NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Speech and Language Processing

Spring semester 2021-2022

2nd Assignment

Exercise 1

Consider two finite voice signals $x(n)$ and $y(n)$, $0 \leq n \leq N-1$ (with zero values outside the analysis window). For LPC analysis with the 'autocorrelation function' method, the autocorrelations

$$R_x(k) = \sum_{n=0}^{N-1-k} x(n)x(n+k), \quad R_y(k) = \sum_{n=0}^{N-1-k} y(n)y(n+k) \tag{1}$$

are needed, which by the Levinson method give us the corresponding optimal LPC coefficients

$$a_x = (a_{x0}, a_{x1}, \ldots, a_{xp}), \quad a_y = (a_{y0}, a_{y1}, \ldots, a_{yp}) \tag{2}$$

with $a_{x0} = a_{y0} = -1$.

1. Prove that the energy of wrong prediction (for $x(n)$) is equal to

$$E_x = \sum_{n=0}^{N-1+p} \left(\sum_{k=0}^{p} a_{xk} x(n-k) \right)^2 = a_x R_x a_x^T \tag{3}$$

   where R_x is a $(p+1) \times (p+1)$ matrix.

2. If linear prediction of the signal $x(n)$ is performed with the optimal coefficients of the signal $y(n)$, prove that the energy of the new hybrid prediction error is equal to

$$E_{xy} = \sum_{n=0}^{N-1+p} \left(\sum_{k=0}^{p} a_{yk} x(n-k) \right)^2 = a_y R_x a_y^T \tag{4}$$

3. Find the range of the ratio E_{xy}/E_x

Exercise 2

Consider in a phoneme sequence the modeling of alternation between unvoiced (U) and voiced (V) sounds with a 4-state HMM model (with parameters λ) with the following probabilities

	State 1	State 2	State 3	State 4
P(V)	0.6	0.7	0.2	0.25
P(U)	0.4	0.3	0.8	0.75

Assume the following state transition probabilities

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 0.15 & 0.3 & 0.3 & 0.25 \\ 0.3 & 0.15 & 0.3 & 0.25 \\ 0.3 & 0.25 & 0.15 & 0.3 \\ 0.25 & 0.3 & 0.25 & 0.2 \end{bmatrix} \tag{5}$$

and equal probabilities for the initial state

$$\pi_i = 0.25, \quad i = 1,2,3,4. \tag{6}$$

We observe the sequence $O_1 O_2 \cdots O_{10}$:

$$\mathbf{O} = (UVUVVUVUVU) \tag{7}$$

1. Compute the probabilities

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 \ldots q_{t-1}, q_t = i, O_1 O_2 \ldots O_t | \lambda], \quad i = 1,2,3,4, \quad t = 1, \ldots, 10 \tag{8}$$

2. Find the most probable sequence of states $\mathbf{Q}^* = (q_1, q_2, \ldots, q_{10})$.

3. Compute the probability $P^* = (\mathbf{O}, \mathbf{Q}^* | \lambda)$

For questions (1) and (2) use the Viterbi algorithm.

Exercise 3

1. The following phonetic lexicon is given:

any	eh n iy
e.	iy
many	m eh n iy
men	m eh n
per	p er
persons	p er s uh n z
sons	s uh n z
suns	s uh n z
to	t uw
tomb	t uw m
too	t uw
two	t uw

2. Compute the phonological distance between two words of the lexicon. Assume that the distance between the phonemes {uh, uw} and the phonemes {er, eh} is half the distance between two random phonemes. The same holds for the plosive consonants {p,t}, the nasal {m,n} and the fricative {s,z}.
   The distance for deletion or insertion of a phoneme is 1.2, while the cost of substitution of two random phonemes is 1.0.
   Note: Design the finite state machine that computes the (smallest) phonological distance between two words.

Solve using the OpenFst library.

Exercise 4

The following phonetic lexicon is given:

any	eh n iy
e.	iy
many	m eh n iy
men	m eh n
per	p er
persons	p er s uh n z
lessons	l eh s uh n z
sons	s uh n z
suns	s uh n z
sunset	s uh n z eh t
to	t uw
tomb	t uw m
too	t uw
two	t uw

1. Desigin a transducer that maps sequences of phonemes to words and get its closure. This is the phonetic lexicon.
2. Using the phonetic lexicon find all the possible sentences (sequences of words) that correspond to the sequence of phonemes 't uw m eh n iy p er s uh n z'
3. Find the optimal (lowest cost) sentence in (b) if the language model is a bigram with the following cost: cost(a — b) = abs (length(a) – length(b)). Therefore the cost when the word 'a' follows the word 'b' is the absolute value of the difference of the length (in characters) of the two words (Note: First, design the finite state machine that corresponds to the bigram language model and afterwards compose it with the phonetic lexicon).

Solve using the OpenFst library.

