k-NN Sampling for Visualization of Dynamic Data Using LION-tSNE - analysis

Gędłek Paweł
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza
Kraków
gedlek@student.agh.edu.pl

Wójtowicz Patryk
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza
Kraków
wojtowicz@student.agh.edu.pl

ABSTRAKT

TODO

1 Struktura raportu

Contents

1 Struktura raportu **1**

2 Metoda tSNE **2**

2.1 Czym właściwie jest tSNE? 2

2.2 Algorytm tSNE - podstawy matematyczne 2

2.3 Sposób wyboru wariancji 3

2.4 Workflow metody tSNE 3

2.5 Pseudokod metody tSNE 3

3 Metoda LION tSNE **3**

3.1 Pseudokod metody LION tSNE 3

3.2 LION tSNE - IDW 4

3.3 LION tSNE - outlier placement 5

4 kNN sampling **5**

4.1 kNN sampling w LION tSNE 5

4.2 Wybór zbioru treningowego poprzez k-NN sampling 5

4.3 Dodanie nowych punktów do modelu tSNE 5

5 Wyniki projektu **6**

6 Wnioski **6**

2 Metoda tSNE

2.1 Czym właściwie jest tSNE?

Algorytm tSNE(t-Distributed Stochastic Neighbor Embedding) którego autorami są Laurens van der Maaten oraz Geoffrey Hinton bazuje na metodzie SNE, której głównym założeniem jest reprezentacja wielowymiarowych danych w możliwy do zobrazowania dla człowieka dwu lub trzy-wymiarowej przestrzeni. Osiąga się to poprzez modelowanie wysoko wymiarowych obiektów poprzez dwu- lub trzy-wymiarowe punkty w taki sposób, że zbliżone obiekty modelowane są poprzez bliskie sobie punkty, a oddalone obiekty modelowane są poprzez oddalone od siebie punkty z duzym prawdopodobienstwem.

2.2 Algorytm tSNE - podstawy matematyczne

- Algorytm tSNE konwertuje odległości między parami punktów w funkcje rozkładu prawdopodobieństwa określająca podobieństwo pomiędzy parami punktów.
- Rozbieżność między podobieństwem wysoko wymiarowych danych z nisko-wymiarowymi danymi jest mierzona poprzez dywergencje Kullbacka-Leiblera i minimalizowana metoda gradientowa poszukiwania minimum lokalnego

Mamy dany zbiór wejściowy $X = x1, x2...xn$ gdzie dla każdego $x_i \in R^D$ jest D-wymiarowym wektorem. Zbiór ten zostanie przekształcony do postaci $Y = y1, y2...yn$ gdzie każde $y_i \in R^d$ jest d-wymiarowym wektorem oraz $d << D$ (zazwyczaj d = 2 lub 3). Podobieństwo pomiędzy para punktów wejściowych xi oraz xj oznaczamy poprzez pj/i , które jest prawdopodobieństwem wybrania x_j jako sąsiada x_i według funkcji gęstości prawdopodobieństwa na rozkładzie normalnym gdzie x_i stanowi centrum. $p_{j|i}$ definiujemy jako:

$$p_{j|i} = \frac{exp(\frac{-d(x_i,x_j)^2)}{2\sigma_i^2})}{\sum_{k\neq i}^{n} exp(\frac{-d(x_i,x_k)^2)}{2\sigma_i^2})} p_{i|i} = 0 p_{i|j} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{1}$$

gdzie:

$d(x_i, x_j)$ - odległość pomiędzy punktami x_i oraz x_j w oryginalnym wymiarze

σ_i - wariancja dla punktu x_i

Aby otrzymać zbiór wyjściowy Y, losujemy n punktów w docelowym wymiarze i dla każdego z nich wyznaczamy podobna funkcje gęstości prawdopodobieństwa (tym razem jest to rozkład Studenta):

$$q_{ij} = \frac{(1 + d(y_i, y_j)^2)^{-1}}{\sum_{k\neq l}^{n}((1 + d(y_k, y_l)^2)^{-1})} \tag{2}$$

gdzie:

$d(y_i, y_j)$ - odległość pomiędzy punktami yi oraz yj w docelowym wymiarze

W ten sposób otrzymujemy łączone rozkłady gęstości prawdopodobieństwa P i Q dla wszystkich punktów ze zbiorów odpowiednio X i Y. Podobieństwo między nimi (a właściwie dowolnymi 2 rozkładami) określa dywergencja Kullbacka-Leiblera:

$$C = KLDIV(P||Q) = \sum_{i}^{n}\sum_{j}^{n} p_{ij} log\frac{p_{ij}}{q_{ij}} \tag{3}$$

która staje sie nasza funkcja kosztu, która chcemy zminimalizować, robi sie to algorytmem spadku po gradiencie (Gradient Descent). Pochodna cząstkowa C dla y_i to:

$$\frac{\delta C}{\delta y_i} = 4\sum_{i}^{n}(p_{ij} - q_{ij})(y_i - y_j)(1 + d(y_i - y_j)^2)^{-1} \tag{4}$$

