Метод SVD

Метод SVD (Singular value decomposition) может использоваться для снижения размерности исходной матрицы предикторов $X_{p\times n}$ на основе ее сингулярного разложения т.е декомпозици по сингулярному значению. Все матрицы имеют SVD, что делает его более стабильным, чем многие другие методы,например, такие как собственное разложение.

Разложение по сингулярному значению

Разложение по сингулярному значению, или сокращенно SVD, - это метод разложения матрицы, предназначенный для сведения матрицы к ее составным частям, чтобы упростить некоторые последующие вычисления матрицы. Рассмотрим действительную матрицу предикторов $X_{p\times n}$. Ее можно представить следующим образом:

$$X = U\Sigma V^T (1)$$

Где X - действительная матрица $p\times n$, которую мы хотим разложить, U - матрица $p\times p$, Σ (часто представляемая заглавной греческой буквой Sigma) - диагональная матрица $p\times n$, а V^T - транспонирование матрицы $n\times n$.

Разложение по сингулярным значениям является основным моментом линейной алгебры. Диагональные значения в Σ-матрице называются сингулярными значениями исходной матрицы X. Столбцы матрицы U называются лево-сингулярными векторами, а столбцы V называются правосингулярными векторами X.

SVD рассчитывается с помощью итерационных численных методов. Мы не будем вдаваться в подробности этих методов. Каждая прямоугольная матрица имеет разложение по сингулярным значениям, хотя результирующие матрицы могут содержать комплексные числа, а ограничения арифметики с плавающей запятой могут привести к тому, что некоторые матрицы не будут аккуратно разлагаться.

Разложение по сингулярным значениям (SVD) предоставляет способ разложить матрицу на сингулярные векторы и сингулярные значения. SVD позволяет нам обнаруживать некоторую информацию того же вида, что и собственное разложение. Тем не менее, SVD является более общим методом.

SVD широко используется как при вычислении других матричных операций, таких как обратная матрица, так и в качестве метода сокращения данных в машинном обучении. SVD также может быть использо-

ван в линейной регрессии наименьших квадратов, сжатии изображений и шумоподавлении данных.Нанесение SVD на матрицу похоже на рентгеновское зрение, вскрывающее ее внутренню структуру.

Рассчитать разложение по сингулярному значению

SVD можно рассчитать, вызвав функцию svd ().

Функция принимает матрицу X и возвращает элементы U, Σ и V^T. Диагональная матрица Сигмы возвращается как вектор сингулярных значений. Матрица V возвращается в транспонированной форме.

Рассмотрим пример код python приведен ниже:

```
‘‘‘
# Singular-value decomposition
import numpy as np
from scipy.linalg import svd
from pandas import Series, DataFrame
import pandas as pd
import openpyxl
import xlrd

#РЕАЛЬНАЯ МАТРИЦА
data_regru =pd.ExcelFile('Region_Data.xlsx')
tabledat =data_regru.parse('dat01')

X =DataFrame(tabledat, columns=['P10','P11','P14'])
X=np.array (X)
X=X[0:5,:]

#нормирование значений элементов массива X
SUMX=X[:,0] +X[:,1] + X[:,2]
X[:,0] =X[:,0] /SUMX
X[:,1] =X[:,1] /SUMX
X[:,2] =X[:,2] /SUMX

print('Матрица X 5x 3')
print(X)

#  Вычисление SVD
U, s, VT = svd(X)
```

```
print (' Матрица U 5x5')
print(U)
print (' Вектор сигма 1x3 ')
print(s)
print ('Матрица VT 3x3 ')
print(VT)
```

Рузультаты вычилений:

Исходная матрица X размерности 5×3

[[1.33302589e-01 6.36527182e-01 2.30170229e-01]
[8.61978207e-04 7.19008264e-01 2.80129757e-01]
[1.03605170e-02 9.28260924e-01 6.13785594e-02]
[1.14442305e-02 6.63935713e-01 3.24620056e-01]
[5.72273019e-03 8.99575505e-01 9.47017645e-02]]

