École Normale Supérieure

Rapport de stage de L3

Techniques d'inférence variationnelle et apprentissage automatique

École Polytechnique
Fédérale de Lausanne
CVLab

Lucas Pluvinage
encadré par Pierre Baqué
sous la supervision de Pascal Fua

12 juin 2017 — 18 août 2017

Résumé

Le CVLab de l'EPFL est un laboratoire de vision par ordinateur. Ce laboratoire travaille entre autre sur des problèmes de tracking et de reconnaissance en trois dimensions. Ce sont des problèmes d'apprentissage structuré dont la représentation par des modèles graphiques probabilistes est particulièrement adaptée. J'ai travaillé avec Pierre Baqué sous la supervision de Pascal Fua, responsable du laboratoire. Pierre Baqué est un doctorant qui s'est particulièrement intéressé aux problèmes d'apprentissage et de prédiction structurés et à l'inférence variationnelle. Ses articles ont permis d'obtenir une méthode efficace d'inférence variationnelle puisque parallélisée, tirant ainsi parti des capacités calculatoires des GPU modernes. L'objectif de ce stage était de comprendre les techniques d'inférence mean-field, leurs spécificités ainsi que de les implémenter dans un cadre d'apprentissage automatique supervisé de problèmes pouvant être ambigus.

Table des matières

Introduction

L'**apprentissage structuré** est un ensemble de méthodes permettant d'apprendre et de prédire des objets ayant une structure particulière. En représentant le problème sous forme de modèle graphique probabiliste, l'apprentissage automatique peut capter des interactions d'ordre supérieur entre les variables et découvrir la structure d'un problème. Par exemple, dans le cadre de la segmentation sémantique, l'utilisation de modèles graphiques est très populaire car elle capte mieux les problématiques de régularité d'une image et la structure que le découpage doit avoir. En apprenant au mieux une distribution de probabilité modélisant la réalité d'un phénomène, on peut ensuite réaliser de l'inférence conditionnelle sur cette distribution pour résoudre le problème appris.

La question de l'apprentissage dans un contexte ambigu (des problèmes ayant plusieurs solutions) n'a pas été très développée. La méthode d'inférence multi-modale offre une perspective nouvelle pour développer une méthode d'apprentissage robuste même dans les cas difficiles.

À l'occasion de ce stage, j'ai donc :
- étudié les méthodes d'inférence mean-field et multi-modal mean-field.
- implémenté ces méthodes en Python/TensorFlow de façon assez efficace.
- créé un framework d'apprentissage automatique et des protocoles de test.
- utilisé le framework dans le cadre de l'apprentissage du jeu du Sudoku pour tester la robustesse des méthodes d'apprentissage sur des jeux de données ambigus.
- étendu le modèle pour permettre l'apprentissage d'une gamme plus large de CRF tout en gardant une complexité calculatoire raisonnable.

1 Contexte

La première étape dans ce stage fut de me mettre à jour sur les articles de Pierre Baqué en étudiant les méthodes d'inférence variationnelles et en particulier celle de mean-field. La lecture de [Bis06] me fut recommandée mais c'est surtout via des discussions successives que j'ai compris les enjeux et les problèmes sous-jacents aux modèles graphiques probabilistes. Le contexte est le suivant : on travaille sur un ensemble de n variables aléatoires $X = (X_i)_{i\in[|1,n|]}$ dont les interactions sont représentées par un modèle graphique probabiliste. Ces variables peuvent être discrètes ou continues. Dans un premier temps nous travaillerons dans le cas discret. On veut à partir de ce modèle pouvoir obtenir des probabilités conditionnelles et des distributions a posteriori.

1.1 Modèles graphiques probabilistes et représentation exponentielle

1.1.1 Définition

Modèle graphique Notons D l'ensemble des valeurs prises par ces variables aléatoires et $d = |D|$. On peut représenter les dépendances entre ces variables par un graphe.
S'il est orienté, c'est un **réseau bayésien** (et le graphe doit être acyclique pour que le modèle ait un sens). Les arêtes représentent les dépendances conditionnelles entre les variables et caractérisent une factorisation de la loi jointe sur les variables.
Si il est non orienté, c'est un **champ aléatoire conditionnel** (CRF : Conditional Random Field). Il peut s'agir d'un graphe ou d'un hypergraphe. Chaque arête ou hyperarête représente une interdépendance probabiliste entre plusieurs variables. Notons $G = (X, E)$, $E = (E_i)_{i\in[|1,k|]}$ étant l'ensemble des arêtes de ce graphe.

Représentation exponentielle Dans le cadre de ce stage, nous avons travaillé sur la représentation exponentielle des CRF, dont la loi jointe fait intervenir une notion d'énergie. Soit $x \in D^n$ une configuration des variables aléatoires. Chaque configuration x est associée à une **énergie** déterminée par la fonction $\phi(X) : D^n \to \mathbb{R}$. Alors la distribution de probabilité représentée par ce modèle est une fonction de ϕ et s'écrit ainsi :

$$P(X = x) = \frac{exp(-\phi(x))}{\sum\limits_{y\in D^n} exp(-\phi(y))} = exp(-\phi(x) - log(Z))$$

où

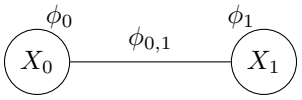
— Z est la fonction de partition de la loi de probabilité.
— ϕ se décompose en une somme de fonctions potentiels sur les arêtes du graphe que l'on notera $(\psi_i)_{i\in[|1,k|]}$.
— $\psi_i : D^{|E_i|} \to R$. À chaque configuration des variables reliées par l'arête i, ψ_i associe une énergie locale d'interaction. Alors $\phi(X) = \sum_{i=1}^{k} \psi_i(E_i)$.

Un CRF dont la distribution est en représentation exponentielle s'approche d'un système en physique statistique qui a tendance à *minimiser son énergie totale d'interaction*. Pour la suite on se restreindra aux interactions entre les paires de variables, on peut décomposer ϕ :

$$\phi(X) = \sum_{i=1}^{n} \phi_i(X_i) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} \phi_{i,j}(X_i, X_j)$$

1.1.2 Exemple

Considérons le plus simple des CRF non trivial, X_i prenant valeur dans $\{0, 1\}$. Avec
— $\phi_0(x) = -\mathbb{1}_{x=0}$
— $\phi_1 = 0$
— $\phi_{0,1}(x_0, x_1) = -3\mathbb{1}_{x_0=x_1}$



On peut calculer l'énergie de chaque configuration et en déduire la distribution de probabilité modélisée.

