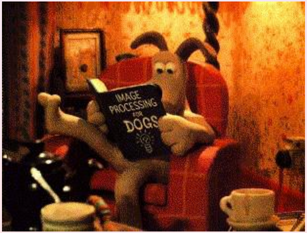
Introduction to Image Processing #4/7

Thierry Géraud

EPITA Research and Development Laboratory (LRDE)



2006

Outline

Outline

Outline

Outline

Outline

Graph Clique (1/2)

A *clique* of an undirected graph is a set of vertices where every couple of vertices are connected.

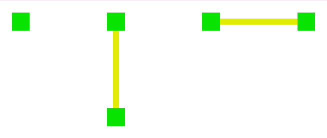
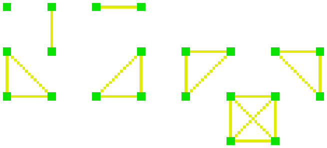
We have:

- the size $\overline{k}$ of a clique k is its number of vertices,
- in a graph finding a clique whose size is given is an NP-complete problem.

http://en.wikipedia.org/wiki/Clique_(graph_theory)

Graph Clique (2/2)

When the graph is a regular grid:

connectivity	4	8
samples		
sizes	from 1 to 3	from 1 to 4

Outline

Please Think Different! (1/3)

Have you ever think that the Sudoku was a probabilistic problem? Why?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

You stick to a *binary* position:

- there is one solution so every other configuration is impossible ;
- there is no way / reason to consider / handle an “intermediate” realization...

Please Think Different! (2/3)

Yet you are able to rank the three last lines below:

$$y_{2,*} = \boxed{\begin{array}{ccc|ccc|ccc} \mathbf{6} & ? & ? & \mathbf{1} & \mathbf{9} & \mathbf{5} & ? & ? & ? \end{array}}$$

$$x^{(1)}_{2,*} = \boxed{\begin{array}{ccc|ccc|ccc} \mathbf{6} & 6 & 6 & \mathbf{1} & \mathbf{9} & \mathbf{5} & 6 & 6 & 6 \end{array}}$$

$$x^{(2)}_{2,*} = \boxed{\begin{array}{ccc|ccc|ccc} \mathbf{6} & 6 & 2 & \mathbf{1} & \mathbf{9} & \mathbf{5} & 4 & 4 & 7 \end{array}}$$

$$x^{(3)}_{2,*} = \boxed{\begin{array}{ccc|ccc|ccc} \mathbf{6} & 3 & 2 & \mathbf{1} & \mathbf{9} & \mathbf{5} & 8 & 4 & 7 \end{array}}$$

So: smallskip

- you consider that some configurations are better (more acceptable) than other ones ;
- yet you cannot state that the last one, $x^{(3)}_{2,*}$, is such as $x^{(3)}$ is the solution of the global problem.

Please Think Different! (3/3)

- When you have filled the grid,
  - however the nature of its contents is,
  - if do not look at the whole grid,
  - then you can adopt a *probabilistic* point of view.
- When you consider the *global* grid-filling problem,
  - it actually is a collection of local problems (lines, columns, blocks),
  - which are definitely *not independant*,
  - but evaluating if they are *likely* close to the solution is very easy to express.
- Yeh, we know how to solve such a problem!

A Probabilistic Model of the Soduku Problem (1/5)

When we have a blank, we have a random variable $X_{i,j}$.

For instance, the second line is modeled as:

6	$X_{2,2}$	$X_{2,3}$	**1**	**9**	**5**	$X_{2,7}$	$X_{2,8}$	$X_{2,9}$

and the two first top blocks are:

5	**3**	$X_{1,3}$	$X_{1,4}$	**7**	$X_{1,6}$
6	$X_{2,2}$	$X_{2,3}$	**1**	**9**	**5**
$X_{3,1}$	**9**	**8**	$X_{3,4}$	$X_{3,5}$	$X_{3,6}$

A Probabilistic Model of the Soduku Problem (2/5)

One way to *reduce* the search space is to restrict the set of values taken by random variables to the only unknown values in each block.

In our example,

- the realizations of $X_{1,3}$, $X_{2,2}$, $X_{2,3}$, and $X_{3,1}$ belong to the set $\{1, 2, 4, 7\}$;
- and a partial realization for the grid is depicted below.

5	**3**	7	6	**7**	3	...
6	1	4	**1**	**9**	**5**	...
2	**9**	**8**	8	2	4	...
	...			...		

A Probabilistic Model of the Soduku Problem (3/5)

With x being a grid realization, let us define for each row i and each possible value $v \in [1, 9]$:

$$h^r_{i,v}(x) \;=\; \sum_{j=1}^{9} \delta(x_{i,j}, v)$$

where δ is the Kronecker symbol ($\delta(a, b)$ is equal to 1 if $a = b$, 0 otherwise).

Similarly, for each colum j:

$$h^c_{j,v}(x) \;=\; \sum_{i=1}^{9} \delta(x_{i,j}, v).$$

A Probabilistic Model of the Soduku Problem (4/5)

x is the expected solution when we have for every value v:
$\forall i,\ h^r_{i,v}(x) = 1$ and $\forall j,\ h^c_{j,v}(x) = 1.$

We can derive from h^r and h^c an energy:

$$U(x) = \sum_v \left(\sum_i |h^r_{i,v}(x) - 1| + \sum_j |h^c_{j,v}(x) - 1| \right)$$

which has the following properties:

$$\begin{array}{lcl} U(x) & \geq & 0 \quad \forall x, \\ U(x) & = & 0 \quad \text{iff } x \text{ is a grid solution.} \end{array}$$

A Probabilistic Model of the Soduku Problem (5/5)

At iteration t, we shall try to change the realization x^t into a realization $x^{t+1} \neq x^t$:

- for that, we randomly pick a couple of blank cells of a block, also randomly chosen;
- the candidate new realization corresponds to swapping the cell respective values; for instance:

5	**3**	7
6	1	4
2	**9**	**8**

$\longrightarrow$

5	**3**	7
6	2	4
1	**9**	**8**

Soduku Solver (1/2)

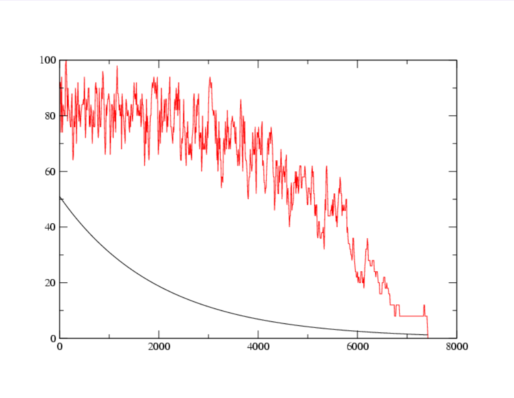
Consider that the couple of values that may swap are located at (i, j) and (i', j'); they are $v = x^t_{i,j}$ and $v' = x^t_{i',j'}$.

