Chapter 1

Ising Model and Boltzmann Machines

The Ising model, named after the physicist Ernest Ising, is one of the earliest types of Markov Network models, which first arose in statistical physics as a model for the energy of a physical system involving a system of interacting atoms. Each atom X_i can take one of two values, $+1$ or -1, representing the direction of the atom's spin.

Consider the atoms are arranged in a lattice as seen in figure **??**. The energy of this system is given by

$$E(X) = -J \sum_{i,j \in Nb(i)} x_i x_j - B \sum_i x_i \tag{1.1}$$

where $Nb(i)$ is the set of neighbors of the i^{th} atom, J represents the interaction strength between a pair of neighbor atoms, and B represents the strength of an external magnetic field. If the interaction strength between variables and the influence of the external field is not the same for all the variables in the system, we have that:

$$E(X) = -\sum_{i,j \in Nb(i)} w_{ij} x_i x_j - \sum_i u_i x_i$$

The configuration probability is given by the Boltzmann distribution with inverse temperature $\beta \geq 0$.

$$P(X) = \frac{1}{Z} \exp\left[-\beta E(X)\right]$$

where Z is the normalizing constant and is obtained with the following expression:

$$Z = \sum_X \exp\left[-\beta E(X)\right]$$

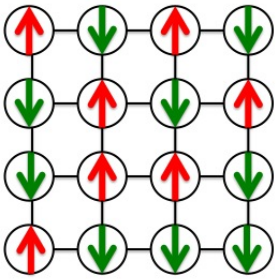


Figure 1.1: 4×4 lattice configuration where arrows represent the spin of each atom.

The energy of a single atom is given by

$$e(X_i) = -\frac{J}{2} \sum_{j \in Nb(i)} x_i x_j - B x_i$$

The factor $1/2$ is needed in order to avoid double counting the interaction between neighbor atoms, and in this way, obtain the configuration energy as the sum of the energy of each one of the atoms, in such a way that

$$E(X) = \sum_i e(X_i)$$

When two neighbor atoms have the same spin, the energy decreases, on the other hand, if those atoms have different spins, the energy increases. The probability behave in the opposite way, if the energy increases, the probability decreases, and if the energy decreases, the probability increases.

In some applications such as image restoration, what we want to do is to maximize the probability of the configuration, or minimize the energy. For this case, atoms represent the pixels in a case of a binary image, and the goal is to remove the noise in the image, then some pixels must be flipped, causing a change in the system's energy. The energy is minimum when a neighboring pixels have the same value (spin), for this reason, the flipped pixels will be those that have a strong disagreement with its neighbors.

In Boltzmann machines, variables take values $\{0, 1\}$, but still the energy function is the same as the one used in Ising models. Here the only way that two neighbor atoms can make a contribution to the energy is that both have a value of 1. If at least one of them has a value of 0, then the contribution to the energy is zero.

Consider the set of neighbors Z of variable $Y = y$, where y = $\{0, 1\}$, the energy is given by:

$$E(Y = y, Z) = -J \cdot \sum_{k \in Z} y \cdot x_k - B \cdot y$$

Then, the probability of a variable Y to take the value of 1, given an assignment to is neighbors is obtained with the following expression:

$$\begin{aligned} P(Y = 1|Z) &= \frac{P(Y = 1, Z)}{P(Z)} \\ P(Y = 1|Z) &= \frac{P(Y = 1, Z)}{P(Y = 0, Z) + P(Y = 1, Z)} \\ &= \frac{\exp\left[-\beta E(Y = 1, Z)\right]}{\exp\left[-\beta E(Y = 0, Z)\right] + \exp\left[-\beta E(Y = 1, Z)\right]} \end{aligned}$$

but we know that when $Y = 0$, then $E(Y = 0, Z) = 0$, obtaining

$$P(Y = 1|Z) = g(z) = \frac{e^{-z}}{1 + e^{-z}}$$

where $z = \beta E(Y = 1, Z)$. Then, the probability of a variable to take the value of 1 is a sigmoid function, which can be used as an activation function for neural networks.

