Study of different approaches to Out of Distribution Generalization

---

YOUBI IDRISSI Badr

HOUDOUIN Pierre

AYADI Imen

HACHICHA Mohamed Amine

2020-2021

Contents

1 Introduction

Humans have an outstanding ability to generalize. The brain is often able to correctly predict outcomes in scenarios it has never witnessed and that are substantially different to the ones it is used to. In other words, it can extrapolate well beyond the scope of its known and memorized experiences. Deep learning models on the other hand, perform well when the new data is within the scope of its training. When faced with novel examples that are significantly different, these models often struggle. These models excel at interpolating and only generalize well to examples that stay within the limits of the training data.

Two elegant frameworks to address this issues are causal inference and domain adaptation. They both intend to learn models capable of generalizing outside of the training distribution. However, they differ in their assumptions and points of view. This work will thus try to introduce both the notion of causality and domain adaptation.

1.1 Causality

'Cum hoc ergo propter hoc' or in English 'With this, therefore because of this'. This famous logical fallacy is taught in every introduction to statistics class to prevent false conclusions when analyzing data. We hear time and again that 'Correlation does not imply causation'. And, although we are taught various aspects of correlation, causation is often left on the sidelines.

Humans have an intuitive understanding of causality. It is how we try to make most decisions. For instance, one knows that atmospheric pressure causes rain to fall that then causes the floor to be wet. The correlation between any two of these events is symmetrical, but the causal link isn't. Let us denote X, Y, and Z three random variables that measure atmospheric pressure, how hard it rains, and how wet the floor is, respectively. Say we want to predict Y to decide whether or not to take an umbrella. If we have a high enough number of **independent, identically distributed** samples of these variables, we will notice a high positive correlation between X (atmospheric pressure) and Y (rain) but also between Z (wetness of floor) and Y (rain). What should we take into account when trying to decide whether or not to take an umbrella? Intuitively, one would answer that we should only take into account X the atmospheric

pressure. The reason is that X is a cause of Y. Therefore, we expect it to be a reliable variable to predict Y. We will see later how we can think of this more formally.

Statistical learning often makes this assumption that data samples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are a realization of **independent, identically distributed** random variables $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$. It then extracts the intricate statistical associations and dependencies between the variables in the data. These statistical dependencies can be summarized with $P_{X,Y}$ the joint probability of X and Y. With this joint probability, we can, for example, regress Y over X by finding the conditional expectation $E[Y|X = x]$. Or, classify a point x to the most probable value of y (in case it is discrete). However, the i.i.d assumption limits us to the case where we have the same underlying distribution for training and testing. Just knowing $P_{X,Y}$ doesn't inform us on what happens when the underlying distribution changes. In practice, however, data is rarely purely i.i.d. The training and testing distributions can be quite different. In that case, the i.i.d assumption can lead to poor performance. This is very often witnessed in deep learning. [1, 2, 3, 4, 5, 6] There are multiple scenarios with a rift between the performance of models on benchmarks vs their performance in real life. Benchmarks are often artificially made i.i.d by shuffling them. Indeed, shuffling the data then splitting it into train and test yields very close distributions. Sometimes, the performance on benchmarks suggests superhuman levels. But when the model is deployed, its accuracy quickly degrades. Is this just a case of lacking data? In the following, we argue that more data doesn't necessarily solve the problem. A paradigm shift is needed.

This report will first briefly present the causal framework in section 2. As we will see in section 3, the concept of invariance emerges naturally from this. Invariance motivates the Invariant Risk Minimization [7] method that is presented in section 3.2. Then we will move to studying IRM empirically with a simple concrete example in section 4. Afterwards, we will derive the expression of the IRM penalty in some special cases and find some useful upperbounds on it, along with an analysis of the limitations of IRM in section 5. Ideas for the improvement of IRM will be presented in section 6, along with experimental results.

1.2 Domain Adaptation

As stated previously, the i.i.d. assumption is very strong and can hurt learning when data comes from different environments. In this setup, the model risks to learn domain specific features which make it unable to generalize well to new domains. This is why Domain Adaptation is a widely studied sub-field of machine learning. The main common characteristic between Domain Adaptation methods and IRM is that both methods learn data representations that are not domain (or environment) specific, which allows us to discard spurious correlations for prediction tasks.

Many domain adaptation techniques can be identified in the literature. For instance, in [8] the authors introduce deep adaptation networks (DAN). In this setup, we assume that the conditional distributions of data along domains are the same and the DAN aims to align marginal distributions by introducing multiple adaptation layers that utilize a multiple kernel variant of MMD (Maximum Mean Discrepancies) to extract features and embed them into the corresponding RKHS.

We can also find in the literature other domain adaptation paradigms that rely on autoencoders. The objective here is to learn invariant and transferable representations along different domains to minimize the reconstruction error. In the case where we have a source domain and a target domain, the encoder is trained on the source domain and then the decoder is trained to minimize the reconstruction error on the target domain. For example, in [9] the authors make use of Stacked Denoising Autoencoders (SDA) to learn high-level features of data to represent both source and target

domain observations in an unsupervised way.

In our project we will consider two other approaches in domain adaptation : Adversarial Domain Adaptation Networks which we will explore in section 7 and Adversarial Information Factorization which will be studied in section 8.

2 Causal Inference

Causal Inference is a big area of research that needs entire monographs for a full formal description. We refer the interested reader to the following: The book of why [10] for a less formal read on causality. Causality [11] for a more involved mathematical treatment. And finally, Elements of Causal Inference [12] for a more recent text. This short paragraph in contrast reflects our current understanding and the necessary elements needed for the rest of this report. Most of the ideas presented in this section come from the books cited above.

Let us reconsider the example taken in the introduction. It is best represented with the graph in figure 1. The nodes are random variables and the edges represent the causal relationship between the variables. It is a directed edge since cause and effect are asymmetrical. Consider the following :

$$Y := f(X, N_1) \text{ and } Z := g(Y, N_2)$$

These relations define the structural equation model (SEM) underlying the data. Where N_1 and N_2 are noise variables such that $N_1 \perp\!\!\!\perp N_2$. These equations are to be understood as assignments. So should the distribution of X, N_1 change, Y would still be $Y := f(X, N_1)$. For example if we fix X to the constant 1 then we would have $Y = f(1, N_1)$. This describes the mechanism with which the variables are generated. One key idea behind this is that the mechanism that generates each variable, is independent of the distribution of the input variables.

Figure 1: Graphical model of X, Y, Z as presented in the introduction

The parents in the graph are the direct causes of each variable. Once X is known, knowing Z won't add any information about Y. We say that Y is conditionally independent of Z given X. So as long as the relation between Y and its parents doesn't change, we can reliably predict Y given X even if the rest of the graph changes. The graph is thus modular. Interventions on one part of the graph don't necessarily affect the whole graph.

To get back to our example, say we collect data in two cities. In the first city, it rains quite often. In this city, we never see a wet floor except when it rains. In the other city, it rains much less. Therefore, people need to water their plants. It happens in this case that the floor is wet without the presence of rain. The data in these two cities come from different but related distributions. For instance, the probability of atmospheric pressure changes from one city to another. We could also have the apparition of a new variable representing the need of the citizens to keep their plants alive. (See figure 2) This in turn, changes the distribution of Z. But all of these interventions don't affect the relation between X and Y. Causal relationships seem to have this **invariance** property with respect to interventions. Since in one of the cities making the floor wet didn't provoke rain, we can conclude that the dependence between the wetness of the floor and the rain is varying. Therefore we can eliminate it from the causes of the rain.

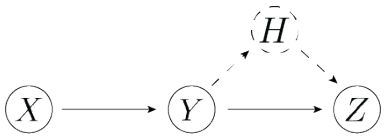


Figure 2: Graphical model of X, Y, Z as presented in the introduction

Let us consider the simple linear example presented in [7] that will be the basis of some of the experiments later on. It also serves as a motivating example. We consider the following simple structural equation model :

$$X := \mathcal{N}(0, \sigma^2) \tag{1}$$

$$Y := X + \mathcal{N}(0, \sigma^2) \tag{2}$$

$$Z := Y + \mathcal{N}(0, 1) \tag{3}$$

Here, we want to predict Y when given X and Z as input. [7] consider a case where the data comes from different environments e. Each has a different σ standard deviation of the noise. It might help to think of these environments as the different cities from which the data comes. We have a closed form solutions for the linear regression $Y^e = \alpha_1 X^e + \alpha_2 Z^e$. [7] The derivation is detailed in Appendix A. We have three cases :

- If we regress Y over X we would have $\hat{\alpha_1} = 1$ and $\hat{\alpha_2} = 0$
- If we regress Y over Z we would have $\hat{\alpha_1} = 0$ and $\hat{\alpha_2} = \frac{\sigma^2}{\sigma^2 + 1/2}$
- If we regress Y over X and Z we would have $\hat{\alpha_1} = \frac{1}{\sigma^2+1}$ and $\hat{\alpha_2} = \frac{\sigma^2}{\sigma^2+1}$

The only case where the regression coefficents don't depend on the environment is the first one. It corresponds to the invariant feature X. Equation 3 in the structural equation model doesn't affect the relation between Y and its parents. It is therefore conceivable that it can change. For instance, in a test environment, Z can be arbitrary. If a tsunami hits a city, the floor will be very wet without the necessary presence of rain. One can therefore have an arbitrarily large expected risk by driving Z to $+\infty$ So $\hat{\alpha_1} = 1$ and $\hat{\alpha_2} = 0$ is optimal when one allows interventions that don't affect Y with relation to its parents. Minimizing the empirical risk over many environments will generally yield solutions that depend on the environment. And that is no matter the amount of data that we put in. So we can always find a way to break the learned classifier or regressor. If we want to generalize well in these cases, we need to search for invariant solutions.

3 Causal Discovery through Statistical Invariance

The previous simple example illustrates how optimal statistical solutions (regressing over both X and Z) are only optimal for a fixed joint distribution. It also shows one aspect that could help us find solutions that will generalize to different but causally related distributions. This key aspect is invariance. As we saw, the optimal minimax solution was the one that didn't depend on the environment.

3.1 Invariant Causal Prediction

Invariant Causal Prediction[13] is a statistical framework that formalizes this idea in the case where there are no hidden variables. They try to select a subset of the input features that are causal with respect to the output.

Given $e \in \mathcal{E}$ an environment and $X^e = (X_1^e, \dots, X_d^e)$ the predictor variables that we observe, we want to predict Y^e. ICP introduces the following hypothesis :

$$H_{0,S}(\mathcal{E}) : \begin{array}{l} \text{There exists } g : \mathbb{R}^{|S|} \times \mathbb{R} \to \mathbb{R} \text{ and } F_\varepsilon \\ \text{such that for all } e \in \mathcal{E} \quad Y^e = g(X_S^e, \varepsilon^e) \\ \text{with } \varepsilon^e \sim F_\varepsilon \text{ and } X_S^e \perp\!\!\!\perp \varepsilon^e \\ \text{where } X_S^e = (X_i^e)_{i \in S} \end{array}$$

This hypothesis tells us that there exists a subset of features that can be used to predict Y^e independently of e. We assume that there is a true *causal* set S^* for which $H_{0,S^*}(\mathcal{E})$ is true for all possible environments $\mathcal{E}$. If $H_{0,S}(\mathcal{E})$ is true for a set $S \subseteq \{1, \dots d\}$, Y_e is a function of these variables that doesn't depend on e. Therefore these variables in S are invariant with respect to $\mathcal{E}$. If $H_{0,S}(\mathcal{E})$ is true for a $\mathcal{E}$ that we currently have access to, maybe it is for a larger $\mathcal{E}'$. Therefore, the variables in S are good *causal* candidates.

$$S(\mathcal{E}) = \bigcap_{S : H_{0,S}(\mathcal{E}) \text{ is true}} S$$

By construction $S(\mathcal{E}) \subseteq S^*$. When $\mathcal{E}$ grows, there are less sets S for which $H_{0,S}(\mathcal{E})$ is true. $S(\mathcal{E})$ then shrinks around S^*. This reflects the fact that with more data from different environments, we can better identify the true causal predictors S^*

ICP estimates $S(\mathcal{E})$. To do that, it develops hypothesis tests for $H_{0,S}(\mathcal{E})$. Then it keeps the intersection of all the sets that the hypothesis test doesn't reject. The proposed tests are of two kinds :

1. Regress Y^e on X^e on each e. Then test if the resulting functions are the same.
2. Regress Y on X (the whole data). Then test if the residual errors on each e are the same.

This approach introduces important concepts, but it remains impractical when faced with many input features as it scales poorly. It also doesn't take into account hidden variables. For example, in computer vision, there are too many input features for this method to be practical. And there are necessarily hidden variables that we don't directly observe such as the position of different objects.

3.2 Invariant Risk Minimization

3.2.1 Presentation of the framework

Another notable method that is inspired from the previous one and that this work will try to improve on is Invariant Risk Minimization [7]. This method also utilizes the availability of a set of training environments $\mathcal{E}_{tr}$. It will use the subtle changes in dependence in these environments to detect spurious features that might be very correlated with the target variable but won't generalize well. For example, when using the wetness of the floor to predict rain. To do this, they take a different route than ICP. Instead of explicitly selecting among the input features, they try to build an invariant representation Φ (potentially nonlinear).

In deep learning, one minimizes an emperical risk : $R(f) = E[\ell(f(X), Y)]$ where ℓ is a loss function (cross entropy or square error for example). When there are multiple environments, the risk also depends on the environment $R^e(f) = E[\ell(f(X^e), Y^e)]$. Invariant Risk Minimization splits f into two parts, a representation Φ and a classifier w such that $f = w \circ \Phi$. Where $\Phi : \mathcal{X} \to \mathcal{H}$ and $w : \mathcal{H} \to \mathcal{Y}$. Invariance in IRM is then defined as the following :

$$\forall e \in \mathcal{E}_{tr} \quad w^* = \arg\min_{w} R^e(w \circ \Phi) \tag{4}$$

Let us clarify the motivation behind this definition of invariance. We want Φ to be such that predicting Y^e given $\Phi(X^e)$ doesn't depend on the environment e. In mean squared regression, $h \to E[Y^e|\Phi(X^e) = h]$ is the function that best approximates Y^e given X^e, by definition of the conditional expectation. If Y^e is a binary variable taking values 0 or 1, then $P(Y^e = 1|\Phi(X^e) = h) = E(Y^e|\Phi(X^e) = h)$. Therefore the optimal classifier with logistic regression is $w = P(Y^e = 1|\Phi(X^e) = h) = E(Y^e|\Phi(X^e) = h)$. In these two cases (logistic and MSE regression), when Φ is invariant we have that for all h in the intersection of the supports of $\Phi(X^e)$.

$$\forall e \in \mathcal{E}_{tr} \quad E[Y^e|\Phi(X^e) = h] = E[Y^{e'}|\Phi(X^{e'}) = h] \tag{5}$$

Equation 5 tells us that the best prediction of Y^e given $\Phi(X^e)$, doesn't depend on the environment e. This idea is in turn motivated by the invariance and independence of causal mechanisms as presented in section 2. For instance, if some latent variables Z cause Y, then as we saw in section 2, for any intervention that doesn't directly affect the causal mechanism between Z and Y, the conditional distribution $P(Y^e|Z^e)$ will stay the same. Where the environments e correspond to different interventions on the causal graph. A variable S such that $P(Y^e|S^e)$ changes can thus be directly detected as non causal.

