t-SNE algorithm implementation

Tomas Ukkonen 2020, `tomas.ukkonen@novelinsight.fi`

Implementation is in `src/neuralnetwork/TSNE.h` and `TSNE.cpp` in `dinrhiw2` repository.

We maximize KL-divergence (p_{ij} calculated from data).

$$D_{\mathrm{KL}}(\boldsymbol{y}_1 \ldots \boldsymbol{y}_N) = \sum_{i \neq j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right),\ q_{ij} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}}$$

The p_{ij} values are calculated from data using formulas.

$$p_{j|i} = \frac{e^{-\|\boldsymbol{x}_j - \boldsymbol{x}_i\|^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|\boldsymbol{x}_k - \boldsymbol{x}_i\|^2 / 2\sigma_i^2}},\ p_{i|i} = 0,\ \sum_j p_{j|i} = 1$$

Symmetric probability values are computed from conditional probabilities using the formula $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \sum_{i,j} p_{ij} = 1$

The variance terms of each data point σ_i^2 is calculated using values $p_{j|i}$ to search for target perplexity $\mathrm{perp}(P_i) = 2^{H(P_i)} = 2^{-\sum_j p_{j|i} \log_2(p_{j|i})}$. Good general perplexity value is maybe 30 so we need to solve σ_i^2 value using bisection method.

First we set minimum $\sigma_{\min}^2 = 0$ and $\sigma_{\max}^2 = \mathrm{trace}(\boldsymbol{\Sigma}_{\boldsymbol{x}})$. We then always select $\sigma_{\mathrm{next}}^2 = \frac{\sigma_{\min}^2 + \sigma_{\mathrm{msx}}^2}{2}$ to half the interval and calculate perplexity at σ_{next}^2 to figure out which half contains the target perpelexity value and stop if error is smaller than 0.1.

Gradient

We need to calculate gradient for each $\boldsymbol{y}_i$ in D_{KL}.

$$\nabla_{\boldsymbol{y}_m} D_{\mathrm{KL}} = \nabla_{\boldsymbol{y}_m} \sum_{i \neq j} -p_{ij} \log(q_{ij}) = -\sum_{i \neq j} \frac{p_{ij}}{q_{ij}} \nabla_{\boldsymbol{y}_m} q_{ij}$$

The general rule to derivate q_{ij} terms is:

$$\nabla \frac{f}{g} = \nabla f g^{-1} = f' g^{-2} g - f g^{-2} g' = \frac{f' g - f g'}{g^2}$$

And when $m \neq i \neq j$ we need to derivate only the second part

$$\nabla_{\boldsymbol{y}_{m \neq i \neq j}} \left(\frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}} \right)$$
$$= -\frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{(\sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1})^2} \nabla_{\boldsymbol{y}_m} \sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}$$

$$\nabla_{\boldsymbol{y}_{m \neq i \neq j}} \sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}$$
$$= \nabla_{\boldsymbol{y}_m} \sum_{l \neq m} (1 + \|\boldsymbol{y}_m - \boldsymbol{y}_l\|^2)^{-1} + \nabla_{\boldsymbol{y}_m} \sum_{k \neq m} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_m\|^2)^{-1}$$
$$= 2\, \nabla_{\boldsymbol{y}_m} \sum_{l \neq m} (1 + \|\boldsymbol{y}_m - \boldsymbol{y}_l\|^2)^{-1}$$
$$= 2 \sum_{l \neq m} \nabla_{\boldsymbol{y}_m} (1 + \|\boldsymbol{y}_m - \boldsymbol{y}_l\|^2)^{-1}$$
$$= 4 \sum_{l \neq m} -(1 + \|\boldsymbol{y}_m - \boldsymbol{y}_l\|^2)^{-2} (\boldsymbol{y}_m - \boldsymbol{y}_l)$$

And when $y = i$ or $y = j$ we need to derivate the upper part too.

$$\nabla_{\boldsymbol{y}_i} \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}} = \frac{1}{\sum_k \sum_{l \neq k} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}} \nabla_{\boldsymbol{y}_i} (1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1} - \frac{f g'}{g^2}$$
$$\nabla_{\boldsymbol{y}_i} (1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1} = -2\, (1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-2} (\boldsymbol{y}_i - \boldsymbol{y}_j)$$

With these derivates we can then calculate derivate of D_{KL} for each $\boldsymbol{y}$. We just select step length for the gradient which causes increase in D_{KL}.

Optimization of computation

For large number of points the update rule is slow ($O(N^2)$ scaling). Extra speed can be archieved by combining large away data points to a single point which is then used to calculate the divergence and gradient. This can be done by using *Barnes-Hut approximation* which changes computational complexity to near linear $O(N \log(N))$.