Apprentissage bio-inspiré

Boi Faltings

Laboratoire d'Intelligence Artificielle
boi.faltings@epfl.ch
http://moodle.epfl.ch/

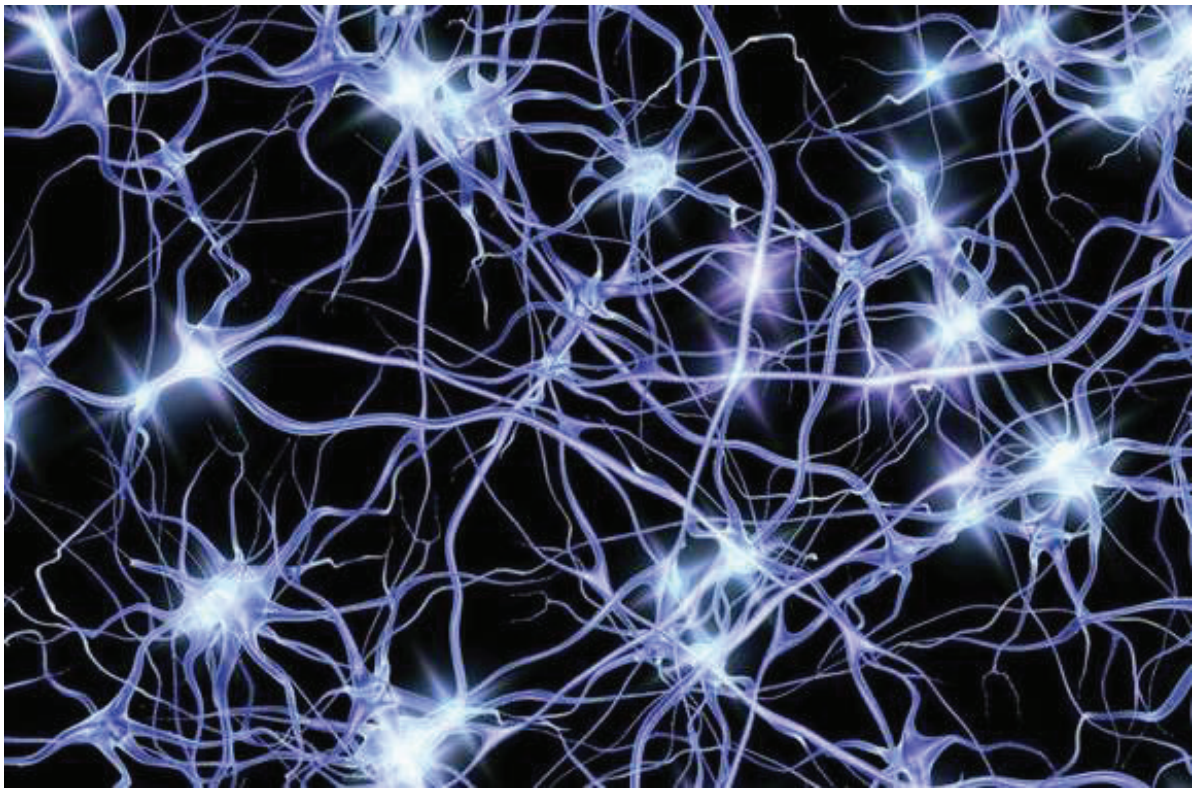
Apprentissage bio-inspiré

Certains techniques d'apprentissage sont inspirées par une analogie à la biologie:

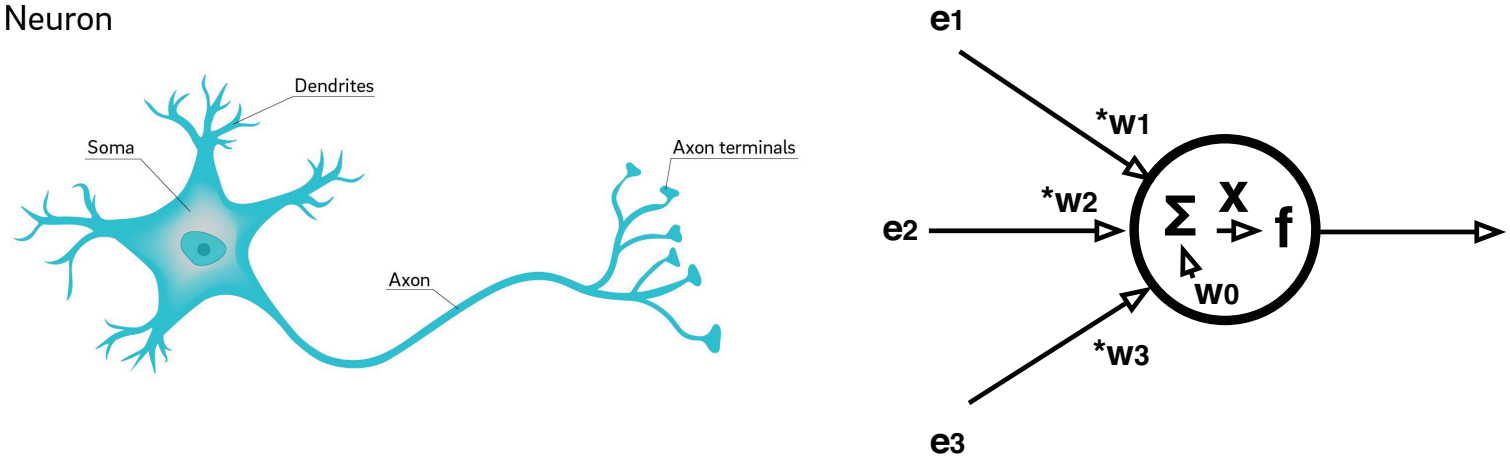
- réseaux de neurones artificiels: cerveau.
- algorithmes génétiques: évolution.

Les techniques reprennent certaines caractéristiques du modèle biologique, mais de façon très simplifiée.

Inspiration Biologique



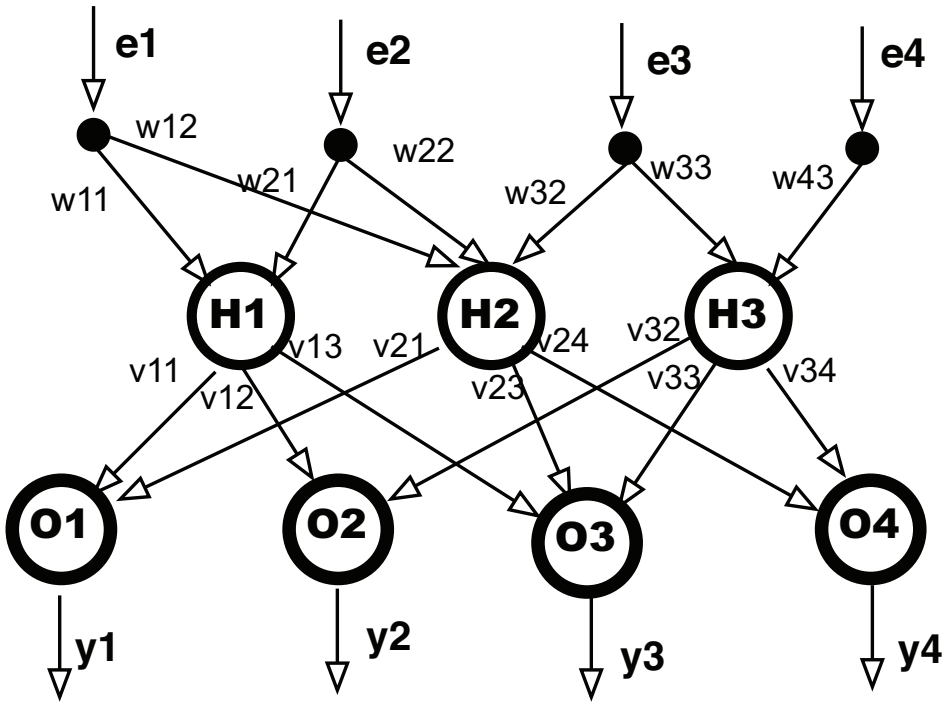
Modèle d'un Neuron



Entrées = Dendrites, Sortie = Axon ; Modèle $\simeq$ perceptron.

