UNIVERSITY OF CALIFORNIA, BERKELEY
DEPARTMENT OF STATISTICS

Homework #4

Student name: *Shizhe Zhang*

Course: *EECS 227AT* – Professor: *Gireeja Ranade*
Date: *February 19, 2026*

Problem 1 Gradients, Jacobians, and Hessians

The gradient of a scalar-valued function $g : \mathbb{R}^n \to \mathbb{R}$ is the column vector of length n, denoted as ∇g, containing the derivatives of components of g with respect to the input variables:

$$(\nabla g(\vec{x}))_i = \frac{\partial g}{\partial x_i}(\vec{x}), \quad i = 1, \ldots, n. \tag{1}$$

The Hessian of a scalar-valued function $g : \mathbb{R}^n \to \mathbb{R}$ is the $n \times n$ matrix, denoted as $\nabla^2 g$, containing the second derivatives of components of g with respect to the input variables:

$$(\nabla^2 g(\vec{x}))_{ij} = \frac{\partial^2 g}{\partial x_i \partial x_j}(\vec{x}), \quad i = 1, \ldots, n,\ j = 1, \ldots, n. \tag{2}$$

The Jacobian of a vector-valued function $\vec{g} : \mathbb{R}^n \to \mathbb{R}^m$ is the $m \times n$ matrix, denoted as $D\vec{g}$, containing the derivatives of components of $\vec{g}$ with respect to the input variables:

$$(D\vec{g}(\vec{x}))_{ij} = \frac{\partial g_i}{\partial x_j}(\vec{x}), \quad i = 1, \ldots, m,\ j = 1, \ldots, n. \tag{3}$$

For the remainder of the class, we will repeatedly have to take gradients, Hessians and Jacobians of functions we are trying to optimize. This exercise serves as a warm up for future problems.

For the first two parts, suppose $A \in \mathbb{R}^{n \times n}$ is a square matrix whose entries are denoted a_{ij} and whose rows are denoted $\vec{a}_1^\top, \ldots, \vec{a}_n^\top$, and $\vec{b} \in \mathbb{R}^n$ is a vector whose entries are denoted b_i.

(a) Compute the Jacobians for the following functions.

i. $\vec{g}(\vec{x}) = A\vec{x}$.

ii. $\vec{g}(\vec{x}) = f(\vec{x})\vec{x}$ where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable.

iii. $\vec{g}(\vec{x}) = f(A\vec{x} + \vec{b})\vec{x}$ where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable.

(b) Compute the gradients and Hessians for the following functions.

i. $g_1(\vec{x}) = \vec{x}^\top A\vec{x}$.

ii. $g_2(\vec{x}) = \|\vec{x}\|_2^2$.

iii. $g_3(\vec{x}) = g_2(A\vec{x} - \vec{b}) = \|A\vec{x} - \vec{b}\|_2^2$. (Use the chain rule and the Jacobians computed in part 1(a).)

iv. $g_4(\vec{x}) = \log(\sum_{i=1}^{n} e^{x_i})$.

v. $g_5(\vec{x}) = g_4(A\vec{x} - \vec{b}) = \log\left(\sum_{i=1}^{n} e^{\vec{a}_i^\top \vec{x} - b_i}\right)$. (Use the chain rule and the Jacobians computed in part 1(a); you can use the gradient ∇g_4 and Hessian $\nabla^2 g_4$ in your answer without having to rewrite it.)

vi. $g_6(\vec{x}) = e^{\|\vec{x}\|_2^2} = e^{g_2(\vec{x})}$. (Use the chain rule and the Jacobians computed in part 1(a).)

vii. $g_7(\vec{x}) = e^{\|A\vec{x} - \vec{b}\|_2^2} = g_6(A\vec{x} - \vec{b})$. (Use the chain rule and the Jacobians computed in part 1(a); you can use the gradient ∇g_6 and Hessian $\nabla^2 g_6$ in your answer without having to rewrite it.)

Consider the case now where all vectors and matrices above are scalar; do your answers above make sense? (No need to answer this in your submission.)

(c) Plot/hand-draw the level sets of the following functions:

i. $g(x_1, x_2) = \frac{x_1^2}{4} + \frac{x_2^2}{9}$

ii. $g(x_1, x_2) = x_1 x_2$

Also point out the gradient directions in the level-set diagram. Additionally, compute the first and second order Taylor series approximation around the point $(1, 1)$ for each function and comment on how accurately they approximate the true function.

Answer.

