**T1：航空发动机和火箭发动机的区别**

首先，火箭发动机是同时需要氧化剂和燃料的，而航空发动机却只用携带燃料就行了，氧化剂相等于是空气中的氧气。其次，火箭发动机可以在真空环境中工作，但航空发动机需要在大气稠密的地方工作。火箭发动机可分为以下几类：

1、化学火箭发动机

化学火箭发动机利用推进剂的化学能，在燃烧室中进行化学反应，产生高温、高压燃气，高速气流向后喷出，产生反作用推力，由燃烧室、喷管以及液体推进剂供应系统或固体推进剂装药组成。所用的推进剂包括燃烧剂和氧化剂，它们既是能源又是工质。

2、电火箭发动机

电火箭发动机是用电能加速工质（工作介质）形成高速射流而产生推力的火箭发动机，能源和工质是分开的。电能由飞行器提供，工质常用氢、氮、氩或碱金属（铯、汞、铷、锂等）的蒸气。电火箭发动机比冲高、寿命长（可起动上万次，累计工作上万小时），但推力小于100牛（10公斤力），适用于航天器的姿态控制、位置保持和星际航行等。

3、核火箭发动机

核火箭发动机，以核为初始能源，通过核反应释放的能量绐液态氢加热，被加热的氢经过喷管膨胀加速后排出，产生推力的火箭发动机。核火箭发动机基本上是液体火箭犮动机的扩展，伹其加热的能源不是来自化学 反应，而是来自核能，使用液态氢作为核火箭发动机的工作流体是因为氢的相对质量最小。

**T2：异常检验算法原理：**

异常检测(Anomaly detection)，是机器学习算法的一个常见应用。它虽然主要用于非监督学习问题，但从某些角度看，它又类似于一 些监督学习问题。异常检测就是发现与大部分对象不同的对象（发现离群点）异常对象的属性值明显偏离期望的属性值，异常检测也称偏差检测；异常在某种意义上是一种例外，也被称为例外挖掘；异常对象是相对罕见的。异常检测主要使用数理统计和数据挖掘技术，算法主要有： 基于模型、基于邻近度、基于密度和基于聚类。

* 基于模型的技术：建立一个数据模型，异常是那些同模型不能完美拟合的对 象。例如，数据分布的模型可以通过估计概率分布的参数来创建。如果一个 对象不服从该分布，则认为它是一个异常。
* 基于邻近度的技术：在对象之间定义邻近性度量，异常对象是那些远离大部 分其他对象的对象。当数据能够以二维或者三维散布图呈现时，可以从视觉 上检测出基于距离的离群点
* 基于邻近度的技术：在对象之间定义邻近性度量，异常对象是那些远离大部 分其他对象的对象。当数据能够以二维或者三维散布图呈现时，可以从视觉 上检测出基于距离的离群点
* 基于聚类的技术：聚类和异常检测的目标是估计分布的参数，以最大化数据 的总似然（概率）。聚类分析用于发现强相关的对象组，异常检测是发现与 其他对象弱相关的对象，因此，聚类可以用于异常检测

**import** scipy**.**io **as** sio

**import** matplotlib**.**pyplot **as** plt

**import** numpy **as** np

**def** opendata**(**textname**):**

data**=**sio**.**loadmat**(**textname**)**

X**=**data**[**'X'**]**

**return** data**,**X

**def** estimate\_parameters\_for\_gaussian\_distribution**(**X**):**

mu**=**np**.**mean**(**X**,**axis**=**0**)**

sigma2**=**np**.**var**(**X**,**axis**=**0**)**

**return** mu**,**sigma2

**def** gaussian\_distribution**(**X**,**mu**,**sigma2**):**

p**=(**1**/**np**.**sqrt**(**2**\***np**.**pi**\***sigma2**))\***np**.**exp**(-(**X**-**mu**)\*\***2**/(**2**\***sigma2**))** ***# the (prob1,prob2) for many points***

