Quadratic Forms in Convolutional Neural Networks

Peter Adema
14460165

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 900
1098 XH Amsterdam

Supervisor
Dr. ir. R. van den Boomgaard

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

Semester 2, 2024-2025

Chapter 1

Introduction

. . . . . .

Chapter 2

Background

First, some mathematical formalisms must be covered to understand the concepts discussed in the implementation and results sections, namely fields and semifields, construction of positive definite matrices and efficient calculation of quadratic forms.

2.1 Convolutional stencil

At the core of a convolutional neural network is the convolutional operation $f * g$. The general form of a continuous convolution of functions f and g can be written as:

$$(f * g)(x) = \int_{y \in \mathcal{D}} f(x - y)\; g(y),$$

where $\mathcal{D}$ is the (continuous) domain of f and g. However, save for a handful of functions whose convolutions can be calculated algebraically, we typically are required to approximate this convolution in the discrete domain. In this case, we approximate the functions f and g by sampling them at fixed intervals, the result of which can be represented as discrete arrays F and G. A discrete convolution could then be written as [1]:

$$(F * G)[x] = \sum_{y \in \mathcal{I}} F[x - y]\; G[y],$$

where $\mathcal{I}$ is the set of all indices in the domain of F and G (e.g. $\mathcal{I} = \mathbb{Z}^2$ for infinitely large 2D images and kernels).

An access pattern such as this, where the new value of a pixel depends on a fixed window of its neighbours, is referred to as a stencil computation [2], and as such, the stencil computation performed by the discrete convolution operator will hereafter be referred to as the convolutional stencil, while the discrete G will be referred to as the (convolutional) kernel.

2.2 Fields, semifields and weighted reductions

In the convolutional stencil, a part of the image is multiplied element-wise with a kernel, and the resulting values are summed to obtain an activation for that point. However, while we typically use scalar addition and multiplication in this calculation, it is also possible to use different operators in the reduction by defining a different field in which the reduction is done. In this section, we will briefly look at the concept of fields insofar as they are relevant to implementing an alternative version of the convolution operator.

In mathematics, a field is a set of values with a pair of operators that work on those values: one operator corresponds to the concept of addition, and one operator corresponds to the concept of multiplication. Fields are, in effect, a generalisation of standard addition and multiplication on integers or reals and allow for describing a set of values other than typical scalars, or an alternate method for combining typical numbers. Formally, a field can be described as a tuple $(\mathcal{F}, \oplus, \otimes)$, where the operators $\oplus$ and $\otimes$ are of the type $\mathcal{F} \times \mathcal{F} \to \mathcal{F}$. Furthermore, the operators $\oplus$ and $\otimes$ are both beholden to the field axioms: informally, a set of rules to ensure they act ’similarly’ to standard scalar addition and multiplication.

These field axioms can be written as (adapted from [3], page 120):

$$\begin{align}
\oplus \text{ is associative: } & \forall a, b, c \in \mathcal{F} \quad a \oplus (b \oplus c) = (a \oplus b) \oplus c \tag{2.1}\\
\otimes \text{ is associative: } & \forall a, b, c \in \mathcal{F} \quad a \otimes (b \otimes c) = (a \otimes b) \otimes c \tag{2.2}\\
\oplus \text{ is commutative: } & \forall a, b \in \mathcal{F} \qquad a \oplus b = a \oplus a \tag{2.3}\\
\otimes \text{ is commutative: } & \forall a, b \in \mathcal{F} \qquad a \otimes b = a \otimes a \tag{2.4}\\
\oplus \text{ has an identity: } & \exists \mathbb{0} \ \forall a \in \mathcal{F} \quad a \oplus \mathbb{0} = a \tag{2.5}\\
\otimes \text{ has an identity: } & \exists \mathbb{1} \ \forall a \in \mathcal{F} \quad a \otimes \mathbb{1} = a \tag{2.6}\\
\oplus \text{ has inverse elements: } & \forall a \ \exists b \in \mathcal{F} \quad a \oplus b = \mathbb{0} \tag{2.7}\\
\otimes \text{ has inverse elements: } & \forall a \ \exists b \in \mathcal{F} \quad a \otimes b = \mathbb{1} \tag{2.8}\\
\otimes \text{ distributes over } \oplus\text{: } & \forall a, b, c \in \mathcal{F} \quad a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \tag{2.9}
\end{align}$$

