SMU: Lecture 6

Monday, March 7, 2022

(Heavily inspired by the Stanford RL Course of Prof. Emma Brunskill, but all potential errors are mine.)

Bandits

(Recap from last week)

Multi-Armed Bandits

$$P[R = r \mid A = i]$$

We can choose actions $\{1,2,3,4\}$ *and each of them leads to a different distribution of rewards.*

Setting

Multi-armed bandit is essentially a degenerate MDP that contains a single state.

Definition: A multi-armed bandit is given by:

A set A containing m actions $a_1, a_2, \ldots, a_m$ (each can be thought of as "pulling an arm").

Reward distributions $P[R_t = r \,|\, A_t = a]$, that is the distribution of rewards at time t given the action at time t.

At each step, the agent takes an action and receives a reward sampled from the above distribution.

The *informal* goal is to maximize the reward $\sum_{t=1}^{T} R_t$…. of course, this is a random variable.

Regret (1/3)

Action-value: $Q(a) = \mathbb{E}[R_t | A_t = a]$.

Similar to MDPs where we had $Q^\pi(s, a)$. However, we do not need s because we now have only one state. So we could rewrite it as $Q^\pi(a)$. But then, since the action only affects the immediate reward and not to which state we get, the whole notion of policy is not very important for Q in this setting, so we drop that as well and end up with $Q(a) = \mathbb{E}[R_t | A_t = a]$.

Regret (2/3)

Optimal value: $V^* = \max_{a \in A} Q(a) = \max_{a \in A} \mathbb{E}[R_t | A_t = a].$

Optimal action: $a^* = \arg\max_{a \in A} Q(a).$

Regret: $L_t = V^* - Q(A_t).$

That is, regret is the "opportunity loss" at time t. Note that we use expected value in the definition of regret (recall how we defined $Q(a)$). That means we are not measuring regret directly in terms of what we observe. Since the parameters of bandits will generally be unknown, it also means we will not be able to compute regret directly.

Regret (3/3)

Total regret: $$L_T^{tot} = \sum_{t=1}^{T} L_t = \sum_{t=1}^{T} (V^* - Q(A_t)).$$

Minimizing total regret is the same as maximizing the expected sum of rewards (i.e. return).

What We Want… (1/2)

We want to find algorithms where the regret will grow slowly with the number of time steps taken.

Note that:

> *When regret does not grow at all after some time, that means that we are already taking the optimal action.*
>
> *Regret is the difference between best possible return and the return under our strategy. So when the regret grows slowly, it means we are already doing quite well.*

What We Want… (2/2)

If we knew the expectations $\mathbb{E}[R_t | A_t = a]$ *then the problem would be trivial, but it would not be reinforcement learning.*

We could try to first estimate $\mathbb{E}[R_t | A_t = a]$ *by taking actions completely randomly. However, then in this first part we would incur high regret and it is also not clear how long we should be estimating (because that actually depends on the values of* $\mathbb{E}[R_t | A_t = a]$*)… So we will need something smarter.*

UCB Algorithm: Basic Idea

Upper-Confidence Bound (UCB) Algorithm

For every action $a \in A$, maintain an upper bound $U_t(a)$ *(the upper bound will change with time, that is why it is indexed by t).*

In every time step t, take the action that has the maximum upper bound, i.e. take the action $\arg\max_{a \in A} U_t(a)$.

After observing the reward, update the estimates.

UCB Algorithm

Initialization:

Take every action $a \in A$ once and record the rewards in $\hat{Q}(a)$.

$t := 1$

Loop:

Compute upper confidence bounds for all actions $a_i \in A$:

$$U_t(a_i) = \hat{Q}(a_i) + \sqrt{\frac{1}{2N(a_i)} \log \frac{2t^2}{\delta}}$$

Use the action $a_t = \arg\max_{a \in A} U_t(a)$ and observe the reward r_t.

Update $N(a_t) := N(a_1) + 1$

Update $\hat{Q}(a_t) := \hat{Q}(a_t) + \frac{1}{N(a_t)}(r_t - Q(a_t))$.

$t := t + 1$

A Primer (Or a Refresher) on Bayesian Learning

The Basic Idea (1)

We suppose that the data (learning example) are generated by a distribution that looks as follows:

$$p(\mathbf{X} \mid \alpha) = \int p(\mathbf{X} \mid \theta) p(\theta \mid \alpha) d\theta$$

where

- $\mathbf{X}$ is the sample, i.e., $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_n]$ where $\mathbf{x}_i$ is the i-th example,
- $p(\mathbf{X} \mid \alpha)$ is the marginal probability (or probability density) of the sample $\mathbf{X}$, given the hyperparameters α
- $p(\theta \mid \alpha)$ is the prior probability of the parameters θ given the hyperparameters α.

The Basic Idea (2)

We suppose that the data (learning example) are generated by a distribution that looks as follows:

$$p(\mathbf{X} \mid \alpha) = \int p(\mathbf{X} \mid \theta) p(\theta \mid \alpha) d\theta$$

We usually want to learn about the hidden parameters θ.

Therefore we need to compute $p(\theta \mid \mathbf{X}, \alpha)$ for which we have:

$$p(\theta \mid \mathbf{X}, \alpha) = \frac{p(\mathbf{X} \mid \theta, \alpha) p(\theta \mid \alpha)}{p(\mathbf{X} \mid \alpha)} \propto p(\mathbf{X} \mid \theta, \alpha) p(\theta \mid \alpha).$$

Example (Slide 1): Bernoulli Distribution

Bernoulli random variable X:

$$P[X = 1] = \theta, P[X = 0] = 1 - \theta.$$

It is the distribution of a biased coin (one that lands "heads" with probability θ and "tails" with probability$1 - \theta$.

Example (Slide 2): A Prior

Prior: θ is sampled from the beta distribution with parameters α and β.

But what is the beta-distribution?

Example (Slide 3): Beta Distribution

Beta distribution:

$$p_{beta}(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

for $x \in (0; 1)$.

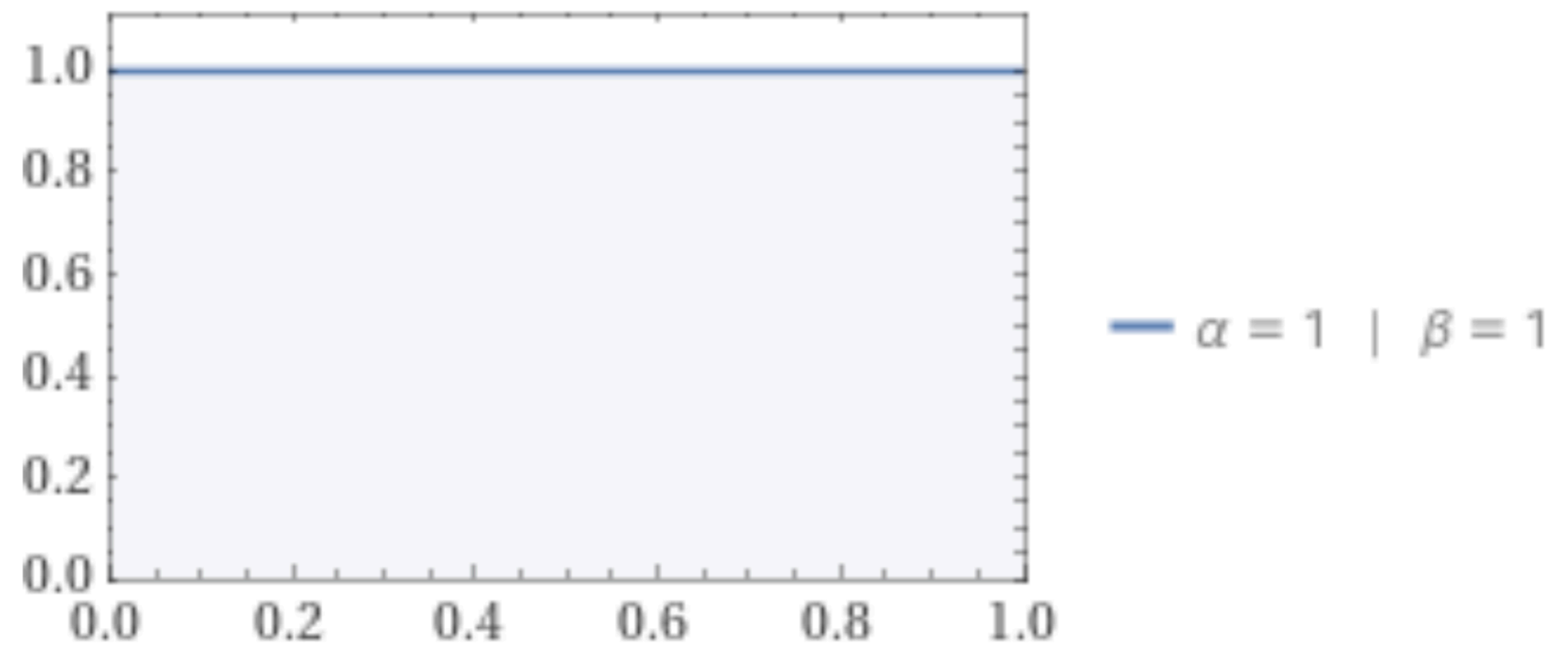
Here, $B(\alpha, \beta)$ is the beta function defined as $B(\alpha, \beta) = \dfrac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ and $\Gamma(\alpha)$ is the gamma function, which satisfies $\Gamma(n) = (n-1)!$.

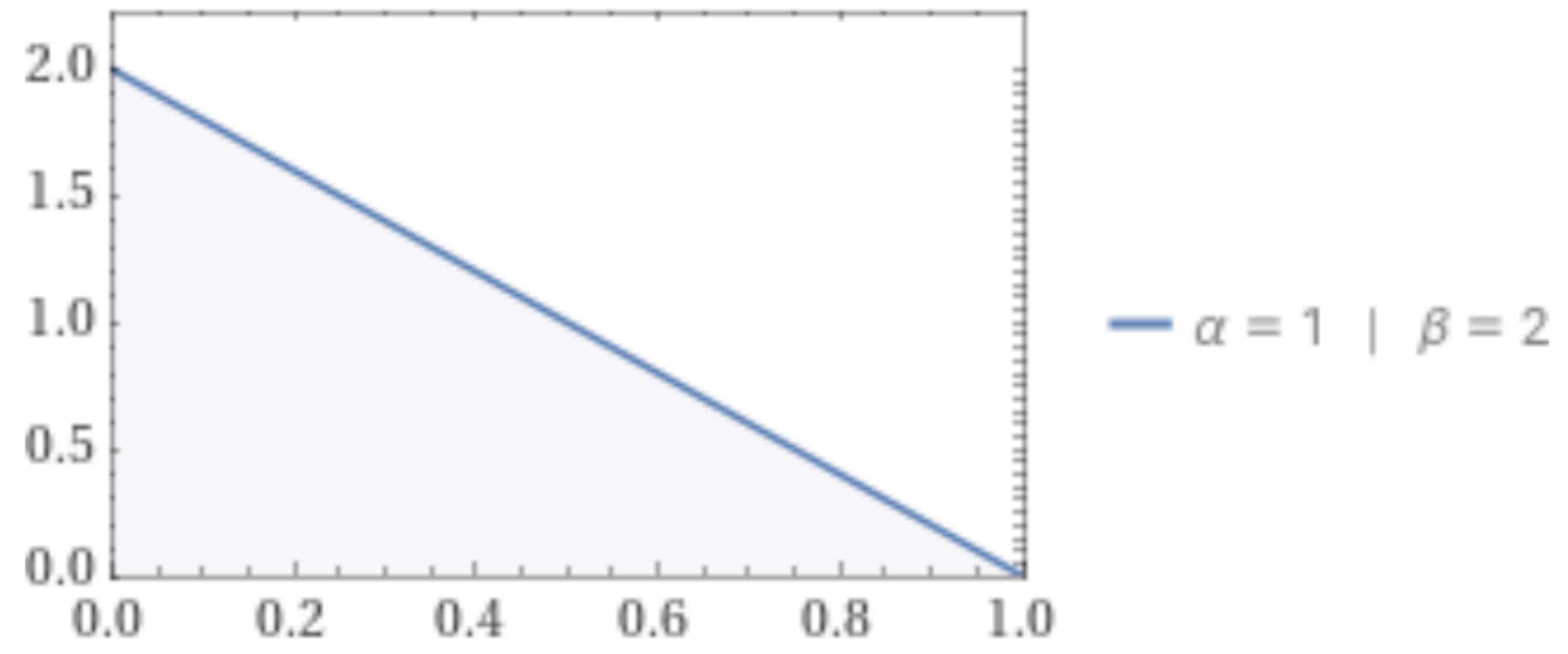
Example (Slide 4): Beta Function

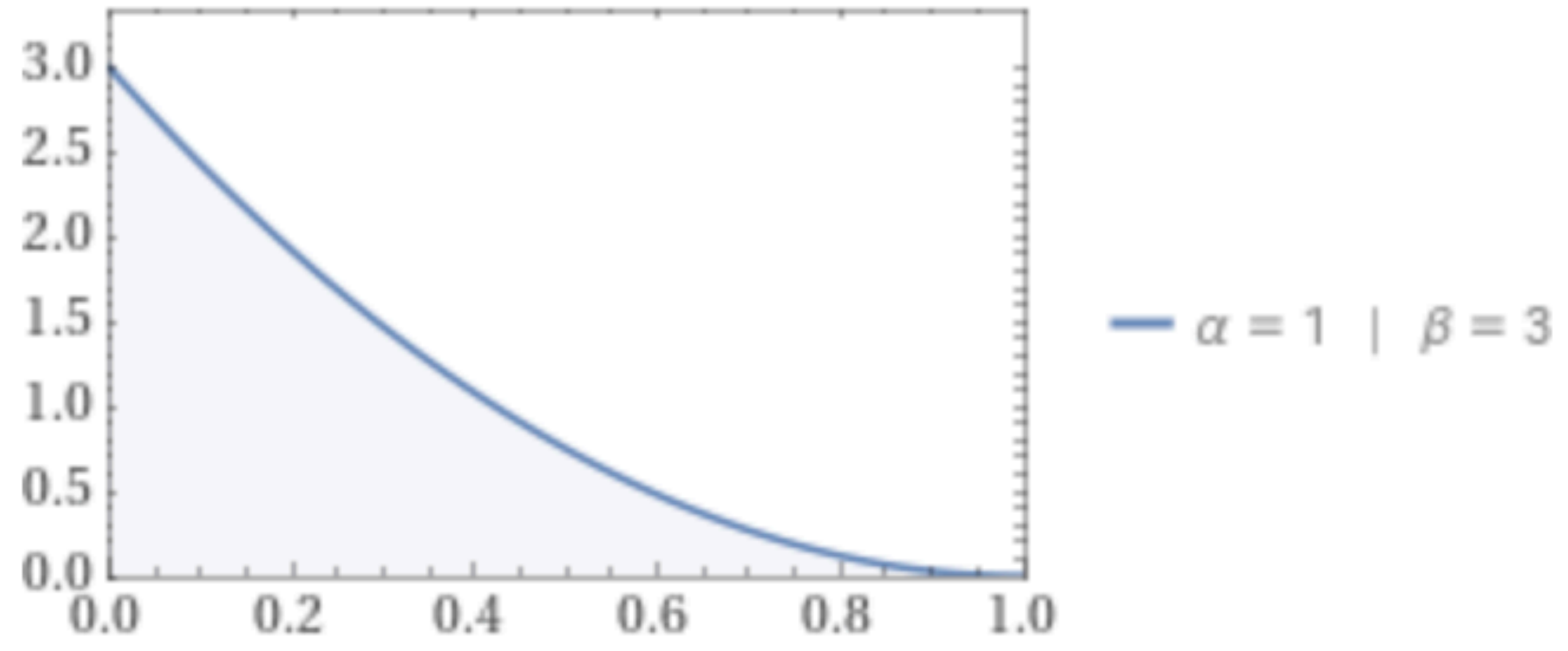
We will also need another the following alternative definiton of the beta function:

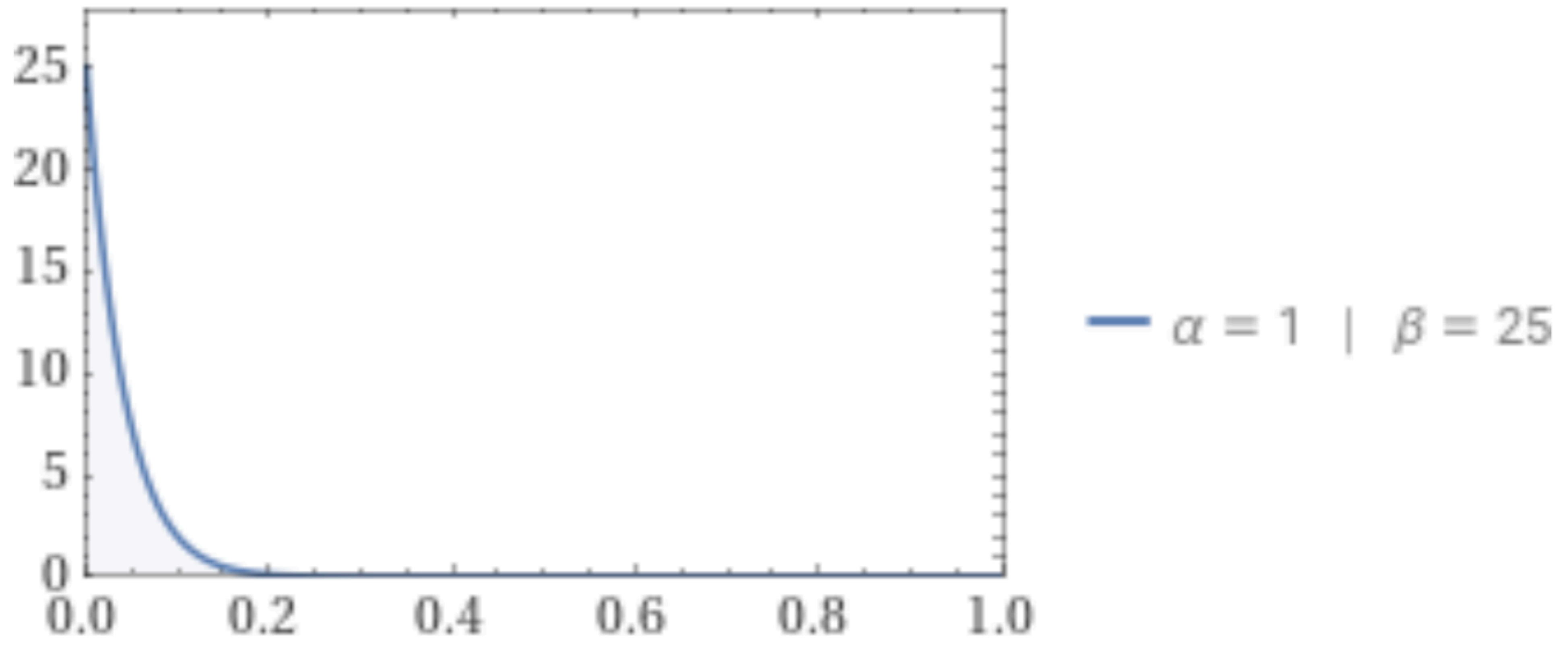
$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx$$

Example (Slide 5): Beta Distribution

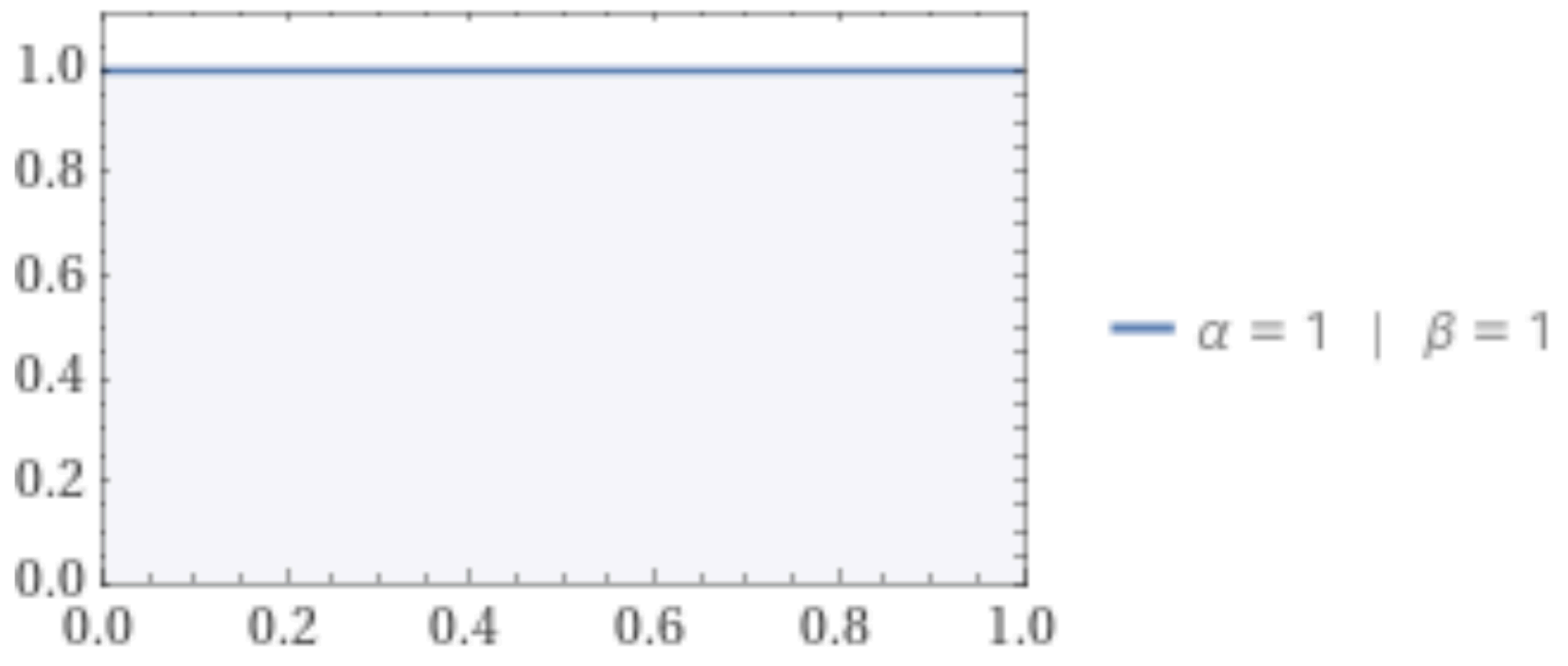


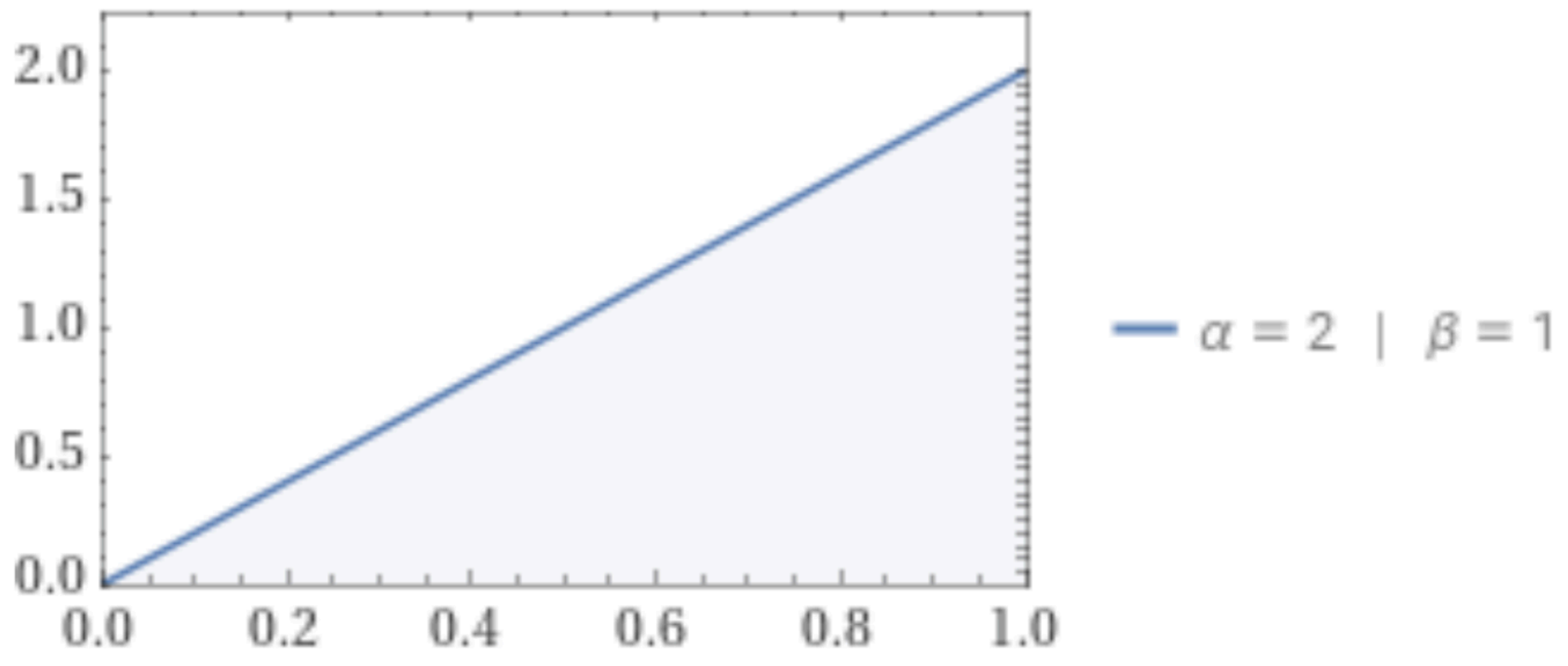


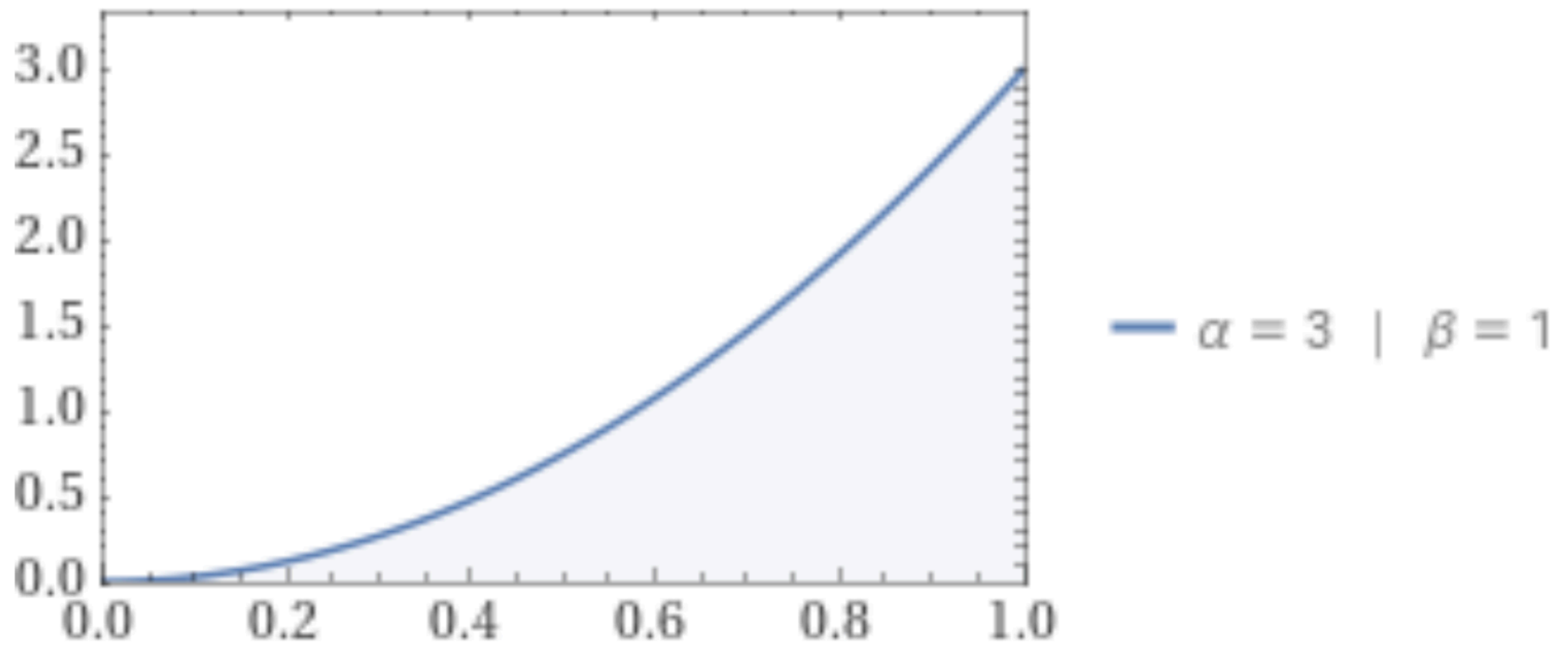


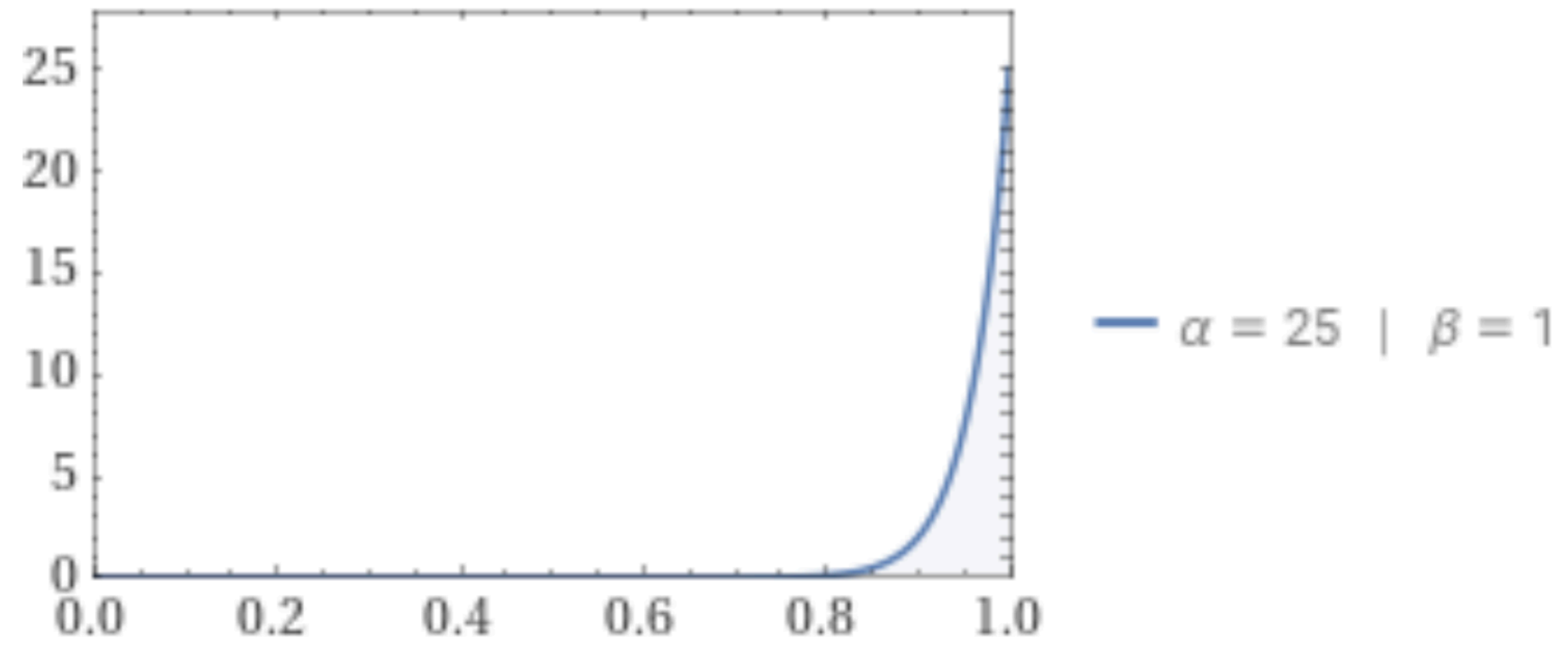


Example (Slide 6): Beta Distribution









Example (Slide 7): Computing the Posterior

Let $\mathbf{X} = [x_1, x_2 \dots, x_n]$ be our sample. Since x_i's are Bernoulli random variebls, $x_i \in \{0,1\}$. We will denote by Pos the number of 1's in $\mathbf{X}$ and by Neg the number of 0's.

