Memo

Master 2 IA

LAURENT Thomas

Années: 2018 - 2019

Contents

Part I

Fouille de donnée

Chapter 1

Rappel

1.1 Probabilités

Quelques rappels de probabilités : Soient X et Y deux variables aléatoires discrètes prenant leurs valeurs dans DX=x1,..,xn et DY=y1,..,ym respectivement.

$P(x_i) = \frac{|x_i|}{\sum_{j=1}^{n} |x_j|}$

$\sum_{i=1}^{n} P(x_i) = 1$

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

$P(x_i, y_i) = p(x_i) * p(y_i)$ Si X et Y sont indépendantes

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

règle de chainage $P(x_1, x_2, x_3, ...x_n = p(x_1)*p(x_2|x_1)*...*p(x_n|x_{n-1}..x_1)$

distribution conditionnel $\forall x \in X, \forall y \in Y => P(x|y)$

Exemple:

:

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

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

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

1.2 Exemple

A	B	$P(AB)$
a_1	b_1	.1
a_2	b_1	.15
a_1	b_2	.3
a_2	b_2	.45

- $P(a_1) = .40$
- $P(a_1|b_1) = .4$
- $P(a_1|b_2) = .4$
- $P(a_2) = .60$
- $P(a_2|b_1) = .6$
- $P(a_2|b_2) = .6$

1.3 Logarithmes en base 2

$Log_2(\frac{x}{y}) = Log_2(x) - Log_2(y)$
$Log_2(x * y) = Log_2(x) + Log_2(y)$

Chapter 2

Pré traitement des données

2.1 Nettoyage des données

2.1.1 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 : $\alpha x := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2} = \sqrt{\frac{1}{n} (\sum_{i=1}^{n} x_i^2) - \bar{x}^2}$

Médiane : Valeur se trouvant au milieu de données ordonnées

Mode :Valeur la plus fréquente

Amplitude :min, max

2.2 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 : $v_n = \frac{v - moyenne}{ecart_t ype}$

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

Chapter 3

Classification

3.1 Évaluation des classifieurs

3.1.1 Matrice de confusion

Percent of correct classification :

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

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

Coût d'erreur $= \sum_1^n cout(class_{reelle}, classe_{predite})$

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

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

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

Chapter 4

Arbre de décision

4.1 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.

4.1.1 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 $DX = x1, .., xn$. 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:

$H(X)\ >= 0$

$H(X)$ 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.

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

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

4.1.2 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:

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

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

$$Entropie(Longue) = 0$$

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

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

4.1.3 Gain Ratio

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

4.2 critères d'arrêt

4.2.1 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 grain est négatif ou nul)

4.2.2 critères d'arrêt: Paramètre utilisateur

Nombre d'objets minimum par feuille

Taille, profondeur de l'arbre

Temps de construction de l'arbre

Chapter 5

Réseau bayésiens

5.1 Classifieur bayésiens

Nuage	$p(N)$
N=$\perp$	N=$\top$
0.5	0.5

Arroseur	$P(A\|N)$	
N	A=$\perp$	A=$\top$
$\perp$	0.8	0.2
$\top$	0.2	0.8

Pluie	$P(P\|N)$	
N	P=$\perp$	P=$\top$
$\perp$	0.99	0.1
$\top$	0.6	0.4

Pelouse	*Mouille*	$P(M\|A,P)$	
A	P	M=$\perp$	M=$\top$
$\perp$	$\perp$	0.9	0.1
$\perp$	$\top$	0.2	0.8
$\top$	$\perp$	0.2	0.8
$\top$	$\top$	0.05	0.95

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

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

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

$= .5 * .4 * 1 *$

5.2 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

5.2.1 Construction d'un réseau bayésien naïf

soit la variable de classe nommé "Volée":

$Volee$	$\top$	$\perp$
X	5/10	5/10

$Couleur$	$\top$	$\perp$
$rouge$	3/5	2/5
$jaune$	2/5	3/5

$Type$	$\top$	$\perp$
$Sport$	4/5	2/5
$4x4$	1/5	3/5

$Origine$	$\top$	$\perp$
$Locale$	2/5	3/5
$Importee$	3/5	2/5

5.2.2 Règle de classification bayésienne

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

5.2.3 Règle de décision

$P(V|CTO) \ = P(VCTO)$ car indépendantes

$= P(C|v) * P(T|V) * P(O|V) * P(V)$

5.2.4 Observation de classe

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

$P(Volee = Non, Rouge, 4x4, Importee) \ = P(Rouge|Non) * P(4x4|Non) * P(Importee|Non) * P(Non)$

$= 2/5 * 3/5 * 2/5 * 1/2$

$P(Volee = Oui, Rouge, 4x4, Importee) \ = P(Rouge|Oui) * P(4x4|Oui) * P(Importee|Oui) * P(Oui)$

$=$

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

$P(Volee = Non, Jaune, Sport) \ = P(Jaune|Non) * P(Sport|Non) * \sum P(\theta|Non) * P(Non)$

$= 2/5 * 4/5 * 1 * 1/2$

$P(Volee = Oui, Jaune, Sport) \ = P(Jaune|Oui) * P(Sport|Oui) * \sum P(\theta|Oui) * P(Oui)$

$=$

Chapter 6

Clustering

6.1 Approche par le Partitionnement

Soit

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

une fonction de distance d() = 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.

6.2 Approche hiérarchiques

Initialisation Au départ, chaque object 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,...

6.2.1 Exemple avec la fonction d = 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	0			
P2	1	0		
P3	7	5	0	
P4	2	3	6	0

	{P1,P2}	P3	P4
{P1,P2}	0		
P3	5	0	
P4	2	6	0

	{P1,P2,P4}	P3
{P1,P2,P4}	0	
P3	5	0

Chapter 7

ItemSet mining

7.1 Itemsets

Support(D) Le nombre de fois ou D est un sous ensemble de l'itemsset.

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) $\frac{4}{4}$

7.2 Règles d'association

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

	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

7.3 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 II

Apprentissage automatique par la pratique

Chapter 8

Rappel

8.1 Matrices et calcules sur les Matrices

8.1.1 Addition

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

8.1.2 Multiplication

$$\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

$(1 * 5) + (2 * 7) = 19$

8.1.3 Transposer

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

8.1.4 Inverse

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

Soit Determinant D = ad - bc

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

Chapter 9

Algorithms Learn a Mapping From Input to Output

9.1 linear ML algorithms

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

Soit : B0 + B1X1 + B2X2 + B3X3 = 0

Où B0,B1,B2,B3 sont les coefficients présent sur l'axe des ordonnées.

Et X1,X2,X3 sont les valeurs en Input.

9.2 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 $(euro, poids, quantites)$

9.3 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

9.4 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.

9.5 Overview of dias and variance

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

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

Variance Error : Et la quantité estimé par la fonction visé qui changera via un différent ensemble de data utilisé.

Irreductible Error : Ne peut pas être réduit

Chapter 10

Overfitting and Underfitting

10.1 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.

10.2 Underfitting

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

Chapter 11

Model Selection

11.1 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 $text_size$.

```
from sklearn.model_selection import train_test_split

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

11.2 Cross validation

S'applique à un jeu de donné de taille moyenne.
La séparation d'un jeu de donnée d'entrainement 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 moyenner les résultats retournée.
Chaque étape de la cross validation va retournée 2 ensemble (respectivement égaux au indices de $train, test$:

```
from sklearn.model_selection import KFold

kf = KFold(n_splits=10, shuffle=True)
for trainI, testI in kf.split(xset):
    xtrain, xtest = xset[trainI], xset[testI]
    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

11.3 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*.

```
from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()

for train_index, test_index in loo.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

11.4 Matrice de confusion, Précision, Pecall, F1

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

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(ytrain, ypredicted))
>> array([[tn,  fp],
          [fn,  tp]])

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 classifier.

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 permettent de le faire à notre place:

```
from sklearn.metrics import precision_recall_fscore_support

prf = precision_recall_fscore_support(ytest, ypredicted)
print(zip(["Precision", "Recal", "F1", "Support"], [numpy.mean(row) for row in prf]))
{"Precision": _, "Recal": _, "F1": _, "Support": _}
```

sklearn.metrics.precision_recall_fscore_support

Paramètres

y_true *array* les y valides.

y_pred *array* les y qui ont était prédit via un classifier.

Retourné

arrays

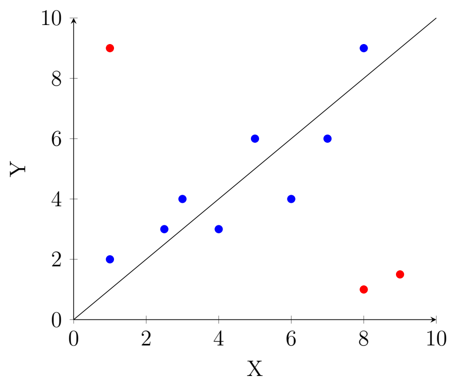
Chapter 12

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.

12.1 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 nouage 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 $= \beta_0 + \beta_1 x_1 + ... \beta_n x_n$

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression().fit(xtrain, ytrain)
reg.score(xtest, ytest)
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 y généré avec le y en paramètre.

12.2 Least squares linear regression

Calculer la régression linéaire avec la méthode Least squares:
Soit:

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

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:

Xm $= \sum x_i \in X = 3$

Ym $= \sum y_i \in Y = 4$

Toutes ligne de régression doivent passer par le point (Xm,Ym).
Calculer tout les écarts des $x_i \in X$ par rapport à Xm (resp 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

$Calculer \beta_1$:

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

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

12.3 Gradient Descent

Soit:

X $= [1, 2, 4, 3, 5]$

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:

$\beta_0 = 0$

$\beta_1 = 0$

α = donnée en énoncé (pour l'exemple égal à 0.01)

Et des fonctions définit tel que:

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 calcules des β_i:

i	$X[i]$	$Y[i]$	$error$	β_0	β_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

12.4 Logistic Regression

12.4.1 Logistic function

Soit:

t $\in \Re[0,1]$ 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.

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

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

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

```
from sklearn.linear_model import LogisticRegression

c = LogisticRegression().fit(xtrain,ytrain)
c.predict(xtest)
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.

12.5 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 $\Bbbk$ groupes d'effectifs $n_{\Bbbk}$.

Noté Y les variables prédire $\{y1, ...y_{\Bbbk}\}$

J variables prédictives $X = (X_1, ...X_j)$

$\mu_{\Bbbk}$ la moyenne (ou *mean* en anglais) valant $lambda(list)- > \frac{\sum list[i]}{taille(list)}$

σ^2 la variance de toutes les classes $\frac{\sum_{i=1}^{n}(x_i - \mu_{\Bbbk})^2}{n - \Bbbk}$

la fonction discriminante pour la classe $\Bbbk$ **avec** x **donné** $D_{\Bbbk}(x) = x * \frac{\mu_{\Bbbk}}{\omega^2} - \frac{\mu_{\Bbbk}^2}{2x\omega^2} + ln(P(k))$

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

12.5.1 bayésien rules

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

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

Où $P(Y = y_{\Bbbk})$ est la probabilité à *priori* d'appartenance à une classe

$P(X|Y = y_{\Bbbk})$ représente la fonction de densité des X conditionnellement à la classe $y_{\Bbbk}$

Chapter 13

Non linear algorithm

13.1 Classification and régression tree

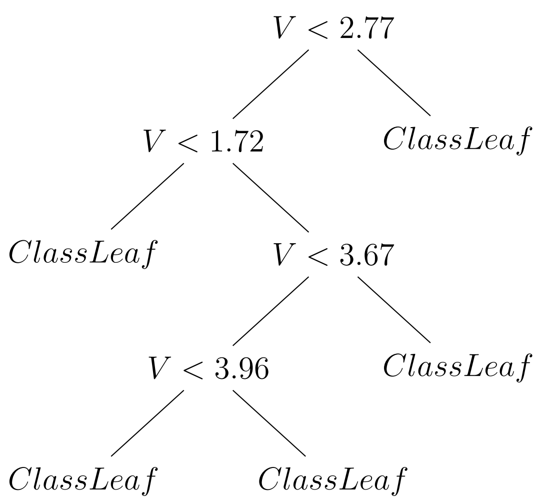
Soit:

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

$$V = 2.67$$

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:

$G =$

$left(X1_1) * (1 - left(X1_1)) +$	$X1_1 = 2.77$
$right(X1_1) * (1 - right(X1_1)) +$	$= 0$ car $V < 2.77 \rightarrow$ Left
$left(X1_2) * (1 - left(X1_2)) +$	$= 0$ car 1.72 < V $\rightarrow$ Right
$right(X1_2) * (1 - right(X1_2)) +$	$X1_2 = 1.72$
$left(X1_3) * (1 - left(X1_3)) +$	$X1_1 = 3.67$
$right(X1_3) * (1 - right(X1_3)) +$	$= 0$ car $V < 3.67 \rightarrow$ Left
$left(X1_4) * (1 - left(X1_4)) +$	$X1_1 = 3.96$
$right(X1_4) * (1 - right(X1_4)) +$	$= 0$ car $V < 3.96 \rightarrow$ Left

```
from sklearn.tree  import DecisionTreeRegressor

c = DecisionTreeRegressor().fit(xtrain,ytrain)
c.predict(xtest)
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 comparent les y généré avec le y en paramètre.

13.2 K moyen

Le K moyen demande une heuristique de type métrique pour comparé les distances entre poins.
Par exemple:

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

```
from sklearn.neighbors import KNeighborsClassifier

c = KNeighborsClassifier(n_neighbors=2).fit(xtrain,ytrain)
c.predict(xtest)
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.

13.3 Support vector machines

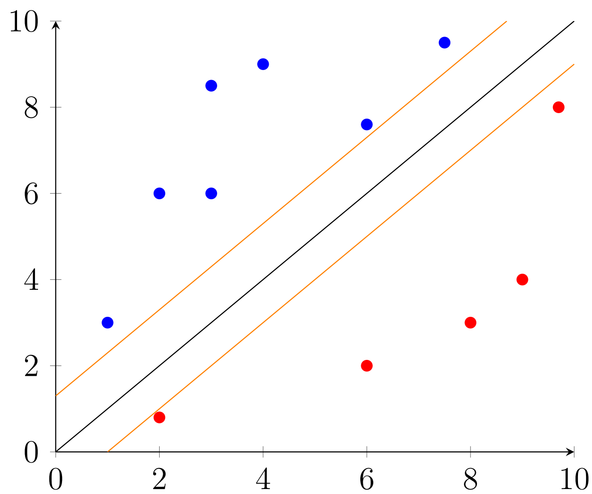
13.3.1 Margin classifier

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 $Margin$.



13.3.2 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:

Tranning 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:

	longeur de la margin	$F1$ *Score*
C_0	0.1	80%
C_1	1	85 % La meilleur borne
C_1	10	85 %

```
from sklearn.svm import SVC

c = SVC().fit(xtrain,ytrain)
c.predict(xtest)
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 III

Outils formel

Chapter 14

Logique classique des propositions

14.1 Vocabulaire

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

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

DAG : Un graphe dirigé acyclique

Taille(Arbre) $= \{toutlessymboles + connecteurs\}$

Var(Arbre) $= \{Touteslesfeuilles\}$

Sous formules(Arbres) $= \{T + \cup_{i=0}^{k} SousFormules(Arbre_i)\}$

Interprétation : ω de $PROP_{ps}$ est une application de PS dans 0.1

Sémantique : $\|\phi\|(\omega)$ d'une formule ϕ de $PROP_{ps}$ dans l'interprétation ω est une élément de 0.1 définit inductive ment par:

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

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

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

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

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

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

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

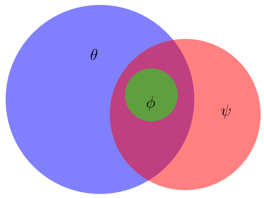
14.2 Propriétés de l'opérateur Models

Réflexivité : $\phi \models \phi$

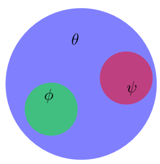
Équivalence à gauche : si $\phi \equiv \theta$ et $\phi \models \psi$ alors $\theta \models \psi$

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

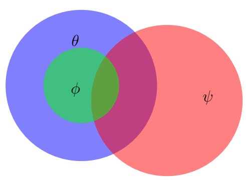
Coupure : si $\phi \wedge \psi \models \theta$ et $\phi \models \psi$ alors $\phi \models \theta$



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



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



14.3 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 $\{\neg, \vee\}, \{vrai, \wedge, \bigoplus\}, \{\neg, \Rightarrow\} ou \{NAND\}$

Suppression des fils équivalent : Soit un arbre D ayant comme sous arbre plus d'une fois le nœud $\alpha = (\top X \top)$, α 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.

14.4 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 φ

Étape inductive :

φ **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'a montrer que φ 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 : $[|x \wedge \neg x|](1^{\rightarrow}) = 0$

14.5 Décomposition de Shannon

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

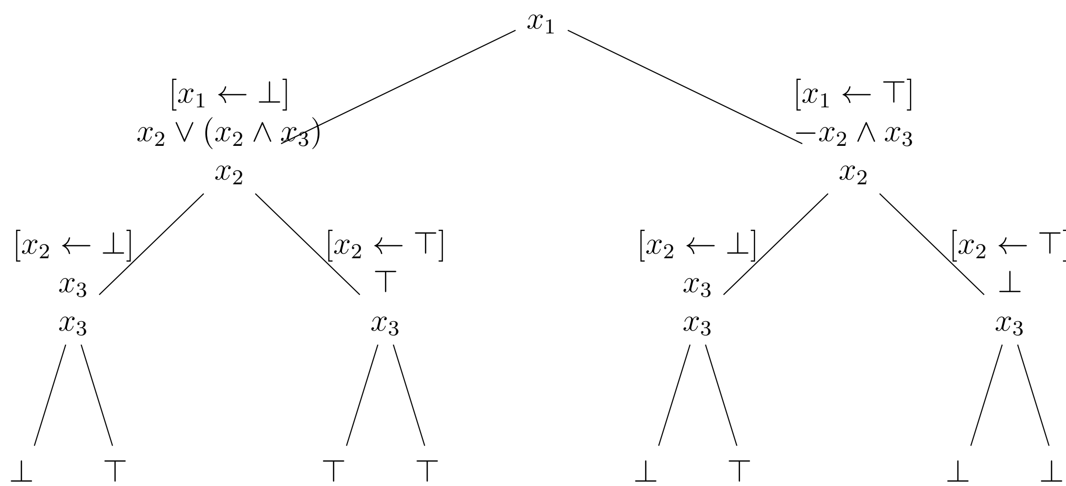
On note $\phi[x \leftarrow 1)$ la formule obtenue en substituant dans ϕ la constante vrai à toutes les occurrences du symbole propositionnel x.

La décomposition de Shannon de ϕ suivant x est la formule:

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

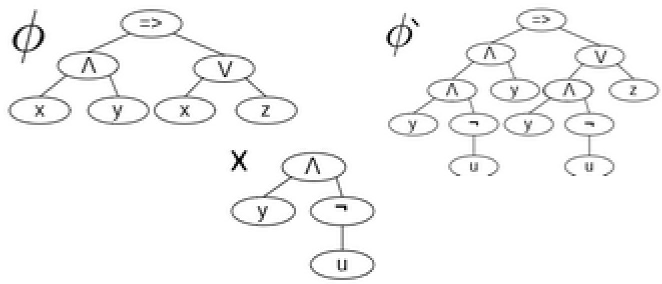
14.6 Arbre de Shannon, ROBDD

Étant donnée un ordre strict total $x_1 < x_2 < x_3$ sur $Var(\phi) = \{x_1, ...X_n\}$
Et une formule $\phi = (\neg x_1 \wedge x_2) \vee (\neg x2 \wedge x_3)$



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

14.6.1 Remplacement ou vérifonctionnalité



$\phi \equiv \phi^{`}$ quelque soit la valeur de x (vrai ou faux).

14.6.2 Substitution

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

14.7 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.

14.7.1 Table de Karnaugh

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

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

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

14.7.2 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.

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

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

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

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

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

Via une table de Karnaugh:

ϕ	$\neg a \neg b$	$\neg ab$	ab	$a \neg b$
$\neg c$				
c	X		X	X

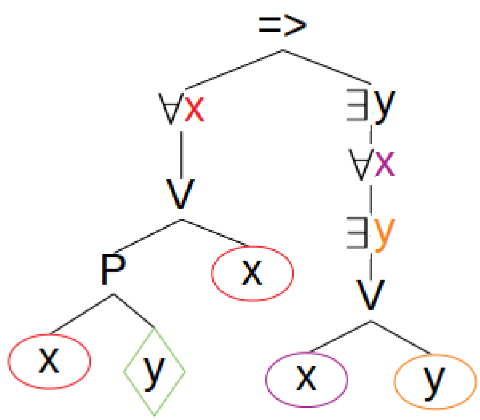
Égal à $(a \wedge c) \vee (\neg b \wedge c)$.

Chapter 15

Logique classique et prédicat du premier ordre

15.1 Syntaxe via les arbres

$\phi =$



15.1.1 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 ϕ.

15.1.2 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 diffèrent 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 ϕ.
Les occurrences liée 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.

15.1.3 Occurrences quantifié

Les occurrences quantifié sont toutes les variable positionné 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 ϕ.

15.1.4 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

15.2 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.

$\phi =$

$$\begin{tikzcd}
 & & = \arrow[dl] \arrow[dr] & \\
 & + \arrow[dl] \arrow[dr] & & y \\
x & & 2 &
\end{tikzcd}$$

$= \ \in \Re$ d'arriter 2

$+ \ \in \Im$ d'arriter 2

$2 \ \in \Im$ d'arriter 0

$X, Y \ \in X$

Avec une interprétation tel que:

$D_i \ = \mathbb{N}$

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

$2_i \ = 3$

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:
$\|x+2\|(I)(X_i) = +_i(\|x\|(I)(X_i), \|2\|(I)(X_i)) = +_i(5,3) = 8$
$\|\phi\|(I)(X_i) = =_i (8,10) = 0(faux)$

$\psi =$

$$
\begin{array}{ccccc}
 & & \exists x & & \\
 & & \downarrow & & \\
 & & = & & \\
 & \swarrow & & \searrow & \\
 & + & & & y \\
\swarrow & & \searrow & & \\
x & & 2 & &
\end{array}
$$

$\|\psi\|(I)(X_i)[x \leftarrow 7]) =$
$=_i (+_i(\|x\|(I)(X_i[x \leftarrow 7]), 3), \|y\|(I)(X_i[x \leftarrow 7])) =$
$=_i (+_i(7,3), 10) =$
$=_i (10,10) = 1(vrai)$

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

Soit ϕ la formule ϕ ci dessus, la formule interprété avec deux assignations différente:

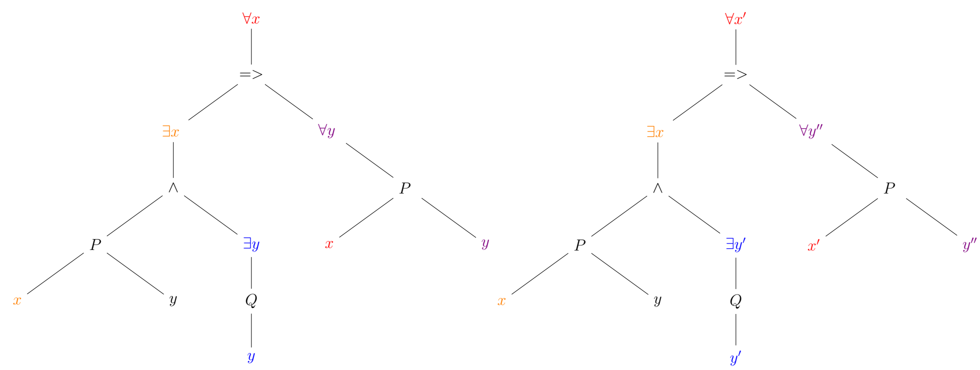
$X_i^1 \;\; x \to 5, y \to 10$

$X_i^2 \;\; x \to 6, y \to 10$

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

15.3 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:



15.4 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 ψ et Q = $\forall$ ou $\exists$ alors :

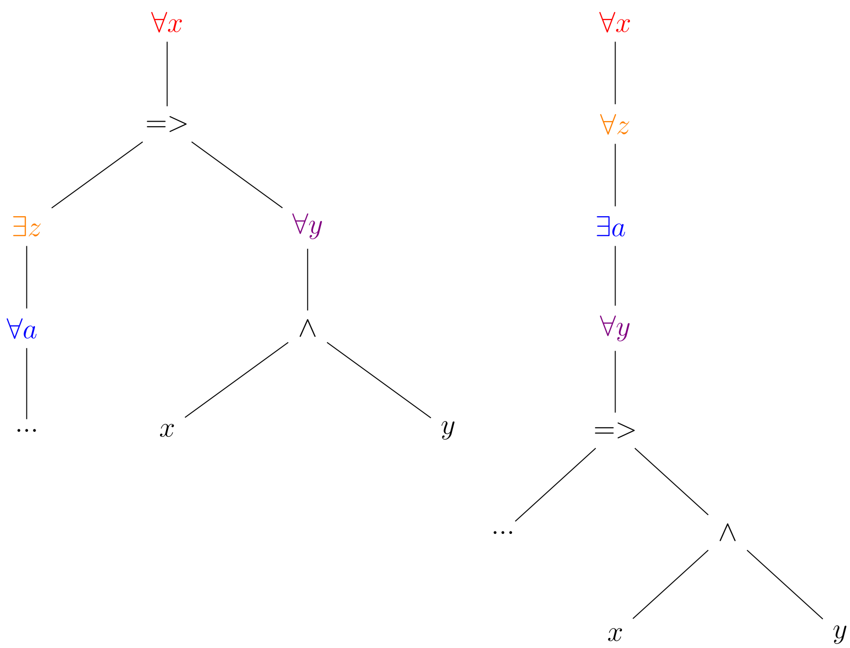
$$Q x \phi \equiv \phi$$
$$Q x(\phi \wedge \psi) \equiv (Q x \phi) \wedge \psi)$$
$$Q x(\phi \vee \psi) \equiv (Q x \phi) \vee \psi)$$

