CORE: Self- and Semi-supervised Tabular Learning with COnditional REgularizations

Xintian Han
NYU
xintian.han@nyu.edu

Rajesh Ranganath
NYU
rajeshr@cims.nyu.edu

Abstract

Self- and semi-supervised learning have achieved great success in modeling languages, images and videos with the help of unlabeled data. However, the existing self- and semi-supervised learning methods rely on the specific structure of the data, which makes them harder to extend to the domain of tabular data. Recently, auto-encoder based self- and semi-supervised learning approach VIME has been proposed for tabular data. But auto-encoder may memorize the input and may not provide more informative signals in the representation than the original input. In this work, we provide a setting under which unlabeled data can help create useful representations for the supervised tasks. And we propose COnditional REgularization (CORE) in addition to the reconstruction loss in auto-encoder approaches to ensure our learned code capture useful signals for the downstream tasks. We show the priority of our regularization in one simulation and two real-world tasks.

1 Introduction

Tabular data is one major type of data in daily life. Deep learning has achieved state-of-the-art performance in modeling tabular data [27, 1]. The success of deep models requires a large labeled dataset. However, labels may not always be available in real life tasks. For example, if a patient does not come back to the hospital again after a single visit, we may lose the label information to determine whether they may have heart failure in 10 years. Self- and Semi- supervised learning methods provide effective ways to utilize the unlabeled data in images [8, 10, 7, 23, 24, 37, 30, 26, 25, 14, 4, 13, 11, 5, 16], text [21, 17, 3, 6, 20, 18, 19, 9, 36], videos [32, 33, 22, 31, 10], and recently tabular data [35, 1, 34]. In the recent work VIME [35], auto-encoders use unlabeled data to help learn predictive representations. The encoder creates a representation from a corrupted version of the input and the decoder reconstructs the input from the representation. However, there is no guarantee that the representation can capture the most useful information in the input that determines the label. The encoder may just memorize the input and the decoder can finish the reconstruction task by itself.

In this work, we study a data generation assumption where unlabeled data can help create useful representations for supervised tasks. In this setting, we propose self-supervised COnditional REgularization (CORE), which ensures that auto-encoders' codes capture useful signals for downstream predictions while discarding datapoint-specific noise. This objective is easily combined with the usual auto-encoder reconstruction loss. We also propose semi-supervised CORE which acts as a consistency loss. Our approach is motivated by [28], which recovers latent confounders in a causal setting by limiting the additional mutual information between one treatment and the confounder given other treatments. CORE limits the mutual information between one feature and the latent representation given other features. The CORE loss plays a similar role as the additional mutual information in [28]. CORE prevents the encoder from memorizing individual noise in the features. CORE outperforms denoising auto-encoders, context auto-encoders and VIME in a simulation experiment and competes

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

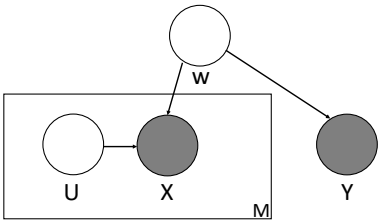


Figure 1: Graphical model of the data generation process.

with the baselines in two real-world experiments on Higgs-Boson identification, and in-hospital mortality prediction.

2 Proposed model: CORE

In this section, we first introduce the key assumption in this work: unlabeled data can help build informative representations for downstream tasks. We then propose our method CORE. Throughout this work, we use the subscript i to denote the i-th datapoint and we use the superscript j to denote the j-th dimension in a single input X.

2.1 Assumption

We assume the generation of the input $X \in \mathbb{R}^M$ and outcome $Y \in \mathbb{R}$ is the following. A low-dimensional signal $W \in \mathbb{R}^D$ is first drawn from $p(W)$, where $D < M$ and individual noises U^j are independently drawn from $p(U^j)$ for $j = 1, \ldots, M$. Our observed input X^j is generated from $p(X^j|W, U^j)$ for $j = 1, \ldots, M$ and outcome Y is generated from $p(Y|W)$. The graphical model of the data generating process is shown in Figure 1. Under these assumptions, predicting Y requires estimation of W and learning to infer W is possible on unlabeled X.

2.2 Self-supervised CORE

We now introduce self-supervised learning on unlabeled data X. Suppose we have an encoder enc and a decoder dec in an auto-encoder. The key insight of the CORE algorithm is to prevent the encoder from memorizing X, while still achieving good reconstruction loss. This discards the noise from U not helpful for predicting Y while recovering W, which does predict Y.

For a given input X, CORE first creates $\hat{X}(j)$ that has the j-th dimension replaced by samples from $p(X^j|X^{-j})$. Sampling from $p(X^j|X^{-j})$ for each j seems to require M different conditional distribution estimators. But with the help of DDLK [29], we only need to generate one *knockoff* $\tilde{X}$. For any index set S, the knockoff $\tilde{X}$ generated by DDLK satisfies the property $(\tilde{X}, X) \overset{d}{=} (\tilde{X}, X)_{\text{swap}[S]}$, where for $j \in S$, the swapping operation exchanges $\tilde{X}^j$ and X^j. After generating $\tilde{X}$ using DDLK, we set $\hat{X}(j)^j = \tilde{X}^j$ and $\hat{X}(j)^{-j} = X^{-j}$ where X is the original input. This algorithm ensures that each $\hat{X}(j)$ is sampled from $P(X^j|X^{-j})$. This simplifies the M conditional sampling processes. Then we introduce our *conditional regularization (CORE)*:

$$\sum_{j=1}^{M} \mathbb{E}_{X,\hat{X}(j)} \|dec(enc(X)) - dec(enc(\hat{X}(j)))\|_2^2$$

The encoder is used for downstream prediction of Y. If the encoder memorizes X^j, then the CORE loss will be large due to the conditional resampling of the j-th dimension in $\hat{X}(j)$. Since it is possible for the encoder to memorize X and $\hat{X}$ and for the decoder to minimize the objective, we keep the decoder constant in the CORE loss i.e., we do not take the gradient descent with the decoder parameters in the regularization loss. We denote this decoder as $ng(dec)$ (for "no gradient").

	Method	MSE
Supervised	Supervised Linear Regression	9444.25
	PCA	11.75
Self-Supervised	CORE	$\mathbf{1.17 \pm 0.05}$
	Denoising Auto-encoder	108.93 ± 6.80
	Context Encoder	1.49 ± 0.05
	VIME	104.07 ± 3.00

Table 1: MSE of all the methods on the simulation experiment

Together with the reconstruction loss in the auto-encoder, CORE minimizes the following objective over the encoder and the decoder:

$$\mathbb{E}_X \|dec(enc(X)) - X\|_2^2 + \alpha \cdot \sum_{j=1}^{M} \mathbb{E}_{X,\hat{X}(j)} \|ng(dec)(enc(X)) - ng(dec)(enc(\hat{X}(j)))\|_2^2,$$

where α is a trade-off parameter between the reconstruction loss and the regularization loss. If we have downstream tasks, α can be selected on the validation performance on the downstream tasks.

2.3 Semi-supervised CORE

Suppose we still have an encoder that learns a representation and a predictor f to predict the label. Then we minimize the following semi-supervised learning loss:

$$\mathbb{E}_{X,Y} l_{sup}(f(enc(X)), Y) + \beta \cdot \sum_{j=1}^{M} \mathbb{E}_{X,\hat{X}(j)} l_c(f(enc(X)), f(enc(\hat{X}(j)))),$$

where β is a trade-off parameter between the supervised loss l_{sup} and the consistency loss l_c, which can be determined by validation performance. This is similar to the semi-supervised objective in VIME [35].

3 Experiments

For self-supervised learning, we compare the proposed CORE method with the following baselines: self-supervised VIME, denoising auto-encoders, and context encoders. For semi-supervised learning, we compare with semi-supervised VIME. We also study self+semi-supervised training, which first trains the encoder using self-supervised learning and then uses semi-supervised training to finetune the pretrained encoder and train the predictor. We compare against self+semi-supervised VIME. The description of baselines, datasets, train/valid/test split and implementation details can be found in appendix B. A sensitivity analysis of hyperparameters and number of labeled training points can be found in Appendix A.

3.1 Simulation Experiment

In this simulation, we use a linear model to simulate data (x_i, y_i) according to the graphical model in Figure 1 . The goal is to predict the output y_i from the input x_i. More details can be found in Appendix B.3. We use a linear layer for the encoder and decoder. For the supervised training, we use the close-form solution from linear regression from representation $enc(X)$ to Y. We compare mean-squared error (MSE) for each method. The results are shown in Table 1. Our method performs better than all baselines. In this experiment, we also compare against PCA, which aims to find components that maximize variance so it will choose the noisy but possibly non-informative dimensions. PCA does not do well on this task. We only study the self-supervised case for this simple linear example.

3.2 Higgs Bosons

The goal of the Higgs task [2] is to distinguish Higgs Bosons from background signals using the features obtained by the detectors. We compare the accuracy of all methods in Table 2. Self-supervised CORE outperforms all methods. Semi-supervised and Self+Semi-supervised CORE also

	Method	Accuracy
Supervised	4-layer perceptron	0.6055 ± 0.0041
	2-layer perceptron	0.6101 ± 0.0032
Self-supervised	CORE	$\mathbf{0.6692 \pm 0.0055}$
	Denoising Auto-encoder	0.6088 ± 0.0055
	Context Encoder	0.6096 ± 0.0154
	VIME	0.6675 ± 0.0056
Semi-supervised	CORE	0.6189 ± 0.0078
	VIME	0.6115 ± 0.0118
Self + Semi-supervised	CORE	0.6667 ± 0.0058
	VIME	0.6595 ± 0.0048

Table 2: Accuracy of all the methods on Higgs dataset

	Method	AUC
Supervised	4-layer perceptron	0.7837 ± 0.0029
	2-layer perceptron	0.7790 ± 0.0021
Self-supervised	CORE	0.7941 ± 0.0051
	Denoising Auto-encoder	0.7918 ± 0.0053
	Context Encoder	0.7806 ± 0.0042
	VIME	0.7914 ± 0.0028
Semi-supervised	VIME	0.7994 ± 0.0037
	CORE	0.7992 ± 0.0048
Self+Semi-supervised	VIME	0.7889 ± 0.0037
	CORE	0.7930 ± 0.0027

Table 3: AUC of all the methods on In-hospital Mortality Prediction

beat the semi-supervised and self+semi-supervised VIME. On this dataset, although semi-supervised training is better than the supervised ones, self+semi-supervised training does not outperform the self-supervised-only approach.

3.3 In-hospital Mortality Prediction

The goal of this task is to predict in-hospital mortality based on the first 48 hours clinical variables such as heart rate, temperature, and blood pressure. We compare AUC of all methods in Table 3. Self-supervised CORE achieves higher (better) AUC than all other methods. Self+semi-supervised CORE also beats the self+semi-supervised VIME. Semi-supervised CORE and VIME have roughly the same performance.

4 Conclusions

In this work, we present CORE, a regularization term that helps representations capture the information shared among features using unlabeled data. In linear simulations, our method beats all baselines. On real-world datasets, our method perform slightly better than the baselines. The linear simulation strictly satisfies our assumptions, while they may not be satisfied in real-world datasets. For example, if the features do not contain independent noise, then CORE may not work. If the features are all categorical (gender, race etc.) (e.g. the UCI Income dataset in [35]), there is little chance for features to have noise. Then CORE may not help to learn better representations. We show in this work that CORE prevents memorization in auto-encoders under the tabular data setting which improves performance in downstream tasks. Future work includes extending CORE to language and image data.

References

[1] S. O. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. *arXiv preprint arXiv:1908.07442*, 2019.

[2] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] X. Chen and K. He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[8] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015.

[9] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*, 2017.

[10] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[11] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[12] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, 2019.

[13] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[14] O. Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020.

[15] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[16] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[17] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

[18] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[19] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[23] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[24] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5898–5906, 2017.

[25] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[26] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[27] S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

[28] R. Ranganath and A. Perotte. Multiple causal inference with latent confounding. *arXiv preprint arXiv:1805.08273*, 2018.

[29] M. Sudarshan, W. Tansey, and R. Ranganath. Deep direct likelihood knockoffs. *arXiv preprint arXiv:2007.15835*, 2020.

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[31] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018.

[32] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015.

[33] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[34] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.

[35] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar. Vime: Extending the success of self- and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33, 2020.

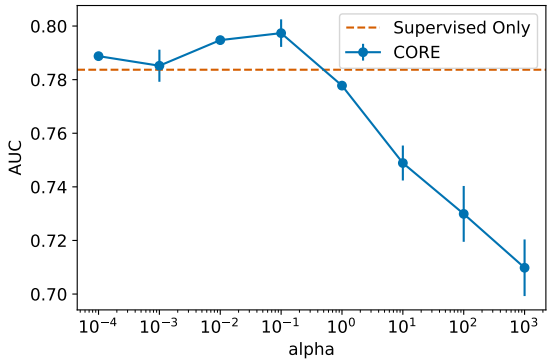
[36] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[37] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

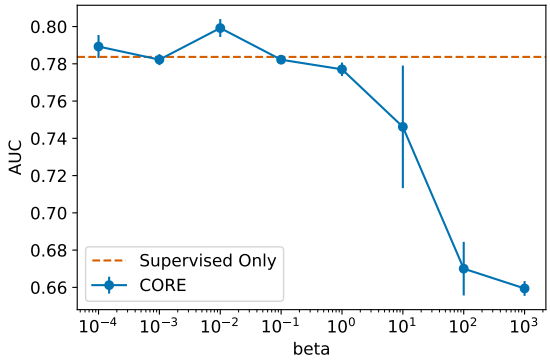
A Sensitivity Analysis

A.1 Hyperparameters α and β

Here we show the AUC on the in-hospital mortality prediction for different self-supervised regularization scale α's and semi-supervised regularization scale β's in Figure 2. As we notice in the plot, when α or β is too small, there is no regularization for the code space, and the performance is close to the supervised baseline. When α or β is too large, the regularization may cause the encoder to learn a constant code to minimize the regularization loss and therefore the supervised performance is bad.



(a) α



(b) β

Figure 2: AUC versus different α's and β's.

A.2 Number of labeled training samples

We show the accuracy versus the number of labeled training samples for the two best method (self-supervised VIME and CORE) on Higgs dataset in Table 4. As the number of labeled training samples grow, the accuracy gets higher and Self-supervised CORE is always better than Self-supervised VIME.

Num of labeled training samples	Self-supervised VIME acc	Self-supervised CORE acc
1000	0.6352 ± 0.0079	0.6357 ± 0.0070
5000	0.6561 ± 0.0074	0.6642 ± 0.0045
10000	0.6675 ± 0.0056	0.6692 ± 0.0055

Table 4: Self-supervised CORE and VIME accuracy on Higgs dataset

B Baseline and Implementation Details

B.1 Baselines

VIME [35]: Pretext tasks are to recover an input sample x from its corrupted input and identify the mask. The corrupted input $\hat{x}$ is created by sampling a binary mask m and replace the masked positions with marginal sample $\tilde{x}$ from the data.

$$\hat{x} = m \odot \tilde{x} + (1 - m) \odot x$$

The pretext tasks are

1. Predict the mask m from the corrupted input $\hat{x}$
2. Recover the input x from the corrupted input $\hat{x}$.

The representation learned from corrupted input that solves the pretext tasks is used in the downstream tasks. In the test time to learn a supervised MLP, they use the true x as the input for the encoder instead of the corrupted input.

In a semi-supervised regime, the unlabeled data is used to construct a consistency loss. The final loss is the supervised loss on the labeled data + consistency loss from the unlabeled data.

Context encoder [26]: The pretext task is inpainting. The goal is to learn a representation to fill in the masked features from the unmasked features. They first sampling a binary mask m and the inpainting task aims to minimize

$$\|(1 - m) \odot (dec(enc(m \odot (x))) - x)\|_2^2$$

Denoising auto-encoder [30]: The pretext task is to learn the representation using an auto-encoder by reconstruction of x from a corrupted version of x. This corrupted version can be adding Gaussian noise to x or randomly replacing some values of x with zero. Since randomly masking values has been used in VIME and context encoder, we only consider adding Gaussian noise corruption here. The corrupted input is created by

$$\hat{x} = x + a * \epsilon,$$

where a is a hyperparameter and ϵ is sampled from the standard normal distribution. The denoising auto-encoder aims to minimize

$$\|dec(enc(\hat{x})) - x\|.$$

For the supervised learning on the labeled dataset, we consider PCA and linear regression on the linear simulation and multi-layer perceptrons for the real-world datasets. In the real-world datasets, we use a 2-layer perceptron for the encoder, decoder and predictor for CORE and the baselines. We consider a 2-layer perceptron which corresponds to the predictor and a 4-layer perceptron which corresponds to encoder + predictor as the supervised baselines.

B.2 Datasets

B.3 Simulation

We first sample θ uniformly from [-1, 1] and $A \in \mathbb{R}^{20 \times 1}$ whose element is sampled i.i.d. uniformly from [-1,1]. We fix θ and A once sampled. We set $a = 0.1$. Then we sample $w_i \in \mathbb{R}$ and noise $\epsilon_{y_i} \in \mathbb{R}$ from standard Gaussian. And we create the output $y_i \in \mathbb{R}$ by

$$y_i = \theta^T w_i + \epsilon_{y_i}$$

The input x_i has 100 dimensions. We sample $\epsilon_{x_i} \in \mathbb{R}^{20}$ from standard Gaussian. Then we create the first 20 dimensions of x_i by

$$x_i = A w_i + a * \epsilon_x,$$

We sample the other 80 dimensions of x_i i.i.d from $N(0, 10)$. These 80 dimensions have large individual noise.

B.3.1 Higgs Bosons

The data is simulated using Monte Carlo methods[1]. The features include kinematic properties measured by the particle detectors (first 21) and functions of the first 21 features (last 7). The features are observed with observation noise. Our method is suitable for this dataset. The dataset is roughly balanced.

B.3.2 In-hospital Mortality Prediction

In this experiment, we use data from a freely available critical care dataset MIMIC-III [15][2]. We create an in-hospital mortality prediction task based on [12][3]. The goal is to predict in-hospital mortality based on the first 48 hours clinical variables such as heart rate. temperature, and blood pressure. There are 17 clinical variables. They are presented as time-series. To make it tabular, we choose the maximum of the features in the first 24 hours and the maximum of the features in the last 24 hours. Then there are 34 tabular features. We use these feature as the input. A person's clinical variable may vary a lot during the day and has a lot of individual noise so CORE is suitable for clinical predictions.

B.4 Train/valid/test Split

B.4.1 Simulatuon Experiment

We sample 7,500 unlabeled data x (5,000 for self-supervised training the encoder and 2,500 for validation) and 2,00 labeled pairs (x, y) (1,00 for training, 50 for validation and 50 for test) for supervised training.

B.4.2 Higgs Bosons

The whole dataset has 11,000,000 datapoints. We randomly select 50,000 unlabeled datapoints for training and 25,000 for validation in the self-supervised task and 5,000 for training, 2,500 each for validation and testing in the supervised task.

B.4.3 In-hospital Mortality Prediction

In this task, we only have 21,139 data points in total. We use 60% for unlabeled training and 20% for unlabeled validation. We use 10% for labeled training and 5% each for labeled validation and test.

B.5 Implementation Details

We explore batch size from {100, 500, 1000}, learnging rate from {0.001, 0.004, 0.0001, 0.0004}, self-supervised regularization scale α from {0.01, 0.1, 1.0, 10.0, 100.0, 1000.0} and semi-supervised regularization scale β from {0.01, 0.1, 1.0, 10.0}. In the linear simulation, the representation size is chosen from {1, 5, 10}. In real-world datasets, the representation size is chosen from {50, 100}. We use a linear encoder and decoder for the linear simulation and a two layer MLP with ReLU activation and hidden size 300 encoder and decoder for the real-world datasets. We report the test performance on the hyperparameter that achieves the best validation metric. An analysis of sensitivity to data size and hyperparameters is included in Appendix A. All the experiments are done on internal clusters which include RTX 8000 and V100. We report the mean and standard deviation of the results over three different seeds.

[1]The data can be obtained from here: https://archive.ics.uci.edu/ml/datasets/HIGGS

[2]We can request access of this de-identified data here https://mimic.mit.edu/iii/gettingstarted/

[3]We use the code from https://github.com/YerevaNN/mimic3-benchmarks, MIT license