Name: Apoorv Pathak **Roll Number: 202051033**

Week 8 :

Lab Assignment 7

Learning Objective:

To model the low level image processing tasks in the framework of Markov Random Field and Conditional Random Field. To understand the working of Hopfield network and use it for solving some interesting combinatorial problems

Reference:

1. http://www.cs.utoronto.ca/~strider/Denoise/Benchmark/
2. Interesting way to denoise an image using random walk: http://www.cs.toronto.edu/~fleet/research/Papers/BMVC_denoise.pdf
3. MRF Image Denoising: https://web.cs.hacettepe.edu.tr/~erkut/bil717.s12/w11a-mrf.pdf
4. Single Neuron and Hopfield Network: Chapter 40, 41, 42
   Information Theory, Inference and Learning Algorithms, David MacKay
   http://www.inference.phy.cam.ac.uk/mackay/itila/

Problem Statement:

A. Many low level vision and image processing problems are posed as minimization of energy function defined over a rectangular grid of pixels. We have seen one such problem, image segmentation, in class. The objective of image denoising is to recover an original image from a given noisy image, sometimes with missing pixels also. MRF models denoising as a probabilistic inference task. Since we are conditioning the original pixel intensities with respect to the observed noisy pixel intensities, it usually is referred to as a conditional Markov random field. Refer to (3) above. It describes the energy function based on data and prior (smoothness). Use quadratic potentials for both singleton and pairwise potentials. Assume that there are no missing pixels. Cameraman is a standard test image for benchmarking denoising algorithms. Add varying amounts of Gaussian noise to the image for testing the MRF based denoising approach. Since the energy function is quadratic, it is possible to find the minima by simple gradient descent. If the image size is small (100x100) you may use any iterative method for solving the system of linear equations that you arrive at by equating the gradient to zero. Extra Credit Challenge: Implement and compare MRF denoising with Stochastic denoising (reference 2).
B. For the sample code hopfield.m supplied in the lab-work folder, find out the amount of error (in bits) tolerable for each of the stored patterns.
C. Solve a TSP (traveling salesman problem) of 10 cities with a Hopfield network. How many weights do you need for the network?

A Markov Random Field is a graph whose nodes model random variables, and whose edges model desired local influences among pairs of them. Local influences propagate globally, leveraging the connectivity of the graph. A natural MRF that models this has a node for each of the pixels. An edge connects two nodes that are adjacent (on the grid).

Real-world images tend to be smooth. That is, adjacent pixels tend to have similar colors. Sure, they might also be abrupt transitions such as edges or object boundaries. However, there is generally a lot of smoothness between these transitions. It is the MRF's edges that somehow favor that adjacent pixels have similar values.

Cliques And Energy Functions

A clique in a graph is a set of nodes in which every pair is connected by an edge. A clique is maximal if it is not part of a larger one. Maximal cliques play a crucial role in how an MRF operates. Intuitively each such clique specifies a set of nodes, all of whose values directly influence each other.

We control the specifics of the interaction among the nodes' values in a clique by a so-called energy function of our choice. This function (one per maximal clique) inputs values for all the clique nodes and outputs its energy, a scalar. The energy function we choose should assign low energy to a good combination of values of these nodes. That is, low energy is good.

The global energy for a particular assignment of values to its nodes is the sum of all the maximal cliques' energies in the graph.

Smoothness-favoring Energy Function

In our image MRF, what energy function would deliver smoothness? Consider $|xi-xj|$. Here *i* and *j* denote adjacent pixels, with *xi* and *xj* denoting their values. This energy function prefers that *xi* and *xj* have similar values. As a result, global energy will be minimized by an image that is maximally smooth.

Note that, the under-the-hood, *xi*, and *xj*, might have multiple dimensions, e.g., RGB for color. We don't care so long as we can meaningfully calculate the distance between the two colors, which is what $|xi-xj|$ is. Well, that's nice, but what good is a maximally smooth image? All pixels would have the same color. Below we see some specific useful image processing problems in which smoothness, hence this MRF, plays an important role.

Image Denoising

Consider this example.

