UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

ECE 417 Multimedia Signal Processing
Spring 2021

FINAL EXAM

Monday, December 13, 2021, 8:00-11:00am

- This is a CLOSED BOOK exam.
- You are permitted two sheets of handwritten notes, 8.5x11.
- Calculators and computers are not permitted.
- If you're taking the exam online, you will need to have your webcam turned on. Your exam will be provided by e-mail and on zoom at exactly 8:00am; you will need to photograph and upload your answers by exactly 11:00am.
- There will be a total of 200 points in the exam. Each problem specifies its point total. Plan your work accordingly.
- You must SHOW YOUR WORK to get full credit.

Name: ______________________________

netid: ______________________________

Signal Processing and Linear Prediction

$$s[n] = Gx[n] + \sum_{m=1}^{N} a_m s[n-m] = h[n] * x[n]$$

$$H(z) = \frac{G}{1 - \sum_{m=1}^{N} a_m z^{-m}} = \frac{G}{\prod_{k=1}^{N}(1 - p_k z^{-1})} = \sum_{k=1}^{N} \frac{C_k}{1 - p_k z^{-1}}$$

$$\mathcal{E} = \sum_{n=-\infty}^{\infty} e^2[n] = \sum_{n=-\infty}^{\infty} \left(s[n] - \sum_{m=1}^{p} a_m s[n-m] \right)^2$$

$$0 = \sum_{n=-\infty}^{\infty} \left(s[n] - \sum_{m=1}^{p} a_m s[n-m] \right) s[n-k], \quad 1 \le k \le p$$

$$\vec{\gamma} = R\vec{a}$$

Image Interpolation

$$y[n_1, n_2] = \begin{cases} x\left[\frac{n_1}{U}, \frac{n_2}{U}\right] & \frac{n_1}{U}, \frac{n_2}{U} \text{ both integers} \\ 0 & \text{otherwise} \end{cases}$$

$$z[n_1, n_2] = h[n_1] *_1 (h[n_2] *_2 y[n_1, n_2])$$

$$h_{\text{rect}}[n] = \begin{cases} 1 & 0 \le n < U \\ 0 & \text{otherwise} \end{cases}, \quad h_{\text{tri}}[n] = \begin{cases} 1 - \frac{|n|}{U} & -U \le n \le U \\ 0 & \text{otherwise} \end{cases}, \quad h_{\text{sinc}}[n] = \frac{sin(\pi n/U)}{\pi n/U}$$

Optical Flow

$$-\frac{\partial f}{\partial t} = (\nabla f)^T \vec{v}$$

$$\vec{v} = (A^T A)^{-1} A^T \vec{b}$$

Gaussians, GMMS, and Principal Components

$$p_{\vec{X}}(\vec{x}) = \sum_{k=0}^{K-1} c_k \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_k)^T \Sigma_k^{-1}(\vec{x}-\vec{\mu}_k)}$$

$$(n-1)\Sigma = V\Lambda V^T, \quad \frac{1}{n-1}\Lambda = V^T \Sigma V, \quad V^T V = V V^T = I$$

$$\sum_{d=1}^{D} \sigma_d^2 = \frac{1}{n-1}\text{trace}\left(X^T X\right) = \frac{1}{n-1}\text{trace}\left(Y^T Y\right) = \frac{1}{n-1}\sum_{d=1}^{D} \lambda_d$$

LSTM

$$\vec{f}[t] = \sigma(u_f \vec{x}[t] + w_f \vec{h}[t-1] + b_f), \quad \vec{i}[t] = \sigma(u_i \vec{x}[t] + w_i \vec{h}[t-1] + b_i), \quad \vec{o}[t] = \sigma(u_o \vec{x}[t] + w_o \vec{h}[t-1] + b_o)$$

$$\vec{c}[t] = \vec{f}[t]\vec{c}[t-1] + \vec{i}[t] g\left(u_c \vec{x}[t] + w_c \vec{h}[t-1] + b_c\right), \quad \vec{h}[t] = \vec{o}[t]\vec{c}[t]$$

Expectation Maximization and Hidden Markov Models

$$Q(\Theta, \hat{\Theta}) = E\left[\ln p(\mathcal{D}_v, \mathcal{D}_h|\Theta)\,\middle|\,\mathcal{D}_v, \hat{\Theta}\right]$$

