Some Bandit Algortithms
Arne Huckemann

When updating the estimated mean $\hat{\mu}_n$ at n time memory efficient, i.e.

$$\hat{\mu}_n = (\hat{\mu}_{n-1}(n-1) + X_n) \cdot \frac{1}{n}$$

we can update the estimated variance $\hat{\sigma}_n^2$ at time n also memory efficient

$$\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{\mu}_i)^2 = \frac{(n-2)\hat{\sigma}_{n-1}^2}{n-1} + \frac{(X_n - \mu_n)^2}{n-1}.$$

We will now look at the performance of the these Algorithms. The time horizon for the multi armed bandit will be n. We will repeat this N times, such that we can estimate the performance of an algorithm better. We will mainly analyse the following objects

- Regret $R_n(\pi)$
- The correct action rate, i.e. the culmulitave average, we play the best arm, i.e. at time t its

$$\frac{1}{t}\sum_{i=1}^{t}\mathbf{1}_{a_i=a_*}$$

Further, we will estimate the expected Regret at each time step and the pointwise variance, i.e. for every time step. Finally, we will look at the expected Q value, at each time step and the expected pointwise probability of choosing an arm. We will also look at the pointwise variance of these objects.
We chose in the following $n = 200$ and $N = 1000$ and 10 gaussians arms with means $1, 2, 3, 4, 6, 7, 10, 11, 11.5, 12$. We chose the best and second best very close, such that the best is harder to find.
The thoery we did, was for $\sigma-$Subgaussians bandits. Because the normal distribution is part of this class, the thoery applies for this application.

Exploration and exploitation trade off:
As we explore longer, our estimates for the Q-Values get better, but Regret is higher. If we exploit earlier, then the chances of choosing a suboptimal arm, is increased. This is formalized in the Failure probability and the regret decomposition Lemma.

Definition 0.1: Failure Probability

We define the probability that an suboptimal arm is chosen at time t as

$$\tau_t(\pi) := \mathbb{P}_\pi(Q_* \neq Q_{A_t})$$

which we call the failure probability.

Lemma 0.2: Regret Decompostion Lemma

We can rewrite the regret as

$$R_n(\pi) = \sum_{t=1}^{n} \sum_{a \in \mathcal{A}} \Delta_a \mathbb{P}_\pi(A_t = a)$$

and get the bounds

$$R_n(\pi) \leq \max_{a \in \mathcal{A}} \Delta_a \sum_{t=1}^{n} \tau_t(\pi) \quad \text{and} \quad R_n(\pi) \geq \min_{a \neq a_*} \Delta_a \sum_{t=1}^{n} \tau_t(\pi)$$

1 Explore then Commit Algorithm

For the explore then commit algorithm is given by

Algorithm 1: Explore then Commit

Data: m, n, bandit model ν ;
Set $\hat{Q}(0) \equiv 0$;
while $t \leq n$ **do**
  Set $A_t = \begin{cases} a_{t mod K+1} & t \leq mK \\ argmax_a \hat{Q}_a(mK) & t > mK \end{cases}$;
  Set Obtain reward X_t by playing arm A_t ;
Output: Actions $A_1, ..., A_n$ and rewards $X_1, ..., X_n$

If we look only look at σ-Subgaussian Bandits then the regret can by bounded by the following.

Lemma 1.1: Regret Bound for explore then commit

For a σ-Subgaussian bandit model ν and the learning strategy π that follows the explore then commit algorithm, i.e. we play m times every arm and then $n - mK$ times the best arm, it holds

$$R_n(\pi) \leq m \sum_{a \in \mathcal{A}} \Delta_a + (n - mK) \sum_{a \in \mathcal{A}} \Delta_a e^{\frac{\Delta_a^2 m}{4\sigma^2}}$$

Therefore, we have linear regret for arbitrary choices of the exploration m. If we choose the exploration as follows, we can get in the special case of two armed bandits regret of logarithmic order.

Corollary 1.2

For a two armed bandit the upper bound for the regret of the explore then commit algorithm is minimized by

$$m = \max\{1, \lceil \frac{4\sigma^2}{\Delta} \log\left(\frac{n\Delta^2}{4\sigma^2}\right) \rceil\}.$$

Using this choice of exploration, we get

Corollary 1.3

For a two armed bandit with optimized exploration the upper bound for the regret of the explore then commit algorithm is given by

$$R_n(\pi) = \min\left\{n\Delta, \Delta + \frac{4\sigma^2}{\Delta}(1 + \max\{0, \log(\frac{n\Delta^2}{4\sigma^2})\})\right\}$$

and for $n \geq \frac{4}{\Delta^2}$ we have the model dependend regret bound

$$R_n(\pi) \leq C_\Delta + \frac{\log(n)}{\Delta}$$

We let the algorithm run, with beforehand optimization of the exploration lenght, i.e. we cheated. In the optimization we found the lenght of 3 exploration steps (which does not make a lot of sense, as we have 10 arms).

