Система контроля версий, тестирование

1 октября 2013 г.

Срок сдачи - 13 октября 2013 года, 23:59.

Максимальный балл - 5.0.

Данное задание направлено на освоение студентами навыков использования систем контроля версий, тестирования, оформления кода.

Задание:

1. Научиться работать с системой SVN. Проделать операции, описанные в разделе ниже.

2. Решить на MATLAB три задачи из список ниже. Распределение задач по студентам приведено на сайте курса. Весь написанный код должен подчиняться популярному style guide для MATLAB [1].

   Замечание. Поскольку в заданиях предполагается проверка многих условий на входные значения, разрешается не выполнять пункт “The usual case should be put in the if-part and the exception in the else-part of an if else statement”, если это улучшает читаемость.

3. Провести юнит-тестирование решённых задач. Инструкции по юнит-тестированию приведены ниже.

4. Написать отчет о проделанной работе (формат PDF). Отчёт предлагается выполнять в системе LaTeX, за отчёт в других системах начисляется штраф в 1 балл. Полезные ссылки: [2] [3]. Считается, что отчёт был выполнен в LaTeX, если в репозитории есть .tex файл и исходники картинок, из которых можно собрать PDF файл с отчётом. Отчет должен включать в себя краткое описание решения (алгоритма) каждого из трех заданий, а также описание работы с системой контроля версий: скриншоты или текст из консоли, иллюстрирующие, как проводить основные действия, такие как обновление и коммит.

5. После выполнения задания написать преподавателю об этом. Никакие файлы отправлять не нужно, будет проверяться последняя ревизия из репозитория. Перед этим обязательно проверьте, что все тесты проходят.

1 Работа с системой SVN

1. Установить на компьютер клиент SVN. Под ОС Windows рекомендуется TortoiseSVN [4].
2. Получить по почте логин, пароль и путь к репозиторию.
3. Сделать checkout из репозитория в рабочую папку.
4. Задание по MATLAB и юнит-тестированию, а также написание отчёта предлагается выполнять в рабочей папке.

В финальной версии репозитория, которая будет проверяться, должны находиться:

1. Исходные файлы.
2. PDF файл с отчётом.
3. Если отчёт написан в системе LaTeX, исходники отчёта: .tex файл и исходники картинок, нужные для сборки отчёта.

Никаких лишних файлов (.asv, .log и т.п.) в финальной версии репозитория быть не должно. Удалите их из репозитория, если они туда попали.

Рекомендации по использованию репозитория (за несоблюдение баллы сниматься не будут):

1. Изменения в решениях разных задач должны находиться в разных коммитах.
2. Приветствуется большое число коммитов.
3. Все коммиты должны содержать краткое описание (commit message).
4. Лишние файлы можно добавить в ignore list клиента SVN.

Замечание. Если по техническим причинам не удаётся воспользуется удалённым репозиторием, допускается использование локального, см. краткое руководство по созданию и работе с локальным репозиторием для TortoiseSVN [5]. Если клиент требует выбрать тип репозитория, выбирайте fsfs. Для проверки задания нужно будет отправить преподавателю заархивированный репозиторий (не рабочую папку).

2 Задачи

В данном задании предлагается решить задачи на языке MATLAB. Акцент делается на юнит-тестирование кода и на проверку входных значений на корректность, что особенно важно, если код будет использован другими людьми. Кроме того, студентам предлагается освоить базовые навыки работы с системой контроля версий SVN.

В рамках задания необходимо выполнить по одной задаче из каждой части согласно распределению вариантов.

К решениям предъявляются следующие требования:

1. В случае если входные данные не позволяют корректно решить задачу, необходимо выдать информативное исключение. Пример проверки и вызова исключения (repeatAllElements - название функции, в которой вызвано исключение):

```
if ~isrow(x)
  error('repeatAllElements:xIsNotRow', 'x is not a row');
end
```

2. Обратите внимание на следующие случаи:

   (a) Аргументы могут потенциально быть не численного типа, а, например, cell array. Полезные функции: isnumeric, isscalar.

   (b) Матрицы и числа также могут быть не вещественными, а комплексными. Пользуйтесь функцией isreal.

   (c) Элементы могут быть равны Inf или NaN. Функции для проверки: isinf, isnan.

   (d) Элементы должны иметь "подходящие" размеры. Функции для проверки: size, isrow, isscalar.

   (e) Если требуется проверить некое условие для нескольких элементов, используйте функции all и any.

3. На разные типы ошибок должны быть различные исключения, и, соответственно, разные блоки if.

4. Схожие проверки для разных аргументов (скажем, наличие Inf в различных аргументах) можно проводить вместе и вызывать одно исключение.

5. Каждая реализованная функция должна быть покрыта юнит-тестами, см. раздел "Юнит-тестирование".

6. Каждая реализованная функция должна быть кратко задокументирована. Документация должна выводиться по команде help.

В частях 1 и 2 под вектором всегда подразумевается вектор-строка!

2.1 Операции без погрешностей

1. Циклически сдвинуть вектор X на N позиций вправо. Если N отрицательно, сдвигать влево. Пример: X = [1 2 3], N = -1. Ответ: [2 3 1]. Функцию circshift использовать нельзя.

2. Реализовать кодирование длин серий (Run-length encoding). Дан вектор x. Необходимо вернуть два вектора одинаковой длины. Первый содержит числа, а второй - сколько раз их нужно повторить. Пример: X = [2 2 2 3 3 3 5]. Ответ: [2 3 5], [3 3 1].