$X_0 \backslash X_1$	0	1
0	-4	-1
1	0	-3

Par exemple la configuration d'énergie minimale a pour probabilité

$$P(X_0 = 0, X_1 = 0) = \frac{exp(4)}{exp(4) + exp(1) + 1 + exp(3)} \simeq 0.70$$

1.1.3 Inférence

L'**inférence** sur des modèles probabilistes est de calculer, étant donné un modèle probabiliste paramétré par θ et un ensemble I de variables connues, la distribution $P_\theta(X|I)$ des variables du modèle. L'inférence exacte, bien que possible, n'est pas computationnellement viable puisqu'elle implique le calcul de la fonction de partition Z qui requiert une somme sur D^n élements, autrement dit un calcul de complexité exponentielle. Il est donc nécessaire de réaliser des approximations, et c'est là que les méthodes variationnelles entrent en jeu.

1.2 Inférence variationelle mean-field

1.2.1 Une méthode d'approximation

Soit P la distribution de probabilité associée à un modèle graphique. Le calcul de cette distribution est exponentiel, à cause de la fonction de partition Z, l'approximation est donc nécessaire. Considérons la distribution de probabilité Q factorisée sous la forme

$$Q(X) = \prod_{i=1}^{n} Q_i(X_i)$$

On notera par la suite $Q_i(X_i = l) = q_{i,l}$. On approxime alors P par Q de façon à minimiser la **divergence de Kullback-Leibler** $KL(Q||P) = \sum_x Q(X = x) log(\frac{Q(X=x)}{P(X=x)})$. Cette valeur représente en théorie de l'information la quantité d'information perdue quand on utilise Q pour approximer P (c'est le nombre de bits moyens supplémentaires nécessaires pour coder une valeur tirée sous P avec un code optimisé pour Q au lieu d'un code optimisé pour P).

Comme expliqué dans [Bis06] - chapitre 10.1., il suffit pour minimiser cette divergence d'optimiser les paramètres $q_{i,j}$ un à un :

$$q^*_{i,l} = exp(\mathbb{E}_{Q(X|X_i=l;q)}(logP(X)) - logZ_i)$$

Cette descente coordonnée par coordonnée est garantie de converger vers un minimum local. Elle est cependant inutilisable d'un point de vue computationnel puisque le calcul est séquentiel.

1.2.2 Implémentation

Préambule sur TensorFlow TensorFlow est une bibliothèque de calcul tensoriel optimisé, open source et supporté par Google. Utilisable en Python, TensorFlow permet de définir ensemble de calculs représentés par un graphe, qu'il compile pour en créer du code optimisé et, quand il le peut, s'exécutant sur GPU. L'avantage du calcul tensoriel, i.e. sur des tableaux multidimensionnels, et qu'il est souvent parallélisable. Les cartes graphiques sont alors l'outil de prédilection pour effectuer ce genre de calculs : par exemple une GTX Titan comportant 2688 cœurs CUDA est beaucoup plus efficace pour calculer parallèlement que les 32 cœurs du serveur sur lequel la carte est montée. Ainsi durant ce stage un des enjeux était de développer des algorithmes travaillant sur des tenseurs pour exploiter au mieux la puissance des GPU via TensorFlow.

J'ai donc utilisé la méthode parallèle de Pierre Baqué [BBFF15] pour mettre à jour tout les $q_{i,l}$ en même temps. Ainsi l'implémentation tire avantage de la capacité des GPU à exécuter des opérations parallèles tout en conservant la garantie de convergence. L'équation de mise à jour est la même, sauf que l'on ajoute un facteur d de "damping" qui permet d'assurer la convergence, contrairement à la mise à jour parallélisée sans damping dont la convergence n'est pas garantie.

$$q^{t+1}_{i,l} = exp(d\,\mathbb{E}_{Q(X|X_i=l;q)}(logP(X)) + (1-d)logq^t_{i,l} - logZ_i)$$

Implémentation par convolution L'implémentation parallèle permet de mettre à jour l'ensemble des $q_{i,l}$ en même temps mais cela rend le calcul efficace seulement si on dispose d'un moyen de calculer parallèlement $E_{Q(X/X_i=l;q)}(logP(X))$ pour tout les couples (i,l). Le moyen le plus simple pour obtenir un calcul efficace tout en gardant un pouvoir expressif suffisant est de supposer que le CRF est **sous forme de grille** et **invariant par translation**. C'est sous ces hypothèses qu'il conviendra de travailler par la suite.
Soit $m \times l = n$ le format de la grille. Alors le calcul devient (en indexant les variables par leur position (i,j) :

$$\mathbb{E}_{Q(X|X_{(i,j)}=l;q)}(logP(X)) = \sum_{\substack{x\in D^n \\ x_{(i,j)}=l}} \prod_{b,c} q_{(b,c),x_{(b,c)}}\phi(x)$$

Maintenant il suffit de séparer ϕ entre les termes qui dépendent de l et les autres qui dépendent seulement de (i,j) qui rentrent dans la fonction de partition $Z_{(i,j)}$.

$$\phi(x) = \phi_{(i,j)}(l) + \sum_{(c,d)!=(i,j)} \phi_{(i,j),(c,d)}(l, x_{(c,d)}) + C_{(i,j)}$$

En remplaçant dans l'équation l'expression ci-dessus, en séparant la somme et en factorisant pour utiliser la propriété $\forall (i,j), \sum_{k\in D} q_{(i,j),k} = 1$, l'expression s'écrit ainsi :

$$\mathbb{E}_{Q(X|X_{(i,j)}=l;q)}(logP(X)) = \phi_{(i,j)}(l) + \sum_{\substack{(c,d)!=(i,j) \\ k\in D}} q_{(c,d),k}\phi_{(i,j),(c,d)}(l,k) + C_{(i,j)}$$

Enfin, en utilisant l'hypothèse d'invariance par translation de ϕ, le calcul ci-dessus correspond à la somme d'un terme local et d'une convolution 2D entre q et ϕ. Cela donne donc, en terme de calcul tensoriel,

$$q^* = softmax(theta^*) = softmax(U + conv2D(q, W))$$

où

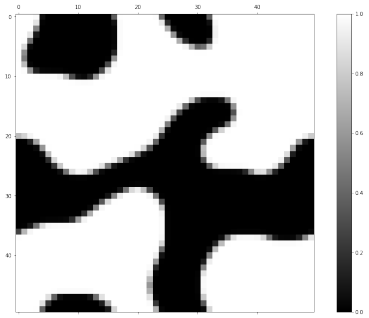
- U représente les potentiels unaires, de dimension $m \times l \times p$.
- W représente les potentiels de couples relatifs à $(0,0)$, de dimension $m \times l \times p \times p$.

