Réseaux de neurones, Word embeddings, Deep Learning… Quelques éclaircissements

Bénédicte PIERREJEAN & Ludovic TANGUY

Thématiques Actuelles de la Recherche en TAL Novembre 2016

Motivations (1)

- Le TAL fait parler de lui dans les médias depuis quelques années :
  - Robots qui parlent
  - Smartphones qui répondent à des blagues
  - Machines qui écrivent des romans
- A chaque fois, on évoque comme raison du succès le "deep learning"
  - nouvelle réponse à tous les problèmes rencontrés par l'intelligence artificielle (et le TAL), annonciatrice de lendemains qui chantent et de F-score élevés, mais qui fait même peur à Stephen Hawking

- *« La technologie du deep learning apprend à représenter le monde. C'est-à-dire comment la machine va représenter la parole ou l'image par exemple »,* pose Yann LeCun, considéré par ses pairs comme un des chercheurs les plus influents dans le domaine. *« Avant, il fallait le faire à la main, expliquer à l'outil comment transformer une image afin de la classifier. Avec le deep learning, la machine apprend à le faire elle-même. Et elle le fait beaucoup mieux que les ingénieurs, c'est presque humiliant !»*

  – Le Monde, 24/07/2015

Motivations (2)

- Les *word embeddings* sont à la mode en TAL
  - En tant que produit des techniques d'analyse distributionnelle (Word2vec et autres)
  - En tant que représentations du lexique utilisées dans des applications
  - Souvent considérés comme étant « du deep learning », alors que c’est un peu plus compliqué que ça

Motivations (3)

- A la base de tout cela, une technologie fondamentale : *les réseaux de neurones*
- On va donc essayer de vous expliquer comment ça marche
- Vous présenter les produits dérivés que sont les *word embeddings*
- Vous expliquer ce qu'est (en gros) le *deep learning*
- Et au fond, qu'est-ce qu'on fait, nous, là-dedans ?

PLAN

- 1/ Petits rappels sur l'apprentissage artificiel
- 2/ Les réseaux de neurones "classiques"
  - Principes et fonctionnement
- 3/ Les méthodes neuronales d’analyse distributionnelle
  - *Word Embeddings*
- 4/ L'apprentissage profond (*deep learning*)
  - Principes techniques et épistémologiques
- 5/ Quelques remarques conclusives

1/ L'apprentissage artificiel : petites piqûres de rappel

Apprentissage artificiel

- "[…] it is necessary to specify methods of problem solution in minute and exact details, a time-consuming and costly procedure. Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort." (Samuel, 1959)
- "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience." (Mitchell 1997)
- Autrement dit, la branche principale de l'IA qui définit des méthodes pour résoudre des problèmes sans une réponse algorithmique, par induction sur des ensemble de données

Une tâche centrale : la classification

- "Most of science can be viewed as attempts to find useful ways to categorize phenomena." (Solomonoff 1957)
- Classer = attribuer une catégorie (dans une liste fermée) à un objet
  - Tagger des mots, filtrer des spams, reconnaître un caractère manuscrit, transcrire la parole, etc.
- Autres tâches connexes :
  - Régression : attribuer un score numérique à une entité
  - Clustering : proposer des catégories sans les avoir prédéfinies (regrouper des objets similaires)

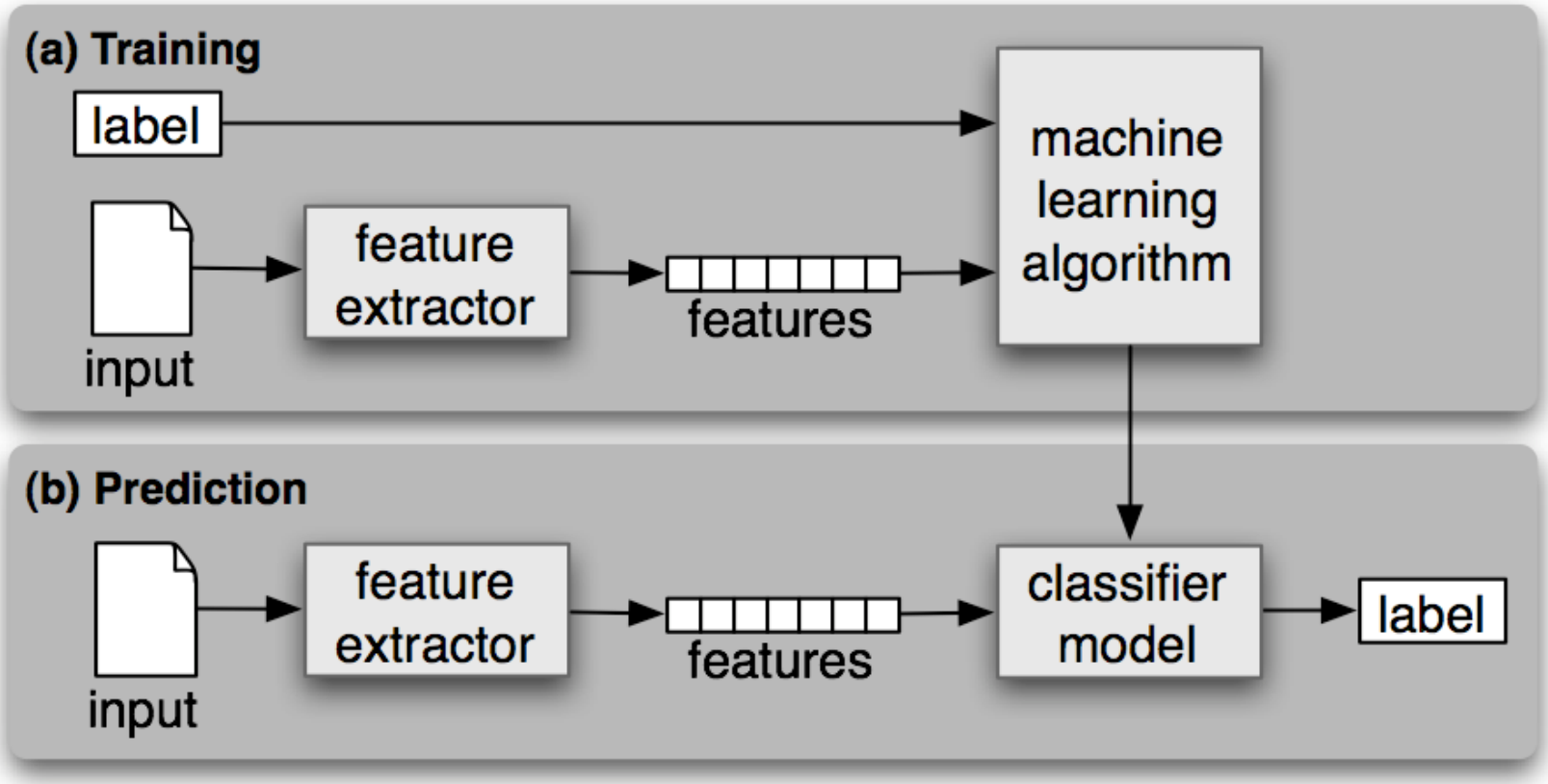
L'approche classique

- Approche procédurale
- Définir des règles de classification à la main
  - Si un objet a telles ou telles caractéristiques c'est un X, sinon c'est un Y, etc.
  - Exemple : le tagging à l'ancienne

L'approche par apprentissage

- Classification par apprentissage *supervisé*
- Procédure :
  - Réunir un ensemble de cas déjà résolus
    - E.g. faire étiqueter les mots d'un texte à la main
  - Expliciter/calculer les caractéristiques supposées pertinentes d'un objet
    - Taille d'un mot, forme (chaîne de début, de fin), position, catégories des mots voisins, etc.
  - Appliquer une méthode d'apprentissage qui va rechercher automatiquement ce qui relie les caractéristiques d'un objet à sa catégorie
    - Et représenter ces liens dans un modèle prédictif
  - Appliquer ce modèle aux nouveaux cas à traiter

(D’après Bird et al. 2009)



Différents modèles

- Symboliques
  - E.g. Règles de décision, arbres de décision
- Bayésiens
  - E.g. Bayésiens naïfs, réseaux bayésiens…
- Régression
  - E.g. Régression logistique (entropie maximale)
- Géométriques
  - E.g. SVM
- Neuronaux
  - On y vient

2/ Les réseaux de neurones

Réseaux de neurones : un peu d'histoire (1)

- Réseaux de neurones = ensemble d'algorithmes ayant pris pour modèle le cerveau humain
- McCulloch & Pitts (1943) : premier modèle conceptuel du réseau de neurones
- Donald Hebb (1949) : *The Organization of Behavior*
  - Chaque fois qu'une connexion entre neurones est utilisée ➔ renforcée
  - Concept fondamental de l'apprentissage humain
- Années 1950 : avancements technologiques, possible de simuler réseau de neurones

Réseaux de neurones : un peu d'histoire (2)

- Widrow & Hoff (1959) : premiers modèles développés utilisant des réseaux de neurones
- Manque de financements, laissé un peu de côté
- Premiers succès : promesses non tenues, questions philosophiques (quels effets de ces nouvelles « machines pensantes »)
- Regain d'intérêt pour les réseaux de neurones dans les années 80

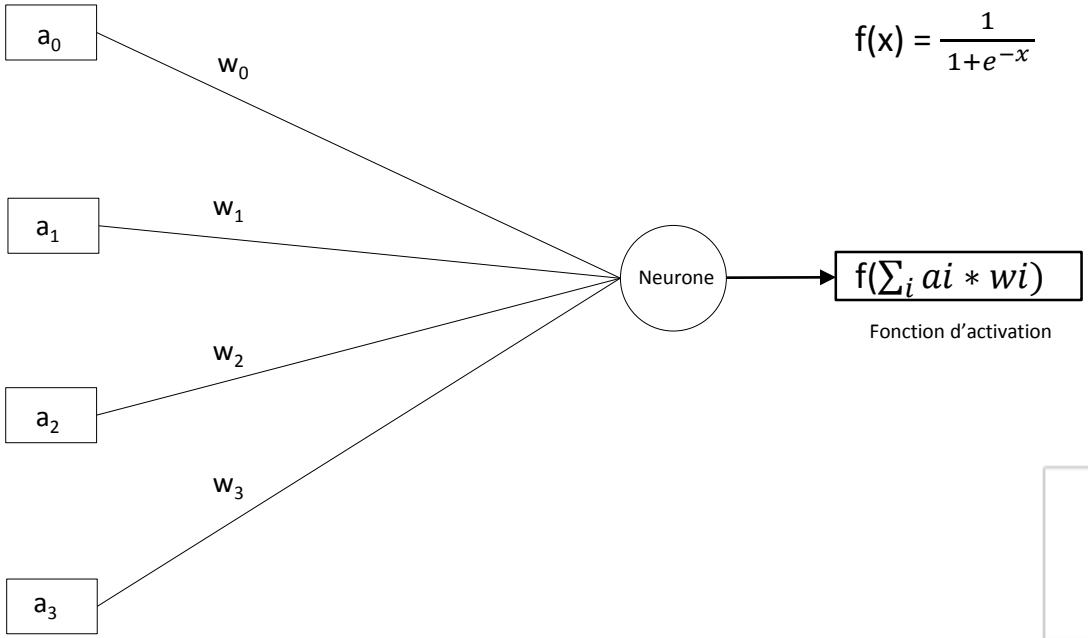
Réseaux de neurones : un peu d'histoire (3)

- Abandon dans les années 90 :
  - Problèmes de vitesse d'apprentissage (notamment par rapport aux SVM)
  - Dépendant des développements du matériel informatique
- Retour en grâce dans les années 2000 :
  - Machines plus rapides, calcul parallèle
  - Succès en traitement d'image, de parole, puis en TAL général
  - Et bien sûr le *Deep Learning*

