Apprentissage automatique et réseaux de neurones

Rapport du mini projet :

« Classification des articles de sports par analyse de l'objectivité »

Spécialité : Master 1 SII

Binôme :

CHIKH Khadidja

REFIK Amina

11/07/2019

Contexte :

Dans le cadre du module “apprentissage automatique et réseaux de neurones”, il nous a été demandé de concevoir une solution qui résoudrera un problème de classification/regression, en adoptant la technique d'apprentissage automatique par réseaux de neurones et utiliser par la suite le modèle entrainé dans une application au choix.

Problème :

Nous nous sommes documentées sur les différents *datasets* proposés, et avons choisi le «*Sports articles for objectivity analysis Data Set* » qui contient de nombreux articles de sports, écrits en anglais et extraits depuis internet.

Notre séléction était basée sur les critères suivants :

-Préférence de thème.

-La difficulté du problème.

-Les capacité de nos machines.

Description des données choisies :

Les données représentent des caractéristiques syntaxiques (extraites en utilisant le logiciel « the Stanford POS Tagger») et sémantiques (extraites en utilisant le logiciel « SENTIWORDNET ») depuis les textes des articles. Le dataset contient 1000 instances (fichiers des textes) avec 59 attributs représentant ces caractéristiques et décrits comme suit :

Attributs syntaxiques (numériques) comme:

CC	Frequency of coordinating conjunctions
CD	Frequency of numerals and cardinals
DT	Frequency of determiners
EX	Frequency of existential there
FW	Frequency of foreign words
INs	Frequency of subordinating preposition or conjunction

Attributs sémantiques (numériques)

semanticobjscore	Frequency of words with an objective SENTIWORDNET score
semanticsubjscore	Frequency of words with a subjective SENTIWORDNET score
sentence1st	First sentence class (comparing the count of sentiment oriented words versus objective or neutral words of the first sentence)
sentencelast	Last sentence class (comparing the count of sentiment oriented words versus objective or neutral words of the last sentence)
txtcomplexity	Text complexity score

L'attribut “classe” (alphabétique)

Label	objective vs. subjective

Avec la distribution : Objective=636 (63,6%) Subjective=364 (36,4%)

Autre attribut :

totalWordsCount	total number of words in the article

Objectif :

Classer ces articles en 2 catégories : « subjectifs » et « objectifs ».

Traitement des données:

Codification de la classe :

L'attribut «label » est notre but. Il prend deux valeurs possibles en format alphabétique, ce qui n'est pas approprié à une utilisation avec réseau de neurones. Pour trouver la bonne codification, nous avons analysé d'abord les attributs *firstsentence* et *lastsentence* ; ceux-ci prennent les valeurs 0 et 1 (objective ou subjective). Notre attribut classe doit donc prendre les mêmes valeurs. Ensuite, nous avons analysé les premières et dernières phrases de quelques textes pour déduire enfin la codification suivantes :

0 pour «objective » , 1 pour « subjective ».

Normalisation des attributs :

En analysant les données des différents attributs (tous à part t*xtcomplexity, firstsentence, lastsentence*), nous avons remarqué que ceux-ci sont relatifs à l'attribut *totalwordscount* i.e. doivent être représentés d'une manière proportionnelle à ce dernier (sinon ça influencera l'étape d'apprentissage). Nous avons donc transformé les valeurs de ces attributs en pourcentages.

Exemple : *totalwordscount* =309 avec *semanticobjscore*=21 devient *totalwordscount* =309 avec *semanticobjscore*=21/309

Remarque : Nous avons réalisé cette partie (traitement) en écrivant un programme en langage C à fin d'obtenir un fichier contenant les données codées et normalisées.

Conception du réseau de neurones :

Recherche des hyperparamètres et architecture adéquats:

- Fonction d'initialisation(à varier):

Nous nous sommes documentées sur quelques fonctions à l'aide de la documentation de Matlab et avons choisi : Randnc,Randnr,Rands,Randsmall.

