Artificial Neural Networks
Optimizing CSP with Hopfield Nets

Mayukh Sattiraju
msattir@clemson.edu

ECE 872
Takehome2 Report

April 10, 2018

Metric Learning and Optimizing Loss functions with Hopefield Nets
ECE 872 Takehome 2 Proposal

Mayukh Sattiraju

March 16, 2018

1 Proposal

For this assignment, I intend to implement a classifier that uses Metric Learning as the error metric. This error would be optimized using an RNN/Hopfield Network. The RNN/Hopfield network would be used to find optimal weights in the weight space. This would be compared with a conventional classifier.

2 Metric Learning

Conventional error functions only minimize the distance of a sample to other samples from the same class. Triple Learning (a form of Metric Learning) tries to optimize three losses - minimize the intra-class error, maximize the inter-class error and minimize the distance to an anchor m. The purpose of the anchor is to effectively cluster data points from a single class.

3 Assignment Deliverables

- As the main goal here is to optimize a constrained problem, I would setup an Hopfield net (if possible an RNN) to optimize a set of constraints. So, I would first solve an N-Queens optimization problem and build on that.
- I would compare clustering methods of regular loss functions to triple learning (metric learning) methods. (Introduction of the problem).
- I would then derive constraints and model an Hopfield net (or an RNN) to be used as the optimizer. (Methods)
- The desired result to be achieved is this new Hopefield Net and Triple Loss optimizer that yields to be a better classifier. A comparison of the methods with various configurations of optimizers (Hopefield Net vs Gradient methods) and loss functions (Metric loss vs Euclidean Loss) would be presented. (Results)

4 References

Hermans, A., Beyer, L., & Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. Retrieved from http://arxiv.org/abs/1703.07737

Rippel, O., Paluri, M.,2015. Metric Learning with Adaptive Density Discrimination. https://research.fb.com/wp-content/uploads/2016/05/metric-learning-with-adaptive-density-discrimination.pdf?

1 Introduction

This assignment deals with implementing Hopfield Nets to optimize a Constraint Satisfaction Problem (CSP).

The assignment first begins with implementing a binary Hopfield Net to optimize a discrete CSP problem. Then this is extended to solve 'triplet loss' [1] a constrained optimization problem.

The discrete CSP problem chosen here is to solve a 4-Queens problem. The problem deals with placing 4 Queens on a 4x4 chess board such that they don't 'attack' each other. Constraints for this problem were developed, and from these constraints a weight matrix (or a interconnection matrix) for the Hopfield net was derived. It is designed such that solutions that satisfy more constraints would have lesser Energy than solutions that satisfy fewer constraints. Using this interconnection matrix and a arbitrary starting state, the algorithm was run. The hopfield net settles to states that have low energy, which, by design, are states that satisfy most constraints.

Triplet Learning is a form of Metric Learning that optimizes a triplet Loss function [2]. The method attempts to cluster data points that belong to the same class, and force points away that belong to different classes. This involves minimizing the distance with intra-class samples and maximizing the distance for inter-class samples.

2 Methods

This section describes the various Network architectures employed to create the mapping with the least error/deviation from the expected output.

2.1 Discrete Hopfield Nets

Hopfield Nets as described in [3] are densely connected Networks. With binary neurons and weights from every other node to any node the neurons of a Hopfield Net toggle their activation based on a Hopfield prescription rule. The interesting fact is that, the toggling of the neurons ensures that the resulting state of the network has a lesser ' GLobal Energy' than the previous state. Taking advantage of this inherit minimization, a CSP can be modelled to fit this energy function, and the hopfield net would settle on a set of activations that has a lesser global energy.

The toggling of a state of Hopfield Net neuron is given by this rule:

$$v_i = \left\{ \begin{array}{ll} 1 & if \quad u_i > 0 \\ 0 & if \quad u_i < 0 \\ previous\ value & otherwise \end{array} \right\} \tag{1}$$

Where $U_{i,j}$ is the net activation to $neuron_{i,j}$.

