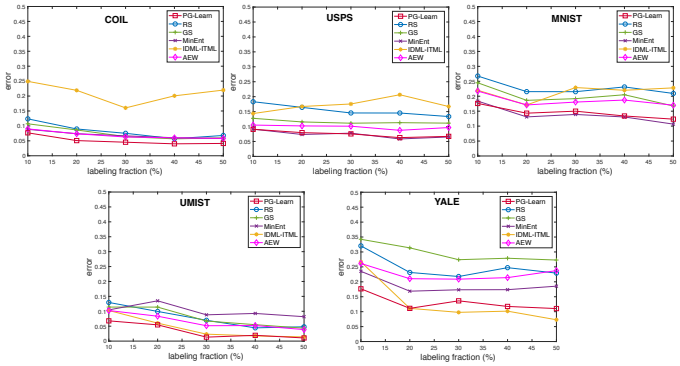

Figure 5: Test error (avg'ed across 3 random samples) as labeled data percentage is increased up to 50%. PG-LEARN performs the best in many cases, and *consistently* ranks in top two among competitors on each dataset and each labeling %.

PG-LEARN significantly outperforms *all* competing methods in accuracy at *all* labeling ratios w.r.t. the paired Wilcoxon signed rank test at $p = 0.01$, as well as achieves the lowest rank w.r.t. test error. On average, *MinEnt* is the closest competition, followed by *AEW*. Despite being supervised, *IDML* does not perform on par. This may be due to labeled data not being sufficient to learn a proper metric in high dimensions, and/or the labels introduced during self-learning being noisy. We also find *Grid* and $Rand_d$ to rank at the bottom, suggesting that learning the graph structure provides advantage over these standard techniques.

Table 3: Average test accuracy and rank (w.r.t. test error) of methods across datasets for varying labeling %. ▲ (p<0.005) and △ (p<0.01) denote the cases where PG-LEARN is significantly better w.r.t. the paired Wilcoxon signed rank test.

Labeled	PG-L	*MinEnt*	*IDML*	*AEW*	*Grid*	$Rand_d$
10% acc.	**0.8819**	0.8594▲	0.8036▲	0.8448▲	0.8129▲	0.7952▲
rank	**1.20**	2.20	4.40	2.80	4.80	5.60
20% acc.	**0.8900**	0.8504▲	0.8118▲	0.8462▲	0.8099▲	0.8088▲
rank	**1.42**	2.83	4.17	2.92	4.83	4.83
30% acc.	**0.9085**	0.8636▲	0.8551▲	0.8613▲	0.8454▲	0.8386▲
rank	**1.33**	3.67	3.83	3.17	4.00	5.00
40% acc.	**0.9153**	0.8617▲	0.8323▲	0.8552▲	0.8381▲	0.8303▲
rank	**1.67**	3.67	3.50	3.67	4.00	4.50
50% acc.	**0.9251**	0.8700△	0.8647▲	0.8635▲	0.8556▲	0.8459▲
rank	**1.50**	3.17	3.83	3.67	4.00	4.83

4.2.2 Parallel Experiments with Noisy Features. Next we fully evaluate PG-LEARN in the parallel setting as proposed in Algo. 2. Graph learning is especially beneficial for SSL in noisy scenarios, where there exist irrelevant or noisy features that would cause simple graph construction methods like kNN and *Grid* go astray. To the effect of making the classification tasks more challenging, we *double* the feature space for each dataset, by adding 100% new noise features with values drawn randomly from standard $Normal(0, 1)$. Moreover, this provides a ground truth on the importance of features, based on which we are able to quantify how well our PG-LEARN recovers the necessary underlying relations by learning the appropriate feature weights.

Setup: We report results comparing PG-LEARN only with *MinEnt*, *Grid*, and $Rand_d$—in this setup, *IDML* failed to learn a metric in several cases due to degeneracy and the authors' implementation[10] of *AEW* gave out-of-memory errors in many cases. This however does not take away much, since *MinEnt* proved to be the second-best after PG-LEARN in the previous section (see Table 3) and *Grid* and $Rand_d$ are the typical methods used often in practice.

Given a budget B units of time and T parallel threads for our PG-LEARN, each competing method is executed for a total of BT units, i.e. all methods receive the same amount of processing time.[12] Specifically, *MinEnt* is started in T threads, each with a random initial configuration that runs until time is up (i.e., to completion, no early-terminations). *Grid* picks (k, σ) from the 2-d grid that we refine recursively, that is, split into finer resolution containing more cells as more allocated time remains, while $Rand_d$ continues picking random combinations of $(k, \boldsymbol{a}_{1:d})$. When the time is over, each method reports the hyperparameters that yield the highest validation accuracy, using which the test accuracy is computed.

Results: Table 4 presents the average test accuracy over 10 random samples from each dataset, using $T = 32$. We find that despite 32× more time, the baselines are crippled by the irrelevant features and increased dimensionality. In contrast, PG-LEARN maintains notably high accuracy that is significantly better than all the baselines on all datasets at $p = 0.01$.

Table 4: Test accuracy on datasets with 100% added noise features, avg'ed across 10 samples; 15 mins of hyperparameter tuning on $T = 32$ threads. Symbols ▲ (p<0.005) and △ (p<0.01) denote the cases where PG-LEARN is significantly better than the baseline w.r.t. the paired Wilcoxon signed rank test.

Dataset	PG-LRN	*MinEnt*	*Grid*	$Rand_d$
COIL	**0.9044**	0.8197▲	0.6311▲	0.6954▲
USPS	**0.9154**	0.8779△	0.8746▲	0.7619▲
MNIST	**0.8634**	0.8006▲	0.7932▲	0.6668▲
UMIST	**0.8789**	0.7756▲	0.7124▲	0.6405▲
YALE	**0.6859**	0.5671▲	0.5925▲	0.5298▲

Figure 6 (a) shows how the test error changes by time for all methods on average, and (b) depicts the validation and the corresponding test accuracies for PG-LEARN on an example run. We see that PG-LEARN gradually improves validation accuracy across threads over time, and test accuracy follows closely. As such, test error drops in time. *Grid* search has a near-flat curve as it uses the same kernel bandwidth on all dimensions, therefore, more time does not help in handling noise. $Rand_d$ error seems to drop slightly but stabilizes at a high value, demonstrating its limited ability to guess parameters in high dimensions with noise. Overall, PG-LEARN outperforms competition significantly in this high dimensional noisy setting as well. Its performance is particularly noteworthy on YALE, which has small $n = 320$ but large $2d > 2K$ half of which are noise.

Finally, Figure 7 shows PG-LEARN's estimated hyperparameters, $\boldsymbol{a}_{1:d}$ and $\boldsymbol{a}_{(d+1):2d}$ (avg'ed over 10 samples), demonstrating that the noisy features $(d + 1) : 2d$ receive notably lower weights.

5 CONCLUSION

In this work we addressed the graph structure estimation problem as part of relational semi-supervised inference. It is now well-understood that graph construction from point-cloud data has

[12] All experiments executed on a Linux server equipped with 96 Intel Xeon CPUs at 2.1 GHz and a total of 1 TB RAM, using Matlab R2015b Distributed Computing Server.