c2st: Classifier Two-Sample Testing for comparing high-dimensional point sets

Steve Schmerler, Peter Steinbach
Helmholtz-Zentrum Dresden-Rossendorf (HZDR)

Introduction

Problem: Test whether two sets of points are samples from the same D-dimensional probability distribution without having access to the PDF. Given two point sets $X \in \mathbb{R}^{N_X \times D}$, $Y \in \mathbb{R}^{N_Y \times D}$, train a binary classifier on inputs which are the concatenation (X, Y) and targets $(x, y), x = 0 \in \mathbb{R}^{N_X}, y = 1 \in \mathbb{R}^{N_Y}$. c2st returns a score between 0.5 and 1. A value close to 0.5 means that the classifier is not better than random guessing, i.e. X and Y are likely from the same distribution. A value close to 1 means the classifier was able to separate X and Y, so they are probably samples from different distributions.

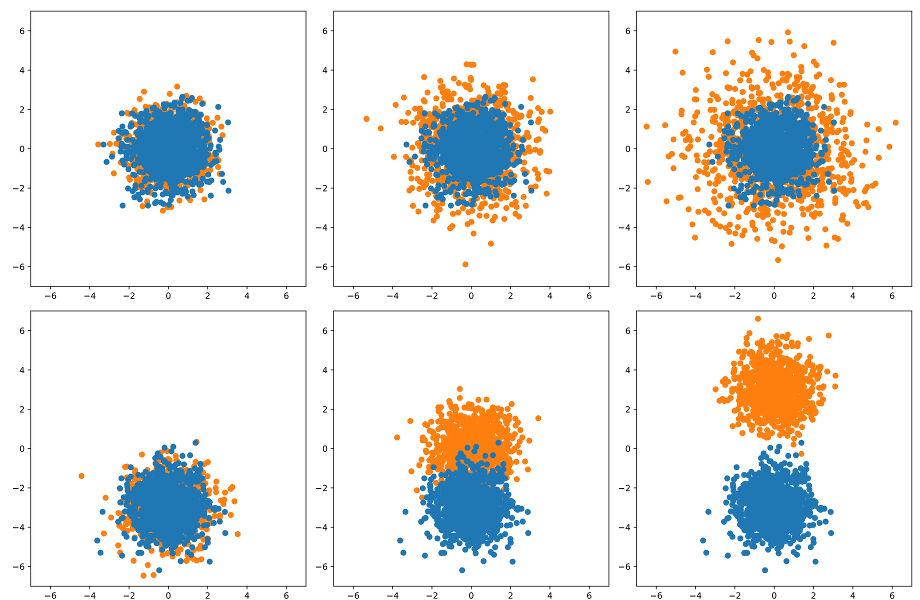

Figure 1: Samples from two $2D$ distributions $X \sim \mathcal{N}(\mu_X, \Sigma_X)$ and $Y \sim \mathcal{N}(\mu_Y, \Sigma_Y)$ with increasing difference in covariance $\Sigma = \sigma\, I$ (top, "scale shift") and mean μ (bottom, "location shift").

Experiments

- 1 c2st run: 5-fold CV (train classifier 5 times), uncertainty of c2st score = sample standard deviation of CV scores
- in total > 5000 runs covering several (mostly sklearn) classifiers and their parameters: `rf` = RandomForestClassifier, `knn` = KNeighborsClassifier, `mlp` = MLPClassifier, `xgb` = XGBClassifier (xgboost), `skbmlp` = Skorch `mlp` variant (sklearn API, PyTorch backend)
- Synthetic data $\mathcal{N}(\mu, \sigma\, I)$, unless stated otherwise $D = 10, N_X = 5000, N_Y = 2500$, balanced accuracy scoring
- if not varied: $\sigma_X = \sigma_Y = 1$, $\mu_X = \mu_Y = 0$; location shift: vary μ_Y, scale shift: vary σ_Y
- we use classifier default parameters when not stated otherwise, except `mlp`: `layers`$=(150, 150)$, adam solver with early stopping; `rf` + `xgb`: `n_estimators=100`

Key observations

- use large sample sizes N
- scale shift is the harder problem where we see failures with some classifiers
- watch out: `knn`, `mlp`; solid: `rf`, `xgb`; use at least 2 classifiers and compare, c2st API: `c2st(X, Y, clf=MyClassifier, ...)`

