Этапы лабораторной работы

Задание. Вариант задания – тот же, что в лабораторных работах 3 и 4

Этап 1.

1. Матрицу наблюдений делим на обучающую и контролирующую так, чтобы в каждую вошло примерно равное число наблюдений каждого класса. По обучающей выборке создаем систему нечеткого вывода одним из 3 методов

1-genfis1

2-genfis2

1-genfis3

Настраиваем эту систему, распознаем обучающую и контролирующую выборки, оцениваем процент верно распознанных объектов и время построения и настройки системы и время распознавания

2. По обучающей выборке (по 2 каким-либо классам) создать персептрон и обучить его

Протестировать персептрон по обучающей и тестирующей выборкам тех же двух классов, определив процент верно распознанных объектов (по каждому классу)

Построить графики разделяющей поверхности и визуальное представление результатов тестирования

Оценить время создания и обучения персептрона и время распознавания по персептрону

3. По обучающей выборке (по всем классам) построить конкурирующую сеть (сеть Кохонена) и обучить ее

Протестировать конкурирующую сеть по обучающей и тестирующей выборкам, определив процент верно распознанных объектов (по каждому классу)

Построить графики разделяющей поверхности и визуальное представление результатов тестирования

Оценить время создания и обучения сети Кохонена и время распознавания по сети Кохонена

Этап 2. Сокращение размерности методом главных компонент

Создание главных компонент, преобразование матрицы к главным компонентам и выполнение всех действий 1 этапа по отношению к преобразованной матрице

Этап 3. Факторный анализ. Выполнение косоугольного вращения факторов с построением матрицы координат факторов. Эта матрица рассматривается как матрица наблюдений и с ней выполняются все действия 1 этапа.

Если возможно, построение графиков biplot

Результаты оформляются в виде таблицы

Этап	Качество fis	Время создания fis	Время распознавания по fis	Качество распознавания по персептрону	Времб мя создания и обучения персептрона	Время распознавания по персептрону	Качество распознавания по сети Кохонена	Время создания и обучения по сети Кохонена	Время распознавания по сети Кохонена
Этап1									
Этап2									
Этап3									

Понижение размерности

Это процесс уменьшения анализируемого множества данных до размера, оптимального с точки зрения решаемой задачи и используемой аналитической модели.

Сокращение размерности может потребоваться когда данные избыточны в информационном плане, т.е. задачу можно решить с тем же уровнем эффективности и точности, но используя меньший объем данных. Это позволяет урезать время и вычислительные затраты на решение задачи. Другой случай связан со слишком большими вычислительными затратами, требуемыми для обработки множества данного размера. Эта ситуация типична для алгоритмов, вычислительная сложность которых экспоненциально растет с увеличением числа наблюдений (т.е. немасштабируемых). Если в первом случае достаточно просто отобрать из всего множества столько признаков и записей, сколько надо, то во втором нужно сократить исходное множество до такого объема, который

Поэтому во втором случае предъявляются очень жесткие требования по отбору данных: сокращение объема должно происходить за счет наименее ценных данных, например, сначала за счет наименее значимых признаков, затем - похожих записей и т.д., пока размерность не окажется приемлемой с точки зрения требуемого объема вычислений.

Существует несколько направлений сокращения размерности множеств данных: сокращение числа признаков (атрибутов), сокращение числа записей и сокращение числа разнообразных значений определенного признака. Наиболее эффективным является сокращение признаков, поскольку в этом случае уменьшается не только объем данных, но и размерность всей задачи. В большинстве случаев решающим фактором за или против исключения признака является его значимость. На практике, определяют значимость всех признаков, исключают все признаки, значимость которых ниже заданного порога. Также следует исключать коррелирующие признаки.

Подмножество данных, полученное в результате сокращения размерности, должно унаследовать от исходного множества столько информации, сколько необходимо для решения задачи с заданной точностью, а вычислительные и временные затраты на сокращение данных не должны обесценивать, полученные от него преимущества. Аналитическая модель, построенная на основе сокращенного множества данных, должна стать проще для обработки, реализации и понимания, чем модель, построенная на исходном множестве.

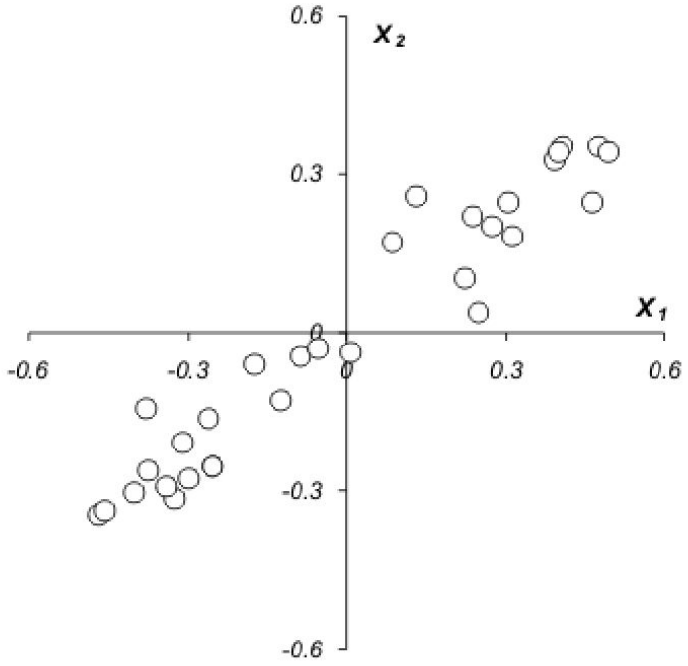
Решение о выборе метода сокращения размерности основывается на априорном знании об особенностях решаемой задачи и ожидаемых результатах, а также ограниченности временных и вычислительных ресурсов.

Метод главных компонент

Постараемся передать суть метода главных компонент, используя интуитивно-понятную геометрическую интерпретацию. Начнем с простейшего случая, когда имеются только две переменные x_1 и x_2. Такие данные легко изобразить на плоскости

Графическое представление двумерных данных

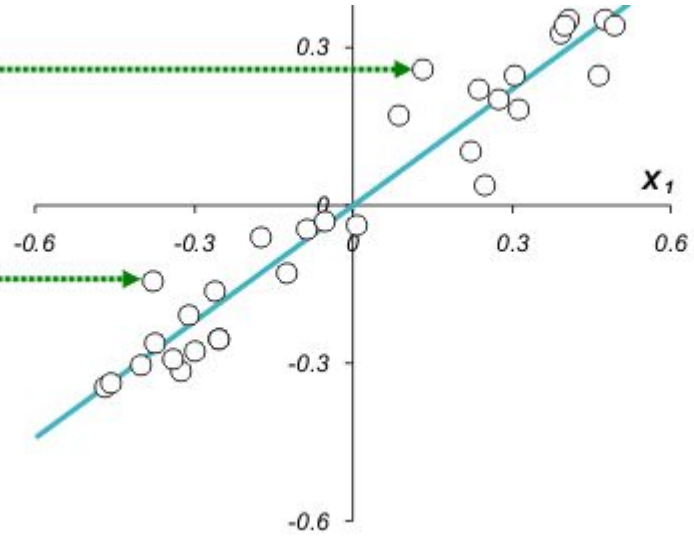
	X_1	X_2
1	0.407	0.353
2	0.475	0.355
3	-0.088	-0.045
4	0.394	0.325
5	0.274	0.202
6	0.131	0.258
7	-0.053	-0.031
8	-0.124	-0.128
9	-0.469	-0.344
10	0.088	0.171
11	-0.261	-0.162
12	0.401	0.341
13	-0.376	-0.143
14	-0.251	-0.255
15	-0.325	-0.316
16	0.464	0.248
17	-0.310	-0.207
18	0.307	0.247
19	-0.399	-0.303
20	-0.253	-0.253
21	-0.341	-0.291



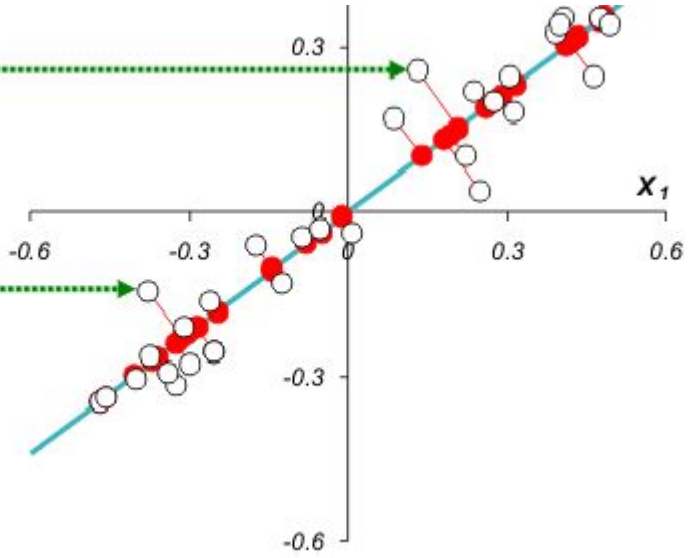
Каждой строке исходной таблицы соответствует точка на плоскости с соответствующими координатами. Они обозначены пустыми кружками на Рис.