Ainsi l'équation de mise à jour parallèle pour l'approximation mean-field se réduit, sous les hypothèses énoncées précédemment, en trois opérations parallélisables sur GPU. L'inférence sur des modèles graphiques en grilles invariants par translation se fait alors de manière optimisée. On note $MF(\phi, M)$ l'algorithme qui a un CRF représenté par ϕ et un ensemble M de variables fixées (dans le cas où on veut faire de l'inférence conditionnelle) associe l'inférence mean-field, i.e. les valeurs de q obtenue après un nombre fixé d'itérations de l'opération ci-dessus.

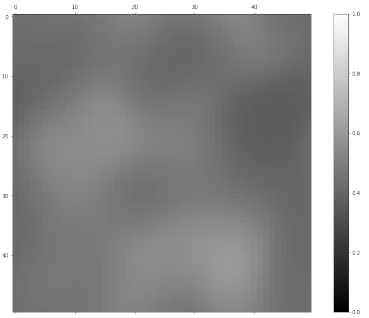
1.2.3 Modèle d'Ising

Le **modèle d'Ising** est un exemple simple de CRF et c'est celui que j'ai implémenté dans un premier temps pour comprendre les enjeux de l'inférence mean-field. Dans ce modèle, les variables aléatoires prennent valeur dans $\{-1, 1\}$ et sont disposées dans sur une grille de dimension arbitraire. Le potentiel d'interaction s'écrit, pour chaque couple de variables voisine $(i, j), (k, l)$, $\phi_{(i,j),(k,l)}(x_0, x_1) = \lambda x_0 x_1$. Un λ négatif indique un potentiel attractif, les variables voisines auront tendance à prendre la même valeur.

Pour $\lambda < 0$, les cases voisines ont tendance à prendre la même valeur, ce qui crée de grandes surfaces connexes minimisant la surface de contact avec l'autre valeur. Un phénomène intéressant est que lorsque les potentiels sont faibles, c'est le terme d'entropie dans la KL-divergence qui prend le dessus.



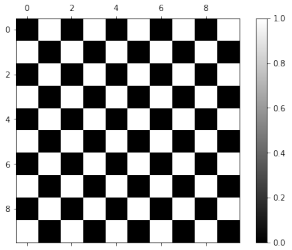
(a) Potentiels attractifs forts



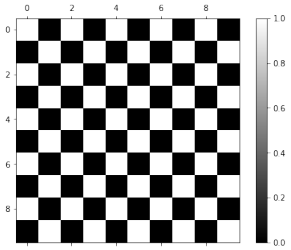
(b) Potentiels attractifs faibles

FIGURE 1 – Modèle d'Ising attractif

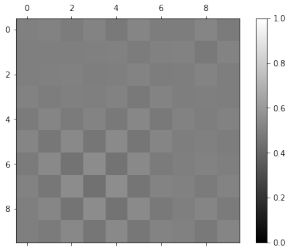
Pour $\lambda > 0$, les cases voisines prennent des valeurs différentes, ce qui donne lorsque les potentiels sont forts l'inférence d'un damier. On remarque que l'on peut converger vers deux types de damiers équivalents.



(a) Potentiels répulsifs forts, 1er minimum global



(b) Potentiels répulsifs forts, 2eme minimum global



(c) Potentiels répulsifs faibles

FIGURE 2 – Modèle d'Ising répulsif

Ainsi l'inférence mean-field ne revient pas à calculer le produit des lois marginales de chaque variable, puisque dans le cadre de ce modèle la symétrie impose $P(X_{(i,j)} = -1) = P(X_{(i,j)} = 1) = 0.5$. La symétrie est brisée par l'initialisation aléatoire de la variable q avant les itérations mean-field.

1.3 Multi-modalité de l'inférence

Comme on peut le remarquer en faisant de l'inférence sur le modèle d'Ising, l'approximation mean-field est parfois trop imprécise. L'objectif était donc d'implémenter l'approximation mean-field multi-modale [BFF16] qui propose un raffinement de la loi approximante.

1.3.1 Comment affiner l'approximation mean-field

Au lieu d'approximer P par un produit sur les lois marginales, l'espace est partitionné en k distributions de probabilités conditionnelles, appelées **modes**, et chaque mode est approximé par mean-field. L'enjeu est donc de partitionner l'espace d'état en k conditionnements C_m représentatifs de la multiplicité des solutions dans la distribution de probabilité et d'associer à chaque mode une probabilité μ_m. On a alors :

$$P(X) = \sum_{m=1}^{k} P(X|C_m)P(C_m)$$

qui s'approxime en

$$Q(X) = \sum_{m=1}^{k} Q_m(X)\mu_m$$

avec

— $Q_m(X) = \text{MF-approximation}(P(X|C_m))$

— $\mu_m = \dfrac{\mathbb{E}_{Q_m}(-\phi(X))}{\sum\limits_{m'=1}^{k} \mathbb{E}_{Q_{m'}}(-\phi(X))}$

Clamping par transition de phase Pour choisir les variables à fixer, un phénomène lié au principe de l'approximation mean-field est utilisé. En effet, soit un paramètre T, la température, qui crée une nouvelle gamme de distributions de probabilité

$$P^T(X = x) = \frac{1}{Z^T} exp(-\frac{1}{T}\phi(x))$$

— Quand $T \to +\infty$, c'est le terme $\sum Qlog(Q)$ de la KL-divergence qui domine, favorisant une haute entropie de Q, donc la distribution uniforme.

