COMP 551 - MiniProject MiniProject 4: Reproducibility in Machine Learning, Fall 2023

Johan Trippitelli - 260917958, Minh Anh Trinh - 260853143, Chris Chong - 260976714

Dec 2, 2023

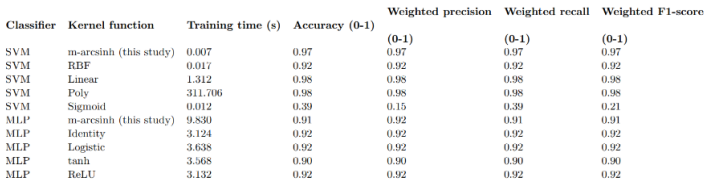
Abstract: Kernel and activation functions are a key part of allowing machine learning algorithms like Support Vector Machine (SVM) and Multi-Layer perceptron (MLP) to do supervised learning. The purpose of the original paper [3] is to demonstrate "m-arcsinh", a modified ("m") version of the inverse hyperbolic sine function ("arcsinh"), as an effective and efficient kernel function for both SVM and activation function for MLP. Results shown from a subset of the original paper's classification datasets from scikit-learn are shown. Using this proposed function, competitive performance is demonstrated when compared to the default "gold standard" kernel functions and activation functions available from scikit-learn.

Introduction: In this project, we are investigating the performance of using m-arcsinh as a kernel function for SVM, as an activation function for MLP, in comparison to the current "gold-standard" sets provided in scikit-learn. Currently, these sets are mutually exclusive, and may have slow/no convergence when applying them to classification. The models to be examined are SVM and MLP, using the same hyperparameters and configurations as the original study [3]. When attempting to redefine a new function that can be used for both SVM and MLP two important conditions must be met. 1) The function must maximize the margin width (difference between left and right margin x perpendicular unit vector) must be properly maximized. 2) The function must improve (or maintain) the standard of classification for an MLP when compared to the "gold-standard". The proposed function, m-arcsinh(x), achieved this by leveraging a weighted interaction between the inverse hyperbolic sine function, m-arcsinh(x) (suitable for MLP), and a slightly nonlinear function, $\sqrt{x}$ (suitable for SVM). These functions were combined to form: $f(x) = 14\sqrt{x} \cdot 13m - arcsinh(x)$ displayed in 8g. which results in a function that is almost linear, good for SVM, but allows for the classification of non-perfectly separable data.

Datasets: In the original paper, 15 different datasets were used. For our paper, we've selected a subset of 5 datasets to be tested and compared with the original results. The first dataset is the "breast cancer wisconsin" datasets - which is a binary classification dataset with 569 total samples and a dimensionality of 30 [4]. The second dataset is the "iris" dataset which consists of 150 flowers samples and 4 features, which classify to one of 3 different types of irises [5]. The third dataset is the "olivetti faces" dataset which contains 10 different 64x64 images of 40 distinct subjects. So a total of 400 samples, which are classified to one of the 40 subjects [6]. The fourth dataset is the wine dataset, which has 178 samples and 13 features for each sample, with each instance being classified to one of 3 types of wines [1] The fifth dataset is the digits dataset, which is made up of 1797 8x8 images of digits ranging from 0-9, with each image being classified to one of the digits ranging from 0-9 [3].

Results of Experiments - Comparing to Original Paper's Results : To evaluate how competitive m-arcsinh could be as a kernel function for SVM and an activation function for MLP, accuracy score was used for performance, and precision, recall, and f1-score was used for reliability, with computational cost being assessed through training time in seconds. Unlike the original paper which used a "AMD E2-9000 Radeon R2 Processor and 4GB of RAM", our results were computed on Google CoLab, which uses an Intel Xeon CPU and 13 GB of RAM [2].

As shown in the below tables, where we have the original results and our experiment results for the breast cancer dataset, although the numbers are not exact, the same trends found in the original paper can be shown. For example, the SVM poly kernel function having a high training time, SVM sigmoid having a low accuracy score, and MLP m-arcsinh having a high training time.