We are searching for a representation Φ that gets rid of non stable features from X^e when e changes. To achieve that, IRM proposes to solve the following bilevel optimization problem :

$$\begin{array}{ll} \min_{\Phi:\mathcal{X}\to\mathcal{H}} & \sum_{e\in\mathcal{E}_{\text{tr}}} R^e(w \circ \Phi) \\ \text{subject to} & w \in \underset{\bar{w}:\mathcal{H}\to\mathcal{Y}}{\arg\min} R^e(\bar{w} \circ \Phi), \text{ for all } e \in \mathcal{E}_{\text{tr}}. \end{array}$$

3.2.2 Simplification of the objective

This is a challenging optimization problem that can hardly be solved in practice. Therefore, they transform it into a relaxed version that is more practical to solve. Another issue is that this problem is ill-posed. There are many Φ and w that represent the same final solution. It suffices to take an invertible mapping Ψ and we get $\Phi' = \Psi^{-1} \circ \Phi$ and $w' = w \circ \Psi$ that is also optimal if Ψ and w are optimal. To get around this, one can fix any w and let Φ change such that w becomes optimal for all environments. This is possible since Φ is an arbitrary mapping. Φ is in a sense absorbing the responsibility of the classifier. w would then be just a dummy classifier made solely for the purpose of measuring the invariance of Φ. Since we can fix w to any function, we might as well make it a linear mapping. Additionally, If the loss function $x \to \ell(x, y)$ is convex then $w \to \ell(w \circ \Phi(X^e), Y^e)$ is also convex as the composition of an affine function with a convex function. Finally taking the expectation conserves convexity so $w \to R^e(w \circ \Phi)$ is also convex. In this case, we only need the first order condition $\nabla_w R^e(w \circ \Phi) = 0$ to be satisfied for w to be optimal for the environment e. Therefore for a fixed linear dummy classifier w the IRM problem is equivalent to :

$$\begin{array}{ll} \min_{\Phi:\mathcal{X}\to\mathcal{H}} & \sum_{e\in\mathcal{E}_{\text{tr}}} R^e(w \circ \Phi) \\ \text{subject to} & \nabla_w R^e(w \circ \Phi) = 0 \text{ for all } e \in \mathcal{E}_{tr} \end{array}$$

We can further simplify the problem by making it regularized instead of constrained.

$$\min_{\Phi:\mathcal{X}\to\mathcal{H}} \sum_{e\in\mathcal{E}_{\text{tr}}} R^e(w\circ\Phi) + \lambda\|\nabla_w R^e(w\circ\Phi)\|$$

With this formulation, [7] successfully train a linear representation on the problem presented in the introduction. They also successfully apply it to a colored MNIST dataset with spurious correlations between color and number. We will later in this report reproduce these results and provide an analysis of the invariance penalty, with its shortcomings and some ideas to improve on it.

4 Analysis of IRM on a concrete example

In this part, we want to highligh the benefits of IRM regression to identify spurious correlations in a machine learning model.

4.1 Introduction

Let us first reintroduce spurious correlations, causal variables and environments more intuitively.

4.1.1 Spurious correlations

A spurious correlation is an undesirable relation between an output variable and an input variable in the model.

For example, in Asia, a model was built using images of the crowd in a subway station to predict the optimal opening time of the metro's doors. After many big failures, especially during events occurring late in the night, it appeared that the model was simply looking at the clock to answer the problem. It explained the poor performances during special events occurring at unusual times. The relation between clock time and how long the doors should stay open was a spurious correlation, there was not a direct causal link, only a high correlation.

Although good performances could be achieved most of the times, the use of clock informations presented a major disadvantage : exporting the model to use it in another station with a digital clock would lead to very poor results. This is what we are focusing on when we use IRM. We want a model that can be used in other situations, other environments with still good results, a model that generalizes well.

4.1.2 Causal variables

To reach this goal, we want the model to focus only on the causal variables. A variable is causal if its influence on the result does not vary accross different environments. If the model only uses causal variables, it can then easily generalise to different environments.

4.1.3 Environments

In the IRM paradigm, all the data are labelled by a number indicating from which environment it was sampled. The goal is then to build a unique model to predict the output for all the different environments. To reach good perfor-

mances, the model will have to use only variables that influence in the same way the result accross the environments: the causal variables.

One can see IRM as a way to integrate the prior information of environments: which distribution the sample was drawn. In a classic model, we would probably not take this information into account and simply merge all the data and shuffle it.

4.2 Presentation of the example

4.2.1 The model

We consider a toy example focusing on predicting housing prices. We suppose in this example that we have available a dataset containg the following informations for each house :

- Number of rooms
- Size of the house
- Size of the garden
- Distance to the closest public transport

We also suppose that we almost have a linear relation between the price of the house and those four variables.

In our model, each house is sold by a real estate agency that is remunerated by taking fees calculated as a percentage of the price of the house. However, all agencies do not take the same percentage and even though they remain close, they are different. The fees and which agency sold the house are also available in the dataset. Thus, for each house, we have six features available :

- Number of rooms
- Size of the house
- Size of the garden
- Distance to the closest public transport
- Fees applied by the agency
- ID of the agency that sold the house

4.2.2 Link with IRM

Now that we have brought the different agencies in the model, we can make use of IRM.

All the houses sold by the same agency will be considered as coming from the same environment. The ID of the agency will thus be indicating from which environment the sample comes from.

The fees applied by the agency will be our spurious correlation. It will be very correlated with the house price, but will be a consequence of the price, and not the cause. A classic model that does not take the environment information into account is likely to use this feature.

The number of rooms, size of the house, size of the garden and distance to the closest public transport are our four causal variables. This is the variables that we want the model to use to be able to predict accuratly the price, no matter which agency sold the house.

4.3 Data generation

4.3.1 Input variables

We generate the six input features using the following formulas :

- Number of rooms : $\theta_1 \sim min\left(max((\mathcal{N}(15,3),1),30\right)$, there are in average 15 rooms per house, with a minimum of 1 and a maximum of 30.
- Size of the house : $\theta_2 \sim min\left(max((\mathcal{N}(180,40),10),1000\right)$, there are in average 180 m^2 per house, with a minimum of 10 and a maximum of 1000.
- Size of the garden : $\theta_3 \sim min\left(max((\mathcal{N}(80,20),0),5000\right)$, there are in average 80 m^2 per house, with a minimum of 0 and a maximum of 5000.
- Distance to the closest public transport : $\theta_4 \sim min\left(max((\mathcal{N}(1,0.3),0.1),20\right)$, houses are in average distant by 1 Km to public transports, with a minimum of 100m and a maximum of 20 Km.
- Fees applied by the agency : $\theta_5 \sim Price \times f(env)$, $f(env)$ is the percentage taken by the agency, constant for each agency (environment).
- Environment : $e \sim \mathcal{U}(\{1,2,3,4\})$, we suppose that there are four different agencies that can sell the houses, with equal probabilities.

Also, in order to highligh the importance of the model's ability to generalize accross the different environments, we suppose that the houses present in the training set can only come from the first three environments, whereas the houses present in the test set can come from all the four environments. Finally, to punish heavily the use of the spurious feature, that is the fees (or commissions) taken by the agency, we will take a very different percentage for the fourth agency only present in the test set.

Of course, everything is done to trap the classical model that will not take into account the environment informations. Indeed, the goal of this toy example is to illustrate the importance of invariance in some situations, and how we can get very surprising results when generalizing, even though we had very satisfying results so far.

4.3.2 Price of the house

As mentionned above, we aim to have a price that has almost a linear relationship with the four causal features. We start by defining the clean price as :

$$\text{Clean price} = 35,000\theta_1 + 7,000\theta_2 + 1,500\theta_3 - 100,000\theta_4$$

However, if we kept this price as the final price, with an exact linear relation between the price and the causal variables, the classical model would have the best results using only the four causal variables, since the commissions vary from one environment to another.

Yet, we want a situation where a naive model uses spurious features, to then highligh how IRM avoids it. Thus, we then add some random gaussian noise on the clean price, to obtain the final price, called also noisy price.

$$\text{Noisy price} = Cleanprice \times (1 + \mathcal{N}(0, \sigma^2))$$

Since the commission will be applied on the noisy price, a naive model now has interest in using the spurious feature to fit better the data and overcome the random noise added to the price.

4.3.3 Trade-off between the noise and the difference between the commissions

The value of σ and of the 4 different commissions have to be chosen carefully.

For a fixed σ, if the commissions are taken too different, θ_5 will no longer be a relevant feature for the classical model, as it will lead to huge errors if used. Thus, the classical model will simply use the four causal variables and accept to commit an error linked to addition of noise.

On the contrary, if the commissions are taken too close, IRM model might also get fooled and use the spurious features. Indeed, the IRM regressor simply adds a penalization in the loss when spurious features are used. Thus, if the change of influence on the result accross the different environments is small enough, the penalization will not be enough to remove the use of the spurious features.

After some calibration tests, we choose to keep the following values :

- Noise : $\sigma = 0.03$
- Fees : 4.9%, 5% and 5.1% for first three environments and 7% for the fourth.
- Penalization of spurious features : $\lambda = 1$

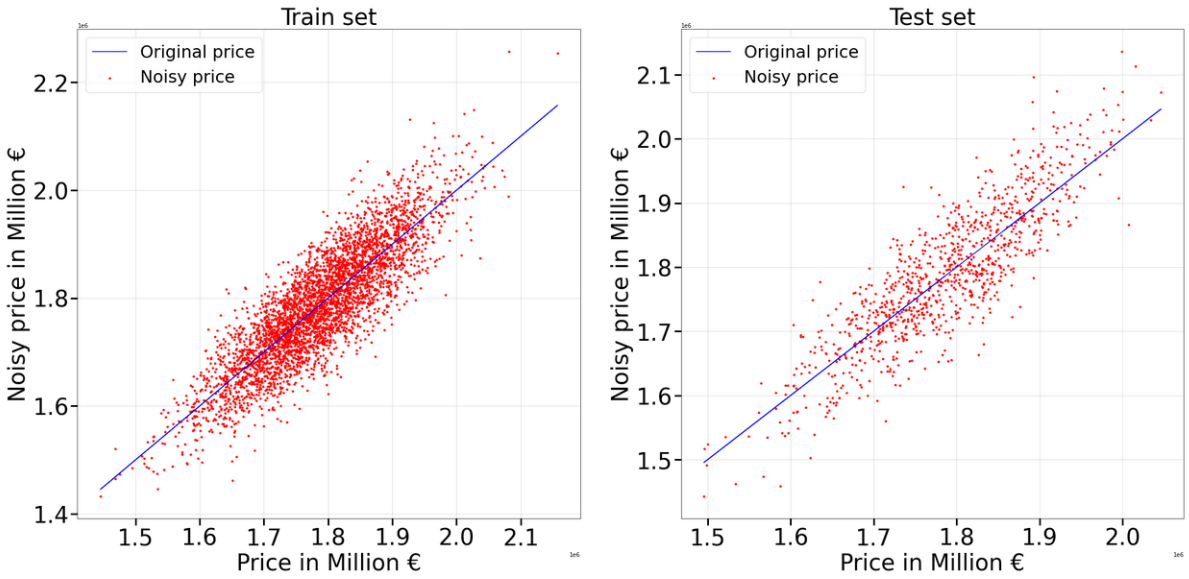


Figure 3: Dataset generated : in abscissa we have the clean price and in ordinate the noisy price

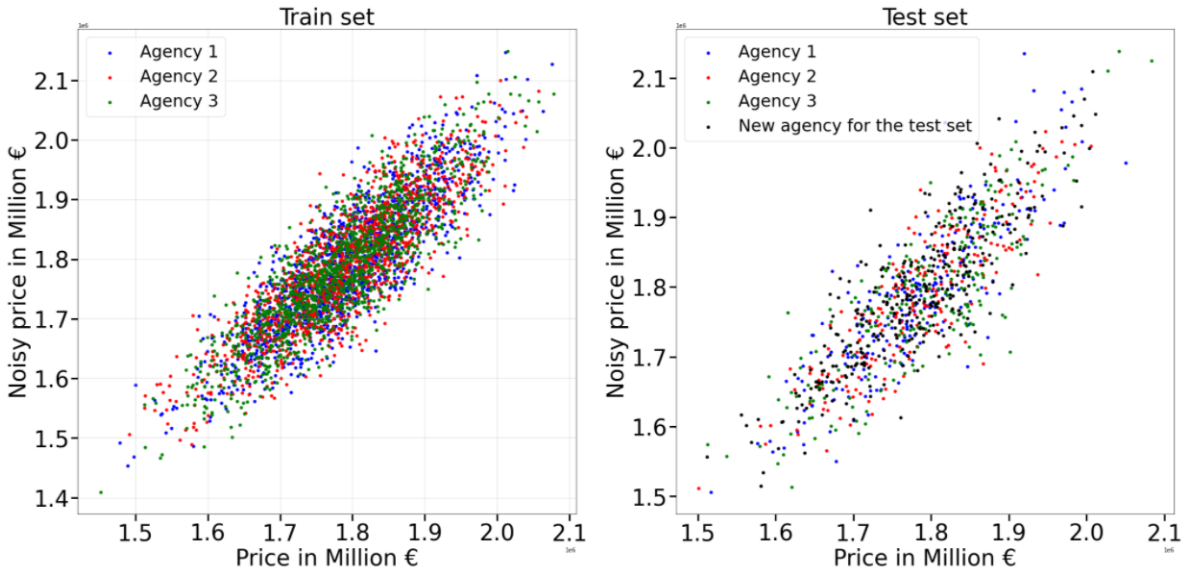


Figure 4: Representation of the different environments of each sample

4.4 First results

4.4.1 Results of the classical model

We train the classic linear model using Scipy, and we obtain the resutls in table 1. At first it seems at first that the model is very far away from the reality. However, if we take a closer look, we can see that the model used all the causal features at roughly 25% of their real impact, and reported the rest left on the spurious correlation.

Features	Value in the model	Classic regression
θ_0	0	-3309
θ_1	35000	7879
θ_2	7000	1551
θ_3	1500	435
θ_4	-100000	-22269
θ_5	0	15

Table 1: IRM Regression results. The numbers are rounded to the nearest integer

Indeed, in average, the commissions represent 5% of the price of the house, thus, a model giving no importance to all features but θ_5, and a value of 20 to θ_5 would do well on the train set, since commissions are also relatively close. We denote such model by the fully-biaised model. Here, the model obtained is slightly different and can be seen as :

$$classic\ model = 0.75 \times fully - biaised\ model + 0.25 \times real\ model$$

The 75% come from the importance granted to θ_5 : $15/20$ and the 25% remaining come from the impact granted to causal features compared to their real value.

