

# Memo

## Master 2 IA

**LAURENT Thomas**

Années: 2018 - 2019

# Contents

<b>I</b>	<b>Probabilité</b>	<b>11</b>
<b>1</b>	<b>Introduction au Probabilités</b>	<b>12</b>
1.1	Mesure de probabilité . . . . .	13
1.2	Variable aléatoire . . . . .	13
1.2.1	Probabilité de manière axiomatique . . . . .	13
1.2.2	Variables aléatoire discrète . . . . .	14
1.3	Probabilités conditionnelle . . . . .	14
1.4	Autres règles . . . . .	15
1.5	Exemples . . . . .	15
1.6	Probabilité Matricielle . . . . .	16
<b>II</b>	<b>Fouille de donnée</b>	<b>17</b>
<b>2</b>	<b>Rappel</b>	<b>18</b>
2.1	Logarithmes en base 2 . . . . .	19
<b>3</b>	<b>Pré traitement des données</b>	<b>20</b>
3.1	Nettoyage des données . . . . .	21
3.1.1	Caractéristiques descriptives . . . . .	21
3.2	Normalisation . . . . .	21
<b>4</b>	<b>Classification</b>	<b>22</b>
4.1	Évaluation des classifieurs . . . . .	23
4.1.1	Matrice de confusion . . . . .	23
<b>5</b>	<b>Arbre de décision</b>	<b>24</b>
5.1	critères de sélection C4.5 . . . . .	25

5.1.1	Entropie . . . . .	25
5.1.2	Gain d'information . . . . .	27
5.1.3	Gain Ratio . . . . .	27
5.2	critères d'arrêt . . . . .	28
5.2.1	Critères d'arrêt . . . . .	28
5.2.2	critères d'arrêt: Paramètre utilisateur . . . . .	28
<b>6</b>	<b>Réseau bayésiens</b>	<b>29</b>
6.1	Classifieur bayésiens . . . . .	30
6.2	Construction et classification avec des réseaux Bayésiens . . .	31
6.2.1	Construction d'un réseau bayésien naïf . . . . .	31
6.2.2	Règle de classification bayésienne . . . . .	32
6.2.3	Règle de décision . . . . .	32
6.2.4	Observation de classe . . . . .	32
<b>7</b>	<b>Clustering</b>	<b>33</b>
7.1	Approche par le Partitionnement . . . . .	34
7.2	Approche hiérarchiques . . . . .	35
7.2.1	Exemple avec la fonction $d = \text{MIN}$ . . . . .	35
<b>8</b>	<b>ItemSet mining</b>	<b>36</b>
8.1	Itemsets . . . . .	37
8.2	Règles d'association . . . . .	37
8.3	Apriori . . . . .	38
<b>III</b>	<b>Apprentissage automatique par la pratique</b>	<b>39</b>
<b>9</b>	<b>Rappel</b>	<b>40</b>
9.1	Matrices et calculs sur les Matrices . . . . .	41
9.1.1	Addition . . . . .	41
9.1.2	Multiplication . . . . .	41
9.1.3	Transposer . . . . .	41
9.1.4	Inverse . . . . .	41
<b>10</b>	<b>Algorithms Learn a Mapping From Input to Output</b>	<b>42</b>
10.1	linear ML algorithms . . . . .	43
10.2	Supervised machine learning . . . . .	43
10.3	Unsupervised machine learning . . . . .	43

10.4	semi-supervised machine learning . . . . .	43
10.5	Overview of bias and variance . . . . .	44
<b>11</b>	<b>Overfitting and Underfitting</b>	<b>45</b>
11.1	Overfitting . . . . .	46
11.2	Underfitting . . . . .	46
<b>12</b>	<b>Model Selection</b>	<b>47</b>
12.1	Train Test Split . . . . .	48
12.2	Cross validation . . . . .	49
12.3	Leave one out . . . . .	50
12.4	Matrice de confusion, Précision, Recall, F1 . . . . .	51
<b>13</b>	<b>Linear Algorithms</b>	<b>53</b>
13.1	Régression linéaire . . . . .	54
13.2	Least squares linear regression . . . . .	56
13.3	Gradient Descent . . . . .	57
13.4	Logistic Regression . . . . .	58
13.4.1	Logistic function . . . . .	58
13.5	Linear Discriminant Analysis . . . . .	59
13.5.1	Bayesian rules . . . . .	59
<b>14</b>	<b>Non linear algorithm</b>	<b>60</b>
14.1	Classification and regression tree . . . . .	61
14.2	K moyen . . . . .	64
14.3	Support vector machines . . . . .	65
14.3.1	Margin classifier . . . . .	65
14.3.2	Soft margin classifier . . . . .	65
<b>IV</b>	<b>Outils formel</b>	<b>67</b>
<b>15</b>	<b>Logique classique des propositions</b>	<b>68</b>
15.1	Vocabulaire . . . . .	69
15.2	Propriétés de l'opérateur Modèles . . . . .	69
15.3	Ensemble de connecteurs fonctionnellement complet . . . . .	71
15.4	Preuve par induction structurelle sur un ensemble de connecteurs non fonctionnellement complet . . . . .	71
15.5	Décomposition de Shannon . . . . .	72

15.6	Arbre de Shannon, ROBDD . . . . .	72
15.6.1	Remplacement ou vérifonctionnalité . . . . .	73
15.6.2	Substitution . . . . .	73
15.7	Notion de impliquant premier . . . . .	73
15.7.1	Table de Karnaugh . . . . .	73
15.7.2	Calcule arithmétique . . . . .	74
<b>16</b>	<b>Logique classique et prédicat du premier ordre</b>	<b>75</b>
16.1	Syntaxe via les arbres . . . . .	76
16.1.1	Occurrences libre . . . . .	76
16.1.2	Occurrences liée . . . . .	76
16.1.3	Occurrences quantifié . . . . .	77
16.1.4	Vocabulaire . . . . .	77
16.2	Sémantique . . . . .	78
16.3	Formule polie . . . . .	80
16.4	Équivalences remarquables . . . . .	80
16.5	Forme Prénexe . . . . .	81
16.6	Scalénisation . . . . .	82
16.7	Forme propositionnelle . . . . .	82
<b>17</b>	<b>Calculabilité et Machine de Turing</b>	<b>83</b>
17.1	Machines de Turing . . . . .	85
17.1.1	Machine de Turing universel . . . . .	85
17.2	RE, coRE et R . . . . .	85
17.2.1	Preuve de R est incluse dans RE . . . . .	86
17.2.2	Preuve de R est incluse dans coRE . . . . .	86
17.3	Problème de l'arrêt . . . . .	87
17.4	réduction fonctionnel . . . . .	87
17.4.1	Exemple de réduction fonctionnel . . . . .	87
<b>V</b>	<b>Recherche Opérationnel</b>	<b>88</b>
<b>18</b>	<b>Introduction à la PL</b>	<b>89</b>
18.1	Modèle linéaire continu à 2 variables . . . . .	90
18.1.1	Recherche de solutions . . . . .	90
18.1.2	recherche de la solution optimal . . . . .	91

<b>19 Le simplexe</b>	<b>93</b>
19.1 Initialisation du simplexe . . . . .	94
19.2 Canonicité du modèle . . . . .	95
19.3 Solution admissible . . . . .	95
19.4 Exemple simple Premier itération . . . . .	96
19.4.1 Choix de la variable entrante . . . . .	96
19.4.2 Choix de la variable sortante . . . . .	96
19.4.3 pivotage . . . . .	97
19.4.4 Nouveau modèle . . . . .	97
19.5 Exemple simple Seconde itération . . . . .	98
19.5.1 Choix de la variable entrante . . . . .	98
19.5.2 Choix de la variable sortante . . . . .	98
19.5.3 pivotage . . . . .	98
19.5.4 Nouveau modèle . . . . .	98
19.6 Exemple simple, troisième itération . . . . .	99
19.6.1 Variable entrante et sortante . . . . .	99
19.6.2 Nouveau modèle . . . . .	99
19.7 Exemple simple, dernière itération . . . . .	100
<b>20 Simplexe à deux phases</b>	<b>101</b>
20.1 Première phase du simplexe à deux phases . . . . .	102
20.1.1 Nouveau modèle . . . . .	103
20.2 Premier phase du simplexe à deux phases, première itération .	103
20.2.1 Variable entrante et sortante . . . . .	103
20.2.2 pivotage . . . . .	103
20.2.3 Nouveau modèle . . . . .	103
20.3 Premier phase du simplexe à deux phases, seconde itération .	104
20.4 Seconde phase du simplexe à deux phases . . . . .	104
20.5 Seconde phase du simplexe à deux phases, première itération .	105
20.5.1 Variable entrante et sortante . . . . .	105
20.5.2 pivotage . . . . .	105
20.5.3 Nouveau modèle . . . . .	105
20.6 Seconde phase du simplexe à deux phases, seconde itération .	106
20.6.1 Variable entrante et sortante . . . . .	106
20.6.2 pivotage . . . . .	106
20.6.3 Nouveau modèle . . . . .	106
20.7 Seconde phase du simplexe à deux phases, troisième itération .	107

## **VI Représentation des connaissances et raisonnement**

### **108**

<b>21 Logique propositionnel</b>	<b>109</b>
21.1 Vocabulaire . . . . .	110
21.2 cohérence d'un ensemble de clauses . . . . .	110
<b>22 Introduction à la logique de description</b>	<b>111</b>
22.1 Attributive Language with Complement . . . . .	112
22.1.1 Propriétés . . . . .	112
22.2 Logique de description . . . . .	112
22.2.1 Sémantique . . . . .	112
22.2.2 Assertions . . . . .	112
22.3 TBoxes et ABoxes . . . . .	113
22.3.1 Subsumption . . . . .	113
22.3.2 Classification . . . . .	113
22.3.3 Instance checking . . . . .	114
22.3.4 Retrieval . . . . .	114
22.3.5 Equivalence of concept . . . . .	114
22.3.6 Concept satisfiability . . . . .	114
22.3.7 ABox consistency . . . . .	115
22.3.8 Réduction et consistance . . . . .	115
<b>23 Méthode des Tableau pour les ALC</b>	<b>116</b>
23.1 Pre processing . . . . .	117
23.1.1 Réécriture . . . . .	117
23.1.2 Vocabulaire . . . . .	117
23.1.3 Règles d'expansion . . . . .	118
23.2 Exemple . . . . .	119
23.3 Exemple 2 . . . . .	120
<b>24 Logique presque tout</b>	<b>121</b>
24.1 Système P . . . . .	123
24.1.1 Exemple . . . . .	124
24.2 Tolérance du Système P . . . . .	125
24.3 Stratification du système P . . . . .	125
24.4 Exemple de stratification possible . . . . .	126
24.4.1 Initialisation . . . . .	126

24.4.2	Première itération . . . . .	126
24.4.3	Seconde itération . . . . .	127
24.5	Exemple de stratification non possible . . . . .	128
24.5.1	Initialisation . . . . .	128
24.5.2	Première itération . . . . .	128
24.5.3	Seconde itération . . . . .	129
<b>25</b>	<b>Logique de description DL Lite</b>	<b>130</b>
25.1	Opérateurs . . . . .	131
25.2	Requêtes . . . . .	131
25.2.1	Grounded query . . . . .	131
25.2.2	Conjonctives Query . . . . .	131
25.3	Fermetures négatives . . . . .	132
25.4	Gestion des contraintes et MultiABox . . . . .	133
25.4.1	Expansion . . . . .	133
25.4.2	Splitting . . . . .	133
25.4.3	Selection . . . . .	134
25.4.4	Modifieurs . . . . .	134
25.4.5	Complex modifieurs . . . . .	135
25.4.6	Décision avec plusieurs ABox . . . . .	135
<b>26</b>	<b>Complexité</b>	<b>137</b>
26.1	Analyse de complexité pour $D(M1, Safe)$ . . . . .	138
26.2	Analyse de la complexité pour $D(M2, Forall)$ . . . . .	139
<b>VII</b>	<b>Théories de la Décision</b>	<b>140</b>
<b>27</b>	<b>Théorie de la décision</b>	<b>141</b>
27.1	Décision dans l'incertain . . . . .	143
27.1.1	Critère de Laplace . . . . .	143
27.1.2	Critère de Wald . . . . .	143
27.1.3	Critère d'Hurwicz . . . . .	143
27.1.4	Min Max Regret . . . . .	143
27.1.5	Exemple . . . . .	143
27.1.6	Différents cadres d'incertitude . . . . .	144



<b>28 Théorie des jeux</b>	<b>145</b>
28.1 Jeux sous forme stratégique . . . . .	146
28.1.1 utilité . . . . .	146
28.1.2 jeux sous forme extensive et stratégique . . . . .	147
28.1.3 Élimination de stratégies dominées . . . . .	148
28.1.4 Équilibre de Nash . . . . .	148
28.1.5 Critère de Pareto . . . . .	148
28.1.6 Niveau de sécurité . . . . .	149
28.1.7 autres Stratégies . . . . .	150
28.1.8 Équilibre de Nash en stratégies mixtes . . . . .	150
28.1.9 Représentation graphique du jeu . . . . .	151
28.1.10 Coopération . . . . .	152
28.1.11 Itération le dilemme des prisonniers . . . . .	152
28.1.12 DIP Itérations . . . . .	153
28.1.13 Les Stratégies . . . . .	153
28.2 Jeux répété . . . . .	155
28.2.1 Jeux à deux joueurs à somme nulle . . . . .	155
28.2.2 Jeu sous forme extensive . . . . .	156
28.2.3 sous jeu . . . . .	156
28.2.4 Menaces non crédibles . . . . .	157
28.2.5 Promesse non crédible . . . . .	158
28.2.6 Limite de la récurrence à rebours . . . . .	158
28.3 Jeux coopératifs à 2 joueurs . . . . .	159
<b>29 Décision de groupe et théorie du vote</b>	<b>160</b>
29.1 Vote entre 2 candidats . . . . .	161
29.2 Gagnant de Condorcet . . . . .	161
29.3 Scrutin majoritaire simple . . . . .	162
29.4 Scrutin majoritaire à deux tour . . . . .	162
29.5 Méthode de vote non rangées . . . . .	163
29.6 Méthode par scorage . . . . .	163
29.7 Méthode de Condorcet cohérentes . . . . .	164
29.7.1 Règle de Copeland . . . . .	164
29.7.2 Règle de Kramer Simpson . . . . .	165
29.7.3 Règle du Tile break . . . . .	165
29.8 Graphe de majorité . . . . .	166
29.9 Vote par comparaison successives . . . . .	167
29.10 Vote simple transférable: méthode de Coombs . . . . .	167

<b>VIII</b>	<b>Apprentissage</b>	<b>168</b>
<b>30</b>	<b>Approche d'apprentissage par la logique</b>	<b>169</b>
30.1	Espace de Version . . . . .	170
30.1.1	convergence des données . . . . .	171
<b>31</b>	<b>Apprentissage statistique</b>	<b>173</b>
31.1	Classification binaire réalisable . . . . .	174
31.1.1	Erreur de généralisation et d'entraînement . . . . .	174
31.1.2	Processus d'apprentissage . . . . .	174
31.1.3	Incertitude de l'apprentissage . . . . .	174
31.1.4	Modèle PAC réalisable . . . . .	175
31.2	Classes d'hypothèses finies . . . . .	175
31.2.1	Minimisation des erreurs empirique . . . . .	175
31.2.2	Théorème de PAC des classes finies . . . . .	175
31.3	Classification binaire agnostique . . . . .	176
31.3.1	Régression agnostique . . . . .	176
31.3.2	Apprentissage agnostique . . . . .	176
31.3.3	Principe de minimisation de risque empirique . . . . .	176
31.4	VC Dimension . . . . .	176
31.4.1	Fonctions linéaires . . . . .	176
31.4.2	Fonctions de croissance et VC-Dimension . . . . .	176
31.4.3	Théorème fondamental de l'apprentissage statistique . . . . .	176
31.4.4	VC-Dimensions Utiles . . . . .	176
<b>32</b>	<b>Apprentissage Online</b>	<b>177</b>
32.1	Analyse convexe . . . . .	178
32.1.1	Combinaison convexe . . . . .	178
32.1.2	Enveloppe convexe . . . . .	178
32.1.3	Theoreme de la séparation des hyperplan . . . . .	180
32.1.4	Fonction convexe et normes . . . . .	181
32.1.5	Gradient . . . . .	182
32.1.6	Fonction de Perte . . . . .	183
32.2	Apprentissage par régression . . . . .	184
32.2.1	Gradient Descent . . . . .	184
32.3	Apprentissage par classification . . . . .	185

**IX Problème de satisfaction de contraintes CSP 186****33 Introduction et modèles 187**

- 33.1 exemple simple . . . . . 188
  - 33.1.1 Graphe de contraintes . . . . . 189
  - 33.1.2 Graphe de compatibilité . . . . . 189

**34 Filtrage 190**

- 34.1 Filtrage du domaine via les contraintes . . . . . 191
  - 34.1.1 Exemple . . . . . 191
- 34.2 Notion de Support . . . . . 192
- 34.3 AC filtre AllDifferent . . . . . 193
- 34.4 AC filtre Cardinalité . . . . . 195
- 34.5 AC filtre Sum à une borne . . . . . 196
- 34.6 AC filtre avec Sum à deux bornes . . . . . 198

**X Problème de satisfaction SAT 199****35 définitions de base 200**

- 35.1 Transformation NNF, CNF . . . . . 201
  - 35.1.1 Transformation glouton . . . . . 201
  - 35.1.2 Transformation via ajout de variables . . . . . 202
- 35.2 Littéral et clause : classification . . . . . 203
  - 35.2.1 Clause active . . . . . 204
  - 35.2.2 Littéral pure . . . . . 204

**36 Classes polynomiales 205**

- 36.1 2-SAT . . . . . 206
- 36.2 Horn-SAT . . . . . 206
- 36.3 Horn-renommable . . . . . 206

**37 Algorithmes de résolution syntaxiques 207**

- 37.1 Algorithmes complets basé sur la résolution . . . . . 208
  - 37.1.1 Méthodes basé sur la résolution . . . . . 208
  - 37.1.2 Modus Ponuce . . . . . 208
  - 37.1.3 règles de subsumption . . . . . 208
  - 37.1.4 Règles de fusion . . . . . 208
- 37.2 Algorithmes complets Procédure de Davis et Putnam . . . . . 208

37.3 Algorithmes complets Méthodes des tableaux . . . . .	209
<b>38 Algorithmes de résolutions sémantique</b>	<b>210</b>
38.1 Procédure de DPLL . . . . .	210

# Part I

## Probabilité

# Chapter 1

## Introduction au Probabilités

Dans un premier temps les probabilités fréquentiste qui est plus général, un exemple simple est le lancer d'une pièce de monnaies non triqué dont on n'a prit soigneusement d'ignorer tout tombé sur la tranche de la pièce, il y a 50% de chance que la pièce tombe sur Pile ou sur Face. on dit aussi  $\frac{1}{2}$  ou .5.

Lancer une pièce est prit comme un événement:

	Pile	Face
Un lancé	.5	.5

Dans un second temps les probabilités subjectif où les variable choisit sont

indépendant d'un individu à un autre, on peut prendre "quelle est la probabilité qu'une maison s'effondre".

On construit une probabilité en répétant la même un même événement puis en notant un ensemble de résultats:

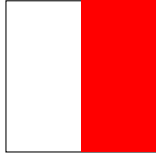
$$P(A) = \lim_{n \rightarrow +\infty} \frac{n_A}{n}$$

où  $n$  le nombre de lancé total et  $n_A$  le nombre de lancé où  $A$  est tombé en résultat

Mais en pratique  $\infty$  n'est pas faisable donc nous prenons un  $n$  un très grand nombre mais inférieur à l'infini.

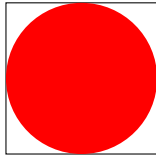
## 1.1 Mesure de probabilité

Vous devez lancer des fléchettes sur un carré, vous ne rater aucun tire (toutes les flèches arrive dans ce carré):



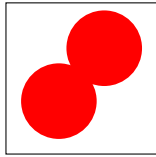
flèche touche le coté rouge.

Il y a  $\frac{1}{2}$  chance que la



flèche touche le coté rouge.

Il y a  $\frac{\pi}{4}$  chance que la



similaire aux intégrales nous allons découper la grille en pixels, les pixels rouges et blanc.  $\frac{\text{nombre de pixel rouge}}{\text{nombre de pixel total}}$

Via un raisonnement

## 1.2 Variable aléatoire

Une variable  $x$  est dite aléatoire si elle est soumise à l'incertitude (au hasard). Dans le cas d'un lancé de pièce non truqué,  $X_1 = 0$  (pile),  $X_2 = 1$  (face).

### 1.2.1 Probabilité de manière axiomatique

Une distribution de probabilité est une fonction  $P$  qui a un événement  $A$  lui associe un réel borné entre 0 et 1. La probabilité sans incertitude (celle qu'on n'est sur quelle va se produire) est égal à 1.

$$0 \leq P(A) \leq 1$$

Si deux événements  $A$  et  $B$  en exclusion mutuelle ( $\nexists x \in A \cap B$ ) on n'a:

$$P(A \cup B) = P(A) + P(B)$$

$$P(A \cap B) = P(A) + P(B) - P(A \cup B)$$

### 1.2.2 Variables aléatoire discrète

Soit  $x$  une variable discrète prennent une valeur dans un ensemble  $X$  fini. on note  $x = x \in X$  et noté  $P(x=x)$ .

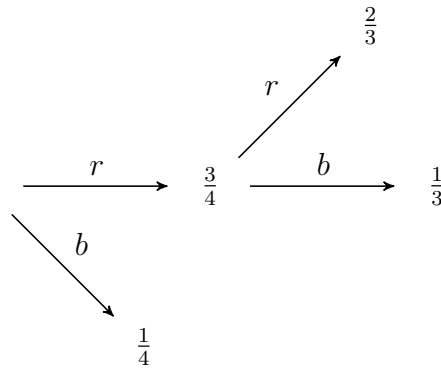
$$P(x) \geq 0$$

$$\sum_{x \in X} P(X) = 1$$

### 1.3 Probabilités conditionnelle

Prenons le tirage de 4 billes, 3 rouges et 1 bleu:  $\omega = \{r, r, r, b\}$ .

Le but est de tirer la seconde bille (sachant que la première na pas était remise) sachant que la première boule tiré est une rouge.



S'écrit comme

$$P(B2|B1 = r) = \frac{P(B1B2)}{P(B2)}$$

$P(B1B2)$  se dit probabilité Jointe.

$P(B2)$  se dit probabilité marginal.

Si  $B1$  et  $B2$  sont deux événement indépendant alors

$$P(B2|B1 = r) = P(B1, B2)$$

Dans le cas où on remet la bille tiré en jeu:

$$P(B2 = r) = P(B2 = r, B1 = r) + P(B2 = r, B1 = b)$$



## 1.4 Autres règles

Règle de bayes:

$$P(x_i|y_i) = \frac{P(y_i|x_i)*p(x_i)}{p(y_i)}$$

Règle de chainage:

$$P(x_1, x_2, x_3, \dots, x_n) = p(x_1) * p(x_2|x_1) * \dots * p(x_n|x_{n-1} \dots x_1)$$

Distribution conditionnel:

$$\forall x \in X, \forall y \in Y \Rightarrow P(x|y)$$

## 1.5 Exemples

Année	Sexe	#	%
M1	M	25	25/55
M1	F	4	4/55
M2	M	25	25/55
M2	F	1	1/55

$$P(sexe = M) = P(Sexe = MetAnnee = M1) + P(Sexe = MetAnnee = M2) = 50/55$$

$$P(Annee = M2|sexe = M) = P(Sexe = MetAnnee = M2)/P(Sexe = M) = \frac{25}{55}/\frac{50}{55} = \frac{25}{50} = \frac{1}{2}$$

A	B	P(AB)
a <sub>1</sub>	b <sub>1</sub>	.1
a <sub>2</sub>	b <sub>1</sub>	.15
a <sub>1</sub>	b <sub>2</sub>	.3
a <sub>2</sub>	b <sub>2</sub>	.45

$$\bullet P(a_1) = .40$$

$$\bullet P(a_1|b_1) = .4$$

$$\bullet P(a_1|b_2) = .4$$

$$\bullet P(a_2) = .60$$

$$\bullet P(a_2|b_1) = .6$$

$$\bullet P(a_2|b_2) = .6$$

## 1.6 Probabilité Matricielle

nous avons:

$$P(X_1 = r) = \frac{3}{4}$$

$$P(X_1 = b) = \frac{1}{4}$$

$$P(X_2 = b|X_1 = r) = \frac{1}{3}$$

$$P(X_2 = b|X_1 = b) = 0$$

On peut calculer:

$$P(X_2 = r|X_1 = r) = 1 - P(X_2 = b|X_1 = r) = 1 - \frac{1}{3} = \frac{2}{3}$$

$$P(X_2 = r|X_1 = b) = 1 - P(X_2 = b|X_1 = b) = 1 - 0 = 1$$

Une représentation matricielle pourrait être:

<i>1rouge</i>	<i>1blue</i>		<i>1rouge</i>	<i>1blue</i>		<i>2rouge</i>	<i>2blue</i>
$\frac{3}{4}$	$\frac{1}{4}$		$\frac{3}{4}$	$\frac{1}{4}$		$\frac{3}{4}$	$\frac{1}{4}$
		<i>2rouge</i>	1	0			
		<i>2blue</i>					

Le tableau de gauche indique la probabilité lors du premier tirage, le tableau à droite lors du second tirage et au milieu la probabilité jointe.

Le tableau de droite peut être obtenue en faisant le produit des deux autres.

A noté que la somme de chaque lignes horizontal doit être égal à 1.

# **Part II**

## **Fouille de donnée**

## Chapter 2

### Rappel

## 2.1 Logarithmes en base 2

$$\text{Log}_2\left(\frac{x}{y}\right) = \text{Log}_2(x) - \text{Log}_2(y)$$

$$\text{Log}_2(x * y) = \text{Log}_2(x) + \text{Log}_2(y)$$

## Chapter 3

### Pré traitement des données

## 3.1 Nettoyage des données

### 3.1.1 Caractéristiques descriptives

**Moyenne (espérance)** :  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

**Ecart moyen** :  $\frac{1}{n} \sum_{i=1}^n |x_i - \bar{x}|$

**Variance** :  $v = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$

**Ecart type** :  $\sigma_x := \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} = \sqrt{\frac{1}{n} (\sum_{i=1}^n x_i^2) - \bar{x}^2}$

**Médiane** : Valeur se trouvant au milieu de données ordonnées

**Mode** : Valeur la plus fréquente

**Amplitude** : min, max

## 3.2 Normalisation

**Min-max** :  $v_n = \frac{v - v_{min}}{v_{max} - v_{min}}$

**Min-max dans l'intervalle [A,B ]** :  $v_n = \frac{v - v_{min}}{v_{max} - v_{min}} * (B - A) + A$

**Z-Score** :  $v_n = \frac{v - moyenne}{ecart_{type}}$

**Decimal scaling** :  $v_n = \frac{v}{100^j}$

# Chapter 4

## Classification



## 4.1 Évaluation des classifieurs

### 4.1.1 Matrice de confusion

Percent of correct classification :

$$\text{PCC}(\%) := \frac{N_c}{N_t} * 100$$

$N_c$  : nombre d'instances correctement classées

$N_t$  : nombre d'instances testées ( $N_t = |D_{test}|$ )

Exemple:

-	c1	c2	c3	c4
c1	0	1	0	0
: c2	1	60	0	1
c3	0	1	23	0
c4	1	0	7	5

Taux d'erreurs : 100-PCC

$$\text{PCC}(\%) = \frac{0+60+23+5}{100} * 100 = 88\%$$

$$\text{Coût d'erreur} = \sum_1^n \text{cout}(\text{class}_{\text{reelle}}, \text{classe}_{\text{predite}})$$

$$\text{coût d'erreur moyen} = \frac{\text{coutderreur}}{N_{\text{erreurs}}}$$

$$\text{Rappel}(C_i) = \frac{N_{c-i}}{N_{t-i}} * 100 \quad (\text{Horizontal}) \quad \text{Ex : } \text{Rappel}(C_3) = (23/24)\%$$

$$\text{Precision}(C_i) = \frac{N_{c-i}}{N_i} * 100 \quad (\text{Vertical}) \quad \text{Ex : } \text{Precision}(C_3) = (23/30)\%$$

# Chapter 5

## Arbre de décision

## 5.1 critères de sélection C4.5

Construction d'un arbre de décision C4.5 La construction d'un arbre de décision avec C4.5 passe par deux phases:

**Phase d'expansion** : La construction se fait selon l'approche descendante et laisse croître l'arbre jusqu'à sa taille maximale.

