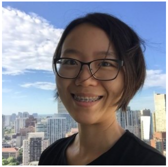
Metrics Matter, Examples from Binary and Multilabel Classification

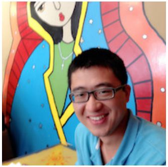
Sanmi Koyejo

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

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathsf{TP}}{2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}$$

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathsf{TP}}{2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}$$

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

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathsf{TP}}{2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}$$

- **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**

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

- **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**

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathsf{TP}}{2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}$$

- **Direct optimization**
- **Mixed combinatorial optimization**
- **Convex lower bound**
  - difficult to construct
  - most statistical properties unknown
- **Logloss + thresholding**

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathsf{TP}}{2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}$$

- **Direct optimization**
- **Mixed combinatorial optimization**
- **Convex lower bound**
- **Logloss + thresholding**
  - simple, most common approach in practice
  - has good statistical properties!

Learning with complex metrics

Goal: Train a DNN to optimize F-measure.

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

- **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 $\theta : \mathcal{X} \mapsto \{0, 1\}$

	$Y = 1$	$Y = 0$
$\theta = 1$	TP $P(Y = 1, \theta = 1)$	FP $P(Y = 0, \theta = 1)$
$\theta = 0$	FN $P(Y = 1, \theta = 0)$	TN $P(Y = 0, \theta = 0)$

Metrics tradeoff which kinds of mistakes are (most) acceptable

Metrics tradeoff which kinds of mistakes are (most) acceptable

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* $\mathbf{\Phi} : [0,1]^4 \mapsto \mathbb{R}$

Performance metrics

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

Examples

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

Performance metrics

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

Examples

- Accuracy (fraction of mistakes) = TP + TN
- Error Rate = 1-Accuracy = FP + FN
- For medical diagnosis example, consider the weighted error = w_1FP + w_2FN, where $w_2 \gg w_1$

Performance metrics

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

Examples

- Accuracy (fraction of mistakes) $=$ TP $+$ TN
- Error Rate $=$ 1-Accuracy $=$ FP $+$ FN
- For medical diagnosis example, consider the weighted error $=$ w_1FP $+$ w_2FN, where $w_2 \gg w_1$

and many more ...

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

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Jaccard} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}.$$

No need for metrics if you can learn a "perfect" classifier!

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

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

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

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

Utility & Regret

- population performance is measured via utility

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

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

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

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

Utility & Regret

- population performance is measured via utility

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

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

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

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

The regret of the classifier θ 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 $\theta_n \Big| \mathcal{D}_n$

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 $\theta_n \big| \mathcal{D}_n$

Example

Empirical risk minimization via SVM:

$$\theta_n = \mathsf{sign}\left(\underset{f \in \mathcal{H}_k}{\operatorname{argmin}} \sum_{\{x_i, y_i\} \in \mathcal{D}_n} \max\left(0, 1 - y_i f(x_i)\right) \right)$$

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

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

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:

$$\mathsf{ACC}(\theta, \mathcal{D}_n) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}_n} 1_{[y_i = \theta(x_i)]}$$

Consider the empirical accuracy:

$$\mathsf{ACC}(\theta, \mathcal{D}_n) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}_n} 1_{[y_i = \theta(x_i)]}$$

- Observe that the classification problem

$$\min_{\theta \in \mathcal{F}} \mathsf{ACC}(\theta, \mathcal{D}_n)$$

  is a combinatorial optimization problem
- optimal classification is computationally hard for non-trivial $\mathcal{F}$ and $\mathcal{D}_n$

Bayes Optimal Classifier

Population Accuracy

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

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

Bayes Optimal Classifier

Population Accuracy

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

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

Weighted Accuracy

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

- Scott (2012) showed that $\theta^*(\mathbf{x}) = \mathrm{sign}\left(P(Y=1|\mathbf{x}) - \rho\right)$

Where do surrogates come from?

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

$$\theta_n = \mathsf{sign}\left(\operatorname*{argmin}_f L(f, \mathcal{D}_n)\right) \xrightarrow{n\to\infty} \theta^*(x)$$

Where do surrogates come from?

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

$$\theta_n = \mathsf{sign}\left(\operatorname*{argmin}_f L(f, \mathcal{D}_n)\right) \xrightarrow{n \to \infty} \theta^*(x)$$

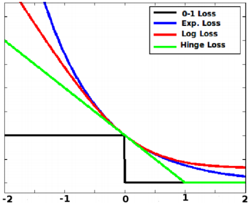
- These are known as *classification calibrated* surrogate losses (Bartlett et al., 2003; Scott, 2012)

Where do surrogates come from?

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

$$\theta_n = \text{sign}\left(\underset{f}{\operatorname{argmin}}\, L(f, \mathcal{D}_n)\right) \xrightarrow{n\to\infty} \theta^*(x)$$

- These are known as *classification calibrated* surrogate losses (Bartlett et al., 2003; Scott, 2012)
- research can focus on how to choose $L, \mathcal{F}$ 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

$$\mathrm{E}\left[\Phi(\widehat{\mathbf{C}})\right] = \left\langle \mathbf{A}, \mathrm{E}\left[\widehat{\mathbf{C}}\right]\right\rangle = \Phi(\mathrm{E}\left[\widehat{\mathbf{C}}\right])$$

Non-decomposability

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

$$\mathrm{E}\left[\Phi(\widehat{\mathbf{C}})\right] = \left\langle \mathbf{A}, \mathrm{E}\left[\widehat{\mathbf{C}}\right]\right\rangle = \Phi(\mathrm{E}\left[\widehat{\mathbf{C}}\right])$$

- However, F_β, Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. $\mathbf{C}$
- Thus imples that the *averaging trick* is no longer valid

$$\mathrm{E}\left[\Phi(\widehat{\mathbf{C}})\right] \neq \Phi(\mathrm{E}\left[\widehat{\mathbf{C}}\right])$$

Non-decomposability

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

$$\mathrm{E}\left[\Phi(\widehat{\mathbf{C}})\right] = \left\langle \mathbf{A}, \mathrm{E}\left[\widehat{\mathbf{C}}\right]\right\rangle = \Phi(\mathrm{E}\left[\widehat{\mathbf{C}}\right])$$

- However, F_β, Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. $\mathbf{C}$
- Thus imples that the *averaging trick* is no longer valid

$$\mathrm{E}\left[\Phi(\widehat{\mathbf{C}})\right] \neq \Phi(\mathrm{E}\left[\widehat{\mathbf{C}}\right])$$

- 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 $\mathcal{U}$ be differentiable wrt. the confusion matrix, then $\exists$ a δ^* such that:

$$\theta^*(x) = \mathrm{sign}\left(\eta_x - \delta^*\right)$$

is a Bayes optimal classifier almost everywhere.

[1] **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 $\mathcal{U}$ be differentiable wrt. the confusion matrix, then $\exists$ a δ^* such that:

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

is a Bayes optimal classifier almost everywhere.

- result does not require concavity of $\mathcal{U}$, or other "nice" properties

1

---

[1] **Condition:** $P(\eta_x = \delta^*) = 0$, easily satisfied e.g. when $P(X)$ 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:

$$\theta^*_{\mathcal{F}} = \operatorname*{argmax}_{\theta \in \mathcal{F}} \mathcal{U}(\theta, \mathcal{P})$$

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:

$$\theta^*_{\mathcal{F}} = \underset{\theta \in \mathcal{F}}{\operatorname{argmax}}\, \mathcal{U}(\theta, \mathcal{P})$$

- Show that the optimal "relaxed" classifier is $\theta^*_{\mathcal{F}} = \mathsf{sign}(\eta_x - \delta^*)$

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:

$$\theta^*_{\mathcal{F}} = \operatorname*{argmax}_{\theta \in \mathcal{F}} \mathcal{U}(\theta, \mathcal{P})$$

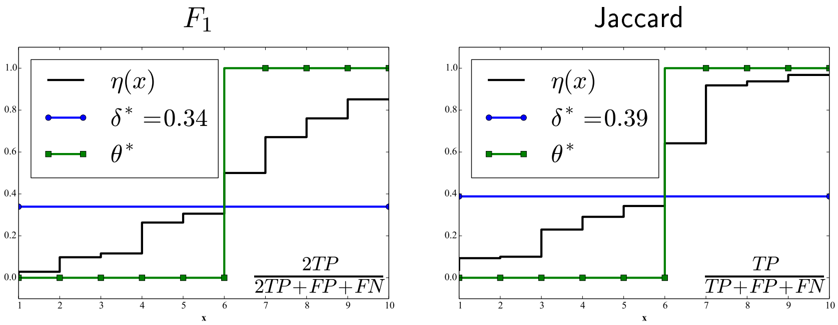
- Show that the optimal "relaxed" classifier is $\theta^*_{\mathcal{F}} = \mathsf{sign}(\eta_x - \delta^*)$
- Observe that $\Theta \subset \mathcal{F}$. Thus $\mathcal{U}(\theta^*_{\mathcal{F}}, \mathcal{P}) \geq \mathcal{U}(\theta^*_{\Theta}, \mathcal{P})$.
- As a result, $\theta^*_{\mathcal{F}} \in \Theta$ implies that $\theta^*_{\mathcal{F}} \equiv \theta^*_{\Theta}$.

