07/03/2018

Synthèse de littérature

Reconnaissance de chiffre manuscrit

SYS843 : réseaux de neuronaux et systèmes flous

Stephane Gu
GUXS01079507

Mise en situation

La reconnaissance de chiffres manuscrit est un problème pour les applications de reconnaissance optique des caractères. Cette reconnaissance a pour objectif de traduire des images de textes imprimés ou manuscrit en fichier texte. Les applications vont du triage de courrier au traitement de chèques bancaire. De manière générale, les applications envisagées permettraient d'automatiser des taches redondantes pour les humains. En effet, un exemple serait pour le transfert de documentation en entreprise du format papier au format numérique. Avec une application permettant de reconnaitre les textes manuscrits ou imprimés, le transfert des documents sur serveur serait beaucoup plus rapide.

Les applications dans ce domaine sont importantes car elles pourraient permettre de réduire l'écart qui existe actuellement entre les documents papiers et les documents numérisés. Numérisé des documents papiers revient pour l'instant a simplement les scanner et à les conserver sous format d'image.

Problématique abordée

Pour résoudre un problème de reconnaissance, l'extraction de caractéristique semble être aussi important que l'algorithme de classification. Ainsi la résolution du problème dépend d'une combinaison des algorithmes d'extraction et de classification. Cependant, avec un choix assez large entre les algorithmes de classification et d'extractions, les combinaisons possibles sont nombreuses. Cependant, comme les performances des algorithmes de classification semblent assez homogènes, nous n'étudierons qu'un seul de ces algorithmes.

Objectifs du projet

L'objectif de ce projet est donc de quantifier l'influence des algorithmes d'extractions de caractéristiques sur la précision de l'algorithme de classification. Nous souhaitons d'une certaine manière à vérifier que la combinaison des algorithmes les plus efficaces est bien la méthode la plus optimisée pour obtenir la meilleure solution au problème de reconnaissance. Pour simplifier le projet, nous n'étudierons que la reconnaissance de chiffres manuscrits.

Méthodologie

En ce qui concerne la méthodologie du projet, nous aborderons le problème de manière classique :

1. Preprocessing : Cette étape a pour but de traiter les images pour faciliter la phase d'extraction de caractéristiques. Ainsi, lors de cette étape, nous normaliserons la taille des images, et pour faciliter le projet, nous limiterons le nombre de classes à identifier. C'est aussi à cette étape que nous nous assurons de la bonne segmentation des images.
2. L'extraction de caractéristiques : Une étape très importante, nous utiliserons l'un des algorithmes présenté plus loin dans l'article pour extraire les caractéristiques nécessaires à la classification. Comme à l'étape précédente, nous avons assuré la bonne segmentation des images, nous extrayons des caractéristiques d'une image avec un seul caractère.
3. Classification : Nous utiliserons l'un des algorithmes pour classifier les différentes images dans leurs classes. Chaque classe représentera donc un chiffre.

Les critères de sélections des algorithmes seront détaillés en fonction des avantages et inconvénients.
En ce qui concerne le choix des bases de données, nous choisissons la base de données MNIST.

Structure du document

Dans cet article, nous aborderons l'ensemble des algorithmes les plus utilisés pour la reconnaissance de caractères manuscrits. Nous commencerons tout d'abord par étudier les algorithmes d'extraction de caractéristiques, pour enchainer sur les algorithmes de classifications. Enfin, nous étudierons les taux d'erreurs de ces algorithmes combinés deux à deux.

Synthèse des techniques

Extraction de caractéristiques

Pour l'extraction de caractéristiques commençons donc par étudier l'extraction de caractéristiques directionnelles (direction feature). Elle se traduit en 3 étapes :

- La normalisation de l'image : on standardise la taille de l'image tel que si on pose r le ratio entre la longueur et la largeur de l'image et r' le ratio de l'image normalisée, on a :

$$r' = \sqrt{\sin\left(\frac{\pi}{2}r\right)},$$

Figure 1: Formule de ratio de l'image normalisée

- Assigner les directions : On décompose l'image en sous-image directionnelle (directional sub-image). On utilise généralement des masques gaussiens comme celui-ci :

$$h(x,y) = \frac{1}{2\pi\sigma_x^2}\exp\left(-\frac{x^2+y^2}{2\sigma_x^2}\right) \quad \text{avec} \quad \sigma_x = \frac{\sqrt{2}t_x}{\pi},$$

