# Specificity Prediction of Plant Acyl-ACP Thioesterases using Machine Learning

Deepro Banerjee\*, Michael Jindra†, Soodabeh Ghaffari\*, Veda Sheersh Boorla\*, Brian F. Pfleger†, Costas D. Maranas\*

\* Department of Chemical Engineering, Pennsylvania State University, University Park, Pennsylvania, USA

† Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, Wisconsin, USA

## Introduction

Thioesterases, which catalyze the hydrolysis of acyl-ACP (acyl carrier protein) during type II fatty acid synthesis, determine both fatty acid chain length and overall yields[1]–[3]. The goal of this study is to classify plant acyl-ACP Thioesterases (TE) based on their substrate specificity. Primary sequence information and machine learning have been widely used to classify proteins into known families/classes [4]–[7]. Here, we present a stacked ensemble framework that comprises of three base models which rely on Support Vector Machines [8] and a meta learner which uses a majority voting criterion to predict the substrate specificities of TEs using their primary sequences as features. The heterogeneity among our base models in the ensemble is governed by the different feature representation techniques used to encode the primary sequences of the enzymes. Ensemble methods are known to provide more robust and accurate predictions [9]–[11] compared to any individual model on small training samples which we will eventually illustrate through this study. Our ensemble framework achieves a mean cross validation accuracy of 0.777 across 10,000 simulations of this study using different training and validation sets.

## Methods

### Dataset Compilation

The input dataset consists of 106 TE sequences and their corresponding substrate specificity. These characterized TE were obtained from multiple literature sources [3], [12]–[14]. These enzymes were divided into the following categories to formulate a classification problem:

* TE with greater than 50% specificity above C12 and less than 10% specificity for C12 and below denoted by the label “1”
* TE with greater than 50% specificity for C12 or below and less than 10% specificity above C12 denoted by the label “3”
* TE with mixed specificity denoted by the label “2”

The length of TE sequences ranges from 300 to 412. Figure 1 shows the length of the sequences against the number of TE instances for each sequence length. The maximum count is of sequence length 399 representing the Cuphea viscosissima, henceforth referred to as Cupv, class of enzymes. The dataset has a higher percentage of TE with more than 50% specificity at or below C12 and less than 10% specificity for above C12 substrates. Figure 2 highlights this trait of the dataset.Enzyme class 3 (the TE with more than 50% specificity at or below C12 and less than 10% specificity for substrates above C12) accounts for about 48% of the training data while Enzyme class 1 (TE with more than 50% specificity above C12 and less than 10% specificity for substrates below C12 accounts) accounts for about 35% of the training data. Only 17% of the training data is represented by Enzyme Class 2, (the TE of mixed specificity).

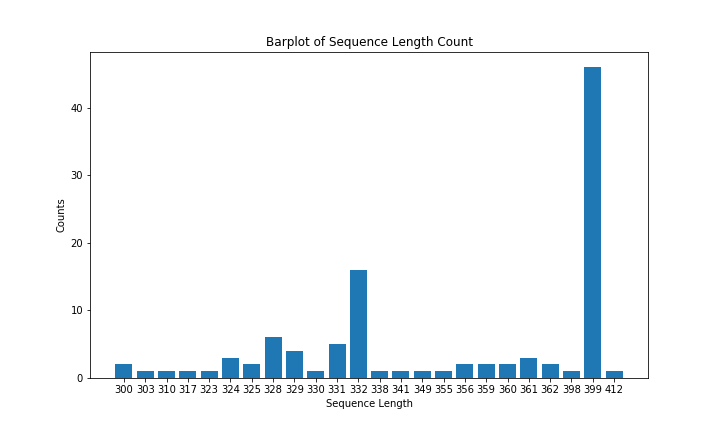


Figure : Sequence length vs count of enzymes with the same sequence length. The most frequent Thioesterase in the dataset is Cupv with 399 residues

A screenshot of a cell phone

Description automatically generated

Figure : Relative frequency distribution of TE arranged in descending order of their magnitude is shown here. Enzyme Class 3 accounts for about 48% of the training data.

### Feature Representation

We have used three different feature representation techniques to encode primary sequence information of the enzymes. The motive behind using different techniques is to extract distinct information from the sequences so that each model can produce relatively uncorrelated output predictions. The three different representations are 1) k-mer motif builder, 2) Grouped Amino Acid encoded motif builder 3) Positional feature builder. Among the three techniques, the first two can be encoded automatically using primary sequences of the enzymes while the last one requires expert level human intervention. Each of the representation techniques is described in the following subsections. The final step of all techniques involved creating a one hot encoded feature vector which is described in Appendix A.

