Computations with Arbitrarily Distributed Random Variables

Bachelor's Thesis

Author
Marten Lienen

Supervised and reviewed by
Prof. Dr. Stefan Harmeling

Reviewed by
Prof. Dr. Michael Leuschel

Heinrich-Heine-Universität Düsseldorf
Computer Vision, Computer Graphics and Pattern Recognition

July 21, 2015

Acknowledgements

Thanks to Andreas Troll and Dr. Georg Jansing from the institute of applied mathematics at HHU for explaining to me, why Gauss-Hermite quadrature is unstable, when it is applied to the product of two Gaussian distributed random variables.

Thanks to Prof. Dr. Stefan Harmeling for the idea, help, feedback and the ongoing affirmation, that there must be a way to make this work.

Contents

Chapter 1

Introduction

In the age of machine learning statistics and random variables have caught the attention of computer scientists and software engineers. They try to estimate the variables' parameters and properties from observed data and compute the distributions of dependent variables from there. In this thesis we tackle the latter problem. Its complexity varies from instance to instance.

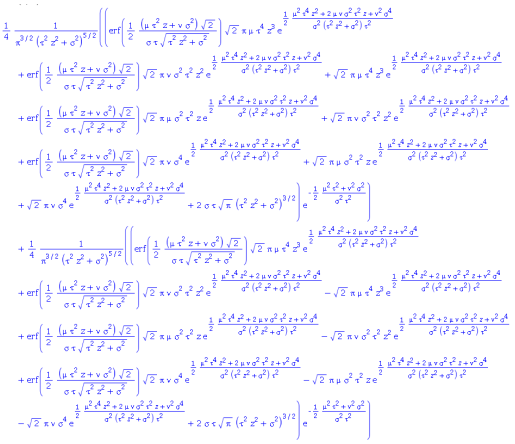
As a simple example you might be interested in the sum of two estimated random variables, which you assume to be Gaussian distributed. The sum's distribution can be worked out formally through basic analysis, which can even be done by hand.

$$X \sim \mathcal{N}(\mu, \sigma^2) \qquad Y \sim \mathcal{N}(\nu, \tau^2) \qquad \Rightarrow \qquad Z = X + Y \sim \mathcal{N}(\mu + \nu, \sigma^2 + \tau^2)$$

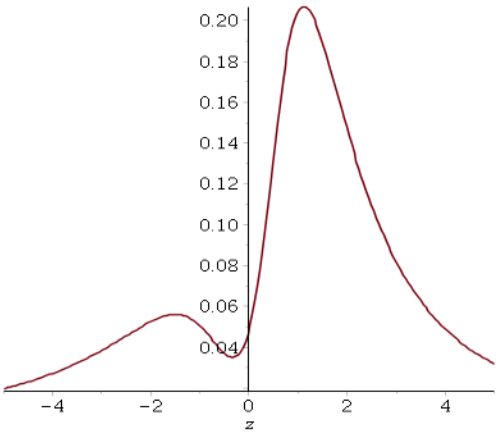
Most cases however are not manageable without the assistance of computing software like Maple [maple]. The following Maple program computes the probability density function (PDF) of the quotient of two random variables as defined in the previous paragraph. A quick look at figure 1.1 should tell you, that you probably would not stand a chance with pen and paper.

```
1 with(Statistics);
2 X := RandomVariable(Normal(mu, sigma));
3 Y := RandomVariable(Normal(nu, tau));
4 Z := X/Y;
5 PDF(Z, z);
```

The approach of ramping up computing power and letting the computer find a symbolic solution also has its limitations though. If you combine enough random variables with different operations, a technical computing solution will either be unable to find a solution or the found solution will contain so many left-over integrals and costly functions, that actually evaluating it is not feasible. At this point you have to resort to sampling. This means, that you sample n times from your random variables and compute n samples of the dependent variable. You can then use sample statistics to estimate basic properties of the dependent variable and a histogram of the samples is an approximation to its PDF. The probabilistic programming community [ppl] has developed various libraries and whole programming languages to explore sampling techniques. The next program is written in one of these called Church [church]. It samples from three random variables with different



(a) Symbolic PDF



(b) Plot for $\mu = 2, \nu = 0.5, \sigma = \tau = 1$

Figure 1.1: Maple finds a symbolic representation of the result's distribution

distributions and computes samples of the dependent variable from them. We obtain the approximation in figure 1.2, even when Maple was not able to handle this at all and just reported ∞.

```
1 (define (sample)
2   (define x1 (gaussian 1 2))
3   (define x2 (gamma 1 3))
4   (define y (exponential 2))
5
6   (/ (* x1 x2) y))
7
8 (define samples (repeat 100000 sample))
```

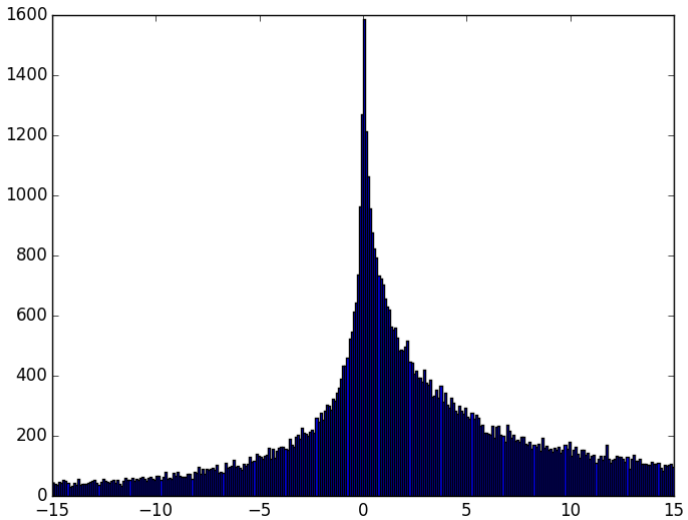


Figure 1.2: Approximation to a complicated PDF obtained by sampling

Symbolic representations give you a way to handle exact solutions in a compact form. The downside is, that they are restricted to simple problems and the resulting formulas can be very complicated and expensive to evaluate. Sampling can cope with every combination, that is computable, but is only approximate. Our approach lies between symbolic representation and sampling, a symbolic approximation. We find a symbolic representation that can approximate arbitrary probability distributions and at the same time is closed under certain operations. So you can do calculations with distributions of your choosing and still obtain a symbolic result.

We will

- formalize the problem and explain our idea in detail (Section 2)
- describe a way to represent any distribution as a mixture of Gaussians (Section 3.1)
- prove, that mixture distributions are closed under certain operations and are thus a good fit as a mathematical framework (Section 3.2)
- explore the viability of Gaussians as component distributions (Section 3.3)
- describe the structure of the julia implementation (Section 4)
- showcase our results and compare them to ones that were derived through sampling (Section 5)
- conclude this thesis with ideas for further work (Section 6)

Chapter 2

Our Approach

Consider $X_i \in \mathbb{R}, i \in \{1, \ldots, n\}$ n independent random variables with known PDFs $p(X_i = x)$, $f : \bigtimes_i \text{supp}(X_i) \to \mathbb{R}$ a function defined on at least the support of all the X_i and $Y = f(X_1, \ldots, X_n)$ the dependent variable. The task is then to derive the probability distribution of Y or at least an approximation to it from the known distributions of the X_i.

As explained in the first chapter, this problem is very hard in its full generality. Therefore we are going to simplify it by restricting the X_i as well as f. First, we restrict the distributions of the X_i to mixture distributions, especially mixtures of Gaussians. The first advantage of mixtures of Gaussians is, that with enough components you can approximate any distribution to arbitrary precision as you can see in figure 2.1. So we have simplified one aspect of the problem, while still being able to model any distribution. The other advantage is, that Gaussian distributions are closed under addition and subtraction and the products and quotients can be closely approximated by mixtures of Gaussians. This is important, because these closure properties of distributions are preserved, when you consider a mixture of them, as we see in section 3.2.

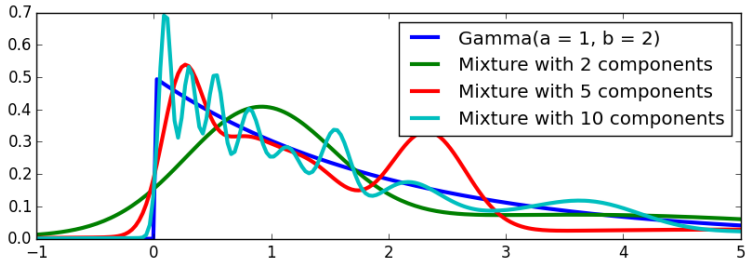


Figure 2.1: A gamma distribution approximated by mixtures of Gaussians

Secondly, we only consider f, that are differentiable and invertible in the first argument. The reasoning is that for such f there is a lemma called *multivariate change of variables* [Mur12, chapter 2.6.2.1], that lets you reduce the distribution of Y to the joint distribution of the X_i. This in turn lets us prove, that the family of mixture distributions is closed under any such f (section 3.2).

With these restrictions in place we can give symbolic approximations for any Y as long as Gaussian distributions are closed under f or the result of f can at least be reasonably well approximated by a mixture of Gaussians.

Chapter 3

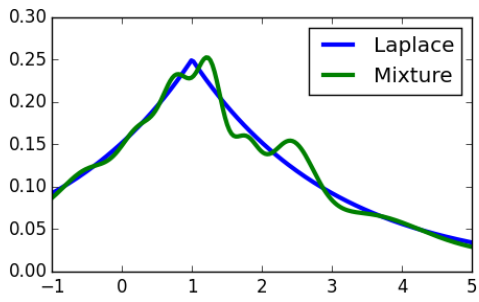
The Details

In this chapter we will work out the details. We begin with explaining how to convert arbitrary distributions into mixtures of Gaussians. Then we will prove, that mixture distributions are closed under certain functions. In the end we will look at Gaussians as component distributions of mixture distributions and give exact results and approximations for basic function applications.

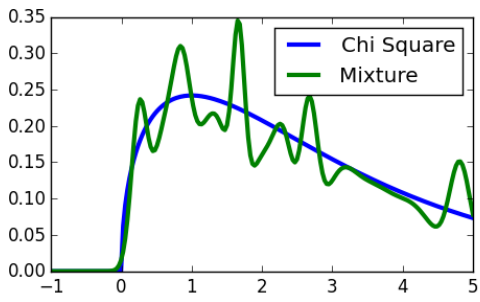
We have written a (very) short and informal introduction to random variables and probability distributions for the uninitiated in appendix A.

3.1 Modelling Distributions as Mixtures of Gaussians

The first step is based on the expectation-maximization (EM) algorithm [Mur12, chapter 11.4.2], an iterative algorithm to fit a mixture of Gaussians to best explain a set of samples. To find a mixture of Gaussians, that models a given distribution p, you first draw a set of samples from p. These samples are obviously distributed according to p, so when you fit a mixture of Gaussians with EM to best explain them, i.e. maximize the likelihood, it will closely model p.



(a) A Laplace distribution with $a = 1, b = 2$



(b) A chi-squared distribution with $k = 3$

Figure 3.1: Some distributions modelled as mixtures of Gaussians

As shown in figure 3.1 EM also works for distributions, that do not share many properties with Gaussians, but yields the best results for distributions that have whole $\mathbb{R}$ as support.

