# MRF for Denoising

Liliana Lo Presti

# Outline

- Recap on Regression Problem
- Variational Methods and link to energy-based models
- Maximum a Posteriori
- Markov Random Fields – ICM and Graph-Cut

# Recap on Regression

- The goal of regression is **to produce a function** f(x) **that passes close** a set of data points $\mathbf{d_k}$ placed at locations $x_k$ such that

$$f(x_k) \sim d_k$$

**Input to the training algorithm:** $\{(x_k , d_k)\}_k$ a set of locations and corresponding expected output

**Output of the training algorithm:** parameters of function f(x)

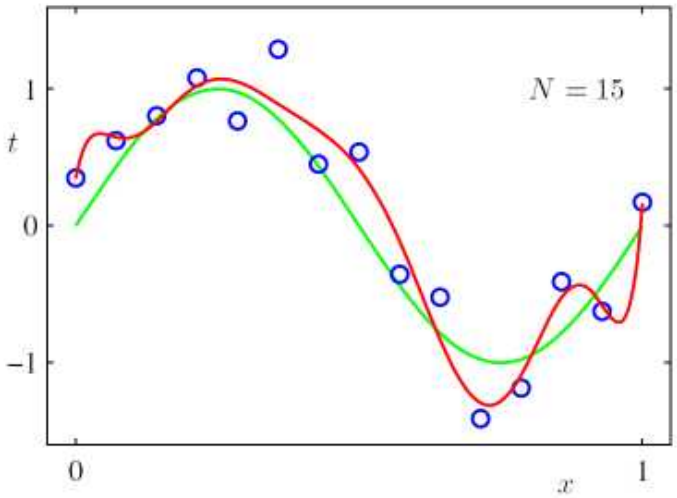
We find the parameters of f by minimizing the average error we committ when using f:

$$E_{\mathrm{D}} = \sum_k \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2$$

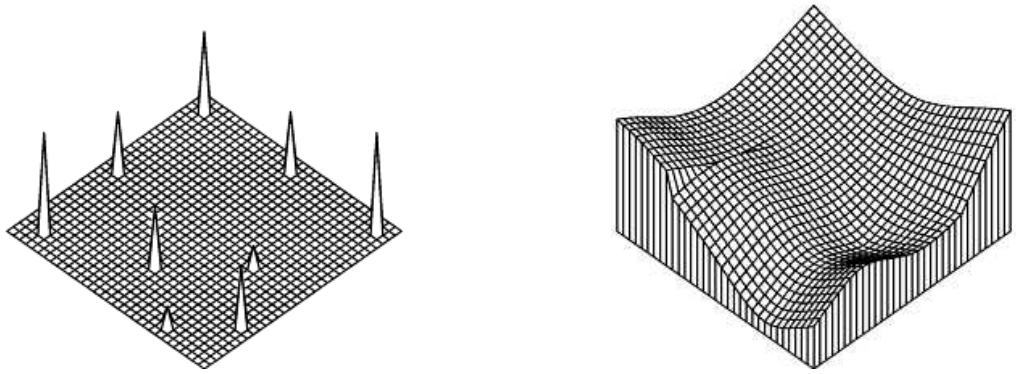
**We use Least square minimization**

# Recap on Regression

- This problem takes different name. It is called Regression in Statistics and machine learning




1D example: x is the input, t is the target
N is the number of data points
The 2 functions are possible approximations of the data



2D example: 9 2D samples, the function approximate a surface

# Regularization

$$\sum_k |\mathbf{w}_k|^p$$

- In **the regularized least squares problem,** the penalty added to constrain the parameters changes based on p, and assumes different meaning

- **If p = 2,** we use the L2-norm and we talk about **ridge regression** (Tikhonov regularization), it admits a **closed-form solution** in which we add a fraction of the identity matrix to a covariance matrix to be sure it has a full rank

- **If p = 1**, we use the L1-norm and we talk about the **LASSO Regression** (LASSO stands for Least Absolute Selection and Shrinkage). It does not have a convenient closed-form solution. The solution is typically found using quadratic programming or more general convex optimization methods.

- LASSO regression forces more entries of w to equal 0 (namely, the solution is sparse). The Tikhonov regularization forces entries of w to be small but not necessarily 0.

# Variational Methods

- This kind of methodology is also used in a family of methods named Variational Methods (VM) where the variables are however continuous values (thus we will have **integrals** in the loss function)
- Variational Methods are among the most classical techniques for optimization of cost functions in higher dimension.

- Many challenges in Computer Vision and in other domains of research can be formulated as variational methods.

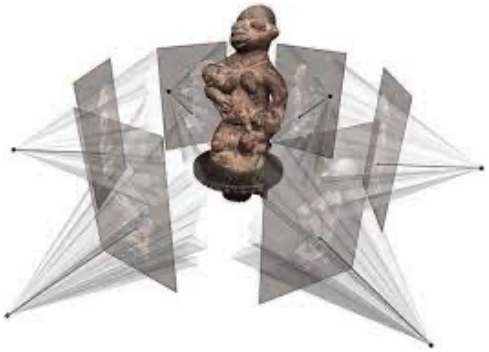
- Examples include **denoising**, deblurring, image segmentation, tracking, optical flow estimation, depth estimation from stereo images or 3D reconstruction from multiple views.