(a)

(i) Consider $A = \begin{bmatrix} a_1 \\ \vdots \\ a_i \\ \vdots \\ a_n \end{bmatrix}$ where a_i is the i-th row of A. Then

$$g_i(\vec{x}) = a_i^\top \vec{x}$$

$$\frac{\partial g_i}{\partial x_j}(\vec{x}) = a_{ij}$$

Thus,

$$D\vec{g}(\vec{x}) = A.$$

(ii)

$$g_i(\vec{x}) = f(\vec{x})\, x_i,$$

For $i = j$,

$$\frac{\partial g_i}{\partial x_j}(\vec{x}) = \frac{\partial f}{\partial x_j}(\vec{x})\, x_i + f(\vec{x}),$$

And for $i \neq j$,

$$\frac{\partial g_i}{\partial x_j}(\vec{x}) = \frac{\partial f}{\partial x_j}(\vec{x})\, x_i$$

Thus,

$$D\vec{g}(\vec{x}) = f(\vec{x})I + \vec{x}\,\nabla f(\vec{x})^\top.$$

(iii) Let $h(\vec{x}) := f(A\vec{x} + \vec{b})$. Then $\vec{g}(\vec{x}) = h(\vec{x})\vec{x}$.
From problem(ii), we have,

$$D\vec{g}(\vec{x}) = h(\vec{x})I + \vec{x}\,\nabla h(\vec{x})^\top,$$

By the chain rule,

$$\nabla h(\vec{x}) = A^\top \nabla f(A\vec{x} + \vec{b}),$$

so

$$D\vec{g}(\vec{x}) = f(A\vec{x} + \vec{b})I + \vec{x}\left(A^\top \nabla f(A\vec{x} + \vec{b})\right)^\top.$$

(b) Fact,

$$f(\vec{x}) = \beta^\top \vec{x} \implies \nabla f(\vec{x}) = \beta$$

(i) $g_1(\vec{x}) = \vec{x}^\top A\vec{x}$:

$$\nabla g_1(\vec{x}) = (A + A^\top)\vec{x}, \qquad \nabla^2 g_1(\vec{x}) = A + A^\top.$$

(ii) $g_2(\vec{x}) = \|\vec{x}\|_2^2 = \vec{x}^\top \vec{x}$:

$$\nabla g_2(\vec{x}) = 2\vec{x}, \qquad \nabla^2 g_2(\vec{x}) = 2I.$$

(iii) $g_3(\vec{x}) = \|A\vec{x} - \vec{b}\|_2^2$:

$$\nabla g_3(\vec{x}) = 2A^\top(A\vec{x} - \vec{b}), \qquad \nabla^2 g_3(\vec{x}) = 2A^\top A.$$

(iv) $g_4(\vec{x}) = \log\Big(\sum_{i=1}^n e^{x_i}\Big)$.

Let $s(\vec{x}) = \sum_{i=1}^n e^{x_i}$ and $p_i(\vec{x}) = \frac{e^{x_i}}{s(\vec{x})}$.

Then

$$\nabla g_4(\vec{x}) = \vec{p}(\vec{x}),$$

For the Hessian, if $i = j$,

$$(\nabla^2 g_4(\vec{x}))_{ij} = p_i(\vec{x})(1 - p_i(\vec{x})),$$

and if $i \neq j$,

$$(\nabla^2 g_4(\vec{x}))_{ij} = -p_i(\vec{x})p_j(\vec{x}).$$

Thus,

$$\nabla^2 g_4(\vec{x}) = \mathrm{diag}(\vec{p}(\vec{x})) - \vec{p}(\vec{x})\vec{p}(\vec{x})^\top.$$

(v) $g_5(\vec{x}) = g_4(A\vec{x} - \vec{b})$:

$$\nabla g_5(\vec{x}) = A^\top \nabla g_4(A\vec{x} - \vec{b}) = A^\top \vec{p}(A\vec{x} - \vec{b}),$$

$$\nabla^2 g_5(\vec{x}) = A^\top \nabla^2 g_4(A\vec{x} - \vec{b})\, A.$$

(vi) $g_6(\vec{x}) = e^{\|\vec{x}\|_2^2}$. Let $f(\vec{x}) = \|\vec{x}\|_2^2 = \vec{x}^\top \vec{x}$. Thus, $\nabla f = 2\vec{x}$ and $\nabla^2 f = 2I$,

$$\nabla g_6(\vec{x}) = e^{f(\vec{x})} \nabla f(\vec{x}) = 2e^{\|\vec{x}\|_2^2} \vec{x},$$