Проведем через них прямую, так, чтобы вдоль нее происходило максимальное изменение данных. На рисунке эта прямая выделена синим цветом; она называется первой главной компонентой - PC1. Затем спроецируем все исходные точки на эту ось. Получившиеся точки закрашены красным цветом. Теперь мы можем предположить, что на самом деле все наши экспериментальные точки и должны были лежать на этой новой оси. Просто какие-то неведомые силы отклонили их от правильного, идеального положения, а мы вернули их на место. Тогда все отклонения от новой оси можно считать шумом, т.е. ненужной нам информацией. Правда, мы должны быть в этом уверены. Проверить шум ли это, или все еще важная часть данных, можно поступив с этими остатками так же, как мы поступили с исходными данными - найти в них ось максимальных изменений. Она называется второй главной компонентой (PC2). И так надо действовать, до тех пор, пока шум уже не станет действительно шумом, т.е. случайным хаотическим набором величин.

4	0.394	0.325
5	0.274	0.202
6	**0.131**	**0.258**
7	-0.053	-0.031
8	-0.124	-0.128
9	-0.469	-0.344
10	0.088	0.171
11	-0.261	-0.162
12	0.401	0.341
13	**-0.376**	**-0.143**
14	-0.251	-0.255
15	-0.325	-0.316
16	0.464	0.248
17	-0.310	-0.207
18	0.307	0.247
19	-0.399	-0.303
20	-0.253	-0.253
21	-0.341	-0.291



4	0.394	0.325
5	0.274	0.202
6	**0.131**	**0.258**
7	-0.053	-0.031
8	-0.124	-0.128
9	-0.469	-0.344
10	0.088	0.171
11	-0.261	-0.162
12	0.401	0.341
13	**-0.376**	**-0.143**
14	-0.251	-0.255
15	-0.325	-0.316
16	0.464	0.248
17	-0.310	-0.207
18	0.307	0.247
19	-0.399	-0.303
20	-0.253	-0.253
21	-0.341	-0.291



В общем, многомерном случае, процесс выделения главных компонент происходит так:

- Ищется центр облака данных, и туда переносится новое начало координат - это нулевая главная компонента (PC0)
- Выбирается направление максимального изменения данных - это первая главная компонента (PC1)
- Если данные описаны не полностью (шум велик), то выбирается еще одно направление (PC2) - перпендикулярное к первому, так чтобы описать оставшееся изменение в данных и т.д.

В результате, мы переходим от большого количества переменных к новому представлению, размерность которого значительно меньше. Часто удается упростить данные на порядки: от 1000 переменных перейти всего к двум. При этом ничего не выбрасывается - все переменные учитываются. В то же время несущественная для сути дела часть данных отделяется, превращается в шум. Найденные главные компоненты и дают нам искомые скрытые переменные, управляющие устройством данных.

Суть метода главных компонент - это существенное понижение размерности данных. Исходная матрица X заменяется двумя новыми матрицами T и P, размерность которых, A, меньше, чем число переменных (столбцов) J у исходной матрицы X

Вторая размерность - число строк сохраняется. Если декомпозиция выполнена правильно - размерность *A* выбрана верно, то матрица **T** несет в себе столько же информации, сколько ее было в начале, в матрице **X**. При этом матрица **T** меньше, и, стало быть, проще, чем **X**.

Формальное описание

Пусть имеется матрица переменных X размерностью (I×J), где I - число строк, а J - это число независимых переменных (столбцов), которых, как правило, много (J>>1). В методе главных компонент используются новые, формальные переменные t_a (a=1,…A), являющиеся линейной комбинацией исходных переменных xj (j=1,…J)

$$t_a = p_{a1}x_1 + \ldots + p_{aJ}x_J$$

С помощью этих новых переменных матрица X представляется в виде произведения двух матриц T и P -

$$\mathbf{X} = \mathbf{T}\mathbf{P}^t + \mathbf{E} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{p}_a^t + \mathbf{E}$$

Матрица T называется матрицей счетов (scores). Ее размерность - (I×A).

Матрица P называется матрицей нагрузок (loadings). Ее размерность (A×J).

E - это матрица остатков, размерностью (I×J).

Новые переменные $\mathbf{t}_a$ называются *главными компонентами* (Principal Components), поэтому и сам метод называется методом главных компонент (PCA). Число столбцов - $\mathbf{t}_a$ в матрице **T**, и $\mathbf{p}_a$ в матрице **P** - равно A, которое называется *числом главных компонент* (PC). Эта величина заведомо меньше числа переменных J и числа образцов I.

Важным свойством PCA является *ортогональность* (независимость) главных
компонент.
Поэтому матрица счетов **T** не перестраивается при увеличении числа компонент,
а к ней просто прибавляется еще один столбец - соответствующий новому направлению.
То же происходит и с матрицей нагрузок **P**.

Алгоритм

- Чаще всего для построения PCA счетов и нагрузок, используется рекуррентный алгоритм, который на каждом шаге вычисляет одну компоненту. Сначала исходная матрица X преобразуется и превращается в матрицу E_0, a=0. Далее применяют следующий алгоритм.
- 2. $\mathbf{p}^t = \mathbf{t}^t \mathbf{E}_a / \mathbf{t}^t\mathbf{t}$
  3. $\mathbf{p} = \mathbf{p} / (\mathbf{p}^t\mathbf{p})^{1/2}$
  4. $\mathbf{t} = \mathbf{E}_a \mathbf{p} / \mathbf{p}^t\mathbf{p}$

  5. Проверить сходимость, если нет, то идти на 2

После вычисления очередной (a-ой) компоненты, полагаем $\mathbf{t}_a=\mathbf{t}$ и $\mathbf{p}_a=\mathbf{p}$.

Для получения следующей компоненты надо вычи $= \mathbf{E}_a - \mathbf{t}\,\mathbf{p}^t$ и применить к ним тот же алгоритм, заменив индекс a на a+1.

После того, как построено пространство из главных компонент,
новые объекты $\mathbf{X}_{new}$ могут быть на него спроецированы,
иными словами – определены матрицы их счетов $\mathbf{T}_{new}$.
В методе PCA это делается очень просто

$$\mathbf{T}_{new} = \mathbf{X}_{new}\,\mathbf{P}$$

PCA и SVD

Метод главных компонент тесно связан с другим разложением - по сингулярным значениям, SVD. В последнем случае исходная матрица X разлагается в произведение трех матриц

$$\mathbf{X}=\mathbf{U}\mathbf{S}\mathbf{V}^t$$

Здесь **U** - матрица, образованная ортонормированными собственными векторами $\mathbf{u}_r$ матрицы $\mathbf{X}\mathbf{X}^t$, соответствующим значениям l_r:

$$\mathbf{X}\mathbf{X}^t\mathbf{u}_r = l_r\mathbf{u}_r;$$

V- матрица, образованная ортонормированными собственными векторами $\mathbf{v}_r$ матрицы $\mathbf{X}^t\mathbf{X}$;

$$\mathbf{X}^t\mathbf{X}\mathbf{v}_r = l_r\mathbf{v}_r;$$

S - положительно определенная диагональная матрица, элементами которой являются $\sigma_1 \geq \ldots \geq \sigma_R \geq 0$ равные квадратным корням из собственных значений l_r

$$\sigma_r = \sqrt{\lambda_r}$$

Связь между PCA и SVD определяется следующими простыми соотношениями

$$\mathbf{T} = \mathbf{U}\mathbf{S} \qquad \mathbf{P} = \mathbf{V}$$

Собственные векторы и собственные значения

Пусть A — это квадратная матрица. Вектор v называется собственным вектором матрицы A, если

$$Av = \lambda v,$$

где число λ называется собственным значением матрицы A. Таким образом преобразование, которое выполняет матрица A над вектором v, сводится к простому растяжению или сжатию с коэффициентом λ. Собственный вектор определяется с точностью до умножения на константу $\alpha \neq 0$, т.е. если v — собственный вектор, то и αv — тоже собственный вектор.

