Desfeux Nicolas
Texier Aurélien

4INFO G2.2

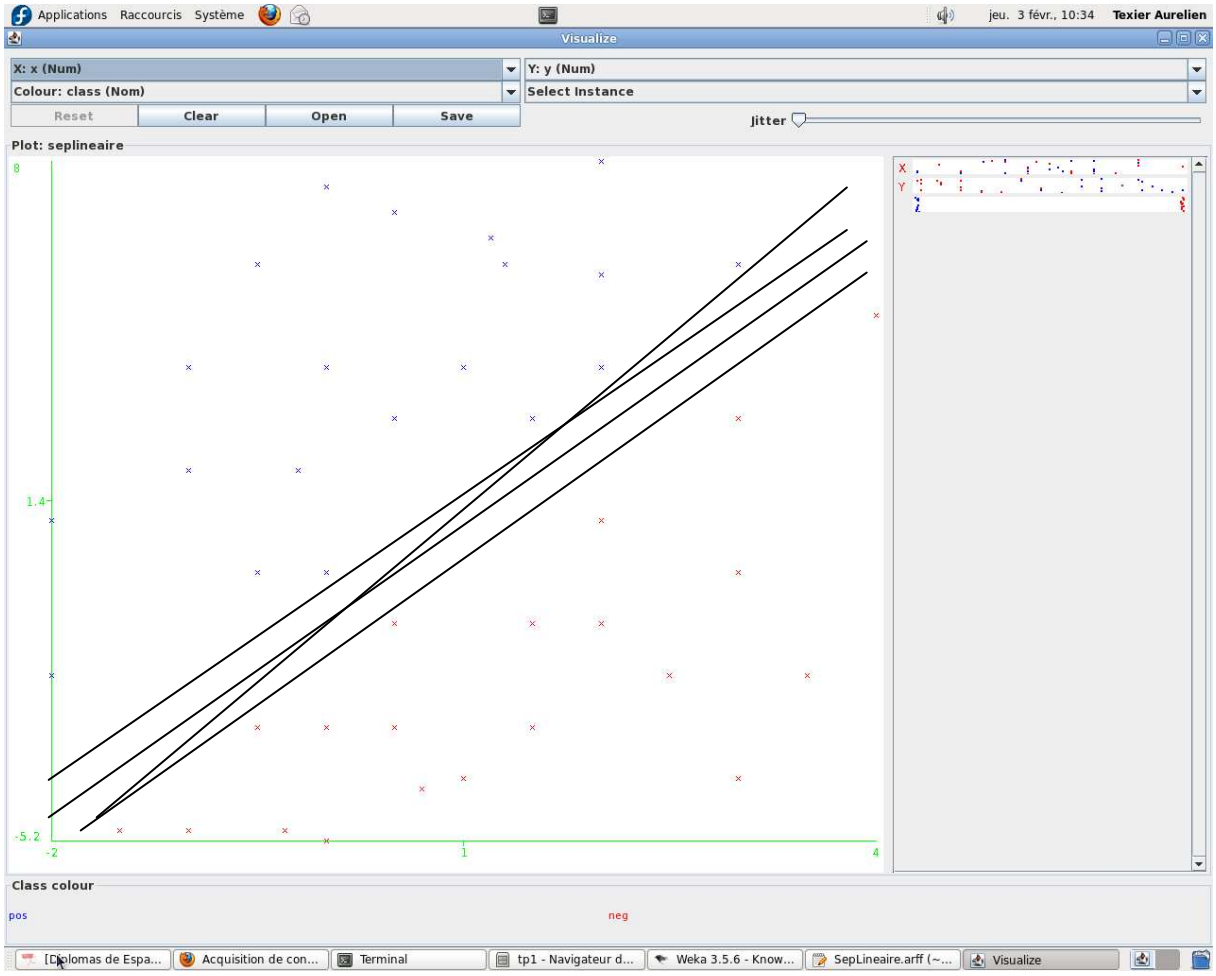
COMPTE-RENDU TP1
ACQUISITION DE CONNAISSANCES

1. Avant-propos

2. Aperçu de Weka

3. Apprentissage d'un SVM

1) Données linéairement séparables




```
=== Classifier model ===

Scheme:   SMO
Relation: seplineaire
```

```
SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: pos, neg

BinarySMO

Machine linear: showing attribute weights, not support vectors.

       1.4531 * x
 +     -0.8174 * y
 -     0.7266

Number of kernel evaluations: 116 (92.649% cached)
```

