rapport de Stage

Romain Gambardella

2023
Juillet

1 Réalisations

J'ai commencé par créer une base de donnée sur le jeu du Futoshiki et faire fonctionner le Futoshiki avec l'EPLL (avant le début du stage).

En fait, après le début du stage je me suis rendu compte qu'il y avait plusieurs comportements bizarres :

1. un gradient était toujours nul lorsqu'il y avait une inégalité
2. des termes négatifs apparaissaient dans les matrices de coûts des inégalitées, ce qui posait de gros problèmes à toulbar pour la résolution

Le point 1) a été expliquée par l'expression du gradient : pour $i > j$ $\frac{\partial l_{PL}(n,\theta)}{\partial \theta_{ij}[v_i,v_j]} = 2 \times \mathbb{1}(y_i = v_i, y_j = v_j) - \mathbb{P}(y_i = v_i)\mathbb{1}(y_j = v_j) - \mathbb{P}(y_j = v_j)\mathbb{1}(y_i = v_i)$ Dans le cas où les cases i et j ne prennent jamais les valeurs v_i et v_j, le gradient est toujours nul.

J'ai essayé plusieurs méthodes pour compenser ce problème :

1. dans le cas du Sudoku, ajouter des termes unaires ne fonctionne pas car les termes unaires ne fonctionnent pas bien, à cause de la superposition des contraintes
2. rajouter du "bruit" casse complètement la PLL.
3. implementer une PLL stochastique à l'ordre 2 (et à l'odre n) vectorisée (fonctionne)

J'ai généralisé le code de Marianne pour que je puisse l'appliquer aux prochains problèmes plus facilement (par exemple le grounding) -> le code marche maintenant sur une version plus complexe du visual sudoku, sans les indices. J'ai vectorisé le code de gestion des plus proches voisins.

2 Visual grounding

In the case of visual grounding, one possible value is added to the graphical model that the programm is supposed to approximate, in addition to 1,2,..,9. This value mean "there was an indice here", and all the solutions given to the PLL are twisted so that this value is in place of the hint previously here.

This effectively prevents grounding, as the NN can no longer help himself with these hints in order to learn the digits.

In this case, just adding leNet output as unary costs no longer helps, as there would be no usefull gradients on these values. Instead, lenet input is given as a feature to the NN, in addition to the other informations.

It can be observed that lenet learns a permutation of the digits.

3 PLL stochastique

Algorithm 1 PLL stochastique

Require: θ' le modèle actuel

Require: *neighbours* une fonction stochastique qui à x associe un ensemble d'ensemble de voisins (dans le cas de la PLL à l'ordre 1, neighbours est déterministe et correspond l'ensemble des V_i tels que $V_i = \{y$ tel que x et y coincident partout sauf à l'indice $i\}$

function PLL(x)
    $pll \leftarrow 1$
    $N \leftarrow neighbours(x)$
    for all $N_i \in N$ **do**
        $proba \leftarrow \mathbb{P}_{\theta'}(X = x \mid X \in N_i)$
    return *pll*
while 1 **do**
    $x \leftarrow$ une réalisation de θ
    $pll \leftarrow PLL(x)$
    $\theta' \leftarrow update(\theta', -log(PLL))$

Algorithm 2 Fonction neighbours pour la PLL d'ordre 2 stochastique

Require: N_{masks} le nombre de couple d'indices à générer ($Card(N) = N_{masks}$)

Require: $N_{2-uplets}$ le nombre de 2-uplets à étudier sur ces indices ($N_{2-uplets} = Card(N_i)$)

function NEIGHBOURS(x)

$masks \leftarrow \{N_{masks}$ 2-tuples d'indices aléatoires$\}$

$N \leftarrow \{\}$

for all $n_{mask} \in 0...N_{masks}$ **do**

$G \leftarrow \{y \in \Omega$ tels que $y \neq x, y_i = x_i \forall i \notin masks[n_{mask}]\}$

$N_{n_{mask}} \leftarrow \{choisir N_{2-uplets}\ y\ dans\ G\} \cup \{x\}$

ajouter $N_{n_{mask}}$ à N

return N

3.1 Cohérence de la PLL d'odre 2 stochastique

Let θ' be the computed model and θ the observed model. Let Ω_x denote all the possible observations.

Let $PLL(\theta')_m)$ be the value of the PLL averaged over m random samples (observations) $X_1, ..., X_m$ i.i.d that follows the distribution of θ, that is :

$PLL(\theta')_m = \sum_{n=0}^{m} PLL(X_n)$

We denote the random neighbourhood used to compute the stochastic $PLL(X_i)$ by $N(X_i)$.

Theorem 1 *Let N be the random fonction used to compute the stochastic PLL.*
Suppose that $\mathbb{P}(N(X) = n \mid X = x) = cte(n),\ \forall\ n \in \mathcal{P}(\Omega_x)$ and $\forall x \in n$ Then :

1. *$PLL(\theta')_m$ converges when m tends to ∞ to a number, that we call $PLL(\theta')$.*
2. *$PLL(\theta')$ is minimum at $\theta' = \theta$.*

We first give another expression for $PLL(\theta')_m$:

$PLL(\theta')_m = \frac{1}{m}\sum_{n=1}^{m} PLL(X_n)$
$= \frac{1}{m}\sum_{n=1}^{\infty}\sum_{n\in N(X_i)} \mathbb{P}_{\theta'}(Y = X_i \mid Y \in n)$
$= \frac{1}{m}\sum_{n=1}^{\infty}\sum_{x\in\Omega_x}\sum_{n\in\mathcal{P}(\Omega_x)} \mathbb{1}(X_n = x, n = N(X_n))\mathbb{P}_{\theta'}(Y = x \mid Y \in n)$

Reordering the sums, we get:
$PLL(\theta')_m$
$= \frac{1}{m}\sum_{n\in\mathcal{P}(\Omega_x)}\sum_{x\in\Omega_x}\sum_{n=1}^{\infty} \mathbb{1}(X_n = x, n = N(X_n))\mathbb{P}_{\theta'}(Y = x \mid Y \in n)$
$\to \sum_{n\in\mathcal{P}(\Omega_x)}\sum_{x\in\Omega_x} \mathbb{P}_d(X = x, n = N(X))\mathbb{P}_{\theta'}(Y = x \mid Y \in n)$
by the law of large numbers, where $\mathbb{P}_d$ is the joint probability distribution of X, an input sample, and N(X).

We have also : $\mathbb{P}_d(X = x, N(X) = n) = \mathbb{P}_d(X = x)\mathbb{P}_d(N(X) = n \mid X = x)$
$= \mathbb{P}_d(X \in n)\mathbb{P}_d(X = x \mid X \in n)\mathbb{P}_d(N(X) = n \mid X = x)$
$= \mathbb{P}_\theta(X = x \mid X \in n) \times K(n)$

where $K(n)$ does not depend on X by assumption.

We hence get :
$PLL(\theta') =$
$\sum_{n\in\mathcal{P}(\Omega_x)} K(n)\sum_{x\in\Omega_x} \mathbb{P}_\theta(X = x \mid X \in n) log(\mathbb{P}_{\theta'}(Y = x \mid Y \in n))$
Which is minimum for $\theta' = \theta$

We immediatly deduce from this that the stochastic PLL of order 2 previously mentionned is indeed minimum at $\theta' = \theta$