Exercise 5

1. The lexicon of a fictional children's language consists of the following syllables: {Ba, Da, Ga, Cha}. A linguist collected the following data from one-year old children:

   Ba Ba Cha Da Da Da Cha Cha Ga Cha Ba Ba Ga Ga Cha Da Da Da Ga Da Da Da Ba Da Ba Cha Cha Ga Cha Da Da Da Ba Cha Ba Ba Ba Da Da Da Cha Cha Ba Ba Ba Ga Ga Cha Da Da Da Ga Da Da Da Da Ba Ba Cha Cha Ga Cha Da Da Da Ba Ba Ba Cha

   The linguist knows that all the words of this child language contain one or two syllables, e.g., Ba, Da, Ga, Cha, BaBa, BaDa, BaGa, BaCha, DaBa, DaDa. Words with one syllable appear just as often as words with two syllables, that is, P(Ba)+P(Da)+P(Ga)+P(Cha) = 0.5

   (a) Calculate the most probable word of two syllables.

   (b) Calculate the most probable order of words in the above order of syllables collected by the linguist.

Solve using OpenFst.

Exercise 6

Back propagation through time: The following RNN is given

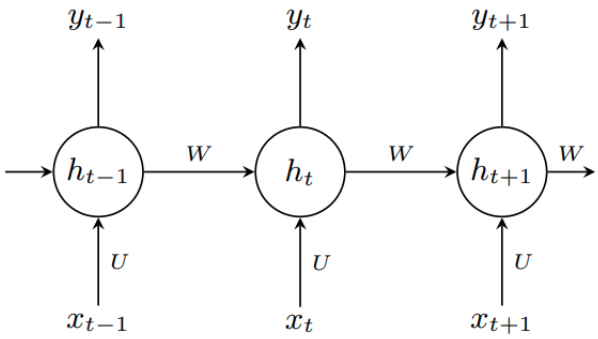
Each state h_t is given by the following pair of equations

$$h_t = \sigma(Wh_{t-1} + Ux_t), \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

Let L be the error function, which is defined as the sum of all the individual errors L_t at each time point t up to the time horizon T. That is, $L = \sum_{t=0}^{T} L_t$, where each individual error term depends on the state h_t.

Based on the above, derive the derivative of the error function with respect to the weight matrix W.

a) Given that $y = \sigma(Wx)$ where $y \in \mathbb{R}^n, x \in \mathbb{R}^d$ and $W \in \mathbb{R}^{n \times d}$, show that $\frac{\partial y}{\partial x} = \text{diag}(\sigma')W \in \mathbb{R}^{n \times d}$ for the Jacobian.



b) Show that $\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \sum_{k=1}^{t} \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$

Vanishing / Exploding Gradients In this section we will examine the problems that arise in vanilla RNN architectures and specifically the problem of vanishing and exploding gradients. We will rely on the values of the weight matrix in order to study these problems.

c) For time horizon value $T = 3$ write the complete form for the equation of part b. Show visually that if we wanted to perform backpropagation for n time steps, we would have to multiply the matrix $diag(\sigma' W)$ by itself n times.

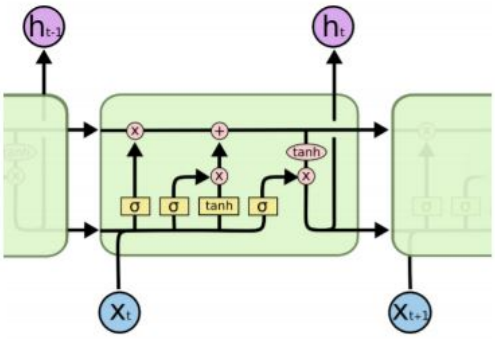
d) Each diagonizable matrix M can be represented through its eigenvalues and eigenvectors and specifically in the form $M = Q\Lambda Q^{-1}$ where Q is the matrix whose i-ith column is the $i-$th eigenvector of M and Λ is a diagonal matrix with the corresponding eigenvalues on the diagonal. Show that the product $\prod_{i=1}^{n} M$ can be written as $M^n = Q\Lambda^n Q^{-1}$

e) Consider the matrix of weights $W = \begin{pmatrix} 0.39 & 0.12 \\ 0.12 & 0.46 \end{pmatrix}$. The eigendecomposition of the matrix is:

$$W = Q\Lambda Q^{-1} = \begin{pmatrix} -0.8 & 0.6 \\ 0.6 & 0.8 \end{pmatrix} \begin{pmatrix} 0.3 & 0 \\ 0 & 0.55 \end{pmatrix} \begin{pmatrix} -0.8 & 0.6 \\ 0.6 & 0.8 \end{pmatrix}$$

Compute W^{30}. What do you observe? In the general case, what happens when the absolute value of W's eigenvalues is less than, greater than, or equal to 1? Analyze the three cases.