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

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

15.5 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 fessant attention que lorsqu'on remonte un quantificateur par de la une négation, on applique le duel sur le quantificateur, Et aussi il faut garder l'ordre des quantificateur par rapport à la profondeur de leur sous arbre:
(Rappel que $A => B \equiv \neg A \vee B$):



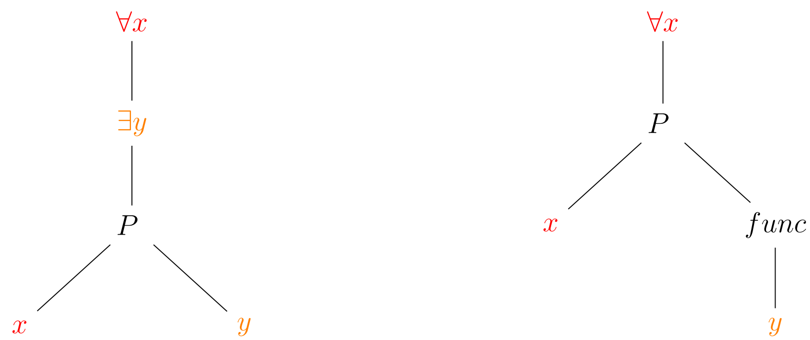
La partie contenant tout les quantificateurs s'appelle le Prefix et la partie sans quantificateurs s'appelle la Matrice.

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

15.6 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:



15.7 Forme propositionnelle

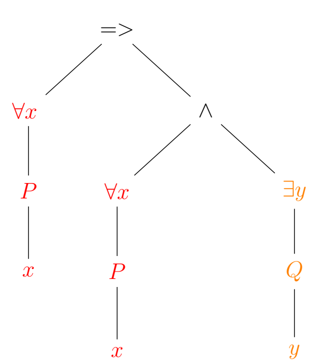
L'ensemble $SFP(\phi)$ des sous-formules premières de $\phi \in FORM_L$ est défini inductivement par:

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

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

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

Si la formule propositionnelle ϕ est propositionnellement valide alors ϕ est valide



$SFP(\phi) = \{$ formules de couleur *rouge*, formules de couleur *orange*$\}$, ϕ est propositionnellement équivalent à $A => (A \vee B)$ qui est propositionnellement valide donc ϕ est valide

Chapter 16

Calculabilité et Machine de Turing

Soit une machine de turing M un quadruplet M = K,$\sum$,δ,s)

K ensemble fini d'état

s $\in$ K état initial

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

- $\triangleright$ marque de début
- $\sqcup$ séparateur ou fin de ruban

δ :$(K$ x $\sum)$ x $((K \cup \{yes, no, \uparrow\})$ x $\sum$ x $\{\leftarrow, \rightarrow, -\})$

- $\{yes, no\}$ état acceptable
- $\{\leftarrow, \rightarrow, -\}$ mouvement de la tête de lecture

16.1 Machines de Turing

Une machine de Turing:

non déterministe est une machine qui pour un état n donné peut dériver sur deux état $n+1$ diffèrent (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 seul 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).

16.1.1 Machine de Turing universel

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

16.2 RE, coRE et R

Un langage Récursif (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 $\{L \in \rho artie(\sum \neg\{-\})*) \text{ tel que } L^{-} \in RE\}$

Remarque: $R \subseteq RE \cap coRE$

16.2.1 Preuve de R est inclue 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 $Ma \in RE$:

$$\begin{tikzcd} & & \uparrow \\ Ma \arrow[r] & Mr \arrow[ur, "\text{no}"] \arrow[dr, "\text{yes}"'] & \\ & & yes \end{tikzcd}$$

16.2.2 Preuve de R est inclue 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 $Ma \in coRE$:

$$\begin{tikzcd} & & \uparrow \\ Ma \arrow[r] & Mr \arrow[ur, "\text{no}"] \arrow[dr, "\text{yes}"'] & \\ & & no \end{tikzcd}$$

16.3 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

16.4 réduction fonctionnel

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

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

16.4.1 Exemple de réduction fonctionnel

Soit $L = \{(i, j)$ tel que i et h sont des indices de machine déterministes telles que pour tout mot d'entrée x, on n'a $M_i(x) =\uparrow$ 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 ssi $M_i(x) \neq\uparrow$
(j,k) $\in$ L ssi $M_j(y) =\uparrow$ et $M_k(y) \neq\uparrow$, $\forall y$.

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

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 ssi $M_i(x) \neq\uparrow$
(j,k) $\in$ L ssi $M_j(y) \neq\uparrow$ et $M_k(y) =\uparrow$, $\forall y$.

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

Part IV

Recherche Opérationnel

Chapter 17

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

17.1 Modèle linéaire continus à 2 variables

Soit le modèle linéaire suivantes:

Déterminer $(x, y) \in \Im^2$

Minimisant $z = 1000x + 1200y$

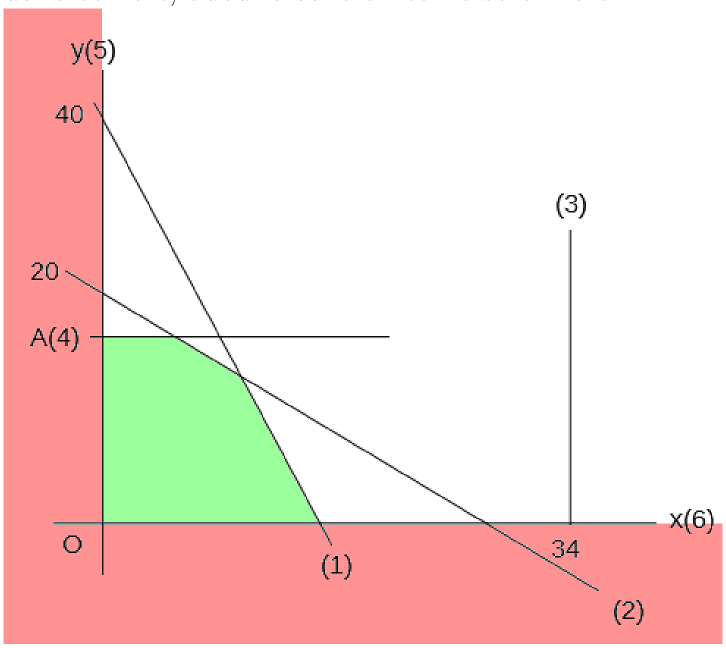
sous les contraintes :

$$
\begin{aligned}
&(1) 8x + 4y \leq 160 \\
&(2) 4x + 6y \leq 120 \\
&(3) x \leq 34 \\
&(4) y \leq 14 \\
&(5) 0 \leq x \\
&(6) 0 \leq y
\end{aligned}
$$

17.1.1 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.



17.1.2 recherche de la solution optimal

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

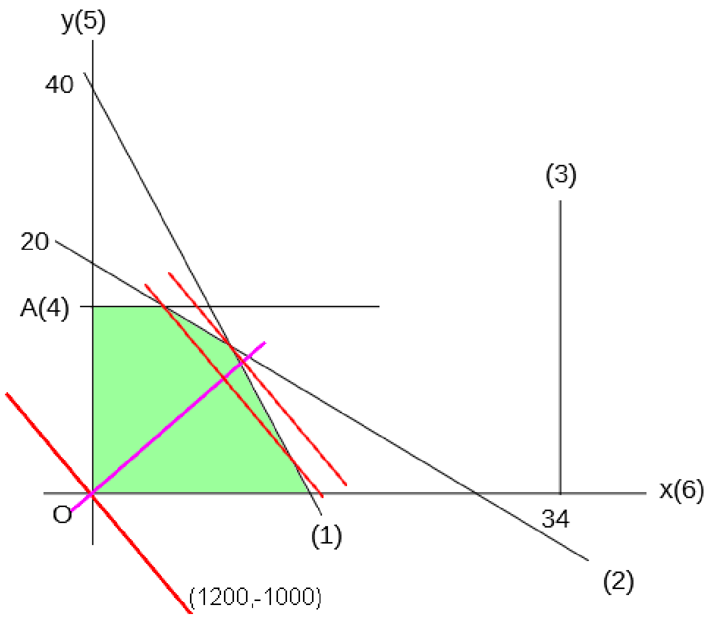
$$z = 1000x + 1200y = 0 = 1000 * (1200) + 1200 * (-1000)$$

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 18

Le simplexe

Soit le modèle linéaire suivantes:

Déterminer $(x, y) \in \Im^2$

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) $0 \leq y$

18.1 Initialisation du simplexe

Pour chaque expression du type (1)(2)(3) intégrer un e_i pour la transformer en équation.
On appel les e_1 des variables d'accumulation, Ce qui fait

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$

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) $e_1, e_2, e_3 \geq 0$

18.2 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.

18.3 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 $A\ Deduire(x, y, e_1, e_2, e_3)$

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 = $(0, 3, 0, 5, 25)$

18.4 Exemple simple Premier itération

18.4.1 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.

18.4.2 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)

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

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

$2x - y + e_3 = 28$ 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.

18.4.3 pivotage

On choisis l'équation associé 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:

$Z = 3x + 7y$ devient

$$\begin{aligned} Z &= 3x + 7(x - e_1 + 3) \\ Z &= 10x - 7e_1 + 27 \end{aligned}$$

$x - e_1 = 3$ est déjà normalisé

$y + e_2 = 8$ devient

$$\begin{aligned} 8 &= x - e_1 + 3 + e_2 \\ 5 &= x - e_1 + e_2 \end{aligned}$$

$2x - y + e_3 = 28$ devient

$$\begin{aligned} 28 &= 2x + (x - e_1 + 3) + e_3 \\ 25 &= 3x - e_1 + e_3 \end{aligned}$$

18.4.4 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$

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

Variables hors base x, e_1

Variables de Base y, e_2, e_3

