S4: KL Divergence based Regularization and mathematical derivation for classification problem

Scribe: 4

Student: Pranjal Umesh Kalekar

CS 7840: Foundations and Applications of Information Theory (fa24)

https://northeastern-datalab.github.io/cs7840/fa24/

Lecturers: Wolfgang Gatterbauer, Javeed Aslam

Dec 4, 2024

Understanding of KL Divergence

KL divergence is a statistical measure used to compare probability density functions (PDFs). In simpler terms, it quantifies how much one probability distribution differs from another.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

$D_{\mathrm{KL}}(P \parallel Q)$ is a type of statistical distance: a measure of how much a model probability distribution Q is different from a true probability distribution P

Using KL Divergence in Combination of Cross Entropy as Regularizing term in Loss Computation

KLCE regularization loss introduced at ICLR 2024 in a tiny paper with some results

KLCE: Regularized Imbalance Node-classification Via KL-divergence and Cross-Entropy

Mohammad T. Teimuri[1], Zahra Dehghanian[1], Gholamali Aminian[2], and Hamid R. Rabiee[1]

[1]*Sharif University of Technology*
[2]*Alan Turing Institute*

KLCE regularization quantifies the difference between the GNN's predicted class distribution ($P(\hat{Y}|X)$) and a target distribution (P_k) that is designed to give more weight to minority classes using KL Diveregence term and CE term with hyperparameters λ and $\lambda KLCE$

Formulation of the concept in mathematical form

KLCE regularization, which combines KL divergence with cross-entropy in the GNN's loss function eventually added to the baseline

$$\mathbf{KLCE} := \lambda_{\mathrm{KLCE}} \mathrm{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) + \mathrm{KL}(P(\hat{\mathbf{Y}}|\mathbf{X}) \| \mathbf{P}_k)$$

Target Class Distribution (Pk): KLCE uses a target class distribution (Pk) that can be adjusted to give more weight to minority classes

KL Divergence Term: measures the difference between the learned distribution of class predictions by the GNN, denoted as $P(\hat{Y}|X)$, and the target distribution **(Pk)**

$$\mathrm{KL}(P(\hat{\mathbf{Y}}|\mathbf{X}) \| \mathbf{P}_k) = \sum_{j=1}^{k} P(\hat{Y} = j|\mathbf{X}) \log \left(\frac{P(\hat{Y} = j|\mathbf{X})}{P_j} \right)$$

Cross-Entropy Term: It focuses on the expected coding length of the data, providing an additional measure of how well the model's predictions align with the desired distribution.

$$\mathrm{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) = -\sum_{j=1}^{k} P(\hat{Y} = j|\mathbf{X}) \log(P_j)$$

Building the KLCE Loss Function:

Finding optimal values of hyperparameters lambda and LambdaKLCE, will lead to the best performance – Optuna is used for optimization minimizing performance measure

Construction of the custom KLCE regularizing loss function and addressing class imbalance while constructing the target distribution.

```python
def klce_loss(p_pred, p_target, baseline_loss, lambda_, lambda_klce):
    """
    KLCE Loss with weighted regularization terms.
    Args:
        p_pred: Predicted class probabilities.
        p_target: Target class distribution.
        baseline_loss: Cross-entropy or baseline loss.
        lambda_: Weight for baseline loss.
        lambda_klce: Weight for KLCE regularization.
    Returns:
        Combined loss value.
    """
    p_target = p_target.unsqueeze(0).expand_as(p_pred)

    kl_divergence = torch.sum(p_pred * torch.log(p_pred / (p_target+ 1e-8)), dim=-1)

    cross_entropy = -torch.sum(p_target * torch.log(p_pred+ 1e-8), dim=-1)

    klce = (lambda_klce * cross_entropy) + kl_divergence

    return baseline_loss + (lambda_ * (klce.mean()))
```

```python
def calculate_target_distribution(data, num_classes):
    """
    Calculate the target class distribution (Pk).
    Args:
        data: Graph data object with labels.
        num_classes: Total number of classes.
    Returns:
        Normalized inverse class distribution.
    """
    _, class_counts = torch.unique(data.y[data.train_mask], return_counts=True)

    class_ratios = class_counts/class_counts.sum()

    inverse_ratios = 1.0 / (class_ratios)

    p_k = inverse_ratios / inverse_ratios.sum()
    return p_k
```

Pk for the regularization term involves two steps. the ratio of each class and a normalized inverse ratio each class. The inversion is intended to emphasize the significance of the minority class.

