DD2424 - Assignement 4

Pierre Rudin

June 13, 2018

1 Introduction

The aim of this assignment is to build and train a Recurrent Neural Network to predict upcoming characters in a text. This will be done by implementing the calculations and algorithms given by the instructions for this assignment in Matlab.

1.1 Data set

The data used for this assignment is Harry Potter

2 Results

2.1 Analyticall gradient checking

The analytically computed gradients where compared to numerically computed gradients, to compute the numerical gradients the code given with the assignment was used.

2.1.1 2-Layer Network

	Max	Mean	Rel. max	Rel. mean
b	6.951524e-10	2.089486e-10	3.615587e-08	3.511327e-09
c	8.490581e-10	5.324573e-10	1.341218e-09	7.265602e-10
U	3.859340e-10	1.284493e-11	1.065243e-06	2.149218e-09
W	4.301770e-10	8.402790e-11	9.309883e-04	6.037901e-07
V	4.228567e-10	8.573024e-11	9.919559e-06	1.824390e-07

Table 1: Absolute and relative differences between analytically and numerically calculated gradients

Largest relative error is 1e-4, or .01 %, Which I consider close enough to be within margin of error.

2.1.2 Smooth loss

The network was trained for 20 epochs with $\eta = .1$ and $m = 100$. The following graph show how the smooth loss evolved during those 20 epochs.

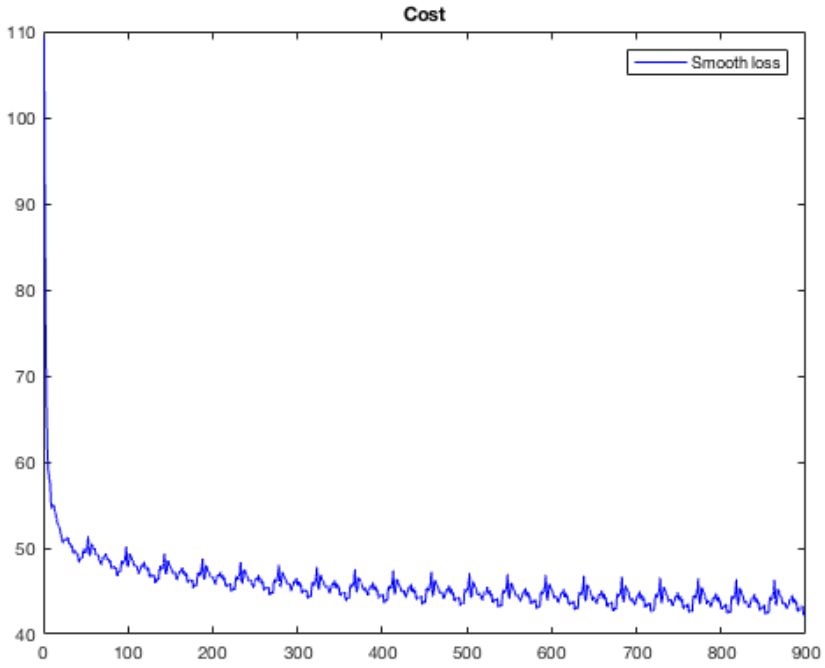

Figure 1: Smooth loss with 20 epochs of training.

2.2 Synthesized text

2.2.1 First three synthesizes

epoch: 1, iteration: 1, smooth_loss: 109.236206
}Bb:1KK!4 L2TTpy)qGjRuJzmbP?(DFK2?qTMhA4-cRHzC2pAV7avLvFigiIS3zQ16mfHA
hh')4D!'-9'CZgjpk6Qk9}6GU T
BwGVXUyrAQpfPrE_6,iSJHS,IonIh?pb1.l?6YO'_70P'xuy4jLvBht p7'J4wK1}Lui?FYXi
IwpY.beWfw:l.x̂Fox-GrNav

epoch: 1,iteration: 10001 smooth_loss: 54.721049
duld the Dosgrint sork "I'r fory the asout bobagike ath wa were jas aed on st they fo 'led affling toke cou higkteveaippankersd,s, ha wank in viccked the toith an cadbecrert anat a havened laste fre b

epoch: 1,iteration: 20001 smooth_loss: 51.507630
ing eas gikn?" "Mr werro erarout fist will to was seertued you tebt a in a Krcitwe zartid pire bith pore matiry wamain tas Chap ik whey be sering ow an chis betored teer as intheaghinste sead d "At t