One use case for fields in machine learning is to describe a weighted reduction (as in a kernel-based convolution) more generally. To better understand this, suppose we have a sequence of N numbers $\mathbf{a} \in \mathcal{F}^N$ (e.g. $[10, 12, 17]$), which we wish to summarise into a single value. This $\mathbf{a}$ could then represent the (flattened) neighbourhood around x that is extracted during the calculation of the correlational stencil $(f \star g)(x)$. We could accomplish the summarisation by repeatedly applying an operator $\oplus$ (e.g. $(10 \oplus 12) \oplus 17$): this would then be referred to as a reduction with $\oplus$ and could also be written as $\bigoplus_{i \in \mathcal{I}} \mathbf{a}[i]$. Typically, we also require that $\oplus$ be associative and commutative (an abelian group, see [3]), enabling the parallelisation of the reduction by reassociating the $\oplus$ and performing some operations out of order [4]. However, we may also wish to weigh some terms of the sequence more heavily in the summary: for this purpose, we could use a second operator $\otimes$ with a second sequence of weights $\mathbf{w} \in \mathcal{F}^N$ (as with the kernel in a convolution or correlation) and write $\bigoplus_{i \in \mathcal{I}} \mathbf{a}[i] \otimes \mathbf{w}[i]$.

We can see that a weighted reduction can be implemented on a field $(\mathcal{F}, \oplus, \otimes)$, as the requirements for a weighted reduction are few and a subset of the field axioms. Therefore, defining a field with appropriate operators is a sufficient condition for performing a weighed reduction. However, it is not a necessary condition: as defined above, the field axioms are overly strict compared to what is necessary for the weighted reduction, and we could relax some assumptions to obtain, e.g., a semiring. In practice, though, the operators typically used for reduction fulfil most, if not all, of the requirements for a field. As such, the spaces in which reductions are performed are typically described as a semifield: a field that does not neccesarily have an additive inverse, relaxing assumption

2.7 from the previous definition [5].

Finally, since weighted reduction can be performed in any semifield, we can perform an operation similar to the convolutional stencil in any semifield. If we see $\mathbf{a}$ as an image neighbourhood around x from f, and $\mathbf{w}$ as a kernel g, then we can see

$$\mathbf{a}[i] = f(x+i) \text{ and } \mathbf{w}[i] = g(i), \text{ so}$$
$$\bigoplus_{i \in \mathcal{I}} \mathbf{a}[i] \otimes \mathbf{w}[i] = \bigoplus_{y \in \mathcal{I}} f[x+y] \otimes g[y]$$

This looks very similar to a correlation, and we can therefore chose to define the (discrete) semifield correlation operator $\circledast$ to mean exactly this:

$$(f \star g)[x] = \sum_{y \in \mathcal{I}} f[x+y]\ g[y]$$
$$(f \circledast g)[x] = \bigoplus_{y \in \mathcal{I}} f[x+y] \otimes g[y] \text{ in some semifield } (\mathcal{F}, \oplus, \otimes)$$

2.3 Nonlinear semifields from mathematical morphology

Knowing that convolutional stencils are weighted reductions and that weighted reductions can be implemented in all semifields, we can examine if there are other semifields in which we can perform a convolution or correlation than the standard linear field. For this, we can take inspiration from mathematical morphology, the mathematics of object and function shapes.

Two core operations from mathematical morphology are dilation and erosion, where dilation corresponds with 'making a function larger' (scaling the umbra of a function), and erosion is 'making a function smaller'. The result of dilation is shown in Fig. 3.1. Examining the local effects of dilation more closely, we can see that it is somewhat similar to taking a local maximum. This similarity can be made more concrete by understanding the formula for discrete dilation:

$$f \boxplus g, \text{ where } (f \boxplus g)[x] = \max_{y \in \mathcal{I}} (f(x+y) + g(y))$$

Is there a need to use the supremum in the discrete case?

