Full Example

Robert Moss
Stanford University, Stanford, CA 94305
mossr@cs.stanford.edu

August 19, 2020

1 Loss Function

In mathematical optimization, statistics, decision theory and machine learning, a *loss function* or *cost function* is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event.[1] An optimization problem seeks to minimize a loss function. An objective function is either a loss function or its negative (sometimes called a *reward function* or a *utility function*), in which case it is to be maximized.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] \tag{1}$$

In statistics, typically a loss function is used for parameter estimation, and the event in question is some function of the difference between estimated and true values for an instance of data. The concept, as old as Laplace, was reintroduced in statistics by Abraham Wald in the middle of the 20th Century. In the context of economics, for example, this is usually economic cost or regret. In classification, it is the penalty for an incorrect classification of an example. In actuarial science, it is used in an insurance context to model benefits paid over premiums, particularly since the works of Harald Cramér in the 1920s. In optimal control the loss is the penalty for failing to achieve a desired value.

```
function loss_function(theta, X, y)
    m = length(y) # number of training examples
    grad = zeros(size(theta))
    h = sigmoid(X * theta)
    J = 1/m * sum((-y'*log(h))-(1 .- y)'*log(1 .- h))
    grad = 1/m*(X'*(h-y))
    return (J, grad)
end
```

[1] https://en.wikipedia.org/wiki/Loss_function

2 Sigmoid

A sigmoid function is a mathematical function having an ”S” shape (sigmoid curve). Often, sigmoid function refers to the special case of the logistic function shown in the first figure and defined by the formula:

$$\sigma(t) = \frac{1}{1 + e^{-t}} \tag{2}$$

Other examples of similar shapes include the Gompertz curve (used in modeling systems that saturate at large values of t) and the ogee curve (used in the spillway of some dams). A wide variety of sigmoid functions have been used as the activation function of artificial neurons, including the logistic and hyperbolic tangent functions. Sigmoid curves are also common in statistics as cumulative distribution functions, such as the integrals of the logistic distribution, the normal distribution, and Student's t probability density functions.[2]

```julia
# return is of size/dimensions: zeros(size(t))
sigmoid(t) = 1 / (1 + exp(-t))
```

Here's an example of plotting a sigmoid function.

```julia
using UnicodePlots
display(lineplot(sigmoid.(-5:0.1:5), title="Sigmoid",
                                     xlabel="t",
                                     ylabel="sigmoid(t)"))
```

3 Test Algorithm

This algorithm is intended to be a test of the `@tex` Julia macro. Let's *not* start a new paragraph. Here's some more text for the LaTeX document.

```julia
function test_algorithm(x::Int, y::Int, z::Int)
    a::Float64 = sqrt(3y,2x,atan(z))
    b::Float64 = 2x^2+5y-z
    c::Float64 = 10b^10
    return b::Float64, c::Float64
end
```

[2] https://en.wikipedia.org/wiki/Sigmoid_function

4 Conformal Gravity

We can immediately see the departures from the GR prediction, where as the two new factors γ and k arise due to the fact that the conformal theory is fourth order and thus must contain two additional terms. It should also be noted that when γ and k are small, we return the exact Schwarzchild solution. For a more rigorous treatment of the derivation see. Now that we have the "Schwarzchild like" solution for conformal gravity, we can effectively follow the procedure above by noting that a galaxy is a disk with exponential density falloff in the radial direction. The only other issue that conformal gravity needs to account for is local vs. global effects. Since the theory is fourth order in construction, we no long possess the power of a global guess law. Hence, the integration must be made both locally and globally which gives rise to the total rotational prediction of the galaxy as

$$v_{\mathrm{CG}}(R) = \sqrt{v_{\mathrm{GR}}^2 + \frac{N^*\gamma^*c^2R^2}{2R_0} I_1\left(\frac{R}{2R_0}\right) K_1\left(\frac{R}{2R_0}\right) + \frac{\gamma_0 c^2 R}{2} - \kappa c^2 R} \quad (3)$$