Classifier	Kernel function	Training time (s)	Accuracy (0-1)	Weighted precision (0-1)	Weighted recall (0-1)	Weighted F1-score (0-1)
SVM	m-arcsinh (this study)	0.007	0.97	0.97	0.97	0.97
SVM	RBF	0.017	0.92	0.92	0.92	0.92
SVM	Linear	1.312	0.98	0.98	0.98	0.98
SVM	Poly	311.706	0.98	0.98	0.98	0.98
SVM	Sigmoid	0.012	0.39	0.15	0.39	0.21
MLP	m-arcsinh (this study)	9.830	0.91	0.92	0.91	0.91
MLP	Identity	3.124	0.92	0.92	0.92	0.92
MLP	Logistic	3.638	0.92	0.92	0.92	0.92
MLP	tanh	3.568	0.90	0.90	0.90	0.90
MLP	ReLU	3.132	0.92	0.92	0.92	0.92

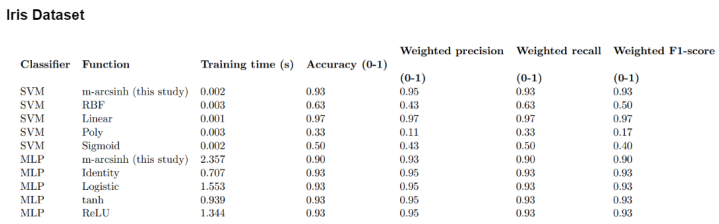
Figure 1: Original Paper Results for Breast Cancer Dataset:

Classifier	Kernel Function	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	0.008	0.99	0.99	0.99	0.99
SVM	RBF	0.0182	0.91	0.92	0.91	0.91
SVM	Linear	1.599	0.96	0.96	0.96	0.96
SVM	Poly	121.889	0.95	0.95	0.95	0.95
SVM	Sigmoid	0.012	0.38	0.14	0.38	0.21
MLP	m-arcsinh	24.659	0.96	0.96	0.96	0.96
MLP	Identity	1.2134	0.95	0.95	0.95	0.95
MLP	Logistic	1.458	0.96	0.96	0.96	0.95
MLP	tanh	2.707	0.96	0.97	0.96	0.96
MLP	ReLU	1.702	0.94	0.94	0.94	0.94

Figure 2: Experimental Results for Breast Cancer Dataset:

This same pattern continues in the comparison with the original results and experiment results on the iris dataset. Similar trends between the original results and the experiment results can be found with the experimental training times for SVM being in the same range as in the original results, and SVM m-arcsinh and SVM linear having higher accuracy scores for SVM. Also, for MLP, m-arcsinh has the highest training time, with the performance numbers for all activation functions for MLP being high and within range of the original results. This is shown in the results for all the other 3 datasets: olivetti faces, wine, and digits which can be found in the appendix. Results for the Olivetti dataset for the original paper and our experimental testing can be found at 8a and 8b respectively. Results for the Wine dataset for the original paper and our experimental testing can be found at 8c and 8d respectively. Results for the Digits dataset for the original paper and our experimental testing can be found at 8e and 8f respectively.

For SVM, the hyperparameters used were the same as outlined in the original paper. With a gamma value set to 0.001, random_state = 13, and class_weight = "balanced". As well for MLP, the random_state was set to 1, and max_iter = 300. Despite this, the numbers we got from our results weren't exactly matching with the original results, though the overall patterns that appeared in the original results were still present in our experimental results. This could be attributed to slight changes in the datasets in the years since this original paper was done.



Iris Dataset

Classifier	Function	Training time (s)	Accuracy (0-1)	Weighted precision (0-1)	Weighted recall (0-1)	Weighted F1-score (0-1)
SVM	m-arcsinh (this study)	0.002	0.93	0.95	0.93	0.93
SVM	RBF	0.003	0.63	0.43	0.63	0.50
SVM	Linear	0.001	0.97	0.97	0.97	0.97
SVM	Poly	0.003	0.33	0.11	0.33	0.17
SVM	Sigmoid	0.002	0.50	0.43	0.50	0.40
MLP	m-arcsinh (this study)	2.357	0.90	0.93	0.90	0.90
MLP	Identity	0.707	0.93	0.95	0.93	0.93
MLP	Logistic	1.553	0.93	0.95	0.93	0.93
MLP	tanh	0.939	0.93	0.95	0.93	0.93
MLP	ReLU	1.344	0.93	0.95	0.93	0.93

Figure 3: Original Paper Results for Iris Dataset:

Classifier	Kernel Function	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	0.0025	1	1	1	1
SVM	RBF	0.00367	0.67	0.53	0.67	0.57
SVM	Linear	0.0025334	1	1	1	1
SVM	Poly	0.003284	0.42	0.17	0.42	0.25
SVM	Sigmoid	0.00357	0.47	0.48	0.47	0.34
MLP	m-arcsinh	11.616	1	1	1	1
MLP	Identity	0.2894	1	1	1	1
MLP	Logistic	0.4894	1	1	1	1
MLP	tanh	0.8652	1	1	1	1
MLP	ReLU	0.5199	1	1	1	1

Figure 4: Experimental Results for Iris Dataset:

From your experimental results, did you reach the same conclusion as the authors?: For SVM, the hyperparameters used were the same as outlined in the original paper. With a gamma value set to 0.001, random_state = 13, and class_weight = "balanced". As well for MLP, the random_state was set to 1, and max_iter = 300. Despite this, the numbers we got from our results weren't exactly matching with the original results, though the overall patterns that appeared in the original results were still present in our experimental results. This could be attributed to slight changes in the datasets in the years since this original paper was done.

