Notes de cours
OPT6 : Apprentissage avancé

Adrien Pavao

Novembre 2017

1 Introduction

L'apprentissage automatique (machine learning) s'appuie sur des bases théoriques. Ce domaine évolue vite et on y trouve des théories et techniques avancées.

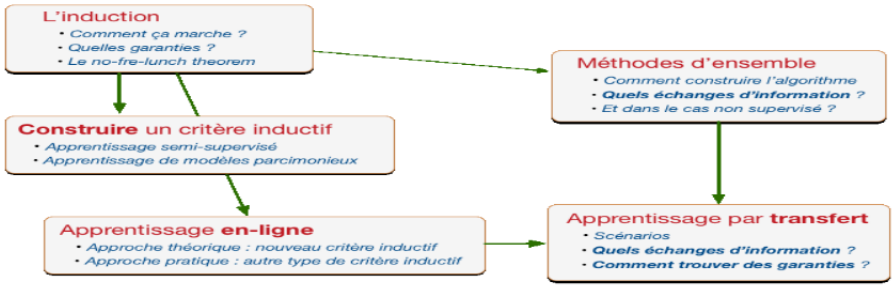

Figure 1: Les points qui seront abordés dans ce cours.

1.1 Quelques notions

- Vladimir Vapnik a posé les bases théoriques de l'apprentissage statistique[1].
- Apprentissage PAC : Probably Approximately Correct[2].
- **No-free-lunch theorem :** Il n'existe pas de méthode d'apprentissage qui soit meilleure qui les autres. Même performance que l'aléatoire sur l'ensemble des problèmes possibles. Si une méthode est efficace sur un problème elle sera mauvaise sur un autre. L'intégrale sur tous les problèmes est toujours la même, quelque soit la méthode. Idem pour les mesures (de distance par exemple). Les méthodes sont adaptées à des **classes** de problèmes. **Tous les algorithmes inductifs se valent.**

[1] https://en.wikipedia.org/wiki/Statistical_learning_theory
[2] https://en.wikipedia.org/wiki/Probably_approximately_correct_learning

- **Adversarial learning :** L'intersection entre le machine learning et la sécurité informatique.
- **Régularisation :** Un processus consistant à ajouter de l'information à un problème pour éviter le surapprentissage. Cette information prend généralement la forme d'une pénalité envers la complexité du modèle. On peut relier cette méthode au principe du **rasoir d'Occam**. D'un point de vue bayésien, l'utilisation de la régularisation revient à imposer une distribution a priori sur les paramètres du modèle.

  Une méthode généralement utilisée est de pénaliser les valeurs extrêmes des paramètres, qui correspondent souvent à un surapprentissage. Pour cela, on va utiliser une norme sur ces paramètres, que l'on va ajouter à la fonction qu'on cherche à minimiser. Les normes les plus couramment employées pour cela sont L_1 et L_2.
- Différentes normes permettent de calculer la taille totale d'un vecteur ou d'une matrice dans un espace vectoriel. Quelques exemples courants :

  Norme L_0 :
  $$||x||_0 = \sqrt[0]{\sum_i x_i^0}$$
  Norme L_1 :
  $$||x||_1 = \sum_i |x_i|$$
  Norme L_2 :
  $$||x||_2 = \sqrt{\sum_i x_i^2}$$
- **Distance de Manhattan :** La distance entre deux points lorsque les déplacements sont faits selon un réseau ou un quadrillage.

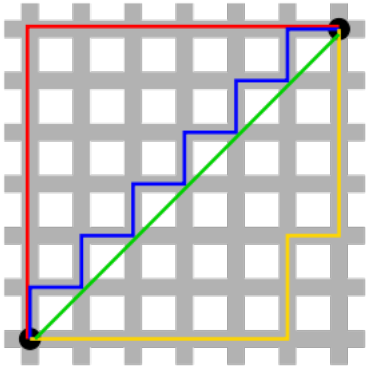

Figure 2: Distance de Manhattan (chemins rouge, jaune et bleu) contre distance euclidienne en vert.

Le nom de cette distance fait sans doute référence à l'architecture en quadrillage du quartier de Manhattan, à New York.

