PROBLÈME DE CLASSIFICATION DE DENSITÉ

Automates cellulaires probabilistes à l'usage du problème de classification de densité

Numéro d'inscription: 19341

Transport d'information des réseaux distribués

- Pas d'autorité centrale dans les grands réseaux informatiques actuels

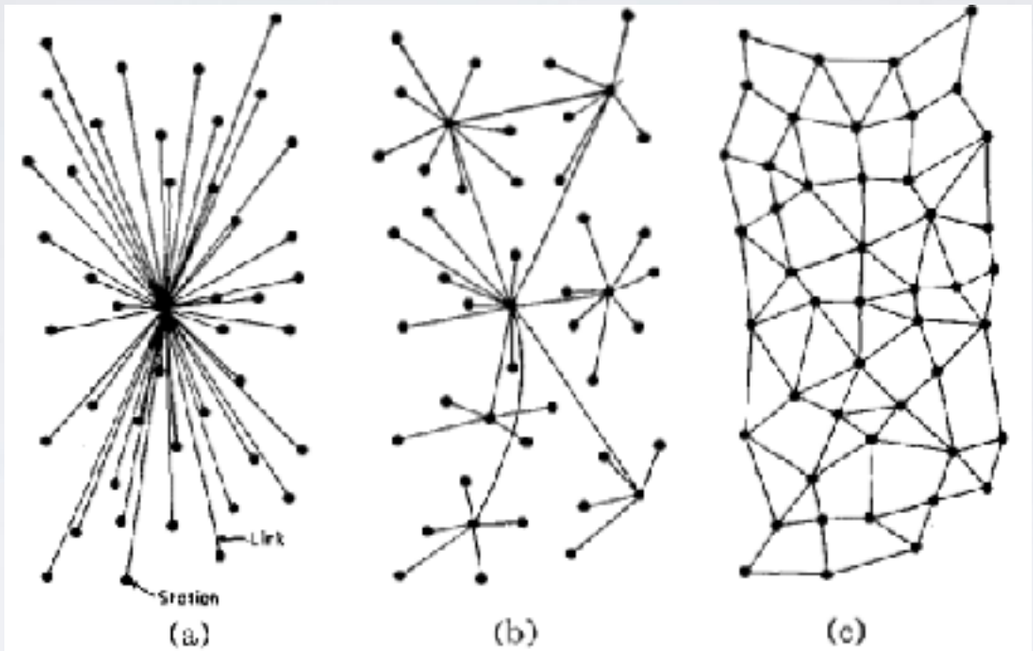


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

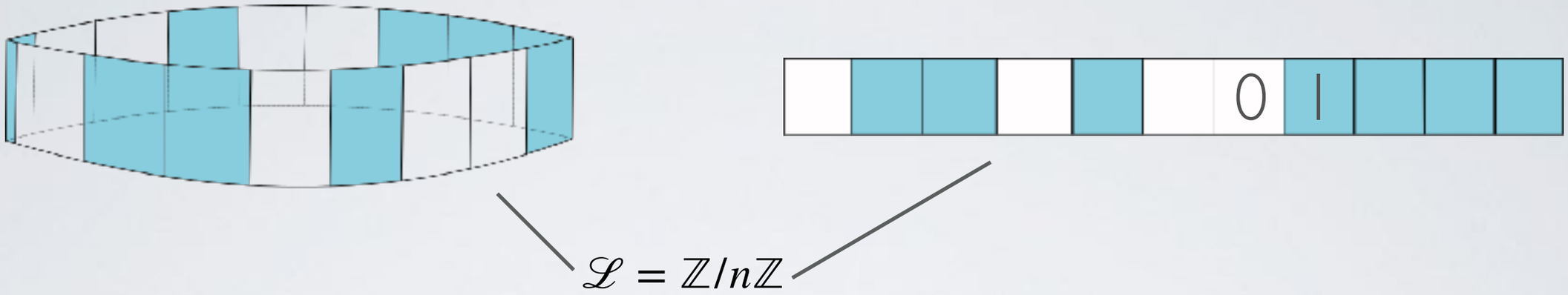
Comment faire émerger une information globale en ne collectant seulement que des informations locales à chaque étape?

Comment faire immerger une information globale en collectant seulement une information locale à chaque étape?

1. Cadre théorique et présentation des notions
2. Approximation arbitraire d'une solution
3. Simulations de la solution approchée

1. Cadre théorique et présentation des notions

Formalisation du problème



Chaque élément de l'ensemble est appelé cellule

Représentons $\mathscr{L} = \mathbb{Z}/n\mathbb{Z}$ arrangé dans un anneau

A un instant (discret) donné, chaque cellule peut prendre un état dans $\{0,1\}$(resp. blanc/bleu)

L'état de l'ensemble des cellules à un instant donné est appelé configuration.