Sur le schéma, nous avons dessiné des droites qui séparent linéairement les données. Il en existe une infinité.

```
=== Evaluation result ===

Scheme: SMO
Relation: seplineaire


Correctly Classified Instances          40               100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               40

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        pos
                 1         0          1         1         1          1        neg

=== Confusion Matrix ===

  a  b   <-- classified as
 20  0 |  a = pos
  0 20 |  b = neg
```

On a ici une valeur du risque empirique qui est nulle, puisque l'on voit clairement sur le schéma que l'apprentissage et l'évaluation se font sur le même jeu de données, ce qui est une mauvaise façon de procéder puisqu'il y a un fort biais. Apprendre et tester sur les mêmes données donne forcément une erreur nulle.
Nous allons maintenant séparer les données en deux : une partie servira à apprendre pour construite le classifieur linéaire, et l'autre partie servira à tester le classifieur ainsi obtenu.

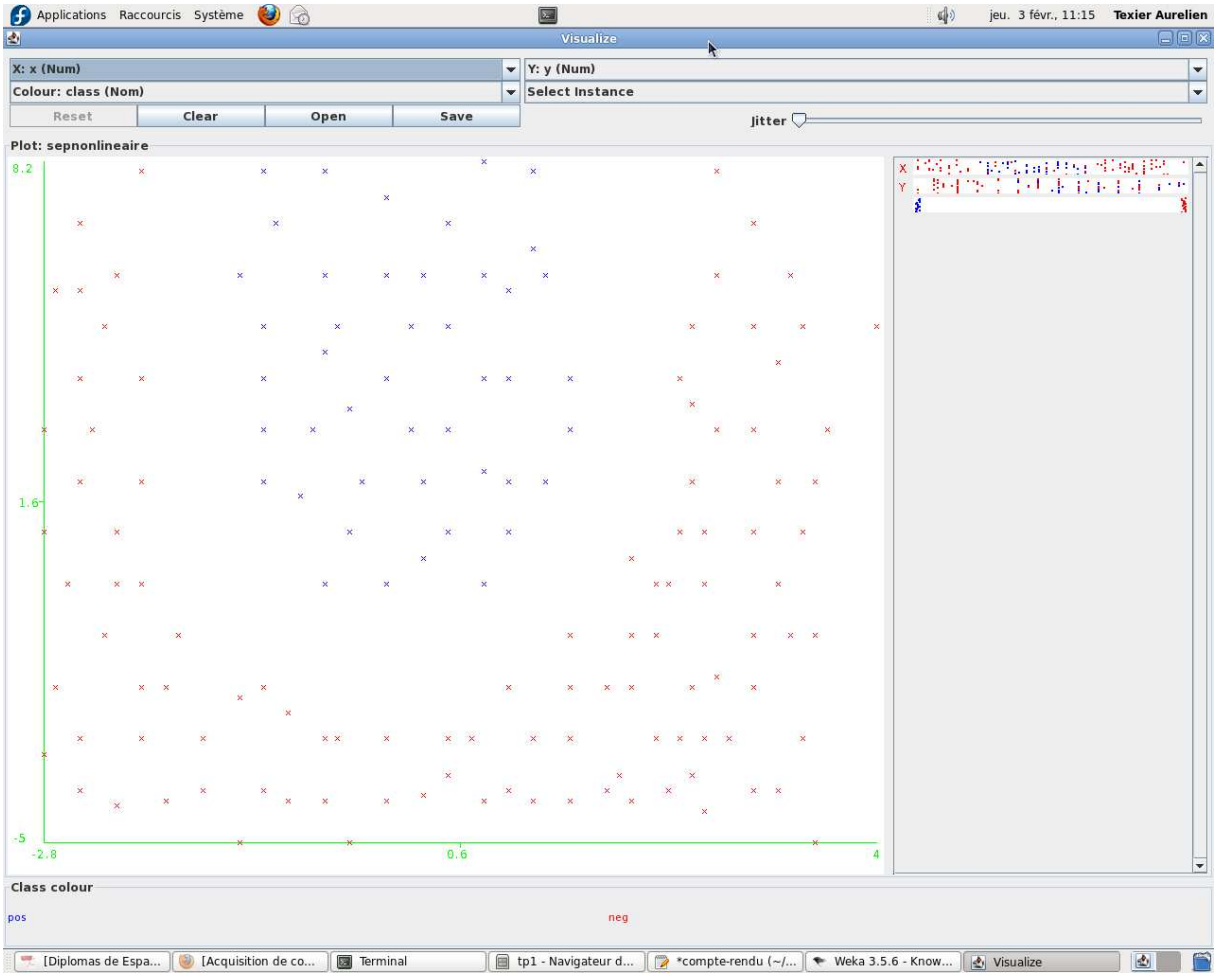
```
=== Classifier model ===

Scheme:   SMO
Relation: seplineaire

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: pos, neg

BinarySMO

Machine linear: showing attribute weights, not support vectors.

       1.0678 * (normalized) x
 +     -2.6899 * (normalized) y
 +      0.9727

Number of kernel evaluations: 106 (65.806% cached)
```

```
=== Evaluation result ===

Scheme: SMO
Relation: seplineaire


Correctly Classified Instances          13               92.8571 %
Incorrectly Classified Instances         1                7.1429 %
Kappa statistic                          0.8571
Mean absolute error                      0.0714
Root mean squared error                  0.2673
Relative absolute error                 14.5455 %
Root relative squared error             53.9974 %
Total Number of Instances               14

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.875    0          1         0.875     0.933      0.938    pos
                 1        0.125      0.857     1         0.923      0.938    neg
```

```
=== Confusion Matrix ===

 a b   <-- classified as
 7 1 | a = pos
 0 6 | b = neg
```

