université de TOURS

LIFAT EA 6300

Laboratoire d'Informatique Fondamentale et Appliquée de Tours

Apprentissage machine et Optimisation combinatoire

Romain Raveaux

Des problèmes, des liens et des pistes

romain.raveaux@univ-tours.fr

Objectifs

1. Montrer les problèmes d'optimisation discrète qui apparaissent en apprentissage machine

2. Montrer les problèmes d'apprentissage qui apparaissent dans les méthodes d'optimisation discrète

3. Montrer les problèmes passés (avant le Deep Learning) et les problèmes plus actuels

4. Donner des pistes de collaborations et des réflexions

Objectifs

1. Montrer les problèmes d'optimisation discrète qui apparaissent en apprentissage machine
   A. Les problèmes d’inférence et apprentissage de modèles graphiques probabilistes : Problème d’étiquetage de graphe
   B. Les problèmes de sélection : Par exemple choisir des vecteurs dans un dictionnaire de manière parcimonieuse. Sélectionner des poids d'un réseau de neurones, Sélection de caractéristiques, Trouver les P-médian, Matrice de rang faible, Factorisation de matrice, ...
   C. Les problèmes d'affectation : Par exemple le problème du transport optimal, le problème du graph matching.
   D. Neural architecture search (NAS)

Objectifs

2. Montrer les problèmes d'apprentissage qui apparaissent dans les méthodes d'optimisation discrète
   A. Apprendre pour améliorer les données d’entrées pour résoudre plus simplement le problème
   B. Apprendre à bien résoudre

Objectifs

4. Donner des pistes de collaborations et des réflexions
    1. Représenter les problèmes d'ordonnancement comme des images
    2. Représenter les problèmes d'ordonnancement comme des graphes
    3. Neural architecture search (NAS), AutoML

Introduction à l’apprentissage machine

L’apprentissage

- Soit X, l’espace d’entrée
  - Par exemple, l’espace de toutes les images possibles
    - $X = R^{3 \times H \times L}$
    - $x \in X$
- Soit Y, l’espace de sortie
  - Y= {0, 1} : Variable discrète propice à l’optimisation discrète
    - 0 : l’image représente un chien
    - 1 : l’image représente un chat
    - $y \in Y$
- Soit f, une fonction reliant l’espace d’entrée à l’espace de sortie
  - $f : X \rightarrow Y$
  - $f(x) \in Y$

L’apprentissage

- Comment construire f ?
  - À la main
    - grâce à des règles, on encode informatiquement la connaissance de l’expert
      - Système expert : si pixels blancs alors image de chat
  - A partir des données → Apprentissage machine
    1. Fixer un modèle pour f : Hypothèses sur notre problème.
    2. Introduire des paramètres (W) dans la fonction f
       - $f : X \times W \rightarrow Y$

L’apprentissage machine

- Il faut des données d’apprentissage
  - $App = \{x_i, y_i\}_{i=1}^{M}$ → Des couples : (image, classe)
  - Besoin d’experts pour fournir les données d’apprentissage surtout les y_i
- L’apprentissage = Trouver les paramètres w tel que le modèle paramétré minimise les erreurs de décisions.
  - Soit l, une fonction qui mesure l’ erreur de décision
  - $w^* = arg \min_{w \in W} \sum_{x_i, y_i \in App} l(\, f(x_i, w)\, , y_i)$ → PB1: Problème d’apprentissage

- Avec w ∈ R^d. d=500 milliards
- Avec M = 30 Milliards

Modèle NLP créée par les GAFAM

LIFAT : d=1 million et M=10 000

L’apprentissage machine

- Phase de décision/prédiction/inférence
  - $Test = \{x_i, y_i\}_{i=1}^{m}$ → Des données pour tester
  - Besoin d’experts pour fournir les données des tests surtout les y_i
  - On utilise la fonction f paramétré pour résoudre notre problème
    - $x_i \in Test$
    - $f(x_i, w^*) = \widehat{y_i}$
  - f doit être rapide à calculer :
    - En test : besoin temps réel ? Voiture autonome, SIRI, Alexa, ….
    - En apprentissage : beaucoup d’appels à f.

L’apprentissage machine

- $w^* = arg \min_{w \in W} \sum_{x_i, y_i \in App} l(\, f(x_i, w)\, , y_i)$ → Phase d’apprentissage
- Juste un problème d’optimisation ???
  - Non pas que car ce que l’on veut c’est que f(w) soit performant sur $Test = \{x_i, y_i\}_{i=1}^{m}$
  - Minimiser une fonction d’erreur sur une base d’apprentissage n’implique pas forcément être bon sur $Test$ → Théorie de l’apprentissage

- Théorie de l’apprentissage :
  - Les problèmes : Sur apprentissage et Sous apprentissage
  - Les solutions : contraindre le problème d’apprentissage (Stochastique Gradient Descente, dropout, regularization, …), insérer des connaissances a priori (bayesian prior …)

L’apprentissage machine : deep learning

- f : est une composition de fonction

$$f = f_1 \circ f_{\dots} \circ f_k$$

- PB1 est minimisé par descente de gradient.

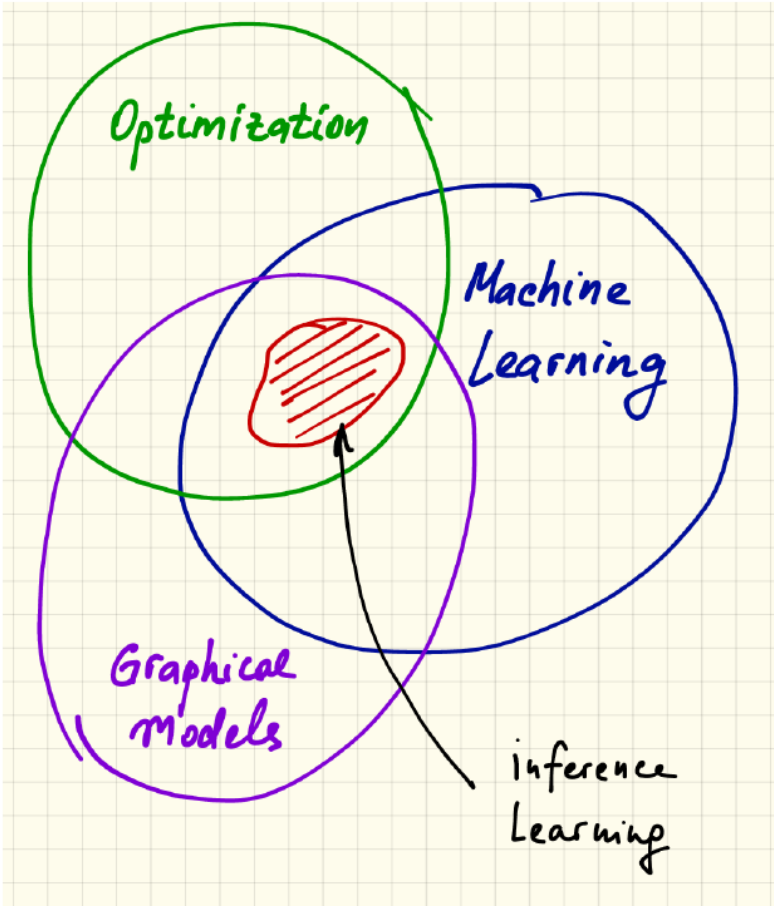
$$w_{t+1} = w_t - \eta \nabla w_t$$

- f : doit être dérivable

Objectifs

1. Montrer les problèmes d'optimisation discrète qui apparaissent en apprentissage machine
   A. Les problèmes d’inférence et apprentissage de modèles graphiques probabilistes : Problème d’étiquetage de graphes

Inférence et apprentissage de modèles graphiques probabilistes

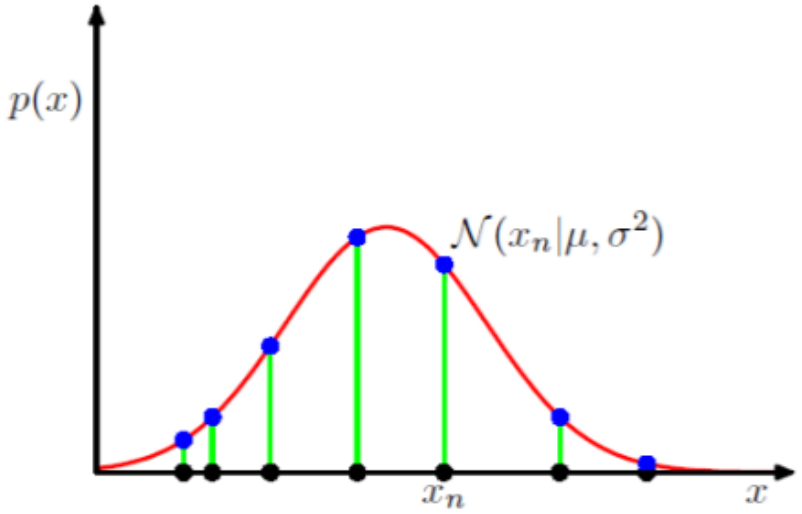


Apprentissage : ajuster des paramètres

Maximum likelihood

$$W^* = arg\max_W [Pr(x_1, \cdots, x_M|W)]$$

$$W^* = arg\max_W [\prod_{i=1}^{M} Pr(x_i|W)]$$

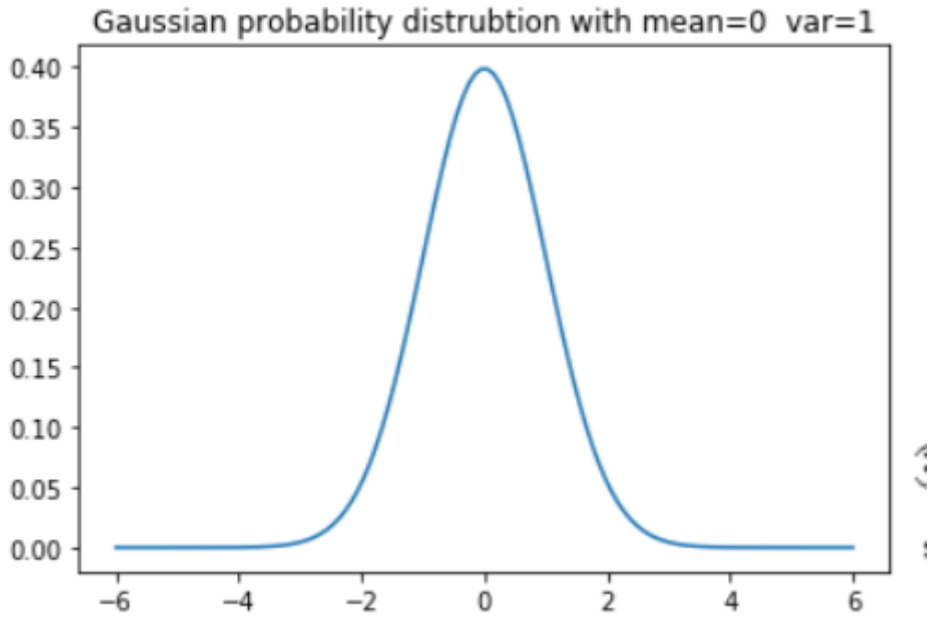


Assuming each data point was drawn independently from the distribution (i.i.d)
Independent and identically distributed → i.i.d

$$\hat{\mu}, \hat{\sigma}^2 = arg\max_{\mu,\sigma^2} Pr(\mathbf{x}|\mu, \sigma^2) \quad \Rightarrow \quad \hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