Figure 2: Masque gaussien pour l'assignement des directions

- Mesurer les caractéristiques : On partitionne les sous images par zone uniforme et l'intensité de chaque zone est accumulé comme mesure tel que si on pose (xo,yo) le centre du masque gaussien, on ait la mesure suivante :

$$F(x_0,y_0) = \sum_x \sum_y f(x,y)h(x-x_0, y-y_0).$$

Figure 3: Formule de la mesure de la caractéristique

Les articles étudiés mentionnent aussi d'autres méthodes de mesurer les caractéristiques directionnelles notamment :

- Le chaincode : On assimile à chaque pixels de contour de l'image normalisée un code a 8 direction

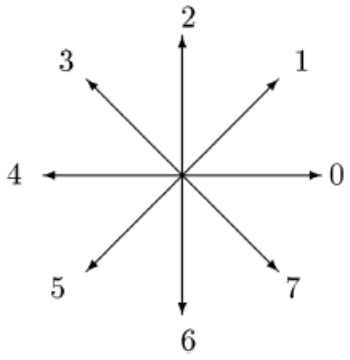

Figure 4 : 8 directions du chaincode

- Un gradient avec un opérateur de Sobel ou de Kirsh : On décompose l'image en 4 orientations plan (orientation planes) ou 8 directions plan (direction planes). Les masques sont les suivants :

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Figure 6 Masque de Sobel

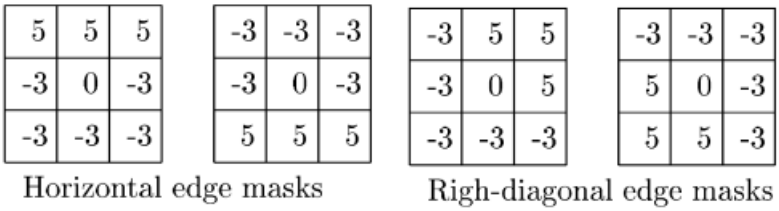

Figure 5 Masque de Kirsh pour les bords horizontaux et diagonaux

tel que, pour le masque de Sobel, si on pose
$g_x(x,y) = f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1) - f(x-1,y-1) - 2f(x-1,y) + f(x-1,y+1)$
et $g_y(x,y) = f(x-1,y+1)+2f(x,y+1) + f(x+1,y+1) – f(x-1,y-1) – 2f(x,y-1) + f(x+1,y-1)$;
l'intensité et la direction du gradient sont calculées par le vecteur $[g_x ; g_y]^T$. De plus, pour l'extraction de caractéristiques pour des caractères, la direction est partitionnée en nombre de régions avec chaque région correspondant à une sous image directionnelle. Ainsi, on calcule l'intensité du gradient avec l'intensité de la sous image correspondante.
Pour le masque de Kirsh on peut calculer les composantes de l'image selon 8 directions. Ainsi, en posant 4 orientations (Horizontal, vertical, diagonal gauche et droite), on a les intensités suivantes :
$G(i,j)_H = \max(|5S_0 – 3T_0|,[5S_4 – 3T_4|)$
$G(i,j)_V = \max(|5S_2 – 3T_2|,[5S_6 – 3T_6|)$
$G(i,j)_L = \max(|5S_3 – 3T_3|,[5S_3 – 3T_3|)$
$G(i,j)_R = \max(|5S_1 – 3T_1|,[5S_1 – 3T_1|)$
avec $S_k = A_k + A_{k+1} + A_{k+2}$; $T_k = A_{k+3} + A_{k+4} + A_{k+5} + A_{k+6} + A_{k+7}$ et A défini de la manière suivante :