- Fonction d'apprentissage (à varier):

Nous nous sommes basées sur des documents **[1] [2]** décrivant des experimentations sur des problèmes qui se rapprochent du nôtre. Dans ces documents on y trouve les descriptions de ces problèmes et des tableaux récapitulatifs, comparant entre les différentes fonctions d'apprentissage selon les performances, le temps, l'exactitude..etc obtenus.

Les fonctions choisies sont : TrainGdm, TrainLm, TrainOss, TrainRp, TrainScg.

-Fonctions d'activation(fixe) :

Des couches cachées et de celle de sortie, nous avons opté pour la fonction Tansig. Cette dernière est un bon choix car elle est suffisamment complexe ainsi que rapide. (**source : Cours AARN**)

-Fonction de calcul de performance (fixe):

Nous avons opté pour la fonction « *crossentropy* », qui est plus appropriée pour notre encodage.

- Taux d'apprentissage:

Parmi les fonctions d'apprentissage que nous avons choisies, il y en a celles qui modifient le taux d'apprentissage progressivement par rapport aux performances à chaque étape(epoch), ce qui permet l'amélioration de l'apprentissage au fur et à mesure.

-Mode de descente du gradient :

Nous avons opté pour l'unique mode permit par Matlab pour les réseaux de neurones de type « feedforward » qui est le mode « *Batch* ». Il est aussi le mode le plus utilisé généralement.

-Architecture du réseau :

Nous avons appliqué la methode de « recherche extensive » : commençant par 1, 2 ensuite 3 couches cachées (au maximum, ceci en se basant sur le cours du module), en variant le nombre de neurones de 10 à 100 par pas de 10.

-Conditions d'arrêt :

temps = 300s, but(performance)=0

-Autres paramètres :

nombre maximum d'*epochs*=300,nombre d'itérations(*retrain*)=30

-Découpage du dataset :

70% : apprentissage, 15% : validation, 15% : test.

-Comparaison entre deux architectures :

Par la meilleure performance et exactitude de validation.

Expérimentations :

Nos choix de pramètres mènent à avoir (10+10*10+10*10*10*)*5*4*30=600066000 résultats possibles, ce qui est énorme en terme de sauvegarde, de temps ainsi qu'en terme de lisibilité.

Pour y remédier, nous avons commencé par les combinaisons d'une seule couche à 10 neurones (1*5*4*30) et observé les résultats :

Architecture	Fonction_apprentissage	Fonction_initialisation_des_poids	Exactitude_de_validation
10	trainrp	randsmall	91.3333
10	trainscg	randsmall	90.6667
10	trainlm	randnr	88.6667
10	trainoss	randsmall	88.6667
10	trainoss	rands	88.6667
10	trainrp	randnc	87.3333
10	trainoss	randnc	87.3333
10	traingdm	rands	87.3333
10	trainlm	rands	86.6667
10	trainscg	randnc	85.3333
10	trainoss	randnr	85.3333
10	traingdm	randnr	84.6667
10	traingdm	randnc	82.6667
10	trainscg	randnr	81.3333
10	trainrp	rands	90
10	trainscg	rands	88
10	trainrp	randnr	88
10	traingdm	randsmall	88
10	trainlm	randsmall	86
10	trainlm	randnc	82

Nous avons remarqué que la combinaison des deux fonctions 'trainrp' et 'randsmall' a donné la meilleure exactitude de validation (91.33%), d'où notre choix a porté sur cette combinaison de fonctions pour l'étape d'apprentissage.

Meilleur résultat avec architecture à une seule couche cachée :

Architecture	Performance_apprentissag	Performance_de_validatio	Performance_de_test	Temps_execution	Exactitude_apprentissage	Exactitude_de_validation	Exactitude_de_test	Exactitude_totale	Condition_Arret_active
20	0,36895	0,26058	0,39665	6,513	84,71	90	86,67	85,8	Validation stop,

Avec 20 neurones : performance de validation=0,26 et exactitude de validation=90%. La suite des résultats est présentée dans l'annexe.