Собственные значения

У матрицы A размерности (N×N) не может быть больше чем N собственных значений. Они удовлетворяют характеристическому уравнению

$\det(A - \lambda I) = 0,$

являющемуся алгебраическим уравнением N-го порядка.

В частности, для матрицы 2×2 характеристическое уравнение имеет вид

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \det\begin{pmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{pmatrix} = (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} = 0$$

Собственные векторы

У матрицы A размерности (N×N) не может быть больше чем N собственных векторов, каждый из которых соответствует своему собственному значению. Для определения собственного вектора v_n нужно решить систему однородных уравнений

$$(A - \lambda_n I)\, v_n = 0.$$

Она имеет нетривиальное решение, поскольку $\det(A - \lambda_n I) = 0$.

Определение главных компонент в Matlab

- PC = princomp(X)
- [PC,SCORE,latent,tsquare] = princomp(X)

PC = princomp(X) функция предназначена для проведения анализа главных компонент многомерной случайной величины X. Входной параметр X является матрицей исходных данных. Столбцы матрицы X соответствуют признакам, строки - наблюдениям многомерной случайной величины. Функция возвращает матрицу главных компонент PC. Матрица PC является множеством собственных векторов ковариационной матрицы cov(X). Размерность матрицы PC будет равна nxn, где n - количество признаков многомерной случайной величины, или число столбцов матрицы X.

[PC,SCORE,latent,tsquare] = princomp(X) функция возвращает матрицу главных компонент PC, матрицу Z-множества данных SCORE, собственные значения latent ковариационной матрицы cov(X), вектор значений статистики T2 Хоттелинга tsquare для каждого из наблюдений.

Z-множество данных формируется проецированием матрицы наблюдений X в пространство главных компонент. Иначе говоря, это матрица наблюдений в координатах главных компонент.

Элементы вектора latent являются дисперсиями столбцов матрицы SCORE.

Статистика T2 Хоттелинга является мерой расстояния в многомерном пространстве отдельных наблюдений относительно центра группирования исходных данных.

Пример ирисов Фишера с genfis1

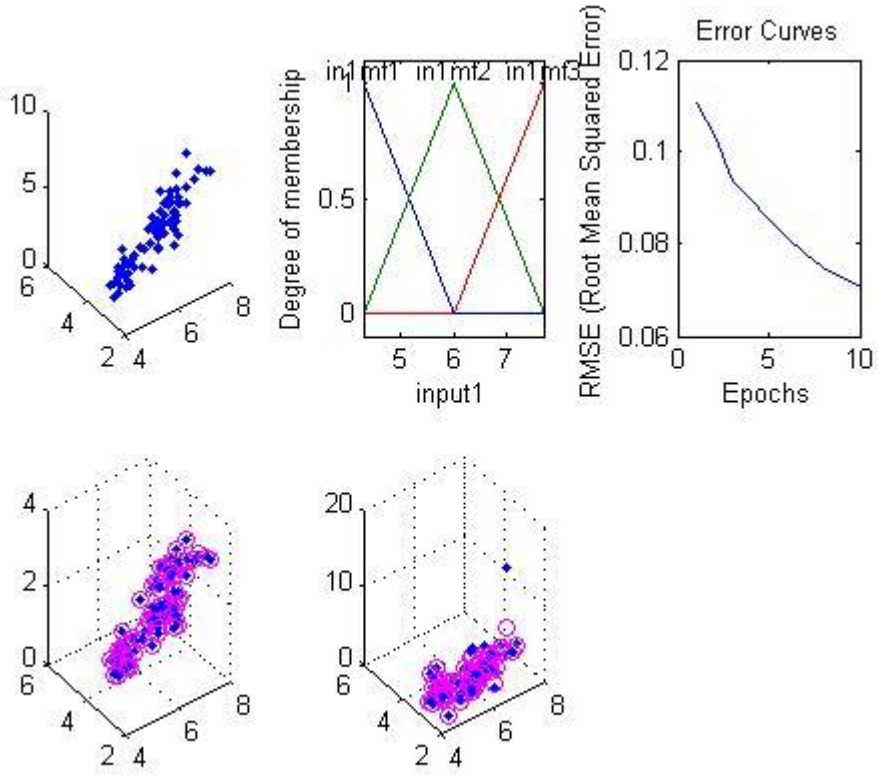
- Выполним построение гибридной сети anfis для ирисов Фишера аналогично предыдущему. Для визуализации используются только первые два признака

```
load fisheriris;
Xt1=meas(1:25,:);
Xt2=meas(51:75,:);
Xt3=meas(101:125,:);
Xt=[Xt1;Xt2;Xt3];
Yt(1:25)=1;
Yt(26:50)=2;
Yt(51:75)=3;
Xc1=meas(26:50,:);
Xc2=meas(76:100,:);
Xc3=meas(126:150,:);
Xc=[Xc1;Xc2;Xc3];
Yc(1:25)=1;
Yc(26:50)=2;
Yc(51:75)=3;
T=[Xt Yt'];
C=[Xc Yc'];
subplot(2,2,1)
plot3(Xt(:,1),Xt(:,2),Xt(:,3),' .');
```

```
grid on
fis = genfis1(T,[3],char('trimf'),char('constant'))
epoch_n = 10;  [fis,trn_error] = anfis(T, fis) ;
 writefis(fis,'gf1');
 subplot(2,2,2) plot(trn_error);xlabel('Epochs');
ylabel('RMSE (Root Mean Squared Error)');
title('Error Curves');anfis_t = evalfis(Xt, fis);
 subplot(2,2,3)
plot3(Xt(:,1),Xt(:,2),Yt,'om' );
hold on  plot3( Xt(:,1),Xt(:,2), anfist,'.b' );  hold off
grid on  anfis_c = evalfis(Xc, fis);
 subplot(2,2,4)
plot3(Xc(:,1),Xc(:,2), Yc, 'om' );
hold on
plot3(Xc(:,1),Xc(:,2),anfis_c,'.b' );
hold off
grid on
```

```
an_t=round(anfis_t);
an_c=round(anfis_c);
proc_t=length(find(an_t==Yt'))/75*100;
proc_c=length(find(an_c==Yc'))/75*100;
```

Графики


Error Curves
RMSE (Root Mean Squared Error)
Epochs
Degree of membership
input1
in1mf1
in1mf2
in1mf3