A_0	A_1	A_2
A_7	(i,j)	A_3
A_6	A_5	A_4

Figure 7:Définition des pixels voisins

- Le PDC (peripheral direction contributivity) : On décompose les pixels au bord du caractère en profiles hiérarchiques et la direction de ces pixels sont déterminées par la longueur dans la zone de course (run-lenghts in stroke area). Cela revient à calculer la distance entre un pixel (x,y) et le bord le plus proche dans 8 directions et a les regrouper dans 4 composantes d_m, m allant de 0 à 4 :

$$d_m = \frac{l_m + l_{m+4}}{\sqrt{\sum_{i=0}^{3} (l_i + l_{i+4})^2}}.$$

Figure 9Composante de direction

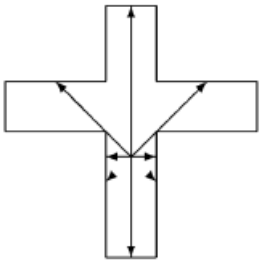

Figure 8Exemple de calcule de contributivité directionnelle

On extrait ensuite les mesure de caractéristiques des composantes de direction des pixels au bord des profils hiérarchiques. En ce qui concerne les profils, ils sont composés des pixels au bord traversé par des traits commençant des 4 bords de l'image (crossed by rays starting from four boundaries of image), et on compte 2 hiérarchies par bords. Les pixels les plus au bord constituent la 1ère hiérarchie, les pixels suivants étant la 2nd. Puis, on partitionne les 8 profils en intervalle uniforme et on compte chaque composante de direction.

Ceci conclut les méthodes d'extraction de caractéristiques principales. On note qu'il existe aussi des caractéristiques de structure de profil (profile structure features) qu'on utilise comme compléments aux caractéristiques de direction. Citons rapidement ces caractéristiques :

- Horizontal crossing counts et peripheral concavity measurements : réaliser un flou gaussien de 1D pour obtenir les mesures. La concavité représente la distance entre les contours et l'enveloppe convexe.

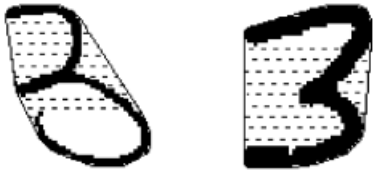

Figure 10 Exemple de caractéristiques concaves

Algorithmes de classification

Abordons maintenant les algorithmes de classification pour les caractères manuscrits.

- Classificateurs neuronaux : on compte 3 algorithmes envisageables pour un classificateur neuronal :
  - Un MLP avec une couche cachée : Pour une classification de classe M, le réseau de neurone aura M sorties. Ainsi, soit une entrée x = $(x_1, ..., x_d)^T$, la sortie de classe k (k allant de 1 a M) est :

$$y_k(\mathbf{x}) = s\left[\sum_{j=1}^{N_h} w_{kj} s(\mathbf{v}_j^{\mathrm{T}}\mathbf{x} + v_{j0}) + w_{k0}\right] = s\left[\sum_{j=1}^{N_h} w_{kj} h_j + w_{k0}\right]$$

Figure 11:Calcul de la sortie de classe k

avec N_h le nombre d'unités cachées, w_{kj} et v_{ji} les poids respectifs de la couche de sortie et de la couche cachée. On pose s(a) = $\frac{1}{1+e^{-a}}$ la fonction sigmoïde.
On calcule alors les poids des couches avec l'algorithme d'erreur de rétro propagation (error back-propagation algorithm)

$$E = \frac{1}{N_x}\left\{\sum_{n=1}^{N_x}\sum_{k=1}^{M}[y_k(\mathbf{x}^n, \mathbf{w}) - t_k^n]^2 + \lambda \sum_{w\in W} w^2\right\}$$

Figure 12: Formule de l'erreur de retro propagation

avec N_x le set d'exemples, λ le coefficient de contrôle de la dégradation des poids (decay of connecting weights), t_k^n la valeur voulue de la classe k (égale à 1 si dans la classe, 0 sinon). Les poids sont ainsi mis à jour avec une descente de gradient stochastique

- Un classificateur RBF : Ce classificateur a 1 couche cachée avec chaque unité cachée étant un noyau gaussien $h_j(x) = \exp(-\frac{\|x-\mu j\|^2}{2\sigma j^2})$ tel que chaque sortie est une combinaison linéaire d'un noyau gaussien avec une sigmoïde non linéaire

$$y_k(\mathbf{x}) = s\left[\sum_{j=1}^{N_h} w_{kj} h_j(\mathbf{x}) + w_{k0}\right]$$

Figure 13 : Calcul de la sortie du RBF

Pour l'apprentissage des paramètres, on initialise les paramètres du noyau à l'aide d'un apprentissage non supervisé et on met à jour les poids avec une descente de gradient pour minimiser l'erreur quadratique moyenne.