First, we compute the data likelihood given θ:

$$p(\mathbf{X} \mid \theta, \alpha, \beta) = P(\mathbf{X} \mid \theta) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} = \theta^{\text{Pos}} \cdot (1-\theta)^{\text{Neg}}.$$

Example (Slide 8): Computing the Posterior

Let $\mathbf{X} = [x_1, x_2 \ldots, x_n]$ be our sample. Since x_i's are Bernoulli random variebls, $x_i \in \{0,1\}$. We will denote by Pos the number of 1's in $\mathbf{X}$ and by Neg the number of 0's.

Next, we compute the marginal distribution of the data given the hyperparameters:

$$p(\mathbf{X} \,|\, \alpha, \beta) = \int_0^1 \prod_{i=1}^n \theta^{x_i}(1-\theta)^{1-x_i} p_{beta}(\theta \,|\, \alpha, \beta) d\theta = \int_0^1 \prod_{i=1}^n \theta^{x_i}(1-\theta)^{1-x_i} \frac{1}{B(\alpha,\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} d\theta =$$

$$= \int_0^1 \theta^{Pos}(1-\theta)^{Neg} \frac{1}{B(\alpha,\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} d\theta = \frac{1}{B(\alpha,\beta)} \int_0^1 \theta^{\alpha+Pos-1}(1-\theta)^{\beta+Neg-1} d\theta =$$

$$= \frac{1}{B(\alpha,\beta)} B(\alpha + \text{Pos}, \beta + \text{Neg})$$

Example (Slide 9): Computing the Posterior

Finally, we compute the posterior distribution of the parameter (where we plug in what we derived on the preceding slides)**…**

$$p(\theta \mid \mathbf{X}, \alpha, \beta) = \frac{p(\mathbf{X} \mid \theta, \alpha, \beta) p(\theta \mid \alpha, \beta)}{p(\mathbf{X} \mid \alpha, \beta)} =$$

$$= \frac{\theta^{\text{Pos}} \cdot (1 - \theta)^{\text{Neg}} \frac{1}{B(\alpha, \beta)} \theta^{\alpha - 1} (1 - \theta)^{\beta - 1}}{\frac{1}{B(\alpha, \beta)} B(\alpha + \text{Pos}, \beta + \text{Neg})} =$$

$$= \frac{\theta^{\alpha + \text{Pos} - 1} (1 - \theta)^{\beta + \text{Neg} - 1}}{B(\alpha + \text{Pos}, \beta + \text{Neg})} = p_{beta}(\theta \mid \alpha + \text{Pos}, \beta + \text{Neg})$$

Example (Slide 10): Computing the Posterior

Finally, we compute the posterior distribution of the parameter (where we plug in what we derived on the preceding slides)**...**

$$p(\theta \mid \mathbf{X}, \alpha, \beta) = \frac{p(\mathbf{X} \mid \theta, \alpha, \beta) p(\theta \mid \alpha, \beta)}{p(\mathbf{X} \mid \alpha, \beta)} =$$

$$= \frac{\theta^{\text{Pos}} \cdot (1-\theta)^{\text{Neg}} \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}}{\frac{1}{B(\alpha, \beta)} B(\alpha + \text{Pos}, \beta + \text{Neg})} =$$

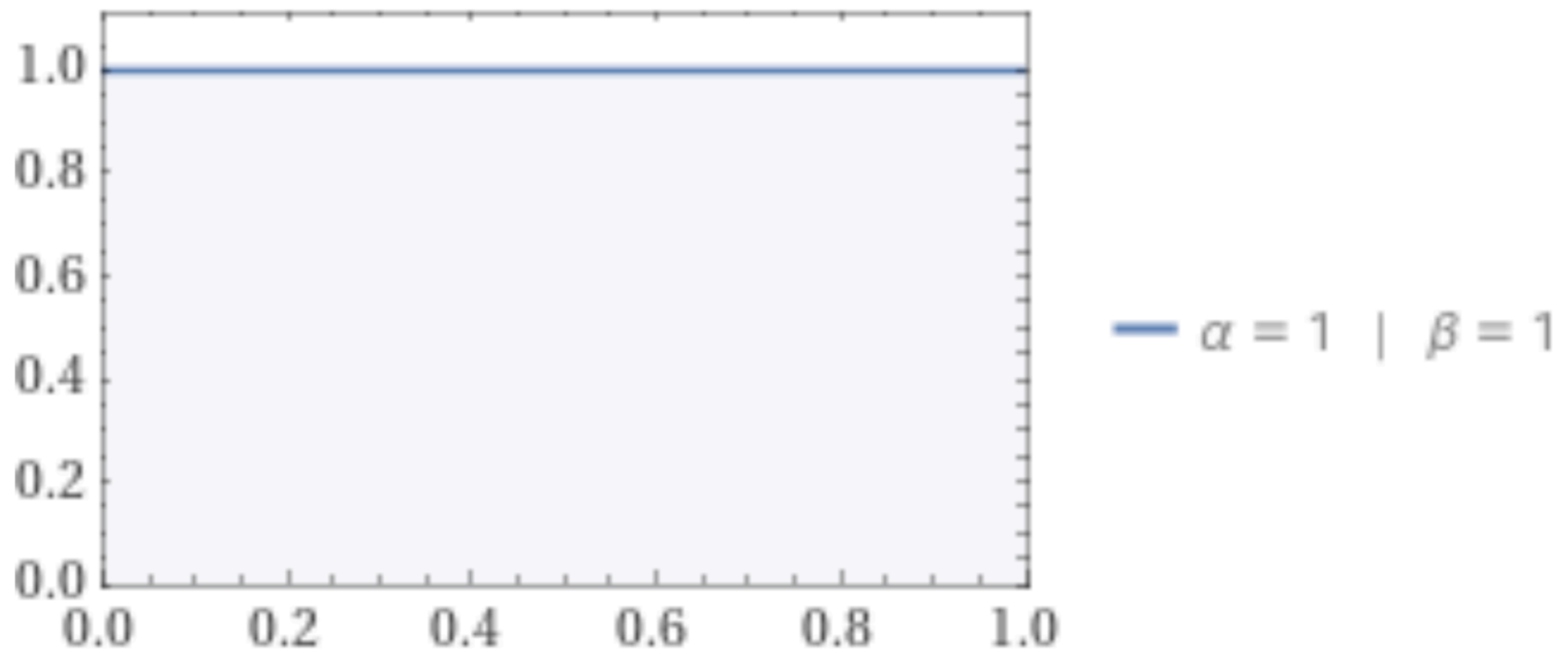
$$= \frac{\theta^{\alpha+\text{Pos}-1}(1-\theta)^{\beta+\text{Neg}-1}}{B(\alpha + \text{Pos}, \beta + \text{Neg})} = p_{beta}(\theta \mid \alpha + \text{Pos}, \beta + \text{Neg})$$

We are going to need this:

$$p(\theta \mid \mathbf{X}, \alpha, \beta) = p_{beta}(\theta \mid \alpha + \text{Pos}, \beta + \text{Neg})$$

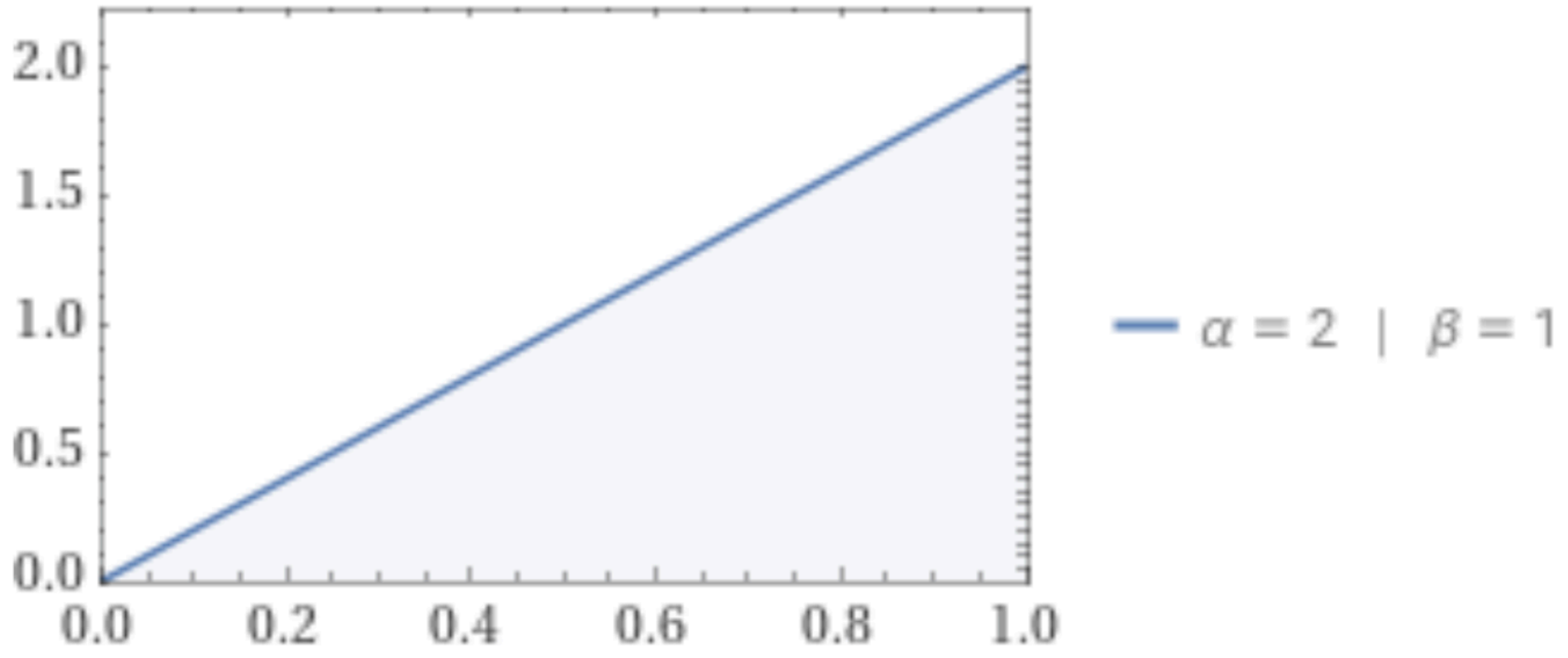
Note: Bayesian Updating (1)

- Suppose we start with $\alpha = 1, \beta = 1$.
- The posterior distribution is $p(\theta) = p_{beta}(\theta \,|\, \alpha = 1, \beta = 1)$



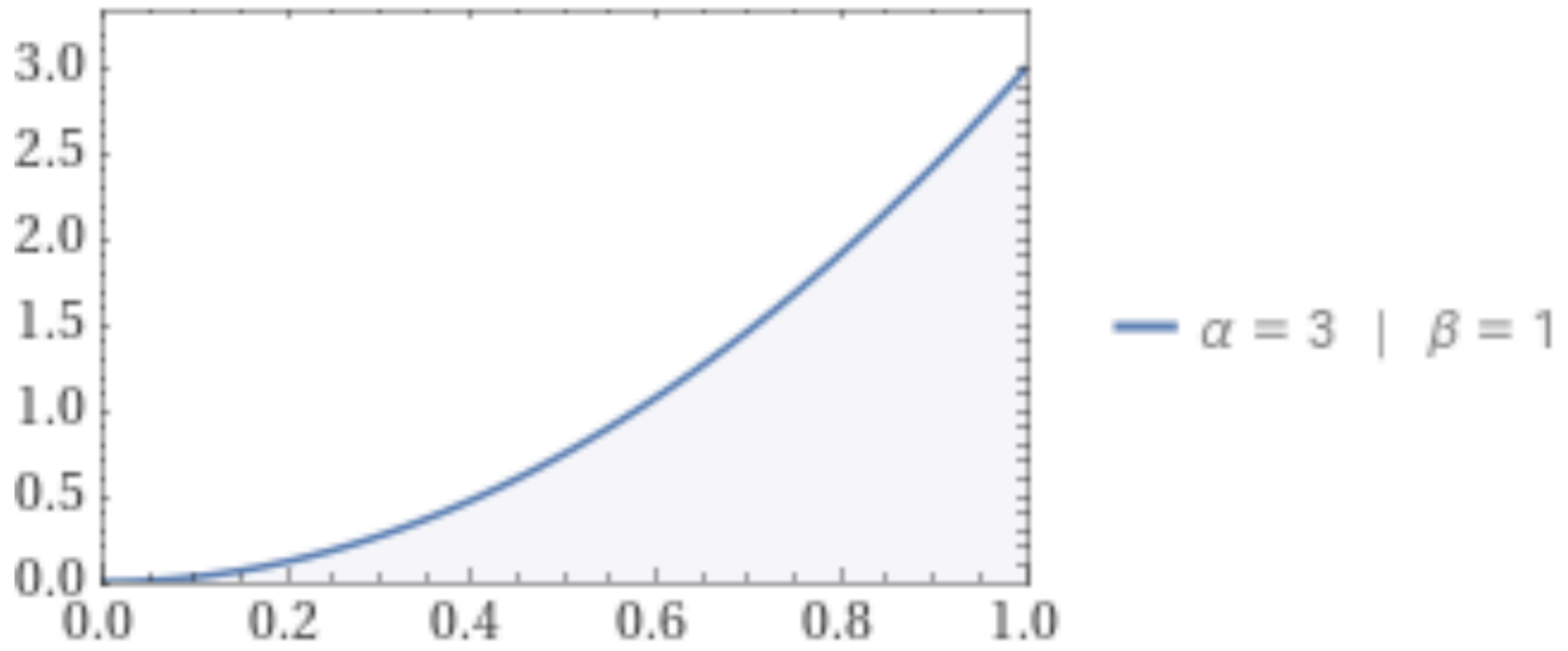
Note: Bayesian Updating (2)

- Suppose we obtain the first sample $x_1 = 1$.
- The posterior distribution is then $p(\theta) = p_{beta}(\theta \,|\, \alpha = 2, \, \beta = 1)$



