## A. Proofs of Theorem 1

**Lemma 1.** *If $T$ is sufficient statistics, we have $p(Y, X|T) = p(Y|T) \cdot p(X|T)$.*

**Lemma 2.** *If $T$ is sufficient statistics, we have $p(Y|T(X)) = p(Y|X)$.*

*Proof.* Find $T'(X)$, s.t. $S(x) = \langle T(X), T'(X)\rangle$ is an invertible mapping of $X$, thus $p(Y|X) = p(Y|S(X)) = p(Y|T(X), T'(X))$. We have,

$$p(Y, T(X), T'(X)|T(X)) = p(Y|T'(X), T(X))p(T'(X)|T(X)) \tag{8}$$

From Lemma 1, we have

$$p(Y, T(X), T'(X)|T(X)) = p(Y|T(X))p(T'(X)|T(X)) \tag{9}$$

By (8) and (9), we obtain $p(Y|T'(X), T(X)) = p(Y|T(X)) = p(Y|X)$. □

**Theorem 1.** *Suppose that there is only covariate shift in $p_{train}$, i.e. $\exists x \in \mathcal{X}_{train}$ s.t. $p_{train}(x) \neq p_{ideal}(x)$ but $p_{train}(Y|X = x) = p_{ideal}(Y|X = x),\ \forall x \in \mathcal{X}_{train}$. Let $T_{train}(X)$ be the MSS representation learned under $p_{train}$, then we have:*

$$H_{train}(T_{train}(x)|T_{ideal}(x)) = 0. \tag{2}$$

*Proof.* Since there is covariate shift between $p_{\text{ideal}}$ and $p_{\text{train}}$, we have $p_{\text{train}}(Y|X) = p_{\text{ideal}}(Y|X), \forall x \in \mathcal{X}_{train}$. Since $T_{train}(X)$ is MSS of $p_{\text{train}}$ and by Lemma 2, we have $p_{\text{train}}(Y|T_{train}(X)) = p_{\text{train}}(Y|X) = p_{\text{ideal}}(Y|X) = p_{\text{ideal}}(Y|T_{ideal}(X)), \forall x \in \mathcal{X}_{train}$. Then $\forall x \in \mathcal{X}_{train}, y \in \mathcal{Y}$,

$$\begin{aligned}
p_{\text{train}}(y|T_{ideal}(x)) &= \sum_{x':T_{ideal}(x')=T_{ideal}(x)} p_{\text{train}}(y|x')p_{\text{train}}(x'|T(x)) \\
&= \sum_{x':T_{ideal}(x')=T_{ideal}(x)} p_{\text{ideal}}(y|x')p_{\text{train}}(x'|T(x)) \\
&= \sum_{x':T_{ideal}(x')=T_{ideal}(x)} p_{\text{ideal}}(y|T(x))p_{\text{train}}(x'|T(x)) \\
&= p_{\text{ideal}}(y|T_{ideal}(x))
\end{aligned} \tag{10}$$

Then we have

$$\begin{aligned}
H_{train}(Y|T_{train}(X)) &= \sum_{x,y} p_{\text{train}}(x, y)[-\log p_{\text{train}}(y|T_{train}(x))] \\
&= \sum_{x,y} p_{\text{train}}(x, y)[-\log p_{\text{ideal}}(y|T_{ideal}(x))] \\
&= \sum_{x,y} p_{\text{train}}(x, y)[-\log p_{\text{train}}(y|T_{ideal}(x))] \\
&= H_{train}(Y|T_{ideal}(X))
\end{aligned} \tag{11}$$

From equation 11 and the definition of sufficient statistics, we have

$$I_{train}(Y; T_{train}(X)) = I_{train}(Y; X) = I_{train}(Y; T_{ideal}(X)) \tag{12}$$

Thus, $T_{ideal}(X)$ is the sufficient statistics of $X$ about $Y$ under $p_{\text{train}}$. By definition, we have

$$H_{train}(T_{train}(X)|T_{ideal}(X)) = 0. \tag{13}$$

□

**Corollary 1.** *Suppose $\mathcal{X}_{train} = \mathcal{X}_{ideal}$ in Theorem 1, then $T_{train}(X)$ is also the MSS under $p_{ideal}$.*

