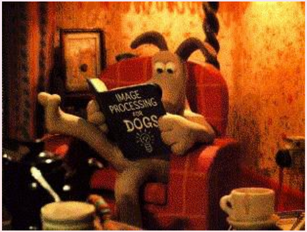
Introduction to Image Processing #3/7

Thierry Géraud

EPITA Research and Development Laboratory (LRDE)



2006

Outline

Outline

Outline

Outline

A Couple of Cross-Road Lectures

This present lecture and the next one are about

- **meta-heuristics**
  - combinatorial optimization
  - searching
  - and *probability*
- **statistics**
  - data analysis,
  - learning, estimation, classification,
  - and *probability*
- so before everything **probability**
- and we will also use some **graph**-related notions!

How All These Topics Relate to Image Processing (1/2)

Take the problem of identifying objects in an image:

- a pixel—or a higher-level primitive, e.g., a region—should be assigned to an object,
- the assignment of a pixel or a primitive to an object depends upon its spatial context in the scene,
- last there are many pixels and many objects.

How All These Topics Relate to Image Processing (2/2)

Let us rephrase this problem:

- the assignment issue can be considered within the *probability* framework,
- if we think about pixels as individuals in a population, the objects form some *classes* of individuals,
- having many variables—pixels and objects— and spatial coupling between those means that this problem is highly *combinational*.

Lecture Path (1/2)

The present lecture contents follow this path:

- first some probability-related tools are presented,
- then we present some metaheuristics,
  - and one of them is probabilistic,
- last we talk about statistics
  - through a short glance at the classification issue,
  - in order to see why we want the triad “probability + metaheuristic + statistical analysis”...

...actually that leads to **very useful methods in image processing and pattern analysis**.

Lecture Path (2/2)

The next lecture is about:

- a very little bit more about
  - graph,
  - and probability,
- an essential survival kit of statistical methods,
  - some of them with the help of probabilistic tools,
- last
  - we actually put things altogether,
  - and we browse many **applications in image processing and pattern analysis**.

Outline

Probabilities and Randoms Variables (1/3)

- *Probabilities* are numbers in [0,1] assigned to ”events” whose occurrence or failure to occur is random.
- *Random variables* are functions that map non-deterministic events to numbers.
- A *multivariate random variable* is a vector of random variables.

Probabilities and Randoms Variables (2/3)

Example:

- let denote by X the expected image result where every point X_i is assigned to an object,
- let denote by $\Omega = \{\omega_j\}$ the set of object labels,
- **we can consider a point in an image as a site for a random variable,**
- so X_i is a random variable with values in the set of possible object labels.

Probabilities and Randoms Variables (2/3)

Example (cont'd):

- the event “$X_i = x_i$” where $x_i \in \Omega$ means that the i^{th} point happens to be assigned to the object label x_i,
- we can thus express probabilities over realizations of random variables, for instance $P(X_i = x_i)$,
- since $X = \{X_i\}$, we can consider X as a multivariate random variable.
- and we we can thus express probabilities over realizations of X, for instance $P(X = x | Y = y)$ with y the input image.

Some Formulas (1/2)

In the following, A and B are events.

- $P(A \cap B)$ is the joint probability of A and B *(probability of simultaneously getting both A and B)*;
  - we have $P(A \cap B) = P(B \cap A)$,
  - if A and B are mutually exclusive, we have $P(A \cap B) = 0$.
- A and B are independent iif $P(A \cap B) = P(A)\,P(B)$.
- $P(A \cup B)$ is the probability of having either A or B
  - we have $P(A \cup B) = P(A) + P(B) - P(A \cap B)$,
  - that's not an interesting probability for this lecture!

Some Formulas (2/2)

Cont'd:

- $P(A|B)$ is the (conditional) probability of A given B *(probability of getting A when B occurs)*;
  - we have $P(A|B) = P(A \cap B)/P(B)$,
  - it is more comprehensive this way:

$$P(A \cap B) = P(B)\,P(A|B).$$

- $L(A|B)$ is the likelihood of A given B *(the reverse way to reason about conditional probabilities)*;
  - we have $L(A|B) = P(B|A)$.

Bayes' Theorem

The Bayes' theorem naturally follows:

$$
\begin{aligned}
P(A|B) &= \frac{P(B|A)\,P(A)}{P(B)} = \frac{L(A|B)\,P(A)}{P(B)} \\
&\propto L(A|B)\,P(A)
\end{aligned}
$$

It should be understood this way:

$$
\textit{posterior probability} = \frac{\textit{likelihood} \times \textit{prior probability}}{\textit{normalizing constant}}
$$

Applying Bayes' Theorem to Our Problem (1/3)

- We are looking for the image x_{sol} of object labels and we have the input image y.
- Let us consider that both x_{sol} and y are the respective realization of the sets, X and Y, of random variables

  one variable per input and output pixel.
- We could want to maximize $P(X = x \,|\, Y = y)$ *(get the most likely solution given the input)*:
  - that is, to find x_{sol} such as $\forall x,\ P(X = x_{sol} \,|\, Y = y) \geq P(X = x \,|\, Y = y)$,
  - put differently,

$$x_{sol} \;=\; arg\ \max_{x} P(X = x \,|\, Y = y).$$

Applying Bayes' Theorem to Our Problem (2/3)

Let's rock:

$$\begin{aligned}
x_{sol} &= arg \max_x P(X = x \mid Y = y) \\
&= arg \max_x \frac{P(Y = y \mid X = x)\, P(X = x)}{P(Y = y)} \\
&= arg \max_x P(Y = y \mid X = x)\, P(X = x) \\
&= arg \max_x L(X = x \mid Y = y)\, P(X = x)
\end{aligned}$$

That result sounds very natural!

Applying Bayes' Theorem to Our Problem (3/3)

So we have:

- to be able to compute both $L(X = x \mid Y = y)$ and $P(X = x)$,
- and to be able to find the global maximum of a function...

and those two objectives are really **not** easy ones!

Outline

The Discrete Uniform Distribution (1/2)

Consider:

- a set of m values $\{v^{(j)}\}$,
- where their random variable V is the discrete uniform distribution.

We have:

$$\forall j \in [1, m] \;\; P(V = v^{(j)}) = \frac{1}{m}$$

We want a sample v with respect to the law $P(V = v^{(j)})$.

The Discrete Uniform Distribution (2/2)

With the C language under a un*x:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <assert.h>

/* the number of possible values */
#define m 3

int main()
{
  srand(getpid());

  float skip = 0,
        /* the array of possible values */
        v_[m+1] = { skip, /* fill me */ };

  /* j is an index taken randomly */
  unsigned j = 1 + (unsigned)(m * rand() / (RAND_MAX + 1.));
  assert(j >= 1 && j <= m);

  float v = v_[j];
  /* done! */

  return EXIT_SUCCESS;
}
```

With Independent Variables

When you have n independent variables V_i:

- that is, the multivariate random variable $X = \{V_i\}$,
- with the random laws $P(V_i = v_i)$,

then you just have to get random numbers—samples—from each V_i independently from the others to get a sample from X.

But:

- a random law is usually **not** uniform!
- variables are **not** always independent...
- ...and the joint distribution is usually **unknown**!

The Case of Non Uniform Laws (Easy)

Exercise:

Design an algorithm to pick random numbers amongst $\{v^{(1)}, v^{(2)}, v^{(3)}\}$ such as:

$$\begin{aligned} P(V = v^{(1)}) &= 0.5 \\ P(V = v^{(2)}) &= 0.2 \\ P(V = v^{(3)}) &= 0.3 \end{aligned}$$

Then generalize to a variable whose discrete law is known by probabilities $P(V = v^{(j)})$ where $j \in [1, m]$.

The Case of Non Independent Variables

Now that's harder!

- Consider the multivariate random variable $X = \{V_1, \ldots, V_n\}$,
- a realization of X is a vector $x = \{v_1, \ldots, v_n\}$ thus we are in an n-dimensional value space!
- we will create a *random walk* in this space that is a sequence x^t of realizations where t denotes time (iteration index)
- our objective is that $\lim_{t\to\infty} x^t$ is a sample taken w.r.t. the law $P(X = x)$!

A Few Remarks

- Actually, we also want that:

$$P(X = x \mid X^0 = x^0) = P(X = x)$$

  so that we do not care about the process initialization (our first realization at step $t = 0$).
- This is an important problem!
  - so the next slides are about algorithms to get a sample of a multivariate random variable whose components are not independent.
- *Just realize that one following part of this lecture is about finding (within an acceptable delay) the solution of a difficult problem in a large search space...*
  - Exercise: just explain this remark!

The Metropolis Algorithm (1/3)

Consider that $P(X = x)$ can be expressed as follows:

$$P(X = x) \; = \; \frac{e^{-U(x)}}{Z}$$

where Z is a normalizing constant and $E(x)$ an energy function.

We can express a transition as an energy change:

$$\Delta U(x^{old} \to x^{new}) \; = \; U(x^{new}) - U(x^{old})$$

and we have:

$$\frac{P(X = x^{new})}{P(X = x^{old})} \; = \; e^{-\Delta U(x^{old} \to x^{new})}$$

The Metropolis Algorithm (2/3)

$$P(X = x^{new}) \,/\, P(X = x^{old})$$

can be interpreted as the *probability of the transition* from x^{old} to x^{new}.

This probability is:

$$P(x^{old} \rightarrow x^{new}) \;=\; e^{-(\,U(x^{new}) - U(x^{old})\,)}.$$

The Metropolis Algorithm (3/3)

- *initialization:* take an initial realization x^0.
- *repeat:*
  - randomly pick a realization x'
  - compute $\alpha = \dfrac{P(x')}{P(x^t)}$
  - update:
    $x^{t+1} = x'$ if $\alpha > 1$ or with the probability α if $\alpha < 1$
    $x^{t+1} = x^t$ otherwise
  - increment t

The Gibbs Sampling Algorithm

- *initialization:* take an initial realization x^0.
- *repeat:*
  - randomly pick a variable V_i
  - compute the law $P(X = x')$
    with $x' = \{v_1^t, .., v_{i-1}^t, v', v_{i+1}^t, v_n^t\}$
    for all possible value v' of V_i
  - update:
    $x^{t+1} = x'$ with the probability $P(X = x')$,
    $x^{t+1} = x^t$ otherwise
  - increment t

URLs (1/2)

- theory http://en.wikipedia.org/wiki/Probability_theory
- axioms http://en.wikipedia.org/wiki/Probability_axioms
- conditional probability
  http://en.wikipedia.org/wiki/Conditional_probability
- likelihood function
  http://en.wikipedia.org/wiki/Likelihood_function
- prior probability
  http://en.wikipedia.org/wiki/Prior_probability
- posterior probability
  http://en.wikipedia.org/wiki/Posterior_probability
- probability distributions
  http://en.wikipedia.org/wiki/Probability_distribution

URLs (2/2)

- Bayesian statistics
  http://en.wikipedia.org/wiki/Bayesian_statistics
