La compression d'échantillons sans limite : De nouveaux résultats pour les pertes à valeurs réelles

Mathieu Bazinet

24 janvier 2025

Avant de commencer

J'ai essayé de rendre la présentation courte et compréhensible. Mais elle est probablement longue et, je l'espère, compréhensible. On verra!

Avant de commencer

J'ai essayé de rendre la présentation courte et compréhensible. Mais elle est probablement longue et, je l'espère, compréhensible. On verra!

Je vais essayer de prendre des questions, mais quand j'ai testé la présentation, j'ai parlé 45 minutes et j'ai un cours dans une heure (à 14h30).

Avant de commencer

J'ai essayé de rendre la présentation courte et compréhensible. Mais elle est probablement longue et, je l'espère, compréhensible. On verra!

Je vais essayer de prendre des questions, mais quand j'ai testé la présentation, j'ai parlé 45 minutes et j'ai un cours dans une heure (à 14h30).
Bonne écoute!

Impact du papier

Le papier a été accepté à AISTATS 2025.
De plus, une version courte du papier a été acceptée ici :

1. Workshop on Machine Learning and Compression de NeurIPS 2024
2. Workshop on Mathematics of Modern Machine Learning de NeurIPS 2024

Impact du papier

Le papier a été accepté à AISTATS 2025.
De plus, une version courte du papier a été acceptée ici :

1. Workshop on Machine Learning and Compression de NeurIPS 2024
2. Workshop on Mathematics of Modern Machine Learning de NeurIPS 2024

Ces travaux sont à la base des travaux suivants :

1. Sample Compression Hypernetworks: From Generalization Bounds to Meta-Learning. Leblanc et al. (Présenté au workshop de compression de NeurIPS 2024, va être soumis à ICML 2025)
2. Sample Compression for Continual Learning. Comeau et al. (Va être soumis à ICML ou UAI 2025)

Problématique

Le but de ce projet est de proposer de nouvelles garanties de généralisations, dans le cadre de la théorie de la compression d'échantillons, qui pourraient être utilisés dans le cadre de l'apprentissage profond.

Problématique

Le but de ce projet est de proposer de nouvelles garanties de généralisations, dans le cadre de la théorie de la compression d'échantillons, qui pourraient être utilisés dans le cadre de l'apprentissage profond.

1. Théorie
   - Garanties de généralisations
   - Compression d'échantillons
2. Nouveaux résultats
3. Expériences
   - Binary MNIST
   - Forêts de régression
   - Amazon Polarity
4. Conclusion

Théorie

Garanties de généralisations

Exemple : Régression

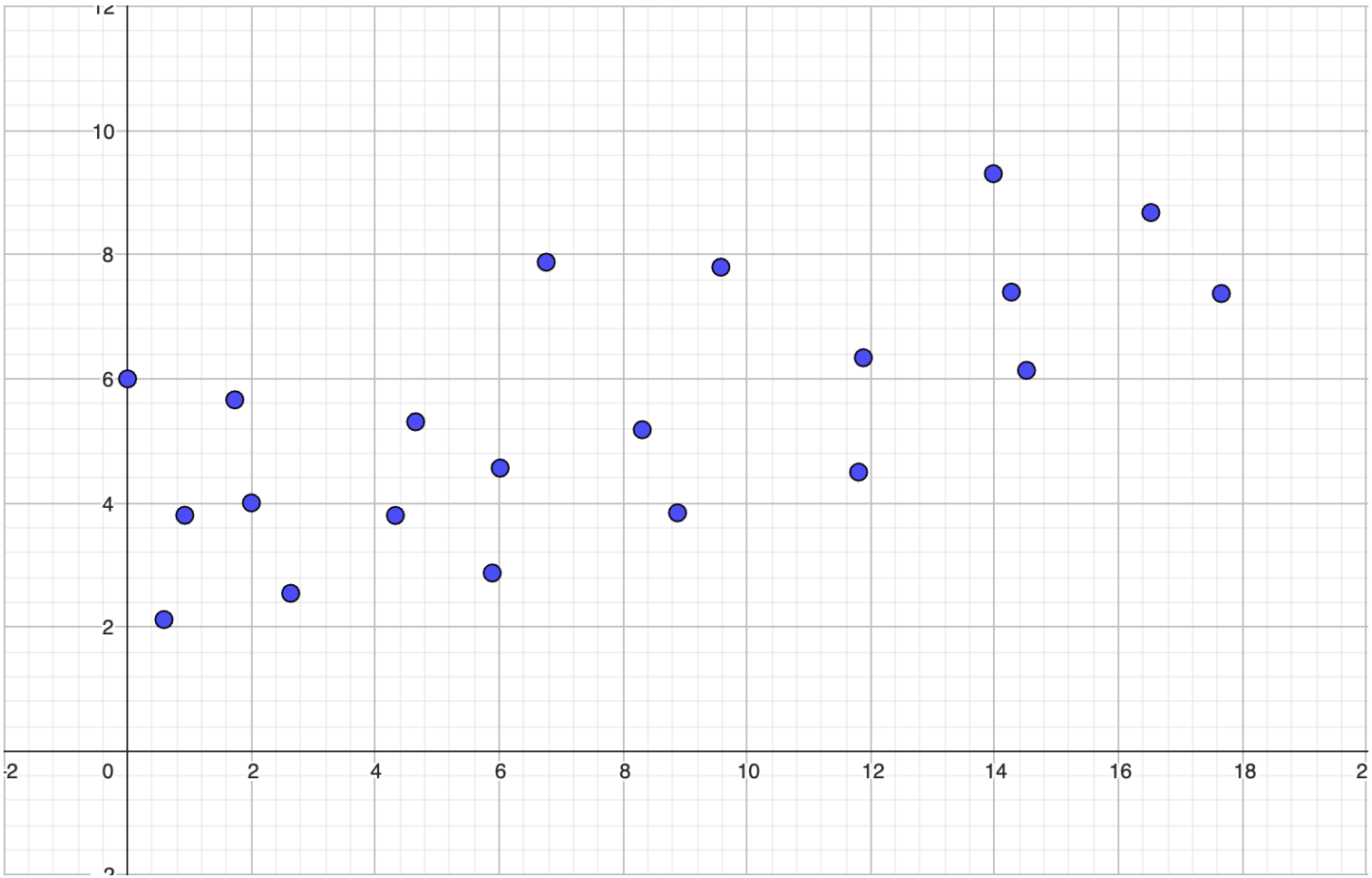


Figure 1: Jeu de données sur lequel on veut faire une régression

Exemple : Régression

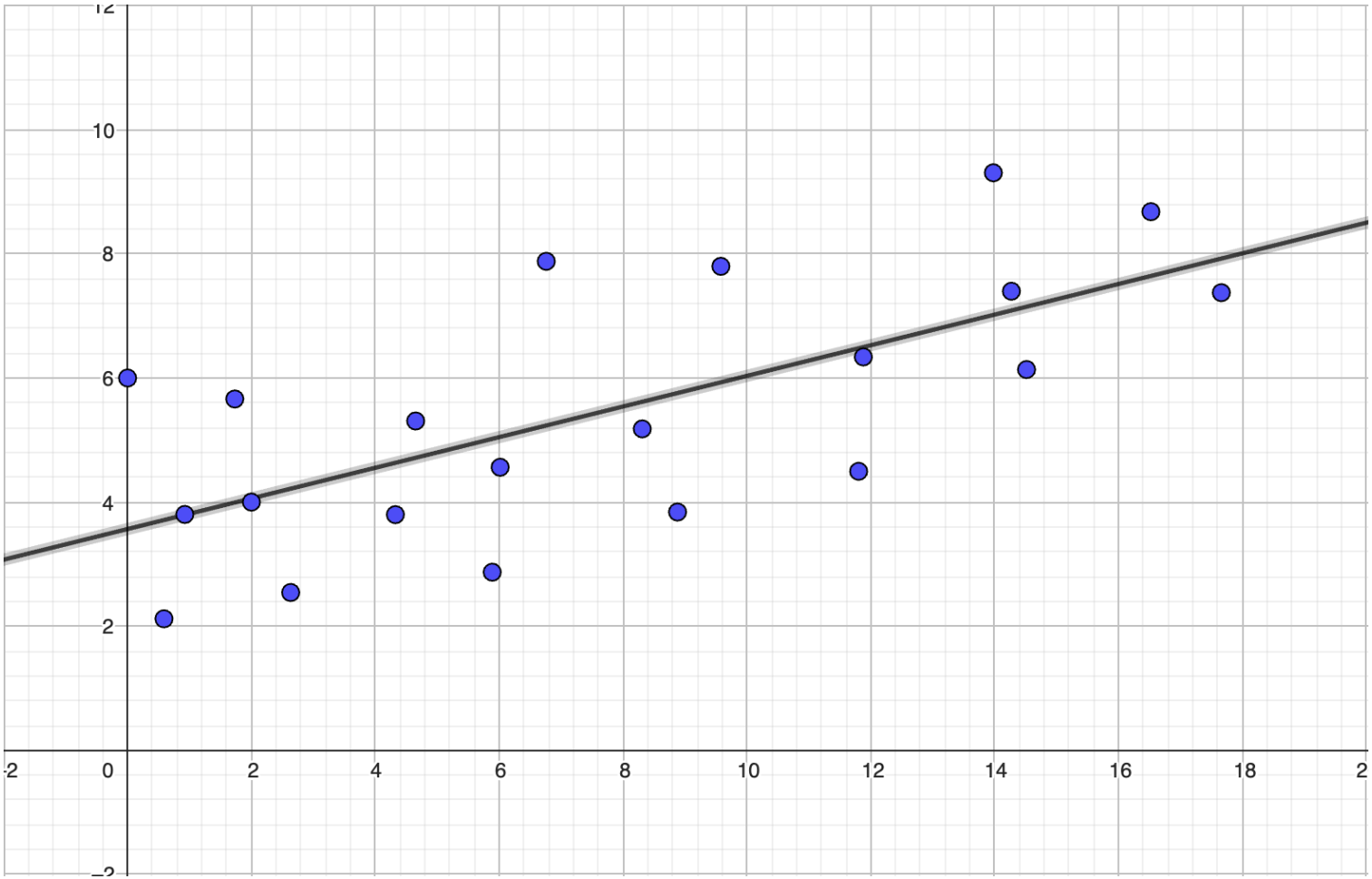


Figure 1: Régression linéaire (ordre 1)

Exemple : Régression

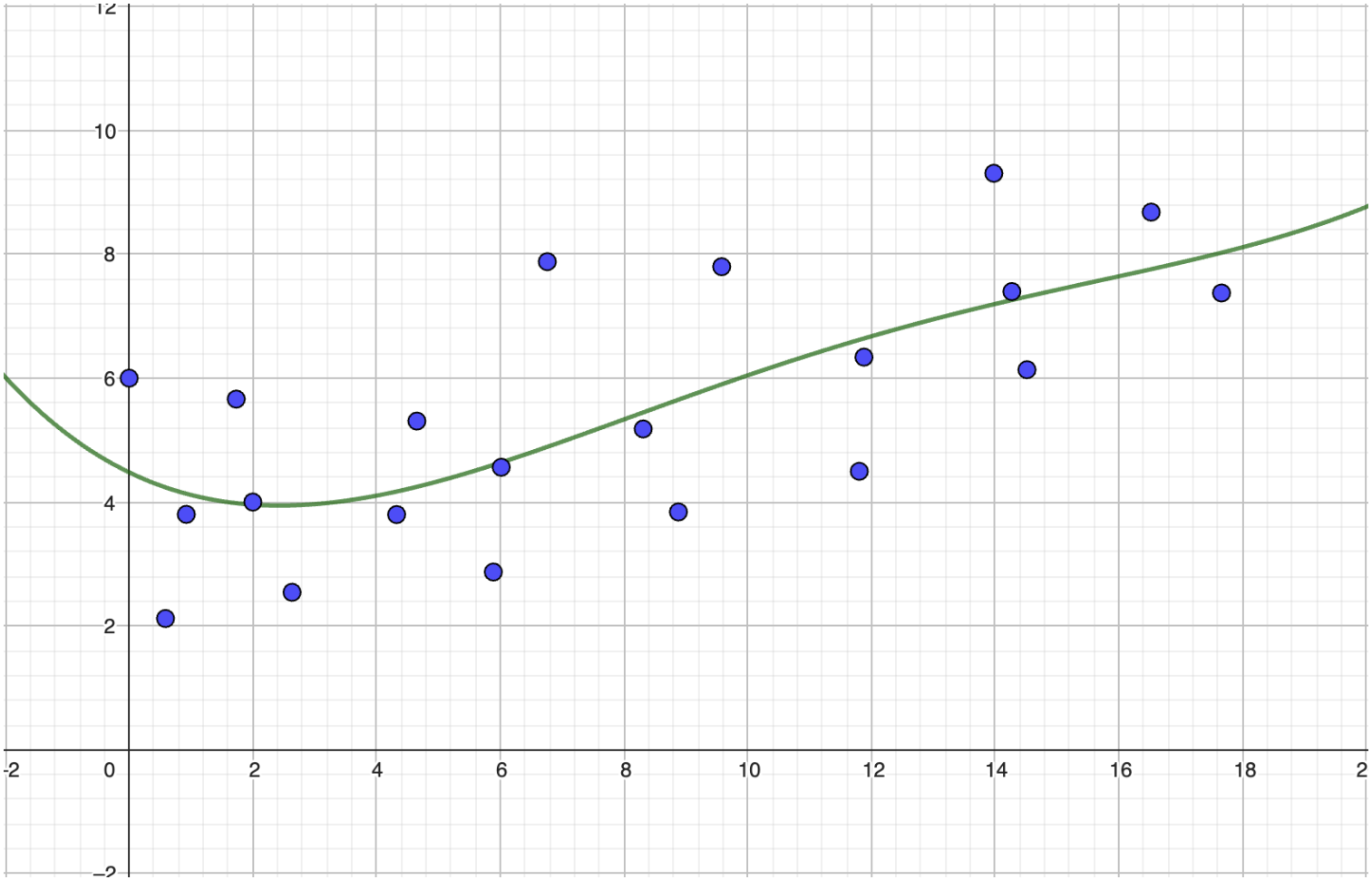


Figure 1: Régression polynômiale d'ordre 4

Exemple : Régression

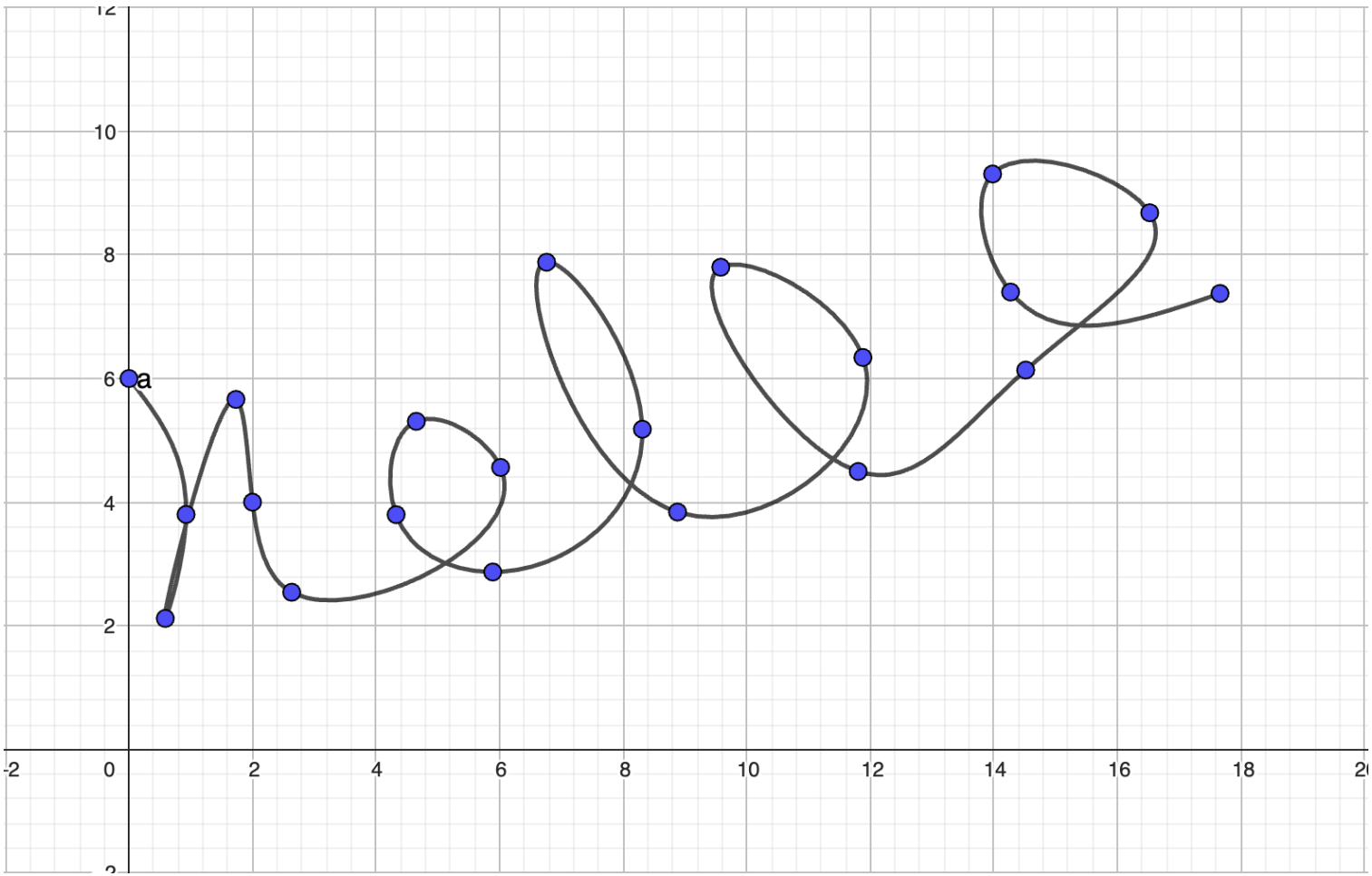


Figure 1: Régression sur-ajustée aux données

Quelle régression choisir?

Si je pige une nouvelle donnée, quelle courbe a le plus de chance d'être bien ajustée?

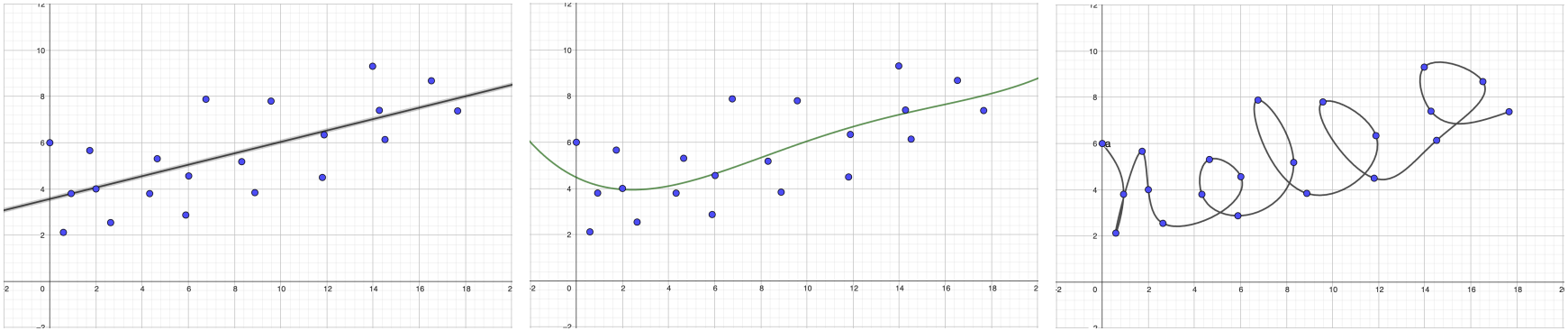


Figure 2: Trois différentes courbes ajustées aux données

Quelle régression choisir?

Si je pige une nouvelle donnée, quelle courbe a le plus de chance d'être bien ajustée?

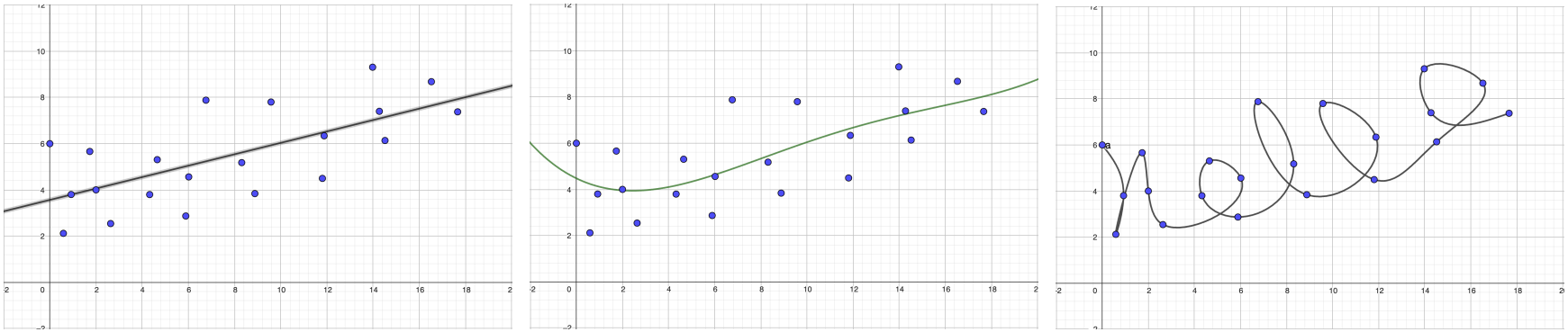


Figure 2: Trois différentes courbes ajustées aux données

On peut utiliser son intuition, par exemple en gardant un sous-ensemble des données comme ensemble de validation. Mais peut-on faire mieux?

Garanties de généralisations

Les garanties de généralisations permettent de garantir, avec forte probabilité, que la perte sur des données jamais vues ne sera jamais pire qu'une certaine valeur.

Garanties de généralisations

Les garanties de généralisations permettent de garantir, avec forte probabilité, que la perte sur des données jamais vues ne sera jamais pire qu'une certaine valeur.

Pour obtenir des garanties de généralisations, nous allons avoir besoin de plusieurs ingrédients.

1. Un jeu de données S
2. Une classe de prédicteurs $\mathcal{H}$
3. Une perte ℓ

Jeu de données

On suppose que toutes les données sont pigées indépendamment dans la même distribution (*i.i.d. assumption*).

On dénote le jeu de données $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, qui contient n données pigées *i.i.d.* de la distribution génératrice $\mathcal{D}$ sur l'espace des données $\mathcal{X}$ et des étiquettes $\mathcal{Y}$.

Jeu de données

On suppose que toutes les données sont pigées indépendamment dans la même distribution (*i.i.d. assumption*).

On dénote le jeu de données $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, qui contient n données pigées *i.i.d.* de la distribution génératrice $\mathcal{D}$ sur l'espace des données $\mathcal{X}$ et des étiquettes $\mathcal{Y}$.

Dans cette présentation, je suppose qu'on travaille en apprentissage supervisée, même si mes résultats n'y sont pas restreints.

Jeu de données

Concrètement, supposons que j'ai un jeu de données où je souhaite classifier lequel de mes chats se trouve dans la photo.

Jeu de données

Alors, si je pige une image aléatoirement, je m'attends à trouver une nouvelle photo d'un de mes deux chats. De plus, avoir pigé la première photo ne devrait pas influencer la probabilité de piger la deuxième.



Classe de prédicteurs

Avant de commencer l'entraînement, on doit choisir une classe de prédicteurs.

Classe de prédicteurs

Avant de commencer l'entraînement, on doit choisir une classe de prédicteurs.

Par exemple, on pourrait choisir la classe des réseaux de neurones avec deux couches cachées contenant chacune 512 neurones.

Classe de prédicteurs

Avant de commencer l'entraînement, on doit choisir une classe de prédicteurs.

Par exemple, on pourrait choisir la classe des réseaux de neurones avec deux couches cachées contenant chacune 512 neurones.

Chaque théorie permet de travailler avec certains types de prédicteurs. Les garanties PAC supportent les arbres de décisions [4], tandis que le PAC-Bayes supporte les réseaux de neurones probabilistes [2, 6].

Fonction de perte

La perte est l'objectif que nous allons souhaiter minimiser.

Posons la perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

Fonction de perte

La perte est l'objectif que nous allons souhaiter minimiser.

Posons la perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. La perte sur le jeu de données est dénotée :

$$\widehat{\mathcal{L}}_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, \boldsymbol{x}_i, y_i)$$

Fonction de perte

La perte est l'objectif que nous allons souhaiter minimiser.

Posons la perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. La perte sur le jeu de données est dénotée :

$$\widehat{\mathcal{L}}_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, \boldsymbol{x}_i, y_i)$$

La perte en test (aussi appelée le vrai risque) est dénotée :

$$\mathcal{L}_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}} \ell(h, \boldsymbol{x}, y)$$

Fonction de perte

La perte est l'objectif que nous allons souhaiter minimiser.

Posons la perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. La perte sur le jeu de données est dénotée :

$$\widehat{\mathcal{L}}_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, \boldsymbol{x}_i, y_i)$$

La perte en test (aussi appelée le vrai risque) est dénotée :

$$\mathcal{L}_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}} \ell(h, \boldsymbol{x}, y)$$

Lors d'un problème de classification, on utilisera généralement la perte 0-1, et l'on dénotera la perte empirique $\hat{R}_S$ et le vrai risque $R_{\mathcal{D}}$.

Garantie de généralisation

Maintenant que l'on a tous les ingrédients, voyons maintenant de quoi à l'air une garantie de généralisation.

Avec $\Delta(q, p)$ une pseudo-distance et

, on a

$$\Delta\Big(\widehat{\mathcal{L}}_S(h), \mathcal{L}_{\mathcal{D}}(h)\Big) \qquad .$$

Garantie de généralisation

Maintenant que l'on a tous les ingrédients, voyons maintenant de quoi à l'air une garantie de généralisation.

Avec $\Delta(q, p)$ une pseudo-distance et $\epsilon(h, \delta)$ un terme de complexité spécifique à la classe de prédicteur, on a

$$\Delta\Big(\widehat{\mathcal{L}}_S(h), \mathcal{L}_{\mathcal{D}}(h)\Big) \leq \epsilon(h, \delta) \qquad .$$

Garantie de généralisation

Maintenant que l'on a tous les ingrédients, voyons maintenant de quoi à l'air une garantie de généralisation.

Avec $\Delta(q, p)$ une pseudo-distance et $\epsilon(h, \delta)$ un terme de complexité spécifique à la classe de prédicteur, on a

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}\left(\Delta\left(\widehat{\mathcal{L}}_S(h), \mathcal{L}_{\mathcal{D}}(h)\right) \leq \epsilon(h, \delta)\right) \geq 1 - \delta.$$

Garantie de généralisation

Il est beaucoup plus intuitif de choisir $\Delta(q, p)$ comme la différence entre le vrai risque et le risque empirique.

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^n} \left(\mathcal{L}_{\mathcal{D}}(h) \leq \widehat{\mathcal{L}}_S(h) + \epsilon(h, \delta) \right) \geq 1 - \delta.$$

Le côté droit de l'équation est donc une borne supérieure sur le vrai risque, qui tient avec une probabilité d'au moins $1 - \delta$.

Compression d'échantillons

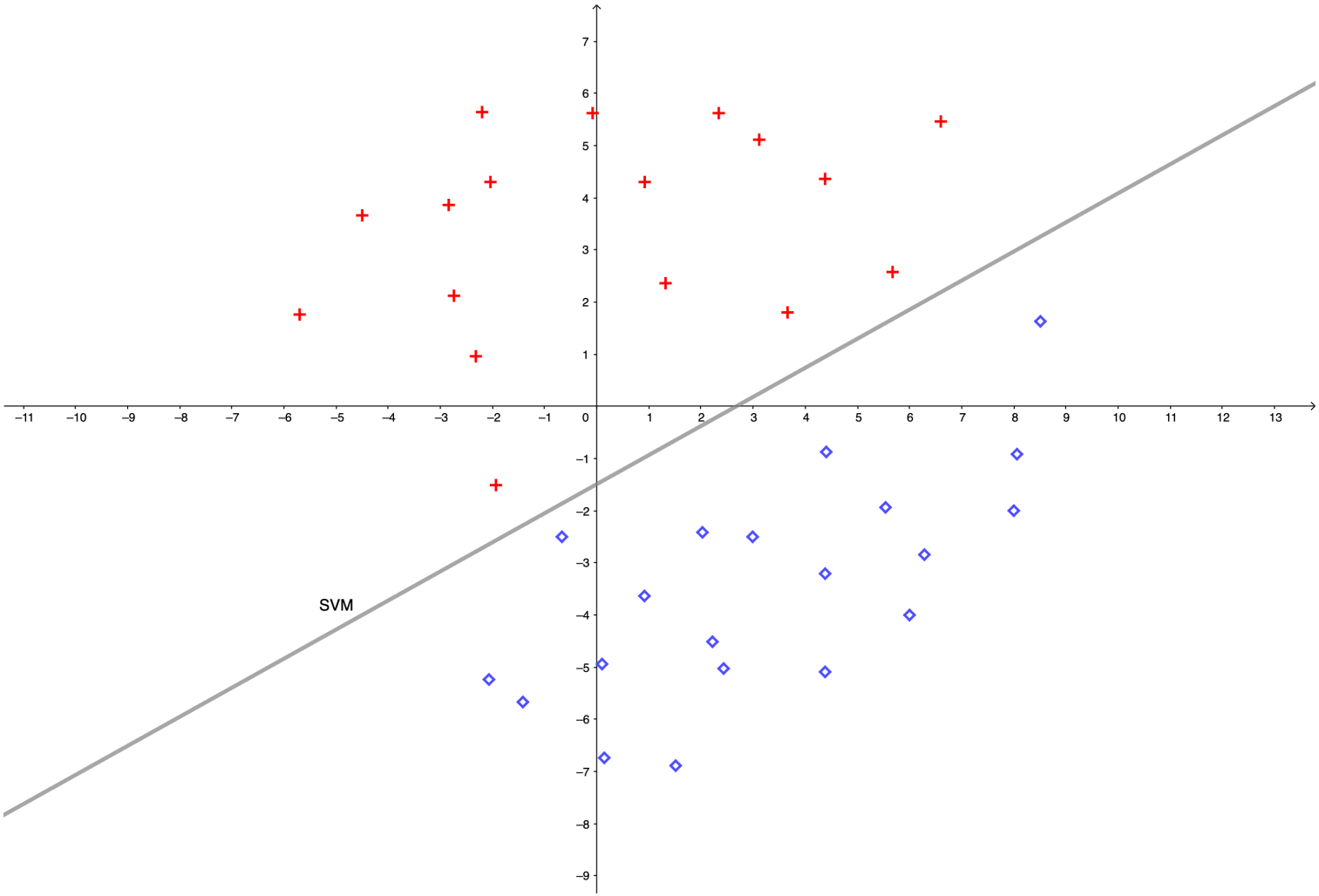


Figure 3: Un séparateur à vaste marge (SVM) sur un jeu de données de classification binaire

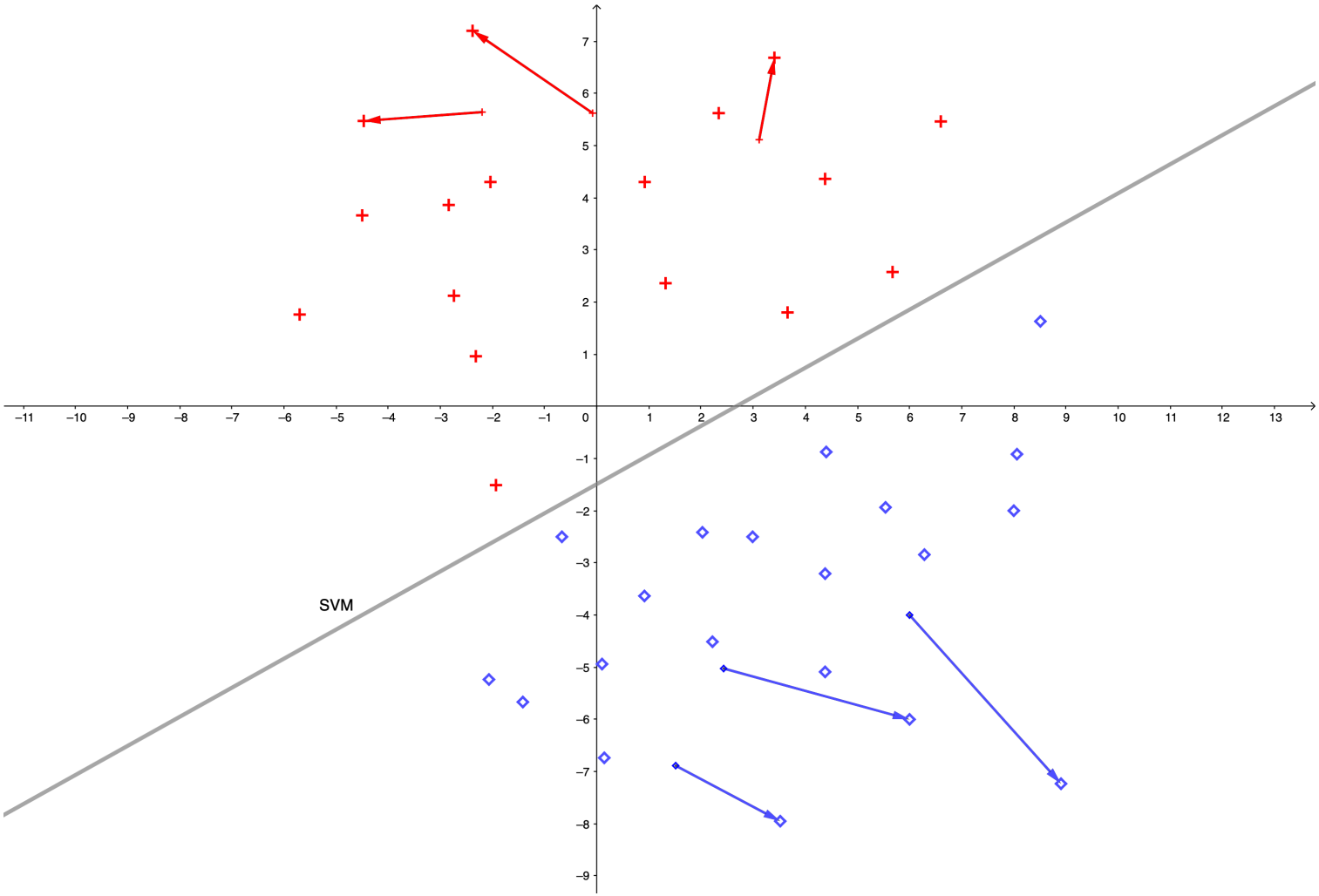


Figure 3: Modifier des données loin de la frontière de décision ne change rien au prédicteur

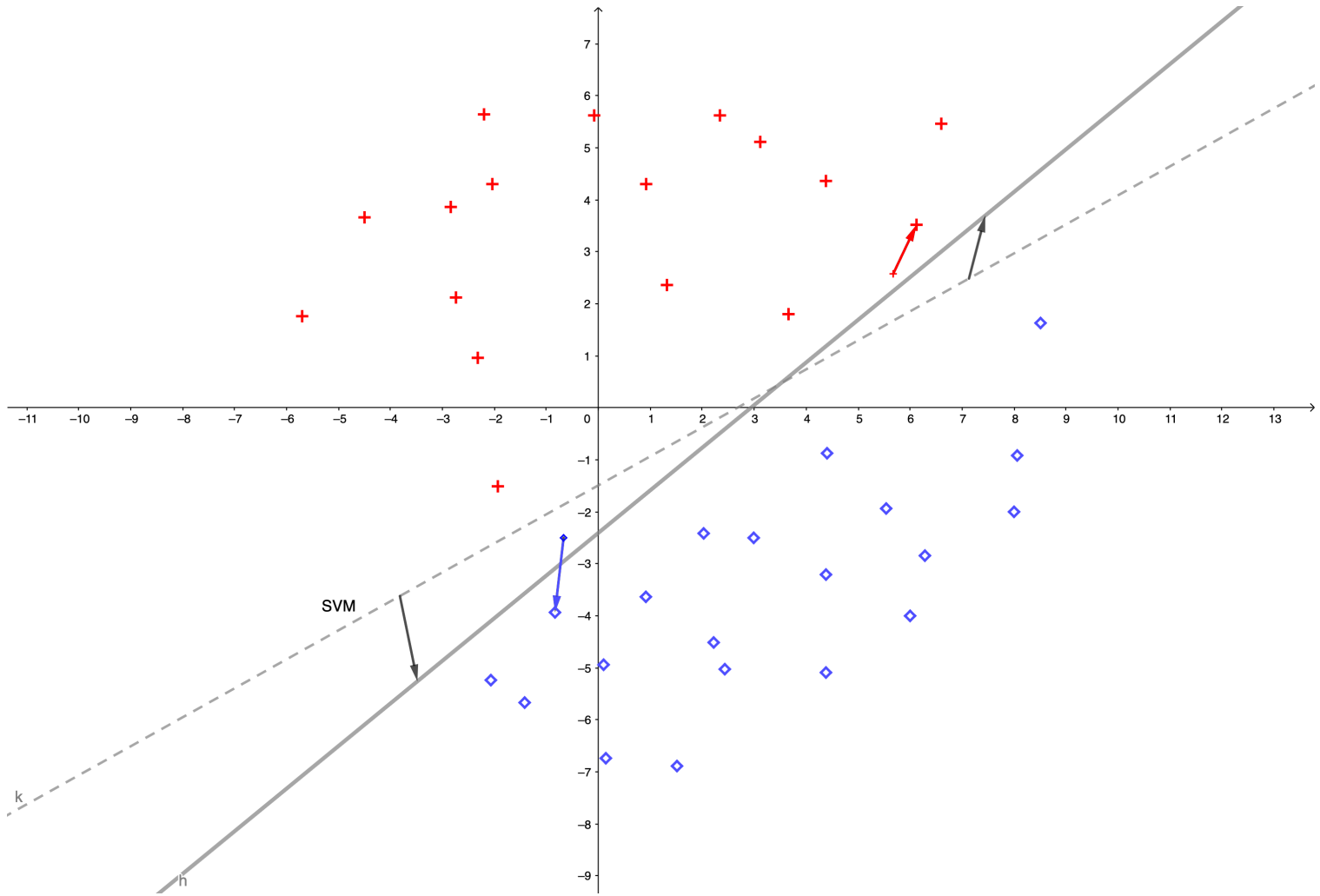


Figure 3: Modifier des données proche de la frontière de décision modifie le prédicteur

Arbres de décisions

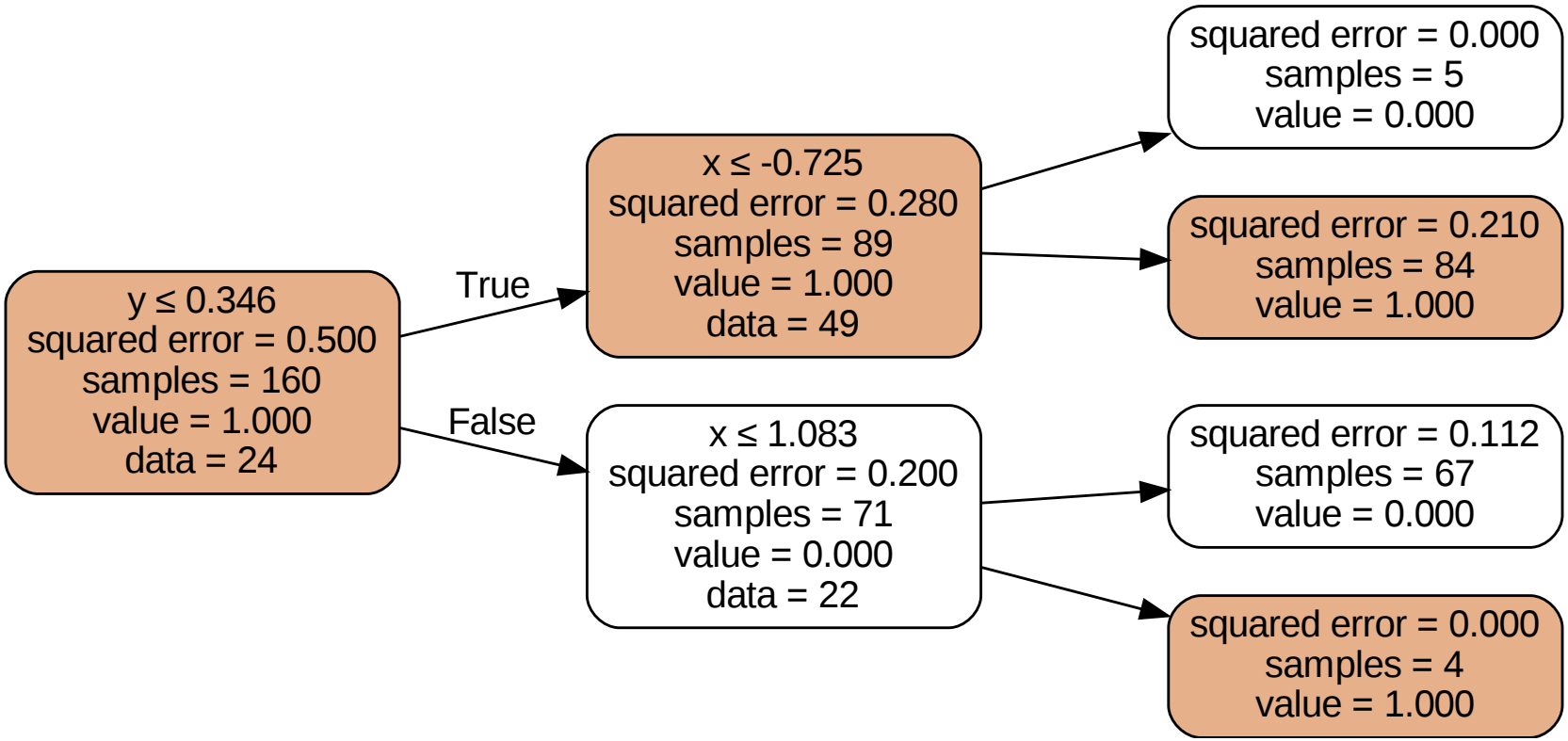


Figure 4: Un arbre de décision entraîné sur les deux lunes[1]

[1] L'implémentation que j'ai choisie est spéciale, puisque les frontières de décisions sont directement placées sur les données au lieu d'entre deux données.

Arbres de décisions

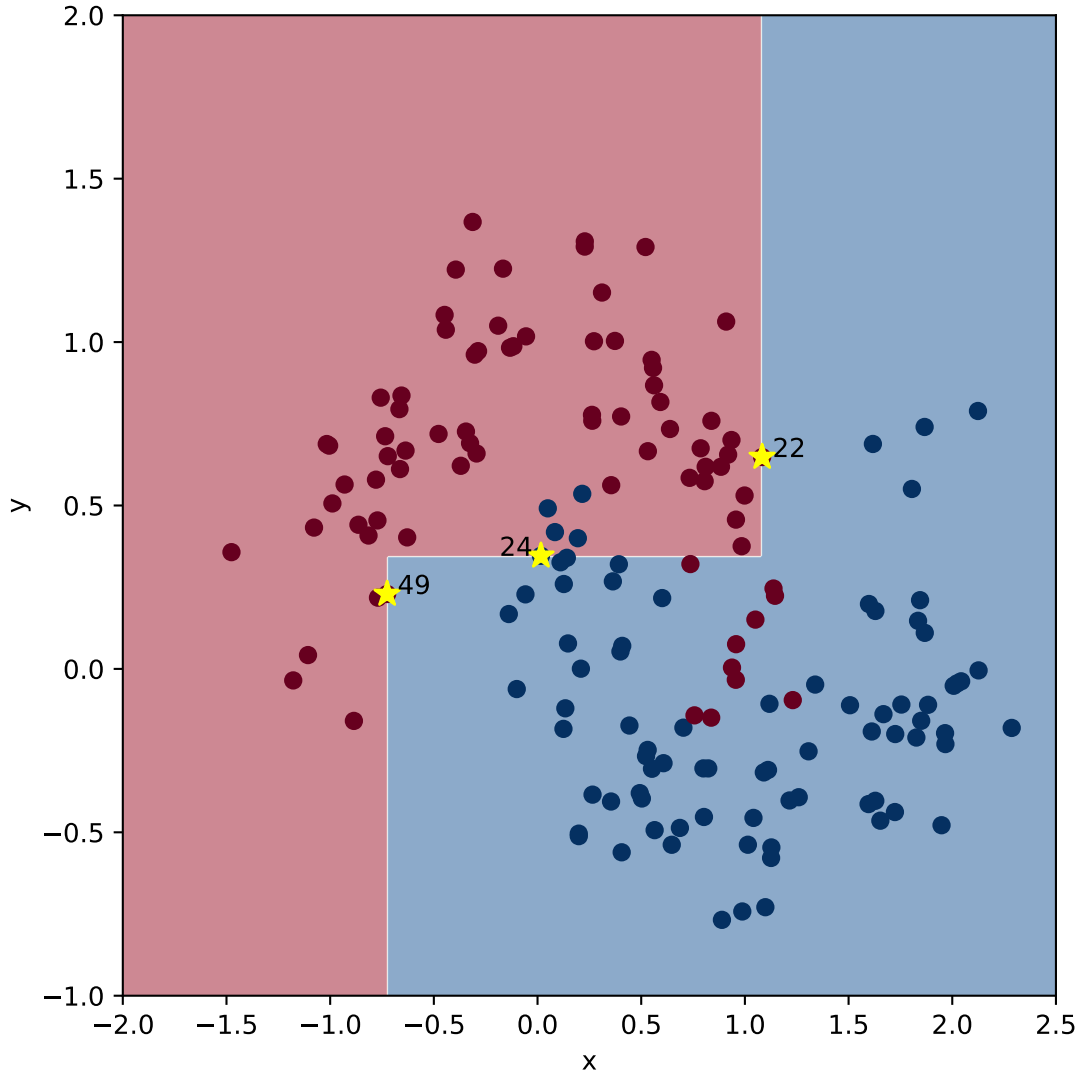
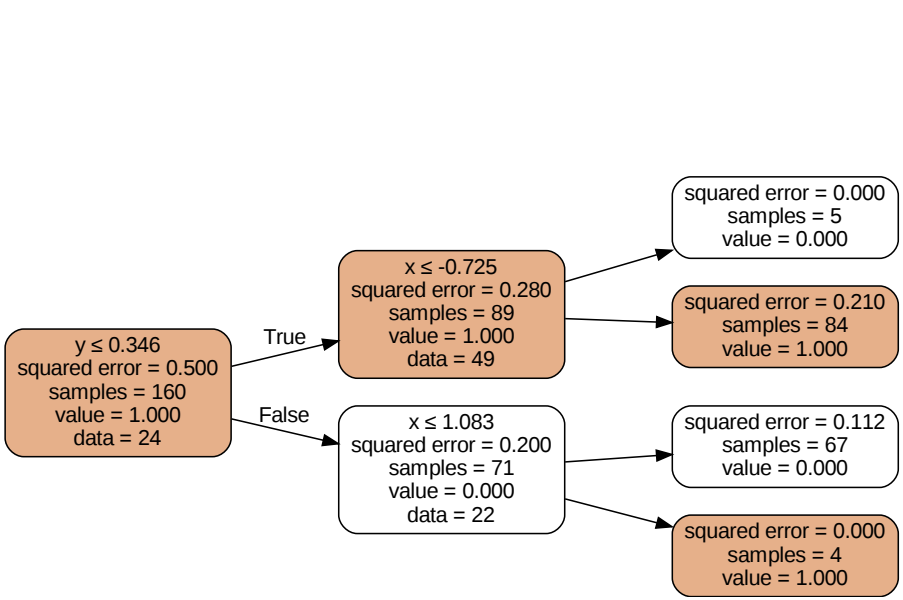
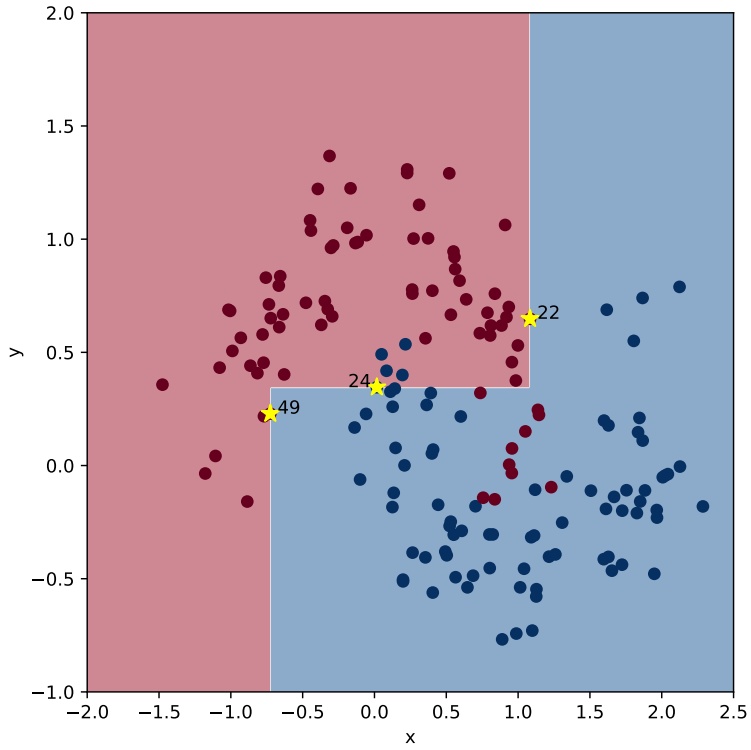


Figure 5: Frontière de décision de l'arbre

Arbres de décisions



(a) Arbre de décision



(b) Frontière de décision

Pour faire la reconstruction, j'ai donc besoin d'avoir la donnée, le *feature* ainsi que le noeud dans l'arbre où utiliser la donnée.

Compression d'échantillons

Pour avoir des garanties de compression d'échantillons, on a donc besoin de trois composantes :

1. Une fonction de reconstruction $\mathcal{R}$
2. Un ensemble de compression $S_{\mathbf{i}}$, défini par le vecteur d'index $\mathbf{i}$
3. Un message σ

Compression d'échantillons

Pour avoir des garanties de compression d'échantillons, on a donc besoin de trois composantes :

1. Une fonction de reconstruction $\mathcal{R}$
2. Un ensemble de compression $S_{\mathbf{i}}$, défini par le vecteur d'index $\mathbf{i}$
3. Un message σ

Pour n'importe quel prédicteur $A(S)$ entraîné sur un jeu de données S, s'il est possible d'écrire $A(S) = \mathcal{R}(S_{\mathbf{i}}, \sigma)$, alors on peut avoir des garanties de généralisations.

Résultats précédents

Théorème ([3])

Pour toute distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, pour toute famille d'ensembles de messages $\{M(\mathbf{i}) \,|\, \mathbf{i} \in \mathcal{P}(n)\}$, pour toute fonction de reconstruction déterministes $\mathcal{R}$ qui produit des prédicteurs compressés $h \in \mathcal{H}$ et pour tout $\delta \in (0, 1]$, avec une probabilité d'au moins $1 - \delta$ sur le tirage de $S \sim \mathcal{D}^n$, nous avons

$$\forall\, \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) :$$

$$R_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)) \leq \overline{Bin}\Big(|\,\mathbf{i}^c\,|, \widehat{R}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)), n, P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})}(\sigma)\delta\Big)$$

avec

$$\overline{Bin}(k, m, \delta) = \sup_{r \in [0,1]} \left\{ \sum_{i=0}^{k} \binom{m}{i} r^i (1-r)^{m-i} \geq \delta \right\}.$$

Nouveaux résultats

Nouvelle borne

Théorème

Pour toute distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, pour toute famille d'ensembles de messages $\{M(\mathbf{i}), | \, \mathbf{i} \in \mathcal{P}(n)\}$, pour toute fonction de reconstruction déterministe $\mathcal{R}$ qui produit des prédicteurs compressés $h \in \mathcal{H}$, pour toute perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, pour toute fonction de comparaison $\Delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ et pour tout $\delta \in (0, 1]$, avec une probabilité d'au moins $1 - \delta$ sur le tirage de $S \sim \mathcal{D}^n$, nous avons

$$\forall \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) :$$

$$\Delta\Big(\widehat{\mathcal{L}}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma))\Big) \leq \frac{1}{|\,\mathbf{i}^c\,|} \log\left(\frac{1}{P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})}(\sigma)} \frac{\mathcal{E}_{\Delta}(\mathbf{i}, \sigma)}{\delta}\right)$$

avec

$$\mathcal{E}_{\Delta}(\mathbf{i}, \sigma) = \mathop{\mathbb{E}}_{T_{\mathbf{i}} \sim \mathcal{D}^{|\mathbf{i}|}} \mathop{\mathbb{E}}_{T_{\mathbf{i}^c} \sim \mathcal{D}^{|\mathbf{i}^c|}} e^{|\mathbf{i}^c| \Delta\left(\widehat{\mathcal{L}}_{T_{\mathbf{i}^c}}(\mathcal{R}(T_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(T_{\mathbf{i}}, \sigma))\right)}.$$

Nouvelle borne

Théorème

Pour toute distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, pour toute famille d'ensembles de messages $\{M(\mathbf{i}), | \, \mathbf{i} \in \mathcal{P}(n)\}$, pour toute fonction de reconstruction déterministe $\mathcal{R}$ qui produit des prédicteurs compressés $h \in \mathcal{H}$, pour toute perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, pour toute fonction de comparaison $\Delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ et pour tout $\delta \in (0, 1]$, avec une probabilité d'au moins $1 - \delta$ sur le tirage de $S \sim \mathcal{D}^n$, nous avons

$$\forall \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) :$$

$$\Delta\Big(\widehat{\mathcal{L}}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma))\Big) \leq \frac{1}{|\,\mathbf{i}^c\,|} \log\left(\frac{1}{P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})}(\sigma)} \frac{\mathcal{E}_\Delta(\mathbf{i}, \sigma)}{\delta}\right)$$

avec

$$\mathcal{E}_\Delta(\mathbf{i}, \sigma) = \mathop{\mathbb{E}}_{T_{\mathbf{i}} \sim \mathcal{D}^{|\mathbf{i}|}} \mathop{\mathbb{E}}_{T_{\mathbf{i}^c} \sim \mathcal{D}^{|\mathbf{i}^c|}} e^{|\mathbf{i}^c| \Delta\left(\widehat{\mathcal{L}}_{T_{\mathbf{i}^c}}(\mathcal{R}(T_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(T_{\mathbf{i}}, \sigma))\right)}.$$

Nouvelle borne

Théorème

Pour toute distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, pour toute famille d'ensembles de messages $\{M(\mathbf{i}), | \, \mathbf{i} \in \mathcal{P}(n)\}$, pour toute fonction de reconstruction déterministe $\mathcal{R}$ qui produit des prédicteurs compressés $h \in \mathcal{H}$, pour toute perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, pour toute fonction de comparaison $\Delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ et pour tout $\delta \in (0, 1]$, avec une probabilité d'au moins $1 - \delta$ sur le tirage de $S \sim \mathcal{D}^n$, nous avons

$$\forall \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) :$$

$$\Delta\Big(\widehat{\mathcal{L}}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma))\Big) \leq \frac{1}{|\,\mathbf{i}^c\,|} \log\left(\frac{1}{P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})}(\sigma)} \frac{\mathcal{E}_\Delta(\mathbf{i}, \sigma)}{\delta}\right)$$

avec

$$\mathcal{E}_\Delta(\mathbf{i}, \sigma) = \mathop{\mathbb{E}}_{T_{\mathbf{i}} \sim \mathcal{D}^{|\mathbf{i}|}} \mathop{\mathbb{E}}_{T_{\mathbf{i}^c} \sim \mathcal{D}^{|\mathbf{i}^c|}} e^{|\mathbf{i}^c| \Delta\left(\widehat{\mathcal{L}}_{T_{\mathbf{i}^c}}(\mathcal{R}(T_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(T_{\mathbf{i}}, \sigma))\right)}.$$

Nouvelle borne

Théorème

Pour toute distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, pour toute famille d'ensembles de messages $\{M(\mathbf{i}), | \, \mathbf{i} \in \mathcal{P}(n)\}$, pour toute fonction de reconstruction déterministe $\mathcal{R}$ qui produit des prédicteurs compressés $h \in \mathcal{H}$, pour toute perte $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, pour toute fonction de comparaison $\Delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ et pour tout $\delta \in (0, 1]$, avec une probabilité d'au moins $1 - \delta$ sur le tirage de $S \sim \mathcal{D}^n$, nous avons

$$\forall \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) :$$

$$\Delta\Big(\widehat{\mathcal{L}}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma))\Big) \leq \frac{1}{|\,\mathbf{i}^c\,|} \log\left(\frac{1}{P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})}(\sigma)} \frac{\mathcal{E}_{\Delta}(\mathbf{i}, \sigma)}{\delta}\right)$$

avec

$$\mathcal{E}_{\Delta}(\mathbf{i}, \sigma) = \mathop{\mathbb{E}}_{T_{\mathbf{i}} \sim \mathcal{D}^{|\mathbf{i}|}} \mathop{\mathbb{E}}_{T_{\mathbf{i}^c} \sim \mathcal{D}^{|\mathbf{i}^c|}} e^{|\mathbf{i}^c| \Delta\left(\widehat{\mathcal{L}}_{T_{\mathbf{i}^c}}(\mathcal{R}(T_{\mathbf{i}}, \sigma)), \mathcal{L}_{\mathcal{D}}(\mathcal{R}(T_{\mathbf{i}}, \sigma))\right)}.$$

Nouveau corollaire

Corollaire

Dans le cadre du théorème précédent, avec une probabilité d'au moins $1-\delta$ *sur le tirage de* $S \sim \mathcal{D}^n$*, nous avons*

$$\forall \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) :$$

$$\mathcal{L}_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)) \leq \widehat{\mathcal{L}}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)) + \sqrt{\frac{1}{2|\,\mathbf{i}^c\,|} \log\left(\frac{1}{P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})}(\sigma)} \frac{2\sqrt{|\,\mathbf{i}^c\,|}}{\delta}\right)}.$$

Expériences

Binary MNIST

MNIST

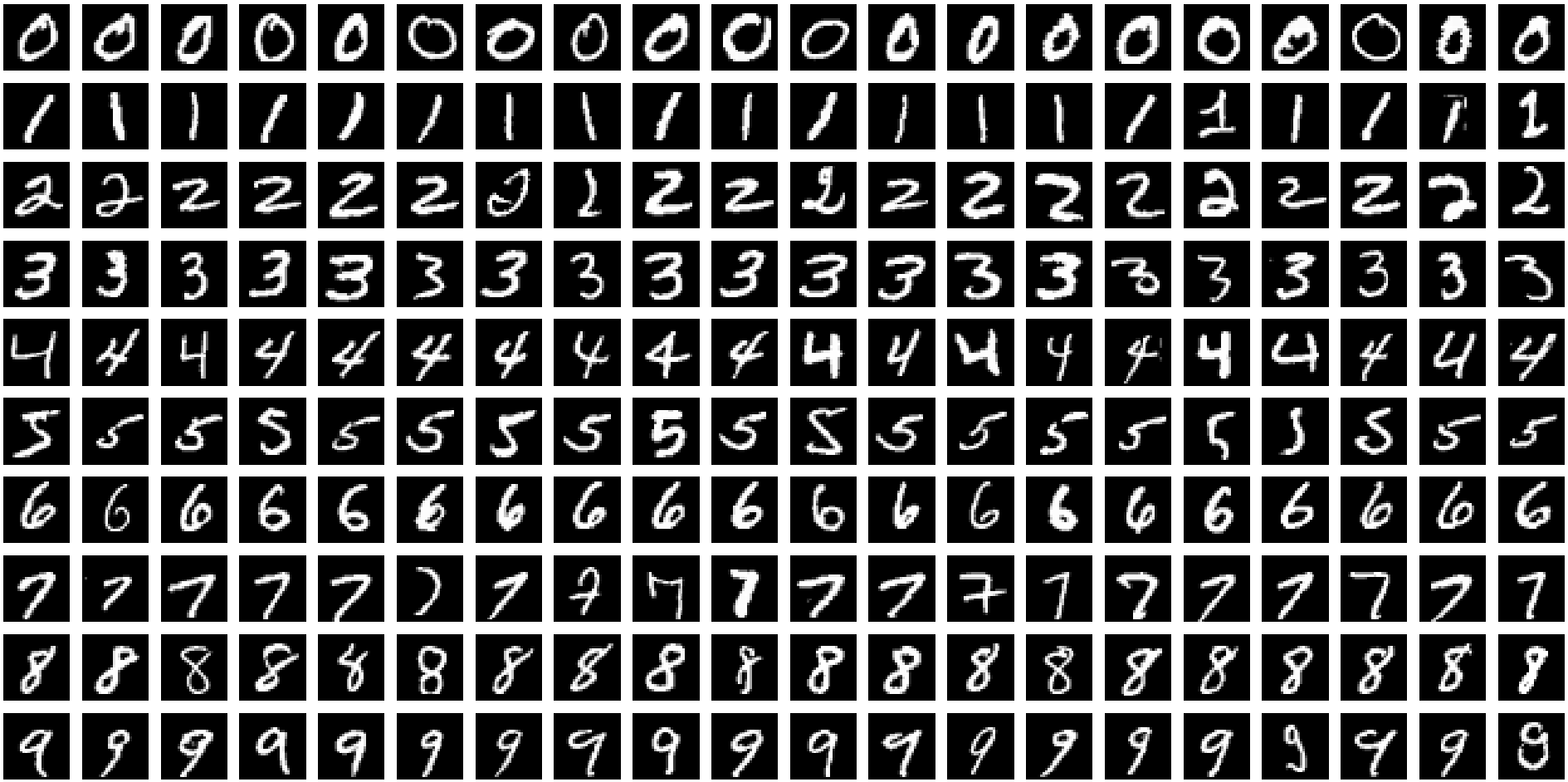


Figure 7: By Suvanjanprasai - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=156115980

Binary MNIST

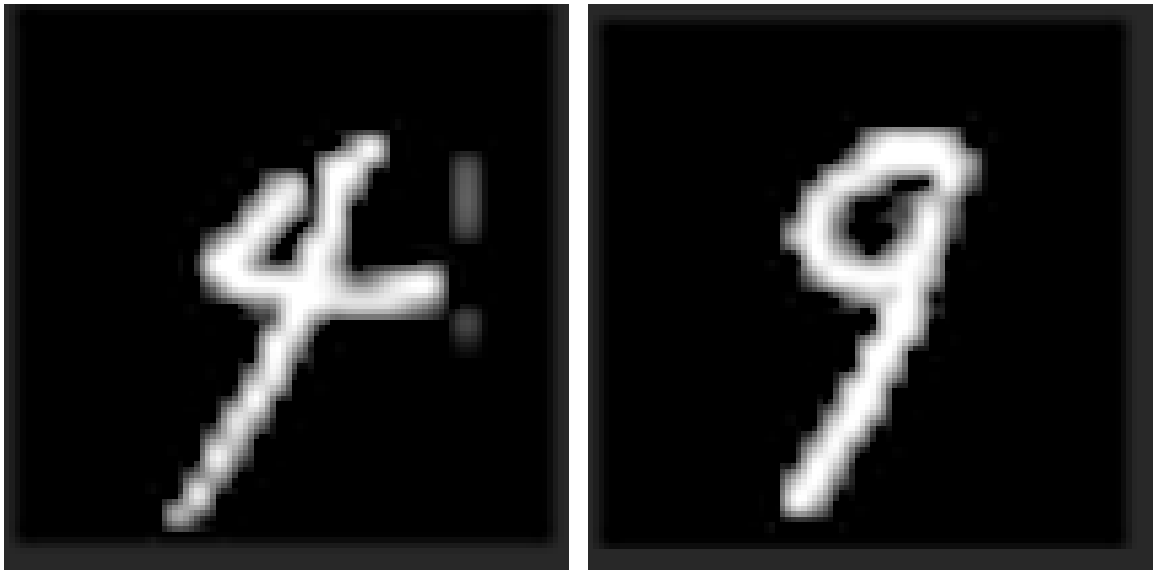


Figure 8: By Suvanjanprasai - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=156115980

Binary MNIST

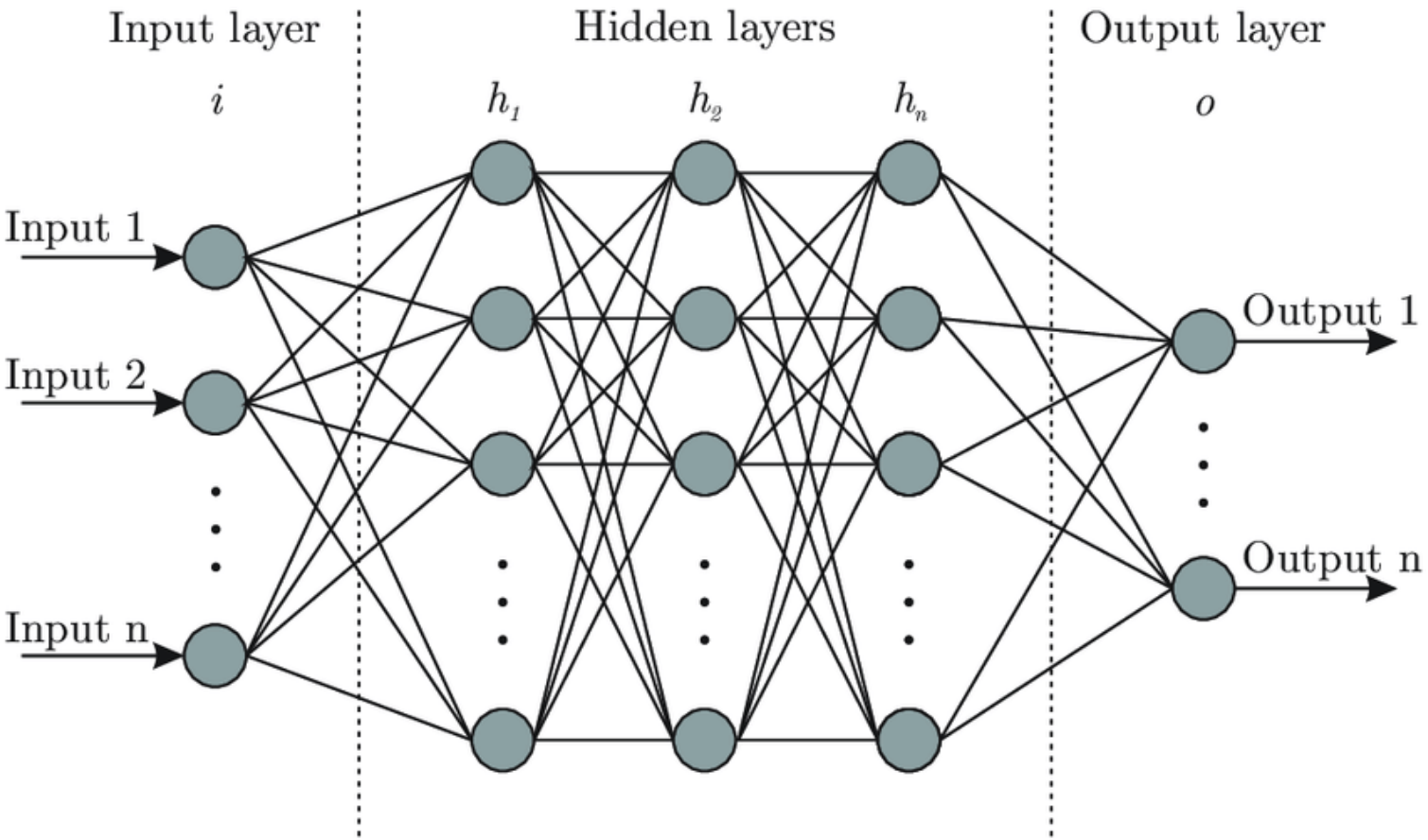


Figure 9: Réseau de neurones (image tirée de [1])

Pick-To-Learn [5]

Pour entraîner notre modèle, au lieu de l'entraîner sur l'entièreté du jeu de données, on utilise Pick-To-Learn. Pick-To-Learn permet de faire la sélection des données importantes.

Algorithme 1 : Pick-To-Learn (P2L)

Initialize : $S_\mathbf{i} = \emptyset$
Initialize : $h_\mathbf{i} = h_0$
Initialize : $(\overline{\boldsymbol{x}}, \overline{y}) = \arg\max_{(\boldsymbol{x},y)\in S} \ell^{\text{x-e}}(h_0, \boldsymbol{x}, y)$
while $-\ln(0.5) \leq \ell^{x\text{-}e}(h_\mathbf{i}, \overline{\boldsymbol{x}}, \overline{y})$ **do**
    $S_\mathbf{i} \leftarrow S_\mathbf{i} \cup \{(\overline{\boldsymbol{x}}, \overline{y})\}$
    $h_\mathbf{i} \leftarrow Update(S_\mathbf{i})$
    $(\overline{\boldsymbol{x}}, \overline{y}) \leftarrow \arg\max_{(\boldsymbol{x},y)\in S_\mathbf{i}^c} \ell^{\text{x-e}}(h_\mathbf{i}, \boldsymbol{x}, y)$
end
return $h_\mathbf{i}$

Binary MNIST

Table 1: Results for the MLPs trained using P2L on the binary MNIST problems. The results displayed obtained the tightest P2L bound. All metrics presented are in percents (%).

Dataset	Test error	kl bound	Binomial bound	$\|\mathbf{i}\|/n$	Baseline test error
MNIST08	0.40±0.08	6.56±0.30	6.51±0.30	1.21±0.07	0.34±0.07
MNIST17	0.47±0.17	4.93±0.27	4.89±0.27	0.85±0.06	0.33±0.09
MNIST23	0.58±0.12	12.21±0.29	12.17±0.29	2.73±0.08	0.36±0.14
MNIST49	1.04±0.10	14.41±0.05	14.37±0.05	3.40±0.02	0.96±0.15
MNIST56	0.65±0.05	10.35±0.31	10.30±0.31	2.18±0.09	0.59±0.01

Forêts de régression

Jeux de données

On choisit 5 jeux de données de régression. On entraîne des forêts de régressions avec Pick-To-Learn.

Puisque c'est de la régression, la perte n'est pas bornée. On suppose donc que la perte est sous-Gaussienne et on utilise une nouvelle version du théorème qui supporte cette assomption.

Nouveau corollaire pour la régression

Corollaire

Dans le cadre du théorème précédent, pour tout $\lambda > 0$, pour une perte ς^2-sous-Gaussienne $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, avec une probabilité d'au moins $1 - \delta$ sur le tirage de $S \sim \mathcal{D}^n$, nous avons

$$\forall \mathbf{i} \in \mathcal{P}(n), \sigma \in M(\mathbf{i}) : \mathcal{L}_{\mathcal{D}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)) \leq \widehat{\mathcal{L}}_{S_{\mathbf{i}^c}}(\mathcal{R}(S_{\mathbf{i}}, \sigma)) + \frac{\lambda \varsigma^2}{2} + \frac{1}{\lambda(n - |\mathbf{i}|)} \log\left(\frac{1}{P_{\mathcal{P}(n)}(\mathbf{i}) P_{M(\mathbf{i})(\sigma)}} \frac{1}{\delta}\right).$$

Forêts de régression

Table 2: Results for the decision forests trained using P2L. We report the RMSE achieved by the models and the generalization bounds on the RMSE. The ratio $|\mathbf{i}|/n$ is presented in percents (%).

Dataset	Train loss	Test loss	Linear bound	$\ell^{\max}$	$\lvert\mathbf{i}\rvert/n$	Baseline test loss
Powerplant	6.11±0.89	6.31±0.95	15.92±0.47	90.6	0.52±0.17	3.59±0.13
Infrared	0.27±0.03	0.30±0.03	1.16±0.08	4.26	2.30±0.60	0.23±0.01
Airfoil	3.67±0.16	3.90±0.18	14.25±0.42	45.13	3.20±0.49	2.10±0.15
Parkinson	7.79±0.33	7.84±0.27	12.02±0.23	41.37	0.42±0.11	2.23±0.16
Concrete	8.18±0.91	8.48±1.41	32.49±1.52	90.63	3.83±0.84	4.70±0.36

Amazon Polarity

Amazon polarity

Jeu de données de commentaires sur les objets vendus par Amazon. Les commentaires avec 4 ou 5 étoiles sont considérés comme positif et les commentaires avec 1 ou 2 étoiles sont considérés comme négatif.

Amazon Polarity

Faisons maintenant un petit jeu! Essayez de deviner lequel des deux commentaire est positif, et lequel est négatif.

« got this for my daughter in NC, she is now making prefect bread. Wish she lived closer to make me some »

« It is very small compared to what i thought it would be, but even so it didn't work when we got it »

Amazon Polarity

Faisons maintenant un petit jeu! Essayez de deviner lequel des deux commentaire est positif, et lequel est négatif.

« got this for my daughter in NC, she is now making prefect bread. Wish she lived closer to make me some » Positif!

« It is very small compared to what i thought it would be, but even so it didn't work when we got it » Négatif!

Amazon Polarity

Recommençons maintenant, mais avec l'entrée qui est fournie au réseau de neurones.

tensor([2009, 2003, 2200, 2235, 4102, 2000, 2054, 1045, 2245, 2009, 2052, 2022, 1010, 2021, 2130, 2061, 2009, 2134, 1005, 1056, 2147, 2043, 2057, 2288, 2009])

tensor([2288, 2023, 2005, 2026, 2684, 1999, 13316, 1010, 2016, 2003, 2085, 2437, 19402, 7852, 1012, 4299, 2016, 2973, 3553, 2000, 2191, 2033, 2070])

Amazon Polarity

Recommençons maintenant, mais avec l'entrée qui est fournie au réseau de neurones.

tensor([2009, 2003, 2200, 2235, 4102, 2000, 2054, 1045, 2245, 2009, 2052, 2022, 1010, 2021, 2130, 2061, 2009, 2134, 1005, 1056, 2147, 2043, 2057, 2288, 2009]) Négatif!

tensor([2288, 2023, 2005, 2026, 2684, 1999, 13316, 1010, 2016, 2003, 2085, 2437, 19402, 7852, 1012, 4299, 2016, 2973, 3553, 2000, 2191, 2033, 2070]) Positif!

Tokenization

Pour obtenir ce genre d'entrée, on utilise un dictionnaire qui contient l'entièreté du vocabulaire du modèle ainsi que leurs indexs. Dans ce cas-ci, c'est le vocabulaire de DistilBERT.

```
Sentence:  Mon nom est Mathieu. Je suis étudiant au doctorat.
Tokens: ['mon', 'no', '##m', 'est', 'math', '##ieu', '.', 'je', 'sui', '##s', 'et', '##udi', '##ant', 'au', 'doctor', '##at'
Token IDs: [12256, 2053, 2213, 9765, 8785, 17301, 1012, 15333, 24086, 2015, 3802, 21041, 4630, 8740, 3460, 4017, 1012]

Sentence:  My name is Mathieu. I am a PhD student.
Tokens: ['my', 'name', 'is', 'math', '##ieu', '.', 'i', 'am', 'a', 'phd', 'student', '.']
Token IDs: [2026, 2171, 2003, 8785, 17301, 1012, 1045, 2572, 1037, 8065, 3076, 1012]
```

Figure 10: Tokenization du français versus la tokenization de l'anglais.

Tokenization

Pour obtenir ce genre d'entrée, on utilise un dictionnaire qui contient l'entièreté du vocabulaire du modèle ainsi que leurs indexs. Dans ce cas-ci, c'est le vocabulaire de DistilBERT.

```
Sentence:  Mon nom est Mathieu. Je suis étudiant au doctorat.
Tokens: ['mon', 'no', '##m', 'est', 'math', '##ieu', '.', 'je', 'sui', '##s', 'et', '##udi', '##ant', 'au', 'doctor', '##at'
Token IDs: [12256, 2053, 2213, 9765, 8785, 17301, 1012, 15333, 24086, 2015, 3802, 21041, 4630, 8740, 3460, 4017, 1012]

Sentence:  My name is Mathieu. I am a PhD student.
Tokens: ['my', 'name', 'is', 'math', '##ieu', '.', 'i', 'am', 'a', 'phd', 'student', '.']
Token IDs: [2026, 2171, 2003, 8785, 17301, 1012, 1045, 2572, 1037, 8065, 3076, 1012]
```

Figure 10: Tokenization du français versus la tokenization de l'anglais.

Même si le vocabulaire de DistilBERT ne contient pas les mots français, grâce à une combinaison de tokens, il est capable de représenter les mots.

BERT

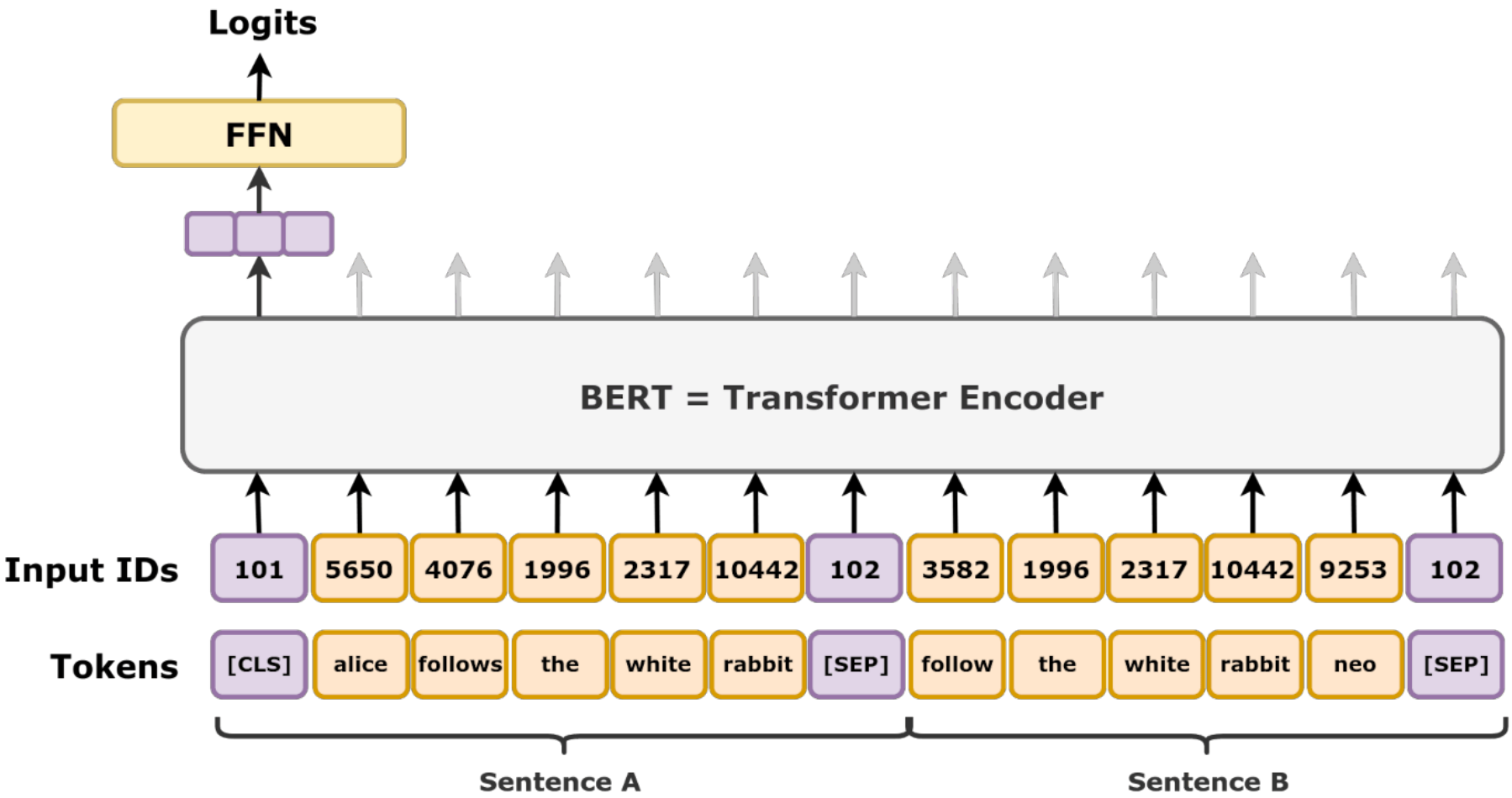
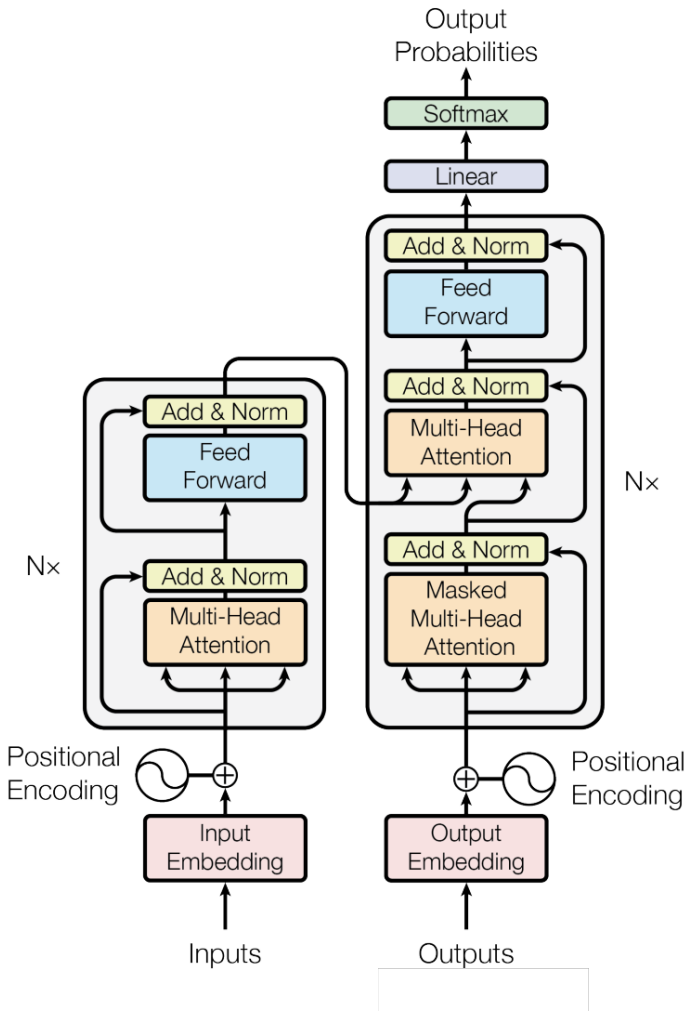


Figure 11: By Daniel Voigt Godoy - https://dvgodoy.github.io/dl-visuals/BERT/, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=151177217

Transformer

BERT

Encoder



GPT

Decoder

Figure 12: Image tirée de https://heidloff.net/article/foundation-models-transformers-bert-and-gpt/

Amazon Polarity

1. Dataset : **Binary classification problems** on Amazon Polarity dataset (we use 10%, 360000 datapoints)
2. Model type : DistilBERT [7] with **66 million parameters**
3. Training algorithm : **Pre-training** on 50% of the dataset, then **Pick-To-Learn** on the other half.

Train method	Train error	Validation error	Test error	kl bound
Pick-To-Learn	4.73±1.09	5.41±1.05	5.60±1.19	13.91±2.73
Baseline	3.11±0.02	4.08±0.04	4.19±0.00	-

Table 3: All metrics present are in percent (%).

Conclusion

Conclusion

En conclusion, on a proposé de nouvelles bornes de compression d'échantillons et nous avons démontré qu'elles peuvent être utilisées pour obtenir des garanties de généralisations *tight*.

Dans le futur, on pourrait s'intéresser à

1. Obtenir des garanties pour des distributions de perte plus complexes (e.g. heavy-tail)
2. S'intéresser au choix des données par les algorithmes de compression

[1] Facundo Bre, Juan Gimenez et Víctor Fachinotti : Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 2017.

[2] Gintare Karolina Dziugaite et Daniel M Roy : Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[3] François Laviolette, Mario Marchand et Mohak Shah : Margin-Sparsity Trade-Off for the Set Covering Machine. *In Machine Learning: ECML 2005*, volume 3720, pages 206–217. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-29243-2 978-3-540-31692-3.

[4] Jean-Samuel Leboeuf, Frédéric LeBlanc et Mario Marchand : Generalization properties of decision trees on real-valued and categorical features. *arXiv preprint arXiv:2210.10781*, 2022.

[5] Dario Paccagnan, Marco Campi et Simone Garatti : The pick-to-learn algorithm: Empowering compression for tight generalization bounds and improved post-training performance. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor et Csaba Szepesvári : Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22(227):1–40, 2021.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond et Thomas Wolf : Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL `https://arxiv.org/abs/1910.01108`.