- The first variational method was demonstrated by Bernoulli in 1697, Euler and Lagrange later provided more general solutions

# VM in Computer Vision…


- Image Segmentation

  Geman & Geman, ‘84, Blake & Zisserman ‘87…



- Multiview Stereo Reconstruction

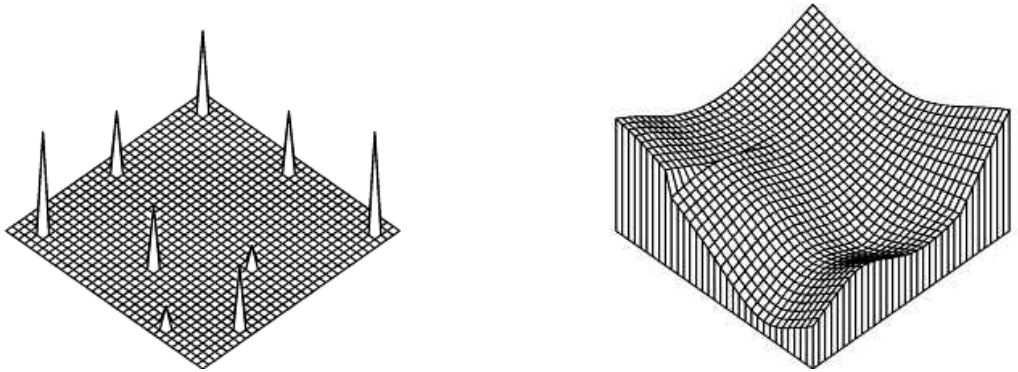
  Faugeras ‘98,  Yezzi& Soatto, ‘03 ….

- Optical Flow Estimation

  Horn & Schunk ‘81, Brox et al. ‘04, …

These problems imply optimization of non-convex function

# Variational Methods and Energy minimization



- Several surfaces can fit the given data points

- However, we want to control the smoothness of these surfaces
- For doing this, we **introduce norms** (measures) on function derivatives to find minimal energy solutions to these norms
- ***Learning*** consists in minimizung an energy function that **associates low energies to correct values, and higher energies to incorrect values.**

# Variational Methods

- Many vision tasks are naturally posed as energy minimization problems on a rectangular grid of pixels, where the energy comprises a *data term* and a *smoothness* term:

$$E(u) = E_{data}(u) + E_{smoothness}(u)$$

- The data term $E_{data}(u)$ expresses our goal that the optimal model **u be consistent with the measurements**.
- The smoothness energy $E_{smoothness}(u)$ is derived from **our prior knowledge about plausible solutions**.

- For instance:
  - **In the Denoising field,** given a noisy image I'(x,y), where some measurements may be missing or corrupted, we want to recover the original image I(x, y), which is typically assumed to be smooth.
  - On one hand, we expect the original image I **does not differ** too much from I', which cab be expressed choosing e proper form of $E_{data}$
  - On the other hand, local smoothness of gray levels is expected and should be enforced by choosing a proper form of $E_{smoothness}$

# Variational Methods

- Let us assume we want to fit a model from a set of samples with some **smoothness properties**. We can do it by formulating the problem as an energy minimization problem
- First, we need **to measure the smoothness of a model**, which can be done by using norms on function derivatives. **In two dimensions**:

$$\mathcal{E}_1 = \int f_x^2(x,y) + f_y^2(x,y)\, dx\, dy = \int \|\nabla f(x,y)\|^2\, dx\, dy$$

Functional integrating the gradient magnitude over the two dimensions

$$\mathcal{E}_2 = \int f_{xx}^2(x,y) + 2f_{xy}^2(x,y) + f_{yy}^2(x,y)\, dx\, dy,$$

Functional integrating the second-order derivatives. It is also called *thin-plate spline*

Intuitivelly, these functionals measure **the variation (non-smoothness) in a function** and can be used as regularizers in fitting problems, **forcing the solution to be smooth**

$$\mathcal{E}_1 = \int f_x^2(x,y) + f_y^2(x,y)\,dx\,dy = \int \|\nabla f(x,y)\|^2\,dx\,dy$$

# Discrete Energy Minimization

$$\mathcal{E}_2 = \int f_{xx}^2(x,y) + 2f_{xy}^2(x,y) + f_{yy}^2(x,y)\,dx\,dy,$$

- We also need to discretize the smoothness functionals.
- Considering how derivatives of discrete functions are computed, the simplest expression of these functionals is (in the book, you find more complete versions):

$$E_1 = \sum_{i,j} [f(i+1,j) - f(i,j)]^2 + [f(i,j+1) - f(i,j)]^2$$

$$E_2 = \sum_{i,j} [f(i+1,j) - 2f(i,j) + f(i-1,j)]^2 + \sum_{i,j} 2[f(i+1,j+1) - f(i+1,j) - f(i,j+1) + f(i,j)]^2$$
$$+ \sum_{i,j} [f(i,j+1) - 2f(i,j) + f(i,j-1)]^2$$

# Variational Methods for images

- In most computer vision applications, the functions we are trying to model are only piecewise continuous
- For instance, in color images, changes in the objects' appearance (the surface color, or albedo) introduces discontinuities.
- This is also true in depth maps or optical fields
- To account for these discontinuities, we can locally control the function continuity [Terzopoulos (1986b)]