We then use the straightforward formula:

$$\begin{aligned}
\Delta U(x^t \to x') = \quad & (1 - \delta(i', i)) \, (& \epsilon^{h^r_{i,v}(x^t)=1} &+ \epsilon^{h^r_{i',v'}(x^t)=1} \\
& & + \ \epsilon^{h^r_{i',v}(x^t)\geq 1} &+ \epsilon^{h^r_{i,v'}(x^t)\geq 1} \ ) \\
+ \ & (1 - \delta(j, j')) \ (& \epsilon^{h^c_{j,v}(x^t)=1} &+ \epsilon^{h^c_{j',v'}(x^t)=1} \\
& & + \ \epsilon^{h^c_{j',v}(x^t)\geq 1} &+ \epsilon^{h^c_{j,v'}(x^t)\geq 1} \ )
\end{aligned}$$

where $\epsilon^a = 1$ if a is true, -1 otherwise.

Soduku Solver (2/2)



the temperature T (black) is decreasing through iterations (x-axis) while the energy (red) converges to 0

Outline

The Questions were... (1/2)

- Can we have a pixel of red pepper in the middle of green pepper pixels?
- Can we have a red pixel in the middle of a green pepper? pixels?
- What is the color of a pixel of a green pepper?

The Questions were... (2/2)

Translation:

- “red pepper” is a possible value of a pixel in the result image X; this value is a label identifying an object;
- “color red” is a possible value of a pixel in the input image Y.

Guess what?

Answers were not expected to be simple—binary—but given with a *probabilistic* point of view.

The Answers are...

Can we have a pixel of red pepper in the middle of green pepper pixels?

yes but $P(\, X_i = \text{“red pepper”} \,|\, X_{\nu_i} = \{\text{“green pepper”}\}\,)$ is low

Can we have a red pixel in the middle of a green pepper? pixels?

yes but $P(\, Y_i = \mathit{red} \,|\, X_{\nu_i} = \{\text{“green pepper”}\}\,)$ is low

What is the color of a pixel of a green pepper?

it is the probability function / distribution
$P(\, Y_i = y_i \,|\, X_i = \{\text{“green pepper”}\}\,)$

Outline

Stochastic Process

A *discrete stochastic process* is a random function the domain of which is discrete.

- The domain can be for instance the time space (index t).
- The process can be seen as a collection of random variables $\{X_t\}$.
- A particular process is defined by expressing the joint probabilities of the various random variables.

Markov Property

A stochastic process has the *Markov property* if the conditional probability of future states of the process, given the present state, depends only of the current state.

Put in formula:

$$\forall h > 0,$$
$$P(\, X_{t+h} = x_{t+h} \,|\, \{X_s = x_s,\, s \leq t\} \,) \;=\; P(\, X_{t+h} = x_{t+h} \,|\, X_t = x_t \,).$$

Markov Chain

A *Markov chain* is a discrete-time stochastic process with the Markov property.

Such a chain can be characterized by:

$$\begin{aligned} & P(\,X_0 = x_0\,) \\ \textit{and} \quad & P(\,X_{t+1} = x_{t+1} \,|\, X_t = x_t\,). \end{aligned}$$

That conditional probability is called the *transition probability* of the process.

Monte Carlo

Monte Carlo methods are a class of computational algorithms for simulating a physical or mathematical system.

The key ideas are:

- *first* to consider that a deterministic problem can be turned into a probabilistic analog,
- *then* to recourse to statistical sampling to solve the problem.

The classical use of Monte Carlo is solving numerical problems such as integral calculi, simulations, optimizations.

Markov Chain Monte Carlo (MCMC)

Monte Carlo Markov Chain (MCMC) methods are a class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its stationary distribution.

A few remarks follow.

- Random walk methods, where the walk follows a Markov chain, are a kind of MCMC methods.
- Solving some problems often require that an *ensemble* of walkers (so more than one) are computed which move around randomly.
- The Metropolis algorithm and the Gibbs sampling are MCMC random walk methods!

Gibbs State

A *Gibbs state* is an equilibrium probability distribution which remains invariant under future evolution of the system.

For example, a stationary or steady-state distribution of a Markov chain, such as achieved by running a MCMC iteration for a sufficiently long time.

Bayesian Network (1/3)

A *Bayesian Network* is a directed acyclic graph where vertices and edges respectively represent variables and the dependence relations between variables.

A variable can be:

- not only a random variable,
- but also an observation,
- or an hypothesis.

Bayesian Network (2/3)

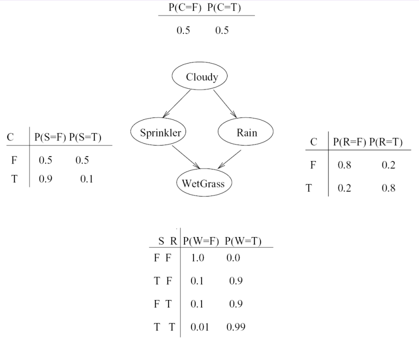
A Bayesian network is a form of *probabilistic graphical model*.

We have:

- parenthood to represent conditional probabilities $P(\, X_i \,|\, parents(X_i)\,)$,
- a graphic to understand and work on systems.

Bayesian Network (3/3)

A very simple one:



See also:

`http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`

Markov Network (1/6)

Let us consider:

- an undirected graph G and its cliques,
- the notation $x_{(k)}$ to designate the realization of the set of random variables associated with k

  that is shorter than $\{x' | x' \in k\}$
- a set of functions ϕ_k
  - with k a kind of clique of G
  - and with $\phi_k(\, x_{(k)}\,) \in \mathbb{R}^+$

Markov Network (2/6)

A Markov Network is such as:

$$P(X = x) \; = \; \frac{\prod_k \phi_k(\, x_{(k)}\,)}{Z}$$

where Z is a normalizing constant.

Now:

- let us assume that we cannot have $\phi_k(\, x_{(k)}\,) = 0$

  put differently: "nothing is impossible"
- so let us rewrite $\phi_k(\, x_{(k)}\,) \; = \; e^{-U_k(\, x_{(k)}\,)}$

We have:

$$P(X = x) \; = \; \frac{e^{-\sum_k U_k(\, x_{(k)}\,)}}{Z}$$

Markov Network (3/6)

About notations and their meaning.

- X a markov network ; actually it is :
  - the multivariate random variable associated with what we are looking for
  - a probabilistic view of our unknown output image
  - the mathematical function that describes or governs our search space
  - and just remember that we can walk within that space to find a solution
- X_i the random variable associated with the i th point/vertex of X

Markov Network (4/6)

About notations and their meaning (cont'd).

- let ν_i denotes the neighborhood of this vertex
- then let us introduce $X^i = X / X_i$
  - where '/' means "except" or "minus"
  - so it is the conterpart of X_i
  - the random network without the i th variable
- and X_{ν_i}
  - $X_{\nu_i} = \left\{ X_j, \text{ the } j \text{ th point is a neighbor of the } i \text{ th point} \right\}$
  - so it means what is around X_i
  - the random network around the i th variable

Markov Network (5/6)

So we have a Gibbs random field for we have:

$$P(X = x) = e^{-U(x)}/Z \quad \text{here with} \quad U(x) = \sum_k U_k(x_{(k)}).$$

and a *very convenient* local (Markovian) property:

$$P(X_i = x_i \,|\, X^i = x^i) = P(X_i = x_i \,|\, X_{\nu_i} = x_{\nu_i}).$$

So:

- A markov network with *no null probability* is a Gibbs field.
- You can *either* handle probabilities *or* energies.
- When we focus on point i, we *only* have to consider this point and its neighborhood.

