Markov Networks for Image Denoising and Stereo

Dmitry Adamskiy

October 21, 2017

1 Image Denoising

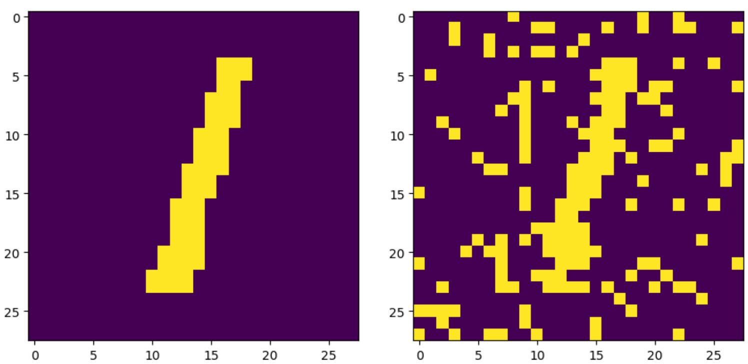

Figure 1: Clean and noisy images

Consider a problem of denoising a (binary) image (see Figure 1). We will try to build a probabilistic model of what the noisy images might look like so that we can use the most probable settings of true pixel values under this model as the denoised image.

1.1 The Model

We model the collection of pixels as Markov Network. The graph of the network is a lattice of size $M \times N$ (in the digit example above $M = N = 28$), corresponding to the 'true' (hidden) pixels of the image, where additionally each node has a corresponding noisy (visible) pixel node connected to it (see Figure 2). There are two types of independence assumptions in this model:

1. Pixels become noisy independent of each other. So one noisy pixel is independent of the rest of the graph given its true value.
2. The true pixel values are (conditionally) independent of the other true values given the neighbours.

The second assumption is quite a big simplification (compare it to Boltzmann machines!).

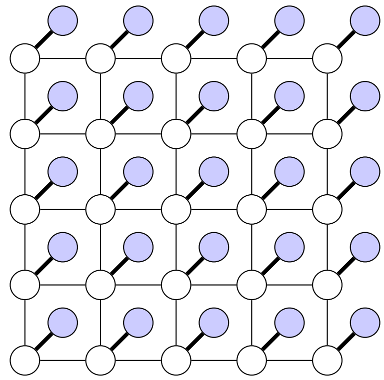

Figure 2: Graphical model of a noisy image. Grey nodes are the noisy (and observed) pixels Y_i, white nodes are the true (and hidden) pixels X_i.

The distribution then factorises as:

$$P(X,Y) \propto \prod_{i=1}^{D} \phi'(X_i, Y_i) \prod_{i\sim j} \psi'(X_i, X_j)$$

where $\prod_{i\sim j}$ means taking the product over the edges of the lattice graph. We are interested in maximising $P(X|Y)$ as a function of X which is equivalent to maximising $P(X,Y)$ or $\log P(X,Y)$. We will consider $\phi'(X_i,Y_i) = e^{-\phi(X_i,Y_i)}$ and $\psi'(X_i,X_j) = e^{-\psi(X_i,X_j)}$ so that we are actually going to *minimise*

$$\arg\min_X E(X) := \arg\min_X \sum_i \phi(X_i, Y_i) + \sum_{i\sim j} \psi(X_i, X_j)$$

Let's say that pixels take values in {0,1} (Note: in the slides it is {-1,+1}). Y_i are fixed so we can view ϕ is a function of one variable (but now dependent on the pixel), $\phi_i(X_i) = \phi(X_i, Y_i)$. To specify the model we need to provide the values for $\phi_i(0)$, $\phi_i(1)$, $\psi(0,0)$, $\psi(0,1)$, $\psi(1,0)$ and $\psi(1,1)$. The natural idea is to make the image smooth, so that $\psi(X_i,X_j) = W|X_i - X_j|$, where W is a parameter of the model.

Now we need to specify $\phi_i(0)$ and $\phi_i(1)$. It is natural to favour images agreeing with the noisy one, so that $\phi_i(X_i) = \phi(X_i,Y_i) = B|X_i - Y_i|$, where B is another parameter of the model. Functions ϕ and ψ are called unary and binary (or pairwise) potentials.