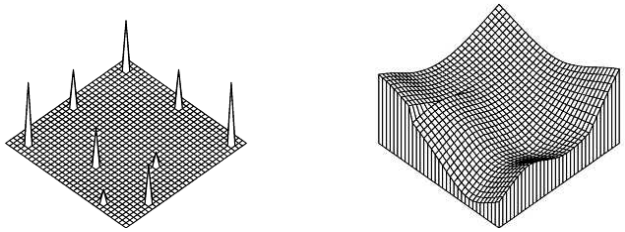
$$\mathcal{E}_{\mathrm{CC}} = \int \rho(x,y)\{[1-\tau(x,y)][f_x^2(x,y)+f_y^2(x,y)] + \tau(x,y)[f_{xx}^2(x,y)+2f_{xy}^2(x,y)+f_{yy}^2(x,y)]\}\,dx\,dy.$$

*gradient*

*thin-plate spline*

- Where the functions $\rho(x,y) \in [0,1]$ and $\tau(x,y) \in [0,1]$ control the continuity and tension (how flat) of the surface

# Variational Methods for images



- In addition to the smoothness term, variational problems also require a data term (or data penalty).
- For regression, this term measures the model error on the dataset

$$\mathcal{E}_{\mathrm{D}} = \sum_i [f(x_i, y_i) - d_i]^2$$

- Thus, **the function to minimize for estimating the model parameters** is

$$\mathcal{E} = \mathcal{E}_{\mathrm{D}} + \lambda \mathcal{E}_{\mathrm{S}}$$

- Where, the last term is one of the two previous functionals (or variants), enforcing smoothness of the model and representing a *smoothness penalty*

# Bayesian perspective

- Regularization involves the minimization of energy functionals defined over (piecewise) continuous functions
- An alternative technique is to formulate a Bayesian or generative model, which separately models the noisy image formation (measurement) process, as well as assuming a statistical prior model over the solution space
- **We will first see that also using a Bayesian approach we will end with minimizing an energy function**
- Then, we will see Markov Random Fields (MRF), a generative model used in computer vision. The log-likelihood of a MRF can be described using local neighborhood interaction (or penalty) terms

# Maximum a Posteriori

- Let's say we have some unknown x (x may have several meaning here, may represent the model parameters) and can observe measurements y. The variables x and y are generally multi-dimensional
- We define the **posterior distribution p(x|y)** as a density function (it measures how probable to have the value x given the observed data)
- We call **p(y|x) the likelihood** over the available data (measuring how well the knowledge of x explains the observed measurements)
- We denote **p(x) the prior** (encoding our knowledge of x without taking into account any observation)

# Maximum a Posteriori

- Based on the Bayes' rule:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}$$

- And taking the negative logarithm:

$$-\log p(\mathbf{x}|\mathbf{y}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + C$$

- Which is the **negative posterior log likelihood**

# Maximum a Posteriori

- To find the most likely value of x (MAP) given y, we want to minimize this negative posterior log likelihood

$$-\log p(\mathbf{x}|\mathbf{y}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + C$$

- This expression *has a term depending on the data y* and *a term depending on the prior on x.* It resambles an energy minimization problem

$$E(\mathbf{x}, \mathbf{y}) = E_{\mathrm{D}}(\mathbf{x}, \mathbf{y}) + E_{\mathrm{P}}(\mathbf{x})$$

- Thus, ***minimizing the energy or finding the MAP solutions are equivalent***

# Markov Random Fields

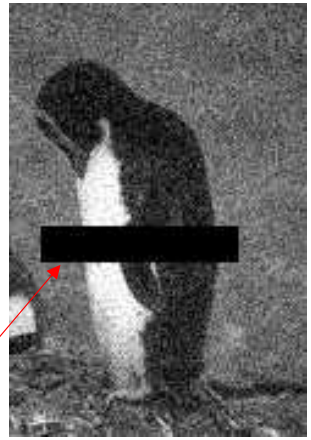
- Markov random fields are ***probabilistic models defined over two or three-dimensional pixel or voxel grids***. MRF are one special case of the more general family of *graphical models*
- When using MRF for image processing applications, *the unknowns x are the set of output pixels (**what we want to estimate**), while y are the input pixels (**what we observe**)*

$$\mathbf{x} = [f(0,0) \dots f(m-1, n-1)] \qquad \mathbf{y} = [d(0,0) \dots d(m-1, n-1)]$$

- MRF are largely used for denoising and segmentation.
- Thus, **the meaning of x depends on the problem we want to solve**.
- Indeed, x may be the denoised pixel value or the label to assign to each pixel.

