Math 444: Project 4

Levent Batakci

March 25, 2021

The work in this project focuses on using Self-Organizing Maps (SOM) as a means to approximate data with low-dimensional manifolds. This is the power of SOM - it can approximate data in subspaces which are not necessarily affine. All of the mentioned implementations are coded in MATLAB and can be found in my public GitHub Repository.

Self-Organizing Maps Background

While we won't delve fully into the details of Self-Organizing Maps (SOM), it is important understand this technique from a conceptual standpoint. Unlike the previous methods of analysis, SOM stands upon a heuristic foundation. That is, it is not constructed from the same level of mathematical rigor as techniques such as PCA, LDA, and k-Medoids are.

SOM seeks to approximate data by using prototypes. If k is the number of dimensions we wish to approximate the data in, then these prototypes have a topology that aligns with the topology of a k-dimensional lattice. Note that a 1-dimensional lattice is essentially just a section of a number line.

The prototypes are initialized randomly about the origin. Then, data is fed in from the source, and the prototypes are adjusted accordingly. The closest prototype - also known as the best matching unit - is adjusted most drastically, and the rest are adjusted proportionally to the topology of the lattice. Moreover, the prototypes are adjusted less and less drastically as time goes on. This helps the system to converge, and reflects the idea that learning should be most rapid when little is known.

This idea can seem pretty abstract, so it can be helpful to compare this technique to familiar concepts. For instance, if you adjust the sleeve of your shirt by pulling it forward - the spots on the shirt closest to the sleeve will move the most, while the spots on other places on the shirt will be adjust minimally. So while the fabric on your shoulder might move a bit, the fabric of your other sleeve will likely not move at all.

Another apt comparison is that between SOM and John Locke's tabula rasa - which translates to blank slate. The tabula rasa is a philosophical concept which states that the mind is initially empty and forms itself through experience and perception. Indeed, SOM works just like this! The prototypes are initally meaningless, and they adjust themselves as they become exposed to more and more data.

Handwritten Digits Analysis

The first data set we analyzed was a subset of the classic MNIST Handwritten Digits data set - which consists of 16×16 pixel images of handwritten digits. Specifically, the data $x^{(j)} \in \mathbb{R}^{256}$, where $1 \leq j \leq 515$. So the data is stored in a matrix $X \in \mathbb{R}^{256 \times 515}$.

To be able to analyze this data more effectively, we reduced it to only look at 3 digits at a time so that we could use a reasonable number of prototypes. In every case, we choose to approximate the reduced

data with a two-dimensional manifold with 100 prototypes. So the topology of the prototypes aligned with that of a 10×10 lattice.

The first three digits we considered were 1, 3, and 9. To visualize the results of the SOM algorithm, we found the closest data point (according to the 2-norm) to each prototype. We displayed all of these image in a grid corresponding to the prototypes. Naturally, the organization of the prototypes on the grid aligned with their corresponding spots on the 2D lattice. This results are shown in **Figure 1**.

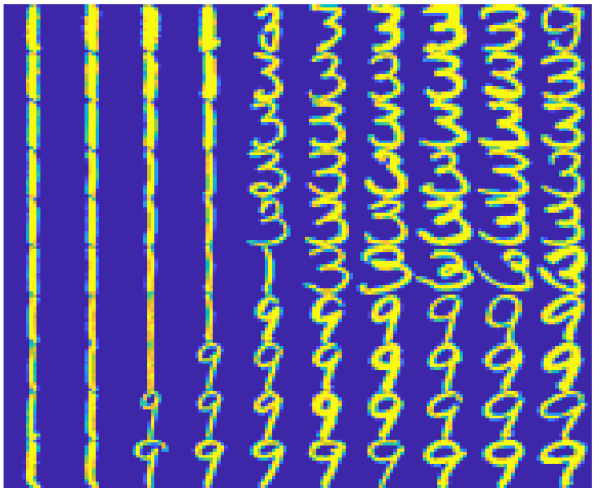

Figure 1: SOM Prototype Visualization on Digts 1, 3, and 9

Immediately, it's apparent that there is a meaningful organization to the prototypes. Indeed, the digits seem to be pretty clearly grouped correctly according to the lattice. The one outlier which can be spotted is the 9 in the top-right corner. That being said, this digit is somewhat fundamentally different from the other nines. Noting this, it is reasonable for it to be grouped with the threes.

We also performed an analysis on the digits 0, 6, and 9. The results can be seen below in **Figure 2**. It's apparent that again, the SOM is able to find prototypes which separate this data quite well on a 2-dimensional manifold.