- Référence d'un livre intéressant : *Hofstadter - "Godel, Escher, Bach".*
- **Complexité de Kolmogorov :** Une fonction permettant de quantifier la taille du plus petit algorithme nécessaire pour engendrer un nombre ou une suite quelconque de caractères. Cette quantité peut être vue comme une évaluation d'une forme de complexité de cette suite de caractères.
- Effets de séquences : Une séquence d'information influe sur le résultat de l'induction (à préciser).
- L'algorithme d'apprentissage du perceptron converge si les données sont linéairement séparables. Cette convergence est en nombre fini d'étapes et ce nombre est indépendant du nombres d'exemples, de la distribution des exemples, et quasiment pas de la dimension de l'espace d'entrée. Il dépend de la marge de séparation des nuages de points et du diamètre de la boule (à préciser).
- **L'espace des versions :** Toutes les hypothèses correctes (qui ne font pas d'erreurs) d'après les données d'apprentissage.

2 Induction

On déduit à partir d'un échantillon.

1. Espace d'hypothèse H.
2. Critere inductif $S \times h \in H \rightarrow$ Score $\in \mathbb{R}$. Par exemple minimiser le taux d'erreur.
3. Methode d'exploration de H.

On ne dispose pas de théorie bien établie de l'induction (exemple Fibonacci).

2.1 Différents types d'apprentissage

- **Descriptif :** Non supervisé. On cherche des régularités dans les données. On ne souhaite pas extrapoler, on ne s'interesse qu'à l'échantillon de données dont on dispose. La "matière noire" de l'apprentissage.
- **Prédictif :** Supervisé. L'échantillon de données sert à apprendre une hypothèse sur les données pour prédire ensuite sur de nouvelles données. On cherche des corrélations entre les données.
- **Prescriptif :** On cherche des **causalités**. Il s'agit d'une tâche difficile.

Les frontières entre ces types d'apprentissage peuvent être floues, on peut par exemple commencer par une description des données pour les comprendre avant de faire un modèle prédictif.

2.2 Différentes méthodes d'apprentissage

Dans ce cours nous allons aborder différentes méthodes d'apprentissage.

- Méthodes d'ensemble : Combiner plusieurs hypothèse (exemple : boosting).
- Supervisé. La méthode classique d'apprentissage passif est appelé *apprentissage batch*. C'est la technique la plus utilisée aujourd'hui.
- L'induction :
  - Apprentissage semi supervisé
  - Apprentissage de modèles parcimonieux (peu de paramètres).
- Apprentissage en ligne (online learning) : Les données arrivent en cours de route.
- **Apprentissage par transfert :** Cela vise à transférer des connaissances d'une ou plusieurs tâches sources vers une ou plusieurs tâches cibles. On peut le voir comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes.

2.3 Types de biais

L'induction necessite d'être biaisé, d'avoir une preference pour certaines hypothèses, de ne pas connaitre toutes les fonctions possibles. Un biais classique et très humain est la préférence des hypothèses les plus simples (rasoir d'Ockham).

- **Biais de représentation (déclaratif) :** On ne peut représenter qu'un petit nombre de fonctions possibles. Le langage dans lequel on exprime le problème ne me permet pas de tout représenter.
- **Biais de recherche (procédural) :** La procédure de recherche avantage certaines hypothèses. Par exemple on reste proche de notre point initial de recherche dans l'espace des fonctions possibles.

Souvent un peu des deux.

2.4 Risques

On distingue deux types de risques :

- **Risque réel** (ce que l'on veut minimiser). La performance à venir si j'utilise l'hypothèse h. Cela prend la forme d'une espérance de coût d'usage de h. Le risque réel (loss function) :

$$R(h) = \int_X^Y l(h(x), y)p(x, y)\mathrm{d}x\mathrm{d}y$$

La meilleure hypothèse est $k^* = ArgminR(h)$ avec $h \in H$. En d'autres termes, on cherche l'hypothèse qui minimise le risque réel. l est la métrique d'évaluation et p la loi de probabilité hypothétique ayant générée la distribution des données.

- **Risque empirique**. Puisque l'on cherche la loi de probabilité, on ne la connait pas et on ne peut donc pas s'en servir pour calculer le risque, en pratique. On calcul donc le risque empirique :

$$\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i)$$

C'est donc la moyenne des distances entre les prédictions avec h et les valeurs réelles des données. $\hat{h}$ est l'hypothèse minimisant le risque empirique.

