Objectif

Montrer comment, à l'aide de TANAGRA, lancer une série de traitements automatiques en passant les diagrammes par la ligne de commande.

Cette fonctionnalité s'avère surtout intéressante quand nous souhaitons effectuer un grand nombre de tests. C'est le cas par exemple lorsque nous voulons comparer les performances de différents algorithmes sur un même fichier ; rechercher automatiquement les paramètres les plus performants pour une méthode ; répéter le même traitement sur différents ensembles de données, etc.

Dans ce cadre, il est plus que souhaitable de sauver les diagrammes en mode texte (format TDM), il sera ainsi plus aisé de les manipuler en dehors de TANAGRA, avec un éditeur de texte par exemple.

Expérimentation

Nous nous sommes proposés d'évaluer les performances de la méthode de sélection de variables FCBF en apprentissage supervisé (http://www.public.asu.edu/~huanliu). Cette méthode ne fonctionne qu'en présence de variables discrètes, il cherche à isoler un ensemble de descripteurs qui sont le plus corrélés, au sens de l'information mutuelle, à la variable à prédire, tout en étant le moins corrélés entre eux.

Nous avons utilisé trois fichiers de données disponibles sur le site UCI (http://kdd.ics.uci.edu/) : VOTE, KR-VS-KP, et SPLICE. Aux descripteurs originels, nous avons adjoint des descripteurs générés au hasard et des descripteurs qui leur sont corrélés. Nous voulons comparer les performances du modèle bayesien naïf, très sensible à la dimensionnalité, sans et avec le processus de sélection.

Etapes de l'expérimentation

Définir le diagramme de traitements réalisant la comparaison

Dans un premier temps, nous avons construit manuellement les diagrammes de traitements, puis nous les avons sauvés dans le format texte TDM. Ci dessous le diagramme correspondant au fichier vote et le fichier TDM associé.

