Fouille de données et Medias Sociaux

M2 DAC

TME 2. Inférence Collective

Ce TME a pour objectif de manipuler quelques modèles d'inférence collective vus en cours. Dans un premier temps, nous proposons de nous intéresser à l'apprentissage d'un classifieur local linéaire, avec prise en compte des labels connus pour le voisinage des noeuds du réseau considéré. Ce classifieur est ensuite utilisé pour la mise en place de l'algorithme ICA, pour réaliser l'inférence des labels sur l'ensemble des noeuds du graphe, connaissant les interdépendances entre les labels des noeuds connectés du graphe. Enfin, nous nous intéresserons au modèle de propagation de labels, permettant d'apprendre un classifieur dont les poids respectent une certaine régularité entre les scores attribués aux labels des noeuds connectés.

1 Classifieur Local

Il s'agit dans un premier temps de construire un classifieur local permettant d'attribuer un label à chaque noeud du graphe avec prise en compte conjointe des attributs du noeud considéré et des labels observés dans son voisinage.
Tout le problème est alors de décider comment agréger les labels des différents voisins. Différents types d'agrégation pourront être envisagés : Mode, Count, Proportion, etc...
Pour l'apprentissage, on se propose, comme pour les sections suivantes, de se placer dans un contexte transductif semi-supervisé où seuls certains labels sont inconnus. Nous utilisons encore une fois la collections WebKB, qui convient tout à fait à ce type de problème, car il contient différents sites web interconnectés, pour lesquels un certain nombre d'attributs locaux sont connus. Les liens entre sites des différentes universités sont donnés dans le sous-repertoire "cites", alors que le contenu et les labels sont dponnés dans le sous-repertoire "content" (que vous avez déjà utilisé). Le repertoire "cites" est composé, comme "content", de quatre sous-repertoires: un par université. Les fichiers contenus dans ces quatre repertoire contiennent une liste de liens, un lien par ligne de chaque fichier, au format: <site_source> <site_destination>.
Alors qu'on pourra faire varier par la suite la proportion de labels connus pour en observer l'impact sur les performances de l'inférence collective, on se propose dans un premier temps de considérer 10% de labels inconnus dans le graphes considérés.

2 ICA

À partir du classifieur local défini dans la section précédente, il s'agit maintenant de mettre en placel'algorithme ICA vu en cours, permettant l'inférence collective des labels sur l'ensemble du réseau.
Cet algorithme itératif se décompose en deux phases:

1. Une phase de *bootstraping*, dans laquelle on infère un label pour chaque noeud de label non connu le label (le label attribué est le label de plus fort score selon le classifieur local) à partir de ses attributs et des labels connus des noeuds de son voisinage

2. Une phase d'inférence collective, qui, tant qu'on n'est pas arrivé à un étiquetage stable :

   - Tire un ordre aléatoire $\mathcal{O}$ sur les noeuds à considérer (les noeuds dont le label n'est pas connu);
   - Puis qui itère sur $\mathcal{O}$ en déterminant le label de plus fort score pour chaque noeud, à partir de ses attributs et les labels (connus ou inférés) des noeuds de son voisinage

3 Propagation de labels

Alors que l'algorithme ICA se base sur un classifieur local, appris a priori, qui considére les labels du voisinage de chaque noeud pour prendre en compte les effets d'auto-correlation dans les réseaux (inter-dépendance des labels des noeuds connectés), d'autres modèles, tels que les modèles de propagation de labels, incluent l'apprentissage du classifieur dans la phase d'inférence.
Les modèles de propagation de labels cherchent à intégrer les régularités sur le graphe dans un classifieur classique. Plutôt que de définir une représentation des noeuds incluant les labels des voisins, on conserve une représentation classique des entrées (attributs x de chaque noeud en entrée du classifieur) et on cherche à définir des poids θ permettant d'observer des corrélation entre les scores données aux labels des noeuds inter-connectés. Ainsi, alors qu'un classifieur classique ne considèrerait que l'adéquation entre le score prédit pour un label $f_\theta(x_i)$ à partir des attributs d'un noeud i et le label connu pour ce noeud, on considère alors un second terme permettant la prise en compte des effets d'auto-correlation dans le réseau. Pour chaque label $l \in \mathcal{L}$:

$$\theta_l^* = argmin_\theta \sum_{x_i \in X_o} \Delta(f_\theta(x_i), y_{l,i}) + \beta \sum_{i,j \in E} E(i,j)(f_\theta(x_i) - f_\theta(x_j))^2 + \lambda ||\theta||^2$$

où Δ est une fonction de coût (par exemple un Hinge-Loss) entre le score prédit pour un noeud i et la valeur de $y_{l,i}$ pour ce noeud (= 1 si l est le label du noeud i, -1 sinon), X_o correspond à l'ensemble des noeuds pour lesquels le label est connu, $E(i,j)$ correspond au poids de la relation entre les noeuds i et j (1 ou 0 dans le cas de graphes

non-pondérés comme celui de WebKB) et β et λ sont des hyper-paramètres permettant de régler respectivement l'importance de l'hypothèse de régularité sur le graphe et celle de la régularisation L2 des paramètres θ. Le terme $\sum_{i,j \in E} E(i,j)(f_\theta(x_i) - f_\theta(x_j))^2$ permet alors d'intégrer une régularité entre les scores donnés aux noeuds connectés du graphe.