1.1 Ising Model and the Monte Carlo Method

Consider a $N \times N$ image with black and white pixels. As we explained before, this can be seen as an Ising model, where each pixel represent an atom and the color of the pixel represent the spin of that atom. Let's leave the interaction between neighbor pixels and the strength of the magnetic field as a constant, in such a way that the energy of the image is expressed with equation 1.1.

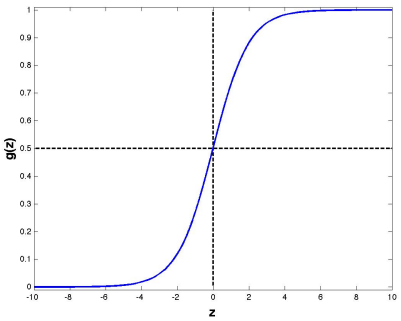
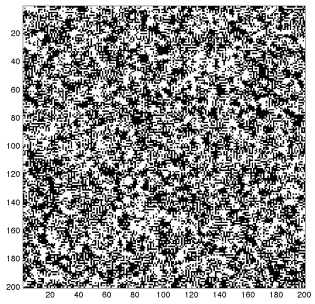


Figure 1.2: Sigmoid function. This function returns values in the range $[0, 1]$

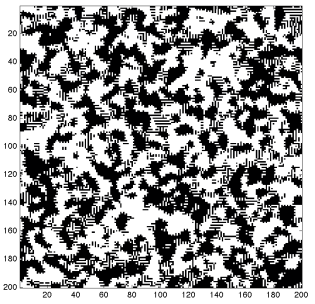
For each pixel we are going to do a random walk trying to minimize the energy of the configuration, following the next steps:

1. Compute the energy $e(X_i)$ for each pixel.
2. Compute the energy $e'(X_i)$ for each pixel, but flipped.
3. Set $\Delta e(X_i) = e'(X_i) - e(X_i)$
4. Flip the node X_i with probability $min(1, \exp(-\beta \Delta e(X_i)))$

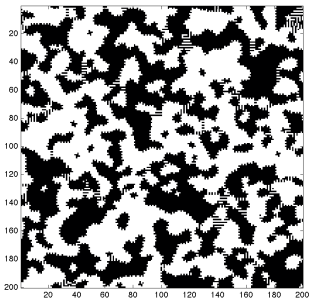
If the energy of the pixel when is flipped is less than the energy of the pixel in its current state, then $\Delta e(X_i) < 0$, thus, $\exp(-\beta \Delta e(X_i)) > 1$ and the pixel is flipped with probability of 1. Figure 1.3 shows the result of the random walk after different number of iterations using $N = 200, \beta = 1, J = 2.0$ and with no influence of any magnetic field, then $B = 0$.



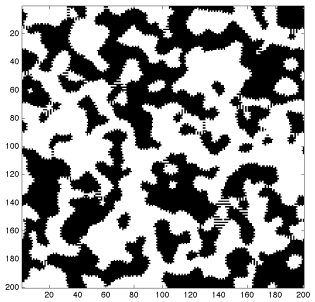
(a) after 1 iteration



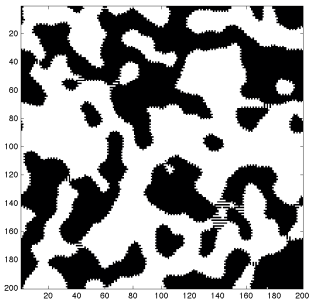
(b) after 5 iterations



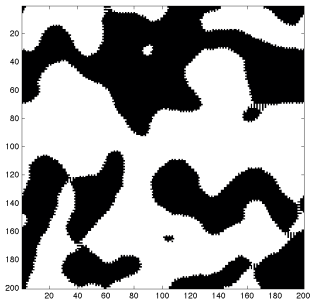
(c) after 25 iterations



(d) after 125 iterations



(e) after 625 iterations



(f) after 3125 iterations

Figure 1.3:

Chapter 2

Image Denoising

Given a noisy binary image Y with pixel values $y_i \in \{-1, +1\}$. The goal is to recover the original image X using the information given by the noisy image. The prior knowledge can be captured using Markov random field as shown in figure 2.1.