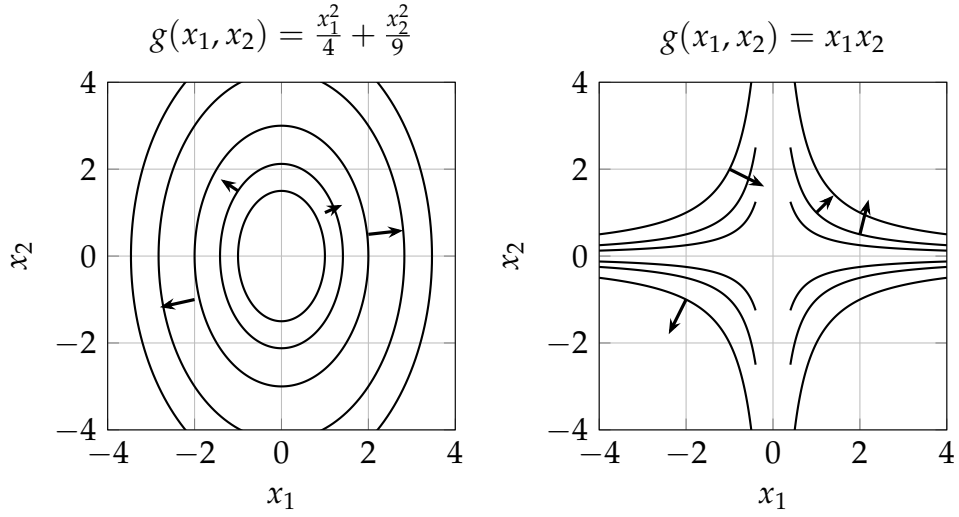
$$\nabla^2 g_6(\vec{x}) = e^{f(\vec{x})} \big(\nabla f(\vec{x}) \nabla f(\vec{x})^\top + \nabla^2 f(\vec{x})\big) = e^{\|\vec{x}\|_2^2} \big(4\vec{x}\vec{x}^\top + 2I\big).$$

(vii) $g_7(\vec{x}) = g_6(A\vec{x} - \vec{b})$.

$$\nabla g_7(\vec{x}) = A^\top \nabla g_6(A\vec{x} - \vec{b})$$

Applying the result in 1(a),

$$\nabla^2 g_7(\vec{x}) = A^\top \nabla^2 g_6(A\vec{x} - \vec{b})\, A$$



(c)

(i) $g(x_1, x_2) = \frac{x_1^2}{4} + \frac{x_2^2}{9}$. Level sets $\{(x_1, x_2) : \frac{x_1^2}{4} + \frac{x_2^2}{9} = c\}$ are ellipses centered at $(0, 0)$.

$$\nabla g(x_1, x_2) = \begin{bmatrix} \frac{x_1}{2} \\ \frac{2x_2}{9} \end{bmatrix}, \qquad \nabla^2 g(x_1, x_2) = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{2}{9} \end{bmatrix}.$$

At $(1, 1)$:

$$g(1, 1) = \frac{1}{4} + \frac{1}{9} = \frac{13}{36}, \qquad \nabla g(1, 1) = \begin{bmatrix} \frac{1}{2} \\ \frac{2}{9} \end{bmatrix}.$$

First-order Taylor at $(1, 1)$:

$$T_1(x_1, x_2) = \frac{13}{36} + \frac{1}{2}(x_1 - 1) + \frac{2}{9}(x_2 - 1) = -\frac{13}{36} + \frac{1}{2}x_1 + \frac{2}{9}x_2.$$

Second-order Taylor at $(1,1)$:

$$T_2(x_1,x_2) = T_1(x_1,x_2) + \frac{1}{2}\begin{bmatrix} x_1-1 & x_2-1 \end{bmatrix}\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{2}{9} \end{bmatrix}\begin{bmatrix} x_1-1 \\ x_2-1 \end{bmatrix}.$$

Since g is quadratic, $T_2 = g$; T_1 is a local linear approximation.

(ii) $g(x_1,x_2) = x_1x_2$. Level sets $\{(x_1,x_2) : x_1x_2 = c\}$ are rectangular hyperbolas.

$$\nabla g(x_1,x_2) = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}, \qquad \nabla^2 g(x_1,x_2) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

At $(1,1)$:

$$g(1,1) = 1, \qquad \nabla g(1,1) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

First-order Taylor at $(1,1)$:

$$T_1(x_1,x_2) = 1 + (x_1-1) + (x_2-1) = x_1 + x_2 - 1.$$

Second-order Taylor at $(1,1)$:

$$\begin{aligned} T_2(x_1,x_2) &= T_1(x_1,x_2) + \frac{1}{2}\cdot 2(x_1-1)(x_2-1) \\ &= 1 + (x_1-1) + (x_2-1) + (x_1-1)(x_2-1) \\ &= x_1x_2. \end{aligned}$$

Again, T_2 is exact because g is a degree-2 polynomial; T_1 is only locally accurate.