On cherche les liens entre toutes ces notions.

Pour Vladimir Vapnik

Plutôt que de minimiser le risque empirique :

- Pour une hypothèse h : $P_{S \sim P_{XY}}(\hat{R}(h) = 0 \wedge R(h) > \epsilon)$

On ne veux pas d'une hypothèse qui paraisse bonne mais qui soit en réalité mauvaise. Le principe de minimisation du risque empirique n'est sain que s'il y a des contraintes sur l'espace des hypothèses.

(formule et principe de consistance universelle ?)

3 Transduction

On ne s'interesse qu'à un point. Le nombre de données requises pour tirer une conclusion n'est plus influencé par le nombre de dimension. On parle de **transductive learning**.

- Transduction : Cas d'induction limite (une question).
- Analogie : Cas de transduction limite (un exemple).

4 Possibilités d'apprendre

4.1 Perceptron

1. On suppose que l'ensemble des hypothèses H est fini.
2. **Cas réalisable :** $\exists h \in H tq R(h) = 0$ ($\neq 0 \rightarrow$ cas non-réalisable.)
3. S est supposé tiré *i.i.d.*

4.2 Dimension Vapnik-Chervonenkis

La d_{VC} est la taille du plus grand ensemble de points tirés aléatoirement que l'on peut pulvériser, c'est-à-dire que l'on peut étiqueter n'importe comment en utilisant une hypothèse de H.

$d_{VC}(H)$ donne une idée de l'expressivité du modèle (de l'espace de réalisation).

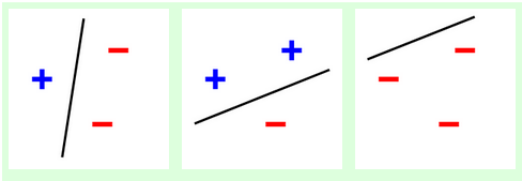

Figure 3: Pulvérisation de 3 points

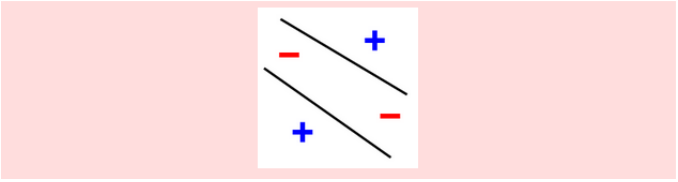

Figure 4: Lorsqu'il y a 4 points, la pulvérisation est impossible

d_{VC} de l'ensemble des droites = 3.

4.3 Apprentissage faible et fort

- **Apprenant faible :** Erreur réelle $< \frac{1}{2} - \gamma$, $\gamma > 0$. Typiquement un decision stump.
- **Apprenant fort :** On peut être infiniment exigeant. $\forall\epsilon$, $\forall\delta$, avec suffisamment d'exemples, A trouvera une hypothèse h d'erreur réelle $< \epsilon$ dans plus de $1 - \delta$ des cas (échantillons). Dans le cas réalisable.

4.4 Borne sur le risque réel

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \sqrt{\frac{8\, d_{VC}(\mathcal{H}) \log \frac{2\,e\,m}{d_{VC}(\mathcal{H})} + 8 \log \frac{4}{\delta}}{m}} \right] > 1 - \delta$$

5 Apprentissage à partir de flux de données

5.1 Flux de données

Un flux de données, ou data stream, est une séquence de mesures

$$S =< x_1, x_2, ..., x_t, ... > \quad (1)$$

$$S =< (\underbrace{x_1}_{\hat{y}_1}, y_1), (\underbrace{x_2}_{\hat{y}_2}, y_2), ..., (\underbrace{x_t}_{\hat{y}_t}, y_t), ... > \quad (2)$$

5.2 Plusieurs séquences

- Clustering
- Classification supervisée

Nécessite la définition d'une distance appropriée. On ne peut pas permuter les éléments de la séquence sans modifier le problème.
Exemple : Génomes, distances d'édition, programmation dynamique.

Tâches classiques

5.3 Prédiction (une séquence)

- ARIMA : régression linéaire
- Chaines de Markov
- HMM (Modèle de Markov Caché)
- Grammaires
- Réseaux de neurones récurrents

Données discretisées (peu de symbols) dans le cas des chaines de Markov et des HMM.