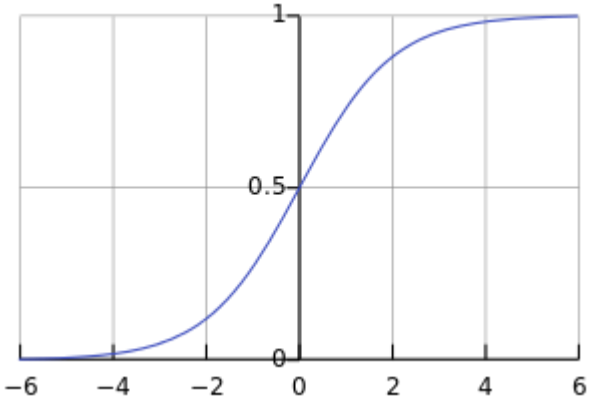
Neurone : définition

- Unité qui prend *n* entrées et produit une seule sortie
- Le neurone en action a lui-même une valeur d'activation
  - Activé ou non (0/1 modèle binaire)
  - Valeur numérique quelconque (modèle continu)
- L'activation d'un neurone dépend :
  - Des valeurs en entrée (a_i)
  - Des poids de chaque connexion (w_i)
- Activation du neurone :
  - Somme des produits entrée*poids
  - Fonction d'activation

Neurone : illustration



$$f(x) = \frac{1}{1+e^{-x}}$$



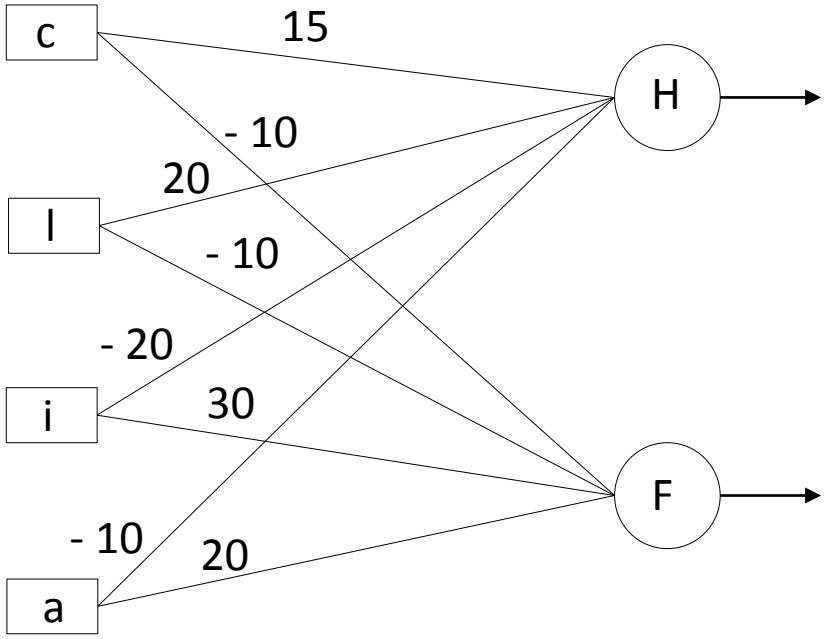
https://en.wikipedia.org/wiki/Sigmoid_function

Witten & Frank (2005)

Perceptron

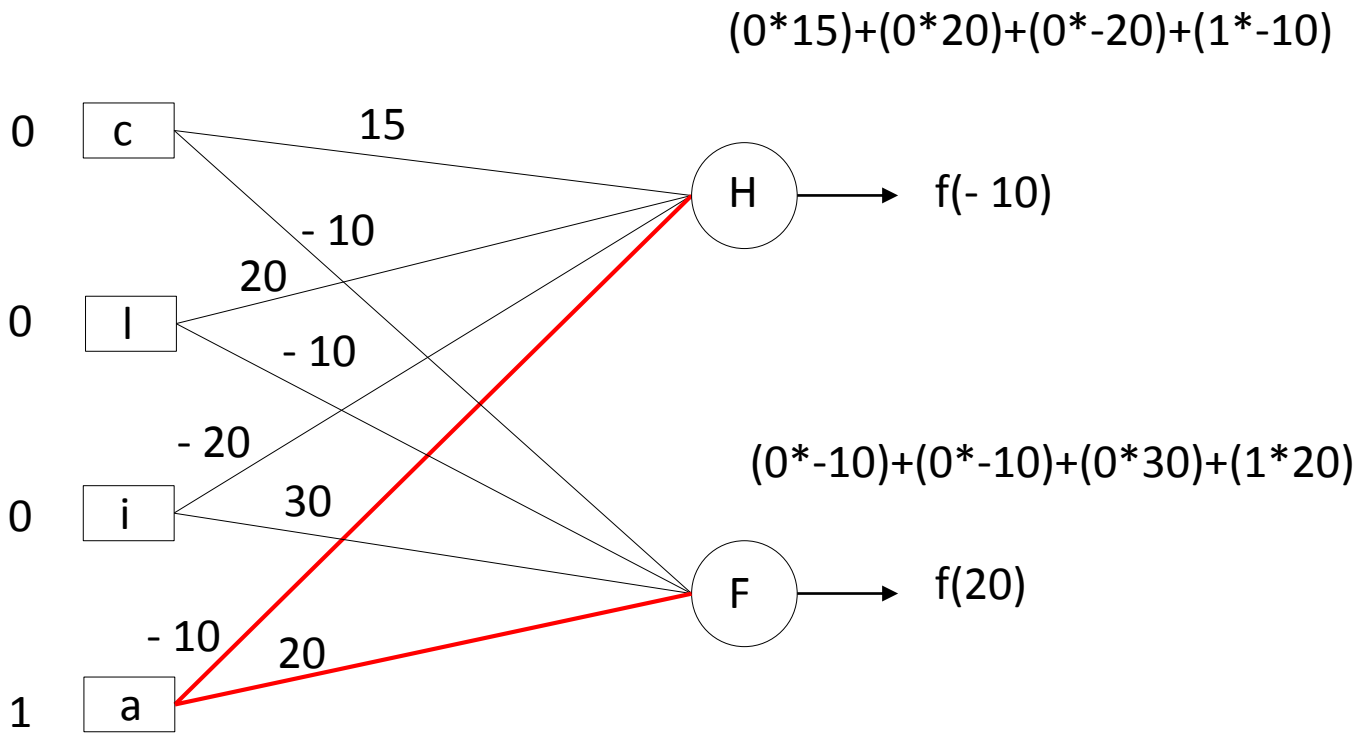
- Réseau de neurones le plus simple
- Deux couches :
  - Entrée : un point d'entrée pour chaque attribut
  - Sortie : un neurone par valeur possible
- Tous les nœuds d'entrée connectés à couche de sortie
- Poids pour chaque connexion nœud entrée/nœud sortie
- Equivalent à régression linéaire/logistique

Perceptron mono-couche: exemple (1)



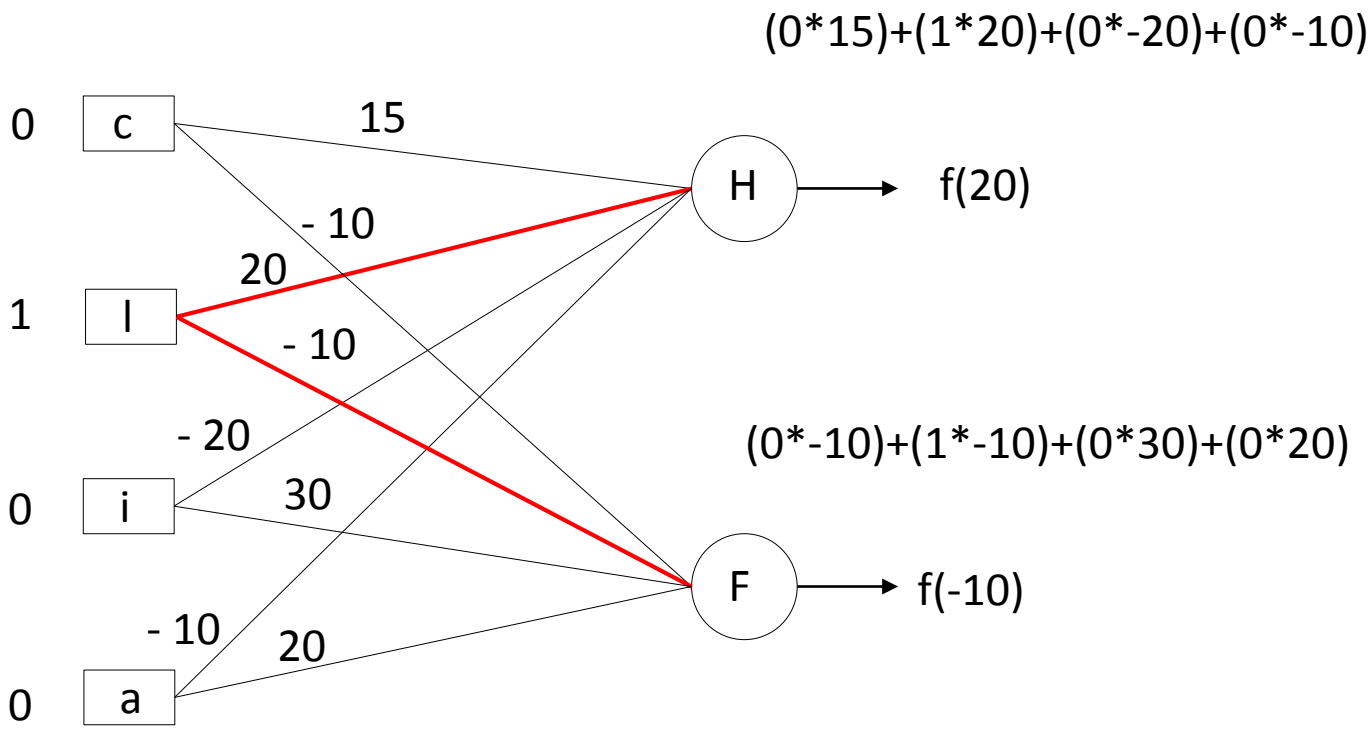
Witten & Frank (2005)

Perceptron mono-couche: exemple (2)



Witten & Frank (2005)

Perceptron mono-couche: exemple (3)



Witten & Frank (2005)