Le jitter ajoute du bruit aux données.
Ici, on a réussit avec ce classifieur à classer 13 instances correctement, 1 autre a été mal classée.
On a une valeur du risque empirique élevée de plus de 50%

2) Données non linéairement séparables




```
=== Evaluation result ===

Scheme: SMO
Relation: sepnonlineaire

Noyau classique (polykernel)
```

```
Correctly Classified Instances         115               76.6667 %
Incorrectly Classified Instances        35               23.3333 %
Kappa statistic                          0.4479
Mean absolute error                      0.2333
Root mean squared error                  0.483
Relative absolute error                 55.4167 %
Root relative squared error            105.4075 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.622    0.171      0.609     0.622     0.615      0.725    pos
                 0.829    0.378      0.837     0.829     0.833      0.725    neg

=== Confusion Matrix ===

  a  b   <-- classified as
 28 17 |  a = pos
 18 87 |  b = neg
```

Avec RBF :

```
=== Evaluation result ===

Scheme: SMO
Relation: sepnonlineaire


Correctly Classified Instances         133               88.6667 %
Incorrectly Classified Instances        17               11.3333 %
Kappa statistic                          0.7416
Mean absolute error                      0.1133
Root mean squared error                  0.3367
Relative absolute error                 26.9167 %
Root relative squared error             73.4619 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.889    0.114      0.769     0.889     0.825      0.887    pos
                 0.886    0.111      0.949     0.886     0.916      0.887    neg

=== Confusion Matrix ===

  a  b   <-- classified as
 40  5 |  a = pos
 12 93 |  b = neg
```

On remarque que l'augmentation de la dimension du noyau de 2 à 3 n'est pas très significative en termes de résultats car on reste à environ 55% de risque empirique absolu.
Par contre, l'adoption d'un noyau RBF améliore la classification de façon notable (27% de risque empirique absolu).

Les équations du classifieur obtenues sont beaucoup plus complexes quand on choisit un noyau de type RBF, comme on peut le constater ci-dessous :

```
=== Classifier model ===

Scheme:   SMO
Relation: sepnonlineaire

SMO

Kernel used:
  RBF kernel: K(x,y) = e^-(0.01* <x-y,x-y>^2)

Classifier for classes: pos, neg

BinarySMO


    3    * <2.5 2 > * X]
-   3    * <1 4 > * X]
-   0.132  * <0 4 > * X]
-   3    * <-1 2 > * X]
+   3    * <2.4 4 > * X]
-   3    * <1 2 > * X]
+   3    * <-2.5 2 > * X]
-   3    * <1 1 > * X]
-   3    * <0.8 8.2 > * X]
+   3    * <-2.5 4 > * X]
-   3    * <0.8 0 > * X]
+   3    * <2.7 3 > * X]
+   3    * <-2.3 5 > * X]
-   3    * <-0.6 3 > * X]
-   3    * <1.2 8 > * X]
-   3    * <0.8 6 > * X]
+   3    * <2.4 1 > * X]
+   3    * <-2 8 > * X]
+   3    * <2 0.5 > * X]
+   3    * <-2 2 > * X]
+   3    * <2.5 5 > * X]
-   3    * <0.3 2 > * X]
+   3    * <-2.2 6 > * X]
-   3    * <1.2 6.5 > * X]
+   3    * <2.5 3.5 > * X]
+   1.6191 * <1.5 -1 > * X]
-   3    * <-0.7 1.7 > * X]
+   3    * <2.6 1 > * X]
+   3    * <3.4 5 > * X]
```