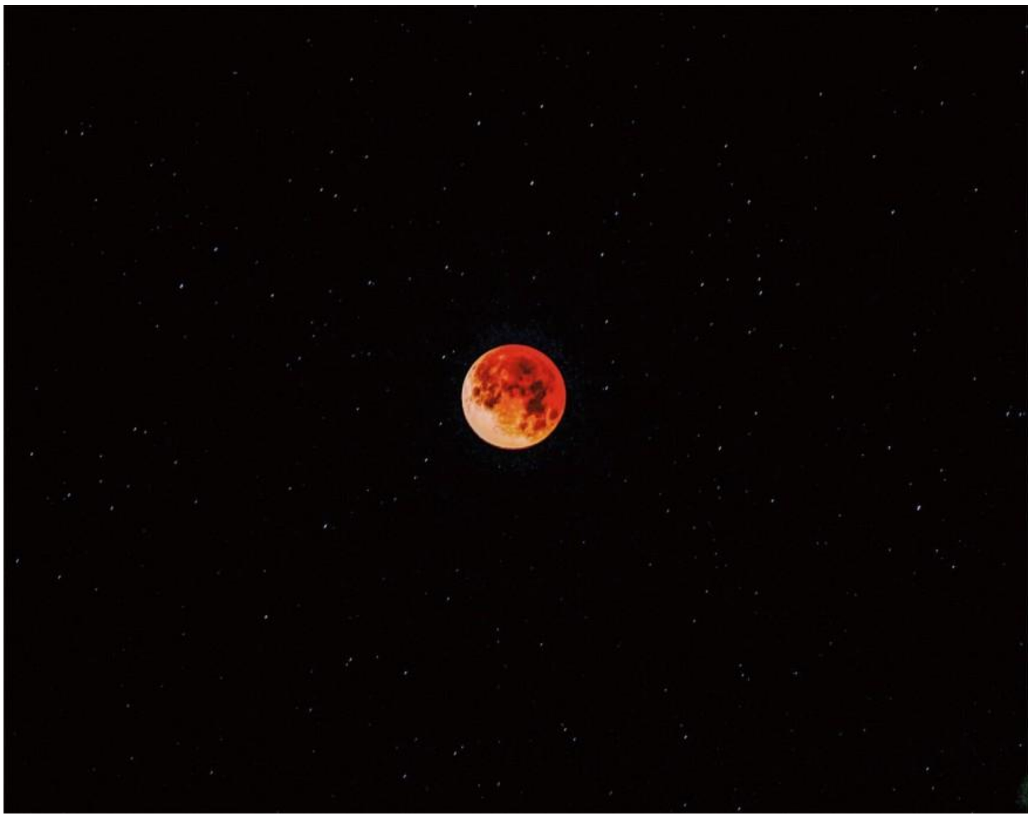
Photo on Unsplash

Look at the white dots (stars). Let’s say someone would like to think of them as noise (even though they are not). With a click of a button, the person can ask this image to be denoised. *The denoiser’s job is to identify these white dots as the most likely culprits and blacken them.*

Why would we expect the MRF to be able to detect these white dots as noise? For the sake of discussion, assume that each white dot is a single pixel. A dot’s pixel is white, and all its neighbors are black. Moreover, black and white are very different-looking colors. The MRF doesn’t like this disharmony. That is, global energy would decrease significantly if we made the spot’s color black. This reasoning applies to all the white dots.

As to any pixel in the bright red sphere, it has at least some neighbors with a similar color. So drastically changing the color of such a pixel will not reduce the energy.

Image Restoration

Here we are given an image that is both *noisy* (i.e. some of its pixels have incorrect values) and *incomplete* (i.e., some of its pixels don't have a value). The two-sheet MRF is a good fit for solving this problem as well. As before, we'll set X's values from the input image. Except for the pixels in the input image that are missing values are left free.

Image Segmentation

Here we'd like to segment the image into maximally homogenous regions. Homogeneity means that some property remains similar in the entire area. The property might be color. Or texture. Or a combination. Maximally means that different regions, especially adjacent ones, are dissimilar. Sound like clustering? Yes, in a 2D setting on image properties: color, etc.

A particular critical case of segmentation is to split the image into foreground and background. In our sphere example, the sphere would be the foreground, and the black sky (including the stars) the background.

