



# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand &amp; Marzook

## Table des matières

<b>I.</b>	<b>Introduction</b>	<b>2</b>
<b>II.</b>	<b>Optical Character Recognition</b>	<b>3</b>
II.1.	Qu'est-ce que l'OCR et comment c'est né ? . . . . .	3
II.2.	Comment ça marche ? . . . . .	5
II.2.a	La préanalyse - Reduction du bruit . . . . .	5
II.2.b	La préanalyse - La segmentation . . . . .	8
II.2.c	L'analyse - Approches de la reconnaissance de caracteres isolé .	10
II.2.d	L'analyse - Approche structurale . . . . .	10
II.2.e	Classieur bayésien . . . . .	11
II.2.f	Méthodes des $k$ plus proches voisins (KPPV) . . . . .	11
II.2.g	L'analyse - Méthodes de séparation linéaire et non linéaire . . .	12
II.2.h	L'analyse - Modèles de Markov cachés . . . . .	12
II.2.i	Programmation dynamique 2D . . . . .	13
II.2.j	L'analyse - Le post traitement . . . . .	13
II.2.k	L'analyse - Generation . . . . .	13
<b>III.</b>	<b>La reconnaissance par Intelligence artificielle</b>	<b>14</b>
III.1.	Présentation des réseaux de neurones . . . . .	14
III.1.a	Les notations . . . . .	14
III.1.b	Fonctionnement d'un neurone . . . . .	15
III.1.c	Idée de réseau de neurones . . . . .	15
III.2.	La méthode de rétropropagation par calcul du gradient . . . . .	16
III.2.a	Qu'est-ce que la rétropropagation . . . . .	16
III.2.b	Les mathématiques de la rétropropagation . . . . .	18
III.3.	Application de la technique à la reconnaissance de caractère . . . . .	19
III.3.a	Utilisation de la BDD MNIST sur les algorithmes . . . . .	19
III.3.b	Créations de nos propres bases . . . . .	19

---

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

---

## I. Introduction

Avant même de rentrer à l'EISTI le deep learning était un sujet qui nous passionnait tous. En effet aujourd'hui le deep learning est un sujet très en vogue dans le monde de l'informatique. C'est pour cela que le jour où nous avons entendu parler de TIPE nous avons tout de suite sauté sur l'occasion pour en apprendre plus sur ce sujet.

Mais le deep learning étant un sujet extrêmement vaste, compliqué et qui évolue constamment il nous fallait trouver un angle d'approche pertinent. Mais quoi de mieux que de commencer par la base. L'OCR étant un peu le "HELLO WORLD" de ce sujet il nous a semblé bon de combiner les deux pour en faire notre sujet d'étude.

Mais alors par quoi commencer ? Nous avons alors divisé le sujet en deux grandes parties distinctes : les différentes approches de l'OCR (optical character recognition ou reconnaissance optique de caractères en français) et le deep learning et ces applications dans la reconnaissance de caractères.

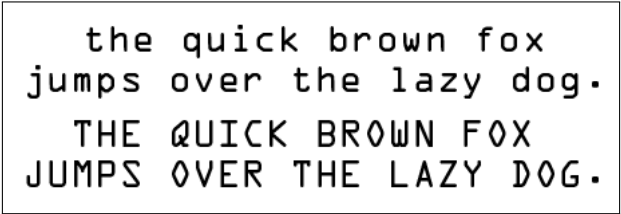
Nous espérons sincèrement que vous prendrez autant de plaisir à lire notre rapport que nous à l'écrire.

## II. Optical Character Recognition

### II.1. Qu'est-ce que l'OCR et comment c'est né ?

L'OCR ou optical character recognition est un procédé qui a pour principe de permettre à une machine de reconnaître des caractères. En effet cela permet, lors d'un scan ou même de la lecture d'un PDF de récupérer simplement le texte, au lieu de devoir le recopier. L'idée de faire reconnaître des caractères à une machine peut paraître simple, mais la mise en place ne l'est pas. En effet la première machine à utiliser ce procédé date de 1929 et est due à un ingénieur Allemand, nommé Gustav Tauschek. Le principe de cette machine était assez simple, elle était formée d'une série de gabarit (ou filtre) représentant des mots, et la machine réagissait lorsque le mot correspondait à un des gabarits.

Une trentaine d'années plus tard, certaines entreprises se mettent à utiliser le principe d'OCR pour automatiser certaines tâches. Malgré cela le principe utilisé n'est pas au point. C'est pour cela qu'en 1966 l'ATF (American Type Founders) décide de créer une police spéciale, reconnaissable autant par les machines que par les Hommes. L'OCR-A est née. C'est la première police déchiffirable par l'humain mais également par la machine. Malheureusement les lettres, extrêmement stylisées pour ne pas se ressembler entre elles sont assez compliquées à comprendre pour l'Homme, et sont donc absolument pas agréables à la lecture. C'est pour cela que seulement 2 années après sa création, l'OCR-A est remplacé par l'OCR-B. Cette dernière est devenue une norme mondiale par l'ISO (Organisation Internationale de normalisation).



the quick brown fox  
jumps over the lazy dog.  
THE QUICK BROWN FOX  
JUMPS OVER THE LAZY DOG.

