Stage-wise Non-uniform Regularization for Cascade Configuration

Anish Yatin Pimpley
CICS
University of Massachusetts Amherst
apimpley@cs.umass.edu

Guided by : Benjamin Marlin
CICS
University of Massachusetts Amherst
marlin@cs.umass.edu

Abstract

In this study, we propose a new approach to building cascaded classifiers in computation sensitive, low power environments. We extend the firm cascade architecture by introducing non-uniform stage-wise regression terms into the loss function. We introduce a modification over the L1 regularizer, tuned to encourage sp arse connections. We evaluate the effects of such a regularizer on use cases with imbalanced classes such as human activity recognition and smoking detection. We discuss the use of regularization strength as a heuristic for a bottom up approach to cascade configuration.

1 Introduction

In the past few years, we have seen the rise of embedded wearable devices for widespread use in real-time information tracking. Typically we receive information on a continuous basis, and the specific events that they track form only a minuscule part of the collected data. This leads to highly imbalanced datasets. Information tends to be relayed to the central device through multiple sources where it is then combined into a cohesive dataset. Running traditional predictors requires processing on a remote machine, forcing passive learning. Embedded devices have battery and computation restrictions, that only allow the use of relatively simple learning models if active learning is required.
Cascade classifiers emerge as a solution for the above problem. The stage based approach to learning lets us use appropriate models in weaker devices, and minimize cumulative computation and communication costs by dropping negative samples early in the prediction process. Recent progress in parallel processing in computational devices opens the possibility of developing cascade architectures for machine learning models well suited for parallel processing.
The motivation behind this study, is to identify a suitable approach to configuring model complexity for individual stages. We explore stage-wise non-uniform regularization as a method for identifying optimal per-stage complexity and number of stages in a cascade. We also propose a modification on the standard L1 regularizer, to encourage sparse connections in individual stages.

2 Related work

Linear Cascade models include a chain of models that predict each data point sequentially in order to classify a point. The models are grouped into stages, where the access to a stage in the model is gated by the output of the stage preceding it. In the case of binary classification problem, only positive samples are passed to deeper stages and prediction is stopped once a stage classifies a point as negative. For highly class imbalanced datasets, cascade classifiers provide significant

computational savings.
Cascade classifiers were popularized by Viola-Jones[1] as a model for face detection. Here, each stage in the cascade was an adaboost model. Earlier stages were ensembles of weak decision stumps built from the most promising features. Deeper stages added features to the previous stage in an hierarchical manner to the individual learners in the ensemble. Each stage is optimized for minimum false negatives. This allowed the cascade to achieve real time detection of faces in images. However, the Viola-Jones paper falls short in certain areas.
The approach mentioned in the Viola-Jones paper cannot be extended to other model beyond boosting. Stages are individual trained, and a joint optimum is never achieved. The configuration of the cascade, especially parameters such as the number of stages and number of boosting rounds per stage are not addressed. Subsequent papers have tried to address these aforementioned issues.
Saberian et al.[2] approach the cost-accuracy tradeoff by modeling the Lagrangian risk as the loss function. They introduce the concept of neutral predictors, using which they were able to identify the optimal configuration of the casacade. Their algorithm FCBoost, does like all others preceding it, model the stages as a boosted ensemble of decision stumps.
The noisy-AND approach was proposed as an alternative for jointly training all stages in the cascade.The output probability of the cascade until a certain stage was considered equivalent to the product of probabilities of the concerned stage and all stages preceding it.The soft cascade architecture treats each stage as a black box, thus can be utilized for any model.
Raykar et al.[3] proposed the soft cascade architecture for jointly optimizing all stages in the cascade. They modify the noisy-AND by providing a knob for controlling the cost-accuracy trade off. This is achieved by the introduction of a term in the cost function, that models the computational cost of the cascade.
The soft cascade operates on hard binary decisions in prediction, despite being trained for products of stage probabilities which are continuous outputs. DadKhahi and Marlin[4] build on the Noisy-AND by modifying the stage combination and regularization terms to better suit the training objective. The output of each stage is made to operate in hard decision mode by applying a normalized logistic nonlinearity to its output. Cost in accounted for by using a per instance regularizer, which penalizes for deeper traversal of a data point into the cascade.. Furthermore, they generalize the firm cascade to a tree structured network, which is well suited for the wearable like arrangements.
The cascade configuration problem has yet to addressed for both the soft and firm cascades. We explore modifications in the firm cascade, in order to make them well suited for cascade configuration.