Inférence

$$Pr(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) = \mathcal{N}(x|\mu,\sigma^2)$$



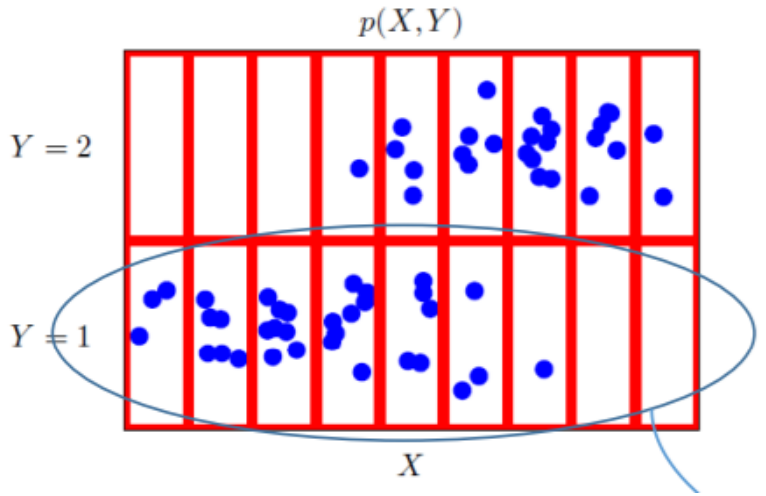
Soit f=Pr(y|x) : fonction reliant l'espace d'entrée à l'espace de sortie

Apprentissage : Maximum de vraisemblance

$$W^* = arg\max_W \left[\prod_{i=1}^{M} Pr(y_i|x_i;W)\right]$$

Inférence :

$$\Pr(y|x^{new})$$



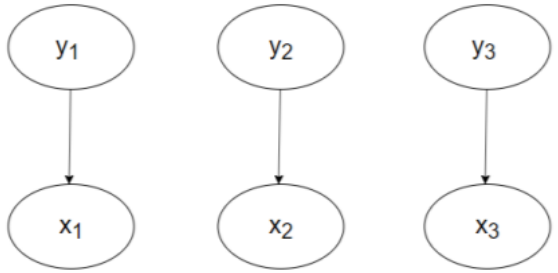
Soit f'=Pr(x|y) : fonction reliant l'espace de sortie à l'espace d'entrée

Apprentissage par maximum de vraisemblance :

$$W^* = arg\max_W \left[\prod_{i=1}^{M} Pr(x_i|y_i;W)\right]$$

Inférence :

$$Pr(x_1,\cdots,x_M|y_1,\cdots,y_M) = \prod_{i=1}^{M} Pr(x_i|y_i)$$



Liaisons entre des modèles locaux

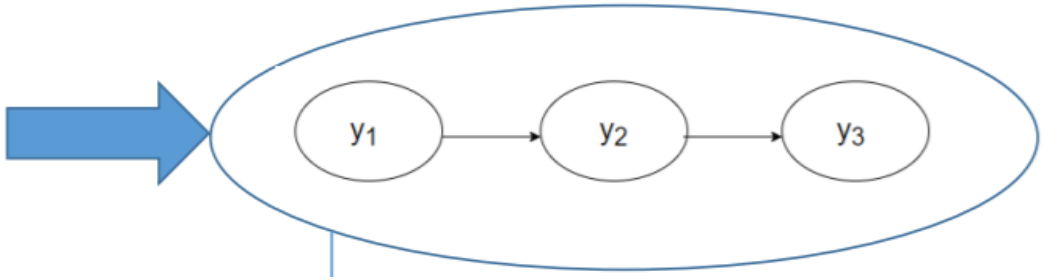
y_2 dépend y_1 → Dépendance séquentielle

Exemple: dans une vidéo, chaque image peut être classée comme une image de chat ou pas

La scène : un chat court dans le jardin

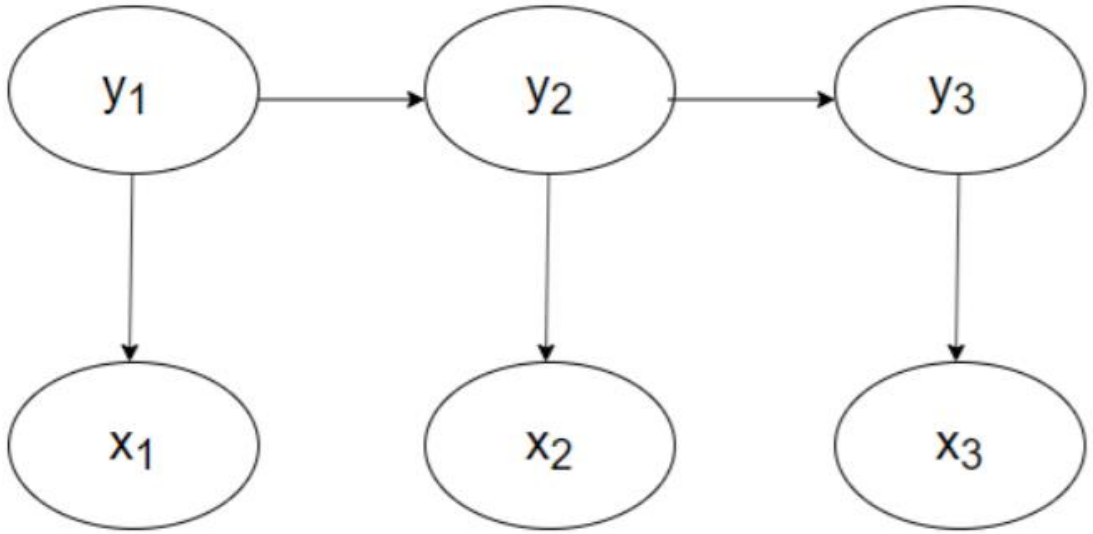
Si l'image i-1 est une image de chat alors l'image i est vraisemblablement une image de chat

$$Pr(y_1, \cdots, y_M) = Pr(y_1) \prod_{i=2}^{M} Pr(y_i|y_{i-1})$$



Liaisons entre des modèles locaux

$$Pr(y_1, \cdots, y_M; x_1, \cdots, x_M) = \left[\prod_{i=1}^{M} Pr(x_i|y_i)\right]\left[Pr(y_1)\prod_{i=2}^{M} Pr(y_i|y_{i-1})\right]$$



Inférence : problème d'optimisation

Inférence : On cherche une séquence y, les variables $y_i \in \{0,1\}$

$$\hat{y_1}, \cdots, \hat{y_M} = arg \max_{y_1, \cdots, y_M} \left[\prod_{i=1}^{M} Pr(x_i^{new}|y_i).Pr(y_1) \prod_{i=2}^{M} Pr(y_i|y_{i-1}) \right]$$

Let us enumerate all possible combinations:

- y_i can take K different states
- A sequence is composed of M elements
- → There are K^M possibilities
- Can we do better than a brute force enumeration ???

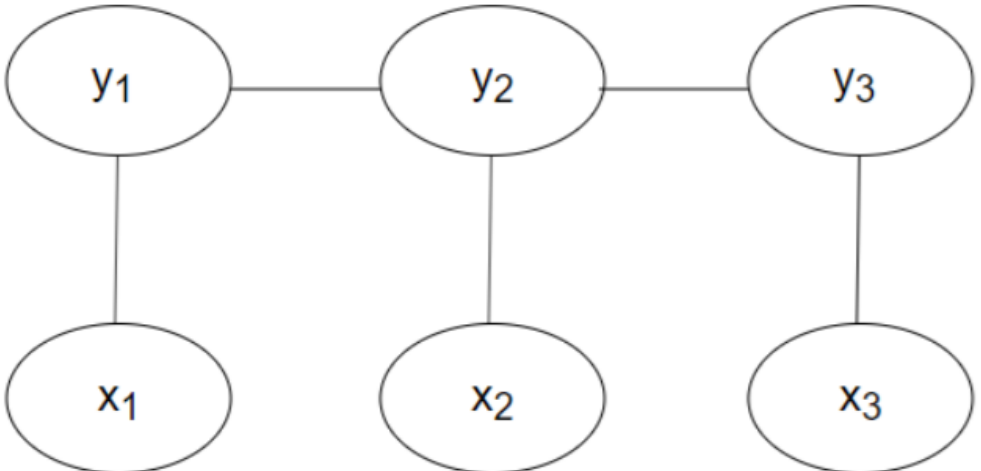
Viterbi : $O(MK^2)$ Programmation dynamique

Inférence : problème d'optimisation : CRF

$$\hat{y_1}, \cdots, \hat{y_M} = arg \max_{y_1, \cdots, y_M} \left[\prod_{i=1}^{M} \phi(x_i^{new}, y_i) \prod_{i=2}^{M} \zeta(y_i, y_{i-1}) \right]$$

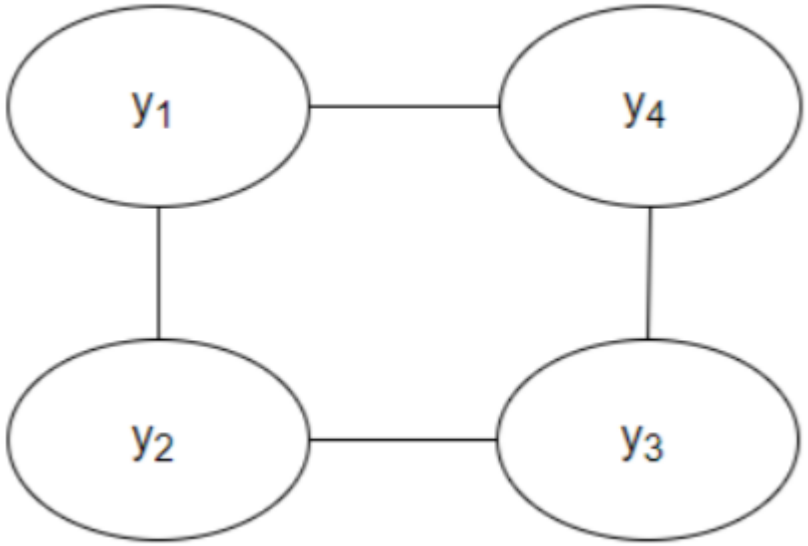
$\Pr(x_i|y_i = k)$ *is replaced by* $\phi(x_i, y_i)$: similarity function

