JMLR: Workshop and Conference Proceedings 101:173–187, 2019 ACML 2019

Learning to Augment with Feature Side-information

Amina Mollaysa[1,2] MAOLAAISHA.AMINANMU@HESGE.CH
Alexandros Kalousis[1,2] ALEXANDROS.KALOUSIS@HESGE.CH
Eric Bruno[3] EBRUNO@EXPEDIA.COM
Maurits Diephuis[2] MDIEPHUIS@GMAIL.COM
[1] *Geneva School of Business Administration, HES-SO University of Applied Sciences of Western Switzerland;* [2] *University of Geneva, Switzerland;* [3] *Expedia, Switzerland*

Abstract

Neural networks typically need huge amounts of data to train in order to get reasonable generalizable results. A common approach is to artificially generate samples by using prior knowledge of the data properties or other relevant domain knowledge. However, if the assumptions on the data properties are not accurate or the domain knowledge is irrelevant to the task at hand, one may end up degenerating learning performance by using such augmented data in comparison to simply training on the limited available dataset. We propose a critical data augmentation method using feature side-information, which is obtained from domain knowledge and provides detailed information about features' intrinsic properties. Most importantly, we introduce an instance wise quality checking procedure on the augmented data. It filters out irrelevant or harmful augmented data prior to entering the model. We validated this approach on both synthetic and real-world datasets, specifically in a scenario where the data augmentation is done based on a task independent, unreliable source of information. The experiments show that the introduced critical data augmentation scheme helps avoid performance degeneration resulting from incorporating wrong augmented data.

1. Introduction

Data augmentation is often used in image recognition problems as an implicit way to protect from overfitting by increasing the dataset size. It is typically done using transformations, such as altering the intensities of RGB channels, cropping, rotation, flipping etc (Krizhevsky et al., 2012), which we know from domain knowledge that do not change the label of the instances generated by the augmentation.

In problems other than imaging it can be considerably more challenging to establish label preserving transformations. For example, in molecule datasets, used for pharmaceutical drug discovery, it is unclear what, if any, transformation can be applied to compounds, such that their molecular properties remain intact. Most of the work in data augmentation has taken place within imaging application and does not transfer in other domains. The transformations that are used are global transformation that are applied on all the training data. However there can be class preserving transformations that are instance dependent. In addition it is not obvious that the resulting augmented data follow the distribution from which the training data are sampled, (Zhang et al., 2017), and can thus introduce significant bias which would harm learning.

© 2019 A. Mollaysa, A. Kalousis, E. Bruno & M. Diephuis.

In this paper we develop data augmentation methods which do not introduce spurious artefacts and instances that deviate from the training data distribution. In addition we explore local, instance dependent, transformations for augmentation instead of the fixed universal ones typically used. We consider learning problems in which to a certain extend we have available domain knowledge in the form of what we call feature side-information. Feature side-information can describe either feature properties and/or feature relations. Feature side-information has been used in recommendation systems (Rao et al., 2015), to describe users; in drug efficiency prediction problems, providing physico-chemical molecule properties; in imaging problems providing additional information about the pixels (Krupka and Tishby, 2007); as well as in text classification problems describing additional properties of the words (Mollaysa et al., 2017). We will adopt the framework of Mollaysa et al. (2017) in which feature-side information comes in the form of vectorial descriptions of additional feature properties which are then used to define feature similarities. Their framework assumes that features that are similar, according to the domain knowledge, should have a similar effect on the model output. This allows to generate, hopefully, label-preserving instances and improve the predictive performance. However, the available domain knowledge might be of poor quality, incomplete, irrelevant, or simply wrong. As a result the generated instances instead of providing useful additional training instances, provide information that is detrimental to learning. We will show how we can learn to qualify the augmented instances and filter them out if their are detrimental to learning, learning along the way how to do safe data augmentation.

2. Related Work

Data augmentation enlarges the dataset using label preserving transformations in order to improve the learning performance (Baird, 1992). Many of the existing approaches are restricted to imaging problems and rely on the application of geometric and/or color augmentation which are known by domain not to affect the class label (Krizhevsky et al., 2012; Perez and Wang, 2017). An alternative and generic approach, which does not use domain knowledge, is the addition of small amounts of noise to stabilise the learned model in small neighborhoods around the training instances (Bishop, 1995).

Hauberg et al. (2015) proposed a model that learns the data augmentation transformation instead of providing it explicitly. They use a generative model to learn a distribution of transformations per class, under the assumption that spatial transformations between images belong to a large class of diffeomorphisms. They then sample an image from the training data and a transformation from the learned distribution to generate a new within-class image. AutoAugment (Cubuk et al., 2018) learns a data augmentation policy that produces sequences of operations by solving a discrete search problem, using as reward the performance on a validation set. While learning to augment has shown great potential, so far it has been mostly restricted to imaging problems, is computationally expensive, and operates in a global manner by applying the same augmentation scheme to all instances in a batch. Instead, an instance-specific transformation has the potential to bring useful diversity in the training data.