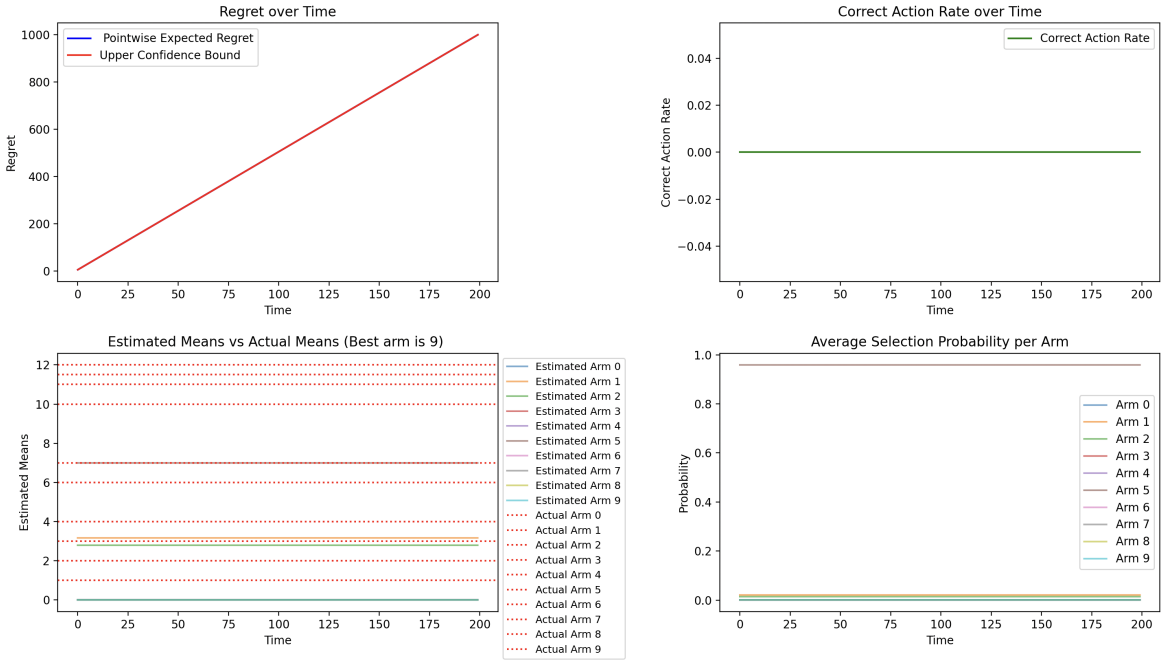

Figure 0.1: ETC

As we can see we have linear regret, the best arm is never tried and thus never found.

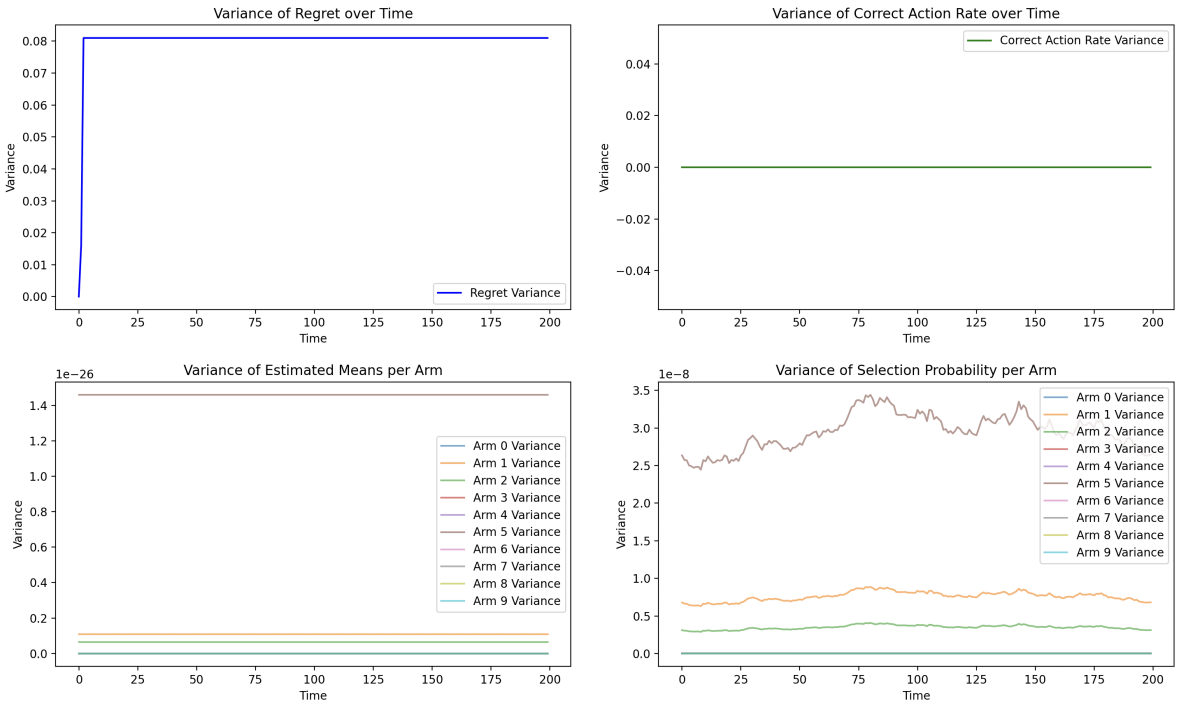

Figure 0.2: ETC Variances

This figure is mostly uninteresting. Only interesting is that the pointwise variance of the regret increases, as we go further in time.

2 ϵ-Greedy

Algorithm 2: ε-greedy bandit algorithm

Data : bandit model ν, exploration rate $\varepsilon \in (0,1)$, vector $\hat{Q}$, n
Initialise $T_a = 0$ for all a;
while $t \leq n$ **do**
  Sample $U \sim \mathcal{U}([0,1])$;
  if $U < \varepsilon$ **then**
    [exploration part]
    Uniformly choose an arm A_t;
  else
    [greedy part]
    Set $A_t = \arg\max_a \hat{Q}_a$;
  Obtain reward X_t by playing arm A_t;
  Set $T_{A_t} = T_{A_t} + 1$;
  Set $\hat{Q}_{A_t} = \hat{Q}_{A_t} + \frac{1}{T_{A_t}}(X_t - \hat{Q}_{A_t})$;
Result: actions $A_1, \ldots, A_n$ and rewards $X_1, \ldots, X_n$

2.1 Constant step size

Lemma 2.1

Let π be a learning strategy, that first, explores every arm once and then plays ϵ-greedy, for $\epsilon \in (0, 1)$, then

$$\lim_{n\to\infty} \frac{R_n(\pi)}{n} = \frac{\epsilon}{K} \sum_{a\in\mathcal{A}} \Delta_a$$

We should expect linear regret. We set $\epsilon = 0.2$.

