Attentional meta-learners are polythetic classifiers

Ben Day, Ramon Viñas, Nikola Simidjievski, Pietro Liò
Department of Computer Science and Technology,
University of Cambridge, Cambridge, UK.
[bd39, rv340, ns779, pl219]@cam.ac.uk

Abstract

Polythetic classifications, based on shared patterns of features that need neither be universal nor constant among members of a class, are common in the natural world and greatly outnumber monothetic classifications over a set of features. We show that threshold meta-learners require an embedding dimension that is exponential in the number of features to emulate these functions. In contrast, attentional classifiers are polythetic by default and able to solve these problems with a linear embedding dimension. However, we find that in the presence of task-irrelevant features, inherent to meta-learning problems, attentional models are susceptible to misclassification. To address this challenge, we further propose a self-attention feature-selection mechanism that adaptively dilutes non-discriminative features. We demonstrate the effectiveness of our approach in meta-learning Boolean functions, and synthetic and real-world few-shot learning tasks.

1 Introduction

Monothetic classifications are based on universal attributes: there is at least one necessary and sufficient attribute for class membership. Polythetic classifications are instead based on combinations of attributes, none of which are sufficient in isolation to indicate membership and, potentially, none of which are necessary. Carl Linnaeus, inventor of the binomial nomenclature for species and “father of taxonomy,” recognised that natural orders could not be defined monothetically, lacking features that were unique and constant over families and that, until such features could be found, such classifications were necessarily polythetic [1,2].

![Botanical and cartoon illustrations](a) Section of Methodus Plantarum Sexualis, Georg Ehret, illustrating Linnaeus’s Systema Naturae, 1735. (b) Intersecting features among natural categories.

Figure 1: Botanical and cartoon illustrations show how class defining attributes need not be exclusive, constant or universal, and that classifications can be contextual.

*Equal contribution.

Preprint. Under review.
Figure 1a illustrates Linnaeus’ system for classifying plants, which relies on polythetic classifications and is still in use today. For example, we can distinguish A from B by the number of filaments, but not B from C; we can distinguish B from C based on whether there are split anthers (the ends), but not C from A, and so on. To recognise classes we are required to consider attributes in the context of other attributes.

Threshold functions are a closely related concept in Boolean algebra, of which monothetic classifications are a subset. Verbally, the difference is between ‘at-least-some-of’ and ‘all-of.’ Formally, a Boolean function of \( n \) inputs \( x_1, x_2, \ldots, x_n \) is a threshold function if, for some threshold \( t \), we can find real-valued weights \( w_1, w_2, \ldots, w_n \) such that
\[
\begin{align*}
    f(x_1, x_2, \ldots, x_n) &= \begin{cases} 
    1, & w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > t \\
    0, & \text{otherwise}
    \end{cases}
\end{align*}
\]
evaluates the function correctly. Figure 2 illustrates the concept for 2-variable Boolean functions. Lacking a preferred feature basis, there is no difference between threshold functions that, strictly speaking, are polythetic, such as \( \text{OR}(x, y) \), and those that are monothetic in the strict sense, such as \( \text{AND}(x, y) \), as we can rotate into a basis where these functions are strictly-monothetic. As such, we can identify monothetic classifications with threshold functions and polythetic classifications with non-threshold functions, and compare their frequency. As the dimensionality increases, the number of Boolean functions grows double-exponentially \( (2^n)^2 \), whereas the number of threshold functions grows exponentially \( 2^{2^n} \), and therefore monothetic classifications represent a vanishingly small proportion of the total (see Appendix A).

Classification meta-learning is typically approached from the perspective of few-shot learning: Can we train a model that generalises to unseen ‘natural’ classes at test time? For example, in the Omniglot task \( \text{[3]} \) we have access to a labelled set of handwritten characters during training and we are tasked with distinguishing new characters, from unseen writing systems, at test time. In this approach a given example always belongs to the same underlying class. Alternatively, we can consider meta-learning over classifications themselves: can we train a model that generalises to unseen ways of categorising? Consider \( \{a, b, d, e, o\} \) and \( \{c, f, h, k, l\} \). We might not have seen letters grouped in this way before, and yet we can be reasonably confident that ‘p’ belongs with the first group and ‘m’ with the second, ‘p’ having one hole like the first and ‘m’ having none like the second. Figure 1b further illustrates how, in this setting, features need to be understood in relation to a given classification: if we are tasked with distinguishing equids (horses, zebras, donkeys) from big cats, the presence of stripes on both zebra and tigers is irrelevant, and potentially misleading. On the other hand, the absence of stripes is the most useful feature when distinguishing horses from zebra.

In this work we consider the limitations of threshold classifiers and how they are able to learn and approximate non-threshold functions in practice; propose using a simple non-parametric alternative

![Figure 2](image-url)

Figure 2: Example threshold and non-threshold functions of 2-variables, the pseudo-variable solution, and the polythetic performance gap for 2-variable functions. (a) A threshold function over two variables, \( \text{AND}(x, y) \), with a solution given by the arrow-line. (b) \( \text{XOR}(x, y) \), which does not have such a threshold solution in 2 dimensions. (c) Appending the pseudo-variable \( \text{XOR}(x, y) \) gives a 3-dimensional embedding in which all 2-variable Boolean functions have threshold solutions. \( \text{XOR}(x, y) \) is a pseudo-variable in that it is determined by the other variables and cannot freely vary, for example the hatched circle at \( (1, 0, 0) \) cannot occur. Right: upper bounds on accuracy and solved-fraction of 2-variable functions by embedding dimension. The performance gap is masked in the accuracy metric by threshold approximations of non-threshold functions.
based on attention that is polythetic; explore the challenges of this approach with regards to spurious feature correlations; and further propose a solution to this challenge that is also non-parametric and based on self-attention. Throughout, we demonstrate the effectiveness of these proposals using synthetic and real-world few-shot learning tasks.

2 Background

We refer to classification functions over \( n \) variables or features simply as classifications, and we are often interested in classifications that only depend on a subset of features, \( \alpha \), and not the remainder, \( \beta = n - \alpha \). Labelled examples provided for a classification are called the support set and the examples we want to predict labels for are called queries. Formal definitions follow where appropriate.

**Non-threshold functions with threshold classifiers.** Perceptrons are not able to learn XOR \(^2\) but deep networks with threshold classifiers can. This is possible because the network can learn additional pseudo-features for the non-threshold functions it observes. Figure 2 gives an example for 2-variables: one can embed the corners of the square at the corners of a tetrahedron with coordinates \((x, y, \text{XOR}(x, y))\) and produce thresholds for every 2-variable Boolean function. There are two problems with this approach, both of which are exacerbated in the meta-learning context: i) the required embedding dimension grows as \( \binom{n}{\alpha} \sim \mathcal{O}(n^\alpha) \), and ii) the method does not generalise to unseen non-threshold functions (see Appendix B).

Prototypical networks \(^3\) approach the problem of few-shot classification with end-to-end trainable feature-extraction and threshold-classification, with great success. The support set is embedded using a learned feature extractor (typically a deep neural network) and averaged within classes to form ‘prototypes.’ Queries are embedded and classified by nearest-neighbours\(^4\), a threshold function in the embedding space. Figure 3 presents examples of the decision boundaries this method induces for 2-variable Boolean functions. Figure 4 shows how this method performs modelling XOR functions of 2-variables, struggling as expected when the embedding dimensions is less than \( n^\alpha \).

**Focus on exclusive-or.** For a binary feature vector \( \mathbf{x} \in \{-1, 1\}^n \), the number \( \chi_A(\mathbf{x}) = \prod_{i \in A} x_i \) is the parity function or exclusive-or (XOR) over the bits \( (x_i)_{i \in A} \). Every function \( f : \{-1, 1\}^n \rightarrow \mathbb{R} \) is a linear combination of the parity functions over \( n \), and they form a linearly independent basis \( \mathbf{6} \). As such, being able to learn the partition functions guarantees that one can learn any other function over that domain. Put another way, the decision boundaries of XOR\(_n\) are at least as complicated as those for any other \( \alpha \)-variable Boolean function: there is one between every possible pair of feature vectors. Additionally, there are no Boolean functions that are less well approximated by a linear

\[^2\] Made differentiable using a softmax over prototype distances.