— Quand $T \to 0$, c'est le terme $-\sum Qlog(P)$ de la KL-divergence qui domine, et cette fois l'inférence mean-field donnera un maximum-a-posteriori de la distribution, c'est-à-dire une assignation des variables qui minimise l'énergie (ou du moins localement, car plus T diminue et plus la fonction à minimiser devient non-convexe).

— En balayant T de T_0, une température faible, à $+\infty$, on remarque que des variables qui étaient fixées quand $T = T_0$ deviennent subitement indécises, le terme d'entropie prenant le dessus. Ce sont ces variables qu'il faut fixer pour découvrir au mieux la distribution de probabilités. Ce phénomène est ce que l'on appelle en physique une **transition de phase**.

1.3.2 Implémentation

L'implémentation de l'inférence multi-modale par mean-field est possible grâce à l'implémentation optimisée du mean-field simple. En effet, on n'a pas dans le cadre général une expression pour la température de transition de phase, donc on est obligé de partir d'une température initiale et de l'augmenter petit à petit pour découvrir une transition de phase.

Data: Le mode initial M
Le CRF représenté par ϕ
Result: (i, j) les coordonnées de la variable à fixer, et l la valeur choisie
$T = 1/5$
$H_0 = \text{Entropie}(MF(\phi, M, T))$
$H_T = \text{Entropie}(MF(\phi, M, T))$
$h_{low} = 0.3\ log_2(d)$
$h_{high} = 0.7\ log_2(d)$
while *not(any($H_0 < h_{low}$ and $H_T > h_{high}$))* **do**
  T = T*1.2
  H_T = Entropie(MF(ϕ, T))
end
q = $MF(\phi, M, T)$
i,j = Argmax(H_T)
l = Argmax(q[i,j])
Renvoyer $(i, j), l$

Algorithm 1: Sélection des variables à fixer

Les constantes sont basées sur les résultats empiriques de Pierre Baqué et fonctionnent bien dans notre cadre de travail. Quand on obtient les variables $(i, j), l$ il suffit ensuite de créer deux nouveau modes à partir de M : $M' = M \bigcup [X_{(i,j)} = l]$ et $M'' = M \bigcup [X_{(i,j)} \neq l]$.

L'algorithme décrit ci-dessus fut développé en TensorFlow et permet de générer plusieurs modes dont on peut calculer ensuite les probabilités. La garantie que les modes choisis approximent au mieux la distribution finale n'est pas assurée mais elle permet dans tous les cas d'affiner l'approximation mean-field classique.

1.3.3 Modèle d'Ising

Le modèle d'Ising, qu'il soit attractif ou répulsif est un exemple typique présentant un caractère multi-modal. Les figures précédentes nous ont montré que ce phénomène de transition de phase se retrouve dans ce modèle. En appliquant l'algorithme du mean-field multi-modal, on obtient une meilleure approximation de la distribution P : en fixant une variable à -1 ou à 1, toutes les autres variables se déterminent "automatiquement", ce qui fait que l'on obtiens deux modes de probabilité 0.5 chacun, correspondant aux configurations où toutes les variables ont la même valeur dans le cas attractif, ou aux deux damiers possibles dans le cas répulsif.

2 Apprentissage automatique multi-modal

Après l'implémentation des méthodes d'inférence, l'enjeu du stage était surtout d'identifier ce que peut apporter la multi-modalité de l'inférence dans un cadre d'apprentissage automatique.

2.1 Enjeux

L'apprentissage profond est une méthode d'apprentissage supervisé très efficace en pratique. Seulement elle trouve ses limites quand le problème à résoudre est ambigu. Parmi ces problèmes on trouve l'apprentissage de modèles inverses et la résolution de puzzles ayant plusieurs solutions. Le jeu d'entraînement peu alors comporter des exemples de type $(X_1, Y_1), (X_2, Y_2)$ avec X_1 et X_2 proches et Y_1 et Y_2 éloignés. Une méthode pour résoudre ce problème est de transformer le problème en un problème non ambigu en pratiquant un sous-échantillonage du jeu d'apprentissage, comme c'est fait dans la résolution du problème de sculpture de flux [SLD+17]. Optnet, une couche d'apprentissage profond se revendiquant de l'état de l'art pour la résolution du Sudoku [AK17], échoue quand on lui présente un jeu de donnée ambigu.

L'enjeu est donc d'explorer à quel point l'inférence structurée permet d'améliorer l'apprentissage automatique. Dans l'idée, les résultats seront meilleurs car :

— on apprend la structure du problème en le représentant par un CRF.
— la génération de plusieurs solutions permet d'exploiter au mieux chaque exemple.

2.2 Apprentissage automatique

Cadre de travail L'objectif est d'évaluer la capacité des CRF à capter les contraintes d'un problème dans un cadre d'apprentissage automatique supervisé. Un jeu de données d'entraînement est dans ce cadre un ensemble de couples (I, O) où I est l'instance du problème (un ensemble de variables fixées) et O une solution du problème donné ayant des contraintes fixées par I.

Principe Pour que le CRF apprenne les contraintes du problème, on cherche à maximiser $P(O|I)$ en modifiant ϕ, les potentiels du CRF. On ne peut pas aisément accéder à P, donc cette opération se fait en maximisant $MF(\phi, I)(O)$. Autrement dit on cherche à maximiser la likelihood des données par rapport à l'approximation mean-field.

Mean-field simple Le calcul de cette approximation est différentiable par rapport ϕ et on peut par conséquent faire une descente de gradient classique pour optimiser le CRF.

Mean-field multi-modal Dans le cadre de mean-field multi-modal, on n'a pas directement accès à la distribution de probabilité modélisée. Cependant on peut y accéder de la façon suivante :

$$Q^{MMMF(\phi,I)}(O) = \sum_{m=1}^{k} Q^{MMMF(\phi,I)}(O|C_m)Q^{MMMF(\phi,I)}(C_m)$$
$$Q^{MMMF(\phi,I)}(O) = Q_m(O)\mu_m \text{ avec m l'indice du mode correspondant à O}$$

$$log(Q^{MMMF(\phi,I)}(O)) = log(Q_m(O)) + log(\mu_m)$$

On peut calculer cette expression de façon différentiable, ce qui permet d'implémenter une descente de gradient comme précédemment.

