SCIENCES
SORBONNE
UNIVERSITÉ

ML -- LASSO & Inpainting

Zixuan FENG

Arnaud DELOL

14/06/2020

I. Préambule : régression linéaire, régressin ridge et LASSO

Dans cette section, nous avons utilisé le jeu de données USPS pour tester la régression linéaire, la régression ridge et l'algorithme du LASSO.

Ce jeu de données contient des images de 10 classes (0-9) et nous avons besoin de deux classes. Nous avons pris la classe 0 et la classe 1. Pour trouver le meilleur alpha pour la régression ridge et l'algorithme LASSO, nous avons utilisé les fonctions *RidgeCV* et *LassoCV* du module *sklearn.linear_model*.

Voici notre résultat :

	Régression linéaire	*Régression ridge*	*Algo LASSO*
alpha		0.01	0.000344896226040576
score	0.9844401527577691	0.9844216685577594	0.9828869896234311
norme	40636267009.58993	0.6514994215590485	0.17193589018122418
nb_non_null	255	254	107

norme : norme du vecteur de poids
nb_non_null : nombre de composantes non nulles du vecteur de poids

En analysant le résultat, nous pouvons savoir :

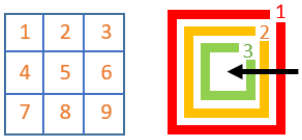
- **score**: linéaire>ridge>LASSO, mais la différence n'est pas évidente.
  --> Il est facile de distinguer les 2 classes dans ce jeu de données, donc la régression linéaire a une bonne performance.
- **norme du vecteur de poids:** linéaire>>ridge>LASSO
  --> C'est important de faire la régularisation sur la régression linéaire
- **nombre de composantes non nulles:** linéaire>ridge>>LASSO
  --> Le vecteur de poids en utilisant l'algo de LASSO est plus "sparse".

L'algorithme LASSO a une bonne performance en régularisation et il peut bien sparser le vecteur de poids.

II. LASSO et Inpainting

Inspirés par les articles, nous avons proposé trois façons de réparer des images :

- random_fill_img :
  On répare l'image en ordre aléatoire.
- simple_fill_img :
  On répare l'image de gauche à droite et de haut en bas.
- better_fill_img :
  On répare d'abord les patchs qui contiennent le plus de pixels exprimés.



simple better

Nous avons fait plusieurs tests sur « lena.jpg » et « beach.jpg », deux images de taille 512*512 pixels.

D'abord on ajoute des bruits aléatoirement avec la probabilité=0,005 et utilise simple_fill_img pour la réparer :

Lena bruité (proba=0.005)

Lena réparé

Ensuite, nous avons fait un bruit de forme rectangulaire :

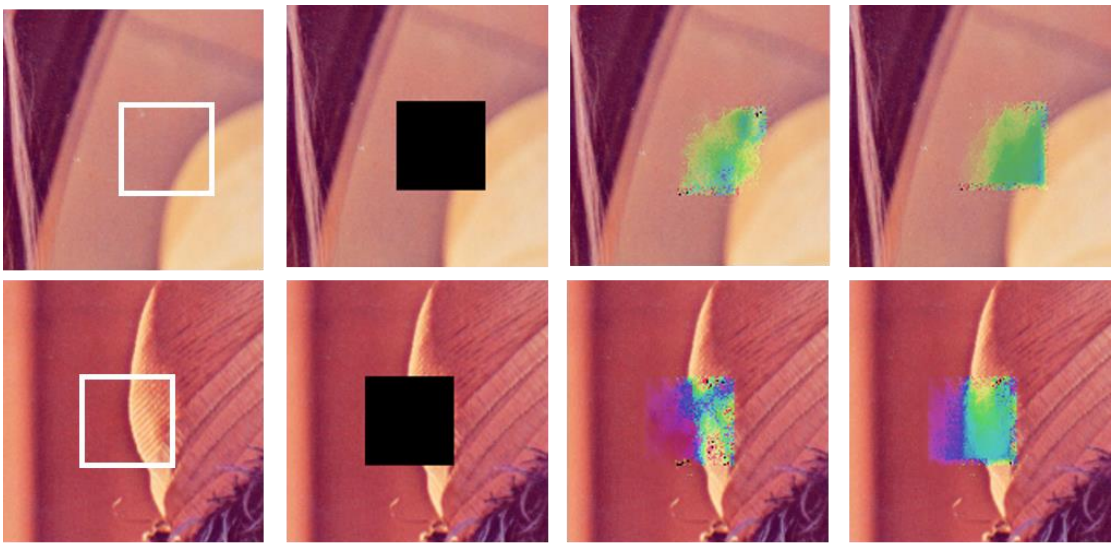
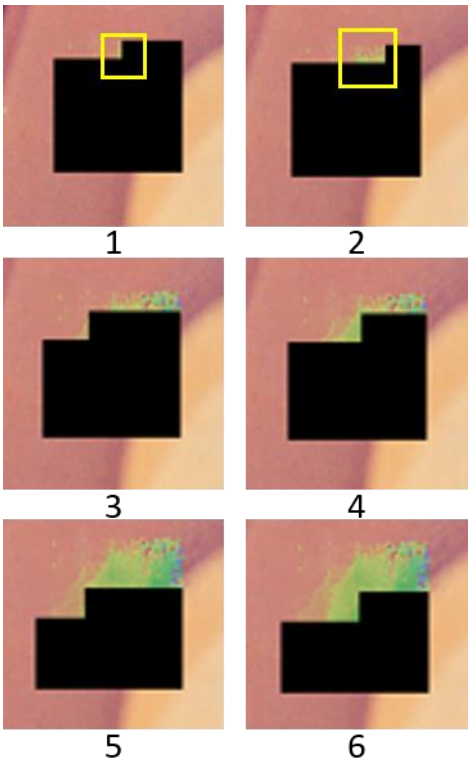