- 3 * <-0.2 2 > * X]

- 3 * <0 0 > * X]

- 3 * <-0.3 3.4 > * X]

- 3 * <0.8 4 > * X]

- 3 * <1 5.7 > * X]

+ 3 * <3 3 > * X]

+ 3 * <-2.2 0 > * X]

- 3 * <1.5 3 > * X]

+ 3 * <-2.5 5.7 > * X]

- 3 * <-1 3 > * X]

- 3 * <-1 8 > * X]

- 3 * <1.3 6 > * X]

- 3 * <-1 4 > * X]

+ 3 * <2.7 8 > * X]

+ 3 * <-2.8 1 > * X]

- 3 * <0.5 3 > * X]

+ 3 * <3 5 > * X]

+ 3 * <3 7 > * X]

+ 3 * <3.2 4.3 > * X]

+ 0.747 * <-1.7 -1 > * X]

- 3 * <0.2 3 > * X]

- 3 * <-1.2 6 > * X]

- 3 * <-0.5 0 > * X]

+ 3 * <2.3 0 > * X]

+ 3 * <2.7 6 > * X]

- 3 * <0.5 1 > * X]

+ 3 * <-2 0 > * X]

+ 3 * <-2.4 3 > * X]

+ 3 * <-2.8 3 > * X]

+ 3 * <2.2 0 > * X]

- 3 * <-0.5 8 > * X]

- 3 * <0.8 2.2 > * X]

- 3 * <1.3 2 > * X]

+ 3 * <-2.5 7 > * X]

- 2.234 * <0.5 7 > * X]

- 3 * <-0.9 7 > * X]

- 3 * <0.3 0.5 > * X]

- 3 * <-0.3 1 > * X]

+ 3 * <-2.2 1 > * X]

+ 3 * <3.3 6 > * X]

+ 3 * <-2 4 > * X]

+ 3 * <-2.7 5.7 > * X]

- 3 * <1.5 4 > * X]

+ 4.2757

Number of support vectors: 72

Number of kernel evaluations: 9250 (84.342% cached)

Le nombre de vecteurs impliqués est ici de 72 (2 dans le cas du polykernel), ce qui n'est pas négligeable.

4. Apprentissage d'un arbre de décision

Données :

@ATTRIBUTE	A1	{b, a}
@ATTRIBUTE	A2	real
@ATTRIBUTE	A3	real
@ATTRIBUTE	A4	{u, y, l, t}
@ATTRIBUTE	A5	{g, p, gg}
@ATTRIBUTE	A6	{c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff}
@ATTRIBUTE	A7	{v, h, bb, j, n, z, dd, ff, o}
@ATTRIBUTE	A8	real
@ATTRIBUTE	A9	{t, f}
@ATTRIBUTE	A10	{t, f}
@ATTRIBUTE	A11	real
@ATTRIBUTE	A12	{t, f}
@ATTRIBUTE	A13	{g, p, s}
@ATTRIBUTE	A14	real
@ATTRIBUTE	A15	real
@ATTRIBUTE	A16	{+,-}

Il y a 690 instances et 15 attributs dans ces données.
Les attributs A2, A3, A11, A14, A15 sont des réels, et donc de type numériques.
Les attributs A1, A9, A10 et A12 et A16 sont de type booléens, puisqu'ils ne peuvent prendre que deux valeurs différentes chacun.
Les autres attributs sont de type nominal, ils représentent certaines caractéristiques étudiées.
L'attribut A16 constitue la classe à prédire. Il peut prendre les valeurs « + » ou « - » qui représentent si le crédit effectué par le client courant a été remboursé ou non (« + » s'il a été remboursé, « - » sinon).
On n'a pas un arbre pur (arbre qui colle bien aux données) car il n'a pas été élagué.
Avec un pourcentage d'apprentissage de 66% et une graine de 1 :

Correctly Classified Instances	198	84.2553 %
Incorrectly Classified Instances	37	15.7447 %

On peut observer que près de 85% des instances ont été bien classées.