$\Pr(y_i|y_{i-1})$ is replaced by $\zeta(y_i, y_{i-1})$: similarity function



Liaisons dangereuses entre des modèles locaux

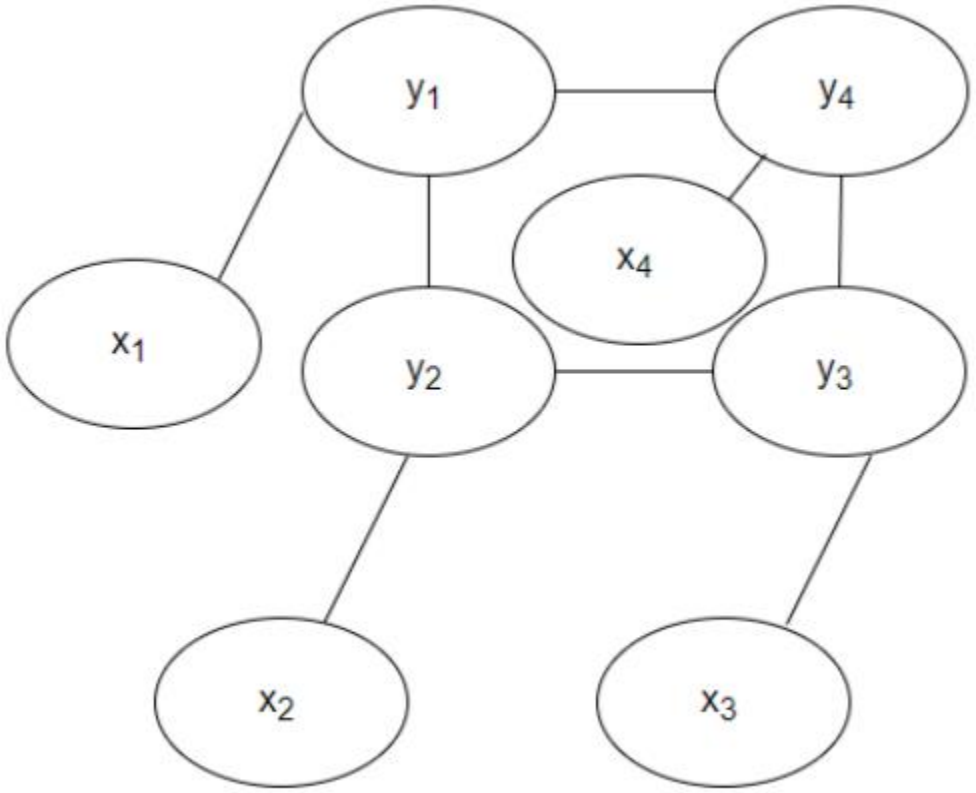
- Dépendance entre les variables : Grille 2D (graphe)
  - La dépendance n'est plus séquentielle



$$Pr(y_1, \cdots, y_M) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{C}} \zeta(y_i, y_j)$$

$$Z = \sum_{y_1 \in \mathcal{Y}} \sum_{y_2 \in \mathcal{Y}} \cdots \sum_{y_M \in \mathcal{Y}} \left[\prod_{(i,j) \in \mathcal{C}} \zeta(y_i, y_j) \right]$$

Liaisons dangereuses entre des modèles locaux

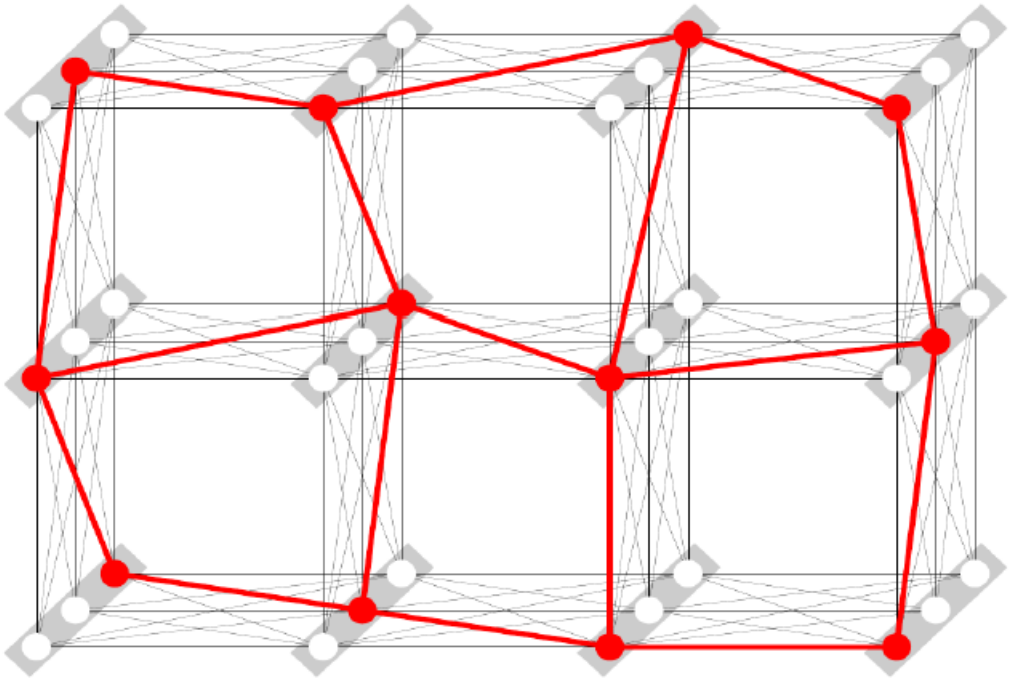


$$Pr(y_1, \cdots, y_M | x_1, \cdots, x_M) = \frac{1}{Pr(x_1, \cdots, x_M)} \frac{1}{Z_2} \left[\prod_{i=1}^{M} \phi(x_i, y_i) \right] \left[\prod_{(i,j) \in \mathcal{C}} \zeta(y_i, y_$$

$$Z_2 = \sum_{y_1 \in \mathcal{Y}} \sum_{y_2 \in \mathcal{Y}} \cdots \sum_{y_M \in \mathcal{Y}} \left[\prod_{i=1}^{M} \phi(x_i, y_i) \right] \left[\prod_{(i,j) \in \mathcal{C}} \zeta(y_i, y_j) \right]$$

Inférence : problème d'optimisation :
Problème d'étiquetage dans un graphe

$$\hat{y}_1, \cdots, \hat{y}_M = arg \max_{y_1, \cdots, y_M} \left[\prod_{i=1}^{M} \phi(x_i^{new} | y_i) \right] \left[\prod_{(i,j) \in \mathcal{C}} \zeta(y_i, y_j) \right]$$



Picture: T Werner. A Linear Programming Approach to Max-sum Problem: A Review

Inférence : problème d'optimisation

$$\hat{y}_1, \cdots, \hat{y}_M = arg \max_{y_1, \cdots, y_M} \left[\prod_{i=1}^{M} \phi(x_i^{new} | y_i) \right] \left[\prod_{(i,j) \in \mathcal{C}} \zeta(y_i, y_j) \right]$$

1. y sont des variables discrètes
2. Les dépendances des variables ne permettent plus la programmation dynamique: Graphe avec des boucles. → Parfois un problème NP-difficile
3. Utilisation de méthodes dite Graph Cut
    1. Modélisation comme des problèmes de Max Flow et Min cut
    2. Heuristique du type alpha-expansion algorithm

Les cas spécifiques

We can consider three cases:

1. binary MRFs (i.e., $y_i \in \{0,1\}$) where the costs for different combinations of adjacent labels are “submodular" (I m not sure I will explain what this means later). Exact MAP inference is tractable here.
2. multi-label MRFs (i.e., $y_i \in \{1, \dots, K\}$) where the costs are “submodular.“ Once more, exact MAP inference is possible.
3. multi-label MRFs where the costs are more general. Exact MAP inference is intractable, but good approximate solutions can be found in some cases.

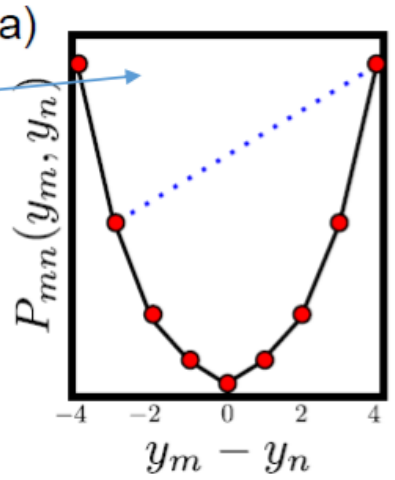
Fonction sous modulaires

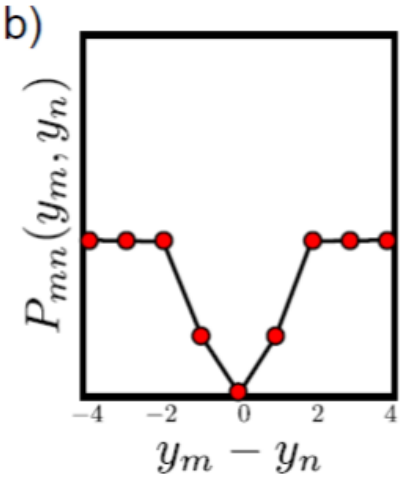
Inference for MRF : Submodular functions

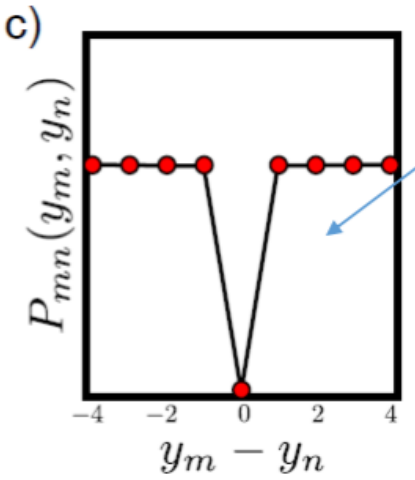
- Case of binary MRF:
- $P_{ij}(0,0) = \theta_{00}\ ;\ P_{ij}(0,1) = \theta_{01}\ ;\ P_{ij}(1,0) = \theta_{10}\ ;\ P_{ij}(1,1) = \theta_{11}$
- Submodular functions
  - $\theta_{01} + \theta_{11} - \theta_{00} - \theta_{11} \geq 0$
- Submodular functions = constraints on cost functions