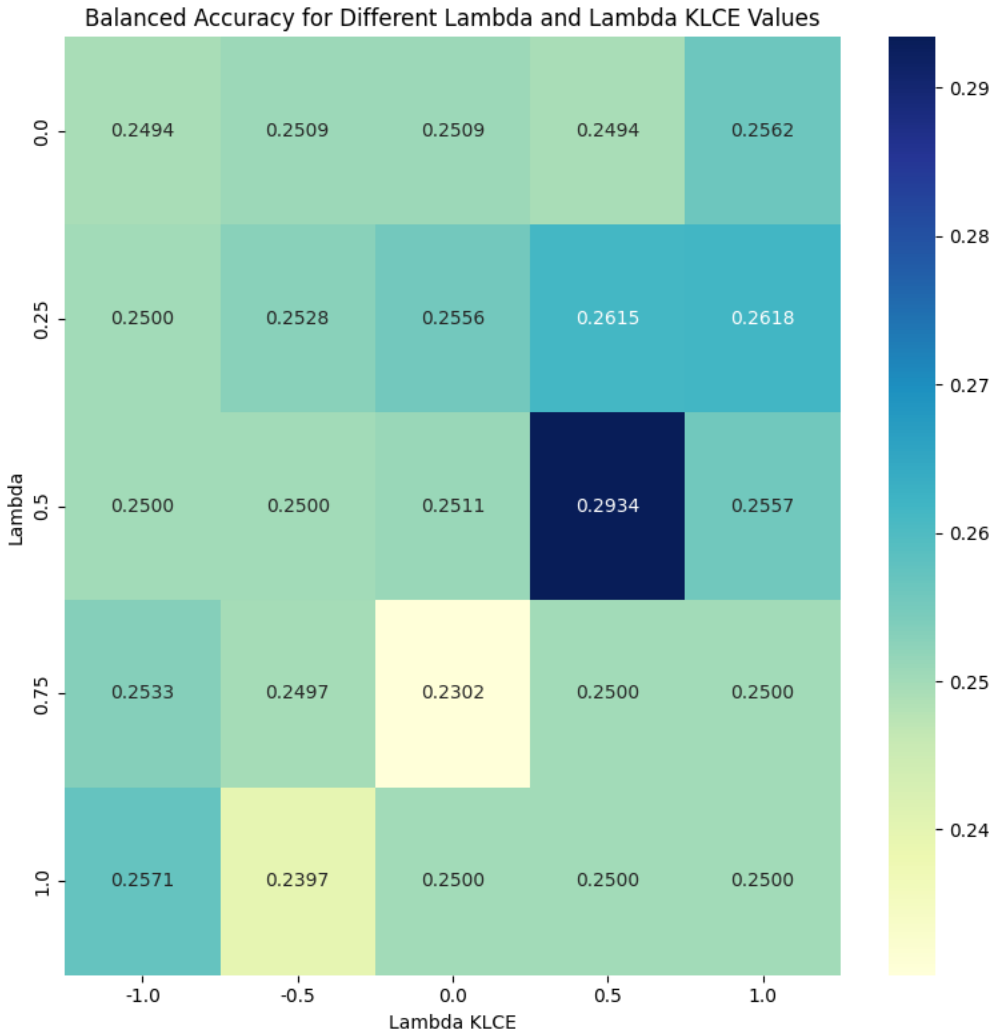
Preparing Highly Imbalanced Synthetically data

Observing the effect of KLCE on highly imbalanced dataset – ideally should be significantly more than other loss functions

Synthetically generated data was trained on 3-layer linear network and softmax activation and performance was evaluated using balanced accuracy and F1 score with variable range of Lamda and LambdaKLCE