image originale | image bruité | ordre aléatoire | ordre simple

Nous observons que le résultat de façon aléatoire est meilleur que celui de l'ordre simple. Dans les deux tests, nous pouvons trouver les frontières des objets mais il y a certains pixels qui sont remplis d'une couleur bizarre, surtout à la fin de « simple_fill_img », c'est-à-dire en bas à droite.

Pour trouver la raison, nous avons listé les détails du processus « simple_fill_img » (voir l'annexe I) :



Dans cette figure, on peut trouver qu'au début, c'est juste quelques pixels bruits (voir l'image 1).

Dans la suite, il croit que les pixels bruits sont des pixels « règles », donc il continue à remplir les patchs avec les couleurs bizarres.

De plus, on peut aussi trouver que cette fonction marche bien pour le remplissage de couleur (en haut à gauche).

Après avoir trouvé la raison, nous avons décidé de créer une fonction plus « intelligente » -- « better_fill_img ». Dans cette fonction, on calcule une valeur de priorité pour chaque patch. Cette valeur est égale au nombre de pixels exprimés dans chaque patch. Voici le résultat :

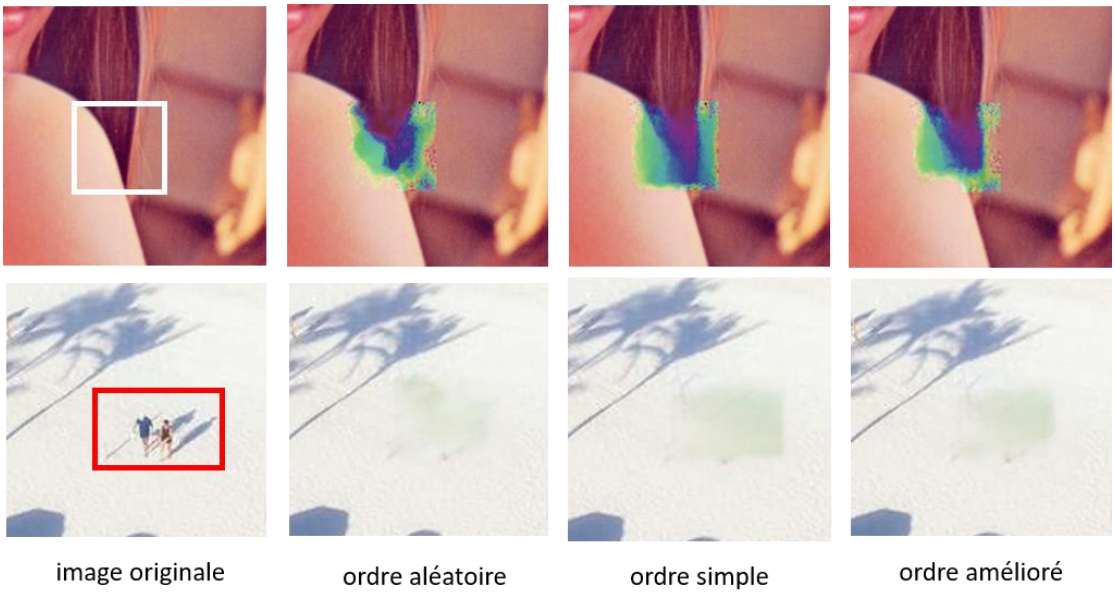


image originale | ordre aléatoire | ordre simple | ordre amélioré

Comme prévu, la fonction améliorée marche mieux aux quatre frontières du bruit mais il ne résout pas le problème de couleur bizarre. Dans cet exemple, nous pouvons bien voir que l'ordre de remplissage est important pour le inpainting.

Dans le futur, ce que nous pouvons faire est de résoudre le problème de couleur et encore améliorer l'ordre en ajoutant l'influence de l'isophote dans la formule de priorité, comme présenté dans l'article de A.Criminisi.[1]

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/criminisi_tip2004.pdf

III. Annexe

I. Evolution de la réparation « simple »