TANAGRA : une plate-forme d'expérimentation pour la fouille de données

Ricco RAKOTOMALALA
Laboratoire ERIC
Université Lumière Lyon 2
5, av. Mendés France
69676 BRON cedex
e-mail : rakotoma@univ-lyon2.fr

Résumé

TANAGRA est un logiciel « open source » gratuit dédié à la fouille de données. Il s'adresse à deux types de publics. D'un côté, il présente une interface graphique aux normes des logiciels de fouille de données actuels, y compris les logiciels commerciaux, le rendant ainsi accessible à une utilisation de type « chargé d'études » sur des données réelles. De l'autre, du fait que le code source est librement disponible et l'architecture interne très simplifiée, il se prête à une utilisation de chercheurs qui veulent avant tout expérimenter de nouvelles techniques en améliorant celles déjà implémentées ou en introduisant de nouvelles. TANAGRA est opérationnel ; les versions stables sont disponibles sur le Web depuis janvier 2004 (http://eric.univ-lyon2.fr/~ricco/tanagra/). Ce site compte en moyenne une vingtaine de visiteurs par jour.

Mots-clés : TANAGRA, Logiciel « open source », Fouille de données, Expérimentation

Abstract

TANAGRA is a data mining software for practitioners and for researchers. It answers two specifications: a software with user friendly GUI for realistic studies; an "open source" software for researcher experimentations. Internal architecture is very simplified; users can easily add new features or data mining methods. TANAGRA is operative; stable versions are available on the web since January 2004 (http://eric.univ-lyon2.fr/~ricco/tanagra/). Website has about 20 visitors a day.

Keywords: TANAGRA, Open source software, Data mining, Experimentation

1 Introduction

La Fouille de Données, ou de manière générique l'Extraction de Connaissance à partir de Données (en anglais *Data Mining and Knowledge Discovery in Databases*), est un domaine de recherche qui a véritablement pris son essor au milieu des années 90. S'il est toujours possible de discuter quant à sa véritable originalité par rapport au traitement statistique des données qui existe déjà depuis longtemps (Hand et al., 2001), il est indéniable en revanche que son avènement s'est accompagné d'une forte accélération de la diffusion de logiciels spécialisés estampillés Data Mining. Les raisons sont multiples ; on pourra citer entre autres : la rencontre entre des communautés différentes (apprentissage automatique, bases de données, analyse de données, statistiques) ; le développement d'Internet qui a permis la diffusion à peu de frais des logiciels, avec pour certains des codes sources ; l'élargissement du champ d'application du traitement des données

telles que la catégorisation de textes et d'images, la bio-informatique, etc. Phénomène révélateur de cette métamorphose, des logiciels de traitement statistique ayant pignon sur rue depuis plusieurs années ont modifié leur positionnement, en se contentant bien souvent d'un remodelage de leur interface et de l'adjonction de méthodes issues de l'apprentissage automatique.

Nous pouvons classer les logiciels en deux catégories très distinctes : les logiciels commerciaux et les plates-formes d'expérimentation. Les premiers proposent une interface graphique conviviale, ils sont destinés à la mise en œuvre de traitements sur des données en vue du déploiement des résultats. Les méthodes disponibles sont souvent peu référencées, il est de toute manière impossible d'accéder à l'implémentation. Les seconds sont constitués d'un assemblage de bibliothèques de programmes. Un chercheur peut facilement accéder au code source pour vérifier les implémentations, ajouter ses propres variantes, et mener de nouvelles expérimentations comparatives. Ces plates-formes ne sont guère accessibles à des utilisateurs non informaticiens.

TANAGRA est un logiciel gratuit et « open source » de fouille de données. Il se positionne de manière très différente car il essaye de jeter un pont entre les deux approches radicalement opposées. Dès l'écriture des premières spécifications, nous tenions à ce qu'il soit à la fois convivial, accessible aux praticiens, utilisable dans les travaux dirigés des enseignements. La définition d'une interface graphique aux standards actuels était donc indispensable. Nous avons adopté la présentation sous forme de « filière » ou « diagramme de traitements » où l'enchaînement des opérations est symbolisé par un graphe orienté dans lequel transitent les données, les nœuds du graphe représentent un traitement effectué sur les données. Cependant, nous tenions également à ce que le logiciel puisse se comporter comme une plate-forme d'expérimentation dans laquelle les chercheurs puissent ajouter simplement leurs méthodes. La principale gageure a donc été la définition d'une architecture logicielle la plus simplifiée possible de manière à porter l'essentiel de l'effort au développement des méthodes. Le chercheur doit être allégé de toute la partie ingrate de la programmation de ce type de logiciel, notamment la gestion de données et la mise en forme des sorties. Point très important à nos yeux, la disponibilité du code source est un gage de crédibilité scientifique ; elle assure la reproductibilité des expérimentations publiées par les chercheurs et, surtout, elle permet la comparaison et la vérification des implémentations.

L'interface du logiciel est en langue anglaise. Il eût été possible d'élaborer un produit bilingue mais cela aurait complexifié inutilement l'architecture du logiciel. Les libellés et messages des boîtes de dialogues sont relativement simples et seront accessibles aux francophones. En revanche, la documentation est toujours disponible en français et en anglais. Le code source est commenté en français.

Dans cet article, nous présentons la philosophie du fonctionnement de TANAGRA et nous répertorierons ses principales fonctionnalités. Nous illustrons l'utilisation du logiciel sur des données réelles et synthétiques. Enfin, nous essaierons de positionner le logiciel par rapport aux outils existants.

2 Fonctionnement et principales fonctionnalités

2.1 Le diagramme de traitements

TANAGRA s'inscrit dans le paradigme actuel de la filière ou diagramme de traitements : les séquences d'opérations appliquées sur les données sont visualisées à l'aide d'un graphe.

Chaque nœud représente un opérateur soit de fouille de données, soit de modélisation, soit de transformation. Il est donc susceptible de produire de nouvelles données (les projections sur un axe factoriel par exemple). Nous le désignons également sous le terme de *composant* en référence au vocabulaire utilisé dans les outils de programmation visuelle. L'arête reliant deux nœuds représente le flux des données vers l'opérateur suivant. Ce mode de représentation qui est le standard actuel des logiciels de fouille de données autorise (au contraire des logiciels pilotés par menus) la

définition d'enchaînement d'opérations sur les données. En même temps il affranchit l'utilisateur (au contraire des outils fonctionnant avec un langage de script) de l'apprentissage d'un langage de programmation. Dans TANAGRA, seule la représentation arborescente est autorisée, la source de données à traiter est unique. La fenêtre principale du logiciel est subdivisée en trois grandes zones (Figure 1) : (a) en bas la série des composants disponibles regroupés en catégories ; (b) sur la gauche, le diagramme de traitements, représentant l'analyse courante ; (c) dans le cadre de droite, l'affichage des résultats consécutifs à l'exécution de l'opérateur sélectionné.