![Figure 3: Confidence heatmaps and decision boundaries of prototype and attention classifiers for three 2-variable Boolean functions. The attention classifier shown uses a temperature of 1, lower temperatures ‘harden’ the classification and produce decision boundaries more closely aligned with the axes (see Appendix C). As XOR\((x, y)\) is not a threshold function, simple prototypes fail to produce a correct classification scheme, in this case the prototypes are equal \((= (0, 0))\) and there is no decision boundary.]

![Figure 4: Accuracy of prototypical networks for XOR\(_2\) over sequence length and embedding dimension. Mean over 1000 tasks. \(|S| = 40\).]
threshold than XOR, which we discuss in further detail in Appendix D. For these reasons, XOR is our polythetic function of choice in derivations, examples, and experiments.

3 Methods

Problem formulation. We are interested in few-shot classification: provided with a small number of labelled points $S = \{x_i, y_i\}_{i \in I}$, the support set, with feature vectors $x_j \in \mathbb{R}^n$ and labels $y_i \in \{1, \ldots, k\}$, we want to make class predictions for the query set $Q = \{x_j\}_{j \in I}$. $I_S$ and $I_Q$ are index sets for the support and queries, respectively. Importantly, the label space is arbitrary and potentially unique to a task: both the number of classes and assigned labels vary over tasks, and examples that share labels under one classification may not necessarily share labels under another. In this meta-learning setting, we can expect that not all features will play an active role in all classifications.

3.1 Attentional classifiers

We propose that attentional classifiers can resolve the problems associated with threshold classifiers in the context of polythetic classification, particularly for meta-learning. The key advantage of classifiers of this form is that they are polythetic by default, limited only in complexity by the size of the support set as decision boundaries are able to be induced around every support example.

Using an attention mechanism for few-shot classification is straightforward: the queries are the query set features, $Q \in \mathbb{R}^{|Q| \times n}$, the keys are the support set features, $K \in \mathbb{R}^{|S| \times n}$, and the values are one-hot encodings of the (arbitrary) support labels, $V \in \mathbb{R}^{|S| \times k}$. We make use of dot-product and squared Euclidean distance attention mechanisms, defined, with temperature $\tau^{-1}$, as

$$
\text{DotAttn}(Q, K, V, \tau) = \text{softmax}(\tau QK^T)V \in \mathbb{R}^{|Q| \times k},
$$

$$
\text{EuclidAttn}(Q, K, V, \tau) = WV \in \mathbb{R}^{|Q| \times k}; \quad W_i = \text{softmax}_j(-\tau |Q_i - K_j|). \quad (3)
$$

In practice, we are likely to have some learned feature-extraction step, $x' = f(x) \in \mathbb{R}^n'$, which could either be pre-trained or, as the classifier is differentiable, trained end-to-end. Vinyals et al. [7] introduced classifiers of this form, $\hat{y} = \sum_{i \in I_S} a(\hat{x}, x_i)y_i$, as Matching Networks (MN) and provided the conditions under which such a model carries out kernel density estimation or $k$-nearest-neighbours classification. Our application closely matches that of Luong et al. [8] which introduced dot-product attention, though we attend over sets rather than sequences. Vaswani et al. [9] first used explicit scaling in hidden attention layers, setting $\tau = 1/n$, but scaling a classifying softmax by a temperature parameter dates to the 19th century work of Boltzmann and later Gibbs [10]. Plötz and Roth [11] note that their neural-nearest-neighbours (N³) block recovers a soft-attention weighting when the number of neighbours is set to 1, and deploy their model on an outlier-detection set-reasoning task. However, the N³ block is used only as a kind of hidden layer and not as a classifier.

The drawback of using attention in this way is that, as the number of features and the size of the support set increase, strong spurious correlations between queries and members of the support set can come to dominate the classification, as shown in Figure 8 and analysed in detail in the following section. This problem was recognised by Luong et al. [8] and resolved in the sequence setting with ‘local attention’, attending only over a subset of elements within some distance of the target position. Sets do not have such an ordering, so we turn instead to feature selection to resolve the problem.

3.2 Attentional feature selection

In meta-learning (and real-world) settings, a set of classification functions typically will not depend on every feature in every case, and the support set is unlikely to span the domain of possible inputs, neither of which is helpful. The problem is resolved by detecting salient features within and between classes. For monothetic classification functions this is straightforward: by definition, averaging features within a class of the support set highlights necessary features whilst diminishing irrelevant features. Prototype methods rely on this process. In the polythetic case, attentional classifiers have the advantage of being able to learn non-threshold functions without the need for pseudo-variables.

---

3In the technical sense as used with regards to perceptrons not being able to learn non-threshold functions.
but do not benefit, as prototypes do, from ‘washing-out’ irrelevant features through averaging. Indeed, attentional classifiers are susceptible even in the monothetic setting to misclassifying on the basis of closely matching features that are not relevant to the task (putting a zebra with the big-cats because its stripes match those of a tiger in the support set, for example). We present a detailed analysis of this phenomena for Boolean functions to provide insight on the general problem.

Misclassification of binary strings. Consider classifications over binary strings \( x \in \{−1, 1\}^n \), with the class determined by XOR \( \alpha \) over \( \alpha \) elements with the remaining \( \beta = n − \alpha \) being irrelevant. We assume each of the \( 2^n \) variations of the active elements are present with equal frequency, \( r \), for a support set \( S \) of size \( |S| = 2^n \), and that the remaining \( \beta \) elements follow a Bernoulli distribution with probability \( p \). Using an attention classifier of the form \( \hat{y} = \sum_{i \in S} \text{softmax}_i (a(\hat{x}, x_i)) y_i \), where \( a(\hat{x}, x_i) \) is a function of the difference between \( \hat{x} \) and \( x_i \) (e.g. dot-product attention, cosine similarity attention, squared Euclidean attention, Laplace attention), how likely is it that we misclassify a query drawn from the same distribution as \( S \)?

There are \( \binom{n}{\alpha} \) strings over the active elements that differ from a given example in \( \delta \) positions. XOR flips the classification with each difference, so if \( \delta \) is even the class is the same, if \( \delta \) is odd the class is different (parity). Setting the query to be from the positive class without loss of generality, and noting that in the case of binary-classification we care only about the difference in weight for each class and not the scale, in the \( \beta = 0 \) case the classifier rule is equivalent to \( \text{sign}(\sum_{\delta=0}^{\beta} (-1)^\delta \binom{n}{\delta} \exp(f(\delta))) \).

With appropriate choices of feature encodings, \( \{−1, 1\} \) for dot-product or cosine attention, \( \{1, 0\} \) for squared Euclidean or Laplacian attention, \( f(\delta) \) is linear in \( \delta \). Continuing with dot-product attention only for clarity (see Appendix [E] for others and extensive derivations for all that follows), for which \( f(\delta) = \alpha − 2\delta \), the classifier behaves as

\[
\text{class} = \text{sign} \left( \sum_{\delta=0}^{\alpha} r(-1)^\delta \left( \frac{\alpha}{\delta} \right) \exp(\alpha - 2\delta) \right) = \text{sign} \left( e - e^{-1} \right)^\alpha = +, \quad (4)
\]

