Astuce du Noyau & Quantification Vectorielle

Matthieu Geist[1,2,3] Olivier Pietquin[1] Gabriel Fricout[2]

[1] Supélec, groupe de recherche IMS, Metz, France
[2] ArcelorMittal Research, cluster MC, Maizières-lès-Metz, France
[3] INRIA Nancy - Grand Est, équipe-projet CORIDA, France

matthieu.geist@supelec.fr

Résumé

L'astuce du noyau est une célèbre approche qui permet de transformer implicitement une méthode linéaire en une non-linéaire en remplaçant les produits scalaires par une fonction noyau. Cependant, peu d'algorithmes de quantification vectorielle en ont bénéficié. En effet, ils impliquent habituellement de calculer des transformations linéaires (par exemple le déplacement d'un prototype), ce qui ne permet pas l'application directe de cette astuce. Cet article introduit une méthodologie générale, consistant à travailler dans une approximation de l'espace de redescription et permettant de combiner les méthodes à noyaux aux algorithmes de quantification vectorielle. En conséquence, l'astuce du noyau peut être appliquée à tout type d'algorithme de quantification vectorielle, et plus généralement à tout algorithme impliquant au plus des transformations linéaires.

Mots Clef

Apprentissage numérique, méthodes à noyaux, quantification vectorielle.

Abstract

The kernel trick is a well known approach allowing to implicitly cast a linear method into a nonlinear one by replacing any dot product by a kernel function. However few vector quantization algorithms have been kernelized. Indeed, they usually imply to compute linear transformations (e.g., moving prototypes), what is not easily kernelizable. This paper introduces the Kernel-based Vector Quantization (KVQ) method which allows working in an approximation of the feature space, and thus kernelizing any Vector Quantization (VQ) algorithm.

Keywords

Machine Learning, Kernel Methods, Vector Quantization.

1 Introduction

Une approche classique pour traiter des problèmes non-linéaires est de projeter l'ensemble des données dans un espace de redescription (généralement de plus grande dimension) qui préserve le groupement inhérent des données tout en simplifiant la structure associée. Cependant, comme cet espace de redescription peut être de très grande dimension (voire de dimension infinie), travailler directement avec les variables projetées est généralement considéré comme une option irréaliste. L'astuce du noyau, que nous dérivons maintenant, permet de s'en affranchir.

Un noyau K est une fonction continue, symétrique et positive. L'astuce du noyau repose sur le théorème de Mercer [13], qui stipule que tout noyau peut être exprimé comme un produit scalaire dans un espace de plus grande dimension. Plus formellement, pour chaque noyau K, il existe une fonction $\varphi : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathcal{F}$ ($\mathcal{X}$ étant l'espace original et $\mathcal{F}$ l'espace de redescription) telle que :

$$\forall x, y \in \mathcal{X} \quad K(x,y) = \langle \varphi(x), \varphi(y) \rangle \tag{1}$$

Notons que la fonction φ associée à un noyau peut parfois être construite explicitement (par exemple pour les noyaux polynomiaux), parfois non (noyaux gaussiens, pour lesquels l'espace de redescription est de dimension infinie). Ainsi, tout algorithme qui utilise uniquement des produits scalaires peut se voir appliquer l'astuce du noyau, ce qui a pour effet de projeter, implicitement, l'espace original $\mathcal{X}$ dans un espace de redescription $\mathcal{F}$.

Cependant, cette astuce présente certaines limitations. Elle ne peut pas être appliqué à des algorithmes qui impliquent de calculer des transformations linéaires, de la forme suivante : $x' = \sum_{i=1}^{m} \lambda_i x_i$. Par exemple, pour l'algorithme des k-moyennes, des prototypes doivent être calculé comme étant le barycentre des données associées, c'est-à-dire informellement $x' = \frac{1}{m} \sum_{i=1}^{m} x_i$. Pour les cartes de Kohonen, un prototype c doit être déplacé vers une nouvelle entrée x, c'est-à-dire $c \leftarrow c + \lambda(x - c)$, où λ peut être compris comme un ratio de déplacement. Dans cet article, le problème que nous traitons est d'effectuer de telles transformations linéaires directement dans l'espace de redescription, de façon à pouvoir étendre tout type d'algorithme de quantification vectorielle à une version basée sur les noyaux.

Pour cela, une méthode de dictionnaire [2] est utilisée de façon à travailler directement dans une approximation de

l'espace de redescription $\mathcal{F}$, toujours sans utiliser φ explicitement. La section 2 présente un bref état de l'art. La section 3 introduit la méthode de dictionnaire, qui construit une représentation parcimonieuse de l'espace de redescription en employant un argument de dépendance linéaire approché. La méthode permettant de combiner noyaux et algorithmes de quantification vectorielle (KVQ pour *Kernel-based Vector Quantization*), précédemment introduite par [6], est présentée dans la section 4. Notons que tant des méthodes hors-ligne qu'en-ligne peuvent être envisagées. La section 5 l'illustre, et la section 6 conclut en proposant quelques perspectives.

2 Etat de l'art

Nous passons en revue certaines approches basées sur les noyaux permettant le partitionnement de données ou la quantification vectorielle. Deux types d'algorithmes sont considérés : ceux basés sur une métrique, et ceux basés sur l'espace de redescription. Le lecteur peut se référer à [3] pour un état de l'art plus complet.