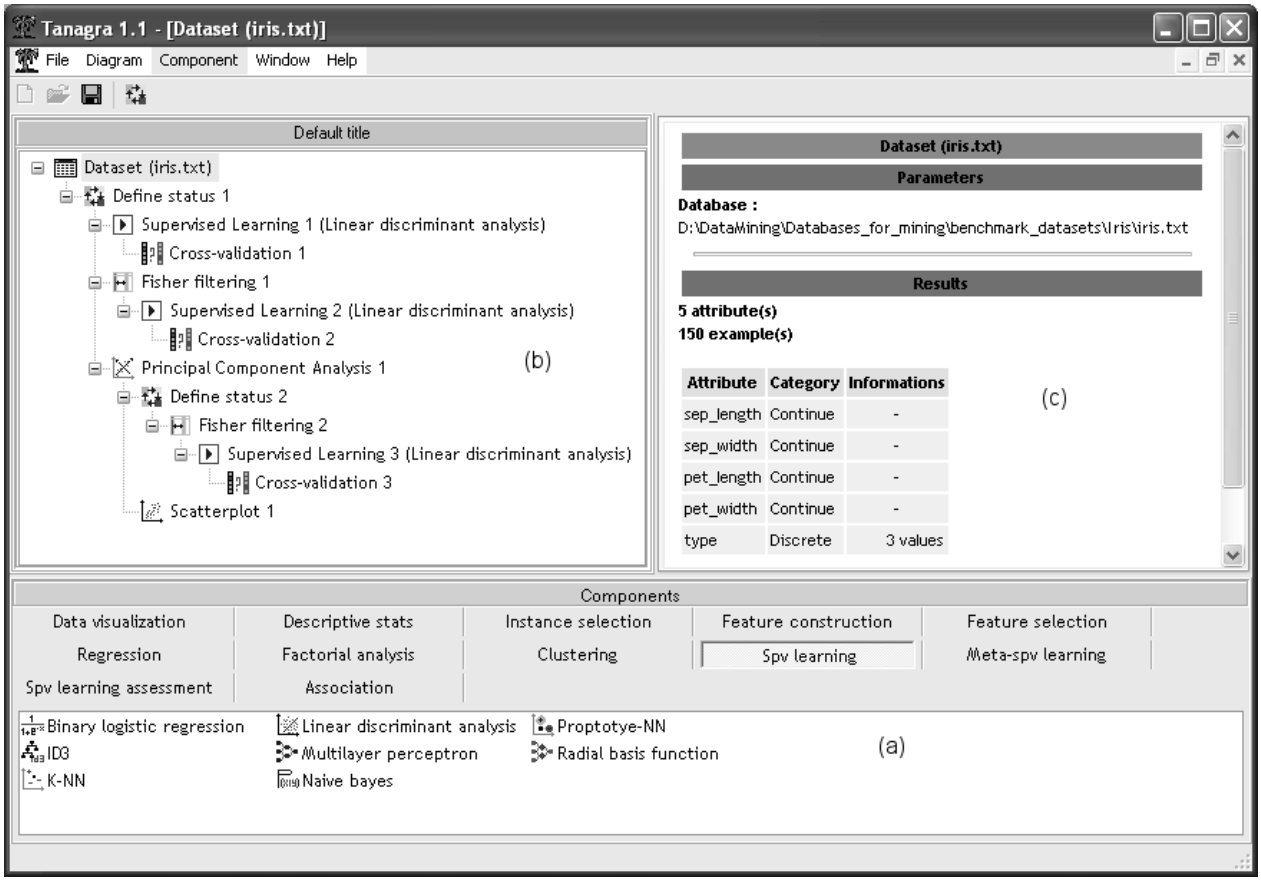

Figure 1 : La fenêtre principale du logiciel TANAGRA

La sauvegarde de la séquence d'instructions – le programme en quelque sorte – définie par un utilisateur peut être réalisée soit sous un format binaire, soit sous la forme d'un fichier texte. Seul le programme est sauvegardé, les résultats ne le sont pas. Le format texte permet à un utilisateur avancé de le manipuler directement afin de définir un nouveau diagramme de traitements. C'est le format que nous préconisons si le volume de données à traiter n'est pas trop important (voir en annexes le détail du format), en effet il suffit de relancer l'exécution pour obtenir automatiquement les nouveaux résultats lorsque les données ont été modifiées. En revanche, si la base est de grande taille, il est préférable d'utiliser le format binaire qui optimise le temps d'exécution sur les entrées-sorties. Son principal inconvénient étant la nécessité de ré-importer les données si elles sont mises à jour.

La possibilité d'enchaîner des méthodes d'apprentissage à travers le diagramme de traitements rend aisé la combinaison des méthodes sans avoir à utiliser un langage de script. Il est ainsi très facile de mettre en œuvre des méthodes spécifiques telles que la régularisation qui consiste à appliquer une analyse discriminante à partir des axes factoriels (Lebart *et al*., 2000). La plupart des logiciels commerciaux, même ceux qui disposent à l'origine d'un langage de

programmation, proposent aujourd'hui ce mode de représentation (DECISIA Spad, SAS Enterprise Miner, SPSS Clementine, Insightfull Miner, STATSOFT Statistica, etc.).

2.2 Les composants

Fonctionnement

Un composant représente un algorithme de traitement de données. Toutes les méthodes sont référencées (voir en annexe la liste des méthodes disponibles, Tableau 3). Le code source étant accessible librement, il est possible de consulter ce qui est implémenté.

Les composants ont pour point commun de prendre en entrée des données en provenance du composant qui le précède ; de procéder à des calculs donnant lieu à l'affichage d'un rapport au format HTML (Figure 3); ils sont le plus souvent paramétrables (Figure 2) ; et enfin, ils transmettent aux composants en aval les données en y ajoutant éventuellement des données produites localement, les prédictions par exemple pour les méthodes supervisées.

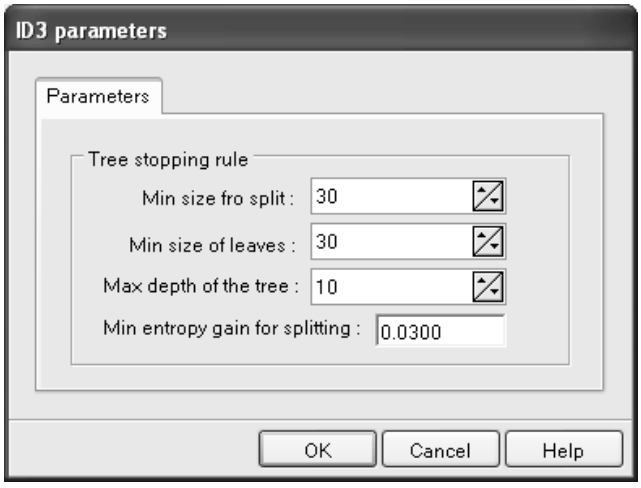

Figure 2 : Boîte de paramétrage de la méthode ID3

En termes de programmation, l'insertion d'un nouveau composant dans le logiciel est simplifiée. La procédure est toujours la même : dériver quelques classes à partir des prototypes existants, dessiner une icône appropriée, mettre à jour le fichier de configuration. La hiérarchie interne des classes respectant le découpage en famille des composants, il est aisé d'identifier rapidement la classe ancêtre adéquate. Les fonctions à ré-écrire sont indiquées dans la documentation du code source.