3 Data sets

We utilized 2 datasets for training and experimentation. Namely, the Human activity recognition (HAR) dataset and puffMarker, a smoking detection dataset.
The puffMarker[5] training dataset has 3836 data points and 37 features. 291 samples are positive, giving us a class imbalance of 7.5% samples belonging to the positive class.
HAR[6] is much larger with 10299 data points and 561 features. Of the 10299, 1544 samples are positive giving us a class imbalance of 15% samples belonging to the positive class.

4 Methodology and Modeling

In this section we expand on the structure of the cascade, the optimization procedure and how the results be used a heuristic for configuring a cascade.

4.1 The Firm Cascade Model and Depth Based Penalization

We chose to build our cascade on top of the aforementioned firm cascade architecture. The following paragraph briefly expands on the approach proposed in its paper.
The firm cascade acts as a gating mechanism in the probability function, such that the cascade's prediction is roughly equal to the output probability of the stage at which a data point is dropped. For positive cases we expect the cascade's prediction to be equal to the output probability of the final

stage. This mechanism is modeled by projecting the probability outputs onto a sigmoid with a steep slope, casting the probabilistic output into hard binary decisions, while retaining differentiability. We then take a sum over all the probability values of all layers with each term being multiplied with a coefficient equivalent to the product of sigmoid transformed probabilities of all preceding layers.
In this case we utilize the representation of the firm cascade for a single device and a linear cascade.The single device linear firm cascade is mathematically represented as follows:

$$P_*(y/x) = \sum_{i=1}^{L} \theta_l * p_l$$

$$\theta_l = \begin{cases} (1 - g_\alpha(p_l)) \prod_{k=1}^{l-1} g_\alpha(p_k), & l < L \\ \prod_{k=1}^{L-1} g_\alpha(p_k), & l = L \end{cases}$$

$$g_\alpha(p) = \frac{1}{(1 + exp(-\alpha(p - 0.5))}$$

P_* is the predicted output class of the cascade. Given the form of the equation ,we expect the value of P_* to lie extremely close to either 1 or 0. In the above case the sigmoid transformation is centered at 0.5.
The g function represents the sigmoid transformation at the output of a layer and θ_l is the aforementioned coefficient for probability of layer 'l'.
The loss function discussed in the original firm cascade paper, combines the cross-entropy for P_* with a regularizer that penalizes based on the degree of propagation of a point into the cascade. The degree of penalization is controlled by the hyper parameter λ. It was modeled as follows:

$$Loss = \sum_{n=1}^{N} CrossEntropy(P_*(y_n|x), y_n) + \lambda * regP(y_n, x_n)$$

$$reg = k1 + \sum_{l=2}^{L} k_l \prod_{k=2}^{l-1} g_\alpha(P_k(y|x))$$

For layer 'l', The regularizer 'regP' outputs a value approximately equal to the sum of coefficients 'k' till the stage at which it is dropped.

4.2 Stage Model Selection

The firm cascade is a black box approach that allows us to plug in any model into any stage of the cascade. However, varying models show varying suitability to a cascade structure. Empirically, we desire for the complexity of stages to increase as we go deeper into the cascade. For real time analysis, We also desire for the model to be fast and computationally cheap at prediction time. In a variety of models, feature selection is often the only tool for controlling model complexity and computational cost. Statistical methods for feature selection do not take into consideration complex relationships between features and have well known short comings. Boosted classifiers have been widely used as a base model for cascades, however they too depend on utilizing highly ranked features for constructing the cascade, which acts as a form of soft feature selection.
Given our requirement for flexibility in model complexity without soft or explicit feature selection, the Multilayer Perceptron serves as an ideal model for this purpose. The multitude of hyper parameters in a neural network allow us to vary the model's complexity with ease.
Neural network models are also well suited to exploit the ever rising number of parallel processing units in embedded devices and are fast at prediction time.
For preliminary experimentation, we have restricted ourselves to single hidden layer neural network models. In previous experimentation on the the given datasets, we did not observe an advantage in using a deeper networks. Rather, a drop in performance was observed. It is also in our interests to observe the behavior of the cascade for simpler models, making model analysis convenient. Here