Markov Network (6/6)

That is really **great** because:

- in actual problems, assuming that nothing is impossible allows to find solutions!
  - remind the Sudoku solving problem...
- we can express—or model— *global* problems while taking only *local* considerations
  - some hard problems can then be solved
- thinking in terms of energies is equivalent to thinking in terms of probabilities
  - and it is often easier!

URLs (1/2)

- Stochastic process

  `http://en.wikipedia.org/wiki/Stochastic_process`
- Monte Carlo

  `http://en.wikipedia.org/wiki/Monte_Carlo_method`
- Bayesian inference

  `http://en.wikipedia.org/wiki/Bayesian_inference`
- Bayesian network

  `http://en.wikipedia.org/wiki/Bayesian_network`
- Gibbs mesure

  `http://en.wikipedia.org/wiki/Gibbs_mesure`

URLs (2/2)

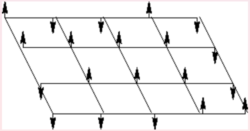
- Markov property
  http://en.wikipedia.org/wiki/Markov_property
- Markov process
  http://en.wikipedia.org/wiki/Markov_process
- Markov chain
  http://en.wikipedia.org/wiki/Markov_chain
- MCMC
  http://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo
- Markov network
  http://en.wikipedia.org/wiki/Markov_network

Outline

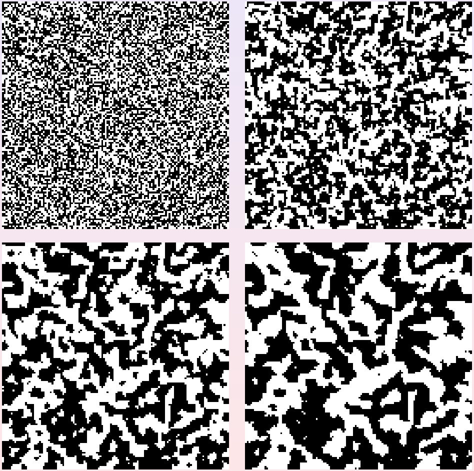
Ising Model (1/2)

The Ising model:

- belongs to statistical mechaniscs;
- is such that every vertex of a graph represents a spin;
- states that each pair of neighbors interacts
  - where parallel spins are favored (energy $-J$),
  - and antiparallel spins are discouraged (energy $+J$);
- is such that the probability of a configuration x of the graph at temperature T follows $e^{-U(x)/T}$.



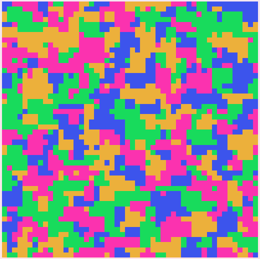
Ising Model (2/2)



Potts Model

The Potts model:

- is a generalization of the Ising model where a realization at site i is no more a spin—binary value—but an n-ary one;
- uses $\sum_{k=(i,j)} J_k \delta(s_i, s_j)$.



$L^1 + \mathrm{TV}$ (1/2)

The $L^1 + \mathrm{TV}$ model is used in function regularization—denoising:

- x should stay close to input data y and the distance between x et y is measured with L^1;
- x should be regularized so the total variation of x should be low; this variation is evaluated through the gradient of x.

For 1D continuous functions:

$$U(x) = \int |x(t) - y(t)|\, dt + \beta \int |\bigtriangledown x(t)|\, dt$$

L^1 + TV (2/2)

An equivalent formula for 2D discrete functions is:

$$U(x) = \sum_i |x_i - y_i| + \beta \sum_i \sum_{x_j \in \nu_i} |x_i - x_j|.$$

β allows for tuning the respective effects of L^1 and TV.

URLs

- Ising model

  `http://en.wikipedia.org/wiki/Ising_model`
- Potts model

  `http://en.wikipedia.org/wiki/Potts_model`

Outline

Probability Distribution

When V is defined over $\mathbb{R}$:

- V can be defined by a distribution (a function) f_V;
- this distribution assigns to every interval of $\mathbb{R}$ a probability
- f_V is a probability distribution—probability density.

We have:

$$P(a \leq V \leq b) \;=\; \int_a^b f_V(v)\,dv\,.$$

Expected Value (1/3)

With V random variable, the expected value of V is:

$$E(V) = \int V\,dP.$$

We have:

$$E(V) = \int_{-\infty}^{\infty} v\, f_V(v)\, dv.$$

Properties:

- it is linear;
- $E(E(V)) = E(V^2) - E(V)^2$
- $E(\,V \mid W = w\,) = \sum_v P(V = v | W = w)\, v$

Expected Value (2/3)

If V is a **discrete** random variable which takes some values v:

$$E(V) = \sum_v P(V = v)\, v.$$

Expected Value (3/3)

The variance of V is:

$$var(V) = E(E(V)) = E(V^2) - E(V)^2.$$

and the standard deviation is:

$$\sigma_V = \sqrt{var(V)}.$$

Covariance (1/3)

The covariance—or cross-covariance—of a couple of **real-valued** random variables V and W is:

$$\begin{aligned} cov(V,W) &= E(\,(V-E(V))\,(W-E(W))\,) \\ &= E(V\,W) - E(V)\,E(W) \\ &= cov(W,V). \end{aligned}$$

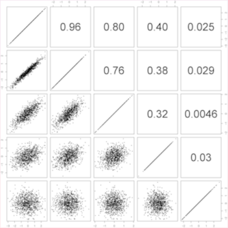
If V and W are independent, $E(V\,W) = E(V)\,E(W)$ so $cov(V,W) = 0$.

We have $cov(V,V) = E(V^2) - E(V)^2 = var(V) = \sigma_V^2$.

Correlation

The correlation between two random variables V and W is:

$$\rho_{V,W} = \frac{cov(V,W)}{\sigma_V\,\sigma_W}.$$



Covariance (2/2)

When V and W are multivariate random variables—**vector-valued**, the covariance is the matrix:

$$cov(V,W) \;=\; E(\,(V - E(V))\,(W - E(W))^T\,),$$

and $cov(W,V) = cov(V,W)^T$.

Covariance (3/3)

We (simply) say that $cov(V, V)$ is the covariance matrix of V.