Formally $U_{i,j}$ can be computed as:

$$u_i = \sum_j w_{i,j} v_j + b_i \tag{2}$$

Where $w_{i,j}$ are the weights from each other neuron, and b_i is the bias to that neuron.

Given weights and activations at any instant, the Energy associated with that configuration of the network is given by

$$E = -\frac{1}{2} v^T W v - b^T v \tag{3}$$

2.2 4-Queens CSP

The 4-Queens problem attempts to place 4-Queens on a 4x4 chessboard such that no queen attacks each other. The problem tries to maximize the number of queens placed while maintaining certain constraints [4].

2.2.1 Constraints

These are the constraints that the problem tries to maintain.

Row Constraint: This constraint ensures there is only one Queen in any row.

$$E_1 = \frac{A}{2} \times \sum_{i=1}^{4} \sum_{j=1}^{4} \sum_{\substack{k=1 \\ k \neq j}}^{4} v_{i,j} v_{i,k} \tag{4}$$

This multiplies pairs of cells to ensure that there is a maximum of 1 queen in any row, and if so the sum is zero. This is violated if there is more than 1 Queen in any Row, and this would produce a non-zero sum.

Column Constraint: This constraint ensures there is only one Queen in any column.

$$E_2 = \frac{B}{2} \times \sum_{j=1}^{4} \sum_{i=1}^{4} \sum_{\substack{k=1 \\ k \neq i}}^{4} v_{i,j} v_{k,j} \tag{5}$$

This multiplies pairs of cells to ensure that there is a maximum of 1 queen in any column, and if so the sum is zero. This is violated if there is more than 1 Queen in any column, and this would produce a non-zero sum.

Diagonal Constraint: This constraint ensures there is only one Queen in any diagonal.

This can be achieved by ensuring that for queen already at (i, j) on the board there can not be another queen at a position (k, l) such that

$$(i - j \neq k - l) \quad and \quad (i + j \neq k + l)$$

Number of Queens: Given only the above the constraints, the network might converge to a trivial solution where no queens are placed on the board. To force the network to place N queens on a nxn board we have a forth constraint. This penalizes any solution that does not have exactly 4 Queens.

$$E_3 = \frac{C}{2} \times \left(\left(\sum_{i=1}^{4} \sum_{j=1}^{4} v_{i,j} - 4 \right)^2 \right) \tag{6}$$

2.3 Weights

Equation [3] gives us a relation between Energy and Weights of a Hopfield Net. This equation can be differentiated twice to derive weights.

