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 $\sigma_{\mathrm{min}}^2 = 0$ and $\sigma_{\mathrm{max}}^2 = \mathrm{trace}(\boldsymbol{\Sigma}_{\boldsymbol{x}})$. We then always select $\sigma_{\mathrm{next}}^2 = \frac{\sigma_{\mathrm{min}}^2 + \sigma_{\mathrm{msx}}^2}{2}$ 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 $O(N\log(N))$.

Improvement of KL divergence based distribution comparision

The information theoretic distribution comparision metric D_{KL} can be improved by using absolute values. This also symmetrices comparision a bit. (See my other notes about information theory/also at the end of this section.)

$$|D|_{\mathrm{KL}}(\boldsymbol{y}_1 ... \boldsymbol{y}_N) = \sum_{i\neq j} p_{ij} \left|\log\left(\frac{p_{ij}}{q_{ij}}\right)\right|$$

Gradient of the absolute value can be computed using a simple trick.

$$D\|f(\boldsymbol{x})\| = D\sqrt{\|f(\boldsymbol{x})\|^2} = \frac{1}{2\sqrt{\|f(\boldsymbol{x})\|^2}} D\|f(\boldsymbol{x})\|^2 = \frac{f(\boldsymbol{x})}{\|f(\boldsymbol{x})\|}\nabla f(\boldsymbol{x}) = \mathrm{sign}(f(\boldsymbol{x}))\nabla f(\boldsymbol{x})$$

This means the improved gradient is:

$$\nabla_{\boldsymbol{y}_m}|D|_{\mathrm{KL}} = \sum_{i\neq j} p_{ij}\,\nabla_{\boldsymbol{y}_m}\left|\log\left(\frac{p_{ij}}{q_{ij}}\right)\right| = -\sum_{i\neq j}\frac{p_{ij}}{q_{ij}}\,\mathrm{sign}\left(\log\left(\frac{p_{ij}}{q_{ij}}\right)\right)\nabla_{\boldsymbol{y}_m} q_{ij}$$

This means we only need to add $\mathrm{sign}(x)$ non-linearity to the gradient calculation code. The $\mathrm{sign}(x)$ non-linearity is well defined everywhere else except at zero where we can select $\mathrm{sign}(0) = 1$ without having much problems in practice.

Justification of the modified KL divergence

The absolute value can be justified by following calculations. Geometric mean of observed symbol string is P and the number of symbols $l = 1...L$ in N symbol long string is n_l. Additionally we let the length of string to go to infinity ($N \to \infty$):

$$P = \left(\prod_k^N p(\boldsymbol{x}_k)\right)^{1/N} = \left(\prod_l^L p(l)^{n_l}\right)^{1/N} \approx \prod_l^L p(l)^{p(l)}$$

By taking the logarithm of P we get formula for entropy: $\log(P) = \sum_l p(l)\log(p(l)) = -H(L)$.

Comparing distributions probabilities we can write ($N \to \infty$):

$$Q_{\boldsymbol{x}} = \left(\frac{\prod_k^N p(\boldsymbol{x}_k)}{\prod_k^N p(\boldsymbol{y}_k)}\right)^{1/N} = \left(\prod_l^L \left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}\right)^{n_l}\right)^{1/N} \approx \prod_l^L \left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}\right)^{p_{\boldsymbol{x}}(l)}.$$

And by taking the logarithm of Q we get Kullback-Leibler divergence:

$$\log(Q_{\boldsymbol{x}}) = \sum_l p_{\boldsymbol{x}}(l)\log\left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}\right) = D_{\mathrm{KL}}$$

Now by always taking the maximum ratio of probabilties when computing Q we don't have the problem that multiplication (in $\prod_l^L \left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}\right)^{n_l}$-term) of probability ratios would cancel each other reducing the usability of comparision.

$$|Q_{\boldsymbol{x}}| = \left(\prod_l^L \max\left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}, \frac{p_{\boldsymbol{y}}(l)}{p_{\boldsymbol{x}}(l)}\right)^{n_l}\right)^{1/N} \approx \prod_l^L \max\left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}, \frac{p_{\boldsymbol{y}}(l)}{p_{\boldsymbol{x}}(l)}\right)^{p_{\boldsymbol{x}}(l)}$$

$$\log|Q_{\boldsymbol{x}}| = \sum_l p_{\boldsymbol{x}}(l)\log\left(\max\left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}, \frac{p_{\boldsymbol{y}}(l)}{p_{\boldsymbol{x}}(l)}\right)\right) = \sum_l p_{\boldsymbol{x}}(l)\left|\log\left(\frac{p_{\boldsymbol{x}}(l)}{p_{\boldsymbol{y}}(l)}\right)\right| = |D_{\boldsymbol{x}}|_{\mathrm{KL}}$$

Further symmetrization can be done by taking the geometric mean:

$$|Q| = (|Q_{\boldsymbol{x}}|\,|Q_{\boldsymbol{y}}|)^{1/2},\ \log(|Q|) = \tfrac{1}{2}(\log|Q_{\boldsymbol{x}}| + \log|Q_{\boldsymbol{y}}|) = \tfrac{1}{2}(|D_{\boldsymbol{x}}|_{\mathrm{KL}} + |D_{\boldsymbol{y}}|_{\mathrm{KL}}).$$

Improvement of the MSE calculation code

Calculating a gradient of absolute value can be also used in minimum least squares (MSE) optimization where we can then easily use norm instead (minimum norm error - MNE) of the squared error which is then less affected by large outlier values.

$$\mathrm{MSE}(\boldsymbol{w}) = E_{\boldsymbol{xy}}\big[\tfrac{1}{2}\|\boldsymbol{y} - f(\boldsymbol{x})\|^2\big],\ \nabla_{\boldsymbol{w}}\,\mathrm{MSE}(\boldsymbol{w}) = E_{\boldsymbol{xy}}[(f(\boldsymbol{x}) - \boldsymbol{y})^T\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})]$$

$$\mathrm{MNE}(\boldsymbol{w}) = E_{\boldsymbol{xy}}[\|\boldsymbol{y} - f(\boldsymbol{x})\|],\ \nabla_{\boldsymbol{w}}\,\mathrm{MNE}(\boldsymbol{w}) = E_{\boldsymbol{xy}}\Big[\tfrac{(f(\boldsymbol{x}) - \boldsymbol{y})^T}{\|f(\boldsymbol{x}) - \boldsymbol{y}\|}\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Big]$$

This means we have to just to scale the backpropagation gradient of each term i by dividing with $\|\boldsymbol{y}_i - f(\boldsymbol{x}_i)\|$. This means that for the large errors the effect to gradient is now smaller.