In the general case, noise is introduced through the contribution of the \( \beta \) irrelevant elements. The contribution of the irrelevant components follows a Binomial distribution with probability \( \tilde{p} \): a pair of irrelevant features are the same, \( (+, +) \) or \( (−, −) \), with probability \( \tilde{p} = p^2 + (1 - p)^2 \), and different with probability \( \tilde{q} = 1 - \tilde{p} \). There are \( \beta \) trials, one for each of the \( \beta \) irrelevant features, and we account for no-matches giving \( (−β) \) and all matches giving \( (+β) \), for a contribution \( 2B(\beta, \tilde{p}) - \beta \), and an overall attention score of \( X(\delta) \sim \exp(\alpha - 2\delta + 2B(\beta, \tilde{p}) - \beta) \), a log-binomial. We find the expectation and variance

\[
\mathbb{E}[X(\delta)] = \sum_{b=0}^{\beta} B(b = b) \exp(\alpha - 2b + 2b - \beta) = \exp(\alpha - 2\delta) \left( \tilde{p}e + \tilde{q}e^{-1} \right)^\beta, \quad (5)
\]

\[
\text{Var}[X(\delta)] = \exp(2\alpha - 4\delta) \left( \left( \tilde{p}e^2 + \tilde{q}e^{-2} \right)^\beta - \left( \tilde{p}e + \tilde{q}e^{-1} \right)^{2\beta} \right). \quad (6)
\]

We can follow Equation (5) to get the expectation and variance of the sum over the support set

\[
\mathbb{E} \left[ \sum_{i \in S} (-1)^\delta X(\delta_i) \right] = \mu = \sum_{\delta=0}^{\alpha} r(−1)^\delta \left( \frac{\alpha}{\delta} \right) \mathbb{E}[X(\delta)] = r(\alpha - e^{-1})^\alpha \left[ (\tilde{p}e + \tilde{q}e^{-1})^\beta \right], \quad (7)
\]

\[
\text{Var} \left[ \sum_{i \in S} X(\delta_i) \right] = \sigma^2 = r(\epsilon^2 + \epsilon^{-2})^\alpha \left[ (\tilde{p}e^2 + \tilde{q}e^{-2})^\beta - (\tilde{p}e + \tilde{q}e^{-1})^{2\beta} \right], \quad (8)
\]

and these suffice to make the key point. Rewriting the terms in square brackets as \( [e^\beta] \) and \( [d^\beta - c^{2\beta}] \), respectively, \( d > c^2 \) over the range of allowed values for \( \tilde{p} \) and the ratio \( \sigma/\mu \propto \exp(1/2 \cdot \ln ((d/c^2)^\beta - 1)) \) i.e. the standard deviation grows exponentially faster in \( \beta \) than the mean. Thus, as \( \beta \) increases the distribution becomes increasingly flat and the rate of misclassification (i.e. the rate at which the sum is negative) increases. We can also see that increasing \( r \), the number of samples of each variation, reduces misclassification as \( \mu \propto r \) but \( \sigma \propto r^{1/2} \), yet having more active elements, \( \alpha \), increases the rate of misclassification as \( (\epsilon^2 + \epsilon^{-2})^{0.5} > (\epsilon - e^{-1}) \). The left plot of Figure [S] attention classification without feature selection, can be understood as sampling this distribution. This is also the expected performance of Matching Networks [3] on these tasks.
Figure 5: Feature values and attention coefficients during the feature-selection self-attention ($n = \{ 0, 5, 10, 25 \}$) within a class of XOR$_2$. Nodes depict examples of the support set: the colour of the left halves represents the active features; the right halves represents the magnitude of the irrelevant features. Edge width and opacity indicate the attention strength between a pair of nodes. The red, green, blue and white groups, different variants, automatically segregate which preserves their active features while the irrelevant features converge, as shown in the feature scores beneath each plot. In this way, the active features identify themselves.

**Feature selection.** Misclassification occurs when irrelevant features overwhelm the signal from the active elements. As this is a problem of highlighting the salient patterns within a set, we propose a self-attention based mechanism for feature selection, presented in Algorithm 1 and illustrated in Figures 5 and 6. Intuitively, the process exploits the over-representation of patterns within features that are relevant to the classification as compared to patterns within the irrelevant features. We first standardise the features to prevent those common to the entire support set, which are not discriminating, from dominating the selection process (when classifying between tigers and lions we can ignore the claws, big teeth and whiskers). Numerical stabilisation has an additional benefit of preventing weakly activated features from being excessively scaled-up. We then repeatedly self-attend within the separate classes of the support set, using dot product attention with scale $\tau$, $X_k \leftarrow \text{DotAttn}(X_k, X_k, X_k, \tau)$. Self-attention over a set of vectors maps each vector to the interior of the convex hull of the original set. If every member of a class has some feature in common, the convex hull is a point in that dimension and the features do not change. In the polythetic case it is patterns of features that matter, and by attending more strongly between elements of the support set with such feature-patterns, these too are preserved. Figure 6, for example, shows how the $(+, -)$ and $(-, +)$ patterns in $(x, y)$ coordinates of the red class are preserved as vectors in these quadrants reinforce each other. We can measure the degree to which a feature has been preserved with the dispersion, and this is how we get a score for each feature which we can then use for feature selection.

The feature scores can be used to rescale features across the support and query sets before applying the classifier, as in Figure 8 or top-k selection. We focus on rescaling as the method that makes the fewest assumptions about the underlying classification, but using top-k is highly effective when the number of active elements is known, as shown in Appendix F.

Figure 6: Feature vectors converging under iterated self-attention on a XOR$_2$ classification of vectors uniformly sampled over the sphere, in 3D (top) and down the $z$-axis (bottom). Colours indicate classes. The vectors quickly align by $xy$-quadrant and the variation in $z$ is ‘washed-out,’ also be seen in the feature selection scores (mean-absolute-deviation) $\{x, y, z\}$. 

$r = 0$  
$[0.78, 0.80, 0.87]$  
$r = 1$  
$[0.81, 0.83, 0.67]$  
$r = 2$  
$[0.89, 0.88, 0.49]$  
$r = 3$  
$[0.99, 0.94, 0.31]$  
$r = 4$  
$[1.06, 0.97, 0.23]$
Algorithm 1: Self-attention feature scoring. Scores can be used for rescaling or masking. Note that the z-normalisation is over the entire support set whilst the self-attention is within classes. The choice of dispersion measure is of secondary importance and discussed in the main text.

**Input** : Support set $S = \{x_i, y_i\}_{i \in I_S}$ with class labels $y_i \in \{1, \ldots, K\}$ and features $x_i \in \mathbb{R}^F$, $S_k$ denoting the subset of $S$ containing all samples with $y_i = k$ and $X_k \in \mathbb{R}^{|S_k| \times F}$ the an arbitrarily ordered matrix of feature vectors belonging to $S_k$; small numerical constant, $\epsilon$; attention temperature, $\tau^{-1}$; repetitions, $R$.

**Output** : Feature scores, $f \in \mathbb{R}^F$.

1. $x_i \leftarrow (x_i - \mu_X) / (\sigma_X + \epsilon); \quad$ // standardise
2. repeat $R$ times
3. for $k \leftarrow 1$ to $K$ do
4. $X_k \leftarrow \text{softmax}(\tau X_k X_k^T) X_k; \quad$ // softmax is row-wise
5. $f \leftarrow \text{dispersion}\{x_i\}_{i \in S_i}; \quad$ // mean-absolute-deviation, standard deviation etc.

**An illustrative example of the method.** Figure 7 provides a global overview of the proposed method. In this example, we are interested in distinguishing between big-cats and equids (horses, donkeys, and in the case zebra). First, we extract features from all samples in the support and query sets. Here, we imagine these features as corresponding to some general 'cat' properties, patterns (such as stripes, dots etc.) and general 'equid' properties. Second, we perform repeated self-attention with respect to the separate classes of the support set (as presented in lines 2 : 4 in Algorithm 1), which yields updated features for each support sample. Next, we aggregate the resulting support features with an appropriate dispersion metric, e.g. mean absolute deviation or standard deviation, (line 5 in Algorithm 1) to obtain a vector of feature scores. These scores quantify the relevance of each feature in a given task. Then, we rescale both the query and (initial) support features, i.e. multiplying by the feature scores, to dilute the task-irrelevant and (potentially) misleading features. In this particular example, this would correspond to diluting the 'patterns' feature, since it is irrelevant to distinguish between cats and equids. Finally, we produce class probabilities via the attentional classifier.

![Diagram of the proposed approach](https://example.com/diagram.png)

Figure 7: Diagram of the proposed approach. Note how the misleading features of stripes and spots, which may cause misclassification, are diluted through the feature selection process.