**Phase d'élagage** : Pour optimiser la taille l'arbre et son pouvoir de généralisation, C4.5 procède à l'élagage (pour supprimer les sous-arbres qui ne minimisent pas le taux d'erreurs)

**Approche de construction d'un AD** : Partitionner récursivement les données en sous-ensembles plus homogènes ... jusqu'à obtenir des partitions qui contiennent des objets qui appartiennent majoritairement à la même classe.

=> Théorie de l'information pour caractériser le degré de mélange, homogénéité, impureté, incertitude...

**Théorie de l'information** : Théorie mathématique ayant pour objet l'étude du contenu informationnel d'un message.

Applications en codage, compression, sécurité...

**Entropie** : Mesure la quantité d'incertitude dans une distribution de probabilités.

### 5.1.1 Entropie

**Entropie** : Mesure la quantité d'incertitude (manque d'information) dans une distribution de probabilités. Soit X une variable aléatoire discrète prenant ses valeurs dans  $DX = x_1, \dots, x_n$ . Soit P la distribution de probabilités associée à X.

$$H(X) = - \sum_{i=1}^n p(x_i) * \log_2(p(x_i))$$

Par convention, quand  $p(x) = 0, 0 * \log(0) = 0$

Exemple:

X	P(X)
x_1	1/3
x_2	1/3
x_3	1/3

$$H(X) = -p(x_1) * \log_2(p(x_1)) - p(x_2) * \log_2(p(x_2)) - p(x_3) * \log_2(p(x_3))$$

$$H(X) = -3(\frac{1}{3} * \log_2(\frac{1}{3})) = \log_2(3) = 1.58$$

Autre exemples:

$$[\frac{1}{2}, \frac{1}{4}, \frac{1}{4}] : H(X) = 1.5$$

$$[1, 0, 0] : H(X) = 0$$

$$[\frac{1}{2}, \frac{1}{2}] : H(X) = 1$$

Propriétés:

$$H(X) \geq 0$$

$H(X)$  est maximale pour une distribution uniforme (toutes les valeurs sont équiprobables).

**Entropie conjointe** : L'entropie conjointe de deux variables aléatoires X et Y est l'incertitude relative à ces deux variables conjointement.

$$Entropie(X, Y) = - \sum_{i,j=1}^n p(x_i, y_j) * \log_2(p(x_i, y_j))$$

**Exemple** :  $[0.2, 0.1, 0.3, 0.4] : H(X, Y) = 1.85$

### 5.1.2 Gain d'information

Soit le data suivant, avec ClientSatisfait la variable de classe:

Mémoire	AutonomieBatterie	Prix	ClientSatisfait
<= 4	longue	<= 150	Oui
> 4	longue	> 150	Oui
> 4	longue	<= 150	Oui
<= 4	longue	> 150	Oui
> 4	longue	> 150	Oui
> 4	courte	> 150	Oui
<= 4	courte	> 150	Non
<= 4	courte	> 150	Non
> 4	courte	<= 150	Oui
<= 4	courte	<= 150	Non
<= 4	moyen	<= 150	Non
> 4	moyen	<= 150	Non
<= 4	moyen	> 150	Oui
> 4	moyen	> 150	Oui
> 4	moyen	<= 150	Non

Le *Gain information* appliqué sur la colonne AutonomieBatterie (AB) serait:

$$Gain(AB) = Entropie(AB) - \frac{5}{15} Entropie(Longue) - \frac{5}{15} Entropie(Courte) - \frac{5}{15} Entropie(Moyen)$$

$$Entropie(AB) = -3(\frac{5}{15} * \log_2(\frac{5}{15}))$$

$$Entropie(Longue) = 0$$

$$Entropie(Courte) = \frac{2}{5} * \log_2(\frac{2}{5}) - \frac{3}{5} \log_2(\frac{3}{5})$$

$$Entropie(Moyen) = \frac{3}{5} \log_2(\frac{3}{5}) - \frac{2}{5} * \log_2(\frac{2}{5})$$

### 5.1.3 Gain Ratio

$$Gainratio(AB) = \frac{Gain(AB)}{Entropie(AB)}$$

## **5.2 critères d'arrêt**

### **5.2.1 Critères d'arrêt**

Si tout les objets d'une partition appartiennent à une même classes

Si il n'y a plus aucun attributs à tester

si le nœud est vide (càd feuille de l'arbre)

Absence d'apport informationnel (le gain est négatif ou nul)

### **5.2.2 critères d'arrêt: Paramètre utilisateur**

Nombre d'objets minimum par feuille

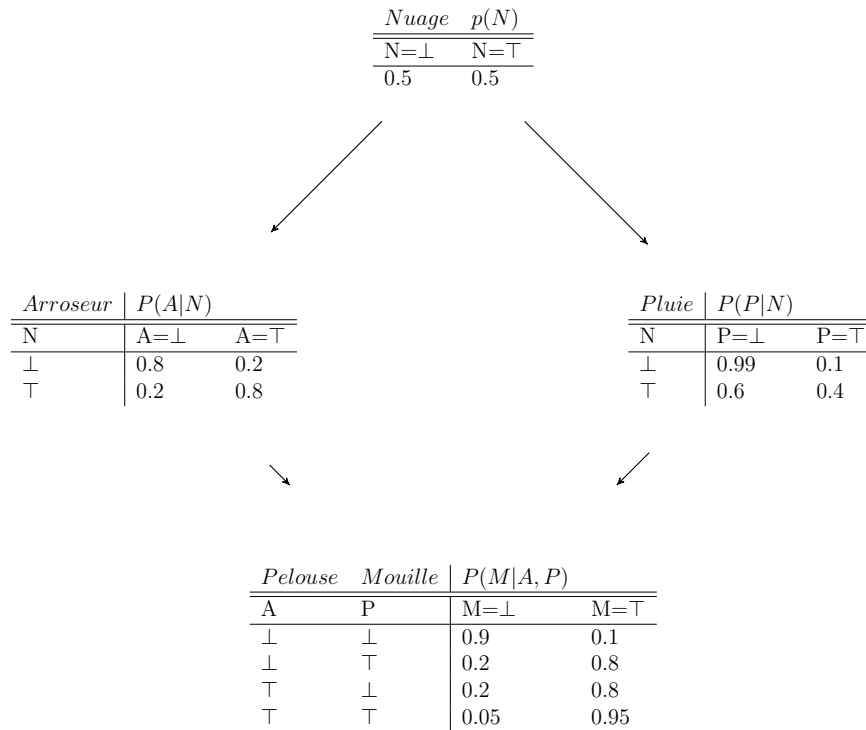
Taille, profondeur de l'arbre

Temps de construction de l'arbre

## Chapter 6

### Réseau bayésiens

## 6.1 Classifieur bayésiens



**Calculer**  $P(N = \top, P = \top, A = \perp, M = \top)$

$$= P(N = \top) * P(P = \top | N = \top) * P(A = \perp | N = \top, P = \top) * P(M = \top | N = \top, P = \top, A = \perp)$$

$$= .5 * .4 * \frac{P(N=\top, P=\top)P(A=\perp)}{P(N=\top, P=\top)} * \frac{P(N=\top, P=\top, A=\perp)P(M=\top)}{P(N=\top, P=\top, A=\perp)}$$

$$= .5 * .4 * 1 *$$



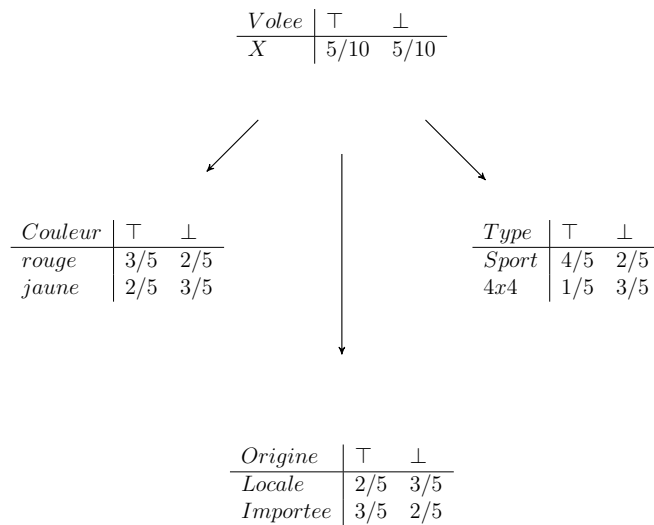
## 6.2 Construction et classification avec des réseaux Bayésiens

Soit le jeu de donnée suivant:

	Couleur	Type	Origine	volée
1	rouge	sport	locale	oui
2	rouge	sport	locale	non
3	rouge	sport	locale	oui
4	jaune	sport	locale	non
5	jaune	sport	importée	oui
6	jaune	4x4	importée	non
7	jaune	4x4	importée	oui
8	jaune	4x4	locale	non
9	rouge	4x4	importée	non
10	rouge	sport	importée	oui

### 6.2.1 Construction d'un réseau bayésien naïf

soit la variable de classe nommé "Volée":



### 6.2.2 Règle de classification bayésienne

$$classes = \max \begin{cases} P(Volee = \top | Rouge, 4x4, Importee) \\ P(Volee = \perp | Rouge, 4x4, Importee) \end{cases}$$

### 6.2.3 Règle de décision

$$\begin{aligned} P(V|CTO) &= P(VCTO) \text{ car indépendantes} \\ &= P(C|v) * P(T|V) * P(O|V) * P(V) \end{aligned}$$

### 6.2.4 Observation de classe

Avec l'observation suivante (Rouge, 4x4, Importée) la classe associée à cette observation est:

$$\begin{aligned} P(Volee = Non, Rouge, 4x4, Importee) &= P(Rouge|Non) * P(4x4|Non) * \\ &P(Importee|Non) * P(Non) \\ &= 2/5 * 3/5 * 2/5 * 1/2 \\ P(Volee = Oui, Rouge, 4x4, Importee) &= P(Rouge|Oui) * P(4x4|Oui) * \\ &P(Importee|Oui) * P(Oui) \\ &= \end{aligned}$$

Avec l'observation incomplète suivante (Jaune, Sport) la classe associée à cette observation est:

$$\begin{aligned} P(Volee = Non, Jaune, Sport) &= P(Jaune|Non) * P(Sport|Non) * \sum P(\theta|Non) * \\ &P(Non) \\ &= 2/5 * 4/5 * 1 * 1/2 \\ P(Volee = Oui, Jaune, Sport) &= P(Jaune|Oui) * P(Sport|Oui) * \sum P(\theta|Oui) * \\ &P(Oui) \\ &= \end{aligned}$$

# Chapter 7

## Clustering

## 7.1 Approche par le Partitionnement

Soit

**une table à segmenter**  $T = 2, 4, 6, 7, 8, 11, 13$

**une fonction de distance**  $d() = \text{Distance euclidienne}$

**k** = 3

**3 clusters au hasard**  $C_1 = 2, C_2 = 4, C_3 = 6$

Pour chaque cluster  $C_i$ , initialiser  $C_i^{center}$  à la moyenne de tout les élément de  $C_i$ .

Pour chaque éléments hors cluster calculer la distance  $D()$ , entre tout les  $C_i^{center}$  et l'élément courant, puis placer cette élément dans le  $C_i$  ayant le résultat le plus petit.

Puis recommencer tant qu'il existe pas une redondance.

## 7.2 Approche hiérarchiques

**Initialisation** Au départ, chaque objet forme un cluster.

**Refaire** Regrouper la paire de cluster les plus proche selon  $D()$  et mettre à jour la matrice de similarité.

**Cas d'arrêt** il ne reste plus qu'un cluster ou le nombre  $k$  de cluster est atteint.

La mesure de la similarité se fait via la fonction de comparaison  $D()$  qui peut par exemple être le MIN, MAX, Centre du groupe, Moyenne du groupe,...

### 7.2.1 Exemple avec la fonction $d = \text{MIN}$

Soit la matrice de similarité ci dessous, avec la condition distance d'arrêt inférieur ou égal à 4.

On commence par trouve l'indice le plus petit pour en suite fusionner:

(Avec  $d(P3, \{P1, P2\}) = \min(d(P3, P1), d(P3, P2)) = \min(7, 5) = 5$

	P1	P2	P3	P4		{P1,P2}	P3	P4		{P1,P2,P4}	P3
P1	0				{P1,P2}	0			{P1,P2,P4}	0	
P2	1	0			P3	5	0		P3	5	0
P3	7	5	0		P4	2	6	0			
P4	2	3	6	0							

# Chapter 8

## ItemSet mining

## 8.1 Itemsets

**Support(D)** Le nombre de fois où D est un sous ensemble de l'itemset.

**Couverture(D)** Les indices de lignes où une D est un sous ensemble de l'itemset.

**Fréquence(D)** Le support divisé par le nombre total d'itemset.

	itemsets
1	{A,B,C,D}
2	{A,B,C}
3	{C,D}
4	{C,E,A}

**Support(A)** 3

**Support(A,C)** 3

**Couverture(D)** {1,3}

**Fréquence(C)**  $\frac{4}{4}$

## 8.2 Règles d'association

**Support(X=>Y)** Le nombre de fois où  $X \cup Y$  est un sous ensemble de l'itemset.

	itemsets
1	{A,B,C,D}
2	{A,B,C}
3	{C,D}
4	{C,E,A}

**Support(A=>B)** 2

**Support(AC=>E)** 1

### 8.3 Apriori

Soit le tableau suivant, Calculer IF (avec une marge minimum de 2):

	itemsets
1	{A,B,C,D}
2	{A,B,C}
3	{C,D}
4	{C,E,A}

$$I_1 \{ A=3, B=2, C=4, D=2, E=1 \}$$

$$F_1 \{ A, B, C, D \}$$

$$C_2 \{ AB=2, AC=3, AD=1, BC=2, BD=1, CD=2 \}$$

$$F_2 \{ AB, AC, BC, CD \}$$

$$C_3 \{ ABC=2, ABD=1, ACD=1 \}$$

$$F_3 \{ ABC \}$$

$$IF \{ A, B, C, D, AB, AC, BC, CD, ABC \}$$



## Part III

# Apprentissage automatique par la pratique

## Chapter 9

### Rappel

## 9.1 Matrices et calculs sur les Matrices

### 9.1.1 Addition

$$\begin{pmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{pmatrix}$$

### 9.1.2 Multiplication

$$\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$
$$(1 * 5) + (2 * 7) = 19$$

### 9.1.3 Transposer

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

### 9.1.4 Inverse

Soit une matrice 2x2 comme :  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Soit Determinant  $D = ad - bc$

Si  $D \neq 0$  alors il existe une matrice inverse égal à :  $\frac{1}{D} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

## Chapter 10

# Algorithms Learn a Mapping From Input to Output

## 10.1 linear ML algorithms

Simplifier les processus d'apprentissage et réduire la fonction sur ce qu'on connaît

**Soit** :  $B_0 + B_1X_1 + B_2X_2 + B_3X_3 = 0$

Où  $B_0, B_1, B_2, B_3$  sont les coefficients présent sur l'axe des ordonnées.

Et  $X_1, X_2, X_3$  sont les valeurs en Input.

## 10.2 Supervised machine learning

L'apprentissage supervisé peut se diviser en 2 partis

**Classification** : Quand les variables en sortie sont des Classe (*Vert, Carre, Homme*)

**Regression** : Quand les variables en sortie sont des valeur numérique (*euro, poids, quantites*)

## 10.3 Unsupervised machine learning

Les problèmes de l'apprentissage non supervisé sont:

**Clustering** : L'art de faire des paquet d'éléments qui ont des points commun, comme regrouper les clients par paquet de choses qu'ils ont le plus en commun.

**Association** : Associer des règles d'apprentissage pour décrire une portion du data, comme une personne qui a acheté un item A et qui est aussi tenté par acheter un item B

## 10.4 semi-supervised machine leaning

L'apprentissage semi supervisé c'est avoir un bonne quantité de données en input X, et un peu de data avec le label Y.

## 10.5 Overview of bias and variance

La prédiction des erreurs pour les algorithmes sont regroupé en 3 points:

**Bias Error** : Simplifier l'hypothèse fait par le modèle pour faire une fonction d'apprentissage plus facile.

**Variance Error** : Et la quantité estimée par la fonction visée qui changera via un différent ensemble de data utilisé.

**Irreducible Error** : Ne peut pas être réduit

## Chapter 11

# Overfitting and Underfitting

## 11.1 Overfitting

L'overfitting intervient lorsque le modèle sur apprend des connaissances, Lorsque l'on sur apprend nous prenons en compte les points plus éloigné de la droite de la fonction.

On peut illustrer l'overfitting en codant un algorithme qui prend en compte les points bleu et rouges de la figure *ap-linear-regression\_1* ce dessous.

## 11.2 Underfitting

C'est l'inverse de l'overfitting, pas assez de données pour pouvoir généraliser le base de connaissance.



# Chapter 12

## Model Selection

## 12.1 Train Test Split

S'applique à de très gros dataset.

Sépare les listes *xset* et *yset* en *train*, *test* sous liste.

Les ensemble de retours *xtrain*, *ytrain* et *xtest*, *ytest* ont le même nombres de lignes et la taille.

La taille des ensembles *test* sont une proportion de la taille du *set* multiplié par la paramètre *test\_size*.

```
1 from sklearn.model_selection import train_test_split
2
3 xtrain, xtest, ytrain, ytest = train_test_split(xset, yset, test_size=0.1, random_state=0)
```

[\*sklearn.model\\_selection.train\\_test\\_split\*](#)

---

### Paramètres

**xset,yset** Souvent de type [\*pandas.DataFrame\*](#).

**test\_size** *float btw 0 & 1* le nombre de rows que *xtest*, *ytest* contiendra en proportion de la taille des entrées.

**random\_state** *Integer* la graine utilisé pour les générateurs de nombre aléatoire.

**shuffle** *Boolean* Mélanger ou pas les sets avant la séparation.

---

### Retourné

**arrays**

## 12.2 Cross validation

S'applique à un jeu de donné de taille moyenne.

La séparation d'un jeu de donnée d'entraînement et de test peuvent donner par hasard des jeux de données non représentatifs.

Pour éviter ce cas, il est nécessaire de reproduire plusieurs fois la procédure puis de moyennner les résultats retournée.

Chaque étape de la cross validation va retournée 2 ensemble (respectivement égaux au indices de *train*, *test*):

```
1 from sklearn.model_selection import KFold
2
3 kf = KFold(n_splits=10, shuffle=True)
4 for trainI, testI in kf.split(xset):
5     xtrain, xtest = xset[trainI], xset[testI]
6     ytrain, ytest = yset[trainI], yset[testI]
```

Exemple simple d'un instance *KFold*(*n\_split* = 3, *shuffle* = *False*) sur un dataSet de taille 15.

Les éléments en rouge seront les éléments sélectionné dans les ensembles de *test* et les éléments en noir seront les *train*:

k=1 A,B,C,D,E,F,G,H,I,J,K,L,M,N,O

k=2 A,B,C,D,E,F,G,H,I,J,K,L,M,N,O

k=3 A,B,C,D,E,F,G,H,I,J,K,L,M,N,O

*sklearn.model\_selection.KFold*

---

### Paramètres

**n\_split** *Integer* Nombre de split à effectuer

**shuffle** *Boolean* Mélanger ou pas les sets avant la séparation.

---

### Retourné

arrays

## 12.3 Leave one out

S'applique à des dataset de petite taille.

Pour chaque item du dataset, le prendre en tant que *test* et le reste en tant que *train*.

```
1 from sklearn.model_selection import LeaveOneOut
2 loo = LeaveOneOut()
3
4 for train_index, test_index in loo.split(X):
5     X_train, X_test = X[train_index], X[test_index]
6     y_train, y_test = y[train_index], y[test_index]
```

## 12.4 Matrice de confusion, Précision, Recall, F1

Tout ces paramètres indique la consistance de la dataSet, ils sont calculé via une matrice de confusion:

```
1 from sklearn.metrics import confusion_matrix
2 print(confusion_matrix(ytrain, ypredicted))
3 >> array([[tn, fp],
4          [fn, tp]])
5
6 tn, fp, fn, tp = confusion_matrix(ytrain,ypredicted).ravel()
```

[\*sklearn.metrics.confusion\\_matrix\*](#)

---

### Paramètres

**y\_true** *array* les y valides.

**y\_pred** *array* les y qui ont était prédit via un classifieur.

---

### Retourné

**arrays**

---

### Méthodes

**ravel()** *arrays* retourne les index dans l'ordre de leurs position:

**tn** les vrai négatifs

**fp** les faux positifs

**fn** les faux négatifs

**tp** les vrai positifs

Les Précision, Recall, F1 peuvent être calculé depuis le tableau de sortie qu'offre *confusion\_matrix*, mais il existe des méthodes permettant de le faire à notre place:

```
1 from sklearn.metrics import precision_recall_fscore_support
2
3 prf = precision_recall_fscore_support(ytest, ypredicted)
4 print(zip(["Precision", "Recal", "F1", "Support"], [numpy.mean(row) for row in prf]))
5 {"Precision": -, "Recal": -, "F1": -, "Support": -}
```

*sklearn.metrics.precision\_recall\_fscore\_support*

---

### Paramètres

**y\_true** *array* les y valides.

**y\_pred** *array* les y qui ont été prédit via un classifieur.

---

### Retourné

**arrays**

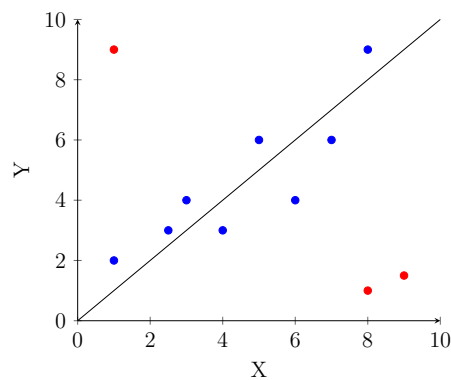
## Chapter 13

### Linear Algorithms

Soit  $X$  l'ensemble des variables indépendantes sur l'axe des l'abscisse et  $Y$  l'ensemble des variable dépendantes sur l'axe des ordonnée.

## 13.1 Régression linéaire

Étant donné un plan à deux dimensions où l'abscisse contient les point d'entrée  $X$  et l'ordonnée contient les points de sortie  $Y$ , et un nuage de points précédaient acquitté de tout point éloigné du nuage.



Avec :  $y = \beta_0 + \beta_1 x$

Pour un hyperPlan (3d) :  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$P - I_n$  :  $y = \beta_0 + \beta_1 x_1 + \dots \beta_n x_n$

```
1 from sklearn.linear_model import LinearRegression
2
3 reg = LinearRegression().fit(xtrain, ytrain)
4 reg.score(xtest, ytest)
5 reg.predict(xtest)
```

[`sklearn.linear\_model.LinearRegression`](#)

### Méthodes

**fit(X,y)** *pandas.DataFrame* Apprend le modèle avec les data  $X$  et  $y$ .

**predict(X)** *pandas.DataFrame* Test l'apprentissage avec les données  $X$  et retourne le  $y$  généré.



**score(**X**,**y**)** *pandas.DataFrame* Retourne le coefficient de prédiction en comparent les  $\hat{y}$  généré avec le  $y$  en paramètre.

## 13.2 Least squares linear regression

Calculer la régression linéaire avec la méthode Least squares:

Soit:

$\mathbf{X} = [1, 2, 3, 4, 5]$  les variables indépendantes d'axe abscisse

$\mathbf{Y} = [2, 4, 5, 4, 5]$  les variables dépendantes d'axe ordonnée

Calculons  $y = \beta_0 + \beta_1 x$

Calcule de la moyenne de X et Y:

$$\mathbf{Xm} = \sum x_i \in X = 3$$

$$\mathbf{Ym} = \sum y_i \in Y = 4$$

Toutes ligne de régression doivent passer par le point  $(\mathbf{Xm}, \mathbf{Ym})$ .

Calculer tout les écarts des  $x_i \in X$  par rapport à  $\mathbf{Xm}$  (resp  $\mathbf{Y}$ ):

X	Y	$X - Xm$	$Y - Ym$	$(X - Xm)^2$	$(X - Xm)(Y - Ym)$
1	2	-2	-2	4	4
2	4	-1	0	1	0
3	5	0	1	0	0
4	4	1	0	1	0
5	5	2	1	4	2

Calculer  $\beta_1$  :

$$\beta_1 = \frac{\sum (X - Xm)(Y - Ym)}{\sum (X - Xm)^2} = \frac{6}{10} = .6$$

$$\beta_0 : Ym = \beta_0 + \beta_1 * Xm : 4 = \beta_0 + .6 * 3 : 4 = \beta_0 + 1.8 : \beta_0 = 2.2$$

### 13.3 Gradient Descent

Soit:

$$\mathbf{X} = [1, 2, 4, 3, 5]$$

$$\mathbf{Y} = [1, 3, 3, 2, 5]$$

$i$  = une variable qui itère les éléments de  $X$  et  $Y$  en bouclant à l'infini.

Une initialisation comme:

$$\beta_0 = 0$$

$$\beta_1 = 0$$

$\alpha$  = donnée en énoncé (pour l'exemple égal à 0.01)

Et des fonctions définit tel que:

$$\mathbf{error} = (\beta_0 + \beta_1 * X[i]) - Y[i]$$

$$\beta_{0+1} = \beta_0 - \alpha * error$$

$$\beta_{1+1} = \beta_1 - \alpha * error * X[i]$$

En appliquant l'algorithme des calculs des  $\beta_i$ :

$i$	$X[i]$	$Y[i]$	$error$	$\beta_0$	$\beta_1$
0	1	1	-1	0.01	0.01
1	2	3	-2.97	0.06	0.03
2	4	3	-1.77	0.18	0.06
3	3	2	-1.61	0.22	0.08
4	5	5	-4.35	0.44	0.12
0	1	1	-0.42	0.45	0.13
1	2	3	-2.28	0.49	0.49

## 13.4 Logistic Regression

### 13.4.1 Logistic function

Soit:

$$t \in \mathbb{R}[0, 1] \text{ égal à } \beta_0 + \beta_2 * x$$

La fonction de logique de régression, les valeur d'entrée  $X$  sont combiné en utilisant les coefficient de valeur pour prédire une sortie  $Y$ . Cette sortie sera une valeur binaire.

$$p(x) = \frac{1}{1+e^{-(P-I_n)}}$$

**Note :**  $p(x)$  peut être interprété comme une fonction de probabilité  $P(X) = P[Y = 1|X]$ .

$$\beta_0 + \beta_1 * x = \ln\left(\frac{P(x)}{1-P(x)}\right) \text{ aussi appelé odds.}$$

```
1 from sklearn.linear_model import LogisticRegression
2
3 c = LogisticRegression().fit(xtrain,ytrain)
4 c.predict(xtest)
5 c.score(xtest, ytest)
```

[\*sklearn.linear\\_model.LogisticRegression\*](#)

---

### Méthodes

**fit(X,y)** *pandas.DataFrame* Apprend le modèle avec les data  $X$  et  $y$ .

**predict(X)** *pandas.DataFrame* Test l'apprentissage avec les données  $X$  et retourne le  $y$  généré.

**score(X,y)** *pandas.DataFrame* Retourne le coefficient de prédiction en comparent les  $y$  généré avec le  $y$  en paramètre.

## 13.5 Linear Discriminant Analysis

L'analyse discriminante linéaire fait partie des techniques d'analyse discriminante prédictive, il s'agit de prédire l'appartenance d'un individu à une classe prédéfinie à partir de ses caractéristiques mesurées à l'aide de variables prédictives.

A notre disposition, un échantillon de  $n$  observations réparties dans  $k$  groupes d'effectifs  $n_k$ .