Матрица U размерности 5×5

[[-0.37801941 -0.31560143 -0.8678742 -0.06491401 0.00880982]
[-0.42664927 -0.35018496 0.36395208 -0.71122063 -0.23884095]
[-0.51203292 0.54959354 -0.00118095 0.2421771 -0.61409941]
[-0.40273896 -0.54960288 0.3278932 0.64694345 0.09842881]
[-0.50069487 0.41671314 0.0825707 -0.11298591 0.74570173]]

Вектор Σ размерности 1×3

[1.79066669 0.28696379 0.1123642]

Матрица V^T размерности 3×3

[[-0.03548289 -0.97197862 -0.23237582]
[-0.14142311 0.2350659 -0.96163586]
[-0.98931313 0.00125831 0.14580106]]

Восстановить Матрицу из SVD

Исходная матрица может быть восстановлена из элементов U, Σ и V^T.

Элементы U, s и V, возвращаемые из svd (), не могут быть умножены напрямую.

Вектор s должен быть преобразован в диагональную матрицу с помощью функции diag (). По умолчанию эта функция создаст квадратную матрицу размером $p \times p$ относительно нашей исходной матрицы. Это вызывает проблему, поскольку размеры матриц не соответствуют правилам умножения матриц, где число столбцов в матрице должно соответствовать количеству строк в последующей матрице. После создания квадрат-

ной диагональной матрицы Σ размеры матриц соответствуют исходной матрице $p \times n$, которую мы разлагаем, следующим образом:

$$U(p \times p).\Sigma(p \times p).V^T(n \times n)(2)$$

Где, на самом деле, мы требуем:

$$U(p \times p).\Sigma(p \times n).V^T(n \times n)(3)$$

Мы можем достичь этого путем создания новой Σ-матрицы со всеми нулевыми значениями, которая равна $p \times n$ (например, больше строк), и заполнить первую $p \times p$ часть матрицы квадратной диагональной матрицей, вычисленной с помощью diag ().

код python реализующий (3):

```
‘‘‘
# Singular-value decomposition
import numpy as np
from numpy import zeros
from numpy import diag
from numpy import dot
from scipy.linalg import svd
from pandas import Series, DataFrame
import pandas as pd
import openpyxl
import xlrd

#РЕАЛЬНАЯ МАТРИЦА X
data_regru =pd.ExcelFile('Region_Data.xlsx')
tabledat =data_regru.parse('dat01')

X =DataFrame(tabledat, columns=['P10','P11','P14'])
X=np.array (X)
X=X[0:5,:]

#нормирование значений элементов массива X
SUMX=X[:,0] +X[:,1] + X[:,2]
X[:,0] =X[:,0] /SUMX
X[:,1] =X[:,1] /SUMX
```

```
X[:,2] =X[:,2] /SUMX

print('Матрица X 5x 3')
print(X)

#  Вычисление SVD
U, s, VT = svd(X)
print (' Матрица U 5x5')
print(U)
print (' Вектор сигма 1x3 ')
print(s)
print ('Матрица VT 3x3 ')
print(VT)

#Восстановление матрицы X из сингулярного разложения
# create m x n Sigma matrix
Sigma = zeros((X.shape[0], X.shape[1]))
# populate Sigma with n x n diagonal matrix
Sigma[:X.shape[1], :X.shape[1]] = diag(s)
# reconstruct matrix
B = U.dot(Sigma.dot(VT))

print(' Восстановленная из SVD матрица X 5x 3')
print(B)
```

