Setup of 5-Armed Bandit

The bandit has 5 arms ({Arm[1],Arm[2],Arm[3],Arm[4],Arm[5]}),
and is also supported with 2 contexts ({ArmContext[1],ArmContext[2]}).
Each context follows standard normal distribution. The reward is given by:
(Reward = i + ArmContext[1]*i − 0.1*ArmContext[2]*i^2 + RandN[])

When no context is available, the expected payoff of Arm[i] is i.
Given the context, the expected payoff of Arm[i] is (i+ArmContext[1]*i-0.1*ArmContext[2]*i^2).

```
In[301]:= RandN[] := RandomVariate[NormalDistribution[]];
ArmContext = <|1 → RandN[], 2 → RandN[]|>;
ArmsList = {Arm[1], Arm[2], Arm[3], Arm[4], Arm[5]}

Out[303]= {<|Reward → 9.52772, NoContextRegret → 4, WithContextRegret → 5.46842|>,
 <|Reward → 5.41839, NoContextRegret → 3, WithContextRegret → 4.86687|>, <|Reward → 0.558041, NoContextRegret → 2, WithContextRegret → 0.166096|>,
 <|Reward → 8.06869, NoContextRegret → 1, WithContextRegret → 1.37113|>, <|Reward → 7.49962, NoContextRegret → 0, WithContextRegret → 0.|>}
```

```
In[304]:= Arm[i_] := (*At each call to the arm[i] (i∈{1,2,3,4,5}),
  do the following:*)
 (Reward = i
    + ArmContext[1] * i
    - 0.2 * ArmContext[2] * (6 - i) ^2
    + RandN[];
  (*This is the real reward returned by the arm.
    The Algorithm can only observe this*)
  NoContextRegret = 5
    - i;
  (*Regret = {Maximum expected reward arm}'s expected reward, here is 5
    - this arm's expected reward*)
  WithContextRegret = Max[Table[(ind + ArmContext[1] * ind - 0.1 * ArmContext[2] * ind^2), {ind, 5}]]
    - (i + ArmContext[1] * i - 0.1 * ArmContext[2] * i^2);
  (*Regret = {Maximum expected reward arm}'s expected reward
     - this arm's expected reward*)
  ArmContext = <|1 → RandN[], 2 → RandN[]|>; (*Update a random context to display*)
  <|"Reward" → Reward,
   "NoContextRegret" → NoContextRegret,
   "WithContextRegret" → WithContextRegret|>
 (*Return the Actual reward and the Regret values*))
```

We allow for 1000 times of playing the bandit. The algorithm’s target is to maximize payoff / minimize regret.

```
In[383]:= TotalTimes = 1000
Out[383]= 1000
```

Bandit Algorithms Without Context

Explore-First MAB Algorithm

This algorithm explores each arm by N times. For the rest, it choose to pull the arm with largest payoff given previous performance.

Mathematicians have proved that the optimal choice of N is: $N = \left(\frac{T}{K}\right)^{2/3} \text{Log}[T]^{1/3}$ for K-armed bandits.

```
In[306]:= OptimalN = Round[(TotalTimes/5)^(2/3) Log[TotalTimes]^(1/3)]
Out[306]= 65
```

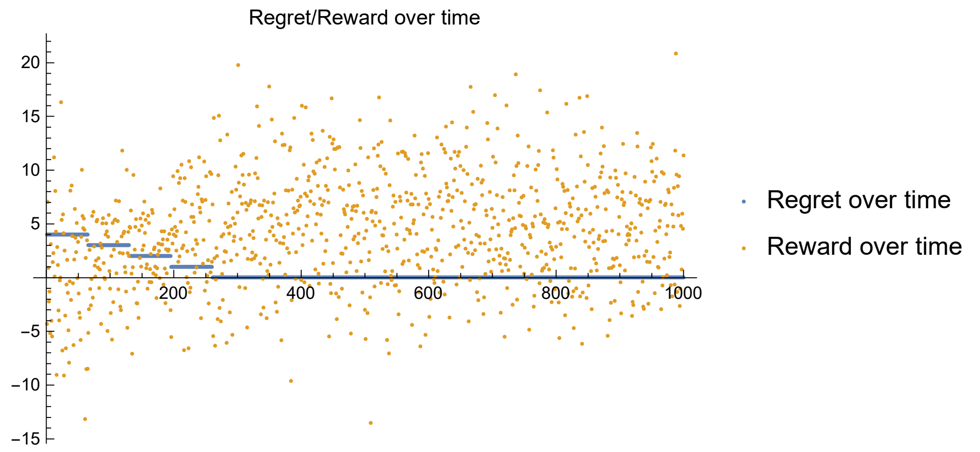
Implement the bandit algorithm:

```
In[307]:= ExploreFirstMABAlgorithm := (
  RegretList = {};
  RewardList = {};
  (*Record regret & reward*)
  MeanRewards = Table[Table[
      Results = Arm[i];
      AppendTo[RegretList, Results["NoContextRegret"]]; AppendTo[RewardList, Results["Reward"]];
      Results["Reward"]
      , {OptimalN}] // Mean, {i, 5}];
  Print["Exploration Ended. Mean Rewards for 5 arms are:", MeanRewards];
  (*This is the explored mean rewards from exploration phase. Print it out.*)
  MaxArmIndex = MaximalBy[Range[5], MeanRewards[[#]] &][[1]];
  Print["selected max-reward arm's index is:", MaxArmIndex];
  (*This is the selected max-reward arm's index.*)
  Table[
   Results = Arm[MaxArmIndex];
   AppendTo[RegretList, Results["NoContextRegret"]];
   AppendTo[RewardList, Results["Reward"]];
   , {TotalTimes - 5 * OptimalN}];
  (*Choose the max-return arm in exploration phase.*)
 )

In[308]:= ExploreFirstMABAlgorithm
```

```
Exploration Ended. Mean Rewards for 5 arms are:{0.409764, 1.8188, 2.31432, 4.12468, 4.28383}

selected max-reward arm's index is:5
```

