Моделирование стохастических процессов на языке Perl: Часть 1. Матричные преобразования на Perl в решении прикладных задач

Михаил Смирнов
Консультант
Независимый разработчик

14.09.2010

Первая статья серии "Моделирование стохастических процессов на языке Perl" является введением в язык данных Perl PDL (Perl Data Language) и содержит обзор основных классов матричных операторов, используемых в задачах моделирования стохастических процессов. Пакет PDL является зарегистрированным модулем архива CPAN (Comprehensive Perl Archive Network), который содержит свободные программные ресурсы, относящиеся к Perl. Остальные статьи цикла посвящены использованию пакета PDL при моделировании прикладных задач на основе Марковских процессов и разработке серверного приложения для предсказания курсов валют на основе Марковских переходных вероятностей.

1. Представление стохастических процессов матрицами переходных вероятностей

Как известно, поведение стохастических систем может описываться Марковскими процессами. Частным случаем Марковского случайного процесса является Марковская цепь. В этом случае, стохастическая система может быть представлена дискретным множеством состояний. Марковский случайный процесс с дискретными состояниями и дискретным временем называют дискретной Марковской цепью. Применение Марковских цепей в решении прикладных задач и будет предметом рассмотрения настоящего цикла статей, посвященных компьютерному моделированию с помощью пакета PDL.

На практике большинство стохастических систем могут быть описаны Марковской цепью или сведены к ней. Примером может служить аналого-цифровое преобразование непрерывного сигнала в последовательность отсчетов с дискретными значениями амплитуды. Такой процесс называется дискретизацией.

Полным описанием Марковской цепи служит матрица переходных вероятностей $\mathbf{P=(p_{i,j})}$:

```
[p0,0 p0,1 p0,2...p0,j...]
[p1,0 p1,1 p1,2...p1,j...]
[p2,0 p2,1 p2,2...p2,j...]
[...................]
[pi,0 pi,1 pi,2...pi,j...]
 [...................],
```

где условная вероятность $p_{i,j}$ является вероятностью перехода из состояния **i** в состояние **j**. Такая вероятность получила название переходной вероятности. Матрица переходных вероятностей является стохастической матрицей, если она обладает следующими свойствами: матрица **P** является квадратной, все элементы $p_{i,j} \geq 0$, и для каждой **i** строки сумма вероятностей $p_{i,j}$ равна единице:

$$\sum_{j=0}^{M-1} p_{i,j} = 1,$$

где **M** наибольшее значение перехода.

Большинство природных процессов и процессов, связанных с деятельностью человека являются случайными и образуют последовательности (цепи) переходов из одного состояния в другое, при этом для них характерно влияние предшествующих событий на последующие. Теория цепей Маркова является естественным и эффективным инструментом для анализа и описания таких процессов.

Одним из наиболее известных приложений в использовании Марковской модели являются системы массового обслуживания ("системы очередей"), связанные с реальными ситуациями, в которых имеются последовательности процедур, случайных по длительности и случайных по началу во времени. Первым интересным проектом, в этом смысле, были телефонные системы, характеризующиеся случайным потоком вызовов абонентов. Аналогичные задачи возникают при расчете нагрузок в энергетических сетях, планировании предприятий массового обслуживания, исследовании загруженности транспортных потоков и во многих других приложениях.

2. Роль языка данных Perl в матричном исчислении

Целью создания языка данных Perl PDL (Perl Data Language) являлось сосредоточение усилий на эффективном манипулировании большими матрицами данных в научных исследованиях, численном анализе, моделировании физических процессов и др. Пакет PDL привнес в стандартные функции Perl возможность компактного хранения и быстрого вычисления матриц, что весьма актуально для компьютерного моделирования. Например, в задачах цифровой обработки изображений, сложение двух матриц изображений размерностью 2048 x 2048 занимает доли секунды. Двумерный массив чисел с плавающей запятой размерностью 1024 x 1024, принимая тип данных PDL, будет занимать всего 4 Мбайта оперативной памяти, а вычисление квадратного корня такой матрицы данных займет не более секунды. Пакет PDL позволяет лаконично записывать математические операторы для работы с матрицами в виде одной строки, например, код для умножения двух матриц

будет иметь вид $a = $b x $c. Кроме того, PDL позволяет взаимодействовать со множеством внешних числовых пакетов, графикой и систем визуализации.

3. Описание основных матричных операторов PDL

Вычислительные матрицы представляют собой математический объект, эквивалентный двумерному массиву. Две матрицы одинакового размера можно поэлементно сложить или вычесть друг из друга. Чтобы выполнить умножение матриц, число столбцов в одной матрице должно совпадать с числом строк во второй. Стандартный алгоритм для умножения матриц большой размерности является процедурой весьма затратной в вычислительном отношении: сначала каждая строка одной матрицы поэлементно умножается на каждый столбец второй, а затем вычисляется сумма поэлементных произведений.

Владение, свободно распространяемым, пакетом PDL обеспечивает практически молниеносное выполнение числовых процедур. Операции с матрицами потребует значительно меньшей памяти и ресурсов процессора, по сравнению со стандартными операциями для работы с массивами. В объектах PDL, для многих математических операторов реализована идея перезагрузки, которая обеспечивает работу с конкретными типами операндов. Рассмотрим основные матричные операторы PDL на конкретных примерах.