2.2.2 Final three synthesizes

epoch: 20,iteration: 20001 smooth_loss: 43.479912
ild cerese ple sorty.. no beying go though at the - Mr they wgraagly brike artyed paghtly. Peass come meet worry artin; he gair, whithis creak a hige, Harry. "He he howly and in yeh it weveride, wen

epoch: 20,iteration: 30001 smooth_loss: 43.995303
rickenge stick was was currac's laher quick whing tide sic a Batie's the telon, hand the twe leat take been beeally. Harry it halp highter ovit, said? He ferilirg, but lyttwion Moody spicket plan at,

epoch: 20,iteration: 40001 smooth_loss: 43.311042
med. . . . woldnged to the -Noak and the his it refacter ous offowly tatust it blentre, Profes. "Its, qit he'd something might, his eleed hand the linding Slompiont, stopped, his wirver masthatiliest,

2.3 Best model

m = 100, η = .1
smooth_loss = 41.850540
"Oh dwy firse "Cran" were a a mady reevering e some finully fac likh bethater streached," Harry," sards git whingecret," said Mrs. Clalle, Mrs. He same hisser. "Yever a bott from Dusbs annthing oneed water theess hurring the rouch as over Kauban't arrow, Kreadn't spepter stawn Marks. "I've toid. "You've yeveraire. "Dake's greet not out oseey inting hear him facked, would veryore Sirush of there sudne mude row!" I lever out for they're scling!" said My-chere, shiss of Ced teachts at the weze you?" hot ha down's dmend, smautdy prosly lialled mise the nehe off -" Harry premoner it," on "Yloksiguply mand deres neered. "I'mid at Harry.

"Gey prypoot overearss. He dound. You re." Harry. "Ercand. "Ther, wis youls telle and stiund the figuer reeved then been Dabbe fore. Faits," "Luptur?" svery uffor he -" You the Crouch. Partly. ...." "Sways flang towh opled you nown. "I were. Harrisshry ouldn't nusn's?" saily of chever for their out -" " "No! Moody mehe, I droft way foe sate. Wh

3 Code

3.1 Main

```
clear all;
clc;
close all;
format longg

%% 0.1 Read in the data

book_fname = 'Datasets/Goblet_book.txt';
fid = fopen(book_fname,'r');
book_data = fscanf(fid,'%c');
fclose(fid);

book_chars = unique(book_data);
char_cnt = length(book_data);
char_to_ind = containers.Map('KeyType','char','ValueType','int32');
ind_to_char = containers.Map('KeyType','int32','ValueType','char');

for i = 1:length(book_chars)
    char_to_ind(book_chars(i)) = i;
    ind_to_char(i) = book_chars(i);
end;

%% 0.2 Set hyper-parameters & initialize the RNN's parameters

m = 100;
K = length(book_chars);
eta = .1;
seq_length = 25;
sig = .01;

RNN.b = zeros(m,1);
RNN.c = zeros(K,1);
RNN.U = randn(m, K)*sig;
RNN.W = randn(m, m)*sig;
RNN.V = randn(K, m)*sig;

ada.b = zeros(size(RNN.b));
```

```
ada.c = zeros(size(RNN.c));
ada.U = zeros(size(RNN.U));
ada.W = zeros(size(RNN.W));
ada.V = zeros(size(RNN.V));

%% training

epochs = 20;
losses = [];
smooth_loss = -log(1 / length(book_chars)) * seq_length;
lowest_loss = smooth_loss;
best_RNN = RNN;
iterations = round(char_cnt / seq_length - .5);

for e = 1 : epochs
    for i = 1 : iterations
        X = zeros(K, seq_length);
        Y = zeros(K, seq_length);

        X_chars = book_data((i-1)*seq_length+1:i*seq_length);
        Y_chars = book_data((i-1)*seq_length+2:i*seq_length+1);

        for p = 1:seq_length
            X(char_to_ind(X_chars(p)),p) = 1;
            Y(char_to_ind(Y_chars(p)),p) = 1;
        end;

        if isequal(mod(i, 10000), 1)
            h0 = zeros(m,1);
            synth = Synthesize(RNN, X(:,1), h0, 200);
        end

        [loss, a, h, p] = ForwardPass(RNN, X, Y, h0, seq_length);
        [RNN, ada] = BackwardPass(RNN, ada, X, Y, a, h, p,
           seq_length);

        smooth_loss = .999 * smooth_loss + .001 * loss;

        if smooth_loss < lowest_loss
            lowest_loss = smooth_loss;
            best_RNN = RNN;
        end
```

```
79
80          if isequal(mod(i, 10000), 1)
81              fprintf('epoch: %d,iteration: %d smooth_loss: %f\n',e
                    ,i,smooth_loss);
82              c = [];
83              for j = 1 : length(synth)
84                  c = [c ind_to_char(synth(j))];
85              end
86              disp(c);
87          end
88
89          if isequal(mod(i, 1000), 1)
90              losses = [losses smooth_loss];
91          end
92      end
93  end
94
95  %% Display 1000 synthesized characters
96  fprintf('Best model, smooth_loss = %f\n',lowest_loss);
97  h0 = zeros(m,1);
98  synth = Synthesize(best_RNN, X(:,1), h0, 1000);
99  c = [];
100 for j = 1 : length(synth)
101     c = [c ind_to_char(synth(j))];
102 end
103 disp(c);
104
105 %% Plot smooth_loss
106 % Plots evolution of the cost
107 figure(1);
108 x = 1:1:epochs * round(iterations / 1000 + .5);
109 plot(x,losses,'b');
110 title('Cost')
111 legend('Smooth loss')
```