**Noté**  $Y$  les variables prédire  $\{y_1, \dots, y_k\}$

$J$  variables prédictives  $X = (X_1, \dots, X_J)$

$\mu_k$  la moyenne (ou *mean* en anglais) valant  $\frac{\sum_{i=1}^n x_i}{n}$

$\sigma^2$  la variance de toutes les classes  $\frac{\sum_{i=1}^n (x_i - \mu_k)^2}{n - k}$

**la fonction discriminante pour la classe  $k$  avec  $x$  donné**  $D_k(x) = x * \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \ln(P(k))$

Où  $P(k)$  vaut la probabilité appliqué aux valeurs de  $Y$

### 13.5.1 bayésien rules

L'objectif est de produire une règle d'affectation  $X(\omega) \rightarrow Y(\omega)$  qui permet de prédire, pour une observation  $\omega$  donné, sa valeur associé de  $Y$  à partir des valeurs prises par  $X$ . via une probabilité

$$P(Y = y_k) = \frac{P(Y=y_k) * P(X|Y=y_k)}{\sum_{i=1}^k P(Y=y_i) * P(X|Y=y_i)}$$

Où  $P(Y = y_k)$  est la probabilité à *priori* d'appartenance à une classe

$P(X|Y = y_k)$  représente la fonction de densité des  $X$  conditionnellement à la classe  $y_k$

## Chapter 14

### Non linear algorithm

## 14.1 Classification and régression tree

Soit:

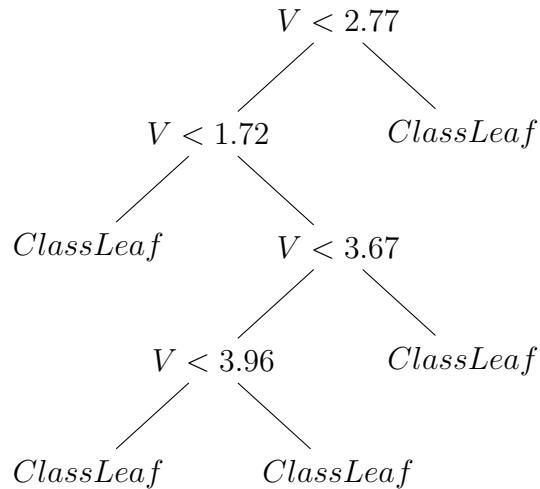
$$G = \sum_{k=1}^n p_k * (1 - p_k)$$

$$V = 2.67$$

$X_1$	$X_2$	$Y$
2.77	2.33	0
1.72	01.78	0
3.67	03.36	0
3.96	4.67	0

Soit un arbre de décision ayant comme fils gauche des *Yes* et fils droit des *No* par rapport à la condition *split*.

Si la valeur  $V < X1_i$  alors on crée un fils gauche, sinon on crée un fils droit:



Soit d'une façon plus calculatoire:

$G =$

$$\begin{array}{ll} \text{left}(X1_1) * (1 - \text{left}(X1_1)) + & X1_1 = 2.77 \\ \text{right}(X1_1) * (1 - \text{right}(X1_1)) + & = 0 \text{ car } V < 2.77 \rightarrow \text{Left} \\ \text{left}(X1_2) * (1 - \text{left}(X1_2)) + & = 0 \text{ car } 1.72 < V \rightarrow \text{Right} \\ \text{right}(X1_2) * (1 - \text{right}(X1_2)) + & X1_2 = 1.72 \\ \text{left}(X1_3) * (1 - \text{left}(X1_3)) + & X1_1 = 3.67 \\ \text{right}(X1_3) * (1 - \text{right}(X1_3)) + & = 0 \text{ car } V < 3.67 \rightarrow \text{Left} \\ \text{left}(X1_4) * (1 - \text{left}(X1_4)) + & X1_1 = 3.96 \\ \text{right}(X1_4) * (1 - \text{right}(X1_4)) + & = 0 \text{ car } V < 3.96 \rightarrow \text{Left} \end{array}$$



```
1 from sklearn.tree import DecisionTreeRegressor
2
3 c = DecisionTreeRegressor().fit(xtrain,ytrain)
4 c.predict(xtest)
5 c.score(xtest, ytest)
```

*sklearn.tree.DecisionTreeRegressor*

---

## Méthodes

**fit(X,y)** *pandas.DataFrame* Apprend le modèle avec les data  $X$  et  $y$ .

**predict(X)** *pandas.DataFrame* Test l'apprentissage avec les données  $X$  et retourne le  $y$  généré.

**score(X,y)** *pandas.DataFrame* Retourne le coefficient de prédiction en comparant les  $y$  généré avec le  $y$  en paramètre.

## 14.2 K moyen

Le K moyen demande une heuristique de type métrique pour comparé les distances entre points.

Par exemple:

**Distance euclidienne**  $\sqrt{\sum_{i=1}^n (a_i, b_i)^2}$

```
1 from sklearn.neighbors import KNeighborsClassifier
2
3 c = KNeighborsClassifier(n_neighbors=2).fit(xtrain,ytrain)
4 c.predict(xtest)
5 c.score(xtest, ytest)
```

[\*sklearn.neighbors.KNeighborsClassifier\*](#)

---

### Paramètres

**n\_neighbors** *Integer* le nombre de clusters

---

### Méthodes

**fit(X,y)** *pandas.DataFrame* Apprend le modèle avec les data  $X$  et  $y$ .

**predict(X)** *pandas.DataFrame* Test l'apprentissage avec les données  $X$  et retourne le  $y$  généré.

**score(X,y)** *pandas.DataFrame* Retourne le coefficient de prédiction en comparent les  $y$  généré avec le  $y$  en paramètre.

## 14.3 Support vector machines

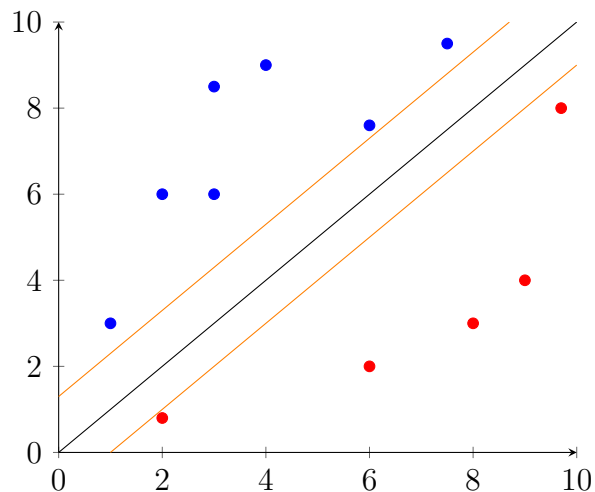
### 14.3.1 Margin classifier

Soit les points:

*Blue* Une *ClassA*

*Rouge* Une *ClassB*

Le support vector machines cherche un hyperplan (de couleur noir) pouvant départager les deux classes, Il en existe une infinité d'hyperplan qui peuvent les départager, donc introduisons un autre concept, celui de l'hyperplan qui maximise la séparation entre les deux classes (les droites *Oranges* appelé *Margin*).



### 14.3.2 Soft margin classifier

Dans le cadre du Soft margin, il n'existe pas de margin séparent les deux classes, il faut donc chercher la droite qui minimise l'erreur. Soit un ensemble de données divisé en trois parties:

**Training Set** sont les données qui seront utiliser pour l'apprentissage

**Test Set** les données qui sont utiliser pour vérifier la satisfesabilité de l'algorithme

**Tunning Set** appeler  $C$  qui sera le taux de violation de la margin accepté

Soit  $C = \{0.1, 1, 10\}$  les longueurs que peut prendre la margin et:

	<i>longueur de la margin</i>	<i>F1 Score</i>
$C_0$	0.1	80%
$C_1$	1	85 % La meilleur borne
$C_1$	10	85 %

```
1 from sklearn.svm import SVC
2
3 c = SVC().fit(xtrain,ytrain)
4 c.predict(xtest)
5 c.score(xtest, ytest)
```

[\*sklearn.svm.SVC\*](#)

---

## Méthodes

**fit(X,y)** *pandas.DataFrame* Apprend le modèle avec les data  $X$  et  $y$ .

**predict(X)** *pandas.DataFrame* Test l'apprentissage avec les données  $X$  et retourne le  $y$  généré.

**score(X,y)** *pandas.DataFrame* Retourne le coefficient de prédiction en comparent les  $y$  généré avec le  $y$  en paramètre.

# **Part IV**

## **Outils formel**

## Chapter 15

### Logique classique des propositions

## 15.1 Vocabulaire

**Déduction**  $\models \alpha$  ssi  $\neg\alpha$  est contradictoire

**Absurde**  $\phi$  est contradictoire ssi  $\neg\phi$  est valide

**DAG** : Un graphe dirigé acyclique

**Taille(Arbre)** =  $\{\text{toutes les symboles} + \text{connecteurs}\}$

**Var(Arbre)** =  $\{\text{Toutes les feuilles}\}$

**Sous formules(Arbres)** =  $\{T + \cup_{i=0}^k \text{SousFormules}(\text{Arbre}_i)\}$

**Interprétation** :  $\omega$  de  $PROP_{ps}$  est une application de PS dans 0.1

**Sémantique** :  $\|\phi\|(\omega)$  d'une formule  $\phi$  de  $PROP_{ps}$  dans l'interprétation  $\omega$  est un élément de 0.1 défini inductivement par:

si  $\phi \in PS$  alors  $\|\phi\|(\omega) = \omega(\phi)$

si  $\phi = cX_1...X_n$  alors  $\|\phi\|(\omega) = C_F(\|x_1\|(\omega)...\|x_n\|(\omega))$

$\omega$  **satisfait**  $\phi$  noté  $\omega \models \phi$  ssi  $\|\phi\|(\omega) = 1$

**Lorsque**  $\omega \models \phi$  on dit que  $\omega$  est un modèle de  $\phi$

**on note**  $\eta(\phi)$  l'ensemble des modèles de  $\phi$

$\omega \in PROP_{ps}$  **est valide** noté  $\models \phi$ , ssi toute interprétation  $\omega$  de  $PROP_{ps}$  satisfait  $\phi$

$\phi \equiv \psi$  sont logiquement équivalents ssi  $\phi \models \psi$  et  $\psi \models \phi$

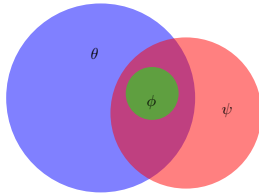
## 15.2 Propriétés de l'opérateur Models

**Réflexivité** :  $\phi \models \phi$

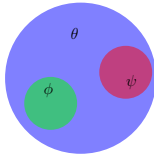
**Équivalence à gauche** : si  $\phi \equiv \theta$  et  $\phi \models \psi$  alors  $\theta \models \psi$

**Affaiblissement à droite (transitivité)** : si  $\phi \models \psi$  et  $\psi \models \theta$  alors  $\phi \models \theta$

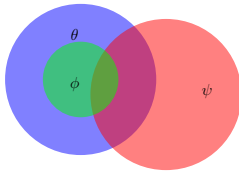
**Coupure** : si  $\phi \wedge \psi \models \theta$  et  $\phi \models \psi$  alors  $\phi \models \theta$



**Ou** :  $\phi \vee \psi \models \theta$  ssi  $\phi \models \theta$  et  $\psi \models \theta$



**Monotonie** : si  $\phi \models \theta$  alors  $\phi \wedge \psi \models \theta$





### 15.3 Ensemble de connecteurs fonctionnellement complet

On dit qu'un ensemble est fonctionnellement complet si avec que les connecteurs de cette ensemble on peut exprimer toutes les formules d'un monde.

$\{\neg, \wedge\}$  est fonctionnellement complet pour la logique propositionnel classique

Il en va de même pour  $\{\neg, \vee\}, \{vrai, \wedge, \oplus\}, \{\neg, \Rightarrow\}$  ou  $\{NAND\}$

**Suppression des fils équivalent** : Soit un arbre D ayant comme sous arbre plus d'une fois le nœud  $\alpha = (\top X \top)$ ,  $\alpha$  peut être remplacé par  $(\top)$  tout en concevant les modèles de D.

**fusion des nœuds** : Soit un arbre D ayant comme sous arbre les nœuds  $(aBc)$  et  $(a'B'c')$  et  $a = a', b = b', c = c'$  alors on peut faire relier les deux branches menant vers ces nœuds vers le même sous arbre.

### 15.4 Preuve par induction structurelle sur un ensemble de connecteurs non fonctionnellement complet

Soit  $\forall P \in \{\wedge, \vee\}_{ps}$ , vérifier P:

**Cas de base**  $\varphi \in PS$  :  $1 \rightarrow (\varphi) = 1$  donc  $1 \rightarrow$  constitue un modèle de  $\varphi$

**Étape inductive** :

$\varphi$  s'écrit :  $[\alpha \wedge \beta]$  ou  $[\alpha \vee \beta]$

Avec  $\alpha, \beta \in \{\wedge, \vee\}_{ps}$

Par hypothèse d'induction,  $\alpha$  et  $\beta$  vérifient P.

Il ne reste plus qu'à montrer que  $\varphi$  vérifie P.

$$\|\alpha \vee \beta\|(1 \rightarrow) = \vee \models (\|\alpha\|(1 \rightarrow), \|\beta\|(1 \rightarrow)) = \vee \models (1, 1) = 1$$

$$\|\alpha \wedge \beta\|(1 \rightarrow) = \wedge \models (\|\alpha\|(1 \rightarrow), \|\beta\|(1 \rightarrow)) = \wedge \models (1, 1) = 1$$

donc  $x \wedge \neg x$  ne vérifie pas P :  $\|x \wedge \neg x\|(1 \rightarrow) = 0$

## 15.5 Décomposition de Shannon

**On note**  $\phi[x \leftarrow 0]$  la formule obtenue en substituant dans  $\phi$  la constante faux à toutes les occurrences du symbole propositionnel  $x$ .

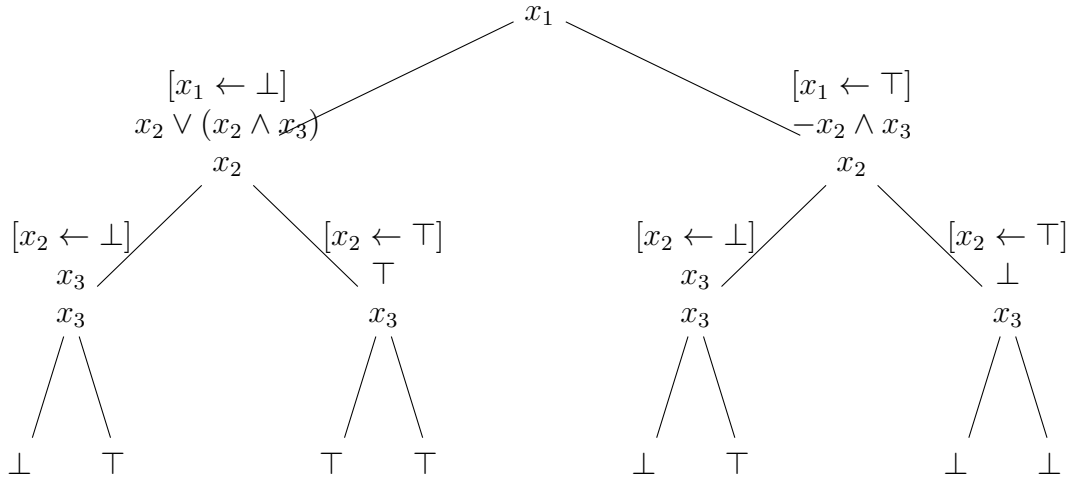
**On note**  $\phi[x \leftarrow 1]$  la formule obtenue en substituant dans  $\phi$  la constante vrai à toutes les occurrences du symbole propositionnel  $x$ .

La décomposition de Shannon de  $\phi$  suivant  $x$  est la formule:

$$(\neg x \wedge \phi[x \leftarrow 0]) \vee (x \wedge \phi[x \leftarrow 1])$$

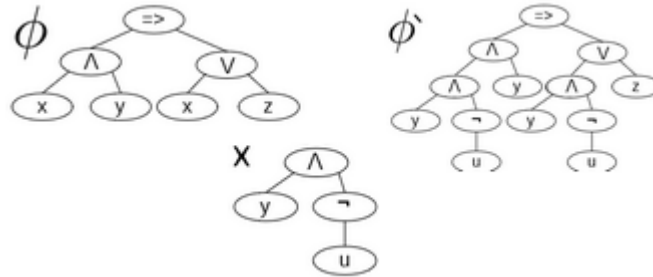
## 15.6 Arbre de Shannon, ROBDD

Étant donnée un ordre strict total  $x_1 < x_2 < x_3$  sur  $Var(\phi) = \{x_1, \dots, x_n\}$   
Et une formule  $\phi = (\neg x_1 \wedge x_2) \vee (\neg x_2 \wedge x_3)$



L'ensemble des modèles de  $\phi$  sont toutes les interprétation où la feuille vaut la valeur  $T$ .

### 15.6.1 Remplacement ou vérifonctionnalité



$\phi \equiv \phi'$  quelque soit la valeur de  $x$  (vrai ou faux).

### 15.6.2 Substitution

Soit un arbre  $D$  ayant comme nœud un sous arbre du type infixe  $\alpha = (x \Rightarrow y)$  et un sous arbre de substitution  $\beta = (\neg x \Rightarrow \neg y)$   
 $(D' = D_{\alpha \leftarrow \beta} \equiv D)$

## 15.7 Notion de impliquant premier

Les impliquant premier sont des sous formules des formules original tel que ces sous formules soit plus petite que la formule d'origine elle conserve les même modèles:

En circuit combinatoire les algo sont appelé Table de Karnaugh ou Quine-McCluskey.

### 15.7.1 Table de Karnaugh

Appliquer l'algorithme avec la formule  $S = \neg a b \neg c d + a \neg b \neg c \neg d + b \neg d$

S	$\neg a \neg b$	$\neg a b$	$a b$	$a \neg b$
$\neg c \neg d$	X	X	X	X
$\neg c d$		X	X	
$c d$		X	X	
$c \neg d$	X	X	X	X

les impliquant premier de  $S$  sont  $b \neg d$

### 15.7.2 Calcule arithmétique

En logique, les impliquant premier sont calculer que à partir d'une formule en mode CNF transposé en DNF et ensuite détransposé en CNF.

$$\phi = (a \wedge b \wedge c) \vee (\neg b \wedge c)$$

$$\phi = (a \vee \neg b) \wedge (a \vee c) \wedge (b \vee \neg b) \wedge (b \vee c) \wedge (c \vee \neg b) \wedge (c \vee c)$$

$$\phi = (a \vee \neg b) \wedge (a \vee c) \wedge (b \vee c) \wedge (c \vee \neg b) \wedge c$$

$$\phi = (a \vee \neg b) \wedge c$$

$$\phi = (a \wedge c) \vee (\neg b \wedge c) \text{ sont les impliquant premier.}$$

Via une table de Karnaugh:

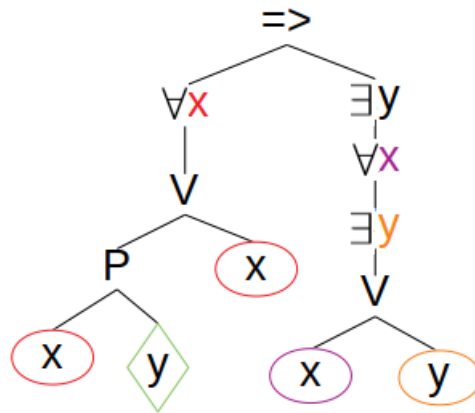
$\phi$	$\neg a \neg b$	$\neg ab$	$ab$	$a \neg b$
$\neg c$				
$c$	X		X	X
Égal à $(a \wedge c) \vee (\neg b \wedge c)$ .				

## Chapter 16

### Logique classique et prédicat du premier ordre

## 16.1 Syntaxe via les arbres

$\phi =$



### 16.1.1 Occurrences libre

Une occurrence libre est une variable n'ayant aucun quantificateur associé de son noeud à la racine de l'arbre.

par exemple le noeud  $y$  ayant un comme contour un losange vert est une occurrence libre, elle sera instancié que lors de l'interprétation de  $\phi$ .

### 16.1.2 Occurrences liée

Une occurrence liée est une variable ayant un quantificateur associé, comme:

**la variable  $x$  entouré d'un rond rouge** est définit via le quantificateur  $\forall x$  présent dans ces noeuds parent

**la variable  $x$  entouré d'un rond violet** est définit par le quantificateur de ces parents  $\forall x$

**la variable  $y$  entouré d'un rond orange** via le quantificateur  $\exists y$

A noté que les  $x$  entouré d'un rond de couleurs rouge sont différent des  $x$  entouré avec un rond orange, donc on peut tout bien renommer les  $x$  de

couleur orangé en  $z$  sans changer le sens de  $\phi$ .

Les occurrences liées se lient sur leur premier père le définissant, comme le  $y$  orange qui se définit que sur le  $\exists y$  le plus proche de lui.

### 16.1.3 Occurrences quantifié

Les occurrences quantifiées sont toutes les variables positionnées derrière un quantificateur, celle-ci montre comme dans la logique classique, le  $\forall$  (où quelque soit) ou  $\exists$  (où il existe au moins un).

On peut noter que sur la figure ci-dessus il y a un  $\exists y$  qui n'est pas associé à un  $y$  en feuille, on peut s'en débarrasser sans changer le sens de  $\phi$ .

### 16.1.4 Vocabulaire

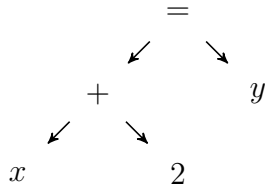
**Formule fermée** est une formule de  $FORM_L$  qui ne contient aucune variable libre.

**Formule instanciée** est une formule qui ne contient aucune occurrence libre ou liée de symbole de variable

## 16.2 Sémantique

Soit  $t$  un terme de  $TERM_L$ , la sémantique de  $t$  dans l'interprétation de  $I$  pour l'assignation  $X_i$  noté  $[[t]](I)(X_i)$  est l'élément de  $D_i$  défini inductivement.

$\phi =$



$= \in \mathfrak{R}$  d'arrêter 2

$+ \in \mathfrak{S}$  d'arrêter 2

$2 \in \mathfrak{S}$  d'arrêter 0

$X, Y \in X$

Avec une interprétation tel que:

$D_i = \mathbb{N}$

$+_1 = \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

$2_i = 3$

Avec une assignation tel que:

$X_i : X \rightarrow \mathbb{N}$

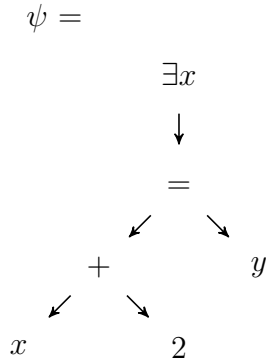
$x \rightarrow 5$

$y \rightarrow 10$



On peut calculer cette sous formule en appliquant chaque terme dans l'interprétation  $I$  pour un assignent  $X_i$ :

$$\begin{aligned}\|x + 2\|(I)(X_i) &= +_i(\|x\|(I)(X_i), \|2\|(I)(X_i)) = +_i(5, 3) = 8 \\ \|\phi\|(I)(X_i) &= =_i(8, 10) = 0(faux)\end{aligned}$$



$$\begin{aligned}\|\psi\|(I)(X_i)[x \leftarrow 7] &= \\ =_i(+_i(\|x\|(I)(X_i[x \leftarrow 7]), 3), \|y\|(I)(X_i[x \leftarrow 7])) &= \\ =_i(+_i(7, 3), 10) &= \\ =_i(10, 10) &= 1(vrai)\end{aligned}$$

Le quantificateur  $\forall$  ou  $\exists$  est plus prioritaire que les variables assigné dans  $X_i$ .

Soit  $\phi$  la formule  $\phi$  ci dessus, la formule interprété avec deux assignations différentes:

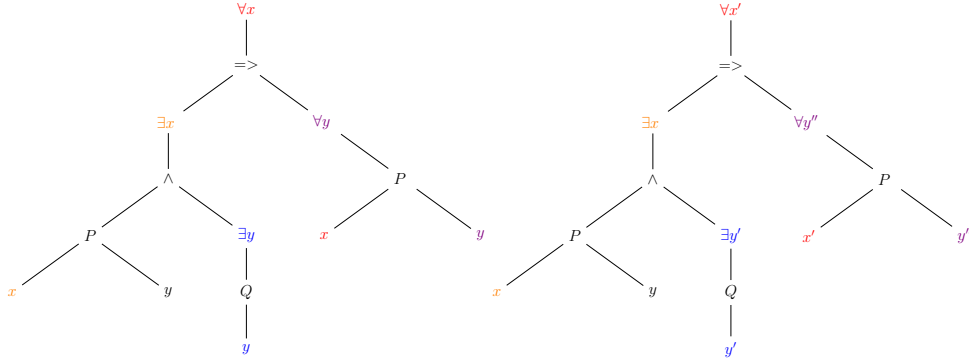
$$X_i^1 \quad x \rightarrow 5, y \rightarrow 10$$

$$X_i^2 \quad x \rightarrow 6, y \rightarrow 10$$

L'interprétation de  $\phi$  avec  $X_i^1$  est équivalent à  $\phi$  avec  $X_i^2$  car le symbole de quantification  $\exists$  est plus prioritaire que les assignations.

### 16.3 Formule polie

Une formule polie est une formule qui pour un nom de variable  $x$ , ne porte pas plusieurs significations. Pour se faire il suffit de renommer les variables. La formule de gauche n'est pas sous forme polie, mais celle de droite l'ai :



### 16.4 Équivalences remarquables

Pour tout  $\phi, \psi \in FORM_L$  et  $x, y \in X$

**Dualité**  $\forall x \phi \equiv \neg \exists x \neg \phi$

$$\forall x (\phi \wedge \psi) \equiv (\forall x \phi) \wedge (\forall x \psi)$$

$$\exists x (\phi \vee \psi) \equiv (\exists x \phi) \vee (\exists x \psi)$$

**Si  $x$  n'est pas libre dans  $\psi$  et  $Q = \forall$  ou  $\exists$  alors :**

$$Qx\phi \equiv \phi$$

$$Qx(\phi \wedge \psi) \equiv (Qx\phi) \wedge \psi$$

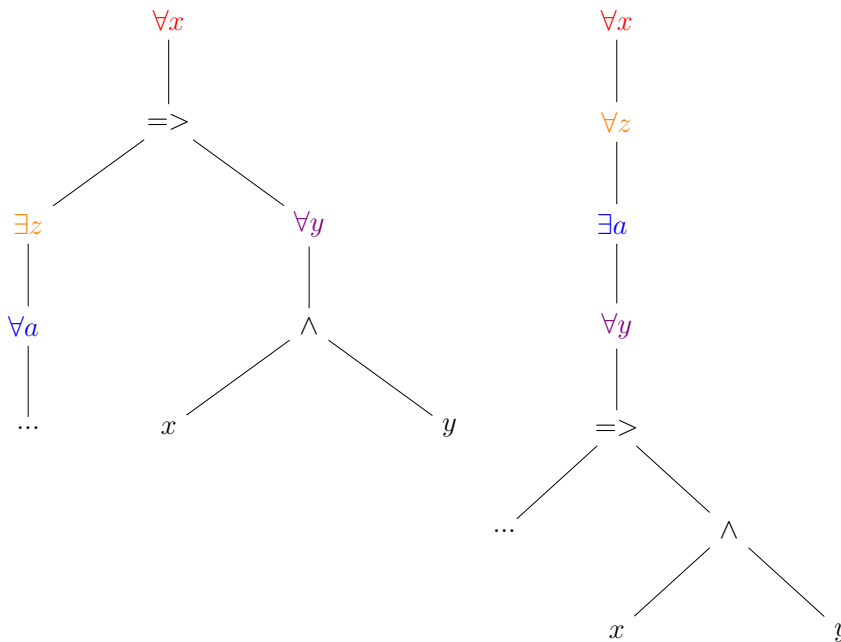
$$Qx(\phi \vee \psi) \equiv (Qx\phi) \vee \psi$$

$$\forall x \forall y \phi \equiv \forall y \forall x \phi$$

$$\exists x \exists y \phi \equiv \exists y \exists x \phi$$

## 16.5 Forme Prénexe

La mise en forme prénexe se fait en transformant la formule en forme polie puis en remontant tout les quantificateurs en haut de l'arbre en faisant attention que lorsqu'on remonte un quantificateur par de la une négation, on applique le dual sur le quantificateur, Et aussi il faut garder l'ordre des quantificateur par rapport à la profondeur de leur sous arbre:  
(Rappel que  $A \Rightarrow B \equiv \neg A \vee B$ ):



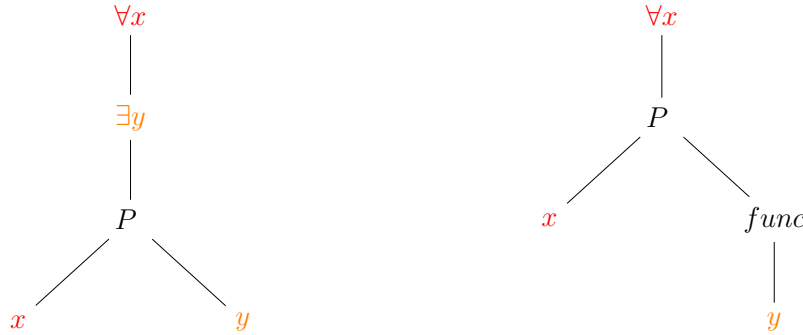
La partie contenant tout les quantificateurs s'appelle le Prefix et la partie sans quantificateurs s'appelle la Matrice.

Si dans la formule ci dessus on aurait changé le  $\Rightarrow$  par un  $\vee$  (ou autre chose sans signe de négation) les quantificateurs de couleur *orange* et *bleu* ne serait pas "dualisé", mais conserveront l'ordre de leurs profondeur.