Meilleur résultat avec architecture à deux couches cachées :

Architecture	formance_apprentiss	Performance_de_validation	Performance_de_test	Temps_execution	Exactitude_apprentissage	Exactitude_de_validation	Exactitude_de_test	Exactitude_totale	Condition_Arret_active
20 30	0.38871	0.24575	0.35148	23.295	84.2857	91.3333	85.3333	85.5	Validation stop.

Avec <20,30> neurones : performance de validation=0,25 et exactitude de validation=91,33%. La suite des résultats est présentée dans l'annexe.

Résultat obtenus avec architecture à trois couches cachées :

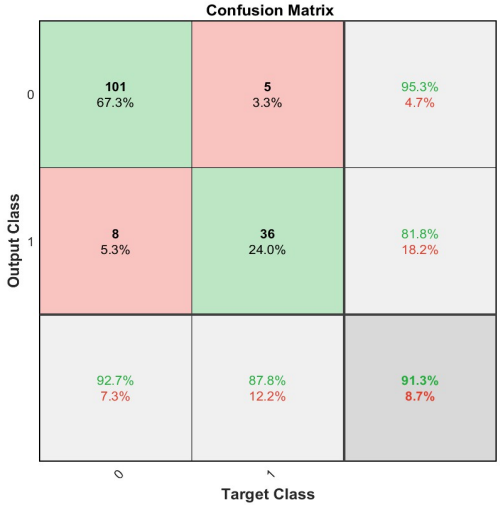
Par manque de temps, nous n'avons pas pu terminer tous les cas possibles. Les résultats obtenus peuvent être consultés sur cd joint.

Remarque :

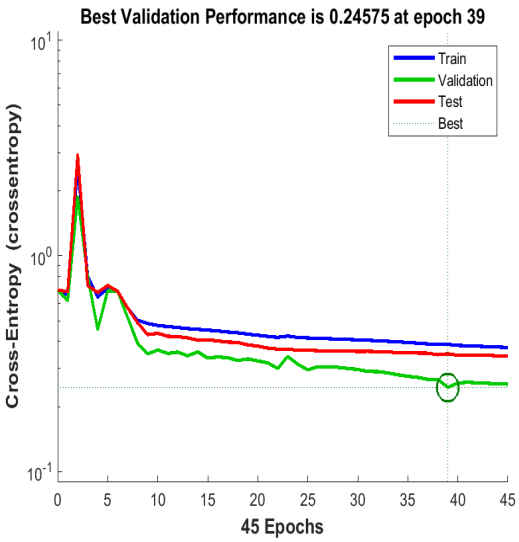
Au cours de l'apprentissage, nous avons remarqué que celui-ci s'arrête toujours à cause de *validation_check.*

Ces résultat nous mènent à adopter l'architecture : **2 couches cachées avec 20 et 30 neurones respectivement.**

Graphes de l'architecture choisie :



Matrice de confusion pour données de validation



Graphe de performance

Possibilités d'exploitation du modèle :

Afin de pouvoir reconstruire notre modèle pour l'exploiter, nous avons rajouter la possibilité de garder les matrices des poids de la meilleure architecture.

Par manque de temps, nous n'avons pu développer une application pour exploiter notre modèle. Mais nous avons réfléchi à des manières possibles de cela :

1) Intégration dans une application classificatrice:

Une application bureau, qui réalisera un traitement de fichiers d'articles de sports existant sur l'ordinateur et selectionnés (entrée) , en extractant les caratéristiques sémantiques et syntaxiques des textes ensuite à l'aide du modèle, les classer en articles subjectifs et objectifs.

Une telle application est très utile pour les personnes qui ont déjà de nombreux articles à étudier et parmi lesquels ils vont selectionner ceux qu'ils considèreront plus approprié à leur besoin de recherche. Exemple : Les étudiants en journalisme.