3.2 Synthesize()

```
1  function synth_indexes = Synthesize(RNN, xt, ht, n)
2  %% 0.3 Synthesize text from your randomly initialized RNN
3      synth_indexes = int16.empty(n,0);
4      K = length(xt);
5      for t = 1:n
6          at = RNN.W * ht + RNN.U * xt + RNN.b;
```

```
        ht = tanh(at);
        ot = RNN.V * ht + RNN.c;
        pt = softmax(ot);

        cp = cumsum(pt);
        a = rand;
        ixs = find(cp-a > 0);
        ii = ixs(1);

        xt = zeros(K,1);
        xt(ii) = 1;

        synth_indexes(t) = ii;
    end;
```

3.3 ForwardPass()

```
function [loss, a, h, p] = ForwardPass(RNN, X, Y, h0, n)
K = length(X);
m = length(h0);

p = zeros(K, n);     % p_1, p_2, .., p_n
h = zeros(m, n+1);   % h_0, h_1, .., h_n
a = zeros(m, n);     % a_1, a_2, .., a_n
h(:, 1) = h0;
loss = 0;

for t = 1:n
    a(:, t) = RNN.U * X(:, t) + RNN.W * h(:, t) + RNN.b;
    h(:, t+1) = tanh(a(:, t));
    ot = RNN.V * h(:, t+1) + RNN.c;
    p(:,t) = softmax(ot);
    loss = loss - log(Y(:,t)' * p(:,t));
end
```

3.4 BackwardPass()

```
function [RNN, ada] = BackwardPass(RNN, ada, X, Y, a, h, p ,n)
    grads = ComputeGrads(X, Y, RNN, a, h, p, n);

%     Gradient check
%     dh = 1e-4;
%     num_grads = ComputeGradsNum(X, Y, RNN, dh);
%     for f = fieldnames(RNN)'
```

```
%         num_g = num_grads.(f{1});
%         ana_g = grads.(f{1});
%
%         diff = abs(num_g - ana_g);
%         max_diff = max(diff(:));
%         avg_diff = mean(diff(:));
%         rel_diff = abs(num_g - ana_g)./abs(num_g + ana_g);
%         rel_diff(isnan(rel_diff)) = 0;
%         max_rel_diff = max(rel_diff(:));
%         avg_rel_diff = mean(rel_diff(:));
%
%         fprintf('Field name: %s, max diff: %d, average diff: %d
    , max relative diff: %d, average relative diff: %d\n', f{1},
    max_diff, avg_diff, max_rel_diff, avg_rel_diff);
%     end

    eta = .1;
    eps = 1e-8;

    for f = fieldnames(grads)'
        grads.(f{1}) = max(min(grads.(f{1}), 5), -5);
    end

    for f = fieldnames(RNN)'
        ada.(f{1}) = ada.(f{1}) + grads.(f{1}).^2;
        RNN.(f{1}) = RNN.(f{1}) - eta * (grads.(f{1}) ./ sqrt(ada
            .(f{1}) + eps));
    end;
```

3.5 ComputeGrads()

```
function grads = ComputeGrads(X, Y, RNN, a, h, p, n)
    m = length(h);

    grad_o = (p - Y)';
    grads.c = (sum(grad_o))';
    grads.V = grad_o' * h(:,2 : end)';

    grad_h = grad_o(n, :) * RNN.V;

    grad_a = zeros(n, m);
    grad_a(n, :) = grad_h * diag(1 - (tanh(a(:, n))).^2);

```

```
13      for t = n-1 : -1 : 1
14          grad_h = grad_o(t, :) * RNN.V + grad_a(t+1, :) * RNN.W;
15          grad_a(t, :) = grad_h * diag(1 - (tanh(a(:, t))).^2);
16      end
17
18      grads.b = (sum(grad_a))';
19
20      grads.U = grad_a' * X';
21      grads.W = grad_a' * h(:,1 : end-1)';
```