FIGURE 1 – Police OCR-A

La pratique de l'OCR est certes intéressante, mais devoir utiliser une police précise est un handicap assez embêtant. C'est exactement pour cette raison que Jacob Rabinow décide, pour palier à ce défaut, de comparer chaque caractères du texte à analyser avec la totalité des lettres d'un alphabet donné. La lettre qui en ressort est donc la lettre qui ressemble le plus à celle analysé. Cette idée date de 1950, avant même la création des polices exprès, mais elle n'est utilisé que plus tard en raison de quelques défauts dont un majeur, le temps qu'elle mettait pour comprendre un mot (au départ la machine était capable d'analyser environ une lettre par minute, de plus une écriture en italique ou quelques ratures et la machine était incapable de reconnaître le caractère).

Aussi étonnant que cela puisse paraître, une des première «entreprise» à ce servir de

---

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

---

cette technologie est ni plus ni moins que la poste dans les années 65. En effet à cette époque, la poste regroupait les lettres/colis pour la même destination ensemble. Ainsi ils ont décidé d'utiliser un OCR afin de trouver où le colis devait être envoyé et en fonction de cela, une étiquette était appliquée, facilitant le travail des postiers.

Dans les années 90 la vente de scanner portable augmente fortement afin d'aider la PAO (publication assistée par ordinateur). Pour la PAO, le scanner et l'OCR sont des choses essentielles. En effet la PAO consiste à rédiger des livres/journaux, en respectant des règles strictes de publications, tout en étant le plus agréable à la lecture possible. C'est pour cela que l'OCR est si important, le plus souvent les premières idées se font à la main, puis grâce au scanner et à l'OCR, on reconstitue cela sur un ordinateur sans avoir à tout refaire, puis on ajuste les textes ou la police d'écriture, avant d'imprimer à grande échelle.

Même de nos jours, avec toutes les avancées technologiques réalisées, l'OCR n'est toujours pas parfaite. En effet en raison de l'amélioration des ordinateurs, un des problèmes sur internet est désormais de reconnaître un humain d'une machine. Pour savoir qu'il s'agit bien d'un humain, des tests ont été développés. C'est ce que l'on nomme les tests de Turing. Un des tests les plus connus aujourd'hui s'appelle le captcha. En effet le principe est simple. Un mot est écrit à l'écran et l'utilisateur doit le réécrire, principale difficulté, les lettres sont «mal» formées ou liées avec celles d'à côtés, ou alors un trait est sur le devant des lettres. Ainsi, un humain est quand même capable de déchiffrer le mot, mais un ordinateur utilisant l'OCR ne peut pas.

---

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

---

## II.2. Comment ça marche ?

### II.2.a La préanalyse - Reduction du bruit

Un des principal soucis avec l'utilisation de l'OCR est la qualité de la «photo» nécessaire pour l'analyse de celle-ci. En effet la reconnaissance marche sur les photos/scans, mais pour cela ce dernier ce doit d'être clair. C'est pour cela qu'une préanalyse est toujours nécessaire. Cette préanalyse passe par deux procédés principaux, la réduction du bruit, ainsi que la segmentation. C'est donc de ces deux principes que nous allons commencer par parler.

Le bruit, qu'est ce que c'est ? Le bruit dont on parle la n'a rien à voir avec une nuisance sonore, en effet il s'agit d'un problème numérique sur une photo. Ce problème mène a un ajout de pixels parasites sur l'image, et conduit ainsi à une perte importante de détails. Or pour une reconnaissance de caractère, plus le bruit est élevé plus la chance d'échouer la reconnaissance est élevée. Ainsi nous avons besoin de techniques servant à amoindrir ces effets. Parmi tous les bruits qui peuvent influencer une image, nous parlerons des trois principaux concernant l'OCR, le bruit de grenaille, de lecture, et de quantification.

Le bruit de grenaille, est un soucis du nombre de protons (ou électron selon le dispositif) transportant l'information par pixel. En effet si ce nombre est trop faible, les informations arrivant concernant la couleur d'un pixel sont trop peu précises, ainsi de nombreux pixels ne sont pas de la onne couleur ou de la bonne teinte, ce qui dans notre cas, peut causer des déformations de lettres. A partir du millier de photons par pixel, le bruit est quasiment invisible a l'œil nu. Le bruit de lecture quant à lui est dû à un manque de luminosité qui force l'appareil ou le scanner, à augmenter la sensibilité ISO (International Organization for Standardization). Plus cette sensibilité augmente, moins l'appareil aura besoin de lumière, mais plus le risque de bruit est élevé. Le dernier dont on parlera, est le bruit de quantification. Lors d'une photo c'est un signal analogique qui est récupéré, or on a besoin d'un signal numérique. Il passe donc par un convertisseur, dans lequel les valeurs sont discrètes, ainsi le signal se doit d'être une approximation de la valeur de départ. Ainsi il y a un risque de bruit.