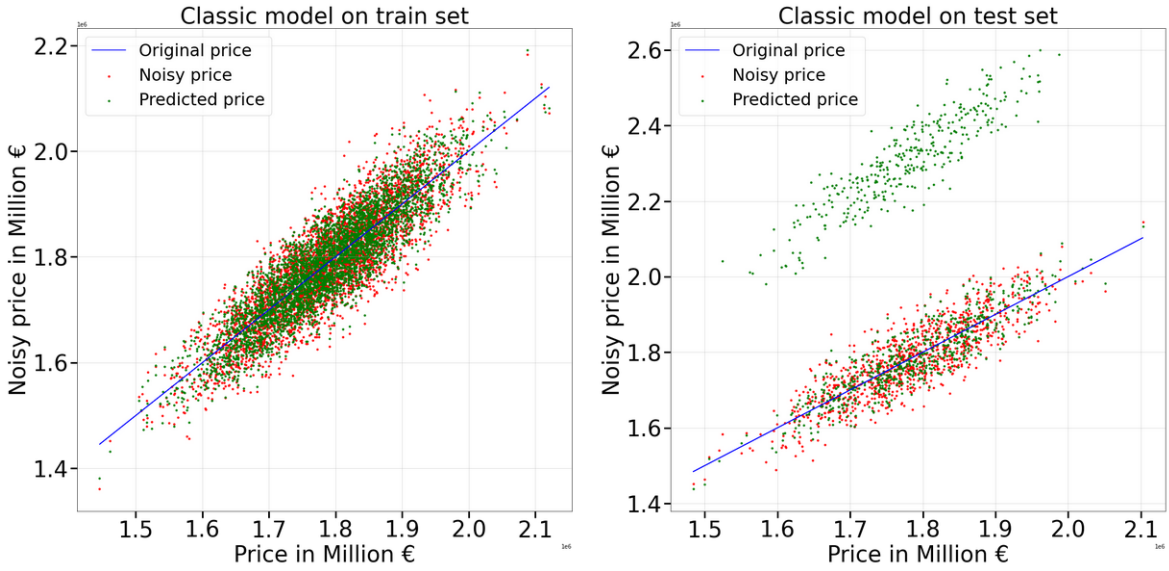


Figure 5: Performances of the classical model

As expected, performances obtained on the training set are excellent. The model manages to predict very accurately the price, and overcome the noise added by using the spurious correlation. However, the drawback of such overfitting pop up when we see the results on the test set. All the houses that were sold by the new agency with a higher commission have huge errors on their price's predictions. As mentionned before, it is totally expected, since we built our toy example precisely to fool the classical model and highligh its limitations on such cases. We are now going to

see how IRM can help us to solve this issue.

4.4.2 Results of the IRMv1 model

With the IRMv1 model, we use a new loss that penalizes the variations of influence of a parameter on the output value.

$$\min_{\Phi:\mathcal{X}\to\mathcal{Y}} \sum_{e\in\mathcal{E}_{\text{tr}}} R^e(\Phi) + \lambda \|\nabla_{w|w=1.0} R^e(w\circ\Phi)\|^2$$

To recall what each variable means, we have :

- $\mathcal{X} : \mathbb{R}^6$, our 5 features and the offset $\theta_0 = 0$
- $\mathcal{Y} : \mathbb{R}$, price of the house
- $\Phi : \mathbb{R}^6 \to \mathbb{R}$, with $\Phi(x) = \theta^T x, x \in \mathbb{R}^6$
- $R^e(\Phi) = \frac{1}{n_e}\sum_{i=1}^{n_e}(\hat{y_i} - y_i)^2$, averaged loss over each environment
- $\lambda : 1$, regularization parameter
- $\|\nabla_{w|w=1.0} R^e(w\circ\Phi)\|^2 = \left(\frac{1}{n_e}\sum_{i=1}^{n_e}\hat{y_i}(\hat{y_i} - y_i)\right)^2$
- e represents our environment, value in $\{1, 2, 3, 4\}$

We use the module scipy to minimize the loss function obtained, and we perform a new linear regression. We obtain the results in table 2

Features	Value in the model	Classic regression	IRM regression
θ_0	0	-3309	1222
θ_1	35000	7879	64179
θ_2	7000	1551	-5553
θ_3	1500	435	3747
θ_4	-100000	-22269	-63961
θ_5	0	15	0

Table 2: Classical regression results. The numbers are rounded

The performances of IRM regression are very poor. The model even struggles with identifying if causal variables increase or decrease the price. However, despite the bad predictions for each causal variable, the model doesn't use the spurious correlation : $\theta_5 = 0$.

The explanation for such poor results is that the IRM loss suffers from a lot of local minima, it is very wavy. Each minimization brings a different minimum, and eventhough almost all the minima are not satisfying, they nevertheless all have $\theta_5 = 0$.

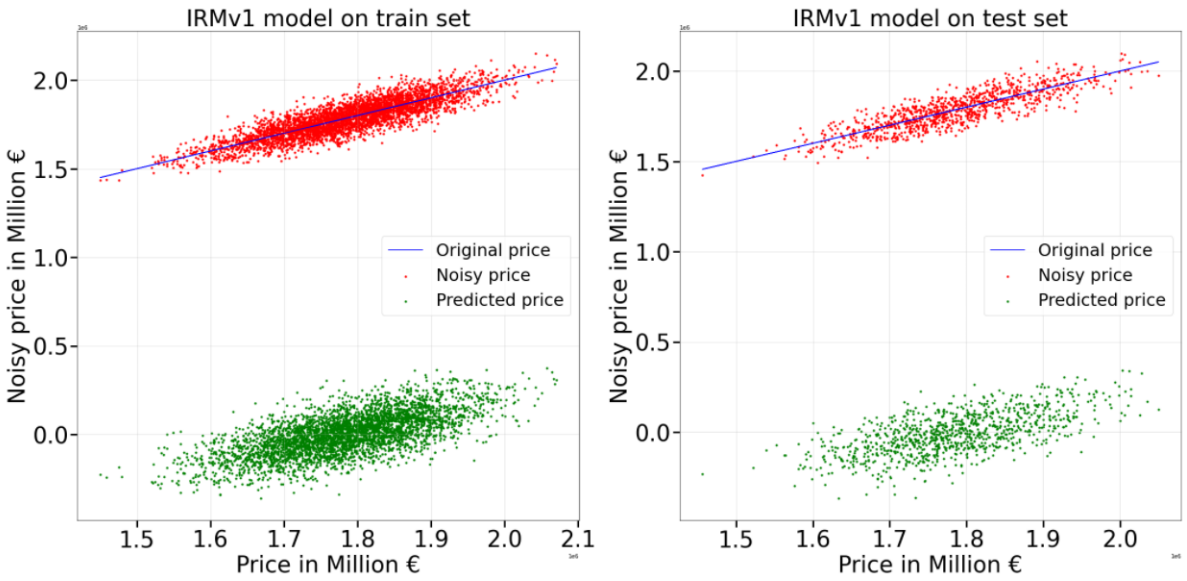


Figure 6: Performances of the IRM model

The first conclusion on IRM regression is that the loss is very wavy with a lot of local minima. This makes the minimization very difficult, and impairs the model to find a satisfying solution. However, it manages very well to identify the spurious features in almost all cases. This is why on the test set predictions, eventhough all points are far away from the true value, we no longer have two clusters of points. The solution is of poor quality, but generalizes well, its behavior remains the same accross the different environments.

4.5 Combining both models

We have two models. The first one is able to get accurate predictions, but uses spurious features. The second one can eliminate the spurious features, but get inaccurate predictions. Thus, it is natural to try to combine the two models to get a final model that eliminates the spurious features and has also accurate predictions.

The idea is to first use IRM regression to identify the spurious features, and keep only the causal ones, and then train a linear model using the classic loss and get good predictions with the features left. One important point is however to have a robust and reliable criterion to decide wheather a feature should be kept or abandoned. Indeed, the coefficients of spurious features found by IRM regression are not exacly equal to 0, thus we can't just remove features with a 0 coefficient.

4.5.1 Contribution of a feature

In order to build such criterion, we first introduce the contribution of a feature. We define :

$$C_j = \frac{\frac{1}{n}\sum i = 1^n \frac{\hat{y_i} - \hat{y_i}^{-\theta_j}}{\hat{y_i}}}{\sum_{j=1}^{p} C_j}, \ \hat{y_i}^{-\theta_j} = \hat{y_i} - \theta_j x_j$$

We compute the contribution of feature j as the relative difference between a prediction with, and without taking feature j into account when computing the output value, averaged over the dataset and normalized, so the sum of the contributions is equal to 1.

The contribution of a variable represents to what extent it contributes to the output value computed. Causal variables should always contribute in the same way (in the sense of positively or negatively) to the output value, whereas non causal variables might contribute postively in some environments, and negatively in others, which will considerably reduce their contribution. Also, this enables us to predict the effect of removing a feature in the model, and see how it might impact the new solution. Eventhough on a scaled dataset, the value of the coefficient of the feature should be very linked to its contribution, we observe that when the values can vary a lot from a sample to another, contributions offer better insight of the influence of a variable on the result.

4.5.2 Criterion for filtering features

Now that we have introduced the contribution of a feature, we can define a criterion to decide wheather or not a feature should be eliminated from the model. We first use IRM regression and obtain some coefficients for each feature.

We can't simply remove the features with small coefficients. Indeed, even if the dataset is scaled, it can simply mean that the impact of the feature is small on the output value, small but relevant. Also, it could lead us to delete a lot of features with small contributions, but that when gathered could represent a much bigger contribution.

The difference between a feature with a small impact on the output value and a spurious feature is not the value of the coefficient. It is how the value changes between training with the classic loss and the IRM loss. Thus, we choose to use the ratio of the contributions between classic model and IRM model.

A spurious feature will have a high contribution in the classic model, and a very small contribution in the IRM model. When computing the ratio, we expect very high values for the spurious features, and smaller values for the causal variables.

Thus, we define the following criterion to decide which features must be kept :

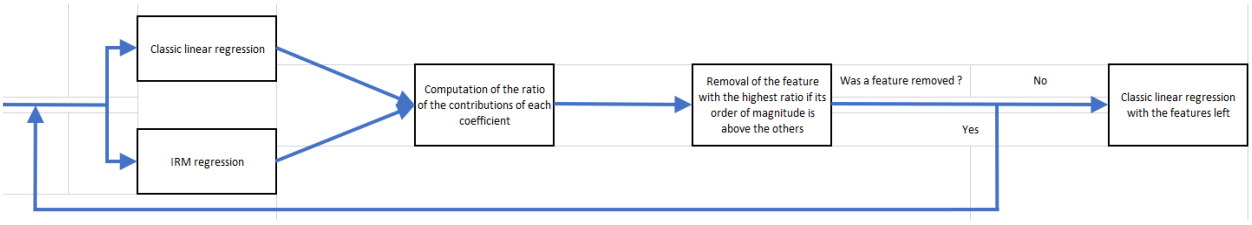


Figure 7: Criterion to eliminate the supposed spurious features

4.6 Final results

4.6.1 Expected contributions

In theory with the true values of the coefficients, the expected contributions on the dataset are shown below. Most of the value of the house comes from the indoor size and the number of rooms. Of course, we have no contribution for θ_5 as the fees dont influence the value of the house.

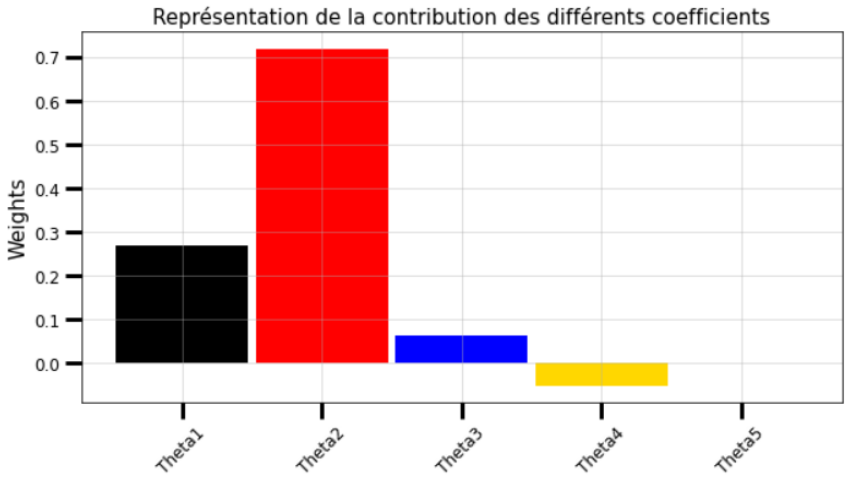


Figure 8: Expected contributions

4.6.2 First training of both models

We start by training two models : one with IRM loss, and another one with the classic loss. We then compute the contributions of each model. We obtain the following results.

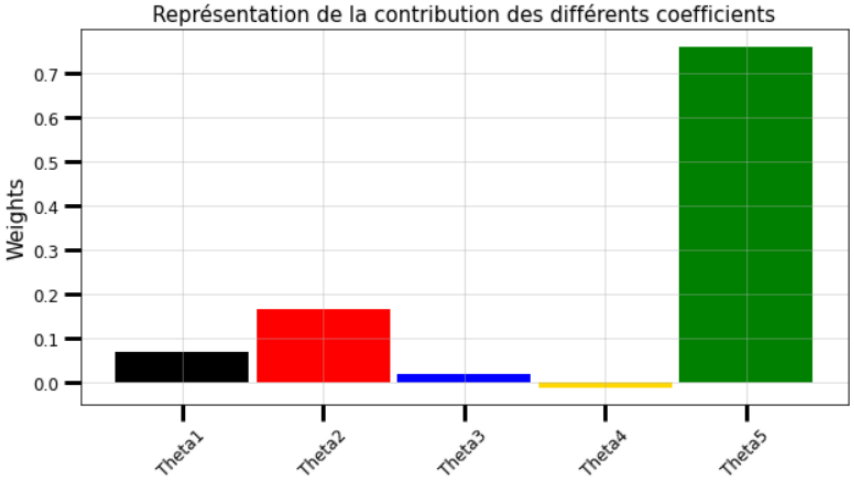


Figure 9: Contributions with the classic model

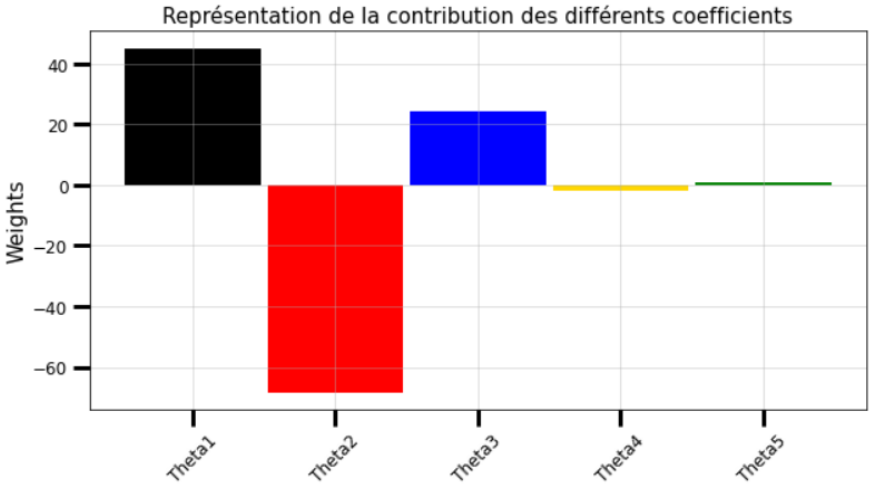


Figure 10: Contributions with the IRM model

Although it seems that the contribution of θ_5 has dropped significantly, we have to be careful to the scale. However, it is clear that its impact is negligible compared to the other features. We can note that accordingly with the negative value found for θ_2, its contribution is very negative.This explains why we had houses with negative prices.

4.6.3 Analysing the ratio of the contributions

In order to remove the spurious features, we now compute the ratios of the contributions.

