CS 747 (Spring 2025) Week 11 Test (Batch 1)

5.35 p.m. – 6.00 p.m., April 7, 2025, LA 001

Name: _____ Roll number: _____

Note. There are two questions, one on each page. Provide your answer to each question in the space given below it. Draw a line (either vertical or horizontal) and do all your rough work on one side of it.

Question 1. What is the connection between the "Policy Gradient Theorem" and the "REINFORCE" algorithm? You do not have to state the theorem or specify the algorithm, only comment on the relationship between the two. [1 mark]

Answer 1. The policy gradient theorem provides an analytical expression for the gradient of the policy value "$J(\theta)$" in terms of the gradients of policy's state-action probabilities "$\pi_\theta(s,a)$". Both gradients are with respect to the policy parameters θ. Although several quantities in the gradient expression are unknown to a learner, they can be substituted with unbiased estimates which are available from sample trajectories generated by following π_θ. The REINFORCE algorithm is precisely a procedure that performs *stochastic* gradient ascent in the θ-space to optimise J.

Question 2. What assumption is made on the class of policies over which REINFORCE optimises? What is the theoretical guarantee provided for REINFORCE (under suitable assumptions)? [2 marks]

Answer 2.

REINFORCE requires that policy's state-action probabilities be differentiable with respect to policy parameters: that is, the gradient vector $\nabla_\theta \pi(s, a)$ be well-defined for all state-action pairs (s, a). In turn this requirement dictates that policies be stochastic (but note that not all stochastic policies are differentiable).

Like other stochastic gradient ascent methods, REINFORCE has the guarantee of converging to a local optimum if the learning rate is annealed suitably.

CS 747 (Spring 2025) Week 11 Test (Batch 2)

6.15 p.m. – 6.40 p.m., April 7, 2025, LA 001

Name: _____ Roll number: _____

Note. There is one question in this test. You can use the space on both pages for your answer. Draw a line (either vertical or horizontal) and do all your rough work on one side of it.

Question 1. For an episodic MDP M in which the start state is s_0, a designer decides to apply policy search over a space of policies. Each policy is parameterised by a d-dimensional real-valued vector for some $d \geq 1$; thus the space of policies is $\mathbb{R}^d$. Pick either "grid search" or "hill climbing" as an example of a policy search method, and explain how it is implemented on M. What is the effect of the dimensionality d on the efficacy of the policy search method you have selected? [3 marks]

Answer 1.

A basic primitive required by policy search is that of *evaluation*: to estimate $V^{\pi_\theta}(s_0)$ for any policy parameter vector θ. This step is usually performed by generating multiple trajectories by starting at s_0 and taking actions according to π_θ, and averaging the cumulative return along these trajectories. The estimate is unbiased; its variance decreases as more trajectories are generated for the average.

In grid search, a finite, "regularly-spaced" subset of $\mathbb{R}^d$ is fixed beforehand. For example, in two dimensions, the subset might contain all vectors whose first dimension is in $\{-1, -0.5, 0, 0.5, 1\}$ and whose second dimension is in $\{-10, -9, -8, \ldots, 8, 9, 10\}$. Every parameter vector from this finite subset is evaluated as above, and the one with the largest evaluation is returned.

Hill-climbing, unlike grid search, is a dynamic procedure that begins at some initial parameter vector, θ_0, which may be viewed as an "anchor". First θ_0 is evaluated as given above. Then, a finite set of "neighbours" of θ_0 are also evaluated. If the neighbour with the largest evaluation also exceeds that of θ_0, this neighbour is made the new anchor θ_1, and the process continued (that is, the neighbours of θ_1 are evaluated and so on). Hill-climbing terminates when the current anchor has no improving neighbours, in which case we may conclude that the anchor is close to a local optimum. An important aspect of the algorithm is the definition of "neighbours"—a common approach is to independently perturb the dimensions of a parameter vector (by adding zero-mean noise) to generate neighbours.

Policy search methods tend to work well when the dimension d is small. Since the number of policies evaluated by grid search is exponential in d, it is rare to perform grid search beyond 5–10 dimensions. Hill-climbing can typically work up to a few hundreds of dimensions, but of course its efficacy also depends on the task and the policy template used.