4 Experiments

We compare the proposed method (FS) with Prototypical Networks (PN) [5], a threshold classifier, and Matching Networks (MN) [7], an attentional classifier without feature-selection, on several synthetic and real-world few-shot learning problems. Importantly, as our approach is non-parametric and operates directly on high-level features, it is agnostic to the choice of feature extractor. We are free to choose an appropriate feature extractor to match the input type (e.g. a convolutional neural network for images or a multi-layer perceptron for tabular data), and in all experiments we use the same embedding model for all methods (see experimental details in Appendix G).

*Our code is publicly available at: [https://github.com/rvinas/polythetic_metalearning](https://github.com/rvinas/polythetic_metalearning)*
Figure 8: The effect of soft feature selection on attention classification of XOR_1 over variant frequency, \( r \), and number of inactive components \( \beta \). Examples of a variant satisfy the XOR in the same way, i.e. the active components are equal, but may not have the same inactive components. Increasing \( r \) assists feature selection, in agreement with the derived misclassification distribution. Soft feature-selection rescales features according to their importance, as determined by the self-attention procedure. This greatly improves performance even at low repetitions, for example at 2 repetitions and 3 inactive components the change in accuracy is +38pp. Neither method is effective at high \( \beta \) with low \( r \).

Table 1: Binary strings. Accuracy by embedding dimension for sequences of length \( n = 5 \) and \( n = 10 \). Mean and standard error calculated over 1000 tasks.

| Model | Emb. | XOR_2 | XOR_3 | XOR_4 | XOR_2 | XOR_3 | XOR_4 |
|-------|------|-------|-------|-------|-------|-------|-------|
| PN [5] | 1    | 57.6 ± 0.5 | 55.3 ± 0.5 | 60.8 ± 0.7 | 51.7 ± 0.4 | 50.2 ± 0.2 | 50.1 ± 0.2 |
|       | \( n \) | 73.6 ± 0.5 | 70.3 ± 0.7 | 91.4 ± 0.4 | 56.7 ± 0.4 | 50.1 ± 0.2 | 50.4 ± 0.2 |
|       | \( 2n \) | 86.3 ± 0.4 | 75.3 ± 0.7 | 100.0 ± 0.0 | 57.7 ± 0.4 | 49.9 ± 0.2 | 50.3 ± 0.2 |
|       | \( n^2 \) | 90.4 ± 0.3 | 77.8 ± 0.7 | 100.0 ± 0.0 | 62.1 ± 0.4 | 50.3 ± 0.2 | 50.6 ± 0.2 |
| FS    | \( n \) | 99.6 ± 0.2 | 100.0 ± 0.0 | 100.0 ± 0.0 | 75.9 ± 1.3 | 82.6 ± 1.1 | 96.3 ± 0.5 |

**Binary strings.** We first consider the problem of meta-learning arbitrary Boolean functions of \( n = \alpha + \beta \) variables, with \( \alpha \) active and \( \beta \) task-irrelevant components. Labels for inputs \( \mathbf{x} \in \{-1, 1\}^n \) are generated by computing the XOR of a random subset of components of size \( \alpha \). Each of the \( 2^n \) variations occurs \( 5 \) times in the support set, \( |S| = 5 \cdot 2^{\alpha} \). The subset of active components is unknown to the meta-learner. Figure 8 highlights the benefit of feature-selection in this setting.

Table 1 summarises the performances of Prototypical Networks and our approach in the binary strings experiment. PN accuracy decreases sharply with sequence length \( n \). Moreover, the number of embedding units required to effectively solve this problem grows rapidly with \( n \), as also shown in Figure 4. This suggests that PN are indeed learning pseudo-variables for each pair of components and highlights the limitation of threshold classifiers for solving polythetic tasks.

**Polythetic MNIST.** We evaluate the ability of the models to jointly extract high-level features and identify polythetic patterns. We construct tasks using MNIST digits [12], where an example consists of 4 coloured digits (RGB). Figure 9 shows an example task. For monothetic tasks, a single high-level feature (e.g. colour of the top-right digit) distinguishes classes. For polythetic tasks, class membership derives from XOR interactions over features. The remaining digits are task-irrelevant.

Table 2 shows the performances on three versions of the polythetic MNIST dataset: clean (excluding non-discriminative digits), colourless (task-irrelevant digits but no colour), and full (both task-irrelevant digits and colour). The models are trained on monothetic tasks and evaluated both on monothetic and polythetic tasks. We note that protonets excel at identifying monothetic features and ignoring non-discriminative features, but have a close to random performance on polythetic tasks. Conversely, matching networks, which are polythetic classifiers by default, are highly sensitive to task-irrelevant features. The proposed approach can simultaneously detect salient features and perform polythetic classifications. Furthermore, as shown in Figure 10, our classifier exhibits robust performances when trained on varying rates of polythetic tasks in a second experiment.
we train a prototypical network for character recognition on the Omniglot training set and evaluate performance on 3-way to make polythetic classifications at test time using a feature draw from the other 20. As discussed, our approach can be used are formed using examples from 30 of the alphabets, test tasks characters from 50 writing systems with 20 examples of each Omniglot. The Omniglot dataset [3] consists of handwritten classifiers are weaker in the monothetic setting, but more than make up for this defect in the polythetic advantage in monothetic tasks but perform no better than chance for polythetic tasks. Attentional in Table 3, and conform to the trend we see in other experiments: threshold classification has the accuracy of 95% from the label combinations. In the multi-categorical pre-training, the model achieved a validation comparisons over 1600 validation examples. The pre-trained model was then used with prototypical and attentional classifiers in the polythetic MNIST few-shot classification setting discussed in Section 4 and detailed further in Appendix C. We compared performance using the multi-headed softmax activations the model was pre-trained with and an elementwise sigmoid for both the monothetic and polythetic settings. The results are presented in Table 4 and conform to the trend we see in other experiments: threshold classification has the advantage in monothetic tasks but perform no better than chance for polythetic tasks. Attentional classifiers are weaker in the monothetic setting, but more than make up for this defect in the polythetic setting.

Table 2: Polythetic MNIST. Evaluation accuracy on monothetic and polythetic tasks in three settings. Mean and standard error calculated over 1000 tasks.

| Model | Monothetic | Polythetic | Monothetic | Polythetic | Monothetic | Polythetic |
|-------|------------|------------|------------|------------|------------|------------|
| PN [5] | 97.9 ± 0.1 | 50.6 ± 0.3 | 92.8 ± 0.3 | 49.9 ± 0.3 | 94.5 ± 0.3 | 49.8 ± 0.2 |
| MN [7] | 79.6 ± 0.6 | 57.6 ± 0.4 | 69.7 ± 0.4 | 61.0 ± 0.5 | 70.1 ± 0.7 | 56.6 ± 0.5 |
| FS     | 96.8 ± 0.1 | 98.3 ± 0.0 | 94.5 ± 0.2 | 98.0 ± 0.0 | 75.0 ± 0.7 | 60.4 ± 0.7 |

Multi-categorical pre-training. In this experiment we first train a classifier in the multi-categorical setting for the full polythetic MNIST task. In this case there are no task-irrelevant digits or colours as the label describes all four digits with their colours: there are four 10-way labels for the digits (∈ ℝ^{4×10}) and four 3-way labels for the colours (∈ ℝ^{3}) for a combined multi-hot output vector ∈ ℝ^{52}. The model architectures match that used in the other experiments, see Appendix C other than adapting the MLP head that takes the flattened output from the convolutional network to fit the multi-categorical labels. The model is pre-trained over 800 batches of 16 examples drawn at random from the label combinations. In the multi-categorical pre-training, the model achieved a validation accuracy of 95.6% on the digit labels and 100% on the colour labels over 1600 validation examples.