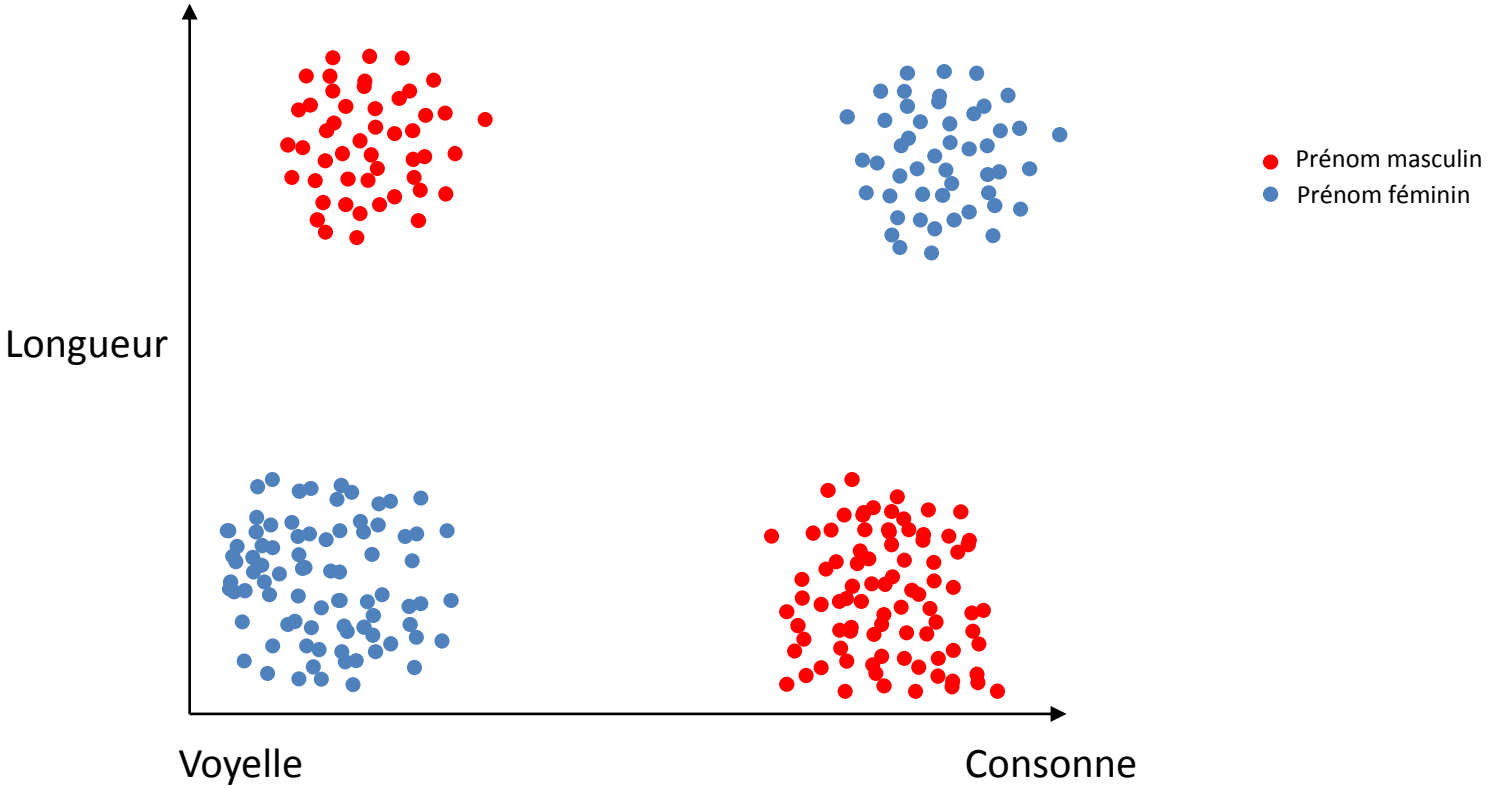
Perceptron multi-couches : structure

- Ensemble d'unités entrées/sorties connectées
- Poids associé à chaque connexion
- Couche d'entrée :
  - Un point d'entrée pour chaque attribut
- Couche cachée :
  - Pas de connexion directe à l'environnement (input/output)
  - Sorties peuvent être transmises à une autre couche cachée
  - Dernière couche cachée : sorties données à la couche de sortie
- Couche de sortie
  - Prédiction du réseau de neurones
- Permet de séparer ce qui n'est pas séparable linéairement

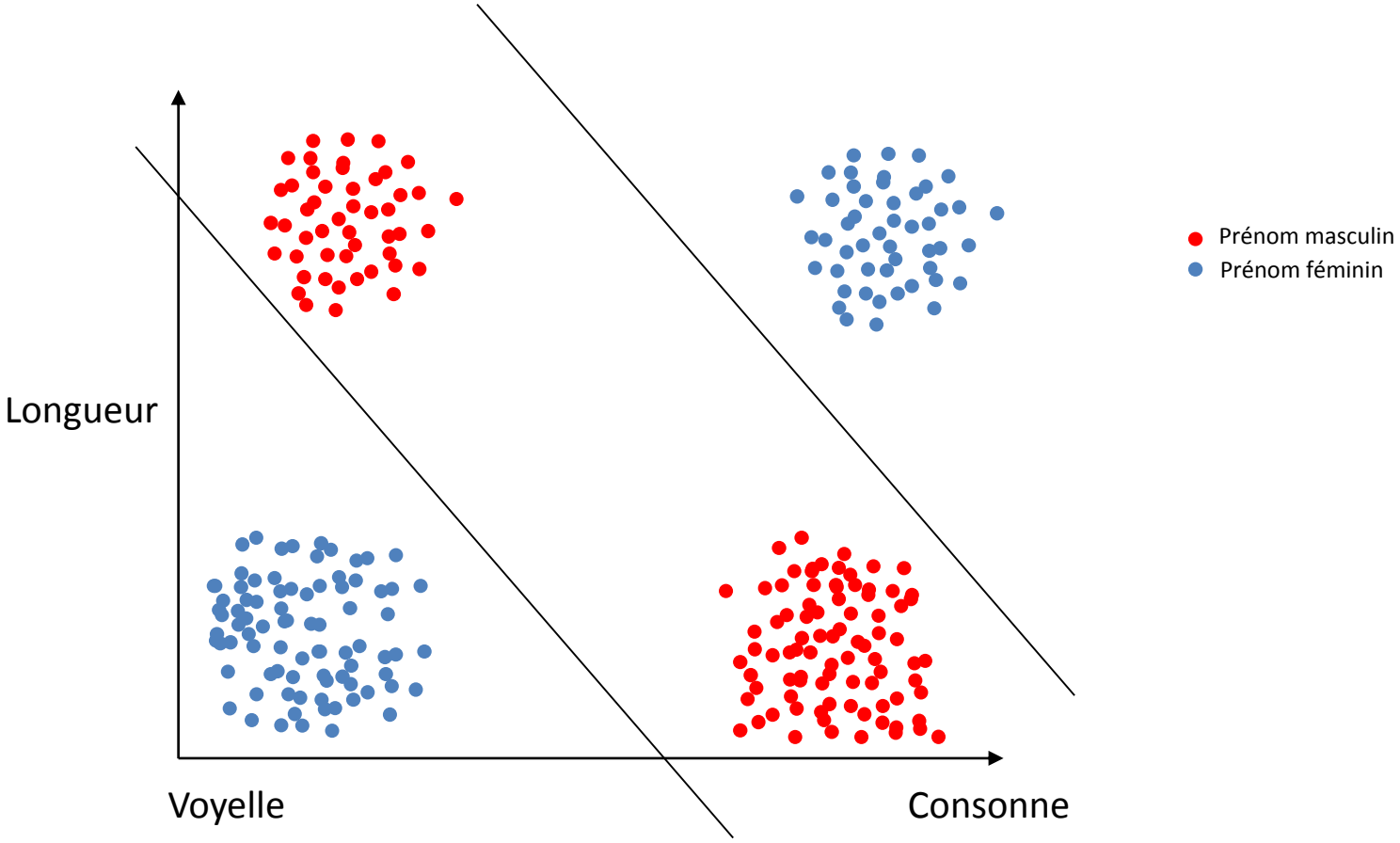
Han & Kamber (2006)

Witten & Frank (2005)