Une distance dans l'espace de redescription peut être calculée grâce à l'astuce du noyau. Pour un noyau K et la fonction associée φ, la distance entre les images de deux éléments x et y de $\mathcal{X}$ peut être calculée en utilisant la bilinéarité du produit scalaire et l'astuce du noyau : $\|\varphi(x) - \varphi(y)\|^2 = K(x,x) - 2K(x,y) + K(y,y)$. Certaines approches utilisent cette métrique alternative pour effectuer directement le partitionnement des données dans l'espace de redescription, dans la mesure où seuls les produits scalaires et les distances sont nécessaires. Cette approche est utilisée par [14] pour appliquer l'astuce du noyau à l'algorithme des k-moyennes. Elle est également appliquée par [1] à l'algorithme du *growing neural gas* (GNG). Cependant, les prototypes sont tout de même déplacés dans l'espace ambiant $\mathcal{X}$ (et donc l'algorithme n'est pas totalement projeté dans l'espace de redescription). Une approche inspirée des k-moyennes et utilisant une optimisation stochastique est proposée par [7] et étendue par [12]. La méthode que nous proposons bénéficie de ce point de vue de la métrique, cependant elle s'applique à une plus large classe d'algorithmes.

Une autre approche est de travailler directement dans l'espace de redescription, ou du moins dans une approximation de ce dernier. Les algorithmes de l'état de l'art expriment généralement un point dans l'espace de redescription comme une combinaison linéaires des images de *toutes* les données, c'est-à-dire $\mathbf{y} = \sum_{i=1}^{n} a_i \varphi(x_i)$ où n est la taille de la base d'entraînement. Une mise à jour des poids de cette combinaison linéaire est ensuite déterminée, selon l'algorithme considéré. Il est à noter que la fonction φ n'est jamais explicitement utilisée. En utilisant cette idée, l'algorithme de la carte auto-organisatrice (SOM pour *self organizing map*) est combiné aux noyaux par [9] et [10], le *neural gaz* (NG) par [11] et un algorithme de partitionnement topographique flou par [8]. Cette approche peut (plus ou moins directement) être appliquée à une plus large classe d'algorithmes que la précédente, dans la mesure où elle permet de prendre en compte des transformations linéaires. Cependant, elle est algorithmiquement inefficace, dans la mesure où *toutes* les données sont considérées. La règle de mise à jour des poids de la combinaison linéaire doit être déterminée pour chaque algorithme. De plus, cette méthode n'est pas adaptée aux algorithmes en-ligne, pour lesquels les données ne sont pas connues à l'avance. La méthode que nous proposons dans cet article est proche de celles que nous venons de présenter, dans la mesure où un vecteur de l'espace de redescription est exprimé comme une combinaison linéaire d'images des données. Cependant, toutes les images ne sont pas nécessaires, et l'ensemble des points utilisé pour approcher l'espace de redescription est construit en-ligne. Notre contribution s'affranchit des difficultés que nous avons soulevées : elle est *générique* (il n'est pas nécessaire de déterminer une règle de mise à jour spécifique à chaque algorithme), elle peut être appliquée à des algorithmes hors-ligne et *en-ligne*, et elle est algorithmiquement *moins coûteuse*, dans la mesure où la représentation de l'approximation de l'espace de redescription maintenue est parcimonieuse.

3 Calcul du dictionnaire

Comme annoncé dans la section 1, l'astuce du noyau correspond à un produit scalaire dans un espace de plus grande dimension, associé à une fonction φ. Bien que l'espace de redescription $\mathcal{F}$ soit de grande dimension (si finie), l'ensemble $\varphi(\mathcal{X})$ peut être de taille plus raisonnable. En partant de cette intuition, l'objectif de la méthode que nous présentons maintenant est de trouver un ensemble de p points de $\mathcal{X}$ tel que $\varphi(\mathcal{X})$ soit approximativement inclus dans $\mathrm{Vect}\{\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)\}$ et que $\{\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)\}$ forme une famille indépendante [2].

La procédure est itérative. Supposons que les échantillons $x_1, x_2, \ldots$ soient séquentiellement observés. Au temps t, un dictionnaire $\mathcal{D}_{t-1} = (\tilde{x}_j)_{j=1}^{p_{t-1}} \subset (x_j)_{j=1}^{t-1}$ contenant p_{t-1} éléments est disponible, où par construction les vecteurs de redescription $\varphi(\tilde{x}_j)$ sont linéairement indépendants dans $\mathcal{F}$. L'échantillon x_t est ensuite observé, et il est ajouté au dictionnaire si $\varphi(x_t)$ est indépendant de $\mathcal{D}_{t-1}$. Pour cela, il faut déterminer s'il existe des poids $a = (a_1, \ldots, a_{p_{t-1}})^T$ permettant de vérifier l'inégalité suivante :

$$\| \sum_{j=1}^{p_{t-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_t) \|^2 \leq \nu \qquad (2)$$

Dans cette expression, ν est un seuil, choisi par l'utilisateur, qui détermine la qualité de l'approximation (et par conséquent la parcimonie de la représentation). Cela peut être déterminé en résolvant le problème suivant :

$$\delta_t = \min_{a \in \mathbb{R}^{p_{t-1}}} \| \sum_{j=1}^{p_{t-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_t) \|^2 \qquad (3)$$

Si $\delta_t > \nu$, les vecteurs de redescription sont indépendants, et $x_t = \tilde{x}_{p_t}$ est ajouté au dictionnaire, sinon ils sont ap-

proximativement dépendants et le dictionnaire n'est pas modifié. Notons qu'avec $\nu = 0$ la dépendance n'est plus approchée.

