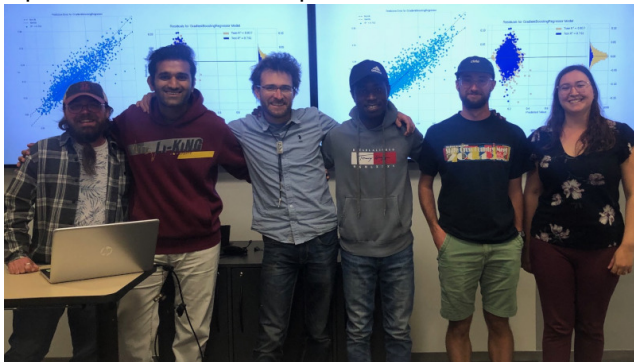


Two new algorithms for scientific applications of machine learning

Toby Dylan Hocking — 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 comparing models trained on data subsets

AUM=Area Under $\min(\text{FPR}, \text{FNR})$, a new differentiable loss for ROC curve optimization

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 Algorithm | Train data | Learned function | Predictions on test data |
|--------------------|------------|------------------|--------------------------|
|--------------------|------------|------------------|--------------------------|

| | | | |
|--------|---|-------|-------------------|
| Learn(|  |) → g | $g(\text{0}) = 0$ |
| | | | $g(\text{1}) = 1$ |
| | | | $g(\text{7}) = 1$ |
| | | | |

| | | | |
|--------|---|-------|-----------------------|
| Learn(|  |) → h | $h(\text{shirt}) = 0$ |
| | | | $h(\text{shirt}) = 0$ |
| | | | $h(\text{pants}) = 1$ |
| | | | |

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

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



Train on one subset and accurately predict on another?



- ▶ What if you do $g(\text{shoe})$, or $h(\text{ring})$?
- ▶ This is a question about **generalization**: how accurate is the learned function on a new/test data set?
- ▶ “Very accurate” if test data are similar enough to train data (best case is i.i.d. = independent and identically distributed)
- ▶ Predicting childhood autism (Lindly *et al.*), train on one year of surveys, test on another.
- ▶ Predicting carbon emissions (Aslam *et al.*), train on one city, test on another.
- ▶ Predicting presence of trees/burn in satellite imagery (Shenkin *et al.*, Thibault *et al.*), train on one geographic area/image, test on another.
- ▶ Predicting fish spawning habitat in sonar imagery (Bodine *et al.*), train on one river, test on another.
- ▶ 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?

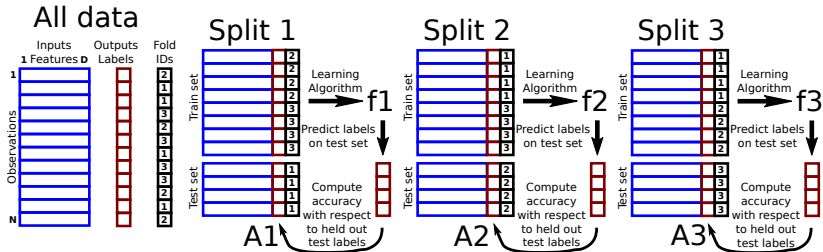
Introduction: two common questions in collaborations involving applications of machine learning

SOAK: Same/Other/All K-fold cross-validation for comparing models trained on data subsets

AUM=Area Under $\min(\text{FPR}, \text{FNR})$, a new differentiable loss for ROC curve optimization

K -fold cross-validation: a standard algorithm used to estimate the 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).

Example data set: 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, one column per survey question.
- ▶ Pre-processing to obtain common columns over the two years, remove missing values, one-hot/dummy variable encoding.
- ▶ Result is $N = 46,010$ rows and $D = 366$ columns.
- ▶ 18,202 rows for 2019; 27,808 rows for 2020.
- ▶ One column is diagnosis with Autism (binary classification, yes or no), can we predict it using the others?
- ▶ Can we train on one year, and accurately predict on another?
- ▶ Can we get a more accurate model by combining data from different years?

Proposed SOAK algorithm (Autism data example)

- ▶ Example: childhood autism prediction data set.

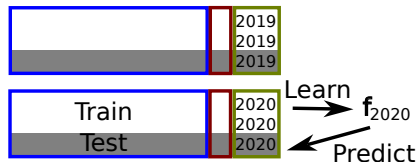
| | | Inputs | | | |
|--------|---|--------|-----------|---|--------|
| | | 1 | Questions | D | Autism |
| People | 1 | | | 0 | 2019 |
| | | | | 0 | 2019 |
| | | | | 1 | 2019 |
| | | | | 0 | 2020 |
| | | | | 1 | 2020 |
| | | | | 1 | 2020 |
| | 2 | | | 1 | 2020 |

Proposed Same Other Cross-Validation

- ▶ Train subset same as test (=regular K -fold CV on 2020).

