Semi-supervised Stochastic Multi-Domain Learning using Variational Inference

Yitong Li **Timothy Baldwin** **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne, Australia
yitongl4@student.unimelb.edu.au
{tbaldwin,tcohn}@unimelb.edu.au

Abstract

Supervised models of NLP rely on large collections of text which closely resemble the intended testing setting. Unfortunately matching text is often not available in sufficient quantity, and moreover, within any domain of text, data is often highly heterogenous. In this paper we propose a method to distill the important domain signal as part of a multi-domain learning system, using a latent variable model in which parts of a neural model are stochastically gated based on the inferred domain. We compare the use of discrete versus continuous latent variables, operating in a domain-supervised or a domain semi-supervised setting, where the domain is known only for a subset of training inputs. We show that our model leads to substantial performance improvements over competitive benchmark domain adaptation methods, including methods using adversarial learning.

1 Introduction

Text corpora are often collated from several different sources, such as news, literature, microblogs, and web crawls, raising the problem of learning NLP systems from heterogenous data, and how well such models transfer to testing settings. Learning from these corpora requires models which can generalise to different domains, a problem known as transfer learning or domain adaptation (Blitzer et al., 2007; Daumé III, 2007; Joshi et al., 2012; Kim et al., 2016). In most state-of-the-art frameworks, the model has full knowledge of the domain of instances in the training data, and the domain is treated as a discrete indicator variable. However, in reality, data is often messy, with domain labels not always available, or providing limited information about the style and genre of text. For example, web-crawled corpora are comprised of all manner of text, such as news, marketing, blogs, novels, and recipes, however the type of each document is typically not explicitly specified. Moreover, even corpora that are labelled with a specific domain might themselves be instances of a much more specific area, e.g., "news" articles will cover politics, sports, travel, opinion, etc. Modelling these types of data accurately requires knowledge of the specific domain of each instance, as well as the domain of each test instance, which is particularly problematic for test data from previously unseen domains.

A simple strategy for domain learning is to jointly learn over all the data with a single model, where the model is not conditioned on domain, and directly maximises $p(y|\mathbf{x})$, where $\mathbf{x}$ is the text input, and y the output (e.g. classification label). Improvements reported in multi-domain learning (Daumé III, 2007; Kim et al., 2016) have often focused on learning twin representations (*shared* and *private* representations) for each instance. The private representation is modelled by introducing a domain-specific channel conditional on the domain, and the shared one is learned through domain-general channels. To learn more robust domain-general and domain-specific channels, adversarial supervision can be applied in the form of either domain-conditional or domain-generative methods (Liu et al., 2016; Li et al., 2018a).

Inspired by these works, we develop a method for the setting where the domain is unobserved or partially observed, which we refer to as unsupervised and semi-supervised, respectively, with respect to domain. This has the added benefit of affording robustness where the test data is drawn from an unseen domain, through modelling each test instance as a mixture of domains. In this paper, we propose methods which use latent variables to characterise the domain, by modelling the discriminative learning problem $p(y|\mathbf{x}) = \int_z p(z|\mathbf{x})p(y|\mathbf{x}, z)$, where z encodes the domain, which must be marginalised

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1923–1934
Florence, Italy, July 28 - August 2, 2019.

out when the domain is unobserved. We propose a sequence of models of increasing complexity in the modelling of the treatment of z, ranging from a discrete mixture model, to a continuous vector-valued latent variable (analogous to a topic model; Blei et al. (2003)), modelled using Beta or Dirichlet distributions. We show how these models can be trained efficiently, using either direct gradient-based methods or variational inference (Kingma et al., 2014), for the respective model types. The variational method can be applied to domain and/or label semi-supervised settings, where not all components of the training data are fully observed.

We evaluate our approach using sentiment analysis over multi-domain product review data and 7 language identification benchmarks from different domains, showing that in out-of-domain evaluation, our methods substantially improve over benchmark methods, including adversarially-trained domain adaptation (Li et al., 2018a). We show that including additional domain unlabelled data gives a substantial boost to performance, resulting in transfer models that often outperform domain-trained models, to the best of our knowledge, setting a new state of the art for the dataset.

2 Stochastic Domain Adaptation

In this section, we describe our proposed approaches to Stochastic Domain Adaptation (SDA), which use latent variables to represent an implicit 'domain'. This is formulated as a joint model of output classification label, y and latent domain z, which are both conditional on $\mathbf{x}$,

$$p(y, z|\mathbf{x}) = p_\phi(\mathbf{z}|\mathbf{x})p_\theta(y|\mathbf{x}, \mathbf{z})\,.$$

The two components are the prior, $p_\phi(\mathbf{z}|\mathbf{x})$, and classifier likelihood, $p_\theta(y|\mathbf{x}, \mathbf{z})$, which are parameterised by ϕ and θ, respectively. We propose several different choices of prior, based on the nature of $\mathbf{z}$, that is, whether it is: (i) a discrete value ("DSDA", see Section 2.2); or (ii) a continuous vector, in which case we experiment with different distributions to model $p(\mathbf{z}|\mathbf{x})$ ("CSDA", see Section 2.3).