**return** np**.**prod**(**p**,**axis**=**1**)** ***# prob for many points***

**def** visualize\_contours**(**mu**,**sigma2**):**

x**=**np**.**linspace**(**5**,**25**,**100**)**

y**=**np**.**linspace**(**5**,**25**,**100**)**

xx**,**yy**=**np**.**meshgrid**(**x**,**y**)**

X**=**np**.**concatenate**((**xx**.**reshape**(-**1**,**1**),**yy**.**reshape**(-**1**,**1**)),**axis**=**1**)**

z**=**gaussian\_distribution**(**X**,**mu**,**sigma2**).**reshape**(**xx**.**shape**)**

cont\_levels**=[**10**\*\***h **for** h **in** range**(-**20**,**0**,**3**)]**

plt**.**contour**(**xx**,**yy**,**z**,**cont\_levels**)**

**def** error\_analysis**(**yp**,**yt**):**

tp**,**fp**,**fn**,**tn**=**0**,**0**,**0**,**0

**for** i **in** range**(**len**(**yp**)):**

**if** yp**[**i**]==**yt**[**i**]:**

**if** yp**[**i**]==**1**:**

tp**+=**1

**else:**

tn**+=**1

**else:**

**if** yp**[**i**]==**1**:**

fp**+=**1

**else:**

fn**+=**1

precision**=**tp**/(**tp**+**fp**)** **if** tp**+**fp **else** 0

recall**=**tp**/(**tp**+**fn**)** **if** tp**+**fn **else** 0

f1**=**2**\***precision**\***recall**/(**precision**+**recall**)** **if** precision**+**recall **else** 0

**return** f1

**def** select\_threshold**(**yval**,**pval**):**

epsilons**=**np**.**linspace**(**min**(**pval**),**max**(**pval**),**1000**)**

l**=**np**.**zeros**((**1**,**2**))**

**for** e **in** epsilons**:** ***# for each epislon, there will be an evaluation f1***

ypre**=(**pval**<**e**).**astype**(**float**)**

f1**=**error\_analysis**(**ypre**,** yval**)**

l**=**np**.**concatenate**((**l**,**np**.**array**([[**e**,**f1**]])),**axis**=**0**)**

index **=** np**.**argmax**(**l**[...,** 1**])**

**return** l**[**index**,** 0**],** l**[**index**,** 1**]**

**def** detection**(**X**,**e**,**mu**,**sigma2**):**

p**=**gaussian\_distribution**(**X**,**mu**,**sigma2**)**

anomaly\_points **=** np**.**array**([**X**[**i**]** **for** i **in** range**(**len**(**p**))** **if** p**[**i**]<**e**])**

**return** anomaly\_points

**def** visualize\_dataset**(**X**):**

plt**.**scatter**(**X**[...,** 0**],** X**[...,** 1**],** marker**=**'x'**,** label**=**'point'**)**

**def** circle\_anomaly\_points**(**X**):**

plt**.**scatter**(**X**[...,** 0**],** X**[...,** 1**],** s**=**80**,** facecolors**=**'none'**,** edgecolors**=**'r'**,** label**=**'anomaly point'**)**

**def** main**(**textname**):**

data**,**X**=**opendata**(**textname**)** ***#X is the training set without y label, the goal is to train the model (miu, sigma2)***

X**=**np**.**vstack**([**X**,[**10**,**10**],[**9**,**9**],[**8**,**8**]])**

visualize\_dataset**(**X**)**

mu**,**sigma2**=**estimate\_parameters\_for\_gaussian\_distribution**(**X**)**

visualize\_contours**(**mu**,** sigma2**)**

Xval **=** data**[**'Xval'**]** ***#cross validation dataset, with both x and y label, this will calcutate the error and thus epislon and F1***

yval **=** data**[**'yval'**]**

e**,** f1 **=** select\_threshold**(**yval**.**ravel**(),** gaussian\_distribution**(**Xval**,** mu**,** sigma2**))** ***#the goal is to find the best epsilon with largest F1***

anomaly\_points **=** detection**(**X**,** e**,** mu**,** sigma2**)** ***# like testing***

circle\_anomaly\_points**(**anomaly\_points**)**

plt**.**title**(**'anomaly detection'**)**

plt**.**legend**()**

plt**.**show**()**

print**(**'e= '**,**e**,**", f1= "**,**f1**)**

main**(**"ex8data1.mat"**)**

e= 7.739625014642587e-05 , f1= 0.8750000000000001

