Presentation for *Improving predictive inference under covariate shift by weighting the log-likelihood function* and *Distributionally Robust Language Modeling*

Tom Tian, Siting Li

February 7, 2024

Motivation

A pioneering work in the field of transfer learning
There's overlap between the source and target distribution, and we want to resample to make the model generalizes better on target domain
One specific case when we break iid-ness

Motivation

Covariate Shift: Assume that the input space of the source domain and the target domain are both $\mathcal{X}$, and the output space are both $\mathcal{Y}$. The marginal distribution of the source domain $P_S(x)$ is different from the marginal distribution of the target domain $P_T(x)$, but the conditional distribution of the two domains are the same

$$P_S \neq P_T$$

$$P_S(y|x) = P_T(y|x)$$

Motivation

1. Assume that x is labeled by the probability model $P_{true}(y|x)$ coming from model space $F = \{P(y|x;\theta)|\theta \in \Theta\}$
2. Use the dataset $D = \{x^{(i)}, y^{(i)}\}$ to fit the probability model $P(y|x;\hat{\theta})$ as an estimate of the real probability model $P(y|x)$.
3. Use variants of maximum likelihood estimation to estimate parameters:

$$\hat{\theta} = \arg\min \sum_i \log P(y^{(i)}|x^{(i)};\theta)$$

As long as $P_{true}(y|x) \in F$, and $P_S(y|x) = P_T(y|x)$, then regardless of whether P_S and P_T are equal, we can use dataset from the source domain to fit the real model

Motivation

If the model space we choose does not include the real model ($P_{true} \notin F$), then we are actually using a misspecified model to evaluate on test set, which leads to poor performance

- No model misspecification: MLE works!
- No covariate shift but we have model misspecification: We'll need to enlarge model space F; Parameter estimation has nothing wrong
- Both covariate shift and model misspecification: What this paper discusses!

Motivation

If the model space we choose does not include the real model ($P_{true} \notin F$), then we are actually using a misspecified model to evaluate on test set. Since $P_S \neq P_T$ and we only use labeled data from the source domain, the performance of this model give by MLE in the target domain will become very poor (particularly when P_S and P_T differ a lot). The paper considers under the misspecified model and $P_S \neq P_T$ case, how are we going to to improve our parameter estimation method.

- No model misspecification: MLE works!
- No covariate shift but we have model misspecification: We'll need to enlarge model space F; Parameter estimation has nothing wrong
- Both covariate shift and model misspecification: What this paper discusses!

Setting

- We have both covariate shift and model misspecification
- P_S and P_T are known

Why unweighted model is bad

For linear regression case, Ordinary Linear Regression is very bad if distribution of source domain is different from target domain, but it performs well if they are the same. We'll want to use Weighted Linear Regression instead.

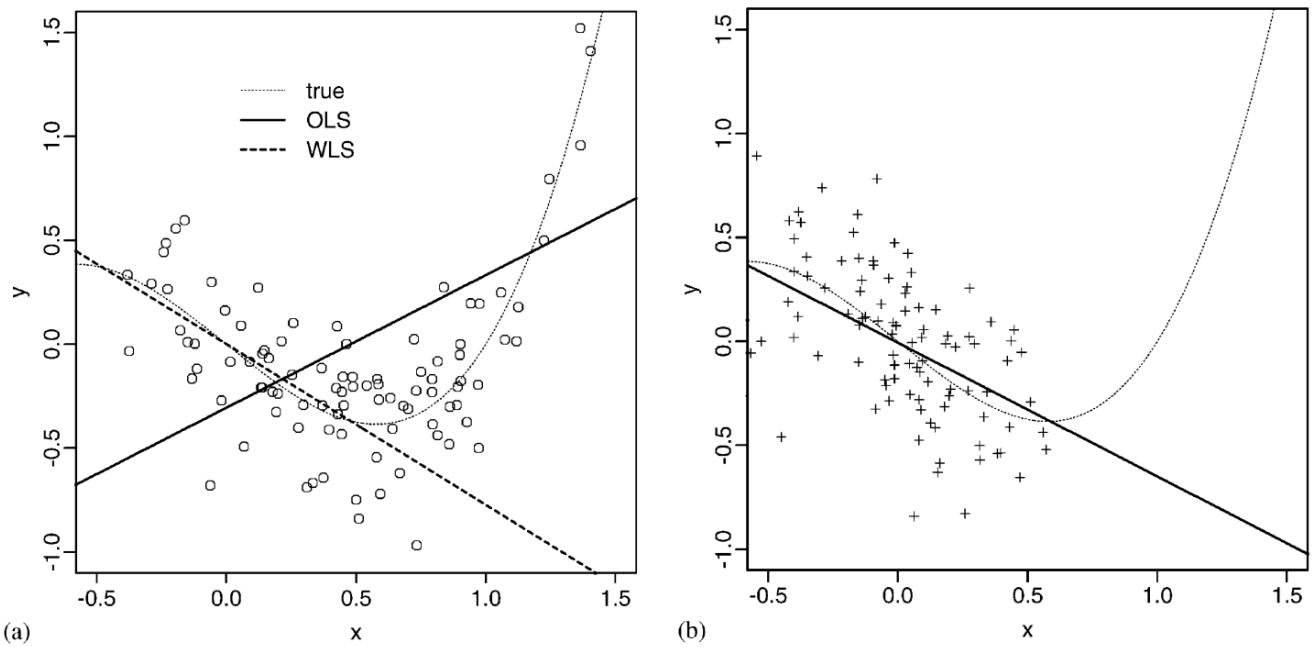


Fig. 1. Fitting of polynomial regression with degree $d = 1$. (a) Samples (x_t, y_t) of size $n = 100$ are generated from $q(y|x)q_0(x)$ and plotted as circles, where the underlying true curve is indicated by the thin dotted line. The solid line is obtained by OLS, and the dotted line is WLS with weight $q_1(x)/q_0(x)$. (b) Samples of $n = 100$ are generated from $q(y|x)q_1(x)$, and the regression line is obtained by OLS.

If we have many samples from source, i.e. $m_S >> m_T$

So that the variance of $\hat{\theta}_w$ is minimal and we can ignore it

$$E_{P_T}(-\log P(y|x;\theta)) = E_{P_S}(-\frac{P_T(x)}{P_S(x)} \log P(y|x;\theta))$$

$$\lim_{m_s \to +\infty} -\frac{1}{m_S} \sum_{i=1}^{m_s} \frac{P_T(x^{(i)})}{P_S(x^{(i)})} \log P(y^{(i)}|x^{(i)};\theta) = E_{P_S}(-\frac{P_T(x)}{P_S(x)} \log P(y|x;\theta))$$

where the first one is by Radon–Nikodym theorem, and the second one is by SLLN

$$\hat{\theta}_w = \arg\min -\frac{1}{m_S} \sum_{i=1}^{m_s} \frac{P_T(x^{(i)})}{P_S(x^{(i)})} \log P(y^{(i)}|x^{(i)};\theta)$$

In transfer learning, typically we have way many data from source domain than target domain, so we typically choose $w(x) = \frac{P_T(x)}{P_S(x)}$, that is, $\lambda = 1$. This is one significant theoretical underpinning on why we do this.

What if we don't have that many?

Lemma 2. *The expected loss is asymptotically expanded as*

$$E_0^{(n)}(\mathrm{loss}_1(\hat{\theta}_w)) = \mathrm{loss}_1(\theta_w^*) + \frac{1}{n}\left\{ K_w^{[1]\prime} b_w + \frac{1}{2}\mathrm{tr}(K_w^{[2]} H_w^{-1} G_w H_w^{-1}) \right\} + \mathrm{o}(n^{-1}), \tag{4.1}$$

where the elements of $K_w^{[1]}$ *and* $K_w^{[2]}$ *are defined by*

$$(K_w^{[k]})_{i_1 \cdots i_k} = -E_0 \left\{ \frac{q_1(x)}{q_0(x)} \left. \frac{\partial^k \log p(y|x,\theta)}{\partial\theta^{i_1} \cdots \partial\theta^{i_k}} \right|_{\theta_w^*} \right\}$$

and b_w *is the asymptotic limit of* $nE_0^{(n)}(\hat{\theta}_w - \theta_w^*)$, *which is of order* $\mathrm{O}(1)$.

where 0 means source, 1 means target, and *loss* means the difference between true conditional $P(y|x)$ and conditional model give by θ assuming distribution of x is with respect to source or target in KL divergence sense. The first term is the goodness of the model (the small the better), and the second term is the complexity of the model (also the small the better)

What if we don't have that many?

