Lecture 5 - *Stochastic Approximation*

OBJECTIVE: This lecture introduces stochastic approximation (SA) concepts for designing approximate RL algorithms and proving their convergence. Instead of visiting each state as in policy and value iteration, we will only consider simulated trajectories of states. This modification enables us to tackle more realistic environments, with larger state spaces and no need to know the transition model or reward function. The algorithms presented in this lecture are also applicable to many other machine learning domains including neural network training, on-line learning and parameter estimation in conditional and Markov random fields.

◇ BANDITS: A MOTIVATING EXAMPLE

Let's consider our K-armed bandits again:

- **[States]**: $\mathbf{x}_t^i$ is the state of bandit i. The entire state is $\mathbf{x}_t^{1:K} \in \mathcal{X}^1 \times \cdots \times \mathcal{X}^K$.
- **[Actions]**: $\mathbf{a}_t \in \mathcal{A}(x) = \{1, 2, \ldots, K\}$.
- **[Rewards]**: $r_t(\mathbf{x}_t^{1:K}, \mathbf{a}_t) = r_t(\mathbf{x}_t^i, \mathbf{a}_t = i) = r_t(i)$.
- **[Transition law]**: The selected bandit evolves according to: $p(\mathbf{x}_{t+1}^i | \mathbf{x}_t^i)$.

Let's assume that our goal is to compute the expected reward $Q_{t+1}(i)$ for bandit i given that it has been chosen $t+1$ times. Then we have:

$$Q_{t+1}(i) = \frac{1}{t+1} \sum_{k=1}^{t+1} r_k(i) = Q_t(i) + \frac{1}{t+1} \left[r_{t+1}(i) - Q_t(i) \right]$$

⋆ Proof:

Note that the new estimate is equal to the old estimate plus a weighted difference between the old estimate and the target. This type of update will appear over and over again.

Once some of the Q's are known we have to deal with the exploration/exploitation trade-off. For bandits this can be handled with Gittin's indices.

◇ COMPUTING FIXED POINTS WITH SA

Suppose we want to solve for $\boldsymbol{\theta}$ in the following stochastic fixed point equation:

$$\boldsymbol{\theta} = \mathbb{E}[f(\boldsymbol{\theta}, \mathbf{x})] = \int f(\boldsymbol{\theta}, \mathbf{x})\, p(\mathbf{x}|\boldsymbol{\theta})\, d\mathbf{x}$$

We can do so by adopting the following updates:

$$\boldsymbol{\theta}^{(t+1)} = (1 - \alpha^{(t)})\boldsymbol{\theta}^{(t)} + \alpha^{(t)} f(\boldsymbol{\theta}^{(t)}, \mathbf{x}^{(t)})$$

where $\alpha \in (0, 1)$ and $\mathbf{x}^{(t)} \sim p(\mathbf{x}|\boldsymbol{\theta})$.

⋆ Derivation:

The previous update can be re-written as a **Robbins-Monro** stochastic approximation:

$$\boldsymbol{\theta}^{(t+1)} = (1 - \alpha^{(t)})\boldsymbol{\theta}^{(t)} + \alpha^{(t)}\mathbb{E}[f(\boldsymbol{\theta}^{(t)}, \mathbf{x})] + \alpha^{(t)} w^{(t)}$$

where

- $\mathbb{E}[f(\boldsymbol{\theta}^{(t)}, \mathbf{x})]$ is the mean field.
- $w^{(t)} = \left\{ f(\boldsymbol{\theta}^{(t)}, \mathbf{x}^{(t)}) - \mathbb{E}[f(\boldsymbol{\theta}^{(t)}, \mathbf{x})] \right\}$ is the stochastic approximation error.

For convergence in a stationary regime, the learning rate must decay to zero. Yet, it cannot decay too quickly as this would prevent the algorithm from visiting areas in the state space far away from the initial assignments.