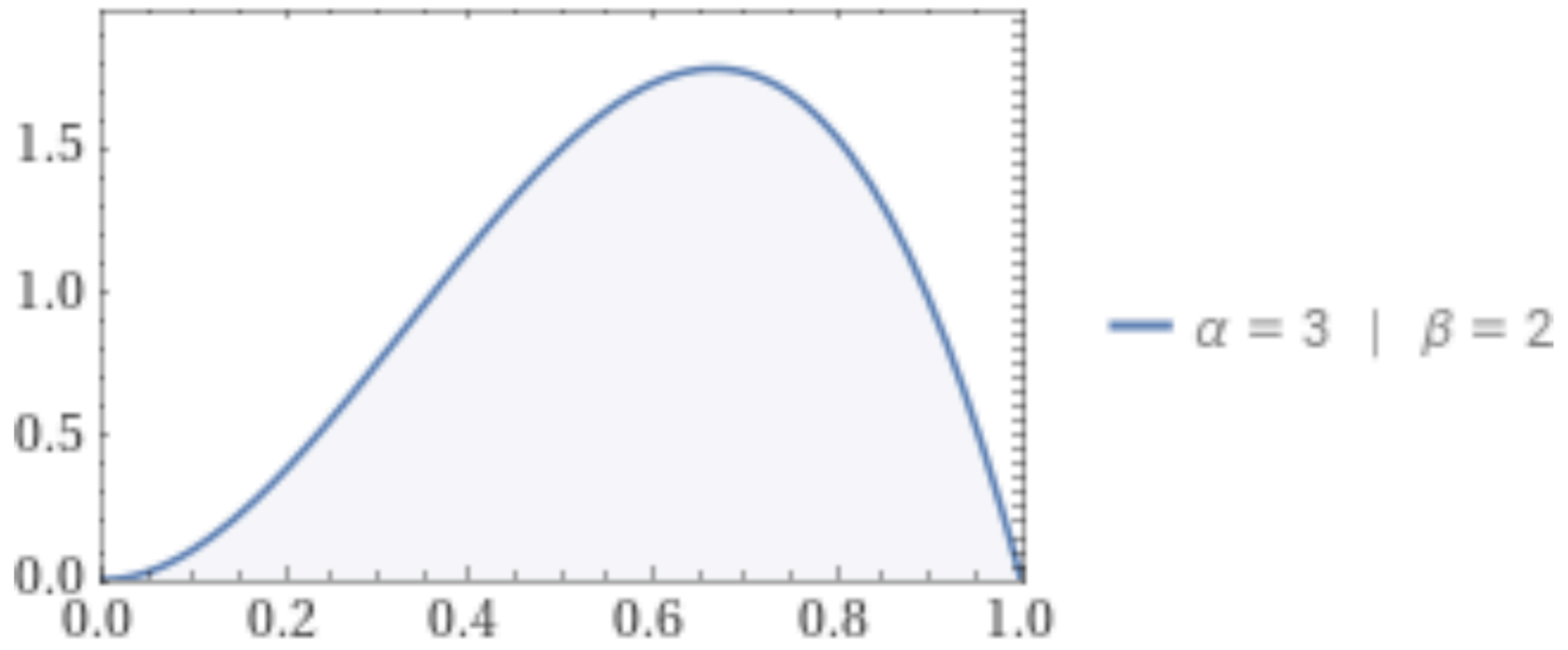
Note: Bayesian Updating (3)

- Suppose we obtain the second sample $x_2 = 1$.
- The posterior distribution is then $p(\theta) = p_{beta}(\theta \,|\, \alpha = 3, \, \beta = 1)$



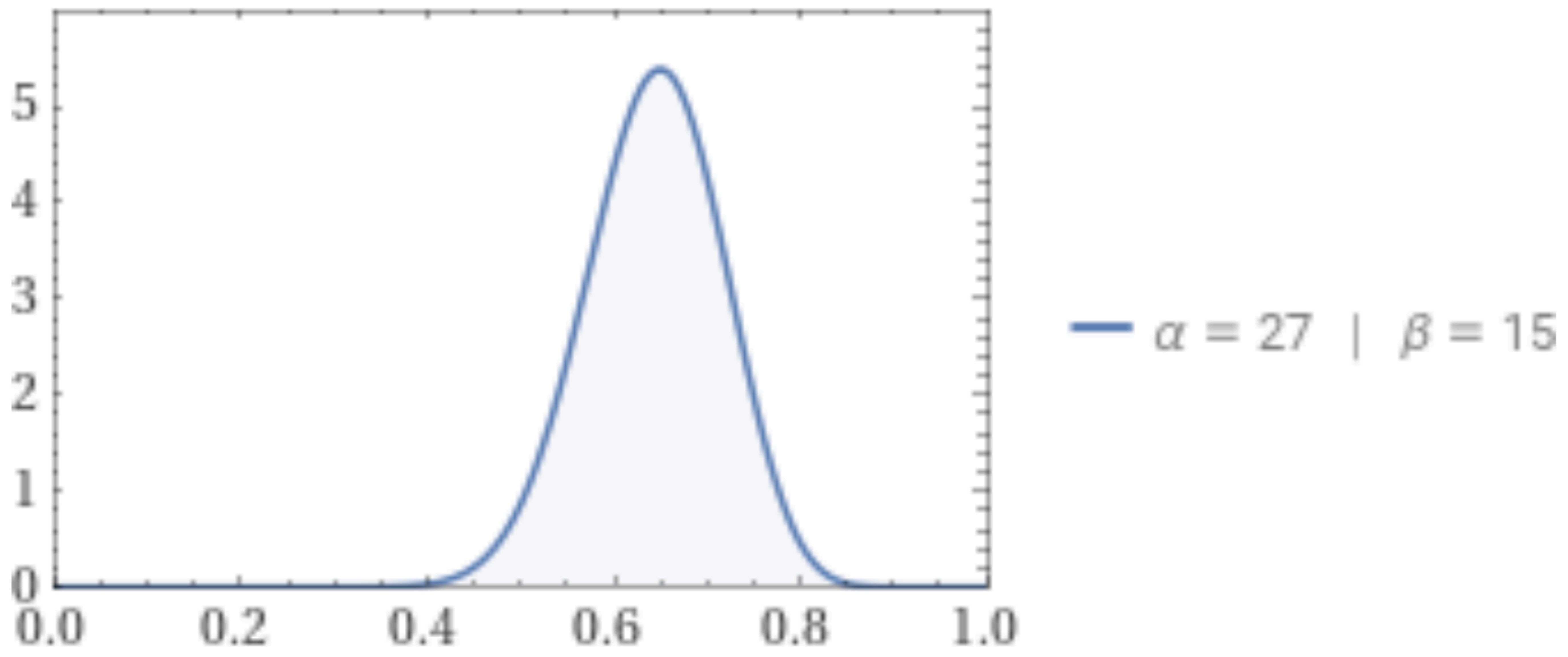
Note: Bayesian Updating (4)

- Suppose we obtain the third sample $x_3 = 0$.
- The posterior distribution is then $p(\theta) = p_{beta}(\theta \,|\, \alpha = 3,\, \beta = 2)$



Note: Bayesian Updating (5)

- Suppose that after 40 coin tosses, we obtained 26 1's and 14 0's..
- Then the posterior distribution is then $p(\theta) = p_{beta}(\theta \mid \alpha = 27, \beta = 15)$



Note 2: “Conjugate Prior”

- What we just observed was an example of “conjugacy”.
- A **conjugate prior** is a prior probability distribution that, when combined with observed data, given a parametric likelihood function, yields a posterior distribution from the same family as the prior distribution.
- Conjugate priors allow for symbolic solutions.

Bandits: Thompson Sampling

Bayesian Bandits

- Formulated in the Bayesian setting.

- They can exploit prior knowledge (if we have it).

- We obtain the posterior distribution of the rewards for each of the arms as $p(r \mid h_{1:t}, \alpha)$ where $h_{1:t}$ is the history of the rewards and α are hyperparameters of the prior.

- **Idea:** We will use the computed posteriors to decide which arm to pull next.

Bernoulli Bandits

- Each arm corresponds to a Bernoulli distribution with an unknown parameter θ_i.

- **Example:** So the rewards are either 0 or 1. For instance, in the example with choosing which ad to show to users, the reward could be 1 if the user clicked on the ad, and 0 otherwise.

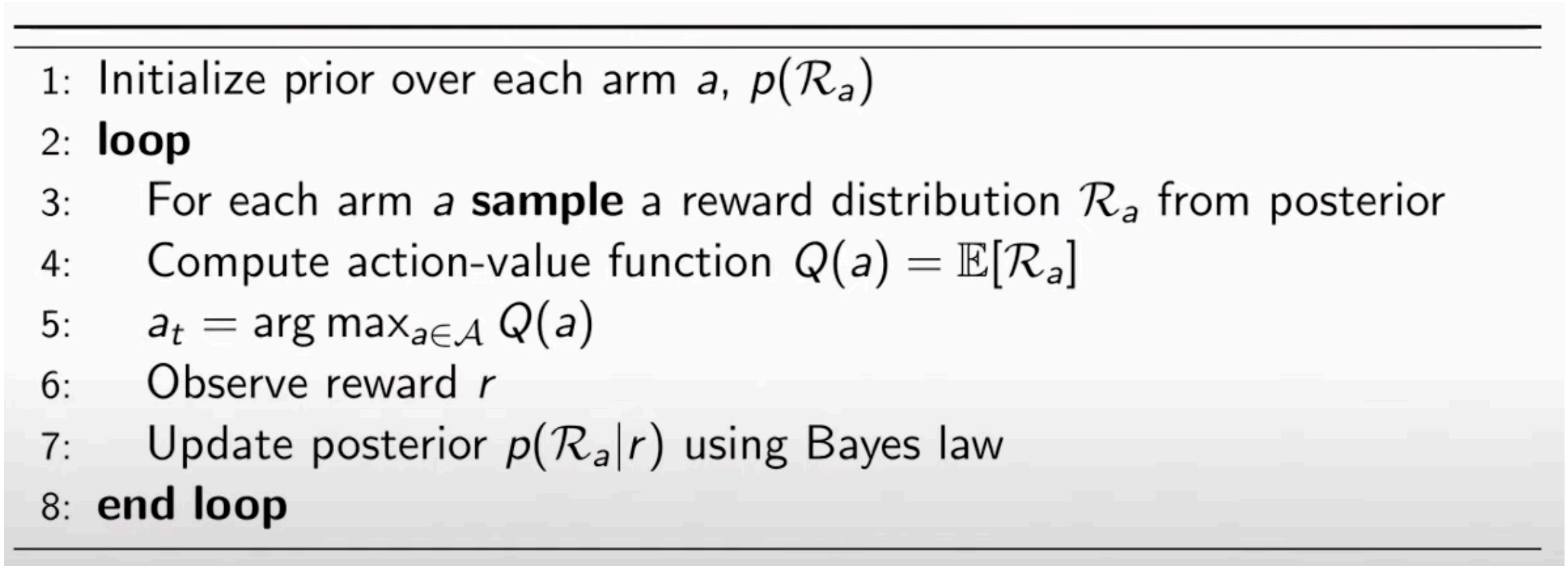
Idea: Probability Matching

- (*An idea that appeared already in 1920's but was mostly forgotten.*)
- **Probability matching** selects the action about which we believe that it is the best action (i.e., here the "belief" is based on the posterior we compute from the observations so far).
- That is:

$$\pi(a \mid h_{1:t}) = P[\forall a' : Q(a) > Q(a') \mid h_{1:t}].$$

- We will see a simple stochastic approach called Thompson Sampling that implements probability matching (computing the above probabilities may be intractable).

Thompson Sampling



```
1: Initialize prior over each arm $a$, $p(\mathcal{R}_a)$
2: loop
3:     For each arm $a$ sample a reward distribution $\mathcal{R}_a$ from posterior
4:     Compute action-value function $Q(a) = \mathbb{E}[\mathcal{R}_a]$
5:     $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$
6:     Observe reward $r$
7:     Update posterior $p(\mathcal{R}_a|r)$ using Bayes law
8: end loop
```

Credit: Prof. Emma Bruskill's Reinforcement learning course

Thompson Sampling “Is” Probability Matching

The following holds for the policy obtained in Thompson sampling:

$$\pi(a \mid h_{1:t}) = P[Q(a) > Q(a') \mid h_{1:t}] = \mathbb{E}\left[\mathbb{I}(a = \arg\max_{a \in \mathcal{A}} Q(a))\right]$$

Here, $Q(a)$ is the sampled action-value (i.e., the parameters of this distribution are sampled from the current posteriors that we have for each $a \in A$).

Thompson Sampling for Bernoulli Bandits

```
1. for t = 1, 2, …, do:
    # sample the Bernoulli model
    A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
    B. end for

    # select and apply the action
    D. $x_t := \mathbf{argmax}_k\ \hat{\theta}_k$
    E. Apply the action $x_t$ and observe the reward $r_t$
    # update the distributions
    F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

Guarantees

There are also guarantees on regret of Thompson sampling (k is a constant):

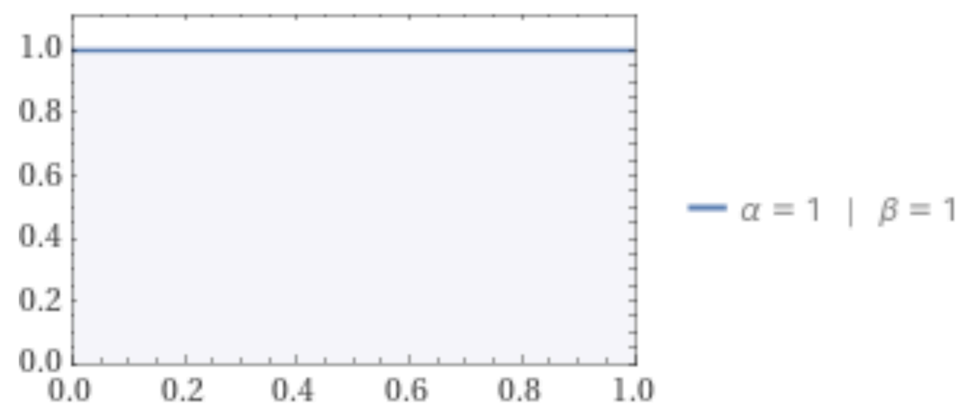
$$\mathbb{E}[R(T)] \leq \mathcal{O}(\sqrt{kT \log T})$$

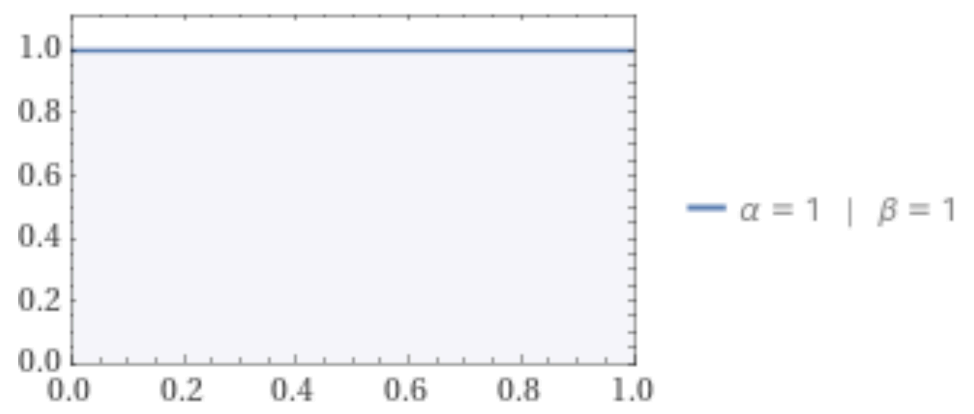
Example (1)

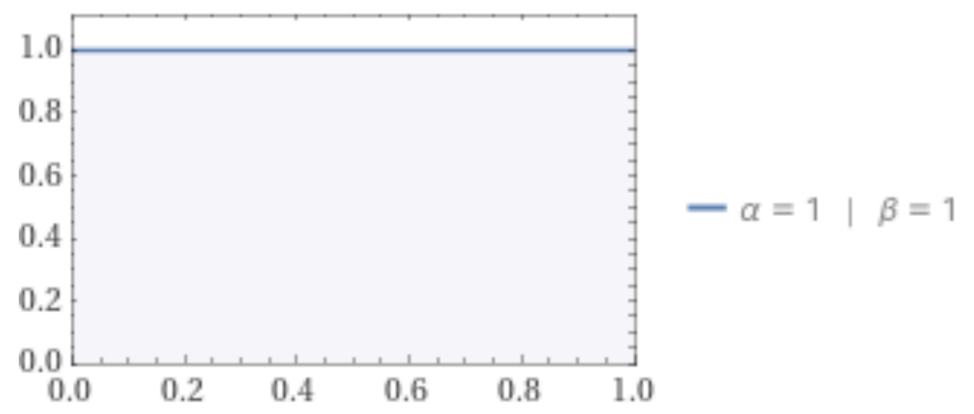
1

2

3







Thompson Sampling for Bernoulli Bandits

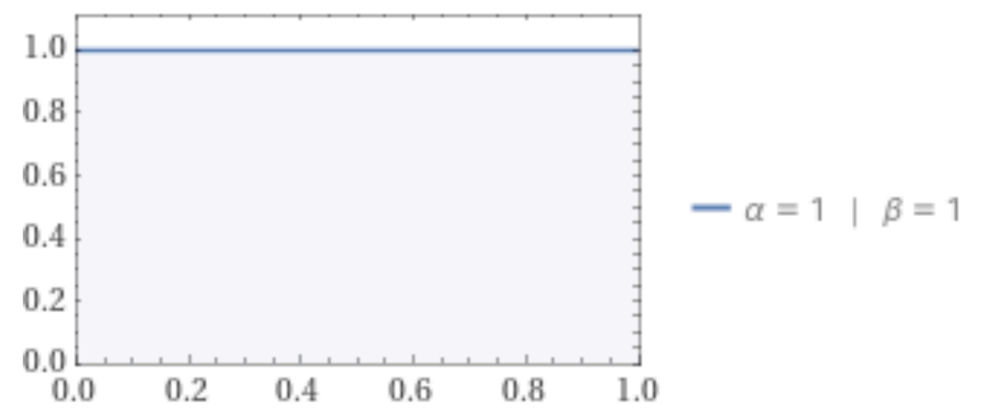
```
1. for t = 1, 2, …, do:
     # sample the Bernoulli model
     A. for k = 1, …, num_arms do:
          • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
     B. end for