Réseaux de Neurones Artificiels

Le plus simple: réseau à deux couches:



H = couche cachée, O = couche de sortie

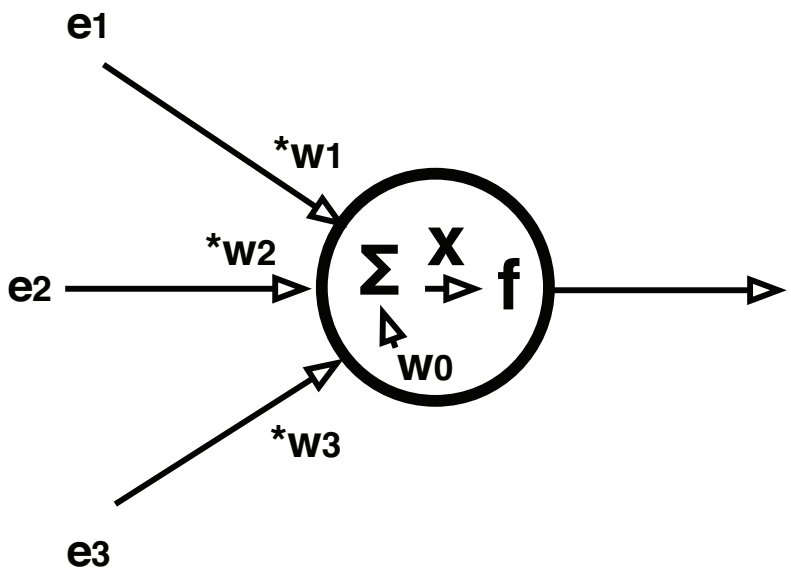
RNA à deux couches

- Structure:
  - entrées $e_1, .., e_k \rightarrow$ couche cachée $H_1, .., H_l$
  - couche cachée $\rightarrow$ couche de sortie $O_1, .., O_m$
  - couche de sortie $O_1, .., O_m \rightarrow$ sorties $y_1..y_m$
- La couche cachée permet de construire des traits complexes et d'ainsi représenter n'importe quelle fonction (si le nombre de neurones est suffisant).
- Comble les lacunes d'un seul perceptron.

Fonction d'un neurone

Chaque neurone calcule une somme des entrées e_i, ponderée par les *poids des connexions* w_i, et ajoute un *biais* w_0:

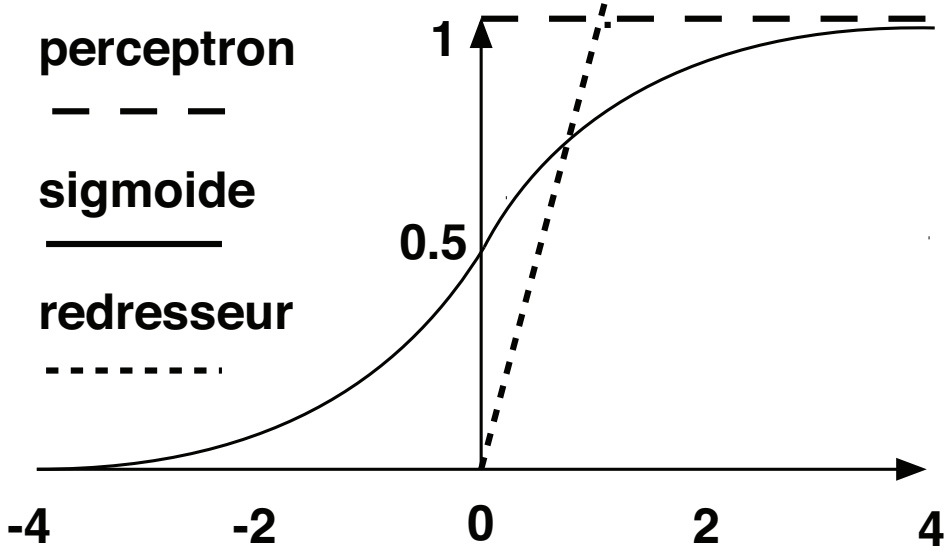
$$x = \sum_{i \in entrees} w_i e_i + w_0$$



La sortie, appellée *l'activation*, est obtenue par une *fonction d'activation* non-linéaire $f(x)$ appliquée à x.

Fonctions d'activation

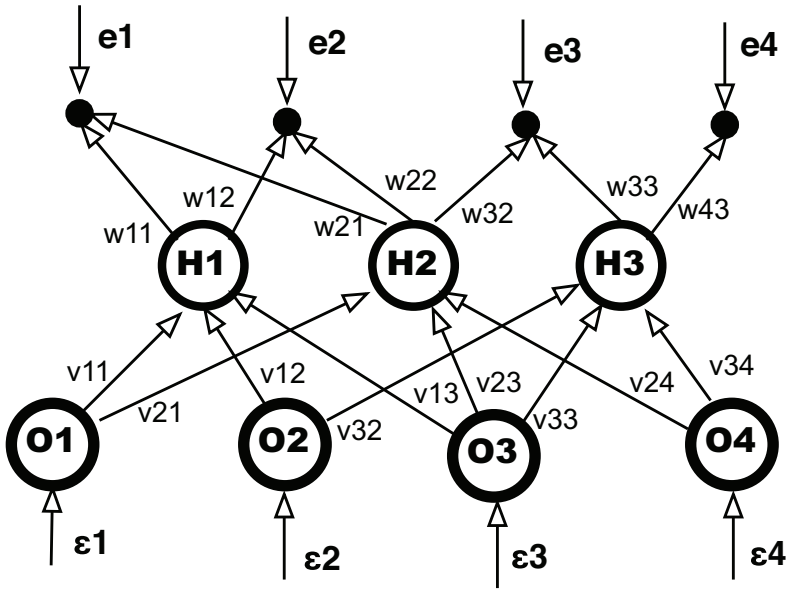
- Fonction de seuil (perceptron): $f(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x \leq 0 \end{cases}$
- Fonction sigmoïde: $f(x) = \frac{1}{1+e^{-x}}$
- ou bien une fonction redresseur: $f(x) = max(0, x)$



Entraînement d'un réseau

- Adaptation de l'algorithme du perceptron: itérativement passer des exemples choisis aléatoirement, et corriger les poids à chaque erreur.
- Rétropropagation (Backpropagation): pour chaque prévision, rétropropager l'erreur aux couches cachées.
- Optimisation par descente du gradient: à chaque couche, ajuster les poids selon le gradient pour réduire l'erreur de classification.
- *End-to-end*: tous les éléments sont entrainés avec un seul signal d'erreur à la sortie.