◇ EXAMPLE: STOCHASTIC POLICY EVALUATION

Let's consider our familiar fixed point $V = T_d V$:

$$V(i) = \sum_{j=1}^{n} p(j|i, d(i)) \left[r(i, d(i), j) + V(j) \right]$$

We can avoid full evaluation of the huge expectation in practice by introducing the following SA known as $TD(0)$:

⋆ Derivation:

◇ EXAMPLE: STOCHASTIC OPTIMIZATION

Suppose we want to minimize a function $L(\boldsymbol{\theta}) = \mathbb{E}[Q(\boldsymbol{\theta}, \mathbf{x})]$. That is, we want to solve the problem:

$$\min_{\boldsymbol{\theta}} \int Q(\boldsymbol{\theta}, \mathbf{x})\ p(\mathbf{x}|\boldsymbol{\theta})\ d\mathbf{x}$$

We can do this by taking the derivative of $L(\boldsymbol{\theta})$ and equating it to zero. We can do this as follows:

We can use this derivation in conjunction with SA to obtain a stochastic gradient descent algorithm:

IMPORTANT: The algorithm is not stochastic because we are adding noise to the gradient. It is stochastic because the reward function is unknown.

Alternatives to this simple approach include **stochastic conjugate gradients** and **stochastic meta descent** (SMD). If the derivative is not available, we can conduct **implicit differentiation**, use finite differences or the **Simultaneous Perturbation Method**.

◇ LAWS OF LARGE NUMBERS

To study the asymptotic behavior of RL algorithms, we apply the strong law of large numbers (SLLN):

Theorem 5 *Let* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ *be a sequence of i.i.d. random variables with common finite mean* $\mathbb{E}[\mathbf{x}_i]$*. Then:*

$$\boldsymbol{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i \xrightarrow{a.s.} \boldsymbol{\theta}^\star = \mathbb{E}[\mathbf{x}_i] \;\; w.p.\; 1$$

Here $a.s.$ stands for **almost sure convergence** or convergence with probability ($w.p.$) 1. That is, convergence occurs for the sets with non-null probability.

The SLLN is a stronger statement than the weak law of large numbers (WLLN) proved in the homework.

We won't prove the SLLN as this is a simple exercise that appears in most probability books, but we will present the two Lemmas that are used to prove it as these can be used

go gain intuition into how SA works.

Lemma 2 Borel Cantelli *Consider the sequence of "error" events* $E_k = \{\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^\star\| \geq \varepsilon\}$ *for* $k \geq 1$ *and any positive* ε*. Then the event* E_k *does not happen infinitely often (i.o.):*

$$\sum_k P(E_k) < \infty \qquad \Longrightarrow \qquad P(E_k \; i.o.) = 0$$

Or, in other words, $\boldsymbol{\theta}_k$ *converges almost surely to* $\boldsymbol{\theta}^\star$*.*

Thus to prove SLLN's, we first show that the sum of probabilities is upper-bounded. This is typically done using Markov's inequality.

Lemma 3 Markov's inequality *Let $f(\boldsymbol{\theta})$ be positive. Then $P(f(\boldsymbol{\theta}) \geq \varepsilon) \leq \frac{\mathbb{E}[f(\boldsymbol{\theta})]}{\varepsilon}$.*

When $f(\boldsymbol{\theta}) = \|\boldsymbol{\theta}_k - \boldsymbol{\theta}\|$, we get Chebyshev's inequality (to be proven in the homework), which enable us to upperbound the probabilities of error events with the variance of the error.

Armed with these lemmas, we can attack the problem of choosing the learning rate of SA algorithms:

$$\boldsymbol{\theta}^{(t+1)} = (1 - \alpha^{(t)})\boldsymbol{\theta}^{(t)} + \alpha^{(t)} f(\boldsymbol{\theta}^{(t)}, \mathbf{x}^{(t)})$$

Summing both sides over t, we have

◇ RECURSIVE SLLN

As in the bandit problem, we can easily derive a recursive estimator of $\boldsymbol{\theta}_k$.