Université Batna 2
Faculté de Mathématiques et Informatique
Département de Mathématique
Année universitaire 2019-2020

Cours Big data et deep learning

Master 1 SAD

Dr Saadna yassmina

Chapitre 2: Machine learning

Introduction

Le machine learning (ML) est un ensemble d'outils statistiques ou géométriques et d'algorithmes informatiques qui permettent d'automatiser la construction d'une fonction de prédiction f à partir d'un ensemble d'observations que l'on appelle l'ensemble d'apprentissage.

Un modèle de machine learning est un procédé algorithmique spécifique qui permet de construire une fonction de prédiction f à partir d'un jeu de données d'apprentissage. La construction de f constitue l'apprentissage ou l'entraînement du modèle. Une prédiction correspond à l'évaluation f(x) de la fonction de prédiction f sur les variables prédictives d'une observation x.

Dans le ML chaque observation passée d'un phénomène, comme par exemple « *M. Dupont a souscrit un crédit immobilier dans telles et telles conditions ce qui a rapporté tel bénéfice à la banque* » est décrite au moyen de deux types de variables :

- Les premières sont appelées les **variables prédictives** (ou attributs ou paramètres), en l'occurrence l'âge du client, son historique bancaire et ses revenus. Ce sont les variables à partir desquelles on espère pouvoir faire des prédictions. Les p variables prédictives associées à une observation seront notées comme un vecteur $\mathbf{x}=(x1,...,xp)$ à p composantes. Un ensemble de N observations sera constitué de N tels vecteurs $\mathbf{x}(1),..., \mathbf{x}(N)$.
- Une **variable cible** dont on souhaite prédire la valeur pour des évènements non encore observés. Dans notre exemple, il s'agirait du bénéfice ou de la perte estimée pour la compagnie d'assurance suite à l'octroi du prêt à un individu. On notera y cette variable cible avec les mêmes significations pour les indices que pour **x**.

Pour fixer les idées, on considère schématiquement que la valeur observée y de la variable cible résulte de la superposition de deux contributions :

- Une **fonction** $F(\mathbf{x})$ des variables prédictives. C'est donc une contribution entièrement déterminée par les variables prédictives **x** de l'observation. C'est le signal que l'on souhaite mettre en évidence.
- Un **bruit** $\varepsilon(\mathbf{x})$ aléatoire. C'est un fourre-tout qui englobe les effets conjugués d'un grand nombre de paramètres dont il est impossible de tenir compte.

Aussi bien F que ε resteront à jamais inconnus mais l'objectif d'un modèle de *machine learning* est d'obtenir une « *bonne approximation* » du signal F à partir d'un ensemble d'observations. Cette approximation sera notée f, on l'appelle la **fonction de prédiction**.

Le *machine learning* est donc une **démarche fondamentalement pilotée** par les données. Il sera utile pour construire des systèmes d'aide à la décision qui s'adaptent aux données et pour lesquels aucun algorithme de traitement n'est répertorié. Donc, un data scientist peut également être amené à guider le processus d'apprentissage en choisissant certaines données plus significatives que d'autres, phase durant laquelle la visualisation des données multidimensionnelles joue un rôle important.

Algorithmes de ML

- Linear regression
- Logistic regression
- Support vector machine SVM
- Decision tree
- K-means
- KNN
- Neural network……

Application de ML

Voici quelques exemples d'utilisation du *machine learning* parmi les plus communs :

- Détecter des comportements frauduleux lors de transactions financières en ligne.
- Estimer un taux de transformation sur un site marchand en fonction du nombre de clics sur certaines pages.
- Prédire les risques de non-solvabilité d'un client en fonction de ses ressources et de son profil socioprofessionnel.
- Anticiper les intentions de résiliation d'un service en fonction des activités d'un souscripteur.
- Découvrir les préférences d'un client que l'on souhaite retenir pour lui suggérer des produits et des services adaptés à ses goûts et ses besoins.
- Pattern recognition