3.2 Closure of Mixtures

Let X and $Y_1, \ldots, Y_n$ be random variables with mixture distribution, i.e.

$$p(X = x) = \sum_{i=1}^{m} \alpha_i \cdot p(X_i = x) \qquad p(Y_i = y) = \sum_{j=1}^{m_i} \beta_{i,j} \cdot p(Y_{i,j} = y)$$

where the X_i and $Y_{i,j}$ are random variables distributed like the component distributions of X respectively Y_i.

Consider a function $f : S \times \bigtimes_{i=1}^{n} S_i \to \mathbb{R}$, where $S \supseteq \operatorname{supp}(X)$ and $S_i \supseteq \operatorname{supp}(Y_i), i \in \{1, \ldots, n\}$. Then we might ask ourselves, how $Z = f(X, Y_1, \ldots, Y_n)$ is distributed. So we are looking to evaluate

$$p(Z = z) = \int_{S_n} \ldots \int_{S_1} p(Z = 1, Y_1 = y_1, \ldots, Y_n = y_n) \, \mathrm{d}y_1 \ldots \mathrm{d}y_n$$

First, we have to tie the joint PDF of Z and $Y_{1,\ldots,n}$ to the joint PDF of X and $Y_{1,\ldots,n}$. Such an expression can be derived using *multivariate change of variables* [Mur12, chapter 2.6.2.1], which I will restate here.

Lemma 1 (Multivariate change of variables)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be differentiable and invertible, $X \in \mathbb{R}^n$ a random vector and $Y = f(X)$. Then the distribution of Y is given by*

$$p(Y = y) = |\det J| \cdot p(X = f^{-1}(y))$$

where J is the Jacobian of f^{-1}, i.e.

$$J = \frac{\partial X}{\partial Y} = \begin{pmatrix} \frac{\partial X_1}{\partial Y_1} & \cdots & \frac{\partial X_1}{\partial Y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial X_n}{\partial Y_1} & \cdots & \frac{\partial X_n}{\partial Y_n} \end{pmatrix}$$

For the rest of this section we will use $p(Y_{1,\ldots,n} = y_{1,\ldots,n})$ as a shorthand for $p(Y_1 = y_1, \ldots, Y_n = y_n)$. Also $f^{-1}_{y_1,\ldots,y_n}$ should be read as the inverse of f in the first argument with $y_1, \ldots, y_n$ held constant, i.e.

$$f^{-1}_{y_1,\ldots,y_n}(f(z, y_1, \ldots, y_n)) = z$$

Lemma 2. *If f is differentiable and invertible in the first argument, we can express the joint PDF of Z and $Y_{1,\ldots,n}$ in terms of the joint PDF of X and $Y_{1,\ldots,n}$ as*

$$p(Z = z, Y_{1,\ldots,n} = y_{1,\ldots,n}) = \left| \frac{\mathrm{d} f^{-1}_{y_1,\ldots,y_n}(z)}{\mathrm{d}z} \right| \cdot p\left(X = f^{-1}_{y_1,\ldots,y_n}(z), Y_{1,\ldots,n} = y_{1,\ldots,n}\right)$$

Proof. Define $g : S \times \bigtimes_{i=1}^{n} S_i \to \mathbb{R}^{n+1}$ as

$$g \begin{pmatrix} x \\ y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} f(x, y_1, \ldots, y_n) \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Then g is invertible and differentiable and the inverse is given by

$$g^{-1} \begin{pmatrix} z \\ y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} f^{-1}_{y_1,\ldots,y_n}(z) \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Next we examine the Jacobian J of g^{-1}. Let $(z, y_1, \ldots, y_n)^T \in \mathbb{R}^{n+1}$.

$$J = \begin{pmatrix} \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial z} & \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial y_1} & \cdots & \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial y_n} \\ \frac{\mathrm{d}y_1}{\mathrm{d}z} & \frac{\mathrm{d}y_1}{\mathrm{d}y_1} & \cdots & \frac{\mathrm{d}y_1}{\mathrm{d}y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\mathrm{d}y_n}{\mathrm{d}z} & \frac{\mathrm{d}y_n}{\mathrm{d}y_1} & \cdots & \frac{\mathrm{d}y_n}{\mathrm{d}y_n} \end{pmatrix} = \begin{pmatrix} \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial z} & \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial y_1} & \cdots & \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial y_n} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

J is an upper-triangular matrix, which means

$$\det J = \frac{\mathrm{d}f^{-1}_{y_1,\ldots,y_n}}{\partial z}$$

Therefore the claimed equality follows from lemma 1. □

Lemma 3. *Let $n, m_1, \ldots, m_n \in \mathbb{N}$, $a_{i,j} \in \mathbb{R}$, $f : \mathbb{R} \to \mathbb{R}$ and X an interval in $\mathbb{R}$.*

$$\int_X \prod_{i=1}^{n} \left(\sum_{j=1}^{m_i} \alpha_{i,j} f_{i,j}(x) \right) \mathrm{d}x = \sum_{j_1=1}^{m_1} \cdots \sum_{j_n=1}^{m_n} \left(\prod_{i=1}^{n} \alpha_{i,j_i} \right) \int_X \left(\prod_{i=1}^{n} f_{i,j_i}(x) \right) \mathrm{d}x$$

Proof.

$$\int_X \prod_{i=1}^{n} \left(\sum_{j=1}^{m_i} \alpha_{i,j} f_{i,j}(x) \right) \mathrm{d}x = \int_X \sum_{j_1=1}^{m_1} \cdots \sum_{j_n=1}^{m_n} \left(\prod_{i=1}^{n} \alpha_{i,j_i} f_{i,j_i}(x) \right) \mathrm{d}x$$

Interchange integral and sums

$$= \sum_{j_1=1}^{m_1} \cdots \sum_{j_n=1}^{m_n} \int_X \left(\prod_{i=1}^{n} \alpha_{i,j_i} f_{i,j_i}(x) \right) \mathrm{d}x$$

Regroup factors

$$= \sum_{j_1=1}^{m_1} \cdots \sum_{j_n=1}^{m_n} \int_X \left(\prod_{i=1}^{n} \alpha_{i,j_i} \right) \left(\prod_{i=1}^{n} f_{i,j_i}(x) \right) \mathrm{d}x$$

Pull out constant factors

$$= \sum_{j_1=1}^{m_1} \cdots \sum_{j_n=1}^{m_n} \left(\prod_{i=1}^{n} \alpha_{i,j_i} \right) \int_X \left(\prod_{i=1}^{n} f_{i,j_i}(x) \right) \mathrm{d}x$$

□

Equipped with lemma 2 and 3 we can now derive the general form of Z's distribution.

Lemma 4. *If f is differentiable and invertible in the first argument, the distribution of Z is a mixture distribution given by the following equation*

$$p(Z = z) = \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) p(f(X_i, Y_{1,k_1}, \ldots, Y_{n,k_n}) = z)$$

Proof.

Expand marginal distribution as an integral over the joint distribution

$$p(Z = z) = \int_{S_n} \dots \int_{S_1} p(Z = z, Y_{1,\dots,n} = y_{1,\dots,n}) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Rewrite it using lemma 2

$$= \int_{S_n} \dots \int_{S_1} \left| \frac{\mathrm{d}x}{\mathrm{d}z} \right| \cdot p\left(X = f^{-1}_{y_1,\dots,y_n}(z), Y_{1,\dots,n} = y_{1,\dots,n}\right) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Use the fact, that X and Y_i are independent, i.e. that the joint PDF can be factorized

$$= \int_{S_n} \dots \int_{S_1} \left| \frac{\mathrm{d}x}{\mathrm{d}z} \right| \cdot p\left(X = f^{-1}_{y_1,\dots,y_n}(z)\right) \prod_{j=1}^{n} p\left(Y_j = y_j\right) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Plug in the distributions of X and Y_j

$$= \int_{S_n} \dots \int_{S_1} \left| \frac{\mathrm{d}x}{\mathrm{d}z} \right| \cdot \left(\sum_{i=1}^{m} \alpha_i \cdot p(X_i = f^{-1}_{y_1,\dots,y_n}(z)) \right) \prod_{j=1}^{n} \left(\sum_{k=1}^{m_j} \beta_{j,k} \cdot p(Y_{j,k} = y_j) \right) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Pull out the coefficients using lemma 3

$$= \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) \int_{S_n} \dots \int_{S_1} \left| \frac{\mathrm{d}x}{\mathrm{d}z} \right| \cdot p(X_i = f^{-1}_{y_1,\dots,y_n}(z)) \cdot \prod_{j=1}^{n} p(Y_{j,k_j} = y_j) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Join the probability densities together

$$= \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) \int_{S_n} \dots \int_{S_1} \left| \frac{\mathrm{d}x}{\mathrm{d}z} \right| \cdot p\left(\begin{array}{c} X_i = f^{-1}_{y_1,\dots,y_n}(z), \\ Y_{1,k_1} = y_1, \dots, Y_{n,k_n} = y_n \end{array} \right) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Reverse application of lemma 2

$$= \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) \int_{S_n} \dots \int_{S_1} p\left(\begin{array}{c} f(X_i, Y_{1,k_1}, \dots, Y_{n,k_n}) = z, \\ Y_{1,k_1} = y_1, \dots, Y_{n,k_n} = y_n \end{array} \right) \, \mathrm{d}y_1 \dots \mathrm{d}y_n$$

Marginalize the Y_i

$$= \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) p(f(X_i, Y_{1,k_1}, \dots, Y_{n,k_n}) = z)$$

□

Lemma 5. *If $f(X_i, Y_{1,k_1}, \dots, Y_{n,k_n})$ has a mixture distribution, Z has a mixture distribution as well.*

Proof. We will not proof it in full rigor, because it is only one rearrangement, but would involve a lot of additional indices for the distribution of $f(X_i, Y_{1,k_1}, \ldots, Y_{n,k_n})$.

$$p(Z = z) = \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) p(f(X_i, Y_{1,k_1}, \ldots, Y_{n,k_n}) = z)$$

Plug in the mixture distribution of f

$$= \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) \sum_{p} \gamma_p p_f(z)$$

Push the coefficients of Z into the inner sum

$$= \sum_{i=1}^{m} \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \sum_{p} \alpha_i \left(\prod_{j=1}^{n} \beta_{j,k_j} \right) \gamma_p p_f(z)$$

□

Lemma 4 and 5 together are the foundation of this thesis. They say, that if X and the Y_i have a mixture of P (e.g. Gaussians) distribution, Z will have a mixture of P distribution as well, if f applied to the component distributions of X and Y_i is distributed as P or a mixture of P. This means, that to prove results in our restricted setting, we only need to prove it for P-distributed variables and it will generalize to mixtures of P automatically.

3.3 Gaussians as Component Distributions

In this section we focus on Gaussian distributions and how to express the distributions of basic $f(X_1, \ldots, X_n)$ as mixtures of gaussians.

3.3.1 Affine Transformations

Let X be a random variable with a Gaussian distribution, i.e. $X \sim \mathcal{N}(\mu, \sigma^2)$. A basic thing to do with X is to apply an affine transformation, because every operation involved is just a transformation of the parameters of the original distribution, as we will presently see. Along the way we will also reduce subtraction and division to addition respectively multiplication, because they enhance the usability of the julia library.

