deep metric learning

Giorgos Tolias
Czech Technical University in Prague
pairwise similarity

- human cognitive process involves ability to detect similarities between objects
- objects can be images, text documents, sound, etc...
- use deep learning to estimate pairwise similarity / distance

Applications
  - information retrieval
  - k-nearest-neighbor classification
  - clustering
  - data visualization
similarity / metric learning

- definition of good similarity measure (metric) is task dependent
- different semantic notion of similarity per task
  - not well captured by hand-crafted representations and standard metrics
- solution: learn it from the data
representation and similarity learning

- space of input examples $\mathcal{X}$
- learn the representation, use standard similarity measures / metrics
  - embed input examples to a representation (vector) space
  - embedding function $f : \mathcal{X} \rightarrow \mathbb{R}^D$
  - $\mathbf{x} = f(x), x \in \mathcal{X}$
  - learn the representation conditioned on a standard metric
- directly learn the similarity function / metric
  - similarity function $s : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
  - involves learning the representation too
- pre-trained network is given, e.g. trained for classification with cross-entropy loss
- use internal activation vectors as representation
- use existing metrics to estimate pairwise similarity
  - Euclidean distance, cosine similarity, ...
training data - labels

- pairwise labels of training examples
  - relevant (positive, matching) pair
  - non-relevant (negative, non-matching) pair

- available image-level class labels
  - within (across) class pairs are positive (negative)

- manual annotation of pairs
  - typically very costly

- instance-discrimination
  - each image its own class
  - positives obtained by augmentations

[Diagram showing relationships between images and class labels]
metric learning: Mahalanobis distance

- learn a parametric distance function from the data
  - input examples are vectors

- example: Mahalanobis distance
  - $M$ is a $D \times D$ positive semi-definite matrix
  - $d_M(x, z) = \sqrt{(x - z)^\top M(x - z)}$, $x, z \in \mathbb{R}^D$
  - $d_M(x, z) = \sqrt{(x - z)^\top L^\top L(x - z)} = \sqrt{(L(x - z))^\top L(x - z)}$
    - $= \sqrt{(Lx - Lz)^\top (Lx - Lz)} = \|Lx - Lz\|_2 = \|f(x) - f(z)\|_2$
  - mapping function $f(x) = Lx$
  - can be modeled by a single fully-connected layer
metric learning: Mahalanobis distance

- learn a parametric distance function from the data
  - input examples are vectors

- example: Mahalanobis distance
  - $M$ is a $D \times D$ positive semi-definite matrix
  - $d_M(x, z) = \sqrt{(x - z)^\top M (x - z)}, \ x, z \in \mathbb{R}^D$
  - $d_M(x, z) = \sqrt{(x - z)^\top L^\top L (x - z)} = \sqrt{(L(x - z))^\top L (x - z)}$
    
  - $d_M(x, z) = \sqrt{(Lx - Lz)^\top (Lx - Lz)} = \|Lx - Lz\|_2 = \|f(x) - f(z)\|_2$
  - mapping function $f(x) = Lx$
  - can be modeled by a single fully-connected layer

- general cases:
  - $f : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ is a feed-forward network
  - input examples are not vectors, $f : \mathcal{X} \rightarrow \mathbb{R}^{D'}$
contrastive loss

- two branch network; 2 networks that share weights

\[(x_i, x_j) \in S\]
\[y_{ij} = 1\]

\[(x_i, x_j) \in D\]
\[y_{ij} = 0\]

\[\ell(x_i, x_j) = \frac{1}{2} y_{ij} \|x_i - x_j\|_2^2 + \frac{1}{2} (1 - y_{ij}) [\tau - \|x_i - x_j\|_2]^2_+\]
contrastive loss

- similar pair gradients

\[
\frac{\partial \ell}{\partial \mathbf{x}_a} = \mathbf{x}_a - \mathbf{x}_p \\
\frac{\partial \ell}{\partial \mathbf{x}_p} = -\frac{\partial \ell}{\partial \mathbf{x}_a}
\]

- dissimilar pair gradients

\[
\frac{\partial \ell}{\partial \mathbf{x}_a} = \tau - \frac{||\mathbf{x}_a - \mathbf{x}_n||}{||\mathbf{x}_a - \mathbf{x}_n||} (\mathbf{x}_n - \mathbf{x}_a) \\
\frac{\partial \ell}{\partial \mathbf{x}_n} = -\frac{\partial \ell}{\partial \mathbf{x}_a}
\]

\[
\ell(x_i, x_j) = \frac{1}{2} y_{ij} ||\mathbf{x}_i - \mathbf{x}_j||_2^2 + \frac{1}{2} (1 - y_{ij}) [\tau - ||\mathbf{x}_i - \mathbf{x}_j||_2]^2_+
\]
triplet loss

