

Vision par ordinateur: Estimation de paramètres

Sébastien Roy
Jean-Philippe Tardif

Département d'Informatique et de recherche opérationnelle
Université de Montréal



Hiver 2007



Au programme

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes



Sommaire

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes



Décomposition symétrique des valeurs propres

Définition

Symmetric eigenvalue decomposition

Pour \mathcal{A} symétrique :

$$\mathcal{A} = \mathcal{Q}_{n \times n} \Lambda \mathcal{Q}^T$$

$$\mathcal{Q} \mathcal{Q}^T = \mathbf{I}, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n),$$

- Colonnes de \mathcal{Q} sont les **vecteurs propres** orthogonaux entre eux
- λ_i sont les **valeurs propres**
- On ordonne $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$



Matrice (semi-)définie positive

Définie positive

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} > 0, \mathbf{x} \neq \mathbf{0}$$

Équivalent à toutes ses valeurs propres positives, *i.e.*

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n > 0$$

Semi-définie positive

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} \geq 0, \forall \mathbf{x}$$

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$$

- Conditions nécessaires pour effectuer certaines décompositions



Coniques : exercices

- À quoi correspond la décomposition d'une conique $\mathcal{C} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T$?
- Dans quel(s) cas \mathcal{C} est définie positive ?
- Semi-définie positive ?



Décomposition QR

Définition

Pour une matrice carrée :

$$\mathbf{A}_{n \times n} = \mathbf{Q}_{n \times n} \mathbf{R}_{n \times n}, \quad \mathbf{Q}^T \mathbf{Q} = \mathbf{I}, \quad \mathbf{R} \text{ triangulaire supérieure droite}$$

Définition

Pour une matrice rectangulaire :

$$\mathbf{A}_{n \times p} = [\mathbf{Q}_1 \quad \mathbf{Q}_2] \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Q}_1^T \mathbf{Q}_1 = \mathbf{I}, \mathbf{Q}_2^T \mathbf{Q}_2 = \mathbf{I},$$

$$\mathbf{Q}_1 \in \mathbb{R}^{n \times p}, \mathbf{Q}_2 \in \mathbb{R}^{n \times (n-p)}, \mathbf{R} \in \mathbb{R}^{n \times p}$$

Utile pour :

- Système d'équations sous-déterminé,
- Décomposition RQ de la matrice de caméra
- Auto-calibrage affine



Décomposition RQ

Définition

Pour une matrice carrée :

$$\mathbf{A}_{n \times n} = \mathbf{R}_{n \times n} \mathbf{Q}_{n \times n}, \quad \mathbf{Q}^T \mathbf{Q} = \mathbf{I}, \quad \mathbf{R} \text{ triangulaire supérieure droite}$$

- Calculable à partir d'une QR (essayez-le!)
- \mathbf{Q} est une matrice de rotation, ou presque.



Décomposition en valeur singulière : SVD

Définition

Singular Value Décomposition

$$\mathcal{A}_{m \times n} = \mathcal{U}_{m \times r} \Sigma (\mathcal{V}_{n \times r})^T$$

$$\mathcal{U}\mathcal{U}^T = \mathbf{I}, \quad \mathcal{V}\mathcal{V}^T = \mathbf{I}$$

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$$

$$\sigma_1 > \sigma_2 > \dots > \sigma_r > 0$$

- σ_i valeurs singulières
- \mathbf{u}_i colonnes de \mathcal{U} vecteurs singuliers gauches
- \mathbf{v}_i colonnes de \mathcal{V} vecteurs singuliers droits
- r est la **Rang** de \mathcal{A} (prochaine diapo)



Décomposition en valeur singulière : SVD

On peut réécrire le tout :

$$\mathcal{A} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Lien avec décomposition symétrique des valeurs propres : pour

$$\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^T$$

$$\mathcal{A}^T \mathcal{A} = \mathcal{V}\Sigma\mathcal{U}^T\mathcal{U}\Sigma\mathcal{V}^T = \mathcal{V}\mathcal{D}^2\mathcal{V}^T$$

- $\mathcal{A}^T \mathcal{A}$ est définie positive

Donc les valeurs singulières au carré de \mathcal{A} sont les valeurs propres de $\mathcal{A}^T \mathcal{A}$

La SVD est **très, très** utile en Vision !