First we need to establish two rearrangement rules for Gaussian distributions.
Lemma 6. *Let $a \in \mathbb{R}$. Then $\mathcal{N}(x + a \mid \mu, \sigma^2) = \mathcal{N}(x \mid \mu - a, \sigma^2)$.*
Proof.

$$\begin{aligned}\mathcal{N}(x + a \mid \mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{((x + a) - \mu)^2}{2\sigma^2} \right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - (\mu - a))^2}{2\sigma^2} \right) = \mathcal{N}(x \mid \mu - a, \sigma^2)\end{aligned}$$

□

Lemma 7. *Let $a \in \mathbb{R} \setminus \{0\}$. Then $\mathcal{N}(ax \mid \mu, \sigma^2) = \frac{1}{|a|} \cdot \mathcal{N}\left(x \mid \frac{\mu}{a}, \frac{\sigma^2}{a^2}\right)$.*

Proof.

$$\begin{aligned}
\mathcal{N}(ax \mid \mu, \sigma^2) =& \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(ax-\mu)^2}{2\sigma^2}\right) \\
=& \frac{1}{\sqrt{2\pi\left(\frac{\sigma a}{a}\right)^2}} \exp\left(-\frac{a^2\left(x-\frac{\mu}{a}\right)^2}{2\sigma^2}\right) \\
=& \frac{1}{|a|} \frac{1}{\sqrt{2\pi\frac{\sigma^2}{a^2}}} \exp\left(-\frac{\left(x-\frac{\mu}{a}\right)^2}{2\frac{\sigma^2}{a^2}}\right) \\
=& \frac{1}{|a|} \cdot \mathcal{N}\left(x \mid \frac{\mu}{a}, \frac{\sigma^2}{a^2}\right)
\end{aligned}$$

□

Next we restate a special case of lemma 1 taken from [Mur12, chapter 2.6.2].

Lemma 8 (Change of variables)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable, monotonic and hence invertible. We can "unapply" f through the following formula*

$$p(f(X) = y) = \left|\frac{\mathrm{d}f^{-1}}{\mathrm{d}y}\right| \cdot p(X = f^{-1}(y))$$

With these in place we can now examine addition, subtraction, multiplication and division with a scalar.

Lemma 9. $-X \sim \mathcal{N}(-\mu, \sigma^2)$

Proof. This follows directly from lemma 7 and 8.

$$p(-X = x) = \mathcal{N}((-1)x \mid \mu, \sigma^2) = \frac{1}{|-1|} \cdot \mathcal{N}\left(x \mid \frac{\mu}{-1}, \frac{\sigma^2}{(-1)^2}\right) = \mathcal{N}(x \mid -\mu, \sigma^2)$$

□

Lemma 10. *Let $a \in \mathbb{R}$. Then $a + X = X + a \sim \mathcal{N}(\mu + a, \sigma^2)$.*

Proof. This follows directly from lemma 6 and 8.

$$p(X + a = x) = p(X = x - a) \cdot \left|\frac{\mathrm{d}(x-a)}{\mathrm{d}x}\right| = \mathcal{N}(x - a \mid \mu, \sigma^2) = \mathcal{N}(x \mid \mu + a, \sigma^2)$$

□

Corollary. *Let $a \in \mathbb{R}$. Then $X - a \sim \mathcal{N}(\mu - a, \sigma^2)$.*

Proof. $X - a = X + (-a)$ □

Corollary. *Let $a \in \mathbb{R}$. Then $a - X \sim \mathcal{N}(a - \mu, \sigma^2)$.*

Proof. $a - X = a + (-X)$ □

Lemma 11. *Let $a \in \mathbb{R} \setminus \{0\}$. Then $aX = Xa \sim \mathcal{N}(a\mu, a^2\sigma^2)$.*

Proof. This follows from lemma 7 and 8.

$$\begin{aligned} p(aX = x) &= \left|\frac{\mathrm{d}\frac{x}{a}}{\mathrm{d}x}\right| \cdot p\left(X = \frac{x}{a}\right) \\ &= \left|\frac{1}{a}\right| \cdot p\left(X = \frac{x}{a}\right) \\ &= \left|\frac{1}{a}\right| \cdot \mathcal{N}\left(\frac{x}{a} \mid \mu, \sigma^2\right) \\ &= \frac{1}{|a|} \cdot |a| \cdot \mathcal{N}\left(x \mid a\mu, a^2\sigma^2\right) \\ &= \mathcal{N}\left(x \mid a\mu, a^2\sigma^2\right) \end{aligned}$$

□

Corollary. *Let $a \in \mathbb{R} \setminus \{0\}$. Then $\frac{X}{a} \sim \mathcal{N}\left(\frac{\mu}{a}, \frac{\sigma^2}{a^2}\right)$.*

Proof. $\frac{X}{a} = X \cdot \frac{1}{a}$ □

Note, that we did not cover $\frac{a}{X}$. Division is not monotonic for a variable denominator, so lemma 8 does not hold and the distribution of $\frac{a}{X}$ cannot be evaluated with the techniques used in this section.

3.3.2 Affine Transformations of n Variables

A natural enhancement of section 3.3.1 is to generalize it to multiple variables. Therefore we will first determine the distribution of the sum of two Gaussian distributed variables.

Lemma 12. *Let $X \sim \mathcal{N}(\mu, \sigma^2), Y \sim \mathcal{N}(\nu, \tau^2)$ be two independent, Gaussian distributed variables. The distribution of their sum is*

$$X + Y \sim \mathcal{N}(\mu + \nu, \sigma^2 + \tau^2)$$

Proof. For a proof see [Kre05, Satz 11.9]. □

With lemma 12 combined with the previous section, we can now calculate the distribution of a linear combination of two Gaussian distributed variables. Applying this inductively, we can evaluate affine transformations of n variables.

3.3.3 Products of Gaussian Random Variables

Multiplication is indeed monotonic and invertible in the first argument, but differs from the previous operations in that Gaussian distributions are not closed under it. Actually the products of two gaussians do not have any standard distribution at all. So to stay inside the realm of mixtures of Gaussians, we have to approximate them with a mixture of Gaussians using numerical integration algorithms.

Let X and Y be two normally distributed variables

$$X \sim \mathcal{N}(\mu, \sigma^2) \qquad Y \sim \mathcal{N}(\nu, \tau^2)$$

and $Z = X \cdot Y$ their product. Then Z is distributed as follows

$$\begin{aligned} p(Z = z) &= \int_{-\infty}^{\infty} \frac{1}{|y|} \cdot p\left(X = \frac{z}{y}\right) \cdot p(Y = y)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \frac{1}{|y|} \cdot \mathcal{N}\left(\frac{z}{y} \mid \mu, \sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \frac{1}{|y|} \cdot |y| \cdot \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \end{aligned}$$

We would like to write this as a mixture of Gaussians, i.e. $\sum_i \alpha_i \cdot \mathcal{N}(\mu_i, \sigma_i^2)$. Gaussian quadrature comes to mind, a family of numerical integration algorithms, that approximate

$$\int_a^b f(x) \cdot w(x)\ \mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i)$$

with n summands for various combinations of a, b and w as long as the integrand can be written as $f(x) \cdot w(x)$, where w is a weight function determined by the specific algorithm and f an arbitrary function from $C^{2n-1}[a, b]$. $C^n[a, b]$ is thereby defined as the set of all functions, that are at least n times continuously differentiable on $[a, b]$. It probably does not need to be mentioned, but the approximation becomes more accurate with increasing n.

Gauss-Hermite Quadrature

Gauss-Hermite quadrature is a special case of Gauss quadrature for integrals of the form

$$\int_{-\infty}^{\infty} f(x) \cdot \exp(-x^2)\ \mathrm{d}x$$

This looks like a perfect fit for our integral, because the PDF of a Gaussian distribution is practically of the form $a \cdot \exp(-x^2)$, so one of the Gaussians can serve as the weight function, while the other persists as part of f creates the Gaussians in the resulting mixture of Gaussians.

$$\begin{aligned} p(Z = z) &= \int_{-\infty}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{(y-\nu)^2}{2\tau^2}\right)\ \mathrm{d}y \end{aligned}$$

To transform the integrand into the form $f(x) \cdot \exp(-x^2)$ we will now substitute

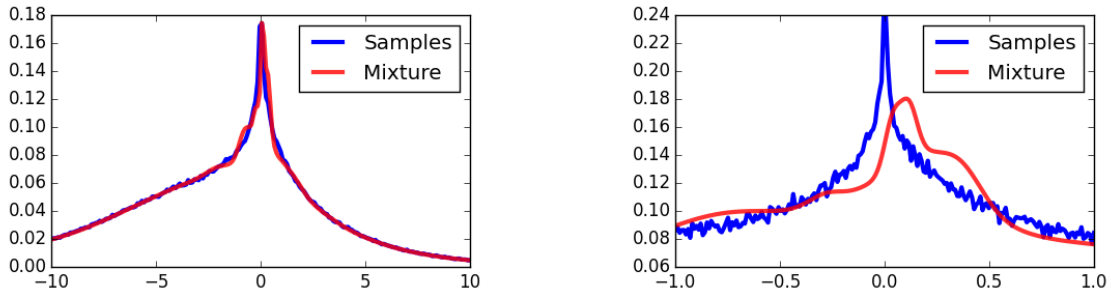
$$x = \frac{y - \nu}{\sqrt{2}\tau} \Leftrightarrow y = \sqrt{2}\tau x + \nu \qquad \frac{\mathrm{d}x}{\mathrm{d}y} = \frac{1}{\sqrt{2}\tau}$$

$$= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathcal{N}\left(z \mid \left(\sqrt{2}\tau x + \nu\right)\mu, \left(\sqrt{2}\tau x + \nu\right)^2 \sigma^2\right) \cdot \exp\left(-x^2\right)\ \mathrm{d}x$$

With the integrand in the right form we can apply the approximation.

$$p(Z = z) \approx \sum_{i=1}^{n} \frac{w_i}{\sqrt{\pi}} \cdot \mathcal{N}\left(z \mid \left(\sqrt{2\tau^2}x_i + \nu\right)\mu, \left(\sqrt{2\tau^2}x_i + \nu\right)^2 \sigma^2\right)$$

where x_i and w_i are the positions and weights as defined by the Gauss-Hermite quadrature method. Interestingly it follows from the construction of the method, that $\sum w_i = \sqrt{\pi}$, so the approximation is a convex combination and thus a real mixture distribution.



(a) An overview of the general fit

(b) Zoomed in around 0

Figure 3.2: A product distribution approximated by a mixture of Gaussians

Figure 3.2a shows, that the calculated mixture of Gaussians closely follows the true PDF obtained by sampling except in a neighborhood of 0. The zoomed-in plot in figure 3.2b confirms indeed, that the approximation still has a similar shape, but is more wobbly than the true PDF in the proximity of 0.

The bad fit around 0 is actually inherent to our problem and choice of algorithm. Remember the requirement of any Gauss quadrature method, that f be in $C^{2n-1}[a, b]$, and then look at the original integral and its integrand as a function of y.

$$\int_{-\infty}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y$$

If $z = 0$ and y approaches 0, the integrand tends towards ∞ from both sides, so it is not even once continuously differentiable in 0, though it is continuously differentiable infinite times everywhere else. So any Gauss quadrature method will be numerically unstable. It is, however, sufficiently stable for our cause, if you evaluate it with z not too close 0, which is also evident in figure 3.2a.