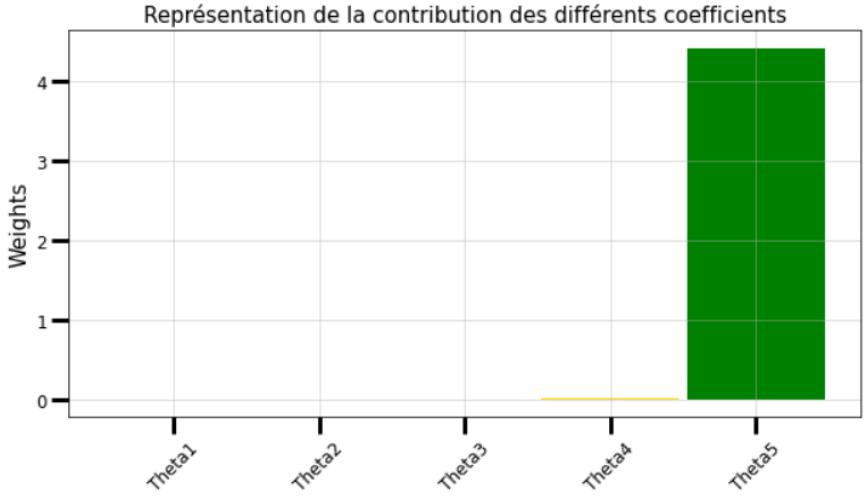


Figure 11: Ratio of contributions : classic / IRM

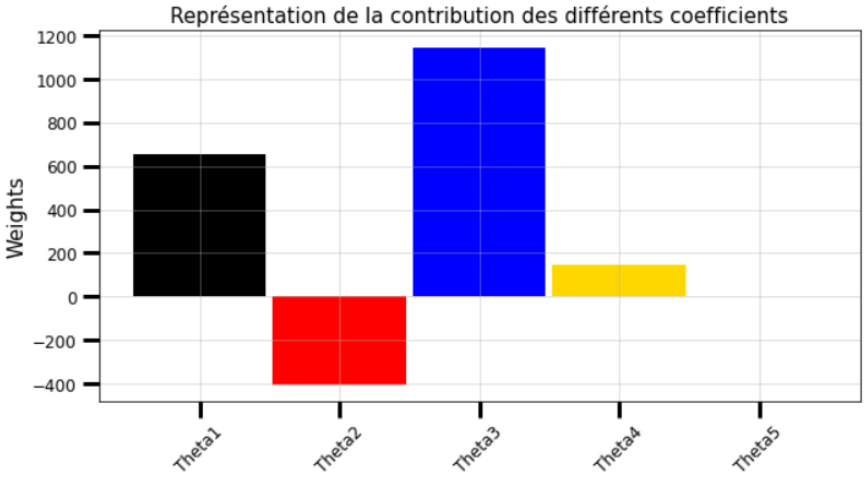


Figure 12: Ratio of contributions : IRM / classic

It is very clear on those two plots that θ_5 is a spurious feature. We can thus remove it.

4.6.4 Results of the final model

Finally, we train a last classic model, without the θ_5 feature in the dataset.

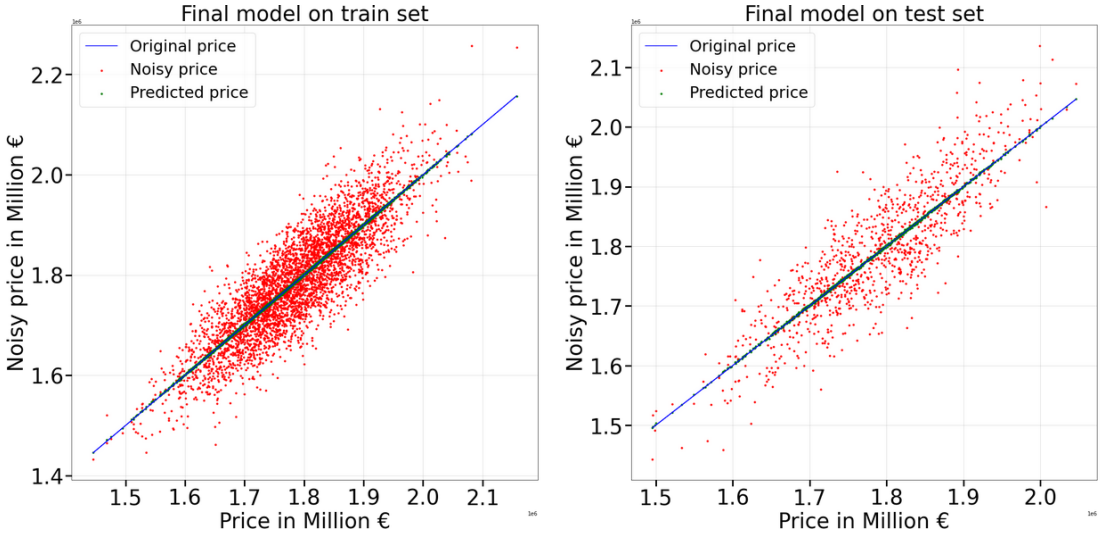


Figure 13: Predictions of the final model

Results obtained are as expected. Since the spurious feature was removed, the model is able to identify the original model used to build the price, and no longer overfits on the training set. This allow us to get very satisfying performances, using only the causal variables.

4.7 Conclusion of the study of the toy example

To conclude the study of this toy example on predicting housing prices, we can keep two elements. First, the loss introduced by IRM paradigm suffers from a lot of local minima, which makes the minimization very hard. A naive minimizer will not manage to find the global minimum and lead to inaccurate results. Second, almost all minima found clearly identify the spurious features. We used this result to build a robust criterion to first eliminate the spurious features, and then retrain a new model with the features left. One possible way to improve the results is thus to improve the minimization process, and directly get an accurate model using only spurious features.

5 Analysis of IRM: shortcomings

5.1 IRM penalty calculation and some bounds

In this section, we detail some calculations that were absent from the original article to get a better idea of the effect of the regularization term. It will also help highlight one of the shortcomings of IRM.

5.1.1 Linear MSE Regression

In the case where the classifier is linear and we're doing Mean Squared Error regression we have the following loss function $\ell(x, y) = \|x - y\|^2$. Notice that Φ could still be non linear. Let's assume we're in a finite dimension case and $\Phi : \mathbb{R}^n \to \mathbb{R}^h$ and $w : \mathbb{R}^h \to \mathbb{R}^d$. Where n is the dimension of the input data, h is the representation's dimension and d is the dimension of the output variable. The classifier w can be represented by a matrix W. Writing explicitly the risk for each environments gives the following :

$$\begin{aligned} R^e(w \circ \Phi) &= E_{X^e,Y^e}(\ell(w \circ \Phi(X^e), Y^e)) \\ &= E_{X^e,Y^e}(\|W\Phi(X^e) - Y^e\|^2) \end{aligned}$$

Let us first compute the gradient with respect to W of the term $\|W\Phi(X^e) - Y^e\|^2$.

$$\begin{aligned} \nabla_W \|W\Phi(X^e) - Y^e\|^2 &= \nabla_W (W\Phi(X^e) - Y^e)^T (W\Phi(X^e) - Y^e) \\ &= 2(W\Phi(X^e) - Y^e)\Phi(X^e)^T \end{aligned}$$

We suppose for the rest of this section that Φ is bounded by K on the support of X^e for all e. If for instance, X^e has a compact support and Φ is continuous this is necessarily the case.

$$\begin{aligned} \|\nabla_W \|W\Phi(X^e) - Y^e\|^2\|_F &= \|2(W\Phi(X^e) - Y^e)\Phi(X^e)^T\|_F \\ &= 2\|(W\Phi(X^e) - Y^e)\|\|\Phi(X^e)\| \quad (*) \\ &\leq 2K \underbrace{\|(W\Phi(X^e) - Y^e)\|}_{\text{integrable}} \end{aligned}$$

Where $\|\cdot\|_F$ is the Frobenius norm. $(*)$ is justified by the fact that for any vectors a and b, $\|ab^T\|_F^2 = \sum_{i,j} a_i^2 b_j^2 = \left(\sum_i a_i^2\right)\left(\sum_j b_j^2\right) = \|a\|^2\|b\|^2$. Therefore, using dominated convergence, we can swap expectation and gradient with

respect to W.

$$\begin{aligned}\nabla_W R^e(w \circ \Phi) &= E_{X^e,Y^e}(\nabla_W \|W\Phi(X^e) - Y^e\|^2) \\ &= E_{X^e,Y^e}(2(W\Phi(X^e) - Y^e)\Phi(X^e)^T)\end{aligned}$$

Now let us find an upper bound for the IRM penalty in this case. For all e we have:

$$\begin{aligned}\|\nabla_W R^e(w \circ \Phi)\|_F^2 &= \|E_{X^e,Y^e}(2(W\Phi(X^e) - Y^e)\Phi(X^e)^T)\|_F^2 \\ &\leq E_{X^e,Y^e}(\|2(W\Phi(X^e) - Y^e)\Phi(X^e)^T\|_F^2) \quad (*) \\ &= E_{X^e,Y^e}(\|2(W\Phi(X^e) - Y^e)\|^2 \|\Phi(X^e)^T\|^2) \\ &\leq E_{X^e,Y^e}(\|2(W\Phi(X^e) - Y^e)\|^2 K^2) \\ \|\nabla_W R^e(w \circ \Phi)\|_F^2 &\leq 4K^2 R^e(w \circ \Phi) \\ \sum_{e \in \mathcal{E}_{tr}} \|\nabla_W R^e(w \circ \Phi)\|_F^2 &\leq 4K^2 \sum_{e \in \mathcal{E}_{tr}} R^e(w \circ \Phi) \qquad (6)\end{aligned}$$

(*) jensen's inequality since $x \to \|x\|_F^2$ is convex. Therefore, the Invariant Risk penalty term is bounded by a constant times the average Expected Risk on all environments. So if we have Φ that has 0 average training error, its IRM penalty is also null. This is bad news since when Φ is a neural network, we can nearly always reach 0 training error by over fitting and it would also reduce the invariance penalty to 0.

5.2 Logistic Regression

Let us do the same calculations as before but in the logistic regression case. In the case where the target Y is a binary variable that takes 0 or 1 as values. In this case the linear classifier w is a function $\mathbb{R}^h \to \mathbb{R}$ and can therefore be represented as a vector w and $w(x) = w^T x$. With logistic regression we have $\ell(x, y) = -[y \log(\sigma(x)) + (1 - y) \log(1 - \sigma(x))]$. With $\sigma : x \to \frac{1}{1+\exp(-x)}$ being the sigmoid function. In this case the risk is written as:

$$R^e(w \circ \Phi) = E_{X^e,Y^e}(\ell(w^T\Phi(X^e), Y^e))$$

Let's first compute the derivative of this loss function with respect to x. We know that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Therefore

$$\begin{aligned}\frac{\partial \ell(x, y)}{\partial x} &= -[y\frac{1}{\sigma(x)}\sigma(x)(1 - \sigma(x)) + (1 - y)\frac{1}{1 - \sigma(x)}(-\sigma(x)(1 - \sigma(x)))] \\ &= -[y - y\sigma(x) + y\sigma(x) - \sigma(x)] \\ &= -[y - \sigma(x)]\end{aligned}$$

We know that for $f : \mathbb{R} \to \mathbb{R}$

$$\nabla_w f(w^T a) = a f'(w^T a)$$

By using the chain rule. Therefore

$$\begin{aligned}\nabla_w \ell(w^T\Phi(X^e), Y^e) &= \Phi(X^e)\frac{\partial \ell(w^\Phi(X^e), Y^e)}{\partial x} \\ &= -\Phi(X^e)[Y^e - \sigma(w^T\Phi(X^e))]\end{aligned}$$

Now by supposing Φ bounded by K as in the previous section we get

$$\begin{aligned}\|\nabla_w \ell(w^T\Phi(X^e), Y^e)\| &\leq K|Y^e - \sigma(w^T\Phi(X^e))| \\ &\leq K(|Y^e| + |\sigma(w^T\Phi(X^e)|)) \\ &\leq 2K\end{aligned}$$

Therefore using dominated convergence, we can swap expectation and gradient.

$$\begin{aligned}\|\nabla_w R^e(w \circ \Phi)\|^2 &= \|E_{X^e,Y^e}(\nabla_w \ell(w^T\Phi(X^e), Y^e))\|^2 \\ &= \|E_{X^e,Y^e}\Phi(X^e)[Y^e - \sigma(w^T\Phi(X^e))]\|^2 \\ &\leq E_{X^e,Y^e}\|\Phi(X^e)[Y^e - \sigma(w^T\Phi(X^e))]\|^2 \quad (*) \\ &\leq E_{X^e,Y^e}\|\Phi(X^e)\|^2[Y^e - \sigma(w^T\Phi(X^e))]^2 \\ &\leq K^2 E_{X^e,Y^e}[Y^e - \sigma(w^T\Phi(X^e))]^2 \\ \sum_{e\in\mathcal{E}_{tr}} \|\nabla_w R^e(w \circ \Phi)\|^2 &\leq \sum_{e\in\mathcal{E}_{tr}} K^2 E_{X^e,Y^e}[Y^e - \sigma(w^T\Phi(X^e))]^2 \end{aligned} \tag{7}$$

(*) using the convexity of $x \to \|x\|_F^2$. Similarily to the case of linear regression we also upper bound the penalty with the average quadratic error of our network $w \circ \Phi$. Again, Φ could overfit the training data and reach 0 average quadratic training error and also drive the invariance penalty to 0.

5.3 Limitations of IRM and Related Work

The IRM framework has generated a lot of interest in developing algorithms for finding invariant representations. Many papers point out some of the flaws of IRM and try to build on them [14, 15, 16, 17]. Although we have looked through some of these methods, a proper review of all of the papers based on IRM would need a standalone article. We leave this to future work. We will limit ourselves to the following article that is of particular interest to this report.

[17] studies IRM theoretically in a case where the latent variables are gaussian. They show that in the linear case, one necessarily needs strictly more environments than there are dimensions of the spurious latent features. In that case, IRM recovers a truly invariant representation across all possible interventions. This guarantee is quite powerful as it ensures generalization and extrapolation well beyond just the scope of the training distribution. This paper criticizes IRM by pointing out that the number of environments needs to scale linearly with the dimension of spurious features and is thus quite limiting. They also study IRM in the nonlinear case but this time using the IRMv1 loss. They show the existence of a representation Φ_ϵ that is invariant on an arbitrarily large proportion of the training data. However, this representation is far from being invariant everywhere. When one changes the distribution far enough from the training distribution, the predictions of Φ_ϵ are identical to those of a representation trained only on the empirical risk. This representation also has a lower loss value than a real invariant solution, which makes it a more attractive one for the optimization process. It is a major limitation since it means that what may seem like an invariant representation is actually far from being one.

This last heavy limitation has been exposed in the case where the IRMv1 loss is used. They use the fact that one can make the IRM penalty arbitrarily small while also making the empirical risk smaller by using spurious features. Since the regularization constant λ is fixed, there is a point at which the added penalty is smaller than the decrease in the average empirical risk. We've read this paper thoroughly and redid most of the proofs in it.

This limitation in the non linear case has motivated us to try an exterior penalty method to tightly control the IRM penalty term. My first hypothesis was that the solutions were not exactly invariant because the emperical risk took over during the optimization process. So if we could force the optimization to converge to a solution with a very small penalty value then it would be better. In the following section, I explore this idea and present some results related to it.

Another problem we've witnessed with IRM is its sensitivity to initialization. The representation $\Phi = 0$ always has an IRM penalty of 0. The IRM penalty thus has a minimum at 0 that attracts gradient descent. The original paper claims that the mean expected risk term would drive the representation away from this point. However, we found in our experiments that when initialized too close to 0, the representation Φ tends to gravitate to it.