5.4 Apprentissage en ligne (online learning)

Apprentissage en cours de route.

L'apprentissage par renforcement peut être vu comme une sorte d'apprentissage en ligne.

1. Scénario :

$$\underbrace{x_1 \rightarrow \hat{y}_1 = h_1(\vec{x_1}) \leftrightarrow y1}_{h_2}$$

$$\underbrace{x_2 \rightarrow \hat{y}_2 = h_2(\vec{x_2}) \leftrightarrow y2}_{h_h}$$

2. Motivations pour ce scénario

   - Cadre Anytime : même en environnement stationnaire
   - Environnement non stationnaire.
     - Types de changements : Sur P_x : **covariate shift**.
     - Sur $P_{Y|X}$: **concept drift** (dérive de concept).
   - Données très volumineuses.
     - Adaptation de domaine (domain adaptation) : "en gros" covariate shift.
     - Apprentissage par transfert (transfer learning) : changement de tâche.

5.5 Apprentissage incrémental

- Environnement stationnaire.
- Séquence pédagogique pour apprendre un concept complexe.

Evaluation de l'apprentissage

Comment faire en environnement non stationnaire ? On ne peux plus utiliser le risque empirique. Prequential criterion. Nombre d'erreurs de prédiction commises sur un intervalle T.

Comment réaliser l'apprentissage

$$A : H \times X \times Y$$

$$(h_t, x_t, y_t) \rightarrow h_{t+1}$$

5.6 Experts : un scénario maximaliste

- La théorie de l'apprentissage en ligne (online learning theory).
- Contre toute séquence y compris les séquences adverserielles.
- Notion de "**comité d'experts**".
- Ensemble de N "experts".
- Expert : $x \rightarrow$ y.
- Aucune hypothèse sur le fonctionnement interne des experts (statique, dynamique...)

(0 → bonne réponse, 1 → erreur)

_	Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6
Exemple 1	1	0	0	1	0	1
Exemple 2	_	_	_	_	_	_

Ressemblance avec les bandits manchots. Les concepts de dilemme exploration-exploitation et de regret s'appliquent.

5.6.1 Algorithmes gloutons

Algorithme glouton déterministe

On prend à chaque étape l'expert qui a fait le moins d'erreur. Déterministe car on selectionne de gauche à droite quand il y a égalité, en partant (du suivant) du dernier que l'on a pris.

_	Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6
Exemple 1	1	[**0**]	0	1	0	1
Exemple 2	1	1	1	0	[**0**]	0
Exemple 3	1	[**0**]	0	0	1	1

Le pire cas du glouton déterministe :

_	Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6
Exemple 1	[**1**]	0	0	0	0	0
Exemple 2	0	[**1**]	0	0	0	0
Exemple 3	0	0	[**1**]	0	0	0

$$\underbrace{L}_{\text{Perte algo}} \leq N(L^*) + N - 1$$

Si l'adversaire tire profit de sa connaissance de l'algorithme, le glouton déterministe est très catastrophique. On peut éviter ça en tirant aléatoirement. Il s'agit donc du glouton aléatoire, ou glouton non-déterministe.

Algorithme glouton aléatoire

Pire cas :

$$L_{RG} \leq (lnN + 1)(L^*) + lnN$$

C'est bien mieux.

5.6.2 Algorithmes de vote

Algorithme du vote majoritaire

- **Cas réalisable :** un expert aura 0 erreur sur la séquence.
  A chaque erreur à l'instant t :

$$W_{t+1} < \frac{W_t}{2}$$

$$L_{CR} \leq [log_2 N]$$

- **Cas non réalisable :** A $t = 0$, $W_0 = N$.
  Pour chaque expert i :

$$w_i(t+1) = \begin{cases} w_i(t) \\ \beta w_i(t) \\ \beta \in [0, 1[\end{cases}$$

  Supposons qu'une erreur soit commise à l'instant t :

$$W(t+1) \leq \frac{W(t)}{2} + \beta \frac{W(t)}{2}$$

  Après m erreurs :

$$W_m \leq W_0 \frac{(1+\beta)^m}{2^m}$$

$$\text{Poids du meilleur expert} = \beta^{m^*}$$

$$m \leq \frac{log_2 N + m^* log_2(\frac{1}{\beta})}{log_2 \frac{2}{1+\beta}}$$

5.7 Approches heuristiques

Dérive de concept (continu, des ruptures rares).