As shown in the results for the breast cancer dataset which can be found at 2, the reliability of m-arcsinh for SVM is high as the f1-score was 0.99 compared to the f1-scores of 0.21 and 0.91 when using sigmoid or RBF. This is while m-arcsinh has a training time of 0.008s, compared to the training times of 0.012s and 121.89s for sigmoid and poly kernel functions respectively. The original paper's results pictured at 1 showed a similar trend which was pointed out by the author. So the overall conclusion that m-arcsinh is a reliable and efficient function to act as both a kernel function for SVM and an activation function for MLP, is indeed the same as the authors of the original paper.

However, something of note is that the author didn't point out the high training time for MLP when using arcsinh, only highlighting the very performance and reliability scores from the breast cancer dataset. In our experiment results, we found that the training times for m-arcsinh for MLP across all datasets were significantly higher than what was shown in the original paper's results. For example, in the results for the iris dataset as shown in 4, MLP m-arcsinh training time was 11.62s compared to the original paper's results of 2.36s as shown in 3. However, this can be attributed to the fact that our MLP implementation was vastly different compared to the way the author described their implementation, because the original paper edited the source code of scikit-learn directly, whereas our implementation is a simple neural network using the custom m-arcsinh function. The high training time for MLP when using m-arcsinh is important as it outlines how although m-arcsinh can be used for both SVM and MLP, there is still much room for improvement on the MLP side.

Specify the hyperparamter tuning and ablation studies that you have performed and their results:

Classifier	Kernel Function	Gamma value	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	1e-05	0.0026	1.0	1.0	1.0	1.0
SVM	m-arcsinh	0.001	0.0022	1.0	1.0	1.0	1.0
SVM	m-arcsinh	0.1	0.0022	1.0	1.0	1.0	1.0
SVM	m-arcsinh	1	0.0022	1.0	1.0	1.0	1.0
SVM	m-arcsinh	10	0.0022	1.0	1.0	1.0	1.0

Figure 5: Variation of Gammas for Iris Dataset

Classifier	Kernel Function	Gamma value	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	1e-05	0.012	0.99	0.99	0.99	0.99
SVM	m-arcsinh	0.001	0.0083	0.99	0.99	0.99	0.99
SVM	m-arcsinh	0.1	0.0083	0.99	0.99	0.99	0.99
SVM	m-arcsinh	1	0.0084	0.99	0.99	0.99	0.99
SVM	m-arcsinh	10	0.0092	0.99	0.99	0.99	0.99

Figure 6: Variation of Gammas for Breast Cancer Dataset

With the kernel function m-arcsinh and the variation of the gamma parameter values for SVM (0.00001, 0.001, 0.1, 1, 10), testing on two datasets: the Iris Dataset and Breast Cancer Dataset, shows consistent results in accuracy score, weighted precision, weighted recall, and weighted F1-score. For the Iris Dataset, all four measures consistently yield a perfect score of 1.0 as shown in figure 5, while for the Breast Cancer Dataset, these values remain high at 0.99 as shown in figure 6. The only noticeable difference occurs in training time, where the smallest gamma value (1e-5) shows the highest training time for both the Iris and Breast Cancer datasets, with respective values of 0.0026s and 0.012s. This suggests that the choice of gamma values within the specified range does not significantly impact the model's predictive performance on these datasets. The stability of accuracy and other evaluation metrics across different gamma values implies that the m-arcsinh kernel function, combined with the selected parameters, results in a robust and consistent performance. While there is a marginal variation in training times, it is worth noting that even the highest training time remains relatively low.