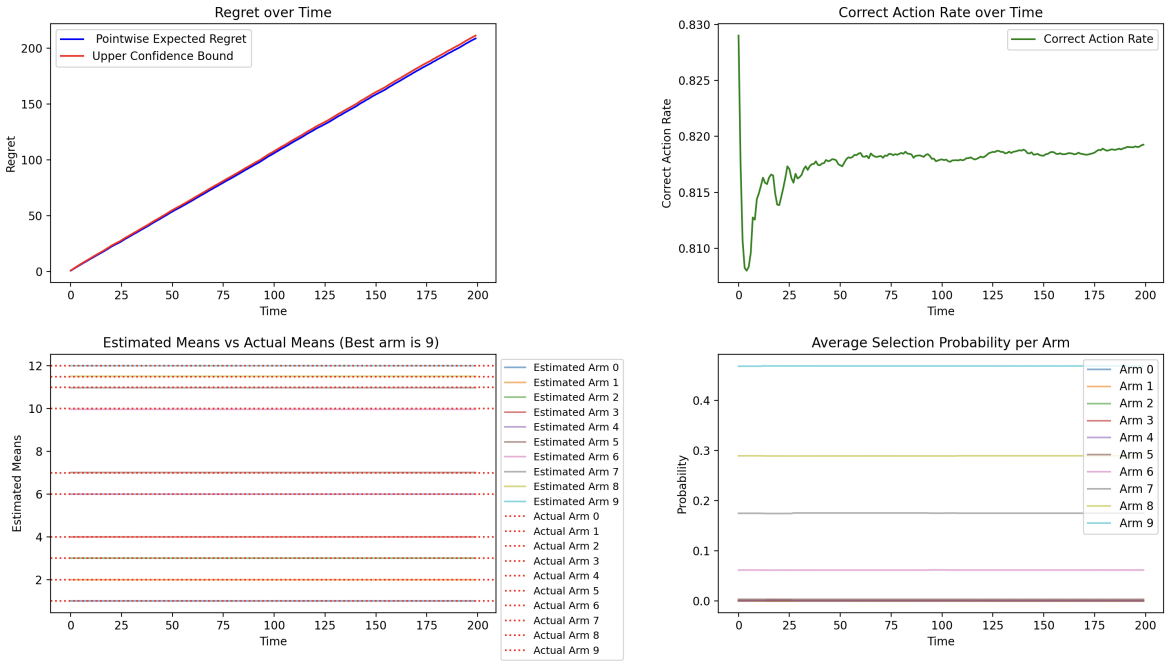

Figure 0.3: Constatn Epsilon Greedy

What is positive is, that the best arm is found and the probability of playing it, is quickly very high, tough it never reaches 1, as we selected $\epsilon = 0.2$, which is very visible in the average correct action rate. Further, all arms have their estimates, close to the true Q-Values.
Its negative, that the regret grows linearly. This is the cost of exploration. The upper confidence bound is very close to the true value, i.e. small variance in the pointswise estimation of the mean.

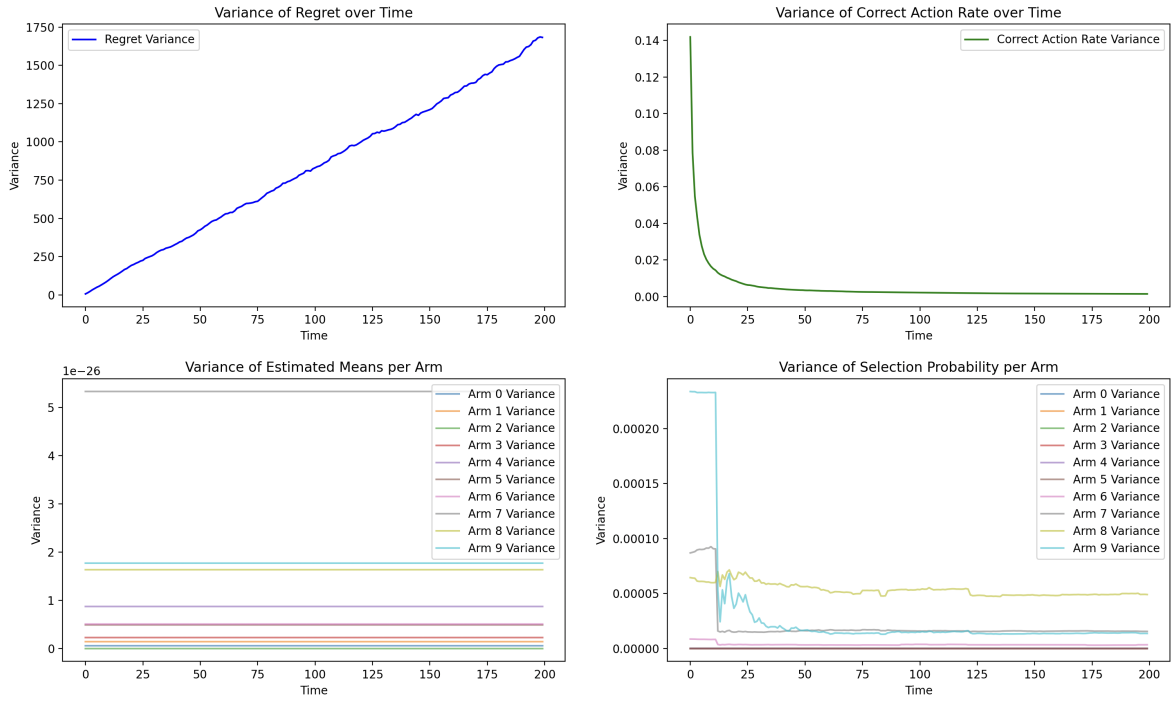

Figure 0.4: Constant Epsilon Greedy Variances

This shows, what is more or less already clear in the above. The variance of the average regret estimation increases, as we go through time, the variance decreases in the estimated average correct action rate. The variances in playing different arms, also decreases in time.

2.2 Decreasing Step size

Theorem 2.2: Explore then ϵ-greedy with decreasing ϵ

Suppose that all arms take values in $[0,1]$ and define $d < \min_{a \neq a^*} \Delta_a$ and $C > \max\{5d^2, 2\}$. Then the ϵ-greedy with decreasing epsilon defined as $\epsilon_t := \max\{1, \frac{CK}{d^2 t}\}$ satisfies

$$\limsup_{n \to \infty} \tau_n(\pi) n \leq \frac{(K-1)C}{d^2},$$

i.e. $\tau_n(\pi) \in \mathcal{O}(\frac{1}{n})$.

Here, we should expect logarithmic regret, as if the failure probability is of order $\mathcal{O}(1/n)$, then the regret should be of order $\mathcal{O}(\log(n))$.
We start with an $\epsilon_1 = 1$ and in each time step we decrease it by 0.05. For other choices, we had very bad performances.

