APM 598: Homework 3 (04/03)

1 n-gram models

Ex 1.

a) Load and tokenize the text attached *'data_HW3_Plato_Republic.txt'*.
Put all the words in lower case to regroup words like 'The' and 'the'.
Compute the total number of words T in the text and the number of unique words (size of the vocabulary).

b) Build a uni-gram. Deduce the 5 most common words with **at least 8 characters**.
Hint: use the method 'most_common' on an object 'nltk.FreqDist'.

c) Build a bi-gram and define a function that given two words (ω_1, ω_2) compute the probability:

$$\mathbb{P}(\omega_2|\omega_1) = \frac{\#\{(\omega_1, \omega_2)\}}{\#\{\omega_1\}}$$

where # denotes the number of occurences of the word (or pair of words) in the corpus.

d) Deduce the so-called perplexity of the bi-gram model defined as:

$$PP = \left(\prod_{k=1..(T-1)} \mathbb{P}(\omega_{k+1}|\omega_k) \right)^{-\frac{1}{T-1}}$$

where T denotes the total number of words in the corpus.

2 Recurrent Neural Networks

Ex 2.

The goal of this exercise is to experiment with a simple Recurrent Neural Network (RNN) model for predicting letters. We only consider four letters "h", "e", "l" and "o" that we embed in $\mathbb{R}^4$:

$$"h" \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad "e" \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad "l" \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad "o" \rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

We consider a RNN with hidden states $\mathbf{h}_t$ in $\mathbb{R}^2$:

$$\begin{cases} \mathbf{h}_t &= \tanh(R\mathbf{h}_{t-1} + A\mathbf{x}_t) \\ \mathbf{y}_t &= B\mathbf{h}_t \end{cases} \tag{1}$$

where $A \in \mathcal{M}_{2,4}(\mathbb{R})$, $R \in \mathcal{M}_{2,2}(\mathbb{R})$ and $B \in \mathcal{M}_{4,2}(\mathbb{R})$ (e.g. A is a 2×4 matrix).

a) Given the input "*hello*" (i.e. $\mathbf{x}_1 = (1,0,0,0), \ldots, \mathbf{x}_5 = (0,0,0,1)$), the initial state $\mathbf{h}_0 = (0,0)$ and the matrices:

$$A = \begin{bmatrix} 1 & -1 & -1/2 & 1/2 \\ 1 & 1 & -1/2 & -1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 1/2 & 1 \\ -1 & 0 \\ 0 & -1/2 \end{bmatrix},$$

find the output $\mathbf{y}_1, \ldots, \mathbf{y}_5$ and deduce the predicted characters (see figure 1).

b) Find (numerically) matrices A, R, B such that the predicted characters are "*olleh*".

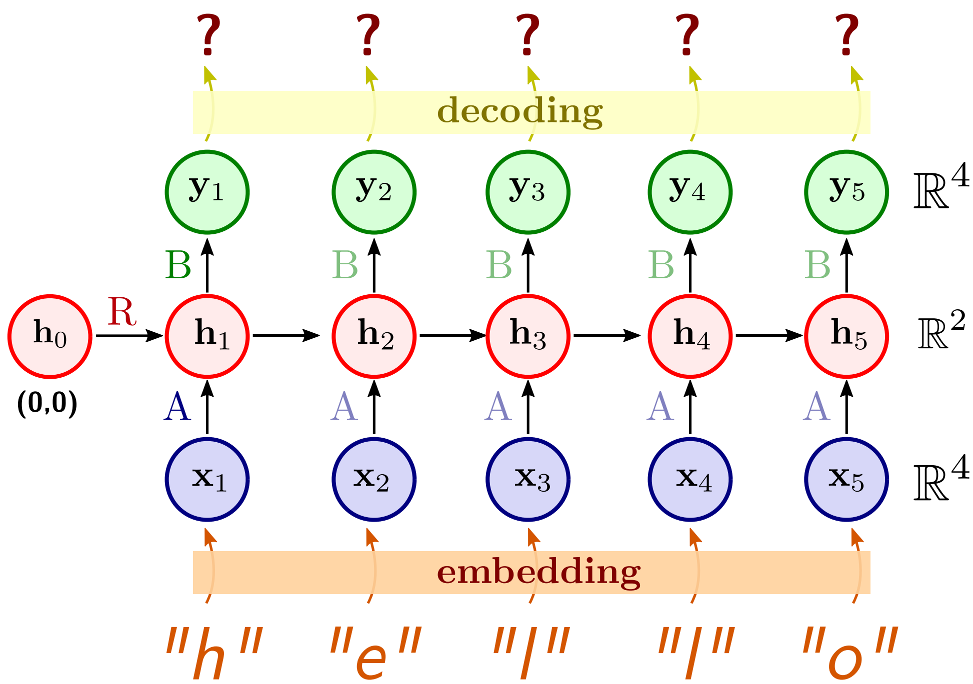

Figure 1: Predictions of a vanilla RNN. After encoding the letters (e.g. "h") into vectors (e.g. $\mathbf{x}_1 = (1,0,0,0)$), the network performs the operations described in eq. (1) to estimate a vector prediction (e.g. $\mathbf{y}_1$). The 'letter' predicted is chosen as the index of the output with the largest value (i.e. find the hot vector the closest to (softmax) of $\mathbf{y}_1$).

