NON-CLASSICAL SECRETED PROTEIN MACHINE LEARNING PREDICTION

Final Year Project 1st Phase Report

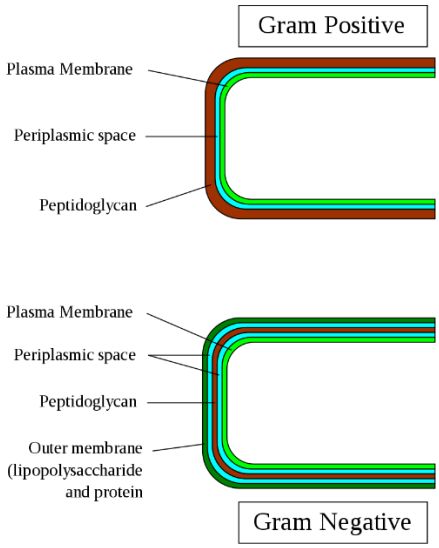
Supervisor: Dr Matthew Chua, Dr Binh Nguyen

Student: Norvin Chandra - A0195335Y

Contents

1. Background

Bacteria species are commonly classified by their Gram staining. Gram staining is method developed by Hans Christian Gram to differentiate bacteria by the chemical and physical properties of their cell walls. Gram-positive bacteria have thick layer of peptidoglycan in cell wall that retains crystal violet. Gram-negative bacteria have thinner peptidoglycan layer that allows crystal violet to wash out on addition of ethanol. This class of bacteria is stained pink or red by counterstain.



Source: Wikipedia

Several Gram-positive bacteria are valuable hosts for production of heterologous proteins as proteins secreted by this class of bacteria are released into culture medium where conditions for correct folding is more appropriate to facilitate isolation and purification of active proteins (Anne et al. 2017). Gram-positive bacteria can secrete protein across their membrane and cell wall using “classical” secretion system via general secretion (Sec) translocon or Twin-Arginine Transport (Tat) pathway. Protein secreted via other pathway is considered as “non-classical” secretion (Yanju Zhang et al. 2019). Given its importance in biotechnology application, identification of protein secreted from Gram-positive bacteria becomes an important task especially protein secreted through “non-classical” pathway given its difficulty to identify due to lack of discernible signal peptide sequence.

2. Deliverables

In this project, we aim to build a classification model to detect “non-classical” secreted protein from Gram-positive bacteria with AUC of more than 0.85 when tested against independent test dataset.

Validation of the model performance will be based on the independent test data set provided in PeNGaRoo website.

3. Current Benchmark Model

The PeNGaRoo team has curated their own verifiable training and independent testing data set on which this project will be based upon.

Current methodology employed by PeNGaRoo team is as follows:

1. Extract features from 3 different groups:
    a. Sequence derived features
    b. Evolutionary information features
    c. Physicochemical property-based features

2. Train Light GBM model on each of the feature group and then use equal-weight averaging of prediction outputs of feature encoding within same group
3. Instead of merging all features into a higher dimensional feature set, the team trained base model with each feature set to learn different useful pattern from various feature sets. When integrating base model, rather than using direct integration which might suffer from information redundancy within same feature group, the team adopted two-layer ensemble model strategy

4. Initial Technical Approach

1. First, we obtain the training and independent test data from PeNGaRoo website
2. We use iFeature web server to generate multiple features from the FASTA format protein sequence.
3. Thereafter we use Auto ML package, TPOT to attempt to discover potential feature and method combination pipelines that generate high accuracy for further study
4. Select 2 or more pipeline combinations to study in detail, manually build and tweak parameters to optimize prediction result
5. We will also explore the possibility to fuse features to train on ensemble models

5. Data Source and Acquisition

Protein sequence data for model training as well as independent test data set are available in PeNGaRoo website, http://pengaroo.erc.monash.edu/download.jsp.

