Advanced Computer Vision

Exercise Sheet 7

Winter Term 2023
Prof. Dr. Niels Landwehr
Dr. Ujjwal

Available: 19.12.2023
Hand in until: 09.01.2024 until 23:59
Exercise session: 12.01.2024

Task 1 – Embedding From Classification Model [20 points]

In the notebook *metric_learning.ipynb* you find a definition for a simple classification model for the MNIST data set. Modify this network by putting in an extra layer right before the final class score layer that consists of only two nodes. This layer will be used as an easy-to-visualize embedding for instances, as for the toy example network shown on Slide 15 in the metric learning lecture. Load the MNIST training data set with the method provided in *metric_learning.ipynb*. Train you model on the MNIST training data using standard cross-entropy loss, recommended for 100 epochs.

Compute the activations in the two-node embedding layer for all instances in the reduced training set you used to train the model. Visualize the resulting embedding by a scatterplot of the embedding activations as shown in the lecture. The file *embedding_crossentropy.png* shows how your embedding should approximately look like (obtained from a model trained for 100 epochs).

Note: if you prefer to use PyTorch that is of course also possible.

Task 2 – Embedding From Contrastive Loss [30 points]

In this exercise, we train the same model as used in Task 1 using a *contrastive loss* on pairs of examples. Let $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathbb{R}^D$ denote a neural network producing a D-dimensional embedding ($D = 2$ in our case). For a pair of instances $\mathbf{x}_i$, $\mathbf{x}_j$ with labels y_i, y_j, the contrastive loss is

$$\ell((f_{\boldsymbol{\theta}}(\mathbf{x}_i), f_{\boldsymbol{\theta}}(\mathbf{x}_j)), (y_i, y_j)) = I(y_i = y_j)D_{i,j} + (1 - I(y_i = y_j)) \max(0, \alpha - D_{i,j})$$

where $D_{i,j} = ||f_{\boldsymbol{\theta}}(\mathbf{x}_i) - f_{\boldsymbol{\theta}}(\mathbf{x}_j)||^2$ is the squared distance in embedding space for the two instances, $||\cdot||$ is the Euclidian norm and α is a hyperparameter of the loss function called the margin. Intuitively, the loss enforces that for pairs from the same class the embeddings are close, while for pairs from different classes embeddings are at least α apart.

The loss function given above works on pairs, so we need some strategy to generate pairs over which the loss is computed. A simple way to compute the pair-based contrastive loss when feeding the model standard minibatches during training is given in the pseudocode below. It describes a loss function **ConstractiveLoss** that takes as input model outputs $f_{\boldsymbol{\theta}}(\mathbf{x}_1), ..., f_{\boldsymbol{\theta}}(\mathbf{x}_n) \in \mathbb{R}^D$ and labels $y_1, ..., y_n$ on a minibatch of size n. It then constructs all possible pairs from the mini batch, and computes and adds up the contrastive loss terms for these pairs.

ContrastiveLoss$(f_{\boldsymbol{\theta}}(\mathbf{x}_1), ..., f_{\boldsymbol{\theta}}(\mathbf{x}_n), y_1, ..., y_n)$
$L = 0$
for i `= 1 to n` **do**
  for j `= i+1 to n` **do**
    $D = ||f_{\boldsymbol{\theta}}(\mathbf{x}_i) - f_{\boldsymbol{\theta}}(\mathbf{x}_j)||^2$
    if $y_i = y_j$ **then**
      $L = L + D$
    else
      $L = L + \max(0, \alpha - D)$
    end if
  end for
end for
return L

Train the model given in Task 1 using contrastive loss. To do so, first remove the final classification layer, so that the model directly produces embeddings $f_{\boldsymbol{\theta}} \in \mathbb{R}^2$. Define a custom loss function in Keras (or PyTorch) that implements the pair-based contrastive loss as sketched in the pseudocode above. Use a small batch size (recommended: 8) because the computation of the loss scales quadratically in the batch size. Use a small learning rate (recommended: 0.0001) and margin parameter $\alpha = 1$. Recommended to train for 100 epochs (can take 1-2 hours). As in Task 1, visualize the resulting embedding.

The file *embedding_contrastive.png* shows how your embedding should approximately look like (obtained from a model trained for 100 epochs).