The problem obviously persists after the substitution, but now we can pinpoint exactly, where the instabily will occur. Recall the transformed integral

$$\int_{-\infty}^{\infty} \mathcal{N}\left(z \mid \left(\sqrt{2}\tau x + \nu\right)\mu, \left(\sqrt{2}\tau x + \nu\right)^2 \sigma^2\right) \cdot \exp\left(-x^2\right)\ \mathrm{d}x$$

We have the same problem as before, when $\sqrt{2}\tau x + \nu$ is small, i.e. $x \approx -\frac{\nu}{\sqrt{2}\tau}$, and $z \approx \left(\sqrt{2}\tau x + \nu\right)\mu$. In figure 3.3 we plot the 10 components with minimal variance. Their x-value is their mean and their y-value their variance. We observe, that all of them are in direct proximity of the instability.

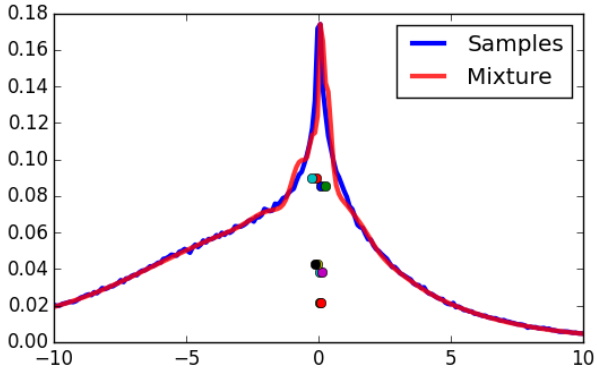


Figure 3.3: The components with minimal variance are in direct proximity of the instability

Gauss-Laguerre Quadrature

Another algorithm in the family of Gauss quadratures is Gauss-Laguerre quadrature. It is made for integrals of the form

$$\int_0^\infty f(x) \cdot \exp(-x)\ \mathrm{d}x$$

which does not match our situation quite as well as Gauss-Hermite does, so we will need to do some more preparatory work. The motivation for trying Gauss-Laguerre quadrature is, that this procedure may cope better with the instability around 0.

To make Gauss-Laguerre quadrature applicable, we have to rewrite the integral from $-\infty$ to ∞ as two integrals from 0 to ∞.

$$\begin{aligned}
p(Z = z) &= \int_{-\infty}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\
&= \int_{-\infty}^{0} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y + \int_{0}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\
&= -\int_{\infty}^{0} \mathcal{N}\left(z \mid -y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(-y \mid \nu, \tau^2)\ \mathrm{d}y + \int_{0}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\
&= \int_{0}^{\infty} \mathcal{N}\left(z \mid -y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(-y \mid \nu, \tau^2)\ \mathrm{d}y + \int_{0}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y
\end{aligned}$$

Both of these integrals can be solved approximately through Gauss-Laguerre integration. Yet we will first have to rewrite them in the required form and once again the integrands are not sufficiently differentiable in 0. This time we may be able to avert the problem though through a very non-mathematical approach. Instead of integrating over $[0, \infty]$ we integrate over $[\varepsilon, \infty]$ for an $\varepsilon > 0$. For some ε the payoff from not evaluating the integrand in the proximity of the instability may be greater than the reduced accuracy from ignoring an ε-strip of the integral.

We begin with rearranging the first integrand to the form $f(y) * e^{-y}$.

$$\begin{aligned}
&\int_{\varepsilon}^{\infty} \mathcal{N}\left(z \mid -y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(-y \mid \nu, \tau^2)\ \mathrm{d}y \\
&= \int_{\varepsilon}^{\infty} \mathcal{N}\left(z \mid -y\mu, y^2\sigma^2\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{(-y-\nu)^2}{2\tau^2}\right)\ \mathrm{d}y \\
&= \int_{\varepsilon}^{\infty} \mathcal{N}\left(z \mid -y\mu, y^2\sigma^2\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\left(\frac{y+\nu}{\sqrt{2\tau^2}}\right)^2\right)\ \mathrm{d}y
\end{aligned}$$

Substitute $t = \left(\frac{y+\nu}{\sqrt{2\tau^2}}\right)^2$, $y = \pm\sqrt{2\tau^2 t} - \nu$, $\frac{\mathrm{d}t}{\mathrm{d}y} = \frac{y+\nu}{\tau^2} = \frac{(\pm\sqrt{2\tau^2 t}-\nu)+\nu}{\tau^2} = \pm\frac{\sqrt{2t}}{\sqrt{\tau^2}}$

$$= \int_{\frac{(\varepsilon+\nu)^2}{2\tau^2}}^{\infty} \mathcal{N}\left(z \mid -\left(\sqrt{2\tau^2 t} - \nu\right)\mu, \left(\sqrt{2\tau^2 t} - \nu\right)^2\sigma^2\right) \cdot \frac{1}{2\sqrt{\pi t}} \exp\left(-t\right)\ \mathrm{d}t$$

Substitute $s = t - \frac{(\varepsilon+\nu)^2}{2\tau^2}$, $t = s + \frac{(\varepsilon+\nu)^2}{2\tau^2}$, $\frac{\mathrm{d}s}{\mathrm{d}t} = 1$

$$= \int_{0}^{\infty} \mathcal{N}\left(z \;\middle|\; \begin{array}{c} -\left(\sqrt{2\tau^2 s + (\varepsilon+\nu)^2} - \nu\right)\mu, \\ \left(\sqrt{2\tau^2 s + (\varepsilon+\nu)^2} - \nu\right)^2\sigma^2 \end{array}\right) \cdot \frac{\exp\left(-\frac{(\varepsilon+\nu)^2}{2\tau^2}\right)}{2\sqrt{\pi\left(s + \frac{(\varepsilon+\nu)^2}{2\tau^2}\right)}} \exp(-s)\ \mathrm{d}s$$

Next we rearrange the second integrand.

$$\begin{aligned}
&\int_{\varepsilon}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\
&= \int_{\varepsilon}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{(y-\nu)^2}{2\tau^2}\right)\ \mathrm{d}y \\
&= \int_{\varepsilon}^{\infty} \mathcal{N}\left(z \mid y\mu, y^2\sigma^2\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\left(\frac{y-\nu}{\sqrt{2\tau^2}}\right)^2\right)\ \mathrm{d}y
\end{aligned}$$

Substitute $t = \left(\frac{y-\nu}{\sqrt{2\tau^2}}\right)^2$, $y = \pm\sqrt{2\tau^2 t} + \nu$, $\frac{\mathrm{d}t}{\mathrm{d}y} = \frac{y-\nu}{\tau^2} = \frac{(\pm\sqrt{2\tau^2 t}+\nu)-\nu}{\tau^2} = \pm\frac{\sqrt{2t}}{\sqrt{\tau^2}}$

$$= \int_{\frac{(\varepsilon-\nu)^2}{2\tau^2}}^{\infty} \mathcal{N}\left(z \mid \left(\sqrt{2\tau^2 t} + \nu\right)\mu, \left(\sqrt{2\tau^2 t} + \nu\right)^2\sigma^2\right) \cdot \frac{1}{2\sqrt{\pi t}} \exp\left(-t\right)\ \mathrm{d}t$$

Substitute $s = t - \frac{(\varepsilon-\nu)^2}{2\tau^2}$, $t = s + \frac{(\varepsilon-\nu)^2}{2\tau^2}$, $\frac{\mathrm{d}s}{\mathrm{d}t} = 1$

$$= \int_{0}^{\infty} \mathcal{N}\left(z \;\middle|\; \begin{array}{c} \left(\sqrt{2\tau^2 s + (\varepsilon-\nu)^2} + \nu\right)\mu, \\ \left(\sqrt{2\tau^2 s + (\varepsilon-\nu)^2} + \nu\right)^2\sigma^2 \end{array}\right) \cdot \frac{\exp\left(-\frac{(\varepsilon-\nu)^2}{2\tau^2}\right)}{2\sqrt{\pi\left(s + \frac{(\varepsilon-\nu)^2}{2\tau^2}\right)}} \exp(-s)\ \mathrm{d}s$$

Now we can plug in the two rearrangements and get the approximation as a mixture of

Gaussians by applying the Gauss-Laguerre quadrature.

$$
\begin{aligned}
p(Z=z) \approx & \int_0^\infty \mathcal{N}\left(z \,\middle|\, \begin{array}{c} -\left(\sqrt{2\tau^2 s + (\varepsilon+\nu)^2} - \nu\right)\mu, \\ \left(\sqrt{2\tau^2 s + (\varepsilon+\nu)^2} - \nu\right)^2 \sigma^2 \end{array}\right) \cdot \frac{\exp\left(-\frac{(\varepsilon+\nu)^2}{2\tau^2}\right)}{2\sqrt{\pi\left(s + \frac{(\varepsilon+\nu)^2}{2\tau^2}\right)}} \exp(-s)\ \mathrm{d}s \\
& + \int_0^\infty \mathcal{N}\left(z \,\middle|\, \begin{array}{c} \left(\sqrt{2\tau^2 s + (\varepsilon-\nu)^2} + \nu\right)\mu, \\ \left(\sqrt{2\tau^2 s + (\varepsilon-\nu)^2} + \nu\right)^2 \sigma^2 \end{array}\right) \cdot \frac{\exp\left(-\frac{(\varepsilon-\nu)^2}{2\tau^2}\right)}{2\sqrt{\pi\left(s + \frac{(\varepsilon-\nu)^2}{2\tau^2}\right)}} \exp(-s)\ \mathrm{d}s \\
\approx & \sum_{i=1}^n w_i \frac{\exp\left(-\frac{(\varepsilon+\nu)^2}{2\tau^2}\right)}{2\sqrt{\pi\left(x_i + \frac{(\varepsilon+\nu)^2}{2\tau^2}\right)}} \cdot \mathcal{N}\left(z \,\middle|\, \begin{array}{c} -\left(\sqrt{2\tau^2 x_i + (\varepsilon+\nu)^2} - \nu\right)\mu, \\ \left(\sqrt{2\tau^2 x_i + (\varepsilon+\nu)^2} - \nu\right)^2 \sigma^2 \end{array}\right) \\
& + \sum_{i=1}^n w_i \frac{\exp\left(-\frac{(\varepsilon-\nu)^2}{2\tau^2}\right)}{2\sqrt{\pi\left(x_i + \frac{(\varepsilon-\nu)^2}{2\tau^2}\right)}} \cdot \mathcal{N}\left(z \,\middle|\, \begin{array}{c} \left(\sqrt{2\tau^2 x_i + (\varepsilon-\nu)^2} + \nu\right)\mu, \\ \left(\sqrt{2\tau^2 x_i + (\varepsilon-\nu)^2} + \nu\right)^2 \sigma^2 \end{array}\right)
\end{aligned}
$$

where the x_i are the roots of the n-th Laguerre polynomial and the w_i the associated weights.

This time things did not turn out as nice as with Gauss-Hermite quadrature. The $\sum_i w_i$ is 1 and the value of the coefficients is overall not constant, so the sums are not a real mixture distribution and the weights have to be normalized. This method also cannot keep up in a direct comparison (figure 3.4). Both methods find the correct mean and Gauss-Laguerre quadrature also produces a shape similar to the true probability density, but apart from that it is pretty far off. The variance in the proximity of the instability is too small, but overall it is too high. In this example the sample variance is roughly 40, closely matched by the Gauss-Hermite approximation, but the variance of the Gauss-Laguerre mixture is 80. There may be a bug in our program or an error in the math of this section regarding the variance, though we did not find any.