With $V = (V_1, .., V_N)^T$ and $W = (W_1, .., W_N)^T$, we have

$$cov(V, V)_{i,j} \; = \; cov(V_i, V_j),$$

and the diagonal of the cross-covariance matrix contains the random variables variances:

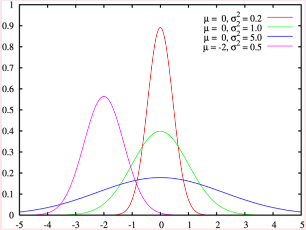
$$cov(V, V)_{i,i} \; = \; var(V_i).$$

Normal Distribution

A random variable V follows a normal distribution if:

$$P(V = v) \; = \; \mathcal{N}(\mu, \sigma^2) \; = \; \frac{1}{\sigma\sqrt{2\pi}} e^{-\dfrac{(v-\mu)^2}{2\sigma^2}}$$

where μ and σ respectively are the mean and standard deviation.



Multivariate Normal Distribution

A random vector $V = (V_1, .., V_N)$ follows a multivariate normal distribution if every linear combination of V_j follows a normal distribution.

We have:

$$f_V(v) = \frac{1}{(2\pi)^{N/2} \, |\Sigma|^{1/2}} \, e^{-\frac{1}{2} (v-\mu)^t \, \Sigma^{-1} \, (v-\mu)}$$

where μ is a vector (size N), Σ a positive definite covariance matrix (size $N \times N$), $|\Sigma|$ the determinant of Σ.

URLs (1/2)

- Probability density function

  http://en.wikipedia.org/wiki/Probability_density_function
- Gaussian function

  http://en.wikipedia.org/wiki/Gaussian_function
- Normal distribution

  http://en.wikipedia.org/wiki/Normal_distribution
- Multivariate normal distribution

  http:
  //en.wikipedia.org/wiki/Multivariate_normal_distribution

URLs (2/2)

- Expected value

  http://en.wikipedia.org/wiki/Expected_value
- Covariance

  http://en.wikipedia.org/wiki/Covariance
- Correlation

  http://en.wikipedia.org/wiki/Correlation
- Covariance matrix

  http://en.wikipedia.org/wiki/Covariance_matrix

Outline

Something Rather Different (1/2)

Now for something quite different:

- assume that we do not know about the probability distribution of a random phenomenon;
- but we have **samples**—or observations or realizations—of that phenomenon;
- we can **assume** that the phenomenon follows a given parametric distribution...
- and the **estimate** the parameters.

Something Rather Different (2/2)

For instance, state that the distribution is normal ($\mathcal{N}(\mu, \sigma)$) so estimate μ and σ.

Please do *not* misunderstand:

- the *actual* distribution of the phenomenon may **not** be the chosen parametric distribution!
- we just have chosen a *model* to be able to work with!!!

Estimating a Normal Distribution

Consider n samples $v^{(j)} \in \mathbb{R}$ of a random variable V.

When the parametric model is the normal distribution, we can compute:

$$\begin{aligned} \mu &= E(V) &&= \frac{1}{n} \sum_{j=1}^{n} v^{(j)} \\ \sigma^2 &= E\left((V-\mu)^2\right) &&= \left(\frac{1}{n} \sum_{j=1}^{n} \left(v^{(j)}\right)^2 \right) - \mu^2. \end{aligned}$$

Thus we assume that:

- $P(V = v) = \mathcal{N}(\mu, \sigma)(v)$,
- the samples $v^{(j)}$ is a set of observations which is *representative* enough of V.

Estimating a Multivariate Normal Distribution

Assuming a multivariate normal distribution, with samples $v^{(j)}$ being vectors, proceed likewise:

$$\begin{aligned} \mu &= \frac{1}{n} \sum_{j=1}^{n} v^{(j)} \\ \Sigma &= \left(\frac{1}{n-1} \sum_{j=1}^{n} v^{(j)} \, {v^{(j)}}^{T} \right) - \mu^2. \end{aligned}$$

with μ vector and Σ the (unbiased) sample covariance matrix.

`http://en.wikipedia.org/wiki/Estimation_of_covariance_matrices`

Mahalanobis Distance (1/2)

The Mahalanobis distance is the distance between a vector and a group of vectors with mean μ and covariance matrix Σ:

$$d(\, v,\, \{v^{(j)}\}\,) \;=\; \sqrt{(v-\mu)^T\,\Sigma^{-1}\,(v-\mu)}.$$

With two samples v and v' of the same distribution with covariance matrix Σ, this distance is a dissimilarity measure:

$$d(\, v,\, v'\,) \;=\; \sqrt{(v-v')^T\,\Sigma^{-1}\,(v-v')}.$$

Mahalanobis Distance (1/2)

If the covariance matrix is diagonal, we have a *normalized* Euclidean distance:

$$d(v, v') = \sqrt{\sum_{i=1}^{N} \frac{(v_i - v'_i)^2}{\sigma_i^2}}.$$

http://en.wikipedia.org/wiki/Mahalanobis_distance

Discrete Unparameterized Distribution (1/2)

Now imagine that you do not want a parameterized model for a distribution of n samples but a discrete distribution.

- A window $\mathcal{W}_s$ centered on the discrete realization v_s contains a given number of samples: n_s; we have:
  - $n_s \;=\; \sum_j \delta_{v(j) \in \mathcal{W}_s}$.
  - $n \;=\; \sum_s n_s$.
- An approximate value of the probability density function at this discrete realization is: P_s.
  - If $\overline{\mathcal{W}}$ is the size of every window $\mathcal{W}_s$,
  - We have $P_s \;=\; P(V = v_s) \;=\; \frac{n_s}{n \times \overline{\mathcal{W}}}$.

Discrete Unparameterized Distribution (2/2)

Yet

- the window size should be *large enough* to contain many samples so that couting them is representative of the distribution;
- the window size should be *small enough* so that we really get a density value.

So:

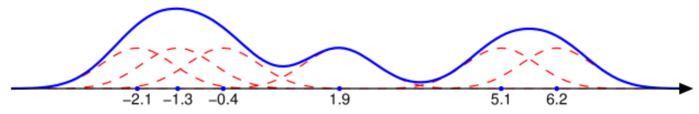
- these two constraints are opposite!
- that method only works when the population is very dense, that is, when we have a lot of (a huge number of) samples...

Parzen Window Method

The idea of the Parzen window method is simple:

the probability density function is estimated thru an extrapolation from a normal elementary contribution of every sample (vector of features).

We have $P(V = v) \; = \; \frac{1}{n} \sum_j \mathcal{N}(\, v^{(j)}, \, \sigma_{\text{parzen}} \,)$.



Outline

About Statistics and Data (1/2)

- *Statistics:* science pertaining to collection, analysis, interpretation, and presentation of data.

- *Data analysis:* act of transforming data to extract useful information and facilitate conclusions.

- *Data mining:* automatic search for patterns in large volumes of data.

About Statistics and Data (2/2)

Just realize that images are data and patterns are scene objects!

http://en.wikipedia.org/wiki/Statistics

http://en.wikipedia.org/wiki/Data_analysis

http://en.wikipedia.org/wiki/Data_mining

Data as a Population (1/2)

We now consider that:

- a pixel (or higher-level primitive) is an individual in a population—an entry in a set of data;
- the population of individuals is the input image,
- we have some data / information about every observed individual

Data as a Population (2/2)

Cont'd:

- yet we represent each individual by a vector of features,
  - often features are not the raw observation information,
  - all individuals are now in a single (*n*-dimensional feature) space called the feature space,
  - the transform "observation $\rightarrow$ feature vector" aims at normalizing data,
  - so in the feature space, individuals can be compared and the population can be processed...

