PLDAC

Rapport de projet

Ma recette santé ! Étude expérimentale du transfert analogique

---

Étudiantes : *Numéros :*
Balkis Bouthaina DIRAHOUI 21113733
Sarah DJOUDER 3970876
Tinhinane CHAFAI 3802927

Encadré par :
Marie-Jeanne Lesot

Mai 2022

Table des matières

Chapitre 1

Introduction

L'analogie est un mécanisme cognitif puissant que les gens utilisent pour faire des inférences et apprendre de nouvelles abstractions. Ce principe cognitif a été transposé en intelligence artificielle pour réaliser des tâches de classification et de régression dans le cadre de **l'analogie computationnelle [6].** Cette dernière repose sur le principe que deux données proches sont associées à des prédictions proches.

Notre projet a pour thème l'une des tâches mises en œuvre par l'analogie computationnelle : **le transfert analogique**. Ce dernier opère un type spécial d'inférence plausible, selon laquelle deux situations jugées similaires sont susceptibles d'avoir des résultats similaires. Le transfert analogique est une tâche d'apprentissage dans laquelle une nouvelle donnée est comparée avec des données similaires dans le but d'interpréter et de prédire la valeur associée. Ce type de raisonnement à partir des données s'applique en classification, comme en régression ou en apprentissage en général. Elle présente notamment les avantages de pouvoir apprendre avec très peu d'instances et de présenter un pouvoir d'explication intrinsèque.

L'objectif du projet est d'implémenter des algorithmes du transfert analogique pour réaliser différentes tâches d'apprentissage (classification, régression) et pouvoir comparer les différentes approches, identifier leurs caractéristiques, points forts et points faibles, complémentarité et incompatibilité. Développer une telle plate-forme de comparaison, proposer un protocole expérimental, pour différentes tâches d'apprentissage sur des données synthétiques contrôlées ainsi qu'une expérimentation qui portera sur des données réelles collectées à partir du site de recettes de cuisine BBC good food [1].

Notre rapport est structuré comme suit :

- **Section 2** : Les définitions du transfert analogique, ses principales propriétés, les différentes notations utiles pour la compréhension des algorithmes, quelques travaux liés ainsi que les deux algorithmes de transfert analogique que nous avons implémenté (CoAT et CCBI)
- **Section 3** : description de notre architecture globale , la réalisation d'une plate forme commune a nos différents algorithmes ainsi que nos implémentations.
- **Section 4** : description du protocole expérimental, l'ensemble des données synthétiques utilisés ainsi que nos critères d'évaluation . Dans cette section, nous fournissons également les résultats expérimentaux sur ces données .
- **Section 5** : Application a des données réelles : nous décrivons les données , la manière de les collecter ainsi les pré-traitement réalisés. Enfin l'expérimentation de nos algorithmes sur ces données et les résultats obtenus .

Nous concluons notre rapport par un résumé de nos contributions et quelques perspectives de développements futurs .

1. https ://www.bbcgoodfood.com/

Chapitre 2

Transfert analogique

Il existe diverses formalisations du transfert analogique, comme les approches par approximation les approches par contrainte, les approches par support de preuve , ou les approches par minimisation d'énergie.Cette section détaille dans un premier lieu la définition du transfert analogique, puis quelques articles de l'état de l'art basé sur ce dernier (une méthode ordinale, et une méthode par contrainte.)

2.1 Notations

CB Base de donnée appelée base cas $\{(s_i, o_i)\}$ où s_i décrit *l'instance*, appelée *situation* et o_i le *label* associé, appelé *outcome*. Tel que $s_i \in \mathbb{R}^d$ et $o_i \in \mathcal{Y}$ en classification (dans $\mathbb{R}$ en régression).

Similarité Mesure de similarité sur les situations notées σ_s ou **simX** selon les articles, et σ_r ou **simL** pour les labels

2.2 Définition

Le transfert analogique consiste à faire l'hypothèse que si deux situations [1] sont similaires dans un aspect, elles peuvent être similaires dans d'autres aspects aussi[12]. Cette méthode d'apprentissage peut être utilisé en inférence, c'est à dire, pour prédire un label pour une situation donnée en se reposant sur les données qui existent déjà.Un exemple d'analogie prédictive est l'algorithme des k-Plus Proche Voisin [15], un algorithme classique d'apprentissage automatique qui prédit la classe d'une situation s_0 selon les classes des k situations les plus **similaires** à s_0 (tel que, la similarité d'un point à ses voisins dans l'espace de description conduit à une similarité des étiquettes associées, qui conduit à prédire l'étiquette la plus fréquente parmi les voisins).Nous étudierons dans notre plate-forme l'analogie prédictive et nous la comparerons aux autres méthodes de prédictions sur différentes taches d'apprentissage.

2.3 Une approche ordinale : CoAT

Nous présentons dans cette partie l'algorithme proposé par Fadi Badra [7] A Dataset Complexity Measure for Analogical Transfer, son algorithme CoAT (*Complexity-based Analogical Transfer*) a pour but de proposer une nouvelle approche de transfert analogique.Et cela, en introduisant deux points : dans un premier lieu, une nouvelle méthode d'inférence (en prenant la complexité d'un jeu de données comme mesure).Dans un second lieu, un nouvel indicateur de la qualité d'une mesure de similarité (une complexité **minimale** veut dire en général une **meilleure** mesure de similarité, et une **meilleure** prédiction).

1. une situation, notée s, est souvent décrite par plusieurs attributs i.e., $\mathrm{s} = (\mathrm{s}^1, ..., s^d)$

Le calcul de la complexité Γ se base sur **l'ensemble du nombre d'inversions** effectué sur le jeux de données (ou base de cas CB). Ainsi, cet algorithme traduit le principe de l'analogie décrite dans la section 2.2, sous forme ordinale : l'ordre induit par la similarité entre deux situations doit être identique à l'ordre induit par la similarité entre les labels.