Les composants implémentés

Les algorithmes sont regroupés en grandes familles ; certaines peuvent être discutables mais il ne nous semblait pas approprié de multiplier les catégories. Nous distinguerons deux grandes super-familles, à savoir les algorithmes d'obédience statistique : statistique descriptive, statistique inférentielle, analyse de données et économétrie ; et les algorithmes issus de l'apprentissage automatique et bases de données : filtrage d'individus et de variables, apprentissage supervisé, règles d'association.

A l'intérieur de ces deux grands groupes, nous avons procédé à un découpage qui respecte peu ou prou la démarche de la fouille de données (Hand *et al*., 2001). En mettant à part l'accès aux données, les composants peuvent être regroupés de la manière suivante :

- La description des données : ils permettent de visualiser rapidement les données, de construire des graphiques, de procéder à des statistiques descriptives, d'effectuer des statistiques comparatives. Ces composants sont intégrés dans les onglets « Data Visualization » et « Descriptive Statistics ».
- La préparation de données : il s'agit principalement de la sélection d'individus, de variables et de la construction de variables. Ils sont rassemblés dans les onglets « Instance Selection », « Feature Selection », « Feature construction ».
- Le traitement : ces composants constituent le cœur du logiciel. Ils effectuent un traitement sur les données et produisent un modèle que l'on peut visualiser dans la page de résultats. Quasiment tous effectuent des projections qui complètent l'ensemble de données courant. Ces composants sont rassemblés dans les onglets « Regression », « Factorial Analysis », « Clustering », « Supervised Learning », « Meta-Supervised Learning ».
- L'évaluation : associés à un seul onglet pour l'instant (« Supervised Learning Assessment »), ils permettent d'évaluer le taux d'erreur de classement des méthodes supervisées d'apprentissage. Ils inscrivent les résultats de chaque exécution dans un fichier qui recense l'ensemble des évaluations réalisées avec le logiciel. Cette particularité sera mise à profit pour l'exécution en mode batch.

Bien entendu, ces découpages n'ont rien de définitif, ils peuvent évoluer au fur et à mesure que la bibliothèque des techniques s'enrichit. Il paraît peu approprié en revanche de créer un groupe pour une seule méthode, c'est pour cette raison que des méthodes statistiques telles que l'analyse de variance se trouvent actuellement dans la famille des techniques descriptives.

2.3 Standardisation des sorties

La mise en forme des sorties constitue un travail important dans les logiciels commerciaux. Chaque méthode est spécifique, il est très difficile de produire une standardisation qui permettrait de réduire l'effort de programmation. Les logiciels de recherche pêchent très souvent dans ce domaine ; la plupart se contentent de sorties en mode texte qui, pour complètes qu'elles soient, se révèlent réellement rébarbatives pour le praticien.

Il existe une tentative de normalisation XML (*Extended Markup Language*) de la représentation des modèles. La norme PMML (*Predictive Model Markup Language*) a été développée par un consortium d'entreprises fortement impliqués dans l'élaboration de logiciels de fouille de données (http://www.dmg.org). Certains logiciels comme « SPSS Clementine » exportent des modèles en s'appuyant sur ce standard. Cette norme reste encore très expérimentale ; les versions se succèdent et il est difficile de discerner ce qui est réellement stable. Cette solution constitue néanmoins l'avenir. En utilisant des feuilles de styles adéquates en XSL (*Extensible Stylesheet Language*), il sera possible de formater de différentes manières un modèle XML en sortie HTML pour disposer de présentations avenantes. Mais, au-delà de la mise en forme et de la présentation des résultats, le véritable enjeu de cette normalisation est ailleurs : grâce à elle, il sera plus facile de procéder au déploiement des modèles, et des interpréteurs « universels » pourront voir le jour.

Pour l'heure, nous nous sommes contentés de produire directement des sorties au format HTML (*Hypertext markup Language*), reconnu par tous les navigateurs. La solution est souple, autant que pour la production de sorties au format texte ; elle permet des mises en formes élaborées sans avoir à procéder à une normalisation compliquée. Dans la figure 3, nous montrons un exemple de sorties de la méthode d'induction d'arbre de décision ID3 de Quinlan (Zighed et Rakotomalala, 2000). Nous accédons à une mise en forme correcte avec un minimum d'efforts de programmation. La construction des rapports HTML constitue une fraction faible de l'implémentation des méthodes, ce qui était le but recherché.

