1 Генерация тестовых данных

Сгенерируем тестовые данные. Положим число признаков $n = 5$, общее число наблюдений — $N = 100$. Пусть большей частью набора будут нормально распределённые случайные вектора с заранее заданной ковариационной матрицей и математическими ожиданиями $(m_1, \ldots, m_n)$, выбросами — две реализации нормально распределённого случайного вектора с такой же ковариационной матрицей и другим математическим ожиданием, две реализации равномерно распределённого на $[m_1-1; m_1+1] \times \cdots \times [m_n-1; m_n+1]$ случайного вектора и одна константа $\mathbf{0} \in \mathbb{R}^n$. Ковариационная матрица, местоположение выбросов и их конкретные значения генерируются для задачи случайным образом.

$$\sum_{i=1}^{m} \int_{-\infty}^{x} p_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

```
covariance = np.zeros((n,n))
for i in xrange(n):
  for j in xrange(n):
    covariance[i,j] = covariance[j, i] if covariance[j, i] != 0 else 2*random.random()-1
covariance = covariance.dot(np.matrix.transpose(covariance))
```

```
covariance:
[[ 2.0219  0.1149 -1.3835  0.5156  1.3031]
 [ 0.1149  0.3081 -0.0256  0.1588 -0.0837]
 [-1.3835 -0.0256  1.5708 -0.3568 -0.4567]
 [ 0.5156  0.1588 -0.3568  1.3434  0.0364]
 [ 1.3031 -0.0837 -0.4567  0.0364  1.3756]]
```

```
outliers = random.sample(range(N), 5)
```

```
outliers:
[10, 75, 6, 98, 8]
```

```
outlier_mean = np.random.randint(1,10, size=n)
```

```
outlier_mean:
[4 7 4 5 9]
```

```
def random_vector_with_outliers(R, N=1):
  k = len(R[0]) # = n
  mean = np.random.uniform(-5,5, size=k)
  x = map(lambda i: mean + np.linalg.cholesky(R).dot(np.random.normal(size=k)), xrange(N))
  for (i, index) in enumerate(outliers):
    if i in [0, 1]:
      x[index] = outlier_mean + np.linalg.cholesky(R).dot(np.random.normal(size=k))
    elif i in [2, 3]:
      x[index] = mean + np.random.uniform(0, 1, size=k)
    else:
      x[index] = np.full((k, ), 0)
  return np.array(x)
```

2 Проверка выборки на выбросы

Положим $\bar{x} = \frac{1}{\#I}\sum_{x\in I} x$, где $x = (x_{i1}, \ldots, x_{ip})$ – вектор, соответствующий одному наблюдению, $\hat{\Sigma}_I = \frac{1}{\#I-1}\sum_{x\in I}(x-\bar{x})(x-\bar{x})^T$. Тогда для расстояния Махаланобиса $D_I^2 = (y-\bar{x})\hat{\Sigma}_I^{-1}(y-\bar{x})^T$ (функции от набора наблюдений I и наблюдения y, к этому набору не относящегося) нам будет известно распределение случайной величины $\frac{(N-n)N}{(N^2-1)n}D^2 \sim F(n, N-n)$.

2.1 Полный набор данных

Реализуем вычисление расстояния Махаланобиса от вектора до набора данных, из которого этот вектор исключён. Параметром метода будет являться порядковый номер исключаемого вектора (можно реализовать более общий вариант с передачей всего набора данных в метод, но в нашем случае, где расстояние Махаланобиса будет пересчитываться для каждого наблюдения из набора, разумнее хранить преподсчитанные данные для всего набора, нежели пересчитывать их каждый раз).

```
xmean = np.mean(X, axis=0)
def mahalanobis(j):
  xnorm = (N*xmean - X[j])/(N-1) # mean of X with X[j] excluded
  sigma = np.matrix(
    np.sum( # np.outer(v1, v2) generates outer product of vectors v1 and v2
      map(lambda x: np.outer(x - xnorm, x - xnorm), np.delete(X, j, axis=0)), axis=0) / (N - 2))
  # matrix.get() returns this matrix inversed, i.e.
  # sigma.dot(sigma.getI()) == np.eye((len(sigma), ))
  dmah = (X[j] - xmean).dot(sigma.getI()).dot(X[j] - xmean)
```

Проверим для каждого $j \in 1:N$ гипотезу $H_0 : X[j]$ не является выбросом. Тестовая статистика $t = \frac{(N-n)N}{(N^2-1)n}D^2$ будет иметь в случае верности нулевой гипотезы распределение $F(n, N-n)$, «идеальным значением» будет $t = 0$, следовательно, критическая область – правый хвост распределения Фишера. Посчитаем граничное значение:

```
critical = stats.f.ppf(0.975, n, N-n)
```

```
critical value:
2.70306913768
```