Les réseaux de neurones

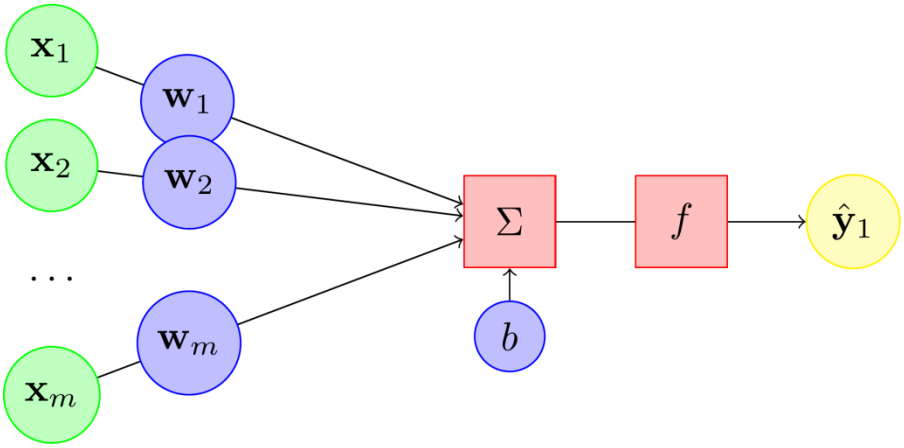
Le cerveau capable d'apprendre et de réaliser des raisonnements complexes est constitué d'un très grand nombre de neurones (environ 10^{15}) reliés entre eux (entre 10^{3}et 10^{4}connexions par neurones). Un réseau de neurones artificiels ou Neural Network est un système informatique s'inspirant du fonctionnement du cerveau humain pour apprendre.

Un réseau neuronal est l'association, en un graphe plus ou moins complexe, d'objets élémentaires, les neurones formels. Les principaux réseaux se distinguent par l'organisation du graphe (en couches, complets. . .), c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones, présence ou non de boucles de rétroaction dans le réseau), par le type des neurones (leurs fonctions de transition ou d'activation) et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques...

Le neurone formel

De façon très réductrice, un neurone biologique est une cellule qui se caractérise par

- des synapses, les points de connexion avec les autres neurones, fibres nerveuses ou musculaires ;
- Des dentrites ou entrées du neurone;
- Les axones, ou sorties du neurone vers d'autres neurones ou fibres musculaires ;
- Le noyau qui active les sorties en fonction des stimulations en entrée.



Par analogie, le neurone formel est un modèle qui se caractérise par un état interne $s \in S$, des signaux d'entrée $x_1, \ldots, x_p$ et une fonction d'activation :

$$s = h(x_1, \ldots, x_p) = g\left(\alpha_0, + \sum_{j=1}^{p} \alpha_j x_j\right) = g(\alpha_0 + \alpha' x)$$

La fonction d'activation opère une transformation d'une combinaison affine des signaux d'entrée, α0, terme constant, étant appelé le biais du neurone.

Cette combinaison affine est déterminée par un vecteur de poids [α0, . . . , αp] associé à chaque neurone et dont les valeurs sont estimées dans la phase d'apprentissage. Ils constituent la mémoire ou connaissance répartie du réseau.

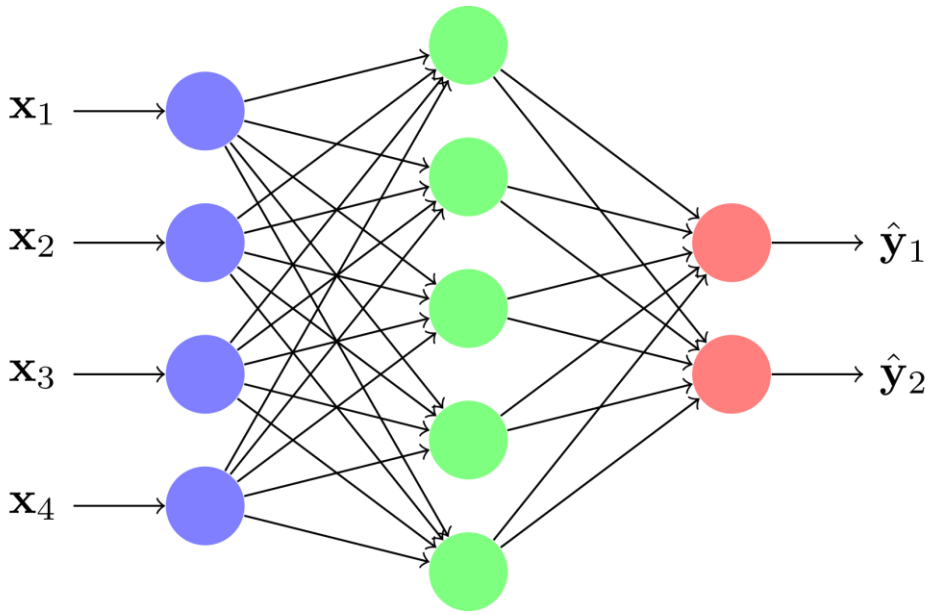
Les différents types de neurones se distinguent par la nature g de leur fonction d'activation. Les principaux types sont :

- linéaire g est la fonction identité,
- seuil g(x) = 1[0,+∞[(x),
- sigmoïde g(x) = 1/(1 + ex),
- ReLU g(x) = max(0, x) (rectified linear unit),
- stochastique g(x) = 1 avec la probabilité 1/(1 + e−x/H), 0 sinon (H intervient comme une température dans un algorithme de recuit simulé).