1.2 Iterated Conditional Modes

How do we find the most like setting of X? One simple algorithm is to start at some initial guess X^*, fix all the pixels and find the minimum with respect to just one at a time and iterate over the pixels (in some random order):

$$X_k^{new} = \arg\min_{X_k} E(X_1^*, X_2^*, \ldots, X_k, \ldots, X_D^*).$$

This is particularly easy to do in our case as there are only two values to choose from

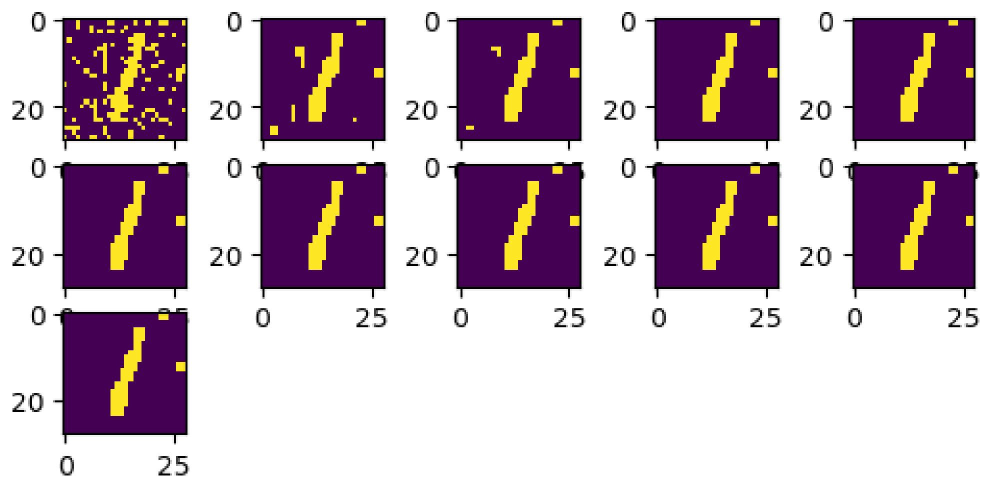

Figure 3: Output of successive iterations of ICM

and $E(X)$ depends on X_k only locally, so to set X_k we need to find the sign of

$$E(X_1,\ldots,X_{k-1},0,X_{k+1},\ldots,X_D) - E(X_1,\ldots,X_{k-1},1,X_{k+1},\ldots,X_D) =$$
$$\phi_k(0) + \sum_{k\sim j}\psi(0,X_j) - \phi_k(1) - \sum_{k\sim j}\psi(1,X_j) =$$
$$(2Y_k-1)B + W\sum_{k\sim j}(2X_j-1).$$

Figure 3 shows an output of one run of this algorithm starting with noisy image, $X^* = Y$ for the model parameters $W = 10$, $B = 1$. It converges to some (local) optimum of $E(X)$. The output of the algorithm depends on the order in which we flip the pixels so that if we run it several times we get different results. Can we find the global optimum? It turns out that for this problem we can and we'll describe the approach in the next section.

1.3 Minimising Energy and Max-Flow/Min-Cut problem

1.3.1 Detour: Max-Flow and Min-Cut

The Maximum Flow Problem is a well-studied problem in Graph Theory. Let $G = (V, E)$ be a directed graph. The capacity is a nonnegative function on the edges $c : E \to \mathbb{R}^+$. Given a graph, the capacity and two nodes s and t (called *source* and *sink*), the goal is to find the maximum flow from s to t. The flow is also a nonnegative function on the edges, $f : E \to \mathbb{R}^+$, such that

1. $f(e) \le c(e)$, $\forall e \in E$ ("flow does not exceed the capacity")
2. $\sum_{u:(u,v)\in E} f_{uv} = \sum_{u:(v,u)\in E} f_{vu}$, $\forall v \in V\backslash\{s,t\}$ ("what flows in the node has to flow out – except for source and sink")

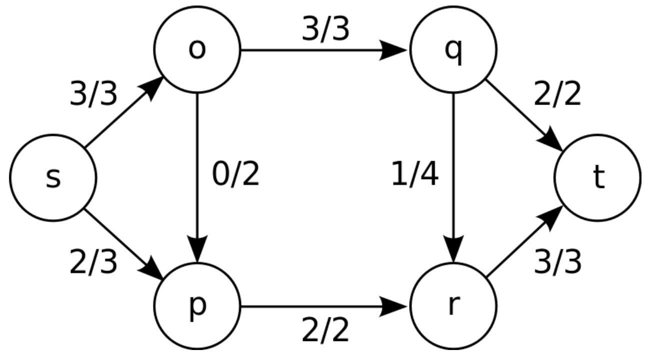

Figure 4: Maximum flow from s to t. For each edge X/Y denotes a flow of X and a capacity of Y. The value of the flow is 5.