Generative models such as VAEs (Kingma and Welling, 2013) and GANs (Goodfellow et al., 2014) can be used to synthesize more data. To gain more fine level control over

the generated samples many alternatives have been prosposed which enable us to generate within class samples, such as the Conditional GANs (Mirza and Osindero, 2014), Render GANs (Sixt et al., 2016), CycleGANs (Zhu et al., 2017a), Conditional VAEs, and their combinations e.g. CVAE-GAN (Bao et al., 2017). Antoniou et al. (2017) use a GAN to augment data to generate within class images without requiring label information. Instead of conditioning on the label as CGAN does, they condition both the discriminator and the generator on an image itself from an identical class. Such conditioning eliminates the need for feeding the label to the model, and enables them to generate samples from unseen classes. Zhu et al. (2017b) used CylceGANs to generate samples for the minority classes to solve the class imbalance problem in classification tasks.

Lemley et al. (2017) do safe data augmentation, they use two networks, where a first network learns the best data augmentation to train second network for a downstream classification task. The loss of the latter network is used to guide the former in learning the data augmentation. The approach is tied to the classification and it is not clear how to extend it beyond that. The quality of the augmented data is not considered as long as they capture the most discriminative features features that can help the downstream classification network with its task.

Learning the data augmentation, is related to the semi-supervised setting in which we try to learn the labels for the unlabeled data. Li and Zhou (2010) and Li and Zhou (2015) learn labels for the unlabeled data such that there is no performance deterioration compared to an inductive SVM. Data augmentation is also related to adversarial training where the adversarial examples are added to the model to improve its robustness against such attacks. (Miyato et al., 2017) seek to identify the adversarial directions in a small neighborhood around the input data and smooth out the model output along those directions.

The above approaches either use a known label preserving transformation or start from a zero prior and learn the transformation from the data. Instead, we focus on the case that we have domain knowledge, that can be incomplete or non-reliable, which we wish to exploit in order to guide data augmentation process, while aleviating the detrimental effects that can be introduced due to its non-reliability.

3. Critical data augmentation

We consider a supervised learning setting in which we are given a set of input-output pairs $D = \{(\mathbf{x}_n, \mathbf{y}_n) \in (\mathcal{X} \times \mathcal{Y}), \mathcal{X} \subseteq \mathbb{R}^d, \mathcal{Y} \subseteq \mathbb{R}^m, n \in \mathbb{N}_n\}$, sampled i.i.d. according to some unknown probability distribution p. In addition we are also given a matrix $\mathbf{Z} : d \times c$, the ith row of which, $\mathbf{z}_i$, contains a description of the ith feature and is provided by domain knowledge. We call $\mathbf{Z}$ the feature side-information matrix. We want to learn a function $\phi : \mathcal{X} \rightarrow \mathcal{Y}$ which minimizes the expected loss $E[L(\phi(\mathbf{x}), \mathbf{y})]$, by using both the input-output data D and the feature side-information matrix $\mathbf{Z}$, figure 1. We do so by using $\mathbf{Z}$ to augment the training data following Mollaysa et al. (2017). Unlike (Mollaysa et al., 2017) where are all augmented instances are used in training we construct an instance based quality control mechanism which learns to reject augmented instances that are produced from non-relevant or non-reliable information. We call this quality control mechanism Critical Data Augmentation (CDA). In the next two sections we describe how we use the side-information for data augmentation and how to learn the quality control mechanism.

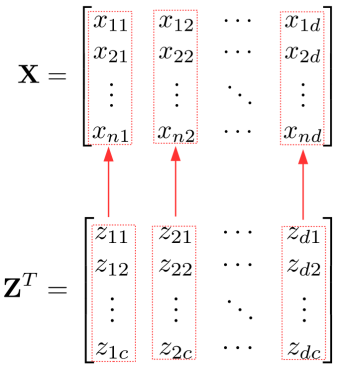

Figure 1: Input instance matrix $\mathbf{x}$ and feature side information matrix $\mathbf{z}$. Each column of $\mathbf{z}$ describes the properties of the one of the feature.

