New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6923: Written Homework 2.
Due Monday, Oct. 27th, 2025, 11:59pm.

Discussion with other students is allowed for this problem set, but solutions must be written-up individually.

10% extra credit will be given if solutions are typewritten (using LaTeX, Markdown, or another mathematical formatting program). MSWord with Equation Editor does not count.

Problem 1: Finer Regularization (15 points)

Suppose we have a linear regression using polynomial features $[1, x, x^2, \ldots, x^{d-1}]$. We have seen how high-degree polynomials can overfit. We will now introduce regularization to prevent this overfitting. Throughout this problem, we will use regularization based on the squared values of the parameters, generalizing standard ℓ_2 regularization.

(a) We may want to increase the regularization strength for higher powers x^k. Let's make the regularization strength for power k equal to λ^k, where $\lambda > 0$ is a hyperparameter. Write down the regression loss of the form

$$\frac{1}{n}\|Xw - y\|^2 + \text{regularizer}$$

where the regularizer penalizes the squared coefficient w_k^2 (corresponding to feature x^k) with strength λ^k. Express your answer using vector notation, where $w = [w_0, w_1, \ldots, w_{d-1}]^T$.

(b) Find the optimal solution w^* to this regularized linear regression problem in closed form. What λ values would make sense if we want the components w_k corresponding to higher degree features to be smaller?

(c) Now, we may want the coefficients w_k and w_{k+1} to be similar to each other for consecutive powers x^k and x^{k+1}. Come up with a regularization strategy based on squared penalties that will encourage this smoothness property. Write down the corresponding loss function. Is it possible to solve the resulting optimization problem in closed form? You don't need to solve it explicitly, just make an argument about whether a closed-form solution exists and explain your reasoning.

Solution

We work with polynomial features $\phi(x) = [1, x, x^2, \ldots, x^{d-1}]$ and parameter vector $w = [w_0, w_1, \ldots, w_{d-1}]^\top \in \mathbb{R}^d$. Let $X \in \mathbb{R}^{n\times d}$ be the design matrix whose i-th row is $\phi(x_i)^\top$, and $y \in \mathbb{R}^n$ the response vector.

(a) Loss with per-coordinate geometric penalties. Let $\Lambda \in \mathbb{R}^{d\times d}$ be diagonal with

$$\Lambda \triangleq \operatorname{diag}\big(\lambda^0, \lambda^1, \ldots, \lambda^{d-1}\big), \qquad \lambda > 0.$$

Penalizing w_k^2 with strength λ^k gives the regularizer $\sum_{k=0}^{d-1} \lambda^k w_k^2 = w^\top \Lambda w$. Hence the objective is

$$\boxed{\mathcal{L}(w) \;=\; \frac{1}{n}\,\|Xw - y\|_2^2 \;+\; w^\top \Lambda w.}$$

Instructor remark. In many texts a scalar ridge weight $\alpha > 0$ multiplies the whole regularizer, i.e. $\alpha\, w^\top \Lambda w$. Your statement fixes the strength of the k-th coordinate to λ^k directly; that's fine, just be aware it implicitly sets the base penalty scale. Also, since the feature 1 appears, you are penalizing the intercept w_0 (because $\lambda^0 = 1$). If you prefer not to penalize the intercept, set the $(0,0)$-entry of Λ to 0.

(b) Closed-form optimizer. $\mathcal{L}(w)$ is a strictly convex quadratic when $\Lambda \succcurlyeq 0$ and at least one of $\frac{1}{n}X^\top X$ or Λ is positive definite. Its gradient is

$$\nabla \mathcal{L}(w) \;=\; \frac{2}{n}\, X^\top (Xw - y) + 2\,\Lambda w.$$

Setting the gradient to zero gives the normal equations

$$\left(\tfrac{1}{n}X^\top X + \Lambda\right) w^* \;=\; \tfrac{1}{n}X^\top y.$$

Thus

$$\boxed{w^* \;=\; \left(\tfrac{1}{n}X^\top X + \Lambda\right)^{-1} \tfrac{1}{n}X^\top y.}$$

This solution is unique whenever $\frac{1}{n}X^\top X + \Lambda$ is invertible (e.g., always if $\lambda > 0$ so that $\Lambda \succ 0$, or more generally if the sum is positive definite).