$$W_{i,j} = -\frac{\partial^2 E}{\partial v_i \partial v_j} \tag{7}$$

The weight matrix for this problem is found to be:

```
W_sym =

[       D, A/2 + D, A/2 + D, A/2 + D, B/2 + D, C/2 + D,       D,       D, B/2 + D,       D, C/2 + D,       D, B/2 + D,       D,       D, C/2 + D]
[ A/2 + D,       D, A/2 + D, A/2 + D, C/2 + D, B/2 + D, C/2 + D,       D,       D, B/2 + D,       D, C/2 + D,       D, B/2 + D,       D,       D]
[ A/2 + D, A/2 + D,       D, A/2 + D,       D,       D, B/2 + D, C/2 + D, C/2 + D,       D, B/2 + D,       D,       D,       D, B/2 + D,       D]
[ A/2 + D, A/2 + D, A/2 + D,       D,       D,       D, C/2 + D, B/2 + D,       D, C/2 + D,       D, B/2 + D, C/2 + D,       D,       D, B/2 + D]
[ B/2 + D, C/2 + D,       D,       D,       D, A/2 + D, A/2 + D, A/2 + D, B/2 + D, C/2 + D,       D,       D, B/2 + D,       D, C/2 + D,       D]
[ C/2 + D, B/2 + D,       D,       D, A/2 + D,       D, A/2 + D, A/2 + D,       D, B/2 + D, C/2 + D,       D,       D, B/2 + D,       D, C/2 + D]
[       D, C/2 + D, B/2 + D, C/2 + D, A/2 + D, A/2 + D,       D, A/2 + D, C/2 + D, C/2 + D, B/2 + D, C/2 + D, C/2 + D,       D, B/2 + D,       D]
[       D,       D, C/2 + D, B/2 + D, A/2 + D, A/2 + D, A/2 + D,       D,       D,       D, C/2 + D, B/2 + D,       D, C/2 + D,       D, B/2 + D]
[ B/2 + D,       D, C/2 + D,       D, B/2 + D,       D, C/2 + D,       D,       D, A/2 + D, A/2 + D, A/2 + D, B/2 + D, C/2 + D,       D,       D]
[       D, B/2 + D,       D, C/2 + D, C/2 + D, B/2 + D, C/2 + D,       D, A/2 + D,       D, A/2 + D, A/2 + D, C/2 + D, B/2 + D, C/2 + D,       D]
[ C/2 + D,       D, B/2 + D,       D,       D, C/2 + D, B/2 + D, C/2 + D, A/2 + D, A/2 + D,       D, A/2 + D,       D, C/2 + D, B/2 + D, C/2 + D]
[       D, C/2 + D,       D, B/2 + D,       D,       D, C/2 + D, B/2 + D, A/2 + D, A/2 + D, A/2 + D,       D,       D,       D, C/2 + D, B/2 + D]
[ B/2 + D,       D,       D, C/2 + D, B/2 + D,       D, C/2 + D,       D, B/2 + D, C/2 + D,       D,       D, A/2 + D, A/2 + D, A/2 + D, A/2 + D]
[       D, B/2 + D,       D,       D,       D, B/2 + D,       D, C/2 + D, C/2 + D, B/2 + D, C/2 + D,       D, A/2 + D,       D, A/2 + D, A/2 + D]
[       D,       D, B/2 + D,       D, C/2 + D,       D, B/2 + D,       D,       D, C/2 + D, B/2 + D, C/2 + D, A/2 + D, A/2 + D,       D, A/2 + D]
[ C/2 + D,       D,       D, B/2 + D,       D, C/2 + D,       D, B/2 + D,       D,       D, C/2 + D, B/2 + D, A/2 + D, A/2 + D, A/2 + D,       D]
```

Figure 1: Weight Matrix

2.3.1 Biases

Biases in this application are important and can be derived by expanding Equation [6]. So, equation [6] is expanded and differentiated once, and the constant term after this differentiation is used as the bias.

$$-\frac{\partial E_2}{\partial v_i} = -\left(-\frac{B}{2}\right)(-8) = 4B \tag{8}$$

These sets of weights and biases and an arbitrary input are used to compute u_i. Then these are mapped to v_i by the rule in Equation [1]. Ones in v_i denote positions where queens are placed.

2.4 Metric Learning

Metric Learning aims to find a representation of an input image such that it would reduce an error metric. Here full-scale images are first converted to a low dimension feature sub-space (called feature embedding) then a transformation is performed such that images from the same class are moved closer to each other and images from different classes are moved away.

In contrast to a regular Eucledian Loss, the objective there is to move similar classes closer to each other. Triplet Learning attempts to push disparate classes away also.

Figure [2] from [2] demonstrates what Triplet Loss aims to achieve.

