>
2. 使用一阶ARMA模型生成图像序列	<pre>1 for i=1:F 2 if i==1 3 X(:,i) = X0;%A*X0 + B*randn(nv,1); 4 else 5 X(:,i) = A*X(:,i-1) + B*randn(nv,1); 6 end 7 I(:, :, i) = reshape(C*X(:,i)+C0,[r c]); 8 end</pre>
3. 生成视频	<pre>1 video_name = ['./\video_generate\','_n=',num2str(n),'_nv=',num2str(nv),'_ith=',num2str(ith),'_F=',num2str(F)]; 2 v = VideoWriter(video_name,'MPEG-4'); 3 v.FrameRate=10; 4 open(v); 5 for j = 1:F 6 writeVideo(v,uint8(I(:, :, j))); 7 end 8 close(v);</pre> <p>由于I中有大于255的数，所以使用uint8将数据缩放在[0,255]间</p>

subspaceAnglesAR.m

input: sys1, sys2

output: theta n维

描述

代码

1. 一般来说, 矩阵 $A \in R^{m \times p}$ 和 $B \in R^{m \times q}$ 的 $q(p \geq q)$ 个主角定义为:

$$\cos \theta_1 = \max_{x_1 \in R^p, y_1 \in R^q} \frac{x_1^T A^T B y_1}{\|A x_1\| \|B y_1\|}$$

$$\vdots$$

$$\cos \theta_k = \max_{\substack{x_k \in R^p, y_k \in R^q \\ A x_k \perp A x_1, \dots, A x_{k-1} \\ B y_k \perp B y_1, \dots, B y_{k-1}}} \frac{x_k^T A^T B y_k}{\|A x_k\| \|B y_k\|}$$

1. 图

$$0 \leq \theta_1 \leq \dots \leq \theta_q \leq \frac{\pi}{2}$$

1.

2. $\cos \theta$ 越大, θ 越小, 表明两个矩阵的距离越近

2. 求解 矩阵 $A \in R^{m \times p}$ 和 $B \in R^{m \times q}$ 的 $q(p \geq q)$ 个主角

$$\begin{pmatrix} 0 & A^T B \\ B^T A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} A^T A & 0 \\ 0 & B^T B \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

1. 图

差和协方差, 总之可以反映A与B的关系

2. 矩阵的维度是 $(p+q) \times (p+q)$ 的, 计算特征值并进行排序

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{p+q}$$

3. 取前 q 大特征值: $\lambda_1 = \cos \theta_1, \dots, \lambda_q = \cos \theta_q \geq 0$

3. 自回归模型 M_1 和 M_2 的子空间主角定义:

1. 两个自回归模型 M_1 、 M_2 可分别由模型参数 A_1 和 C_1 、 A_2 和 C_2 进行表征, 它们的维度是 $\infty \times n$ 的可观测矩阵如下:

$$O_\infty(M_i) = \begin{bmatrix} C_i^T, A_i^T C_i^T, (A_i^T)^2 C_i^T, \dots \end{bmatrix}^T$$

1. 图

2. 则角度可定义为 $[M_1 \triangleleft M_2] = [\mathcal{O}_\infty(M_1) \triangleleft \mathcal{O}_\infty(M_2)]$

4. 求解自回归模型 M_1 和 M_2 的子空间主角

1. 求解 $\mathcal{O}_\infty(M_1)$ 和 $\mathcal{O}_\infty(M_2)$ 的方差和协方差, 可构造离散的李雅普诺夫等式:

$$1. Z^T X Z - X + Q = 0$$

$$Z = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix}, Q = \begin{pmatrix} C_1^T \\ C_2^T \end{pmatrix} (C_1 \ C_2)$$

