Méthodes à noyaux: théorie, implémentation et utilisation

Vincent KUBICKI - InriaTech

Inria Lille - Nord Europe

22 Juin 2018

Généralités

- Évolution de méthodes standards
- Implémentation personnelle pour le contrat Coreye.
- Remplacement de l'outil actuel en C archaïque $\mapsto$ Scala.
- Démonstration de faisabilité sur la factorisation des méthodes à noyaux.

Pourquoi ? (1 / 4 maths)

- Algos ont besoin d'opérations algébriques
- Opérations algébriques $\mathcal{X} = \mathbf{R}$: $2 \times 2 = 4$
- Produit scalaire $\mathcal{X} = \mathbf{R}^2$: $[1, 2] \cdot [3, 4] = 11$
- Pb 1 : $\mathcal{X} = \text{ChaînesCar}$: toto $\times$ piscine $=$???
- Pb 2 : produit scalaire inadapté :

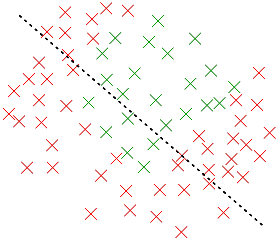

Figure 1 – Pas de séparation linéaire

Comment ? (2 / 4 maths)

- Fonction $f : \mathcal{X} \to \mathbb{R}$
- RKHS $\mathcal{H}$ avec évaluation ponctuelle $f(x) = \langle f, K_x \rangle_{\mathcal{H}}$
- Définition noyau :

$$\begin{aligned} k :& (\mathcal{X}, \mathcal{X}) \to \mathbb{R} \\ & (x_1, x_2) \mapsto \langle K_{x_1}, K_{x_2} \rangle_{\mathcal{H}} \end{aligned}$$

- Application d'un noyau :
  - $\mathcal{X} = \mathbf{R}^2$: $k([1, 2], [3, 4]) = -12.2345$
  - $\mathcal{X} = \text{ChaînesCar}$: $k(\text{toto}, \text{piscine}) = 72.3$

Entracte graphique 1

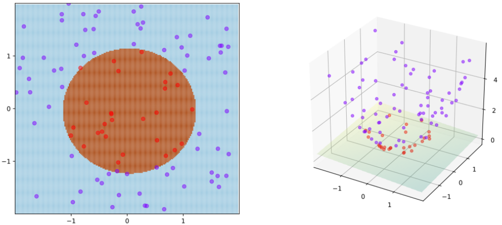

Figure 2 – $k(x_1, x_2) = \langle x_1, x_2 \rangle + ||x_1||^2||x_2||^2$

Entracte graphique 2

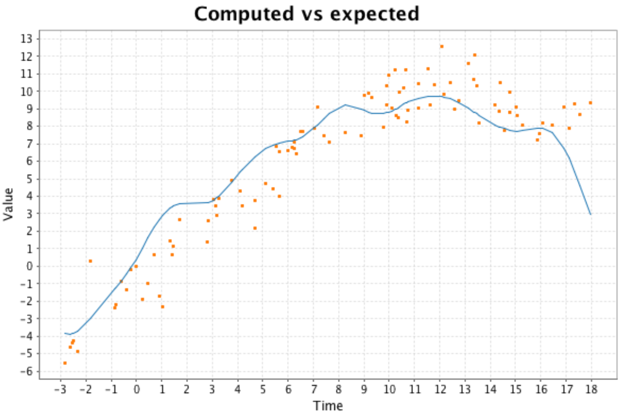

Figure 3 – $k(x_1, x_2) = \exp\left(-\frac{\|\mathbf{x_1} - \mathbf{x_2}\|^2}{2\sigma^2}\right)$

Conclusion de l'entracte

Complexifier l'espace d'étude, pas l'algorithme.

Génération de caractéristiques (3 / 4 maths)

- Le noyau "crée" les caractéristiques.
- Espace de dimension infinie.