The Loss function for Triplet Learning is defined below. It computes a loss over a triplet of 3 images. An anchor image and a positive image (both from a single class) and a negative

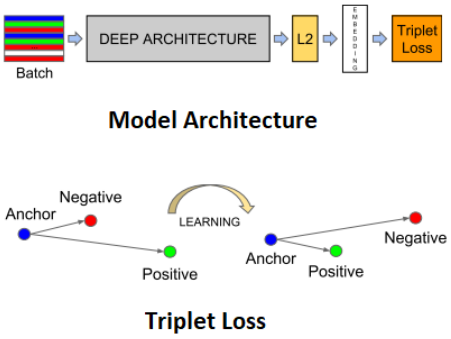

Figure 2: Triplet Loss

image (from a different class). Here the loss attempts to push the positive sample closer to the anchor image, and push the negative sample away from the anchor image.

$$\begin{array}{ll} \min\left(\|f\left(x_i^a\right) - f\left(x_i^p\right)\|_2^2\right) & \textit{where a is the anchor p is a positive example} \\ \max\left(\|f\left(x_i^a\right) - f\left(x_i^n\right)\|_2^2\right) & \textit{where a is the anchor n is a negative example} \end{array} \tag{9}$$

2.5 Triplet Learning CSP

Converting this problem to a constraint satisfaction problem is proposed below.

The problem of minimization and maximization can be combined to an overall maximization or minimization problem. Here it is converted to a overall minimization problem

$$Triplet\ Loss = \sum_{n\ samples} \left[\|f\left(x_i^a\right) - f\left(x_i^p\right)\|_2^2 - \|f\left(x_i^a\right) - f\left(x_i^n\right)\|_2^2 + \alpha\right]_+ \tag{10}$$

This is used instead of the conventional eucledian loss. So during the testing phase, inputs from the same class are mapped close to each other and inputs from different classes are mapped far from each other.

2.5.1 Gradient Methods

This is the above triplet loss function implemented using standard Gradient Decent methods on a 2-Layered Fully connected Network.

Listing 1: Metric Learning Gradient Decent output

```
l:1 en:0.077086 ep:0.042181 Er:0.16146
l:2 en:0.17971 ep:0.023405 Er:0.11014

l:4116 en:0.42515 ep:0.028149 Er:-0.012574
l:4117 en:0.42515 ep:0.028149 Er:-0.012574
l:4118 en:0.42515 ep:0.028149 Er:-0.012574
l:4119 en:0.42515 ep:0.028149 Er:-0.012574
```

```
l:4120 en:0.42515 ep:0.028149 Er:-0.012574
```

Here we see that the distance between the anchor and the negative sample is higher than the distance between the positive sample and the anchor. This is the mapping that we expect a network to learn. In the conventional method, using the eucledian distance, the distance between the positive and the anchor would reduce, but it doesn't guarentee that the distance between the anchor and negative would increase. This clustering makes it easy to later classify the dataset.

2.5.2 Hopfield Net/Lyapanov Function

To solve this using a Binary Hopfield Net, it was difficult to write this loss function as a Quadratic function. The mapping of continuous constraints to a continuous problem was the biggest hurdle.

Also as the permutations of weights that can be used to compute this loss was a very high dimension space, hence applying hopfield nets in the native form was unfruitful.

I attempted to use the error function $-\frac{1}{2}o'Wo$ with a highly reduced weight space, to help cluster the input data. I.e., W would be a matrix of highly reduced feature representation of the input images, and O would determine which pairs of input images are considered in that particular iteration. But this couldn't compute similarity between feature vectors. Thus, even this approach was unsuccessful.

3 Results

Only results pertaining to using Hopfield Nets to solve the N-Queens problem is discussed here. (As I was not able to implement triplet learning with a Hopfield Net).

Here is a sample output. The first array is a random initial state. The final output is the matrix below.

```
iter:998 Energy:-900
iter:999 Energy:-900
    1    0    0    1
    1    0    0    0
    1    1    1    1
    0    1    0    1

    0    0    1    0
    1    0    0    0
    0    0    0    0
    0    0    0    1

Hyper parameters: A:1000 B:1000 C:1000 D:100
```

Figure [3] shows how the energy of the network developed. In about 2-3 iterations it converged to a solution that had a minimum energy.