Pareil si on remplace dans la formule le  $\Rightarrow$  par un  $\vee$  (ou autre chose sans signe de négation) et on s'intéresse exclusivement au quantificateur *orange* et *violet*, ( $\{\exists z, \forall, \forall y\}$ ) l'ordre de parcourt des sous arbres n'a aucune importance sur l'arbre final, (*GRD*) ou (*DRG*).

## 16.6 Scalénisation

Soit la formule suivante, scaléniser une formule c'est pour tout quantificateurs  $\exists y$  dépendant d'un quantificateur  $\forall x$ ,  $y$  peut se déduire via une fonction:



## 16.7 Forme propositionnelle

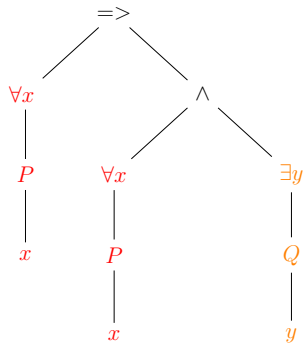
L'ensemble  $SFP(\phi)$  des sous-formules premières de  $\phi \in FORM_L$  est défini inductivement par:

Si  $\phi$  est un atome ou une formule du type  $\forall\psi$  ou  $\exists\psi$  alors  $SFP(\phi) = \{\phi\}$

Si  $\phi$  est une formule du type  $\neg\psi$  alors  $SFP(\phi) = SFP(\psi)$

Si  $\phi$  est une formule du type  $\psi \wedge \theta$  ou  $\psi \vee \theta$  ou  $\psi \Rightarrow \theta$  alors  $SFP(\phi) = SFP(\psi) \cup SFP(\theta)$

**Si la formule propositionnelle  $\phi$  est propositionnellement valide alors  $\phi$  est valide**



$SFP(\phi) = \{ \text{formules de couleur } \textcolor{red}{rouge}, \text{formules de couleur } \textcolor{orange}{orange} \}$ ,  $\phi$  est propositionnellement équivalent à  $A \Rightarrow (A \vee B)$  qui est propositionnellement valide donc  $\phi$  est valide

## Chapter 17

# Calculabilité et Machine de Turing

Soit une machine de turing  $M$  un quadruplet  $M = (K, \Sigma, \delta, s)$

$K$  ensemble fini d'état

$s \in K$  état initial

$\Sigma$  ensemble fini de symboles supposé disjoint de  $K$  et de deux symboles:

$\triangleright$  marque de début

$\sqcup$  séparateur ou fin de ruban

$\delta : (K \times \Sigma) \times ((K \cup \{yes, no, \uparrow\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\})$

$\{yes, no\}$  état acceptable

$\{\leftarrow, \rightarrow, -\}$  mouvement de la tête de lecture

## 17.1 Machines de Turing

Une machine de Turing:

**non déterministe** est une machine qui pour un état  $n$  donné peut dériver sur deux état  $n + 1$  différent (un état est aussi appelé une configuration, une dérivation peu aussi s'appeler une transition).

**Déterministe** est une machine qui pour un état  $n$  donné n'a qu'une seule possibilité de transition (autrement dit il n'y a que 1 seul  $n + 1$  unique).

**Décideur** est une machine qui pour un mot  $x \in L$  termine avec l'indice *yes* ou *no*.

**Accepteur** est une machine qui pour un mot  $x \in L$  termine avec l'indice *yes* ou  $\uparrow$  (boucle).

### 17.1.1 Machine de Turing universel

Prend un couple  $M((i,x))$  et l'exécute  $M_i(x)$ .

## 17.2 RE, coRE et R

**Un langage Récurif (R)** pour tout  $L \in R$  on peut trouver une Machine de Turing  $M$  déterministe qui décide  $L$ .

$\forall x \in (\sum \neg\{-\})^*$ , si  $x \in L$  alors  $M(x) = yes$  sinon  $M(x) = no$ .

**Un langage récursivement énumérable (RE)** pour tout  $L \in RE$  on peut trouver une Machine de Turing  $M$  déterministe qui accepte  $L$ .

$\forall x \in (\sum \neg\{-\})^*$ , si  $x \in L$  alors  $M(x) = yes$  sinon  $M(x) = \uparrow$ .

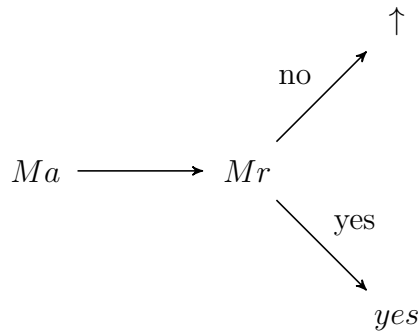
(coRE) sont tout les  $\{L \in partie(\sum \neg\{-\})^* \mid L^c \in RE\}$

Remarque:  $R \subseteq RE \cap coRE$

### 17.2.1 Preuve de $R$ est incluse dans $RE$

Montrer que  $L \in RE$  revient à pour une Machine de Turing Déterministe MT tel que MT accepte L lui associer une output différent.

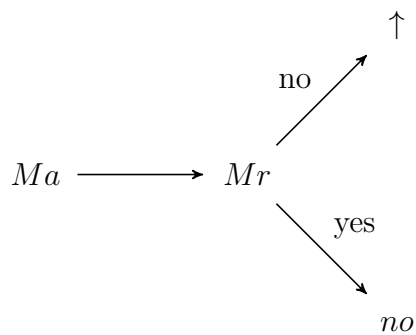
Si on sait qu'il existe un décideur  $Mr \in R$ , alors construire un accepteur  $Ma \in RE$ :



### 17.2.2 Preuve de $R$ est incluse dans $coRE$

Montrer que  $L \in coRE$  revient à pour une Machine de Turing Déterministe MT tel que MT reconnait L lui associer une output différent.

Si on sait qu'il existe un reconnait  $Mr \in R$ , alors construire un accepteur  $Ma \in coRE$ :





## 17.3 Problème de l'arrêt

$T(i, x, n)$   $i$  représente un indice de Machine

Vérifier si  $i$  décide un programme Si:

**No**  $\rightarrow$  FAUX

**YES** faire tourner  $M_i(x)$  sur  $n$  étapes.

Soit  $M - i(x)$  s'arrête avant les  $n$  étapes  $\rightarrow$  VRAI sinon FAUX

## 17.4 réduction fonctionnel

On dit que  $L_1 \leq_f L_2$  si il existe une réduction fonctionnel comme:

$$f : L_1 \rightarrow L_2 : x \rightarrow f(x)$$

### 17.4.1 Exemple de réduction fonctionnel

Soit  $L = \{(i, j) \mid \text{tel que } i \text{ et } h \text{ sont des indices de machine déterministes telles que pour tout mot d'entrée } x, \text{ on n'a } M_i(x) = \uparrow \text{ et } M_j(x) \neq \uparrow\}$

**Montrer que L est RE-difficile** revient à prouver  $HALTING \leq_f L$

$$f(i, x) \rightarrow (j, k):$$

$$(i, x) \in HALTING \text{ ssi } M_i(x) \neq \uparrow$$

$$(j, k) \in L \text{ ssi } M_j(y) = \uparrow \text{ et } M_k(y) \neq \uparrow, \forall y.$$

$$\begin{cases} M_j(y) & \text{boucle} \\ M_k(y) & M_i(x) \end{cases}$$

**Montrer que L est coRE-difficile** revient à prouver  $\neg HALTING \leq_f L$

$$f(i, x) \rightarrow (j, k):$$

$$(i, x) \in \neg HALTING \text{ ssi } M_i(x) \neq \uparrow$$

$$(j, k) \in L \text{ ssi } M_j(y) \neq \uparrow \text{ et } M_k(y) = \uparrow, \forall y.$$

$$\begin{cases} M_j(y) & y \\ M_k(y) & M_i(x) \end{cases}$$

**Part V**

**Recherche Opérationnel**

## Chapter 18

### Introduction à la PL

Construire une modèle linéaire, c'est donc:

**identifier** les variables de décision du problème

**déterminer** : la fonction objectif du modèle

**déterminer** : les contraintes du modèle

## 18.1 Modèle linéaire continu à 2 variables

Soit le modèle linéaire suivantes:

**Déterminer**  $(x, y) \in \mathfrak{S}^2$

**Minimisant**  $z = 1000x + 1200y$

**sous les contraintes** :

$$(1) 8x + 4y \leq 160$$

$$(2) 4x + 6y \leq 120$$

$$(3) x \leq 34$$

$$(4) y \leq 14$$

$$(5) 0 \leq x$$

$$(6) 0 \leq y$$

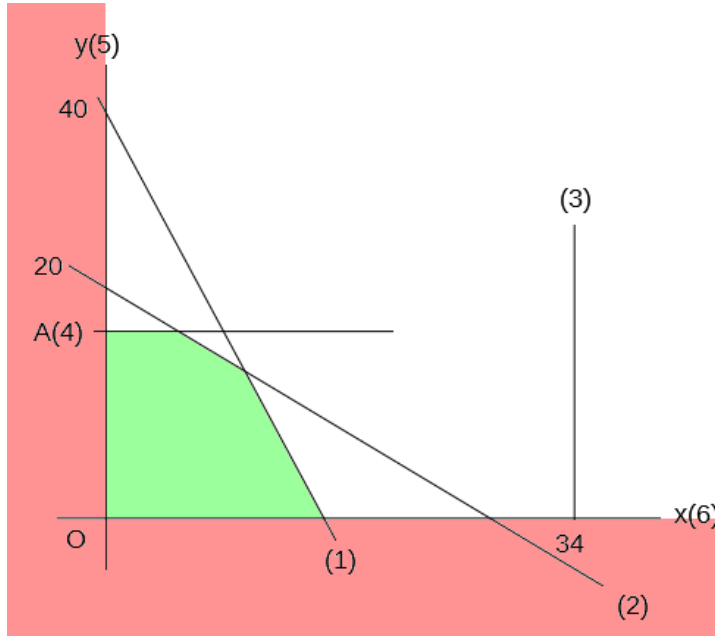
### 18.1.1 Recherche de solutions

Après avoir tracé graphiquement tout les points:

Pour chaque contrainte, tracer la droite et repérer le demi plan des solution: exemple pour (5) et (6), x et y doivent être supérieurs ou égal à 0, d'où le demi plan des solution sont toutes les valeurs positives.

La partie En vert représente la région admissible, quelque soit le point choisis

dans ce vert, aucune contrainte ne sera violé.



### 18.1.2 recherche de la solution optimal

Changer l'équation  $z$  tel que  $z$  soit égal à 0

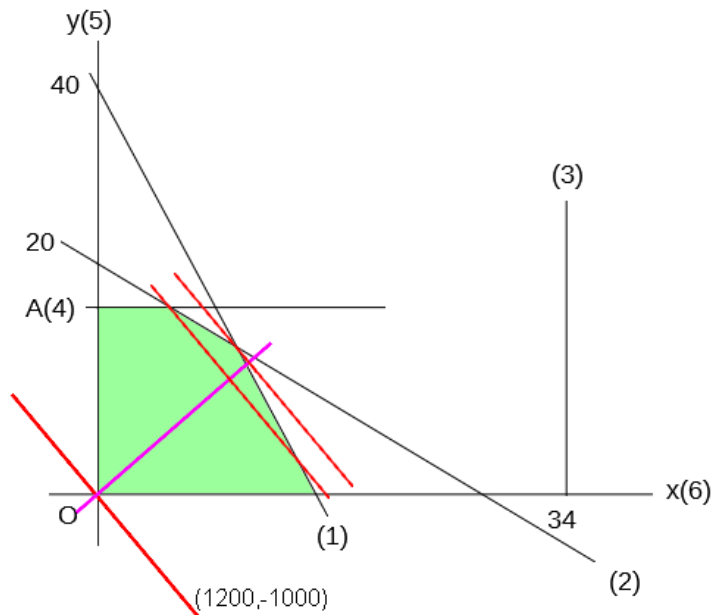
$$z = 1000x + 1200y = 0 = 1000 * (1200) + 1200 * (-1000)$$

Traçons la droite  $(0, 0), (1200, -1000)$

**Un point extrême** : est un point se trouvant sur l'intersection de 2 contraintes et étant dans la zone admissible.

**L'altitude** : est la droite (rouge) la plus haute touchant un point extrême, ce point sera le vecteur  $(x, y)$  le plus optimal pour  $z$ .

Les droites rouges doivent être toutes parallèles.



Dans cette exemple le point  $(15,10)$  est le point extrême maximal pour l'équation  $z$ .

## Chapter 19

### Le simplexe

Soit le modèle linéaire suivantes:

**Déterminer**  $(x, y) \in \mathfrak{S}^2$

**Maximisant**  $Z = 3x + 7y$

**sous les contraintes :**

$$(1) -x + y \leq 3$$

$$(2) y \leq 8$$

$$(3) 2x - y \leq 28$$

$$(5) 0 \leq x$$

$$(6) 0 \leq y$$

## 19.1 Initialisation du simplexe

Pour chaque expression du type (1)(2)(3) intégrer un  $e_i$  pour la transformer en équation.

On appelle les  $e_i$  des variables d'accumulation, Ce qui fait

**Déterminer**  $(x, y, e_1, e_2, e_3) \in \mathfrak{S}^5$

**Maximisant**  $Z = 3x + 7y$

**sous les contraintes :**

$$(1) -x + y + e_1 = 3$$

$$(2) y + e_2 = 8$$

$$(3) 2x - y + e_3 = 28$$

$$(5) 0 \leq x$$

$$(6) 0 \leq y$$

$$(7) e_1, e_2, e_3 \geq 0$$



## 19.2 Canonicité du modèle

Soit les valeurs (pour la première itération)

**Hors Base**  $(x, y)$

**Base**  $(e_1, e_2, e_3)$

Un modèle est canonique que si:

**si toutes les variables de Base** ne sont pas dans  $Z$ .

## 19.3 Solution admissible

$$(1) -x + y + e_1 = 3$$

$$(2) x - e_1 + e_2 = 5$$

$$(3) 3x - e_1 + e_3 = 25$$

**Variable hors base**  $= x, e_1$

**Variable Base**  $= y, e_2, e_3$

**Avec comme solution admissible**  $A \text{ Deduire}(x, y, e_1, e_2, e_3)$

Pour toute variable présente dans l'ensemble *Hors base* la valeur admissible est égal à 0

Donc solution admissible  $= (0, y, 0, e_2, e_3)$

Les 3 dernières valeurs sont les résultat des équations (soit 3, 5 et 25).

Pour chaque équation nous lisons les termes de droit à gauche et ignorons ceux qui sont dans l'ensemble *Hors Base*:

Donc solution admissible  $= (0, 3, 0, 5, 25)$

## 19.4 Exemple simple Premier itération

### 19.4.1 Choix de la variable entrante

**Gain marginale** prendre la variable non négatif ayant le plus haut coefficient.

$(x, y)$  sont deux choix possible, le tout est de choisir une bonne heuristique, comme celle du meilleur gain marginale, ou via la comparaison (en mode graphique):

$Y$  sera choisit, donc  $Y$  sera notre variable entrante.

### 19.4.2 Choix de la variable sortante

Pour chaque résultat d'équation, le diviser par sa valeur de  $Y$  (le résultat devant être positif sinon l'ignorer)

$$-x + y + e_1 = 3 \text{ donne } \frac{3}{1} = 3 \text{ (1 car } y = 1 * y)$$

$$y + e_2 = 8 \text{ donne } \frac{8}{1} = 8$$

$$2x - y + e_3 = 28 \text{ donne } \frac{28}{1} = 28$$

Prendre le minimum des variables, donc se sera 3.

la variable présente dans la Base sera prise comme variable sortante, dans notre cas  $e_1$ .

### 19.4.3 pivotage

On choisit l'équation associée à la variable  $e_1$  pour définir la variable entrante  $y$ .

On n'a:

$$y = \frac{1}{1} * (x - e_1 + 3)$$

Puis on crée les nouvelles équations via le nouveau  $y$ :

$$Z = 3x + 7y \text{ devient}$$

$$Z = 3x + 7(x - e_1 + 3)$$

$$Z = 10x - 7e_1 + 21$$

$$x - e_1 = 3 \text{ est déjà normalisé}$$

$$y + e_2 = 8 \text{ devient}$$

$$8 = x - e_1 + 3 + e_2$$

$$5 = x - e_1 + e_2$$

$$2x - y + e_3 = 28 \text{ devient}$$

$$28 = 2x + (x - e_1 + 3) + e_3$$

$$25 = 3x - e_1 + e_3$$

### 19.4.4 Nouveau modèle

Voici le nouveau modèle:

$$\text{Déterminer } (x, y, e_1, e_2, e_3) \in \mathbb{S}^5 \quad (1) \quad -x + y + e_1 = 3$$

$$(2) \quad x - e_1 + e_2 = 5$$

$$\text{Maximisant } Z = 10x - 7e_1 + 21$$

$$(3) \quad 3x - e_1 + e_3 = 25$$

$$\text{Variables hors base } x, e_1$$

$$(5) \quad 0 \leq x$$

$$\text{Variables de Base } y, e_2, e_3$$

$$(6) \quad 0 \leq y$$

$$\text{Solution admissible } (0, 3, 0, 5, 25)$$

$$\text{et } Z = 21$$

$$(7) \quad e_1, e_2, e_3 \geq 0$$

A ne pas oublier de vérifier la canonicité du modèle.

## 19.5 Exemple simple Seconde itération

### 19.5.1 Choix de la variable entrante

$X$  sera choisit, donc  $X$  sera notre variable entrante.

### 19.5.2 Choix de la variable sortante

$$\frac{5}{1} = 5$$

$$\frac{25}{3} = 8.3$$

Prendre le minimum des variables, donc se sera 5, donc  $e_2$ .

### 19.5.3 pivotage

$$x = \frac{1}{1} * (e_1 - e_2 + 5)$$

Puis on crée les nouvelles équations via le nouveau  $y$ :

$Z = 10x - 7e_1 + 27$  devient

$$Z = 10(e_1 - e_2 + 5) - 7e_1 + 27$$

$$Z = 3e_1 - 10e_2 + 71$$

$-x + y + e_1 = 3$  devient

$$3 = -(e_1 - e_2 + 5) + y + e_1$$

$$8 = y + e_2$$

$3x - e_1 + e_3 = 25$  devient

$$25 = 3(e_1 - e_2 + 5) - e_1 + e_3$$

$$10 = 2e_1 - 3e_2 + e_3$$

### 19.5.4 Nouveau modèle

Voici le nouveau modèle:

<b>Déterminer</b> $(x, y, e_1, e_2, e_3) \in \mathbb{S}^5$	(1) $y + e_2 = 8$
<b>Maximisant</b> $Z = 3e_1 - 10x + 71$	(2) $x - e_1 + e_2 = 5$
<b>Variables hors base</b> $e_2, e_1$	(3) $2e_1 - 3e_2 + e_3 = 10$
<b>Variables de Base</b> $y, x, e_3$	(5) $0 \leq x$
<b>Solution admissible</b> $(5, 8, 0, 0, 10)$ et $Z = 71$	(6) $0 \leq y$
	(7) $e_1, e_2, e_3 \geq 0$

A ne pas oublier de vérifier la canonicité du modèle.

## 19.6 Exemple simple, troisième itération

### 19.6.1 Variable entrante et sortante

La variable entrante sera  $e_1$

La variable sortante sera  $e_3$  car:

$\frac{8}{0}$  est NULL,  $\frac{5}{1}$  car négatif,  $\frac{10}{2} = 5$

### 19.6.2 Nouveau modèle

Voici le nouveau modèle:

<b>Déterminer</b> $(x, y, e_1, e_2, e_3) \in \mathbb{S}^5$	(1) $-\frac{1}{2}e_2 + \frac{e_3}{2} + e_1 = 10$
<b>Maximisant</b> $Z = 86 - \frac{11}{2}e_2 - \frac{3e_3}{2}$	(2) $e_2 + y = 8$
<b>Variables hors base</b> $e_2, e_3$	(3) $e_1 - \frac{3}{2}e_2 + \frac{e_3}{2} = 5$
<b>Variables de Base</b> $y, x, e_1$	(5) $0 \leq x$
<b>Solution admissible</b> $(10, 8, 5, 0, 0)$ et $Z = 86$	(6) $0 \leq y$
	(7) $e_1, e_2, e_3 \geq 0$

## 19.7 Exemple simple, dernière itération

Stop car  $e_2$  et  $e_3$  sont inférieure à 0 dans  $Z$ .

## Chapter 20

### Simplexe à deux phases

Soit le modèle suivant:

**Déterminer**  $(x, y) \in \mathfrak{S}^2$

**Maximisant**  $Z = 2x + 3y$

**sous les contraintes :**

(1)  $x + y \leq 4$

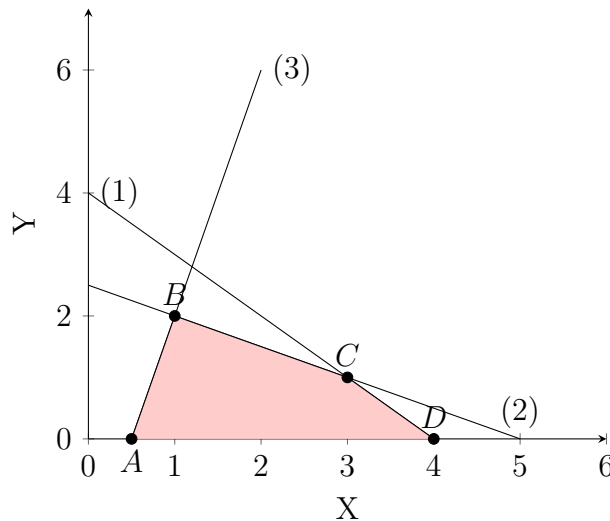
(2)  $x + 2y \leq 5$

(3)  $4x - y \geq 2$

(4-5)  $x, y \geq 0$

Lorsque le sens de l'équation est  $\leq$  il faut ajouter une variable  $e_i$ , dans le cas des équations  $\geq$  il faut ajouter une variable d'excédant  $a$  dans la contrainte concerné et instaurer  $Z$  à  $-a$

La représentation graphique ci dessous:



## 20.1 Première phase du simplexe à deux phases

Pour toutes expression sous la forme  $A \geq -i$ , multiplier les deux coté par  $-1$  et inverser le signe pour obtenir des équations positif.

Si une contrainte est jugé redondante, alors elle peut être éliminé sans changer



le modèle.

Le modèle ci dessus n'est pas canonique, donc nous allons exprimer  $Z$  en fonction de l'équation portant le symbole  $a$ :

### 20.1.1 Nouveau modèle

Voici le nouveau modèle:

**Déterminer**  $(x, y, e_1, e_2, e_3, a) \in \mathbb{S}^5$       **Solution admissible**  $(0, 0, 4, 5, 0, 2)$   
et  $Z = -2$

**Maximisant**  $Z = -a = 4x - y - e_3 - 2$       (1)  $x + y + e_1 = 4$   
 ~~$Z = 2x + 3y$~~       (2)  $x + 2y + e_2 = 5$

**Variables hors base**  $x, y, a$       (3)  $4x - y - e_3 + a = 2$

**Variables de Base**  $e_1, e_2, e_3$       (4-5)  $x, y, e_i, a \geq 0$

Ce modèle est canonique.

## 20.2 Premier phase du simplexe à deux phases, première itération

### 20.2.1 Variable entrante et sortante

La variable entrante sera  $x$

La variable sortante sera  $a$  car:

$$\frac{4}{1}, \frac{5}{1}, \frac{2}{4} = \frac{1}{2}$$

### 20.2.2 pivotage

$$x = \frac{1}{2} * (y + e_3 - a + 2) = \frac{y}{4} + \frac{e_3}{4} - \frac{a}{4} + \frac{1}{2}$$

### 20.2.3 Nouveau modèle

Voici le nouveau modèle:

<b>Déterminer</b> $(x, y, e_1, e_2, e_3, a) \in \mathfrak{S}^5$	<b>Solution admissible</b> $(\frac{1}{2}, 0, \frac{7}{2}, \frac{9}{2}, 0, 0)$ et $Z = 0$
<b>Maximisant</b> $Z = -a$ $\%oldZ = 2x + 3y$	(1) $\frac{5}{4}y + e_1 + \frac{e_3}{4} - \frac{a}{4} = \frac{7}{2}$ (2) $\frac{9}{4}y + e_2 + \frac{e_3}{4} - \frac{a}{4} = \frac{9}{2}$
<b>Variables hors base</b> $y, a$	(3) $x - \frac{y}{4} - \frac{e_3}{4} + \frac{a}{4} = \frac{1}{2}$
<b>Variables de Base</b> $e_1, e_2, x$	(4-5) $x, y, e_i, a \geq 0$

### 20.3 Premier phase du simplexe à deux phases, seconde itération

Nous sommes en présence d'un système optimal car  $Z$  à l'altitude 0.  
Une solution admissible serait ( $PG =$ ):

$A(\frac{1}{2}, 0)$  est le point extrême correspondant:

$$\begin{aligned} y_A &= 0 \\ 4x_A - y_A &= 2 \end{aligned}$$

Comme  $z = 0$  on passe en phase 2.

### 20.4 Seconde phase du simplexe à deux phases

Voici le nouveau modèle:

<b>Déterminer</b> $(x, y, e_1, e_2, e_3) \in \mathfrak{S}^5$	<b>Solution admissible</b> $(\frac{1}{2}, 0, \frac{7}{2}, \frac{9}{2}, 0)$ et $Z = 0$
<b>Maximisant</b> $Z = oldZ = 2x + 3y$	(1) $\frac{5}{4}y + e_1 + \frac{e_3}{4} = \frac{7}{2}$ (2) $\frac{9}{4}y + e_2 + \frac{e_3}{4} = \frac{9}{2}$
<b>Variables hors base</b> $y$	(3) $x - \frac{y}{4} - \frac{e_3}{4} = \frac{1}{2}$
<b>Variables de Base</b> $e_1, e_2, x$	(4-5) $x, y, e_i \geq 0$

On retire toutes les occurrences de  $a$ .

Le modèle n'est pas canonique car  $x$  est hors base, donc remplacer  $x$  dans  $Z$  car il est défini :

$$Z = 2x + 3y = 2\left(\frac{y}{4} + \frac{e_3}{4} + \frac{1}{2}\right) + 3y = \frac{7}{2}y + \frac{e_3}{2} + 1$$

Voici le nouveau modèle :

**Déterminer**  $(x, y, e_1, e_2, e_3) \in \mathbb{S}^5$  et  $Z = 1$

**Maximisant**  $Z = \frac{7}{2}y + \frac{e_3}{2} + 1$  (1)  $\frac{5}{4}y + e_1 + \frac{e_3}{4} = \frac{7}{2}$

**Variables hors base**  $y, e_3$  (2)  $\frac{9}{4}y + e_2 + \frac{e_3}{4} = \frac{9}{2}$

**Variables de Base**  $e_1, e_2, x$  (3)  $x - \frac{y}{4} - \frac{e_3}{4} = \frac{1}{2}$

**Solution admissible**  $(\frac{1}{2}, 0, \frac{7}{2}, \frac{9}{2}, 0)$  (4-5)  $x, y, e_i \geq 0$

Ce modèle est canonique.

## 20.5 Seconde phase du simplexe à deux phases, première itération

### 20.5.1 Variable entrante et sortante

La variable entrante sera  $y$

La variable sortante sera  $e_2$  car :

$$\frac{\frac{7}{2}}{\frac{5}{4}} = \frac{14}{5}, \frac{\frac{9}{2}}{\frac{9}{4}} = 2, \leq 0$$

### 20.5.2 pivotage

$$y = \frac{4}{9} * (-e_2 - \frac{e_3}{4} + \frac{9}{2}) = -\frac{4}{9}e_2 - \frac{e_3}{9} + 2$$

### 20.5.3 Nouveau modèle

Voici le nouveau modèle:

<b>Déterminer</b>	$(x, y, e_1, e_2, e_3) \in \mathfrak{S}^5$	et $Z = 8$
<b>Maximisant</b>	$Z = -\frac{14}{9}e_2 + \frac{e_3}{9} + 8$	(1) $x - \frac{5}{9}e_2 + \frac{e_3}{9} = 1$
<b>Variables hors base</b>	$e_2, e_3$	(2) $y + \frac{4}{9}e_2 + \frac{e_3}{9} = 2$
<b>Variables de Base</b>	$e_1, y, x$	(3) $x + \frac{e_2}{9} - \frac{2}{9}e_3 = 1$
<b>Solution admissible</b>	$(1, 2, 1, 0, 0)$	(4-5) $x, y, e_i \geq 0$

Ce modèle est canonique.