Для каждого наблюдения будем принимать гипотезу о принадлежности исходному распределению, если $t_i <$ critical. Результаты теста для каждого наблюдения:

```
[[ 1.0457]] adopt
[[ 0.7614]] adopt
[[ 1.8781]] adopt
[[ 0.0813]] adopt
[[ 1.1605]] adopt
[[ 1.7996]] adopt
[[ 1.3681]] adopt outlier [ 4.9424  1.7848 -0.8041 -3.9543 -0.6798]
[[ 0.6608]] adopt
[[ 18.3129]] reject outlier [ 0.  0.  0.  0.  0.]
[[ 0.3323]] adopt
[[ 16.8292]] reject outlier [ 2.6487  7.5196  5.933   4.6204  8.3381]
[[ 0.7986]] adopt
```

[[0.3004]] adopt
[[0.4898]] adopt
[[1.1149]] adopt
[[0.185]] adopt
[[0.4042]] adopt
[[0.645]] adopt
[[0.3759]] adopt
[[1.2891]] adopt
[[0.4401]] adopt
[[1.7471]] adopt
[[0.1778]] adopt
[[0.7712]] adopt
[[0.9823]] adopt
[[1.7491]] adopt
[[0.9871]] adopt
[[0.4989]] adopt
[[1.047]] adopt
[[0.825]] adopt
[[0.9851]] adopt
[[0.1957]] adopt
[[0.3739]] adopt
[[1.6055]] adopt
[[0.5036]] adopt
[[0.1936]] adopt
[[0.5144]] adopt
[[1.1171]] adopt
[[0.9064]] adopt
[[0.1245]] adopt
[[0.9226]] adopt
[[0.7847]] adopt
[[1.0354]] adopt
[[0.9227]] adopt
[[0.6368]] adopt
[[0.5173]] adopt
[[1.1537]] adopt
[[0.1734]] adopt
[[0.2972]] adopt
[[1.0726]] adopt
[[0.9636]] adopt
[[0.4922]] adopt
[[0.7781]] adopt
[[0.4672]] adopt
[[0.4781]] adopt
[[1.0275]] adopt
[[1.1618]] adopt
[[0.8831]] adopt
[[1.275]] adopt
[[1.2205]] adopt
[[0.762]] adopt
[[0.4388]] adopt
[[0.0615]] adopt
[[1.2633]] adopt

```
[[ 0.6132]] adopt
[[ 1.3029]] adopt
[[ 0.4034]] adopt
[[ 0.5201]] adopt
[[ 0.1339]] adopt
[[ 0.9367]] adopt
[[ 0.7078]] adopt
[[ 0.1007]] adopt
[[ 0.5064]] adopt
[[ 0.4215]] adopt
[[ 0.1816]] adopt
[[ 19.2255]] reject outlier [ 6.2772  8.3381  2.4341  5.3055  9.6133]
[[ 0.1926]] adopt
[[ 1.1194]] adopt
[[ 0.8238]] adopt
[[ 0.1872]] adopt
[[ 1.1504]] adopt
[[ 0.4132]] adopt
[[ 0.5887]] adopt
[[ 0.1768]] adopt
[[ 0.1181]] adopt
[[ 0.5048]] adopt
[[ 1.6632]] adopt
[[ 0.4865]] adopt
[[ 0.4065]] adopt
[[ 0.6746]] adopt
[[ 1.1645]] adopt
[[ 0.7609]] adopt
[[ 0.0242]] adopt
[[ 0.5754]] adopt
[[ 1.3486]] adopt
[[ 0.4405]] adopt
[[ 0.353]] adopt
[[ 0.3836]] adopt
[[ 0.1814]] adopt outlier [ 4.401   1.8795 -1.3817 -3.8892 -0.8363]
[[ 0.4179]] adopt
```

2.2 «Учебный» набор данных

Из-за того, что в выборку включаются не только «правильные» значения, но и другие выбросы, эффект сглаживается, и некоторые выбросы остаются включёнными в выборку. Попробуем теперь не включать в выборку выбросы вовсе, считать исходной группой только «правильные» данные.

```
xmean_wo = np.mean(np.delete(X, outliers, axis=0), axis=0)
sigma_wo = np.matrix(
    np.sum( # np.outer(v1, v2) generates outer product of vectors v1 and v2
      map(lambda x: np.outer(x - xmean_wo, x - xmean_wo),
          np.delete(X, outliers, axis=0)), axis=0) / (N - len(outliers) - 1))
sigma_wo_inv = sigma_wo.getI()
mahalanobis_without_outliers = lambda y: (y - xmean_wo).dot(sigma_wo_inv).dot(y - xmean_wo)
```