3.1. Data Augmentation with Feature Side-information

We leverage the information provided by $\mathbf{Z}$ to do data augmentation and improve the generalization performance. Given two features i, j, with side-information vectors $\mathbf{z}_i, \mathbf{z}_j$, we define a similarity matrix $\mathbf{S}$ which provides the pairwise similarity of the different features. The s_{ij} element of $\mathbf{S}$ gives the similarity of the i and j features and is computed with some similarity function, e.g. $s_{ij} = \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)$. To do data augmentation we follow the assumption introduced in Mollaysa et al. (2017) which states that if two features are similar then they should affect the model output in the same manner. We use this assumption to define an output preserving transformation $T : \mathbf{x} \rightarrow \hat{\mathbf{x}}$, which takes a real training instance $\mathbf{x} \in \mathcal{X}$ and produces $\hat{\mathbf{x}}$ by modifying the values of similar feature pairs. We define T as:

$$T(\mathbf{x}, s_{ij}, \lambda) = \hat{\mathbf{x}} = \mathbf{x} - \lambda \mathbf{e}_i + \lambda \mathbf{e}_j, \;\; \mathbf{x} \in \mathcal{X}, s_{ij} \in \mathbf{S}, \lambda \in \Omega, \tag{1}$$

where $\mathbf{e}_i$, $\mathbf{e}_j$, are the d-dimensional unit vectors with the ith and jth dimensions respectively equal to one. λ is a scalar representing the size of the augmentation and Ω is computed from the data distribution and gives the range of the augmentation size.

Given T we define $\mathbf{X}_n^c$, the set of augmented instances the originate from $\mathbf{x}_n$, as:

$$\mathbf{X}_n^c = \{\hat{\mathbf{x}}_n | \mathbf{x}_n - \lambda \mathbf{e}_i + \lambda \mathbf{e}_j, \lambda \in \Omega, s_{ij} \geq c\}, \tag{2}$$

c is a similarity threshold which determines the level of similarity above which why can apply the perturbation. Perturbing the values of features that have a high similarity should produce an augmented instance the output of which should be similar to the original instance:

$$\boldsymbol{\phi}(\mathbf{x}_n - \lambda \mathbf{e}_i + \lambda \mathbf{e}_j) \approx \mathbf{y}_n. \tag{3}$$

Such a transformation T when applied on feature pairs with high similarity results in output-preserving data augmentation, i.e., all generated $\hat{\mathbf{x}} \in \mathbf{X}_n^c$ have the same output $\mathbf{y}_n$. We can thus exploit in learning both the original and the augmented data, with the latter having

a stabilization effect on changes of values of similar feature pairs, through the following learning problem:

$$\min \sum_{(\mathbf{x}_n,\mathbf{y}_n)\in D} [L(\phi(\mathbf{x}_n),\mathbf{y}_n) + \gamma \sum_{\hat{\mathbf{x}}_n^l \in \mathbf{X}_n^c} L(\phi(\hat{\mathbf{x}}_n^l),\mathbf{y}_n)]. \tag{4}$$

γ is a regularization hyper-parameter controlling the importance of data augmentation.

So far we guide data augmentation by the feature similarity. The only constraint that we impose above is that the feature similarity should be higher than the c threshold in order to perform data augmentation for a given feature pair. However there is a risk here that we introduce augmentations that are detrimental to learning when the domain knowledge is partial or of low quality. The similarity matrix might not reflect correctly the feature similarities for the learning task at hand, some of the feature similarities might be wrong or simply irrelevant. Even if the similarities are correct the size of the augmentation can vary within the input space; depending on the landscape of the true model we can have regions in the input space that can afford large augmentation size (the true model is flat in these regions) or regions in which it is very sensitive, thus allowing only very small augmentations. These observations point to the need to redefine data augmentation as a local transformation in the input space. Learning local, output-preserving, transformations is equivalent to learning a probability distribution of $p(\mathbf{x}|\mathbf{y}_i)$ for every $\mathbf{y}_i$ in the training set, which we can use subsequently to sample augmented instances $\hat{\mathbf{x}} \sim p(\mathbf{x}|\mathbf{y}_i)$. When we have limited training data this is not possible. Thus instead of learning the distribution, we will use the feature side-information to propose augmented samples and learn to filter those that do not align with the underlying data distribution. In the next section we describe how we will perform the filtering.

3.2. Filtered data augmentation

In imaging problems it is relatively easy to asses the quality of data augmentation strategies, such as rotation and translation, simply by visual inspection. Obviously in the setting that we consider this is not possible. If we had access to the true model, let us denote it by $\phi^*(\mathbf{x})$, we could use it to control which augmentations are correct. For example given a pair of features with very high similarity and an appropriate augmentation size the following approximate equality would hold:

$$\boldsymbol{\phi}^*(\mathbf{x}) \approx \boldsymbol{\phi}^*(\mathbf{x} + \lambda\mathbf{e}_i - \lambda\mathbf{e}_j), \tag{5}$$

We do not have access to $\boldsymbol{\phi}^*(\mathbf{x})$ but we can use the training data to get an approximation $\tilde{\boldsymbol{\phi}}$ of it:

$$\tilde{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \sum_{(\mathbf{x}_n,\mathbf{y}_n)\in D} L(\phi(\mathbf{x}_n),\mathbf{y}_n) + \gamma R(\phi), \tag{6}$$

where $\gamma \in \mathbb{R}_{\geq 0}$ is the regularization hyper-parameter and $R : \phi \to \mathbb{R}_{\geq 0}$ is a regularizer such as L_2. We can use $\tilde{\boldsymbol{\phi}}$ to control how the model reacts to perturbations of similar features. If the two features are truly similar, and the augmentation size is correct, then the model should have similar response on the original and augmented data. Therefore, we can use the

