Metrics Matter,
Examples from Binary and Multilabel Classification

Sanmi Koyejo

University of Illinois at Urbana-Champaign
Joint work with

B. Yan @UT Austin
K. Zhong @UT Austin
P. Ravikumar @CMU

N. Natarajan @MSR India
I. Dhillon @UT Austin
Goal: Train a DNN to optimize F-measure.

\[ F_1 = \frac{2TP}{2TP + FN + FP} \]
Learning with complex metrics

**Goal:** Train a DNN to optimize F-measure.

\[
F_1 = \frac{2TP}{2TP + FN + FP}
\]

- Direct optimization
- Mixed combinatorial optimization
- Convex lower bound
- Logloss + thresholding
Goal: Train a DNN to optimize F-measure.

\[ F_1 = \frac{2TP}{2TP + FN + FP} \]

- **Direct optimization**
  - F-measure is not an average. Naïve SGD is not valid
  - The sample F-measure is non-differentiable
- **Mixed combinatorial optimization**
- **Convex lower bound**
- **Logloss + thresholding**
Goal: Train a DNN to optimize F-measure.

\[ F_1 = \frac{2TP}{2TP + FN + FP} \]

- Direct optimization
- Mixed combinatorial optimization
  - e.g. cutting plane method (Joachims, 2005)
  - may require exponential complexity
  - most statistical properties unknown
- Convex lower bound
- Logloss + thresholding
Goal: Train a DNN to optimize F-measure.

\[ F_1 = \frac{2TP}{2TP + FN + FP} \]

- Direct optimization
- Mixed combinatorial optimization
- Convex lower bound
  - difficult to construct
  - most statistical properties unknown
- Logloss + thresholding
Goal: Train a DNN to optimize F-measure.

\[ F_1 = \frac{2TP}{2TP + FN + FP} \]

- Direct optimization
- Mixed combinatorial optimization
- Convex lower bound
- Logloss + thresholding
  - simple, most common approach in practice
  - has good statistical properties!
Goal: Train a DNN to optimize F-measure.

\[
F_1 = \frac{2TP}{2TP + FN + FP}
\]

- Direct optimization
- Mixed combinatorial optimization
- Convex lower bound
- Logloss + thresholding

Why does thresholding work?
The confusion matrix summarizes binary classifier mistakes

- $Y \in \{0, 1\}$ denotes labels, $X \in \mathcal{X}$ denotes instances, let $X, Y \sim P$
- The classifier $\theta : \mathcal{X} \mapsto \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>$Y = 1$</th>
<th>$Y = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta = 1$</td>
<td>TP $P(Y = 1, \theta = 1)$</td>
<td>FP $P(Y = 0, \theta = 1)$</td>
</tr>
<tr>
<td>$\theta = 0$</td>
<td>FN $P(Y = 1, \theta = 0)$</td>
<td>TN $P(Y = 0, \theta = 0)$</td>
</tr>
</tbody>
</table>
Metrics tradeoff which kinds of mistakes are (most) acceptable

A medical test determines that a patient has a 30% chance of having a fatal disease. Should the doctor treat the patient? Choosing not to treat a sick patient (test is false negative) could lead to serious issues. Choosing to treat a healthy patient (test is false positive) increases risk of side effects.
Case Study

A medical test determines that a patient has a 30% chance of having a fatal disease. Should the doctor treat the patient?

- Choosing not to treat a sick patient (test is false negative) could lead to serious issues.
- Choosing to treat a healthy patient (test is false positive) increases risk of side effects.
Performance metrics

We express tradeoffs via a metric $\Phi : [0, 1]^4 \rightarrow \mathbb{R}$
Performance metrics

We express tradeoffs via a metric $\Phi : [0, 1]^4 \rightarrow \mathbb{R}$

Examples

- Accuracy (fraction of mistakes) = $TP + TN$
- Error Rate = $1$-Accuracy = $FP + FN$
Performance metrics

We express tradeoffs via a metric $\Phi : [0, 1]^4 \rightarrow \mathbb{R}$

Examples

- Accuracy (fraction of mistakes) = $\text{TP} + \text{TN}$
- Error Rate = 1-Accuracy = $\text{FP} + \text{FN}$
- For medical diagnosis example, consider the weighted error = $w_1 \text{FP} + w_2 \text{FN}$, where $w_2 \gg w_1$
Performance metrics

We express tradeoffs via a *metric* $\Phi : [0,1]^4 \to \mathbb{R}$

**Examples**

- **Accuracy** (fraction of mistakes) = $\frac{TP}{TP + TN}$
- **Error Rate** = $1$-Accuracy = $\frac{FP}{TP + FN}$
- For medical diagnosis example, consider the weighted error = $w_1 FP + w_2 FN$, where $w_2 \gg w_1$

and many more . . .

$$\text{Recall} = \frac{TP}{TP + FN}, \quad F_\beta = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2 FN + FP},$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Jaccard} = \frac{TP}{TP + FN + FP}.$$
No need for metrics if you can learn a "perfect" classifier!

When is a perfect classifier learnable?
The true mapping between input and labels is deterministic i.e. there is no noise. The function class is sufficient (realizability) and optimal is computable. We have sufficient data.

In practice: real-world uncertainty e.g. hidden variables, measurement error. The true function is unknown, optimization may be intractable. Data are limited.

Thus, in most realistic scenarios, all classifiers will make mistakes!
No need for metrics if you can learn a "perfect" classifier!

When is a perfect classifier learnable?
No need for metrics if you can learn a "perfect" classifier!

When is a *perfect* classifier learnable?

- the *true* mapping between input and labels is deterministic i.e. there is no noise
- function class is sufficiently flexible (realizability) *and* optimal is computable
- we have sufficient data

In practice: real-world uncertainty e.g. hidden variables, measurement error, true function is unknown, optimization may be intractable, data are limited. Thus, in most realistic scenarios, all classifiers will make mistakes!
No need for metrics if you can learn a "perfect" classifier!

When is a perfect classifier learnable?

