МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
КАФЕДРА №51

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

		Веселов А.И
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

ВИЗУАЛИЗАЦИЯ С ИСПОЛЬЗОВАНИЕМ МЕТОДА t-SNE

по курсу: МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.	5711М		Пятаков В.С.
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2018

Цель работы

Анализ и проектирование метода нелинейного понижения размерности данных t-SNE.

Задание

1. Разобраться с теоретической базой методов визуализации данных ;
2. Реализовать метод визуализации t-SNE ;
3. Реализовать метод визуализации t-SNE с помощью готовых библиотек;
4. Оценить проектируемый метод с помощью визуальной оценки построенных зависимостей сравнивая результаты с готовым методом на основе встроенных библиотек;
5. Сделать выводы по полученным результатам.

Порядок выполнения работы

Визуализация данных

Объемы и сложность данных постоянно растут. В результате, существенно увеличивается и их размерность.

Часто бывает, что пространство малой размерности вложено в пространство большой размерности сложным нелинейным образом. Эта структура, скрытая в данных, может быть восстановлена только с помощью специальных математических методов.

К ним относится подраздел машинного обучения как нелинейное уменьшение размерности .

Допустим, у нас имеется набор изображений 8x8 пикселей (рисунок 1) и таких изображений 1000. Одна такая картинка, будет находится в пространстве размерности D=64, а человек может представлять объекты только в пространствах размерности $D \le 3$.

Точка данных – это точка x_i в исходном пространстве данных R^D , где D = 64 – размерность пространства данных. Всего N = 1000 точек данных.

Точка отображения – это точка y_i в пространстве отображения R^d, где $d \le 3$- размерность пространства отображения. Это пространство будет содержать целевое представление набора данных. Между точками данных и точками отображения имеет место *биекция* : каждая точка отображения представляет одно исходное изображение.

Используя методы визуализации, мы хотим сохранить структуру данных. Более конкретно, если две точки данных расположены близко друг к другу, мы хотим, чтобы две соответствующие точки отображения также располагались близко друг к другу

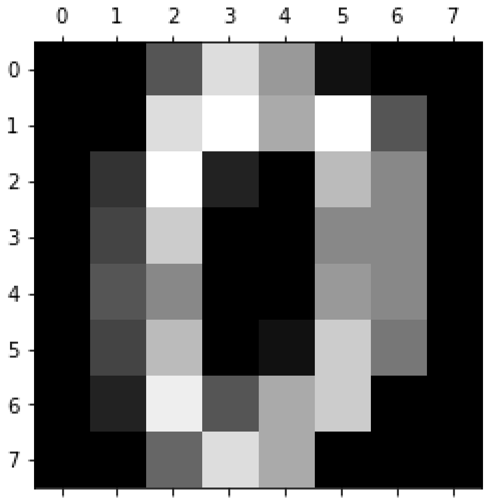

Рисунок 1 - изображение размером 8x8

Метод визуализации данных SNE\

У нас есть набор данных с точками, описываемыми многомерной переменной с размерностью пространства существенно больше трех. Необходимо получить новую переменную, существующую в двумерном или трехмерном пространстве, которая бы в максимальной степени сохраняла структуру и закономерности в исходных данных

В методе SNE используется вероятностный подход к решению данной задачи. В начале работы алгоритм делает преобразование многомерной евклидовой дистанции между точками в исходном пространстве в условные вероятности, отражающие сходство точек.

В математическом виде можем это записать как:

$$p_{j|i} = \frac{\exp\left(\frac{-\| x_i - x_j \|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\| x_i - x_k \|^2}{2\sigma_i^2}\right)} \quad (1)$$

Эта формула показывает, насколько точка x_j близка к точке x_i при гауссовом распределении вокруг x_i с заданным отклонением σ. Сигма будет различной для каждой точки. Она выбирается так, чтобы точки в областях с большей плотностью имели меньшую дисперсию.

Покажем это на качественном уровне. Рассмотрим два примера, когда точки в пространстве находятся близко друг к другу, и следовательно имеют высокую плотность и ситуацию напротив, с маленькой плотностью, когда точки удалены.

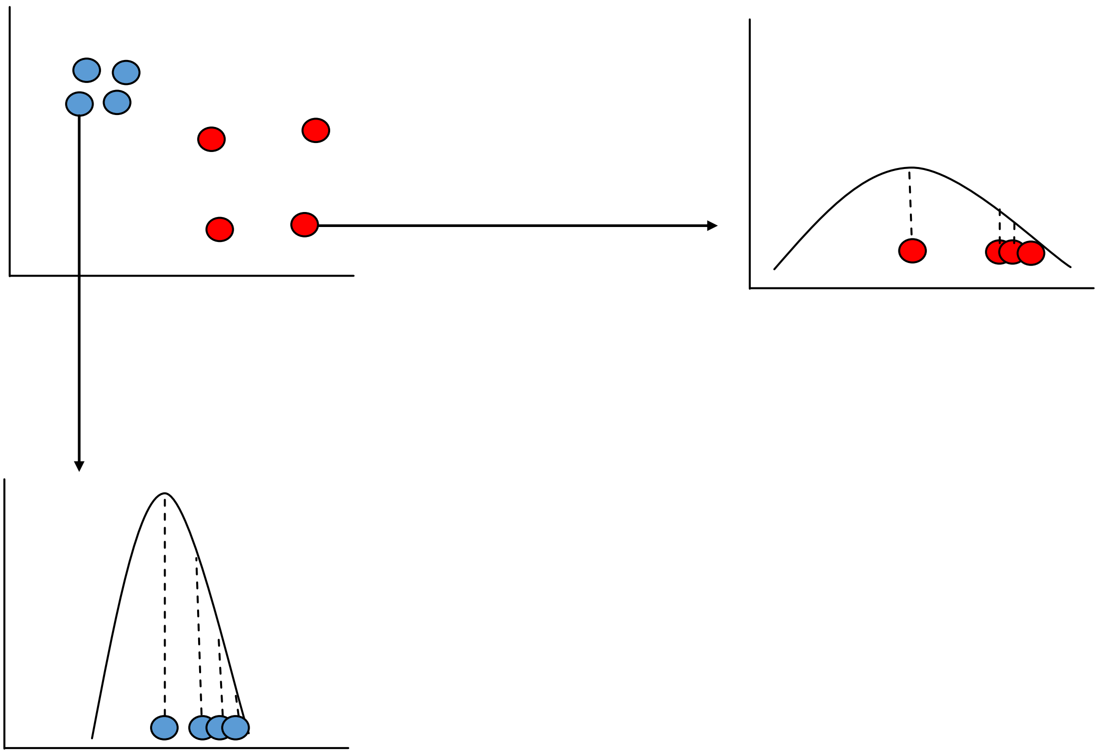

Рисунок 2- зависимость дисперсии Гауссовского ядра от областей плотности точек

Из рисунка 2 можно сделать вывод о том, что Гауссовский колокол для областей с большей плотностью имеет маленькую дисперсию, и следовательно он уже и выше. В областях с маленькой плотностью , Гауссовский колокол напротив широкий и маленький.

Для того, чтобы правильно найти σ, используется оценка перплексии , которая в свою очередь, может быть записана как:

$$Perp(P_i) = 2^{H(P_i)},$$

где $H(P_i)$ –энтропия Шеннона в битах, которую можно найти исходя из формулы:

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \qquad (2)$$

В данном случае перплексия может быть интерпретирована, как сглаженная оценка эффективного количества «соседей» для точки x_i, то есть сколько примерно соседей, будет у точки x_i Отметим также, что σ определяется для каждой пары точек x_i и x_j.

Так как, задачей визуализации данных является перенос точек из пространства большой размерности в малое, необходимо установить правило, по которому будет производится перенос. В данном алгоритме, мы будем соотносить условные вероятности из исходного пространства и пространства отображения.

