

Optimisation

Bruno Vallet

Equipe ACTE - LASTIG - IGN

[https://www.umr-lastig.fr/bruno-vallet/
bruno.vallet@ign.fr](https://www.umr-lastig.fr/bruno-vallet/bruno.vallet@ign.fr)

Master PPMD

Introduction parcours Géo imagerie

Cours de photogrammétrie:

La chaîne de traitement d'image

Données brutes
(images, laser)

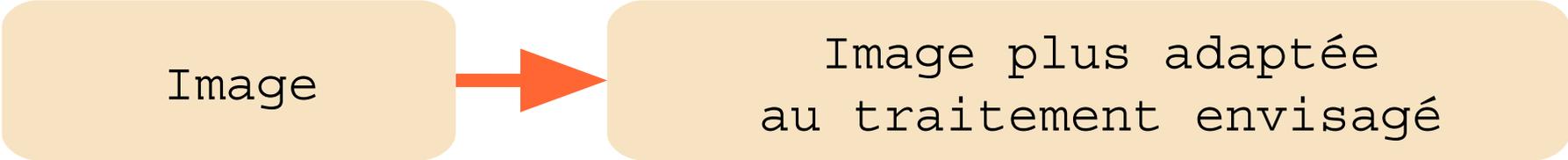


Données utilisables

Souvent composée de plusieurs maillons élémentaires

Filtrage avancé:

Image



```
graph LR; A[Image] --> B[Image plus adaptée au traitement envisagé]
```

Image plus adaptée
au traitement envisagé

Image



```
graph LR; A[Image] --> B[Opérateur différentiel]
```

Opérateur différentiel

Extraction de points d'intérêt:

Image



Points d'intérêt
 $\{\mathbf{x}_i = (l, c, d_1, d_2, \dots)_i \in \mathbb{R}^n\}$

- Utilise Filtrage

Appariement:

$\{\mathbf{x}_i\}\{\mathbf{y}_j\}$



Appariements
 $\{(\mathbf{x}_i, \mathbf{y}_j)\}$

Extraction de contours:

Image



Points de contours
 $\{\mathbf{x}_i = (l, c)_i \in \mathbb{R}^2\}$

- Utilise Filtrage

Extraction de structure:

$\{\mathbf{x}_i \in \mathbb{R}^n\}$



Primitives
 $\{\mathcal{P}_i\}$

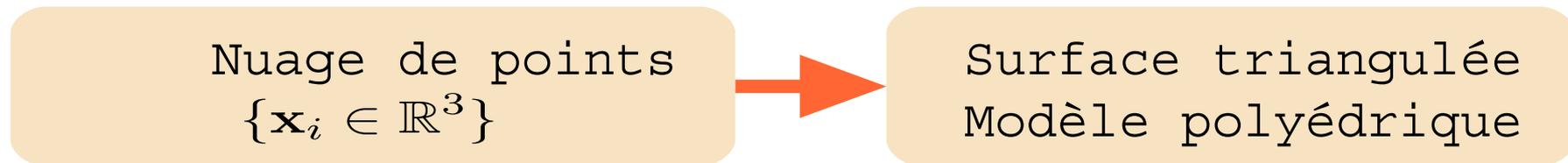
- Peut utiliser Extraction de contours, Appariement dense

Appariement dense:



- Peut utiliser Filtrage, Extraction de contours, Points d'intérêt, Appariement

Reconstruction 3D:



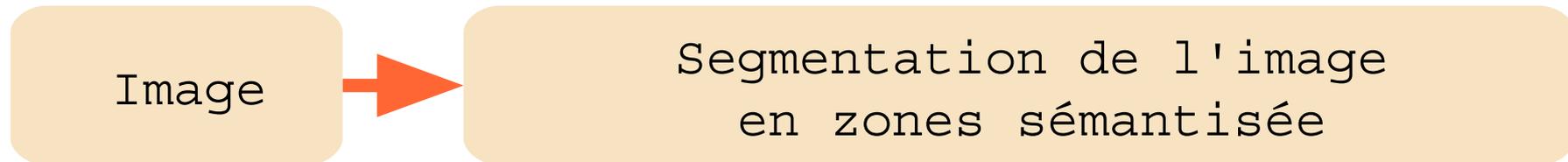
- Peut utiliser Appariement, Segmentation, Extraction de contours et de primitives

