Probability Distribution Function

In this lesson, we will begin with an overview of probability distribution functions and then move on to discussing continuous distributions.

WE'LL COVER THE FOLLOWING

We've been mostly looking at small, discrete distributions in this course, but we started this series by looking at continuous distributions. Now that we have some understanding of how to solve probability problems on simple discrete distributions and Markov processes, let's go back to continuous distributions and see if we can apply some of these learnings to them.

What is a Probability Distribution Function?

Let's start with the basics. What exactly do we mean by a **probability distribution function**? So far in this course, we've mostly looked at the discrete analog, a non-normalized probability mass function. That is, for an `IDiscreteDistribution<T>` we have a weight function that gives us a value for each `T`. The probability of sampling a particular `T` is the weight divided by the total weight of all `T`s.

Of course, had we decided to go with double weights instead of integers, we could have made a normalized probability mass function: that is, the "weight" of each particular `T` is automatically divided by the total weight, and we get the probability out. In this scenario, the total weight adds up to 1.0.

We know from our exploration of the weighted integer distribution that we

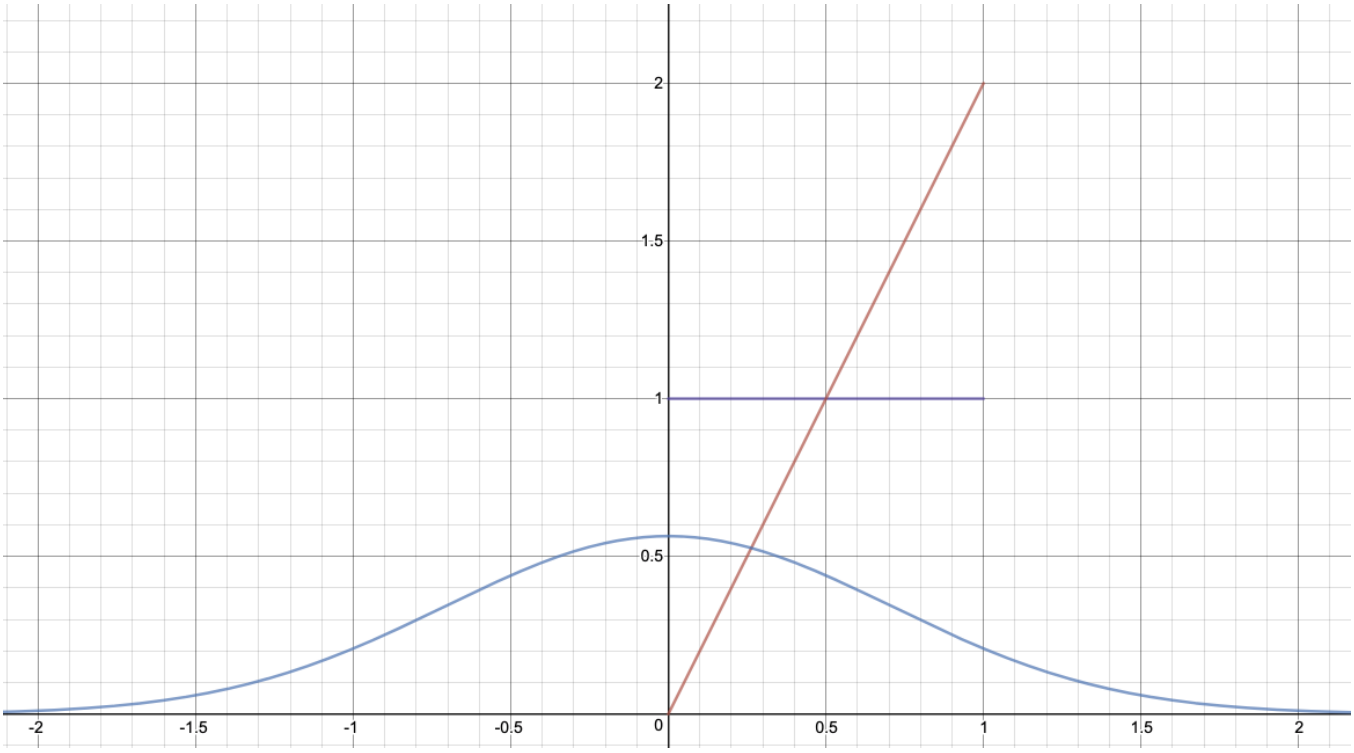
can think of our probability distributions as making a rectangle where various sub-rectangles are associated with particular values; we then *"throw a dart"* at the rectangle to sample from the distribution; where it lands gives us the sample.