## 20.6 Seconde phase du simplexe à deux phases, seconde itération

### 20.6.1 Variable entrante et sortante

La variable entrante sera  $e_3$

La variable sortante sera  $e_1$  car:

$$\frac{1}{\frac{1}{9}} = 9, \frac{2}{\frac{1}{9}} = 18, \leq 0$$

### 20.6.2 pivotage

$$e_3 = 9(-e_1 + \frac{5}{9}e_2 + 1) = -9e_1 + 5e_2 + 9$$

### 20.6.3 Nouveau modèle

Voici le nouveau modèle:

<b>Déterminer</b>	$(x, y, e_1, e_2, e_3) \in \mathfrak{S}^5$	<b>Variables de Base</b>	$e_3, y, x$
<b>Maximisant</b>	$Z = -e_1 - e_2 + 9$	<b>Solution admissible</b>	$(3, 1, 0, 0, 9)$ et $Z = 9$
<b>Variables hors base</b>	$e_2, e_1$	(1)	$9y + 5e_2 + e_3 = 9$

$$(2) \ y - e_1 + e_2 = 1$$

$$(4-5) \ x, y, e_i \geq 0$$

$$(3) \ x + 2e_1 - e_2 = 3$$

Ce modèle est canonique.

## **20.7    Seconde phase du simplexe à deux phases, troisième itération**

Il n'existe pas de variable entrante car  $e_1$  et  $e_3 \leq 0$

## Part VI

# Représentation des connaissances et raisonnement

## Chapter 21

### Logique propositionnel

## 21.1 Vocabulaire

Les *Logiques propositionnelles* sont définies via les symboles suivants:

$\top, \perp, C, \neg C, C \wedge C, C \vee C, C \Rightarrow C$

**Littéral** est un atome ou la négation d'un atome

**Clause** est une disjonction de littéraux

**Cube** est une conjonction de littéraux

**CNF** est une forme normale conjonctive (une conjonction de clauses)

**DNF** est une forme normale disjonctive (une disjonction de cubes)

## 21.2 cohérence d'un ensemble de clauses

Soit  $K$  un ensemble de clauses pouvant être réduit via les axiomes:

$$x \vee x \vee y_1 \vee \dots y_n \equiv x \vee y_1 \vee \dots y_n$$

$$x \vee \neg x \vee y_1 \vee \dots y_n \equiv \text{'top'}$$

$$x \vee \top \equiv \top$$

$$x \vee \perp \equiv x$$

Si  $K$  est vide alors  $K$  est cohérente

Si  $\perp \in K$  alors  $K$  est incohérente

$K_{x \leftarrow \top}$  est le résultat du remplacement des occurrences de  $x$  par  $\top$

$K_{x \leftarrow \perp}$  est le résultat du remplacement des occurrences de  $x$  par  $\perp$



## Chapter 22

### Introduction à la logique de description

## 22.1 Attributive Language with Complement

Les  $ALC$  sont définis via les symboles suivant:

$\top, \perp, C, \neg C, C \sqcap C, C \sqcup C, \forall r.C, \exists r.C$

### 22.1.1 Propriétés

Pour toutes les interprétations  $\iota = \langle \Delta^I, .^I \rangle$ , et pour tout  $C, D \in \ell_{ALC}$ :

$$\begin{aligned}
 (\neg \neg C)^I &= C^I & (\neg \exists r.C)^I &= (\forall r. \neg C)^I \\
 (\neg (C \sqcap D))^I &= (\neg C \sqcup \neg D)^I \\
 (\neg (C \sqcup D))^I &= (\neg C \sqcap \neg D)^I & \exists r. \perp &\equiv \perp \\
 (\neg \forall r.C)^I &= (\exists r. \neg C)^I & \forall r. \top &\equiv \top
 \end{aligned}$$

## 22.2 Logique de description

Défini via les symboles suivant:

$\ell_{ALC}, C \sqsubseteq C, \sqsupseteq C$

### 22.2.1 Sémantique

$\iota \models C \sqsubseteq D$  (*satisfait*  $C \sqsubseteq D$ ) si  $C^I \subseteq D^I$

$\iota \models C \equiv D$   $\iota \models C \sqsubseteq D$  et  $\iota \models C \sqsupseteq D$

### 22.2.2 Assertions

$a : C$   $a$  est une instance de  $C$

$(a, b) : r$   $a$  et  $b$  sont attachés avec la relation  $r$

## 22.3 TBoxes et ABoxes

Soit une base de connaissance  $KB = \langle T, A \rangle$  où:

$$T = \begin{cases} EmpStud \equiv Student \sqcap Employee \\ Student \sqcap \neg Employee \sqsubseteq \neg \exists pays.Tax \\ EmpStud \sqcap \neg Parent \sqsubseteq \exists pays.Tax \\ EmpStud \sqcap Parent \sqsubseteq \neg \exists pays.Tax \\ \exists worksFor.Company \sqsubseteq Employee \end{cases}$$

$$A = \begin{cases} ibm : Company \\ mary : Parent \\ john : EmpStud \\ (john, ibm) : workFor \end{cases}$$

### 22.3.1 Subsumption

D'après la TBoxes et la ABoxes ci dessus, dire que A subsume B c'est dire que A est plus spécifique que B:

Does *EmpStud* subsume *Student*  $\sqcap$  *Employee* ? : yes

Does *Student*  $\sqcap$  *Parent* subsume *EmpStud*  $\sqcap$  *Parent* ? : yes

Does  $\exists pays.\perp$  subsume *EmpStud* ? : No

### 22.3.2 Classification

Les schémas de classification aide pour trouver les subsumptions:



### 22.3.3 Instance checking

On n'a

*ibm* est une instance de *Company*

*mary* est une instance de *Parent*

*john* est une instance de *EmpStud*, *Student*, *Employee*

*john* n'est pas une instance de  $\neg$ *Parent*

*(john, ibm)* est une instance de *workFor*

### 22.3.4 Retrieval

*Student* ?{*john*}

$\neg \exists$ *pays.Tax* ?{*mary*}

$\neg(\neg$ *Employee*  $\sqcap$   $\exists$ *pays.Tax*) ?{*john, mary*}

$\forall$ *worksFor.Company* ?{ }

*Employee*  $\sqcup$   $\forall$ *pays.* $\neg$ *Tax*  $\sqcup$  *Company* ?{*ibm, john, mary*}

$\neg$ *Tax*  $\sqcup$   $\exists$ *pays.* $\perp$   $\sqcup$   $\forall$ *workdFor.* $\forall$ *pays.* $\top$  ?{*ibm, john, mary*}

### 22.3.5 Equivalence of concept

Are *Student*  $\sqcap$  *Employee*  $\sqcap$   $\neg$ *EmpStud* and  $\exists$ *worksFor.* $\perp$  équivalent? *Yes*

Are *Student*  $\sqcap$   $\forall$ *worksFor.* $\neg$ *Company* and *Student*  $\sqcap$   $\neg$ *Employee* équivalent?  
*No*

### 22.3.6 Concept satisfiability

*EmpStud*  $\sqcap$  *Parent*  $\sqcap$   $\exists$ *pays.* $\top$  satisfiable? *Yep*

$\neg \forall$ *worksFor.* $\neg$ *Company*  $\sqcap$   $\neg$ *Employee* satisfiable? *No*

*Employee*  $\sqcap$  *Company* satisfiable ? *Yep*

### 22.3.7 ABox consistency

Is  $A_2 = A \cup \{\textit{john} : \exists \textit{worksFor} . \neg \textit{Company}\}$  consistent wrt  $T$  ? : *Yes*

Is  $A_3 = A \cup \{\textit{mary} : \exists \textit{pays} . \textit{Tax}\}$  consistent wrt  $T$  ? : *No*

### 22.3.8 Réduction et consistance

Soit  $KB = \langle T, A \rangle, C, D \in \iota_{ALC}, a \in I$  and  $a'$  new in  $KB$

**Concept subsumption wrt  $T$**  :  $KB \models C \sqsubseteq D$  ssi  $\langle T, A \cup \{a' : C \sqcap \neg D\} \rangle$  est inconsistent

**Instance chacking** :  $KB \models a : C$  ssi  $\langle T, A \cup \{a : \neg C\} \rangle$  est inconsistent

**Concept satisfiability wrt  $T$**  :  $C$  est satisfiable wrt  $T$  ssi  $\langle T, A \cup \{a' : C\} \rangle$  est consistent

$KB \models \textit{EmpStud} \sqcap \textit{Parent} \sqsubseteq \neg \exists \textit{pays} . \textit{Tax} \sqcap \textit{Employee}$  ?

$KB \cup \{a : \textit{EmpStud} \sqcap \textit{Parent} \sqcap (\exists \textit{pays} . \textit{Tax} \sqcup \neg \textit{Employee})\} \models \perp?$ , for  $a$  new

$KB \models \textit{john} : \textit{Student} \sqcap \exists \textit{empBy} . \top$  ?

$KB \cup \{\textit{john} : \neg(\textit{Student} \sqcap \exists \textit{empBy} . \top)\} \models \perp?$

Is  $\textit{EmpStud} \sqcap \neg \exists \textit{pays} . \textit{Tax}$  satisfiable wrt  $KB$  ?

$KB \cup \{a : \textit{EmpStud} \sqcap \neg \exists \textit{pays} . \textit{Tax}\} \not\models \perp?$ , for  $a$  new

## Chapter 23

# Méthode des Tableau pour les ALC

## 23.1 Pre processing

### 23.1.1 Réécriture

Réécrite chaque:

$$C \sqsubseteq D \text{ dans } T \text{ en } \top \sqsubseteq \neg C \sqcup D$$

$$A \sqsubseteq \exists r.B \text{ en } \top \sqsubseteq \neg A \sqcup \exists r.B$$

Changer la  $KB$  en  $NNF$  ( $\neg$  occurs only in front of concept names)

$$\neg\neg C \rightarrow C$$

$$\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$$

$$\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D$$

$$\neg(\exists r.C) \rightarrow \forall r.\neg C$$

$$\neg(\forall r.C) \rightarrow \exists r.\neg C$$

### 23.1.2 Vocabulaire

**Blocage/Blocking** l'apparition d'une boucle infini dans le déroulement de l'algorithme

**Clash** Quand il existe une contradiction d'un noeud feuille vers l'un de ses ascendant

### 23.1.3 Règles d'expansion

$\sqsubseteq_T$  – rule

**Si**  $a : C \in A, \top \sqsubseteq D \in T$  **et**  $a : D \notin A$  **alors**  
 $A := A \cup \{a : D\}$

$\sqcap$  – rule

**Si**  $a : C \sqcap D \in A$  **et**  $\{a : C, a : D\} \not\subseteq A$  **alors**  
 $A := A \cup \{a : C, a : D\}$

$\sqcup$  – rule

**Si**  $a : C \sqcup D \in A$  **et**  $\{a : C, a : D\} \cap A = \emptyset$  **alors**  
 $A := A \cup \{a : E\}, \text{ for some } E \in \{C, D\}$

$\exists$  – rule

**Si**  $a : \exists R.C \in A$  **et il n'y a pas de**  $b$  **st**  $\{(a, b) : R, b : C\} \subseteq A$  **et**  
 $a$  **n'est pas en en blocage** **alors**  
 $A := A \cup \{(a, c) : R, c : C\}, \text{ for } c \text{ new in } A$

$\forall$  – rule

**Si**  $\{a : \forall R.C, (a, b) : R\} \subseteq A$  **et**  $b : C \notin A$  **alors**  
 $A := A \cup \{b : C\}$

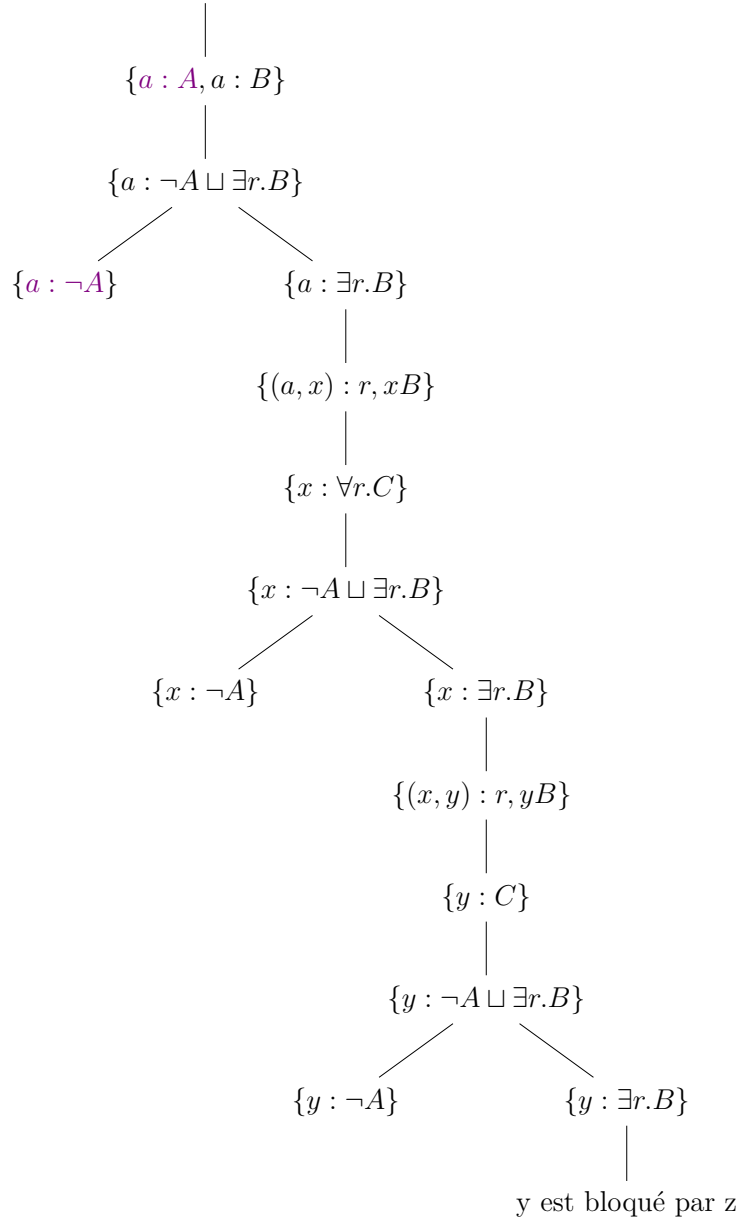


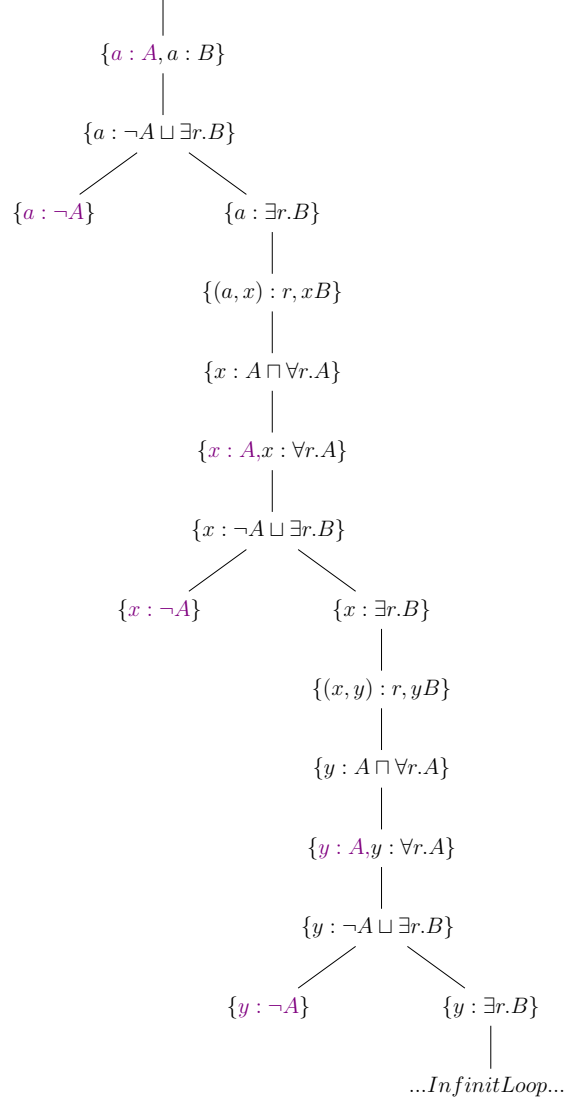
## 23.2 Exemple

$$T = \{A \sqsubseteq \exists r.B\} \equiv \{\top \sqsubseteq \neg A \sqcup \exists r.B\}$$

$$A = \{a : A \sqcap B, a : \forall r.\forall r.C\}$$

$$\{a : A \sqcap B, a : \forall r.\forall r.C\}$$



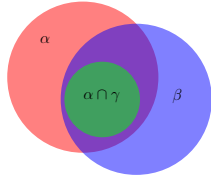


## Chapter 24

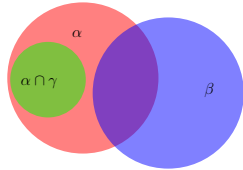
### Logique presque tout

Soit le nouvelle opérateur binaire  $\Subset$  disent pour *presque tout* A est dans B.

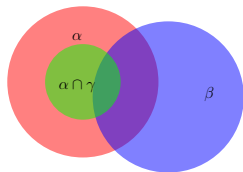
**Bonne distribution :**



**Mauvaise distribution :**



**Cas général :**



## 24.1 Système P

Réflexivité :

Almost all :  $\alpha \vdash \alpha$

ensembliste :  $A \in A$

Équilibrage à gauche :

Almost all : Si  $\models \alpha \Leftrightarrow \beta$  et  $\alpha \vdash \gamma$  alors  $\beta \vdash \gamma$

ensembliste : Si  $A = B$  et  $A \in C$  alors  $B \in C$

Équilibrage à droite :

Almost all : Si  $\alpha \models \beta$  et  $\gamma \vdash \alpha$  alors  $\gamma \vdash \beta$

ensembliste : Si  $A \subseteq B$  et  $C \in A$  alors  $C \in B$

Coupure :

Almost all : Si  $(\alpha \wedge \beta) \vdash \gamma$  et  $\alpha \vdash \beta$  alors  $\alpha \vdash \gamma$

ensembliste : Si  $(A \cap B) \in C$  et  $A \in B$  alors  $A \in C$

Monotonie :

Almost all : Si  $\alpha \vdash \beta$  et  $\alpha \vdash \gamma$  alors  $\alpha \wedge \beta \vdash \gamma$

ensembliste : Si  $A \in B$  et  $A \in C$  alors  $(A \cap B) \in C$

Ou :

Almost all : Si  $\alpha \vdash \gamma$  et  $\beta \vdash \gamma$  alors  $\alpha \vee \beta \vdash \gamma$

ensembliste : Si  $A \in C$  et  $B \in C$  alors  $(A \cup B) \in C$

### 24.1.1 Exemple

Soit:

$Q$  : être québécoises

$C$  : être canadiens

$F$  : le fait de parler français

$A$  : le fait de parler anglais

$S$  : le fait d'aimer le sirop d'érable

Presque tout les canadiens ne parlent pas le français :  $C \models \neg F$

Presque tout les québécois parlent le français :  $Q \models F$

Les québécois aiment le sirop d'érable :  $Q \Rightarrow S \equiv Q \models S$

Les québécois sont canadiens  $Q \Rightarrow C \equiv Q \models C$

Presque tout les québécois canadiens parlent le français

Nous avons  $Q \models C$  et  $Q \models F$

Avec la monotonie on obtient  $Q \wedge C \models F$

Presque tout les québécois canadiens parlent le français ou l'anglais

Avec  $Q \wedge C \models F$

Par ailleurs nous avons  $F \models F \vee A$

Alors via l'équilibrage à droite  $Q \wedge C \models F \vee A$

## 24.2 Tolérance du Système P

Soit la basse de connaissance:

$$\begin{array}{l} \Delta \quad C \Rightarrow \neg F \\ \quad Q \Rightarrow F \\ \\ W \quad Q \Rightarrow S \\ \quad Q \Rightarrow C \end{array}$$

Pour une formule de type  $A \Rightarrow B$  dans  $\Delta$  dire si il existe une interprétation qui vérifie  $A \Rightarrow B$  et qui satisfait chacune des règles de  $\Delta$  et  $W$

**Pour la formule  $C \Rightarrow \neg F$  est satisfait**

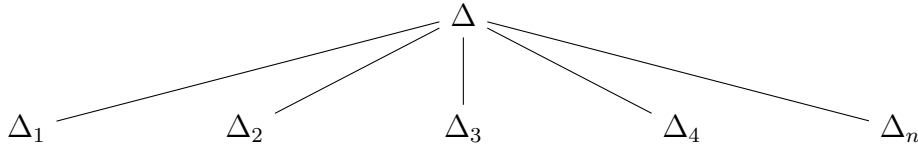
$$\begin{array}{l} \Delta \quad C^1 \Rightarrow \neg F^0 \\ \quad Q^0 \Rightarrow F^0 \\ \\ W \quad Q^0 \Rightarrow S^s \\ \quad Q^0 \Rightarrow C^1 \end{array}$$

**Pour la formule  $Q \Rightarrow F$  n'est pas satisfait**

$$\begin{array}{l} \Delta \quad C^1 \Rightarrow \neg F^1 \equiv \neg \top \vee \perp \\ \quad Q^1 \Rightarrow F^1 \\ \\ W \quad Q^1 \Rightarrow S^s \\ \quad Q^1 \Rightarrow C^1 \end{array}$$

## 24.3 Stratification du système P

$\Delta$  stratifiable (ou cohérente) c'est le fait de pouvoir diviser  $\Delta$  en  $\Delta_i$ .  
 $\Delta_i$  est plus général que  $\Delta_{i+1}$



Si  $\alpha \rightarrow \beta$  est une conséquences de  $\Delta$ , alors  $\{\alpha \rightarrow \neg\beta\} \cup \Delta$  est incohérente.  
 A chaque tour dans  $\Delta$  appliquer la tolérances et si il y a une interprétation, bouger la formule dans  $\Delta_i$ , Si  $\Delta_i$  est vide alors ce n'est pas stratifiable, si  $\Delta$  est vide alors c'est stratifiable.

## 24.4 Exemple de stratification possible

### 24.4.1 Initialisation

$$\Delta = \{C \rightarrow \neg F, Q \rightarrow F\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\alpha \rightarrow \beta = (Q \wedge C) \rightarrow \neg F$$

### 24.4.2 Première itération

On n'a:

$$\Delta = \{C \rightarrow \neg F, Q \rightarrow F, (Q \wedge C) \rightarrow F\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{\}$$

Pour  $C^1 \rightarrow \neg F^0$  est toléré par l'algorithme donc transféré dans  $\Delta_1$  à la fin du tour:

$$\Delta = \{C^1 \rightarrow \neg F^0, Q^0 \rightarrow F^0, (Q^0 \wedge C^1) \rightarrow F^0\}$$

$$W = \{Q^0 \Rightarrow S^s, Q^0 \Rightarrow C^1\}$$

$$\Delta_1 = \{\}$$

Pour  $Q \rightarrow F, (Q \wedge C) \rightarrow F$  ne sont pas tolérés par l'algorithme:

$$\Delta = \{C \rightarrow \neg F, Q \rightarrow F, (Q \wedge C) \rightarrow F\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{\}$$



### 24.4.3 Seconde itération

On n'a:

$$\Delta = \{Q \rightarrow F, (Q \wedge C) \rightarrow F\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{C \rightarrow \neg F\}$$

$$\Delta_2 = \{\}$$

Pour  $Q \rightarrow F$  et  $(Q \wedge C) \rightarrow F$  sont toléré donc seront transféré dans  $\Delta_2$  à la fin du tour:

$$\Delta = \{\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{C \rightarrow \neg F\}$$

$$\Delta_2 = \{Q \rightarrow F, (Q \wedge C) \rightarrow F\}$$

$\Delta$  est vide donc  $\{\alpha \rightarrow \neg\beta\} \cup \Delta$  est stratifiable.

## 24.5 Exemple de stratification non possible

### 24.5.1 Initialisation

$$\Delta = \{C \rightarrow \neg F, Q \rightarrow F\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\alpha \rightarrow \beta = (Q \wedge C) \rightarrow (F \vee A)$$

### 24.5.2 Première itération

On n'a:

$$\Delta = \{C \rightarrow \neg F, Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{\}$$

Pour  $C^1 \rightarrow \neg F^0$  est toléré par l'algorithme donc transféré dans  $\Delta_1$  à la fin du tour:

$$\Delta = \{C^1 \rightarrow \neg F^0, Q^0 \rightarrow F^0, (Q^0 \wedge C^1) \rightarrow (\neg F^0 \wedge \neg A^a)\}$$

$$W = \{Q^0 \Rightarrow S^s, Q^0 \Rightarrow C^1\}$$

$$\Delta_1 = \{\}$$

Pour  $Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)$  ne sont pas tolérés par l'algorithme:

$$\Delta = \{C \rightarrow \neg F, Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{\}$$

### 24.5.3 Seconde itération

On n'a:

$$\Delta = \{Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{C \rightarrow \neg F\}$$

$$\Delta_2 = \{\}$$

Pour  $Q \rightarrow F$  et  $(Q \wedge C) \rightarrow (\neg F \wedge \neg A)$  ne sont pas tolérés:

$$\Delta = \{Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)\}$$

$$W = \{Q \Rightarrow S, Q \Rightarrow C\}$$

$$\Delta_1 = \{C \rightarrow \neg F\}$$

$$\Delta_2 = \{\}$$

$\Delta_2$  est vide donc  $\{\alpha \rightarrow \neg\beta\} \cup \Delta$  n'est pas stratifiable.