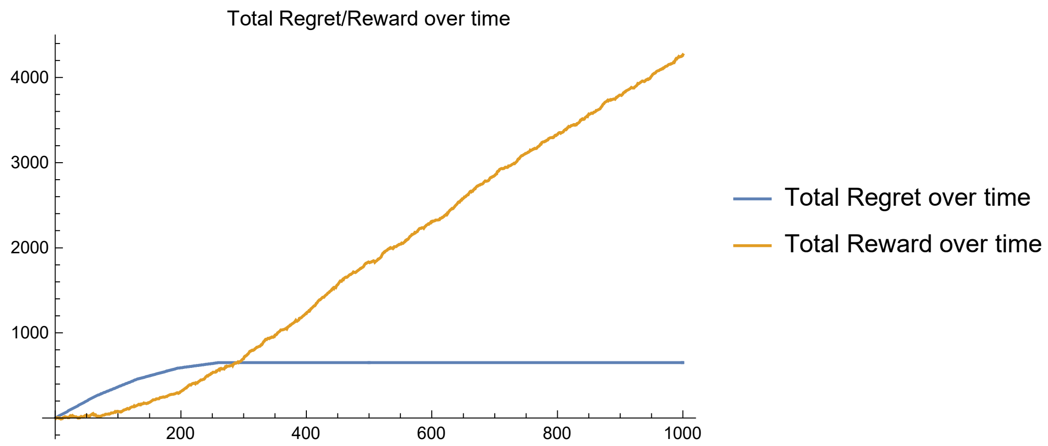
In[312]:=
```
ListPlot[ {RegretList, RewardList}, PlotLabel → "Regret/Reward over time",
 PlotLegends → {"Regret over time", "Reward over time"}, PlotStyle → Directive[{PointSize[Small]}] ]
```

Out[312]=



In[313]:=
```
ListLinePlot[ {Accumulate[RegretList], Accumulate[RewardList]},
 PlotLabel → "Total Regret/Reward over time", PlotLegends → {"Total Regret over time", "Total Reward over time"} ]
```

Out[313]=



In[314]:=
```
Total[RewardList] / TotalTimes
```

Out[314]= 4.2679

ϵ-Greedy MAB Algorithm

Mathematicians have proved that, to minimize the upper-bound of regret, the optimal choice of ϵ is: $\epsilon = (T)^{-1/3} (5\,\text{Log}[T])^{1/3}$ for K-armed bandits.

In[315]:=
```
Optimalϵ = N[ (TotalTimes)^-1/3 (5 Log[TotalTimes])^1/3 ]
```

Out[315]= 0.325663

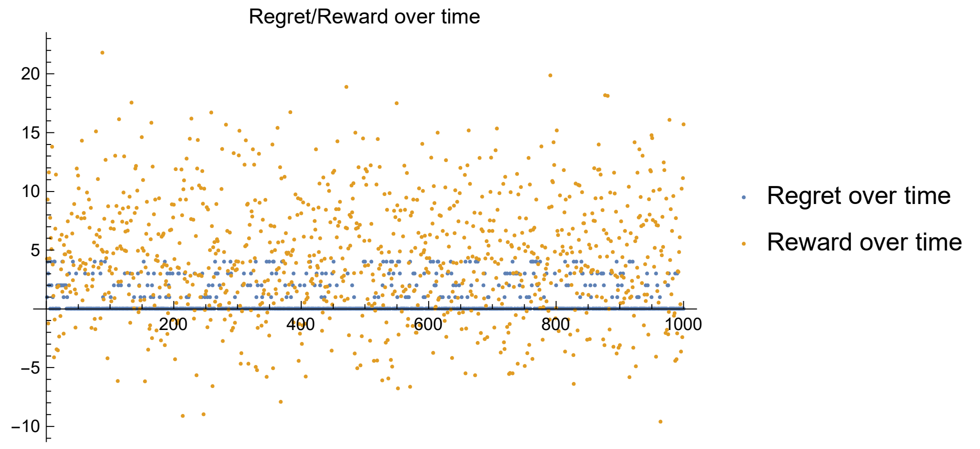
Implement the bandit algorithm:

In[316]:=
```
ϵGreedyMABAlgorithm := (
  RegretList = {};
  RewardList = {};
  (*Record regret & reward*)
  ExpectedRewards = {10, 10, 10, 10, 10};
  ArmDrawCounts = {0, 0, 0, 0, 0};
  (*initial expectation.*)
  Table[
   If[RandomReal[] < Optimalϵ,
     (*flip a coin*)
    tmpArm = RandomInteger[{1, 5}];
    Results = Arm[tmpArm];
    ExpectedRewards[[tmpArm]] = (ExpectedRewards[[tmpArm]] * ArmDrawCounts[[tmpArm]] + Results["Reward"]) / (ArmDrawCounts[[tmpArm]] + 1);
    ArmDrawCounts[[tmpArm]] ++;
    AppendTo[RegretList, Results["NoContextRegret"]];
    AppendTo[RewardList, Results["Reward"]];,
     (*randomly draw an arm*)
    Results = Arm[MaxArmIndex];
    ExpectedRewards[[MaxArmIndex]] =
      (ExpectedRewards[[MaxArmIndex]] * ArmDrawCounts[[MaxArmIndex]] + Results["Reward"]) / (ArmDrawCounts[[MaxArmIndex]] + 1);
    ArmDrawCounts[[MaxArmIndex]] ++;
    AppendTo[RegretList, Results["NoContextRegret"]];
    AppendTo[RewardList, Results["Reward"]];
     (*else, just draw the best arm so far*)
   ];
   MaxArmIndex = MaximalBy[Range[5], ExpectedRewards[[#]] &][[1]];
   (*update maxarmindex*)
   , {TotalTimes}];
  (*Choose the max-return arm in exploration phase.*)
 )
```

In[317]:=
```
ϵGreedyMABAlgorithm
```

In[319]:=
```
ListPlot[ {RegretList, RewardList}, PlotLabel → "Regret/Reward over time",
 PlotLegends → {"Regret over time", "Reward over time"}, PlotStyle → Directive[{PointSize[Small]}] ]
```

Out[319]=



In[320]:=
```
ListLinePlot[ {Accumulate[RegretList], Accumulate[RewardList]},
 PlotLabel → "Total Regret/Reward over time", PlotLegends → {"Total Regret over time", "Total Reward over time"} ]
```

Out[320]=

In[321]:=
```
Total[RewardList] / TotalTimes
```

Out[321]= 4.44966

Thompson-Sampling MAB Algorithm

Thompson sampling algorithm explores each arm at the Bayesian posterior probability of the arm being max-return arm. The algorithm requires a “prior” distribution, so the calculation is pretty difficult.

When no contexts are available, the distribution of each arm’s payoff is dependent on i:

In[105]:=
```
TransformedDistribution[
 i + i * u - 0.1 v * i^2 + w,
 {u ≈ NormalDistribution[], v ≈ NormalDistribution[], w ≈ NormalDistribution[]}]
```

Out[105]= $\text{NormalDistribution}\left[0. + i,\ \sqrt{1 + i^2 + 0.01\, i^4}\right]$

Typically, we don’t have a prior knowledge so accurate that we know exactly how the returns are distributed. Instead we usually have to infer from the observed data. Mathematica provides built-in function to do this task: EstimatedDistribution.

We specify Normal distribution as a prior distribution.

In[ ]:=
```
FindDistributionParameters[data,NormalDistribution[μ,θ]]
```

And, as we inferred the distribution parameter, from 计量经济学知识, we know the error of the estimated parameters. Thus, we can calculate the probability of one arm surpassing the other.

In[70]:= $\text{Probability}\left[x \le y,\ \left\{x \approx \text{NormalDistribution}\left[\mu_x,\ \frac{\theta_x}{\sqrt{\text{Len}_{xdata} - 1}}\right],\ y \approx \text{NormalDistribution}\left[\mu_y,\ \frac{\theta_y}{\sqrt{\text{Len}_{ydata} - 1}}\right]\right\}\right]$

Out[70]= $\frac{1}{2}\,\text{Erfc}\left[\frac{\mu_x - \mu_y}{\sqrt{2}\sqrt{\frac{\theta_x^2}{-1+\text{Len}_{xdata}} + \frac{\theta_y^2}{-1+\text{Len}_{ydata}}}}\right]$

In[ ]:=
```
FindDistributionParameters[ArmDrawResults[[1]],NormalDistribution[μ,θ]]
```

The probability of certain arm being the highest-return arm, is the product of expressions like this.

Also, we want to give the Thomson algorithm a little bit of prior knowledge before observing. We will spare 20 observing chances for each arm to provide a little bit prior knowledge to infer from.