On augmente à 80% le pourcentage d'apprentissage :

Correctly Classified Instances	125	90.5797 %
Incorrectly Classified Instances	13	9.4203 %

On peut donc avec ces résultats déduire qu'avec un pourcentage d'apprentissage plus élevé, on arrive à mieux classer les instances, ce qui est assez intuitif. En effet, plus on apprend sur un grand nombre d'instances, et plus on diminue le biais du classifieur.

Avec une graine de 2 : (66% training)

Correctly Classified Instances	200	85.1064 %
Incorrectly Classified Instances	35	14.8936 %

Avec une graine de 5 : (66% training)

Correctly Classified Instances	208	88.5106 %

```
Incorrectly Classified Instances        27               11.4894 %
```

On voit qu'avec une graine différente, on a un pourcentage d'instances bien classées qui est proche de celui avec une graine de 1. Ceci montre que le hasard n'a pas beaucoup d'influence sur les résultats obtenus, ce qui est rassurant (même si la graine de 5 montre un meilleur pourcentage de bien classés).

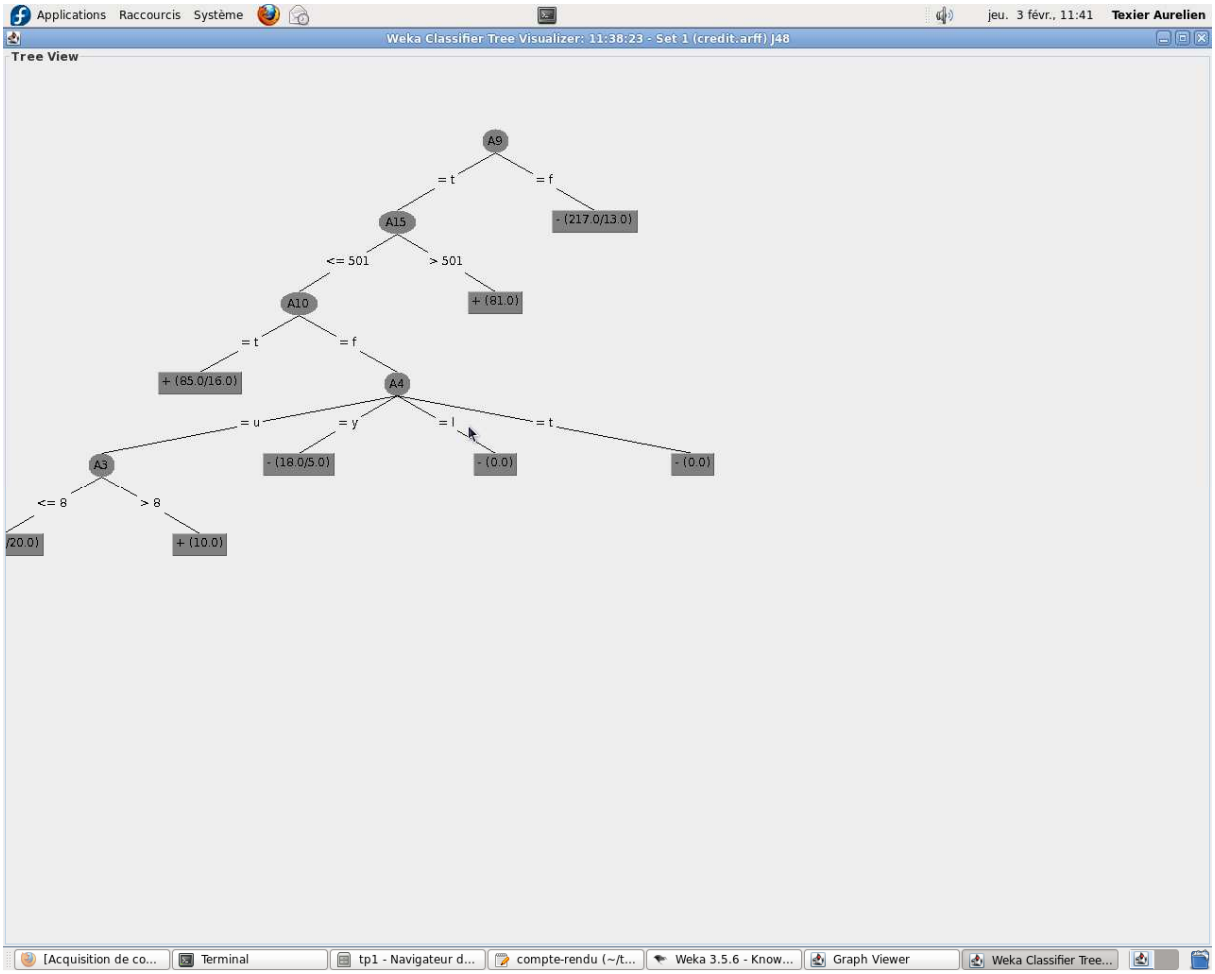
Pour améliorer la stratégie d'évaluation, il faut trouver un bon équilibre au niveau du pourcentage d'apprentissage. Trop d'apprentissage entraîne une quantité de test peu significative.