Any necessary details for reproducing the results, but were not specified in the original paper: Some specifics that were left out of the paper, but were needed to reproduce the results included the method of calculating the training time, preprocessing for the datasets and loading them into the models, and using a custom activation function for MLP, without editing the source code for scikit-learn. In order to reproduce the results from the paper a new function m_arcsinh was created to be used as the kernel of SVMs and the hidden layer activation of a simple MLP. The new function was defined as follows in 7:

```
# Updated custom activation function
def m_arcsinh_tf(x):
    return tf.multiply(
        1 / 3 * tf.asinh(x),
        1 / 4 * tf.sqrt(tf.abs(x))
    )
```

Figure 7: custom MLP activation function for modified arcsinh

This new function was utilized in the SVM by simply calling our defined classifier with the function definition instead of a string representation to the industry standard function (such as "rbf" or "sigmoid"). This functionality was, however, not present when defining the sklearn MLP classifier. The documentation related to the paper suggested altering the _base.py file within the local machine's sklearn files to include the m_arcsinh function. However, after some research it was determined that it was not recommended to alter the library's base code. Therefore instead of following the paper's guidelines a simple model was created using the tensor flow python library. By configuring the tensor flow classifier extra code was added but the industry guidelines were met and the classifier was defined to be identical to the sklearn classifier. However one drawback

from creating our own classifier was reduced performance. As a result of this for some of the testing using the MLP the max_iters were surpassed. In order to maintain the relationship of our data the data from these tests were still included in the report. However, with more time to research some optimization methods such as momentum or penalization methods could lead to quicker convergence and a better replication of the data seen from the paper's results.

Challenges that you have faced and how did you solve them: To calculate training time, the time would be recorded before fitting the data to a specific classifier, then recorded again after the data was done fitting to a specific classifier. The difference between these two values would be calculated and this would be used as the training time. For preprocessing on datasets, the datasets would be loaded using the scikit-learn load functions provided for each classification dataset, and divided into an X data variable and a y target variable, then split to the same test sizes as highlighted in the original paper. For example, the digits dataset was split into a 70-30 training-testing split, and the breast cancer dataset had a 80-20 dataset. Since our experiments were run on CoLab instead of a local environment, we couldn't edit the source code of scikit-learn to add the m-arcsinh function to the MLP classifier, so we instead built a simple neural network using tensor flow for the new activation function, and substituted that in our experiment results.

Summarize the key takeaways from the project and possibly directions for future investigation: Although m-arcsinh isn't the absolute best kernel function for SVM and activation function for MLP in every single scenario, it still achieves high performance in comparison to the current "gold standard" functions that are given by default in scikit-learn. As stated in the original paper, it is a strong candidate to be considered when searching for an optimal kernel/activation function for SVM/MLP. Changing the hyperparameter gamma for SVM has no effect on its measurement metrics. However, in the original paper, the author states that m-arcsinh can be used where "powerful computational hardware may not always be available", but its high training time in comparison to other methods in our testing could cause that to be challenging in a real world scenario. This function's strength is its applicability to both SVMs and MLPs, beyond solely the performance numbers it can achieve. A direction for future investigation would be modifying m-arcsinh further to optimize for its high training time for MLP.

References and Appendix:
Figure 5:

Olivetti Faces

Classifier	Function	Training time (s)	Accuracy (0-1)	Weighted precision (0-1)	Weighted recall (0-1)	Weighted F1-score (0-1)
SVM	m-arcsinh (this study)	0.143	0.91	0.91	0.91	0.90
SVM	RBF	1.452	0.31	0.28	0.31	0.27
SVM	Linear	1.124	0.99	0.99	0.99	0.99
SVM	Poly	1.071	0.85	0.85	0.85	0.83
SVM	Sigmoid	1.364	0.00	0.00	0.00	0.00
MLP	m-arcsinh (this study)	105.341	0.75	0.78	0.75	0.75
MLP	Identity	109.109	0.75	0.78	0.75	0.75
MLP	Logistic	94.982	0.75	0.78	0.75	0.75
MLP	tanh	103.759	0.75	0.78	0.75	0.75
MLP	ReLU	104.581	0.75	0.78	0.75	0.75

(a) Original Paper Results for Olivetti Faces Dataset:

Classifier	Kernel Function	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	0.115	0.88	0.89	0.88	0.86
SVM	RBF	0.364	0.33	0.35	0.33	0.32
SVM	Linear	0.40	0.98	0.98	0.98	0.97
SVM	Poly	0.316	0.83	0.87	0.83	0.82
SVM	Sigmoid	0.31	0	0	0	0
MLP	m-arcsinh	20.35	0.96	0.97	0.96	0.96
MLP	Identity	10.41	0.96	0.98	0.96	0.96
MLP	Logistic	6.65	0.94	0.96	0.94	0.94
MLP	tanh	13.23	0.96	0.96	0.96	0.96
MLP	ReLU	12.58	0.83	0.86	0.83	0.81

