Learning to Predict With Unavailable Features: an End-to-End Approach via Knowledge Transferring

Chun-Chen Lin, Li-Wei Chang, Chun-Pai Yang, and Shou-De Lin

National Taiwan University, Taipei, Taiwan
{r07922006,r08922041,d05922017}@ntu.edu.tw,
sdlin@csie.ntu.edu.tw

Abstract. Learning from data with missing values has become a commonly faced challenge in real-world applications. This paper emphasizes on a specific scenario that a subset of training feature dimensions becomes unavailable during the prediction stage. In this certain case, most of the existing approaches suffer from the vacancy of designated feature dimensions, thus not capable of providing quality results. This paper proposes a novel neural-based learning framework to leverage the knowledge obtained during training to alleviate the influence from missing of certain features during prediction. Our solution incorporates two knowledge transferring strategies allowing models to learn from decaying features as well as from a teacher network trained with full information. Experiment results show promising outcomes comparing with the state-of-the-art imputation-based solutions.

Keywords: Machine learning · Data mining · Missing data · Knowledge distillation.

1 Introduction

Machine Learning (ML), in particular supervised learning, models have been extensively studied and implemented for real-world applications. A typical working flow of supervised learning systems looks like: first, train model parameters on available data (that is, the *training set*) given a designated optimization objective, then perform model prediction on demand (or predicting on a *testing set*). Predictions are generated by the learned mapping function from features to labels.

However, ML models sometimes suffer from the *incompleteness of data* in real-world scenarios. There has been many works focusing on handling missing data features [2, 18, 16, 6, 20, 21, 11]. Different from most of the existing works that consider arbitrary data missingness or missing at random with certain distributions [17, 14], this paper considers a particular scenario that, to our knowledge, has not yet been addressed previously: a known subset of feature dimensions are *unavailable* during prediction, but they are available during training. Fig. 1 shows an illustration of such scenario.

	f_1	f_2	f_3	f_4	f_5	f_6
$\boldsymbol{x}_1$	0.2	2.5	1.1	0.1	-1	1
$\boldsymbol{x}_2$	1.4	4.0	1.5	0.2	1	1
$\boldsymbol{x}_3$	0.8	7.2	1.8	0.1	-1	0
$\boldsymbol{x}_4$	1.5	3.4	1.2	0.1	-1	1

(a) Training data

	f_1	f_2	f_3	f_4	f_5	f_6
$\boldsymbol{x}_1$	1.0	N/A	1.4	0.1	N/A	0
$\boldsymbol{x}_2$	1.5	N/A	1.7	0.2	N/A	1
$\boldsymbol{x}_3$	1.2	N/A	1.5	0.2	N/A	1
$\boldsymbol{x}_4$	0.9	N/A	1.5	0.1	N/A	0

(b) Test data (with unavailable features)

Fig. 1: An example of unavailable feature dimensions when performing model prediction. As shown above, in training data (a), all six features are available. However, in test data (b), feature dimensions f_2 and f_5 are missing (not available) for *all* data samples.

We have realized that such situation can happen in the real world for at least two reasons. First, some ML applications demand very fast or even real-time response, such as web searching, high-speed trading, real-time bidding (RTB), decision making for robots, etc. Since they demand fast response, features that require longer time to produce will not be available in the prediction phrase, despite that they are available in the training dataset. Taking real-time bidding as an example, the model usually needs to decide within 0.1 second (or even less) how much it shall bid for the user. However, for some content-based features (e.g., the deep embeddings of the websites this user just visited) or statistical features (e.g., the average visiting counts of the web pages this user visited in the past week) can take longer amount of time to generate, and thus only available for training but not prediction.

Another real-world scenario happens due to the concern of privacy or security, in particular in application domains like finance, medical, or healthcare-related ones. For instance, a bank might want to use machine learning models to determine whether to offer a loan to a person. Although features relevant to this person's medical record or tax payment history can be very beneficial for the model to make a decision, but candidates might not be willing to provide them on the spot due to privacy/security concern. On the other hand, those information can be available for people who already been offered or rejected a loan (i.e., available in training), it would be a waste to completely ignore them in building a model. Note that here we assume that which feature dimensions are available is already known in advance, which seems to be reasonable since we normally know in advance the features take time to generate and which features might be privacy-sensitive.

To handle this situation, the most straightforward solution is to simply ignore those feature dimensions during the model training process (e.g., training with only f_1, f_3, f_4, f_6 in Fig. 1), thus the unavailability of certain features will

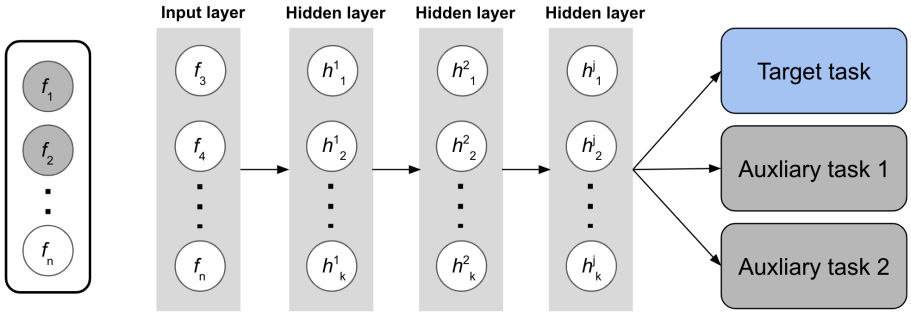


Fig. 2: An illustration of multi-task learning for tackling the problem of unavailable test features. Assume the input feature f_1 and f_2 (marked in grey) is unavailable at prediction phase. All the features except f_1 and f_2 are fed into input layer during training, and the auxiliary tasks are to predict f_1 and f_2 during multi-task learning.

not affect model predictions. However, doing so prohibits us from leveraging the potentially useful information in building the model. Another commonly-adopt solution for this scenario is to perform data imputation on those specific dimensions. In other words, we can still train a model using all features; and during prediction, we first fulfill the missing values and then used the imputed data to perform prediction. This strategy has two potential drawbacks. First, such two-stage framework requires even longer amount of time to generate the prediction results, which conflicts with one of the original goal to perform fast prediction. Second, the error on the imputation stage will be propagated to the second stage, which can affect the performance of the trained model.

Apart from conventional strategies, here we propose a simple yet novel baseline using multi-task learning framework as a potential solution to tackle this problem. As shown in Fig. 2, the original task can serve as *target* task while *auxiliary* tasks are to predict the features which are unavailable. The goal is to learn a representation generally helpful for the target task via multi-task learning. Note that to train a multi-task framework of n tasks, we normally need $n-1$ hyper-parameters to specify the relative importance of each task in the joint objective function. This can become a major concern during training when the number of tasks (i.e. unavailable features) grows since the search space of hyper-parameter can become untraceable. Nevertheless, here we consider this solution as one of the baselines to be compared with later on.

In this paper, we propose a generic neural-network based framework to make the best usage of the unavailable features for training. Our learning framework consists of two parts: (i) a self knowledge-transfer component and (ii) a teacher-student transfer framework. For the first part, we develop a decaying scheme on the targeted weights in neural networks to gradually transferring the information of carried by the unavailable features to the available ones. For the latter part, we establish a teacher-student network structure to distill the knowledge from

the full-feature trained teacher network to the target student network using only available features. The experiments show significantly superior results of our framework compared to methods describe above, including state-of-the-art approaches handling missing/incomplete features.

Overall, the main contributions of this paper can be summarized in three-fold:

- We propose the challenge of handling unavailable features during prediction, and highlight two usage scenarios in the demand of fast prediction and privacy-concern predication.
- Bypassing the requirement of performing missing feature imputation, we propose a novel knowledge-transferring framework to handle this situation based on a deep neural network structure.
- We evaluate our method on several benchmark datasets to demonstrate the effectiveness of the proposed solution.

2 Methodology

This section introduces our solution to tackle the problem that combines two ideas: (a) self-transferring and (b) teacher-student transferring models.

2.1 Self-Transfer through Weight Decaying

First, we establish a self-transfer strategy, training a single neural network model utilizing a weight decay mechanism to gradually decay and transfer the information carried by the unavailable features to the model weights for the available features. It forces the model to learn relatively robust and useful intermediate representations gradually throughout the decay process.

Fig. 3-b illustrates the idea of our weight decay mechanism designed to be applied on the input features that are unavailable during the model prediction phase. The connection weights between the first layer and the next layer are gradually reduced by consistently multiplying a scalar r (which is between zero and one) in each training iteration. Therefore, in the earlier training stage, the model does utilize the information from unavailable features, but progressively the responsibility is shifted to the available ones due to the decay of weights for unavailable features. Eventually, the values of the unavailable feature nodes becomes irrelevant since their associated weights have been decayed to zeros.

One realization of the *weight-decay* strategy is shown in Algorithm 1. The multiplier r starting from 1 is decreased by a fixed value $\frac{1}{T}$ at each iteration until it becomes 0. For example, we can set T to 5,000 to let the weight decaying process take 5,000 steps to finish. We will cover the analysis of decay steps in Section 3.5.

Since the *weight-decay* method starts from the model trained with full features, the information from the *features* which are unavailable at prediction phase can be leveraged in the process. During the weight decaying process, the

Algorithm 1 FirstLayerWeightDecay

Require: Given trained network weight $\mathcal{W}^{(0)} = \{W_i\}_{i=1}^{L}$ with the full feature dimensions $\mathcal{F}$; the unavailable subset of feature dimensions $\mathcal{D} \subset \mathcal{F}$; the number of decay steps T.

Ensure: New network weight $\bar{\mathcal{W}}$ using only available feature dimensions $\mathcal{F} \setminus \mathcal{D}$.

$r \leftarrow 1$.
$k \leftarrow 0$.
while $r > 0$ **do**
  $r \leftarrow r - \frac{1}{T}$.
  for $d \in D$ **do**
    Set a multiplier r to $W_1[d, x]$ of $\mathcal{W}^{(k)}$, $\forall x$.
  end for
  $\mathcal{W}^{(k+1)} \leftarrow$ NetworkFineTuning $\left(\mathcal{W}^{(k)}\right)$.
  $k \leftarrow k + 1$.
end while
return $\bar{\mathcal{W}} := \mathcal{W}^{(k)}$.

model should adapt to the situation in which the *features* are becoming unavailable due to the shrinking of corresponding weights. At the end, the information originally extracted from the *features* are completely transferred to other parts of the model. Finally, the model no longer relies on the unavailable *features* during prediction. Note that such weight-decay mechanism can itself serve as a stand-alone solution.

2.2 Teacher-Student Framework for model-wise knowledge transferring

Previously, we consider training a model while adopting the networks to the decayed features become unavailable eventually. Here we consider another strategy using model-wise transferring that distills information from the model utilizing full-information to the model that can use only partial information.

Based on this idea, we first train a model that utilizes the complete feature set. It can be considered as a more accurate model because of the accessibility to the omniscience information. As a contrary, there is another model that is trained only on available features, which is supposed to be inferior but in reality it is the only model to be used during prediction. Our goal then becomes trying to improve the quality of the less-capable model through learning from the stronger omniscience model.

Here we adopt the framework of student-teacher network originally proposed for knowledge distilling to achieve the purpose. The model that utilizes only available features is considered as the student network, while the model utilizes all features is treated as the teacher. In particular, the teacher distills the knowledge of complete data to the student via the distillation loss, guiding the student to align in the output probability distribution. The distillation loss is defined as

cross-entropy loss of the teacher's and student's output probability distributions p and q,

$$H(p, q) = -\sum_{x \in X} p(x) \log q(x).$$

Fig. 3-(a+c) illustrates the concept of the knowledge transferring framework.

The teacher model is first trained on the training data of *full* feature information (complete training data). We then fixed the teacher model for distilling the knowledge to the student via the distillation loss. Student is trained on training data of only *available* features (incomplete training data) with two losses: (1) distillation loss from teacher's output probability distribution (*soft labels*), and (2) prediction loss from the ground truth label (*hard labels*). Eventually only the student model is used for prediction since it does not require the unavailable features. Note that the teacher and the student share the same model architecture. There is a trade-off λ between distillation loss and student loss when training the student. It affects how much the knowledge is transferred from the teacher to the student. The total loss of training the student is defined as:

$$L_{\text{Total}} = \lambda L_{\text{Distill}} + (1 - \lambda) L_{\text{Student}},$$

where L_{Distill} is the distillation loss and L_{Student} is the student loss. That is, if λ is set to be large, the student will consider the information from the teacher more than the information from the (incomplete) ground truth data. We observe that setting the λ to be above 0.5 yields the best performance in general, implying that the knowledge from the teacher plays an important role when unavailability occurs. The analysis of the λ is shown in Section 3.5.

2.3 Integrating the self-transfer and model-based transfer strategies

In this section, we would like to discuss how to integrate the two proposed ideas into one single framework. In a nutshell, we first conduct the weight-decay mechanism to train a student model that requires only available features to perform prediction, then fine-tune this network using the guidance from the teacher network, which is trained using the full dataset. Such way of integration allows the student network to observe the unavailable features before starting to learn from the teacher, which allows it to converge to a better local-optimal point. Fig. 3 illustrates the idea of integration.

The complete process can be stated in four steps:

1. Train the teacher model on complete training data.
2. Pre-train the student model with the weight-decay mechanism.
3. Transfer the knowledge from the teacher model to the student model with teacher-student framework.
4. Predicting using only the fine-tuned student model.

We will later evaluate the effectiveness of these two individual strategies as well as their combination.

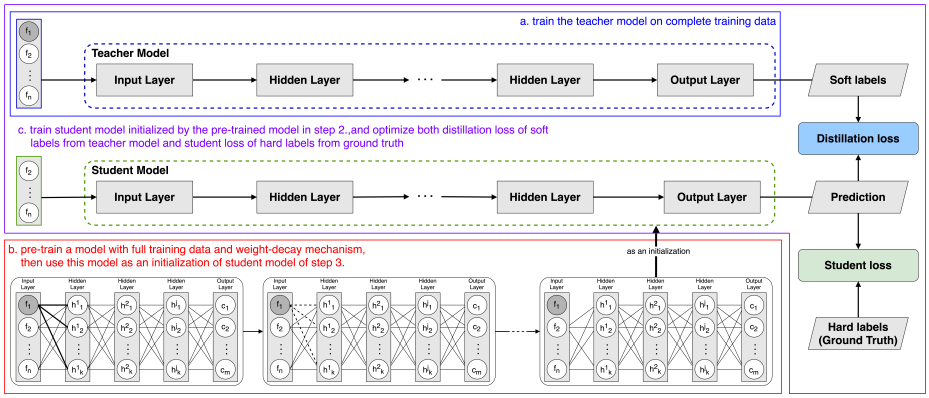


Fig. 3: An illustration of the integrated learning framework of self-transfer and model-based transfer strategies for tackling the problem of unavailable test features.

Table 1: Datasets for model evaluation.

	Avila	**Adult**	**Hepmass**	**Eye**	**Letter**	**Spam**
Instances	20,867	48,842	10,500,000	14,980	20,000	4,601
Features	10	14	27	15	16	57

3 Experiments

3.1 Experiment Setups

We consider six real-world datasets from UCI Machine Learning Repository [12], Avila, Adult, Hepmass, Eye, Letter and Spam, to quantitatively evaluate the performance of our proposed methods in comparisons with baselines and state-of-the-art imputation methods. Table 1 shows the properties of the datasets. In particular, we evaluate the performance using several missing rates on features including 20%, 40% and 60%. Note that the original data of Avila, Adult, Hepmass, Eye, Letter and Spam are fully observed. We randomly drop features and repeat the experiments. The results come from the average of 20 different random droppings.

Here we have four different hypotheses to answer:

1) Is our knowledge-transfer framework better than imputation-based solutions?
2) Is the knowledge-transfer framework better than the multi-task learning model proposed in Fig. 2?
3) Can the two proposed ideas, self-transferring and model-based transferring, be integrated seamlessly to further improve the performance?

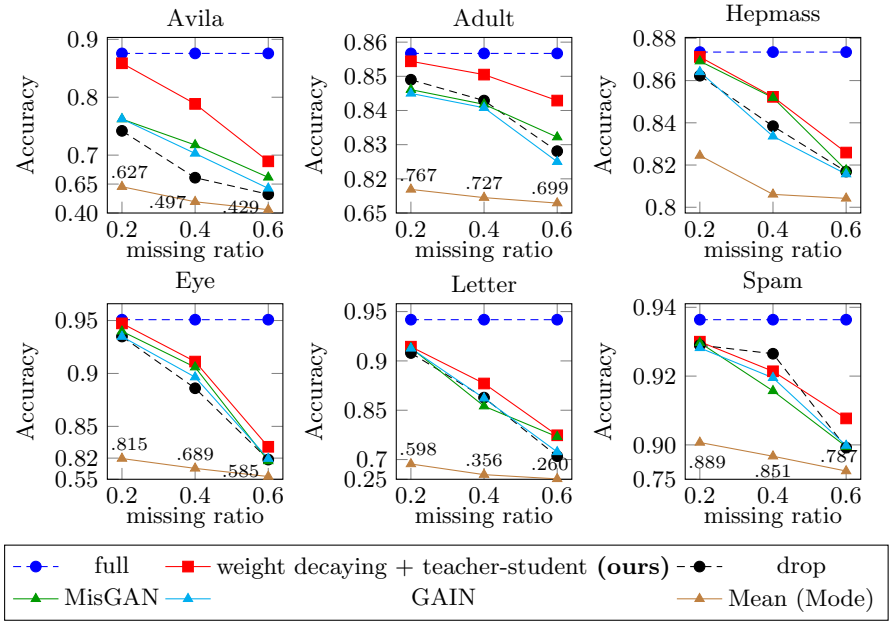


Fig. 4: Overall results on Avila, Adult, Eye, Letter and Spam with missing ratio 0.2, 0.4 and 0.6.

4) How to set and how sensitive are the hyper-parameters?

We use 5-fold cross validation and conduct each experiment 20 times. We report average categorical accuracy as the performance metric across 20 experiments.

For classification, we use fully-connected networks with 3 dense layers of dimension 32 and an SGD optimizer with learning rate 0.01. This is used as the final-stage classification model for two imputation baseline models, multi-task baseline model, and our student network to assure fair comparison.

3.2 Comparison with Imputation-based Methods

We consider state-of-the-art imputation methods GAIN [21] and MisGAN [11] as the main competitors and mean (mode) imputation for numerical (categorical) data as the baselines. Note that we do not compare with the other imputation methods such as [2, 18, 16, 20, 6] since in the literature they are not as strong as GAIN and MisGAN. For mean (mode) imputation, we replace missing values in the testing data with mean (mode) of corresponding features in training data. To compare these imputation methods against ours, we evaluate their performance of post-imputation prediction.

Note that MisGAN and GAIN rely on randomly masking training data since they assume random missing during training. In our scenario, we then allow such

Table 2: Overall results on Avila, Adult, Eye and Letter with missing **two** most important test features.

	Methods	**Accuracy**			
		Avila	**Adult**	**Eye**	**Letter**
Upper-bound	Full	.8757 ± .0767	.8567 ± .0025	.9508 ± .0031	.9427 ± .0042
Lower-bound	Drop	.6741 ± .0785	.8356 ± .0018	.9154 ± .0040	.8877 ± .0154
Ours	Weight-decay + Teacher-student	**.7887 ± .0336**	**.8461 ± .0016**	**.9298 ± .0046**	**.9001 ± .0036**
Multi-task	Multi-task	.6834 ± .0281	.8296 ± .0016	.9215 ± .0027	.8957 ± .0052
Baselines	MisGAN	.7123 ± .0122	.8295 ± .0027	.9208 ± .0048	.8902 ± .0082
	GAIN	.7157 ± .0117	.8294 ± .0038	.9181 ± .0051	.8927 ± .0067
	Mean (Mode)	.5671 ± .0859	.7721 ± .0092	.6902 ± .0117	.5137 ± .0265

mask to focus only on the missing of unavailable features to boost their performance. For hyperparameter tuning of them, we try four different *obs_prob* settings $\{0.01, 0.2, 0.5, 0.8\}$ and corresponding *p_miss* settings $\{0.99, 0.8, 0.5, 0.2\}$.

As shown in Fig. 4, the proposed method outperforms the baselines for the all cases. *Full* represents when all the features are available during prediction, which can serve as the upper-bound of this task. *Drop* is when we only utilize the observed variables during training and prediction time, which can serve as the lower-bound of our proposed methods.

Mean (Mode) imputation seems to perform worst among all the methods, which is reasonable. Both GAIN and MisGAN do not perform well either. The possible reason is that there is still a gap between the imputed missing values and the real values under the certain missing scenario, and the error propagated from of imputation can hurt the training performance. They can even perform worse than *Drop* for some cases. For the case of missing ratio 0.4 on Spam, all the methods including ours perform worse than simply dropping the features. One possible reason is that the room for improvement (about 2%) upon the case is not enough for the more complex methods to show their effect.

Note that that the models' performance decreases when increasing the number of unavailable features, and the performance of mean (mode) imputation degrades the most. However, the results show that the proposed method is the most robust against more missing features compared to others.

3.3 Comparison with Multi-task Learning

This is a comparison between two non-imputation frameworks. Fig. 2 illustrates the idea of multi-task learning where there are two unavailable features and thus two auxiliary tasks. Note that the number of auxiliary tasks in multi-task learning is equivalent to the number of unavailable features, which can bring computation challenge to the optimization in practice as search space for hyper-parameters (one for each task) grows in polynomial. Thus here we only perform grid-search for multi-task learning given 2 most important features unavailable, where the importance of specific feature is measure by the gap between the accuracy when the feature is removed from training. There are two hyper-parameters

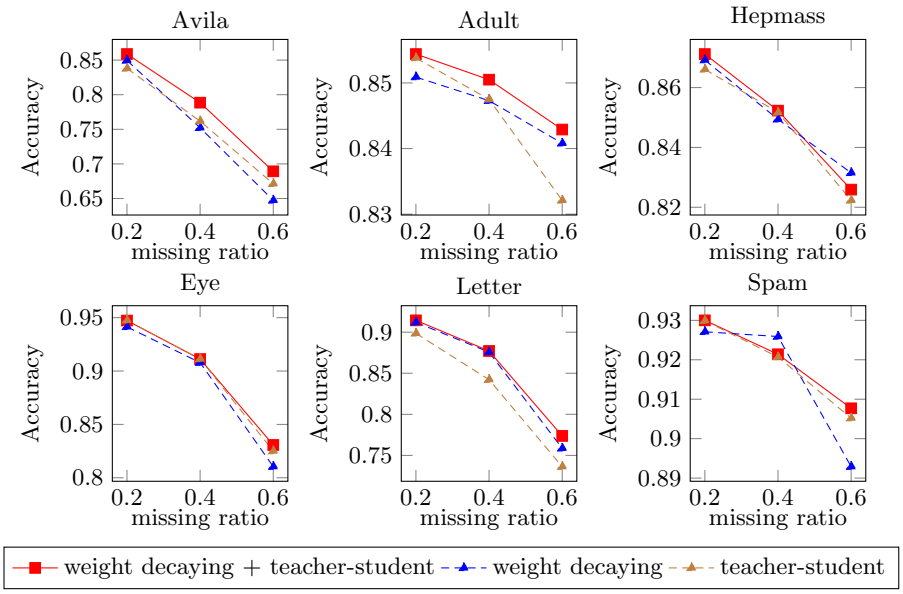


Fig. 5: Comparison among weight-decay, teacher-student network learning and their combinations on Avila, Adult, Eye, Letter and Spam with missing ratio 0.2, 0.4 and 0.6.

to set given 2 auxiliary tasks, and we perform grid search to report the best performance of hyper-parameter selection.

The results are shown in Table 2. We can observe that the multi-task learning is sometimes comparable to the best imputation-based model. Though there is still a gap between the knowledge transfer based methods. Nevertheless, such framework suffers the serious drawback as the polynomial growing of the hyper-parameter search space given more features missing.

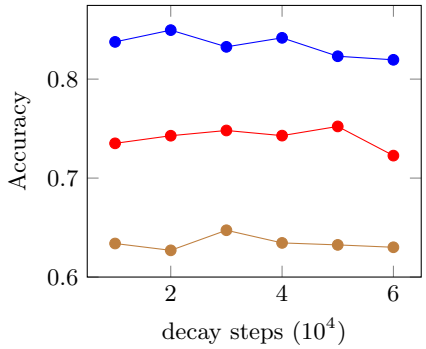
3.4 Comparison among different knowledge transferring strategy

We then perform experiments to compare among our proposed strategies: weight-decay mechanism, teacher-student framework and their combination. As shown in Fig. 5, In a nutshell, each strategy generates significant improvement and their combination can achieve the best performance in general.

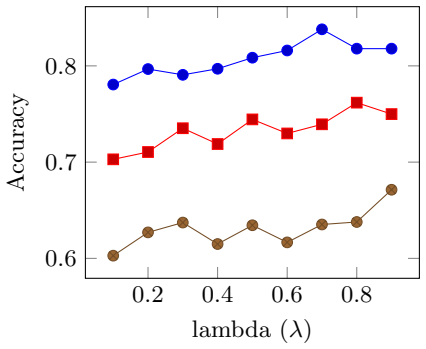
3.5 Discussions with hyper-parameter analyses

Here we provide some in-depth analysis of the proposed model.

Weight-decay Mechanism We analyze the proposed weight-decay mechanism by setting different decay steps. Remind that the model weights for unavailable



(a) Analysis of weight-decay mechanism with different decay steps on five datasets with missing ratio 0.2 (blue), 0.4 (red) and 0.6 (brown).



(b) Analysis of the teacher-student framework with different λ on five datasets with missing ratio 0.2 (blue), 0.4 (red) and 0.6 (brown).

Fig. 6: Hyper-parameter sensitivity analyses on Avila dataset.

features are gradually decayed to zero within decay steps. The number of decay steps is inversely proportional to the decay rate. We show the results in Fig. 6(a) of different decay steps in Avila dataset as demonstrations. The performance shows that the decay ratio is not critically sensitive, though tuning it up can still improve the performance.

Teacher-Student Framework Here are two hyper-parameters, the temperature τ and the weight $\lambda \in [0, 1]$ between the distillation loss and the student loss in our teacher-student framework. Empirically, when the student model is very small comparing to the teacher model, lower temperatures work better. Since our teacher model and student model share the same model architecture, we set τ to default value of 1. λ affects how much the knowledge is transferred from the teacher to the student. That is, if the λ is set to be large, the student will consider the information from the teacher more than the information from the ground truth data and vice versa. Note that if λ is set to 0, it is essentially the same as *Drop* method which only utilizes available features during training. We investigate the impact of teacher on student by setting different λ. As shown in Fig. 6(b), the best results are achieved at λ above 0.5 on the average, implying that the knowledge from the teacher plays an important role when missingness occurs.

4 Related Work

4.1 Missing Data Imputation

Data with missing values is a well-researched topic in statistics. Most of the early work on missing data, including definitions, multiple imputation and subsequent analysis is attributed to works of Little and Rubin [13].

Missing data can be categorized into three types: (1) the data is missing completely at random (MCAR) (there is no dependency on any of the variables), (2) the data is missing at random (MAR) with dependency on the observed variables, (3) the data is missing not at random (MNAR) with dependency on both observed variables and unobserved variables; thus, data missing cannot be fully accounted for by the observed variables.

The problem of unavailable test features can be the case of MNAR. One way to deal with the problem is by imputation. That is, we need to impute the values of the unavailable test features then predict on the test data with observed values and imputed values using the model trained on complete training data.

Current state-of-the-art imputation methods are based on discriminative and generative models. For example, GAIN is a method for imputing missing data by adapting the GAN framework. In GAIN, the generator's goal is to accurately impute missing data, and the discriminator's goal is to distinguish between observed and imputed components. The discriminator is trained to minimize the classification loss (when classifying which components were observed and which have been imputed), and the generator is trained to maximize the discriminator's misclassification rate. Thus, these two networks are trained using an adversarial process. To achieve this goal, GAIN builds on and adopts the standard GAN architecture. To ensure that the result of this adversarial process is the desired target, the GAIN architecture provides the discriminator with additional information in the form of "hints". This hinting ensures that the generator generates samples according to the true underlying data distribution. In GAIN, theoretical results are shown under the MCAR assumption.

On the other hand, MisGAN is a GAN-based framework for learning from incomplete data. The proposed framework learns a complete data generator along with a mask generator that models the missing data distribution. The masks are used to "mask" generated complete data by filling the indicated missing entries with a constant value. The complete data generator is trained so that the resulting masked data are indistinguishable from real incomplete data that are masked similarly. Specifically, they use two generator-discriminator pairs (G_m, D_m) and (G_x, D_x) for the masks and data respectively. In MisGAN, they focus on missing completely at random (MCAR) case, where the two generators are independent of each other and have their own noise distributions. Further, they equip the framework with an adversarially trained imputer to impute missing data. Although the work focus on the MCAR case, MisGAN can be easily extended to cases (e.g., MAR, MNAR) where the output of the data generator is provided to the mask generator.

The imputation approaches mentioned above have a common unifying theme: they mainly focus on MCAR case, where the missingness occurs completely at random. They attempt to learn an imputer adversarially to impute the missing values. Once the missing values are imputed, they can use the model trained with complete data to predict on test data with observed and imputed values. We refer the procedure as *two-stage* method: (1) impute missing values, (2) predict on data with observed and imputed values. However, the imputation methods

still suffer from the gap between the imputed values and the real values. Hence, the performance of final target task will also degrade to some extent when the missingness occurs.

In this paper, we propose a completely different approach from two-stage methods for the problem of unavailable test features: instead of focusing on the imputation of missing values, we directly consider the final target task and transfer the information of unavailable features to learn a robust and meaningful intermediate representations for final prediction.

4.2 Self-supervised Learning Methods

Self-supervised learning has opened up a huge opportunity to better utilize unlabeled data as it has been widely used in natural language processing [10] and computer vision [4, 9]. Self-supervised learning empowers us to exploit a variety of labels that come with the data. To make use of unlabeled data, one way is to set the learning objectives properly to get supervision from the data itself, which is why it is called *self-supervised learning*. With an self-supervised loss function from self-supervised task, we are aim to learn an intermediate representation with the expectation that this representation can carry good semantic or structural meanings and can be beneficial to a variety of practical downstream tasks. For example, we might rotate images at random and train a model to predict how each input image is rotated. Further, we expect the model to learn high-quality latent variables for real-world tasks, such as constructing an object recognition classifier with very few labeled samples.

On the other hand, we consider utilizing self-supervised learning techniques to tackle the problem of unavailable test features. We realize it by *multi-task learning*, in which the original task can serve as *target* task while *auxiliary* tasks are to predict the features which are unavailable at prediction phase. Note that the number of auxiliary tasks is proportional to the number of unavailable features for prediction. The goal is to learn a representation generally helpful for target task through multi-task learning. To the best of our knowledge, it is the first attempt to tackle the problem of unavailable test features via multi-task learning.

4.3 Knowledge Distillation

Knowledge distillation is a well-known method for model compression, in which a small model is trained to mimic a pre-trained, large model with respect to the model capacity. This training setting is sometimes referred to as *teacher-student*, where the large model is the teacher and the small model is the student. It was first pioneered by [1], and a generalized version was proposed by [7]. It has been successfully used in several applications such as object detection [3], acoustic models [15, 5] and natural language processing [8, 19].