Ex 3. [vanishing/exploding gradient]

We would like to illustrate one of the issue with *vanilla RNN*, namely the vanishing or exploding gradient phenomenon. Rather than computing the gradient of the loss function, we simply are going to investigate how a small perturbation in the input $\mathbf{x}_1$ will affect the output $\mathbf{y}_t$ (see figure 2).

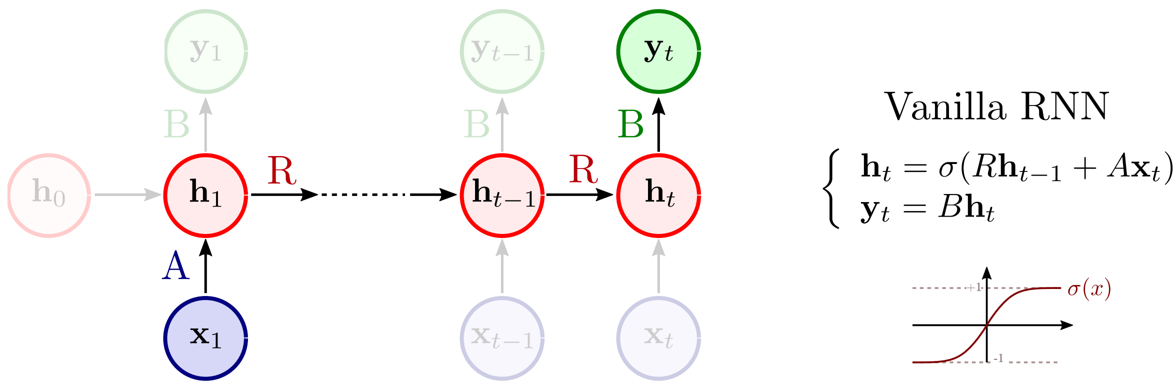

Figure 2: To study how a perturbation of $\mathbf{x}_1$ affects $\mathbf{y}_t$, we suppose in this exercise that $\mathbf{x}_2 = \ldots \mathbf{x}_t = \mathbf{0}$ and $\mathbf{h}_0 = \mathbf{0}$. Due to the iterations of the matrix R in the estimation of $\mathbf{y}_t$, the perturbation of $\mathbf{x}_1$ could have small or large influence on $\mathbf{y}_t$.

We consider a standard RNN defined with three matrices A, R, B and $\sigma(x) = \tanh(x)$ (see figure 2).

a) Compute the differential $D_{\mathbf{h}_{t-1}}\mathbf{h}_t$, i.e. compute the differential of the function $\mathbf{h} \to \sigma(R\mathbf{h} + A\mathbf{x}_t)$.
Deduce that:

$$\|D_{\mathbf{x}_1}\mathbf{y}_t\| \leq \|B\| \cdot \left(\prod_{k=1}^{t-1} |\sigma'(R\mathbf{h}_{k-1} + A\mathbf{x}_k)|_\infty\right) \cdot \|R\|^{t-1} \cdot \|A\|, \tag{2}$$

where $\|.\|$ is a (multiplicative) matrix norm and $|\sigma'(\mathbf{h})|_\infty = \max(|\sigma'(h_1)|, \ldots, |\sigma'(h_d)|)$ where d is the dimension of the vector $\mathbf{h}$.

b) From now on, we take $t = 30$ and suppose $\mathbf{x}, \mathbf{y}, \mathbf{h} \in \mathbb{R}^2$ with:

$$A = B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; R = \begin{bmatrix} \frac{1}{2} & -1 \\ -1 & \frac{1}{2} \end{bmatrix}, \quad \mathbf{x}_2 = \mathbf{x}_3 = \cdots = \mathbf{x}_{30} = \mathbf{h}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Denote $\mathbf{x}_1 = (0, 0)$ and $\mathbf{y}_{30}$ the output after $t = 30$ iterations.
Similarly, denote the perturbation $\mathbf{x}_1^\varepsilon = (\varepsilon, -\varepsilon)$ and $\mathbf{y}_{30}^\varepsilon$ the output after $t = 30$ iterations starting from $\mathbf{x}_1^\varepsilon$.
Compute and plot (in log-log scale) the difference $\|\mathbf{y}_{30} - \mathbf{y}_{30}^\varepsilon\|$ for $\varepsilon \in (10^{-4}, \ldots, 10^{-9})$.
Explain the result using eq. (2).

c) Proceed similarly as b) using $\mathbf{x}_1 = (2, 1)$ and $\mathbf{x}_1^\varepsilon = (2 + \varepsilon, 1 - \varepsilon)$.
Why does the perturbation have a small effect in this case compare to b)? Use eq. (2) to explain it.

Extra) Proceed similarly as b) using $\mathbf{x}_1 = (0,0)$ and $\mathbf{x}_1^\varepsilon = (\varepsilon, \varepsilon)$. Why is the perturbation having a small effect? In general, let $\widetilde{\mathbf{x}}_1 = (\varepsilon, \delta)$ with (ε, δ) small and $\widetilde{\mathbf{y}}_{30}$ the output of the network, do you expect $\|\mathbf{y}_{30} - \widetilde{\mathbf{y}}_{30}\|$ to be small ($\ll 1$) or large (≈ 1)?