Here, f is the function (or object or image) to be dilated, and g is a concave structuring function describing how the dilation will occur. An intuitive explanation would be to see the structuring function g as a (negated) distance function and dilation $\boxplus$ as the operation that takes the highest value also 'close' to x. If g is a step function with a value zero near its centre and $-\infty$ outside (Fig. 3.1, first row), we can see that this is precisely taking the maximum value within the range specified by g. However, we may also wish to use a quadratic (Fig. 3.1, second row) or other concave function as g. Depending on f, we may still be able to perform the dilation with an arbitrary g (using algebraic solutions, or leveraging separability), but to compute the dilation in the general case we may wish to clip all functions g to be above $-\infty$ in only a constrained domain (Fig. 3.1, third row). The dilation with the clipped version of $f \boxplus g_{clipped}$ could then be seen as an approximation of the dilation $f \boxplus g$ with the full (unclipped) g, while having the advantage that the set of relevant indices $\mathcal{I}$ is bounded in size.

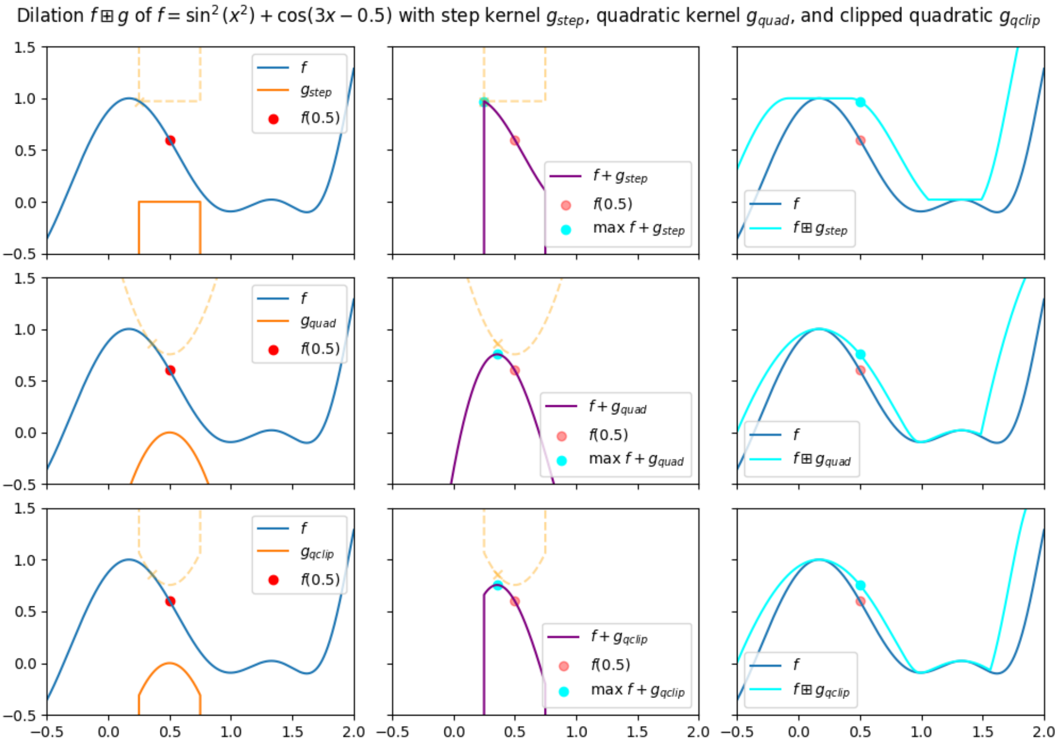

Figure 2.1: Illustration of the effects of dilation with three kernels on a sinusoidal f. $g_{step} = 0$ in a region of size 0.5, $-\infty$ outside. $g_{quad} = -5x^2$, and $g_{qclip} = g_{quad} + g_{step}$.
An alternative intuition for dilation is also illustrated, corresponding with 'lowering' a negated version of g' down towards the point x until it intersects f, and taking the value of the lowered and flipped $g'(x)$ as the result of the dilation at point x.

Looking at the operation performed by dilation more closely, we can see that it is, in effect, a maximum operation weighted by a distance function. Similarities with what was discussed in the previous section may lead us to believe that this can also be seen as a weighted reduction in an appropriate subspace, and this is indeed the case. By defining $\oplus = \max$ and $\otimes = +$, we obtain the tropical max semifield $T_+ = (\mathbb{R} \cup \{-\infty\}, \max, +)$ with neutral elements $(\mathbb{0} = -\infty, \mathbb{1} = 0)$ [5]. A correlation $f \circledast g$ in this tropical subspace T_+ would then be:

$$\begin{aligned}(f \circledast g)[x] &= \bigoplus_{y \in \mathcal{I}} f[x+y] \otimes g[y] \\ &= \max_{y \in \mathcal{I}} \left(f[x+y] + g[y]\right) \qquad \text{(in } T_+\text{)} \\ &= (f \boxplus g)[x]\end{aligned}$$

This result is interesting because we can see the standard max pooling layer in a convolutional neural network as a dilation with a fixed, step-function-like g (a 2D version of g_{step} from Fig. 3.1). A logical next step might then be to examine the effects of using a different structuring function for the pooling layer, and in subsequent sections we will do exactly that for the 2D quadratic structuring function.