2) Intégration dans une application client/serveur :

La partie serveur sera chargée de classifier les aricle sportifs des differents sites web (comme ceux des *Columnists*) en se basant sur le modèle qui y est intégré, et en communiquant avec les bases de données des sites afin d'attribuer une note à chacun, calculée à partir du pourcentage des articles objectifs publiés sur ceux-ci. Cette note est mise à jour suite à chaque modification des contenus.

La partie serveur communiquera avec la partie client (via une extension du navigateur) et retournera les résultats à travers le *front-end* qui pourra contrôler l'affichage des articles objectifs ou subjectifs uniquement (fournis par un moteur de recherche ou par un site web).

Conclusion :

Dans ce mini projet, nous avons pu réaliser un nombre important d'experimentations sur différentes architectures de reseaux de neurones, ce qui nous a permis de voir de diverses résultats et décider par la suite l'architecture la plus adéquate à notre apprentissage, qui a atteint comme exactitude maximale 91,33% et une performance de 0,25.

Références :

[1] « *Comparison of Neural Network Training Functions for Hematoma Classification in Brain CT Images » -IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. II (Jan. 2014), PP 31-35*

lien : https//pdfs.semanticscholar.org/14ac/e90f795e7b463d0168883d671e044f5c8375.pdf

[2] *« Summary of the training functions in Matlab's NN toolbox Vladimir Vacic »*

lien : http://alumni.cs.ucr.edu/~vladimir/cs171/nn_summary.pdf

Et la documentation de Matlab

Répartition des tâches :

Tâche	Membres
Documentation sur datasets	K. CHIKH, A. REFIK
Choix du dataset	K. CHIKH, A. REFIK
Décision de codification et normalisation	K. CHIKH, A. REFIK
Ecriture du programme de traitement des données	A.REFIK
Documentation sur les paramètres et choix de ces derniers	K. CHIKH, A. REFIK
Ecriture du script	K. CHIKH, A. REFIK
Apprentissage avec architecture à une couche cachée	A.REFIK
Apprentissage avec architecture à deux couches cachées	K. CHIKH
Apprentissage avec architecture à trois couches cachées	A. REFIK
Idée d'implémentation n°1	K.CHIKH
Idée d'implémentation n°2	K.CHIKH, A.REFIK
Rédaction de rapport	K.CHIKH, A.REFIK
Graphes de distribution des données d'apprentissage	A.REFIK

ANNEXE :

Suite de la description des données syntaxiques :

JJ	Frequency of ordinal adjectives or numerals
JJR	Frequency of comparative adjectives
JJS	Frequency of superlative adjectives
LS	Frequency of list item markers
MD	Frequency of modal auxiliaries
NN	Frequency of singular common nouns
NNP	Frequency of singular proper nouns
NNPS	Frequency of plural proper nouns
NNS	Frequency of plural common nouns
PDT	Frequency of pre-determiners
POS	Frequency of genitive markers
PRP	Frequency of personal pronouns
PRP$	Frequency of possessive pronouns
RB	Frequency of adverbs
RBR	Frequency of comparative adverbs
RBS	Frequency of superlative adverbs
RP	Frequency of particles
SYM	Frequency of symbols
TOs	Frequency of "to" as preposition or infinitive marker
UH	Frequency of interjections
VB	Frequency of base form verbs
VBD	Frequency of past tense verbs
VBG	Frequency of present participle or gerund verbs
VBN	Frequency of past participle verbs
VBP	Frequency of present tense verbs with plural 3rd person subjects
VBZ	Frequency of present tense verbs with singular 3rd person subjects
WDT	Frequency of WH-determiners
WP	Frequency of WH-pronouns
WP$	Frequency of possessive WH-pronouns
WRB	Frequency of WH-adverbs
baseform	Frequency of infinitive verbs (base form verbs preceded by “to”)
Quotes	Frequency of quotation pairs in the entire article
questionmarks	Frequency of questions marks in the entire article
exclamationmarks	Frequency of exclamation marks in the entire article
fullstops	Frequency of full stops
commas	Frequency of commas
semicolon	Frequency of semicolons
colon	Frequency of colons

