Recherche de contenu par comparaisons

Jill-Jênn Vie

26 septembre 2013

Le présent document est le rapport d'un stage effectué au laboratoire commun Inria-Microsoft Research [1] du 1er avril au 31 juillet 2013, sous la direction de Laurent Massoulié.

Table des matières

1 Contexte général et introduction **2**
1.1 Description du problème considéré 2
1.2 Notions-clés et paramètres de complexité 2
1.3 Stratégies considérées pour le choix des questions 4
1.4 Notre contribution 5

2 Analyse d'un algorithme glouton bayésien **5**
2.1 Cas sans erreur 5
2.2 Cas avec erreur : mises à jour bayésiennes 6
2.3 Réduction d'entropie à un pas 7

3 Résultats expérimentaux **9**
3.1 Effet des différents paramètres 9
3.2 Discussion 9

A Implémentation en Python **11**

B Bibliographie détaillée **19**

1. Campus de l'école polytechnique - Bâtiment Alan Turing 1 rue Honoré d'Estienne d'Orves 91120 Palaiseau

1 Contexte général et introduction

1.1 Description du problème considéré

La recherche de contenu par comparaisons revient à déterminer une cible t parmi un ensemble fini d'objets E à l'aide de questions posées à un oracle qui, lorsqu'on lui présente deux objets de E, indique lequel des deux est « le plus proche » de la cible t qu'il a en tête. Plus précisément, si $t, x, y \in E$:

$$Oracle_t(x, y) = \begin{cases} +1 & \text{si } x \preccurlyeq_t y \\ -1 & \text{si } x \succ_t y. \end{cases}$$

où $\preccurlyeq_\square$ est une relation permettant de comparer deux éléments de E par rapport à un troisième. Si l'on dispose d'une distance d sur les objets, mais ce ne sera pas toujours le cas, on a généralement : $x \preccurlyeq_t y \Leftrightarrow d(t, x) \leqslant d(t, y)$.

L'objectif est de déterminer la cible en sollicitant l'oracle le moins possible ; à ces fins, chaque question à poser sera déterminée en fonction des réponses obtenues précédemment (problème d'apprentissage actif, ou *active learning*) de manière à apporter une quantité significative d'information sur la cible. La connaissance du système à un instant donné peut être représentée soit par un sous-ensemble d'objets élagué au fur et à mesure jusqu'à ne contenir plus qu'un élément, soit par une distribution de probabilité sur les objets qui s'affine en fonction des réponses aux questions, jusqu'à ce qu'un critère d'arrêt soit satisfait, indiquant la fin du jeu.

```
p ← 𝕌_E, la loi uniforme sur E
Tant que max_{x∈E} p(x) est inférieur à un certain seuil
    Choisir une question à poser à l'oracle
    Poser cette question
    Mettre à jour p en fonction de la réponse de l'oracle
Retourner la cible (échec ou succès)
```

FIGURE 1 – Un squelette d'algorithme de recherche de contenu par comparaisons.

Le présent rapport se déroule comme suit : après une présentation des différentes méthodes existantes dans la section 1, la section 2 détaille les stratégies du papier principal d'étude et justifie le choix de la question suivante ainsi que la formule de mise à jour de la loi de probabilité sur les objets après chaque réponse de l'oracle. La section 3 présente les différents résultats de nos recherches.

1.2 Notions-clés et paramètres de complexité

Plusieurs modèles existent pour affronter ce problème, et leurs algorithmes respectifs sont évalués selon plusieurs aspects et s'expriment principalement en fonc-

tion du nombre n d'éléments de E :

— la complexité en temps et espace pour déterminer la question à poser ;
— la complexité en nombre de questions pour déterminer la cible ;
— le pourcentage de réussite même si l'oracle est autorisé à se tromper.

Entropie. Une borne minimale immédiate sur le nombre de questions à poser est l'entropie de la distribution de probabilité μ sur les objets :

$$H(\mu) = \sum_{x \in \text{supp}(\mu)} -\mu(x) \log_2 \mu(x).$$

Pour faire deviner un nombre à k bits d'entropie, c'est-à-dire compris entre 0 et $2^k - 1$, il est nécessaire de poser au moins k questions (par exemple, « Est-ce que le i-ième bit du nombre est 1 ? »).

Métrique. Si les objets sont plongés dans un espace métrique (E, d), alors pour toute paire d'objets (x, y), on peut déterminer $\{z \in E, d(z, x) \leqslant d(z, y)\}$, l'ensemble des points plus proches de x que de y (si on considère la distance euclidienne dans $\mathbb{R}^d$ on peut parler de l'hyperplan séparateur $\{z \in \mathbb{R}^d, d(z, x) = d(z, y)\}$), permettant d'élaguer l'espace de recherche après avoir posé la question $Oracle_t(x, y)$.

Recherche binaire généralisée. Un algorithme connu pour déterminer une hypothèse parmi un ensemble au moyen de questions à réponses binaires est étudié en section 2. À chaque étape, il détermine de façon gloutonne la question qui sépare l'espace d'hypothèses en deux sous-ensembles de masses de probabilité proches. Sa complexité en nombre de questions est excellente en pratique (un facteur constant fois l'algorithme optimal) mais sa complexité pour choisir chaque question dans le cadre de la recherche de contenu par comparaisons est en $O(n^3)$ car il y a autant de questions possibles que de paires de points, ce qui est rédhibitoire.

Constante de doublage. Certains modèles considérant des distances non euclidiennes (Karbasi et al., 2011, 2012) font intervenir dans le calcul de complexité un paramètre supplémentaire. Notons $B_x(R) = y \in E, d(x, y) \leqslant R$ la boule de centre x et de rayon R. Si on étend la distribution aux ensembles d'objets : $\mu(A) = \sum_{x \in A} \mu(x)$, on appelle *constante de doublage* de μ la borne supérieure du quotient $\frac{\mu(B_x(2R))}{\mu(B_x(R))}$ pour tous x et R. Il s'agit donc de borner la masse de probabilité de toute boule par rapport à la masse de probabilité de la boule de même centre et de rayon moitié. L'algorithme proposé par Karbasi et al. (2012) trouve une cible avec une complexité de $O(4c^6(1 + H(\mu))$ questions (où $H(\mu)$ dénote l'entropie de μ), et une complexité de $O(n(\log n + c^6) \log c)$ pour choisir chaque question [2].

2. Nous avons pu réduire ces bornes à respectivement $O(4c^4(1+H(\mu)))$ et $O(n(\log n+c^4) \log c$.

Constante de désordre. D'autres modèles ne considèrent aucune métrique a priori sur les objets (Lifshits and Zhang, 2009 ; Tschopp and Diggavi, 2009), ainsi il est seulement possible d'ordonner les voisins d'un objet du plus proche au plus lointain. On considère alors le *rang* d'un objet y par rapport à un objet x, noté $\text{rank}_y(x)$ [3] et la boule de centre x et de rayon $R \in \mathbb{N}$ est l'ensemble des R plus proches voisins de x. Pour compenser l'absence d'inégalité triangulaire, on considère qu'il existe une constante D appelée *constante de désordre* vérifiant les relations suivantes, identiques à permutation près :

$$\begin{aligned}\forall x, y, z, \text{rank}_y(x) &\leq D(\text{rank}_y(z) + \text{rank}_z(x))\\ \text{rank}_y(x) &\leq D(\text{rank}_y(z) + \text{rank}_x(z))\\ \text{rank}_y(x) &\leq D(\text{rank}_z(y) + \text{rank}_z(x))\\ \text{rank}_y(x) &\leq D(\text{rank}_z(y) + \text{rank}_x(z)).\end{aligned}$$

Il est alors possible (Lifshits and Zhang, 2009) de retrouver le plus proche voisin en temps $O(D^4 \log n)$ et espace $O(D^5 n + Dn \log n)$, après un précalcul en temps $O(D^7 n \log^2(n))$.

Résistance à l'erreur. Le plus faible modèle d'erreur consiste à supposer que les erreurs sont indépendantes identiquement distribuées (iid) et que l'oracle répond toujours faux avec probabilité $\varepsilon < \frac{1}{2}$, auquel cas il suffit de lui poser chaque question plusieurs fois et de ne conserver que la réponse majoritaire pour obtenir une probabilité de succès arbitrairement proche de 1. Aucun modèle de la littérature ne considère le cadre plus réaliste d'erreurs persistantes, où l'oracle fournit toujours la même réponse lorsqu'on lui présente la même requête.

1.3 Stratégies considérées pour le choix des questions

Pour choisir la question suivante à poser, on peut considérer plusieurs approches gloutonnes.

Recherche exhaustive. Parmi toutes les questions, on choisit celle qui réalise la meilleure réduction d'entropie sur la distribution des hypothèses. Mais cette approche a une complexité en $O(mC)$ où m est le nombre de questions, typiquement quadratique en la taille du catalogue, et C est le coût de l'évaluation d'une réduction d'entropie.

3. Notamment, $\text{rank}_y(x)$ n'est pas forcément égal à $\text{rank}_x(y)$.

Meilleur parmi k choix. De la littérature des tables de hachage (Mitzenmacher, 2001) nous vient la méthode suivante : sélectionner k questions au hasard selon la loi dont on dispose, puis ensuite choisir la meilleure d'entre elles. Ainsi il n'y a que k évaluations de réduction d'entropie à effectuer et le coût de cette approche devient $O(kC) = O(C)$.

1.4 Notre contribution

Après avoir étudié les principales méthodes existantes, voici quelques problématiques que nous avons formulées dans le cadre $E = \mathbb{R}^d$ muni de la distance euclidienne :

— Comment se comporte l'approche « meilleur parmi k choix » par rapport à la recherche exhaustive ?

— Comment la recherche exhaustive se comporte-t-elle dans le cadre d'erreurs indépendantes identiquement distribuées ?

— Comment évolue le nombre de questions lorsque la dimension augmente ?

Nous avons recouru à une implémentation d'un algorithme de la littérature afin de tester différentes hypothèses.

2 Analyse d'un algorithme glouton bayésien

2.1 Cas sans erreur

Nowak (2009) considère un espace d'hypothèses $\mathcal{H}$ et un espace de questions $\mathcal{X}$. Les hypothèses sont des applications de $\mathcal{X}$ dans $\{-1, 1\}$ et il s'agit de trouver l'hypothèse correcte h^* de l'ensemble, en éliminant les autres éléments au moyen des questions. Il est simple de voir qu'il vaut mieux à chaque tour poser la question qui va diviser l'espace d'hypothèses en deux ensembles de cardinaux proches, en d'autres termes la question x qui va minimiser $|\sum_{h\in H} h(x)|$, quantité valant zéro lorsqu'il y a autant d'hypothèses telles que $h(x) = 1$ que d'hypothèses telles que $h(x) = -1$.

Initialisation : $n = 0, \mathcal{H}_0 = \mathcal{H}$.
Tant que $|\mathcal{H}_n| > 1$
    Choisir la question $x_n = \operatorname{argmin}_{x\in\mathcal{X}} |\sum_{h\in\mathcal{H}_n} h(x)|$
    Poser la question x_n pour obtenir la réponse $y_n = h^*(x_n)$
    Mettre à jour $\mathcal{H}_{n+1}$ avec $\{h \in \mathcal{H}_n, h(x_n) = y_n\}$ et incrémenter n
Retourner l'unique élément de $\mathcal{H}_n$.

FIGURE 2 – Algorithme de recherche binaire généralisée.

Nowak introduit une notion de k-voisinage comme suit : on note $\mathcal{H} = (h_1, \ldots, h_N)$. Deux questions x et x' sont dites k-voisines si leurs réponses en fonction des hy-

pothèses $(h_1(x), \ldots, h_N(x))$ et $(h_1(x'), \ldots, h_N(x'))$ diffèrent sur au plus k composantes. On dit que la paire $(\mathcal{X}, \mathcal{H})$ a la propriété de k-voisinage si le graphe de k-voisinage de l'espace est connexe.

Théorème 1. Si $(\mathcal{X}, \mathcal{H})$ a la propriété de k-voisinage, alors l'algorithme de recherche binaire généralisée termine et retourne la bonne hypothèse après au plus $\lceil \frac{\log N}{\log(\lambda^{-1})} \rceil$ questions, où $\lambda = \max\{\frac{1+c^*}{2}, \frac{k+1}{k+2}\}$.

Cette borne a ensuite été améliorée par Golovin and Krause (2010) :

Théorème 2. L'algorithme de recherche binaire généralisée fait au plus $OPT \cdot (H_{\max}(\mu) + 1)$ questions en moyenne pour identifier l'hypothèse $h^* \in \mathcal{H}$, où OPT est le nombre moyen minimum de questions posées par n'importe quelle méthode adaptative pour trouver la cible et $H_{\max}(\mu) = \max_{x \in \text{supp}(\mu)} \log \frac{1}{\mu(x)}$.

2.2 Cas avec erreur : mises à jour bayésiennes

Nowak propose une version de l'algorithme de recherche binaire généralisée résistant aux erreurs iid. Cet algorithme dépend d'un coefficient β nommé « *boost* bayésien » censé représenter une estimation du taux d'erreurs iid.

`Initialisation :` p_0 `uniforme sur` $\mathcal{H}$.
`Tant que` $\max_{h \in \mathcal{H}} p_n(h)$ `est inférieur à un certain seuil`
    `Choisir la question` $x_n = \operatorname{argmin}_{x \in \mathcal{X}} |\sum_{h \in \mathcal{H}} p_n(h) h(x)|$
    `Poser la question` x_n `pour obtenir la réponse bruitée` $y_n =^? h^*(x_n)$
    `Mettre à jour` $p_{n+1}(h)$ `avec` $\begin{cases} (1-\beta) p_n(h) & \text{si } y_n = h(x_n) \\ \beta p_n(h) & \text{sinon} \end{cases}$
    `Normaliser` p_{n+1}
    `Incrémenter` n
`Retourner` $\operatorname{argmax}_{h \in \mathcal{H}} p_n(h)$

FIGURE 3 – Algorithme de recherche binaire généralisée résistant aux erreurs iid.

En d'autres termes, l'algorithme favorise les hypothèses concordant avec h^* sur la question x_n et pénalise les autres. Justifions cette mise à jour bayésienne en notant ε la probabilité d'erreur. $\{h^* = h\}$ signifie que l'hypothèse h est celle cherchée, $\{y_n = h(x_n)\}$ que h concorde avec la réponse de l'oracle et $\{y_n = h^*(x_n)\}$ que l'oracle n'a pas fait d'erreur (événement survenant avec probabilité $1 - \varepsilon$). On a :

$$p_{n+1}(h) = \mathbf{Pr}(h^* = h | y_n = h(x_n)) \cdot 1_{\{y_n = h(x_n)\}} + \mathbf{Pr}(h^* = h | y_n \neq h(x_n)) \cdot 1_{\{y_n \neq h(x_n)\}}.$$

Or, par exemple :

$$\begin{aligned}\mathbf{Pr}(h^* = h | y_n = h(x_n)) &= \frac{\mathbf{Pr}(y_n = h(x_n)|h^* = h)\mathbf{Pr}(h^* = h)}{\mathbf{Pr}(y_n = h(x_n))} \\ &\propto \mathbf{Pr}(y_n = h(x_n)|h^* = h) \cdot p_n(h)\end{aligned}$$

où $\mathbf{Pr}(y_n = h(x_n)|h^* = h)$ vaut

$$\begin{aligned}&\mathbf{Pr}(y_n = h(x_n)|h^* = h, y_n = h^*(x_n))\mathbf{Pr}(y_n = h^*(x_n)) \\ &\quad + \mathbf{Pr}(y_n = h(x_n)|h^* = h, y_n \neq h^*(x_n))\mathbf{Pr}(y_n \neq h^*(x_n)) \\ &\quad = 1 \cdot (1 - \varepsilon) + 0 \cdot \varepsilon \\ &\quad = 1 - \varepsilon.\end{aligned}$$

Finalement :

$$p_{n+1}(h) \propto (1 - \varepsilon)p_n(h) \cdot 1_{\{y_n = h(x_n)\}} + \varepsilon p_n(h) \cdot 1_{\{y_n \neq h(x_n)\}}.$$

2.3 Réduction d'entropie à un pas

Pour le problème que nous avons étudié, les hypothèses correspondent aux points x de l'espace $E \subset \mathbb{R}^d$ à n éléments, muni de la distance euclidienne. Les questions sont de la forme « La cible est-elle plus proche de ce x que de ce y ? » et sont donc représentées par des paires d'éléments de E : $\mathcal{X} = \mathcal{P}_2(E)$. On associe donc à chaque point de l'espace z l'hypothèse h_z définie comme suit :

$$\forall \{x, y\} \in \mathcal{X}, h_z(\{x, y\}) = \begin{cases} +1 & \text{si } d(z, x) \leqslant d(z, y) \\ -1 & \text{sinon.} \end{cases}$$

Proposition 1. À un instant n donné de l'algorithme à la Figure 3, la question donnée par l'expression $x_n = \operatorname{argmin}_{x \in \mathcal{X}} |\sum_{h \in \mathcal{H}} p_n(h)h(x)|$ est celle réalisant la meilleure réduction d'entropie.

Démonstration. Notons $\mathcal{H}'$ l'ensemble des points concordant avec la réponse « 1 » de l'oracle, c'est-à-dire $\mathcal{H}' = \{h \in \mathcal{H}, h(x_n) = 1\}$, et soit $C = \sum_{h \in \mathcal{H}'} p(h) = p(h^*(x_n) = 1)$ et $C' = 1 - C$. Le problème revient à montrer que la meilleure question x_n est celle pour laquelle C est le plus proche de $1/2$. Tout d'abord :

$$\begin{aligned}\mathbf{Pr}(y_n = 1) &= \mathbf{Pr}(y_n = 1|y_n = h^*(x_n))\mathbf{Pr}(y_n = h^*(x_n)) \\ &\quad + \mathbf{Pr}(y_n = 1|y_n \neq h^*(x_n))\mathbf{Pr}(y_n \neq h^*(x_n)) \\ &= C(1 - \varepsilon) + C'\varepsilon.\end{aligned}$$

On note $D = C(1-\varepsilon)+C'\varepsilon$ et $D' = 1 - D$ et on considère que le *boost* bayésien β a été choisi comme valant la probabilité d'erreur ε.

$$
\begin{aligned}
&H(p_n) - H(p_{n+1}) \\
&= -\sum_{h\in\mathcal{H}} p(h)\log p(h) \\
&\quad + \underbrace{\mathbf{Pr}(y_n = 1)}_{D}\left(\sum_{h\in\mathcal{H}'}\frac{(1-\varepsilon)p(h)}{D}\log\frac{(1-\varepsilon)p(h)}{D} + \sum_{h\notin\mathcal{H}'}\frac{\varepsilon p(h)}{D}\log\frac{\varepsilon p(h)}{D}\right) \\
&\quad + \underbrace{\mathbf{Pr}(y_n = -1)}_{D'}\left(\sum_{h\notin\mathcal{H}'}\frac{(1-\varepsilon)p(h)}{D'}\log\frac{(1-\varepsilon)p(h)}{D'} + \sum_{h\in\mathcal{H}'}\frac{\varepsilon p(h)}{D'}\log\frac{\varepsilon p(h)}{D'}\right) \\
&= -\sum_{h\in\mathcal{H}} \underbrace{p(h)\log p(h) - (1-\varepsilon)p(h)\log p(h) + \varepsilon p(h)\log p(h)}_{0} \\
&\quad + \sum_{h\in\mathcal{H}}(1-\varepsilon)p(h)\log(1-\varepsilon) + \sum_{h\in\mathcal{H}}\varepsilon p(h)\log\varepsilon \\
&\quad - ((1-\varepsilon)C + \varepsilon C')\log D - ((1-\varepsilon)C' + \varepsilon C)\log D' \\
&= (1-\varepsilon)\log(1-\varepsilon) + \varepsilon\log\varepsilon - D\log D - (1-D)\log(1-D).
\end{aligned}
$$

Cette valeur est maximale pour $D = \frac{1}{2} = (1-2\varepsilon)C + \varepsilon$ soit $C = \frac{1-2\varepsilon}{2(1-2\varepsilon)} = \frac{1}{2}$.

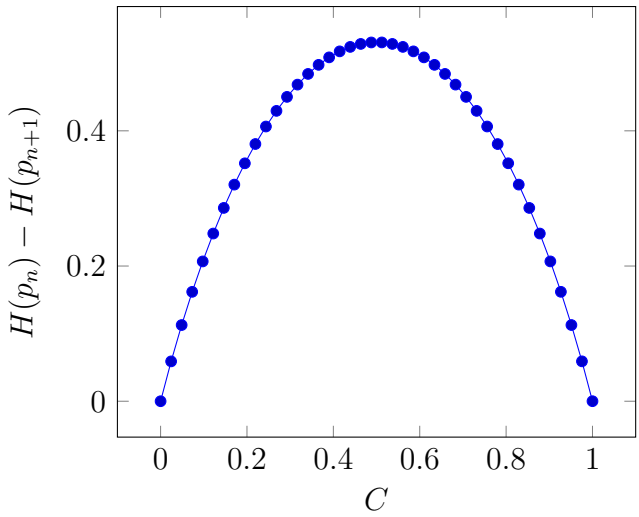

FIGURE 4 – Évolution de la réduction d'entropie en fonction de la probabilité que la vraie réponse à la question posée soit 1.

□

3 Résultats expérimentaux

Nous avons implémenté la méthode de recherche binaire généralisée résistant aux erreurs iid (cf. Figure 3) dans notre cadre de métrique euclidienne. Les différents paramètres de notre algorithme, dont le code est en annexe, sont les suivants.

— le nombre d'éléments N ;
— la dimension de l'espace $\mathbb{R}^d$;
— la stratégie utilisée, entre recherche exhaustive et « meilleur parmi k choix » ;
— la probabilité d'erreur iid ε ;
— le boost bayésien β ;
— le seuil de probabilité à partir duquel on considère que la cible est trouvée.

Une interface graphique a été développée pour représenter l'évolution de la distribution de probabilité sur les points de l'espace lors d'une partie (cf. Figure 5).

3.1 Effet des différents paramètres

Probabilité d'erreur. Le nombre de questions à poser est proportionnel à la probabilité d'erreur et explose lorsque celle-ci atteint $1/2$ (cf. Figure 6).

Stratégies. Alors que la recherche exhaustive a un coût rédhibitoire $O(n^3)$, les stratégies « meilleur parmi k choix ont un coût $O(n^2)$ et sont, de manière surprenante, aussi efficaces à une faible constante près. Plus k est grand, meilleur est le choix fait mais moins l'algorithme est rapide (cf. Figure 7). On constate aussi que le nombre de questions est logarithmique en le nombre d'éléments.

Dimension de l'espace. Dans le pire cas (cf. Figure 8), le nombre de requêtes est linéaire en la dimension de l'espace. Mais dans le cas général (cf. Figure 9), notre algorithme se révèle plus efficace lorsque la dimension augmente.

3.2 Discussion

La stratégie de « meilleur parmi k choix » se révèle très efficace, mais une preuve rigoureuse de sa performance est difficile. On retrouve le même problème lorsqu'on cherche à justifier que l'algorithme de recherche binaire généralisée se comporte mieux quand la dimension augmente. Le caractère discret de notre approche étant complexe à appréhender, peut-être serait-il plus simple de considérer le cas continu. Une intuition derrière ce résultat : lorsqu'on choisit des points uniformément au hasard dans la boule $[0, 1]^d$, avec forte probabilité ils sont loin du centre, donc l'hyperplan médiateur est à une distance faible du centre de la boule, ce qui conduit à une bisection qui coupe la boule en deux parties de volume proche, donc une bonne bisection.

En plus de l'approche expérimentale ayant permis de valider que l'heuristique « meilleur parmi k choix » était nécessaire pour obtenir un algorithme passant à l'échelle, nous avons testé cette approche sur des données réelles : l'algorithme de recherche binaire généralisé avec mises à jour bayésiennes a été implémenté pour chercher des films à partir de tags IMDb suggérés à l'utilisateur, via des questions de la forme : « Le film contient-il le tag suivant ? ». L'utilité de la résistance à l'erreur est alors immédiate, chaque interlocuteur ne répondant pas de la même manière aux questions posées.

Un travail reste à faire dans le cadre de la recherche de contenu par comparaisons sur des données réelles, la difficulté principale reposant sur l'identification de la géométrie sous-jacente à l'espace des données : peut-on plonger les objets dans une métrique, si possible euclidienne ? Quelle est la constante de doublage de la distribution ? Quelle est la constante de désordre ?

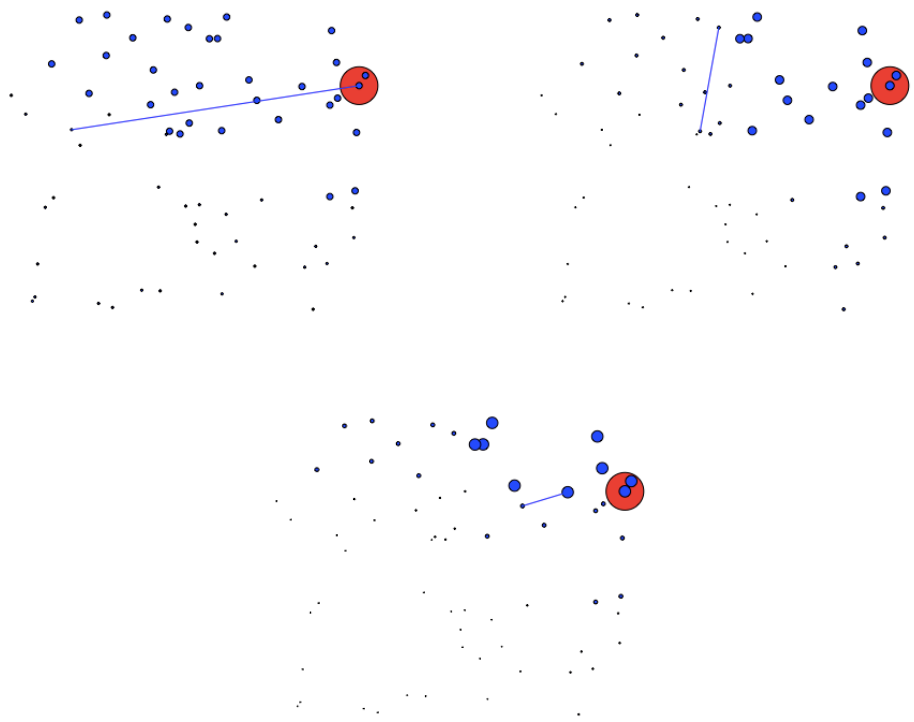

FIGURE 5 – Évolution d'une partie avec une stratégie de recherche exhaustive. La cible t est indiquée en rouge, le rayon des disques représente la masse de probabilité en chaque point et le segment bleu indique les deux points choisis x et y qui vont constituer la question : « Est-ce que t est plus proche de x que de y ? ». Ici, l'oracle ne fait aucune erreur, ce qui permet de resserrer le champ de recherche vers la cible.

Références

Daniel Golovin and Andreas Krause. Adaptive submodularity : A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2010.

Amin Karbasi, Stratis Ioannidis, and Laurent Massoulié. Adaptive content search through comparisons. *CoRR*, abs/1107.3059, 2011.

Amin Karbasi, Stratis Ioannidis, and Laurent Massoulié. Comparison-based learning with rank nets. *CoRR*, abs/1206.4674, 2012.

Yury Lifshits and Shengyu Zhang. Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design. In Claire Mathieu, editor, *SODA*, pages 318–326. SIAM, 2009.

Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10) :1094–1104, 2001.

Robert D. Nowak. The geometry of generalized binary search. *CoRR*, abs/0910.4397, 2009.

Dominique Tschopp and Suhas N. Diggavi. Approximate nearest neighbor search through comparisons. *CoRR*, abs/0909.2194, 2009.

A Implémentation en Python

La version de l'algorithme ici présentée est édulcorée, pour une meilleure lisibilité. Le code complet peut être trouvé à l'adresse suivante :

`https://bitbucket.org/jilljenn/search-through-comparisons/src`

```
1  # coding=utf8
2  import math, random, json
3  from itertools import combinations
4  from datetime import datetime
5  import numpy as np
6  import matplotlib.pyplot as plt
7
8  ALL = 1000
9  RANDOM = 0
10
11 def dist(h1, h2):
12     return sum(pow(h1[i] - h2[i], 2) for i in range(len(h1)))
```

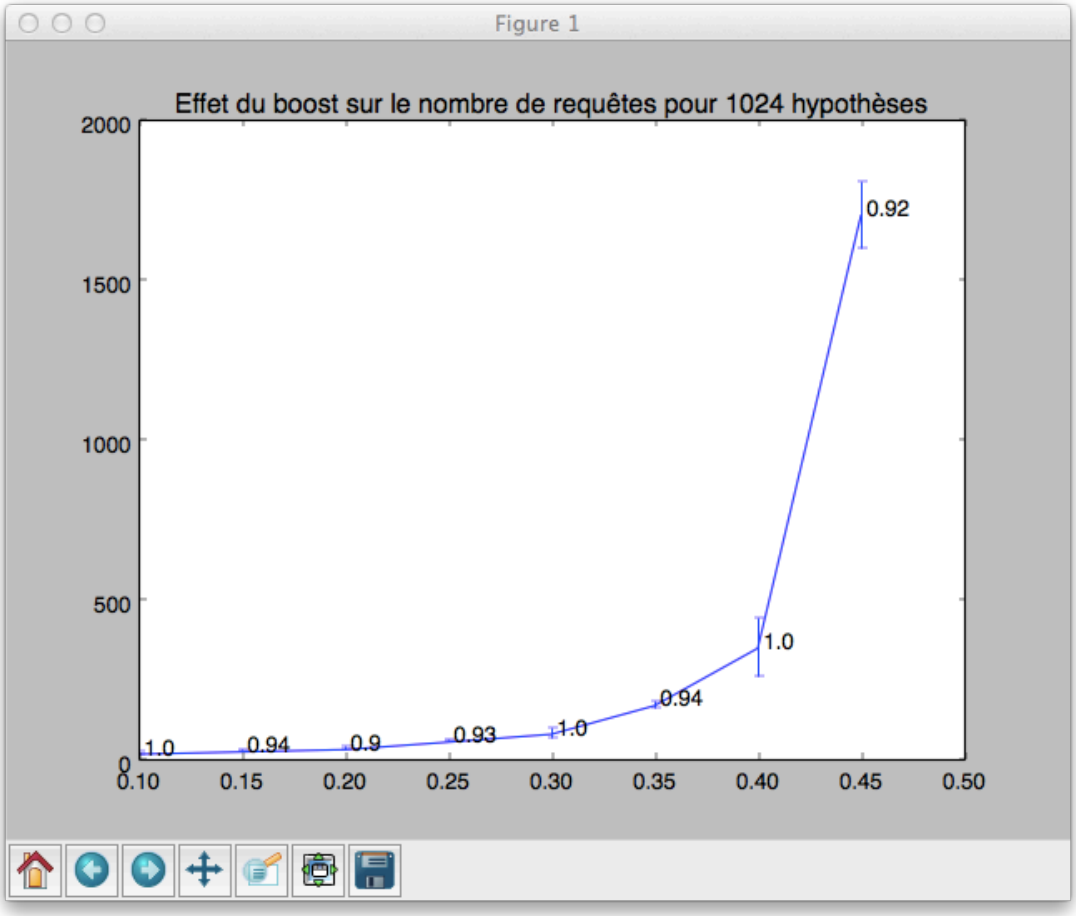

FIGURE 6 – Évolution du nombre moyen de questions en fonction de la probabilité d'erreur. Lorsque cette dernière tend vers $1/2$, le nombre de questions nécessaires pour trouver la cible explose. Les étiquettes des points correspondent au taux de succès.

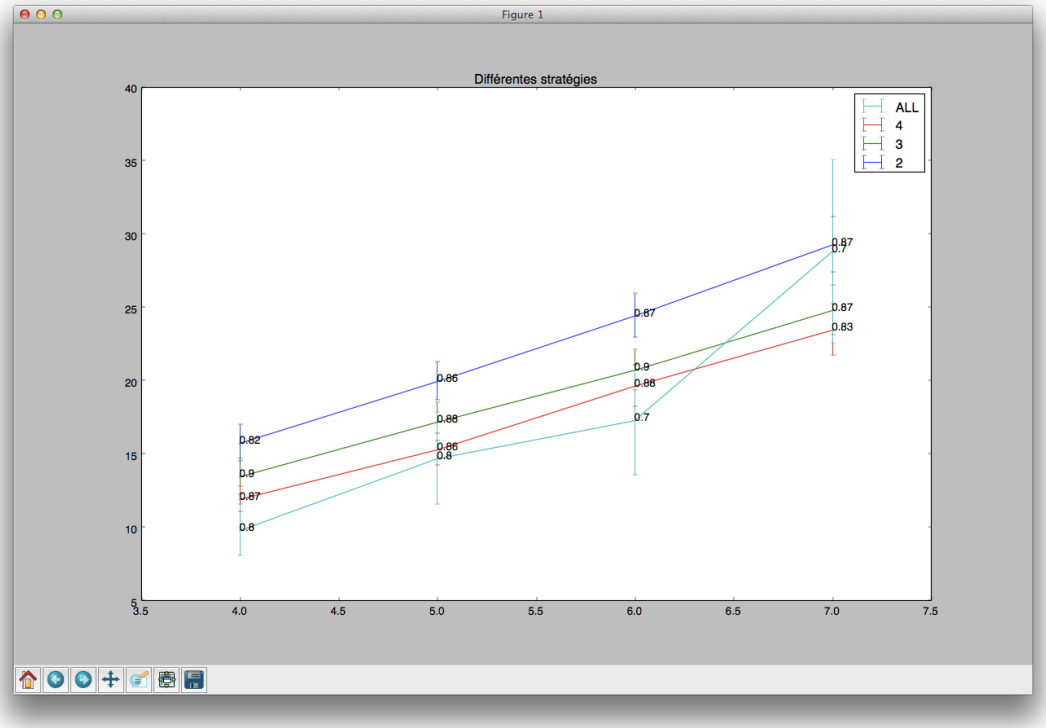

FIGURE 7 – Évolution du nombre moyen de questions en fonction du logarithme du nombre d'éléments, pour les stratégies « meilleur parmi $\{2, 3, 4\}$ choix » et recherche exhaustive.

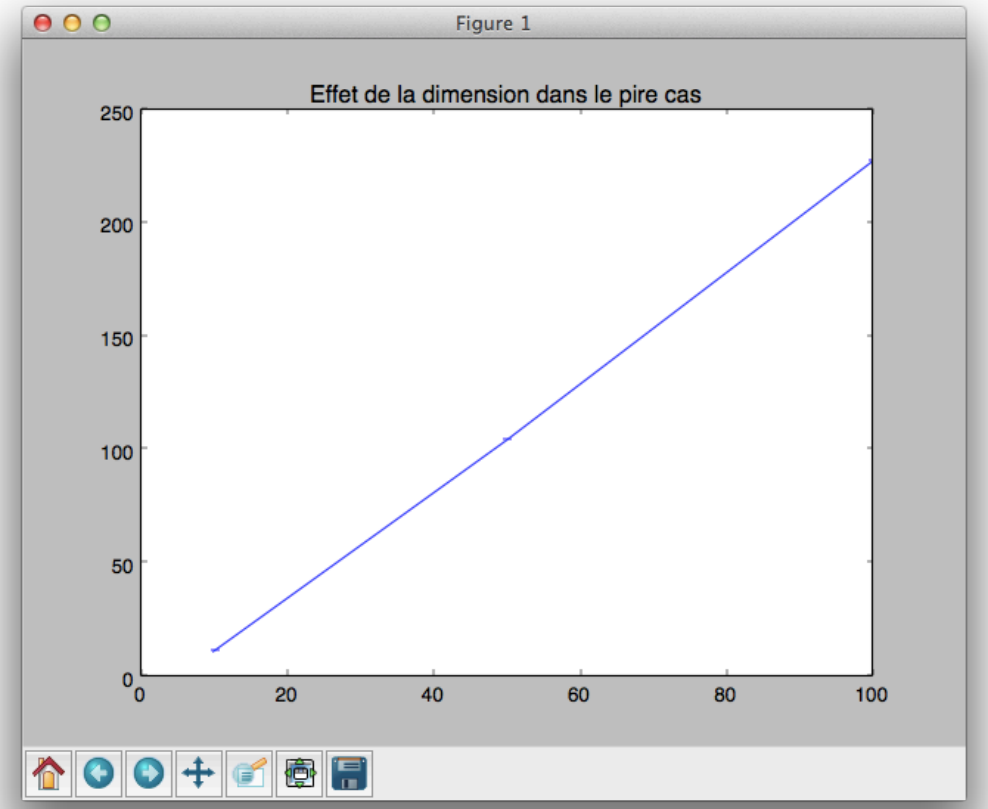

FIGURE 8 – Évolution du nombre moyen de questions en fonction de la dimension dans le pire cas. Ici, comme points de $\mathbb{R}^d$, on a choisi ceux de la base canonique. Ainsi tout point est équidistant de tout autre et chaque requête ne permet de réduire l'espace de recherche que d'un élément, d'où proportionnalité entre le nombre de questions et la dimension.

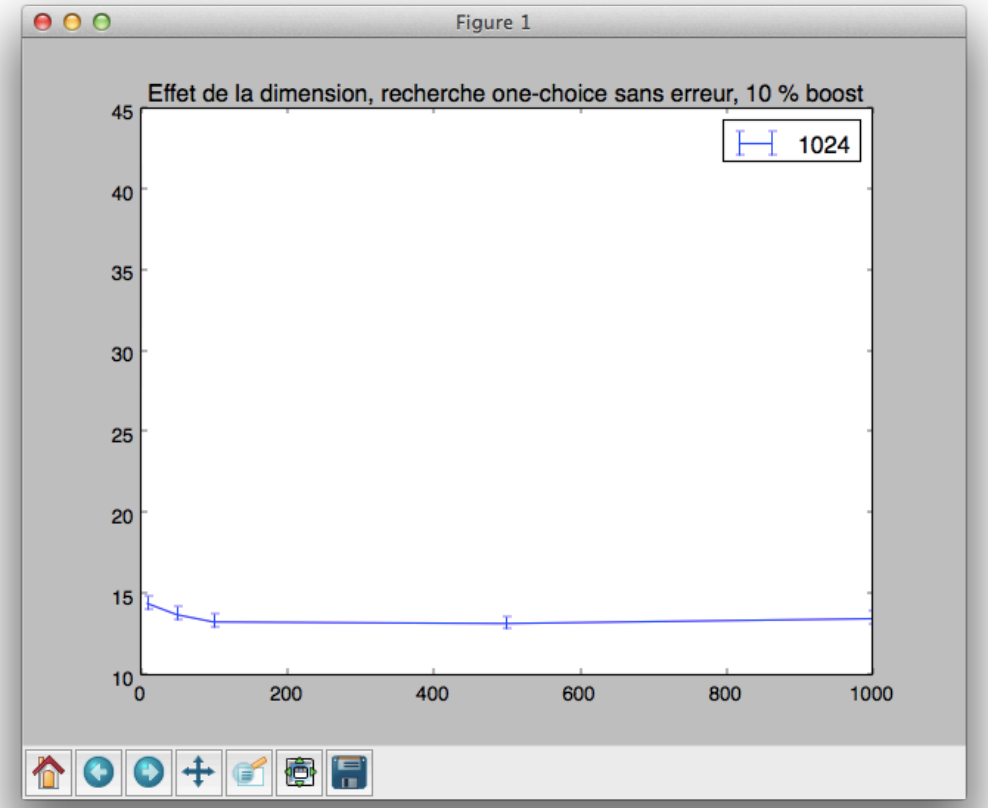

FIGURE 9 – Évolution du nombre moyen de questions en fonction de la dimension dans le cas général. C'est un résultat contre-intuitif : l'algorithme se révèle plus efficace lorsque la dimension augmente.

```
13
14 def normalize(p):
15     return [float(i) / sum(p) for i in p]
16
17 def sample2(p): # Samples according to non-uniform distribution
18     r = random.random()
19     s = 0
20     i = 0
21     while s < r :
22         s += p[i]
23         i += 1
24     return i - 1
25
26 def entropy(p):
27     return -sum(i * math.log(i) for i in p)
28
29 def surround(p): # Recursiw rounding of list of elements
30     if isinstance(p, (list, tuple)):
31         return map(surround, p)
32     return round(p, 3)
33
34 def display(t):
35     if len(t) < 10:
36         return surround(t)
37     else :
38         return '[long]'
39
40 def avgstdist(l): # Displays mean and variance of number of queries
41     n = len(l)
42     mean = float(sum(l)) / n
43     var = float(sum(i * i for i in l)) / n - mean * mean
44     print mean, '±', round(1.96 * math.sqrt(var / n), 5)
45     return mean, round(1.96 * math.sqrt(var / n), 5)
46
47 def get_nb_queries(prefix=None, N=1 << 7, d=2, b=0.2, \
48     confidence=0.85, strategy=2, error=0):
49     """ Parameters :
50     N : Number of hypotheses
51     d : Dimension
52     b : Bayesian boost
```

```
    confidence : Confidence threshold (according to which the target is chosen)
    strategy : RANDOM, 2 (power of two choices), 3, etc. or ALL (exhaustive)
    """
    nb_experiments = 20
    graph = True
    homogeneous = True # Is the initial distribution uniform?

    values = []
    couples = list(combinations(range(N), 2))
    nb_success = 0
    for _ in range(nb_experiments):
        hyp = [tuple(random.random() for _ in range(d)) for _ in range(N)]
        p = [1. / N] * N if homogeneous else \
            normalize([random.random() for _ in range(N)]) # Priori
        itarget = sample2(p) # or random.choice(range(N))
        target = hyp[itarget]
        nb_tours = 1
        while True :
            if strategy == RANDOM :
                query = random.choice(range(N * (N - 1) / 2))
            else :
                if strategy == ALL :
                    indexes = range(N * (N - 1) / 2)
                else :
                    h_indexes = set()
                    attempts = []
                    # Until we get enough contestants
                    while len(h_indexes) < strategy :
                        i = sample2(p) # Samples a new contestant
                        attempts.append(i)
                        h_indexes.add(i)
                    h_indexes = list(h_indexes)
                    indexes = [couples.index((h1, h2)) \
                        for h1, h2 in combinations(sorted(h_indexes), 2)]
                        # Shamelessly inefficient
                contestants = []
                for ii in indexes :
                    i1, i2 = couples[ii]
                    contestants.append((abs(sum([p[i] for i in range(N) if \
                        dist(hyp[i], hyp[i1]) < dist(hyp[i], hyp[i2])]) - 0.5), ii))
```

```
93                  contestants.sort()
94                  # Choosing the best of query contestants
95                  _, query = contestants[0]
96              i1, i2 = couples[query]
97              mistake = random.random() < error
98              if graph and d == 2:
99                  plt.clf()
100                 plt.scatter(target[0], target[1], c='r', s=1000) # Target in red
101                 plt.scatter(*zip(*hyp), s=[1000 * i for i in p])
102                 plt.plot(*zip(hyp[i1], hyp[i2]), c='r' if mistake else 'b')
103                 plt.draw()
104                 plt.show(block=False)
105                 raw_input()
106             closest_h, farthest_h = (hyp[i1], hyp[i2]) \
107                 if dist(target, hyp[i1]) < dist(target, hyp[i2]) \
108                 else (hyp[i2], hyp[i1])
109             if mistake :
110                 closest_h, farthest_h = farthest_h, closest_h # Oops !
111             else :
112                 np = normalize([p[i] * (1 - b \
113                     if dist(hyp[i], closest_h) <= dist(hyp[i], farthest_h)
114                     else b) for i in range(N)])
115             p = np # Posteriori becomes priori
116             ranking = sorted((p[i], i) for i in range(N))
117             if ranking[-1][0] > confidence : # Stopping rule
118                 break
119             nb_tours += 1
120         if ranking[-1][1] == itarget : # Correctly guessed
121             nb_success += 1
122         values.append(nb_tours)
123     print nb_success, u'réussis sur', nb_experiments
124     mean, interval = avgstdist(values)
125
126 get_nb_queries(N=1 << 7, d=2, b=0.1, confidence=0.9, strategy=3, error=0.1)
```

Bibliography for Search Through Comparisons

Jill-Jênn Vie

26 septembre 2013

Title	General	Implemented	Theoretical	Scope	Error-tolerant	Clever Data Structures	Kill feature
The Geometry of Generalized Binary Search *Nowak, 2009*	Yes		Yes		iid		k-neighborliness
Combinatorial Algorithms for NN, ND & SWD *Lifshits & Zhang, 2009*			Yes	No metric		Yes	up-aside-down
Approximate NN Search Through Comparisons *Tschopp & Diggavi, 2010*				No metric			Unreadable
Adaptive Content Search Through Comparisons *Karbasi et al., 2011*			Yes	Doubling constant		No	
Comparison-Based Learning with Rank Nets *Karbasi et al., 2012*		Yes		Doubling constant	iid	Yes	Zoom
Active Ranking using Pairwise Comparisons *Jameson & Nowak, 2011*		Yes		Euclidean	Persistent		Ambiguity threshold
Adaptive Submodularity *Golovin & Krause, 2010*	Yes		Yes	Submodular functions			Submodularity
Fast construction of Nets in Low-D Metrics *Har-Peled & Mendel, 2010*	Yes		Yes			Yes	