Solution admissible $(0, 3, 0, 5, 25)$ et Z = 21

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

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

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

(5) $0 \leq x$

(6) $0 \leq y$

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

A ne pas oublier de vérifier la canonicité du modèle.

18.5 Exemple simple Seconde itération

18.5.1 Choix de la variable entrante

X sera choisit, donc X sera notre variable entrante.

18.5.2 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.

18.5.3 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

$$\begin{aligned} Z &= 10(e_1 - e_2 + 5) - 7e_1 + 27 \\ Z &= 3e_1 - 10e_2 + 71 \end{aligned}$$

$-x + y + e_1 = 3$ devient

$$\begin{aligned} 3 &= -(e_1 - e_2 + 5) + y + e_1 \\ 8 &= y + e_2 \end{aligned}$$

$3x - e_1 + e_3 = 25$ devient

$$\begin{aligned} 25 &= 3(e_1 - e_2 + 5) - e_1 + e_3 \\ 10 &= 2e_1 - 3e_2 + e_3 \end{aligned}$$

18.5.4 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$

Maximisant $Z = 3e_1 - 10x + 71$

Variables hors base e_2, e_1

Variables de Base y, x, e_3

Solution admissible $(5, 8, 0, 0, 10)$ et Z = 71

(1) $y + e_2 = 8$

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

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

(5) $0 \leq x$

(6) $0 \leq y$

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

A ne pas oublier de vérifier la canonicité du modèle.

18.6 Exemple simple, troisième itération

18.6.1 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, $\frac{10}{2} = 5$

18.6.2 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$

Maximisant $Z = 86 - \frac{11}{2}e_2 - \frac{3e_3}{2}$

Variables hors base e_2, e_3

Variables de Base y, x, e_1

Solution admissible $(10, 8, 5, 0, 0)$ et Z = 86

(1) $-\frac{1}{2}e_2 + \frac{e_3}{2} + e1 = 10$

(2) $e_2 + y = 8$

(3) $e_1 - \frac{3}{2}e_2 + \frac{e_3}{2} = 5$

(5) $0 \leq x$

(6) $0 \leq y$

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

18.7 Exemple simple, dernière itération

Stop car e_2 et e_3 sont inférieur à 0 dans Z.

Chapter 19

Simplexe à deux phases

Soit le modèle suivant:

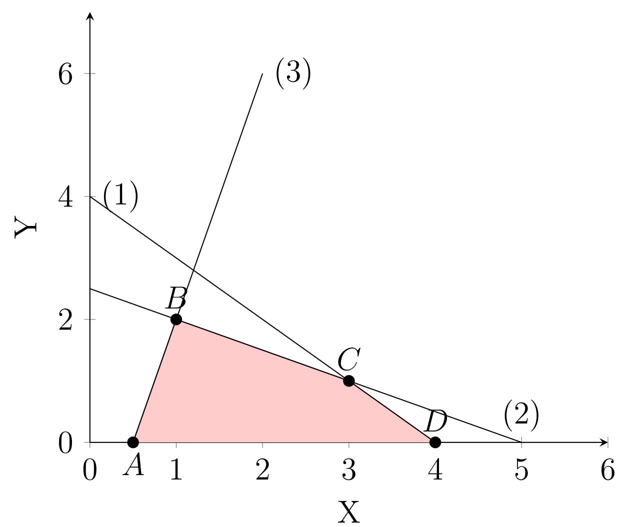
Déterminer $(x, y) \in \Im^2$

Maximisant $Z = 2x + 3y$

sous les contraintes :

(1) $x + y \leqslant 4$
(2) $x + 2y \leqslant 5$
(3) $4x - y \geqslant 2$
(4-5) $x, y \geqslant 0$

Lorsque le sens de l'équation est $\leqslant$ il faut ajouter une variable e_i, dans le cas des équations $\geqslant$ il faut ajouter une variable d'excédant a dans la contrainte concerné et instaurer Z à $-a$
La représentation graphique ci dessous:



19.1 Première phase du simplexe à deux phases

Pour toutes expression sous la forme $A \geqslant -i$, 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:

19.1.1 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3, a) \in \Im^5$

Maximisant $Z = -a = 4x - y - e_3 - 2$ $\%oldZ = 2x + 3y$

Variables hors base x, y, a

Variables de Base e_1, e_2, e_3

Solution admissible $(0, 0, 4, 5, 0, 2)$ et Z $= -2$

(1) $x + y + e_1 = 4$

(2) $x + 2y + e_2 = 5$

(3) $4x - y - e_3 + a = 2$

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

Ce modèle est canonique.

19.2 Premier phase du simplexe à deux phases, première itération

19.2.1 Variable entrante et sortante

La variable entrante sera x

La variable sortante sera a car:

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

19.2.2 pivotage

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

19.2.3 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3, a) \in \Im^5$

Maximisant $Z = -a$
$\%oldZ = 2x + 3y$

Variables hors base y, a

Variables de Base e_1, e_2, x

Solution admissible $(\frac{1}{2}, 0, \frac{7}{2}, \frac{9}{2}, 0, 0)$ et Z = 0

(1) $\frac{5}{4}y + e_1 + \frac{e_3}{4} - \frac{a}{4} = \frac{7}{2}$

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

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

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

19.3 Premier phase du simplexe à deux phases, seconde itération

Nous somme en présente 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:

$$y_A = 0$$
$$4x_A - y_A = 2$$

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

19.4 Seconde phase du simplexe à deux phases

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$

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

Variables hors base y

Variables de Base e_1, e_2, x

Solution admissible $(\frac{1}{2}, 0, \frac{7}{2}, \frac{9}{2}, 0)$ et Z = 0

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

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

(3) $x - \frac{y}{4} - \frac{e_3}{4} = \frac{1}{2}$

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

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

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$ et Z = 1

Maximisant $Z = \frac{7}{2}y + \frac{e_3}{2} + 1$

Variables hors base y, e_3

Variables de Base e_1, e_2, x

Solution admissible $(\frac{1}{2}, 0, \frac{7}{2}, \frac{9}{2}, 0)$

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

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

(3) $x - \frac{y}{4} - \frac{e_3}{4} = \frac{1}{2}$

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

Ce modèle est canonique.

19.5 Seconde phase du simplexe à deux phases, première itération

19.5.1 Variable entrante et sortante

La variable entrante sera y

La variable sortante sera e_2 car:

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

19.5.2 pivotage

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

19.5.3 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$ et Z = 8

Maximisant $Z = -\frac{14}{9}e_2 + \frac{e_3}{9} + 8$

Variables hors base e_2, e_3

Variables de Base e_1, y, x

Solution admissible $(1, 2, 1, 0, 0)$

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

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

(3) $x + \frac{e_2}{9} - \frac{2}{9}e_3 = 1$

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

Ce modèle est canonique.

19.6 Seconde phase du simplexe à deux phases, seconde itération

19.6.1 Variable entrante et sortante

La variable entrante sera e_3

La variable sortante sera e_1 car:

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

19.6.2 pivotage

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

19.6.3 Nouveau modèle

Voici le nouveau modèle:

Déterminer $(x, y, e_1, e_2, e_3) \in \Im^5$

Maximisant $Z = -e_1 - e_2 + 9$

Variables hors base e_2, e_1

Variables de Base e_3, y, x

Solution admissible $(3, 1, 0, 0, 9)$ et Z = 9

(1) $9y + 5e_2 + e_3 = 9$

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

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

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

Ce modèle est canonique.

19.7 Seconde phase du simplexe à deux phases, troisième itération

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

Part V

Représentation des connaissances et raisonnement

Chapter 20

Logique propositionnel

20.1 Vocabulaire

Les *Logiquespropositionnel* sont définit via les symboles suivant:
$\top, \bot, C, \neg C, C \wedge C, C \vee C, C \Rightarrow C$

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 normal conjonctive (une conjonction de clauses)

DNF est une forme normal disjonctive (une disjonction de cubes)

20.2 cohérence d'un ensemble de clauses

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

$x \vee x \vee y_1 \vee ...y_n \equiv x \vee y_1 \vee ...y_n$

$x \vee \neg x \vee y_1 \vee ...y_n \equiv \text{`}top$

$x \vee \top \equiv \top$

$x \vee \bot \equiv x$

Si K est vide alors K est cohérente

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

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

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

Chapter 21

Introduction à la logique de description

21.1 Attributive Language with Complement

Les ALC sont définit via les symboles suivant:
$\top, \bot, C, \neg C, C \sqcap C, C \sqcup C, \forall r.C, \exists r.C$

21.1.1 Propriétés

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

$(\neg\neg C)^I = C^I$

$(\neg(C \sqcap D))^I = (\neg C \sqcup \neg D)^I$

$(\neg(C \sqcup D))^I = (\neg C \sqcap \neg D)^I$

$(\neg\forall r.C)^I = (\exists r.\neg C)^I$

$(\neg\exists r.C)^I = (\forall r.\neg C)^I$

$\exists r.\bot \equiv \bot$

$\forall r.\top \equiv \top$

21.2 Logique de description

Définit via les symboles suivant:
$\ell_{ALC}, C \sqsubseteq C, \sqsupseteq C$

21.2.1 Sémantique

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

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

21.2.2 Assertions

$a : C$ a est une instance de C

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

21.3 TBoxes et ABoxes

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

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

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

21.3.1 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:

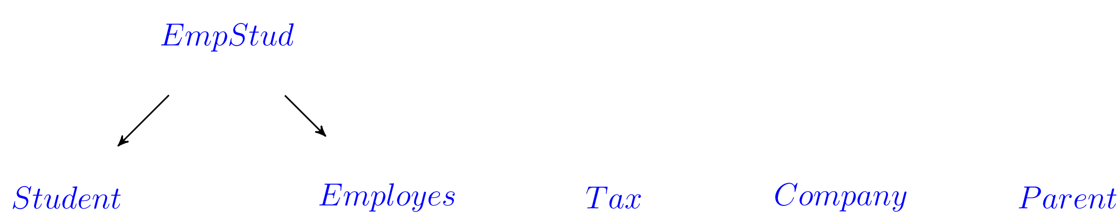
Does $EmpStud$ subsume $Student \sqcap Employe$? : yes

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

Does $\exists pays.\bot$ subsume $EmpStud$? : No

21.3.2 Classification

Les schémas de classification aide pour trouver les subsumptions:



21.3.3 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 $workFor$

21.3.4 Retrieval

$Student$?$\{john\}$

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

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

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

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

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

21.3.5 Equivalance of concept

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

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

21.3.6 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

21.3.7 ABox consistency

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

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

21.3.8 Réduction et consistance

Soit $KB = \langle T, A \rangle, C, D \in \iota_{ALC}, a \in I$ and $a^{\text{`}}$ new in KB

Concept subsumption wrt T : $KB \vDash C \sqsubseteq D$ ssi $\langle T, A \cup \{a^{\text{`}} : C \sqcap \neg D\}\rangle$ est inconsistant

Instance chacking : $KB \vDash a : C$ ssi $\langle T, A \cup \{a : \neg C\}\rangle$ est inconsistant

Concept satisfiability wrt T : C est satisfiable wrt T ssi $\langle T, A \cup \{a^{\text{`}} : C\}\rangle$ est consistent

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

$KB \vDash john : Student \sqcap \exists empBy.\top$?
$KB \cup \{john : \neg(Student \sqcap \exists empBy.\top)\} \vDash \bot$?

Is $EmpStud \sqcap \neg\exists pays.Tax$ **satisfiable wrt** KB ?
$KB \cup \{a : EmpStud \sqcap \neg\exists pays.Tax \nvDash \bot$?, for a new

Chapter 22

Méthode des Tableau pour les ALC

22.1 Pre processing

22.1.1 Réécriture

Réécrite chaque:

$C \sqsubseteq D$ dans T en $\top \sqsubseteq \neg C \sqcup D$

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

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

$\neg\neg C \ \rightarrow C$

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

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

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

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

22.1.2 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

22.1.3 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** $\{a : C, a : D \nsubseteq A$ alors
$A := A \cup \{a : C, a : D\}$

$\sqcup-$ rule

Si $a : C \sqcup D \in A$ **et** $\{a : C, a : D\} \cap A = \varnothing$ alors
$A := A \cup \{a : E\}$, 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
$A := A \cup \{(a, c) : R, c : C\}$, for c new in A

$\forall-$ rule

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

22.2 Exemple

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

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

- $\{a : A \sqcap B, a : \forall r.\forall r.C\}$
  - $\{a : A, a : B\}$
    - $\{a : \neg A \sqcup \exists r.B\}$
      - $\{a : \neg A\}$
      - $\{a : \exists r.B\}$
        - $\{(a, x) : r, xB\}$
          - $\{x : \forall r.C\}$
            - $\{x : \neg A \sqcup \exists r.B\}$
              - $\{x : \neg A\}$
              - $\{x : \exists r.B\}$
                - $\{(x, y) : r, yB\}$