The distillation method transfers knowledge from a pre-trained, cumbersome (teacher) model to a small (student) model by using the distribution of output

class probabilities produced by the teacher model as "soft labels" along with the standard ground-truth labels as "hard labels" for training the student model.

This method was originally developed to tackle the stringer constraints on computational resources. Therefore, it has a smaller number of parameters in the student model to reduce the usage of storage and computation. In our scenario of unavailable test features, the constraint is not the number of parameters but the completeness of input features. The model trained on complete training data can be treated as *teacher* as it gains complete knowledge of input features. The model trained on limited input features can be treated as *student*. Then we can utilize the knowledge distillation technique to transfer the knowledge, from the teacher network to the student network, in order to help the student learn as the same way as teacher under the constraint of incompleteness of input features.

5 Conclusion

In this paper, we proposed a novel and effective learning framework for the scenario of unavailable prediction features. Different from the common practices utilizing imputation-based solutions to firstly fill the data before conducting model prediction, we proposed a unified end-to-end, knowledge transfer-based solution to achieve better performances compared with not only conventional approaches but also the state-of-the-art methods. Future work includes more in-depth studies to understand the pros and cons between imputation-based and non-imputation solutions for prediction tasks suffering from incomplete data.

References

1. Buciluundefined, C., Caruana, R., Niculescu-Mizil, A.: Model compression. p. 535–541. KDD '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1150402.1150464
2. Buuren, S.v., Groothuis-Oudshoorn, K.: mice: Multivariate imputation by chained equations in r. Journal of statistical software pp. 1–68 (2010)
3. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30, pp. 742–751. Curran Associates, Inc. (2017)
4. Doersch, C., Zisserman, A.: Multi-task self-supervised visual learning. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
5. Fukuda, T., Suzuki, M., Kurata, G., Thomas, S., Cui, J., Ramabhadran, B.: Efficient knowledge distillation from an ensemble of teachers. In: Interspeech. pp. 3697–3701 (2017)
6. García-Laencina, P.J., Sancho-Gómez, J.L., Figueiras-Vidal, A.R.: Pattern classification with missing data: a review. Neural Computing and Applications **19**(2), 263–282 (2010)
7. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015)

8. Kim, Y., Rush, A.M.: Sequence-level knowledge distillation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 1317–1327. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1139
9. Komodakis, N., Gidaris, S.: Unsupervised representation learning by predicting image rotations. In: International Conference on Learning Representations (ICLR). Vancouver, Canada (Apr 2018), https://hal-enpc.archives-ouvertes.fr/hal-01832768
10. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. In: International Conference on Learning Representations (2020)
11. Li, S.C.X., Jiang, B., Marlin, B.: Learning from incomplete data with generative adversarial networks. In: International Conference on Learning Representations (2019)
12. Lichman, M.: UCI machine learning repository (2013), URL http://archive.ics.uci.edu/ml
13. Little, R.J., Rubin, D.B.: Statistical analysis with missing data, vol. 793. John Wiley & Sons (2019)
14. Liu, C.: Missing data imputation using the multivariate t distribution. Journal of Multivariate Analysis **53**(1), 139 – 158 (1995). https://doi.org/https://doi.org/10.1006/jmva.1995.1029
15. Lu, L., Guo, M., Renals, S.: Knowledge distillation for small-footprint highway networks. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4820–4824 (2017). https://doi.org/10.1109/ICASSP.2017.7953072
16. Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. Journal of Machine Learning Research **11**(80), 2287–2322 (2010)
17. Santos, M.S., Soares, J.P., Henriques Abreu, P., Araújo, H., Santos, J.: Influence of data distribution in missing data imputation. In: ten Teije, A., Popow, C., Holmes, J.H., Sacchi, L. (eds.) Artificial Intelligence in Medicine. pp. 285–294. Springer International Publishing, Cham (2017)
18. Stekhoven, D.J., Bühlmann, P.: MissForest—non-parametric missing value imputation for mixed-type data. Bioinformatics **28**(1), 112–118 (10 2011). https://doi.org/10.1093/bioinformatics/btr597
19. Sun, S., Cheng, Y., Gan, Z., Liu, J.: Patient knowledge distillation for BERT model compression. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 4323–4332. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1441
20. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. p. 1096–1103. ICML '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1390156.1390294
21. Yoon, J., Jordon, J., van der Schaar, M.: GAIN: Missing data imputation using generative adversarial nets. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5689–5698. PMLR, Stockholmsmässan, Stockholm Sweden (10–15 Jul 2018)