2. 令

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}$$

可求解

3. x_{11}, x_{12}, x_{21} 和 x_{22} 分别对应了 $O_\infty(M_1)^T O_\infty(M_1)$, $O_\infty(M_1)^T O_\infty(M_2)$, $O_\infty(M_2)^T O_\infty(M_1)$ 和 $O_\infty(M_2)^T O_\infty(M_2)$ 维度均是 $n \times n$

```
1 A1 = sys1.A;
2 C1 = sys1.C;
3 A2 = sys2.A;
4 C2 = sys2.C;
5
6 n = size(A1, 1);
7 m = size(C1, 1);
8 Z = [A1 zeros(n); zeros(n) A2];
9 C = [C1 C2];
10 X = dlyap(Z', C' * C); % ① X = DLYAP(A, Q) A * X * A' - X + Q = 0 ② 分块矩阵的转秩
```

2. 求解自回归模型 M_1 和 M_2 的子空间主角

$$\begin{pmatrix} 0 & x_{12} \\ x_{21} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x_{11} & 0 \\ 0 & x_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

1.

2. 可变换为求

$$\begin{pmatrix} 0 & x_{11}^{-1}x_{12} \\ x_{22}^{-1}x_{21} & 0 \end{pmatrix} = \begin{pmatrix} x_{11} & 0 \\ 0 & x_{22} \end{pmatrix}^{-1} \begin{pmatrix} 0 & x_{12} \\ x_{21} & 0 \end{pmatrix}$$

的特征值

3. 矩阵的维度是 $2n \times 2n$ ，计算特征值并进行排序

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{2n}$$

4. 则取前 n 个 $\lambda_1 = \cos \theta_1, \dots, \lambda_n = \cos \theta_n \geq 0$

```

1 E = eig([zeros(n) pinv(X(1:n,1:n))*X(1:n,n+1:2*n)]...%eigenvalues
2 pinv(X(n+1:2*n,n+1:2*n))*X(n+1:2*n,1:n) zeros(n)]);
3 E = real(E);
4 E = max(-ones(size(E)),E); %保证cos(theta)的值域是[-1,1],
5 E = min(ones(size(E)),E);
6 E = sort(E,'descend');
7 theta = acos(E(1:n));
解出的theta∈[0,90°]

```

• 线性系统的可控性和可观性:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

1. 给定系统的动态方程为:

$$\begin{aligned} \dot{x}_1 &= 4x_1 + u \\ \dot{x}_2 &= -5x_2 + 2u \end{aligned}$$

2. 将其表示为标量方程组的形式 $y = -6x_2$ 3. 可控就是状态可以转移到任何状态: 取值不同的 u , 可以解出不同的 x_1 和 x_2 , 因此能选择控制量 u 使状态初始点到4. 可观就是每个时刻的状态都可以被观测到: y 只能反映状态变量 x_2 , 因此系统是不完全观测的

$$Q = [B \ AB] = \begin{bmatrix} 1 & 4 \\ 2 & -10 \end{bmatrix}$$

5. 可控性判据:

的rank=2, 因此可控

$$R^T = [C^T \ A^T C^T] = \begin{bmatrix} 0 & 0 \\ -6 & 30 \end{bmatrix}$$

6. 可观性判据:

的rank=1, 因此不可观

• 李雅普诺夫稳定

$$\dot{x} = f(x(t)), \quad x(t_0) = x_0,$$

其中 $x(t) \in U$ 是系统的状态向量, $f: U \rightarrow E$ 是 U 上的连续函数。假设函数 f 有一个零点: $f(a) = 0$, 则常数函数: $x = a$ 是动力系统的驻定解 (或称平衡解)。称 a 是动力系统的平衡点

1. 称点 a 李雅普诺夫稳定 (简称稳定), 如果对每个 $\epsilon > 0$, 均存在 $\delta > 0$, 使得对所有满足 $\|x_0 - a\| < \delta$ 的 x_0
2. 称点 a 渐近稳定, 如果点 a 李雅普诺夫稳定, 且存在 $\delta > 0$, 使得对所有满足 $\|x_0 - a\| < \delta$ 的 x_0 , $\lim_{t \rightarrow \infty} x(t) = a$ 附近的轨迹均能维持在 a 附近; 渐进稳定是初始条件在平衡点 a 附近的轨迹均能趋近于 a