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

Les principaux moyens de gérer le bruit sont les filtres. Il en existe de nombreux, mais commençons par le filtrage linéaire. Si filtre est dit linéaire continue, cela signifie que c'est une filtre de convolution (discret ou continu). Je vais donc présenter de façon simplifier et plus imagé le principe. L'image qu'on récupère est considéré comme une matrice ou chaque pixel est une case, contenant le nombre correspondant à un intensité de couleurs. On utilise à côté, une seconde matrice, nommée noyau de taille 3x3 ou 5x5. Ici on prendra une 3x3 pour rendre la chose plus simple. Le filtre, s'effectue pixel par pixel, et consiste a choisir un pixel (qu'on appelle pixel principal), et on défini la zone d'action du filtre. Ici 3x3 car c'est la taille du noyau. On prends donc une matrice 3x3

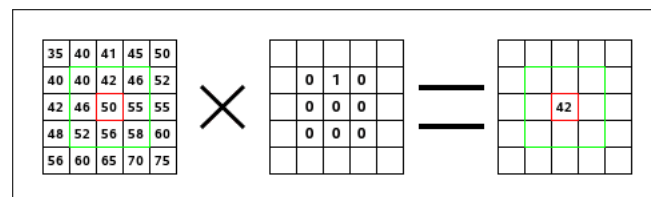


FIGURE 2 – Fonctionnement du filtre

dans l'image principale dont le centre est le pixel principal. On multiplie ensuite chaque pixel de la zone d'action avec le pixel du noyau correspondant. Et on additionne le tout. On obtient ainsi la nouvelle valeur pour le pixel principal. Il suffit de recommencer avec chaque pixel pour avoir toute l'image filtrée.

Avec en vert le champ d'action, en rouge le pixel principal. Au centre, on retrouve le noyau.

Dans cet exemple, le calcul a été effectué comme suit :

$$(40 * 0) + (42 * 1) + (46 * 0) + (46 * 0) + (50 * 0) + (55 * 0) + (52 * 0) + (56 * 0) + (58 * 0) = 42$$

ce filtre représentant ce noyau permet de décaler tous les pixels vers le bas. Chaque filtre, possède donc son noyau représentatif.

Pour les filtres suivants, une seule étape supplémentaire est nécessaire, celle de diviser le résultat par le nombre de cellules du noyau (donc 9), cela permet de conserver la linéarité. Nous allons donc maintenant parler des différents filtres existants.

Le filtre Passe haut, son noyau est représenté juste en dessous

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

FIGURE 3 – Matrice passe-haut

Ce filtre permet de favoriser les pixels à haute intensité, et donc améliore les détails et les contrastes, mais il peut augmenter le bruit, et est peu utilisé dans l'OCR. Le filtre Passe bas, représenté ici,

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand &amp; Marzook

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

FIGURE 4 – Matrice passe-bas

agis à l'opposé du filtre précédent, en effet il permet d'adoucir légèrement les détails pour réduire le bruit. C'est l'un des filtres utiles à l'OCR, souvent utilisé associé au filtre passe haut, pour gérer le mieux la réduction du bruit sans la perte trop importante des détails. Le filtre moyenne existe également et est un type de filtre passe bas son noyau est exclusivement constitué de '1/9' pour que le pixel principal prenne la moyenne des pixels aux alentours. Il existe également un filtre assez important, nommé le filtre médian. Comme son nom l'indique, le principe est de prendre les 9 valeurs présentes dans le champ d'action, les classer dans l'ordre croissant, puis prendre la 5ème valeur. Le pixel principal devient alors cette valeur là. Ce filtre permet principalement de retirer les pixels avec une couleur beaucoup trop différente de ceux aux alentours. Il existe ensuite de nombreux autres filtres plus complexes que ceux cités précédemment, chacun ayant une utilité précise, comme les filtres : Laplacien, Gradient ou Gaussien.

Ainsi des filtres tels que ceux expliqués précédemment comme le passe-bas par exemple sont absolument essentiels dans la mise en place d'un OCR puisque la présence de bruit est absolument à éviter. Malgré tout cela, la préanalyse n'est pas finie pour autant, en effet il reste une étape extrêmement importante, la segmentation

## II.2.b La préanalyse - La segmentation

Avant de commencer toute segmentation de l'image, il y a un point important à gérer. En effet il faut permettre à l'ordinateur de reconnaître le texte du fond. Pour cela, une des méthodes les plus simple se nomme, binarisation. Cela consiste à créer deux classes, et à séparer tous les pixels de l'images dans ces deux classes. La limite entre ces deux classe est nommé seuil, dépend du cas, c'est la partie la plus compliqué de l'algorithme à mettre en place. Ce seuil se base le plus souvent sur un histogramme des nuances de gris, et nécessite dans la plupart des cas, le passage d'un filtre passe-bas au préalable. En effet le bruit affecte fortement l'efficacité de cette segmentation. Pour en revenir, au seuil, il est difficile à choisir puisqu'il va séparer de manière totale les différents pixels, et la version la plus simple est nommé seuillage global, l'équation ramenant à cela peut se résumé à :

$$\forall (i, j) \in N * M, \begin{cases} 1 & \text{si } f(i, j) > S \\ 0 & \text{sinon} \end{cases}$$

où

- $(i, j)$  est un pixel
- $I$  est l'image binarisée
- $f(i, j)$  est la valeur du pixel
- $S$  Le seuil choisi

