Out-of-Distribution Generalization via Risk Extrapolation

"learning to define a cow"

J. Setpal

March 28, 2023

Outline

Outline

Introduction

Training a neural network can be thought of as **modelling a multivariate distribution**; i.e. $p(y|x)$ where y is the expected output, and x is the training data.

Introduction

Training a neural network can be thought of as **modelling a multivariate distribution**; i.e. $p(y|x)$ where y is the expected output, and x is the training data.

The objective is to maximize likelihood:

$$\mathcal{L}(\mathbf{W}, \{(x_i, y_i)\}_{i=1}^{N}) = \underset{\mathbf{W}}{\operatorname{argmax}} \prod_{i=1}^{N} p(\mathbf{y}_i | \mathbf{x}_i; \mathbf{W})$$

Introduction

Training a neural network can be thought of as **modelling a multivariate distribution**; i.e. $p(y|x)$ where y is the expected output, and x is the training data.

The objective is to maximize likelihood:

$$\mathcal{L}(\mathbf{W}, \{(x_i, y_i)\}_{i=1}^{N}) = \underset{\mathbf{W}}{\operatorname{argmax}} \prod_{i=1}^{N} p(\mathbf{y}_i | \mathbf{x}_i; \mathbf{W})$$

This is almost *too powerful*!

Introduction

Training a neural network can be thought of as **modelling a multivariate distribution**; i.e. $p(y|x)$ where y is the expected output, and x is the training data.

The objective is to maximize likelihood:

$$\mathcal{L}(\mathbf{W}, \{(x_i, y_i)\}_{i=1}^{N}) = \underset{\mathbf{W}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{y}_i | \mathbf{x}_i; \mathbf{W})$$

This is almost *too powerful*! Most models are overparameterized, and maximum likelihood does not care about the causal basis for the data.

Worst-Group Performance

This results in the network fitting to **spurious correlations** in the dataset.

Worst-Group Performance

This results in the network fitting to **spurious correlations** in the dataset.

We can specify potential spurious correlations as **groups**, and individually observe the performance of those groups.

Worst-Group Performance

This results in the network fitting to **spurious correlations** in the dataset.

We can specify potential spurious correlations as **groups**, and individually observe the performance of those groups.

	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

While the overall accuracy on the training set is maximized, it assumes that Group-1 is going to be 90% of the test set.

Worst-Group Performance

This results in the network fitting to **spurious correlations** in the dataset.

We can specify potential spurious correlations as **groups**, and individually observe the performance of those groups.

	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

While the overall accuracy on the training set is maximized, it assumes that Group-1 is going to be 90% of the test set.

The prediction on an input from Group-4 is too inaccurate to be considered.

Domain Robustness

To correct this, our new objective is to ensure *domain robustness*: our model must generalize to a new, unseen **test domain**.

Domain Robustness

To correct this, our new objective is to ensure *domain robustness*: our model must generalize to a new, unseen **test domain**.

We can formalize this as a **pertubation set** of risk:

$$\mathcal{R}_{\mathcal{F}}^{\mathsf{OOD}}(\theta) = \max_{e \in \mathcal{F}} \mathcal{R}_e(\theta)$$

where $\mathcal{F}$ is the set of possible test domains, & θ is our predictor.

Domain Robustness

To correct this, our new objective is to ensure *domain robustness*: our model must generalize to a new, unseen **test domain**.

We can formalize this as a **pertubation set** of risk:

$$\mathcal{R}_{\mathcal{F}}^{\mathsf{OOD}}(\theta) = \max_{e \in \mathcal{F}} \mathcal{R}_e(\theta)$$

where $\mathcal{F}$ is the set of possible test domains, & θ is our predictor.

The authors state that $\mathcal{F}$ cannot be arbitrary, and we are restricted our assumptions of the possible test domains.

Domain Robustness

To correct this, our new objective is to ensure *domain robustness*: our model must generalize to a new, unseen **test domain**.

We can formalize this as a **pertubation set** of risk:

$$\mathcal{R}_{\mathcal{F}}^{\mathsf{OOD}}(\theta) = \max_{e \in \mathcal{F}} \mathcal{R}_e(\theta)$$

where $\mathcal{F}$ is the set of possible test domains, & θ is our predictor.

