Solutions for ST340 Lab 5

2019–20

Bernoulli Bandits

```r
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 3.5.2
```

(a) Set Bernoulli success parameters for each arm.

```r
ps <- c(0.4,0.6)
```

(b) This is a template for an Epsilon-greedy algorithm, runs for n steps:

```r
epsilon.greedy <- function(ps,epsilon,n) {
  as <- rep(0,n)
  rs <- rep(0,n)

  # initial number of plays and number of successes is 0 for each arm
  ns <- rep(0,2); ss <- rep(0,2)

  # at first, play each arm once
  for (i in 1:2) {
    a <- i
    r <- runif(1) < ps[a]
    ns[a] <- ns[a] + 1
    ss[a] <- ss[a] + r
    as[i] <- a
    rs[i] <- r
  }

  # now follow the epsilon greedy strategy
  for (i in 3:n) {
    # with probability epsilon, pick an arm uniformly at random
    if (runif(1) < epsilon) {
      a <- sample(2,1)
    } else { # otherwise, choose the "best arm so far".
      a <- which.max(ss/ns)
    }
    # simulate the reward
    r <- ifelse(runif(1) < ps[a], 1, 0)

    # update the number of plays, successes
    ns[a] <- ns[a] + 1
    ss[a] <- ss[a] + r

    # record the arm played and the reward received
    as[i] <- a
    rs[i] <- r
  }
  return(list(as=as,rs=rs))
}
```

Run `epsilon.greedy` with the given `ps` and a choice of `epsilon` and see how well it does.

```
eg.out <- epsilon.greedy(ps=ps,epsilon=.1,n=1e4)
sum(eg.out$rs)/length(eg.out$rs)
```

```
## [1] 0.594
```

This should be close to 0.6.

(c) Implement a `sample_arm` routine, for use in the Thompson sampling code below.

```
sample_arm.bernoulli  <- function(ns,ss) {
  alphas <- 1 + ss      # successes
  betas <- 1 + ns - ss # failures

  t1 <- rbeta(1,alphas[1],betas[1])
  t2 <- rbeta(1,alphas[2],betas[2])
  if (t1 > t2) {
    return(1)
  } else {
    return(2)
  }
}
```

The code above assumes that $\alpha_0 = \beta_0 = 1$ (see slide 15 from the lecture).

```
thompson.bernoulli <- function(ps,n) {
  as <- rep(0,n)
  rs <- rep(0,n)

  # number of times each arm has been played
  # and number of corresponding successes
  ns <- rep(0,2); ss <- rep(0,2)

  for (i in 1:n) {
    a <- sample_arm.bernoulli(ns,ss)
    r <- ifelse(runif(1) < ps[a], 1, 0)
    ns[a] <- ns[a] + 1
    ss[a] <- ss[a] + r
    as[i] <- a
    rs[i] <- r
  }
  return(list(as=as,rs=rs))
}
```

(d) Run the Thompson scheme and compare its performance to that of `epsilon.greedy`.

```
thompson.bernoulli.out <- thompson.bernoulli(ps=ps,n=1e4)
sum(thompson.bernoulli.out$rs)/length(thompson.bernoulli.out$rs)
```

```
## [1] 0.6147
```

Should be close to 0.6; who got more rewards?

```
sum(eg.out$rs)
```

```
## [1] 5940
```

```r
sum(thompson.bernoulli.out$rs)
```

```
## [1] 6147
```