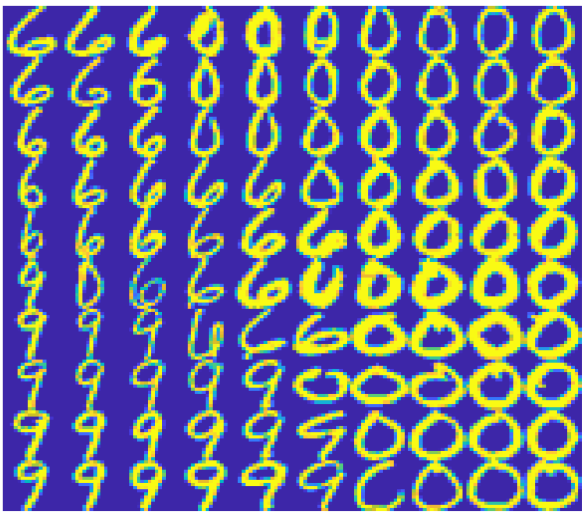

Figure 2: SOM Prototype Visualization on Digts 1, 3, and 9

Overall, we conclude that the data does separate well on a two-dimensional manifold despite living in 256-dimensional space! Intuitively, this makes sense since we view images as 2D objects.

Breast Cancer Analysis

The next data we analyzed was the Breast Cancer Wisconsin data set. The data is of the form $x^{(j)} \in \mathbb{R}^{30}$where $1 \leq j \leq 569$, and each of the components of data components is a integer between 1 and 10. So the data is stored in a matrix $X \in \mathbb{R}^{30\times 569}$. The data is also accompanied by an annotation matrix I which places each data point into one of two clusters - benign and malignant.

Particularly, we are interested in whether the data holds insight to a natural separation of these two diagnoses. Again, we chose to approximate the data with a two-dimensional manifold. Furthermore, we again used 100 prototypes - corresponding to a 10×10 lattice.

To visualize the results of the SOM algorithm, we took a slightly different approach than with the digits. We constructed a grid of of circles representing the prototypes. Then, for each data point, we determined the closest prototype, and placed a dot (randomly) in the corresponding circle on the grid. Specifically, we colored the benign dots green and the malignant dots red. The results are shown in **Figure 3**.

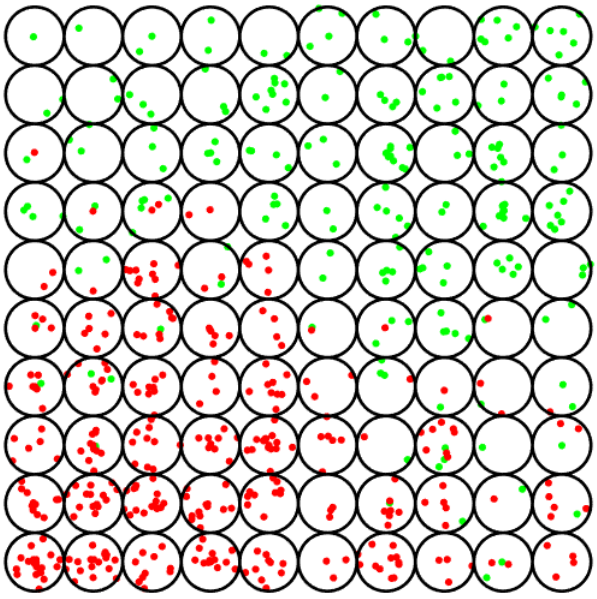

Figure 3: SOM Results on Breast Cancer Data

Immediately, it's apparent that the two diagnostic groups separate very well. There are several prototypes with a mix of benign (green) and malignant (red) data points, but most prototypes have an overwhelming majority of one of the other. Furthermore, benign-majority prototypes and malignant-majority prototypes are grouped very clearly on the lattice.

We are lead to conclude that the breast cancer data does in fact hold insight which differentiates the two diagnoses. With something so serious, however, it would be imperative to exercise caution when making a conclusion about a data point near the boundary of the prototype groups. Overall, SOM did a far better job than any of the other techniques thus far at separating the malignant tumors from the benign!

Appendix

As stated before, please feel free to visit the public GitHub Repository.

1 SOM - Parameters

In the SOM background section, we did not talk at all about parameters. Indeed, there are many parameters to consider when running SOM. For instance, how fast should the model learn at the start? How fast should it learn after the learning period? How long should the learning period be? These are just a few questions which give rise to necessary parameters for the algorithm. The code includes a defaultParams functions to compute good rule-of-thumb parameters to use.