Utilité de la SVD : Rang d'une matrice

Définition

Le maximum entre :

- Le nombre de colonnes linéairement indépendantes
- Le nombre de rangées linéairement indépendantes

Vérifiable par le nombre de valeurs singulières non-nulles



Utilité de la SVD : Pseudo-inverse

(ou inverse de Moore-Penrose)

Définition

La pseudo-inverse de $\mathcal{A}_{m \times n} = \mathcal{U}\Sigma\mathcal{V}^T$ est donnée par :

$$\mathcal{A}^\dagger = \mathcal{V}\Sigma^\dagger\mathcal{U}^T$$

Propriétés :

- $\mathcal{A}\mathcal{A}^\dagger\mathcal{A} = \mathcal{A}$
- $\mathcal{A}^\dagger\mathcal{A}\mathcal{A}^\dagger = \mathcal{A}^\dagger$
- $(\mathcal{A}\mathcal{A}^\dagger)^* = \mathcal{A}\mathcal{A}^\dagger$
- $(\mathcal{A}^\dagger\mathcal{A})^* = \mathcal{A}^\dagger\mathcal{A}$
- On peut calculer Σ^\dagger efficacement

Si le rang de \mathcal{A} est plein ($= m$), $\mathcal{A}^\dagger = (\mathcal{A}^T \mathcal{A})^{-1} \mathcal{A}^T$



Utilité de la SVD : Pseudo-inverse

Pour une matrice diagonale :

$$D_{ii} = \begin{cases} 0 & \text{pour } D_{ii} = 0 \\ D_{ii}^{-1} & \text{sinon.} \end{cases}$$

- Si les colonnes sont linéairement indépendantes :

$$\mathcal{A}^\dagger \mathcal{A} = I$$

- Si les rangées sont linéairement indépendantes :

$$\mathcal{A} \mathcal{A}^\dagger = I$$

- Si les colonnes et rangées sont linéairement indépendantes :

$$\mathcal{A}^\dagger = \mathcal{A}^{-1}$$



Utilité de la SVD : Moindre carré linéaire

Définition

Optimiser

$$\arg \min_{\mathbf{x}} \|\mathcal{A}\mathbf{x} - \mathbf{b}\|$$

- On utilise la pseudo-inverse

$$\mathcal{A}^\dagger \mathbf{b}$$

Mais attention, il y a plusieurs façon de calculer \mathcal{A}^\dagger dans ce cas :

- Formule directe
- ★ SVD



Utilité de la SVD : Système linéaire homogène

Définition

Optimiser

$$\arg \min_{\mathbf{x}} \|\mathcal{A}\mathbf{x}\| \text{ sujet à la contrainte } \|\mathbf{x}\| = 1$$

- La contrainte est nécessaire puisque le minimum serait $\mathbf{x} = 0$

Pour $\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^T$ Le vecteur singulier droit correspondant à la plus petite valeur singulière (dernière colonne de \mathcal{V})



Utilité de la SVD : Système linéaire homogène

Pourquoi ?

$$\begin{aligned} \|\mathcal{A}\mathbf{x}\| &= \|\mathcal{U}\Sigma\mathcal{V}^T\mathbf{x}\| \\ &= \|\Sigma\mathcal{V}^T\mathbf{x}\| \quad \text{puisque } \|\mathcal{U}\mathbf{z}\| = \|\mathbf{z}\| \end{aligned}$$

On renomme $\mathbf{y} = \mathcal{V}^T\mathbf{x}$, notre nouveau problème est :

$$\|\mathcal{D}\mathbf{y}\| \text{ sujet à } \|\mathcal{V}^T\mathbf{y}\| = \|\mathbf{y}\| = 1$$

et comme \mathcal{D} est diagonale avec éléments ordonnés en ordre décroissant, la solution est

$$\mathbf{y} = (0 \quad \dots \quad 0 \quad 1)^T$$

et donc notre solution pour \mathbf{x} est

$$\mathcal{V}(0 \quad \dots \quad 0 \quad 1)^T \text{ puisque } \mathcal{V}\mathcal{V}^T = I$$

i.e. la dernière colonne de \mathcal{V}



Définition

Pour $\mathcal{A}_{n \times n}$ symétrique définie positive on peut la factoriser :

$$\mathcal{A} = \mathcal{L}_{n \times n} \mathcal{L}^T$$

avec \mathcal{L} est triangulaire inférieure

- Décomposition LU symétrique

Utile pour :

- Calibration de caméra

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes

Définition

L'étude des problèmes pour lesquels nous voulons minimiser ou maximiser une fonction réelle par un choix de valeurs réelles ou entières dans un certain ensemble

- "un certain ensemble" réfère à certaines contraintes.
 - Égalités, Inégalités
 - Un élément qui satisfait aux contraintes est dit *faisable*

Minimum

Global (ce qu'on cherche idéalement)