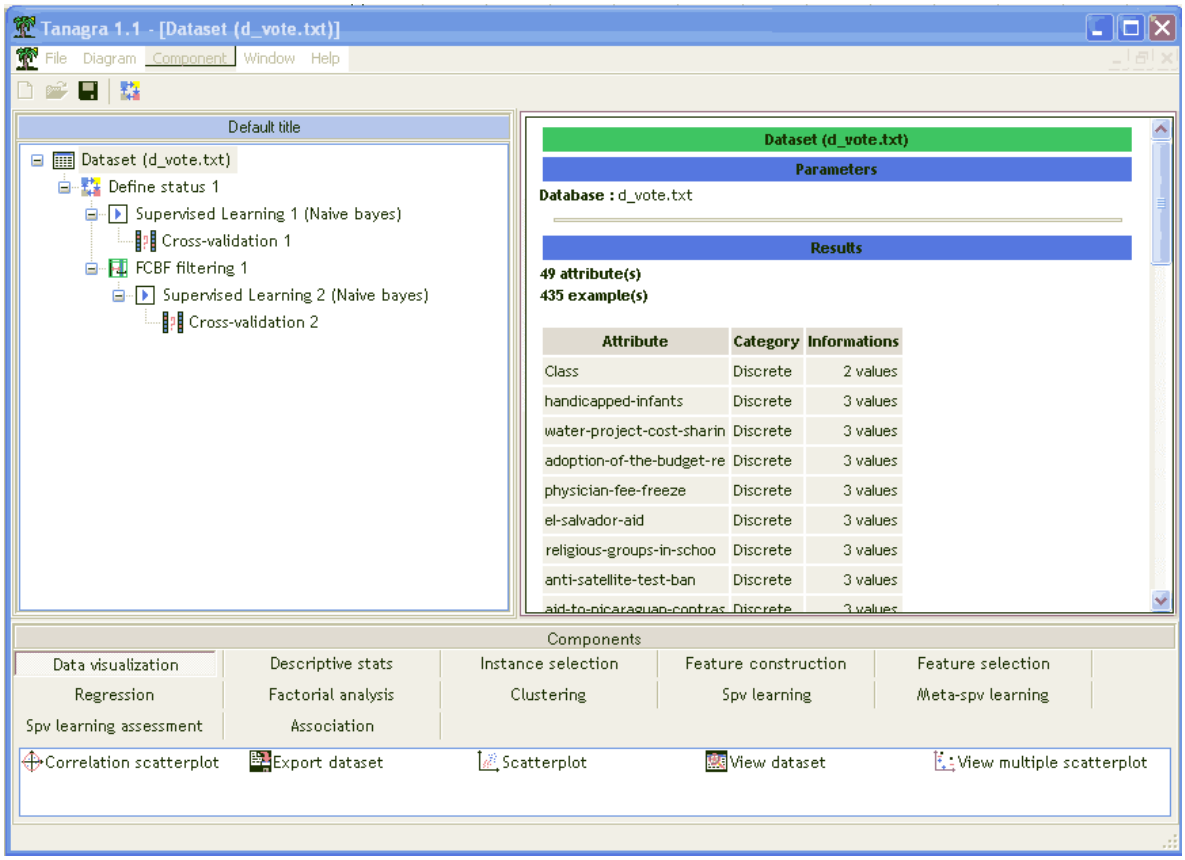

Figure 1 : Diagramme comparant les performances du bayesien naïf, sans et avec la sélection FCBF

```
[Diagram]
Title=Default title
Database=d_vote.txt

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

[Define status 1]
MLClassGenerator=TMLGenFSDefStatus
target_count=1
target_1=Class
input_count=48
input_1=handicapped-infants
input_2=water-project-cost-sharin
input_3=adoption-of-the-budget-re
input_4=physician-fee-freeze
input_5=el-salvador-aid
input_6=religious-groups-in-schoo
input_7=anti-satellite-test-ban
input_8=aid-to-nicaraguan-contras
input_9=mx-missile
input_10=immigration
```

```
input_11=synfuels-corporation-cutb
input_12=education-spending
input_13=superfund-right-to-sue
input_14=crime
input_15=duty-free-exports
input_16=export-administration-act
input_17=noise1
input_18=noise2
input_19=noise3
input_20=noise4
input_21=noise5
input_22=noise6
input_23=noise7
input_24=noise8
input_25=noise9
input_26=noise10
input_27=noise11
input_28=noise12
input_29=noise13
input_30=noise14
input_31=noise15
input_32=noise16
input_33=corr1
input_34=corr2
input_35=corr3
input_36=corr4
input_37=corr5
input_38=corr6
input_39=corr7
input_40=corr8
input_41=corr9
input_42=corr10
input_43=corr11
input_44=corr12
input_45=corr13
input_46=corr14
input_47=corr15
input_48=corr16
illus_count=0
successors=2
succ_1=Supervised Learning 1 (Naive bayes)
succ_2=FCBF filtering 1

[Supervised Learning 1 (Naive bayes)]
MLClassGenerator=TMLGCompOneInstance
embedded_spv=1
embedded_section=Supervised Learning 1 (Naive bayes)--Naive bayes
successors=1
succ_1=Cross-validation 1

[Supervised Learning 1 (Naive bayes)--Naive bayes]
MLClassGenerator=TMLGCompNaiveBayes
```

```
[Cross-validation 1]
MLClassGenerator=TMLGenCompAssesCV
isSaveResults=1
results_filename=experiments.txt
nb_repetitions=5
nb_folds=2
successors=0

[FCBF filtering 1]
MLClassGenerator=TMLGenFSFcbf
delta=0
successors=1
succ_1=Supervised Learning 2 (Naive bayes)

[Supervised Learning 2 (Naive bayes)]
MLClassGenerator=TMLGCompOneInstance
embedded_spv=1
embedded_section=Supervised Learning 2 (Naive bayes)--Naive bayes
successors=1
succ_1=Cross-validation 2

[Supervised Learning 2 (Naive bayes)--Naive bayes]
MLClassGenerator=TMLGCompNaiveBayes

[Cross-validation 2]
MLClassGenerator=TMLGenCompAssesCV
isSaveResults=1
results_filename=experiments.txt
nb_repetitions=5
nb_folds=2
successors=0
```

La lecture du diagramme à partir du fichier TDM est relativement facile, il respecte le format INI, chaque section correspond à un composant de la chaîne de traitements.