on forth, every stage is assumed to be a single hidden layer neural network model. Furthermore, every stage will be using all features at its disposal.

4.3 Stage-wise modified L1 Regularization

The firm cascade model discussed in sec.4.1 provides no means for the cascade to control the per-stage complexity. For that reason, the stage and cascade need to be hand configured. This pushes us towards searching over a unreasonably large hyper parameter space, especially if we have no heuristic for pruning a significant part of the search space.
For 1 hidden layer neural networks, the number of hidden nodes and by extension total number of connections are positively correlated with the capacity of the model. A taller network is capable of forming more complex decision boundaries. The same nodes and connections are also positively correlated with the number of computations and by extension energy cost at prediction time.
Thus, an accuracy vs computational cost tradeoff can be achieved by controlling the number of connections or sparsity of a network. This can be achieved by introducing a term in the cost function that approximately models the number of connections in a network. L1 regularization is known to force the weights of a network towards zero. However, a small non-zero weight term still incurs the same computational cost as a large weight. For that reason a stock L1 regularizer fails to appropriately model the number of connections.
We modify the L1 regularizer, by passing the every weight term through a sigmoid transformation centered at value close to zero. This forces non-zero weights larger than the threshold to a common value of 1 and all values close to zero are hard set to zero. This also renders the contribution of all non-zero weights as equal, which corresponds directly to their contribution in computational cost. The sigmoid transformation allows us to retain differentiability for back propagation. We then take the L1 term of these sigmoid transformed weights. It is modeled as follows:

$$reg(i) = regStr(i) * (expL1(Whidden) + expl1(Woutput))$$

$$exp1 = \sum_{w \in W} sig(absolute(W))$$

$$sig = 1/(1 - e^{-\alpha*(x-threshold)})$$

The output of the regularizer is directly proportional to the number of non-zero weights in the network. After training, we prune all connections with weights below the threshold, thus obtaining a sparse network, with the regularization strength (regSt(i)) acting as a knob for controlling degree of sparsity for the i^{th} stage. When combining the reg(i) terms for every stage, we multiply each term with a propagation coefficient. This coefficient is an approximately binary measure of whether or not the point has propagated to the stage (i). The combined regularizer (regAll) and final loss function are represented as follows:

$$regAll = \sum_{i=1}^{L}(reg(i) * \prod_{j=1}^{i-1} g(p_j)$$

$$Loss = \sum_{n=1}^{N} CrossEntropy(P_*(y_n|x), y_n) + \lambda * regP(y_n, x_n) + regAll$$

for L stages, L different regularization strengths ($regStr$) need to be input by the user. We desire for earlier stages to be less complex and thus more sparse. Thus the $regStr$ decreases as we go deeper into the network. We propose using a value linearly proportional to the negative exponent of the stage number (i) represented as $10^{-(m*i+c)}$. In such a scenario, we can obtain the $regStr$ values for all stages by knowing slope (m) and offset (c) of line.
The model is trained in the backwards direction, with the the deepest model being pre-trained first, and preceding stages then being jointly trained one set at a time until we reach the first stage. We have also replaced the relu activation function with softmax, because of exponential blowup of weights during training. The framework is built on top of theano with lasagna. RMSprop is used for updating weights.