A distribution P is a log-linear model over a Markov network $\mathcal{H}$ if is associated with:

- a set of features $\mathcal{F} = \{f_1(D_1), ..., f_k(D_k)\}$, where each D_i is a clique in H.
- a set of weights $w_1, ..., w_k$, such that

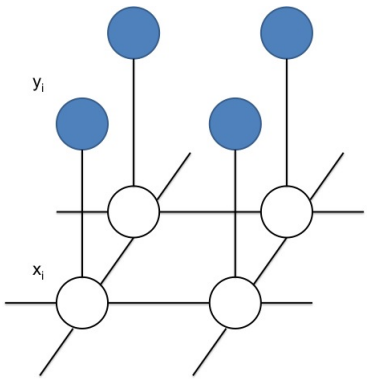


Figure 2.1: Markov rando field for image denoising, where x_i represent the pixels of the noise free image, and y_i the corresponding pixel in the noisy image.

$$P(X_1, ..., X_n) = \frac{1}{Z} \exp\left[-\sum_{i=1}^{k} w_i f_i(D_i)\right]$$

where $\sum_{i=1}^{k} w_i f_i(D_i)$ is the energy function. For the image denoising example, the cliques of the graph are given by the connections between each pixel x_i and its neighbors and by the connection of pixel x_i and the corresponding pixel y_i in the noisy image. According to this, the energy function of our model can be expressed by:

$$E(x, y) = h \sum_{ij} x_{ij} - \beta \sum_{i,j \in Nb(i)} x_i x_j - \eta \sum_{i} x_i y_i$$

where $Nb(i)$ are the indexes of the pixels that are neighbors of pixel i. Then, the distribution P for our model is given by

$$P(x, y) = \frac{1}{Z} \exp(-E(x, y))$$

Models where variables can have two states $\{-1, +1\}$ and interact with its neighbors are called *Ising models*. which have been widely studied in statistical physics as a model for the energy of a physical system involving a system of interacting atoms.

For the case of image restoration we want to obtain the model of minimum energy that give us the maximum probability. In order to achieve this we used an iterative algorithm called *"Iterative Conditional Modes"*. which is explained bellow.

2.1 Iterative Conditional Modes

The Iterative Conditional Modes (ICM) algorithm proposed by Kittler and Föglein in 1984. This algorithm tries to minimize the energy of each node individually in every iteration, when there is no change in the values of the nodes from one iteration to another we can say that the algorithm has converged to a local optimum.

Because neighboring pixels tend to have the same values, it is expected that there will be a strong correlation between them. If there a re single black pixel surrounded by white pixels, is highly probable that this is a flipped pixel. also there will be a strong correlation between each pair x_i, y_i, assuming there is a small level of noise.

Then, the ICM implementation is similar to the gradient descent method, only that ICM tries to minimize each node individually trying to minimize the

energy of the model. Because it consists in a greedy behavior, the algorithm can converge to a local optimum, instead to the global one.

The energy of a single node in iteration k is given by:

$$E\left(x_i|y_i\right) = hx_i - \beta \sum_{j \in Nb(i)} x_i x_j^{(k)} - \eta x_i y_i$$

Thus, the new value of node x_i is obtained according to the following equation:

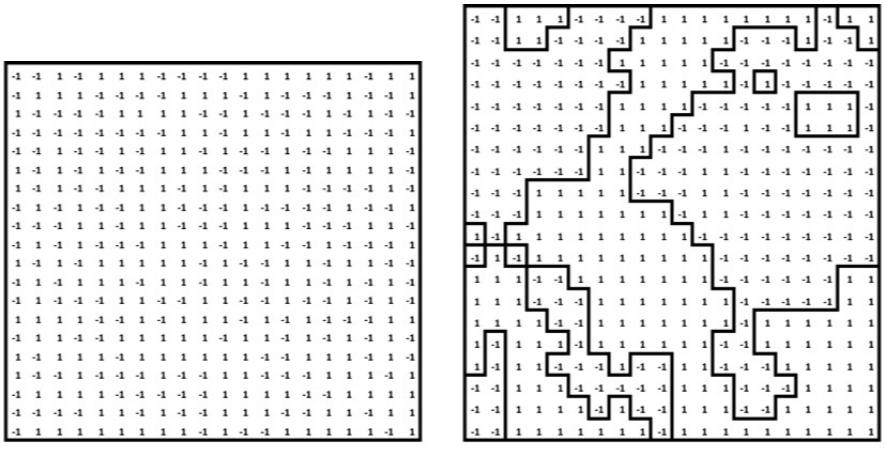
$$x_i^{(k+1)} = \arg \min_{x_i \in \{-1,+1\}} E(x_i|y_i)$$