# Markov Random Fields - Examples

Y

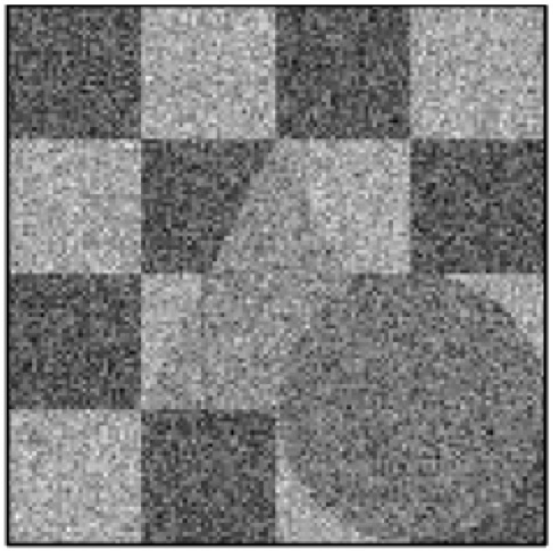


X

Missing points

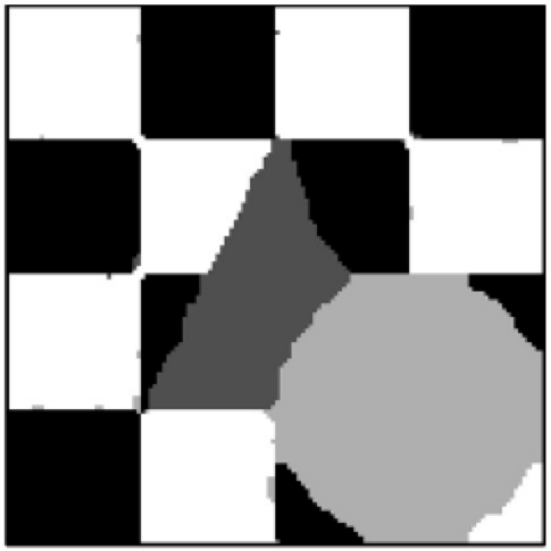
Denoising and inpainting applications

Y



Input noisy image (SNR=5dB)

X



CNN-MRF Segmented Result

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=159243cf25fef f7c85d1bb37c03b3ee61fa4ad19

# Markov Random Fields

- As already said, we need to formulate the problem in such a way to get a funtion in the form

$$E(\mathbf{x}, \mathbf{y}) = E_{\mathrm{D}}(\mathbf{x}, \mathbf{y}) + E_{\mathrm{P}}(\mathbf{x})$$

- In MRF, the data penalty **$E_D$ represents the link between x and y,** and depends on the problem to solve
- The prior energy **$E_P$ expresses constraints on the pixels in the same neighborhood.** It is generally expressed as

$$E_{\mathrm{P}}(\mathbf{x}) = \sum_{\{(i,j),(k,l)\} \in \mathcal{N}(i,j)} V_{i,j,k,l}(f(i,j), f(k,l)),$$

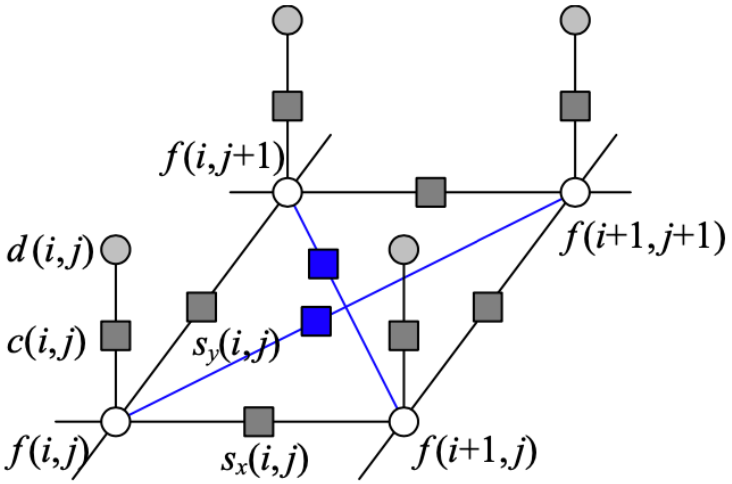
- Depending on V, this penalty sometimes correspond to use a Boltzmann distribution as prior p(x)

$$p_i = \frac{1}{Q} e^{-\varepsilon_i/(kT)}$$

K is the Boltzmann constant, T is the temperature, epsilon is the energy of the state i, Q is used to normalize and get a valid distribution

(The Boltzmann distribution is a probability distribution that gives the probability of a certain state as a function of that state's energy and temperature of the system to which the distribution is applied)

# Markov Random Field as Graphical Model




**Random Variables:**

**f(i,j)** are the **unknown random variables** we want to estimate

**d(i,j)** are the **observed variables**

**Links (factors):**

**s(i,j)** are representing the interactions between the unkwon variables, and are determines the **interaction Potentials**. Their number depends on the kind of neighboor we consider ($N_4$ or $N_8$)

**c(i,j)** denote the data penalty functions and **link observed and unknown variables,** are called **unary Potentials**

The factors are (improper) probability functions whose product is the (un-normalized) posterior distribution

Neighbors define **cliques** within the graphical model.
Cliques are basic units that capture all possible dependencies that are possible and not to be missed.
In undirected graphical models, **the joint distribution can be factorized into a product of clique potentials (unary and interaction)**
These clique potentials represent a notion of “goodness” or “compatibility” of the variables.