The presence of the linear and quadratic potential terms are negligible on solar system scales, but would begin to dominate at galactic scales. The key feature of the solution is the requirement that the k term be negative which forces the quadratic term to compete and eventually dominate over the linear term. The result is the termination of stable orbits in the galaxy and the ultimate fall off of a galactic rotation curve very far from the center. It is this feature that was tested in a sample of the fourteen largest studies galaxies, and due to the recent Milky Way data described above, can now be tested in our own galaxy.

```julia
function conformal_gravity(Rkpc::Float64)
    # Mannheim and O'Brien's conformal gravity contributions
    Rkm = kpc_to_km(Rkpc)
    v_rot = Array(Any, R_size)
    mod = cm_to_kpc(1)
    norm = kpc_to_km(1)
    cMod = km_to_cm(c)*mod
    BMod = km_to_cm(B)*mod
    m = (3.06*float32(10)^-30)/mod
    g = (5.42*float32(10)^-41)/mod
    k = (2*9*float32(10)^-11)
    R0kpc = km_to_kpc(R0)
    R0km = R0
    for i=1:R_size
```

```
        Xkpc = Rkpc[i]
        Xkm = Rkm[i]
        besx2 = Xkpc/(2*R0kpc)
        besx8 = Xkpc/(8*R0kpc)
        veln = norm^2*((BMod*cMod^2*Xkpc^2)/(2*R0kpc^3)) *
            (besseli(0,besx2)*besselk(0,besx2) - besseli(1,besx2) *
             besselk(1,besx2))
        velm = norm^2*((m*cMod^2*Xkpc)/2)
        velb = norm^2*(((g*cMod^2*Xkpc^2)/(2*R0kpc)) *
            (besseli(1,besx2)*besselk(1,besx2)))
        velk = norm^2*((k*cMod^2*Xkpc^2)/2)
        veln_gas = norm^2*((N_g*BMod*cMod^2*Xkpc^2)/(2*64*R0kpc^3)) *
            (besseli(0,besx8)*besselk(0,besx8)-besseli(1,besx8) *
             besselk(1,besx8))
        velb_gas = norm^2*((N_g*g*cMod^2*Xkpc^2)/(8*R0kpc)) *
            (besseli(1,besx8)*besselk(1,besx8))
        v_rot[i] = sqrt((N*(veln + velb)) + velm - velk + veln_gas +
            velb_gas + (v_bulge_n_inner(Xkpc, bulge_b, bulge_t) +
            v_bulge_b_inner(Xkpc, bulge_b, bulge_t)))
    end
    return v_rot::Float64
end
```

Similar to all other observed rotation curves, the Milky Way suffers from the same missing mass problem. The prediction set forth by general relativity (GR) can be found by starting with a single point mass solution to the field equations, and then modeling the galaxy as a collection of point masses arranged in a disk. We assume for simplicity that the disk is infinitely thin, and the distribution of mass falls exponentially as

$$\Sigma(R) = \Sigma_0 e^{-\frac{R}{R_0}} \tag{4}$$

where R_0 is the luminous scale length, and Σ_0 is the central density. Upon integrating over the disk in cylindrical coordinates, one arrives at the familiar

$$v_{\rm GR}(R) = \sqrt{\frac{N^*\beta^*c^2R^2}{2R_0^3}\left[I_0\left(\frac{R}{2R_0}\right)K_0\left(\frac{R}{2R_0}\right) - I_1\left(\frac{R}{2R_0}\right)K_1\left(\frac{R}{2R_0}\right)\right]}, \tag{5}$$

where I_0, I_1, K_0, and K_1 are Bessel functions. This is the well established Freeman curve, and is assumed that each parameter is fixed. The only free parameter in this equation then is the overall number of stars,

$$N^* = \frac{M_{disk}}{M_\odot}. \tag{6}$$

Although this is a free parameter for fitting purposes, it is physically bounded by the preservation of the mass to light ratio. Since the curve is derived as described above, then the mass to light ratio should be on the order of unity.

5 Messy

This function is to push the `@tex` macro.

```
function
messy(arg1::Int,
      arg2::UInt32,
      arg3::Vector{Float64})
    r::Float64 = arg1 + arg2 / sum(arg3)
    return r
end
```

6 F

Here's a one-liner:

```
F(x,y,c=10) = 2x^2 + 5y^3 + c
```

7 G

Here's a two-liner:

```
G(x,y,c=10) =
    2x^2 + 5y^3 + c
```