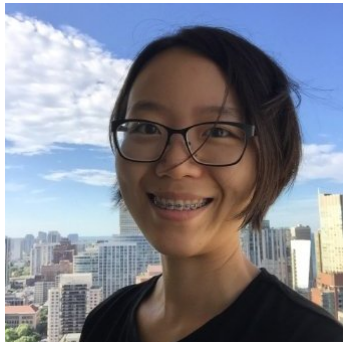
# Beyond Accuracy: Scalable Classification with Complex Metrics

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

# Binary classification

- Perhaps *the* classic problem in machine learning
- Often a subroutine in more complex problems e.g. multiclass / multilabel classification

Formally:

- Let $Y \in \{0, 1\}$ denote labels, $X \in \mathcal{X}$ denote instances
- Find classifier $\theta : \mathcal{X} \mapsto \{0, 1\}$, using training samples $\mathcal{D}_n = \{X_i, Y_i\}_{i=1}^n$
- Classifier is selected from the function class $\mathcal{F}$ e.g. linear functions, neural networks ...

# Binary classification

- Perhaps *the* classic problem in machine learning
- Often a subroutine in more complex problems e.g. multiclass / multilabel classification

## Formally:

- Let $Y \in \{0, 1\}$ denote labels, $X \in \mathcal{X}$ denote instances
- Find classifier $\theta : \mathcal{X} \mapsto \{0, 1\}$, using training samples $\mathcal{D}_n = \{X_i, Y_i\}_{i=1}^n$
- Classifier is selected from the function class $\mathcal{F}$ e.g. linear functions, neural networks ...

kaggle Host Competitions Datasets Kernels Jobs Community Sign up Login

Completed • Swag

# Dogs vs. Cats

Wed 25 Sep 2013 – Sat 1 Feb 2014 (2 years ago)

Dashboard

Home
Data

Information
Description
Evaluation
Rules
Prizes
Winners

Forum

Leaderboard
Public
Private

Visualization

Private Leaderboard

## Create an algorithm to distinguish dogs from cats

In this competition, you'll write an algorithm to classify whether images contain either a dog or a cat.  This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.

We can learn a classifier that makes no mistakes when:

- we have sufficient data
- function class is sufficiently flexible,
- there is no noise i.e. the *true* mapping between $X$ and $Y$ is deterministic

In practice:

- data are limited
- we don't want function classes that are too flexible c.f. *overfitting, bias vs. variance tradeoff*
- real-world uncertainty e.g. hidden variables, measurement error.

Thus in most realistic scenarios, all classifiers will eventually make mistakes!

We can learn a classifier that makes no mistakes when:

- we have sufficient data
- function class is sufficiently flexible,
- there is no noise i.e. the *true* mapping between $X$ and $Y$ is deterministic

In practice:

- data are limited
- we don't want function classes that are too flexible c.f. *overfitting, bias vs. variance tradeoff*
- real-world uncertainty e.g. hidden variables, measurement error.

Thus in most realistic scenarios, all classifiers will eventually make mistakes!

We can learn a classifier that makes no mistakes when:

- we have sufficient data
- function class is sufficiently flexible,
- there is no noise i.e. the *true* mapping between $X$ and $Y$ is deterministic

In practice:

- data are limited
- we don't want function classes that are too flexible c.f. *overfitting, bias vs. variance tradeoff*
- real-world uncertainty e.g. hidden variables, measurement error.

Thus in most realistic scenarios, all classifiers will eventually make mistakes!