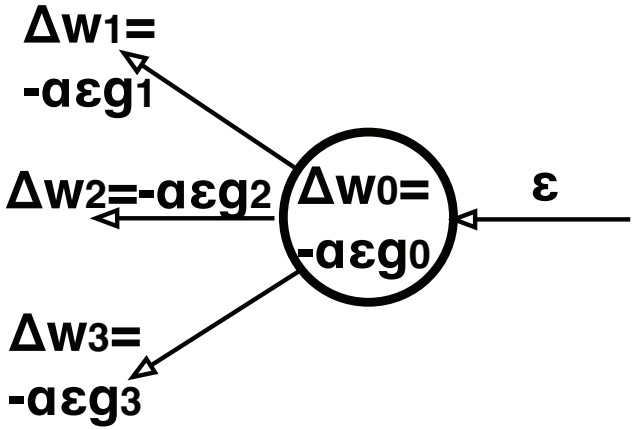
Rétropropagation dans un réseau



- Erreurs de la couche de sortie $\Rightarrow$ correction des poids v_{ij} de la couche de sortie.
- Rétropropagation des erreurs à la couche cachée et correction des poids w_{ij} de la couche cachée.
- Peut s'étendre à de nombreuses couches cachées.

Apprentissage...

Couche de sortie: erreur ϵ du neurone mesurée:

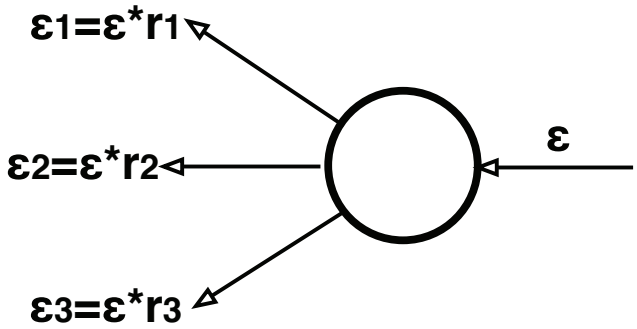


1. *correction* de w_i par gradient

$$g_i = \frac{\partial \epsilon_j}{\partial w_i} = \frac{\partial x}{\partial w_i} \cdot \frac{df}{dx} = e_i \frac{df}{dx}$$

$$\Delta w_i = -\alpha \epsilon g_i \ (\alpha \in [0..1] = \text{taux d'apprentissage})$$

... et rétropropagation



2. *rétropropagation* de l'erreur à la couche cachée par gradient

$$r_i = \frac{\partial \epsilon}{\partial e_i} = \frac{\partial x}{\partial e_i} \cdot \frac{df}{dx} = w_i \frac{df}{dx}$$

$\epsilon_i^c = \sum_j r_{ij} \epsilon_j$ (somme sur toutes les neurones j connectés)

Forme du gradient

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

$$\frac{df}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = f(x)(1-f(x))$$

- Fonction redresseur: $f(x) = max(0, x)$:

$$\frac{df}{dx} = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x \leq 0 \end{cases}$$

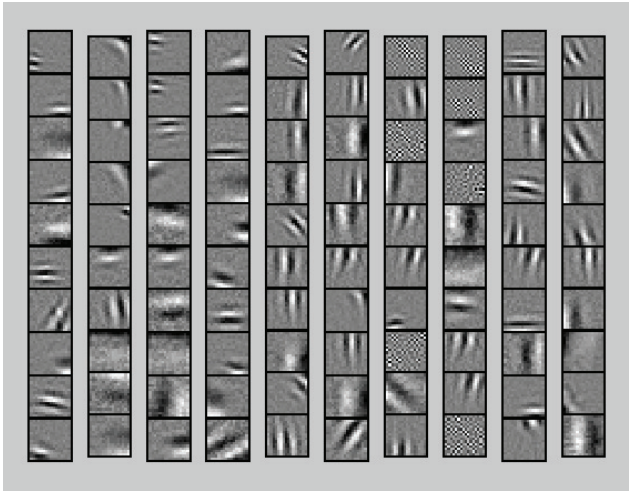
Simple à implémenter, surtout pour la fonction redresseur.

Deep Learning

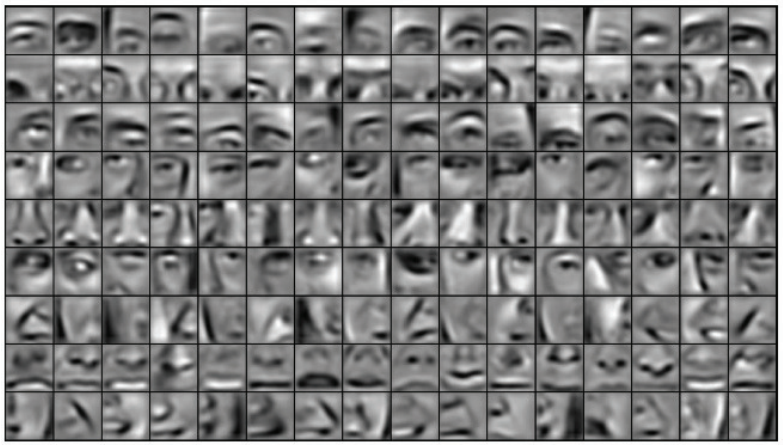
- Réseau à nombreuses couches cachées = réseau *profond*. (par exemple: 30 couches).
- *Deep learning* = apprentissage par rétropropagation.
- Souvent difficile à faire converger, exige une expérimentation avec les paramètres et la connectivité du réseau.
- Nécessite beaucoup de données.
- Cependant, on peut parfois obtenir de très bons résultats, pourvu une grande quantité de données d'entraînement.

Pourquoi beaucoup de couches?

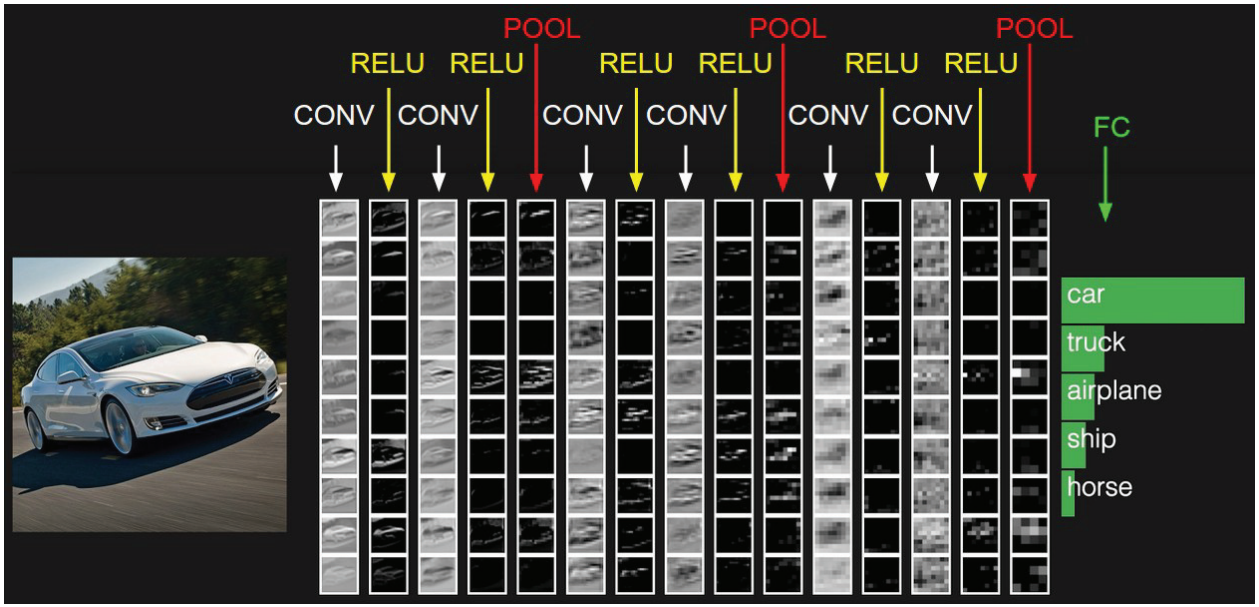
- Perceptron = discrimination de formes simplement séparables:



- Multicouche ⇒ frontières plus complexes:



Réseaux de neurones convolutionnels



- Couches CONV: application de filtres (apprises sur les données.
- Couches RELU: combinaison des filtres
- Couches POOL: mise en commun des réponses des filtres (présence/absence d'attributs).

Problème du gradient disparaissant

- Perceptron: gradient de l'erreur = vecteur de l'exemple.
- Multi-couche: gradient doit être propagé à travers la fonction d'activation.
- Fonction sigmaoide: derivée $<< 1$
  $\Rightarrow$ gradient devient de plus en plus faible.
- Fonction redresseur ($f(x) = max(0, x)$): derivée $= 1$
  $\Rightarrow$ taille du gradient reste constante.
- Permet d'entraîner des réseaux à beaucoup de couches.
- Version différentiable: $f(x) = ln(1 + e^x)$: $\frac{df}{fx} = \frac{e^x}{1+e^x}$

Minima locaux

- La descente du gradient conduit souvent à un minimum local.

$\Rightarrow$ introduire une variation aléatoire (comme dans la satisfaction de contraintes).

- Dropout: aléatoirement enlever quelques neurones du calcul du gradient.
- Autre interprétation: résultat de l'apprentissage est robuste à des variations $\Rightarrow$ plus fiable.

RNA et Logique

Ancienne pensée (Turing, Minsky, etc.)

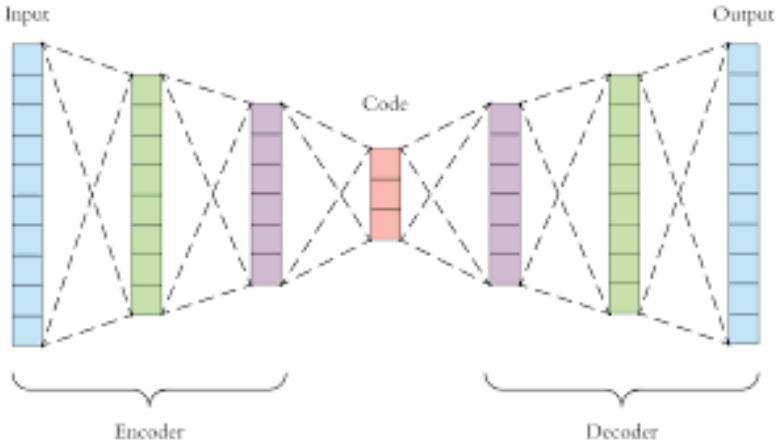
- Chaque sortie d'un neuron correspond à une proposition logique (activée = vrai).
- Un neuron implémente des règles, la somme ponderée des entrées est l'approximation d'une combinaison logique.

$\Rightarrow$ un réseau de neurones implémente un raisonnement logique.

Pensée moderne (Hinton, LeCun):

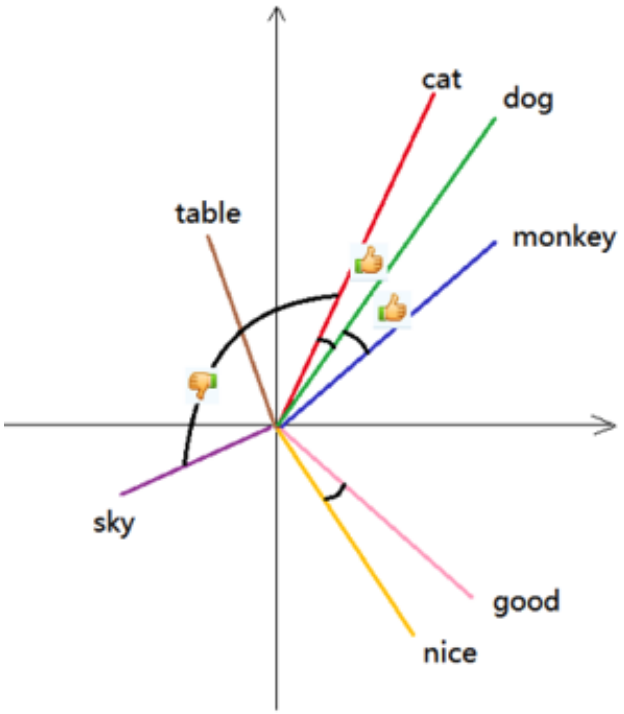
- *Embedding* = vecteur d'activation de neurones d'une couche $\simeq$ proposition logique.
- Propagation d'une couche à l'autre = inférence probabiliste sur les embeddings.
- Modèle analogue, non pas digital $\Rightarrow$ permet rétropropagation.

Autoencoders

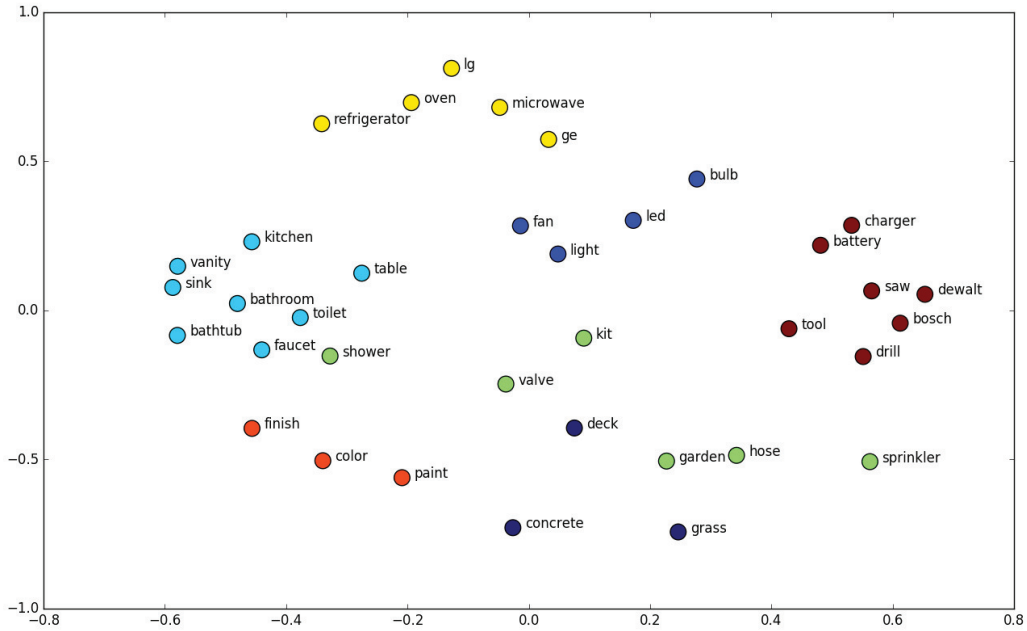


- Réseau *encodeur* = réseau profond qui représente des entrées de haute dimension dans une couche à plus faible dimension.
- Réseau *decodeur* = invers.
- Autoencoder = couplage des deux.
- Entrainé pour reproduire les données fournies à l'entrée mais en passant par une couche à faible dimension.
- Code doit permettre la reconstruction: apprend *embedding* de l'information dans un espace reduit.

Visualization d'un Embedding



Projection of the embedding vectors to 2-D



Concepts similaires $\Leftrightarrow$ embeddings similaires.

Calcul avec les embeddings

- Embedding d'un objet complexe = somme des embeddings d'objets simples: embedding d'un image = somme des embeddings des objets.
- Analogies:

$$emb(femme) - emb(reine) = emb(homme) - emb(roi)$$

- Combinaison de critères par une somme des embeddings.
- etc.: permet de "calculer" avec des concepts.

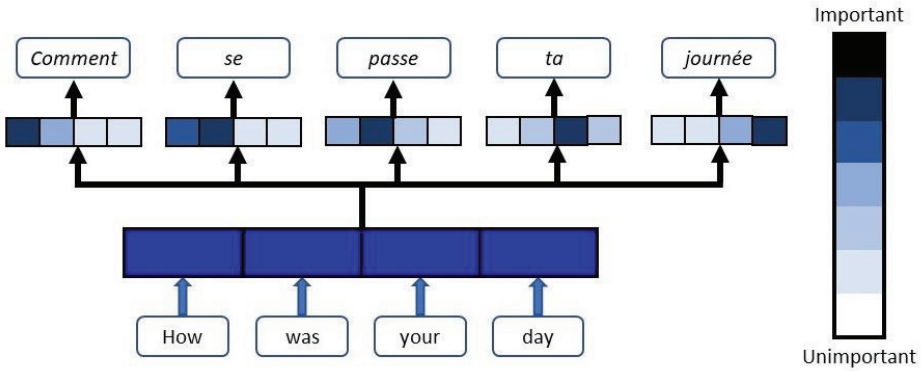
Réseaux recurrents

- Traitement de la langue: la signification des mots dépend des mots précedents.

$\Rightarrow$ construire un embedding d'une phrase incrémentalement en ajoutant les mots un par un.

- Réseau recurrent: prend comme entrées l'embedding construit jusque-là et le prochain mot, et sort l'embedding qui inclut ce prochain mot.
- L'embedding d'une phrase est obtenu après integration du dernier mot.
- Avantage: l'influence de chaque mot dépend de son context.

Mecanismes d'attention



- Les réseaux recurrents ont des difficultés à se "souvenir" des mots distants.
- Meilleur technique: garder tous les mots dans une séquence de couches, et utiliser directement des mots precedents comme entrées.
- Mécanisme plus simple: Long-Short Term Memory (LSTM).

Le Deep Learning en pratique

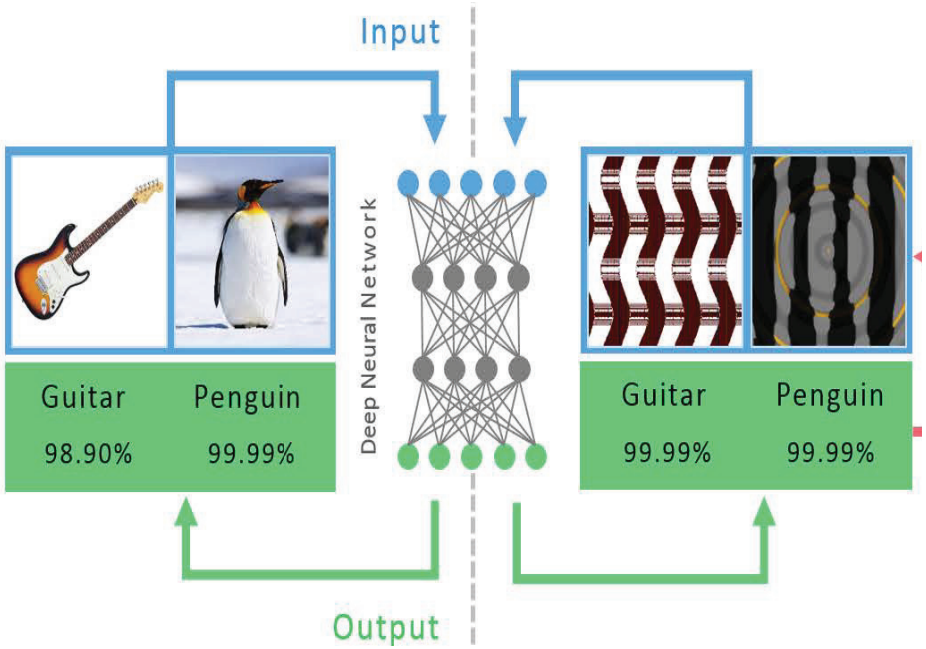
- Technique très vieille (populaire dans les années 1980)
- Mais pendant longtemps a fourni de mauvais résultats: manque de données et puissance de calcul.
- Depuis 2010, de nombreuses succès en particulier pour la vision et la compréhension de la parole.
- Intégration avec d'autres techniques IA promet des applications dans d'autres domaines.
- Exemple: AlphaGo, intégration dans un système de planification adversaire.

Imprévisibilité du Deep Learning

- Résultat de l'apprentissage est obtenu par optimisation

$\Rightarrow$ qualité ne peut pas être garantie

- Exemple: système de vision

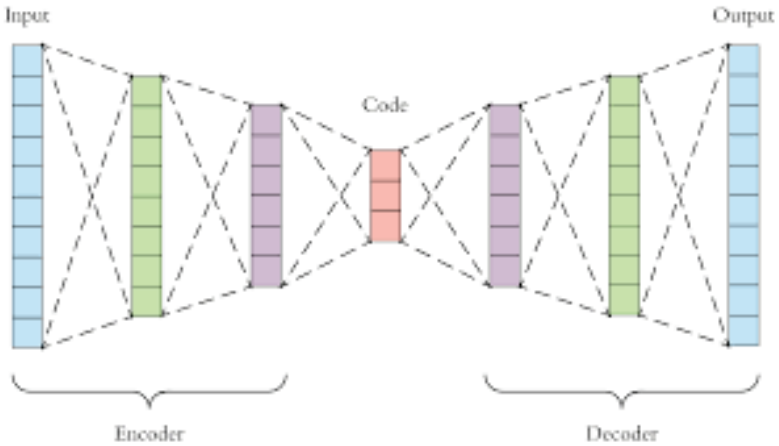


Source: A.Nguyen, J.Yosinski, J. Clune: Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, 2014

Intelligence Artificielle Générative

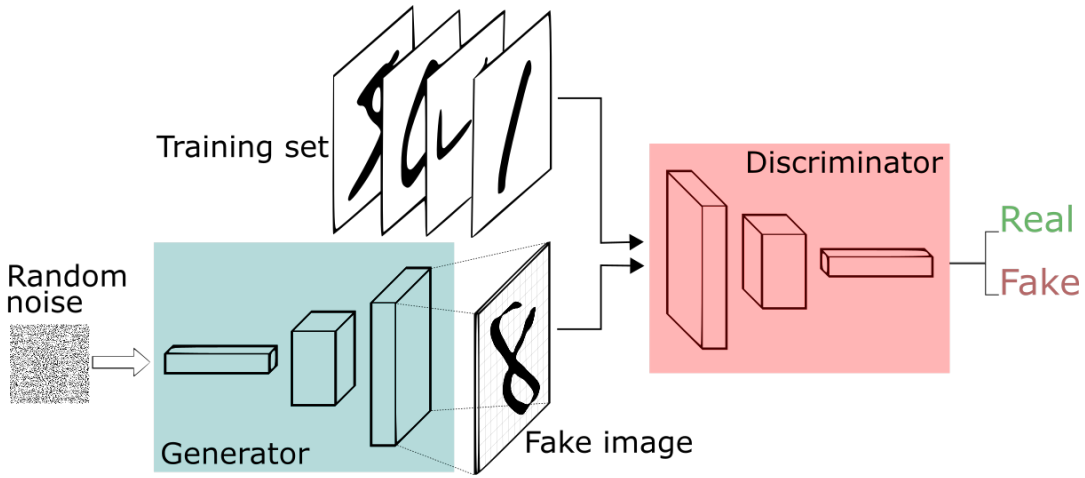
- Jusqu'à récemment, l'intelligence artificielle s'est limité à l'interpretation de textes et images.
- Grand progrès dans la capacité de calcul rend possible la génération
- Base = décodage des embeddings.

Génération d'images



- Focus: décodage d'un embedding
- Quel sera la sortie pour un code aléatoire?
- Decode(code(reine) - code(femme) + code(homme)) = ?

Generative Adversarial Networks (GAN)



- Entrainement du discriminateur et générateur en meme temps.
- Discriminateur fournit l'evaluation pour l'apprentissage du générateur.

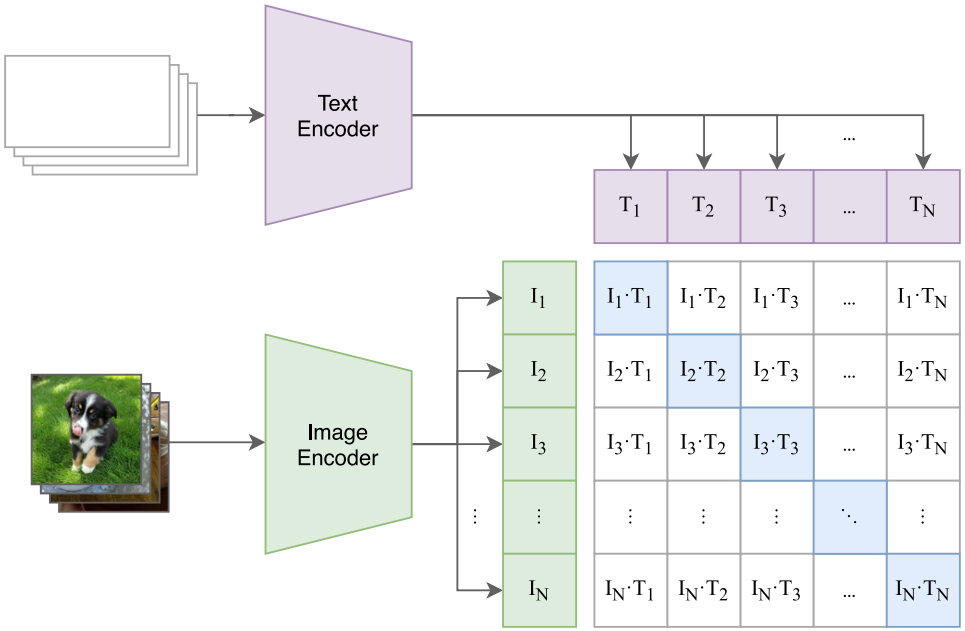
Source: https://sthalles.github.io/intro-to-gans/

Modèles de diffusion

- Un GAN a tendance au *mode collapse*: génération répétée d'images tres similaires.
- Demande un element aléatoire pour assurer la diversité.
- Diffusion: variation par un bruit aléatoire, modèle entrainé pour la transformation inverse.

Source: NVIDIA

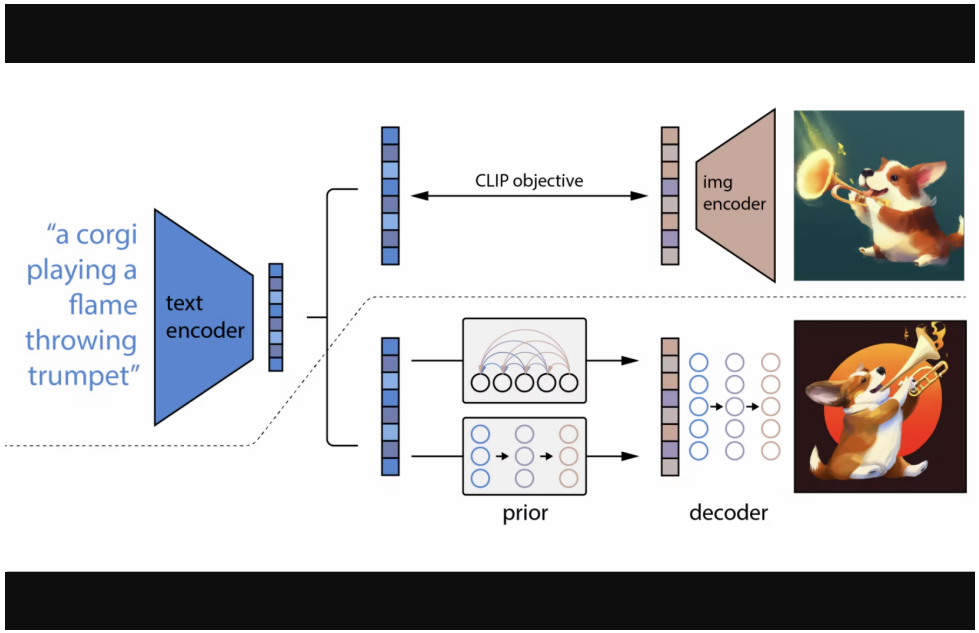
DALL-E



- Genère des images qui correspondent à une déscription.
- Entrainé sur des images et leur légendes.
- Apprend une traduction entre embeddings des phrases et d'image: CLIP

Alec Radford et al.: Learning Transferable Visual Models From Natural Language Supervision

DALL-E 2



- Text ⇒ embedding CLIP ⇒ point de départ diffusion inverse.

A. Ramesh et al.: Hierarchical Text-Conditional Image Generation with CLIP Latents

Modèles de langage

- Prédire le prochain mot sur la base d'un *context* de mots precedents.
- "Ce jour-là, le temps était ..."

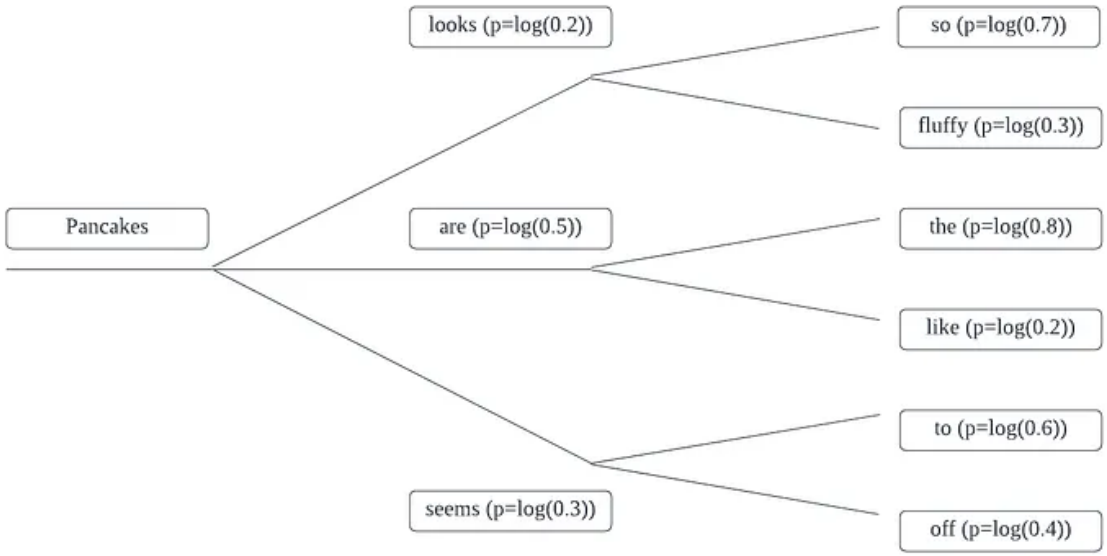
"pluvieux"	p = 0.23
"merveilleux"	p=0.15
'tempetueux"	p = 0.05
...	...

- Modèle statistique appris sur un grand volume de text.
- Auto-supervisé: le text fournit les labels pour l'apprentissage.

Grandes modèles de langage

- Représentation compacte du modèle avec des réseaux de neurones artificiels.
- Utilisamt une grande quantité de texts, dialogues, pages web, blogs, courriels, etc.
- Réseau de neurones enorme: plus de 1000 milliards de paramètres (GPT 4).

Génération



- A partir d'un context initial (prompt), générer le text mot par mot, et ajouter les mots générés à la fin du context.
- Prendre simplement le mot le plus probable donne un text incohérent.
- Recherche de la meilleure séquence par beam search (A*).

Jason LZP: Beam Search Decoding For Text Generation In Python

ChatGPT

- Modèle du langage contient également des dialogues, mais ce n'est pas suffisant.
- Pour en faire un chatbot qui répond à des questions et commandes, il faut un entrainement spécifique par des humains.

$\Rightarrow$ Modification du modèle pour donner des réponses appropriés.

Alignement

- Le modèle de langage est un amalgam désequilibré de nombreuses textes.
- Les continuations les plus probables ne sont pas toujours *alignés* avec les besoins.

$\Rightarrow$ il faut ajuster le modèle pour produire les réponses souhaitées.

- Technique utilisée: évaluation de plusieurs réponses et renforcement des réponses preferées.
- Par des utilisateurs ou par des évaluateurs contractés par internet (crowdwork).

Reinforcement learning from human feedback (RLHF)

- Apprentissage par renforcement demande qu'on attache des valeurs numeriques aux réponses.
- Cependant, pour optimiser le modèle de langage, la seule chose qui compte est de sélectionner la bonne réponse.
- La strategie optimale peut être apprise directement sans devoir passer par une optimisation.
- Algorithme beaucoup plus efficace: Direct Policy Optimization (DPO)

Complexité

- Pour que les produits de la génération soient réalistes, il faut que le modèle soit entrainé sur un énorme volume de données.
- GPT-4 est entrainé sur 1 Petabyte (1 million GB) de text, équivalent à 1 milliard de livres, une bibliothèque de 50'000 km de long.
- Le modèle a plus de 1000 milliards de paramètres: la génération d'un seul token (mot) utilise plus de 1000 milliards opérations de calcul.
- On dit que le coût d'un entrainement s'elève à 100 millions de dollars américains.

IA générative

- L'intelligence artificielle générative ouvre des nouvelles applications tels que la production de livres.
- La qualité syntaxique et sémantique des productions est optimale et souvent très convaincante, mais...
- cela va au dépens de la qualité du contenu:
  - images physiquement impossibles
  - textes avec contenus fausses.
- Problème éthique de l'utilisation de l'IA pour la désinformation.

Apprentissage par découverte

- Est-ce qu'un ordinateur peut "créer" de nouvelles connaissances sans entrées extérieures?
- Oui! Car il existe une différence entre:
  - connaissances explicites: comment peindre un joli tableau
  - connaissances implicites: comment évaluer un tableau
- Un programme peut être créatif en générant différents tableaux et retenant ceux qui sont jugés jolis!
  $\Rightarrow$ apprentissage par découverte.

Algorithmes pour l'apprentissage par découverte

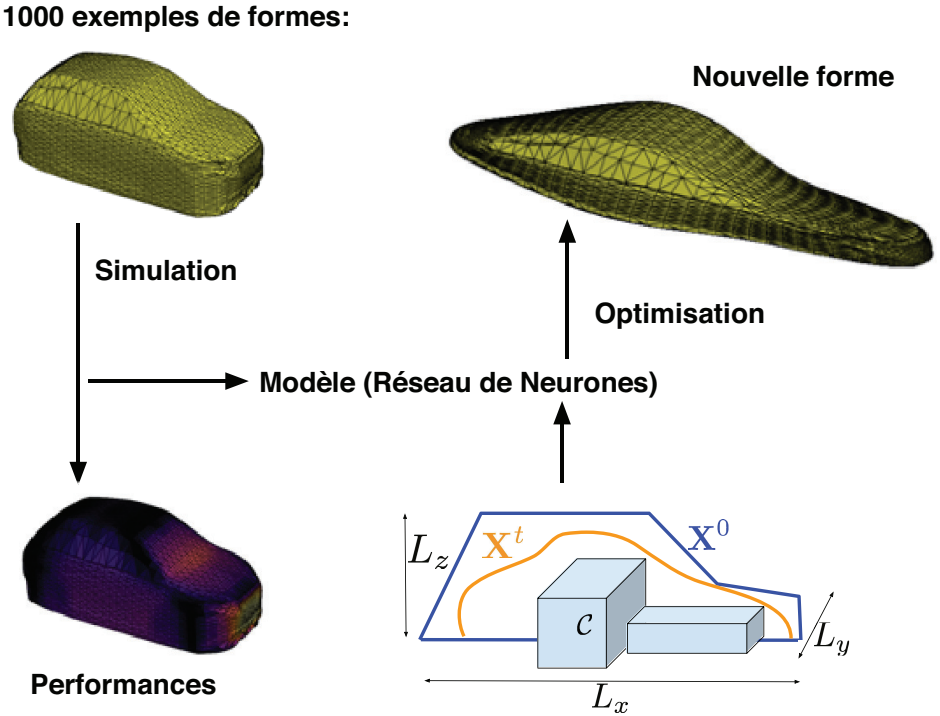
- Réseaux de neurones: utilisation du modèle comme fonction d'évaluation.
- Algorithmes génétiques: inspirés par l'évolution biologique.
- Découverte par règles: généralisation d'algorithmes génétiques.