Table 3: Polythetic MNIST with multi-categorical pre-training. Evaluation with softmax and sigmoid activations. Mean and standard error over 1000 tasks.

| Model | Multi-softmax | Sigmoid |
|-------|---------------|---------|
|       | Monothetic    | Polythetic | Monothetic | Polythetic |
| PN    | 97.72 ± 0.12 | 49.75 ± 0.27 | 94.55 ± 0.24 | 50.36 ± 0.27 |
| FS    | 93.90 ± 0.24 | 92.96 ± 0.28 | 88.54 ± 0.32 | 86.69 ± 0.38 |

Omniglot. The Omniglot dataset [3] consists of handwritten characters from 50 writing systems with 20 examples of each character drawn by different people. Tasks in the training set are formed using examples from 30 of the alphabets, test tasks draw from the other 20. As discussed, our approach can be used to make polythetic classifications at test time using a feature extractor trained with a threshold classifier (FS*). To test this, we train a prototypical network for character recognition on the Omniglot training set and evaluate performance on 3-way

Table 4: Omniglot. Mean and standard error over 1000 tasks.

| Model | Characters | Alphabets |
|-------|------------|-----------|
| PN    | 98.6 ± 0.0 | 83.4 ± 0.3 |
| FS    | 96.2 ± 0.0 | 94.2 ± 0.2 |
| FS*   | 98.1 ± 0.0 | 96.0 ± 0.2 |
alphabet recognition (inherently polythetic) with the test set. Table 4 shows the results of this experiment (1000 tasks). Applying our method with a pre-trained feature-extractor (FS*) reduces the accuracy of character recognition by 0.5pp yet improves alphabet recognition by 12.6pp. The end-to-end trained model (FS) is also competitive in character recognition and improves in alphabet recognition. Additionally, we note that regularising the embedding space increases disentanglement, with potential interpretability benefits (see Appendix H). We include other pre-training examples in Appendix H.

5 Related work

We consider the ability of meta-classifiers to generalise over unseen examples, and in particular, to adapt to polythetic tasks. Our work aligns with metric-learning approaches for few-shot classification [13] that apply distance functions between query and support examples in a common embedding space [5,7,14,15,16,17]. Specifically, Prototypical Networks [5,14] construct class-wise support prototypes in the embedding space and classify queries using nearest-neighbours by Euclidean distance. While such an approach can successfully tackle monothetic tasks, it is limited with respect to both the required embedding dimensionality and task-adaptiveness in a polythetic context. Allen et al. [16] introduced multi-modal mixture prototypes to allow prototype models to fit more complex data distributions, and can be seen as an alternative to attentional approaches.

On the other hand, attentional meta-classifiers such as Matching Networks (MN) [7] and Attentive Neural Processes (ANP) [18], can naturally adapt to polythetic tasks. MNs use an attention mechanism over distance functions between the embedded support and query examples. ANPs combine self-attention with cross-attention from the support set (which they call the ‘context’) to produce a task-context. Other approaches use an attention-based re-weighting of the query examples based on the support set [19], or cross-attention to focus on more informative regions of the input [20], promoting feature disentanglement in the constructed embedding. While all of these approaches introduce valid mechanisms for improving task-adaptability, they lack a mechanism to focus exclusively on relevant features, which is crucial for meta-learning polythetic classifications.

6 Conclusion

In this work, we articulated the differences between monothetic and polythetic classifiers in meta-learning scenarios. We identified non-threshold classifications with polythetic functions and showed that, to evaluate these functions, threshold classifiers require an embedding space that is exponential in the number of features. Moreover, we recognised attention classifiers as polythetic by default and analysed their limitations in the presence of task-irrelevant features. To address this problem, we introduced a novel self-attention mechanism for feature-selection that dilutes non-discriminative features. Finally, we illustrated the effectiveness of the proposed method in several synthetic and real-world few-shot learning tasks. This work has the potential to enhance our understanding of meta-learning systems, particularly in relation to generalisation. The key remaining challenges for future work are improving the end-to-end model to match the pre-trained performance and scaling the method.
Acknowledgments and Disclosure of Funding

We would like to thank Cristian Bodnar, Jacob Moss, and Shayan Iranipour for valuable feedback on this work.

References

[1] P.F. Stevens. Linnaeus, Carl von (1707-78). 1998. doi: 10.4324/9780415249126-Q059-1. URL https://www.rep.routledge.com/articles/biographical/linnaeus-carl-von-1707-78/v-1/sections/the-natural-method

[2] Birger Hjørland. Classification. Knowledge Organization, 44(2):97–128, 2017. URL http://www.isko.org/cyclo/classification

[3] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In Proceedings of the annual meeting of the cognitive science society, volume 33, 2011.

[4] Marvin Minsky and Seymour Papert. Perceptrons. 1969.

[5] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/cb8da676746f12812ae4290ae7bc42-Paper.pdf

[6] Ryan O’Donnell. Analysis of Boolean functions. Cambridge University Press, 2014.

[7] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833854983612fb05e3ec9148c-Paper.pdf

[8] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Lluís Márquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 1412–1421. The Association for Computational Linguistics, 2015. doi: 10.18653/v1/d15-1166. URL https://doi.org/10.18653/v1/d15-1166

[9] Ashish Vaswani, Noam Shazeer, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] J Willard Gibbs. Elementary principles in statistical mechanics. Charles Scribner’s Sons, 1902.

[11] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/f0e52b27a7a5d6a1a87373dffa53db5-Paper.pdf

[12] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2, 2010.

[13] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.
[14] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018.

[15] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1199–1208, 2018.

[16] Kelsey R. Allen, Evan Shelhamer, Hanul Shin, and Joshua B. Tenenbaum. Infinite mixture prototypes for few-shot learning. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 232–241. PMLR, 2019.

[17] Boris N. Oreskhin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 719–729, 2018.

[18] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=SkE6PjC9KX

[19] Zihang Jiang, Bingyi Kang, Kuangqi Zhou, and Jiashi Feng. Few-shot classification via adaptive attention. CoRR, abs/2008.02465, 2020.

[20] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 4005–4016, 2019.

[21] Nicolle Gruzling. Linear separability of the vertices of an n-dimensional hypercube. 2007. doi: 10.24124/2007/bpgub464.

[22] Anwar A Irmatov. On the number of threshold functions. Discrete Mathematics and Applications, 3(4):429–432, 1993.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. Iclr, 2(5):6, 2017.
A Threshold functions and Boolean functions

Table 5: Threshold functions, $t(n)$, and Boolean functions, $B(n) = 2^{2^\frac{n}{2}}$, of $n$ variables. The exact number of threshold functions is known up to $n = 9$ and an upper bound is $t(n) < 2^{n^2}$ [21][22]. Both grow quickly, but $t(n) \ll B(n)$ for large $n$. Learning all the non-threshold functions separately is not feasible.

| $n$ | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
|-----|----|----|----|----|----|----|----|----|----|----|
| $t(n)$ | 2  | 4  | 14 | 104| 94,572| $1.5 \times 10^7$ | $8.4 \times 10^9$ | $1.8 \times 10^{13}$ | $1.4 \times 10^{17}$ |
| $B(n)$ | 2  | 4  | 16 | 256| 65,536| $4.3 \times 10^9$ | $1.8 \times 10^{13}$ | $3.4 \times 10^{18}$ | $1.2 \times 10^{27}$ | $1.3 \times 10^{34}$ |

B Partitioned XOR performances

We hypothesise that, in order to solve polythetic tasks, prototypical networks need to create pseudo-variables in the embedding space (e.g. XOR for each pair of components for binary strings tasks). To test this, we train a prototypical network on binary strings tasks (where the active components are always chosen from the $s = 4$ first components) and evaluate the performance on unseen combinations of components (i.e. combinations of the last $n - s$ components). Table 6 show the accuracies for $s = 4$ and sequences of length $n = 10$. These results highlight the inability of prototypical networks to generalise to unseen combinations and support our hypothesis.

