1 VOUW: A Framework

1.1 Encoding Models and Instantiations

Even though we will not actually be encoding A, we want the calculation of the encoded length to be as precise as possible. The simple reason for this is that the only information that we provide to the MDL equation and thus the optimization process, is this encoded length. A coarse granularity may lead to unwanted results and overfitting. While this may sound inexact we will make it more precise by using Shannon's Entropy to reason about the optimal length for any element we encode.

From the definitions of $\mathcal{H}_A$ and $\bar{\mathcal{H}}_A$ we know that patterns both occur as elements in H_A as well as in the regions of $A|H_A$. It makes sense that we encode the complete patterns once for the models and then refer to them from each region by a code word. This makes the practical materialisation of the model a *code table* that maps each pattern to some code word. So how do we generate these code words? Recall that we are only interested in the theoretical length of a code and not so much in the actual word itself. To compute the optimal length we first and foremost need to find out what the chance is that a certain pattern occurs and this in turn depends on the prevalence in the dataset.

Definition 1. *Given a set of instantiations $A|H_A$, we take* $\texttt{usage}(X)$ *of a pattern X to be the prevalence of X^* in $A|H_A$. More precisely*

$$\texttt{usage}(X) = |\{R \mid R = (Y, t, \boldsymbol{p}) \in \{A|H_A\}, Y^* = X^*\}|$$

From this definition we see that the *usage* of a pattern is sum a of how often any of its variants occur as an instance. Using this function we find the probability that a certain pattern occurs simply by $P(X) = \frac{\texttt{usage}(X)}{|\{A|H_A\}|}$. The optimal length of a code word L^C can then be found by Shannon's Entropy. Now that we know the length of a code word the pattern itself must also be encoded. Recall that a pattern consists of pairs of spatial and magnitude offsets. The number of bits required for the spatial offsets depend directly on the *area* the pattern covers in A. We define this area informally as the difference in rows and columns between the smallest and the largest offset in a pattern X. So this is not necessarily the actual number of elements in X, as there could be gaps or X could have some irregular shape. Furthermore, instead of computing the area, we compute the *width* and *height* of a pattern separately. These are defined in two steps: first we define $\texttt{rowMax}(X) = i \iff \big((i,j), \boldsymbol{w}\big) \in X \wedge \nexists \big((i',j'), \boldsymbol{w'}\big) \in X$ s.t. $i' > i$, and analogously $\texttt{rowMin}(X)$, $\texttt{colMax}(X)$ and $\texttt{colMin}(X)$ as the largest and smallest row and column occurring in an offset of X respectively. We can then simple say that $\texttt{width}(X) = \texttt{colMax}(X) - \texttt{colMin}(X)$ and define $\texttt{height}(X)$ analogously for the row offsets.

We define the height and width of a pattern because we would like to encode the row and column offsets individually, while we could also have just taken the offset within the total area of X. The latter case would have resulting in a slightly smaller code length and would have given equal code length for the same area *regardless of shape*. Consider two patterns X and Y with equal areas 2×8 and 4×4 respectively. However, a single spatial offset in X has a length of $\log(2) + \log(8) = 4$ while for Y this is $\log(4) + \log(4) = 4$.

In a $M \times N$ matrix there can be at most $4(M-1)(N-1)$ distinctive spatial offsets and this number can be used to obtain the minimum number of bits each offset can be encoded with. The magnitude offset depends on the number of possible values in A, which is $b(A)$. Assuming the distribution of values in A is uniform, we can also use a fixed value here. This brings the length of an encoded pattern to:

$$L(X) = |X| \cdot \left(-\log\left(\big(4(M-1)(N-1)\big)^{-1}\right) - \log\left(b(A)^{-1}\right)\right)$$

This leaves only the instantiations still to be encoded, which form a list rather than a table. Each region simply refers to its pattern X by using the code word we computed earlier and we already know its length. The pivot is again a fixed number that depends on the total number of possible pivots - MN in this case. Encoding the variant is harder because we have never clearly defined $|X^*|$. In principle the upper bound for the number of variants for a given pattern X is $b(A)^{|X|}$, since we established X^* is at least finite. This is not a practical figure however, given that there probably are far less elements in A. A solution is to limit the total number of variants for X to the number that *we know about* at a given moment. We can find this number in a similar way to the `usage` function.

Definition 2. *Given a set of instantiations $A|H_A$, we define*

$$\texttt{variants}(X) = |\{Y \circ t \mid R = (Y, t, \boldsymbol{p}) \in \{A|H_A\}, Y^* = X^*\}|$$

Which basically defines $\texttt{variants}(X)$ as the number of distinct variants of X that occur within $A|H_A$. In a similar way as the encoded length of a pattern, we can now define the encoded length of region.

$$L(R) = -log\big(MN^{-1}\big) - \log\left(\texttt{variants}(X)^{-1}\right) + L^C(X)$$

Where X is the pattern in R. We can now compute the lengths of both region and patterns as well as the length of a code word for each pattern. By summing of all patterns and regions we can finally give the total length of the model and the set of instantiations.

$$L(H_A) = \sum_{X \in H_A} L^C(X) + L(X)$$
$$L(A|H_A) = \sum_{R \in A|H_A} L(R)$$