6 Ideas for improvement of IRM and Experiments

6.1 Adaptive Regularization

6.1.1 Exterior Penalty Method

We propose to use an exterior penalty method. With this type of method the constrained problem of the form

$$\mathcal{P} : \begin{cases} \min_x & f(x) \\ \text{s.t.} & g(x) = 0 \end{cases}$$

is approximated with the following (unconstrained) penalized problem

$$\mathcal{P}_\lambda : \min_x f(x) + \lambda \|g(x)\|^2$$

The solutions of $\mathcal{P}_\lambda$ converge to the solution of $\mathcal{P}$ as $\lambda \to +\infty$. (Section 6.2.1.1 in [18]). In practice, a series of problems $\mathcal{P}_\lambda$ are solved by increasing λ by an order of magnitude at each step. To aid the optimization process, the solution to the problem at step i is used to initialize the optimization problem at step $i + 1$. The following ideas are inspired from this method.

6.1.2 Simple linear experimental setup

The experiments in this section are based on the simple scenario described in section 2. We consider a training dataset with two environments with standard deviations $\sigma_1 = 0.1$ and $\sigma_2 = 1.5$. Each environment has 5000 samples. There are no mini-batches during optimization. We train with the full 10000 samples at each epoch. The Adam optimizer is used with a learning rate of 10^{-3}. The cost function used is the squared error : $\ell : x, y \mapsto (x - y)^2$ We use a linear representation $\Phi(x, z) = \Phi_0 x + \Phi_1 z$. We will denote the application and the vector $(\Phi_0, \Phi_1)^T$ as the same quantity Φ. The linear classifier w is the fixed constant 1. Thus $w \circ \Phi(x, z) = w\Phi(x, z) = \Phi(x, z)$ Although in the forward pass, the classifier doesn't change anything, it is used later to compute the IRM penalty term $\|\nabla_w R^e(w \circ \Phi)\|$. We run all the experiments below 32 times with different seeds. However, these 32 seeds stay the same between experiments so that the initializations of Φ are the same for all experiments. Note that since the spurious variable is of dimension 1, IRM can theoretically recover the invariant representation with just 2 different environments. [17] To test the attained solutions, similar to [14], we create a test set from the same environments as training but we shuffle the spurious variable across the samples. This gets rid of the correlation between this variable and the target variable. If

a representation uses this spurious variable, its mean squared error will be very high on the test set. As we said in section 5.3, the IRMv1 loss tends to gravitate towards the null representation $\Phi = 0$. This is best seen with figure 14.

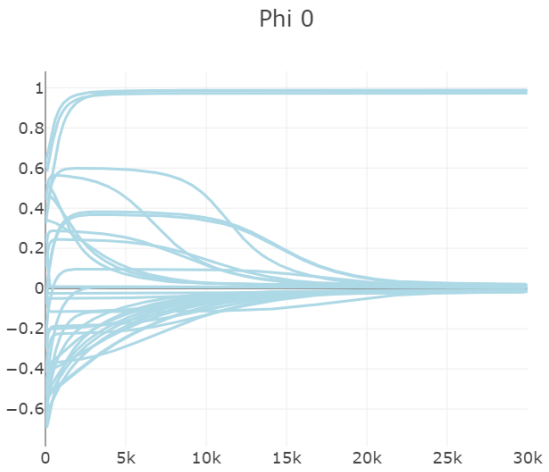


Figure 14: Evolution of Φ_0 Vanilla IRMv1
Launched 32 runs with IRMv1 loss with different seeds. This plot shows the evolution of the first component of Φ that corresponds to the invariant feature. 29 runs converge to the 0 representation. Only 3 found the invariant representation.

6.1.3 Proposal

Instead of tuning the λ parameter and fixing it throughout the whole training, we begin by fixing λ to a small value (for example 1). Once the parameters converge (we'll see later how we measure this), we update the regularization constant by multiplying it by M (in our experiments $M = 10$). This lets gradient descent take the solution away from the null representation at first. Then once the parameters have converged, it changes the optimization landscape by making the IRM penalty more prominent. Figure 15 shows the evolution of solutions through the loss landscape using this adaptive regularization compared to the vanilla IRMv1 loss.

6.1.4 Simple adaptive regularization

At first we measured the convergence of the parameters with the norm of the gradient. Once it attained a certain threshold T we updated the regularization constant. However the threshold value was a little hard to choose. So instead we went for measuring convergence with a relative change, so for the parameter vector θ we measure convergence as

$$\Delta\theta = \frac{\|\nabla_\theta L\|}{\|\theta\|} \tag{8}$$

Where L is the total loss being minimized. See algorithm 1 for the structure of the simple adaptive regularization.

This simple modification results in much better convergence to the invariant solution, although it's still far from perfect. See figure 16.

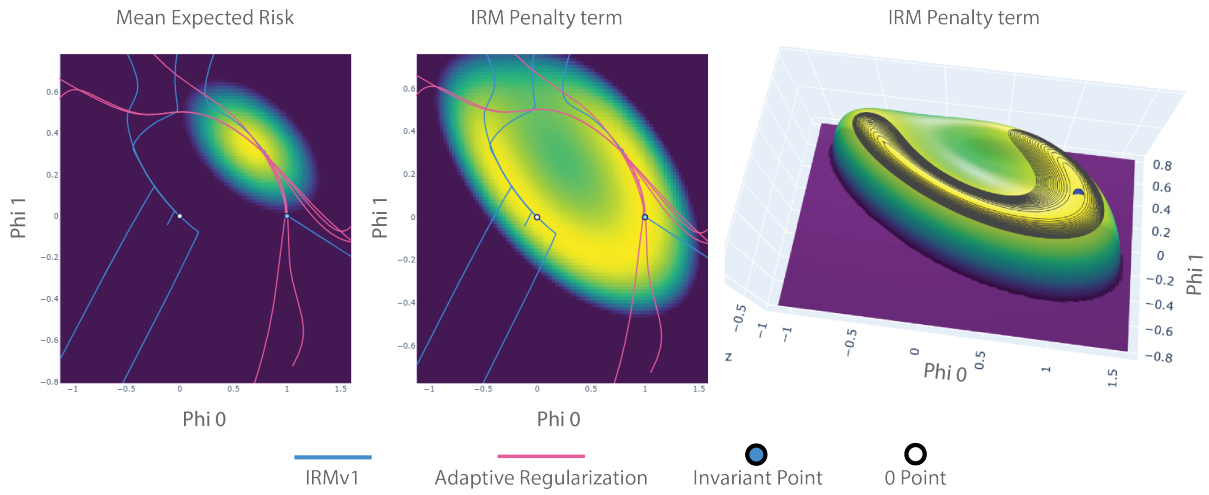


Figure 15: Visualization of the loss landscape

Visualization of the loss landscape in the simple linear example presented above. On the left is the mean empirical risk. It is a convex function with a minimum around (0.6,0.3). On the middle and right is the IRM penalty term visualized in two different ways. The right plot is to emphasize the non-convexity of the penalty term. What is shown is actually clipped$(-\text{penalty})$ for visualization purposes. This penalty term is non-convex even for the simple linear example presented here. It has multiple minima, the notable ones being (0,0) and (1,0). The latter corresponds to the invariant solution we wish to find. The lines in the left and middle plots present the evolution of multiple runs. The blue lines represent the evolution of Φ with the $IRMv1$ loss. When Φ is initialized on the left of (0,0), the optimization converges to (0,0). The pink lines represent optimization using the adaptive regularization scheme. Notice that they seek out the minimum mean empirical risk at first then continue on to the invariant solution. They thus avoid the null representation.

However, this optimization procedure is limited for the following reasons. First, the threshold heavily depends on the data and the architecture of the neural network. In these simple linear toy examples, it is easy to check for the convergence and tune the threshold. However, in a more complex setting, it is unrealistic to put a threshold on the gradient of all the parameters to measure convergence. Second, this imposes an exponentially growing penalty term uniformly on the whole network. This thus assumes that different parts of the network converge at the same speed.

We improve on this simple adaptive regularization scheme in two ways. First, we keep an exponential average of the gradient squares, as is done in Adam [19]. We then apply the thresholding to the relative change on this quantity. The exponential average of the squares approximates the second-order moment of the parameter gradients. [20] It is thus much more reliable to put a threshold on this.

Second, we apply a regularization constant for each parameter separately. This is done by modifying the gradients in the backward pass of each parameter instead of minimizing the combination of the two losses. See algorithm 2

This decouples the network parameters and allows for more fine-grained thresholding and penalization. First of all, it makes the algorithm much less sensitive to the choice of the threshold. As long as it is small enough, the training is relatively stable. Also, if parts of the network start converging to a spurious sub representation, the penalty is exponentially increased to prevent this. The other parts of the network that are still moving towards a empirical risk

Algorithm 1: Simple Adaptive Regularization

Input: λ_{init}, T, M

Output: Φ_θ

Data: Training data

$\lambda := \lambda_{init}$

while *Current Iteration* $<$ *Max Iterations* **do**

  Perform gradient descent step on $\sum_{e \in \mathcal{E}_{tr}} R^e(w \circ \Phi_\theta) + \lambda \|\nabla_w R^e(w \circ \Phi_\theta)\|$

  if $\Delta\theta \leq T$ **then**

    $\lambda := M\lambda$

Algorithm 2: Adaptive Regularization

Input: λ_{init}, T, M, β

Output: Φ_θ

Data: Training data

$\lambda := \lambda_{init} *$ ones_like(θ) $e :=$ zeros_like(θ)

while *Current Iteration* $<$ *Max Iterations* **do**

  $g_{erm} = \nabla_\theta R^e(w \circ \Phi_\theta)$

  $g_{irm} = \nabla_{\Phi_\theta} \|\nabla_w R^e(w \circ \Phi_\theta)\|$

  $g = g_e + \lambda \odot g_p$

  $e = \beta e + (1 - \beta) g^2$

  Perform gradient step with g

  for i *such that* $\frac{e[i]}{|\theta[i]|} < T$ **do**

    $\lambda[i] := M\lambda[i]$ `/* the bracket [i] is for indexing the vector */`

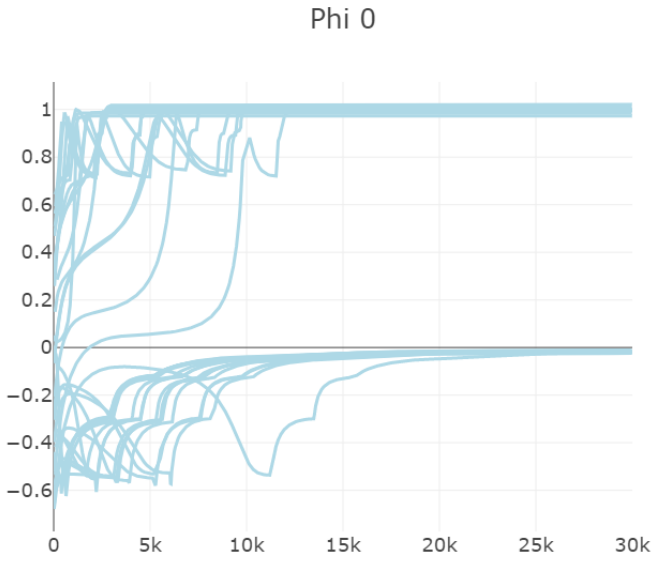


Figure 16: Evolution of Φ_0 with simple adaptive regularization
Launched 32 runs with IRMv1 loss with a simple adaptive regularization scheme with the same seeds as figure 14. This plot shows the evolution of the first component of Φ that corresponds to the invariant feature. Half of the runs converge to the invariant representation

critical point but have not converged, continue to do so. Figure 17 shows the drastic improvement in convergence to the invariant solution. Figure 18 shows the evolution of these decoupled regularization constants throughout training for the 32 different runs.

This adaptive regularization scheme also addresses one of the limitations of IRM mentioned in [17]. Indeed, by letting the regularization parameter grow arbitrarily, the IRM penalty will never become negligible in comparison with the empirical error.

To quantify how these improvements perform in a test scenario, we added two experiments. The first one is just training a linear regression using gradient descent. We refer to this experiment as ERM for Expected Risk Minimization. The second one is also a linear regression but this time, the spurious variables are shuffled as is done in the test set. This operation is not possible in a real scenario where we don't know which variables are spurious. But it serves as a kind of upper bound of the performance. We refer to this experiment as "oracle". This is similar to what is done in [14], where they expand more on these simple linear unit tests and benchmark IRM against other algorithms. Figure 19 is a box plot summarizing the test results on the 32 runs for each experiment.

6.1.5 Issues with this approach

Although this approach works quite well with this toy example, there are multiple concerns that need to be addressed. From a theoretical perspective, [18] shows that using a differentiable penalty function can lead to ill conditioned problems (Section 6.2.2 of [18]). For differentiable penalty functions, the regularization constant λ might need to be

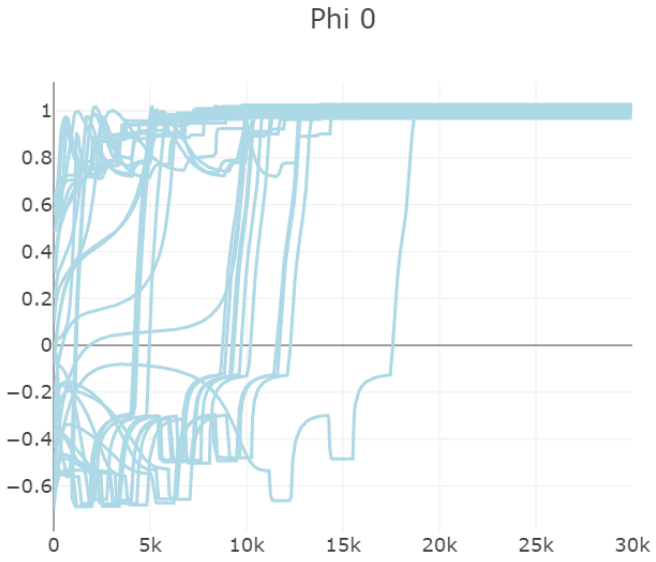


Figure 17: Evolution of Φ_0 with adaptive regularization

Launched 32 runs with IRMv1 loss with adaptive regularization with the same seeds as figure 14. This plot shows the evolution of the first component of Φ that corresponds to the invariant feature. All runs converge to the invariant representation

arbitrarily large to reach the real minimizer in the constrained set. This translates to poor performance in practice since overflow and numerical stability is bound to happen if we don't limit the regularization constant. We have witnessed this in the non linear experiments, where many times, the gradients would become NaN. Choosing a non differentiable penalty (like the absolute value) could help solve this problem, but we would then need tools more involved than gradient descent (e.g. proximal algorithms). We leave this matter to future work.

6.2 Non linear experiments: Colored MNIST

6.2.1 Colored MNIST

The IRM paper introduces a synthetic dataset to test the performance of IRM in the non linear case. This dataset is called Colored MNIST. In this dataset we take the original MNIST dataset and assign the label 0 to numbers smaller than 5 and label 1 to numbers bigger than 5. Then 25% of these labels are swapped to introduce label noise. Then, a spurious correlation is introduced by coloring the numbers. During training, a majority p_e of the class 0 is colored green and a majority p_e of the class 1 is colored red. However, the proportion p_e changes from one environment to another. Therefore the dependence between color and label is not invariant. During test phase however, p_e is made very small so that the correlation between color and labels is reversed. Therefore a classifier that relies on color to classify will get a worse than chance classification.