Table 6: Accuracy of prototypical networks on unseen non-threshold functions by embedding dimension. We use sequences of length $n = 10$ and $r = 5$ repetitions for each XOR variation. The number of inactive components for XOR, is $n - \alpha$. At train time, the active components are always within the $s = 4$ first components. At test time, the active components are randomly chosen from the last $n - s$ components. In other words, the model never observes test combinations at train time. The inability to generalise in this scenario suggests that protonets need pseudo-variables (i.e. XOR functions applied to each pair of components) to solve the binary strings task.

| Emb. $\times$ XOR | XOR2 | XOR3 | XOR4 |
|------------------|------|------|------|
| 1                | 50.180 ± 0.365 | 50.100 ± 0.223 | 50.063 ± 0.163 |
| n                | 50.586 ± 0.376 | 50.135 ± 0.229 | 49.767 ± 0.161 |
| $2n$             | 49.990 ± 0.367 | 50.228 ± 0.233 | 50.133 ± 0.158 |
| $n^2$            | 51.041 ± 0.374 | 49.980 ± 0.233 | 49.632 ± 0.161 |

C Temperature in attention classification

The softmax in attention mechanisms permits a temperature scaling that interpolates between argmax and uniform (and argmin.) This is controlled by $T = \frac{1}{\beta}$, as

$$\text{softmax}_i(x, \beta) = \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)}$$

with softmax converging to argmax as $\beta \to \infty$, and to uniform (i.e. all elements equal to the reciprocal of the length of the vector) as $\beta \to 0$.

For attention classifiers, a decrease in temperature increases the model confidence and can cause decision boundaries to move. As $\beta \to \infty$ and the softmax converges to argmax, the classifier tends to the single nearest neighbour classification scheme; for $\beta = 0$ the classifier returns the support set class balance. For Boolean functions, changes in temperature effect the degree to which decision boundaries are axis-aligned. For example, Figure 3 shows the decision boundary for $f(x, y) = \text{AND}(x, y)$ using
a softmax temperature of $1$ at $y = -\frac{1}{2} \ln (\tanh x)$, which we derive as
\[
p(\text{class 1}) = p(\text{class 0})
\]
\[
\frac{\exp (\beta (x + y))}{\sum \exp (...) - \exp (-\beta (x + y))} = \frac{\exp (\beta (-x - y)) + \exp (\beta (x - y))}{\sum \exp (...) - \exp (-\beta (x - y))}
\]
\[
sinh(\beta x) (\cosh(\beta y) + \sinh(\beta y)) = \cosh(\beta x) (\cosh(\beta y) - \sinh(\beta y))
\]
\[
tanh(\beta x) = \frac{\cosh(\beta y) + \sinh(\beta y)}{\cosh(\beta y) - \sinh(\beta y)}
\]
\[
tanh(\beta x) = \exp(-2\beta y)
\]
\[
y = -\frac{1}{2\beta} \ln (\tanh (\beta x)).
\]

The effect of decreasing the temperature on the decision boundary is shown in Figure 11.

D **No Boolean function is less well approximated by a linear classifier than XOR.**

**General case.** In the general case we have a $n$-variable Boolean function $B(x)$ defined over $x \in \{-1, 1\}^n$. The set of parity functions over $n$-bits (each function taking parity over a distinct combination of bits, for a total of $2^n$ functions) form a linearly independent basis for the Boolean functions, i.e.

\[
B(x) = \sum_{S \subseteq [n]} b_S \prod_{i \in S} x_i. \tag{9}
\]

Using these coefficients we form a polynomial threshold function that faithfully evaluates $B(x)$ as

\[
f(x) = \text{sign} \left( \sum_{S \subseteq [n]} b_S \prod_{i \in S} x_i \right) \in \{-, +\}. \tag{10}
\]

Our linear threshold is of the form

\[
h(x) = \text{sign} \left( a_0 + \sum_{i \in [n]} a_i x_i \right) = \text{sign} (a \cdot x + a_0) \in \{-, +\} \tag{11}
\]

which we can simplify by setting $x' = x|_1$ and $a' = a|_{a_0}$ to get $h(x) = \text{sign} (a'x')$. Errors occur when the sign of $f(x)$ and $h(x)$ oppose, which we can count with the negative reals indicator.
function $\mathbb{I}_{\mathbb{R}}$ as

$$\text{errors} = \sum_{x \in \{-1, 1\}^n} \mathbb{I}_{\mathbb{R}} \left( \text{sign} \left( f(x) \right) \text{sign} \left( h(x) \right) \right) \tag{12}$$

$$= \sum_{x \in \{-1, 1\}^n} \mathbb{I}_{\mathbb{R}} \left( f(x) \left( a' \cdot x' \right) \right). \tag{13}$$

Factoring out the $a'$ and 'vectorising' we have

$$\text{errors} = \sum_{\text{rows}} \mathbb{I}_{\mathbb{R}} \left( \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \\ \vdots \\ f(x_{2^n}) \end{bmatrix} \otimes \begin{bmatrix} - & - & \ldots & - & + \\ + & - & \ldots & + & \vdots \\ - & + & \ldots & - & \vdots \\ + & + & \ldots & + & + \end{bmatrix} \right) \cdot a' \tag{14}$$

where $\{-, +\}$ are used in place of $\{-1, +1\}$ and $\otimes$ indicates the Hadamard product i.e. the $+/-$ row-vectors are multiplied by the row-scalars $[f(x_i)]_{i \in \text{rows}} = \lambda \in \{-1, 1\}^{2^n}$. The problem can now be reframed as finding a non-zero vector $a'$ that gives the fewest negative values when dotted with the Hadamard product.

The row-vectors point at $n$ corners of the $(n-1)$-cube and there are no equal vectors nor collinear subsets for any value of $\lambda$, so the set can always be projected to the corners of an $n$-cuboid without overlap. For example, if we have the 2-variable function $B(x) = \text{AND}(x_1, x_2)$ then we get

$$\text{errors} = \sum_{\text{rows}} \mathbb{I}_{\mathbb{R}} \left( \begin{bmatrix} - \\ - \\ + \\ + \end{bmatrix} \otimes \begin{bmatrix} - & - & + \\ + & - & - \\ - & + & + \\ + & + & + \end{bmatrix} \right) \cdot a' = \sum_{\text{rows}} \mathbb{I}_{\mathbb{R}} \left( \begin{bmatrix} + & + & - \\ - & + & - \\ + & - & - \end{bmatrix} \cdot a' \right) \tag{15}$$

which project onto the plane with normal $(1, 1, 1)$ to form a rectangle (a 2-cuboid.) A procedure for finding the projection is to compare the midpoints of pairs of vectors: the vectors between the midpoints define projections that produce $n$-cuboids. In our particular case we can pair up the vectors by those that are parallel in the first $n$ elements to find midpoints (e.g. the first and last have $\{-\}^n$ and $\{+\}^n$, respectively).

Given we can do this for any value of $\lambda$, there exists a solution at least as good as that for the $n$-cuboid for any $B(x)$. How good is that solution? This depends on $n$ and whether $n$ is odd or even. Coming towards the cuboid 'corner-first', the corners are aligned in planes with the number of corners given by the binomial coefficients (e.g. for a rectangle these go $1-2-1$.) There are $n+1$ binomial coefficients, $+1$ for $\binom{n}{0}$, and likewise there are an odd number of planes for even dimensional cuboids. We can select $a'$ so as to align with these planes and get

$$\text{corner aligned} = \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} \tag{16}$$

correctly classified i.e. the sum of the binomial coefficients up to halfway, which for even $n$ includes the middle value.

Alternatively, we can come at the cuboid 'edge-first' and the corners are aligned in planes with the number of corners given by $2 \times$ the binomial coefficients (e.g. for a cube these go $2-4-2$.) In odd dimensions this is advantageous as we can make use of getting-the-middle-value

$$\text{edge aligned} = 2 \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} \binom{n-1}{i} \tag{17}$$

We cannot, however, come at the cuboid 'face-first' as the corners would no longer be aligned in planes (e.g. coming face-first at a 4-cube sees the corners 4-8-4 with the middle 8 being a cube.) The corner-aligned and edge-aligned approaches (Equations 16, 17) can be expressed together as

$$\text{cuboid approach} = 2^{n-1} + \binom{n-1}{\lfloor n-1/2 \rfloor} \tag{18}.$$
XOR  It is helpful to conceive of XOR over $n$ bits, XOR$_n$, as a binary colouring of the corners of an $n$-cube with the rule that no two neighbours (corners connected by an edge) have the same colour (we use red and blue in plots and refer to the groups by these colours). From the bit perspective, the corners differ in a single bit and so have opposite parity and label. This colouring means that axis aligned thresholds, parallel to a face of the cube, will always meet points in balanced groups (equal numbers of red and blue) and so have accuracy of one half. However, the symmetry of the cube permits a better approach. Planes of alternating colour align with normal vectors $v \in \{-1, +1\}^n$: the corner in the direction of the normal is one colour, its neighbours are displaced with rotational symmetry so as to form a plane and are of the other colour, their neighbours are displaced to form a third plane and are of the original colour, and so on. Figure 12 shows the 3-variable case.