#### k-mer motif builder

This representation is similar to the N-gram representation for language models [15]. Here, the entire training dataset (all enzyme sequences) is scanned to search for all possible sequence motifs of length k. A one hot encoded feature set is created with sequence motifs which are repeated in at least 2 instances of enzymes in the training data. This step is taken for two reasons, 1) To avoid building a large set of features and 2) It is expected that enzymes of a specific substrate specificity will have common motifs and the main aim of the model is to learn these common motifs, not a motif specific to a particular enzyme.

Hypothetical Example:

Given 3 enzyme sequences , and where

and choosing , the set of possible 2-mers are {‘LP’, ‘LT’, ‘ML’, ‘PL’, ‘PW’, ‘ST’, ‘TM’, ‘TP’, ‘TT’, ‘WS’} among which only {‘ML’,‘TP’} are present in at least two instances. Hence the feature set will be a one-hot encoded vector of length 2 with ‘ML’ as the first index and ‘TP’ as the second index. The feature representation for the enzymes will be as follows:

#### Grouped Amino Acid encoded motif builder

This representation technique is similar to the k-mer motif builder, but it performs an additional pre-processing step on the primary sequence before searching for all possible sequence motifs of length k. In the preprocessing step, the 20 amino acid types were categorized into five classes according to their physicochemical properties, e.g. hydrophobicity, charge and molecular size. The five classes were the aliphatic group (: GAVLMI), aromatic group (: FYW), positive charge group (: KRH), negative charged group (: DE) and uncharged group (: STCPNQ). Finally, each amino acid in the sequence was mapped to their respective class to create a compressed sequence with only five different types of values per position. For example, a primary sequence ‘MLTPWS’ is represented as ‘’. Using this compressed representation, k-mer motif builder is used to construct a feature set as described in the above subsection.

#### Positional feature builder

For this feature representation technique, a total of 43 residue positions were selected from the aligned enzyme sequences based on their importance in prior acyl-ACP thioesterase mutagenesis studies as well as from structural analysis of the UcFatB binding pocket [1]. Of these 43 positions, 15 were selected based on a previous study which successfully converted a long-chain acyl-ACP thioesterase from *Cuphea viscosissima* to have short-chain preference by interchanging the residues which varied in a sequence alignment [3]. An additional 3 positions were selected based on an early mutagenesis study of the acyl-ACP thioesterase from *Umbellularia californica*, which converted the predominantly C12-specific thioesterase to a predominantly C14-specific thioesterase using sequence analysis of the homolog from *Cinnamomum camphora* to guide the design [14]. Inspection of the *Umbellularia californica* structure, namely residues within the *α1* helix and the *β2*, *β4*, and *β5* helices, led to the identification of 5 additional positions [1]. The remaining positions not encompassed within the criteria above were selected based on a sequence alignment among 6 thioesterases from 3 plant species. Each of the 3 plant species had a representative of a short-chain and a long-chain acyl-ACP thioesterase. The three species included were *Umbellularia californica* (accession numbers Q41635.1 and Q41634), *Cuphea palustris* (accession numbers Q39554 and Q39555), *Cuphea hookeriana* (accession numbers AAC49269.1 and AAC48990.1), and *Cuphea viscosissima* (accession numbers AEM72522.1 and AEM72523.1). After selecting the 43 positions, amino acids in those positions were represented as a one hot encoded feature vector. Each position apart from the 20 types of amino acids may also contain a gap (result of a multiple sequence alignment). Thus a 21-dimensional one hot encoded feature vector is used to represent each position.

### Ensemble Formulation

Our ensemble framework involved three base learners which provided an output to a meta learner that predicted the enzyme class. Although all of our base learners were trained using the same principle (SVM), the heterogeneity among them was governed by the three different feature representation techniques described above. The workflow of our model is presented Figure 3.

#### Base Model Training

The dataset was divided into a separate training and validation set based on a 75%-25% split. The dimensionality of the feature space was reduced using Principal Component Analysis because of our small sample size. The primary aim of reducing the feature space was to decrease the number of parameters required to train the model and make our model more generalizable. A Support Vector Machine model was trained using the lower dimensional training set. 3-fold cross validation was carried out on the training set to learn the optimal set of hyperparameters. The optimal hyperparameters for each base model is given in Appendix B. The optimal set of hyperparameters were used to retrain an SVM on the entire training set. Finally, the trained SVM was used to predict the enzyme classes of the held-out validation set. This procedure was performed 10,000 times using different random seeds (which changed our training/validation set thus affecting our model) to check the dependence of our model on training data and assess it’s robustness.

