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.

```
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:
[[ 1.9342  0.1358  0.6208 -0.6007  0.3643]
 [ 0.1358  3.1558  0.5881 -0.5367 -0.9235]
 [ 0.6208  0.5881  1.3179 -0.8067 -0.0151]
 [-0.6007 -0.5367 -0.8067  0.9547  0.127 ]
 [ 0.3643 -0.9235 -0.0151  0.127   1.6526]]
```

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

```
outliers:
[47, 3, 25, 70, 75]
```

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

```
outlier_mean:
[5 8 2 1 3]
```

```
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 $t_i < \mathsf{critical}$. Результаты теста для каждого наблюдения:

```
[[ 0.8789]] adopt
[[ 0.6975]] adopt
[[ 0.3099]] adopt
[[ 7.4146]] reject outlier [ 5.4478  7.4584  4.6565 -1.2107  2.3139]
[[ 1.3286]] adopt
[[ 1.487]] adopt
[[ 0.5469]] adopt
[[ 0.6403]] adopt
[[ 0.85]] adopt
[[ 0.9897]] adopt
[[ 0.4704]] adopt
[[ 0.2717]] adopt
[[ 1.3955]] adopt
[[ 0.0829]] adopt
[[ 0.8869]] adopt
[[ 1.6757]] adopt
[[ 0.675]] adopt
[[ 1.5473]] adopt
[[ 2.504]] adopt
[[ 0.9388]] adopt
```

```
[[ 1.166]] adopt
[[ 0.3128]] adopt
[[ 2.3233]] adopt
[[ 1.4193]] adopt
[[ 0.7419]] adopt
[[ 0.1894]] adopt outlier [ 4.1844 -2.2948 -0.1964  0.6355  2.8791]
[[ 0.9703]] adopt
[[ 0.5698]] adopt
[[ 0.1658]] adopt
[[ 0.6239]] adopt
[[ 0.3946]] adopt
[[ 1.1959]] adopt
[[ 0.7826]] adopt
[[ 0.5175]] adopt
[[ 0.4206]] adopt
[[ 1.2968]] adopt
[[ 1.8465]] adopt
[[ 2.0677]] adopt
[[ 0.7312]] adopt
[[ 1.1792]] adopt
[[ 1.1065]] adopt
[[ 0.5507]] adopt
[[ 0.5761]] adopt
[[ 0.3778]] adopt
[[ 0.4757]] adopt
[[ 0.4905]] adopt
[[ 0.7884]] adopt
[[ 10.9971]] reject outlier [ 4.1982  9.3807  1.8188  2.4793  0.4692]
[[ 2.6828]] adopt
[[ 0.4777]] adopt
[[ 1.192]] adopt
[[ 0.5871]] adopt
[[ 0.2016]] adopt
[[ 0.2123]] adopt
[[ 0.2528]] adopt
[[ 1.0433]] adopt
[[ 1.1198]] adopt
[[ 0.3795]] adopt
[[ 1.2318]] adopt
[[ 0.7081]] adopt
[[ 1.1907]] adopt
[[ 0.4682]] adopt
[[ 1.1046]] adopt
[[ 2.3368]] adopt
[[ 1.3612]] adopt
[[ 0.4896]] adopt
[[ 1.0163]] adopt
[[ 1.1983]] adopt
[[ 0.9416]] adopt
[[ 0.5743]] adopt
[[ 0.2665]] adopt outlier [ 4.2161 -2.4717 -0.9323  1.1813  3.1867]
[[ 1.417]] adopt
```

