Passing Intermediate Feature Representation Between Cascade Stages

Ajinkya Indulkar
CICS
University of Massachusetts Amherst
Amherst, MA 01003
aindulkar@cs.umass.edu

Benjamin M. Marlin
CICS
University of Massachusetts Amherst
Amherst, MA 01003
marlin@cs.umass.edu

Abstract

In this project, we propose a new approach to passing feature representations from one stage of a cascade to another. We build upon existing cascade literature and explore the idea of what data should be passed between stages. Different stages in the cascade have access to different features and passing all data to the following stage is not always the best option. In this project, we jointly optimize the entire cascade by using intermediate layer outputs in the preceding stage as new features in the following stage. This work is motivated by research being performed in the area of mHealth or mobile health where we use on-body sensors and other devices to collect data and detect patterns. We also assume the availability of a smart phone where the data from sensors is collated to perform a more complex analysis. And additional reasoning behind this idea is that preceding stages are already performing computations to learn these feature representations that may be important for the following stages. Furthermore, the cost of transmission is greatly reduced by using a compact feature representation instead of the full feature set. This work aims to reduce the transmission cost and the energy used in computation for just-in-time adaptive interventions. We evaluate our feature representation on sensor-based human activity recognition and mobile health detector learning problems.

1 Introduction

Mobile health or mHealth, as it is popularly known as, aims to use recent advances in wearable sensor technology to develop systems that can monitor a person's health and deliver just-in-time adaptive intervention (JITAI)[1]. Typically, we have one or more on-body sensors that are connected to a user's smart phone. Such a system can then be used to deliver JITAI. It can be used for a variety of applications including smoking cessation[2], drug control[3], stress control[4] etc. But JITAI can only be used if the system is capable of real-time feedback. For instance, in case of smoking cessation, the first lapse is much more likely to lead to a full relapse[5]. Therefore, it is important to perform real-time intervention. However, the current state-of-the-art solutions only do passive data collection followed by offline detection on a separate device. The reason for this is that the wearable sensors don't have ample resources to perform any kind of meaningful analysis and doing so could even result in poor accuracy and recall because of the limited complexity of the model. Using more complex models will not provide real-time results and might also result in high power consumption. While such delayed results could act as a stepping stone towards a bigger goal, it is not suitable for JITAI. To that end, we develop an intermediate feature representation that can replace the actual features that are sent to the next stage in a cascade.

Our setup will be made up of heterogeneous on-body sensors and a smart-phone where each device will have access to a separate feature set. Each node will also have different computing capabilities

and will therefore have models of differing complexity. We will then jointly optimize the cascade with this new feature representation. During optimization we also try different regularization values to obtain the best possible performance for a given cascade. During prediction we operate the trained cascade in a hard setting to get early drop-off of negative cases. In our experiments, we compare the performance our new feature representation with that of actual features that have been used traditionally.

2 Related work

A cascade classifier is a collection of classifiers that collectively perform classification on data. They are typically used for highly class-imbalanced datasets. A point in the data passes through these classifiers in the cascade and it is classified as positive only if all the stages of the cascade classify as positive.

This idea of a cascade was made popular by Viola and Jones[6]. Viola and Jones introduced the concept of increasingly complex boosted cascades that quickly classify background regions that are unlikely to contain faces as negative. Thus the more complex later stages in the cascade only receive very few points that likely contain faces. Since the point is classified as negative if any of the stages predicts the point as zero, the threshold of the classifiers can be adjusted to bring the number of "false negatives" close to 0. The problem with Viola-Jones' paper was that they did not outline a technique to specify the cascade configuration i.e. the number of stages in a cascade and the number of boosting stages within a stage. They reported results with five cascade stages with 1, 10, 25, 25 and 50 features respectively in each stage. Although the results were better than state-of-the-art, there was no well defined way to set the number of stages and the number of boosting stages within a cascade stage.

Raykar *et al.*[7] took this further and introduced a generalized approach to jointly optimize a cascade. They introduced the concept of a soft cascade that allowed the joint training of the entire cascade. However, during prediction, the cascade was operated in a hard-decision mode. This mismatch between the soft-mode during optimization and hard-mode during prediction is unnatural. They also introduced a computational cost parameter that allowed one to trade-off between accuracy and cost. This is important as typically for low-power and low-capability devices, we require low cost and for high-power devices we require high accuracy. Raykar *et al.*[7] assumed that the cascade is ordered according to the feature acquisition cost i.e. the first stage will have the least cost of obtaining features whereas the features used in later stages will have progressively higher acquisition costs. While this may be true for some cases, our system will not consider feature acquisition cost as we assume features or subset of features are available at a given node. Instead there will be an associated informal feature transmission cost that we aim to minimize.