Some recovered and new results

Metric	Form	Optimal threshold
F_β	$\frac{(1+\beta^2)TP}{(1+\beta^2)TP+\beta^2 FN+FP}$	$\delta^* = \frac{\mathcal{L}^*}{1+\beta^2}$
Cost-sensitive learning	$c_0 + c_1 TP + c_2\gamma(\theta)$	$\delta^* = -\frac{c_2}{c_1}$
Precision	$\frac{TP}{TP+FP}$	$\delta^* = \mathcal{L}^*$
Recall	$\frac{TP}{TP+FN}$	$\delta^* = 0$
Weighted Accuracy	$\frac{2(TP+TN)}{2(TP+TN)+FP+FN}$	$\delta^* = \frac{1}{2}$
Jaccard Coefficient	$\frac{TP}{TP+FP+FN}$	$\delta^* = \frac{\mathcal{L}^*}{1+\mathcal{L}^*}$

F_β (Ye et al., 2012), Monotonic metrics (Narasimhan et al., 2014)

Simulated examples



- Finite sample space $\mathcal{X}$, so we can exhaustively search for θ^*

Algorithm 1 (Koyejo et al., 2014)

Step 1: Conditional probability estimation

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

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

Algorithm 1 (Koyejo et al., 2014)

Step 1: Conditional probability estimation

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

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

Step 2: Threshold search

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

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

$\hat{\theta}_{\hat{\delta}}$ is consistent

Algorithm 2 (Koyejo et al., 2014)

Step 1: Weighted classifier estimation)

For classification-calibrated loss (Scott, 2012)

$$\hat{f}_\delta = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{x_i, y_i \in \mathcal{D}_n} \ell_\delta(f(x_i), y_i)$$

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

Step 2: Threshold search

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

$\hat{\theta}_{\hat{\delta}}$ is consistent

Algorithm 3 (Yan et al., 2016)

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

Algorithm 3 (Yan et al., 2016)

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

Online Algorithm

Iteratively update

1. $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010)
2. $\hat{\delta}_t$ using *normalized gradient ascent*

Online algorithm sample complexity

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

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

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

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

Batch algorithm

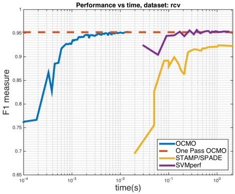
Data set/Metric	LR+Plug-in	LR+Batch	XGB+Plug-in	XGB+Batch
news20-Q-Mean	0.948 (3.77s)	0.948 (0.001s)	0.874 (3.87s)	0.875 (0.003s)
news20-H-Mean	0.950 (3.70s)	0.950 (0.003s)	0.859 (3.61s)	0.860 (0.003s)
news20-F1	0.949 (3.49s)	0.948 (0.01s)	0.872 (5.07s)	0.874 (0.01s)
default-Q-Mean	0.664 (14.3s)	0.667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

Online Complex Metric Optimization (OCMO)

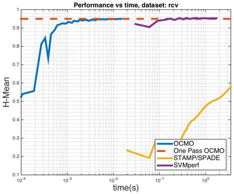
Metric	Algorithm	RCV1	Epsilon	KDD-A	KDD-B
F1	OCMO	0.952 (0.01s)	0.804 (4.87s)	0.934 (2.43s)	0.941 (5.01s)
	sTAMP	0.923 (14.44s)	0.585 (133.23s)	-	-
	SVM^{perf}	0.953 (1.72s)	0.872 (20.39s)	-	-
H-Mean	OCMO	0.964 (0.02s)	0.891 (4.85s)	0.764 (2.5s)	0.733 (5.16s)
	sPADE	0.580 (15.74s)	0.578 (135.26s)	-	-
	SVM^{perf}	0.953 (1.72s)	0.872 (20.39s)	-	-
Q-Mean	OCMO	0.964 (0.01s)	0.889 (4.87s)	0.551 (2.11s)	0.506 (4.27s)
	sPADE	0.688 (15.83s)	0.632 (136.46s)	-	-
	SVM^{perf}	0.950 (1.72s)	0.872 (20.39s)	-	-

'–' means the corresponding algorithm does not terminate within 100x that of OCMO.

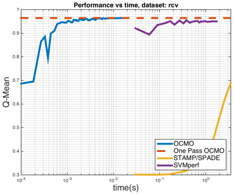
Performance vs run time for various online algorithms



(a) F1 measure on rcv1



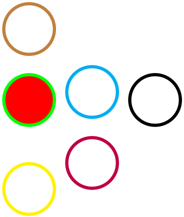
(b) H-Mean on rcv1

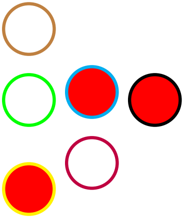


(c) Q-Mean on rcv1

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

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

The Multilabel Classification Problem

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

Example: Hamming Loss

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

Optimal Prediction for Hamming Loss

$$\theta_m^*(\mathbf{x}) = \mathsf{sign}\left(\mathbb{P}(Y_m = 1|\mathbf{x}) - \frac{1}{2}\right)$$

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

Multilabel Confusion

Recall the binary confusion matrix

	$Y = 1$	$Y = 0$
$\theta = 1$	TP $P(Y = 1, \theta = 1)$	FP $P(Y = 0, \theta = 1)$
$\theta = 0$	FN $P(Y = 1, \theta = 0)$	TN $P(Y = 0, \theta = 0)$

[1] We focus on linear-fractional metrics e.g. Accuracy, F_β, Precision, Recall, Jaccard

Multilabel Confusion

Recall the binary confusion matrix

	$Y=1$	$Y=0$
$\theta=1$	TP P($Y=1$, $\theta=1$)	FP P($Y=0$,$\theta=1$)
$\theta=0$	FN P($Y=1$, $\theta=0$)	TN P($Y=0$,$\theta=0$)

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

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

[1] We focus on linear-fractional metrics e.g. Accuracy, F_β, 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

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

$$\widehat{\mathbf{C}}_m = \frac{1}{N}\sum_{n=1}^{N} \widehat{\mathbf{C}}_{m,n},$$

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

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

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

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

Bayes optimal classifier:

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

Instance Average

Average over labels for each example

$$\widehat{\mathbf{C}}_n = \frac{1}{M} \sum_{m=1}^{M} \widehat{\mathbf{C}}_{m,n},$$

Instance Average

Average over labels for each example

$$\widehat{\mathbf{C}}_n = \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{C}}_{m,n}, \quad \Psi_{\mathsf{instance}} := \frac{1}{N}\sum_{n=1}^{N}\Psi(\widehat{\mathbf{C}}_n).$$

Instance Average

Average over labels for each example

$$\widehat{\mathbf{C}}_n = \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{C}}_{m,n}, \quad \Psi_{\textsf{instance}} := \frac{1}{N}\sum_{n=1}^{N}\Psi(\widehat{\mathbf{C}}_n).$$

Bayes optimal classifier:

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

- 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

$$\widehat{\mathbf{C}} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \widehat{\mathbf{C}}_{m,n},$$

Micro Average

Average over both examples and labels

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

Micro Average

Average over both examples and labels

$$\widehat{\mathbf{C}} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \widehat{\mathbf{C}}_{m,n}, \quad \Psi_{\mathsf{instance}} := \Psi(\widehat{\mathbf{C}}).$$

Bayes optimal classifier:

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

Micro Average

Average over both examples and labels

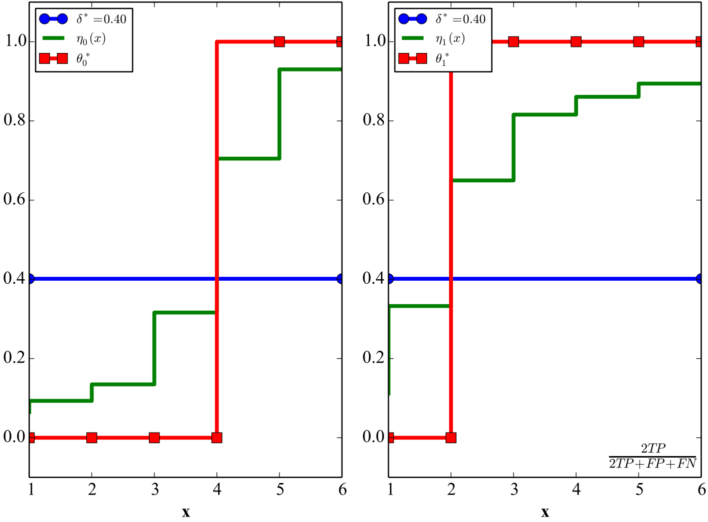
$$\widehat{\mathbf{C}} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \widehat{\mathbf{C}}_{m,n}, \quad \Psi_{\text{instance}} := \Psi(\widehat{\mathbf{C}}).$$

Bayes optimal classifier:

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

- 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



Empirical Evaluation

Dataset	BR	Plugin F_1	Macro-Thres	BR	Plugin Jaccard	Macro-Thres
Scene	0.6559	**0.6847**	0.6631	0.4878	**0.5151**	0.5010
Birds	**0.4040**	**0.4088**	0.2871	0.2495	**0.2648**	0.1942
Emotions	0.5815	**0.6554**	0.6419	0.3982	**0.4908**	0.4790
Cal500	0.3647	**0.4891**	0.4160	0.2229	**0.3225**	0.2608

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.

Dataset	BR	Plugin F_1	Macro-Thres	BR	Plugin Jaccard	Macro-Thres
Scene	0.5695	**0.6422**	0.6303	0.5466	**0.5976**	0.5902
Birds	0.1209	**0.1390**	**0.1390**	0.1058	**0.1239**	0.1195
Emotions	0.4787	**0.6241**	0.6156	0.4078	**0.5340**	0.5173
Cal500	0.3632	**0.4855**	0.4135	0.2268	**0.3252**	0.2623

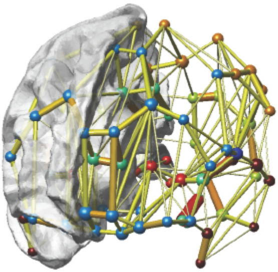
Table: Comparison of plugin-estimator methods on multilabel F_1 and Jaccard metrics. Reported values correspond to *instance-averaged* metric (F_1 and Jaccard) computed on test data (with standard deviation, over 10 random validation sets for tuning thresholds). Plugin is consistent for instance-averaged metrics, and performs the best consistently across datasets.

Dataset	BR	Plugin F_1	Macro-Thres	BR	Plugin Jaccard	Macro-Thres
Scene	0.6601	**0.6941**	0.6737	0.5046	**0.5373**	0.5260
Birds	0.3366	**0.3448**	0.2971	0.2178	**0.2341**	0.2051
Emotions	0.5440	**0.6450**	**0.6440**	0.3982	**0.4912**	**0.4900**
Cal500	0.1293	0.2687	**0.3226**	0.0880	0.1834	**0.2146**

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 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:

Conclusion and open questions

- Optimal classifiers for a large family of metrics have a simple threshold form 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 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 $\mathrm{sign}(P(Y = 1|X) - \delta)$
- 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 I

Francis R Bach. Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *Journal of Machine Learning Research*, 15(1):595–627, 2014.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Large margin classifiers: Convex loss, low noise, and convergence rates. In *NIPS*, pages 1173–1180, 2003.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems*, pages 1585–1593, 2015.

Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent binary classification with generalized performance metrics. In *Advances in Neural Information Processing Systems*, pages 2744–2752, 2014.

Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Advances in Neural Information Processing Systems*, pages 1493–1501, 2014.

Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *32nd International Conference on Machine Learning (ICML)*, 2015.

Mark D Reid and Robert C Williamson. Composite binary losses. *The Journal of Machine Learning Research*, 9999:2387–2422, 2010.

Clayton Scott. Calibrated asymmetric surrogate losses. *Electronic J. of Stat.*, 6:958–992, 2012.

Yaroslav D Sergeyev. Global one-dimensional optimization using smooth auxiliary functions. *Mathematical Programming*, 81(1):127–146, 1998.

Bowei Yan, Kai Zhong, Oluwasanmi Koyejo, and Pradeep Ravikumar. Online classification with complex metrics. In *arXiv:1610.07116v1*, 2016.

Nan Ye, Kian Ming A Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measures: a tale of two approaches. In *Proceedings of the International Conference on Machine Learning*, 2012.

Backup Slides

Two Step Normalized Gradient Descent for optimal threshold search

Input: Training sample $\{X_i, Y_i\}_{i=1}^{n}$, utility measure $\mathcal{U}$, conditional probability estimator $\hat{\eta}$, stepsize α.
Randomly split the training sample into two subsets $\{X_i^{(1)}, Y_i^{(1)}\}_{i=1}^{n_1}$ and $\{X_i^{(2)}, Y_i^{(2)}\}_{i=1}^{n_2}$;
Estimate $\hat{\eta}$ on $\{X_i^{(1)}, Y_i^{(1)}\}_{i=1}^{n_1}$.
Initialize $\delta = 0.5$;
while not converged **do**
    Evaluate TP, TN on $\{X_i^{(2)}, Y_i^{(2)}\}_{i=1}^{n_2}$ with $f(x) = \text{sign}(\hat{\eta} - \delta)$.
    Calculate $\nabla\mathcal{U}$;
    $\delta \leftarrow \delta - \alpha \frac{\nabla\mathcal{U}}{\|\nabla\mathcal{U}\|}$.
end while
Output: $\hat{f}(x) = \text{sign}(\hat{\eta} - \delta)$.

Online Complex Metric Optimization (OCMO)

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

Initialize η_0, $\delta_0 = 0.5$;
while data stream has points **do**
  Receive data point (x_t, y_t)
  $\eta_t = g(\eta_{t-1})$;
  $\delta_t^{(0)} = \delta_t$, $\mathsf{TP}_t^{(0)} = \mathsf{TP}_{t-1}, \mathsf{TN}_t^{(0)} = \mathsf{TN}_{t-1}$;
  for $i = 1, \cdots, T_t$ **do**
    if $\eta_t(x_t) > \delta_t^{(i-1)}$ **then**
      $\mathsf{TP}_t^{(i)} \leftarrow \frac{\mathsf{TP}_{t-1} \cdot (t-1) + (1+y_t)/2}{t}$, $\mathsf{TN}_t^{(i)} \leftarrow \mathsf{TN}_{t-1} \cdot \frac{t-1}{t}$;
    else $\mathsf{TP}_t^{(i)} \leftarrow \mathsf{TP}_{t-1} \cdot \frac{t-1}{t}$, $\mathsf{TN}_t^{(i)} \leftarrow \frac{\mathsf{TN}_{t-1} \cdot t + (1-y_t)/2}{t+1}$;
    end if
    $\delta_t^{(i)} = \delta_t^{(i-1)} - \alpha \frac{\nabla \mathcal{G}(\mathsf{TP}_t, \mathsf{TN}_t)}{\|\nabla \mathcal{G}(\mathsf{TP}_t, \mathsf{TN}_t)\|}$, $\mathsf{TP}_t = \mathsf{TP}_t^{(i)}, \mathsf{TN}_t = \mathsf{TN}_t^{(i)}$;
  end for
  $\delta_{t+1} = \delta_t^{(T_t)}$;
  $t = t + 1$;
end while
Output (η_t, δ_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 δ
- $\mathcal{U}(\theta_\delta, P)$ is Lipschitz wrt δ
- $\mathcal{U}(\theta_\delta, P)$ is strictly locally quasi-concave wrt δ

---

[a] η_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 $x^* \in \mathrm{argmin}\, f(x)$. 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

Sample Complexity

Batch Algorithm

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

Sample Complexity

Batch Algorithm

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

Comparison to threshold search

- complexity of NGD is $O(nt) = O(n/\epsilon^2)$, where t is the number of iterations and ϵ 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, $\mathcal{R}(\hat{\theta}_{\hat{\delta}}, \mathcal{P}) \leq C \int |\hat{\eta} - \eta| d\mu$

Comparison to threshold search

- complexity of NGD is $O(nt) = O(n/\epsilon^2)$, where t is the number of iterations and ϵ 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 η estimation error at step t given by $r_t = \int |\eta_t - \eta| d\mu$, with appropriately chosen step size, $\mathcal{R}(\hat{\theta}_{\delta_t}, \mathcal{P}) \leq \frac{C \sum_{i=1}^{t} r_i}{t}$