Indexation:



- Peut utiliser **Filtage, Extraction de points d'intérêt et de contours**

Classification:



- Peut utiliser **Segmentation**

Segmentation:

Image I



$$\{S_j \subset S \mid \bigcup S_j = I \text{ et } S_i \cap S_j = \emptyset\}$$

- Peut utiliser Filtrage, Extraction de contours

Détection de changement:

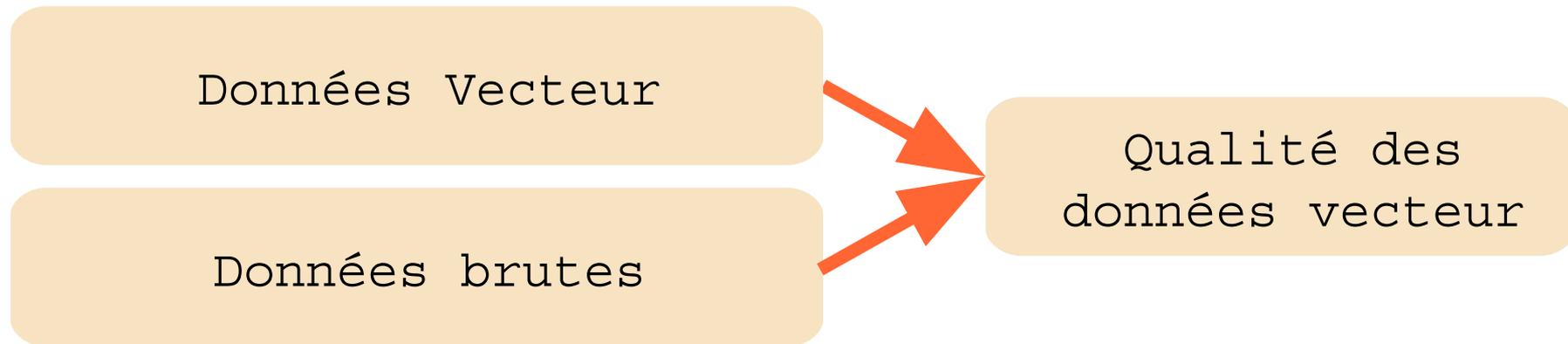
Données brutes ou
vecteur à une date T1

Données brutes ou
à une date T2

Changements
entre T1 et T2

- Peut utiliser Filtrage, Extraction de contours, Classification

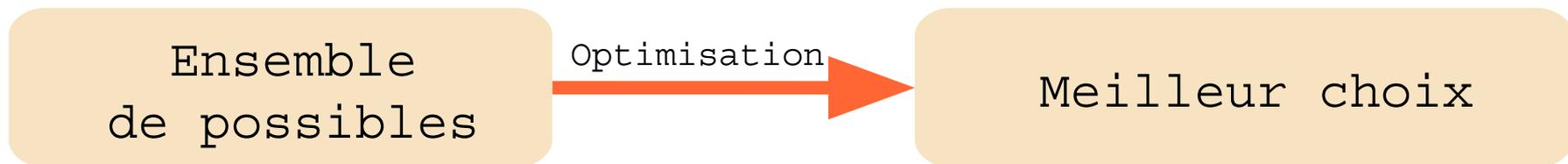
Qualification:



- Peut utiliser Filtrage, Extraction de contours, Classification

Introduction
Cours Optimisation

Le problème de l'optimisation:



Ensemble de possibles:

Espace de recherches \mathcal{R} = objets d'intérêt/choix possibles de l'optimisation

Exemples:

- Optimisation continue: $\mathcal{R} = \mathbb{R}^n$
- Objets paramétrés à \mathbb{R} degrés de liberté
- Optimisation discrète
- Optimisation mixte
- Dimension variable : sont aussi inconnus

Critères:

On se donne un score/énergie $E: \mathcal{R} \rightarrow \mathbb{R}$ à minimiser.