Le compromis plasticité-stabilité

- **Stabilité :** Interêt à considerer un maximum de données
- **Plasticité :** Pouvoir s'adapter si le concept change, savoir oublier.

6 Apprentissage par transfert et adaptation de domaine

Utilités principales :

- Gain d'information : Si je n'ai pas assez d'information dans mon domaine cible, l'apprentissage du domaine source améliore mes performances
- Gain de temps : On n'apprend pas à nouveau sur le domaine cible

Notations usuelles :

- S : Domaine source
- T : Domaine cible
- $D_s(P_x^s)$: Distribution des exemples sources
- $D_s(P_x^T)$: Distribution des exemples cibles
- h_s : Hypothèse apprise sur la source
- h_T : Hypothèse apprise sur la cible
- S_s : Echantillon de données sources
- S_T : Echantillon de données cibles

En vrac :

- Souvent, on suppose que $|S_T| << |S_s|$.
- La **divergence de Kullback-Leibler** est une mesure de dissimilarité entre deux distributions de probabilités P et Q.
- Jeu pathologique : Plus on explore loin, plus on perd d'information.

6.1 Définition et illustration

$$\text{Source } X_s \to y_s$$

$$\text{Cible } X_T \to y_T$$

$$X \to Y$$

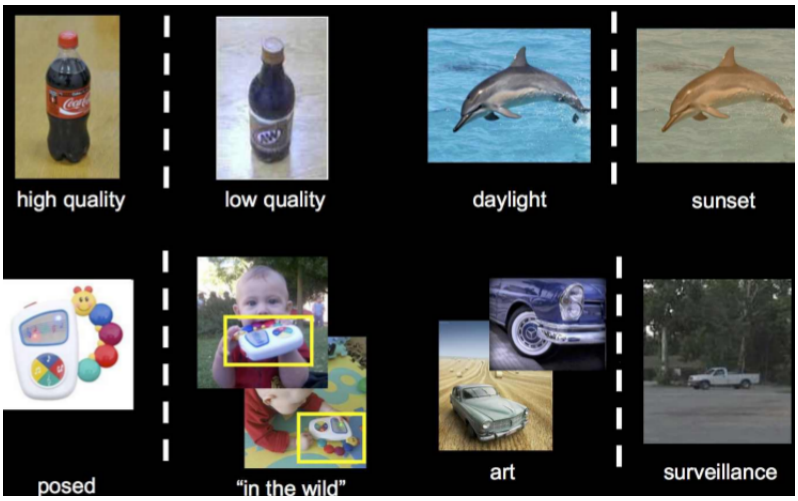

Figure 5: Exemples de transferts de domaine

Les trois grandes approches (slides 87) :

- Instance-based method : reweighting
- Features-based method : find new representation spaces
- Iterative method : adjustement

6.2 Une approche de l'apprentissage par transfert

Le problème motivant le travail. Classification précoce de série temporelles. On va apprendre des fonctions de $X_t \rightarrow \underbrace{X_s}_{\text{Toutes les mesures possibles}}$

On apprend des fonctions π_i par le boosting. A la fin :

$$H(\vec{x}_t) = signe\left\{ \sum_{i=1}^{N} \alpha_i h_s \circ \underbrace{\pi_i(\vec{x}_t)}_{\in X_s} \right\}$$

6.3 Questions

Que transfert-on généralement dans les réseaux neuronnaux profonds ?

On garde les premières couches et on ré-apprend les dernières couches. Les premières couches capturent les descripteurs pertinents pour la tâche source que l'on suppose pertinent pour la tâche cible. On espère qu'il représente les aspects généraux du problèmes, tandis que les dernières couches sont plus spécialisées.

Comment évaluer la distance entre une hypothèse source et une hypothèse cible ?

- Nombre de "projecteurs" nécessaire pour assurer le transfert entre h_s et h_t.
- Un ensemble de test et on mesure $|err(h_s) - err(h_t)|$
- Facilité de trouver des projecteurs

Quels sont les critères qui devraient jouer un rôle dans le succès d'un transfert ?