Les modèles linéaires, sigmoïdaux, ReLU, softmax sont bien adaptés aux algorithmes d'apprentissage impliquant (cf. ci-dessous) une rétro-propagation du gradient car leur fonction

d'activation est différentiable ; ce sont les plus utilisés. Le modèle à seuil est sans doute plus conforme à la réalité biologique mais pose des problèmes d'apprentissage. Enfin le modèle stochastique est utilisé pour des problèmes d'optimisation globale de fonctions perturbées ou encore pour les analogies avec les systèmes de particules.

Perceptron multicouche



- Le perceptron multicouche (PMC) est un réseau composé de couches successives.
- Une couche est un ensemble de neurones n'ayant pas de connexion entre eux.
- Une couche d'entrée lit les signaux entrant, un neurone par entrée xj, une couche en sortie fournit la réponse du système. Selon les auteurs
- La couche d'entrée qui n'introduit aucune modification n'est pas comptabilisée.
- Une ou plusieurs couches cachées participent au transfert.
- Dans un perceptron, un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante.

Un perceptron multicouche réalise donc une transformation des variables d'entrée :

$$Y = f(x_1, \ldots, x_p; \alpha)$$

Où

- α est le vecteur contenant chacun des paramètres $\alpha_{j,k,l}$ de la j ème entrée du $k$ème neurone de la $l$ème couche.
- la couche d'entrée ($l = 0$) n'est pas paramétrée, elle ne fait que distribuer les entrées sur tous les neurones de la couche suivante.

Apprentissage

Supposons que l'on dispose de :

- Une base d'apprentissage de taille n $(x_i^1, \ldots., x_i^p; y_i)$ des variables explicatives $X^1, \ldots \ldots X^P$ et de la variable à prévoir Y.
- un neurone de sortie linéaire.
- Une couche à q neurones dont les paramètres sont optimisés par moindres carrés.

L'apprentissage est l'estimation des paramètres $\alpha_{j=0,p,k=1,q}$ et $\beta_{k=0,q}$ par minimisation de la fonction perte quadratique ou de celle d'une fonction d'entropie en classification :

$$Q(\alpha, \beta) = \sum_{i=1}^{n} Q_i = \sum_{i=1}^{n} [y_i, -f(x, \alpha, \beta)]^2$$

Différents algorithmes d'optimisation sont proposés, ils sont généralement basés sur une évaluation du gradient par rétro-propagation.

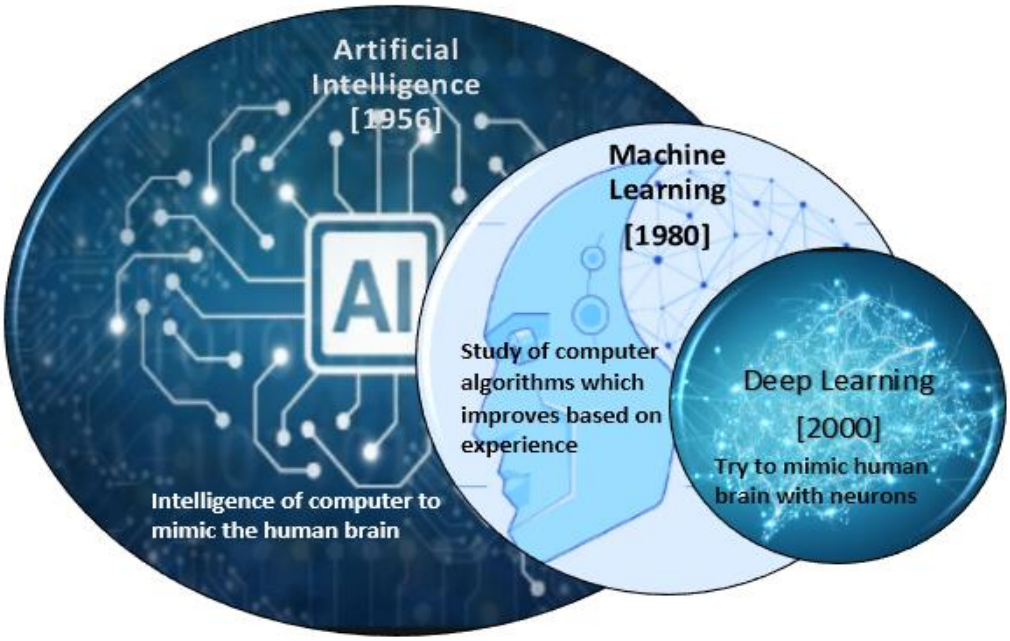
Algorithm 1 Rétro propagation élémentaire du gradient