Grâce à cette méthode on peut donc séparé le fond du texte, en obtenant à la fin des pixels blanc, le fond, et des noirs, le texte. Les problèmes du seuillage globale (outre la difficulté à trouver le seuil), sont dû à la qualité de l'image, un document mal éclairé ou un changement de couleur de police claire risque de faussé sérieusement le résultat. Il existe une méthode plus efficace, nommé le seuillage local. Le principe est le même que précédemment, à un détail près, très important, le seuil dépend de la zone à analyser. Il existe deux méthode principales pour cela : La méthode de Bernsen, pour l'appliquer, on doit ce focalisé sur un petit espace autour du pixel à analyser et le filtre ressemble donc a :

$$S(i, j) = \frac{(\max(i, j) + \min(i, j))}{2}$$

où

- $S(i, j)$  est le seuil au point  $(i, j)$
- $\max(i, j)$  valeur max niveau gris dans l'espace autour de  $(i, j)$
- $\min(i, j)$  valeur min niveau gris dans l'espace autour de  $(i, j)$



# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

La seconde méthode, celle de Niblack, en utilisant les moyennes et écart type à la place des minimums et maximum pour être moins sensible au bruit

$$S(i, j) = x(i, j) + 0,2 * y(i, j);$$

où

- $S(i, j)$  est le seuil au point  $(i, j)$
- $x(i, j)$  est l'écart type dans l'espace autour de  $(i, j)$
- $y(i, j)$  est la moyenne des niveaux de gris dans cet espace

Ces deux méthodes sont les bases de la binéarisation mais connaissent de gros défaut à cause des erreurs présentes sur le document, mais sont extrêmement efficace si le document est quasi parfait. Il existe dans ce domaine, encore de nombreuses autres méthodes pour effectuer les mêmes choses, avec des taux de réussite plus ou moins élevé. Après avoir distingué le fond des textes, faut-il encore séparer chaque phrase des autres. Et encore une fois le nombre de méthode pour y parvenir est extrêmement élevé. Mais il existe deux types d'algorithmes principaux, la segmentation ascendante, puis celle descendante. Pour la première des deux, le principe consiste à partir des pixels, et à utiliser des approximations, en les liant à leurs voisins si ceux-ci possèdent des propriétés en commun (la couleur, ou la nuance de gris par exemple). Une des premières méthodes à être utilisées pour cela et à l'être toujours est la RLSA (run length smoothing algorithm).

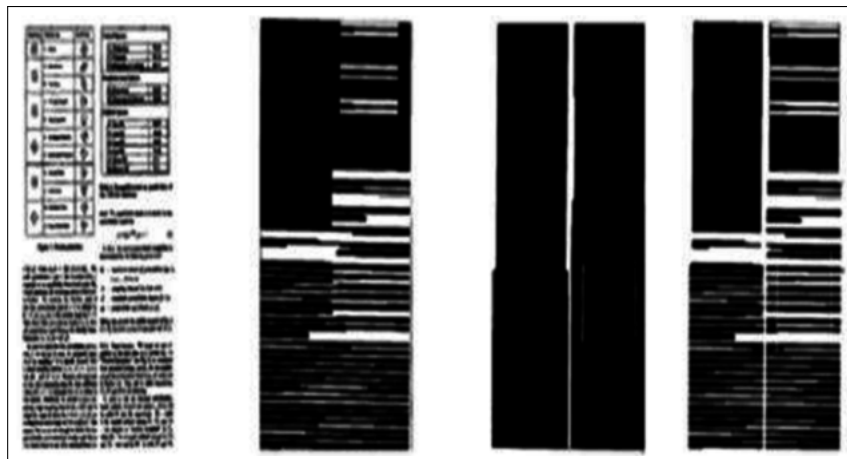


FIGURE 5 – Dans l'ordre, image originale, lissage horizontal, lissage vertical et enfin RLSA

Cet algorithme représenté ci-dessus, possède de grands points communs avec le seuillage global utilisé précédemment. En effet, il permet d'effectuer dans le cas global une séparation de phrases avec le reste, et nécessite de définir un seuil, qui dépend du document. Le principe n'est pas extrêmement compliqué. Si l'algorithme repère une