*Proof.* Since $\mathcal{X}_{train} = \mathcal{X}_{ideal} = \mathcal{X}$, with the similar derivation of equation 10, we have $\forall x \in \mathcal{X}, y \in \mathcal{Y}$

$$p_{\text{ideal}}(y|T_{ideal}(x)) = p_{\text{ideal}}(y|T_{train}(x)) \tag{14}$$

Together with Theorem 1, we have $T_{train}(x)$ is also the MSS under $p_{\text{ideal}}$. □

## B. Connections between MLE and Learning Minimal Sufficient Statistics

### B.1. Information Bottleneck (IB) Method

The information bottleneck (IB) method (Tishby et al., 2000) is an information theoretic principle introduced to extract relevant information that an input $X \in \mathcal{X}$ contains about an output random variable $Y \in \mathcal{Y}$. Defined on a joint distribution of $X$ and $Y$, IB learns a mapping function $T(X)$ by optimizing the trade-off between the mutual information $I(X;T)$ and $I(Y;T)$ such that $T(X)$ is a compressed representation of $X$ (quantified by $I(X;T)$) that is most informative about $Y$ (quantified by $I(Y;T)$). Let $T$ be parameterized by $\theta$, the objective of IB optimizes the trade-off between $I(Y;T_\theta(X))$ and $I(X;T_\theta(X))$:

$$\min_{\theta} -I(Y,T_\theta(X)) + \beta I(X;T_\theta(X)) \tag{15}$$

where $\beta$ is a positive Lagrange multiplier.

Schwartz-Ziv & Tishby (2017) casts finding of minimal sufficient statistics (MSS) $T(X)$ as a constrained optimization problem using data-processing inequality (Cover, 1999):

$$\begin{aligned} &\min_{T(X)} \ \ I(T(X);X) \\ &s.t \ \ I(T(X);Y) = I(X;Y) \end{aligned} \tag{16}$$

This corresponds to the IB method (Eq. 15) which extends the notion of relevance between functions of samples and parameters in conventional MSS to any joint distribution of $X$ and $Y$. The IB method provides a computational framework for finding approximate MSS in a soft manner by trading off the sufficiency for $Y$ (I(Y; T(X))) and the minimality of the statistic ($I(X,T(X))$) with the Lagrange multiplier $\beta$ (Schwartz-Ziv & Tishby, 2017; Shamir et al., 2010).

### B.2. Connections between MLE and IB

Given that the IB objective is approximately learning MSS in a soft manner, we next build the connections between the popularly adopted maximum likelihood estimation (MLE) in supervised learning and the IB objective. We show that under certain assumptions, MLE is approximating the IB objective defined on the joint distribution of $p_{\text{train}}(X,Y)$.

To facilitate the discussions, we decompose the model parameters into $\theta$ and $\phi$ that denote the parameters of the feature extractor $T_\theta(x)$ and the classifier respectively. MLE minimizes the expected negative log probability under $p_{\text{train}}(X,Y)$:

$$\min_{\theta,\phi} \mathbb{E}_{x,y\sim p_{\text{train}}(X,Y)}[-\log p_{\theta,\phi}(x,y)] \tag{17}$$

$$\iff \min_{\theta,\phi} \mathbb{E}_{x,y\sim p_{\text{train}}(X,Y)}[-\log p_{\phi}(y|T_\theta(x)) - \log p_\theta(x)] \tag{18}$$

Usually, we only model the conditional distribution $p_\phi(Y|X)$ and assume that $p_\theta(X) = p_{\text{train}}(X)$ which is independent from $\theta$. With the assumption that $p_\theta(x) \propto p^\beta(T_\theta(x))), \beta > 0$, (18) can be rewritten as:

$$\min_{\theta,\phi} \mathbb{E}_{x,y\sim p_{\text{train}}(X,Y)}[-\log p_{\phi}(y|T_\theta(x))] + \beta \mathbb{E}_{x\sim p_{\text{train}}(X)}[-\log p(T_\theta(x))] \tag{19}$$

Assume that the neural network parameterized by $\phi$ is a universal function approximator, then we can replace $\min_{\theta,\phi}$ with $\min_\theta$ and (19) can be written as:

$$\min_{\theta} H(Y|T_\theta(X)) + \beta H(T_\theta(X)) \tag{20}$$

$$\begin{aligned} &\text{by (1) } I(Y;T_\theta(X)) = H(Y) - H(Y|T_\theta(X)) \\ &\quad\ \ (2)\ H(T_\theta(X)) = I(X;T_\theta(X)) + H(T_\theta(X)|X) = I(X;T_\theta(X)) \end{aligned}$$

$$\iff \min_{\theta} -I(Y;T_\theta(X)) + \beta I(X;T_\theta(X)) \tag{21}$$

We can see that under the assumption of $p_\theta(x) \propto p^\beta(T_\theta(x)))$, the MLE objective can be converted into the same form as the IB objective. In practice, we usually do not model $p_{\text{train}}(X)$ and only optimize the first term $I(Y;T_\theta(X))$ in (21). However, previous work (Schwartz-Ziv & Tishby, 2017; Geiger, 2020) has shown that deep neural networks (DNNs) are implicitly minimizing $I(X;T_\theta(X))$ with a wide range of activation functions and architectures, which are manifested as a second compression phase during learning with SGD. Thus, we can presumably consider MLE as approximating the IB objective, which is equivalent to learning the MSS on the train distribution $p_{\text{train}}(X,Y)$.

## C. Details of the Online Greedy Algorithm for Group DRO

**Algorithm 2:** Online greedy algorithm for group DRO (Oren et al., 2019)

**Input** : $\alpha$; $m$: total number of groups

Initialize historical average group losses $\hat{L}^{(0)}$; historical estimate of group probabilities $\hat{p}^{train(0)}$; learning rate $\eta$

**for** $t = 1, \cdots, T$ **do**

Sample a mini-batch batch $B = (\mathbf{x}, \mathbf{y}, \mathbf{g})$ uniformly from $p_{\text{train}}$

▷ *Update the historical vectors of* $\hat{L}^{(t)}$ *and* $\hat{p}^{train(t)}$ *for each group* $g \in \{1, \cdots, m\}$

$\hat{L}^{(t)}(g) \leftarrow \text{EMA}(\{\ell(\mathbf{x}_i, \mathbf{y}_i; \theta^{(t-1)}) : \mathbf{g}_i = g\},\ \hat{L}^{(t-1)}(g))$

$\hat{p}^{train(t)} \leftarrow \text{EMA}(\#\text{samples of each group in B},\ \hat{p}^{train(t-1)})$

▷ *Update the worst-case distribution* $q^{(t)}$

Sort $\hat{p}^{train(t)}$ in the order of decreasing $\hat{L}^{(t)}$ and denote the sorted group indexes $\boldsymbol{\pi}$

$q^{(t)}(g_{\boldsymbol{\pi}_i}) = \min\{\frac{\hat{p}^{train(t)}(g_{\boldsymbol{\pi}_i})}{\alpha}, 1 - \sum_{j=1}^{i-1} \frac{\hat{p}^{train(t)}(g_{\boldsymbol{\pi}_j})}{\alpha}\}$

▷ *Update model parameters* $\theta$

$\theta^{(t)} = \theta^{(t-1)} - \frac{\eta}{|B|} \sum_{i=1}^{|B|} \frac{q^{(t)}(\mathbf{g}_i)}{\hat{p}^{train(t)}(\mathbf{g}_i)} \nabla \ell(\mathbf{x}_i, \mathbf{y}_i; \theta^{(t-1)})$

**end**

EMA refers to exponential weighted moving average such that $\text{EMA}(v_1, v_2) = \gamma v_1 + (1 - \gamma) v_2$, where $\gamma \in (0, 1)$.

## D. Synthetic Experiments: on Investigation Spurious Features under Covariate Shift

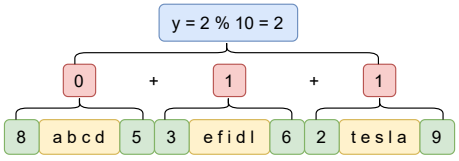


*Figure 5.* An illustrative example of the synthetic task.

**Synthetic Experiments** We design synthetic experiments where data is generated based on the ground-truth rules and different biases are injected. We show that even in the presence of necessary information to learn the rules, the ERM model (specifically, we examine MLE) can still learn spurious features or miss robust features under covariate shift. The synthetic task aims to predict an integer $y \in \{0, \cdots, 9\}$ conditioned on a sequence $x$ as shown in Fig. 5. Concretely, $x$ is composed of $m$ chunks, where each chunk $c_i$ has $|c_i|$ characters that are randomly sampled from an alphabet $\mathcal{V}$. We prepend an integer $c_i^1$ and append an integer $c_i^2$ to each chunk $c_i$, and both $c_i^1$ and $c_i^2$ are uniformly sampled from $[1, 10]$. The target integer $y$ is predicted following the rules: each triple of $(c_i^1, c_i, c_i^2)$ produces an indicator value $d_i$; $d_i = c_i^2 - c_i^1$ if $c_i^2 > c_i^1$, otherwise $d_i = 0$; then $y = (\sum_{i=1}^{m} d_i) \mod 10$. We set $3 \leq m \leq 6$, $3 \leq |c_i| \leq 5$ and $|\mathcal{V}| = 26$,. We use a one-layer bidirectional LSTM (Hochreiter & Schmidhuber, 1997) to model the input sequence and use the final hidden states of the LSTM to predict the target value. We create training data following the the above description and design two settings that introduce covariate shift to examine if the model can learn the rules with ERM.

**(a) Setting 1 — ERM-trained models can miss robust features under covariate shift**: We create the training data by imposing $c_m^2 > c_m^1$ on the last chunk $c_m$ of all the training samples. When we create the training data, the rules applied to each chunk are the same as described above, which means that the model does not need to learn additional rules for the last chunk. We are interested in examining whether the model trained with ERM will apply the rules learned from other chunks to the last one or it will miss the robust features of the last chunk. At test time, we evaluate on two groups of test sets: $\mathcal{D}_{out}$ where $c_m^2 \leq c_m^1$, different from the training data, and $\mathcal{D}_{in}$ where $c_m^2 > c_m^1$, consistent with the training data. From Tab. 5, we see that the test accuracy on $\mathcal{D}_{out}$ is much lower that that on $\mathcal{D}_{in}$. This demonstrates that the model only learns robust features from chunks $c_1^{m-1}$ but misses the robust features of the last chunk $c_m$. We conjecture that the model trained with ERM learns in a lazy way where it tries to minimize the entropy of learned features by memorizing patterns and taking shortcuts as discussed further in Appendix B.2.

**(b) Setting 2 — ERM-trained models can learn spurious features under covariate shift**: In the second setting, we inject spurious patterns into the training data that co-occur with the rules we aim to learn. As both robust rules and spurious patterns co-exist in the training data, we would like to see whether the model picks up the spurious ones or the robust

| | $\mathcal{D}_{in}$ | $\mathcal{D}_{out}$ |
|---|---|---|
| Setting 1 | 99.93 $\pm$ 0.02 | 14.68 $\pm$ 2.60 |
| Setting 2 | 100.00 $\pm$ 0.00 | 10.26 $\pm$ 0.25 |

*Table 5.* Test accuracy of the synthetic task.

ones. Specifically, each training input sequence has a chunk $c_j$ that includes a special segment of characters, e.g. *a b*. The remainder of $d_j = c_j^2 - c_j^1$ and the sum of all indicators $\sum_{i=1}^{m} d_i$ mod by 10 are the same such that the target label $y$ is always the same as the indicator $d_j$. Similarly, we test on two cases: i) $\mathcal{D}_{in}$ where every sequence includes a special chunk as in the training set; ii) $\mathcal{D}_{out}$ where characters in each chunk are uniformly sampled. We can see from Tab. 5 that the model learns to use the spurious patterns to predict the target label instead of the general rules.

## E. Experimental Details

### E.1. Models and Training Details

**Model Specific Settings** In our method, we adopt two criterions in GC-DRO to determine when to update $q(x, y|g)$ for each groups: (1) update when the robust validation accuracy drops (2) update at every epoch. With (2), $q(x, y|g)$ is updated more frequently. For MNLI and Celeb-A, we use the second criterion. For FDCL18, we use the first criterion, because this is a relatively smaller dataset and updating $q(x, y|g)$ less frequently makes training more stable. Every time $q(x, y|g)$ is updates, we clear the historical losses in EMA that is used for updating $q(g)$. We use exponentially weighted moving average (EMA) to compute the historical losses for both $q(g)$ and $q(x, y|g)$, for which we denote $\text{EMA}_\text{G}$ and and $\text{EMA}_\text{CG}$ respectively. As shown above, we use $\gamma$ to denote the coefficient for current value in EMA, thus $1-\gamma$ is used to the historical value. We found that the value of $\gamma$ is an important hyperparameter in some cases to achieve better performance, since the final $q$ distribution is computed through sorting the losses accumulated via EMA. Basically, a higher $\gamma$ pays more attention to the current value. We search over $\{0.1, 0.5\}$ for both $\gamma$ used in $\text{EMA}_\text{G}$ and $\text{EMA}_\text{CG}$ respectively. Through the robust accuracy on the validation set, we set both $\gamma$'s to be 0.5 for the NLP tasks except that for the imperfect partition of toxicity detection we set $\gamma$ used in $\text{EMA}_\text{G}$ to be 0.1. For the image task, we set both $\gamma$'s to be 0.1. For the $\gamma$ used in accumulating the historical fractions of groups, we always use a small value 0.01.

**Training Details** For the NLP tasks, we finetune a base Roberta model (Liu et al., 2019; Ott et al., 2019) and we segment the input text into the sub-word tokens using the tokenization described in (Liu et al., 2019). During training, we sample minibatches that contain at most 4400 tokens. We train MNLI using Adam (Kingma & Ba, 2014) with an intitial learnig rate of $1e-5$ for 35 epochs and FDCL18 for 45 epochs, and we linearly decay the learning rate at every step until the end of training. For the image task, we fine-tune a ResNet-18 (He et al., 2016) for 50 epochs with batch size of 256. We use SGD with learning rate of $1e-4$. At the end of every epoch, we evaluate the robust accuracy on the validation set. We train on one Volta-16G GPU and it takes around 2 - 5 hours to finish one experiments for different datasets.

### E.2. Implementation of the Group DRO Loss

We referred to the implementation of greedy group DRO in Sagawa et al. (2020a), where they use the exact formulation in Eq. 5 to compute the expected loss, which leads to inferior performance compared to the exponentiated-gradient based optimization as reported in Sagawa et al. (2020a). The implementation computed the final loss by first computing the average loss over instances for each group (MC for the inner expectation), then compute the full expected value over the averaged group loss, as shown below:

$$\ell(\mathbf{x}, \mathbf{y}, \mathbf{g}; \theta) = \sum_g q(g)\bar{\ell}(g) = \sum_g q(g)\frac{1}{C_g}\sum_{\{i, \forall \mathbf{g}_i = g\}} \ell(\mathbf{x}_i, \mathbf{y}_i; \theta), \tag{22}$$

where $(\mathbf{x}, \mathbf{y}, \mathbf{g})$ is a mini-batch and $C_g$ is the number of samples that belong to group $g$ in the mini-batch. We can see that instances that belong to different groups are weighted correspondingly by the number of group size in a mini-batch. This causes that instances in large group get unfairly lower weights, especially when its probability in the $q$ distribution is low. We fix this by directly computing the expected loss over the joint distribution of $q(x, y, g)$, i.e. $\mathbb{E}_{(x_i,y_i,g_i)\sim q(x,y,g)}\ell(x_i, y_i, g_i; \theta) = \mathbb{E}_{(x_i,y_i,g_i)\sim p_\text{train}(x,y,g)}\frac{q(x_i,y_i,g_i)}{p_\text{train}(x_i,y_i,g_i)}\ell(x_i, y_i, g_i)$. Specifically, we do this by summing over all the importance weighted instance losses using corresponding group weights and taking average. This allows us to obtain unbiased gradient estimates of $\theta$.

$$\frac{1}{N}\sum_i \frac{q(\mathbf{g}_i)}{p_\text{train}(g_i)}\ell(\mathbf{x}_i, \mathbf{y}_i; \theta) \tag{23}$$