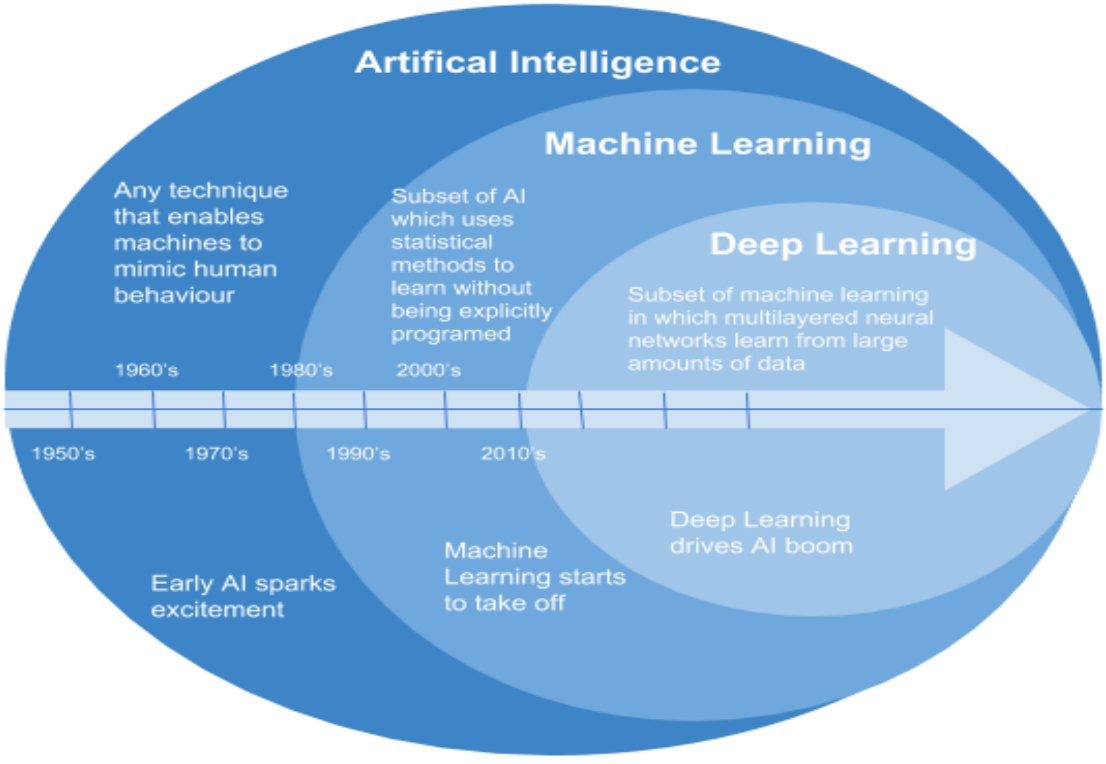
`Initialisation` des poids $b_{jk\ell}$ par tirage aléatoire selon une loi uniforme sur $[0, 1]$.
Normaliser dans $[0, 1]$ les données d'apprentissage.
while $Q >$ errmax ou niter<itermax **do**
  Ranger la base d'apprentissage dans un nouvel ordre aléatoire.
  for chaque élément $i = 1, \ldots, n$ de la base **do**
    Calculer $\varepsilon(i) = y_i - f(x_i^1, \ldots, x_i^p; (b)(i-1))$ en propageant les entrées vers l'avant.
    L'erreur est rétro-propagée dans les différentes couches afin d'affecter à chaque entrée une responsabilité dans l'erreur globale.
    Mise à jour de chaque poids $b_{jk\ell}(i) = b_{jk\ell}(i-i) + \Delta b_{jk\ell}(i)$
  end for
end while

Deep learning

Pour certains types de données, en particulier pour les images, les perceptrons multicouches ne sont pas bien adaptés. En effet, ils sont définis pour les vecteurs comme des données d'entrée, donc, pour les appliquer aux images, il faut transformer les images en vecteurs, en perdant de plus en plus les informations spatiales contenues dans les images. Avant le développement de l'apprentissage profond pour la vision par ordinateur, l'apprentissage était basé sur l'extraction de variables d'intérêt, appelées features, mais ces méthodes nécessitent beaucoup d'expérience pour le traitement d'images. Les réseaux de neurones convolutionnels (CNN) introduits par LeCun ont révolutionné le traitement d'image et supprimé l'extraction

manuelle des features. CNN agit directement sur les matrices, voire sur les tenseurs pour les images à trois canaux de couleurs RGB. Les CNN sont maintenant largement utilisés pour la classification d'images, la segmentation d'images, la reconnaissance d'objets, la reconnaissance faciale…….