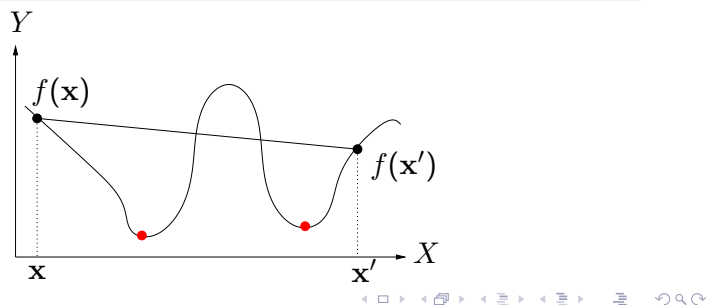
\mathbf{x}^* est dit minimum global de f si

$$f(\mathbf{x}^*) \leq f(\mathbf{x})$$

Local

\mathbf{x}^* est dit minimum local de f s'il existe ϕ tel que

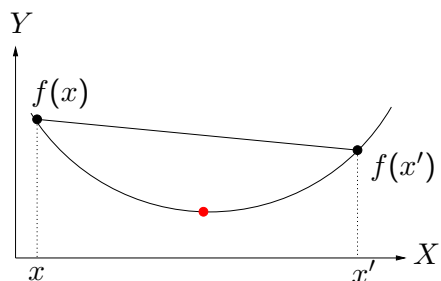
$$\forall \mathbf{x} \text{ t.q. } \|\mathbf{x} - \mathbf{x}^*\| \leq \phi, f(\mathbf{x}^*) \leq f(\mathbf{x})$$



Fonction convexe

Définition

$$f(\theta x + (1 - \theta)x') \leq \theta f(x) + (1 - \theta)f(x')$$



- Plusieurs de nos problèmes sont convexes

Un seul minimum (global) qu'on peut "facilement" trouver

Voir : Stephen Boyd and Lieven Vandenberghe, "Convex Optimization"

Sommaire

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes

Exemples de problèmes convexes

Système d'équations linéaires

$$\arg \min_{\mathbf{x}} \|\mathcal{A}\mathbf{x} - \mathbf{b}\|$$

(pseudo-inverse)

Système d'équations homogène

$$\arg \min_{\mathbf{x}} \|\mathcal{A}\mathbf{x}\| \text{ sujet à la contrainte } \|\mathbf{x}\| = 1$$

(SVD)

Système homogène et contrainte linéaire

$$\arg \min_{\mathbf{x}} \|\mathcal{A}\mathbf{x}\| \text{ sujet à la contrainte } \|\mathbf{x}\| = 1 \text{ et } \|\mathcal{C}\mathbf{x}\| = 0$$

(2 SVD)

etc...

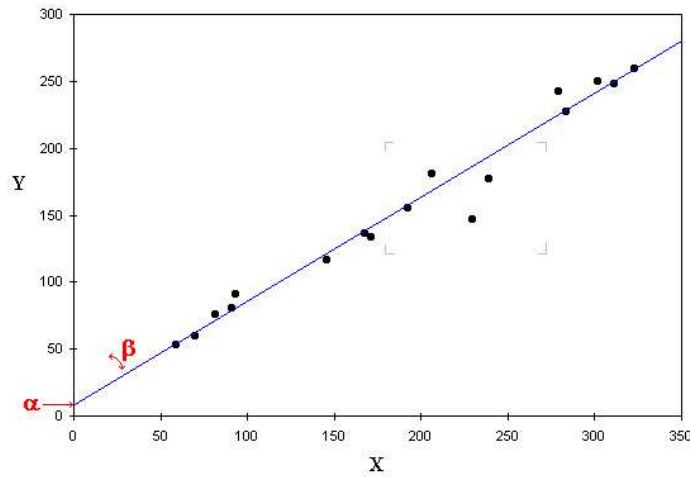


FIG.: <http://www.le.ac.uk/bl/gat/virtualfc/Stats/regression/regr1.html>

Définition

$$y_i = \alpha + \beta u_i + e_i \leftrightarrow e_i = v_i - \alpha - \beta u_i$$