Saberian *et al.*[8] introduce a new algorithm called Fast Cascade Boosting (FCBoost) that uses a Lagrangian risk that jointly accounts for accuracy and speed. It is built on the idea of AdaBoost and is in fact a generalization of AdaBoost. It allows us to vary the trade-off between accuracy and speed of classification. It outperformed some state-of-the-art classifiers at the time. However, it made an assumption that all features are available at all stages in the cascade and an implicit feature selection is performed by using decision stumps as the underlying weak learners in the boosting stages.

Dadkhani *et al.*[9] presented a new method of learning cascade classifiers that are well modeled by a network of classifiers. They generalized linear cascades to tree-structured cascades which model a smart-phone as a parent node and other on-body sensors as child nodes. They showed a way to perform joint training on the combined cascade with regularization parameters that constraints the propagation of a point down the cascade i.e. it forces the classifier to classify the point as negative early in the cascade. This not only reduces the computational cost required to classify a point but also results in reduced power consumption and transmission costs. Furthermore it is assumed that stages contain progressively complex classifiers that classify difficult points. This paper brings JITAI closer to reality as it works towards real-time monitoring. They introduce a gating function during training that more correctly models the hard decisions made by the cascade during prediction. More importantly, they acknowledge the problem of data locality that other works had not yet addressed. Data locality refers to the idea that different stages of a cascade could live on different heterogeneous devices with limited communication capabilities and therefore some stages may only have access to limited feature sets. For example, if a system contains two wrist sensors

and a respiratory sensor connected to a smart-phone, the wrist sensors have access to data from the accelerometer and gyroscope of the device. The wrist sensors don't have automatic access to each others' data. Furthermore, the respiratory sensor only has access to respiratory data and not the data from wrist sensors. While in theory they can send data to each other, these devices often have limited battery and communication capabilities which limit the amount of data that can be sent to other devices. They were able to achieve significant improvements in accuracy, F1-score and computational time. One limitation was that they used wall-clock time to model the cost. This is often very inaccurate because of the low-level linear algebra routines used by python computing libraries.

3 Data sets

We use two mobile health datasets in our project. Data for both datasets have been collected from test subjects. The first is *Human Activity Recognition Using Smartphones Dataset* [10]. It has 561 features that are collected from two sensors: an accelerometer and a gyroscope. It is labeled with various activities such as Walkling, Sitting etc. But for our purposes we treat `WALKING_UPSTAIRS` as a positive activity and all other activities as negative.

The second dataset that we use is the puffMarker dataset[2]. It is a smoking dataset that contains data from human volunteers who wore two wrist sensors and a respiratory chest sensor. Similar to the first dataset, this dataset is highly class imbalanced.

4 Methodology

For this project, we primarily use datasets that are highly class-imbalanced as they correctly model data for JITAI applications. We use the Firm Cascade Framework introduced by Dadkhani *et al.*[9]. Specifically we look at a linear firm cascade with two stages of increasing complexity.

We assume a linear cascade model consisting of L stages. Each stage is a probabilistic classifier $P_l(y|\mathbf{x})$ where each $\mathbf{x}$ is a point in the dataset and the output is binary $y \in \{0, 1\}$. We denote the output of the entire cascade by $P_*(y|\mathbf{x})$. We use the shorthand $p_l = P_l(y|\mathbf{x})$. The way this cascade works is a data case $\mathbf{x}$ is classified positive only if it is classified by all stages as positive. A stage calls a point positive if $p_l > 0.5$. We use a gating function g_α that puts a weight on the output of each stage. It acts as an extremization function that sets probabilities less than 0.5, very close to 0 and probabilities larger than 0.5 very close to 1. So the output of the cascade is either the output of the first stage in the classifier that classifies it as negative or the output of the last stage of the cascade. Equation 1 to 4 describe this logic. Exact details of the working can be found in Dadkhani *et al.*[9].

$$P_*(y|\mathbf{x}) = \sum_{l=1} \theta_l \cdot p_l \tag{1}$$

$$\theta_l = \begin{cases} (1 - g_\alpha(p_l)) \prod_{k=1}^{l-1} g_\alpha(p_k) & l < L \\ \prod_{k=1}^{L-1} g_\alpha(p_k) & l = L \end{cases} \tag{2}$$

$$f_\alpha(p) = \frac{1}{1 + \exp(-\alpha(p - 0.5))} \tag{3}$$

$$g_\alpha(p) = \frac{f_\alpha(p) - f_\alpha(0)}{f_\alpha(1) - f_\alpha(0)} \tag{4}$$

The model is trained by maximizing the log likelihood of $P_*(y|\mathbf{x})$. For our dataset $\mathcal{D} = \{(y_n, \mathbf{x}_n)|1 \leq n \leq N\}$, we minimize the loss given by equation 5 or more precisely the cross-entropy loss in equation 6 The regularizer r applies a penalty proportional to the number of stages used in the cascade. It applies a fixed penalty for the first stage and adds a per-stage cost to each subsequent cascade actually used. We use RMSProp to minimize the loss given by equation 5 and the weights as parameters. More details for the training procedure can be found in Dadkhani *et al.*[9].

$$\mathcal{L}(\mathcal{D}) = \sum_{n=1}^{N} \ell(y_n, \mathbf{x_n}) - \lambda r(y_n, \mathbf{x_n}) \tag{5}$$

$$\ell(y, \mathbf{x}) = y \log P_*(y|\mathbf{x}) + (1-y) \log(1 - P_*(y|\mathbf{x})) \tag{6}$$

$$r(y, \mathbf{x}) = \kappa_1 + \sum_{l=2}^{L} \kappa_l \prod_{k=1}^{(l-1)} g_\alpha(P_k(y|\mathbf{x}) \tag{7}$$

Figure 1 shows the three configurations that we consider in our experiments. *C1* is a single stage model with a 2-hidden-layer neural network (2LNN). It has access to all features. *C2* will act as a baseline against which we will compare our later experiments. *C2* is a two stage cascade with the first stage being a 1-hidden-layer neural network and the second stage being a 2-hidden-layer neural network. The first stage only has access to a subset **F1** of all features **D**. Similarly the second stage has access to a subset **F2** of all features **D**. Therefore:

$$F1 \cup F2 = D \tag{8}$$

and

$$F1 \cap F2 = \emptyset \tag{9}$$

We initialize parameter values by training in reverse order from L down to 1. So, in case of *C2* we

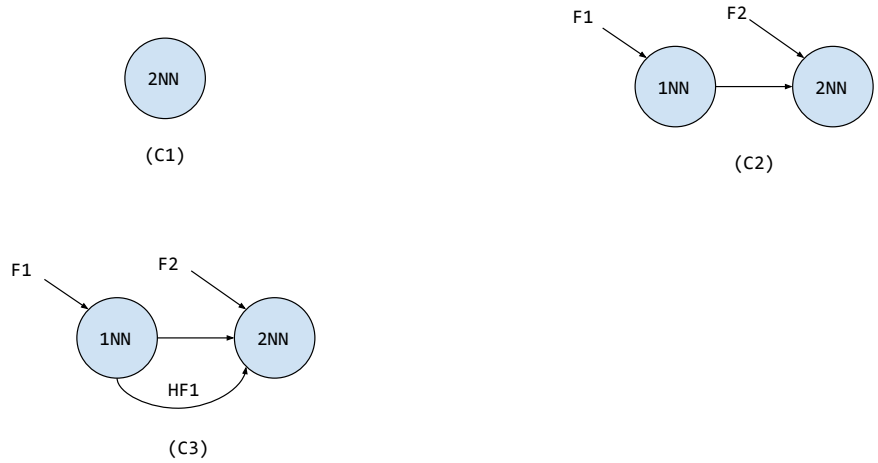

Figure 1: This figure shows the different cascade architectures used in evaluation. C1 is a single-stage model with a 2-hidden-layer neural network. C2 and C3 are two stage cascades with the first stage as a 1-hidden-layer neural network and the second stage as a 2-hidden-layer neural network. The figure shows which feature sets are available to the two stages.

train the 2-hidden-layer neural network first and use the trained weights to initialize the weights for the final joint training. We then train the entire cascade jointly. *C3* is similar to *C2*, in that it consists of two stages with the first stage being a 1-hidden-layer neural network and the second stage being a 2-hidden-layer neural network. Similar to *C2*, the first stage has access to **F1** feature set. The second stage has access to **F2** and in addition to those, it has access to feature set **HF1**. **HF1** is the output of the hidden layers in the first stage. The reasoning behind this is that the first stage already computed a feature representation of the features available in the first stage **F1**. We train *C3* in three phases. The first phase is a simple pretraining of the second stage. The second phase is joint training of the first and second stage. But we don't use **HF1** features for this stage since we don't have access to **HF1**. The third phase is joint training of the first stage and second stage after introducing **HF1**.

