Supplementary information: fast and accurate m6A calling using single-molecule long read sequencing

1 Supplementary Methods

1.1 Fibertools algorithm

We derive m6A labels from GMM-filtered ipdRatios from ipdSummary. These labels can contain false positives. The lack of clean m6A labels makes the supervised training approach less suitable since it assumes that accurate labels are available. Therefore, we present Fibertools, a semi-supervised approach that makes a more reasonable assumption that our m6A class is a mixed population of true and false positives and that our non-m6A class is a clean set. Our training approach derives from Percolator and mokapot, proteomics tools for identifying peptides from mass spectrometry data [1, 2].

Given a set of candidate m6A calls, Fibertools aims to maximize the number of m6A calls at a target estimated precision. First, we split our dataset into training and validation sets stratified by class labels (see Table S3 for the training and validation datasets sizes for different Fiber-seq chemistry experiments). Then our method proceeds in two phases. In the first phase, we use the central base's inter-pulse distance (IPD) score as a classifier and generate an m6A classification score for all examples in the validation set. The validation examples are then ranked by the classification score, and estimated precision is computed at every score threshold. In general, precision is defined as the ratio of true positives over the total number of positive calls (i.e., true positives plus false positives). In our case, for a set of m6A (P) and non-m6A calls (N) meeting a score threshold, the denominator of the precision calculation is simply P + N. For the numerator, i.e., the number of true positives, we assume that some of the P m6A calls are false positives. In particular, we assume that each non-m6A call above the threshold corresponds to c false positive m6A calls, where c is the ratio of positive to negative labels in our validation dataset. Thus, we define estimated precision, *EP*, as

$$EP = \frac{P - cN}{P + N} \tag{1}$$

Using a ranked list of EP at different score thresholds, we select the score threshold with the target EP in the validation set. At the end of this phase, we have a score threshold for identifying m6A calls at the target estimated precision. In practice, 12-14% of our data had m6A labels, and we used the target estimated precision of 95%.

The second phase is iterative, and each iteration consists of three steps. The first step is selecting a high-confidence m6A training set using the score threshold from the first phase for the first iteration and step 3 of the second phase for subsequent iterations. The second step consists of training a CNN model on this training data. In the final step, the validation data is re-scored using the trained CNN model from the second step, and a new score threshold is generated with the re-scored validation data. We will now describe each step in detail.

In the first step, a score threshold is used to select a putative m6A set. This score threshold is derived from the IPD score classifier in the first iteration and a trained CNN in subsequent iterations. At the end of this step, we have a training set selected using a given score threshold and classifier.

The second step is training a CNN model to discriminate between m6A and non-m6A examples. We initialize our model using the CNN model from the supervised approach. This transfer learning approach allows fast convergence of our training procedure. We retain the same training hyper-parameters as the supervised approach. At the end of this step, we produce a CNN classifier trained on training data from

the previous step. We train the 2.2, 3,2 and Revio chemistry models for two epochs. During an epoch, we compute average precision on the validation data every x iteration, where x refers to the number of batches corresponding to 10% of training data. We save the model with the best average precision on validation data. Training for two epochs was sufficient to learn a model with better average precision than the previous round for all three fiber-seq chemistries.

The final step is finding a new score threshold by re-scoring the validation examples using the trained CNN from step two. The initial process of estimating EP at different score thresholds is repeated, and a new score threshold with a target EP (95%) is selected.

In the case of a successful second phase training, the number of positives in the validation data identified at target precision increases with every iteration and plateaus when most m6A examples in the validation data have been identified. We define two conditions for convergence, both of which must be satisfied. First, more than 70% of putative m6A calls from the validation set have been identified. Second, the number of additional m6A calls in a new iteration is less than 1% of the total putative m6A calls. In practice, it took 12, 11 and 3 repetitions of phase two training to converge 2.2, 3.2 and Revio chemistry Fiber-seq experiments, respectively (Fig. S9).

Algorithm 1 The SemiM6ACalling Algorithm The input variables are: $\mathcal{TD}$ = training data, $\mathcal{VD}$ = validation data, c = ratio of positive and negative m6A calls in $\mathcal{TD}$ and $\mathcal{VD}$, t = target precision threshold, *IPDClassifier* = m6A classifier using the IPD score of the central base, $\mathcal{I}$ = the number of iterations.

procedure SemiM6ACalling($\mathcal{TD}$, $\mathcal{VD}$, c, t, *IPDClassifier*, $\mathcal{I}$)
  $s_t \leftarrow$ scoreAtTargetPrecision(*IPDClassifier*, $\mathcal{VD}, t, c$) ▷ Compute score threshold s_t at precision t
  for $i \leftarrow 1 \ldots \mathcal{I}$ **do**
    $\mathcal{TD}_t \leftarrow$ selectTrainingSet($\mathcal{TD}, s_t$) ▷ Compute new training set $\mathcal{TD}_t$
    $\mathcal{CNN}_w \leftarrow$ trainCNN($\mathcal{TD}_t$) ▷ Train the CNN classifier $\mathcal{CNN}_w$
    $s_t \leftarrow$ scoreAtTargetPrecision($\mathcal{CNN}_w, \mathcal{VD}, t, c$) ▷ Recompute s_t
  end for
  return $(s_t, \mathcal{CNN}_w)$
end procedure

References

[1] William E Fondrie and William S Noble. mokapot: fast and flexible semisupervised learning for peptide detection. *Journal of Proteome Research*, 20(4):1966–1971, 2021.

[2] Lukas Käll, Jesse D Canterbury, Jason Weston, William Stafford Noble, and Michael J MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature methods*, 4(11):923–925, 2007.