Introduction to Large Language Models
Assignment- 8

Number of questions: 10 **Total mark: 10 X 1 = 10**

QUESTION 1: [1 mark]

In standard instruction tuning with a decoder-only LM, which tokens typically contribute to the next-token prediction loss?

a) Only the prompt tokens
b) Only the response tokens
c) Both prompt and response tokens
d) Neither; loss is computed at the sequence level only

Correct Answer: b

Explanation:

Instruction tuning trains a model to generate a desired response given an instruction (prompt). In a decoder-only model, the prompt and response are concatenated (e.g., [prompt_tokens] [response_tokens]). The model's objective is to predict the next token.

The loss (e.g., cross-entropy) is only calculated for the **response tokens**. The prompt tokens serve as the conditioning context, but we are not training the model to predict the prompt itself, only to predict the correct *answer* that should follow it.

QUESTION 2: [1 mark]

Why can using multiple instruction templates for the same task help?

a) It only increases the dataset size.
b) It regularizes the reward model.
c) It improves generalization by exposing the model to different phrasings of the instruction.
d) It ensures the same tokenization across tasks.

Correct Answer: c

Explanation:

The goal of instruction tuning is to teach the model to follow instructions in general, not just to memorize a few specific task formats.

By training on **multiple templates**—which are different phrasings of the same underlying instruction (e.g., "Based on the paragraph above, can we conclude..." vs. "Can we infer the following?") —the model is exposed to a wider variety of linguistic structures.

This "rephrasing... helps the model learn and generalize more effectively" to new, unseen instructions at test time.

 (a) is a side effect, but not the primary reason.

 (b) is incorrect; instruction tuning is separate from training a reward model.

 (d) is incorrect; different phrasings will likely have different tokenizations.

QUESTION 3: [1 mark]

As the model size grows, what happens to prompt length and initialization sensitivity in prompt tuning?

a) Both matter more.
b) Both matter less.
c) Length matters less but initialization matters more.
d) Initialization matters less but length matters more.

Correct Answer: b

Explanation:

The lecture slides present experimental results on "Prompt Tuning," which uses continuous/soft prompts.

1. **Prompt Length:** As the model size increases, the performance gap between different prompt lengths becomes very small and they converge to a similar high score .
2. **Initialization:** While smaller models are very sensitive to how the soft prompt is initialized (e.g., "Random Uniform" performs poorly), larger models achieve high performance regardless of the initialization method.

Therefore, as models scale, they become more robust, and both prompt length and initialization "matter less."

QUESTION 4: [1 mark]

Which of the following statement(s) is/are true about the POSIX metric for quantifying prompt sensitivity?

a) POSIX is independent of the correctness of the generated responses and captures sensitivity as a property independent of correctness
b) POSIX is a length-normalized metric
c) POSIX compares the generated responses against the ground-truth to quantify prompt sensitivity

d) POSIX captures the variance in the log-likelihood of the same response for different input prompt variations

Correct Answer: a, b, d

Explanation:

- **(a) True:** The POSIX metric measures the *stability* of a model's output probabilities in response to prompt variations, not whether those outputs are factually correct.
- **(b) True:** The POSIX formula explicitly includes the term $1/L_{y_j}$ for length normalization, to accommodate arbitrary response lengths.
- **(c) False:** The POSIX formula only compares the model's probabilities for its *own* generated responses (yj) given different prompts (xi and xj). It does not use a ground-truth label.
- **(d) True:** The core of the metric is the log of the likelihood ratio, which directly captures the relative-change in log-likelihood of a response yj when the prompt is changed from xj to an intent-aligned variant xi. This is a measure of variance in the model's confidence.

QUESTION 5: [1 mark]

Which statement is true about prompt sensitivity as captured by POSIX?

a) Larger models always have lower prompt sensitivity than smaller ones.
b) Larger models always have higher prompt sensitivity than smaller ones.
c) Prompt sensitivity decreases for models with a parameter count above a certain threshold.
d) Increasing parameter count does not necessarily reduce prompt sensitivity.

Correct Answer: d

Explanation:

The experimental results in the lecture slides demonstrate that the relationship between model size and sensitivity is not linear or guaranteed.

As we see, even in the case of Llama-2, a 13B model is not guaranteed to always have lesser prompt sensitivity than a 7B model. We can thus infer that an increase in parameter count does not necessarily decrease prompt sensitivity. This directly supports option (d).

QUESTION 6: [1 mark]

In training a reward model with pairwise preferences $(x,\ y^{+}, y^{-})$, the Bradley-Terry style objective encourages:

a) Maximizing $r_\theta(x, y^-) - r_\theta(x, y^+)$
b) Minimizing the entropy of the policy
c) Maximizing $\log \sigma\left(r_\theta(x, y^+) - r_\theta(x, y^-)\right)$
d) Setting $r_\theta(x, y)$ equal to the log-probability under π_{ref}

Correct Answer: c

Explanation:

The goal of training a reward model (RM) is to teach it to assign a higher score (rθ) to the preferred response (y+) than to the rejected response (y−).

