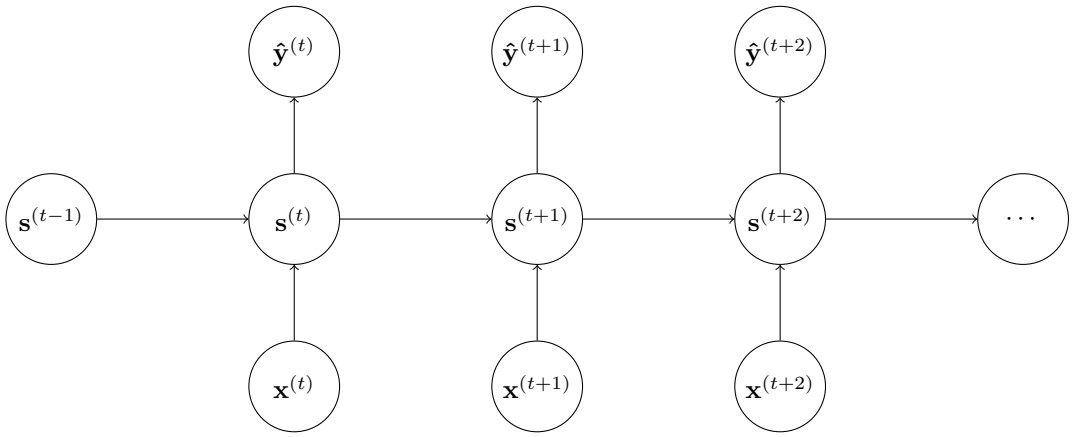
Assignment 2: Language Modelling

–,–

28 Marzo 2017

1 Question 1

a)



b)

The sigmoid function is designed for a 2-class problem and in the ouput layer we would like to have the probability of each possible $\widehat{y}$ (that can have a large number of possibilities). Although an adaptation can be made to use $(n-1)$ sigmoids functions in the output layer instead of the softmax, it is more convinient to use the softmax because it give a distribution for all the possibilities of $\widehat{y}$.

c), d)

```
function sigmoid(x)
    1./(1+exp(-x))
end
function softmax(x)
    exp(x)/sum(exp(x))
end

U = [.5 .3 ;.4 .2]
V = [.2 .5 .1; .6 .1 .8]
W = [0.4 0.2; 0.3 0.1; 0.1 0.7]
x1 = [0;1;0]
s0 = [.3;.6]
s1 = sigmoid(V*x1 + U*s0)
y1 = softmax(W*s1)
println(s1)
println(y1)
```

s1 = [0.696,0.584]

y1 = [0.337,0.297,0.366]

Calculations:

U*s0 =

$$\begin{pmatrix} .5 & .3 \\ .4 & .2 \end{pmatrix} \begin{pmatrix} .3 \\ .6 \end{pmatrix} + \begin{pmatrix} .2 & .5 & .1 \\ .6 & .1 & .8 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} .83 \\ .34 \end{pmatrix}$$

applying the sigmoid function to each element, gives s1: **s1 = [0.696,0.584]**

$$\begin{pmatrix} .4 & .2 \\ .3 & .1 \\ .1 & .7 \end{pmatrix} \begin{pmatrix} .696 \\ .584 \end{pmatrix} = \begin{pmatrix} .395 \\ .267 \\ .478 \end{pmatrix}$$

applying the softmax function to each element, gives y1: **y1 = [0.337,0.297,0.366]**

2 Question 2

a)

$d^{(k)}$ is one hot encoding (just one entry contains a one). In consequence, if we only take the corresponding entry of the vector y and take the log of this number, is equivalent to the expression (7) in the assignment sheet.

```
println(-log(.366))
println(-log(.208))
println(-log(.317))
```

-log(.366) = 1.005
-log(.208) = 0.506
-log(.317) = 1.145

d)

With BPTT we can take into consideration more layers for the current output, intuitively, we are considering words that we have previously seen (context) for compute the current $\widehat{y}$. This is important because sometimes the next word might have a dependence structure with more than one word in the past. In standard back propagation, we do not consider this direct dependence in previous words, we just want a method to *backpropagate* the error to find the optimal values of our weights.