(b) Experimental Results for Olivetti Faces Dataset

Wine Dataset

Classifier	Function	Training time (s)	Accuracy (0-1)	Weighted precision (0-1)	Weighted recall (0-1)	Weighted F1-score (0-1)
SVM	m-arcsinh (this study)	0.003	0.89	0.89	0.89	0.89
SVM	RBF	0.003	0.72	0.77	0.72	0.72
SVM	Linear	0.162	0.94	0.95	0.94	0.94
SVM	Poly	0.094	0.97	0.97	0.97	0.97
SVM	Sigmoid	0.003	0.31	0.09	0.31	0.14
MLP	m-arcsinh (this study)	0.008	0.72	0.77	0.72	0.72
MLP	Identity	0.016	0.72	0.77	0.72	0.72
MLP	Logistic	0.008	0.72	0.77	0.72	0.72
MLP	tanh	0.001	0.72	0.77	0.72	0.72
MLP	ReLU	0.008	0.72	0.77	0.72	0.72

(c) Original Paper Results for Wine Dataset:

Classifier	Kernel Function	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	0.004	1	1	1	1
SVM	RBF	0.005	0.75	0.81	0.75	0.76
SVM	Linear	0.27	1	1	1	1
SVM	Poly	0.31	1	1	1	1
SVM	Sigmoid	0.004	0.22	0.05	0.22	0.08
MLP	m-arcsinh	5.78	1	1	1	1
MLP	Identity	0.0214	0.28	0.13	0.28	0.17
MLP	Logistic	0.26	1	1	1	1
MLP	tanh	0.49	0.97	0.97	0.97	0.97
MLP	ReLU	0.22	0.92	0.93	0.92	0.92

(d) Experimental Results for Wine Dataset

Digits Dataset

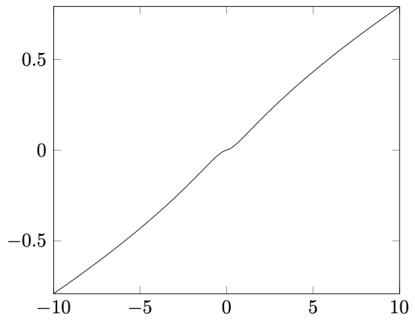
Classifier	Function	Training time (s)	Accuracy (0-1)	Weighted precision (0-1)	Weighted recall (0-1)	Weighted F1-score (0-1)
SVM	m-arcsinh (this study)	0.037	0.95	0.95	0.95	0.95
SVM	RBF	0.116	0.97	0.97	0.97	0.97
SVM	Linear	0.033	0.93	0.93	0.93	0.93
SVM	Poly	0.043	0.95	0.95	0.95	0.95
SVM	Sigmoid	0.332	0.68	0.69	0.68	0.66
MLP	m-arcsinh (this study)	28.650	0.92	0.92	0.92	0.92
MLP	Identity	5.452	0.91	0.91	0.91	0.91
MLP	Logistic	14.182	0.94	0.94	0.94	0.93
MLP	tanh	7.258	0.93	0.93	0.93	0.93
MLP	ReLU	7.834	0.92	0.92	0.92	0.92

(e) Original Paper Results for Digits Dataset:

Classifier	Kernel Function	Training Time(s)	Accuracy Score	Weighted Precision	Weighted Recall	Weighted F1-score
SVM	m-arcsinh	0.043	0.98	0.98	0.98	0.98
SVM	RBF	0.076	0.99	0.99	0.99	0.99
SVM	Linear	0.0272	0.98	0.98	0.98	0.98
SVM	Poly	0.03	0.99	0.99	0.99	0.99
SVM	Sigmoid	0.230	0.72	0.74	0.72	0.71
MLP	m-arcsinh	42.28	0.97	0.97	0.97	0.97
MLP	Identity	1.968	0.97	0.97	0.97	0.97
MLP	Logistic	2.72	0.98	0.98	0.98	0.98
MLP	tanh	2.72	0.97	0.97	0.97	0.97
MLP	ReLU	1.316	0.98	0.98	0.98	0.98

(f) Experimental Results for Digits Dataset

Figure 6g:



(g) Proposed Function:

r

References

[1] Stefan Aeberhard. The wine dataset, 1994.

[2] Tamal Das. Google colab: Everything you need to know, 2023.

[3] Luca Parisi. m-arcsinh: An efficient and reliable function for svm and mlp in scikit-learn. *arXiv*, 2020.

[4] scikit learn. breast cancer wisconsin dataset (classification)., 2023.

[5] scikit learn. The iris dataset, 2023.

[6] scikit learn. The olivetti faces dataset, 2023.