AI Alignment proposal №5: Robustifying AI Systems Against Distributional Shift

aialignmentproposals.substack.com/p/ai-alignment-proposal-5-robustifying

Abstract

Distributional shift poses a significant challenge for deploying and maintaining AI systems. As the real-world distributions that models are applied to evolve over time, performance can deteriorate. This article examines techniques and best practices for improving model robustness to distributional shift and enabling rapid adaptation when it occurs.

Techniques and practices for improving model robustness

Distributional shift, where test data differs from the distributions models were trained on, is an inevitable phenomenon when deploying machine learning systems to the real world. Data distributions naturally evolve over time — consumer preferences change, new data collection processes are used, populations shift. This can severely degrade model performance if not addressed.

Several approaches can help improve robustness and adaptability:

Causality-Based Modeling for Invariance

Causality-based modeling aims to incorporate causal assumptions about relationships between variables into the model topology and training process. This enables constructing representations invariant to certain distributional shifts, leading to more robust models.

A key technique is invariant risk minimization (IRM). First, hypothesized causal graphs are defined that capture assumed causal relationships between variables. Then, specific invariance criteria are formalized — e.g. predictions should be invariant to shifts in a certain input variable.

Based on this, data is split into environments exhibiting different distributions. The model is trained to satisfy the invariance criteria across these environments via meta-learning. A regularization term is added to the loss function that penalizes model parameters that violate the desired invariance.

For example, consider predicting loan default. The model could be regularized to keep predictions invariant even as the distribution of applicant age shifts across environments. This constructs a robust representation aligned with the causal assumption that age alone does not cause default.

Architectural choices can also encourage invariance. Convolutional neural nets exhibit translation invariance. Causal convolutions extend this for hypothesized causal relationships beyond space. Models can also be structured for hierarchical composition of invariant representations.

IRM requires formalizing the shifts of interest and causal assumptions. If done judiciously, however, it provides a principled way to train models robust to distributional shifts they will encounter during deployment. Models learn to ignore spurious correlations and focus on stable causal patterns.

Details on implementing invariant risk minimization (IRM) to improve robustness to distribution shifts:

1. Formulate causal assumptions as a causal graph. Connect input variables to outputs using directed edges representing hypothesized causal relationships and mechanisms.

2. Formalize desired invariance criteria based on the causal graph. For example, predict loan default invariantly across applicant age groups.

3. Split training data into environments exhibiting different distributions per the invariance criteria. For the loan example, environments could be age groups.

4. Add an invariant risk term to the loss function that penalizes model parameter differences across environments. Measure differences using a metric like KL divergence.

5. Train the model end-to-end on the environments with the augmented loss function encouraging invariant representations.

6. For convolutional neural nets, define causal convolutions where kernel weights are shared for certain hypothesized invariant relationships.

7. Validate invariance by checking for prediction consistency across synthesized counterfactual distributions in each environment.

8. Operationally, detect the environment/distribution at inference time and activate the specialized submodel trained to be invariant for that distribution.

9. Monitor inference-time metrics across environments and retrain model as needed to maintain invariance criteria.

The keys are formally defining distributional shifts of interest, encoding causal assumptions into model topology and training, and validating/enforcing achieved invariance. This approach aligns models with causal mechanisms to improve reliability despite shifting spurious correlations.

Continual Learning for Distribution Shift

Continual learning, also known as incremental learning, involves updating model parameters continually as new data arrives rather than retraining from scratch on large batches. This enables efficiently adapting models to shifting distributions.

A core challenge is avoiding catastrophic forgetting of previous knowledge when learning on new data distributions. Regularization techniques constrain training to preserve important parameters:

- Elastic weight consolidation identifies parameters critical for old tasks and aggressively regularizes them during new training. This prevents overriding parameters that encode prior distributions.
- Experience replay mixes a small ratio of old training data into new batches. The model is trained on this composite batch, preventing drift on old distributions. Curriculum learning can gradually change the ratio.
- Momentum-based regularization uses the momentum from parameter updates on old distributions to stabilize training on new data.

Additional techniques like rehearsal and dual-memory models also retain knowledge of prior distributions.

For natural language models, retrieved context training augments fine-tuning on new text with hidden layer representations from the original model. This provides a memory of old distributions to regularize fine-tuning.

Careful hyperparameter tuning to balance plasticity on new distributions and stability on old is key. But continual learning can enable models to efficiently and non-disruptively adapt as world distributions shift.

Details on implementing continual learning to adapt models to distribution shifts:

1. Start with a base model trained on diverse data to encode general knowledge. Use a neural network architecture suited for transfer learning.
2. Deploy the model to make predictions on new incoming data. Track performance metrics on this new distribution.
3. When metrics indicate a distribution shift causing degraded performance, sample new data to use for incremental training.

4. For experience replay, store subsets of old training data to mix into new batches. For retrieved context, store old model hidden states.

5. Introduce elastic weight consolidation or momentum regularization layers in the model architecture. These constrain parameter changes during training.