Рузультаты вычилений:
Исходная матрица X размерности 5×3
[[1.33302589e-01 6.36527182e-01 2.30170229e-01]
[8.61978207e-04 7.19008264e-01 2.80129757e-01]
[1.03605170e-02 9.28260924e-01 6.13785594e-02]
[1.14442305e-02 6.63935713e-01 3.24620056e-01]
[5.72273019e-03 8.99575505e-01 9.47017645e-02]]
Восстановленная из SVD матрица B размерности 5×3
[[1.33302589e-01 6.36527182e-01 2.30170229e-01]
[8.61978207e-04 7.19008264e-01 2.80129757e-01]
[1.03605170e-02 9.28260924e-01 6.13785594e-02]
[1.14442305e-02 6.63935713e-01 3.24620056e-01]

[5.72273019e-03 8.99575505e-01 9.47017645e-02]]

SVD для псевдообратного обращения

Псевдообращение - это обобщение обратной матрицы для квадратных матриц на прямоугольные матрицы, где число строк и столбцов не равно. Он также называется инверсией Мура-Пенроуза после двух независимых исследователей метода или Обобщенной инверсией. Инверсия матриц не определена для матриц, которые не являются квадратными.Когда A имеет больше столбцов, чем строк, то решение линейного уравнения с использованием псевдообратного представления дает одно из многих возможных решений. Псевдообращение обозначается как X^+ , где X - это инвертируемая матрица, а $+$ - верхний индекс.

Псевдообращение вычисляется с использованием сингулярного разложения A:

$$X^+ = VD^+U^T(4)$$

Где X^+ - псевдообратная, D^+ - псевдообратная диагональная матрица $Sigma$, а U^T - транспонирование U.

Мы можем получить U и V из операции SVD.

$$X = U.\Sigma.V^T(5)$$

D^+ можно рассчитать путем создания диагональной матрицы из Σ, вычисления обратной величины каждого ненулевого элемента в Σ и выполнения транспонирования, если исходная матрица была прямоугольной.

Псевдообращение обеспечивает один из способов решения уравнения линейной регрессии, в частности, когда строк больше, чем столбцов, что часто имеет место. NumPy предоставляет функцию pinv () для вычисления псевдообращения прямоугольной матрицы.

Можно вычислить псевдообращение вручную через SVD и сравнить результаты с функцией pinv (). Сначала мы должны рассчитать SVD. Далее мы должны вычислить обратную величину каждого значения в массиве s. Затем массив s можно преобразовать в диагональную матрицу с добавленным рядом нулей, чтобы сделать его прямоугольным. Наконец, мы можем вычислить псевдообратное по элементам.

Конкретная реализация:

$$X^+ = V.D^+.U^V(6)$$

код python, реализующий оба подхода

```
‘‘‘
#ПСЕВДООБРАЩЕНИЕ МАТРИЦЫ X pinv ()
# calculate pseudoinverse
print('Псевдообращение X функкция pinv ()')
C = pinv(X)
print(C)

#ПСЕВДООБРАЩЕНИЕ МАТРИЦЫ X SVD
# calculate svd
U, s, VT = svd(X)
# reciprocals of s
d = 1.0 / s
# create m x n D matrix
D = zeros(X.shape)
# populate D with n x n diagonal matrix
D[:X.shape[1], :X.shape[1]] = diag(d)
# calculate pseudoinverse
E = VT.T.dot(D.T).dot(U.T)
print('Псевдообращение X  SVD')
print(E)
‘‘‘
```

Результаты вычилений:

Псевдообращение функкция pinv ()

[[7.80424501 -3.02338988 -0.25030993 -2.60810422 -0.92244099]

[-0.06305332 -0.05119134 0.72811821 -0.22792657 0.61405303]

[-0.01947407 1.70111636 -1.77681224 2.31948786 -1.22431791]]

Псевдообращение с помощью SVD

[[7.80424501 -3.02338988 -0.25030993 -2.60810422 -0.92244099]

[-0.06305332 -0.05119134 0.72811821 -0.22792657 0.61405303]

[-0.01947407 1.70111636 -1.77681224 2.31948786 -1.22431791]]

SVD для уменьшения размерности

Популярное применение SVD для уменьшения размерности.

