Федеральное государственное автономное образовательное учреждение высшего образования«Московский физико-технический институт (национальный исследовательский университет)»

Проектная работа:

Справка по использованию statistics and machine learning toolbox

Выполнил студент:
Сериков Василий Романович
Группа: Б03-102

Москва, 2024 г.

Содержание

1 Введение

Statistics and Machine Learning Toolbox (SMLT) в MATLAB предоставляет широкий спектр функций для обработки экспериментальных данных и постановки эксперимента. Эта справка описывает основные функции SMLT и их применение для решения типичных задач. SMLT является одним из наиболее популярных и мощных инструментов для обработки экспериментальных данных, постановки эксперимента и машинного обучения в MATLAB.

История развития:
SMLT был впервые представлен в MATLAB в 1993 году под названием Statistics Toolbox. В течение многих лет инструмент постоянно развивался и совершенствовался, добавлялись новые функции и возможности. В 2012 году инструмент был переименован в Statistics and Machine Learning Toolbox, чтобы отразить его расширенные возможности в области машинного обучения.

Состав инструмента:
SMLT состоит из множества функций и инструментов для обработки экспериментальных данных, постановки эксперимента и машинного обучения. Инструмент включает в себя следующие компоненты:
1) Функции для описательной статистики, такие как вычисление среднего значения, дисперсии, медианы, квантилей и других характеристик.
2) Функции для визуализации данных, такие как построение гистограмм, бокс-плотов, диаграмм рассеяния и других графиков.
3) Функции для статистического вывода, такие как проверка гипотез, построение доверительных интервалов, регрессионный анализ и другие методы.
4) Функции для машинного обучения, такие как обучение классификаторов, регрессионных моделей, моделей кластеризации и других алгоритмов.
5) Функции для обработки временных рядов, такие как анализ спектра, фильтрация, прогнозирование и другие методы.
6) Функции для оптимизации, такие как линейное и нелинейное программирование, оптимизация по критерию максимального правдоподобия и другие методы.

Преимущества инструмента:
SMLT имеет множество преимуществ перед другими инструментами для обработки экспериментальных данных, постановки эксперимента и машинного обучения. Некоторые из них:
1) Возможность использования в среде MATLAB, которая предоставляет широкие возможности для математического моделирования, визуализации данных и других вычислений.
2) Наличие большого количества готовых функций и инструментов для решения типичных задач в области статистики и машинного обучения.
3) Возможность расширения функциональности инструмента за счет использования дополнительных тулбоксов и библиотек.
4) Наличие подробной документации и примеров кода, которые помогают освоить инструмент и эффективно использовать его возможности.
5) Возможность интеграции с другими инструментами и языками программирования, такими как Python, C++ и другими.

Функции для анализа данных:
Среди функций для анализа данных в Statistics and Machine Learning Toolbox можно выделить следующие:
1) Функции для вычисления статистических характеристик, таких как среднее значение, ме-

диана, дисперсия, коэффициент вариации и другие.
2) Функции для проверки гипотез, такие как t-тест, критерий хи-квадрат, критерий Колмогорова-Смирнова и другие.
3) Функции для анализа дисперсии, такие как однофакторный и многофакторный дисперсионный анализ, анализ ковариации и другие.
4) Функции для анализа временных рядов, такие как авторегрессионная модель, модель движущего среднего, модель авторегрессии с движущим средним и другие.
5) Функции для кластерного анализа, такие как иерархический кластерный анализ, k-средних, метод главных компонент и другие.

Функции для машинного обучения:
Среди функций для машинного обучения в Statistics and Machine Learning Toolbox можно выделить следующие:
1) Функции для создания и обучения классификаторов, таких как линейные и нелинейные классификаторы, деревья решений, нейронные сети и другие.
2) Функции для создания и обучения регрессионных моделей, таких как линейная регрессия, логарифмическая регрессия, регрессия на основе деревьев и другие.
3) Функции для создания и обучения моделей кластеризации, таких как k-средних, иерархической кластеризации, кластеризации на основе плотности и других методов.
4) Функции для оценки качества обученных моделей, такие как вычисление точности, чувствительности, специфичности, F1-меры и других метрик.
5) Функции для подбора гиперпараметров, такие как поиск сетки, кросс-валидация, байесовская оптимизация и другие методы.