Instructor remark. If you decide not to penalize w_0, replace Λ by $\mathrm{diag}(0, \lambda^1, \ldots, \lambda^{d-1})$. The same formula holds if the matrix remains invertible (typically true unless the data are perfectly collinear in the unpenalized subspace).

(c) Smoothness across consecutive coefficients. A standard way to encourage $w_k \approx w_{k+1}$ is to penalize squared first differences:

$$\sum_{k=0}^{d-2} \mu_k \,(w_{k+1} - w_k)^2 \quad \text{with} \quad \mu_k \geq 0.$$

Let $D \in \mathbb{R}^{(d-1)\times d}$ be the first-difference operator

$$D \;=\; \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}, \qquad \text{so that } (Dw)_k = w_{k+1} - w_k.$$

Let $M = \mathrm{diag}(\mu_0, \ldots, \mu_{d-2}) \succeq 0$. Then the smoothness penalty is

$$\|M^{1/2} D w\|_2^2 \;=\; w^\top D^\top M D\, w.$$

A corresponding loss (pure smoothness, or combined with part (a)) is

$$\boxed{\mathcal{L}_{\text{smooth}}(w) \;=\; \frac{1}{n}\,\|Xw - y\|_2^2 \;+\; w^\top D^\top M D\, w} \quad \text{or} \quad \boxed{\mathcal{L}_{\text{combo}}(w) \;=\; \frac{1}{n}\,\|Xw - y\|_2^2 \;+\; w^\top \Lambda w \;+\; w^\top D^\top M D\, w.}$$

These are convex quadratic objectives with closed-form solutions obtained by solving a linear system. For the combined penalty:

$$\nabla \mathcal{L}_{\text{combo}}(w) \;=\; \frac{2}{n} X^\top (Xw - y) + 2\Lambda w + 2 D^\top M D\, w,$$

so the optimal w^* satisfies

$$\left(\tfrac{1}{n}X^\top X + \Lambda + D^\top M D\right) w^* \;=\; \tfrac{1}{n}X^\top y,$$

and hence

$$\boxed{w^* \;=\; \left(\tfrac{1}{n}X^\top X + \Lambda + D^\top M D\right)^{-1} \tfrac{1}{n}X^\top y.}$$

A unique solution exists whenever the coefficient matrix is invertible (e.g., if $\Lambda \succ 0$, or more generally if the sum is positive definite).

Instructor remarks and possible tweaks.

- For the last part, students are not expected to write the solution in a matrix form. It's totally fine if they write it as a sum of many scalar terms. They also don't need to come up with the closed-form solution.

Problem 2: Convergence of Gradient Descent for Linear Regression (20 points)

Consider unregularized linear regression without an explicit bias term. Given a design matrix $X \in \mathbb{R}^{n \times d}$, response vector $y \in \mathbb{R}^n$, and parameter vector $w \in \mathbb{R}^d$, the loss function is defined as:

$$L(w) = \frac{1}{n}\|Xw - y\|^2$$

Your goal is to prove that gradient descent converges to the optimal solution through the following steps.

(a) Write down the gradient descent update rule explicitly in terms of the parameters w. That is, compute $\nabla_w L(w)$ and express the update:

$$w_{t+1} = w_t - \eta \nabla_w L(w_t)$$

where $\eta > 0$ is the learning rate.

(b) Rewrite the gradient descent update rule from Part (1) in the form:

$$w_{t+1} = Aw_t + b.$$

for some matrix $A \in \mathbb{R}^{d \times d}$ and vector $b \in \mathbb{R}^d$.

Suppose we know that the recursion $w_{t+1} = Aw_t + b$ converges to some limit w_∞ as $t \to \infty$. What is w_∞ equal to? Express your answer in terms of A and b; then, express the answer in terms of X, y, n, η. Carefully state any additional assumptions (e.g. invertibility) you might need.

(c) Using the result from Part (b), express w_t in terms of the initialization w_0, the matrix A, the vector b, and the iteration number t.

