Team Interpretability and Community Modeling: Final Report CSE 481 N

Michael Zhang, Shobhit Hathi, and Yifan Xu

June 2019

Abstract

Deep models are generally thought of as black boxes that achieve state of the art results at the cost of transparency. Recent work has attempted to make deep models more transparent and interpretable by utilizing an instance learning paradigm, but is hampered by performance bottlenecks. We propose an extension to such a work that alleviates the performance bottleneck and makes the training more robust by replacing the softmax layer in traditional deep models with learnable prototypes.

1 Introduction

Deep classifiers can generally be decomposed into a feature extractor that distills a high-dimensional input into a low-dimensional representation, followed by a softmax layer that produces a multinomial distribution over the space of possible output classes from an instance's learned representation. Recent work from Card et al. [2019] proposed *deep weighted averaging classifiers* (DWAC) as a alternative to the softmax layer in deep classifiers. DWAC makes predictions by computing a weighted sum over all training instances based on a learned metric of similarity. Although using DWAC still doesn't allow us to understand *why* the model relates a pair of instances, we can observe *what* the model is learning to relate together and *how* it affects the model's predictions. In addition to this heightened transparency from DWAC, the authors of Card et al. [2019] also find that this transparency can be directly utilized in improving model robustness under the framework of *conformal methods*

A major drawback of DWAC is that predictions require comparisons against the entire training set. Card et al. [2019] propose an approximate objective that makes training tractable, but even with these changes the model's performance, with regards to both speed and accuracy, suffers on large datasets. Motivated by these drawbacks, we propose an alternative to DWAC that utilizes a small set of *prototypes* that the model computes a weighted average over to make a prediction. Functionally, theses prototypes replace the role of the training set in the original DWAC model at both training and test time. We find that our new *prototyped deep weighted averaging classifier* model (ProtoDWAC) maintains or improves on many of the same desireable characteristics as the original DWAC model (i.e. accuracy, robustness and transparency), while making training and testing significantly faster, more stable, and better suited for larger datasets, but sacrifices some level of interpretability.

2 Background: Deep Weighted Averaging Classifiers

Traditionally, deep classifiers produce output probabilities on classes by applying the softmax equation to h, the outputs of the final hidden layer, where $h = Wf(x) + b$. Here we make explicit the separation between the feature extractor portion of the deep model, $f(x)$, which yields a low-dimensional representation of the input and the softmax layer.

In contrast, DWAC models retain the same feature extractor $f(x)$ while replacing the affine transformation and softmax with the following:

$$P(y = k|x) = \frac{\sum_{(x_t,y_t)\in\mathcal{T}} \mathbb{I}(y_t = k)w(f(x_t), f(x))}{\sum_{(x_t,y_t)\in\mathcal{T}} w(f(x_t), f(x))} \tag{1}$$

where $w(a, a')$ is a function of similarity between the embeddings a and a'. In Card et al. [2019] and for the rest of this work, we will define w to be the gaussian kernel with standard deviation $\sigma = \frac{1}{2}$:

$$w(a, a') = \exp(-\|a - a'\|_2^2) \tag{2}$$

Because comparing each training example against the entire training set is a significant slow down (increasing the computational complexity of training by a factor of $\mathcal{O}(n)$, where n is the size of the training set), DWAC approximates $P(y = k|x)$ by comparing every training instance with the other instances in the same minibatch $\mathcal{B}$. In essence, we are replacing $\mathcal{T}$ in equation. 1 with $\mathcal{B}$, and x_t and y_t with x_b and y_b.

3 Prototyped Deep Weighted Averaging Classifiers

When using ProtoDWAC, we replace the weighted sum against the training set with a set of prototypes, $\mathcal{P}$, which are each associated with a given label. In practice, we associate a constant number of prototypes with each output class. Now we are replacing $\mathcal{T}$ in equation. 1 with $\mathcal{P}$, x_t and y_t with x_p and y_p.

3.1 Training and Time Complexity

In ProtoDWAC, we do not need to make the training time approximations as in the original DWAC model becuase the set of prototypes is vastly smaller than the training set. Additionally, ProtoDWAC avoids issues posed to the DWAC model with regards to class imbalances in the training set as prototypes can be assigned uniformly to each class, a guarantee that was not met in the original DWAC model.

At test time, predicting on a single instance takes $\mathcal{O}(|\mathcal{T}|z)$ where z is the dimensions of the hidden representation. In contrast softmax models and ProtoDWAC (for a constant number of prototypes per class) take $\mathcal{O}(zk)$ where k is the number of output classes. Using prototypes allows us to remove the primary bottleneck of the DWAC model: the linear scaling factor with respect to the size of the training set.