- Metropolis-Hastings algorithm
  http://en.wikipedia.org/wiki/Metropolis-Hastings_algorithm
- Gibbs sampling algorithm
  http://en.wikipedia.org/wiki/Gibbs_sampling
- Markov chain http://en.wikipedia.org/wiki/Markov_chain
- Monte Carlo
  http://en.wikipedia.org/wiki/Monte_Carlo_method
- MCMC
  http://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo
- Bayesian network
  http://en.wikipedia.org/wiki/Bayesian_network

Just for Fun

Have you ever think of solving Sudoku
as a *probabilistic* problem?

Why?

Subliminal image:

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

Outline

Definitions

- *Heuristic:* particular technique of directing one's attention in learning, discovery, or problem-solving.

- *Metaheuristic:* general method for solving computational problems by combining heuristics in an efficient way.

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

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

Common Metaheuristics

- Exact search
  - brute-force
  - branch and bound
- “Path-based” search
  - greedy algorithm
  - hill-climbing
  - gradient descent
  - tabu search
- Randomized search
  - random optimization
  - genetic algorithms
  - simulated annealing

Brute-Force Search

Systematically enumerate all possible candidates for the solution and check whether each candidate satisfies the problem's statement.

However:

- its cost is proportional to the number of candidate solutions
- it is thus not relevant when the search space is too large

http://en.wikipedia.org/wiki/Brute-force_search

Branch and Bound Search (1/2)

The how-to:

- structure the search space into a search *tree*
    - a branch is a sub-space
    - the childhood relationship represents space spliting
- use *bounds*
    - the optimal solution in a sub-space is between upper and lower bounds
    - the minimum upper bound seen among all sub-spaces examined so far is recorded (m)

Branch and Bound Search (2/2)

The how-to (cont'd):

- search the tree
  - when inspecting a node (sub-space), its bounds are computed
  - if its lower bound is greater than *m* then prune this branch
  - otherwise, update *m* if needed and inspect children

Exercise:
express the closest point search in a binary image as a branch-and-bound problem.

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

Path-based Searches (1/2)

- greedy algorithms:
  - at each stage take the locally optimum candidate
  - `http://en.wikipedia.org/wiki/Greedy_algorithm`
- hill-climbing (in a graph):
  - at each stage move to an adjacent candidate to get closer to the solution
    `http://en.wikipedia.org/wiki/Hill_climbing`
- gradient-descent (with an objective function):
  - at each stage take a step proportional to the negative of the gradient
    `http://en.wikipedia.org/wiki/Gradient_descent`

Path-based Searches (2/2)

- tabu search:
  - clever and efficient local search method using a memory
    `http://en.wikipedia.org/wiki/Tabu_search`

Exercise:
express the closest point search in a binary image as a local search.

`http://en.wikipedia.org/wiki/Local_search_(optimization)`

Outline

As luck would have it...

When

- your search space is huge
- you do not want to be trapped by a *l*ocal solution knowing that the global solution or a better one is elsewhere...
- your problem is in NP

you need a random optimization method!

http://en.wikipedia.org/wiki/NP_(complexity)

Random Optimization (1/2)

Many problems belong to the class of

- global optimization
  - when you cannot rely on local optimization
  - when the problem variables are highly coupled
- and combinatorial optimization
  - when the set of feasible solutions can be reduced to a discrete one

Random Optimization (2/2)

Random optimization methods outperform other methods with significantly faster convergence towards the global solution.