3. Реализовать декодирование длин серий. Даны два вектора одинаковой длины, values и counts, полученные в результате кодирования длин серий. Необходимо воспроизвести исходную последовательность. Пример: values = [8 7 2 1], counts = [3 2 1 1]. Ответ: [8 8 8 7 7 2 1].

4. Дан вектор целых чисел (если числа вещественные, округлить). Подсчитать суммарное число единичных бит в этих числах. Пример: X = [1 5]. Ответ: 3.

5. Дана прямоугольная матрица. Подсчитать произведение ненулевых элементов на диагонали. Пример: X = [1 0 1; 2 0 2; 3 0 3; 4 4 4]. Ответ: 3.

2.2 Операции с погрешностями

1. Дана матрица X размера (N+1) на N: N+1 точка N-мерного евклидова пространства. Найти объём симплекса, натянутого на эти точки. Объём может быть нулевым, но не может быть отрицательным!

2. Даны два вектора X и Y длины N. Точки X и Y задают диагональ прямоугольного параллелепипеда. Найти объём. Объём может быть нулевым, но не может быть отрицательным!

3. Даны 2 пары точек на плоскости. Вычислить точку пересечения двух прямых, задаваемых парами точек. Если прямые не пересекаются, вернуть вектор из NaN соответствующей размерности.

4. Даны 2 пары точек на плоскости. Вычислить точку пересечения прямой, задаваемой первой парой точек, с отрезком, задаваемой второй парой точек. Если прямая и отрезок не пересекаются, вернуть вектор из NaN соответствующей размерности.

5. Даны 3 точки на плоскости и 1 число. Вычислить точки пересечения прямой, задаваемой первой парой точек и окружности с центром в третьей точке и радиусом, заданным числом. Если прямая и окружность не пересекаются, вернуть вектор из NaN соответствующей размерности.

2.3 Алгоритмы машинного обучения

Пусть X - вещественная матрица объектов размера N на F, где N - количество объектов, F - количество признаков, y - бинарный вектор-столбец ответов длины N.

1. Решается задача классификации с двумя классами. X - обучающая выборка, y - ответы на ней. Дополнительно дана выборка контрольных объектов T размера M на F, а также число k. Реализовать метод k ближайших соседей по евклидовой метрике для классификации выборки T. Функция должна возвращать вектор-столбец длины M с предсказанными ответами.

2. Решается задача классификации с двумя классами. X - обучающая выборка, y - ответы на ней. Дополнительно дана выборка контрольных объектов T размера M на F, а также число k. Реализовать метод k ближайших соседей l_1 (city block distance) для классификации выборки T. Функция должна возвращать вектор-столбец длины M с предсказанными ответами.

3. Найти номер оптимального признака для ветвления и оптимальную точку ветвления по энтропийному критерию.

4. Найти оптимальный признак для ветвления и оптимальную точку ветвления по информационному критерию.

5. Найти номер оптимального признака для ветвления и оптимальную точку ветвления по критерию Джини.

3 Юнит-тестирование

К каждой из задач необходимо написать юнит-тесты с использованием фреймворка xUnit.

1. Скачать фреймворк xUnit [6] и распаковать архив.

2. Добавить в MATLAB'е в Path папку matlab_xunit\xunit.

3. Если xUnit подключился, то команда runtests должна заработать, но выдать ошибку "No test cases found".

4. Название файлов с набором юнит-тестов должно начинаться с test.

5. Для запуска тестов можно либо запустить файл с тестами, либо выполнить в консоли команду runtests.

6. Каждой реализованной задаче должен соответствовать один набор тестов. В них должны быть протестированы все ветви кода, в том числе обработка ошибок.

7. Писать юнит-тест на некорректное число аргументов функции не нужно. Это проверяет MATLAB, и создаёт информативное исключение.

8. Проверки разной функциональности желательно делать в разных функциях. Так вы сможете понять, что сломалось, по названию теста, который перестал проходить.

9. Проверки одного блока if (одного типа ошибок) можно делать в одной функции.

10. Перед отправкой задания проверьте, что все тесты проходят при помощи команды runtests.

Пример набора тестов (файл testCalcEntropy.m):

```
function test_suite = testCalcEntropy
initTestSuite;

function testOneLabel
assertElementsAlmostEqual(calcEntropy([0 0 0]), 0);

function testTwoLabels
assertElementsAlmostEqual(calcEntropy([1 2]), 1);

function testInputIsNotNumericFails
f = @() calcEntropy({1, 2});
assertExceptionThrown(f, 'calcEntropy:XIsNotNumeric');

f = @() calcEntropy(struct());
assertExceptionThrown(f, 'calcEntropy:XIsNotNumeric');
```

Для лучшего понимания конструкции “f = @() calcEntropy(1, 2)” рекомендуется прочитать статью Anonymous Functions в документации MATLAB.

Полезные функции: assertEqual, assertElementsAlmostEqual, assertVectorsAlmostEqual, assertExceptionThrown.

Список литературы

[1] Style guide по языку MATLAB. http://www.datatool.com/downloads/matlab_style_guidelines.pdf

[2] К.В. Воронцов. Полезная информация для пользователей LaTeX. http://www.ccas.ru/voron/latex.html

[3] К.В. Воронцов. LaTeX2e в примерах. http://www.ccas.ru/voron/download/voron05latex.pdf

[4] TortoiseSVN. http://tortoisesvn.net/

[5] Краткое руководство по работе с TortoiseSVN с локальным репозиторием. http://thinkinging.com/2007/04/12/creating-a-local-subversion-repository-with-tortoisesvn/

[6] Фреймворк для юнит-тестирования XUnit. http://www.mathworks.com/matlabcentral/fileexchange/22846-matlab-xunit-test-framework