> We will be abbreviating "*Probability Distribution Function*" as **PDF** for the rest of the course.

Continuous Distribution

Continuous distributions can be thought of in much the same way. Suppose we have a function from double to double, always non-negative, such that the total area under the curve is 1.0. Here are some examples:

```
double PDF1(double x) => x < 0.0 | x >= 1.0 ? 0.0 : 1.0;
double PDF2(double x) => x < 0.0 | x >= 1.0 ? 0.0 : 2 * x;
double PDF3(double x) => Exp(-(x * x)) / Sqrt(PI);
```



What is the meaning of these as probability distributions? Plainly the *"higher"* the function, the *"more likely"* any particular value is, but what does it even mean to say that in our `PDF2` and `PDF3` distributions, that 0.5 is *"less likely"* than 0.6 but they are *"equally likely"* in our `PDF1` distribution?

One way to think of it is that again, we *"throw a dart"* at the area under the

curve. Given any subset of that area, the probability of the dart landing inside it is proportional to the area of the subset, and the value sampled is the x coordinate of the dart.

We can make this a little bit more formal by restricting our areas to little rectangles:

1. Take a value, say 0.5.
2. Now take a tiny offset, call it ϵ. Doesn't matter what it is, so long as it is "pretty small".
3. The probability of getting a sample value between 0.5 and $0.5 + \epsilon$ is $PDF(0.5) \times \epsilon$. That is the area of a thin rectangle.

That's it! Let's say we've got values 0.5 and 0.6, and an ϵ of 0.01. How do our three distributions compare with each other near these values?

- With `PDF1`, the probability of getting a sample between 0.5 and 0.51 is approximately 0.010, and the probability of getting a sample between 0.6 and 0.61 is also approximately 0.010.
- With `PDF2`, the probability of getting a sample between 0.5 and 0.51 is approximately 0.010, and the probability of getting a sample between 0.6 and 0.61 is approximately 0.012.
- With `PDF3`, the probability of getting a sample between 0.5 and 0.51 is approximately 0.006, and the probability of getting a sample between 0.6 and 0.61 is approximately 0.004.

And moreover, *the approximation becomes better as ϵ gets smaller*.

> What if we want to know the probability of getting a value between two doubles that are not "really close together"? In that case, we can do "quadrature": divide it up into a bunch of regions that are all "really small" and sum up the areas of all those little rectangles to get an approximation of the area. Or, we can use integral calculus to find the exact answer. But for this course, we're going to try to find ways to sample from a distribution that has a particular PDF without doing any quadrature approximations or any integral calculus.

All right, we understand what PDFs are. To summarize:

- A PDF is a pure function from double to double.
- A PDF is defined over the entire range of doubles. $PDF(x) >= 0.0 \; for \; all \; x$.
- Integrating the PDF over the entire real line gives us a total area of 1.0.
- The probability that a sample is between x and $x + \epsilon$ is approximately $PDF(x) \times \epsilon$, if ϵ is small.

For our purposes, we'll assume that a "non-normalized" PDF has these properties:

- A PDF is a pure function from `T` to double...
- ... defined over the entire range of `T`.
- $PDF(t) >= 0.0 \; for \; all \; t$.
- Integrating the PDF over all possible values of `T` gives us a total area of A, for some finite positive value A. That is, the distribution is not necessarily "normalized" to have an area of 1.0. (You'd think that we can "easily" get a normalized distribution from a non-normalized distribution by dividing every weight by A but it is not always easy to know what that area is.)
- The probability that a sample is within ϵ of t is approximately $PDF(t) \times \epsilon \div A$ if ϵ is small.