# Binary Markov Random Fields

- In a Binary Markov random field the variable x is binary.
- It may be used **to denoise black and white scanned images** as well in foreground/background image segmentation
- For denoising purposes, the data penalty of a pixel (i,j) reflects the **agreement** between the scanned and final images

$$E_{\mathrm{D}}(i,j) = w\delta(f(i,j), d(i,j))$$

- With the delta function being 1 or 0 depending if tehre is agreement or no, and w is a weight
- The smoothness penalty expresses the level of agreement between neighboring pixels (providing some form of local coherence)

$$E_{\mathrm{P}}(i,j) = s\delta(f(i,j), f(i+1,j)) + s\delta(f(i,j), f(i,j+1)) + \ldots.$$

# Example

At iteration k, assume w=s=-1:

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

Noisy Image d

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Restored image f

$$E_{\mathrm{D}}(i,j) = w\delta(f(i,j), d(i,j))$$

| | | | | |
|---|---|---|---|---|
| -1 | -1 | 0 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 |
| 0 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 0 |

$$E_{\mathrm{P}}(i,j) = s\delta(f(i,j), f(i+1,j)) + s\delta(f(i,j), f(i,j+1)) \text{ + ....}$$

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | -3 | -4 | -4 | |
| | | | | |
| | | | | |

And so on

# Binary Markov Random Fields - Minimization

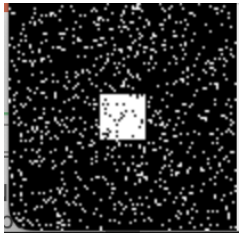
- Once we have formulated our energy function, **we need to minimize it**
- The simplest approach is to perform gradient descent, **flipping one state at a time if it produces a lower energy**.
- This approach is known as contextual classification, **iterated conditional modes (ICM),** or highest confidence first (HCF) if the pixel with the largest energy decrease is selected first.
- This method tends to find local minima
- For binary images, a much better technique is to re-formulate the energy minimization as a max-flow/min-cut graph optimization problem (**graph cuts**)
- Other approaches are based on dynamic programming

# Binary Markov Random Fields - ICM

$$\delta(a,b) = \begin{cases} 1 & if\ a == b \\ 0 & else \end{cases}$$

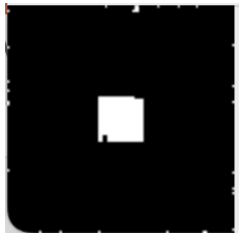
$$E_D(i,j) = -\delta(d(i,j), f(i,j))$$

$$E_P(i,j) = -\delta\big(f(i,j), f(i-1,j)\big) - \delta\big(f(i,j), f(i+1,j)\big) \\ -\delta(f(i,j), f(i,j-1)) - \delta(f(i,j), f(i,j+1))$$

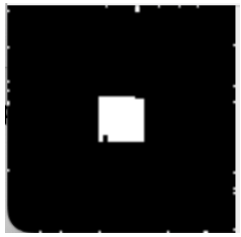


Noisy Image

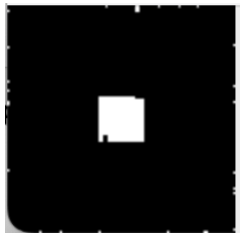
1° iteration changed 833 pixels



2° iteration Changed 27 pixels



3° iteration Changed 1 pixels



Final

**Algorithm 1** ICM - BMRF

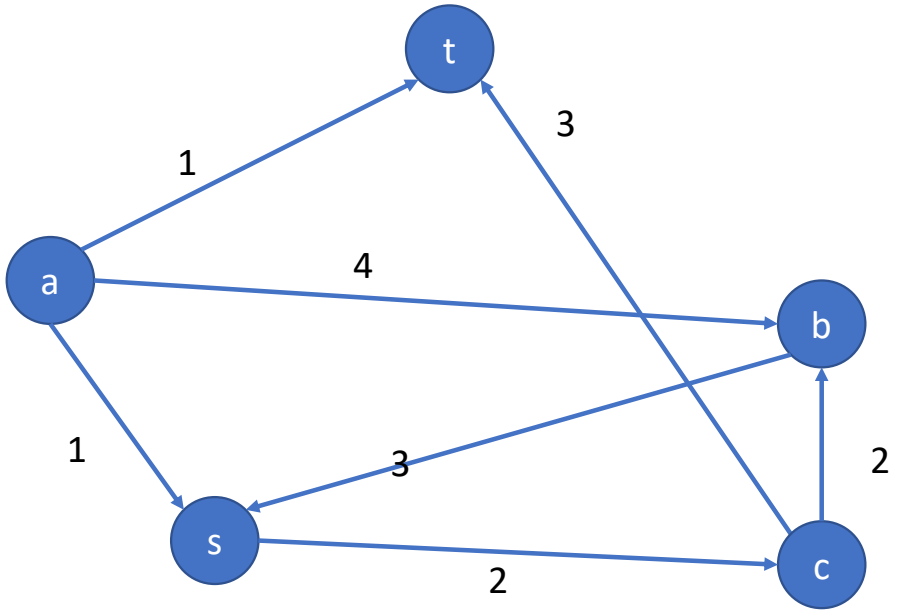
**Input:** $d$ (binary image with size $H \times W$)
**Input:** $ITER$, maximal number of iterations
**Output:** $f$ (binary image with size $H \times W$)

```
iter = 0
changed = True
while changed and iter < ITER do
    changed = False
    Compute data term E_D with current f
    Compute penalty term E_P with current f
    Compute E = E_D + E_P
    for (i = 1; i ≤ H; i = i + 1) do
        for (j = 1; j ≤ W; j = j + 1) do
            Flip current variable: t_{i,j} = 1 − f[i, j]
            Compute local data term E'_D with proposed change t_{i,j}
            Compute local penalty term E'_P with proposed change t_{i,j}
            if (E'_D + E'_P < E[i, j] ) then
                Accept change to f: f[i, j] = t_{i,j}
                changed = True
            end if
        end for
    end for
    iter+ = 1
end while
return count
```

A better implementation should take into account padding!