                  - $\{y : C\}$
                    - $\{y : \neg A \sqcup \exists r.B\}$
                      - $\{y : \neg A\}$
                      - $\{y : \exists r.B\}$
                        - y est bloqué par z

22.3 Exemple 2

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

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

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

$\{a : A, a : B\}$

$\{a : \neg A \sqcup \exists r.B\}$

$\{a : \neg A\}$ $\quad$ $\{a : \exists r.B\}$

$\{(a, x) : r, xB\}$

$\{x : A \sqcap \forall r.A\}$

$\{x : A, x : \forall r.A\}$

$\{x : \neg A \sqcup \exists r.B\}$

$\{x : \neg A\}$ $\quad$ $\{x : \exists r.B\}$

$\{(x, y) : r, yB\}$

$\{y : A \sqcap \forall r.A\}$

$\{y : A, y : \forall r.A\}$

$\{y : \neg A \sqcup \exists r.B\}$

$\{y : \neg A\}$ $\quad$ $\{y : \exists r.B\}$

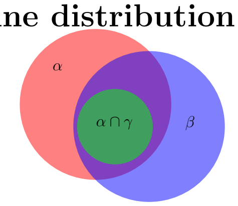
$...InfinitLoop...$

Chapter 23

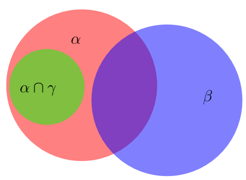
Logique presque tout

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

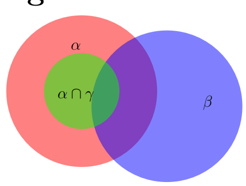
Bonne distribution :



Mauvaise distribution :



Cas général :



23.1 Système P

Réflexivité :

Almost all : $\alpha \mid\!\sim \alpha$

ensembliste : $A \Subset A$

Équilibrage à gauche :

Almost all : Si $\models \alpha \Leftrightarrow \beta$ et $\alpha \mid\!\sim \gamma$ alors $\beta \mid\!\sim \gamma$

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

Équilibrage à droite :

Almost all : Si $\alpha \models \beta$ et $\gamma \mid\!\sim \alpha$ alors $\gamma \mid\!\sim \beta$

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

Coupure :

Almost all : Si $(\alpha \wedge \beta) \mid\!\sim \gamma$ et $\alpha \mid\!\sim \beta$ alors $\alpha \mid\!\sim \gamma$

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

Monotonie :

Almost all : Si $\alpha \mid\!\sim \beta$ et $\alpha \mid\!\sim \gamma$ alors $\alpha \wedge \beta \mid\!\sim \gamma$

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

Ou :

Almost all : Si $\alpha \mid\!\sim \gamma$ et $\beta \mid\!\sim \gamma$ alors $\alpha \vee \beta \mid\!\sim \gamma$

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

23.1.1 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 : $C \mid\!\sim \neg F$

Presque tout les québécois parlent le français : $Q \mid\!\sim F$

Les québécois aiment le sirop d'érable : $Q \Rightarrow S \equiv Q \mid\!\sim S$

Les québécois sont canadiens $Q \Rightarrow C \equiv Q \mid\!\sim C$

Presque tout les québécois canadiens parlent le français

Nous avons $Q \mid\!\sim C$ et $Q \mid\!\sim F$

Avec la monotonie on obtient $Q \wedge C \mid\!\sim F$

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

Avec $Q \wedge C \mid\!\sim F$

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

Alors via l'équilibrage à droite $Q \wedge C \mid\!\sim F \vee A$

23.2 Tolérance du Système P

Soit la basse de connaissance:

$$\begin{array}{ll} \Delta & C \Rightarrow \neg F \\ & Q \Rightarrow F \\ W & Q \Rightarrow S \\ & Q \Rightarrow C \end{array}$$

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

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

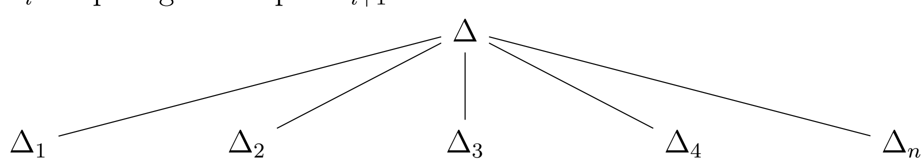
$$\begin{array}{ll} \Delta & C^1 \Rightarrow \neg F^0 \\ & Q^0 \Rightarrow F^0 \\ W & Q^0 \Rightarrow S^s \\ & Q^0 \Rightarrow C^1 \end{array}$$

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

$$\begin{array}{ll} \Delta & C^1 \Rightarrow \neg F^1 \equiv \neg\top \vee \bot \\ & Q^1 \Rightarrow F^1 \\ W & Q^1 \Rightarrow S^s \\ & Q^1 \Rightarrow C^1 \end{array}$$

23.3 Stratification du système P

Δ stratifiable (ou cohérente) c'est le fait de pouvoir diviser Δ en Δ_i.
Δ_i est plus général que Δ_{i+1}



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

23.4 Exemple de stratification possible

23.4.1 Initialisation

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

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

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

23.4.2 Première itération

On n'a:

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

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

$$\Delta_1 = \{\}$$

Pour $C^1 \rightarrow \neg F^0$ est toléré par l'algorithme donc transféré dans Δ_1 à la fin du tour:

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

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

$$\Delta_1 = \{\}$$

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

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

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

$$\Delta_1 = \{\}$$

23.4.3 Seconde itération

On n'a:

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

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

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

$$\Delta_2 = \{\}$$

Pour $Q \rightarrow F$ et $(Q \wedge C) \rightarrow F$ sont toléré donc seront transféré dans Δ_2 à la fin du tour:

$$\Delta = \{\}$$

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

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

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

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

23.5 Exemple de stratification non possible

23.5.1 Initialisation

$$\begin{aligned}
\Delta &= \{C \rightarrow \neg F, Q \rightarrow F\}\\
W &= \{Q => S, Q => C\}\\
\alpha \rightarrow \beta &= (Q \wedge C) \rightarrow (F \vee A)
\end{aligned}$$

23.5.2 Première itération

On n'a:

$$\begin{aligned}
\Delta &= \{C \rightarrow \neg F, Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)\}\\
W &= \{Q => S, Q => C\}\\
\Delta_1 &= \{\}
\end{aligned}$$

Pour $C^1 \rightarrow \neg F^0$ est toléré par l'algorithme donc transféré dans Δ_1 à la fin du tour:

$$\begin{aligned}
\Delta &= \{C^1 \rightarrow \neg F^0, Q^0 \rightarrow F^0, (Q^0 \wedge C^1) \rightarrow (\neg F^0 \wedge \neg A^a)\}\\
W &= \{Q^0 => S^s, Q^0 => C^1\}\\
\Delta_1 &= \{\}
\end{aligned}$$

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

$$\begin{aligned}
\Delta &= \{C \rightarrow \neg F, Q \rightarrow F, (Q \wedge C) \rightarrow (\neg F \wedge \neg A)\}\\
W &= \{Q => S, Q => C\}\\
\Delta_1 &= \{\}
\end{aligned}$$