Результаты теста:

```
[[ 1.4279]] adopt
[[ 0.8319]] adopt
[[ 1.9598]] adopt
[[ 0.3221]] adopt
[[ 1.8657]] adopt
[[ 1.6153]] adopt
[[ 6.7957]] reject outlier [ 4.9424  1.7848 -0.8041 -3.9543 -0.6798]
[[ 0.9209]] adopt
[[ 109.341]] reject outlier [ 0.  0.  0.  0.  0.]
[[ 0.3706]] adopt
[[ 569.7186]] reject outlier [ 2.6487  7.5196  5.933   4.6204  8.3381]
[[ 0.7909]] adopt
[[ 0.6642]] adopt
[[ 0.6462]] adopt
[[ 1.3659]] adopt
[[ 0.2301]] adopt
[[ 0.6002]] adopt
[[ 0.6188]] adopt
[[ 0.5631]] adopt
[[ 1.2845]] adopt
[[ 0.4928]] adopt
[[ 1.6677]] adopt
[[ 0.2298]] adopt
[[ 0.8435]] adopt
[[ 1.1806]] adopt
[[ 1.7804]] adopt
[[ 1.4651]] adopt
[[ 0.7867]] adopt
[[ 1.1811]] adopt
[[ 1.3417]] adopt
[[ 1.1551]] adopt
[[ 0.2372]] adopt
[[ 0.5894]] adopt
[[ 1.9035]] adopt
[[ 0.7172]] adopt
[[ 0.5935]] adopt
[[ 0.9426]] adopt
[[ 1.3148]] adopt
[[ 0.936]] adopt
[[ 0.1503]] adopt
[[ 0.937]] adopt
[[ 1.3894]] adopt
[[ 1.0191]] adopt
[[ 1.4962]] adopt
[[ 0.8173]] adopt
[[ 0.8126]] adopt
[[ 1.1406]] adopt
[[ 0.1872]] adopt
[[ 0.3036]] adopt
[[ 1.7456]] adopt
[[ 1.0941]] adopt
[[ 0.7167]] adopt
```

```
[[ 1.0429]] adopt
[[ 0.6414]] adopt
[[ 0.6258]] adopt
[[ 1.1941]] adopt
[[ 1.1909]] adopt
[[ 0.8558]] adopt
[[ 1.4675]] adopt
[[ 1.1849]] adopt
[[ 2.0962]] adopt
[[ 0.9719]] adopt
[[ 0.0762]] adopt
[[ 1.6117]] adopt
[[ 1.0999]] adopt
[[ 1.9867]] adopt
[[ 0.4922]] adopt
[[ 0.6506]] adopt
[[ 0.3873]] adopt
[[ 1.0726]] adopt
[[ 1.0737]] adopt
[[ 0.3606]] adopt
[[ 0.4932]] adopt
[[ 0.9718]] adopt
[[ 0.7634]] adopt
[[ 563.5101]] reject outlier [ 6.2772  8.3381  2.4341  5.3055  9.6133]
[[ 0.2607]] adopt
[[ 1.322]] adopt
[[ 0.9147]] adopt
[[ 0.3053]] adopt
[[ 1.5698]] adopt
[[ 0.7714]] adopt
[[ 0.7205]] adopt
[[ 1.7062]] adopt
[[ 0.2937]] adopt
[[ 0.6181]] adopt
[[ 2.057]] adopt
[[ 0.492]] adopt
[[ 0.433]] adopt
[[ 0.6638]] adopt
[[ 1.1764]] adopt
[[ 0.9399]] adopt
[[ 0.2189]] adopt
[[ 0.9615]] adopt
[[ 2.0068]] adopt
[[ 0.5786]] adopt
[[ 0.7944]] adopt
[[ 0.4731]] adopt
[[ 0.3191]] adopt outlier [ 4.401   1.8795 -1.3817 -3.8892 -0.8363]
[[ 0.473]] adopt
```

«Правильность» исходных данных не оказывает сильного влияния на результат: критерий Махаланобиса отбрасывает результаты с соотношением между компонентами векторов, не похожим на таковое в большинстве наблюдений, но мало чувствителен к подозрительно маленькому разбросу.