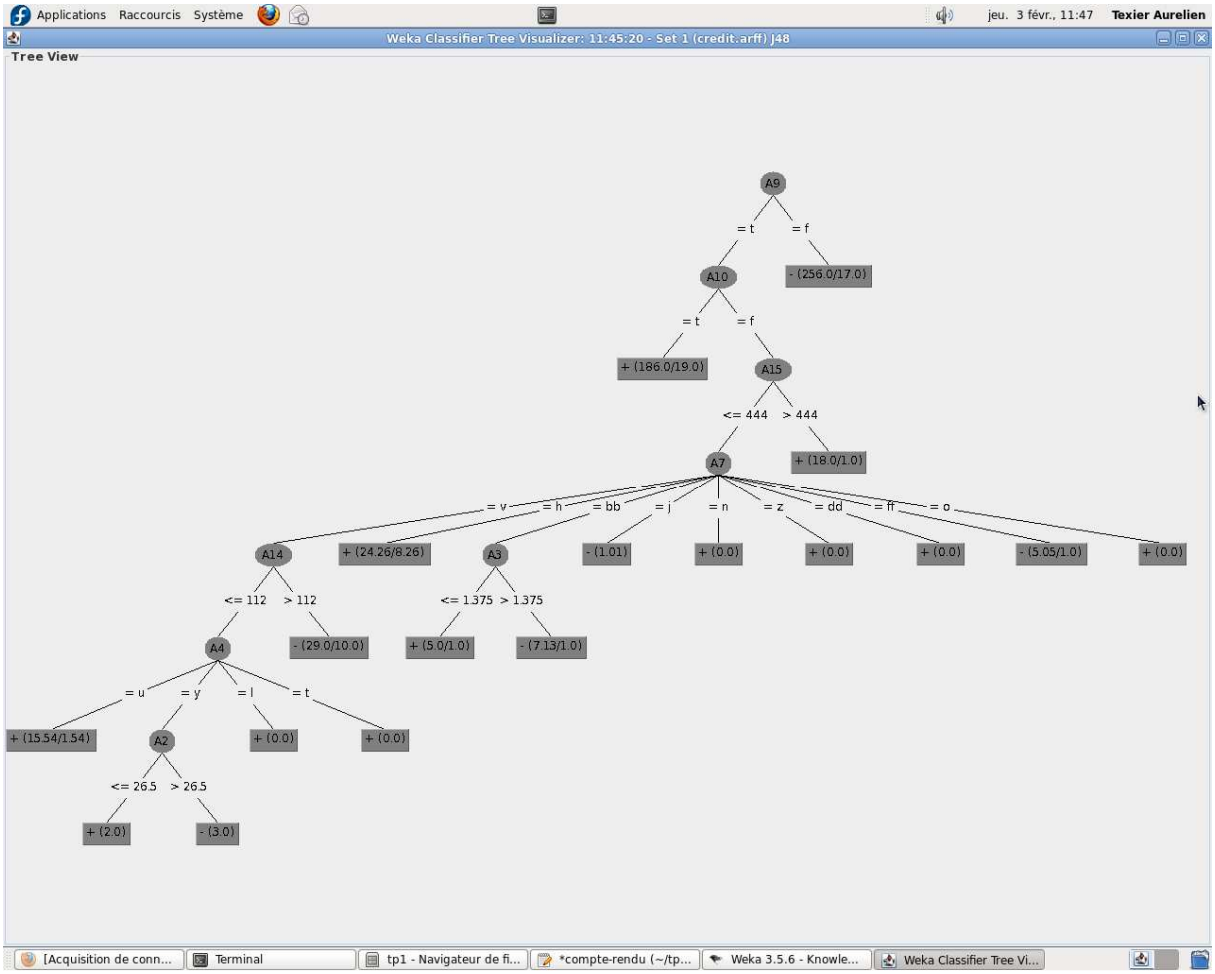
5. Comparaison de classifieurs

On choisit d'utiliser le Random Classifier avec 10 arbres.

```
=== Evaluation result ===

Scheme: RandomForest
Relation: credit.arff



Correctly Classified Instances         201               85.5319 %
Incorrectly Classified Instances        34               14.4681 %
Kappa statistic                          0.7006
Mean absolute error                      0.2509
Root mean squared error                  0.3531
Relative absolute error                 52.0752 %
Root relative squared error             71.9419 %
Total Number of Instances              235

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.832    0.129      0.814     0.832    0.823      0.889    +
                 0.871    0.168      0.884     0.871    0.878      0.888    -

=== Confusion Matrix ===

   a   b   <-- classified as
  79  16 |   a = +
  18 122 |   b = -
```

Avec 50 arbres, ça ne change pas le résultat.