- Le degré de similarité entre tâches source et cible.
- Information cible :
  - Taille de S_t suffisante
  - Pas trop de bruit
- Il faut une "capacité" de l'espace des projecteurs limités
- La performance de h_s (sur la source)

7 Le filtrage collaboratif

7.1 Introduction

L'objectif est de selectionner les "articles" pertinents pour vous.

- 1960s : Par mots clés prédéfinis (peu performant).
- 1980s : Apprendre les mots-clés → variables latentes. Il s'agit d'une approche par le contenu.
- 1994 : Premier papier décrivant les bases du collaborative filtering. On va tirer profit des préférences émises par les autres utilisateurs (en fonction de leur ressemblance avec l'utilisateur cible).
- 2001 : Amazon, fait son coming out
- 2006 : Prix Netflix
  - $\sim$ 100 000 000 de préférences émises (notes entre 1 et 5)
  - $\sim$ 500 000 utilisateurs → chaque utilisateur avait noté 200 films en moyenne
  - $\sim$ 25 000 films

  Les données sont incomplètes. Pour mesurer la performance :

$$Perf = \frac{1}{m} \sum (\underbrace{r_{ij}}_{\text{Vraie note}} - \underbrace{\hat{r}_{ij}}_{\text{Note estimée}})^2$$

7.2 Les grandes approches

7.2.1 Approches basées sur le contenu (content-based)

Exemples de problèmes, notation de films par des utilisateurs. On a 4 clients et 5 films. Chaque utilisateur, à chaque film, donne ou non une note entre 0 et 5. On a des informations sur les films (taux de romance, taux d'action) pour définir les genres. Ce sont les descripteurs de film.

-	Alice	Bob	Fred	Marie	Romance	Action
Film 1	2	?	4	0	0.2	0.9
...	...	...	...	...	...	...

On modélise cela par un modèle linéaire.
Fonction : $utilisateur * film \to \mathbb{R}^n$ dans [0, 5].
$u(x) = u_1 * x_1 + u_2 * x_2 + u_3$. Le dernier u est le biais.
On a donc un problème d'optimisation. Pour un utilisateur i :

$$\vec{u_i^*} = Argmin \sum (R_{ij} - < \vec{u}_i, \vec{x}_j >)^2 + \underbrace{\alpha ||\vec{u}_i||^2}_{\text{Terme de régularisation}}$$

Procédé de descente du gradient (η le pas d'apprentissage) :

$$\vec{u}_i \leftarrow \vec{u}_i + \eta \sum -(R_{i,j} - < \vec{u}_i, \vec{x}_j >)\vec{u}_i$$

Remarques :

- Un modèle par utilisateur (pas de liens entre les modèles)
- Problème du choix des descripteurs

7.2.2 Approches utilisant la similarité entre utilisateurs

-	Item 1	Item 2	Item 3	Item 4
Jean	-	2	7	8
Marie	4	1	-	7
Chistian	3	8	-	4

Calcul de similarité entre les lignes
Formule de Bravais-Pearson :

$$p = \frac{Cov(\vec{R}_i, \vec{R}_j)}{\sqrt{Var(\vec{R}_i) * Var(\vec{R}_j)}} = \frac{\sum_k (R_{i,k} - \bar{R}_i)(R_{j,k} - \bar{R}_j)}{\sqrt{\sum_k (R_{i,k} - \bar{R}_i)^2} \sqrt{\sum_k (R_{j,k} - \bar{R}_j)^2}}$$

- k : nombre de plus proches voisins pris en compte
- u_v : un utilisateur voisin

$$\hat{R}_{l,j} = \bar{R}_l + \frac{\sum_{v=1}^{k} r(u_l, u_v)(R_{v,j} - \bar{R}_v)}{\sum_{v=1}^{k} |r(u_l, u_v)|}$$

7.2.3 Approches par factorisation de matrice

On aimerait appliquer la **décomposition en valeurs singulières** (SVD). Cependant, cela ne marche pas directement. Lorsque la matrice X est creuse, on ne peut pas utiliser la SVD. En 2006, Simon Funk suggère de passer outre.

Deux grandes familles de méthodes d'optimisation (slide 44) :

1. Par gradient stochastique
2. Par optimisation alternée sur les utilisateurs et les items