It can also be shown that erosion ('shrinking a function') corresponds with a minimum weighted by a (negated) distance function:

$$(f \boxminus g)[x] = \min y \in \mathcal{I}(f[x+y] - g[y])$$

Using similar logic as above, we can show that in the corresponding tropical min semifield $T_{-} = (\mathbb{R} \cup \{\infty\}, \min, +)$ with neutral elements $(\mathbb{0} = \infty, \mathbb{1} = 0)$, the correlation with $g_{neg}(x) = -g(x)$ is equivalent to erosion:

$$\begin{aligned}(f \circledast g_{neg})[x] &= \bigoplus_{y \in \mathcal{I}} f[x+y] \otimes g_{neg}[y] \\ &= \min y \in \mathcal{I}(f[x+y] + g_{neg}[y]) \\ &= \min y \in \mathcal{I}(f[x+y] - g[y]) \\ &= (f \boxminus g)[x]\end{aligned}$$

2.4 Other nonlinear fields

Log and root [5] ......

2.5 Quadratic distance functions

In general, we may wish to weigh dimensions differently. ......

2.6 Learning positive definite matrices

While there are many methods for parameterising a 2 × 2 positive definite matrix, since the calculation of the distance as used in the quadratic distance function is equivalent to the Mahanalobis distance we may wish to view the matrix as a covariance matrix $\Sigma \in \mathbb{R}^{2\times 2}$. Since Σ must be symmetric, we know by the spectral theorem that Σ is diagonalisable [6]:

$$\begin{gathered}\text{For some orthogonal matrix } Q \in \mathbb{R}^{2\times 2}\text{, and} \\ \text{for some diagonal matrix } D \in \mathbb{R}^{2\times 2}\text{,}\end{gathered}$$

$$\Sigma = QDQ^{T} \tag{2.10}$$

$$= Q \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} Q^{T} \tag{2.11}$$

Here, Q (as an orthogonal matrix) can either be a rotation or reflection. However, since σ_1 and σ_2 are freely chosen, fixing Q to be a rotation does not reduce expressivity (see Appendix 5.1). As such, we can write:

$$\Sigma = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{2.12}$$

This parameterisation can be efficiently inversed for the quadratic form, as

$$\begin{aligned}
\Sigma^{-1} &= (QDQ^T)^{-1} \tag{2.13}\\
&= (Q^T)^{-1}D^{-1}Q^{-1} \tag{2.14}\\
&= Q\begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} Q^T \tag{2.15}
\end{aligned}$$

In order for Σ to be positive-definite, σ_1^2 and σ_2^2 are required to be strictly positive, while there are no constraints on ϕ. As such, for any $\boldsymbol{\theta} \in \mathbb{R}^3$:

$$\begin{aligned}
\text{Let } \boldsymbol{\theta} &= \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \log\sigma_1 \\ \log\sigma_2 \\ \phi \end{bmatrix}, \text{ then a valid } \Sigma^{-1} \text{ would be} \tag{2.16}\\
\Sigma^{-1} &= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 \\ \sin\theta_3 & \cos\theta_3 \end{bmatrix} \begin{bmatrix} e^{-2\theta_1} & 0 \\ 0 & e^{-2\theta_1} \end{bmatrix} \begin{bmatrix} \cos\theta_3 & \sin\theta_3 \\ -\sin\theta_3 & \cos\theta_3 \end{bmatrix} \tag{2.17}
\end{aligned}$$

Since we placed no assumptions on $\boldsymbol{\theta}$, it is safe for a black-box or gradient-based optimiser to adjust in any direction, as the resulting Σ will always be always a positive definite matrix.

