Fast low-rank metric learning
Brief notes

Dan Oneaţă

- Implemented NCA objective function in its basic version and tested it on various data sets. Possible issues:
  - The objective function is non-convex; this raises the question of how to initialize the projection matrix A.
    - Random initialization. Simplest idea; obviously, the quality of the results varies. We can try multiple random initializations and pick the best. But does a good starting point guarantee that we will arrive in a better optimum than if we have started from an worse point?
    - Using the first d most discriminative projection directions given by PCA or LDA transformations[1]. These provide good initialization for problems that contain convex classes. For more complicated data sets, this harms the performance.
    - TO DO: Using the projection directions given by the logistic regression. We could implement this fairly easily for two-class problems. More thought is needed if we want to use this idea for multi-class problems.
  - We might encounter numerical issues because the points are situated far away in the initial space. We need to compute:

$$p_i = \frac{\sum_{j \in C_i} \exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}.$$

    If $d_{ij} > \text{ct}, \forall j, j \neq i$ and, let's say, $\text{ct} > 30$, then $p_i \to \frac{0}{0}$. Solutions:
    - Normalize the data set: zero mean and unit variance on each dimension. This guarantees that we start in a more compact space. After this, the projection matrix A will learn which of the dimensions is more discriminative and it will compensate for the initial standardization.
    - Center the data (*i.e.*, zero mean) and then use an initial A that virtually projects the whole data in a space very close together: `A:=0.01*randn(d,D)`. (Laurens van der Maaten's implementation)

[1] Also implemented PCA and LDA for high-dimensional data, *i.e.*, $D \gg N$.

    * However, the `NaN` problem can still appear. The simplest way to deal with this is to replace each `NaN` entry with a very small number: `P:=max(P,eps)` (again, Laurens' implementation). Alternatively, we could apply the log-sum-exp trick.

- Compact support kernels.
  - Derived the mathematical expression for the simplest polynomial kernel with compact support that is differentiable.
  - Also implemented a NCA objective function based on this. The projections are quite similar to the normal NCA that uses the squared distance exponential kernel. However, the compact support version achieves worse general results. As we would expect, the optimization process gets stuck quite easily—the points' positions are quite constrained.
  - Derived the complete equations for NCA casted as a kernel-density estimation problem with compact support kernels and Gaussian background distribution.
  - TO DO: implement the above objective function (not sure how to implement it in an efficient manner).
- k-d trees
  - Implemented k-d tree building and nearest neighbour search in MATLAB. (TO DO: use bucket of points at leaves.)
  - Tested the performance of k-d tree using random A and final version of A that is learnt during the training.
  - Tested how the point contribution varies during the training. This motivates the use of k-d trees.
- Optimization.
  - For the optimization process, Carl's `minimize.m` was mainly used.
  - Implemented stochastic gradient descent using the bold driver heuristic. (TO DO: try to add momentum).
  - Implemented Option 1: move gradually in the parameter space by taking into consideration only a few points and compute their contribution with respect to the whole data set. This ensures that after a few epochs we arrive in a regime where pruning using k-d tree works well. However, care must be taken because we can over confidently go in a direction from which is hard to recover. Possible remedy: use small regularization term while doing the optimization on few points and then run NCA + k-d trees on the whole data set.
  - Implemented Option 2: this uses mini-batches that are constructed by recursive projection clustering.

- Others.
  - Replicated experiments from the article.
  - Implemented the dimensionality annealing idea. This might be useful, if parameters are chosen correctly. Also it might be combined with some better learning procedure, like Option 1.