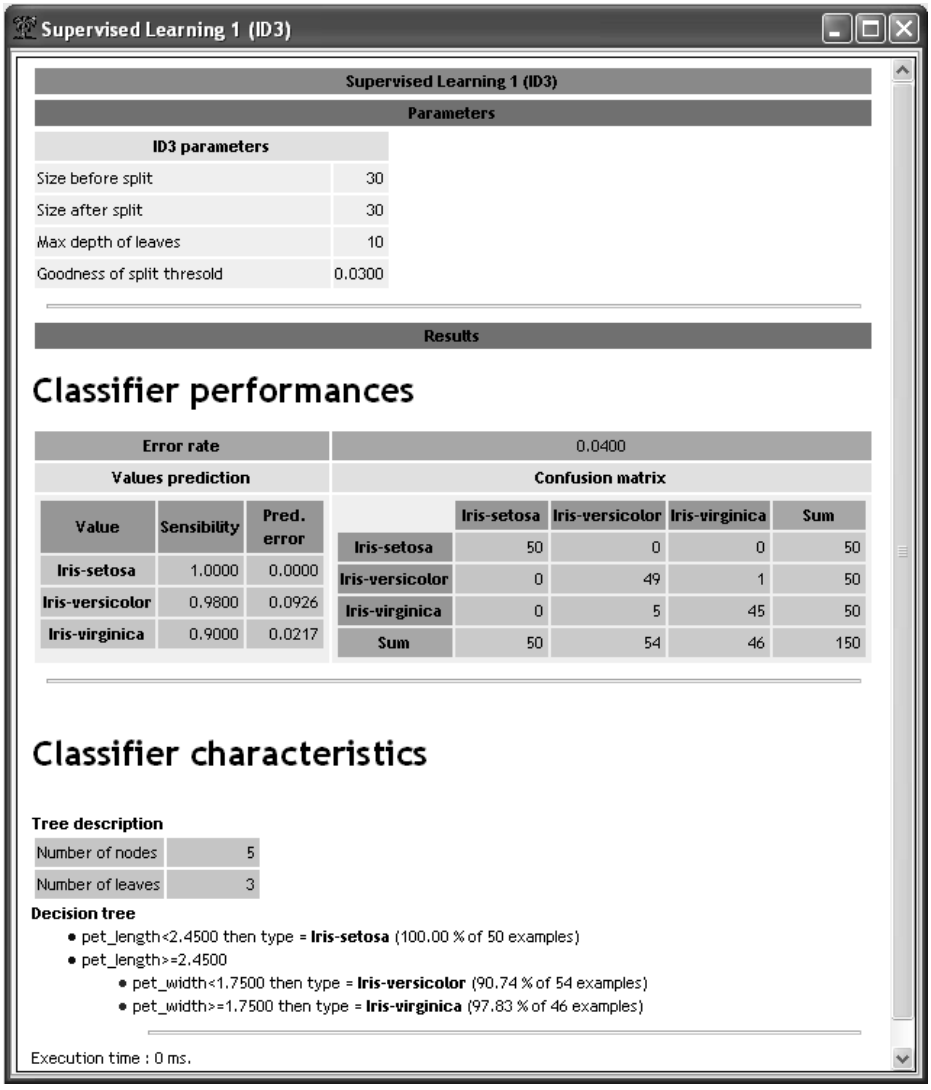

Figure 3 : Rapport au format HTML de ID3 (Arbre de décision) sur le fichier IRIS

2.4 Mode d'exécution : interactif ou par lots (batch)

A l'instar des logiciels utilisant la représentation des traitements sous forme de diagramme, TANAGRA peut être piloté de manière interactive. En s'appuyant sur les outils de l'interface, le praticien peut construire manuellement ses traitements, puis lancer l'exécution de l'ensemble. Ce diagramme peut être sauvegardé pour une exécution ultérieure ; dans ce cas le praticien a le choix entre deux options : soit il charge le diagramme via les menus adéquats puis lance l'exécution ; soit il passe le diagramme à l'exécutable via la ligne de commande.

Cette deuxième option définit un autre mode d'exécution : le traitement par lots. Dans ce cas, le logiciel charge le diagramme, l'exécute puis s'arrête, sans jamais faire apparaître la fenêtre principale du logiciel. Tous les résultats sont consignés dans des rapports au format HTML générés automatiquement. Certains composants, les évaluations de l'apprentissage supervisé notamment, ont la possibilité d'inscrire leurs résultats dans un fichier commun défini à la conception du diagramme. Ce mode d'exécution ouvre la porte aux expérimentations. En effet, il est possible de générer de nombreuses variantes d'un scénario de traitements, en modulant les paramètres d'une méthode de fouille de données. Nous l'avons mis en œuvre par exemple dans un processus du détection du nombre de descripteurs optimal pour le classement automatique de protéines à partir de leurs structures primaires (Mhamdi *et al*., 2004). La sauvegarde du diagramme au format texte se prête remarquablement bien à ce type d'étude. Grâce à sa simplicité et sa lisibilité, le chercheur peut

le manipuler aisément ; il peut aussi le générer automatiquement par programme pour les expérimentations à grande échelle.

2.5 Structures internes et performances

TANAGRA est implémenté en PASCAL OBJET, il est compilé avec la version 6 de BORLAND DELPHI. Une version gratuite de l'environnement de programmation est disponible sur le site de l'éditeur, les bibliothèques de calcul que nous avons utilisées sont référencées sur notre site web, il est aisé de recompiler le logiciel à partir du code source. L'exécutable est un fichier binaire qui fonctionne exclusivement sur le système d'exploitation WINDOWS.

Une particularité du logiciel réside dans l'obligation de charger, sous forme recodée, la totalité des données en mémoire. La gestion d'une banque interne ou encore le couplage fort avec un serveur de base de données se révèle trop exigeant en ingénierie informatique et dépasserait le cadre de la recherche sur les techniques de fouille de données. La plupart des plates-formes d'expérimentation connues ont adopté la même démarche. Une observation est codée sur 1 octet pour une variable discrète qui ne pourra donc pas prendre plus de 255 modalités ; une variable continue est codée sur 4 octets, limitée à 8 chiffres significatifs. Un fichier d'un million d'observations avec 1000 variables continues occupe donc approximativement 382 Mo en mémoire centrale. Tout dépend dès lors des problèmes que l'on souhaite appréhender. Un PC de bureau doté de 512 Mo de mémoire vive par exemple peut traiter l'ensemble des clients d'une grande banque régionale pour un ciblage marketing. En revanche, traiter l'ensemble des transactions journalières d'une enseigne de grande distribution en chargeant les données en mémoire paraît inconcevable. Certaines techniques telles que les règles d'association, très gourmandes dès lors que l'on charge tout en mémoire, s'avèrent très vite impraticables lorsque la taille de la base augmente.

Pour donner un ordre d'idées des temps de traitement, nous avons recueillis les durées d'exécution sur quelques fichiers du serveur de données UCI (Hettich et Bay, 1999) et un fichier que nous avons utilisé pour nos propres publications (Discrimination de protéines, Mhamdi *et al*, 2004). Nous avons adopté les paramétrages par défaut pour que l'expérience soit reproductible ; l'ordinateur utilisé est un PC Pentium 4 à 3.0 Ghz tournant sous Windows XP Professionnel avec 1 Go de mémoire. Les temps indiqués sont des moyennes sur 10 exécutions (Tableau 1).

Nous avons distingué le temps d'importation de la base car c'est une opération particulièrement gourmande en temps de calcul. Elle comprend l'accès au disque (le format importé est du texte avec séparateur tabulation qu'il faut donc scanner ligne à ligne) et le recodage interne. L'intérêt du fichier binaire est justement de réduire considérablement les temps d'accès par la suite, puisqu'il n'est plus nécessaire de refaire l'analyse du fichier texte lors d'un second chargement. Si l'on prend le cas du fichier « Cover Type », le chargement en mémoire dans ce cas est de 2.5 secondes, à comparer aux 223 secondes nécessaires à l'importation du fichier texte.

Fichier	**Individus**	**Variables (dont continues)**	**Importation (en secondes)**	**Traitement (en secondes)**
Breast	699	10 (9)	0.05	K-Means (2 groupes) : 0.01
Wave (1)	5000	22 (21)	0.78	ACP (production de 10 axes) : 0.26 Analyse discriminante sur 10 axes : 0.03
Adult	48842	15 (6)	9.5	Arbre de décision (ID3) : 0.95
Wave (2)	500000	22 (21)	81	ACP (production de 10 axes) : 27.0 Analyse discriminante sur 10 axes : 25.5
Cover type	581012	55 (0)	223	Arbre de décision (ID3) : 7.5
Discrimination de protéines	87	7146 (0)	6.7	Validation croisée (10-folds) : 2.9 *Comprenant :* Sélection des 10 meilleures variables (Khi-2) Modèle d'indépendance conditionnelle

Tableau 1 : Durée des traitements sur quelques fichiers de données

3 Etudes de cas

Dans cette partie, nous mettrons en avant deux études qui illustrent les utilisations types de TANAGRA. La première est assez théorique ; elle consiste à évaluer les différences de comportement entre la régression logistique et l'analyse discriminante, deux méthodes a priori semblables. La seconde sera plus dans l'optique étude « réaliste » : nous essaierons de construire une typologie à partir de comportements de vote de parlementaires au congrès américain.

