Pseudo code

for episode = 1, 2, ...
    s_0
    for t = 0, 1, 2, ...
        a = agent.getAction(s)
        s' ~ P(s, a, s') # assuming deterministic agent.
        r = R(s, a, s')
        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 \( J(\pi) \)
- Can access only an estimate \( \hat{J}(\pi) \),
  e.g. evaluation of \( \pi \) on a sample of \( N \) episodes,
  average the \( i \)th discounted reward
  \( \hat{J}(\pi) = \frac{1}{N} \sum_{i=1}^{N} \gamma \sum_{t=0}^{i} R_t \)
A way to represent $\pi$: As a table $p$

<table>
<thead>
<tr>
<th>States $S$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>...</th>
</tr>
</thead>
</table>

- # of these parameters is $|S| \times |A|$
- Valid if each row sums to 1 and all entries $\geq 0$

Want a function optimization method that obeys this constraint

Tabular Softmax Action Selection

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

$$\pi^*(s,a) = \frac{e^{p(s,a)}}{\sum_a e^{p(s,a)}}$$

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

Conventional to call these parameters $\Theta$ (a vector formed from $p$)
A couple of alternatives:

Modeling Diabetes: Policy: Before a meal inject \( BG - B_{G\text{target}} + \frac{\text{size of meal}}{\theta_1} \theta_2 \) where:
- \( BG \): blood glucose (measured by pump)
- \( B_{G\text{target}} \): level the doctor believes is good
- \( \text{size of meal} \): estimated by patient
- \( \theta_1, \theta_2 \): 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 \((\theta_1, \theta_2)\) pairs
- Compute \( \hat{J}(\bar{x}) \), i.e., \( \hat{J}(\theta) \)
- Move mean toward higher \( \hat{J}(\bar{x}) \)
  - Can do weighted sum-regression method
  - \( K_e \): elite population \( \geq K \) 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: \( \Theta \) - mean parameter vector
\( \Sigma \) - covariance, initially \( \sigma I \)
\( K \) - population
\( K_e \) - elite population
\( N \) - # of episodes

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

sort \( \{(\Theta_k, \hat{J}_k)\} \) descending on \( \hat{J}_k \)

\( \Theta \leftarrow \frac{1}{K_e} \sum_{k=1}^{K_e} \Theta_k \) (for \( k \leq K_e \) according to quality estimates \( \hat{J}_k \))

\( \Sigma \leftarrow \frac{1}{K_e} \sum_{k=1}^{K_e} (\Theta_k - \Theta)(\Theta_k - \Theta)^T \) \( \leftarrow \) can be numerically unstable so...

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

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

Demo of project
Things in Visual Studio