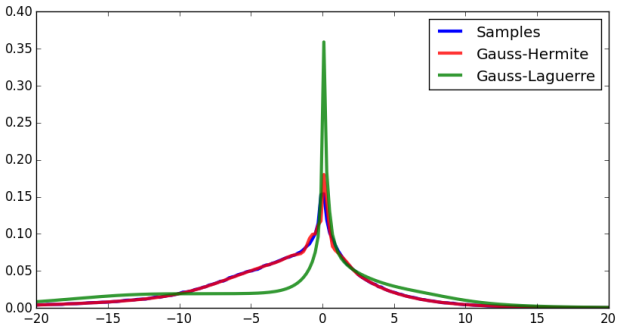


Figure 3.4: Gauss-Laguerre cannot keep up with Gauss-Hermite

3.3.4 Quotients of Gaussian Random Variables

Division shares all the important properties with multiplication: It, too, is differentiable and monotonic in the first argument and Gaussian distributions are not closed under

it. So we have to use numerical approximation again. This time we only try Gauss-Hermite quadrature, because the integrand has a very similar form and it served us well for multiplication.

Let X and Y be two normally distributed variables

$$X \sim \mathcal{N}(\mu, \sigma^2) \qquad Y \sim \mathcal{N}(\nu, \tau^2)$$

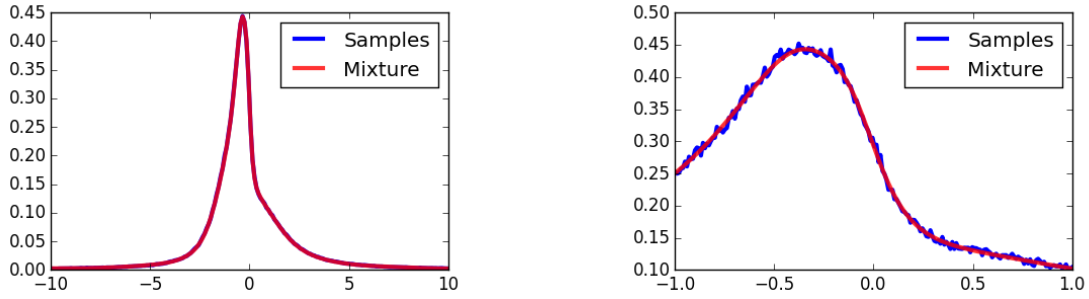
and $Z = \frac{X}{Y}$ their quotient. Then Z is distributed as follows

$$\begin{aligned} p(Z = z) &= \int_{-\infty}^{\infty} |y| \cdot p\left(X = zy\right) \cdot p(Y = y)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} |y| \cdot \mathcal{N}\left(zy \mid \mu, \sigma^2\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} |y| \cdot \frac{1}{|y|} \cdot \mathcal{N}\left(z \mid \frac{\mu}{y}, \frac{\sigma^2}{y^2}\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \mathcal{N}\left(z \mid \frac{\mu}{y}, \frac{\sigma^2}{y^2}\right) \cdot \mathcal{N}(y \mid \nu, \tau^2)\ \mathrm{d}y \end{aligned}$$

Now we can apply the same substitution as in the multiplication case to bring it into Gauss-Hermite quadrature form and get

$$p(Z = z) \approx \sum_{i=1}^{n} \frac{w_i}{\sqrt{\pi}} \cdot \mathcal{N}\left(z \,\middle|\, \frac{\mu}{\sqrt{2\tau^2}x_i + \nu}, \frac{\sigma^2}{\left(\sqrt{2\tau^2}x_i + \nu\right)^2}\right)$$

Figure 3.5 shows, that the fit is even better this time and there are no instabilities around 0. This is surprising, if you consider the problems we had with approximating a product distribution.



(a) Overview (b) Zoomed in around 0

Figure 3.5: Gauss-Hermite approximates a ratio distribution perfectly

The explanation is, that in this case the integrand is actually in $C^{\infty}[-\infty, \infty]$, because you

can eliminate y from all denominators.

$$\begin{aligned}
\mathcal{N}\left(z \mid \frac{\mu}{y}, \frac{\sigma^2}{y^2}\right) \cdot \mathcal{N}(y \mid \nu, \tau^2) =& \frac{1}{\sqrt{2\pi\frac{\sigma^2}{y^2}}} \exp\left(-\frac{\left(z-\frac{\mu}{y}\right)^2}{2\frac{\sigma^2}{y^2}}\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{(y-\nu)^2}{2\tau^2}\right) \\
=& \frac{y}{2\pi\sigma\tau} \cdot \exp\left(-\frac{\frac{1}{y^2}(zy-\mu)^2 y^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{(y-\nu)^2}{2\tau^2}\right) \\
=& \frac{y}{2\pi\sigma\tau} \cdot \exp\left(-\frac{(zy-\mu)^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{(y-\nu)^2}{2\tau^2}\right)
\end{aligned}$$

So the exponents are just polynomials. Since polynomials, compositions and products of functions in C^∞ are in turn in C^∞, the whole integrand is in C^∞.

Chapter 4

Implementation

The accompanying implementation is written in julia [jl], a new programming language for technical computing. It was originally developed by a group of four applied mathematicians at MIT and has since amassed a significant following in the applied mathematics and data science communities.

The foundation of the implementation is a library called Distributions.jl [distr], that defines types for all well-known distributions and some exotic ones. A minor role plays StatsBase.jl [stbs] in our implementation of EM, where it approximates the mean and variance of a distribution from samples. Finally we depend on GaussQuadrature.jl [gq] as well as FastGaussQuadrature.jl [fgq] for computing the weights and integration points for Gauss-Laguerre respectively Gauss-Hermite quadrature.

The central structure in the implementation is a type called `RandomVariable`, that attaches parameters for the different integration algorithms to a mixture distribution from Distributions.jl. The parameters themselves are wrapped in a type per algorithm. The rest of the library is pretty much just chapter 3 translated from math to julia.

Notable is julia's multiple dispatch system, which allows us to integrate the `RandomVariable` type seamlessly into julia's number system, so that you can pass a `RandomVariable` everywhere you could pass a number as long as you only use the supported operations.

The source code is attached in appendix B as well as published on github [gh].

Chapter 5

Demos

In this chapter we will present example plots for all operations, that we have examined, and combinations of them.

5.1 Modelling with EM

In section 3.1 we described, how to model arbitrary distributions as a mixture of Gaussians. It would be interesting to see, how well this method performs on various distributions and how many components you need, to get a reasonable fit.

We observe, that there are two classes in regards to how well they can be approximated by a mixture of Gaussians. The first group contains the symmetric distributions with support $\mathbb{R}$. Examples are student's T (figure 5.1), Cauchy (figure 5.2) and Laplace distributions (figure 5.3).

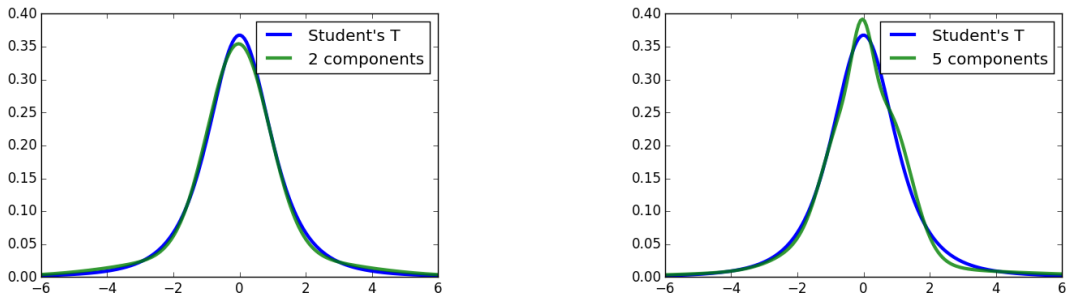


Figure 5.1: A student's T distribution is well approximated with a handful of components

All of these are very closely matched by a mixture of Gaussians with just 5 components, but for some distributions even 2 yield reasonable results.

The second class consists of the distributions, that are only defined on parts of $\mathbb{R}$. Some of them are defined on $\mathbb{R}_{\geq 0}$, for example the Gamma (figure 5.4) or Weibull distributions (figure 5.5).

Others are only defined on a finite interval, for instance a triangular (figure 5.6) or a uniform distribution (figure 5.7).

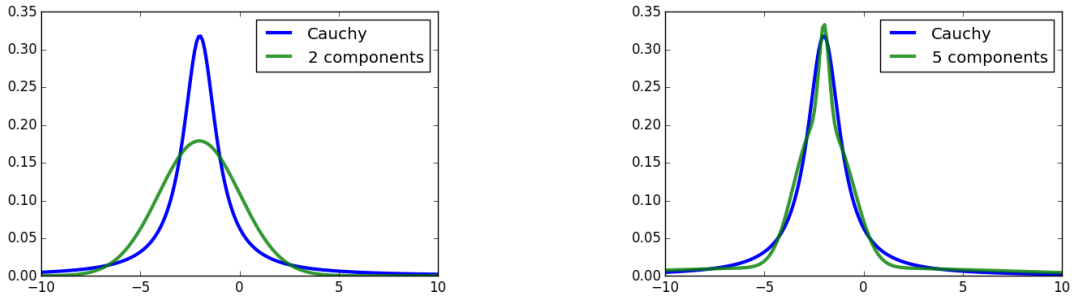


Figure 5.2: A Cauchy distribution needs a few more components for a good fit

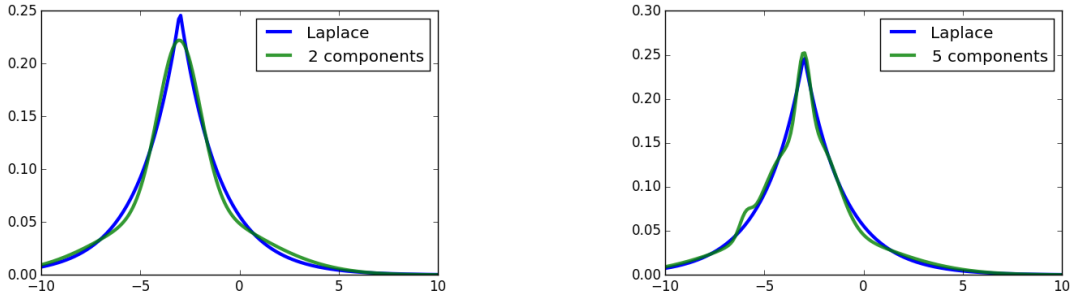


Figure 5.3: Two components are too few to recreate the peak of a Laplace distribution

Yet they all share the property, that they have corners, points where the probability density more or less abruptly becomes 0. Modelling these points is hard. There is a trade-off between the number of components and the smoothness of the approximation. You may get a reasonable approximation with a handful of components, but it will not model the corner very well, because the few components will have to be broad and smooth to model the bulk of the probability mass. The corners may be very important though. If your random variable is Gamma distributed, you really do not want any probability mass in $\mathbb{R}_{\leq 0}$. And a high number of components will accomplish that. It will, however, also make the approximation a lot less smooth, because the corner has to be modelled by approaching it with progressively more peaked components.

An interesting observation is, that you need less components to model a 90° corner than if it were of another degree.