Для это нам необходимо оценить условную вероятность точек отображения, которые обозначим как y_i и y_j. Тогда условную вероятность можно оценить используя формулу 1, с стандартным отклонением $1/\sqrt{2}$. Она будет иметь следующий вид:

$$q_{j|i} = \frac{\exp(-\| y_i - y_j \|^2)}{\sum_{k \neq i} \exp(-\| y_i - y_k \|^2)}$$

Если точки отображения y_i и y_j корректно моделируют сходство между исходными точками высокой размерности x_i и x_j, то соответствующие условные вероятности $p_{j|i}$ и $q_{j|i}$ будут эквивалентны.

Как оценить качество переноса? В данном методе для оценки качества, с которым $q_{j|i}$ отражает $p_{j|i}$ используется дивергенция Кульбака-Лейблера. Алгоритм SNE минимизирует сумму таких расстояний для всех точек отображения при помощи градиентного спуска. Функция потерь с использованием дивергенция Кульбака-Лейблера для данного метода будет определяться, как:

$$Cost = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \qquad (3)$$

Градиент, а именно частная производная по переменной y_i примет следующий вид:

$$\frac{\partial Cost}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Авторы данного метода [2] предлагают следующую физическую аналогию для процесса оптимизации: Представим, что все точки отображения соединены пружинами. Жесткость пружины, соединяющей точки i и j, зависит от разности между сходством двух точек данных и сходством двух точек отображения, т.е. $p_{j|i} - q_{j|i}$. Теперь мы позволим системе изменяться согласно законам физики. Если расстояние между двумя точками отображения большое, а между точками данных малое, – точки отображения притягиваются. Если наоборот – точки отображения отталкиваются. Целевое отображение будет получено при достижении равновесия. Алгоритмически, поиск равновесия предлагается делать с учетом моментов:

$$Y^{(t)} = Y^{(t)-1} + \eta \frac{\partial Cost}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)}),$$

где η – параметр, определяющий скорость обучения ,а α – момент.

Метод визуализации данных t-SNE

Алгоритм t-SNE, который также относят к методам множественного обучения признаков, был опубликован в 2008 году [1] голландским исследователем Лоуренсом ван дер Маатеном и Джеффри Хинтоном.

Метод визуализации данных t-SNE имеет ряд отличий от классического SNE. В первую очередь у данного метода симметричная форма сходства в многомерном пространстве и более простой градиент, а также вместо распределения гаусса для пространства отображения используется распределение Стьюдента (рисунок 3), смысл которого в "тяжелых" хвостах, благодаря которым облегчается оптимизация алгоритма и решается проблема скученности.

Функцию потерь можно упростить, заменив операцию минимизации сумм дивергенций Кульбака-Лейблера между условными вероятностями $p_{j|i}$ и $q_{j|i}$ на операцию минимизации одиночной дивергенции между безусловной вероятностью P в многомерном пространстве и безусловной вероятностью Q в пространстве отображения. Тогда функцию потерь можно переписать в следующем виде:

$$Cost = KL(P \| Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

где p_{ij}, $q_{ij} = 0$, $p_{ij} = p_{jij}$, $q_{ij} = q_{jij}$ для всех i и j.

Для оценки безусловной вероятности p_{ij} используется следующая формула:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}.$$

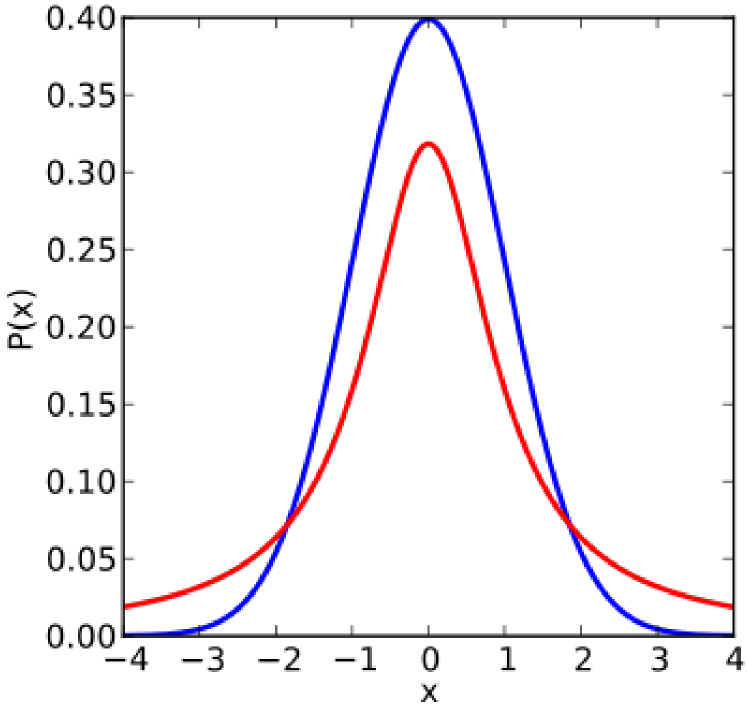

Рисунок 3 - плотность t-распределения (красная) с 1 степенью свободы и распределения гаусса (синяя)

Проблема скученности заключается в том, что расстояние между двумя точками в пространстве отображения, соответствующими двум среднеудаленным точкам в многомерном пространстве, должно быть существенно больше, нежели расстояние, которое позволяет получить гауссово распределение. Проблему решают хвосты Стьюдента.

Совместная вероятность для пространства отображения в этом случае будет определяться как:

$$q_{ij} = \frac{(1 + \| y_i - y_j \|^2)^{-1}}{\sum_{k \neq l} (1 + \| y_k - y_l \|^2)^{-1}} \qquad (4)$$

Тогда можно переписать градиент в следующем виде:

$$\frac{\partial Cost}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \| y_i - y_j \|^2)^{-1} \qquad (5)$$

Алгоритм работы метода t-SNE

В упрощенном виде алгоритм t-SNE можно записать в виде псевдокода [1]:

Исходные данные: набор данных X = {x1, x2, …, xn};
Параметр дивергенции: перплексия;
Параметры оптимизации: количество итераций T, скорость обучения η, момент α(t);
На выходе получим: представление данных Y(T) = {y1, y2, …, yn};
begin
вычислить попарное сходство pj|i с перплексией (используя формулу 1)
установить pij = (pj|i + pi|j)/2n
инициализировать Y(0) = {y1, y2, …, yn} точками нормального распределения (mean=0, sd=1e-4)
for t = 1 **to** T **do**
вычислить сходство точек в пространстве отображения qij (по формуле 4)
вычислить градиент δCost/δy (по формуле 5)
установить Y(t) = Y(t-1) + ηδCost/δy + α(t)(Y(t-1) - Y(t-2))
end
end

Распределение Стьюдента

Так как в методе t-SNE , для точек отображения было выбрано распределение Стьюдента, в отличие от точек исходного пространства, там распределение гауссово, нужно пояснить, почему оно так.

Известно, что объем N-мерного шара радиуса r пропорционален r^N. При больших N, если выбирать случайные точки в шаре, большинство точек будет располагаться около поверхности, и очень небольшое количество – около центра.