Now suppose $w_0 = 0$. Derive the limit:

$$w_\infty = \lim_{t \to \infty} w_t.$$

Hint: For a matrix A with all eigenvalues in the interval $(-1, 1)$, we have:

$$(I - A)^{-1} = I + A + A^2 + A^3 + \ldots = \sum_{k=0}^{\infty} A^k.$$

(d) Complete the proof: show that for an appropriate choice of learning rate η, the matrix A from Part (b) will have all eigenvalues in the interval $(-1, 1)$. Then, show that $A^t w_0 \to 0$ as $t \to \infty$ for any initialization $w_0 \in \mathbb{R}^d$, given that A has eigenvalues in $(-1, 1)$.

(e) Now suppose we use a stochastic (noisy) gradient of the form:

$$g_t = \nabla_w L(w_t) + \epsilon_t$$

where $\epsilon_t \in \mathbb{R}^d$ is a noise vector with zero mean: $\mathbb{E}[\epsilon_t] = 0$ for all t.

The update rule becomes:

$$w_{t+1} = w_t - \eta g_t.$$

What is the expected value $\mathbb{E}[w_t]$ in this case? Express it in terms of w_0, X, y, n, t, η. How does it compare to the deterministic gradient descent solution?

Solution

We consider the (unregularized) squared-loss

$$L(w) = \frac{1}{n}\|Xw - y\|^2, \qquad X \in \mathbb{R}^{n \times d},\ y \in \mathbb{R}^n,\ w \in \mathbb{R}^d,$$

with no explicit bias term. Let $H := X^\top X \succeq 0$.

(a) Update rule. Using $\nabla_w \|a\|^2 = 2J^\top a$ with $a = Xw - y$ and $J = X$, we have

$$\nabla_w L(w) \;=\; \frac{2}{n} X^\top (Xw - y) \;=\; \frac{2}{n}\big(Hw - X^\top y\big).$$

Gradient descent with stepsize $\eta > 0$ gives

$$w_{t+1} \;=\; w_t - \eta \nabla_w L(w_t) \;=\; w_t - \frac{2\eta}{n}\big(Hw_t - X^\top y\big).$$

(b) Affine recursion and its limit. Rewrite the update as

$$w_{t+1} \;=\; Aw_t + b, \qquad A \coloneqq I - \frac{2\eta}{n} H, \quad b \coloneqq \frac{2\eta}{n} X^\top y.$$

If the recursion converges to some limit w_∞, taking $t \to \infty$ on both sides yields

$$w_\infty \;=\; Aw_\infty + b \quad \Longrightarrow \quad (I - A)w_\infty = b.$$

Whenever $I - A$ is invertible on the relevant subspace, the fixed point is

$$w_\infty \;=\; (I - A)^{-1} b.$$

Plugging our A, b,

$$I - A = \frac{2\eta}{n} H \quad \Longrightarrow \quad w_\infty \;=\; \Big(\tfrac{2\eta}{n} H\Big)^{-1} \Big(\tfrac{2\eta}{n} X^\top y\Big).$$

Full column rank case: if $H = X^\top X$ is invertible, then

$$w_\infty \;=\; (X^\top X)^{-1} X^\top y,$$

the ordinary least-squares (OLS) solution. *Rank-deficient case:* the limit equals the minimum-norm least-squares solution $X^\dagger y$ (see part (d) for the convergence statement and the role of initialization); algebraically, $(I - A)^{-1}$ should be interpreted as the inverse on range(H), i.e., $H^\dagger$.

(c) Closed form for w_t and the limit from $w_0 = 0$. The affine recursion unrolls to

$$w_t \;=\; A^t w_0 + \sum_{k=0}^{t-1} A^k b.$$

If $\rho(A) < 1$ (all eigenvalues of A strictly inside the unit disk), then $A^t \to 0$ and the Neumann series converges:

$$\sum_{k=0}^{\infty} A^k \;=\; (I - A)^{-1}.$$

Thus for any w_0,

$$\lim_{t \to \infty} w_t \;=\; (I - A)^{-1} b,$$

and specifically for $w_0 = 0$,

$$w_\infty \;=\; \Big(\sum_{k=0}^{\infty} A^k\Big)\, b \;=\; (I - A)^{-1} b.$$

In the full-rank case this equals $(X^\top X)^{-1} X^\top y$; when H is singular, the same formula holds on range(H) and, starting from $w_0 = 0$, the limit is the minimum-norm solution $X^\dagger y$ (the component in $\ker(H)$ remains zero for all t).