Пример 1. Объявление типа данных PDL; задание одномерной $d1 и двумерной $d2 матриц, соответственно:

```
use PDL;
use PDL::Matrix;
$d1 = pdl [ [5,6,1,2] ];
$d2 = pdl [ [2,3,7,8] , [3,4,9,1] ];
print "d2 = ".$d2;
```

Результат:

```
$d2 = [
            [2 3 7 8]
            [3 4 9 1]
          ].
```

Пример 2. Построение диагональной единичной матрицы.

```
$dunit = zeroes(4,4);
$dunit->diagonal(0,1) ++;
print "\$dunit = ".$dunit;
```

В первой строке кода осуществляется задание размера матрицы и обнуление всех элементов с помощью функции zeroes().

Результат:

```
$dunit = [
                [1 0 0 0]
                [0 1 0 0]
                [0 0 1 0]
                [0 0 0 1]
               ].
```

Для получения нулевой матрицы с одним столбцом (вектором) элементов будем использовать оператор vzeroes():

```
$vert = vzeroes(4);
print  \$vert = .$vert;
```

Результат:

```
$vert = [
              [0]
              [0]
              [0]
              [0]
             ].
```

Пример 3. Формирование двумерной последовательности чисел с помощью функции sequence() и блочная выборка элементов матрицы:

```
$sq = sequence(10,4);
print "\$sq = ".$sq;
$sec = sec($sq,3,5);
print "\$sec = ".$sec;
```

Результат:

```
$sq = [
           [ 0  1  2  3  4  5  6  7  8  9]
           [10 11 12 13 14 15 16 17 18 19]
           [20 21 22 23 24 25 26 27 28 29]
           [30 31 32 33 34 35 36 37 38 39]
          ],

$sec = [
             [ 3  4  5]
             [13 14 15]
             [23 24 25]
             [33 34 35]
            ].
```

Матрица $sec получена выборкой элементов с 3 по 5 в каждой строке с помощью функции sec().

Пример 4. Создание Марковской матрицы переходных вероятностей и выборка строк и столбцов матрицы. По определению Марковская цепь описывается матрицей, с суммой элементов в строке равной единице для всех строк:

```
$p = [
          [0.5   0.5       0        0       0   0]
          [0.2   0.8       0        0       0   0]
          [  0   0.25    0.5     0.25       0   0]
          [0.3      0      0      0.7       0   0]
          [   0     0      0     0.35  0.65   0]
          [   0     0      0        0       0   1]
         ].
```

Выборки значений столбцов можно получить с помощью функции slice():

```
$col = $p->slice('3,:');
print "\$col = ".$col;
```

Результат выборки 3-го столбца матрицы $p:

```
$col = [
            [     0]
            [     0]
            [0.25]
            [  0.7]
            [0.35]
            [     0]
           ].
```

Выборка значений строк так же осуществляется с помощью функции slice():

```
$row = $p->slice('3:4,4');
print "\$row = ".$row;
```

В последнем примере осуществляется выборка элементов строки с 3 по 4 позицию в 4-ой строке матрицы $p:

```
$row = [
             [0.35 0.65]
            ].
```

Пример 5. Умножение матриц на примере Марковской матрицы. Как известно, важнейшим свойством Марковских процессов является память о предшествующих переходах. Для цепи Маркова с дискретным временем t, состояние системы отстоящее в цепи на h шагов определяется произведением матрицы переходных вероятностей на саму себя h раз: Ph(i,j). В PDL умножение матриц записывается одной строкой $pn = $p x $p. Напишем код позволяющий последовательно получить 2-х и 3-х шаговую Марковскую модель, соответственно:

```
$r = $p;
for($h=0;$h<2;$h++){
                $t = $h + 2;
                $pn = $r x $p;
                print "p$t = ".$pn;
                $r = $pn;
}
```

Результат:

```
p2 = [
        [  0.35     0.65        0          0          0      0]
        [  0.26     0.74        0          0          0      0]
        [ 0.125    0.325     0.25        0.3          0      0]
        [  0.36     0.15        0       0.49          0      0]
        [ 0.105        0        0      0.473      0.422      0]
        [     0        0        0          0          0      1]
       ]
p3 = [
        [0.305   0.695        0          0          0      0]
        [0.278   0.722        0          0          0      0]
        [0.218    0.39     0.12       0.272          0      0]
        [0.357     0.3        0       0.343          0      0]
        [0.194   0.053        0       0.479      0.274      0]
        [    0       0        0          0          0      1]
       ].
```

Как видно из последней распечатки, результирующая матрица так же является стохастической. Кроме перемножения вычислительных матриц с помощью оператора x в PDL предусмотрено скалярное умножение массивов данных с помощью оператора *. Последняя процедура наиболее широко используется при работе с изображениями, и действительно заслужила называться молниеносной. Так скалярное произведение двух матриц вещественных чисел с плавающей запятой и размерностью 2048 x 2048 составляет менее 1 секунды.

4. Выводы

Работа с модулями PDL ни коим образом не ограничивается матричным исчислением. Количество модулей в PDL весьма значительно и охватывает решения во многих приложениях связанных с научными изысканиями. Успешные примеры использования PDL включают построение погодных карт, численное моделирование волновых уравнений, решение газодинамических задач. Все приложения перечислить трудно. Популярность применения PDL, как уже было сказано выше – высокая скорость обработки матриц большой размерности и компактное хранение больших массивов данных с плавающей запятой.

Об авторе

Михаил Смирнов

Михаил Смирнов — независимый web-программист и разработчик прикладного программного обеспечения (Perl, PHP, Borland Bilder C++) в области информационных технологий , кандидат технических наук. Основные направления деятельности изложены на личном сайте http://www.smirnov.sp.ru и затрагивают области синтеза и цифровой обработки изображений, цифровой голографии, распознавания образов, анализа и моделирования случайных процессов.