# Binary Markov Random Fields and Graph-Cut

- Let us consider a graph G(V, E) where V are nodes and E are edges
- Each edge can have assigned a weight c



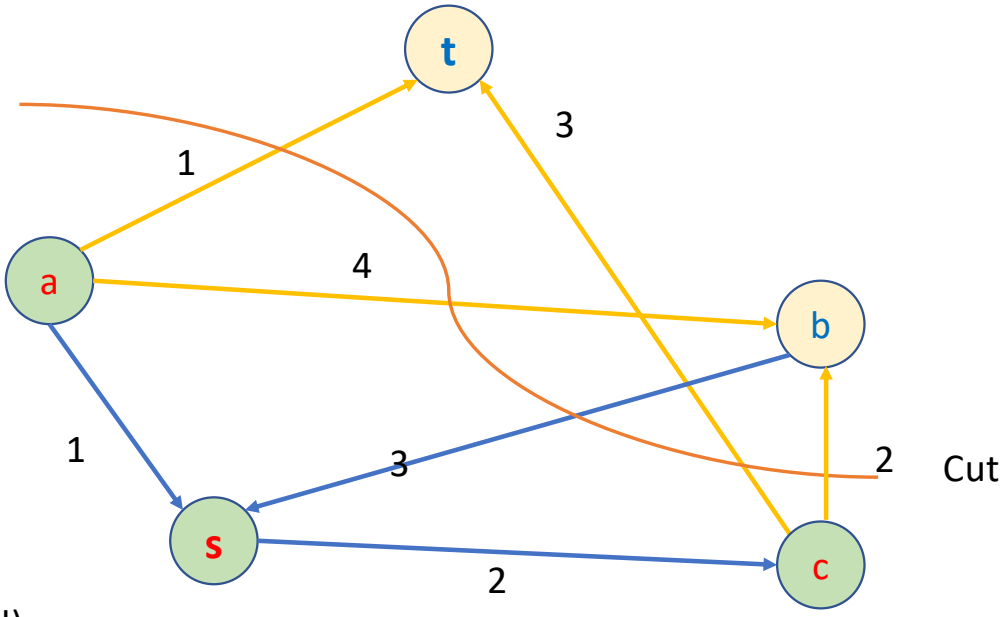
# Binary Markov Random Fields and Graph-Cut

- Let us focus on two vertices, s and t
- A cut of G w.r.t. s and t is a partition of the vertices of the graph into two groups S and T such that

$$\mathcal{V} = S \cup T, \quad S \cap T = \varnothing$$

$$s \in S, \quad t \in T$$

Total sum of the weights of the edges going **from $S$ to $T$** is called the cost of the cut



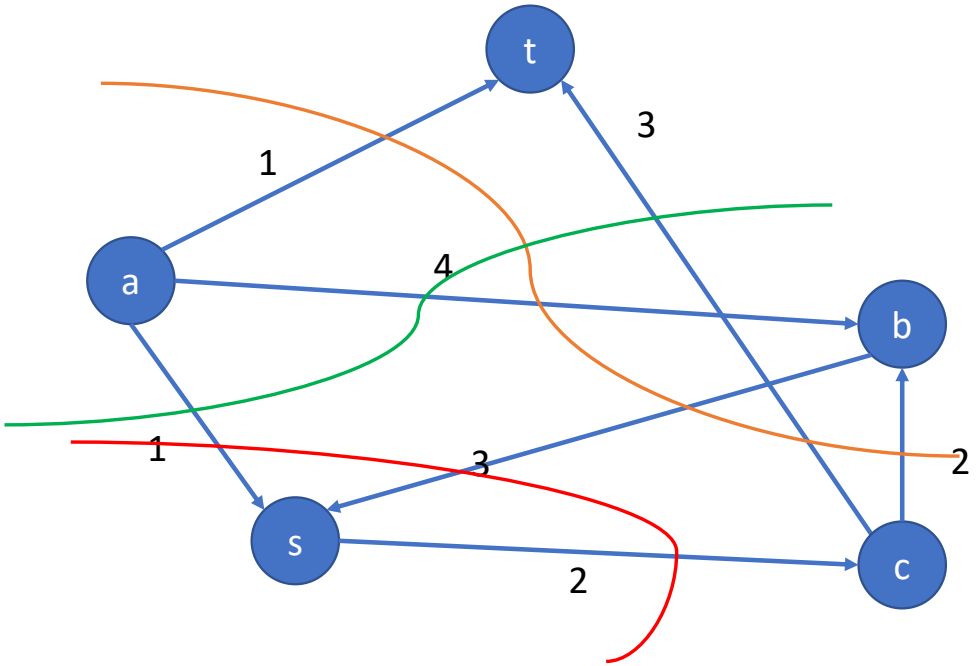

**S={s, a, c}, T ={t, b}**

Cost = 1 + 4 + 3 + 2 (the edge from b to s is not inclided)