Problem 2 Neural Networks and Backpropagation

Neural networks are parametric functions that have been widely used to fit complex patterns in vision and natural languages. Given some training data of the form $(\vec{x}_i, y_i)$, a neural network N is trained to minimize a loss function on the data. This is often done using gradient descent, an optimization method we will cover later in this class. Gradient descent requires us to compute the gradients of the loss function with respect to the parameters of the neural network. In practice, computational frameworks for neural networks compute the gradients automatically and efficiently via back-propagation, which uses the chain rule to recursively compute the gradients of the loss function. In this problem, we study a toy neural network trained on a single data point $(\vec{x}, y)$.

In particular, consider the following simplified three-layer neural network N, representing a map from $\mathbb{R}^d$ to $\mathbb{R}$ whose parameters are $(\vec{w}_1, w_2, w_3) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}$:

$$p_1 = \vec{w}_1^\top \vec{x} \tag{4}$$
$$h_1 = \sigma(p_1) \tag{5}$$
$$p_2 = w_2 h_1 \tag{6}$$
$$h_2 = \sigma(p_2) \tag{7}$$
$$z = w_3 h_2 \tag{8}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear function (also called the activation function), whose derivative is denoted by $\sigma' : \mathbb{R} \to \mathbb{R}$.

We want the output of the network $z = N(\vec{x})$ to match the true label y. A natural choice of the loss function encouraging this behavior is the squared loss:

$$L(y, z) \stackrel{\text{def}}{=} \frac{1}{2}(y - z)^2. \tag{9}$$

In the parts that follow, we will compute the derivative of L with respect to the parameters $\vec{w}_1, w_2, w_3$.

(a) Compute the following gradients and partial derivatives sequentially from left-to-right:

$$\frac{\partial L}{\partial z}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial h_2}, \frac{\partial L}{\partial p_2}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial h_1}, \frac{\partial L}{\partial p_1}, \nabla_{\vec{w}_1} L, \nabla_{\vec{x}} L. \tag{10}$$

Here $\nabla_{\vec{x}} L$ is the gradient whose entries are the derivatives of L with respect to the entries of $\vec{x}$, etc. We compute the first 4 derivatives for you.

$$\frac{\partial L}{\partial z} = z - y, \frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial z} h_2, \frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial z} w_3, \frac{\partial L}{\partial p_2} = \frac{\partial L}{\partial h_2} \sigma'(p_2). \tag{11}$$

Note how $\frac{\partial L}{\partial w_3}$ can be calculated using $\frac{\partial L}{\partial z}$ and $\frac{\partial L}{\partial p_2}$ can be calculated using $\frac{\partial L}{\partial h_2}$. In numerical computation, the result of $\frac{\partial L}{\partial z}$ and $\frac{\partial L}{\partial h_2}$ can thus be reused. This technique of saving computations for calculating the derivatives of a neural network is called back-propagation.

Use the chain rule to calculate the 5 remaining derivatives $\frac{\partial L}{\partial w_2}$, $\frac{\partial L}{\partial h_1}$, $\frac{\partial L}{\partial p_1}$, $\nabla_{\vec{w}_1} L$, and $\nabla_{\vec{x}} L$.

(b) Now suppose that Equation (8) is written as

$$z' = w_3 h_2 + h_1. \tag{12}$$

That is, we define a new neural network N' with parameters $(\vec{w}_1, w_2, w_3) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}$ as follows:

$$p_1 = \vec{w}_1^\top \vec{x} \tag{13}$$
$$h_1 = \sigma(p_1) \tag{14}$$
$$p_2 = w_2 h_1 \tag{15}$$
$$h_2 = \sigma(p_2) \tag{16}$$
$$z' = w_3 h_2 + h_1. \tag{17}$$

This introduces a change in the network architecture, called the skip connection.