On note $\mathscr{E}_n = \{0,1\}^{\mathscr{L}}$ l'ensemble des configurations et on a $|\mathscr{E}_n| = 2^n$

Pour $p \in \{0,1\}$ notons $|x|_p$ le nombre d'occurence de p dans la configuration x

On définit la densité $\rho(x)$ d'une configuration $x \in \mathscr{E}_n$ par: $\rho(x) = \dfrac{|x|_1}{n}$

Afin d'éviter le cas où $|x|_0 = |x|_1$,on supposera n impair.

Notons $1 = 1^{\mathscr{L}}$ et $0 = 0^{\mathscr{L}}$ les configurations constituées uniquement de 1 et de 0

I. Cadre théorique et présentation des notions

Automates cellulaires élémentaires

Fonction de transition locale: $\phi : \{0,1\}^3 \to \{0,1\}$

Règle de transition globale $\Phi : \mathscr{E}_n \to \mathscr{E}_n$ associée à ϕ, fonction qui à tout configuration x^t renvoie la configuration x^{t+1} telle que:

$$\forall c \in \mathscr{L}, x_c^{t+1} = \phi(x_{c-1}^t, x_c^t, x_{c+1}^t)$$

Exemple: Règle 184 ou règle du trafic

Fonction de transition:

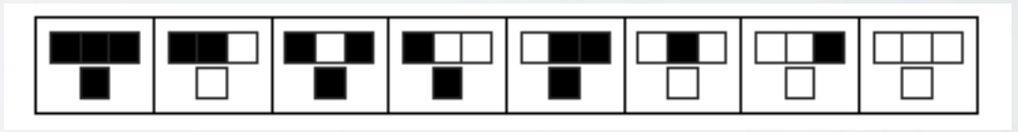
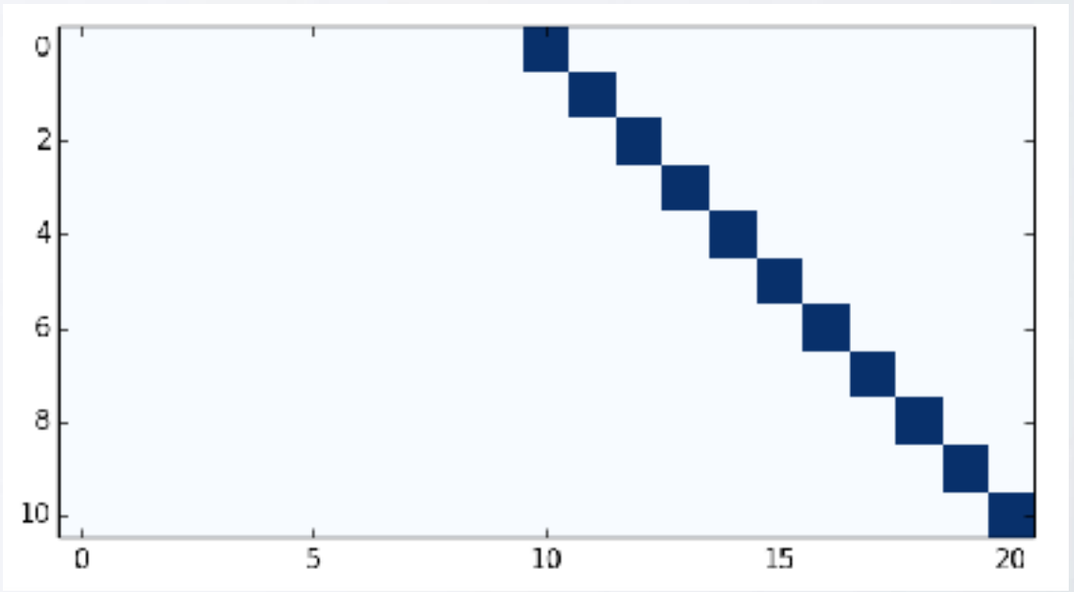


Diagramme espace-temps:



I. Cadre théorique et présentation des notions

Automates cellulaires élémentaires probabilistes

Fonction de transition: $f : \{0,1\}^3 \to [0,1]$

Règle de transition globale $F : \mathscr{E}_n \to \mathscr{E}_n$ associée à ϕ, fonction qui à tout configuration aléatoire x^t renvoie la configuration aléatoire x^{t+1} telle que:

$$\forall c \in \mathscr{L}, x_c^{t+1} = \mathscr{B}_c^t(f(x_{c-1}^t, x_c^t, x_{c+1}^t))$$

où $(\mathscr{B}_c^t)_{c\in\mathscr{L},t\in\mathbb{N}}$ est une suite de variables aléatoires de Bernoulli

Classificateur de densité

On dit que la configuration x est un point fixe pour la fonction F avec probabilité 1.

