Closed Set

<u>Objective:</u> Multiple-instance, Closed-set Butterfly Classification using DiNo Features (d = 384)

<u>Approaches:</u> **I started with whole features.**

<u>Approach 1:</u> Since data is highly imbalanced, for my first approach I combined oversampling and under sampling from imbalanced-learning[1] to generate synthetic data. For oversampling I tried SMOTE and for under sampling I tried Random, Tomek links and Edited Nearest Neighbors.

<u>Approach 2:</u> I tried all my basic models with same hyperparameter range as in Task 1. This time Logistic Regression with class_weight = 'balanced' performed best. SMOTE did not give better result compared to this approach. PCA did not help as feature size is 384. I have used minmax normalization for all my experiments. Results are summarized below:

No.	Approaches	Hyper parameter tuning	Best Validation Mean Class Accuracy

1	**Oversampling + Under sampling**	Solver = Liblinear
		cost_param = [1 10, 20, 30], penalty = 'l2'
			0.7809
			(SMOTE, Logistic Regression, C=20)

2	**Basic Models**	PCA n = [150, 200, 250, 500, 750] C = [0.1, 1, 10, 100]	**0.7813 (Best!)** (n = 250, Logistic Regression, C= 1)

I then shifted my focus to part features.

<u>Approach 1:</u> I averaged 9-part features with different combination of weights. Then I concatenated the average feature to whole feature to get d=768. Giving middle tiles higher weights gave me the best mean class accuracy on validation data.

<u>Approach 2:</u> I combined all features (d=3840). The rationale is to not lose any information and let linear model do the feature selection. This gave me best result among all my methods.

<u>Approach 3:</u> I trained 9 classifiers based on each part TILE, get prediction on validation data, and used them as part weights. Mean class accuracy was worse than just taking average of part features.

<u>Approach 4:</u> Test data has more images for sample id compared to validation images. To further improve my accuracy from my best model, I averaged features from same sample id. I replaced features with the average feature. The rationale was average feature will be more general compared to individual features and force all same sample ids to have same prediction. This approach decreased validation accuracy by 1.5%. But I still wanted to use this on test data as its distribution was different from validation.

<u>Approach 5:</u> I tried PCA and Logistic Regression with penalty = 'l1' on concatenated features to reduce dimensions. My validation accuracy was lower than just using all combined features.

Results are summarized below:

No.	Approaches	Best Validation Mean Class Accuracy
1	**Averaging parts features and concatenate to whole features**	0.8522 (Logistic Regression, C= 10) Giving middle tiles higher weights
2	**Combining all features**	**0.8571 (Best!)** (Logistic Regression, C=10)
3	**Sample id averaging**	0.8435 (Logistic Regression, C=10, d=3840)
4	**TILES approach**	0.8383 (Logistic Regression, C=10)
5	**Dim Reduction**	0.8433 (Logistic Regression, penalty = 'l1', C=15)

Test results:

No.	Model	Validation accuracy	Test accuracy
1	**Averaging parts features and concatenate to whole features (weight = 1/9) + Logistic Regression**	0.8522	0.5517
2	**Combining all features and sample id averaging + Logistic Regression**	0.8435	0.57

Open Set

Objective: Multiple-instance, Open-set Butterfly Classification using DiNo Features (d = 384)

Approaches: This is continuation of Task 2 with open-set classification. **First, I revisited multiple-instance learning to adopt a more methodical approach.**

Approach 1: I tried to implement margin maximization for multiple instances [3]. It is a two-step iterative optimization process illustrated in Figure 2.

My objective function for bi-convex programming was

min $\boldsymbol{\lambda^T \lambda}$ such that

$$\boldsymbol{y_i(w^T X_i{}^* \lambda_i + w_0) \geq 1 - \xi_i}$$

$$\boldsymbol{\sum_j \lambda_j = 1, \qquad \lambda_j \geq 0}$$

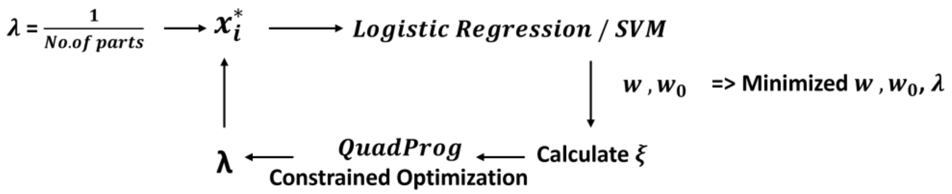
$\lambda = \frac{1}{No. of\ parts}$
x_i^*
Logistic Regression / SVM
w, w_0
=> Minimized w, w_0, λ
Calculate ξ
QuadProg
Constrained Optimization
λ

Figure 2: Margin Maximization for multiple instances [3]

I could not get any optimum solution from Quadprog. My objective function formulation needs to be revisited.