We don't know $P(y|x)$!
This comes to the criterion, which is the main contribution of the paper.

Then, $\mathrm{IC}_w/2n$ *is an estimate of the expected loss unbiased up to* $\mathrm{O}(n^{-1})$ *term:*

$$E_0^{(n)}(\mathrm{IC}_w/2n) = E_0^{(n)}(\mathrm{loss}_1(\hat{\theta}_w)) + \mathrm{o}(n^{-1}). \tag{5.2}$$

When n (in fact it's the size of source domain m_S) is large, the criterion is sufficiently nice

Main contribution

We adapt MLE to MWLE (where W for Weighted)

$$\hat{\theta}_w = \arg\min \sum_i w(x^{(i)}) \log P(y^{(i)}|x^i;\theta)$$

$$IC_w = -2\sum_{i=1}^{m_S} \frac{P_T(x^{(i)})}{P_S(x^{(i)})} \log P(y^{(i)}|x^i;\hat{\theta}_w) + 2\,Tr(\hat{J}_w \hat{H}_w^{-1})$$

$$J_w = E_S\left[\frac{P_T(x)}{P_S(x)} \nabla_\theta \log P(y|x;\theta)\Big|_{\theta^*} \otimes w(x)\nabla_\theta \log P(y|x;\theta)\Big|_{\theta^*}\right]$$

$$H_w = -E_S\left[w(x)^2 \nabla_\theta^2 \log P(y|x;\theta)\Big|_{\theta^*}\right]$$

and we use plug-in estimators for $\hat{J}_w$ and $\hat{H}_w$, which are consistent estimators

θ^* is the θ that attains ordinary MLE of $P(y|x)$ (but not the true model because of misspecification!) (note that S and T have the same $P(y|x)$, so there's no ambiguity)

Comparing with the following:

Akaike Information Criterion

$$AIC = -2 \log q(x|\theta) + 2k$$

Takeuchi Infomration Criterion

$$TIC = -2 \log q(x|\theta) + 2 \hat{Tr}(J(\theta_0) H(\theta_0)^{-1})$$

$$J(\theta) = E_p \left[\nabla_\theta \log q(x|\theta) \Big|_{\theta_0} \otimes \nabla_\theta \log q(x|\theta) \Big|_{\theta_0} \right]$$

$$H(\theta) = -E_p \left[\nabla_\theta^2 \log q(x|\theta) \Big|_{\theta_0} \right]$$

where p is the true model distribution, k is the number of parameters, q is a specific candidate model distribution, θ_0 is the θ that attains MLE wrt p, and $\log q(x|\theta)$ is the logarithm maximum likelihood of the model
$J(\theta)$ is the expected gradient of likelihood. $H(\theta)$ is the expected negative Hessian of likelihood.

Main contribution

When we have some candidate weight functions $w(x)$ and the model space F, we can calculate IC_w using only the labeled data from source domain and choose the w with minimal IC_w. This paper predicates that $w(x) = (\frac{P_T(x)}{P_S(x)})^\lambda, \lambda \in [0, 1]$, so choosing appropriate $w(x)$ becomes choosing appropriate λ.

This paper assumes P_S and P_T are known, which seems unrealistic, but this paper assumes we can always use plug-in estimator give by the data to have such empirical distribution.

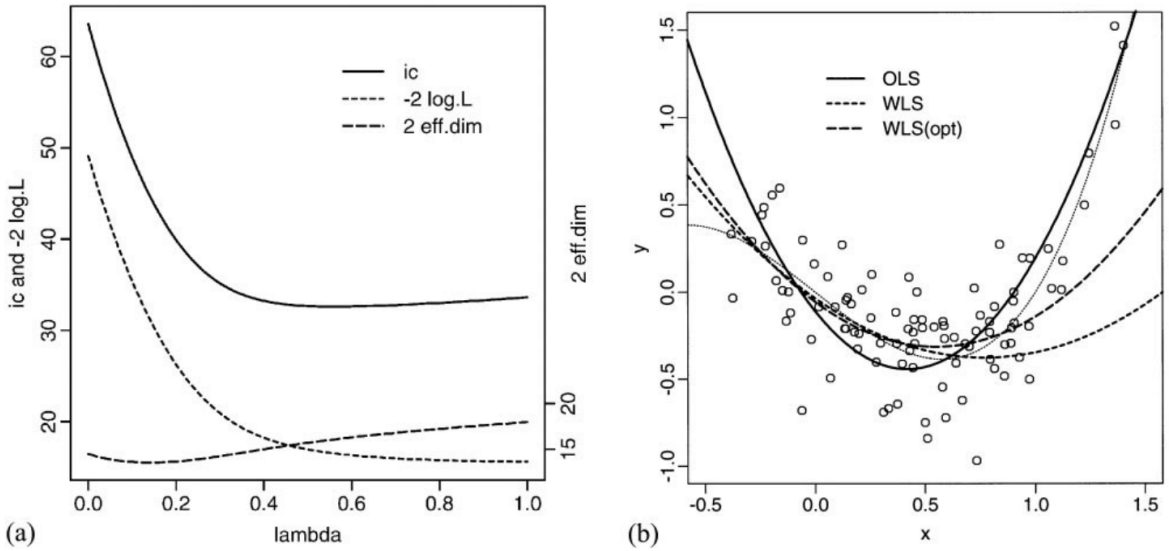


Fig. 2. (a) Curve of IC_w versus $\lambda \in [0,1]$ for the model of Section 2 with $d=2$. The weight function (5.3) connecting from $w(x) \equiv 1$ (i.e. $\lambda=0$) to $w(x)=q_1(x)/q_0(x)$ (i.e. $\lambda=1$) was used. Also shown are $-2L_1(\hat{\theta}_w)$ in dotted lines, and $2\,\mathrm{tr}(J_w H_w^{-1})$ in broken lines. (b) The regression curves for $d=2$. The WLS curve with the optimal $\hat{\lambda}$ as well as those for OLS ($\lambda=0$) and WLS ($\lambda=1$) are drawn.

Table 1
IC_w values with weight (5.3) for $\lambda=0$, $\lambda=1$, and $\lambda=\hat{\lambda}$. Also shown is $\hat{\lambda}$ value. Calculated for the polynomial regression example of Section 2 with $d=0,\ldots,4$

	$d=0$	$d=1$	$d=2$	$d=3$	$d=4$
$\lambda=0$	138.72	174.02	63.59	28.97	31.75
$\lambda=1$	73.96	33.23	33.64	34.80	34.98
$\lambda=\hat{\lambda}$	73.92	32.68	32.62	28.96	31.75
$\hat{\lambda}$	0.95	0.77	0.56	0.01	0.00

We know if there is no model misspecification, we just use non-weighted regression model. If the class is less and less rich, we will be more heavy on weighting.

Experiments

Table 3
Asymptotic convergence of (5.2). $2n\,\mathrm{loss}_1(\hat{\theta}_w)+2L_1(\hat{\theta}_w)$ is calculated for the Monte-Carlo replicates, and its average is tabulated in the left columns. For $n \geqslant 300$, this agrees very well with the average of $2\,\mathrm{tr}(\hat{J}_w\hat{H}_w^{-1})$ tabulated in the right columns. $2\,\mathrm{tr}(J_wH_w^{-1})$ is shown in the $n=\infty$ row

n	$d=0$				$d=1$				$d=2$			
	$\lambda=0$		$\lambda=1$		$\lambda=0$		$\lambda=1$		$\lambda=0$		$\lambda=1$	
50	1.8	1.7	9.9	7.7	6.8	6.1	15.7	11.1	26.4	13.1	24.8	13.4
100	1.5	1.4	9.2	8.1	6.0	5.7	14.2	12.0	22.6	14.9	19.8	14.9
300	1.3	1.3	8.8	8.4	5.4	5.3	13.3	12.6	18.5	15.7	17.3	16.0
1000	1.2	1.2	8.7	8.6	5.1	5.1	13.1	12.9	16.2	15.4	16.7	16.3
∞		1.2		8.6		5.0		13.0		14.9		16.4