2.3 Implémentation

TensorFlow est très populaire dans le monde de l'apprentissage automatique, et de ce fait propose une fonction `tf.gradients` qui calcule automatiquement le gradient d'une valeur par rapport à un ensemble de variables. De plus les fonctions d'inférence mean-field et multi-modal mean-field furent implémentées pour supporter le calcul par lot, ce qui optimise encore le calcul et permet de faire de l'apprentissage en un temps raisonnable.

Après quelques expériences, j'ai mis à jour le script d'apprentissage automatique pour utiliser l'optimiseur Adam [KB14]. Adam réalise une descente de gradient dont les paramètres (taux d'apprentissage) sont optimisés au fur et à mesure de l'apprentissage, ce qui apporte en pratique une convergence bien plus efficace.

2.4 Application au problème du Sudoku

2.4.1 Introduction

Sudoku Le Sudoku est un puzzle sur une grille de 9x9 cases dont le but est de remplir les cases en respectant un ensemble de conditions :

- Chaque ligne doit comporter l'ensemble des chiffres de 1 à 9.
- Chaque colonne doit comporter l'ensemble des chiffres de 1 à 9.
- Chacun des 9 carrés de taille 3x3 pavant la grille doit comporter l'ensemble des chiffres de 1 à 9.

On peut facilement généraliser ce problème à des grilles de taille $N^2 \times N^2$, N définissant l'ordre du Sudoku. Il est intéressant de noter que la généralisation du Sudoku est NP-complète [YS03].

Jeu de données d'apprentissage Classiquement, les grilles de Sudoku sont générées pour avoir une unique solution, mais on peut très bien créer un jeu d'entraînement comportant des grilles ayant strictement plus d'une solution. Pour cela on crée une grille de Sudoku complète et on supprime des variables jusqu'à ce que la grille comporte plusieurs solutions puis le couple (grille partielle, grille complète) est sauvegardé. Cette étape de création du jeu de donnée n'a pas été particulièrement optimisée, les jeux d'entraînement comportant au plus 10000 exemples.

Padding Le modèle étant dans un premier temps invariant par translation, chaque variable dispose des mêmes contraintes relativement à ses voisins. Un padding de variables prenant pour valeur zéro a été ajouté entre les carrés pour donner au CRF la capacité d'identifier les carrés.

$X_{0,3}$	$X_{1,3}$			$X_{2,3}$	$X_{3,3}$		
$X_{0,2}$	$X_{1,2}$			$X_{2,2}$	$X_{3,2}$		
$X_{0,1}$	$X_{1,1}$			$X_{2,1}$	$X_{3,1}$		
$X_{0,0}$	$X_{1,0}$			$X_{2,0}$	$X_{3,0}$		

FIGURE 3 – Padding pour le Sudoku d'ordre 2. Les cases vides prennent toutes la valeur 0. Les cases blanches correspondent à la grille de Sudoku et prennent valeur dans $[|1, 4|]$.

2.4.2 Résultats

Pour comparer les différents algorithmes d'apprentissage, plusieurs jeux de données furent générés :
- le dataset simple comporte uniquement des exemples ayant une seule solution.
- le dataset difficile comporte des exemples à strictement plus d'une solution, avec une moyenne de 2.5 solutions par exemple.
- le dataset difficile (12) comporte des exemples à structure plus de 8 solutions, avec 12 solutions en moyenne.

Chaque jeu de donnée comporte 10000 exemples, et l'entraînement se fait sur 50 itérations. Les tests se sont fait sur le dataset difficile et le score correspond au pourcentage de grilles correctes générées. Enfin, la ligne de base est un réseau de neurones convolutionnel auquel on a ajouté l'information de localisation et entraîné de la même manière.

On remarque dans ce graphique 4 que le réseau de neurones convolutionnel échoue complètement au test de l'apprentissage sur un jeu de données ambigu, là où le mean-field simple est bien plus résilient (on passe de 100% à 74%). Cela s'explique par le fait qu'un réseau de neurones convolutionnel n'est pas construit pour apprendre la structure du problème. Si pour une entrée particulière le réseau renvoie une réponse correcte mais pas celle attendue, les gradients se propageront de manière à augmenter l'ambiguïté sur les variables fausses de façon à diminuer l'entropie croisée.

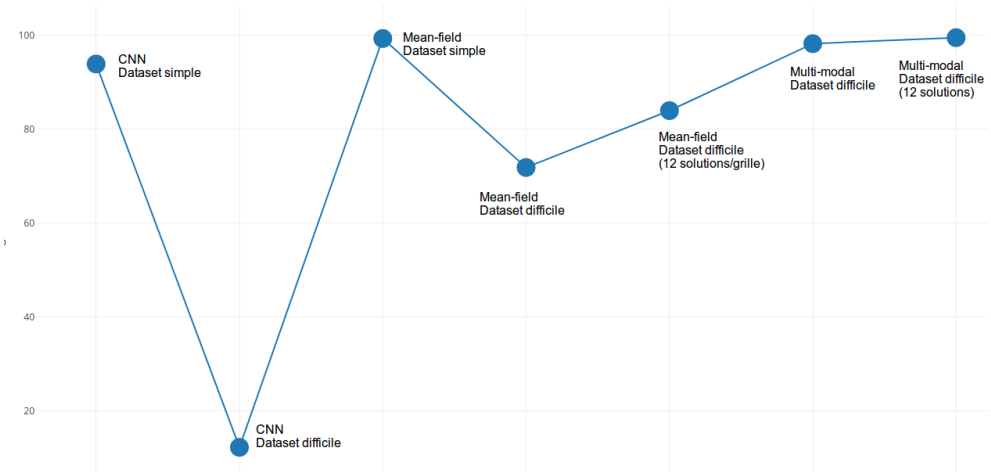


FIGURE 4 – Comparaison entre réseau de neurones convolutionnel, apprentissage mean-field et version multi-modale sur le Sudoku 4x4. En ordonnée, le pourcentage d'exemples réussis sur le jeu de test à plusieurs solutions.