> Aside: That last point is a little vague; what does it mean to be "within ϵ of some t" if t is not double? We're going to just not worry about that. It'll be fine. Particularly since we're mostly going to look at situations where `T` is double anyways.

This becomes useful for cases like the multidimensional case where `T` is (double, double), say, and by "within ϵ" we mean *"inside a square of side $\sqrt{\epsilon}$"* or some such thing. Putting this all on a solid mathematical footing would take us too far off-topic; you get the idea and let's move on.

Making a Definition in the Type System

As we'll see, continuous distributions are a much harder beast to tame than the small discrete distributions we've looked at before. However, there are still things we can do with them. Let's start by making a definition in the type system for a distribution that has one of our relaxed PDFs:

```
public interface IWeightedDistribution<T> : IDistribution<T>
{
  double Weight(T t);
}
```

This means that every *discrete distribution* gets to be a *weighted distribution*. Let's implement that:

```
public interface IDiscreteDistribution<T> : IWeightedDistribution<T>{
  IEnumerable<T> Support();
  new int Weight(T t);
}
```

And now we have some minor taxes to pay; we'll need to add

```
double IWeightedDistribution<R>.Weight(T t) => this.Weight(t);
```

to all our existing discrete distributions, but that is easily done.

We can also go back and fill in weight functions in our `Normal` class:

```
private static readonly double piroot = 1.0 / Sqrt(2 * PI);
public double Weight(double x) =>
  Exp(-(x - μ) * (x - μ) / (2 * σ * σ)) * piroot / σ;
```

And our `StandardContinuousUniform` class:

```
public double Weight(double x) =>
  0.0 <= x & x < 1.0 ? 1.0 : 0.0;
```

We've got a new variation on our old concept of weighting a distribution; let's see where we can go with this.

Implementation

Let's have a look at the code for this lesson now:

Program.cs

Bernoulli.cs

BetterRandom.cs

Distribution.cs

DistributionBuilder.cs

Empty.cs

Episode25.cs

Extensions.cs

IDiscreteDistribution.cs

IDistribution.cs

IWeightedDistribution.cs

Markov.cs

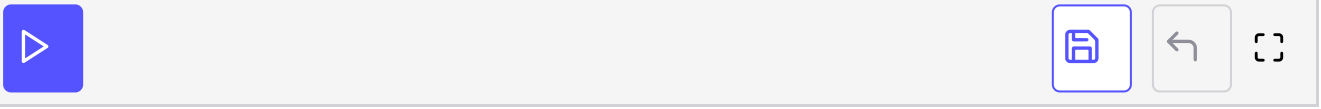
MarkovBuilder.cs

Normal.cs

Projected.cs

```
using System;
using System.IO;
using System.Linq;

namespace Probability
{
    static class Episode25
    {
        public static void DoIt()
        {
            Console.WriteLine("Episode 25 -- Random Shakespeare Company");
            var builder = new MarkovBuilder<string>();

            // corpus is provided
            var sentences = File.ReadLines("Shakespeare.txt")
              .Words()
              .Sentences();

            foreach (var sentence in sentences)
            {
                builder.AddInitial(sentence[0]);
                for (int i = 0; i < sentence.Count - 1; i += 1)
                    builder.AddTransition(sentence[i], sentence[i + 1]);
            }
```

```
            }

            var markov = builder.ToDistribution();

            Console.WriteLine(markov
              .Samples()
              .Take(100)
              .Select(x => x.SpaceSeparated())
              .NewlineSeparated());
        }
    }
}
```



We know how to efficiently sample from any small, discrete distribution with integer weights, but distributions based on arbitrary PDFs are much harder beasts to tame. Given an arbitrary non-normalized PDF, can we sample from it?