## Chapter 25

### Logique de description DL Lite

## 25.1 Opérateurs

Pour une *ABox*:

$\neg$  négation

$\exists$  Rôle  $\rightarrow$  Concept

$$\begin{pmatrix} A & , B \\ C & , D \end{pmatrix} \exists \rightarrow \begin{pmatrix} A \\ C \end{pmatrix}$$

$\neg$  Rôle  $\rightarrow$  Rôle

$$\begin{pmatrix} A & , B \\ C & , D \end{pmatrix} \neg \rightarrow \begin{pmatrix} B & , A \\ D & , C \end{pmatrix}$$

## 25.2 Requêtes

### 25.2.1 Grounded query

**Sous la forme**  $(\wedge_{i=1}^n A_i(a)) \wedge (\wedge_{j=1}^m P_j(a, b))$

**Avec**  $A_i$  des concepts et  $P_i$  des rôles.

**Exemple** : *Student*(Jean)  $\wedge$  *Teacher*(Paul)  $\wedge \dots \wedge$  *HasSupervisor*(Jean, Paul)

### 25.2.2 Conjonctives Query

**Sous la forme**  $q = \{x | \exists y. conj1(x, y) \wedge conj2(Bob, y) \wedge conj3(y)\}$

Si x donne une liste non vide alors c'est une réponse de type *array*

Sinon c'est une sortie de type *boolean*

### 25.3 Fermetures négatives

**Sur  $\text{DL-Lit}_{core}$**  Tout les axiomes négatifs de la  $TBox$  sont dans  $cln(T)$

si  $B_1 \sqsubseteq B_2 \in T$  and  $B_2 \sqsubseteq \neg B_3 \in T$  alors  $B_1 \sqsubseteq \neg B_3 \in T$

si  $B_1 \sqsubseteq B_2 \in T$  and  $B_3 \sqsubseteq \neg B_2 \in T$  alors  $B_1 \sqsubseteq \neg B_3 \in T$

Avec les règles ce dessus dérivons les *negated closure*:

<b><math>\text{DL-Lit}_{core}</math> TBox</b>	$cln(T)$
$Teacher \sqsubseteq \neg Student$	$Teacher \sqsubseteq \exists HasSupervisor$
$Teacher \sqsubseteq \exists TeachesTo$	$\exists HasSupervisor^\neg \sqsubseteq \neg Student$
$\exists TeachesTo^\neg \sqsubseteq Student$	$\exists TeachesTo^\neg \sqsubseteq \neg Teacher$
$Student \sqsubseteq \exists HasSupervisor$	
$\exists HasSupervisor^\neg \sqsubseteq Teacher$	

## 25.4 Gestion des contraintes et MultiABox

### 25.4.1 Expansion

Note  $o_{cl}$ , Qui va agrandir la ABox avec les axiomes de la TBox

$TBox$	$ABox$	La MultiABox M est composé que d'une ABox.
$\exists P \sqsubseteq B$	$A(a)$	$B(a)$ est ajouté grâce au second axiome
$A \sqsubseteq B$	$P(c,b)$	$B(c)$ est ajouté grâce au premier axiome
$A \sqsubseteq \neg C$	$B(a)$	
	$B(c)$	

### 25.4.2 Splitting

Note  $o_{incl}$ , Qui va Séparer les conflits en créant plusieurs ABox

$TBox$	$MultiAboxes( ABox_1, ABox_2 )$	
$C \sqsubseteq \neg B$	$B(a)$	$C(e)$
	$C(a)$	$B(e)$
	$B(b)$	$B(b)$

$$o_{incl} = \{ \{B(a), B(b)\}, \{B(b), C(a)\}, \{C(a)\}, \{C(e)\}, \{B(e)\} \}$$

### 25.4.3 Selection

Note  $o_{card}$ , Qui crée une nouvelle ABox contenant tout les ABox ayant le plus haut cardinal

$TBox$	$ABox_1$	$ABox_2$	$ABox_3$
$C \sqsubseteq \neg B$	$P(c,b)$	$C(a)$	$B(c)$
	$B(a)$	$B(b)$	
$o_{incl} = \{ABox_1, Abox_2\}$			

### 25.4.4 Modifieurs

$$o_{cl}(o_{cl}(M)) = o_{cl}(M)$$

$$o_{incl}(o_{incl}(M)) = o_{incl}(M)$$

$$o_{card}(o_{card}(M)) = o_{card}(M)$$

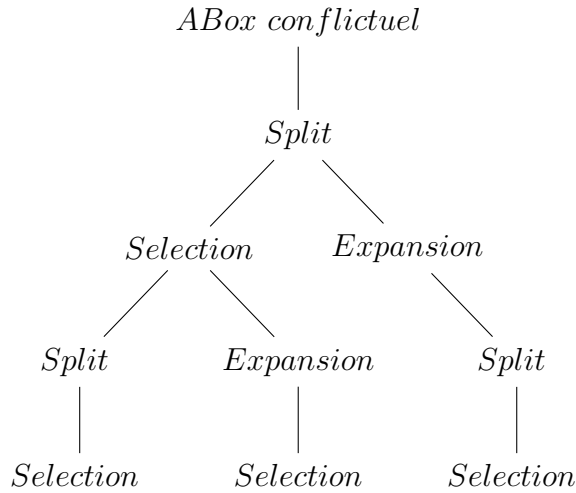
$$o_{cl}(O_d(O_{cl}(M))) = o_d(o_{cl}(M))$$

$$o_{incl}(O_d(O_{incl}(M))) = o_d(o_{incl}(M))$$

$$_d = \{incl, card, cl\}$$



### 25.4.5 Complex modifieurs



### 25.4.6 Décision avec plusieurs ABox

**Universal Inférence** : Si toutes les ABox répondent la réponse R, alors R sera retourné

**Existencial Inférence** : Si au moins une ABox retourne T, alors R sera retourné

**Safe inférence** : Faire l'intersection de toutes les ABox puis calculer le résultat

**Mogority inférence** : Si plus de la moitié des ABox répondent avec le résultat R, alors R sera prise

**Base inférence** : Si plus de  $\alpha$  ABox répondent avec le résultat R, alors R sera prise

La différence entre la Safe inférence et l'Universal inférence:

Soit  $TBox = \{A \sqsubseteq \neg B, A \sqsubseteq E, B \sqsubseteq E\}$ ,  $ABox = \{A(a), B(a)\}$   
Via la résolution des contraintes on obtient:  
 $A_1 = \{A(a)\}$ ,  $A_2 = \{B(a)\}$   
avec comme  $x = E(a)$

**Pour la stratégie  $\forall$**

$(T, A_1) \models E(a) \rightarrow OUI$

$(T, A_2) \models E(a) \rightarrow OUI$

Conclusion OUI

**Pour la stratégie *Safe***

$(T, (A_1 \cap A_2)) \models E(a)$

$\emptyset \models E(a) \rightarrow NON$

Conclusion NON

## Chapter 26

## Complexité

## 26.1 Analyse de complexité pour D(M1, Safe)

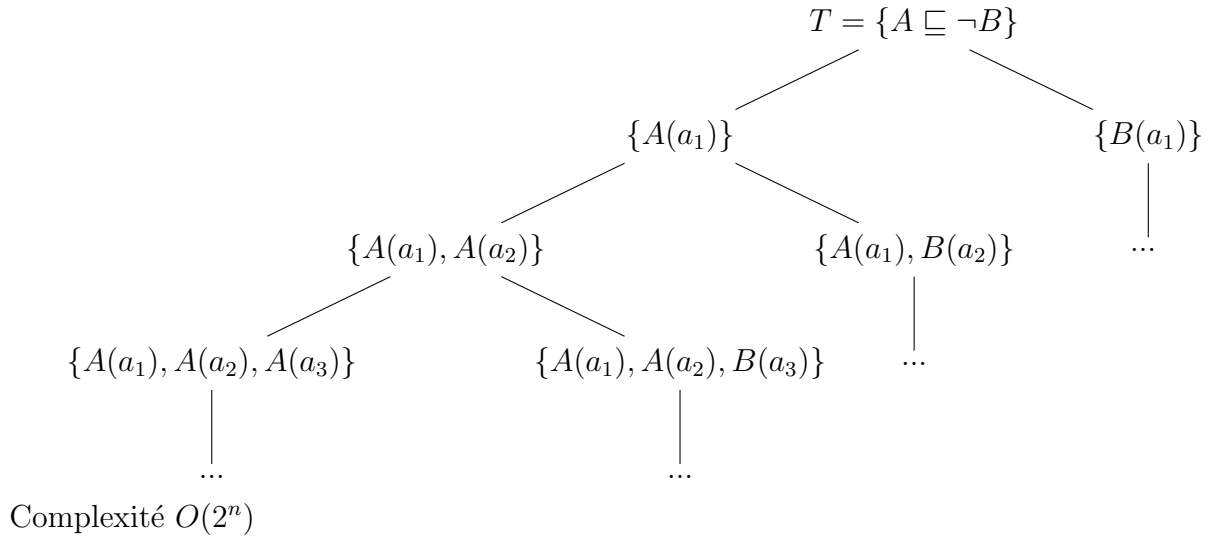
Une approche naïf non satisfiable serait de:

- (1) Calculer les  $R_1 \dots R_n$  après avoir appliqué le *modifiersplitting*
- (2) Calculer l'intersection des  $R_i$

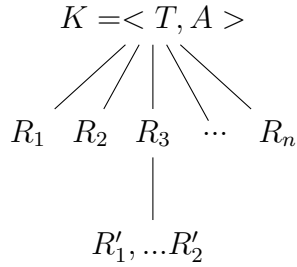
Un cas extrême pour résoudre le problème

$$T = \{A \sqsubseteq \neg B\}$$

$$A = \{A(a_1), B(a_1), \dots A(a_n), B(a_n)\}$$



## 26.2 Analyse de la complexité pour D(M2,Forall)



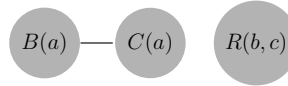
Splitting et Selection selon le cardinal  
le plus haut:  
 $R'_1, \dots R'_r$

Soit la transformation de ce problème vers un problème dont la complexité est connue, Prenons K-MIS qui est similaire à ce problème de Splitting et Selection:

$$K = \langle T, A \rangle$$

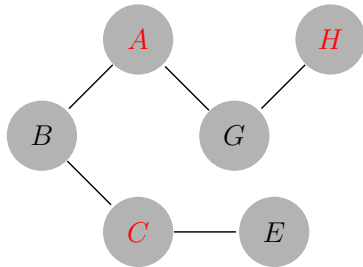
$$T = \{B \sqsubseteq \neg C\}$$

$$A = \{B(a), C(a), R(b, c)\}$$



Soit le nouveau graph, effectuer la transformation inverse du K-MIS vers DLlite.

Soit K-MIS un problème NP-Complet qui consiste à déterminer le nombre maximum de nœuds tel que ces nœuds une fois colorié ne sont pas adjacent à un autre nœud colorié, K=3 dans l'exemple ci dessous



Concept =  $\{A, B, C, E, G, H\}$ ,  
Individu =  $\{e\}$ , Role =  $\{\}$

$$TBox = \{H \sqsubseteq \neg G, A \sqsubseteq \neg H, B \sqsubseteq \neg A, C \sqsubseteq \neg B, E \sqsubseteq \neg C\}$$

$$ABox = \{A(e), B(e), C(e), E(e), G(e), H(e)\}$$

$$Cln(T) = T$$

**Part VII**

**Théories de la Décision**

# Chapter 27

## Théorie de la décision

La problématique est celle d'un agent qui doit prendre la meilleure décision, parmi un ensemble de choix possibles (actes), qui selon l'état du monde, mèneront à des conséquences (résultats/outcomes) différentes.

Soient  $A = \{a_1, \dots, a_k\}$  les actes possibles

Soient  $S = \{s_1, \dots, s_m\}$  les états du monde

Soient  $C = \{c_1, \dots, c_n\}$  les conséquences

On n'a donc  $A \times S \rightarrow C$

Le but est de trouver le  $a_i$  qui permet d'obtenir les meilleures conséquences  $c_j$ .

On distingue 3 type de théories de la décision:

**Décisions sous certitude** il n'y a qu'une état du monde.

**Décision dans l'incertain** il y a plusieurs états du monde.

**Décision dans le risque** il y a plusieurs états du monde, dont on connaît la probabilité.

train	voiture
10	20

Décision sous certitude

	train	voiture
Normal	10	20
Bouchon	10	0

Décision dans incertitude

		train	voiture
Normal	80%	10	20
Bouchon	20%	10	0

Décision dans le risque



## 27.1 Décision dans l'incertain

### 27.1.1 Critère de Laplace

Choisir l'acte dont la conséquence moyenne est la meilleure.

$$\operatorname{argmax}_{a \in A} \sum_{s \in S} \frac{1}{|A|} * u(a(s))$$

### 27.1.2 Critère de Wald

Choisir l'acte dont la pire conséquence est la meilleure (maximum).

$$\operatorname{argmax}_{a \in A} \min_{s \in S} u(a(s))$$

### 27.1.3 Critère d'Hurwicz

Meilleur compromis entre meilleure et pire conséquences ( $\alpha \in [0, 1]$ )

$$\operatorname{argmax}_{a \in A} (\alpha * \min_{s \in S} u(a(s))) + ((1 - \alpha) * \max_{s \in S} u(a(s)))$$

### 27.1.4 Min Max Regret

Choisir l'acte dont on regrettera le moins les conséquences

$$\operatorname{argmax}_{a \in A} \max_{s \in S} R(a, s) \text{ avec } R(a, s) = \max_{b \in A} u(b(s)) - u(a(s))$$

### 27.1.5 Example

Actes	Etats du monde						
	$s_1$	$s_2$	$s_3$	<i>Laplace</i>	<i>Wald</i>	<i>Hurwicz</i> <sub>.5</sub>	<i>MinMaxRegret</i>
$a_1$	55 <sub>21</sub>	10 <sub>12</sub>	13 <sub>13</sub>	26	10	34	21
$a_2$	40 <sub>36</sub>	19 <sub>3</sub>	22 <sub>4</sub>	27	19	31	36
$a_3$	30 <sub>48</sub>	20 <sub>0</sub>	26 <sub>0</sub>	26	22	28	46
$a_4$	76 <sub>0</sub>	2 <sub>20</sub>	0 <sub>26</sub>	26	0	38	26

### 27.1.6 Différents cadres d'incertitude

Décision dans le risque (incertitude probabiliste) : MinMax Regret

Décision dans l'incertain (incertitude qualitative) : Prade

Décision sous incertitude stricte : Wald, Hurwicz

Décision sous ignorance total : Konieczny, Marquis

## Chapter 28

### Théorie des jeux

## 28.1 Jeux sous forme stratégique

Un jeu sous forme stratégique est défini par:

un ensemble  $N = \{1, \dots, n\}$  de joueurs

pour chaque joueurs  $i$  un ensemble de stratégies  $S_i = \{s_1, \dots, s_{n_i}\}$

pour chaque joueurs  $i$  une fonction de valuation  $u_i : S_1 \times \dots \times S_n \rightarrow R_i$   
qui pour un ensemble de stratégies associe les gains du joueur  $i$

On notera:

$s$  un profil de stratégies  $\{s_1, \dots, s_n\}$  où  $\forall i s_i \in S_i$

$s_{-i}$  le profil  $s$  des stratégies autre que celle du joueurs  $i$

$S$  l'espace des stratégies

### 28.1.1 utilité

On appelle utilité la mesure de chaque situation aux yeux de l'agent, celle ci n'est si une mesure du gain matériel, monétaire, etc, mais une mesure subjective du contentement de l'agent.

### 28.1.2 jeux sous forme extensive et stratégique

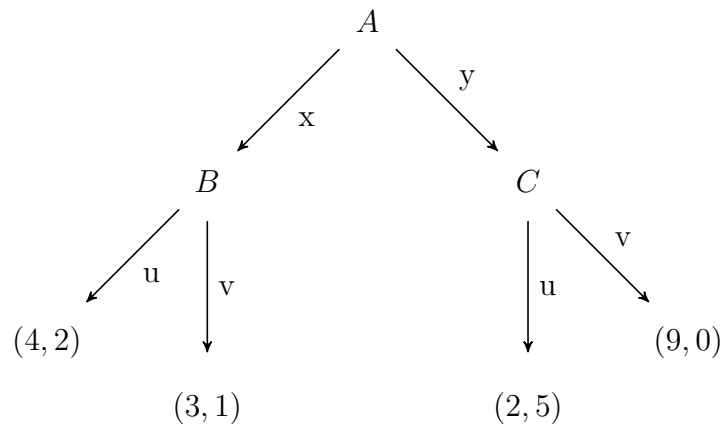
	u	v
x	4,2	3,1
y	2,5	9,0

Forme stratégique

x et y étant les choix représenté par le joueur 1.

u et v étant les choix représenté par le joueur 2.

Si le joueur 1 choisit x et le joueur 2 v alors le joueur 1 gagnera 3 et le joueur 2 gagnera 1.



Forme Extensive

### 28.1.3 Élimination de stratégies dominées

Une stratégie  $s_i$  est (strictement) dominé pour le joueur  $i$  si il existe une stratégie  $s'_i$  telle que pour tout profil  $s_{-i}$

$$u_i(s'_i, s_{-i}) > u_i(s_i, s_{-i})$$

Une stratégie faiblement dominé est sous la forme:

$$u_i(s'_i, s_{-i}) \geq u_i(s_i, s_{-i})$$

	u	v
x	4,2	3,1
y	2,5	9,0

	u	$v$
x	4,2	$3,1$
y	2,5	$9,0$

	u	$v$
x	4,2	$3,1$
$y$	$2,5$	$9,0$

Le profil (4,2) est sélectionné donc Joueur 1 gagnera 4 et Joueur 2 gagnera 2.

### 28.1.4 Équilibre de Nash

Un jeu peut avoir plusieurs ou aucun équilibre de Nash.

	u	v	w
x	3,0	0,2	0,3
y	2,0	$1,1$	2,0
z	0,3	0,2	3,0

Deux équilibre de Nash sont interchangeable si la permutation des termes gauche garde l'équilibre de Nash actif.

Voici un cas particulier:

	u	v
x	2,1	0,0
y	0,0	1,2

Deux équilibre de Nash sont présent  $(2,1)$  et  $(1,2)$ . Comme il y a une hésitation entre les deux cas, alors l'utilisation du flip coin est envisageable:

$$u_1(< (x, 1/2), (y, 1/2) >) = 1/2 * 2 + 1/2 * 0 = 1$$

$$u_1(< (u, 1/2), (v, 1/2) >) = 1/2 * 0 + 1/2 * 1 = 1/2$$

### 28.1.5 Critère de Pareto

Soit la table:

	u	v
x	4,4	3,1
y	2,3	7,5

Pour u, x est meilleur que y

Pour v, y est meilleur que x

Pour x, u est meilleur que v

Pour y, v est meilleur que u

Un profil  $s$  domine un profil  $s'$  dans le sens de Pareto si pour tout les joueurs  $s$  est au moins meilleur que  $s'$  et que pour un joueur  $s$  est meilleur strictement que  $s'$ .

Un profil  $s$  domine strictement un profil  $s'$  dans le sens de Pareto si pour tout les joueurs  $s$  est meilleur que  $s'$ .

	u	v
x	4,4	3,1
y	2,3	7,5

(x,u) 4,4

(y,v) 7,5 est meilleur

### 28.1.6 Niveau de sécurité

Pour un tableau:

	u	v
x	9,9	0,8
y	8,0	7,7

Dans le cas d'un jeu avec des joueurs non rationnel, l'un des deux joueur peut duper l'autre et ainsi gagner 8 et faire gagner 0 à l'autre joueur.

On défini le niveau de sécurité d'une stratégie  $s_i$  pour le joueur  $i$  comme le gain minimum que peut apporter cette stratégie quel que soit le choix des autres joueurs.

On défini le niveau de sécurité d'un joueur  $i$  comme le niveau de sécurité maximal des stratégies de  $i$ .

Le meilleur choix serait de prendre (y,v) pour assurer un minimum de gain pour chaque personnes.

### 28.1.7 autres Stratégies

Jusque là nous avons utilisé que les stratégies pures, c'est à dire les option qui se présente au joueurs.

Une stratégies mixte est une distribution de probabilité sur l'ensemble des stratégies pures.

Une stratégie local du joueur  $i$  est une distribution de probabilités  $p_i$  définit sur l'ensemble des stratégies pure du joueurs  $i$ .

Une stratégie comportemental du joueur  $i$  est un vecteur de stratégies locales du joueur  $i$ .

### 28.1.8 Équilibre de Nash en stratégies mixtes

Soit le problème:

		y	1-y
		f	c
x	f	2,1	0,0
1-x	c	0,0	1,2

Soit  $y$ , la probabilité avec laquelle le joueur 2 joue f, quelle est la meilleure réponse du joueur 1?:

$$u_1(<(f, y), (c, 1 - y)>) = y * 2 + (1 - y) * 0 = 2y$$

$$u_1(<(f, y), (c, 1 - y)>) = y * 0 + (1 - y) * 1 = 1 - y$$

Donc:

Si  $2y > 1 - y$  avec  $(y > 1/3)$ , la meilleur réponse du joueur 1 est de jouer  $f$

Si  $2y < 1 - y$  avec  $(y < 1/3)$ , la meilleur réponse du joueur 1 est de jouer  $c$

Si  $2y = 1 - y$  avec  $(y = 1/3)$ , le joueur 1 peut jouer l'un ou l'autre.

Soit  $x$ , la probabilité avec laquelle le joueur 1 joue f, quelle est la meilleure réponse du joueur 2?:

$$u_1(<(f, x), (c, 1 - x)>) = x * 1 + (1 - x) * 0 = x$$

$$u_1(<(f, x), (c, 1 - x)>) = x * 1 + (1 - x) * 0 = 2(1 - x)$$



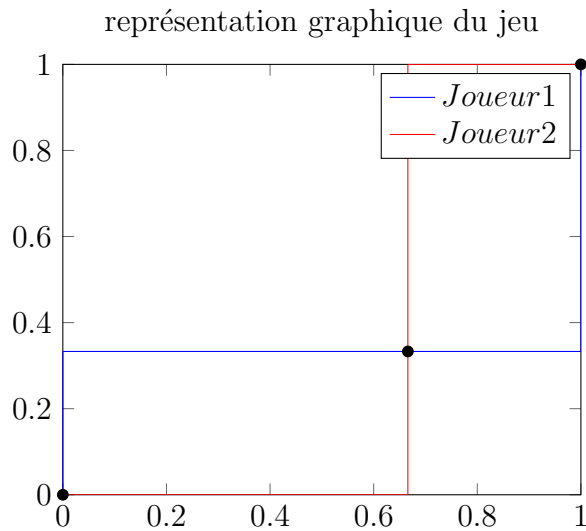
Donc:

Si  $x > 2(1 - x)$  avec  $(x > 2/3)$ , la meilleur réponse du joueur 2 est de jouer  $f$

Si  $x < 2(1 - x)$  avec  $(x < 2/3)$ , la meilleur réponse du joueur 2 est de jouer  $c$

Si  $x = 2(1 - x)$  avec  $(x = 2/3)$ , le joueur 2 peut jouer l'un ou l'autre.

### 28.1.9 Représentation graphique du jeu



### 28.1.10 Coopération

	f	c
f	2,1	0,0
c	0,0	1,2

Que se passe t'il si les 2 joueurs peuvent communiquer avant de jouer?:

$$u_1 = u_2 = \frac{1}{2} * 2 + \frac{1}{2} = \frac{3}{2}$$

Lorsque tout les joueurs peuvent observer un même événement aléatoire, ils peuvent alors s'accorder sur des équilibres corrélés.

Selon un accord prit via un flip coin, ou via une parution d'un évènement, si les deux joueurs se mette d'accord sur le fait de tirer  $f$  ou  $c$ , mais le joueur désavantagé peut ne pas jouer le choix prit, mais il s'expose à ne rien gagner.

### 28.1.11 Itération le dilemme des prisonniers

Deux personne sont arrêtées ensemble en possession d'armes à feu sont soupçonnés d'un délit fait en commun, Les policiers les séparent et disent à chacun:

Si l'un des deux avoue et que l'autre ne dit rien, le premier est libéré et le second emprisonné 5 ans.

Si les deux avouent, les deux iront 4 ans en prison.

Si les deux ne disent rien alors ils seront emprisonné 2 ans.

Donnant le tableau suivant:		P2 avoue	P2 rien
	P1 avoue	4,4	0,5
	P1 rien	5,0	2,2

Mais les valeurs inscrit ne représente pas le gain, donc il faut inverser les valeurs par rapport au maximum (5):

	P2 avoue	P2 rien
P1 avoue	1,1	5,0
P1 rien	0,5	3,3

Si le joueur 1 avoue et le joueur 2 ne dit rien alors le joueur 1 gagnera 5 ans et le joueur 2 gagner 0 ans (car il sera en prison).

### 28.1.12 DIP Itérations

Les joueurs se rencontrent plusieurs fois

A chaque itération les joueurs ont connaissances des coups précédents

ils ne connaissent pas les terme du jeu

le gain d'un joueur est le cumul des gains de chaque rencontre

Pour favoriser le coopération on ajoute la contrainte

$$X + T < 2R$$

	Coopérer	non coopérer
Coopérer	$R = 3$ récompense pour coopération mutuelle	$S = 0$ salaire de la dupe
non coopérer	$T = 5$ tentation à trahir	$P = 1$ punition pour la trahison mutuelle

### 28.1.13 Les Stratégies

Dans une rencontre les joueurs peuvent avoir plusieurs comportements différent, appliquons les sur le problème DIP:

**gentille** le joueur sera gentil quitte à perdre

**méchante** le joueur ne laisse rien passé, tout doit être acquérir

**par pattern** suivre la même séquence de choix à l'infini

**rancunière** donnant donnant jusqu'à l'erreur de l'adversaire

**lunatique** aléatoirement

**majoritaire gentille** joue ce que l'autre joue

**majoritaire méchante** trahir

**donnant donnant** joue le coup précédent de l'autre

Quand on fait jouer ses stratégies dans le cadre d'un tournoi on obtient les scores suivant:

place	stratégie	points
1	donnant donnant	42
2	majoritaire gentille	19
3	rancunière	4
4	lunatique	0

le top 3 n'est pas compliqué à déduire, se sont tous des stratégies adaptatif (qui s'adapte à l'adversaire).

le top 2 a un pouvoir pour pardonner l'adversaire donc un jugement moins punitif.

le top 1 est simple, il incarne la simplicité.

la gentillesse prime aussi.

## 28.2 Jeux répété

Soit un jeu  $G = \{S, \{u_i\} i = 1, \dots, n\}$  où  $S$  est l'ensemble fini des profils stratégie et  $u_i$  est la fonction d'utilité du joueur  $i$ .

On note  $(G, T)$  le jeu répété obtenu en jouant  $T$  fois le jeu de base  $G$ ,  $(G, \infty)$  correspond à un nombre infini de tour.

On peut également distinguer les jeux répétés un nombre fini, mais indéfini de fois : à chaque tour, il y a une probabilité  $1 - q$  que le jeu s'arrête.

Facteur d'actualisation: Lorsqu'un jeu est répété, il se peut que les gains obtenus à l'itération courante  $u_t$  soient plus/moins importants aux yeux de l'agent que les gains à l'itération suivante  $u_{t+1}$ . Pour modéliser cela on peut utiliser un facteur d'actualisation  $\phi$ :

$$u_t = \phi u_{t+1}$$

Le facteur d'actualisation  $\phi = \frac{u_t}{u_{t+1}}$  représente donc l'attrait du joueur pour les gains actuels.

### 28.2.1 Jeux à deux joueurs à somme nulle

Dans le cas d'un jeu à somme nulle pour chaque case le joueur 1 va gagner la somme indiquée et le joueur 2 va perdre la somme indiquée:

	$y_1$	$y_2$	$y_3$	$y_4$
$x_1$	18	3	0	2
$x_2$	0	3	8	20
$x_3$	5	4	5	5
$x_4$	9	3	0	20

si le joueur 1 prend  $x_1$  et le joueur 2  $y_1$  alors le joueur 1 gagnera 18 et le joueur 2 perdra 18.

Le joueur 1 tente de maximiser son niveau de sécurité:

$$v_x = \max_i(\min_j u(x_i, y_j))$$

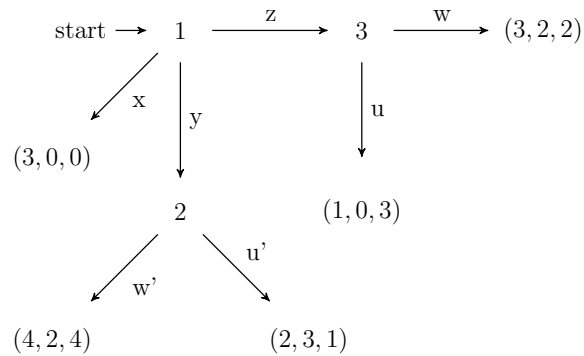
Le joueur 2 tente de minimiser le niveau de sécurité du joueur 1:

$$v_y = \min_j(\max_i u(x_i, y_j))$$

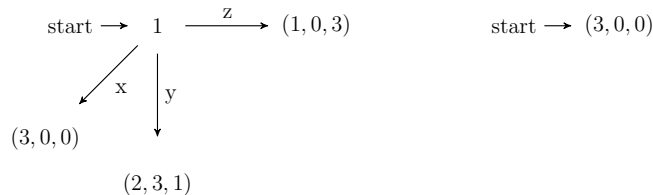
### 28.2.2 Jeu sous forme extensive

Un jeu se déroulant de la racine jusqu'à une feuille, le noeud indique quel joueur doit jouer, donc un ordre de passage est explicite.

Soit le jeu suivant sous forme d'arbre:



Via la récurrence à rebours, On remonte les noeuds optimaux de l'arbre vers la racine:



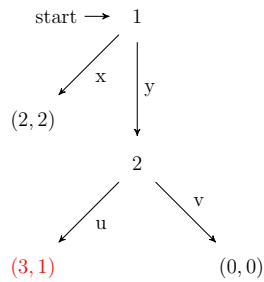
### 28.2.3 sous jeu

Un sous jeu d'un jeu sous forme extensive est un jeu composé d'un noeud (qui est un ensemble d'information singleton) de tous les noeuds successeurs de ce noeud, de tout les arcs reliant ces noeuds, et des unités associées à tout les noeuds terminaux successeurs.

Le grand arbre ci dessus contient 3 sous jeux ayant comme racine (1,2,3).

### 28.2.4 Menaces non crédibles

Voici le même jeu sous forme de tableau et sous forme d'arbre



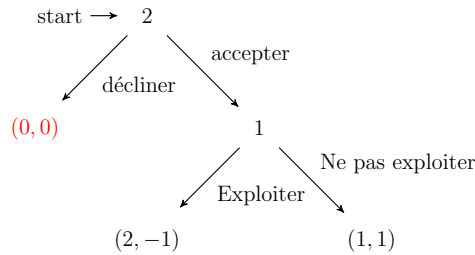
	u	v
x	(2,2)	(2,2)
y	(3,1)	(0,0)

L'équilibre de Nash de l'arbre indique que la meilleure stratégie (d'un point de vue rationnel) est  $(3, 1)$  et via le tableau on obtient  $\{(3, 1), (2, 2)\}$ . l'équilibre de Nash  $(xv)$  n'est pas crédible car il repose sur la menace non-crédible du joueur 2 de jouer  $v$ , autrement dit, dans le premier tour, si le joueur 1 joue  $x$ , le joueur 2 n'a aucun choix à faire, hors dans le tableau le joueur 2 peut proposer un choix.