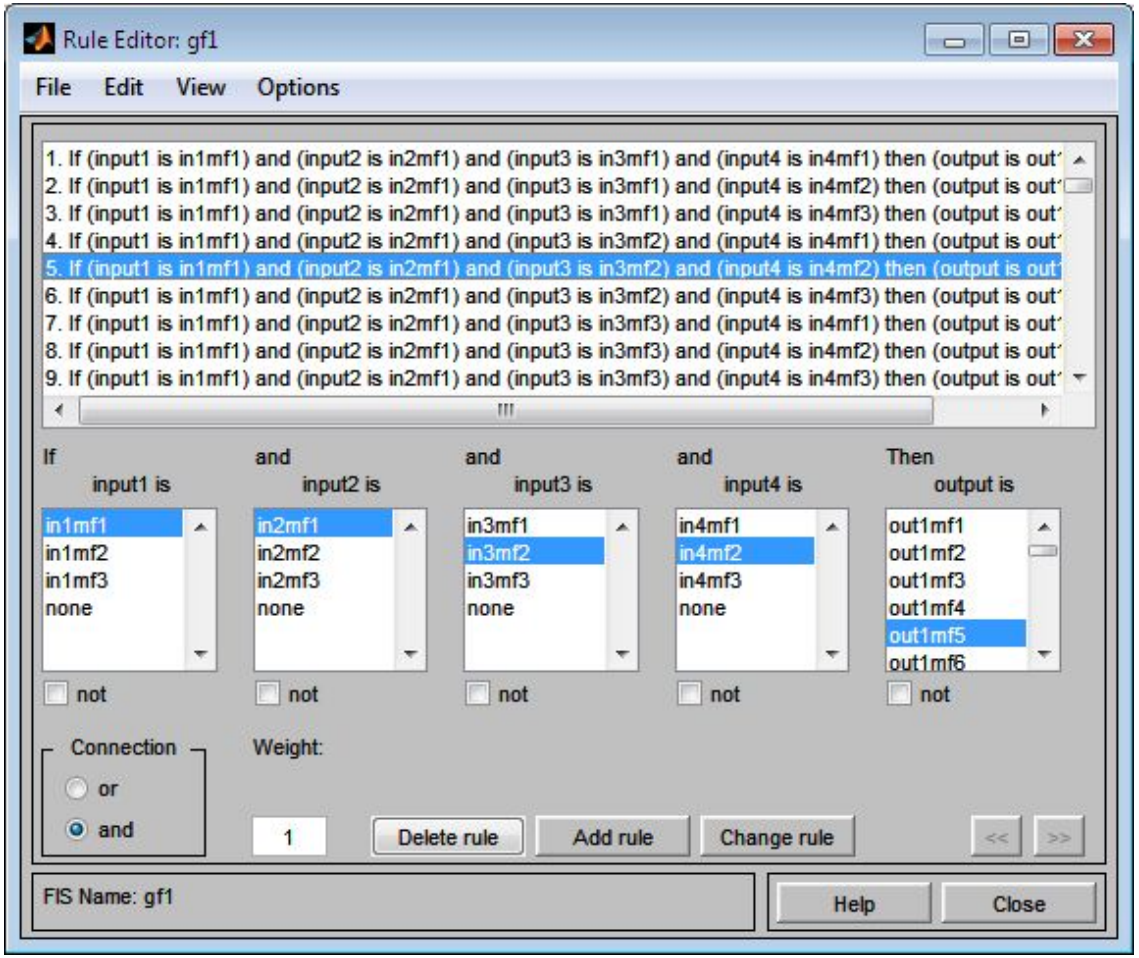
Процент распознанных

- По обучающей выборке 100%
- По тестовой выборке 86.67%

Время создания системы нечеткого вывода и ее обучения 2.4488

Время вычисления по системе нечеткого вывода 0.4497

Система нечеткого вывода



Метод главных компонент

Используем 2 первые главные компоненты

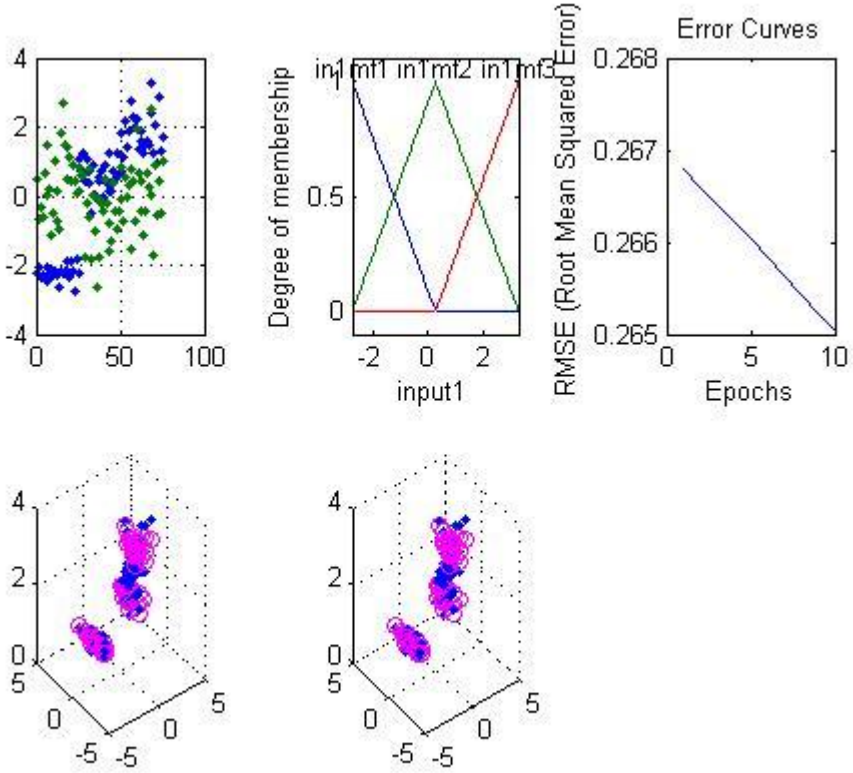
Будем стандартизировать данные путем деления каждого столбца на его стандартное отклонение.

```
load fisheriris;
stdr = std(meas);
meas = meas./repmat(stdr,150,1);
[coefs,scores,variances,t2] = princomp(meas);
Yt(1:25)=1;
Yt(26:50)=2;
Yt(51:75)=3;
Xt1=scores(1:25,1 :2);
Xt2=scores(51:75,1:2);
Xt3=scores(101:125,1:2);
Xc1=scores(26:50,1:2);
Xc2=scores(76:100,1:2);
Xc3=scores(126:150,1:2);
Xc=[Xc1;Xc2;Xc3];
Yc(1:25)=1;
Yc(26:50)=2;
Yc(51:75)=3;
tic
```

```
Xt=[Xt1;Xt2;Xt3];
T=[Xt Yt'];
C=[Xc Yc'];
subplot(2,3,1)
plot(Xt(:,1),Xt(:,2),' .');
grid on
fis = genfis1(T,[3],char('trimf'),char('constant'));
subplot(2,3,2)
 plotmf(fis,'input',1);
 epoch_n = 10;
 [fis,trn_error] = anfi    ``````````````````                                                    s(T,
fis) ;
 writefis(fis,'gfp1');
 subplot(2,3,3)
plot(trn_error);
xlabel('Epochs');
ylabel('RMSE (Root Mean Squared Error)');
title('Error Curves');
t1=toc
```

```
Xt=[Xt1;Xt2;Xt3];
T=[Xt Yt'];
C=[Xc Yc'];
subplot(2,3,1)
plot(Xt(:,1),Xt(:,2),' .');
grid on
fis = genfis1(T,[3],char('trimf'),char('constant'));
subplot(2,3,2)
 plotmf(fis,'input',1);
 epoch_n = 10;
 [fis,trn_error] = anfi  ``````````````````                                                    s(T,
fis) ;
 writefis(fis,'gfp1');
 subplot(2,3,3)
plot(trn_error);
xlabel('Epochs');
ylabel('RMSE (Root Mean Squared Error)');
title('Error Curves');
t1=toc
tic
anfis_t = evalfis(Xt, fis);
 subplot(2,3,4)
plot3(Xt(:,1),Xt(:,2),Yt,'om' );
hold on
plot3(Xt(:,1),Xt(:,2),anfis_t,'.b' );
hold off
```

Графики


4
2
0
-2
-4
0
50
100
in1mf1 in1mf2 in1mf3
Degree of membership
1
0.5
0
-2
0
2
input1
Error Curves
RMSE (Root Mean Squared Error)
0.268
0.267
0.266
0.265
0
5
10
Epochs
4
2
0
5
0
-5
-5
0
5
4
2
0
5
0
-5
-5
0
5