3.1 Régression logistique et analyse discriminante

Il est d'usage de dire que la régression logistique et l'analyse discriminante offrent les mêmes performances dans la plupart des cas (Hastie *et al*., 2000). En effet, elles reposent sur le même biais de représentation (au sens anglo-saxon de « *bias* ») ; elles induisent des séparations linéaires dans l'espace des descripteurs. Ces mêmes auteurs ajoutent que des différences peuvent survenir lorsque les hypothèses de l'analyse discriminante sont violées. En effet, elles n'ont pas un biais de préférence identique : la régression maximise la vraisemblance, l'analyse discriminante les moindres carrés, et surtout cette dernière s'appuie sur le postulat selon lequel les matrices de variances covariances conditionnelles sont identiques. Clairement donc, même si les deux méthodes semblent théoriquement capables de trouver les mêmes solutions, l'analyse discriminante en « préfère » certaines, au risque de se tromper lorsque l'hypothèse sous-jacente n'est pas appropriée.

Dans cette section, nous allons construire deux exemples qui illustrent ces différences de comportement. Nous avons généré 2 variables (X1, X2) suivant une distribution uniforme U(0, 1), puis nous avons construit deux variables à prédire qui caractérisent deux frontières différentes dans l'espace de représentation [$\varepsilon \equiv U(0,0.1)$] :

$$Y = \begin{cases} oui, \ si \ X_2 + X_1 - 1 + \varepsilon > 0 \\ non, \text{sinon} \end{cases}$$

$$Z = \begin{cases} oui, \ si \ X_2 + 4 * X_1 - 1 + \varepsilon > 0 \\ non, \text{sinon} \end{cases}$$

Y respecte l'hypothèse d'égalité des matrices de variances covariances. Même si les distributions ne sont pas gaussiennes, le poids et la forme des nuages sont bien les mêmes de part et d'autre de la frontière (Figure 5, a). Il en est tout autrement en ce qui concerne Z où l'on voit bien que le déséquilibre est manifeste (Figure 6, a).

Nous avons construit le même diagramme de traitements (Figure 4). Dans les deux cas, il consiste à appliquer successivement les deux méthodes, puis projeter les données dans un graphique plan pour matérialiser la vraie frontière (a), les séparations induites par l'analyse discriminante (b) et la régression logistique (c).

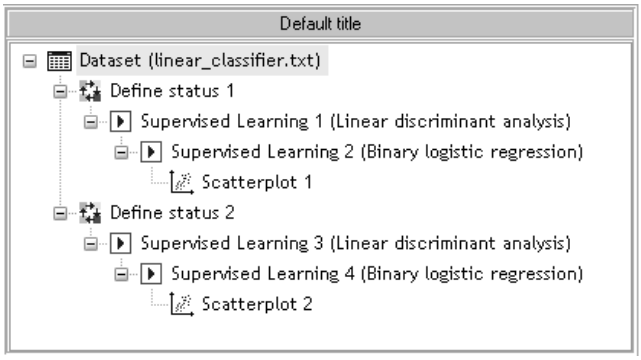

Figure 4 : Comparaison de l'analyse discriminante et de la régression logistique

Le fichier généré comporte 5000 observations. Les résultats sont tout à fait conformes à ce que l'on pouvait obtenir au regard de la littérature. En effet, pour le problème Y, les deux méthodes concordent, leur taux d'erreur, qui est quasiment le même, correspond au bruit que nous avons introduit lors de la définition de la frontière (Figure 5). Il en est tout autrement en ce qui concerne le problème Z où les préférences (les hypothèses) de l'analyse discriminante l'amène à produire une solution manifestement erronée. La régression logistique, elle, trouve le bon séparateur (Figure 6).

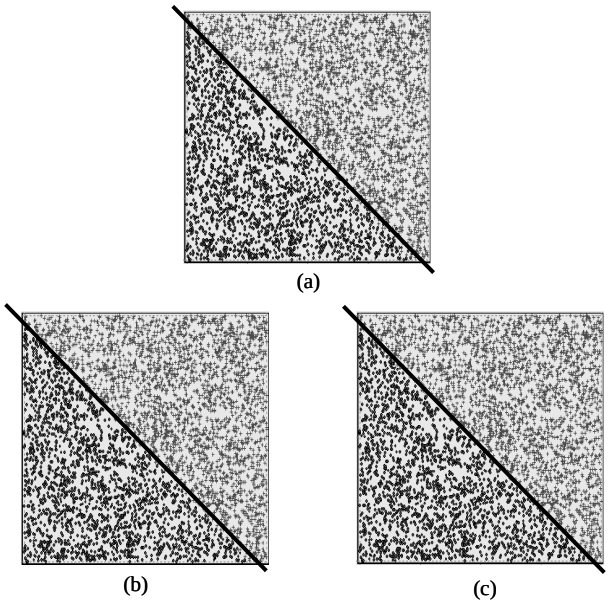

Figure 5 : Vraie frontière (a) et celles induites par l'analyse discriminante (b) - la régression logistique (c) sur un problème où le poids et la forme des sous-nuages de points sont identiques

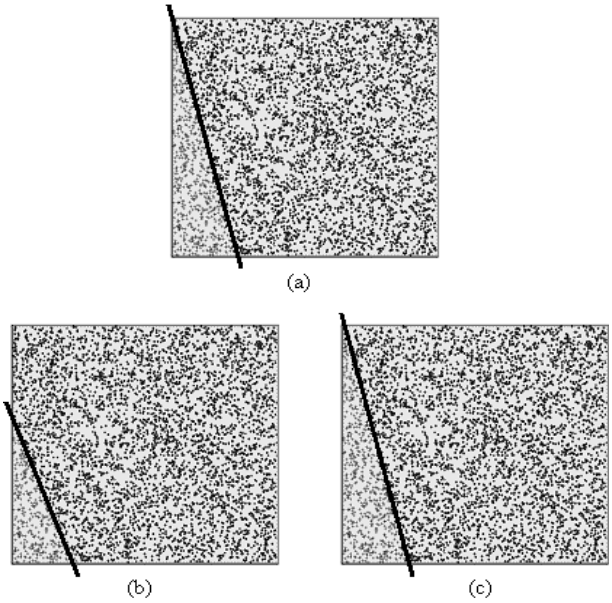

Figure 6 : Vraie frontière (a) et celles induites par l'analyse discriminante (b) - la régression logistique (c) sur un problème où le poids et la forme des sous-nuages de points sont très différents

3.2 Typologie de parlementaires

Le fichier des « Votes au congrès » est issu du serveur UCI (Hettich et Bay, 1999). Il recense les bulletins déposés par des parlementaires sur différents sujets (« oui », « non » ou « on ne sait ce qu'il a voté » qui résume en fait différents comportements). L'objectif est de construire une typologie à partir du comportement de vote et de rapprocher par la suite les groupes obtenus avec l'appartenance politique des députés (« démocrates » ou « républicains »). La base comporte 17 attributs en tout, dont l'appartenance politique, pour 435 individus. Nous procéderons de la manière suivante : (a) projeter les individus sur les axes factoriels d'une analyse en correspondance multiples en utilisant les 16 descripteurs ; (b) choisir les axes factoriels pertinents pour élaborer une typologie en deux classes à l'aide des K-Means (Lebart et al., 2000) ; (c) interpréter les groupes en introduisant entre autres l'appartenance politique comme variable illustrative. Le diagramme de traitements est décrit ci-dessous (Figure 7).