When a pixel is flipped the term $-\eta x_i y_i$ will be positive, increasing the energy of that pixel. Then, why change it? The answer is given by the term $-\beta \sum_{j \in Nb(i)} x_i x_j^{(k)}$ if this term makes the energy to decrease when the pixel is flipped, then the value of that pixel is changed.

2.2 Results

Creating random values in a 20×20 matrix and running the ICM algorithm we obtain the result shown in figure 2.2, we can see how the algorithm tries to remove the noise, being possible to identify areas with the same label.

The following results were obtained using $h = 0, \beta = 1.0$ and $\eta = 2.1$. Each image was corrupted changing 10% of the pixels. The noisy images are shown in column on the left and the resulting images after applying the ICM algorithm in



(a) Original matrix (b) After the ICM algorithm

Figure 2.2: ICM applied to 20×20 random matrix

column on the right. The initial approximation for each pixel was $x_i = y_i$.

The noise used for testing is a "salt & pepper" noise, created just by flipping the 10% of the pixels. The images shown bellow is the result after 5 iterations of the ICM algorithm, again, because the behavior is similar to the gradient descent, in few iterations the algorithm reaches a local optimum.

Figure 2.3: ICM for image restoration where 99.75% of the pixels agree with the orginal image.

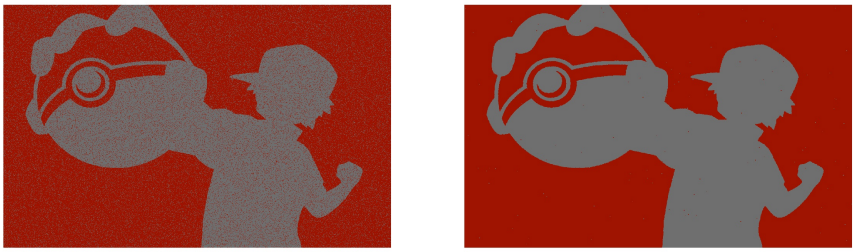


Figure 2.4: ICM for image restoration where 99.85% of the pixels agree with the orginal image.

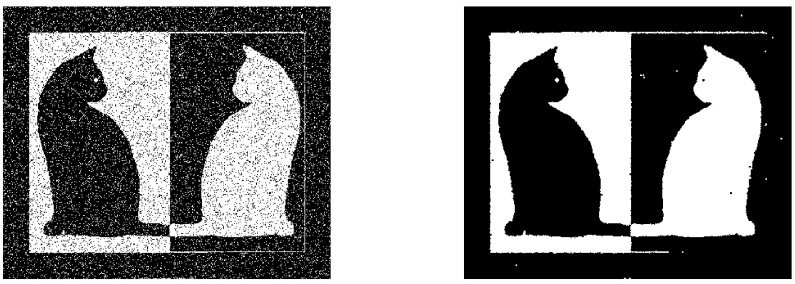


Figure 2.5: ICM for image restoration where 99.26% of the pixels agree with the orginal image.

Figure 2.6: ICM for image restoration where 91.77% of the pixels agree with the orginal image.