Optimisation

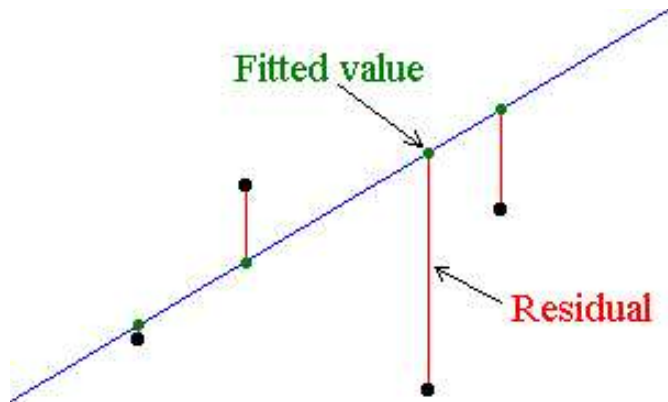
$$\arg \min_{\alpha, \beta} \sum_i e_i^2$$

$$\arg \min_{\alpha, \beta} \left\| \underbrace{\begin{bmatrix} 1 & u_1 \\ \vdots & \vdots \\ 1 & u_n \end{bmatrix}}_A \underbrace{\begin{pmatrix} \alpha \\ \beta \end{pmatrix}}_x - \underbrace{\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}}_b \right\|_2^2 \rightarrow \arg \min_x \|Ax - b\|_2^2$$

Erreur algébrique

Idéalement, on devrait minimiser la distance des points à la ligne.

- Problème non-linéaire (voir plus loin)
- Approximation linéaire : erreur algébrique
 - Distance en Y



Estimation de polynôme

$$y_i = \alpha_0 + \alpha_1 u_i^1 + \dots + \alpha_n u_i^m + e_i = e_i + \sum_i \alpha_i u_i^i$$

Optimisation

$$\arg \min_{\alpha_i} \sum_i e_i^2$$

$$\arg \min_{\alpha_i} \left\| \underbrace{\begin{bmatrix} 1 & u_1 & \dots & u_1^m \\ 1 & u_2 & \dots & u_2^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & u_n & \dots & u_n^m \end{bmatrix}}_A \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \right\|_2^2 \rightarrow \arg \min_x \|Ax - b\|_2^2$$

(Erreur algébrique)

Calcul de conique

Définition

$$(u_i \ v_i \ 1) \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = au_i^2 + du_i + bu_iv_i + cv_i^2 + ev_i + f + e_i = 0$$

Optimisation

$$\arg \min_{a,b,c,d,e,f} \sum_i e_i^2$$

$$\arg \min_{a,b,c,d,e,f} \left\| \underbrace{\begin{bmatrix} u_1^2 & u_1 & u_1 v_1 & v_1^2 & v_1 & 1 \\ u_2^2 & u_2 & u_2 v_2 & v_2^2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n^2 & u_n & u_n v_n & v_n^2 & v_n & 1 \end{bmatrix}}_{\mathcal{A}} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} \right\|_2^2 \rightarrow \arg \min_{\mathbf{x}} \|\mathcal{A}\mathbf{x}\|_2^2 \text{ pour } \|\mathbf{x}\|_2$$

Navigation icons

Système homogène

Pourquoi ne pas fixer e.g. $f = 1$? Résoudre :

$$\arg \min_{a,b,c,d,e} \left\| \begin{bmatrix} u_1^2 & u_1 & u_1 v_1 & v_1^2 & v_1 \\ u_2^2 & u_2 & u_2 v_2 & v_2^2 & v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n^2 & u_n & u_n v_n & v_n^2 & v_n \end{bmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \right\|_2^2$$

- Pourquoi pas a ou b ... erreur algébrique
- Remplacer un système homogène pour une régression est utile si \mathcal{A} est creuse (*sparse*)

Navigation icons

Calcul d'homographie : méthode linéaire algébrique

Nous avons une relation d'un plan à l'autre :

$$\mathbf{p} \propto \mathcal{H}\mathbf{q}$$

ou bien

$$\alpha \mathbf{p} = \mathcal{H}\mathbf{q}, \forall \alpha$$

On a en fait un système à trois équations :

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & -p_3 q_1 & -p_3 q_2 & -p_3 q_3 & p_2 q_1 & p_2 q_2 & p_2 q_3 \\ p_3 q_1 & p_3 q_2 & p_3 q_3 & 0 & 0 & 0 & -p_1 q_1 & -p_1 q_2 & -p_1 q_3 \\ -p_2 q_1 & -p_2 q_2 & -p_2 q_3 & p_1 q_1 & p_1 q_2 & p_1 q_3 & 0 & 0 & 0 \end{bmatrix}}_{\mathcal{A}} \mathbf{H} = 0$$

$$\mathbf{H} = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9)^T$$

(avec la troisième équation redondante.)

Avec plusieurs correspondances, ceci est un système linéaire homogène :

$$\|\mathcal{A}\mathbf{H}\|^2$$

et l'erreur est $\|\epsilon\|^2$, avec $\epsilon = \mathcal{A}\mathbf{H}$

Navigation icons

Normalisation des données

Pourquoi normaliser les données ?