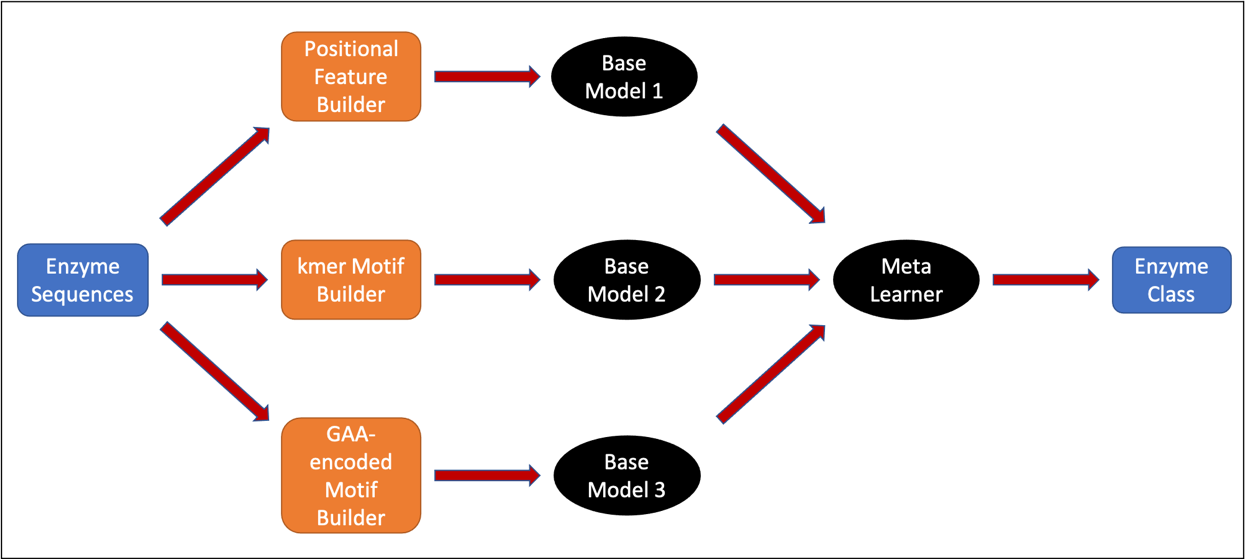


Figure : Workflow of our ensemble model. Different feature representations create separate models, and the final model output is the majority vote of the predictions made by each individual model.

#### Principal Component Analysis (PCA) formulation

The steps involved in PCA were, 1) Standardization: The dataset was initially scaled so that each variable contributes equally.

where is the scaled value, is the original value, is the mean of the variable and is the standard deviation of the variable, 2) Covariance Matrix Computation: The covariance of each variable was calculated using the following formula:

The result was a square matrix of dimensions where is the number of variables, 3) Eigen Vector and Eigen Value Computation: The eigen vector for a matrix was calculated by solving the following formula:

4) k eigenvectors with the largest eigenvalues determination: The eigenvectors were sorted in decreasing order of their eigenvalues and the first k eigenvectors became the Principal Components. A matrix is created by combining the k eigenvectors, 5) Recasting data along the Principal Components: Finally, the original sample space was projected along this new subspace via the following equation:

where was the new sample space and was the previous sample space.

#### Support Vector Machine (SVM) Formulation

Given a training set of instance-label pairs where is a vector of features and is the label, support vector machine [8] solves the following optimization problem:

subject to:

Here the training vector is mapped to a higher dimensional space by the function . is the weight vector that represents the training parameters. is a penalty parameter for the error term . The objective of the SVM is the find a separating hyperplane with the maximal margin defined as between each class.

#### Meta Learner Formulation

Our Meta Learner is a hard-voting based majority vote classifier that compares the predictions of the base learners and outputs the class of enzymes that received the maximum votes. Thus, if the predictions of the three base learners are respectively, then the prediction of the meta learner is . In case there is a three-way tie between outputs of the base learners, the prediction of the positional feature builder based model is selected as the ensemble output.

## Results

### Positional feature builder-based model performed marginally better than other base models

The positional feature builder based model that requires expert intervention to preselect the positions potentially responsible for TE specificity showed slightly better performance compared to the two other automated feature encoder based models. The mean accuracy score for the positional model was 0.771 compared to 0.747 and 0.753 for the kmer based model and grouped amino acid encoded model respectively across 10000 simulations. In terms of worst case accuracy as well, the positional model outperformed the other two (0.44 versus 0.4 for both). The histogram of test set accuracy for the positional model over 10000 simulations is shown in Figure 4 while for the other two models are given in Appendix C.