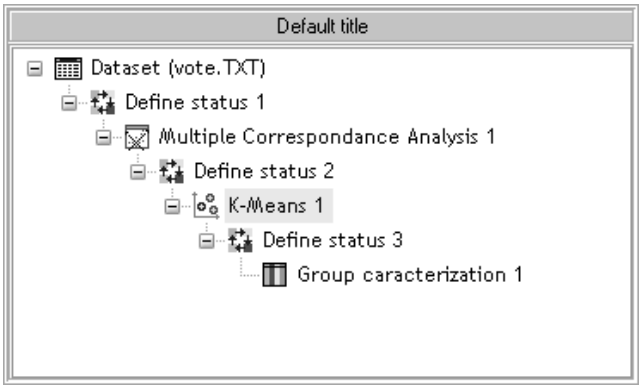

Figure 7 : Diagramme de traitements de la typologie des votes au congrès

(a) Les 5 premiers axes factoriels de l'analyse des correspondances multiples sur les 16 descripteurs prennent en compte 50% de la dispersion des observations ; nous avons décidé de les utiliser pour la construction de la typologie.

(b) Nous avons utilisé l'algorithme de McQueen en fixant à 2 le nombre de classes à produire ; il est important de ne pas normaliser les variables dans ce cas. Nous avons réalisé 5 essais de 10 itérations chacun ; la meilleure explique 40% de l'inertie totale : la première classe regroupe 197 députés (C1), la seconde 238 (C2).

(c) Reste alors à caractériser la typologie à l'aide du composant « Group characterization ». Pour ce faire, nous calculons des statistiques descriptives en classant les variables par ordre d'importance à l'aide de la valeur test (Lebart *et al.*, 2000). Nous introduisons l'attribut « appartenance politique » comme variable illustrative ; l'idée est de vérifier si le regroupement naturel correspond à un regroupement institutionnel. Cette procédure, dite de validation externe, est fréquemment utilisée par les chercheurs pour comparer les techniques de classification. Dans notre cas, les résultats sont édifiants : l'appartenance politique arrive très vite dans l'ordre des variables caractéristiques. Il semble que les classes sont avant tout caractérisées par l'attitude de vote sur les sujets tels que « l'aide au contras au Nicaragua », « l'aide au Salvador », « le gel des taxes sur la recherche physique ». Les cinq variables caractéristiques sont résumées dans le tableau suivant (Tableau 2).

Classe 1 : 197 observations					Classe 2 : 238 observations				
Variable	**Valeur**	**V-Test**	**% dans le groupe**	**% dans la base**	**Variable**	**Valeur**	**V-Test**	**% dans le groupe**	**% dans la base**
El-salvador-aid	Yes	18.1	96	49	El-salvador-aid	No	17.8	87	48
Aid-to-nicaragan-contras	No	17.1	85	41	Aid-to-nicaragan-contras	Yes	17.5	94	56
Physian-fee-freeze	Yes	17.0	85	41	Physian-fee-freeze	No	16.5	92	57
Mx-missile	No	16.1	90	47	**Class**	**Democrat**	**15.8**	**95**	**61**
Class	**Republican**	**15.8**	**79**	**39**	Adopt-budget	Yes	15.3	91	58

Tableau 2: Caractérisation de la typologie des comportements de votes

La première classe (C1) est composée en majeure partie de républicains (79%), la seconde classe (C2) compte 95% de démocrates. Il aurait été possible également d'utiliser un tableau de contingence « Composant Cross Tabulation », pour calculer la proportion de républicains (respectivement démocrates) couverts par la classe C1 (respectivement C2).

3.3 Didacticiels en ligne

Ces deux exemples montrent les possibilités offertes par TANAGRA. Plusieurs didacticiels au format PDF sont disponibles sur le site web. Pour chaque groupe de méthodes ou de fonctionnalités programmées, le lecteur trouvera une documentation articulée autour d'une étude de cas.

4 Positionnement du logiciel

Les logiciels commerciaux sont très nombreux. Leurs principaux critères de positionnement sont l'ergonomie, la capacité à appréhender différentes sources de données, la possibilité de transformer les données, l'aisance avec laquelle il est possible de combiner les approches, la richesse de la bibliothèque des méthodes, les possibilités en matière d'exploration interactive, le déploiement des modèles. Plusieurs travaux ont tenté de les comparer de manière objective (Goebel et Le Gruenwald, 1999). Ces logiciels présentent un défaut grave pour le chercheur : les méthodes ne sont pas référencées avec précision, et lorsqu'elles le sont, il est impossible de vérifier les implémentations. Ceci constitue un frein majeur aux publications car il est difficilement crédible de comparer des méthodes sans savoir exactement ce qui a été programmé. Il y a un autre point important pour les chercheurs : ces logiciels sont souvent vendus à des prix élevés. Pour avoir une idée de l'offre actuelle en logiciels, on pourra par exemple se référer au portail http://www.kdnuggets.com/software/index.html.

A l'opposé, les plates-formes d'expérimentation des chercheurs se présentent souvent comme un assemblage d'algorithmes réunis dans une même structure. Leur point commun est l'accès libre au code source ; les implémentations sont donc vérifiables. Ils sont souvent diffusés par des laboratoires de recherche spécialisés dans le domaine de la fouille de données. Dans le domaine de la statistique, la mise en commun de code source pour le traitement de données a été largement initiée depuis plusieurs dizaines d'années : des bibliothèques de code en FORTRAN sont disponibles sur de nombreux serveurs ; la communauté des statisticiens utilisent et diffusent également des bibliothèques de programmes écrits avec R qui est à la fois un langage et un environnement de développement dédié au calcul statistique (http://www.r-project.org/). Dans le domaine de l'apprentissage automatique, la diffusion des plates-formes d'expérimentation ouvertes a réellement commencé dans les années 90 avec TOOLDIAG (Rauber *et al.*, 1993), IND (Buntine, 1991), OC1 (Murthy *et al.*, 1994) et MLC++ (Kohavi et Sommerfield, 2002). Leur principale caractéristique, au-delà de la simple collection de procédures, est la définition d'une architecture

globale permettant d'ajouter de nouveaux algorithmes de manière simple, en profitant par exemple des mécanismes de la programmation objet. Le projet le plus abouti à ce jour est sans aucun doute le projet WEKA (Witten et Frank, 2000). L'avantage est manifeste pour peu que l'on ait la capacité d'appréhender les structures utilisées : la bibliothèque des méthodes est souvent de grande qualité, il est très facile de dériver un module pour des traitements particuliers, par exemple pour traiter un domaine spécifique ; l'adjonction de nouveaux algorithmes et leur comparaison avec les techniques existantes est aisée. En revanche, pour les utilisateurs non informaticiens, ces logiciels ne sont guères exploitables. Certes, certains d'entre eux tentent de se rendre plus conviviaux en plaquant par-dessus des interfaces graphiques, mais la documentation est rare et les résultats peu lisibles. Ce qui somme toute est assez compréhensible, ces logiciels sont principalement destinés à des expérimentations sur des méthodes et non à des études sur des données réelles.

