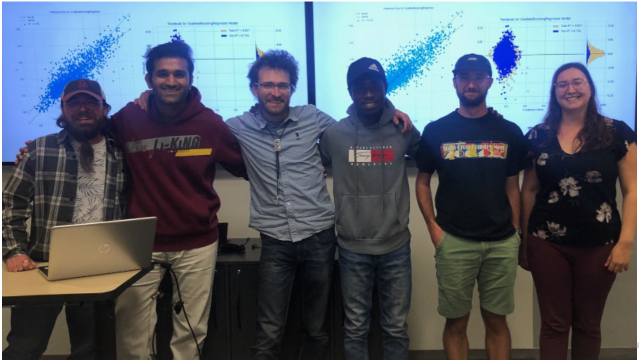
Two new algorithms for scientific applications of machine learning

Toby Dylan Hocking — toby.dylan.hocking@usherbrooke.ca
Learning Algorithms, Statistical Software, Optimization (LASSO Lab) — https://lassolab.org
Dpartement d'Informatique, Universit de Sherbrooke



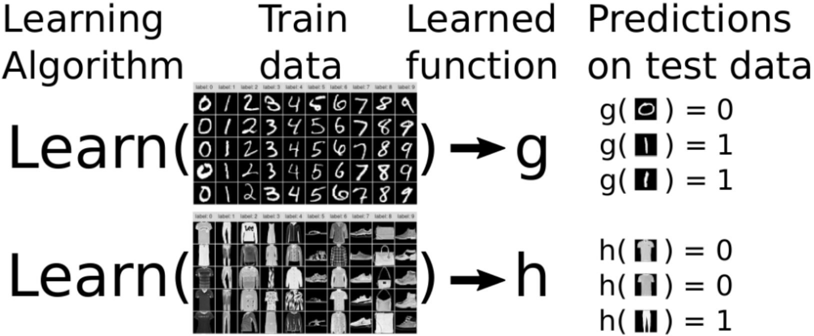
Introduction: two common questions in collaborations involving applications of machine learning

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

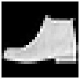
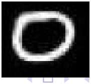
AUM: Area Under Min(FPR,FNR), a new differentiable loss for ROC curve optimization

Learning two different functions using two data sets

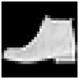
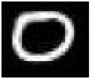
Figure from chapter by Hocking TD, *Introduction to machine learning and neural networks* for book *Land Carbon Cycle Modeling: Matrix Approach, Data Assimilation, and Ecological Forecasting* edited by Luo Y (Taylor and Francis, 2022).



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

Q: what happens if you do *g*(), or *h*()?

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($$)$, or $h($$)$? (**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?

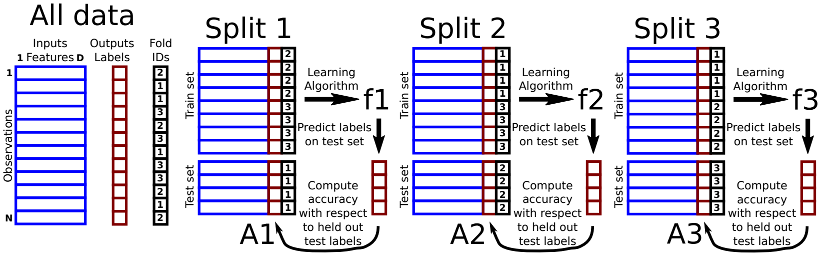
Introduction: two common questions in collaborations involving applications of machine learning

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

AUM: Area Under Min(FPR,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.



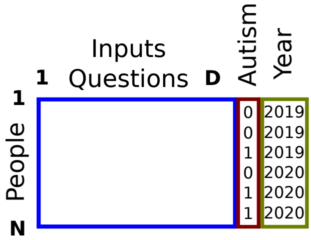
Hocking TD *Intro. to machine learning and neural networks* (2022).

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)

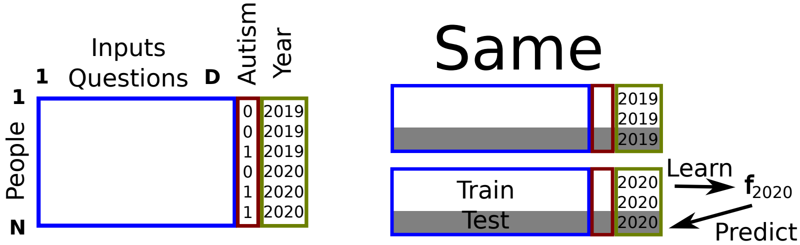
- Example: childhood autism prediction data set.



