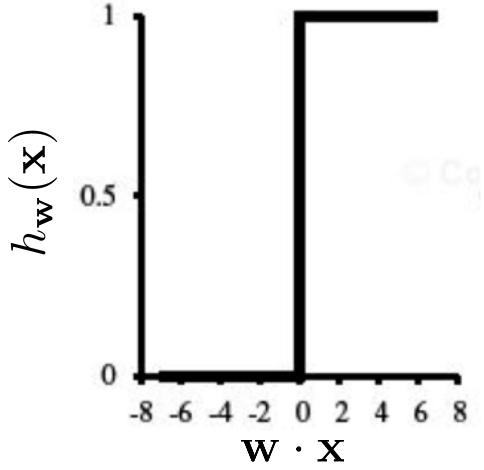
Apprentissage vue comme la minimisation d'une perte

- La procédure de descente de gradient stochastique est applicable à n'importe quelle perte dérivable partout

- Dans le cas du perceptron, on a un peu triché:
  - la dérivée de $h_{\mathbf{w}}(\mathbf{x})$ n'est pas définie lorsque $\mathbf{w} \cdot \mathbf{x} = 0$

- L'utilisation de la fonction *Threshold* (qui est constante par partie) fait que la courbe d'entraînement peut être instable

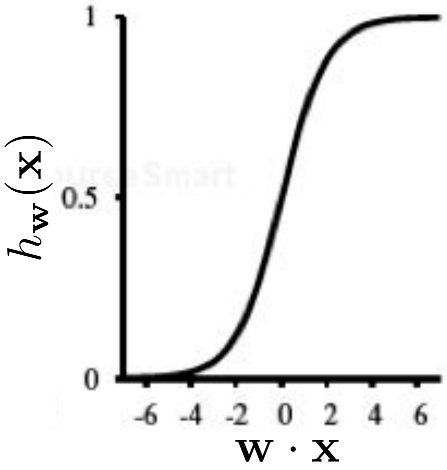


Hugo Larochelle et Froduald Kabanza

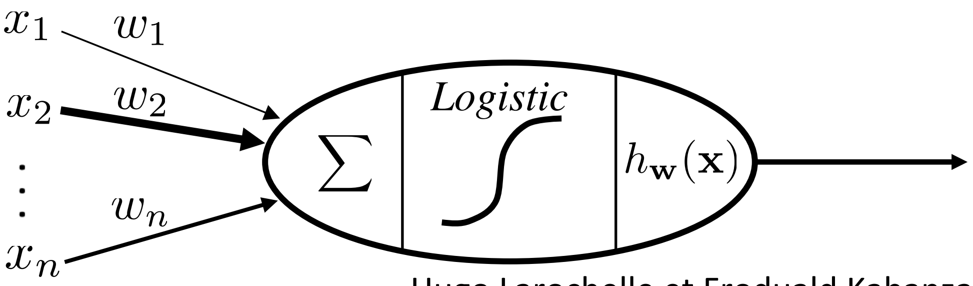
Régression logistique

- **Idée**: plutôt que de prédire une classe, prédire une probabilité d'appartenir à la classe 1

$$p(y = 1|\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$



- Choisir la classe la plus probable selon le modèle
  - si $h_{\mathbf{w}}(\mathbf{x}) >= 0.5$ choisir la classe 1
  - sinon, choisir la classe 0



Dérivation de la règle d'apprentissage

- Pour obtenir une règle d'apprentissage, on définit d'abord une perte

$$Loss(y_t, h_{\mathbf{w}}(\mathbf{x}_t)) = -y_t \log h_{\mathbf{w}}(\mathbf{x}_t) - (1 - y_t) \log(1 - h_{\mathbf{w}}(\mathbf{x}_t))$$

  - si $y_t = 1$, on souhaite maximiser la probabilité $p(y_t = 1|\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x}_t)$
  - si $y_t = 0$, on souhaite maximiser la probabilité $p(y_t = 0|\mathbf{x}) = 1 - h_{\mathbf{w}}(\mathbf{x}_t)$

- On dérive la règle d'apprentissage comme une descente de gradient

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(y_t, h_{\mathbf{w}}(\mathbf{x}_t)) \quad \forall i$$

  ce qui donne

$$w_i \leftarrow w_i + \alpha(y_t - h_{\mathbf{w}}(\mathbf{x}_t))x_{t,i} \quad \forall i$$

- La règle est donc la même que pour le Perceptron, mais la définition de $h_{\mathbf{w}}(\mathbf{x}_t)$ est différente