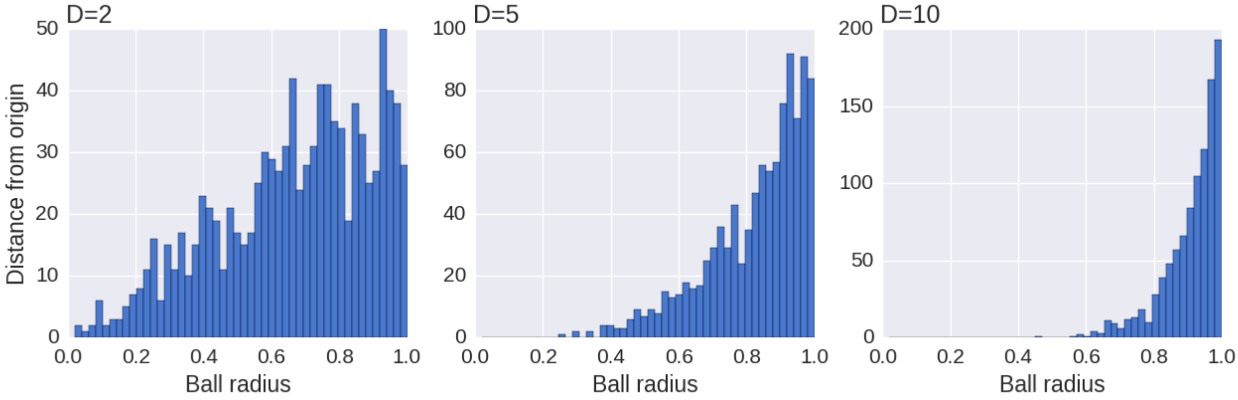

Рисунок 4 - распределение расстояний точек при различном количестве измерений

При уменьшении размерности набора данных, если использовать гауссово распределение для точек данных и точек отображения, мы получим *дисбаланс*в распределении расстояний для соседей точек. Это объясняется тем, что распределение расстояний существенно отличается для пространства большой размерности и для пространства малой размерности. Тем не менее, алгоритм пытается воспроизвести одинаковые расстояния в обоих пространствах. Этот дисбаланс создает избыток сил притяжения, что иногда приводит к неудачному отображению

Алгоритм t-SNE решает эту проблему, используя распределение Стьюдента с одной степенью свободы (или распределение Коши) для точек отображения. В отличие от гауссова распределения, это распределение имеет значительно более «тяжелый» хвост, что позволяет *компенсировать* дисбаланс. Для данного сходства между двумя точками данных, две соответствующие точки отображения должны находиться намного дальше друг от друга, чтобы их сходство соответствовало сходству точек данных. Это можно увидеть на рисунке 5.

Использование этого распределения обеспечивает более эффективную визуализацию данных, при которой группы точек более отчетливо отделены друг от друга.

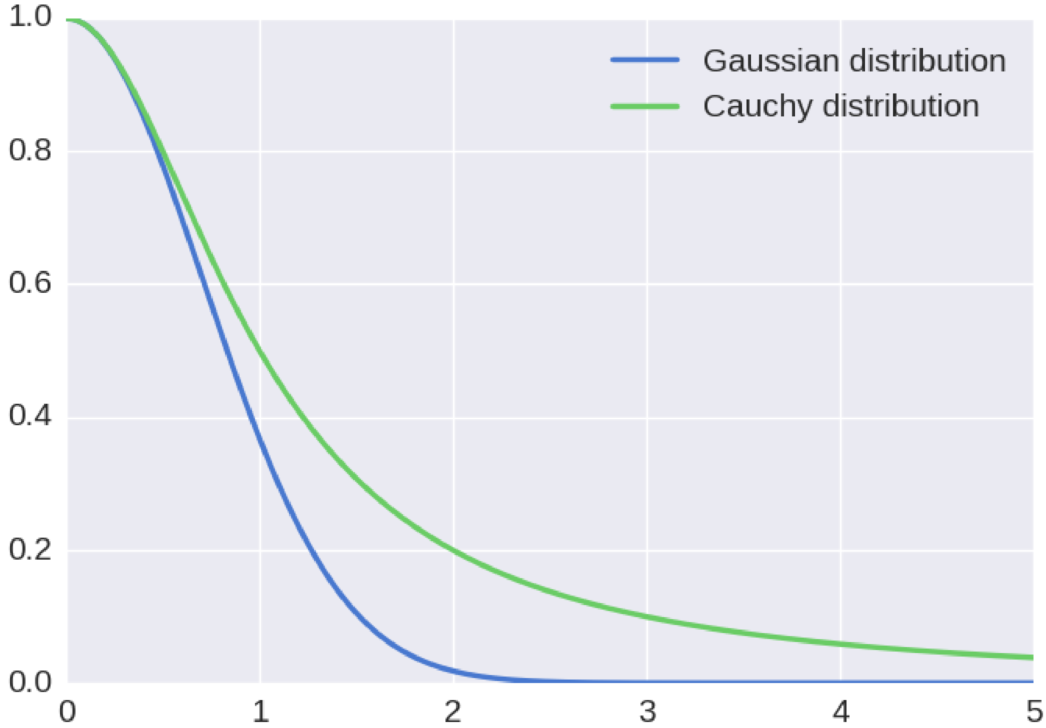

рисунок 5 - распределение Гаусса и Стьюдента

Результаты моделирования

В качестве проверки работы моделирующей программы, будет рассмотрен пример ее работы на наборе данных **MNIST**. Одна из частей этого набора содержит 60000 изображений рукописных цифр от 0 до 9, 28x28 пикселей.

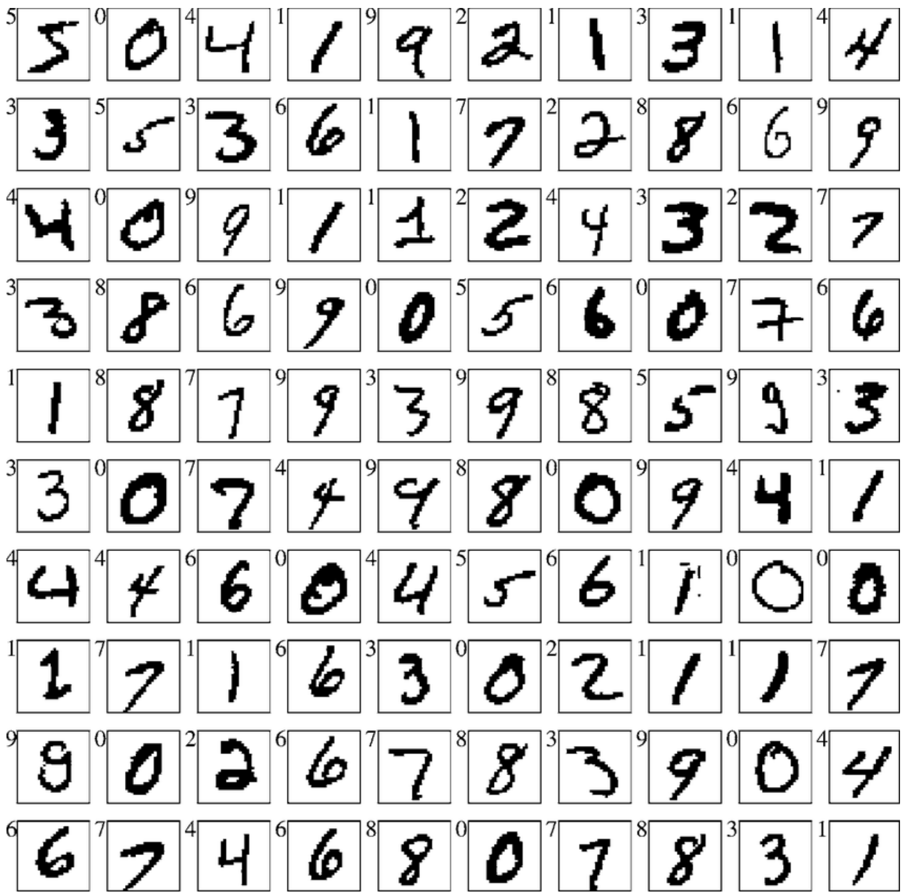

Рисунок 6 - набор данных MNIST

Построим отображения набора из 5000 изображений с параметром перплексии 35 и 50 и сравним полученные результаты с выходом программы использующей встроенные библиотеки.