Chapter 3

Energy Minimization with Graph Cuts

The problem we want to solve here is to minimize the energy of a directed graph, whose nodes can take two values $\{0, 1\}$ and the cost of the edges represent the energy between a pair of nodes. Consider the case of a binary image, This can be seen as a directed graph G, where each pixel p_i represents a node that is connected by direct edges with its neighbors. The energy of the graph is given by

$$E(G) = \sum_i D(p_i) + \sum_{q \in Nb(p_i)} V(p_i, q), \tag{3.1}$$

where $V(a, b)$ represent the energy potential of the edge that connects node a with node b, and $Nb(a)$ represent the set of neighbors of node a. One typical value for this is $V(a, b) = |a - b|$ which is a semi-metric, because it satisfies the following two properties: $V(a, b) = V(b, a) \geq 0$ and $V(a, b) = 0 \Leftrightarrow a = b$. To be graph presentable, the energy function 3.1 must satisfy the following inequality.

$$V(0, 0) + V(1, 1) \leq V(0, 1) + V(1, 0) \tag{3.2}$$

Now, suppose what we have is noisy image, and we want to obtain the original image, then we must label each pixel with its correct value. Thus, some pixel labeled as 0 must be flipped to 1, or in the other way around. This can be seen as energy minimization problem, first we must add a node $\alpha = 0$ as a source and a node $\beta = 1$ as a sink, as shown in figure 3.1. Given this arrangement, there are three kind of edges, edge t_p^α that goes from α to pixel p, edge t_p^β from pixel p to β, and finally edge $e_{p,q}$ that goes from pixel p to pixel q.

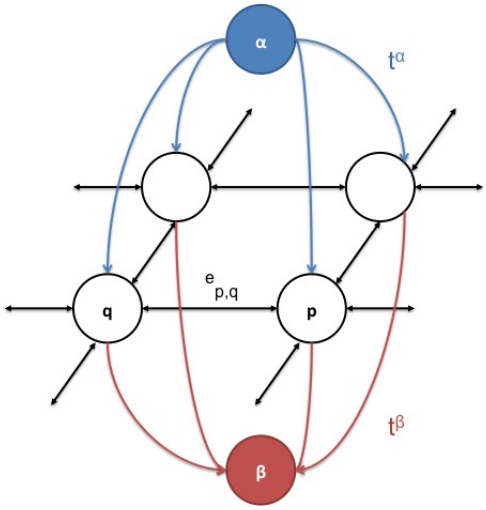


Figure 3.1: Graph reprsenting and image, with nodes α and β added to it.

Edges t_p^α and t_p^β are called t-links, and edges $e_{p,q}$ are called n-links. The disagree between pixel p and node z, where $z = \{\alpha, \beta\}$ is obtained with

$$D(z, p) = (z - p)^2$$

The weights associated to each edge are given in table 3.1.

edge	weight
t_p^α	$D(\alpha, p) + \sum_{q \in Nb(p)} V(\alpha, q)$
t_p^β	$D(\beta, p) + \sum_{q \in Nb(p)} V(\beta, q)$
$e_{p,q}$	$V(p, q)$

Table 3.1: Weights of each one of the edges in the graph

The goal is to separate the graph in two parts S and T, in such a way that $\alpha \in S$ and $\beta \in T$, and all the pixels in the S side will be labeled with the value of α, and the pixels in the T side will be labeled with the value of β. This can be done by obtaining the minimum cut in the graph,or by obtaining the maximum flow in the graph from α to β.

The cut in a graph is the sum of all edge weights that goes from S to T. There are different algorithms that can be used to find the maximum flow in a network, one of those is the Ford-Fulkerson algorithm that will be described shortly.

3.1 Ford-Fulkerson Algorithm

A flow network $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has nonnegative capacity $c(u, v) \geq 0$. If $(u, v) \notin E$, then for convenience we define $c(u, v) = 0$.There are two vertices called the source s and the sink t. The goal is to calculate the maximum flow from s to t. The maximum flow can be seen as the maximum amount of water that can go from the source to the sink, using graph edges as pipes. A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following properties:

- **Capacity Constraint:** $\forall u, v \in V$, we require $0 \leq f(u, v) \leq c(u, v)$.
- **Skew Symmetry:** $\forall u, v \in V f(u, v) = -f(v, u)$
- **Flow Conservation:** $\forall u \in V : u \neq s \bigwedge u \neq t \Rightarrow \sum_{v \in V} f(u, v) = 0$