The value of the flow is $\sum_{i:(s,i)\in E} f(s,i)$. Maximum flow is the one with the maximum value among all the flows from s to t. See Figure 4 for an example of a graph with a (maximum) flow. There is a theorem establishing an equivalence of finding the maximum flow with finding the minimum s-t cut. An s-t cut is a partition of graph nodes into two sets, s-set and t-set, such that source is in s-set and sink is in t-set. A value of the cut is the sum of capacities of all the edges going from s-set to t-set. A minimum s-t-cut is the one with a minimum value. There are polynomial time algorithms for solving these problems and some of them are tailored for graphs typical to computer vision problems (and run in almost linear time on those).

1.3.2 Reduction to min-cut

How does all this graph machinery help us solve the energy minimisation problem? Suppose that the energy function satisfies the following constraints:

- $\phi_i(0) \geq 0$, $\phi_i(1) \geq 0$, $\forall i$
- $\forall \{i,j\}$, $\psi_{ij}(0,0) = \psi_{ij}(1,1) = 0, \psi_{ij}(1,0) \geq 0, \psi_{ij}(0,1) \geq 0$.

(note that the energy we defined for the denoising task has this property). We can then write the energy as

$$E(X) = \sum_{i\in V}(X_i\phi_i(1) + (1-X_i)\phi_i(0)) + \sum_{(i,j)\in E}(X_i(1-X_j)\psi(1,0) + X_j(1-X_i)\psi(0,1)) \quad (1)$$

Then consider the following directed graph $G' = (V', E')$. The vertex set is $V' = V \cup \{s,t\}$. For each edge $(i,j) \in E$ make two directed edges (i,j) and (j,i) in E'. Make directed edges (s,v) and (v,t) for each vertex $v \in V$.

Set capacities as follows:

- $c(s,i) = \phi_i(1)$, $c(i,t) = \phi_i(0)$, for each $i \in V$
- $c(i,j) = \psi_{ij}(0,1)$, $c(j,i) = \psi_{ij}(1,0)$, for each edge $(i,j) \in E$

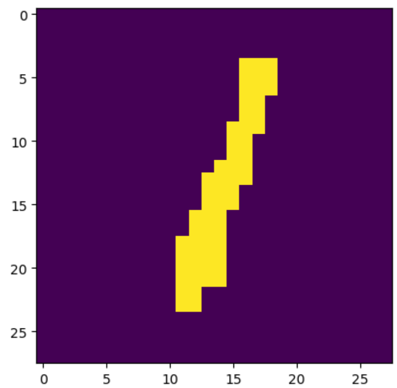

Figure 5: Denoised image with the energy function minimised using min-cut approach

Figure 6: A stereo pair of images

Now every s-t-cut of graph G' could be interpreted as setting $X_i = 0$ for $i \in s$-set and $X_i = 1$ for $i \in t$-set. And the value of the cut is exactly the value of energy (1) under such setting. Thus we could use any max-flow/min-cut algorithm for solving the energy minimisation problem. Figure 5 shows the result of energy minimisation for the denoising problem above.

It turns out that using a reparametrisation trick we could weaken the condition on energy to

$$\psi_{ij}(0,0) + \psi_{ij}(1,1) \le \psi_{ij}(0,1) + \psi_{ij}(1,0) \tag{2}$$

This condition is called *regularity* (see this paper for details).

2 Stereo

Now that we are equipped with the min-cut technique, let's try to solve a more ambitious problem of depth perception from a stereo pair of images. Figure 6 shows a pair of images taken from slightly different angles.