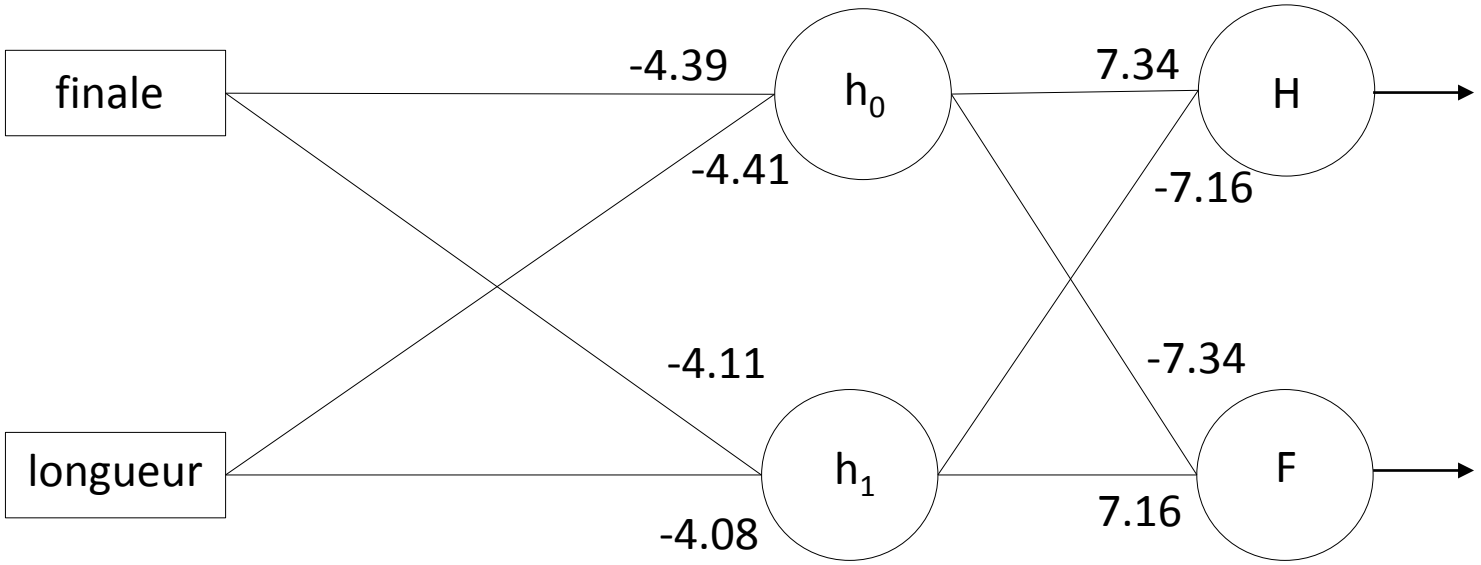
Perceptron multi-couche : exemple



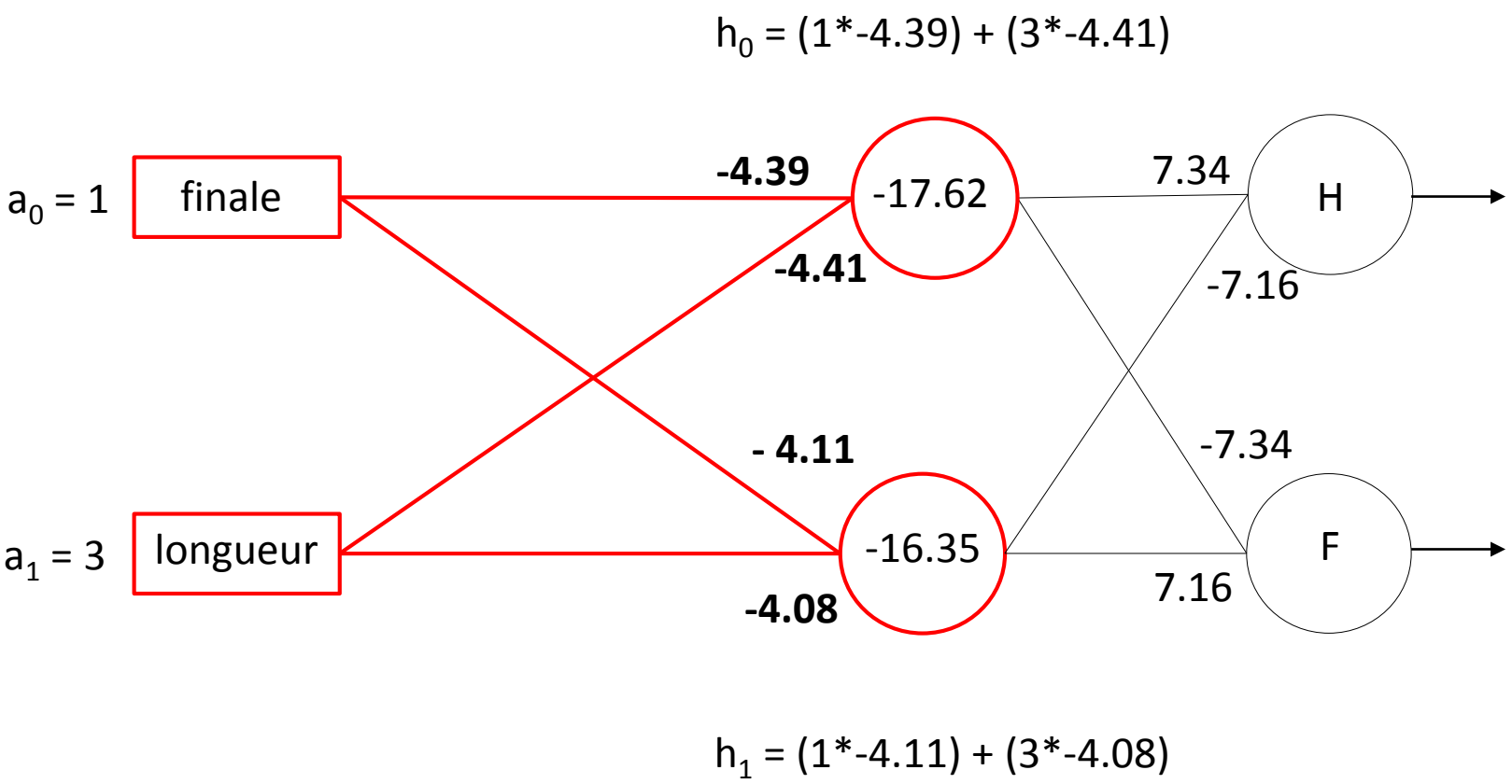
Perceptron multi-couche : exemple



Perceptron multi-couche : exemple

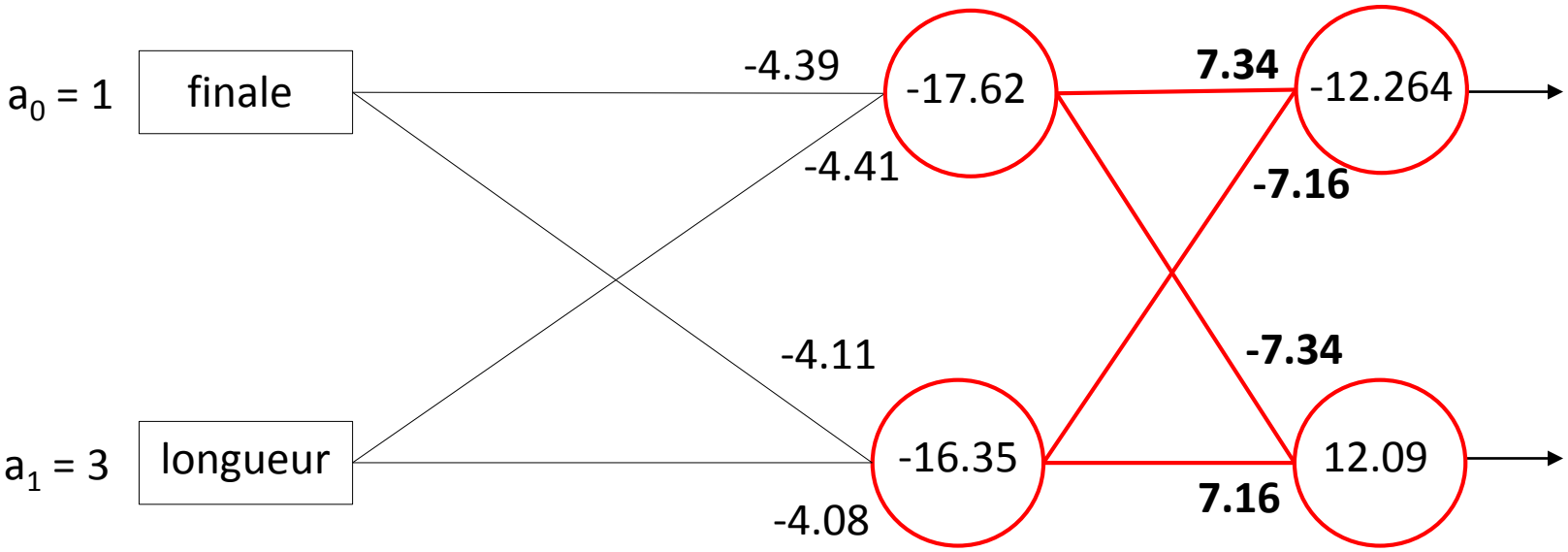


Perceptron multi-couche : exemple



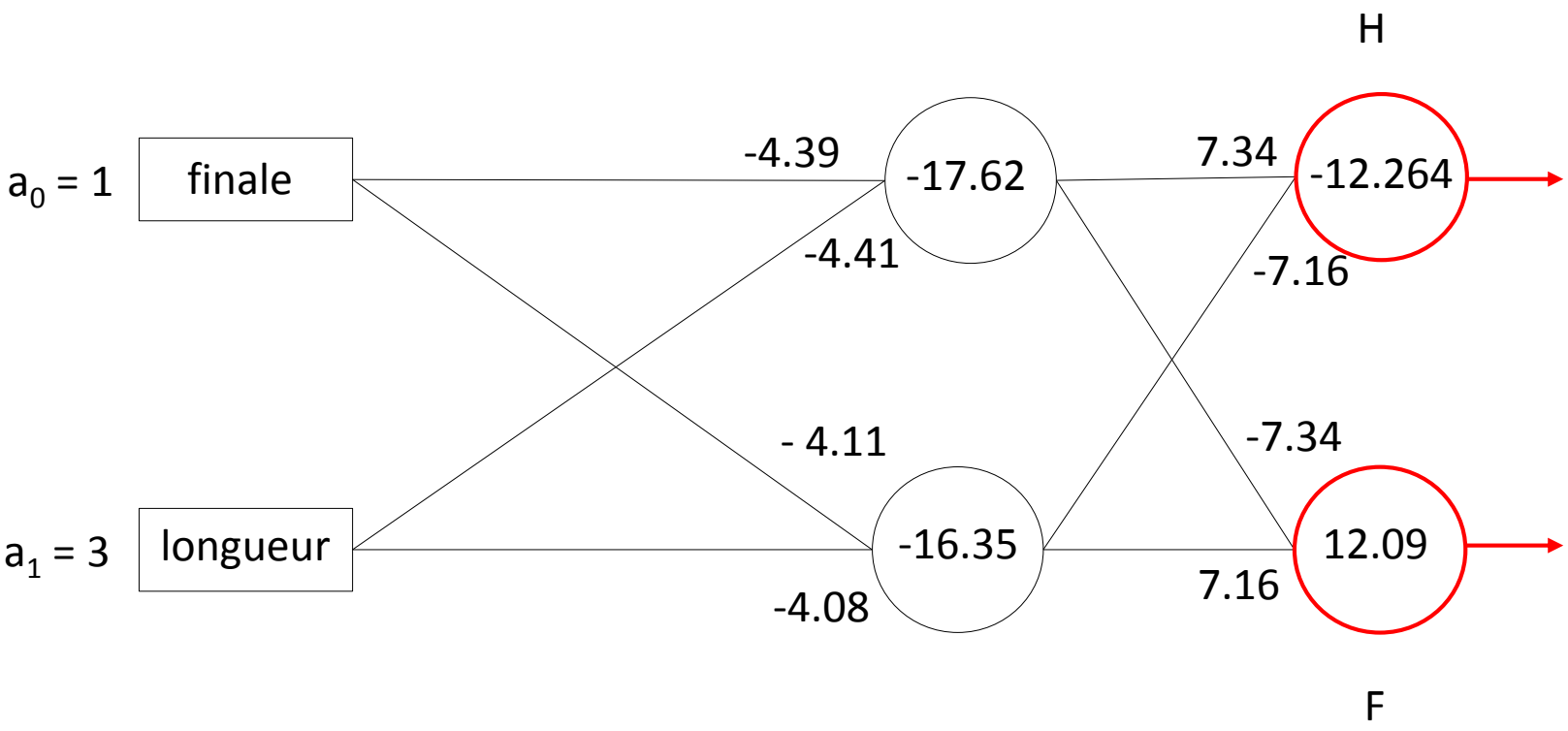
Perceptron multi-couche : exemple

H = (-17.62*7.34) + (-16.35*-7.16)



F = (-17.62*-7.34) + (-16.35*7.16)

Perceptron multi-couche : exemple

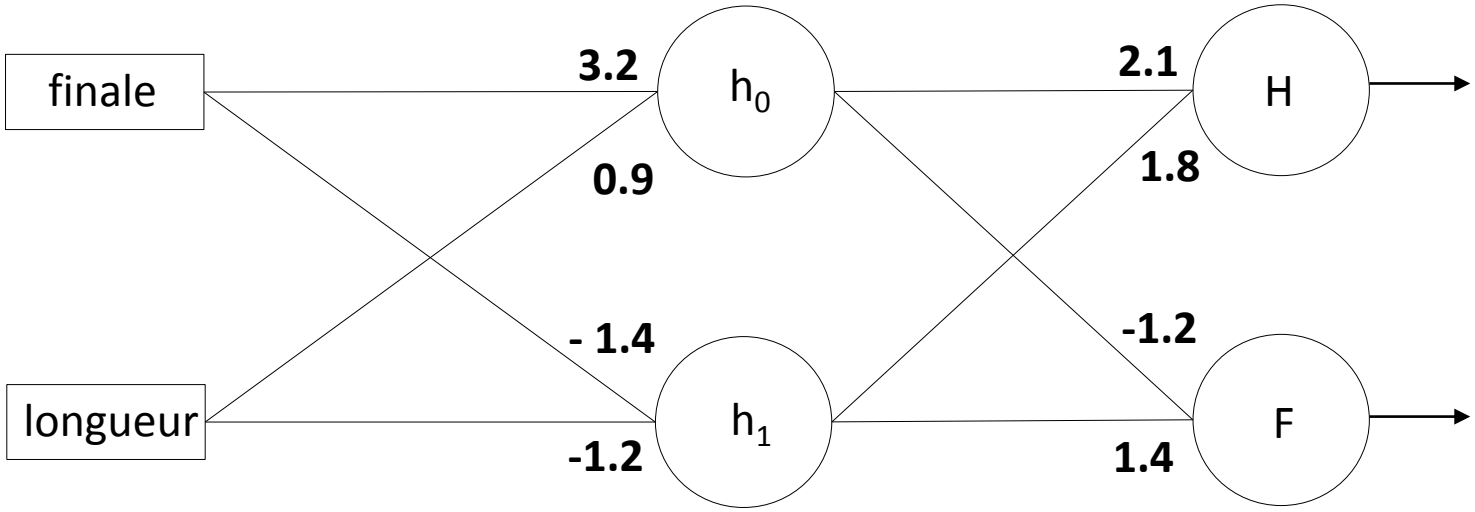


Apprentissage (1)

- But : déterminer les poids pour chaque connexion du réseau
  - En se basant sur les données d'apprentissage (entrées et sorties connues)
- Algorithme de base : rétro-propagation
  - Comparaison entre prédiction donnée par le réseau et résultat attendu
  - Ajustement des poids pour diminuer le taux d'erreur (de la couche de sortie à la couche d'entrée)