This verifies is *IC* is indeed a good estimate of loss on target domain asymptotically (verifies equation 5.2)

Experiments

Table 4
The expected loss for the selected weight and the selected model. $2n$ times of $\text{loss}_1(\hat{\theta}_w) + E_1(\log q(y|x))$ is calculated for the replicates, and its average is tabulated in the columns of $\lambda = 0, 1$ for $d = 0, 1, 2$. The average of the loss of the selected weight is shown in the columns of $\hat{\lambda}$. The right most columns show the average of the loss of the selected model

β_3	$d=0$			$d=1$			$d=2$			$\hat{d}$		
	0	1	$\hat{\lambda}$	0	1	$\hat{\lambda}$	0	1	$\hat{\lambda}$	0	1	$\hat{\lambda}$
1.0	98.0	50.7	50.9	123.8	12.3	11.9	71.7	16.0	16.7	73.2	15.0	15.3
0.5	96.0	61.1	61.1	49.9	9.0	8.2	25.6	11.8	11.5	26.9	11.0	10.8
0.2	142.2	68.4	68.4	12.6	7.8	6.4	8.5	10.4	8.1	10.1	9.6	7.9
0.1	152.8	71.1	71.0	6.0	7.8	5.7	5.9	10.4	7.2	6.3	9.6	6.9
0.0	162.2	73.8	73.7	3.6	7.7	4.8	5.0	10.2	6.6	4.4	9.5	6.0

The selection criterion generally performs well in both correctly specified case and blatantly misspecified case

Switch gear to overparameterized setting of deep learning network

Jonathon Byrd, Zachary C. Lipton (2018)
What is the Effect of Importance Weighting in Deep Learning?

Switch gear to overparameterized setting of deep learning network

In deep learning networks, many practical datasets are separable. What is the effect of importance weighting?
What is the role of importance weighting for large over-parameterized deep learning networks?
One thing: On an over-parameterized model, even if we have perfect accuracy (100%) on the training set, we can still improve generalization ability by continue training (obtain better performance on the test set)

Theory

Linear model will converge to a solution with infinite norm for separable data.

Lemma 1 *Let* $\mathbf{w}(t)$ *be the iterates of gradient descent (eq. 2) with* $\eta < 2\beta^{-1}\sigma_{\max}^{-2}(\mathbf{X})$ *and any starting point* $\mathbf{w}(0)$. *Under Assumptions 1 and 2, we have: (1)* $\lim_{t\to\infty}\mathcal{L}(\mathbf{w}(t)) = 0$, *(2)* $\lim_{t\to\infty}\|\mathbf{w}(t)\| = \infty$, *and (3)* $\forall n : \lim_{t\to\infty}\mathbf{w}(t)^\top \mathbf{x}_n = \infty$.

Linear model will converge to the max-margin solution of svm for separable data.

Theorem 3 *For any dataset which is linearly separable (Assumption 1), any* β*-smooth decreasing loss function (Assumption 2) with an exponential tail (Assumption 3), any stepsize* $\eta < 2\beta^{-1}\sigma_{\max}^{-2}(\mathbf{X})$ *and any starting point* $\mathbf{w}(0)$, *the gradient descent iterates (as in eq. 2) will behave as:*

$$\mathbf{w}(t) = \hat{\mathbf{w}}\log t + \boldsymbol{\rho}(t)\ , \tag{3}$$

where $\hat{\mathbf{w}}$ *is the* L_2 *max margin vector (the solution to the hard margin SVM):*

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}}\|\mathbf{w}\|^2 \ \text{ s.t. } \mathbf{w}^\top\mathbf{x}_n \geq 1, \tag{4}$$

and the residual grows at most as $\|\boldsymbol{\rho}(t)\| = O(\log\log(t))$, *and so*

$$\lim_{t\to\infty}\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} = \frac{\hat{\mathbf{w}}}{\|\hat{\mathbf{w}}\|}.$$

Furthermore, for almost all data sets (all except measure zero), the residual $\rho(t)$ *is bounded.*

Theory

- Implicit bias of SGD can explain why on an over-parameterized model, even if we have perfect accuracy (100%) on the training set, continued training can still improve generalization ability (obtain better performance on the test set). This is because even if The decision boundary has perfectly separated training data, if we continue training, we can get a max-margin solution, which is better than a random perfect accuracy classifier
- For the overparameterized model, any data is separable (100% accuracy on the train set). This means that overparameterized model will converge to the max margin solution of svm
- The max margin solution only depends on the position of the support vector and has nothing to do with the relative weight between samples, so importance weight doesn't affect convergent solution

Empirical result

Importance Weight is effective in the early stages of training. For over-parameterized models, this effect disappears with training. IW has no effect on the the final converging solution

IW is still effective for under-parameterized models (depending on whether training data is separable)

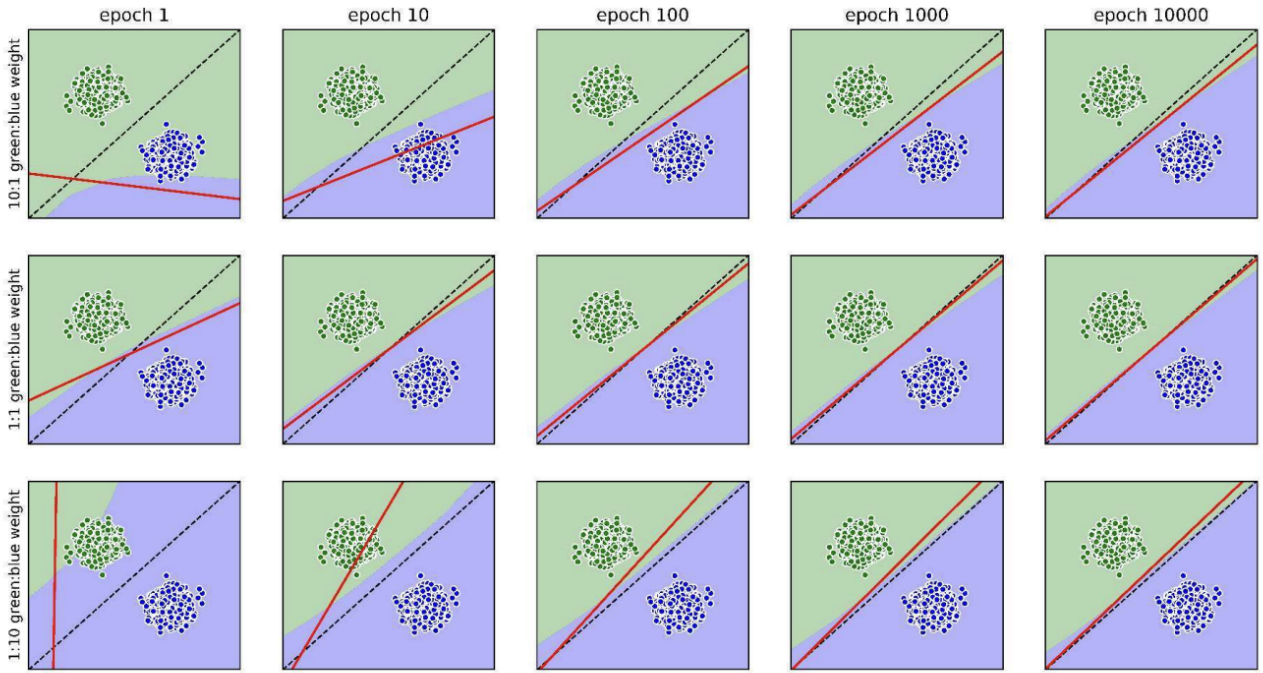


Figure 1. Convergence of decision boundaries over epochs of training with different importance weights (top to bottom). Points are colored according to their true labels, with background shading depicting the decision surface of an MLP with a single hidden layer of size 64. The red line shows the logistic regression decision boundary. The dotted black line shows the max-margin separator.

Empirical result

L2 norm regularization can partially reflect the change caused by IW, because L2 norm prevents weights from having infinite norm, so under L2 regularization, the model cannot converge to the max margin solution, so different IW will give different solution.