2 Code - SOM

```
function M = SOM(X, k, parameters)

%Get data dimensions
[n, p] = size(X);

%Solve for lattice width
N = k^(1/2);
if(mod(N,1) >= 1e-14)
    disp("Lattice cardinality does not fit the number of dimensions!");
end
N = round(N);

%Initialize the parameters
Tmax = parameters(1);
a0 = parameters(2);
a1 = parameters(3);
gamma0 = parameters(4);
gamma1 = parameters(5);
T0 = parameters(6);

%Initialize the prototypes
M = zeros(n,k);
for i = 1:k
    m = RandRange(0.001,0, n);
    M(:,i ) = m;
end

C = zeros(2,k);
for i = 1:k
    C(:, i) = getCoord(i,N);
end

%LEARN!
t=0;
while(t<= Tmax)
    %Update parameters
    a = Decayed(t, a0, a1, T0);
    gamma = Decayed(t, gamma0, gamma1, T0);

    %Choose a random data point to introduce
    x = datasample(X,1,2);
```

```
    [~, bmu_I] = min(vecnorm(M - x));

    %Update prototypes!
    c = getCoord(bmu_I,N);
    diff = (x-M);

    H = exp(-1/(2*gamma^2) * (vecnorm(c-C).^2));
    M = M + a * H .* diff;

    t=t+1;
end
end
```

3 Code - Digits Analysis

```
 %Levent Batakci
%3/21/2021

%Load the data
load HandwrittenDigits.mat

%Trim the data
c1 = 1;
c2 = 3;
c3 = 9;
X(:, I~=c1 & I~=c2 & I~=c3) = [];
[n,p] = size(X);

k=100;
N = k^(1/2);
params = defaultParams(k, 2);
M = SOM(X, k, params);


%Visualize protoypes by finding closest matches
V = zeros(16*N,16*N);
for i = 1:N
    for j = 1:N
        key = getKey(i,j,N);
        m = M(:, key);

        %Find the best matching unit
        [~, bmu_I] = min(vecnorm(X - m));
        bmu = X(:, bmu_I);

        V((i-1)*16+1:i*16, (j-1)*16+1:j*16) = reshape(bmu,16,16)';
    end
end

imagesc(V)
set(gca,'visible','off');
```

4 Code - Breast Cancer Analysis

```
%Levent Batakci
%3/21/2021

%Load data
load WisconsinBreastCancerData_Unpacked.mat

%Sort data
X = Data_WCD_Matrix;
I = I_Label';
benign = X(:,I==1);
malignant = X(:,I==2);

%SOM!
k = 100;
N = k^(1/2);
params = defaultParams(k,2);
M = SOM(X, k, params);

hold on
%Place the benign data!
for x = benign
    [~, bmu_I] = min(vecnorm(M - x));
    c = getCoord(bmu_I, N);

    %Place a dot
    theta = RandRange(2*pi,0,1);
    r = RandRange(0.5,0,1);
    x_ = r*cos(theta) + c(1);
    y_ = r*sin(theta) + c(2);
    scatter(x_, y_, 250, 'g.');
end

%Place the malignant data!
for x = malignant
    [~, bmu_I] = min(vecnorm(M - x));
    c = getCoord(bmu_I, N);

    %Place a dot
    theta = RandRange(2*pi,0,1);
    r = RandRange(0.5,0,1);
    x_ = r*cos(theta) + c(1);
    y_ = r*sin(theta) + c(2);
    scatter(x_, y_, 250, 'r.');
end

%%SET UP CIRCLES!%%
cirlces = zeros(N,N);
q1 = [1:N]'*ones(1,N);
q2 = ones(N,1)*[1:N];
Q = [q1(:) q2(:)];
% Define the distance squared matrix
```

```
D2 = zeros(k,k);
for i = 1:k
    for j = 1:i
        D2(i,j) = norm(Q(i,:) - Q(j,:))^2;
        D2(j,i) = D2(i,j);
    end
end

% Plotting the buttons on the map
thplot = linspace(0,2*pi,k);
cc = cos(thplot);
ss = sin(thplot);
for j = 1:k
    plot(Q(j,1)*ones(1,k) + 0.5*cc,Q(j,2)*ones(1,k)+0.5*ss,'k-','LineWidth',2)
    hold on
end
axis('square')
axis([0,N+1,0,N+1])
set(gca,'FontSize',20)
set(gca, 'visible', 'off')
%%%%%
```