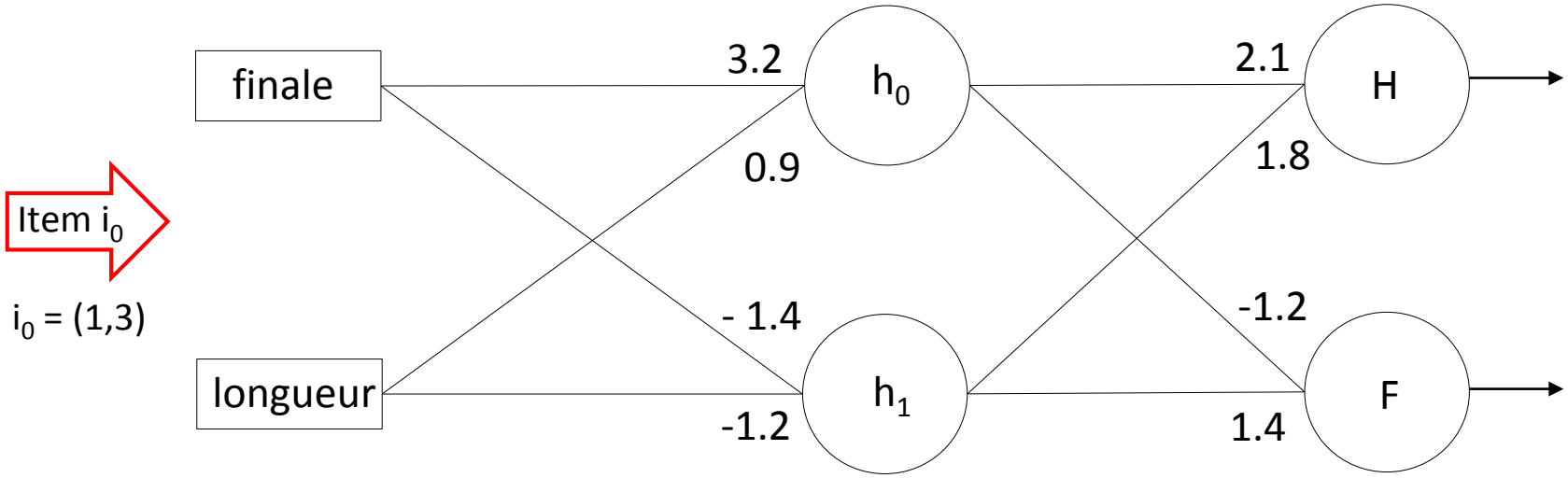
Han & Kamber (2006)

Apprentissage (2)



- Poids définis de manière aléatoire

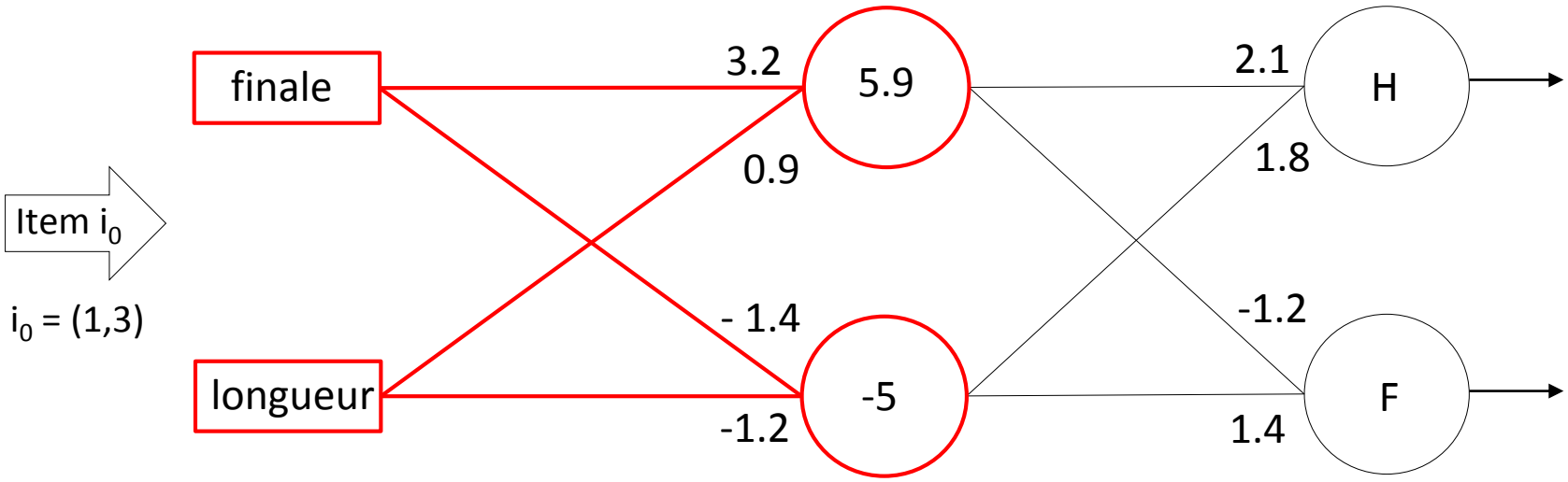
Apprentissage (2)



- Item donné en entrée

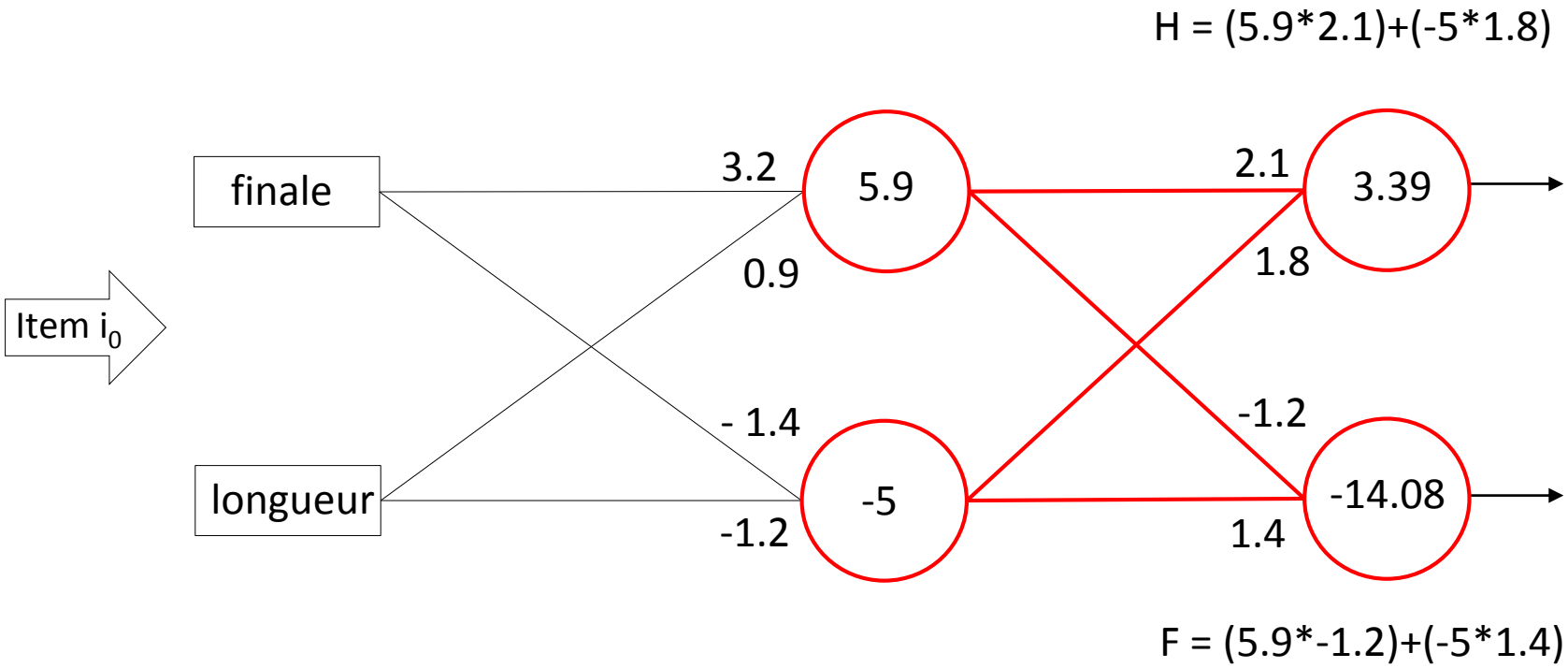
Apprentissage (2)

$h_0 = (1*3.2)+(3*0.9)$

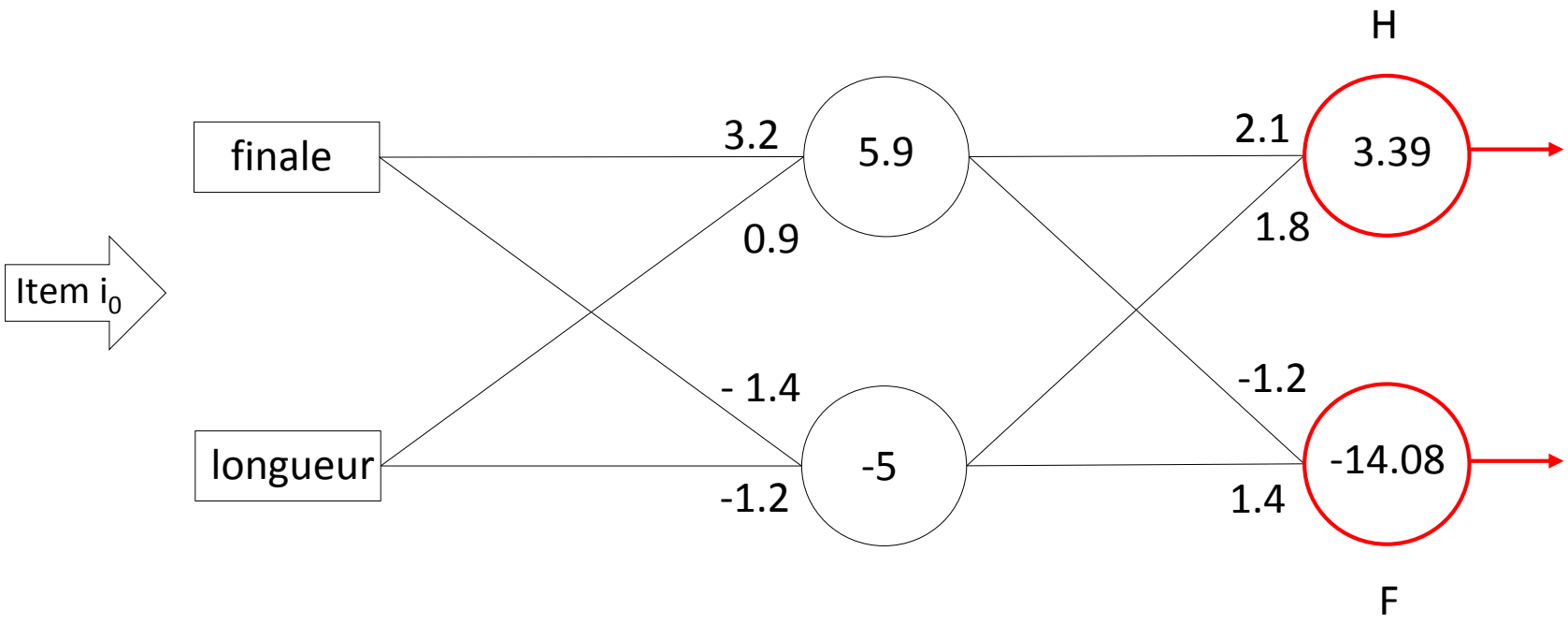


$h_1 = (1*-1.4)+(3*-1.2)$

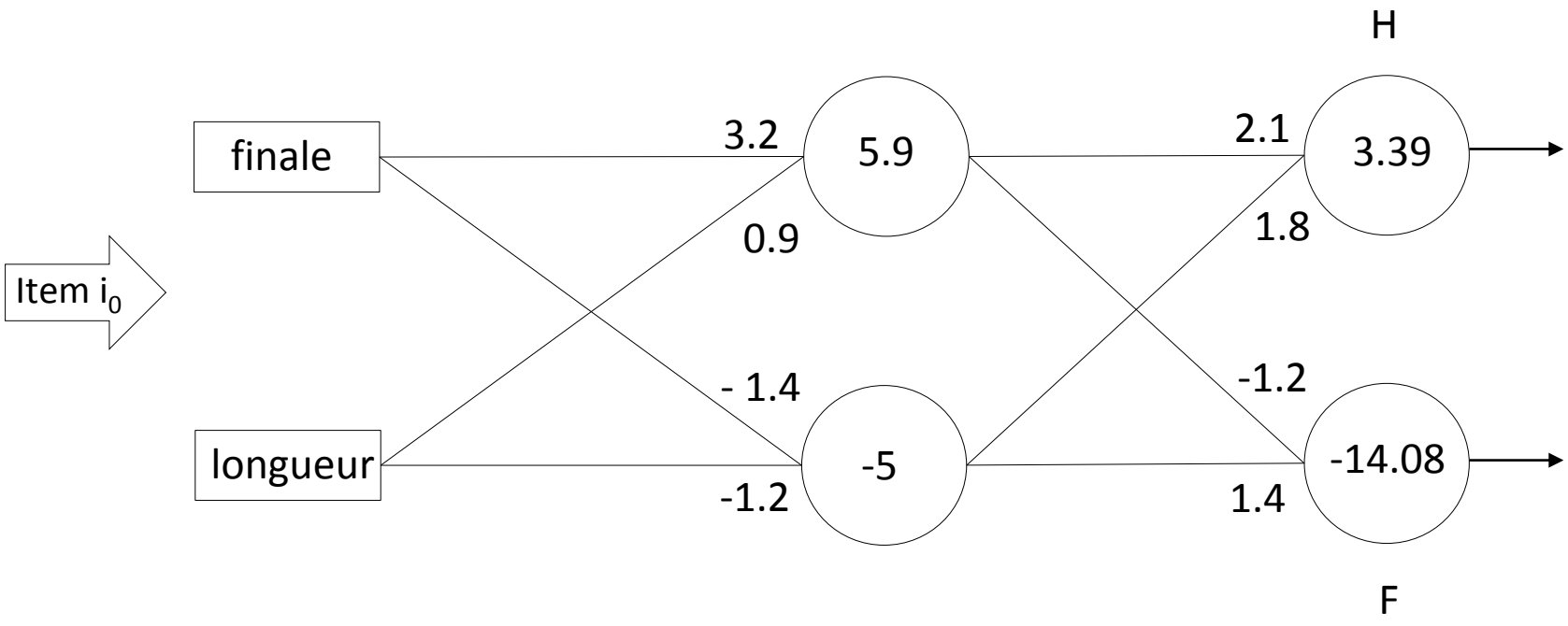
Apprentissage (2)



Apprentissage (2)



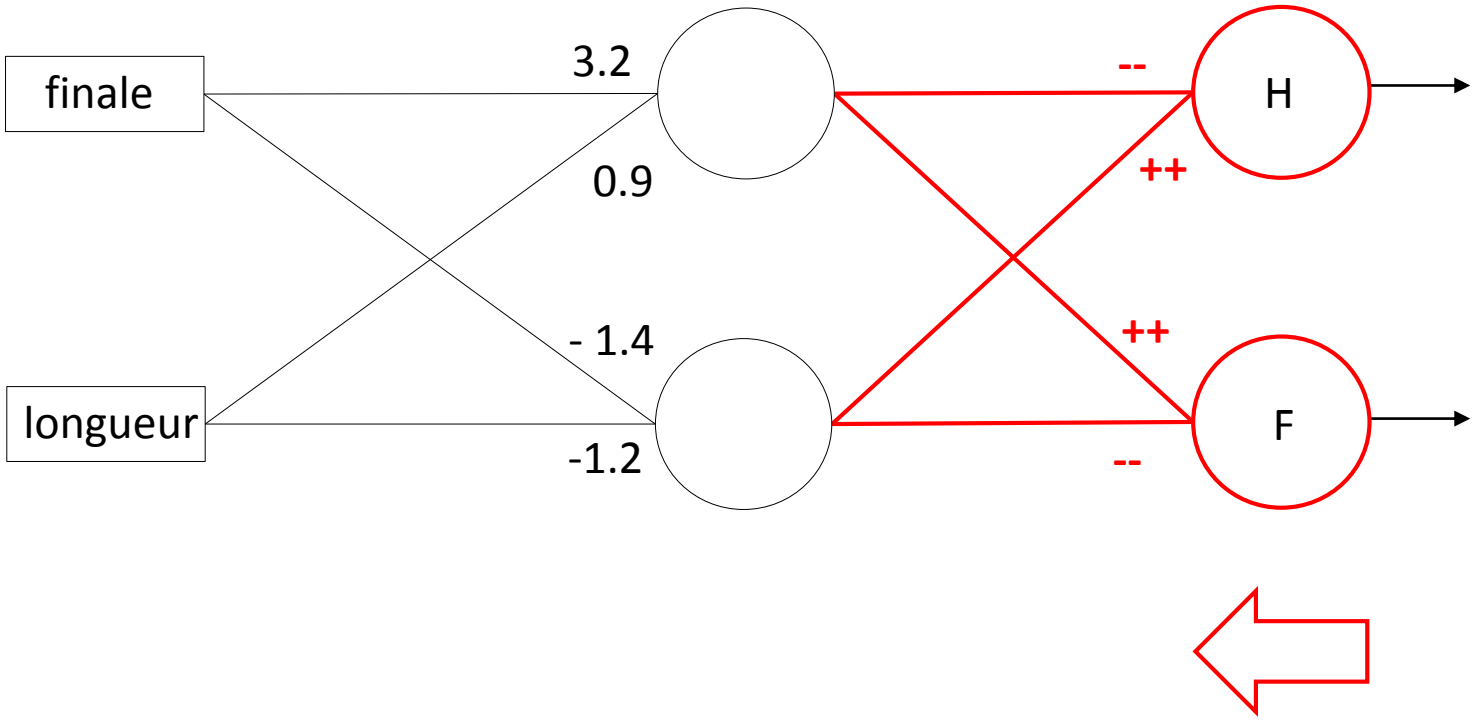
Apprentissage (2)



Résultat obtenu ⇨ (1,3) = H

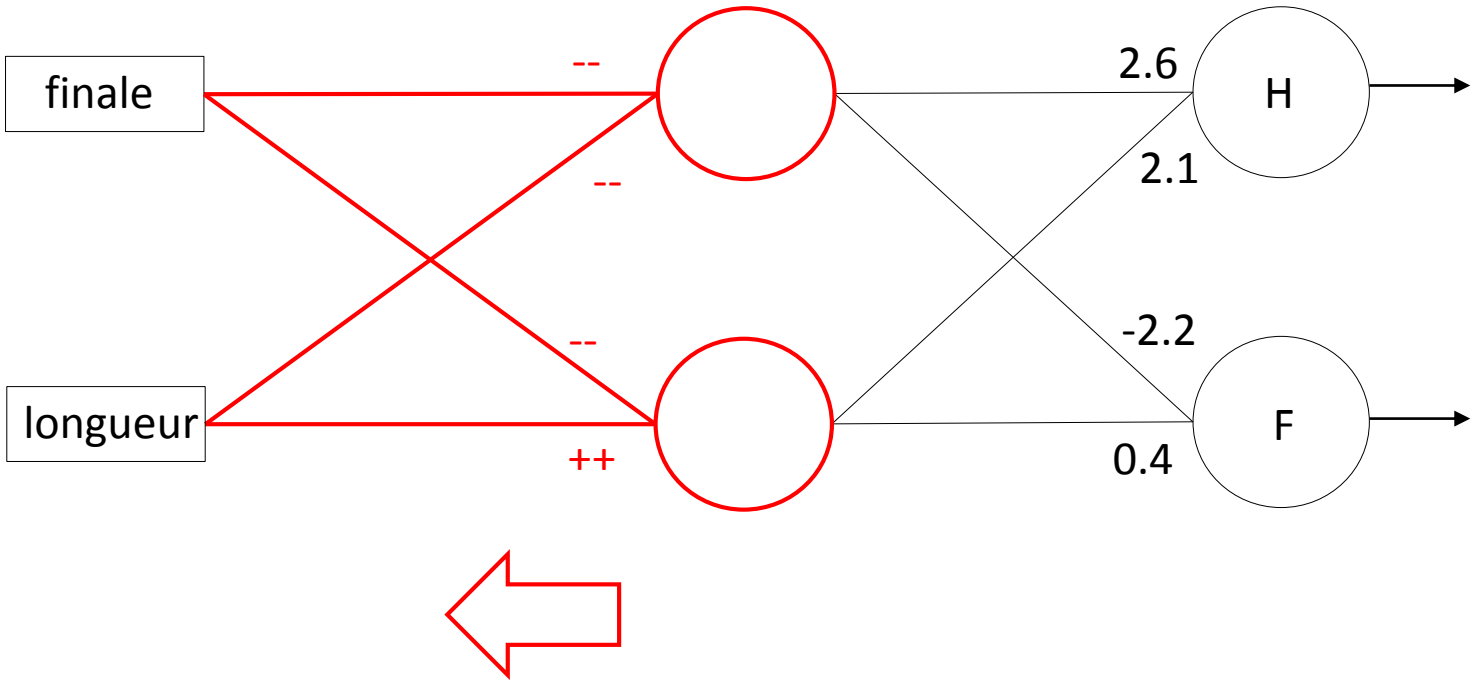
Valeur attendue ⇨ (1,3) = F

Apprentissage (2)



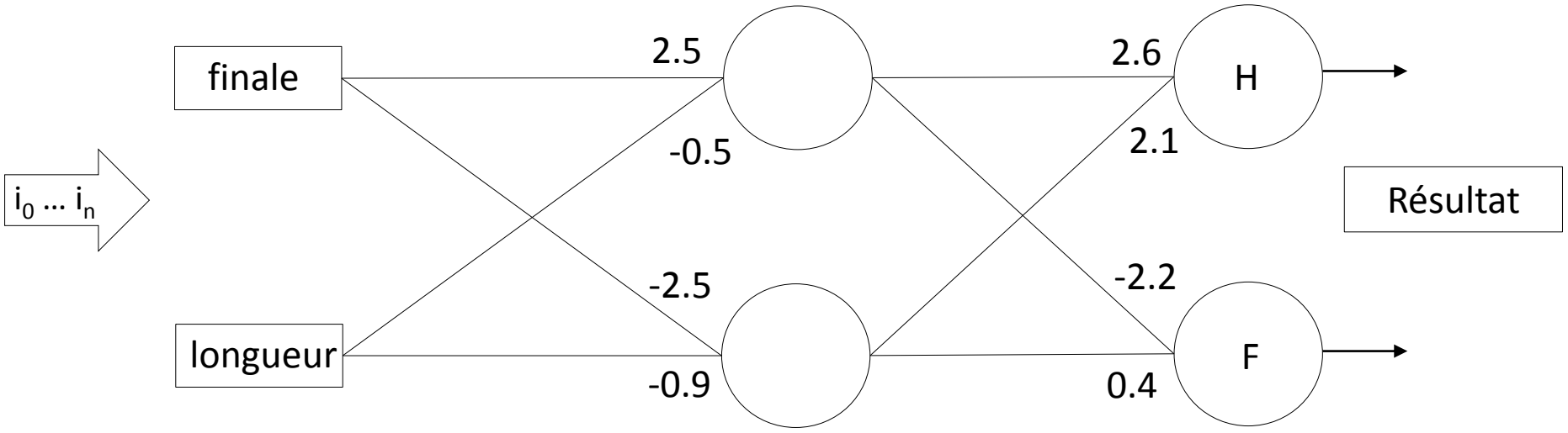
- Ajustement des poids

Apprentissage (2)