Данные с большим количеством объектов, таких как больше объектов (столбцов), чем наблюдений (строк), могут быть уменьшены до меньшего подмножества объектов, которые наиболее актуальны для проблемы прогнозирования. В результате получается матрица с более низким рангом, которая, как говорят, аппроксимирует исходную матрицу.

Для этого мы можем выполнить SVD-операцию с исходными данными и выбрать верхние k самых больших значений в Σ. Эти столбцы можно выбрать из Σ, а строки - из V^T.

Приближенный B исходного вектора X может быть восстановлен.

$$B = U.\Sigma k.V^T k(7)$$

В обработке естественного языка этот подход может использоваться для матриц вхождений слов или частот слов в документах и называется скрытым семантическим анализом или скрытым семантическим индексированием.

На практике мы можем сохранить и работать с описательным подмножеством данных, называемым T. Это плотная сводка матрицы или проекции.

$$T = U.\Sigma k(8)$$

Кроме того, это преобразование может быть вычислено и применено к исходной матрице X, а также к другим аналогичным матрицам.

$$T = V^T k.X(9)$$

Пример ниже демонстрирует сокращение данных с SVD.

Сначала определяется матрица X размерностсти 5×3. SVD рассчитывается и выбираются только первые две функции. Элементы рекомбинированы для точного воспроизведения исходной матрицы. Наконец, преобразование рассчитывается двумя разными способами.

Код python, реализующий данный подход

```
‘‘‘
 import numpy as np
from numpy import zeros
from numpy import diag
from numpy import dot
from numpy.linalg import pinv
from scipy.linalg import svd
from pandas import Series, DataFrame
import pandas as pd
import openpyxl
import xlrd
```

```
#РЕАЛЬНАЯ МАТРИЦА X
data_regru =pd.ExcelFile('Region_Data.xlsx')
tabledat =data_regru.parse('dat01')

X =DataFrame(tabledat, columns=['P10','P11','P14'])
X=np.array (X)
X=X[0:5,:]

#нормирование значений элементов массива X
SUMX=X[:,0] +X[:,1] + X[:,2]
X[:,0] =X[:,0] /SUMX
X[:,1] =X[:,1] /SUMX
X[:,2] =X[:,2] /SUMX

print('Матрица X 5x 3')
print(X)


# Singular-value decomposition
U, s, VT = svd(X)
# create m x n Sigma matrix
Sigma = zeros((X.shape[0], X.shape[1]))
# populate Sigma with n x n diagonal matrix
Sigma[:X.shape[1], :X.shape[1]] = diag(s)
# select
n_elements = 2
Sigma = Sigma[:, :n_elements]
VT = VT[:n_elements, :]
# reconstruct
B = U.dot(Sigma.dot(VT))
#print(B)
# transform
T1 = U.dot(Sigma)
print('сокращенная матрица размерности 5x2 1 способ')
print(T1)
T2 = X.dot(VT.T)
print('сокращенная матрица размерности 5x2 2 способ')
```

```
print(T2)
```

При выполнении примера сначала печатается определенная матрица, а затем восстановленное приближение, за которым следуют два эквивалентных преобразования исходной матрицы.

Результаты вычилений:

сокращенная матрица размерности 5x2 : 1 способ

[[-0.67690677 -0.09056618]

[-0.76398663 -0.1004904]

[-0.91688029 0.15771344]

[-0.72117125 -0.15771612]

[-0.89657762 0.11958158]]

сокращенная матрица размерности 5x2: 2 способ

[[-0.67690677 -0.09056618]

[-0.76398663 -0.1004904]

[-0.91688029 0.15771344]

[-0.72117125 -0.15771612]

[-0.89657762 0.11958158]]

Scikit-learn предоставляет класс TruncatedSVD, который напрямую реализует эту возможность. Может быть создан класс TruncatedSVD, в котором необходимо указать количество желаемых функций или компонентов, например, например. 2. После создания можно подогнать преобразование (например, вычислите $V^T k$), вызвав функцию fit (), затем применить ее к исходной матрице, вызвав функцию transform (). Результатом является преобразование X, названное T выше.

