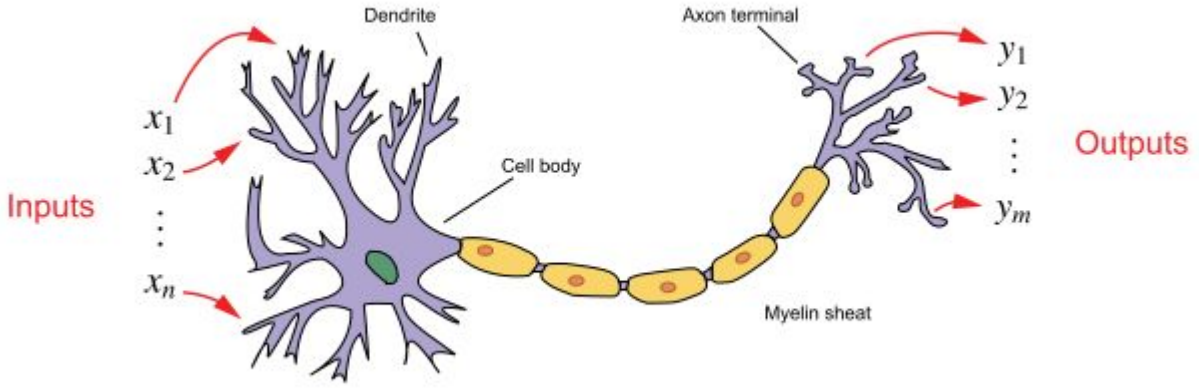
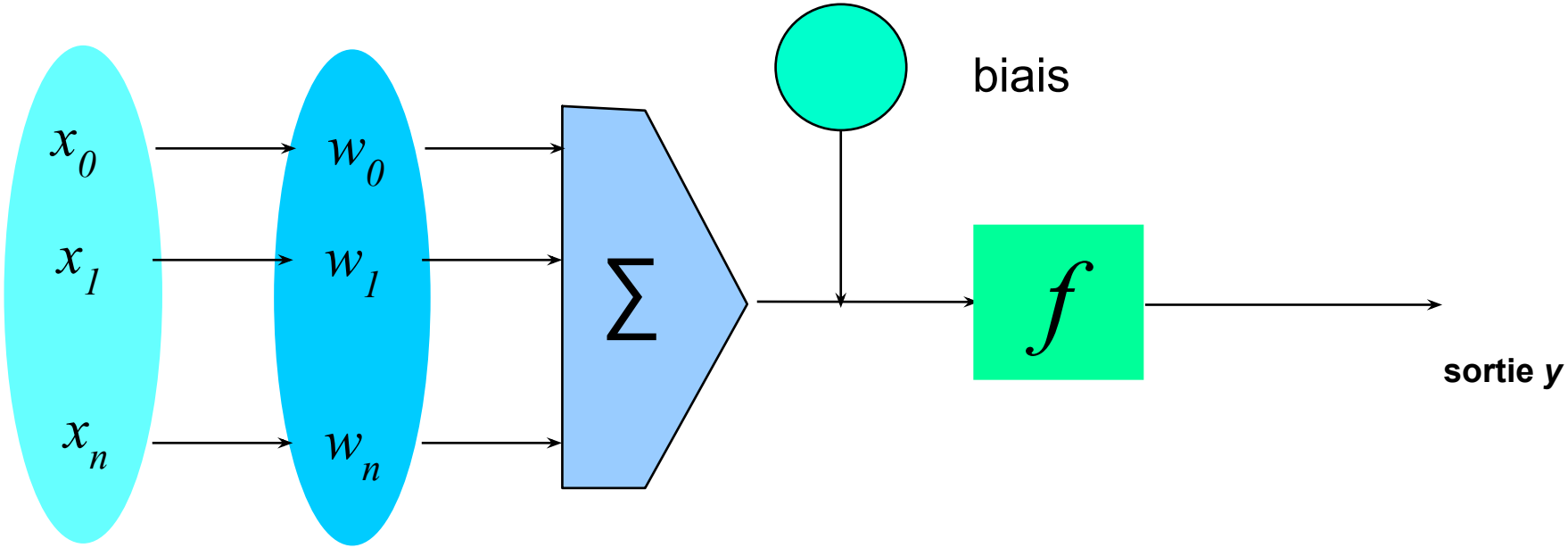
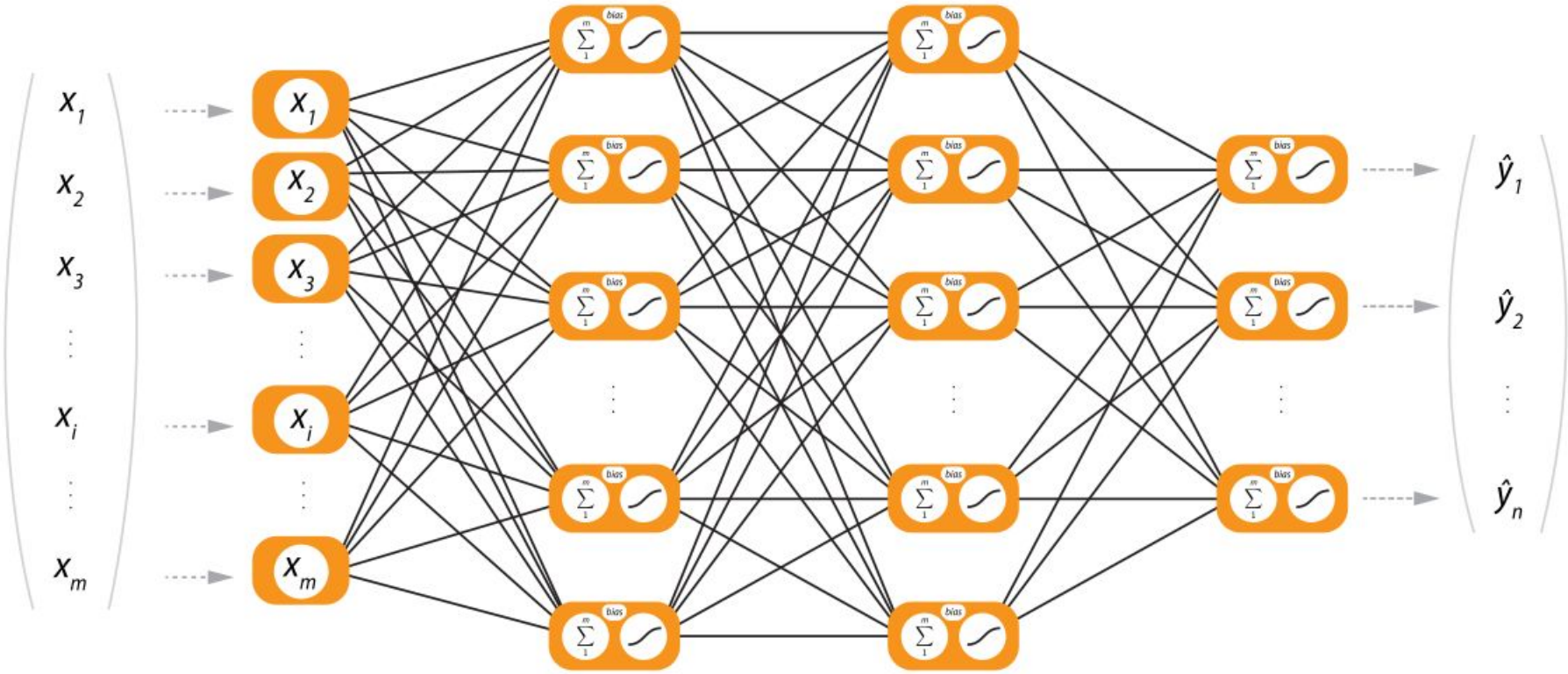
Un neurone



vecteur *x* n-dimensionnel est intégré en *y* par le produit scalaire ($x_i.w_i$), le biais et la fonction d'activation



Un réseau de neurones



couche d'entrée

couches cachées

couche de sortie

démo : http://playground.tensorflow.org

- Objectif
  - obtenir un ensemble de poids permettant de classer correctement (presque tous) les objets du jeu d'apprentissage

- Étapes
  - Initialiser les poids avec des valeurs aléatoires
  - Passer les objets au réseau un par un
  - pour chaque unité (neurone)
    - calculer l'entrée (combinaison linéaire de toutes les entrées)
    - calculer la sortie en utilisation la fonction d'activation
    - calculer l'erreur
    - mettre à jour le biais et les poids

Perceptron (multicouche) (eng: MLP pour *multilayer perceptron*)

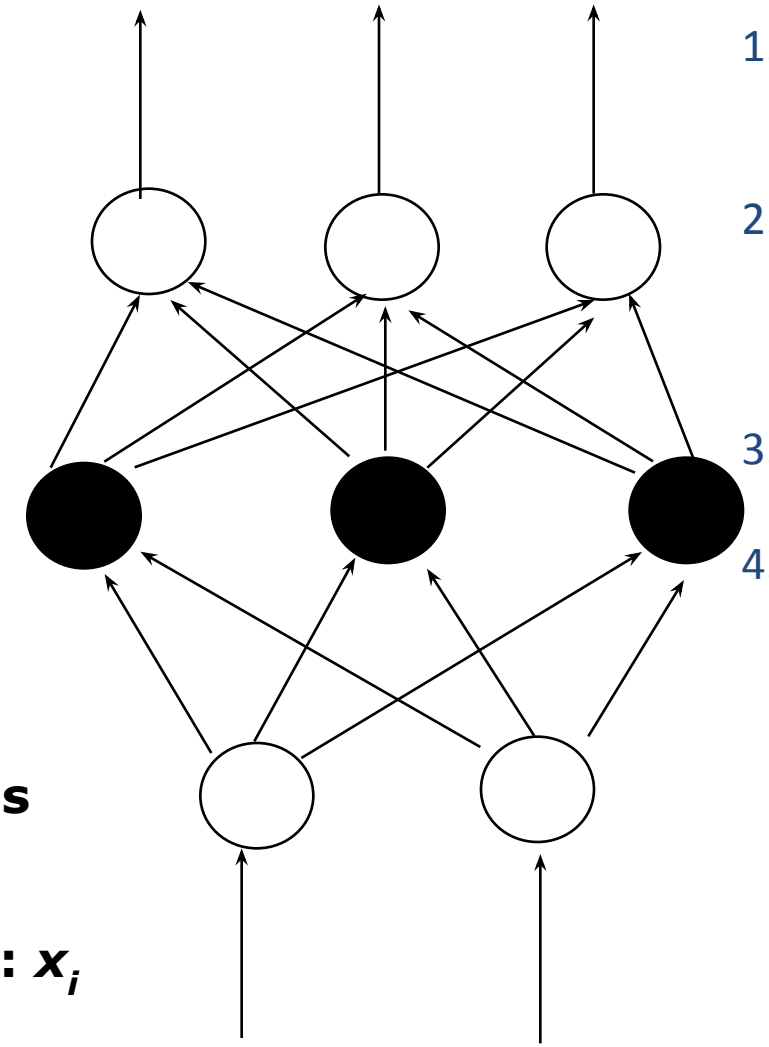
vecteur sortie

nœuds sorties

nœuds cachés

nœuds entrées

vecteur entrée : x_i



1. calcul de l'erreur en sortie pour chaque neurone
2. augmentation ou diminution du biais et du poids pour obtenir une faible erreur locale
3. attribution d'une erreur au neurones précédents
4. retour à l'étape 2 pour la couche précédente

- Avantages
  - précision souvent élevée
  - robuste, marche lorsque le jeu d'apprentissage contient des erreurs
  - sortie peut être valeur discrète, continue, ou un vecteur de plusieurs valeurs discrètes ou continues

- Critiques
  - apprentissage long
  - difficile de comprendre le modèle
  - difficile d'incorporer des connaissances