5 Experiments and Results

In this section we compare our results with identically structured stock firm cascade and a baseline 1 hidden layer neural network. Performance is evaluated based on Accuracy, F1 score, run time and cascade complexity. We report results for 2 stage cascades with each stage having 10 hidden nodes in each layer. The networks are fully connected at train time.
puffMarker train-test split : 3400 - 436 (31 positive)
HAR train test-split : 7352 - 2947 (496 positive)
Identifying a appropriate hyper parameters for all of the models, namely stock firm cascade and our modification of the firm cascade proved to be difficult. The cost function turned out to be very sensitive to the regularization strengths and learning rates. We compared results for the following models and the results are shown below:

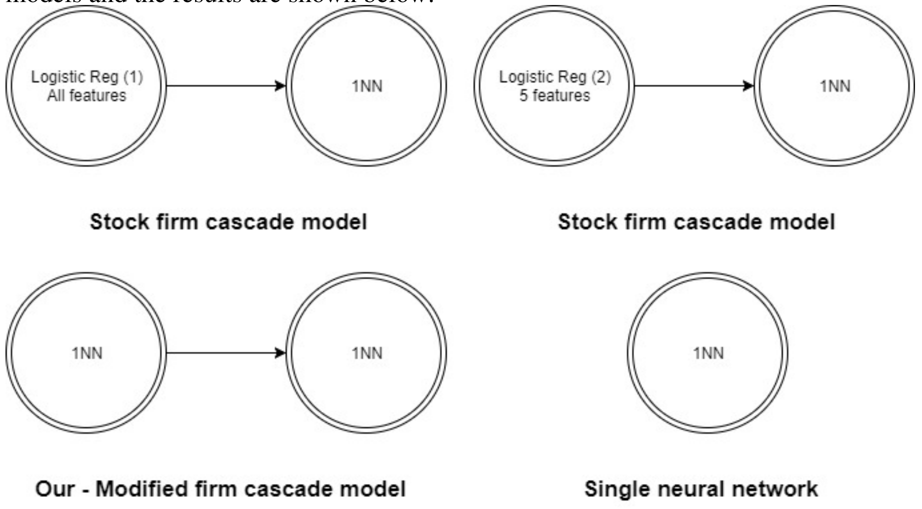

Figure 1: models tested

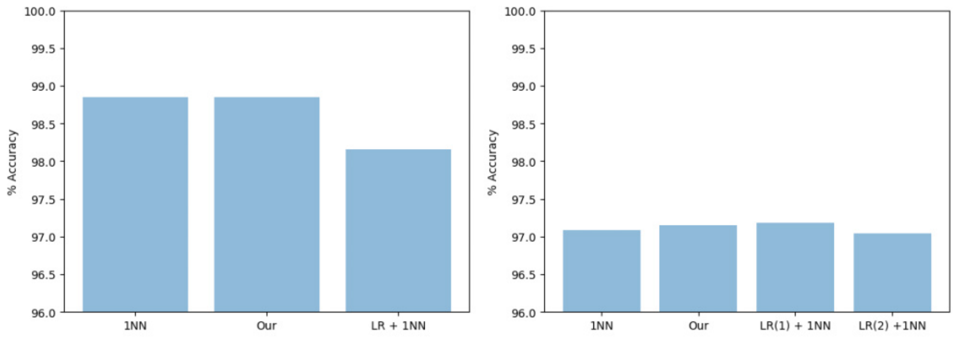

Figure 2: accuracy : puffMarker(left), HAR(right)

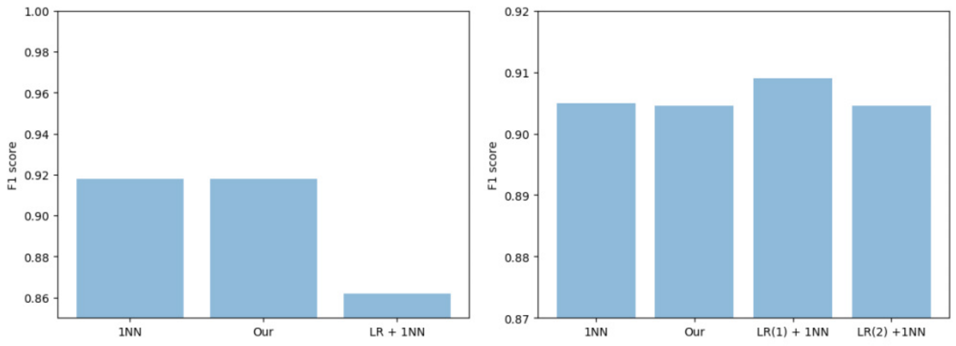

Figure 3: F1 : puffMarker(left), HAR(right)