Le meilleur choix est celui qui minimise

Ex:

- Attache aux données: distance à des mesures/observations/données externes
- Régularité: caractéristique propre du choix
- Formulations Bayésiennes (maximisation de la probabilité du choix)

Avantages :

On se donne des objectifs explicites plutôt qu'un façon de faire (heuristique)

- Robuste si l'énergie est bien définie
- Mesure explicite de la qualité du choix
- Possibilité de comparer des choix donc des algorithmes
- Possibilités théoriques de donner des garanties sur les algos et leurs solutions sous certaines conditions

Plan du cours:

- Introduction
- I Optimisation continue
- II Optimisation sous contrainte
- III Optimisation discrète
- Conclusion

I - Optimisation continue

Ensemble de possibles:

Espace de recherches $\mathcal{R} = \mathbb{R}^n$

Deux types d'approches:

- Directes
- Itératives

Approche directe:

- Possible seulement si on peut mettre le score sous la forme :

$$E(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2 \quad \mathbf{x}, \mathbf{b} \in \mathbb{R}^n \quad M \in \mathbb{R}^n \times \mathbb{R}^n$$

- Ce critère est convexe donc minimum quand son gradient est nul :

$$\nabla E(\mathbf{x}) = 2A^t(\mathbf{Ax} - \mathbf{b}) = 0$$

- On a alors une solution explicite :

$$\mathbf{x} = (A^t A)^{-1} A^t \mathbf{b} = A^\# \mathbf{b}$$

Unicité:

- Si $A^t A$ n'est pas inversible, cela veut dire que le minimum n'est pas unique, il faut rajouter un critère.
- On choisit souvent la solution de plus petite norme :

$$E(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|^2 + \epsilon\|\mathbf{x}\|^2$$

$$\mathbf{x} = (A^t A + \epsilon Id)^{-1} A^t \mathbf{b}$$

Matrices creuses:

- Si n est trop grand, la matrice à inverser peut ne pas tenir en mémoire.
- Pourtant ces matrices sont souvent très creuses (nombreux termes nuls)
- La plupart des outils d'optimisation/d'algèbre linéaire proposent un format de matrice creuse.
- Représentations internes très optimisées complexes
- Création simple : on donne des listes de triplets (i, j, val) pour les coefficients non nuls.
- Comme l'inverse est souvent pleine, des algorithmes calculent directement $\mathbf{x} = A^{-1}\mathbf{b}$ sans calculer explicitement A^{-1}

Gradient conjugué:

- Une autre façon de faire est de minimiser itérativement le long des dimensions en suivant le gradient
- On s'assure d'éviter les redondances en projetant le gradient dans l'espace orthogonal aux directions déjà optimisées.
- Théoriquement on atteint la solution exacte en n itérations.
- En pratique on a une très bonne approximation en quelques itérations
- On a besoin de ne calculer que le gradient :

$$A^t(A\mathbf{x} - \mathbf{b})$$

Cas général :

- On suppose uniquement que l'objectif est :
 - minoré (donc qu'il existe au moins un minimum)
 - suffisamment lisse (sinon cf optimisation discrète)
- Pas de forme close, on utilise des méthodes itératives
- Caractéristiques principales :
 - Robustesse à la non-convexité (existence de minima locaux) et à l'initialisation
 - Vitesse de convergence
 - Besoin d'accès aux dérivées

Méthode de Newton:

- Généralise les moindres carrés pour des énergies non quadratiques :

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [HE(\mathbf{x}_n)]^{-1} \nabla E(\mathbf{x}_n)$$

- Basé sur le développement limité :

$$E(\mathbf{x}_n + d\mathbf{x}) \approx E(\mathbf{x}_n) + \nabla E(\mathbf{x}_n) d\mathbf{x} + d\mathbf{x}^t HE(\mathbf{x}_n) d\mathbf{x}$$

- Si $E(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2$

$$\nabla E(\mathbf{x}) = A^t(\mathbf{Ax} - \mathbf{b}) \quad HE(\mathbf{x}) = A^t A$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [A^t A]^{-1} A^t (\mathbf{Ax}_n - \mathbf{b}) = A^\# \mathbf{b}$$

Méthode de Newton:

- Utiliser si :
 - Le Hessien n'est pas trop dur à calculer ET inverser
 - Le problème n'est pas trop non-linéaire
 - Sous entend une bonne régularité
 - On stabilise avec la méthode de Levenberg-Marquardt
- Problème :
 - Ce n'est pas toujours le cas

Descente de gradient :

- Si le Hessien ne peut être calculé, au moins le gradient indique vers où chercher :