Chart, histogram

Description automatically generated

Figure : Positional Model test accuracy histogram over 10000 simulations. Model has a worst case accuracy of 0.44, mean accuracy of 0.771 and standard deviation of output prediction, 0.081

### Ensemble model significantly increases worst case accuracy over base models

The primary purpose of using an ensemble framework was to decrease the amount of variance in predictions which was expected in our case given our small sample size. Our results indicate that the ensemble model significantly increases the worst-case prediction accuracy over our best performing base model from 0.44 to 0.52. It also marginally decreased the standard deviation of prediction output from 0.0805 for the positional model to 0.077 and increased the mean accuracy score from 0.771 to 0.777. The histogram of test set accuracy for the ensemble model over 10000 simulations is shown in Figure 5.

Chart, histogram

Description automatically generated

Figure : Ensemble Model test accuracy histogram. The model produced a worst case accuracy of 0.52, mean accuracy of 0.777 and standard deviation of 0.077 over 10000 simulations

## Discussion

Thioesterases comprise a large enzyme group whose members show varying substrate specificity. The ThYme database [16] contains multiple TEs whose substrate specificity remain uncharacterized. Our study has revealed how it is possible to classify TEs according to their substrate specificity using primary sequence information with fairly high accuracy. We created an ensemble framework that uses the output of three different base models to predict the substrate specificity of TEs. We show the advantage of using an ensemble framework over any individual model in terms of both robustness and accuracy especially in problems with small training sets. Our ensemble framework achieved a mean accuracy of 0.777 and a worst-case accuracy of 0.52 on separate held out validation dataset across 10000 simulations. Henceforth, we intend to deploy our model to characterize the substrate specificity of TEs given in the ThYme database. As further validation of our model we plan to experimentally determine the substrate specificity of some of the TEs whose primary sequence information is present in the database to create an independent test set and check our model’s performance on that test set. Additionally, our position specific model currently requires domain specific knowledge and expert level intervention to build the feature set. We are currently working on automating the position selection procedure using Machine Learning.

## References

[1] Y. Feng, Y. Wang, J. Liu, Y. Liu, X. Cao, and S. Xue, “Structural Insight into Acyl-ACP Thioesterase toward Substrate Specificity Design,” *ACS Chem. Biol.*, vol. 12, no. 11, pp. 2830–2836, 2017, doi: 10.1021/acschembio.7b00641.

[2] M. J. Grisewood *et al.*, “Computational Redesign of Acyl-ACP Thioesterase with Improved Selectivity toward Medium-Chain-Length Fatty Acids,” *ACS Catal.*, vol. 7, no. 6, pp. 3837–3849, 2017, doi: 10.1021/acscatal.7b00408.

[3] F. Jing, L. Zhao, M. D. Yandeau-Nelson, and B. J. Nikolau, “Two distinct domains contribute to the substrate acyl chain length selectivity of plant acyl-ACP thioesterase,” *Nat. Commun.*, vol. 9, no. 1, 2018, doi: 10.1038/s41467-018-03310-z.

[4] R. Saidi, M. Maddouri, and E. Mephu Nguifo, “Protein sequences classification by means of feature extraction with substitution matrices,” *BMC Bioinformatics*, 2010, doi: 10.1186/1471-2105-11-175.

[5] D. Wang and G. Bin Huang, “Protein sequence classification using extreme learning machine,” 2005, doi: 10.1109/IJCNN.2005.1556080.

[6] B. C. Smith, B. Settles, W. C. Hallows, M. W. Craven, and J. M. Denu, “SIRT3 substrate specificity determined by peptide arrays and machine learning,” *ACS Chem. Biol.*, 2011, doi: 10.1021/cb100218d.

[7] N. K. Mishra, J. Chang, and P. X. Zhao, “Prediction of membrane transport proteins and their substrate specificities using primary sequence information,” *PLoS One*, 2014, doi: 10.1371/journal.pone.0100278.

[8] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, 1995, doi: 10.1023/A:1022627411411.

[9] P. Yang, Y. Hwa Yang, B. B. Zhou, and A. Y. Zomaya, “A Review of Ensemble Methods in Bioinformatics,” *Curr. Bioinform.*, 2010, doi: 10.2174/157489310794072508.