Again, compute the following gradients and partial derivatives with respect to the loss function

$$L(y, z') = \frac{1}{2}(y - z')^2 :$$

$$\frac{\partial L}{\partial z'}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial h_2}, \frac{\partial L}{\partial p_2}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial h_1}, \frac{\partial L}{\partial p_1}, \nabla_{\vec{w}_1} L, \nabla_{\vec{x}} L. \tag{18}$$

We compute the first 4 derivatives for you.

$$\frac{\partial L}{\partial z'} = z' - y, \tag{1}$$
$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial z'} h_2, \tag{2}$$
$$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial z'} w_3, \tag{3}$$
$$\frac{\partial L}{\partial p_2} = \frac{\partial L}{\partial h_2} \sigma'(p_2). \tag{19}$$

Use the chain rule to calculate the 5 remaining derivatives $\frac{\partial L}{\partial w_2}$, $\frac{\partial L}{\partial h_1}$, $\frac{\partial L}{\partial p_1}$, $\nabla_{\vec{w}_1} L$, and $\nabla_{\vec{x}} L$.

(c) In optimizing a neural network using gradient descent, we need the gradient of the loss function with respect to the parameters of the network. Please express $\frac{\partial L}{\partial w_3}$, $\frac{\partial L}{\partial w_2}$, and $\nabla_{\vec{w}_1} L$ for N and N' respectively with no dependence on partial derivatives of other variables. We compute $\frac{\partial L}{\partial w_3}$ for you, as follows.

- For N, we have $\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial z} h_2 = (z - y) h_2$.
- For N', we have $\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial z'} h_2 = (z' - y) h_2$.

Express the remaining derivatives $\frac{\partial L}{\partial w_2}$ and $\nabla_{\vec{w}_1} L$ within N and N'.

(d) Many activation functions σ have the property that $\sigma' \leq 1$. For example, the sigmoid function

$$\sigma(p) = \frac{1}{1+e^{-p}}$$

is sometimes used as an activation function. Its derivative,

$$\sigma'(p) = \frac{e^{-p}}{(1+e^{-p})^2}$$

has the range $(0, 1/4]$. That is, $\sigma' < 1$.

Consider the case when σ' is much smaller than 1, such that $\sigma'(p)\sigma'(q) \approx 0$ for any p, q, but $\sigma'(p) \not\approx 0$ for any p. Consider the derivatives $\frac{\partial L}{\partial w_3}$, $\frac{\partial L}{\partial w_2}$, and $\nabla_{\vec{w}_1} L$ within the neural network N; with the above approximations, which of them will approximately be zero? Also answer this question for the neural network N'.

NOTE: Some of the above gradients will indeed be approximately zero, and this is called the vanishing gradient problem in deep learning.

Answer.

(a)

$$\begin{aligned} p_1 &= \vec{w}_1^\top \vec{x}, \\ h_1 &= \sigma(p_1), \\ p_2 &= w_2 h_1, \\ h_2 &= \sigma(p_2), \\ z &= w_3 h_2, \end{aligned}$$

$$L(y, z) = \frac{1}{2}(y-z)^2.$$

$$\begin{aligned} \frac{\partial L}{\partial z} &= z - y, \\ \frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_3} = (z-y)\, h_2, \\ \frac{\partial L}{\partial h_2} &= \frac{\partial L}{\partial z}\frac{\partial z}{\partial h_2} = (z-y)\, w_3, \\ \frac{\partial L}{\partial p_2} &= \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial p_2} = (z-y)\, w_3\, \sigma'(p_2). \end{aligned}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial p_2}\frac{\partial p_2}{\partial w_2} = \frac{\partial L}{\partial p_2}\, h_1 = (z-y)\, w_3\, \sigma'(p_2)\, h_1.$$

$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial p_2}\frac{\partial p_2}{\partial h_1} = \frac{\partial L}{\partial p_2}\, w_2 = (z-y)\, w_3\, \sigma'(p_2)\, w_2.$$

$$\frac{\partial L}{\partial p_1} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial p_1} = \frac{\partial L}{\partial h_1}\,\sigma'(p_1) = (z-y)\,w_3\,\sigma'(p_2)\,w_2\,\sigma'(p_1).$$

$\nabla_{\vec{w}_1} L$. Since $p_1 = \vec{w}_1^{\top}\vec{x}$, we have $\nabla_{\vec{w}_1} p_1 = \vec{x}$, hence

$$\nabla_{\vec{w}_1} L = \frac{\partial L}{\partial p_1}\,\nabla_{\vec{w}_1} p_1 = \frac{\partial L}{\partial p_1}\,\vec{x} = (z-y)\,w_3\,\sigma'(p_2)\,w_2\,\sigma'(p_1)\,\vec{x}.$$

$\nabla_{\vec{x}} L$. Since $\nabla_{\vec{x}} p_1 = \vec{w}_1$,

$$\nabla_{\vec{x}} L = \frac{\partial L}{\partial p_1}\,\nabla_{\vec{x}} p_1 = \frac{\partial L}{\partial p_1}\,\vec{w}_1 = (z-y)\,w_3\,\sigma'(p_2)\,w_2\,\sigma'(p_1)\,\vec{w}_1.$$