Cette définition conduit à la notion d'**inversion**, pour les cas qui ne respectent pas ce critère (pour un triplet de cas tels que l'ordre de σ_s diffère de l'ordre de σ_r)

Formellement, pour deux situation s_i et s_j dans CB , une situation s_0 donnée (donc le triplet (s_0, s_i, s_j)), une mesure de similarité σ_s sur les situations (les données s_i) et une mesure de similarité σ_r sur l'ensemble des labels (o_i) :

$$Inv(s_0) = \{s_i s_j | \sigma_s(s_0, s_i) \geq \sigma_s(s_0, s_j) \; et \; \sigma_r(s_0, s_i) < \sigma_r(s_0, s_j)\}$$

Le cardinal de cet ensemble est noté :

$$\gamma(s_0) = |inv(s_0)|$$

Enfin, la complexité Γ est définie comme suit :

$$\Gamma(\sigma_s, \sigma_r, CB) = \sum_{(s_0, f(s_0))} \gamma(s_0)$$

L'algorithme de cet article est comme suit :

Algorithme 1 : CoAT version non-optimisée

Données : Y,s_0, σ_s, σ_r
Résultat : y : sortie prédite pour la nouvelle situation s_0
début
  $\ell \leftarrow []$
  pour $y \in Y$ **faire**
    `// Calculer la complexité de la base avec un label attribué.`
    changer_label(s_0,y)
    $\ell.append(\Gamma(\sigma_s, \sigma_r, CB))$
  fin pour
  `// Retourner le label qui minimise la complexité.`
  retourner $Y[\ell.index(min(\ell))]$
  fin

2.4 Une approche par contrainte : CCBI

Nous présentons dans cette partie l'algorithme proposé par Eyke Hullermeier [4] Credible Case-Based Inference Using Similarity Profiles. Cette approche utilise toujours le principe des problèmes similaires ont des solutions proches comme Knn mais en se basant sur une intersection de voisinages issue de certains **seuils de similarité** : En effet contrairement à Coat et knn (qui font tout en inférence) CCBI réalise un **apprentissage** d'une certaine fonction en **phase d'entraînement** selon ces seuils-la .
Cette fois on ne considère pas des triplets $(\sigma_0, \sigma_i, \sigma_j)$ mais des **couples** (x_i, λ_{xi}) (situation,label) et comparer les valeurs de similarité entre les deux espaces **situations/labels**

En premier lieu on a une contrainte de similarité exigeant que les solutions soient toujours au moins aussi similaires que les problèmes :

$$\forall x, y \in X : simX(x, y) \leq simL(\lambda_x, \lambda_y)$$

A partir de cette contrainte, on peut raisonner comme suit :
Considérons un cas $(x1, \lambda_{x1})$ de la base de cas M et soit $x1$ qui sera α1-**similaire** a la cible problème

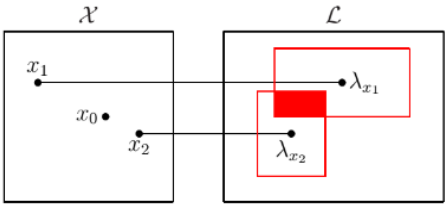


FIGURE 2.4.1 – Intersection des αi-voisinages pour prédire λ_{x_0} [4]

$x0$: soit $simX(x1, x0) = \alpha 1$. D'après l'équation précédente, la solution inconnue λ_{x0} doit alors être un élément du α1-voisinage de λ_{x1} , c'est-à-dire de l'ensemble :

$$\lambda_{x0} \in N_{\alpha 1}(\lambda.x1) = \{\lambda \in L | sim_L(\lambda, \lambda_{x1}) \geq \alpha 1\}$$

De même, si on a un autre d'autre cas (x_i, λ_{xi}) tel que
$simX(xi, x0) = \alpha i$, la solution λ_{x0} est aussi un élément du αi-voisinage de λ_{xi} et donc de l'intersection :

$$\lambda_{x0} \in \cap_{i=1}^{n} N_{sim_X(x_i,x_0)}(\lambda_{xi})$$

La figure 2.4.1 illustre ce raisonnement .

On ne peut pas comparer directement les valeurs de simX et simL, Hullermeier propose d'apprendre une transformation ζ de simX en correspondance telle que $simL = \zeta(simX)$

Pour cela on considère un relâchement de la contrainte avec une fonction attribuant à chaque degré de similarité α entre deux problèmes le plus grand degré de similarité $\beta = \zeta(\alpha)$ Les solutions de deux problèmes α-similaires sont garanties au moins β-similaires Ce qui donne des profils de similarité :

$$\zeta(\alpha) =^{df} \inf_{(x,y \in X, simX(x,y)=\alpha)} simL(\lambda_x, \lambda_y)$$

Donc l'ensemble de prédiction pour un λ_{x0} avec la contrainte relâchée devient :

$$\lambda_{x0} \in \bigcap_{i=1}^{n} \mathcal{N}_{\zeta(sim_{(X)}(x_i,x_0))}(\lambda_{xi})$$

Étant donné un espace d'hypothèses H, c'est-à-dire une classe de fonctions $\zeta : [0, 1] \to [0, 1]$ l'apprentissage revient a choisir une parmi ces hypothèses sur la base de données d'entraînement M .
L'hypothèse intéressante parmi ces fonctions ζ est la plus cohérente avec les données fournies, c'est-à-dire que la contrainte de similarité doit être satisfaite pour tous les cas x de M :

$$(x, \lambda_x), (y, \lambda_y) \in M : (sim_X(x, y) = \alpha \Longrightarrow sim_L(\lambda_x, \lambda_y) \geq h(\alpha).$$

Compte tenu des hypothèses de ces observations, l'apprentissage d'une hypothèse de similarité peut être réalisé à l'aide d'une fonction caractéristique :

$$h : x \mapsto \sum_{k=1}^{n} \beta k.\mathbb{1}_{(A_k)}(x)$$

A_k c'est le k^{eme} intervalle résultant de la discrétisation avec un pas ϵ et β_k c'est le minimum de la similarité des labels sim_L sur l'intervalle A_k

$$\beta_k =^{df} \min_{x_i,x_j : sim_X(x_i,x_j) \in A_k} sim_{L(\lambda_{xi}, \lambda_{xj})}$$

La figure 2.4.2 montre la discrétisation sur l'intervalle ainsi que les seuils d'apprentissage des β_k pour approcher la fonction ζ

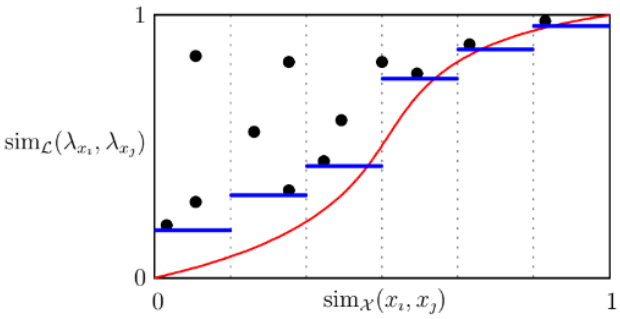


FIGURE 2.4.2 – Approximation de la fonction h en apprenant les β_k sur les sous-intervalles [4]

Une description de l'algorithme CCBI est récapitulée ci-dessous :