4 Conformal Methods

Conformal Methods refer to a broad range of methods that can be applied to any existing classification or regression system to provide theoretical guarantees on error rates. For our purposes we offer a condensed description of conformal methods that is tailored to our specific setting of probabilistic models for classification.[1] Conformal methods methods rely on a scalar metric of *nonconformity*, η, that can be computed on our trained model, $\mathcal{M}$, for each test instance, x, and output class, k, pair, i.e.

$$\eta(x, k) = A(\mathcal{M}, (x, k)) \tag{3}$$

This measure of nonconformity intuitively correlates to a measure of how *atypical* a test instance is of the given class. These measures of nonconformity can be defined many ways, but for probabilistic classification models, a natural measure of nonconformity is the inverse (multiplicative or additive) of the probability of a test instance being assigned the given class, i.e.

$$\eta(x, k) = -P_{\mathcal{M}}(y = k|x) \tag{4}$$

[1]For a more general overview, see Card et al. [2019]

Conformal methods work by comparing the nonconformity score of each possible prediction against those of each example inside a calibration set, $(x, y) \in \mathcal{C}$, that has been held out from training. We compute a p-value for each possible output label equal to the proportion of calibration instances with a higher nonconformity score. More precisely, our p-values for a test instance and hypothesized label pairing is computed as such:

$$p(x, k) = \frac{\sum_{(x_c, y_c) \in \mathcal{C}} \mathbb{I}(\eta(x_c, y_c) \geq \eta(x, k))}{|\mathcal{C}|} \tag{5}$$

Under this framework, we can provide our classifiers a desired error rate, ϵ, and allow our model to predict a possibly-empty set of labels $\{k \in \mathcal{Y} : p(x, k) > \epsilon\}$ for each test instance with guarantees that the predicted label set will contain the true label with error rate ϵ with high probability.

The authors of Saunders et al. [1999] proposed two metrics for evaluating classification models using conformal methods:

- **Confidence** is equal to the largest $1 - \epsilon$ such that the predicted label set contains exactly one label. This measure corresponds to the probability, according to the model, that the predicted label is correct.
- **Credibility** is equal to the smallest $1 - \epsilon$ such that the predicted label set is empty. This measure corresponds to the probability, according to the model, that none of the possible labels are correct.

In settings where predicting a set of labels doesn't make sense, using conformal methods can allow us to yield a single label as well as an associated confidence and credibility score for the prediction.

4.1 Measures of Nonconformity

As noted in Card et al. [2019], DWAC models have a natural measure of nonconformity that's different than the traditional inverse probability transformation in equation. 4. Although less *exact* than the inverse probability, the inverse of the numerator from equation 1 also grows inversely to the probability. Additionally, using the inverse numerator as a measure of non-conformity also has an intuitive interpretation as it correlates with the summed embedded distances from all instances of the same class in the training set. For ProtoDWAC, we also experiment with using the analog of this as a measure of nonconformity.

5 Experiments

We compare our ProtoDWAC model to the original DWAC model, and a baseline model, which uses a traditional softmax for the final layer. All 3 models use the feature extractor introduced by Mullenbach et al. [2018], a CNN with self attention run over pretrained GloVe word vectors [Pennington et al., 2014].

We show our results on two different datasets: the first was extracted from the movie database IMDb by Maas et al. [2011], and is a binary classification problem, where the model must predict whether a movie review is positive or negative. The second is taken from the programming question and answer site stackoverflow.com Xu et al. [2015], where the model must predict which category a post made on the website belongs to out of 20 possible categories.

Dataset	# classes	# instances	vocabulary size
IMDb	2	50000	11200
StackOverflow	20	20000	15000

Table 1: Properties of datasets used in this paper

5.1 Accuracy and Calibration

We report our model accuracies, as well as the accuracy of the predicted probabilities (calibration), measured in terms of mean absolute error (MAE), using the adaptive binning approach of Nguyen and O'Connor [2015]. We use a random 10% of the training data as a validation/calibration set.

	Accuracy			Calibration		
Dataset	Softmax	DWAC	ProtoDWAC	Softmax	DWAC	ProtoDWAC
IMDb	0.905	0.904	0.905	0.029	0.024	0.029
StackOverflow	0.869	0.866	0.867	0.009	0.010	0.011

Table 2: Accuracy and Calibration results

5.2 Out-of-domain data experiments

In addition to metrics capturing the performance of our models, we present plots showing the conformal metric of credibility, which is inversely proportional to the probability that none of the possible labels are correct. Intuitively, credibility measures how much we can trust our model, as it gives an insight into how strongly our model believes that it is possible to correctly label the data. In cases where the credibility scores are low, we have a signal not to trust our model's predictions, which is helpful if we are incorporating the model in any real world decision making pipeline.

To understand how meaningful our ProtoDWAC model's credibility scores are, we experiment with training on IMDb data, and trying to predict on StackOverflow data – a task where our model should assign low credibility scores to the data, as the testing distribution is entirely different from training.

We present histograms for credibility scores for the baseline model, the DWAC model, and our ProtoDWAC model below.

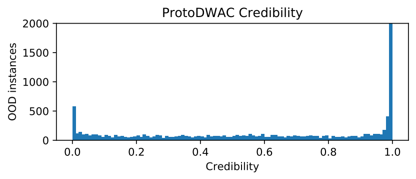

Figure 1: Credibility scores for ProtoDWAC, evaluated on out of domain (OOD) data

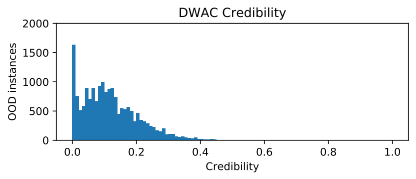

Figure 2: Credibility scores for DWAC, evaluated on OOD data

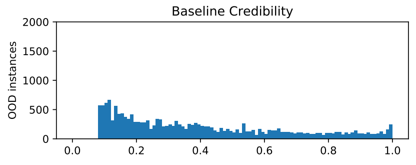

Figure 3: Credibility scores for baseline, evaluated on OOD data

While our ProtoDWAC model performs on par with the other two models with respect to accuracy and calibration, the credibility scores expose a flaw in our approach. Our model should assign low credibility scores to the OOD (Out of Domain) instances, but instead, the mode of its credibility distribution 1.0. This is in contrast to the DWAC model, which assigns fairly low credibility scores. More work needs to be done to fully understand why our model gives high credibility scores in a scenario where the data *should* be extremely difficult to label correctly. However, we hypothesize that the ProtoDWAC model learns to embed instances closer to the prototypes during training, so at test time, despite the instances belonging to a different distribution, the instances are still embedded close to the prototypes, tricking our model into believing that it can predict a correct label for the instances. We plan on investigating this more in the future, and experiment with better ways of initializing and updating our prototypes to see if this results in more desirable credibility scores.

6 Related Work

The original DWAC model and our ProtoDWAC model both draw on work done in instance based learning. Instance based learning refers to the class of learners which make test time predictions by explicitly comparing unseen instances to a weighted combination of training instances, rather than by learning parameters that generalize to new instances.

One of the most well known examples of an instance based learner is the SVM [Boser et al., 1992]. The objective of the model is to learn a separating hyperplane that maximizes the margin (the distance between training points and the hyperplane). This margin, and the hyperplane by extension, are defined by the points in the training set. These points are referred to as ”support vectors”, as they support the margin. This approach of classification is similar to our approach as our prototypes function like support vectors, as they define the decision boundaries for classification. While the prototypes are learned embeddings, and are not explicitly defined as training points, they exist in the same embedding space as our training instances, and can be represented as some linear combination of our training points.

Another key feature of the DWAC and ProtoDWAC model is using a Gaussian kernel to weight the distance between two points. Kernels are not typically used in neural models, as neural networks are able to explicitly learn representations in a different dimensional space than the input without using kernels. However, some work has been done in incorporating kernels in neural networks, particulary convolutional neural networks (CNNs). Mairal et al. [2014] introduce a ”convolutional kernel network”, which is a CNN trained to approximate a kernel feature map as an unsupervised training criteria. This is related to the work in the original DWAC paper, and our ProtoDWAC model because incorporating kernels in a neural model is a non trivial task, and Mairal et al. [2014] show that the approach is valid and can offer strong results. However, while they are essentially learning a kernel function, we choose a Gaussian kernel operating on Euclidean distance.

7 Conclusion

In this paper, we have demonstrated that it is possible to create an alternative to DWAC that utilizes a small set of learnable prototypes with many of the desirable characteristics as the original DWAC model, with significantly faster training and testing performance. However, while our ProtoDWAC model inherits many of the advantages of the DWAC model, it fails to produce the desired credibility scores of the original DWAC model, perhaps implying that more work needs to be done in intelligently incorporating prototypes into the DWAC framework.

8 Acknowledgements

This work would not have been possible without the guidance of our instructor Noah Smith, and our TA’s Elizabeth Clark and Lucy Lin. In particular, we appreciate the feedback given during blog posts and during our updates. In addition, we would like to thank Akshat Shrivastava for his feedback while writing this paper, Divye Jain for discussing ideas while we were conceptualizing the project, and all our classmates who gave feedback anonymously on Piazza, and asked questions during our updates.

References

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Dallas Card, Michael Zhang, and Noah A. Smith. Deep weighted averaging classifiers. In *Proceedings of FAT**, 2019.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635, 2014.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In *NAACL-HLT*, 2018.

Khanh Nguyen and Brendan O'Connor. Posterior calibration and exploratory analysis for natural language processing models. *arXiv preprint arXiv:1508.05154*, 2015.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Craig Saunders, Alexander Gammerman, and Vladimir Vovk. Transduction with confidence and credibility. In *IJCAI*, 1999.

Jiaming Xu, Wang Peng, Tian Guanhua, Xu Bo, Zhao Jun, Wang Fangyuan, Hao Hongwei, et al. Short text clustering via convolutional neural networks. 2015.