Exemple :

$(1, 1), (20, 400), (100, 700), (400, 800), (700, 1000)$ etc.

Polynôme du 4e degré :

$$\mathcal{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 20 & 400 & 8000 & 160000 \\ 1 & 100 & 10000 & 1000000 & 100000000 \\ 1 & 400 & 160000 & 64000000 & 25600000000 \\ 1 & 700 & 490000 & 343000000 & 240100000000 \end{bmatrix}$$

Navigation icons

La normalisation sert à améliorer la stabilité numérique

En 2D

Il faut faire les étapes suivantes :

- Δ : moyenne des $\tilde{x}_i \rightarrow \Delta \tilde{x}_i$

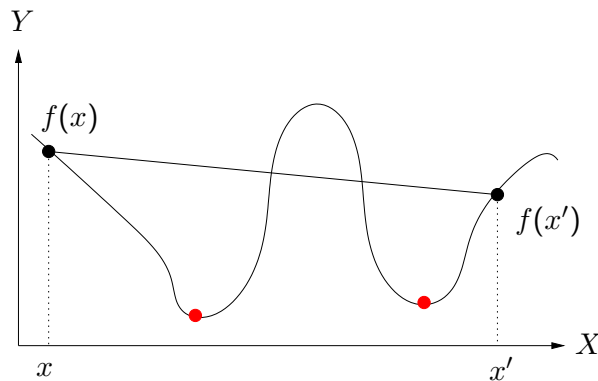
- $s = \frac{\sqrt{2}}{\text{mean} \|\Delta \tilde{x}_i\|}$

$$\begin{bmatrix} s & 0 & s\Delta_x \\ 0 & s & s\Delta_y \\ 0 & 0 & 1 \end{bmatrix}$$

- En 3D : le facteur d'échelle est $s = \frac{\sqrt{3}}{\text{mean} \|\Delta \tilde{x}_i\|}$

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes

Fonction non-convexe



- Plusieurs minimum locaux

Méthode analytique

Parfois possible mais :

- Presque jamais
- Souvent numériquement instable

Polynôme à une variable

$$f(x) = \sum_i \alpha_i x^i$$

Minimum lorsque

$$g(x) = \frac{\partial f}{\partial x} = 0$$

On cherche les racines de $g(x)$ Pour f du degré 4 : la première racine de g est

$$-\frac{a_3}{4a_4} - \frac{24a_2a_4 - 9a_3^2}{62^{2/3} a_4 \sqrt[3]{-54a_3^3 + 216a_2a_4a_3 - 432a_1a_4^2 + \sqrt{4(24a_2a_4 - 9a_3^2)^3 + (-54a_3^3 + 216a_2a_4a_3 - 432a_1a_4^2)^2}} + \frac{\sqrt[3]{-54a_3^3 + 216a_2a_4a_3 - 432a_1a_4^2 + \sqrt{4(24a_2a_4 - 9a_3^2)^3 + (-54a_3^3 + 216a_2a_4a_3 - 432a_1a_4^2)^2}}}{12\sqrt[3]{2}a_4}$$

Méthode analytique

- Souvent on préfère une solution numérique

Polynôme à une variable :
Mathematica :

```
deg = {0, 1, 2, 3, 4, 6, 7};  
Random[] Power[x, #] & /@ deg;  
eq = Plus @@ %  
f[x_] = eq;  
Roots[f[x] == 0, x]
```



Méthode itérative

- Part d'un point \mathbf{x}_0
- Converge vers un minimum local
- Il faut donc \mathbf{x}_0 le plus proche possible de x^*

Typiquement en vision :

- Solution sous-optimal, formulation convexe $\rightarrow \mathbf{x}_0$
- Raffinement non-linéaire

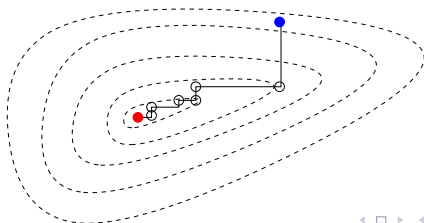


Alternation

Définition

- Fixer tous les paramètres sauf 1.
- Minimiser le problème simplifié pour cette variable.
- Recommencer pour un autre paramètre.