- Un classificateur polynomial quadratique sur sous-espace linéaire : Comme c'est un réseau de neurone a 1 couche, les entrées du réseau seront les polynômes issus des mesures d'entrée. Si on note z la composante principale sur le sous-espace de dimension m (m<d), la sortie de classe k est donc :

$$y_k(\mathbf{x}) = s\left[\sum_{i=1}^{m}\sum_{j=i}^{m} w_{kij}^{(2)} z_i(\mathbf{x}) z_j(\mathbf{x}) + \sum_{i=1}^{m} w_{ki}^{(1)} z_i(\mathbf{x}) + w_{k0}\right]$$

Figure 14: Formule de la sortie d'un PC quadratique

avec $z_j(x)$ la projection de x sur la jème axe principal du sous-espace.
Les poids sont aussi calculés par descente de gradient pour minimiser l'erreur quadratique moyenne. De plus, nous recalculons les projections de la manière suivante :

$$z_j(\mathbf{x}) = \frac{(\mathbf{x} - \mu_x)^{\mathrm{T}} \phi_j}{\sqrt{\lambda_1}}, \qquad j = 1, 2, \ldots, m,$$

avec μ_x le vecteur moyen, φ_j pour j allant de 1 a m le vecteur propre. Ici λ_1 est la valeur propre la plus grande correspondant au vecteur propre φ_1.

- Learning vector quatization (LVQ) classifiers : Un classificateur LVQ utilise le principe du plus proche voisin avec un apprentissage supervisé. On utilise comme algorithme d'apprentissage l'erreur minimum de classification (MCE) tel que pour M classes $\{C_k \mid k = 1, 2, \ldots, M\}$ il y a n_k

prototypes $\{m_{kj} \mid j = 1,\ldots,n_k\}$. Soit les données d'entrainement $\{(x^n,c^n) \mid n= 1,\ldots,N\}$ (c^n la classe associé au label x^n), on calcule les prototypes en minimisant la fonction suivante :

$$L_0 = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{M} l_k(\mathbf{x}^n) I(\mathbf{x}^n \in C_k)$$

avec I la fonction indicatrice (égale à 1 si x^n appartient à la classe C_k, 0 sinon)
On calcule ainsi $l_k = l_k(\mu_k) = \frac{1}{1+e^{-\xi\mu k}}$ avec $\mu_k = d(x,m_{ki}) - d(x,m_{rj})$ où $d(x,m_{ki})$ est la distance minimale aux prototypes des aux vraies classes du modèle d'entrainement et $d(x,m_{rj})$ la distance minimale a la classe rivale la plus proche tel que $d(x,m) = \|x - m\|^2$.
On minimise L_0 par approximation stochastique et on initialise les classes prototypes par regroupement des k moyennes (k-means clustering). Puis, on met à jour les prototypes de classes de la manière suivante :

$$\mathbf{m}_{ki} = \mathbf{m}_{ki} + 2\alpha(t)[\xi l_k(1-l_k)(\mathbf{x}-\mathbf{m}_{ki}) + \lambda(\mathbf{x}-\mathbf{m}_{ki})] \qquad \mathbf{m}_{rj} = \mathbf{m}_{rj} - 2\alpha(t)\xi l_k(1-l_k)(\mathbf{x}-\mathbf{m}_{rj})$$

Figure 15: Equations de mises à jours des distances

Avec α(t) la rythme d'apprentissage (on suppose suffisamment petit et décroissant avec le temps).