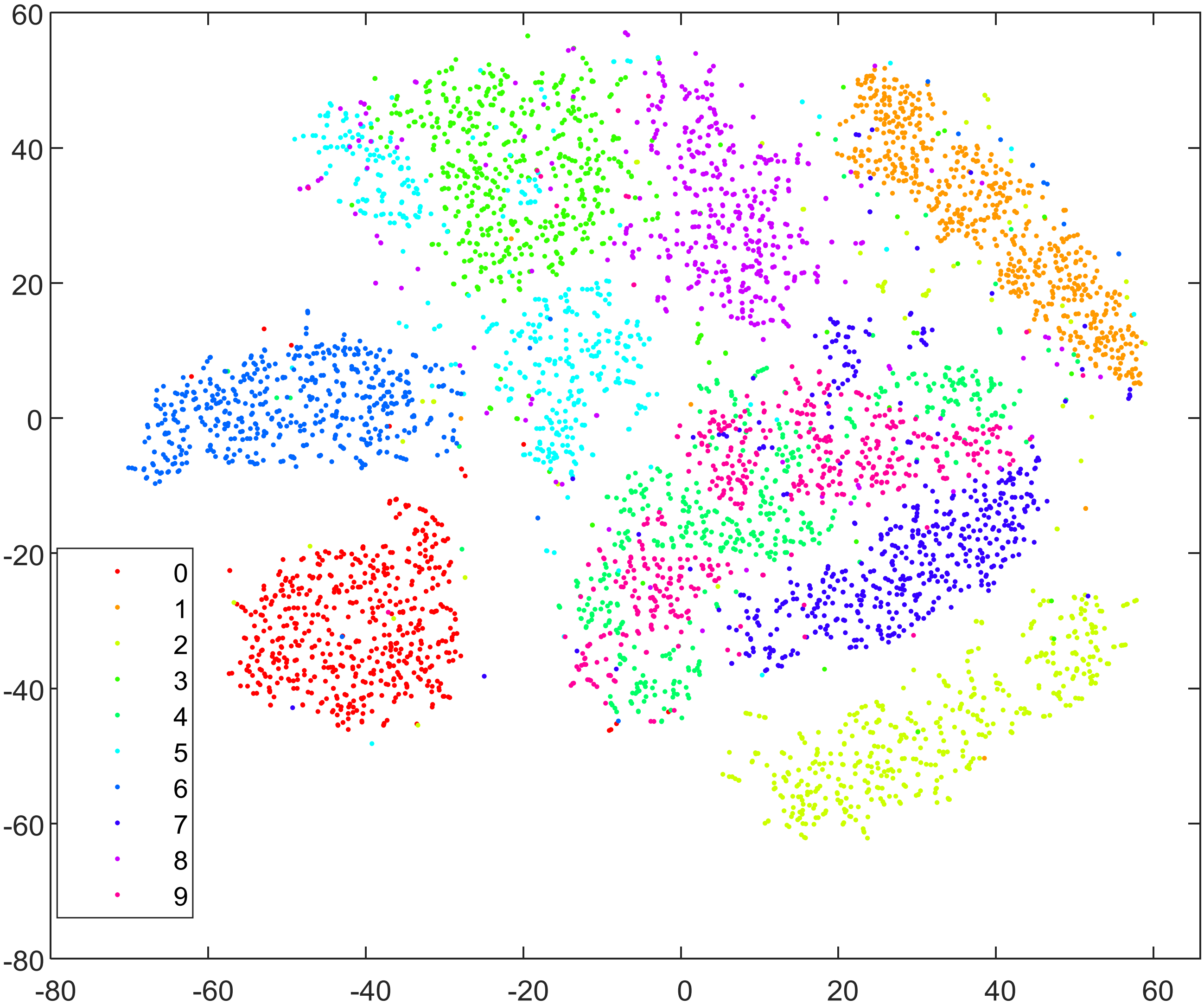

Рисунок 7 - результат работы программы моделирующий алгоритм t-SNE на наборе данных MNIST для перплексии 50

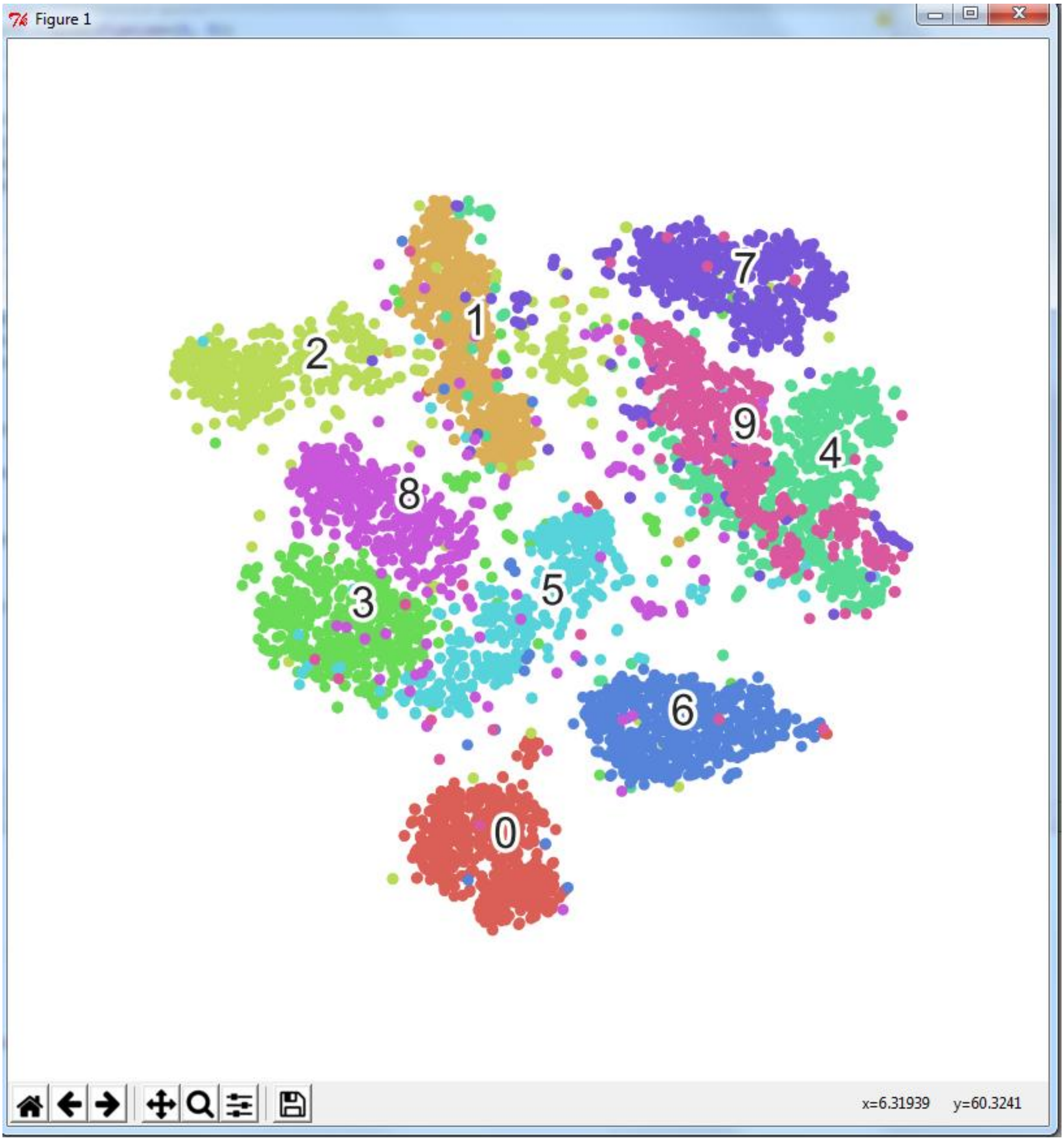

Рисунок 8 - результат работы программы моделирующий алгоритм t-SNE при помощи готовых библиотек на наборе данных MNIST для перплексии 50