2. 李雅普诺夫稳定性第二定理

• $V(x) \geq 0$ 只有在 $x = 0$ 处等号成立 (正定函数)1. • $\dot{V}(x(t)) < 0$ (负定)

, 则系统为渐进稳定

例如考虑以下的系统

$$\dot{x} = -x^3$$

希望用李雅普诺夫函数来确认 $x = 0$ 附近的稳定性。令

$$V(x) = 0.5x^2$$

$V(x)$ 本身为正定函数。而 $V(x)$ 的导函数如下

$$\dot{V}(x(t)) = \frac{\partial V}{\partial x}(-x^3) = -x^4$$

2. 为负定函数，因此上述系统在 $x = 0$ 附近为渐近稳定。 $\dot{V}(x(t)) = -x^4$

3. 李雅普诺夫函数

1. 对于 $\dot{x} = Ax$

$$\dot{V}(x) = \dot{x}^T Px + x^T P \dot{x} = (Ax)^T Px + x^T PAx$$

2. 选取 $V(x) = x^T Px$ ，则 $\dot{V}(x) = x^T (A^T P + PA)x$

3. 要求系统是渐进稳定的，需满足 $A^T P + PA < 0$ 李雅普诺夫不等式

4. 等价于 $A^T P + PA = -Q$ 李雅普诺夫方程， Q 是任意正定实对称矩阵

实际应用是，通常先选取一个正定矩阵 Q ，
带入李雅普诺夫方程，解出矩阵 P ，然后按
判定 P 的正定性，进而判断 系统渐进稳定

5. 常取 $Q = I$ (单位阵)

定理 (连续时间版本)：给定任意 $Q > 0$ ，存在唯一 $P > 0$ 满足 $A^T P + PA + Q = 0$ 的充份必要条件是线性系
可以验证系统的稳定性。

6. **定理** (离散时间版本)：给定任意 $Q > 0$ ，存在唯一 $P > 0$ 满足 $A^T P A - P + Q = 0$ 的充份必要条件是线性系
在李雅普诺夫方程中只有一个转秩是 A 转秩

《基于视频的电熔镁炉工况识别系统→4.基于子空间角度的核函数→2.四种距离》

calculateMetricLDS.m

1. input: _n=20_nv=1_724.mat 724个视频的 A B C X C0=Ymean=w[k] Q + 距离序号

- 1 - Finsler
- 2 - Matrin
- 3 - Gap
- 4 - Frobenius

2. output: dist 距离矩阵

3. 调用subspaceAnglesAR.m进行子空间角度计算 参考见基于视频的电熔镁炉工况识别系统→4.基于子空间角度的核函数→1.子空间角度

4. 调用plotDistance.m绘制距离矩阵

5. 被kernal_matrix.m脚本调用

距离公式

代码

1. FinslerDistance

$$d_F(M_1, M_2) = \theta_{\max}$$

angles的维数是 n(状态维数)*video_num(视频个数)*video_num

```
1 N = size(angles, 2);
2 dist = zeros(N, N);
3
4 if N > 1
5     for i = 1:N
6         for j = 1:i-1
7             dist(i, j) = max(angles(:, i, j));
8         end
9     end
10    dist = dist + dist';
11
12 else
13    dist = max(angles);
14 end
```

2. MartinDistance



$$d_M(M_1, M_2)^2 = \ln \prod_{i=1}^n \frac{1}{\cos^2 \theta_i}$$


```
1 N = size(angles, 2);
2 dist = zeros(N, N);
3
4 if N > 1 %这样的循环就不计算主对角线上的元素了
5     for i = 1:N
6         for j = 1:i-1
7             dist(i, j) = sqrt(-sum(log(cos(angles(:, i, j)).^2)));
8         end
9     end
10    dist = dist + dist';
11
12 else
13    dist = sqrt(-sum(log(cos(angles).^2)));
14 end
```

3. GapDistance

$d_g(M_1, M_2) = \sin \theta_{\max}$

```
1 N = size(angles, 2);
2 dist = zeros(N, N);
3
4 if N > 1
5     for i = 1:N
6         for j=1:i-1
7             dist(i, j) = sin(max(angles(:, i, j))));
8         end
9     end
10    dist = dist + dist';
11 else
12    dist = sin(max(angles));
13 end
```