     # select and apply the action
     D. $x_t := \mathbf{argmax}_k \hat{\theta}_k$
     E. Apply the action $x_t$ and observe the reward $r_t$
     # update the distributions
     F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

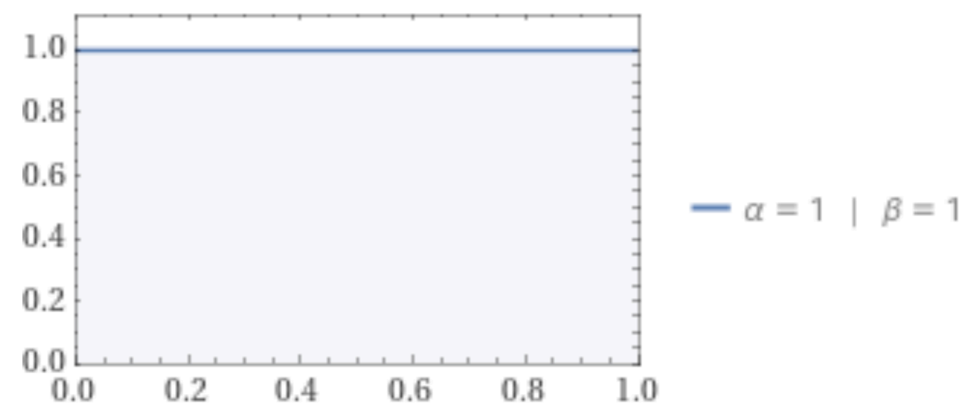
Example (2)

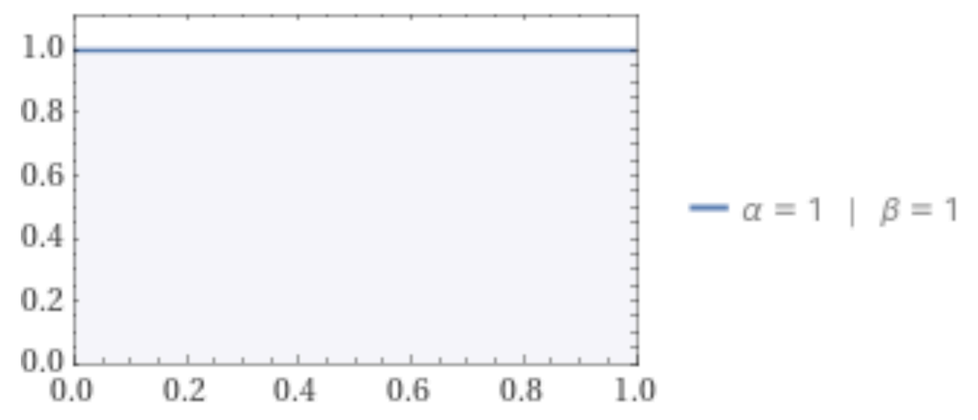
1

2

3







Sample θ_i:

$\theta_1 = 0.4$

$\theta_2 = 0.45$

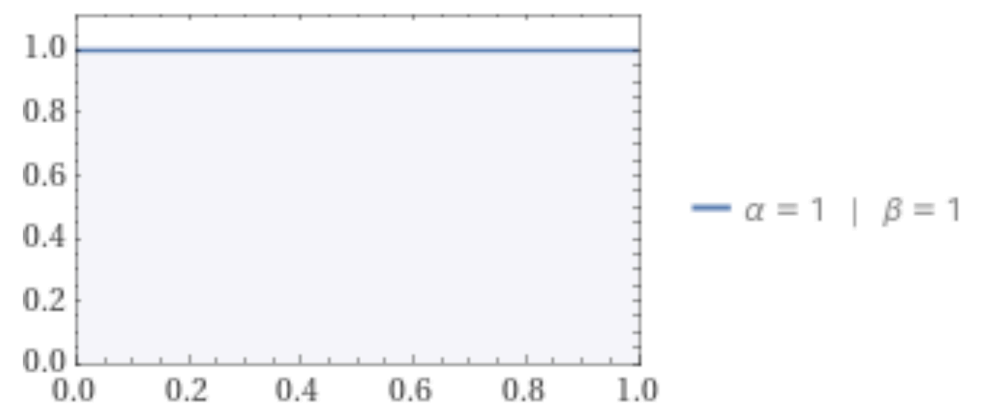
$\theta_3 = 0.75$

Thompson Sampling for Bernoulli Bandits

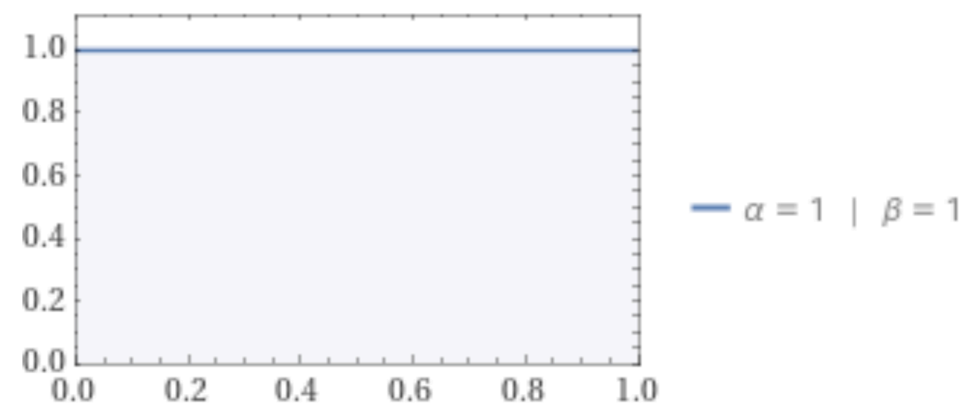
```
1. for t = 1, 2, …, do:
   # sample the Bernoulli model
   A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \textbf{beta}(\alpha_k, \beta_k)$
   B. end for

   # select and apply the action
   D. $x_t := \textbf{argmax}_k \; \hat{\theta}_k$
   E. Apply the action $x_t$ and observe the reward $r_t$
   # update the distributions
   F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

Example (3)

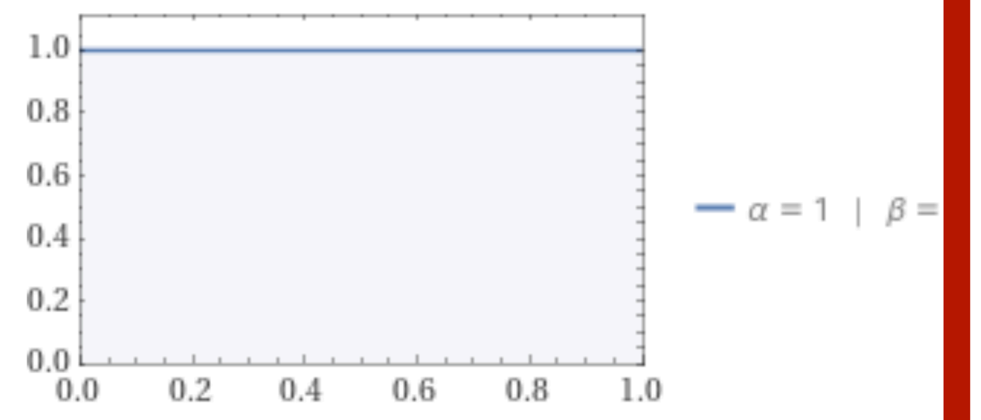
1



2



3



Argmax: $\theta_1 = 0.4$ $\theta_2 = 0.45$ $\theta_3 = 0.75$

Thompson Sampling for Bernoulli Bandits

```
1. for t = 1, 2, …, do:
   # sample the Bernoulli model
   A. for k = 1, …, num_arms do:
         • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
   B. end for

   # select and apply the action
   D. $x_t := \mathbf{argmax}_k \hat{\theta}_k$
   E. Apply the action $x_t$ and observe the reward $r_t$
   # update the distributions
   F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

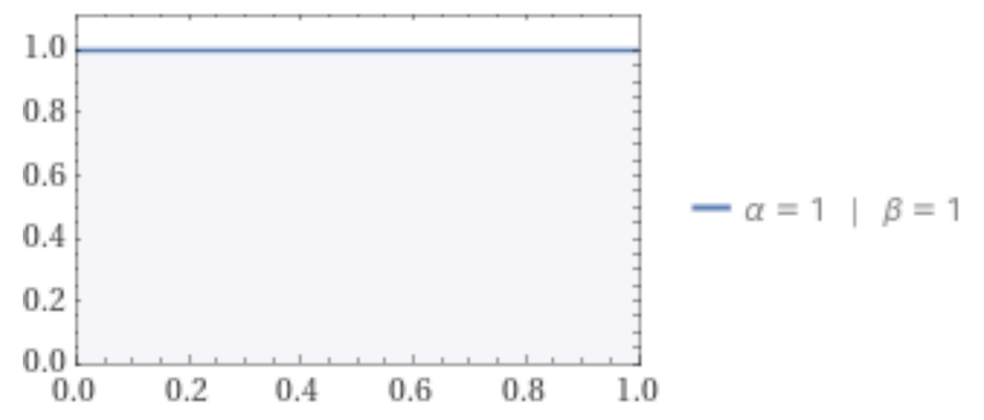
Example (4)

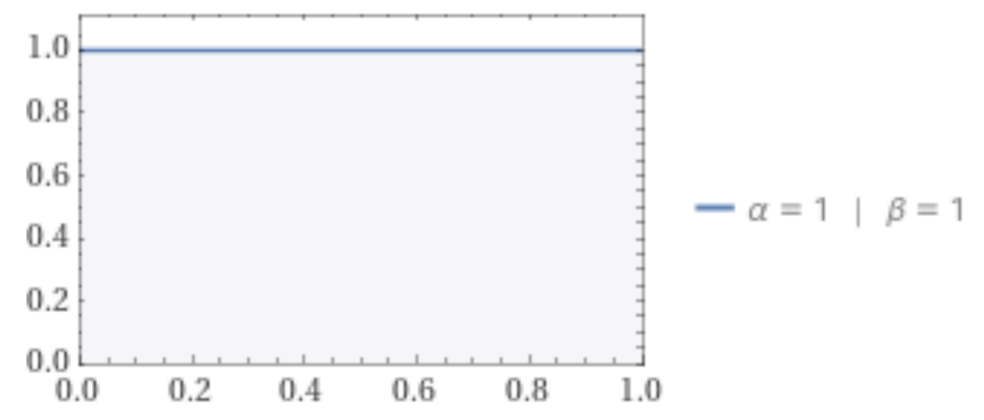
PULL ARM 3

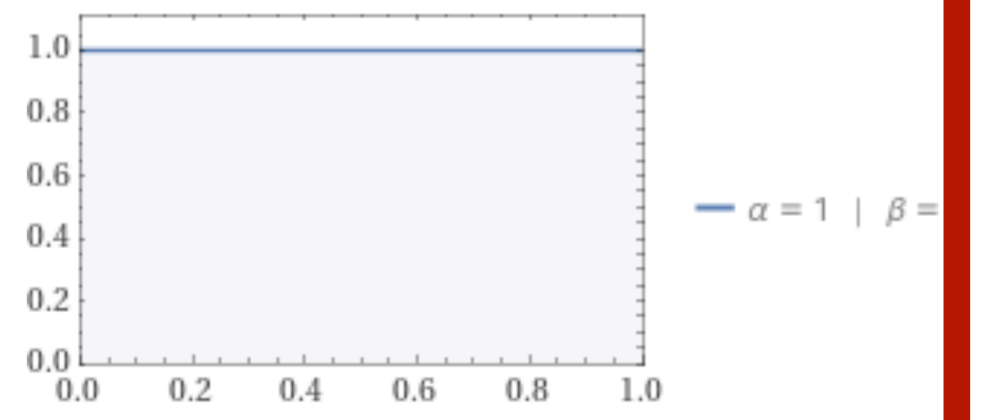
1

2

3







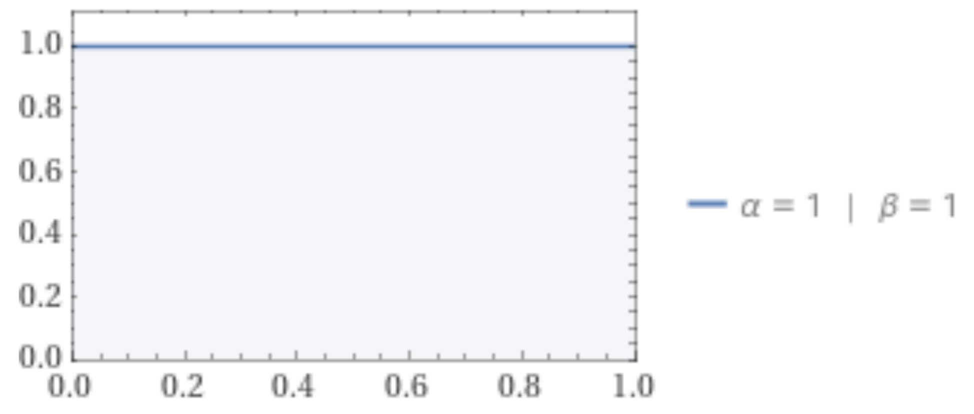
Example (5)