Différentes fonctions de coûts

Quadratic
(Convex) cost
functions



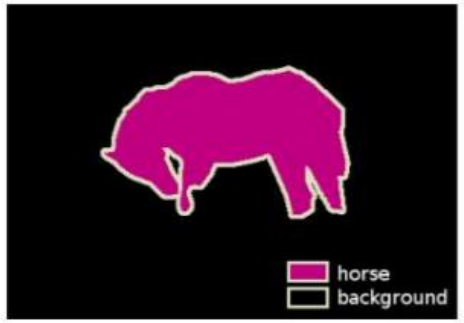




Potts model
Non-convex

Des exemples d'application

$\mapsto$



input: images $\mapsto$ output: segmentation masks

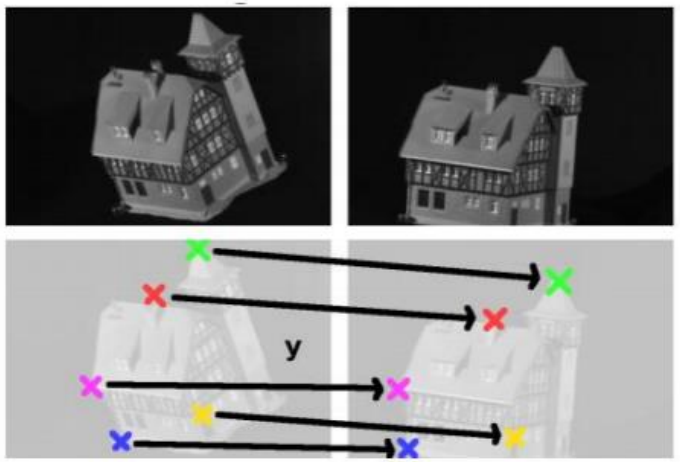
- input space $\mathcal{X} = \{\textit{images}\} \mathrel{\hat{=}} [0, 255]^{3 \cdot M \cdot N}$
- output space $\mathcal{Y} = \{\textit{segmentation masks}\} \mathrel{\hat{=}} \{0, 1\}^{M \cdot N}$
- (structured output) prediction function: $f : \mathcal{X} \to \mathcal{Y}$

$$f(x) := \operatorname*{argmin}_{y \in \mathcal{Y}} E(x, y)$$

- energy function $E(x, y) = \sum_i w_i^\top \varphi_u(x_i, y_i) + \sum_{i,j} w_{ij}^\top \varphi_p(y_i, y_j)$

Des exemples d'application

input: image pairs



output: mapping $y : x_i \leftrightarrow y(x_i)$

- prediction function: $f : \mathcal{X} \to \mathcal{Y}$

$$f(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} F(x, y)$$

- scoring function $F(x, y) =$
$\sum_i w_i^\top \varphi_{sim}(x_i, y(x_i)) + \sum_{i,j} w_{ij}^\top \varphi_{geom}(x_i, x_j, y(x_i), y(x_j))$

Apprentissage : problème d'optimisation : CRF

$$\hat{W} = arg\max_{W} \frac{1}{(Z_2(W_1, W_2))^N} \prod_{n=1}^{N} \left[\prod_{i=1}^{M} \phi(x_i^{(n)}, y_i^{(n)}, W_1) \prod_{(i,j)\in\mathcal{C}} \zeta(y_i^{(n)}, y_j^{(n)}, W_2) \right]$$

$$Z_2(W) = Z_2(W_1, W_2) = \sum_{y_1\in\mathcal{Y}} \sum_{y_2\in\mathcal{Y}} \cdots \sum_{y_M\in\mathcal{Y}} \left[\prod_{i=1}^{M} \phi(x_i, y_i, W_1) \prod_{(i,j)\in\mathcal{C}} \zeta(y_i, y_j, W_2) \right]$$

$w_1 \in R^{d1}$ et $w_2 \in R^{d2}$

Algorithme de résolution : Structured learning

- Structured perceptron

  - Let β be a weight vector of length n.
  - For a pre-determined number of iterations:
    - For each sample x in the training set with true output t:
      - Find a feasible solution $y^* = \underset{y \in GEN(x)}{arg\,min}\,(\beta^T \phi(x, y))$
      - Update β, from y^* to t: $\beta = \beta + \alpha(-\Phi(x, y^*) + \Phi(x, t))$

Algorithme de résolution : Structured learning : Structured SVM

This simple paradigm can be extended to the large margin framework [Tsochantaridis et al., 2005]:

$$\min_{\beta} \frac{1}{2}||\beta||_2^2 \tag{B.5}$$

Such that: (B.6)

$$\beta.\Phi(x, y^*) - \beta.\Phi(x, t) \geq 1 \tag{B.7}$$

$$\forall (x, t) \in \mathcal{D} \text{ and } y^* \in \mathcal{Y} \tag{B.8}$$

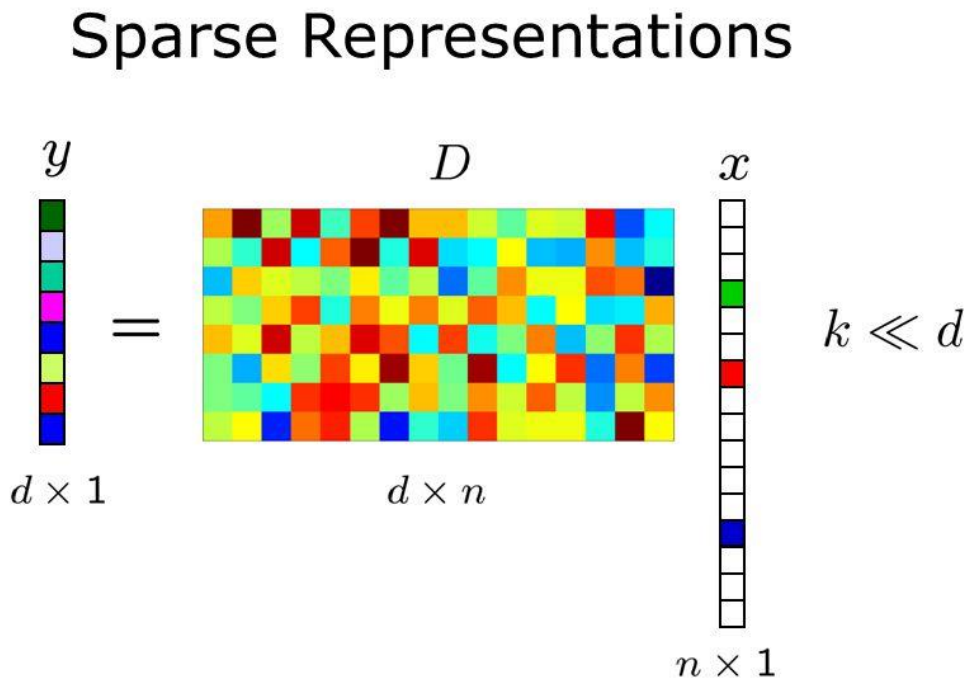
There are an exponential number of constraints for each input.

Objectifs

1. Montrer les problèmes d'optimisation discrète qui apparaissent en apprentissage machine

   B. Les problèmes de sélection : Par exemple choisir des vecteurs dans un dictionnaire de manière parcimonieuse. Sélectionner des poids d'un réseau de neurones, Sélection de caractéristiques, ...

Sélectionner une composition de vecteurs



Applications *du jour*:

Compressive sensing,
inpainting, denoising, deblurring, ...

Problème 1 (Régression aux moindres carrés).

$$x^\star = \min_{x \in \mathbb{R}^m} \frac{1}{2}\|Dx - y\|_2^2$$

La régularisation de Tikhonov

$$\min_{x \in \mathbb{R}^m} \frac{1}{2} \|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{x}\|_2^2.$$

Parcimonie : y est une composition d'un nombre réduit de sous espaces

Le but des problèmes d'optimisation qui suivent est donc de trouver K parmi ces sous-espaces. K coefficients non nuls. On veut que K soit le plus petit possible.

Problème 3 (ℓ_0 Exacte).

$$\min_{x \in \mathbb{R}^m} \|\mathbf{x}\|_0 \; t.q. \; \mathbf{y} = \mathbf{D}\mathbf{x}$$

Pseudo norme l_0 = Elle dénombre simplement son nombre de coefficients non nuls

Parcimonie : y est une composition de sous espaces

Le but des problèmes d'optimisation qui suivent est donc de trouver K parmi ces sous-espaces.

Problème 3 (ℓ_0 Assoupli)

$$\min_{x \in \mathbb{R}^m} \|x\|_0 \; t.q. \; \|Dx - y\|_2 \leqslant \tau_L$$

Ce problème
est NP-difficile

Pseudo norme l_0 = Elle dénombre simplement son nombre de coefficients non nuls

Le problème tel qu'il est traité en CV ou ML :

Problème 4 (*Lasso*).

$$\mathcal{R}_{Lasso}(\mathbf{D}, \mathbf{y}, \lambda) : \mathbf{x}^\star \triangleq \underset{\mathbf{x}\in\mathbb{R}^m}{\arg\min} \frac{1}{2}\|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

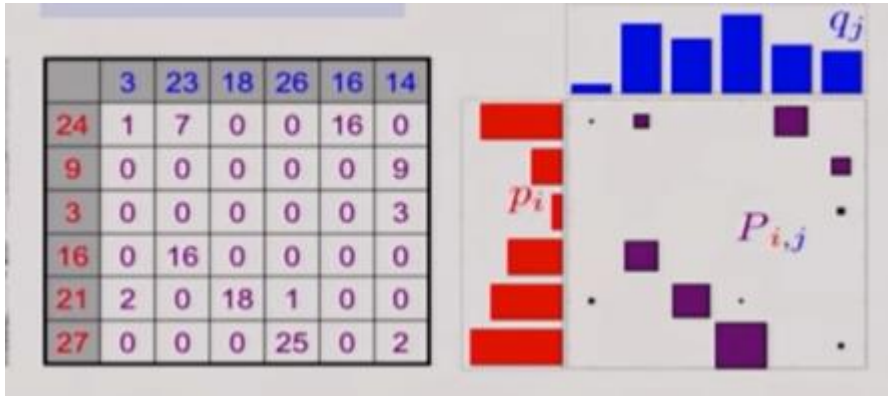
Objectifs

1. Montrer les problèmes d'optimisation discrète qui apparaissent en apprentissage machine

    C. Les problèmes d'affectation : Par exemple le problème du transport optimal, le problème du graph matching.

Le problème de Kantorovitch (KP)