Applications de l'apprentissage par découverte

- Optimisation: développement de nouvelles formes de turbines, d'horaires, etc.
- Apprentissage de règles de conduite, par exemple pour des voitures autonomes.
- Conception: conception de systèmes intégrés et d'autres systèmes complexes.

Inversion d'un Réseau de Neurones

Apprendre un réseau qui évalue les performances aérodynamiques d'une forme de voiture:



"Inversion" du réseau dans un processus d'optimisation.
Exemple: optimisation de forme pour le vélo le plus rapide du monde. (Neural Concept, spin-off de l'EPFL)

Algorithmes génétiques

Etant donné:

- une population initiale de n solutions potentielles (chromosomes).
- une fonction d'évaluation d'une solution.

un algorithme génétique génère de nouvelles solutions en appliquant deux opérateurs sur les solutions de la population initiale:

- la *mutation* change une seule solution de la population de manière aléatoire;
- la *combinaison* de deux solutions en crée une nouvelle.

Algorithmes génétiques (2)

La prochaine génération consiste en:

- k solutions dont la fonction d'évaluation est la plus élevée, et
- un nombre (n-k) de solutions choisies au hasard parmi les autres

L'application itérative de la procédure optimise la population selon le critère d'évaluation donné.
Avantage: le critère d'évaluation peut être arbitrairement complexe, par exemple une simulation ou un essai. D'autres techniques d'optimisations demandent des fonctions d'évaluation précises.

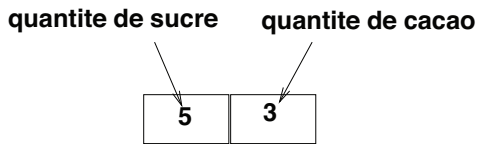
Exemple

La qualité du chocolat dépend fortement de la quantité de sucre et de cacao:

sucre									
9	1	2	3	4	5	4	3	2	1
8	2	3	4	5	6	5	4	3	2
7	3	4	5	6	7	6	5	4	3
6	4	5	6	7	8	7	6	5	4
5	5	6	7	8	9	8	7	6	5
4	4	5	6	7	8	7	6	5	4
3	3	4	5	6	7	6	5	4	3
2	2	3	4	5	6	5	4	3	2
1	1	2	3	4	5	4	3	2	1
	1	2	3	4	5	6	7	8	9

cacao

Une solution peut être représentée par un "chromosome":



Exemple (2)

Les opérations sont alors:

- mutation: la valeur de sucre ou de cacao augmente ou décroit de 1.
- combinaison z de deux chromosomes x et y: sucre(z) = sucre(x), cacao(z) = cacao(y)

Pour un problème dont la fonction d'évaluation est simple, la mutation suffit pour obtenir la meilleure solution.

Solution par mutation

0: $\{(1\ 1)\} \Rightarrow \{(1\ 2)\ (1\ 1)\}$
1: $\{(1\ 2)\ (1\ 1)\} \Rightarrow \{(1\ 3)\ (1\ 2)\ (1\ 1)\}$
2: $\{(1\ 3)\ (1\ 2)\ (1\ 1)\} \Rightarrow$
$\{(1\ 4)\ (2\ 2)\ (1\ 3)\ (2\ 1)\ (1\ 2)\ (1\ 1)\}$
3: $\{(1\ 4)\ (1\ 3)\ (1\ 2)\ (1\ 1)\} \Rightarrow$
$\{(2\ 4)\ (2\ 3)\ (3\ 1)\ (1\ 4)\ (1\ 3)$
$(1\ 2)\ (1\ 1)\}$
4: $\{(2\ 4)\ (1\ 4)\ (1\ 3)\ (2\ 1)\} \Rightarrow$
$\{(2\ 5)\ (1\ 5)\ (2\ 3)\ (2\ 2)\ \ldots\}$
5: $\{(2\ 5)\ (1\ 5)\ (2\ 3)\ (2\ 2)\} \Rightarrow$
$\{(3\ 5)\ (1\ 5)\ (3\ 2)\ (1\ 4)\ \ldots\}$
6: $\{(3\ 5)\ (1\ 5)\ (3\ 2)\ (1\ 4)\} \Rightarrow$
$\{(4\ 5)\ (3\ 5)\ (1\ 5)\ (3\ 2)\ (1\ 4)\ (3\ 1)\}$
7: $\{(4\ 5)\ (1\ 5)\ (1\ 4)\ (3\ 1)\} \Rightarrow$
$\{(5\ 5)\ (4\ 5)\ (2\ 5)\ (2\ 1)\}$

L'utilité de la combinaison

- La combinaison est nécessaire quand la fonction d'évaluation contient des minima ou maxima locaux. Par exemple:

sucre									
9	1	2	3	4	5	4	3	2	1
8	2	0	0	0	0	0	0	0	2
7	3	0	0	0	0	0	0	0	3
6	4	0	0	7	8	7	0	0	4
5	5	0	0	8	9	8	0	0	5
4	4	0	0	7	8	7	0	0	4
3	3	0	0	0	0	0	0	0	3
2	2	0	0	0	0	0	0	0	2
1	1	2	3	4	5	4	3	2	1
	1	2	3	4	5	6	7	8	9
									cacao

- $\Rightarrow$ il est difficile pour l'optimisation de traverser la "fosse" de valeurs 0, car toute mutation qui s'approche du bon résultat aura une valeur d'évaluation $= 0$ et ne sera pas retenue.
- Par contre, la combinaison permet de trouver un bon résultat:
  $(5\ 1)$ avec $(1\ 5) \Rightarrow (5\ 5)$

L'importance de la diversité

- L'opérateur de combinaison est utile uniquement si la population contient une bonne diversité.
- On peut améliorer les performances en tenant compte de la diversité des individus dans la mesure d'évaluation. Par exemple, on peut caractériser le degré d'individualisme par:

$$g(s) = \frac{1}{\sum_i \frac{1}{d^2(s,s_i)}}$$

et utiliser comme fonction d'évaluation la somme f(s)+g(s), où f(s) est la fonction d'évaluation.

Autres méthodes

Réseaux neuronaux:

- Backpropagation: semblable aux algorithmes génétiques, mais structure plus rigide.
- Reseaux de Kohonen: analyse statistique avec résultat analogue au conceptual clustering.

Utilisation de principes du domaine au lieu d'heuristiques.

Application: AlphaGo

- Réseaux de neurones ont besoin de beaucoup de données.
- Programme de jeux: génération illimitée de données en jouant programme contre programme.
- AlphaGo joue au go en cherchant des coups par algorithme de recherche, mais la recherche est dirigée par deep learning.

$\Rightarrow$ entraînement en jouant programme contre programme.

$\Rightarrow$ a battu le meilleur joueur humain, Lee Sedol, en Mars 2016.

Résumé

- Réseaux de neurones artificiels: inspirés du cerveau humain, apprentissage supervisé de classifications par un réseau à couches multiples.
- Logique digitale remplacée par algèbre continue sur des vecteurs.
- Intelligence artificielle générative: création d'images ou de textes sur la base d'un prompt.
- Alignement aux objectifs de l'utilisateur.
- Algorithmes génétiques: inspirés de l'evolution, apprend des solutions par expérimentation en utilisant la mutation et combinaison de concepts.
- Apprentissage par découverte: généralisation qui applique des heuristiques pour la génération de l'évaluation de concepts.