2 Обработка экспериментальных данных

Очистка данных:
Очистка данных - это важный этап подготовки данных для последующего анализа. В SMLT существует несколько функций для очистки данных, одна из которых - *rmmissing*. Эта функция удаляет строки с пропущенными значениями из таблицы. Например, если у нас есть таблица *data* с пропущенными значениями, то мы можем удалить эти строки следующим образом:

```
data = rmmissing(data);
```

Если же нужно удалить столбцы с пропущенными значениями, то можно использовать опцию *'columns'*:

```
data = rmmissing(data,'columns');
```

Статистический анализ:
Статистический анализ - это неотъемлемая часть обработки экспериментальных данных. В SMLT существует множество функций для статистического анализа, в том числе *mean*, *std*, *corrcoef* и другие.

Функция *mean* вычисляет среднее арифметическое значение элементов массива. Например, если у нас есть вектор x, то среднее арифметическое вычисляется следующим образом:

```
m = mean(x);
```

Функция *std* вычисляет стандартное отклонение элементов массива. Например, стандартное отклонение вектора *x* вычисляется следующим образом:

```
s = std(x);
```

Функция *corrcoef* вычисляет матрицу коэффициентов корреляции Пирсона. Например, если у нас есть две переменные *x* и *y*, то матрица коэффициентов корреляции вычисляется следующим образом:

```
R = corrcoef(x,y);
```

Визуализация данных:
Функция *fitrlinear* обучает линейную регрессионную модель, *fitrtree* - модель регрессии на базе дерева решений.

Визуализация данных - это важный этап анализа экспериментальных данных. В SMLT существует множество функций для визуализации данных, в том числе *histogram*, *boxplot*, *scatter* и другие.

Функция *histogram* строит гистограмму, которая показывает распределение данных. Например, если у нас есть вектор *x*, то гистограмма строится следующим образом:

```
histogram(x);
```

Функция *boxplot* строит бокс-плот, который показывает статистическое распределение данных. Например, если у нас есть вектор *x*, то бокс-плот строится следующим образом:

```
boxplot(x);
```

Функция *scatter* строит точечную диаграмму, которая показывает зависимость между двумя переменными. Например, если у нас есть две переменные *x* и *y*, то точечная диаграмма строится следующим образом:

```
scatter(x,y);
```

3 Постановка эксперимента

Планирование эксперимента:
Планирование эксперимента - это важный этап постановки эксперимента. В SMLT существует несколько функций для планирования эксперимента, одна из которых - *designmatrix*. Эта функция создает матрицу планирования эксперимента. Например, если у нас есть факторы *A*, *B* и *C*, то матрица планирования эксперимента создается следующим образом:

```
D = designmatrix([A,B,C],'full');
```

Функция *rng* инициализирует генератор случайных чисел для создания случайных выборок. Например, если нужно создать случайную выборку размера *n* из нормального распре-

деления с параметрами *mu* и *sigma*, то это можно сделать следующим образом:

```
rng('shuffle');
x = normrnd(mu,sigma,n,1);
```

Анализ результатов эксперимента:
Анализ результатов эксперимента - это неотъемлемая часть постановки эксперимента. В SMLT существует множество функций для анализа результатов эксперимента, в том числе *anova1*, *ttest2*, *regress* и другие.

Функция *anova1* проводит однофакторный дисперсионный анализ. Например, если у нас есть переменная *y* и фактор *A*, то однофакторный дисперсионный анализ проводится следующим образом:

```
[p,tbl,stats] = anova1(y,A);
```

Функция *ttest2* проводит двухвыборочный t-тест. Например, если у нас есть две выборки *x* и *y*, то двухвыборочный t-тест проводится следующим образом:

```
[h,p] = ttest2(x,y);
```

Функция *regress* проводит линейную регрессию. Например, если у нас есть переменная *y* и матрица предикторов *X*, то линейная регрессия проводится следующим образом:

```
mdl = fitlm(X,y);
```

Дополнительные функции:
Кроме вышеупомянутых функций, в SMLT существует множество других функций для постановки эксперимента. Например, функция *anovan* проводит n-факторный дисперсионный анализ, функция *kruskalwallis* - критерий Краскела-Уоллиса, функция *friedman* - критерий Фридмана, функция *durbinwatson* - статистика Дербина-Уотсона, функция *levene* - критерий Левена, функция *median* вычисляет медиану элементов массива, функция *mode* - моду, функция *quantile* - квантили, функция *interquartile* - межквартильное расстояние, функция *skewness* - коэффициент асимметрии, функция *kurtosis* - коэффициент эксцесса и другие.