В приведенном ниже примере демонстрируется класс TruncatedSVD.

```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=2)
svd.fit(X)
result = svd.transform(X)
print(result)
```

Результаты вычилений:

[[0.67690677 0.09056618]
[0.76398663 0.1004904]
[0.91688029 -0.15771344]
[0.72117125 0.15771612]
[0.89657762 -0.11958158]]

Мы видим, что значения соответствуют значениям, вычисленным вручную выше, за исключением знака некоторых значений. Мы можем ожидать некоторую нестабильность, когда дело доходит до знака, учитывая природу выполняемых вычислений и различия в используемых библиотеках и методах. Эта нестабильность знака не должна быть проблемой на практике, пока преобразование обучено для повторного использования.

Пример реализации метода SVD для снижения размерности на *Python* и *R*.

В качестве исходных данных используем информацию об основных социально-экономических показателях российских регионов за 2018 год. Исходная матрица предикторов имеет размерность 81x16 в качестве выходной переменной используется номинальная переменная тип экономики региона. В реализации на Python переменной "тип экономики": A-аграрный; D-добывающий, P- промышленный; а при реализации на R добовляется значение: T-торговый.

Код на Python

Здесь из исходной таблицы выбираются три столбца P10,P11,P14 , а выходная переменная P18 принимает три значения A,D,P

```python
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
from pandas import Series, DataFrame
import pandas as pd
from sklearn.decomposition import TruncatedSVD

result =pd.read_table('dan04.txt',sep='\s+')
print(result)

y=result['P18']
```

```
X =DataFrame(result, columns=['P10','P11','P14'])
y=np.array(y)
X=np.array (X)

SUMX=X[:,0] +X[:,1] + X[:,2]
X[:,0] =X[:,0] /SUMX
X[:,1] =X[:,1] /SUMX
X[:,2] =X[:,2] /SUMX

#описательные статистики
print('Описательные статистики предикторов')
for num in range(3) :
  print('Номер переменной: %.0f' %num)
  print('число элентов массива =',len(X[:,num]))
  print('среднее значение =',np.mean(X[:,num]))
  print('стандартное отклонение =',np.std(X[:,num]))
  print('мин мах =',np.min(X[:,num]), np.max(X[:,num]))

#построение исходного графика 3 D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
xs = X[:,0]
ys = X[:,1]
zs =X[:,2]
ax.scatter(xs, ys, zs,c=y, cmap=plt.cm.nipy_spectral,
 edgecolor='k', s=100)
ax.set_title("Исходное пространство предикторов"
 ,fontsize=14, fontweight='bold')
ax.set_xlabel('P10' ,fontsize=12, fontweight='bold')
ax.set_ylabel('P11' ,fontsize=12, fontweight='bold')
ax.set_zlabel('P14' ,fontsize=12, fontweight='bold')
plt.show()

 # Преобразование данных, уменьшающее размерность до 2
svd = TruncatedSVD(n_components=2,  algorithm = 'randomized',
 n_iter=7, random_state=42)
X =svd.fit_transform(X )
print('Компоненты')
```

```
print(svd.components_)
print('Величина объяненной дисперсии')
print(svd.explained_variance_)
print('Процент отклонения, объясняемый
каждым из выбранных компонентов.')
print(svd.explained_variance_ratio_)
print('Сингулярные значения')
print(svd.singular_values_)

y = np.choose(y, [1, 2, 0]).astype(np.float)
plt.clf()
plt.cla()
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.nipy_spectral,
edgecolor='k',s=100)
plt.title("Снижение размерности методом SVD" ,fontsize=14,
 fontweight='bold')
plt.xlabel("SVD1" ,fontsize=12, fontweight='bold')
plt.ylabel("SVD2" ,fontsize=12, fontweight='bold')
plt.show()
```