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \nabla E(\mathbf{x}_n)$$



- On peut améliorer en optimisant le pas à chaque itération (double boucle)
- Arrêt quand l'énergie ne décroît plus assez :
 - Seuil sur la décroissance
 - Seuil relatif à la décroissance depuis le début

Descente de gradient: bracketing

- On part d'un intervalle $[\lambda_1, \lambda_4]$ censé contenir le minimum et d'un sous intervalle : $[\lambda_2, \lambda_3] \subset [\lambda_1, \lambda_4]$
- On calcule $E(\lambda_i)$
- On élimine le bord extérieur adjacent au bord intérieur le moins bon
- On itère en tirant un point au centre du plus grand intervalle restant

Si on a pas accès à la dérivée:

- Pas d'indication de direction où chercher
- On un choix initial
- On tire un nouveau choix dans son voisinage
- On accepte ce choix de façon probabiliste en fonction de son énergie
- Une « température » décroît avec le temps (les itérations)
- Plus la température est basse, moins facilement on accepte les nouveaux choix

II - Optimisation sous contraintes

L'espace de recherche peut être très complexe:

- Dans ce cas, il est plus simple de travailler sur un espace facile comme \mathbb{R}^n et de rajouter des contraintes sur les variables :
- (Hyper) Plan $\mathbf{x} \in \mathbb{R}^n \quad \mathbf{a} \cdot \mathbf{x} = b$
- Variété $\mathbf{x} \in \mathbb{R}^n \quad A\mathbf{x} = \mathbf{b}$
- (Hyper) Sphere $\mathbf{x} \in \mathbb{R}^n \quad \|\mathbf{x} - \mathbf{c}\|^2 = r$
- Simplexe : $\mathbf{x} \in \mathbb{R}^{+n} \quad A\mathbf{x} = \mathbf{b}$
- Segmentation : ???

Blocage de variables:

- On a parfois besoin d'imposer des valeurs à des variables, c'est à dire d'imposer :
- On écrit alors :

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_l \\ \mathbf{x}_b \end{pmatrix} \quad A = (A_l | A_b)$$

- On minimise

$$\| (A_l | A_b) \begin{pmatrix} \mathbf{x}_l \\ \mathbf{x}_b \end{pmatrix} - \mathbf{b} \|^2 = \| A_l \mathbf{x}_l + A_b \mathbf{x}_b - \mathbf{b} \|^2$$

$$\mathbf{x}_l = A_l^\# (\mathbf{b} - A_b \mathbf{x}_b)$$

Contraintes linéaires:

- On a parfois besoin d'imposer
- Solution de facilité : on minimise

$$E(\mathbf{x}) = \|B\mathbf{x} - \mathbf{c}\|^2 + \epsilon \|A\mathbf{x} - \mathbf{b}\|^2$$

$$\mathbf{x} = (B^t B + \epsilon A^t A)^{-1} (B^t \mathbf{x}_b + A^t \mathbf{b})$$

- La façon générale d'appliquer des contraintes de la forme:

$$C_i(\mathbf{x}) = 0$$

- On cherche les points selle de:

$$\mathcal{L}(\mathbf{x}, \lambda_i) = E(\mathbf{x}) + \sum_i \lambda_i C_i(\mathbf{x})$$

- On trouve ces points selles en annulant les gradients

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda_i) = \nabla_{\mathbf{x}} E(\mathbf{x}) + \sum_i \lambda_i \nabla_{\mathbf{x}} C_i(\mathbf{x}) = 0$$

$$\nabla_{\lambda_i} \mathcal{L}(\mathbf{x}, \lambda_i) = C_i(\mathbf{x}) = 0$$

Contraintes linéaires:

- Les contraintes sont de la forme:

$$\mathcal{L}(\mathbf{x}, \lambda) = \|\mathbf{Ax} - \mathbf{b}\|^2 + \lambda^t(\mathbf{Bx} - \mathbf{c})$$

- On réunit et dans un vecteur :

$$\mathcal{L}(\mathbf{x}, \lambda) = (\mathbf{x}^t \lambda^t) \begin{pmatrix} A & B^t/2 \\ B/2 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} - (\mathbf{x}^t \lambda^t) \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

- Le point selle est :

$$\begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} 2A & B^t \\ B & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

Contrainte quadratiques:

- La contrainte est de la forme: $\mathbf{x}^t C \mathbf{x} = d$

$$\mathcal{L}(\mathbf{x}, \lambda) = \|A\mathbf{x} - \mathbf{b}\|^2 + \lambda(\mathbf{x}^t C \mathbf{x} - d)$$

$$(A^t A + \lambda C)\mathbf{x} = A^t \mathbf{b}$$

- Dans le cas $\mathbf{b} = 0$ et $C = D^t D$ c'est un problème aux valeurs propres généralisé. $\mathbf{y} = D\mathbf{x}$ $\mathbf{x} = D^{-1}\mathbf{y}$

$$(D^{-1t} A^t A D^{-1} + \lambda Id)\mathbf{y} = 0$$

- Problème aux valeurs propres classiques

Contraintes d'inégalité:

- Un problème de minimisation avec contraintes d'inégalité peut souvent se ramener à minimiser :



- Sous la contrainte $Ax = b \quad x \geq 0$
- Sans la contrainte, le problème est mal posé
- La contrainte définit un polytope convexe
- Le minimum est forcément un sommet du polytope
- La solution est donnée par la méthode du simplexe
- On minimise en suivant les arêtes du polytope jusqu'à l'optimum.

III Optimisation discrète

Plus de possibilité de dériver:

- Si on a toujours une notion de voisinage, on peut utiliser le recuit simulé
- On peut utiliser la force brute (tester tous les choix possibles)
- Algorithmes génétiques
- Méthodes de graphes
- Programmation dynamique
- Généralisations du recuit simulé

Simuler l'évolution:

- Inspiré de l'évolution de Darwin
- On tire un certain nombre de choix (population)
- Ils se reproduisent en favorisant les meilleurs
- Besoin de définir un « mélange » de choix pour la « reproduction »

Mélange reproductif:

- Pour chaque variable, on tire aléatoirement la valeur d'un des deux parents
- Sauf pour un petit nombre qu'on tire au hasard
 - modélise les mutations exceptionnelles
 - permet d'explorer des possibilités hors de la population initiale
- Arrêt quand l'énergie ne décroît plus assez

Coupe optimale dans un graphe:

- Etant donné un graphe avec des poids sur ses arêtes
- Trouver la coupe de ce graphe qui minimise les poids des arêtes coupées (max flow/min cut)
- De nombreuses énergies discrètes peuvent se mettre sous cette forme
- De nombreux algorithmes ont été développés pour le résoudre.
- A adapter en fonction des caractéristiques du graphe
- Très utilisé en segmentation, reconstruction de surface, labellisation.

Généralisations :

- Normalized cuts :
 - Les graph cuts produisent souvent des coupes très asymétriques
 - On modifie l'objectif pour équilibrer les deux parties
- Nombre de partitions :
 - Pour $n > 2$, le problème est plus dur
 - Pour n fixé : alpha-expansion
 - Combiner des bipartitions (one versus all)
 - Pour n inconnu :
 - L'optimum sur n est toujours $n=1$
 - Il faut redéfinir une énergie pénalisant les faibles valeurs de n

Problèmes de labellisation:

- Optimal si on a un ordre de dépendance sur les objets à labelliser
- Pour chaque objet trouver la meilleure labellisation des objets précédents pour chaque label
- Le résultat permet de faire de même sur l'objet suivant
- Ce calcul sur le dernier objet donne l'optimum global

Problèmes de dimension variable:

- Par exemple extraction de structure : nombre d'objets à trouver inconnu
- On a plusieurs transitions possibles vers un état d'après :
 - Modification des paramètres (recuit simulé classique)
 - Naissance/mort d'un objet
 - Modification structurelle d'un objet
- RJMCMC (Rational Jump Markov Chain Monte Carlo) :
 - Permet de générer ces transitions aléatoires suivant une loi de probabilité définie par l'utilisateur
 - A $t = 0$: échantillonnage uniforme
 - A $t = \infty$: échantillonnage des minima locaux uniquement

Conclusion

- L'optimisation est un problème fondamental et omniprésent en informatique
- La façon la plus saine de poser un problème est de définir des objectifs à atteindre et d'appliquer la méthode d'optimisation adéquate
- La difficulté de l'optimisation dépend:
 - De la régularité de l'objectif
 - De la complexité de l'espace des possibles
- Il vaut souvent mieux travailler dans un espace plus simple en rajoutant des contraintes