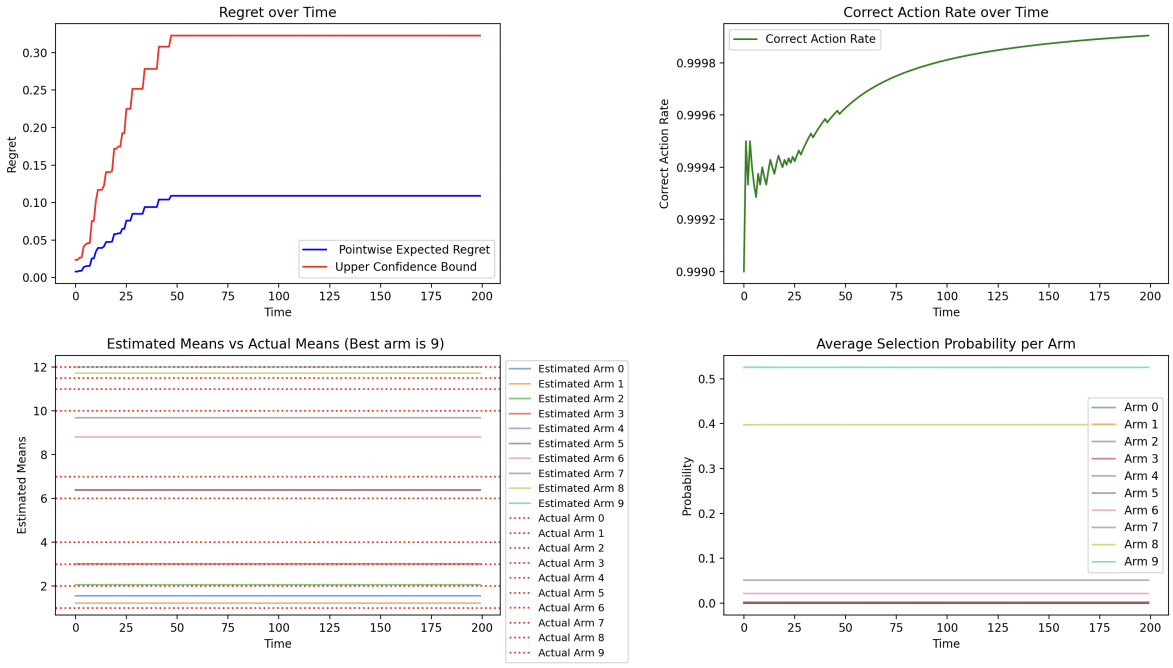

Figure 0.5: Decreasing Epsilon Greedy

This is up to now the best performance. What is positice is, that the regret grows slower than linear regret and does not increase, after ϵ was decreased sufficiently. The average corrected avtion ratie also goes to 1, which is what we want.
What is interesting, is that the good arms have very accurate estimates and the bad ones, not so accurate ones.
What is negative is not visible in the plots. To get this performance, a few trials with different ϵ_n had to be tested, which had very bad performances, even worse ones as the constant $\epsilon-$Greedy algorithms.

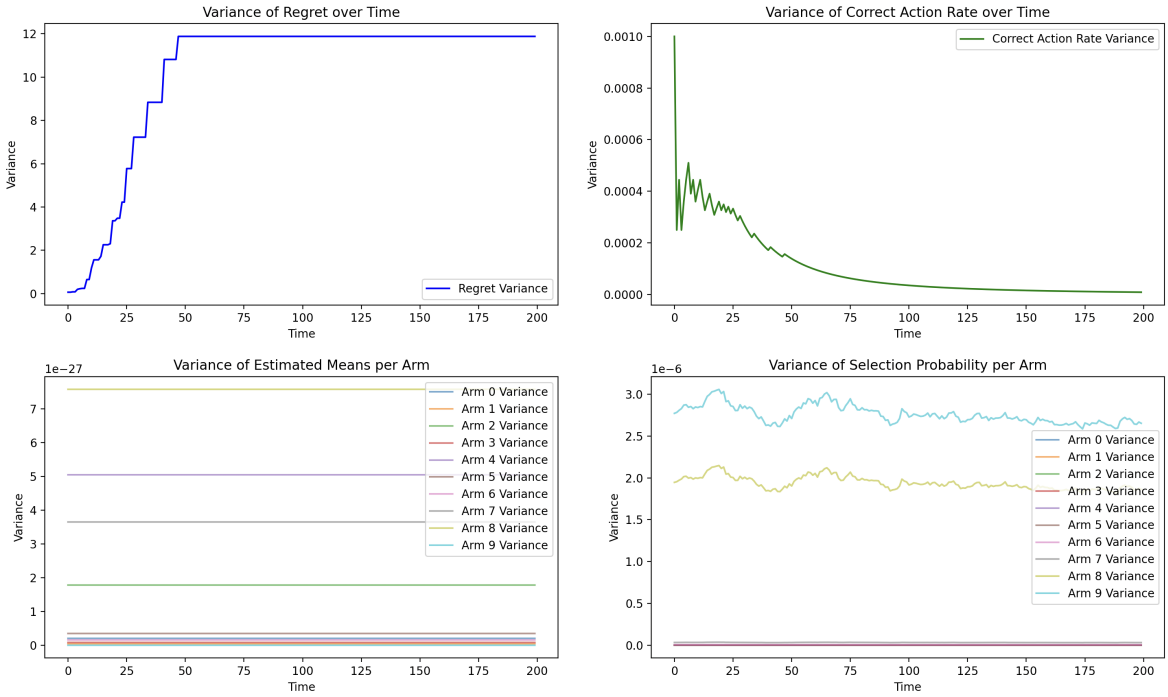

Figure 0.6: Decreasing Epsilon Greedy Variances

What is interesting again, is that the plot of the variance of the regret estimate for each time step, has the same form as the plot of its expectation, i.e. after a specific time, the variance stays constant for the estimation. This suppositely comes from the fact that, when ϵ is so small, that we only exploit, either the best or second best arm, as they are close, and sometimes they best arm is found and sometimes it is not. We can also see this in the probability of selection an arm, as for the best and second best arm, they have the highest expectation and variances.

3 UCB Algorithm

Algorithm 3: UCB Algorithm

Data : bandit model ν, $\delta \in (0,1)$, vector $\hat{Q}$, n,vector $\hat{Q}$
Initialise $T_a = 0$ for all a;
while $t \leq n$ **do**
  $A_t = argmax_a UCB_a(t-1,\delta)$;
  Obtain reward X_t by playing A_t;
  $T_{A_t} += 1$;
  $\hat{Q}_{A_t}(t) = \hat{Q}_{A_t}(t) + \frac{1}{T_{A_t}}(X_t - \hat{Q}_{A_t}(t))$
Result: actions $A_1, \ldots, A_n$ and rewards $X_1, \ldots, X_n$

We did not derive any theoretical results on the regret. We chose $\delta = 0.1$.