On dit que F est un classificateur (de densité) si 1 et 0 sont les seuls points fixes de F

Pour un classificateur $\mathscr{C}$ on définit le temps de classification $T(x)$, variable aléatoire à valeurs dans $\mathbb{N} \cup \{+\infty\}$telle que:

$$T(x) = \min\{t \,|\, x^t \in \{0,1\}\}$$

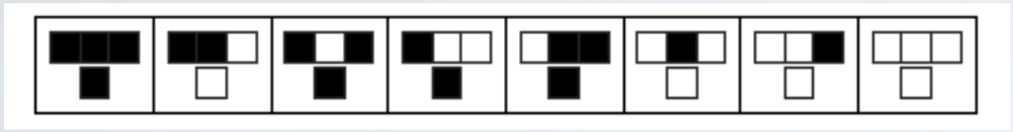
$\mathscr{C}$classifie correctement une configuration x si:

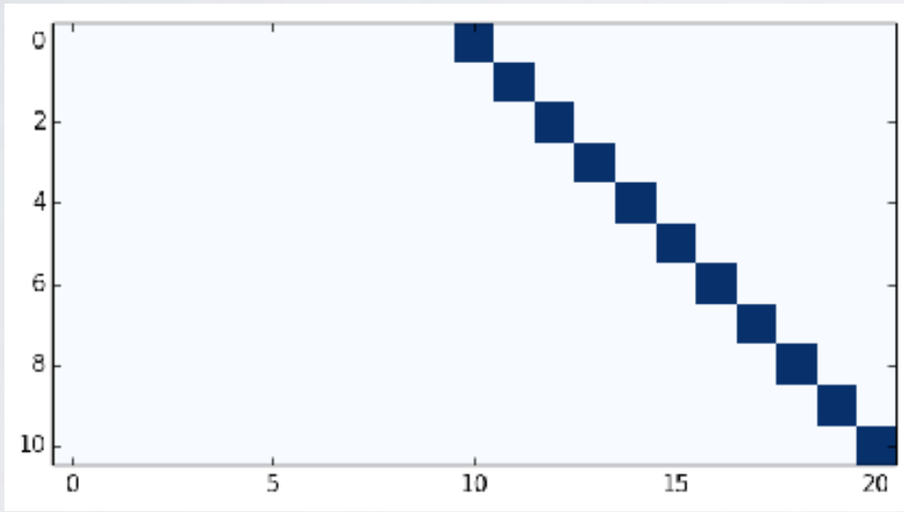
$$(i)\ T(x) \text{ est fini}$$
$$(ii)\ x^{T(x)} = 1 \text{ si } \rho(x) > 1/2 \text{ et } x^{T(x)} = 0 \text{ si } \rho(x) < 1/2$$

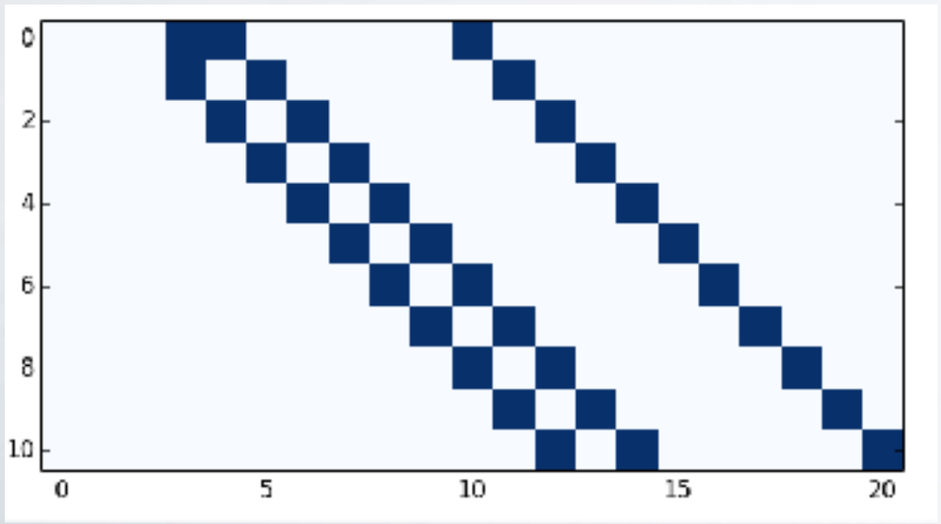
2. Approximation arbitraire d'une solution

Définition: Pour $q \in \{0,1\}$, on dit qu'une configuration x est q-archipelago si toutes ses cellules à l'état q dont isolées.