The following data is downloaded for the project:

- PeNGaRoo_train_N.fasta (negative)
- PeNGaRoo_train_P.fasta (positive)
- PeNGaRoo_independent_test_N.fasta
- PeNGaRoo_independent_test_P.fasta

6. Data Processing

- Load each training and independent test data to iFeature webserver to obtain multiple protein features for modelling
- iFeature generated 21 feature sets (note: “descriptors” is not a feature set, rather it is a file containing all features)
- Thereafter combine respective positive and negative feature pairs to form complete training data set and independent testing data set
- Proceed to check the balance of data:

```
tpc_train_N.shape
```

```
(446, 8002)
```

```
tpc_train_P.shape
```

```
(141, 8002)
```

- There is more negative observation than positive observation in the training data set, thus the training data is imbalanced.

	features
0	aac
1	apaac
2	cksaagp
3	cksaap
4	ctdc
5	ctdd
6	ctdt
7	ctriad
8	dde
9	descriptors
10	dpc
11	gaac
12	gdpc
13	geary
14	gtpc
15	ksctriad
16	moran
17	nmbroto
18	paac
19	qsorder
20	socnumber
21	tpc

7. Initial Modelling

a. TPOT package 1st iteration

We run 1st iteration of TPOT using the following parameter setting

```
pipeline_optimizer = TPOTClassifier(generations = 50,
                                    verbosity=2,
                                    max_time_mins=30,
                                    max_eval_time_mins=0.04,
                                    population_size=50,
                                   random_state = random_seed)
```

We limited the run time to 30 mins for each feature set and append the accuracy score on validation test data for each feature set to a data frame that collate score from all feature set. After sorting the score in descending order, we can identify best performing feature sets for further analysis.

	features	score
18	paac	0.925170
1	apaac	0.918367
9	descriptors	0.904762
3	cksaap	0.897959
8	dde	0.884354
21	tpc	0.877551
17	nmbroto	0.863946
10	dpc	0.863946
0	aac	0.863946
7	ctriad	0.857143
4	ctdc	0.857143
19	qsorder	0.857143
16	moran	0.850340
6	ctdt	0.843537
15	ksctriad	0.843537
5	ctdd	0.836735
13	geary	0.836735
14	gtpc	0.829932
12	gdpc	0.823129
2	cksaagp	0.823129
20	socnumber	0.809524
11	gaac	0.789116

b. TPOT package 2nd iteration

2nd iteration was run with max_time_mins removed allowing the algorithm to run all the way to 100 generations per feature set taking approximately 4 days to complete.

```
pipeline_optimizer = TPOTClassifier(generations = 100,
                                    verbosity=2,
                                    max_eval_time_mins=0.04,
                                    population_size=100,
                                   random_state = random_seed)
```

It seems to give quite different rank for each feature set compared to 1st iteration.

3	cksaap	0.938776
8	dde	0.925170
21	tpc	0.918367
9	descriptors	0.918367
25	dpc	0.911565
1	apaac	0.904762
23	ksctriad	0.897959
7	ctriad	0.897959
22	apaac	0.891156
18	paac	0.891156
17	nmbroto	0.891156
0	aac	0.891156
6	ctdt	0.884354
19	qsorder	0.884354
4	ctdc	0.877551
14	gtpc	0.870748
12	gdpc	0.857143
2	cksaagp	0.850340
13	geary	0.843537
16	moran	0.843537
11	gaac	0.843537
20	socnumber	0.829932
5	ctdd	0.823129

c. Attempt to recreate TPOT Iteration 1 "paac" pipeline and test on Independent Test Data

```
pipe2 = make_pipeline(
    make_union(
        FunctionTransformer(copy),
        FunctionTransformer(copy)
    ),

    PCA(iterated_power=5, svd_solver="randomized"),
    ExtraTreesClassifier(bootstrap=False, criterion="gini",
                         max_features=0.7000000000000001,
                         min_samples_leaf=2,
                         min_samples_split=5,
                         n_estimators=100)
)
```

Decent accuracy on validation test data

```
pipe2.fit(paac_X_train, paac_y_train)
results2 = pipe2.predict(paac_X_test)
accuracy_score(paac_y_test, results2)
```

```
0.8707482993197279
```