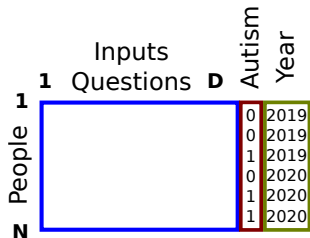
| People | Inputs | | Autism | Year |
|--------|--------|-----------|--------|------|
| | 1 | Questions | D | |
| 1 | | | 0 | 2019 |
| | | | 0 | 2019 |
| | | | 1 | 2019 |
| | | | 0 | 2020 |
| | | | 1 | 2020 |
| | | | 1 | 2020 |

Same

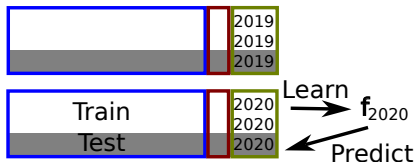


Proposed SOAK algorithm (Autism data example)

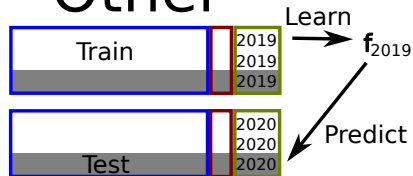
- ▶ Train subset (2019) different from test (2020).



Same

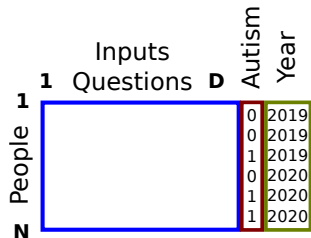


Other

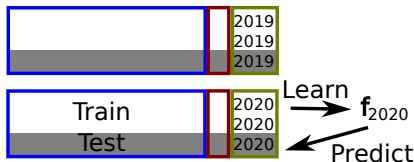


Proposed SOAK algorithm (Autism data example)

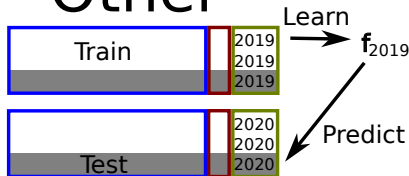
- Repeat for each test subset (2019,2020).



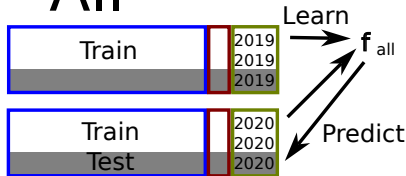
Same



Other

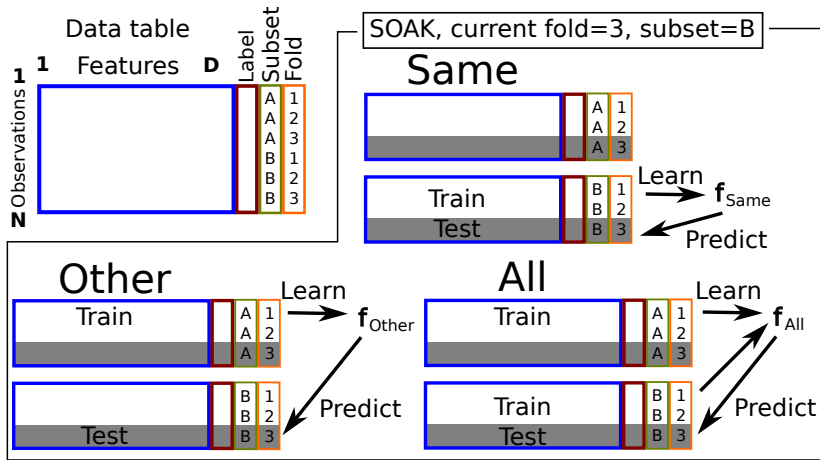


All



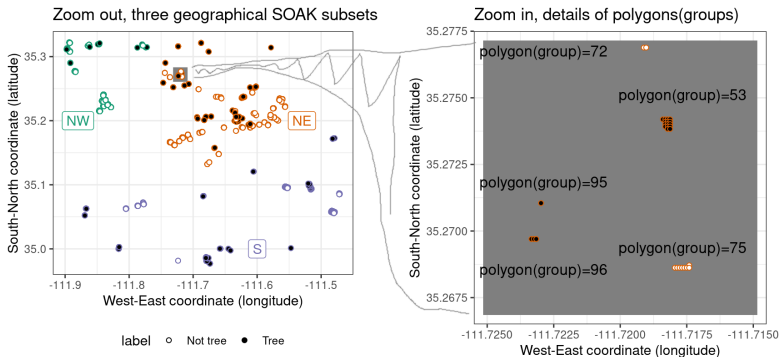
Proposed SOAK algorithm (generic data)

- ▶ Key new idea is **subset** column in data table.
- ▶ Compute test error for each subset (A/B) and fold (1/2/3).