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{x}_t), \quad 1 \le j \le N,\ 2 \le t \le T$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\vec{x}_{t+1}) \beta_{t+1}(j), \quad 1 \le i \le N,\ 1 \le t \le T-1$$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^{N} \alpha_t(k)\beta_t(k)}$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(\vec{x}_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{N}\sum_{\ell=1}^{N} \alpha_t(k) a_{k\ell} b_\ell(\vec{x}_{t+1}) \beta_{t+1}(\ell)}$$

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N}\sum_{t=1}^{T-1} \xi_t(i,j)}$$

$$\Sigma'_i = \frac{\sum_{t=1}^{T} \gamma_t(i)(\vec{x}_t - \vec{\mu}_i)(\vec{x}_t - \vec{\mu}_i)^T}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\vec{\mu}'_i = \frac{\sum_{t=1}^{T} \gamma_t(i)\vec{x}_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

Neural Nets

$$\xi_{i,k}^{(\ell)} = w_{k,0}^{(\ell)} + \sum_{j=1}^{p} w_{k,j}^{(\ell)} h_{i,j}^{(\ell-1)}$$

$$h_{i,k}^{(\ell)} = g(\xi_{i,k}^{(\ell)})$$

$$\frac{d\mathcal{L}}{d\xi_{i,k}^{(\ell)}} = \dot{g}(\xi_{i,k}^{(\ell)}) \frac{d\mathcal{L}}{dh_{i,k}^{(\ell)}}$$

$$\frac{d\mathcal{L}}{dh_{i,j}^{(\ell-1)}} = \sum_{k} \frac{d\mathcal{L}}{d\xi_{i,k}^{(\ell)}} w_{k,j}^{(\ell)}$$

$$\frac{d\mathcal{L}}{dw_{k,j}^{(\ell)}} = \sum_{i} \frac{d\mathcal{L}}{d\xi_{i,k}^{(\ell)}} h_{i,j}^{(\ell-1)}$$

$$\dot{\sigma}(x) = \sigma(x)(1 - \sigma(x))$$

$$w_{k,j}^{(\ell)} \leftarrow w_{k,j}^{(\ell)} - \eta \frac{dE}{dw_{k,j}^{(\ell)}}$$

1. (17 points) A particular signal, $x[n]$, has an autocorrelation function whose first two samples are:

$$R_0 = E\left[x^2[n]\right]$$
$$R_1 = E\left[x[n]x[n-1]\right]$$

Suppose we want to model the signal spectrum as

$$|X(\omega)| \approx \frac{G}{|1 - ae^{-j\omega}|}$$

where G, R_0, and R_1 are arbitrary constants. Write a as a function of G, R_0, and/or R_1.

Solution: We can find the LPC coefficients using

$$\vec{\gamma} = R\vec{a}$$

where

$$R = \begin{bmatrix} R_{xx}[0] & \cdots & R_{xx}[p-1] \\ \vdots & \ddots & \vdots \\ R_{xx}[p-1] & \cdots & R_{xx}[0] \end{bmatrix}, \quad \vec{\gamma} = \begin{bmatrix} R_{xx}[1] \\ \vdots \\ R_{xx}[p] \end{bmatrix}$$

In this case, $p = 1$, so we have

$$R_{xx}[1] = R_{xx}[0]a$$

Therefore $a = R_{xx}[1]/R_{xx}[0] = R_1/R_0$.

2. (17 points) In a particular 2×2 block of pixels, the image gradient $\nabla x[n_1, n_2]$ and the temporal rate of change $\frac{\partial x}{\partial t}$ are given by the following table, where $a, \ldots, l$ are arbitrary constants:

(n_1, n_2)	$\frac{\partial x[n_1,n_2,t]}{\partial n_1}$	$\frac{\partial x[n_1,n_2,t]}{\partial n_2}$	$\frac{\partial x[n_1,n_2,t]}{\partial t}$
(0,0)	a	e	i
(0,1)	b	f	j
(1,0)	c	g	k
(1,1)	d	h	l

Suppose that we want to model the video using optical flow, i.e.,

$$x[n_1 + v_1, n_2 + v_2, t] \approx x[n_1, n_2, t] \tag{1}$$

Find v_1 and v_2 so that the approximation in Eq. (1) is satisfied, for all four pixels of the image, with minimum mean-squared error. Your answer can include unresolved matrix multiplications, matrix inversions, determinants and so on, but it should not include any variables other than $a, b, c, d, e, f, g, h, i, j, k, l$.

