Activity 2 - One-Hot Vectors (30 minutes)

This activity is inspired by Stefan Karpinski's "The Unreasonable Effectiveness of Multiple Dispatch" talk, and Carsten Bauer's "Julia for HPC Course @ UCL ARC" workshop.

One-hot encoding

One-hot encoding is generally used in machine learning for classification tasks where there is a finite set of categories. In one-hot vectors, only one index is hot (value of 1) and all the others are cold (value of 0). An example can be seen below:

$$v = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].$$

Then, if the previous vector represented a classification of an image from the dataset of handwritten numbers, MNIST, the image would represent a "3".

Let's create a custom type

Think about what information an implementation of a one-hot vector actually has to store.

Task 1: Using `struct`, define a `OneHot` type which represents a vector with only a single hot bit.

Extending `Base` functions

Now, let's use our new type! Let's say we are interested in defining the operation

$$t = \sum_{i=1}^{N} v_i^{\mathrm{T}}\,\mathbf{A}\,v_i,$$

for a collection of one-hot vectors $\{v_1, \ldots, v_N\}$, where $\mathbf{A}$ is an $n \times n$ matrix and v_i is an $n \times 1$ one-hot vector.

In code, this operation could look like this:

```julia
function innersum(A, vs)
    t = zero(eltype(A))
    for v in vs
        y = A * v
        for i in 1:length(vs[1])
            t += v[i] * y[i]
        end
    end
    return t
end
A = rand(3,3)
vs = [rand(3) for i in 1:10] # This should be replaced by a `Vector{OneHot}`
innersum(A, vs)
```

Task 2: Extend all the necessary `Base` functions such that the previous computation works for a matrix `A` and a vector of `OneHot` vectors `vs` (i.e. `vs isa Vector{OneHot}`).

Task 3: Benchmark the speed of `innersum` when called with a vector of `OneHot` vectors (i.e. `vs = [OneHot(3, rand(1:3)) for i in 1:10]`) and when called with a vector of `Vector{Float64}` vectors, respectively. What do you observe?

Sub-types

Task 4: Now, define a `OneHotVector` type which is identical to `OneHot` but is declared to be a subtype of `AbstractVector{Bool}` and extend only the functions `Base.getindex(v::OneHotVector, i::Int)` and `Base.size(v::OneHotVector)`.

Hint: Here, the function `size` should return a `Tuple{Int64}` indicating the length of the vector, i.e. `(3,)` for a one-hot vector of length 3.

Task 5: Try to create a single `OneHotVector` and try to run the `innersum` function using the new `OneHotVector` type. What changes do you observe? Do you have to implement any further methods?