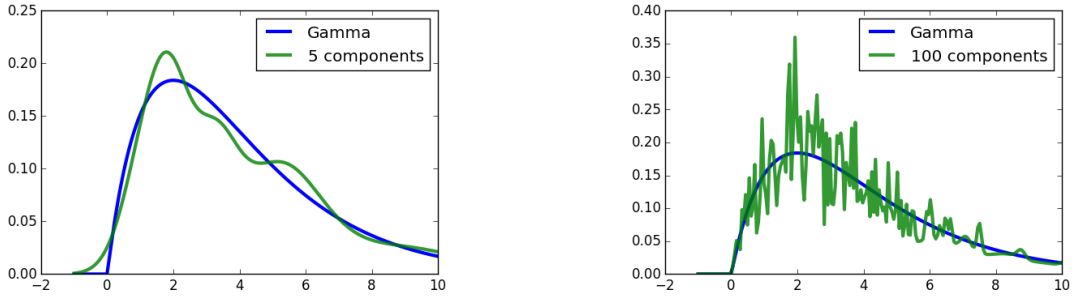


Figure 5.4: You can have few smooth components or a good fit at the corner

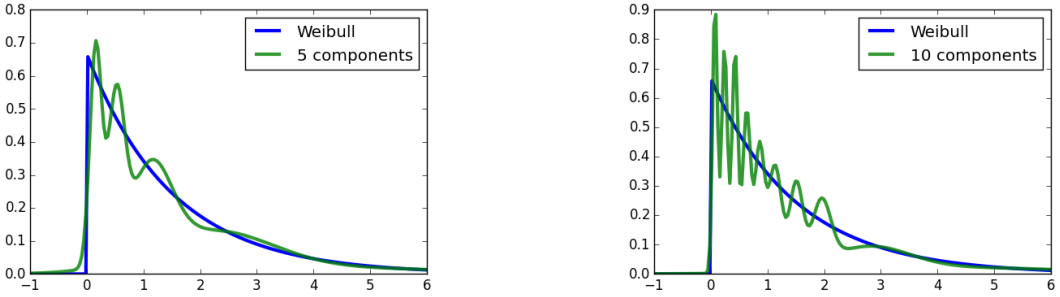


Figure 5.5: Recreating a 90° corner is easier

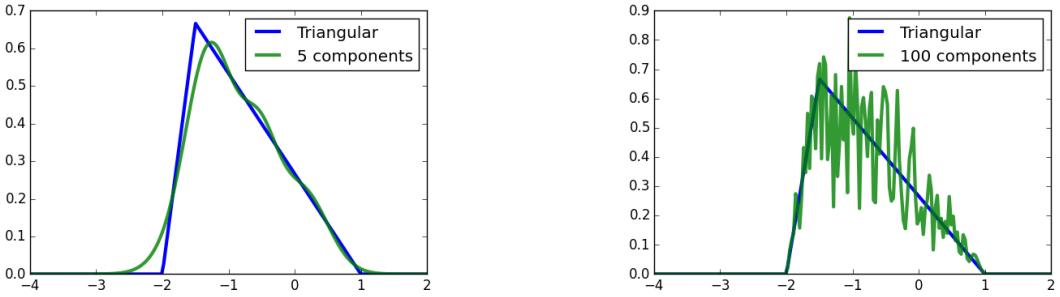


Figure 5.6: Double the corners, double the problems

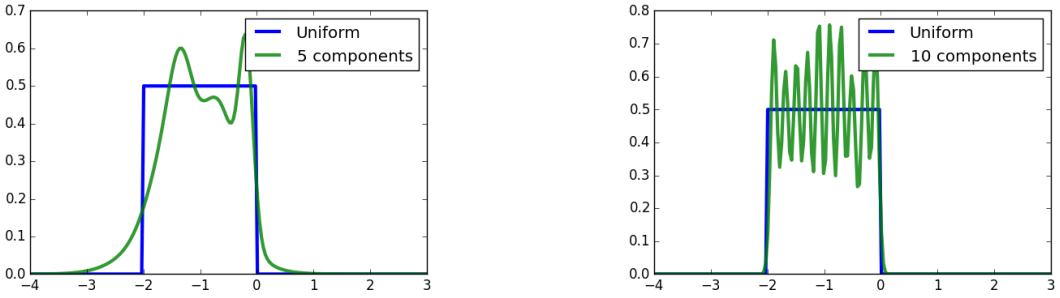


Figure 5.7: Modelling a uniform distribution works surprisingly well

5.2 Operations on Mixtures of Gaussians

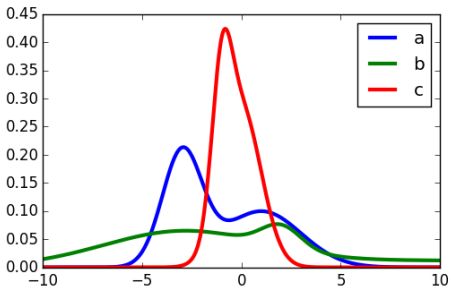
Next we examined arithmetic operations on mixtures of Gaussians, which we will showcase in this section.

5.2.1 Affine Transformations

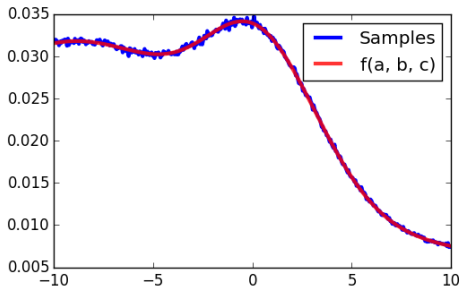
Affine transformations are less interesting in the context of this thesis, because they can be calculated exactly through parameter transformations. We will plot one anyway to demonstrate, that the attached library can correctly compute it. In figure 5.8 we apply the affine transformation

$$f(a, b, c) = \frac{3a}{5} + 2b - c - 3$$

to the variables plotted in figure . The computed mixture plotted in figure 5.8b is congruent to the sampled PDF as expected.



(a) The variables to transform



(b) The resulting density on top of the sampled density

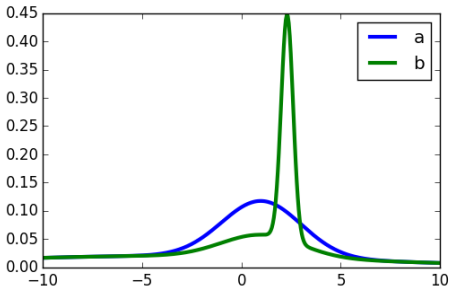
Figure 5.8: Affine transformations are calculated exactly

5.2.2 Products with Gauss-Hermite

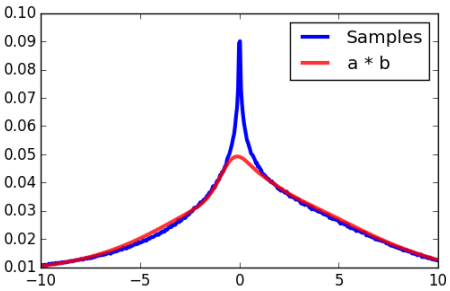
Multiplication with Gauss-Hermite works quite well for the most part as displayed in figure 5.9. Towards the edges it closely approximates the sampled PDF approximation, but the peak in the middle cannot easily be recreated. You have to use really high numbers of components as we did in figure 5.10 to sort this out.

5.2.3 Products with Gauss-Laguerre

As our quick test in section 3.3.3 already showed, this does not really work right now. At least the results are far worse than the ones obtained through Gauss-Hermite. It produces a shape similar to the sampled PDF, but overshoots around the instability (figure 5.11). The positive thing to take away from this is, that the idea to ignore an ε-strip has the desired effect of controlling the error around the instability (figure 5.12).

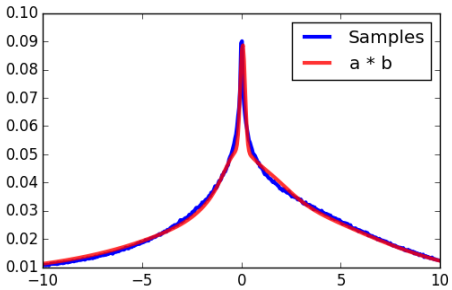


(a) The variables to multiply

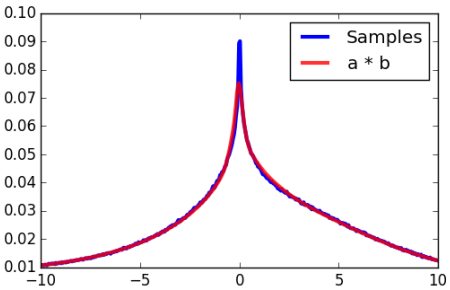


(b) Approximating the product distribution with 300 components

Figure 5.9: Approximating a product distribution



(a) 600 components

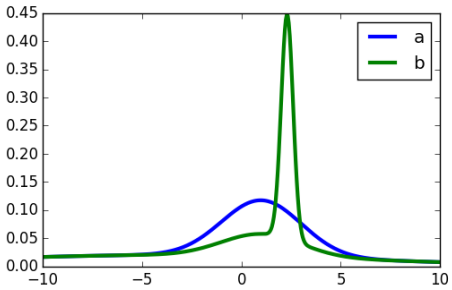


(b) 12000 components

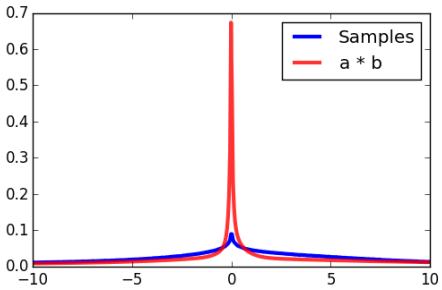
Figure 5.10: You really need a lot of components to model it near the instability

5.2.4 Quotients

In contrast to multiplication division works really well and Gauss-Hermite quadrature produces a good approximation with only a handful of components as shown in figure 5.13.

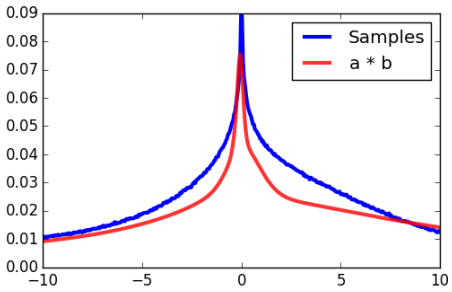


(a) The variables to multiply

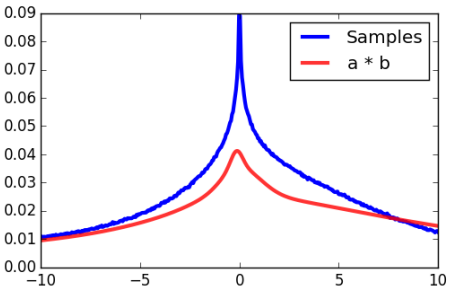


(b) Approximation with $\varepsilon = 0$

Figure 5.11: Gauss-Laguerre produces pretty inaccurate results

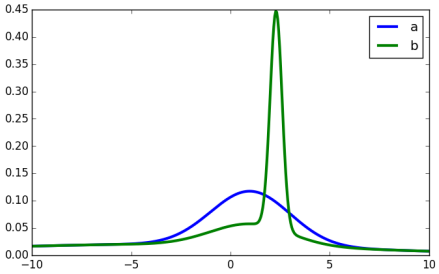


(a) Approximation with $\varepsilon = 0.05$

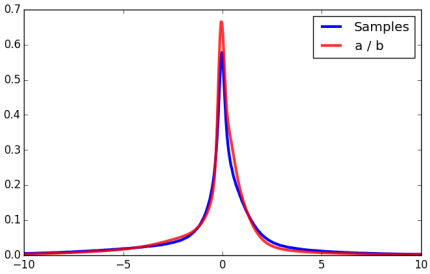


(b) Approximation with $\varepsilon = 0.1$

Figure 5.12: At least the idea works, that you can control the height of the peaks through ε



(a) The variables to divide



(b) Approximation with 12 components

Figure 5.13: Just 12 components give you a good fit already

5.3 Complex Examples

In this section we will present two examples, that are more complex in that they combine the results from the previous two sections.