En utilisant la bilinéarité du produit scalaire, remplacé par une fonction noyau, et en définissant les matrices $\tilde{K}_{t-1}$ de taille $p_{t-1} \times p_{t-1}$ et $\tilde{k}_{t-1}(x)$ de taille $p_{t-1} \times 1$ par

$$(\tilde{K}_{t-1})_{i,j} = K(\tilde{x}_i, \tilde{x}_j) \quad (4)$$

$$\text{et } (\tilde{k}_{t-1}(x))_i = K(x, \tilde{x}_i) \quad (5)$$

le test de dépendance linéaire peut s'exprimer sous forme matricielle :

$$\delta_t = \min_{a \in \mathbb{R}^{p_{t-1}}} \{a^T \tilde{K}_{t-1} a - 2a^T \tilde{k}_{t-1}(x_t) + K(x_t, x_t)\} \quad (6)$$

Ce problème quadratique peut être résolu analytiquement, et sa solution est donnée par :

$$a_t = \tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}(x_t) \quad (7)$$

$$\text{et } \delta_t = K(x_t, x_t) - \tilde{k}_{t-1}(x_t)^T a_t \quad (8)$$

Si $\delta_t \leq \nu$, $\varphi(x_t)$ est approximativement dépendant de $\mathcal{D}_{t-1}$ et peut donc être écrit de la façon suivante :

$$\varphi(x_t) = \sum_{i=1}^{p_{t-1}} a_i \varphi(\tilde{x}_i) + \varphi_t^{res} \text{ avec } \|\varphi_t^{res}\| \leq \sqrt{\nu} \quad (9)$$

$$\approx \sum_{i=1}^{p_{t-1}} a_i \varphi(\tilde{x}_i) \quad (10)$$

Sinon, $\delta_t > \nu$ et $x_t = \tilde{x}_{p_t}$ est ajouté au dictionnaire (indépendance).

Sous cette forme, la complexité algorithmique du calcul du dictionnaire est $O(p_{t-1}^3)$ par itération, en raison de l'inversion de la matrice $\tilde{K}_t$. Cependant, cette dernière peut être calculée efficacement. Si $\delta_t \leq \nu$, aucun point n'est ajouté au dictionnaire et $\tilde{K}_t^{-1} = \tilde{K}_{t-1}^{-1}$. Si x_t est ajouté au dictionnaire, la matrice $\tilde{K}_t$ s'écrit par bloc, et grâce à la formule de l'inverse de la matrice partitionnée, son inverse peut être calculée incrémentalement :

$$\tilde{K}_t = \begin{pmatrix} \tilde{K}_{t-1} & \tilde{k}_{t-1}(x_t) \\ \tilde{k}_{t-1}(x_t)^T & K(x_t, x_t) \end{pmatrix} \quad (11)$$

$$\text{et } \tilde{K}_t^{-1} = \frac{1}{\delta_t} \begin{pmatrix} \delta_t \tilde{K}_{t-1}^{-1} + a_t a_t^T & -a_t \\ -a_t^T & 1 \end{pmatrix} \quad (12)$$

Cela réduit le coût algorithmique à $O(p_{t-1}^2)$ par itération. Plus de détails sur cette approche sont donnés par [2], l'algorithme 1 la résume et la figure 1 l'illustre. Dans le reste de cet article, le sous-ensemble de l'espace de redescription $\mathcal{F}$ engendré par la base $\varphi(\mathcal{D})$ sera noté $\tilde{\mathcal{F}}_{\mathcal{D}}$. Les variables en gras, comme $\mathbf{x}$, représenteront des éléments de l'espace engendré par les images des éléments du dictionnaire, et les variables classiques, comme x, représenterons des éléments de l'espace ambiant $\mathcal{X}$.

Algorithme 1 : Calcul du dictionnaire.

Initialisation;
$\mathcal{D}_1 = \{x_1\}$;
$\tilde{K}_1^{-1} = \frac{1}{K(x_1, x_1)}$;
$p_1 = 1$;

Calcul du dictionnaire;
pour $t = 2, \ldots N$ **faire**
  Observer l'échantillon x_t;
  Calcul de $\tilde{k}_{t-1}(x_t)$, voir équation (5) ;
  Test de dépendance linéaire approchée ;
  $a_t = \tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}(x_t)$;
  $\delta_t = K(x_t, x_t) - \tilde{k}_{t-1}(x_t)^T a_t$;
  si $\delta_t > \nu$ **alors**
    Ajouter x_t au dictionnaire : $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{x_t\}$;
    Calculer $\tilde{K}_t^{-1}$, voir équation (12) ;
    $p_t = p_{t-1} + 1$;
  sinon
    Laisser le dictionnaire inchangé $\mathcal{D}_t = \mathcal{D}_{t-1}$;
    $\tilde{K}_t^{-1} = \tilde{K}_{t-1}^{-1}$;
    $p_t = p_{t-1}$;

4 Quantification vectorielle dans l'espace de redescription

Supposons qu'une base $\varphi(\mathcal{D}) = \{\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)\}$ soit disponible. Rappelons que les points $\tilde{x}_1, \ldots, \tilde{x}_p$ sont explicitement connus, mais pas leurs images $\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)$. Chaque vecteur de redescription $\varphi(x) \in \mathcal{F}$, avec $x \in \mathcal{X}$, peut être approximativement exprimé dans cette base. Il existe un vecteur $a^{\mathbf{x}} = (a_1^{\mathbf{x}}, \ldots, a_p^{\mathbf{x}})^T$ tel que :

$$\varphi(x) \approx \mathbf{x} = \sum_{i=1}^{p} a_i^{\mathbf{x}} \varphi(\tilde{x}_i) \quad (13)$$

Comme nous l'avons vu section 3, le vecteur $a^{\mathbf{x}}$ peut être facilement calculé. Le principe de la contribution que nous proposons est de directement travailler dans $\tilde{\mathcal{F}}_{\mathcal{D}}$, l'image $\varphi(x)$ d'un point $x \in \mathcal{X}$ étant représentée par ses coefficients $a^{\mathbf{x}}$ dans la base, si nécessaire.