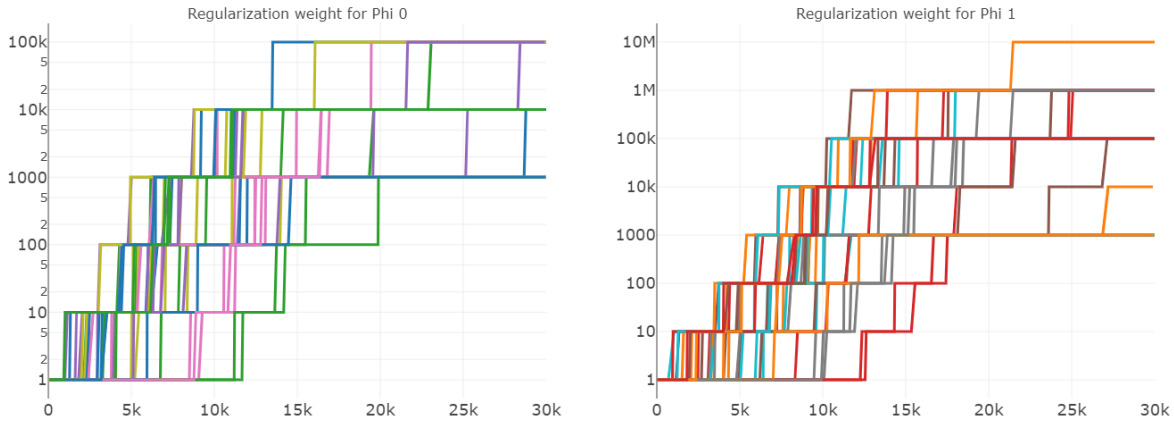


Figure 18: Evolution of regularization constants

Same experiment as figure 17. This plot shows the evolution of the regularization constants for Φ_0 and Φ_1. We can see that the regularization constants increase in steady intervals. This optimization can be thus seen as a series of penalized problems that converge to the original constrained problem in IRM. The parameter Φ_1 that corresponds to the spurious feature in this problem is much more heavily penalized than Φ_0. This parameter is thus forced to go to 0. Decoupling the regularization of the two parameters made for a more flexible optimizaiton process.

6.2.2 Reproducing IRMv1

Algorithm	Acc. train envs.	Acc. test env.
ERM	87.4 ± 0.2	17.1 ± 0.6
IRM (paper)	70.8 ± 0.9	$\mathbf{66.9 \pm 2.5}$
IRM (reproduction)	70 ± 1.56	$\mathbf{68.19 \pm 1.61}$
Random guessing (hypothetical)	50	50
Optimal invariant model (hypothetical)	75	75
ERM, grayscale model (oracle)	73.5 ± 0.2	73.0 ± 0.4

Table 3: Reproduction of IRMv1 on ColroedMNIST

Table 1: Accuracy (%) of different algorithms on the Colored MNIST synthetic task. ERM fails in the test environment because it relies on spurious color correlations to classify digits. IRM detects that the color has a spurious correlation with the label and thus uses only the digit to predict, obtaining better generalization to the new unseen test environment.

In Table 3 are the results of the original IRM paper along with our reproduction. It was quite challenging to get IRM working on the Colored MNIST dataset. Since it was a reimplementation from scratch, I noticed some details that were crucial to the functioning of IRM that weren't mentioned in the original article. First and foremost, in the IRMv1 colorMNIST code, the authors use an annealing period of 190 epochs where the regularization constant

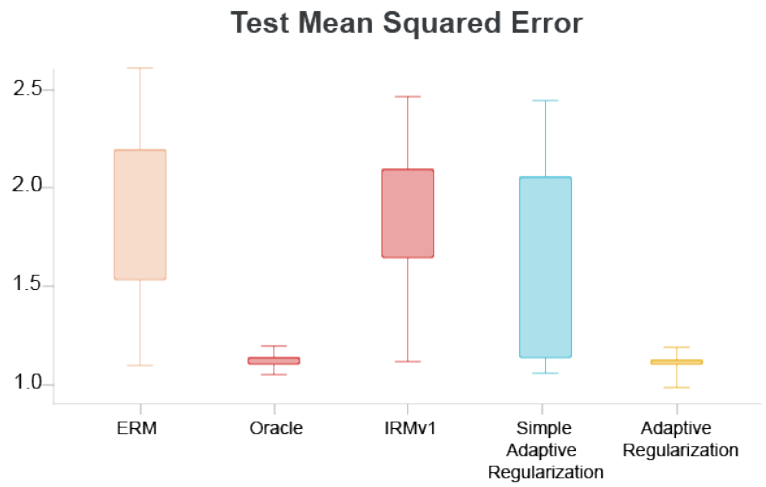


Figure 19: Linear test results

Test results. We see that IRMv1 here performs rather poorly in average, barely better than ERM. It should be noted that the hyperparameters were not extensively tuned in order to simulate a real-world case where a full hyperparameter search is not always possible. This result is thus more about the sensitivity of IRMv1 to initialization and the regularization hyperparameters rather than its true potential performance. The simple adaptive regularization scheme improves on IRM by making more of the solutions converge to the invariant one. Adaptive Regularization as defined before performs the best, with a test score nearly identical to the oracle.

is set to 1. Once that period is over, it is set to a much higher constant (90000). Without this annealing period, the training converges directly to a non informative solution (50% training and validation error) that is equivalent to random guessing. This made me realize that the neural network needs to first converge to a solution that has learned useful features even if they're spurious and then regularize it. When regularizing too soon, the neural network just learns to output random or constant features that don't depend on the input. Figure 20 shows the evolution of a successfully trained invariant network. This crucial annealing period can be seen as a manual version of the adaptive regularization method presented in section 6.1.4. Indeed, the annealing period was manually tuned so that after the network converges, we apply heavy regularization on it. To showcase this, we used the simple adaptive regularization method with a threshold on the relative change of parameters at 0.01, an initial regularization weight of 1 and a multiplicative constant of 90000. This gives the evolution of accuracies in figure 21. In contrast to the annealing period, the simple adaptive regularization method gives us a meaningful and normalized threshold hyperparameter to tune. Indeed, the threshold parameter can be understood as a percentage of change allowed to consider that the model converged. In comparison to the number of epochs before applying heavy regularization in annealing, this threshold can be more easily transposed to other problems.

6.3 Main IRM Limitation and take away message

After experimenting with the original IRMv1 loss and discussing with Dr. David Lopez-Paz and Dr. Martin Arjovsky, we have come to realize that a good summarization of the current problems with the IRM was that we can only

Figure 20: Evolution of training accuracy averaged over 16 runs. At first, the network converges to a high accuracy and as soon as the penalty is increased, the training accuracy decreases and the validation accuracy increases. This means that the network has discarded the color feature in favor of the digit shapes.

Figure 21: Evolution of training accuracy with simple adaptive regularization method averaged over 16 runs. We notice that the evolutions are quite similar to optimizing with an annealing period.

guarrantee the following :

$$\boxed{\Phi \text{ is Invariant} \implies \text{IRMv1 penalty is } 0}$$

Therefore a small penalty value is only a **necessary** condition and not a sufficient one, especially in the non linear case. The penalty still has a lot of issues distinguishing between useful and useless 0 training error solutions. We have witnessed this extensively with the adaptive regularization method that could very easily tightly control the penalty and make it very small. However, that would often coincide with non informative solutions. This is one of the reasons why we used the simple adaptive regularization method instead of the more advanced one in the non linear case. The more complex adaptive regularization made the penalty very small too fast. So to overcome this issue of being only

a necessary condition, one has to navigate the loss landscape "manually" to find the truly invariant solution and avoid non informative ones. This is of course far from ideal since this needs a lot of effort even in a simple scenario such as this one. Multiple ways to overcome this are currently being considered and experimented with :

- Using a non linear classifier : up until now, we have assumed w to be linear and fixed. This creates an imbalance between Φ and w. This in turn leads to Φ having too much freedom and destroying any gradient signal with respect to w.
- Using higher order optimality measures : measuring the optimality of w with the norm of the gradient is only valid if w is linear and the loss is convex. Therefore, if we want to use non linear w, then we would probably need to have an inner training loop for w to approximate the distance of the current w to the optimal one.
- Rethinking the split into representation/classifier : Splitting the model into Φ and w could lead to some problems such as Φ not depending on the input, which would make any w optimal. Even if w is arbitrarily complex, if Φ destroys too much information, then w can't distinguish between environments anymore and the invariance penalty reduces to 0. We could for example have some parameters picked randomly throughout the network to measure invariance.

All of these ideas have been implemented and are currently being experimented with. However, we don't report results of these experiments here since they are not ready yet.

6.4 Experiment Code

One of the major contributions of this project was the design of very modular experimental code. Since this is work that will continue beyond the scope of this project, the code was made to be fully customizable with modules that can be very easily swapped out. The code was built from the ground up, using PyTorch Lightning to automate the training loop, logging, and checkpointing. We also used hydra which is a configuration manager that enables us to compose different parts together. Each experiment is associated with a big configuration file where all the hyperparameters are ordered hierarchically. This allows for easier reproduction and verification. We also tried to abide by the principle of separation of concerns by limiting what each part of the code could do to a bare minimum. We encourage the reader to refer to the code (link). This is a public version of the code where I omitted parts that were contributed by David Lopez-Paz and Martin Arjovsky as their work is still private. All the code in here is done from scratch, IRM has been reimplemented and tested by reproducing the original paper's experiments (both linear and non linear).

7 Domain Adaptation with Adversarial Networks

7.1 Model Presentation and Description

One of the most popular approaches in domain adaptation is the use of adversarial networks. In our project, we are considering the DANN architecture (Domain-Adversarial Neural Networks) introduced by Ganin et al. in [21]. The network architecture is given by the figure 22 below.

The network is composed of 3 main blocks:

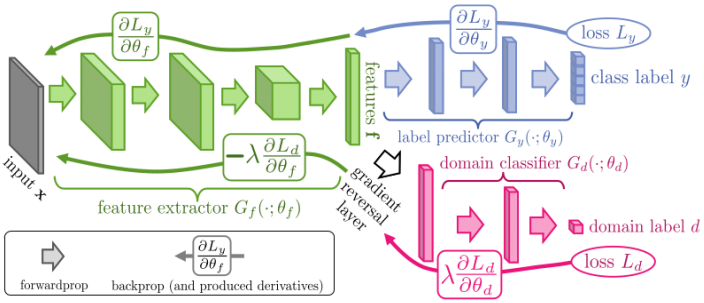


Figure 22: DANN architecture

- The feature extractor G_f which aims to extract features that are domain invariant.
- The label predictor G_y which is a fully connected network that is used to predict the label of the observations.
- The domain classifier G_d that is used to predict the domain from which comes each observation.

The tricky part in DANN lays in the gradient reversal layers: in fact instead of minimizing the domain classification loss, we will rather maximize it by introducing a minus sign to reverse the gradients when performing backpropagation. During training the network has access to N training observations and their labels, and N' test observations but without their labels. Therefore the global network loss can be written as:

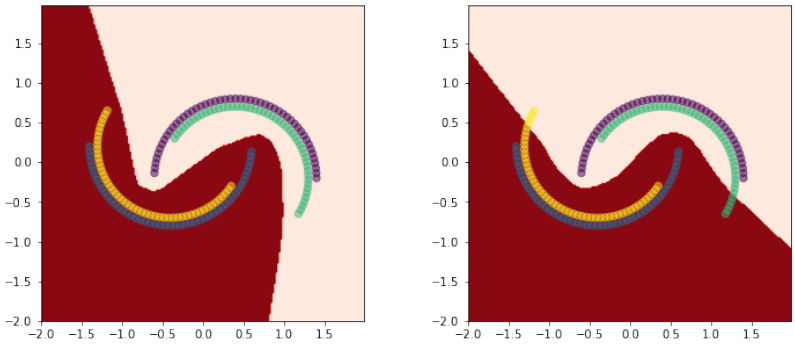
$$L(X, y, y_d, \theta_f, \theta_y, \theta_d) = \frac{1}{N}\sum_{i=1}^{N} L_y(X^i, y^i, \theta_y, \theta_f) - \lambda \frac{1}{N+N'}\sum_{i=1}^{N+N'} L_d(X^i, y_d^i, \theta_d, \theta_f)$$

where:

- θ_f, θ_d and θ_y are the parameters of the 3 network blocks.
- X^i are the observations.
- y^i are the target labels.
- y_d^i are the domain labels (0 for source and 1 for target if we have just two domains).
- L_y and L_d are the label prediction loss and the domain prediction loss respectively.
- λ is a hyperparameter of the model. For our experiments, we used an adaptive scheduling for this parameter : we start with $\lambda = 0$ to let the network focus on predicting target labels for the training set correctly and then we increase λ progressively (up until $\lambda = 2$) to penalize feature representations that are domain specific by maximizing the domain prediction loss.

Note that in the second sum we have a total of $N + N'$ terms.

We did a series of experiments to assess the advantages and also the limits of DANN for both domain adaptation and eliminating spurious correlations.



(a) DANN architecture: the network classifies correctly all observations in the Source and Target environments

(b) ANN: The network fails to generalize for the Target environment

Figure 23: Decision boundaries for the twinning moons experiment

7.2 Experiment 1: Twinning Moons

In this experiment, we consider a classification task where we have two environments:

- The training environment (or the Source environment) where we have two semi-circles labelled as 0 and 1 respectively.
- The test environment (or the Target environment) where we also have two semi-circles that correspond to the rotation of the semi-circles of the Source environment by an angle of 30 degrees.

For this task we compared the performance of DANN to a standard shallow fully-connected neural network. We drew the decision boundaries of both networks and we can see them in Figure 23

As we can see from the drawn decision boundaries, DANN generalizes better to the target environment.

7.3 Experiment 2: Modified MNIST

In this experiment we also consider two domains:

- The source (training) domain consists of the standard MNIST dataset.
- The target (test) domain consists of the MNIST images where the digits are coloured and we add a random background to each image. We can see some examples in Figure 24

We compare the DANN performance with a CNN as a feature extractor to a standard CNN network. After tuning the hyperparameters for both methods, we give the top results for each method in the table 4:

Both methods perform very well on the training data but the CNN performance deteriorates on the target environment as it is incapable to generalize: the features that are extracted are domain specific. As for DANN, we can see that, using the convenient value for the regularization parameter λ we achieve satisfactory results.

We noticed during this experiment that if we keep λ constant during training then :

Figure 24: Modified MNIST Dataset examples

Network Architecture	Source Accuracy	Target Accuracy
Convolutional DANN	99 %	84 %
CNN	99 %	16 %

Table 4: DANN Modified MNIST resutls

- If λ is too small then the network behaves like a CNN and fails to generalize for the Target set.
- If λ is too big then the regularization becomes too heavy and the network performs poorly both on source and test sets.

Which is an expected behaviour, but at the same time reminds us of the necessity to tune well this hyperparameter.

7.4 Experiment 3: Office-Caltech Dataset

The Office-Caltech Dataset is a popular benchmarking dataset for Domain Adaptation tasks. It consists of 10 categories of images ("bike", "backpack",..) from 4 different source environments: "Amazon", "Webcam", "DSLR" and "Caltech 10".

In this experiment we also consider two environments: We used Caltech-10 as a source set and Amazon images as a target set.

We compare here also the performance of DANN to a standard CNN. For the feature extraction block, we have tried some architectures, and the VGG architecture [22] gave the best results. We give the source and target accuracies in the table 5:

Network Architecture	Source Accuracy	Target Accuracy
VGG DANN	85 %	58 %
Raw VGG	98 %	75 %

Table 5: DANN results on Caltech-10

We can notice that for this experiment, the raw VGG architecture performs better especially for the target set. The

possible reason behind this is that DANN is not well suited for small size datasets: for this dataset we had around 1000 training images which means that we only have around 100 images for each category on average.