Графические результаты:

Код R

Здесь из исходной таблицы выбираются четыре столбца P10,P11,P14,P16 , а выходная переменная P18 принимает четыре значения A,D,P,T

```r
library(gmodels)
library(lattice)
library(vegan)
library(grDevices)
library(reshape2)
library(ggplot2)
library(ggthemes)

load("xRegion.RData")

#ФОРМИРОВАНИЕ data.frame ОСНОВНЫЕ
```

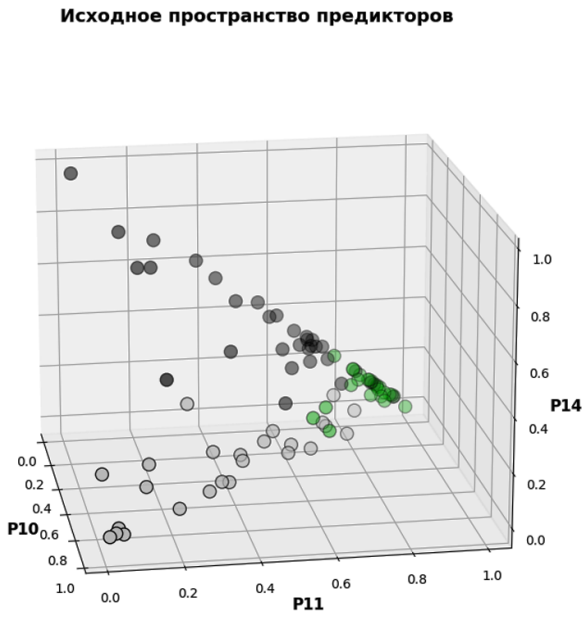

Рис. 1: Данные в исходном трехмерном пространстве

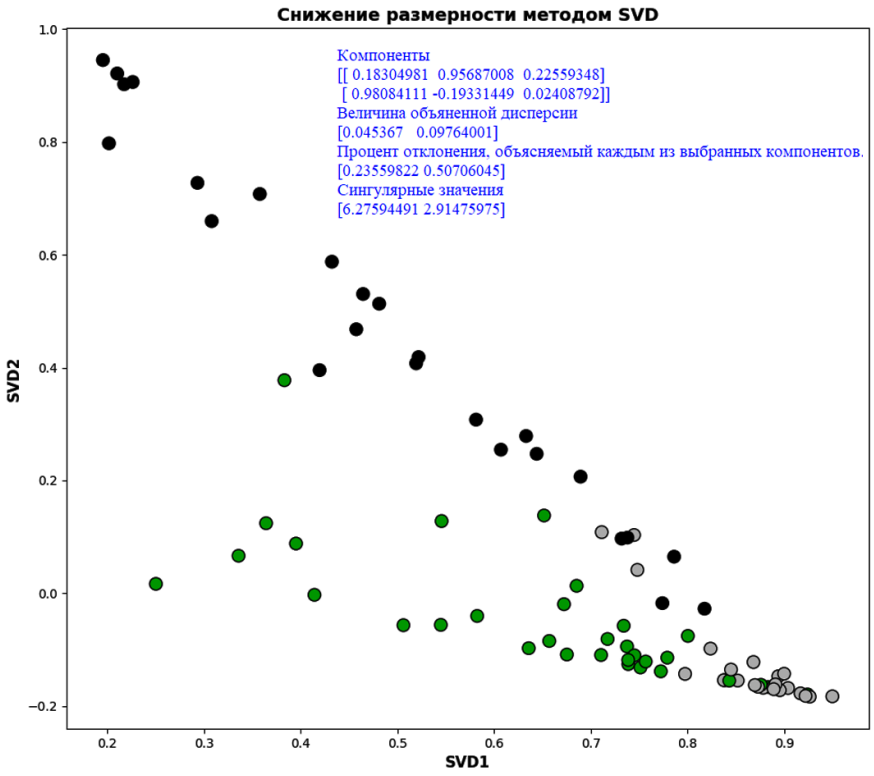

Рис. 2: Данные в редуцированном двухмерном пространстве

```
№СОЦИАЛЬНО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ РЕГИОНОВ РФ
xreg <-data.frame(P1=x$Pok_001,P2=x$Pok_002,
P3=x$Pok_003,P4=x$Pok_004,
P5=x$Pok_005,P6=x$Pok_006,P7=x$Pok_007,
P8=x$Pok_008,P9=x$Pok_009,P10=x$Pok_010,
P11=x$Pok_011,P12=x$Pok_012,P13=x$Pok_013,
P14=x$Pok_014,P15=x$Pok_015,P16=x$Pok_016,
P17=x$Pok_017,P18=x$Pok_018)
#xreg
#Формирование фактора тип экономики региона
xreg_f <- as.factor(x$Pok_018)

#выбор исследуемого подмножества показателей регионов
w=c(10,11,14,16)
reg.ekon <-xreg[,w]
sumreg <-rowSums(reg.ekon)