- the true mapping between input and labels is deterministic i.e. there is no noise
- function class is sufficiently flexible (realizability) and optimal is computable
- we have sufficient data

In practice:

- real-world uncertainty e.g. hidden variables, measurement error
- true function is unknown, optimization may be intractable
- data are limited
No need for metrics if you can learn a "perfect" classifier!

When is a *perfect* classifier learnable?

- the *true* mapping between input and labels is deterministic i.e. there is no noise
- function class is sufficiently flexible (realizability) and optimal is computable
- we have sufficient data

In practice:

- real-world uncertainty e.g. hidden variables, measurement error
- true function is unknown, optimization may be intractable
- data are limited

Thus, in most realistic scenarios, all classifiers will make mistakes!
Utility & Regret

- population performance is measured via utility

\[ U(\theta, P) = \Phi(TP, FP, FN, TN) \]

- we seek a classifier that *maximizes* this utility within some function class \( \mathcal{F} \)
population performance is measured via utility

$$U(\theta, P) = \Phi(TP, FP, FN, TN)$$

we seek a classifier that *maximizes* this utility within some function class $\mathcal{F}$

The *Bayes optimal* classifier, when it exists, is given by:

$$\theta^* = \arg\max_{\theta \in \Theta} U(\theta, P), \text{ where } \Theta = \{ f : \mathcal{X} \mapsto \{0, 1\} \}$$
Utility & Regret

- population performance is measured via utility

\[ \mathcal{U}(\theta, P) = \Phi(TP, FP, FN, TN) \]

- we seek a classifier that \textit{maximizes} this utility within some function class \( \mathcal{F} \)

The \textit{Bayes optimal} classifier, when it exists, is given by:

\[ \theta^* = \text{argmax}_{\theta \in \Theta} \mathcal{U}(\theta, P), \quad \text{where} \quad \Theta = \{ f : \mathcal{X} \mapsto \{0, 1\} \} \]

The \textit{regret} of the classifier \( \theta \) is given by:

\[ \mathcal{R}(\theta, P) = \mathcal{U}(\theta^*, P) - \mathcal{U}(\theta, P) \]
Towards analysis of the classification procedure

- In practice $P(X, Y)$ is unknown, instead we observe $\mathcal{D}_n = \{(X_i, Y_i) \sim P\}_{i=1}^n$
- The classification procedure estimates a classifier $\theta_n|\mathcal{D}_n$
Towards analysis of the classification procedure

- In practice $P(X, Y)$ is unknown, instead we observe
  \[ \mathcal{D}_n = \{(X_i, Y_i) \sim P\}_{i=1}^n \]
- The classification procedure estimates a classifier $\theta_n | \mathcal{D}_n$

Example

Empirical risk minimization via SVM:

\[
\theta_n = \text{sign} \left( \arg\min_{f \in \mathcal{H}_k} \sum_{\{x_i, y_i\} \in \mathcal{D}_n} \max (0, 1 - y_i f(x_i)) \right)
\]
Consistency

Consider the sequence of classifiers \( \{ \theta_n(x), \ n \to \infty \} \)

A classification procedure is consistent when \( \mathcal{R}(\theta_n, P) \xrightarrow{n \to \infty} 0 \) i.e. the procedure is eventually Bayes optimal
Consider the sequence of classifiers \( \{ \theta_n(x), \ n \to \infty \} \)

A classification procedure is consistent when \( R(\theta_n, P) \xrightarrow{n \to \infty} 0 \) i.e. the procedure is eventually Bayes optimal.

Consistency is a desirable property:
- implies stability of the classification procedure, related to generalization performance.
Optimal Binary classification with Decomposable Metrics
Consider the empirical accuracy:

\[
\text{ACC}(\theta, \mathcal{D}_n) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}_n} 1[y_i = \theta(x_i)]
\]
Consider the empirical accuracy:

\[
\text{ACC}(\theta, D_n) = \frac{1}{n} \sum_{(x_i, y_i) \in D_n} 1[y_i = \theta(x_i)]
\]

- Observe that the classification problem

\[
\min_{\theta \in \mathcal{F}} \text{ACC}(\theta, D_n)
\]

is a combinatorial optimization problem

- optimal classification is \textit{computationally hard} for non-trivial \(\mathcal{F}\) and \(D_n\)
Bayes Optimal Classifier

Population Accuracy

\[ \mathbb{E}_{X,Y \sim P} \left[ 1[Y = \theta(X)] \right] \]

- Easy to show that \( \theta^*(x) = \text{sign} \left( P(Y = 1|x) - \frac{1}{2} \right) \)
Bayes Optimal Classifier

Population Accuracy

$$\mathbb{E}_{X,Y \sim P} \left[ 1_{[Y = \theta(X) = 1]} \right]$$

- Easy to show that $$\theta^*(x) = \text{sign} \left( P(Y = 1 | x) - \frac{1}{2} \right)$$

Weighted Accuracy

$$\mathbb{E}_{X,Y \sim P} \left[ (1 - \rho)1_{[Y = \theta(X) = 1]} + \rho 1_{[Y = \theta(X) = 0]} \right]$$

- Scott (2012) showed that $$\theta^*(x) = \text{sign} \left( P(Y = 1 | x) - \rho \right)$$
Where do surrogates come from?

Observe that there is no need to estimate $P$, instead optimize any surrogate loss function $L(\theta, D_n)$ where:

$$\theta_n = \text{sign} \left( \arg\min_f L(f, D_n) \right) \xrightarrow{n \to \infty} \theta^*(x)$$

These are known as classification calibrated surrogate losses (Bartlett et al., 2003; Scott, 2012) research can focus on how to choose $L$, $F$ which improve efficiency, sample complexity, robustness... surrogates are often chosen to be convex e.g. hinge loss, logistic loss
Where do surrogates come from?

Observe that there is no need to estimate $P$, instead optimize any 
\textit{surrogate} loss function $L(\theta, D_n)$ where:

$$\theta_n = \text{sign} \left( \arg\min_f L(f, D_n) \right) \xrightarrow{n \to \infty} \theta^*(x)$$

- These are known as \textit{classification calibrated} surrogate losses (Bartlett et al., 2003; Scott, 2012)
Where do surrogates come from?

Observe that there is no need to estimate $P$, instead optimize any *surrogate* loss function $L(\theta, D_n)$ where:

$$\theta_n = \text{sign} \left( \arg\min_f L(f, D_n) \right) \xrightarrow{n \to \infty} \theta^*(x)$$

- These are known as *classification calibrated* surrogate losses (Bartlett et al., 2003; Scott, 2012)
- research can focus on how to choose $L, F$ which improve efficiency, sample complexity, robustness . . .
- surrogates are often chosen to be convex e.g. hinge loss, logistic loss
Non-decomposability

- A common theme so far is decomposability i.e. linearity wrt. confusion matrix

\[ E \left[ \Phi(\hat{C}) \right] = \left\langle A, E \left[ \hat{C} \right] \right\rangle = \Phi(E \left[ \hat{C} \right]) \]
Non-decomposability

- A common theme so far is *decomposability* i.e. linearity wrt. confusion matrix

\[
E \left[ \Phi(\hat{C}) \right] = \langle A, E \left[ \hat{C} \right] \rangle = \Phi(E \left[ \hat{C} \right])
\]

- However, \( F_\beta \), Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. \( C \)

- Thus implies that the *averaging trick* is no longer valid

\[
E \left[ \Phi(\hat{C}) \right] \neq \Phi(E \left[ \hat{C} \right])
\]
Non-decomposability

- A common theme so far is *decomposability* i.e. linearity wrt. confusion matrix

\[ E \left[ \Phi(\hat{C}) \right] = \langle A, E \left[ \hat{C} \right] \rangle = \Phi(E \left[ \hat{C} \right]) \]

- However, \( F_\beta \), Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. \( C \)

- Thus implies that the *averaging trick* is no longer valid

\[ E \left[ \Phi(\hat{C}) \right] \neq \Phi(E \left[ \hat{C} \right]) \]

- Primary source of difficulty for analysis, optimization, ...
Optimal Binary classification with Non-decomposable Metrics
The unreasonable effectiveness of thresholding

Theorem (Koyejo et al., 2014; Yan et al., 2016)

Let $\eta_x = P(Y = 1|X = x)$ and let $U$ be differentiable wrt. the confusion matrix, then $\exists$ a $\delta^*$ such that:

$$\theta^*(x) = \text{sign}(\eta_x - \delta^*)$$

is a Bayes optimal classifier almost everywhere.

\footnote{Condition: $P(\eta_x = \delta^*) = 0$, easily satisfied e.g. when $P(X)$ is continuous.}
The unreasonable effectiveness of thresholding

Theorem (Koyejo et al., 2014; Yan et al., 2016)

Let $\eta_x = P(Y = 1|X = x)$ and let $U$ be differentiable wrt. the confusion matrix, then $\exists$ a $\delta^*$ such that:

$$\theta^*(x) = \text{sign}(\eta_x - \delta^*)$$

is a Bayes optimal classifier almost everywhere.

- result does not require concavity of $U$, or other "nice" properties

\[\text{Condition: } P(\eta_x = \delta^*) = 0, \text{ easily satisfied e.g. when } P(X) \text{ is continuous.}\]
Proof Sketch

Let $\mathcal{F} = \{ f \mid f : \mathcal{X} \mapsto [0, 1] \}$ and $\Theta = \{ f \mid f : \mathcal{X} \mapsto \{0, 1\} \}$

Consider the relaxed problem:

$$\theta^*_F = \arg\max_{\theta \in F} U(\theta, \mathcal{P})$$
Proof Sketch

Let $\mathcal{F} = \{ f \mid f : X \mapsto [0, 1] \}$ and $\Theta = \{ f \mid f : X \mapsto \{0, 1\} \}$

- Consider the relaxed problem:
  \[
  \theta^*_\mathcal{F} = \operatorname{argmax}_{\theta \in \mathcal{F}} \mathcal{U}(\theta, \mathcal{P})
  \]

- Show that the optimal “relaxed” classifier is $\theta^*_\mathcal{F} = \text{sign}(\eta_x - \delta^*)$
Let $\mathcal{F} = \{ f \mid f : \mathcal{X} \mapsto [0, 1]\}$ and $\Theta = \{ f \mid f : \mathcal{X} \mapsto \{0, 1\}\}$

- Consider the relaxed problem:

$$
\theta^{\star}_\mathcal{F} = \arg\max_{\theta \in \mathcal{F}} U(\theta, \mathcal{P})
$$

- Show that the optimal “relaxed” classifier is $\theta^{\star}_\mathcal{F} = \text{sign}(\eta_x - \delta^{\star})$

- Observe that $\Theta \subset \mathcal{F}$. Thus $U(\theta^{\star}_\mathcal{F}, \mathcal{P}) \geq U(\theta^{\star}_\Theta, \mathcal{P})$.

- As a result, $\theta^{\star}_\mathcal{F} \in \Theta$ implies that $\theta^{\star}_\mathcal{F} \equiv \theta^{\star}_\Theta$. 
Some recovered and new results

<table>
<thead>
<tr>
<th>METRIC</th>
<th>FORM</th>
<th>OPTIMAL THRESHOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_\beta$</td>
<td>$\frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2FN + FP}$</td>
<td>$\delta^* = \frac{L^*}{1 + \beta^2}$</td>
</tr>
<tr>
<td>Cost-sensitive learning</td>
<td>$c_0 + c_1TP + c_2\gamma(\theta)$</td>
<td>$\delta^* = \frac{-c_2}{c_1}$</td>
</tr>
<tr>
<td>Precision</td>
<td>$\frac{TP}{TP + FP}$</td>
<td>$\delta^* = L^*$</td>
</tr>
<tr>
<td>Recall</td>
<td>$\frac{TP}{TP + FN}$</td>
<td>$\delta^* = 0$</td>
</tr>
<tr>
<td>Weighted Accuracy</td>
<td>$\frac{2(TP + TN)}{2(TP + TN) + FP + FN}$</td>
<td>$\delta^* = \frac{1}{2}$</td>
</tr>
<tr>
<td>Jaccard Coefficient</td>
<td>$\frac{TP}{TP + FP + FN}$</td>
<td>$\delta^* = \frac{L^<em>}{1 + L^</em>}$</td>
</tr>
</tbody>
</table>

