IntermediateProblems

May 16, 2018

1 Intermediate Problems

1.1 MyRange and LinSpace Problem

Part 1

Let's create our own implementation of the range type. The `Range` type is what you get from `1:2:20`. It's form is `start:step:stop`. If you know `start`, `step`, and `stop`, how do you calculate the `i`th value? Create a type `MyRange` which stores `start`, `step`, and `stop`. Can you create a function `_MyRange(a,i)` which for `a` being a `MyRange`, it returns what `a[i]` should be? After getting this correct, use the Julia array interface in order to define the function for the `a[i]` syntax on your type.

Part 2

Do ?linspace. Make your own LinSpace object using the array interface.

http://ucidatascienceinitiative.github.io/IntroToJulia/Html/ArrayIteratorInterfaces

(Note, `linspace` has extra accuracy enhancing changes. Just do the "simple" implementation")

Part 3 Check out the call overloading notebook:

http://ucidatascienceinitiative.github.io/IntroToJulia/Html/CallOverloading

Overload the call on the UnitStepRange to give an interpolated value at intermediate points, i.e. if `a=1:2:10`, then `a(1.5)=2`.

Part 4 Do your implementations obay dimensional analysis? Try using the package `Unitful` to build arrays of numbers with units (i.e. an array of numbers who have values of Newtons), and see if you can make your LinSpace not give errors.

1.2 Operator Problem

In mathematics, a matrix is known to be a linear operator. In many cases, this can have huge performance advantages because, if you know a function which "acts like a matrix" but does not form the matrix itself, you can save the time that it takes to allocate the matrix (sometimes the matrix may not fit in memory!)

Recall the Strang matrix

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix}$$

Define a type `StrangMatrix` and define a dispatch such that `A*x` acts like a Strang matrix on a vector.

1.2.1 Advanced Bonus

Iterative solvers solve `Ax=b` and only require the definition of matrix multiplication. Thus utilize IterativeSolvers.jl to solve `Ax=b` for `b=rand(100)` using your lazy matrix type. Hint: you will need to define `A_mul_B!`. You will also need to define a different version of your Strang matrix which holds a size and has `Base.eltype` defined.

1.3 Regression Problem

```
In [ ]: #### Prepare Data For Regression Problem

        X = rand(1000, 3)                # feature matrix
        a0 = rand(3)                     # ground truths
        y = X * a0 + 0.1 * randn(1000);  # generate response

        # Data For Regression Problem Part 2
        X = rand(100);
        y = 2X  + 0.1 * randn(100);
```

Given an Nx3 array of data (`randn(N,3)`) and a Nx1 array of outcomes, produce the data matrix `X` which appends a column of 1's to the front of the data matrix, and solve for the 4x1 array β via β`X = b` using `qrfact`, or `\`, or the definition of the OLS estimator. (Note: This is linear regression).

Compare your results to that of using `llsq` from `MultivariateStats.jl` (note: you need to go find the documentation to find out how to use this!). Compare your results to that of using ordinary least squares regression from `GLM.jl`.

Regression Problem Part 2 Using your OLS estimator or one of the aforementioned packages, solve for the regression line using the (X,y) data above. Plot the (X,y) scatter plot using `scatter!` from Plots.jl. Add the regression line using `abline!`. Add a title saying "Regression Plot on Fake Data", and label the x and y axis.

1.4 Type Hierarchy Problem

Make a function `person_info(x)` where, if `x` is a any type of person, print their name. However, if `x` is a student, print their name and their grade. Make a new type which is a graduate student, and have it print their name and grade as well. If `x` is anything else, throw an error. Do not use branching (`if`), use multiple dispatch to solve the problem!

Note that in order to do this you will need to re-structure the type hierarchy. Make an AbstractPerson and AbstractStudent type, define the subclassing structure, and write dispatches on these abstract types. Note that you cannot define subclasses of concrete types!

(Not only is the multiple-dispatch way more Julian, we will see later that it also has a lot of performance enhancements due to how it interacts with the compiler).

1.5 Distribution Quantile Problem (From Josh Day)

To find the quantile of a number `q` in a distribution, one can use a Newton method

$$\theta_{n+1} = \theta_n - \frac{cdf(\theta) - q}{pdf(\theta)}$$

to have $\theta_n \to$ the value of for the `q`th quantile. Use multiple dispatch to write a generic algorithm for which calculates the `q`th quantile of any `UnivariateDistribution` in Distributions.jl, and test your result against the `quantile(d::UnivariateDistribution,q::Number` function.

Hint: Use $\theta_0 =$ mean of the distribution