(b)

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial p_2}\frac{\partial p_2}{\partial w_2} = \frac{\partial L}{\partial p_2}\,h_1 = (z'-y)\,w_3\,\sigma'(p_2)\,h_1.$$

$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial z'}\frac{\partial z'}{\partial h_1} + \frac{\partial L}{\partial p_2}\frac{\partial p_2}{\partial h_1}.$$

Since $\frac{\partial z'}{\partial h_1} = 1$ and $\frac{\partial p_2}{\partial h_1} = w_2$, we get

$$\frac{\partial L}{\partial h_1} = (z'-y)\cdot 1 + \big((z'-y)\,w_3\,\sigma'(p_2)\big)\,w_2 = (z'-y)\Big(1 + w_2 w_3 \sigma'(p_2)\Big).$$

$$\frac{\partial L}{\partial p_1} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial p_1} = \frac{\partial L}{\partial h_1}\,\sigma'(p_1) = (z'-y)\Big(1 + w_2 w_3 \sigma'(p_2)\Big)\sigma'(p_1).$$

$$\nabla_{\vec{w}_1} L = \frac{\partial L}{\partial p_1}\,\nabla_{\vec{w}_1} p_1 = \frac{\partial L}{\partial p_1}\,\vec{x} = (z'-y)\Big(1 + w_2 w_3 \sigma'(p_2)\Big)\sigma'(p_1)\,\vec{x}.$$

$$\nabla_{\vec{x}} L = \frac{\partial L}{\partial p_1}\,\nabla_{\vec{x}} p_1 = \frac{\partial L}{\partial p_1}\,\vec{w}_1 = (z'-y)\Big(1 + w_2 w_3 \sigma'(p_2)\Big)\sigma'(p_1)\,\vec{w}_1.$$

(c)

Network N.

$$\frac{\partial L}{\partial w_2} = (z-y)\,w_3\,\sigma'(p_2)\,h_1, \qquad \nabla_{\vec{w}_1} L = (z-y)\,w_3\,\sigma'(p_2)\,w_2\,\sigma'(p_1)\,\vec{x}.$$

Network N'.

$$\frac{\partial L}{\partial w_2} = (z'-y)\,w_3\,\sigma'(p_2)\,h_1, \qquad \nabla_{\vec{w}_1} L = (z'-y)\Big(1 + w_2 w_3 \sigma'(p_2)\Big)\sigma'(p_1)\,\vec{x}.$$

(d)

Network N. From part (c),

$$\frac{\partial L}{\partial w_3} = (z-y)h_2, \qquad \frac{\partial L}{\partial w_2} = (z-y)\, w_3\, \sigma'(p_2)\, h_1, \qquad \nabla_{\vec{w}_1} L = (z-y)\, w_3\, \sigma'(p_2)\, w_2\, \sigma'(p_1)\, \vec{x}.$$

Under the approximation $\sigma'(p_2)\sigma'(p_1) \approx 0$, we have

$$\nabla_{\vec{w}_1} L \approx \vec{0}.$$

$$\frac{\partial L}{\partial w_2} \not\approx 0, \qquad \frac{\partial L}{\partial w_3} \not\approx 0$$

Network N'. From part (c),

$$\frac{\partial L}{\partial w_3} = (z'-y)h_2, \qquad \frac{\partial L}{\partial w_2} = (z'-y)\, w_3\, \sigma'(p_2)\, h_1,$$

$$\nabla_{\vec{w}_1} L = (z'-y)\Big(1 + w_2 w_3 \sigma'(p_2)\Big)\sigma'(p_1)\, \vec{x}.$$

$$\nabla_{\vec{w}_1} L = (z'-y)\sigma'(p_1)\vec{x} \;+\; (z'-y)\, w_2 w_3\, \sigma'(p_2)\sigma'(p_1)\, \vec{x}.$$

$$\nabla_{\vec{w}_1} L \approx (z'-y)\sigma'(p_1)\vec{x},$$

which is *not* approximately zero in general (since $\sigma'(p_1) \not\approx 0$). As in N, $\frac{\partial L}{\partial w_2}$ has only one σ' factor and $\frac{\partial L}{\partial w_3}$ has none, so none of them will be approximately zero in general.