According with the lecture and Mikolov et al., this method has improved the performance for language modelling. The disadvantages are that, as we are considering small numbers (gradients ¡ 1), when we make the multiplication between them, this number goes to zero very fast. As a consequence, new solutions as LSTM had been given.

3 Question 3

a) Parameter tunning

Table 1: My caption

Hidden units	Learning rate	Look-back	Loss
25	0.5	0	5.0758
25	0.5	2	5.0532
25	0.5	5	5.0899
25	0.1	0	5.3104
25	0.1	2	5.2904
25	0.1	5	5.2886
25	0.05	0	5.5192
25	0.05	2	5.5188
25	0.05	5	5.4740
50	0.5	0	5.0644
50	0.5	2	***5.0507***
50	0.5	5	5.1373
50	0.1	0	5.2767
50	0.1	2	5.2439
50	0.1	5	5.2226
50	0.05	0	5.4027
50	0.05	2	5.4282
50	0.05	5	5.4077

In this parameter tunning I found that the best combination was hidden units = 50, learning rate = 0.5 and look-back = 2.

Analysing the look-back parameter: There are 6 combinations of learning rate and hidden units (25, .05), (25, .1), (25, .5), (50, .05), (50, .1), (50, .5). In 4 of them when the look-back is increased, the loss is a monotonically decreasing function, but for two of them (50, .5), (25, .5), this is not a monotonically decreasing function , so it might be an interaction between large look-back and large learning rates.

Analysing the learning rate parameter: There is a consistent relationship that between a larger learning rate (at least with the values given and 10 epochs). This may be because we can *learn more* in each step and still decreasing.

Analysing the number of hidden units: We have 9 combinations of learning-rate and look-back. In 7 of them, more hidden layers imply a better result. So it might suggest that more hidden layers imply a better result (at least for this experiment)

b) train

```
Retained 2000 words from 38444 (84.00% of all tokens)


Training model for 10 epochs
training set: 25000 sentences (batch size 100)
Optimizing loss on 1000 sentences
Vocab size: 2000
Hidden units: 50
Steps for back propagation: 2
Initial learning rate set to 0.5, annealing set to 5

calculating initial mean loss on dev set: 8.51790537669

epoch 1, learning rate 0.5000 instance 25000  new loss: 5.05989714529
epoch 2, learning rate 0.4167 instance 25000  new loss: 4.83765971886
epoch 3, learning rate 0.3571 instance 25000  new loss: 4.71860219517
epoch 4, learning rate 0.3125 instance 25000  new loss: 4.65802435299
epoch 5, learning rate 0.2778 instance 25000  new loss: 4.60486724507
epoch 6, learning rate 0.2500 instance 25000  new loss: 4.58907794693
epoch 7, learning rate 0.2273 instance 25000  new loss: 4.53861274363
epoch 8, learning rate 0.2083 instance 25000  new loss: 4.50926045915
epoch 9, learning rate 0.1923 instance 25000  new loss: 4.49982859479
epoch 10, learning rate 0.1786  instance 25000  new loss: 4.46668947832

training finished after reaching maximum of 10 epochs
best observed loss was 4.46668947832, at epoch 10
setting U, V, W to matrices from best epoch


Unadjusted: 95.612
Adjusted for missing vocab: 160.727
```

c) Generate sequence

['<s>', 'executive', 'UUUNKKK', 'had', 'what', 'of', 'UUUNKKK', 'by', 'article', ';', 'UUUNKKK', 'UUUNKKK', 'of', 'a', 'UUUNKKK', 'UUUNKKK', ',', '(', 'UUUNKKK', 'UUUNKKK', ',', 'UUUNKKK', 'drug', 'and', 'was', 'a', 'UUUNKKK', 'of', 'UUUNKKK', 'on', 'UUUNKKK']
Loss: 104.455
Perplexity: 32.519

['<s>', 'among', 'provide', 'UUUNKKK', 'of', 'the', 'last', 'loans', 'UUUNKKK', 'UUUNKKK', 'system', '</s>']
Loss: 52.684
Perplexity: 120.231

['<s>', '“', 'the', 'UUUNKKK', ',', 'supply', 'court', 'UUUNKKK', 'because', 'UUUNKKK', 'the', 'group', 'of', 'attempt', 'also', 'just', 'with', 'the', 'sold', 'account', '.', '</s>']
Loss: 96.359
Perplexity: 98.349