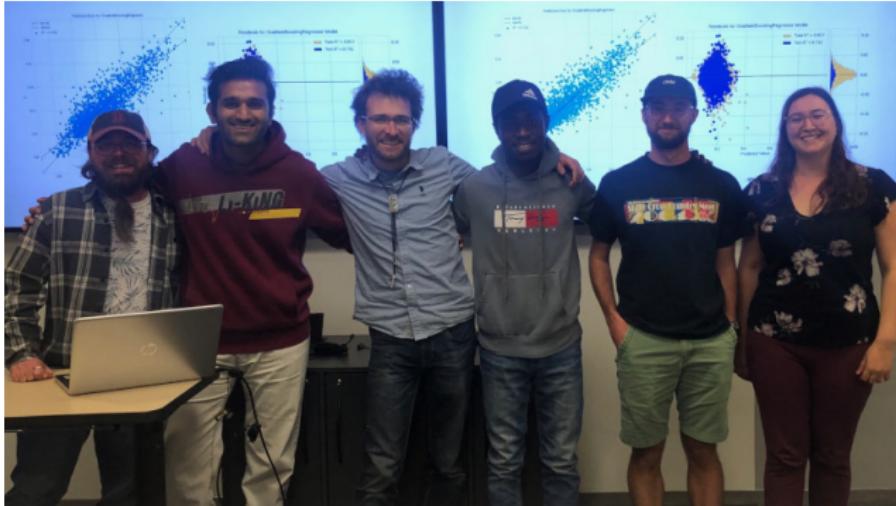


# Two new algorithms for scientific applications of machine learning

Toby Dylan Hocking — [toby.dylan.hocking@usherbrooke.ca](mailto:toby.dylan.hocking@usherbrooke.ca)

Learning Algorithms, Statistical Software, Optimization  
(LASSO Lab) — <https://lassolab.org>

Département d'Informatique, Université de Sherbrooke



Introduction: two common questions in collaborations involving applications of machine learning

SOAK: Same/Other/All K-fold cross-validation for estimating similarity of patterns in data subsets (arXiv:2410.08643)

AUM: Area Under Min(FPR,FNR), a new differentiable loss for ROC curve optimization (JMLR'23)

## Learning two different functions using two data sets

Figure from chapter by Hocking TD, *Introduction to machine learning and neural networks* for book *Land Carbon Cycle Modeling: Matrix Approach, Data Assimilation, and Ecological Forecasting* edited by Luo Y (Taylor and Francis, 2022).

Learning      Train      Learned      Predictions  
Algorithm      data      function      on test data

Learn()  $\rightarrow g$        $g(\text{ } \square \text{ }) = 0$   
 $g(\text{ } \blacksquare \text{ }) = 1$   
 $g(\text{ } \blacksquare \text{ }) = 1$

Learn()  $\rightarrow h$        $h(\text{ } \blacksquare \text{ }) = 0$   
 $h(\text{ } \blacksquare \text{ }) = 0$   
 $h(\text{ } \blacksquare \text{ }) = 1$

**Learn** is a learning algorithm, which outputs  $g$  and  $h$ .



Q: what happens if you do  $g(\text{ } \blacksquare \text{ })$ , or  $h(\text{ } \blacksquare \text{ })$ ?

## Train on one subset and accurately predict on another?

- ▶ This is a question about **generalization**: how accurate is the learned function on a new/test data subset which is **qualitatively different** in some respect?
- ▶ “Very accurate” if test data are similar enough to train data (best case is i.i.d. = independent and identically distributed)



- ▶ What if you do  $g(\text{[image]})$ , or  $h(\text{[image]})$ ? (**different pattern**)
- ▶ Predicting childhood autism (Lindly *et al.*), train on one year of surveys, test on another. (**different time periods**)
- ▶ Predicting carbon emissions (Aslam *et al.*), train on one city, test on another. (**different geographic regions**)
- ▶ Predicting presence of trees/burn in satellite imagery (Shenkin *et al.*, Thibault *et al.*), train on one geographic area/image, test on another. (**different geographic regions**)
- ▶ Predicting fish spawning habitat in sonar imagery (Bodine *et al.*), train on one river, test on another. (**geographic regions**)
- ▶ But how do we check if “very accurate” in each situation?

## How to deal with class imbalance?

- ▶ In binary classification, standard learning algorithms can yield sub-optimal prediction accuracy if train data have imbalanced labels.
- ▶ Predicting childhood autism (Lindly *et al.*), 3% autism, 97% not.
- ▶ Predicting presence of trees/burn in satellite imagery (Shenkin *et al.*, Thibault *et al.*), small percent of trees in deserts of Arizona, small percent of burned area out of total forested area in Quebec.
- ▶ Predicting fish spawning habitat in sonar imagery (Bodine *et al.*), small percent of suitable spawning habitat, out of total river bed.
- ▶ How do we adapt our learning algorithm, to handle the class imbalance?

Introduction: two common questions in collaborations involving applications of machine learning

SOAK: Same/Other/All K-fold cross-validation for estimating similarity of patterns in data subsets (arXiv:2410.08643)

AUM: Area Under Min(FPR,FNR), a new differentiable loss for ROC curve optimization (JMLR'23)

## Example data with subsets: predicting childhood autism

- ▶ Collaboration with Lindly *et al.*
- ▶ Downloaded National Survey of Children's Health (NSCH) data, years 2019 and 2020, from  
<http://www2.census.gov/programs-surveys/nsch>
- ▶ One row per person ( $N = 46,010$  rows), one column per survey question ( $D = 366$  columns).
- ▶ One column is diagnosis with Autism (binary classification, yes or no), can we predict it using the others?
- ▶ 18,202 rows for 2019; 27,808 rows for 2020.

Proposed SOAK algorithm is a generalization of standard K-fold cross-validation, that can be used to answer two questions:

- ▶ Can we train on one year, and accurately predict on another?
- ▶ Can we get a more accurate model by combining data from different years?

# K-fold cross-validation: a standard algorithm used to estimate prediction accuracy in machine learning

- ▶  $K = 3$  folds shown in figure below, meaning three different models trained, and three different prediction/test accuracy rates computed.
- ▶ It is important to use several train/test splits, so we can see if there are statistically significant differences between algorithms.



Hocking TD *Intro. to machine learning and neural networks* (2022).

## Proposed SOAK algorithm (Autism data example)

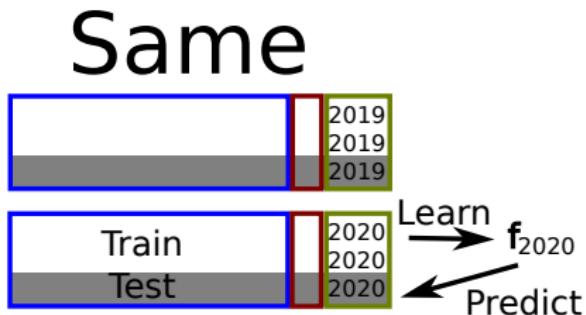
- ▶ Example: childhood autism prediction data set.
- ▶ Subsets of interest are years, which can be represented by adding a new column to the data table.

Inputs	Autism	Year
Questions	D	
1	0	2019
1	0	2019
1	1	2019
1	0	2020
1	1	2020
1	1	2020

# Proposed SOAK algorithm (Autism data example)

- ▶ Train subset same as test (=regular  $K$ -fold CV on 2020).
- ▶ Can we get a more accurate model than this baseline Same model? (if we train on Other/All years)

Inputs Questions	D	Autism	Year
1			
	0	2019	
	0	2019	
	1	2019	
	0	2020	
	1	2020	
	1	2020	



# Proposed SOAK algorithm (Autism data example)

- ▶ Test subset fixed (2020), train on other subset/year (2019).
- ▶ Can we train on one year, and accurately predict on another?  
Compare Same/Other test error.

People	Inputs Questions	D	Autism	Year
			1	N
1			0	2019
			0	2019
			1	2019
			0	2020
			1	2020
			1	2020

## Other

Train	Test	2019	2020
		2019	2019
		2019	2020



## Same

Train	Test	2019	2020
		2019	2020
		2019	2020

Learn  $\rightarrow f_{2020}$

Predict

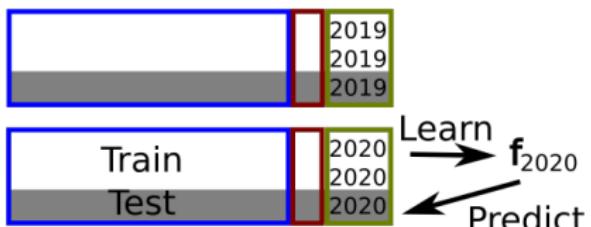
- Compare Same and Other error when predicting on test subset (2020)
- If subsets are similar, (2019 data are useful for learning how to predict in 2020) then the larger subset (Same or Other) should have smaller test error

# Proposed SOAK algorithm (Autism data example)

- ▶ Train set includes data from both subsets/years (2019, 2020).
- ▶ Can we get a more accurate model by combining data from different years? Compare Same/All test error.

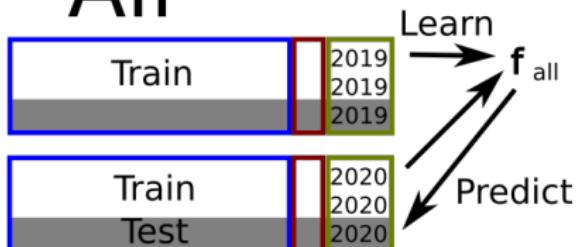
People	Inputs Questions	D	Autism	Year
			1	N
1			0	2019
			0	2019
			1	2019
			0	2020
			1	2020
			1	2020

## Same



- Compare Same and All error when predicting on test subset (2020)
- If subsets are similar, (2019+2020 data are useful for learning how to predict in 2020) then All test error should be less than Same (more relevant data available)

## All



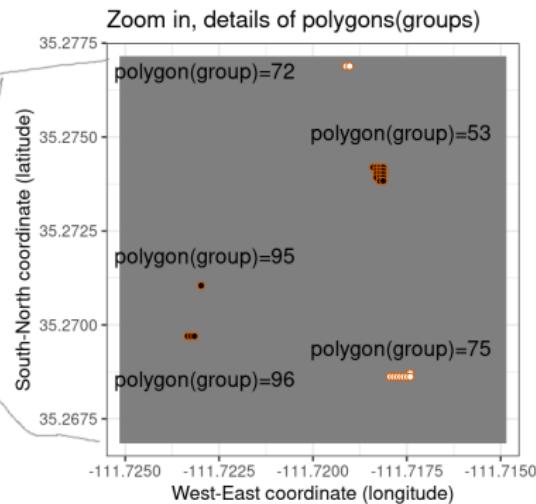
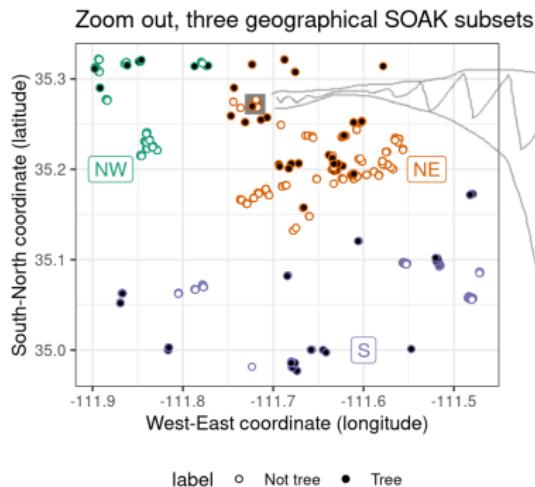
# Proposed SOAK algorithm (generic data)

- ▶ Key new idea is **subset** column in data table.
- ▶ Example:  $K = 3$  folds, two subsets (A/B).
- ▶ Compute test error for each fold (1/2/3) and subset (A/B).



# Proposed SOAK algorithm new to ML frameworks

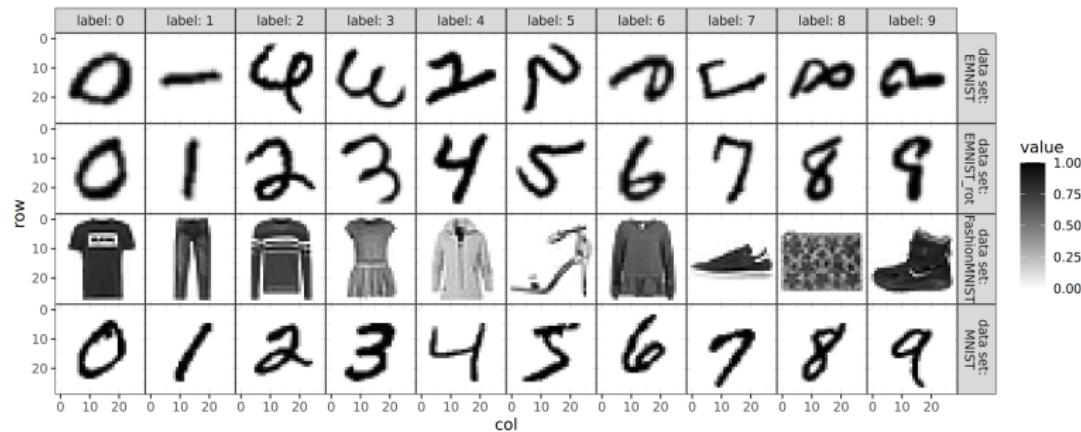
- ▶ ML frameworks like scikit-learn and mlr3 implement cross-validation with **groups** of samples that must stay together when splitting.
- ▶ For example (below), satellite image segmentation, trees vs background, Shenkin *et al.*: samples=pixels, grouped by polygon, SOAK subsets are geographical regions (NW, NE, S).
- ▶ SOAK: good predictions on one test **subset**, after training on Same/Other/All subsets? (can use together with groups)



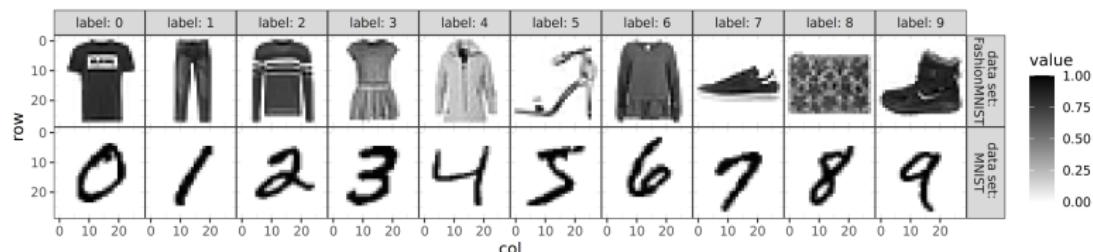
# ImagePair data: train on MNIST and accurately predict on MNIST variants?

Recall: what happens if you do  $g(\text{boot})$ , or  $h(\text{o})$ ?

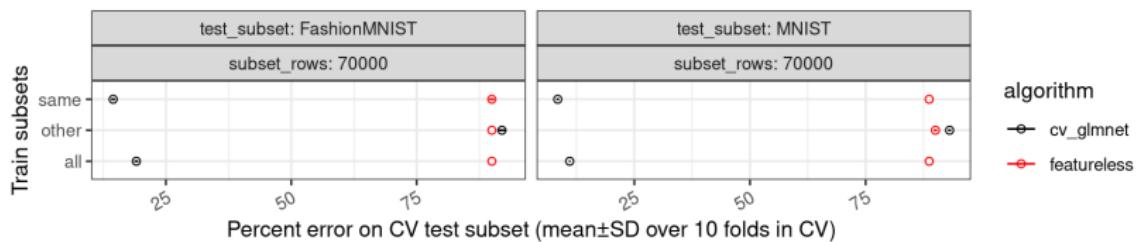
- ▶ Create three data sets by combining MNIST with variants: EMNIST, EMNIST\_rot, FashionMNIST.
- ▶ Each of these three new data sets has two subsets: MNIST/variant.



# SOAK for MNIST+FashionMNIST data

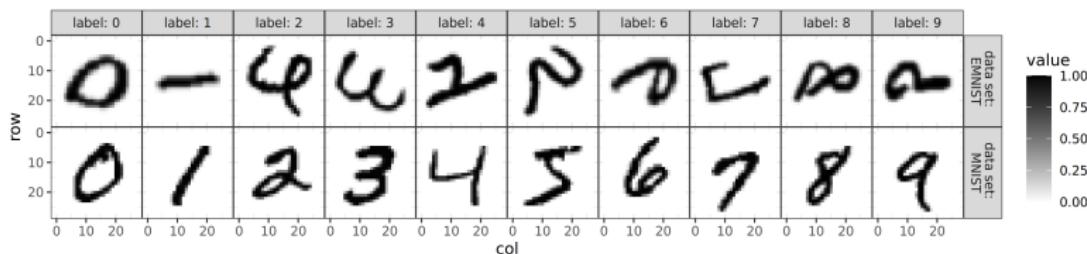


Data set: MNIST\_FashionMNIST

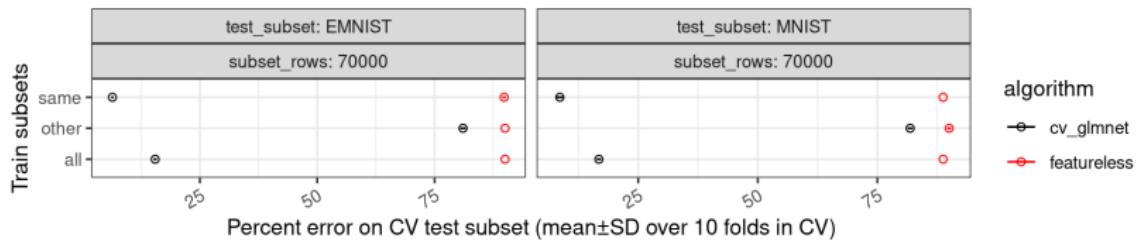


- ▶ **cv\_glmnet** is L1-regularized linear model.
- ▶ **featureless** baseline always predicts most frequent class.
- ▶ **Other cv\_glmnet** has greater test error than **featureless**, which indicates that the patterns are too different for the linear model to learn anything useful for prediction.

# SOAK for MNIST+EMNIST data

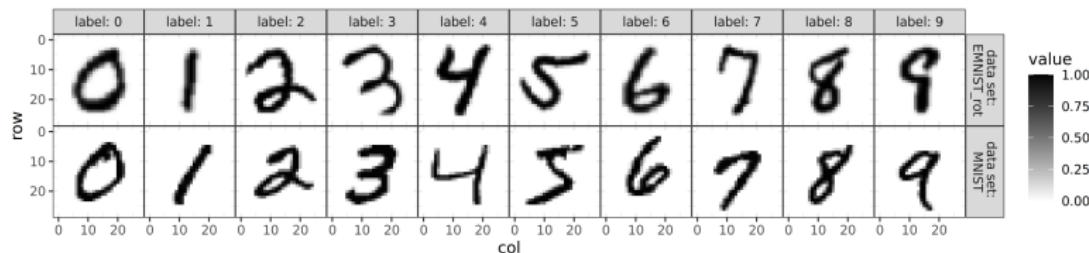


Data set: MNIST\_EMNIST

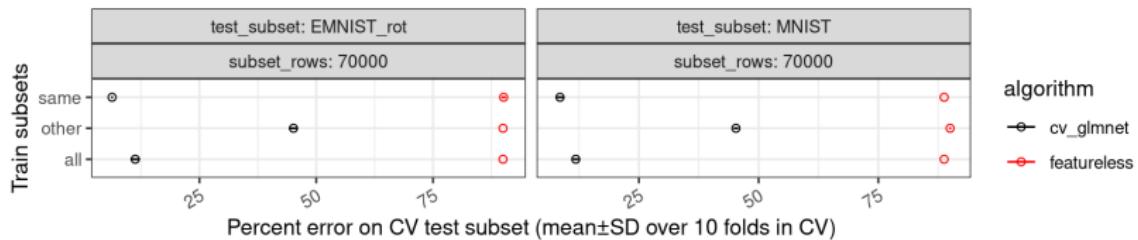


- ▶ **Other cv\_glmnet** has significantly smaller test error than **featureless**, so something is learned/transferable between data sets, but it is still clear that the pattern is very different.

# SOAK for MNIST+EMNIST\_rot data



Data set: MNIST\_EMNIST\_rot

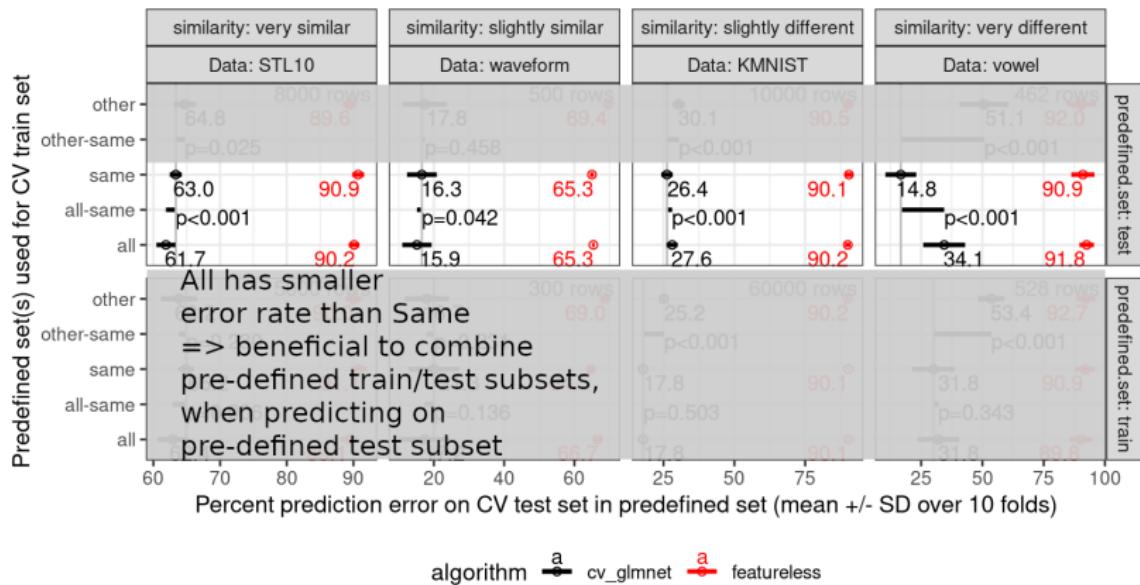


- ▶ **Other cv\_glmnet** has even smaller test error, indicating even more similarity between MNIST and EMNIST\_rot data sets.
- ▶ But **Other/All** test error are still not as small as **Same**, indicating there is a significant difference between learnable/predictable patterns in MNIST/EMNIST\_rot data.

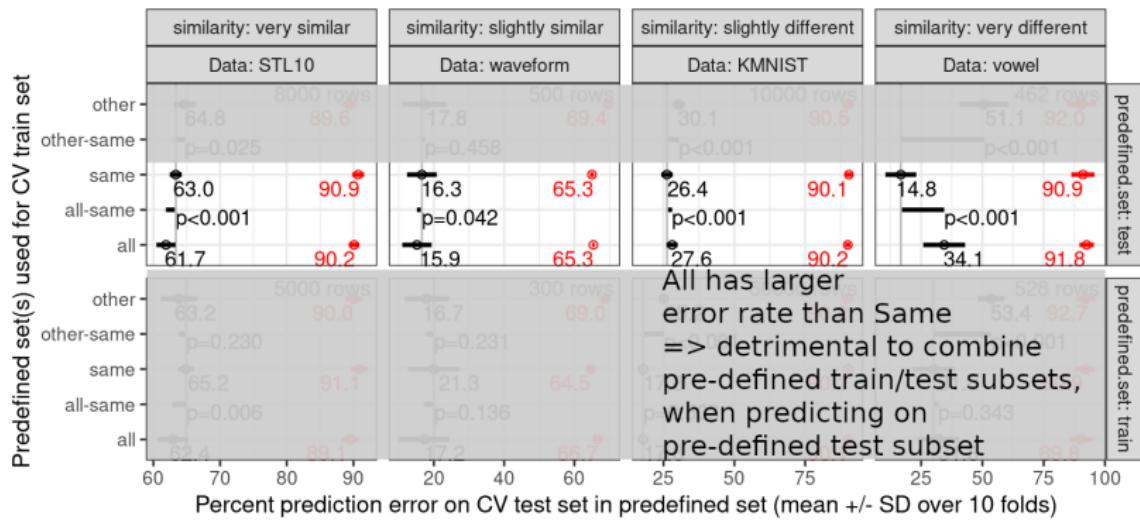
## Benchmark data with pre-defined train/test subsets

- ▶ Machine learning researchers evaluate new algorithms using benchmark data sets, which sometimes have pre-defined train/test subsets.
- ▶ For example KMNIST is an image classification data set with 60,000 images in a pre-defined train subset, and 10,000 images in a pre-defined test subset.
- ▶ STL10 is another image classification data set with 5000 images in a pre-defined train subset, and 8000 images in a pre-defined test subset.
- ▶ Are the learnable/predictable patterns in the pre-defined train/test subsets similar? (expected if random sampling was used to create pre-defined subsets)
- ▶ Or are they different?
- ▶ Use pre-defined train/test subsets in SOAK, to see if patterns are learnable/predictable across pre-defined subsets.

# Benchmark data with pre-defined train/test subsets

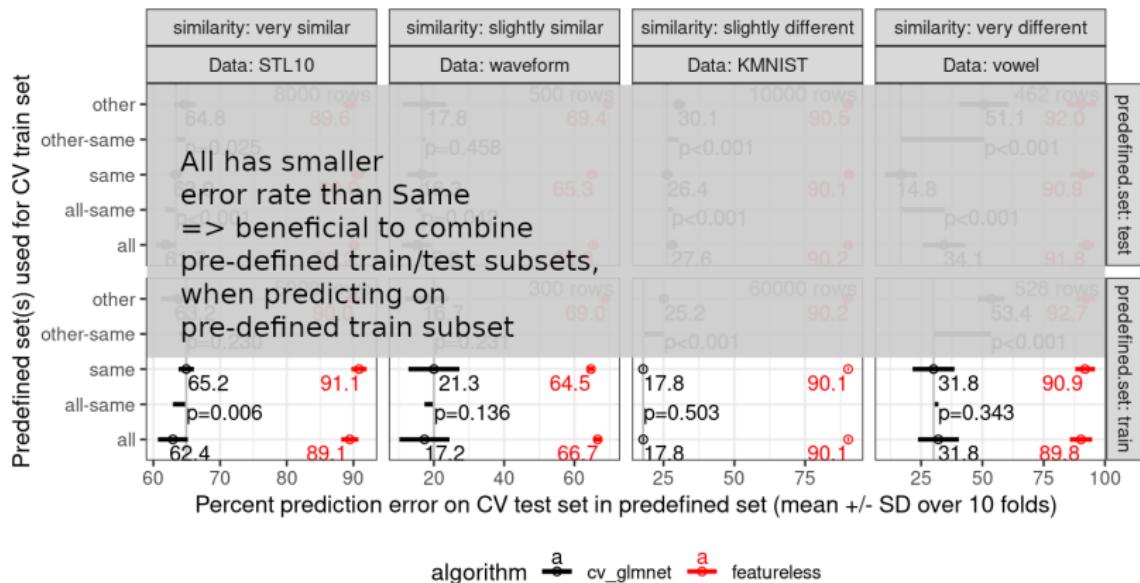


# Benchmark data with pre-defined train/test subsets

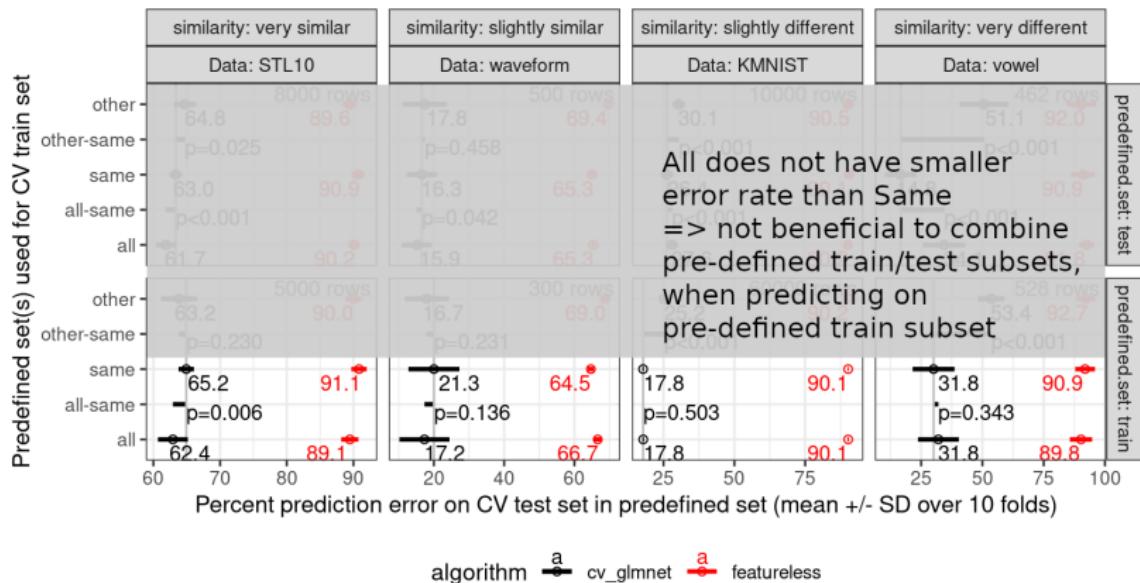


algorithm    cv\_glmnet    featureless

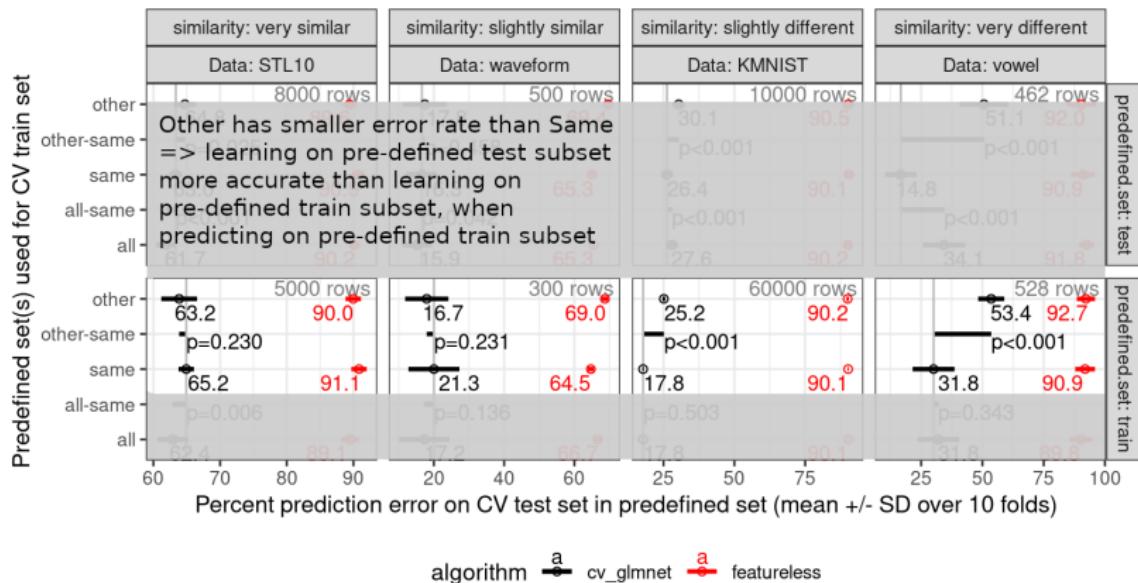
# Benchmark data with pre-defined train/test subsets



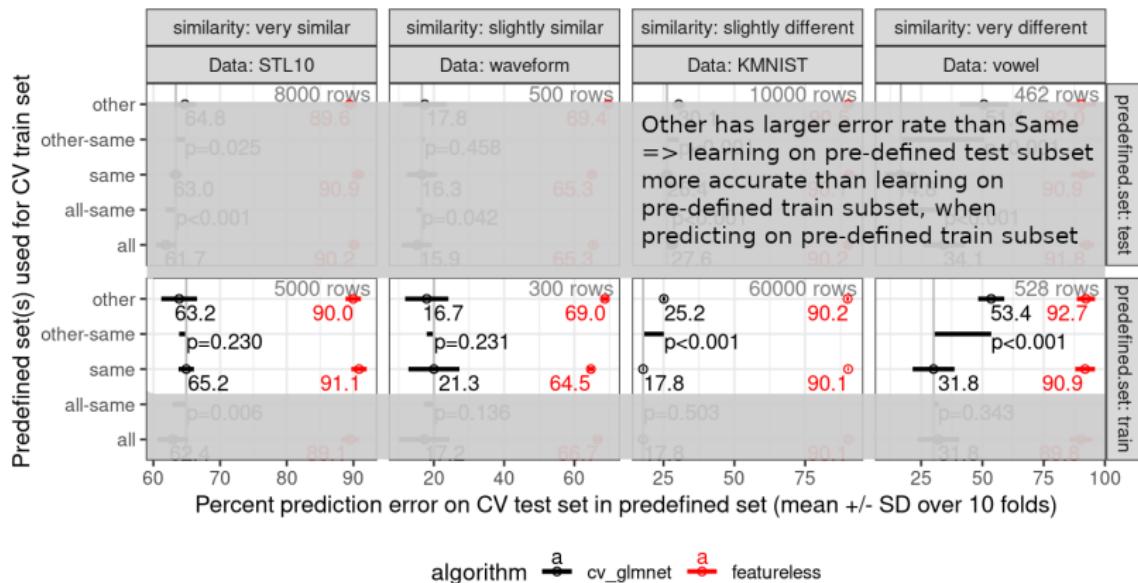
# Benchmark data with pre-defined train/test subsets



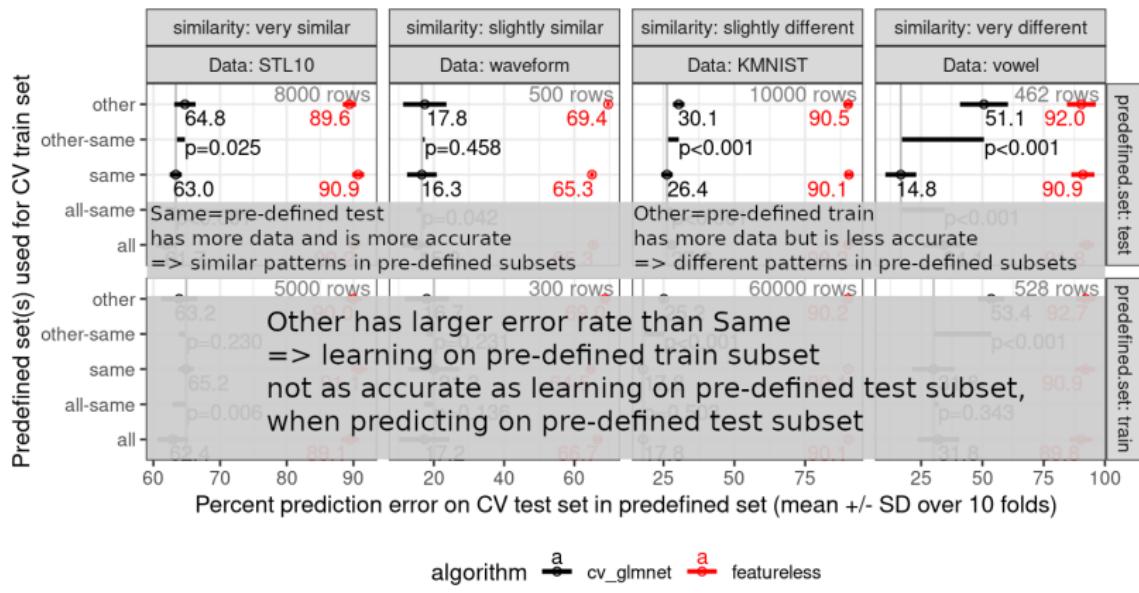
# Benchmark data with pre-defined train/test subsets



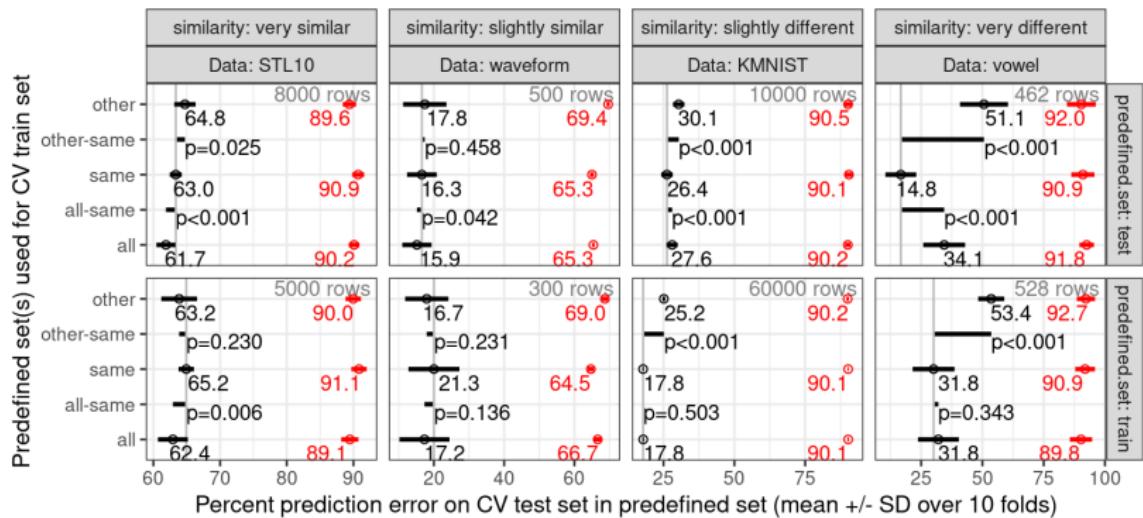
# Benchmark data with pre-defined train/test subsets



# Benchmark data with pre-defined train/test subsets



# Benchmark data with pre-defined train/test subsets



algorithm cv\_glmnet featureless

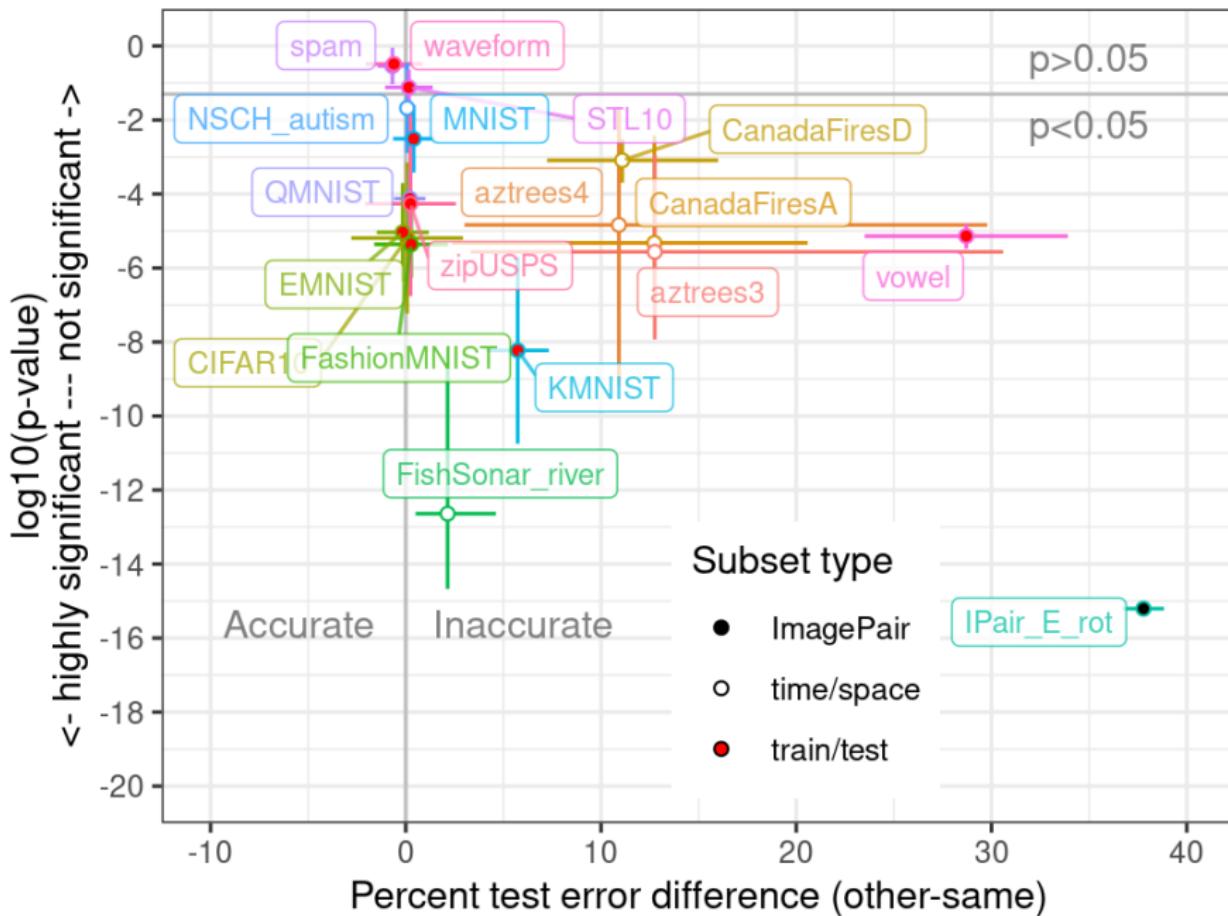
- ▶ Left data sets have similar pre-defined train/test subsets (expected), All error is always less than Same, and Other error is either greater or less than same, depending on subset sizes.
- ▶ Right data sets have different pre-defined train/test subsets (surprising), All error is always greater than or equal to Same, and Other error is always greater than Same.

# 20 classification data sets analyzed using SOAK

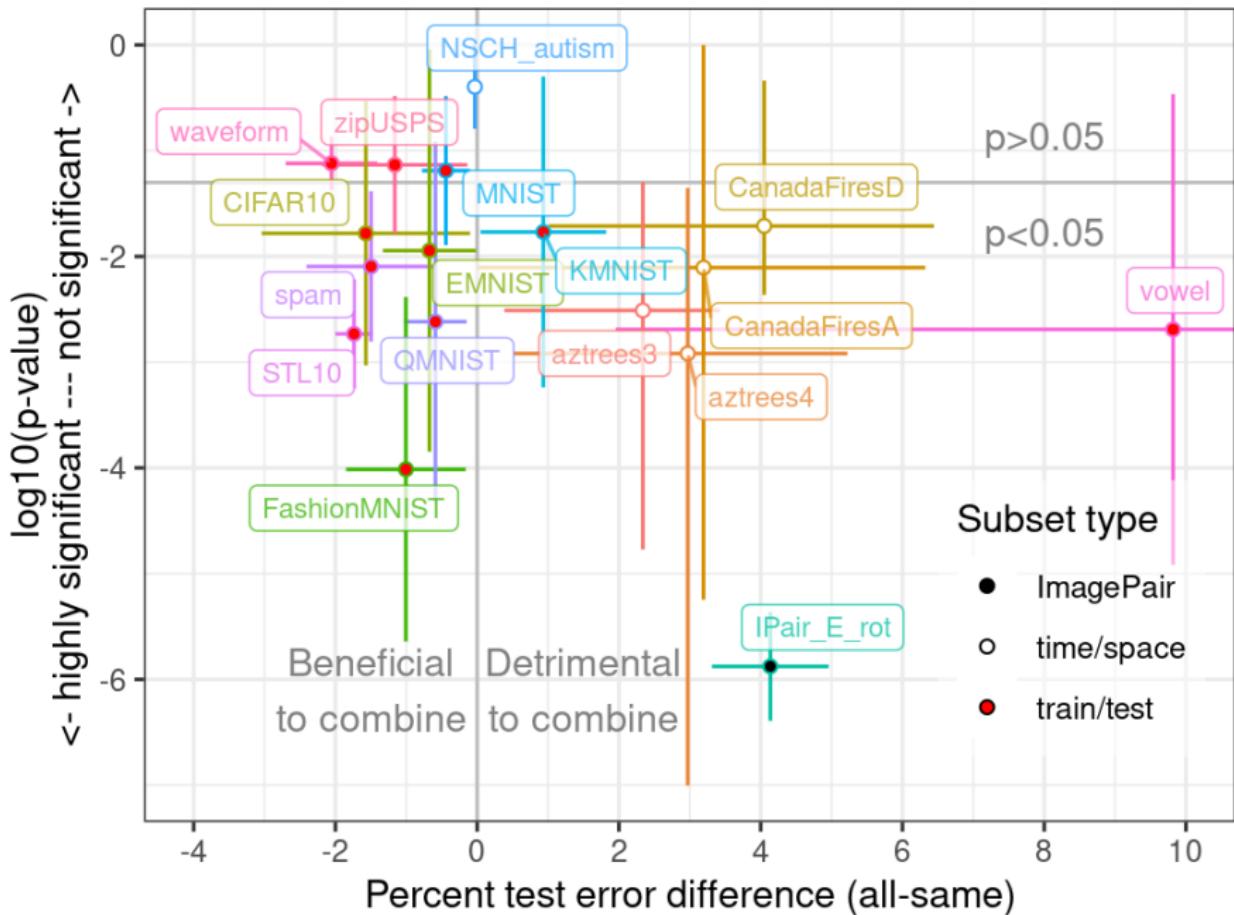
We considered MNIST and variants as subsets (**ImagePair**),  
data from collaborations (**time/space** subsets),  
and benchmark data with pre-defined **train/test** subsets.

Type	Data	rows	features	classes	subsets	imb.
1 ● ImagePair	IPair_E	140000	784	10	2	1.0
2 ● ImagePair	IPair_E_rot	140000	784	10	2	1.0
3 ● ImagePair	IPair_Fashion	140000	784	10	2	1.0
4 ○ time/space	CanadaFiresA	4827	46	2	4	7.0
5 ○ time/space	CanadaFiresD	1491	46	2	4	1.6
6 ○ time/space	FishSonar_river	2815744	81	2	4	1.2
7 ○ time/space	NSCH_autism	46010	364	2	2	1.5
8 ○ time/space	aztrees3	5956	21	2	3	2.0
9 ○ time/space	aztrees4	5956	21	2	4	4.9
10 ● train/test	CIFAR10	60000	3072	10	2	5.0
11 ● train/test	EMNIST	70000	784	10	2	6.0
12 ● train/test	FashionMNIST	70000	784	10	2	6.0
13 ● train/test	KMNIST	70000	784	10	2	6.0
14 ● train/test	MNIST	70000	784	10	2	6.0
15 ● train/test	QMNIST	120000	784	10	2	1.0
16 ● train/test	STL10	13000	27648	10	2	1.6
17 ● train/test	spam	4601	57	2	2	2.0
18 ● train/test	vowel	990	10	11	2	1.1
19 ● train/test	waveform	800	21	3	2	1.7
20 ● train/test	zipUSPS	9298	256	10	2	3.6

## Accurate prediction on a new subset?



# Is it beneficial to combine subsets?



## Discussion and Conclusions

- ▶ Proposed SOAK algorithm shows if data subsets are similar enough for learning/prediction.
- ▶ It is a concept that is new to ML frameworks (proposed **subset** column/idea not the same as **group**).
- ▶ In Autism data, there was a slight benefit to combining years.
- ▶ In fires/trees/fish data, we observed significant differences between images/regions/rivers.
- ▶ Some pre-defined train/test subsets in benchmark data are similar (STL10/waveform), others are not (KMNIST/vowel).
- ▶ Free/open-source R package available in mlr3 framework (easy parallelization over algorithms, data sets, train/test splits) <https://github.com/tdhock/mlr3resampling>
- ▶ These slides are reproducible, using the code in <https://github.com/tdhock/cv-same-other-paper>

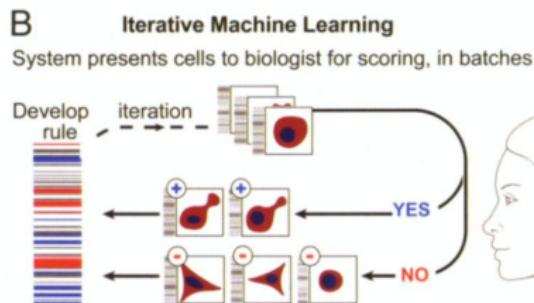
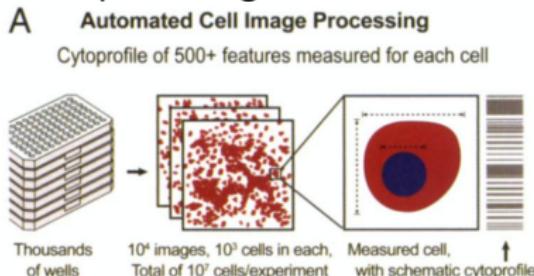
Introduction: two common questions in collaborations involving applications of machine learning

SOAK: Same/Other/All K-fold cross-validation for estimating similarity of patterns in data subsets (arXiv:2410.08643)

AUM: Area Under Min(FPR,FNR), a new differentiable loss for ROC curve optimization (JMLR'23)

# Review of supervised binary classification

- ▶ Given pairs of inputs  $\mathbf{x} \in \mathbb{R}^P$  and outputs  $y \in \{0, 1\}$  can we learn a score  $f(\mathbf{x}) \in \mathbb{R}$ , predict  $y = 1$  when  $f(\mathbf{x}) > 0$ ?
- ▶ Example: email,  $\mathbf{x}$  = bag of words,  $y$  = spam or not.
- ▶ Example: images. Jones *et al.* PNAS 2009.

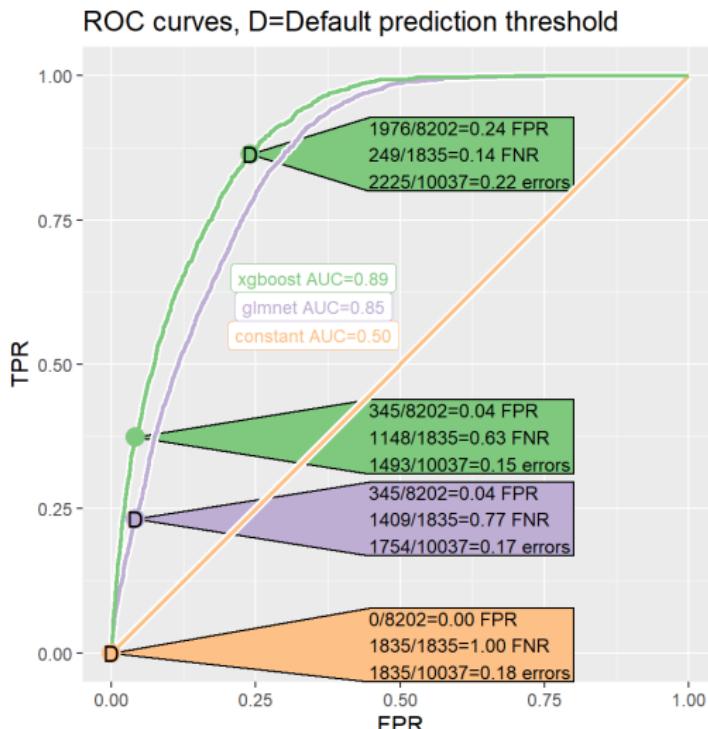


Gradient descent algorithms (Logistic regression, SVM, etc) minimize a differentiable surrogate of zero-one loss = sum of:

**False positives:**  $f(\mathbf{x}) > 0$  but  $y = 0$  (predict budding, but cell is not).

**False negatives:**  $f(\mathbf{x}) < 0$  but  $y = 1$  (predict not budding, but cell is).

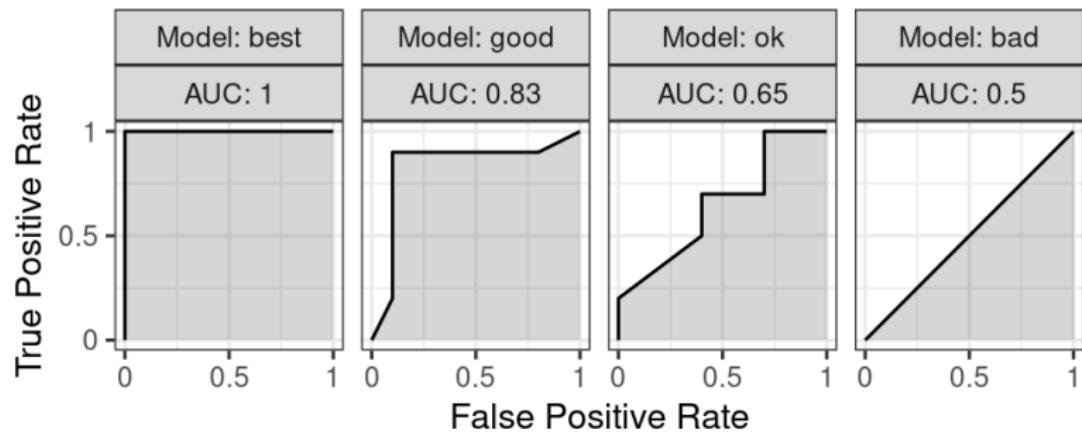
# ROC curves: fair comparison with different default FPR



- ▶ At defaults (D), glmnet has fewer errors (misleading).
- ▶ At FPR=4%, xgboost has fewer errors (fair comparison).

# Receiver Operating Characteristic (ROC) Curves

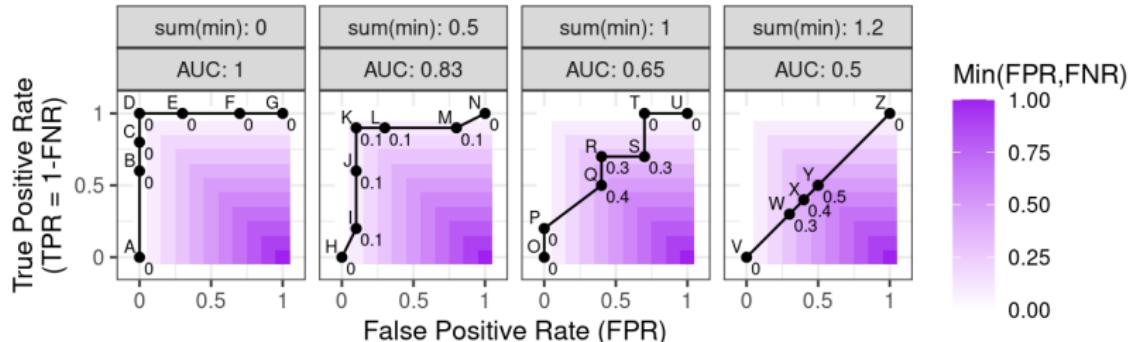
- ▶ Classic evaluation method from the signal processing literature (Egan and Egan, 1975).
- ▶ ROC curve of learned  $f$  is plot of True Positive Rate vs False Positive Rate: each point on the ROC curve is a different constant  $c \in \mathbb{R}$  added to the predicted values:  $f(\mathbf{x}) + c$ .
- ▶  $c = \infty$  means always predict positive label ( $\text{FPR}=\text{TPR}=1$ ).
- ▶  $c = -\infty$  means always predict negative label ( $\text{FPR}=\text{TPR}=0$ ).
- ▶ Best classifier has a point near upper left ( $\text{TPR}=1$ ,  $\text{FPR}=0$ ), with large Area Under the Curve (AUC).



## Research question and new idea

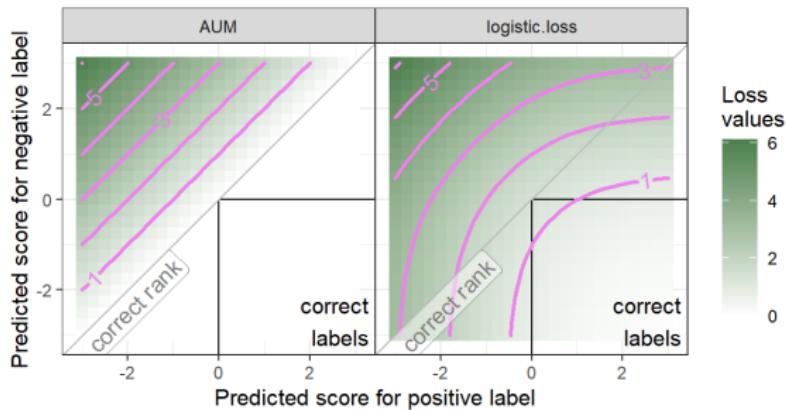
Can we learn a binary classification function  $f$  which directly optimizes the ROC curve?

- ▶ Most algorithms involve minimizing a differentiable surrogate of the zero-one loss, which is not the same.
- ▶ The Area Under the ROC Curve (AUC) is piecewise constant (gradient zero almost everywhere), so can not be used with gradient descent algorithms.
- ▶ We proposed (Hocking, Hillman 2023) to encourage points to be in the upper left of ROC space, using a loss function which is a differentiable surrogate of the sum of  $\min(\text{FPR}, \text{FNR})$ .

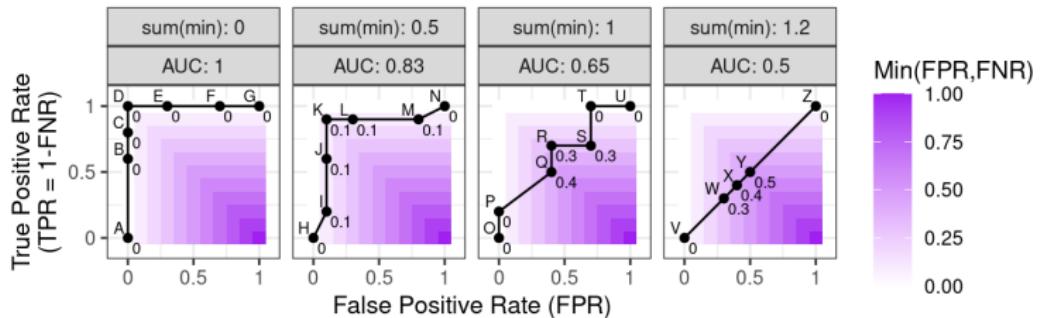


# Comparing proposed loss with baselines

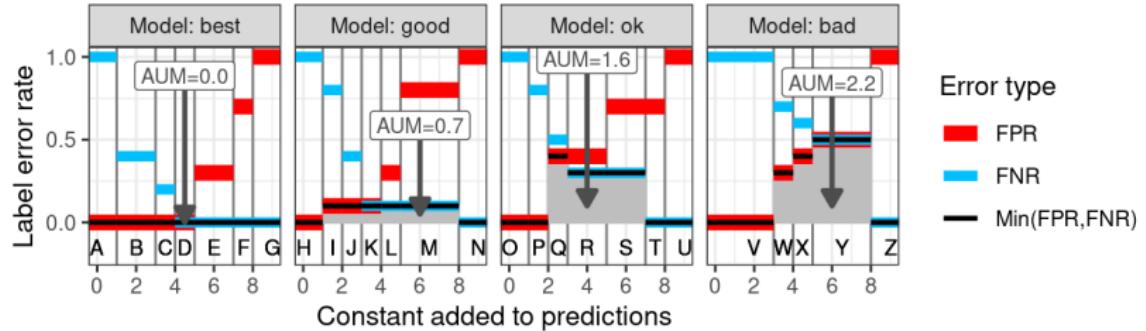
- ▶ Classic baselines: hinge and logistic loss, sum over samples,  $\ell[yf(x)]$ .
- ▶ Bamber (1975) proved ROC-AUC relation to Mann-Whitney U statistic (double sum over all pairs of positive and negative samples).
- ▶ Recently: SVM<sup>struct</sup> (Joachims 2005), X-risk (Yang 2022), All Pairs Squared Hinge (Rust and Hocking 2023), sum loss over pairs of positive and negative samples,  $\ell[f(x^+) - f(x^-)]$ .
- ▶ Proposed: sort-based AUM loss (sum over points on ROC curve).
- ▶ Figure below: loss for two samples: one positive, one negative.



# Large AUC $\approx$ small Area Under Min(FP,FN) (AUM)

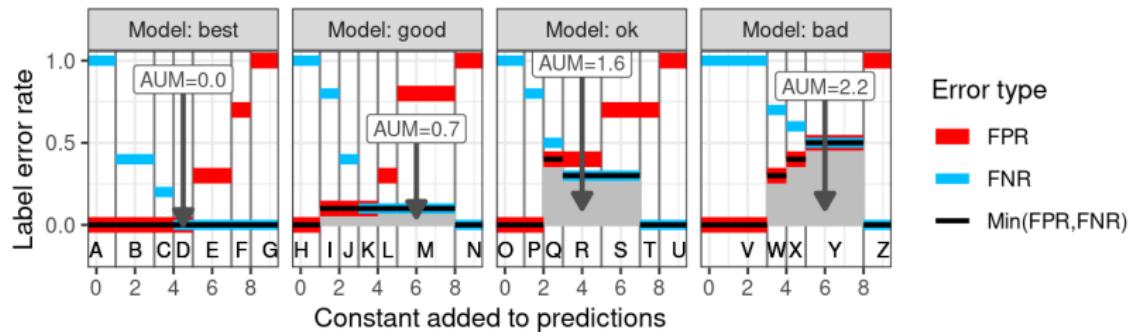


Above: purple heat map = numbers near dots = distance to top or left  
= same as black min error rate functions below.



Hocking, Hillman, *Journal of Machine Learning Research* (2023).

# Computing Sum of Min (SM) over all ROC points

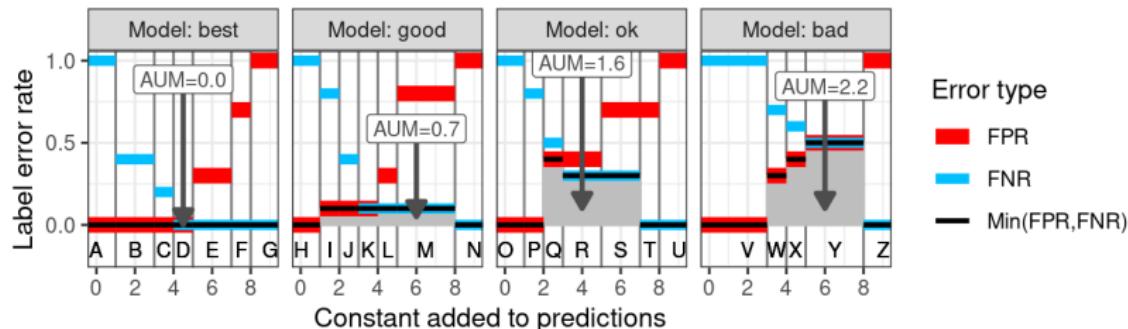


- ▶ For  $N$  samples, there are  $\leq N + 1$  points on the ROC curve,
- ▶ with  $T_1 \leq \dots \leq T_N \in \mathbb{R}$  (grey lines), sorted thresholds of  $c$ ,
- ▶ and corresponding min error values  $M_1, \dots, M_N$  (black).
- ▶ Then if  $I$  is the indicator function, we can write the sum of the min (SM), over all ROC points, as:

$$SM = \sum_{i=2}^N I[T_i \neq T_{i-1}] M_i = \sum_{i: T_i \neq T_{i-1}} M_i.$$

( $\neq$  required: a tie  $T_i = T_{i-1}$  deletes a point from the ROC curve)

# Computing proposed loss, Area Under Min (AUM)



The proposed AUM can be interpreted as an L1 relaxation of SM,

$$SM = \sum_{i=2}^N I[T_i \neq T_{i-1}] M_i = \sum_{i:T_i \neq T_{i-1}} M_i.$$

$$AUM = \sum_{i=2}^N [T_i - T_{i-1}] M_i.$$

AUM is therefore a surrogate loss for ROC-SM minimization.  
L1 relaxation  $\Rightarrow$  constant/non-zero gradients.

## ROC curve pytorch code uses argsort

```
def ROC_curve(pred_tensor, label_tensor):
    sorted_indices = torch.argsort(-pred_tensor)
    ... # torch.cumsum() etc
    return { # a dictionary of torch tensors
        "FPR":FPR, "FNR":FNR, "TPR":1 - FNR,
        "min(FPR,FNR)":torch.minimum(FPR, FNR),
        "min_constant":torch.cat([
            torch.tensor([-torch.inf]), uniq_thresh]),
        "max_constant":torch.cat([
            uniq_thresh, torch.tensor([torch.inf])]) }
>>> pd.DataFrame(ROC_curve(torch.tensor(
...     [2.0, -3.5, -1.0, 1.5]), torch.tensor([0,0,1,1])))
   FPR  FNR  TPR  min(FPR,FNR)  min_constant  max_constant
0  0.0  1.0  0.0          0.0           -inf         -2.0
1  0.5  1.0  0.0          0.5          -2.0         -1.5
2  0.5  0.5  0.5          0.5          -1.5          1.0
3  0.5  0.0  1.0          0.0           1.0          3.5
4  1.0  0.0  1.0          0.0           3.5           inf
```

## AUC and proposed AUM both use ROC curve

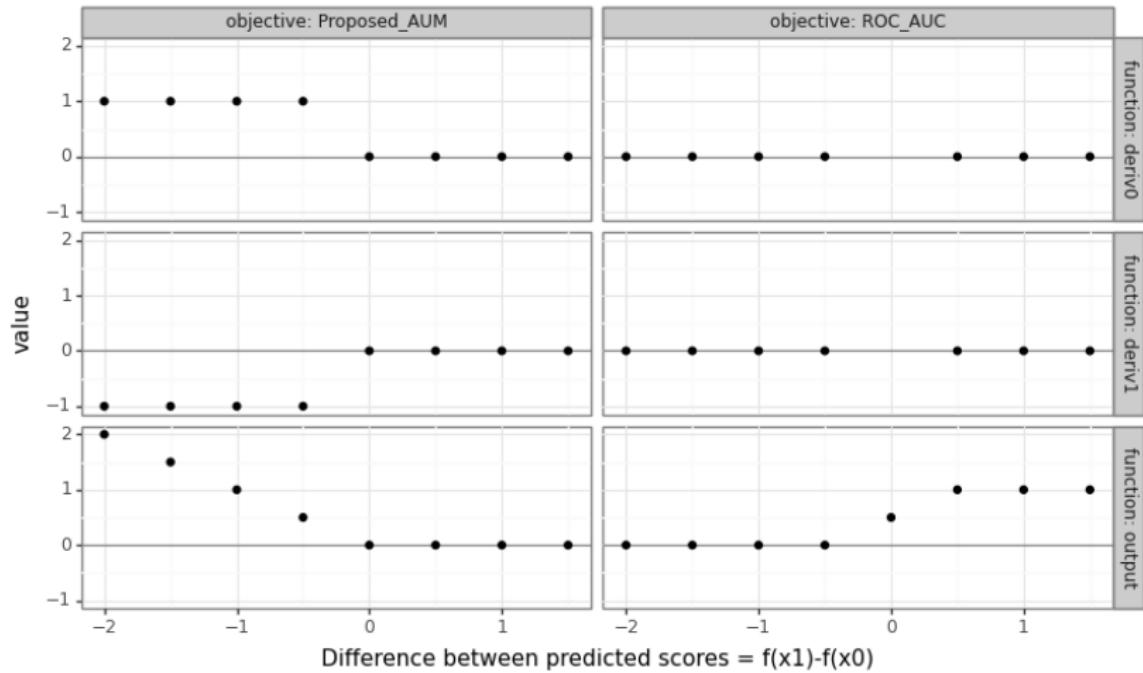
```
def ROC_AUC(pred_tensor, label_tensor):
    "Classic metric, but gradient zero almost everywhere"
    roc = ROC_curve(pred_tensor, label_tensor)
    FPR_diff = roc["FPR"] [1:]-roc["FPR"] [: -1]
    TPR_sum = roc["TPR"] [1:]+roc["TPR"] [: -1]
    return torch.sum(FPR_diff*TPR_sum/2.0)

def Proposed_AUM(pred_tensor, label_tensor):
    "Surrogate loss, non-zero gradient for predictions"
    roc = ROC_curve(pred_tensor, label_tensor)
    min_FPR_FNR = roc["min(FPR,FNR)"] [1:-1]
    constant_diff = roc["min_constant"] [1:].diff()
    return torch.sum(min_FPR_FNR * constant_diff)
```

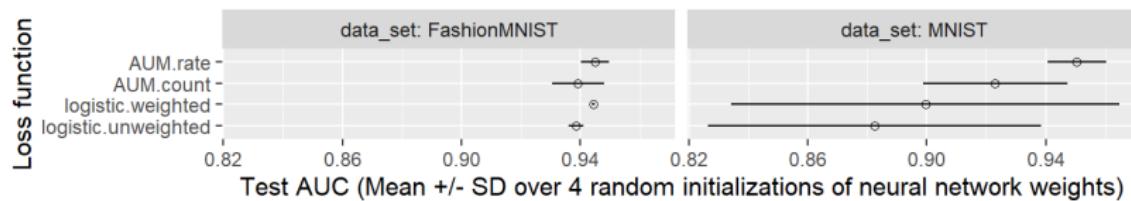
<https://tdhock.github.io/blog/2024/torch-roc-aum/>

# Proposed AUM pytorch code, auto-grad demo

- ▶ Assume two samples,  $(x_0, y_0 = 0), (x_1, y_1 = 1)$ ,
- ▶ Plot objective and gradient with respect to predicted scores.