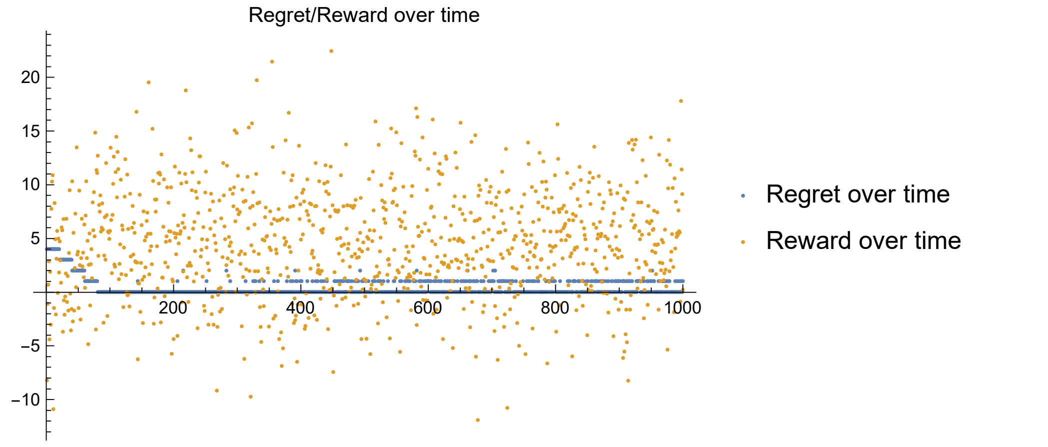
Implement the bandit algorithm:

In[322]:=
```
ThompsonSamplingMABAlgorithm := (
  RegretList = {};
  RewardList = {};
  (*Record regret & reward*)
  ArmDrawResults = Table[Table[
     Results = Arm[i];
     AppendTo[RegretList, Results["NoContextRegret"]]; AppendTo[RewardList, Results["Reward"]];
     Results["Reward"] + 0 (*optimistic boost. helps the algorithm to explore better*)
     , {20}], {i, 5}];
  DistParams = Table[({μ, θ} /. FindDistributionParameters[ArmDrawResults[[i]], NormalDistribution[μ, θ]]), {i, 5}];
  (*μ,θ*)
  ProbabilityOfBeingMax = Table[
    Product[1/2 Erfc[(DistParams[[j, 1]] - DistParams[[ind, 1]]) / √2 √(DistParams[[j, 2]]^2 / (-1 + Length[ArmDrawResults[[j]]]) +
          DistParams[[ind, 2]]^2 / (-1 + Length[ArmDrawResults[[ind]]]))], {j, Drop[Range[5], {ind}]}]
    , {ind, 5}];
  (*get initial startup knowledge, plus give every arm a optimistic boost.*)
  Table[
   tmpArm = RandomChoice[ProbabilityOfBeingMax → Range[5]];
   (*this is the sampling algorithm's choice*)
   Results = Arm[tmpArm];
   AppendTo[RegretList, Results["NoContextRegret"]];
   AppendTo[RewardList, Results["Reward"]];
   (*record regrets & rewards*)
   AppendTo[ArmDrawResults[[tmpArm]], Results["Reward"]];
   (*the Algorithm record the results.*)
   DistParams[[tmpArm]] = ({μ, θ} /. FindDistributionParameters[ArmDrawResults[[tmpArm]], NormalDistribution[μ, θ]]);
   (*update the params of distribution*)
   ProbabilityOfBeingMax = Table[
     Product[1/2 Erfc[(DistParams[[j, 1]] - DistParams[[ind, 1]]) / √2 √(DistParams[[j, 2]]^2 / (-1 + Length[ArmDrawResults[[j]]]) +
           DistParams[[ind, 2]]^2 / (-1 + Length[ArmDrawResults[[ind]]]))], {j, Drop[Range[5], {ind}]}]
     , {ind, 5}];
   (*update bayesian probability*)
   , {TotalTimes - 5 * 20}];
  (*Choose the max-return arm in exploration phase.*)
 )
```

In[323]:=
```
ThompsonSamplingMABAlgorithm
```

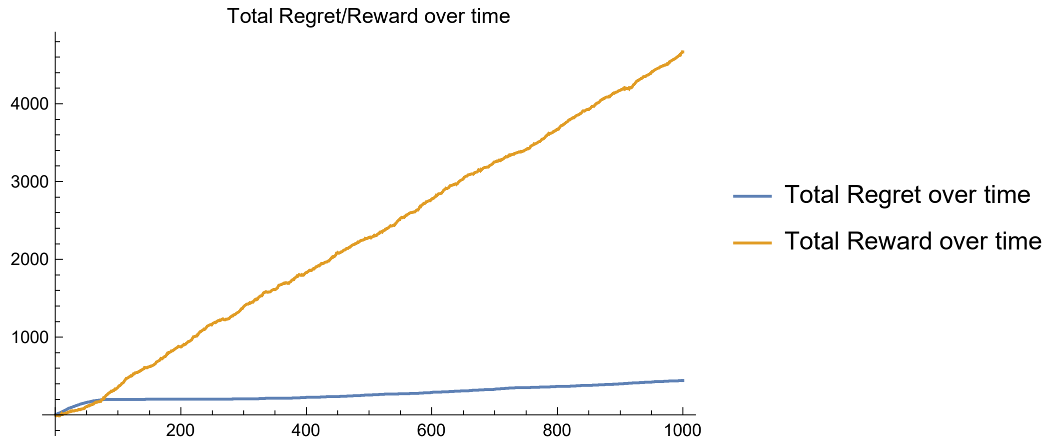
In[324]:=
```
ListPlot[ {RegretList, RewardList}, PlotLabel → "Regret/Reward over time",
 PlotLegends → {"Regret over time", "Reward over time"}, PlotStyle → Directive[{PointSize[Small]}] ]
```

Out[324]=



In[325]:=
```
ListLinePlot[ {Accumulate[RegretList], Accumulate[RewardList]},
 PlotLabel → "Total Regret/Reward over time", PlotLegends → {"Total Regret over time", "Total Reward over time"} ]
```

Out[325]=



Note that TS algorithm is pretty unstable. The initial draws will affect a lot. Sometimes it almost always choose the right arm, while sometime not that lucky.