Solution: The optical flow equation is

$$-\frac{\partial x}{\partial t} = \frac{\partial x}{\partial n_1} v_1 + \frac{\partial x}{\partial n_2} v_2 \tag{2}$$

Approximately satisfying Eq. (1) for all four pixels in the image simultaneously would give

$$\begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \approx - \begin{bmatrix} i \\ j \\ k \\ l \end{bmatrix}$$

The error in Eq. (1) is minimized by the pseudo-inverse, i.e.,

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = - \left(\begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}^T \begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix} \right)^{-1} \begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}^T \begin{bmatrix} i \\ j \\ k \\ l \end{bmatrix}$$

3. (17 points) The gram matrix of a dataset is the matrix whose $(i,j)^{\text{th}}$ element is $\vec{x}_i^T \vec{x}_j$, the inner product of $\vec{x}_i$ and $\vec{x}_j$. A particular dataset has a gram matrix with the following eigenvector/eigenvalue decomposition:

$$G = \begin{bmatrix} -0.19 & -0.22 \\ -0.33 & -0.49 \\ -0.52 & 0.15 \\ -0.34 & 0.77 \\ -0.68 & -0.29 \\ 0.04 & -0.04 \end{bmatrix} \begin{bmatrix} 20 & 0 \\ 0 & 45 \end{bmatrix} \begin{bmatrix} -0.19 & -0.33 & -0.52 & -0.34 & -0.68 & 0.04 \\ -0.22 & -0.49 & 0.15 & 0.77 & -0.29 & -0.04 \end{bmatrix}$$

Suppose that Σ is the sample covariance of the same dataset, and suppose that $\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$. Draw the set of points $\{\vec{x} : \vec{x}^T \Sigma^{-1} \vec{x} = 1\}$. Specify the numerical value of the coordinate of every point where this set intersects the axes.

Solution: The drawing should be an ellipse centered at $(0,0)$, intersecting the axes at $(2,0)$, $(-2,0)$, $(0,3)$, and $(0,-3)$.

4. (17 points) Suppose you are studying the running behaviors of trained vs. untrained athletes. You have a sequence of feature vectors $\vec{x}_t$, where t is time (measured in centiseconds) and $\vec{x}_t$ is a vector of features computed from a motion sensor being worn at the ankle. You have trained a neural network to compute $b_j(\vec{x}_t) = p(\vec{x}_t|q_t = j)$, where $q_t \in \{1 = \text{heel strike}, 2 = \text{roll}, 3 = \text{lift}, 4 = \text{swing}\}$ denotes the gait phase. You also know the following probabilities:

$$a_{i,j} = p(q_t = j|q_{t-1} = i)$$
$$\alpha_t(i) = p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i)$$
$$\beta_t(i) = p(\vec{x}_{t+1}, \ldots, \vec{x}_T|q_t = i)$$

Your goal is to identify all of the instants when the heel first touches the ground, i.e., at each time step τ $(1 \leq \tau \leq T)$, you want to find

$$P_{HS}(\tau) = p(q_{\tau-1} = 4, q_\tau = 1|\vec{x}_1, \ldots, \vec{x}_T)$$

Write a formula for $P_{HS}(\tau)$ in terms of $\alpha_t(i)$, $\beta_t(i)$, $a_{i,j}$, and $b_i(\vec{x}_t)$, for any values of i, j, t that you find useful.

Solution:

$$P_{HS}(\tau) = \frac{\alpha_{\tau-1}(4)a_{4,1}b_1(\vec{x}_\tau)\beta_\tau(1)}{\sum_{i=1}^4 \sum_{j=1}^4 \alpha_{\tau-1}(i)a_{i,j}b_j(\vec{x}_\tau)\beta_\tau(j)}$$

5. (17 points) In a neural network with residual connections (ResNet), the k^{th} activation at layer ℓ, $h_k^{(\ell)}$, is equal to the activation of the same node at the previous layer, plus a computed residual $g(\xi_k^{(\ell)})$:

$$
\begin{aligned}
\xi_k^{(\ell)} &= \sum_{j=1}^{N} w_{k,j}^{(\ell)} h_j^{(\ell-1)}, \quad 1 \le k \le N, \\
h_k^{(\ell)} &= h_k^{(\ell-1)} + g\left(\xi_k^{(\ell)}\right), \quad 1 \le k \le N,
\end{aligned}
$$

where $g(\cdot)$ is a scalar nonlinearity, and $w_{k,j}^{(\ell)}$ is a network weight. Suppose that the training loss is $\mathcal{L}$, and suppose you already know $\frac{d\mathcal{L}}{dh_k^{(\ell)}}$. Find $\frac{d\mathcal{L}}{dh_j^{(\ell-1)}}$ in terms of $\frac{d\mathcal{L}}{dh_k^{(\ell)}}$, $\dot{g}(\xi) = \frac{\partial g}{\partial \xi_k^{(\ell)}}$, and $w_{k,j}^{(\ell)}$.