Principal Component Analysis (1/2)

The principal component analysis:

- is a technique to simplify a data set;
- is a linear transform that transforms data to a new coordinate system;
- the greatest variance is on the 1st axis, the 2nd greatest variance on the 2nd axis, and so on;
- is also known as Karhunen-Loève transform.

http://en.wikipedia.org/wiki/Principal_component_analysis

Principal Component Analysis (2/2)

The how-to:

- compute the empirical mean μ;
- compute the covariance matrix Σ;
- compute the eigenvectors and eigenvalues λ_p;
- rearrange the system with *decreasing* eigenvalue so $\lambda_p \geq \lambda_{p+1}$;
- compute the cumulative energy $E_p = \sum_q \lambda_q$;
- select the principal eigenvectors, with $p \leq p_{max}$, so that $E_{max} \geq \tau \sum_p E_p$;
- express data in this basis.

Other Kinds of Analysis

- *Factor analysis:* aims at studying variability among observed random variables in term of fewer unobserved random variables called *factors*; the observed variables are modeled as linear combination of factors + some error terms.

- *Linear discriminant analysis* aims at finding the linear combination of features which best separate two or more classes.

http://en.wikipedia.org/wiki/Factor_analysis

http://en.wikipedia.org/wiki/Linear_discriminant_analysis

Objectives (1/2)

The objective can be multiple.

- If the population is composed of one single group of individuals:
  - we want to characterize this group,
  - we say that we are *learning*—how this group is / looks like.

Exercise:
Which kind of data analysis is relevant in that case?

Objectives (2/2)

Cont'd:

- If the population is composed of different groups of individuals.
  - in an image, groups usually come from the presence of different objects,
  - objects naturally form *clusters / classes*,
  - we aim at identifying these clusters / classes,
  - and often the big deals consists in achieving to *separate* clusters / classes,

Lecture Focus

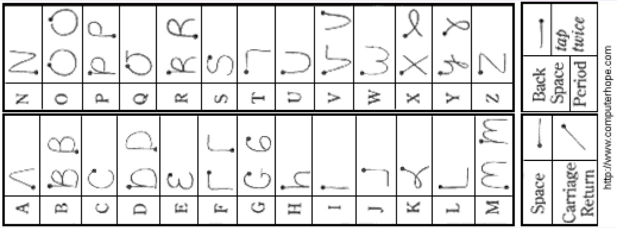
In the following we will focus on data clustering and statistical classification.

http://en.wikipedia.org/wiki/Data_clustering

http://en.wikipedia.org/wiki/Statistical_classification

Exercise

Consider the Palm Pilot alphabet:



- Express character recognition in terms of a data analysis problem.
- Imagine different sets of some relevant features.

Outline

Statistics is About Counting (1/3)

Given a gray-level image I, we can count the number of gray-level occurrances of its pixels:

$$h(g) \;=\; \sum_{p,\, I(p)=g} 1,$$

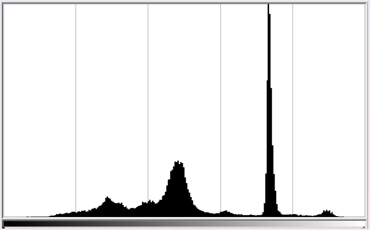
where:

- g is a gray-level value, e.g., $\in [0, 255]$
- and p an image point.

h is the image *histogram*.

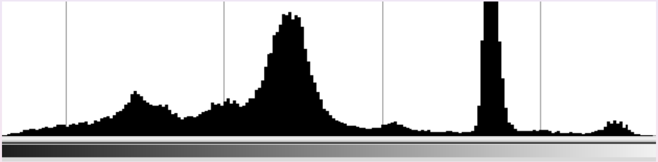
Statistics is About Counting (2/3)

For instance, with I (left), we get the histogram h (right):



The x-axis shows increasing gray-levels from black (left) to white (right); statistics is here *about gray-levels*.

Statistics is About Counting (2/3)



Although the image is not well-contrasted, we clearly see:

- 6 histogram peaks at least
- which translate the existance of several clusters / classes.

A Few Remarks (1/3)

In the previous example:

- we have $N = 1$
  - a pixel has one single feature, this is not much to analyze data!
  - we can observe that clusters / classes are not well-separated,
- we have a digital image,
  - so the image is quantized (usually on 8 bit) and features are *discrete* values (from 0 to 255),
  - often feature components are not discrete but $\in \mathbb{R}$.

A Few Remarks (2/3)

In the previous example (cont'd):

- we have small objects in the image,
  - for instance, the white parts of the windows and of the roof represent less than 200 pixels in the image,
  - often we have to perform statistics on sub-populations that have very few samples (individuals).

A Few Remarks (3/3)

Other examples.

- When we have color image,
  - for instance, encoded on red-green-blue (RGB for short) with 8 bit per component,
  - then we have a straightforward 3-dimensional feature space.
- When we have texture information,
  - for instance, we have computed some characteristics of the local texture around each pixel,
  - we can take these values into account in the pixel feature vector.

Outline

What a Class Is (1/2)

First add a *distance* to the feature space; then:

- a class is a set of individuals which are very similar
  $\longrightarrow$ the distance between every couple of individuals of the same class is low,
- two distinct classes are dissimilar
  $\longrightarrow$ the distance between every couple of individuals taken in two distinct classes is high,
- a special “class”, the *rejection* class, contains individuals that cannot be classified...

What a Class Is (2/2)

About the rejection class:

- two criteria can cause an individual to fall in the rejection class;
- the ambiguity criterion rejection,
  $\longrightarrow$ the two tiniest distances between an individual and classes are too similar,
- the distance criterion rejection,
  $\longrightarrow$ the tiniest distance between an individual and classes is too high.

In the following, we will not discuss the use of such a class.

Automatic Classification

Automatic classification:

- Classification is automatic when there is *no explicit learning* step relying on a *human expert*.
- An automatic classifier thus provides us with classes from the raw input data—the population.
- Though there is somehow an *implicit* learning process within the classifier...

http://en.wikipedia.org/wiki/Data_clustering

http://en.wikipedia.org/wiki/Machine_learning

http://en.wikipedia.org/wiki/Unsupervised_learning

Supervised Classification

Supervised classification:

- A classifier can be *supervised* if there is an explicit learning step relying on a human expert.
- On one hand a first population, classified by a human expert, is used to learn some characteristics about classes.
- On the other hand, a population to process is classified w.r.t. to what has been learned.