6. Train the model incrementally on batches containing new and old data. Gradually increase the ratio of new to old.

7. Tune regularization hyperparameters until convergence and metrics on new data improve while old data metrics remain stable.

8. Repeat the incremental training periodically as model metrics indicate the need for adaptation. Expand the stored old data cache over time.

9. Evaluate whether catastrophic forgetting is occurring after each round of training. Re-amplify regularization if needed.

10. For operationalization, the old data memory can be checked at inference to determine which version of the model to use on an input.

The main implementation requirements are instrumenting and monitoring the deployed model, efficiently storing old training data, and configuring the model for constrained incremental training. This enables responsive adaptation with limited compute and data.

Hybrid Global-Local Modeling

A powerful technique for adapting to distributional shift is combining a global model trained on diverse data with local models specialized for new distributions. The global model provides overall coverage, while local models adapt and improve performance on shifted data slices.

The global model can be a large foundational model pretrained on broad heterogeneous data at a high computational cost. This model aims to encode general knowledge and representations. Local models are then tailored to specific application domains or geographic regions using limited data from those distributions.

When a distribution shift emerges in a domain, a specialized local model can be rapidly retrained or transferred from the global model using techniques like model finetuning. Since the local model inherits general knowledge from the global model, only a small dataset of the new distribution is needed for adaptation.

Operationally, the global model handles common cases across domains. Inputs detected as distributionally shifted get routed to their respective local model for specialized handling. The local models essentially act as experts focused on new distributions. Their lightweight nature allows quickly swapping models in and out as shifts occur.

This global-local approach balances generalizability with efficient adaptability. The global model avoids retraining on all new data, while local models provide targeted adaptation. Hybrid modeling therefore enables responding to distribution shifts in a scalable and computationally efficient manner.

Details on practically implementing a global-local modeling approach to handle distribution shifts:

1. Train a large general-purpose global model on diverse data covering expected common cases. Use a high-capacity architecture like a transformer to encode broad representations. Pretrain on unlabeled data before fine-tuning if helpful.

2. Split incoming production data into domains/regions using metadata like geography, customer type, etc. Profile data distributions for each slice over a period to detect major shifts.

3. When a distribution shift emerges in a specific slice, extract a sample of new data to retrain a local model. Finetune a copy of the global model on this data or train a small specialized model with the global model's embeddings as input.

4. Route new query data points to the global model by default. Add a distribution detection component (e.g. based on input metadata or density estimation) to identify when inputs are from a shifted distribution.

5. For detected OOD inputs, pass them instead to the respective local model for inference. The local model can also tag its higher confidence predictions to further grow its training set.

6. Periodically profile performance per domain and retrain/update local models as needed. Control frequency to balance freshness and stability.

7. Manage local model versions, evaluate quality, and degrade then replace low-performing models. Maintain a manageable number of active models.

8. Monitor overall metrics and triggers retraining of global model if general performance drops. Finetune on a sampled subset of local model data for efficiency.

The key implementation requirements are a production-ready distribution detection component, infrastructure for low-latency routing/retrieval of the specialized models, and pipelines for constantly profiling, assessing, and updating both global and local models. The result is an adaptive system adept at handling evolving shifts across regions, customer bases, or other data slices.

Uncertainty Quantification for Detecting Distribution Shifts

When a model encounters inputs that are out-of-distribution (OOD) from its training data, its predictions will have higher uncertainty. Quantifying and exposing this uncertainty enables detecting when distributional shift is impacting the model.

Bayesian neural networks model weight distributions rather than point estimates. At inference, this provides a posterior predictive distribution capturing uncertainty. The variance of the distribution indicates OOD inputs where the model has lower confidence.

Dropout at test time also gives a distributional output as different nodes are dropped over multiple passes. Increased variance highlights uncertain predictions.

Ensembling trains multiple models on the same data. Disagreement between ensemble members on a new input signifies its difference from the training distribution.

Once uncertain inputs are identified, several handling approaches include:

- Flagging for human review to determine if the prediction is still adequate or not
- Routing the input to an alternate model specialized on the new distribution
- Using the uncertainty to update and improve the model by reweighting or adding this OOD data

Overall, modeling uncertainty makes distribution shift transparent. It enables pinpointing drops in model reliability and deploying appropriate interventions to improve robustness and adaptation.

Details on implementing uncertainty quantification to detect and handle distribution shifts:

1. Select an architecture that can represent uncertainty, like Bayesian NN or ensemble.
2. During training, validate that uncertainty is higher on out-of-distribution data compared to in-distribution data.
3. Instrument prediction service to capture uncertainty metrics like predictive variance, model disagreement, or mutual information on each inference request.
4. Set thresholds on uncertainty metrics to classify predictions as confident (in-distribution) or uncertain (out-of-distribution).
5. For uncertain predictions, log input and optionally send to human review queue to check quality.
6. Route uncertain inputs to an alternate model adapted for out-of-distribution data to see if it produces confident predictions.
7. Monitor rates of uncertain predictions to detect increases indicating distribution shift. Trigger retraining if uncertainty rises.
8. Capture high-uncertainty inputs and retrain models on this data to improve coverage of new distributions.