In[326]:=
```
Total[RewardList] / TotalTimes
```

Out[326]=
```
4.66514
```

UCB MAB Algorithm

A final MAB algorithm for non-contextual algorithm is UCB algorithm. This algorithm is also adaptive, and it always chooses the arm with maximum Upper Confidence Bound.

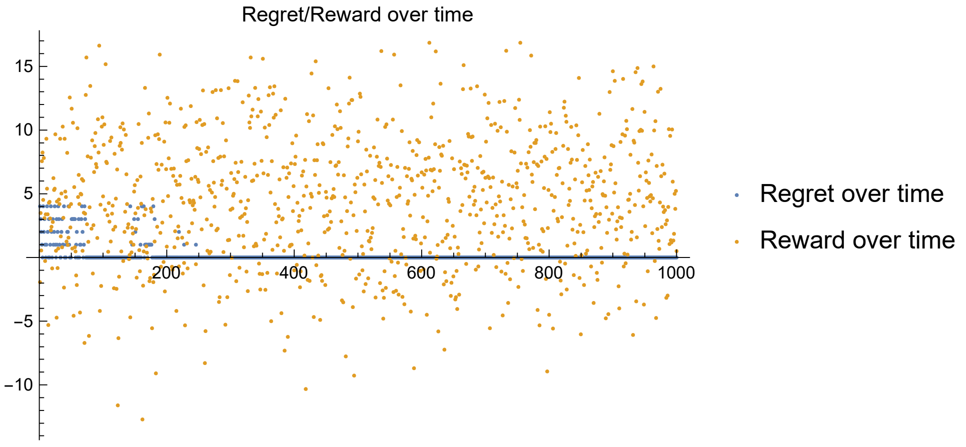
Implement the bandit algorithm:

In[338]:=
```
UCBMABAlgorithm := (
  RegretList = {};
  RewardList = {};
  (*Record regret & reward*)
  ArmDrawResults = {{20}, {20}, {20}, {20}, {20}};
  UCBList = {500, 500, 500, 500, 500};
  (*have to make an optimistic projection*)
  Table[
   tmpArm = MaximalBy[Range[5], UCBList[[#]] &][[1]];
   (*this is the sampling algorithm's choice*)
   Results = Arm[tmpArm];
   AppendTo[RegretList, Results["NoContextRegret"]];
   AppendTo[RewardList, Results["Reward"]];
   (*record regrets & rewards*)
   AppendTo[ArmDrawResults[[tmpArm]], Results["Reward"]];
   (*the Algorithm record the results.*)
   UCBList[[tmpArm]] = Quantile[EstimatedDistribution[ArmDrawResults[[tmpArm]], NormalDistribution[μ, θ]], 1 - 0.05];
   (*update UCB. we set the confidence level at 0.05 (one-sided)*)
   , {TotalTimes}];
 )
```

In[339]:=
```
UCBMABAlgorithm
```

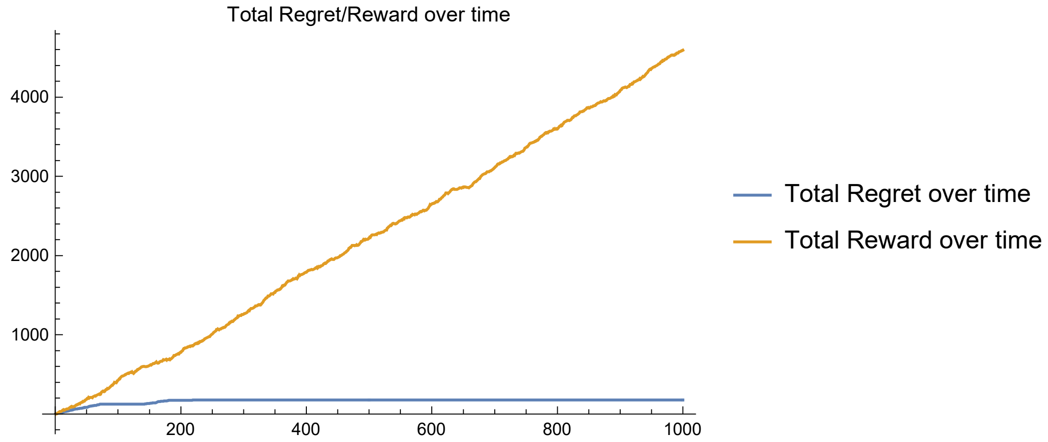
In[340]:=
```
ListPlot[ {RegretList, RewardList}, PlotLabel → "Regret/Reward over time",
 PlotLegends → {"Regret over time", "Reward over time"}, PlotStyle → Directive[{PointSize[Small]}] ]
```

Out[340]=



In[341]:=
```
ListLinePlot[ {Accumulate[RegretList], Accumulate[RewardList]},
 PlotLabel → "Total Regret/Reward over time", PlotLegends → {"Total Regret over time", "Total Reward over time"} ]
```

Out[341]=



In[342]:=
```
Total[RewardList] / TotalTimes
```

Out[342]=
```
4.5904
```

UCB is even more unstable than TS, and its performance under large-variance scenario is just poor. Basically UCB forbade itself from exploring arms from its maximizing greedy behaviour.