(d) Stepsize condition and decay of $A^t w_0$. The eigenvalues of A are

$$\lambda_i(A) \;=\; 1 - \frac{2\eta}{n}\lambda_i(H), \qquad \lambda_i(H) \geq 0.$$

Requiring all $\lambda_i(A) \in (-1, 1)$ is equivalent to

$$-1 \;<\; 1 - \frac{2\eta}{n}\lambda_i(H) \;<\; 1 \iff 0 \;<\; \frac{2\eta}{n}\lambda_i(H) \;<\; 2$$

for every i with $\lambda_i(H) > 0$. Hence the stepsize condition is

$$0 \;<\; \eta \;<\; \frac{n}{\lambda_{\max}(H)}.$$

Equivalently, since ∇L is L-Lipschitz with $L = \frac{2}{n}\lambda_{\max}(H)$, this is $0 < \eta < \frac{2}{L}$.

Under this condition, the spectral radius $\rho(A) < 1$, so $A^t \to 0$ on range(H) and therefore $A^t w_0 \to 0$ for any w_0.

$$\boxed{\text{If } X^\top X \text{ invertible and } 0 < \eta < \frac{n}{\lambda_{\max}(X^\top X)}: \qquad w_t \to (X^\top X)^{-1} X^\top y.}$$

(e) Stochastic (noisy) gradients, expectation of the iterates. Let $g_t = \nabla L(w_t) + \epsilon_t$ with $\mathbb{E}[\epsilon_t] = 0$ for all t (no independence assumption is needed for the *expectation* due to linearity of ∇L). The update is

$$w_{t+1} \;=\; w_t - \eta g_t \;=\; \Big(I - \tfrac{2\eta}{n}H\Big) w_t + \tfrac{2\eta}{n} X^\top y \;-\; \eta\,\epsilon_t \;=\; A w_t + b - \eta \epsilon_t.$$

Taking expectations and using $\mathbb{E}[\nabla L(w_t)] = \frac{2}{n} H\,\mathbb{E}[w_t] - \frac{2}{n} X^\top y$,

$$\mathbb{E}[w_{t+1}] \;=\; A\,\mathbb{E}[w_t] + b.$$

Hence

$$\boxed{\mathbb{E}[w_t] \;=\; A^t w_0 + \sum_{k=0}^{t-1} A^k b},$$

which is *exactly* the same affine recursion as in the deterministic case. Therefore, under the same stepsize condition as in (d), $\mathbb{E}[w_t]$ converges to the same limit as deterministic GD:

$$\mathbb{E}[w_t] \;\to\; (I - A)^{-1} b = \begin{cases} (X^\top X)^{-1} X^\top y, & \text{if } X^\top X \text{ invertible,} \\ X^\dagger y, & \text{if } X^\top X \text{ singular and } w_0 \in \text{range}(X^\top). \end{cases}$$

(With additional assumptions on ϵ_t one can also characterize the steady-state covariance, but the expectation matches the deterministic trajectory.)

Instructor notes / feedback.

- Each part is 4 points.
- *Rank deficiency:* As written, part (b) might tempt students to invert $I - A = \frac{2\eta}{n} X^\top X$. This is fine if you assume X has full column rank; the assumption needs to be clearly stated, otherwise remove 1 point.

Problem 3: Convexity Analysis (16 points)

In this problem, we will work on convexity. Recall that a function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be **convex** if for any two points $x_1, x_2 \in \mathbb{R}^d$ and any scalar $t \in [0, 1]$, we have

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

Use this definition to prove convexity below. If you want to use other definitions or sufficient conditions for convexity, prove that they follow from this definition.

(a) Prove that the quadratic function of a scalar input $f(x) = x^2$ is convex.

(b) Prove that a shifted quadratic function of a scalar input $f(x) = (x-b)^2$ is convex, where $b \in \mathbb{R}$.

(c) Prove that the linear regression loss function $f(w) = \|Xw - y\|^2$ is convex, where $X \in \mathbb{R}^{n \times d}$, $w \in \mathbb{R}^d$, and $y \in \mathbb{R}^n$.

