**Machine Learning Optimization Assignment: Otomoto Marketing Campaigns**

* **Objective**

Optimize an existing Artificial Neural Network (ANN) model for Otomoto’s marketing segmentation by using some of the best available optimization methods to make it work better and help the campaigns do their job more efficiently.

**Recreate the Existing Model**

* **Baseline ANN Model with SGD optimizer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metric** | **Class 0** | **Class 1** | **Macro Avg** | **Weighted Avg** |
| **Precision** | 0.84 | 0.68 | 0.76 | 0.8 |
| **Recall** | 0.91 | 0.54 | 0.72 | 0.81 |
| **F1-Score** | 0.88 | 0.6 | 0.74 | 0.8 |
| **Support** | 1036 | 373 | 1409 (total) | 1409 (total) |
| **Accuracy** |  |  |  | **0.81** |

* **Model Performance**:

Accuracy is 81%, which seems high because there are way more samples of one type of data than the other.

Macro average F1-score (0.74) shows how well the model does overall, taking into account both classes and not giving more weight to any one class than the other.

Weighted average F1-score (0.80) helps correct for the fact that some classes have more examples than others and makes it look like the model is doing better overall.

Class imbalance mitigation (like adding extra examples in Class 1 or changing the weights of each class) might need to be used to get better results in that class.

**Select Optimization Algorithms**

**Candidate Algorithms & Justification**

| **Algorithm** | **Use Case** | **Pros** | **Cons** |
| --- | --- | --- | --- |
| **Adam** | Default choice for most ANNs | automatically adjusts the learning rate | often learns quickly without getting stuck |
| **RMSprop** | deep and recurrent networks | best for non-stationary objectives | prone to changes in parameters |
| **Adagrad** | Sparse data (e.g., categorical features) | Changes the learning rates for each parameter | Learning rates can get too small |
| **Genetic Algorithm** | Genetic algorithm helps adjust hyperparameters | easier to improve results overall and find the best solution | takes a lot of processing time. |

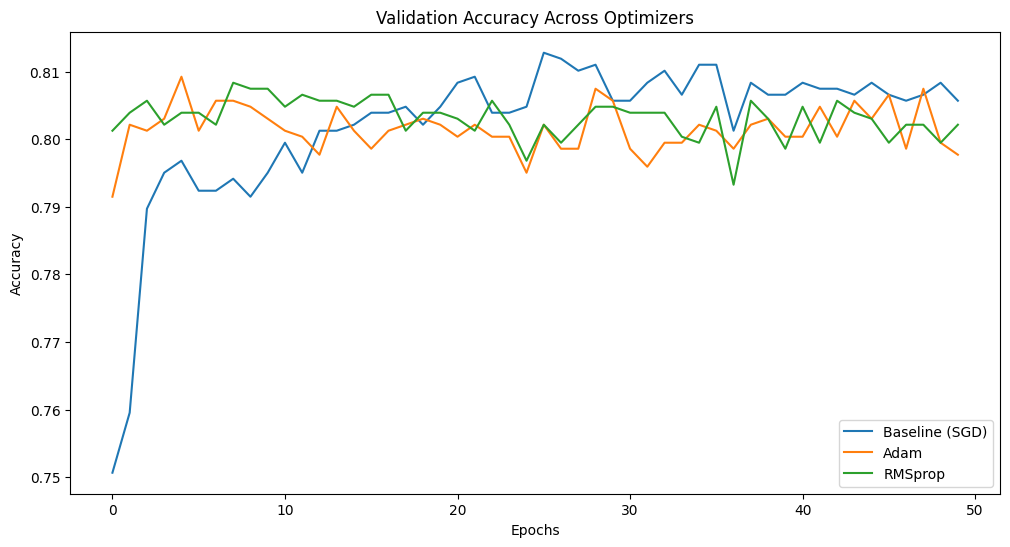
* Baseline optimizer: Stochastic Gradient Descent (SGD).
* According to Kingma and Ba (2015), Adam (Adaptive Moment Estimation) is a well-known adaptive optimizer.
* RMSprop: An optimizer that works well with noisy gradients and recurrent data (Tieleman and Hinton, 2012).
* (Akiba et al., 2019) Optuna: Bayesian hyperparameter optimization framework.

