t-SNE Notes

Pavlin Poličar

1 Fast KL Divergence

During computation of negative gradients, we do not know the value of the normalization term Z during intermediate steps. Therefore, in order to compute the KL divergence of the embedding, we would need at least two passes over the data points, first to compute the unnormalized q_{ij}s, and secondly to normalize them and compute the KL divergence. By rewriting the KL divergence in terms of unnormalized q_{ij}s, we can compute the entire error with a single pass over the data points by accumulating the $\sum_{ij} p_{ij}$ and $\sum_{ij} q_{ij}$ in the first pass.

$$\begin{aligned} KL(P \mid\mid Q) &= \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_{ij} p_{ij} \log \left(p_{ij} \frac{Z}{\hat{q}_{ij}} \right) \end{aligned}$$

where $\hat{q}_{ij}$ denotes the unnormalized values q_{ij}

$$= \sum_{ij} p_{ij} \log \frac{p_{ij}}{\hat{q}_{ij}} + \sum_{ij} p_{ij} \log Z$$

Therefore the first term requires a single pass over all i, js and the second term can be computed in constant time if we accumulate the sums of P and Q.

2 KL Divergence with exaggeration

The implemented optimization methods don't have a notion of exaggeration, they simply take an affinity matrix P containing the probabilities of points j appearing close to i. Exaggeration is used to scale P by some constant factor α (in fact this means that p_{ij}s are not proper probabilities) to help separate clusters in the beginning of the optimization. These methods also compute the KL divergence during optimization (for efficiency), and as such, the error is incorrect because we don't account for the scaling α.

This section derives a quick and simple correction for the KL divergence error term so we can get the true error of the embedding even when P is exaggerated.

$$KL(P \mid\mid Q) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{1}$$

We need to introduce the scaling i.e. exaggeration factor α to every p_{ij} term

$$= \sum_{ij} \frac{\alpha}{\alpha} p_{ij} \log \frac{\alpha p_{ij}}{\alpha q_{ij}} \tag{2}$$

Exaggeration means that the p_{ij} terms get multiplied by α, so we need to find an expression for the KL divergence that includes only αp_{ij} and q_{ij} and some other factor that will correct for α.

$$= \frac{1}{\alpha} \sum_{ij} \alpha p_{ij} \left(\log \frac{\alpha p_{ij}}{q_{ij}} - \log \alpha \right) \tag{3}$$

$$= \frac{1}{\alpha} \left(\sum_{ij} \alpha p_{ij} \log \frac{\alpha p_{ij}}{q_{ij}} - \sum_{ij} \alpha p_{ij} \log \alpha \right) \tag{4}$$

The first term is computed by the gradient method (since it only knows about the scaled P), the second term can easily be computed post-optimization, allowing us to get the correct KL divergence.

3 Transform

3.1 Direct optimization

3.2 General framework of cost functions

[Bunte et al., 2012]

3.3 MDS interpolation

MDS Interpolation [Bae et al., 2010]. A similar approach might be able to be applied to tSNE. In essence, they run MDS on a sample of points. Then for each new point, we compute the k-nearest neighbors and optimize the stress function w.r.t. only those points. In their paper, they derive equations that can be used for efficient optimization via majorization.

3.4 Kernel tSNE

[Gisbrecht et al., 2012] claim to outperform direct mapping tSNE using a direct kernel mapping. This paper is not very useful. The graph is misleading and the table at the end is informative, but run only on small datasets. Their subsequent paper is much better and throughout.

In [Gisbrecht et al., 2015], kernel tSNE is described in more detail and parameters are chosen in a more principled manner.

Describes how to integrate class labels into embedding using Fischer information.

The issue of kernel tSNE is that we have to compute the inverse of the interaction matrix K. We can use P as the interaction matrix, and P is sparse, but the inverse of that is very dense, and for any reasonably sized data set, this is unfeasable.

References

[Bae et al., 2010] Bae, S.-H., Choi, J. Y., Qiu, J., and Fox, G. C. (2010). Dimension reduction and visualization of large high-dimensional data via interpolation. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 203–214. ACM.

[Bunte et al., 2012] Bunte, K., Biehl, M., and Hammer, B. (2012). A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, 24(3):771–804.

[Gisbrecht et al., 2012] Gisbrecht, A., Lueks, W., Mokbel, B., and Hammer, B. (2012). Out-of-sample kernel extensions for nonparametric dimensionality reduction. In *ESANN*, volume 2012, pages 531–536.

[Gisbrecht et al., 2015] Gisbrecht, A., Schulz, A., and Hammer, B. (2015). Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, 147:71–82.