Solution: The total derivative rule gives us

$$
\begin{aligned}
\frac{d\mathcal{L}}{dh_j^{(\ell-1)}} &= \sum_{k=1}^{N} \frac{d\mathcal{L}}{dh_k^{(\ell)}} \frac{\partial h_k^{(\ell)}}{\partial h_j^{(\ell)}} \\
&= \frac{d\mathcal{L}}{dh_j^{(\ell)}} + \sum_{k=1}^{N} \frac{d\mathcal{L}}{dh_k^{(\ell)}} \dot{g}(\xi_k^{(\ell)}) w_{k,j}^{(\ell)}
\end{aligned}
$$

6. (17 points) An RBF-softmax is similar to a regular softmax nonlinearity, but instead of being a generalization of the logistic sigmoid, it is a generalization of a nonlinearity called a **radial basis function** (RBF), which is a kind of simplified Gaussian. An RBF-softmax has the following form:

$$\hat{y}_k = \frac{w_k e^{-\|\vec{x}-\vec{\mu}_k\|^2}}{\sum_{\ell=1}^{N} w_\ell e^{-\|\vec{x}-\vec{\mu}_\ell\|^2}},$$

where $\vec{x} = [x_1, \ldots, x_D]^T$ is the input vector, $\hat{y}_k$ is the k^{th} output, and w_k and $\vec{\mu}_k = [\mu_{1,k}, \ldots, \mu_{D,k}]^T$, for $1 \le k \le K$, are trainable parameters.

Find $\frac{d\hat{y}_k}{dw_j}$ for all $j \in \{1, \ldots, K\}$. Your answer may contain any of the variables used in the problem statement. Your answer should not include any unresolved derivatives.

Solution:

$$\begin{aligned}
\frac{d\hat{y}_k}{dw_j} &= \frac{e^{-\|\vec{x}-\vec{\mu}_k\|^2}}{\sum_{\ell=1}^{N} w_\ell e^{-\|\vec{x}-\vec{\mu}_\ell\|^2}} 1\left[k=j\right] - \frac{e^{-\|\vec{x}-\vec{\mu}_k\|^2}}{\left(\sum_{\ell=1}^{N} w_\ell e^{-\|\vec{x}-\vec{\mu}_\ell\|^2}\right)^2} e^{-\|\vec{x}-\vec{\mu}_j\|^2} \\
&= \begin{cases} \frac{1}{w_k}\hat{y}_k(1-\hat{y}_k) & k=j \\ -\frac{1}{w_k}\hat{y}_k\hat{y}_j & \text{otherwise} \end{cases}
\end{aligned}$$

7. (17 points) A particular CNN has a grayscale image input, $x[n_1, n_2]$, and a one-channel output:

$$\xi[n_1, n_2] = w[n_1, n_2] * x[n_1, n_2],$$

where $*$ denotes convolution. The output is then max-pooled over the entire image:

$$\hat{y} = \max_{0 \le n_1 < N_1} \max_{0 \le n_2 < N_2} \xi[n_1, n_2]$$

Suppose the weights and the input image are given by

$$w[n_1, n_2] = \begin{cases} e^{-(n_1^2 + n_2^2)} & -3 \le n_1 \le 3, \quad -3 \le n_2 \le 3 \\ 0 & \text{otherwise} \end{cases}$$

$$x[n_1, n_2] = \begin{cases} e^{-((n_1-15)^2 + (n_2-12)^2)} & 0 \le n_1 \le 63, \ 0 \le n_2 \le 63 \\ 0 & \text{otherwise} \end{cases}$$

What is $\frac{d\hat{y}}{dw[2,1]}$? Your answer should be an explicit function of numerical constants; there should not be any variables in your answer.