(d) Prove that the scalar function $f(x) = |\sin x + 0.5x|$ is not convex. You may present specific values $x_1, x_2 \in \mathbb{R}$ and $t \in [0, 1]$ that violate the definition of convexity. (It is acceptable to compute the values of $\sin x$ using a calculator or programming language such as Python.)

Additionally, explain what would happen if we run gradient descent on this function initialized at $x_0 = 10$ with a small step size.

Solution

Throughout, recall the definition: f is convex on $\mathbb{R}^d$ iff $f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$ for all x_1, x_2 and $t \in [0, 1]$.

(a) $f(x) = x^2$ is convex. For $x_1, x_2 \in \mathbb{R}$ and $t \in [0, 1]$,

$$\big(tx_1 + (1-t)x_2\big)^2 = tx_1^2 + (1-t)x_2^2 - t(1-t)(x_1 - x_2)^2 \;\leq\; tx_1^2 + (1-t)x_2^2,$$

since $(x_1 - x_2)^2 \geq 0$. This is exactly the convexity inequality.

(b) $f(x) = (x-b)^2$ is convex (direct proof). Fix $x_1, x_2 \in \mathbb{R}$ and $t \in [0, 1]$. Consider

$$f\big(tx_1 + (1-t)x_2\big) = \big(tx_1 + (1-t)x_2 - b\big)^2 = \big(t(x_1 - b) + (1-t)(x_2 - b)\big)^2.$$

Expand the square:

$$\big(t(x_1 - b) + (1-t)(x_2 - b)\big)^2 = t(x_1 - b)^2 + (1-t)(x_2 - b)^2 - t(1-t)\big((x_1 - b) - (x_2 - b)\big)^2.$$

Since $\big((x_1 - b) - (x_2 - b)\big)^2 = (x_1 - x_2)^2 \geq 0$, we have

$$\big(t(x_1 - b) + (1-t)(x_2 - b)\big)^2 \;\leq\; t(x_1 - b)^2 + (1-t)(x_2 - b)^2.$$

Therefore,

$$f\big(tx_1 + (1-t)x_2\big) \;\leq\; tf(x_1) + (1-t)f(x_2),$$

which is exactly the convexity inequality. Equality holds iff $x_1 = x_2$ or $t \in \{0, 1\}$.

(c) $f(w) = \|Xw - y\|^2$ is convex (direct proof). Fix $w_1, w_2 \in \mathbb{R}^d$ and $t \in [0,1]$. Set

$$z_1 \triangleq Xw_1 - y, \qquad z_2 \triangleq Xw_2 - y \ \in \ \mathbb{R}^n.$$

Then, by linearity of X,

$$f\big(tw_1 + (1-t)w_2\big) = \big\|X\big(tw_1 + (1-t)w_2\big) - y\big\|^2 = \big\|t(Xw_1 - y) + (1-t)(Xw_2 - y)\big\|^2 = \|tz_1 + (1-t)z_2\|^2.$$

Expand the squared norm explicitly:

$$\|tz_1 + (1-t)z_2\|^2 = \big(tz_1 + (1-t)z_2\big)^\top \big(tz_1 + (1-t)z_2\big) = t^2\|z_1\|^2 + (1-t)^2\|z_2\|^2 + 2t(1-t)\langle z_1, z_2\rangle.$$

Now add and subtract $t(1-t)\|z_1\|^2 + t(1-t)\|z_2\|^2$:

$$\begin{aligned}\|tz_1 + (1-t)z_2\|^2 &= t\|z_1\|^2 + (1-t)\|z_2\|^2 - t(1-t)\big(\|z_1\|^2 + \|z_2\|^2 - 2\langle z_1, z_2\rangle\big)\\ &= t\|z_1\|^2 + (1-t)\|z_2\|^2 - t(1-t)\|z_1 - z_2\|^2.\end{aligned}$$

Since $\|z_1 - z_2\|^2 \geq 0$, we obtain

$$\|tz_1 + (1-t)z_2\|^2 \ \leq \ t\|z_1\|^2 + (1-t)\|z_2\|^2.$$

Recalling $z_i = Xw_i - y$, this is precisely

$$f\big(tw_1 + (1-t)w_2\big) \ \leq \ tf(w_1) + (1-t)f(w_2),$$

which proves convexity by the definition.