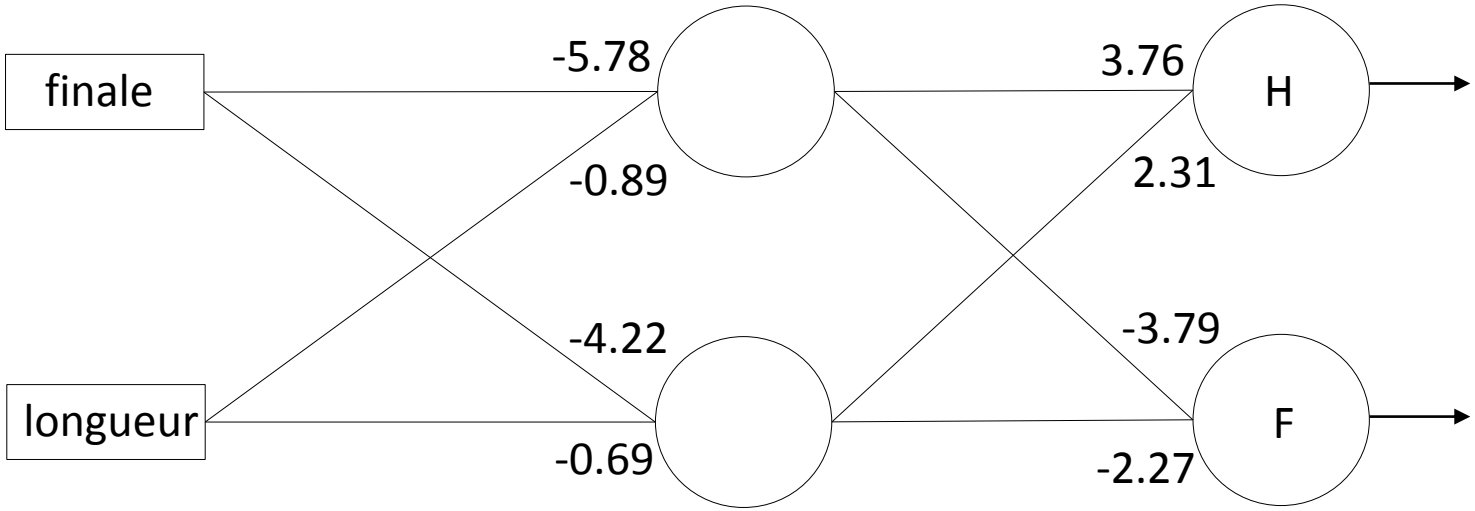
- Ajustement des poids

Apprentissage (2)



- Processus répété avec tous les items ($i_1 \dots i_n$) avec les nouveaux poids
- Stabilisation

Apprentissage (2)



3/ Les réseaux de neurones pour la sémantique distributionnelle

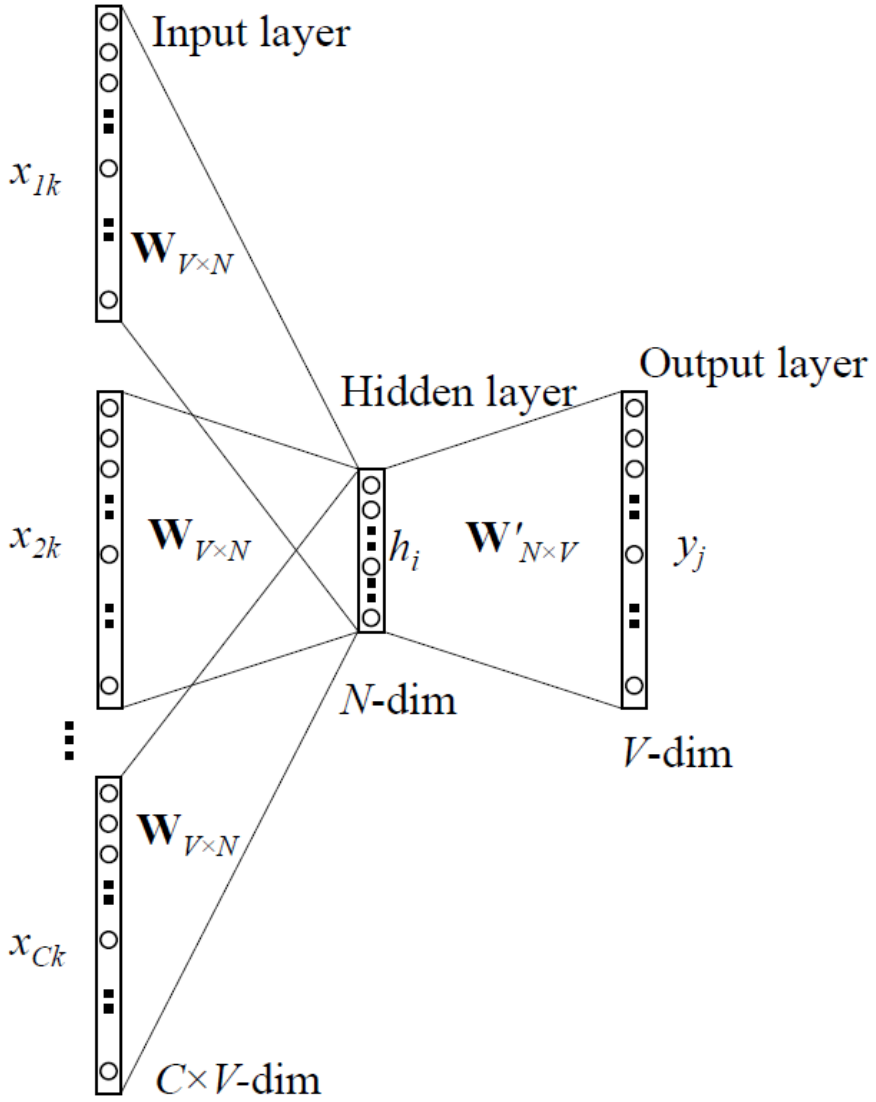
Principes

- Comment transformer l'analyse distributionnelle en un problème de classification supervisée ?
- Solution :
  - En entrée : une description du contexte d'un mot
  - En sortie : le mot qui apparaît dans ce contexte
    - (ou l'inverse, en fait c'est pareil)
  - Le système va "apprendre" que certains mots sont prédits par les mêmes contextes
- Résultat
  - C'est l'état du modèle prédictif qui est intéressant, pas sa capacité à prédire
  - Comme c'est un réseau de neurones, on récupère à la fin les pondérations de chaque mot avec la couche cachée

Word2vec

- Modèle le plus connu (Mikolov et al. 2013)
  - Pas le plus récent (cf. Bengio et al. 2003)
  - Mais un code disponible et très efficace
- Version de base : uniquement sur les cooccurrences graphiques
  - lemmatisées/catégorisées ou non
- Deux approches :
  - CBOW (continuous bag of words)
  - Skipgram

Word2vec : l'approche CBOW



Déroulement

- Représentations des mots
  - Couches d'entrée et de sortie
  - Principe "1 hot" :
    - Soit V la taille du vocabulaire retenu
    - chaque mot est représenté par un vecteur de dimension V
    - Ce vecteur contient des 0 partout, sauf 1 pour la dimension correspondant à ce mot
  - Couche cachée de taille N arbitraire (e.g. 200, 500, 1000…)
- Balayage du corpus : extraction de situations
  - X mots de contexte -> mot-cible
  - Apprentissage des poids liant la couche de sortie à la couche cachée
- Au final, une matrice de taille VxN
  - Chaque mot de V est représenté par un vecteur de taille N
  - Les mots similaires sont activés par les mêmes neurones de la couche cachée et ont donc des vecteurs similaires

Un exemple en couleur

- (Bénédicte : démo en couleur avec Wevi)
- https://ronxin.github.io/wevi/

Autres versions

- Traitement des contextes syntaxiques avec Word2vec
  - (Levy & Coldberg 2014)
- Même principe, avec des contextes de la forme REL+MOT
  - E.g contextes de « manger » = « chat_sujet », « souris_objet », etc.
  - Les mots et les contextes sont représentés dans le même espace vectoriel (couche cachée)

Les word embeddings

- On appelle *word embeddings* les représentations vectorielles des mots ainsi constituées
  - Matrice qui associe chacun des V mots aux N neurones de la couche cachée
- Intérêts :
  - Représentation compacte (N << V)
  - Par des vecteurs denses (pas de zéro)
  - "Renfermant" le comportement distributionnel des mots

1-hot vs embedding

	Chat	Chien	Phacochère	Espoir	Amour	Choucroute
Chat	1	0	0	0	0	0
Chien	0	1	0	0	0	0
Choucroute	0	0	0	0	0	1

	D1	D2	D3	D4	D5	D6
Chat	0,3	0,87	0,01	0,2	0,1	0,001
Chien	0,29	0,96	0,05	0,25	0,13	0,02
Choucroute	0,01	0,2	0,8	0,98	0,2	0,004

Utilisations

- Usage direct pour la sémantique distributionnelle
  - Similarité (type cosinus) entre les *embeddings*
  - Compositionnalité des sens par addition des vecteurs
- Représentation vectorielle du lexique : word embeddings vs 1-hot
  - Continu et pas discret
  - Comparable
  - De dimension réduite
- Apparemment avantageux pour toute tâche de TAL par apprentissage qui nécessite une représentation du lexique
  - traduction, tagging, parsing, reconnaissance d'entités nommées, traitement de la parole, RI, EI, etc.

4/ Le *Deep Learning*

Du neuf avec du vieux

- L'apprentissage profond est avant tout l'utilisation de réseaux de neurones avec de *nombreuses couches cachées*
- L'idée était présente dès le début (années 1960), mais
  - la puissance de calcul ne suivait pas
  - certains problèmes théoriques persistaient concernant la rétro-propagation à travers ces couches multiples

Actuellement

- On peut construire des réseaux énormes
  - E.g. GoogleNet, pour la classification d'images, possède 27 couches (dont la taille varie de quelques dizaines de neurones à plus de 1000)
- Qui tournent généralement sur des machines parallèles
  - E.g. AlphaGo (qui a battu les champions au jeu de go) utilise 1202 CPUs et 176 GPUs

Pourquoi toutes ces couches ?

- Le principe est que chaque couche intermédiaire
  - Représente des régularités observées dans les couches inférieures
  - Améliore le traitement des couches supérieures
- Chaque couche cachée permet donc d'*abstraire* des informations
  - Comme le font les traits descriptifs définis pour représenter les entrées dans un système classique

Pas juste un problème de puissance

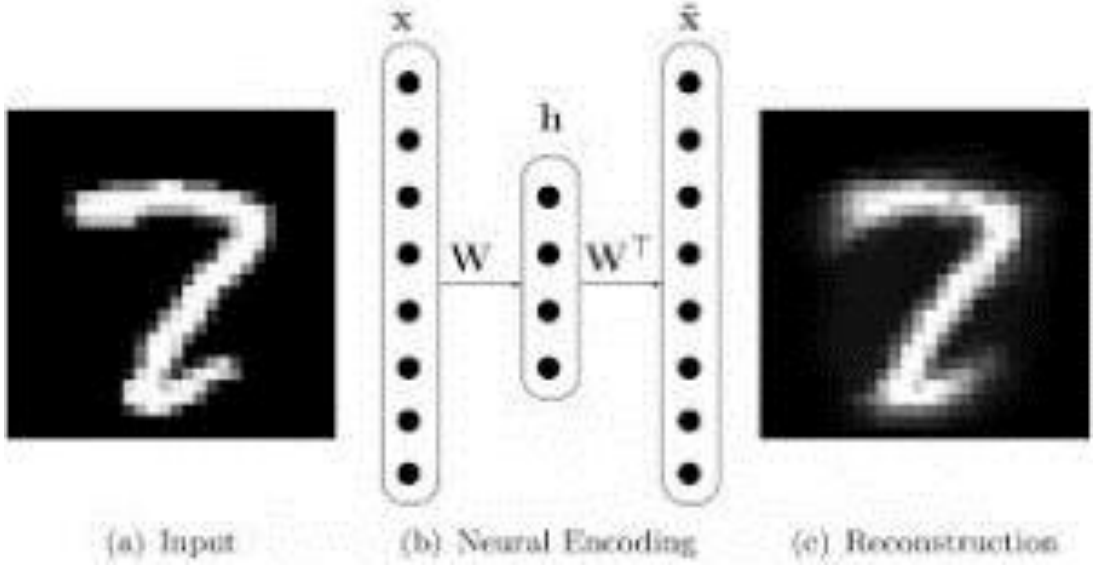
- Il existe des obstacles théoriques à l'entraînement de tels réseaux
  - En gros, que la communication passe mal des couches de sorties vers l'entrée
- Pour pallier cela, quelques nouvelles techniques
  - La géométrie des réseaux
  - La façon dont ils sont entraînés