```
[[ 0.7085]] adopt
[[ 1.8776]] adopt
[[ 0.5421]] adopt
[[ 2.0644]] adopt outlier [ 0.  0.  0.  0.  0.]
[[ 0.7624]] adopt
[[ 0.3729]] adopt
[[ 2.2555]] adopt
[[ 0.6143]] adopt
[[ 0.5225]] adopt
[[ 0.6402]] adopt
[[ 0.3889]] adopt
[[ 0.2515]] adopt
[[ 0.6217]] adopt
[[ 0.3541]] adopt
[[ 0.5172]] adopt
[[ 1.1818]] adopt
[[ 1.0751]] adopt
[[ 0.1588]] adopt
[[ 0.7811]] adopt
[[ 0.2234]] adopt
[[ 1.6243]] adopt
[[ 1.7146]] adopt
[[ 0.5418]] adopt
[[ 0.2524]] adopt
[[ 0.8399]] adopt
[[ 0.5514]] adopt
[[ 0.3784]] adopt
[[ 0.6476]] 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:

```
[[ 0.8476]] adopt
[[ 0.7071]] adopt
[[ 0.3047]] adopt
[[ 11.3573]] reject outlier [ 5.4478  7.4584  4.6565 -1.2107  2.3139]
[[ 1.341]] adopt
[[ 1.9066]] adopt
[[ 0.7698]] adopt
```

```
[[ 0.7557]] adopt
[[ 1.2602]] adopt
[[ 0.9915]] adopt
[[ 0.6623]] adopt
[[ 0.2717]] adopt
[[ 1.246]] adopt
[[ 0.0786]] adopt
[[ 1.0179]] adopt
[[ 1.8234]] adopt
[[ 0.6829]] adopt
[[ 1.4242]] adopt
[[ 2.1964]] adopt
[[ 0.9775]] adopt
[[ 1.1807]] adopt
[[ 0.4036]] adopt
[[ 2.176]] adopt
[[ 1.2844]] adopt
[[ 0.9243]] adopt
[[ 0.3126]] adopt outlier [ 4.1844 -2.2948 -0.1964  0.6355  2.8791]
[[ 1.2655]] adopt
[[ 0.5789]] adopt
[[ 0.1768]] adopt
[[ 0.8152]] adopt
[[ 0.6597]] adopt
[[ 1.2051]] adopt
[[ 0.7177]] adopt
[[ 0.8773]] adopt
[[ 0.4055]] adopt
[[ 1.2283]] adopt
[[ 1.8612]] adopt
[[ 2.0214]] adopt
[[ 0.7052]] adopt
[[ 1.1316]] adopt
[[ 1.1115]] adopt
[[ 0.8694]] adopt
[[ 0.7001]] adopt
[[ 0.3946]] adopt
[[ 0.755]] adopt
[[ 0.6461]] adopt
[[ 1.1259]] adopt
[[ 15.5219]] reject outlier [ 4.1982  9.3807  1.8188  2.4793  0.4692]
[[ 2.4318]] adopt
[[ 0.5824]] adopt
[[ 1.6251]] adopt
[[ 0.6197]] adopt
[[ 0.2325]] adopt
[[ 0.2228]] adopt
[[ 0.3585]] adopt
[[ 0.9486]] adopt
[[ 1.1306]] adopt
[[ 0.443]] adopt
[[ 1.145]] adopt
```

```
[[ 0.6724]] adopt
[[ 1.4917]] adopt
[[ 0.467]] adopt
[[ 0.9968]] adopt
[[ 2.3605]] adopt
[[ 1.3821]] adopt
[[ 0.4926]] adopt
[[ 1.0078]] adopt
[[ 1.3249]] adopt
[[ 0.9602]] adopt
[[ 0.6056]] adopt
[[ 0.3633]] adopt outlier [ 4.2161 -2.4717 -0.9323  1.1813  3.1867]
[[ 1.5871]] adopt
[[ 0.8107]] adopt
[[ 1.7905]] adopt
[[ 0.5561]] adopt
[[ 2.4056]] adopt outlier [ 0.  0.  0.  0.  0.]
[[ 0.7659]] adopt
[[ 0.5115]] adopt
[[ 2.3682]] adopt
[[ 0.6867]] adopt
[[ 0.537]] adopt
[[ 0.6561]] adopt
[[ 0.3761]] adopt
[[ 0.4102]] adopt
[[ 0.6532]] adopt
[[ 0.4717]] adopt
[[ 0.5173]] adopt
[[ 1.1129]] adopt
[[ 1.1364]] adopt
[[ 0.1689]] adopt
[[ 1.0978]] adopt
[[ 0.2231]] adopt
[[ 1.8147]] adopt
[[ 1.8181]] adopt
[[ 0.5258]] adopt
[[ 0.2495]] adopt
[[ 0.804]] adopt
[[ 0.6158]] adopt
[[ 0.4432]] adopt
[[ 0.6129]] 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.