2.1 Stochastic Channel Gating

For all of our models the likelihood, $p_\theta(y|\mathbf{x}, \mathbf{z})$, is formulated as a multi-channel neural model, where $\mathbf{z}$ is used as a gate to select which channels should be used in representing the input. The model comprises k channels, with each channel computing an independent hidden representation,

$$\mathbf{h}_i = \text{CNN}_i(\mathbf{x}; \theta)|_{i=1}^k$$

using a convolutional neural network.[1] The value of z is then used to select the channel, by computing $\mathbf{h} = \sum_{i=1}^k z_k \mathbf{h}_i$, where we assume $\mathbf{z} \in \mathbb{R}^k$ is a continuous vector. For the discrete setting, we represent integer z by its 1-hot encoding $\mathbf{z}$, in which case $\mathbf{h} = \mathbf{h}_z$. The final step of the likelihood passes $\mathbf{h}$ through a MLP with a single hidden layer, followed by a softmax, which is used to predict class label y.

2.2 Discrete Domain Identifiers

We now turn to the central part of our method, the prior component. The simplest approach, DSDA (see Figure 1a), uses a discrete latent variable, i.e., $z \in [1, k]$ is an integer-valued random variable, and consequently the model can be considered as a form of mixture model. This prior predicts z given input $\mathbf{x}$, which is modelled using a neural network with a softmax output. Given z, the process of generating y is as described above in Section 2.1. The discrete model can be trained for the maximum likelihood estimate using the objective,

$$\log p(y|\mathbf{x}) = \log \sum_{z=1}^{k} p_\phi(z|\mathbf{x})p_\theta(y|\mathbf{x}, z), \quad (1)$$

which can be computed tractably,[2] and scales linearly in k.

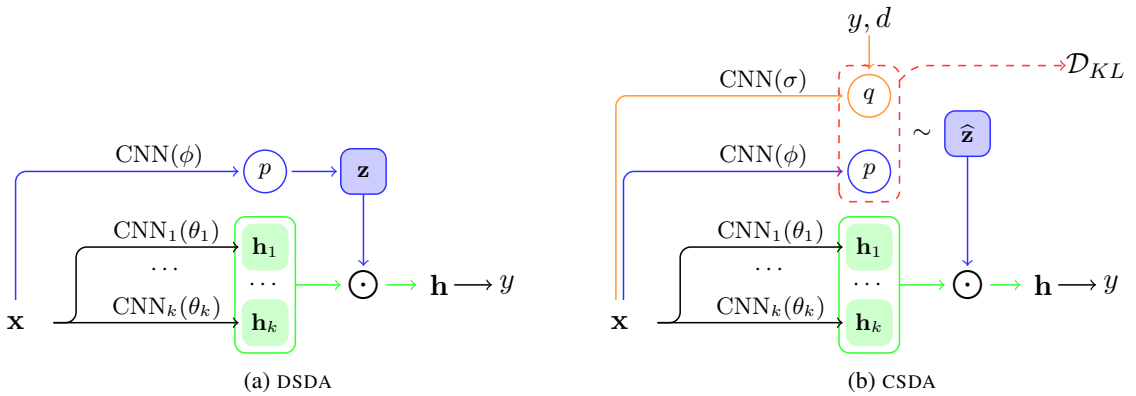
DSDA can be applied with supervised or semi-supervised domains, by maximising the likelihood $p(z = d|\mathbf{x})$ when the ground truth domain d is observed. We refer to this setting as "DSDA +sup." or "DSDA +semisup", respectively, noting that in this setting we assume the number of channels, k, is equal to the known inventory of domains, D.

2.3 Continuous Domain Identifiers

For the DSDA model to work well requires sufficiently large k, such that all the different types of data can be clearly separated into individual mixture components. When there is not a clear delineation between domains, the inferred domain posterior is likely to be uncertain, and the approach

[1] Our approach is general, and could be easily combined with other methods besides CNNs.

[2] This arises from the finite summation in (1), which requires each of the k components to be computed separately, and their results summed. This procedure permits standard gradient back-propagation.