$\tilde{\boldsymbol{\phi}}(\mathbf{x})$ to control the quality of the augmented data. Intuitively if the difference of the model output between the original instance and its augmentation, difference which we define as

$$d = ||\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x} + \lambda\mathbf{e}_i - \lambda\mathbf{e}_j)||, \tag{7}$$

is smaller than some threshold parameter ϵ then we can accept this augmentation. If on the other hand the distance is larger than ϵ then this implies that the two features i and j affect the model in a different manner and we should reject the augmentation.

We proceed by generating for each training instance $(\mathbf{x}_n, \mathbf{y}_n) \in D$, the set of its augmentations $\{(\hat{\mathbf{x}}_n^k, \mathbf{y}_n) | \hat{\mathbf{x}}_n^k \in \mathbf{X}_n^c, k := 1, 2, 3 \dots K\}$. We denote by $d_n^k = ||\tilde{\boldsymbol{\phi}}(\mathbf{x}_n) - \tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_n^k)||$ the model's output difference between the original instance $\mathbf{x}_n$ and its $\hat{\mathbf{x}}_n^k$ augmentation. We need to establish a mechanism that will exclude $\hat{\mathbf{x}}_n^k$ from the learning if $d_n^k > \epsilon$ and retain it otherwise. We do so by using the $\text{ReLu}(\epsilon - d_n^k)$ function which will act as a gate, returning 0 when $\epsilon \leq d_n^k$ a fact that we will use to exclude $\hat{\mathbf{x}}_n^k$ from the learning. Concretely we define the following learning problem:

$$\min_{\boldsymbol{\phi}} \sum_{(\mathbf{x}_n, \mathbf{y}_n) \in D} [L(\phi(\mathbf{x}_n), \mathbf{y}_n) + \gamma \sum_{\hat{\mathbf{x}}_n^k \in \mathbf{X}_n^c} \frac{1}{\epsilon - d_n^k} \text{ReLu}(\epsilon - d_n^k) L(\boldsymbol{\phi}(\hat{\mathbf{x}}_n^k), \mathbf{y}_n)], \tag{8}$$

The first term in the outer sum is the loss on the original instance while the second term is the loss over its set of augmentations. Note how $\text{ReLu}(\epsilon - d_n^k)$ together with $\frac{1}{\epsilon - d_n^k}$ controls whether the $\mathbf{x}_n^k$ augmented instance will contribute to the loss, $d_n^k \leq \epsilon$, or will be ignored, $d_n^k > \epsilon$. As a result, we train the model using the original training data and a part of the augmented, by rejecting the wrong augmentations. We treat ϵ as a hyper-parameter to tune; we determine its range of values through the distribution of $d_n^k = ||\tilde{\boldsymbol{\phi}}(\mathbf{x}_n) - \tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_n^k)||$ in the validation. Note that if the selected value of ϵ is $\epsilon = inf$, this means that all augmentations are correct and we include them in the training. This also means that the $\mathbf{S}$ matrix is reliable and the size of augmentation is correct. In this case the equations (8) and (4) are equivalent. If on the other hand the optimal ϵ is 0 this means that there is no valid augmentation and we can only train with the real data i.e. eq (6).

4. Optimization

The model we described above can be used both for classification and regression. We use a standard feed-forward neural network to learn ϕ. We first train the network with the existing training data with an early stopping step to get $\tilde{\phi}$. We then initialize our network with $\tilde{\phi}$ in the eq (8) and start training using both real and and augmented data. We do the data augmentation on the fly. We augment each instance $\mathbf{x}_n$ in the mini batch by uniformly picking a pair of similar features and applying the T transformation and let the ReLu determine which augmented instances will be filtered as described above. We can also update the function approximator $\tilde{\boldsymbol{\phi}}$ during training with the newly learned ϕ. We tested both using a fixed $\tilde{\boldsymbol{\phi}}$ during the full training of the main objective eq (8) as well as using the previous update of ϕ as the new $\tilde{\boldsymbol{\phi}}$ at each update of main objective (8). The latter performs better, we thus used it in all our experiments.

In regression problems $\phi(\mathbf{x}_n)$ is the network output with identity activation on the last layer. Therefore the objective function of eq (8) using an L_2 loss is enough to learn

nearly identical outputs for both the original instance and its augmentations by minimizing $\|y_n - \phi(\mathbf{x}_n)\|^2 + \gamma\|y_n - \phi(\hat{\mathbf{x}}_n)\|^2$.