Les couches de CNN

Un CNN (convolution neural network) est composé de plusieurs couches :

- Convolutional layers,
- Pooling layers
- Fully connected layers.

a. Convolution Layer

La convolution discrète entre deux fonctions f et g est définie comme suit :

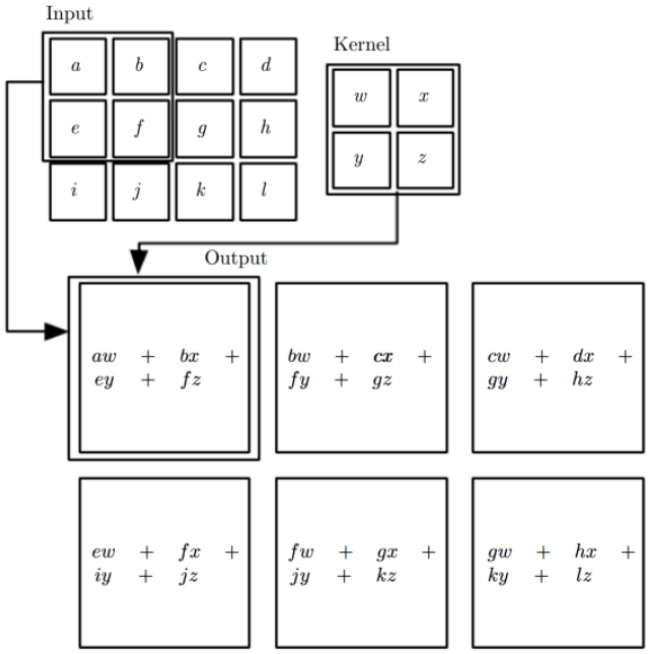
$$(f * g)(x) = \sum_t f(t)g(x+t)$$

Pour les signaux bidimensionnels tels que les images, nous considérons les convolutions 2D

$$(K * I)(i,j) = \sum_{m,n} K(m,n)I(i+n, j+m)$$

K est un noyau de convolution appliqué à un signal 2D (ou image) I.

Le principe de la convolution 2D est de faire glisser un noyau de convolution sur l'image. A chaque position, on obtient la convolution entre le noyau et la partie de l'image qui est actuellement traitée. Ensuite, le noyau se déplace d'un nombre de pixels, s appelé stride. Lorsque le stride est petit, nous obtenons des informations redondantes. Parfois, nous ajoutons également un remplissage à zéro, qui est une marge de taille p contenant des valeurs nulles autour de l'image afin de contrôler la taille de la sortie. Supposons que nous appliquions des noyaux C_0 (également appelés filtres), chacun de taille $k \times k$ sur une image. Si la taille de l'image d'entrée est $W_i \times H_i \times C_i$ (W_i la largeur, H_i la hauteur et C_i le nombre de canaux, généralement$C_i = 3$), la taille de la sortie est $W_0 \times H_0 \times C_0$ où C_0 correspond au nombre de noyaux que nous considérons.