# Which kinds of mistakes are (more) acceptable?

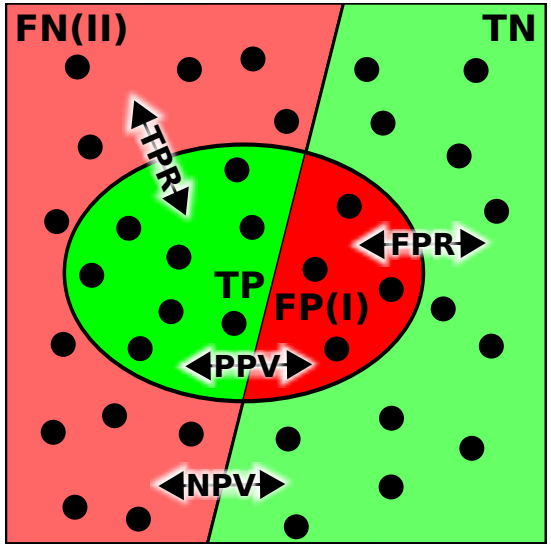
# Which kinds of mistakes are (more) acceptable?

## Case Study

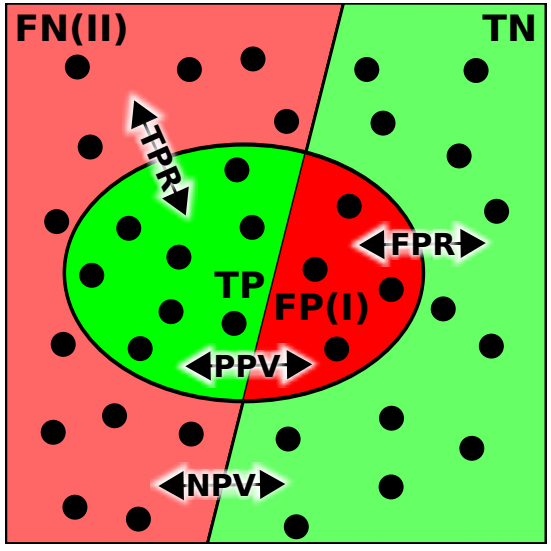
A medical test determines that a patient has a 30% chance of having a fatal disease. Should the doctor treat the patient?

- choosing to treat a healthy patient (false positive) increases risk of side effects.
- choosing not to treat a sick patient (false negative) could lead to serious issues.

# The confusion matrix $\mathbf{C}$ summarizes classifier mistakes



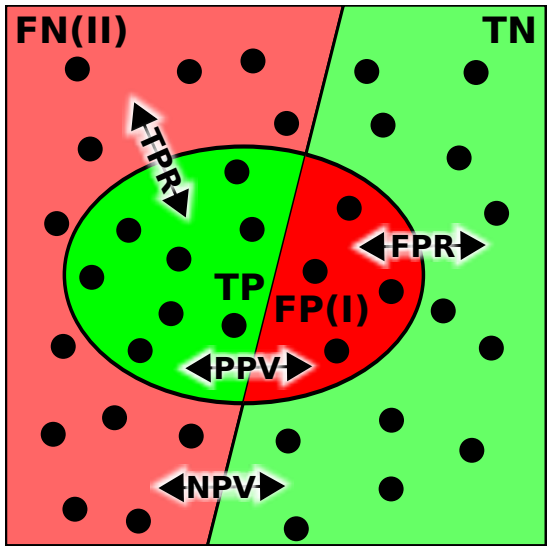
# The confusion matrix $\mathbf{C}$ summarizes classifier mistakes




Let $X, Y \sim P$

| | $Y = 1$ | $Y = 0$ |
|---|---|---|
| $\theta = 1$ | TP<br>P($Y = 1, \theta = 1$) | FP<br>P($Y = 0, \theta = 1$) |
| $\theta = 0$ | FN<br>P($Y = 1, \theta = 0$) | TN<br>P($Y = 0, \theta = 0$) |

# The confusion matrix $\mathbf{C}$ summarizes classifier mistakes




Let $X, Y \sim P$

| | $Y = 1$ | $Y = 0$ |
|---|---|---|
| $\theta = 1$ | TP<br>P($Y = 1, \theta = 1$) | FP<br>P($Y = 0, \theta = 1$) |
| $\theta = 0$ | FN<br>P($Y = 1, \theta = 0$) | TN<br>P($Y = 0, \theta = 0$) |

- we can approximate the confusion using finite samples e.g.

$$\widehat{\mathsf{TP}} = \frac{1}{n}\sum_{i=1}^{n} 1_{[y_i=1,\, \theta(x_i)=1]}, \quad \widehat{\mathsf{FP}} = \frac{1}{n}\sum_{i=1}^{n} 1_{[y_i=0,\, \theta(x_i)=1]}.$$

# Performance metrics

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

Examples

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

and many more . . .

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

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

# Performance metrics

Now we may 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 = $w_1\mathsf{FP} + w_2\mathsf{FN}$, where $w_2 \gg w_1$

and many more . . .

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

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

# Performance metrics

Now we may 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 = $w_1$FP + $w_2$FN, where $w_2 \gg w_1$

and many more . . .

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

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

# Performance metrics

Now we may 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 = $w_1$FP + $w_2$FN, where $w_2 \gg w_1$

## and many more . . .

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

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

# Utility & Regret

- performance is measured via utility $\mathcal{U}(\theta, P) = \Phi(\mathbf{C})$
- 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 $\theta$ is given by:

$$\mathcal{R}(\theta, P) = \mathcal{U}(\theta^*, P) - \mathcal{U}(\theta, P)$$

# Utility & Regret

- performance is measured via utility $\mathcal{U}(\theta, P) = \Phi(\mathbf{C})$
- 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 $\theta$ is given by:

$$\mathcal{R}(\theta, P) = \mathcal{U}(\theta^*, P) - \mathcal{U}(\theta, P)$$

# Utility & Regret

- performance is measured via utility $\mathcal{U}(\theta, P) = \Phi(\mathbf{C})$
- 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 $\theta$ 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 $\theta_n \big| \mathcal{D}_n$

**Example**

Empirical risk minimization via SVM:

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{\{x_i, y_i\} \in \mathcal{D}_n} \max\left(0, 1 - y_i f(x_i)\right)$$

$$\theta_n = \operatorname{sign}(f^*)$$

# 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:

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{\{x_i, y_i\} \in \mathcal{D}_n} \max\left(0, 1 - y_i f(x_i)\right)$$

$$\theta_n = \mathsf{sign}(f^*)$$

# 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:

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{\{x_i, y_i\} \in \mathcal{D}_n} \max\left(0, 1 - y_i f(x_i)\right)$$

$$\theta_n = \mathsf{sign}(f^*)$$

# 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 eventually estimates the Bayes optimal classifier

Consistency is a desirable property:

- implies stability of the classification procedure, related to generalization ability
- interestingly, seeking consistent classifiers is often *easier* than direct optimization!

# 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 eventually estimates the Bayes optimal classifier

Consistency is a desirable property:

- implies stability of the classification procedure, related to generalization ability
- interestingly, seeking consistent classifiers is often *easier* than direct optimization!

# Optimal classification for Decomposable Metrics

Consider the empirical accuracy:

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

- Observe that the classification problem

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

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

Consider the empirical accuracy:

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

- Observe that the classification problem

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

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

# Bayes Optimal Classifier

## Population Accuracy

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

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

## Weighted Accuracy

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

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

# Bayes Optimal Classifier

## Population Accuracy

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

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

## Weighted Accuracy

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

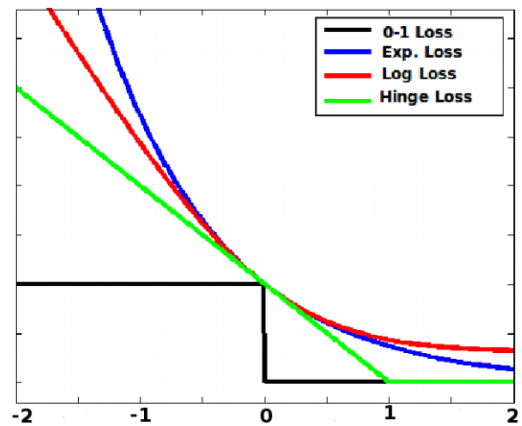
- Scott (2012) showed that $\theta^*(\mathbf{x}) = \text{sign}\left(P(Y=1|\mathbf{x}) - \rho\right)$

# 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:

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

- 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

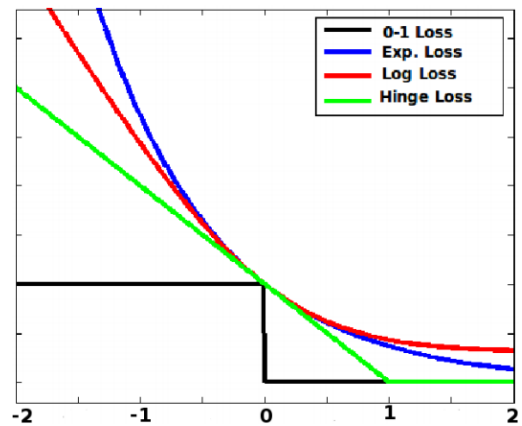


# Where do surrogates come from?

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

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

- 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

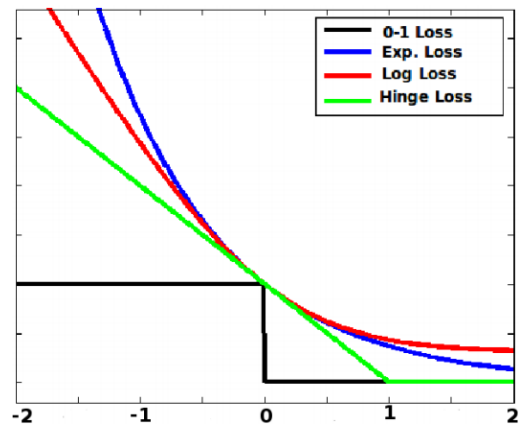


# Where do surrogates come from?

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

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

- 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 $\Phi(\theta, \mathbf{C}) = \langle \mathbf{A}, \mathbf{C} \rangle$
- $F_\beta$, Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. $\mathbf{C}$

**Question**

Is decomposability necessary for the optimal classifier to be *simple* i.e. a pointwise thresholding?

No! counter-examples include $F_\beta$ (Ye et al., 2012), Fractional-linear (Koyejo et al., 2014), Monotonic metrics (Narasimhan et al., 2014), min-max metric (Poor, 2013)

# Non-decomposability

- A common theme so far is *decomposability* i.e. linearity wrt. confusion matrix $\Phi(\theta, \mathbf{C}) = \langle \mathbf{A}, \mathbf{C} \rangle$
- $F_\beta$, Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. $\mathbf{C}$

## Question

Is decomposability necessary for the optimal classifier to be *simple* i.e. a pointwise thresholding?

No! counter-examples include $F_\beta$ (Ye et al., 2012), Fractional-linear (Koyejo et al., 2014), Monotonic metrics (Narasimhan et al., 2014), min-max metric (Poor, 2013)

# Non-decomposability

- A common theme so far is *decomposability* i.e. linearity wrt. confusion matrix $\Phi(\theta, \mathbf{C}) = \langle \mathbf{A}, \mathbf{C} \rangle$
- $F_\beta$, Jaccard, AUC and other common utility functions are *non-decomposable* i.e. non-linear wrt. $\mathbf{C}$

## Question

Is decomposability necessary for the optimal classifier to be *simple* i.e. a pointwise thresholding?

No! counter-examples include $F_\beta$ (Ye et al., 2012), Fractional-linear (Koyejo et al., 2014), Monotonic metrics (Narasimhan et al., 2014), min-max metric (Poor, 2013)

# Optimal classification for Non-decomposable Metrics

# The unreasonable effectiveness of thresholding

Some notation: $\eta_x = P(Y = 1|X = x)$, $\pi = P(Y = 1)$

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

Let $\mathcal{U}$ be either:

1. fractional-linear i.e. $\Phi(\mathbf{C}) = \frac{\langle \mathbf{A}, \mathbf{C} \rangle}{\langle \mathbf{B}, \mathbf{C} \rangle}$
2. or differentiable and monotonically increasing wrt. TP and TN

then $\exists$ an oracle $\delta^*$ s.t. if $P(\eta_x = \delta^*) = 0$, the Bayes optimal classifier satisfies:

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

condition $P(\eta_x = \delta^*) = 0$ is easily satisfied e.g. when $P(X)$ is continuous.

# The unreasonable effectiveness of thresholding

Some notation: $\eta_x = P(Y = 1|X = x)$, $\pi = P(Y = 1)$

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

Let $\mathcal{U}$ be either:

1. fractional-linear i.e. $\Phi(\mathbf{C}) = \frac{\langle \mathbf{A}, \mathbf{C} \rangle}{\langle \mathbf{B}, \mathbf{C} \rangle}$
2. or differentiable and monotonically increasing wrt. TP and TN

then $\exists$ an oracle $\delta^*$ s.t. if $P(\eta_x = \delta^*) = 0$, the Bayes optimal classifier satisfies:

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

condition $P(\eta_x = \delta^*) = 0$ is easily satisfied e.g. when $P(X)$ is continuous.

# Proof Sketch

- Consider the relaxed problem:

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

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

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

# Proof Sketch

- Consider the relaxed problem:

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

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

# Proof Sketch

- Consider the relaxed problem:

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

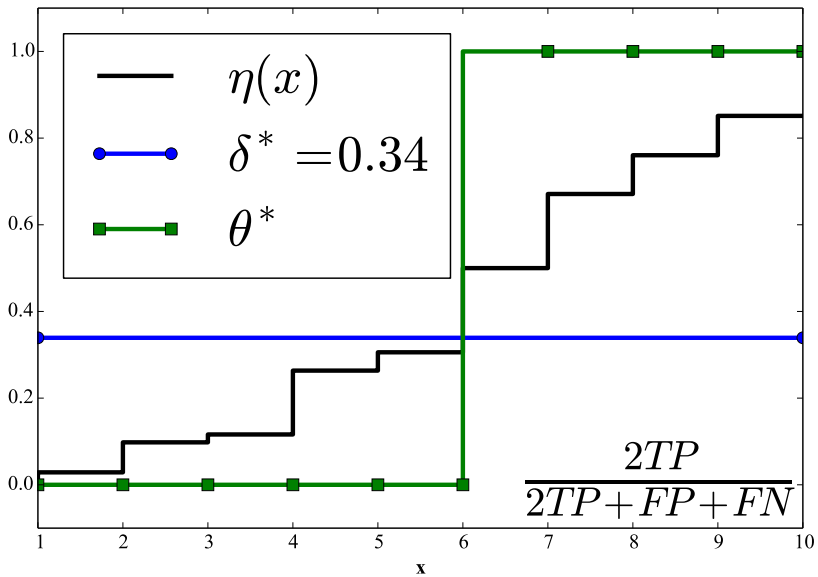
  where $\mathcal{F} = \{f \mid f : \mathcal{X} \mapsto [0, 1]\}$
- Show that the optimal "relaxed" classifier is $\theta^*_{\mathcal{F}} = \text{sign}(\eta_x - \delta^*)$
- Observe that $\Theta \subset \mathcal{F}$. Thus $\mathcal{U}(\theta^*_{\mathcal{F}}, \mathcal{P}) \geq \mathcal{U}(\theta^*_{\Theta}, \mathcal{P})$.
- As a result, $\theta^*_{\mathcal{F}} \in \Theta$ implies that $\theta^*_{\mathcal{F}} \equiv \theta^*_{\Theta}$.

# Some recovered and new results

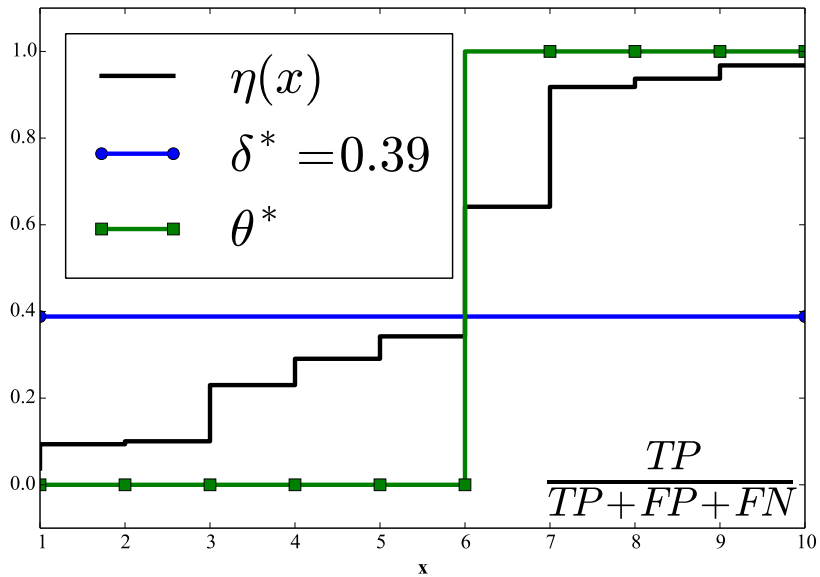
| Metric | Form | Optimal threshold |
|---|---|---|
| $F_\beta$ | $\frac{(1+\beta^2)TP}{(1+\beta^2)TP+\beta^2FN+FP}$ | $\delta^* = \frac{\mathcal{L}^*}{1+\beta^2}$ |
| Cost-sensitive learning | $c_0 + c_1TP + c_2\gamma(\theta)$ | $\delta^* = -\frac{c_2}{c_1}$ |
| Precision | $\frac{TP}{TP+FP}$ | $\delta^* = \mathcal{L}^*$ |
| Recall | $\frac{TP}{TP+FN}$ | $\delta^* = 0$ |
| Weighted Accuracy | $\frac{2(TP+TN)}{2(TP+TN)+FP+FN}$ | $\delta^* = \frac{1}{2}$ |
| Jaccard Coefficient | $\frac{TP}{TP+FP+FN}$ | $\delta^* = \frac{\mathcal{L}^*}{1+\mathcal{L}^*}$ |

# Simulated examples

$F_1$




Jaccard




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

# Empirical estimation via threshold search

## Step 1

- **Option 1:** estimate for $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010), then

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

- **Option 2:** 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

$\max_\delta \mathcal{U}(\hat{\theta}_\delta, \mathcal{D}_n)$ is one dimensional, efficiently computable using exhaustive search (Sergeyev, 1998).

# Empirical estimation via threshold search

## Step 1

- **Option 1:** estimate for $\hat{\eta}_x$ via. proper loss (Reid and Williamson, 2010), then

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

- **Option 2:** 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

$\max_\delta \mathcal{U}(\hat{\theta}_\delta, \mathcal{D}_n)$ is one dimensional, efficiently computable using exhaustive search (Sergeyev, 1998).

# Consistency

## Threshold search is consistent (Koyejo et al., 2014)

$$\mathcal{R}(\hat{\theta}_\delta, P) \xrightarrow{n \to \infty} 0$$

- Threshold search is $\mathcal{O}(n^2)$ with naïve implementation, $\mathcal{O}(n \log n)$ by pre-sorting $\hat{\eta}_x$, difficult analyze convergence.
- Cutting plane surrogate methods (Joachims, 2005) may have exponential complexity, and limited statistical guarantees.

## Motivating questions

- Can we improve on the computational complexity of threshold search?
- What is the convergence rate of the resulting procedure?

# Consistency

## Threshold search is consistent (Koyejo et al., 2014)

$$\mathcal{R}(\hat{\theta}_\delta, P) \xrightarrow{n \to \infty} 0$$

- Threshold search is $\mathcal{O}(n^2)$ with naïve implementation, $\mathcal{O}(n \log n)$ by pre-sorting $\hat{\eta}_x$, difficult analyze convergence.
- Cutting plane surrogate methods (Joachims, 2005) may have exponential complexity, and limited statistical guarantees.

## Motivating questions

- Can we improve on the computational complexity of threshold search?
- What is the convergence rate of the resulting procedure?

# Consistency

## Threshold search is consistent (Koyejo et al., 2014)

$$\mathcal{R}(\hat{\theta}_\delta, P) \xrightarrow{n \to \infty} 0$$

- Threshold search is $\mathcal{O}(n^2)$ with naïve implementation, $\mathcal{O}(n \log n)$ by pre-sorting $\hat{\eta}_x$, difficult analyze convergence.
- Cutting plane surrogate methods (Joachims, 2005) may have exponential complexity, and limited statistical guarantees.

## Motivating questions

- Can we improve on the computational complexity of threshold search?
- What is the convergence rate of the resulting procedure?

# 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 $x^* \in \operatorname{argmin} f(x)$. 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

## 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 $\epsilon$ 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, $\mathcal{R}(\hat{\theta}_{\delta_t}, \mathcal{P}) \leq \frac{C \sum_{i=1}^{t} r_i}{t}$

# 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 $\epsilon$ 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, $\mathcal{R}(\hat{\theta}_{\delta_t}, \mathcal{P}) \leq \frac{C \sum_{i=1}^{t} r_i}{t}$

# 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 $\epsilon$ 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, $\mathcal{R}(\hat{\theta}_{\delta_t}, \mathcal{P}) \leq \frac{C \sum_{i=1}^{t} r_i}{t}$

# 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 $\epsilon$ 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, $\mathcal{R}(\hat{\theta}_{\delta_t}, \mathcal{P}) \leq \frac{C \sum_{i=1}^{t} r_i}{t}$

# Examples

## Ordinary logistic regression

Sample complexity of ordinary logistic regression is $O(\frac{1}{\sqrt{n}})$. Thus, batch algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.

## Regularized logistic regression

Consider high dimensional ($p \gg n$) regularized M-estimation (Negahban et al., 2009). Under regularity conditions, the $\ell_2$ estimation error is upper bounded by $O\left(\frac{s \log p}{n}\right)$. Thus, batch algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.

## Online algorithm

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.

# Examples

## Ordinary logistic regression

Sample complexity of ordinary logistic regression is $O(\frac{1}{\sqrt{n}})$. Thus, batch algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.

## Regularized logistic regression

Consider high dimensional ($p \gg n$) regularized M-estimation (Negahban et al., 2009). Under regularity conditions, the $\ell_2$ estimation error is upper bounded by $O\left(\frac{s \log p}{n}\right)$. Thus, batch algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.

## Online algorithm

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.

# Examples

## Ordinary logistic regression

Sample complexity of ordinary logistic regression is $O(\frac{1}{\sqrt{n}})$. Thus, batch algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.

## Regularized logistic regression

Consider high dimensional ($p \gg n$) regularized M-estimation (Negahban et al., 2009). Under regularity conditions, the $\ell_2$ estimation error is upper bounded by $O\left(\frac{s \log p}{n}\right)$. Thus, batch algorithm achieves $O(\frac{1}{\sqrt{n}})$ regret.

## Online algorithm

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% |

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

# Batch algorithm

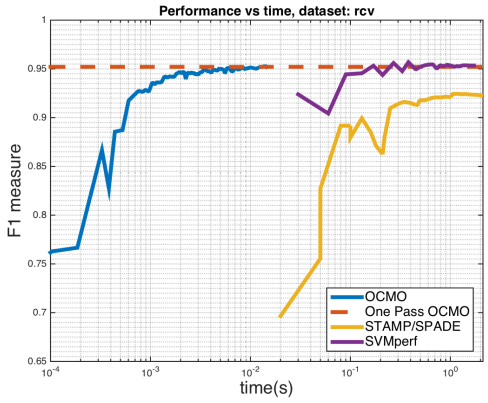
| Data set/Metric | LR+Plug-in | LR+Batch | XGB+Plug-in | XGB+Batch |
|---|---|---|---|---|
| news20-Q-Mean | 0.948 (3.77s) | 0.948 (0.001s) | 0.874 (3.87s) | 0.875 (0.003s) |
| news20-H-Mean | 0.950 (3.70s) | 0.950 (0.003s) | 0.859 (3.61s) | 0.860 (0.003s) |
| news20-F1 | 0.949 (3.49s) | 0.948 (0.01s) | 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)

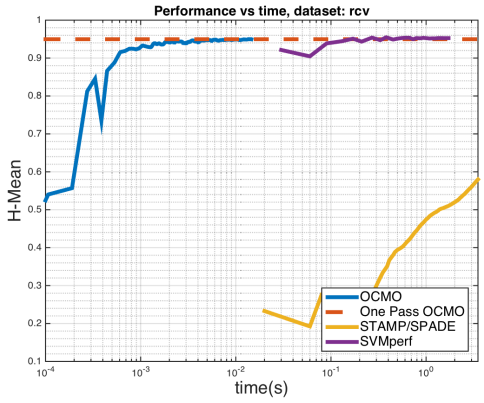
| Metric | Algorithm | RCV1 | Epsilon | KDD-A | KDD-B |
|---|---|---|---|---|---|
| F1 | OCMO | 0.952 (0.01s) | 0.804 (4.87s) | 0.934 (2.43s) | 0.941 (5.01s) |
| | sTAMP | 0.923 (14.44s) | 0.585 (133.23s) | - | - |
| | $SVM^{perf}$ | 0.953 (1.72s) | 0.872 (20.39s) | - | - |
| H-Mean | OCMO | 0.964 (0.02s) | 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.

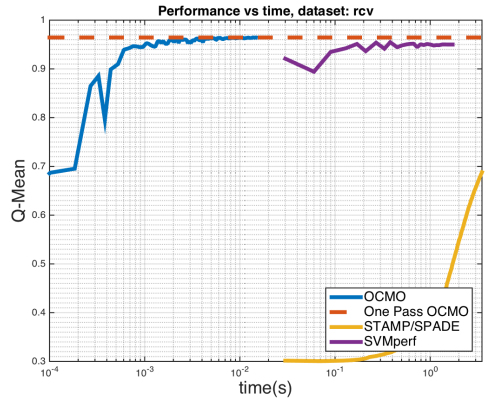
# Performance vs run time for various online algorithms




(a) F1 measure on rcv1




(b) H-Mean on rcv1




(c) Q-Mean on rcv1

# Conclusion

# Conclusion and open questions

- Optimal classifiers for a large family of binary 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?
- Can we construct surrogate loss functions i.e. which avoid estimating $P(Y = 1|X)$?

# Conclusion and open questions

- Optimal classifiers for a large family of binary 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?
- Can we construct surrogate loss functions i.e. which avoid estimating $P(Y = 1|X)$?

# Conclusion and open questions

- Optimal classifiers for a large family of binary 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?
- Can we construct surrogate loss functions i.e. which avoid estimating $P(Y = 1|X)$?

# Conclusion and open questions

- Optimal classifiers for a large family of binary 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?
- Can we construct surrogate loss functions i.e. which avoid estimating $P(Y = 1|X)$?

# Conclusion and open questions

- Optimal classifiers for a large family of binary 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?
- Can we construct surrogate loss functions i.e. which avoid estimating $P(Y = 1|X)$?

# 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.

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.

Sahand Negahban, Bin Yu, Martin J Wainwright, and Pradeep K Ravikumar. A unified framework for high-dimensional analysis of $m$-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, pages 1348–1356, 2009.

H Vincent Poor. *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

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

1: **Input:** Training sample $\{X_i, Y_i\}_{i=1}^n$, utility measure $\mathcal{U}$, 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 $\nabla\mathcal{U}$;
8: $\delta \leftarrow \delta - \alpha \frac{\nabla\mathcal{U}}{\|\nabla\mathcal{U}\|}$.
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 $\alpha$;

Initilize $\eta_0$, $\delta_0 = 0.5$;
**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 $(\eta_t, \delta_t)$.