Explicite, manuelle

$$\text{"po po po..."} \mapsto \begin{bmatrix} \text{moy. taille} \\ \text{n mots} \\ \vdots \\ \text{app. dico} \end{bmatrix}$$

Implicite, avec noyaux

$$\text{"po po po..."} \mapsto \begin{bmatrix} 12.0 \\ 25.3 \\ -14.5 \\ \vdots \end{bmatrix}$$

Unification via matrice de Gram (4 / 4 maths)

$$\sqrt{k(x_1, x_1) + k(x_2, x_2) - 2 \cdot k(x_1, x_2)}$$

$$\text{données} \mapsto \begin{array}{|c|c|c|} \hline k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \hline k(x_2, x_1) & \cdots & k(x_2, x_n) \\ \hline \ldots & \ddots & \ldots \\ \hline k(x_n, x_1) & \cdots & k(x_n, x_n) \\ \hline \end{array} \mapsto \text{algorithmes spécialisés}$$

Modularité

Commun

- Entrées / Sorties
- Matrice de Gram
- KerEval
- ...

Données

- Réels
- Vecteurs
- Matrices
- ...

Noyaux

- Linéaire
- Polynomial
- Gaussien
- Laplacien
- ...

Algorithmes

- Régression
- Classification non supervisée
- Test de distribution
- Segmentation
- Classification supervisée
- ...

Exemple de données

algo	lambda	cacheGram
regression	1	true

algo.csv

commun

CRIM	ZN	INDUS	CHAS	NC
1	1	1	1	
Gaussian(1.0)	Gaussian(1.0)	Gaussian(1.0)	Gaussian(1.0)	Ga

desc.csv

CRIM	ZN	INDUS	CHAS	NOX	RM
Real	Real	Real	Real	Real	Re
8.71675	0	18.1	0	0.693	6.
0.54452	0	21.89	0	0.624	6.
0.04294	28	15.04	0	0.464	6.
13.3598	0	18.1	0	0.693	5.
0.7857	20	3.97	0	0.647	7.
0.10008	0	2.46	0	0.488	6.

learnData.csv

13.1
17.8
20.6
12.7
30.7
32.5
41.3
36.1
16.8
14.1

learnY.csv

apprentissage →

-1.56128968812002
-1.30405810761556
1.42735075249551
0.24075317886306
0.26312805493215
1.12719368051962
5.15710781286711
2.77621160886438
-2.57371331942236
1.4119343387515
-0.625652088534706
-0.536980693991607

paramBeta.csv

CRIM	ZN	INDUS	CHAS	NOX	RM
Real	Real	Real	Real	Real	Rea
15.5757	0	18.1	0	0.58	5.
0.31533	0	6.2	0	0.504	8.
0.03961	0	5.19	0	0.515	6.
7.05042	0	18.1	0	0.614	6.
9.18702	0	18.1	0	0.7	5.
0.66351	20	3.97	0	0.647	7.
11.9511	0	18.1	0	0.659	5.
24.8017	0	18.1	0	0.693	5.
0.32264	0	21.89	0	0.624	5.
15.1772	0	18.1	0	0.74	6.
15.8603	0	18.1	0	0.679	5.

predictData.csv

prédiction →

15.1064684421859
42.5320832088463
18.9696827605349
10.6815746076759
13.9322788787389
38.5922421652867
21.4955501233267
9.66534219006098
17.3325227943665
7.73577452615764
7.71931546258032
16.3074536511108
18.3408804879515

predictY.csv

Similaire / MixtComp

- Génération de modèles à partir de briques de base.
- Généralisation de codes au coup par coup des chercheurs.
- Données complexes sans plongement explicite.
- Plus lent mais plus précis que les méthodes de base, type régressions linéaires / logistiques.

Avantages / MixtComp

- Algorithmes multiples, adaptés à chaque tâche.
- Algorithmes non stochastiques, convergence plus rapide, moins de cas limites.
- API 1 vs 16 par "modèle".
- Pas de modèle par variable $\mapsto$ moins de dégénérescence.

Inconvénients / MixtComp

- Poids et paramètres de noyau.
- Pas de données manquantes (pour le moment).
- Algorithmes de base fortement consommateurs en temps et mémoire.
- Pas d'outil d'analyse génériques (à voir algo par algo).

La vie d'une ville en noyaux

- Régression : prix d'une maison.
- Class.non sup. : groupes de maisons similaires.
- Test de dist. : groupes appartiennent à même zone ?
- Segmentation : quartier s'embourgeoise ?
- Class. supervisée : maison vendue en moins de trois mois ?

Conclusion

- Apprentissage / prédiction $\mapsto$ démonstration de faisabilité complète.
- À venir : tests sur données réelles et optimisations.
- Structures algébriques et programmation fonctionnelles.
- Potentiel important / faibles moyens déployés pour le moment.