Algorithme 2 : algorithme d'apprentissage de CCBI

Données : CB, σ_s, σ_r, A_k, β_k ,ϵ
Résultat : β_k
début
  // Init β_k
  // discrétiser l'intervalle de similarité selon un pas ϵ
  $A_k = [A_1, A_2, ..., A_\epsilon]$ $\beta_k \leftarrow [1, ..., 1]$
  // On parcours les données par couple
  pour $(x_1, x_2) \in CB$ **faire**
    // 1.Pour chaque couple calculer la similarité 'simX' des deux situations
    $simX \leftarrow \sigma_s(x_1, x_2)$
    // 2.Retrouver l'indice de l'intervalle parmi les A_k auquel appartient la similarité simX trouvée
    $indices \leftarrow indices\ if\ simX \in A_k[indices]$
    // 3.Calculer la similarité des labels associés 'simL$(\lambda_{x1}, \lambda_{x2})'$.
    simL $\leftarrow \sigma_r(\lambda_{x1}, \lambda_{x2})$
      // 4.Affecter à β_k[indices] le minimum entre β_k et 'simL'
    $\beta_k[indices] \leftarrow \min(\beta_k, simL)$
  fin pour
fin

Inférence pour CCBI :

Algorithme 3 : algorithme d'inférence de CCBI

Données : $CB, \sigma_s, \sigma_r, \epsilon$
Résultat : y : sortie prédite pour la nouvelle situation x_0
début
  $\ell \leftarrow []$
  pour $(x_1, x_2) \in CB$ **faire**
    // 1.Pour cas 'x' dans la base de données calculer la similarité 'simX' des deux situations x_0 et x
    $simX \leftarrow \sigma_s(x,\ x_0)$
    $indice \leftarrow A_k.where(simX >= A_k[indice])$
    $seuil\_label \leftarrow \beta_k[indice]$
    pour *j1, j2 in labels :* **faire**
      $simL \leftarrow simL(j1, j2)$
      si $simL > seuil\_label$ **alors**
        $ens\_solutions.append(j2])$
      fin si
    fin pour
  fin pour
  $y \leftarrow mid(solutions)$
  $CB \leftarrow add(x_0,\ y)$
  retourner y
fin

Chapitre 3

Plate-forme de comparaison implémentée

Cette section présente l'architecture globale de notre plate-forme d'expérimentation ainsi que les outils d'implémentation utilisés et les détails d'implémentation, notamment l'impleméntation de l'amelioration réalisée sur la version naïve (non optimisée) de l'algorithme CoAT cité dans la section 2.3.

3.1 Architecture globale

Notre plate-forme a pour objectif de comparer les approches de transfert analogique ainsi que les approches de référence en apprentissage automatique, et cela, tout en proposant différentes expérimentations. Ces expérimentations auront pour objectif d'étudier la robustesse de ces approches, notamment face a la quantité de données (temps d'exécution).Pour ce faire, différentes fonctionnalités ont été implémentées :

- **Modèle à choisir :** La plate-forme offre ainsi la possibilité de choisir le modèle à tester. La plate-forme offre trois modèles classiques comme SVM [13] ou les arbres de décision [14] ainsi que trois modèles de transfert analogique comme les plus proches voisins [15], CoAT (version non-optimisée et optimisée) et CCBI.
- **Paramètres de chaque modèle :** Ensuite, chaque modèle ayant ses propres paramètres (le nombre de voisins k pour knn [15] ou CCBI, les poids w de chaque similarité pour CoAT), il est possible de préciser ces derniers.
- **Mesure de similarité :** Il est également possible de choisir une mesure de similarité pour chaque modèle (cela peut être alors une mesure comme σ_s mentionné dans 2.3.
- **Paramètres d'apprentissage :** Enfin, dans le but de tester les performances des modèles, par exemple avec une validation croisée, la plate-forme offre la possibilité de préciser le nombre d'exemples en test ainsi que le nombre de groupe k en k-cross validation.

3.2 Implémentation

Nous avons implémenté notre projet avec le langage Python (programmation orienté objet), on a construit un objet abstrait (ou classe abstraite) nommée "classifieur" dont héritent tous nos objets/modèles de transfert analogique implémentés et qui partagent les mêmes attributs et méthodes.

3.2.1 Attributs partagés :

- CB : des arrays représentant l'ensemble utilisé pour l'apprentissage (dans le cas de CCBI), et pour la prédiction.

— Parammodel : un dictionnaire qui sert a définir les différents paramètres des modèles lors de leur instanciation notamment les mesures de similarité tel que : σ_r, σ_l etc...

3.2.2 Méthodes partagées :

— fit()

1. CCBI : dans le fit, le modèle apprends les β_k cité dans la section précédente 2.3 qui est l'array contenant des valeurs/seuils qui servira à la prédiction dans la méthode predict().

— predict()

1. CoAT : prend en entrée une situation s_0 et retourne son label.
   Version optimisée : Retourne le label en calculant le cardinal de l'ensemble d'inversions définit dans la section 2.3
   Version non optimisée : Retourne le label en calculant la complexité Γ, en calculant l'ensemble des inversions pour **chaque** situation dans la base de cas.
2. CCBI : Prend une entrée X et prédit un intervalle de solutions possibles pour X dans le cas de la régression, ici on choisi le milieu.

— Mesure de similarité

1. CoAT :
   σ_s : mesure de similarité des données d'entrée.
   σ_r : mesure de similarité de sortie.
2. CCBI : simX() et simL() qui calcule respectivement la similarité entre deux cas du dataset et la similarité de leurs solutions.

— score() : prend en paramètre les prédictions et les vrais labels et revoie le score des modèles en classification et régression.

3.2.3 Méthodes/attributs propres aux deux modèles :

CoAT :

— Attributs :

1. W qui représente le vecteur des poids.

— Méthodes :

1. inv() : qui prend en paramètres un exemple s_0 de de CB, CB et W et retourne la cardinalité de l'ensemble ou sigmaS et sigmaR sont inverser pour un s_0.
2. Complex() : qui prend en paramètres le dataset (CB) et retourne la complexité de ce dernier et fait appel à inv().

CCBI :

— Attributs :

1. β_k , A_k (intervalle entre 0 et 1 avec un pas de discrétisation ϵ).

Enfin, pour les algorithmes de référence SVM, Knn ou Dtrees nous avons utilisés les implémentions disponibles dans "ScikitLearn" [1].