4. Frobenius

 $d_f(M_1, M_2)^2 = 2 \sum_{i=1}^n \sin^2 \theta_i$

```
1 N = size(angles, 2);
2 dist = zeros(N, N);
3
4 if N > 1
5     for i = 1:N
6         for j=1:i-1
7             dist(i, j) = sqrt(2*sum(sin(angles(:, i, j)).^2));
8         end
9     end
10    dist = dist + dist';
11 else
12    dist = sqrt(2*sum(sin(angles).^2));
13 end
```

《基于视频的电熔镁炉工况识别系统→5.分类器设计》

input: _n=20_nv=1_martin_gram

output: 给视频均贴上标签

classifier: svmclassifier.py和ownclassifier.m

描述	代码
1. 导入模块	<pre>1 import scipy.io 2 import numpy as np 3 from sklearn import svm</pre>
2. 数据准备 1. 导入从matlab中得到的mat文件， 获取gram矩阵 2. 使用scipy.io模块的loadmat读取 mat文件 1 In[3]: type(mat) 2 Out[3]: dict	<pre>1 mat = scipy.io.loadmat('._n=20_nv=1_martin_gram.mat') 2 num = int(mat['num']) 3 X = mat['data'][:,0] 4 X = X[:,np.newaxis] 5 Y = mat['data'][:,1] 6 gram = mat['dist']</pre> <p>num: 25 每一类的视频个数 X: (75,1) 75个炉口视频的视频编号 Y: (75,) 75个炉口视频的标签 gram: (75,75) 75个炉口视频互相间的martin距离</p>
3. 模型选择 1. 根据高斯核函数意义设计 gram_final, 使得: 当视频样本相 似度高时, 核函数值为趋于1; 反 之, 趋于0	<pre>1 gamma=0.3 2 gram_final = np.exp(-gamma*gram)</pre>
2. 拆分数据集, 得到gram_train及 gram_test 1. gramtrain= Xtrain*Xtrain' ; grampredict= Xtest*Xtrain'	<pre>1 train_each_num=5 2 train_index=list(range(0,train_each_num))+list(range(num,num+train_each_num))+list(range(num*2,num*2+train_ 3 gram_train = gram_final[train_index,:][:,train_index] 4 gram_test = gram_final[:,train_index]</pre> <p>第3行不能这样写: 1 In[9]: gram_final[train_index,train_index].shape 2 Out[9]: (15,)</p> <p>参见2.数据计算→numpy→4.ndarray取值操作</p>
3. 拟合模型	<pre>1 clf = svm.SVC(kernel='precomputed') 2 clf.fit(gram_train, Y[train_index])</pre>
4. 评估	<pre>1 Z = clf.predict(gram_test) 2 p = sum(Y==Z)/Y.size 3 print(p)</pre> <p>0.9466666666666667</p>
5. 此外, 还设计了一种分类器, 可以比较未 标记视频与已标记的三个类别的视频的距 离, 距离哪个类别近, 就预测为哪一类。	<pre>1 %% init 2 clear; close all; clc; 3 addpath(genpath('.')); 4 5 name = '_n=20_nv=1_martin_gram.mat'; 6 load(name); 7 X=data(:,1); 8 [a,b,c] = deal(X(1),X(num+1),X(num*2+1)); 9 10 name = '_n=20_nv=1_724_martin.mat'; 11 load(name); 12 label = zeros(724,1); 13 [label(a),label(b),label(c)] = deal(0,1,2); 14 for i=1:724 15 if i==a i==b i==c 16 continue; 17 else 18 if min([dist(a,i),dist(b,i),dist(c,i)]) == dist(a,i) 19 label(i) = 0; 20 elseif min([dist(a,i),dist(b,i),dist(c,i)]) == dist(b,i) 21 label(i) = 1; 22 else 23 label(i) = 2; 24 end 25 end 26 end</pre>