Solution:

$$\frac{d\hat{y}}{dw[2,1]} = e^{-5}$$

8. (17 points) Sometimes, it's not obvious, in advance, what loss function should be used to train a neural network. For example, suppose that we have a training database containing vector triples of the form $(\vec{x}, \vec{y}, \vec{z})$. Suppose we know that the set of vectors, $\vec{x}$, can be divided in half through the origin such that for half of the vectors, $\vec{y}$ is a linear transformation of $\vec{x}$, while for the other half, $\vec{z}$ is a linear transformation of $\vec{x}$. In other words, for some matrices U_{ideal} and V_{ideal} that we don't know, and for some vector $\vec{w}_{\text{ideal}}$ that we don't know:

   - If $\vec{w}_{\text{ideal}}^T\vec{x} \geq 0$ then $\vec{y} = U_{\text{ideal}}\vec{x}$.
   - If $\vec{w}_{\text{ideal}}^T\vec{x} < 0$ then $\vec{z} = V_{\text{ideal}}\vec{x}$.

   Devise a differentiable non-negative loss function, $\mathcal{L}$, that will approach zero as the estimated values of $\vec{w}, U$, and V approach their true values. Write your loss as a function of the estimated parameters $\vec{w}, U$, and V, and as a function of the vectors in just one data triple, $(\vec{x}, \vec{y}, \vec{z})$.

Solution: First, we want differentiable functions of U and V that will be minimized when $\vec{y} = U\vec{x}$ and $\vec{z} = V\vec{x}$. Most of the functions that do this are norms of the vectors $(\vec{y} - U\vec{x})$ and $(\vec{z} - V\vec{x})$, for example, the squared L2 norms, $\|\vec{y} - U\vec{x}\|^2$ and $\|\vec{z} - V\vec{x}\|^2$, are good choices.

Second, we want to multiply $\|\vec{y} - U\vec{x}\|^2$ by some modifier that goes to zero when $\vec{w}^T\vec{x} < 0$. The unit step function would do the trick, but it's not differentiable; we need something that can be differentiated. The ReLU nonlinearity will do the trick:

$$\mathcal{L} = \text{ReLU}(\vec{w}^T\vec{x})\|\vec{y} - U\vec{x}\|^2 + \text{ReLU}(-\vec{w}^T\vec{x})\|\vec{z} - V\vec{x}\|^2$$

The sigmoid is also a good choice. It doesn't go to zero immediately when $\vec{w}^T\vec{x} < 0$, but it goes to zero when $\vec{w}^T\vec{x} \ll 0$. Since the problem specification doesn't actually dictate the norm of $\vec{w}$ (it can be any scalar times $\vec{w}_{\text{ideal}}$, and still meet the problem specifications), the sigmoid will also work here:

$$\mathcal{L} = \sigma(\vec{w}^T\vec{x})\|\vec{y} - U\vec{x}\|^2 + \sigma(-\vec{w}^T\vec{x})\|\vec{z} - V\vec{x}\|^2$$

9. (17 points) Suppose y is a scalar continuous piece-wise linear function of the scalar variable x, with

$$\frac{dy}{dx} = \begin{cases} 0 & x < x_0 \\ s_i & x_i \leq x < x_{i+1}, \quad 0 \leq i < N \\ s_N & x_N \leq x \end{cases}$$

This function, $y(x)$, can be exactly represented by a ReLU neural network of the form

$$y(x) = \sum_{i=0}^{N} w_i \text{ReLU}(x + b_i)$$

Find w_i and b_i, for all $0 \leq i \leq N$, in terms of s_j and x_j, for any $0 \leq j \leq N$ that you find to be useful.

Solution: We know that

$$\text{ReLU}(x + b_i) = \begin{cases} x + b_i & x + b_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

So we can get the breakpoints exactly right by setting

$$b_i = -x_i$$

Setting the slopes equal, we get that

$$s_i = \sum_{j=0}^{i} w_j$$

which can be inverted to find that

$$w_0 = s_0$$
$$w_i = s_i - s_{i-1}, \quad 1 \leq i \leq N$$

10. (17 points) Suppose we have five variables, u, v, w, x, y. All but seven of their partial derivatives are zero; for example, $\frac{\partial y}{\partial u}(u, v, w, x, y) = \frac{\partial y}{\partial x}(u, v, w, x, y) = 0$. The only seven nonzero partial derivatives are

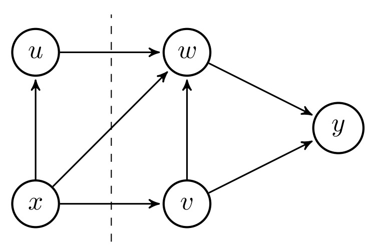
$$\begin{aligned} \frac{\partial u}{\partial x}(u, v, w, x, y) &= a, & \frac{\partial v}{\partial x}(u, v, w, x, y) &= b \\ \frac{\partial w}{\partial u}(u, v, w, x, y) &= c, & \frac{\partial w}{\partial v}(u, v, w, x, y) &= d \\ \frac{\partial w}{\partial x}(u, v, w, x, y) &= e, & \frac{\partial y}{\partial v}(u, v, w, x, y) &= f \\ \frac{\partial y}{\partial w}(u, v, w, x, y) &= g, & & \end{aligned}$$

In terms of the constants a, b, c, d, e, f, and g, find $\nabla_{\begin{bmatrix} x \\ u \end{bmatrix}} y$, the gradient of y with respect to the vector $[x, u]^T$.