Contextual Bandit Algorithms

Contextual Thompson-Sampling MAB Algorithm

Extension to naive Thompson Sampling algorithm. Define a “prior” distribution with consideration of contexts.

We specify Normal distribution with its mean dependent on context as a prior distribution. Effectively, the context-generated distribution will also be a normal distribution.

```
In[]:= FindDistributionParameters[data, NormalDistribution[μ, θ]]
```

Calculate the probability of one arm surpassing the other (Same as before).

```
In[]:= Probability[x ≤ y, {x ≈ NormalDistribution[μx, θx/Sqrt[Lenxdata - 1]], y ≈ NormalDistribution[μy, θy/Sqrt[Lenydata - 1]]}]
```

$$\text{Out[]}= \frac{1}{2}\,\text{Erfc}\left[\frac{\mu_x-\mu_y}{\sqrt{2}\sqrt{\frac{\theta_x^2}{-1+\text{Len}_{\text{xdata}}}+\frac{\theta_y^2}{-1+\text{Len}_{\text{ydata}}}}}\right]$$

The probability of certain arm being the highest-return arm, is the product of expressions like this. Note we have to estimate the parameters using nonlinear model fit.

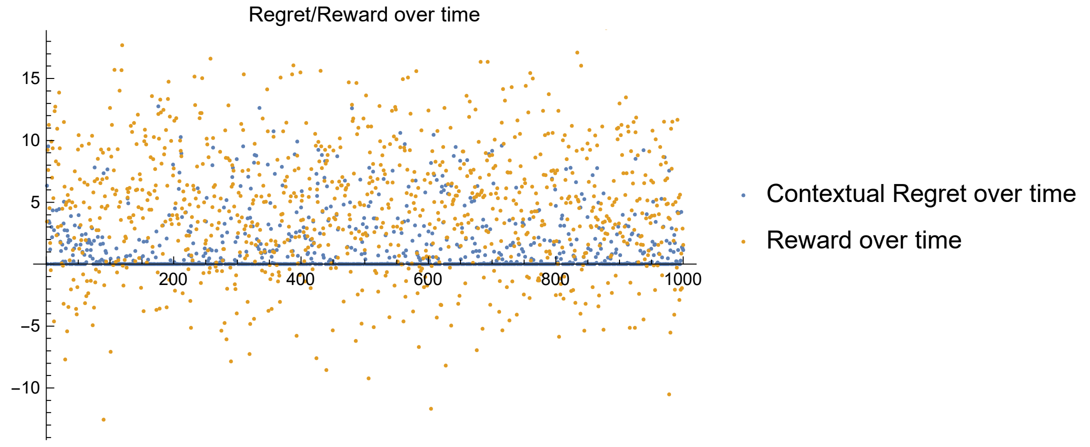
Also, we give the Thomson algorithm a little bit of optimistic prior knowledge before observing.

Implement the bandit algorithm:

```
In[377]:= ThompsonSamplingContextualMABAlgorithm := (
  RegretList = {};
  RewardList = {};
  (*Record regret & reward*)
  ArmDrawResults = {{{1, 0, 21}, {0, 1, 10}, {-1, 0, 19.5}, {0, -0.5, 10}},
    {{1, 0, 21}, {0, 1, 10}, {-1, 0, 19.5}, {0, -0.5, 10}}, {{1, 0, 21}, {0, 1, 10}, {-1, 0, 19.5}, {0, -0.5, 10}},
    {{1, 0, 21}, {0, 1, 10}, {-1, 0, 19.5}, {0, -0.5, 10}}, {{1, 0, 21}, {0, 1, 10}, {-1, 0, 19.5}, {0, -0.5, 10}}};
  (*optimistic prior expectation*)
  Table[
   DistParams = Table[
     nlm = NonlinearModelFit[ArmDrawResults[[ind]], a + b x + c y, {a, b, c}, {x, y}];
     bestFitParams = ({a, b, c} /. nlm["BestFitParameters"]);
     bestFitParamsErrors = nlm["ParameterErrors"];
     EstVariance = nlm["EstimatedVariance"];
     tmpDist = (TransformedDistribution[
          a + b x + c y + ϵ, {ϵ ≈ NormalDistribution[0, Sqrt[EstVariance]], a ≈ NormalDistribution[bestFitParams[[1]], bestFitParamsErrors[[1]]],
           b ≈ NormalDistribution[bestFitParams[[2]], bestFitParamsErrors[[2]]],
           c ≈ NormalDistribution[bestFitParams[[3]], bestFitParamsErrors[[3]]]}
         ] /. {x → ArmContext[1], y → ArmContext[2]});
     {tmpDist[[1]], tmpDist[[2]]}
     , {ind, 5}];
   ProbabilityOfBeingMax = Table[
     Product[1/2 Erfc[(DistParams[[j, 1]] - DistParams[[ind, 1]]) / Sqrt[2] Sqrt[(DistParams[[j, 2]]^2 / (-1 + Length[ArmDrawResults[[j]]]) +
            DistParams[[ind, 2]]^2 / (-1 + Length[ArmDrawResults[[ind]]]))]], {j, Drop[Range[5], {ind}]}]
     , {ind, 5}];
   (*find the Distribution Parameters, and the probability*)
   tmpArm = RandomChoice[ProbabilityOfBeingMax → Range[5]];
   (*this is the sampling algorithm's choice*)
   Results = Arm[tmpArm];
   AppendTo[RegretList, Results["WithContextRegret"]];
   AppendTo[RewardList, Results["Reward"]];
   (*record regrets & rewards*)
   AppendTo[ArmDrawResults[[tmpArm]], {ArmContext[1], ArmContext[2], Results["Reward"]}];
   (*the Algorithm record the context and results.*)
   , {TotalTimes}];
 )

In[392]:= ThompsonSamplingContextualMABAlgorithm
```