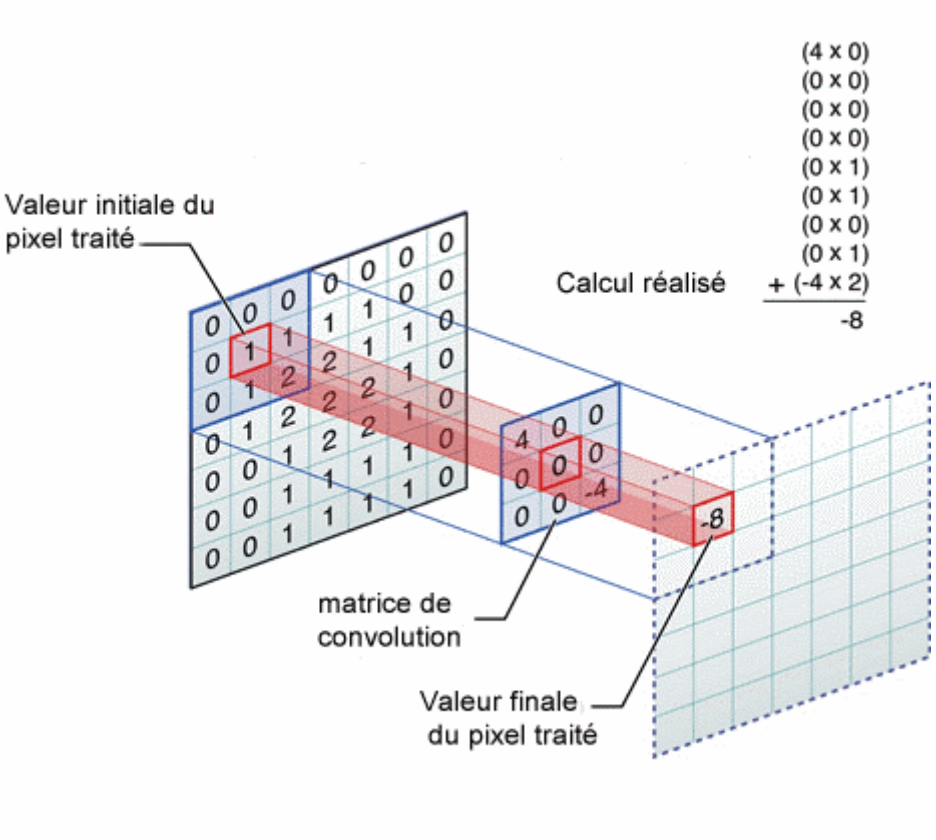


$$W_0 = \frac{W_i - k + 2p}{s} + 1$$

$$H_0 = \frac{H_i - k + 2p}{s} + 1$$

Si l'image comporte 3 canaux et si $K_l = (l = 1, \dots, C_0)$ désigne $5 \times 5 \times 3$ noyaux (où 3 correspond au nombre de canaux de l'image d'entrée), la convolution de l'image I avec le noyau K_l correspond à la formule:

$$k_l * I(i,j) = \sum_{c=0}^{2} \sum_{n=0}^{4} \sum_{m=0}^{4} K_l(n,m,c) I(i+n-2, i+m-2, c)$$



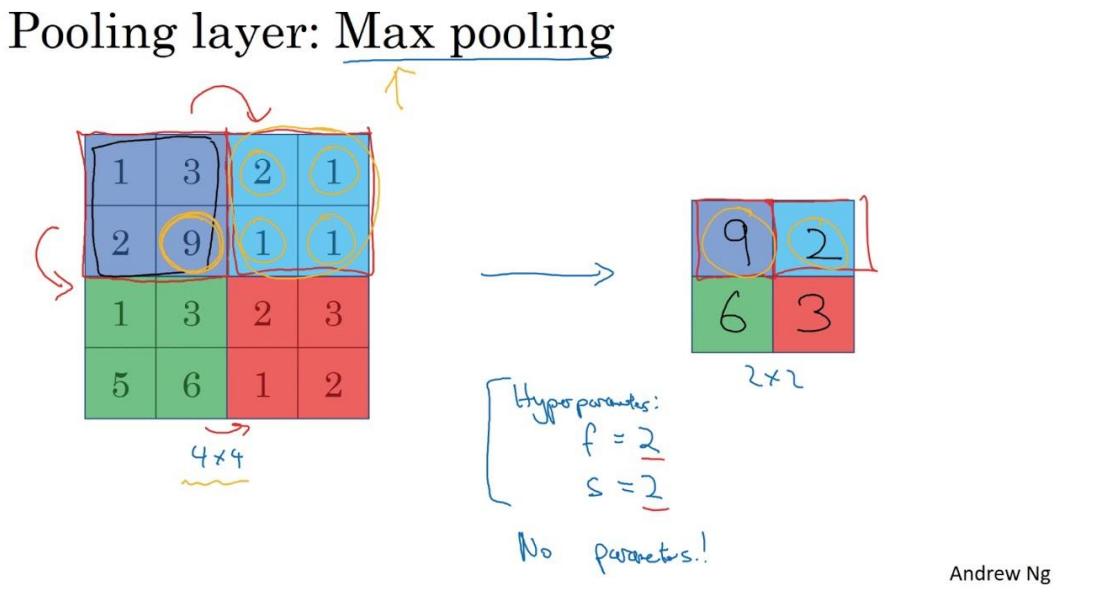
Les opérations de convolution sont combinées avec une fonction d'activation θ (généralement la fonction d'activation Relu): si l'on considère un noyau K de taille $k \times k$, si x est un $k \times k$ patch de l'image, l'activation est obtenue en faisant glisser la fenêtre de taille $k \times k$ et calculer $z(x) = \theta(K * x + b)$, où b est un biais.