FIXME: Reminder

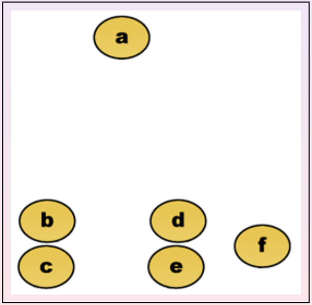
Say something about:

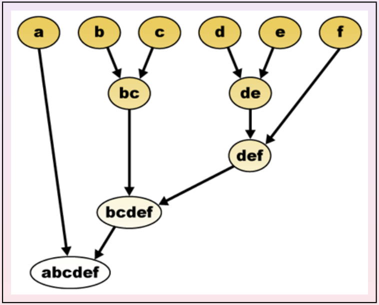
- hierarchical clustering v. partional clustering;
- data clustering v. classification;
- *k*-nearest neighbor.

http://en.wikipedia.org/wiki/Nearest_neighbor_(pattern_recognition)

Agglomerative Hierarchical Clustering

On the left vectors in a 2D feature space and on the right a hierarchical clustering:





To build the hierarchy (the classification), some particular distance in feature space is required.

k-Means Algorithm (1/5)

Given:

- the number k of expected classes,
- the individuals represented by the set $\{ v^{(j)} \}_{j=1..n}$ of vectors in the feature space,
- the classes $\omega_l \in \Omega$ with $l = 1..k$

the k-means algorithm is an iterative process to group vectors into clusters while minimizing:

$$U = \sum_{l=1}^{k} \sum_{j,\, v^{(j)} \in \omega_l} |v^{(j)} - \mu_l|^2$$

with μ_l the mean vector of all $v^{(j)} \in \omega_l$, that is, the center of the l^{th} class.

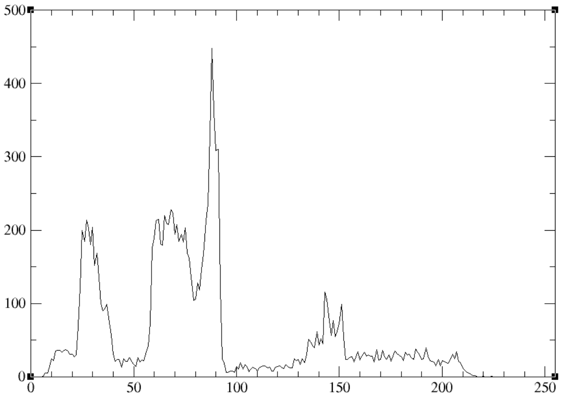
k-Means Algorithm (2/5)

More precisely the algorithm:

- *Initialization:* chose class centers μ_l (with $l = 1..k$) in the feature space;
- *Repeat until convergence:*
  - compute the classes, that is, assign a class to every $v^{(j)}$:
    $v^{(j)} \in \omega_l$ if $\forall l',\ d(v^{(j)}, \mu_l) \leq d(v^{(j)}, \mu_{l'})$
  - compute the number of vectors in each class ω_l:
    $n_l = \sum_j \delta_{v^{(j)} \in \omega_l}$
  - update the center of each class ω_l:
    $\mu_l = \frac{1}{n_l} \sum_j v^{(j)}$

k-Means Algorithm (3/5)

Input data (left) and its gray-level histogram (right):



k-Means Algorithm (4/5)

Results with k from 3 (left) to 6 (right):

The gray-level values in the classified images correspond to the respective centers of classes.

http://en.wikipedia.org/wiki/K-means_algorithm

k-Means Algorithm (5/5)

The classification process is the assignment:

$$v^{(j)} \;\longrightarrow\; x^{(j)} \in \Omega.$$

and a population after classification is $x = \{x^{(j)}\}$.

We thus have:

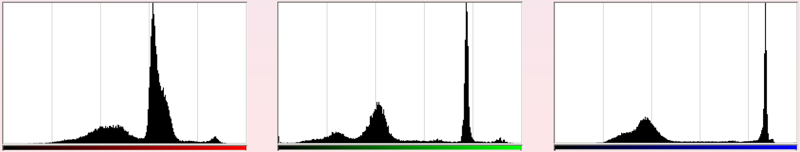
- y the raw population (set of observations, measures);
- $\{\, v^{(j)} \,\}$ the feature vectors representing y in the feature space;
- and x an output classification.

Outline

Example (1/3)

Now consider this color image:

We can compute the histogram of its color components (red, left; green, middle; blue, right):

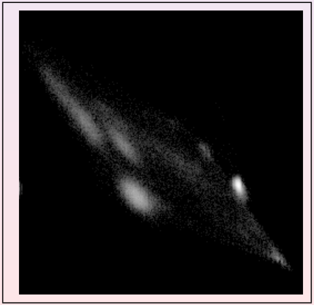


Example (2/3)

Actually we can represent data in the 3D RGB space:

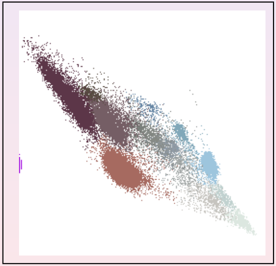
FIXME: insert a picture here!

or in 2D if we discard the blue component:



Example (3/3)

From left to right below: the original image, the classification in RG space, and the classified image.



Another Look at the k-Means Algorithm (1/4)

Actually the k-means algorithm, while minimizing

$$U(v) = \sum_{l=1}^{k} \sum_{j,\, v^{(j)} \in \omega_l} |v^{(j)} - \mu_l|^2,$$

assumes that:

- all classes follow normal distributions, respectively centered in y, but with the *same* covariance!
- ...

Another Look at the k-Means Algorithm (2/4)

so it assumes that:

- the gray-level distributions are:

$$P(\, v \,|\, \omega_l \,) \;=\; \frac{\mathcal{N}(\, \mu_l, \, \sigma \,)(\, v \,)}{Z}$$

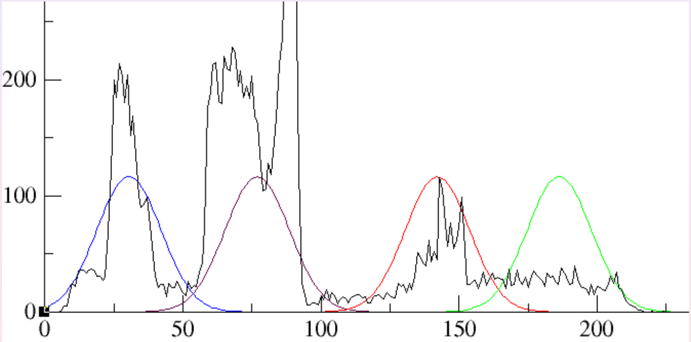
  where Z is a normalization constant;

- and the class assignment decision is:

$$v \rightarrow \omega_l \text{ where } l \;=\; arg \max_{l'} P(\, v \,|\, \omega_{l'} \,)$$

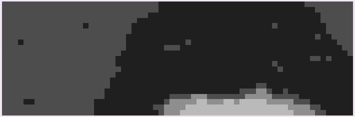
Another Look at the k-Means Algorithm (3/4)

Precisely, the Gaussian functions $P(\, v \mid \omega_l \,)$ are the following:



The limits between classes in the gray-level space correspond to the values where the functions cross.

Another Look at the k-Means Algorithm (4/4)



Finding classes in the feature space does *not* take into account the (spatial) context of pixels in the image. Otherwise the isolated pixels would be removed.

