

UNIVERSITÉ DE LIÈGE
FACULTÉ DES SCIENCES APPLIQUÉES

THÉORIE DE L'INFORMATION
ET DU CODAGE

NOTES RELATIVES AUX TRAVAUX PRATIQUES

Année académique 2008-2009

Table des matières

1	Introduction	2
1.1	Outils mis à disposition	2
1.2	Organisation	2
2	Compressive sensing	3
2.1	Principe	3
3	Utilisation de cvx	4
4	Analyse d'un signal creux	5
4.1	Différents types de matrices	5
4.1.1	de variables gaussiennes	5
4.1.2	de variables de Bernoulli	5
4.1.3	d'échantillonnage aléatoire	5
4.2	Questions	5
5	Analyse d'un signal à la représentation creuse	7
5.1	Questions	7
6	Comportement par rapport au bruit	7
6.1	Questions	7
7	Compression d'images	8
7.1	Questions	8

1 Introduction

Ces notes reprennent l'ensemble des informations nécessaires aux travaux pratiques sur ordinateur du cours de théorie de l'information et du codage.

Pour l'année académique 2008-2009, le travail porte essentiellement sur l'étude de l'échantillonnage comprimé. Le but de ces travaux pratiques est de permettre aux étudiants de manipuler de façon concrète les notions théoriques introduites grâce à différentes expériences.

1.1 Outils mis à disposition

Un article scientifique présentant le concept de l'échantillonnage comprimé vous sera fourni, ainsi que plusieurs fichiers Matlab illustrant ce procédé. Pour faire fonctionner ces fichiers, il est nécessaire de disposer de la toolbox *cvx*¹.

Pour installer cette toolbox, il faut procéder de la manière suivante :

- i. Télécharger le fichier zip d'archive².
- ii. Le décompresser sur votre ordinateur, mais pas dans le répertoire *toolbox* de Matlab.
- iii. Ouvrir Matlab, et modifier le répertoire de travail pour qu'il soit le répertoire *cvx* où l'archive s'est extraite.
- iv. Utiliser la commande '*cvx_setup*'. Si tout va bien, vous obtenez le message ci-dessous. Attention, il peut être nécessaire de répéter cette dernière étape à chaque nouveau lancement de Matlab.

```
Testing the cvx distribution. If this script aborts with  
an error, please report the error to the authors.
```

```
-----  
No errors! cvx has been successfully installed.
```

1.2 Organisation

- Le travail doit être réalisé par groupe de deux étudiants.
- Une fois constitué, le groupe enverra un e-mail à l'adresse fschnitzler@ulg.ac.be signalant les noms des étudiants constituant ce groupe, ainsi qu'une adresse e-mail. Les codes Matlab nécessaires à la réalisation du travail seront envoyés à cette adresse.

On demande aux étudiants de remettre un rapport écrit décrivant, discutant et justifiant de façon suffisamment détaillée les résultats obtenus par expérimentation. Le but est de montrer qu'on a bien compris les notions sur lesquelles porte le travail.

Les changements apportés aux codes, tels que la modification des paramètres, doivent être clairement détaillés, de façon à pouvoir reconstituer les simulations à partir des fichiers initialement transmis.

Le rapport du groupe doit être transmis par e-mail à fschnitzler@ulg.ac.be trois jours avant l'examen oral, et une copie papier remise le jour-même. Vous serez interrogé sur votre bonne compréhension du travail.

Une séance de questions-réponses sera organisée le vendredi 12 décembre. Vous pouvez également envoyer vos éventuelles questions par e-mail.

1. <http://www.stanford.edu/~boyd/cvx/>
2. <http://www.stanford.edu/~boyd/cvx/cvx.zip>

2 Compressive sensing

Ceci est une brève introduction au compressive sensing, principalement présente pour fixer les notations. Il vous est demandé de vous référer à cet article³ pour une explication plus détaillée.

2.1 Principe

L'approche traditionnelle pour l'acquisition de données se base sur le théorème de Shannon-Nyquist : pour acquérir un signal ayant une bande passante de taille W , il faut échantillonner à une fréquence supérieure à $2W$. Le compressive sensing exploite le fait que de nombreux signaux réels peuvent s'exprimer de manière creuse et l'incohérence entre certains types de bases pour réduire ce nombre d'échantillons.

Un vecteur S -creux est un vecteur qui a au plus S composantes non nulles. Beaucoup de signaux naturels, lorsqu'ils sont exprimés dans une base particulière, ont une représentation comportant de nombreux coefficients négligeables. On peut par exemple citer la transformation en cosinus, la transformation de Fourier et les ondelettes. La compression de données exploite ce fait, en supprimant ces faibles coefficients, ce qui réduit légèrement la qualité du signal, mais le rend creux. Pour calculer le vecteur x qui exprime un signal f de longueur n dans une base Ψ , il faut projeter le vecteur f sur chacun des vecteurs ψ_i : $x_i = \langle f, \psi_i \rangle$. On reconstitue le signal original en effectuant la somme des vecteurs de la base, pondérés par les coefficients x_i :

$$f = \sum_{i=1}^n x_i \psi_i$$

La cohérence μ entre deux bases Φ et Ψ mesure la plus forte corrélation entre deux éléments de ces deux bases. Elle se calcule de la manière suivante :

$$\mu = \sqrt{n} * \max_{k,j} |\langle \phi_k, \psi_j \rangle|$$

Une faible cohérence entre les bases indique que chacun des vecteurs de l'une d'elle, s'il était exprimé dans l'autre base, aurait toutes ses composantes faibles. C'est-à-dire que, si l'on échantillonne un signal f à l'aide de la base Φ , on récupère de l'information sur chacun des vecteurs de Ψ .

Si le signal f est creux dans la base Ψ , l'information contenue dans le signal y , obtenu en échantillonnant par la base Φ , est redondante. On peut donc ne garder qu'une partie du vecteur y . Une représentation schématique des diverses transformations est visible dans la Figure 2.1. Il est possible de reconstruire le signal x (et donc f) à partir d'un nombre d'échantillons m bien inférieur à ce que prévoit le théorème de Shannon-Nyquist. Pour ce faire, il faut chercher, parmi toutes les solutions reproduisant les échantillons y obtenus, celle qui est la plus creuse. En pratique, cette approche étant difficile, on cherchera la solution de norme l_1 la plus faible, c'est-à-dire celle qui minimise

$$\sum_{i=1}^n |x_i|$$

3. <http://www.acm.caltech.edu/~emmanuel/papers/spm-robustcs-v05.pdf>

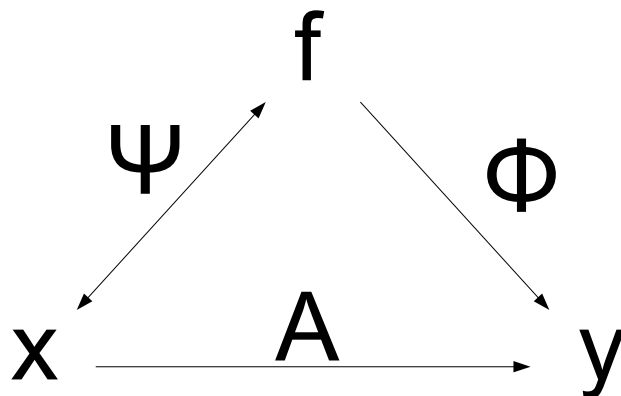


FIGURE 1 – Un signal f , possédant une représentation x S-creuse dans une certaine base Ψ , est échantillonné par une partie d'une base Φ , incohérente avec Ψ . A partir de ces échantillons y , il est possible de retrouver la forme creuse x .

3 Utilisation de cvx

Une fois `cvx` installé, il est possible de l'utiliser, en fournissant à Matlab une spécification `cvx`, ce qui peut être fait dans une fonction, dans un script, ou directement dans la fenêtre de commandes.

Voici un exemple de spécification `cvx`. Celle-ci comporte les éléments suivants :

- `cvx_begin` au début, et `cvx_end` à la fin, ce qui délimite l'environnement `cvx`, et permet à Matlab de le reconnaître comme tel.
- une déclaration de variables. Dans cet exemple est déclaré un vecteur x (sous-entendu de réels) de longueur n (défini plus haut). Pendant les calculs effectués par `cvx`, les variables sont représentées par des objets Matlab spéciaux. Une fois les calculs effectués, ces variables sont converties dans le format classique, et sont disponibles dans Matlab.
- le but de l'optimisation : minimiser la norme d'ordre 1 du vecteur x .
- les conditions sous lesquelles le problème doit être optimisé : $Ax = y$.

```
cvx_begin
variable x(n) ; %déclaration de variables
minimize(norm(x,1)); %objectif
subject to
A*x == y; %condition
cvx_end
```

4 Analyse d'un signal creux

Le code `SignalCreux.m` génère aléatoirement un signal 3-creux, en effectue l'échantillonnage au moyen d'une matrice Π choisie entre 3 possibilités, définies plus bas (matrice de variables gaussiennes, de variables de Bernoulli, ou par échantillonnage aléatoire), et reconstruit le signal en minimisant $\|\tilde{f}\|_1$ tel que $\Pi^* \tilde{f} = y$, où \tilde{f} représente le signal reconstruit.

Les paramètres sont les suivants :

- S : nombre de composantes non-nulles du signal.
- nn : longueur du signal.
- m : nombre d'échantillons.
- pi.format : type de matrice utilisé.

4.1 Différents types de matrices

4.1.1 de variables gaussiennes

Chaque $\Pi(i, j)$ est une variable aléatoire gaussienne de variance un. Chacun des vecteurs $c(j)$ de la base Π est ensuite normalisé.

4.1.2 de variables de Bernoulli

Les $\Pi(i, j)$ forment une suite de variables aléatoires de Bernoulli de valeurs ± 1 équiprobables, c'est-à-dire qu'elles ont chacune une probabilité 0.5 d'être plus ou moins un. Chacun des vecteurs de la matrice est ensuite normé.

4.1.3 d'échantillonnage aléatoire

Cette matrice réalise l'échantillonnage classique du signal. Chacun des vecteurs $c(j)$ peut être vu comme un vecteur $\delta(m - k)$, où $f(k)$ est l'échantillon de f qui est extrait par ce vecteur. On a donc $x(j) = f(k)$.

4.2 Questions

1. Identifiez les matrices Φ et Ψ du problème.
2. Ecrivez un script qui calcule la valeur de μ à partir de ces deux matrices.
Suggestion : utilisez la fonction 'max'.
3. Calculez l'entropie de la source émettant le signal creux. On considérera que la source S émet des messages 3-creux équiprobables de longueur 290. Les valeurs non-creuses sont des entiers compris entre 1 et 255.
4. Suggérez un codage de source réversible exploitant le caractère creux des messages émis par cette source. Quel est le taux de compression obtenu, par rapport au vecteur reprenant les 290 valeurs du signal ?
5. Vérifiez que ce code respecte l'inégalité de Kraft. Est-il possible de trouver un codage produisant des mots plus courts ?
6. Quel est l'intérêt du compressive sensing, comparé au code que vous avez suggéré ?
7. Ecrivez une fonction `CS(f,phi,psy)`, qui calcule les observations y et reconstruit un signal \tilde{f} à partir de y par minimisation de $\|\tilde{x}\|_1$ tel que $A\tilde{x} = y$.
8. La minimisation de $\|\tilde{x}\|_2$ sous la même condition produit-elle également de bons résultats ? Comment expliquez-vous la différence de performance ?

9. Comparez le comportement de l'algorithme, pour les 2 premières matrices proposées, lorsque la valeur de m ou de S est modifiée. Testez plusieurs valeurs pour m et S (une seule à la fois), et effectuez plusieurs simulations avec les mêmes valeurs. Quelle matrice se comporte le mieux ? Pouvez-vous justifier ce résultat théoriquement ?

5 Analyse d'un signal à la représentation creuse

Le code `RepresentationCreuse.m` génère aléatoirement un signal de longueur 512 qui a une représentation 6-creuse dans une certaine base Ψ . Vous disposez également de 3 fonctions permettant de créer différentes matrices Ψ : `Build_Fourier.m`, `Build_Haar.m` et `Build_Noiselets.m`.

5.1 Questions

10. Plusieurs matrices Ψ sont proposées. Recherchez celle qui représente une base dans laquelle le signal est creux.
Conseil : Certaines de ces matrices sont à valeurs complexes. Après une transformation, pensez à prendre la valeur absolue (ou réelle) du vecteur obtenu, si sa partie imaginaire est faible, dans le cas où vous souhaiteriez réaliser un graphique. La partie imaginaire peut être négligeable, et due aux approximations numériques.
11. Le signal a initialement été échantillonné à 512 Hz. Selon le théorème de Shannon-Nyquist, quelle est la fréquence maximale qui peut être correctement représentée? Que se passe-t-il si une fréquence d'échantillonnage plus basse est utilisée? Comparez avec le compressive sensing.
12. Pour chaque valeur de m entre 10 et 30, calculez le nombre de mauvaises reconstructions, et la moyenne de l'erreur maximale de reconstruction de x ($\|\tilde{x} - x\|_2$), en effectuant 30 simulations pour chaque m . Estimez la valeur de la constante C utilisée dans la formule du nombre minimal d'échantillons garantissant presque certainement la reconstruction correcte du signal (avec une probabilité δ).
Suggestion : Utilisez une condition sur l'erreur admissible sur n'importe quel échantillon pour détecter une mauvaise reconstruction. (par exemple : 0.01)

6 Comportement par rapport au bruit

Vous disposez d'un code `Build_Noise(arg_ecart , arg_m).m`, qui construit un vecteur colonne z de taille arg_m , contenant du bruit gaussien de moyenne nulle et d'écart type $\sigma = arg_ecart$.

6.1 Questions

13. Utilisez cette fonction, et le code de l'exemple précédent, pour construire un vecteur d'échantillons bruités $y^* = y + z$, de variance 1.
14. Exprimez $E(\epsilon^2)$ en fonction des grandeurs du problème (m et σ), avec $\epsilon = \|z\|_2$.
15. Ecrivez un script permettant de calculer $\|z\|_2$.
16. Modifiez le programme de reconstruction pour reconstruire le signal \tilde{x} à partir de y^* . Utilisez $\sigma\sqrt{m}$ comme borne pour la puissance du bruit, et utilisez 100 échantillons (paramètre m).
Conseil : Il suffit de modifier la condition sous laquelle se fait la minimisation de $\|\tilde{x}\|_1$. La norme d'ordre deux pour un vecteur s'appelle de la manière suivante pour cvx : `norm(x,2)`, et l'inégalité par le symbole traditionnel $<$.
17. Faites diverses simulations en modifiant l'écart type du bruit entre 0,25 et 2. Calculez la norme 2 de l'erreur de reconstruction, en fonction de la puissance (réelle) du bruit. Quelle relation observez-vous entre la puissance du bruit et l'erreur de reconstruction?

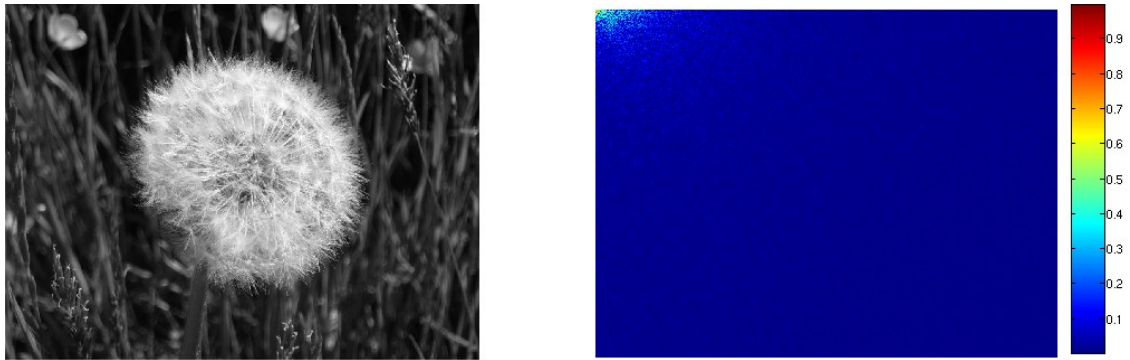


FIGURE 2 – Une image, et sa transformée en cosinus discret. La transformée est creuse, l’image peut donc être compressée en ne gardant que les coefficients significatifs de sa transformée. A l’origine, le nombre de coefficients est identique pour chacune des deux représentations.

7 Compression d’images

Le code `CompressionImage.m` extrait les données de l’image ‘photographe’, et en traite un morceau de 64 sur 64 pixels seulement, pour des raisons de temps de calcul. Le but de cette dernière partie du travail est de comparer les possibilités du compressive sensing pour la compression d’images, par rapport à une technique existante : JPEG-2000. Cette méthode est basée sur la transformée en cosinus discret, puis sur la suppression d’un certain nombre de coefficients négligeables. Un exemple d’une telle transformation est visible dans la Figure 2.

Deux techniques de compression sont testées : la première exploitant la transformation en cosinus discret (DC), la seconde le compressive sensing (CS). Le nombre de coefficients utilisés par ces 2 techniques est identique, et est déterminé par les paramètres $K1$ et $K2$. $K1$ définit le nombre de coefficients de la transformée que les deux méthodes utilisent. Il s’agit des coefficients basses fréquences, ceux qui sont généralement de plus haute énergie. $K2$ configure le nombre de coefficients supplémentaires de chaque méthode. Dans le cas de DC, il s’agit des coefficients suivants de la transformée en cosinus (acquis selon la méthode JPEG). Pour le compressive sensing, il s’agit de coefficients qui peuvent être considérés comme obtenus par projection sur des vecteurs aléatoires. (Il s’agit en réalité de projection sur des ‘noiselets’) Chaque méthode utilise donc $K1 + K2$ coefficients.

L’image DC est reconstruite par calcul de la pseudo-inverse, c’est-à-dire l’inverse de la transformation, à partir des seuls coefficients préservés. CS utilise une adaptation du principe de la minimisation de la norme l_1 . Une troisième image (LPTV), est reconstituée, en appliquant l’algorithme de reconstruction de CS aux coefficients de DC. Ceci est fait afin de vérifier que les performances ne sont pas dues à la méthode de reconstruction, mais bien à la technique d’échantillonnage.

La comparaison entre les trois images reconstituées se fait au moyen du rapport signal à bruit avec l’image originale.

Note : Il n’est pas demandé de comprendre toutes les opérations du code. L’explication donnée ci-dessus est suffisante. Les seuls paramètres à modifier sont $K1$ et $K2$, ainsi que la récupération des résultats. Vous êtes évidemment libres d’expérimenter par vous-même.

7.1 Questions

18. Calculez le taux de compression obtenu, en fonction de $K1$ et $K2$, pour les deux techniques de compression.

- 19.** Pour $K_1 = 700$, calculez le rapport signal sur bruit de DC, LPTV et CS, pour des valeurs de K_2 allant de 0 à 3000, avec un pas de 300. Affichez ces différentes valeurs sur un même graphique, et analysez-le.

□