5.3.1 Introduction

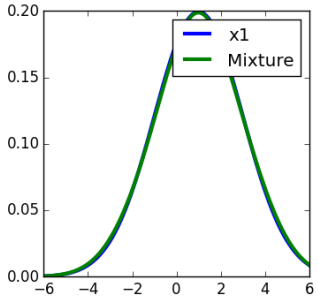
We will first try our library on the example from the introduction, that maple could not work out. Recall that it was about the operation

$$\frac{X_1 \cdot X_2}{Y}$$

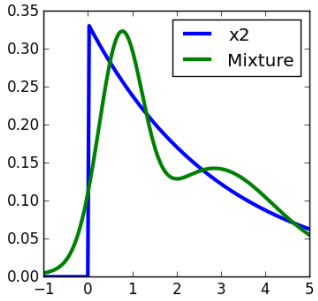
with the random variables distributed as follows

$$X_1 \sim \mathcal{N}(1,2) \qquad X_2 \sim Gamma(1,3) \qquad Y \sim Exponential(2)$$

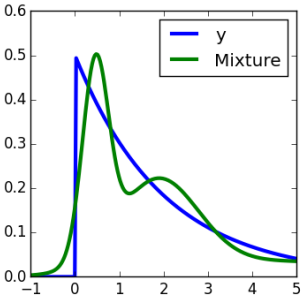
We begin with modelling the inputs as mixtures of Gaussians. However we only use 3 components and model the variables as displayed in figure 5.14. The mixtures recreate the bulk of the probability mass, but as expected the performance around sharp corners is poor with such a low number of components. Surprisingly the actual result in figure 5.15 turns out to look very nice. Even though the inputs were only roughly matched, the end result is almost a pixel perfect match to the sampled PDF.



(a) Gaussian distribution is of course perfectly approximated



(b) Three components only capture the rough outline of a Gamma distribution



(c) The same is true for the exponential distribution

Figure 5.14: The input variables approximated as mixtures of Gaussians

5.3.2 Reciprocal

Approximating the reciprocal of a random variable would make for an interesting experiment. Remember, that this was the only operation, that we could not work out in section 3.3.1. Trying techniques like Gauss-Hermite quadrature on this problem is also impossible, since there is no integral to evaluate. We may however be able to calculate it with a trick. Instead of trying to work out $\frac{1}{X}$, we will use our library to calculate $\frac{U}{X}$, where X is the variable, that we would like to obtain the reciprocal distribution from, and U is a uniform distribution from $1-\varepsilon$ to $1+\varepsilon$. You can see the distributions we used in figure 5.16.

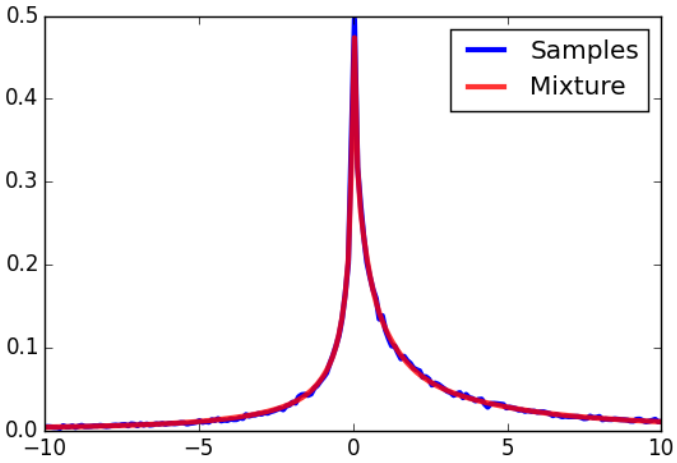
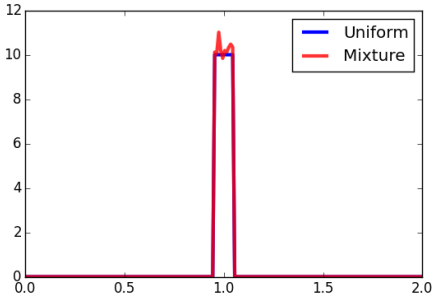


Figure 5.15: Our result closely matches the sampled PDF regardless of the rough fit on the inputs

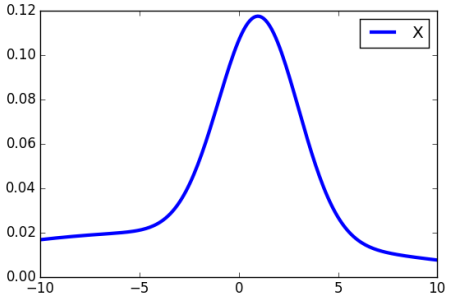
Figure 5.17 shows, that it unfortunately did not work as intended. The picture is nonetheless very interesting, because it actually visualizes, how our algorithm works. Notice, that the peaks all look very similar to our approximation to 1. And indeed, if you recall the actual formula for the quotient approximation from section 3.3.4

$$p(Z=z) \approx \sum_{i=1}^{n} \frac{w_i}{\sqrt{\pi}} \cdot \mathcal{N}\left(z \,\middle|\, \frac{\mu}{\sqrt{2\tau^2}x_i + \nu}, \frac{\sigma^2}{\left(\sqrt{2\tau^2}x_i + \nu\right)^2}\right)$$

you can see, that the approximation's components are all transformed copies of the numerator variable. So it takes the "uniform" distribution and copies it everywhere, but the copied distributions are so narrow, that they have no overlap at all, and it looks like a forest of lonely firs. If you look closely, the components' variances even make sense. The distributions are narrow and high, where the actual distributions is high, and small and wide, where it is flat.



(a) Our approximation to 1



(b) A mixture of Gaussians

Figure 5.16: Our 1 and the variable to divide it by

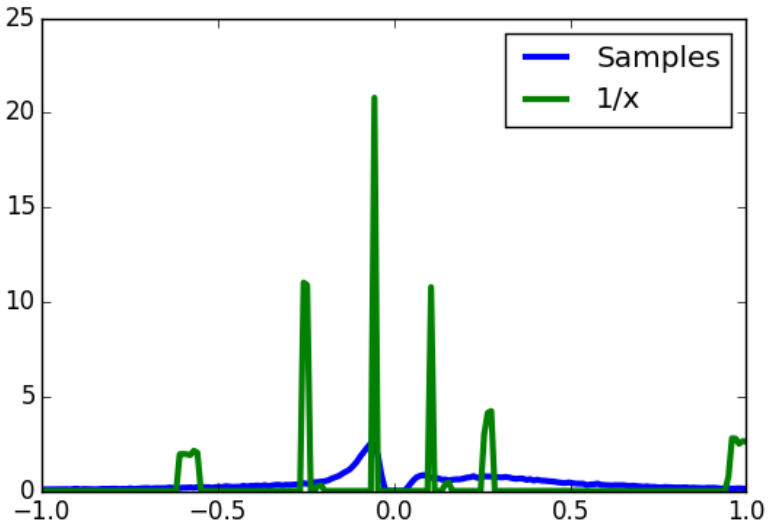


Figure 5.17: Lots of peaks instead of one smooth curve

Chapter 6

Conclusions and Further Work

We have shown, that the way of symbolic approximation works and yields results comparable to ones derived through sampling or symbolic manipulation. It has to be said though, that the scope of this project is currently limited. To really keep up with the competing techniques, mixture approximation needs improvements in several areas ranging from easy to hard.

The first and obvious improvement would be to implement more operations. At the moment we are limited to basic arithmetic, but there surely are more functions, that are differentiable and monotonic in the first argument. A related idea is the implementation of a fallback algorithm for unimplemented operations. A naive approach is to fall back on sampling and sample from the inputs, apply the operation and then fit a new mixture to the samples with EM. But maybe there is another way, that uses the fact, that everything is mixture distributed, more directly and avoids the involvment of sampling.

An orthogonal idea is the use of other component distributions, which we already hinted at throughout section 3.2. The problem with Gaussian components is, that the math only allows to apply functions, that are defined on at least the complete support of the inputs. This forbids things like taking the logarithm of a mixture of Gaussians, because they will always have non-zero probability of being negative no matter what the means and variances are. Gamma and inverse Gaussian distributions are promising candidates, that only allow positive reals, but at the same time have two parameters to influence mean and variance and may thus give similar flexibility as a mixture of Gaussians.

Presumably rather easy would be the generalization to $f : \mathbb{R}^n \to \mathbb{R}^m$, i.e. allow f to have multiple outputs.

From a performance point of view it would be useful to put a bound on the number of components, that a mixture can have. Maybe there is a way to cut components from a mixture and amend the remaining components' parameters, such that the additional error from this operation is controllable and bounded. In the current implementation the product and quotient of two Gaussian variables are approximated by a mixture with n components if you are evaluating it with Gauss-Hermite at n integration points. This means, that the product of three mixtures with a, b and c components results in a mixture with $abcn^2$ components, which a huge number since n is normally in the low hundreds.

The hardest part with the biggest payoff at the same time would be extending our techniques to dependent variables. You would have to redo almost everything in much more complex,

because lemma 4 does not hold for dependent variables. The upside is, that you will be able to evaluate, for example, the square of a random variable and generally terms with variables that are dependent on each other in non-trivial ways.

Appendix A

Introduction to Random Variables

In this thesis we work with random variables. Random variables are mathematical objects, that do not have a definite value, but rather take on specific values with a certain probability. The probability for a specific value is given by a function, called the probability distribution, which is associated to the random variable.

There are infinitely many functions, that are valid probability distributions, but only some of them are important and get a name. One of them is the Gaussian distribution, symbolized with $\mathcal{N}(\mu, \sigma^2)$. It has two parameters, the mean μ and the variance σ^2. The former tells you, which value is the most probable, while the latter says, how fast numbers get less probable the farther they are from μ.

The notation $X \sim \mathcal{N}(\mu, \sigma^2)$ means, that the random variable X has the associated distribution $\mathcal{N}$ with parameters μ and σ^2. The function, that tells you, how high the probability density is at x, i.e. how probable x is, is written $p(X = x)$. To determine this, p plugs the value x into the associated distribution of X. In this case we have

$$p(X = x) = \mathcal{N}\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

The term in the middle is an alias for $p(X = x)$ if the distribution of X is known or you want to put it into the reader's focus.

