Conjugate Prior Explained

With examples & proofs

Aerin Kim Jan 8 · 4 min read ★

1. What is Prior?

Prior probability is **the probability of an event before we see the data**.
In Bayesian Inference, the prior is our guess about the probability based on what we know now, before new data becomes available.

2. What is Conjugate Prior?

Conjugate prior just can not be understood without knowing Bayesian inference.

Bayesian Inference — Intuition and Implementation
The art of Bayesian Inference lies in how you implement it...

For the rest of the blog, I'll assume you know the concepts of prior, sampling and posterior.

Conjugate prior in essence

For some likelihood functions, if you choose a certain prior, the posterior ends up being in the same distribution as the prior. Such a prior then is called a Conjugate Prior.

It is always best understood through examples. Below is the code to **calculate the posterior of the binomial likelihood. θ** is the probability of success and our goal is to pick the **θ that maximizes the posterior probability.**

```
import numpy as np
import scipy.stats as stats

success_prob = 0.3
data = np.random.binomial(n=1, p=success_prob, size=1000) # sucess is 1, failure is 0.

# Domain θ
theta_range = np.linspace(0, 1, 1000)

# Prior P(θ)
a = 2
b = 8
theta_range_e = theta_range + 0.0001
prior = stats.beta.cdf(x = theta_range_e, a=a, b=b) - stats.beta.cdf(x = theta_range, a

# The sampling dist. aka Likelihood P(X|θ)
likelihood = stats.binom.pmf(k = np.sum(data), n = len(data), p = theta_range)

# Posterior
posterior = likelihood * prior
normalized_posterior = posterior / np.sum(posterior)
```

conjugate_prior.py hosted with ❤️ by **GitHub** **view raw**

A question to you: Is there anything that concerns you in the code block above?

There are two things that make the posterior calculation expensive.

First, we are computing the posterior for every single θ.

Why do we have to calculate the posterior for thousands of thetas? Because you are normalizing the posterior (line 21). Even if you choose not to normalize the posterior, the end goal is to find the **maximum** of the posteriors (Maximum a posteriori). In order to find the maximum in a vanilla way, we need to consider every candidate — the likelihood $\mathbf{P(X|\theta)}$ for every $\boldsymbol{\theta}$**.**

Second, if there is no closed-form formula of the posterior distribution, we have to find the maximum by numerical optimization, such as gradient descent or newtons method.

3. How does the Conjugate Prior help?

When you know that your prior is a conjugate prior, you can skip the `posterior = likelihood * prior` computation. Furthermore, if your prior distribution has a closed-form form expression, you already know what the maximum posterior is going to be.

In the example above, the beta distribution is a conjugate prior to the binomial likelihood. What does this mean? It means **during the modeling phase, we already know the posterior will also be a beta distribution.** Therefore, after carrying out more experiments, **you can compute the posterior simply by adding the number of acceptances and rejections to the existing parameters α, β respectively**, instead of multiplying the likelihood with the prior distribution**.** This is very convenient! (Proof in the next section.)

As a data/ML scientist, your model is never complete. You have to update your model as more data come in (and that's why we use Bayesian Inference).
As you saw, the computations in Bayesian Inference can be heavy or sometimes even intractable. However, if we could use the closed-form formula of the conjugate prior, the computation becomes very light.

