Exploring Carbide

Recreating Carbide's Approach to Backpropagation

Hendrik Schmidt and Nico Scordialo

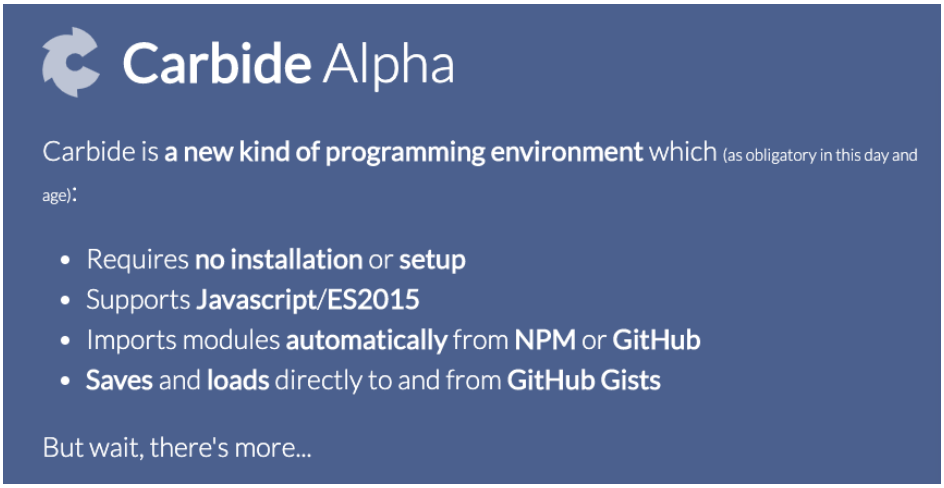
End User Development, SS2020
Software Architectures

21st July 2020

Outline

1. Motivation: Backpropagation in Carbide
2. Numbers
3. Strings
4. Collections
5. Overview and Outlook

Motivation: Backpropagation in Carbide

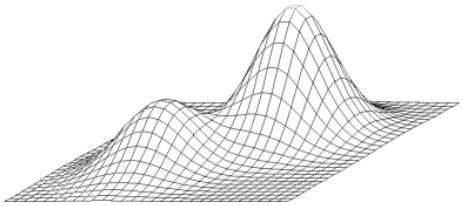


Backpropagation for Numbers

Carbide says:

> We're currently using a version of NumericJS's uncmin function ...

`uncmin`



- uses a hill-climbing algorithm for nonlinear numerical optimization
- did not always terminate correctly when we tried it out

Backpropagation for Numbers

Carbide says:

> We're currently using a version of NumericJS's uncmin function ...

`Ceres.js`

- is a numerical optimization library
- proved to be more stable when we used it

Backpropagation for Numbers

```
const weights = [46, 99, 23]
const average = array => array.reduce((a,b) => a + b)/array.length
average(weights) // 56 => changing this to 60
```

- when changing the desired output of `average(weights)` we want `weights` to adjust
- Ceres.js attempts to minimize a given function

Optimizing a cost function

- optimizing `Math.abs(average(weights) - 60)` yields fitting `weights` for the desired average

Backpropagation for Numbers

Demo

12
22
20
21

> 1 =

Backpropagation for Numbers

Limitations

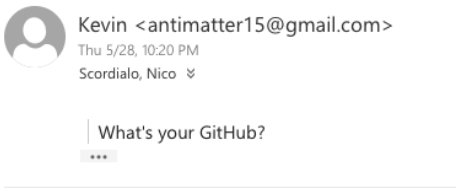
- problems that cannot be solved with hill-climbing numerical optimization
- e.g. cryptography and hashing

Backpropagation for Strings

Carbide says:

> While there have been many decades of work done on improving numerical optimization, there isn't nearly as much work done on generalizing the idea to other data structures.

Getting the source code



Backpropagation for Strings

Carbide's source code

```
// this is like totally the most important function
export function evaluate_with(compiled, targetId, params){
    var output;
    Guillermo, 4 years ago | 2 authors (Kevin Kwok and others)
    function __(probeId, value){
        if(probeId in params){
            return params[probeId]
        }else if(probeId === targetId){
            output = value;
        }
        return value
    }
    __.display = function(){}
    __.trackLoop = function(){}
    compiled.func.call(null, __);
    return output;
}
```

- did not work out of the box
- provided different functionality locally than on production

10

Backpropagation for Strings

```
const s = "hello"
const t = "world"
s.concat(t) // "helloworld" => changing this to "hellow0rld"
```

1. Find Probes

1. "hello"
2. "world"

Backpropagation for Strings

```
const s = "hello"
const t = "world"
s.concat(t) // "helloworld" => changing this to "hellow0rld"
```

2. Calculate Levenshtein Steps

- "hellowo<u>r</u>ld" to "hellow<u>0</u>rld"
- replace o with 0, represented as `[6,7,"0"]`

Backpropagation for Strings

```
const s = "hello"
const t = "world"
s.concat(t) // "helloworld" => changing this to "hellow0rld"
```

3. Filter Probes

1. ~~"hello"~~
2. "world"

Backpropagation for Strings

```
const s = "hello"
const t = "world"
s.concat(t) // "helloworld" => changing this to "hellow0rld"
```

4. Find Index for Change

1. "worl~"
2. "wor~d"
3. "wo~ld"
4. "w~rld" => "hellow~rld"

Backpropagation for Strings

```
const s = "hello"
const t = "world"
s.concat(t) // "helloworld" => changing this to "hellow0rld"
```

5. Perform Replacement

- "world" => "w0rld"

Backpropagation for Strings

Demo

Backpropagation for Strings

Limitations

```
const s = "world"
const t = s.split("")
  .map(char => char.charCodeAt(0))
  .map(code => String.fromCharCode(code-1))
  .join("") // "vnqkc" => changing this to "Vnqkc"
```

- problems that require operations other than just performing Levenshtein steps

---

Backpropagation for Collections

Carbide says:

> IM GOING TO WRITE ABOUT THIS REAL SOON NOW

- works only for simple modifications of nested elements

Overview and Outlook

Analyzed, documented and implemented

- Backpropagation for numbers ✅
- Backpropagation for array operations ✅
- Backpropagation for strings ✅

To be investigated

- Replacement in nested objects
- Differences to Carbide's production version

Hendrik Schmidt und Nico Scordialo

End User Development, SS2020
Software Architectures

21st July 2020