7.5 Experiment 4: Coloured MNIST

For our final and main experiment, we use the synthetic dataset of the IRM paper to compare DANN performance to the IRM method.

Let us recall that in this experiment, and for each environment e a majority p_e if the class 0 is colored green and a majority p_e of the class 1 is colored red. Typically, in the training environments p_e is rather close to 1 and in the test environment p_e is close to 0 so that we reverse the correlation between color and labels. We want to see if our classifier relies on this spurious correlation to give predictions.

At the beginning, we only experimented with 2 environments: a source environment e_{train} and a target environment e_{test}. We maintained $p_{e_{test}} = 0.1$ for the target (test) environment, and we noticed that the performance of our network varies greatly when $p_{e_{train}}$ changes. These changes are illustrated in Figure 25 below.

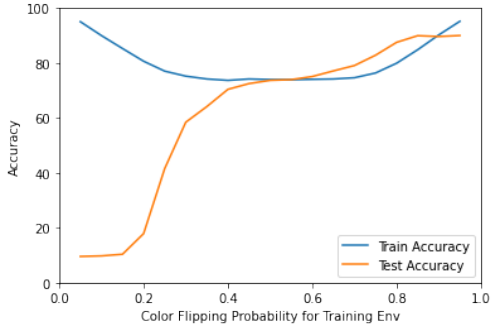


Figure 25: Source and Target accuracy depending on $q_{e_{source}} = 1 - p_{e_{source}}$

We can see that the network performs very poorly on the target set especially when $q_{e_{source}} = 1 - p_{e_{source}} < 0.2$. This behaviour is similar to an ERM network: the network uses the spurious color correlation to make predictions. Then, when $q_{e_{source}}$ increases (i.e. $p_{e_{source}}$ decreases) the DANN performance gets better on the target set until the accuracy on target set becomes equivalent to the accuracy on the source set for $q_{e_{source}} > 0.5$ which is expected since now both source and target sets have similar colour correlations, and also $q_{e_{source}} \sim 0.5$ means that colour is randomly assigned to images.

Many hypotheses can explain the poor performance for high values of $p_{e_{source}}$. First of all, we tried to increase the regularization parameter λ to make the network less prone to depending on spurious correlations. This gave slightly better results but they are still very disappointing.

Besides, the poor performance may be due to the low number of environments. In fact, for the original DANN model, Ganin et al. only consider two domains: a source domain and a target domain. And it may be useful to use as many environments as possible especially if the new environments have different data distributions. This is why we made some slight changes to the original architecture by replacing the final layer of the domain classifier to have as many output neurons as environments and we used a Softmax activation instead of a Sigmoid activation function.

Next, we compared our DANN performance for different numbers of used environments while keeping the same

global average colour correlation probability p_{source}. For example, instead of considering just one source environment with $p_{e_{source}} = 0.7$ with N observations, we consider 3 different environments such that $p_{e1} = 0.55$, $p_{e2} = 0.7$ and $p_{e3} = 0.85$ and each environment has $\frac{N}{3}$ observations. In fact, we want to test if having many environments with different probabilities gives better performances than having a single source environment with an equivalent global colour correlation.

We tested with a global correlation $p_{source} = 0.7$. Due to some instabilities, for each setup (number of environments), we run the experiment 5 times and we calculate the average of the highest target accuracy after 100 epochs. The results are reported in the table below.

Number of source environments	**List of p_{e_i}**	**Source Accuracy**	**Target Accuracy**
1	(0.7)	75 %	56 %
2	(0.55, 0.85)	74 %	59 %
3	(0.55, 0.7, 0.85)	74 %	62 %
5	(0.6, 0.65, 0.7, 0.75, 0.8)	75 %	64 %

Table 6: Evolution of accuracy using more environments

We can conclude from the table above, that having more environments can in fact help give better results (even though, for each setup we have the same global colour probability $p_{source} = 0.7$).

Finally, to compare DANN performance with IRM, we reproduced the same setup of the IRM paper for the coloured MNIST experience where we have 3 environments:

- A first source environment with $p_{e1} = 0.9$
- A second source environment with $p_{e2} = 0.8$
- A target environment with $p_{e_t} = 0.1$

The final results are shown in the table below:

Algorithm	Acc. train envs.	Acc. test env.
ERM	87.4	17.1
IRM	70.8	**66.9**
DANN	84.1	32.4
Random guessing (hypothetical)	50	50
Optimal invariant model (hypothetical)	75	75

Table 7: Comparison with IRM

We can see that DANN performs better than ERM, which means that it doesn't totally rely on spurious correlations. But this result is still disappointing when compared to the accuracy given by IRM. This may be due to various reasons, for example :

- The number of training environments is low. As we saw, with DANN the more environments we have, the better the results that we get.

- The used probabilities p_{e1} and p_{e2} are close to 1 and we have already seen in Figure 25 that in this regime DANN performs very poorly with 2 environments. Therefore we cannot expect a big improvement with only 3 environments.

8 Adversarial Information Factorization

8.1 Motivation

In [23], the authors designed a novel generative model architecture designed to learn representations for images that factor out a single attribute from the rest of the representation. The method is called Adversarial Information Factorization (AIF). As an application, such approach was used to edit the attribute of smiling or not on the dataset celebA [24] : the identity of a given person is independent of whether or not he is smiling or not, then, the attribute of smiling can be changed without changing the identity of the person.
The underlying idea of AIF is to factorize an attribute from an image and separate it from the identity or the important object of that image. We recall that our goal in our framework of several environments is to identify which properties of the training data describe spurious correlations and which properties represent the phenomenon of interest. Thus, we can see the strong connection with AIF. In fact, we can use AIF to factorize the environment on images: in this way, once we get rid from its effect, the operation of predicting the phenomenon of interest become classical and can be performed with basic classifiers.

8.2 Model

The used architecture is a combination of conditional-VAE [25] (variational auto-encoder) and GAN (generative adversarial network) to which we add an auxiliary network. The whole architecture is described in the figure 26. To be

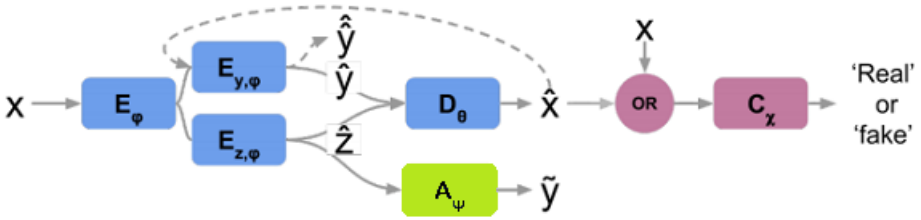


Figure 26: Architecture of the model used for Adversarial Information Factorization

simple, we assume first that we have 2 environments e_0 and e_1. The train set is the set $\{(x_i, y_i) \text{ for } i \in [\![1, N]\!]\}$ where x_i is an input image and the label $y_i \in \{0, 1\}$ indicating the environment e ($y_i = 0$ if $e = e_0$ and $y_i = 1$ if $e = e_1$).
The first part of the network consists of the VAE. So far, for an input image x, the encoder E_ϕ of the VAE, with its two encoding parts $E_{z,\phi}$ and $E_{y,\phi}$, returns the latent encoding variable $\hat{z}$ and the label $\hat{y}$. In fact, $\hat{z}$ captures the identity of the object (phenomenon of interest) while $\hat{y} \in \{0, 1\}$ captures the environment E_ϕ. If $\hat{z}$ contains some information about the label y, that should instead be encoded within the attribute $\hat{y}$. The decoder D_θ tries to reconstruct the image and returns $\hat{x}$.

Then, a discriminator C_ξ is trained to classify decoded samples $\hat{x}$ as fake and samples x from the dataset as real. The decoded samples are re-injected through the encoder (see dashed lines in Figure 26) to get a label $\hat{\hat{y}}$, which is used to contain all the environment information in $\hat{x}$. This updates the decoder of the VAE.
The tricky part of this architecture is the auxiliary network A_ψ (exclusively composed of linear layers) as it is trained to correctly predict y from $\hat{z}$. The encoder E_ϕ is simultaneously updated to encourage A_ψ to make incorrect classification. In other words, the auxiliary network is promoted to not place environment information y in $\hat{z}$. We reach the end of the training once A_ψ is confused to the maximum and thus unable to predict correctly y from $\hat{z}$. Consequently, we manage to separate the latent identity $\hat{z}$ and the environment information y.
To define the objective function of the whole model, we define:

- Loss related to the discriminator C_ξ :

$$L_{C_\xi} = \frac{1}{3}\left[l(y_{\text{real}}, C_\xi(x)) + l(y_{\text{fake}}, C_\xi\left(D_\theta\left(E_\phi(x)\right)\right)) + l(y_{\text{real}}, C_\xi(D_\theta(z))\right]$$

- Loss related to the decoder D_θ :

$$L_{D_\theta} = l(x, \hat{x}) + \beta\, l(y, \hat{\hat{y}}) - \gamma\, L_{C_\xi}$$

- Loss related to the auxiliary network A_ψ:

$$L_{A_\psi} = l\left(y, A_\psi(E_{z,\phi}(x))\right)$$

- Loss related to the encoder E_ϕ :

$$L_{E_\phi} = l(x, \hat{x}) + \beta\, l(y, \hat{\hat{y}}) + \rho\, l(y, \hat{y}) - \gamma L_{C_\xi} + \alpha\, l_{\text{KL}}$$

  where l denotes the binary cross-entropy, $l_{\text{KL}} = KL(q_\psi(z-x)||p(z))$ a regularization term (not represented in the figure 26) and α, β, γ and ρ are positive hyper-parameters. Notice that A_ψ has a min-max objective problem: $min_\psi\ max_\phi L_{A_\psi}$.
  Training then consists of the following steps :

  1. minimize L_{D_θ} with respect to θ
  2. minimize $L_{E_\phi} - L_{A_\psi}$ with respect to ϕ
  3. maximize L_{C_ξ} with respect to ξ
  4. minimize L_{A_ψ} with respect to ψ

8.3 Experiments

To lead a comparative experimentation and make a link with the Invariant Risk Minimization and Domain Adaptation, we tested the Adversarial Information Factorization approach on the coloured MNIST dataset . The experimental setup is described in 6.2.1. For this experience, we choose $\alpha = 1, \beta = 1, \gamma = 0.9, \rho = 1$, 64 for the batch size, 500 for the number of epochs , ADAM as optimizer with a stepsize of 10^{-4}. We report the result in table 8:
We can see that AIF is similar to IRM in the fact that it detects that the color has spurious correlation with the label when $p_{e_{\text{train}}}$ is small. However, the accuracies remain lower than the one of the IRM. Thus, AIF outperforms than ERM but IRM still outperforms both of them.

$p(e_{\text{train}_0})$, $p(e_{\text{train}_1})$, $p(e_{\text{test}})$	Acc. train envs.	Acc. test env.
$(0.2,\ 0.1,\ 0.3)$	62.4 ± 0.5	60.1 ± 1.3
$(0.2,\ 0.1,\ 0.9)$	69.8 ± 0.9	66.1 ± 1.8
$(0.9,\ 0.8,\ 0.1)$	65.8 ± 0.9	59.1 ± 1.1

Table 8: Results of Experiment 1 on colored MNIST (two colors : green and red) with 2 environments : $p(e)$ is the probability of flipping the color in environment e for $e \in \{e_{\text{train}_0}, e_{\text{train}_1}, e_{\text{test}}\}$

9 Relationship between IRM and adversarial methods: Information point-view

At first glance, one could wonder how two approaches of so different fields as IRM and adversarial learning are able to fulfill the same task and can be used in the framework of environments. In fact, while the IRM approach can be performed with any architecture and consists instead in modifying the empirical risk minimization's optimization problem, the adversarial techniques attempt to fool models by supplying deceptive input. However, we will show in this section that the optimization problems of both IRM and adversarial methods treated in this project (Domain Adaptation and AIF) are very close: this link becomes clear if we lead a different point of view of the framework based on the Information Theory, namely the Information Bottleneck Criterion [26].

9.1 Concepts from information theory

Let X, Y and Z random variable with values over $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively.

- **Mutual Information**:
$$I(X,Y) = D_{\text{KL}}(P_{(X,Y)})||P_X \times P_Y)$$
where D_{KL} is the Kullback–Leibler divergence.
- $I(X,Y)$ measures the information that X and Y share: It measures how much knowing one of these variables reduces uncertainty about the other.
- **Conditional Mutual Information**:
$$I(X,Y|Z) = D_{\text{KL}}(P_{(X,Y)|Z)})||P_{X|Z} \times P_{Y|Z})$$
- Since $D_{\text{KL}}(p||q) = 0$ iif $p = q$ almost everywhere, then, $X \perp Y|Z$ iif $I(X,Y|Z) = 0$.
- **Information Bottleneck Criterion**: is a technique in information theory [] that consists in resolving the optimization problem:
$$min_{P_{Z|X}} I(X,Z) - \beta I(Z,Y) \tag{9}$$
where β is a Lagrange multiplier. It is designed for finding the best trade-off between accuracy and complexity when summarizing a random variable X, given a joint probability distribution $p(X,Y)$ between X and an observed relevant variable Y.

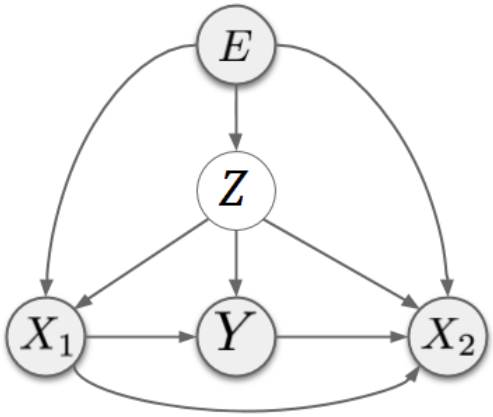


Figure 27: Bayesian Network as a graphical model of a framework of spurious correlations

9.2 IRM as Information Bottleneck criterion

Instead of treating the environment e as an index outside of the structural equation model, we can think of the environment E as an observable random variable, thus as an inherent part of the generative process.
The general generative model of data in the IRM framework can be presented by the graphical model in Figure 27:
In such modelling, E is the environment, X_1 and X_2 the features describing the dataset and Y the label to predict. We assume the existence of a latent variable Z. The edges represents causality in this Bayesian network.
The environment E has an impact on each factor in this model, except the factor $Y|X1, E$. In other words, to follow the assumption of IRM, the relationship of variable Y to its observable causal variable X_1 and hidden variable Z is stable across all environments. In terms of dependencies, we can consider that:

- $Y \not\perp E$
- $Y \not\perp E|X_1$ (because Z induces spurious link between X_1 and Y) but $Y \perp E|X_1, Z$ (as assumed by IRM setup)
- $Y \not\perp E|X_1, X_2$ ($X = (X_1, X_2)$ depends on environment)

The goal of IRM is to find a representation of observable variables $\Phi(X)$, such that $Y \perp E|\Phi(X)$ (ie$I(Y, E|\Phi(X)) = 0$) and such that we can predict Y accurately from $\Phi(X)$. In other words, reduce the uncertainty between Y and $\Phi(X)$ or equivalently maximize $I(Y, \Phi(X))$. Hence, this lead to the optimization problem:

$$\begin{cases} max_{\Phi} I(Y, \Phi(X)) \\ \text{subject to } I(Y, E|\Phi(X)) = 0 \end{cases} \tag{10}$$

The unconstrained form of the Equation 10 could be expressed as:

$$max_{\Phi} I(Y, \Phi(X)) - \beta I(Y, E|\Phi(X))$$

Now, let us go back to the IRM formulation.

