Metric Learning

Background

Metric learning is the task of learning a distance function over objects. The idea is to learn a function that maps input patterns into a target space such that the L1/L2 norm in the target space approximates the “semantic” distance in the input space. It is commonly used as a pretext task for Self Supervised Learning (with Image Classification being the downstream task). For this, we first apply random transformations to images in the dataset. Then we build a trainable system that non-linearly maps the raw images to points in a low dimensional space so that the distance between these points is small if the images are transformations of each other and large otherwise.

The loss functions used with metric learning are called Ranking Loss. Unlike other loss functions, such as Cross-Entropy Loss or Mean Square Error Loss,whose objective is to learn to predict a label directly, the objective of RankingLoss is to predict relative distances between inputs. The three most common ranking losses are as follows:

Pairwise Loss / Contrastive Loss

This loss forces representations to have 0 distance for positive pairs, and a distance greater than a margin m for negative pairs.

Let a represent the anchor sample, p be a sample belonging to the same cluster as a and n belonging to some other cluster. Let r_a, r_p and r_n represent the latent representations of a, p and n respectively. d is a distance function. Then the pairwise ranking loss function L can be written as:

$$L = \begin{cases} d(r_a, r_p) & \text{Positive Pair} \\ max(0, m - d(r_a, r_n) & \text{Negative Pair} \end{cases}$$

If y is a flag set to 1 in case a pair r_0 and r_1 is similar and set to 0 if the pair is dissimilar, then the equation can be written as:

$$L(r_0, r_1, y) = y||r_0 - r_1|| + (1 - y)max(0, m - ||r_0 - r_1||)$$

Triplet Loss

The triplets are formed by an anchor sample x_a, a positive sample x_p and a negative sample x_n. The objective is that the distance between the anchor sample and the negative sample representations $d(r_a, r_n)$ is greater than $d(r_a, r_p)$ by a margin of atleast m.

$$L(r_a, r_p, r_n) = max(0, m + d(r_a, r_p) - d(r_a, r_n))$$

N Pair Loss

During one update, the triplet loss only compares an example with one negative example while ignoring negative examples from the rest of the classes. As a consequence, the embedding vector for an example is only guaranteed to be far from the selected negative class but not necessarily the others. Thus, we can end up only differentiating an example from a limited selection of negative classes yet still maintain a small distance from many other classes. In practice, the hope is that, after looping over sufficiently many randomly sampled triplets, the final distance metric can be balanced correctly; but individual update can still be unstable and the convergence would be slow. Specifically, towards the end of training, most randomly selected negative examples can no longer yield non-zero triplet loss error. N Pair Loss recruits multiple negatives for each update.

Consider an $(N+1)$-tuplet of training examples $\{x, x^+, x_1, \ldots, x_{N-1}\}$: x^+ is a positive example to x and $\{x_i\}_{i=1}^{N-1}$ are negative. The N Pair loss is defined as follows:

$$\begin{aligned} L(\{x, x^+, \{x_i\}_{i=1}^{N-1}\}; f) &= -log \frac{exp(f^T f^+)}{exp(f^T f^+) + \sum_{i=1}^{L-1} exp(f^T f_i)} \\ &= log\left(1 + \sum_{i=1}^{L-1} exp(f^T f_i - f^T f^+)\right) \end{aligned}$$

where f is an embedding kernel defined by deep neural network. The loss equation is similar to the multi-class logistic loss (i.e., softmax loss) formulation where $f^T f^+$ is treated as the logit value of input for the actual class of the input and $f^T f_i$ as the logit value for class i.

For efficient N Pair Loss calculation, we construct optimal mini batches such that each mini batch (of size N) has N-1 negative examples. This reduces the computational burden of evaluating deep embedding vectors. Section 3.2 of N Pair Loss Research Paper goes over it in more detail.

Related Work

DrLIM

LeCun, et al, 2006 uses Pairwise/Contrastive Loss to learn a Reduced Dimensionality Mapping that captures a lot of semantic meaning. Images of airplanes are very well mapped to a lower dimensional manifold, such that airplanes with the same orientation (elevation and azimuths) lie close to each other. This is in spite of the fact that many a times the Euclidean distance for similar images is very different because of the lighting and other factors. The approach of this paper, however, is not one of self-supervision. They use a tagged dataset to derive instance similarity.

Video Processing

The movement of an object is traced by a sequence of video frames. The difference between how the same object is captured on the screen in close frames is usually not big, commonly triggered by small motion of the object or the camera. Therefore any visual representation learned for the same object across close frames should be close in the latent feature space. Motivated by this idea, Wang & Gupta, 2015 proposed a way of unsupervised learning of visual representation by tracking moving objects in videos.

Basically, patches with motion are tracked over a small time window (e.g. 30 frames). The first patch x and the last patch x^+ are selected and used as a training data points. To avoid the model from directly minimizing the difference between two feature vectors by mapping everything to the same value, a random third patch x^- is added. The model learns the representation by enforcing the distance between two tracked patches to be closer than the distance between the first patch and a random one in the feature space. $D(x, x^-) > D(x, x^+)$, where $D(.)$ is the cosine distance.

$$D(x_1, x_2) = 1 - \frac{f(x_1)f(x_2)}{||f(x_1)||||f(x_2)||}$$

The loss function is defined as:

$$\mathcal{L}(x, x^+, x^-) = max(0, D(x, x^+) - D(x, x^-) + M) + \text{weight decay regularization term}$$

where M is a scalar constant controlling the minimum gap between two distances; $M = 0.5$ in the paper. The loss enforces $D(x, x^-) >= D(x, x^+) + M$ at the optimal case.

MoCo

Above mentioned papers use triplet loss or pairwise loss for unsupervised learning. As we just studied, N Pair Loss converges faster than Triplet Loss and gives better results. MoCo does exactly that. It actually makes the N in N Pair Loss independent of the mini-batch size by queueing samples from previous mini batches and using rolling momentum based updates. The dataset used in this paper is again an image dataset. They follow an instance discrimination pre-text task: a query matches a key if they are encoded views (e.g., different crops) of the same image.