(a) DSDA (b) CSDA

Figure 1: Model architectures for latent variable models, DSDA and CSDA, which differ in the treatment of the latent variable, which is discrete ($d \in [1, k]$), or a continuous vector ($\hat{\mathbf{z}} \in \mathbb{R}^k$). The lower green model components show k independent convolutional network components, and the blue and yellow component the prior, p, and the variational approximation, q, respectfully. The latent variable is used to gate the k hidden representations (shown as $\odot$), which are then used in a linear function to predict a classification label, y. During training CSDA draws samples ($\sim$) from q, while during inference, samples are drawn from p.

reduces to an ensemble technique. Thus, we introduce the second modelling approach as Continuous domain identifiers (CSDA), inspired by the way in which LDA models the documents as mixtures of several topics (Blei et al., 2003).

A more statistically efficient method would be to use binary functions as domain specifiers, i.e., $\mathbf{z} \in \{0, 1\}^k$, effectively allowing for exponentially many domain combinations (2^k). Each element of the domain z_i acts as a gate, or equivalently, attention, governing whether hidden state $\mathbf{h}_i$ is incorporated into the predictive model. In this way, individual components of the model can specialise to a very specific topic such as politics or sport, and yet domains are still able to combine both to produce specialised representations, such as the politics of sport. The use of a latent bit-vector renders inference intractable, due to the marginalisation over exponentially many states. For this reason, we instead make a continuous relaxation, such that $\mathbf{z} \in \mathbb{R}^k$ with each scalar z_i being drawn from a probability distribution parameterised as a function of the input $\mathbf{x}$. These functions can learn to relate aspects of $\mathbf{x}$ with certain domain indexes, e.g., the use of specific words like *baseball* and *innings* relate to a domain corresponding to "sport", thereby allowing the text domains to be learned automatically.

Several possible distributions can be used to model $\mathbf{z} \in \mathbb{R}^k$. Here we consider the following distributions:

Beta which bounds all elements to the range $[0, 1]$, such that $\mathbf{z}$ lies in a hyper-cube;

Dirichlet which also bounds all elements, as for Beta, however $\mathbf{z}$ are also constrained to lie in the probability simplex.

In both cases,[3] each dimension of $\mathbf{z}$ is controlled by different distribution parameters, themselves formulated as different non-linear functions of $\mathbf{x}$. We expect the Dirichlet model to perform the best, based on their widespread use in topic models, and their desirable property of generating a normalised vector, resembling common attention mechanisms (Bahdanau et al., 2015).

Depending on the choice of distribution, the prior is modelled as

$$p(\mathbf{z}|\mathbf{x}) = \text{Beta}\left(\boldsymbol{\alpha}^B, \boldsymbol{\beta}^B\right) \quad (2a)$$

$$\text{or} \quad p(\mathbf{z}|\mathbf{x}) = \text{Dirichlet}\left(\alpha_0 \boldsymbol{\alpha}^D\right), \quad (2b)$$

where the prior parameters are parameterised as neural networks of the input. For the Beta prior,

$$\boldsymbol{\alpha}^B = \text{elu}(\text{f}_{\alpha,B}(\mathbf{x})) + 1 \quad (3a)$$

$$\boldsymbol{\beta}^B = \text{elu}(\text{f}_{\beta,B}(\mathbf{x})) + 1, \quad (3b)$$

where $\text{elu}(\cdot) + 1$ is an element-wise activation function which returns a positive value (Clevert et al., 2016), and $\text{f}_\omega(\cdot)$ is a nonlinear function with parameters ω—here we use a CNN. The Dirichlet prior uses a different parameterisation,

$$\alpha_0 = \exp(\text{f}_{D,0}(\mathbf{x})) \quad (4a)$$

$$\boldsymbol{\alpha}^D = \text{sigmoid}(\text{f}_D(\mathbf{x})), \quad (4b)$$

[3] We also compared Gamma distributions, but they underperformed Beta and Dirichlet models.

where α_0 is a positive-valued overall concentration parameter, used to scale all components in (2b), thus capturing overall sparsity, while $\boldsymbol{\alpha}^D$ models the affinity to each channel.

2.4 Variational Inference

Using continuous latent variables, as described in Section 2.3, gives rise to intractable inference; for this reason we develop a variational inference method based on the variational auto-encoder (Kingma and Welling, 2014). Fitting the model involves maximising the evidence lower bound (ELBO),

$$\begin{aligned}\log p_{\phi,\theta}(y|\mathbf{x}) &= \log \int_{\mathbf{z}} p_\phi(\mathbf{z}|\mathbf{x}) p_\theta(y|\mathbf{z},\mathbf{x}) \\ &\geq \mathop{\mathbb{E}}_{q_\sigma}\left[\log p_\theta(y|\mathbf{z},\mathbf{x})\right] \\ &\quad - \lambda \mathcal{D}_{\mathrm{KL}}\big(q_\sigma(\mathbf{z}|\mathbf{x},y,d)||p_\phi(\mathbf{z}|\mathbf{x})\big)\,,\end{aligned} \tag{5}$$

