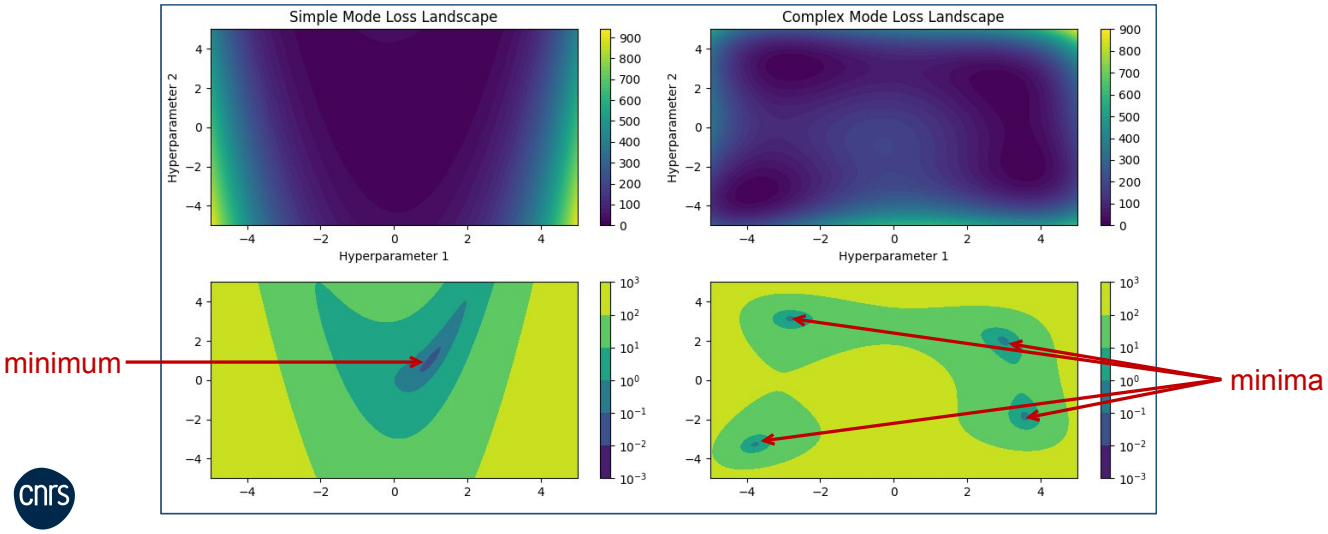
TP5 HPO Takeaways



IDRIS

paramètre <— s’apprend pendant l’entraînement
hyperparamètre <— se fixe <u>avant</u> l’entraînement

HPO = chercher la configuration qui minimise une perte, sans passer par l’essai-erreur à la main.



1. Introduction et paysage de perte (Rosenbrock / Himmelblau)

- Un hyperparamètre se fixe avant l’entraînement, alors qu’un paramètre s’apprend pendant l’entraînement.
- (optimisation d’hyperparamètres) HPO = chercher la configuration qui minimise une perte, sans passer par l’essai-erreur à la main.
  - **Maximiser la performance** : améliorer la métrique qui compte (exactitude, précision, rappel, ou toute autre).
  - **Être efficace** : réduire le temps d’entraînement et les ressources de calcul, sans dégrader les résultats.
  - **Bien généraliser** : (que le modèle fonctionne aussi bien sur des données qu’il n’a jamais vues.**)**
- Deux frameworks dans ce TP : **Optuna** et **Ray Tune**.
- **Optuna** est simple à prendre en main. C’est un excellent outil pour comprendre les bases du HPO, ainsi que ses subtilités.
- **Ray Tune** passe à l’échelle et propose des fonctionnalités avancées. C’est un choix solide pour les cas plus complexes et gourmands en ressources.

La perte dépend des hyperparamètres selon un « paysage ».

Exemple :

Rosenbrock (mode simple) a une vallée étroite avec un seul minimum, en (1, 1).

Himmelblau (mode complex) a quatre minima.

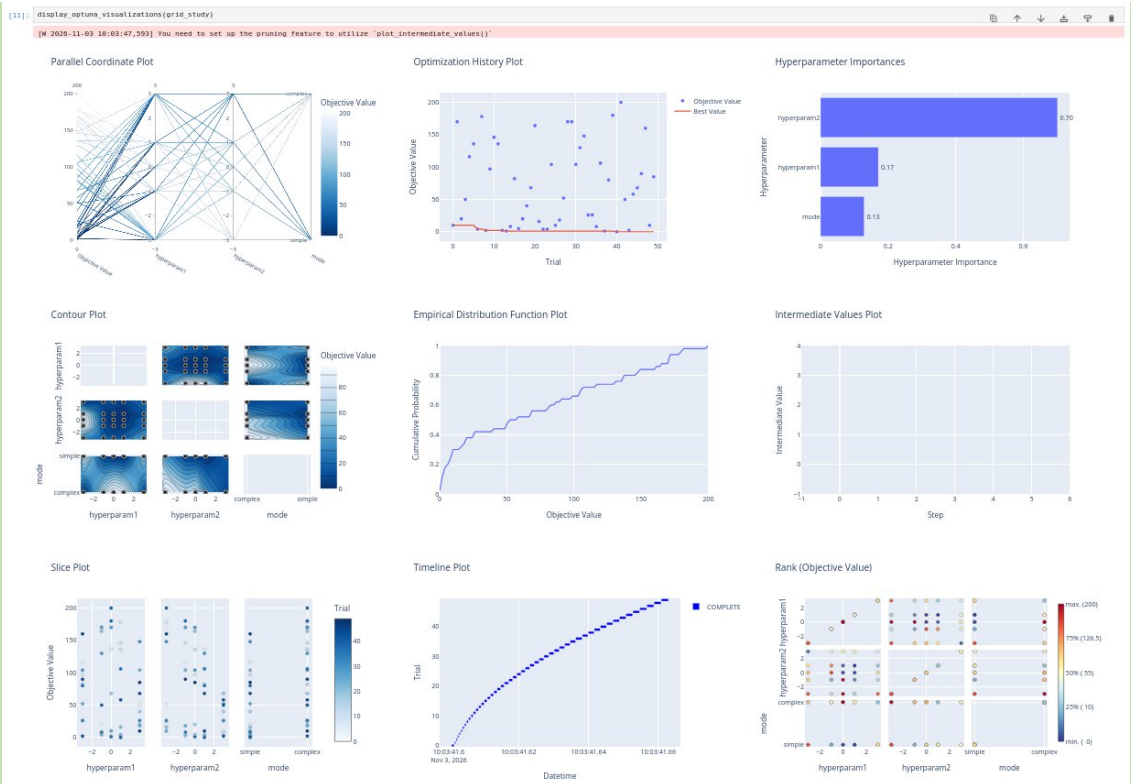
On a vu que l’échelle log fait apparaître le minimum, invisible en échelle linéaire.

Sur un vrai réseau, ce paysage est de grande dimension et on ne peut pas le visualiser.

La fonction objectif

- C’est la **pierre angulaire** d’Optuna.
- Elle évalue la qualité d’un jeu d’hyperparamètres selon une métrique choisie à l’avance, en général la perte ou l’exactitude d’un modèle.

Grid



2. L’objective Optuna

- Le schéma à retenir : objective(trial) → suggest_* → retourne un nombre.

Optuna appelle cette fonction en boucle.

- Le nom du paramètre ('hyperparam1') sert de clé de suivi dans Optuna.

3. Grid vs Random vs TPE

comparaison de 3 stratégies clefs :

- **Grid Search** (force brute), teste tout les points prédéfinits donc peut être très fastidieux
- **Random sampling** : tire les hyperparamètres au hasard, sans a priori. Il sert de **référence**, et peut trouver rapidement de bonnes valeurs quand l’espace est simple.
- **TPE** (optimisation bayésienne) : utilise un modèle probabiliste pour proposer intelligemment les prochains essais à partir des précédents. Il est efficace dans les espaces complexes et de grande dimension.

Observations :

Méthode Commentaire

Grid Ici le point (1, 1) était dans la grille, donc c’est de la chance de conception.

Random Sert de référence. Il explore sans apprendre.

TPE Converge vers la bonne zone en apprenant des essais précédents.

Grid

Contour, Slice et Rank : les points forment un **quadrillage** (colonnes régulières). La grille ne regarde que les valeurs prévues à l’avance.

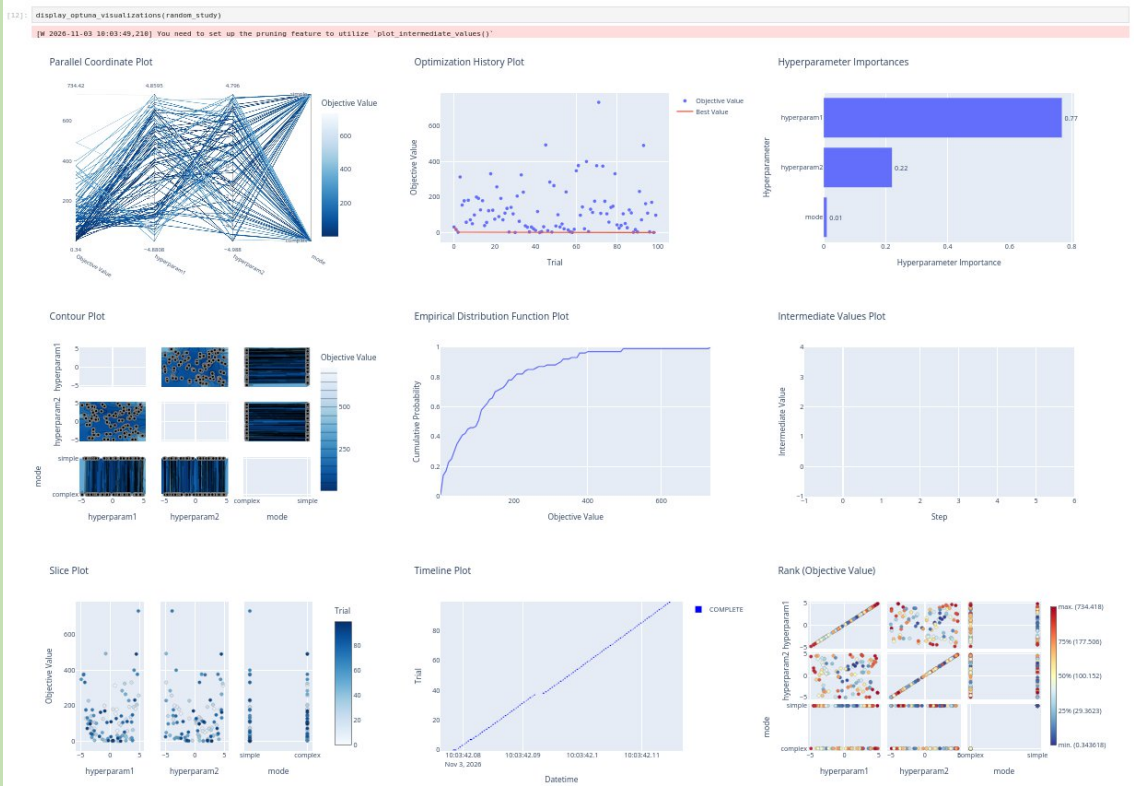
Historique : les points restent dispersés jusqu’à 200 tout du long. La ligne rouge tombe à ~0 dès les premiers essais. Elle a trouvé (1, 1) tôt

- La grille s’est arrêtée à 50 essais, car 5 × 5 × 2 = 50 combinaisons.
- En pratique elle explose avec le nombre d’hyperparamètres, et elle ne trouve que ce qu’on a mis dans la grille.

Random

Contour : points répartis **uniformément** sur tout le carré. Les points hauts (jusqu’à ~730) restent présents jusqu’à la fin, car Random ne tire aucune leçon.

Random



2. L'objective Optuna

- Le schéma à retenir : objective(trial) → suggest_* → retourne un nombre.

Optuna appelle cette fonction en boucle.

- Le nom du paramètre ('hyperparam1') sert de clé de suivi dans Optuna.

3. Grid vs Random vs TPE

comparaison de 3 stratégies clefs :

- **Grid Search** (force brute), teste tout les points prédéfinits donc peut être très fastidieux
- **Random sampling** : tire les hyperparamètres au hasard, sans a priori. Il sert de **référence**, et peut trouver rapidement de bonnes valeurs quand l'espace est simple.
- **TPE** (optimisation bayésienne) : utilise un modèle probabiliste pour proposer intelligemment les prochains essais à partir des précédents. Il est efficace dans les espaces complexes et de grande dimension.

Observations :

Méthode Commentaire

Grid Ici le point (1, 1) était dans la grille, donc c'est de la chance de conception.

Random Sert de référence. Il explore sans apprendre.

TPE Converge vers la bonne zone en apprenant des essais précédents.

Grid

Contour, Slice et Rank : les points forment un **quadrillage** (colonnes régulières). La grille ne regarde que les valeurs prévues à l'avance.

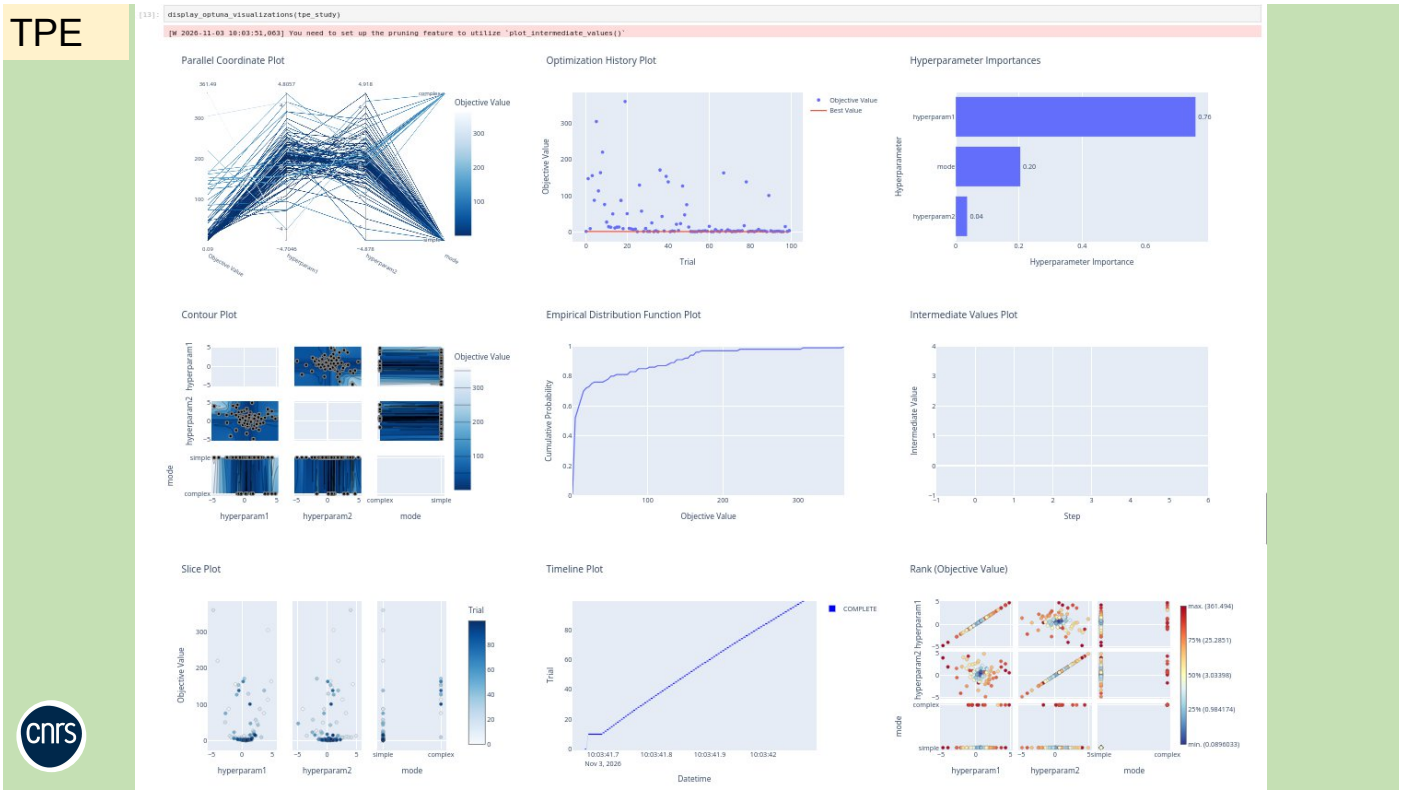
Historique : les points restent dispersés jusqu'à 200 tout du long. La ligne rouge tombe à ~0 dès les premiers essais. Elle a trouvé (1, 1) tôt

- La grille s'est arrêtée à 50 essais, car 5 × 5 × 2 = 50 combinaisons.
- En pratique elle explose avec le nombre d'hyperparamètres, et elle ne trouve que ce qu'on a mis dans la grille.

Random

Contour : points répartis **uniformément** sur tout le carré. Les points hauts (jusqu'à ~730) restent présents jusqu'à la fin, car Random ne tire aucune leçon.

TPE



2. L’objective Optuna

- Le schéma à retenir : objective(trial) → suggest_* → retourne un nombre.

Optuna appelle cette fonction en boucle.

- Le nom du paramètre ('hyperparam1') sert de clé de suivi dans Optuna.

3. Grid vs Random vs TPE

comparaison de 3 stratégies clefs :

- **Grid Search** (force brute), teste tout les points prédéfinits donc peut être très fastidieux
- **Random sampling** : tire les hyperparamètres au hasard, sans a priori. Il sert de **référence**, et peut trouver rapidement de bonnes valeurs quand l’espace est simple.
- **TPE** (optimisation bayésienne) : utilise un modèle probabiliste pour proposer intelligemment les prochains essais à partir des précédents. Il est efficace dans les espaces complexes et de grande dimension.

Observations :

Méthode Commentaire

Grid Ici le point (1, 1) était dans la grille, donc c’est de la chance de conception.

Random Sert de référence. Il explore sans apprendre.

TPE Converge vers la bonne zone en apprenant des essais précédents.

Grid

Contour, Slice et Rank : les points forment un **quadrillage** (colonnes régulières). La grille ne regarde que les valeurs prévues à l’avance.

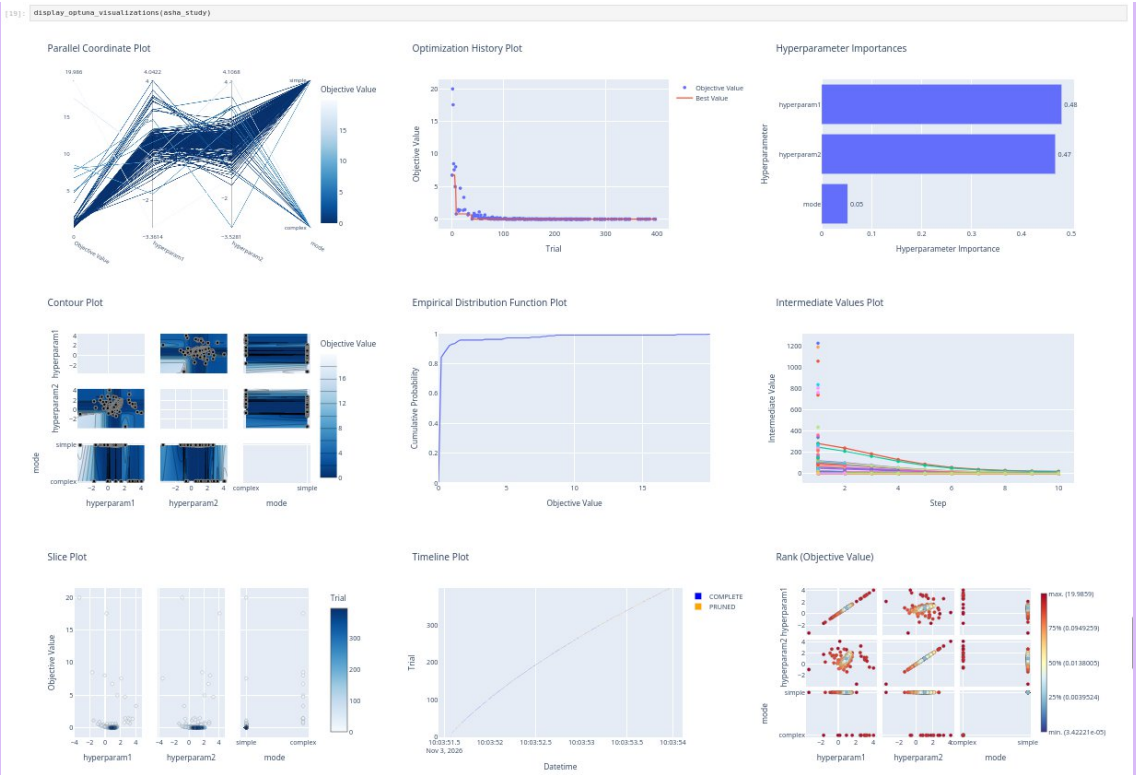
Historique : les points restent dispersés jusqu’à 200 tout du long. La ligne rouge tombe à ~0 dès les premiers essais. Elle a trouvé (1, 1) tôt

- La grille s’est arrêtée à 50 essais, car 5 × 5 × 2 = 50 combinaisons.
- En pratique elle explose avec le nombre d’hyperparamètres, et elle ne trouve que ce qu’on a mis dans la grille.

Random

Contour : points répartis **uniformément** sur tout le carré. Les points hauts (jusqu’à ~730) restent présents jusqu’à la fin, car Random ne tire aucune leçon.

Asha



4. Pruning (ASHA / Hyperband) (objet)

- Idée centrale : on ne termine pas les essais qui s'annoncent mauvais, et on économise du calcul.
- Prérequis —> trial.report() puis trial.should_prune() dans l'objective.
- Toute fonction objective fait trois choses dans cet ordre : elle **reçoit** un trial, elle **demande** à ce trial des valeurs d'hyperparamètres, puis elle **renvoie** un nombre qui mesure la performance.
- Il faut donc une métrique intermédiaire, par exemple la perte à chaque epoch.
- **SuccessiveHalvingPruner** : implémentation d'ASHA. Il évalue les essais à intervalles réguliers et **élimine la moitié la moins prometteuse**, pour concentrer les ressources sur les essais les plus prometteurs.
- **HyperbandPruner** : il évalue les essais à **différents paliers d'étapes** et élimine à chaque fois les moins prometteurs, pour concentrer les ressources sur ceux qui ont le plus de potentiel.

Observations :

Les courbes qui s'arrêtent avant l'étape 10 sont les essais élagués. C'est le pruning visible.

- ASHA : beaucoup de points isolés à l'étape 1 (valeurs de 400 à 1 200), donc arrêtés quasi immédiatement.
- Hyperband : on voit des courbes qui s'arrêtent à l'étape 2, d'autres à l'étape 6. Ce sont les « barrages » successifs. Les essais les plus hauts (jusqu'à 1 500) sont éliminés tôt, ceux qui sont proches de 0 vont jusqu'à 10.

Optimization History

La meilleure perte tombe à ≈ 0 en **une dizaine d'essais**, puis reste là. Tous les points ensuite sont collés à 0.

Ce qu'on observe, c'est le sampler TPE par défaut qui converge vite, combiné au pruner.

Le pruner n'améliore pas la qualité du résultat, il réduit le coût pour y arriver !