TANAGRA, qui s'inscrit très clairement dans la catégorie des plates-formes d'expérimentation, tente néanmoins de pallier cette carence en se rendant accessible aux utilisateurs néophytes via une interface graphique simple, sans sacrifier la simplicité de l'architecture interne. De ce point de vue, il permet une utilisation de type « chargé d'études » car il est possible de définir des traitements assez complexes sans avoir à écrire une seule ligne de code. Ceci permet de mettre à la disposition des non-chercheurs les nouvelles techniques d'exploration des données. Bien entendu, les férus de programmation peuvent également descendre dans le code source pour programmer des expérimentations spécifiques. Dans ce cas, TANAGRA se positionne clairement comme un outil pour les chercheurs.

Au sein de notre laboratoire, TANAGRA fait suite à plusieurs projets développés depuis plusieurs années. Le plus connu d'entre eux a été le projet SIPINA piloté par D. Zighed depuis une vingtaine d'années (Rakotomalala et Zighed, 1998). Nous avons repris le développement de la version 2.5 en 1994, puis nous avons été le principal maître d'œuvre de la version recherche dont l'implémentation a réellement commencé en 1998. Au-delà de la disponibilité de l'outil sur le web (http://eric.univ-lyon2.fr/~ricco/sipina.html) et des contacts que nous avons pu nouer avec de nombreux chercheurs dans le monde, cette plate-forme nous a beaucoup servi pour développer nos propres expérimentations qui ont donné lieu à des publications. SIPINA était avant tout dédié à l'apprentissage supervisé ; il nous est apparu au fil du temps que son architecture n'était plus adaptée, notamment parce qu'il n'était pas possible d'enchaîner automatiquement des méthodes de construction et de sélection de variables. De plus, il était nécessaire pour chaque méthode ajoutée de définir une interface de visualisation spécifique. TANAGRA a donc intégré dès le départ les spécifications adéquates pour dépasser ces limitations qui étaient devenues contraignantes.

Enfin, au-delà de la position de principe, l'accès au code est primordial dès que l'on veut réellement comparer des méthodes. En effet, dans la très grande majorité des cas, deux informaticiens qui implémentent le même algorithme, à partir du même article de référence, produiront un code similaire. Cependant les résultats peuvent différer dans des situations qu'il faut pouvoir identifier. A titre d'exemple, nous citerons une anecdote rapportée par Bauer (Bauer et Kohavi, 1999) qui a testé différentes implémentations du modèle d'indépendance conditionnelle (la méthode « bayesien naïf »). Pour rappel, la probabilité d'affectation dans une classe dépend du produit des probabilités conditionnelles de chaque descripteur. Les résultats étaient comparables sur les bases « normales », mais lorsque le nombre de descripteurs candidats était très élevé, certaines implémentations donnaient des résultats erratiques. Après vérification, il s'agissait tout simplement d'un débordement de capacité sur les réels dû au produit d'un grand nombre de valeurs inférieures à 1. S'ils n'avaient pas eu accès au code, l'identification de l'erreur aurait été impossible.

5 Conclusion et perspectives

TANAGRA est un logiciel en continuelle évolution. Son architecture ouverte permet d'ajouter aisément des nouvelles techniques de fouille de données. L'accès libre au code source lui garantit sa pérennité auprès des chercheurs.

Notre premier enjeu aujourd'hui est d'assurer la diffusion du logiciel afin qu'il soit utilisé dans différents domaines, tant dans les études réelles que dans l'enseignement de la fouille de données. Les commentaires des utilisateurs nous permettent d'affiner les fonctionnalités du logiciel, améliorant ainsi son efficacité. Depuis le début de l'année 2004, nous comptons une vingtaine de visiteurs par jour sur notre site web. Le logiciel est référencé sur le principal portail anglo-saxon du data mining (http://www.kdnuggets.com).

Notre second objectif est de fédérer les bonnes volontés pour élargir la bibliothèque des méthodes de fouille de données. L'accès libre au code doit permettre à chacun de faire évoluer le logiciel à sa guise. Ce deuxième objectif est un peu plus difficile à atteindre ; seuls quelques chercheurs dans l'entourage proche de notre laboratoire l'ont réellement réalisé à ce jour.

Références

Bauer E., Kohavi R., *An empirical comparison of voting classification algorithms: Bagging, boosting and variants*. Machine Learning, Vol. 36, N°1-2, pages 105-139, 1999.

Buntine W., *About the IND tree package*, Technical Report, NASA Ames Research Center, Moffet Field, California, September 1991.

Goebel M., Le Gruenwald, *A survey of data mining and knowledge discovery software tools*, SIGKDD Explor. Newsl., Vol. 1, N°1, pages 20-33, ACM Press, 1999.

Hand D., Manilla H., Smyth P., *Principles of data mining*, Bardford Books, 2001.

Hettich S., Bay S., *The UCI KDD Archive* [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

Kohavi R., Sommerfield D., MLC++. In Will Klosgen and Jan M. Zytkow, editors, *Handbook of Data Mining and Knowledge Discovery*, chapter 24.1.2, pages 548-553. Oxford University Press, 2002.

Lebart L., Morineau A., Piron M., *Statistique exploratoire multidimensionnelle*, Dunod, 2000.

Mhamdi F., Elloumi M., Rakotomalala R., *Textmining, feature selection and data mining for protein classification*, International Conference on Information and Communication Technologies, pages 457-458, 2004.

Murthy S.K., Kasif S., Salzberg S., *A System for Induction of Oblique Decision Trees*, Volume 2, pages 1-32, 1994.

Rakotomalala R., Zighed D.A., SIPINA_W : *une plate-forme logicielle pour l'extraction automatique de connaissances à partir de données*, Journée ECD-ESIEA « Extraction de Connaissances à partir de Données », Centre de Recherche du Groupe ESIEA, Paris, Mars 1998.

Rauber, T. W., Barata, M. M., Garçao, A. S., *A Toolbox for Analysis and Visualization of Sensor Data in Supervision*. In: Tooldiag `93-International Conference on Fault Diagnosis, Toulouse, France:, Vol. 3. pages 906 – 913, 1993.

Hastie T., Tibshirani R., Friedman J., *The elements of statistical learning - Data Mining, inference and prediction*, Springer, 2001.

Witten I., Frank E., *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco, 2000.

Zighed D., Rakotomalala R., *Graphes d'Induction : Apprentissage et Data Mining*, Hermès, 2000.