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

série de pixels blanc entre deux pixels noirs, et que cette distance est inférieure au seuil donné, alors ces pixels blanc deviennent noirs. Il suffit ainsi d'effectuer tout l'algorithme horizontalement puis verticalement (avec deux seuils pas forcément égaux) pour obtenir le RLSA. Pour parler de la segmentation descendante, nous allons expliquer la méthode dites «XY-cut». Cette méthode, prends le document au complet et pratique une projection horizontale (ou verticale). Une projection horizontale (ou verticale) est donc la somme des niveaux de gris sur la ligne (ou colonne). l'image est donc projeté horizontalement, et, aux endroits où les valeurs sont les plus faibles, découpés pour former des bandes. Puis on effectue une projection verticale sur les bandes pour découper en colonne etc. jusqu'à un certain seuil. Malgré que cette méthode soit efficace, si le texte est un incliné l'algorithme ne pourra jamais le retranscrire correctement. Ces deux méthodes, si poussé au maximum avec des seuils extrêmement faible, peuvent permettre de segmenter les phrases en mots, et même en lettre malgré une précision loin d'être parfaite

## II.2.c L'analyse - Approches de la reconnaissance de caracteres isolé

La reconnaissance est une tâche diicile à réaliser selon l'application demandée. C'est pour cela que les chercheurs ont développé plusieurs approches. On distingue l'approche statistique et l'approche structurelle. Une différence essentielle réside dans la représentation de la forme : vecteur de caractéristiques dans l'approche statistique, agencement de primitives pour l'approche structurelle. Nous allons rappeler quelques unes de ces approches couramment utilisées en les classant en 4 groupes : structurelle, statistique, neuronale et stochastique. Cette classification comprend une part d'arbitraire car celle-ci n'est pas unique. Nous nous limitons ici à la présentation de méthodes dites à apprentissage supervisé c'est à dire où la classe des observations est connue lors de l'apprentissage.

## II.2.d L'analyse - Approche structurelle

Les méthodes structurelles se basent sur la structure physique des caractères. Elles cherchent à décomposer le caractère en primitives et à décrire leurs relations. Les primitives sont de type topologique comme un arc, une boucle, etc, et une relation peut être la position relative d'une primitive par rapport à une autre. Généralement le nombre de primitives est assez limité et leur enchaînement peut se décrire par un ensemble de règles d'assemblages. Dans cette approche on distingue plusieurs méthodes :

- méthodes syntaxiques : Ces méthodes sont directement issues de la théorie des langages formels. Elles se basent sur une grammaire formelle. Chaque caractère est représenté par une phrase dans un langage où le vocabulaire est constitué de primitives. La reconnaissance consiste à déterminer si la phrase de la description du caractère peut être générée par la grammaire. L'inconvénient de ces méthodes est l'absence d'algorithmes efficaces pour l'inférence grammaticale directe.
- comparaison des graphes : Cette méthode consiste à construire un graphe où les nœuds contiennent les primitives et les liens entre ces primitives. Ainsi la

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

- reconnaissance consiste à faire une mise en correspondance entre ce graphe et d'autres graphes re— présentant des caractères de référence construits lors de la phase d'appren- tissage. Cette méthode donne des résultats acceptables.
- comparaison de chaînes : Dans ce cas, les caractères sont représentés par des chaînes de primitives. La méthode consiste à mesurer la similitude entre les chaînes du caractère à reconnaître et un modèle de référence par calcul de distance.

## II.2.e Classieur bayésien

L'application des méthodes statistiques bayésiennes à la reconnaissance de forme a été formalisée par Chow. Nous supposons que le problème de repré- sentation des caractères admet un modèle probabiliste. Cette méthode dénit l'appartenance d'un caractère à une classe avec un minimum d'erreur et évalue le risque de la décision à prendre pour un caractère donné. Nous cherchons donc la classe qui maximise la probabilité d'appartenance parmi l'ensemble des classes. L'intérêt de cette méthode est qu'elle repose sur des bases mathématiques très bien déniées. Le modèle Bayésien est un modèle mathématique statistique. Il exprime la probabilité de "A sachant B" en fonction des probabilités de "B sachant A" et de la probabilité de A. Ce modèle Bayésien est très utile pour construire des Réseaux bayésiens qui sont à la fois :

- des modèles de représentation des connaissances
- des «machines à calculer» les probabilités conditionnelles
- une base pour des Systèmes d'aide à la décision

## II.2.f Méthodes des $k$ plus proches voisins (KPPV)

Ces méthodes consistent, étant donné un point  $x$  de  $R$  représentant le caractère à reconnaître, à déterminer la classe de chacun des  $k$  points les plus proches de  $x$  parmi l'ensemble des caractères d'apprentissage et à retenir pour la décision la classe la plus représentée. Si  $k = 1$ ,  $x$  est donc attribué à la classe de son plus proche voisin. De plus ces méthodes s'inscrivent dans le cadre des méthodes non bayésiennes et non paramétriques. Les méthodes des KPPV sont lentes en phase de décision puisqu'il faut calculer un nombre très important de distances. Nous remarquons que nous pouvons utiliser plusieurs types de distances parmi lesquelles la distance tangente qui utilise une approximation de Taylor du premier ordre. Pour cette distance il existe deux formes : la distance tangente uni-latérale et la distance tangente bilatérale. Pour résoudre le problème du temps de calcul des chercheurs ont proposé des variantes sub—optimales nécessitant moins de calculs ?