Proposed SOAK algorithm is new/unique

- ▶ 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)



Proposed SOAK algorithm, interpretation/expectations

For a fixed test set from one subset:

If train/test are similar/iid,

All should be most accurate.

Same/Other should be less accurate, because there is less data available (if other is larger than same, then other should be more accurate than same, etc).

If train/test are different (not iid),

Same should be most accurate.

Other should be substantially less accurate.

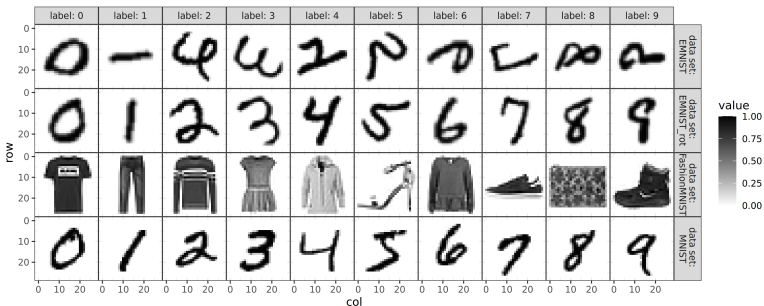
All accuracy should be between same and other.

- ▶ Can we train on one year, and accurately predict on another? Compare Same/Other test error.
- ▶ Can we get a more accurate model by combining data from different years? Compare Same/All test error.

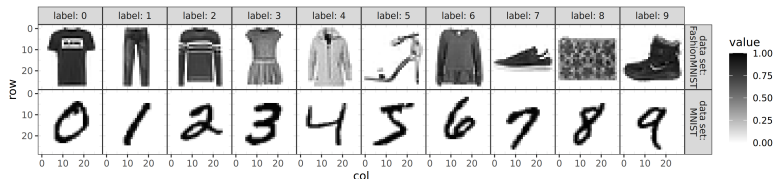
ImagePair data: train on MNIST and accurately predict on MNIST variants?

Recall: what happens if you do $g(\text{boot image})$, or $h(\text{0 image})$?

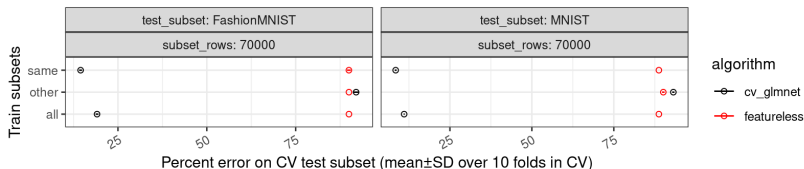
- ▶ Boot image comes from FashionMNIST data, which were used to learn h .
- ▶ 0 image comes from MNIST data, which were used to learn g .



SOAK for MNIST+FashionMNIST data

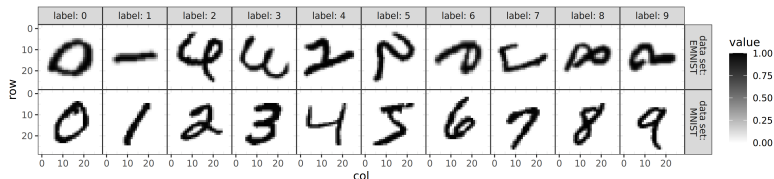


Data set: MNIST_FashionMNIST

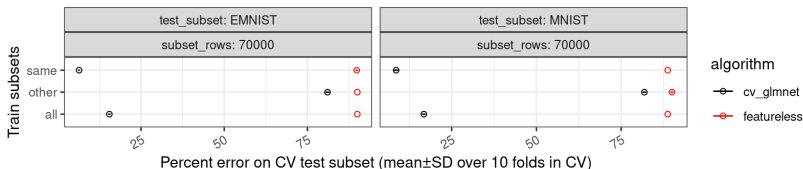


- ▶ `cv_glmnet` is L1-regularized linear model, featureless baseline always predicts most frequent class.
- ▶ Other `cv_glmnet` has greater test error than featureless, which indicates that the patterns are too different to learn anything at all.

SOAK for MNIST+EMNIST data

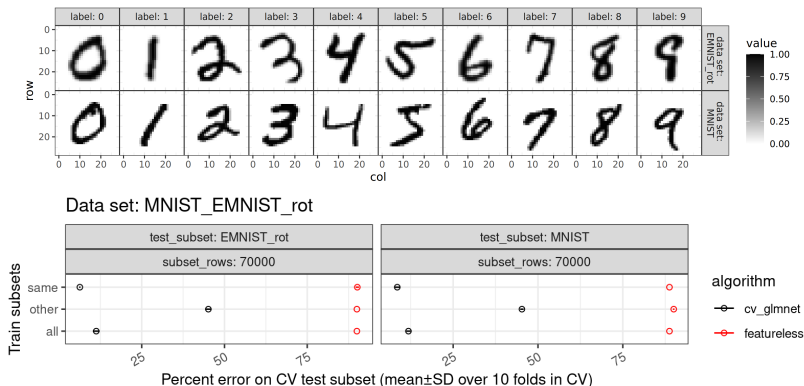


Data set: MNIST_EMNIST



- ▶ Other `cv_glmnet` has significantly smaller test error than featureless, so something is learned/transferrable between data sets, but it is still clear that the pattern is very different.