The Ford-Fulkerson algorithm consists of three steps. The first step is try to find a path from the source to the sink. The flow can go trough an edge (u, v) if and only if $c(u, v) - f(u, v) > 0$. the second step is to find the residual capacity of the path p founded in step 1. The residual capacity is the maximum amount by which we can increase the flow on an edge in p and is defined by:

$$c_f(p) = \min\{c(u, v) - f(u, v) : (u, v) \text{ is on } p\}$$

Finally, we must add the residual capacity $c_f(p)$ to the flow in all the edges in p.

$$f(u, v) = f(u, v) + c_f(p) : (u, v) \text{ is on } p$$
$$f(v, u) = f(v, u) - c_f(p) : (u, v) \text{ is on } p$$

Algorithm 1 shows the steps that must be followed. the complexity of this algorithm depends in the way we find for the path p, one solution is to use a

Algorithm 1 Ford-Fulkerson Algorithm

```
1: maxflow ← 0
2: while there is a path p from s to t do
3:    c_f(p) = min{c(u,v) − f(u,v) : (u,v) is on p}
4:    maxflow ← maxflow + c_f(p)
5:    for all edges (u,v) ∈ p do
6:       f(u,v) ← f(u,v) + c_f(p)
7:       f(v,u) ← f(v,u) − c_f(p)
8:    end for
9: end while
10: return maxflow
```

depth first search or a breadth first search, which have complexity $O(V + E)$.

Figure 3.2 shows the different iterations of the Ford-Fulkerson algorithm. First, we have no flow in the edges of the network. Then the path $p = s - A - t$ is founded and the residual capacity $c_f(p)$ is 10, that corresponds to the flow that can pass trough node A to the sink, then the flow in all edges in the path is augmented by 10. In the next iteration the path $s - B - t$ is founded and the flow is augmented by 10 in all the edges in the path. Finally, the path $s - A - B - t$ is founded, and a residual capacity of 5 is added to the flow in the edges of the path. The maximum flow is the sum of the residual capacities in each path founded, for this case we have that eh maximum flow is 25.

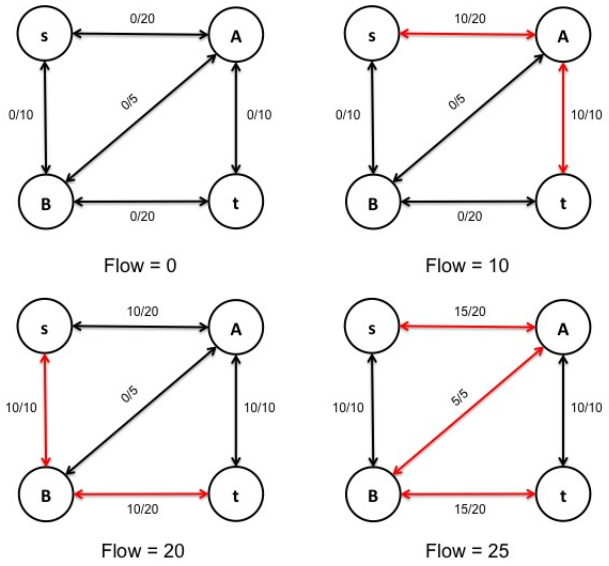


Figure 3.2: Ford-Fulkerson algorithm

Now, if we run a DFS from the source passing trough the edges where flow can pass and paint all the visited nodes in blue, and the rest of the nodes in red, we obtain the figure 3.3. The sum of the capacities of the edges that goes from a blue node to a red node is precisely the minimum cut of the graph. The blue nodes are those in the S side and the red nodes are those in the T side. The minimum cut can be defined using the following expression:

$$minCut = \sum_{u \in S} \sum_{v \in T} c(u, v)$$

Getting back to the pixel labeling and the graph shown in figure 3.1 we must the cut that minimum energy as possible, and applying the same concept of the example, where the nodes were labeled with two colors, red and blue, we then must label the nodes in the S side with α and the nodes in the T side with β.

