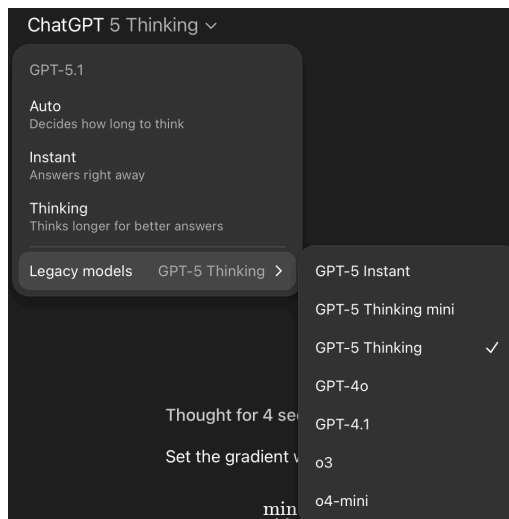


Special Participation A: Homework 2 with GPT-5

Executive summary: In this special participation, I interactively engage GPT-5 on the non-coding parts of Homework 2. My experience using it was boring and it one-shotted every question. I did not have to use any special strategies or gesture towards an answer to make it answer the problems correctly. I noticed one slight misconception in its reference to SignSGD and one missing transpose that made a column vector into a row vector. For the most part, it was hallucination free. However, it did actually manage to point out a typo on the homework on part (b) of problem 2, deducing that the problem is incorrect and is “ill-posed.”

First, I make sure to utilize chain-of-thought by selecting GPT-5 Thinking at the top.



For convenience, I use ChatGPT's image prompting feature since I do not want to type the homework LaTeX out.

1. Optimizers as Penalized Linear Improvement with different norm penalties

In lecture, you saw the locally linear perspective of a neural network and the loss by Taylor expanding the loss around the current value of the parameters. This approximation is only very good in a near neighborhood of those values. One way to proceed with optimization is to consider the size of the neighborhood as a hyperparameter and to bound our update to stay within that neighborhood while minimizing our linear approximation to the loss. You saw in lecture that the choice of norm in defining that neighborhood also matters.

In this problem, you will work out for yourself a slightly different perspective. Instead of treating the norm as a constraint (with the size of the acceptable norm as a hyperparameter), we can do an unconstrained optimization with a weighted penalty that corresponds to the squared norm — where that weight is a hyperparameter.

At each iteration, we wish to maximize linear improvement of the objective (as defined by the dot-product between the gradient and the update) locally regularized by a penalty on the size of the update. This can be expressed (in traditional minimization form) as:

$$u = \underset{\Delta\theta}{\operatorname{argmin}} \underbrace{g^T \Delta\theta}_{\text{Linear Improvement}} + \underbrace{\frac{1}{\alpha} d(\Delta\theta)}_{\text{Distance Penalty}}, \quad (1)$$

where $g = \nabla f(\theta)$ is the gradient of the loss, α is a scalar, and d is a scalar-output distance function $\mathbb{R}^{d(\text{model})} \rightarrow \mathbb{R}^+$.

(a) Let's assume *Euclidean distance* is the norm that captures our sense of relevant neighborhoods in parameter space. Then our objective can be:

$$u = \underset{\Delta\theta}{\operatorname{argmin}} g^T \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_2^2. \quad (2)$$

What is the analytical solution for u in the above problem? What standard optimizer does this recover?

Set the gradient w.r.t. $\Delta\theta$ to zero:

$$\min_{\Delta\theta} g^T \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_2^2 \Rightarrow \nabla_{\Delta\theta} \left(g^T \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_2^2 \right) = g + \frac{2}{\alpha} \Delta\theta = 0.$$

Thus

$$u = \Delta\theta^* = -\frac{\alpha}{2} g.$$

This is exactly **(stochastic) gradient descent** in the Euclidean norm, with step size $\eta = \alpha/2$.

I first feed the raw question to the model to see what it can do with it. It infers that it needs to complete (a). It one-shots it perfectly, without any text prompting strategy at all. This is notated pretty close to the staff solution as well which I attach here for reference.

Solution: To solve for u , we can take the gradient of the objective with respect to $\Delta\theta$ and set it to zero:

$$\nabla_{\Delta\theta} \left(g^T \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_2^2 \right) = g + \frac{2}{\alpha} \Delta\theta = 0. \quad (3)$$

Then, solving for $\Delta\theta$, we get:

$$\Delta\theta = -\frac{\alpha}{2} g, \quad (4)$$

(b) Now, consider an alternative way of capturing local neighborhood size – the infinity norm over parameters. Recall that this is defined as $\|x\|_\infty = \max_i |x_i|$. Our objective is now:

$$u = \operatorname{argmin}_{\Delta\theta} g^\top \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_\infty. \quad (3)$$

What is the analytical solution for u in this case? Which optimizer does this correspond to?

Thought for 36s >

Short answer: the penalized problem with an ℓ_∞ norm is generally **ill-posed**.

$$\min_{\Delta\theta} g^\top \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_\infty$$

For any radius $t \geq 0$, the best point with $\|\Delta\theta\|_\infty \leq t$ is

$\Delta\theta = -t \operatorname{sign}(g)$, giving objective

$$-t\|g\|_1 + \frac{t}{\alpha} = t\left(\frac{1}{\alpha} - \|g\|_1\right).$$

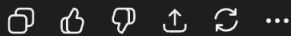
Hence:

- If $\|g\|_1 < \frac{1}{\alpha}$, the minimum occurs at $t = 0$, so $u = \Delta\theta^* = \mathbf{0}$.
- If $\|g\|_1 = \frac{1}{\alpha}$, every $t \geq 0$ tied; any $u = -t \operatorname{sign}(g)$ is optimal.
- If $\|g\|_1 > \frac{1}{\alpha}$, the objective is unbounded below (finite minimizer).

Because of that, people usually use the **constrained** form with an ℓ_∞ radius ρ :

$$\min_{\|\Delta\theta\|_\infty \leq \rho} g^\top \Delta\theta \implies u^* = -\rho \operatorname{sign}(g),$$

which corresponds to the **sign gradient descent** update (also the FGSM-style ℓ_∞ step).



Here I try the same strategy with (b), only to realize that my version of (b) is wrong. Interestingly, it catches the error in the problem, but does not infer what the problem should be.

(b) Now, consider an alternative way of capturing local neighborhood size – the squared infinity norm over parameters. Recall that this is defined as $\|x\|_\infty = \max_i |x_i|$. Our objective is now:

$$u = \underset{\Delta\theta}{\operatorname{argmin}} g^\top \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_\infty^2. \quad (6)$$

What is the analytical solution for u in this case? Which optimizer does this correspond to?

here is the corrected version of (b)

Thought for 11s >

$$u = \underset{\Delta\theta}{\operatorname{argmin}} g^\top \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_\infty^2$$

Let $t = \|\Delta\theta\|_\infty = \max_i |\Delta\theta_i|$.

For a fixed t , the linear term is minimized by pushing each coordinate to the boundary with the opposite sign of the gradient:

$$\Delta\theta_i = -t \operatorname{sign}(g_i) \Rightarrow g^\top \Delta\theta = -t \sum_i |g_i| = -t \|g\|_1.$$

The objective becomes a 1-D convex problem in $t \geq 0$:

$$\phi(t) = -t \|g\|_1 + \frac{1}{\alpha} t^2.$$

Minimize: $\phi'(t) = -\|g\|_1 + \frac{2}{\alpha} t = 0 \Rightarrow t^* = \frac{\alpha}{2} \|g\|_1$.

Therefore

$$u^* = \Delta\theta^* = -\frac{\alpha}{2} \|g\|_1 \operatorname{sign}(g) \quad (\text{with } u^* = \mathbf{0} \text{ if } g = \mathbf{0}).$$

This corresponds to a **sign gradient descent** step (a.k.a. signSGD), i.e., moving in the element-wise sign of the gradient, with a step magnitude proportional to $\|g\|_1$.

Here, I prompt it with the actual problem I obtained from another pdf. It one-shots it again. I'm pretty sure that the math checks out as well. One slight misconception is that it refers to this as SignSGD even though SignSGD doesn't scale by the 1-norm of the gradient.

2. Optimizers and their convergence

Consider \mathcal{O} : a simplified Adam-style optimizer without weight decay that has iterates

$$\theta_{t+1} \leftarrow \theta_t - \alpha_t M_t \nabla f_t(\theta_t) \quad (4)$$

where f_t is the loss at iteration t and α_t is the step size (learning rate).

Further suppose that the adaptive scaling matrix M_t is recomputed over each epoch of training and just consists of a diagonal populated by the inverses of the square roots of the mean squared value for the gradients during the epoch for that specific coordinate.

For this part, we have exactly $n = 1$ training point corresponding to the single equation

$$[1, 0.1, 0.01]\theta = 1 \quad (5)$$

with a 3-dimensional learnable parameters θ . Suppose that we start with $\theta_0 = \mathbf{0}$ and use squared loss $f_t(\theta) = (1 - [1, 0.1, 0.01]\theta)^2$.

(a) What specific vector θ would standard vanilla SGD (i.e. (4) with $M_t = I$ and $\alpha_t = \alpha$) converge to assuming $\alpha > 0$ was small enough to give convergence?



Thought for 10s >

Let $a = [1, 0.1, 0.01]^T$ and $f(\theta) = (1 - a^T \theta)^2$.

Then $\nabla f(\theta) = -2(1 - a^T \theta) a$, so vanilla GD with small $\alpha > 0$ updates only in the span of a . Starting at $\theta_0 = \mathbf{0}$, the limit must be the point in $\text{span}(a)$ satisfying the normal equation $a^T \theta = 1$:

$$\theta^* = \frac{1}{\|a\|_2^2} a = \frac{1}{1^2 + 0.1^2 + 0.01^2} [1, 0.1, 0.01] = \frac{1}{1.0101} [1, 0.1, 0.01] \approx [0.99, 0.099, 0.0099].$$

So SGD converges to $\theta^* = \frac{a}{\|a\|_2^2}$ (numerically $\approx [0.99, 0.099, 0.0099]$).

Doing problem 2, it one-shots the problem again. There is not much of interest here besides that.

(b) What specific vector θ would the simplified version of Adam \circlearrowleft converge to assuming appropriate step-sizes $\alpha_t > 0$ to give convergence?

Thought for 10s >

Let $a = [1, 0.1, 0.01]^T$ and $f(\theta) = (1 - a^\top \theta)^2$.

$$\nabla f(\theta) = -2(1 - a^\top \theta) a.$$

Over an epoch (there's only one sample), the per-coordinate RMS of the gradient is

$$\sqrt{\mathbb{E}[g_i^2]} = 2 |1 - a^\top \theta| |a_i|.$$

Thus

$$M_t = \text{diag}\left(\frac{1}{2|1 - a^\top \theta_t| |a_i|}\right) = c_t \text{diag}\left(\frac{1}{|a_i|}\right),$$

with a scalar $c_t = \frac{1}{2|1 - a^\top \theta_t|}$.

Therefore the Adam-style step direction is

$$-M_t \nabla f(\theta_t) \propto \text{g}\left(\frac{1}{|a_i|}\right) a = \mathbf{1},$$

since all $a_i > 0$. Starting from $\theta_0 = 0$, the iterates stay in $\text{span}(\mathbf{1})$ and converge to the point on the minimizer hyperplane $a^\top \theta = 1$ lying in that span:

$$\theta^* = \frac{1}{\sum_i a_i} \mathbf{1} = \frac{1}{1 + 0.1 + 0.01} [1, 1, 1] \approx [0.9009, 0.9009, 0.9009].$$

(If a had mixed signs, the limit would be $\frac{1}{\sum_i |a_i|} \text{sign}(a)$.)

GPT-5 one-shots this problem as well. One minor nitpick is that it doesn't add a transpose, so the optimal theta is a row vector even though it should be a column vector. Besides that, this solution is unremarkable.

(c) Consider a learning approach that first did training input feature rescaling (so that each feature had unit second-moment), then ran SGD to convergence, and then converted the solution for the rescaled problem back to the original units. What specific vector θ would it give as its final solution (for use in original coordinates)?

Thought for 10s >

Let $a = [1, 0.1, 0.01]^T$ and $f(\theta) = (1 - a^\top \theta)^2$.

Feature rescaling to unit second moment for each coordinate (with only one sample) uses

$$s_i = \frac{1}{|a_i|}, \quad x'_i = s_i a_i = \text{sign}(a_i).$$

So in the rescaled problem the feature vector is $a' = [1, 1, 1]$. Running SGD to convergence on $(1 - (a')^\top \theta')^2$ yields the minimum-norm solution in $\text{span}(a')$:

$$\theta'^* = \frac{a'}{\|a'\|_2^2} = \frac{1}{3} [1, 1, 1].$$

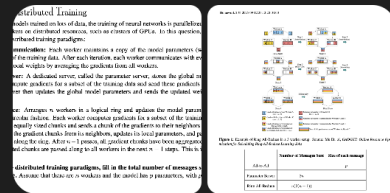
Mapping back to original coordinates (since $x' = Sx$ with $S = \text{diag}(s_i)$), the weights transform as $\theta = S\theta' \Rightarrow \theta_i = \theta'_i / x_i$. Hence

$$\theta^* = \left[\frac{1}{3}, \frac{10}{3}, \frac{100}{3} \right]$$

(which indeed satisfies $a^\top \theta^* = 1$).

This makes the same mistake of framing theta star as a row vector. It also writes a prime as a row vector as well. Other than that, the solution is fine.

I skip the next two problems and also (7) since they're coding problems.



Thought for 4 seconds >

Here are the fills:

- **All-to-All:**
messages: $n(n-1)$
size/message: p
- **Parameter Server:**
messages: $2n$ (n workers \rightarrow server, then server $\rightarrow n$ workers)
size/message: p
- **Ring All-Reduce:**
messages: $n \cdot 2(n-1)$
size/message: p/n

Thinking

- Filling in table for distributed training paradigms
I need to finish the table for distributed training paradigms, especially for All-to-All, Parameter Server, and Ring All-Reduce. The number of messages for All-to-All looks like it should be $n \cdot (n-1)$. For Parameter Server, the size per message is p , which makes sense based on the model's design. I'll figure out the rest.

✓ Thought for 4 seconds
Done

The last problem (5) was done in one shot as well very quickly. The answers check out with the solutions. It took a mere 4 seconds for me to get a response. Interestingly, the message is very laconic and non-conversational. I have two hypotheses for why this happened: maybe the answer structure (fill in the blank) caused this, or the backend used a router to send me to a

different model, likely Thinking mini.

GPT-5 Instant

GPT-5 Thinking mini ✓

GPT-5 Thinking