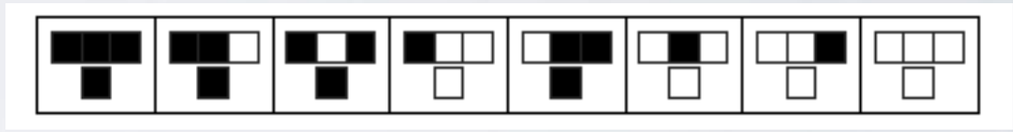
Règle 184 ou règle du trafic

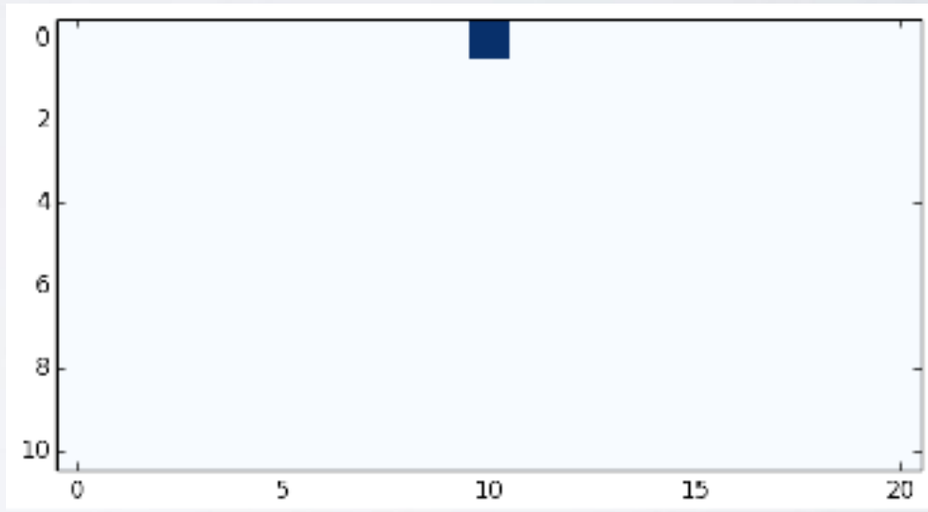


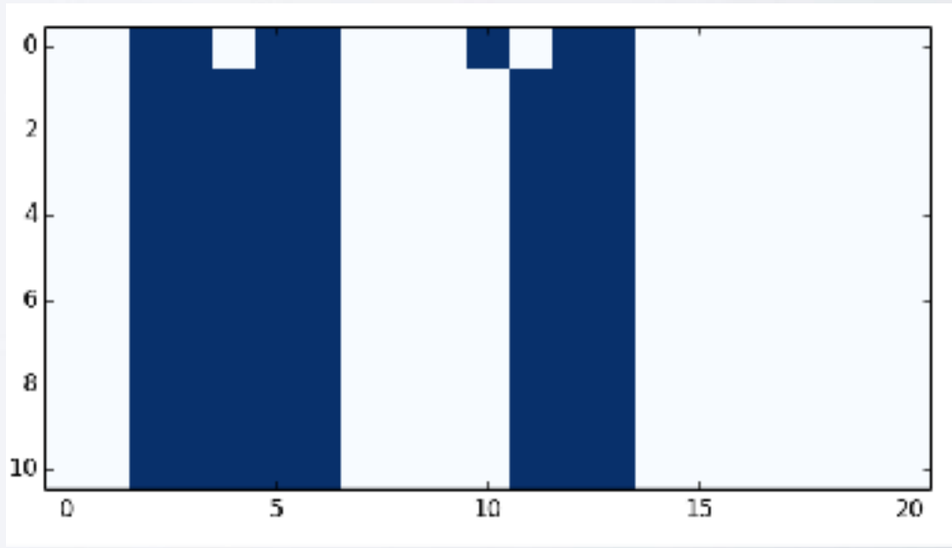




Règle 232 ou règle de la majorité





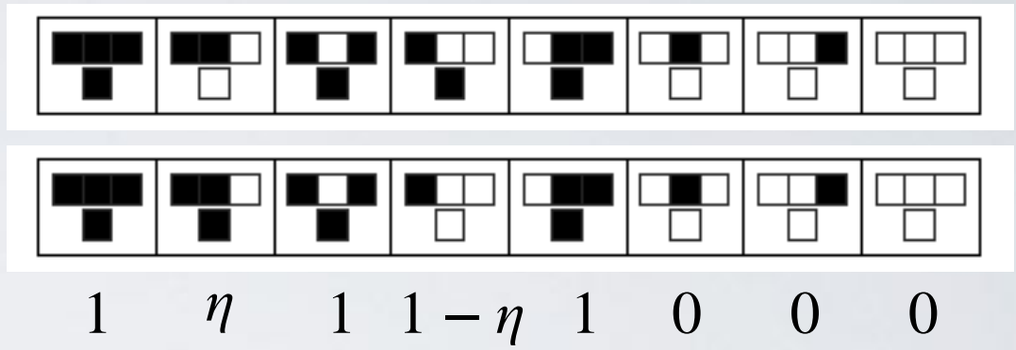


2. Approximation arbitraire d'une solution

Classificateur $\mathscr{C}$

Règle du trafic avec probabilité $1-\eta$

Règle de la majorité avec probabilité η



Théorème: Pour tout $p \in [0,1[$,il existe un paramètre η pour le classificateur $\mathscr{C}$ tel que $\mathscr{C}$ classifie bien la densité avec probabilité supérieure à p

Lemme 1: Un archipelago est bien classifié avec probabilité 1.