ellipsis	Frequency of ellipsis
pronouns1st	Frequency of first person pronouns (personal and possessive)
pronouns2nd	Frequency of second person pronouns (personal and possessive)
pronouns3rd	Frequency of third person pronouns (personal and possessive)
compsupadjadv	Frequency of comparative and superlative adjectives and adverbs
past	Frequency of past tense verbs with 1st and 2nd person pronouns
imperative	Frequency of imperative verbs
present3rd	Frequency of present tense verbs with 3rd person pronouns
present1st2nd	Frequency of present tense verbs with 1st and 2nd person pronouns

Suite des résultats avec une couche cachée :

Architecture	Performance_apprentissage	Performance_de_validation	Performance_de_test	Temps_execution	Exactitude_apprentissage	Exactitude_de_validation	Exactitude_de_test	Exactitude_totale	Condition_Arret_active
20	0,36895	0,26058	0,39665	6,513	84,71	90	86,67	85,8	Validation stop,
30	0,37976	0,28148	0,36324	12,137	84,14	86,67	84,67	84,6	Validation stop,
40	0,35306	0,2896	0,46327	5,336	85,43	89,33	82,67	85,6	Validation stop,
100	0,3044	0,29259	0,49271	13,784	86,86	89,33	78	85,9	Validation stop,
60	0,41071	0,30316	0,50098	2,338	83,57	88,67	76,67	83,3	Validation stop,
50	0,3435	0,30543	0,52483	7,415	85,57	88,67	80	85,2	Validation stop,
10	0,35164	0,30575	0,47447	7,489	85,71	91,33	77,33	85,3	Validation stop,
80	0,37358	0,31009	0,4527	5,058	83,57	88	79,33	83,6	Validation stop,
90	0,36715	0,31744	0,3815	7,471	85,14	88	84	85,4	Validation stop,
70	0,35636	0,33068	0,50533	6,218	85,43	87,33	77,33	84,5	Validation stop,

Suite des résultats avec deux couches cachées :

Architecture	erformance_apprentissag	Performance_de_validation	Performance_de_test	Temps_execution	Exactitude_apprentissage	Exactitude_de_validation	Exactitude_de_test	Exactitude_totale	Condition_Arret_active
20 30	0.38871	0.24575	0.35148	23.295	84.2857	91.3333	85.3333	85.5	Validation stop.
20 10	0.37256	0.25264	0.45867	01/08/15	86	92	80	86	Validation stop.
20 20	0.3297	0.25391	0.4015	21.215	86.8571	90.6667	80.6667	86.5	Validation stop.
30 70	0.3642	0.26616	0.41041	26.221	85.1429	91.3333	83.3333	85.8	Validation stop.
100 70	0.28845	0.27114	0.52089	95.903	88.1429	89.3333	80	87.1	Validation stop.
20 100	0.38312	0.27827	0.40859	25.083	84.7143	88.6667	82	84.9	Validation stop.
10 50	0.40259	0.28032	0.5133	12.749	82.8571	92.6667	80	83.9	Validation stop.
20 50	0.38461	0.28317	0.4656	34.061	84.8571	86.6667	78	84.1	Validation stop.
30 30	0.34556	0.28339	0.42822	159.049	85.2857	88	80.6667	85	Validation stop.
100 40	0.38133	0.28617	0.39092	44.341	84.1429	88	81.3333	84.3	Validation stop.
70 80	0.29619	0.28633	0.53388	63.827	87.8571	86.6667	78	86.2	Validation stop.
30 60	0.29958	0.28809	0.47189	148.217	87.7143	86.6667	82	86.7	Validation stop.
20 70	0.34441	0.28938	0.41716	39.597	85.5714	88.6667	84	85.8	Validation stop.
50 40	0.31768	0.28947	0.42167	50.551	88	88	83.3333	87.3	Validation stop.