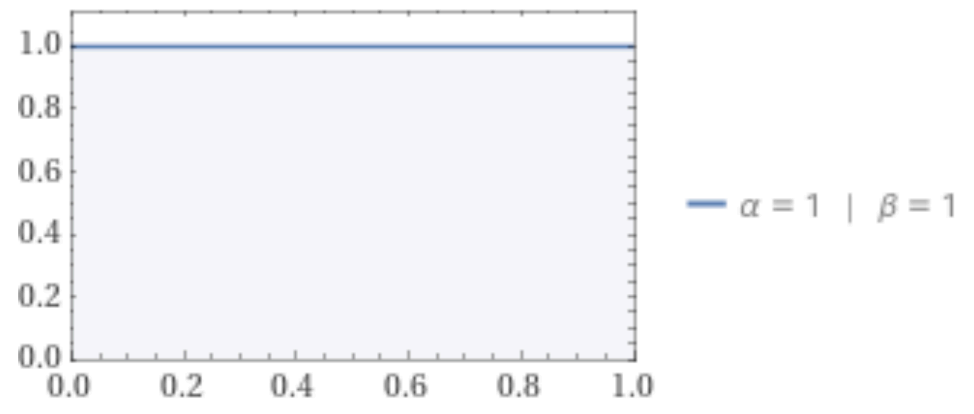
PULL ARM 3

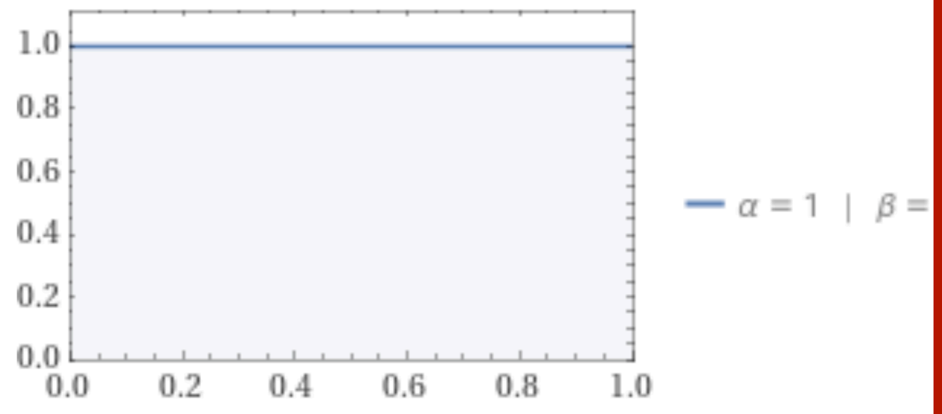
1

2

3







Observed reward $r_1 = 1.$

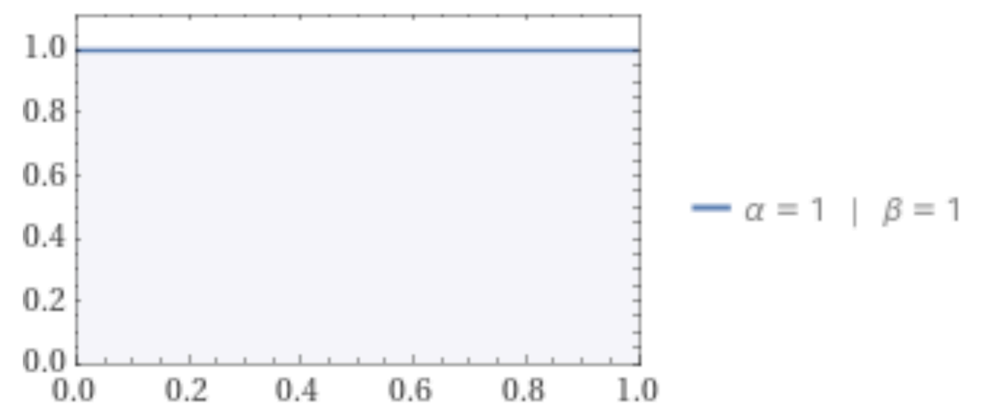
Thompson Sampling for Bernoulli Bandits

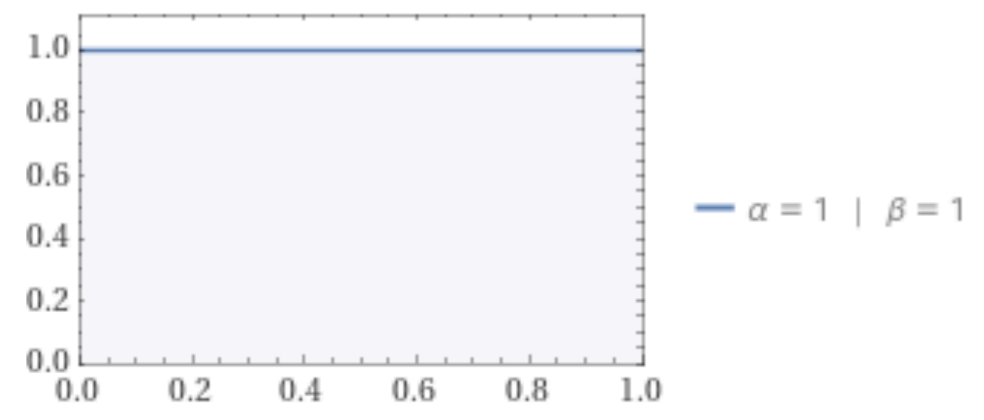
```
1. for t = 1, 2, …, do:
    # sample the Bernoulli model
    A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
    B. end for

    # select and apply the action
    D. $x_t := \mathbf{argmax}_k \hat{\theta}_k$
    E. Apply the action $x_t$ and observe the reward $r_t$
    # update the distributions
    F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

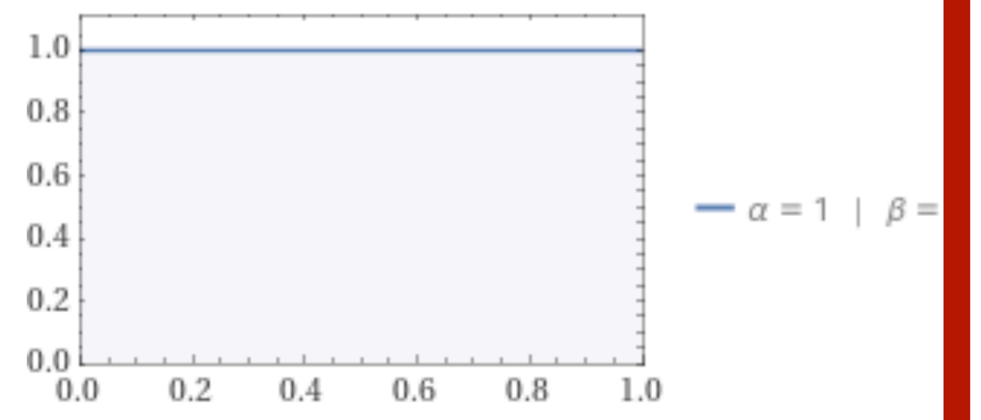
Example (6)

PULL ARM 3

1 **2** **3**



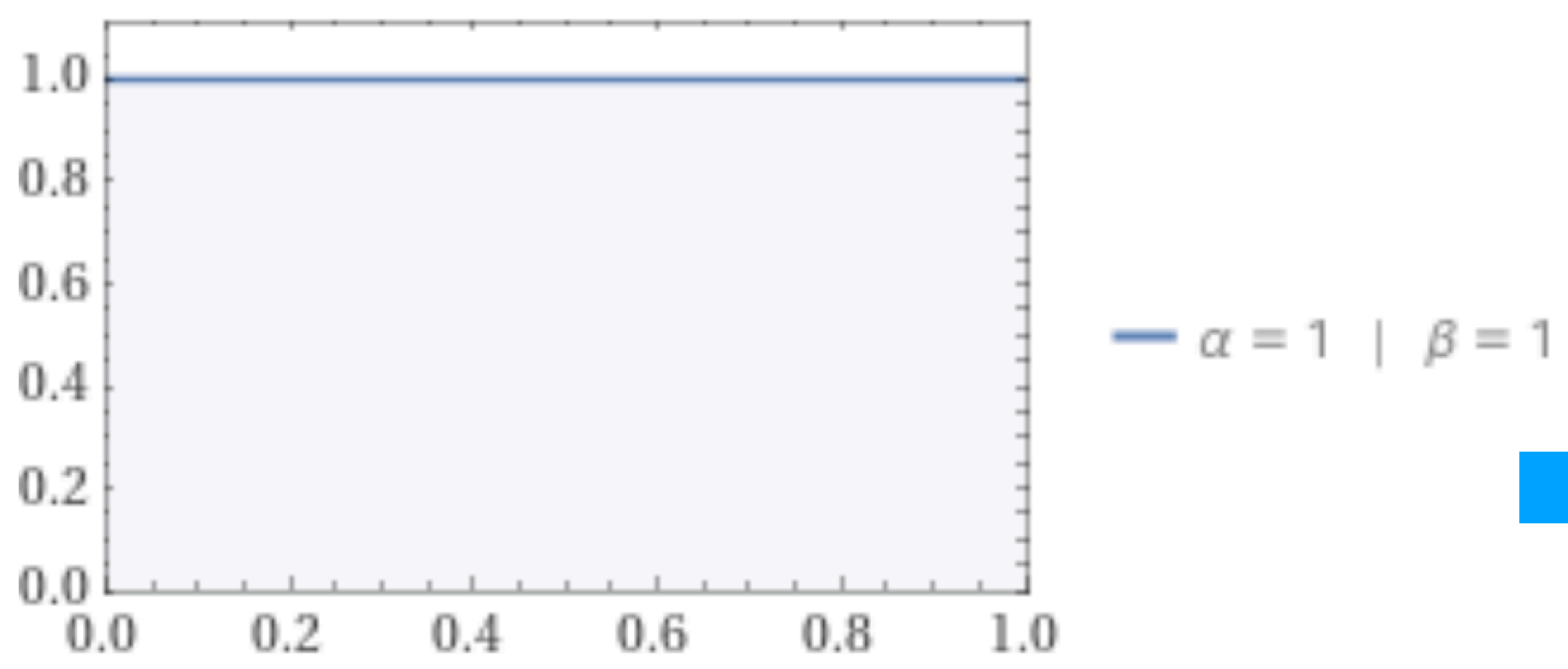


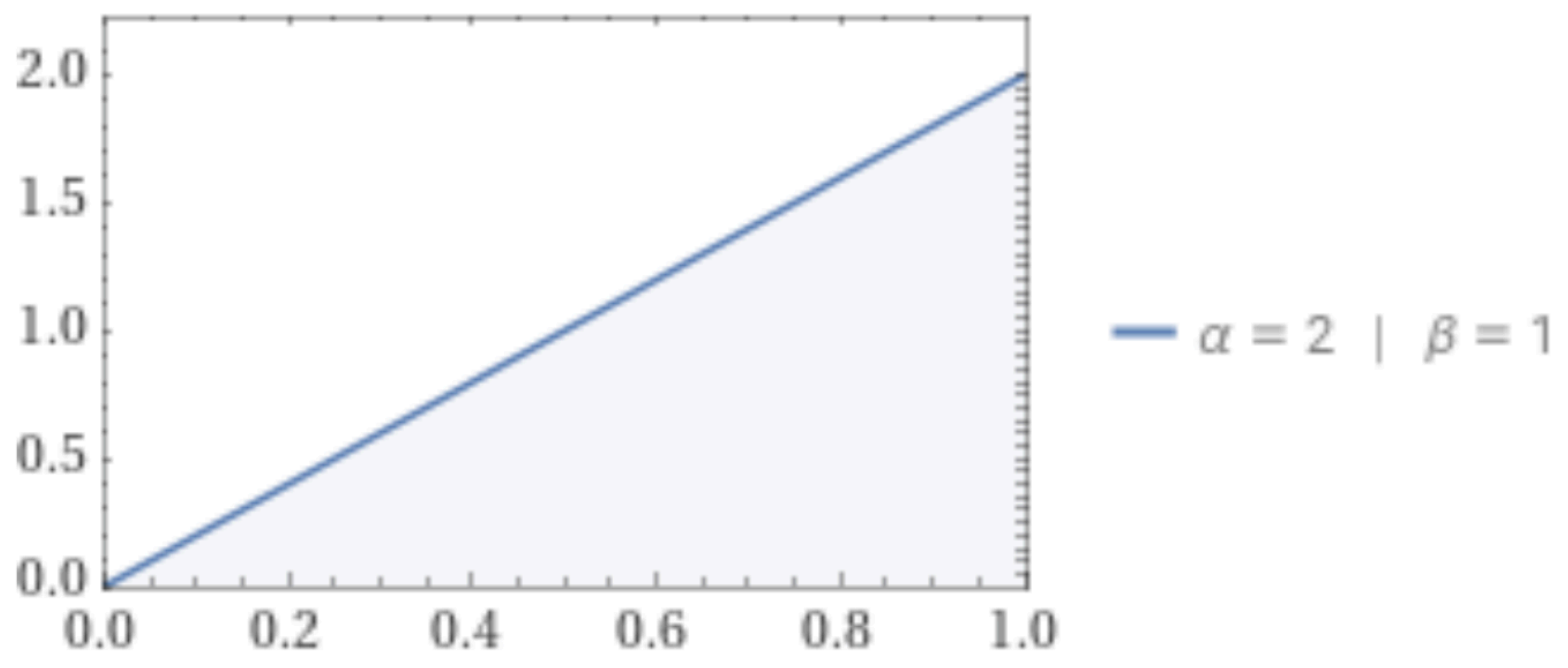


Observed reward $r_1 = 1$.

Update $\alpha_3 = 1 + 1 = 2$, $\beta_3 = 1$.

Example (7)





Update $\alpha_3 = 1 + 1 = 2$, $\beta_3 = 1$.

Thompson Sampling for Bernoulli Bandits

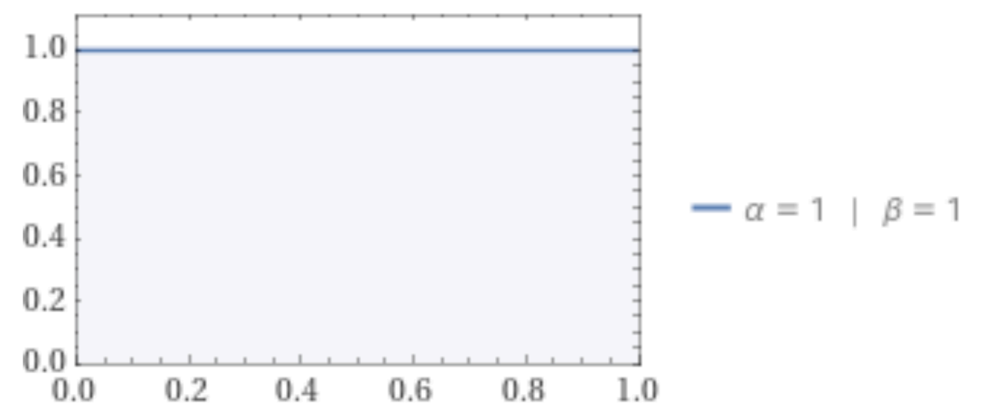
```
1. for t = 1, 2, …, do:
    # sample the Bernoulli model
    A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
    B. end for

    # select and apply the action
    D. $x_t := \mathbf{argmax}_k \hat{\theta}_k$
    E. Apply the action $x_t$ and observe the reward $r_t$
    # update the distributions
    F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