4 Машинное обучение

Машинное обучение - это одна из ключевых областей применения SMLT. В этом разделе будут рассмотрены некоторые функции SMLT для машинного обучения, которые были упомянуты ранее.

Классификация:
Классификация - это задача машинного обучения, которая заключается в предсказании класса объекта на основе его характеристик. В SMLT существует множество функций для классификации, в том числе *fitcsvm*, *fitctree*, *fitcdiscr* и другие. Функция *fitcsvm* обучает модель классификации на базе метода опорных векторов. Например, если у нас есть матрица предикторов *X* и вектор ответов *Y*, то модель классификации обучается следующим образом:

```
mdl = fitcsvm(X,Y);
```

Функция *fitctree* обучает модель классификации на базе дерева решений. Например, если у нас есть матрица предикторов X и вектор ответов Y, то модель классификации обучается следующим образом:

```
mdl = fitctree(X,Y);
```

Функция *fitcdiscr* обучает модель классификации на базе дискриминантного анализа. Например, если у нас есть матрица предикторов X и вектор ответов Y, то модель классификации обучается следующим образом:

```
mdl = fitcdiscr(X,Y);
```

Регрессия:

Регрессия - это задача машинного обучения, которая заключается в предсказании числового значения на основе характеристик объекта. В SMLT существует множество функций для регрессии, в том числе *fitrlinear*, *fitrtree*, *fitrensemble* и другие.

Функция *fitrlinear* обучает линейную регрессионную модель. Например, если у нас есть матрица предикторов X и вектор ответов Y, то линейная регрессионная модель обучается следующим образом:

```
mdl = fitrlinear(X,Y);
```

Функция *fitrtree* обучает модель регрессии на базе дерева решений. Например, если у нас есть матрица предикторов X и вектор ответов Y, то модель регрессии обучается следующим образом:

```
mdl = fitrtree(X,Y);
```

Функция *fitrensemble* обучает ансамблевую модель регрессии. Например, если у нас есть матрица предикторов X и вектор ответов Y, то ансамблевая модель регрессии обучается следующим образом:

```
mdl = fitrensemble(X,Y);
```

Оценка качества модели:

Оценка качества модели - это один из важных этапов машинного обучения. В SMLT существует множество функций для оценки качества модели, в том числе *confusionmat*, *roccurve*, *liftcurve* и другие.

Функция *confusionmat* строит матрицу ошибок классификации. Например, если у нас есть модель классификации *mdl* и вектор ответов Y, то матрица ошибок классификации строится следующим образом:

```
C = confusionmat(Y, predict(mdl,X));
```

Функция *roccurve* строит ROC-кривую. Например, если у нас есть модель классификации *mdl* и вектор ответов Y, то ROC-кривая строится следующим образом:

```
[X,Y,T,AUC] = roccurve(Y, predict(mdl,X));
```

Функция *liftcurve* строит кривую лифта. Например, если у нас есть модель классификации *mdl* и вектор ответов *Y*, то кривая лифта строится следующим образом:

```
lift = liftcurve(Y,predict(mdl,X));
```

Дополнительные функции:
Кроме вышеупомянутых функций, в SMLT существует множество других функций для машинного обучения. Например, функция *fitcknn* обучает модель классификации на базе k-ближайших соседей, функция *fitcecoc* обучает модель классификации на базе кодирования ошибок, функция *fitcensemble* обучает ансамблевую модель классификации, функция *fitrknn* обучает модель регрессии на базе k-ближайших соседей, функция *fitrensemble* обучает ансамблевую модель регрессии и другие.

5 Примеры применения SMLT

В этом разделе будут приведены несколько примеров применения SMLT для решения типичных задач.

Обработка экспериментальных данных:
Предположим, что у нас есть экспериментальные данные о зависимости температуры воздуха от времени. Необходимо построить график зависимости и вычислить коэффициент корреляции Пирсона.