Problem 3 PCA and Senate Voting Data

In this problem, we consider a matrix of senate voting data, which we manipulate in Python. The data is contained in a $n \times d$ data matrix X, where each row corresponds to a senator and each column to a bill. Each entry of X is either 1, -1 or 0, depending on whether the senator voted for the bill, against the bill, or abstained, respectively.

(a) Suppose we want to assign a score to each senator based on their voting pattern, and then observe the empirical variance of these scores. To describe this, let us choose a $\vec{a} \in \mathbb{R}^d$ and a scalar $b \in \mathbb{R}$. We define the score for senator i as:

$$f(\vec{x}_i, \vec{a}, b) = \vec{x}_i^\top \vec{a} + b, \quad i = 1, 2, \ldots, n. \tag{20}$$

Note that $\vec{x}_i^\top$ denotes the ith row of X and is a row vector of length d, as in the previous problem.

Let us denote by $\vec{z} = f(X, \vec{a}, b)$ the column vector of length n obtained by stacking the scores for each senator. Then

$$\vec{z} = f(X, \vec{a}, b) = X\vec{a} + b\vec{1} \in \mathbb{R}^n \tag{21}$$

where $\vec{1}$ is a vector with all entries equal to 1. Let us denote the mean value of $\vec{z}$ by $\mu(\vec{z}) = \frac{1}{n}\vec{1}^\top \vec{z}$. Let $\vec{\mu}(X) \in \mathbb{R}^d$ denote the vector containing the mean of each column of X. Then

$$\begin{aligned}
\mu(\vec{z}) &= \frac{1}{n}\sum_{i=1}^{n} z_i && (22)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\vec{a}^\top \vec{x}_i + b) && (23)\\
&= \vec{a}^\top \left(\frac{1}{n}\sum_{i=1}^{n} \vec{x}_i\right) + b && (24)\\
&= \vec{a}^\top \vec{\mu}(X) + b. && (25)
\end{aligned}$$

The empirical variance of the scores can then be obtained as

$$\sigma^2(\vec{z}) = \frac{1}{n}(\vec{z} - \mu(\vec{z})\vec{1})^\top(\vec{z} - \mu(\vec{z})\vec{1}) \tag{26}$$

$$= \frac{1}{n}((X\vec{a} + b\vec{1}) - (\vec{a}^\top\vec{\mu}(X) + b)\vec{1})^\top((X\vec{a} + b\vec{1}) - (\vec{a}^\top\vec{\mu}(X) + b)\vec{1}) \tag{27}$$

$$= \frac{1}{n}(X\vec{a} + b\vec{1} - (\vec{a}^\top\vec{\mu}(X))\vec{1} - b\vec{1})^\top(X\vec{a} + b\vec{1} - (\vec{a}^\top\vec{\mu}(X))\vec{1} - b\vec{1}) \tag{28}$$

$$= \frac{1}{n}(X\vec{a} - (\vec{a}^\top\vec{\mu}(X))\vec{1})^\top(X\vec{a} - (\vec{a}^\top\vec{\mu}(X))\vec{1}) \tag{29}$$

$$= \frac{1}{n}(X\vec{a} - \vec{1}\vec{\mu}(X)^\top\vec{a})^\top(X\vec{a} - \vec{1}\vec{\mu}(X)^\top\vec{a}) \tag{30}$$

$$= \frac{1}{n}((X - \vec{1}\vec{\mu}(X)^\top)\vec{a})^\top((X - \vec{1}\vec{\mu}(X)^\top)\vec{a}) \tag{31}$$

$$= \frac{1}{n}\vec{a}^\top(X - \vec{1}\vec{\mu}(X)^\top)^\top(X - \vec{1}\vec{\mu}(X)^\top)\vec{a}. \tag{32}$$

Note that this variance is therefore a function of the centered data matrix $X - \vec{1}\vec{\mu}(X)^\top$ in which the mean of each column is zero. It also does not depend on b.

For the remainder of this problem, we assume that the data has been pre-centered (i.e., $\vec{\mu}(X) = \vec{0}$); note that this has been pre-computed for you in the code notebook. Assume also that $b = 0$, so that $\mu(\vec{z}) = 0$. Defining $f(X, \vec{a}) \stackrel{\text{def}}{=} f(X, \vec{a}, 0)$ and replacing $\vec{z}$ with $f(X, \vec{a})$, we can then write the simpler empirical variance formula

$$\sigma^2(f(X, \vec{a})) = \frac{1}{n}\vec{a}^\top X^\top X\vec{a}. \tag{33}$$

Suppose we restrict $\vec{a}$ to have unit-norm. In the provided code, find the maximum empirical variance $\sigma^2(f(X, \vec{a}))$ over all unit-norm $\vec{a}$, and find the $\vec{a}$ that maximizes it.