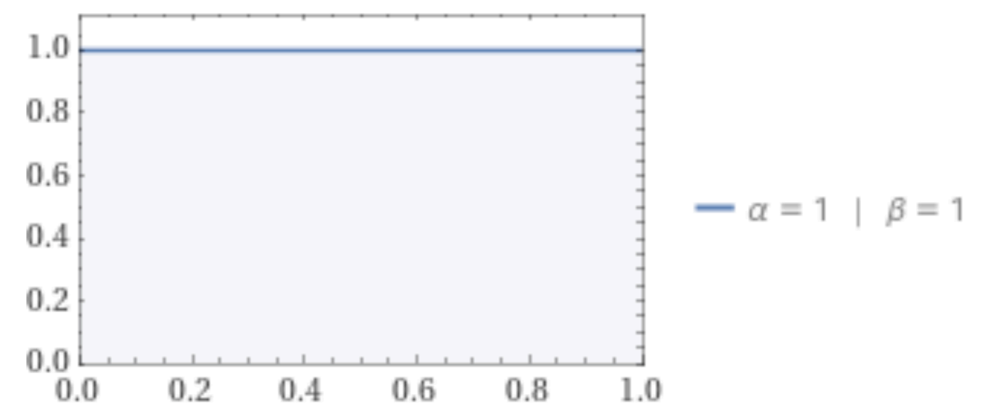
Example (8)

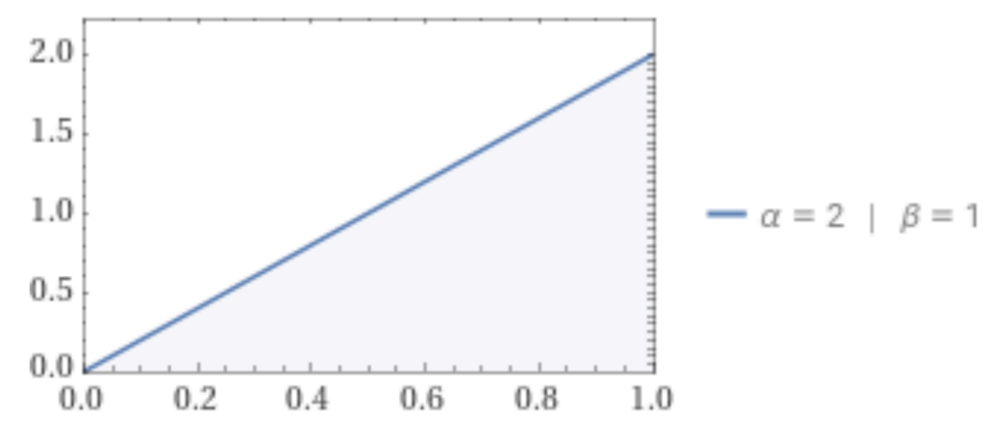
1

2

3







Sample θ_i:

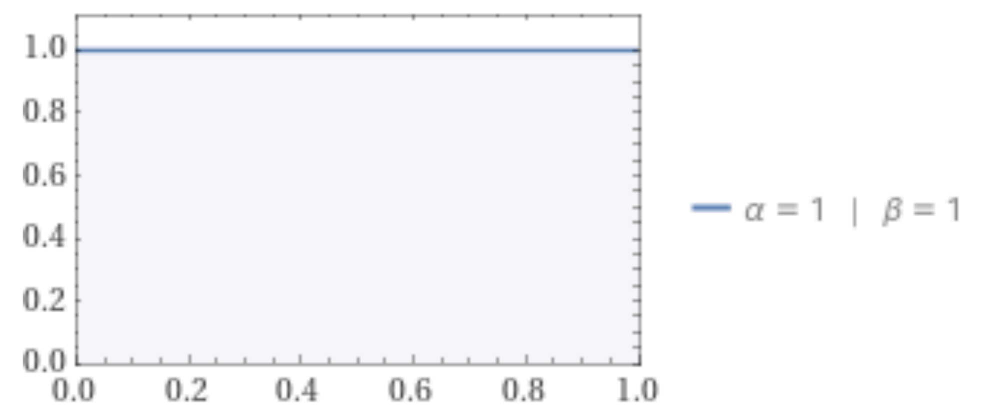
$\theta_1 = 0.3$ $\theta_2 = 0.8$ $\theta_3 = 0.69$

Thompson Sampling for Bernoulli Bandits

```
1. for t = 1, 2, …, do:
   # sample the Bernoulli model
   A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \textbf{beta}(\alpha_k, \beta_k)$
   B. end for

   # select and apply the action
   D. $x_t := \textbf{argmax}_k \, \hat{\theta}_k$
   E. Apply the action $x_t$ and observe the reward $r_t$
   # update the distributions
   F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

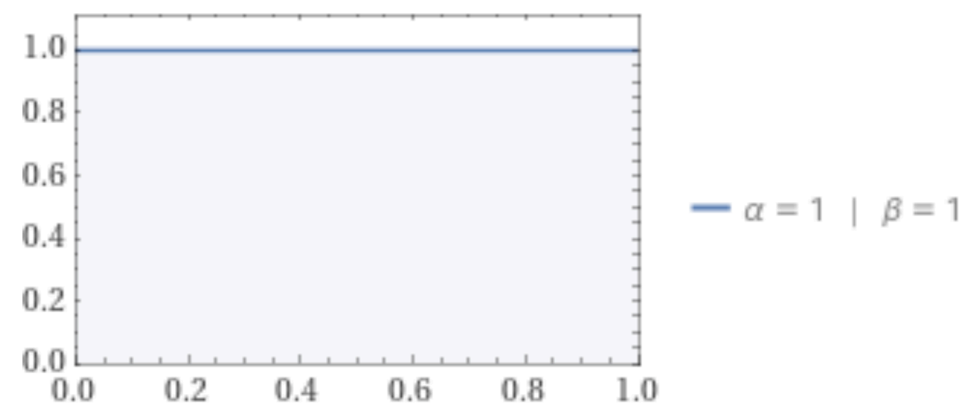
Example (9)

1



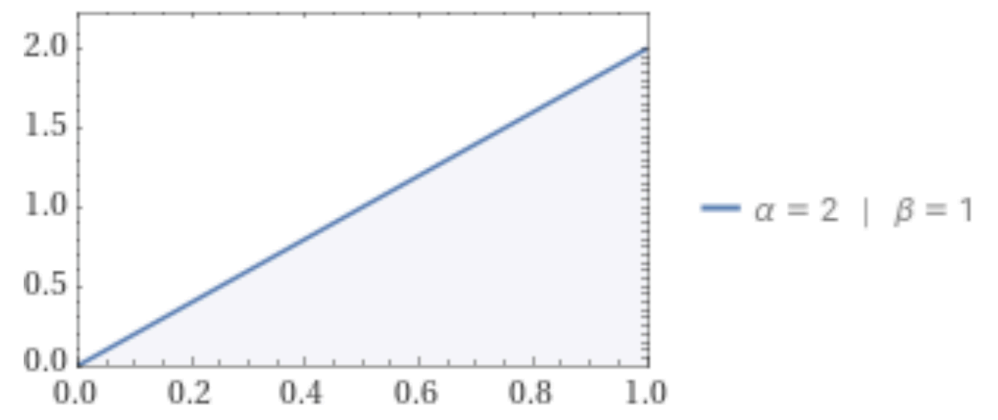
$\theta_1 = 0.3$

2



$\theta_2 = 0.8$

3



$\theta_3 = 0.69$

Sample θ_i:

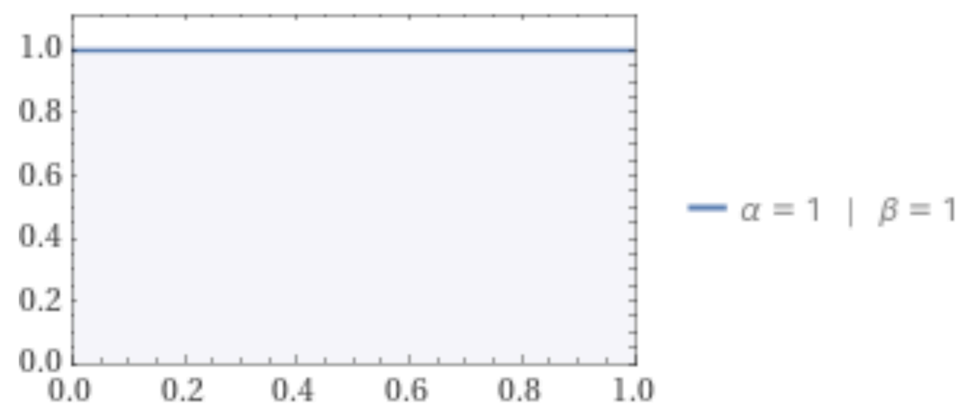
Thompson Sampling for Bernoulli Bandits

```
1. for t = 1, 2, …, do:
    # sample the Bernoulli model
    A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
    B. end for

    # select and apply the action
    D. $x_t := \mathbf{argmax}_k \hat{\theta}_k$
    E. Apply the action $x_t$ and observe the reward $r_t$
    # update the distributions
    F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

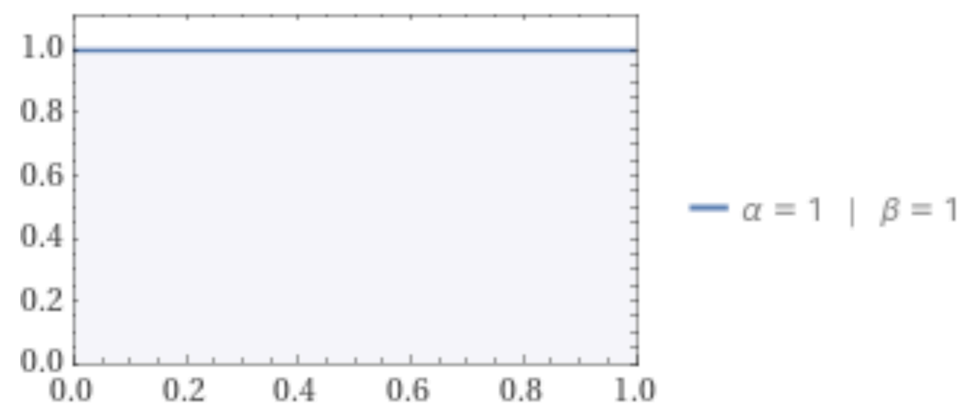
Example (10)

PULL ARM 2

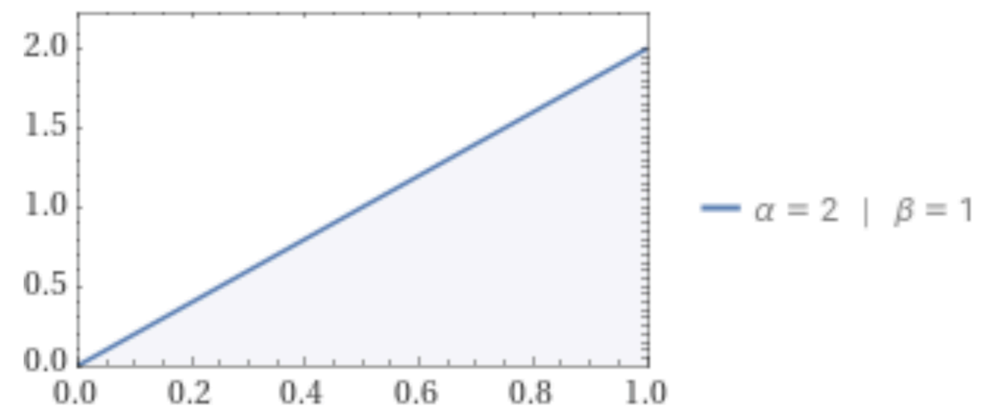
1



2



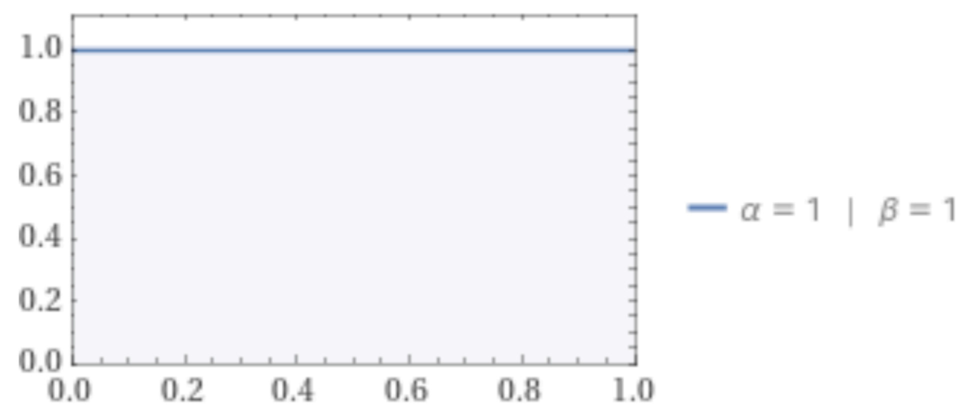
3



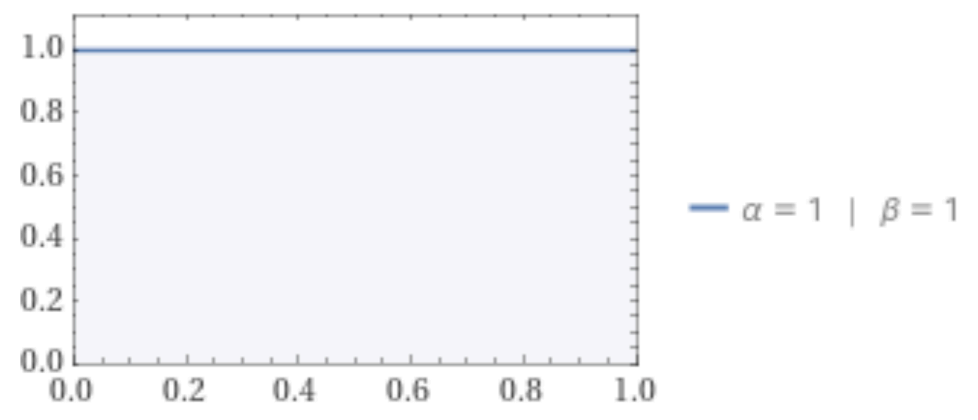
Example (11)

PULL ARM 2

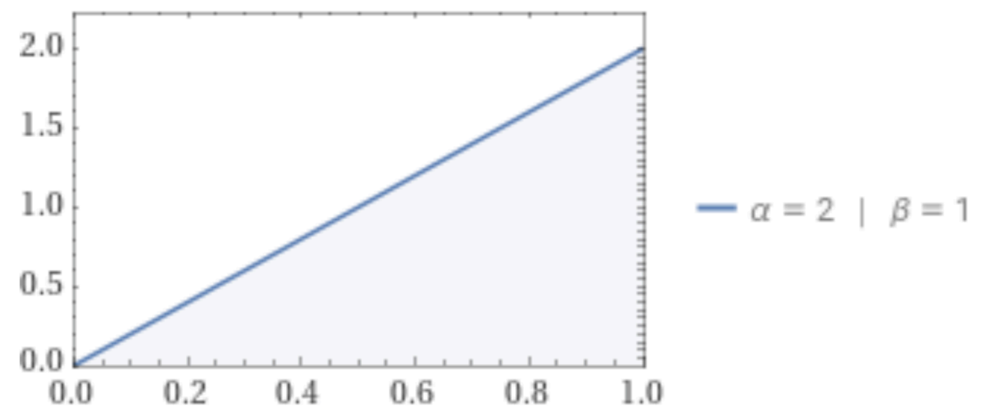
1



2



3



Observed reward $r_1 = 0$.