Un équilibre de Nash d'un jeu sous forme extensive est un équilibre parfait en sous jeu si toute restriction du profil de stratégies à un sous jeu est un équilibre de Nash pour ce sous jeu. Pour les jeux à informations parfaites, la notion d'équilibre parfait en sous jeu coïncide avec la notion de récurrence à rebours.

### 28.2.5 Promesse non crédible

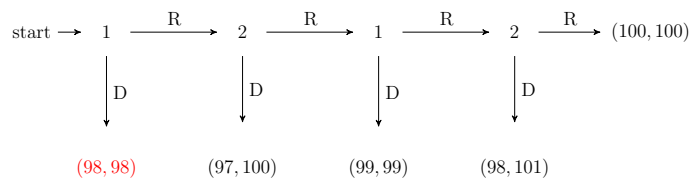
Soit un contrat pour travailler en entreprise, le joueur 2 étant vous et le joueur 1 étant l'entreprise:



Dans le cas d'un contrat non répété il est mieux de choisir  $(0, 0)$ , mais dans le cas d'un jeu répété si vous êtes le premier candidat et que vous acceptez le poste mais que vous vous faites exploiter, alors vous allez attirer de nouveaux candidats pour qu'ils choisissent l'option de ne pas aller travailler là-bas, donc l'entreprise ne devrait pas vous exploiter pour son bien.

### 28.2.6 Limite de la récurrence à rebours

Soit le jeu suivant, vous êtes le joueur 1:



Le calcul de l'équilibre de Nash nous donne  $(98, 98)$  mais cet équilibre ne marche que si le joueur 2 est un robot / rationnel, si on indique que les sommes sont des Millions d'euro, les choix seront bien plus différents, si les deux joueurs sont altruistes alors les deux vont gagner  $(100, 100)$  si l'un des joueurs est mauvais alors celui-ci va aller le plus haut possible et choisir  $D$  juste en guise de dernier choix.



## 28.3 Jeux coopératifs à 2 joueurs

## Chapter 29

# Décision de groupe et théorie du vote

Soit un ensemble de candidats  $X = \{a, b, c, \dots\}$ , un sous ensemble de  $X$  est noté un agenda.

Soit une relation d'ordre  $<, >$  (réflexif, transitif, total) permettent de trier les éléments de  $X$  selon des critères.

Un ensemble d'individu est appelé un profil, chaque individu a le même poids.

On n'aura deux façon de procéder:

Déterminer la relation de préférence (l'ordre des candidats).

Déterminer le gagnant d'un vote. (faisable aussi via la première façon).

## 29.1 Vote entre 2 candidats

$A > B$  si le nombre d'individu qui préfèrent A est plus grande que le nombre d'individu qui préfèrent B.

Quatre propriétés:

**Domaine universel** la méthode de vote donne un résultat quel que soit le profil.

**Anonymat** La méthode de vote traite tout les votant de la même manière.

**Neutralité** La méthode de vote traite tous les candidats de la même manière.

**Monotonie** Si un candidat est élu pour un profil donné, il sera forcément élu pour un profil modifié où ce candidat reçoit plus de vote.

## 29.2 Gagnant de Condorcet

Un gagnant de Condorcet  $A$  et un profil  $N$  si pour tout autre candidat  $B \in N$ , le candidat  $A$  est majoritairement préféré à  $B$ .

$$a >_1 b >_1 c$$

$$\text{Pour } X = \{a, b\} \quad a = 2, b = 1$$

$$a >_2 c >_2 b$$

$$\text{Pour } X = \{a, c\} \quad a = 2, c = 1$$

$$c >_3 b >_3 a$$

$$a \text{ est un gagnant de Condorcet}$$

Pour tout les profil, il existe qu'un gagnant, mais il peut en avoir aucun:

$$a >_1 b >_1 c$$

$$\text{Pour } X = \{a, b\} \quad a = 2, b = 1$$

$$b >_2 c >_2 a$$

$$\text{Pour } X = \{a, c\} \quad a = 1, c = 2$$

$$c >_3 a >_3 b$$

$$\text{Pour } X = \{b, c\} \quad b = 2, c = 1$$

Une méthode de choisir le gagnant de Condorcet quand il existe est appelée Condorcet cohérente.

### 29.3 Scrutin majoritaire simple

Adapter le vote entre 2 candidats, soit 21 votants:

**10:**  $a >_1 b >_1 c$

Résultat: a 10, b 6, c 5

**6:**  $b >_2 c >_2 a$

Le candidat a est élu

**5:**  $c >_3 b >_3 a$

Mais le problème est que plus de vote sont en faveur de  $b$  que de  $a$ , on parle de perdant de Condorcet (ou de vote pour le pire candidat).

### 29.4 Scrutin majoritaire à deux tour

Hors majorité absolue, le vote se fera via un scrutin à 2 tours:

**10:**  $a >_1 b >_1 c$

Premier tour: a 10, b 6, c 5, le candidat c est éliminé

**6:**  $b >_2 c >_2 a$

Second tour : a 10, b 11

**5:**  $c >_3 b >_3 a$

Le candidat b est élu

Le scrutin majoritaire à deux tours est manipulable, il peut être profitable à un individu de mentir sur ses préférences.

Le scrutin majoritaire à deux tours est non-monotone et n'incite pas à la participation.

Le scrutin majoritaire à deux tours n'est pas séparable. l'union de ce même vote via deux circonscriptions différentes ayant comme élu le même candidat ne garantit pas que ce même candidat sera élu lors de la fusion des circonscriptions.

Le scrutin majoritaire à deux tours permet la dictature de la majorité, si dans deux circonscriptions:

51 :  $a > b > c \dots > z$

49 :  $z > b > c \dots > a$

Le candidat a gagne via utilisation de la règle de dictature alors que le candidat b a 100% de vote sur le second choix de préférence.

## 29.5 Méthode de vote non rangées

Soit  $m$  candidats, Une méthode de vote non rangé (nonranking voting rule) est un sous ensemble non vide de  $\{1, 2, \dots, m-1\}$  représentent le nombre de candidats pour lesquels un individu peut voter.

Le candidat ayant le plus de voix est élu.

$\{1\}$  scrutin majoritaire simple

$\{2\}$  Chaque individu doit voter pour 2 candidats

$\{1, 2\}$  Chaque individu peut voter pour 1 ou 2 candidats

$\{m-1\}$  Chaque individu doit voter contre un candidat

$\{1, m-1\}$  Chaque individu doit voter pour un candidat ou contre un candidat

$\{1, 2, m-1\}$  Chaque individu doit voter pour autant de candidats qu'il veut. (vote par approbation)

C'est une méthode de vote la moins manipulable.

## 29.6 Méthode par scorage

Soit  $m$  candidats, une méthode de vote par scorage est défini par:

Soit une séquence non décroissant d'entiers:  $s_0 \leq s_1 \leq \dots s_{m-1}$  tel que  $s_0 < s_{m-1}$ .

Chaque individu doit fournir son ordre strict des candidats

Pour chaque individu on attribut  $s_0$  au dernier candidat,  $s_1$  à l'avant dernier ....

Le candidat ayant reçu le plus de points est élu.

La méthode de vote par scorage respecte les propriétés de Monotonie, Séparabilité, Continuité, Participation.

Le méthode de vote pas scorage n'est pas Condorcet cohérente.

## 29.7 Méthode de Condorcet cohérentes

### 29.7.1 Règle de Copeland

Le meilleur Candidat est celui qui bat tout les autres candidats:

Pour chaque candidats  $a$  lui attribuer la valeur  $+1$  si un candidat  $b$  ( $b \neq a$ ) est préféré à  $b$  -1 sinon.

Le candidat élu aura le plus le plus haut score de Copeland.

$$\mathbf{5:} \quad a > b > c > d$$

$$\mathbf{4:} \quad b > c > d > a$$

$$\mathbf{3:} \quad d > c > a > b$$

$$\begin{array}{ll} (a, b) = (+1, -1) & (b, c) = (+1, -1) \\ (a, c) = (-1, +1) & (b, d) = (+1, -1) \\ (a, d) = (-1, +1) & (c, d) = (+1, -1) \end{array}$$

$$\begin{array}{l} \text{cop}(a) = \left| \begin{array}{ccc} +1 & -1 & -1 \\ -1 & & +1 & +1 \\ & +1 & -1 & +1 \\ & & +1 & -1 & -1 \end{array} \right| = -1 \\ \text{cop}(b) = \left| \begin{array}{ccc} +1 & -1 & -1 \\ -1 & & +1 & +1 \\ & +1 & -1 & +1 \\ & & +1 & -1 & -1 \end{array} \right| = +1 \\ \text{cop}(c) = \left| \begin{array}{ccc} +1 & -1 & -1 \\ -1 & & +1 & +1 \\ & +1 & -1 & +1 \\ & & +1 & -1 & -1 \end{array} \right| = +1 \\ \text{cop}(d) = \left| \begin{array}{ccc} +1 & -1 & -1 \\ -1 & & +1 & +1 \\ & +1 & -1 & +1 \\ & & +1 & -1 & -1 \end{array} \right| = -1 \end{array}$$

Les candidats  $b$  et  $c$  sont élus.

### 29.7.2 Règle de Kramer Simpson

Le meilleur candidat est celui qui engendra le moins de regret chez les votants: Pour chaque candidat  $a \neq b$  calculer  $N(a, b)$  le nombre d'individus préférant  $a$  à  $b$ .

Le score de Simpson est le minimum des  $N(a, b)$ , le candidat élu est celui qui a le plus de haut score de Simpson.

$$\mathbf{5:} \quad a > b > c > d$$

$$\mathbf{4:} \quad b > c > d > a$$

$$\mathbf{3:} \quad d > c > a > b$$

$$\begin{array}{ll} (a, b) = (8, 4) & (b, c) = (9, 3) \\ (a, c) = (5, 7) & (b, d) = (9, 3) \\ (a, d) = (5, 7) & (c, d) = (9, 3) \end{array}$$

$$\begin{array}{l} cop(a) = \left| \begin{array}{ccc} 8 & 5 & 5 \\ & & \end{array} \right| = 5 \\ cop(b) = \left| \begin{array}{ccc} 4 & & 9 \quad 9 \\ & & \end{array} \right| = 4 \\ cop(c) = \left| \begin{array}{ccc} & 7 & 3 \quad 9 \\ & & \end{array} \right| = 3 \\ cop(d) = \left| \begin{array}{ccc} & & 7 \quad 3 \quad 3 \\ & & \end{array} \right| = 3 \end{array}$$

Le candidat  $a$  est élu.

### 29.7.3 Règle du Tie break

En cas d'égalité, une règle annexe peut s'appliquer pour départager les élus. Lors des présidentielle on élit le candidat le plus vieux.

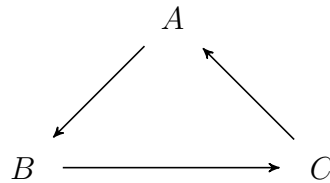
## 29.8 Graphe de majorité

La représentation graphique n'est pas transitif:

**5:**  $a > b > c > d$

**4:**  $b > c > d > a$

**3:**  $d > c > a > b$



Un gagnant de Condorcet est un candidat qui a un arc vers tous les autres candidats.

**Copeland** : Le meilleur candidat est celui qui requiert le moins de suppression d'autre candidats pour devenir un gagnant de Condorcet.

**Young** : Le meilleur candidat est celui qui requiert le moins de suppression d'individus pour devenir un gagnant de Condorcet.

**Dodgson** : Le meilleur candidat est celui qui demande le moins de changement dans les préférences des individus.

**Kramer-Simpson** : Le meilleur candidat est celui qui assure le moins de regret.

**Kemeny** : Le meilleur candidat est celui qui est rangé le plus haut dans la hiérarchie, une hiérarchie est construite via la relation de préférence  $>$  la plus proche du profil.

Les règles de Copeland et de Kramer-Simpson sont monotones.

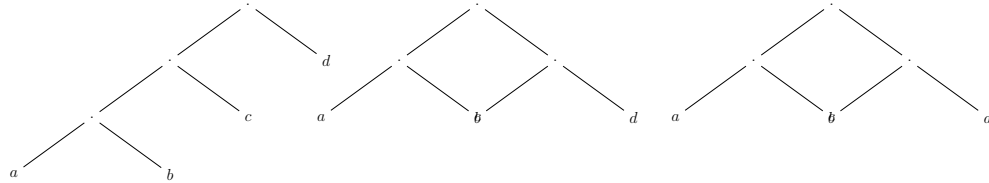
Aucune règle Condorcet cohérente ne peut satisfaire la séparabilité.

Aucune règle Condorcet cohérente ne peut satisfaire Participation.



## 29.9 Vote par comparaison successives

Le résultat dépend de l'ordre des comparaisons:



## 29.10 Vote simple transférable: méthode de Coombs

Chaque individu indique son ordre de préférence  $>_i$ . Pour  $n$  candidats, on fait  $n - 1$  tours (à moins d'avoir la majorité strict sur un candidat), à chaque tours on élimine le candidat qui a le moins de points. Si un candidat est éliminé, alors il sera supprimé de toutes les listes de préférences des individus.

# Part VIII

## Apprentissage

## Chapter 30

# Approche d'apprentissage par la logique

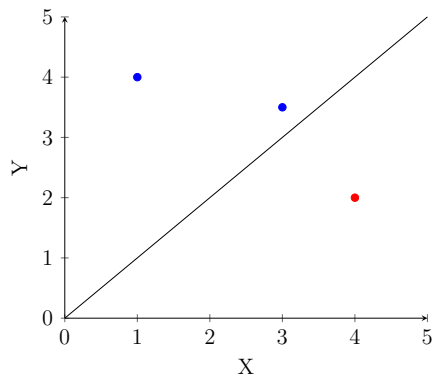
Une approche simple concernant l'apprentissage de problèmes dont le domaine de sortie est boolean serait de passer par la logique classique pour pouvoir simplifier la compréhension du problème.

## 30.1 Espace de Version

Pour un problème suivant:

A	B	C	D	accaptable?
1	1	1	0	oui
1	1	0	1	oui
0	1	1	1	non

D'où il suffirait d'une fonction donnant dans le domaine Boolean, associer un algorithme de classification simple:



Ayant comme points de couleurs *Rouge* les points donnant la valeur de vérité False et les points de couleurs *Blue* les point donnant la valeur de vérité True.

Mais ce ne serait pas donner un gros mode de résolution à un problème qui peut être simplifié?

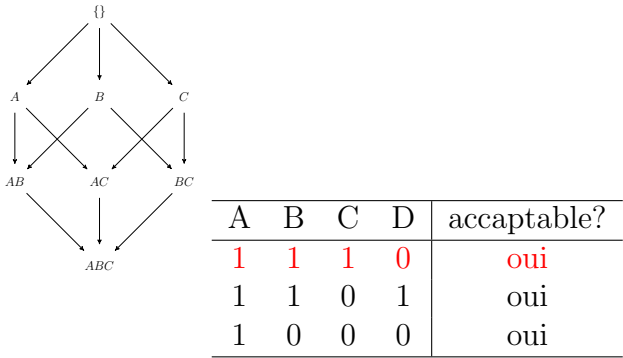
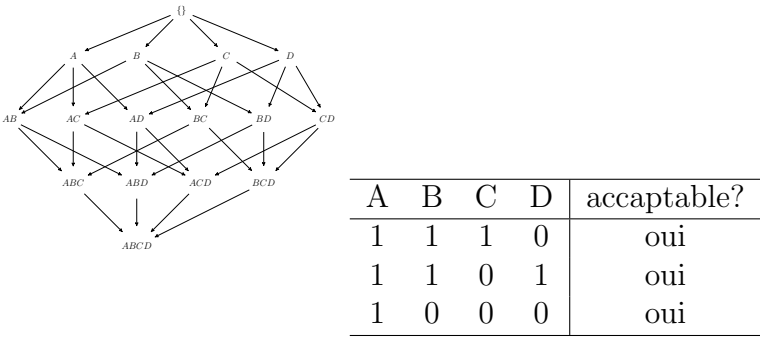
Pour les cas suivants:

- Faciliter la compréhension du problème
- Comprendre pourquoi une décision donné pour une entrée

30.1.1 convergence des données

Dans le tableau d'acceptation on peut transformé la règle 3 en son dual via la Lois de De Morgan:

Et via un treillis de donnée pour chaque entré positif on peut compter le nombre d'occurrence de motif en faveur de l'acceptation de la ligne:



A	B	C	D	acceptable?
1	1	1	0	oui
1	1	0	1	oui
1	0	0	0	oui

A	B	C	D	acceptable?
1	1	1	0	oui
1	1	0	1	oui
1	0	0	0	oui

Par itération et réduction du treillis on sait que  $A$  est un attribut très discriminant, qui fait revenir le problème à seulement la valeur de  $A$ .

# Chapter 31

## Apprentissage statistique

Dans ce chapitre nous nous intéressons à des fonctions  $h \in H$  à qui pour une liste  $X$  à  $d$  dimension de domaine réelle associe un label  $y$  dans le domaine  $[-1, +1]$ . Un  $x \in X$  peut être une couleur, un réel, une chose négatif ou encore une mesure quelconque.

## 31.1 Classification binaire réalisable

Chaque entrée  $x \in X$  est tirée aléatoirement et indépendamment selon une distribution de probabilité  $d$  qui est fixée mais inconnue de l'apprenant. Chaque sortie  $y \in Y$  est calculé via la fonction cible  $h^* \in H$  qui est inconnue de l'apprenant.

### 31.1.1 Erreur de généralisation et d'entraînement

La performance d'une hypothèse  $h \in H$  est calculé par le nombre d'erreurs que la fonction peut commettre en probabilité selon  $d$ :

$$l_d(h) = P_{x \sim d}[h(x) \neq h^*(x)]$$

En pratique, l'apprenant n'a accès qu'à une petite partie nommée  $S \subset X$  (qui peut contenir des doublons) dont les éléments sont générés aléatoirement via  $d$ . Le risque empirique de  $h$  par rapport à  $S$  est donné par :

$$l_s(h) = \frac{1}{|S|} |\{x \in S : h(x) \neq h^*(x)\}|$$

Le nombre d'erreur moyen que fait  $h$  sur  $S$

### 31.1.2 Processus d'apprentissage

Le processus d'apprentissage n'est pas si différent que dans la première partie du Memo:

Soit une distribution  $d$ , chaque requête vers  $d$  va choisir un échantillon aléatoirement pour créer un ensemble  $S$  qui va servir à faire apprendre  $h$  lors de la phase d'apprentissage, tester lors de la phase de test et retenir les erreurs vues par la fonction d'analyse.

### 31.1.3 Incertitude de l'apprentissage

Il existe deux mesures de l'incertitude en apprentissage statistique

Paramètre de confiance qui donne la qualité de l'échantillonnage

Paramètre d'erreur qui donne un indice sur les bonnes prédictions futures



### 31.1.4 Modèle PAC réalisable

Une classe d'hypothèses  $H$  est dite PAC (probability approximately correct) s'il existe une fonction  $\{0, 1\}^2 \rightarrow \{0, 1, 2, \dots\}$  telle que pour toute paire  $(\phi(\text{confiance}), \psi(\text{erreur}))$  pour toute distribution  $d$  sur  $X$  et toute fonction cible  $h^* \in H$ :

Après avoir observé un échantillon  $S$  de  $X$  tiré aléatoirement selon  $d$ , et de taille au moins  $m(\phi, \psi)$ .

L'apprenant retourne une hypothèse  $h \in H$ , telle qu'avec une probabilité au moins  $1 - \phi$ , l'erreur de généralisation  $l_d(h)$  est d'au plus  $\psi$ .

## 31.2 Classes d'hypothèses finies

Supposons  $X = [0, 1]^d$

Toutes fonction  $h : [0, 1]^d \rightarrow [0, 1]$  est appelée fonction booléenne.

Une classe d'hypothèses booléennes est un sous ensemble  $H$  de  $[0, 1]^d \rightarrow [0, 1]$ .

### 31.2.1 Minimisation des erreurs empirique

Le principe est de trouver dans  $H$  l'hypothèse qui fait le moins d'erreurs sur l'échantillon  $S$ :

$$h_S \in \operatorname{argmin}_{h \in H} L_S(h)$$

### 31.2.2 Théorème de PAC des classes finies

Toutes classe d'hypothèse  $H$  finie est PAC-apprenable avec une complexité d'échantillonnage

$$m(\phi, \psi) \leq \frac{\ln(|H|/\phi)}{\psi}$$

## 31.3 Apprentissage agnostique

### 31.3.1 Principe de minimisation de risque empirique

## 31.4 VC Dimension

### 31.4.1 Fonctions linéaires

### 31.4.2 Fonctions de croissance et VC-Dimension

### 31.4.3 Théorème fondamental de l'apprentissage statistique

Une version Qualitative:

Soit  $H$  une classe de fonctions  $X$  dans  $\{0, 1\}$  mesurée selon la fonction de perte discrète (nombre d'erreurs), Alors les propriétés suivantes sont équivalentes:

$H$  est agnostiquement PAC apprenable

$H$  a la propriété de convergence uniforme

$H$  a une VC-dimension finie

Une version Quantitative:

Soit  $H$  une classe de fonctions de  $X$  dans  $\{0, 1\}$ , mesurée selon la fonction de perte discrète (nombre d'erreurs). Supposons que  $VCdim(H) = D < \infty$ . Alors il existe une constante  $C$  telle que  $H$  est agnostiquement PAC apprennable avec une complexité d'échantillonnage d'au plus:

$$m(\alpha, \epsilon) \leq C \left( \frac{D + \ln(\frac{1}{\alpha})}{\epsilon^2} \right)$$

### 31.4.4 VC-Dimensions Utiles

## Chapter 32

# Apprentissage Online

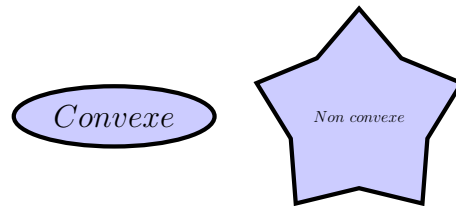
L'apprentissage online est un jeu à somme nulle répétitif à deux joueur (théorème minmax ou équilibre de Nash), les joueurs sont l'environnement et le joueur.

L'apprenant reçoit une observation de l'environnement et donne une prédiction, et l'environnement va donner la vérification sur la prédiction.

## 32.1 Analyse convexe

### 32.1.1 Combinaison convexe

Une forme convexe est une forme pour qui n'importe quel droite dont les 2 points font partie de la forme est dans la forme.

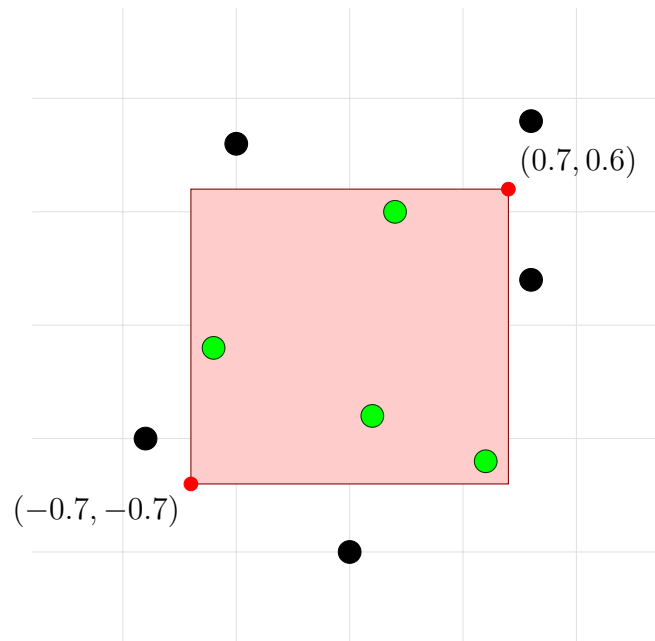


### 32.1.2 Enveloppe convexe

L'enveloppe convexe d'une forme est la bordure



Si un espace d'application est convexe alors le problème est polynomial, sinon le problème est NP-complet.



Dans cette figure on calcule l'enveloppe en essayant de capturer tout les points vert, un carré de coordonné  $(-0.7, -0.7)(0.7, 0.6)$  peut largement faire le boulot et résoudre le problème de classification.

Le calcul est polynomial (enfin même inférieur) car le calcul de la classification d'un point se fait en un 4 instructions simple:

$$classification = \text{lambda } x, y : -0.7 < x < 0.7 \text{ and } -0.7 < y < 0.6$$

Ce raisonnement est valide car il n'y a aucun point noir dans le rectangle rouge.

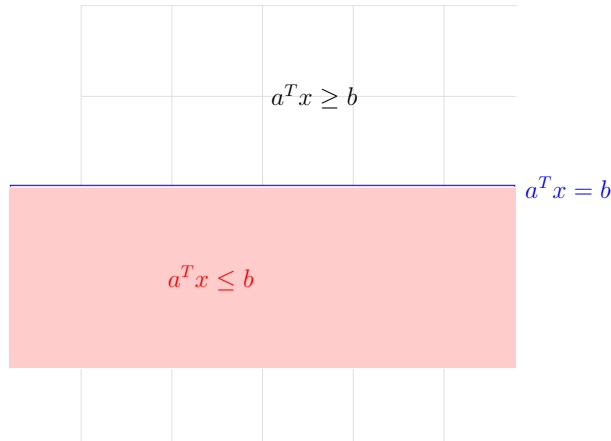
### 32.1.3 Theoreme de la séparation des hyperplan

Un hyperplan est un ensemble de points sous la forme (représenté en blue):

$$H = \{x | a^T x = b\}, a \neq 0 \text{ and } b \in R$$

Un hyperespace est un ensemble de points sous le forme (représenté en rouge):

$$H = \{x | a^T x \leq b\}, a \neq 0 \text{ and } b \in R$$

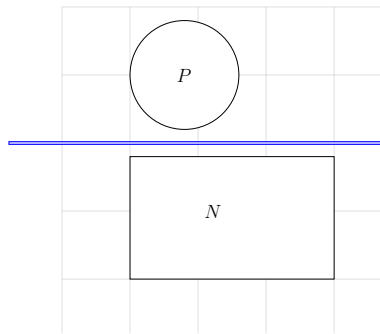


Soit  $P$  et  $N$  deux ensemble de points connexe tel qu'il n'existe aucune intersection, alors il existe une  $a$  ( $\neq 0$ ) et un  $b$  tel que:

$$a^T x < b \forall x \in P \text{ and}$$

$$a^T x \geq b \forall x \in N$$

L'ensemble des  $\{x | a^T x = b\}$  est l'hyperplan de séparation de  $P$  et  $N$  (et aussi une fonction linéaire de classification).