In general, even such binary segmentation is not easy to do. Unlike the simple background in our example (black sky), others may have varying textures or colors. Plus, the image may contain multiple objects. In fact, when this is the case, how is the foreground even defined?

An effective way to both simplify and disambiguate the problem is called *interactive segmentation*. The user draws a bounding box around the desired object to be segmented. The algorithm takes this box as a rough initial segmentation and tries to improve upon it.

Take the sky example modified so that there are two distinct spheres in it. Think of them like the sun and the moon. The user might put the bounding box around the moon. So both the sun and the sky would be in the background.

Mathematical overview of Conditional Random Fields

Having discussed the above definitions, we will now go over Conditional Random Fields, and how they can be used to learn sequential data.

As we showed in the previous section, we model the Conditional Distribution as follows:

$$\hat{y} = argmax_y P(y|x)$$

Conditional Distribution

In CRFs, our input data is sequential, and we have to take the previous context into account when making predictions on a data point. To model this behavior, we will use Feature Functions, that will have multiple input values, which are going to be:

1. The set of input vectors, X
2. The position i of the data point we are predicting
3. The label of data point i-1 in X
4. The label of data point i in X

We define the feature function as:

$$f(X, i, l_{i-1}, l_i)$$

Feature Function

The purpose of the feature function is to express some kind of characteristic of the sequence that the data point represents. For instance, if we are using CRFs for Parts-of-Speach tagging, then

f (X, i, L{i - 1}, L{i}) = **1** if L{i - 1} is a Noun, and L{i} is a Verb. **0** otherwise.

Similarly, f (X, i, L{i - 1}, L{i}) = **1** if L{i - 1} is a Verb and L{i} is an Adverb. **0** otherwise.

Each feature function is based on the label of the previous word and the current word and is either a 0 or a 1. To build the conditional field, we next assign each feature function a set of weights (lambda values), which the algorithm is going to learn:

$$P(y, X, \lambda) = \frac{1}{Z(X)} exp\{\sum_{i=1}^{n} \sum_{j} \lambda_j f_i(X, i, y_{i-1}, y_i)\}$$

$$\text{Where: } Z(x) = \sum_{y' \in y} \sum_{i=1}^{n} \sum_{j} \lambda_j f_i(X, i, y'_{i-1}, y'_i)$$

Probability Distribution for Conditional Random Fields

To estimate the parameters (lambda), we will use Maximum Likelihood Estimation. To apply the technique, we will first take the Negative Log of the distribution, to make the partial derivative easier to calculate:

$$L(y, X, \lambda) = -log\{\prod_{k=1}^{m} P(y^k | x^k, \lambda)\}$$

$$= -\sum_{k=1}^{m} log[\frac{1}{Z(x_m)} exp\{\sum_{i=1}^{n} \sum_{j} \lambda_j f_j(X^m, i, y_{i-1}^k, y_i^k)]$$

Negative Log Likelihood of the CRF Probability Distribution

To apply Maximum likelihood on the Negative Log function, we will take the *argmin* (because minimizing the negative will yield the maximum). To find the minimum, we can take the partial derivative with respect to lambda, and get:

$$\frac{\partial L(X, y, \lambda)}{\partial \lambda} = \frac{-1}{m} \sum_{k=1}^{m} F_j(y^k, x^k) + \sum_{k=1}^{m} p(y|x^k, \lambda) F_j(y, x^k)$$

$$\text{Where: } F_j(y, x) = \sum_{i=1}^{n} f_i(X, i, y_{i-1}, y_i)$$

Partial Derivative w.r.t. lambda

We use the Partial Derivative as a step in Gradient Descent. Gradient Descent updates parameter values iteratively, with a small step, until the values converge. Our final Gradient Descent update equation for CRF is:

$$\lambda = \lambda + \alpha[\sum_{k=1}^{m} F_j(y^k, x^k) + \sum_{k=1}^{m} p(y|x^k, \lambda) F_j(y, x^k)]$$

Gradient Descent Update Equation for CRF

As a summary, we use Conditional Random Fields by first defining the feature functions needed, initializing the weights to random values, and then applying Gradient Descent iteratively until the parameter values (in this case, lambda) converge. We can see that CRFs are similar to Logistic Regression since they use the Conditional Probability distribution, but we extend the algorithm by applying Feature functions as our sequential inputs.