LSTMs: A recurrent network architecture that solves the problem of vanishing / exploding gradients is the Long Short Term Memory network (LSTM). The architecture and operations performed by the network are shown in the figure (the $\odot$ symbol indicates element-wise multiplication – hadamard product):



$$f_t = \sigma(W_f h_{t-1} + U_f x_t)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t)$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t)$$
$$\tilde{C}_t = \tanh(W_c h_{t-1} + U_c x_t)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t \odot \tanh(C_t)$$

f) Read this article and briefly explain the role of gates f_t, i_t and o_t

g) Explain which of the quantities are always positive (or zero)

To understand how LSTM tackles the problem of vanishing gradients we need to calculate the partial derivatives $\frac{\partial L}{\partial \theta}$, where θ the network parameters (W_f, W_o, W_i, W_c). In the case of LSTM, instead of the hidden state h_t we are interested in the cell state C_t. Like h_t in simple RNNs, C_t depends on the previous values $C_{t-1}, \ldots C_0$. This leads to a simplified equation of form:

$$\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \sum_{k=1}^{t} \frac{\partial L}{\partial C_t} \frac{\partial C_t}{\partial C_k} \frac{\partial C_k}{\partial W}$$

h) The equation is simplified, as we ignore dependencies from the terms $f_t, i_t, \tilde{C}_t$ for C_t. Are we interested in dependence on these terms to study the phenomenon of vanishing gradients? Why?

i) Given that:

$$\frac{\partial C_t}{\partial C_k} = \prod_{i=k+1}^{t} \frac{\partial C_i}{\partial C_{i-1}}$$

and if you consider that $f_t = 1$ and $i_t = 0$ calculate the quantity $\frac{\partial C_t}{\partial C_k}$.

j) (Bonus) Show that in the general case the recursive relation is of the form

$$\frac{\partial C_t}{\partial C_{t-1}} = \sigma'() \cdot W_f \cdot \delta \odot C_{t-1} + f_t + \sigma'() \cdot W_i \cdot \delta \cdot \tilde{C}_t + i_t \odot tanh'()\delta,$$

where $\delta = o_{t-1} \odot tanh'(C_{t-1})$.

Why after all is it better to use the cell state than the hidden state (regarding vanishing gradients)?

Hint: Remember the product rule for differentiation. It also applies to the hadamard product: $(x \odot f(x))' = x' \odot f(x) + x \odot f'(x)$

Exercise 7

Key-Query-Value self-attention in neural networks: In Transformers, we perform self-attention, which roughly means that we draw the keys, values, and queries from the same data. More precisely, let $\{x_1, \ldots, x_n\}$ be a sequence of vectors in $\mathbb{R}^d$. Think of each x_i as representing word i in a sentence. One form of self-attention[1] defines keys, queries, and values as follows. Let $V, K, Q \in \mathbb{R}^{d \times d}$ be parameter matrices. Then

$$v_i = V x_i \quad i \in \{1, \ldots, n\} \tag{9}$$
$$k_i = K x_i \quad i \in \{1, \ldots, n\} \tag{10}$$
$$q_i = Q x_i \quad i \in \{1, \ldots, n\} \tag{11}$$

Then we get a context vector for each input i; we have $c_i = \sum_{j=1}^{n} \alpha_{ij} v_j$, where α_{ij} is defined as $\alpha_{ij} = \frac{\exp(k_j^\top q_i)}{\sum_{\ell=1}^{n} \exp(k_\ell^\top q_i)}$. Note that this is single-headed self-attention.

In this problem, we will show how key-value-query attention like this allows the network to use different aspects of the input vectors x_i in how it defines keys, queries, and values. Intuitively, this allows networks to choose different aspects of x_i to be the "content" (value vector) versus what it uses to determine "where to look" for content (keys and queries.)

1. First, consider if we didn't have key-query-value attention. For keys, queries, and values we'll just use x_i; that is, $v_i = q_i = k_i = x_i$. We'll consider a specific set of x_i. In particular, let u_a, u_b, u_c, u_d be mutually orthogonal vectors in $\mathbb{R}^d$, each with equal norm $\|u_a\| = \|u_b\| = \|u_c\| = \|u_d\| = \beta$, where β is very large. Now, let our x_i be:

$$x_1 = u_d + u_b \tag{12}$$
$$x_2 = u_a \tag{13}$$
$$x_3 = u_c + u_b \tag{14}$$

   If we perform self-attention with these vectors, what vector does c_2 approximate? Would it be possible for c_2 to approximate u_b by adding either u_d or u_c to x_2? Explain why or why not.

2. Now consider using key-query-value attention as we've defined it originally. Using the same definitions of x_1, x_2 and x_3 as in part 1, specify matrices K, Q, V such that $c_2 \approx u_b$, and $c_1 \approx u_b - u_c$.
   Hint: First find V such that $v_1 = u_b$ and $v_3 = u_b - u_c$, then work on Q and K.

[1] In this case we ignore the scaling factor $1/\sqrt{d_k}$ that is proposed at "Attention is all you need" (Vaswani et al.)