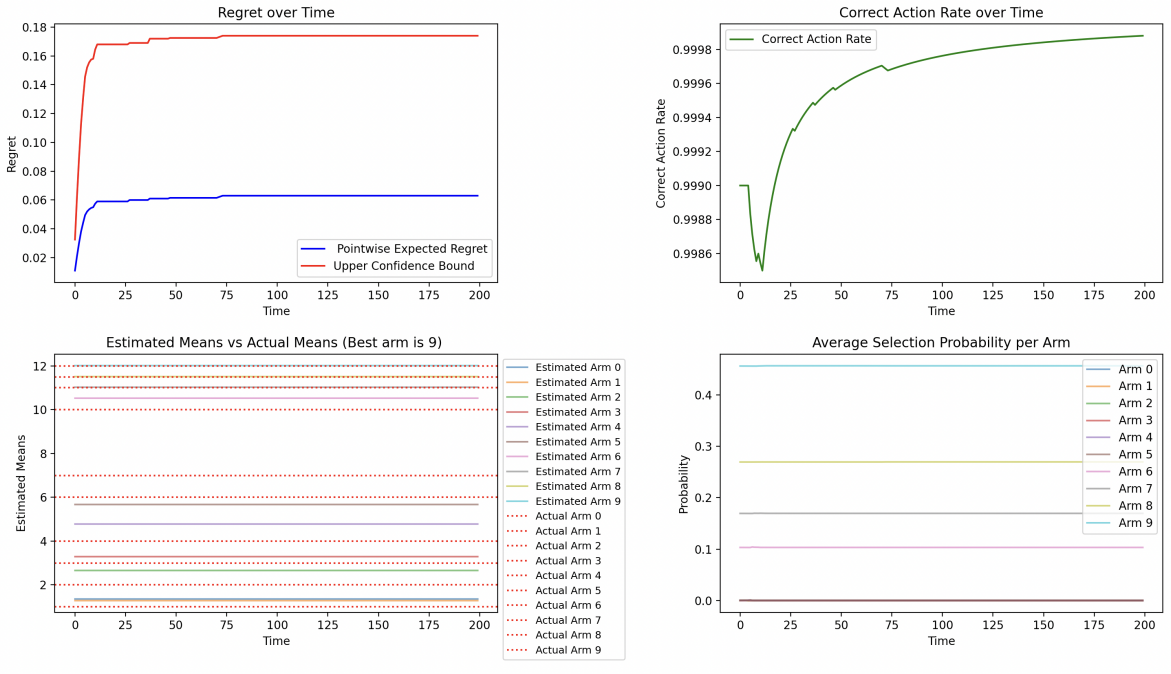

Figure 0.7: UCB

In absolute values, the regret of UCB is lower, than the one of ϵ-Greedy with decreasing ϵ and it has the same shape. Further, the variance of estimation is also lower. I.e. we can definetely say, that UCB is better. The Correct action rate converges to one as well.

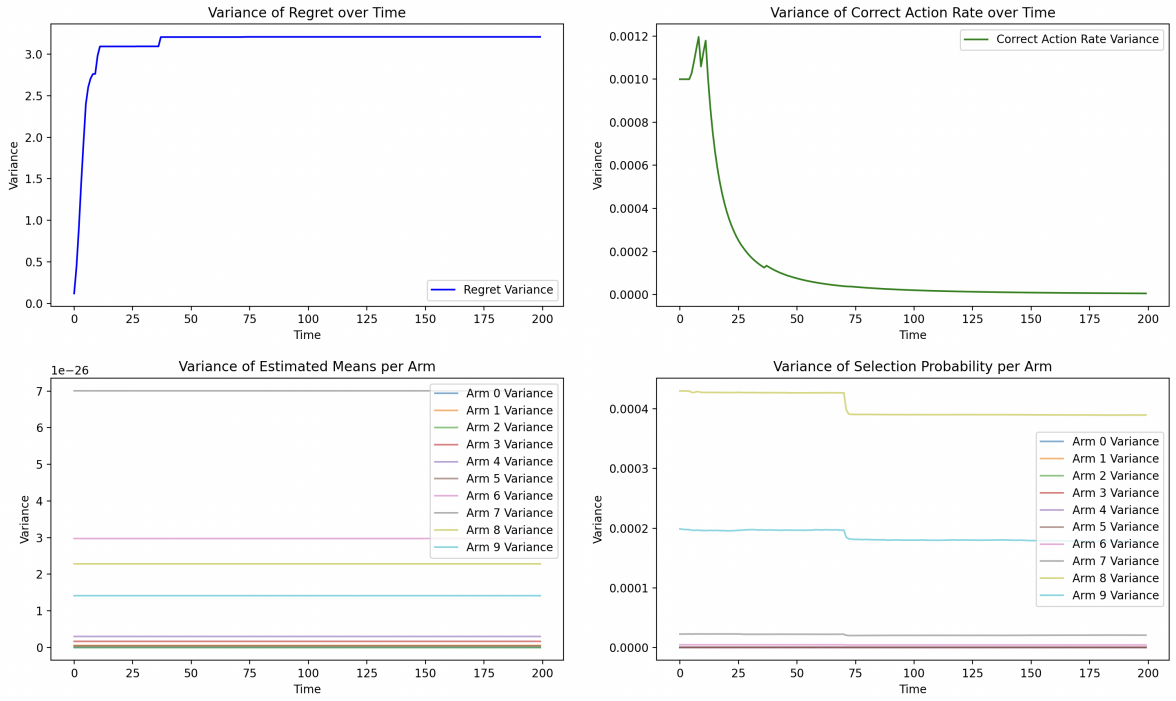

Figure 0.8: UCB Variances

4 Purely Greedy Algorithm

Algorithm 4: Purely greedy bandit algorithm

Data : bandit model ν, vector $\hat{Q}$, n
Initialise $T_a = 0$ for all a;
while $t \leq n$ **do**
  Set $A_t = \arg\max_a \hat{Q}_a$;
  Obtain reward X_t by playing arm A_t;
  Set $T_{A_t} = T_{A_t} + 1$;
  Set $\hat{Q}_{A_t} = \hat{Q}_{A_t} + \frac{1}{T_{A_t}}(X_t - \hat{Q}_{A_t})$;
Result: actions $A_1, \ldots, A_n$ and rewards $X_1, \ldots, X_n$

The key in the Greedy algorithm, is how we initialize the vector $\hat{Q}(0)$. In the algorithm, we chose it as a very large value. But in general, it is never clear what is large enough.
We did not derive any theoretical results, as it completely depends on the initialization.