The further away the object is from the photographer, the smaller horizontal displacement for the pixels of that object is. As an illustration, consider shifting the (central bit of the)

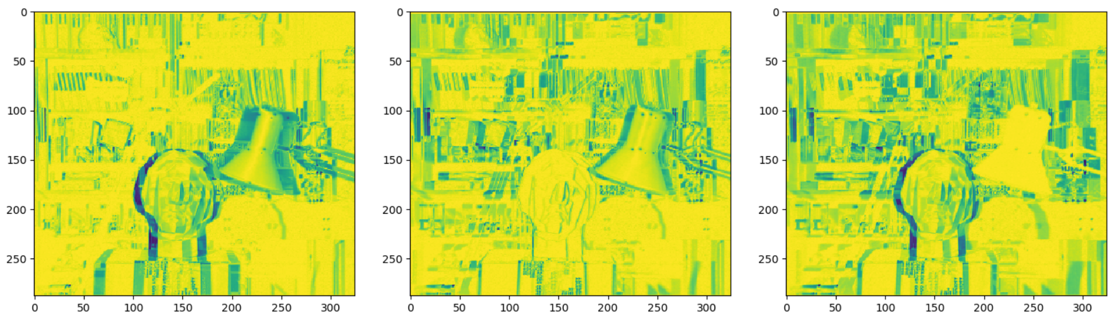

Figure 7: Similarity maps for various displacements

left image 12, 20 and 28 pixels to the left (see Figure 7) and then comparing corresponding pixels of two images. We can see that for the displacement of 12 pixels, the camera in the background becomes aligned, the 20-pixel displacement corresponds to matching plaster heads in the foreground and the 28-pixel one matches the lamps on the two images (note that if you look just at the left image, it is not obvious whether the lamp is actually in front of the bust!). So learning the displacements corresponds to learning the depth map of the image.

How can we build an energy function for such a scenario? The variables X_i are now the displacements of i-th pixel. We could define unary potentials $\phi(X_i)$ to be the distance between i-th pixel of the left image and the pixel of the right image after the horisontal shift:

$$\phi(X_i) = D(I_{left}(i_x, i_y), I_{right}(i_x + X_i, i_y)).$$

where (i_x, i_y) are the coordinates of the $i - th$ pixel of the left image and $D(\cdot, \cdot)$ is a colour difference function.

The binary potentials as before encourage the smoothness of the labeling:

$$\psi(X_i, X_j) = W\mathbb{1}\{X_i \neq X_j\}.$$

Now we would like to apply the min-cut technique and get a solution. However, we have a problem: the displacements X_i are no longer binary variables.

2.1 α-expansion

Unfortunately, the problem for non-binary variables is no longer tractable, but we can use a simple approximation technique. Let $K = \{K_1, K_2, \ldots, K_T\}$ be the set of all possible displacements. Start with an initial guess $X^0 = (X_1^0, X_2^0, \ldots, X_D^0)$, where $X_i^0 \in K$. Now on each iteration of the algorithm fix the label $\alpha \in K$. Consider a new energy minimisation problem for the new binary variables Z_i where $Z_i = 0$ corresponds to $X_i = X_i^0$ and $Z_i = 1$ corresponds to $X_i = \alpha$. Now for this problem the potentials are

- $\phi_i'(0) = \phi_i(X_i^0)$, $\phi_i'(1) = \phi_i(\alpha)$
- $\psi_{ij}'(0,0) = \psi_{ij}(X_i^0, X_j^0)$, $\psi_{ij}'(1,1) = \psi_{ij}(\alpha, \alpha)$, $\psi_{ij}'(0,1) = \psi_{ij}(X_i^0, \alpha)$, $\psi_{ij}'(1,0) = \psi_{ij}(\alpha, X_j^0)$.

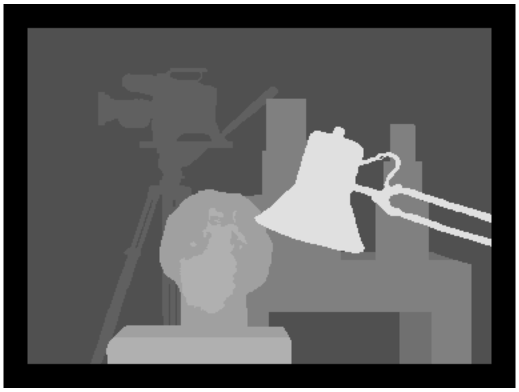

Figure 8: Ground truth depth map for the stereo pair

The regularity condition (2) then becomes

$$\psi_{ij}(\alpha, \alpha) + \psi_{ij}(\beta, \gamma) \leq \psi_{ij}(\beta, \alpha) + \psi_{ij}(\alpha, \gamma)$$

for all $\alpha, \beta, \gamma \in K$. If $\psi(\alpha, \alpha) = 0$ and $\psi(\beta, \gamma) = \psi(\gamma, \beta)$ as in our case, the condition becomes triangle inequality.

Using α-expansion allows one to build depth maps as shown in Figure 8.