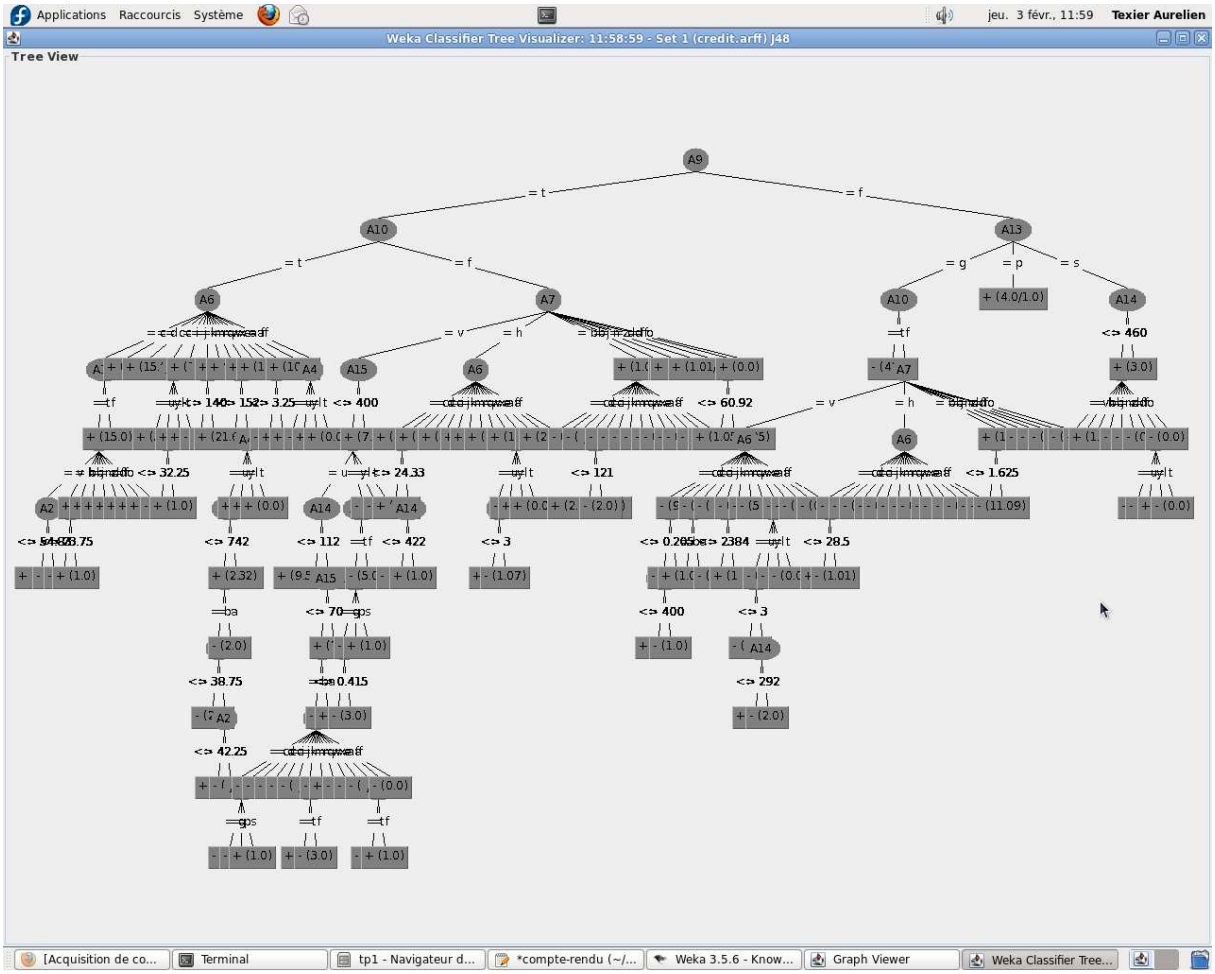

Applications Raccourcis Système
jeu. 3 févr., 11:41 Texier Aurelien
Weka Classifier Tree Visualizer: 11:38:23 - Set 1 (credit.arff) J48
Tree View
A9
= t
= f
A15
- (217.0/13.0)
<= 501
> 501
A10
+ (81.0)
= t
= f
+ (85.0/16.0)
A4
= u
= y
= l
= t
A3
- (18.0/5.0)
- (0.0)
- (0.0)
<= 8
> 8
/20.0)
+ (10.0)
[Acquisition de co...
Terminal
tp1 - Navigateur d...
compte-rendu (~/t...
Weka 3.5.6 - Know...
Graph Viewer
Weka Classifier Tree...



6. Validation

Non élagué avec minNumObj = 1

Correctly Classified Instances	193	82.1277 %
Incorrectly Classified Instances	42	17.8723 %



Non élagué avec minNumObj=2

Correctly Classified Instances	202	85.9574 %
Incorrectly Classified Instances	33	14.0426 %

Non élagué avec minNumObj=3

Correctly Classified Instances	203	86.383 %
Incorrectly Classified Instances	32	13.617 %

On observe avec des trois essais que la classification est meilleure, mais qu'en même temps le nombre de feuilles croît de façon anarchique. Ceci s'explique par le fait qu'on oblige l'arbre résultant à présenter peu de résultats par feuille, et l'on n'élague pas l'arbre résultat.

7. Fusion de classifieurs

La technique utilisée ici est inspirée du Boosting. Cette technique consiste à réitérer un certain nombre de fois l'apprentissage en augmentant le poids des instances mal classés à chaque itération. Nous avons fait la première expérience avec 3 itérations :

```
=== Evaluation result ===

Scheme: AdaBoostM1
Relation: credit.arff


Correctly Classified Instances         201               85.5319 %
```

```
Incorrectly Classified Instances       34               14.4681 %
Kappa statistic                          0.7104