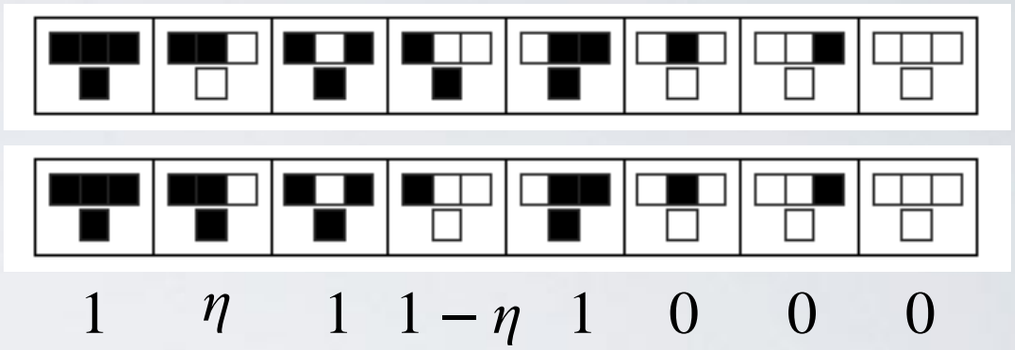
Lemme 2: Pour tout $p \in [0,1[$,il existe un paramètre η pour le classificateur $\mathscr{C}$ tel que pour toute configuration $x \in \mathscr{E}_n$, , la probabilité d'évolution vers un archipelago x_A de densité $\rho(x) = \rho(x_A)$ est supérieure à p .

2. Approximation arbitraire d'une solution

Classificateur $\mathscr{C}$

Règle du trafic avec probabilité $1-\eta$

Règle de la majorité avec probabilité η



Lemme 1: Un archipelago est bien classifié avec probabilité 1.

Montrons d'abord que le successeur d'un q-archipelago est q-archipelago.

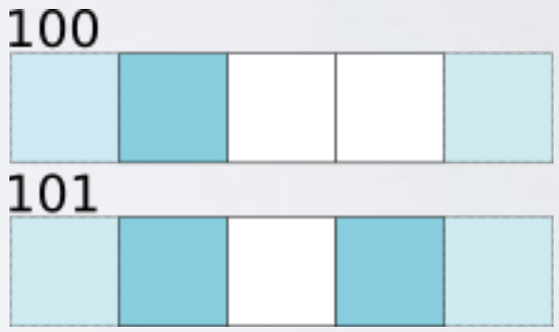
Soit $x \in \mathscr{E}_n$ q-archipelago . WLOG, x est 1-archipelago.

Soit y un potentiel successeur de x.

Notons $C := \{c \in \mathscr{L} \,|\, y_c = 1\}$.

Pour $c \in C$, on a $(x_{c-1}, x_c, x_{x+1}) \in \{111,110,101,100,011\}$.

Or x est 1-archipelago donc: $(x_{c-1}, x_c, x_{x+1}) \in \{100,101\}$.



Et il n'est pas possible de reproduire ces motifs en se décalant d'un pas, donc $y_{c-1} = 0, y_{c+1} = 0$.

Montrons que x tend vers la configuration 0.

Puisque n est impair, il existe à tout instant t un motif 100 disparaissant avec probabilité η.

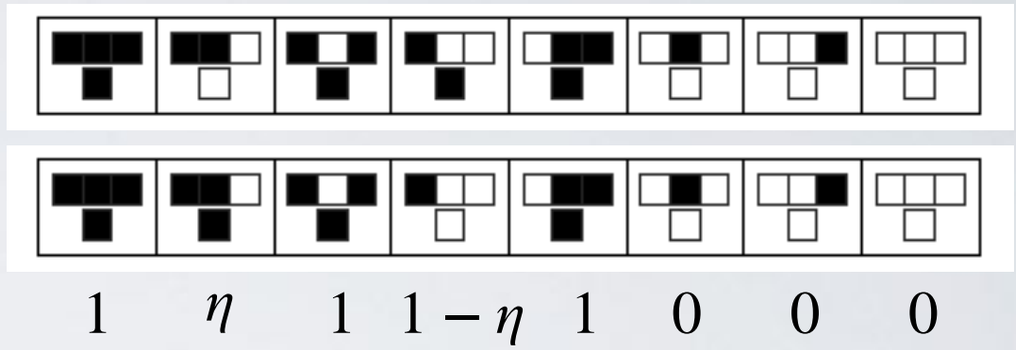
Ainsi, $(|x^t|_1)_{t\in\mathbb{N}}$ est une suite décroissante tendant vers 0 avec probabilité 1.

2. Approximation arbitraire d'une solution

Classificateur $\mathscr{C}$

Règle du trafic avec probabilité $1-\eta$

Règle de la majorité avec probabilité η



Lemme 2: Pour tout $p \in [0,1[$,il existe un paramètre η pour le classificateur $\mathscr{C}$ tel que pour toute configuration $x \in \mathscr{E}_n$, , la probabilité d'évolution vers un archipelago x_A de densité $\rho(x) = \rho(x_A)$ est supérieure à p .