3.2.4 Optimisation de CoAT

Le calcul de la complexité Γ est effectué en un temps remarquable dans l'algorithme proposé dans l'article de la section 2.3 (une étude du temps d'exécution sera effectué dans la section 4).Tel que, cette équation consiste a recalculer pour chaque situation dans la base de cas, la somme du cardinal des inversions en rajoutant la situation s_0, ce qui peut être très coûteux.

Nous proposons alors une version qui, au lieu de recalculer la somme des inversions pour chaque donnée comme dans l'équation de Γ, calcule tout simplement le cardinal de l'inversion obtenu en

1. https ://scikit-learn.org/stable/

rajoutant la situation s_0 au jeu de données. Nous validons que cette implémentation donne bien les mêmes résultats dans la section 4.1.
Le pseudo code de cette version est donnée comme suit :

Algorithme 4 : CoAT version optimisée

Données : Y,$\mathrm{s}_0, \sigma_s, \sigma_r$
Résultat : y : sortie prédite pour la nouvelle situation s_0
début
  $\ell \leftarrow []$
  pour $y \in Y$ **faire**
    `// Au lieu de recalculer les inversions de toute la base avec Γ, il suffit de calculer l'inversion pour le label attribué.`
    $changer\_label(s_0,y)$
    $\ell.append(Inv(\sigma_s, \sigma_r, CB))$
  fin pour
  `// Retourner le label qui minimise le nombre d'inversions.`
  retourner $Y[\ell.index(min(\ell))]$
  fin

Chapitre 4

Expérimentations

Cette section présente dans un premier lieu la validation des résultats obtenus sur des jeux de données à titre illustratif, ensuite le protocole expérimental mis en oeuvre trois modèles de transfert analogique comme les k-Plus Proche Voisins et deux modèles d'apprentissage automatique classiques comme SVM, enfin, la section présente l'étude des résultats obtenus.

4.1 Validation des résultats

Nous validons dans cette partie les résultats obtenus grâce aux jeux de données à titre illustratif pour les modeles CoAT (il est à noter que les mesures de similarité utilisées ici (et par la suite) sont celles décrites dans l'article [7]) et CCBI (de même, les mesures utilisées sont celle de [4]).

CoAT [7] Nous reprenons la méthode CoAT [7] et le jeu de données illustratif de l'article(représenté dans le tableau 4.1).Le tableau 4.1 représente 4 situations (appartements), et leur attributs (nombre de chambres, surface et prix).Nous voulons prédire selon les mesures de similarité cité dans [7] le prix d'un appartement t.

La figure 4.1.1 représente les valeurs de la complexité Γ pour chaque prix attribué.Nous pouvons voir que le prix correspondant à une complexité Γ minimale est de 800, ce résultat est compatible avec celui retrouvée dans l'article [7].

CCBI Nous avons reproduit l'expérimentation réalisée dans l'article qui est un problème de régression simple, la fonction à apprendre est le polynôme $x \rightarrow x^2$. 25 exemples d'apprentissage (x_i,λ_{x_i}) sont donnés, où les x_i sont uniformément distribués dans X = [0, 1] et les λ_{x_i} sont distribués suivant une loi normale avec une moyenne x_i^2 et un écart type de $1/10$.
La figure 4.1.2 montre alors que pour des données triées dans [0,1] les intervalles des solutions possibles (compris entre les deux lignes brisées) prédit par CCBI suivent bien la trajectoire du polynôme x^2 (ligne bleue), de plus nos prédictions sont similaire aux prédictions présentées dans l'article [4] , ce qui valide donc notre expérimentation.

4.2 Comparaison du temps d'exécution

Cette section décrit les expérimentations effectués pour étudier le temps d'exécution de la méthode ordinale CoAT , la méthode par contrainte CCBI, et les algorithmes classique d'apprentissage automatique. Les tests ont été effectué sur des machines CPU Intel 1 core, 2 threads.

Optimisation de CoAT : Cette section a pour but d'étudier l'impact des différentes optimisations sur le temps d'exécution du modèle CoAT [7].La figure 4.2.1 illustre la comparaisons des versions naïve et optimisée détaillées dans la section 2.3 en variant le nombre d'exemples du jeux

Situation	nb_rooms	area	price
s_1	1	midtown	440
s_2	2	midtown	600
s_3	1	downtown	700
s_4	3	downtown	900
t	2	downtown	?

TABLE 4.1 – Jeu de données de l'article coAT [7]

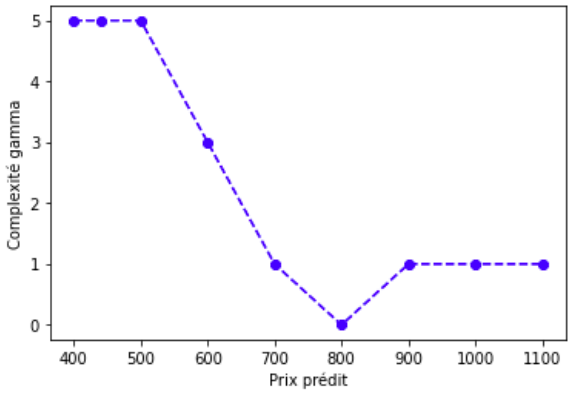


FIGURE 4.1.1 – Complexité pour chaque prix potentiel

de données n . Nous pouvons remarquer que les optimisations de calculs ont réussi à améliorer (largement) la complexité temporelle de CoAT.

Dans la figure 4.2.2 nous pouvons voir d'une façon plus claire l'évolution du temps d'exécution des deux versions.Nous pouvons remarquer qu'après avoir *fit* un polynôme de degré 2 sur le nombre d'exemple sur la version optimisée (ligne rouge) que nous retrouvons un temps d'exécution polynomial en $O(n^2)$.Néanmoins, la version naïve est en un temps cubique $O(n^3)$.

CCBI et CoAT : Ensuite nous comparons le temps d'exécutions des deux modèles de transfert analogique, l'approche ordinale CoAT (Version optimisé) et l'approche par contraintes CCBI (en comptant le temps d'apprentissage) sur le même jeu de données du problème de régression que celui présenté dans la section 4.2 , d'après la Figure 4.2.3 on remarque que le temps d'exécution de la version optimisé de CoAT est plus conséquent en le comparant à celui de CCBI, de plus le temps augmente considérablement en fonction du nombre d'exemples.

CCBI et modèles d'apprentissage automatique : Enfin nous comparons le temps d'exécution de CCBI et des modèles de Machine Learning (SVM, knn, Dtrees) sur les données de la fonction polynomiale $x \to x^2$, figure 4.2.4, le temps d'exécution de CCBI est plutôt rapides mais knn et SVM sont plus rapide, on remarque aussi que jusqu'à 100 exemple d'apprentissage environs le temps de CCBI est stable puis commence à augmenter

4.3 Performances

Cette section décrit dans un premier lieu le protocole expérimental effectué pour les modèles de transfert analogique ainsi que les résultats obtenus pour ce dernier.