Partial Conclusion

- We may want to turn classification into a probabilistic problem.
- We rather would like to maximize $P(\,\omega_{l'} \mid v\,)$;
so to introduce *prior* probabilities in the model.
- We prefer:
  - to have the best distribution estimates as possible,
  - automatic methods over supervised ones.
- We expect our solution to take into account contextual information.
- We really like global solutions (not local ones).

Outline

We Said... (1/2)

- We are looking for a realization of X, an image of object labels and we have the input image y, realization of Y.
- We want to maximize $P(X = x|Y = y)$, that is, get the most probable solution given the input.
- $x_{sol} = \dfrac{P(Y = y|X = x)\,P(X = x)}{P(Y = y)}$

We Said... (2/2)

An histogram h counts people in a feature space:

- $h(v)$ is the number of individuals whose feature (or feature vector) is v;
- this value can be interpreted in terms of the probability $P(v) = h(v)/n$.
- and in feature space we have different classes.

!!!

Thus the classification process can be expressed in terms of probability.

Input (1/6)

Let us first consider $P(Y = y)$, that is, the input image *whatever* the objects within the scene are. More precisely, focus on $P(Y_i = y_i)$, that is, on the i^{th} pixel; then:

let us assume that input pixels are *independent*.

This assumption is very critizable: the captor can mix observations from one pixel to a neighbor one...

However we thus state that:

$$P(Y = y) \; = \; \Pi_i \, P(Y_i = y_i).$$

Input (2/6)

Cont'd:

$$
\begin{aligned}
P(Y_i = y_i) &= P(\, Y_i = y_i \, \cap \, (\cup_l X_i = \omega_l)\,) \\
&= P(\, \cup_l (Y_i = y_i \, \cap \, X_i = \omega_l)\,) \\
&= \textstyle\sum_l P(\, Y_i = y_i \, \cap \, X_i = \omega_l \,) \\
&= \textstyle\sum_l P(\, Y_i = y_i \,|\, X_i = \omega_l \,)\, P(X_i = \omega_l)
\end{aligned}
$$

Imagine that you have a learning process for each class:

- if your problem is stationary, these probabilities are functions that do *not* depend upon the location of the i^{th} point in the image;
- the prior probability and the likelihood can be estimated.

Input (3/6)

So

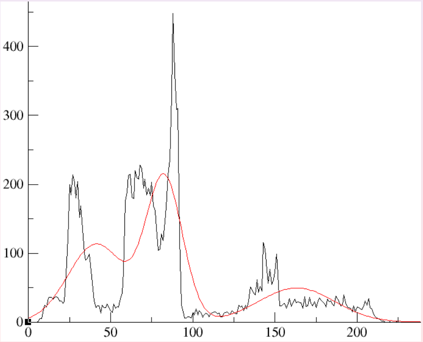
- the l^{th} class has a given probability to appear:
  - $P(X_i = \omega_l) \;=\; P_l$
  - either you do not know so you say that each class has the same probability to appear: $P_l = \frac{1}{k}$,
  - or you have n_l samples from this class in your population thus: $P_l = \frac{n_l}{n}$.
- for each class, the likelihood is defined as a probability density function of the input data:
  - $P(\, Y_i = y_i \,|\, X_i = \omega_l \,) \;=\; f_l(y_i)$
  - with $f_l(v)$ learned from the samples of the l^{th} class,
  - for instance, $f_l(v) = \mathcal{N}(\mu_l, \sigma_l)(v)$

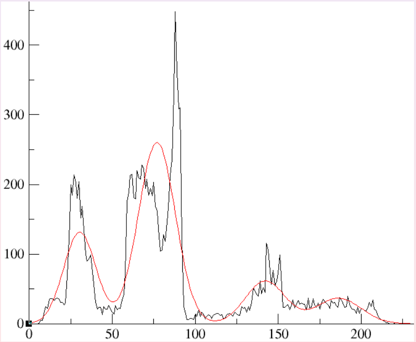
Input (4/6)

- If we take the results of the k-means algorithm, we have a rough classification thus classes and samples for these classes.
- We can estimate μ_l, σ_l, and n_l for each class.
- And compute:
$$P(v) = \frac{1}{n} \sum_l \mathcal{N}(\mu_l, \sigma_l)(v)\, n_l.$$
- We should find that $n \times P(v)$ is close to the image histogram $h(v)$.

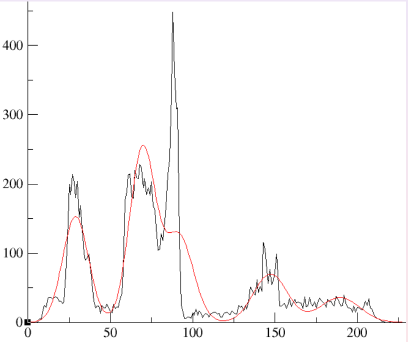
Input (5/6)

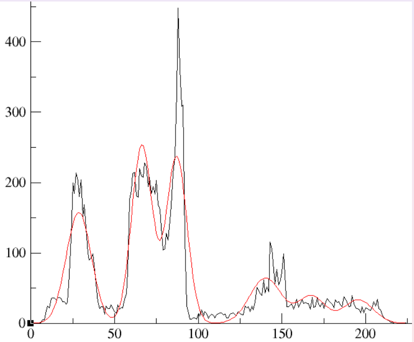
The estimates $n \times P(v)$, with k varying from 3 to 6, is depicted in red, below and in the next slide:





Input (6/6)





Now the underlying distributions really stick to data.

Output (1/X)

$$
\begin{aligned}
P(X = x \mid Y = y) &= P(Y = y \mid X = x)\, P(X = x) \,/\, P(Y = y) \\
&\propto P(Y = y \mid X = x)\, P(X = x)
\end{aligned}
$$

Assumptions:

- an input pixel value does *not* depend on what the objects located at the image other pixels are
  $P(Y_i = y_i \mid X = x) = P(Y_i = y_i \mid X_i = x_i)$
- the input pixels are independent
  $P(Y = y \mid X = x) = \Pi_i\, P(Y_i = y_i \mid X_i = x_i)$
- $P(X = x)$ is a Markovian network.

Output (2/X)

The Markovian assumption gives:

$$\begin{aligned} P(X = x) &= \Pi_i \, P(X_i = x_i \,|\, X_{\nu_i} = x_{\nu_i}) \\ &= e^{-\sum_k U_k(x_{(k)})} / Z \end{aligned}$$

where $x_{(k)}$ is a realization of the clique k.

So:

$$P(X_i = x_i \,|\, X_{\nu_i} = x_{\nu_i}) = \frac{1}{Z} \, \Pi_{k \text{ such as } X_i \in X_{(k)}} \, e^{-U_k(x_{(k)})}.$$

In the following, we shorten " k such as $X_i \in X_{(k)}$ " into " $k \ni i$ ".