## II.2.g L'analyse - Méthodes de séparation linéaire et non linéaire

Ces méthodes statistiques sont basées sur la définition des fonctions permettant de séparer partiellement ou totalement des classes représentées par les vecteurs paramètres de leurs échantillons. Ainsi on peut distinguer deux catégories de fonctions :

- fonctions de discrimination linéaires , dans ce cas on parle d'un hyperplan séparateur (surface). Donc le problème consiste en la recherche du meilleur hyperplan permettant de discriminer les classes. Dans le cas de non séparabilité des classes, on passe dans un espace de plus grande dimension où l'on cherche un séparateur linéaire : ce sont les méthodes SVM (Support Vector Machines).
- fonctions de discrimination non linéaires : fonctions linéaires par morceaux, séparation par des quadriques.

## II.2.h L'analyse - Modèles de Markov cachés

Un modèle de Markov caché (HMM) est défini par les données suivantes :

- un ensemble d'états
- un ensemble de symboles observables dans chaque état
- une matrice des probabilités de transitions
- une matrice des probabilités d'observation
- un ensemble de densités de probabilité initiale.

Ce modèle utilise essentiellement des données unidimensionnelles, ce qui a permis leur application directe en traitement de la parole, et ensuite pour l'OCR. On peut distinguer deux types de HMMs :

- les HMMs discrets qui modélisent les observations avec des variables discrètes
- les HMMs semi—continus qui modélisent les observations avec un mélange de gaussiennes

L'inconvénient est que les HMMs sont mono—dimensionnels alors que les images de caractères sont en dimension 2, ce qui donne des résultats parfois peu satisfaisants pour une application directe des HMMs.

## II.2.i Programmation dynamique 2D

C'est une approche par segmentation de l'image de caractère en régions. Cette segmentation s'obtient par un algorithme de programmation dynamique 2D en optimisant la congruence globale par des optimisations partielles sur une des sous-régions (utilisation de champs de Markov, cliques et voisinages d'ordre 1).

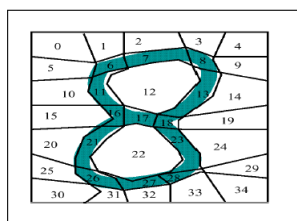


FIGURE 6 – Exemple de segmentation

## II.2.j L'analyse - Le post traitement

Le post traitement intervient après les étapes de reconnaissance de caractère afin d'affiner les erreurs des algorithmes. Le post traitement utilisant des méthodes linguistiques et contextuelles pour réduire le nombre d'erreurs de reconnaissance : systèmes à base de règles, ou méthodes statistiques basées sur des dictionnaires de mots, de syllabes, de N-grammes (séquences de caractères ou de mots). Dans les systèmes industriels, des techniques spécialisées pour certaines zones de texte (noms, adresses postales) peuvent utiliser des bases de données pour éliminer les solutions incorrectes.

## II.2.k L'analyse - Generation

Pour finir un fichier au format demandé est donné à l'utilisateur avec la mise en page pour les meilleurs systèmes.