4.1 Model Selection

We selected neural networks as a base classifier for our model as it has an intermediate data representation i.e. hidden layer outputs that can be transmitted to the next stage in the cascade. This would not have been possible by using a classifier such as Logistic Regression.

4.2 Hyper parameter selection and generalization error

We used a standard train-test split to calculate generalization error. For the HAR dataset the train-test split was 70%-30%. The data was inherently stratified since the test subjects for the two splits were different. For the PuffMarker dataset, the train-test split was 88%-12%. We conducted grid search over a linear range to identify optimal hyper parameter λ for our models.

4.3 Platform and libraries

The project was build with Python 2.7. We used theano and lasagne for training our model with RMSProp. Numpy was used for handling data. Some other helper libraries were used and can be found in the `environment.yml` file. All experiments were run on standard personal laptop setup with Intel(R) Core i5-7300HQ CPU @ 2.50GHz. The experiments for the PuffMarker dataset were performed on a cluster with a Intel(R) Xeon(R) CPU E5-2440 0 @ 2.40GHz CPU.

5 Experiments and Results

5.1 Human Activity Recognition dataset

The first dataset that we perform our experiments on is the Human Activity Recognition dataset as described in the previous sections. It consists of 561 features out of which 345 are Accelerometer features and the remaining 216 are Gyroscope features. We don't perform any explicit feature selection but an implicit feature selection is performed by feature availability on devices i.e. we assume that the first stage is the Accelerometer and the only features that are available to it are the Accelerometer features. The first stage is a 1-hidden-layer neural network with 10 hidden units. Similarly, we assume the second stage is the Gyroscope with access only to the Gyroscope features. The second stage is a 2-hidden-layer neural network with 10 and 20 hidden units in the first and second layer respectively. Swapping the feature sets in the two stages i.e. using Gyroscope features in the first stage and Accelerometer features in the second stage also corroborate our results. Different number of hidden units were tried but these values resulted in better performance. We set the value of α to 16 as it resulted in better performance.

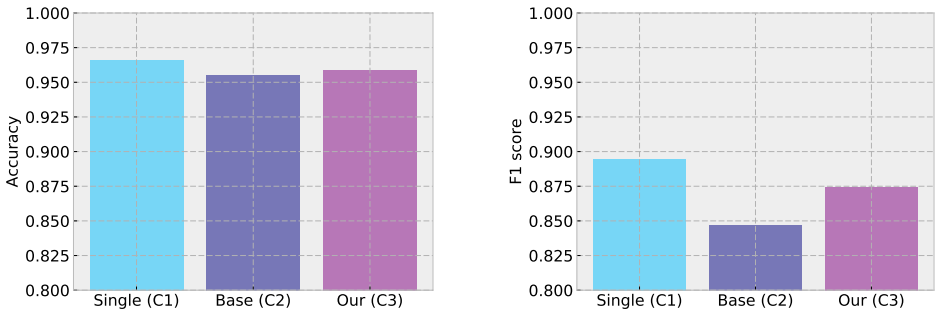

Figure 2: HAR accuracy and F1 score

The results are shown in Figure 2. It shows the maximum Accuracy and F1 score obtained after a hyperparameter grid search over the regularizer strength λ values. We can see that the single stage model *C1* outperforms our cascade models *C2* and *C3* by a small margin. Therefore it trades-off accuracy for computation complexity. It was observed that reducing the regularization strength did improve the overall accuracy but it greatly reduced the precision of the first stage. Therefore those results were excluded from the report as they offer no value to our model. However, for comparison,

our model *C3* outperformed the base model *C2*. Out of the 2947 samples, the first stage of our cascade *C3* was able to classify 2400 negative examples leaving only 547 samples for the second stage.

5.2 puffMarker dataset

The second dataset that we perform our experiments on is the puffMarker dataset as described in the previous sections. It consists of 37 features out of which 19 are respiratory features from a chest band worn by test subjects. Similar to the last dataset, we don't perform any explicit feature selection but an implicit feature selection is performed by feature availability on devices. We use the respiratory features for the first stage and all the remaining features for the second stage. The first stage is a 1-hidden-layer neural network with 10 hidden units and the second stage is a 2-hidden-layer neural network with 10 and 20 hidden units in the first and second layer respectively. Different number of hidden units were tried but these values resulted in better performance. We set the value of α to 16 as it resulted in better performance.