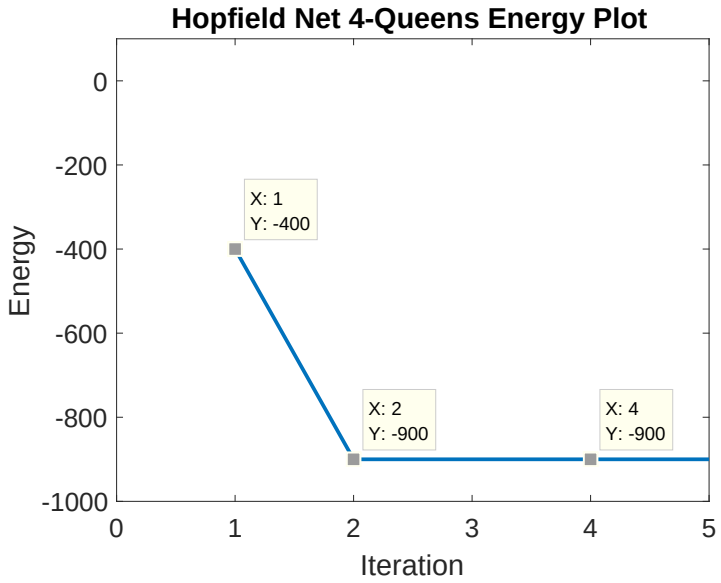

Figure 3: 4-Queens Hopfield Net

Though this solution does not violate any of the first 3 constraints, it violates the number of queens constraint. There exists a solution with 4 queens but this network is unable to find that solution as that would need re-arranging many queens. And this is not attempted, as to go to a better solution (Global Minima) the network would have to go to states that have a greater energy than this local minima. Which, by design, the network wouldn't choose. Hence, the choice of starting point matters.

Changing the hyper parameters changes the solution. Here is a version that forces more queens. Here 4-queens are placed but one row-constraint is not met.

```
iter:998 Energy:-4500
iter:999 Energy:-4500
     1     0     0     1
     1     0     0     0
     1     1     1     1
     0     1     0     1

     0     0     1     0
     1     0     0     0
     0     0     0     1
     0     0     0     1

Hyper parameters: A:1000 B:1000 C:500 D:1000
```

4 Conclusions

Hopfield Nets and similar hard-coded networks perform a particular optimization task very well, even an NP-Hard problem such as N-Queens. But the applicability of these Networks lies in the success of mapping constraints and developing a suitable quadratic loss function. For the 4-Queens problem, a hopfield implementation was very straightforward. The discrete nature of the problem also made it easy to implement. Applicability of Hopfield Nets in the continuous domain would have been a direction that might have helped with the triplet learning task. The hopfield nets used here search only a 2-Dimensional weight space and these were unable to search a N-Dimensional (image) feature space. Thus, to minimize errors in higher dimensions a N-Dimensional Hopefield might have been necessary.

5 Future Work

As seen in the previous result, hopfield nets suffer from being unable to come out of local minima. Thus implementing Boltzman or stochastic networks would have been the suitable direction forward.

I now feel that searching a d-dimensional weight space to find a suitable combination of weights would require a N-Dimensional Hopfield net. Conceptualizing this N-Dimensional Hopfield Net and coming up with a similar Lypunov energy function would be a theoretical future direction.

I also came across a paper that used an idea of persistent memory and learned the weights of a Hopfield Network. Thus not having to hardcode them. This also would have been a interesting direction to solve a kind of optimization problems.

References

[1] Hermans, A., Beyer, L., & Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. Retrieved from http://arxiv.org/abs/1703.07737

[2] Schroff, F., Kalenichenko, D., & Philbin, J. (n.d.). FaceNet: A Unified Embedding for Face Recognition and Clustering. Retrieved from https://arxiv.org/pdf/1503.03832.pdf

[3] Hopfield J. J., Tank D. W. Neural Computations of Decisions in Optimization Problems

[4] Jacek Mandziuk. (n.d.). Neural Networks for the N-Queens Problem. Retrieved from http://www.mini.pw.edu.pl/ mandziuk/PRACE/nqp-rev.pdf