- three branch network; 3 networks that share weights

\[(x_a, x_p) \in S\]
\[(x_a, x_n) \in D\]

\[\ell(x_a, x_p, x_n) = \left[ ||x_a - x_p||_2^2 - ||x_a - x_n||_2^2 + \alpha \right]_+\]

[Schroff et al. 2015]
triplet loss

- gradients

\[ \frac{\partial \ell}{\partial x_p} = 2(x_p - x_a) \]
\[ \frac{\partial \ell}{\partial x_n} = 2(x_a - x_n) \]
\[ \frac{\partial \ell}{\partial x_a} = 2(x_a - x_p) - 2(x_a - x_n) = 2(x_n - x_p) \]

\[ \ell(x_a, x_p, x_n) = \left[ \|x_a - x_p\|^2_2 - \|x_a - x_n\|^2_2 + \alpha \right]_+ \]
pairwise losses

contrastive

\[ \left[ x_a^\top x_n - x_a^\top x_p \right]^+ \]

triplet

\[ \log(1 + e^{x_a^\top x_n - x_a^\top x_p}) \] [Sohn 2016]
mini-batches & hard negatives

- mini-batch construction [Roth et al., 2020]
  - randomly sample \( n \) classes and \( b/n \) examples per class
  - greedy approach to maximize covered space
    - next example maximizes the distances to already included examples
  - match training dataset statistics (distribution of pairwise distances)
    - set of random mini-batches: pick to minimize distribution distance

- sampling of negatives matters
  - random sampling: zero loss for most pairs/triplets
  - hard negatives: negative pair, but nearby in the representation space

- online sampling
  - within batch single hardest, semi-hard mining [Schroff et al. 2015],
    distance-weighted sampling [Wu et al. 2017]

- offline sampling
  - nearest-neighbor search: guaranteed hard negatives in the batch
  - hardness changes: repeat the process during training
histogram loss [Ustinova & Lempitsky, 2016]

- minimize probability that similarity of a random negative pair is higher than the similarity of a random positive pair

$$\mathbb{E}_{u \sim p^-}[\Phi^+(u)] = \int_{-1}^{1} p^-(u) \Phi^+(u) \, du = \int_{-1}^{1} p^-(u) \left[ \int_{-1}^{u} p^+(v) \, dv \right] \, du$$

- approximated by

$$\sum_{r=1}^{R} \left( h_r^- \sum_{q=1}^{r} h_q^+ \right) = \sum_{r=1}^{R} (h_r^- \phi_q^+)$$

- histogram for positive pairs: 

$$h_r^+ = \frac{1}{|\mathcal{P}^+|} \sum_{(i,j): (x_i, x_j) \in \mathcal{P}^+} \delta_{i,j,r}$$

- equivalently for the negative pairs
- **Average-Precision (AP)** is a common retrieval metric

\[
AP_q = \frac{1}{|S_P|} \sum_{i \in S_P} \text{precision}@\text{ranking}_i \\
= \frac{1}{|S_P|} \sum_{i \in S_P} \frac{\#\text{positives-up-to-ranking}_i}{\text{ranking}_i} \\
= \frac{1}{|S_P|} \sum_{i \in S_P} \frac{\mathcal{R}(i, S_P)}{\mathcal{R}(i, S_\Omega)}
\]

- AP is not differentiable

- optimize a smooth approximation instead [Brown et al. 2020]
**smooth AP loss**

- rewrite AP as

\[
AP_q = \frac{1}{|S_P|} \sum_{i \in S_P} \frac{1 + \sum_{j \in S_P, j \neq i} 1\{D_{ij} > 0\}}{1 + \sum_{j \in S_P, j \neq i} 1\{D_{ij} > 0\} + \sum_{j \in S_N, j \neq i} 1\{D_{ij} > 0\}}
\]

- replace the indicator function with sigmoid

\[
AP_q = \frac{1}{|S_P|} \sum_{i \in S_P} \frac{1 + \sum_{j \in S_P, j \neq i} G(D_{ij})}{1 + \sum_{j \in S_P, j \neq i} G(D_{ij}) + \sum_{j \in S_N, j \neq i} G(D_{ij})}
\]
similarity function learning

- learn similarity function $s : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
- input is an example pair
- higher cost for inference

[Zagoruyko & Komodakis, 2015]
beyond binary supervision

\[ \ell(x_a, x_i, x_j, y_a, y_i, y_j) = \left( \log \frac{\|x_a - x_i\|^2}{\|x_a - x_j\|^2} - \log \frac{D(y_a, y_i)}{D(y_a, y_j)} \right)^2 \]

[Kim et al.’19]
self-supervised representation learning

[Khosla et al. 2020]
applications

visual search

local descriptors [Mishkin'16]

visual localization

image classification [Song'16]
applications

data visualization

video tracking [Tao’16]

data exploration [Johnson et al.’17]