23.5.3 Seconde itération

On n'a:

$$\Delta = \{Q \to F, (Q \land C) \to (\neg F \land \neg A)\}$$

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

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

$$\Delta_2 = \{\}$$

Pour $Q \to F$ et $(Q \land C) \to (\neg F \land \neg A)$ ne sont pas toléré:

$$\Delta = \{Q \to F, (Q \land C) \to (\neg F \land \neg A\}$$

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

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

$$\Delta_2 = \{\}$$

Δ_2 est vide donc $\{\alpha \to \neg\beta\} \cup \Delta$ n'est pas stratifiable.

Chapter 24

Logique de description DL Lite

24.1 Opérateurs

Pour une *ABox*:

$\neg$ négation

$\exists$ Rôle $\rightarrow$ Concept

$$\begin{matrix} (A & ,B) \\ (C & ,D) \end{matrix} \ \exists \rightarrow \ \begin{matrix} (A) \\ (C) \end{matrix}$$

$^{\neg}$ Rôle $\rightarrow$ Rôle

$$\begin{matrix} (A & ,B) \\ (C & ,D) \end{matrix} \ ^{\neg} \quad \rightarrow \quad \begin{matrix} (B & ,A) \\ (D & ,C) \end{matrix}$$

24.2 Requêtes

24.2.1 Grounded query

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

Avec A_1 des concepts et P_i des rôles.

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

24.2.2 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*

24.3 Fermetures négatives

Sur DL-Lite_{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*:

DL-Lite_{core} **TBox**	$cln(T)$
$Teacher \sqsubseteq \neg Student$	$Teacher \sqsubseteq \exists HasSupervisor$
$Teacher \sqsubseteq \exists TeachesTo$	$\exists HasSupervisor^{\urcorner} \sqsubseteq \neg Student$
$\exists TeachesTo^{\urcorner} \sqsubseteq Student$	$\exists TeachesTo^{\urcorner} \sqsubseteq \neg Teacher$
$Student \sqsubseteq \exists HasSupervisor$	
$\exists HasSupervisor^{\urcorner} \sqsubseteq Teacher$	

24.4 Gestion des contraintes et MultiABox

24.4.1 Expansion

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

$TBox$	$ABox$
$\exists P \sqsubseteq B$	A(a)
$A \sqsubseteq B$	P(c,b)
$A \sqsubseteq \neg C$	$B(a)$
	$B(c)$

La MultiABox M est composé que d'une ABox.
$B(a)$ est ajouté grâce au second axiome
$B(c)$ est ajouté grâce au premier axiome

24.4.2 Spliting

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

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

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

24.4.3 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)	

$o_{incl} = \{ABox_1, Abox_2\}$

24.4.4 Modifieurs

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

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

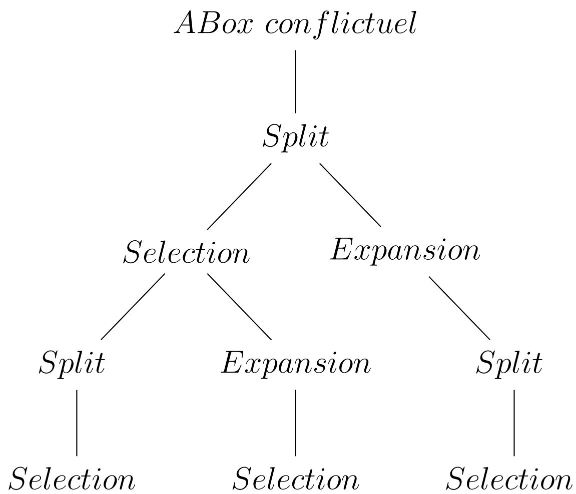
$o_{card}(o_{card}(M)) = o_{card}(M)$

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

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

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

24.4.5 Complex modifieurs



24.4.6 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 moiter des ABox répondent avec le résultat R, alors R sera prise

Base inférence : Si plus de α 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 $Safe$

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

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

Conclusion NON

Chapter 25

Complexité

25.1 Analyse de complexité pour D(M1,Safe)

Une approche naïf non satisfiable serait de:

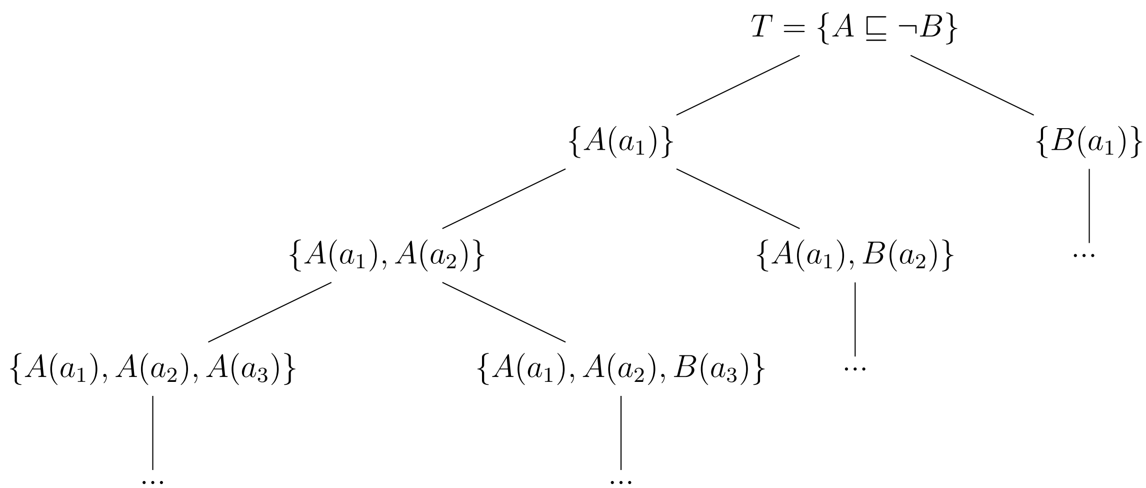
(1) Calculer les $R_1 ... R_n$ après avoir appliqué le $modifiersplitting$

(2) Calculer l'intersection des R_i

Un cas extrême pour résoudre le problème

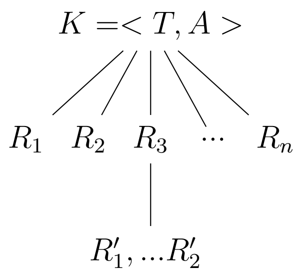
$T = \{A \sqsubseteq \neg B\}$

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



Complexité $O(2^n)$

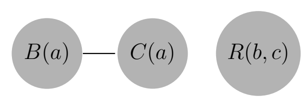
25.2 Analyse de la complexité pour D(M2,Forall)



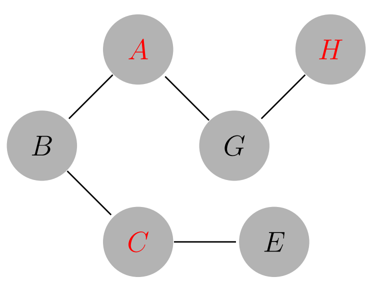
Splitting et Selection selon le cardinal le plus haut:
$R'_1, ...R'_r$

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:

$$K =< T, A >$$
$$T = \{B \sqsubseteq \neg C\}$$
$$A = \{B(a), C(a), R(b, c)\}$$



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 VI

Théories de la Décision

Part VII

Apprentissage

Chapter 26

Approche d'apprentissage par la logique

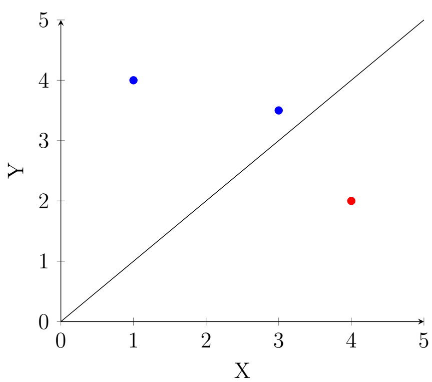
Une approche simple concernant l'apprentissage de problèmes dont le domaine de sortie est boolean serait de passer par la logique classique pour pouvoir simplifier la compréhension du problème.

26.1 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?

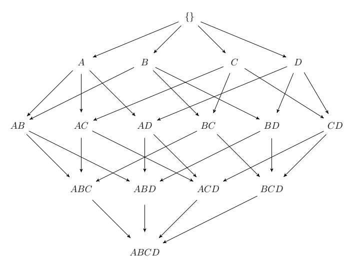
Pour les cas suivants:

- Faciliter la compréhension du problème
- Comprendre pourquoi une décision donné pour une entrée

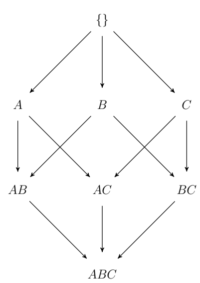
26.1.1 convergence des données

Dans le tableau d'acceptation on peut transformé la règle 3 en son dual via la Lois de De Morgan:

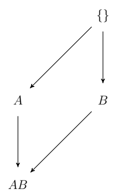
Et via un treillis de donnée pour chaque entré positif on peut compter le nombre d'occurrence de motif en faveur de l'acceptation de la ligne:



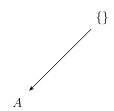
A	B	C	D	accaptable?
1	1	1	0	oui
1	1	0	1	oui
1	0	0	0	oui



A	B	C	D	accaptable?
1	1	1	0	oui
1	1	0	1	oui
1	0	0	0	oui



A	B	C	D	accaptable?
1	1	1	0	oui
1	1	0	1	oui
1	0	0	0	oui



A	B	C	D	accaptable?
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 et un attribut très discriminant, qui fait revenir le problème à seulement la valeur de A.

26.2 Set Cover Problem

Part VIII

Algorithmes pour l'inférence et les Contraintes

Chapter 27

Constraint satisfaction problem

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.

27.1 exemple simple

Trouver une assignation Minimal et Maximal pour chaque variables de P dans le problème suivant:

vars(P) $= w, x, y, z$

Dom(P)

Dom(w,x) $= \{1, 2, 3\}$
Dom(y,z) $= range(4)$

Contraintes

$w = x$

$x \leq y + 1$

$y > z$

$(x, z) \in \{(1, 2), (2, 1), (2, 4), (3, 3)\}$

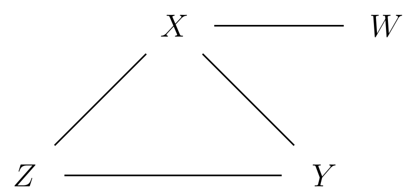
Une solution serait:

Minimal $w = x = 1, y = 3, z = 2$

Maximal $w = x = z = 3, y = 4$

27.1.1 Graphe de contraintes

Chaque variable est un nœud et chaque contraintes est représenté par une arête entre les variables concerné.



27.1.2 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: