# Repousser les limites

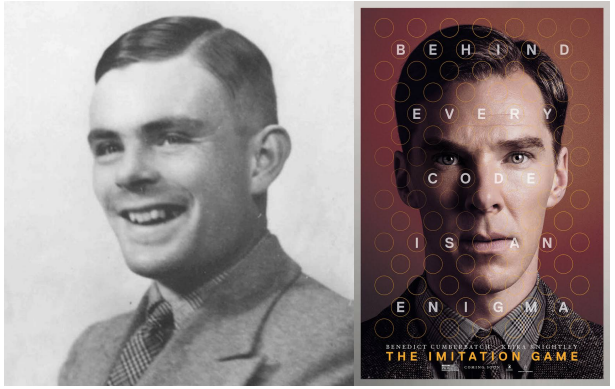
## Des problèmes impossibles à l'intelligence artificielle

Denis Kuperberg

ENS Lyon, CNRS

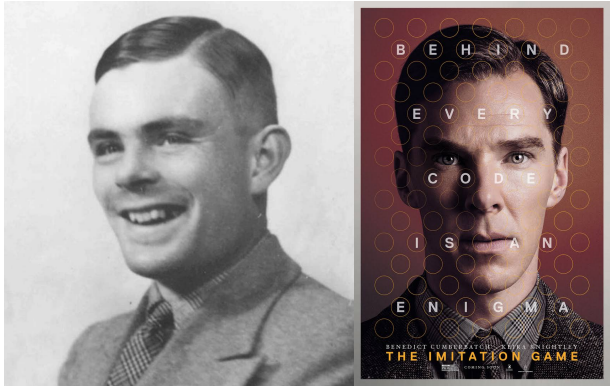
SummerIA
21 août 2017

Début de l'informatique: Turing (1936)

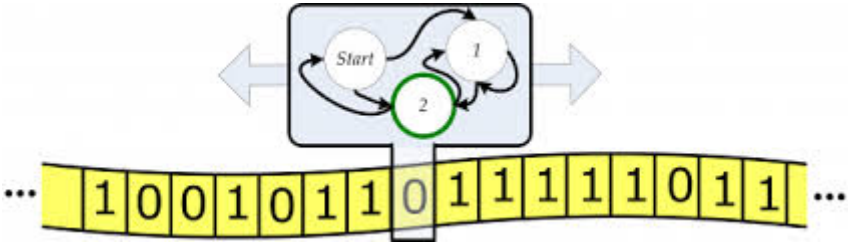


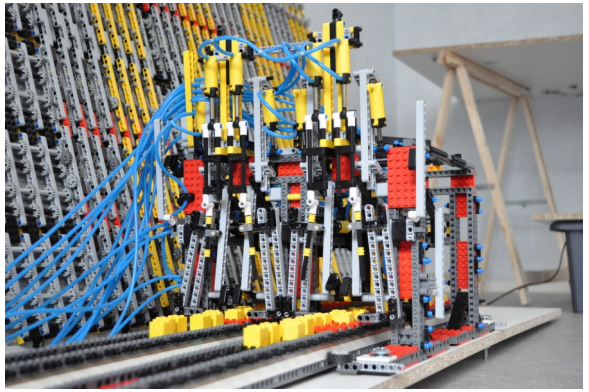

Notion mathématique d'algorithme: Machine de Turing
Il existe une machine universelle, peut simuler n'importe quelle autre machine $\rightarrow$ ordinateur.

Début de l'informatique: Turing (1936)




Notion mathématique d'algorithme: Machine de Turing
Il existe une machine universelle, peut simuler n'importe quelle autre machine → ordinateur.



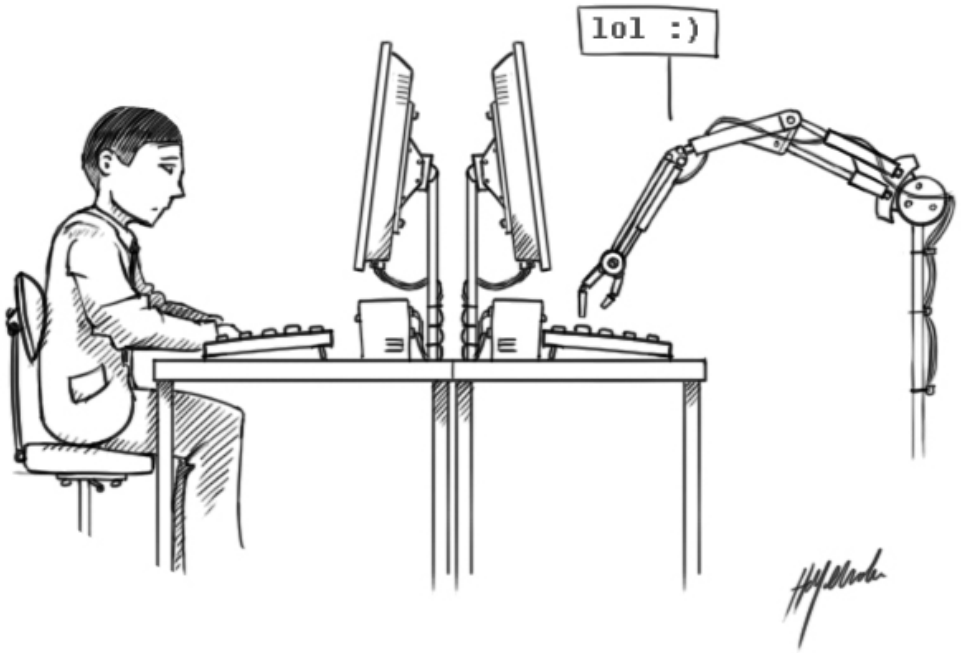
# L'intelligence artificielle

Turing entrevoit les conséquences de son invention, et rêve de l'Intelligence Artificielle (film).

# L'intelligence artificielle

Turing entrevoit les conséquences de son invention, et rêve de l'Intelligence Artificielle (film).

Test de Turing: IA réussit si elle peut se faire passer pour un humain dans une conversation.



# Vers l'intelligence artificielle

Pour créer l'Intelligence Artificielle, 2 vues s'opposent:

- **Approche symbolique**: Machine de Turing manipule des symboles
- **Approche neuronale**: Frank Rosenblatt : imiter le cerveau

Les deux approches ont finalement contribué grandement à l'IA.

# Vers l'intelligence artificielle

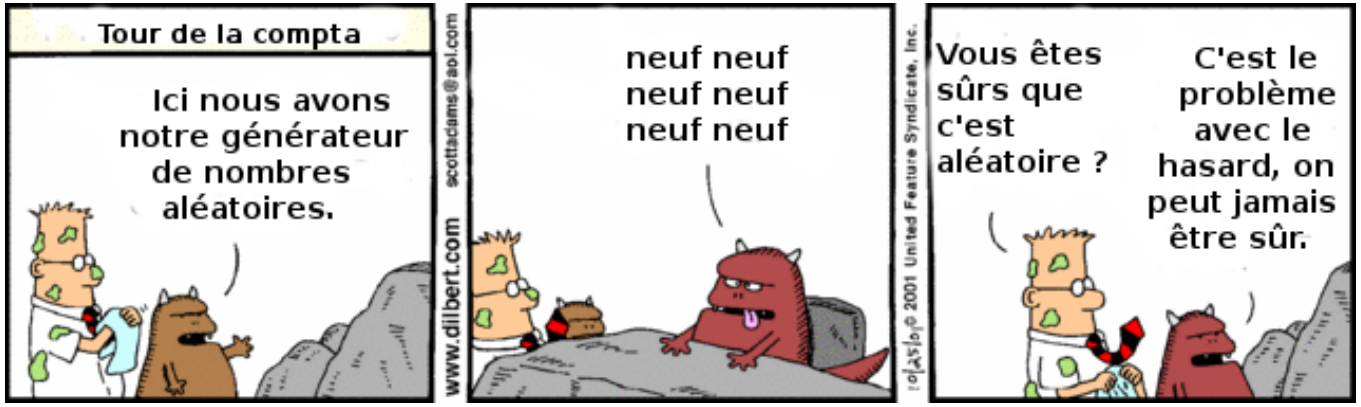
Pour créer l'Intelligence Artificielle, 2 vues s'opposent:

- **Approche symbolique**: Machine de Turing manipule des symboles
- **Approche neuronale**: Frank Rosenblatt : imiter le cerveau

Les deux approches ont finalement contribué grandement à l'IA.

Avant de voir l'intelligence artificielle, posons une question plus "simple"...