- Facile à implémenter
- Chaque itération est rapide si peu de paramètres (typiquement 2 paramètres)
- Convergence lente
- Fonctions non-dérivables (aucune dérivée à calculer)
- Utiliser avec modération



Gradient

Dérivée de $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$$Df(\mathbf{x})_i = \frac{\partial f(\mathbf{x})}{\partial x_i}$$

Matrice $1 \times n$

Gradient

$$\nabla f(\mathbf{x}) = Df(\mathbf{x})^T$$

Vecteur colonne $n \times 1$ des dérivés dans chaque dimension



Hessienne

Dérivée seconde de $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$$D\nabla f(\mathbf{x}) = \nabla^2 f(\mathbf{x})_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$$

Matrice symétrique $n \times n$ (puisque l'ordre de différentiation n'est pas important)

- Si définie positive \rightarrow minimum local
- Si définie négative \rightarrow maximum local
- Si valeur propre positive et négative \rightarrow col (*saddle point*)
- Sinon, aucune conclusion possible



Descente de gradient

Approximation du premier ordre

$$f(\mathbf{x} + \mathbf{v}) \approx \hat{f}(\mathbf{x} + \mathbf{v}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{v}$$

$$\Delta_{step} = \arg \min_v \hat{f}(\mathbf{x} + \mathbf{v})$$

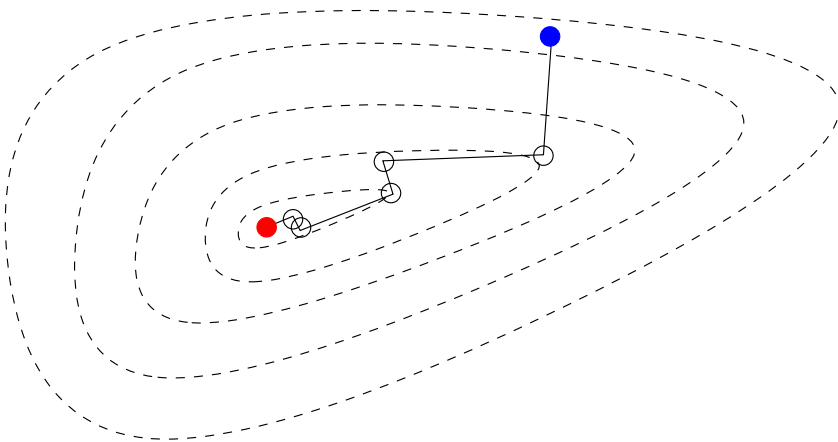
$$\Delta_{step} = -t \nabla f(\mathbf{x})$$

$$\text{donc : } \mathbf{x} := \mathbf{x} + t \Delta_{step}$$

- $\nabla f(\mathbf{x})^T \mathbf{v}$ dérivée directionnelle de f dans la direction de \mathbf{v}
- Descente si négative
- t peut être = 1 ou calculé par une méthode de recherche linéaire (*line search*)
- Choix du système de coordonnées est important



Descente de gradient



Newton

À l'origine : une méthode pour trouver les racines d'une fonction.

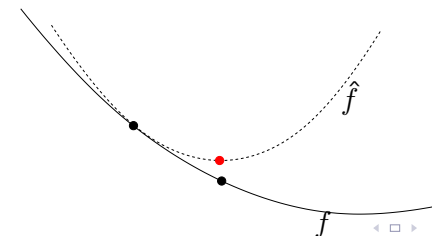
Définition

- Minimise un approximation locale du second ordre
- Invariant à un changement de coordonnées affine

$$\hat{f}(\mathbf{x} + \mathbf{v}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{v} + \frac{1}{2} \mathbf{v}^T \nabla^2 f(\mathbf{x}) \mathbf{v}$$

$$\Delta_{step} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

$$\Delta_{step} = \arg \min_v \hat{f}(x + v)$$



Il y a des méthodes plus sophistiquées

- Conjugate gradient
- Levenberg-Marquardt (descente de gradient et Gauss-Newton combinée)
- Stabilité
- Rapidité de convergence
- Implémentation spécifique lorsque l'Hessienne est creuse par bloc
- ...
- Ajouter des contraintes aux paramètres
 - Exemple typique : paramètres d'une matrice de rotation $\mathcal{R}^T \mathcal{R} = I$

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes

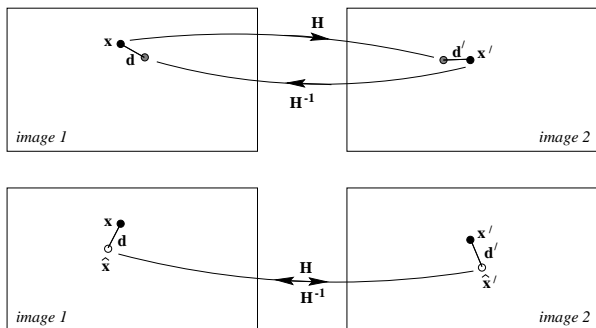
Retour aux homographies

Solution plus correcte :

- Erreur géométrique
- Erreur de reprojection symétrique

Aucun point à l' ∞

Idéalement, on aimerait trouver une homographie en minimisant une erreur ayant un sens géométrique plus clair.