2.3 Sposób wyboru wariancji

Wariancje σ_i dla punktu x_i wybiera się na podstawie parametru algorytmu ustawianego przez użytkownika zwanego Perplexity. Definiujemy:

$$Perp(i) = 2^{H(P_i)} H(P_i) = \sum_j p_{j|i} log(\frac{1}{p_{j|i}}) \tag{5}$$

gdzie:

$H(P_i)$ - entropia Shannona dla zmiennej losowej P_i

Dla rozkładu normalnego im większa entropia, tym większa wariancja, tym "grubsze ogony"funkcji dzwonowej, tym większe prawdopodobieństwo wybrania bardziej odległych sąsiadów punktu x_i . Zazwyczaj dla wszystkich punktów $Perp(i)$ ustawiane jest na taka sama wartosc p. Im mniej "gesty"jest nasz zbiór danych tym Perp powinno być większe.

2.4 Workflow metody tSNE

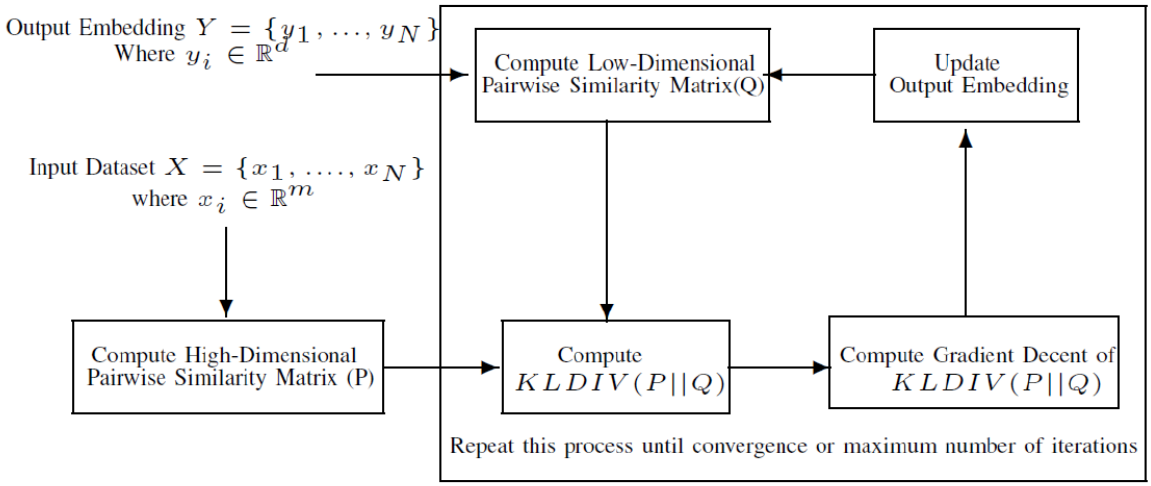

Figure 1: Algorytm tSNE - workflow modelu

2.5 Pseudokod metody tSNE

3 Metoda LION tSNE

Algorytm t-SNE nie odpowiada na pytanie jak dodawać nowe dane (lub wizualizować dynamiczne zmiany danych) do utworzonego modelu. Z pomocą przychodzi metoda LION tSNE (Local Interpolation with Outlier coNtrol t-Distributed Stochastic Neighbor Embedding). Korzysta ona z 2 metod dodawania nowych punktów:

- Dla inlierów czyli punktów które mogą potencjalnie należeć do jakiegoś klastra - Inverse Distance Weight Interpolation (IDW)
- Dla outlierów specjalna heurystyka (Outlier Placement) oszacowania ich pozycji na wizualizacji zapewniajaca odpowiednia odległosc od innych punktów

3.1 Pseudokod metody LION tSNE

Z oryginalnego datasetu wybierane są losowo punkty i na ich podstawie tworzone jest mapowany do niższego wymiaru (za pomocą standardowego algorytmu t-SNE). Następnie:

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Figure 2: Algorytm tSNE - pseudokod

Algorithm 1 LION-tSNE - General Approach

function LION-TSNE(x, p, r_x, $r_c lose$, X_{train}, Y_{train})
  $neighbor\_indices = select\_neighbors\_in\_radius(x, X_{train}, r_x)$
  $X_{neighb} = X_{train}[neighbor\_indices]$
  $Y_{neighb} = Y_{train}[neighbor\_indices]$
  if $len(neighbor\_indices) > 1$ **then**
    $y = local\_IDW\_interpolation(X_{neighb}, Y_{neighb})$
  else if $len(neighbor\_indices) == 1$ **then**
    $y = single\_neighbor\_placement(X_{neighb}, Y_{neighb})$
  else
    $y = outlier\_placement()$
  end ifreturn y
end function

Figure 3: Algorytm tSNE - pseudokod

3.2 LION tSNE - IDW

Gdy algorytm decyduje, że dany punkt jest inlierem, stosowana jest technika Inverse Distance Weight Interpolation (IDW). Dla nowego punktu x jego pozycja F(x) w nowym wymiarze jest wyliczana następująco:

$$F(x) = \sum_{d(x-x_i) \leq r_x} w_i(x) y_i \tag{6}$$

,gdzie $w_i(x) = \frac{d(x-x_j)^{-p}}{\sum_{d(x-x_j) \leq r_x} d(x-x_j)^{-p}}$

r_x - promien odległości dla lokalnego sąsiedztwa w którym jest dodawany nowy punkt, parametr wywołania

p - parametr wywołania

Zauważmy, że gdy $x \to x_i$ to $d(x - x_i)^{-p} \to \infty$ i $w_i(x) \to \infty$ i $F(x) \to y_i$