Thompson Sampling for Bernoulli Bandits

```
1. for t = 1, 2, …, do:
   # sample the Bernoulli model
   A. for k = 1, …, num_arms do:
        • Sample $\hat{\theta}_k = \mathbf{beta}(\alpha_k, \beta_k)$
   B. end for

   # select and apply the action
   D. $x_t := \mathbf{argmax}_k \, \hat{\theta}_k$
   E. Apply the action $x_t$ and observe the reward $r_t$
   # update the distributions
   F. $(\alpha_k, \beta_k) := (\alpha_k + r_t, \beta_k + 1 - r_t)$
2. end for
```

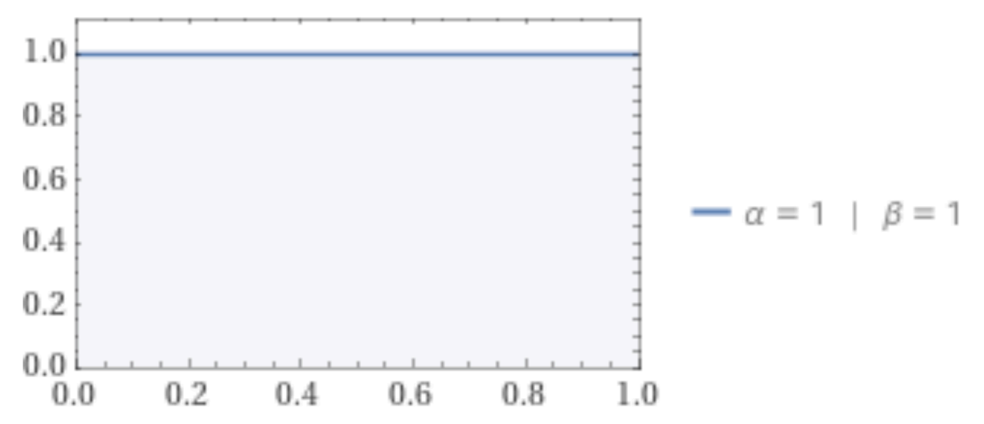
Example (12)

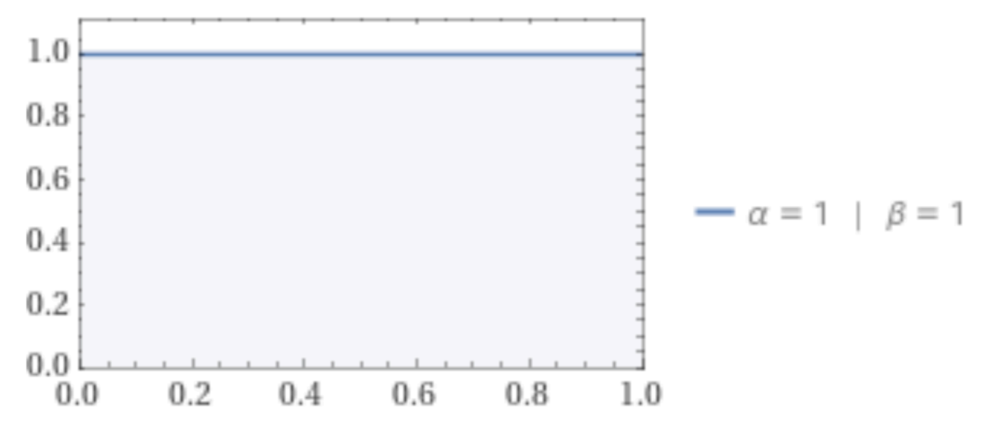
PULL ARM 2

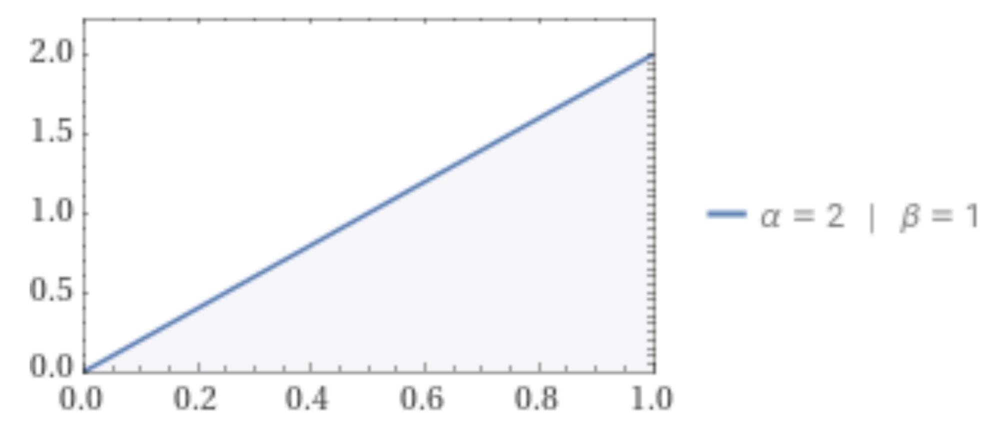
1

2

3



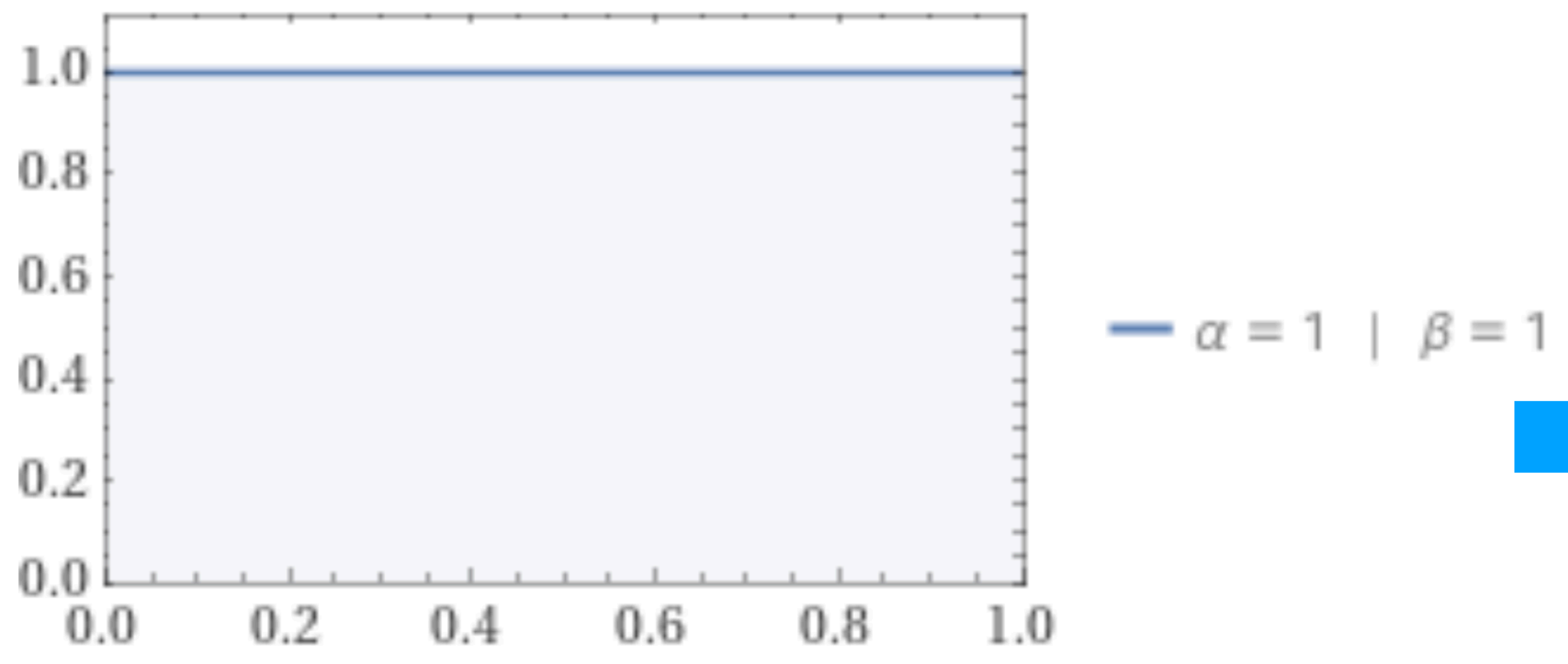


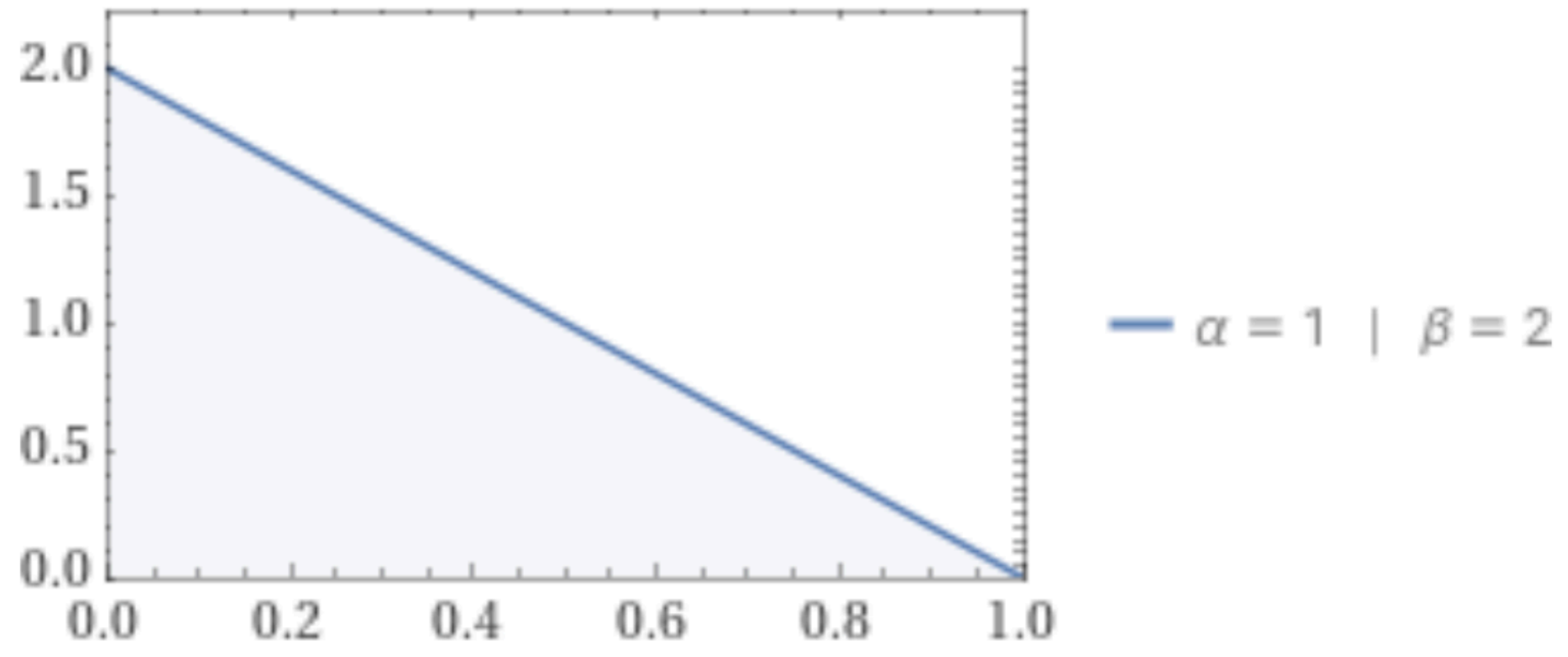


Observed reward $r_1 = 0$.

Update $\alpha_2 = 1, \beta_2 = 1 + 1 = 2$.

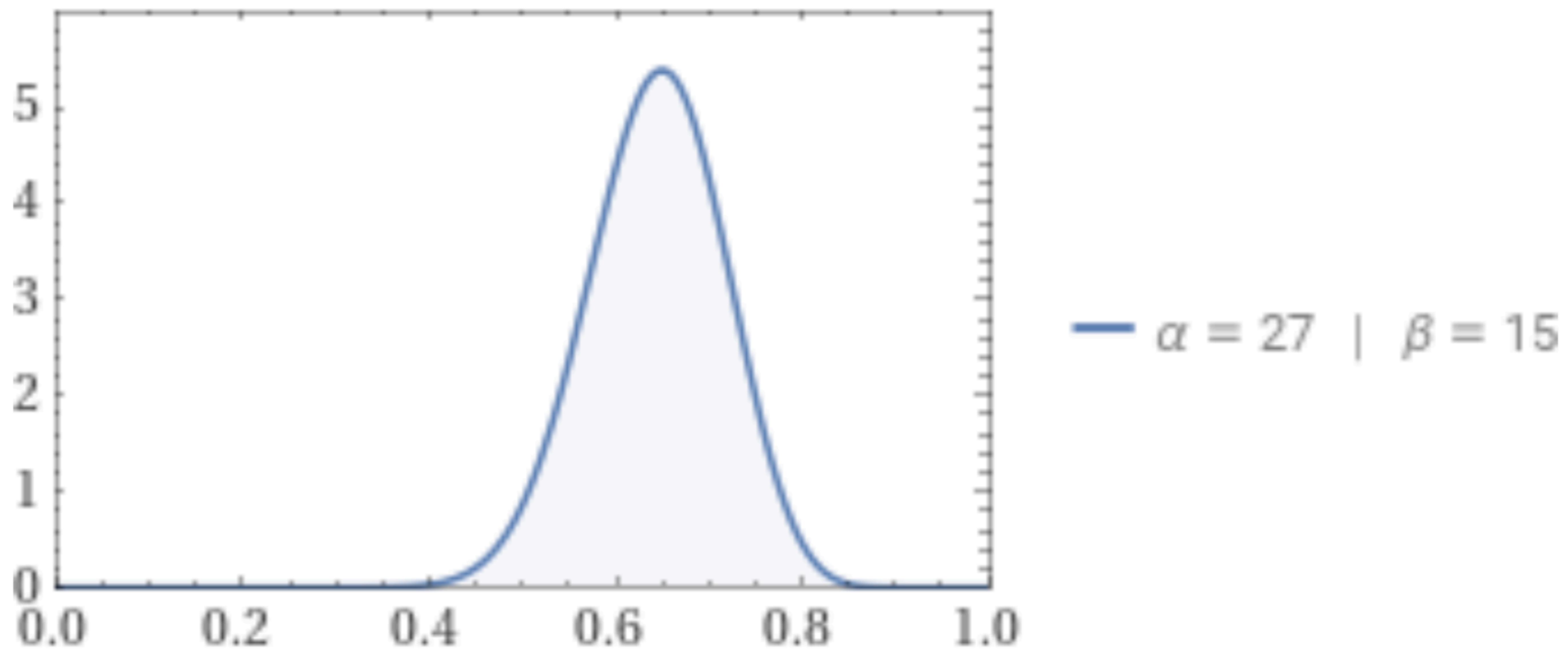
Example (12)





Update $\alpha_2 = 1, \beta_2 = 1 + 1 = 2$**.**

And so on…



Conclusions - What We Did Not Cover in the RL Part

What We Did Not Cover

- **A lot…**
- **Some examples:**
  - **Contextual bandits** - e.g., when Facebook serves you advertisement, they use personalization - context.
  - **Policy gradients** - A class of deep RL methods where policies are also parametrized by neural networks.
  - **Proofs** … of almost anything.

Some Examples of RL Applications

Atari Games (Deep Q-Learning): https://www.youtube.com/watch?v=V1eYniJ0Rnk

Open AI Plays Hide-And-Seek: https://www.youtube.com/watch?v=Lu56xVlZ40M

DeepMind's AlphaStar - Starcraft: https://www.youtube.com/watch?v=jtlrWblOyP4

There are many other applications of reinforcement learning (e.g., reinforcement learning from human feedback for LLMs…)