## III. La reconnaissance par Intelligence artificielle

### III.1. Présentation des réseaux de neurones

#### III.1.a Les notations

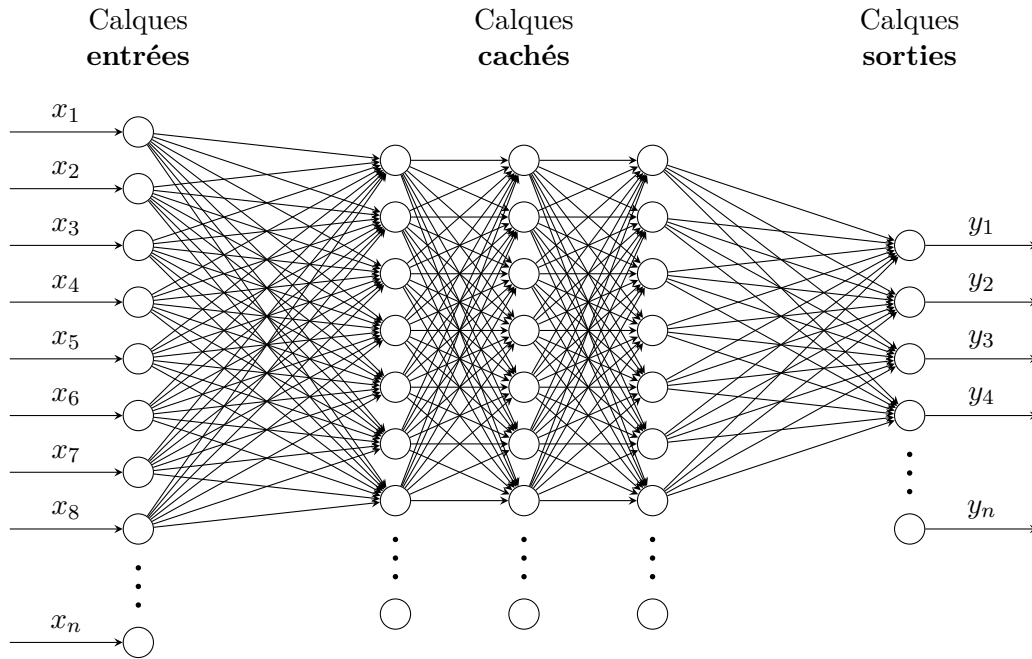


FIGURE 7 – Représentation d'un réseau de neurones

#### Notations utilisées :

- $x_i^{(n)}$  : donnée d'entrée à la colonne  $n$  et ligne  $i$ .
- $y_i^{(n)}$  : donnée de sortie à la colonne  $n$  et ligne  $i$ .
- $w_{ba}^{(n)}$  : poids du neurone  $b$  vers  $a$  et colonne  $n$ .
- $b_i^{(n)}$  : biais à la colonne  $n$  et ligne  $i$ .
- $t_i$  : sortie **voulue** à la ligne  $i$ .
- $X^{(n)}$  : Matrice des entrées à la colonne  $n$ .
- $Y_i^{(n)}$  : Matrice sortie à la colonne  $n$ .
- $W^{(n)}$  : Matrice des poids de la ligne  $n - 1$  à  $n.s$
- $B^{(n)}$  : Matrice des biais à la colonne  $n$ .

On note  $\sigma(x) = \frac{1}{1 + e^x}$  et  $\sigma(x)' = \sigma(x)(1 - \sigma(x))$

$$\text{et } h(n) = \sum_n w_i^{(n)} y_i^{(n-1)}$$



---

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

---

III.1.b Fonctionnement d'un neurone

III.1.c Idée de réseau de neurones

## III.2. La méthode de rétropropagation par calcul du gradient

### III.2.a Qu'est-ce que la rétropropagation

Nous avons vu que les réseaux de neurones “apprenaient”, nous avons aussi montré que cet apprentissage était le coeur des réseaux de neurones et leur force. Mais nous n'avons pas encore expliqué le fonctionnement de cet apprentissage. C'est ce dont cette partie va traiter.

Pour l'instant un réseau de neurone n'est qu'une grosse fonction qui prend  $n$  vecteur  $x_i$  (par exemple tous les pixels d'une image) en entrée et s'entraîne à donner en sortie  $m$  résultats  $y_i$  (par exemple la valeur d'un chiffre).

Sauf que contrairement à des fonctions simples comme une application linéaire, qui est déterminée entièrement par l'image d'une base, on ne peut pas donner une “base d'image de caractères” à notre réseau de neurone. Le fonctionnement d'un réseau de neurone est donc calqué sur la nature, et plus précisément sur le fonctionnement du cerveau : lui travaille grâce à l'apprentissage.

Mais là encore une question se pose, comment définir le mot “apprentissage” en mathématique (et donc en informatique) ? Cette grande question a été résolue en partie en 1980 par David Rumelhart, Geoffrey Hinton, et Ronald Williams dans le papier “**Learning representation by back-propagating errors**”. En appliquant cette méthode au réseau de neurone, il sera capable de s'affiner pour avoir un pourcentage d'erreur faible, voir très faible dans les meilleurs cas.

Le but de l'algorithme est de calculer un minimum local de la fonction “réseau de neurone”. C'est à dire trouver la solution la plus optimale au problème posé, dans notre cas, la reconnaissance d'image. C'est à dire trouver la meilleure combinaison de poids et de biais (étant donné que ce sont les seules variables que nous pouvons changer). Pour calculer l'efficacité de la combinaison de poids et biais, le réseau va devoir calculer son erreur

$$E = \frac{1}{2} \sum_n (t_i - y_i)^2$$

Le but de la méthode par rétropropagation est de faire tendre cette fonction vers un minimum local, au mieux vers 0. Il est important de noter ici, qu'il est possible que le réseau ne puisse pas trouver la solution la plus optimale. Cependant, il est certain qu'il en trouvera une qui est optimisée.



# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

Entraîner et nourrir le réseau qu’une fois sera loin de suffir à faire fonctionner l’algorithme de rétropropagation. Il va falloir répéter cette étape de nombreuses fois. C’est pourquoi la phase d’entraînement et de “nourrissage” du réseau de neurone est séparée en epoch (base de donnée entière), batch (partie d’un epoch) et itérations.

Nous devons séparer l’étape du nourrissage pour, dans un premier temps optimiser le temps de calcul nécessaire, mais aussi pour éviter de “gaver” le réseau. C’est à dire trop le nourrir, menant vers le gavage, c’est à dire un réseau qui est incapable de minimiser sa fonction et donc qui ne convergera jamais vers une solution.