Second, I focused on outlier detection. I have tried OneClassSVM, LocalOutlierFactor before with little success. I have seen one-vs-rest has performed better than other outlier detection approaches. Due to time constraints I decided to focus on one-vs-rest method and try Bayesian Hierarchical Model [5].

Holdout data: I took out 50 classes as unseen which constitute of 38% in validation data. Validation data was 21% of train data which is the same train: test proportion.

Approach 1: **One-vs-rest approach.** I considered both SVM and Logistic Regression. SVM tends to give more outliers compared to Logistic Regression. I also carefully chosen outlier threshold to get my target proportion of outliers.

Approach 2: **Bayesian Hierarchical Model.** It seems intuitive that Bayesian Hierarchical Model should work very well on our butterfly image data as it is inherently hierarchical. I borrowed model from Badirli et. al.[5] work. The authors used side information which I do not have. Instead, I used different clustering approaches to group seen classes to form local priors. During prediction I used these local priors to catch outliers that may belong to that group but not from any seen class. I used two clustering approaches: Based on genus and Agglomerative clustering. Results are summarized below:

No.	**Approaches**	**Hyper parameter tuning**	**Best Validation Accuracy**
1	**One-vs-rest** **(SVM, Logistic Regression)**		seen f1: **0.6854** unseen f1: **0.6800** Harmonic mean: **0.6827** (Logistic Regression, C=10, threshold = 0.5)

2	**Bayesian Hierarchical Model**
	(Clustering = Genus)
		n = 399
		d = 768 (MIL weight λ=1/9)
		k0 = [1, 10, 100]
		k1 = [0.1, 1, 10, 100]
		m = [d+2, 100*d]
		s = [1, 10]
			seen f1: 0.74
			unseen f1: 0.57
			Harmonic mean: 0.65
			(k0=0.10, k1=0.01, m=d+2, s=10.0)

3	**Bayesian Hierarchical Model**	n = [50, 100, 150, 200, 300]
		d = 768
		k0 = [0.1, 1, 10]
		k1 = [0.01, 0.1, 1, 10]
			seen f1: 0.73
			unseen f1: 0.61
			Harmonic mean: 0.67

	(Clustering = Agglomerative)	m = [d+2, 100*d]
		s = [1, 10]
			(n=100, k0=0.10, k1=10, m=100*d, s=1.0)

Inference on Test data:

No.	Approaches	Confidence threshold	Unseen %
1	**One-vs-rest with LR**	-1.8	42.17
2	**One-vs-rest with SVM**	-0.45	43
3	**BHM + Genus**		63
4	**BHM + Agglomerative (n=100)**		38.6

Observations: BHM with Genus clustering gave significantly high outliers (63%). I also needed to be restrictive on confidence threshold for one-vs-rest methods to keep outlier percentage in a reasonable range.

Test results:

No.	Model	Validation Harmonic mean	Test Harmonic mean
1	**One-vs-rest with LR**	0.6827	0.478093
2	**Logistic Regression + (BHM + Agglomerative (n=100)) for unseen classes**	0.68	0.448237
3	**BHM + Agglomerative (n=100)**	0.67	0.441698

Note: **BHM + Agglomerative(n=100) gave unseen accuracy 46% but suffered on seen classes (42.48%).**

Lessons Learnt

1. Ensemble technique does not work well on features generated by deep learning methods.
2. High C in Logistic Regression can overfit data.
3. Investigate test predictions from different models before any submission.
4. One-vs-rest and BHM predictions had only 53% common. Two different methods who gave similar performance but had dissimilar predictions can be combined to get best of both worlds.

References

1. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html
3. Dundar, M. Murat, Sunil Badve, Gokhan Bilgin, Vikas Raykar, Rohit Jain, Olcay Sertel, and Metin N. Gurcan. "Computerized classification of intraductal breast lesions using histopathological images." *IEEE Transactions on Biomedical Engineering* 58, no. 7 (2011): 1977-1984.
4. https://scikit-learn.org/stable/modules/outlier_detection.html
5. Badirli, Sarkhan, Zeynep Akata, and Murat Dundar. "Bayesian zero-shot learning." In *European Conference on Computer Vision*, pp. 687-703. Springer, Cham, 2020.