In classification problems ϕ is given by the last layer's pre-activation units. These are followed by a softmax activation. As we augment by perturbing two features at a time once we apply the softmax cross entropy loss, we observe that the difference between $L_\phi(\boldsymbol{\phi}(\mathbf{x}_n), \mathbf{y}_n)$ and $L_\phi(\boldsymbol{\phi}(\hat{\mathbf{x}}_n), \mathbf{y}_n)$ becomes very small. This makes the optimization harder to guide the model to learn to output similar outputs for both original $\mathbf{x}_n$ and its augmentation $\hat{\mathbf{x}}_n$. Therefore we also add the constraint $\|\phi(\mathbf{x}_n) - \phi(\hat{\mathbf{x}}_n)\|^2$ in the objective. Thus the second term of the objective function given in eq (8) now becomes:

$$\sum_{\hat{\mathbf{x}}_n^k \in \mathbf{X}_n^c} \frac{1}{\epsilon - d_n^k} \text{ReLu}(\epsilon - d_n^k)(\gamma_1 L(\boldsymbol{\phi}(\hat{\mathbf{x}}_n^k), \mathbf{y}_n) + \gamma_2 \|\phi(\mathbf{x}_n) - \phi(\hat{\mathbf{x}}_n^k)\|^2). \tag{9}$$

5. Experimental setup

We compare our method, Critical Data Augmentation (CDA), against two baselines. We train the first baseline only on real training data using no augmentation, this corresponds to setting $\epsilon = 0$ in the objective function of eq (8); we denote this baseline as NN. We train the second baseline using the real training data and their respective augmentations with no rejection, i.e. setting $\epsilon = \inf$ in the objective function of eq (8), this is essentially the method introduced in (Mollaysa et al., 2017). We denote this latter baseline as Data Augmentation (DA). All network architectures are the same. We perform experiments on synthetic data that we designed specificly, Fashion MNIST and real world data. The code[1] for the models are developed in Tensorflow and can be found here.

5.1. Experiments on synthetic data

We want to evaluate how critical data augmentation performs in the presence of noisy domain knowledge. We design the datasets so that groups of features give rise to latent factors which determine the output value. The feature side information provides information on which group a feature belongs to and gives rise to a respective binary similarity matrix. We then distort the similarty matrix with different levels of noise and experiment with the different methods.

More precisely we generate the synthetic data as follows. We uniformly sample instances from $\mathbb{R}^d$ and generate the instance matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$. We cluster features to p clusters ($p < d$). We use the clusters to define a latent space; each cluster gives rise to a latent feature. The value of a latent feature is the sum of the values of the features that belong to its cluster. On the latent space we apply a linear transformation that projects to a space of lower dimensionality m. On this space we apply an element-wise sigmoid and the final class assignment is given by the index of the maximum sigmoid value. This generates us output matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$. As already mentioned the feature cluster membership gives rise to the similarity matrix $\mathbf{S}$; the similarity S_{ij} of the i, j features is one if they are in the same

1. https://anonymous.4open.science/r/c4538ad2-89d7-4c2b-aef9-d32f9b5021be/

cluster and zero otherwise. We randomly corrupt zero entries and set them to one, thus wrongly indicating two features as similar while in reality they are not. We only do this single direction perturbatio because by construction our method can only learn to reject augmentations over features which are wrongly indicated as similar, but it cannot uncover similarities that are wrongly missed. We set $n = 5000$, $d = 3000$, $p = 600$ and $m = 5$, which produce a similarity matrix $\mathbf{S}$ with 0.19% of nonzero elements. We randomly corrupt 0.10% and 0.20% of the zero entries of the similarity matrix. The pairwise data augmentation we provide is only one type of augmentation we can consider. The rejection mechanism is generic and does depend on it. We thus also experiment with a simpler data augmentation strategy in which we simply add Gaussian noise on the inputs, i.e. $\hat{\mathbf{x}}_n = \mathbf{x}_n + \eta$ where $\eta \sim \mathcal{N}(0, \sigma)$. We apply such perturbation on every intances in the artificially generated instances matrix $\mathbf{X}$ to generate the augmented version $\hat{\mathbf{x}}_i$ for $\mathbf{x}_i \in \mathbf{X}$.

The goal of this experiment is to check if the rejection mechanism can kick in and learn to filter out some of these random augmentations. We tune σ in the set $\{0.01, 0.02, \ldots, 0.10\}$ and the best value is obtained at $\sigma = 0.02$. To learn ϕ we use a three layer network, with 100 and 50 hidden units in the first and second layer. We use ReLU on the hidden layers as the activation function, followed by batch normalization (Ioffe and Szegedy, 2015). We use the softmax cross-entropy as the loss function and Adam (Kingma and Ba, 2014) for the gradient update. We set the maximum number of epochs to 2000 and do early stopping.