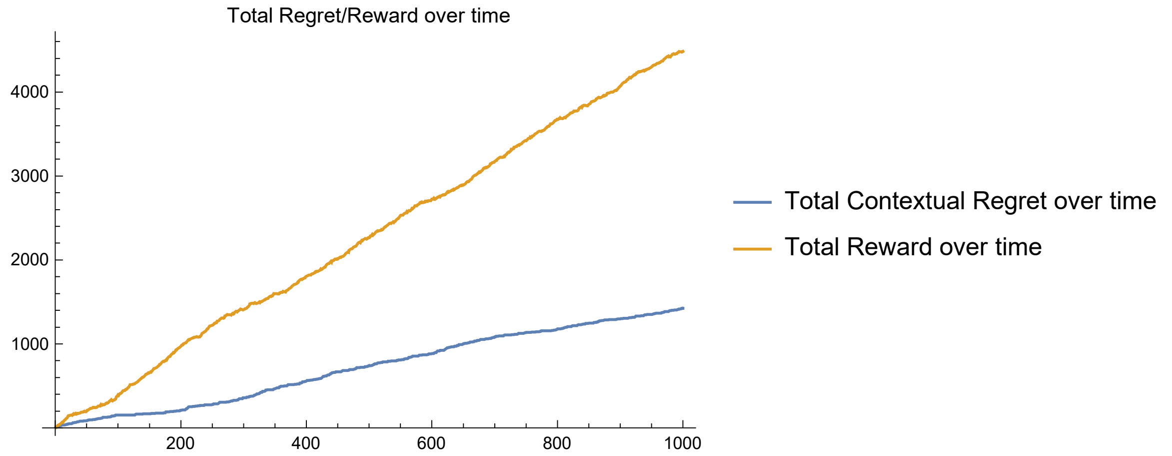
In[393]:=
```
ListPlot[{RegretList, RewardList}, PlotLabel → "Regret/Reward over time",
 PlotLegends → {"Contextual Regret over time", "Reward over time"}, PlotStyle → Directive[{PointSize[Small]}]]
```

Out[393]=



In[394]:=
```
ListLinePlot[{Accumulate[RegretList], Accumulate[RewardList]}, PlotLabel → "Total Regret/Reward over time",
 PlotLegends → {"Total Contextual Regret over time", "Total Reward over time"}]
```

Out[394]=



In[395]:=
```
Total[RewardList] / TotalTimes
```

Out[395]= 4.48149

Contextual UCB: LinUCB Algorithm

UCB algorithm can also be modified to utilize context information.

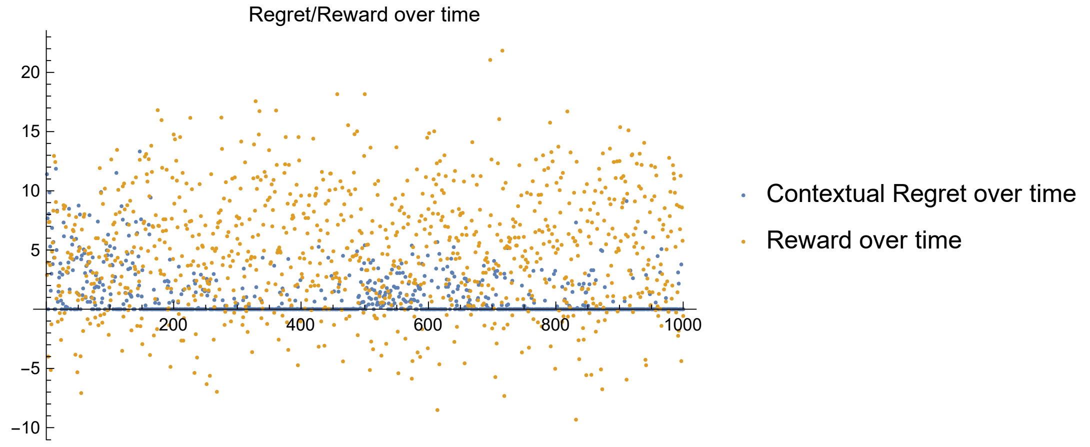
Implement the bandit algorithm:

In[483]:=
```
LinUCBContextualMABAlgorithm := (
  RegretList = {};
  RewardList = {};
  (*Record regret & reward*)
  ArmDrawResults = {{{1, 0, 15}, {0, 1, 10}, {-1, 0, 12.5}, {0, -0.5, 10}},
    {{1, 0, 15}, {0, 1, 10}, {-1, 0, 12.5}, {0, -0.5, 10}}, {{1, 0, 15}, {0, 1, 10}, {-1, 0, 12.5}, {0, -0.5, 10}},
    {{1, 0, 15}, {0, 1, 10}, {-1, 0, 12.5}, {0, -0.5, 10}}, {{1, 0, 15}, {0, 1, 10}, {-1, 0, 12.5}, {0, -0.5, 10}}};
  UCBList = {10000, 10000, 10000, 10000, 10000};
  (*have to make an optimistic projection*)
  Table[
   UCBList = UCBList * 0.5 + Table[
      nlm = NonlinearModelFit[ArmDrawResults[[ind]], a + b x + c y, {a, b, c}, {x, y}];
      bestFitParams = ({a, b, c} /. nlm["BestFitParameters"]);
      bestFitParamsErrors = nlm["ParameterErrors"];
      EstVariance = nlm["EstimatedVariance"];
      Quantile[TransformedDistribution[
         a + b x + c y + ϵ, {ϵ ≈ NormalDistribution[0, √EstVariance], a ≈ NormalDistribution[bestFitParams[[1]], bestFitParamsErrors[[1]]],
          b ≈ NormalDistribution[bestFitParams[[2]], bestFitParamsErrors[[2]]],
          c ≈ NormalDistribution[bestFitParams[[3]], bestFitParamsErrors[[3]]]}
        ], 1 - 0.00005] /. {x → ArmContext[1], y → ArmContext[2]}
      , {ind, 5}];
   (*find the UCB FOR THE CONTEXT STATUS QUO. we set the confidence level at 0.00005 (one-sided)*)
   tmpArm = MaximalBy[Range[5], UCBList[[#]] &][[1]];
   (*this is the sampling algorithm's choice*)
   Results = Arm[tmpArm];
   AppendTo[RegretList, Results["WithContextRegret"]];
   AppendTo[RewardList, Results["Reward"]];
   (*record regrets & rewards*)
   AppendTo[ArmDrawResults[[tmpArm]], {ArmContext[1], ArmContext[2], Results["Reward"]}];
   (*the Algorithm record the context and results.*)
   , {TotalTimes}];
 )
```

In[492]:=
```
LinUCBContextualMABAlgorithm
```

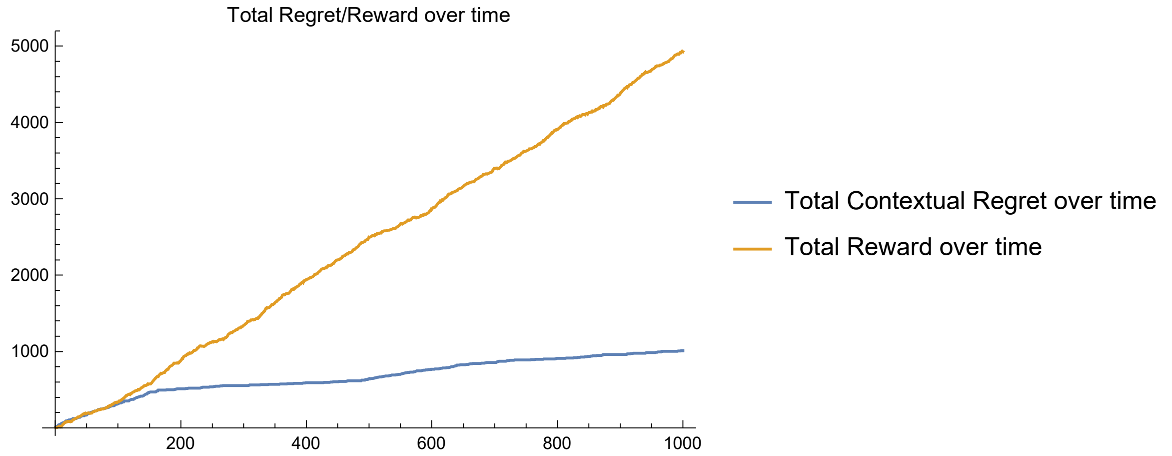
In[493]:=
```
ListPlot[ {RegretList, RewardList}, PlotLabel → "Regret/Reward over time",
 PlotLegends → {"Contextual Regret over time", "Reward over time"}, PlotStyle → Directive[{PointSize[Small]}] ]
```

Out[493]=



In[494]:=
```
ListLinePlot[ {Accumulate[RegretList], Accumulate[RewardList]}, PlotLabel → "Total Regret/Reward over time",
 PlotLegends → {"Total Contextual Regret over time", "Total Reward over time"} ]
```

Out[494]=



In[495]:=
```
Total[RewardList] / TotalTimes
```

Out[495]=
```
4.92722
```

Contextual bandits have greater potential to optimize reward, because they observe better information. You can see contextual regret is still pretty high, but the performance has also exceeded non-contextual bandits.