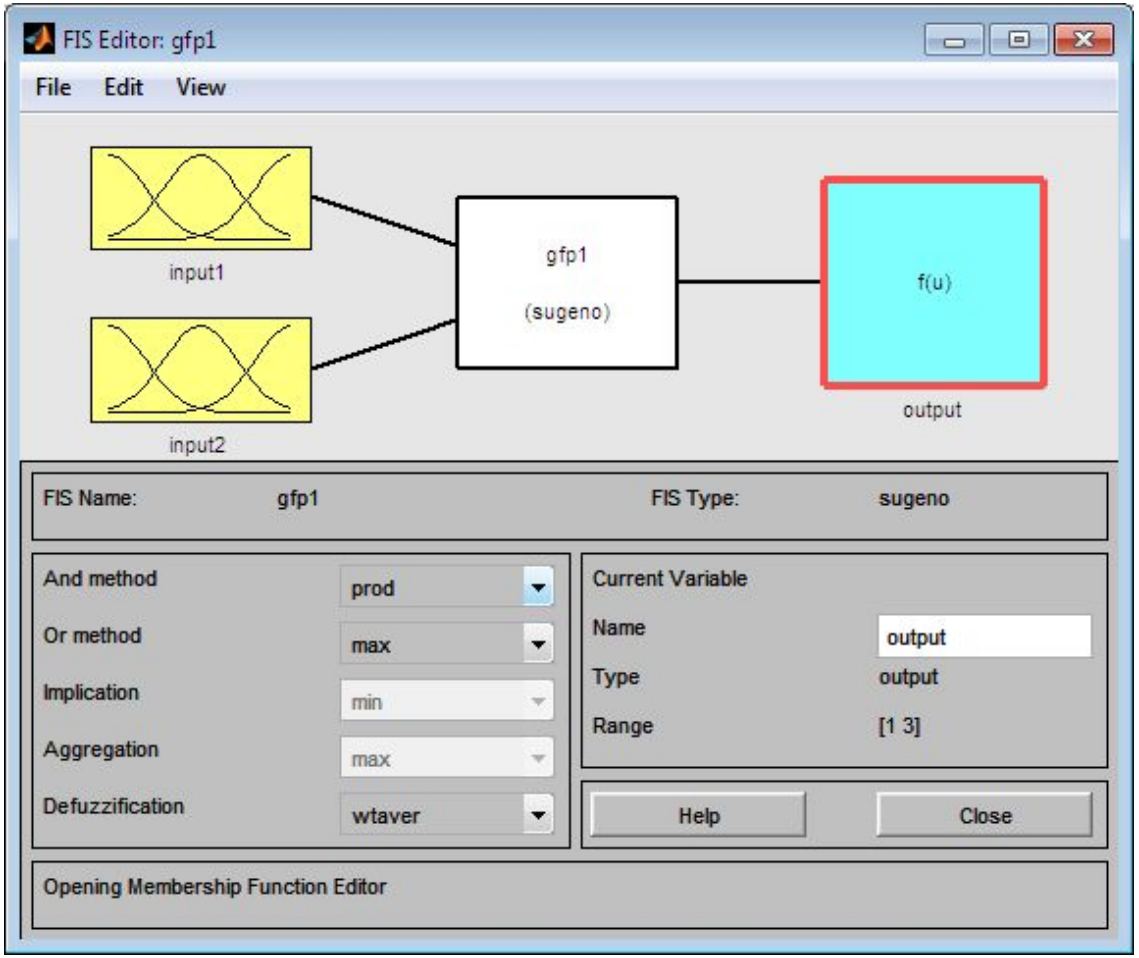
Процент распознанных

- По обучающей выборке 92%
- По тестовой выборке 92%

Время создания системы нечеткого вывода и ее обучения 1.1167

Время вычисления по системе нечеткого вывода 0.0573

Система нечеткого вывода



Факторный анализ

Многомерные данные часто содержат большое число признаков и часто эти признаки перекрываются в том смысле, что их группы могут быть зависимыми.

Например, в десятиборье каждый спортсмен участвует в 10 соревнованиях, но некоторые из них могут рассматриваться как скоростные, а другие как силовые и т.д.

Таким образом, 10 результатов спортсмена могут рассматриваться зависящими от меньшего набора из 3 или 4 типов спортивных способностей

Факторный анализ – это способ подобрать модель для многомерных данных, способную оценить именно такого рода взаимозависимости.

Модель факторного анализа

В модели факторного анализа измеренные переменные зависят от меньшего количества ненаблюдаемых (скрытых) факторов.

Так как каждый фактор может в общем случае зависеть от нескольких переменных, они называются «общими факторами».

Предполагается, что каждая переменная зависит от линейной комбинации общих факторов и коэффициенты этой линейной комбинации называются нагрузками.

Модель простого факторного анализа может быть представлена в виде

$X=\mu+\lambda f+e$ (1)

- где X- вектор наблюдений многомерной случайной величины,
- λ -матрица нагрузок простых факторов,
- μ- вектор средних значений признаков многомерной случайной величины ,
- f- вектор взаимно независимых, стандартизованных факторов,
- e - вектор независимых специфических факторов.
- Вектор значений многомерной случайной величины равен $X = \{X_1, X_2, \ldots, X_d\}$
- , где X_i - i-й признак многомерной случайной величины.

Размерность матрицы λ равна d x m, где d- число специфических факторов, или размерность многомерной случайной величины , m- число простых факторов. Элемент матрицы $\lambda_{i,j}$ называется нагрузкой i-й переменной на j-й фактор, или наоборот, нагрузкой j-го фактора на i-ю переменную.

Число элементов векторов μ,f и e равно d.

Другой формой записи модели простого факторного анализа является выражение

$Cov(X)=\lambda\lambda^T+C$ (2)

где $C=cov(e)$. C является диагональной матрицей дисперсий независимых специфических факторов. Размерность матрицы C равна $d \times d$.

Как следует из (1) или (2), значение признака многомерной случайной величины X_j может быть выражено суммой взвешенных простых факторов f и остаточного члена e. Весами простых факторов f являются элементы матрицы нагрузок λ. Количество простых факторов f должно быть меньше числа признаков многомерной случайной величины X. Это позволяет уменьшить размерность задачи с d до m. В простом факторном анализе предполагается, что простые факторы взаимно независимы и их дисперсии равны единице, а специфические факторы e_j не зависимы от какого-либо f_i, где $i=1\ldots m$,$j=1\ldots d$

Функция factoran

[lambda,psi] = factoran(X,m)

функция возвращает выходной параметр psi - вектор точечных оценок дисперсий специфических факторов, рассчитанный методом максимального правдоподобия. Число элементов вектора psi равно d.

Максимально возможное количество простых факторов определяется

зом

$$(d+m) \le (d-m)^2 .$$

.

Пример факторного анализа

Факторные нагрузки

В течение 100 недель были зарегистрированы процентные изменения цен на акции десяти компаний. Из этих десяти компаний первые 4 могут быть классифицированы как технологические, следующие 3 как финансовые, и последние 3 как розничные

Представляется разумным, что цены на акции для компаний, которые находятся в той же отрасли, могут меняться однотипно при изменении экономических условий.

Факторный анализ может дать количественное доказательство того, что компании, входящие в каждый сектор, имеют похожие изменения в цене акций в одни и те же недели.

В этом примере вначале загружаются данные и вызывается функция factoran, определяющая модель с 3 простыми факторами.

По умолчанию factoran вычисляет нагрузки с вращениями, чтобы сделать интерпретацию результатов проще. Но в этом примере пока рассматривается решение без вращений.

```
load stockreturns
[Loadings,specificVar,T,stats] = ...
  factoran(stocks,3,'rotate','none');
```

Первые два выходных аргумента factoran представляют собой расчетные нагрузки и расчетные специфические отклонения.

Каждая строка матрицы нагрузок представляет собой данные по одной из десяти компаний, а каждый столбец соответствует общему признаку. Без использования вращений трудно интерпретировать факторы в этом примере, потому что многие данные содержат довольно большие коэффициенты для двух или более факторов.