We tune the filtering parameter ϵ on a set of values obtained from the histogram of $d = \|\tilde{\phi}(\mathbf{x}) - \tilde{\phi}(\hat{\mathbf{x}})\|^2$ which we establish on the validation set. We also include zero and infinity in these values in order to check if the model reduces to one of the two baselines by choosing to reject all augmentations, i.e. $\epsilon = 0$, or accepting all of them, i.e. $\epsilon = \infty$. We provide the values[2] from which we tune ϵ in the code. We tune the regularizer parameters γ_1, γ_2 in $\{0.01, 0.1, 0, 10, 100\}$. We split the data to a train, validation, and test set, respectively with 60%, 20%, 20% of the total instances. We tune all hyperparameters based on the performance on the validation set. We report the test set accuracy of the different methods in table 1. The best result of the hyper-parameters are obtained at when $\gamma_1 = 1, \gamma_2 = 100$, $\epsilon = 0.025$ for the first two dataset in tabel 1 and $\gamma_1 = 1, \gamma_2 = 100$, $\epsilon = 0.01$, $\sigma = 0.02$ for the last dataset. We establish the statistical significance of the results using a McNemar's test with a p-value of 0.05.

As it is obvious from the results in table 1, the use of a rejection mechanism that qualifies the utility of the augmentations brings a statistical significant performance improvement with respect to the baseline that unconditionally accepts all augmentation, as well as with respect to the baseline that does no augmentation. As expected the performance gap increases in favor of CDA with the noise level in the similarity matrix. In addition the rejection mechanism kicks in with both types of augmentation, i.e. whether we are doing similarity based pairwise augmentation or whether we are injecting random noise on the inputs.

The ϵ parameter controls the level of rejection of augmentation, $\epsilon = 0$ rejects all augmentation collapsing the model to the standard NN baseline and $\epsilon = \infty$ accepts all augmentations collapsing the method to the DA baseline. In order to get a better understanding of

2. This values are chosen according to the histogram of $d = \|\tilde{\phi}(\mathbf{x}) - \tilde{\phi}(\hat{\mathbf{x}})\|^2$. For artificial dataset, we set the set where we chose ϵ as $\{0, 0.0001, 0.001, 0.01, 0.012, 0.013, 0.014, 0.015, 0.016, 0.017, 0.018, 0.019, 0.02, 0.025, 0.023, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06, 0.065, 0.070, 0.075, 1, 10, 100, 10000\}$

Dataset	CDA	DA	NN
S-noise level 0.10%	56.4+/+	56.1 +	50.4
S-noise level 0.20%	55.2 +/+	54.4+	50.4
$\hat{\mathbf{x}}_n = \mathbf{x}_n + \eta \ \eta \sim \mathcal{N}(0, \sigma)$	52.7+/+	51.8+	50.4

Table 1: Classification accuracy and performance comparisons on synthetic problems. CDA: training with filtered augmentations. DA: training with all augmentations. NN: training with no augmentations. The $+, -$ and $=$ signs give the McNemar's statistical significance test results of the comparison of the performance of a given model to those on its right. +/=/- refer to significantly better/equal/worse.

the performance of CDA as a function of ϵ we plot in figure 2 the validation set accuracy as a function of ϵ. As we clearly see there is an optimal rejection threshold, which typically will be a function of the dataset, and affects the performance in a significant manner.

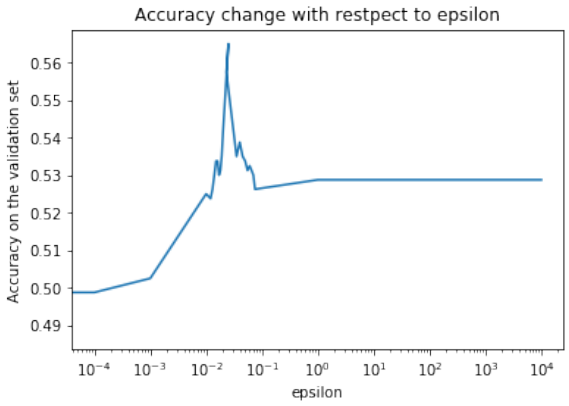

Figure 2: Validation set accuracy for different values of ϵ, **S**-noise level=10%. When $\epsilon = 0$, the model is equivalent to training using onle the real instances. When $\epsilon = \infty$, the model accepts all augmentations and trains on both the real and augmented data.