- Discriminative learning quadratic discriminant function (DLQDF) classifier : Algorithme basé sur le MQFD2 (modified quadratic function discriminant 2) qui est égale à
$g_2(x,w_i) = \sum_{j=1}^{k}\frac{1}{\lambda ij}[(x-\mu i)^T\phi ij]^2 + \frac{1}{\delta i}\{\|x-\mu i\|^2 - \sum_{j=1}^{k}[(x-\mu i)^T\phi ij]^2\} + \sum_{j=1}^{k}\log(\lambda ij) + (d-k)\log(\delta i)$
avec μ le vecteur moyen de la classe w_i ; λ_{ij} et ϕ_{ij} les valeurs propres et vecteur propres de la covariance de la classe w_i.
On note k le nombre de composantes principales tel qu'après avoir classé les valeurs propres dans l'ordre décroissant (les vecteurs correspondants aussi) pour tout i > k, $\lambda_{ij} = \delta_i$ une constante.
On initialise les paramètres du DLQDF à partir des paramètres de MQDF2 et on optimise ces paramètres avec le critère de MCE correspondant à la fonction suivante :

$$L_1 = \frac{1}{N}\sum_{n=1}^{N}[l_c(\mathbf{x}^n) + \beta g_2(\mathbf{x}^n, \omega_c)]$$

Figure 16 : Distance quadratique de la vrai classe pour le DLQDF

Avec w_c la vraie classe du modèle d'entrainement, β le coefficient de régularisation.
Enfin, pour évoluer les paramètres par descente de gradient stochastique, on implémente aussi le théorème de Gram-Schmidt pour maintenir l'orthogonalité des vecteurs propres et ainsi permettre à notre fonction de discrimination de vérifier l'hypothèse de densité Gaussienne (assumption of Gaussian density).

- Support vector classifiers (SVM) : Comme ce classificateur est binaire, pour réussir une classification multi-classes, il faut combiner plusieurs SVMs. Ainsi, un problème a M classes peut être décomposé en M problème binaire avec chaque classificateur séparant une classe d'une autre ou décomposé en $\binom{M}{2}$ problèmes binaires avec chaque classificateur séparant une paire de classe. On considère cette option pour le reste de cette revue.
Soit le modèle x, la fonction de discrimination est calculée par :

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} y_i\alpha_i k(\mathbf{x},\mathbf{x}_i) + b$$

Figure 17: Fonction de discrimination d'un SVM

avec l le nombre de modèle d'apprentissage, y_i la valeur voulue pour le modèle d'apprentissage x_i (+1 pour la première classe, -1 pour la seconde), b un biais et $k(x,x_i)$ une fonction noyau tel que $k(x,x_i) = \phi(x).\phi(x_i)$ où $\phi(x)$ est le vecteur de caractéristique dans l'espace objet étendu (expanded feature space) d'un dimensionnalité potentiellement infinie.
On utilise souvent deux types de noyaux, un noyau polynomial ou un noyau RBF. Ils sont calculés de la manière suivante : $k(x,x_i,p) = (1 + x.x_i)^p$ et $k(x,x_i,\sigma^2) = \exp(-\frac{\|x-xi\|^2}{2\sigma^2})$ respectivement.
Comme la fonction de discrimination est vue comme une fonction de discrimination linéaire généralisée, le vecteur poids est $w = \sum_{i=1}^{l} yi.\alpha i.\phi(xi)$ et on détermine les coefficients α_i avec les données d'apprentissages en résolvant le problème d'optimisation suivant :

- Minimiser $t(w) = \frac{1}{2}\|w\|^2$
- Appliquer a $y_i f(x_i) \gg 1 - \xi_i$ avec $\xi_i \gg 0$ et i allant de 1 a l.

Ce problème revient à résoudre le problème suivant :

- Maximiser $W(\alpha) = \sum_{i=1}^{l} \alpha i - \sum_{i,j=1}^{l} \alpha i.\alpha j.yi.yj.k(xi,xj)$
- Appliquer a $0 \ll\propto i \ll C$, pour i allant de 1 à l et $\sum_{i=1}^{l} \alpha i.yi = 0$, C le paramètre de contrôle de la tolérance au erreur de classification pendant l'apprentissage.

Pour résoudre ce problème d'optimisation, on propose 2 algorithmes :

- Le SMO de Platt (sequential minimal optimization) : Cet algorithme a comme principe de résoudre le problème d'optimisation le plus petit possible à chaque itération. Comme la résolution inclue 2 multiplicateurs de Lagrange pour le problème le plus petit, l'algorithme choisit 2 multiplicateurs de Lagrange à chaque itération, trouve la valeur optimale pour ces multiplicateurs puis met à jour le SVM pour refléter ces valeurs optimales.
- Le SOR de Mangasarian (successive overrelaxation) : De la même manière, le SOR permet de résoudre le problème d'optimisation en calculant un multiplicateur à la fois.