$F_\beta$ (Ye et al., 2012), Monotonic metrics (Narasimhan et al., 2014)
Simulated examples

Finite sample space $\mathcal{X}$, so we can exhaustively search for $\theta^*$
Algorithm 1 (Koyejo et al., 2014)

Step 1: Conditional probability estimation

Estimate $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010), then

$$\hat{\theta}_\delta(x) = \text{sign}(\hat{\eta}_x - \delta)$$
Algorithm 1 (Koyejo et al., 2014)

Step 1: Conditional probability estimation
Estimate $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010), then

$$\hat{\theta}_\delta(x) = \text{sign}(\hat{\eta}_x - \delta)$$

Step 2: Threshold search

$$\max_\delta U(\hat{\theta}_\delta, \mathcal{D}_n)$$

One dimensional, efficiently computable using exhaustive search (Sergeyev, 1998).

$\hat{\theta}_\delta$ is consistent
Algorithm 2 (Koyejo et al., 2014)

Step 1: Weighted classifier estimation

For classification-calibrated loss (Scott, 2012)

\[ \hat{f}_\delta = \arg\min_{f \in \mathcal{F}} \sum_{x_i, y_i \in \mathcal{D}_n} \ell_\delta(f(x_i), y_i) \]

consistently estimates \( \hat{\theta}_\delta(x) = \text{sign}(\hat{f}_\delta(x)) \)

Step 2: Threshold search

\[ \max_{\delta} \mathcal{U}(\hat{\theta}_\delta, \mathcal{D}_n) \]

\( \hat{\theta}_\delta \) is consistent
Algorithm 3 (Yan et al., 2016)

Under additional assumptions, $\mathcal{U}(\theta_\delta, P)$ is differentiable and strictly locally quasi-concave wrt. $\delta$. 
Algorithm 3 (Yan et al., 2016)

Under additional assumptions, $U(\theta_\delta, P)$ is differentiable and strictly locally quasi-concave wrt. $\delta$

**Online Algorithm**

Iteratively update

1. $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010)
2. $\hat{\delta}_t$ using *normalized gradient ascent*
Let $\eta$ estimation error at step $t$ given by $r_t = \int |\eta_t - \eta| d\mu$, with appropriately chosen step size, $\mathcal{R}(\hat{\theta}_{\delta_t}, \mathcal{P}) \leq C \frac{\sum_{i=1}^{t} r_i}{t}$

**Example: Online logistic regression**

Parameter converges at rate $O(\frac{1}{\sqrt{n}})$ by averaged stochastic gradient algorithm (Bach, 2014). Thus, online algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.
Empirical Evaluation
Datasets

<table>
<thead>
<tr>
<th>datasets</th>
<th>default</th>
<th>news20</th>
<th>rcv1</th>
<th>epsilon</th>
<th>kdda</th>
<th>kddb</th>
</tr>
</thead>
<tbody>
<tr>
<td># features</td>
<td>25</td>
<td>1,355,191</td>
<td>47,236</td>
<td>2,000</td>
<td>20,216,830</td>
<td>29,890,095</td>
</tr>
<tr>
<td># test</td>
<td>9,000</td>
<td>4,996</td>
<td>677,399</td>
<td>100,000</td>
<td>510,302</td>
<td>748,401</td>
</tr>
<tr>
<td># train</td>
<td>21,000</td>
<td>15,000</td>
<td>20,242</td>
<td>400,000</td>
<td>8,407,752</td>
<td>19,264,097</td>
</tr>
<tr>
<td>%pos</td>
<td>22%</td>
<td>67%</td>
<td>52%</td>
<td>50%</td>
<td>85%</td>
<td>86%</td>
</tr>
</tbody>
</table>

- $\eta$ estimation: logistic regression and boosting tree
- Baselines: threshold search (Koyejo et al., 2014), SVM$^{perf}$ and STAMP/SPADE (Narasimhan et al., 2015)
### Batch algorithm

<table>
<thead>
<tr>
<th>Data set/Metric</th>
<th>LR+Plug-in</th>
<th>LR+Batch</th>
<th>XGB+Plug-in</th>
<th>XGB+Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>news20-Q-Mean</td>
<td>0.948 (3.77s)</td>
<td>0.948 (0.001s)</td>
<td>0.874 (3.87s)</td>
<td>0.875 (0.003s)</td>
</tr>
<tr>
<td>news20-H-Mean</td>
<td>0.950 (3.70s)</td>
<td>0.950 (0.003s)</td>
<td>0.859 (3.61s)</td>
<td>0.860 (0.003s)</td>
</tr>
<tr>
<td>news20-F1</td>
<td>0.949 (3.49s)</td>
<td>0.948 (0.01s)</td>
<td>0.872 (5.07s)</td>
<td>0.874 (0.01s)</td>
</tr>
<tr>
<td>default-Q-Mean</td>
<td>0.664 (14.3s)</td>
<td>0.667 (0.19s)</td>
<td>0.688 (13.7s)</td>
<td>0.701 (0.22s)</td>
</tr>
<tr>
<td>default-H-Mean</td>
<td>0.665 (12.1s)</td>
<td>0.668 (0.17s)</td>
<td>0.693 (12.4s)</td>
<td>0.708 (0.18s)</td>
</tr>
<tr>
<td>default-F1</td>
<td>0.503 (14.2s)</td>
<td>0.497 (0.19s)</td>
<td>0.538 (16.2s)</td>
<td>0.538 (0.15s)</td>
</tr>
</tbody>
</table>
Online Complex Metric Optimization (OCMO)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Algorithm</th>
<th>RCV1</th>
<th>Epsilon</th>
<th>KDD-A</th>
<th>KDD-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>OCMO</td>
<td>0.952 (0.01s)</td>
<td>0.804 (4.87s)</td>
<td>0.934 (2.43s)</td>
<td>0.941 (5.01s)</td>
</tr>
<tr>
<td></td>
<td>sTAMP</td>
<td>0.923 (14.44s)</td>
<td>0.585 (133.23s)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SVM&lt;sub&gt;perf&lt;/sub&gt;</td>
<td>0.953 (1.72s)</td>
<td>0.872 (20.39s)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>H-Mean</td>
<td>OCMO</td>
<td>0.964 (0.02s)</td>
<td>0.891 (4.85s)</td>
<td>0.764 (2.5s)</td>
<td>0.733 (5.16s)</td>
</tr>
<tr>
<td></td>
<td>sPADE</td>
<td>0.580 (15.74s)</td>
<td>0.578 (135.26s)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SVM&lt;sub&gt;perf&lt;/sub&gt;</td>
<td>0.953 (1.72s)</td>
<td>0.872 (20.39s)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Q-Mean</td>
<td>OCMO</td>
<td>0.964 (0.01s)</td>
<td>0.889 (4.87s)</td>
<td>0.551 (2.11s)</td>
<td>0.506 (4.27s)</td>
</tr>
<tr>
<td></td>
<td>sPADE</td>
<td>0.688 (15.83s)</td>
<td>0.632 (136.46s)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SVM&lt;sub&gt;perf&lt;/sub&gt;</td>
<td>0.950 (1.72s)</td>
<td>0.872 (20.39s)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

'-' means the corresponding algorithm does not terminate within 100x that of OCMO.
Performance vs run time for various online algorithms

(a) F1 measure on rcv1    (b) H-Mean on rcv1    (c) Q-Mean on rcv1
Optimal Multilabel classification with Non-decomposable Averaged Metrics
Multilabel Classification

- Multiclass: only one class associated with each example
- Multilabel: multiple classes associated with each example
### Applications

<table>
<thead>
<tr>
<th>Data type</th>
<th>Application</th>
<th>Resource</th>
<th>Labels Description (Examples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>categorization</td>
<td>news article</td>
<td>Reuters topics (agriculture, fishing)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>web page</td>
<td>Yahoo! directory (health, science)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>patent</td>
<td>WIPO (paper-making, fibreboard)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>email</td>
<td>R&amp;D activities (delegation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>legal document</td>
<td>Eurovoc (software, copyright)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>medical report</td>
<td>MeSH (disorders, therapies)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>radiology report</td>
<td>ICD-9-CM (diseases, injuries)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>research article</td>
<td>Heart conditions (myocarditis)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reference</td>
<td>ACM classification (algorithms)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>adjectives</td>
<td>Bibsonomy tags (sports, science)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bibsonomy tags (ai, kdd)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>semantics (object-related)</td>
</tr>
<tr>
<td>image</td>
<td>semantic annotation</td>
<td>pictures</td>
<td>concepts (trees, sunset)</td>
</tr>
<tr>
<td>video</td>
<td>semantic annotation</td>
<td>news clip</td>
<td>concepts (crowd, desert)</td>
</tr>
<tr>
<td>audio</td>
<td>noise detection</td>
<td>sound clip</td>
<td>type (speech, noise)</td>
</tr>
<tr>
<td></td>
<td>emotion detection</td>
<td>music clip</td>
<td>emotions (relaxing-calm)</td>
</tr>
<tr>
<td>structured</td>
<td>functional genomics</td>
<td>gene</td>
<td>functions (energy, metabolism)</td>
</tr>
<tr>
<td></td>
<td>proteomics</td>
<td>protein</td>
<td>enzyme classes (ligases)</td>
</tr>
<tr>
<td></td>
<td>directed marketing</td>
<td>person</td>
<td>product categories</td>
</tr>
</tbody>
</table>
The Multilabel Classification Problem

- **Inputs:** $X \in \mathcal{X}$, **Labels:** $Y \in \mathcal{Y} = [0, 1]^M$ (with $M$ labels)
- **Classifier** $\theta : \mathcal{X} \mapsto \mathcal{Y}$

**Example: Hamming Loss**

$$U(\theta) = \mathbb{E}_{X,Y \sim P} \left[ \sum_{m=1}^{M} 1[Y_m = \theta_m(X)] \right] = \sum_{m=1}^{M} \mathbb{P}(Y_m = \theta_m(X))$$

**Optimal Prediction for Hamming Loss**

$$\theta^*_m(x) = \text{sign} \left( \mathbb{P}(Y_m = 1|x) - \frac{1}{2} \right)$$

Well known convex surrogates e.g. hinge loss (Bartlett et al., 2006)
# Multilabel Confusion

Recall the binary confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>$Y = 1$</th>
<th>$Y = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta = 1$</td>
<td>$\hat{TP}$ \hspace{1cm} $P(Y = 1, \theta = 1)$ \hspace{1cm} $FP$ \hspace{1cm} $P(Y = 0, \theta = 1)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\hat{FP}$ \hspace{1cm} $P(Y = 0, \theta = 1)$ \hspace{1cm} $\hat{FN}$ \hspace{1cm} $P(Y = 1, \theta = 0)$</td>
<td></td>
</tr>
<tr>
<td>$\theta = 0$</td>
<td>$\hat{FN}$ \hspace{1cm} $P(Y = 1, \theta = 0)$ \hspace{1cm} $\hat{TN}$ \hspace{1cm} $P(Y = 0, \theta = 0)$</td>
<td></td>
</tr>
</tbody>
</table>

$^1$We focus on linear-fractional metrics e.g. Accuracy, $F_\beta$, Precision, Recall, Jaccard
Multilabel Confusion

Recall the binary confusion matrix

\[ \hat{C}_{m,n} = \begin{bmatrix} \hat{TP}_{m,n} = 1 \left[ \theta_m(x^{(n)})=1, y_m^{(n)}=1 \right] , & \hat{FP}_{m,n} = 1 \left[ \theta_m(x^{(n)})=1, y_m^{(n)}=0 \right] \\ \hat{FN}_{m,n} = 1 \left[ \theta_m(x^{(n)})=0, y_m^{(n)}=1 \right] , & \hat{TN}_{m,n} = 1 \left[ \theta_m(x^{(n)})=0, y_m^{(n)}=0 \right] \end{bmatrix} \]

Similar idea for multilabel classification, now across both labels \( m \) and examples \( n \).

\[ \begin{array}{c|c|c} & Y = 1 & Y = 0 \\ \hline \theta = 1 & TP & FP \\ P(Y = 1, \theta = 1) & P(Y = 0, \theta = 1) \\ \theta = 0 & FN & TN \\ P(Y = 1, \theta = 0) & P(Y = 0, \theta = 0) \end{array} \]

\(^1\)We focus on linear-fractional metrics e.g. Accuracy, \( F_\beta \), Precision, Recall, Jaccard
Label Averaging

Most popular multilabel metrics are averaged metrics

**Some notation**: Let $\eta_m(x) = \mathbb{P}(Y_m = 1|x)$

**Macro-Averaging**

Average over examples for each label
Lab el A veraging

Most popular multilabel metrics are averaged metrics

**Some notation:** Let $\eta_m(x) = \mathbb{P}(Y_m = 1|x)$

**Macro-Averaging**

Average over examples for each label

$$\hat{C}_m = \frac{1}{N} \sum_{n=1}^{N} \hat{C}_{m,n},$$
Label Averaging

Most popular multilabel metrics are averaged metrics

**Some notation:** Let $\eta_m(x) = \mathbb{P}(Y_m = 1|x)$

**Macro-Averaging**

Average over examples for each label

$$\hat{C}_m = \frac{1}{N} \sum_{n=1}^{N} \hat{C}_{m,n}, \quad \Psi_{\text{macro}} := \frac{1}{M} \sum_{m=1}^{M} \Psi(\hat{C}_m).$$
Label Averaging

Most popular multilabel metrics are averaged metrics

Some notation: Let $\eta_m(x) = \mathbb{P}(Y_m = 1|x)$

Macro-Averaging

Average over examples for each label

$$\hat{C}_m = \frac{1}{N} \sum_{n=1}^{N} \hat{C}_{m,n}, \quad \Psi_{\text{macro}} := \frac{1}{M} \sum_{m=1}^{M} \Psi(\hat{C}_m).$$

Bayes optimal classifier:

$$\theta_m^*(x) = \text{sign}(\eta_m(x) - \delta_m^*) \quad \forall m \in [M]$$
Instance Average

Average over labels for each example

\[ \hat{C}_n = \frac{1}{M} \sum_{m=1}^{M} \hat{C}_{m,n}, \]
Instance Average

Average over labels for each example

\[ \hat{C}_n = \frac{1}{M} \sum_{m=1}^{M} \hat{C}_{m,n}, \quad \Psi_{\text{instance}} := \frac{1}{N} \sum_{n=1}^{N} \Psi(\hat{C}_n). \]
Instance Average

Average over labels for each example

\[ \hat{C}_n = \frac{1}{M} \sum_{m=1}^{M} \hat{C}_{m,n}, \quad \Psi_{\text{instance}} := \frac{1}{N} \sum_{n=1}^{N} \Psi(\hat{C}_n). \]

Bayes optimal classifier:

\[ \theta^*_m(x) = \text{sign}(\eta_m(x) - \delta^*) \quad \forall m \in [M] \]

- Only require marginals \( \eta_m(x) \) i.e. label correlations have weak affect on optimal classification
- **Note:** Marginals may still be deterministically coupled across labels e.g. low rank, shared DNN representation
- Shared threshold across labels
Micro Average

Average over both examples and labels
Micro Average

Average over both examples and labels

\[ \hat{C} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \hat{C}_{m,n}, \]

Bayes optimal classifier:

\[ \theta^*_m(x) = \text{sign}(\eta_m(x) - \delta^*), \quad \forall m \in [M] \]

Bayes optimal is identical to instance averaging

Only require marginals \( \eta_m(x) \) i.e. label correlations have weak effect on optimal classification

Shared threshold across labels
Micro Average

Average over both examples and labels

$$\hat{C} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \hat{C}_{m,n}, \quad \Psi_{\text{instance}} := \Psi(\hat{C}).$$
Micro Average

Average over both examples and labels

\[ \hat{C} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \hat{C}_{m,n}, \quad \Psi_{\text{instance}} := \Psi(\hat{C}). \]

Bayes optimal classifier:

\[ \theta^*_m(x) = \text{sign}(\eta_m(x) - \delta^*) \quad \forall m \in [M] \]
Micro Average

Average over both examples and labels

\[ \hat{C} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \hat{C}_{m,n}, \quad \Psi_{\text{instance}} := \Psi(\hat{C}). \]

Bayes optimal classifier:

\[ \theta^*_m(x) = \text{sign}(\eta_m(x) - \delta^*) \quad \forall m \in [M] \]

- Bayes optimal is identical to instance averaging
- Only require marginals \( \eta_m(x) \) i.e. label correlations have weak affect on optimal classification
- Shared threshold across labels
Simulated Micro-averaged F1

\[ \delta^* = 0.40 \]

\[ \eta_0(x) \]

\[ \theta_0^* \]

\[ \eta_1(x) \]

\[ \theta_1^* \]
Empirical Evaluation
<table>
<thead>
<tr>
<th>Dataset</th>
<th>BR $F_1$</th>
<th>Plugin $F_1$</th>
<th>Macro-Thres</th>
<th>BR Jaccard</th>
<th>Plugin Jaccard</th>
<th>Macro-Thres</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>0.6559</td>
<td><strong>0.6847</strong></td>
<td>0.6631</td>
<td>0.4878</td>
<td><strong>0.5151</strong></td>
<td>0.5010</td>
</tr>
<tr>
<td>Birds</td>
<td><strong>0.4040</strong></td>
<td><strong>0.4088</strong></td>
<td>0.2871</td>
<td>0.2495</td>
<td><strong>0.2648</strong></td>
<td>0.1942</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.5815</td>
<td><strong>0.6554</strong></td>
<td>0.6419</td>
<td>0.3982</td>
<td><strong>0.4908</strong></td>
<td>0.4790</td>
</tr>
<tr>
<td>Cal500</td>
<td>0.3647</td>
<td><strong>0.4891</strong></td>
<td>0.4160</td>
<td>0.2229</td>
<td><strong>0.3225</strong></td>
<td>0.2608</td>
</tr>
</tbody>
</table>

**Table:** Comparison of plugin-estimator methods on multilabel $F_1$ and Jaccard metrics. Reported values correspond to *micro-averaged* metric ($F_1$ and Jaccard) computed on test data (with standard deviation, over 10 random validation sets for tuning thresholds). Plugin is consistent for micro-averaged metrics, and performs the best consistently across datasets.
Table: Comparison of plugin-estimator methods on multilabel $F_1$ and Jaccard metrics. Reported values correspond to instance-averaged metric ($F_1$ and Jaccard) computed on test data (with standard deviation, over 10 random validation sets for tuning thresholds). Plugin is consistent for instance-averaged metrics, and performs the best consistently across datasets.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>BR</th>
<th>Plugin $F_1$</th>
<th>Macro-Thres</th>
<th>BR</th>
<th>Plugin Jaccard</th>
<th>Macro-Thres</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>0.6601</td>
<td><strong>0.6941</strong></td>
<td>0.6737</td>
<td>0.5046</td>
<td><strong>0.5373</strong></td>
<td>0.5260</td>
</tr>
<tr>
<td>Birds</td>
<td>0.3366</td>
<td><strong>0.3448</strong></td>
<td>0.2971</td>
<td>0.2178</td>
<td><strong>0.2341</strong></td>
<td>0.2051</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.5440</td>
<td><strong>0.6450</strong></td>
<td><strong>0.6440</strong></td>
<td>0.3982</td>
<td><strong>0.4912</strong></td>
<td><strong>0.4900</strong></td>
</tr>
<tr>
<td>Cal500</td>
<td>0.1293</td>
<td>0.2687</td>
<td><strong>0.3226</strong></td>
<td>0.0880</td>
<td>0.1834</td>
<td><strong>0.2146</strong></td>
</tr>
</tbody>
</table>

**Table:** Comparison of plugin-estimator methods on multilabel $F_1$ and Jaccard metrics. Reported values correspond to the *macro-averaged* metric computed on test data (with standard deviation, over 10 random validation sets for tuning thresholds). Macro-Thres is consistent for macro-averaged metrics, and is competitive in three out of four datasets. Though not consistent for macro-averaged metrics, Plugin achieves the best performance in three out of four datasets.
Correlated Binary Decisions

- Same procedure applies to more general correlated binary decisions using averaged metrics

Example application:
- point estimates of brain networks from posterior distributions
Conclusion
Conclusion and open questions

- Optimal classifiers for a large family of metrics have a simple threshold form $\text{sign}(P(Y = 1|X) - \delta)$
- Proposed scalable algorithms for consistent estimation
Conclusion and open questions

- Optimal classifiers for a large family of metrics have a simple threshold form \( \text{sign}(P(Y = 1|X) - \delta) \)
- Proposed scalable algorithms for consistent estimation

Open Questions:

Can we elucidate utility functions from feedback?
Can we characterize the entire family of utility metrics with thresholded optimal decision functions?
What of more general structured prediction?
Conclusion and open questions

- Optimal classifiers for a large family of metrics have a simple threshold form $\text{sign}(P(Y = 1|X) - \delta)$
- Proposed scalable algorithms for consistent estimation

Open Questions:
- Can we elucidate utility functions from feedback?
Conclusion and open questions

- Optimal classifiers for a large family of metrics have a simple threshold form $\text{sign}(P(Y = 1|X) - \delta)$
- Proposed scalable algorithms for consistent estimation

Open Questions:
- Can we elucidate utility functions from feedback?
- Can we characterize the entire family of utility metrics with thresholded optimal decision functions?
Conclusion and open questions

- Optimal classifiers for a large family of metrics have a simple threshold form: \( \text{sign}(P(Y = 1|X) - \delta) \)
- Proposed scalable algorithms for consistent estimation

Open Questions:
- Can we elucidate utility functions from feedback?
- Can we characterize the entire family of utility metrics with thresholded optimal decision functions?
- What of more general structured prediction?
Questions?

sanmi@illinois.edu
References
References


Backup Slides
Two Step Normalized Gradient Descent for optimal threshold search

1: **Input:** Training sample \( \{X_i, Y_i\}_{i=1}^n \), utility measure \( U \), conditional probability estimator \( \hat{\eta} \), stepsize \( \alpha \).

2: Randomly split the training sample into two subsets \( \{X_i^{(1)}, Y_i^{(1)}\}_{i=1}^{n_1} \) and \( \{X_i^{(2)}, Y_i^{(2)}\}_{i=1}^{n_2} \);

3: Estimate \( \hat{\eta} \) on \( \{X_i^{(1)}, Y_i^{(1)}\}_{i=1}^{n_1} \).

4: Initialize \( \delta = 0.5 \);

5: **while** not converged **do**

6: Evaluate TP, TN on \( \{X_i^{(2)}, Y_i^{(2)}\}_{i=1}^{n_2} \) with \( f(x) = \text{sign}(\hat{\eta} - \delta) \).

7: Calculate \( \nabla U \);

8: \( \delta \leftarrow \delta - \alpha \frac{\nabla U}{\|\nabla U\|} \).

9: **end while**

10: **Output:** \( \hat{f}(x) = \text{sign}(\hat{\eta} - \delta) \).
Online Complex Metric Optimization (OCMO)

Require: online CPE with update $g$, metric $\mathcal{U}$, stepsize $\alpha$;

1: Initialize $\eta_0, \delta_0 = 0.5$;
2: \textbf{while} data stream has points \textbf{do}
3: \hspace{1em} Receive data point $(x_t, y_t)$
4: \hspace{1em} $\eta_t = g(\eta_{t-1})$;
5: \hspace{1em} $\delta_t^{(0)} = \delta_t$, $\text{TP}_t^{(0)} = \text{TP}_{t-1}$, $\text{TN}_t^{(0)} = \text{TN}_{t-1}$;
6: \hspace{1em} \textbf{for} $i = 1, \cdots, T_t$ \textbf{do}
7: \hspace{2em} \textbf{if} $\eta_t(x_t) > \delta_t^{(i-1)}$ \textbf{then}
8: \hspace{3em} $\text{TP}_t^{(i)} \leftarrow \frac{\text{TP}_{t-1} \cdot (t-1) + (1+y_t)/2}{t}$, $\text{TN}_t^{(i)} \leftarrow \frac{\text{TN}_{t-1} \cdot t-1}{t}$;
9: \hspace{2em} \textbf{else} $\text{TP}_t^{(i)} \leftarrow \frac{\text{TP}_{t-1} \cdot t-1}{t}$, $\text{TN}_t^{(i)} \leftarrow \frac{\text{TN}_{t-1} \cdot t + (1-y_t)/2}{t+1}$;
10: \hspace{1em} \textbf{end if}
11: \hspace{1em} $\delta_t^{(i)} = \delta_t^{(i-1)} - \alpha \frac{\nabla G(\text{TP}_t, \text{TN}_t)}{\|\nabla G(\text{TP}_t, \text{TN}_t)\|}$, $\text{TP}_t = \text{TP}_t^{(i)}$, $\text{TN}_t = \text{TN}_t^{(i)}$;
12: \hspace{1em} \textbf{end for}
13: \hspace{1em} $\delta_{t+1} = \delta_t^{(T_t)}$;
14: \hspace{1em} $t = t + 1$;
15: \hspace{1em} \textbf{end while}
16: Output $(\eta_t, \delta_t)$. 


Scaling up Classification with Complex Metrics
Additional properties of $\mathcal{U}$

Informal theorem (Yan et al., 2016)

Suppose $\mathcal{U}$ is fractional-linear or monotonic, under weak conditions\(^a\) on $P$:

- $\mathcal{U}(\theta_\delta, P)$ is differentiable wrt $\delta$
- $\mathcal{U}(\theta_\delta, P)$ is Lipschitz wrt $\delta$
- $\mathcal{U}(\theta_\delta, P)$ is strictly locally quasi-concave wrt $\delta$

\(^a\eta_x\) is differentiable wrt $x$, and its characteristic function is absolutely integrable
Algorithms

Normalized Gradient Descent (Hazan et al., 2015)

Fix $\epsilon > 0$, let $f$ be strictly locally quasi-concave, and $x^* \in \text{argmin } f(x)$. NGD algorithm with number of iterations $T \geq \kappa^2 \|x_1 - x^*\|^2 / \epsilon^2$ and step size $\eta = \epsilon / \kappa$ achieves $f(\bar{x}_T) - f(x^*) \leq \epsilon$.

Batch Algorithm

1. Estimate $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010)
2. Solve $\max_{\delta} \mathcal{U}(\hat{\theta}_\delta, \mathcal{D}_n)$ using normalized gradient ascent

Online Algorithm

Interleave $\hat{\eta}_t$ update and $\hat{\delta}_t$ update
Sample Complexity

With appropriately chosen step size,\[ R(\hat{\theta}, \hat{\delta}, P) \leq C \int |\hat{\eta} - \eta| d\mu \]

Comparison to threshold search complexity of NGD is \[ O(n) = O(n/\epsilon^2) \], where \( t \) is the number of iterations and \( \epsilon \) is the precision of the solution when \( \log n \geq 1/\epsilon^2 \), the batch algorithm has favorable computational complexity vs. threshold search.

Online Algorithm

Let \( \eta \) estimation error at step \( t \) given by \[ r_t = \int |\eta_t - \eta| d\mu, \]
with appropriately chosen step size,\[ R(\hat{\theta}, \delta_t, P) \leq C \sum_{t=1}^{\infty} r_t \]
Sample Complexity

Batch Algorithm

With appropriately chosen step size, \( \mathcal{R}(\hat{\theta}_\delta, \mathcal{P}) \leq C \int |\hat{\eta} - \eta| d\mu \)
Sample Complexity

Batch Algorithm

With appropriately chosen step size, $R(\hat{\theta}, \mathcal{P}) \leq C \int |\hat{\eta} - \eta| d\mu$

Comparison to threshold search

- complexity of NGD is $O(nt) = O(n/\epsilon^2)$, where $t$ is the number of iterations and $\epsilon$ is the precision of the solution
- when $\log n \geq 1/\epsilon^2$, the batch algorithm has favorable computational complexity vs. threshold search
Sample Complexity

Batch Algorithm

With appropriately chosen step size, \( R(\hat{\theta}_\delta, \mathcal{P}) \leq C \int |\hat{\eta} - \eta| d\mu \)

Comparison to threshold search

- complexity of NGD is \( O(nt) = O(n/\epsilon^2) \), where \( t \) is the number of iterations and \( \epsilon \) is the precision of the solution
- when \( \log n \geq 1/\epsilon^2 \), the batch algorithm has favorable computational complexity vs. threshold search

Online Algorithm

Let \( \eta \) estimation error at step \( t \) given by \( r_t = \int |\eta_t - \eta| d\mu \), with appropriately chosen step size, \( R(\hat{\theta}_\delta, \mathcal{P}) \leq \frac{C \sum_{i=1}^{t} r_i}{t} \)