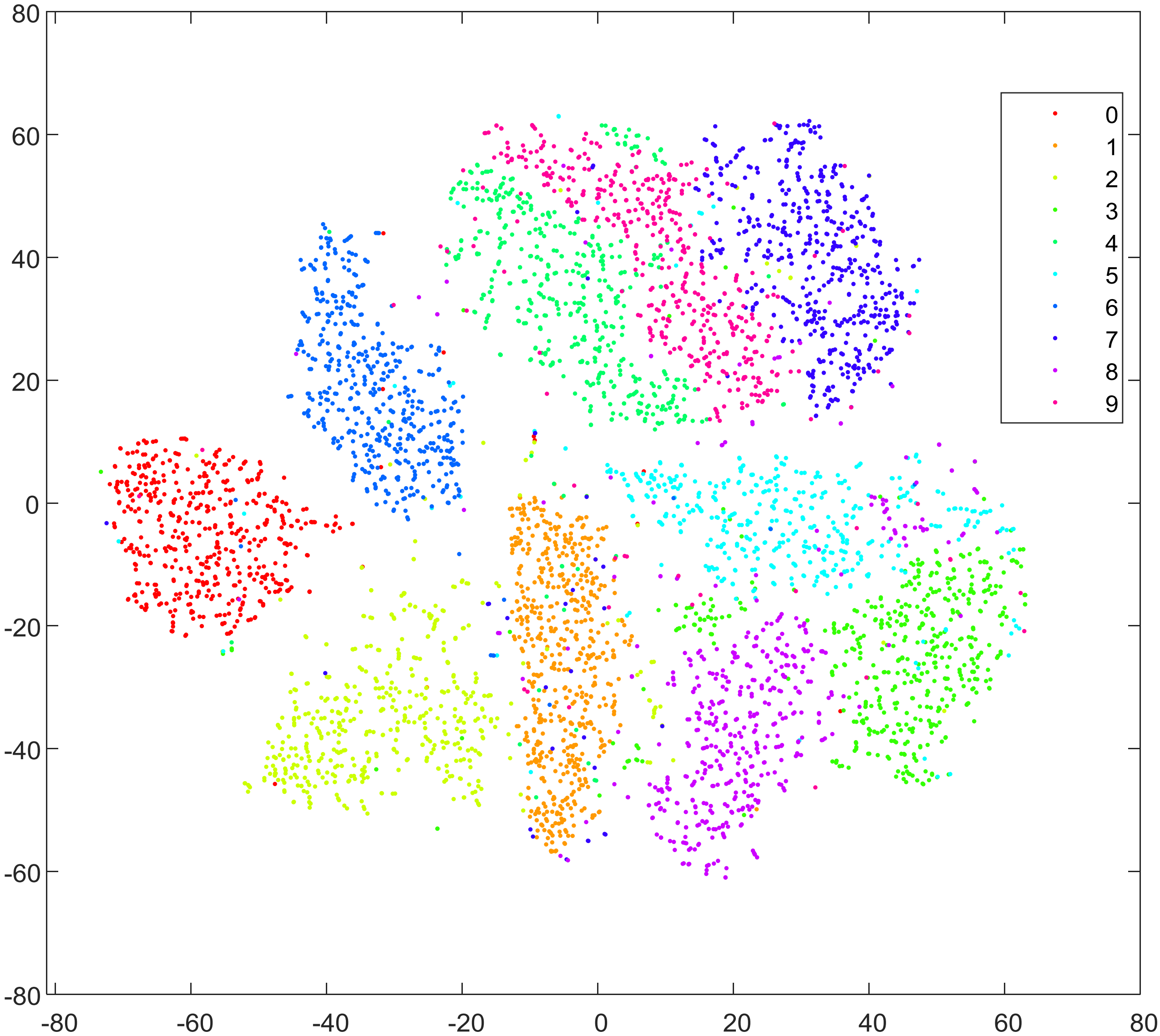

Рисунок 9 - результат работы программы моделирующий алгоритм t-SNE на наборе данных MNIST для перплексии 35

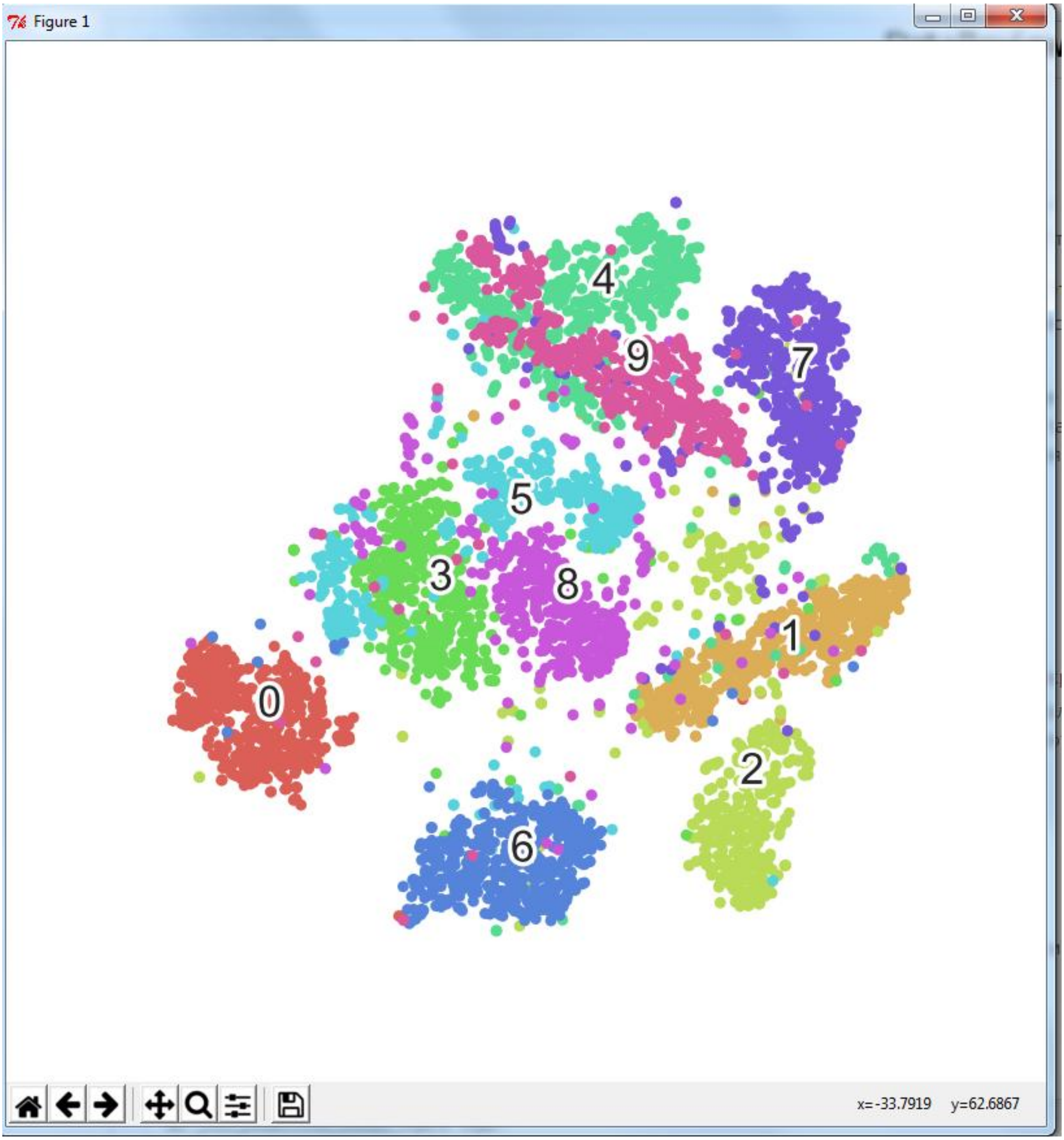

Рисунок 10 - результат работы программы моделирующий алгоритм t-SNE при помощи готовых библиотек на наборе данных MNIST для перплексии 35

Что именно делает алгоритм визуализации можно пронаблюдать, если посмореть процесс понижения размерности в пространствах, которые для нас удобны.

Для этого возьмем несколько классов набора данных и построим их отображение сначала в 3-х мерном пространстве, после применим алгоритм и перенесем в двумерное пространство, и в коце в одномерное. Априори предполагаем, что если алгоритм работет верно, то точки в пространстве отображения, которые принадлежат одному классу, должны собираться друг возле друга и наоборот.