# Binary Markov Random Fields and Graph-Cut

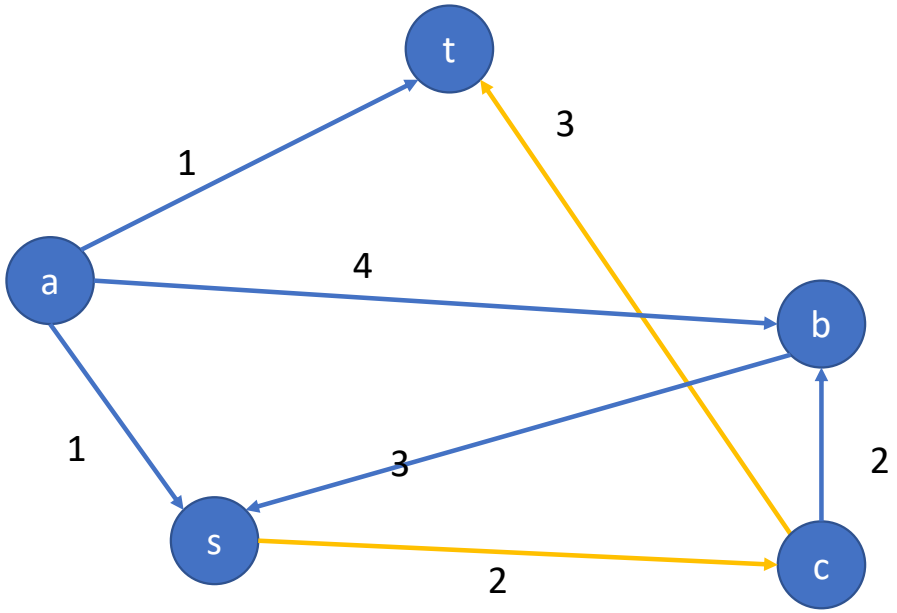
- Thus a cut is a partition of the graph into two sets
- There are many possible cuts w.r.t. s and t

- Each cut has its own cost
- A general problem in graph theory is computing the cut with the Minimum cost (min-cut)
- The algorithm solving the problem is referred to graph-cut ()



# Min Cut = Max Flow

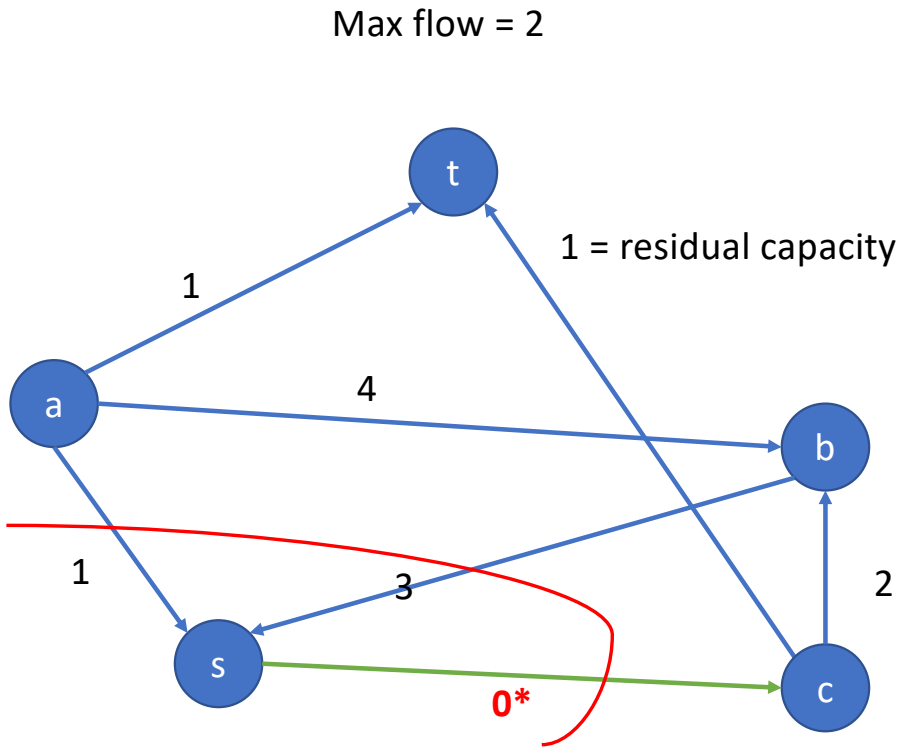
- The min cut problem is the dual of the max flow problem
- In max flow, we assume there is a source node s and a target node t
- Our gloal is maximizing the flow that can cross the edges of the graph considering that each edge has a limited capacity (the cost of the edge)
- The algorithm to compute max flow (for instance the **Ford-Fulkerson** algorithm but other methods exist) determines iteratively one path in the graph per time, to move as much flow as possible.




Along the path s, c, t only 2 units can pass
This affects the residual capacities of the edges
We will update capacity while iteratively finding new path till saturation of the edges

