Binyamin Friedman
Michtom School of Computer Science
Brandeis University

March 15, 2024

ASSIGNMENT 2

1 Paper Reading

1. What are the contributions of this paper?

   "SphereFace: Deep Hypersphere Embedding for Face Recognition" introduces the angular Softmax loss, an alternative to regular Softmax that aims to enable convolutional neural networks to learn angularly discriminative features. Instead of making prior assumptions about the distribution of features in Euclidean space, A-Softmax loss spans features on a hypersphere manifold. The authors also find an effective lower bound for the m parameter of A-Softmax, using experimental and theoretical methods, such that the loss function enforces the requirement that the minimal inter-class distance is larger than maximum intra-class distance. This requirement is theoretically very important, as it guarantees that a model has the means to discriminate between different classes. Notably, A-Softmax loss achieves very competitive results in several benchmarks and has measurable improvements over regular Softmax.

2. Illustrate the three properties of the proposed A-Softmax.

   As per the original paper, **Property 1.** is that A-Softmax loss defines a large angular margin learning task with difficulty adjustable via the m parameter. When m is larger, the learned region on the hypersphere manifold is more constrained, and the learning task becomes more difficult. **Property 2.** is that $m_{min} \geq 2 + \sqrt{3}$ in the binary classification case, where m_{min} denotes the minimum value of m such that the loss function enforces the requirement that the minimal inter-class distance is larger than maximum intra-class distance. **Property 3.** is that $m_{min} \geq 3$ in the multi-class classification case. In experiments, $m \geq 4$ was an especially effective bound.

3. What is the evaluative metric used for the LFW dataset? How does it calculate?

   To evaluate their A-Softmax CNN models on LFW, the authors extracted the features after the first FC layer. The final representation of a face was obtained by concatenating its original face features with its horizontally flipped features. Then, the metric for face verification was calculated with the cosine distance between two sets of features. They achieved very high results, up to 99.42% accuracy on LFW with a 64-layer CNN. In these tests, A-Softmax loss consistently achieved 1.5-2% greater accuracy than regular Softmax loss.

2 Sphereface Implementation

A 4-layer convolutional neural network using A-Softmax loss was created and trained in PyTorch. PyTorch's provided LFWPeople dataset wrapper was used for training and its provided LFWPairs dataset wrapper was used for cosine similarity evaluation. Some simple data augmentation was done by randomly flipping faces horizontally. The convolutional layers follow the setup of the original paper. A few different open-source implementations of A-Softmax loss were considered, and one was chosen that used an implementation common to SphereFace, ArcFace, and CosFace. Normalization

was done on the input to the model but not on the fully connected layer (this broke training in tests, perhaps because of a faulty implementation).

The training procedure ran for 128 epochs. A batch size of 128 samples was used. Basic learning rate scheduling was used; the learning rate started at 0.001 and was lowered such that it decreased by 10 times every 64 epochs. After the first 80 epochs, the testing accuracy essentially peaked at around 74%. See Figure 1. The loss continued to decrease at a linear rate for the final epochs. Since the testing rate had converged, learning was stopped at this point. See Figure 2.

While this implementation achieved a significantly better result than the 65.80% achieved in the assignment template, it is still subpar compared to the results reported in SphereFace. The authors report a testing accuracy of 98.2% for the 4-layer CNN. It is possible that this difference amounts to better data augmentation or perhaps a better tuned A-Softmax loss implementation. The results achieved here do suggest that better performance could be achieved with better data augmentation and regularization. This conclusion comes from the fact that the training loss was still steadily decreasing well after the training accuracy stopped improving. Further experimentation is warranted, along with testing of the official code implementations.

Note that the code base was specifically designed to be modular. The classifier, model, training and testing routines, and program arguments were all separated into different files, which allowed for easy comparison with the reference template provided with the assignment.

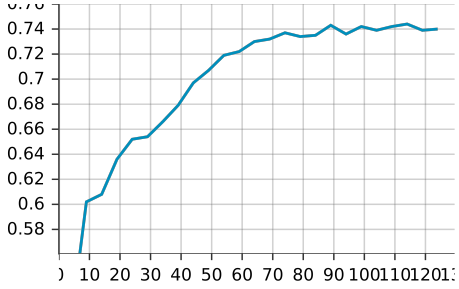

Figure 1: The testing rate accuracy increases steadily (with small fluctuations) for the first 70 epochs, and then appears to mostly level out at around 74%.

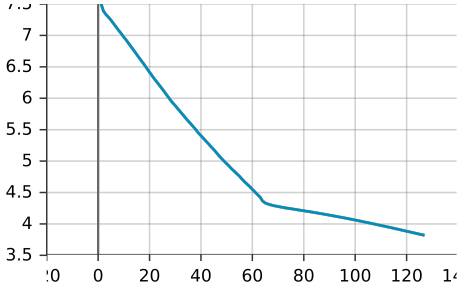

Figure 2: The loss steadily decreases for the duration of training, even after testing rate accuracy stops increasing.

Submitted by Binyamin Friedman on March 15, 2024.