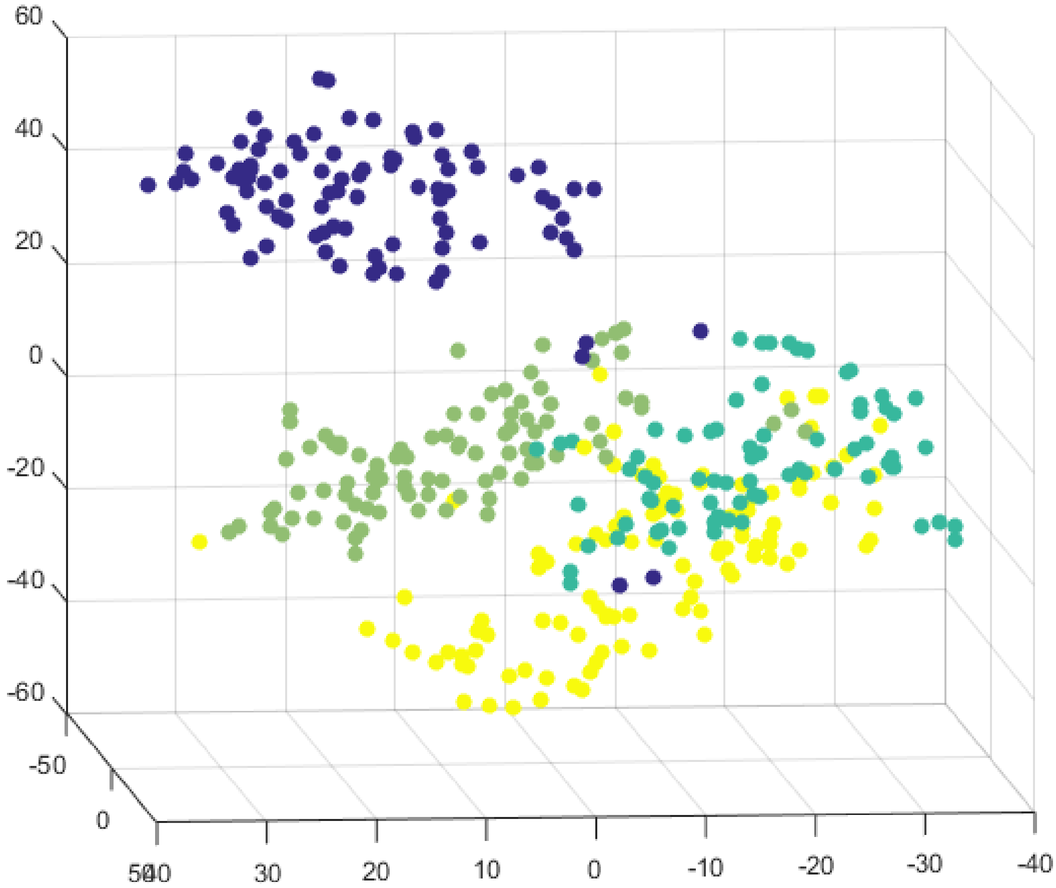

Рисунок 11 - результат работы программы моделирующий алгоритм t-SNE на наборе данных MNIST для 4 классов в пространстве размерности 3

Теперь перенесем наши точки в пространство меньшей размерности.

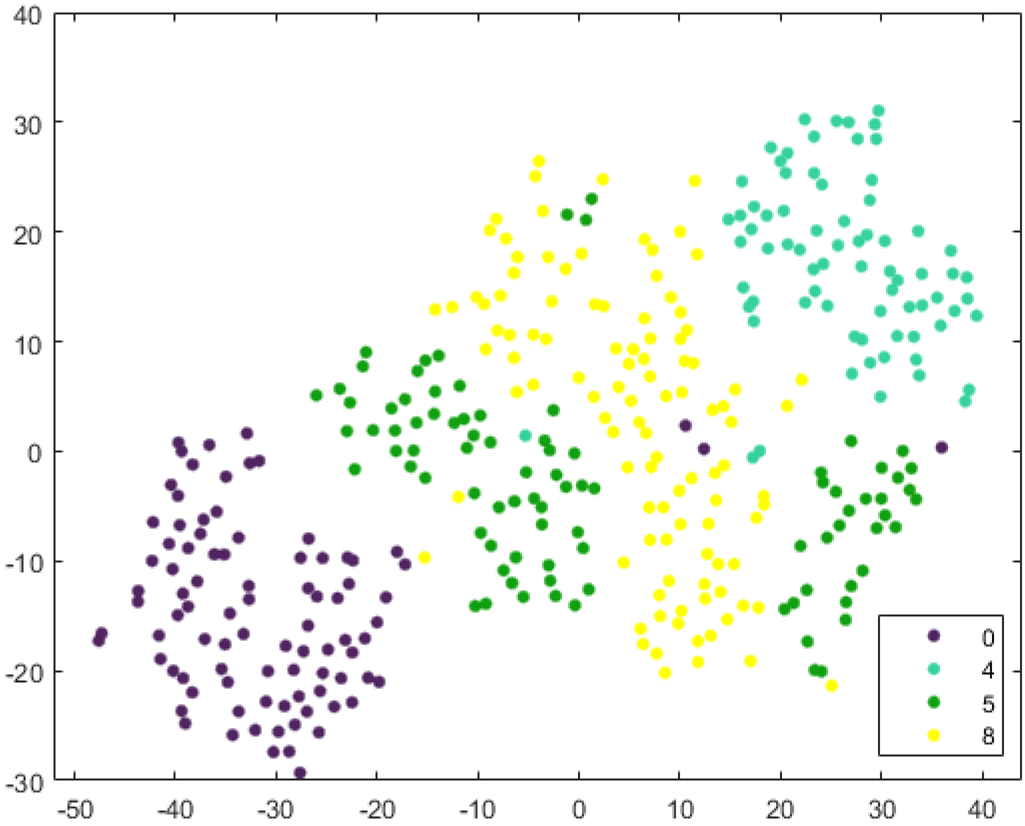

Рисунок 12 - результат работы программы моделирующий алгоритм t-SNE на наборе данных MNIST для 4 классов в пространстве размерности 2

Ну и на последок в одномерное пространство.

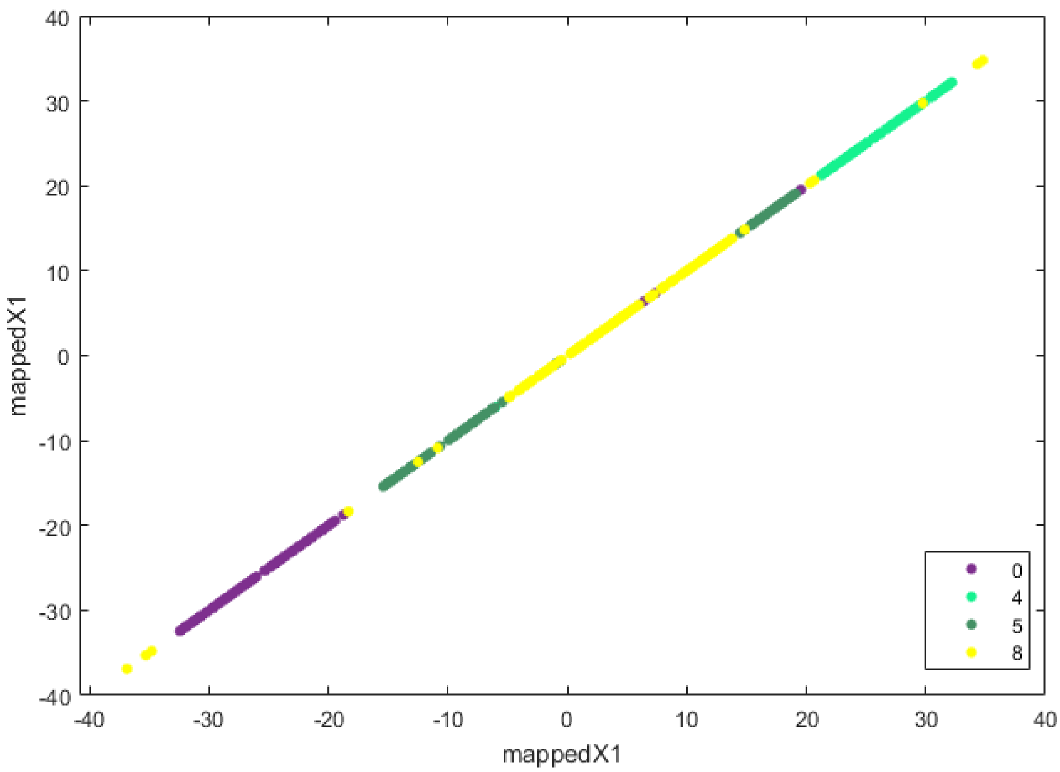

Рисунок 13 - результат работы программы моделирующий алгоритм t-SNE на наборе данных MNIST для 4 классов в пространстве размерности 1

Из полученных результатов, можно сделать вывод о том что алгоритм работает правильно и точки с одинаковыми классами сближаются друг с другом, и отдаляются от других классов.

