Quadratic Forms in Convolutional Neural Networks

Peter Adema

14460165

April 3, 2025

Left blank until I get the official front page.

1   Introduction

......

2   Background

First, some mathematical formalisms must be covered to understand the concepts discussed in the implementation and results sections, namely fields and subfields, 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 $f * g$, where, in practice, g is approximated with a fixed-size discrete kernel. This kernel may be fixed, but most modern machine learning architectures learn the kernel as part of the model's parameters (see O'Shea and Nash (2015) for an introduction). As such, the calculation for the output of a single pixel is of the form:

$$(f * g)[x] = \Sigma_{y \in \mathcal{I}} f[x - y] \cdot g[y], \text{ where } \mathcal{I} \text{ is the set of indices valid for } g,$$

possibly also summing over multiple input channels. 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 (Roth, Mellor-Crummey, Kennedy, & Brickner, 1997), and as such, the stencil computation performed by the discrete convolution operator will hereafter be referred to as the convolutional stencil.

2.2   Fields, subfields 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 the reduction in the convolutional stencil.

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 are as follows (Beachy, 2006):

$$\text{Associativity for both: } (a \circ b) \circ c = a \circ (b \circ c) \tag{1}$$
$$\text{Commutativity for both: } a \circ b = a \circ a \tag{2}$$
$$\oplus \text{ identity: } \exists 0' \, \forall a \in \mathcal{F} \quad a \oplus 0' = a \tag{3}$$
$$\otimes \text{ identity: } \exists 1' \, \forall a \in \mathcal{F} \quad a \otimes 1' = a \tag{4}$$
$$\oplus \text{ has inverse elements: } \forall a \in \mathcal{F} \, \exists b \in \mathcal{F} \quad a \oplus b = 0' \tag{5}$$
$$\otimes \text{ has inverse elements: } \forall a \in \mathcal{F} \, \exists b \in \mathcal{F} \quad a \otimes b = 1' \tag{6}$$
$$\text{Distributivity: } a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \tag{7}$$
$$\tag{8}$$

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 numbers $A \in \mathcal{F}^N$ (e.g. $[10, 12, 17]$), which we wish to summarise into a single value (as with the image neighbourhood in a convolution stencil). We could accomplish this 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 $\bigoplus_{i \in \mathcal{I}} A_i$. Typically, we also require that $\oplus$ be associative and commutative (an abelian group, see Beachy (2006)), enabling the parallelisation of the reduction by reassociating the $\oplus$ and performing some operations out of order (Paszke, Johnson, Frostig, & Maclaurin, 2021). 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 $W \in \mathcal{F}^N$ (as with the kernel in a convolution) and write $\bigoplus_{i \in \mathcal{I}} A_i \otimes W_i$.

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 (Bellaard, Sakata, Smets, & Duits, 2024).

Finally, since weighted reduction can be performed in any semifield, we can perform an operation similar to the convolutional stencil in any semifield. Such a convolution (in a semifield other than the linear) is denoted as $\circledast$ to distinguish it from standard convolution, while correlation (convolution with a mirrored kernel) is denoted as $\star$.

2.3   Nonlinear semifields from computational 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 than the standard linear field. For this, we can take inspiration from computational morphology, the mathematics of object and function shapes.

Two core operations from computational 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 these operators is shown in Fig. 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 dilation:

$$f \boxplus g, \text{ where } (f \boxplus g)(x) = \max_y \left(f(x + y) + g(y) \right)$$

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. 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. 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), 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. 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.
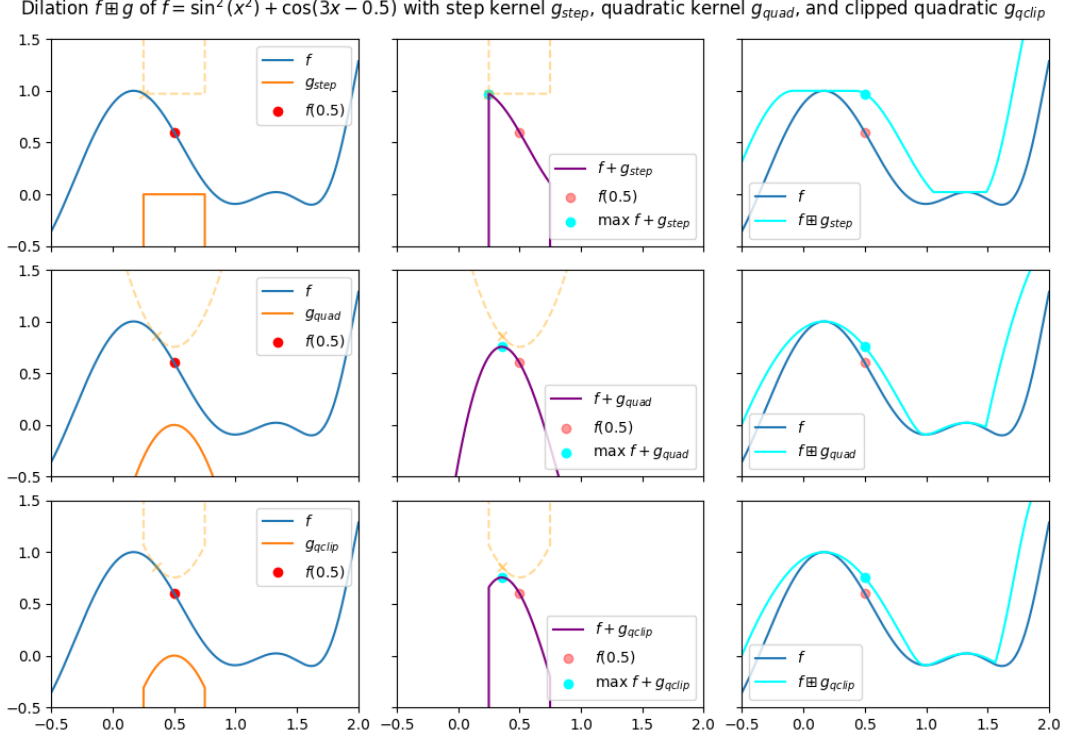