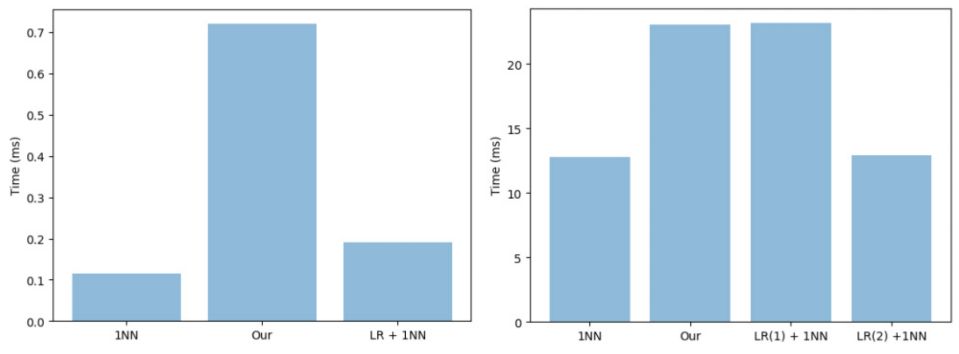

Figure 4: Time : puffMarker(left), HAR(right)

We pruned all connections with values less than 10% of the 90th percentile of weights. Using this heuristic the model pruned 30% and 17% connections in the 2 stages on the puffmarker dataset. On the HAR dataset, the 2 stages were pruned by 38% and 20% respectively.
While we do see a rise in runtime, it is not clear if our theano implementation leverages the sparse connections for faster predictions. Accuracy varies within range of the best results obtained previously for the firm cascade. The model beats the baseline classifier accuracy in both cases.
The regularization strength for stages for both datasets was fixed at $[10^{-2}, 10^{-7}]$. The value of λ for the propagation regularizer was set to 0.1 . The value of alpha for all exponents was fixed at 16.

6 Discussion and conclusions

In this project, we proposed that a modified L1 regularizer can be used to decide the complexity of stages in a cascade. We were able to successfully demonstrate the same for a 2 stage cascade. The results matched or exceeded the models that we compared to, on most metrics except time. The modified loss function converged successfully despite the steep sigmoidal transforms imposed on the L1 regularizer.

6.1 Future work:

An immediate goal would be to validate the performance of the the model is less controlled environments. It is vital to test how the proposal generalizes for longer cascades, starting with a 3 stage model. The model also needs to be tested over a larger range of hyper parameters.
L1 regularization is one of the simplest methods for forcing sparsity, and is outdone by many better pruning algorithms. It would be interesting to see the effect of a similar transform on other regularization based techniques.An iterative approach to pruning, may hold potential for identifying a more nuanced threshold for pruning connections.
In the near future, a generalized cascade definition with neural networks and stages with multiple hidden layers can be targeted as tangible goals.

References

[1] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[2] Mohammad Saberian and Nuno Vasconcelos. Boosting algorithms for detector cascade learning. *Journal of Machine Learning Research*, 15:2569–2605, 2014.

[3] Vikas C Raykar, Balaji Krishnapuram, and Shipeng Yu. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 853–860. ACM, 2010.

[4] Hamid Dadkhahi and Benjamin M Marlin. Learning tree-structured detection cascades for heterogeneous networks of embedded devices. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1773–1781. ACM, 2017.

[5] Nazir Saleheen, Amin Ahsan Ali, Syed Monowar Hossain, Hillol Sarker, Soujanya Chatterjee, Benjamin Marlin, Emre Ertin, Mustafa Al'Absi, and Santosh Kumar. puffmarker: a multi-sensor approach for pinpointing the timing of first lapse in smoking cessation. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 999–1010. ACM, 2015.

[6] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.