An interesting inequality of the mutual information of some random variables x and Y is:

$$\begin{aligned}
I(X,Y) &= \int\int \log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right) p_{(X,Y)}(x,y)dx\,dy \\
&= \int\int \log\left(\frac{p_{(X|Y)}(x)p_Y(y)}{p_X(x)p_Y(y)}\right) p_{(X,Y)}(x,y)dx\,dy \\
&= \int\int \log\left(\frac{p_{(X|Y)}(x)}{p_X(x)}\right) p_{(X,Y)}(x,y)dx\,dy \\
&= \mathbb{E}_{p_{(X,Y)}}\left[\log\left(\frac{p_{X|Y}}{p_X}\right)\right] \\
&= \mathbb{E}_{p_{(X,Y)}}\left[\log\left(\frac{p_{X|Y}q_{X|Y}}{p_X\ q_{X|Y}}\right)\right] \\
&= \mathbb{E}_{p_{(X,Y)}}\left[\log\left(\frac{p_{X|Y}}{q_{X|Y}}\right)\right] + \mathbb{E}_{p_{(X,Y)}}\left[\log\left(\frac{q_{X|Y}}{p_X}\right)\right] \\
&= \mathbb{E}_{p_Y} D_{\mathrm{KL}}(p_{X|Y}||q_{X|Y}) + \mathbb{E}_{p_{(X,Y)}}\left[\log\left(q_{X|Y}\right)\right] - \underbrace{\mathbb{E}_{p_{(X,Y)}}\left[\log\left(p_X\right)\right]}_{-H(X)} \\
&\geq \mathbb{E}_{p_{(X,Y)}}\left[\log\left(q_{X|Y}\right)\right] + H(X)
\end{aligned} \tag{11}$$

Since $I(X,Y) = H(X) - H(X|Y) = H(X) - \mathbb{E}_{p_{(X,Y)}}\left[\log\left(p_{X|Y}\right)\right]$, then, the Inequality 11 becomes tight when $p_{X|Y} = q_{X|Y}\ a.s.$.

To find a lower bound of $I(Y,\Phi(X))$,we take $q_{X|Y}(x) = \frac{p_X(x)}{Z(y)}e^{f(\Phi(X),Y)}$ where the normalization term $Z(y) := \mathbb{E}_X\left(e^{f(\Phi(X),Y)}\right)$,then:

$$I(\Phi(X),Y) \geq \mathbb{E}_e\left(\mathbb{E}_{X^e,Y^e}(f(\Phi(X^e),Y^e))\right) - \mathbb{E}_Y(\log(Z(Y)) = \mathbb{E}_e\mathcal{R}^e(fo\Phi) - \mathbb{E}_Y(\log(Z(Y))$$

We would like to highlight the fact that this inequality is not present in literature and gives us a link with the original formulation of the objective function of IRM.

In literature, other useful inequalities [27] were shown if f is parametrized by θ and we use a log-loss l:

$$\begin{aligned}
I[Y,E|\phi(x)] &\geq \min_\theta E_{x,y}\ell f(y|\phi(x);\theta) - \mathbb{E}_e \min_{\|d\|^2\leq\epsilon} \mathbb{E}_{x,y|e}\ell f(y|\phi(x);\theta+d) \\
&= \min_\theta \mathbb{E}_e\left\{\mathcal{R}^e(f_\theta\circ\phi) - \min_{\|d_e\|\leq\epsilon}\mathcal{R}^e(f_{\theta+d_e}\circ\phi)\right\}
\end{aligned} \tag{12}$$

Thus, by a first order Taylor approximation of $\mathcal{R}^e(f_{\theta+d_e}\circ\phi)$ around θ, and show that, as $\epsilon\to 0$, we get:

$$I[Y,E|\phi(x)] \geq \min_\theta \mathbb{E}_e\|\nabla_\theta\mathbb{E}_{x,y|e}[\ell f(y|\phi(x),\theta)]\|_2 = \min_\theta \mathbb{E}_e\|\nabla_\theta\mathcal{R}^e(f_\theta\circ\phi)\|_2 \tag{13}$$

This recalls us with the gradient penalty of the original formulation of IRM.

9.3 Information theory behind the adversarial approach

9.3.1 Information and GAN objective

The Information Bottleneck Criterion could be seen as a minmax objective function:

$$\max_\phi\{I[Y,\phi(X)] - \beta I[Y,E|\phi(X)]\} = \max_\phi\max_r\min_q \mathbb{E}_{x,y,e}\left[\log r(y|\phi(x)) - \frac{\beta}{1+\beta}\log q(y|\phi(x),e)\right] \tag{14}$$

The key of such inequality is to see that for some random variables X,Y and Z:

$$\begin{aligned}
I(X,Y|Z) &= \int\int \log\left(\frac{p_{(X,Y)|Z}(x,y,z)}{p_{X|Z}(x,z)p_{Y|Z}(y,z)}\right) p_{(X,Y,Z)}(x,y,z)dx\,dy \\
&= \int\int \log\left(\frac{p_{X|(Y,Z)}(x,y,z)}{p_{X|Z}(x,z)}\right) p_{(X,Y,Z)}(x,y,z)dx\,dy \\
&= \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}}{p_{X|Z}}\right)\right] \\
&= \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}\ q_{X|Z}}{p_{X|Z}\ q_{X|Z}}\right)\right] \\
&= \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}}{q_{X|Z}}\right)\right] - \underbrace{\mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|Z}}{q_{X|Z}}\right)\right]}_{\mathbb{E}_{P_Z}(D_{\text{KL}}(p_{X|Z}||q_{X|Z}))\geq 0} \\
&\leq \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}}{q_{X|Z}}\right)\right]
\end{aligned} \tag{15}$$

This inequality is tight if $p_{X|Z} = q_{X|Z}\ a.s.$. Thus,

$$I(X,Y|Z) = min_{q_{X|Z}}\mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}}{q_{X|Z}}\right)\right]$$

Moreover,

$$\begin{aligned}
\mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}}{q_{X|Z}}\right)\right] &= \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{r_{X|(Y,Z)}}{q_{X|Z}}\right)\right] + \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{p_{X|(Y,Z)}}{r_{X|(Y,Z)}}\right)\right] \\
&= \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{r_{X|(Y,Z)}}{q_{X|Z}}\right)\right] + \mathbb{E}_{P_{Y,Z}}(D_{\text{KL}}(p_{X|(Y,Z)}||r_{X|(Y,Z)})) \\
&\geq \mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{r_{X|(Y,Z)}}{q_{X|Z}}\right)\right]
\end{aligned} \tag{16}$$

This inequality becomes tight when $r_{X|(Y,Z)} = p_{X|(Y,Z)}$. Finally, we conclude that:

$$I(X,Y|Z) = min_{q_{X|Z}}max_{r_{X|(Y,Z)}}\mathbb{E}_{P_{X,Y,Z}}\left[\log\left(\frac{r_{X|(Y,Z)}}{q_{X|Z}}\right)\right]$$

With a little bit of math juggling, we get the inequality 14. Such inequality shows that the information bottleneck criterion reassembles to the standard objectives of GANs. Thus, there is a strong connection between information theory and adversarial networks.

9.3.2 Information theory in AIF

In [28], authors introduces the concepts of redundancy and sufficiency in a framework of representation learning where the challenge is to find a distribution $p(z|x)$ that maps data observations x into a representation z, capturing some desired characteristics.

- A representation z of x is **sufficient** for deriving the label y if $I(x,y|z) = 0$. Equivalently, $I(x,y) = I(y,z)$[1]

[1] $I(x,y,z) = I(x,y|z) + I(x,y) = I(y,z|x) + I(y,z)$

- If x_1 and x_2 be two **mutually redundant** views for a target y if $I(x_1, y|x_2) = 0$

Therefore, x_1 and x_2 are two mutually redundant views for a target y and z_1 a sufficient representation x_1,then, $I((x_1, x_2), y) = I(z, y)$.By factorizing the mutual information between x_1 and z_1 , we can identify two components:

$$\underbrace{I(x_1, z_1)}_{\text{predictive information for } x_1} = \underbrace{I(x_1, z_1|x_2)}_{\text{superfluous information}} + \underbrace{I(x_2, z_1)}_{\text{predictive information for } x_2}$$

To adapt these notations with the framework of environments, we can consider x_1 in environment e_1 and x_1 in environment e_2 such that they represent the same object and thus have to be assigned to the same target (for example, the target is the digit in the case of colored MNIST).
Therefore, the sufficiency for y can be guaranteed if the representation z_1 of x_1 is sufficient forx_2, and thus decreasing $I(z_1, x_1|x_2)$ will increase the robustness of the representation by discarding superfluous information. So, we can combine these two requirements using a relaxed Lagrangian objective :

$$\mathcal{L}_1(\theta, \alpha) = I_\theta(z_1, x_1|x_2) - \alpha I_\theta(x_2, z_1) \tag{17}$$

to be minimized with respect to parameters θ denoting the dependency on the parameters of the distribution $p_\theta(z_1|x_1)$. Similarly, a minimal sufficient representation z_2 of the x_2 for x_1 could be described by the relaxed Lagrangian objective:

$$\mathcal{L}_2(\psi, \beta) = I_\psi(z_2, x_2|x_1) - \beta I_\psi(x_1, z_2) \tag{18}$$

to be minimized with respect to parameters ψ denoting the dependency on the parameters of the distribution $p_\psi(z_2|x_2)$. The average of the two loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ can be upper bounded as follows[2]:

$$\mathcal{L}(\theta, \psi, \gamma)) = I_{\theta,\psi}(z_1, z_2) + \gamma D_{\text{SKL}}(p\theta(z_1|x_1)||p_\psi(z_2|x_2)) \tag{19}$$

Hence, the resulted loss looks like the one of the encoder in the architecture used for the AIF method.

10 Discussion and Future Work

This project report briefly introduces some causal inference and domain adaptation notions along with why it is an important field of research especially in deep learning. A notion that stands out in causality is that of invariance. Previous work exploits the invariance of causal mechanisms to try to infer them. We explain some of this work and analyze it critically. We then propose an optimization procedure to find the invariant representation discussed in IRM. We test this optimization procedure on a simple linear case. While these experiments are toy examples with little relation to deep learning, for now, they help us understand better how these methods work and how to improve on them. So naturally the next steps in this project is to further investigate the non-linear case. Domain adaptation is a big and mature field of Machine Learning and we could therefore benefit from the ideas stemming there.

[2] $\gamma = \frac{2}{\alpha+\beta}$

A Derivation of linear regression solutions

We are trying to find α_1 and α_2 such that they minimize the following :

$$\min_{\alpha_1,\alpha_2} E[(Y - (\alpha_1 X + \alpha_2 Z))^2]$$

This is a classic linear regression problem that we know admits a solution. By using the structural equation model we can replace Y and Z with their expressions.

$$\begin{aligned}
&E[(Y - (\alpha_1 X + \alpha_2 Z))^2] \\
&= E[(X + N_1 - (\alpha_1 X + \alpha_2 (X + N_1 + N_2)))^2] \\
&= E[((1 - \alpha_1 - \alpha_2)X + (1 - \alpha_2)N_1 - \alpha_2 N_2)^2]
\end{aligned}$$

Since N_1, N_2 and X are centered independent gaussian random variables we know that

$$\begin{aligned}
&(1 - \alpha_1 - \alpha_2)X + (1 - \alpha_2)N_1 - \alpha_2 N_2 \\
&\sim \mathcal{N}(0, (1 - \alpha_1 - \alpha_2)^2\sigma^2 + (1 - \alpha_2)^2\sigma^2 + \alpha_2^2)
\end{aligned}$$

Since $(1 - \alpha_1 - \alpha_2)X + (1 - \alpha_2)N_1 - \alpha_2 N_2$ is centered, its variance is equal to its second moment. Therefore,

$$E[(Y - (\alpha_1 X + \alpha_2 Z))^2] = (1 - \alpha_1 - \alpha_2)^2\sigma^2 + (1 - \alpha_2)^2\sigma^2 + \alpha_2^2$$

The problem here being convex (quadratic minimization problem) we just need to find a critical point. The gradient of the expression above is

$$\begin{aligned}
&\nabla E[(Y - (\alpha_1 X + \alpha_2 Z))^2] \\
&= \begin{pmatrix} 2(\alpha_1 + \alpha_2 - 1)\sigma^2 \\ 2(\alpha_1 + \alpha_2 - 1)\sigma^2 + 2(\alpha_2 - 1)\sigma^2 + 2\alpha_2 \end{pmatrix}
\end{aligned}$$

Solving for $\nabla E[(Y - (\alpha_1 X + \alpha_2 Z))^2] = 0$ is a linear system that gives us $\alpha_1 = \frac{1}{1+\sigma^2}$ and $\alpha_2 = \frac{\sigma^2}{1+\sigma^2}$

Fixing $\alpha_1 = 0$ or $\alpha_2 = 0$ leaves us with a 1D convex minimization problem. Similar (easier) arguments can be made in those cases.

List of Algorithms

List of Figures

List of Tables

References

[1] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification Presents Challenges for Credibility in Modern Machine Learning. 3

[2] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut Learning in Deep Neural Networks. 3

[3] Bernhard Schölkopf. Causality for Machine Learning. 3

[4] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The Pitfalls of Simplicity Bias in Neural Networks. 3

[5] Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient Starvation: A Learning Proclivity in Neural Networks. 3

[6] David Lopez-Paz. From Dependence to Causation. 3

[7] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. 3, 5, 6, 8

[8] Mingsheng Long, Y. Cao, J. Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. *ArXiv*, abs/1502.02791, 2015. 3

[9] Xavier Glorot, Antoine Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 06 2011. 3

[10] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Penguin UK. 4

[11] Judea Pearl. *Causality.* Cambridge University Press, 2 édition edition. 4

[12] Jonas Peters, Dominik Janzing, and Bernhard Scholkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press. 4

[13] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: Identification and confidence intervals. 78(5):947–1012. 6

[14] Benjamin Aubin, Agnieszka Słowik, Martin Arjovsky, Leon Bottou, and David Lopez-Paz. Linear unit-tests for invariance discovery. page 6. 22, 23, 27

[15] Yo Choe, Jiyeon Ham, and Kyubyong Park. An Empirical Study of Invariant Risk Minimization. 22

[16] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. Out-of-Distribution Generalization via Risk Extrapolation (REx). 22

[17] Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. The Risks of Invariant Risk Minimization. 22, 23, 27

[18] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Society for Industrial Applied Mathematics,U.S., 2019. 23, 27

[19] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 25

[20] Lukas Balles and Philipp Hennig. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. 25

[21] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016. 32

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 35

[23] Antonia Creswell, Anil A. Bharath, and Biswa Sengupta. Conditional autoencoders with adversarial information factorization. *CoRR*, abs/1711.05175, 2017. 38

[24] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild, 2015. 38

[25] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 38

[26] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015. 40

[27] Fabrice Rossi, Amaury Lendasse, Damien François, Vincent Wertz, and Michel Verleysen. Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. *CoRR*, abs/0709.3427, 2007. 42

[28] Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning robust representations via multi-view information bottleneck, 2020. 43