L'apprentissage multi-modal permet de retrouver les performances initiales du mean-field même sur un jeu de données ambigu. Ces résultats se retrouvent aussi dans l'apprentissage du Sudoku 9x9, même si les performances sont moins bonnes étant donné la difficulté du problème.

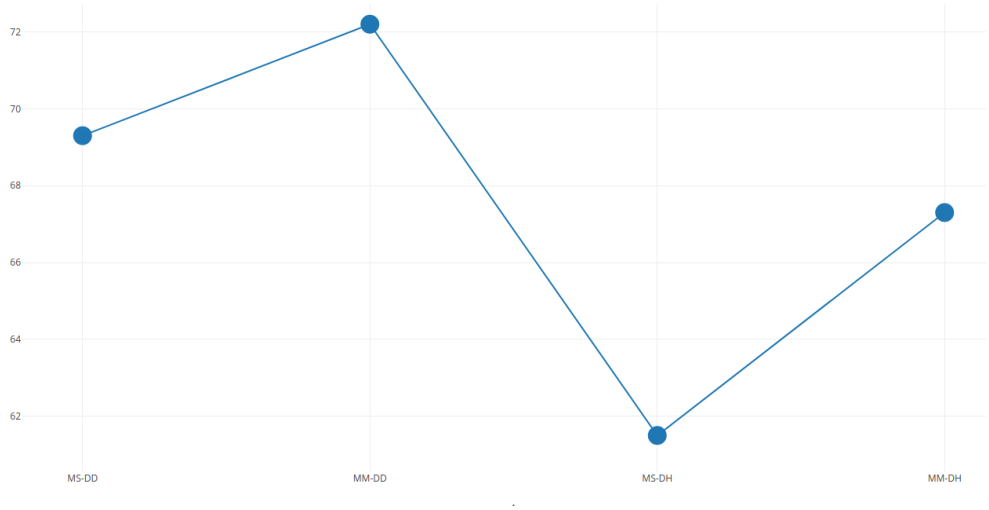


FIGURE 5 – Comparaison entre mean-field simple (MF), multi-modal mean-field (MF) sur les jeux de données simple (DS) et difficile (DD) sur le Sudoku 9x9. En ordonnée, le pourcentage d'exemples réussis sur le jeu de test à plusieurs solutions.

Enfin, un avantage supplémentaire de l'apprentissage de CRF est que la fonction d'énergie apprise constitue un puissant **discriminateur** de grilles. Après l'apprentissage, on peut générer plusieurs grilles grâce à l'inférence multi-modale, calculer leur énergie et si elles sont correctes, pour obtenir l'histogramme suivant. Il nous montre qu'il y a bien un saut d'énergie entre grille correcte et grille incorrecte, et que les grilles correctes sont toutes équivalents en terme d'énergie (et donc de probabilité).

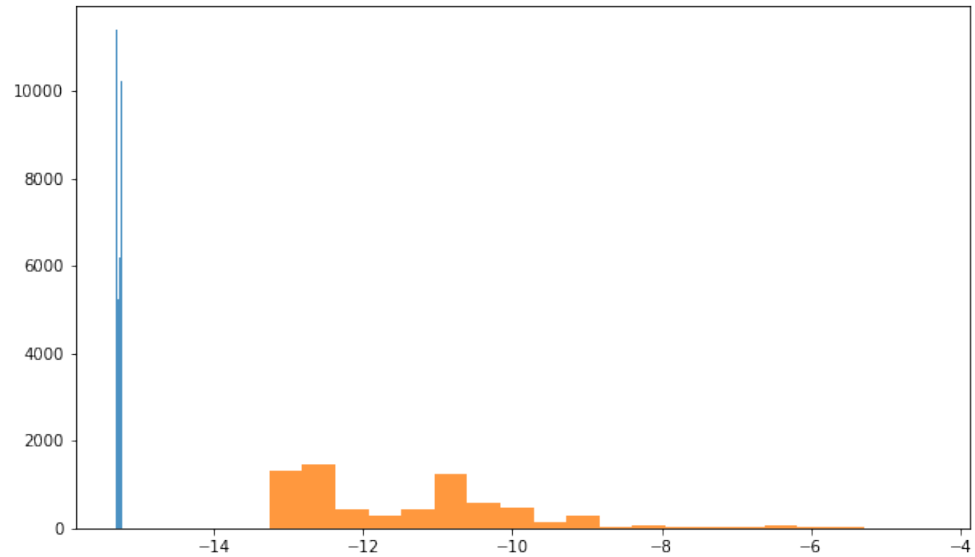


FIGURE 6 – Pouvoir discriminateur du CRF appris. En bleu l'énergie des grilles correctes générées. En orange l'énergie des grilles incorrectes générées. Histogramme dont l'abscisse représente l'énergie des grilles sur le CRF appris.

2.4.3 Conclusion

L'objectif était de créer un framework d'apprentissage automatique de CRF sans connaissance a priori. Cependant l'ajout d'un padding entre les carrés est un ajout d'information important qui fait que l'on ne peut pas comparer ces résultats à l'état de l'art. En revanche, cela nous montre bien que le multi-modal permet d'améliorer l'apprentissage lorsque le problème est ambigu. De plus la résilience du mean-field classique quand le dataset est difficile est intéressante dans la mesure où l'apprentissage d'un réseau profond classique à la même tâche échoue complètement sur le dataset multi-modal.

Cependant les résultats en 9x9 ne sont pas parfaits, cela est dû au fait que la résolution de certaines grilles de Sudoku ne peut se faire que par backtracking : il faudrait donc explorer beaucoup de modes pour générer une grille finale correcte. Une génération de 4 modes crée alors seulement des minima locaux d'énergie, autrement dit des grilles vérifiant un maximum de contraintes. Cependant l'enjeu n'était pas de résoudre exactement le Sudoku, mais plutôt d'en apprendre les règles du jeu. La résolution se fait par exemple par réduction à un problème de satisfiabilité [ERT12].

3 Expansion du modèle par l'ajout de filtres

Si on veut enlever le padding, on doit ajouter d'une façon ou d'une autre une information de localisation. L'objectif est de représenter une gamme plus complète de CRF, tout en gardant une facilité computationnelle. On cherche donc un modèle transitoire entre le CRF invariant par translation ($n^2 \times p^2$ paramètres) et le CRF dense complètement connecté ($n^4 \times p^2$ paramètres).