5.2. Text document classification

To evaluate the performance of the different methods on a set of real world datasets we use the collection of document classification datasets provided in (Kusner et al., 2015). The datasets are: BBC sports articles (BBCSPORT) labeled as one of athletics, cricket, football, rugby, tennis; tweets labeled with sentiments'positive', 'negative', or 'neutral' (TWITTR); recipes labeled by their region of origin (RECIPE); of medical abstracts labeled by different cardiovascular disease groups (OHSUMED); sentences from academic papers labeled by

Date set	n	d	Unique words(avg)	m
BBCsport	590	9759	80.9	5
Twitter	2486	4076	6	3
Classic	5675	7628	34.5	4
Amazon	6400	4502	28.8	4
20NEWS	11293	6859	51.7	20
Recipe	3496	4992	44.7	15
Ohsumed	3999	7643	50	10
Reuter	5485	5939	33	8

Table 2: Dataset description

publisher name (CLASSIC); Amazon reviews labeled by product category (AMAZON); news dataset labeled by the news topics (REUTER); news articles classified into 20 different categories (20NEWS). We removed all the words in the SMART stop word list (Salton and Buckley, 1988). To speed up training, we removed words that appear but a few times over all the documents of a dataset. Concretely, in 20NEWS we reduced the dictionary size by removing words with a frequency less or equal to three. In the OHSUMED and CLASSIC datasets, we removed words with frequency one and in the REUTER dataset words with a frequency equal or less than two. In table 2 we give a description of the final datasets on which we experiment including the number of classes m and the average number of unique words per document.

We represent documents as bag-of-words, where obviously the features are the words. We use the word2vec representations of the words, (Mikolov et al., 2013), as the feature side-information. The word2vec representations reflect a word's semantic and syntactic meaning. If two words are semantically similar, their word2vec representations will be also similar each other. We apply a heat kernel on the word2vec representation to obtain the feature similarity matrix S, i.e. $S_{ij} = \exp(-\frac{1}{2\sigma^2}(\mathbf{z}_i - \mathbf{z}_j)^T(\mathbf{z}_i - \mathbf{z}_j))$, $\mathbf{z}_i$, $\mathbf{z}_j$ are the word2vec representations. We set $\sigma = 1.15$ and $c = 0.5$. This means that we only consider two features, i, j, as similar when $s_{ij} \geq 0.5$. We tune ϵ from the set $\{0, 0.05, 0.3, 0.7, 1, 1.5, 2, 2.5, 10, 10000\}$ and the regularisation parameters γ_1, γ_2, from the set $\{0.01, 0.1, 0, 10, 100\}$.

We divide the datasets in train, validation, test sets with the same 60%, 20%, 20% proportion as in the synthetic datasts, except the 20NEWS, OHSUMED, RECIPE datasets that come with their predefined train-test separation. We use the same optimization and parameter tuning setting as the one in the synethetic experiments. With respect to the network architectures we changed the network hidden layer size to 500 and 100 hidden units in the first and second hidden layer respectively.

We provide the performance results in Table 3. The tuned best values of the hyper-parameters that are used to generate the results are given in Table 4. CDA is significantly better than DA in five out of the eight datasets, significantly worse in two and equivalent once. In addition it also outperforms NN in a statistically significant manner in five out of

Date set	CDA	DA	NN
BBCsport	99.32+/=	97.97-	99.32
Twitter	74.43+/+	73.15-	73.47
Classic	96.96+/+	96.82=	96.82
Amazon	94.18 +/-	93.06 +	934.25
20NEWS	80.12+/=	79.76=	79.63
Recipe	64.18 +/+	63.38=	63.15
Ohsumed	65.26-/+	65.45+	65.04
Reuter	96.93+/=	96.71-	96.98

Table 3: Classification accuracy and performance comparisons on document datasets. Interpretation of column names and +/ = / = is the same as in table 1.

Date set	ϵ	γ_1	γ_2	Date set	ϵ	γ_1	γ_2
BBCsport	0.7	10	10	20NEWS	0.7	100	100
Twitter	2.5	100	10	Recipe	0.7	0.1	1
Classic	1.5	1	0.1	Ohsumed	0.3	10	10
Amazon	0.7	10	10	Reuter	0.7	1	0.1

Table 4: The best hyperparameter values that are used to generate results in Table 3.

the eight datasets, for the remaining three there is no performance difference. The results provide clear evidence that qualifying the quality of augmentations, instead of uncriticically accepting all of them, can bring important performance improvements.

A possible explanation for this phenomena is the fact that the text data is quite sparse. Furthermore, it is constrained by the definition of T and the fact that feature values in this particular dataset can't be negative. This means that critical data augmentation is restricted to feature pairs where at least one of them is none zero. Furthermore, when we do the augmentation we only consider most similar feature pairs where $s_{ij} \geq 0.5$ (around 30%) according to the similarity matrix. The word2vec representation is rather stable and gives global semantic and synthetic similarity of the words. Therefore, the similarity measure is given by the similarity matrix rather correct and there is very little probability that the similarity matrix actually contains totally wrong information.