Autism	Year
0	2019
0	2019
1	2019
0	2020
1	2020
1	2020

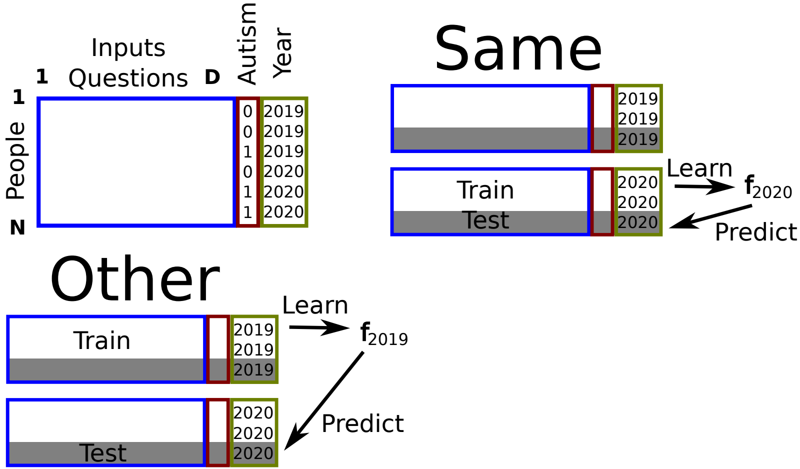
Proposed SOAK algorithm (Autism data example)

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



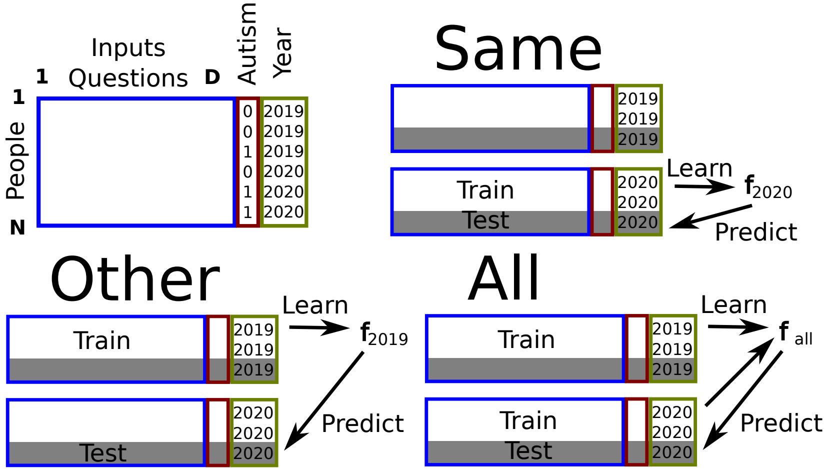
Proposed SOAK algorithm (Autism data example)

- Test subset fixed (2020), train on other subset/year (2019).



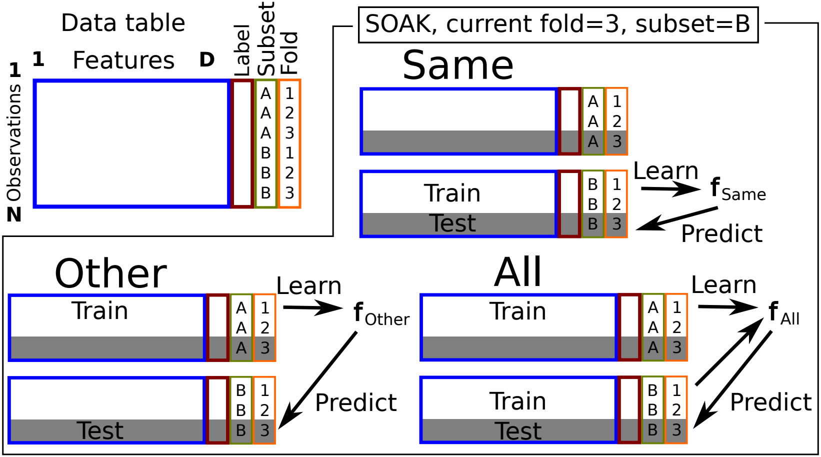
Proposed SOAK algorithm (Autism data example)

- Train set includes data from both subset/years (2019, 2020).



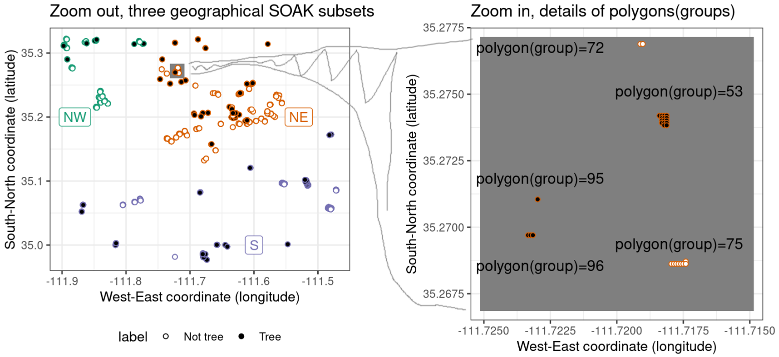
Proposed SOAK algorithm (generic data)

- Key new idea is **subset** column in data table.
- 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)



Proposed SOAK algorithm, interpretation/expectations

For a fixed test set from one subset:
If subsets have similar learnable/predictable patterns,
(this is a weaker condition than i.i.d.)

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 subsets have different learnable/predictable patterns,

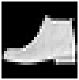
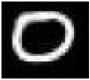
Same should be most accurate.

Other should be substantially less accurate.

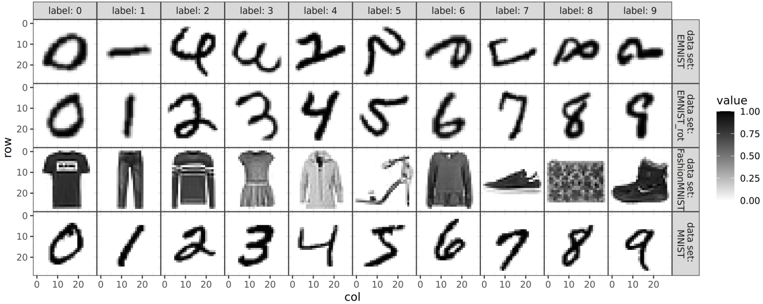
All accuracy should be between Same and Other.

- 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.

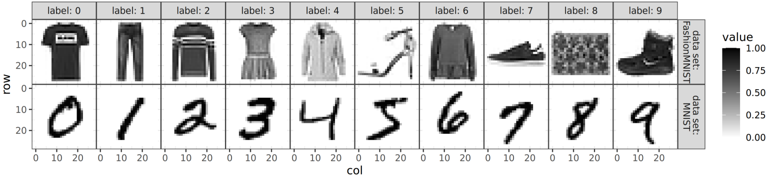
ImagePair data: train on MNIST and accurately predict on MNIST variants?

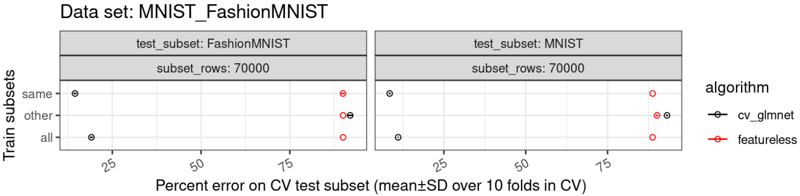
Recall: what happens if you do $g($$)$, or $h($$)$?

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



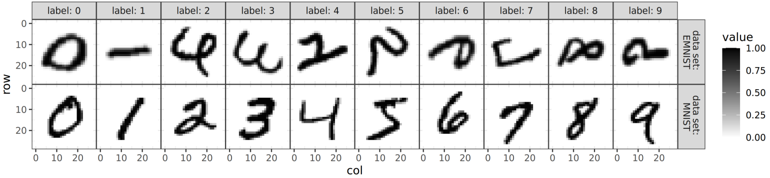
SOAK for MNIST+FashionMNIST data

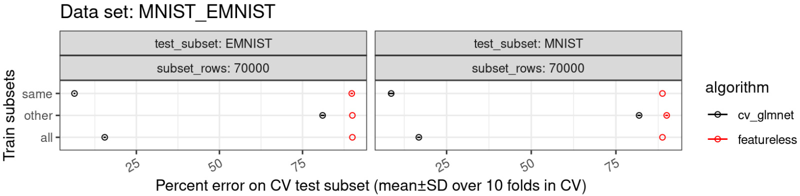




- **cv_glmnet** is L1-regularized linear model.
- **featureless** baseline always predicts most frequent class.
- **Other cv_glmnet** has greater test error than **featureless**, which indicates that the patterns are too different for the linear model to learn anything useful for prediction.

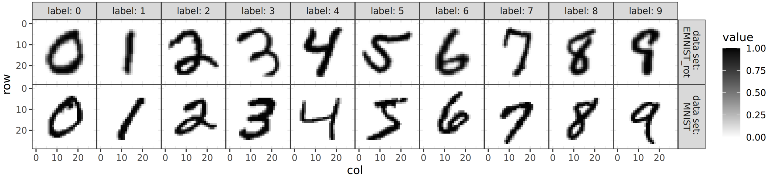
SOAK for MNIST+EMNIST data

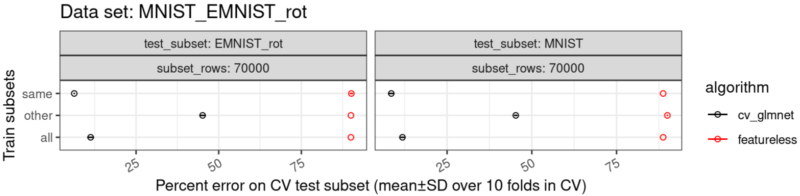




- **Other cv_glmnet** has significantly smaller test error than **featureless**, so something is learned/transferable between data sets, but it is still clear that the pattern is very different.

SOAK for MNIST+EMNIST_rot data



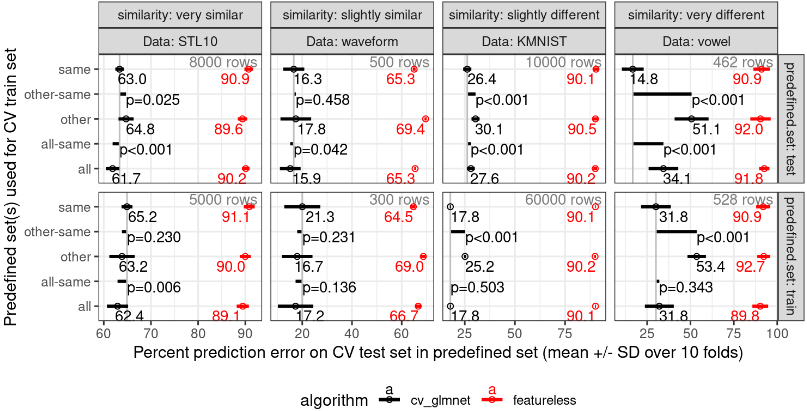


- **Other cv_glmnet** has even smaller test error, indicating even more similarity between MNIST and EMNIST_rot data sets.
- But **Other/All** test error are still not as small as **Same**, indicating there is a significant difference between learnable/predictable patterns in MNIST/EMNIST_rot data.

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 MNIST is a data set of images of handwritten digits (want to predict which digit, 0 to 9), with 60,000 images pre-defined as train, and 10,000 images pre-defined as testc.
- spam is a data set of emails (want to predict spam or not, binary), with 3065 train and 1536 test emails.
- Are the learnable/predictable patterns in the pre-defined train/test subsets similar?
- Or are they different?

Benchmark data with pre-defined train/test subsets

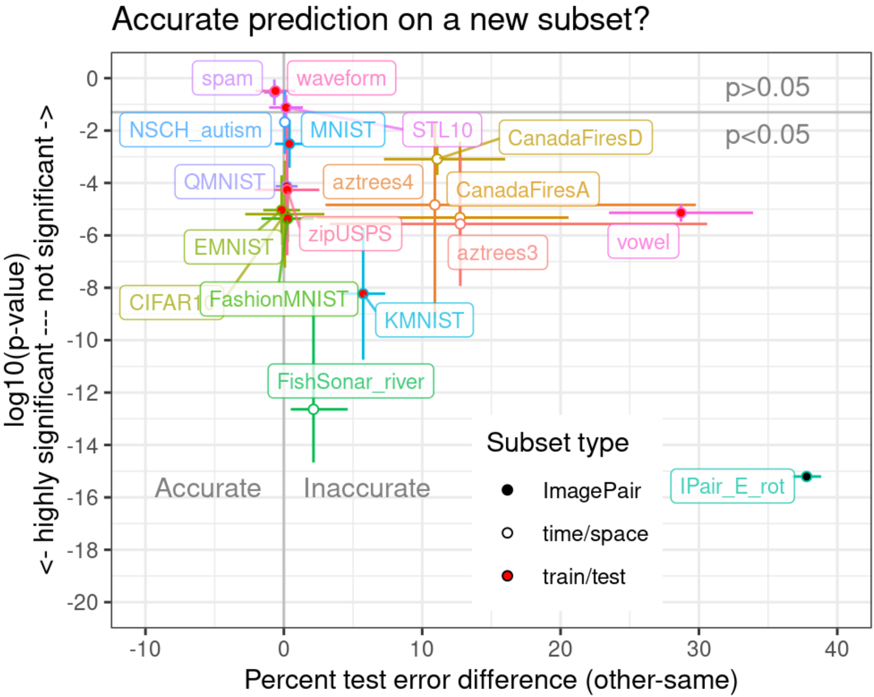


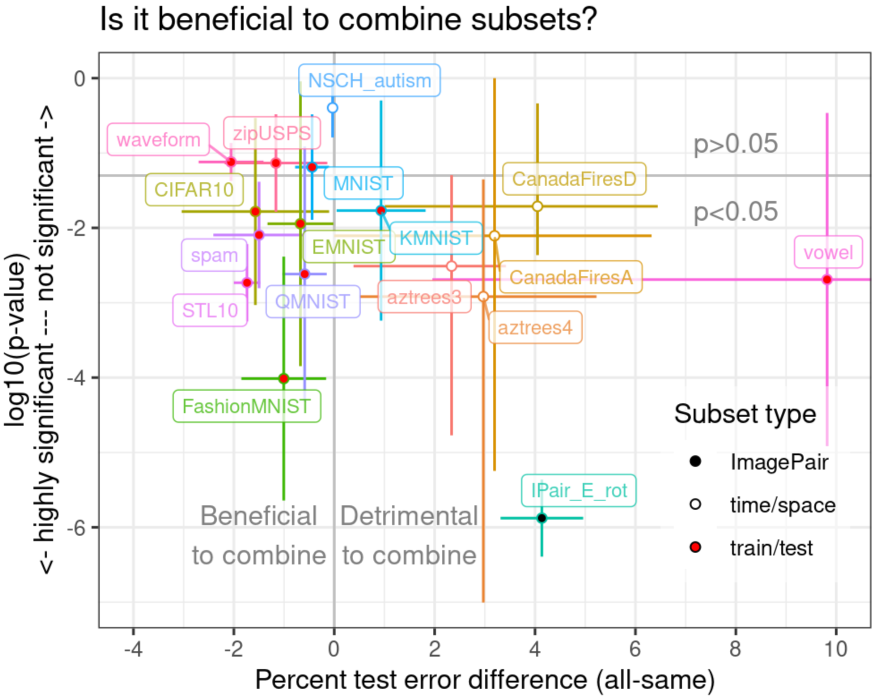
- Use pre-defined train/test subsets in SOAK, to see if patterns are learnable/predictable in different pre-defined subsets.
- Left, similar subsets, all error less than same. (expected)
- Right, different subsets, all error greater than same. (surprising)

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?
p>0.05
p<0.05
spam
waveform
NSCH_autism
MNIST
STL10
CanadaFiresD
QMNIST
aztrees4
CanadaFiresA
vowel
zipUSPS
EMNIST
aztrees3
CIFAR10
FashionMNIST
KMNIST
FishSonar_river
IPair_E_rot
Accurate
Inaccurate
Subset type
● ImagePair
○ time/space
● train/test
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-10
0
10
20
30
40
log10(p-value)
<- highly significant --- not significant ->
Percent test error difference (other-same)


Is it beneficial to combine subsets?
log10(p-value)
<- highly significant --- not significant ->
Percent test error difference (all-same)
0
-2
-4
-6
-4
-2
0
2
4
6
8
10
p>0.05
p<0.05
Beneficial
to combine
Detrimental
to combine
NSCH_autism
waveform
zipUSPS
MNIST
CanadaFiresD
CIFAR10
EMNIST
KMNIST
spam
vowel
CanadaFiresA
STL10
QMNIST
aztrees3
aztrees4
FashionMNIST
IPair_E_rot
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 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`

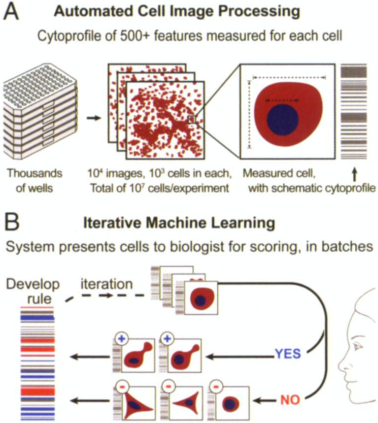
Introduction: two common questions in collaborations involving applications of machine learning

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

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

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.

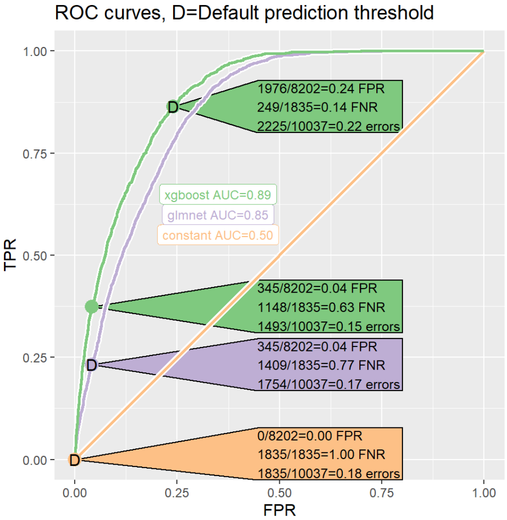

A
Automated Cell Image Processing
Cytoprofile of 500+ features measured for each cell
Thousands of wells
10⁴ images, 10³ cells in each, Total of 10⁷ cells/experiment
Measured cell, with schematic cytoprofile
B
Iterative Machine Learning
System presents cells to biologist for scoring, in batches
Develop rule
iteration
YES
NO

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).

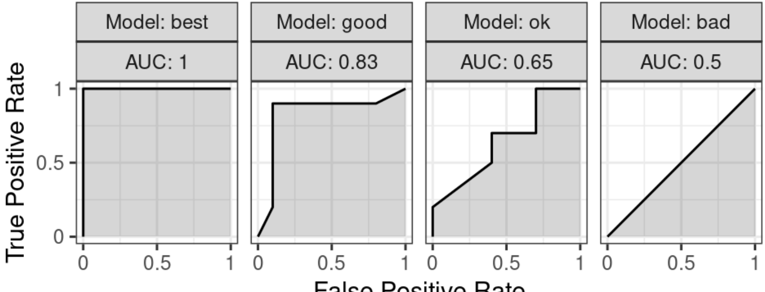
ROC curves for evaluation, especially useful with imbalance



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

Receiver Operating Characteristic (ROC) Curves

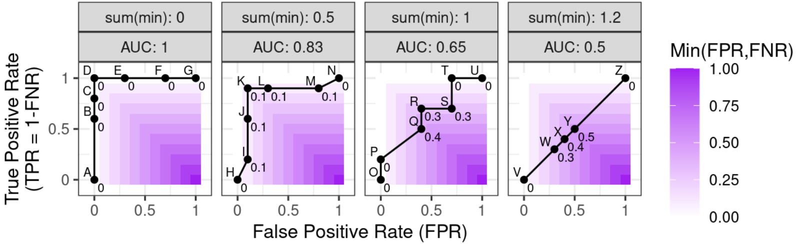
- Classic evaluation method from the signal processing literature (Egan and Egan, 1975).
- ROC curve of learned f is plot of True Positive Rate vs False Positive Rate: each point on the ROC curve is a different constant $c \in \mathbb{R}$ added to the predicted values: $f(\mathbf{x}) + c$.
- $c = \infty$ 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).



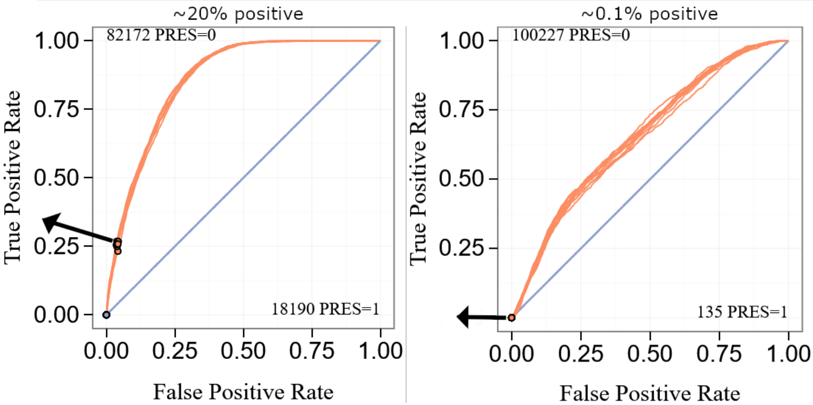
Research question and new idea

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(FPR,FNR).

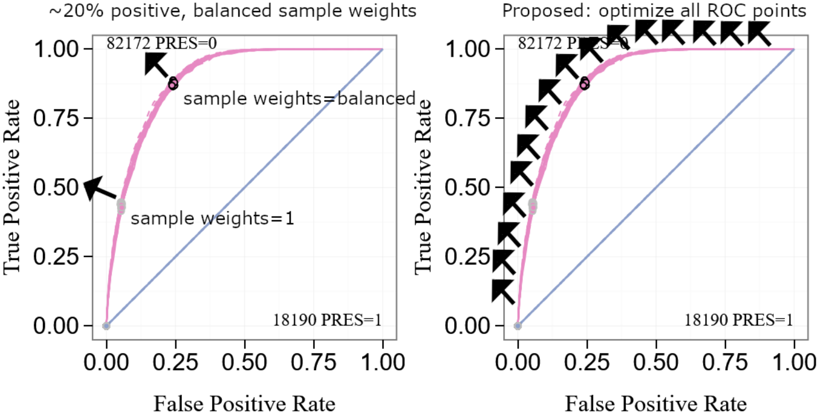


Gradients of sample-based loss are influenced by imbalance



- 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)

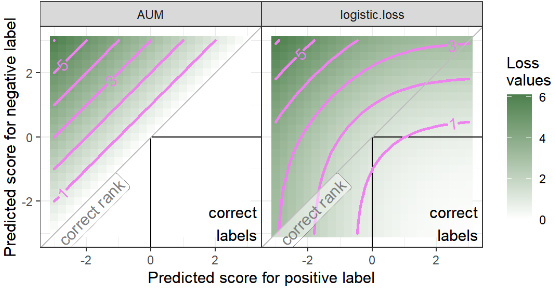
Gradients using balanced sample weights, proposed loss



- 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.

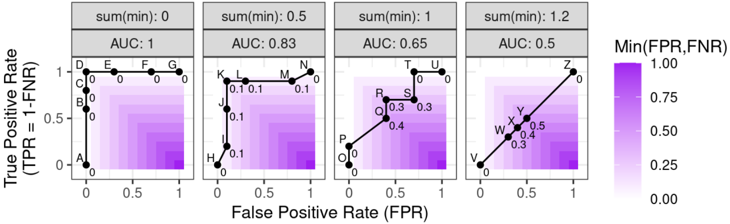
Comparing proposed loss with baselines

- 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$^{\text{struct}}$ (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.

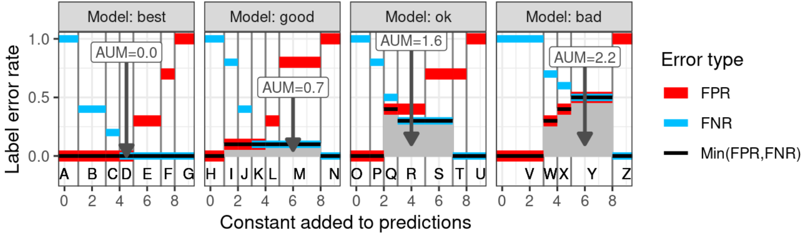


Barr, Hocking, Morton, Thatcher, Shaw, *TransAI* (2022).

Large AUC ≈ small Area Under Min(FP,FN) (AUM)

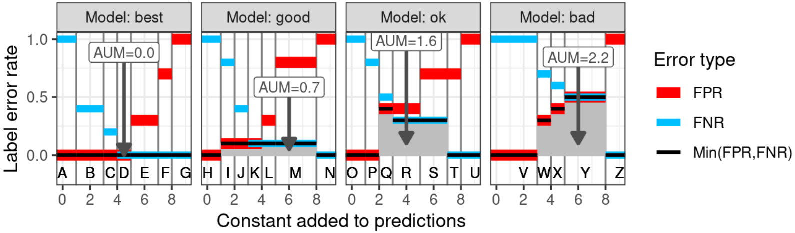


Above: purple heat map = numbers near dots = distance to top or left = same as black min error rate functions below.



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

Computing Sum of Min (SM)

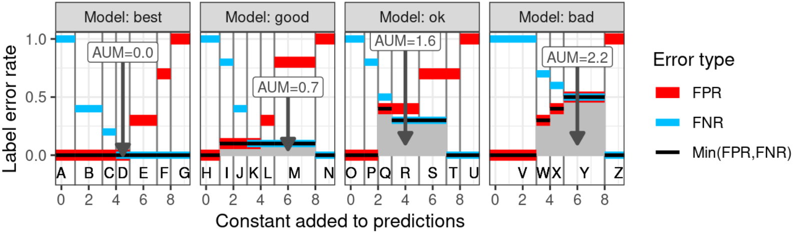


- For N samples, there are $\leq N+1$ points on the ROC curve,
- with sorted thresholds $T_1 \leq \cdots \leq T_N \in \mathbb{R}$,
- and corresponding min error values $M_1, \dots, M_N$.
- Then if I is the indicator function, we can write the sum of the min (SM), over all ROC points, as:

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

($\neq$ 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,

$$\begin{aligned} \text{SM} &= \sum_{i=2}^{N} I[T_i \neq T_{i-1}] M_i = \sum_{i: T_i \neq T_{i-1}} M_i. \\ \text{AUM} &= \sum_{i=2}^{N} [T_i - T_{i-1}] M_i. \end{aligned}$$

AUM is therefore a surrogate loss for ROC optimization, and it is differentiable almost everywhere ⇒ gradient descent learning!

ROC curve pytorch code uses argsort

```
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])])  }
>>> 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
```

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

AUC and AUM both use ROC curve

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

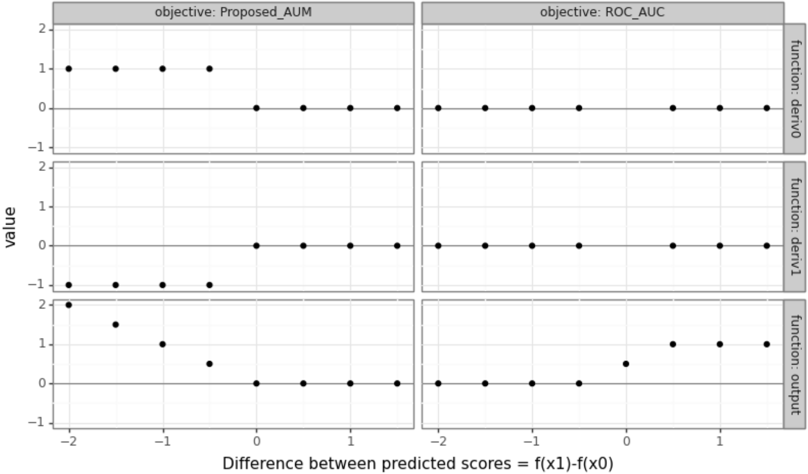
```
def ROC_AUC(pred_tensor, label_tensor):
    roc = ROC_curve(pred_tensor, label_tensor)
    FPR_diff = roc["FPR"][1:]-roc["FPR"][:-1]
    TPR_sum = roc["TPR"][1:]+roc["TPR"][:-1]
    return torch.sum(FPR_diff*TPR_sum/2.0)

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/

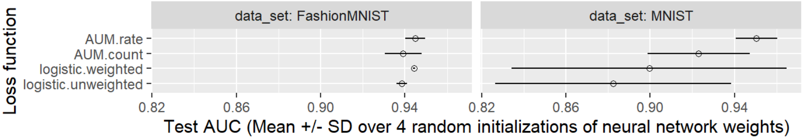
AUM pytorch code, auto-grad demo

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



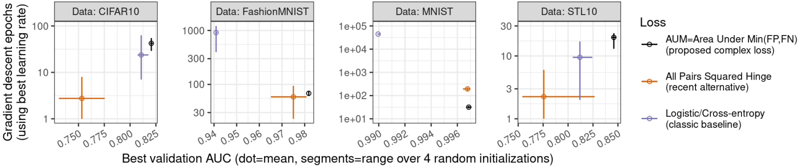
`https://tdhock.github.io/blog/2024/torch-roc-aum/`

Comparing 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.

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