3.3 LION tSNE - outlier placement

Główna idea outlier placementu jest nastepujaca: jesli x jest outlierem to odpowiadająca jej po zmapowaniu wartość y powinna także być zwizualizowana jako outlier. Aby odnaleźć takie wartości, musimy znaleźć położenie y, takie że nie ma żadnych sąsiadów w promieniu ry . Promień ry jest jednym z parametrów algorytmu i jeśli zostanie wybrana zbyt duza wartość, wtedy z racji na duże odległości między punktami, zmniejsza się czytelność wykresu natomiast jesli promien zostanie wybrany zbyt mały klastry i wartości odstające mogą być nierozróżnialne. Dlatego też wartość ry może być wyznaczona na podstawie pewnego percentylu rozkładu odległości najbliższych sąsiadów w przestrzeni y (np. 95-ty lub 99-ty percentyl).

4 kNN sampling

4.1 kNN sampling w LION tSNE

Autorzy artykułu zasugerowali, że sposób próbkowania danych zastosowany w tSNE jest niewystarczający chociażby przy dynamicznie zmieniających się danych. Jednocześnie przedstawili kilka kroków potrzebnych do zrealizowania idei k-NN samplingu.

1. Czyszczenie danych za pomocą eliminacji redundantnych punktów i zainicjalizowanie pustych zmiennych odpowiednimi wartościami.
2. Dobór właściwego zbioru treningowego poprzez k-NN sampling.
3. Rzutowanie zbioru treningowego na nisko wymiarowa przestrzeń oraz dodanie nowych punktów do modelu tSNE.
4. Dla nowych danych interpolacja ich do istniejącego modelu za pomocą LION-tSNE
5. Wyliczenie precyzji k-NN samplingu dla całego zbioru danych.

4.2 Wybór zbioru treningowego poprzez k-NN sampling

Zaproponowana idea k-NN samplingu opiera się na wyliczeniu Nearest Neighbor score (NNscore) oraz Mutual Nearest Neighbor score (MNNscore). Mamy dany graf skierowany $G = (V, E)$, gdzie krawędź $E(v1, v2)$ oznacza, ze v_2 jest sąsiadem v_1, natomiast sąsiedztwo v_1 oznaczamy jako N_v1. Stopień wychodzący każdego wierzchołka jest równy k, a stopień wchodzący zalezy od wartosci współczynnika sąsiedztwa innych wierzchołków. W celu wyznaczenia optymalnego zbioru treningowego dobieramy odpowiednie k oraz zbiór punktów wejściowych mapujemy na wierzchołki grafu k-NN.

Wyznaczamy NN score, który odpowiada stopniu wchodzącemu wierzchołka x_i:

$$NNscore(x_i) = |\{x_j | x_i \in N_{xj}\}| \forall_{j \neq i} x_j \in X \tag{7}$$

Gdzie X jest zbiorem danych wejściowych a N_{xj} sąsiedztwem x_j. Następnie wyliczamy MNNscore, który dla x_i jest równy co najwyżej k:

$$MNNscore(x_i) = |\{x_j | x_i \in N_{xj} \wedge x_i | x_j \in N_{xi}\}| \forall_{j \neq i} x_j \in X \tag{8}$$

Ostatecznie dobór punktu x_i jako próbki treningowej oraz jego sąsiedztwo jest dany jako:

$$train\_sample = first\_index\{argmax_{x_i \in X}\{NNscore(x_i)\} \cap_{x_i \in X} \{MNNscore(x_i)\}\} \tag{9}$$

4.3 Dodanie nowych punktów do modelu tSNE

W LION-tSNE stosujemy IDW i Outlier Placement. Nowe dane mogą być dodawane do modelu tSNE na dwa sposoby, na podstawie wyliczonych parametrów rxNN, ryNN oraz rclose . Wartość rxNN oznacza minimalny promien zbioru wejściowego, co z pomocą pewnej heurystyki pozwala na wskazanie czy dany punkt jest wartością właściwą czy

odstająca. Parametr rclose pozwala na zidentyfikowanie próbek odstających powiązanych z wartościami odstającymi znajdującymi się w modelu tSNE. Natomiast ryNN wyznacza minimalna odległość pomiędzy punktami ze zbioru wejściowego, a próbkami odstającymi oraz dla wartości odstających pomiedzy nimi.

5 Wyniki projektu

TODO

6 Wnioski

TODO

Źródła

[1] Bheekya Dharamsotu ; K. Swarupa Rani ; Salman Abdul Moiz ; C. Raghavendra Rao Paper: k-NN Sampling for Visualization of Dynamic Data Using LION-tSNE. *https://ieeexplore.ieee.org/abstract/document/8990391*