3.1 Définition d'un modèle plus expressif

Au lieu d'avoir un unique filtre qui gouverne l'énergie d'interaction locale de chaque variable, le modèle dispose d'une banque de filtres et d'une fonction qui pour chaque variable choisit une composition de ces filtres à appliquer. Nous appelons ce nouveau modèle le **CRF multi-filtre**.

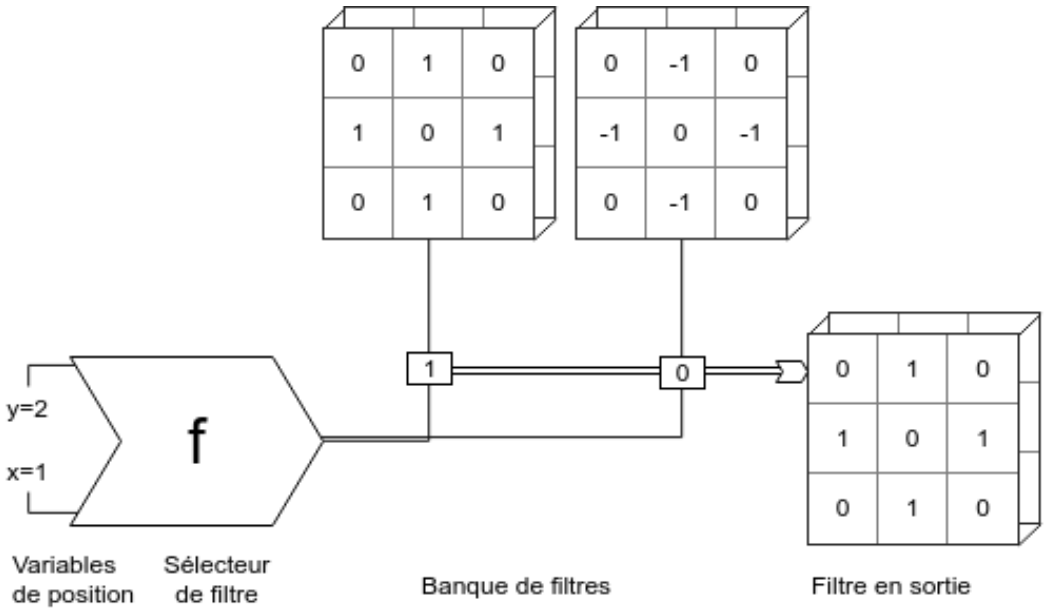


FIGURE 7 – Exemple de sélecteur sur deux filtres.

Implémentation La transition d'un à plusieurs filtres est assez simple. En effet comme on travaille sur une banque de filtres, l'opération de mise à jour effectuée est une combinaison linéaire des opérations de mise à jour pour chaque type de filtre, les coefficients étant calculés pour chaque position dans la grille grâce à la fonction de sélection. Cette fonction est dans notre cas un réseau de neurones dense. Ce réseau de sélection est appris comme le reste des paramètres.

3.2 Application au Sudoku et résultats

Grâce au pouvoir supplémentaire de représentation accordé au modèle, on se permet de retirer la séparation entre les différents carrés. Ainsi plus aucune information a priori sur la résolution du Sudoku n'est donnée lors de l'apprentissage, tout en gardant un modèle qui se veut général : c'est l'idée de l'apprentissage profond.

L'effet de la suppression du padding démontre l'existence d'une barrière logique qui empêche la résolution de l'ensemble des grilles de Sudoku, comme on peut le voir dans la figure 8. En effet, on ne peut pas modéliser par un CRF invariant par translation la contrainte d'unicité des variables sur chaque carré.

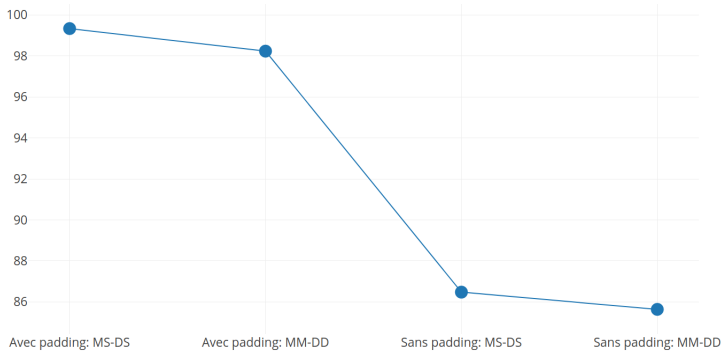
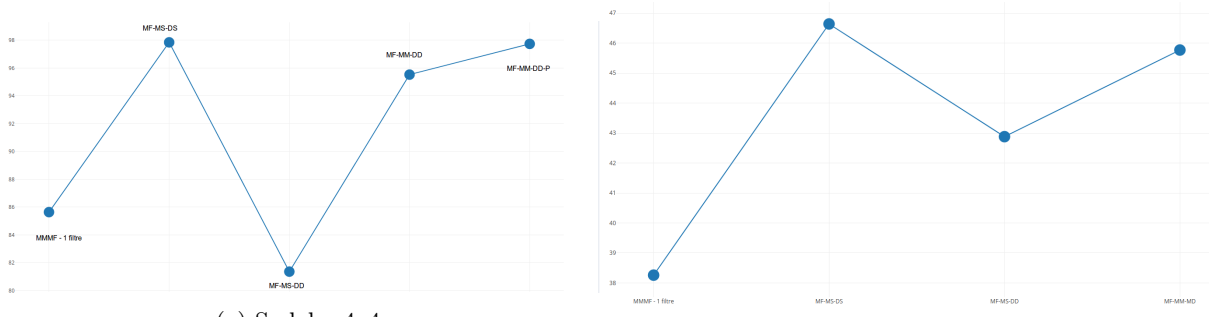


FIGURE 8 – Effet de la suppression du padding. Impossibilité formelle de modéliser par un CRF invariant par translation les contraintes sur les carrés.

En revanche l'expansion du modèle permet bien de retrouver les performances initiales, et cette fois sans apport d'information extérieur. Cette figure 9a nous montre que l'apprentissage multi-modal sur le Sudoku 4x4 permet encore de dépasser les problèmes d'ambiguité du jeu de données difficile et d'obtenir des scores presque parfaits.