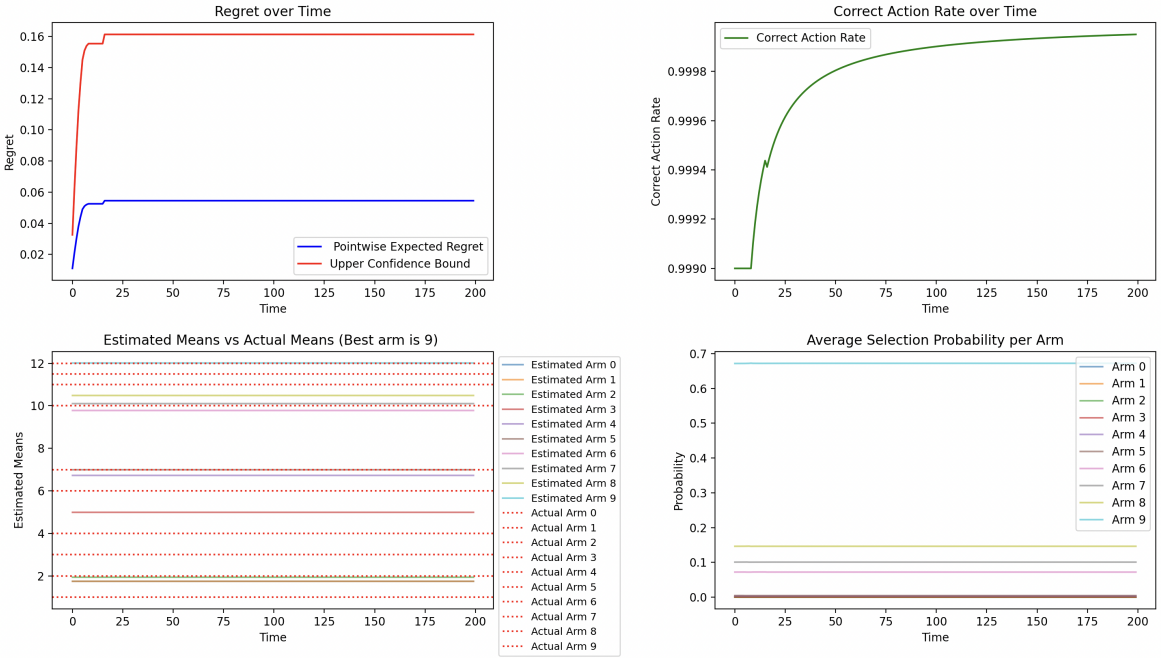

Figure 0.9: Greedy

We see completely the same performance as the UCB algorithm in all metrics. But it has to be noted that we cheated, as we knew what a "large" initialization was of the Q-Values. Thus, if the Q-Vales would not have been initialized in a "not large enough" way, we would/could have linear regret and never found the best arm.

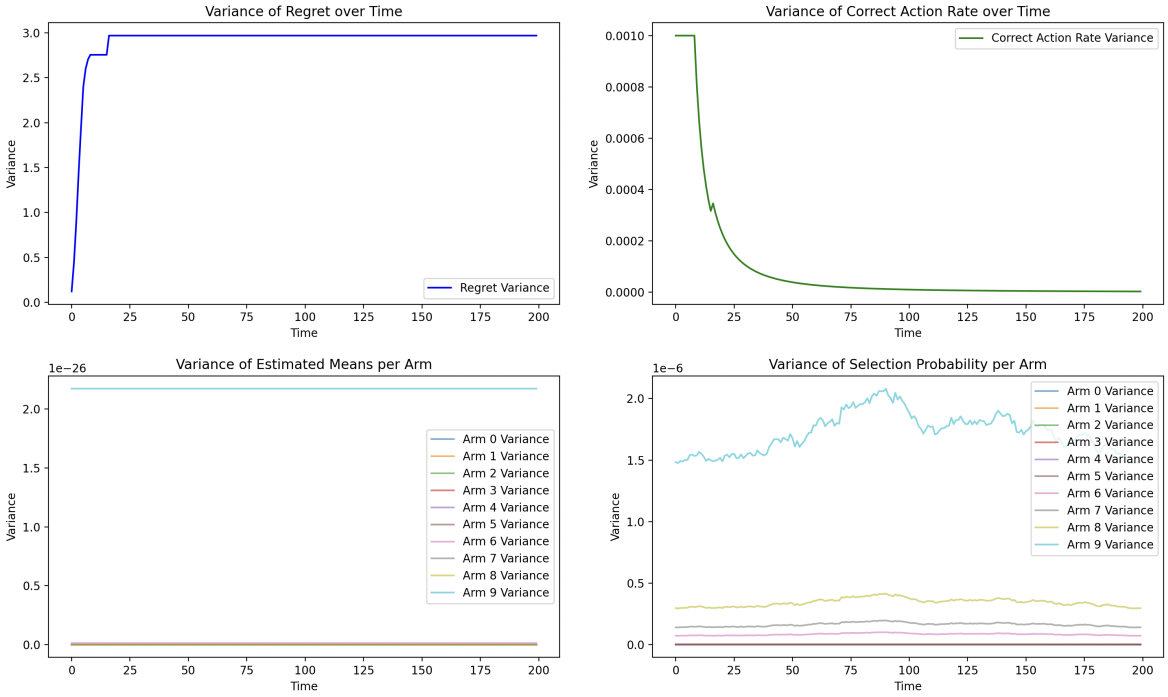

Figure 0.10: Greedy Variances

5 Boltzman

Algorithm 5: Simple Boltzmann exploration

Data : bandit model ν, vector $\hat{Q}$, parameter θ
Initialise $T_a = 0$ for all a;
while $t \leq n$ **do**
  Sample A_t from $\mathrm{SM}(\theta, \hat{Q})$;
  Obtain reward X_t by playing arm A_t;
  $T_{A_t} = T_{A_t} + 1$;
  $\hat{Q}_{A_t} = \hat{Q}_{A_t} + \frac{1}{T_{A_t}}(X_t - \hat{Q}_{A_t})$;
Result: actions $A_1, \ldots, A_n$ and rewards $X_1, \ldots, X_n$

We did not derive any theoretical results, only that we have a connection to UCB, if we choose $\theta_n = \sqrt{T_a(n)/C}$, i.e. as we have a more accurate estimate, we explore less and less, i.e. increase θ_n. We chose $\theta = 10$, i.e. did not choose it such that we would get UCB.