(d) $f(x) = |\sin x + 0.5\,x|$ is not convex. Take $x_1 = 0$, $x_2 = \pi$, and $t = \frac{1}{2}$. Then

$$f(0) = |\sin 0 + 0| = 0, \qquad f(\pi) = |\sin \pi + 0.5\pi| = |0 + \tfrac{\pi}{2}| \approx 1.5708,$$

so $\frac{1}{2}f(0) + \frac{1}{2}f(\pi) \approx 0.7854$. At the midpoint $x = \frac{\pi}{2}$,

$$f\big(\tfrac{\pi}{2}\big) = \Big|\sin\big(\tfrac{\pi}{2}\big) + 0.5 \cdot \tfrac{\pi}{2}\Big| = |1 + \tfrac{\pi}{4}| \approx 1.7854.$$

Thus $f(\frac{\pi}{2}) > \frac{1}{2}f(0) + \frac{1}{2}f(\pi)$, violating convexity.

What happens with gradient descent from $x_0 = 10$ (small step size)? Let $g(x) = \sin x + 0.5\,x$. For $x \geq 2$, $g(x) > 0$ (since $0.5x$ dominates), so near $x_0 = 10$ we have $f(x) = g(x)$ and f is smooth with

$$f'(x) = g'(x) = \cos x + 0.5, \qquad f''(x) = -\sin x.$$

Gradient descent updates $x_{t+1} = x_t - \eta(\cos x_t + 0.5)$. Fixed points (in this smooth region) satisfy $\cos x = -0.5$, i.e.

$$x \in \big\{\tfrac{2\pi}{3} + 2\pi k \ , \ \tfrac{4\pi}{3} + 2\pi k\big\}_{k \in \mathbb{Z}}.$$

Among these, points with $x = \frac{4\pi}{3} + 2\pi k$ are *local minima* since $f''(x) = -\sin x > 0$ there (indeed $\sin(\frac{4\pi}{3}) = -\frac{\sqrt{3}}{2}$). The one closest to $x_0 = 10$ is $x^\star = \frac{4\pi}{3} + 2\pi \cdot 1 \approx 10.471$. For sufficiently small η (e.g., $\eta < 2/f''(x^\star) \approx 2/(\sqrt{3}/2)$), gradient descent will monotonically converge to this nearby *local* minimizer around $x^\star \approx 10.47$, not to the global minimizer at $x = 0$ (where $f(0) = 0$). Hence, on this nonconvex, nonsmooth objective, initialization matters and GD settles into a nearby basin.

Problem 4: Backpropagation (20 points)

Consider a ReLU MLP neural network with:

- 1 input dimension
- 1 hidden layer with h hidden units

- 1 output dimension
- A skip connection from input to output

We train on a single example (x, y) where $x, y \in \mathbb{R}$ using MSE loss.
Let the parameters be:

- $W_1 \in \mathbb{R}^{h \times 1}$: weights from input to hidden layer
- $b_1 \in \mathbb{R}^h$: biases for hidden layer
- $W_2 \in \mathbb{R}^{1 \times h}$: weights from hidden to output layer
- $b_2 \in \mathbb{R}$: bias for output layer

(a) (5 points) Write down the complete mathematical formulas for the forward pass and loss computation. Be explicit about each step. Use $\odot$ to denote element-wise multiplication.

(b) (5 points) Following the backpropagation algorithm, compute the gradients with respect to all parameters of the model and the input x. Show your work step by step, clearly indicating how you apply the chain rule at each stage.

You should compute: $\frac{\partial \mathcal{L}}{\partial b_2}$, $\frac{\partial \mathcal{L}}{\partial W_2}$, $\frac{\partial \mathcal{L}}{\partial W_1}$, $\frac{\partial \mathcal{L}}{\partial b_1}$, $\frac{\partial \mathcal{L}}{\partial x}$.

Feel free to ignore the fact that ReLU is not differentiable at zero for this derivation: assume that all the pre-activation values are not equal to 0.

(c) (5 points) How does the skip connection affect the parameter gradients? Specifically, address the following questions:

- Examine the formula for $\frac{\partial \mathcal{L}}{\partial x}$ that you derived in part (b) and identify the contribution from the skip connection versus the contribution from the standard path through the network.
- Does the skip connection create a "gradient highway" in this architecture? Explain what this means and whether it applies here.
- For which parameters does the skip connection help prevent vanishing gradients, and for which parameters does it NOT help? Provide a clear explanation for your answer.
- Discuss how the skip connection affects gradient flow to the input versus gradient flow to the parameters of the hidden layer.
- How would your answer change if we had a much deeper network with many hidden layers instead of just one?

(d) (5 points) Implement the model in PyTorch and verify your analytical gradients using autograd. Submit your python code and a brief discussion, including comparison of your analytical formulas to the autograd results.

Solution

Solution: Problem 4 — Backpropagation with a Skip (20 pts)

We use the ReLU $\sigma(z) = \max\{0, z\}$ with derivative $\sigma'(z) = \mathbf{1}\{z > 0\}$ (choose subgradient 0 at $z = 0$).

(a) Forward pass and loss (5 pts). Given a single example $(x, y) \in \mathbb{R} \times \mathbb{R}$,

$$z_1 = W_1 x + b_1 \in \mathbb{R}^h, \qquad a_1 = \sigma(z_1) \in \mathbb{R}^h,$$

$$\hat{y} = W_2 a_1 + b_2 + x \quad \text{(skip connection)},$$

$$\mathcal{L} = \tfrac{1}{2}(\hat{y} - y)^2.$$

We will also use $s \triangleq \sigma'(z_1) \in \{0, 1\}^h$ and the elementwise product $\odot$ when needed.

(b) Gradients (5 pts). Let $e \triangleq \hat{y} - y$. Then $\frac{\partial \mathcal{L}}{\partial \hat{y}} = e$.

Output layer and skip:

$$\frac{\partial \mathcal{L}}{\partial b_2} = e, \qquad \frac{\partial \mathcal{L}}{\partial W_2} = e\, a_1^\top \in \mathbb{R}^{1\times h}.$$

Backprop into a_1 and z_1:

$$\frac{\partial \mathcal{L}}{\partial a_1} = W_2^\top e \in \mathbb{R}^h, \qquad \frac{\partial \mathcal{L}}{\partial z_1} = \left(W_2^\top e\right) \odot s \in \mathbb{R}^h.$$

Hidden layer parameters: Since $z_1 = W_1 x + b_1$ with $W_1 \in \mathbb{R}^{h\times 1}$,

$$\frac{\partial \mathcal{L}}{\partial W_1} = \left(\tfrac{\partial \mathcal{L}}{\partial z_1}\right) x^\top = \left((W_2^\top e) \odot s\right) x \in \mathbb{R}^{h\times 1},$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial z_1} = (W_2^\top e) \odot s \in \mathbb{R}^h.$$

Gradient w.r.t. input x: there are two paths, the skip and the hidden path.

$$\frac{\partial \hat{y}}{\partial x} = 1 + W_2 \,\mathrm{Diag}(s)\, W_1 = 1 + \sum_{k=1}^{h} W_{2,k}\, s_k\, W_{1,k} \in \mathbb{R}.$$

Hence

$$\boxed{\frac{\partial \mathcal{L}}{\partial x} = e\left(1 + \sum_{k=1}^{h} W_{2,k}\, s_k\, W_{1,k}\right).}$$

(c) Effect of the skip on gradients (5 pts).

- **Decomposing $\partial\mathcal{L}/\partial x$.** From the box above,

$$\frac{\partial \mathcal{L}}{\partial x} = \underbrace{e}_{\text{skip path}} + \underbrace{e \sum_k W_{2,k}\, s_k\, W_{1,k}}_{\text{hidden path}}.$$

  The term e is *exactly* the contribution of the skip connection.

- **"Gradient highway"?** A gradient highway is a direct, unattenuated route for gradients to flow. Here the skip creates a highway *to* x (since $\partial\hat{y}/\partial x$ has a constant 1 term). With only one hidden layer, there is no deep stack to rescue, so the benefit is mainly the clean e term to the input.