On admettra que la règle du trafic permet de faire évoluer toute configuration vers un archipelago en au plus $\lceil \frac{n}{2} \rceil$ étapes.

Φ : fonction globale de la règle du trafic et $y^t = \Phi^t(x)$

$p \in [0,1[$, $n \in \mathbb{N}$. WLOG, $\rho(x) < 1/2$

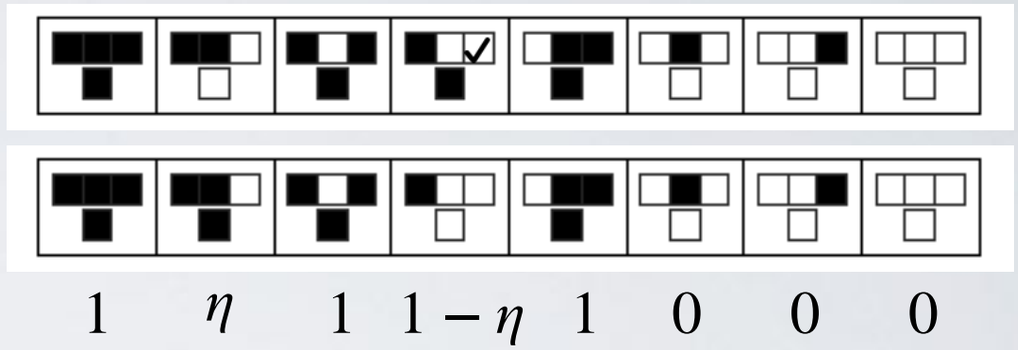
Évaluons la probabilité que $\mathscr{C}$ agisse comme la règle du trafic dans les $T = \lceil \frac{n}{2} \rceil$ premières étapes.

2. Approximation arbitraire d'une solution

Classificateur $\mathscr{C}$

Règle du trafic avec probabilité $1-\eta$

Règle de la majorité avec probabilité η



On admettra que la règle du trafic permet de faire évoluer toute configuration vers un archipelago en au plus $\lceil \frac{n}{2} \rceil$ étapes.

Φ : fonction globale de la règle du trafic et $y^t = \Phi^t(x)$

$p \in [0,1[, n \in \mathbb{N}$. WLOG, $\rho(x) < 1/2$

Évaluons la probabilité que $\mathscr{C}$ agisse comme la règle du trafic dans les $T = \lceil \frac{n}{2} \rceil$ premières étapes.

Soit $D^t = |\{c \in \mathscr{L}, x_c^t = y_c^t\}|$

Remarquons que $\mathscr{C}$ et Φ sont différents pour les voisinages 110 et 100.

Et: $\forall x \in \mathscr{E}_n, |x|_{001} = |x|_{100}$ et $|x|_{011} = |x|_{110}$ Donc: $|x|_{001} + |x|_{110} \leq \lceil \frac{n}{2} \rceil$