4.3.1 Protocole expérimental

Dans cette section, nous allons annoter les étapes de notre protocole expérimentale. Nous avons réaliser cela a travers une série d'expérimentations au moyen d'une validation croisée. Nous allons ensuite les comparer entre eux ainsi qu'a d'autres modèles classiques.

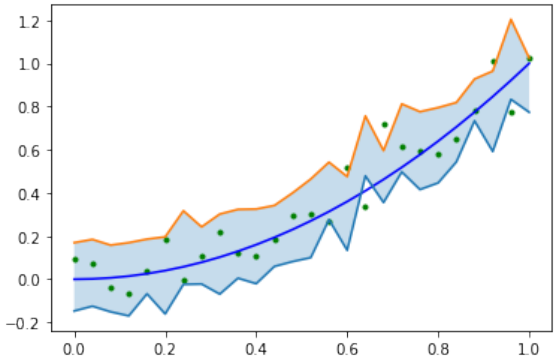


FIGURE 4.1.2 – Approximation de $x \to x^2$(ligne pleine bleue), sous forme de bande de confiance, en utilisant CCBI (région entre les lignes brisées), les points verts indique les exemples.

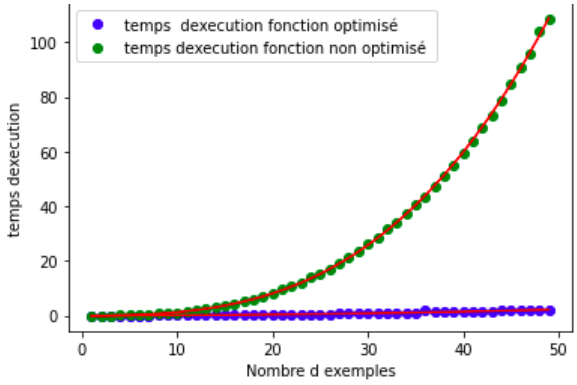


FIGURE 4.2.1 – Temps d'exécution (s) selon le nombre d'exemples n

(a) **CoAT** Nous effectuons un test de performances en comparant le score (Accuracy) sur les données Iris [1]

(b) **CCBI**

Pour CCBI nous mesurons la performance sur plusieurs ensembles de données du référentiel UCI et de Archives Statlib, d'abord par comparaison des performances (modèle de machine learning vs CCBI) puis les performances de CCBI sur deux datasets cité dans l'article sur les données UCI (Autompg [2] et cpu [3]) avec les mesure de similarité suivante :

(a) CONF : mesurée en termes de fréquence des prédictions correctes (intervalle prédit couvre la vraie valeur).

(b) PREC : mesurée en termes de distance moyenne d'un intervalle prédit (plus l'intervalle est petit, plus la précision est est bonne).

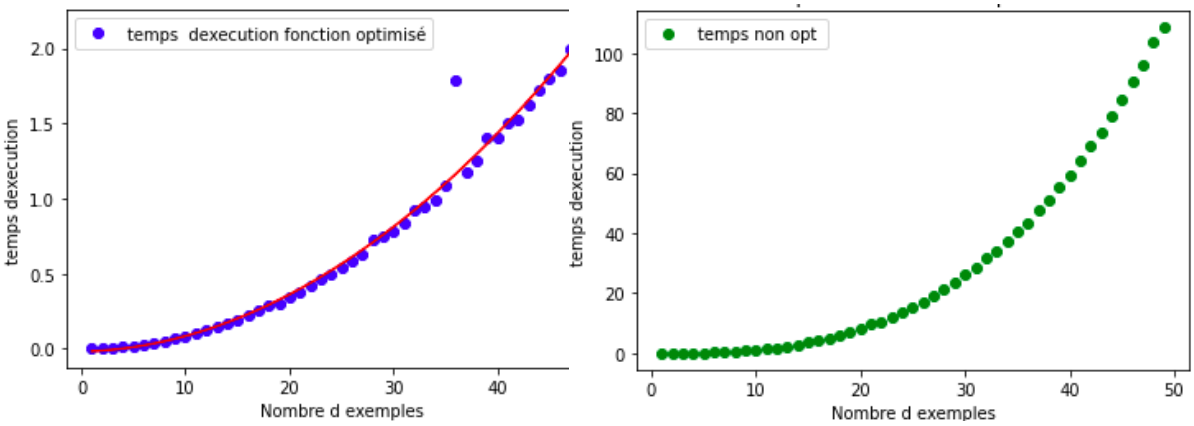
(c) MAE : mesurée en termes de distance moyenne entre la vraie valeur et l'estimation (centre de l'intervalle)

4.3.2 Résultats

Dans cette section, nous allons présenter les résultats obtenus par le modèle CoAT et CCBI avec d'autres modèles classique sur les données.

Le tableau 4.2 illustre la performances de CoAT. Nous pouvons voir que CoAT a soit de meilleures performances ou des performances égales sur les données iris (avec le modèle de transfert analogique knn, ou les autres modèles comme SVM).

1. https ://archive.ics.uci.edu/ml/datasets/iris
2. https ://archive.ics.uci.edu/ml/datasets/auto+mpg
3. https ://archive.ics.uci.edu/ml/datasets/Computer+Hardware



(a) Version optimisé (b) Version non optimisé

FIGURE 4.2.2 – Comparaison du temps d'exécution (s) selon le nombre d'exemples n

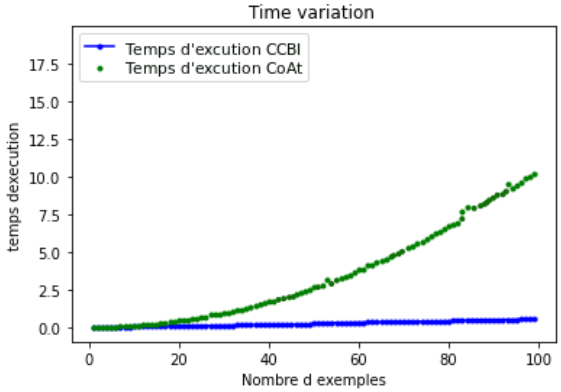


FIGURE 4.2.3 – Comparaison du temps d'exécution(s) de CoAT et CCBI selon le nombre d'exemples n

Ensuite le tableau 4.3 illustre la performances de CCBI. On remarque que les modeles classique ont un meilleur score que le score de CCBI sur les données UCI [4]), néanmoins CCBI a obtenu un très bon score.