On voudrait minimiser la distance entre les points originaux et les points transformés.

Erreur géométrique

Pour n points, on veut minimiser :

$$\arg \min_{\mathcal{H}} \sum_i dist(\mathbf{p}_i, \mathcal{H}^{-1} \mathbf{q}_i)^2 + dist(\mathcal{H} \mathbf{p}_i, \mathbf{q}_i)^2$$

où la fonction d est une fonction de distance géométrique des points

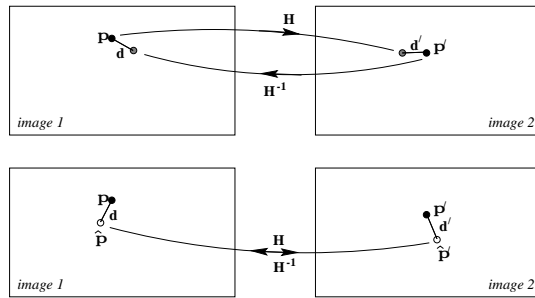
$$dist(\mathbf{p}, \mathbf{q}) = \left\| \frac{(p_1, p_2)}{p_3} - \frac{(q_1, q_2)}{q_3} \right\|$$

Au long, par exemple

$$dist(\mathbf{p}, \mathcal{H} \mathbf{q})^2 = \left(\frac{p_1}{p_3} - \frac{h_1 q_1 + h_2 q_2 + h_3 q_3}{h_7 q_1 + h_8 q_2 + h_9 q_3} \right)^2 + \left(\frac{p_2}{p_3} - \frac{h_4 q_1 + h_5 q_2 + h_6 q_3}{h_7 q_1 + h_8 q_2 + h_9 q_3} \right)^2$$

ce qui n'est pas linéaire !

Erreur de reprojection symétrique



En plus d'estimer l'homographie, on cherche les points corrigés $\mathbf{p}_i \rightarrow \hat{\mathbf{p}}_i, \mathbf{q}_i \rightarrow \hat{\mathbf{q}}_i$ qui vérifie parfaitement \mathcal{H} Pour n points, on cherche :

$$\arg \min_{\mathcal{H}, \hat{\mathbf{p}}_i, \hat{\mathbf{q}}_i} \sum_i^n dist(\mathbf{p}_i, \hat{\mathbf{p}}_i)^2 + dist(\hat{\mathbf{q}}_i, \mathbf{q}_i)^2, \quad \text{sujet à } \hat{\mathbf{q}}_i = \mathcal{H}\hat{\mathbf{p}}_i$$

Il faut donc minimiser $4n+9$ paramètres. On note que le nombre d'inconnues est très grand, ce qui est un gros inconvénient. Par contre, ceci est un problème **sparse** et on peut résoudre assez efficacement.

Note : Dans le cas des homographies, cette méthode n'améliore pas significativement. Elle est surtout donnée à titre d'exemple, puisqu'elle est utile dans certaines circonstances (matrice fondamentale, calcul de conique...).

Implémentation

- Mathematica
FindMinimum : Newton, Levenberg-Marquardt, Conjugate Gradient (pas la même chose que *Minimize*)
- Matlab
lsqnonlin
- C
Levenberg-Marquardt : *levmar*
- Fortran avec wrapper C
Levenberg-Marquardt : MINPAK

Sommaire

- 1 Décompositions utiles
- 2 Optimisation ou estimation de paramètres
 - Introduction
 - Problèmes convexes
 - Problèmes non-convexes
- 3 Homographie
- 4 Méthodes robustes

Données erronées

Estimateur robuste

Estimateur qui n'est pas affecté par des données qui ne respectent pas l'hypothèse concernant le modèle de bruit

- Sujet complexe

Données valides

Hypothèse :

- Loi Gaussienne (Normale), $\mu = 0$

Données erronées

Hypothèse :

- Loi uniforme

Méthode naïve

Enlever les données

- Utiliser toutes les données
- Calculer la distance des points
- Enlever les plus mauvais
- Itérer

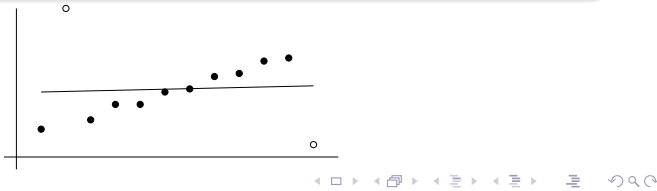
Peut fonctionner si :