Пронаблюдав перенос точек из большего пространства в меньшее, можно лучше понять процесс визуализации и понижения размерности.

Вывод

В результате проектирования курсового проекта было проделано следующее:

1. Был проведен анализ и проектирование метода нелинейного понижения размерности данных t-SNE;
2. Ознакомились с теоретической базой методов визуализации данных ;
3. Реализован метод визуализации t-SNE ;
4. Реализован метод визуализации t-SNE с помощью готовых библиотек;
5. Построены отображения набора данных MNIST для обоих реализаций алгоритма;
6. По построенным зависимостям можно сделать вывод о том, что моделирующая программа работает правильно, так как результаты совпадают визуально, с результатами, которые дает программа на основе готовых библиотек;
7. Так же было проведен небольшой анализ переноса из 3-х мерного пространства в одномерное, для лучшего понимания работы алгоритма , а так же наблюдением за обнаружением и воспроизведением скрытых структур данных.
8. Из всего вышеуказанного можно сделать вывод о том, что алгоритм t-SNE обеспечивает эффективный метод визуализации сложных наборов данных. Он успешно обнаруживает скрытые структуры в данных, демонстрирует группы и компенсирует нелинейные отклонения по измерениям.

Список использованной литературы

1. L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008
2. G.E. Hinton and S.T. Roweis. Stochastic Neighbor Embedding. In Advances in Neural Information. Processing Systems, volume 15, pages 833–840, Cambridge, MA, USA, 2002. The MIT Press
3. https://en.wikipedia.org/wiki/Perplexity
4. https://habr.com/company/wunderfund/blog/326750/
5. https://statquest.org/2017/09/18/statquest-t-sne-clearly-explained/
6. https://habr.com/post/267041/
7. https://github.com/khmelkoff/Xtsne/blob/master/tsne.R
8. https://en.wikipedia.org/wiki/Binary_search_algorithm
9. https://ru.coursera.org/learn/unsupervised-learning/lecture/Bn22S/mietod-t-sne
10. http://itas2016.iitp.ru/pdf/1570303407.pdf
11. http://www.machinelearning.ru/wiki/images/e/e0/Problem_statement_dim_reduce.pdf
12. http://nbviewer.jupyter.org/urls/gist.githubusercontent.com/AlexanderFabisch/1a0c648de22eff4a2a3e/raw/59d5bc5ed8f8bfd9ff1f7faa749d1b095aa97d5a/t-SNE.ipynb
13. http://datareview.info/article/algoritm-t-sne-illyustrirovannyiy-vvodnyiy-kurs/
14. https://lvdmaaten.github.io/publications/papers/AISTATS_2009.pdf
15. https://ru.coursera.org/learn/unsupervised-learning/lecture/e72bH/mietod-ghlavnykh-komponient-rieshieniie
16. https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b
17. https://lvdmaaten.github.io/tsne/
18. https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf
19. https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/

ПРИЛОЖЕНИЕ

Листинг программы в среде matlab

```
clear all;
close all;
clc;

% загружаем и обрабатываем исходные данные
load 'C:\mnist\mnist_train'

n=5000;% количество элементов в выборке

ind = randperm(size(train_X, 1));%рандомизируем нашу выборку
train_X = train_X(ind(1:n),:);
train_labels = train_labels(ind(1:n))-1;

%% нормализуем входные данные, вычитая мин, деля на максимум и вычитая среднее

 train_X = (train_X - min(train_X(:)))/ max(train_X(:));

      for i=1: size(train_X,1)%вычитаем их каждой строки среднее

               train_X(i,:)=train_X(i,:)-mean(train_X, 1);

      end

%% входные параметры

Dim_space = 2;    % размер отображающего пространства
perplexity = 50; % перплексия
T = 500;         % количество итераций
mu = 500;         % скорость обучения
momentum = 0.5;  % момент(инерция)
numberplot=1;    %выводить мне график или нет
%% вычисление матрицы условных вероятнотей P

      P = give_condi_P(train_X, perplexity, 1e-5);

%% понижение размерности с t-SNE

      ydata = t_sne_algoritm(P, Dim_space,momentum,mu,T);

%% построение графиков
if Dim_space == 2

      figure(1)
      gscatter(ydata(:,1), ydata(:,2), train_labels(:));
if numberplot==1
      figure(2)
      plotin2D( ydata,train_labels )
end

else

      figure(1)
      scatter3(ydata(:,1), ydata(:,2), ydata(:,3), 40, train_labels, 'filled');

      figure(2)
      plotin3d( ydata,train_labels );

end
```

Функция для получения матрицы условных вероятностей

```
function [P, gamma] = give_condi_P(X, u, tol)


    %% переменные
    n = size(X, 1);                         % количество состояний в матрице условных
вероятностей
    P = zeros(n, n);                        % матрица вероятностей
    gamma = ones(n, 1);                      % что то про точность, наверно сигма
    logU = log(u);                          % log of perplexity (= entropy)

    %% вычисление попарных растояний

    sum_X = sum(X .^ 2, 2);
      Matr=-2 * X * X';% так как мы берем гамму а не сигму , то 2 в числитель перейдет

     for i=1: size(sum_X,1)
          A(i,:)=sum_X'+ Matr(i,:);
     end

      for i=1: size(sum_X,1)
          D(:,i)=sum_X+ A(:,i);%матрица расстояний евклида
      end

    %% Пройдем по всем точкам

    for i=1:n


        % мин и макс значения для ядра гауса
        gammamin = -Inf;
        gammamax = Inf;

        % Вычисления ядра и ентропии для текущей гаммы
        Di = D(i, [1:i-1 i+1:end]);
        [PerInc, thisP] = perl_comp(Di, gamma(i));

        % В предалах доспустимых значений перплексия или нет
        Perldiff = PerInc - logU;
        tries = 0;
        while abs(Perldiff) > tol && tries < 50

            % бинарный посик гаммы
            if Perldiff > 0
                gammamin = gamma(i);
                if isinf(gammamax)
                    gamma(i) = gamma(i) * 2;
                else
                    gamma(i) = (gamma(i) + gammamax) / 2;
                end
            else
                gammamax = gamma(i);
                if isinf(gammamin)
                    gamma(i) = gamma(i) / 2;
                else
                    gamma(i) = (gamma(i) + gammamin) / 2;
                end
            end

            % Пересчитаем значения перплексии и вероятности
            [PerInc, thisP] = perl_comp(Di, gamma(i));
            Perldiff = PerInc - logU;
            tries = tries + 1;
        end

        % Запишем итоговые условные вероятности
        P(i, [1:i - 1, i + 1:end]) = thisP;
    end
end
```