$X = \{x_i\}_{i=1}^{|X|}$ and $Y = \{y_j\}_{j=1}^{|Y|}$ are two sets with x_i and $y_j \in \mathbb{R}^d$. Associated with these two sets, we denote by $Pr(x_i) = a_i$ and $Pr(y_j) = b_j$. $Pr(x_i)$ is the probability to observe x_i. A solution of KP is matrix $\mathbf{P} \in \mathbb{R}_+^{|X| \times |Y|}$. We denote by $\mathbf{p} \in \mathbb{R}_+^{|X| \cdot |Y|}$, a column-wise vectorized replica of $\mathbf{P}$. The KP is defined as follows :



Le problème de Kantorovitch

Problem 3. Kantorovitch Problem (KP)

$$\hat{\mathbf{p}} = \underset{\mathbf{p}}{\operatorname{argmin}} \sum_{x_i \in X} \sum_{y_j \in Y} c(x_i, y_j) \cdot \mathbf{p}_{i,j} \tag{5a}$$

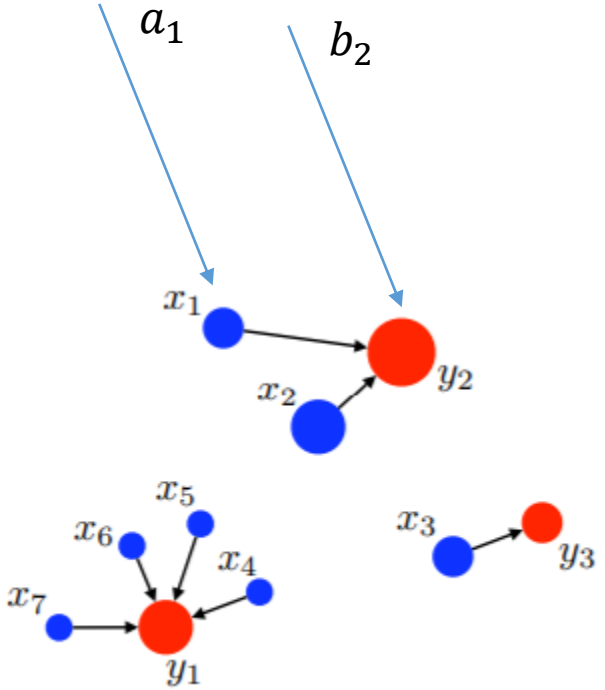
$$\text{subject to} \quad \mathbf{p} \in \mathbb{R}_+^{|X| \cdot |Y|} \tag{5b}$$

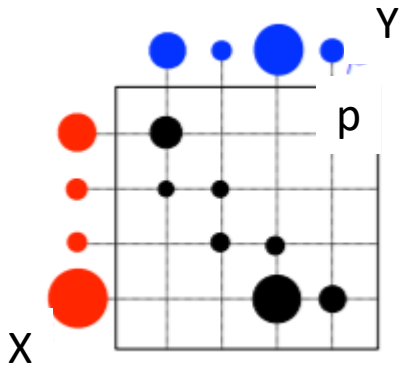
$$\sum_{a_i \in X} \mathbf{p}_{i,j} = b_j \quad \forall b_j \in Y \tag{5c}$$

$$\sum_{b_j \in Y} \mathbf{p}_{i,j} = a_i \quad \forall a_i \in X \tag{5d}$$

$$\sum_{a_i \in X} a_i = \sum_{b_j \in Y} b_j \tag{5e}$$

There exists a solution to KP only if $\sum_{a_i \in X} a_i = \sum_{b_j \in Y} b_j$.





Problème Linéaire : Résolution en O(n^3)

Distance de Wasserstein

- Valeur de la fonction objectif du problème de Kantorovitch
- Distance entre 2 distributions X et Y.

Heuristique Rapide et différentiable

- L’algorithme de Sinkhorn-Knopp
- Near linear time

Objectifs

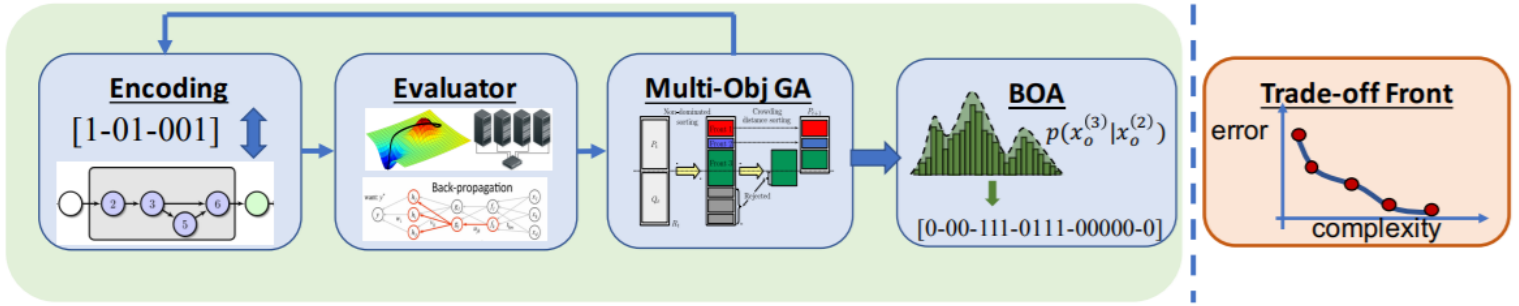
1. Montrer les problèmes d'optimisation discrète qui apparaissent en apprentissage machine

.

D. Neural architecture search (NAS)

Neural architecture search (NAS)

1. **Search space**: The NAS search space defines a set of operations (e.g. convolution, fully-connected, pooling) and how operations can be connected to form valid network architectures. The design of search space usually involves human expertise, as well as unavoidably human biases.
2. **Search algorithm**: A NAS search algorithm samples a population of network architecture candidates. It receives the child model performance metrics as rewards (e.g. high accuracy, low latency) and optimizes to generate high-performance architecture candidates.
3. **Evaluation strategy**: We need to measure, estimate, or predict the performance of a large number of proposed child models in order to obtain feedback for the search algorithm to learn. The process of candidate evaluation could be very expensive and many new methods have been proposed to save time or computation resources.



Objectifs

- Montrer les problèmes d'apprentissage qui apparaissent dans les méthodes d'optimisation discrète
  - Apprendre pour améliorer les données d’entrées pour résoudre plus simplement le problème
  - Apprendre à bien résoudre

Objectifs

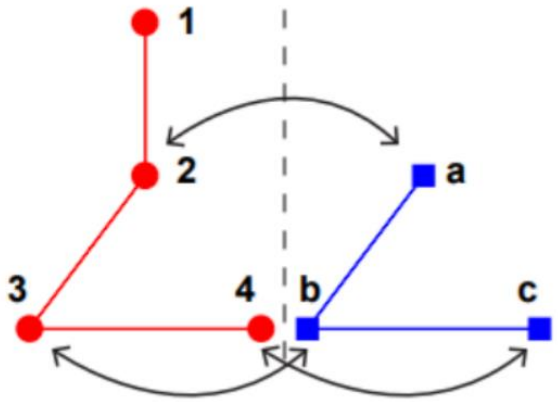
- Montrer les problèmes d'apprentissage qui apparaissent dans les méthodes d'optimisation discrète
  - Apprendre pour améliorer les données d’entrées pour résoudre plus simplement le problème

Exemple des problèmes d'affectation

- Avec la bonne matrice d'affectation le problème devient trivial
- Apprendre à remplir la matrice des couts pour que le problème devienne facile à résoudre

Euros Jobs/Persons	Person 1	Person 2	Person 3
Job A	1	5	10
Job B	6	1	5
Job C	6	3	1

Exemple des problèmes d'affectation



Graphs

Assignment Matrix : (Mat(**v**))

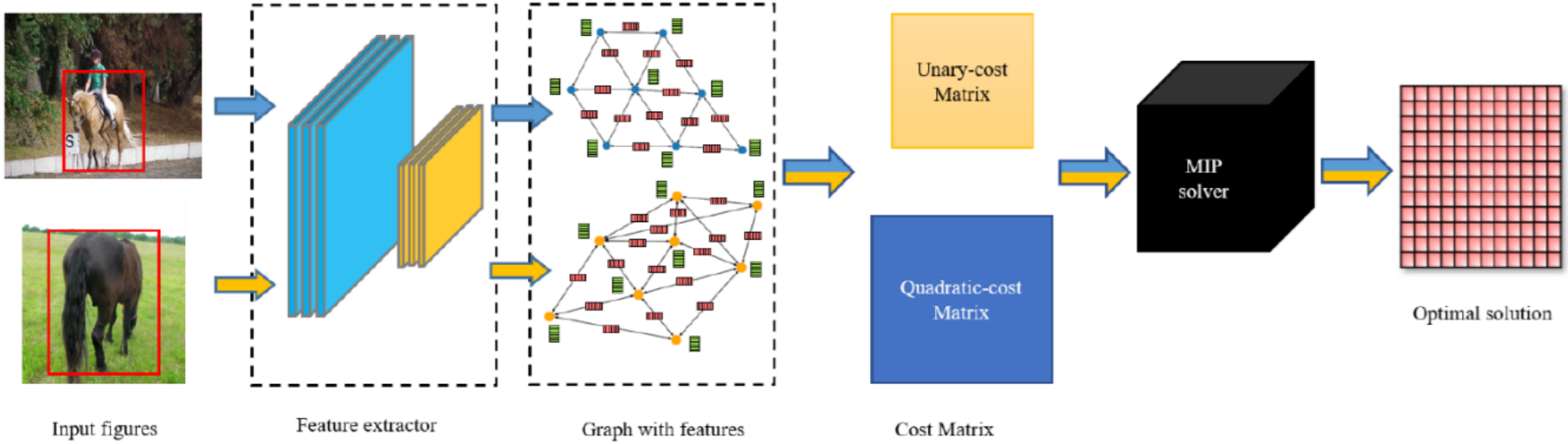
	a	b	c
1	0	0	0
2	1	0	0
3	0	1	0
4	0	0	1

Similarity Matrix (K)