The number of corners in a given plane is $\binom{n}{d}$ where $d$ is the Hamming distance, e.g. immediate neighbours have distance 1, and so there are $\binom{n}{1} = n$ of them and they have the opposing colour because their Hamming distance is odd. To select a linear threshold, we can imagine moving through these planes corner by corner. Clearly, if it is to our advantage to move the threshold beyond a given corner in a plane, it is to our advantage to move the threshold beyond them all, as they are of the same colour. The remaining possibilities are then between each plane, which we enumerate as $k \in \{0, n+1\}$ with $k = 0$ and $k = n + 1$ falling outside the cube. The (un-normalised) accuracy of a given threshold is the larger of (red-below + blue-above) and (red-above + blue-below). Without loss of generality, assume we start at a red corner, then we have

\[
R_k = \sum_{d=0}^{\lfloor k/2 \rfloor} \binom{n}{2d} \left(2(n-d-1)\right) \tag{19}
\]

\[
B_k = \sum_{d=0}^{\lfloor k/2 \rfloor} \binom{n}{2d-1} \tag{20}
\]
noting that \( \sum_{i=0}^{n} 0 = 0 \) and \( (-1)^{n} = 0 \). It follows that \( R_{\uparrow} = 2^{n-1} - R_{\downarrow}, B_{\uparrow} = 2^{n-1} - B_{\downarrow} \). Combining to get accuracies, we have

\[
R_{\uparrow} + B_{\downarrow} = \sum_{d=1}^{\lfloor \frac{n+1}{2} \rfloor} \binom{n}{2(d-1)} + 2^{n-1} - \sum_{d=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2d-1} \tag{21}
\]

\[
= 2^{n-1} + (-1)^k \binom{n-1}{k} \tag{22}
\]

and similarly

\[
R_{\downarrow} + B_{\uparrow} = 2^{n-1} - \sum_{d=1}^{\lfloor \frac{n+1}{2} \rfloor} \binom{n}{2(d-1)} + \sum_{d=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2d-1} \tag{23}
\]

\[
= 2^{n-1} + (-1)^{k+1} \binom{n-1}{k} \tag{24}
\]

The larger of these being

\[
\text{accuracy} = 2^{n-1} + \binom{n-1}{k} \tag{25}
\]

which is maximised for \( k = \lfloor \frac{n-1}{2} \rfloor \). The best accuracy we can achieve for XOR \( _n \) is therefore

\[
\max \text{ XOR accuracy} = 2^{n-1} + \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}. \tag{26}
\]

This is equal to the minimum achievable with the cuboid approach to any \( B(x) \). Finally, there are thresholds of this kind with normal vectors parallel to every corner. Between these directions are ‘edge’- and ‘face-aligned’ planes, for which the planes consist of equal numbers of red/blue corners and any threshold gives an accuracy of one half. Intermediate directions give some intermediate accuracy, so the maximum accuracy is that for the ‘corner-aligned’ approach, and so there are no Boolean functions less well approximated by a linear threshold than XOR. The first few values are \( \{3, 6, 11, 22\} \) which correspond to normalised accuracies of \( \{75\%, 75\%, 68.75\%, 68.75\%\} \), and for large \( n \) the accuracy slowly tends to one half.

### E Misclassification distribution full derivation

We first present a more detailed derivation for the case of dot-product attention. We will make repeated use of the Binomial theorem

\[
(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}. \tag{27}
\]

Firstly, to Equation[4] The class is determined by the sign of the sum of contributions at an even distance subtract those at an odd distance. Putting those into a single sum we have

\[
\text{class} = \text{sign} \left( \sum_{\delta=0}^{\alpha} r (-1)^{\delta} \binom{\alpha}{\delta} \exp(\alpha - 2\delta) \right). \tag{28}
\]

We can factor out the number of repetitions, \( r \), and as it is positive it doesn’t change the sign. We can then rearrange to match the Binomial theorem form and recover the form in the main text:

\[
\text{class} = \text{sign} \left( \sum_{\delta=0}^{\alpha} \binom{\alpha}{\delta} e^{\alpha - \delta} \left( \frac{-1}{e} \right)^{\delta} \right) = \text{sign} \left( (e - e^{-1})^\alpha \right) = +. \tag{29}
\]

Next we give the Binomial distribution of the irrelevant feature contribution as \( 2B(\beta, \bar{p}) - \beta \) with \( \bar{p} = p^2 - (1 - p)^2 = 2p^2 - 2p + 1 \). Something that is important to the behaviour of the mean and variance later, and breaks the usual symmetry of \( p \) and \( q = (1 - p) \), is that \( \bar{p} \geq 0.5 \) and is quadratic in
\( p \) defined by the three points \((p, \bar{p}) = \{(0, 1), (0.5, 0.5), (1, 1)\}. \( q \) has the usual definition \(1 - \bar{p} \), so \( \bar{q} \leq 0.5 \) and so on. The contribution at a difference \( \delta \) is then \( X(\delta) \sim \exp (\alpha - 2\delta + 2B(\beta, \bar{p}) - \beta) \).

The expectation is computed in the usual way
\[
E[X(\delta)] = \sum_i P(X = x_i)x_i = \sum_{b=0}^{\beta} P(B(\beta, \bar{p}) = b) \exp (\alpha - 2\delta + 2b - \beta)
\]
\[
= \exp (\alpha - 2\delta) \sum_{b=0}^{\beta} \left( \frac{\beta}{b} \right) \bar{p}^b \bar{q}^{(\beta - b)} \exp (2b - \beta)
\]
\[
= \exp (\alpha - 2\delta) \sum_{b=0}^{\beta} \left( \frac{\beta}{b} \right) \bar{p}^b \bar{q}^{(\beta - b)} = \exp (\alpha - 2\delta) \left( \bar{p}e + \bar{q}e^{-1} \right)^\beta.
\]

Finding the variance in the traditional way, \( \text{Var}[X(\delta)] = E[X(\delta)^2] - E[X(\delta)]^2 \), first \( E[X(\delta)^2] \) following the derivation for \( E[X(\delta)] \):
\[
E[X(\delta)^2] = \sum_i P(X = x_i)x_i^2 = \sum_{b=0}^{\beta} P(B(\beta, \bar{p}) = b) \exp (2\alpha - 4\delta + 4b - 2\beta)
\]
\[
= \exp (2\alpha - 4\delta) \sum_{b=0}^{\beta} \left( \frac{\beta}{b} \right) \bar{p}^b \bar{q}^{(\beta - b)} \exp (4b - 2\beta)
\]
\[
= \exp (2\alpha - 4\delta) \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^\beta.
\]

From this we can write the variance
\[
\text{Var}[X(\delta)] = \exp (2\alpha - 4\delta) \left( \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^\beta - \left( \bar{p}e + \bar{q}e^{-1} \right)^{2\beta} \right).
\]