- Très petit ratio de mauvaises données
- Elles sont bornées

Re-pondération

- Pondérer un point par l'inverse de son erreur
- Itérer jusqu'à convergence

Ne fonctionne pas très bien, fortement déconseillée



Vote dans l'espace des solutions

- Discrétisation de l'espace solution
- Pour chaque point, voter pour les solutions possibles
- Choisir la solution qui a le plus de vote
- Déterminer les bons points et recalculer la solution

Exemple : Transformée de Hough



Vote dans l'espace des solutions

Quelques difficultés :

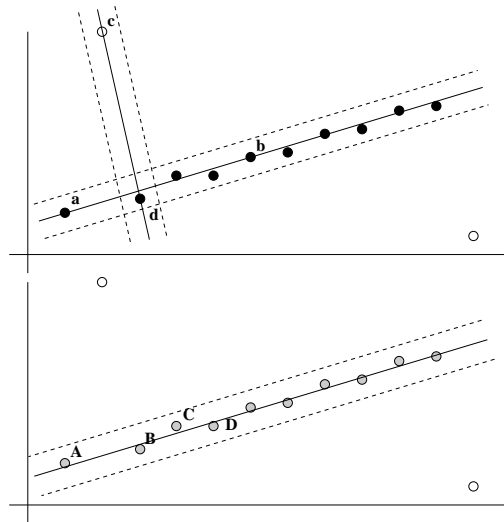
- Comment discrétiser ?
Exponentiel en le nombre de paramètres
- Le vote n'est pas toujours simple/possible
e.g. Homographie
- Surtout utile pour la segmentation
e.g. plusieurs lignes
- # de paramètres doit rester petit
3 et moins en général

RANSAC

RANdom SAMpling And Consensus : idée générale

- Trouver un sous-ensemble (minimal) de points au hasard
- Calculer les paramètres pour les points de cet ensemble
- Calculer l'erreurs pour chaque point
- Répéter pour trouver la solution qui "explique" le plus de points

- Bon choix :
ligne \overline{ab}
- Mauvais choix :
ligne \overline{cd}



- Résultat final



Entrée :

- D : # de données
- d : # minimal de points
- t : distance maximal pour les points non-aberrants
- T : # minimal de non-aberrants
- N : # maximal d'itérations

Pour $i < N$:

- Trouver un ensemble E_i de d points
- Calculer le modèle (candidat) M_i avec E_i
- Calculer l'erreur de toutes les données pour M_i
- Déterminer S_i : les points valides
- Si $|S_i| > T$,
Calculer le modèle final avec les points dans S_i et terminer
- Sinon
Répéter



- Inutile d'essayer toutes les possibilités
- Hypothèse : on connaît le ratio d'aberrants w (ou une borne supérieure)

On cherche N pour une probabilité p d'avoir trouvé un ensemble sans aberrant

w probabilité qu'un point soit valide

$\epsilon = 1 - w$ soit un aberrant

w^d d'avoir d points valides

$1 - w^d$ d'avoir un mauvais ensemble

$(1 - w^d)^N$ d'avoir juste des mauvais ensembles après N essais

donc

$$(1 - w^d)^N = 1 - p$$

et

$$N = \log(1 - p) / \log(1 - w^d) = \log(1 - p) / \log(1 - (1 - \epsilon)^d)$$



- Choisir d le plus petit possible pour le calcul des paramètres
- d plus petit $\rightarrow N$ plus petit \rightarrow meilleur performance



Mise à jour de N en fonction des nouvelles solutions calculés

$$N = \infty$$

$$nb = 0$$

Tant que ($nb < N$)

Trouver une solution M_i pour un ensemble E_i

$\hat{w} = |S_i|/|D|$ est une borne pour w

Ajuster N avec \hat{w} (comme précédemment)

$$nb = nb + 1$$



Deux façons :

Méthode simple

- Re-estimation du modèle avec les données valides
- Sous-optimale

MLE (Maximum Likelihood estimator)

- fonction de coût robuste (non-convexe) :

$$E = \sum e_{rb}(dist(\mathbf{x}_i))$$
$$e_{rb}(d) = \begin{cases} d^2 & \text{pour } d^2 < t^2 \\ t^2 & \text{sinon.} \end{cases}$$

t^2 est une constante.



FIN

Certaines figures sont la courtoisie de

R. Hartley et A. Zisserman

Multiple View Geometry in Computer Vision, 1ère Édition, 2002

