Exact Line Search for AUM

Description & Preliminary Results

Jadon Fowler @ Northern Arizona University
with Dr. Toby Dylan Hocking

Contents

Receiver Operating Characteristic curves and optimizing AUC.

A differentiable surrogate for AUC known as AUM.

Optimizing AUM using gradient descent.

Different line search methods for AUM.

My research: an exact line search algorithm.

Future work to be done this semester.

Area Under the ROC Curve (AUC)

- Setup: model outputs a prediction vector given some examples, comparing the values to a threshold of zero to get classifications.
- Different points on the ROC curve are obtained by adding a constant to the predicted values (TPR vs FPR).
- AUC is an evaluation metric which accounts for every possible threshold
- AUC can be interpreted as the probability of ranking a positive example higher than a negative example

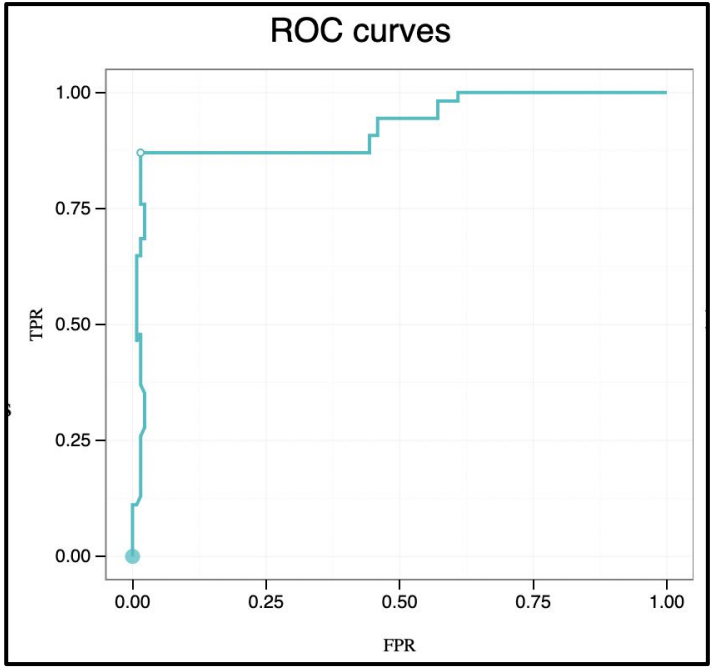
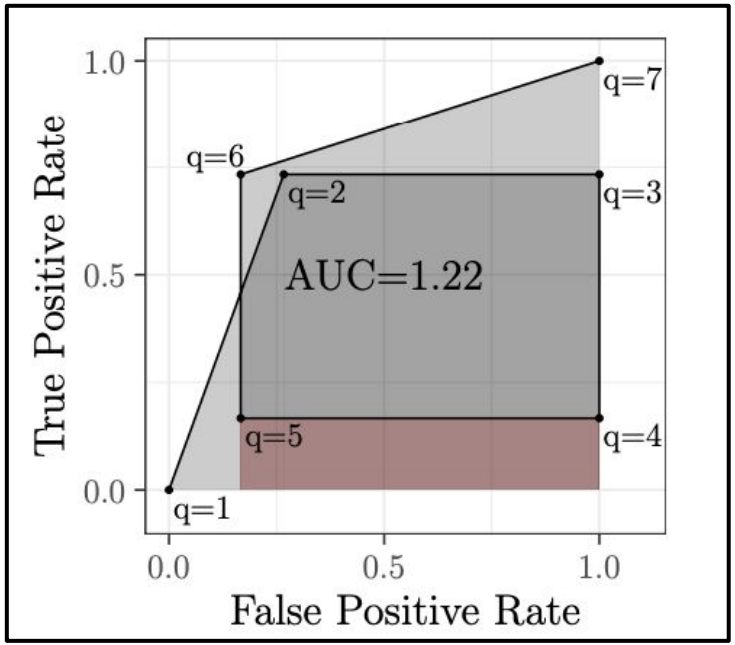


Figure from Dr. Hocking

Non-convex ROC curves

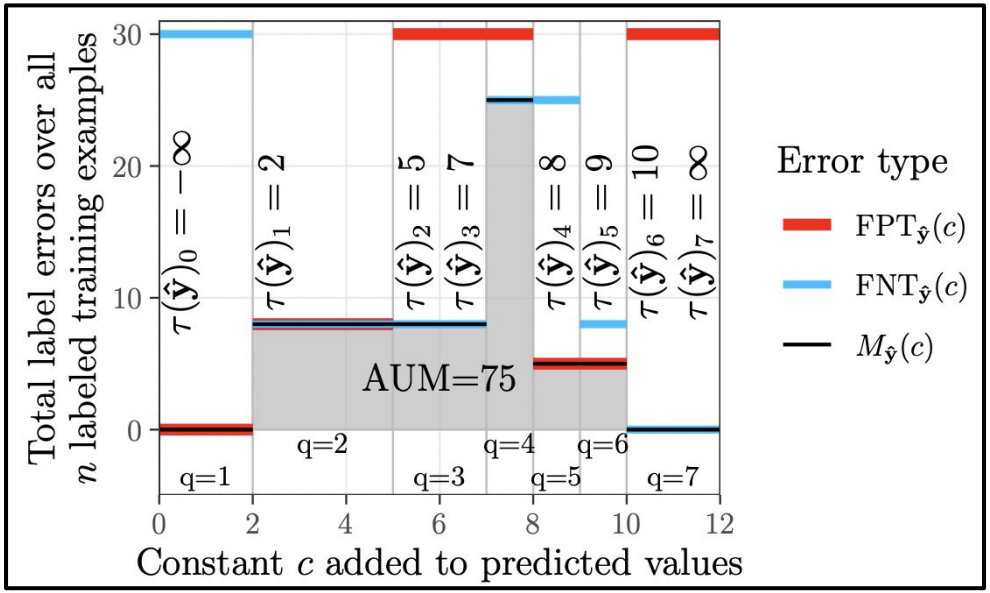


- In binary classification problems, there's a relationship between the increasing the classification thresholding and the TPR/FPR.
- In change point detection, gradual increases to the constant added to this prediction vector don't imply gradual increases to ROC points.
- Calculating AUC can give us values > 1

Hillman & Hocking 2021

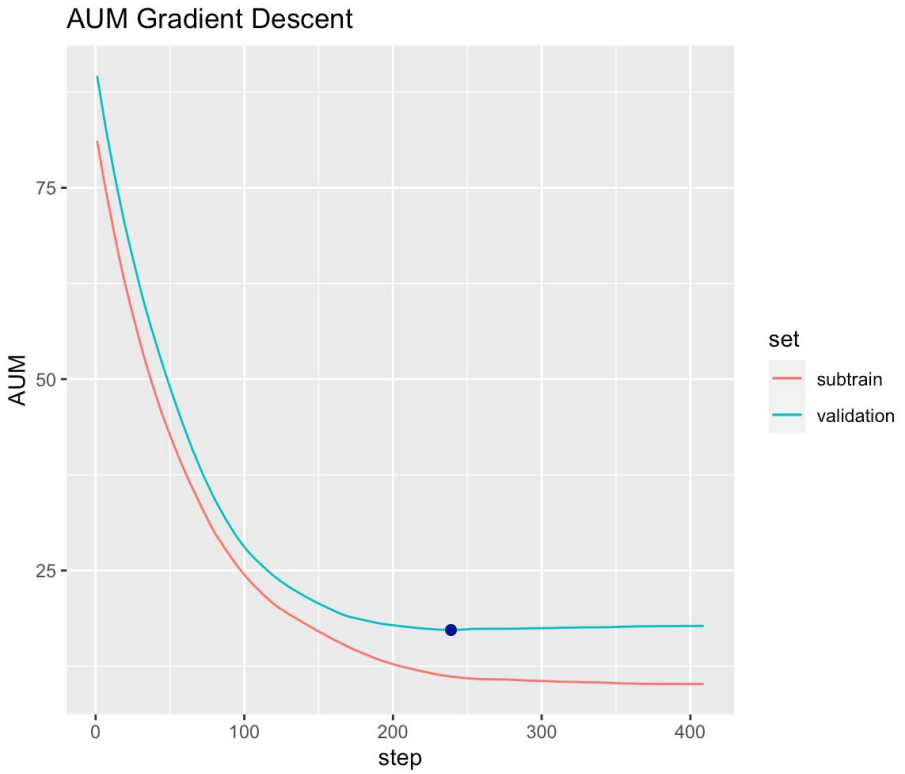
Area Under the Minimum(FP, FN)

- Differentiable surrogate loss function for AUC that optimizes the individual ROC points
- AUM(c) = Area under the Minimum of the False Positive Total and the False Negative Total for some constant c added to our prediction vector
- FPT / FNT change at breakpoints
- Shown than minimizing AUM corresponds to maximizing AUC
- Implementation of this in Dr. Hocking's `aum` package



Gradient Descent

- Each step of gradient descent decreases the AUM
- Split into test / test sets (and further subtrain / validation)
- Functionality to calculate gradients provided by the `aum` package
- Problem: how do we calculate the *step size*?



Line Search Methods

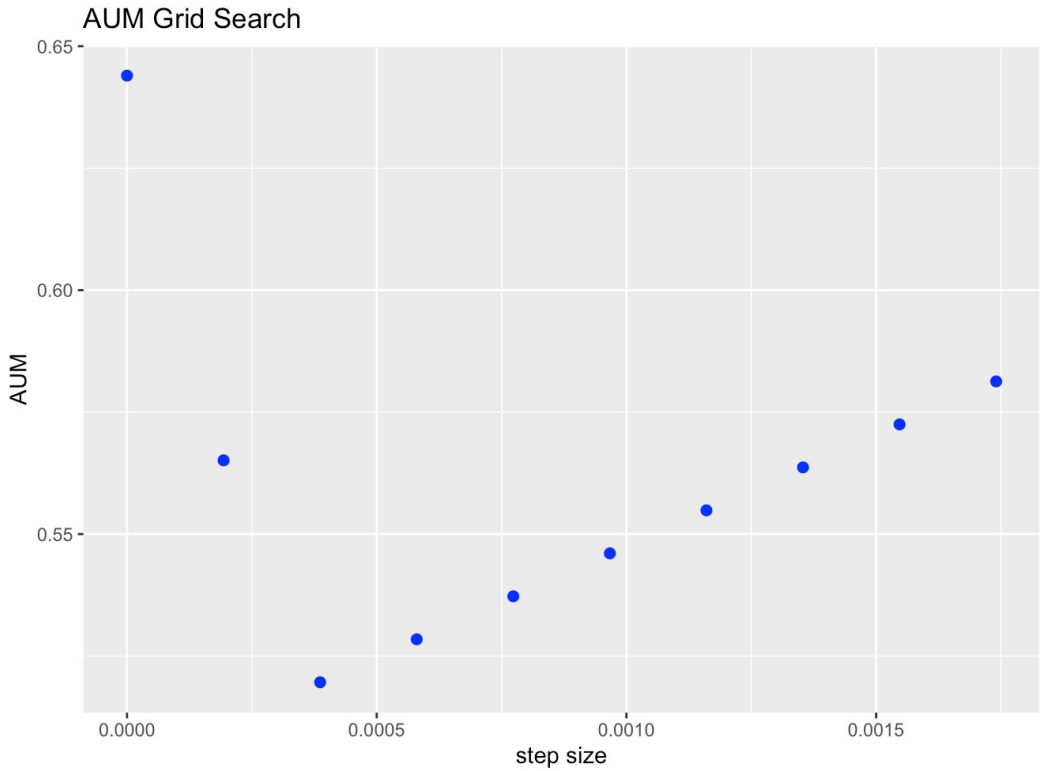
Exact Line Search
vs. Inexact Line Search

Different methods for obtaining a good step size:

- Constant Step Size
- Step Halving
- *Grid Search*
- *Exact Line Search* *(my research!)*

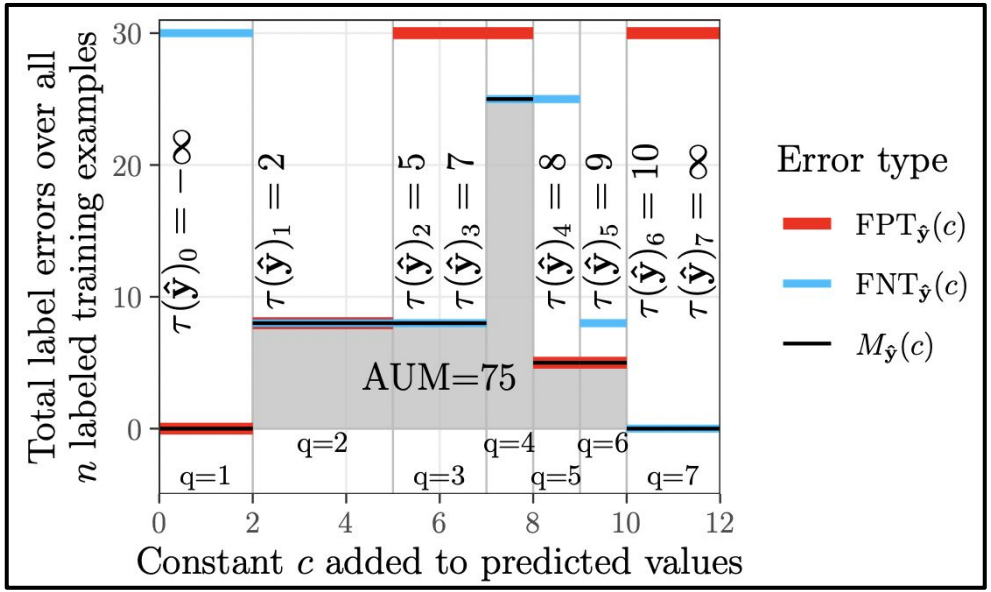
Grid Search

- Searches a “grid” of N points
- Each step size checked requires recalculating the AUM
- For `G` grid points, complexity is `O(G*n*log(n))`



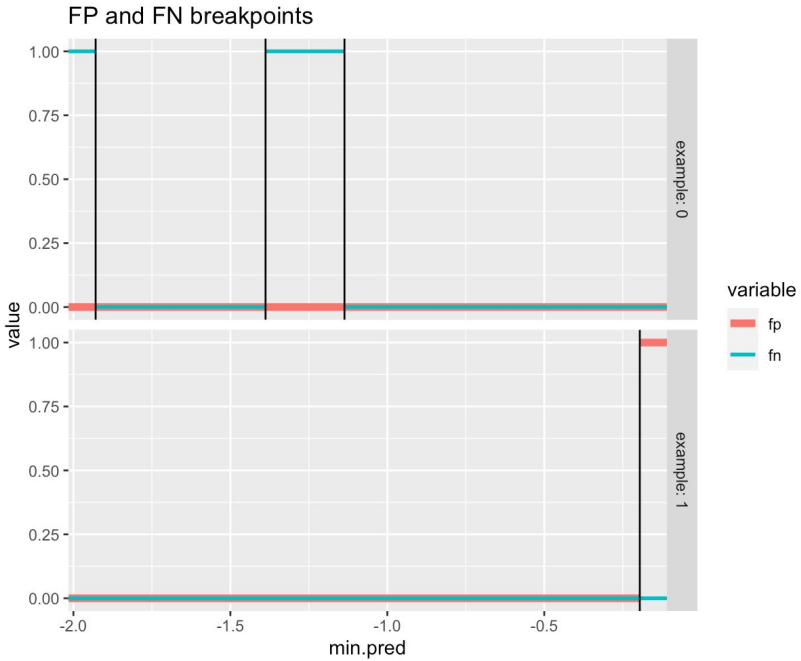
New Research: building a more efficient algorithm

- Use the piecewise nature of the FP & FN functions
- Every breakpoint has a function that describes how the FP & FN change with step size
- Idea: use threshold functions to calculate AUM faster

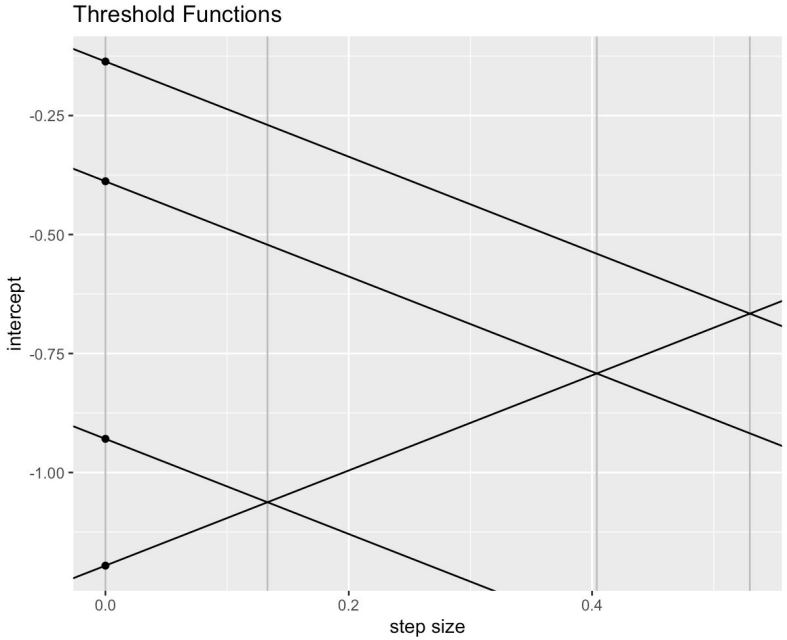


Breakpoints → Threshold Functions

Simple example: 4 breakpoints

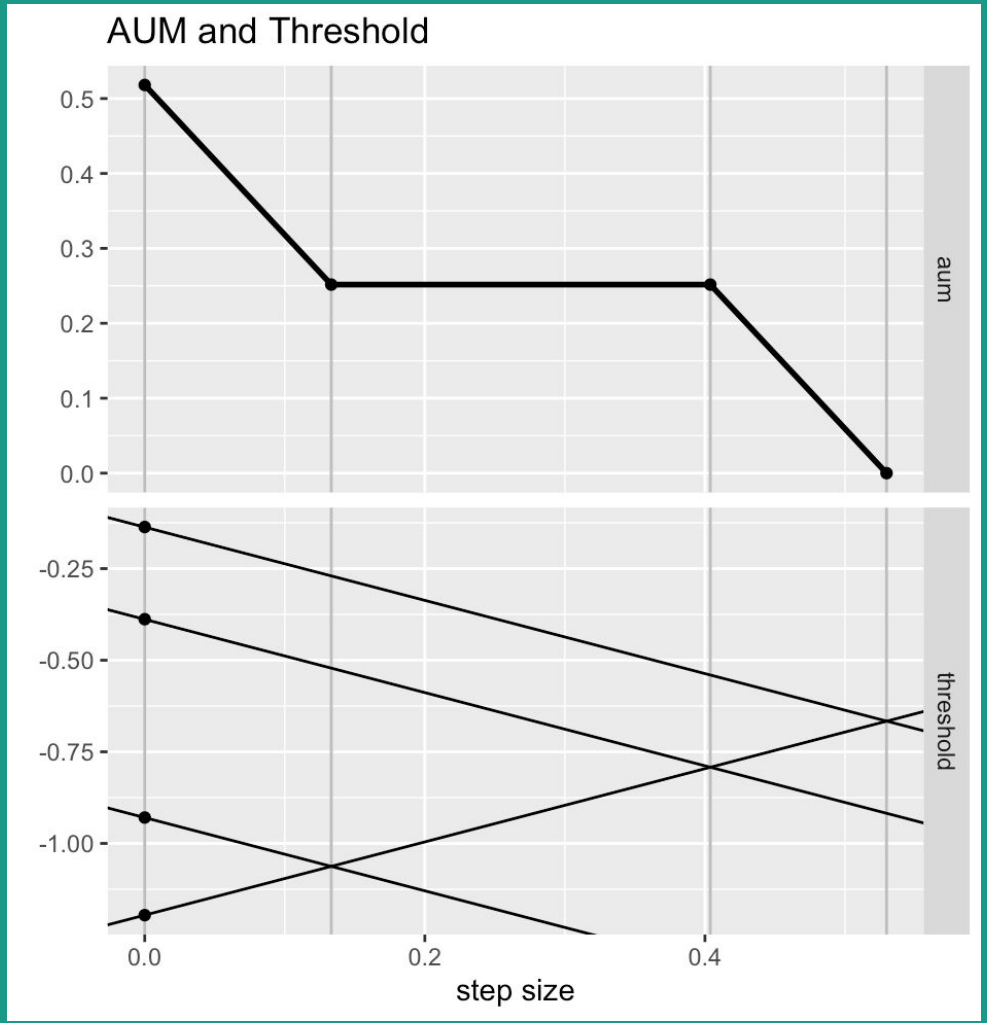


4 threshold functions



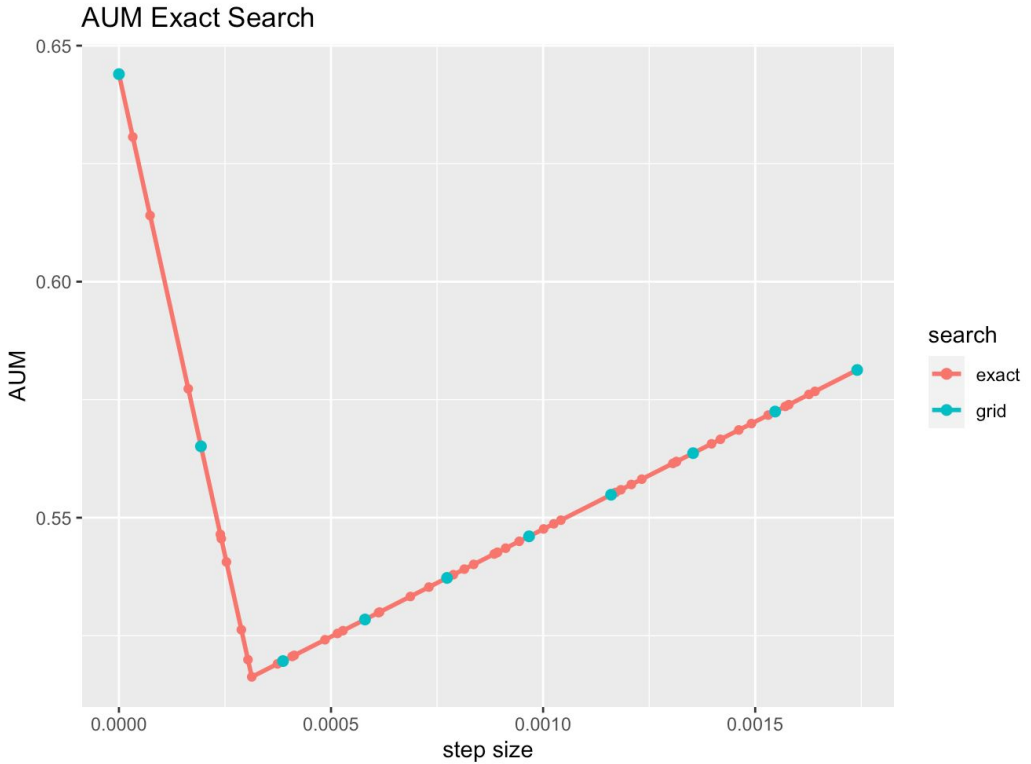
Threshold functions tell us where AUM changes

Intersections of these functions mark changes in the AUM slope.



Exact Line Search

- Best result so far: it works!
- Iterate up to `I` intersection points to find where AUM changes
- For `I` max iterations, complexity is `O(I*log(n))`
- Implemented in C++ in the `aum` package



Future Work

- Richer analysis with other methods
  - Time comparison, training on many datasets
  - Look at tradeoffs, some approaches may work at different stages of gradient descent
  - Compare against other ways of optimizing ROC
- Completing the WIP paper