Loadings

Loadings =

0.8885	0.2367	-0.2354
0.7126	0.3862	0.0034
0.3351	0.2784	-0.0211
0.3088	0.1113	-0.1905
0.6277	-0.6643	0.1478
0.4726	-0.6383	0.0133
0.1133	-0.5416	0.0322
0.6403	0.1669	0.4960
0.2363	0.5293	0.5770
0.1105	0.1680	0.5524

Анализируя отклонения, можно видеть, что модель указывает на значительные изменения цены акций при изменении общих факторов.

specificVar

specificVar =

0.0991
0.3431
0.8097
0.8559
0.1429
0.3691
0.6928
0.3162
0.3311
0.6544

Отклонение, равное 1, указывает на то, что нет общего фактора в этой переменной, отклонение, равное 0, что переменная полностью определяется общими факторами.

Эти данные, похоже, где-то посередине.

Структура stats позволяет проверить нулевую гипотезу H_0, состоящую в том, что число простых факторов равно m.

stats.p -значение уровня значимости, соответствующее выборочной статистике stats.chisq.

В нашем примере

stats.p = 0.8144

Чтобы определить, можно ли выбрать меньшее число факторов, чем 3, построим модель с двумя простыми факторами.

Значение p для второй модели весьма показательно и отвергает гипотезу двух факторов, указывая, что более простая модель недостаточна для объяснения группировки в данных.

```
[Loadings2,specificVar2,T2,stats2] = ...
  factoran(stocks, 2,'rotate','none');

stats2.p =      3.5610e-006
```

Вращение факторов

Как показывают результаты, нагрузки, подсчитанные по факторам, не подвергающимся вращениям, имеют сложную структуру.

Цель вращения факторов состоит в нахождении параметров, при которых каждая переменная имеет только несколько больших нагрузок.

То есть каждая переменная будет зависеть от нескольких факторов, предпочтительно от одного. Это часто может сделать более простой интерпретацию того, что представляют собой факторы.

Если рассматривать каждую строку матрицы нагрузок как координаты точки в М-мерном пространстве, то каждый фактор соответствует координатной оси.

Вращение факторов эквивалентно повороту этих осей и вычислению новых нагрузок в повернутой координатной системе. Неважно, каким способом это делается. Некоторые методы оставляют оси ортогональными, в то время, как другие являются косоугольными, изменяющими угол между ними. Для нашего примера осуществим вращение нагрузок с использованием критерия promax, простого косоугольного метода.

```
[LoadingsPM,specVarPM] = factoran(stocks,3,'rotate','promax');
LoadingsPM
LoadingsPM =
    0.9452    0.1214   -0.0617
    0.7064   -0.0178    0.2058
    0.3885   -0.0994    0.0975
    0.4162   -0.0148   -0.1298
    0.1021    0.9019    0.0768
    0.0873    0.7709   -0.0821
   -0.1616    0.5320   -0.0888
    0.2169    0.2844    0.6635
    0.0016   -0.1881    0.7849
   -0.2289    0.0636    0.6475
```

Вращение promax создает более простую структуру нагрузок, в каждой из которых большинство компаний имеет большую нагрузку только по одному фактору.

Чтобы увидеть эту структуру более ясно, используем функцию biplot для изображения каждой компании, использующей свои факторы нагрузок как координаты.

biplot(coefs) создает график коэффициентов матрице coefs. График является двумерным , если coefs имеет два столбца или трехмерным, он имеет три столбца. coefs обычно содержит нагрузки факторов или коэффициенты главных компонент. Оси в biplot представляют собой главные компоненты или скрытые факторы (столбцы coefs) и наблюдаемые переменные (строки coefs), представленные как векторы.

График biplot позволяет визуализировать величину и знак вклада каждой из переменной в каждый из двух или трех факторов и как каждое наблюдение представлено в терминах этих компонент.

biplot накладывает ограничения, заставляя элемент с наибольшей величиной в каждом столбце coefs быть положительным. Это может развернуть некоторые векторы в coefs в противоположном направлении, но часто делает график легче для понимания. Интерпретация графика не меняется, так как изменение знака коэффициента не изменяет его смысл.

biplot(coefs,'Name',Value) определяет одну или несколько пар имя/значение входных параметров.

Наиболее употребимые параметры:

- **Scores**

Выводит матрицу coefs

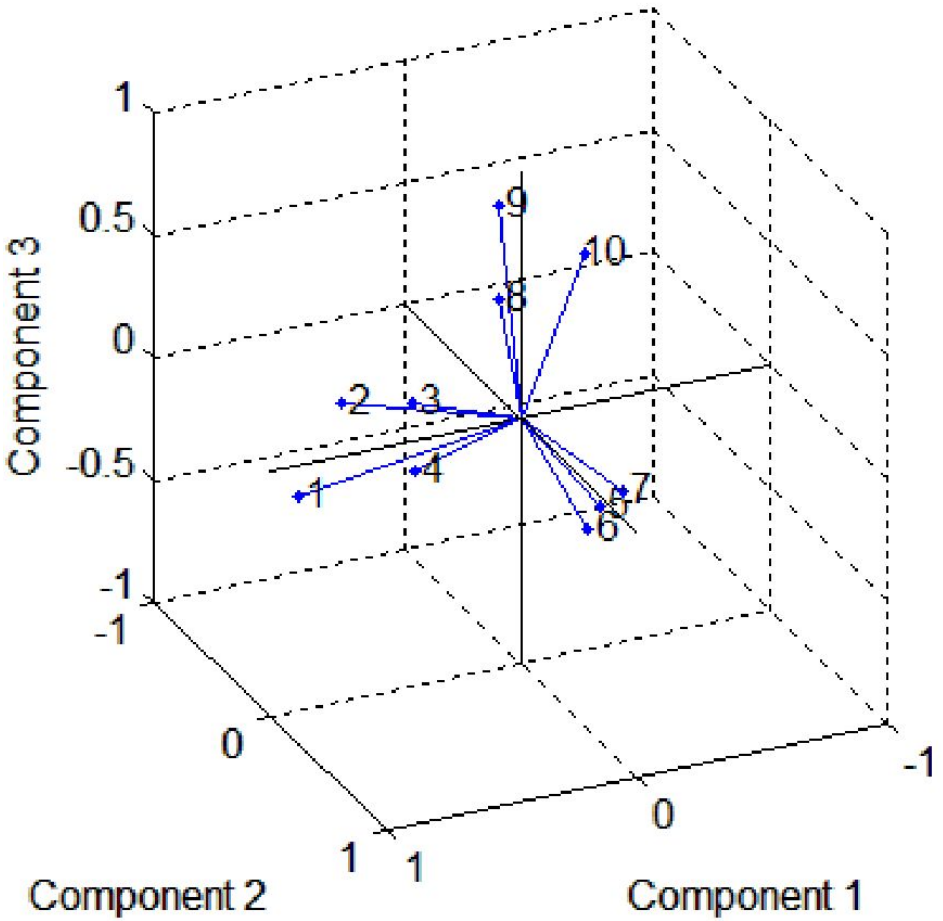
- **VarLabels**

Метки каждого вектора (переменной) с текстами в символьном массиве или массиве ячеек.

Для нашего примера

```
biplot(LoadingsPM,'varlabels',num2str((1:10)'));
```

выведет рисунок:


Component 3
1
0.5
0
-0.5
-1
-1
0
1
Component 2
1
0
-1
Component 1
1
2
3
4
5
6
7
8
9
10

Этот график показывает, что косоугольное вращение привело нагрузки факторов к простейшей структуре.

Каждая компания зависит в первую очередь только от одного фактора и возможно описать каждый фактор в терминах данных, на которые он влияет. На основе тех компаний, которые расположены вблизи оси, можно сделать вывод, что первая ось представляет финансовый сектор, вторая -торговлю и третья - сектор технологии.

Первоначальное предположение, что данные изменяются в первую очередь внутри сектора, по-видимому подтверждаются данными.