**Evaluate the Optimized Model**

**Metrics Comparison**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision (Macro Avg)** | **Recall (Macro Avg)** | **F1-Score (Macro Avg)** | **Training Time** |
| **Baseline (SGD)** | 0.81 | 0.76 | 0.72 | 0.74 | 2 ms/step |
| **Adam Optimizer** | 0.8 | 0.75 | 0.71 | 0.73 | 3 ms/step |
| **RMSprop Optimizer** | 0.8 | 0.75 | 0.69 | 0.71 | 3 ms/step |
| **Optimized (Optuna)** | 0.8 | 0.75 | 0.72 | 0.73 | 3 ms/step |

Baseline SGD was found to be the best among the optimizers, with an accuracy score of 0.81 and a macro F1-Score of 0.74. Adam and Optuna models achieve similar results, with both having an F1-Score of 0.73. RMSprop has a lower Recall (0.69) and F1-Score (0.71) when compared to Adam. Training each step of an SGD algorithm takes 2 ms, a bit faster than the other algorithms’ 3 ms.



**Report Your Findings**

• SGD got the best results with an accuracy of 0.81 and a macro-average F1-score of 0.74, meaning it did better than the other optimizers used in this experiment.

• Adam and Optuna gave pretty much the same F1-score of 0.73, but weren’t quite as good when it came to getting the right positive answers and accuracy.

• RMSprop didn't work as well as the other methods, mainly because it got lower performance in recall (0.69) and the macro-average F1-score (0.71).

• Training the model on each step was faster with SGD at only 2 milliseconds, while the other methods took about 3 milliseconds.

**Recommendations**

1. Stick with SGD as the main optimizer for the ANN segmentation model because it outperforms other methods and does so much faster.

2. Both Adam and Optuna may be useful when dealing with complex or large data where faster generalization can come from adaptive learning rates.

3. Solve the imbalance by applying oversampling or class weighting, in order to increase performance for the minority class (Class 1).

4. Add Optuna or other similar frameworks in the future to fine-tune and examine if the model can be improved.

5. Check the performance of your model frequently to make sure it stays up to date with new marketing information.

**Future Work**

1. Advanced Class Imbalance Techniques

o Use a method like SMOTE, ADASYN, or cost-sensitive learning to help lessen the problem of Class 1 not working well.

2. Explore Ensemble Models

o Combine ANN with methods like Bagging or Boosting so the system can handle more kinds of data and do better when presented with real-world problems.

3. Deep Architecture Exploration

o Try using larger and more complex neural networks, like convolutional or recurrent neural networks, if you have more complex or sequential data you want to work with.

4. Optimization Algorithm Extensions

o Look at other popular optimizers such as Lookahead, AdaBelief, and Ranger to see if they might help the model converge quicker.

5. Deployment & Monitoring

o Put the improved model into use as part of your real-time marketing segmentation process, and make sure to run A/B tests to check how well the campaigns are doing.

**References**

Kingma, D.P. and Ba, J.L., 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*. Available at: <https://arxiv.org/abs/1412.6980>.

Tieleman, T. and Hinton, G., 2012. Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*. Available at: <https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>.

Bergstra, J., Yamins, D. and Cox, D.D., 2013. Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms. *Proceedings of the 12th Python in Science Conference*, pp.13-20. Available at: <https://proceedings.neurips.cc/paper_files/paper/2013/file/4f8e6432caee5fe2e60a35829a4f4d5a-Paper.pdf>.

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M., 2019. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*, pp.2623–2631. Available at: <https://doi.org/10.1145/3292500.3330701>.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp.321-357. Available at: <https://jair.org/index.php/jair/article/view/10302>.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. Cambridge: MIT Press.

Brownlee, J., 2019. How to Configure the Learning Rate When Training Deep Learning Neural Networks. *Machine Learning Mastery*. Available at: <https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks>

Sun, Y., Wong, A.K.C. and Kamel, M.S., 2009. Classification of Imbalanced Data: A Review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(4), pp.687–719. Available at: <https://doi.org/10.1142/S0218001409007326>.