Для решения этой задачи можно воспользоваться функциями *plot* и *corrcoef*. Пусть у нас есть векторы *t* и *T*, содержащие значения времени и температуры соответственно. Тогда график зависимости и коэффициент корреляции Пирсона можно получить следующим образом:

```
plot(t,T);
xlabel('Time, s');
ylabel('Temperature, C');
R = corrcoef(t,T);
```

Рассмотрим еще пример. Предположим, что у нас есть данные о росте растений в зависимости от количества удобрений. Мы хотим провести анализ дисперсии, чтобы определить, влияет ли количество удобрений на рост растений.

Для этого мы можем воспользоваться функцией *anova1*, которая выполняет однофакторный дисперсионный анализ. Данные можно загрузить в MATLAB с помощью функции *readtable*. Затем мы можем выполнить анализ дисперсии и проверить гипотезу о том, что средние значения роста растений в разных группах одинаковы.

```
% Load data
data = readtable('plant_growth.csv');

% Performing one-way ANOVA
[p,tbl,stats] = anova1(data.Growth, data.Fertilizer);

%  Testing the hypothesis of equality of average values
alpha = 0.05;
if p < alpha
    disp('The hypothesis of equality of average values is rejected')
else
    disp('The hypothesis of equality of average values is accepted')
end
```

Постановка эксперимента:

Предположим, что необходимо провести эксперимент по изучению влияния температуры на прочность материала. Для этого нужно спланировать эксперимент, провести его и проанализировать результаты.

Для решения этой задачи можно воспользоваться функциями *designmatrix*, *rng*, *anova1*. Пусть у нас есть фактор *Temperature* с тремя уровнями (20, 50, 80 градусов Цельсия) и фактор *Material* с двумя уровнями (материал А и материал Б). Тогда матрица планирования эксперимента, случайная выборка и результаты однофакторного дисперсионного анализа могут быть получены следующим образом:

```
D = designmatrix([Temperature,Material],'full');
rng('shuffle');
% Sample size
n = 10;
% Random sample
X = D(:,1:2)*[20,50,80;0,0,1]+normrnd(0,5,n,2);
% Result of experiment
Y = X*[1;2;3]+normrnd(0,10,n,1);
% One-way ANOVA
[p,tbl,stats] = anova1(Y,D(:,3));
```

Машинное обучение:

Предположим, что необходимо построить модель классификации для предсказания вида растения по его характеристикам. Для этого нужно обучить модель на обучающей выборке и проверить ее на тестовой выборке.

Для решения этой задачи можно воспользоваться функциями *fitcsvm*, *predict*. Пусть у нас есть матрица предикторов *X* и вектор ответов *Y*, содержащие характеристики растений и их виды соответственно. Тогда модель классификации и ее точность на тестовой выборке могут быть получены следующим образом:

```
% Model training
mdl = fitcsvm(Xtrain,Ytrain);
% Prediction on test sample
Ypred = predict(mdl,Xtest);
% Model accuracy
accuracy = sum(Ypred==Ytest)/length(Ytest);
```

Рассмотрим еще пример. Предположим, что у нас есть данные о клиентах банка, и мы хотим построить модель классификации, которая будет предсказывать, будет ли клиент брать кредит в банке.

Для этого мы можем воспользоваться функцией *fitcecoc*, которая обучает модель классификации на основе метода ошибок и кодов. Данные можно загрузить в MATLAB с помощью функции *readtable*. Затем мы можем разбить данные на обучающую и тестовую выборки, обучить модель классификации и проверить ее качество на тестовой выборке.

```
% Load data
data = readtable('bank_data.csv');

% Dividing data into training and test samples
cvp = cvpartition(data.Response,'HoldOut',0.3);
dataTrain = data(cvp.training,:);
dataTest = data(cvp.test,:);

% Classification model training
mdl = fitcecoc(dataTrain,'Response','Learners','logistic');

% Prediction on test sample
```

```
    labelsPred = predict(mdl, dataTest);

    % Calculating model quality
    confMat = confusionmat(dataTest.Response, labelsPred);
    accuracy = sum(diag(confMat))/sum(confMat(:));
    disp(['Model accuracy: ', num2str(accuracy)])
```

Использования функций для кластеризации:

Рассмотрим пример использования функций для кластеризации в Statistics and Machine Learning Toolbox. Предположим, что у нас есть данные о покупателях интернет-магазина, и мы хотим разделить их на группы в зависимости от их поведения при покупке. Для этого мы можем воспользоваться функцией *kmeans*, которая выполняет кластеризацию на основе метода k-средних. Данные можно загрузить в MATLAB с помощью функции *readtable*. Затем мы можем выполнить нормализацию данных, выбрать количество кластеров, выполнить кластеризацию и визуализировать результаты.

```
% Load data
data = readtable('online_shoppers.csv');

% Data normalization
dataNorm = normalize(data{:,2:end});

% Selecting the number of clusters
numClusters = 3;

% Performing clustering
[idx,C] = kmeans(dataNorm,numClusters);

% Visualization of results
figure;
gscatter(dataNorm(:,1),dataNorm(:,2),idx);
title('Clustering online store clients');
xlabel('Time on site (normalized)');
ylabel('Total cost of purchases (normalized)');
```

Использования функций для регрессионного анализа:

Рассмотрим пример использования функций для регрессионного анализа в Statistics and Machine Learning Toolbox. Предположим, что у нас есть данные о ценах на недвижимость в разных районах города, и мы хотим построить модель регрессии, которая будет предсказывать цену на недвижимость в зависимости от площади и количества комнат. Для этого мы можем воспользоваться функцией *fitlm*, которая обучает линейную модель регрессии. Данные можно загрузить в MATLAB с помощью функции *readtable*. Затем мы можем разбить данные на обучающую и тестовую выборки, обучить модель регрессии и проверить ее качество на тестовой выборке.

```
% Load data
data = readtable('real_estate_prices.csv');

% Dividing data into training and test samples
cvp = cvpartition(size(data,1),'HoldOut',0.3);
dataTrain = data(cvp.training,:);
dataTest = data(cvp.test,:);

% Training a regression model
mdl = fitlm(dataTrain,'Price Area+Rooms');

% Prediction on a test sample
labelsPred = predict(mdl, dataTest);

% Calculating model quality
mse = mean((dataTest.Price - labelsPred).^2);
rmse = sqrt(mse);
```

```
disp(['Standard deviation: ', num2str(rmse)])
```

Использования функций для глубокого обучения:
Рассмотрим пример использования функций для глубокого обучения в Statistics and Machine Learning Toolbox. Предположим, что у нас есть набор изображений, и мы хотим обучить нейронную сеть, которая будет классифицировать изображения по их классам.

Для этого мы можем воспользоваться функцией *trainNetwork*, которая обучает нейронную сеть на основе метода обратного распространения ошибки. Данные можно загрузить в MATLAB с помощью функции *imageDatastore*. Затем мы можем разбить данные на обучающую и тестовую выборки, определить архитектуру нейронной сети, обучить ее и проверить ее качество на тестовой выборке.

```
% Load data
imds = imageDatastore('image_dataset','IncludeSubfolders',
                      true,'LabelSource','foldernames');

% Dividing data into training and test samples
[imdsTrain,imdsTest] = splitEachLabel(imds,0.3);

% Defining a neural network architecture
layers = [
imageInputLayer([28 28 1])
convolution2dLayer(3,8,'Padding','same')
reluLayer
maxPooling2dLayer(2,'Stride',2)
fullyConnectedLayer(10)
softmaxLayer
classificationLayer];

% Neural network training
options = trainingOptions('sgdm', ...
'MaxEpochs',20, ...
'MiniBatchSize',128, ...
'ValidationData',imdsTest, ...
'ValidationFrequency',500, ...
'Verbose',false, ...
'Plots','training-progress');
net = trainNetwork(imdsTrain,layers,options);

% Prediction on a test sample
labelsPred = classify(net,imdsTest);

% Calculating model quality
confMat = confusionmat(imdsTest.Labels,labelsPred);
accuracy = sum(diag(confMat))/sum(confMat(:));
disp(['Model accuracy: ', num2str(accuracy)])
```

6 Дополнительные возможности

SMLT предоставляет множество дополнительных возможностей для обработки экспериментальных данных, постановки эксперимента и машинного обучения. В этом разделе будут рассмотрены некоторые из них.

Обработка временных рядов:
SMLT предоставляет множество функций для обработки временных рядов. Например, функция *diff* вычисляет разность между соседними элементами временного ряда, функция *detrend* удаляет трендовую компоненту из временного ряда, функция *seasonal* выделяет сезонную компоненту из временного ряда, функция *arima* обучает модель ARIMA для прогнозирования временного ряда и другие.

Анализ частот:
SMLT предоставляет множество функций для анализа частот. Например, функция *fft* вычисляет быстрое преобразование Фурье, функция *periodogram* вычисляет периодограмму, функция *spectrogram* вычисляет спектрограмму, функция *coherence* вычисляет когерентность между двумя сигналами и другие.

Обработка изображений:
SMLT предоставляет множество функций для обработки изображений. Например, функция *imread* читает изображение с диска, функция *imresize* изменяет размер изображения, функция *imfilter* фильтрует изображение, функция *imshow* отображает изображение, функция *imhist* вычисляет гистограмму изображения и другие.

Обработка сигналов:
SMLT предоставляет множество функций для обработки сигналов. Например, функция *filter* фильтрует сигнал, функция *conv* вычисляет свертку сигналов, функция *corr* вычисляет корреляцию сигналов, функция *spectrum* вычисляет спектр сигнала, функция *findpeaks* находит пики сигнала и другие.

Оптимизация:
SMLT предоставляет множество функций для оптимизации. Например, функция *fminunc* минимизирует функцию с помощью метода квазиньютоновской оптимизации, функция *fmincon* минимизирует функцию с ограничениями, функция *fminsearch* минимизирует функцию с помощью метода Нелдера-Мида, функция *patternsearch* минимизирует функцию с помощью метода поиска по шаблону и другие.

Статистический контроль качества:
SMLT предоставляет множество функций для статистического контроля качества. Например, функция *controlchart* строит график управления качеством, функция *capability* вычисляет способность процесса, функция *pchart* строит p-график, функция *npchart* строит np-график, функция *uchart* строит u-график и другие.

Многомерная статистика:
SMLT предоставляет множество функций для многомерной статистики. Например, функция *pca* вычисляет главные компоненты, функция *mdscale* вычисляет многомерное масштабирование, функция *cluster* выполняет кластеризацию, функция *discriminant* вычисляет дискриминантный анализ, функция *canoncorr* вычисляет каноническую корреляцию и другие.

7 Рекомендации по использованию SMLT

При использовании SMLT необходимо соблюдать некоторые рекомендации, чтобы эффективно решать задачи обработки экспериментальных данных, постановки эксперимента и машинного обучения.

Подготовка данных:
Перед использованием функций SMLT необходимо подготовить данные. Это включает в себя очистку данных от ошибок и пропусков, нормализацию и стандартизацию данных, выбор подходящих предикторов и ответов, разбиение данных на обучающую и тестовую выборки и другие.

Выбор подходящих функций:
SMLT предоставляет множество функций для решения различных задач. Необходимо выбирать подходящие функции в зависимости от типа данных, цели анализа и других факторов. Для этого можно воспользоваться документацией SMLT, примерами кода и рекомендациями других пользователей.

Обучение и тестирование модели:
При использовании функций машинного обучения необходимо обучить модель на обучаю-

щей выборке и проверить ее на тестовой выборке. Для этого можно воспользоваться функциями *fit* и *predict*. Необходимо также выбирать подходящие параметры модели, такие как тип ядра, параметр регуляризации, глубина дерева и другие.

Оценка качества модели:

При использовании функций машинного обучения необходимо оценивать качество модели. Для этого можно воспользоваться функциями *confusionmat*, *roc*, *lift* и другими. Необходимо также выбирать подходящие метрики качества, такие как точность, чувствительность, специфичность, F1-мера и другие.

Визуализация результатов:

При использовании функций SMLT необходимо визуализировать результаты анализа. Для этого можно воспользоваться функциями *plot*, *histogram*, *boxplot*, *scatter* и другими. Визуализация результатов помогает лучше понять данные, обнаружить закономерности и выявить ошибки.

8 Заключение

В этой справке были рассмотрены основные функции Statistics and Machine Learning Toolbox, которые являются мощными инструментами для обработки экспериментальных данных, постановки эксперимента и машинного обучения в MATLAB. Эти инструменты предоставляют широкий спектр функций и возможностей для решения типичных задач в этих областях. Знакомство с этими инструментами позволит эффективно решать задачи обработки экспериментальных данных, постановки эксперимента и машинного обучения в MATLAB.

9 Список литературы

1. The MathWorks, Inc. Statistics and Machine Learning Toolbox Documentation.
2. The MathWorks, Inc. MATLAB Documentation.