Output (3/X)

We have:

$$
\begin{aligned}
P(\,X = x \,|\, Y = y\,) &\propto P(\,Y = y \,|\, X = x\,)\,P(\,X = x\,) \\
&\propto (\,\Pi_i\, P(\,Y_i = y_i \,|\, X_i = x_i\,)\,)\;(\,\Pi_{k \ni i}\, e^{-\,U_k(\,x_{(k)}\,)}\,)
\end{aligned}
$$

If we change:

$$
P(\,Y_i = y_i \,|\, X_i = x_i\,) \quad \text{into} \quad e^{-U^a(\,y_i\,;\,x_i\,)} \,/\, Z'
$$

...

Output (4/X)

...we end up with:

$$log(\,P(\,X = x\,|\,Y = y\,)\,) \;\propto\; -\sum_i \left(U^a(\,y_i;\, x_i\,) + \sum_{k \ni i} U_k(\,x_{(k)}\,) \right).$$

which can be transformed (changing U_k with multiplicative constants) into:

$$log(\,P(\,X = x\,|\,Y = y\,)\,) \;\propto\; -\left(\sum_i U^a(\,y_i;\, x_i\,) + \sum_k U_k(\,x_{(k)}\,) \right).$$

actually we are counting each clique k several times (precisely $\overline{k}$ times); these multiplicative constants can just be handled by the definition of U_k!

Output (5/X)

Maximizing $P(\,X = x \mid Y = y\,)$ is thus minimizing:

$$\sum_i \left(U^a(\,y_i;\,x_i\,) \;+\; \sum_k U_k(\,x_{(k)}\,) \right).$$

A rewriting gives:

$$P(\,X = x \mid Y = y\,) \;\propto\; e^{-\sum_k U'_k(\,x_{(k)};y_{(k)}\,)}.$$

and $P(\,X = x \mid Y = y\,)$ is also a Markov random field.

Output (6/X)

Understand that:

- U^a allows to take into account data
  - it is a *data attachment* term;
  - it relates x_i and y_i;
- U_k expresses how the solution looks like
  - it is a *regularization* term;
  - it relates x_i with its neighborhood for $\overline{k} > 1$;
  - it allows to take into account a prior when $\overline{k} = 1$.

So What?

Many problems in image processing can be expressed with U^a and U_k.

We have to maximize $U(x)$ and the search space is huge.

We can rely for instance on a “Metropolis + simulated annealing” process.

Outline

Denoising (1/5)

The input is an image corrupted by some noise and the process should remove this noise.

An output realization at every pixel is a value taken in the same space than the pixel values of the input image.

gray-levels $\longrightarrow$ gray-levels, colors $\longrightarrow$ colors...

Though iterations $x^{(t)}$ is randomly taken into that space.

Denoising (2/5)

Consider the $L^1 + TV$ model:

$$U(x) = \sum_i |x_i - y_i| + \beta \sum_i \sum_{x_j \in \nu_i} |x_i - x_j|.$$

if we choose 4-connectivity, we actually have:

$$\begin{array}{lcll} U^a(\, y_i ;\, x_i \,) & = & |x_i - y_i| & \\ U_k(\, x_{(k)} \,) & = & \beta\, |x_i - x_j| & \text{if } \overline{k} = 2 \\ U_k(\, x_{(k)} \,) & = & 0 & \text{if } \overline{k} = 1 \end{array}$$

where any clique $x_{(k)}$ of size 2 is defined by $x_{(k)} = (x_i, x_j)$.

Denoising (3/5)

When we minimize $U(x)$:

- with $|x_i - y_i|$ we ensure that x is not too far from y
  - that means that we want to keep our data!
- with $|x_i - x_j|$ we ensure that we cannot have a pixel of x whose value is too different from those of its neighbors
  - that means that we do not want to keep noise pixels!

The result is thus a *compromise* between globally keeping data *and* changing data (removing noise).

Denoising (4/5)

Denoising (5/5)

Artistic Binarization (1/3)

Consider a gray-level input image, we want a simple binary output:

- it contains large regions, respectively black and white,
- we can recognize the original image.

We have $x_i \in \mathcal{B}$ (*true* or 1 for white and *false* or 0 for black).

Assume that the input is quantized on q bit; gray values go from 0, black, to $2^q - 1$, white.

Artistic Binarization (2/3)

Choosing 4-connectivity, we actually can set:

$$U^a(\, y_i \,;\, x_i \,) = \begin{cases} y_i & \text{if } x_i = 0 \\ (2^q - 1) - y_i & \text{if } x_i = 1 \end{cases}$$

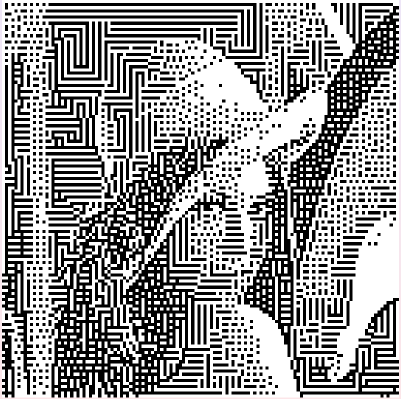
and:

$$\begin{aligned} U_k(\, x_{(k)} \,) &= \beta \, \delta_{x_i \neq x_j} && \text{if } \overline{k} = 2 \\ U_k(\, x_{(k)} \,) &= 0 && \text{if } \overline{k} = 1 \end{aligned}$$

Artistic Binarization (3/3)

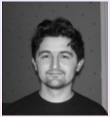
Dithering

Texturing



Classification (1/4)

Take back our favorite object recognition / classification problem:



Now

- we have $x_i \in \Omega$;
- we assume that we have learned the probability distributions related to every class ω_l, so we have estimated $P(\, y \mid \omega_l \,)$ and $P(\, \omega_l \,)$;
- we want "clean" regions in the output labeled image, meaning that, they are spatially coherent (no isolated points) and their contours are smooth (not chaotic).

Classification (2/4)

We have:

- $P(\,Y_i = y_i \,|\, X_i = \omega_l\,) \;=\; f_l(y_i)$
  - with $f_l(y_i) \;=\; \mathcal{N}(\mu_l, \sigma_l)(y_i)$
  - so the data term energy is straightforwardly:

$$U^a(\,y_i;\, l\,) \;\propto\; \frac{(y_i - \mu_l)^2}{\sigma_l^2}$$

- $P(\,X_i = \omega_l\,) \;=\; P_l$
  - with $P_l = \frac{n_l}{n}$
  - so the energy term for cliques of size 1 is:

$$U_k(\,x^{(k)}\,) \;\propto\; -log(P_l)$$

where the clique is reduced to a singleton $x^{(k)} = \{x_i\}$ and $l = x_i$.

Classification (3/4)

For cliques with size greater than 1:

- we want to handle the *context* while classifying;
- we expect regions so we must have regularization terms.

So we use the Potts model for cliques with size 2:

$$U_k(\, x^{(k)}\,) \;=\; \beta\, \delta x_i \neq x_j$$

where $x_{(k)} \;=\; (x_i,\, x_j)$.

Classification (4/4)

From left to right: the original image, the *m*-kmeans result with $k = 4$, the Markovian result with classes learned from the previous image, the former result depicted in false colors.

Exercise

Express the sudoku problem in terms of a Markov network.