$|x|_{001} + |x|_{100} + |x|_{011} + |x|_{110} \leq n$

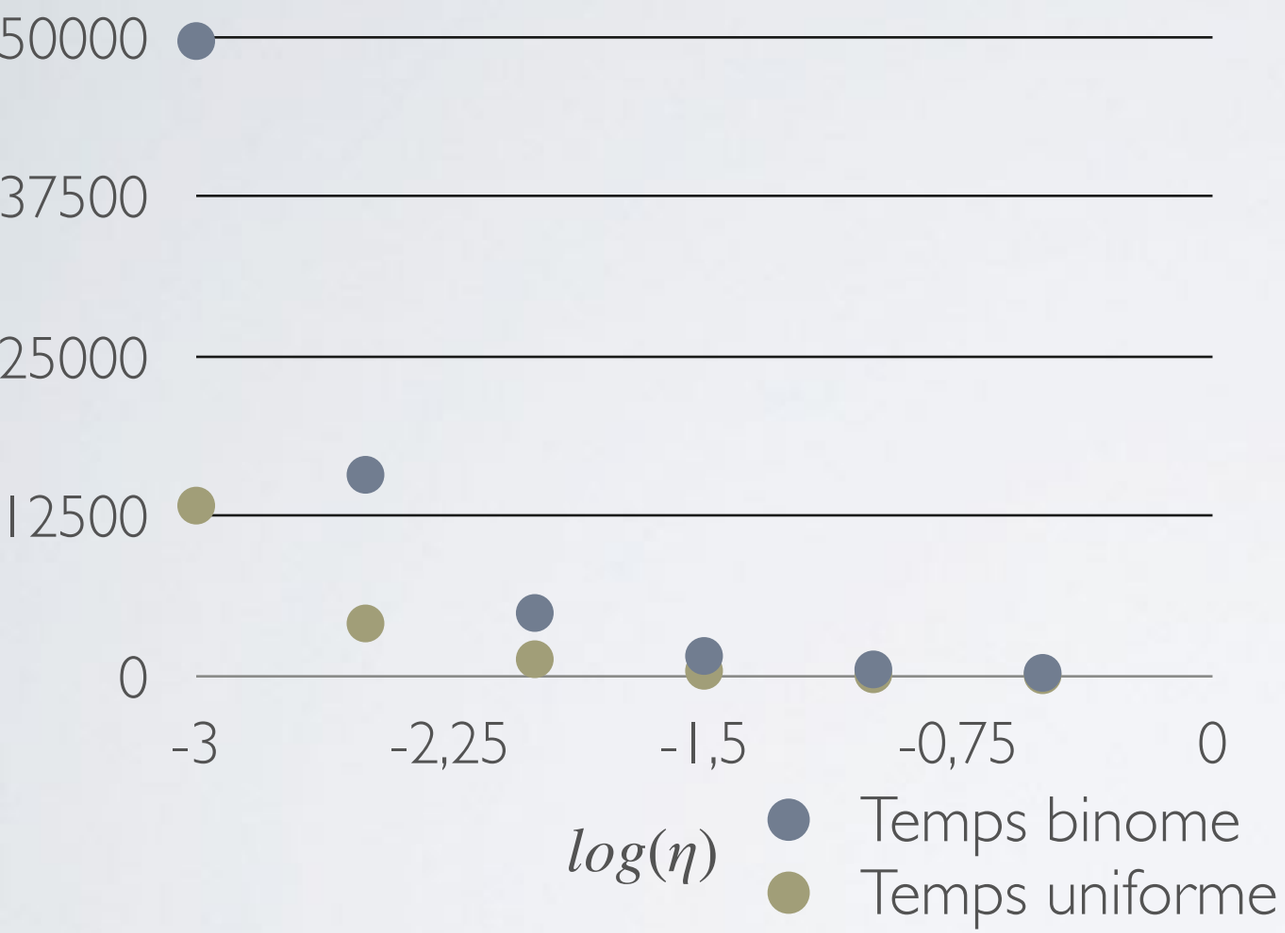
La probabilité p_{id} que $\mathscr{C}$ se comporte comme Φ durant les T premières étapes est donc telle que:

$$p_{id} = \prod_{t=1}^{T} (1-\eta)^{(|x^t|_{110}+|x^t|_{100})} \geq (1-\eta)^{\lceil \frac{n}{2} \rceil T} \geq (1-\eta)^{T^2}$$

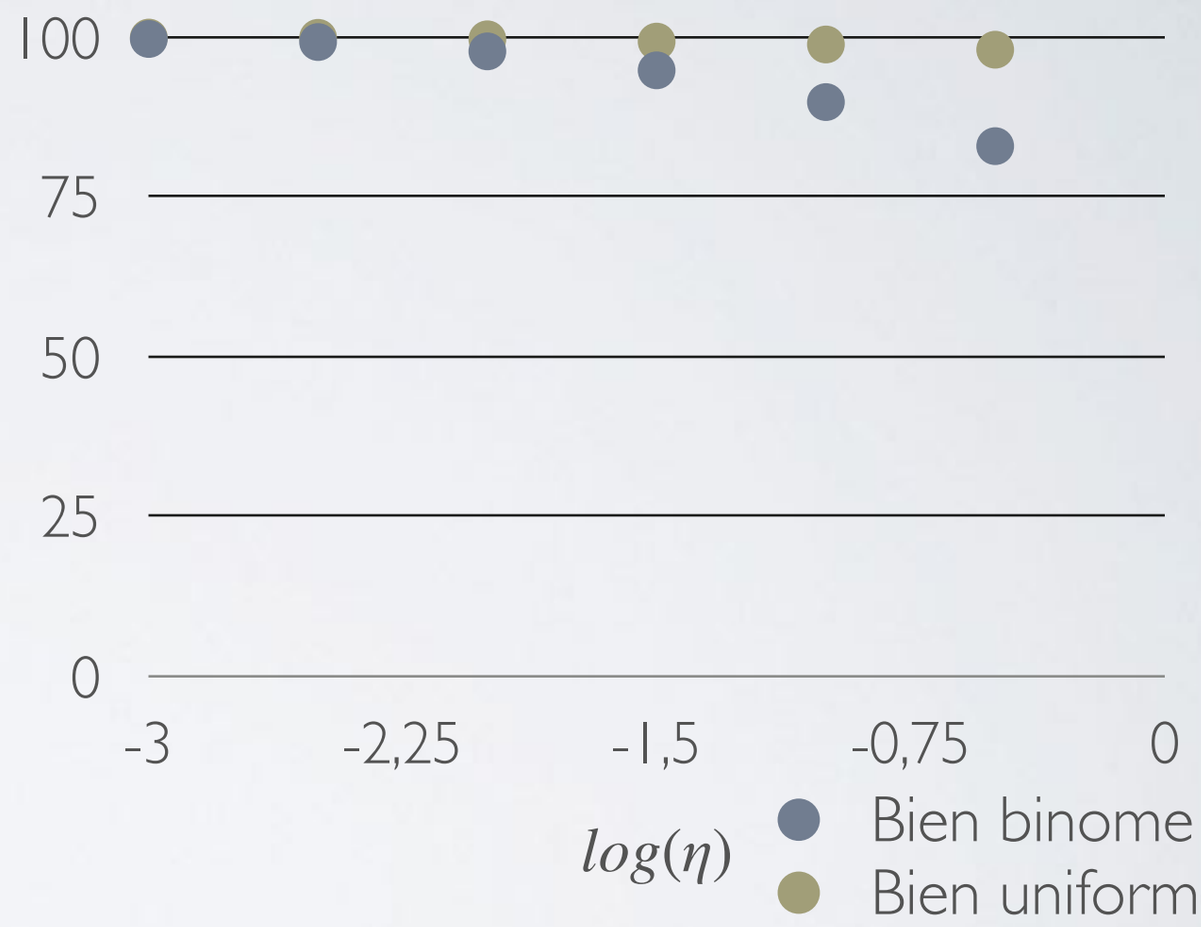
Et on souhaite: $p_{id} \geq p$ ie. $(1-\eta)^{T^2} \geq p \Rightarrow \eta \leq 1 - p^{\frac{1}{T^2}}$