Another important term for this thesis is mixture distribution. Assume some random variables $Y_1, \ldots, Y_n$. Now we can define another random variable Y, that, when forced to take on a value, takes on a value from Y_1 with probability α_1, from Y_2 with probability α_2 and so on. This allows Y to have a complex distribution, even when all of its components Y_1 to Y_n only have simple distributions such as Gaussians. There is also a formula for the resulting probability density function of Y.

$$p(Y = y) = \sum_{i=1}^{n} \alpha_i \cdot p(Y_i = y)$$

Appendix B

Source Code

B.1 Transforms.jl

The main file, that defines the central type `RandomVariable` and operations on mixtures independent from the actual component distributions.

```
module Transforms

import Distributions: Distribution, Univariate, Continuous, MixtureModel,
components, probs, component_type, Categorical, mean, var

include("integration.jl")
include("em.jl")

include("components/normal.jl")

"The special case of mixture models, that we work with."
typealias Mixture{T<:Distribution} MixtureModel{Univariate, Continuous, T}

"A random variable with integration parameters."
immutable RandomVariable{Component<:Distribution, T<:IntegrationAlgorithm}
    distribution::Mixture{Component}
    alg::T

    function RandomVariable(distribution::Mixture{Component}, alg::T)
        new(distribution, alg)
    end
end

"A convenience constructor, so you do not have to specify the types."
function RandomVariable{
    T<:Distribution,
    S<:IntegrationAlgorithm}(distribution::Mixture{T}, alg::S)
    RandomVariable{T, S}(distribution, alg)
end
```

```
30
31  "Construct a random variable with a default integration algorithm."
32  function RandomVariable(distribution::Mixture)
33      RandomVariable(distribution, GaussHermiteQuadrature(5))
34  end
35
36  "Approximate an arbitrary distribution with a mixture of Gaussians."
37  function RandomVariable(distribution::Distribution,
38                          samples::Integer, components::Integer,
39                          alg::IntegrationAlgorithm=GaussHermiteQuadrature(5))
40      samples = rand(distribution, samples)
41      mixture = EM!(samples, components)
42
43      RandomVariable(mixture, alg)
44  end
45
46  function mean(x::RandomVariable)
47      mean(x.distribution)
48  end
49
50  function var(x::RandomVariable)
51      var(x.distribution)
52  end
53
54  "Transform a random variable with a monadic function."
55  function monadictransform(op::Function, x::RandomVariable)
56      d = x.distribution
57
58      RandomVariable(Mixture{component_type(d)}(map(op, components(d)), d.prior),
59                     x.alg)
60  end
61
62  -(x::RandomVariable) = monadictransform(-, x)
63
64  +(x::RandomVariable, y::Real) = monadictransform(X -> X + y, x)
65  +(x::Real, y::RandomVariable) = y + x
66  -(x::RandomVariable, y::Real) = x + (-y)
67  -(x::Real, y::RandomVariable) = (-y) + x
68
69  *(x::RandomVariable, y::Real) = y == 0 ? 0 : monadictransform(X -> X * y, x)
70  *(x::Real, y::RandomVariable) = y * x
71  /(x::RandomVariable, y::Real) = x * (1 / y)
72
73  "Transform two random variables with a diadic function."
74  function diadictransform(op::Function, x::RandomVariable, y::RandomVariable)
75      dx = x.distribution
76      dy = y.distribution
77
78      prior = vec(probs(dx) * probs(dy)')
```

```
79      cs = vec([op(i, j) for i = components(dx), j = components(dy)])
80
81      # The type of the resulting components
82      t = typeof(cs[1])
83
84      # Cast from Vector{Any} to Vector{t}
85      cs = t[c for c = cs]
86
87      RandomVariable(Mixture{t}(cs, Categorical(prior)), x.alg)
88  end
89
90  +(x::RandomVariable, y::RandomVariable) = diadictransform(+, x, y)
91  -(x::RandomVariable, y::RandomVariable) = x + (-y)
92
93  """
94  Transform two random variables with a diadic function.
95
96  Use this instead of #{diadictransform}, if op already returns a mixture
97  distribution.
98  """
99  function diadicmixturetransform(op::Function,
100                                  x::RandomVariable, y::RandomVariable)
101     dx = x.distribution
102     dy = y.distribution
103
104     # Weights if the components were not mixtures themselves
105     basicPrior = vec(probs(dx) * probs(dy)')
106
107     # Components approximated by mixtures
108     mixtures = vec([op(x.alg, i, j) for i = components(dx), j = components(dy)])
109
110     prior = vcat([basicPrior[i] * probs(mixtures[i]) for i = 1:length(mixtures)]...)
111     cs = vcat(map(components, mixtures)...)
112
113     # The type of the resulting components
114     t = typeof(cs[1])
115
116     RandomVariable(Mixture{t}(cs, Categorical(prior)), x.alg)
117 end
118
119 *(x::RandomVariable, y::RandomVariable) = diadicmixturetransform(*, x, y)
120 /(x::RandomVariable, y::RandomVariable) = diadicmixturetransform(/, x, y)
121
122 end
```

B.2 em.jl

Our implementation of the EM algorithm.

```
import Distributions: Normal
import StatsBase: mean_and_var

const S2PI = sqrt(2 * pi)

"PDF of a normal distribution."
function p(x::Float64, μ::Float64, σ2::Float64)
    exp(-(x - μ)^2 / (2 * σ2)) / (S2PI * sqrt(σ2))
end

function EM!(X::Vector{Float64}, n::Integer, iters::Integer=20)
    N = length(X)

    # Guess initial model parameters
    X = sort(X)
    k = int(ceil(N / n))
    params = [mean_and_var(X[(1 + (i - 1) * k):min(i * k, end)]) for i = 1:n]
    iμ = map(first, params)
    iσ2 = map(last, params)

    # Initialize model parameters
    π::Vector{Float64} = [1 / n for i = 1:n]
    μ::Vector{Float64} = iμ
    σ2::Vector{Float64} = iσ2

    for j = 1:iters
        # E step
        r = Array(Float64, N, n)

        for i = 1:N
            R = dot(π, [p(X[i], μ[k], σ2[k]) for k = 1:n])

            r[i, :] = π .* [p(X[i], μ[k], σ2[k]) for k = 1:n] / R
        end

        # M step
        rk = [sum(r[:, k]) for k = 1:n]
        nπ = rk / N
        nμ = [dot(r[:, k], X) for k = 1:n] ./ rk
        nσ2 = [dot(r[:, k], (X - μ[k]).^2) for k = 1:n] ./ rk

        π, μ, σ2 = nπ, nμ, nσ2
    end

    Mixture{Normal}([Normal(μ[i], sqrt(σ2[i])) for i = 1:n], Categorical(π))
```

```
46 end
```

B.3 integration.jl

Types for the different integration algorithms and their parameters.

```
1  import FastGaussQuadrature: gausshermite
2  import GaussQuadrature: laguerre
3
4  "Exchangable algorithm for integral approximation"
5  abstract IntegrationAlgorithm
6
7  "Parameters for Gauss-Hermite quadrature"
8  immutable GaussHermiteQuadrature <: IntegrationAlgorithm
9      # Number of points
10     n::Integer
11
12     # Evaluation points for the integrand
13     X::Vector{Float64}
14
15     # Weights for the weighted sum approximation
16     W::Vector{Float64}
17
18     function GaussHermiteQuadrature(n::Integer)
19         (X, W) = gausshermite(n)
20
21         new(n, X, W)
22     end
23 end
24
25 "Parameters for Gauss-Laguerre quadrature"
26 immutable GaussLaguerreQuadrature <: IntegrationAlgorithm
27     # Number of points
28     n::Integer
29
30     # Evaluation points for the integrand
31     X::Vector{Float64}
32
33     # Weights for the weighted sum approximation
34     W::Vector{Float64}
35
36     # How much to ignore around the pole. Has to be >= 0.
37     ε::Real
38
39     function GaussLaguerreQuadrature(n::Integer, ε::Real)
40         (X, W) = laguerre(n, 0.0)
41
```

```
        new(n, X, W, ε)
    end
end
```

B.4 components/normal.jl

The implementation of Gaussian distributions as mixture components.

```
# Implement basic arithmetic on normal distributions

import Distributions: Normal
import CurveFit: linear_fit

function -(x::Normal)
    Normal(-x.μ, x.σ)
end

function +(x::Normal, y::Real)
    Normal(x.μ + y, x.σ)
end

function *(x::Normal, y::Real)
    Normal(y * x.μ, y * x.σ)
end

function +(x::Normal, y::Normal)
    Normal(x.μ + y.μ, sqrt(x.σ^2 + y.σ^2))
end

function *(params::GaussHermiteQuadrature, x::Normal, y::Normal)
    t = sqrt(2) * y.σ
    normals = [Normal((t * ξ + y.μ) * x.μ,
                      abs(t * ξ + y.μ) * x.σ)
               for ξ = params.X]

    Mixture{Normal}(normals, Categorical(params.W / sqrt(pi)))
end

function *(p::GaussLaguerreQuadrature, x::Normal, y::Normal)
    n, W, X, ε = p.n, p.W, p.X, p.ε
    μ, σ, σ2 = x.μ, x.σ, x.σ^2
    ν, τ, τ2 = y.μ, y.σ, y.σ^2

    # Constants and formulas for parameters for the left sum
    lS = (ε + ν)^2
    lT = lS / (2 * τ2)
    lnumerator = exp(-lT)
```

```
40      lw(w, x) = w * lnumerator / (2 * sqrt(pi * (x + lT)))
41      lμ(x) = -(sqrt(2 * τ2 * x + lS) - ν) * μ
42      lσ(x) = (sqrt(2 * τ2 * x + lS) - ν) * σ
43
44      lWeights = Float64[lw(W[i], X[i]) for i = 1:n]
45      lComponents = Normal[Normal(lμ(x), lσ(x)) for x = X]
46
47      # Constants and formulas for parameters for the right sum
48      rS = (ε - ν)^2
49      rT = rS / (2 * τ2)
50      rnumerator = exp(-rT)
51      rw(w, x) = w * rnumerator / (2 * sqrt(pi * (x + rT)))
52      rμ(x) = (sqrt(2 * τ2 * x + rS) + ν) * μ
53      rσ(x) = (sqrt(2 * τ2 * x + rS) + ν) * σ
54
55      rWeights = Float64[rw(W[i], X[i]) for i = 1:n]
56      rComponents = Normal[Normal(rμ(x), rσ(x)) for x = X]
57
58      w = [lWeights, rWeights]
59      c = [lComponents, rComponents]
60
61      # Normalize the weights to 1
62      w = w / sum(w)
63
64      Mixture{Normal}(c, Categorical(w))
65  end
66
67  function /(params::GaussHermiteQuadrature, x::Normal, y::Normal)
68      normals = [Normal(x.μ / (sqrt(2) * y.σ * ξ + y.μ),
69                        abs(x.σ / (sqrt(2) * y.σ * ξ + y.μ)))
70                 for ξ = params.X]
71
72      Mixture{Normal}(normals, Categorical(params.W / sqrt(pi)))
73  end
```

Bibliography

[church] *Church programming language.* URL: http://projects.csail.mit.edu/church/wiki/Church.

[distr] *A Julia package for probability distributions and associated functions.* URL: https://github.com/JuliaStats/Distributions.jl.

[fgq] *Gauss quadrature nodes and weights in Julia.* URL: https://github.com/ajt60gaibb/FastGaussQuadrature.jl.

[gh] *Source Code Repository of this Project.* URL: https://github.org/cqql/Transforms.jl.

[gq] *Julia package to compute weights and points for the classical Gauss quadrature rules.* URL: https://github.com/billmclean/GaussQuadrature.jl.

[jl] *Julia programming language.* URL: http://julialang.org.

[Kre05] Ulrich Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik.* 2005.

[maple] *Maple.* URL: http://www.maplesoft.com/products/maple/.

[Mur12] Kevin P. Murphy. *Machine Learning - A Probabilistic Perspective.* 2012.

[ppl] *Probabilistic Programming.* URL: http://probabilistic-programming.org/wiki/Home.

[stbs] *Basic statistics for Julia.* URL: https://github.com/JuliaStats/StatsBase.jl.

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel verfasst und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, als solche kenntlich gemacht habe.

Düsseldorf, 15. Juli 2015 Marten Lienen