However poor result on independent test data with accuracy of 0.5

```
ind_result2 = pipe2.predict(paac_indX_test)
accuracy_score(paac_indy_test, ind_result2)
```

```
0.5
```

d. Attempt Stacking a few classifiers on "ctdc" feature set

```
# get a stacking ensemble of models
def get_stacking():
    # define the base models
    level0 = list()
    #level0.append(('lr', LogisticRegression()))
    level0.append(('lsvm', LinearSVC()))
    level0.append(('svm', SVC()))
    level0.append(('et', ExtraTreesClassifier()))
    # define meta learner model
    level1 = LogisticRegression()
    # define the stacking ensemble
    model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
    return model
```

```
def get_models():
    models = dict()
    models['lr'] = LogisticRegression()
    models['lsvm'] = LinearSVC(C=15.0, dual=False,
                               loss='squared_hinge',
                               penalty='l2', tol=0.01)
    models['et'] = ExtraTreesClassifier(bootstrap=False,
                    criterion = 'entropy',
                    max_features=0.3500000000000003,
                    min_samples_leaf=18,
                    min_samples_split=9,
                    n_estimators=100)
    models['svm'] = SVC()
    models['xgb'] = XGBClassifier()
    models['stacking'] = get_stacking()
    return models
```

```
# evaluate a given model using cross-validation
def evaluate_model(model):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
    return scores
```

Decent score on validation data

```
# get the models to evaluate
models = get_models()
```

```
# Training Data
X = scaler.fit_transform(ctdc_train.iloc[:,:-1])
y = ctdc_train.iloc[:,-1]
```

```
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
```

```
>lr 0.886 (0.044)
>lsvm 0.886 (0.051)
>et 0.855 (0.042)
>svm 0.874 (0.040)
>xgb 0.866 (0.050)
>stacking 0.890 (0.041)
```

```
# Apply Stack model
stackmodel = models['stacking'].fit(X,y)
```

Getting poor accuracy of 0.5 again on independent test data

```
ctdc_indy_pred = stackmodel.predict(ctdc_indX_test)
```

```
print("Accuracy:",metrics.accuracy_score(ctdc_indy_test, ctdc_indy_pred))
```

```
Accuracy: 0.5
```

There seem to be problem with the independent test data as regardless of modelling method, the accuracy when tested against independent test data is always exactly 0.5.

8. Challenges

- Understanding each feature set is challenging for someone not in the domain
- Learning new sophisticated modelling technique on the go
- Attempt to reproduce PeNGaRoo model using original source code (in R) and original data set to understand the mechanism resulted in fatal error
- Difficulty in finding source code for journals featuring feature fusion
- Attempt to find potential pipeline using TPOT takes time. 100 generations with population size of 100 takes about 4 days to complete
- Potential pipeline + feature combinations produce poor accuracy when tested on independent test data set
- Lack of alternative independent data set for testing

9. Next Steps

- Further research on sequence-derived features, evolutionary information-based features, physicochemical property-based features
- Further research on other feature transformation / fusion and ensemble modelling technique
- Find out reason for the poor performance of current models when tested on independent test set
- Seek guidance on possible methodology and sample codes to reverse engineer method to implement on this project

10. References

- Gram-positive bacteria. (2020, June 15). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Gram-positive_bacteria#/media/File:Gram-Cell-wall.svg
- Bruckner, Monica Z., "Microbial Life Educational Resources", *Gram Staining,* Carleton College Science Education Resource Center, https://serc.carleton.edu/microbelife/research_methods/microscopy/gramstain.html#:~:text=Gram%20staining%20is%20a%20common,these%20cells%20red%20or%20violet.
- Anne,J. et al. (2017) Protein secretion in Gram-positive bacteria: from multiple pathways to biotechnology. Curr. Top. Microbiol. Immunol., 404, 267–308.
- Zhang, Y. et al (2019) PeNGaRoo, a combined gradient boosting and ensemble learning framework for predicting non-classical secreted proteins. Bioinformatics, 36, 704-712.