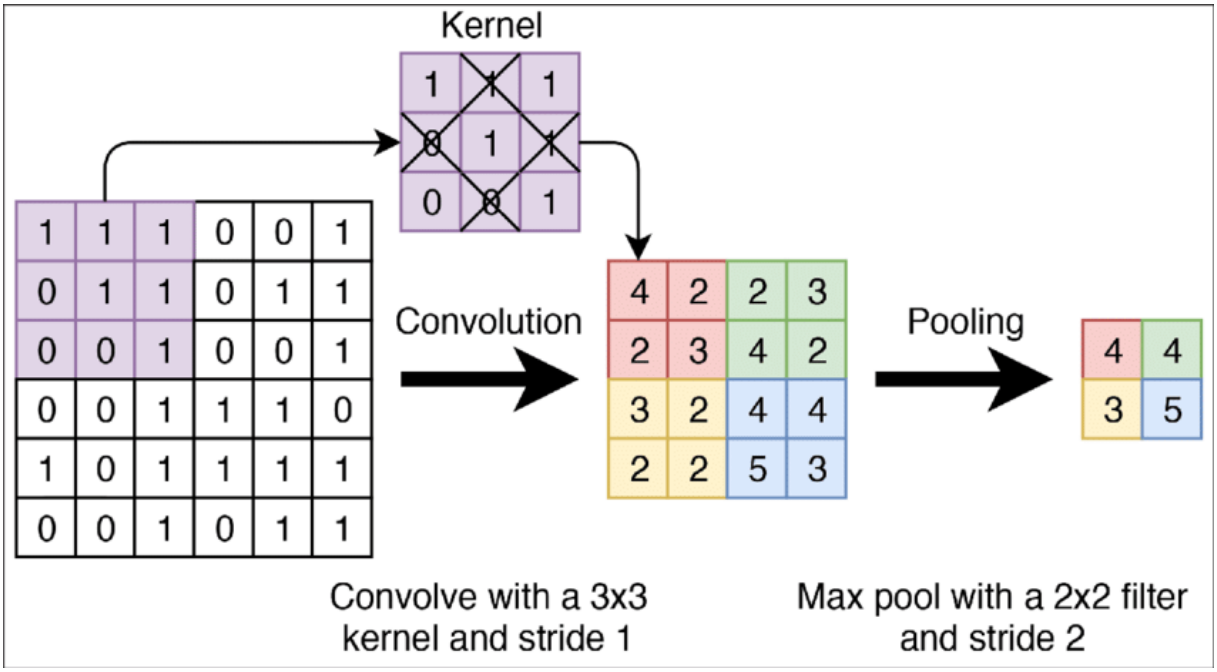
C'est dans la couche de convolution que l'on retrouve la force du CNN, en effet, le CNN va apprendre les filtres (ou noyaux) qui sont les plus utiles pour la tâche que l'on a à faire (comme la classification). Un autre avantage est que plusieurs couches de convolution peuvent être envisagées: la sortie d'une convolution devient l'entrée de la suivante.

b. Pooling layer

CNN dispose également de couches de pooling, qui permettent de réduire la dimension, également appelée subsampling, en prenant la moyenne ou le maximum sur des patchs de l'image (average-pooling ou max-pooling). Comme les couches convolutives, les couches pooling travaillent sur des petits patches de l'image, nous avons également un stride. Si on considère 2*2 patchs, sur lesquels on prend la valeur maximale pour définir la couche de sortie, et un stride s = 2, on divise par 2 la largeur et la hauteur de l'image. Bien entendu, il est également possible de réduire la dimension avec la couche convolutive, en prenant un stride s supérieure à 1, et sans remplissage nul.

Un autre avantage du pooling est qu'il rend le réseau moins sensible aux petites translations des images d'entrée.



