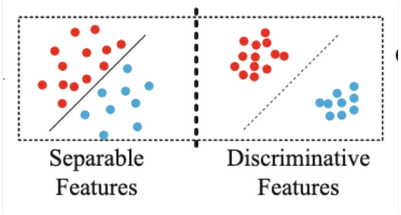
A Discriminative Feature Learning Approach for Deep Face Recognition

- [[📚 WenECCV16.pdf]]
- **Basic Idea**:
  - The Triplet Loss is based on sample-to-sample comparisons and hence, quickly becomes prohibitive with the increase in the size of the dataset. Moreover, the triplets needs to be carefully chosen (neither too hard nor too easy).
  - Use softmax based classifer to learn good features which can even distinguish unseen faces.
  - The learned features should be such that inter-class separation is maximised whereas intra-class variability is minimised.

- **Separable vs Discriminative Features**:
- 🟡 **P2**



- 🟡 **P2** Discriminative power characterizes features in both the compact intra-class variations and separable inter-class differences, as shown in Fig. 1
- 🟡 **P2** However, the softmax loss only encourage the separability of features. The resulting features are not sufficiently effective for face recognition.
- 🟡 **P2** Because the stochastic gradient descent (SGD) [19] optimizes the CNNs based on mini-batch, which can not reflect the global distribution of deep features very well. Due to the huge scale of training set, it is impractical to input all the training samples in every iteration. As alternative approaches, contrastive loss [10,29] and triplet loss [27] respectively construct loss functions for image pairs and triplet.

- **Proposed Approach**:
  - Learn a class center for each class (a vector of the same dimension as the embedding/feature dimension)
  - While training, simultaneously update the centers and minimise the distance between the deep features and their corresponding class centers.
  - Train with both softmax loss and center loss (balanced using a hyper-parameter): Softmax loss ensures features of the different classes stay apart.
  - Note that similar objective can also be achieved using a margin (which is used in later papers). Also, margin is used in Triplet Loss to ensure that discriminative power of the learned features. (**TRY**: Is the Triplet Loss or Contrastive Loss successful without a margin?)

  - **Center Loss**:
    - Motivation: Minimise the intra-class variations while keeping the features of different classes separable.
    - $$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} ||x_i - c_{y_i}||_2^2$$

      where $c_{y_i} \in \mathbb{R}^d$ denotes the y_ith class center of deep features.
    - Notice the similarity of training procedure/loss with *K-means clustering*. (**THINK**: Something on the line of *Expectation Minimization Algorithm*).
    - 🟡 **P5** he formulation effectively characterizes the intra-class variations. Ideally, the c_{y_i}
      🟡 **P5** should be updated as the deep features changed. In other words, we need to take the entire training set into account and average the features of every class in each iteration, which is inefficient even impractical. Therefore, the center loss can not be used directly.
    - To address the above problem, they introduce two modifications:
      - 🟡 **P5** First, instead of updating the centers with respect to the entire training set, we perform the update based on mini-batch. In each iteration, the centers are computed by averaging the features of the corresponding classes (In this case, some of the centers may not update).
      - **Robustness to Noise Labels**: 🟡 **P5** Second, to avoid large perturbations caused by few mislabelled samples, we use a scalar α to control the learning rate of the centers.
    - The gradients are computed as follows:

$$\frac{\partial \mathcal{L}_C}{\partial \mathbf{x}_i} = \mathbf{x}_i - c_{y_i}$$

NOTE: Since in the present iteration, c_{y_i} is a constant, i.e., it's value will not be influenced by $\mathbf{x}_i$, hence, $\frac{\partial \mathcal{L}_C}{\partial \mathbf{c}_{y_i}} = 0$.

$$\Delta c_j = \frac{\sum_{i=1}^m \delta(y_i = j) \cdot (c_j - x_i)}{1 + \sum_{i=1}^m \delta(y_i = j)}$$

NOTE:

- where $\delta(condition) = 1$ if the $condition$ is satisfied, and $\delta(condition) = 0$ if not. α is restricted in [0, 1].
- 1 is added in the denominator to ensure that if no example belongs to class j, then the update $\Delta c_j = 0$.
- The update rule is similar to an incremental update rule for calculating average of a quantity:

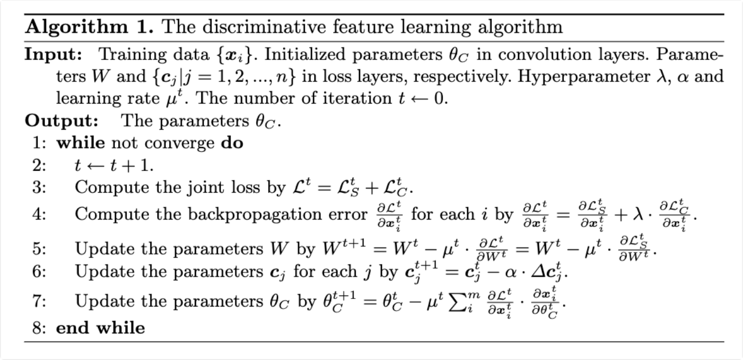
$$\bar{x}_{n+1} = \bar{x}_n - \alpha(\bar{x}_n - x_n)$$
$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$

where α is the learning rate for calculating the average. For exact, average calculation use $\alpha = \frac{1}{n}$. For proof, refer to *Sutton and Barto, Reinforcement Learning, 2nd Edition Pg 31.*

- Final loss function is given by:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= -\sum_{i=1}^{m} log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^n e^{W_j^T x_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} ||x_i - c_{y_i}||_2^2 \end{aligned}$$

- 🟡 **P6**



Algorithm 1. The discriminative feature learning algorithm

Input: Training data $\{\boldsymbol{x}_i\}$. Initialized parameters θ_C in convolution layers. Parameters W and $\{\boldsymbol{c}_j | j = 1, 2, ..., n\}$ in loss layers, respectively. Hyperparameter λ, α and learning rate μ^t. The number of iteration $t \leftarrow 0$.

Output: The parameters θ_C.

```
while not converge do
    $t \leftarrow t + 1$.
    Compute the joint loss by $\mathcal{L}^t = \mathcal{L}_S^t + \mathcal{L}_C^t$.
    Compute the backpropagation error $\frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t}$ for each $i$ by $\frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t} = \frac{\partial \mathcal{L}_S^t}{\partial \boldsymbol{x}_i^t} + \lambda \cdot \frac{\partial \mathcal{L}_C^t}{\partial \boldsymbol{x}_i^t}$.
    Update the parameters $W$ by $W^{t+1} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}^t}{\partial W^t} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}_S^t}{\partial W^t}$.
    Update the parameters $\boldsymbol{c}_j$ for each $j$ by $\boldsymbol{c}_j^{t+1} = \boldsymbol{c}_j^t - \alpha \cdot \Delta \boldsymbol{c}_j^t$.
    Update the parameters $\theta_C$ by $\theta_C^{t+1} = \theta_C^t - \mu^t \sum_i^m \frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t} \cdot \frac{\partial \boldsymbol{x}_i^t}{\partial \theta_C^t}$.
end while
```

- **DISCUSSION**:
  - **The necessity of joint supervision.** 🟡 **P7** If we only use the softmax loss as supervision signal, the resulting deeply learned features would contain large intra-class variations. On the other hand, if we only supervise CNNs by the center loss, the deeply learned features and centers will degraded to zeros (At this point, the center loss is very small).

- **Tricks of the Trade**:
  - Use of PReLU activation function.
  - For score computation during testing, extract features from each image and its horizontally flipped one, and concatenate them as the representation. The score is computed by the Cosine Distance of two features after PCA.

Unlinked References