Les composants validation croisée (en italique) jouent un rôle important dans le processus d'évaluation, en effet ils inscrivent automatiquement leurs résultats dans le même fichier de sortie « experiments.txt ». Les résultats sont donc collectés au fur et à mesure de l'exécution de chaque branche du diagramme. « Cross validation 1 » calcule le taux d'erreur du processus d'apprentissage sans la sélection de variables, « Cross validation 2 » en revanche intègre FCBF. Si la sélection est efficace, on s'attend à ce que les taux calculés dans le second cas soient meilleurs.

Il ne nous reste plus qu'à définir les diagrammes de traitements pour chaque fichier à analyser en utilisant le même modèle (kr-vs-kp.tdm et splice.tdm).

Construire le script d'exécution

La seconde étape consiste alors à définir le fichier script qui permettra de lancer l'ensemble des diagrammes. Sous WINDOWS, il s'agit d'un fichier .BAT très simple. Sa forme doit être proche de notre exemple ci-dessous, en veillant tout simplement à ce que les chemins soient corrects (experiments.bat).

```
d:\temp\exe\tanagra vote.tdm
d:\temp\exe\tanagra splice.tdm
d:\temp\exe\tanagra kr-vs-kp.tdm
```

Lire les résultats

Après l'exécution du script, les résultats sont automatiquement retranscrits dans les rapports au format HTML. Il ne reste plus qu'à les ouvrir.