Le dernier tableau le tableau 5.1 illustre la performances évaluées avec nos mesures précédemment définie dans la section 4.3.1. Les résultats obtenus sont convainquant avec une confidence d'une moyenne de 0.96 pour les Dataset traité ci-dessous, avec une précision très basse (ce qui est une bonne chose, car nous le rappelons que la précision ici c'est la longueur moyennes des intervalles prédit).

Comme nous pouvons le voir ci-dessus, Un inconvénient des approches basées sur le transfert analogique est que le coût (temps d'exécution) d'une tache de classification/régression de nouvelles instances peut être élevé. Cela est dû au fait que presque tous les calculs ont lieu au moment de la prédiction. En outre, nous avons également pu voir que le temps d'exécution de CCBI est meilleur que celui de CoAT.

4. https ://archive.ics.uci.edu/ml/datasets.php

Modèle	**SVM**	**Knn**	**Dtrees**	**CoAT**
Score (Accuracy)	0.98	0.98	0.90	0.98

TABLE 4.2 – Performances des modèles sur les données iris

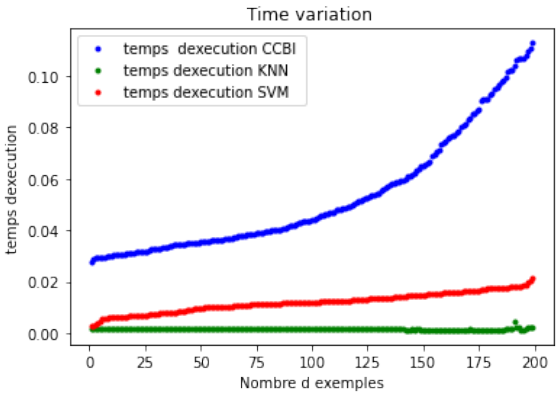


FIGURE 4.2.4 – Le temps d'exécution par rapport au nombre d'exemples

Modèle	**SVM**	**Knn**	**Dtrees**	**CCBI**
Score	0.98	0.99	0.98	0.94

TABLE 4.3 – Performances des modèles sur le dataset cpu de la base de données UCI

Mesure	**CONF**	**PREC**	**MAE**
mpg	0.94	0.06	0.01
cpu	0.98	0.35	0.15

TABLE 4.4 – Performances des modèles sur le dataset cpu de la base de données UCI

Chapitre 5

Application à des données réelles

Une expérimentation de nos algorithmes porte sur des données réelles collectées a partir du site de recettes de cuisine BBC good food [1]. On part d'un jeu de données de recettes de cuisine pour lesquelles on connaît la liste des ingrédients, leur quantité et les valeurs nutritionnelles de la recette (kcal, sucres, fibres, sel,protéines,...)

Cette section présente notre processus de collecte de données,les différents pré-traitements réalisés pour les avoir dans un format utilisable par nos algorithmes ainsi les expérimentations que nous avons réalisé .

5.1 Collecte de données

Dans un premier temps on a exploré le site de BBC good food pour voir les informations disponibles sur chacune des recettes publiées.

On a collecté un dataset format json qui contient des recettes de cuisine (résultats du scrapping des recettes sur ce site de BBC good food) Le dataset contient environ 3000 recettes avec 30 attributs pour chacune , ou on distingue les 3 types :

1. **Les attributs catégoriels** : origine d'une recette , son niveau de difficulté pour sa préparation,...
2. **Les attributs numériques** : La note attribuée par des gens,l'apport calorique d'une recette ,...
3. **Les attributs textuels** : les ingrédients composants une recette

La figure 5.1 montre un extrait de ce dataset json pour une seule recette .

5.2 Pré-traitement

En voyant une description d'une recette (dans la figure 5.1) qui est globalement du texte brut contenant du bruit (texte inutile pour nos tâches) , pour avoir les informations nécessaires qui nous intéressent on a fait un parser pour extraire les attributs et les champs utiles pour nos tâches .

5.2.1 Parser

Parmi les traitements effectués par notre parser :

1. Extraire toutes les valeurs du champ **nutrition-informations** qui permettent d'avoir la quantité des différentes valeurs nutritionnelles : **sugar, Carbohydrate, Kcal, Protein, Salt, Saturated fat, Fat...** La figure 5.2.1 montre la distribution de certains attributs nutritionnels dans l'ensemble des recettes prises.
2. Convertir ces valeurs nutritionnelles en une même unité pour toutes les recettes (le tout en grammes)

1. https ://www.bbcgoodfood.com/

{"page": {"article": {"author": "Jane Hornby", "description": "Cover just the top of your Christmas cake with icing, then use cinnamon sticks, ribbon and tea lights for an easy yet attractive decorative finish", "id": "5428746", "tags": []}, "recipe": {"collections": ["Christmas cake"], "cooking_time": 11700, "prep_time": 5400, "serves": 20, "keywords": ["1 of 5 a day", "Christmas", "Xmas", "Fruit cake", "Traditional cake", "Classic cake", "basic Christmas cake", "easy Christmas cake", "boozy Christmas cake", "how to make Christmas cake", "Festive cake", "Christmas baking", "Stir up sunday"], "ratings": 100, "nutrition_info": ["Added sugar 69g", "Carbohydrate 79g", "Kcal 518 calories", "Protein 5g", "Salt 0.2g", "Saturated fat 8g", "Fat 17g"], "ingredients": ["unsalted butter", "light muscovado sugar", "egg", "plain flour", "mixed spice", "orange", "pecan", "apple juice", "unsalted butter", "maple syrup", "dark rum", "mixed dried fruit", "dried cranberries", "dark rum", "maple syrup", "apricot jam", "icing sugar", "natural marzipan", "water", "sugarpaste", "unscented tea lights", "cinnamon sticks", "ribbon", "bay leaf", "rosemary"], "courses": ["Afternoon tea", "Dessert", "Treat"], "cusine": "British", "diet_types": [], "skill_level": "Easy", "post_dates": "1446336000"}, "channel": "Recipe", "title": "Midwinter candle cake"}}

FIGURE 5.1.1 – Extrait d'une recette dans le dataset json

Nb recettes	**Attributs**	**tâches classification**	**tâches régression**
1000	49	5	2
exemple	ingrédeints,nutritions,..	saine,végétarienne,	apport calorique,rating

TABLE 5.1 – Caractéristiques du jeu de données contruit

3. Binarisation des valeurs catégorielles **(one hot encoding)** pour les attributs : degré de préparation d'une recette , l'origine de la recette , ...)
4. Avoir les attributs qui nous permet d'étiqueter une recette (la recette saine(healthy) ou non , vegétarienne , sans-gluten , sans-oeufs , law-calories ,...) figure 5.2.2
   Grace a ce type d'attributs on fera des classes binaires pour nos recettes.
5. Avoir les valeurs qui nous permet de savoir si une recette a bien été aimée par des gens (ratings)

On a utilisé comme outils du parsing : les **regex pattern en python** avec comme cibles les attributs qui nous intéressent, ainsi que les champs valeurs qui les composent .

5.2.2 Traitement des ingrédients d'une recette

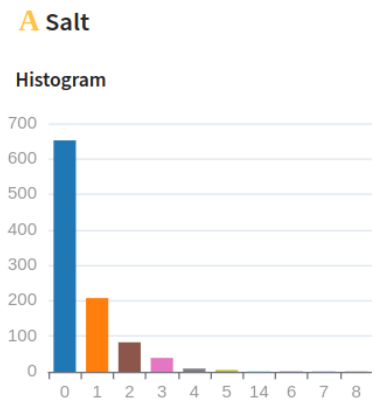
Le champ ingrédient d'une recette est sous format texte brut. Pour cela on a fait une représentation en **sac de mots** pour pouvoir enlever les ingrédients très rares (qui n'apparaissent que dans une seule recette par exemple) et avoir un certain **vocabulaire cohérent** sur lequel on mettra l'ensemble de nos recettes. Pour réaliser cela on a procédé par l'approche **TF-IDF**.
La figure 5.2.3 montre un nuage de mots qui visualise l'ensemble du vocabulaire pris selon la fréquences des ingrédients dans les recettes .

5.2.3 Caractéristique du jeu de données construit

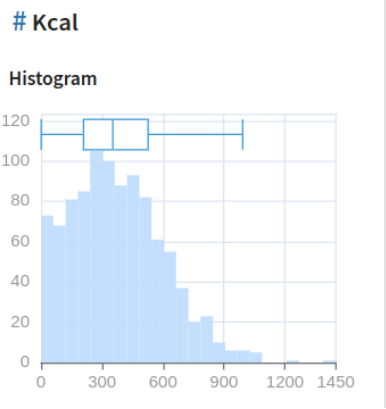
Aprés les différents traitement effectués sur les données bruts, un beau jeu de données est construit pour les modèles d'apprentissage :

5.3 Expérimentations

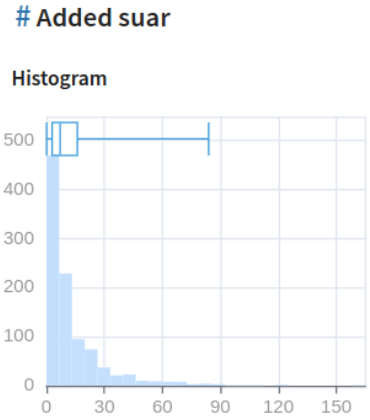
Dans cette partie, nous présentons les résultats des modèles sur les données BBC Good Food sur des tâches de classification et de régression.



(a) Distribution du sel



(b) Distribution des calories



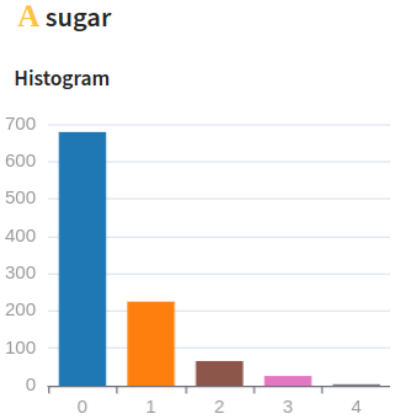


FIGURE 5.2.1 – Distribution des valeurs nutritionnelles

Modèle	SVM	Knn	Dtrees	CoAT
Score (Accuracy)	0.72	0.82	0.78	0.80

TABLE 5.2 – Performances des modèles en classification sur les données BBC Good Food

5.3.1 Classification : la recette est-elle saine ou non ?

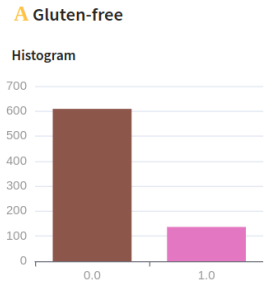
Cette section présente une tâche de classification ou nous essayons de prédire si une recette est-elle saine, ou non.

Le tableau 5.2 représente les performances des modèles de transfert analogique CoAT,knn et des autres modèles.Nous pouvons voir que CoAT a pu surpasser les modèles SVM et Dtrees, et cela pourrait être expliqué par le nombre de données. Nous pouvons même préciser que, quand on a peu de données, le modèle CoAT a de meilleurs performances que les autres modèles d'apprentissage.Néanmoins, son temps d'exécution peut être pénalisant si le nombre d'exemples commence a grandir.

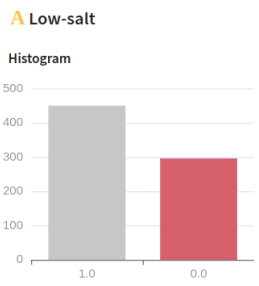
5.3.2 Régression : quel est l'apport calorique de la recette ?

Dans cette partie, nous comparons les modèles CCBI et CoAT dans la tâche de régression où nous essayons de prédire le nombre de calories d'une recette.

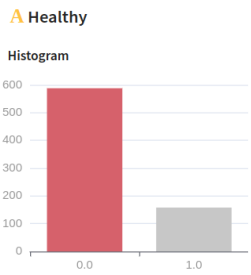
Le tableau 5.2 représente les performances des modèles CoAT et CCBI , ainsi que la régression linéaire sur cette tache .



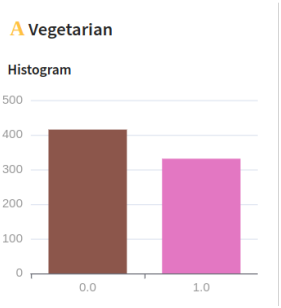
(a) Gluten-free recipies



(b) Low-salt recipies



(c) Healthy recipies



(d) Vegetarien recipies

FIGURE 5.2.2 – Variables catégorielles pour les tâches de classification considérées

Pour évaluer la tache de régression on a utilisé le **R2-score** issu à partir de l'erreur MAE (R2-score $= 1 - MAE$) CoAT est plus performant que CCBI en terme de R2-score .