- **Which parameter gradients benefit?** The parameter gradients remain

$$\frac{\partial \mathcal{L}}{\partial W_2} = e\, a_1^\top, \quad \frac{\partial \mathcal{L}}{\partial b_2} = e, \quad \frac{\partial \mathcal{L}}{\partial W_1} = \left((W_2^\top e) \odot s\right) x, \quad \frac{\partial \mathcal{L}}{\partial b_1} = (W_2^\top e) \odot s.$$

  The skip does *not* add a new direct path from the loss to W_1 or b_1; those still pass through s (ReLU gates) and W_2. Thus it does not prevent vanishing for hidden-layer parameters when activations are mostly off ($s \approx 0$) or W_2 is small. It trivially helps b_2 only insofar as e changes with the forward value, but there is no structural rescue like for x.

- **Gradient flow to input vs. to hidden params.** The skip guarantees a nonzero route to the input (e term), regardless of s or W_2. In contrast, $\partial\mathcal{L}/\partial W_1$ and $\partial\mathcal{L}/\partial b_1$ still rely on $(W_2^\top e) \odot s$ and can vanish if ReLUs are inactive.

- **If the network were much deeper.** In deep nets, residual (skip) links between layers create highways that alleviate vanishing/exploding by adding identity paths in the Jacobian. Then gradients to *earlier layers' parameters* inherit direct components that bypass many nonlinearities. Our single residual to the input is the shallow analogue of that idea, but it only helps the input here.

(d) PyTorch check (5 pts). Below is a minimal script that (i) computes forward and autograd gradients, (ii) computes *analytical* gradients using the formulas above, and (iii) compares them.

```
import torch

torch.manual_seed(0)
h = 5
x = torch.randn((), requires_grad=True)
y = torch.randn(())

W1 = torch.randn(h, 1, requires_grad=True)
b1 = torch.randn(h,    requires_grad=True)
W2 = torch.randn(1, h, requires_grad=True)
b2 = torch.randn((),   requires_grad=True)

# ----- Autograd -----
z1 = W1 @ x.view(1,1) + b1.view(h,1)      # shape (h,1)
a1 = torch.relu(z1)                       # (h,1)
y_hat = (W2 @ a1).view(()) + b2 + x       # scalar, includes skip
L = 0.5 * (y_hat - y)**2
L.backward()

grads_auto = {
    "db2": b2.grad.item(),
    "dW2": W2.grad.clone(),               # (1,h)
    "dW1": W1.grad.clone(),               # (h,1)
    "db1": b1.grad.clone(),               # (h,)
    "dx" : x.grad.item(),
}

# ----- Analytical -----
with torch.no_grad():
    e = (y_hat - y).item()
    s = (z1.view(-1) > 0).float()         # indicator vector (h,)
    a1_flat = a1.view(-1)                 # (h,)

    dL_db2 = e
    dL_dW2 = e * a1_flat.view(1, -1)      # (1,h)
    tmp = (W2.view(-1) * e) * s           # (h,)
    dL_dW1 = (tmp.view(-1,1)) * x.item()  # (h,1)
    dL_db1 = tmp.clone()                  # (h,)
    dL_dx  = e * (1.0 + (W2.view(-1) * s * W1.view(-1)).sum().item())

grads_manual = {
    "db2": dL_db2,
    "dW2": dL_dW2,
    "dW1": dL_dW1,
    "db1": dL_db1,
    "dx" : dL_dx,
}

# ----- Sanity check -----
def max_abs_diff(A, B):
    if isinstance(A, torch.Tensor):
        A = A.detach()
    if isinstance(B, torch.Tensor):
```

```
        B = B.detach()
    return (A - B).abs().max().item()

print("max|db2|:", abs(grads_auto["db2"] - grads_manual["db2"]))
print("max|dW2|:", max_abs_diff(grads_auto["dW2"], grads_manual["dW2"]))
print("max|dW1|:", max_abs_diff(grads_auto["dW1"], grads_manual["dW1"]))
print("max|db1|:", max_abs_diff(grads_auto["db1"], grads_manual["db1"]))
print("max|dx| :", abs(grads_auto["dx"]  - grads_manual["dx"]))
```

Expected outcome. All max absolute differences print near machine precision (e.g., $\leq 10^{-7}$), confirming the analytical gradients.

Instructor notes.

- The $\frac{1}{2}$ in $\mathcal{L}$ avoids an extra factor of 2 in gradients; if you omit it, multiply every gradient by 2.