xreg_nor <-reg.ekon/sumreg
Y <-xreg_nor

#Метод SVD
library(svd)
X=Y
X <- as.matrix(X)
y.svd1 <-propack.svd(X, neig = 2, opts = list())
y.svd1$u

y_propack.svd <-data.frame(y1=y.svd1$u[,1],y2=y.svd1$u[,2])
y_propack.svd <-cbind(y_propack.svd,xreg_f)

#График регионов по классам в двухмерном пространстве svd 1
x11()
g4 <-ggplot(data= y_propack.svd, aes(x=y1,y=y2,colour = xreg_f),
colors = "Accent")
g4 <-g4 +geom_point(size=4)
g4 <-g4 +labs(title ="Регионы в пространсте двух компонент svd 1",
x="Компонента 1", y="Компонента  2",colour = "Классы")
g4
```

```
y.svd2 <-trlan.svd(X, neig = 2, opts = list(), lambda = NULL, U = NULL)
y.svd2$u

y_propack.svd2 <-data.frame(y1=y.svd2$u[,1],y2=y.svd2$u[,2])
y_propack.svd2 <-cbind(y_propack.svd2,xreg_f)

#График регионов по классам в двухмерном пространстве svd 2
x11()
g4 <-ggplot(data= y_propack.svd2, aes(x=y1,y=y2,colour = xreg_f),
colors = "Accent")
g4 <-g4 +geom_point(size=4)
g4 <-g4 +labs(title ="Регионы в пространсте двух компонент svd 2",
x="Компонента 1", y="Компонента  2",colour = "Классы")
g4
```

  ‘‘‘

Графические результаты:

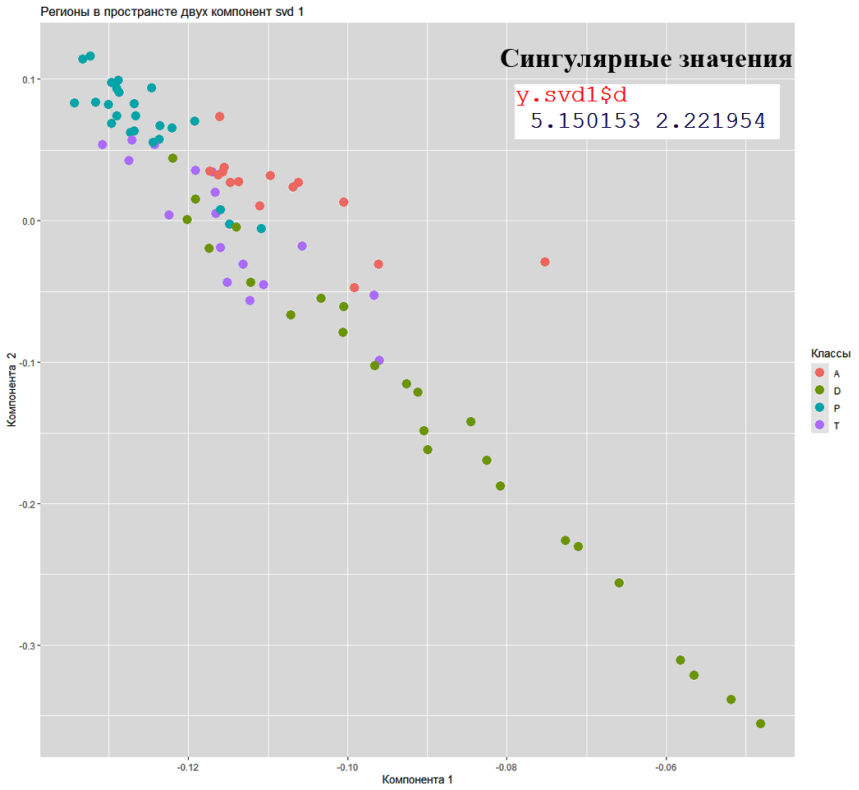

Рис. 3: Данные в редуцированном двухмерном пространстве 1

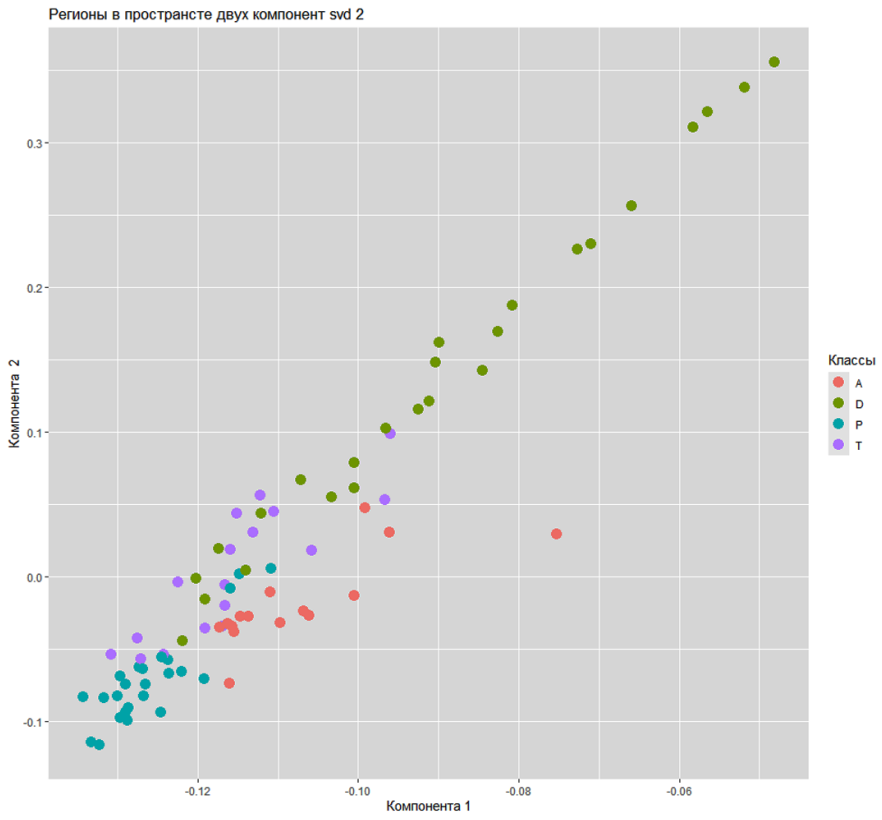

Рис. 4: Данные в редуцированном двухмерном пространстве 2