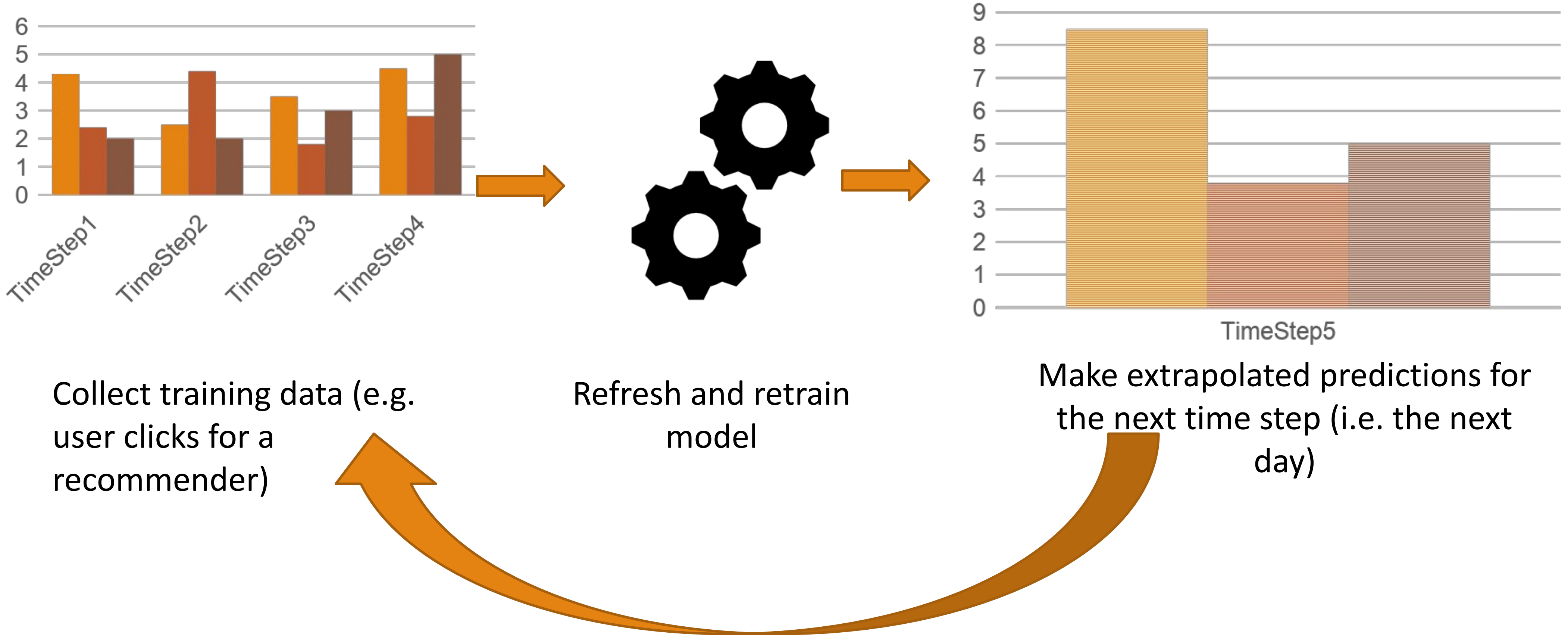
Training for the Future: A Simple Gradient Interpolation Loss to Generalize Along Time

Anshul Nasery*, Soumyadeep Thakur*, Vihari Piratla, Abir De, Sunita Sarawagi

IIT Bombay

* denotes Equal contribution

Modern ML Pipelines



Collect training data (e.g. user clicks for a recommender)

Refresh and retrain model

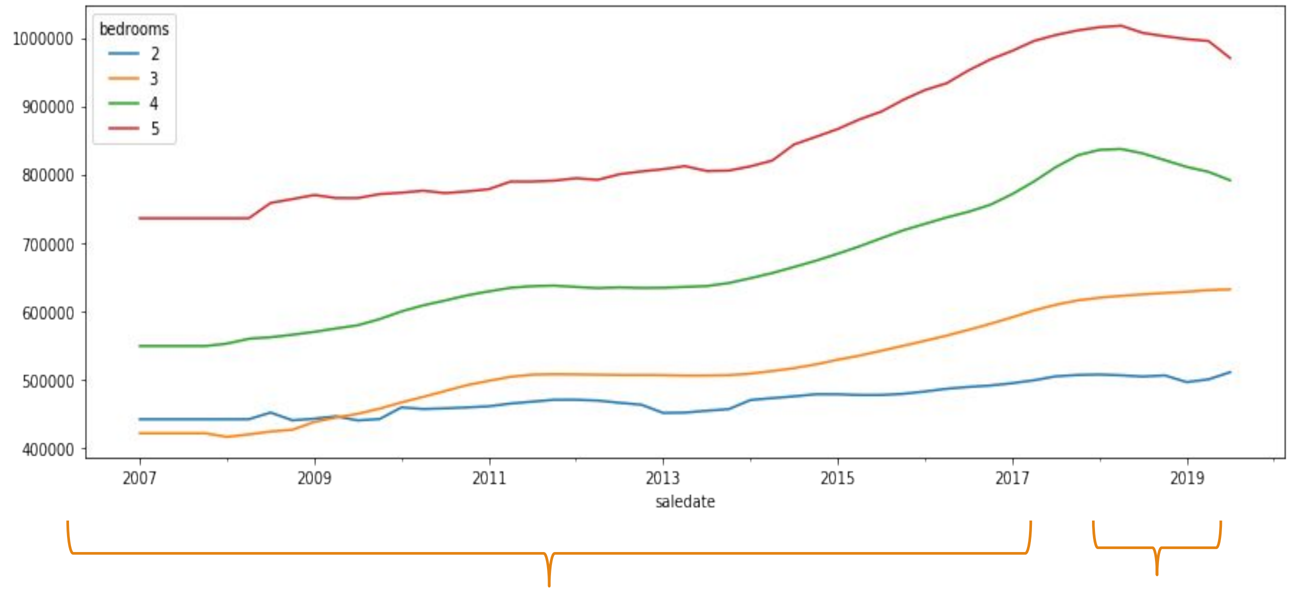
Make extrapolated predictions for the next time step (i.e. the next day)

Discard model and get new data

Modern ML Pipelines

Real world data often exhibits a temporal drift, making extrapolation challenging.

House Prices



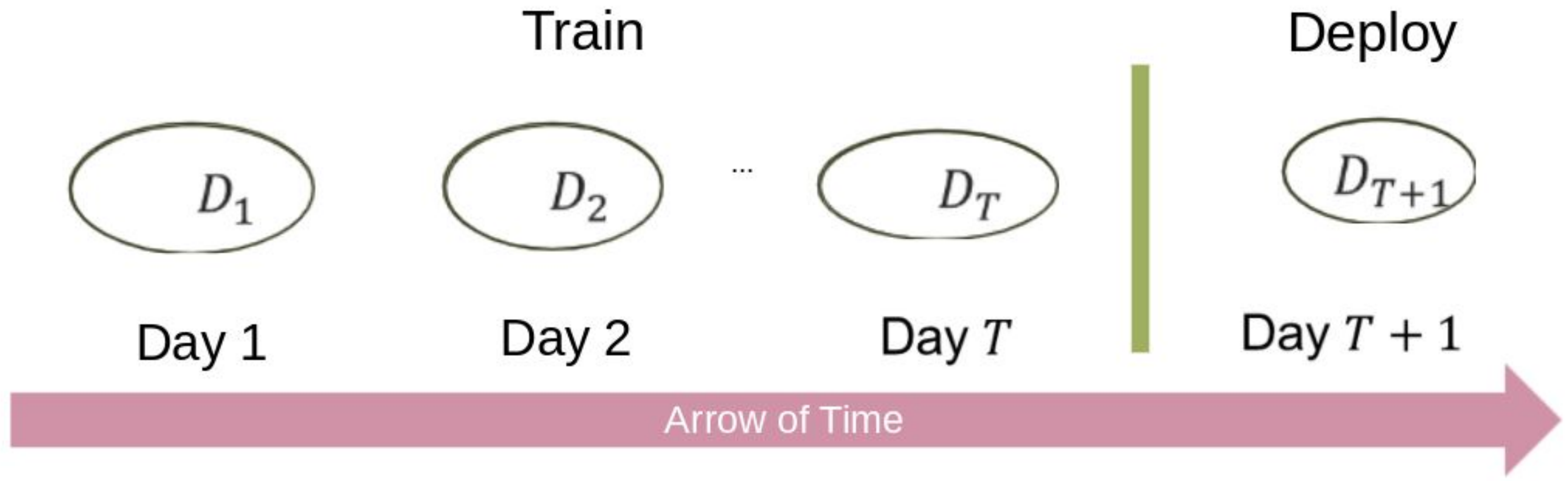
Train

Test

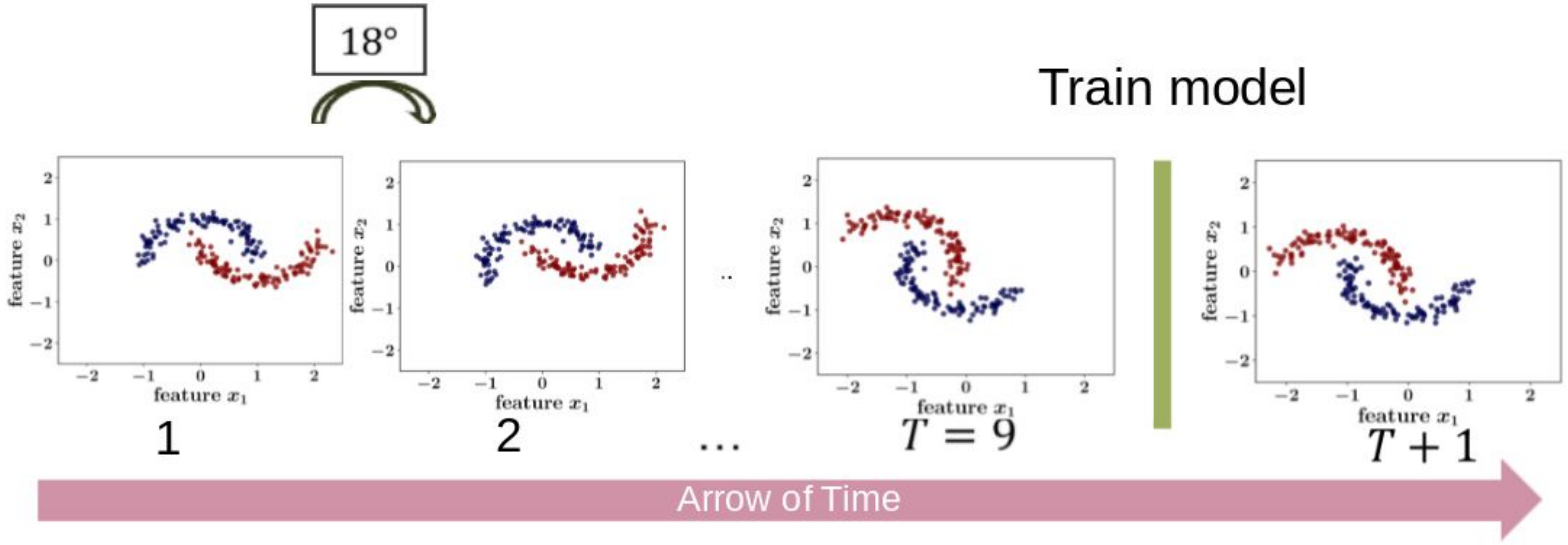
Training for the future

Given labelled data points from source domains, and **no data** from target domain in the immediate future

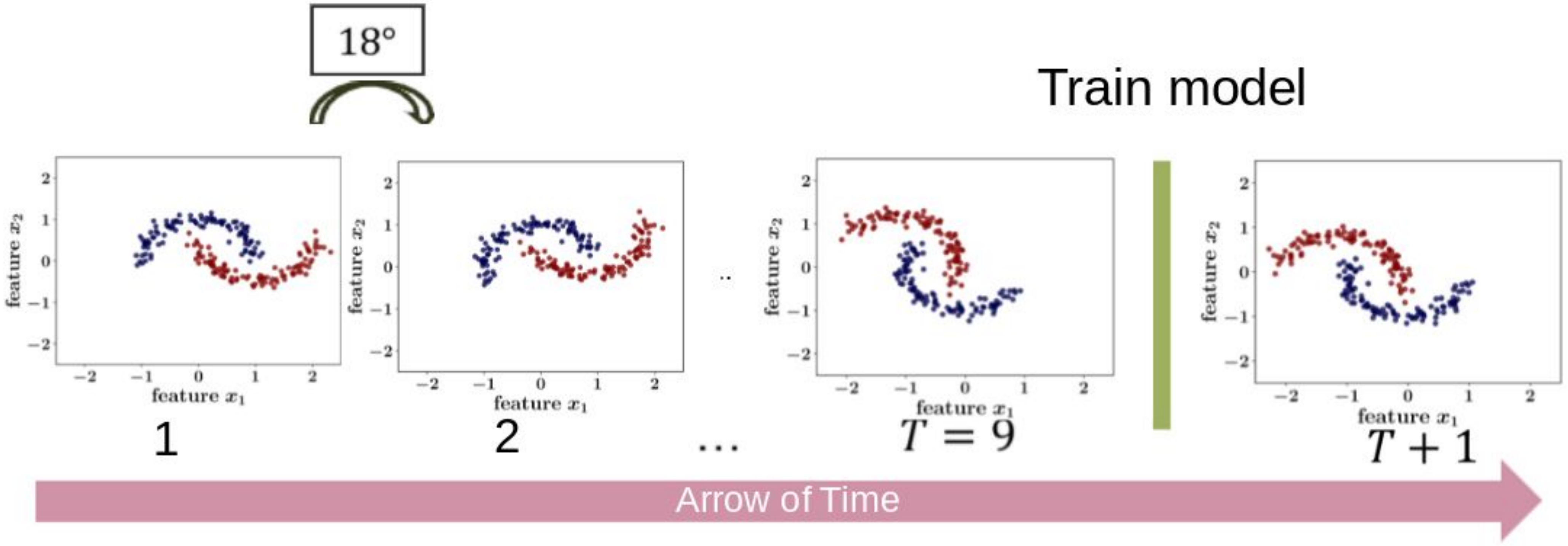
Goal: Achieve high accuracy on target domain



Example: Rotating two moons

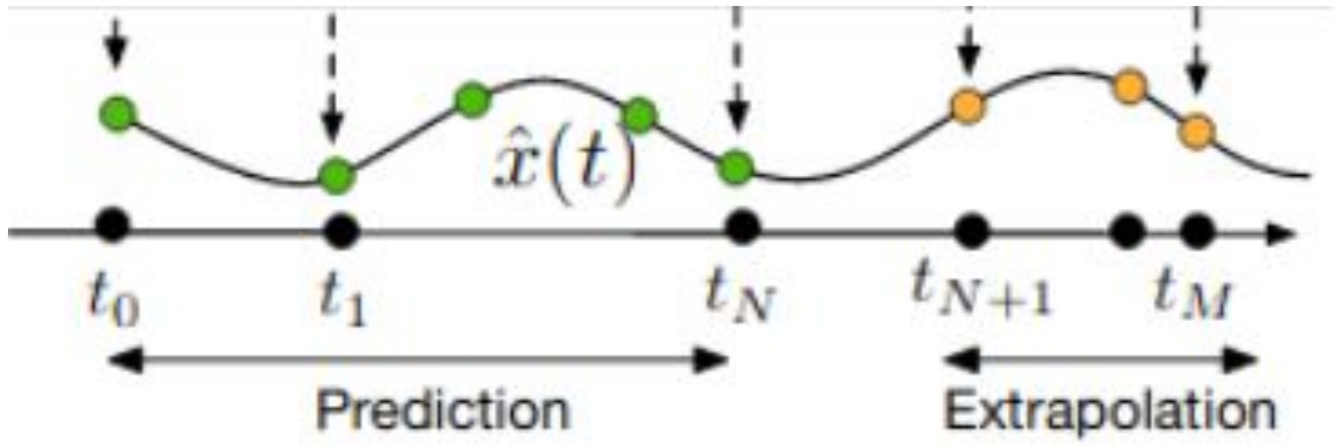


Example: Rotating two moons



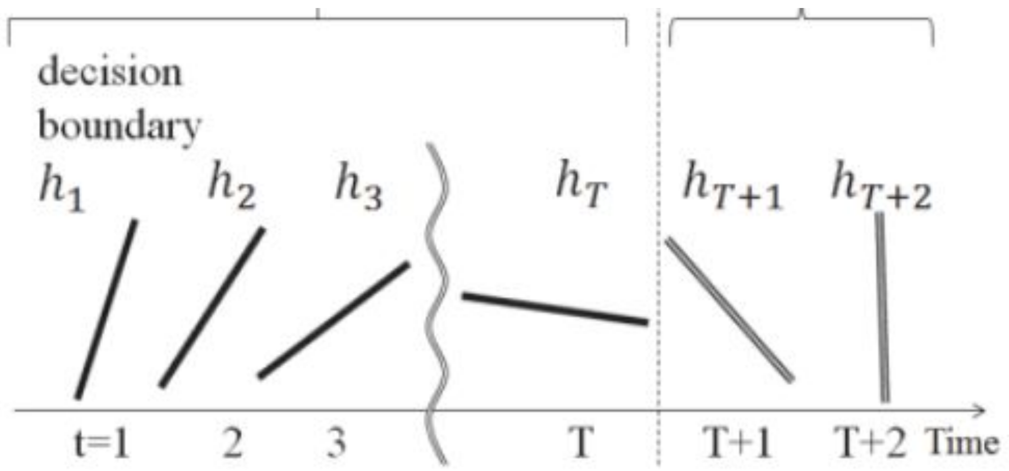
Training for the future

- Since we only care about performance at test time, the evaluation setup differs from online learning.
- We do not have the entire trajectories of datapoints through time, rendering time series methods difficult to apply



Training for the future

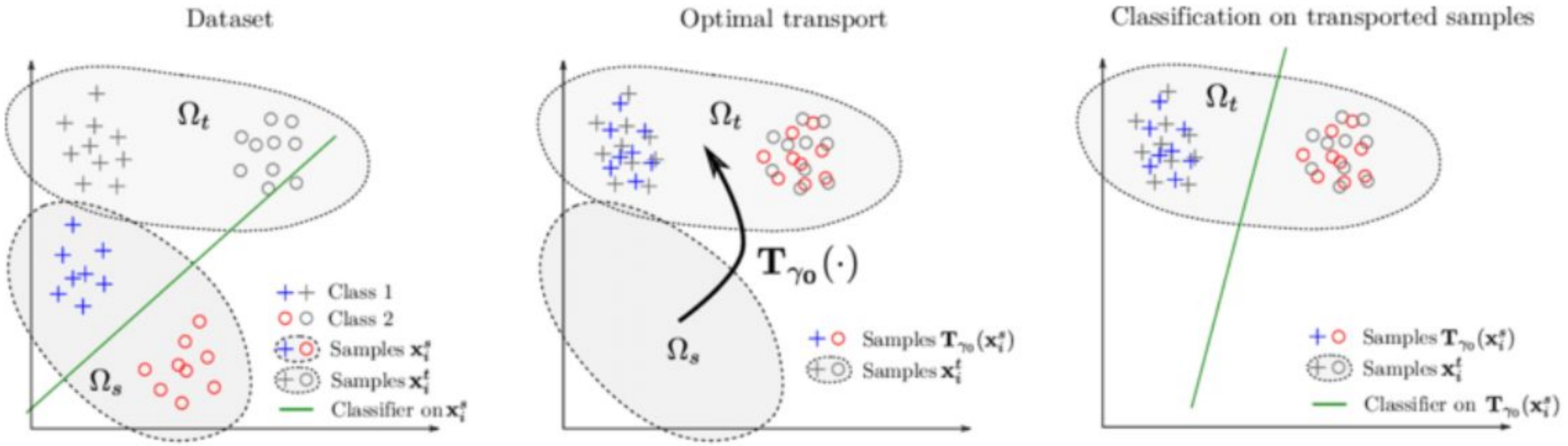
- Since we only care about performance at test time, the evaluation setup differs from online learning.
- We do not have the entire trajectories of datapoints through time, rendering time series methods difficult to apply.
- Our NN model hence has to infer the shifting decision boundary and extrapolate it to the near future.



Related Work

Our setting is closely related to continuous domain adaptation and predictive domain adaptation.

One class of methods for this problem tries to transform source data to target time using Optimal transport[1], or kernel embeddings[2,3]. However these often need unlabelled target data.

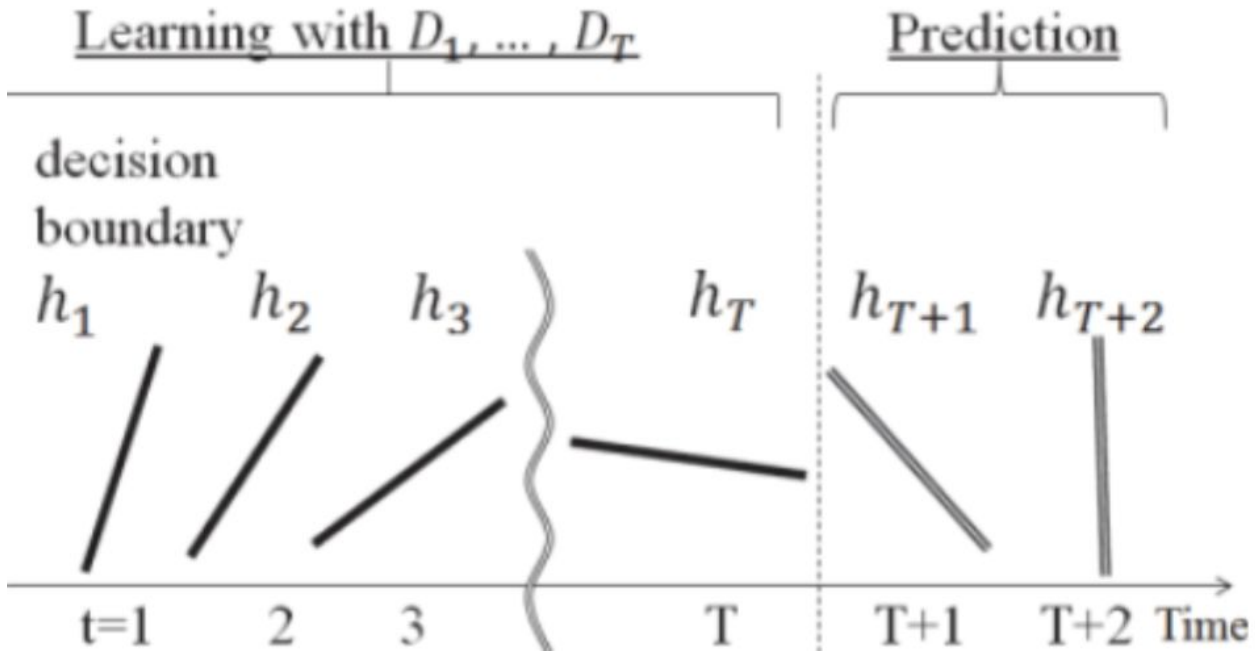


Dataset
Ω_t
Ω_s
Class 1
Class 2
Samples $\mathbf{x}_i^s$
Samples $\mathbf{x}_i^t$
Classifier on $\mathbf{x}_i^s$
Optimal transport
Ω_t
$\mathbf{T}_{\gamma_0}(\cdot)$
Ω_s
Samples $\mathbf{T}_{\gamma_0}(\mathbf{x}_i^s)$
Samples $\mathbf{x}_i^t$
Classification on transported samples
Ω_t
Samples $\mathbf{T}_{\gamma_0}(\mathbf{x}_i^s)$
Samples $\mathbf{x}_i^t$
Classifier on $\mathbf{T}_{\gamma_0}(\mathbf{x}_i^s)$

[1] – Ortiz-Jiménez, Guillermo, et al. "Forward-backward splitting for optimal transport based problems." *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020
[2] – Yang, Yongxin, and Timothy M. Hospedales. "Multivariate regression on the grassmannian for predicting novel domains." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
[3] – Lampert, Christoph H. "Predicting the future behavior of a time-varying probability distribution." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Related Work

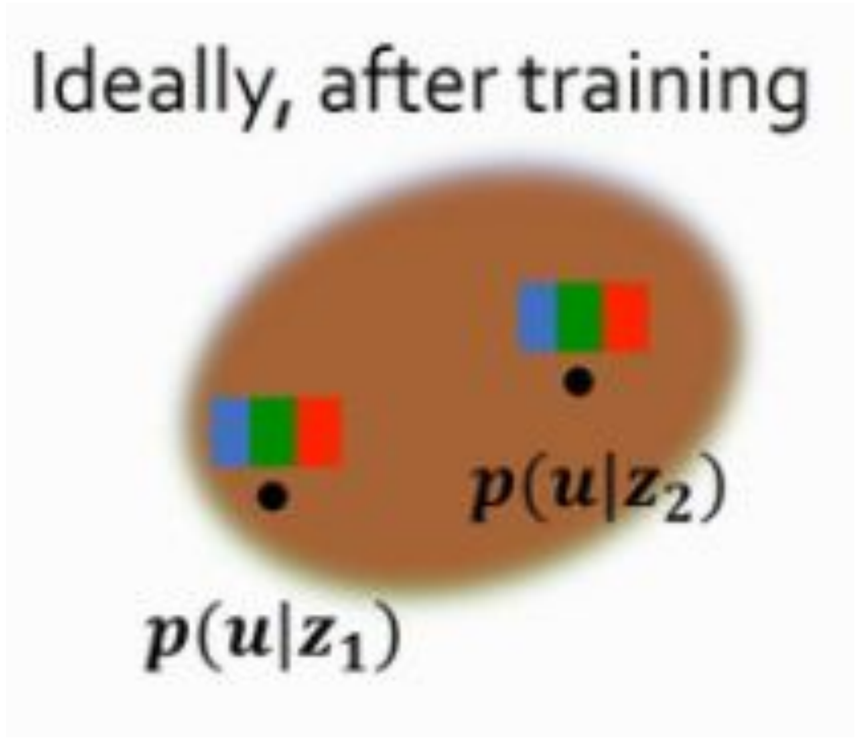
Another set of approaches treat model parameters as a function of time, using Gaussian Process based smoothness on decision boundaries[5,6], or kernel smoothing on time sensitive NN parameters[4].



[4] – Mancini, Massimiliano, et al. "Adagraph: Unifying predictive and continuous domain adaptation through graphs." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
[5] – Kumagai, Atsutoshi, and Tomoharu Iwata. "Learning non-linear dynamics of decision boundaries for maintaining classification performance." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. No. 1. 2017.
[6] – Kumagai, Atsutoshi, and Tomoharu Iwata. "Learning future classifiers without additional data." *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

Related Work

A third kind of approach aims to learn time invariant representations for examples in an adversarial manner[7].

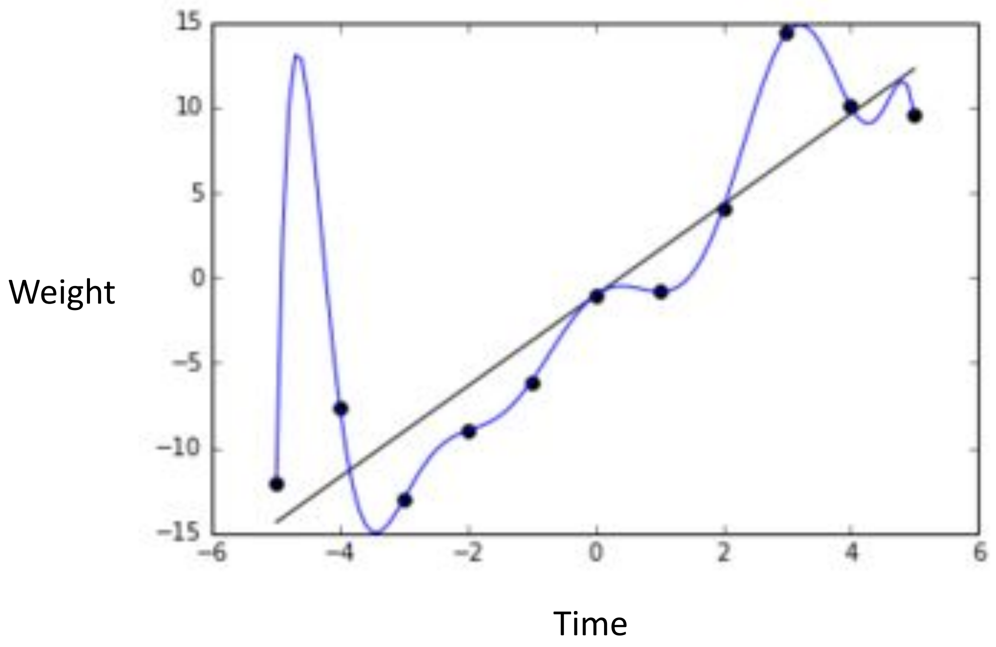


[7] – Wang, Hao, Hao He, and Dina Katabi. "Continuously indexed domain adaptation." *arXiv preprint arXiv:2007.01807* (2020).

Our Contributions

Our method is based on three key insights -

1. Need to have a time sensitive architecture
2. Need to somehow provide supervision on time stamps from the future
3. Need to regularize the temporal complexity of the learnt function



Easy to overfit the temporal trend with overparameterized neural nets!

Our Contributions

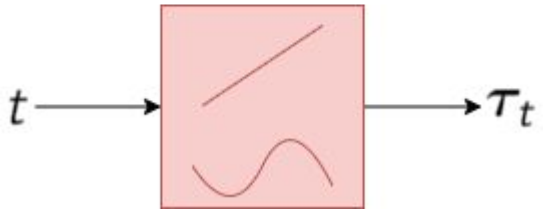
Our method is based on three key insights -

1. Need to have a time sensitive architecture
2. Need to somehow provide supervision on time stamps from the future
3. Need to regularize the temporal complexity of the learnt function

Time Sensitive Network

We use the Time2Vec [8] - captures complex dependencies such as periodicity

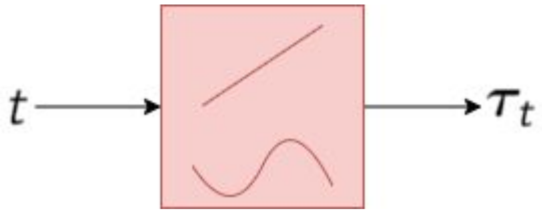
$$\tau_{\mathbf{t}}[\mathbf{a}] = \begin{cases} \omega_a t + b_a & 1 \leq a \leq m_p \\ \sin(\omega_a t + b_a) & m_p \leq a \leq m \end{cases}$$



Time Sensitive Network

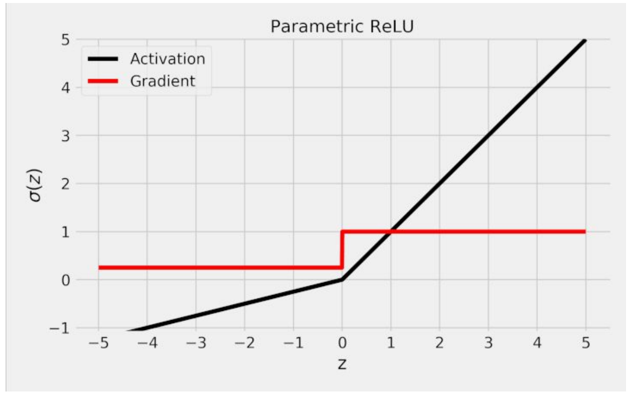
We use the Time2Vec [8] - captures complex dependencies such as periodicity

$$\tau_{\mathbf{t}}[\mathbf{a}] = \begin{cases} \omega_a t + b_a & 1 \leq a \leq m_p \\ \sin(\omega_a t + b_a) & m_p \leq a \leq m \end{cases}$$

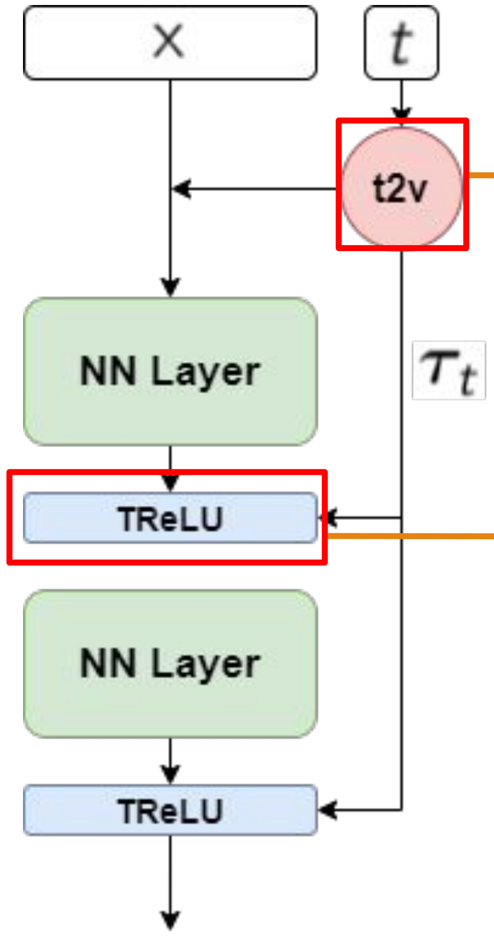


A novel time dependent leaky ReLU whose threshold and slope is computed by neural-nets taking time as input

$$\mathrm{TReLU}_{\phi}(\mathbf{x}, t) = \begin{cases} \mathbf{h}_{\phi}(\tau_t) \odot (\mathbf{x} - \mathbf{g}_{\phi}(\tau_t)) + \mathbf{v}(\tau_{\mathbf{t}}) & \mathbf{x} < \mathbf{g}_{\phi}(\tau_t) \\ \mathbf{x} & \mathbf{x} \geq \mathbf{g}_{\phi}(\tau_t) \end{cases}$$



Time Sensitive Network



$$\tau_{\mathbf{t}}[\mathbf{a}] = \begin{cases} \omega_a t + b_a & 1 \leq a \leq m_p \\ \sin\left(\omega_a t + b_a\right) & m_p \leq a \leq m \end{cases}$$

$$\mathrm{TReLU}_{\phi}(\mathbf{x}, t) = \begin{cases} \mathbf{h}_{\phi}(\tau_t) \odot \left(\mathbf{x} - \mathbf{g}_{\phi}(\tau_t)\right) + \mathbf{v}(\tau_{\mathbf{t}}) & \mathbf{x} < \mathbf{g}_{\phi}(\tau_t) \\ \mathbf{x} & \mathbf{x} \geq \mathbf{g}_{\phi}(\tau_t) \end{cases}$$

Our Contributions

Our method is based on three key insights -

1. Need to have a time sensitive architecture
2. Need to somehow provide supervision on time stamps from the future
3. Need to regularize the temporal complexity of the learnt function

Gradient Interpolation

Despite using a time-sensitive architecture, ERM may overfit on $\mathcal{D}^1 \ \dots \ \mathcal{D}^{\mathcal{T}}$ - since there is no relation or constraint between the predictions of the network on the different time stamps

GI Loss:

$$\mathcal{J}_{GI}(y;\, F_\theta(\mathbf{x},\, t)) \;=\; l(y;\, F_\theta(\mathbf{x},\, t)) \;+\; \lambda\, l\left(y;\, F_\theta(\mathbf{x},\, t-\delta) \;+\; \delta\, \frac{\partial F_\theta\,(\mathbf{x}, t-\delta)}{\partial t}\right)$$

Prediction loss

Prediction loss on interpolated logits

Gradient Interpolation

Despite using a time-sensitive architecture, ERM may overfit on $\mathcal{D}^1 \ldots \mathcal{D}^\mathcal{T}$ - since there is no relation or constraint between the predictions of the network on the different time stamps

GI Loss:

$$\mathcal{J}_{GI}(y;\, F_\theta(\mathbf{x},\, t)) \;=\; l(y;\, F_\theta(\mathbf{x},\, t)) \,+\, \lambda\, l\left(y;\, F_\theta(\mathbf{x},\, t-\delta) \,+\, \delta\, \frac{\partial F_\theta\,(\mathbf{x}, t-\delta)}{\partial t}\right)$$

Prediction loss

Prediction loss on interpolated logits

The second term provides "supervision" on nearby time steps and encourages smoother functions.

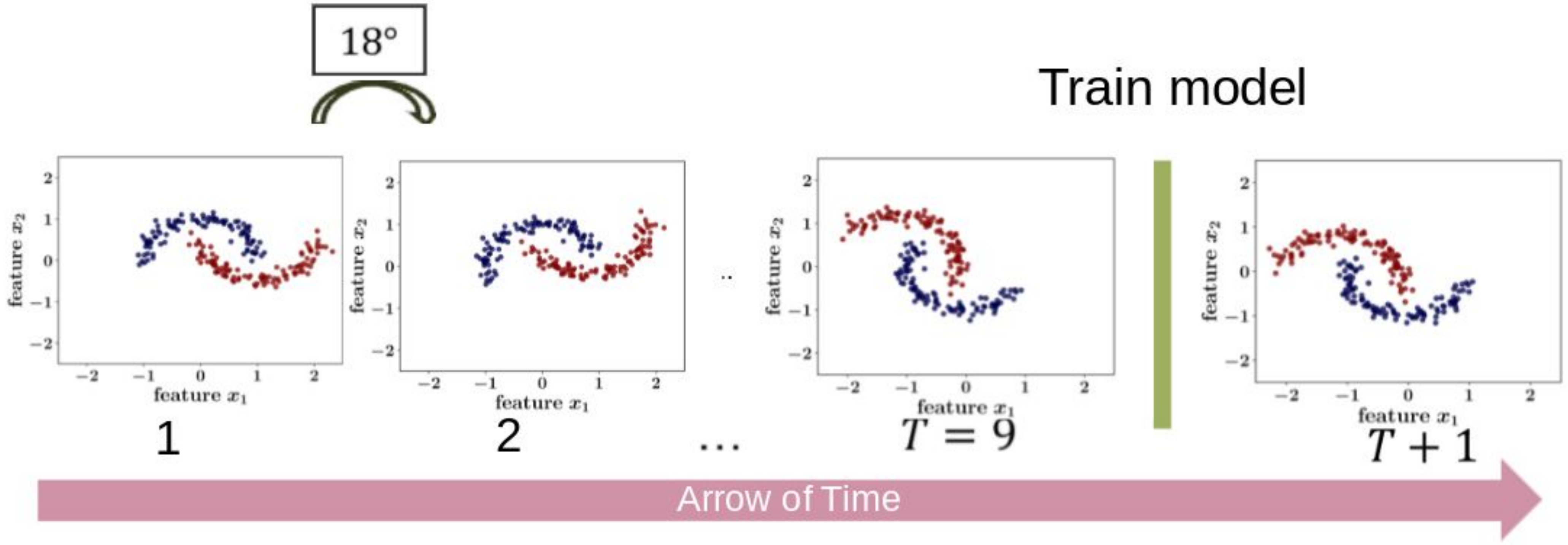
A negative δ also encourages extrapolation

δ is chosen adversarially in a window of [-Δ , Δ]

Training Algorithm

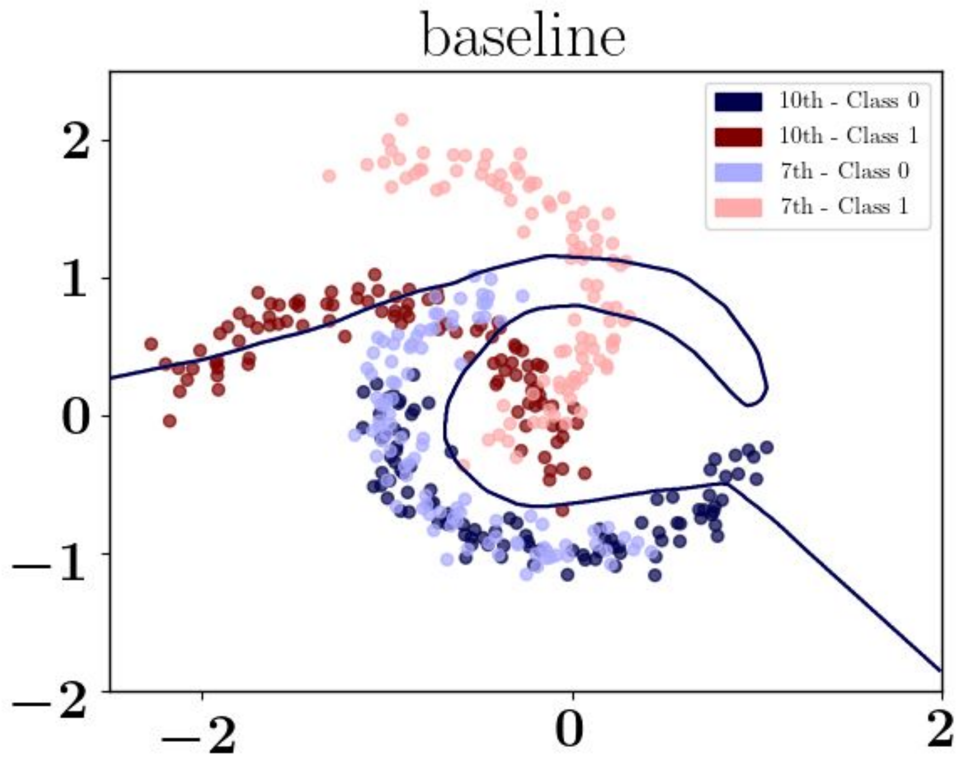
1. Pre-train time sensitive network on train domains with ERM
2. Finetune using GI loss-
   a. For each minibatch, sample δ uniformly between [-Δ,Δ].
   b. Compute δ by doing k steps of gradient ascent on GI loss
   c. Clamp δ in a window of [-Δ,Δ]
   d. Use the value of δ obtained in the above step to compute GI loss for model.
   e. Minimize GI loss obtained above through a single gradient descent step.

Synthetic Data: Rotating two moons

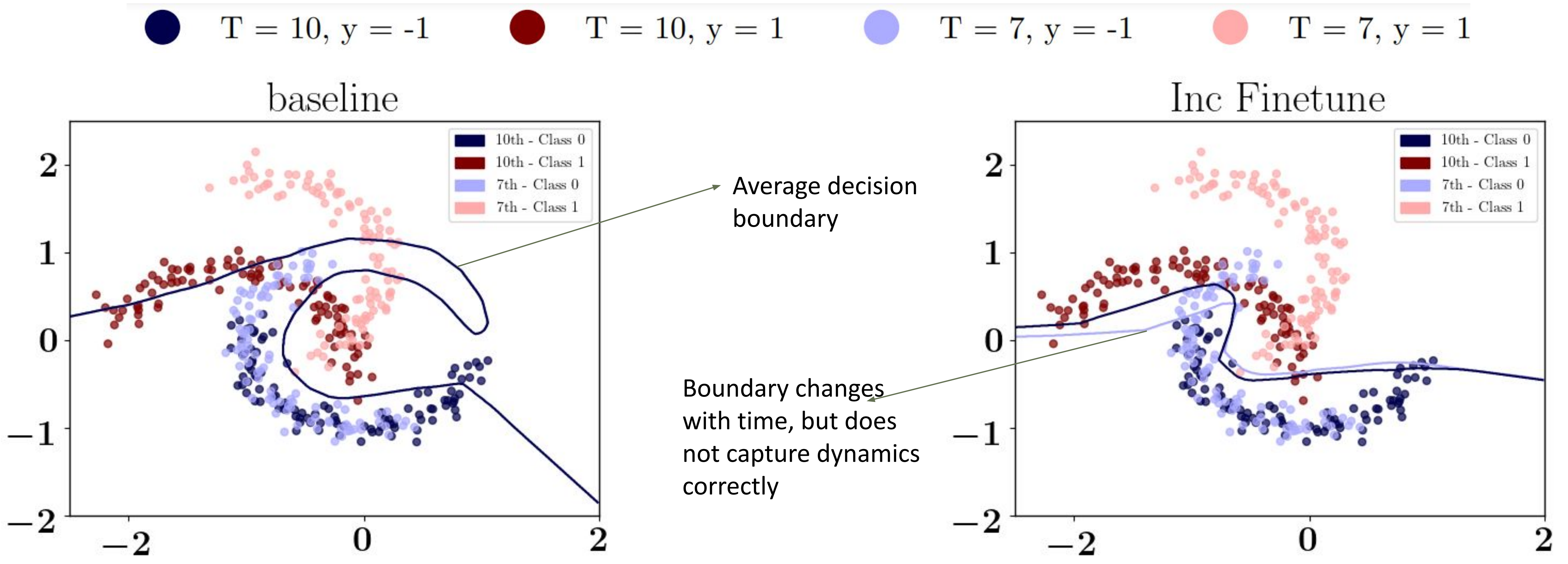


Need of time sensitivity

● T = 10, y = -1 ● T = 10, y = 1 ● T = 7, y = -1 ● T = 7, y = 1



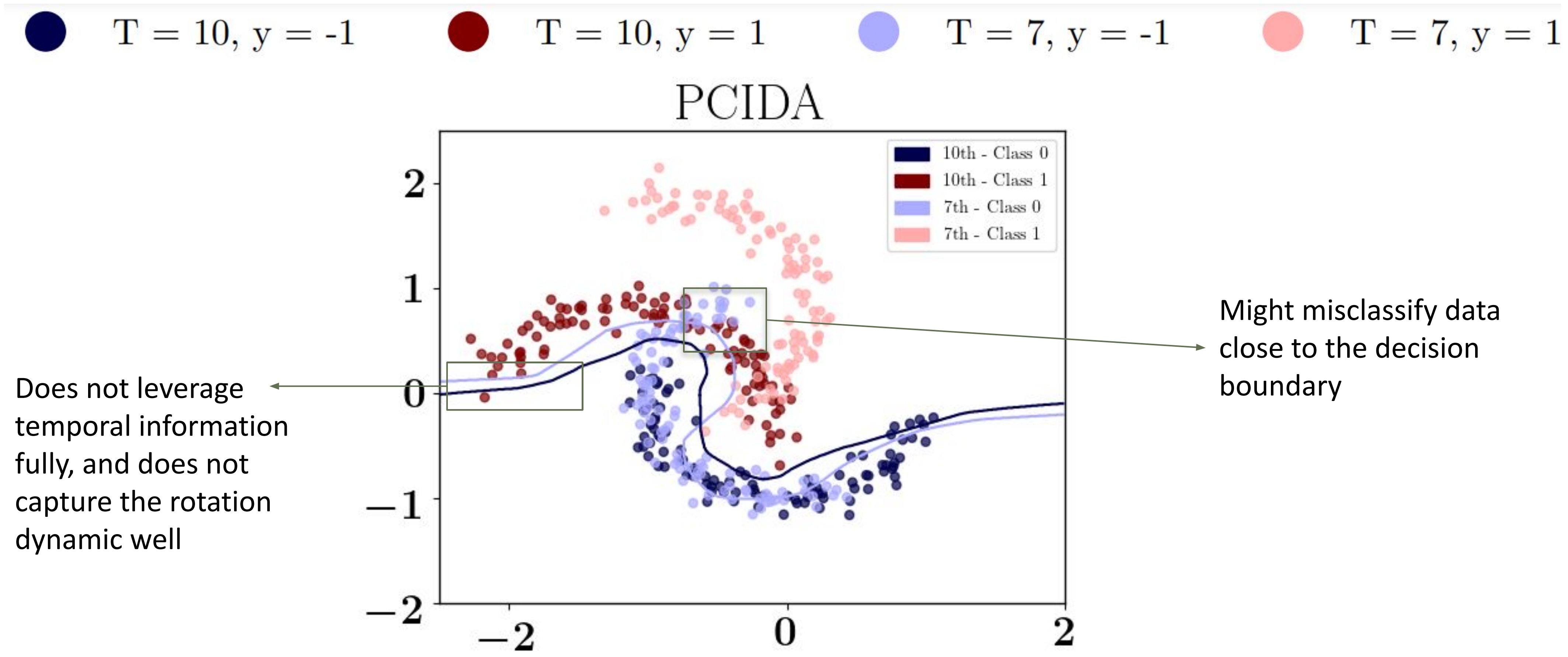
Need of time sensitivity



Average decision boundary

Boundary changes with time, but does not capture dynamics correctly

Can time invariant representations work?



Does not leverage temporal information fully, and does not capture the rotation dynamic well

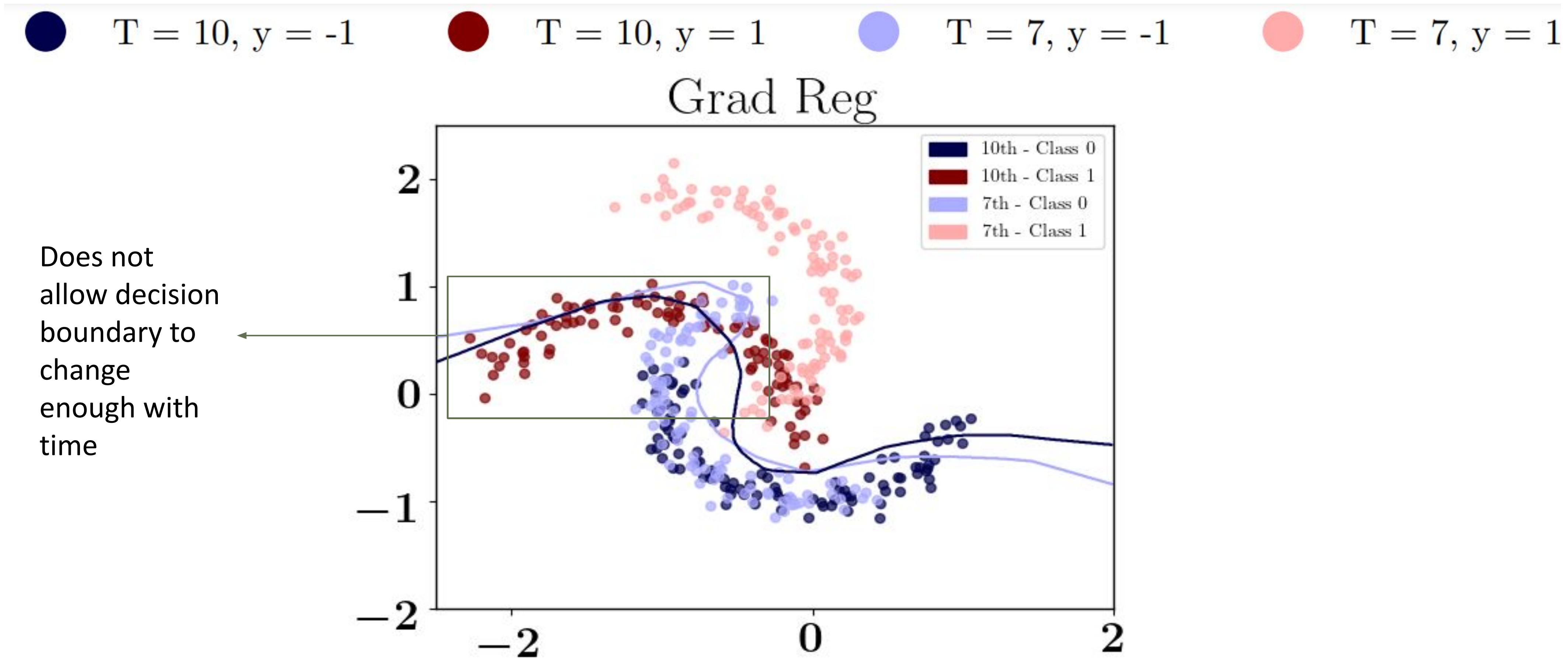
Might misclassify data close to the decision boundary

Can regularizing the gradient work?

- GI smooths out the *$w_j(t)$* values to have a lower curvature
- Not equivalent to regularizing the gradients w.r.t time, i.e.

$$\mathcal{J}_{GR}(y;\, F_\theta(\mathbf{x},\, t)) \,=\, l(y;\, F_\theta(\mathbf{x},\, t)) \,+\, \lambda \left|\left| \frac{\partial F_\theta(\mathbf{x},\, t)}{\partial t} \right|\right|_2^2$$

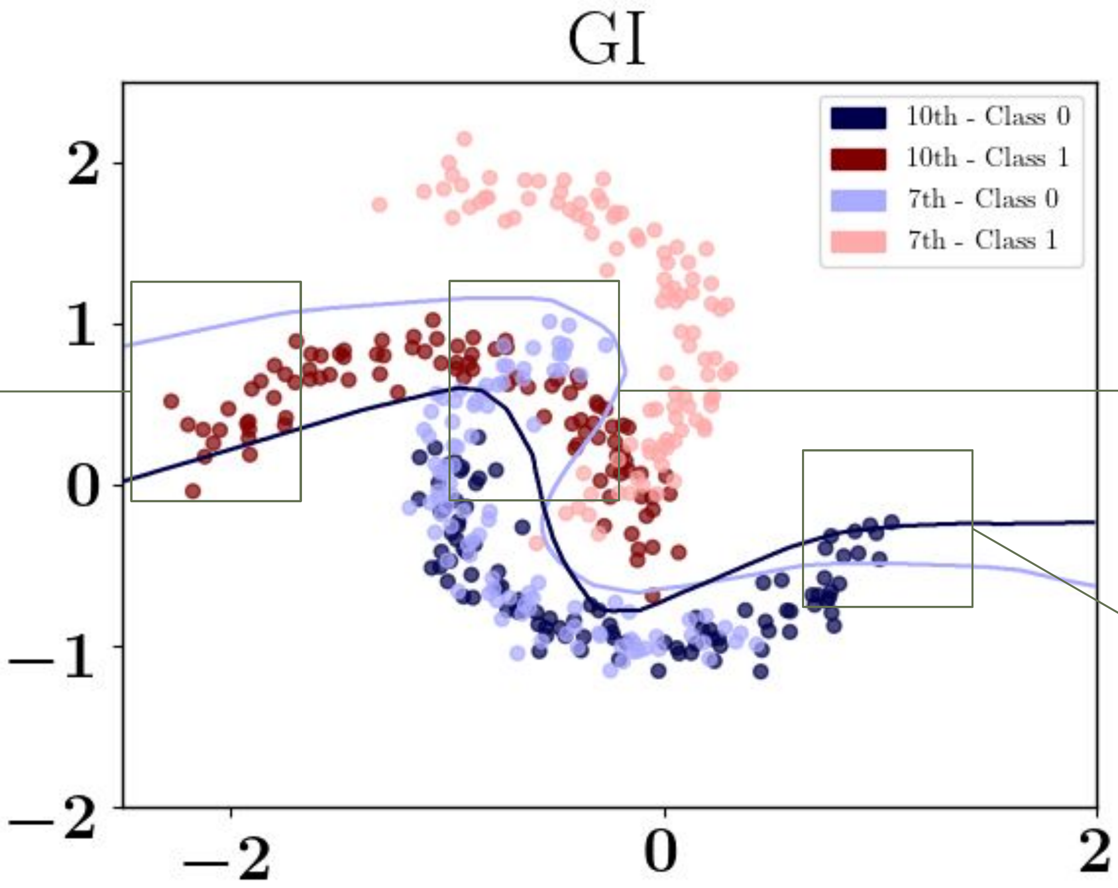
Can regularizing the gradient work?



Does not allow decision boundary to change enough with time

Our method

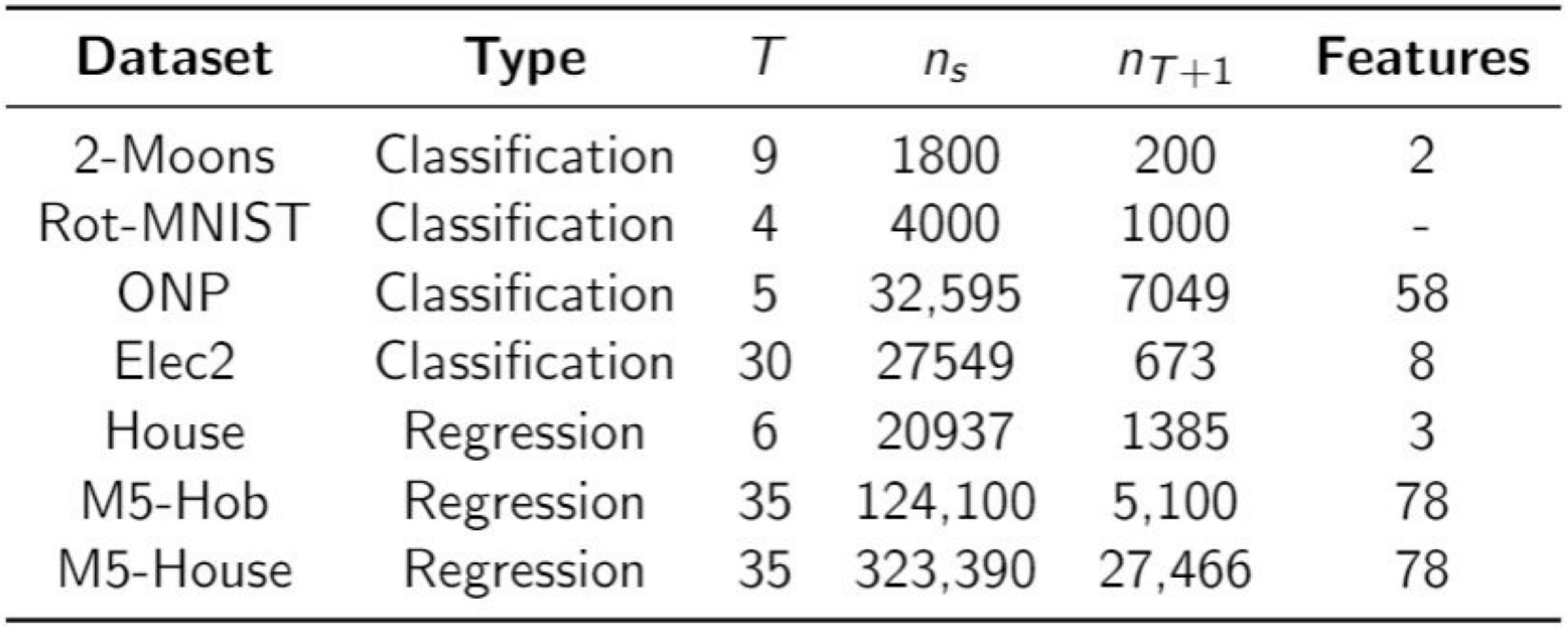
Captures temporal dependencies of the decision boundary



Learns rotation by fitting appropriate functions at all times

Data aware regularization of time varying decision boundary

Experiments – Datasets

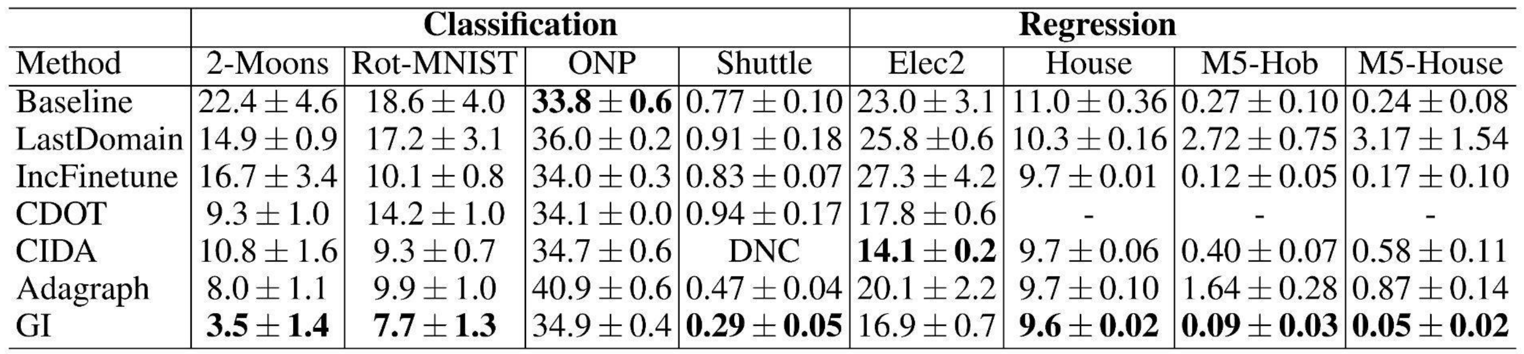


Dataset	**Type**	T	n_s	n_{T+1}	**Features**
2-Moons	Classification	9	1800	200	2
Rot-MNIST	Classification	4	4000	1000	-
ONP	Classification	5	32,595	7049	58
Elec2	Classification	30	27549	673	8
House	Regression	6	20937	1385	3
M5-Hob	Regression	35	124,100	5,100	78
M5-House	Regression	35	323,390	27,466	78

Experiments - Baselines

- ERM: Train a time-insensitive neural net on source data only
- LastDomain: Train a time-insensitive neural net on last source domain only
- IncFinetune: ERM on 1st source domain, finetune on subsequent domains
- Transformation based method - CDOT [1]
- Domain invariant representations - CIDA [8]
- Time sensitive network - Adagraph [4]

Experimental Results



	Classification				Regression			
Method	2-Moons	Rot-MNIST	ONP	Shuttle	Elec2	House	M5-Hob	M5-House
Baseline	22.4 ± 4.6	18.6 ± 4.0	$\mathbf{33.8 \pm 0.6}$	0.77 ± 0.10	23.0 ± 3.1	11.0 ± 0.36	0.27 ± 0.10	0.24 ± 0.08
LastDomain	14.9 ± 0.9	17.2 ± 3.1	36.0 ± 0.2	0.91 ± 0.18	25.8 ± 0.6	10.3 ± 0.16	2.72 ± 0.75	3.17 ± 1.54
IncFinetune	16.7 ± 3.4	10.1 ± 0.8	34.0 ± 0.3	0.83 ± 0.07	27.3 ± 4.2	9.7 ± 0.01	0.12 ± 0.05	0.17 ± 0.10
CDOT	9.3 ± 1.0	14.2 ± 1.0	34.1 ± 0.0	0.94 ± 0.17	17.8 ± 0.6	-	-	-
CIDA	10.8 ± 1.6	9.3 ± 0.7	34.7 ± 0.6	DNC	$\mathbf{14.1 \pm 0.2}$	9.7 ± 0.06	0.40 ± 0.07	0.58 ± 0.11
Adagraph	8.0 ± 1.1	9.9 ± 1.0	40.9 ± 0.6	0.47 ± 0.04	20.1 ± 2.2	9.7 ± 0.10	1.64 ± 0.28	0.87 ± 0.14
GI	$\mathbf{3.5 \pm 1.4}$	$\mathbf{7.7 \pm 1.3}$	34.9 ± 0.4	$\mathbf{0.29 \pm 0.05}$	16.9 ± 0.7	$\mathbf{9.6 \pm 0.02}$	$\mathbf{0.09 \pm 0.03}$	$\mathbf{0.05 \pm 0.02}$

Table 1: Comparison of our proposed method against existing methods on all the seven datasets in terms of misclasssication error (in %)for first four datasets and mean absolute error (MAE) for last three datasets. The standard deviation over five runs follows the ± mark. We observe that GI outperforms almost all the baselines.

- More data helps and heuristically choosing a suffix of available data may be sub-optimal

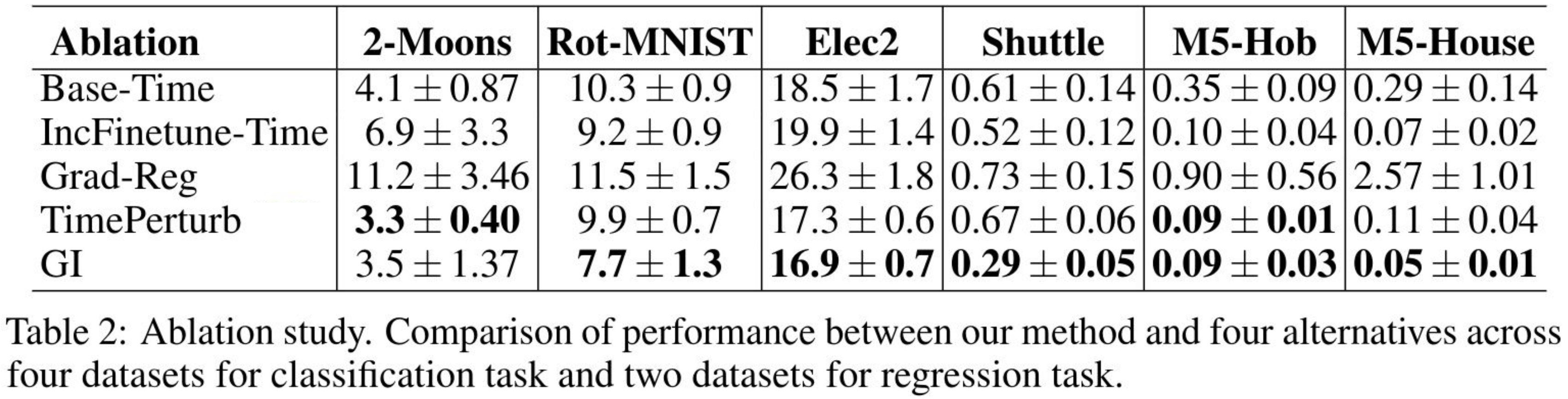
- Just incremental fine-tuning where most recent data is seen last by the model is often a strong baseline.
- The CIDA method that creates time-invariant representations shows improvements but in two cases it is worse than incremental fine-tune.

Experimental Results - Ablations

How does GI finetuning compare with other methods?

Experimental Results - Ablations

How does GI finetuning compare with other methods?



Ablation	2-Moons	Rot-MNIST	Elec2	Shuttle	M5-Hob	M5-House
Base-Time	4.1 ± 0.87	10.3 ± 0.9	18.5 ± 1.7	0.61 ± 0.14	0.35 ± 0.09	0.29 ± 0.14
IncFinetune-Time	6.9 ± 3.3	9.2 ± 0.9	19.9 ± 1.4	0.52 ± 0.12	0.10 ± 0.04	0.07 ± 0.02
Grad-Reg	11.2 ± 3.46	11.5 ± 1.5	26.3 ± 1.8	0.73 ± 0.15	0.90 ± 0.56	2.57 ± 1.01
TimePerturb	$\mathbf{3.3 \pm 0.40}$	9.9 ± 0.7	17.3 ± 0.6	0.67 ± 0.06	$\mathbf{0.09 \pm 0.01}$	0.11 ± 0.04
GI	3.5 ± 1.37	$\mathbf{7.7 \pm 1.3}$	$\mathbf{16.9 \pm 0.7}$	$\mathbf{0.29 \pm 0.05}$	$\mathbf{0.09 \pm 0.03}$	$\mathbf{0.05 \pm 0.01}$

Table 2: Ablation study. Comparison of performance between our method and four alternatives across four datasets for classification task and two datasets for regression task.

Base-Time: ERM loss with time-sensitive network

GradReg: Gradient Regularization

IncFinetune-Time: Incrementally finetune on preceeding domains with time-sensitive network

TimePerturb: Make time-sensitive network robust to time

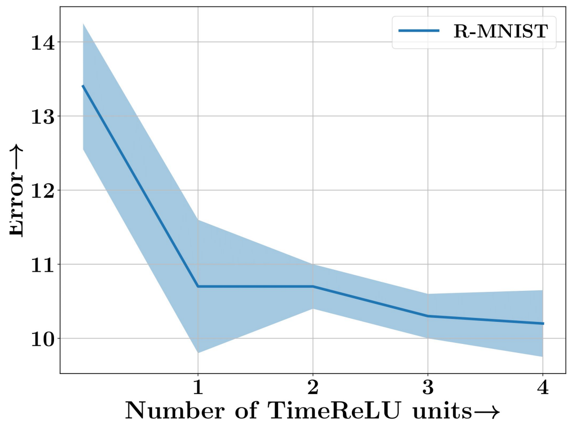
Experimental Results - Ablations

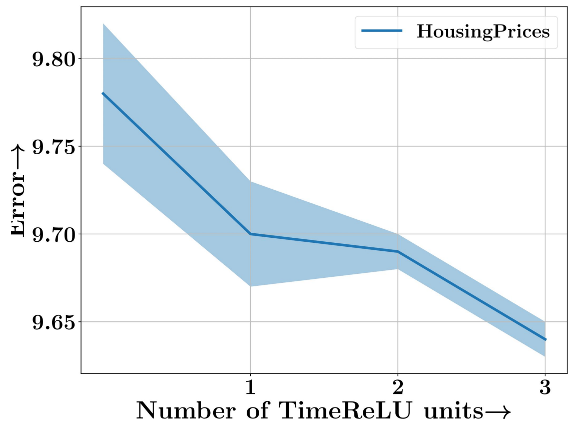
How well does TReLU work?

Experimental Results - Ablations

How well does TReLU work?

- Start with a deep network with only ReLU activations
- Incrementally change the activations to TReLU starting from the deepest layer





Even a single TReLU gives significant gains

Summary

- We proposed a time sensitive architecture and a gradient interpolated loss to enable models to extrapolate to future data.
- Through extensive empirical evaluation on real world and synthetic datasets, we demonstrate the efficacy of each component of our method over existing baselines.
- Future work would include reducing the time spent on computing δ, since it is the main bottleneck in our method.