Solution: The gradient is defined to be

$$\nabla_{\begin{bmatrix} x \\ u \end{bmatrix}} y = \begin{bmatrix} \frac{\partial y}{\partial x}(u, x) \\ \frac{\partial y}{\partial u}(u, x) \end{bmatrix},$$

i.e., the vector of partial derivatives while keeping constant only the other elements of the same vector. Drawing a flow graph, we find



In this case we can write the total derivative rule as

$$\begin{aligned} \frac{\partial y}{\partial x}(u, x) &= \frac{dy}{dv}\frac{\partial v}{\partial x}(u, v, w, x, y) + \frac{dy}{dw}\frac{\partial w}{\partial x}(u, v, w, x, y) \\ &= \left(\frac{\partial y}{\partial v} + \frac{dy}{dw}\frac{\partial w}{\partial v}\right)\frac{\partial v}{\partial x}(u, v, w, x, y) + \frac{dy}{dw}\frac{\partial w}{\partial x}(u, v, w, x, y) \\ &= (f + gd)b + ge \end{aligned}$$

Similarly,

$$\begin{aligned} \frac{\partial y}{\partial u}(u, x) &= \frac{dy}{dw}\frac{\partial w}{\partial u}(u, v, w, x, y) \\ &= gc \end{aligned}$$

So

$$\nabla_{\begin{bmatrix} x \\ u \end{bmatrix}} y = \begin{bmatrix} fb + gdb + ge \\ gc \end{bmatrix}.$$

11. (30 points) Consider a bidirectional two-layer recurrent network that has been trained to perform the following computations.

- The first layer has forward and backward cells which perform the following computations given an input $x \in \Re$ and prior hidden states $f \in \Re$, $b \in \Re$:

$$\textbf{forward}: f_t = \sin(x_t w_x + f_{t-1} w_h + b)^2,$$
$$\textbf{backward}: b_t = \sin(x_t w_x + b_{t+1} w_h + b)^2,$$

  where the weights are $w_x = \frac{\pi}{4}$, $w_h = \frac{\pi}{2}$, and $b = \frac{\pi}{2}$.

- The second layer has forward and backward cells which perform the following computations given an input $\vec{\xi} \in \Re^2$ and a prior hidden states $y \in \Re$, $z \in \Re$:

$$\textbf{forward}: y_t = \cos\left(\frac{\pi}{2}(\vec{w}_x^T \vec{\xi}_t + w_h y_{t-1} + b)\right),$$
$$\textbf{backward}: z_t = \cos\left(\frac{\pi}{2}(\vec{w}_x^T \vec{\xi}_t + w_h z_{t+1} + b)\right),$$

  where the weights are $w_x = [2, 1]^T$, $w_h = 2$, and $b = 1$. Assume that the prior hidden state, before each cell reads its first input, is 0.

(a) Consider the input sequence $[x_1, x_2, x_3] = [4, 1, 7]$. What are the forward outputs $[f_1, f_2, f_3]$ and the backward outputs $[b_3, b_2, b_1]$ from the first layer?

Solution:

$$[f_1, f_2, f_3] = [1, \frac{1}{2}, 1]$$
$$[b_3, b_2, b_1] = [\frac{1}{2}, 0, 1]$$

(b) Now consider the outputs $[f_1, f_2, f_3] = [3, 1, 3]$ from the forward cell in the first layer and the outputs $[b_3, b_2, b_1] = [3, 1, 0]$ from the backward cell in the first layer. Let $\vec{\xi_t} = [f_t, b_t]^T$. What are the forward outputs $[y_1, y_2, y_3]$ and the backward outputs $[z_3, z_2, z_1]$ from the second layer?

Solution:

$$[y_1, y_2, y_3] = [0, 1, 1]$$
$$[z_3, z_2, z_1] = [-1, -1, 0]$$