Rank, Slice, Contour

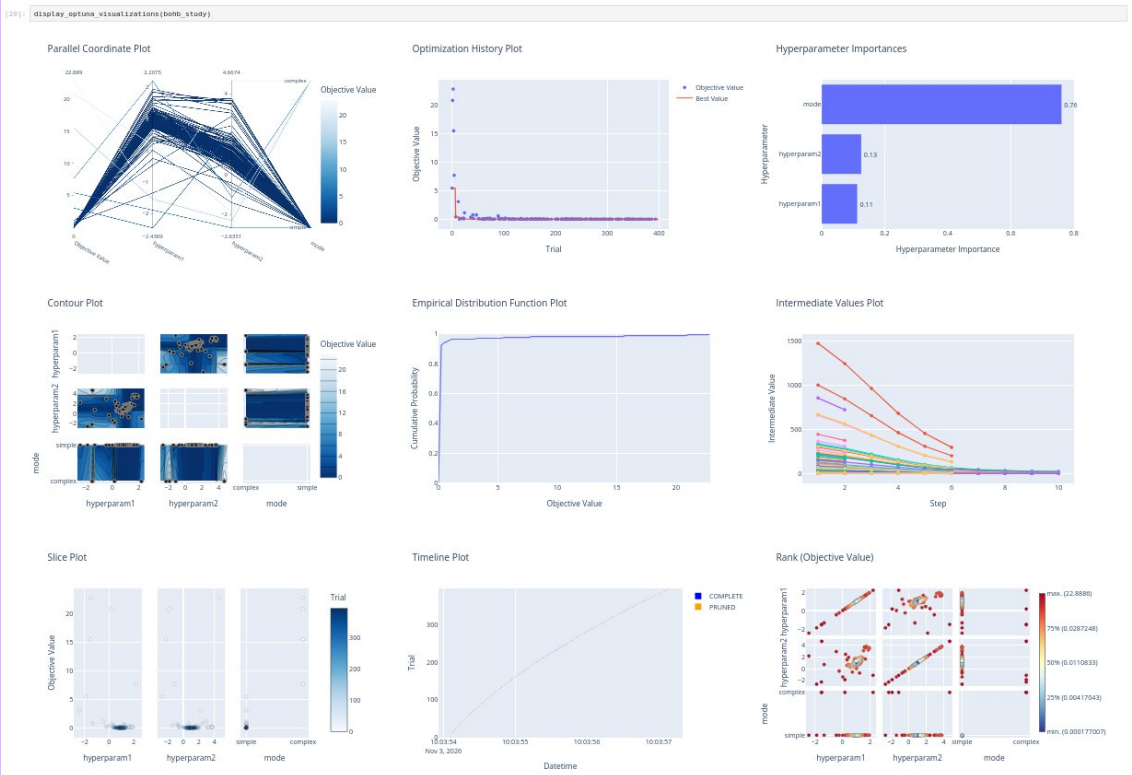
Tous les points se regroupent autour de (1, 1) en mode simple. C'est le bon minimum, atteint dans les deux études (meilleures pertes de l'ordre de 3e-5 pour ASHA et 2e-4 pour Hyperband).

Timeline

La légende contient maintenant **COMPLETE** et **PRUNED** (en orange). À cette échelle, les barres sont trop fines pour distinguer les deux. Zoomer avec la souris permet de les voir.

EDF

Bohb



4. Pruning (ASHA / Hyperband) (objet)

- Idée centrale : on ne termine pas les essais qui s'annoncent mauvais, et on économise du calcul.
- Prérequis —> trial.report() puis trial.should_prune() dans l'objective.
- Toute fonction objective fait trois choses dans cet ordre : elle **reçoit** un trial, elle **demande** à ce trial des valeurs d'hyperparamètres, puis elle **renvoie** un nombre qui mesure la performance.
- Il faut donc une métrique intermédiaire, par exemple la perte à chaque epoch.
- **SuccessiveHalvingPruner** : implémentation d'ASHA. Il évalue les essais à intervalles réguliers et **élimine la moitié la moins prometteuse**, pour concentrer les ressources sur les essais les plus prometteurs.
- **HyperbandPruner** : il évalue les essais à **différents paliers d'étapes** et élimine à chaque fois les moins prometteurs, pour concentrer les ressources sur ceux qui ont le plus de potentiel.

Observations :

Les courbes qui s'arrêtent avant l'étape 10 sont les essais élagués. C'est le pruning visible.

- ASHA : beaucoup de points isolés à l'étape 1 (valeurs de 400 à 1 200), donc arrêtés quasi immédiatement.
- Hyperband : on voit des courbes qui s'arrêtent à l'étape 2, d'autres à l'étape 6. Ce sont les « barrages » successifs. Les essais les plus hauts (jusqu'à 1 500) sont éliminés tôt, ceux qui sont proches de 0 vont jusqu'à 10.

Optimization History

La meilleure perte tombe à ≈ 0 en **une dizaine d'essais**, puis reste là. Tous les points ensuite sont collés à 0.

Ce qu'on observe, c'est le sampler TPE par défaut qui converge vite, combiné au pruner.

Le pruner n'améliore pas la qualité du résultat, il réduit le coût pour y arriver !

Rank, Slice, Contour

Tous les points se regroupent autour de (1, 1) en mode simple. C'est le bon minimum, atteint dans les deux études (meilleures pertes de l'ordre de 3e-5 pour ASHA et 2e-4 pour Hyperband).

Timeline

La légende contient maintenant **COMPLETE** et **PRUNED** (en orange). À cette échelle, les barres sont trop fines pour distinguer les deux. Zoomer avec la souris permet de les voir.

EDF