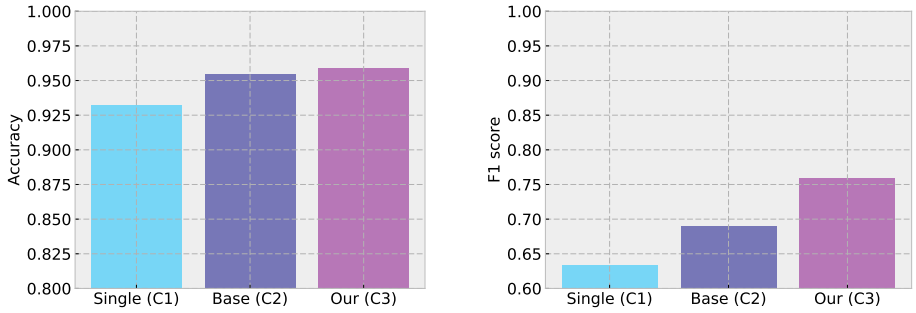

Figure 3: PuffMarker accuracy and F1 score

The results are shown in Figure 3. We calculate the maximum accuracy and F1 score obtained after a hyperparameter grid search. Both cascade models *C2* and *C3* outperform a single stage model *C1* in accuracy and F1 score. For *C3*, out of the 436 data points, the first stage was able to predict 389 negatives.

6 Discussion and conclusions

In this project, we proposed a novel approach to passing feature representations from one stage of a cascade to another. We built upon existing cascade literature and explored the idea of what data should be passed between stages. Using neural networks as base predictors for a stage in the cascade, we found that when hidden layer outputs from a stage are passed to the following stage, the performance of our overall cascade increases. This feature representation is highly compact when compared to the original feature set used in the preceding stage. In some cases our cascade performed better than a single stage cascade. We were able to achieve better accuracy and F1 score for both data sets using our new feature representation. This will act as a stepping stone for achieving real-time performance JITAI systems and will bring them closer to reality.

For future work, we plan to develop a cost model that quantifies the performance gains resulting from our new feature representation. In this project, we assumed that using a cascade instead of a typical single stage will result in speedier predictions since that is the basic premise in a cascade architecture. But it should be quantified by counting the elementary operations and calculating the running time of our cascade. Comparing the running times of our cascade to a typical single stage model should give us more accurate information. Furthermore, we plan to compare the performance of our feature representation to the actual features. In this project, we simply partition the features according to the availability on a node. It is possible that a node has access to different feature sets with each set having a different feature acquisition cost. In the future we plan to consider feature acquisition costs in our framework. Moreover, we used a simple two-stage cascade for our experiments. The actual cascade configuration was not addressed in this project. We should have a way to configure cascades automatically from our device structure and energy and computation capabilities. Finally, we assumed our stages to be separate devices with limited computation capabilities but we simulated it on a computer. In the future, we plan to deploy it on actual hardware and assess the performance.

References

[1] Inbal Nahum-Shani, Shawna N Smith, Ambuj Tewari, Katie Witkiewitz, Linda M Collins, Bonnie Spring, and S Murphy. Just in time adaptive interventions (jitais): An organizing framework for ongoing health behavior support. 2014.

[2] 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.

[3] Syed Monowar Hossain, Amin Ahsan Ali, Md Mahbubur Rahman, Emre Ertin, David Epstein, Ashley Kennedy, Kenzie Preston, Annie Umbricht, Yixin Chen, and Santosh Kumar. Identifying drug (cocaine) intake events from acute physiological response in the presence of free-living physical activity. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 71–82. IEEE Press, 2014.

[4] Kurt Plarre, Andrew Raij, Syed Monowar Hossain, Amin Ahsan Ali, Motohiro Nakajima, Mustafa Al'absi, Emre Ertin, Thomas Kamarck, Santosh Kumar, Marcia Scott, et al. Continuous inference of psychological stress from sensory measurements collected in the natural environment. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 97–108. IEEE, 2011.

[5] Saul Shiffman, Jean A Paty, Maryann Gnys, Jon A Kassel, and Mary Hickcox. First lapses to smoking: within-subjects analysis of real-time reports. *Journal of consulting and clinical psychology*, 64(2):366, 1996.

[6] 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.

[7] 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.

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

[9] 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.

[10] Jorge-Luis Reyes-Ortiz, Davide Anguita, Alessandro Ghio, and X Parra. Human activity recognition using smartphones data set. *UCI Machine Learning Repository*, 2013.