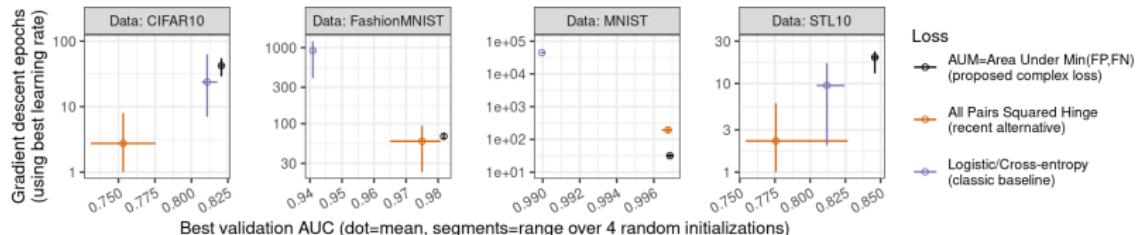


# Comparing proposed AUM with weighted logistic loss



- ▶ Two image classification data sets.
- ▶ LeNet5 convolutional neural network, batch size 1000.
- ▶ Step size from  $10^{-4}$  to  $10^2$  (keep best).
- ▶ AUM rate uses Area Under Min of FPR/FNR, more accurate in these data than AUM count (FP/FN totals).
- ▶ logistic unweighted is usual binary cross-entropy loss (uniform weight=1 for each sample).
- ▶ for logistic weighted, we compute class frequencies,  $n_1 = \sum_{i=1}^N I[y_i = 1]$  and  $n_0$  similar; then weights are  $w_i = 1/n_{y_i}$  so that total weight of positive class equals total weight of negative class (more accurate in these data).

# AUM gradient descent increases validation AUC, four image classification data sets

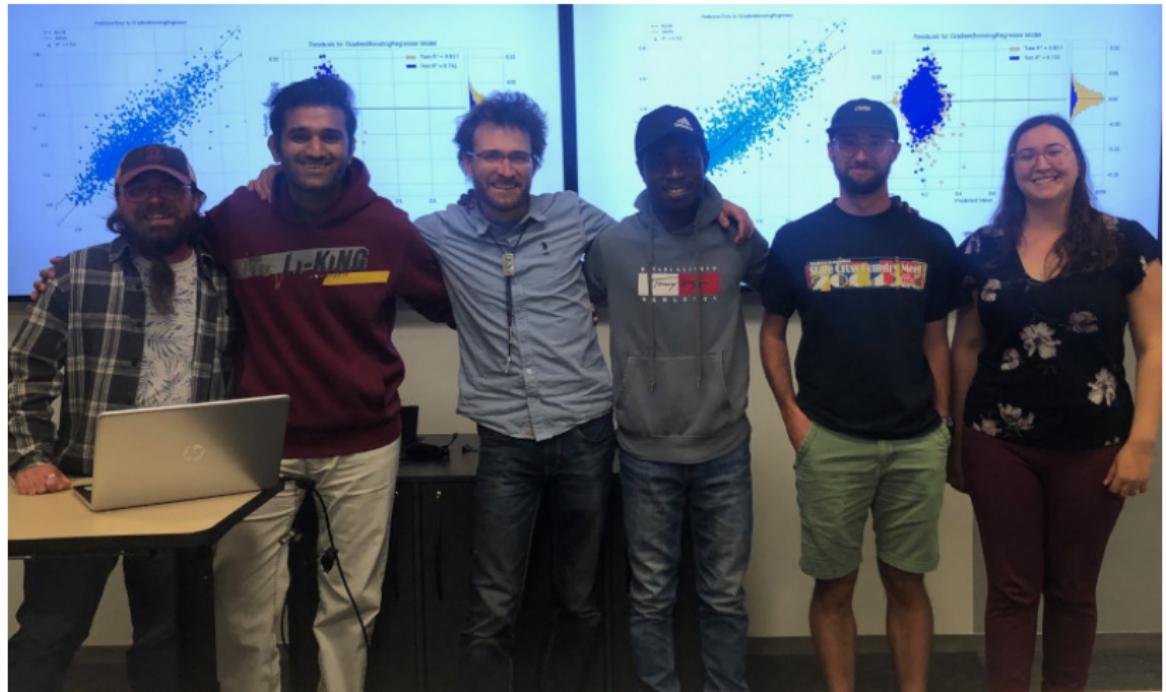


- ▶ Unbalanced binary classification: 10% negative, 90% positive.
- ▶ Gradient descent with constant step size, best of  $10^{-4}$  to  $10^5$ .
- ▶ Full gradient (batch size = number of samples).
- ▶ Linear model, max iterations = 100,000.
- ▶ Max Validation AUC comparable or better than baselines: logistic loss and all paired squared hinge (X-risk etc).
- ▶ Number of epochs comparable to baselines.
- ▶ Time per epoch is  $O(N \log N)$  (sort), small log factor larger than standard logistic/cross-entropy loss,  $O(N)$ .

## Discussion and future work

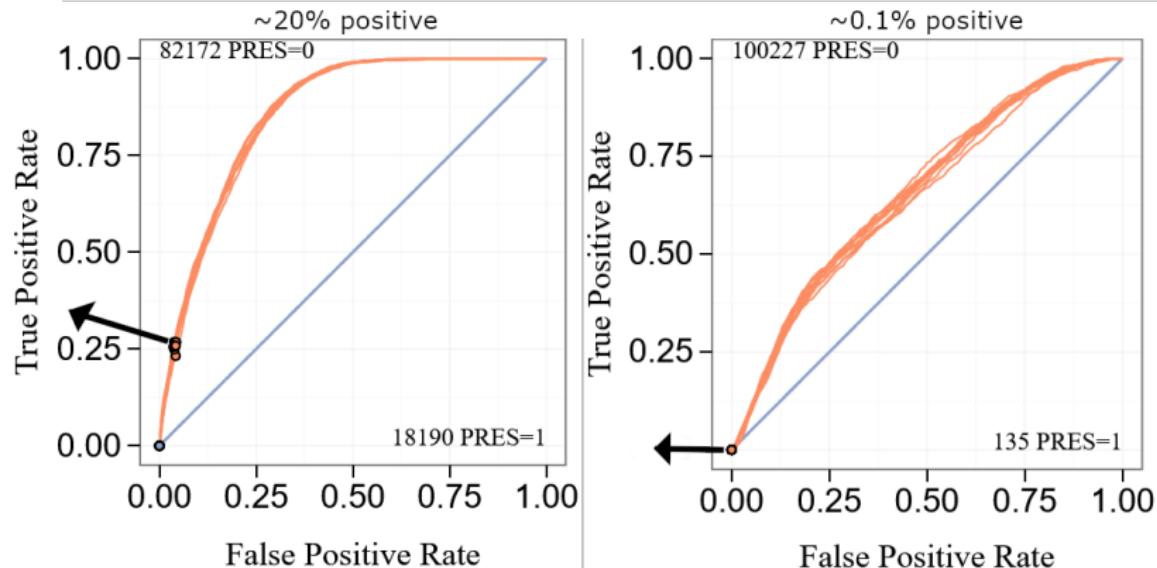
- ▶ Classic surrogate loss functions sum over samples.
- ▶ Proposed AUM loss similar to recent losses that sum over all pairs of positive and negative examples (both can be implemented by sorting predicted scores).
- ▶ Proposed AUM loss uses a different/novel relaxation.
- ▶ Proposed AUM loss implemented in pytorch code, can be used as a drop-in replacement for logistic/binary cross-entropy loss, <https://tdhock.github.io/blog/2024/torch-roc-aum/>
- ▶ Best use with stochastic gradient algorithms? At least one positive and one negative example is required in each batch.
- ▶ Algorithms like SVM? (margin/kernel)
- ▶ How to adapt to multi-class setting, and other problems such as ranking/information retrieval?
- ▶ See our JMLR'23 paper for an application to change-point detection, and arXiv:2410.08635 for an efficient line search that exploits the piecewise linear/constant nature of AUM/AUC.

# Thanks to my students and collaborators!



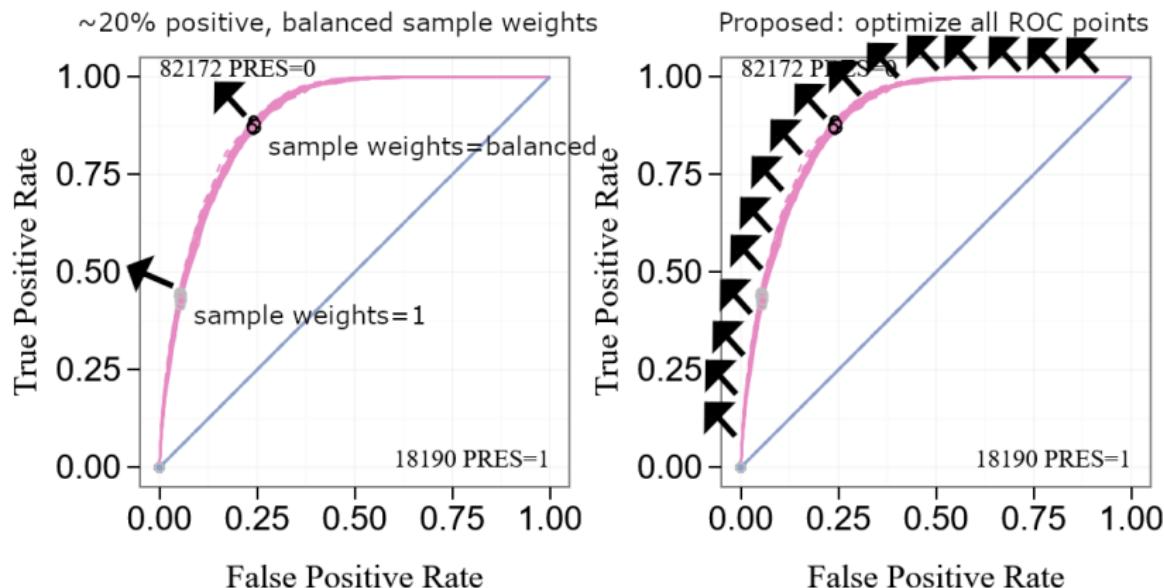
Please email me if you are interested to collaborate:  
[toby.dylan.hocking@usherbrooke.ca](mailto:toby.dylan.hocking@usherbrooke.ca)

# Gradients of sample-based loss are influenced by imbalance



- ▶ Left: some imbalance, 20% positive labels, gradient 4x stronger along X axis / False Positive Rate.
- ▶ Right: large imbalance, 0.1% positive labels, gradient 1000x stronger along X axis / False Positive Rate. (True Positive / Y axis gradients essentially ignored)

## Gradients using balanced sample weights, proposed loss



- ▶ Left: gradient 4x stronger along X axis for sample weights=1. Balanced sample weights mean equal influence for gradients along both axes, based on the current prediction threshold.
- ▶ Right: proposed method computes gradients based on all ROC points, not just the current prediction threshold.