Recurrent Neural Network

Austin Bean

06/20/2019

1 nn notes

- Word embeddings from word2vec
- Train RNN before this or after?
- Which features, in other words

1.1 import some labeled data

- Word embeddings come from word2vec
- Embeddings are organized in the file train_embedding.jl
- Labels come from a Stata file
- Import data, pull out

First load the data

```
using Flux, CSVFiles, DataFrames, Word2Vec

dat1 = DataFrame(load("embedded_data.csv"));
```

Now separate the labels:

```
target = convert(Array{Float64,1}, dat1[:x1])
```

```
4956-element Array{Float64,1}:
 28.0
 40.0
 30.0
 40.0
 24.0
 24.0
 24.0
 40.0
 96.0
  6.0
  ⋮
 24.0
```

```
32.0
16.0
18.0
12.0
16.0
24.0
16.0
24.0
```

And load the inputs - in this case, the longest diet sentence is 22 words long, so there are at most 22 columns in this data.

```
input = convert(Array{Float64,2}, dat1[:,[:x2, :x3, :x4, :x5, :x6, :x7, :x8, :x9, :x10,
    :x11, :x12, :x13, :x14, :x15, :x16, :x17, :x18, :x19, :x20, :x21, :x22]])
```

```
4956×21 Array{Float64,2}:
 -0.137149    0.00755574  -0.0218786  …  0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.0336286     0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056   …  0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.00755574  -0.129056      0.0  0.0  0.0  0.0  0.0  0.0
  ⋮                                   ⋱  ⋮                        ⋮
 -0.137149    0.0716542    0.0263198     0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.0716542    0.127922      0.0  0.0  0.0  0.0  0.0  0.0
 -0.129515    0.0646356    0.0263198     0.0  0.0  0.0  0.0  0.0  0.0
 -0.0386516  -0.0672315    0.0263198  …  0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.0263198   -0.108932      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.091354    -0.108932      0.0  0.0  0.0  0.0  0.0  0.0
  0.146102    0.00719765  -0.118743      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.0263198   -0.108932      0.0  0.0  0.0  0.0  0.0  0.0
 -0.137149    0.0716542    0.0263198  …  0.0  0.0  0.0  0.0  0.0  0.0
```

Load the vector of word embeddings:

```
embed_1 = wordvectors("./diet_embed");
```

```
get_vector(embed_1, "NEOSURE")
```

```
100-element Array{Float64,1}:
  0.048023060590953875
 -0.019088513012644727
  0.09206458419563016
 -0.0351294038688977З
 -0.0016522138597125774
  0.18057508027176666
 -0.05083951571024515
 -0.1213282295740296
  0.009952371402730635
  0.13227588014740368
  ⋮
  0.10443202068050489
  0.003981116682573781
  0.13861994425284802
  0.0960978185374775
```

```
-0.012032454432930977
-0.07205140303458357
 0.07778728768060178
-0.2055739039675145
-0.006134332557238634
```

- Now setup the RNN.
- Bi-directional RNN is best, I think.
- This should take each (word, state) as an input and output a (state) to be fed back in.

1.2 Recurrent RNN from Goldberg:

- We are given an input $x_1, \ldots, x_n$. Each input is represented by a vector $x_i \in \mathbb{R}^{d_{in}}$
- In my case the inputs are the words. Each word has a single-dimensional input $\mathbb{R}$ from the embeddings

given by word2vec, though this need not be the case more generally. Single sentences are stored in the `input` table above.

- The RNN maps the vector of inputs $x_1, \ldots, x_n$ to the output $y_n \in \mathbb{R}^{d_{out}}$

$$y_{1:n} = RNN^*(x_{1:n})$$
$$y_i = RNN(x_{1:i})$$
$$x_i \in \mathbb{R}^{d_{in}}, y_i \in \mathbb{R}^{d_{out}}$$

- Think of $y_{1:i}$ as a different output for each possible $i = 1, \ldots, n$.
- $$y_{1:n} = RNN^*(x_{1:n})$$
  represents the entire sequence of outputs $y_{1:1}, y_{1:2}, y_{1:3}, \ldots$

1.3 Preserving a state over time

We can take the output of the system at some stage $i = 1, \ldots, n$ and re-input that as an input to the next step of the system. This is a recursive version of the RNN.

The RNN is defined by two functions $O(s)$ and $R(s, x)$ where $O : \mathbb{R}^{f(d_{out})} \to \mathbb{R}^{d_{out}}$ maps the current state into the output and $R(s, x) : \mathbb{R}^{f(d_{out})} \times \mathbb{R}^{d_{in}} \to \mathbb{R}^{f(d_{out})}$ maps the input x and the current state s to the output. These functions are stable over time and take inputs of constant dimension.

In other words, at each step $i = 1, \ldots, n$ we have a state vector s_{i-1} and we use this state vector as the input to the function R defining the RNN.

$$s_i = R(s_{i-1}, x_i)$$

$$y_i = O(s_i)$$

In this way, the state vector s_i preserves information about what happened in the preceding sequence of inputs all the way back to the first input (s_0, x_1). This can be seen by substituting in the functions $R()$ for s at any time period t:

$$s_t = R(s_{t-1}, x_t)$$
$$s_t = R(\overbrace{R(s_{t-2}, x_{t-1})}^{=s_{t-1}}, x_t)$$
$$s_t = R(R(R(s_{t-3}, x_{t-2}), x_{t-1}), x_t)$$
$$\vdots$$
$$s_t = R(R(R(\ldots R(s_0, x_1), \ldots), x_{t-1}), x_t)$$

The first s_0 vector can be reasonably set to 0 - presumably we have no useful information which would cause us to set it to something else. At each stage we will want to concatenate the current state s_{i-1} with the current input x_i to feed this to the function $R(s, x)$.

Features in this case are given very easily by the row of the `input` table, but this does not capture the fact that the set of features input to the model might be growing.

2 Implementation

Now define some functions.

```
"""`inp_vec(x, i; bi=false)`this function takes a set of features and returns input
    vector.Can do bidirectional as well w/ `bi = true`.Returns a one-dimensional vector
    of embeddings of words either up to `i`or from `1:i` and then `i:end` (including `i`
    twice)."""
function inp_vec(x, i; bi=false)
  if bi
    return vcat(copy(x[1:i]), reverse(x[i:end]))
  else
    return copy(x[1:i])
  end
end
```

```
Main.WeaveSandBox32.inp_vec
```