Apprentissage non supervisé

- Les couches inférieures (côté *input*) d'un réseau profond sont généralement entraînées de façon *non supervisée*
  - i.e. sans recevoir d'information sur la nature des objets traités
  - Elles ne servent qu'à représenter des régularités observées dans les données
- Plusieurs techniques, dont la plus simple est l'auto-encodage

Auto-encodeur

- Réseau de neurones entraîné avec une couche de sortie qui est une copie conforme de la couche d'entrée
  - Son seul but est d'apprendre une (ou plusieurs) couches cachées qui stabilisent la redondance entre les caractéristiques des données
  - Une fois apprises, ces couches sont insérées dans le réseau profond



Intérêt des phases non supervisées

- Les données brutes non catégorisées sont plus faciles à trouver en grand nombre
- L'auto-encodage permet de réduire les dimensions des vecteurs d'entrée
  - En identifiant la redondance entre les features
  - En construisant des représentations synthétiques
  - Sans perdre d'information puisqu'on apprend au réseau à reproduire l'entrée

Le deep learning classique

- « Deep belief network »
- Un réseau de neurones à plusieurs couches cachées et généralement d'architecture complexe
  - Dont les couches de bas niveau ont été configurées sans supervision
    - Sur un très grand nombre de données brutes
  - Dont les couches de haut niveau ont été configurées avec supervision
    - Sur un moins grand nombre de données annotées

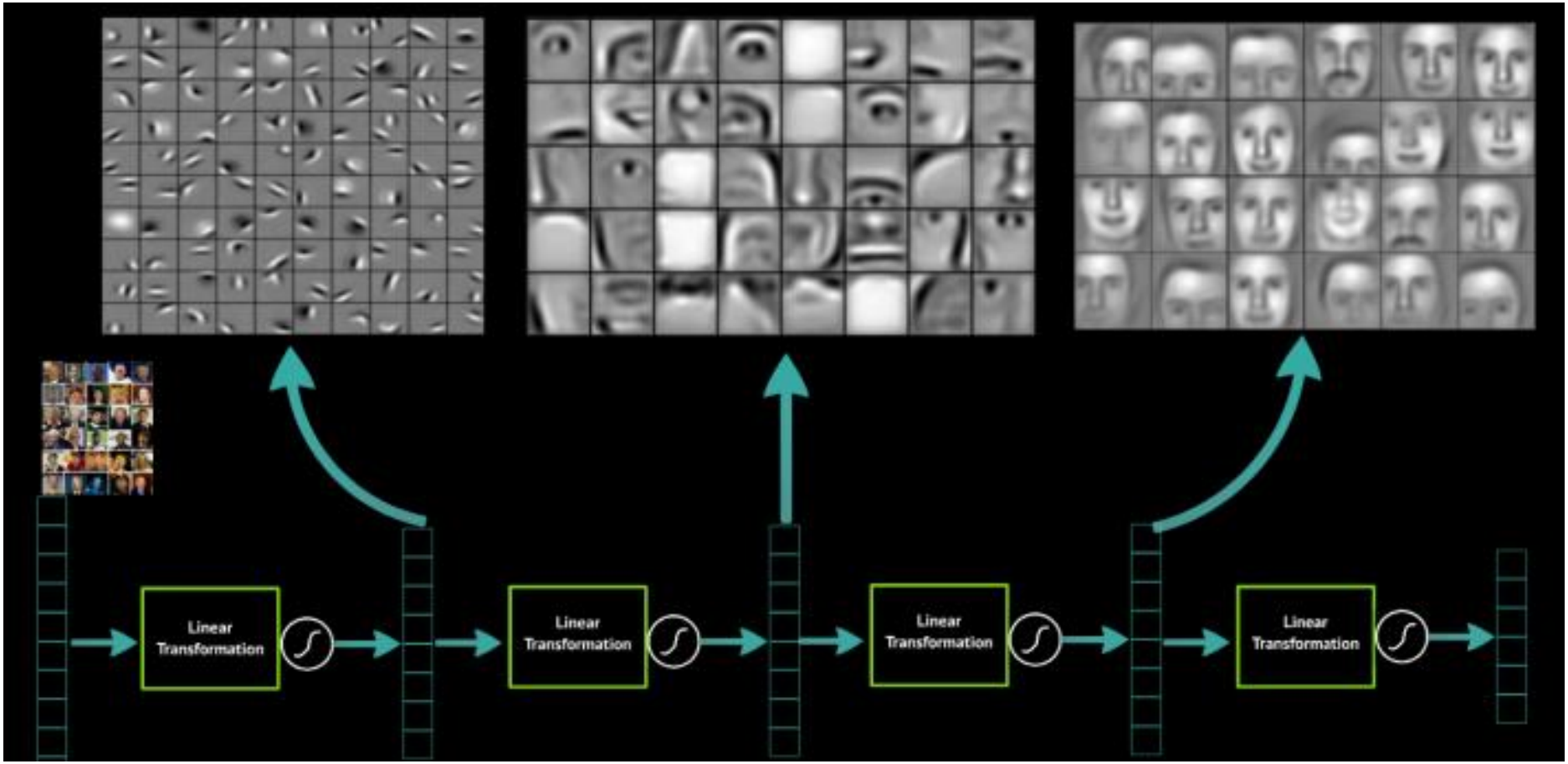
Variante : Réseaux convolutifs

- Conçus initialement pour le traitement d'image, en s'inspirant des connaissances sur le cortex visuel
- Les couches intermédiaires ne traitent qu'une partie des données d'entrée
  - E.g. juste une partie d'une image à traiter
  - Avec un recouvrement entre elles, et un partage des paramètres
- Au final, les couches du milieu se concentrent sur certaines caractéristiques, et ce sont les couches supérieures qui synthétisent leurs résultats

Exemple : Histoire de chats

- Réseau de neurone pour la classification d'images
  - Pré-apprentissage sur des millions de photos de toute nature
  - Apprentissage supervisé sur quelques cas
  - Capable de « conceptualiser » très rapidement la notion de chat
- Et même capable de représenter une photo « prototypique » de chat
  - Il suffit d'activer le réseau à l'envers (feed-backward)
  - En activant le neurone de sortie correspondant au chat
  - Et on obtient les pixels de la couche d'entrée

Le point de départ : classification d'images



Emprunté à datarobot.org

Réseaux de neurones utilisés à l'envers (backward propagation)

Hartebeest

Measuring Cup

Ant

Starfish

Anemone Fish

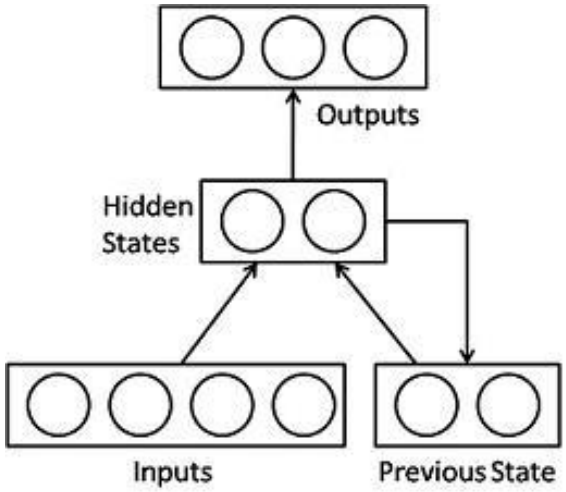
Banana

Parachute

Screw

Variante : Réseaux récurrents

- Avec des connexions qui forment des cycles dans le réseau
  - L'état interne du réseau après l'entrée n° i-1 est disponible lorsque l'entrée n°i est traitée
- Permet de gérer des modèles de langues
  - Probabilité d'avoir un mot/car après avoir trouvé les n mots/car précédents



Exemple : Génération de texte

- Modèles très utilisés pour la génération de texte
  - Corpus de départ pour entraîner le modèle ; quelques caractères en entrée puis le réseau « prédit » la suite
  - Ex : la Wikipedia caractère par caractère :

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's overeignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and et inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 5|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the onquest of India with the conflict. Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement ased n a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and infantry resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab esolution]] (PJS) http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm Official economics djoint or the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

4/ Mots de la fin

Clarifions

- Word2vec ce n'est **PAS** du deep learning
  - Le réseau n'a qu'une seule couche cachée
  - Le réseau est entraîné normalement, et ça va vite
  - Les word embeddings non plus
- **Par contre**, les word embeddings sont souvent utilisés pour faire du deep learning
  - En condensant la représentation du lexique

La philosophie du deep learning

- Objectif : faire TOUT apprendre par le réseau
  - A partir d'observations non supervisées des données
  - « Comme le font les humains » à qui on n'explique pas tout en détails
- Et donc se débarrasser de ces traits compliqués qu'on utilise pour l'apprentissage classique
  - Considérés comme une perte de temps et induisant des biais
  - E.g. préférer des n-grammes de caractères à des catégories syntaxiques ou sémantiques prédéfinies

Quelle place pour nous ?

- Plus de feature engineering
  - Qui était pourtant une porte d'entrée des connaissances linguistiques dans les systèmes statistiques
- Plus de boîte grise
  - Très compliqué de voir ce qui se passe au milieu du système
    - Même d'expliquer pourquoi deux mots sont similaires dans des embeddings
- Un coup d'entrée très élevé
  - Besoin de puissance de calcul et de très grosses masses de données pour l'apprentissage non-supervisé

Mais soyons positifs (avec Manning, 2016)

- Revisiter les problème de catégorisation avec des approches plus « continues » :

  I encourage people to not get into the rut of doing no more than using word vectors to make performance go up a couple of percent. Even more strongly, I would like to suggest that we might return instead to some of the interesting linguistic and cognitive issues that motivated noncategorical representations and neural network approaches.

Ouvrons le débat (toujours avec Manning)

- *Overall, I think we should feel excited and glad to live in a time when Natural Language Processing is seen as so central to both the further development of machine learning and industry application problems. The future is bright. However, I would encourage everyone to think about problems, architectures, cognitive science, and the details of human language, how it is learned, processed, and how it changes, rather than just chasing state-of-the-art numbers on a benchmark task.*

Références

- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. O'Reilly.
- Han, J. & Kamber M. (2006). *Data Mining : Concepts and Techniques*. Elsevier, Morgan Kaufmann Publishers.
- Levy, O., & Goldberg, Y. (2014). Dependency-Based Word Embeddings. *Proceedings of ACL*, pp. 302-308.
- Manning, C. (2016). Computational linguistics and deep learning. *Computational Linguistics*, 41(4).
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Roberts, E. https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/index.html
- Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal*, 211-229.
- Solomonoff, R. (1957). An Inductive Inference Machine. *IRE Convention Record, Section on Information Theory*, Part 2, pp. 56–62.
- Witten, I. H. & Frank E. (2005). *Data Mining : Practical Machine Learning Tools and Techniques*. Elsevier, Morgan Kaufmann Publishers.
- https://deeplearning4j.org/neuralnet-overview