En revanche, nous avons constaté dans la figure 9b un échec de l'apprentissage dans le cadre du Sudoku 9x9 qui ne dépasse pas 50% en terme de score. En théorie un CRF multi-filtre peut représenter les contraintes liées au Sudoku 9x9, mais sans information a priori l'apprentissage est lent et très dépendant des paramètres.



(a) Sudoku 4x4

(b) Sudoku 9x9

FIGURE 9 – Comparaison des apprentissages mean-field multi-filtre. MF : Multi-Filtre. MS : Mean-field Simple. MM : Mean-field Multi-modal. DS : Dataset Simple. DD : Dataset Difficile (plusieurs solutions). P : pré-apprentissage mono-modal puis apprentissage multi-modal.

4 Conclusion

Apport du travail effectué Au cours de ce stage j'ai pu exploiter les méthodes classiques d'apprentissage automatique pour les intégrer dans un cadre d'apprentissage de problèmes structurés. Le framework développé en Python est assez général pour être réutilisé, et comporte les deux idées principales explorées : l'apprentissage multi-modal et la modélisation de CRF multi-filtre. Les expériences effectuées et

l'application au Sudoku démontrent la robustesse de l'apprentissage mean-field dans le cadre de problèmes ambigus et difficiles à résoudre via des méthodes d'apprentissage automatique classiques. Le code est disponible sur Github à l'adresse https://github.com/TheLortex/mean-field-inference.

Perspectives soulevées Les résultats obtenus soulèvent deux problèmes :

- l'apprentissage d'un CRF modélisé par le multi-filtre est bien plus chaotique que l'apprentissage d'un CRF invariant par translation. En effet la sélection par réseau de neurones doit elle aussi être apprise, ce qui rend la fonction de coût à minimiser très irrégulière. L'optimiseur Adam aide à obtenir une descente de gradient efficace mais elle est très dépendante des conditions initiales et des paramètres. Un enjeu soulevé par ce problème est donc de comprendre comment améliorer les méthodes d'apprentissage de CRF, que ce soit en mono ou multi-modal.
- même dans le cadre introduit par le modèle du CRF multi-filtre, ce framework ne s'introduit pas directement pour résoudre n'importe quel problème. L'intégration dans un cadre d'apprentissage profond est donc à étudier. On peut par exemple imaginer un réseau profond paramétrant une couche d'inférence mean-field qui infère le résultat voulu, ou même plusieurs résultats dans un cadre multi-modal.

Ressenti sur le stage Personnellement, ce stage fut une expérience très enrichissante pour moi. J'ai découvert un pan de l'apprentissage automatique qui m'était inconnu, alors que sa base mathématique dispose d'un intérêt prometteur. Le développement d'un code dans un domaine aussi expérimental est une expérience frustrante (avancées par essais et erreurs) mais surtout gratifiante quand le résultat est à la hauteur de ses espérances. Enfin, le cadre de travail à l'EPFL était très positif : je travaillais dans le bureau de mon maître de stage qui, tout en me laissant une liberté d'expérimentation, était ravi de répondre à mes questions.

Je remercie donc chaleureusement Pierre Baqué et Pascal Fua qui m'ont encadré cet été, m'offrant une très motivante introduction à la recherche en informatique.

Travail à suivre Il est prévu pour la suite de publier les résultats de ces travaux au JMLR (Journal of Machine Learning Research) en tant que partie du corpus d'articles de Pierre Baqué sur les méthodes mean-field. De plus, après discussion avec un membre du CVLab travaillant sur le problème de la reconstruction de surface 3D à partir d'une image 2D de cette surface, nous avons convenu qu'il serait intéressant d'intégrer l'apprentissage mean-field dans le modèle utilisé. En effet, le modèle utilisé est pour l'instant un réseau profond à base de couches convolutionnelles et il trouve sa limite lorsque les images d'entraînement ne sont pas toutes éclairées de la même façon. Dans ce cas il y a une ambiguïté naturelle qui apparaît entre concavité et convexité qui fait que le réseau renvoit une surface 3D moyenne entre les deux résultats possibles. Comme dans le cas du modèle d'Ising, on espère qu'une couche finale d'inférence mean-field pourrait briser la symétrie et choisir l'une des deux solutions correctes.

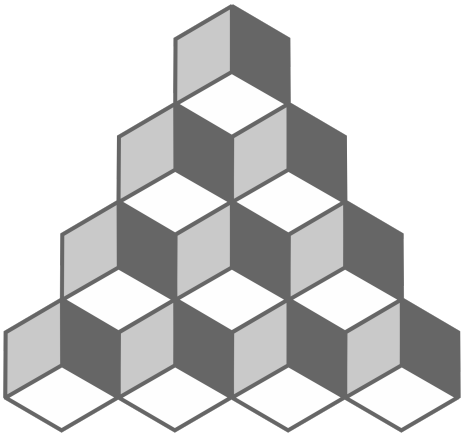


FIGURE 10 – Selon que la lumière éclaire d'au dessus ou d'en dessous, l'interprétation 3D est différente.

Références

[AK17] Brandon Amos and J. Zico Kolter. Optnet : Differentiable optimization as a layer in neural networks. *CoRR*, abs/1703.00443, 2017.

[BBFF15] Pierre Baqué, Timur M. Bagautdinov, François Fleuret, and Pascal Fua. Principled parallel mean-field inference for discrete random fields. *CoRR*, abs/1511.06103, 2015.

[BFF16] Pierre Baqué, François Fleuret, and Pascal Fua. Multi-modal mean-fields via cardinality-based clamping. *CoRR*, abs/1611.07941, 2016.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[ERT12] M. Ercsey-Ravasz and Z. Toroczkai. The chaos within Sudoku. *Sci Rep*, 2 :725, 2012.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[SLD+17] Daniel Stoecklein, Kin Gwn Lore, Michael Davies, Soumik Sarkar, and Baskar Ganapathysubramanian. Deep learning for flow sculpting : Insights into efficient learning using scientific simulation data. In *Scientific reports*, 2017.

[YS03] Takayuki YATO and Takahiro SETA. Complexity and completeness of finding another solution and its application to puzzles. E86-A, 05 2003.