Goal-Oriented Learning

Rizal Fathony
Post-Doctoral Fellow @ Carnegie Mellon University

Machine Learning

Machine Learning Applications

Successful applications across different areas

Search engine

E-commerce

Computer Vision

Speech Recognition

Text Analysis

Health & Medical

Financial

Industry

Bioinformatics

Science

Machine Learning Applications

Successful applications across different areas

Search engine

E-commerce

Computer Vision

Speech Recognition

Text Analysis

Health & Medical

Financial

Industry

Bioinformatics

Science

Despite of the success stories

A missing piece

in the current learning algorithms

Machine Learning Pipeline

- Formulate a problem
- Prepare data
- Choose an evaluation metric
- Choose a model
- Train the model
- Evaluate the performance

Evaluation Metric

Example: Digit Recognition

Evaluation Metric:

Performance Metric: Accuracy

$$\text{Accuracy} = \frac{\text{\# correct prediction}}{\text{\# sample}}$$

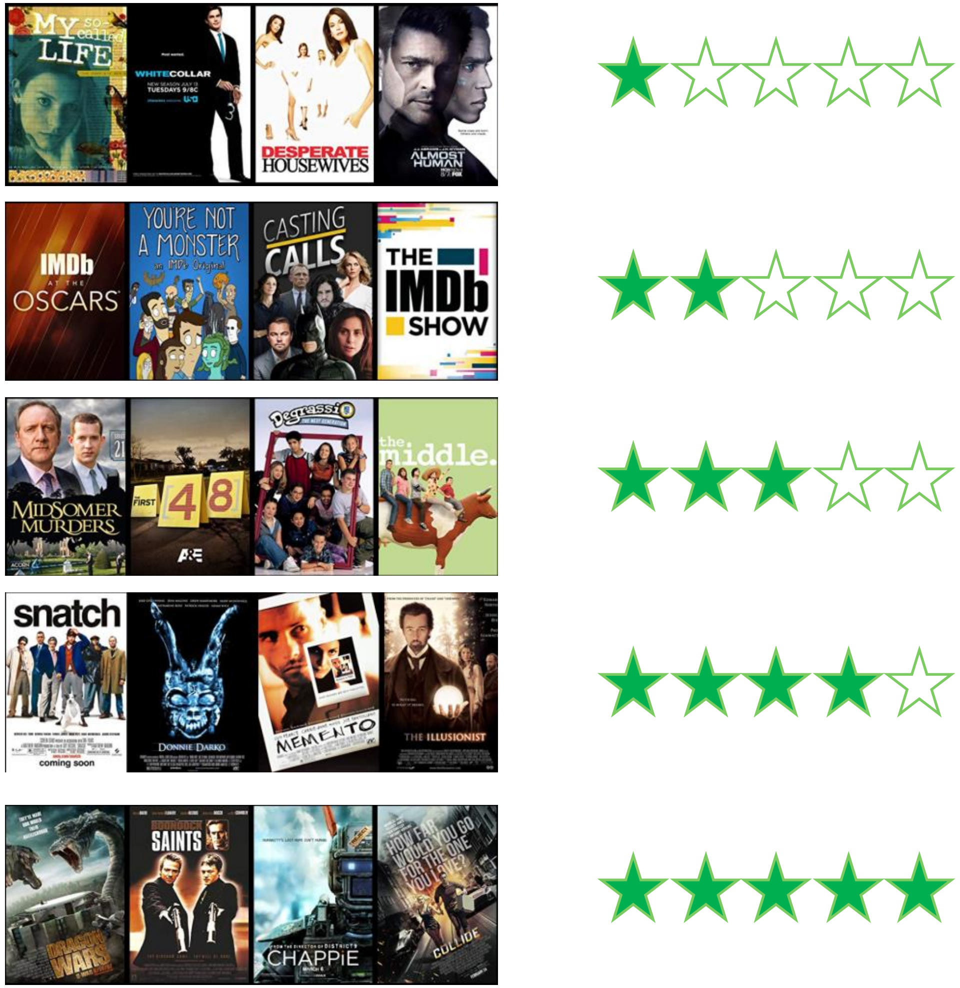
Loss Metric: Zero-One Loss

$$\text{Zero-One Loss} = \frac{\text{\# incorrect prediction}}{\text{\# sample}}$$

Most widely used metric!

Accuracy metric is not always perfect

Example: Movie Rating Prediction



Evaluation Metric:

Accuracy metric:

does not consider distances

Predicted vs Actual Label:

Distance ⬆ ⟶ Loss ⬆

Loss Metric: Absolute Loss

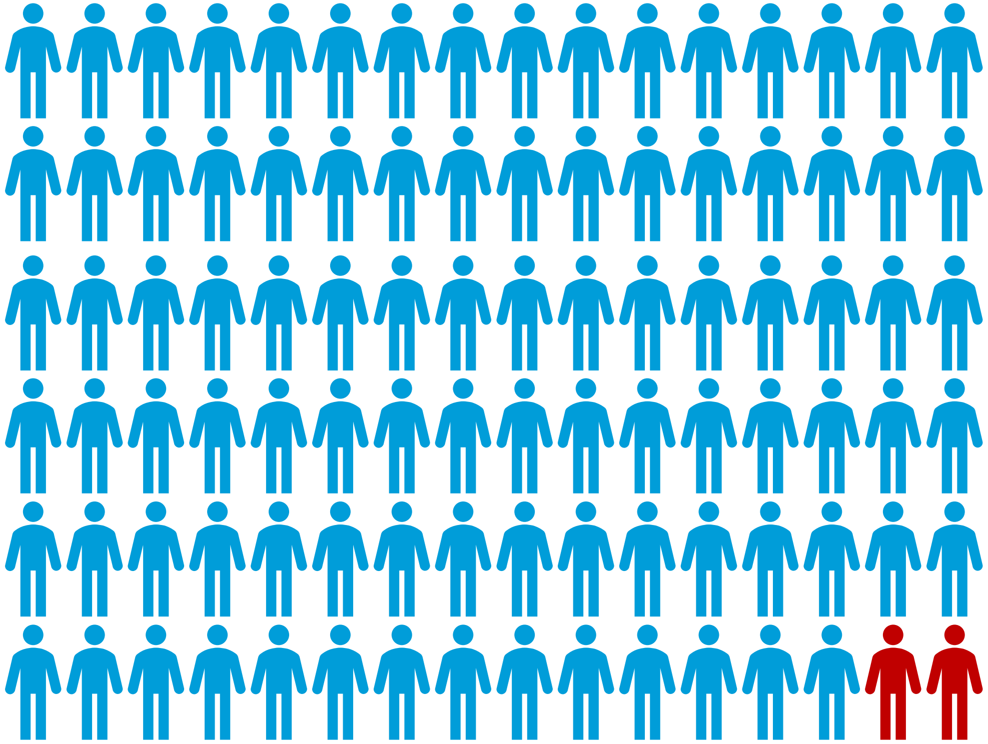
$$\text{AbsoluteLoss} = \frac{1}{n}\sum_i |\hat{y}_i - y_i|$$

$\hat{y}_i$: predicted label

y_i : true label

Accuracy metric is not always desirable

Example: Disease Prediction
(imbalance dataset)



98% of the samples: healthy (negative samples)
2% of the samples: have disease (positive samples)

Predict all samples as negative:
Accuracy metric: 98%

Confusion Matrix

		Actual		
		Positive	Negative	
Pred.	Positive	True Pos. (TP)	False Pos. (FP)	Predicted Pos. (PP)
	Negative	False Neg. (FN)	True Neg. (TN)	Predicted Neg. (PN)
		Actual Pos. (AP)	Actual Neg. (AN)	All Data (ALL)

$$\text{Precision} = \frac{\#\ \text{true positive}}{\#\ \text{predicted positive}} \qquad \text{Recall} = \frac{\#\ \text{true positive}}{\#\ \text{actual positive}}$$

$$\text{Specificity} = \frac{\#\ \text{true negative}}{\#\ \text{actual negative}} \qquad \text{Sensitivity} = \frac{\#\ \text{true positive}}{\#\ \text{actual positive}}$$

$$\text{F1-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$F_\beta\text{-score} = \frac{(1+\beta^2) \cdot \text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

External data is needed

Example: Stock market prediction

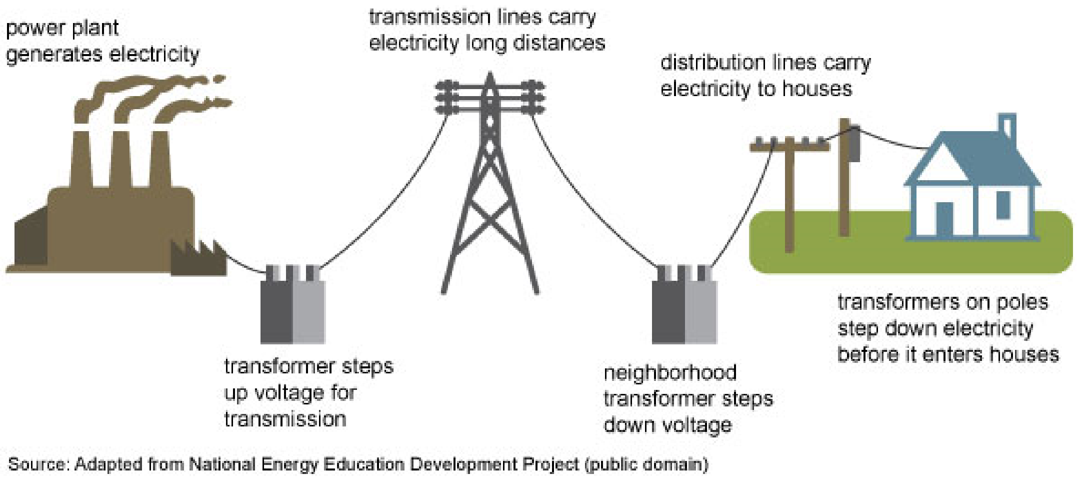
Prediction tasks:

Predict the stock prices

Evaluation metric:

Revenue when making investments based on the prediction

Example: Electricity demand prediction



Prediction tasks:

Predict the electricity demands on a certain time

Evaluation metric:

The cost of electricity generation given the prediction

Machine Learning Pipeline

Formulate a problem

Prepare data

Choose an evaluation metric

Choose a model

Train the model

Evaluate the performance

Goal vs Training Model Mismatch (1)

Example: Disease prediction

Choose an evaluation metric

vs

Choose a model

Train the model

Goal: optimize specificity & sensitivity

Most of ML models:
- No support for specificity & sensitivity metric
- Optimize the cross-entropy objective
  (a proxy for accuracy metric)

Goal vs Training Model Mismatch (2)

Example: Electricity demand prediction

Choose an evaluation metric

vs

Choose a model

Train the model

Goal: minimize the cost of electricity production

Most of the existing models:
- optimize cross-entropy (discrete models)
- optimize mean squared error (continuous models)

Goal vs Training Model Mismatch (3)

Machine Learning Tasks	Evaluation Metrics	Common Training Objectives
Medical/health areas	Specificity & sensitivity	Cross entropy
Text classification	Precision, Recall, F1-score	Cross entropy
Classification with imbalance data	F1-score, AUC, MCC	Cross entropy
Rating prediction	Absolute loss, Kappa score	Cross entropy, MSE
Electricity prediction	Electricity production cost	Cross entropy, MSE
Stock market prediction	Revenue	MSE

Discrepancy:
Evaluation metrics vs training objective → Inferior performance results
(Cortes & Mohri, 2004; Eban et.al, 2016)

Real World Consequence

Discrepancy:
Evaluation metrics vs training objective → Inferior performance results → Real world consequence

Machine Learning Tasks	Results	Real world consequence
Disease prediction	Suboptimal prediction performance	Inaccurate disease test
Online advertising prediction	Misplaced ads	Revenue Lost
Electricity prediction	Over-production	Increasing production cost
Stock market prediction	Suboptimal prediction	Revenue Lost

Bringing
Evaluation metric + training model
in harmony

Goal-Oriented Learning

Choose an evaluation metric

Choose a model

Train the model

Outline of the Talk

1 Motivation

2 Current Approaches

3 A New Learning Framework

4 Designing Learning Algorithms

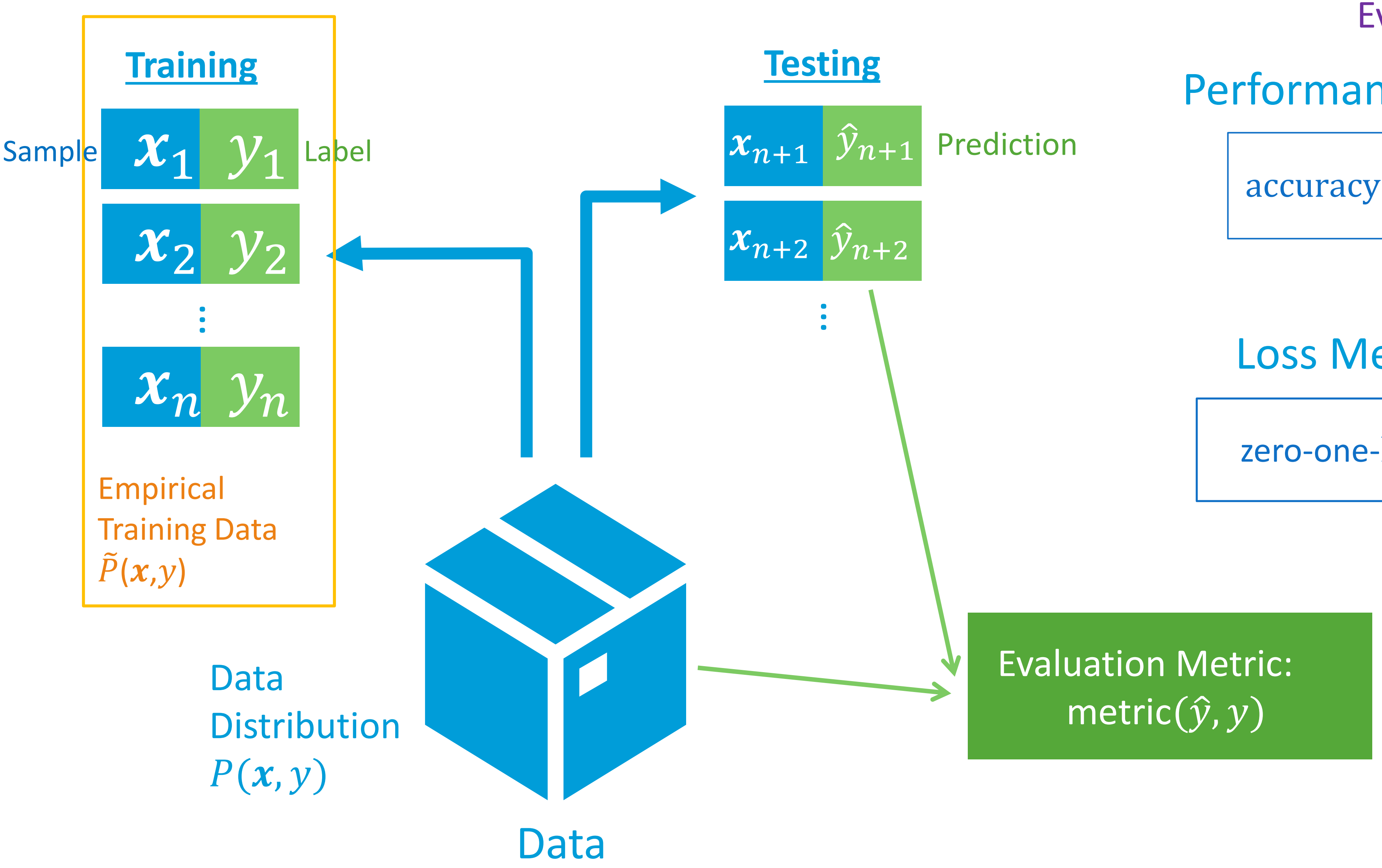
5 Summary and Potentials

6 Future Directions

Current Approaches

Approach for designing learning algorithms

Supervised Learning | Binary Classification



Evaluation Metric:

Performance Metric: Accuracy

$$\text{accuracy}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n} \sum_i I(\hat{y}_i = y_i)$$

Correct prediction

Loss Metric: Zero-One Loss

$$\text{zero-one-loss}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n} \sum_i I(\hat{y}_i \neq y_i)$$

Incorrect prediction

Standard Approach for Learning Algorithms

Empirical Risk Minimization (ERM) [Vapnik, 1992]

- Assumes a family of parametric hypothesis function f (e.g. linear discriminator)
- Finds the hypothesis f^* that minimize the empirical risk:

Loss metric: e.g. zero-one loss metric

Empirical loss

$$\min_f \frac{1}{n}\sum_{i=1}^{n} \text{loss}(f(\mathbf{x}_i), y_i) = \min_f \mathbb{E}_{\tilde{P}(\mathbf{x},y)}[\text{loss}(f(\mathbf{x}), y)]$$

Prediction True label

Empirical Training Data

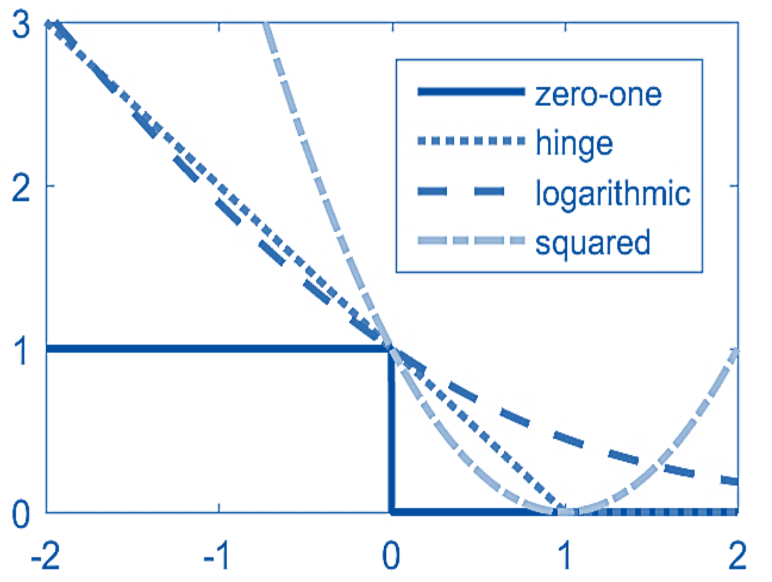
Intractable optimization!

Since the zero-one loss (accuracy) is: discrete & non-continuous

(Steinwart and Christmann, 2008)

Surrogate Losses

ERM: prescribes the use of convex surrogate loss to avoid intractability



Example: Binary classification with accuracy metric

Original loss metric: discrete

$$\min_f \mathbb{E}_{\tilde{P}(\mathbf{x},y)} \left[\text{ZeroOneLoss}(f(\mathbf{X}), Y)\right]$$

Support Vector Machine (SVM)

$$\min_f \mathbb{E}_{\tilde{P}(\mathbf{x},y)} \left[\text{HingeLoss}(f(\mathbf{x}), y)\right]$$

Convex surrogate loss

Logistic Regression (LR)

Probabilistic prediction

$$\min_f \mathbb{E}_{\tilde{P}(\mathbf{x},y)} \left[\text{LogLoss}(\hat{P}_f(\hat{y}|\mathbf{x}), y)\right]$$

Convex surrogate loss

More Complex Evaluation Metrics

ERM: Extend the binary surrogate losses to the settings.

Binary classification | accuracy

SVM & Logistic Regression:

✓ Perform well in practice

✓ Statistical consistency

SVM: ✓ Dual sparsity
(solution depends on few samples)

Multiclass classification | accuracy

Multiclass SVMs: many formulations.
Each formulation lacks one or more:

✗ Lacks statistical consistency

✗ Do not perform well in practice

(Tewari & Bartlett, 2006; Liu 2007; Dogan et.al. 2016)

More complex evaluation metrics

Logistic Regression-based model: None

✗ No model for complex metrics

SVM-based model: SVM-perf (Joachims, 2005)

✓ Works on many complex metrics

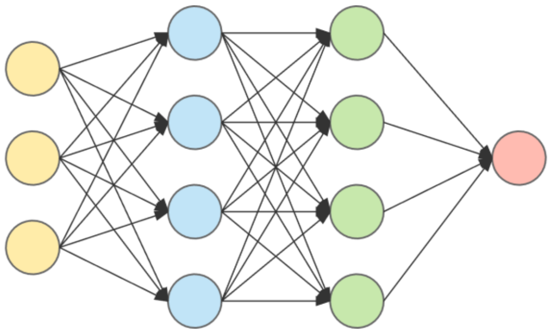
✗ Lacks statistical consistency

✗ Does not provide easy tool to extend the method to custom metrics

Most of other models:

✗ Hard to extend to custom metrics

Neural Networks Learning



Currently the popular machine learning model.

Use the ‘classical’ surrogate losses as the last layer (objective).

Binary & multiclass classification

Objective: Cross entropy objective

= Logistic regression (log-loss surrogate)

More complex evaluation metrics

Most of ‘classical’ models:

❌ Not applicable to NN learning

NN-targeted models:
(Eban et.al, 2016; Song et.al, 2016; Sanyal, et.al, 2018)

❌ Only support few metrics

❌ No support for custom metrics

Practitioners’ perspective

Aim to optimize an evaluation metric tailored specifically for their problem.
(e.g. specificity, sensitivity, F-beta score)

No learning models can optimize their specific evaluation metrics.

Choose the standard cross entropy instead

Mismatch between Goal vs Training Model

A New Learning Framework

A different approach on designing learning algorithms

A Different Approach in Learning Algorithm Design

Empirical Risk Minimization

Optimize Original Evaluation Metric

Discrete, Intractable

Approximate the metric

Exact training data

Empirical Risk Minimization with Convex Surrogate Loss

Convex, Tractable

More complex metric → Harder to construct good surrogate losses

Exact evaluation metric

Approximate training data

Adversarial Prediction

(Fathony et.al, '18; Asif et.al '16)

Convex, Tractable

No need to independently construct surrogate loss for every metric

Adversarial Prediction

Adversarial Prediction (Fathony et.al, 2018; Asif et.al, 2016)

Original Loss Metric

Discrete, Intractable

Original loss metric

$$\min_f \mathbb{E}_{\tilde{P}(\mathbf{x},y)} [\mathrm{loss}(f(\mathbf{x}), y)]$$

Empirical data (x, y)

Approximate the loss metric with convex surrogates →

Empirical Risk Minimization

Convex, Tractable

$$\min_f \mathbb{E}_{\tilde{P}(\mathbf{x},y)} [\mathrm{surrogate}(f(\mathbf{x}), y)]$$

Empirical data (x, y)

Convex surrogate loss

↓ Probabilistic prediction

Predictor

Original loss metric

$$\min_{\mathcal{P}(\hat{y}|\mathbf{x})} \mathbb{E}_{\tilde{P}(\mathbf{x},y)\mathcal{P}(\hat{y}|\mathbf{x})} [\mathrm{loss}(\hat{y}, y)]$$

Probabilistic predictor

Empirical data (x, y)

↓ Evaluate against an adversary instead of empirical label

Predictor Adversary

Original loss metric

$$\min_{\mathcal{P}(\hat{y}|\mathbf{x})} \max_{\mathcal{Q}(\check{y}|\mathbf{x})} \mathbb{E}_{\tilde{P}(\mathbf{x})\mathcal{P}(\hat{y}|\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})} [\mathrm{loss}(\hat{y}, \check{y})]$$

Predictor's Probability

Adversary's probability

Empirical sample (x only)

Constraint the adversary →

Adversarial Prediction

Convex, Tractable

Predictor Adversary

Original loss metric

$$\min_{\mathcal{P}(\hat{y}|\mathbf{x})} \max_{\mathcal{Q}(\check{y}|\mathbf{x}) \in \Xi} \mathbb{E}_{\tilde{P}(\mathbf{x})\mathcal{P}(\hat{y}|\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})} [\mathrm{loss}(\hat{y}, \check{y})]$$

$$\Xi \triangleq \{\mathcal{Q} \mid \mathbb{E}_{\tilde{P}(\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})}[\phi(\mathbf{x}, \check{y})] = \mathbb{E}_{\tilde{P}(\mathbf{x},y)}[\phi(\mathbf{x}, y)]\}$$

Features

Features

Empirical data (x, y)

Adversary distribution's statistics

Empirical statistics

Use original evaluation metric	Approximate the training data

Adversarial Prediction: Dual Formulation

Primal:

$$\min_{\mathcal{P}(\hat{y}|\mathbf{x})} \max_{\mathcal{Q}(\check{y}|\mathbf{x}) \in \Xi} \mathbb{E}_{\tilde{P}(\mathbf{x})\mathcal{P}(\hat{y}|\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})} [\text{loss}(\hat{y}, \check{y})]$$

$$\Xi \triangleq \{\mathcal{Q} \mid \mathbb{E}_{\tilde{P}(\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})}[\phi(\mathbf{x}, \check{y})] = \mathbb{E}_{\tilde{P}(\mathbf{x},y)}[\phi(\mathbf{x}, y)]\}$$

Lagrange duality, minimax duality
Flip the optimization order

Dual:

$$\min_{\theta} \mathbb{E}_{\tilde{P}(\mathbf{x},y)} \left[\max_{\mathcal{Q}(\check{y}|\mathbf{x})} \min_{\mathcal{P}(\hat{y}|\mathbf{x})} \mathbb{E}_{\mathcal{P}(\hat{y}|\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})} [\text{loss}(\hat{y}, \check{y}) + \theta^{\intercal} (\phi(\mathbf{x}, \check{y}) - \phi(\mathbf{x}, y))] \right]$$

Empirical data

Adversary

Predictor

discrete loss metric

Feature differences (from the constraint)

Lagrange multiplier for the constraints

Convex w.r.t. θ

ERM: How to construct a surrogate loss for a given evaluation metric?

Adversarial Prediction: How to solve the maximin problem above?

Designing Learning Algorithms

Adversarial prediction formulations for various machine learning tasks

Evaluation Metrics

Decomposable Metrics

Can be decomposed into sample-wise sum

Example: accuracy, ordinal, taxonomy-based, classification with abstention metrics, and cost-sensitive metrics.

Binary and multiclass classification

Non-Decomposable Metrics

Cannot be decomposed into sample-wise sum

Example: F1-score, GPR, informedness, MCC, Kappa score.

Binary and multiclass classification

Decomposable Metrics

Decomposable Metrics

(Fathony et.al, NeurIPS 2016 & 2017, CoRR 2018)

Decomposable metrics:

$$\mathrm{loss}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_i \mathrm{loss}(\hat{y}_i, y_i)$$

Loss metric for the whole training set

Sample-wise loss metric

Vector notations:

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Example for binary classification

Dual:

$$\min_{\theta} \mathbb{E}_{\tilde{P}(\mathbf{x},y)} \left[\max_{\mathcal{Q}(\check{y}|\mathbf{x})} \min_{\mathcal{P}(\hat{y}|\mathbf{x})} \mathbb{E}_{\mathcal{P}(\hat{y}|\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})} \left[\mathrm{loss}(\hat{y}, \check{y}) + \theta^{\intercal} \left(\phi(\mathbf{x}, \check{y}) - \phi(\mathbf{x}, y) \right) \right] \right]$$

Simple loss metrics: Analytical solution

Example: - Zero-one loss metric (accuracy performance) [NeurIPS 2016]

- Absolute & squared loss metric [NeurIPS 2017]
- Classification with abstention [CoRR 2018]

Technique: Analyze the Nash equilibrium solution of the zero-sum game.

Decomposable Metrics | Complex Loss Metric

(Fathony et.al, CoRR 2018)

Decomposable metrics:

$$\text{loss}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_i \text{loss}(\hat{y}_i, y_i)$$

Loss metric for the whole training set

Sample-wise loss metric

Dual:

$$\min_\theta \mathbb{E}_{\tilde{P}(\mathbf{x},y)} \left[\max_{\mathcal{Q}(\check{y}|\mathbf{x})} \min_{\mathcal{P}(\hat{y}|\mathbf{x})} \mathbb{E}_{\mathcal{P}(\hat{y}|\mathbf{x})\mathcal{Q}(\check{y}|\mathbf{x})} \left[\text{loss}(\hat{y}, \check{y}) + \theta^\intercal \left(\phi(\mathbf{x}, \check{y}) - \phi(\mathbf{x}, y) \right) \right] \right]$$

More complex losses: Reformulation as a Linear Program

Example: - Taxonomy-based loss metric

- Cost-sensitive loss metric

Technique: Reformulate as a linear program, and use standard LP solver

size of the LP: k+1, where k = # of class

Example: Multiclass Classification with Accuracy Metric

(Fathony et.al, NeurIPS 2016)

	Dual Sparsity?	Statistical Consistency?	**Perform well** in low dimensional feature? (Dogan et.al., 2016)
Multiclass Logistic Regression	✗	✓	✓
Multiclass Support Vector Machine			
1. The WW Model (Weston et.al., 2002)	✓	✗	✓
2. The CS Model (Crammer and Singer, 1999)	✓	✗	✓
3. The LLW Model (Lee et.al., 2004)	✓	✓	✗
Adversarial Prediction	✓	✓	✓

Non-Decomposable Metrics

Non-Decomposable Metric

		Actual		
		Positive	Negative	
Pred.	Positive	True Pos. (TP)	False Pos. (FP)	Predicted Pos. (PP)
Pred.	Negative	False Neg. (FN)	True Neg. (TN)	Predicted Neg. (PN)
		Actual Pos. (AP)	Actual Neg. (AN)	All Data (ALL)

Example:

Binary Classification with F1-score metric

$$\text{F1-score}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\ \text{TP}}{\text{PP} + \text{AP}} = \frac{2\sum_i \hat{y}_i y_i}{\sum_i \hat{y}_i + \sum_i y_i}$$

Dual | Decomposable metric:

decomposable loss metric

$$\min_\theta \frac{1}{n} \sum_i \left[\max_{\mathcal{Q}(\check{y}_i|\mathbf{x}_i)} \min_{\mathcal{P}(\hat{y}_i|\mathbf{x}_i)} \sum_{\hat{y}_i, \check{y}_i} \mathcal{P}(\hat{y}_i|\mathbf{x}_i)\mathcal{Q}(\check{y}_i|\mathbf{x}_i) \left[\text{loss}(\hat{y}_i, \check{y}_i) + \theta^\intercal \left(\phi(\mathbf{x}_i, \check{y}_i) - \phi(\mathbf{x}_i, y_i)\right)\right] \right]$$

sample-wise conditional distributions

$\mathcal{P}(\hat{y}_i|\mathbf{x}_i)$
Size: 2 (binary)

Dual | Non-decomposable metric:

F1-score non-decomposable loss metric

$$\max_\theta \left[\min_{\mathcal{Q}(\check{\mathbf{y}}|\mathbf{x})} \max_{\mathcal{P}(\hat{\mathbf{y}}|\mathbf{x})} \sum_{\hat{\mathbf{y}}, \check{\mathbf{y}}} \mathcal{P}(\hat{\mathbf{y}}|\mathbf{x})\mathcal{Q}(\check{\mathbf{y}}|\mathbf{x}) \left(\frac{2\sum_i \hat{y}_i \check{y}_i}{\sum_i \hat{y}_i + \sum_i \check{y}_i} - \theta^\intercal \sum_i^n \left[\phi(\mathbf{x}, \check{y}_i) - \phi(\mathbf{x}, y_i)\right] \right) \right]$$

Full training set conditional distributions

$\mathcal{P}(\hat{\mathbf{y}}|\mathbf{x})$
Size: 2^n

Marginalization technique: optimize over marginalization distribution instead:

Original: $\mathcal{P}(\hat{\mathbf{y}}|\mathbf{x})$ Size: 2^n Intractable! → Marginalization: $\mathcal{P}(\hat{y}_i = 1, \sum_i \hat{y}_i = k|\mathbf{x})$ Size: n^2 Tractable!

Generic Non-Decomposable Performance Metrics

(Fathony & Kolter, AISTATS 2020)

More complex performance metric

$$\text{metric}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_j \frac{a_j \mathsf{TP} + b_j \mathsf{TN} + f_j(\mathsf{PP}, \mathsf{AP})}{g_j(\mathsf{PP}, \mathsf{AP})}$$

		Actual: Positive	Actual: Negative	
Pred.	Positive	True Pos. (TP)	False Pos. (FP)	Predicted Pos. (PP)
Pred.	Negative	False Neg. (FN)	True Neg. (TN)	Predicted Neg. (PN)
		Actual Pos. (AP)	Actual Neg. (AN)	All Data (ALL)

Cover a vast range of performance metric families

Including most common use cases of non-decomposable metrics: Precision, Recall, F_β-score, Balanced Accuracy, Specificity, Sensitivity, Informednes, Markedness, MCC, Kappa score, etc…

Practitioners can define their novel custom metrics

Metrics that specifically targeted to their novel problems.

Dual | Marginalization technique:

Original: $\mathcal{P}(\hat{\mathbf{y}}|\mathbf{x})$ Size: 2^n Intractable! → Marginalization: $\mathcal{P}(\hat{y}_i = 1, \sum_i \hat{y}_i = k|\mathbf{x})$ & $\mathcal{P}(\hat{y}_i = 0, \sum_i \hat{y}_i = k)$ Size: $2n^2$ Tractable!

Integration with Machine Learning Pipeline

(Fathony & Kolter, AISTATS 2020)

F_β score definition

$$\frac{(1+\beta^2)\ \text{TP}}{\beta^2\ \text{AP} + \text{PP}}$$

Programming Interface for Practitioners

Easily incorporate custom performance metric into ML pipeline

```
model = Chain(
  Dense(nvar, 100, relu),
  Dense(100, 100, relu),
  Dense(100, 1), vec)

objective(x, y) = mean(
  logitbinarycrossentropy(model(x), y))

opt = ADAM(1e-3)
Flux.train!(objective, params(model),
  train_set, opt)
```

Leaning using binary cross entropy

```
model = Chain(
  Dense(nvar, 100, relu),
  Dense(100, 100, relu),
  Dense(100, 1), vec)

@metric FBeta beta
function define(::Type{FBeta}, C::ConfusionMatrix, beta)
    return ((1 + beta^2) * C.tp) / (beta^2 * C.ap + C.pp)
end
f2_score = FBeta(2)
special_case_positive!(f2_score)

objective(x, y) = ap_objective(model(x), y, f2_score)

opt = ADAM(1e-3)
Flux.train!(objective, params(model), train_set, opt)
```

Leaning using AP formulation for F2-metric

*) The codes are written in Julia

AP-Perf: Supports a wide variety of evaluation metrics

(Fathony & Kolter, AISTATS 2020)

Code examples for other performance metrics:

Geometric Mean of Precision and Recall (GPR)

$$\frac{TP}{\sqrt{PP \cdot AP}}$$

```
@metric GM_PrecRec      # Geometric Mean of Prec and Rec
function define(::Type{GM_PrecRec}, C::ConfusionMatrix)
    return C.tp / sqrt(C.ap * C.pp)
end
gpr = GM_PrecRec()
special_case_positive!(gpr)
```

Cohen's Kappa score

$$\frac{(TP + TN)/ALL - (AP \cdot PP + AN \cdot PN)/ALL^2}{1 - (AP \cdot PP + AN \cdot PN)/ALL^2}$$

```
@metric Kappa
function define(::Type{Kappa}, C::ConfusionMatrix)
    pe = (C.ap * C.pp + C.an * C.pn) / C.all^2
    num = (C.tp + C.tn) / C.all - pe
    den = 1 - pe
    return num / den
end
kappa = Kappa()
special_case_positive!(kappa)
special_case_negative!(kappa)
```

*) The codes are written in Julia

Novel Custom Metrics

Write-your-own Novel Metrics

```
@metric NovelMetric
function define(::Type{NovelMetric}, C::ConfusionMatrix)
  # write the definition of your new metric
end
novel_metric = NovelMetric()
```

Example:

a weighted modification to the Cohen's Kappa score and the Mathews correlation coefficient (MCC)

$$0.3 \cdot \frac{(0.7\,\text{TP} + 0.3\,\text{TN})/\text{ALL} - (0.7 \cdot \text{AP} \cdot \text{PP} + 0.3 \cdot \text{AN} \cdot \text{PN})/\text{ALL}^2}{1 - (0.7 \cdot \text{AP} \cdot \text{PP} + 0.3 \cdot \text{AN} \cdot \text{PN})/\text{ALL}^2} + 0.7 \cdot \frac{\text{TP}/\text{ALL} - (\text{AP} \cdot \text{PP})/\text{ALL}^2}{\sqrt{\text{AP} \cdot \text{PP} \cdot \text{AN} \cdot \text{PN}}/\text{ALL}^2}$$

```
@metric NovelMetric
function define(::Type{NovelMetric}, C::ConfusionMatrix)
  pe = (0.7 * C.ap * C.pp + 0.3 * C.an * C.pn) / C.all^2
  num = (0.7 * C.tp + 0.3 * C.tn) / C.all - pe
  den = 1 - pe
  kappa = num / den

  num2 = C.tp / C.all - (C.ap * C.pp) / C.all^2
  den2 = sqrt(C.ap * C.pp * C.an * C.pn) / C.all^2
  mcc = num2 / den2

  return 0.3 * kappa + 0.7 * mcc
end

novel_metric = NovelMetric()
special_case_positive!(novel_metric)
special_case_negative!(novel_metric)
```

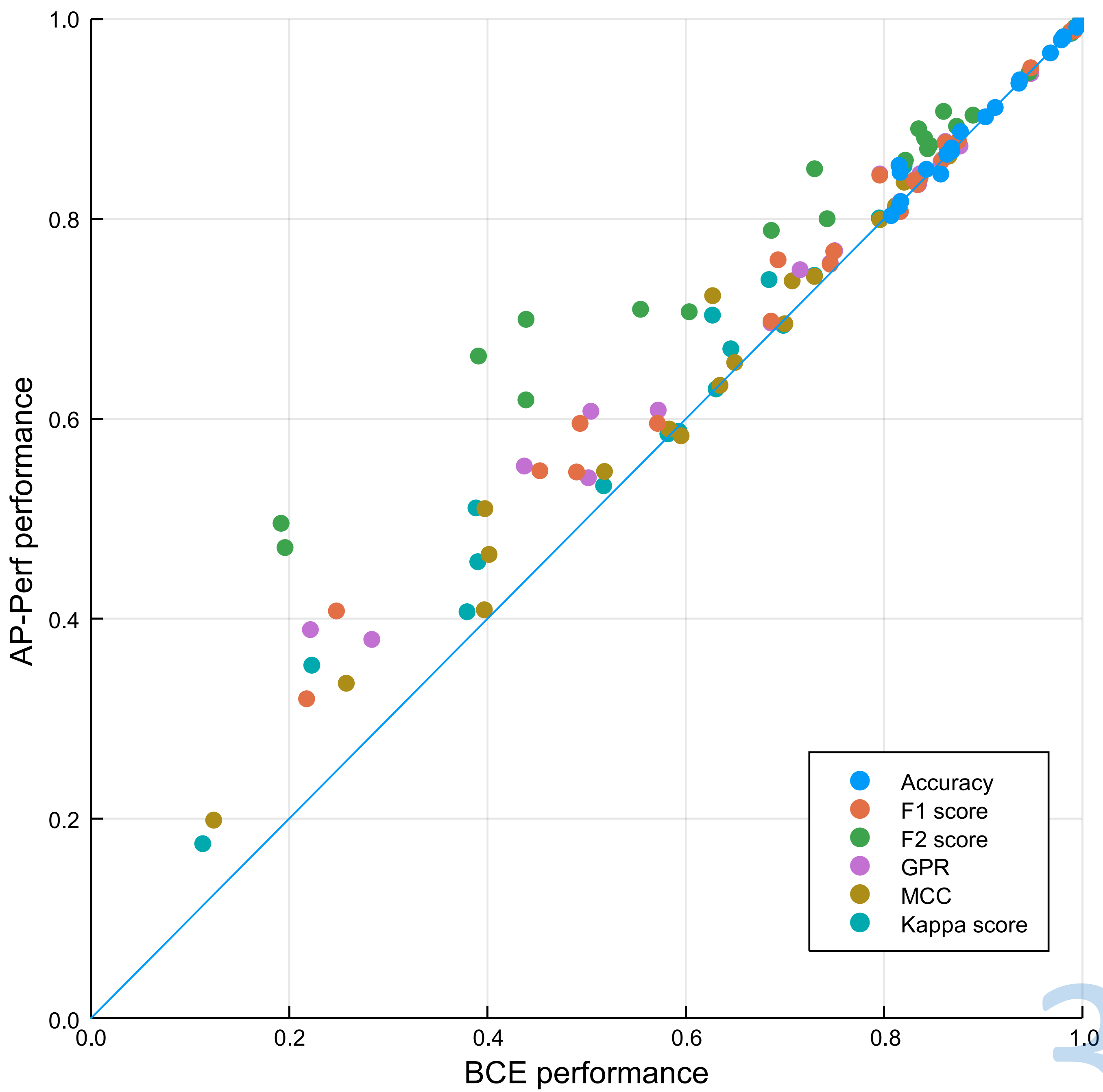
*) The codes are written in Julia

Empirical Results

Datasets:
20 UCI Datasets,
MNIST, Fashion MNIST

Neural Networks:
Multi Layer Perceptron,
Convolutional NN

Performance Metrics:
1) Accuracy
2) F1 score
3) F2 score
4) Geom. Prec. Rec. (GPR)
5) Mathews Cor. Coef. (MCC)
6) Cohen's Kappa score



Summary | Non-decomposable Metrics

	Statistical Consistency	**Support Neural Network Learning**	**Support Custom Metrics**	**Easy Interface for Practitioners** (to optimize custom metrics)
SVM-Perf (Joachim, 2005)	✗	✗	✓	✗
Plug-in based classifiers (Koyejo et al, 2014; Narashiman et al, 2014)	✓	✗	✗	✗
Global objectives (Eban et al, 2014)	✗	✓	✗	✗
DAME & DUPLE (Sanyal et al, 2018)	✗	✓	✗	✗
Adversarial Prediction (Fathony & Kolter, AISTATS 2020)	✓	✓	✓	✓

Other Machine Learning Areas

Other Machine Learning Areas

Conditional Graphical Models

(Fathony et.al., NeurIPS 2018)

Application examples:
character recognition, activity recognition, part-of-speech tagging

Adversarial prediction benefits:
- aligns with the evaluation metrics (vs CRF)
- provides statistical consistency (vs SSVM)

Overall empirical performance:
better than CRF and Structured SVM

Bipartite Matching in Graphs

(Fathony et.al., ICML 2018)

Application examples:
word alignment in translation, object tracking in video, documents ranking

Adversarial prediction benefits:
- computationally efficient (vs CRF)
- provides statistical consistency (vs SSVM)

Overall empirical performance:
better than the Structured SVM
(the CRF is intractable in our experiment setup)

Fairness in ML

(Rezaei*, Fathony*, et.al., AAAI 2020)
* equal contributors

Fairness formulation for the robust log-los classifier (logistic regression)

Benefits: convex, unique solution, single predictor, good performance, faster runtime

Summary and Potentials

Benefits and Challenges

Adversarial Prediction vs ERM Framework

Benefits

1. **No need to think about surrogate loss**
   Adversarial prediction formulation can work directly on the original metrics

2. **Accepts most evaluation metrics**
   Including continuous and discrete metrics

3. **Facilitates writing custom metrics**
   Enables practitioners to write novel custom metrics specifically tailored for the problem

4. **Good performances in theory and practice**
   Provides statistical consistency guarantee and performs competitively in practice.

Challenges

1. **Solving the formulation**
   Solving the adversarial prediction formulation efficiently for specific metric may require clever techniques, e.g. marginalization technique

2. **Running time**
   Current runtime is noticeably slower than optimizing the cross-entropy objective. Improvement is needed to solve the resulting dual formulation.

Potential

Adversarial Prediction + Programming Interface for Custom Metrics

Potential:
Reshaping the culture of the practitioners in applied machine learning

Now:

- Choose an evaluation metric from a popular list of metrics
- Pick a model that optimizes something else

Future:

- Design a custom metric that align specifically with the application goal
- Run a model that optimizes the designed metric

Bringing

Evaluation metric + training model

in harmony

Goal-Oriented Learning

Choose an evaluation metric

Choose a model

Train the model

References

- **| UAI 2015 | Adversarial Cost-Sensitive Classification**
  Kaiser Asif, Wei Xing, Sima Behpour, and Brian D. Ziebart.
- **| NeurIPS 2016 | Adversarial Multiclass Classification: A Risk Minimization Perspective**
  Rizal Fathony, Anqi Liu, Kaiser Asif, Brian D. Ziebart.
- **| NeurIPS 2017 | Adversarial Surrogate Losses for Ordinal Regression**
  Rizal Fathony, Mohammad Bashiri, Brian D. Ziebart.
- **| ArXiV 2018 | Consistent Robust Adversarial Prediction for General Multiclass Classification**
  Rizal Fathony, Kaiser Asif, Anqi Liu, Mohammad Bashiri, Wei Xing, Sima Behpour, Xinhua Zhang, Brian D. Ziebart.
- **| NeurIPS 2018 | Distributionally Robust Graphical Models**
  Rizal Fathony, Ashkan Rezaei, Mohammad Ali Bashiri, Xinhua Zhang, and Brian Ziebart
- **| ICML 2018 | Efficient and Consistent Adversarial Bipartite Matching**
  Rizal Fathony*, Sima Behpour*, Xinhua Zhang, and Brian Ziebart
- **| AAAI 2020 | Fairness for Robust Log Loss Classification**
  Ashkan Rezaei*, Rizal Fathony*, Omid Memarrast, and Brian Ziebart
- **| AISTATS 2020 | AP-Perf: Incorporating Generic Performance Metrics in Differentiable Learning**
  Rizal Fathony and Zico Kolter

Thank You