FIGURE 5.2.3 – WordCloud des ingrédients dans notre corpus de recettes

Modèle	**LR**	**CCBI**	**CoAT**
R2-Score	0.82	0.80	0.91

TABLE 5.3 – Performances des modèles en régression sur les données BBC Good Food

Chapitre 6

Conclusion

Le concept d'analogie a été longuement décrit et étudié depuis les philosophes grecs ; ses applications récentes s'intéressent en particulier aux sciences cognitives, en effet, ces dernières années le transfert analogique a intéressé largement la littérature, particulièrement ses application dans l'apprentissage artificiel.

Ainsi, la réalisation de ce projet nous a offert l'opportunité d'étudier en détail quelques aspects de cette méthode et de comprendre son réel fonctionnement.

Pour ce faire, nous avons aborder trois grandes approches du transfert analogiques, la première méthode ordinale en s'inspirant de l'algorithme CoAT [7] ou deux hypothèses principales sont appliquées : une **hypothèse de rationalité** ou les valeurs sont totalement ordonnées, ainsi qu'une **contrainte de continuité** (chaque fois qu'une situation s_i ressemble plus à une situation s_0 qu'une situation s_j, cet ordre doit être conservé par les labels). De plus, étant donné que cet algorithme a un temps d'exécution conséquent, nous avons proposé dans ce travail une version optimisée en terme de temps d'exécution en remplaçant le calcul effectué pour toute la base de données par le calcul effectué pour une seule instance s_0, l'instance pour laquelle on cherche a prédire un label.La deuxième méthode étudiée est l'algorithme par contraintes CCBI [4], qui se base sur des profils de similarités entre les deux espaces (situations,labels), et enfin la méthode classique d'apprentissage automatiques les k-Plus Proches Voisins.

A partir de ces travaux et d'implémentations réalisés d'algorithme classiques d'apprentissage automatique, nous avons implémenté une plate-forme de comparaison expérimentale qui a pour objectifs de comparer les méthodes de transferts analogique ainsi que les méthodes d'apprentissage automatique dans les tâches de classifications et régression, tel que nous avons comparé ces derniers sur divers jeux de données classique, et avons notamment étudier la robustesse de ces modèles face aux quantité de données (temps d'exécution).

Enfin, nous avons collecté des données réels du site de recette BBC Good Food et avons fait des experimentations sur ces derniers, ce qui nous a permis de constater que, bien que l'idée du transfert analogique est séduisante, son temps d'exécution sur de grands jeux de données peut être pénalisant. Ceci dit, dans le cas ou on a très peu de données pour l'apprentissage comme pour les données médicales (sur des maladies , ou des patients,...) les performances des méthodes de transfert analogique sont plus intéressantes.

Comme perspectives futures, nous envisageons dans un premier temps d'évaluer les méthodes de transfert analogique sur d'autres tâches comme la tâche d'adaptation. D'autre part, il serait intéressant de faire évoluer notre approche optimisée de l'algorithme CoAT en réduisant encore plus le temps d'exécution, qui bien que a été réduit a un $O(n^2)$ pourrait être encore plus optimisé.

Bibliographie

[1] Henri Prade. 2016. Reasoning with data-a new challenge for AI ? In *International Conference on Scalable Uncertainty Management* (Int Conf , Springer, 274–288.

[2] Myriam Bounhas, Henri Prade, and Gilles Richard. 2017. Analogy-based classifiers for nominal or numerical data. *International Journal of Approximate Reasoning* 91, (2017),36–55.DOI : https ://doi.org/10.1016/j.ijar.2017.08.010

[3] Vahid Jalali, David Leake, and Najmeh Forouzandehmehr. 2017. Learning and Applying Case Adaptation Rules for Classification : An Ensemble Approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, Melbourne, Australia, 4874–4878. DOI : https ://doi.org/10.24963/ijcai.2017/685

[4] Eyke Hullermeier. 2007. Credible Case-Based Inference Using Similarity Profiles. *IEEE Trans. Knowl. Data Eng.* 19, 6 (2007), 847–858. DOI : https ://doi.org/10.1109/TKDE.2007.190620

[5] David W. Aha. 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36, 2 (1992), 267–287. DOI : https ://doi.org/10.1016/0020-7373(92)90018-G

[6] Eyke Hüllermeier. 2003. Possibilistic instance-based learning. *Artificial Intelligence* 148, 1–2 (2003), 335–383. DOI : https ://doi.org/10.1016/S0004-3702(03)00019-5

[7] Fadi Badra. 2020. A Dataset Complexity Measure for Analogical Transfer. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, Yokohama, Japan, 1601–1607. DOI : https ://doi.org/10.24963/ijcai.2020/222

[8] Kathleen Hanney and Mark T. Keane. 1996. Learning adaptation rules from a case-base. In *Advances in Case-Based Reasoning* (Lecture Notes in Computer Science), Springer, Berlin, Heidelberg, 179–192. DOI : https ://doi.org/10.1007/BFb0020610

[9] Vahid Jalali and David Leake. 2016. Enhancing case-based regression with automatically-generated ensembles of adaptations. *J Intell Inf Syst* 46, 2 (2016), 237–258. DOI : https ://doi.org/10.1007/s10844-015-0377-0

[10] L. Miclet, S. Bayoudh, and A. Delhay. 2008. Analogical Dissimilarity : Definition, Algorithms and Two Experiments in Machine Learning. *jair* 32, (August 2008), 793–824. DOI : https ://doi.org/10.1613/jair.2519

[11] Claudio A. Policastro, André C. P. L. F. Carvalho, and Alexandre C. B. Delbem. 2008. A hybrid case adaptation approach for case-based reasoning. *Appl Intell* 28, 2 (April 2008), 101–119. DOI : https ://doi.org/10.1007/s10489-007-0044-4

[12] Fadi Badra, Karima Sedki, and Adrien Ugon. 2018. On the Role of Similarity in Analogical Transfer. DOI : https ://www.webofscience.com/wos/woscc/full-record/WOS :000717233500033

[13] Nello Cristianini and Elisa Ricci. 2008. Support Vector Machines. In Encyclopedia of Algorithms, Ming-Yang Kao (ed.). Springer US, Boston, MA, 928–932. DOI : https ://doi.org/10.1007/978-0-387-30162-$4_4$15

[14] J. R. Quinlan. 1986. Induction of decision trees. Mach Learn 1, 1 (March 1986), 81–106. DOI : https ://doi.org/10.1007/BF00116251

[15] Leif E. Peterson. 2009. K-nearest neighbor. Scholarpedia 4, 2 (February 2009), 1883.DOI : https ://doi.org/10.4249/scholarpedia.1883