# Qu'est-ce que le hasard ?




**Problème:** Comment définir le hasard ?

Vous créez un site de poker

Il faut tirer les cartes au hasard.

Vous créez un site de poker

Il faut tirer les cartes au hasard.
Vous achetez un générateur de nombres aléatoires ($\approx$ 1000€):

Vous créez un site de poker

Il faut tirer les cartes au hasard.
Vous achetez un générateur de nombres aléatoires ($\approx$ 1000€):

- 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3

Vous créez un site de poker

Il faut tirer les cartes au hasard.
Vous achetez un générateur de nombres aléatoires ($\approx$ 1000€):

- 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3
- 2 3 5 7 1 1 1 3 1 7 1 9 2 3 2

Vous créez un site de poker

Il faut tirer les cartes au hasard.
Vous achetez un générateur de nombres aléatoires ($\approx$ 1000€):

- 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3
- 2 3 5 7 1 1 1 3 1 7 1 9 2 3 2
- 3 9 0 2 1 9 5 3 8 4 1 2 0 8 3

# Liste aléatoire ?

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

# Liste aléatoire ?

$$\pi = 3,141592\,6\,5\,3\,5\,8\,9\,7\,9\,3\,2\,3\,8\,4\,6\,2\,6\,4\,3\,3\,8\,3\,2\,7\ldots$$

# Liste aléatoire ?

$$\pi = 3,141592\ 6\,5\,3\,5\,8\,9\,7\,9\,3\,2\,3\,8\,4\,6\,2\,6\,4\,3\,3\,8\,3\,2\,7\ 9502\ldots$$

Semble aléatoire,
mais un joueur qui reconnait $\pi$ peut tricher.

# Liste aléatoire ?

$$\pi = 3,141592\ 6\,5\,3\,5\,8\,9\,7\,9\,3\,2\,3\,8\,4\,6\,2\,6\,4\,3\,3\,8\,3\,2\,7\ 9502\ldots$$

Semble aléatoire,
mais un joueur qui reconnait $\pi$ peut tricher.

**Idée:** Une liste n'est pas aléatoire si on peut la décrire simplement.

# Liste aléatoire ?

$$\pi = 3,141592\ 6\,5\,3\,5\,8\,9\,7\,9\,3\,2\,3\,8\,4\,6\,2\,6\,4\,3\,3\,8\,3\,2\,7\ 9502\ldots$$

Semble aléatoire,
mais un joueur qui reconnait $\pi$ peut tricher.

**Idée:** Une liste n'est pas aléatoire si on peut la décrire simplement.
C'est-à-dire s'il existe un algorithme court pour la générer.

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.
**Exemple**:

- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80.

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.
**Exemple**:

- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80.

  Algorithme :"Ecrire les nombres pairs de 0 à 80"

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.
**Exemple**:

- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80.

  Algorithme :"Ecrire les nombres pairs de 0 à 80"

```
for i in range(41):
    print(2*i),
```

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.
**Exemple**:

- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80.

  Algorithme :"Ecrire les nombres pairs de 0 à 80"

  ```
  for i in range(41):
      print(2*i),
  ```

- 3 9 0 2 1 9 5 3 8 4 1 2 0 8 3

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.
**Exemple**:

- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80.

  Algorithme :"Ecrire les nombres pairs de 0 à 80"

  ```
  for i in range(41):
      print(2*i),
  ```

- 3 9 0 2 1 9 5 3 8 4 1 2 0 8 3
  Algorithme : "Print("3 9 0 2 1 9 5 3 8 4 1 2 0 8 3")

# Complexité de Kolmogorov

Si $s$ est une suite de nombres,
sa **complexité de Kolmogorov** $K(s)$ est la taille du plus petit algorithme qui la génère. I.e. sa meilleure compression possible.

**Exemple**:

- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80.

  Algorithme :"Ecrire les nombres pairs de 0 à 80"

  ```
  for i in range(41):
      print(2*i),
  ```

- 3 9 0 2 1 9 5 3 8 4 1 2 0 8 3

  Algorithme : "Print("3 9 0 2 1 9 5 3 8 4 1 2 0 8 3")

**Liste aléatoire**: $K(s) \approx$ taille de $s$,
i.e. pas de compression possible de $s$.
**Indépendant** du moyen de production de $s$.

# Le problème de l'induction

Problème central en Intelligence Artificielle: quel est le suivant ?

2 4 6 8 10 ?

# Le problème de l'induction

Problème central en Intelligence Artificielle: quel est le suivant ?

2 4 6 8 10 ?

Y a-t-il une "bonne" réponse ? Comment la définir ?

# Le problème de l'induction

Problème central en Intelligence Artificielle: quel est le suivant ?

2 4 6 8 10 ?

Y a-t-il une "bonne" réponse ? Comment la définir ?

Algorithme le plus simple: "Ecrire les nombres pairs"

# Le problème de l'induction

Problème central en Intelligence Artificielle: quel est le suivant ?

2 4 6 8 10 ?

Y a-t-il une "bonne" réponse ? Comment la définir ?

Algorithme le plus simple: "Ecrire les nombres pairs"
$\rightarrow$ Meilleure réponse: 12.

Pour faire une IA, il suffit de trouver le plus petit programme qui génère la liste déjà passée.
I.e. il suffit de pouvoir calculer la complexité de Kolmogorov.

# Indécidabilité

Dommage, c'est Impossible.

# Indécidabilité

Dommage, c'est Impossible.

**Pourquoi ?** Si on pouvait calculer la fonction $K()$, on pourrait résoudre le problème de l'arrêt: savoir si un programme va s'arrêter. Et on sait que ce dernier problème est impossible.

# Indécidabilité

Dommage, c'est Impossible.

**Pourquoi ?** Si on pouvait calculer la fonction $K()$, on pourrait résoudre le problème de l'arrêt: savoir si un programme va s'arrêter. Et on sait que ce dernier problème est impossible.

On peut même aller plus loin:

**Théorème**

*Pour toute liste s, on ne peut pas être sûr que* $K(s) > 500Mo$.
*Il est impossible de prouver qu'un ensemble de données ne peut pas être compressé dans un zip de moins de* $500Mo$

**Exemple**: On peut peut-être compresser tous les disques durs de la Terre en un zip de 500Mo.

# Démonstration

**Théorème**

*Pour toute suite s, on ne peut pas prouver que* $K(s) > 500Mo$.

# Démonstration

### Théorème

*Pour toute suite s, on ne peut pas prouver que* $K(s) > 500Mo$.

**Démonstration par l'absurde**: Supposons qu'il existe une suite $s$ qui admet une preuve $D$ que $K(s) > 500\text{Mo}$.

# Démonstration

**Théorème**

*Pour toute suite s, on ne peut pas prouver que* $K(s) > 500Mo$.

**Démonstration par l'absurde**: Supposons qu'il existe une suite $s$ qui admet une preuve $D$ que $K(s) > 500\text{Mo}$.

On construit le programme suivant (par exemple en Python):

```
pour chaque couple (x,Y) de (suite,preuve):
    si Y est une preuve que K(x) > 500Mo:
            print(x)
            quit()
```

# Démonstration

### Théorème

*Pour toute suite s, on ne peut pas prouver que* $K(s) > 500Mo$.

**Démonstration par l'absurde**: Supposons qu'il existe une suite $s$ qui admet une preuve $D$ que $K(s) > 500\text{Mo}$.

On construit le programme suivant (par exemple en Python):

```
pour chaque couple (x,Y) de (suite,preuve):
    si Y est une preuve que K(x) > 500Mo:
            print(x)
            quit()
```

Puisque $(s, D)$ existe, le programme va trouver à un moment un $(x, Y)$ qui fonctionne et écrire $x$.

# Démonstration

### Théorème

*Pour toute suite s, on ne peut pas prouver que* $K(s) > 500Mo$.

**Démonstration par l'absurde**: Supposons qu'il existe une suite $s$ qui admet une preuve $D$ que $K(s) > 500\text{Mo}$.

On construit le programme suivant (par exemple en Python):

```
pour chaque couple (x,Y) de (suite,preuve):
    si Y est une preuve que K(x) > 500Mo:
            print(x)
            quit()
```

Puisque $(s, D)$ existe, le programme va trouver à un moment un $(x, Y)$ qui fonctionne et écrire $x$.

MAIS notre programme fait moins de 500Mo. C'est donc un témoin que $K(x) < 500\text{Mo}$. Contradiction

# Démonstration

### Théorème

*Pour toute suite s, on ne peut pas prouver que* $K(s) > 500Mo$.

**Démonstration par l'absurde**: Supposons qu'il existe une suite $s$ qui admet une preuve $D$ que $K(s) > 500$Mo.

On construit le programme suivant (par exemple en Python):

```
pour chaque couple (x,Y) de (suite,preuve):
    si Y est une preuve que K(x) > 500Mo:
            print(x)
            quit()
```

Puisque $(s, D)$ existe, le programme va trouver à un moment un $(x, Y)$ qui fonctionne et écrire $x$.

MAIS notre programme fait moins de 500Mo. C'est donc un témoin que $K(x) < 500$Mo. Contradiction

Un tel couple $(s, D)$ ne peut donc pas exister. Pour tout $s$, il ne peut pas y avoir pas de démonstration $D$ qui montrerait que $K(s) > 500$Mo.

L'intelligence artificielle "parfaite" ne peut donc pas exister.

Comment on s'en sort en pratique ?

L'intelligence artificielle "parfaite" ne peut donc pas exister.

Comment on s'en sort en pratique ?

On cherche toujours à trouver le meilleur programme court qui génère une liste.

**Compresser c'est comprendre**.

Et:

rien à compresser $\leftrightarrow$ rien à comprendre $\leftrightarrow$ hasard.

# Reconnaissance d'images

chat

pas chat

chat

pas chat

# Reconnaissance d'images

chat

pas chat

chat

pas chat

?

# Reconnaissance d'images

chat

pas chat

chat

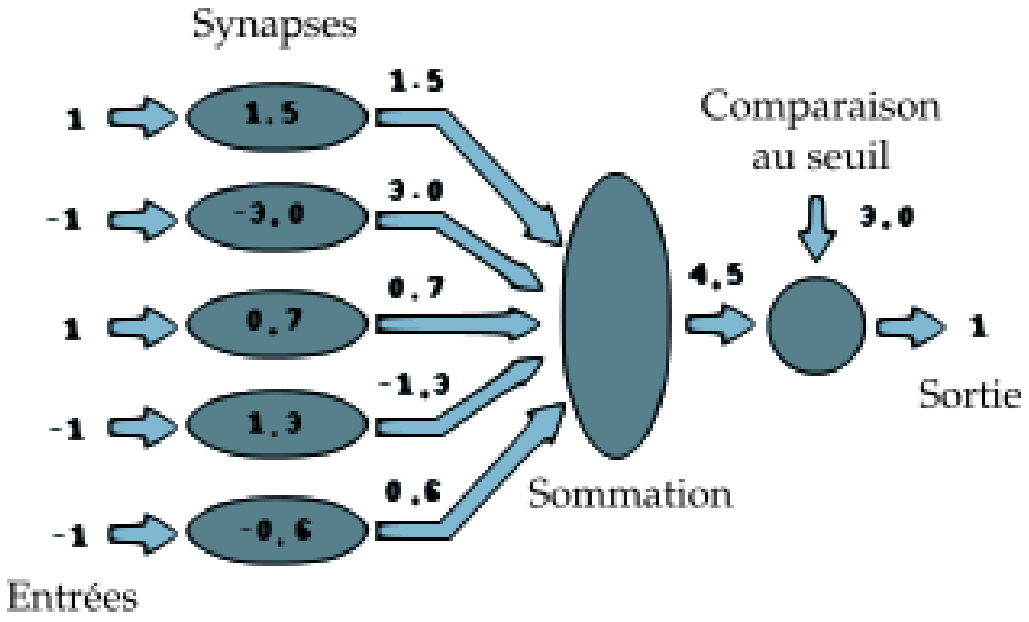
pas chat

?

**But**: Trouver un petit programme qui reconnait les chats.

**Outil**: réseaux de neurones.

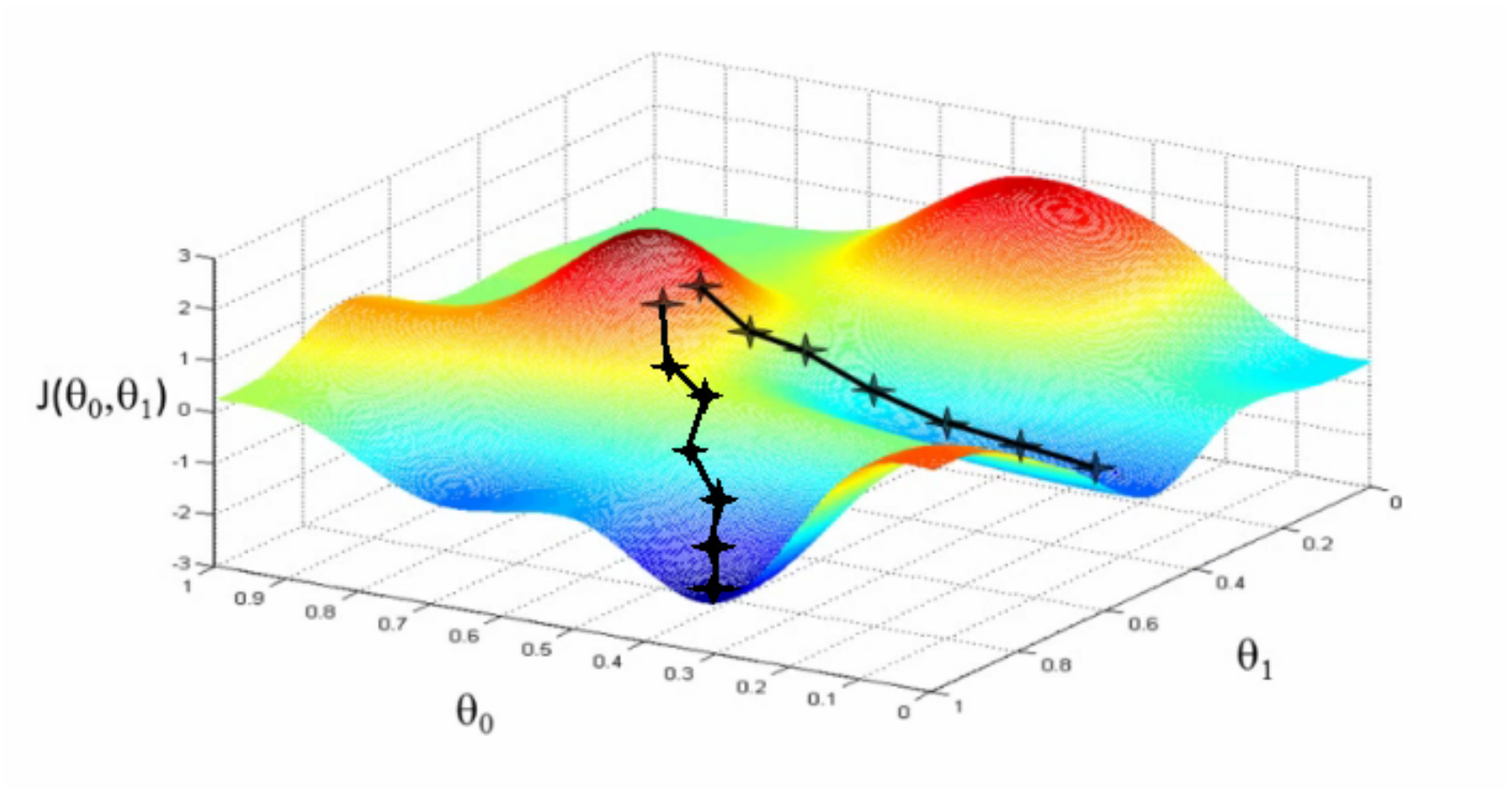
Reseaux de neurones: organisés en couches
Chaque “neurone” fait un calcul simple sur ce qu’il reçoit et envoie le résultat à des neurones de la couche suivante.



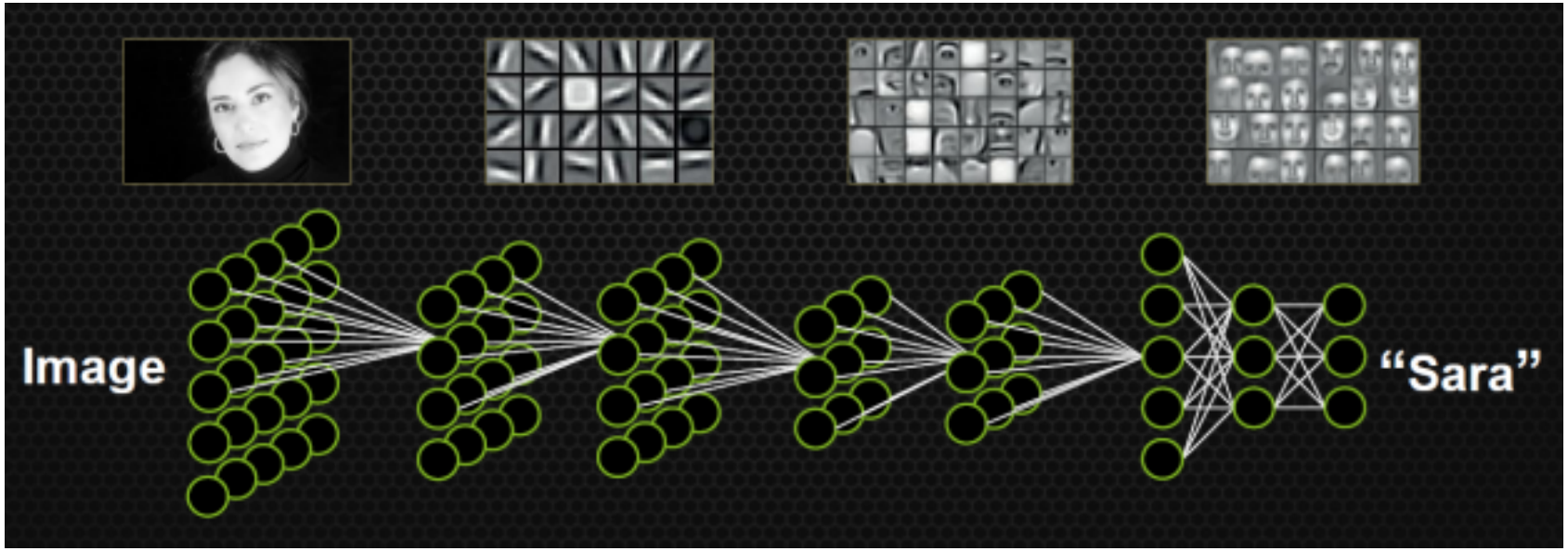

travail d’un neurone

Les réseaux de neurones sont des petits programmes.
On peut les ajuster petit à petit.

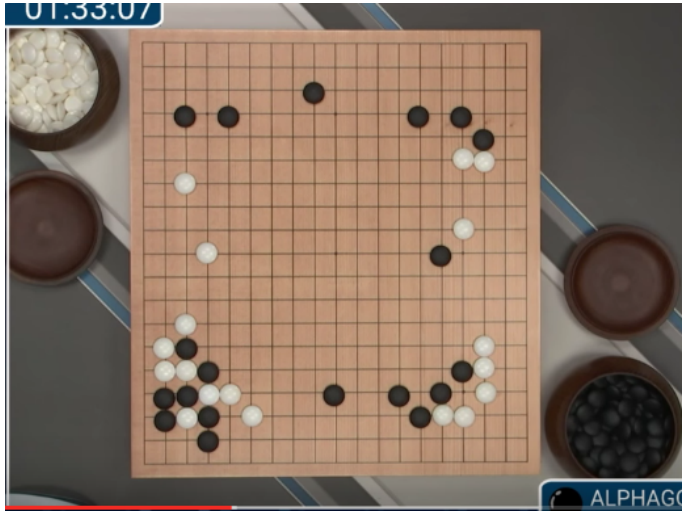
Les réseaux de neurones sont des petits programmes.
On peut les ajuster petit à petit.
Léger changement de paramètres → plus proche du résultat voulu.



**Premiers succès**: reconnaissance d'images
**Réseau profond**: identifie lui-meme les concepts pertinents.



**Mars 2016**: Défi de Google à Lee Sedol, champion de go, avec l'IA AlphaGo.




Victoire d'AlphaGo contre Lee Sedol 4-1 puis contre Ke Jie 5-0 en mai 2016.
Succès incontestable des réseaux de neurones.
AlphaGo trouve des coups qu'un humain n'aurait pas joué.

Différents **ingrédients**:

- Réseaux de neurones (2 types):
  - Estimer la valeur des positions
  - Proposer des bons coups
- Construction des réseaux:
  - 30 millions de parties humaines
  - *Renforcement*: jouer contre lui-même

Et la suite ?

- Commerce
- Problèmes éthiques ?
- Diagnostic médical
- Recherche scientifique
- ...
- Meilleure compréhension de l'humain ?