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

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

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

Simulated Annealing

Simulated annealing is

- a generic probabilistic meta-algorithm for global optimization problems
- inspired from annealing in metallurgy where heat
  - unstucks atoms from their initial positions (a local minimum of the internal energy)
  - causes atoms to wander randomly through states of higher energy
  - is very slowly decreased to get some chances of finding configurations with lower internal energy than the initial one.

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

Analogies

The analogies between this method and metallurgy are listed below:

problem	$\leftrightarrow$	physical system
objective function	$\leftrightarrow$	system internal energy
search space	$\leftrightarrow$	set of system states
a feasible solution	$\leftrightarrow$	a system state
expected solution	$\leftrightarrow$	state of global energy minimum
a slight change in search space	$\leftrightarrow$	a transition to a neighbor state
a problem variable	$\leftrightarrow$	a component of the state
a descent	$\leftrightarrow$	decreasing energy

Exercise:
express analogies for genetic algorithms.

Algorithm

- *initialization:*
  - pick an initial state, s^0
  - set the temperature a high enough initial value, T^0
- *iteration:*
  - consider some neighbour s' of the current state s^t
  - compute the probability $P(s^t \rightarrow s', T^t)$ of this transition
  - update:
    $s^{t+1} = s'$ with this probability
    $s^{t+1} = s^t$ otherwise
  - increment t
    nota bene: T^{t+1} is slightly lower than T^t

When Metropolis and Simulated Annealing Meet (1/5)

On one hand, we have:

- **the Metropolis algorithm**
- $x_{\lim} = \lim_{t\to\infty} x^t$ is a sample taken w.r.t. the law $P(X = x)$
- where $x_{\lim}$ is obtained in *infinite* time :-(
- and it uses $P(X = x) = e^{-U(x)}/Z$ and $P(x^t \to x^{t+1}) = e^{-(U(x^{t+1})-U(x^t))}$.

On the other hand, we have:

- **the simulated annealing**
- $s_{sol} = \lim_{t\to\infty} s^t$ is the value that maximizes $P(s)$
- where s_{sol} is obtained in *finite* time :-)
- and it uses $P(s^t \to s^{t+1}, T^t)$.

When Metropolis and Simulated Annealing Meet (2/5)

Focus on these formulas and their meaning:

- $P(X = x) = \frac{e^{-U(x)}}{Z}$
- $P(x^t \to x^{t+1}) = e^{-(U(x^{t+1}) - U(x^t))}$

So are you able to guess the definitions of:

- $P(X = x, T)$
- $P(x^t \to x^{t+1}, T)$

Hints:
what happens when the temperature is high?
and when it is low?

When Metropolis and Simulated Annealing Meets (3/5)

Actually we have:

$$P(\,X = x,\,T\,) \;=\; \frac{e^{-U(x)/T}}{Z}$$

$$P(\,x^t \rightarrow x^{t+1},\,T\,) \;=\; e^{-(\,U(x^{t+1}) - U(x^t)\,)/T}$$

Exercise: consider that the probability $P(X = x)$ has exactly *one* maximum, for $x = x_{sol}$,

- proove that $\lim_{T \rightarrow +\infty} P(\,X = x,\,T\,)$ is a uniform law,
- proove that $\lim_{T \rightarrow 0} P(\,X = x_{sol},\,T\,) \;=\; 1.$

When Metropolis and Simulated Annealing Meets (4/5)

We have:

- $\lim_{t\to\infty} x^t \; = \; x_{\lim}$ is a sample taken w.r.t. $P(X = x)$
- $\lim_{T\to 0} P(\, X = x_{sol}, \, T \,) \; = \; 1$

Put in words:

- we know how to get a value from any random variable,
- we have a random variable, $P(X = x, T \to 0)$, whose only possible value is x_{sol} that maximizes $P(X = x)$.

When Metropolis and Simulated Annealing Meets (5/5)

When we mix those two limits, we expect that:

- $\lim_{t\to\infty} \left(\lim_{T\to 0} x^t \right)$ is a sample of $P(\, X = x,\, T = 0\,)$
- this sample can only be x_{sol}
- and this double limit is computable in *finite* time!!!

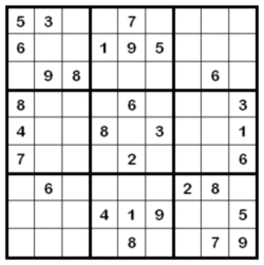
Conclusion: **iif** the convergence $\lim_{t\to\infty} T^t = 0$ is slow enough:

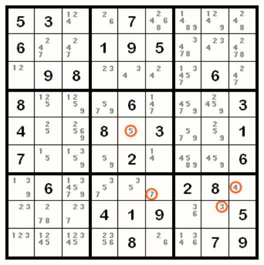
- we have: $\lim_{t\to\infty} P(\, X = x^t,\, T^t\,) = 1$
- so: $\lim_{t\to\infty} x^t = arg \max_x P(X = x)$.

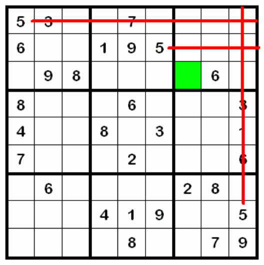
Outline

A Problem to Solve and Some Data

You have already seen that:







What do the above figures depict?

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

Exercise!

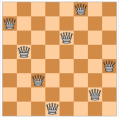
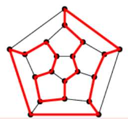
To think different, answer these questions:

- What the search space can be?
- What the system energy can be?
- How to easily solve a Sudoku problem?

See Also (1/2)

- Other methods:
  - random-restart hill climbing
    `http://en.wikipedia.org/wiki/Random-restart_hill_climbing`
  - greedy randomized adaptive search procedure (GRASP)
    `http://en.wikipedia.org/wiki/Greedy_randomized_adaptive_search_procedure`
  - swarm intelligence
    `http://en.wikipedia.org/wiki/Swarm_intelligence`
  - genetic algorithms
    `http://en.wikipedia.org/wiki/Genetic_algorithms`

See Also (2/2)

- Famous problems:
  - the eight queens
    http://en.wikipedia.org/wiki/Eight_queens_problem
    
  - the travelling salesman problem
    http:
    //en.wikipedia.org/wiki/Traveling_salesman_problem
    

Outline

About Statistics and Data

- *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.

`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...

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,

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

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

Statistics is About Counting (1/2)

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

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

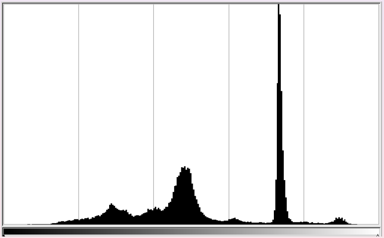
where:

- 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 (2/2)

For instance, with I (left figure below), we get h (right figure below); the x-axis shows increasing gray-levels from black (left) to white (right):



Although the image is not well-contrasted, we clearly see (at least) 6 clusters / classes (histogram peaks).

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.

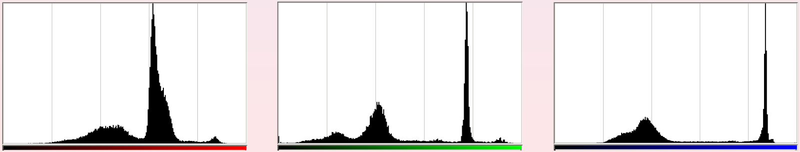
- When we have color image,
  - for instance, encoded on red-green-blue (RGB for short) with 8 bit per component,
  - then we have a straightforward 3-dimensional feature space.
- When we have texture information,
  - for instance, we have computed some characteristics of the local texture around each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

Outline

Example (1/3)

Now consider this color image:

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

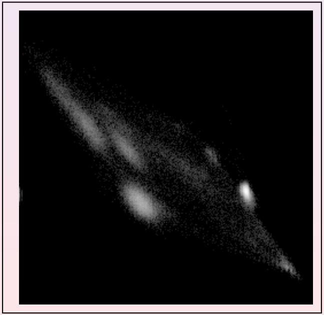


Example (2/3)

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

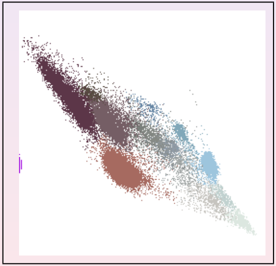
FIXME: insert a picture here!

or in 2D if we discard the blue component:



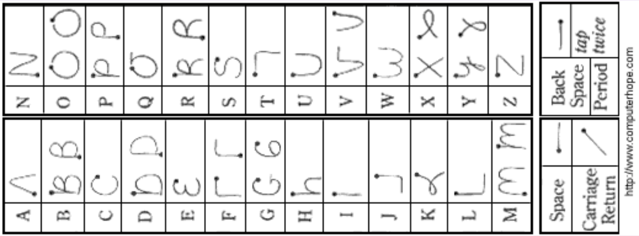
Example (3/3)

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


Exercise

Consider the Palm Pilot alphabet:



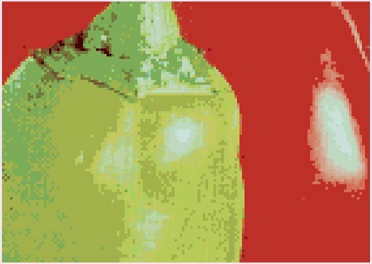
- Express character recognition in terms of a data analysis problem.
- Imagine different sets of some relevant features.

Outline

A Classification Result

Let Us Have a Closer Look (1/2)

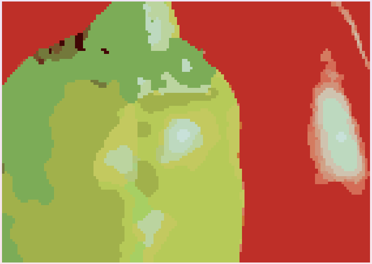
Do you prefer:



...

Let Us Have a Closer Look (2/2)

... or:



?

Re-Phrasing the Question

Another way to put the same question is:

which classification result will lead to an **easier** image understanding process?

Exercise:
tell why!

Conclusion As an Exercise

Consider the following questions:

- 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?
- what is the color of a pixel of a green pepper?

Exercise:

- Express their answers in a scientific way.
- Model the recognition problem.
- Provide us with a solution.