Analyse critique

Dans cette partie nous traiterons des différentes performances des algorithmes présentés jusqu'à présent. Commençons par établir les vecteurs de caractéristiques qui seront utilisés pour évaluer les performances :

- Le vecteur blr : Obtenu à l'aide d'un chaincode a 4 orientation. On calcule 25 mesures à l'aide de masque gaussien de 5x5 sur chaque orientation.
- Le vecteur mul : Obtenu aussi avec un chaincode a 4 orientation mais avec 11 « crossing counts » et 22 mesures de concavité. On obtient ces mesures à l'aide d'un floutage gaussien a 1 dimension (1D Gaussian blurring).
- Le vecteur e-mul : Chaincode a 8 directions. De la même manière qu'avec le vecteur blr, on utilise des masques gaussiens de 5x5 sur les 8 directions.
- Le vecteur grd : Gradient a 4 orientation. On utilise aussi des masques gaussiens de 5x5 pour calculer 25 mesures.
- Le vecteur e-grd : Gradient a 8 directions et calcule de 25 mesures avec un masque gaussien 5x5.
- Le vecteur grg : Gradient a 4 orientation tiré d'une image en pseudo échelle de gris. Une image en pseudo échelle de gris est une image binaire normalisée en échelle de gris.
- Le vecteur e-grg : Gradient a 8 directions tiré d'une image en pseudo échelle de gris
- Le vecteur Kir : Gradient à 1 orientation par opérateur de Kirsh. On calcule 25 mesures en par convolution d'un masque gaussien de 5x5

- Le vecteur pdc : Obtenu à l'aide d'un PDC à 4 orientation. Chacun des 8 profils (4 en bordures et 2 hierarchiques) sont partitionnés en 7 intervalles et la direction des composantes de chaque intervalle sont comptés

Chacun de ces vecteurs sont transformés avant la classification tel que $y = x^{0.5}$.
Pour permettre un étude complète, l'étude réalise des tests avec 3 bases de données différentes :

- La base CENPARMI : avec 6000 images, 4000 images (400 par classes) sont utilisées pour l'entrainement et 2000 pour les tests
- La base CEDAR : la base d'entrainement (nommé br) contient 18468 images (le nombre d'image par classe varie). La base de test contient 2711 images pauvrement segmentées (bs) et 2213 images bien segmentées (goodbs).
- La base MNIST : Pour éviter des problèmes d'homogénéité de style d'écriture, cette base a été composé des bases de données spéciales SD3 et SD7. Elle contient 60 000 images pour l'entrainement et 10 000 images pour la base de test.

Pour rappel, toutes les images ont été normalisée a une taille de 35x35 pour la classification.

Abordons maintenant les paramètres à définir pour chaque algorithme :

- MLP : Weight decay initialisé a 0.05
- RBF : Weight decay initialisé a 0.02
- PC : Weight decay initialisé a 0.1
- LVQ : coefficient de régularisation initialisé a 0.05/var si on pose var comme la variance estimée de la base d'entrainement après regroupement
- DLVQ : coefficient de régularisation initialisé a 0.1/var
- SVC Polynomial kernel : on initialise p = 5 et C = 1
- SVC RBF kernel : on initialise $\sigma^2 = 0{,}3.var$ et C = 10

De plus, pour permettre une bonne généralisation des performances des algorithmes, la taille des classificateurs a été déterminé empiriquement pour obtenir une performance optimale. Comme la taille des classificateurs dépends de la taille de l'échantillon d'entrainement, nous avons les valeurs suivantes :

Database	MLP (N_h)	RBF (N_h)	PC (m)	LVQ (n_p)	LQDF (k)
CENPARMI	40	50	60	4	20
CEDAR	80	80	60	8	30
MNIST	300	300	70	30	40

Figure 18 : Taille des classificateurs en fonction de la base d'entrainement et du paramètre spécifique a chaque algorithme