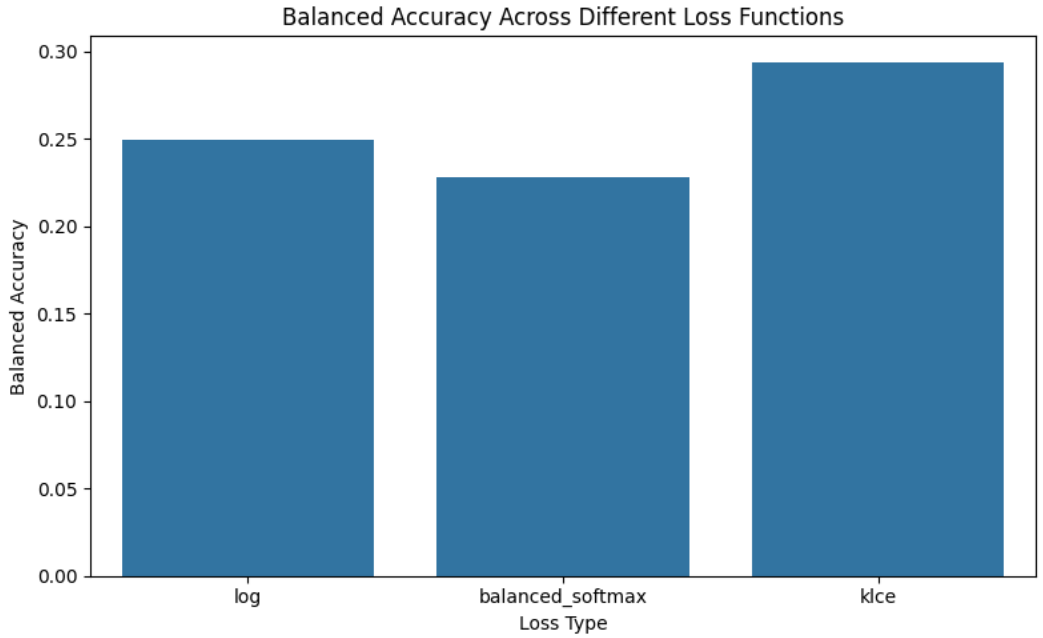
```
Class Distribution:
Class 0: 1022 samples (51.10%)
Class 1: 578 samples (28.90%)
Class 2: 275 samples (13.75%)
Class 3: 125 samples (6.25%)
```

