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.

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?

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.

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.