On obtient ainsi les résultats suivant :

	blr	mul	e-blr	e-mul	grd	e-grd	grg	e-grg	Kir	pdc	Aver	Rank
k-NN	2.35	2.20	2.10	1.85	2.35	1.80	2.65	2.35	5.65	3.00	2.630	8
MLP	2.15	2.05	2.05	1.75	2.10	1.40	2.05	1.70	2.90	2.00	2.015	7
RBF	1.75	1.65	1.65	1.40	1.80	1.25	1.55	1.55	3.25	1.95	1.780	5
PC	1.55	1.40	1.25	1.20	1.65	1.20	1.25	1.20	2.30	2.05	1.505	3
LVQ	2.15	1.50	1.80	1.50	2.60	1.60	1.85	1.40	3.50	2.10	2.00	6
LQDF	1.50	1.50	1.20	1.10	1.65	1.05	1.45	**0.95**	2.35	1.95	1.470	2
SVC-poly	1.55	1.40	1.45	1.45	1.70	1.45	1.35	1.20	1.85	1.70	1.510	4
SVC-rbf	1.10	1.00	1.20	**0.95**	1.30	1.15	1.10	**0.95**	2.05	1.55	1.235	1
Average	1.763	1.587	1.587	1.40	1.894	1.363	1.656	1.412	2.981	2.037		
Rank	7	4	4	2	8	1	6	3	10	9		

Figure 19 : Taux d'erreur sur la base d'entrainement de CENPARMI

	blr	mul	e-blr	e-mul	grd	e-grd	grg	e-grg	Kir	pdc	Aver	Rank
k-NN	1.17	1.08	0.90	0.95	1.27	1.13	1.22	0.99	2.85	1.54	1.310	8
MLP	1.08	0.95	0.95	0.81	1.13	0.99	1.17	1.04	1.36	1.04	1.052	5
RBF	1.04	0.90	0.86	0.77	1.22	0.95	1.13	0.99	1.76	1.17	1.079	6
PC	0.81	**0.68**	0.81	0.77	0.99	0.86	1.08	0.99	1.68	1.08	0.975	4
LVQ	1.13	**0.68**	0.86	0.77	1.17	1.08	1.27	0.90	2.12	1.08	1.106	7
LQDF	0.81	0.86	0.90	0.72	0.95	0.72	0.86	0.86	1.13	1.13	0.894	3
SVC-poly	0.63	0.77	0.81	0.81	0.95	0.68	0.99	0.81	1.27	0.99	0.871	2
SVC-rbf	0.68	**0.54**	0.68	**0.54**	0.72	0.72	0.86	0.68	1.40	0.99	0.781	1
Average	0.919	0.808	0.846	0.768	1.050	0.891	1.073	0.907	1.696	1.127		
Rank	6	2	3	1	7	4	8	5	10	9		

Figure 21: Taux d'erreur sur la base d'entrainement de CEDAR

	blr	mul	e-blr	e-mul	grd	e-grd	grg	e-grg	Kir	pdc	Aver	Rank
k-NN	1.58	1.37	1.35	1.26	1.45	1.26	1.47	1.35	3.12	1.76	1.597	8
MLP	1.14	1.08	0.98	0.88	1.03	0.89	1.03	0.92	1.46	1.24	1.065	6
RBF	0.97	0.89	0.87	0.81	1.00	0.90	0.95	0.85	1.38	1.06	0.968	4
PC	0.89	0.90	0.72	0.75	0.97	0.85	0.90	**0.69**	1.29	1.23	0.919	3
LVQ	1.34	1.22	0.94	1.02	1.27	1.05	1.31	1.09	2.12	1.49	1.285	7
LQDF	1.07	0.93	0.86	0.90	1.14	0.87	0.87	0.85	1.22	1.16	0.987	5
SVC-poly	0.91	0.86	0.83	0.76	1.05	0.79	0.90	0.70	1.23	0.97	0.90	2
SVC-rbf	0.84	0.87	0.72	0.74	0.90	0.75	0.76	**0.61**	1.16	0.88	0.823	1
Average	1.093	1.015	0.909	0.890	1.101	0.920	1.024	0.883	1.623	1.224		
Rank	7	5	3	2	8	4	6	1	10	9		