Figure 1: Illustration of the effects of dilation with three kernels on a sinusoidal f. $g_{step} = 0$ in a region of size 0.5, $-\infty$ outside. $g_{quad} = -5x^2$, and $g_{qclip} = g_{quad} + g_{step}$. An alternative intuition for dilation is also illustrated, corresponding with 'lowering' a negated 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 $(-\infty, 0)$ respectively (Bellaard et al., 2024). Dilation, as described above, is then equivalent to a correlation $f \circledast (-g)$ in T_+. Does this need more steps? Probably does ...

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. 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 distance function. Using similar logic as above, erosion $\boxminus$ can be shown to also correspond with a correlation $f \circledast (-g)$ in the tropical min subfield $T_- = (\mathbb{R} \cup \{\infty\}, \min, +)$ with neutral elements $(\infty, 0)$. I write -g here, but g is a function. Should I be more precise?

2.4 Other nonlinear fields

Log and root (Bellaard et al., 2024) ......

2.5 Quadratic distance functions

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

2.6 Learning positive definite matrices

We can parametrise any 2×2 positive definite matrix using the same variables as a 2×2 covariance matrix (which also helps with interpretability). The relevant constraints are then:

$$\sigma_1 > 0, \text{ so we can use } \sigma_1 = \exp(\theta_1) \tag{9}$$

$$\sigma_2 > 0, \text{ so we can use } \sigma_2 = \exp(\theta_2) \tag{10}$$

$$\rho \in (-1, 1), \text{ so we can use } \rho = \tanh(\theta_3) \tag{11}$$

$$\tag{12}$$

Suppose $\Sigma \in \mathbb{R}^{2\times 2}$ is a positive definite matrix, then for some σ_1, σ_2, ρ :

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{bmatrix}, \text{ but also for some lower diagonal } L \in \mathbb{R}^{2\times 2} :$$

$$\Sigma = LL^T = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11}^2 & \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 \end{bmatrix}$$

As such, we know that:

$$l_{11} = \sqrt{\sigma_1^2} = \sigma_1$$

$$l_{21} = \frac{\sigma_1\sigma_2\rho}{\sigma_1} = \sigma_1\sigma_2\rho$$

$$l_{22} = \sqrt{\sigma_2^2 - \sigma_1\sigma_2\rho}$$

......

2.6.1 Efficient quadratic calculation

$$\mathbf{x}^T\Sigma^{-1}\mathbf{x} = \mathbf{x}^T(LL^T)^{-1}\mathbf{x}$$

$$= \mathbf{x}^T(L^T)^{-1}L^{-1}\mathbf{x}$$

$$= (L^{-1}\mathbf{x}) \cdot (L^{-1}\mathbf{x})$$

$$\text{Suppose } \mathbf{b} = L^{-1}\mathbf{x},$$

$$L\mathbf{b} = \mathbf{x}, \text{ so}$$

$$\mathbf{b} = \textsc{solve-triangular}(L, \mathbf{x})$$

$$\mathbf{x}^T\Sigma^{-1}\mathbf{x} = \mathbf{b} \cdot \mathbf{b}$$

Tri-solve ......

References

Beachy, J. A. (2006). *Abstract algebra (third e) john a beachy and william d blair waveland press, inc.(2006)*.

Bellaard, G., Sakata, S., Smets, B. M. N., & Duits, R. (2024). *Pde-cnns: Axiomatic derivations and applications.* Retrieved 2025-03-31, from https://arxiv.org/abs/2403.15182

O'Shea, K., & Nash, R. (2015, 11). An introduction to convolutional neural networks. *ArXiv e-prints*.

Paszke, A., Johnson, M. J., Frostig, R., & Maclaurin, D. (2021). 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).

Roth, G., Mellor-Crummey, J., Kennedy, K., & Brickner, R. G. (1997). Compiling stencils in high performance fortran. In *Proceedings of the 1997 acm/ieee conference on supercomputing* (p. 120). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/509593.509605` doi: 10.1145/509593.509605