This parameterisation has the advantage of being easily interpretable: e^{θ_1} and e^{θ_2} are the standard deviations in the first and second principal axis of the quadratic, while θ_3 is the angle the first principal axis forms with the x-axis.

Chapter 3

Experiments

....

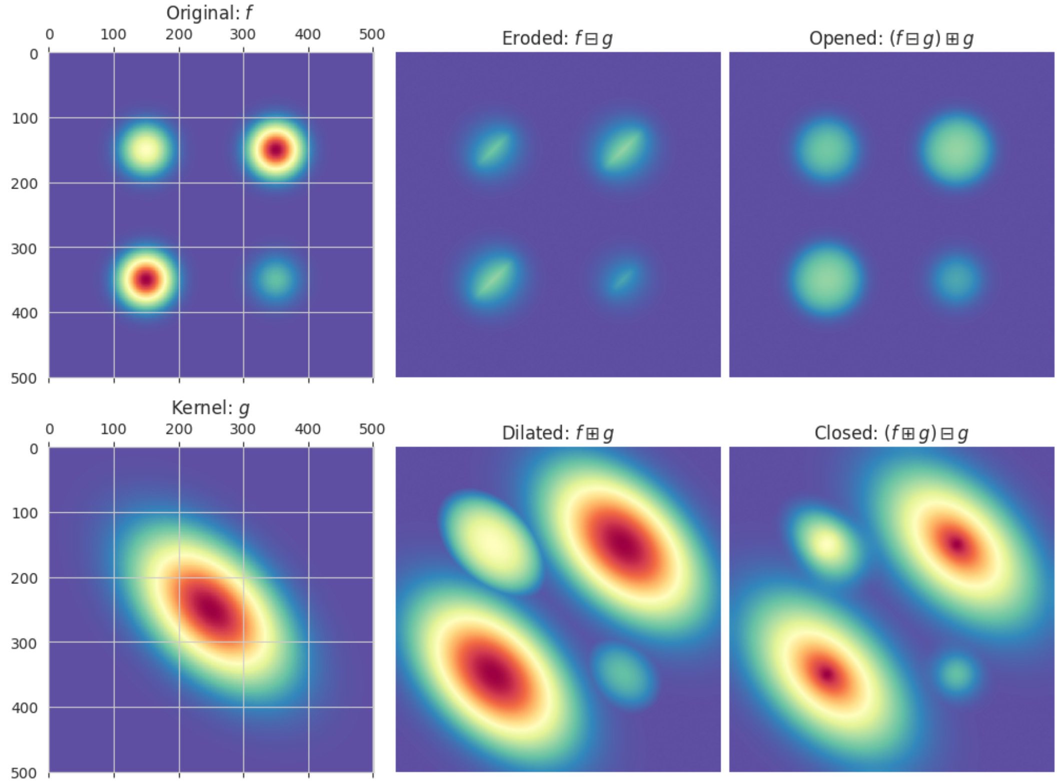

Figure 3.1: Illustration of the effects of dilation, erosion and their combination in two dimensions.

Chapter 4

Conclusion

Bibliography

[1] R. Szeliski, *Computer vision: algorithms and applications.* Springer Nature, 2022.

[2] G. Roth, J. Mellor-Crummey, K. Kennedy, and R. G. Brickner, "Compiling stencils in high performance fortran," in *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*, SC '97, (New York, NY, USA), p. 1–20, Association for Computing Machinery, 1997.

[3] J. A. Beachy, *Abstract Algebra (Third Edition).* 2006.

[4] A. Paszke, M. J. Johnson, R. Frostig, and D. Maclaurin, "Parallelism-preserving automatic differentiation for second-order array languages," in *Proceedings of the 9th ACM SIGPLAN International Workshop on Functional High-Performance and Numerical Computing*, pp. 13–23, 2021.

[5] G. Bellaard, S. Sakata, B. M. N. Smets, and R. Duits, "Pde-cnns: Axiomatic derivations and applications," 2024.

[6] D. Poole, "Linear algebra: A modern introduction," 2015.

Chapter 5

Appendix

5.1 Redundancy of mirroring in QDQ^T

Suppose we had some positive definite $\Sigma \in \mathbb{R}^{2\times 2}$, we could then use orthogonal diagonalisation to write $\Sigma = QDQ^T$ for some orthogonal Q and diagonal D.