Nous commençons par faire une remarque importante pour la suite : si $\tilde{\mathcal{F}}_{\mathcal{D}}$ est engendré par des images de points de $\mathcal{X}$, tout point de $\tilde{\mathcal{F}}_{\mathcal{D}}$ n'a pas forcément d'antécédent dans $\mathcal{X}$. En effet, toute combinaison des vecteurs $\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)$ peut être envisagé. Pour un vecteur de redescription spécifique $\mathbf{y} \in \tilde{\mathcal{F}}_{\mathcal{D}}$, qui peut être écrit

$$\mathbf{y} = \sum_{i=1}^{p} a_i^{\mathbf{y}} \varphi(\tilde{x}_i) \quad (14)$$

il ne peut être affirmé qu'il existe un point $y \in \mathcal{X}$ tel que $\varphi(y) = \mathbf{y}$, par exemple si $\mathbf{y}$ est une combinaison linéaire d'images de données de la base d'entraînement. Ceci est illustré sur la figure 1.

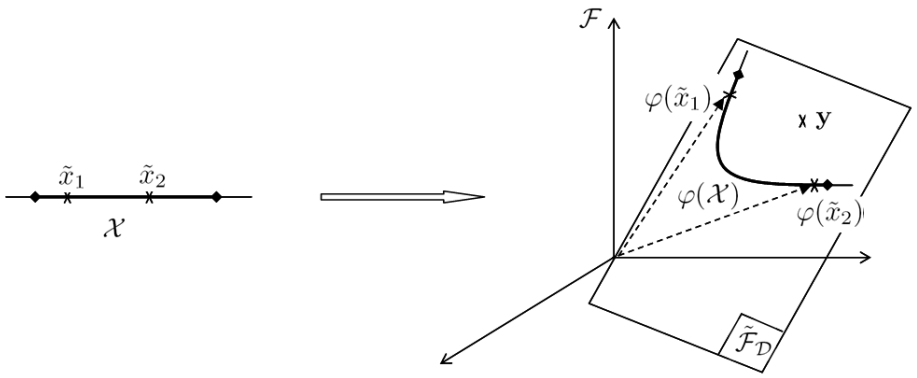


FIG. 1 – Le dictionnaire $\mathcal{D} = \{\tilde{x}_1, \tilde{x}_2\}$ est construit à partir de $\mathcal{X}$. L'image de ce dictionnaire $\varphi(\mathcal{D})$ est utilisé pour engendrer $\tilde{\mathcal{F}}_{\mathcal{D}}$. L'image $\varphi(\mathcal{X})$ de l'espace ambiant $\mathcal{X}$ est approximativement inclus dans $\tilde{\mathcal{F}}_{\mathcal{D}}$, dans le sens où la distance entre $\varphi(\mathcal{X})$ et $\tilde{\mathcal{F}}_{\mathcal{D}}$ est d'au plus $\sqrt{\nu}$. Le point $\mathbf{y} \in \tilde{\mathcal{F}}_{\mathcal{D}}$ est une combinaison linéaire de $\varphi(\tilde{x}_1)$ et $\varphi(\tilde{x}_2)$, cependant il n'a pas d'antécédent dans $\mathcal{X}$.

L'astuce classique du noyau s'applique aux algorithmes pour lesquels seuls des produits scalaires doivent être calculés, et elle est étendue ici aux algorithmes nécessitant également de calculer des transformations linéaires. Nous montrons maintenant comment réaliser ces deux opérations élémentaires dans le cadre de travail posé.

4.1 Calcul de produit scalaire

Supposons que l'algorithme original requiert de calculer un produit scalaire entre deux points x et y de $\mathcal{X}$. En appliquant l'astuce du noyau, cela revient à calculer $\langle \varphi(x), \varphi(y) \rangle = K(x, y)$. Si x et y sont connus, il est possible de calculer directement $K(x, y)$. Mais rappelons que l'approche proposée consiste à travailler directement dans l'espace engendré par $\varphi(\mathcal{D})$. Il est donc possible que seuls $\mathbf{x}$ et $\mathbf{y}$ de $\tilde{\mathcal{F}}_{\mathcal{D}}$ soient connus. Par exemple, ces vecteurs de redescription peuvent être une combinaison d'éléments de $\tilde{\mathcal{F}}_{\mathcal{D}}$, et comme nous l'avons expliqué il n'y a aucune raison *a priori* qu'ils aient des antécédents dans $\mathcal{X}$. Nous pouvons exprimer ces vecteurs de redescription dans la base $\varphi(\mathcal{D})$:

$$\mathbf{x} = \sum_{i=1}^{p} a_i^{\mathbf{x}} \varphi(\tilde{x}_i) \tag{15}$$

$$\text{et } \mathbf{y} = \sum_{i=1}^{p} a_i^{\mathbf{y}} \varphi(\tilde{x}_i) \tag{16}$$

Le produit scalaire se calcule alors de la façon suivante :

$$\begin{aligned} \langle \mathbf{x}, \mathbf{y} \rangle &= \langle \sum_{i=1}^{p} a_i^{\mathbf{x}} \varphi(\tilde{x}_i), \sum_{i=1}^{p} a_i^{\mathbf{y}} \varphi(\tilde{x}_i) \rangle \\ &= \sum_{i,j=1}^{p} a_i^{\mathbf{x}} a_j^{\mathbf{y}} K(\tilde{x}_i, \tilde{x}_j) \\ \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle &= (a^{\mathbf{x}})^T \tilde{K} a^{\mathbf{y}} \end{aligned} \tag{17}$$

où $\tilde{K}$ est défini section 3. Si $\mathbf{x}$ appartient à l'espace engendré par $\varphi(\mathcal{D})$ et y à l'espace ambiant, alors le produit scalaire entre $\mathbf{x}$ et $\varphi(y)$ se calcule comme suit :