Classifier parameters

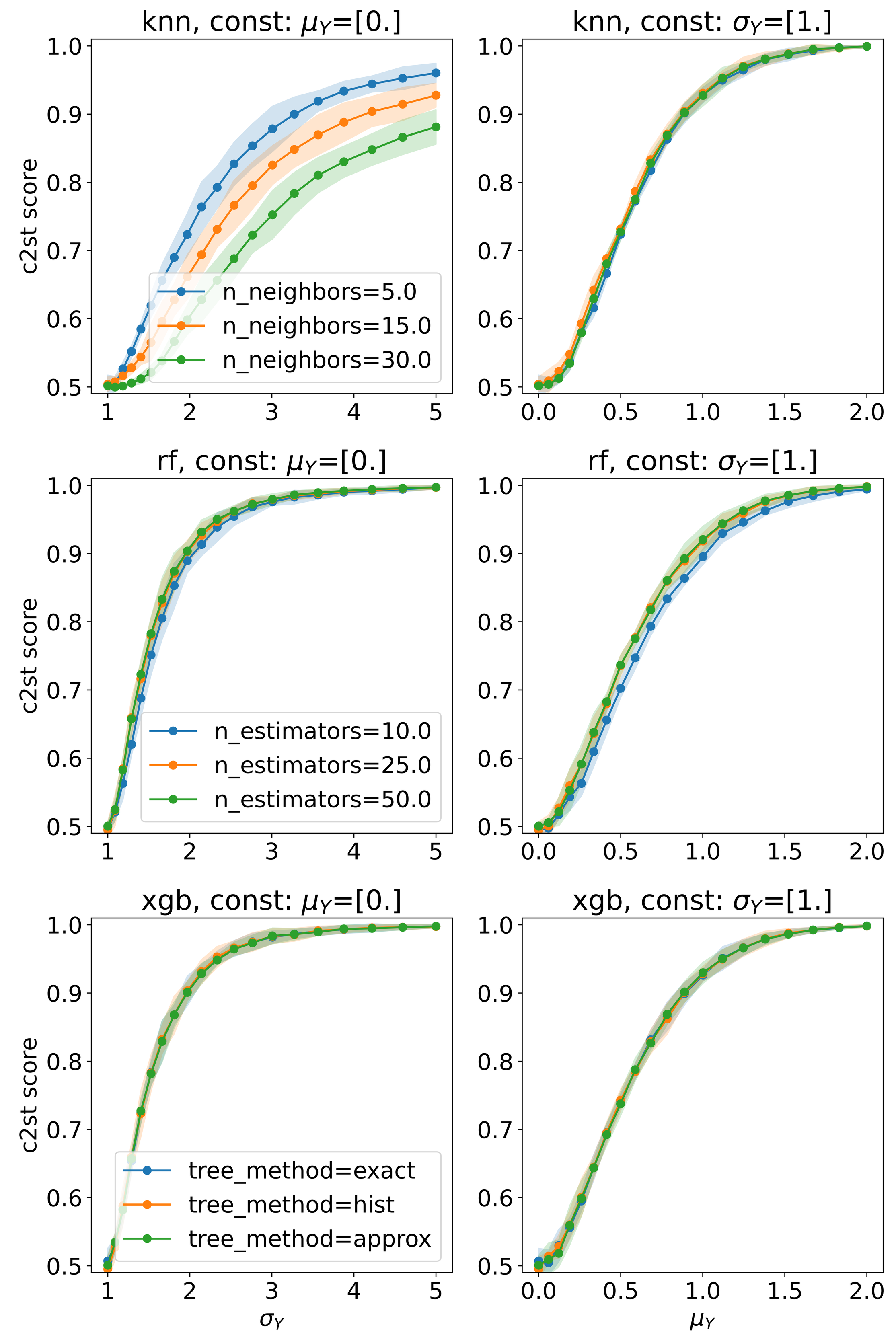

Figure 2: Selection of parameter scans for `knn`, `rf`, `xgb`.

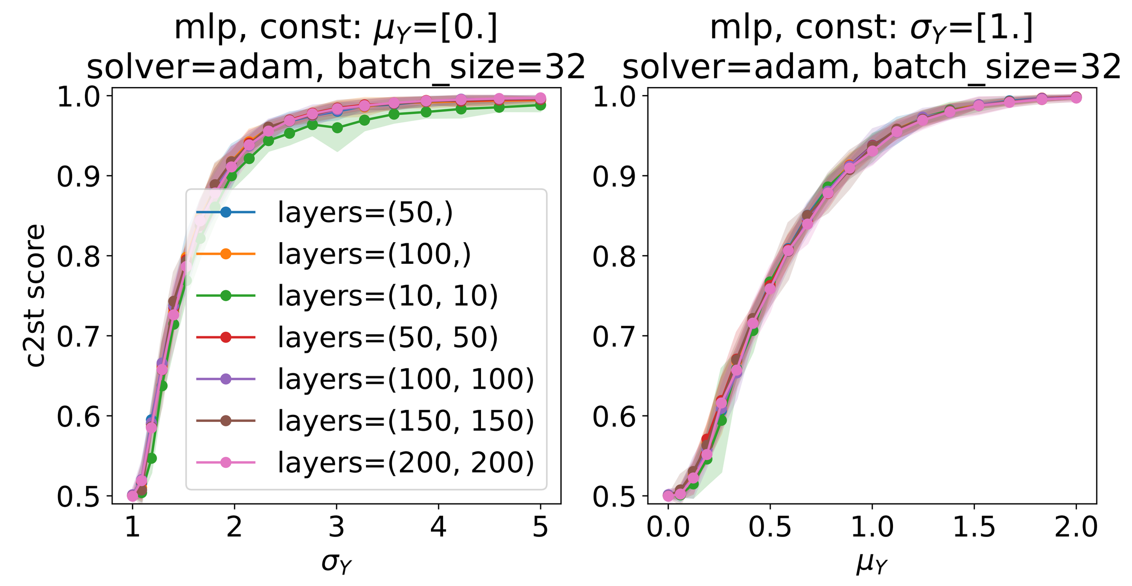

Figure 3: Scan size of hidden `mlp` layers. We use ReLU, adam, early stopping based on 10% hold-out set validation accuracy during training.

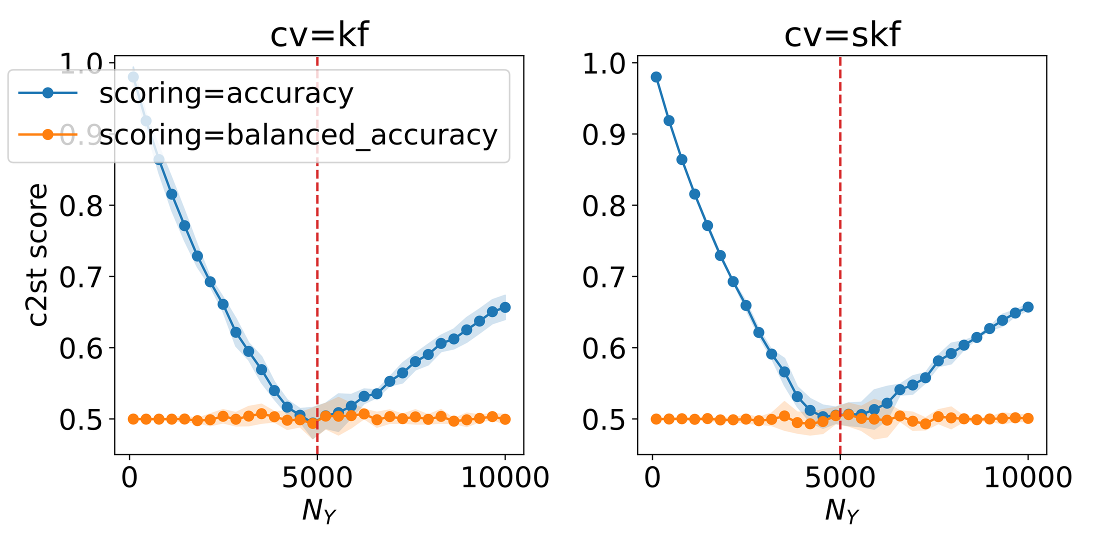

Figure 4: Importance of using balanced accuracy. Use same μ and σ, $N_X = 5000$, vary N_Y, expect score of 0.5 (`rf` classifier, KFold (`kf`) vs. StratifiedKFold (`skf`) cross-validation splitting).

Dimensionality D

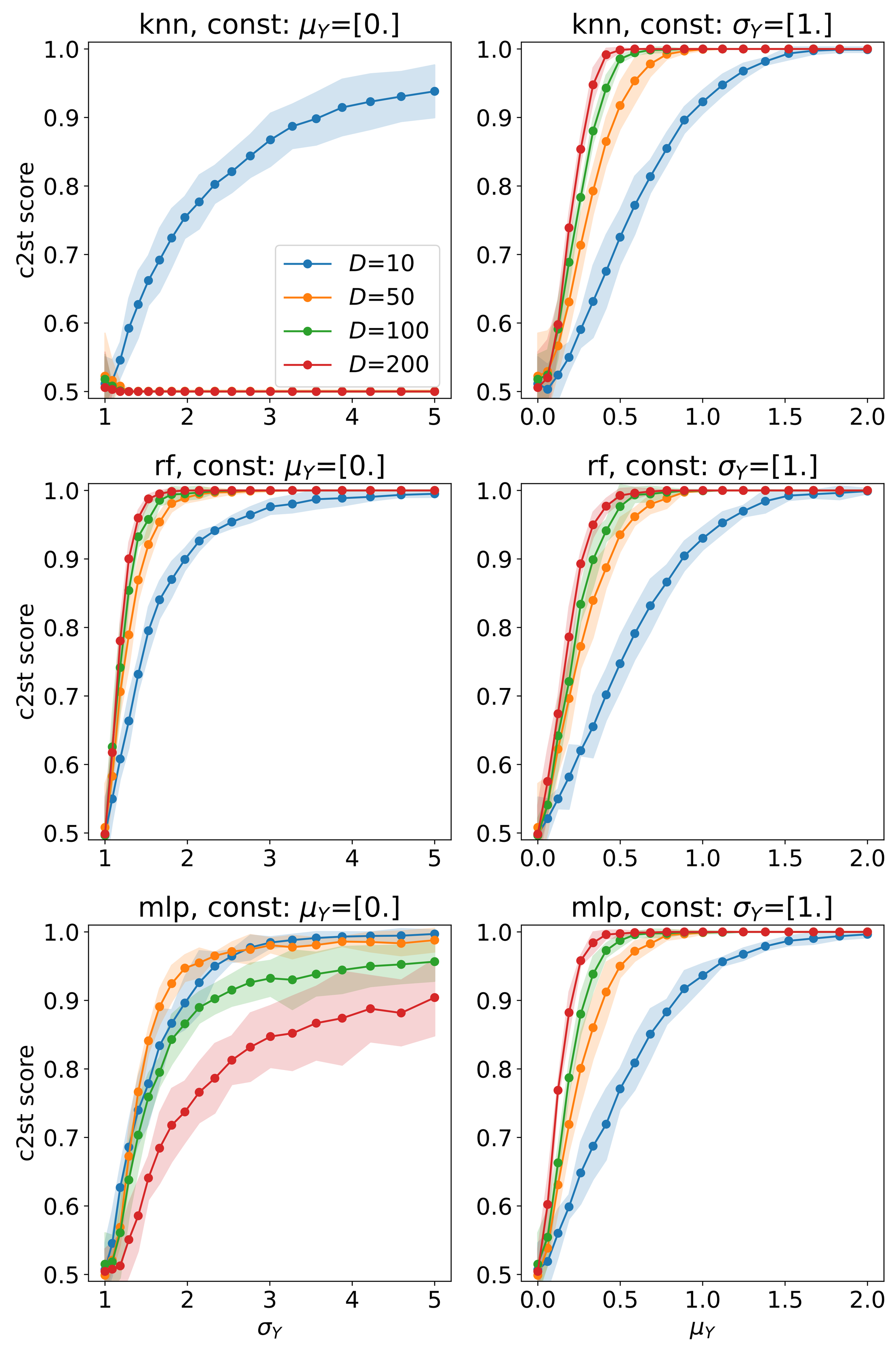

Figure 5: Effect of dimension D at low $N = 1000$. The `knn` failure mode persists when increasing N, while `mlp` will recover `rf`-like behavior.

Sample size N

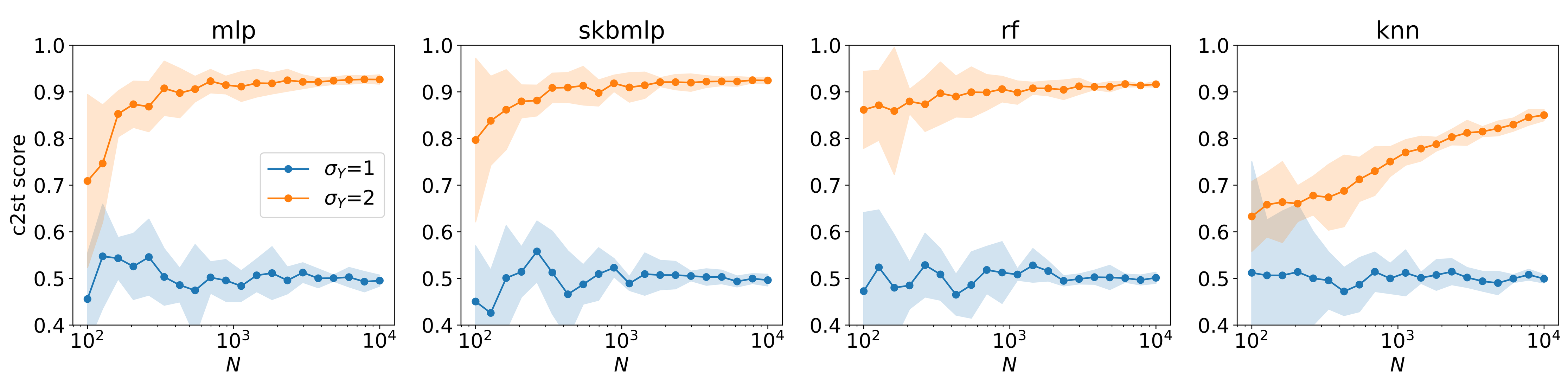

Figure 6: Increase $N_X = N_Y$ in two σ_Y cases. Easy problem: $\sigma_X = \sigma_Y = 1$, expect score 0.5. Hard problem scale shift ($\sigma_Y = 2$): converged c2st score is ≈ 0.93. Except for `knn`, all classifiers provide converged scores for $N > 10^3$ and decreasing uncertainty. `knn` is not converged yet, see also fig. 2 scale shift, where $N_X = 5000$. Convergence behavior will also depend on D (here $D = 10$, see also fig. 5).

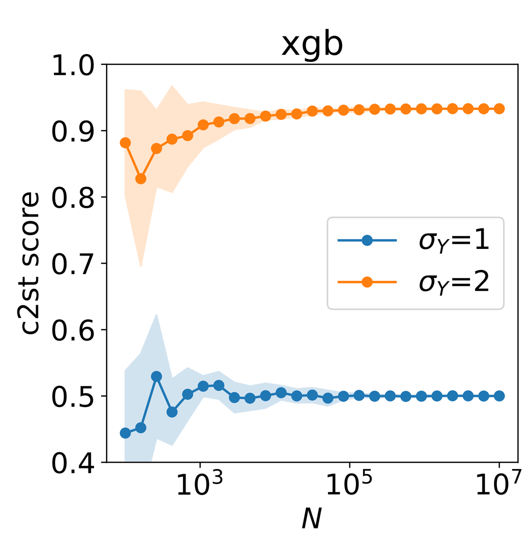

Figure 7: With `xgb` we can handle 10^7 points and more, runtime ≈ 1 min on one GPU (P100 and better). For $N > 10^5$ scores become nearly constant, uncertainty vanishes.

Resources and References

Code: https://github.com/psteinb/c2st, xgboost: https://xgboost.readthedocs.io, Skorch: https://skorch.readthedocs.io, psweep: https://pypi.org/project/psweep (parameter study tooling), D. Lopez-Paz and M. Oquab. "Revisiting Classifier Two-Sample Tests". In: *5th International Conference on Learning Representations, ICLR*. 2017. URL: http://arxiv.org/abs/1610.06545