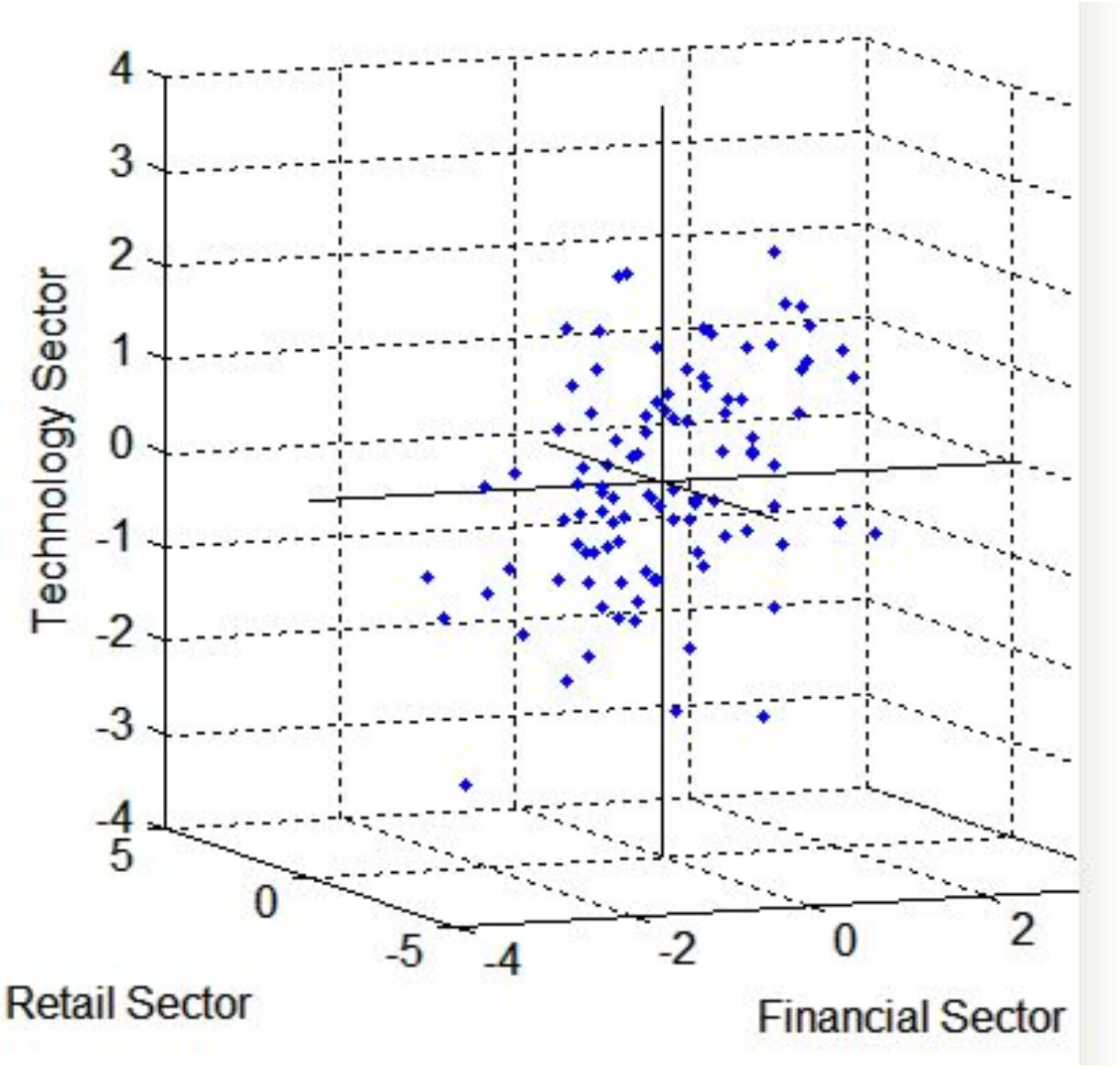
Координаты факторов

Часто полезно классифицировать наблюдения на основе координат их факторов. Например, если принята трехфакторная модель и интерпретация вращения факторов, можно классифицировать каждую неделю с учетом того, насколько это было выгодно для каждого из трех секторов на основе данных из 10 наблюдаемых компаний.

Так как в этом примере данные отражают изменение цен на сырье и акции, можно заставить factoran вернуть значения каждого из трех факторов после их вращения для каждой недели (пятый выходной параметр).

Затем можно построить график по этим координатам, чтобы видеть, как различные сектора влияют на данные каждой недели.

```
[LoadingsPM,specVarPM,TPM,stats,F] = ...
  factoran(stocks, 3,'rotate','promax');
%TPM – матрица вращения
%матрица F является матрицей с размерностью n×m. % Строки матрицы F соответствуют наблюдениям, а % Столбцы - факторамиplot3(F(:,1),F(:,2),F(:,3),'b.')
line([-4 4 NaN 0 0 NaN 0 0], [0 0 NaN -4 4 NaN 0 0],...
    [0 0 NaN 0 0 NaN -4 4], 'Color','black')
xlabel('Financial Sector')
ylabel('Retail Sector')
zlabel('Technology Sector')
grid on
```


Technology Sector
4
3
2
1
0
-1
-2
-3
-4
5
0
-5
Retail Sector
-4
-2
0
2
Financial Sector

Косоугольное вращение часто создает коррелированные факторы.

Рисунок дает некоторые доказательства корреляции между первым и третьим факторами и можно исследовать это дальше, вычисляя корреляционную матрицу факторов.

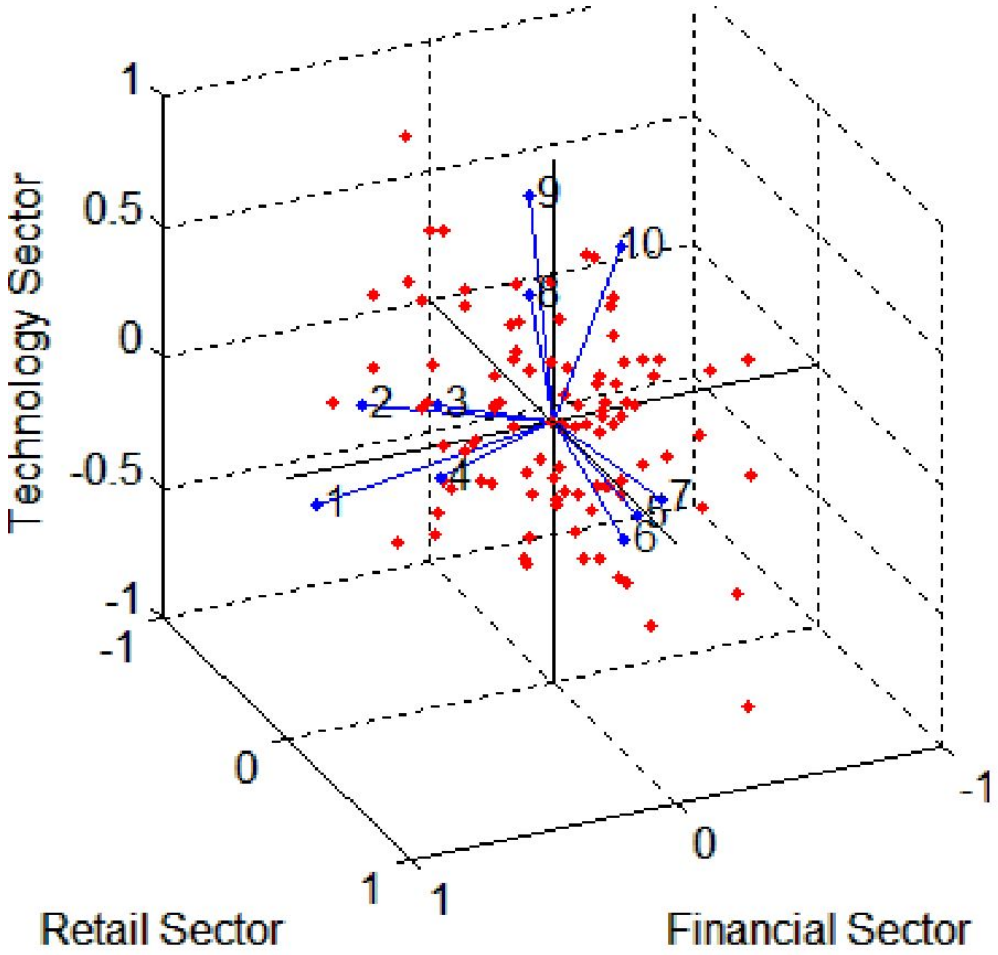
```
inv(TPM'*TPM)
ans =
    1.0000    0.1559    0.4082
    0.1559    1.0000   -0.0559
    0.4082   -0.0559    1.0000
```

Визуализация результатов

Можно использовать функцию biplot для визуализации факторных нагрузок для каждой переменной и факторных множеств для каждого наблюдения на одном графике.