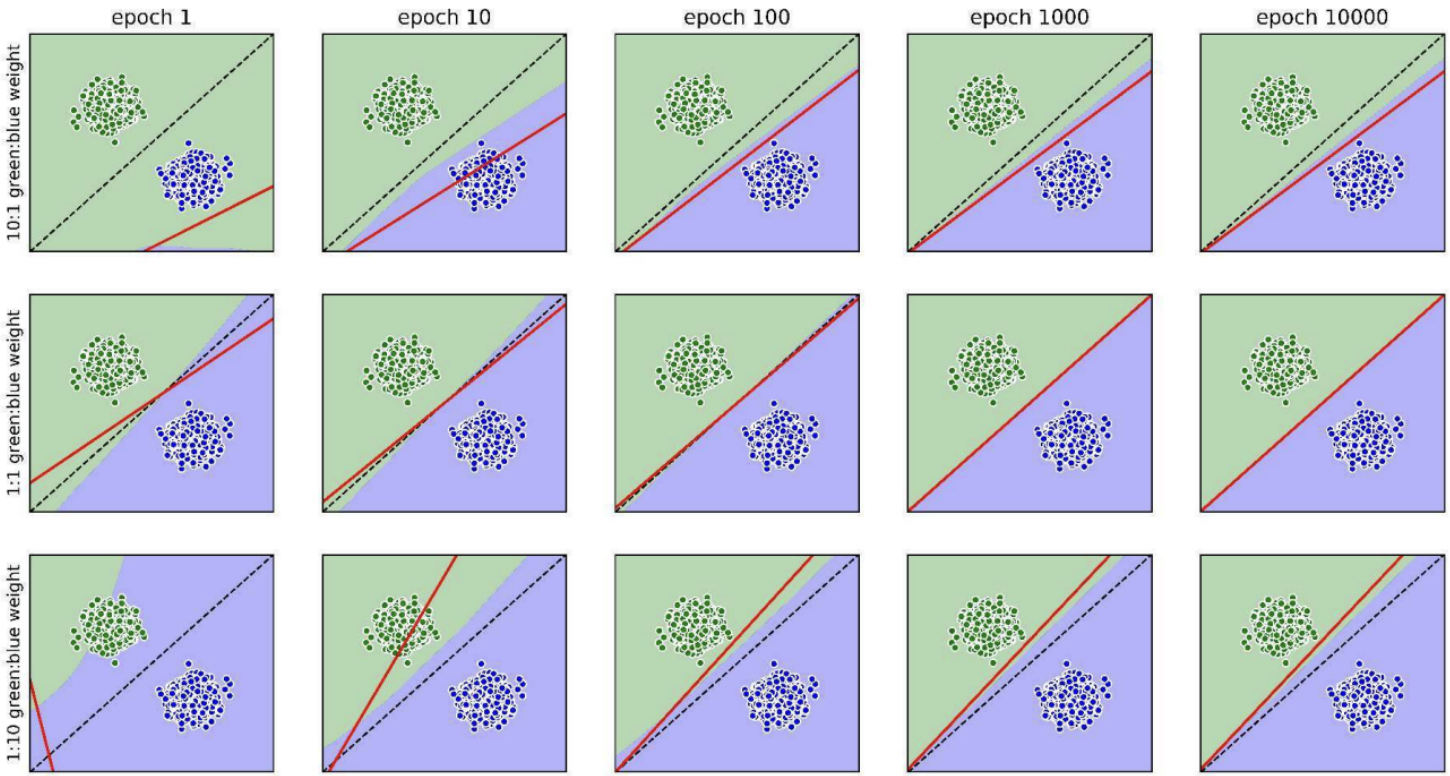


Figure 2. Same scenario as Figure 1, except both logistic regression and MLP are trained with L2 regularization.

In terms of language model

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, Percy Liang (2023)
Data Selection for Language Models via Importance Resampling

In terms of language model

Given a large and diverse original dataset (e.g. The Pile) and a smaller target dataset, we want to select a subset from the original data whose distribution is similar to the target distribution. A
A natural approach is to resample the original data based on importance weights (importance resampling), estimating importance weights on high-dimensional data such as text is often hard.

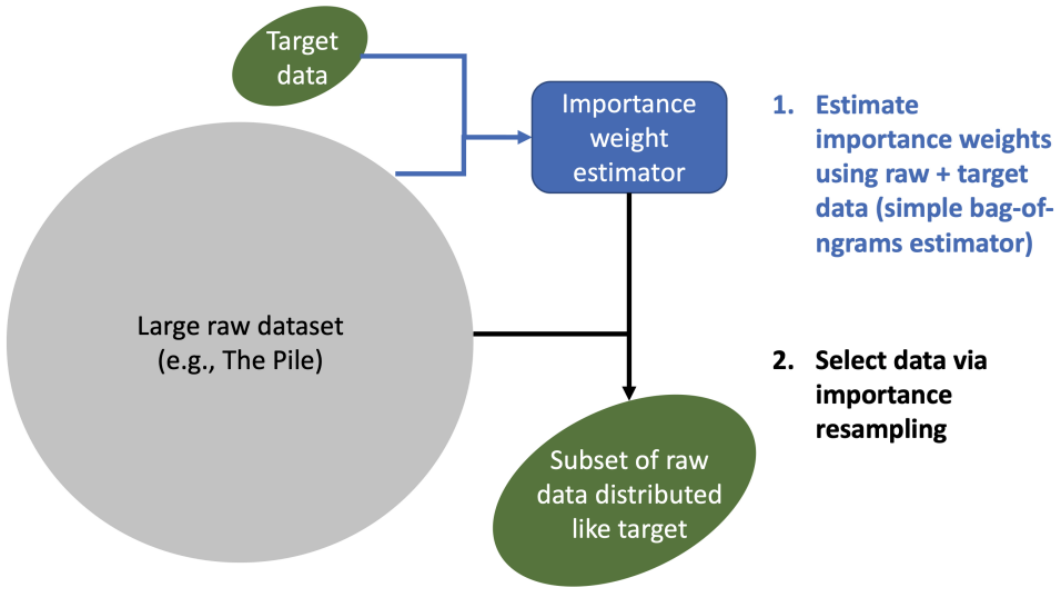


Figure 1: Given a large raw dataset such as The Pile (Gao et al., 2020) and a smaller target dataset (e.g., Wikipedia + books), we aim to select a subset of the raw data that is distributed like the target in some feature space. Our method, DSIR, first estimates importance weights using raw and target data in an n-gram feature space. The importance weights are used to resample a subset of the raw dataset.

In terms of language model

- Idea: Map the original and target data to a certain feature space and resample original data by the importance weights in this feature space
- What is a feature space that is both computationally efficient and captures aspects of pre-training data that are relevant to downstream tasks?
- Strong correlation between n-gram feature space and downstream task performance
- Computable by KL reduction, which measures how much the KL divergence of the selected data to the target data is reduced compared to random data in terms of n-gram feature space

$$KL(target||random) - KL(target||selected)$$

KL reduction

- A strong correlation between KL reduction and the average downstream performance of the eight downstream datasets
- Easy to compute and predicts downstream accuracy without training a language model

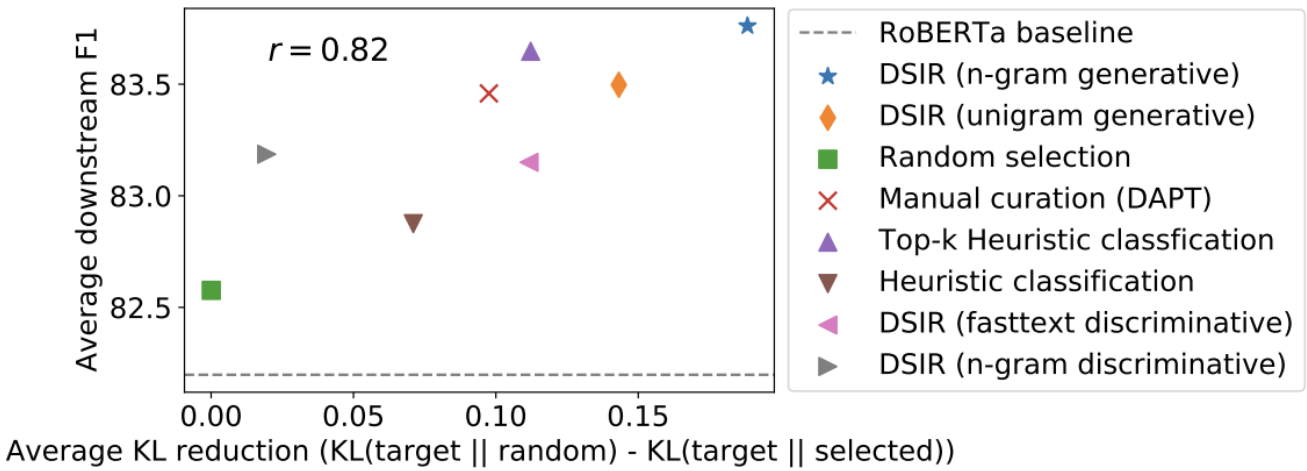


Figure 3: Plot of average KL reduction on the n-gram feature space, defined as how much the selected dataset reduces KL divergence to the target distribution over just random sampling from The Pile, against average downstream F1 score over the 8 continued pretraining datasets in Table 1. There is a strong correlation between KL reduction and downstream performance (Pearson $r = 0.82$).

References

Barum Park (2018)

AIC and BIC

https://barumpark.com/blog/2018/aic-and-bic/

Hidetoshi Shimodaira (2000)

Improving predictive inference under covariate shift by weighting the log-likelihood function

Journal of Statistical Planning and Inference 90 (2000) 227-244

Jonathon Byrd, Zachary C. Lipton (2018)

What is the Effect of Importance Weighting in Deep Learning?

https://arxiv.org/abs/1812.03372

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, Percy Liang (2023)

Data Selection for Language Models via Importance Resampling

https://arxiv.org/abs/2302.03169

Distributionally Robust Language Modeling

Yonatan Oren*[1] Shiori Sagawa*[1] Tatsunori B. Hashimoto*[1,2] Percy Liang[1]
(* equal contribution)
[1]Stanford Computer Science [2]Stanford Statistics
{yonatano,thashim}@stanford.edu {ssagawa,pliang}@cs.stanford.edu

Presenter: Siting Li

Feb 7, 2024

Distributionally Robust Optimization (Ben-Tal et al. 2013)

- ERM: $$\hat{\theta}_{\text{ERM}} := \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\hat{P}}[\ell(\theta; (x, y))]$$
- The test distribution may be different from the training distribution.
- We still hope that the model trained on training set performs well on the test set.
- **DRO Objective**: $$\min_{\theta \in \Theta} \left\{ \mathcal{R}(\theta) := \sup_{Q \in \mathcal{Q}} \mathbb{E}_{(x,y)\sim Q}[\ell(\theta; (x, y))] \right\}.$$
  - Q: **uncertainty set**.
  - Does it consider all kinds of distribution shift?
  - Advantages over fine-tuning and domain adaptation

Sagawa et al. 2020.

How to define the uncertainty set Q?

- Considering all distributions is unrealistic since there is no free lunch.
- Here, the paper considers subpopulation shift: The test distribution should be a **subpopulation** of the training distribution.
- Moreover, the subpopulation should not be far from the training distribution – **conditional value at risk (CVaR)**:

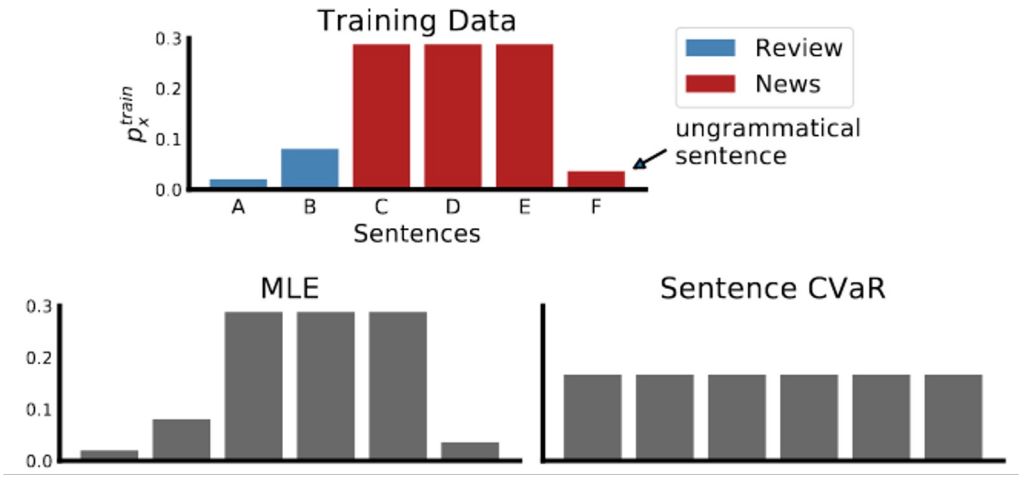
$$\mathcal{P}_{\mathrm{X}}^{\alpha} := \{p_{\mathrm{X}} : \alpha p_{\mathrm{X}}(x) \leq p_{\mathrm{X}}^{\mathrm{train}}(x) \ \ \forall x\}.$$

- Other definitions: using the f-divergence (Ben-Tal et al. 2013, Hu et al. 2018).

Oren et al. 2019, Rockafellar and Uryasev. 2000.

How to define the uncertainty set Q?

- However, such worst-case subpopulations are attained by adversarially choosing the hardest, most unusual cases. (“Overly pessimistic”)



“Meaningful” Subpopulation?

↓

Prior Knowledge

Hu et al. 2018, Oren et al. 2019

Group DRO

- **Group DRO**: Divide the data points x into groups (z), and use group as the granularity for analysis:

$$\mathcal{P}_{\mathrm{z}}^{\alpha} := \{p_{\mathrm{z}} : \alpha p_{\mathrm{z}}(z) \leq p_{\mathrm{z}}^{\mathrm{train}}(z) \ \ \forall z\}$$

$$\sup_{p_{\mathrm{z}} \in \mathcal{P}_{\mathrm{z}}^{\alpha}} \mathbb{E}_{z \sim p_{\mathrm{z}}} \left[\mathbb{E}_{x \sim p_{\mathrm{x}|z}} \left[\ell(x; \theta) \right] \right]$$

- Proposed in Hu et al. 2018, termed as “latent category”.
- Obtain group information by clustering.
- Use existing group information: domain label in WILDS.

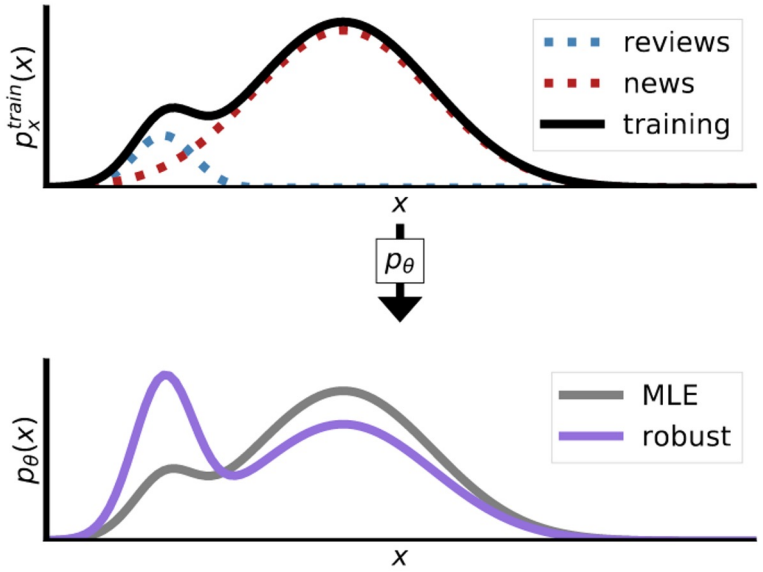
Oren et al. 2019.

Train Group DRO Models

$$\sup_{p_Z \in \mathcal{P}_Z^{\alpha}} \mathbb{E}_{z \sim p_Z} \left[\mathbb{E}_{x \sim p_{X|z}} \left[\ell(x; \theta) \right] \right]$$

- Dynamically reweighting the loss for different groups to ensure the “sup”?

- Previous approaches for DRO (e.g. **Lagrangian duality**. (Duchi et al. 2018.), **Online algorithms** (Namkoong and Duchi. 2018.)) did not consider **the group structure**.
- Hu et al. 2018 considered two specific f-divergences.



Oren et al. 2019.

New Approach of DRO

- Still formulate the problem as a **two-player minimax game**:

$$\inf_{\theta} \sup_{p_z \in \mathcal{P}_z^{\alpha}} \mathbb{E}_{z \sim p_z} \left[L(z; \theta) \right]$$

- At each iteration t, p_z is updated by selecting an optimal value with respect to historical losses up to the current iteration:

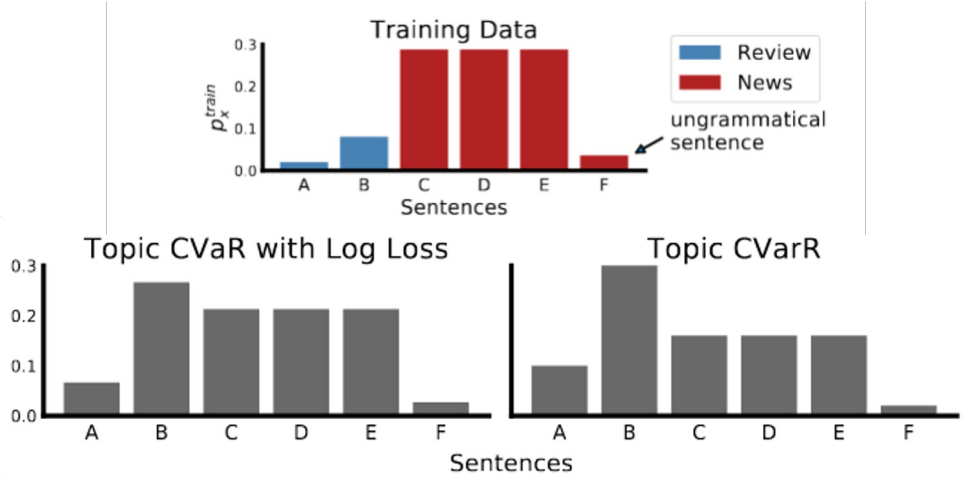
$$p_z{}^{(t)} = \operatorname*{argmax}_{p_z \in \mathcal{P}_z^{\alpha}} \mathbb{E}_{z \sim p_z} \left[\hat{L}^{(t)}(z; \theta^{(1:t)}) \right]$$

- The implementation is relatively easy.

Oren et al. 2019.

Is that all?

- This paper considers the **language modeling** task, instead of classification!
- Log loss can introduce imbalance between high-entropy groups and lower ones.



- Hu et al. 2018 pointed out the importance of loss function too.

Oren et al. 2019.

Modifying the Loss Term

New term!

- So they proposed another loss:

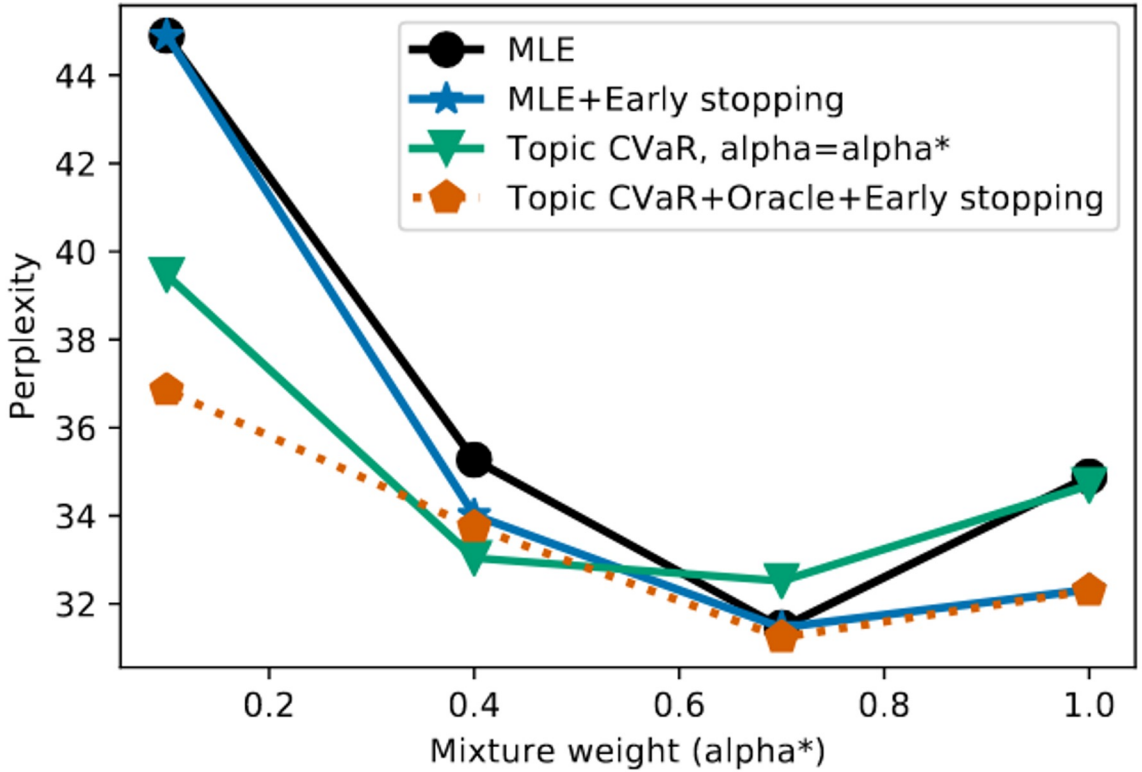
$$\sup_{p_{\mathrm{Z}} \in \mathcal{P}_{\mathrm{Z}}^{\alpha}} \mathbb{E}_{z \sim p_{\mathrm{Z}}} \left[\mathbb{E}_{x \sim p_{\mathrm{X}|z}} \left[\log p_{\mathrm{X}|z}\left(x \mid z\right) - \log p_{\theta}(x) \right] \right]$$

- They compared the loss within the group, which really set the group as the granularity of analysis.

- For estimating this new term, they use a simple bigram model for each group. (“ensemble of weak teachers”)

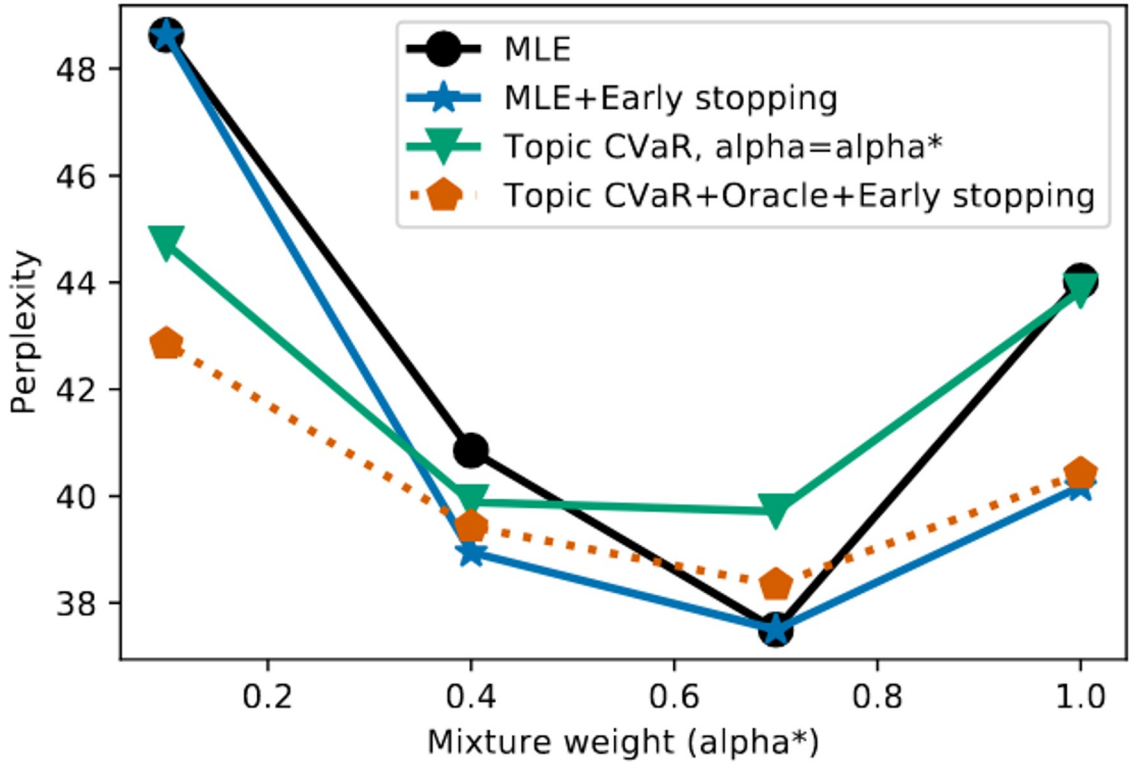
Oren et al. 2019.

Results on the Mixture of {YELP, ONEBWORD}



Oren et al. 2019.

Transfer on the TRIPADV Hotel Review



Oren et al. 2019.

Summary

- Contributions in algorithm design:

  (1) Extended the CVaR to a more realistic group version.

  (2) Modified the loss term to adapt it to language modeling.

  (3) Proposed a new algorithm to solve the Group DRO.

- Limitation: The two-player minimax game can be highly unstable during training, and there is no convergence guarantee. (Like the Generative Adversarial Network!)

DISTRIBUTIONALLY ROBUST NEURAL NETWORKS FOR GROUP SHIFTS: ON THE IMPORTANCE OF REGULARIZATION FOR WORST-CASE GENERALIZATION

Shiori Sagawa*
Stanford University
ssagawa@cs.stanford.edu

Pang Wei Koh*
Stanford University
pangwei@cs.stanford.edu

Tatsunori B. Hashimoto
Microsoft
tahashim@microsoft.com

Percy Liang
Stanford University
pliang@cs.stanford.edu

Problem Setting

- Group DRO for classification, but there is no restriction on alpha.
- They consider the **generalization problem** of overparameterized neural networks, and groups may differ in generalization error.

$$\hat{\theta}_{\text{DRO}} := \arg\min_{\theta \in \Theta} \left\{ \hat{\mathcal{R}}(\theta) := \max_{g \in \mathcal{G}} \mathbb{E}_{(x,y) \sim \hat{P}_g} [\ell(\theta; (x, y))] \right\}$$

Sagawa et al. 2020.[14]

Poor Generalization on Some Groups

			Average Accuracy		Worst-Group Accuracy	
			ERM	DRO	ERM	DRO
Standard Regularization	Waterbirds	Train	100.0	100.0	100.0	100.0
		Test	97.3	97.4	60.0	76.9
	CelebA	Train	100.0	100.0	99.9	100.0
		Test	94.8	94.7	41.1	41.1
	MultiNLI	Train	99.9	99.3	99.9	99.0
		Test	82.5	82.0	65.7	66.4

Sagawa et al. 2020.[15]

Method 1: Adding Strong Regularizers

			Average Accuracy		Worst-Group Accuracy	
			ERM	DRO	ERM	DRO
Strong ℓ_2 Penalty	Waterbirds	Train	97.6	99.1	35.7	97.5
		Test	95.7	96.6	21.3	84.6
	CelebA	Train	95.7	95.0	40.4	93.4
		Test	95.8	93.5	37.8	86.7
Early Stopping	Waterbirds	Train	86.2	80.1	7.1	74.2
		Test	93.8	93.2	6.7	86.0
	CelebA	Train	91.3	87.5	14.2	85.1
		Test	94.6	91.8	25.0	88.3
	MultiNLI	Train	91.5	86.1	78.6	83.3
		Test	82.8	81.4	66.0	77.7

- Comparison with H. Shimodaira. 2000.: Misspecification is bad…?

Sagawa et al. 2020.[16]

Method 2: Group Adjustments

- **Group-adjusted DRO estimator** inspired by generalization theory:

$$\hat{\theta}_{\text{adj}} := \arg\min_{\theta \in \Theta} \max_{g \in \mathcal{G}} \left\{ \mathbb{E}_{(x,y) \sim \hat{P}_g} [\ell(\theta; (x, y))] + \frac{C}{\sqrt{n_g}} \right\}. \tag{5}$$

The scaling with $1/\sqrt{n_g}$ reflects how smaller groups are more prone to overfitting than larger groups,

	Average Accuracy		Worst-Group Accuracy	
	Naïve	Adjusted	Naïve	Adjusted
Waterbirds	96.6	93.7	84.6	90.5
CelebA	93.5	93.4	86.7	87.8

Sagawa et al. 2020.[17]

Comparison with Importance weighting

$$\hat{\theta}_w := \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y,g)\sim\hat{P}}[w_g \, \ell(\theta; (x, y))]$$

- Importance weighting and DRO can learn equivalent models in the convex setting under some importance weights.
- But not necessarily equal when the models are non-convex.

- Counterexample provided in the paper.

New Approach of DRO

- This algorithm has convergence guarantee, and is more stable.

Algorithm 1: Online optimization algorithm for group DRO

Input: Step sizes η_q, η_θ; P_g for each $g \in \mathcal{G}$

Initialize $\theta^{(0)}$ and $q^{(0)}$

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

- $g \sim \text{Uniform}(1, \ldots, m)$ // Choose a group g at random
- $x, y \sim P_g$ // Sample x, y from group g
- $q' \leftarrow q^{(t-1)}$; $q'_g \leftarrow q'_g \exp(\eta_q \ell(\theta^{(t-1)}; (x, y)))$ // Update weights for group g
- $q^{(t)} \leftarrow q' / \sum_{g'} q'_{g'}$ // Renormalize q
- $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_\theta q_g^{(t)} \nabla \ell(\theta^{(t-1)}; (x, y))$ // Use q to update θ

end

Sagawa et al. 2020.

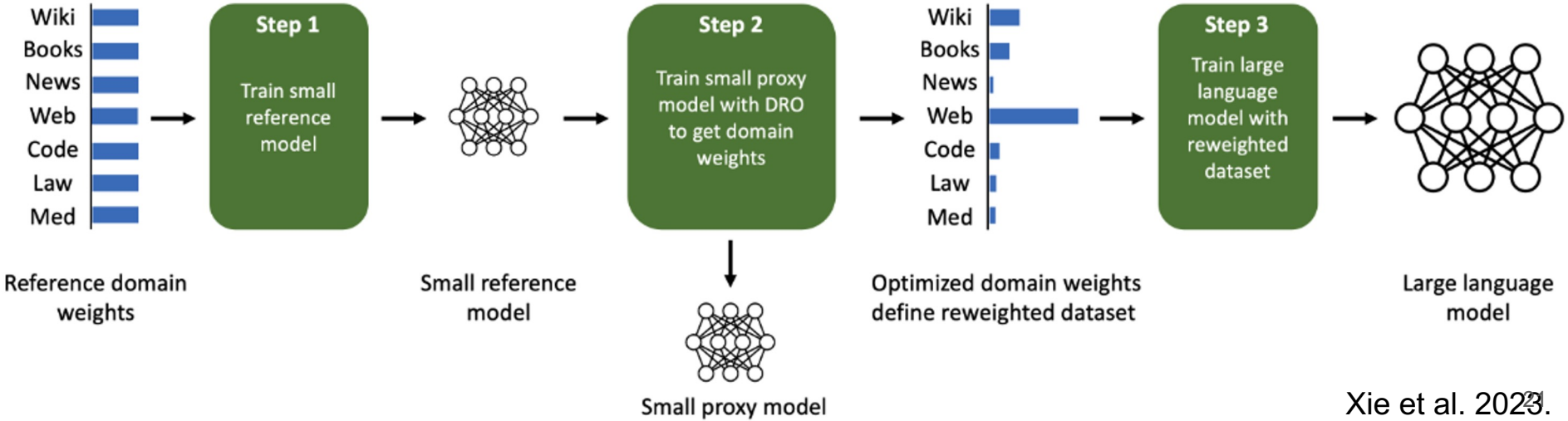
DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining

Sang Michael Xie[*1,2], Hieu Pham[1], Xuanyi Dong[1], Nan Du[1], Hanxiao Liu[1], Yifeng Lu[1], Percy Liang[2], Quoc V. Le[1], Tengyu Ma[2], and Adams Wei Yu[1]

[1]Google DeepMind
[2]Stanford University

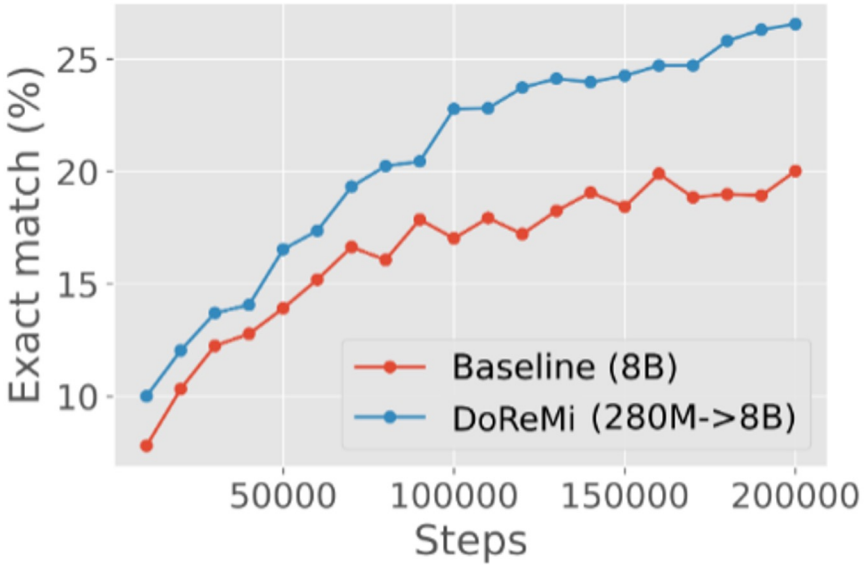
Pipeline

- Goal: Determine the weights for different sources (of Pile and GLAM).
- Using Oren et al. 2019's framework with Sagawa et al. 2020's optimizer to obtain weights for different domains.

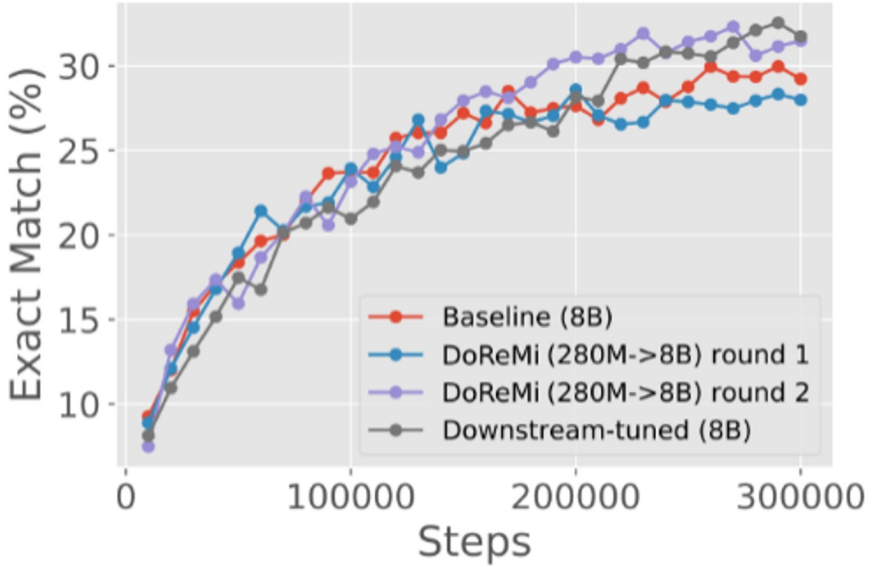


Xie et al. 2023.

Results



(a) The Pile



(b) GLaM dataset

Xie et al. 2023.

Results

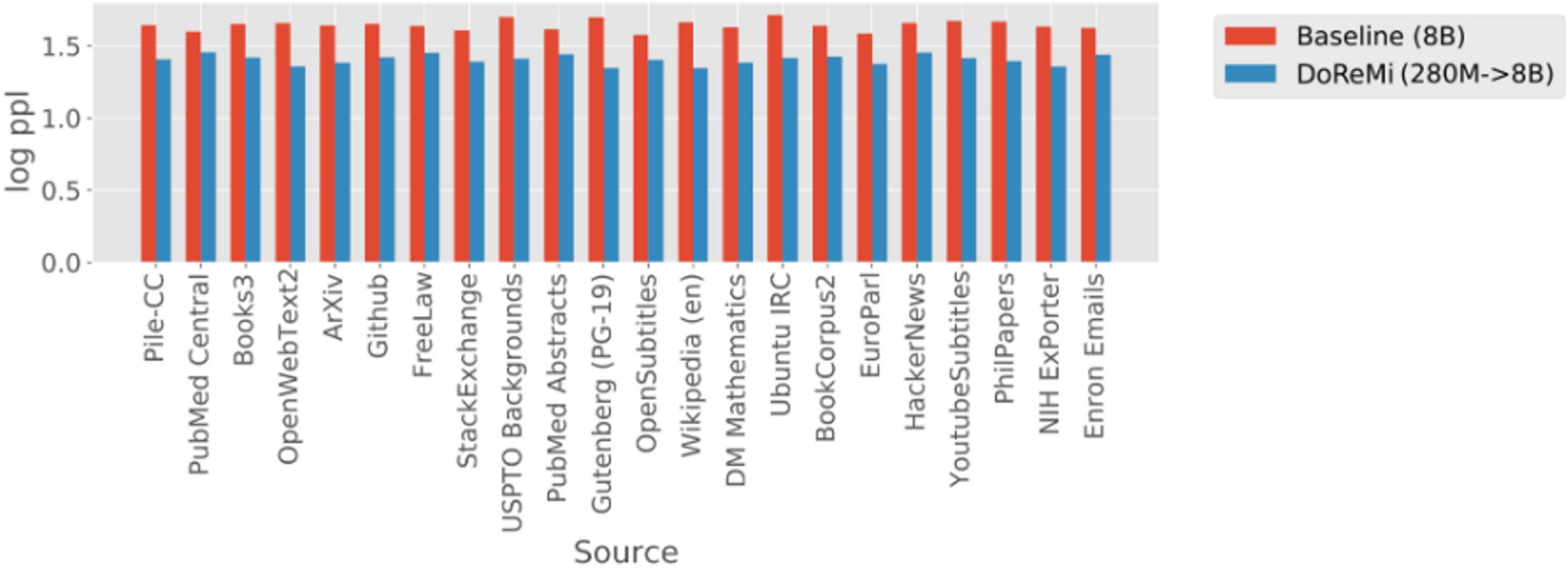


Figure 4: Per-domain log-perplexity of 8B models on The Pile. Despite downweighting some domains, DoReMi improves log-perplexity on all domains.

Xie et al. 2023.

References

A. Ben-Tal, D. den Hertog, A. D. Waegenaere, B. Melenberg, and G. Rennen. 2013. Robust solutions of optimization problems affected by uncertain probabilities. Management Science, 59:341–357.

J. Duchi, P. Glynn, and H. Namkoong. 2016. Statistics of robust optimization: A generalized empirical likelihood approach. arXiv.

H. Namkoong and J. Duchi. 2016. Stochastic gradient methods for distributionally robust optimization with f-divergences. In Advances in Neural Information Processing Systems (NeurIPS).

W. Hu, G. Niu, I. Sato, and M. Sugiyama. 2018. Does distributionally robust supervised learning give robust classifiers? In International Conference on Machine Learning (ICML).

Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. Distributionally robust language modeling. In Empirical Methods in Natural Language Processing (EMNLP), 2019.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In International Conference on Learning Representations (ICLR), 2020.

R. T. Rockafellar and S. Uryasev. 2000. Optimization of conditional value-at-risk. Journal of Risk, 2:2141.

H. Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference, 90:227–244.

Xie, S. M., Pham, H., Dong, X., Du, N., Liu, H., Lu, Y., Liang, P., Le, Q. V., Ma, T., & Yu, A. W. (2023). DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining. ArXiv. /abs/2305.10429

Discuss Questions

Improving predictive inference under covariate shift by weighting the log-likelihood function:

- The paper argues that if we don't have model misspecification, even if P_S and P_T are different, as long as we have P(y|x) are the same, the model can generate accurate prediction. Does this contradict to the distributional shift argument we previously saw?
- In the empirical experiment, sometimes we see that the loss of weighted dataset is greater than unweighted dataset, so weighing is completely useless. What might be the cause of this problem?

Distributionally Robust Language Modeling:

- What are the possible trade-offs between worst-group accuracy and other performance metrics of the model?
- Apart from assigning weights for different domains, what are some other possible advice from these papers for LLM training?