$$\begin{aligned} \langle \mathbf{x}, \varphi(y) \rangle &= \langle \sum_{i=1}^{p} a_i^{\mathbf{x}} \varphi(\tilde{x}_i), \varphi(y) \rangle \\ &= \sum_{i=1}^{p} a_i^{\mathbf{x}} K(\tilde{x}_i, y) \\ \Leftrightarrow \langle \mathbf{x}, \varphi(y) \rangle &= (a^{\mathbf{x}})^T \tilde{k}(y) \end{aligned} \tag{18}$$

où $\tilde{k}$ est défini section 3. Rappelons que calculer un distance revient à évaluer un produit scalaire, et donc :

$$\|\mathbf{x} - \mathbf{y}\|^2 = (a^{\mathbf{x}} - a^{\mathbf{y}})^T \tilde{K} (a^{\mathbf{x}} - a^{\mathbf{y}}) \tag{19}$$

$$\|\mathbf{x} - \varphi(y)\|^2 = (a^{\mathbf{x}})^T \tilde{K} a^{\mathbf{x}} - 2(a^{\mathbf{x}})^T \tilde{k}(y) + K(y, y) \tag{20}$$

L'approche proposée n'est donc pas restrictive et permet toujours de calculer des distances et des produits scalaires, comme l'astuce du noyau standard. Cependant, il est possible qu'un point de $\tilde{\mathcal{F}}_{\mathcal{D}}$ n'ait pas d'antécédent, ce qui implique plus de calculs que la simple application de la fonction noyau.

4.2 Calcul de transformation linéaire

Supposons que l'algorithme original requiert de calculer une transformation linéaire :

$$x' = \sum_{i=1}^{m} \lambda_i x_i \tag{21}$$

Pour être consistant avec l'astuce du noyau, pour projeter ce type d'algorithme dans l'espace de redescription il faut calculer un point $\mathbf{x}' \in \mathcal{F}$ tel que

$$\mathbf{x}' = \sum_{i=1}^{m} \lambda_i \mathbf{x}_i \tag{22}$$

avec $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \tilde{\mathcal{F}}_{\mathcal{D}}$ et en utilisant les mêmes coefficients $(\lambda_i)_{i=1}^m$. Il existe une façon simple de faire cela directement dans l'espace de redescription approché

Comme on ne considère pas plus que des combinaisons linéaires, chaque vecteur $\mathbf{x}_i$ peut s'exprimer comme :

$$\mathbf{x}_i = \sum_{j=1}^{p} a_j^{\mathbf{x}_i} \varphi(\tilde{x}_j) \tag{23}$$

Il est alors assez direct d'exprimer $\mathbf{x}'$ dans la même base :

$$\begin{aligned} \mathbf{x}' &= \sum_{i=1}^{m} \lambda_i (\sum_{j=1}^{p} a_j^{\mathbf{x}_i} \varphi(\tilde{x}_j)) \\ \Leftrightarrow \mathbf{x}' &= \sum_{j=1}^{p} (\sum_{i=1}^{m} \lambda_i a_j^{\mathbf{x}_i}) \varphi(\tilde{x}_j) \end{aligned} \tag{24}$$

Autrement dit, $\mathbf{x}'$ est associé au vecteur de coordonnées

$$\begin{aligned} a^{\mathbf{x}'} &= (\sum_{i=1}^{m} \lambda_i a_1^{\mathbf{x}_i}, \ldots, \sum_{i=1}^{m} \lambda_i a_p^{\mathbf{x}_i})^T \\ \Leftrightarrow a^{\mathbf{x}'} &= A\lambda \end{aligned} \tag{25}$$

où $A = (a^{\mathbf{x}_1}, \ldots, a^{\mathbf{x}_m})$ est une matrice de taille $p \times m$ et $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ un vecteur de taille $m \times 1$.
Le fait d'approximer l'espace de redescription permet ainsi de prendre en compte les transformations linéaires. Grâce à cette approche, tout algorithme de quantification vectorielle (qui n'implique que des produits scalaires et des transformations linéaires) peut très simplement être projeté dans l'espace de redescription. Notons que cette méthode peut être appliquée à des algorithmes hors-ligne et en-ligne. Pour les algorithmes hors-ligne, le dictionnaire peut être calculé à partir de la base d'entraînement ou d'un ensemble compact l'incluant. Pour les algorithmes en-ligne, le dictionnaire peut être calculé soit au fil de l'observation de nouveaux échantillons, soit dans une phase de prétraitement à partir d'un ensemble compact englobant l'espace de travail. Nous résumons cette contribution dans l'algorithme 2.

Algorithme 2 : Projection des algorithmes de quantification vectorielle dans l'espace de redescription.

Calculer le dictionnaire $\mathcal{D}$ et les matrices $\tilde{K}$ et $\tilde{K}^{-1}$, *à partir* des échantillons disponibles ou d'un ensemble compact les incluant, *dans* une phase de prétraitement ou en-ligne ;

Remplacer chaque échantillon x par le vecteur associé $a^{\mathbf{x}} = \tilde{K}^{-1}\tilde{k}(x)$ de $\mathbf{x} = \varphi(x)$ exprimé dans la base $\varphi(\mathcal{D})$;

Remplacer chaque produit scalaire $\langle x, y \rangle$ par le produit scalaire $(a^{\mathbf{x}})^T \tilde{K} a^{\mathbf{y}}$. Utiliser directement K ou $\tilde{k}$ si possible (voir section 4.1) ;

Remplacer chaque distance $\|x - y\|^2$ par la distance $(a^{\mathbf{x}} - a^{\mathbf{y}})^T \tilde{K} (a^{\mathbf{x}} - a^{\mathbf{y}})$. Utiliser directement K ou $\tilde{k}$ si possible (voir section 4.1) ;

Remplacer chaque transformation linéaire $y = (x_1, \ldots, x_m)(\lambda_1, \ldots, \lambda_m)^T$ par la transformation linéaire correspondante $a^{\mathbf{y}} = (a^{\mathbf{x}_1}, \ldots, a^{\mathbf{x}_m})(\lambda_1, \ldots, \lambda_m)^T$;

5 Illustration de l'approche

Nous nous proposons dans cette section d'illustrer cette méthode en l'appliquant à deux algorithmes de quantification vectorielle, à savoir l'algorithme des k-moyennes et GNG-T [5]. Le premier a un but pédagogique, et illustre l'application de notre méthode, et le second a pour objectif de montrer l'effet de différents noyaux sur la quantification, via la représentation des cellules de Voronoï associés.