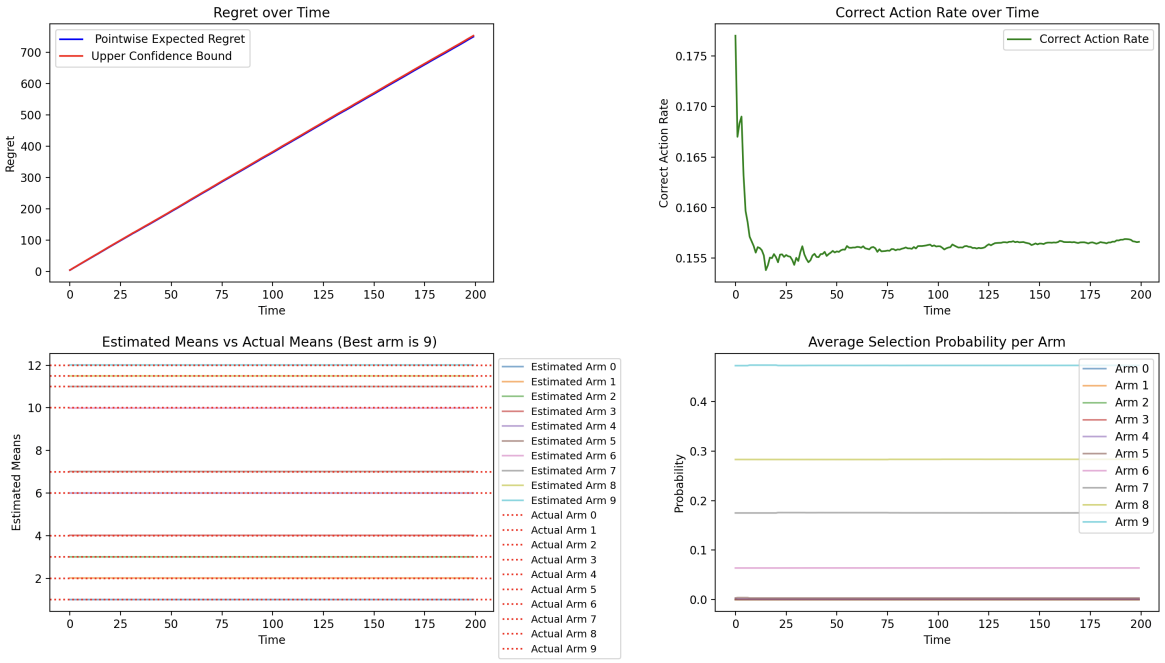

Figure 0.11: Boltzman

As we can see, we have linear regret, the best arm is found, but it is not played, because the correct action rate does not go to one, but the Q-Values are very accurate. I.e. the problem here is that θ was chosen too small and thus we only explored, but did not exploit. This is were the decreasing θ_n should come into play.

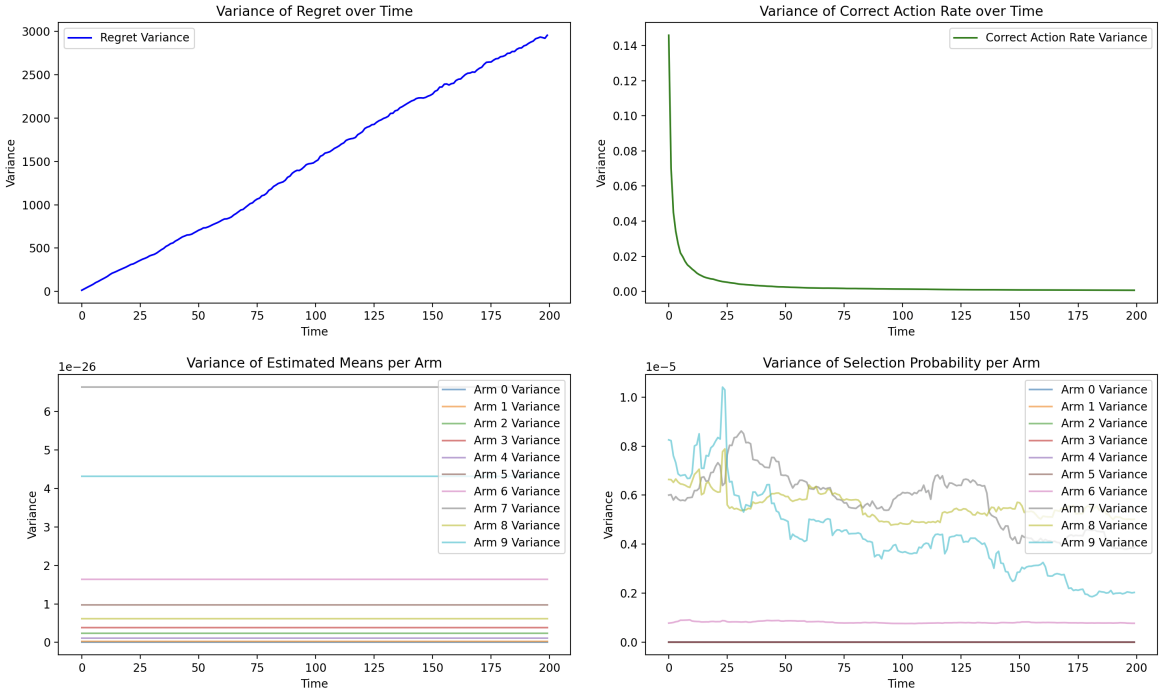

Figure 0.12: Boltzman Variances

6 Policy Gradient

Policy Gradient is a completely different approach. Here we try to estimate the value function of a policy that is parametrized by the Boltzman distribution. As we can write the gradient of the value function as an expectation, we can sample from it using Monte Carlo methods. We get the following iteration sceme.

$$\theta_{n+1} = \theta_n + \alpha \frac{1}{N} \sum_{i=1}^{N} X^i_{A^i_n} \nabla \log(\pi_\theta(A^i_n)), \quad n \in \mathbb{N}.$$

Here the goal is no longer to minimze the regret, but to find the best arm as fast as possible. So we will be mainly interested in the correct action rate and the avarage selction probability of the arm. We will play in every step Boltzman, and then update the θ vector according to Policy Gradient. We should see that if it works, we have "Boltzman" with increasing θ, i.e. UCB. However, the speed of convergence is unclear.