SOAK for MNIST+EMNIST_rot data

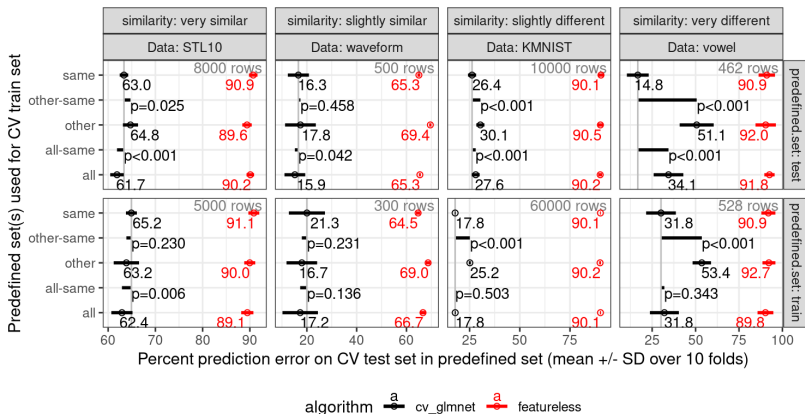


- ▶ Other cv_glmnet has even smaller test error, indicating even more similarity between MNIST and EMNIST_rot data sets.
- ▶ But Other/All test error are still not as small as Same, indicating there is a significant difference between learnable/predictable patterns in MNIST/EMNIST_rot data.

Benchmark data with a pre-defined train/test split

- ▶ Machine learning researchers evaluate new algorithms using benchmark data sets, which sometimes have pre-defined train/test splits.
- ▶ For example MNIST is a data set of images of handwritten digits (want to predict which digit, 0 to 9), with 60,000 train and 10,000 test images.
- ▶ spam is a data set of emails (want to predict spam or not, binary), with 3065 train and 1536 test emails.
- ▶ Are the patterns in the pre-defined train/test sets similar/iid?
- ▶ Or are they different?

Benchmark data with a pre-defined train/test split



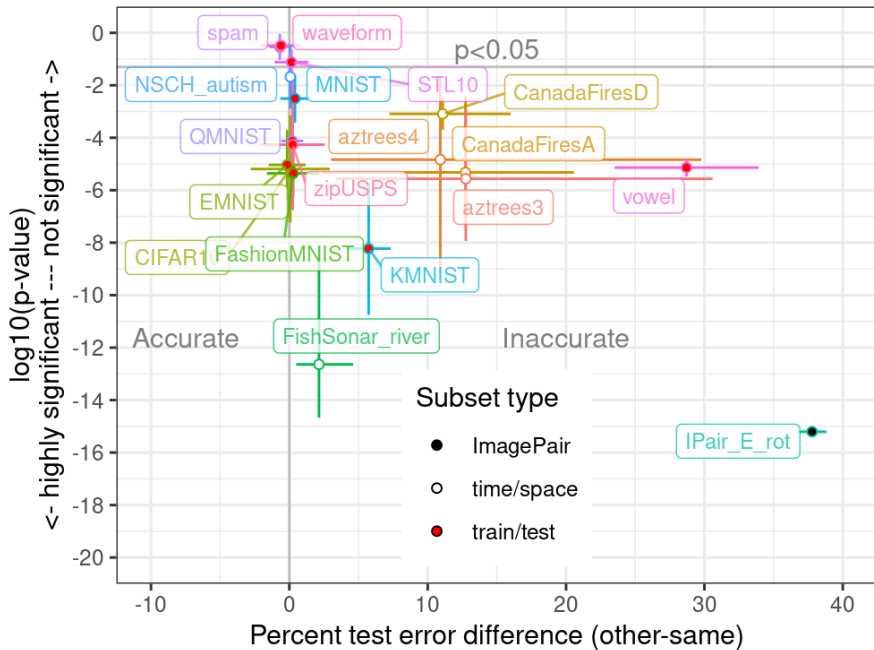
- ▶ Use pre-defined split as SOAK subset, to see if patterns are learnable/predictable in different pre-defined subsets.
- ▶ Left, similar subsets, all error less than same. (expected)
- ▶ Right, different subsets, all error greater than same. (surprising)

Data sets analyzed

- ▶ Sorted by test error difference between all and same.
- ▶ Different groups on top/positive.
- ▶ Similar groups on bottom/negative.

| | data.name | rows | features | classes | n.groups | all-same |
|----|-------------------------|---------|----------|---------|----------|----------|
| 1 | vowel | 990 | 10 | 11 | 2 | 9.98 |
| 2 | CanadaFires_downSampled | 1491 | 46 | 2 | 4 | 4.02 |
| 3 | CanadaFires_all | 4827 | 46 | 2 | 4 | 3.39 |
| 4 | aztrees4 | 5956 | 21 | 2 | 4 | 2.28 |
| 5 | aztrees3 | 5956 | 21 | 2 | 3 | 2.05 |
| 6 | FishSonar_river | 2815744 | 81 | 2 | 4 | 1.69 |
| 7 | KMNIST | 70000 | 784 | 10 | 2 | 0.87 |
| 8 | NSCH_autism | 46010 | 364 | 2 | 2 | -0.03 |
| 9 | MNIST | 70000 | 784 | 10 | 2 | -0.53 |
| 10 | QMNIST | 120000 | 784 | 10 | 2 | -0.70 |
| 11 | spam | 4601 | 57 | 2 | 2 | -0.77 |
| 12 | EMNIST | 70000 | 784 | 10 | 2 | -0.85 |
| 13 | FashionMNIST | 70000 | 784 | 10 | 2 | -0.97 |
| 14 | zipUSPS | 9298 | 256 | 10 | 2 | -1.44 |
| 15 | waveform | 800 | 21 | 3 | 2 | -1.54 |
| 16 | CIFAR10 | 60000 | 3072 | 10 | 2 | -1.77 |
| 17 | STL10 | 13000 | 27648 | 10 | 2 | -1.97 |

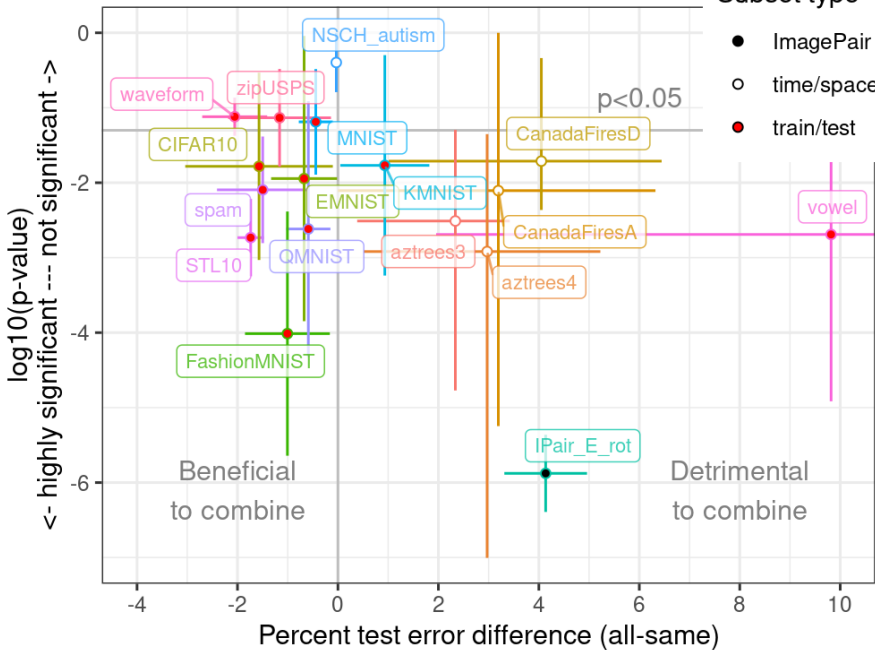
Accurate prediction on a new subset?



Is it beneficial to combine subsets?

Subset type

- ImagePair
- time/space
- train/test



Discussion and Conclusions

- ▶ Proposed SOAK algorithm shows if data subsets are similar enough for learning/prediction.
- ▶ 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 splits in benchmark data are similar/iid (STL10/waveform), others are not (KMNIST/vowel).
- ▶ Free/open-source R package available in mlr3 framework (easy parallelization over algorithms, train/test splits, data sets) <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 comparing models trained on data subsets

AUM=Area Under $\text{Min}(\text{FPR}, \text{FNR})$, a new differentiable loss for ROC curve optimization

Problem: 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.

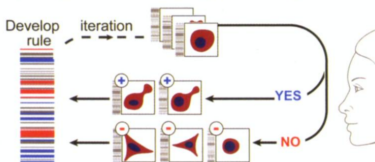
A Automated Cell Image Processing

Cytoprofile of 500+ features measured for each cell



B Iterative Machine Learning

System presents cells to biologist for scoring, in batches

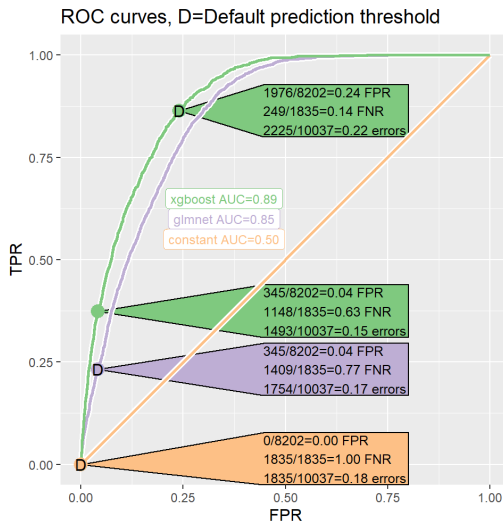


Most algorithms (SVM, Logistic regression, etc) minimize a differentiable surrogate of zero-one loss = sum of:

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).

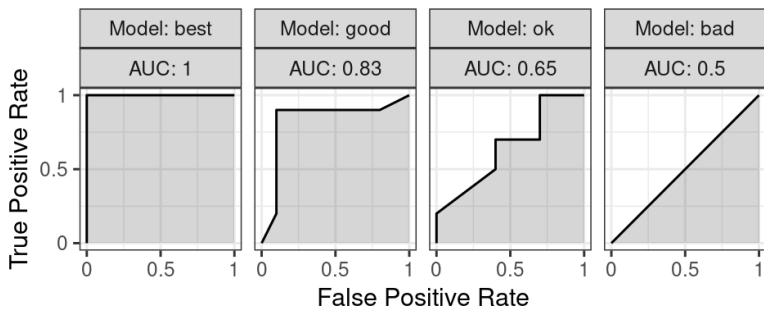
ROC curves for evaluation, especially useful with imbalance



- ▶ At default FPR=24% (D), glmnet has fewer errors.
- ▶ At FPR=4%, xgboost has fewer errors.

Receiver Operating Characteristic (ROC) Curves

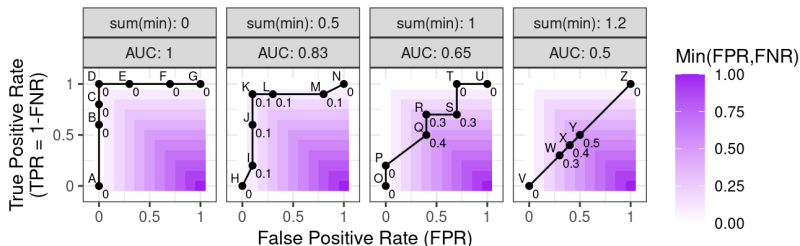
- ▶ Classic evaluation method from the signal processing literature (Egan and Egan, 1975).
- ▶ ROC curve of learned f is plot of True Positive Rate vs False Positive Rate: each point on the ROC curve is a different constant $c \in \mathbb{R}$ added to the predicted values: $f(\mathbf{x}) + c$.
- ▶ $c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



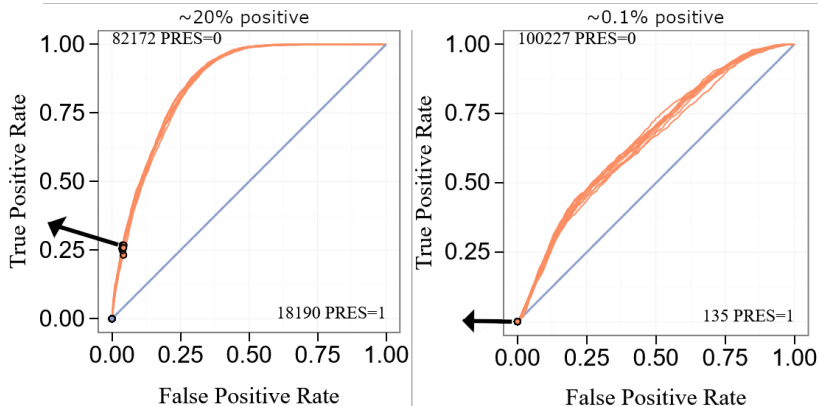
Research question and new idea

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 $\min(\text{FPR}, \text{FNR})$.

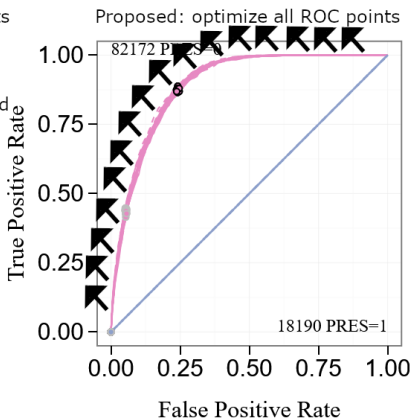
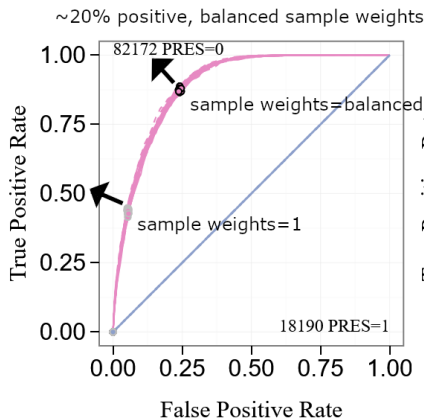


Gradients of sample-based loss are influenced by imbalance



- ▶ 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)

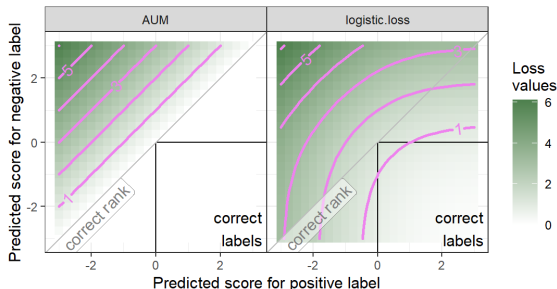
Gradients using balanced sample weights, proposed loss



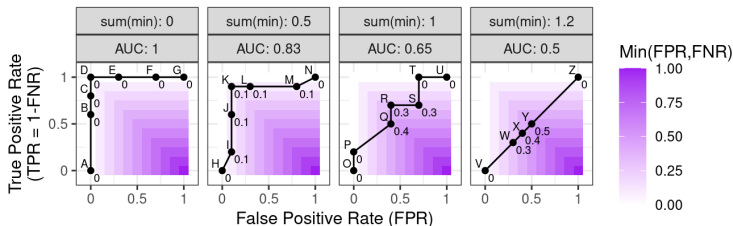
- ▶ Left: gradient 4x stronger along X axis for sample weights=1. Balanced sample weights mean equal influence for gradients along both axes, based on the current prediction threshold.
- ▶ Right: proposed method computes gradients based on all ROC points, not just the current prediction threshold.

Comparing proposed loss with baselines

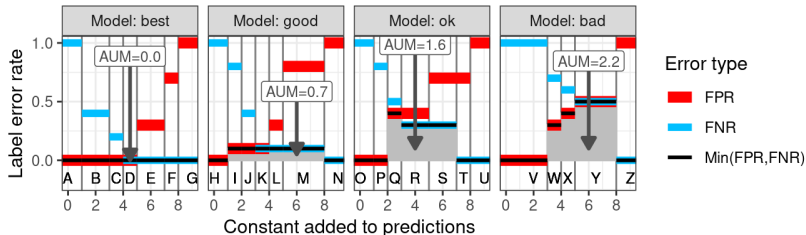
- ▶ Classic baselines: hinge and logistic loss, sum over samples, $\ell[yf(x)]$.
- ▶ Bamber (1975) proved ROC-AUC relation to Mann-Whitney U statistic (double sum over all pairs of positive and negative samples).
- ▶ Recently: SVM^{struct} (Joachims 2005), X-risk (Yang 2022), All Pairs Squared Hinge (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)

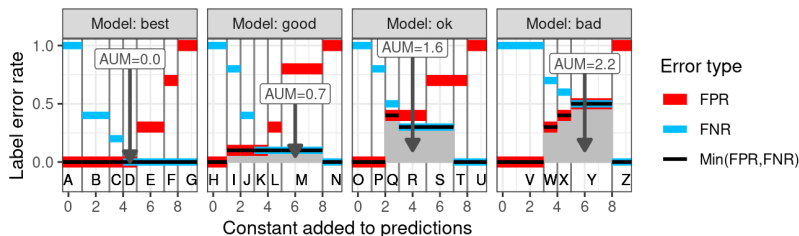


Above: purple heat map = numbers near dots = distance to top or left
= same as black min error rate functions below.



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

Computing Sum of Min (SM)

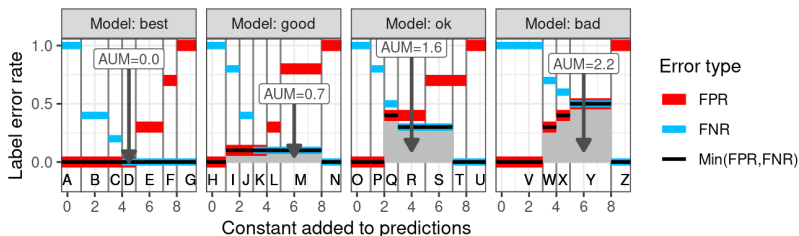


- ▶ For N samples, there are $\leq N + 1$ points on the ROC curve,
- ▶ with sorted thresholds $T_1 \leq \dots \leq T_N \in \mathbb{R}$,
- ▶ and corresponding min error values M_1, \dots, M_N .
- ▶ Then if I is the indicator function, we can write the sum of the min (SM), over all ROC points, as:

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

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

Computing Area Under Min (AUM)



The 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 optimization, and it is differentiable almost everywhere \Rightarrow gradient descent learning!

ROC curve pytorch code

```
def ROC_curve(pred_tensor, label_tensor):  
    sorted_indices = torch.argsort(-pred_tensor)  
    ...  
    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])])  
    }
```

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

AUC and AUM pytorch code uses argsort

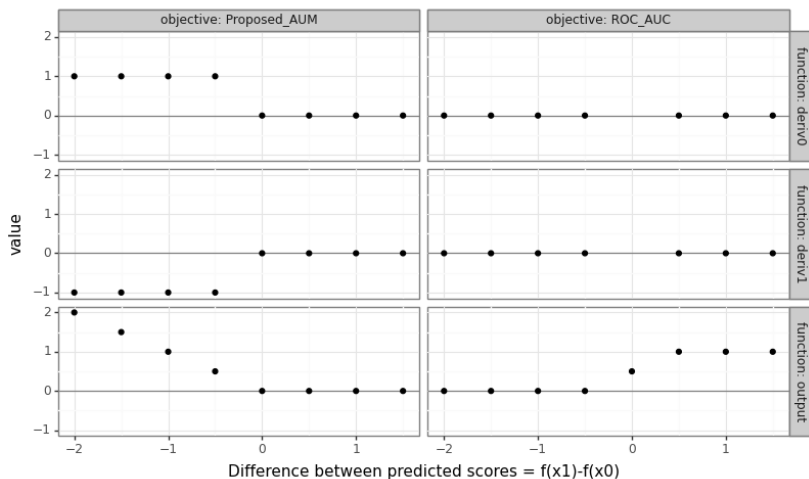
- ▶ ROC AUC and proposed AUM are both implemented by first computing the ROC curve.

```
def ROC_AUC(pred_tensor, label_tensor):  
    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):  
    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)
```

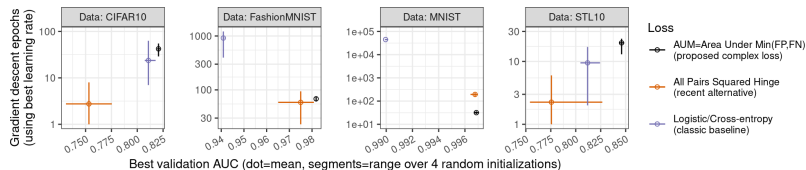
<https://tdhock.github.io/blog/2024/torch-roc-aum/>

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.



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

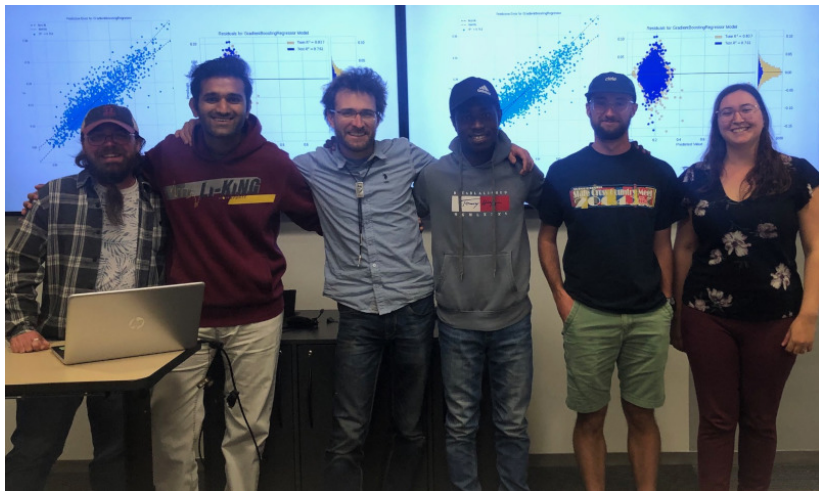


- ▶ Unbalanced binary classification: 10% negative, 90% positive.
- ▶ Gradient descent with constant step size, best of 10^{-4} to 10^5 .
- ▶ Full gradient (batch size = number of samples).
- ▶ Linear model, max iterations = 100,000.
- ▶ Max Validation AUC comparable or better than baselines: logistic loss and all paired squared hinge (like LibAUC/X-risk).
- ▶ 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 baselines are simple (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.
- ▶ Margin-based algorithms like SVM?
- ▶ Works well in binary classification, how to adapt to multi-class setting, or 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!



Contact: toby.dylan.hocking@usherbrooke.ca