	1a	2a	3a	4a	1b	2b	3b	4b	1c	2c	3c	4c
1a	1	0	0	0	0	5	0	0	0	0	0	0
2a	0	10	0	0	5	0	9	0	0	0	0	0
3a	0	0	2	0	0	9	0	3	0	0	0	0
4a	0	0	0	1	0	0	3	0	0	0	0	0
1b	0	5	0	0	1	0	0	0	0	2	0	0
2b	5	0	9	0	0	3	0	0	2	0	4	0
3b	0	9	0	3	0	0	9	0	0	4	0	10
4b	0	0	3	0	0	0	0	2	0	0	10	0
1c	0	0	0	0	0	2	0	0	4	0	0	0
2c	0	0	0	0	2	0	4	0	0	2	0	0
3c	0	0	0	0	0	4	0	10	0	0	3	0
4c	0	0	0	0	0	0	10	0	0	0	0	8

$$S(\mathbf{v}) = \mathbf{v}^T.K.\mathbf{v}$$ ➡ Objective function

Integration of combinatorial solver into a deep learning model



https://arxiv.org/abs/2108.00394 : Attention communication commerciale

Definition 2. Loss function L

$$L : \Gamma \times \Gamma$$

$$(L \circ GMS \circ NN)(G_1, G_2, \mathbf{v}^{gt}) = L(\mathbf{v}^{gt}, GMS(NN(G_1, G_2, \theta)))$$

$$\hat{\mathbf{v}} = GMS(NN(G_1, G_2, \theta))$$

L is a composition of functions

Γ : the set of all possible matchings

Definition 1. Neural Network NN

$$NN : \mathbb{G} \times \mathbb{G} \times \Theta \rightarrow [\mathbb{R}^N, \mathbb{R}^n]$$

$$NN(G_1, G_2, \theta) \mapsto [\mathbf{sv} \in \mathbb{R}^{|V_1|\cdot|V_2|}, \mathbf{se} \in \mathbb{R}^{|E_1|\cdot|E_2|}]$$

- What NN does ?
  1. Extraction of local features from the raw inputs
  2. Computation of similarities (dot product)

- Graph Matching Solver (GMS):
  - An algorithm that solves the graph matching problem

$$GMS : [\mathbf{sv}, \mathbf{se}] \mapsto \hat{\mathbf{v}}$$

Definition 3. Learning graph matching problem with a neural network

$$\theta^* = arg\min_{\theta} \sum_{((G_1,G_2),\mathbf{v}^{gt})\in TrS} L(\mathbf{v}^{gt}, \underbrace{GMS(NN(G_1,G_2,\theta))}_{\Phi(\theta;G_1,G_2)})$$

Definition 4. Learning problem formulated as a gradient-based problem

$$\text{Find } \theta^* \text{ such that } \sum_{((G_1,G_2),\mathbf{v}^{gt})\in TrS} \frac{\partial L(\mathbf{v}^{gt}, GMS(NN(G_1,G_2,\theta^*)))}{\partial \theta^*} = 0$$

Can be solved by a Gradient Descent method (Adam, RMSprop, …)

$$\theta^{(t+1)} = \theta^{(t)} - lr. \sum_{((G_1,G_2),\mathbf{v}^{gt})\in TrS} \frac{\partial L(\mathbf{v}^{gt}, GMS(NN(G_1,G_2,\theta)))}{\partial \theta}$$

Definition 5. Gradient of the loss

$$\frac{\partial L(\theta)}{\partial \theta} = \frac{\partial L(\mathbf{v})}{\partial \mathbf{v}} \cdot \frac{\partial GMS(\mathbf{sv}, \mathbf{se})}{\partial [\mathbf{sv}, \mathbf{se}]} \cdot \frac{\partial NN(\theta)}{\partial \theta}$$

- The chain rule of derivatives
- Back-propagation

Problems : Informative Gradient Problem

- The gradient of the loss L exists almost everywhere.
- The gradient of the loss L is a constant zero.

Problem 3. Informative gradient problem

$$\frac{\partial L(\theta)}{\partial \theta} \sim 0$$

- Because : $\frac{\partial GMS(\mathbf{sv}, \mathbf{se})}{\partial [\mathbf{sv}, \mathbf{se}]} \sim \mathbf{0}$

- Unhelpful for optimization by gradient-descent methods.

Proposal : Integration of our model into a deep learning architecture.

- Gradient of GMS : $\frac{\partial GMS(\mathbf{sv}, \mathbf{se})}{\partial[\mathbf{sv}, \mathbf{se}]}$

- To compute the gradient we need to **perturbate** the input of GMS according to the **variations of the loss**.

$$\mathbf{sv}_\lambda = \hat{\mathbf{sv}} + \lambda \frac{\partial L(\mathbf{v})}{\partial \mathbf{v}}(\hat{\mathbf{v}}) \quad \Big| \quad \mathbf{se}_\lambda = \hat{\mathbf{se}} + \lambda \frac{\partial L(\mathbf{v})}{\partial \mathbf{e}}(\hat{\mathbf{v}}) = \hat{\mathbf{se}}$$

- A **perturbated solution** is obtained by solving GMS with the new input:

$$\hat{\mathbf{v}}_\lambda, \hat{\mathbf{e}}_\lambda = GMS(\mathbf{v}, \mathbf{e}, \mathbf{sv}_\lambda, \mathbf{se}_\lambda)$$

- The hyper-parameter $\lambda > 0$ controls the trade-off between “informativeness" of the gradient and “faithfulness" to the original function L.

Forward pass

Algorithm 2: Forward algorithm of the graph matching layer.

Data: $\hat{\mathbf{sv}}$ and $\hat{\mathbf{se}}$: node-to-node and edge-to-edge similarities. Features provided by the neural network (NN)

Result: $\hat{\mathbf{v}}, \hat{\mathbf{e}}$: A matching solution

```
// Run the graph matching solver
```

1 $\hat{\mathbf{v}}, \hat{\mathbf{e}} := GMS(\hat{\mathbf{sv}}, \hat{\mathbf{se}})$

```
// Save matching variables and similarities as needed for
   backward algorithm
```

2 SAVE $\hat{\mathbf{v}}, \hat{\mathbf{e}}, \hat{\mathbf{sv}}$ and $\hat{\mathbf{se}}$

3 **return** $\hat{\mathbf{v}}, \hat{\mathbf{e}}$

- Backward pass

Algorithm 3: Backward algorithm of the graph matching layer.

Data: $\frac{\partial L(\mathbf{v})}{\partial \mathbf{v}}(\hat{\mathbf{v}})$: Gradient of the loss with respect to $\mathbf{v}$ at the point $\mathbf{v} = \hat{\mathbf{v}}$.

Data: λ : Parameter of fidelity to the original loss L.

Result: $\frac{\partial GMS(\mathbf{sv},\mathbf{se})}{\partial[\mathbf{sv},\mathbf{se}]}$: Gradient of the graph matching layer

// Load the matching variables and similarities

1 LOAD $\hat{\mathbf{v}}, \hat{\mathbf{e}}, \hat{\mathbf{sv}}$ and $\hat{\mathbf{se}}$

// Calculate the modified similarities

2 $\mathbf{sv}_\lambda = \hat{\mathbf{sv}} + \lambda \frac{\partial L(\mathbf{v})}{\partial \mathbf{v}}(\hat{\mathbf{v}})$

// Run the graph matching solver

3 $\hat{\mathbf{v}}_\lambda, \hat{\mathbf{e}}_\lambda := GMS(\mathbf{sv}_\lambda, \hat{\mathbf{se}})$

4 **return** $-\frac{1}{\lambda}[\hat{\mathbf{v}} - \hat{\mathbf{v}}_\lambda]$

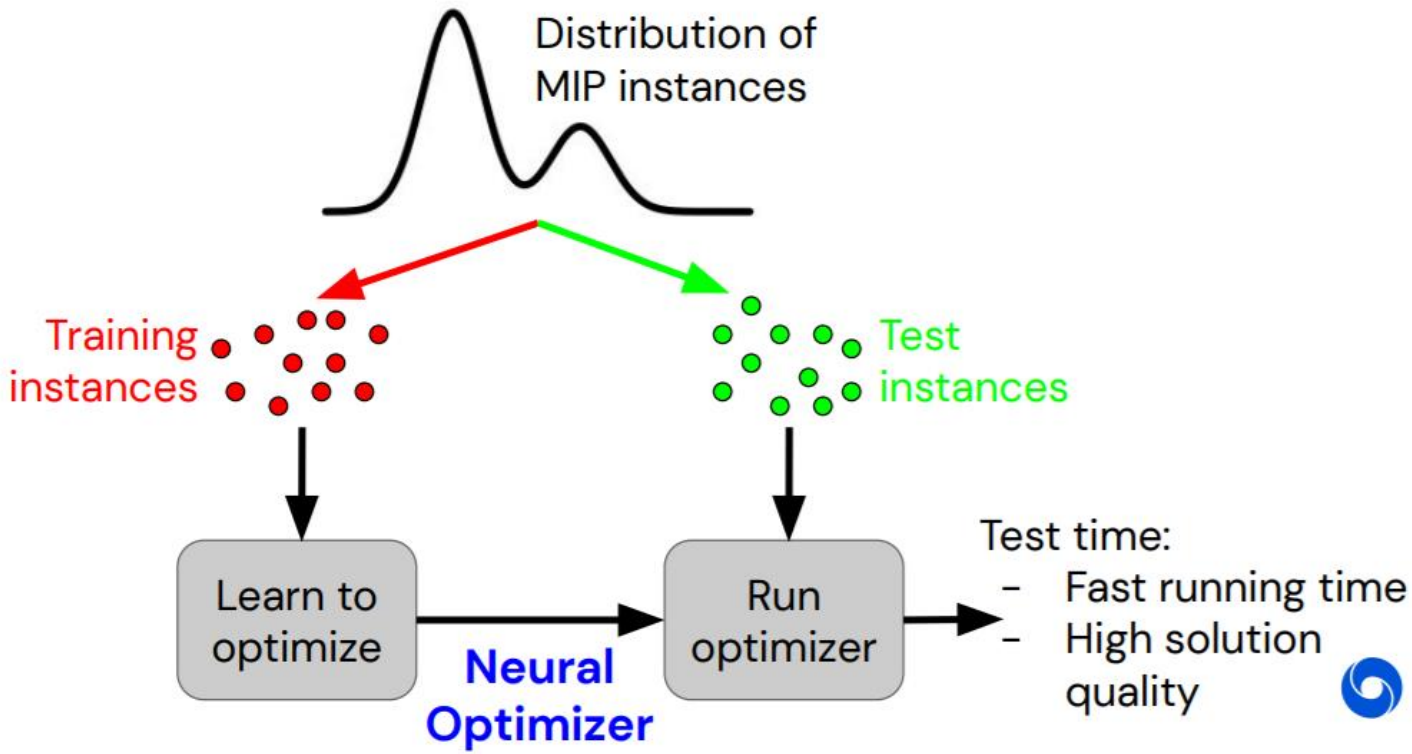
The backward algorithm is simple to implement and only runs the solver once on modified input.

Objectifs

- Montrer les problèmes d'apprentissage qui apparaissent dans les méthodes d'optimisation discrète
  - Apprendre à bien résoudre

Why Learning?