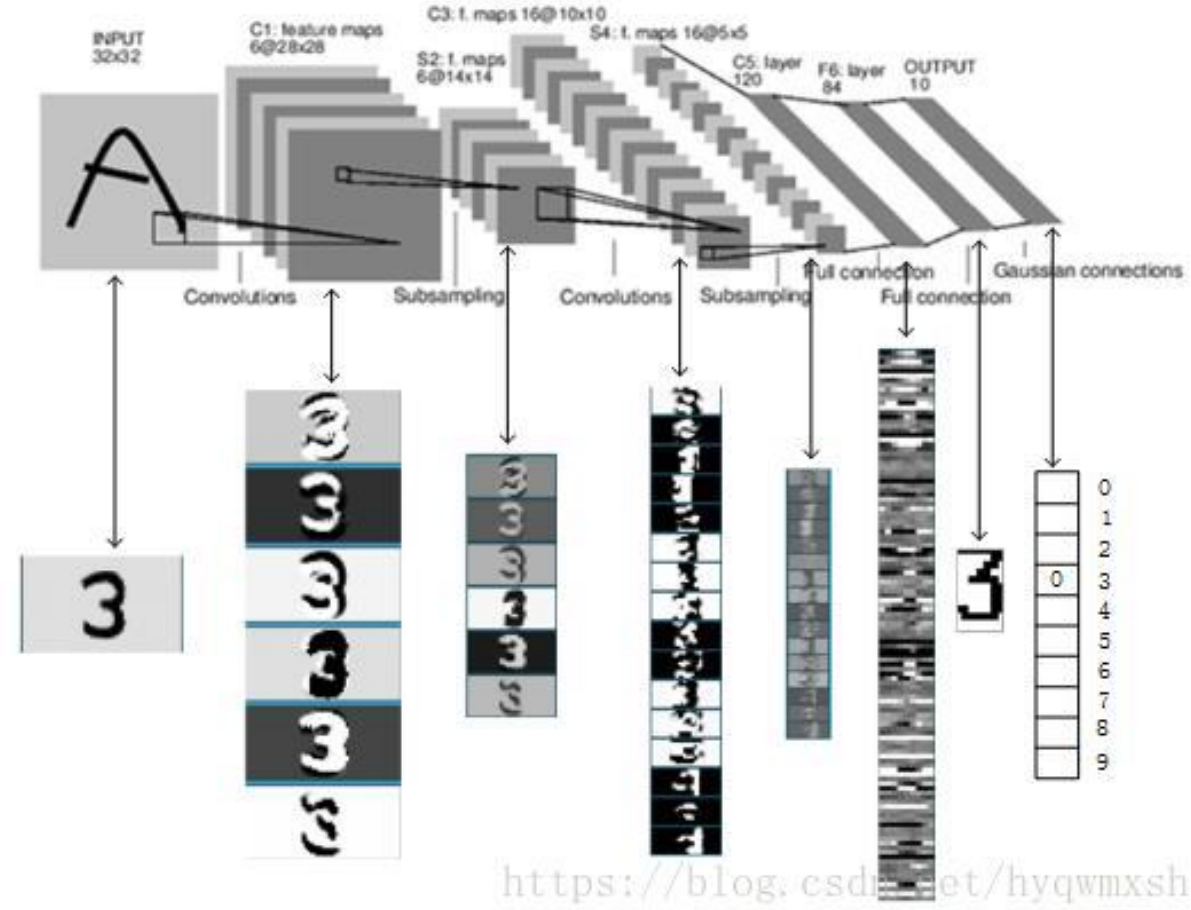


c. Full connected Layer

Après plusieurs couches de convolution et de mise en commun, le CNN se termine généralement par plusieurs couches entièrement connectées. Le tenseur que nous avons en sortie de ces couches est transformé en vecteur puis nous ajoutons plusieurs couches de perceptron.

Architecture

Nous avons décrit les différents types de couches composant un CNN. Nous présentons maintenant comment ces couches sont combinées pour former l'architecture du réseau. Le choix d'une architecture est très complexe et c'est plus une ingénierie qu'une science exacte. Il est donc important d'étudier les architectures qui se sont avérées efficaces et de s'inspirer de ces exemples célèbres. Dans le CNN le plus classique, nous enchaînons plusieurs fois une couche de convolution suivie d'une couche de pooling et nous ajoutons à la fin des couches entièrement connectées. Le réseau LeNet, proposé par l'inventeur du CNN, Yann LeCun est de ce type, comme le montre la figure suivante. Ce réseau était dédié à la reconnaissance de chiffres Mnist. Il est composé uniquement de quelques couches et de quelques filtres, en raison des limitations informatiques à l'époque.



Références

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.".

Ketkar, N., & Santana, E. (2017). *Deep Learning with Python* (Vol. 1). Berkeley, CA: Apress.

Cielen, D., Meysman, A., & Ali, M. (2016). *Introducing data science: big data, machine learning, and more, using Python tools*. Manning Publications Co..

Downey, A. (2012). *Think Python*. " O'Reilly Media, Inc.".