The authors state that $\mathcal{F}$ cannot be arbitrary, and we are restricted our assumptions of the possible test domains.

Risk Extrapolation uncovers **invariant relationships** between the input and outputs.

Domain Robustness

To correct this, our new objective is to ensure *domain robustness*: our model must generalize to a new, unseen **test domain**.

We can formalize this as a **pertubation set** of risk:

$$\mathcal{R}^{\mathsf{OOD}}_{\mathcal{F}}(\theta) = \max_{e \in \mathcal{F}} \mathcal{R}_e(\theta)$$

where $\mathcal{F}$ is the set of possible test domains, & θ is our predictor.

The authors state that $\mathcal{F}$ cannot be arbitrary, and we are restricted our assumptions of the possible test domains.

Risk Extrapolation uncovers **invariant relationships** between the input and outputs. A model that bases predictions on an invariant relationship is an **invariant predictor**.

Outline

Empirical Risk Minimization (ERM)

One way to correct the poor worst-group performance is simply **importance re-weighting**.

Empirical Risk Minimization (ERM)

One way to correct the poor worst-group performance is simply **importance re-weighting**.

Instead of traditional likelihood maximization, ERM minimizes the average loss <u>across domains</u>:

$$\mathcal{J}_{\mathsf{ERM}}(\theta) \doteq \operatorname*{argmin}_{\theta} \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \ell(x_i, y_i; \theta)$$

Empirical Risk Minimization (ERM)

One way to correct the poor worst-group performance is simply **importance re-weighting**.

Instead of traditional likelihood maximization, ERM minimizes the average loss across domains:

$$\mathcal{J}_{\mathrm{ERM}}(\theta) \doteq \underset{\theta}{\operatorname{argmin}} \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \ell(x_i, y_i; \theta)$$

Just Train Twice (Liu, et.al; 2021) presents an interesting approach that obtains group information by multi-stage training, re-weighting the cost and subsequently re-training.

Invariant Risk Minimization (IRM)

A more robust is to define **group-invariance**. The approach motivates the model to learn invariant relationships, building an invariant predictor.

Invariant Risk Minimization (IRM)

A more robust is to define **group-invariance**. The approach motivates the model to learn invariant relationships, building an invariant predictor.

They generate a bi-leveled optimization task, where the objectives are:

Invariant Risk Minimization (IRM)

A more robust is to define **group-invariance**. The approach motivates the model to learn invariant relationships, building an invariant predictor.

They generate a bi-leveled optimization task, where the objectives are: a) minimizing risk,

Invariant Risk Minimization (IRM)

A more robust is to define **group-invariance**. The approach motivates the model to learn invariant relationships, building an invariant predictor.

They generate a bi-leveled optimization task, where the objectives are: a) minimizing risk, and b) actually make relevant predictions.

Invariant Risk Minimization (IRM)

A more robust is to define **group-invariance**. The approach motivates the model to learn invariant relationships, building an invariant predictor.

They generate a bi-leveled optimization task, where the objectives are: a) minimizing risk, and b) actually make relevant predictions.

This is phrased as a penalized loss:

$$\mathcal{J}_{\mathrm{IRM}}(\theta, \mathcal{D}) \doteq \sum_{e \in \mathcal{E}} \mathcal{R}^e(\theta \circ \mathcal{D}) + \lambda \cdot \mathbb{D}(\theta, \mathcal{D}, e)$$

where $\lambda \in [0, \infty)$ is a hyper-parameter balancing prediction power and invariance, $\mathbb{D}$ represents loss-specific risk.

Distributionally Robust Optimization (DRO)

When only a a single domain (group) is available, it is common to assume $p(Y|X)$ is fixed.

Distributionally Robust Optimization (DRO)

When only a a single domain (group) is available, it is common to assume $p(Y|X)$ is fixed. This is called the **covariate shift assumption**. REx does not make this assumption.

Distributionally Robust Optimization (DRO)

When only a a single domain (group) is available, it is common to assume $p(Y|X)$ is fixed. This is called the **covariate shift assumption**. REx does not make this assumption.

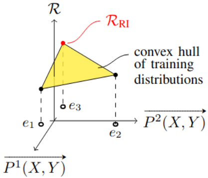
For data containing multiple domains, test distributions are assumed to be **convex combinations** of the training distribution.

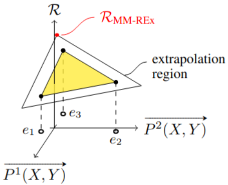
Distributionally Robust Optimization (DRO)

When only a a single domain (group) is available, it is common to assume $p(Y|X)$ is fixed. This is called the **covariate shift assumption**. REx does not make this assumption.

For data containing multiple domains, test distributions are assumed to be **convex combinations** of the training distribution. This is equivalent to setting $\mathcal{F} \doteq \mathcal{E}$:

$$\mathcal{R}_{\text{RI}}(\theta) \doteq \max_{\sum_e \lambda_e = 1,\ \lambda_e \geq 0} \sum_{e=1}^{m} \lambda_e \mathcal{R}_e(\theta) = \max_{e \in \mathcal{E}} \mathcal{R}_e(\theta)$$





Outline

Minimax Risk Extrapolation (MM-REx)

Minimax-REx fundamentally *extrapolates* on DRO.

Minimax Risk Extrapolation (MM-REx)

Minimax-REx fundamentally *extrapolates* on DRO. By setting $\lambda_e \geq \lambda_{\text{min}}$:

$$\mathcal{R}_{\text{MM-REx}}(\theta) \doteq \max_{\sum_e \lambda_e = 1,\ \lambda_e \geq \lambda_{\text{min}}} \sum_{e=1}^{m} \lambda_e \mathcal{R}_e(\theta)$$
$$\doteq (1 - m\lambda_{\text{min}}) \max_e \mathcal{R}_e(\theta) + \lambda_{\text{min}} \sum_{e=1}^{m} \mathcal{R}_e(\theta)$$

where m is the number of domains, λ_{min} defines the degree of extrapolation.

Minimax Risk Extrapolation (MM-REx)

Minimax-REx fundamentally *extrapolates* on DRO. By setting $\lambda_e \geq \lambda_{\text{min}}$:

$$\begin{aligned}\mathcal{R}_{\text{MM-REx}}(\theta) &\doteq \max_{\sum_e \lambda_e = 1,\ \lambda_e \geq \lambda_{\text{min}}} \sum_{e=1}^{m} \lambda_e \mathcal{R}_e(\theta) \\ &\doteq (1 - m\lambda_{\text{min}}) \max_e \mathcal{R}_e(\theta) + \lambda_{\text{min}} \sum_{e=1}^{m} \mathcal{R}_e(\theta)\end{aligned}$$

where m is the number of domains, λ_{min} defines the degree of extrapolation.

The updated risk function extrapolates on convex combinations defined earlier.

Minimax Risk Extrapolation (MM-REx)

Minimax-REx fundamentally *extrapolates* on DRO. By setting $\lambda_e \geq \lambda_{\text{min}}$:

$$
\begin{aligned}
\mathcal{R}_{\text{MM-REx}}(\theta) &\doteq \max_{\sum_e \lambda_e = 1,\ \lambda_e \geq \lambda_{\text{min}}} \sum_{e=1}^{m} \lambda_e \mathcal{R}_e(\theta) \\
&\doteq (1 - m\lambda_{\text{min}}) \max_e \mathcal{R}_e(\theta) + \lambda_{\text{min}} \sum_{e=1}^{m} \mathcal{R}_e(\theta)
\end{aligned}
$$

where m is the number of domains, λ_{min} defines the degree of extrapolation.

The updated risk function extrapolates on convex combinations defined earlier.

If $\lambda_{\text{min}} < 0$, MM-REx sets negative weights to all but the worst-performing group.

Minimax Risk Extrapolation (MM-REx)

Minimax-REx fundamentally *extrapolates* on DRO. By setting $\lambda_e \geq \lambda_{\text{min}}$:

$$\mathcal{R}_{\text{MM-REx}}(\theta) \doteq \max_{\sum_e \lambda_e = 1,\ \lambda_e \geq \lambda_{\text{min}}} \sum_{e=1}^{m} \lambda_e \mathcal{R}_e(\theta)$$

$$\doteq (1 - m\lambda_{\text{min}}) \max_e \mathcal{R}_e(\theta) + \lambda_{\text{min}} \sum_{e=1}^{m} \mathcal{R}_e(\theta)$$

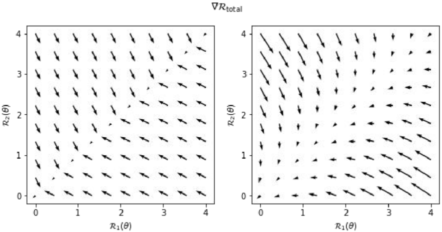
where m is the number of domains, λ_{min} defines the degree of extrapolation.

The updated risk function extrapolates on convex combinations defined earlier.

If $\lambda_{\text{min}} < 0$, MM-REx sets negative weights to all but the worst-performing group.
As $\lambda_{\text{min}} \to -\infty$, it enforces equality between training risks. This is proposed as a definition of **fairness**.

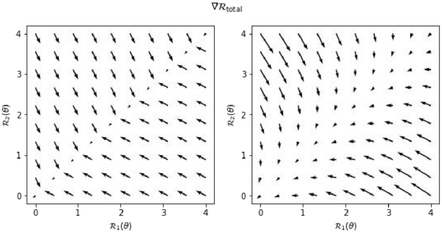
Variance Risk Extrapolation (V-REx)

While MM-REx defines the extrapolation procedure very cleanly, the resultant gradient (left) is extreme:



Variance Risk Extrapolation (V-REx)

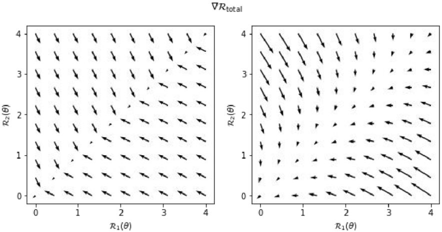
While MM-REx defines the extrapolation procedure very cleanly, the resultant gradient (left) is extreme:



Making it difficult to converge. Instead, using the **variances of the risks** (right) obtains smoother gradients, allowing for stabler optimization.

Variance Risk Extrapolation (V-REx)

While MM-REx defines the extrapolation procedure very cleanly, the resultant gradient (left) is extreme:



Making it difficult to converge. Instead, using the **variances of the risks** (right) obtains smoother gradients, allowing for stabler optimization.

Adding the variance-based risk regularizer, we obtain the following:

$$\mathcal{R}_{\text{V-REx}}(\theta) \doteq \beta\sigma^2(\{\mathcal{R}_i\}_{i=1}^m) + \sum_{e=1}^{m} \mathcal{R}_e(\theta)$$

where $\beta \in [0, \infty)$ & $\beta \to \infty$ motivates risk equality.

Outline

Results

Algorithm	ColoredMNIST	VLCS	PACS	OfficeHome
ERM	52.0 ± 0.1	77.4 ± 0.3	85.7 ± 0.5	67.5 ± 0.5
IRM	51.8 ± 0.1	78.1 ± 0.0	84.4 ± 1.1	66.6 ± 1.0
V-REx	52.1 ± 0.1	77.9 ± 0.5	85.8 ± 0.6	66.7 ± 0.5

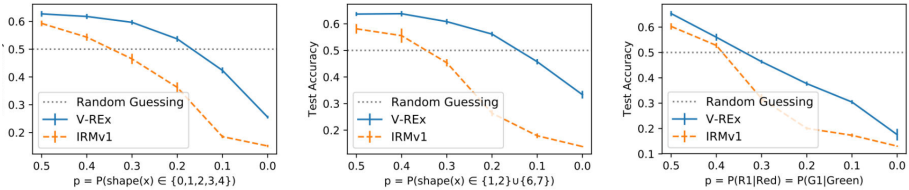
V-REx shows comparable performance on domain generalization benchmarks.

Results

Algorithm	ColoredMNIST	VLCS	PACS	OfficeHome
ERM	52.0 ± 0.1	77.4 ± 0.3	85.7 ± 0.5	67.5 ± 0.5
IRM	51.8 ± 0.1	78.1 ± 0.0	84.4 ± 1.1	66.6 ± 1.0
V-REx	52.1 ± 0.1	77.9 ± 0.5	85.8 ± 0.6	66.7 ± 0.5

V-REx shows comparable performance on domain generalization benchmarks.

However, when you include a *covariate shift*, V-REx **outperforms** IRM on dataset variants that include domain shift:



Thank you!

Have an awesome rest of your day!

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