5.1 k-moyennes

Nous nous intéressons ici à l'algorithme standard des k-moyennes. Nous dérivons l'algorithme original ainsi que sa transformation obtenue par la méthode proposée. Nous illustrons ensuite l'algorithme transformé sur un problème jouet, non linéairement séparable.

Algorithme original. L'algorithme des k-moyennes est un simple algorithme de quantification vectorielle qui utilise un critère d'erreur quadratique comme mesure de distortion. Il est initialisé avec une partitionnement aléatoire, et réassigne continuellement les données aux partitions en se basant sur la distance entre les données et le centre de la partition (également appelé prototype), ce jusqu'à ce qu'un critère de convergence soit atteint. Il peut être résumé comme suit :

1. choix de k prototypes arbitraires ;
2. associer chaque donnée au prototype le plus proche (distance euclidienne) ;
3. recalculer les prototypes comme barycentres des données associés. Autrement dit, si les points $\check{x}_1, \ldots, \check{x}_{N_j}$ sont associé à la $j^{\text{ème}}$ partition, alors le nouveau prototype est le suivant :

$$c_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \check{x}_i \tag{26}$$

4. réitérer les deux précédentes étapes jusqu'à la convergence.

Algorithme transformé. Nous transformons l'algorithme des k-moyennes en appliquant la méthode résumé dans l'algorithme 2. Un ensemble de N échantillons $X = \{x_1, \ldots, x_N\}$ doit être partitionné en k groupes. Dans un premier temps le dictionnaire est calculé à partir de X. Les vecteurs de coefficients associés aux échantillons doivent également être déterminé, c'est-à-dire qu'il faut calculer la matrice $A_N = (a^{\mathbf{x}_1}, \ldots, a^{\mathbf{x}_N})$ (notons qu'en pratique, cette matrice est déterminé pendant le calcul du dictionnaire). L'algorithme transformé peut être résumé comme suit :

1. choix de k prototypes arbitraires ;
2. associer chaque donnée au prototype le plus proche (dans l'espace de redescription). C'est-à-dire que $\forall i \in [1, N]$, le point $x_i \in \mathcal{X}$, dont l'image dans l'espace de redescription est approchée par $\mathbf{x}_i$, est assignée à la partition j de centre $\mathbf{c}_j \in \tilde{\mathcal{F}}_{\mathcal{D}}$ qui vérifie $\forall j' \neq j$

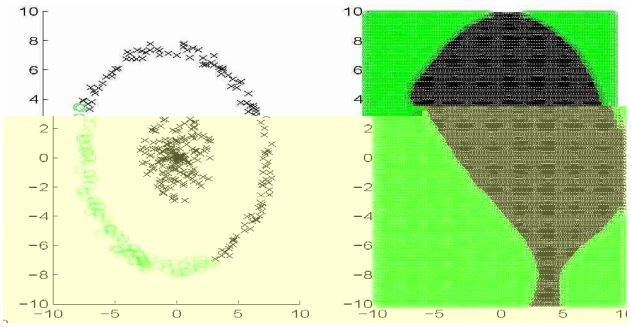
$$(a^{\mathbf{x}_i} - a^{\mathbf{c}_j})^T \tilde{K} (a^{\mathbf{x}_i} - a^{\mathbf{c}_j}) < (a^{\mathbf{x}_i} - a^{\mathbf{c}_{j'}})^T \tilde{K} (a^{\mathbf{x}_i} - a^{\mathbf{c}_{j'}}) \quad (27)$$

3. Calculer les nouveaux prototypes comme étant les barycentres des données associées à la partition. Si à la partition j sont associés les points $\check{x}_1, \ldots, \check{x}_{N_j}$, le nouveau prototype $\mathbf{c}_j$ est calculé comme suit :

$$a^{\mathbf{c}_j} = \frac{1}{N_j} \sum_{i=1}^{N_j} a^{\check{\mathbf{x}}_j} \quad (28)$$

4. réitérer les deux précédentes étapes jusqu'à la convergence.

Il est à noter que l'algorithme des k-moyennes peut être exprimé uniquement en termes de distances et de produits scalaires [14], et l'approche que nous proposons n'est donc pas nécessaire pour le projeter dans l'espace de redescription. Cependant l'objectif ici était de démontrer la faisabilité et la facilité d'application de notre contribution sur un exemple simple.



a. Initialisation possible.

b. Convergence typique.

FIG. 2 – Test synthétique pour la transformation des k-moyennes.

Expérimentation. Pour illustrer l'algorithme modifié nous proposons quelques résultats sur un exemple synthétique. Deux partitions concentriques de 200 points chacune sont générées (tirage uniforme), voir figure 2. Notons que ce problème n'est pas linéairement séparable. Un noyau gaussien est choisi :

$$K(x, y) = \exp\left(-\frac{(x-y)^T(x-y)}{2\sigma^2}\right) \quad (29)$$

L'écart-type est choisi égal à $\sigma = 3,5$. Le facteur de parcimonie pour le dictionnaire est choisi égal à $\nu = 0,3$. Le dictionnaire correspondant contient une dizaine d'éléments. Nous considérons $k = 2$ partitions.
Une initialisation est montrée sur la figure 2.a, et une convergence typique (requérant moins d'une dizaine d'itérations) sur la figure 2.b. La partie gauche de chaque figure montre l'association des données aux partitions, et la partie droite les cellules de Voronoï correspondantes. A la convergence, les centres des partitions n'ont pas d'antécédent dans $\mathbb{R}^2$.