Функция моделирующая алгоритм t-SNE

```
function ydata = t_sne_algoritm(P, no_dims,momentum,mu,T)
%% переменные

    n = size(P, 1);                                     % количество точек
    final_momentum = 0.8;
    momentum_iter_change = 250;
    stop_lying_iter = 100;                              % до какого будем делать
раннее гиперусиление                                    % скорость обучения
    min_gain = .01;                                     % начальное уселение для
дельта-бар-дельта


    P(1:n + 1:end) = 0;    % так как условные в i|i=0 то по диалогонали встраеваем нули
    P = 0.5 * (P + P');                                 % формула для вычисления
без.вероятности как P=(p(j|i)+p(i|j))/2n
    P = max(P ./ sum(P(:)), realmin);                   %P=(p(j|i)+p(i|j))/2n,
n=sum(P(:))
    divKL = sum(P(:) .* log(P(:)));                     % функция потерь дивергениция
Кульбака-Лейблера

     P = P * 4;                                      % раннее гиперусиление, чтобы
лучше искать глобальные минимумы


%% инициализиурем отображение

        ydata = .0001 * randn(n, no_dims);% рандомим наши отображения с помощью
нормального распреелениия

    y_incs  = zeros(size(ydata));% приращение отображений
    gains = ones(size(ydata));% усиление для ускорения градиента

%% погнали

    for iter=1:T

        % вычисоение Q безусловной вероятнсти по формуле 4
        sum_ydata = sum(ydata .^ 2, 2);

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        Matr=-2 * (ydata * ydata');

        for i=1: size(ydata,1)
            A(i,:)=sum_ydata'+ Matr(i,:);
        end

         for i=1: size(ydata,1)
            AA(:,i)=sum_ydata+ A(:,i);
          end

        t_rasp=1./(1+AA); % в итоге получим распределение стъюдента



        t_rasp(1:n+1:end) = 0;
        Q = max(t_rasp ./ sum(t_rasp(:)), realmin);                  %получили Q

        % вычисление градиента по формуле 5
        L = (P - Q) .* t_rasp;% (P-Q)*(1+||yi-yj]]^2)^-1
        y_grads = 4 * (diag(sum(L, 1)) - L) * ydata;% ФОРМУЛА 5

        % дельта-бар-дельиа типа
        gains = (gains + .2) .* (sign(y_grads) ~= sign(y_incs)) ...
              + (gains * .8) .* (sign(y_grads) == sign(y_incs));
        gains(gains < min_gain) = min_gain;
```

```
        y_incs = momentum * y_incs - mu * (gains .* y_grads);%приращение У с учетом
град
        ydata = ydata + y_incs;
%          ydata = bsxfun(@minus, ydata, mean(ydata, 1));% нормализуем

          for i=1: size(ydata,1)%вычитаем их каждой строки среднее

               ydata(i,:)=ydata(i,:)-mean(ydata, 1);

          end
        % изменяем момент
        if iter == momentum_iter_change
            momentum = final_momentum;
        end
        %убираем гиперсусиление
        if iter == stop_lying_iter
            P = P ./ 4;
        end

        % чтобы не уснуть смотрим результат
        if ~rem(iter, 10)
            cost = divKL - sum(P(:) .* log(Q(:)));% функция стоймости кульбака-
Лейблера

            disp(['Итерация ' num2str(iter) ': дивергенция К-Л: ' num2str(cost)]);
        end


    end
```

Функция для вычисления гаусова ядра и перплексии

```
function [PerInc, P] = perl_comp(D, gamma)
%формула 1
    P = exp(-D * gamma);%числитель оно же ядро гауса
    sumP = sum(P);%знаменатель
    PerInc = log(sumP) + gamma * sum(D .* P) / sumP;%перплексия, если в (1) посставить
(2), энтропия или лог(перп)
    P = P / sumP;%формула 1 вычисение p(j|i)
end
```

Функция для вывода графика в виде чисел

```
function  plotin2D( mappedX,train_labels )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

for i=1:length(train_labels)
     gscatter(mappedX(i,1), mappedX(i,2), train_labels(i),'w');
    hold on

    strValues = strtrim(cellstr(num2str(train_labels(i))));
    if train_labels(i)==1
    text(mappedX(i,1), mappedX(i,2),strValues, 'HorizontalAlignment','center',
'VerticalAlignment','middle', 'fontsize', 12,'Color',[1 0 0] );
    elseif train_labels(i)==2
            text(mappedX(i,1), mappedX(i,2),strValues, 'HorizontalAlignment','center',
'VerticalAlignment','middle', 'fontsize', 12,'Color',[0.4 0.6 1]  );
     elseif train_labels(i)==3
             text(mappedX(i,1), mappedX(i,2),strValues,
'HorizontalAlignment','center', 'VerticalAlignment','middle', 'fontsize', 12,'Color' ,
[1 0.170 0]  );
         elseif train_labels(i)==4
                 text(mappedX(i,1), mappedX(i,2),strValues,
'HorizontalAlignment','center', 'VerticalAlignment','middle', 'fontsize', 12,'Color',
[0 1 0] );
             elseif train_labels(i)==5
                     text(mappedX(i,1), mappedX(i,2),strValues,
'HorizontalAlignment','center', 'VerticalAlignment','middle', 'fontsize',
12,'Color',[1 0.1 0.8]  );
```

```
                 elseif train_labels(i)==6
                         text(mappedX(i,1), mappedX(i,2),strValues,
'HorizontalAlignment','center', 'VerticalAlignment','middle', 'fontsize',
12,'Color',[0.2 0.4 0.3]  );
                     elseif train_labels(i)==7
                             text(mappedX(i,1), mappedX(i,2),strValues,
'HorizontalAlignment','center', 'VerticalAlignment','middle', 'fontsize',
12,'Color',[0.1 0.4 0.6] );
   elseif train_labels(i)==8
            text(mappedX(i,1), mappedX(i,2),strValues, 'HorizontalAlignment','center',
'VerticalAlignment','middle', 'fontsize', 12,'Color',[0 0 1]  );
      elseif train_labels(i)==9
          text(mappedX(i,1), mappedX(i,2),strValues, 'HorizontalAlignment','center',
'VerticalAlignment','middle', 'fontsize', 12,'Color',[0.67 0.223 0.79]  );
    else
        text(mappedX(i,1), mappedX(i,2),strValues, 'HorizontalAlignment','center',
'VerticalAlignment','middle', 'fontsize', 12,'Color',[0.6 1 0.1]  );
    end


end
end
```

Листинг программы в среде Python

```
# That's an impressive list of imports.
import numpy as np
from numpy import linalg
from numpy.linalg import norm
from scipy.spatial.distance import squareform, pdist

# We import sklearn.
import sklearn
from sklearn.manifold import TSNE
from sklearn.datasets import load_digits
from sklearn.preprocessing import scale

# We'll hack a bit with the t-SNE code in sklearn 0.15.2.
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.manifold.t_sne import (_joint_probabilities,
_kl_divergence)

RS = 20150101
# We import seaborn to make nice plots.
import seaborn as sns
sns.set_style('darkgrid')
sns.set_palette('muted')
sns.set_context("notebook", font_scale=1.5,
rc={"lines.linewidth": 2.5})

# We'll use matplotlib for graphics.
import matplotlib.pyplot as plt
import matplotlib.patheffects as PathEffects
import matplotlib

import cPickle, gzip

# Load the dataset
f = gzip.open('mnist.pkl.gz', 'rb')
train_set, valid_set, test_set = cPickle.load(f)
f.close()


x=train_set[0:1]
y=train_set[1:2]

y=y[0][0:4999]

x=x[0][0:4999]
```

```python
digits = load_digits()
digits.data.shape



# We first reorder the data points according to the handwritten numbers.
x = np.vstack([x[y==i]
               for i in range(10)])
y = np.hstack([y[y==i]
               for i in range(10)])

digits_proj = TSNE(random_state=RS,perplexity=50.0).fit_transform(x)

def scatter(x, colors):
    # We choose a color palette with seaborn.
    palette = np.array(sns.color_palette("hls", 10))

    # We create a scatter plot.
    f = plt.figure(figsize=(8, 8))
    ax = plt.subplot(aspect='equal')
    sc = ax.scatter(x[:,0], x[:,1], lw=0, s=40,
                    c=palette[colors.astype(np.int)])
    plt.xlim(-25, 25)
    plt.ylim(-25, 25)
    ax.axis('off')
    ax.axis('tight')

    # We add the labels for each digit.
    txts = []
    for i in range(10):
        # Position of each label.
        xtext, ytext = np.median(x[colors == i, :], axis=0)
        txt = ax.text(xtext, ytext, str(i), fontsize=24)
        txt.set_path_effects([
            PathEffects.Stroke(linewidth=5, foreground="w"),
            PathEffects.Normal()])
        txts.append(txt)

    return f, ax, sc, txts


scatter(digits_proj, y)


plt.show()
```