[10] T. G. Dietterich, “Ensemble methods in machine learning,” 2000, doi: 10.1007/3-540-45014-9\_1.

[11] D. Marbach *et al.*, “Wisdom of crowds for robust gene network inference,” *Nat. Methods*, 2012, doi: 10.1038/nmeth.2016.

[12] T. A. Voelker and H. M. Davies, “Alteration of the specificity and regulation of fatty acid synthesis of Escherichia coli by expression of a plant medium-chain acyl-acyl carrier protein thioesterase,” *J. Bacteriol.*, 1994, doi: 10.1128/jb.176.23.7320-7327.1994.

[13] A. Jones, H. M. Davies, and T. A. Voelker, “Palmitoyl-acyl carrier protein (ACP) thioesterase and the evolutionary origin of plant acyl-ACP thioesterases,” *Plant Cell*, 1995, doi: 10.1105/tpc.7.3.359.

[14] L. Yuan, T. A. Voelker, and D. J. Hawkins, “Modification of the substrate specificity of an acyl-acyl carrier protein thioesterase by protein engineering,” *Proc. Natl. Acad. Sci. U. S. A.*, 1995, doi: 10.1073/pnas.92.23.10639.

[15] D. Jurafsky and J. H. Martin, “Language Modeling with N- grams,” *Speech Lang. Process.*, 2016.

[16] D. C. Cantu, Y. Chen, M. L. Lemons, and P. J. Reilly, “ThYme: A database for thioester-active enzymes,” *Nucleic Acids Res.*, 2011, doi: 10.1093/nar/gkq1072.

[17] S. Developers, “Support Vector Machines — scikit-learn 0.17.1 documentation,” *http://scikit-learn.org*, 2014. .

## Appendix A

### One-Hot Encoded Feature Representation

Each amino acid can be represented as a one-hot encoded vector. Since there are 20 amino acids, a vector of length 20 is defined with each index represented by an amino acid. For a specific amino acid, the index corresponding to that amino acid will be 1 and every other index will be zero. If A is a vector whose index is represented by a specific amino acid , then to represent the amino acid a,

Thus, a sequence of length N can be represented as a N\*20 matrix. This representation of a sequence is not only highly inefficient, but it also fails to capture the contextual information in the data.

## Appendix B

All models were trained using scikit-learn [17] modules. A random seed was specified to reproduce results. The initial dataset was divided into training and validation set based on a 75-25 percentage split. After dividing the models, the training set was encoded into three different feature representations based on the feature building techniques described in the Methods section. After encoding the primary sequences of the enzymes and creating the feature set, Principal Component Analysis was used to reduce the high dimensional feature space into an uncorrelated low dimensional feature space. The number of components used was decided through hyperparameter optimization. Each individual model was then trained using the Support Vector Classifier of scikit-learn library. The class weight was set to “balanced” to take care of imbalance in our dataset. The “balanced” mode automatically adjusts the weights inversely proportional to the class frequencies in the input data. This step was taken to ensure that the model would penalize an incorrect prediction for Enzyme Class 2, which has the least training instances, by a greater factor than that for Class 1 or 3. The SVC regularization hyperparameter C, kernel and kernel coefficient along with the number of PCA components were all optimized using the GridSearchCV module of scikit learn and setting the cross validation split to 3. The optimized hyperparameters were then used to train a model using the training dataset. The trained models were used to predict TE specificities on held out validation set. This whole procedure was repeated 10000 times by varying the random seed which resulted in different training and cross validation set thus affecting the model performance. The objective of training our model multiple times was to check its robustness to the training set. The hyperparameters obtained for each of the three individual models are given in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Name | PCA Components | SVM C Regularization Parameter | SVM Kernel | SVM Gamma |
| Positional feature based | 40 | 1 | linear | scale |
| k-mer motif based | 40 | 1 | linear | scale |
| GAA encoder based | 40 | 20 | linear | scale |

In additional to the above hyperparameters, the k-mer motif and GAA encoded motif models had an additional hyper parameter k that denotes the length of the motif to be considered. It was set to be 7 for both the models.

## Appendix C

Chart, histogram

Description automatically generated

Figure : kmer Motif based model test accuracy histogram over 10000 simulations. Model has a worst case accuracy of 0.4, mean accuracy of 0.747 and standard deviation of output prediction, 0.084

Chart, histogram

Description automatically generated

Figure : GAA encoded motif model test accuracy histogram over 10000 simulations. Model has a worst case accuracy of 0.4, mean accuracy of 0.753 and standard deviation of output prediction, 0.079