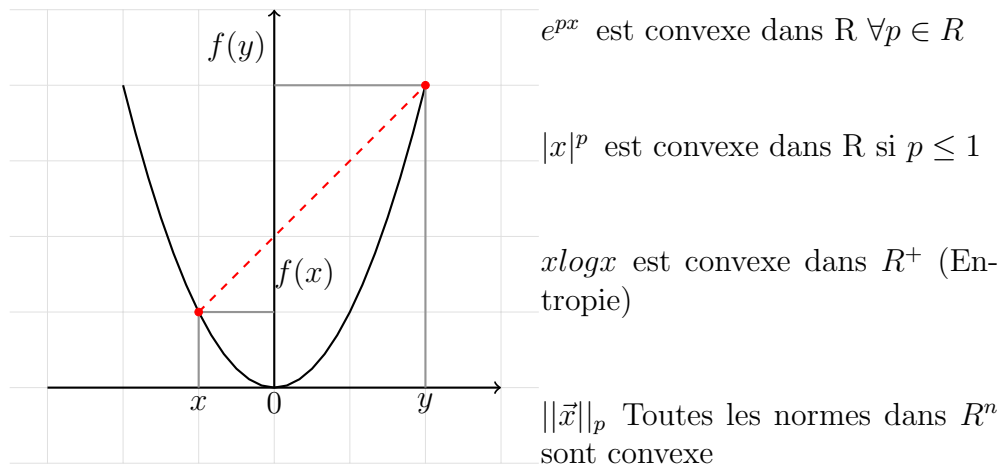
### 32.1.4 Fonction convexe et normes

Une fonction  $f$  est dite convexe si son domaine est un ensemble convexe:

$$\forall x, y \in \text{dom}(f), \theta \in [0, 1]$$

$$f(\theta x + (1 - \theta)x) \leq \theta f(x) + (1 - \theta)f(y)$$

Autrement dit (via la droite rouge), tout le domaine de la fonction est à l'intérieur de l'ensemble convexe de la fonction parabolique:



La *norme 1* est donné par la somme des coefficient sous forme absolu, on dit aussi le calcul de la distance de Manhattan:

$$\|\vec{x}\|_1 = |x_1| + \dots |x_n|$$

La *norme 2* est obtenu par le produit scalaire, elle est aussi utilisé pour calculer la distance entre 2 points dans un espace vectoriel:

$$\|\vec{x}\|_2 = \sqrt{|x_1|^2 + \dots |x_n|^2}$$

La *norme p* est une généralisation de l'espace de fonction, si on prend  $p = 2$  on se réduit au produit scalaire:

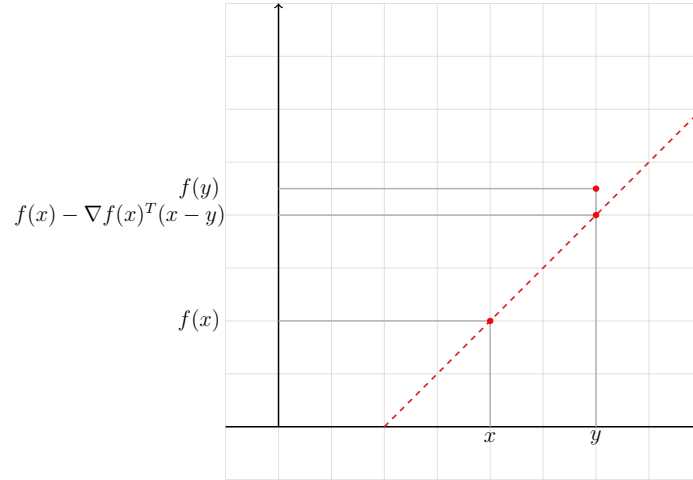
$$\|\vec{x}\|_p = (|x_1|^p + \dots |x_n|^p)^{\frac{1}{p}}$$

La *norme  $\infty$* :

$$\|\vec{x}\|_\infty = \max(|x_1|, \dots |x_n|)$$

### 32.1.5 Gradient

Soit  $f : R^n \rightarrow R$  et  $x$  un point intérieur du domaine de  $f$ , Si on peut créer une tangente  $\frac{\partial f(x)}{\partial x_i}$  (en rouge), on peut dire que  $f$  est différentiable (une dérivée existe) passant par  $x$ :



Si  $f : R^n \rightarrow R$  est différentiable, alors  $f$  est convexe si et seulement si le domaine de  $f$  est convexe et  $\forall x, y \in f$ :

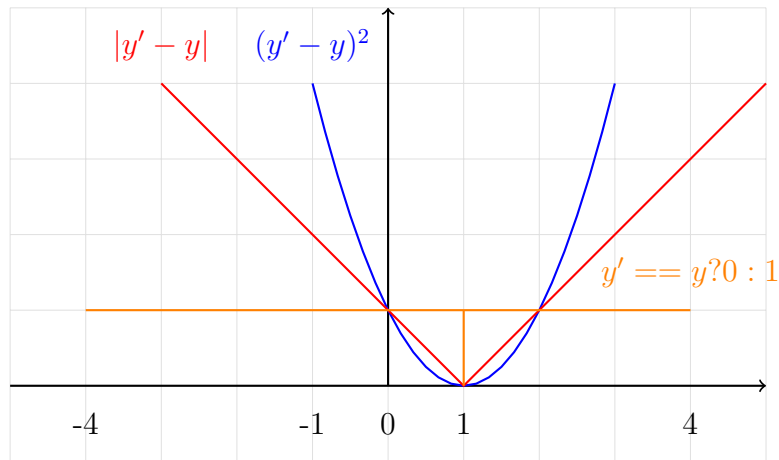
$$f(x) - f(y) \leq \nabla f(x)^T(x - y)$$



### 32.1.6 Fonction de Perte

Une fonction de perte est une fonction de  $R \times R \rightarrow R^+$  tel que pour un couple  $(y', y)$  d'une prédiction et d'une vraie réponse, comparer les deux valeurs et retourner si elle sont égal.

$$f(y', y) \rightarrow y' == y$$



La formule orange nommé *Zero – one* qui n'est pas convexe et qui est la première au quel on pense:

$$f(y', y) \rightarrow y' == y ? 0 : 1$$

La formule rouge nommé *absolut* qui est convexe mais trop punitif pour les petites erreurs:

$$f(y', y) \rightarrow |y' - y|$$

La formule bleu nommé *quadratique* qui est convexe et minimise les petits erreurs mais maximise les grosses erreurs:

$$f(y', y) \rightarrow |y' - y|^2$$

## 32.2 Apprentissage par régression

La régression est un problème de minimisation:

**Paramètre** : Une fonction de mise à jour  $f$

**Initialisation**  $w_0 = 0$

**Pour toutes les entrées** :

- Recevoir l'observation  $x_t$
- Faire une prédiction  $y' = w_{t-1}^T x_i$
- Calculer la fonction de perte  $(y', y)$
- Re apprendre le modèle en cas de mauvaise prédiction  $w_t = f(w_{t-1})$

### 32.2.1 Gradient Descent

**Initialisation**  $w_0 = 0$

**Pour toutes les entrées** :

- Recevoir l'observation  $x_t$
- Faire une prédiction  $y' = w_{t-1}^T x_i$
- Calculer la fonction de perte  $(y'_t, y_t)^2$
- Re apprendre le modèle  $w_t = w_{t-1} - \eta(y'_t - y_t)x_t$

## 32.3 Apprentissage par classification

## Part IX

# Problème de satisfaction de contraintes CSP

## Chapter 33

### Introduction et modèles

Une Solution d'un problème CSP est une assignation d'une valeur à chaque variables de P tel que toutes les contraintes de P soit satisfaite.

### 33.1 exemple simple

Trouver une assignation Minimal et Maximal pour chaque variables de P dans le problème suivant:

$$\mathbf{vars(P)} = w, x, y, z$$

$$\mathbf{Dom(P)}$$

$$\mathbf{Dom(w,x)} = \{1, 2, 3\}$$

$$\mathbf{Dom(y,z)} = range(4)$$

**Contraintes**

$$w = x$$

$$x \leq y + 1$$

$$y > z$$

$$(x, z) \in \{(1, 2), (2, 1), (2, 4), (3, 3)\}$$

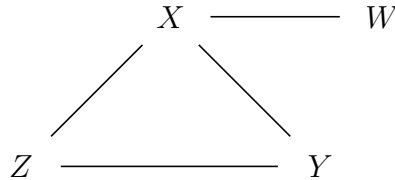
Une solution serait:

$$\mathbf{Minimal} \quad w = x = 1, y = 3, z = 2$$

$$\mathbf{Maximal} \quad w = x = z = 3, y = 4$$

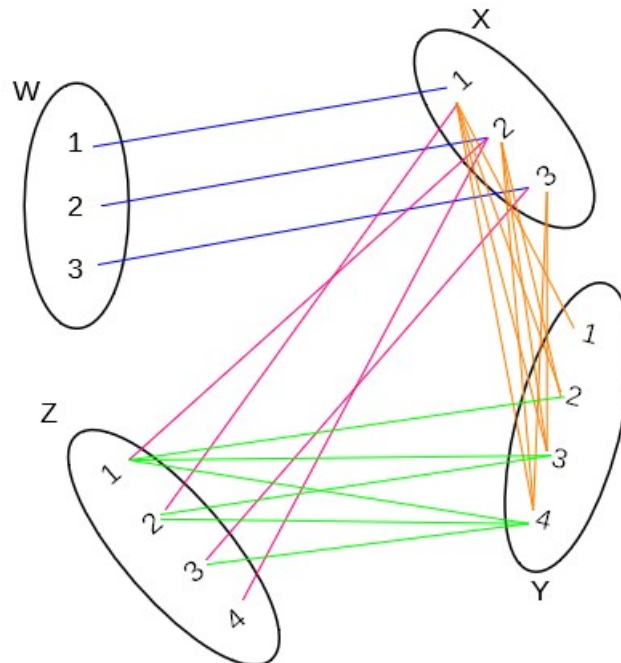
### 33.1.1 Graphe de contraintes

Chaque variable est un nœud et chaque contraintes est représenté par une arête entre les variables concerné.



### 33.1.2 Graphe de compatibilité

Représenter toutes variables avec des ensemble contenant toutes les valeur de leurs domaine puis relier chaque éléments de l'ensemble avec un autre tel que les contraintes ne soit pas violet:



# Chapter 34

## Filtrage



## 34.1 Filtrage du domaine via les contraintes

Un problème peut être défini comme une intersection de sous problème à qui un sous problème est représenté par un filtre du domaine des variable et une contrainte qui va réduire le domaine des variable à un état de consistance peu importe les valeurs des variables.

**Arc Consistency (AC)** Tous les valeurs inconsistant sont identifié et retiré

**Bound Consistency (BC)** Les valeurs inconsistant sont les bornes des domaine et ils sont identifié et retiré

### 34.1.1 Exemple

Contrainte  $w + 3 = z$

Avec

$$dom(w) = \{1, 3, 4, 5\}$$

$$dom(z) = \{4, 5, 8\}$$

Avec un filtre AC:

$$dom(w) = \{1, 5\} \text{ et } dom(z) = \{4, 8\}$$

Avec un filtre BC:

$$dom(w) = \{1, 3, 4, 5\} \text{ et } dom(z) = \{4, 5, 8\}$$

## 34.2 Notion de Support

Soit  $c_{xyz}$  une contrainte tel que  $c_{xyz}$  égal:

$$dom(x, y) = \{a, b\} \text{ et } dom(z) = \{b, c\}$$

On n'a  $T = rel(c_{xyz})$  et  $V = dom(x) \times dom(y) \times dom(z)$ :

$(z, b)$  a un support mais  $(z, c)$  n'en n'a pas

T		
a	a	a
<i>a</i>	<i>b</i>	<i>b</i>
<i>a</i>	<i>c</i>	<i>c</i>
b	a	a
b	b	b
c	a	a
<i>c</i>	<i>c</i>	<i>c</i>

V		
a	a	b
<i>a</i>	<i>a</i>	<i>c</i>
<i>a</i>	<i>b</i>	<i>b</i>
<i>a</i>	<i>b</i>	<i>c</i>
b	a	b
<i>b</i>	<i>a</i>	<i>c</i>
b	b	b
<i>b</i>	<i>b</i>	<i>c</i>

### 34.3 AC filtre AllDifferent

Soit un sudoku block avec la contrainte *allDifferent*( $w, x, y, z$ ):

3	w	6
x	8	y
1	4	z

$$\mathbf{x} = \{5, 7, 9\}$$

$$\mathbf{y} = \{7\}$$

$$\mathbf{w} = \{2, 5\}$$

$$\mathbf{z} = \{2, 5\}$$

Dans ce cas,  $y$  contient un singleton, donc il sera trivialement résolu en affectant  $y = 7$  et en supprimant tout les occurrences de 7 dans les autres variables:

3	w	6
x	8	7
1	4	z

$$\mathbf{x} = \{5, 7, 9\}$$

$$\mathbf{w} = \{2, 5\}$$

$$\mathbf{z} = \{2, 5\}$$

nous pouvons construire le Hall sets suivant:  $(w, z)vs(2, 5)$  et donc supprimer toutes les occurrences de 2 et 5 dans les autres variables:

3	w	6
9	8	7
1	4	z

$$\mathbf{x} = \{5, 9\}$$

$$\mathbf{w} = \{2, 5\}$$

$$\mathbf{z} = \{2, 5\}$$

Ce qui nous donne 2 solutions,  $(w=2, z=5)$  ou  $(w=5, z=2)$

3	w	6
9	8	7
1	4	z

$$\mathbf{w} = \{2, 5\}$$

$$\mathbf{z} = \{2, 5\}$$

Soit un autre exemple plus complexe avec une contrainte *allDifferent*( $u, v, w, x, y, z$ ) et un domaine  $\{1, 2, 5, 6, 7, 9\}$ :

$$\begin{array}{ll} \mathbf{u} = \{1, 2\} & \mathbf{x} = \{2, 6\} \\ \mathbf{v} = \{2, 9\} & \mathbf{y} = \{6, 7, 9\} \\ \mathbf{w} = \{1, 5, 6\} & \mathbf{z} = \{1, 9\} \end{array}$$

On peut créer une Hall sets de taille 3 ( $u, v, z$ ) = (1, 2, 9) et éliminer toutes les occurrences de 1, 2, 9 dans les autres variables:

$$\begin{array}{ll} \mathbf{u} = \{1, 2\} & \mathbf{x} = \{6\} \\ \mathbf{v} = \{2, 9\} & \mathbf{y} = \{6, 7\} \\ \mathbf{w} = \{5, 6\} & \mathbf{z} = \{1, 9\} \end{array}$$

On peut affecter x à 6 et supprimer toutes les occurrences de 6 dans les autres variables:

$$\begin{array}{ll} \mathbf{u} = \{1, 2\} & \mathbf{x} = 6 \\ \mathbf{v} = \{2, 9\} & \mathbf{y} = 7 \\ \mathbf{w} = 5 & \mathbf{z} = \{1, 9\} \end{array}$$

On obtient donc la tableau suivant qui pour n'importe quelle valeurs de  $u, v, w, x, y, z$  donne:

u	v	w	x	y	z
1	2	5	6	7	1
2	9				9

## 34.4 AC filtre Cardinalité

Une contrainte de cardinalité note  $cardinality(X, Y, L, U)$  qui a  $X$  est l'ensemble des variables,  $Y$  le domaine des variables de  $X$ ,  $(L_i, U_i)$  le range d'occurrences de la variable  $Y_i$  dans tout le modèle.

Si  $(L_i = 1, U_i = 3)$  alors la variable  $Y_i$  ne pourra être présent de 1 à 3 fois maximum, si  $(L_i = 2, U_i = 2)$  alors la variable  $Y_i$  devra être présent 2 fois.

Prenons:

**X** les agents {Peter,Paul,Mary,John,Bob,Mike,Julia}

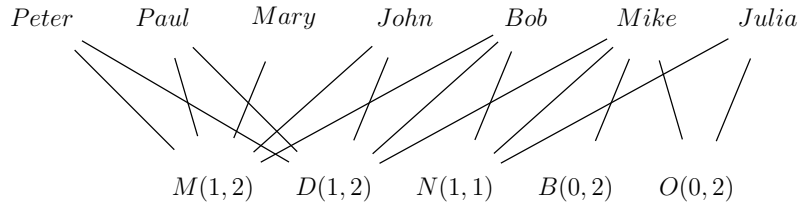
**Y** les activités {Morning (M),Day (D),Night (N),Backup (B),Off (O)}

**L** {1,1,1,0,0}

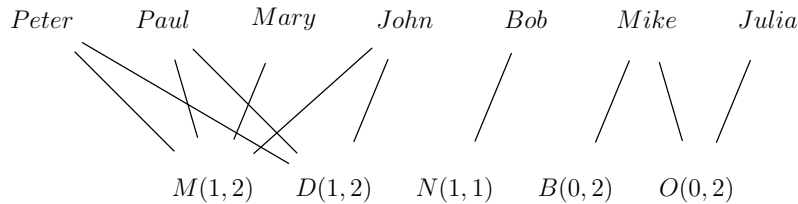
**U** {2,2,1,2,2}

Nous voudrions réduire ce genre de tableau suivant:

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Peter	D	N	N	N	O	O	O
Paul	O	O	D	D	M	M	B
Mary	M	M	D	D	O	O	N



On peut assigner *Bob* à la tâche *N* et assigner à *Julia* la tâche *O*, le reste sera des combinaison possible:



### 34.5 AC filtre Sum à une borne

Soit un filtre *sum* avec les variables  $(x, y, z, a, b, c)$  dans le domaine  $range(1, 5)$  appliqué sur l'équation:

$$2x + 3y + 2z + a + 4b + 2c \geq 50$$

x	y	z	a	b	c
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4

Si la somme des maximums de chaque variable ne respecte pas la contrainte, alors il n'existe aucun assignement des variables tel que la contrainte serait satisfaite:

$$2 * 4 + 3 * 4 + 2 * 4 + 4 + 4 * 4 + 2 * 4 = 8 + 12 + 8 + 4 + 16 + 8 = 56 \geq 50$$

Pour chaque variables, enlever son impacte dans le résultat précédemment obtenue et refaire l'opération de dépilé avec les différents  $x$  tant que un  $x_i$  ne satisfait pas la contrainte:

Pour x:

$$2 * 4 + 3 * 4 + 2 * 4 + 4 + 4 * 4 + 2 * 4 = 8 + 12 + 8 + 4 + 16 + 8 = 56 \geq 50$$

$$56 - 2 * 4 + 2 * 1 = 50 \geq 50, \text{ donc } x = 1 \text{ fonctionne}$$

(pareil pour z,c)

Pour y:

$$56 - 3 * 4 + 3 * 1 = 44 \geq 50, \text{ donc } y = 1 \text{ ne fonctionne pas}$$

$$56 - 3 * 4 + 3 * 2 = 50 \geq 50, \text{ donc } y = 2 \text{ fonctionne}$$

Pour  $b$ :

$$56 - 4 * 4 + 4 * 1 = 40 \geq 50, \text{ donc } b = 1 \text{ ne fonctionne pas}$$

$$56 - 4 * 4 + 4 * 2 = 44 \geq 50, \text{ donc } b = 2 \text{ ne fonctionne pas}$$

$$56 - 4 * 4 + 4 * 3 = 50 \geq 50, \text{ donc } b = 3 \text{ fonctionne}$$

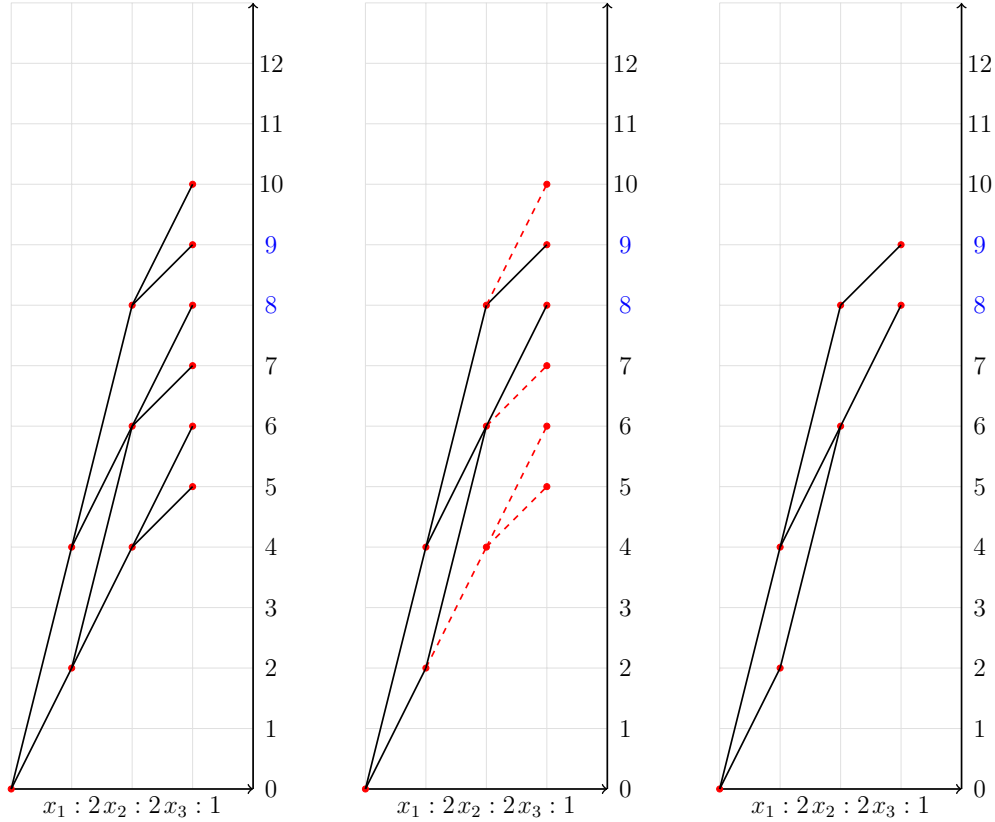
On obtient:

x	y	z	a	b	c
1		1	1		1
2	2	2	2		2
3	3	3	3	3	3
4	4	4	4	4	4

## 34.6 AC filtre avec Sum à deux bornes

Soit  $x, y, z$  avec comme domaine  $dom(x) = dom(y) = dom(z) = 1, 2$  et avec une contrainte  $8 \leq 2x + 2y + z \leq 9$ .

Via résolution graphique on obtient et réduit le graphe comme ceci:



Dans un premier temps on représente toutes les possibilités, dans un second temps on élimine récursivement tout les feuille de l'arbre qui pointe sur des noeuds hors objectif (dans notre cas hors de 8,9). Toutes les branches seul qui ne sont pas relié aux bornes 8 et 9 seront supprimé.

On peut attribuer au variables toutes les possibilités de chemin de l'arbre:

$$x_1 = 1, x_2 = 2, x_3 = 1 : 1 * 2 + 2 * 2 + 1 * 2 = 8$$

$$x_1 = 2, x_2 = 2, x_3 = 1 : 2 * 2 + 2 * 2 + 1 * 1 = 9$$



## Part X

# Problème de satisfaction SAT

# Chapter 35

## définitions de base

## 35.1 Transformation NNF, CNF

Une forme NNF (Negative Normal Forme) est une formule donnée avec uniquement les connecteurs logiques  $\wedge \vee \neg$ .

en remplaçant les  $\rightarrow$  et  $\leftrightarrow$ :

$$\phi \rightarrow \psi \text{ donne } \neg\phi \vee \psi$$

$$\phi \leftrightarrow \psi \text{ donne } (\neg\phi \vee \psi) \wedge (\phi \vee \neg\psi)$$

descendre les négations au niveau atomique:

$$\neg(\phi \wedge \psi) \text{ donne } \neg\phi \vee \neg\psi$$

$$\neg(\phi \vee \psi) \text{ donne } \neg\phi \wedge \neg\psi$$

$$\neg\neg\phi \text{ donne } \phi$$

Une forme CNF (Normal Conjunctive Forme) est une conjonction de disjonctions de littéraux:

$$\textbf{exemple} : (\neg A \vee B) \wedge (\neg C \vee B \vee D) \wedge (A \vee B)$$

### 35.1.1 Transformation glouton

Toutes formules peut être réduite à CNF en appliquant récursivement la loi de DeMorgan:

$$(\phi \wedge \psi) \vee \gamma \text{ donne } (\phi \vee \gamma) \wedge (\psi \vee \gamma)$$

Mais rarement utilisé car la complexité est exponentielle dans le pire des cas.

### 35.1.2 Transformation via ajout de variables

Soit la formule suivante:

$$\neg((\neg(a \vee b)) \leftrightarrow (c \rightarrow d)) \rightarrow ((e1 \wedge e2 \wedge e3) \vee (f1 \wedge f2 \wedge f3) \vee (g1 \wedge g2 \wedge g3))$$

réduire en NNF:

$$((a \vee b \vee \neg c \vee d) \wedge ((c \wedge \neg d) \vee (\neg a \wedge \neg b)) \vee ((e1 \wedge e2 \wedge e3) \vee (f1 \wedge f2 \wedge f3) \vee (g1 \wedge g2 \wedge g3))$$

Appliquer la formule:

$$((a \vee b \vee \neg c \vee d) \wedge ((c \wedge \neg d) \vee (\neg a \wedge \neg b)) \vee ((e1 \wedge e2 \wedge e3) \vee (f1 \wedge f2 \wedge f3) \vee (g1 \wedge g2 \wedge g3))$$

$$i \leftrightarrow (c \wedge \neg d)$$

$$j \leftrightarrow (\neg a \wedge \neg b)$$

$$k \leftrightarrow (e1 \wedge e2 \wedge e3)$$

$$l \leftrightarrow (f1 \wedge f2 \wedge f3)$$

$$m \leftrightarrow (g1 \wedge g2 \wedge g3)$$

donne:

$$((a \vee b \vee \neg c \vee d) \wedge (i \vee j) \vee (k \vee l \vee m))$$

$$n \leftrightarrow (a \vee b \vee \neg c \vee d) \wedge (i \vee j)$$

ce qui donne:

$$(n \vee k \vee l \vee m)$$

Après distribution des nouvelles variables:

$$\begin{aligned} & (n \vee k \vee l \vee m) \wedge \\ & i \leftrightarrow (c \wedge \neg d) \wedge \\ & j \leftrightarrow (\neg a \wedge \neg b) \wedge \\ & k \leftrightarrow (e1 \wedge e2 \wedge e3) \wedge \\ & l \leftrightarrow (f1 \wedge f2 \wedge f3) \wedge \\ & m \leftrightarrow (g1 \wedge g2 \wedge g3) \wedge \\ & n \leftrightarrow (a \vee b \vee \neg c \vee d) \wedge (i \vee j) \end{aligned}$$

donne la formule CNF suivante:

$$\begin{aligned} & ((n \vee k \vee l \vee m) \wedge (\neg i \vee c) \wedge (\neg i \vee \neg d) \wedge \\ & (\neg j \vee \neg a) \wedge (\neg j \vee \neg b) \wedge (\neg k \vee e1) \wedge (\neg k \vee e2) \wedge (\neg k \vee e3) \wedge \\ & (\neg l \vee f1) \wedge (\neg l \vee f2) \wedge \neg l \vee f3) \wedge (\neg m \vee g1) \wedge (\neg m \vee g2) \wedge \neg m \vee g3) \wedge \\ & (\neg n \vee a \vee b \vee \neg c \vee d) \wedge (\neg n \vee i \vee j) \end{aligned}$$

## 35.2 Littéral et clause : classification

Soit la formule suivante avec les littéraux de couleur vert des littéraux équivalent à  $\top$  et en bleu les littéraux équivalent à  $\perp$ :

$$(a \vee \neg b) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee c \vee d)$$

Via déduction la clause:

$(a \vee \neg b)$  est falsifié

$(\neg a \vee b \vee \neg c)$  est satisfaite

$(a \vee c \vee d)$  est active

### 35.2.1 Clause active

Une clause active est unitaire si elle a exactement un littéral non affecté:

$$(a \vee c) \wedge (b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$$

est I une interprétation tel que  $I(a) = \top$  et  $I(b) = \perp$ .

Dans ce cas, une clause unitaire admet qu'une seule solution pour être satisfaite:

$$a \wedge b \rightarrow \neg c$$

$c$  doit être affecté à  $\top$ .

### 35.2.2 Littéral pure

Une variable est dite pure dans une formule si ses littéraux sont soit tous positifs ou tous négatifs:

$$(a \vee c) \wedge (\neg a \vee c)$$

## Chapter 36

### Classes polynomiales

### **36.1 2-SAT**

### **36.2 Horn-SAT**

### **36.3 Horn-renommable**



## Chapter 37

### Algorithmes de résolution syntaxiques

## 37.1 Algorithmes complets basé sur la résolution

### 37.1.1 Méthodes basé sur la résolution

Soit une règles de résolution:

$$\frac{w_1=(\neg a \vee \alpha) \wedge (w_2=(a \vee \beta))}{(r=(\alpha \vee \beta))}$$

On n'a  $w_1 \wedge w_2 \models r$  car via affectation de la valeur de  $a$  on retrouve bien  $r$ .

### 37.1.2 Modus Ponuce

### 37.1.3 règles de subsumption

Soit:

$$(w_1 = (a \vee b)) \text{ et } (w_2 = (a \vee b \vee c))$$

On peut dire que  $w_1 \models w_2$ .

### 37.1.4 Règles de fusion

## 37.2 Algorithmes complets Procédure de Davis et Putnam

Prend que des formules CNF.

*Tantque*  $(\perp \neg \in)$  *et*  $(\top \neg \in \sigma)$  *faire* :

appliquer la règles de littéraux unitaires

appliquer la règles de littéraux pures

choisir une variable  $p \in \text{Atomes}(\sigma)$  et construire une nouvelle formule  
 $\sigma = \sigma \text{ sans } p$

Soit:

$$(a \vee c)(b \vee c)(d \vee c)(\neg a \vee \neg b \vee \neg c)$$

Appliquer la règles de littéraux unitaires:

$$(a \vee c)(b \vee c)(d \vee c)(\neg a \vee \neg b \vee \neg c)$$

Appliquer la règles de littéraux pures

$$(a \vee c)(b \vee c)(\textcolor{red}{d} \vee \textcolor{red}{c})(\neg a \vee \neg b \vee \neg c)$$

$$(a \vee c)(b \vee c)(\neg a \vee \neg b \vee \neg c)$$

Éliminer la variable c

$$(a \vee \neg a \vee \neg b)(b \vee \neg a \vee \neg b) = \top$$

Rarement utilisé en pratique.

### 37.3 Algorithmes complets Méthodes des tableaux

Prend n'importe quel type de formules:

## Chapter 38

# Algorithmes de résolutions sémantique

### 38.1 Procédure de DPLL