5.3. Fashion-MNIST

To further investigate the effect of the proposed data augmentation scheme, we tested on Fashion MNIST (Xiao et al., 2017). This dataset includes 60k training and 10k test instances. We further separated the training data so that we have 50k, 10k, 10k number

Dataset	CDA	DA	NN
Similar feature pairs augmentation	87.63=/+	87.37+	84.35
Gaussian noise perturbation $\eta \sim \mathcal{N}(0, \sigma)$	88.24+/+	86.73+	84.35

Table 5: Classification accuracy and performance comparisons on Fashion MNIST. The interpretation of column names and +/ = / = is identical to Table 1.

of instances in the training, validation, and test set respectively. Two different types of transformations are considered. In the first type of transformation, we consider X-coordinate and Y-coordinate of each pixel as the feature side-information. In Fashion MNIST, each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels. This gives us a feature side-information matrix $\mathbf{Z}_{784\times 2}$, whose each element $\mathbf{z}_i = (x_i, y_i)$ represents the x_i and y_i coordinate of the i th pixel. The similarity matrix is then calculated according to the Euclidean distance of the pixels' coordinates. Two pixels that have a Euclidean distance $\|\mathbf{z}_i - \mathbf{z}_j\| \leq 8$ are considered to be similar, i.e., $s_{ij} = 1$ while the rest is considered dissimilar, i.e., $s_{ij} = 0$. We augment each image in the mini batch during training by randomly selecting a feature pair and do the transformation defined in eq (1). In the original dataset, the pixel values are between 0 and 255. Prior to training, we normalize the input data to follow a standard normal distribution. As the pixel values can be negative, when we do the augmentation, we let the subtracted value λ from pixel i to be its current value. To increase the effect of augmentation, during training, for each instance, we augmented 5 similar feature pairs, that are randomly selected from the similarity matrix, instead of only one pair. The filtering parameter ϵ is tuned from a set $\{0.02, 0.06, 0.1, 0.3, 0.5, 0.8, 1, 100\}$. The best value is obtained at $\epsilon = 0.3$

For the second type of transformation, we consider Gaussian additive noise, where the parameter σ is tuned from a set $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ and ϵ is from $\{0, 0.3, 0.6, 1, 1.5, 2, 3, 100\}$. As our focus is not to find the best network architecture, but the effect of the augmentation scheme, we take the most simple feed forward network with one hidden layer of size 128 to see if the augmentations improve the generalization performance over the baseline model. Note that for the vanilla data augmentation (model DA), where we do not apply the filtering process, but take all the augmentation in during training. The best hyper parameter for the Gaussian additive noise is archived at $\sigma = 0.2$. On the other hand, for the critical data augmentation model, CDA, the best parameter is chosen as $\sigma = 0.4$, $\epsilon = 3$. The result is displayed in Table 5. What we observe is that both data augmentation schemes significantly improve the generalization performance compared to training with only existing data. Critical data augmentation has a significant impact on the Gaussian additive noise distortion such that it allows us to augment with higher variance noise compared to the DA model.

6. Conclusion and future works

Data augmentation provides and effective and easy way to increase the training sample size and hopefully, if done correctly, lead to better performing models. In domain other then imaging where it is straightforward to define valid augmentations defining what a proper augmentation is, is far from obvious. Typically potential data augmentations are defined on the basis of domain knowledge. However, very often such knowledge is incomplete or even wrong. Uncritically accepting augmentations based on such domain knowledge can have detrimental effects on the predictive performance instead of improving it as the goal is. The alternative approach that relies on generative models is not only computational expensive but practically impossible when the available training data are limited.

In this paper we presented a method that we call critical data augmentation. The method exploits available domain knowledge to do instance augmentation. Unlike previous methods, it learns to critically qualify the produced augmentations and learns to reject them from being used as a part of the learning data. We evaluated our method both on synthetic datasets, exploring various levels of defective domain knowledge as well as on real world datasets. The results clearly show that critical data augmentation brings performance improvements compared to baselines that either, uncritically, use all augmentations or do no augmentation at all.

References

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. November 2017.

Henry S Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. CVAE-GAN: Fine-Grained image generation through asymmetric training. March 2017.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. June 2014.

Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John W Fisher, III, and Lars Kai Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. October 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. December 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

Eyal Krupka and Naftali Tishby. Incorporating prior knowledge on features into learning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007*, pages 227–234, 2007. URL http://proceedings.mlr.press/v2/krupka07a.html.

Matt J Kusner, Yu Sun, Nicholas I Kolkin, and Kilian Q Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966, 2015.

Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation-learning an optimal data augmentation strategy. *IEEE Access*, 2017.

Yu-Feng Li and Zhi-Hua Zhou. Improving semi-supervised support vector machines through unlabeled instances selection. *arXiv preprint arXiv:1005.1545*, 2010.

Yu-Feng Li and Zhi-Hua Zhou. Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):175–188, 2015.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. November 2014.

Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. April 2017.

Amina Mollaysa, Pablo Strasser, and Alexandros Kalousis. Regularising non-linear models using feature side-information. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2508–2517, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/mollaysa17a.html.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems*, pages 2098–2106, 2015.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Leon Sixt, Benjamin Wild, and Tim Landgraf. RenderGAN: Generating realistic labeled data. November 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image translation using Cycle-Consistent adversarial networks. March 2017a.

Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks. November 2017b.