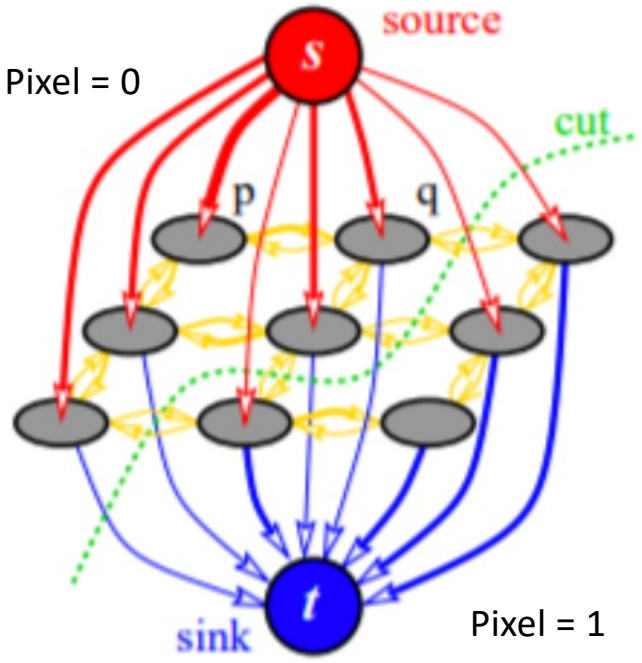
# Min Cut = Max Flow

- Once there are no more path to find in the graph because edges get saturated, we have found the max – flow
- The set of saturated edges is the corresponding minimum cut
- When the weights are all nonnegative, Max flow can be found in **polynomial time**




Min-cut with cost = 2 (cost of saturated edge )

# Min Cut for binary Images

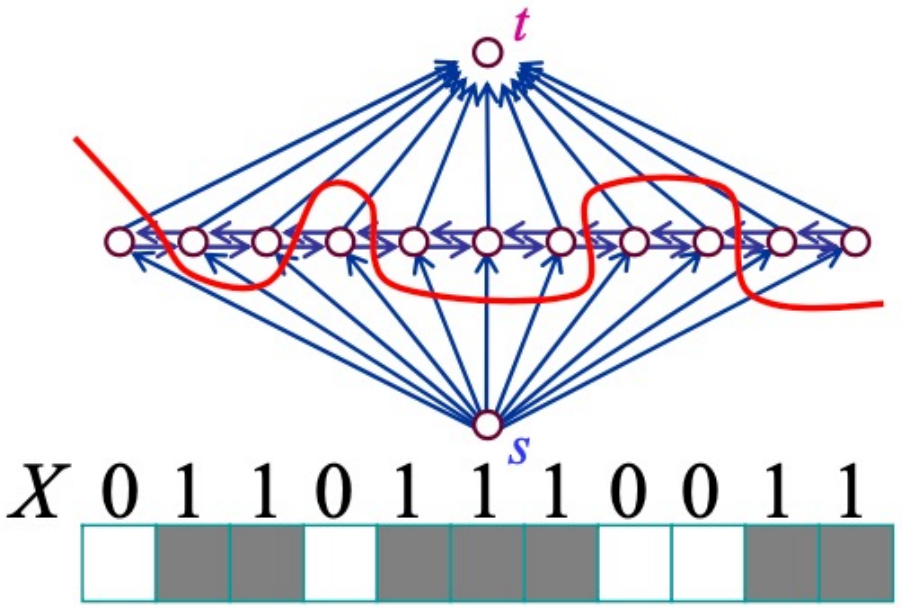



(b) A cut on $\mathcal{G}$

- All pixels are nodes of the graph
- We include two further vertices: the source node s and the sink node t
- The edges between pixels are those needed for ensuring $N_4$ adjacency
- s and t are connected with all pixels of our image
- Edges represent:
  - The cost of belonging to a class or the other (foreground/background or 0-1 in black and white noisy images) if incident on the source/sink
  - The pairwise potential between pixels (agreement in neighborhood)

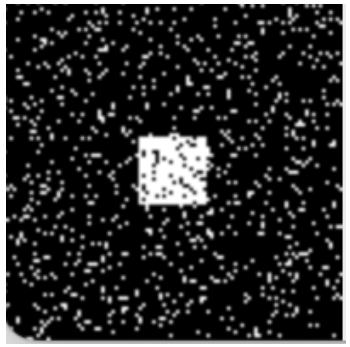
# 1-D Interpretation

- To make simpler the interpration, consider the following graph

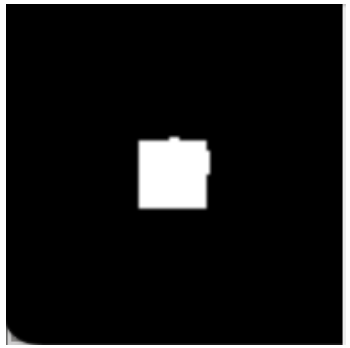



Once the min-cut has been found, pixel in the set S assumes value 0 and pixels in the set T assumes value 1

# Example – Graph Cut



Input



Output

```
import maxflow
# pip install PyMaxflow
# This is a library to compute min-
cuts (max-flow) in graphs

# Create the graph.
g = maxflow.Graph[int]()
nodeids = g.add_grid_nodes(im.shape) #each pixel is a node in
                                     the graph 4-connected by default
# Add non-terminal edges with the same capacity.
g.add_grid_edges(nodeids, 1) # K = 1
# Add the terminal edges.
g.add_grid_tedges(nodeids, im, 1-im) #This is D_i(xi) pixels are
#the capacities of the edges from the source node. The inverted
#image pixels are the capacities of the edges to the sink node.

g.maxflow() #this is the dual of min-cuts
sgm = g.get_grid_segments(nodeids)
restored_image = 1-sgm.astype(np.float32)
```

$$E(\mathbf{x}) = \sum_i D_i(x_i) + \sum_{(i,j)\in\mathcal{C}} K|x_i - x_j|.$$

http://pmneila.github.io/PyMaxflow/tutorial.html#getting-started

Questions?