Next we want to find the expectation of the sum of the contributions at each difference \( \delta \). As pointed out in the lead up to Equation 4, there are \( \binom{\alpha}{\delta} \) many strings at a difference of \( \delta \). We apply \( E[\sum_i X_i] = \sum_i E[X_i] \) and \( \text{Var}[X - Y] = \text{Var}[X] + \text{Var}[Y] \), and, remembering to change the sign with each increase in \( \delta \),

\[
E \left[ \sum_{i \in S} (-1)^\delta X(\delta_i) \right] = \sum_{\delta=0}^{\alpha} r(-1)^\delta \binom{\alpha}{\delta} E[X(\delta)]
\]
\[
= r \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^\beta \sum_{\delta=0}^{\alpha} \binom{\alpha}{\delta} e^{(\alpha - \delta)} \left( \frac{-1}{e} \right)^\delta
\]
\[
= r \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^\beta \left( e - e^{-1} \right)^\alpha,
\]

and

\[
\text{Var} \left[ \sum_{i \in S} (-1)^\delta X(\delta_i) \right] = \text{Var} \left[ \sum_{i \in S} X(\delta_i) \right] = \sum_{\delta=0}^{\alpha} r \binom{\alpha}{\delta} \text{Var}[X(\delta)]
\]
\[
= r \left( \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^\beta - \left( \bar{p}e + \bar{q}e^{-1} \right)^{2\beta} \right) \sum_{\delta=0}^{\alpha} \binom{\alpha}{\delta} \left( e^2 \right)^{\alpha - \delta} \left( e^{-2} \right)^\delta
\]
\[
= r \left( \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^\beta - \left( \bar{p}e + \bar{q}e^{-1} \right)^{2\beta} \right) \left( e^2 + e^{-2} \right)^\alpha.
\]

E.1 Different attention mechanisms

For dot-product and cosine-similarity attention, ‘angular’ difference mechanisms, we use \((+, -)\) to encode the input variables. This is because we want the score to be greater when the variables are
the same and lesser when they are opposed (if we were to use \((1, 0)\) we’d have \(0 \times 0 \neq 1 \times 1\) and \(0 \times 1 = 0 \times 0\)). With these methods we get

\[
\begin{align*}
   f_{\text{dot}}(\delta) &= \alpha - 2\delta, \\
   f_{\text{cos}}(\delta) &= 1 - \frac{2\delta}{\alpha}.
\end{align*}
\]

The change for cosine-similarity introduces a factor of \(\exp(1/\alpha)\) to the mean and \(\exp(2/\alpha)\) to the variance, and the overall picture doesn’t change.

\[
\begin{align*}
   \mathbb{E} \left[ \sum_{i \in S} (-1)^{\delta_i} X_{\text{cos}}(\delta_i) \right] &= r \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^{\beta} \left( e^{1/\alpha} - e^{-1/\alpha} \right)^{\alpha}, \\
   \text{Var} \left[ \sum_{i \in S} (-1)^{\delta_i} X_{\text{cos}}(\delta_i) \right] &= r \left( \left( \bar{p}e^2 + \bar{q}e^{-2} \right)^{\beta} - \left( \bar{p}e + \bar{q}e^{-1} \right)^{2\beta} \right) \left( e^{2/\alpha} + e^{-2/\alpha} \right)^{\alpha}.
\end{align*}
\]

For squared Euclidean distance attention (which coincides with the ‘Laplace attention’ L1 norm in this encoding), if we encode the inputs as \((1, 0)\) we get \(f_{L^2}(\delta) = -(\sqrt{\delta})^2 = -\delta\), introducing a factor of \(\exp(-\alpha)\) to the mean and \(\exp(-2\alpha)\) to the variance, which also doesn’t change the overall picture.
F  Top-k feature selection

Top-k feature selection masks out all but the \( k \) highest scoring features. That is \( x \leftarrow m \odot x \) with

\[
m_i = \begin{cases} 
1 & \text{if } \text{rank}(s)_i \leq k \\
0 & \text{otherwise}
\end{cases}
\]  (44)

As compared to soft feature selection which rescales as \( x \leftarrow s \odot x \), or \( x \leftarrow s' \odot x \) with normalised \( s' \).

![Diagram showing comparison of direct feature selection vs. top-k feature selection](image)

Figure 13: Comparing soft and top-k feature selection in the same setting as Figure 8, classification of XOR over variant frequency, \( r \), and number of inactive components \( \beta \). Examples of a variant satisfy the XOR in the same way, i.e. the active components are equal, but may not have the same inactive components. The top-k version uses a binary mask to leave the \( k \) highest scoring features unchanged and zeroing the rest. This produces significant improvements over even the soft feature selection method at high values of \( \beta \), but requires knowledge of the number of active elements.
G Experimental details

We use the same feature extractor architecture and train loop for all the baselines.

**Feature extractor.** We leverage a convolutional neural network with 4 blocks as a feature extractor. Each block consists of a convolutional layer (64 output channels and $3 \times 3$ filters), followed by batch normalisation (momentum 0.01), a ReLU activation, and $2 \times 2$ max pooling:

$$\text{Conv2d}(64, 3 \times 3) \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{MaxPool}(2 \times 2)$$

Then, we flatten the output and apply a linear layer to map the data into a 64-dimensional embedding space (unless otherwise stated). As explained in the main manuscript, each method then manipulates these embeddings in different way.

**Train loop.** We train all models in an episodic manner. At each training iteration, we follow these steps:

- First, we sample task-specific support and query sets. For polythetic MNIST, the support set has 96 samples (2 classes, 2 groups per class, and 24 group-specific examples per group). The query set consists of 32 samples (2 classes, 2 groups per class, and 8 group-specific examples per group). For Omniglot, the support set consists of 5 examples for 20 classes (20-way, 5 shot). The query set consists of 15 examples per class.
- Second, we compute embeddings using the feature extractor and produce class probabilities for the query points. The way in which these probabilities are computed depends on the method (e.g. attentional classification for matching networks or softmax over prototype distances for prototypical networks).
- Finally, we compute the cross entropy for the query examples and optimise the feature extractor via gradient descent. We employ an Adam optimiser with learning rate 0.001.

We train the models for 10,000 iterations (i.e. tasks) for all experiments, except for full polythetic MNIST (100,000 tasks). We then compute the performances on a held-out dataset and average the results across 1,000 tasks.

**Multi-categorical model.** The MLP head variation used for the multi-categorical task has two hidden layers with 512 units each and ReLU activations, before linear layers with softmaxes for each of the label heads with sizes $\{10, 10, 10, 4, 4, 4\}$. 

\[21\]
H Regularisation of the embedding space

We conjecture that regularising the embedding space by encouraging better disentanglement might have potential benefits in terms of interpretability and (potentially) generalisation performance. To study this, we leverage a probabilistic feature extractor \( q(z|x) \) that produces embeddings \( z \) for a given example \( x \) as follows:

\[
z \sim q(z|x), \quad q(z|x) = \mathcal{N}(\mu, \text{diag}(\sigma^2)), \quad \mu = f_\mu(h), \quad \sigma^2 = \exp \left( \frac{1}{2} f_\sigma(h) \right), \quad h = f(x)
\]

where \( f(\cdot) \) is a convolutional neural network (as described in Appendix G), and \( f_\mu(\cdot) \) and \( f_\sigma(\cdot) \) are linear layers that produce the parameters of \( q(z|x) \). We leverage the reparameterisation trick to sample the embeddings from \( q(z|x) \). Similar to \( \beta \)-VAE, we minimise the following loss function:

\[
\mathcal{L} = \text{cross entropy} - \beta D_{KL}(q(z|x)||\mathcal{N}(0, I))
\]

where \( \beta \) is a hyperparameter and \( D_{KL} \) is the Kullback-Leibler divergence between the latent distribution \( q(z|x) \) and an isotropic normal prior\(^2\). Essentially, the rationale behind such approach is to prioritise solutions where each embedding unit follows an independent normal distribution.

Our proof-of-principle study points out that such a regularisation mechanism leads to a more disentangled embedding space. For instance, Figures 14, 15, and 16 show the marginal distributions of each embedding unit for three groups of samples in a certain monothetic MNIST task. The distinctive feature of this task is the digit colour in the top-right corner (all the other attributes are not discriminative). In this scenario, we note that component 10 exhibits a smaller standard deviation and that the means are different for each group, which suggests that it is capturing the information of top-right digit’s colour.

![Figure 14: Embedding distributions for support set (first group).](image)
Figure 15: Embedding distributions for support set (second group).

Figure 16: Embedding distributions for support set (third group).