Mean absolute error                      0.2194
Root mean squared error                  0.3229
Relative absolute error                 45.5297 %
Root relative squared error             65.8066 %
Total Number of Instances              235

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
               0.937    0.2      0.761     0.937    0.84       0.912    +
               0.8      0.063    0.949     0.8      0.868      0.912    -

=== Confusion Matrix ===

   a   b   <-- classified as
  89   6 |   a = +
  28 112 |   b = -
```

Puis nous avons fait une deuxième expérience avec 10 itérations :

```
=== Evaluation result ===

Scheme: AdaBoostM1
Relation: credit.arff


Correctly Classified Instances         203               86.383  %
Incorrectly Classified Instances        32               13.617  %
Kappa statistic                          0.7256
Mean absolute error                      0.1917
Root mean squared error                  0.3039
Relative absolute error                 39.7952 %
Root relative squared error             61.9212 %
Total Number of Instances              235

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
               0.926    0.179    0.779     0.926    0.846      0.936    +
               0.821    0.074    0.943     0.821    0.878      0.936    -

=== Confusion Matrix ===

   a   b   <-- classified as
  88   7 |   a = +
  25 115 |   b = -
```

On remarque alors que la différence entre les deux expériences n'est pas flagrante. On passe de 85.5% de bien classés, à 86.4%. Cependant, on observe aussi malgré tout une diminution de l'erreur absolue et de l'erreur relative quand on augmente les itérations.