1. The **Bradley-Terry (BT) model** defines the probability that y+ is preferred as a function of the *difference* in their scores, passed through a sigmoid function (σ)
2. To train the RM, we use **Maximum Likelihood Estimation**, which aims to find the parameters θ that maximize the log-probability of the observed human preferences in our dataset.
3. This directly leads to the objective function: max ∑ log σ(r(x,y+)−r(x,y−)). This is exactly option (c).

Option (a) would do the opposite (prefer y−). Options (b) and (d) relate to the *policy optimization* phase, not the *reward model training* phase.

QUESTION 7: [1 mark]

Which of the following are recommended while performing REINFORCE-style policy optimization?

a) Use the log-derivative trick to obtain an unbiased gradient estimator.
b) Weight token-level log-probs by the advantage function to reduce variance.
c) Use importance weights and clip them when sampling from a fixed policy.
d) Avoid any clipping to preserve gradient magnitude.

Correct Answers: a, b, c

Explanation:

(a) **True:** The **log-derivative trick** is the core mathematical technique used to rewrite the policy gradient objective into an expectation, which allows us to approximate the gradient using samples from the policy .
(b) **True:** Standard REINFORCE has high variance. To reduce this, the gradient is weighted by the **advantage function** (At=Qt−Vt) instead of the full cumulative reward. This measures *how much better* an action was than the average, leading to more stable training .
(c) **True:** To improve sample efficiency, PPO (a REINFORCE-style algorithm) uses **importance weights** to allow for multiple gradient updates using

samples from an *old* policy . To ensure stability, these importance weights are **clipped** .

(d) **False:** This is incorrect. Clipping importance weights is a crucial part of PPO to prevent large, unstable gradient updates.

QUESTION 8: [1 mark]

Which method combines reward maximization and minimizing KL divergence?

a) REINFORCE
b) Monte Carlo Approximation
c) Proximal Policy Optimization
d) Constitutional AI

Correct Answer: c

Explanation:

Proximal Policy Optimization (PPO) is the specific algorithm used to optimize the combined objective . It uses a clipped surrogate objective that approximates this KL-constrained function, effectively balancing reward-seeking with policy stability.

□ (a) REINFORCE is the general algorithm family, but PPO is the specific method that formally incorporates the KL constraint.

□ (b) Monte Carlo Approximation is a *technique* used within these algorithms, not the method itself.

□ (d) Constitutional AI is a method for *generating preference data* to train the reward model , not the policy optimization algorithm.

QUESTION 9: [1 mark]

Which of the following is the reason for performing alignment beyond instruction tuning in LLMs?

a) Instruction tuning guarantees safety on harmful queries.
b) Alignment can prevent outputs that a model might otherwise deem correct, but humans find unacceptable.
c) Alignment is only needed for small models.
d) Instruction tuning already optimizes a human preference model.

Correct Answer: b

Explanation:

Instruction tuning might produce both a helpful answer *and* a harmful one ("You should stop eating entirely..."). Instruction tuning *cannot* reliably prevent the harmful output.

Alignment (like RLHF) is the next step, which uses human preference data to fine-tune the model. Its specific purpose is to prevent certain outputs that the model assumes to be correct, but humans consider wrong (or harmful/unacceptable).

QUESTION 10: [1 mark]

Let π_θ be the probability of choosing token a_t in state s_t assigned by the current policy being optimized, π_k be that by the old/reference policy and $\in > 0$ be the clip parameter. When the token-level advantage A_t is positive, PPO-CLIP maximizes which of the following expression at step t?

a) $\max\left(\frac{\pi_\theta}{\pi_k}\ ,\ 1-\in\right) A_t$
b) $\max\left(\frac{\pi_k}{\pi_\theta}\ ,\ 1-\in\right) A_t$
c) $\min\left(\frac{\pi_k}{\pi_\theta}\ ,\ 1+\in\right) A_t$
d) $\min\left(\frac{\pi_\theta}{\pi_k}\ ,\ 1+\in\right) A_t$

Correct Answer: d

Explanation:

The PPO-CLIP objective is designed to prevent the new policy (π_θ) from moving too far from the old policy (π_k) in a single update.

- When the advantage A_t is **positive**, it means the action a_t was *good*, and we want to *increase* its probability. This means we want to increase the ratio $r_t = \frac{\pi_\theta}{\pi_k}$.
- However, to ensure stability, we "clip" this increase. We don't want the ratio r_t to go higher than (1+ϵ).
- Therefore, the algorithm takes the **minimum** of the *actual* ratio (r_t) and the *clipped* ratio ((1+ϵ)). This clipped ratio is then multiplied by the positive advantage A_t.

The objective to maximize is: $\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_k(a_t|s_t)}\ ,\ 1+\in\right) A_t\,(s_t, a_t)$. This corresponds exactly to option (d).