- Exploit distribution-specific structure to construct better optimizers



$$\min_x c^T x$$

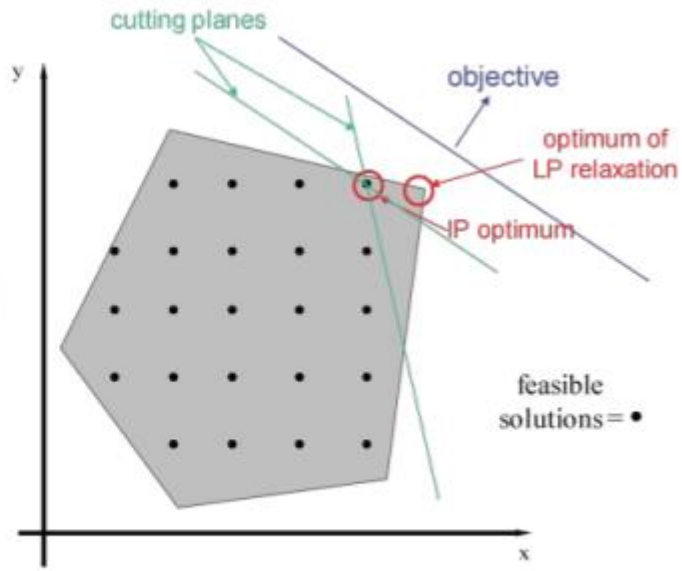
Objective function

$$s.t. \quad Ax \leq b$$

Linear constraints

$$x_i \in \mathbb{Z} \quad \forall i$$

Integrality constraint



Cutting Planes

Input MIP

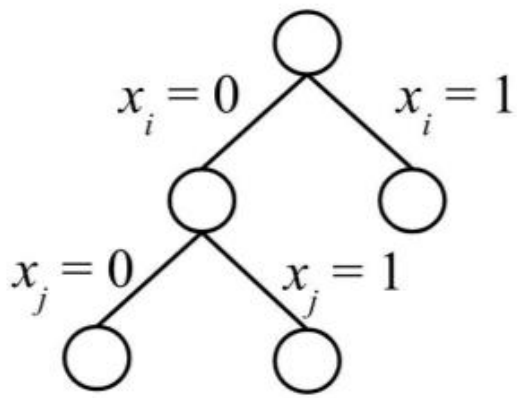
$\{A, b, c\}$

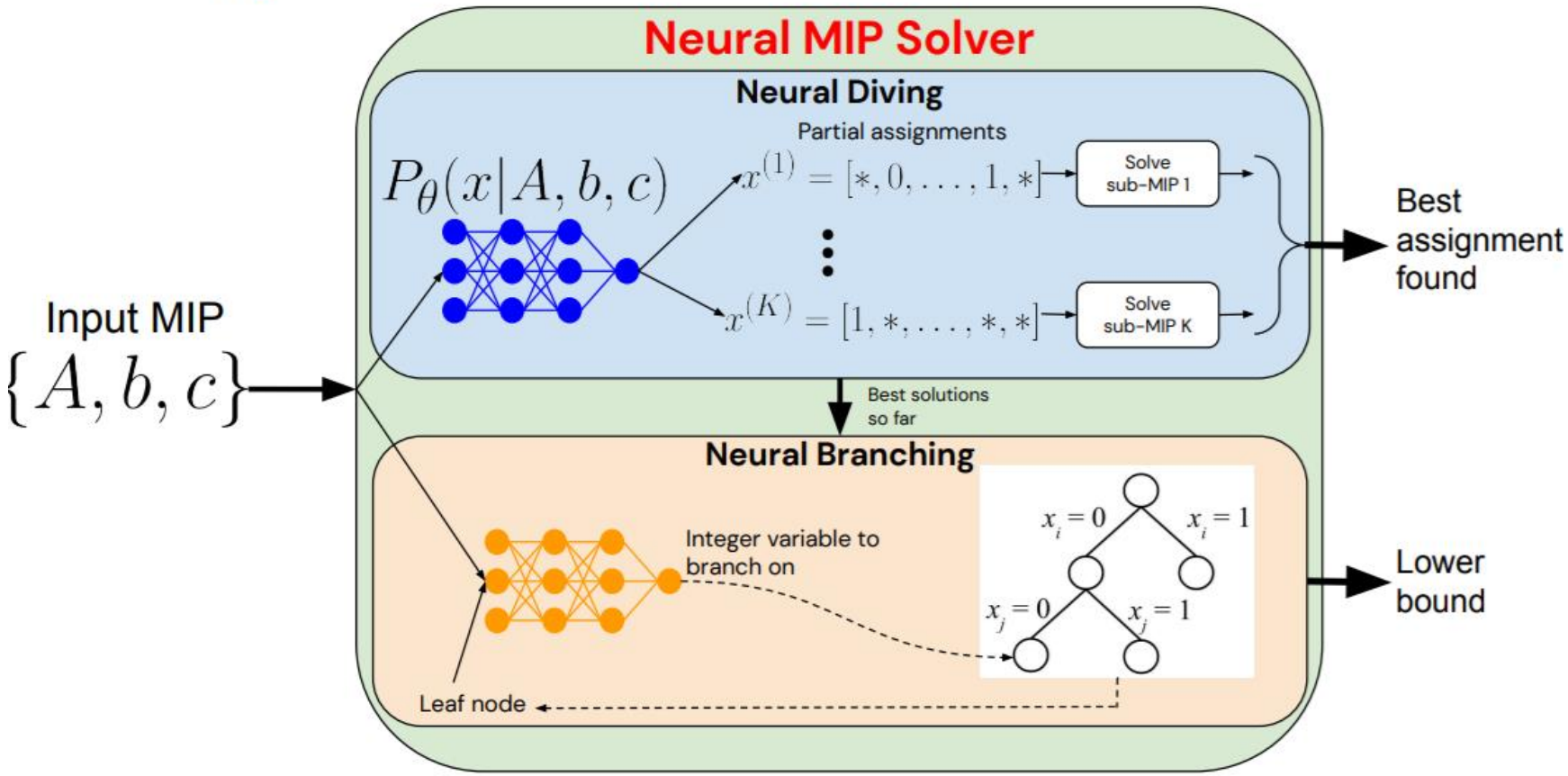
MIP Solver

Best assignment found

$x = [0, 1, \ldots, 0, 0]$

Proof of lower bound

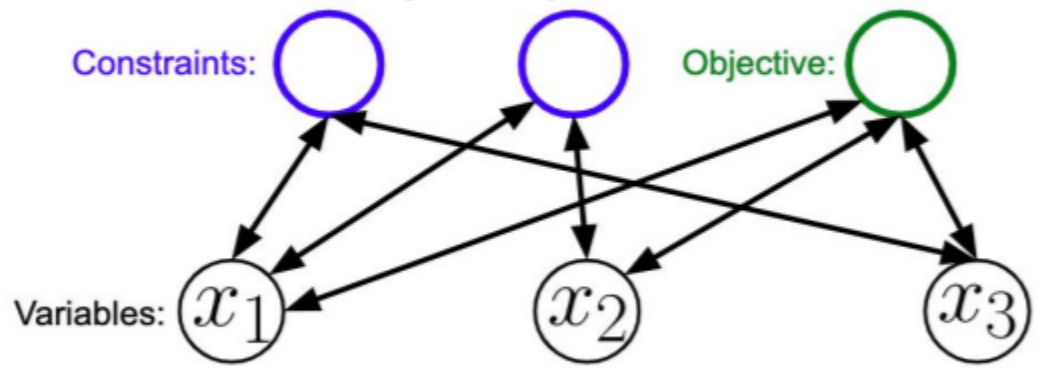



Neural MIP Solver
Neural Diving
Partial assignments
$P_\theta(x|A,b,c)$
$x^{(1)} = [*, 0, \ldots, 1, *]$
$x^{(K)} = [1, *, \ldots, *, *]$
Solve sub-MIP 1
Solve sub-MIP K
Best assignment found
Input MIP
$\{A, b, c\}$
Best solutions so far
Neural Branching
Integer variable to branch on
$x_i = 0$
$x_i = 1$
$x_j = 0$
$x_j = 1$
Leaf node
Lower bound

Mixed Integer Program

$$\begin{aligned} & \min_{x} d_1 x_1 + d_2 x_2 + d_3 x_3 \\ s.t. \quad & A_{11} x_1 + A_{13} x_{13} \leq b_1 \\ & A_{21} x_1 + A_{22} x_2 \leq b_2 \\ & x \in \mathbb{Z} \end{aligned}$$

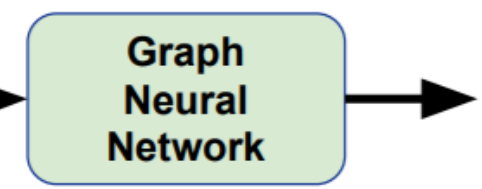
Graph representation

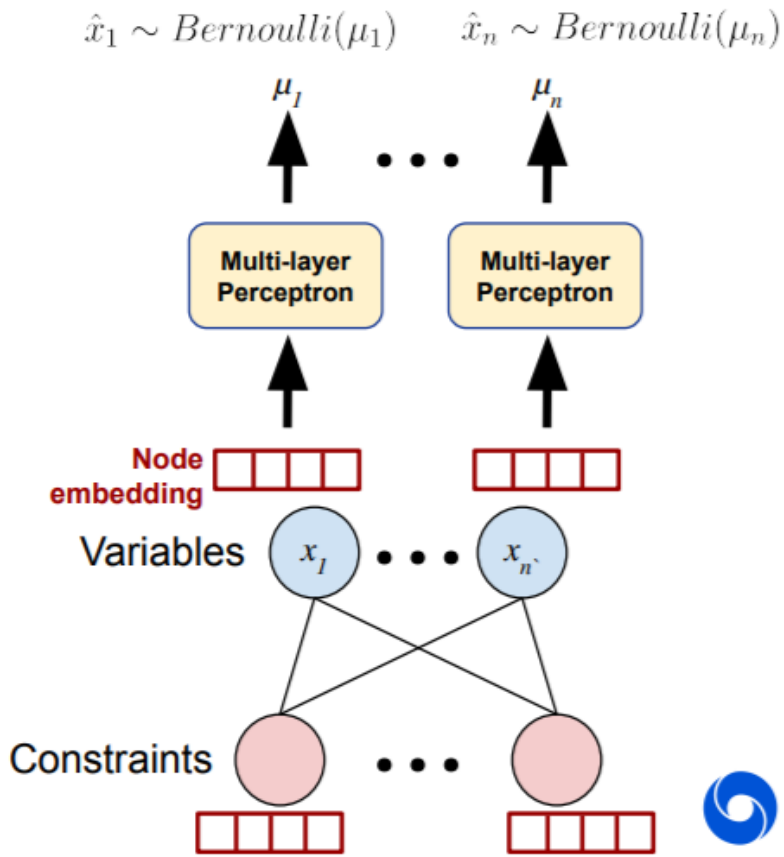


Train the model on (MIP, assignment) pairs generated using an existing solver on a training set of MIPs.

Input MIP

$\{A, b, c\}$





Achieves best ever objective value on three of the open MIPs!