In 2.6, the claim was made that requiring Q to be a rotation (and not a reflection) did not decrease expressivity of the representation, i.e. all positive definite Σ are representable as RDR^T with R being a rotation. To show this, we can suppose some Q that represents a reflection, and see that Q can be written as a rotation with angle ϕ (R_ϕ) of a reflection in the x-axis [6]:

$$Q = \begin{bmatrix} \cos\phi & \sin\phi \\ \sin\phi & -\cos\phi \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5.1}$$

Then, we can write out the orthogonal diagonalisation using 5.1:

$$\begin{aligned} \Sigma &= QDQ^T & (5.2) \\ &= \left(R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\right) D \left(R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\right)^T & (5.3) \\ &= R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} R_\phi^T & (5.4) \\ &= R_\phi \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} R_\phi^T & (5.5) \\ &= R_\phi D R_\phi^T & (5.6) \end{aligned}$$

□

5.2 Alternative parameterisation for $\Sigma \in \mathbb{R}^{2\times 2}$

A different way of parameterising the covariance matrix used for modelling the quadratic form would be to use the Pearson correlation coefficient ρ instead of the angle ϕ:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{bmatrix}, \text{ still with parameter vector } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}.$$

The relevant constraints are then:

$$\begin{aligned} \sigma_1 > 0, &\text{ so we can use } \sigma_1 = \exp(\theta_1) \\ \sigma_2 > 0, &\text{ so we can use } \sigma_2 = \exp(\theta_2) \\ \rho \in (-1, 1), &\text{ so we can use } \rho = \tanh(\theta_3) \end{aligned}$$

We may then wish to keep the covariance matrix in its Cholesky decomposed form, where we find an lower triangular L such that $\Sigma = LL^T$:

$$\begin{aligned} \Sigma &= \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{bmatrix} \\ &= LL^T = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix} \\ &= \begin{bmatrix} l_{11}^2 & l_{11}l_{21} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 \end{bmatrix} \end{aligned}$$

As such, we know that:

$$\begin{aligned} l_{11} &= \sqrt{\sigma_1^2} = \sigma_1 = \exp(\theta_1) \\ l_{21} &= \frac{\sigma_1\sigma_2\rho}{\sigma_1} = \sigma_2\rho = \exp(\theta_2)\tanh(\theta_3) \\ l_{22} &= \sqrt{\sigma_2^2 - \sigma_2\rho} = \sqrt{\exp(2\theta_2) - \exp(\theta_2)\tanh(\theta_3)} \end{aligned}$$

which is then a valid parameterisation for the Cholesky decomposed form (the higher-dimensional version of this parameterisation of L based on Σ corresponds with the Cholesky-Banachiewicz algorithm for the Cholesky decomposition). If we keep the covariance matrix in this triangular form, we can see the quadratic form can be calculated in a more efficient manner:

(based on the PyTorch code for the multivariate normal PDF)

$$\begin{aligned} \mathbf{x}^T\Sigma^{-1}\mathbf{x} &= \mathbf{x}^T(LL^T)^{-1}\mathbf{x} \\ &= \mathbf{x}^T(L^T)^{-1}L^{-1}\mathbf{x} \\ &= \mathbf{x}^T(L^{-1})^TL^{-1}\mathbf{x} \\ &= ((L^{-1})\mathbf{x})^T(L^{-1}\mathbf{x}) \\ &= (L^{-1}\mathbf{x}) \cdot (L^{-1}\mathbf{x}) \\ \text{Suppose } \mathbf{b} &= L^{-1}\mathbf{x}, \text{ then} \\ L\mathbf{b} &= \mathbf{x}, \text{ so} \\ \mathbf{b} &= \textsc{solve-triangular}(L, \mathbf{x}) \\ \mathbf{x}^T\Sigma^{-1}\mathbf{x} &= \mathbf{b} \cdot \mathbf{b} \end{aligned}$$

where SOLVE-TRIANGULAR performs efficient backsubstitution to avoid computing the inverse. This method is significantly (>5x) faster on the CPU it was tested on, while still showing modest performance improvements on the GPU it was tested on ($\sim 10\%$). However, interpretation of the Perason correlation coefficient may be more challenging compared to interpreting the angular offset of the first primary axis, and the calculation of the quadratic forms is a negligible part of the model runtime on the GPU, so it was chosen to instead parameterise Σ with the angle ϕ.