Results on Synthetic data

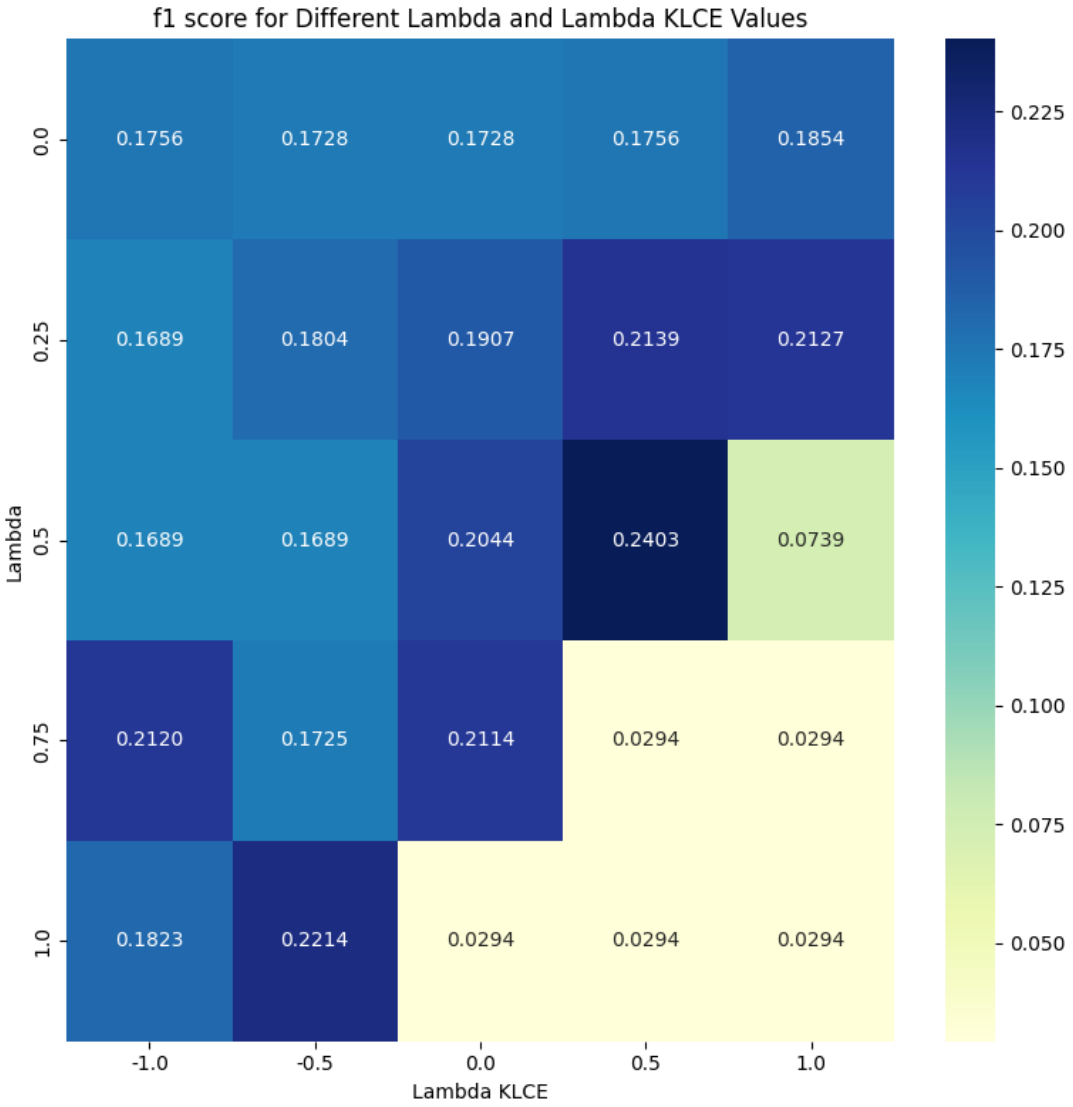


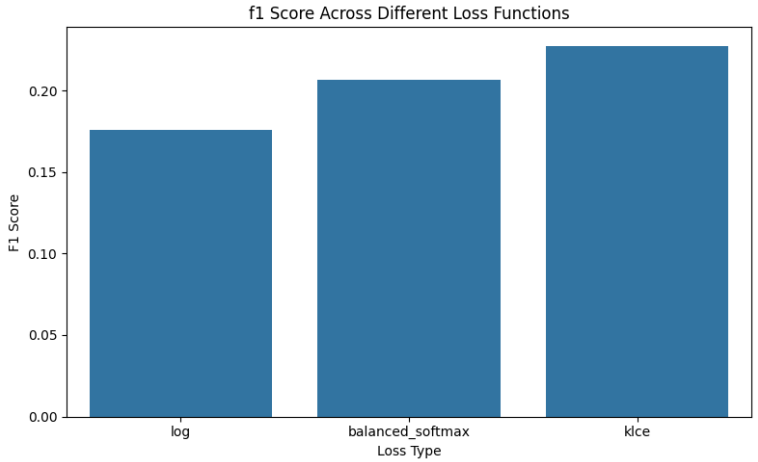
Loss Function Comparison:

	Loss Type	Balanced Accuracy
0	log	0.249408
1	balanced_softmax	0.228424
2	klce	0.294150



Results on Synthetic data





Both performance measures lead us to optimal value of 0.5 for both hyperparameters.

KLCE excels on highly imbalanced data, as expected, showcasing substantial improvements of nearly 18% and 29% over the baseline log loss and balanced softmax methods, respectively.

Continuation of this concept will be covered in final project where the comparison will be done with GNNs for Node classification

References:

- https://proceedings.neurips.cc/paper_files/paper/2003/file/0abdc563a06105aee3c6136871c9f4d1-Paper.pdf, A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications