Omnipredictors[1]: One Predictor to Rule Them All

Heavily adapted from P. Gopalan's Talk at IAS

J. Setpal

April 18, 2024

[1] Gopalan, Kalai, Reingold, Sharan, Wieder

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

How do we mathematically encode $\hat{y} \approx y$?

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

How do we mathematically encode $\hat{y} \approx y$? A loss (distance) function!

4. Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; $L(\hat{y}, y) \approx 0$ iff $\hat{y} \approx y$; L is continuous.

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

How do we mathematically encode $\hat{y} \approx y$? A loss (distance) function!

4. Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; $L(\hat{y}, y) \approx 0$ iff $\hat{y} \approx y$; L is continuous.

How can we update our weights to optimize against this loss function?

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

How do we mathematically encode $\hat{y} \approx y$? A loss (distance) function!

4. Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; $L(\hat{y}, y) \approx 0$ iff $\hat{y} \approx y$; L is continuous.

How can we update our weights to optimize against this loss function?

5. Gradient Descent! $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}}$

Iterate (5) until convergence.

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

How do we mathematically encode $\hat{y} \approx y$? A loss (distance) function!

4. Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; $L(\hat{y}, y) \approx 0$ iff $\hat{y} \approx y$; L is continuous.

How can we update our weights to optimize against this loss function?

5. Gradient Descent! $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}}$

Iterate (5) until convergence.

L is minimized over $\mathcal{D}$, not over the real world.

Supervised Learning Synopsis

We'll start with an *overview* of supervised learning paradigm:

1. Dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{N}$; $N \ll \infty$; $\mathcal{D} \sim$ "Real World"
2. Parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$
3. Objective: Train $\boldsymbol{\theta}$ s.t. $f_{\boldsymbol{\theta}}(x) = \hat{y} \approx y$

How do we mathematically encode $\hat{y} \approx y$? A loss (distance) function!

4. Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; $L(\hat{y}, y) \approx 0$ iff $\hat{y} \approx y$; L is continuous.

How can we update our weights to optimize against this loss function?

5. Gradient Descent! $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}}$

Iterate (5) until convergence.

L is minimized over $\mathcal{D}$, not over the real world. This is **empirical risk**:

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} L(f_{\boldsymbol{\theta}}(x_i), y_i) \tag{1}$$

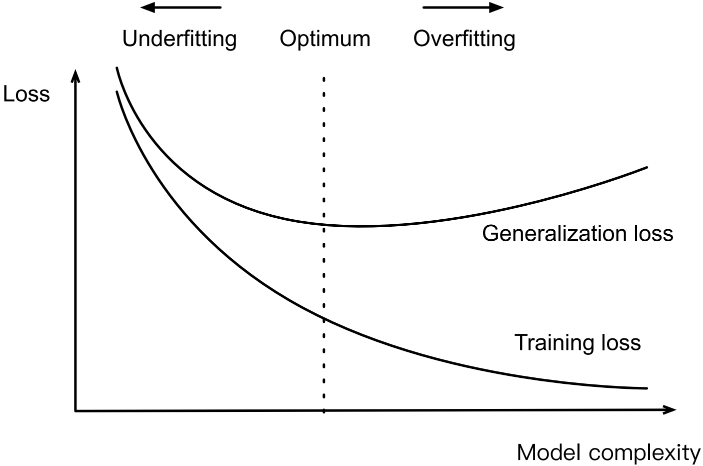
Generalization Error

We also split $\mathcal{D}$ into training, validation, test splits to minimize overfitting.

Generalization Error

We also split $\mathcal{D}$ into training, validation, test splits to minimize overfitting.

Usually $L_{valid} \not\approx L_{train}$ after training. That's our generalization gap.



Challenge Statement

Problem: Different loss functions typically have divergent geometries.

[2] usually, local

Challenge Statement

Problem: Different loss functions typically have divergent geometries.

This means gradient descent obtains a **different**[2] **optima** for two such functions, despite sharing minima for $\boldsymbol{\theta}$ s.t. $\hat{y} \approx y$.

[2] usually, local

Challenge Statement

Problem: Different loss functions typically have divergent geometries.

This means gradient descent obtains a **different**[2] **optima** for two such functions, despite sharing minima for $\boldsymbol{\theta}$ s.t. $\hat{y} \approx y$.

Let's evaluate this empirically on ℓ_1 and ℓ_2 losses, which optimize for median and mean respectively:

$$\ell_1 = |y - \hat{y}|,\ \ell_2 = (y - \hat{y})^2 \tag{2}$$

$$x \sim f(\epsilon \sim \mathcal{U}[0,1]) := \begin{cases} 0 & \epsilon \leq 0.4 \\ \mathcal{U}[0.8,1] & \text{otherwise} \end{cases} \tag{3}$$

[2] usually, local

Challenge Statement

Problem: Different loss functions typically have divergent geometries.

This means gradient descent obtains a **different**[2] **optima** for two such functions, despite sharing minima for $\boldsymbol{\theta}$ s.t. $\hat{y} \approx y$.

Let's evaluate this empirically on ℓ_1 and ℓ_2 losses, which optimize for median and mean respectively:

$$\ell_1 = |y - \hat{y}|,\ \ell_2 = (y - \hat{y})^2 \tag{2}$$

$$x \sim f(\epsilon \sim \mathcal{U}[0,1]) := \begin{cases} 0 & \epsilon \leq 0.4 \\ \mathcal{U}[0.8, 1] & \text{otherwise} \end{cases} \tag{3}$$

Omnipredictors provides a framework for rigorous guarantees, deriving $\tilde{p} \approx p^*$: a predictor that is able to *simultaneously minimize* a family of convex loss functions.

[2] usually, local

Multigroup Fairness

We can split $\mathcal{D}$ into various *subgroups* based on **shared characteristics**. These can be explicit or implicit (i.e. subgroups we don't know of):

	Group-1	**Group-2**	**Group-3**	**Group-4**
Accuracy	0.9593	0.6249	0.3157	0.2664
Loss	0.0021	0.4102	1.3457	1.7664
Proportion	0.9	0.08	0.0075	0.0025

Multigroup Fairness

We can split $\mathcal{D}$ into various *subgroups* based on **shared characteristics**. These can be explicit or implicit (i.e. subgroups we don't know of):

	Group-1	**Group-2**	**Group-3**	**Group-4**
Accuracy	0.9593	0.6249	0.3157	0.2664
Loss	0.0021	0.4102	1.3457	1.7664
Proportion	0.9	0.08	0.0075	0.0025

Empirical Risk is *only* 0.0492, but inference is unreliable for subgroups 2-4.

Multigroup Fairness

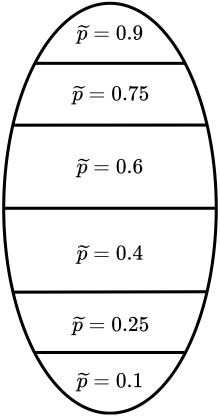
We can split $\mathcal{D}$ into various *subgroups* based on **shared characteristics**. These can be explicit or implicit (i.e. subgroups we don't know of):

	Group-1	**Group-2**	**Group-3**	**Group-4**
Accuracy	0.9593	0.6249	0.3157	0.2664
Loss	0.0021	0.4102	1.3457	1.7664
Proportion	0.9	0.08	0.0075	0.0025

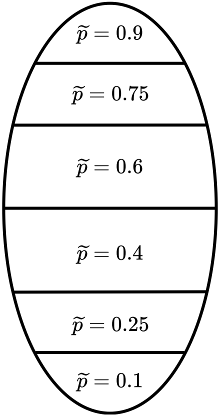
Empirical Risk is *only* 0.0492, but inference is unreliable for subgroups 2-4.

One notion of fairness stipulates equal risk for every subgroup. However, finding subgroups is hard for high-dimensional data.

Multiaccuracy & Multigroup Fairness

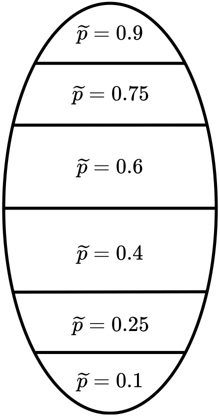


Multiaccuracy & Multigroup Fairness



Let C be the collection of subsets. We probe it further for correlations.

Multiaccuracy & Multigroup Fairness

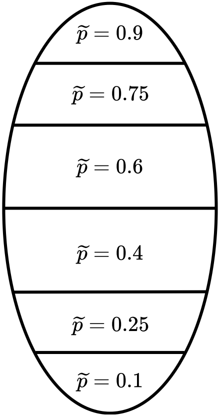


Let C be the collection of subsets. We probe it further for correlations.

$\tilde{p}$ is (C, α)-multiaccurate if:

$$\max_{c \in C} |\mathbb{E}[c(x)(y - \tilde{p}(x))]| \leq \alpha \tag{4}$$

Multiaccuracy & Multigroup Fairness



Let C be the collection of subsets. We probe it further for correlations.

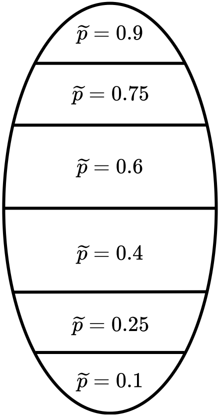
$\tilde{p}$ is (C, α)-multiaccurate if:

$$\max_{c \in C} |\mathbb{E}[c(x)(y - \tilde{p}(x))]| \leq \alpha \tag{4}$$

$\tilde{p}$ is (C, α)-multicalibrated if:

$$\max_{c \in C} E[|\mathbb{E}[c(x)(y - \tilde{p}(x))]|] \leq \alpha \tag{5}$$

Multiaccuracy & Multigroup Fairness



Let C be the collection of subsets. We probe it further for correlations.

$\tilde{p}$ is (C, α)-multiaccurate if:

$$\max_{c \in C} |\mathbb{E}[c(x)(y - \tilde{p}(x))]| \leq \alpha \tag{4}$$

$\tilde{p}$ is (C, α)-multicalibrated if:

$$\max_{c \in C} E[|\mathbb{E}[c(x)(y - \tilde{p}(x))]|] \leq \alpha \tag{5}$$

If we can find correlation with the error, there's some advantage to be gained. We minimize this to train a **weak agnostic learner**.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*).$ We denote optimal action $t := k^*_\ell \circ p^*$

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*)$. We denote optimal action $t := k_\ell^* \circ p^*$

This paper connects multigroup fairness with the notion of a weak agnostic learner, to formulate (L, C)-omnipredictors.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*)$. We denote optimal action $t := k_\ell^* \circ p^*$

This paper connects multigroup fairness with the notion of a weak agnostic learner, to formulate (L, C)-omnipredictors. Specifically, it trains $g_{\boldsymbol{\theta}} \approx p^*$.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*)$. We denote optimal action $t := k_\ell^* \circ p^*$

This paper connects multigroup fairness with the notion of a weak agnostic learner, to formulate (L, C)-omnipredictors. Specifically, it trains $g_{\boldsymbol{\theta}} \approx p^*$.

Intuitively: the idea is to *extract the predictive power* of the data.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*)$. We denote optimal action $t := k_\ell^* \circ p^*$

This paper connects multigroup fairness with the notion of a weak agnostic learner, to formulate (L, C)-omnipredictors. Specifically, it trains $g_{\boldsymbol{\theta}} \approx p^*$.

Intuitively: the idea is to *extract the predictive power* of the data.

Let L_{cvx} be a set of Lipschitz, convex, bounded losses.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*)$. We denote optimal action $t := k_\ell^* \circ p^*$

This paper connects multigroup fairness with the notion of a weak agnostic learner, to formulate (L, C)-omnipredictors. Specifically, it trains $g_{\boldsymbol{\theta}} \approx p^*$.

Intuitively: the idea is to *extract the predictive power* of the data.

Let L_{cvx} be a set of Lipschitz, convex, bounded losses. If $f_{\boldsymbol{\theta}}$ is C-multicalibrated with some error α, it is an (L_{cvx}, C, α)-omnipredictor.

Omnipredictors

If we know p^*, it is easy for us take take the optimal action.

For $y \in \{0, 1\},\ y \sim \text{Bernoulli}(p^*)$. We denote optimal action $t := k_\ell^* \circ p^*$

This paper connects multigroup fairness with the notion of a weak agnostic learner, to formulate (L, C)-omnipredictors. Specifically, it trains $g_{\boldsymbol{\theta}} \approx p^*$.

Intuitively: the idea is to *extract the predictive power* of the data.

Let L_{cvx} be a set of Lipschitz, convex, bounded losses. If $f_{\boldsymbol{\theta}}$ is C-multicalibrated with some error α, it is an (L_{cvx}, C, α)-omnipredictor.

Multicalibration implies omniprediction for all convex loss functions.

Training Agnostic Predictors

We can use this framework to train a predictor s.t. a new model trained just on <u>one loss function</u> performs equivalently to the omnipredictor.

$$C = \{c : \mathcal{X} \to \mathcal{Y}\} \tag{6}$$

$$w : (0,1] \to (0,1] \ s.t. \ w(\alpha) \le \alpha \tag{7}$$

Training Agnostic Predictors

We can use this framework to train a predictor s.t. a new model trained just on <u>one loss function</u> performs equivalently to the omnipredictor.

$$C = \{c : \mathcal{X} \to \mathcal{Y}\} \tag{6}$$
$$w : (0, 1] \to (0, 1] \; s.t. \; w(\alpha) \leq \alpha \tag{7}$$

We then train with the following objective:

$$\min_{\boldsymbol{\theta}} \mathbf{Cov}_{\mathcal{D}}[c(x), y] \tag{8}$$

Then, with probability $1 - \delta$ the weak learner returns c s.t. $\mathbf{Cov}_{\mathcal{D}}[c(x), y] \geq w(\alpha)$.

Training Agnostic Predictors

We can use this framework to train a predictor s.t. a new model trained just on <u>one loss function</u> performs equivalently to the omnipredictor.

$$C = \{c : \mathcal{X} \to \mathcal{Y}\} \tag{6}$$

$$w : (0,1] \to (0,1] \; s.t. \; w(\alpha) \leq \alpha \tag{7}$$

We then train with the following objective:

$$\min_{\boldsymbol{\theta}} \mathbf{Cov}_{\mathcal{D}}[c(x), y] \tag{8}$$

Then, with probability $1 - \delta$ the weak learner returns c s.t. $\mathbf{Cov}_{\mathcal{D}}[c(x), y] \geq w(\alpha)$.

We use this to compute an α-multicalibrated partition by a layered branching program that runs in $\mathcal{O}(\frac{l}{w(\alpha/2)})^{\mathcal{O}(l)}$.

Thank you!

Have an awesome rest of your day!

Slides:

`https://cs.purdue.edu/homes/jsetpal/slides/omnipredictors.pdf`