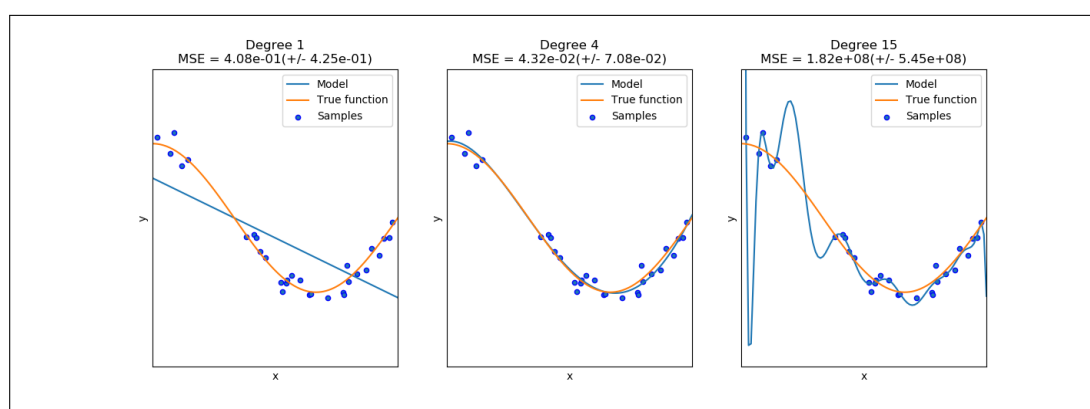


FIGURE 8 – Exemple de nourrissage de réseaux

On voit clairement que les images de gauche et de droite font le travail demandé, mais celui de gauche n’est pas précis du tout, et celui de droite l’est beaucoup trop. Le réseau de droite sera sûrement très lent à entraîner car il est trop précis. Aussi il pourra devenir moins précis pour des cas un peu spéciaux.

Ainsi on ne nourrit pas le réseau avec un batch de la taille de l’epoch, mais plutôt avec plusieurs batch, eux-même ayant plusieurs itérations sur le réseau. Malheureusement, la meilleur découpe des batch et le nombre d’itération optimale n’est pas obtainable pour l’instant par calcul (peut-être prochainement avec un réseau de neurone ?), il faut donc essayer. A chaque fois qu’un batch a été traité par le réseau de neurone, on calcul l’erreur moyenne de chaque batch. C’est cette erreur moyenne qui est utilisée pour le calcul du gradient.

## III.2.b Les mathématiques de la rétropropagation

Le calcul du gradient est le coeur de la rétropropagation, et donc le coeur des réseaux de neurones. Optimiser l'erreur revient à trouver un minimum. C'est à dire trouver un endroit où la fonction "dérivée" s'annule. Or nous n'avons pas de forme dérivable de cette fameuse fonction. Pour optimiser le réseau il suffit donc d'utiliser la méthode des infi-décimaux. C'est à dire calculer  $\nabla E$  (gradient de E). Voici la méthode :

- On calcul l'erreur  $e_i^n = \sigma'(h_i^n)[t_i - y_i]$
- On propage l'erreur calculé vers l'arrière :  $e_j^{(n-1)} = g'^{(n-1)}(h_j^{(n-1)}) \sum_i w_{ij} e_i^{(n)}$ ,  
avec  $e_j^{(n)} = \sum_i [t_i - y_i] \frac{\partial y_i}{\partial h_j^{(n)}}$
- Notre but est de minimiser  $E = \frac{1}{2} \sum_i (t_i - y_i)^2$ ,  
c'est à dire  $\frac{\partial E}{\partial w_{ab}^{(l)}} = \sum_k \frac{\partial x_k^{(n-1)}}{\partial w_{ab}^{(l)}} \sum_i w_{ik}^{(n)} e_i^{(n)}$

Une fois avoir calculé les erreurs, on met à jour les poids et les biais de neurones :

$$w_{ij}^{(n)} = w_{ij}^{(n)} + \lambda e_i^{(n)} x_j^{(n-1)}$$

Le gradient calculé, il suffit de mettre à jour les poids et les biais avec leur valeur respective (calculée avec  $-\nabla E$ ).

Cette méthode est récursive. On ne peut pas calculer les poids du neurones à la ligne  $n$  si on a pas déjà calculé les poids à la ligne  $n + 1$ . D'où cette idée de back propagation. Il faut aller de l'erreur vers les poids de l'entrée. Pour résumer le fonctionnement d'un réseau de neurone par CNN, voici l'algo en pseudo code :

### Pseudo Code - 1: CNN

```

1 Initialiser les poids
2 repeter
3     calculer le "feed-forward"
4     calculer l'erreur
5     calculer le gradient
6     mettre a jour les poids
7 jusqu'a precision > precision voulue

```



---

# Reconnaissance de caractères

TIPE - Planchon, Croce, Durand & Marzook

---

III.3. Application de la technique à la reconnaissance de caractère

III.3.a Utilisation de la BDD MNIST sur les algorithmes

III.3.b Créations de nos propres bases