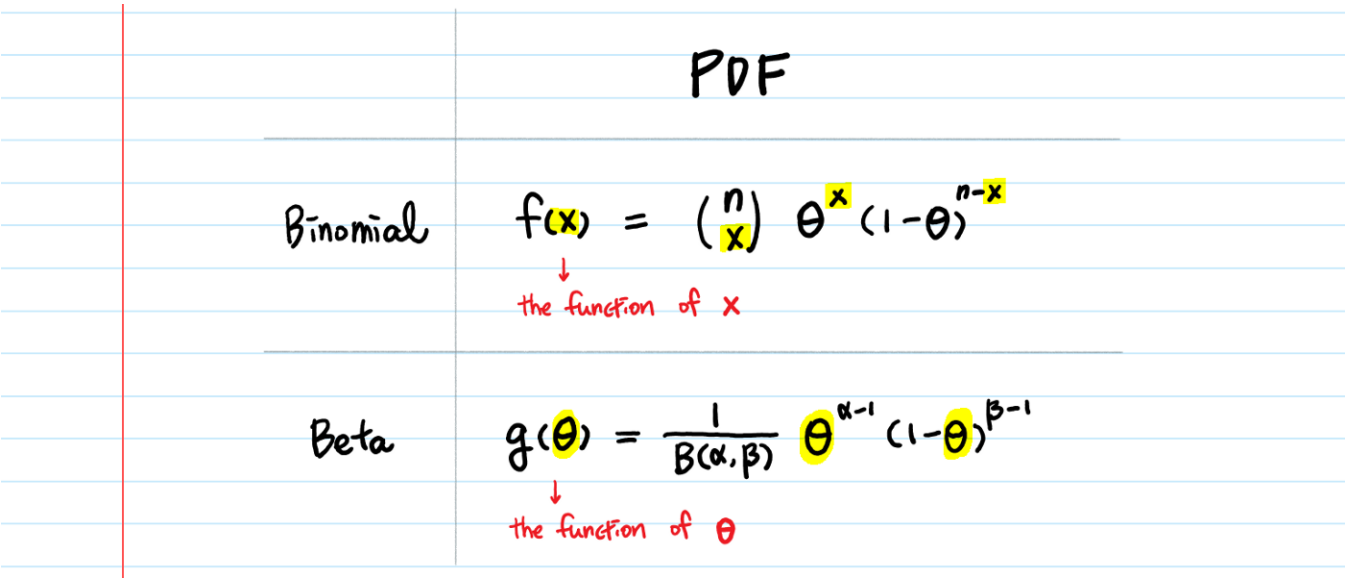
4. Proof — Why is a Beta distribution a conjugate prior to Binomial

likelihood?

When we use the Beta distribution as a prior, a posterior of binomial likelihood will also follow the beta distribution.

Show Beta begets Beta.

What do the PDFs of Binomial and Beta look like?

