687 2017-09-19

Note Title 9/19/2017

Pseudocode

```
for episode = 1, 2, ...
    s ~ d_0
    for t = 0, 1, 2, ...
        a = agent.getAction(s)
        s' ~ P(s, a, ·)
        r = R(s, a, s')  # assuming deterministic
        agent.train(s, a, r, s')
        if s' == s_∞ break
        s = s'
```

Black Box Optimization for Policy Search

- Simple Agent
- Ignore MDP structure
- Phase this as an optimization problem

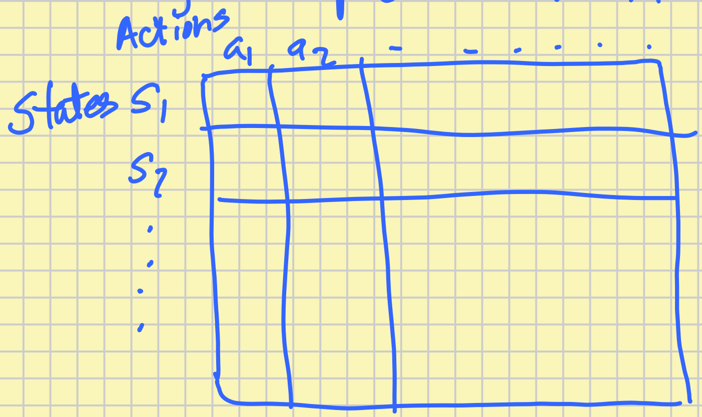
$$\arg\max_{\pi} J(\pi)$$

- Can access only an estimate $\hat{J}(\pi)$, e.g. evaluation of π on a sample of N episodes, average the discounted reward

$$\hat{J}(\pi) = \frac{1}{N}\sum_{i=1}^{N} G_i, \quad G_i \triangleq \sum_{t=0}^{\infty} \gamma^t R_t^i$$

← ith episode

A way to represent π: As a table p



- # of these parameters is $|\mathcal{S}| \times |\mathcal{A}|$
- valid if each row sums to 1 and all entries ≥ 0

Want a function optimization method that obeys this constraint

$\Downarrow$

Tabular Softmax Action Selection

- Store π as an $|\mathcal{S}| \times |\mathcal{A}|$ matrix
- No constraints on the matrix
- Increasing $p(s,a)$ increases $\pi(s,a)$

$\Downarrow$

$$\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{a'} e^{p(s,a')}}$$

This guarantees $\left(\sum_a \pi(s,a)\right) = 1$ and $\pi(s,a) \geq 0$

Conventional to call these parameters θ (a vector formed from p)

A couple of alternatives

Modeling Diabetes: Policy: Before a meal inject $\frac{BG - BG_{target}}{\theta_1} + \frac{\text{size of meal}}{\theta_2}$

BG = blood glucose (measured by pump)

BG_{target} = level the doctor believes is good

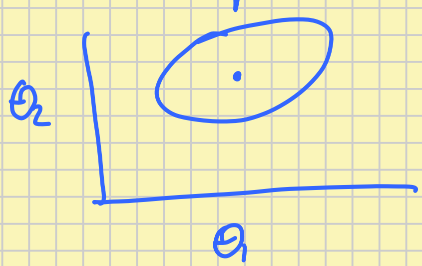
size of meal = estimated by patient

θ_1, θ_2 : parameters we may try to learn

Cross-Entropy Method (was used for Tetris) → simple but powerful, general

1. Generate a random data sample according to a parameterized mechanism
2. Update parameters based on the samples to do better at the next iteration

- Mechanism = distribution over policies
- Random sample = histories generated by sampled policies



2-D Gaussian

- Draw (θ_1, θ_2) pairs
- Compute $\hat{J}(\pi)$, i.e., $\hat{J}(\theta)$
- Move mean toward higher $\hat{J}(\pi)$
  - Can do weighted sum - regression method
  - K_e : elite population $\leq K$ population
    best - update mean w/ mean over K_e
  - Do likewise with covariance (covariance over K_e)

Pseudo-Code for Cross-Entropy Method (CEM)

Input: θ - mean parameter vector
Σ - covariance, initially σI
K - population
K_e - elite population
N - # of episodes

Why called this? Has to do with how to go from one θ distribution to the next while minimizing a certain loss function.

for $k = 1:K$
$\theta_k \sim N(\theta, \Sigma)$
$\hat{J}_k = \text{evaluate}(\theta_k, N)$

$\underset{k=1,K}{\text{sort}}\ (\theta_k, \hat{J}_k, \text{descending on } \hat{J}_k)$

$$\theta \leftarrow \frac{1}{K_e} \sum_{k=1}^{K_e} \theta_k$$ (top K_e θ_k according to quality estimates $\hat{J}_k$)

$$\Sigma \leftarrow \frac{1}{K_e} \sum_{k=1}^{K_e} (\theta_k - \theta)(\theta_k - \theta)^T$$ ← can be numerically unstable so....

$$\Sigma \leftarrow \frac{1}{K_e + \epsilon}\left(\epsilon I + \sum_{k=1}^{K_e} (\theta_k - \theta)(\theta_k - \theta)^T\right)$$

Demo of project things in Visual Studio