5.2 GNG-T

Nous appliquons également la méthode proposée à l'algorithme de quantification vectorielle GNG-T, proposé par [5]. La transformation de l'algorithme n'est pas plus compliquée que dans le cas des k-moyennes, et nous ne la dérivons pas ici. L'objectif est d'appliquer GNG-T à des données synthétiques en forme d'anneau (les points noirs sur la figure 3) de façon à observer l'influence que peuvent avoir différents noyaux sur la quantification (cette influence étant comprise via les cellules de Voronoï associées).

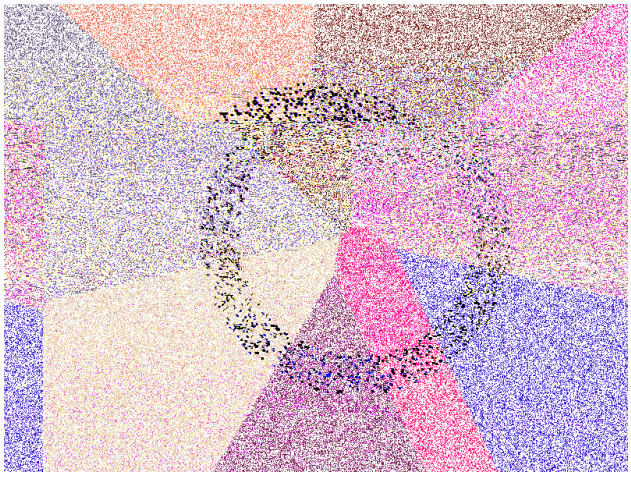


FIG. 3 – Produit scalaire euclidien.

Produit scalaire usuel. Dans un premier temps nous considérons comme noyau le produit scalaire usuel :

$$K(x, y) = x^T y \quad (30)$$

En conséquence, c'est l'algorithme GNG-T original qui est obtenu, et la quantification résultante est illustrée sur la figure 3. Les cellules de Voronoï obtenues sont classiques, avec un partitionnement régulier.

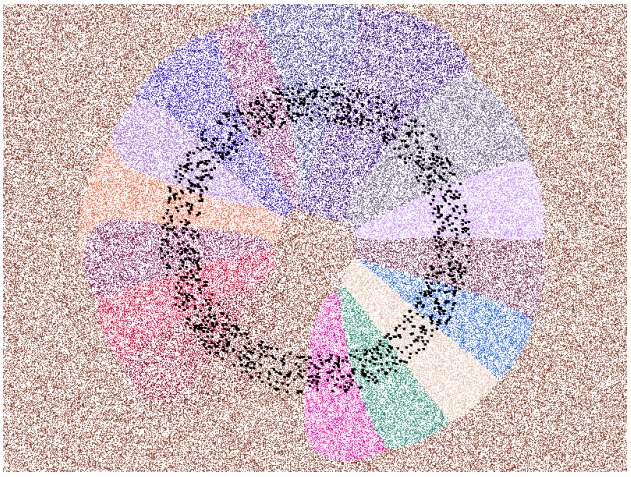


FIG. 4 – Noyau gaussien.

Noyau gaussien. Nous utilisons un noyau gaussien pour générer la figure 4 :

$$K(x,y) = \exp\left(-\frac{(x-y)^T(x-y)}{2\sigma^2}\right) \quad (31)$$

La structure générale des cellules de Voronoï est similaire à celle obtenue pour le produit scalaire classique, le noyau gaussien étant local. Notons tout de même que le centre et l'extérieur appartiennent à la même cellule, ce qui met en lumière les résultats des k-moyennes.

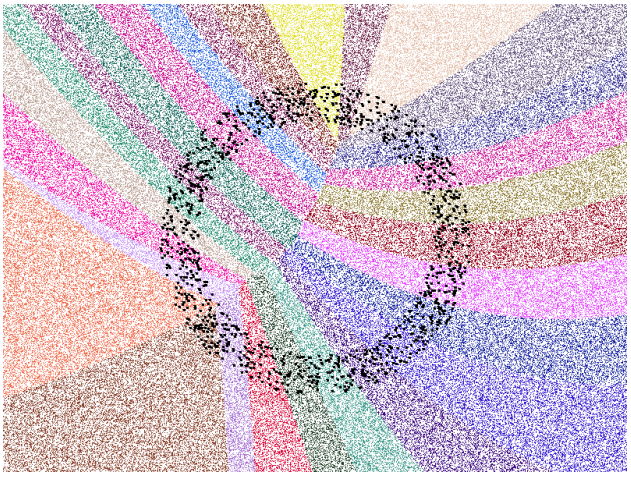


FIG. 5 – Noyau sigmoïdal.

Noyau sigmoïdal. Un noyau sigmoïdal est utilisé pour générer la figure 5 :

$$K(x,y) = \tanh\left(a\left(x^T y\right) + b\right) \quad (32)$$

La structure des cellules de Voronoï est moins régulière, et caractéristique du noyau utilisé.