ANNEXES

Liste des méthodes disponibles à ce jour

Toutes les méthodes implémentées sont associées à au moins une publication. Lorsque nous n'avons pas pu avoir connaissance de l'article original, nous citons l'ouvrage à partir duquel nous avons extrait l'algorithme. Toutes les références sont disponibles sur le site web. De manière générale, 5 ouvrages nous ont inspirés quant à l'élaboration et l'organisation des techniques de fouille de données :

- T. Hastie, R. Tibshirani, J. Friedman, *The elements of statistical learning* - Data Mining, inference and prediction, Springer, 2001.
- L. Lebart, A. Morineau et M. Piron, *Statistique exploratoire multidimensionnelle*, Dunod, 2000.
- T. Mitchell, *Machine learning*, Mc Graw-Hill International Editions, 1997.
- G. Saporta, *Probabilités, analyse des données et statistique*, Technip, 2000.
- I. Witten, E. Frank, *Data mining - Practical machine learning tools and Techniques with JAVA implementations*, Morgan Kaufmann Publishers, 2000.

Groupe	Méthode	Description
Data visualization	Correlation scatter plot	Calcule et représente dans le plan, la corrélation d'un groupe de variables avec l'abscisse et l'ordonnée
	Scatter plot	Graphique XY
	View Multiple Scatter plot	Calcule et représente dans le plan les moyennes des différentes valeurs d'une ou plusieurs variables discrètes
	Export Dataset	Exporte les données au format texte
	View Dataset	Voir les données dans une grille
Descriptive stats	Cross tabulation	Tableau de contingence et indicateurs associés
	Linear correlation	Corrélation linéaire de Pearson
	Non parametrical test	Test de Kruskall et Wallis
	Univariate Continuous stat	Statistiques descriptives pour les attributs continus (moyenne, écart-type, min, max, coefficient de variation)
	Group characterization	Statistiques descriptives conditionnelles, calcul de la valeur test
	More univariate continuous stat	Autres statistiques descriptives pour les attributs continus (médiane, quantile, etc.)
	One-way ANOVA	ANOVA à 1 facteur
	Univariate discrete stat	Statistiques descriptives pour les attributs discrets
Instance selection	Sampling	Echantillonnage simple
	Stratified sampling	Echantillonnage stratifié
	Recover examples	Inverse la sélection courante
Feature construction	0_1_binarize	Codage disjonctif complet
	EqWidth disc	Discrétisation avec des intervalles de largeur égales
	MDLPC	Discrétisation supervisée
	Trend	Construction d'un attribut « tendance » (valeurs 1, 2, 3, etc.)
	EqFreq disc	Discrétisation avec des intervalles de même effectif
	Formula	Construire manuellement une variable à partir d'une formule
	Standardize	Normalisation d'une ou plusieurs variables
Feature selection	CFS filtering	Sélection supervisée des descripteurs discrets
	Define status	Définir le rôle des attributs dans l'analyse
	Fisher filtering	Sélection supervisée des descripteurs continus
	MODTree	Sélection supervisée des descripteurs discrets
	CHI-2 filtering	Sélection supervisée des descripteurs discrets
	FCBF filtering	Sélection supervisée des descripteurs discrets
	MIFS filtering	Sélection supervisée des descripteurs discrets

	Remove constant	Supprime de la sélection courante les variables constantes
Regression	Multiple linear regression	Régression linéaire multiple
Factorial analysis	Multiple correspondence analysis	Analyse des correspondances multiples
	Principal component analysis	Analyse en composantes principales
Clustering	HAC	Classification ascendante hiérarchique mixte
	Kohonen-SOM	Cartes de Kohonen
	K-Means	Nuées dynamiques (McQueen et Forgy)
	LVQ	Learning vector quantizer
Spv Learning	Binary logistic regression	Régression logistique binaire (la variable à prédire ne peut comporter que 2 modalités)
	K-NN	K plus proches voisins
	Multi-layer perceptron	Perceptron multi-couches
	Prototype-NN	Utilise des noyaux définis par classification comme prototypes dans une algorithme de plus proches voisins
	ID3	Arbre de décision
	Linear discriminant analysis	Analyse discriminante linéaire
	Naïve bayes	Modèle d'indépendance conditionnelle
	Radial basis function	Réseau de neurones avec noyau à base radiale
Meta-spv Learning	Arcing [Arc-x4]	Méthode à base de vote
	Boosting	Méthode à base de vote
	Bagging	Méthode à base de vote
	Supervised learning	Apprentissage supervisé unique
Spv-learning assessment	Cross-validation	Validation croisée
	Train-test	Schéma apprentissage-test
Association	A priori	Construction des règles d'association

Tableau 3 : Liste des techniques implémentées (novembre 2004)

Format de fichier de sauvegarde (texte)

Le fichier de sauvegarde au format texte, extension « TDM » dans TANAGRA, respecte les spécifications des fichiers INI Windows. Dans l'exemple ci-dessous, nous appliquons une analyse discriminante sur le fichier IRIS de Fisher (Figure 8).

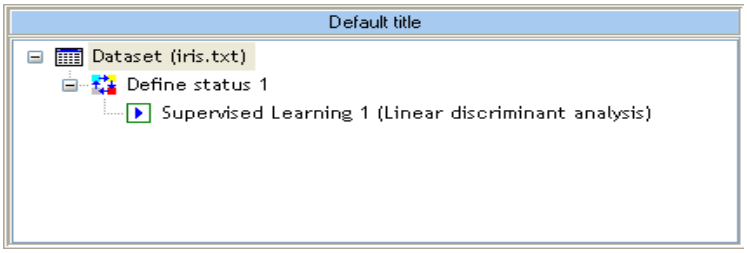

Figure 8 : Analyse discriminante sur le fichier IRIS

```
[Diagram]
Title=Default title
Database=D:\DataMining\Databases_for_mining\benchmark_datasets\Iris\iris.txt

[Dataset]
MLClassGenerator=TMLGenDataset
successors=1
succ_1=Define status 1

[Define status 1]
MLClassGenerator=TMLGenFSDefStatus
target_count=1
target_1=type
input_count=4
input_1=sep_length
input_2=sep_width
input_3=pet_length
input_4=pet_width
illus_count=0
successors=1
succ_1=Supervised Learning 1 (Linear discriminant analysis)

[Supervised Learning 1 (Linear discriminant analysis)]
MLClassGenerator=TMLGCompOneInstance
embedded_spv=1
embedded_section=Supervised Learning 1 (Linear discriminant analysis)--Linear
discriminant analysis
successors=0

[Supervised Learning 1 (Linear discriminant analysis)--Linear discriminant analysis]
MLClassGenerator=TMLGCompLDiscAnalysis
```

Figure 9: Fichier TDM du traitement du fichier IRIS

Nous observons (Figure 9) que la section DIAGRAM décrit les paramètres globaux du diagramme, dont le fichier de données. La suivante, DATASET, indique la classe interne qui génère l'icône, il permet surtout de spécifier les composants situés en aval : « successors » indique le nombre de successeurs, « succ_1 » le nom du premier composant situé en aval. Il en est de même pour les autres sections correspond chacune à un composant.

Même s'il fonctionne et permet de décrire toute l'arborescence, ce format de description n'est pas satisfaisant. En effet, la structure est retranscrite à travers un subterfuge provisoire. A l'avenir, il serait souhaitable d'utiliser un format qui s'y prête naturellement. Le format XML semble tout indiqué.