Figure 20: Taux d'erreur sur la base d'entrainement de MNIST

Extraction de caractéristiques

Selon ces trois tableaux, on arrive aux conclusions suivantes :

- On constate qu'on a de meilleurs résultats quand on utilise 8 directions par rapport à 4 orientations
- De manière générale, les méthodes chaincode et gradient par opérateur de Sobel apportent de meilleurs résultats mais ces derniers dépendent de la base de données, surtout pour l'approche gradient tel qu'on définit l'approche gradient binaire comme les caractéristiques tirées d'une image binaire par l'approche gradient et l'approche gradient pseudo-gris comme les caractéristiques tirées d'une image en échelle pseudo-gris.
    - Si on utilise des images binaires, l'approche gradient n'a pas de réel avantage par rapport à chaincode car elles représentent les mêmes caractéristiques d'un même signal
    - L'approche gradient pseudo-gris présente un avantage pour les bases de données CENPARMI et MNIST. On peut expliquer cette différence de performances par la résolution de l'image. En effet pour les bases CENPARMI et MNIST, l'image de base a une taille inférieure à l'image normalisée. Cette augmentation de résolution rend la classification plus précise.

Algorithmes de classification

De la même manière, à partir des données des tableaux nous tirons les conséquences suivantes :

- L'algorithme avec le taux d'erreur le plus bas est l'algorithme SVC avec un noyau RBF mais c'est aussi l'algorithme le plus couteux en calcul et espace mémoire.
- Les 3 autres meilleurs algorithmes sont SVM avec noyau polynomial, PC et DLQDF

    - On note que les méthodes PC et DLQDF sont aussi performante car elles sont adaptées à la reconnaissance de caractère
    - Les méthodes SVM semblent adaptées à différents types de problèmes de reconnaissance.
- L'étude ne prend en compte que le taux d'erreur comme critère. Il ne prend pas en compte le temps de calcul ou la complexité
    - Si on souhaite prioriser la complexité, les approches PC et DLQDF sont à privilégier car ce sont des approches à faible complexité

Conclusion

Suite à cette revue, nous pouvons donc réaliser les choix des différents algorithmes. Commençons par l'algorithme de classification, puisque nous comptons étudier l'impact des algorithmes d'extraction en priorité. Dans l'article étudié, les chercheurs ont priorisé l'étude de la précision et n'a donc pas mentionner les coûts en calcul et en espace mémoire. N'ayant pas les mêmes moyens que ces chercheurs, pour cette étude nous choisissons un algorithme avec une faible complexité. Ainsi, nous choisissons l'algorithme DLQDF qui combine un taux d'erreur compétitif et une complexité faible. En ce qui concerne le choix des algorithmes d'extractions, nous souhaitons comparer l'impact des deux meilleurs algorithmes d'extractions donc nous choisissons les algorithmes chaincode et gradient par opérateur de Sobel.

Bibliographie

Cheng-Lin Liu et al., 2002. Handwritten digit recognition using state-of-the-art techniques, Frontiers in Handwriting Recognition

C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Knowledge Discovery Data Mining 2 (2) (1998) 1–43.

C.-L. Liu, Y.-J. Liu, R.-W. Dai, Preprocessing and statistical/structural feature extraction for handwritten numeral recognition, in: A.C. Downton, S. Impedovo (Eds.), Progress of Handwriting Recognition, World Scientific, Singapore, 1997, pp. 161–168

S.-W. Lee, Multilayer cluster neural network for totally unconstrained handwritten numeral recognition, Neural Networks 8 (5) (1995) 783–792.

G. Srikantan, S.W. Lam, S.N. Srihari, Gradient-based contour encoder for character recognition, Pattern Recognition 29 (7) (1996) 1147–1160.

Concordia University, Center for Pattern Recognition and Machine intelligence. http://www.concordia.ca/research/cenparmi/resources/herosvm.html

Olvi L. Mangasarian, David R. Musicant, Successive Overrelaxation for Support Vector Machines, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 10, NO. 5, SEPTEMBER 19997

John C. Platt, Fast Training of Support Vector Machines using Sequential Minimal Optimization, Advances in Kernel Methods - Support Vector Learning, MIT Press 2007