9. For Bayesian NNs, shift uncertain input inferencing to MC Dropout mode for better uncertainty estimates.

10. Analyze uncertainty metrics associated with features to determine which distributional changes are causing uncertainty.

The key requirements are instrumenting models for uncertainty capture, installing triggers and routing based on uncertainty thresholds, and leveraging uncertain data to guide model improvement. This enables uncertainty to drive adaptation to evolving distributions.

Monitoring and Automated Retraining

Continuously monitoring performance metrics provides signals when distribution shift is impacting model effectiveness. Metrics like accuracy, F1 score, precision/recall, etc. can be calculated on an updated test set representing the deployment distribution.

Significant drops in these metrics indicate a shift away from the training distribution. Thresholds can be defined to trigger automated retraining pipelines when metrics breach certain levels.

However, metric variance must be modeled to avoid oversensitive triggering. Temporary fluctuations or outliers shouldn't trigger retraining. The monitoring system must distinguish meaningful dips requiring adaptation.

Updating the test set periodically is also crucial to accurately reflect the latest deployment distribution. If the set becomes stale, shifts may not register in the metrics.

Access to sufficient data from the new distribution is needed for retraining. Data augmentation techniques like SMOTE can synthesize additional representative points. Transfer learning fine-tunes models on small new datasets.

Overall, monitoring provides a scalable way to determine when distribution shift necessitates adaptation. Automated triggering then executes pipelines to efficiently refresh models. Along with data synthesis and transfer learning, this workflow enables continuously realigning models with evolving distributions.

Details on implementing monitoring and automated retraining for distribution shift adaptation:

1. Maintain a representative test set sampling current deployment data distribution. Refresh periodically, e.g. monthly.

2. Instrument model predictions to capture key performance metrics like accuracy, F1 score, etc. on an ongoing basis.

3. Calculate metrics on test set data. Model metric variability to determine stable thresholds for retraining triggers.

4. Set up monitoring dashboard visualizing metrics over time, trends, and comparisons to thresholds. Alert on threshold breaches.

5. When threshold breach detected, trigger retraining pipeline:

- Sample new data from monitoring system to transfer learn model. Synthesize additional data if needed.
- Load base model architecture and weights. Retrain on new dataset with tight regularization to avoid catastrophic forgetting.
- Calculate metrics on new test set. If improved, replace model in production. If not, reassess thresholds.

1. For efficiency, pipeline can start with shallow retraining focused only on later layers before doing full retraining.
2. Maintain versioned records of model parameters before and after retraining to enable reverting or barging if issues emerge.
3. Monitor system in production to verify metrics improve and thresholds are set appropriately.

The key requirements are creating frameworks for continuous monitoring, modeling metric variability, developing retraining pipelines, and validating retrained models before deployment. This enables closing the loop on monitoring, automated adaptation, and improved robustness.

Counter-arguments

Counter-argument:

Some argue that continuously retraining on new data is sufficient to address distribution shift. However, this can be computationally expensive and data hungry. Alternative approaches help minimize retraining needs. It has also been posited that the best solution is to expand training data diversity upfront. But anticipating all shifts is infeasible, so adaptivity remains imperative.

Rebuttal:

A multi-faceted approach combining robust modeling, adaptation techniques, and data collection is ideal. Relying solely on expansive data diversity or continuous retraining is likely insufficient and inefficient. The techniques discussed provide complementary ways to both minimize the impacts of distribution shifts and rapidly adapt models when necessary.

Counter-argument:

Some argue that trying to proactively robustify models against distribution shift is ineffective, and that continuously retraining models from scratch on new data is the best approach. They contend that techniques like causal modeling or uncertainty quantification add unnecessary complexity for marginal improvements in adaptivity.

Rebuttal:

While retraining on new data is an important part of adapting to distribution shifts, solely relying on retraining has downsides. Large batched retraining on fresh data can be computationally expensive and time consuming, causing lags in adapting models. Data collection itself can be costly. The proposed techniques offer complementary benefits such as improved resource efficiency, reduced data requirements, and the ability to flag prediction uncertainties. Used judiciously, they provide pragmatic ways to balance adaptability and practical constraints. However, further research identifying optimal combinations of these techniques is certainly warranted.

Conclusion

To create viable real-world AI systems that are robust to evolving data distributions, a proactive approach is needed. Techniques like causality-based modeling, uncertainty quantification, online learning, and monitoring of deployment metrics each contribute complementary benefits for handling distribution shift. Used together, they enable continuously adapting models in an efficient, targeted manner as new data emerges. The expanded discussion in this article provides concrete details on implementing these techniques to minimize performance degradation from distribution changes. Their real-world efficacy can be further refined through ongoing research and testing. However, this combination of approaches provides a pragmatic way forward for developing AI systems that gracefully handle shifting data distributions while balancing practical constraints. Continued advancement in this domain remains crucial for enabling reliable and stable model performance despite the inevitability of distributional changes in real deployment environments.