Noyau gaussien-sigmoïdal. Le dernier noyau que nous considérons, qui sert à générer la figure 6, est le noyau gaussien-sigmoïdal :

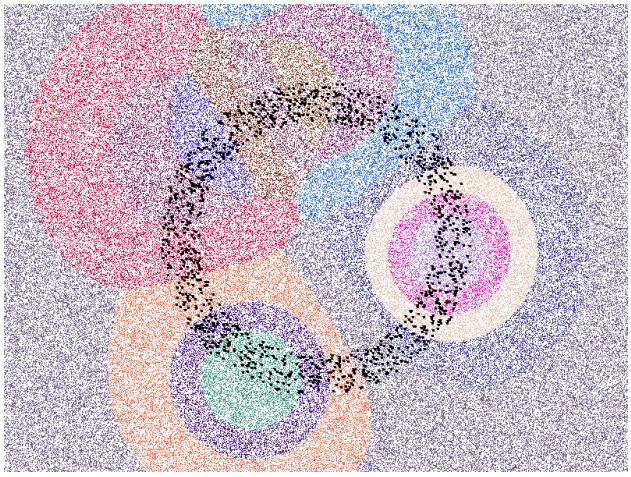


FIG. 6 – Noyau gaussien-sigmoïdal.

$$K(x,y) = \tanh\left(a\exp\left(-\frac{(x-y)^T(x-y)}{2\sigma^2}\right) + b\right) \quad (33)$$

La structure des cellules de Voronoï est beaucoup moins régulière qu'auparavant, et assez particulière.

Nous avons présenté ces différentes expérimentations pour illustrer l'effet de l'utilisation d'un noyau pour la quantification vectorielle, il n'y a pas de conclusion particulière à en tirer, l'objectif étant surtout pédagogique. Nous proposons quelques perspectives d'utilisation de la quantification vectorielle projetée dans l'espace de redescription dans la section suivante.

6 Conclusion

Nous avons proposé une nouvelle approche générique permettant d'étendre l'applicabilité de l'astuce du noyau aux algorithmes nécessitant non seulement de calculer des produits scalaires, mais également des transformations linéaires, et ce qu'ils soient hors-ligne ou en-ligne. Bien que les résultats obtenus soient approchés (dans la mesure où l'on travaille dans une approximation de l'espace de redescription), cette contribution est une extension de l'astuce du noyau qui peut s'avérer intéressante dans le cadre de la quantification vectorielle. Comparé aux articles passés en revue dans la section 2, l'approche proposée permet de prendre en compte des transformations linéaires (contrairement aux approches qui ne considèrent que des produits scalaires et des distances) et elle est générique (comparé aux autres approches basées directement sur l'espace de redescription). Elle peut être appliqué aux algorithmes hors-ligne et en-ligne, dans la mesure où la construction du dictionnaire est incrémentale. Comparé aux autres méthodes travaillant directement dans l'espace de redescription, elle est plus efficace en terme de complexité (car une représentation parcimonieuse est considérée). De plus, le compromis entre parcimonie et qualité de l'approxima-

tion peut être contrôlée grâce au facteur ν. Un travail est en cours pour appliquer des algorithmes de quantification vectorielle transformés en phase de prétraitement à l'application d'une SVM (*Support Vector Machine*), l'idée étant de réduire la taille de la base d'entraînement sans perte du pouvoir de généralisation. Une autre perspective intéressante est de projeter dans l'espace de redescription des algorithmes de quantification vectorielle utilisés dans un but de compression, qui trouvent leur application en codage de la parole par exemple.

Remerciements

Les auteurs souhaitent remercier Hervé Frezza-Buet pour le développement de la libraire KVQ en C++ [4].

Références

[1] L. D'AMATO, J. A. MORENO et R. MUJICA : Reducing the Complexity of Kernel Machines with Neural Growing Gas in Feature Space. *In IBERAMIA*, pages 799–808, 2004.

[2] Y. ENGEL, S. MANNOR et R. MEIR : The Kernel Recursive Least Squares Algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, 2004.

[3] M. FILIPPONE, F. CAMASTRA, F. MASULLI et S. ROVETTA : A survey of kernel and spectral methods for clustering. *Pattern Recogn.*, 41(1):176–190, 2008.

[4] H. FREZZA-BUET : KVQ C++ library. http ://ims.metz.supelec.fr/spip.php ?article87.

[5] H. FREZZA-BUET : Following non-stationary distributions by controlling the vector quantization accuracy of a growing neural gas network. *Neurocomputing*, 71(7-9):1191–1202, 2008.

[6] Matthieu GEIST, Olivier PIETQUIN et Gabriel FRICOUT : Kernelizing Vector Quantization Algorithms. *In European Symposium on Artificial Neural Networks (ESANN 09)*, Bruges, Belgique, 2009.

[7] M. GIROLAMI : Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, May 2002.

[8] T. GRAEPEL et K. OBERMAYER : Fuzzy Topographic Kernel Clustering. *In* W. BRAUER, éditeur : *5th GI Workshop Fuzzy Neuro Systems 98*, pages 90–97, 1998.

[9] R. INOKUCHI et S. MIYAMOTO : LVQ Clustering and SOM Using a Kernel Function. *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, 17(1):88–94, 2005.

[10] D. MACDONALD et C. FYFE : A new kernel clustering algorithm. *In International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, volume 1, pages 317–320, 2000.

[11] A. K. QIN, et P. N. SUGANTHAN : Kernel Neural Gas Algorithms with Application to Cluster Analysis. *In 17th International Conference on Pattern Recognition*, volume 4, pages 617–620, Washington, DC, USA, 2004. IEEE Computer Society.

[12] D. S. SATISH et C. C. SEKHAR : Kernel based clustering and vector quantization for speech recognition. *In 14th IEEE Signal Processing Society Workshop*, pages 315–324, 2004.

[13] V. N. VAPNIK : *Statisical Learning Theory*. John Wiley & Sons, Inc., 1998.

[14] S. VISHWANATHAN et Narasimha M. MURTY : Kernel Enabled K-Means Algorithm. Rapport technique, National ICT Australia, Statistical Machine Learning Program, 2002.