MIP Name	New < previous best objective value (lower = better)	Previous best solver
milo-v12-6-r1-75-1	**1153756.398** < 1153880	CPLEX, Dec 2019
neos-1420790	**3121.29** < 3121.42	CPLEX, Dec 2019
xmas10-2	**-497** < -495	Gurobi 9.0, Feb 2020

Objectifs

4. Donner des pistes de collaborations et des réflexions

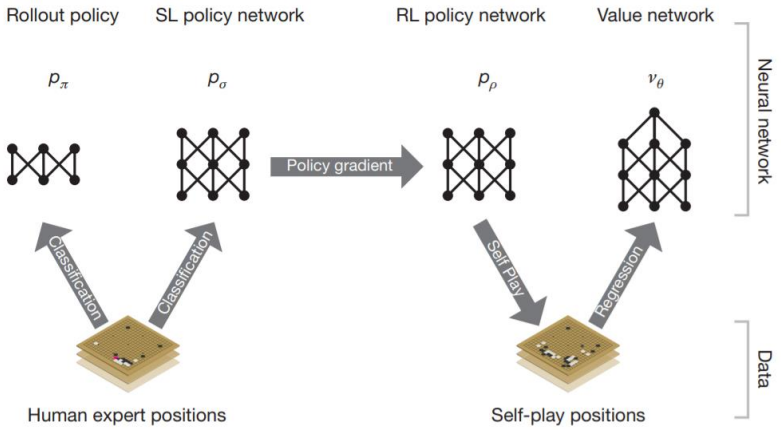
Des pistes pour la suite

- Des collaborations existantes :
  - Thèse en cours : C. Lenté, H. Cardot, N. Monmarché
  - Vincent, Romain :
    - Apprentissage de voisinages dans des recherches locales

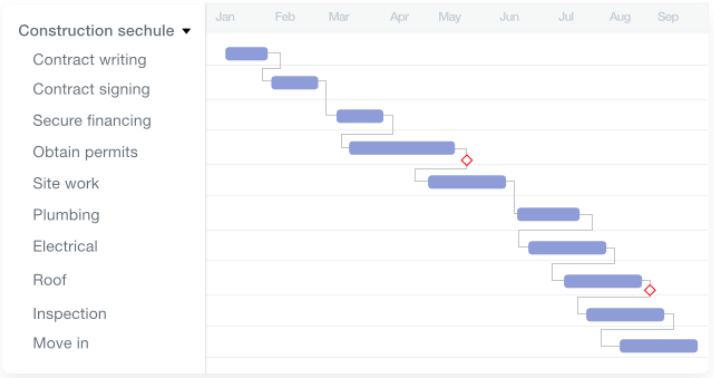
- Des pistes de collaborations
  - Julien Mille : Problème de sélections de paramètres dans les réseaux de neurones (en cours)
  - Hugo Raguet : Inférence modèle probabiliste : Graph Cut ? Est-ce encore en cours?
  - Moncef Hidane : Problème de Kantorovitch non équilibré pour le débruitage d'image
  - Vincent, Romain, Ronan, Hugo : Projet ANR déposé
    - Apprentissage de voisinages dans des recherches locales
    - Apprentissage de schéma de branchage pour des problèmes de permutation

Des pistes pour la suite et des réflexions

- Donner des pistes de collaborations et des réflexions
  - Voir **une instance d'un problème d'ordonnancement** comme une **image**
    - RFAI : Fortes compétences en Image
    - Représenter les ordonnancements comme des images de diagrammes de Gantt
      - Le diagramme de Gantt contient toutes les informations du problème
    - Appliquer les techniques issues de la vision par ordinateur (CNN, …)
    - Ça a marché pour beaucoup de problèmes : Jeu de Go, Atari Games, Puissance 4, …



Un peu comme la Thèse de Nic



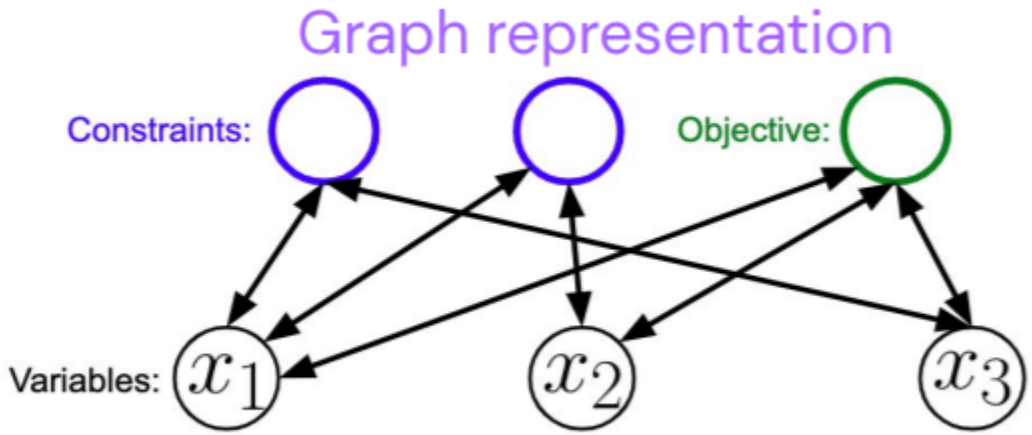
Idée appliquée à la classification de graphes par Frédéric Rayar

Des pistes pour la suite et des réflexions

- Donner des pistes de collaborations et des réflexions
  - Voir une **instance d'un problème d'ordonnancement** comme un **graphe**
    - RFAI : Fortes compétences en Graphe & apprentissage : ANR CodeGNN
    - Représenter le problème comme un graphe
    - Appliquer nos méthodes de réseaux de neurones à base de graphes.

Mixed Integer Program

$$\begin{aligned} & \min_x \; d_1 x_1 + d_2 x_2 + d_3 x_3 \\ s.t. \; & A_{11} x_1 + A_{13} x_{13} \leq b_1 \\ & A_{21} x_1 + A_{22} x_2 \leq b_2 \\ & x \in \mathbb{Z} \end{aligned}$$



Des pistes pour la suite et des réflexions

- Neural architecture search (NAS), AutoML
  - ROOT : Forte compétences en modélisation et méthodes de résolution
  - Algo génétique, multi-critères, MILP, …

Merci de votre attention

- Avez-vous des questions ?

Liens utiles et sources de la présentation

- http://helper.ipam.ucla.edu/publications/dlc2021/dlc2021_17001.pdf
- https://arxiv.org/pdf/2105.02343.pdf
- https://arxiv.org/pdf/2103.03206.pdf
- https://arxiv.org/pdf/2105.15183.pdf
- https://www.ipam.ucla.edu/programs/workshops/deep-learning-and-combinatorial-optimization/
- https://arxiv.org/pdf/1803.00567.pdf

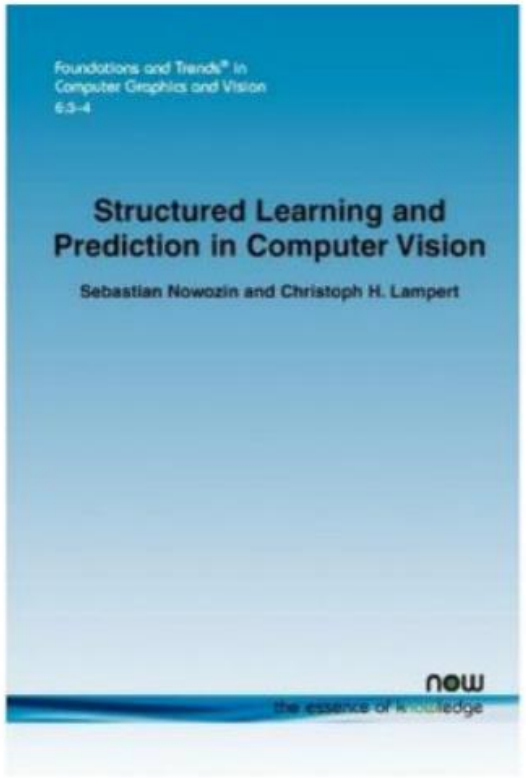
Books

Foundations and Trends in
Computer Graphics and Vision

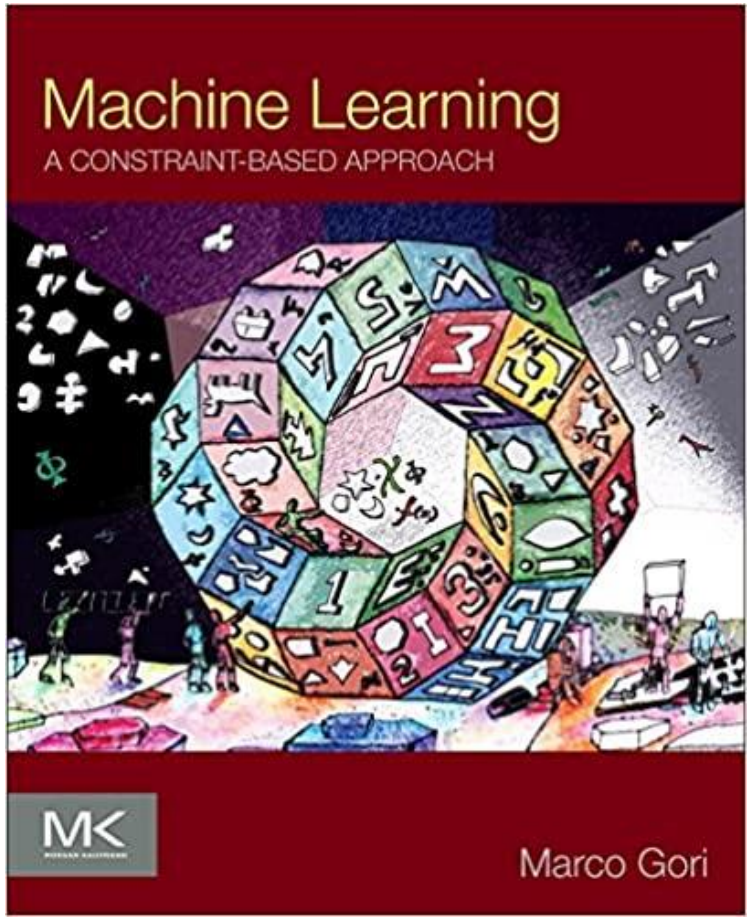
now publisher

`http://www.nowpublishers.com/`

Available as PDF on
`http://www.ist.ac.at/~chl/`



Books


Machine Learning
A CONSTRAINT-BASED APPROACH
MK
Marco Gori

Source : Parcimonie :

- Thèse: Antoine BONNEFOY
- Élimination dynamique : accélération des algorithmes d’optimisation convexe pour les régressions parcimonieuses