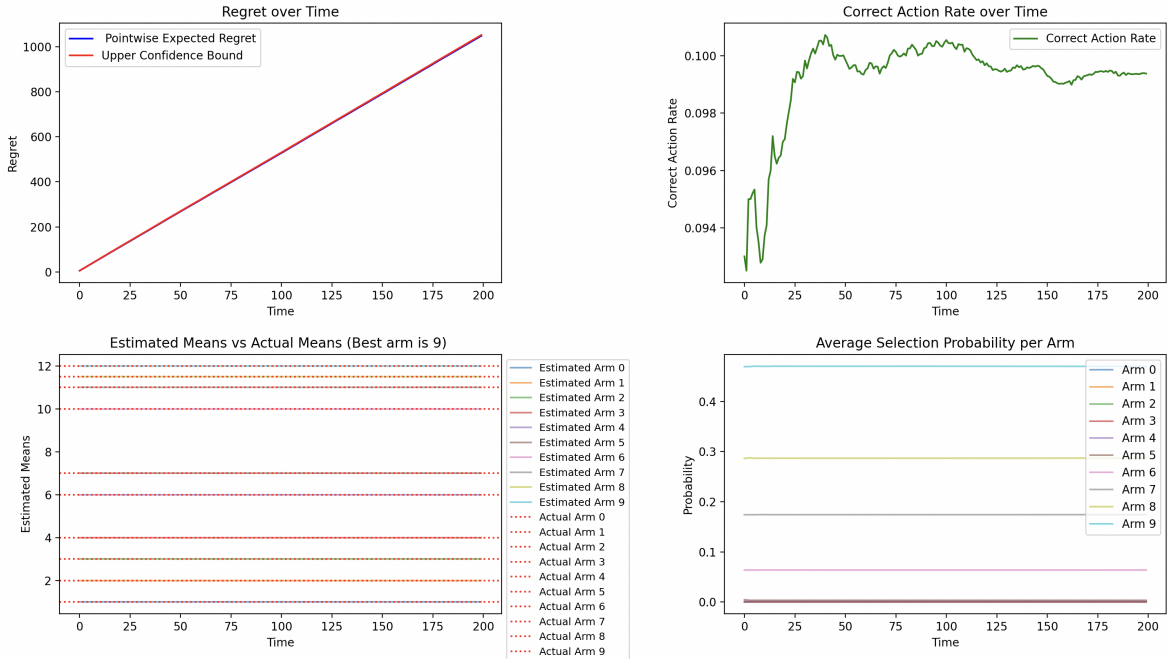

Figure 0.13: Policy Gradient

As we can see we have similar, performances as in Boltzmann with constant θ. After a lot of tweaking of the step size and the initial θ, we only got this performance.

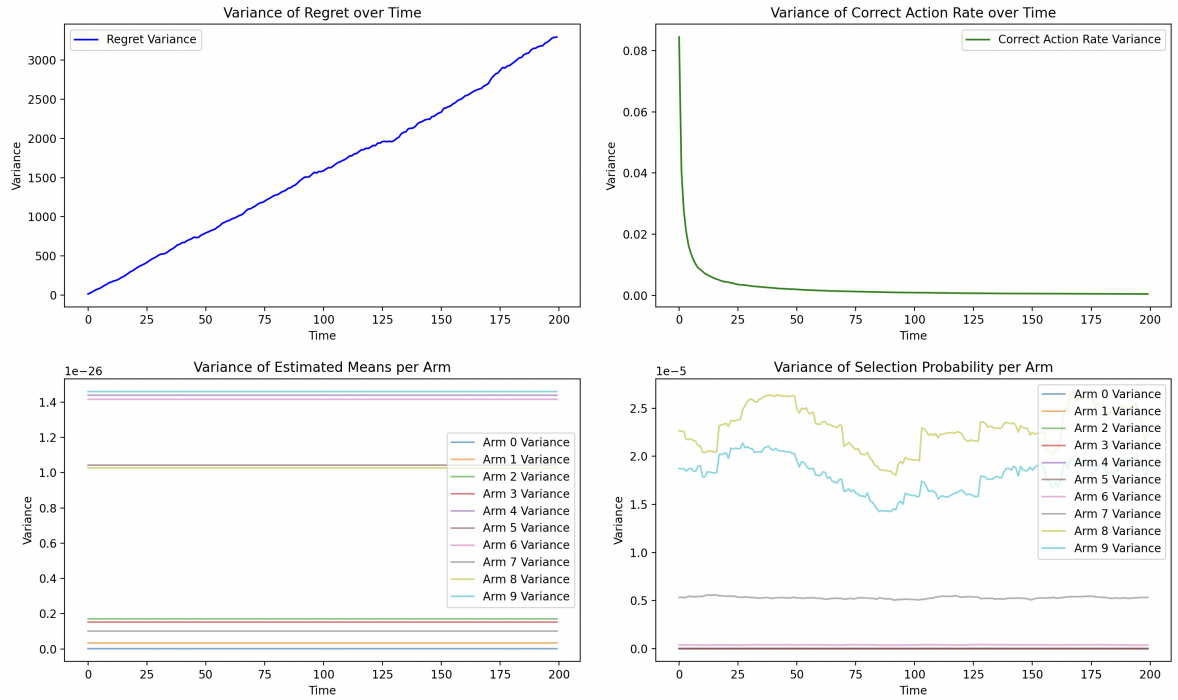

Figure 0.14: Policy Gradient Variances

In conclusion, we see that UCB is clearly the best choice.