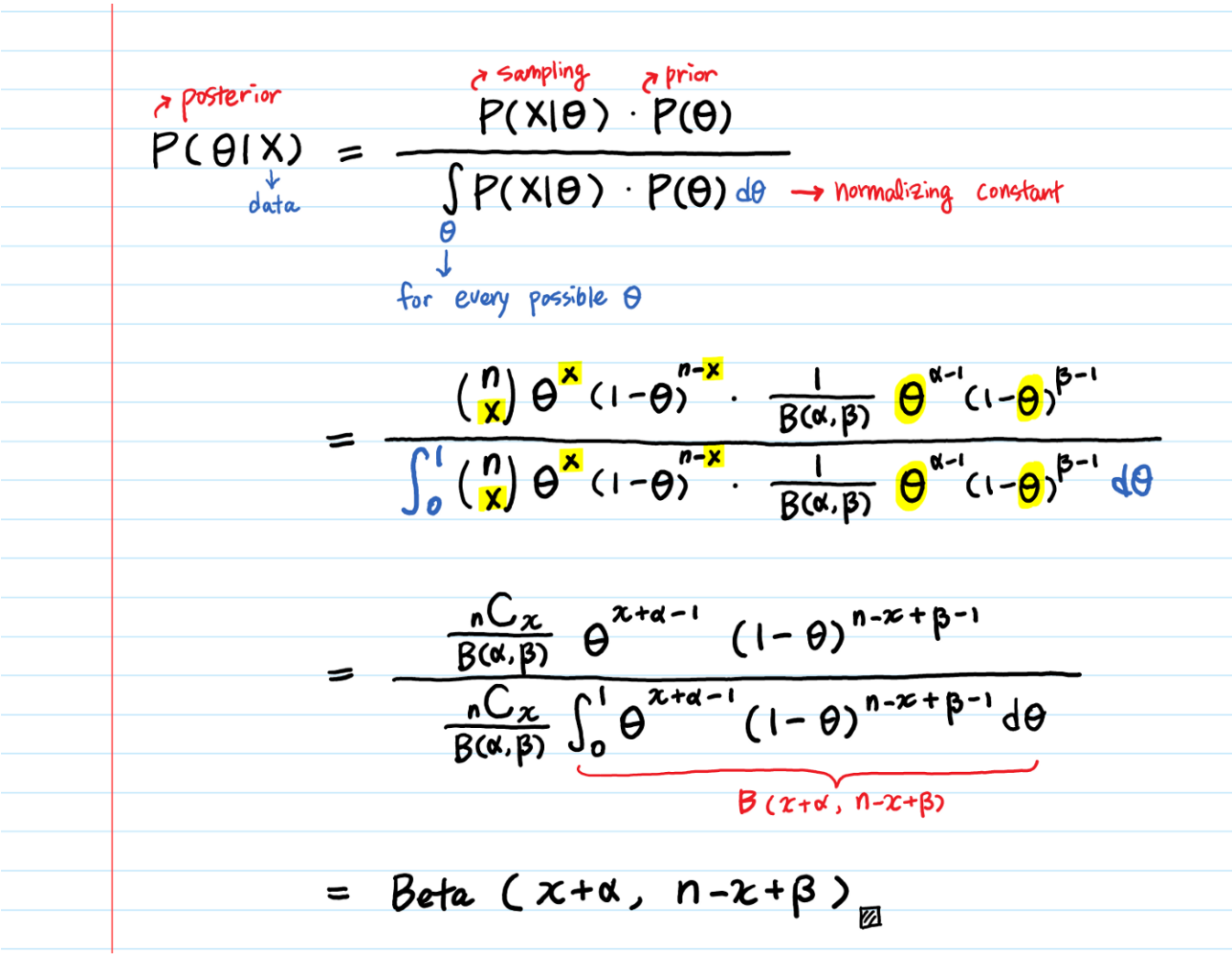


Let's plug them into the famous Bayes formula.

θ is the probability of success.

x is the number of successes.

n is the total number of trials, therefore **n-x** is the number of failures.



Why does the last integral become **B(x+α, n-x+β)**? → https://bit.ly/2t1i2KT

The prior distribution **P(θ)** was **Beta(α, β)** and after getting **x** successes and **n-x** failures from the experiments, the posterior also becomes a Beta distribution with parameters **(x+α, n-x+β).**

What's nice here is you will know this analytically without doing the computation.

5. Conjugate Prior Distributions

The Beta distribution is a conjugate prior for the **Bernoulli, binomial, negative binomial** and **geometric** distributions (seems like those are the distributions that involve success & failure).

<Beta posterior>

```
Beta prior * Bernoulli likelihood → Beta posterior
Beta prior * Binomial likelihood → Beta posterior
Beta prior * Negative Binomial likelihood → Beta posterior
Beta prior * Geometric likelihood → Beta posterior


<Gamma posterior>
Gamma prior * Poisson likelihood → Gamma posterior
Gamma prior * Exponential likelihood → Gamma posterior

<Normal posterior>
Normal prior * Normal likelihood (mean) → Normal posterior
```

Conjugate prior **P(θ)** in an equation:

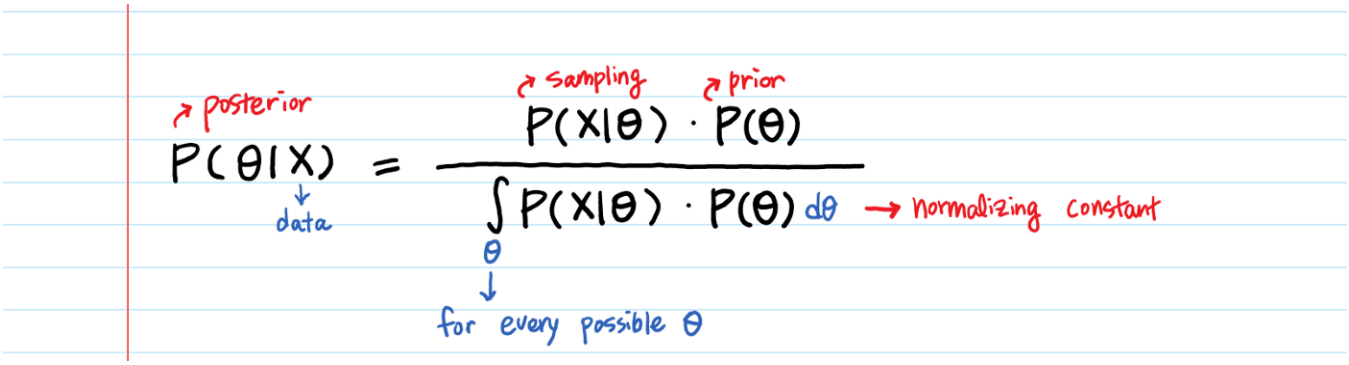
```
P(θ) such that P(θ|D) = P(θ)
```

An interesting way to put this is that even if you do all those experiments and multiply your likelihood to the prior, your initial choice of the prior distribution was **so good that the final distribution is the same as the prior**.

Conjugate prior = Convenient prior

A few things to note:

1. When we use the conjugate prior, sequential estimation (updating the counts after each observation) gives the same result as a batch estimation.
2. In order to **find the maximum posterior**, **you don't have to normalize** the multiplication of likelihood (sampling) and the prior (the integration for every possible **θ** in the denominator).



You can still find the maximum without normalizing. However, if you want to compare posteriors from different models, or calculate the point estimates, you need to normalize.