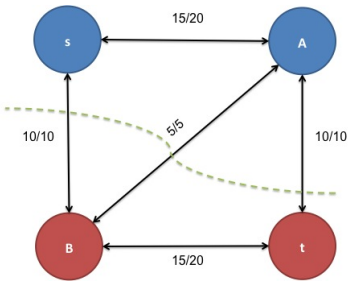
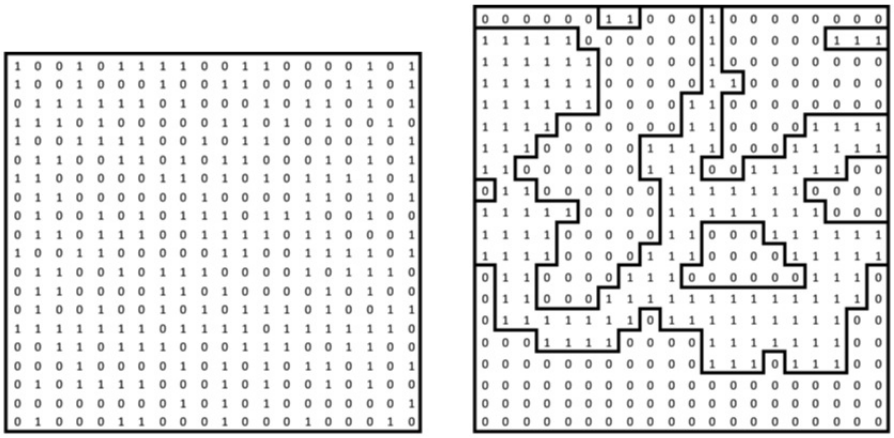


Figure 3.3: Minimum cut of the graph. The green line represent the minimal cut that separates the graph in two parts, in such a way that $s \in S$ and $t \in T$

3.2 Results

If we create a random matrix of size 20×20, the result obtained by the Cut-Graph Energy Minimization algorithm groups set of values with the same label. This is because there are certain patterns in the original matrix, some areas have more zeros than ones, and some have more ones than zeros. These cells of these areas tend to obtain the same label as shown in figure 3.4.



(a) Original image (b) Result after Ford-Fulkerson algorithm

Figure 3.4: Graph-Cut Energy Minimization algorithm applied to a random 20×20 matrix

The algorithm was implemented in C using the open source library OpenCV to read and save the images.

Figure 3.5: Graph cut energy minimization with 99.44% of the pixels agree with the orginal image.

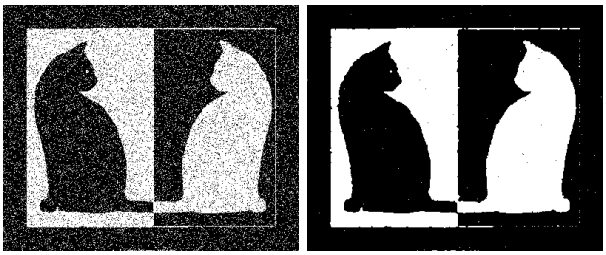


Figure 3.6: Graph cut energy minimization with 99.19% of the pixels agree with the orginal image.

Figure 3.7: Graph cut energy minimization with 88.19% of the pixels agree with the orginal image.

Bibliography

[1] Boykov, Y., Veksler O. and Zabih. R (1999) *Fast Approximate Energy Minimization via Graph Cuts*. Cornell University.

[2] Kolmogorov, V. and Zabih, R. (2004) *What Energy Functions Can Be Minimized via Graph Cuts*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No 2.

[3] Boykov, Y., Veksler O. (2006) *Graph Cuts in Vision and Graphichs: Theories and Applications*. Handbook of Mathematical Models in Computer Vision. Springer.

[4] Bishop. C (2009) *Pattern Recognition and Machine Learning*. Chapter 8. Probabilistic Graphical Models. Springer.

[5] Cormen, T., Leiserson, C., Rivest, R. and Stein, C. (2009) *PIntroduction to Algorithms*. Chapter 26. Maximum Flow. The MIT Press.