Например, следующий график выводит данные по акциям и метки каждой из 10 компаний.

```
biplot(LoadingsPM,'scores',F,'varlabels',num2str((1:10)'))
xlabel('Financial Sector')
ylabel('Retail Sector')
zlabel('Technology Sector')
axis square
view(155,27)
```


Technology Sector
1
0.5
0
-0.5
-1
-1
0
1
Retail Sector
1
0
-1
Financial Sector
1
2
3
4
5
6
7
8
9
10

В этом случае biplot является трехмерным. Каждая из 10 компаний представлена на этом графике вектором, и направление и длина вектора указывает, как каждая компания зависит от факторов.

Например, можно видеть, что после косоугольного вращения первые 4 компании имеют положительные нагрузки по первому фактору и незначительные нагрузки по двум другим факторам. Первый фактор, интерпретируемый как финансовый сектор, представлен горизонтальной осью. Зависимость этих 4 компаний от этого фактора соответствует тому, что эти 4 вектора направлены примерно вдоль этой оси. Подобным образом, зависимость компаний 5, 6 и 7 в основном ль второго факторar, интерпретируемого как торговый сектор, представлено векторами, направленными примерно вдоль этой оси.

Каждое из 100 наблюдений представлено на графике точкой, и их положение указывает координаты каждого наблюдения по трем факторам.

Например, точки в верхней части графика имеют максимальные координаты для фактора технологический сектор. Точки масштабированы, поэтому только их относительное положение может наблюдаться на графике.

Можно использовать инструмент Data Cursor из пункта меню Tools в окне рисунка для идентификации элементов графика.

Нажатие компании (вектора) покажет нагрузки для каждого фактора.

Нажатие наблюдения (точки) покажет координаты наблюдения по каждому фактору.

Пример с ирисами Фишера

Так как число признаков d=4,то согласно ограничению $(d+m) \le (d-m)^2$.

можно создать только один фактор (m=1).

В этом случае построение графика biplot невозможно

Создадим модель с косоугольными вращениями и сохраним координаты факторов:

```
[LoadingsPM,specVarPM,TPM,stats,F] = ...
  factoran(meas, 1,'rotate','promax');
```

Массив F будет в дальнейшем играть роль матрицы измерений. По ней создадим обучающую и контролирующую последовательности

```
load fisheriris
[Loadings2,specificVar2,T2,stats2] = ...
  factoran(meas,1,'rotate','none');
[LoadingsPM,specVarPM,TPM,stats,F] = ...
  factoran(meas, 1,'rotate','promax');
Xt1=F(1:25);
Xt2=F(51:75);
Xt3=F(101:125);
Xt=[Xt1;Xt2;Xt3];
Yt(1:25)=1;
Yt(26:50)=2;
Yt(51:75)=3;
Xc1=F(26:50);
Xc2=F(76:100);
Xc3=F(126:150);
Xc=[Xc1;Xc2;Xc3];
Yc(1:25)=1;
Yc(26:50)=2;
Yc(51:75)=3;
T=[Xt Yt'];
C=[Xc Yc'];
tic
```

Построим систему нечеткого вывода с помощью genfis1 и настроим ее с помощью anfis

```
fis = genfis1(T,[2],char('trimf'),char('constant'))
epoch_n = 10;
 [fis,trn_error] = anfis(T, fis) ;
```

Выдадим график ошибок

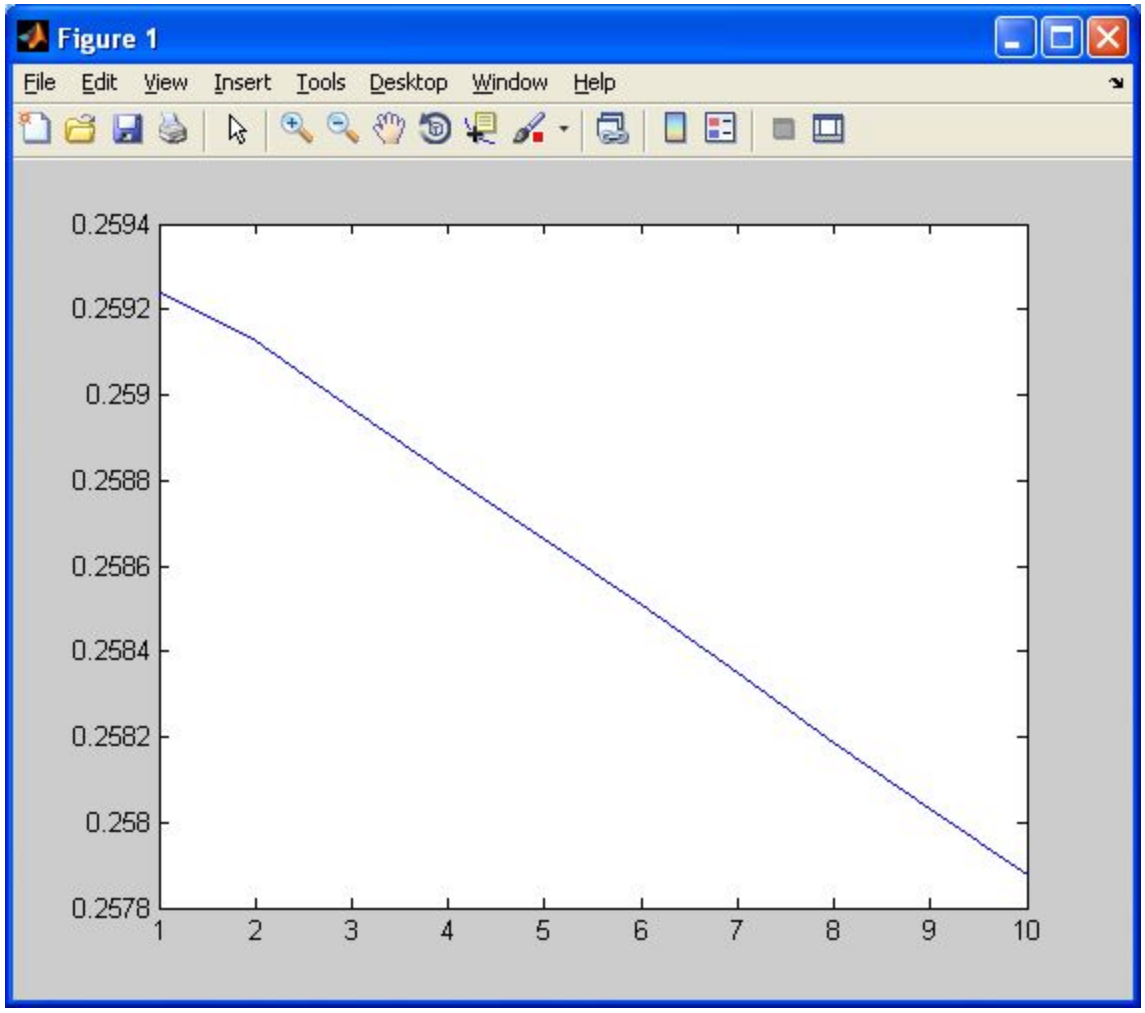


Figure 1
File Edit View Insert Tools Desktop Window Help
0.2594
0.2592
0.259
0.2588
0.2586
0.2584
0.2582
0.258
0.2578
1 2 3 4 5 6 7 8 9 10

Посчитаем выходные значения по системе нечеткого вывода для обучающих и контролирующих данных:

```
anfis_t = evalfis(Xt, fis);
anfis_c = evalfis(Xc, fis);
 an_t=round(anfis_t);
an_c=round(anfis_c);
```

Оценим процент распознанных данных по обучающей и контролирующей выборке

proc_t=length(find(an_t==Yt'))/75*100

proc_c=length(find(an_c==Yc'))/75*100

- По обучающей: 94.6667
- По контролирующей: 93.3333