3. Simulations de la solution approchée

Nombre d'étapes en fonction de eta



Taux de bonne classification en fonction de eta



MERCI POUR VOTRE ATTENTION!

ANNEXE

Algorithme

```python
from __future__ import division
from random import randint, random, shuffle
import numpy as np
import matplotlib.pyplot as plt

# Paramètres de la simulation

N = 149

# Conversion liste <-> entier

def ToInt(L) :
    return int("".join(str(e) for e in L), 2)

def ToList(y) :
    return [ int(c) for c in bin(u+y)[3:] ]

# Règles d'évolution

u, s, m = 2**N, 2**(N-1), 2**N-1

def X(y) :
    return (y*2)&m|1 if y&s else (y*2)&m

def Z(y) :
    return y//2|s if y&1 else y//2

def Rule232(y) :
    x, z = X(y), Z(y)
    return (x&y|x&z|y&z)

def Rule184(y) :
    x, z = X(y), Z(y)
    return (z&(m-y))|(y&x)
```

Algorithme

```
36  # Génération de listes
37
38  def Gen(n) :
39      L = [1]*n + [0]*(N-n)
40      shuffle(L)
41      return L
42
43  def GenD() :
44      return Gen(randint(0, N-1))
45
46  def GenP() :
47      return [ randint(0, 1) for _ in range(N) ]
48
49  def GenB() :
50      n = randint(0, m)
51      L = []
52      for i in range(N) :
53          L.append(n%2)
54          n //=2
55      return L
56
```

Algorithme

```
# Simulation

def Test(L, eta, log=False) :
    p, lg, y, c = sum(L)/N, [], ToInt(L), 0
    while y and y!=m :
        c += 1
        if random()<eta :
            y = Rule232(y)
        else :
            y = Rule184(y)
        if log :
            lg.append(y)

    return p, c, (y==0 and p<=0.5 or y!=0 and p>=0.5), lg


def Sim(nb_essais, eta, type_binom) :
    succ, cumc, log = 0, 0, []
    for i in range(nb_essais) :
        p, c, b, _ = Test([GenD,GenB][type_binom](), eta)
        if b :
            succ += 1
        cumc += c
    print("Eta = ", eta, "\t Taux de succès : ", 100*succ/nb_essais, "%  T_moy = ",
round(cumc/nb_essais))

def Draw(eta, type_binom) :
    L = [GenD,GenB][type_binom]()
    M = np.array([ ToList(y) for y in Test(L, eta, log=True)[3] ])
    plt.matshow(M, cmap="Blues")
    plt.title("p = {}".format(sum(L)/len(L)))
    plt.show()
```

Algorithme

```
print("Qb, 10000 essais :")
Sim(nb_essais=10000, eta=10**(-3.0), type_binom=True)
Sim(nb_essais=10000, eta=10**(-2.5), type_binom=True)
Sim(nb_essais=10000, eta=10**(-2.0), type_binom=True)
Sim(nb_essais=10000, eta=10**(-1.5), type_binom=True)
Sim(nb_essais=10000, eta=10**(-1.0), type_binom=True)
Sim(nb_essais=10000, eta=10**(-0.5), type_binom=True)


print("Qd, 10000 essais :")
Sim(nb_essais=10000, eta=10**(-3.0), type_binom=False)
Sim(nb_essais=10000, eta=10**(-2.5), type_binom=False)
Sim(nb_essais=10000, eta=10**(-2.0), type_binom=False)
Sim(nb_essais=10000, eta=10**(-1.5), type_binom=False)
Sim(nb_essais=10000, eta=10**(-1.0), type_binom=False)
Sim(nb_essais=10000, eta=10**(-0.5), type_binom=False)

# Test (tracé de l'évolution)

Draw(eta=0.2, type_binom=True)
```