(b) We next consider party affiliation as a predictor for how a senator will vote. Follow the instructions in the notebook to compute the mean voting vector for each party and relate it to the direction of maximum variance.

(c) Recall from problem 1 that given a vector $\vec{z} = X\vec{u}$ (i.e., the vector of scalar projections of each row of X along $\vec{u}$), we can compute its empirical variance as

$$\sigma^2(\vec{z}) = \vec{u}^\top\Sigma\vec{u}, \tag{34}$$

where $\Sigma(X) = \frac{1}{n}X^\top X$ is the empirical covariance matrix of X. We will show in a future homework problem that the variance along each principal component $\vec{a}_i$ is precisely its corresponding eigenvalue of $\Sigma(X)$, i.e., $\lambda_i\{\Sigma(X)\}$. (For now, just note that this fact should make intuitive sense, since PCA is searching for directions of maximum variance of the data, and these occur along the covariance matrixs eigenvectors.)

In the Notebook, compute the sum of the variance along $\vec{a}_1$ and $\vec{a}_2$ and plot the data projected on the $\vec{a}_1\vec{a}_2$ plane.

(d) Suppose we want to find the bills that are most and least contentious i.e., those that have high variability in senators votes, and those for which voting was almost unanimous. Follow the instructions in the notebook to compute a measure of contentiousness for each bill, plot the vote counts for exemplar bills, and comment on the voting trends.

(e) Suppose we want to infer the political affiliations of two senators whose voting records are known to us. Follow the instructions in the notebook to infer the political affiliation of the Green and Grey colored senators using PCA.

(f) Finally, we can use the defined score $f(X, \vec{a}, b)$, computed along the first principal component $\vec{a}_1$ to classify the most and least extreme senators based on their voting record. Follow the instructions in the Notebook to compute these scores and comment on their relationship to partisan affiliation.

Answer.

(a)

$$\sigma^2(\vec{z}) = \frac{1}{n}\vec{a}^\top X^\top X\vec{a} = \vec{a}^\top \Sigma(X)\vec{a}, \qquad \Sigma(X) \stackrel{\text{def}}{=} \frac{1}{n}X^\top X.$$

Maximizing variance over $\|\vec{a}\|_2 = 1$ is the Rayleigh quotient problem

$$\max_{\|\vec{a}\|_2=1} \vec{a}^\top \Sigma(X)\vec{a}.$$

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ be the eigenvalues of $\Sigma(X)$ with orthonormal eigenvectors $\{\vec{a}_1, \ldots, \vec{a}_d\}$.

$$\max_{\|\vec{a}\|_2=1} \vec{a}^\top \Sigma(X)\vec{a} = \lambda_1, \quad \text{achieved by } \vec{a} = \pm\vec{a}_1.$$

(b)-(f) c.f. notebook.

Problem 4 SVD Transformation

In this problem we will interpret the linear map defined by matrix $A \in \mathbb{R}^{n\times n}$ by looking at its singular value decomposition, $A = UDV^\top$. Recall that here $U, D, V \in \mathbb{R}^{n\times n}$ and U, V are orthonormal matrices while D is a diagonal matrix. We will first look at how $V^\top$, D and U each separately transform the unit circle

$$C = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$$

and then look at their effect as a whole.

(a) Let $\vec{x} \in \mathbb{R}^n$ and $\vec{z} = V^\top \vec{x}$. Show that

$$\vec{x} = \sum_{i=1}^{n} z_i \vec{v}_i,$$

where $\vec{z} = [z_1 \ \cdots \ z_n]^\top$ and $V = [\vec{v}_1 \ \cdots \ \vec{v}_n]$. This shows that $\vec{z}$ is the vector of coordinates that represents $\vec{x}$ in the basis defined by the columns of V.

For the rest of the problem we restrict ourselves to the case where $A \in \mathbb{R}^{2\times 2}$ and move to the notebook.

Answer. Since V is an orthonormal matrix, we have $V^\top V = I$. Therefore,

$$\vec{z} = V^\top \vec{x} \implies V\vec{z} = VV^\top \vec{x} = \vec{x}.$$

where $V\vec{z} = z_1\vec{v}_1 + \cdots + z_n\vec{v}_n$ is the linear combination of the columns of V with coefficients given by the entries of $\vec{z}$.

Problem 5 Homework Process

With whom did you work on this homework? List the names and SIDs of your group members.
NOTE: If you didnt work with anyone, you can put "none" as your answer.

Answer. none