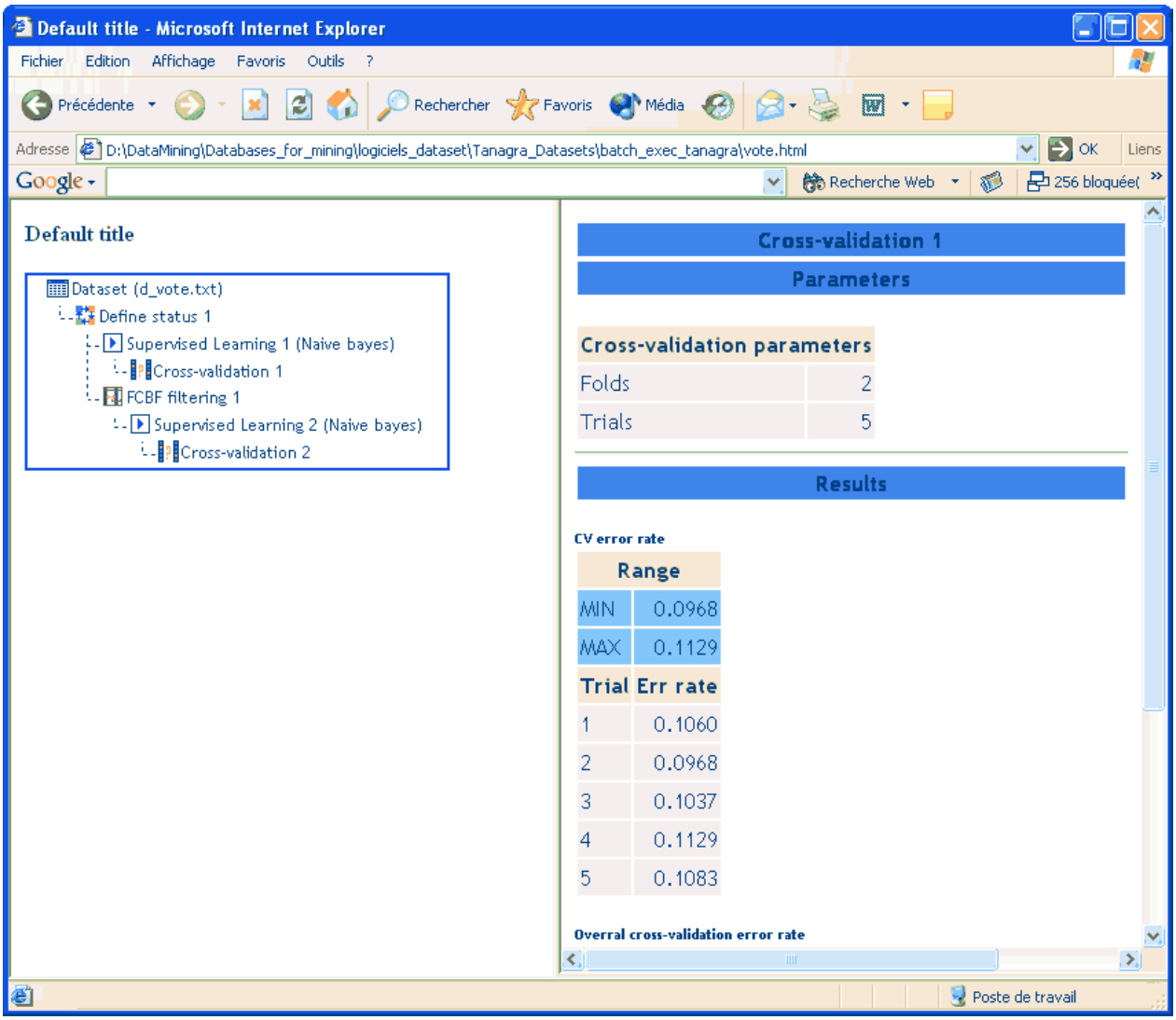

Figure 2 : Rapport HTML pour le fichier "vote.tdm"

Concernant notre exemple, nous disposons de surcroît du fichier de résultats généré par la validation croisée. Chaque ligne correspond au résultat d'une exécution. Dans notre modèle de diagrammes, les composants n°1 correspondent à l'apprentissage sans processus de sélection de variables, les composants n°2 intègrent FCBF.

Nous observons de gauche à droite : le nom du diagramme de traitements, le fichier de données utilisé, la date et l'heure d'exécution du composant, le composant, et le taux d'erreur qui a été collecté.

Diagramme	Données	Date exécution	Composant	Taux d'erreur
vote.tdm	d_vote.txt	15/10/2004 19:47	Cross-validation 1	0.094931
vote.tdm	d_vote.txt	15/10/2004 19:47	Cross-validation 2	**0.058986**
splice.tdm	d_splice.txt	15/10/2004 19:47	Cross-validation 1	0.065705
splice.tdm	d_splice.txt	15/10/2004 19:47	Cross-validation 2	**0.04721**
kr-vs-kp.tdm	d_kr-vs-kp.txt	15/10/2004 19:47	Cross-validation 1	0.133229
kr-vs-kp.tdm	d_kr-vs-kp.txt	15/10/2004 19:47	Cross-validation 2	**0.07985**

Dans ces fichiers de données, qui ont été délibérément choisis, la sélection induit systématiquement une amélioration manifeste des performances de l'apprentissage supervisé. Ce n'est pas toujours aussi idyllique en général.

Extensions

Nous imaginons aisément les extensions possibles d'un tel outil, nous pourrions, par exemple, inscrire dans le fichier de résultats le nombre de descripteurs effectivement sélectionnés. En réalité, il y a autant d'extensions possibles qu'il y a de préoccupations de chercheurs. Ces développements sont trop spécifiques, l'accès au code source permettra à tout un chacun d'introduire ses spécificités.

De même, cet outil prendra toute son ampleur si nous avons la possibilité de générer automatiquement les fichiers TDM. Cela est particulièrement intéressant par exemple lorsque l'on veut faire varier le paramètre d'une méthode dans le but de déterminer sa valeur optimale. Le format TDM étant assez accessible, écrire un petit programme qui le génère automatiquement ne devrait pas poser de problèmes.