where q_σ is the variational distribution, parameterised by σ, chosen to match the family of the prior (Beta or Dirichlet) and λ is a hyper-parameter controlling the weight of the KL term. The ELBO in (5) is maximised with respect to σ, ϕ and θ, using stochastic gradient ascent, where the expectation term is approximated using a single sample, $\hat{\mathbf{z}} \sim q_\sigma$, which is used to compute the likelihood directly. Although it is not normally possible to backpropagate gradients through a sample, which is required to learn the variational parameters σ, this problem is usually sidestepped using a reparameterisation trick (Kingma and Welling, 2014). However this method only works for a limited range of distributions, most notably the Gaussian distribution, and for this reason we use the implicit reparameterisation gradient method (Figurnov et al., 2018), which allows for inference with a variety of continuous distributions, including Beta and Dirichlet. We give more details of the implicit reparameterisation method in Appendix A.2.

The variational distribution q, is defined in an analagous way to the prior, p, see (2–4b), i.e., using a neural network parameterisation for the distribution parameters. The key difference is that q conditions not only on $\mathbf{x}$ but also on the target label y and domain d. This is done by embedding both y and d, which are concatenated with a CNN encoding of $\mathbf{x}$, and then transformed into the distribution parameters. Semi-supervised learning with respect to the domain can easily be facilitated by setting d to the domain identifier when it is observed, otherwise using a sentinel value $d = \texttt{UNK}$, for domain-unsupervised instances. The same trick is used for y, to allow for vanilla semi-supervised learning (with respect to target label). The use of y and d allows the inference network to learn to encode these two key variables into z, to encourage the latent variable, and thus model channels, to be informative of both the target label and the domain. This, in concert with the KL term in (5), ensures that the prior, p, must also learn to discriminate for domain and label, based solely on the input text, $\mathbf{x}$.

For inference at test time, we assume that only $\mathbf{x}$ is available as input, and accordingly the inference network cannot be used. Instead we generate a sample from the prior $\hat{\mathbf{z}} \sim p(\mathbf{z}|\mathbf{x})$, which is then used to compute the maximum likelihood label, $\hat{y} = \arg\max_y p(y|\mathbf{x},\hat{\mathbf{z}})$. We also experimented with Monte Carlo methods for test inference, in order to reduce sampling variance, using: (a) prior mean $\bar{\mathbf{z}} = \mu$; (b) Monte Carlo averaging $\bar{y} = \frac{1}{m}\sum_i p(y|\mathbf{x},\hat{\mathbf{z}}_i)$ using $m = 100$ samples from the prior; and (c) importance sampling (Glynn and Iglehart, 1989) to estimate $p(y|\mathbf{x})$ based on sampling from the inference network, q.[4] None of the Monte Carlo methods showed a significant difference in predictive performance versus the single sample technique, although they did show a very tiny reduction in variance over 10 runs. This is despite their being orders of magnitude slower, and therefore we use a single sample for test inference hereafter.

3 Experiments

3.1 Multi-domain Sentiment Analysis

To evaluate the proposed models, we first experiment with a multi-domain sentiment analysis dataset, focusing on out-of-domain evaluation where the test domain is unknown.

We derive our dataset from Multi-Domain Sentiment Dataset v2.0 (Blitzer et al., 2007).[5] The task is to predict a binary sentiment label, i.e., positive vs. negative. The unprocessed dataset has more than 20 domains. For our purposes, we filter out domains with fewer than 1k labelled instances

[4] Importance sampling estimates $p(y|\mathbf{x}) = \mathbb{E}_q[p(y,\mathbf{z}|\mathbf{x})/q(\mathbf{z}|\mathbf{x},y,d)]$ for each setting of y using $m = 100$ samples from q, and then finds the maximising y. This is tractable in our settings as y is a discrete variable, e.g., a binary sentiment, or multiclass language label.

[5] From https://www.cs.jhu.edu/~mdredze/datasets/sentiment/.

Domain	$\mathcal{F}(\mathbf{x}, y, d)$	$\mathcal{Y}(\mathbf{x}, y, ?)$
apparel	1,000	1,000
baby	950	950
camera & photo	1000	999
health & personal care	1,000	1,000
magazines	985	985
music	1,000	1,000
sports & outdoors	1,000	1,000
toys & games	1,000	1,000
video	1,000	1,000

Table 1: Numbers of instances (reviews) for each training domain in our dataset, under the two categories $\mathcal{F}$ (domain and label known) and $\mathcal{Y}$ (label known; domain unknown), in which "?" represents the "UNK" token, meaning the given attribute is unobserved.

or fewer than 2k unlabelled instances, resulting in 13 domains in total.

To simulate the semi-supervised domain situation, we remove the domain attributions for one half of the labelled data, denoting them as domain-unlabelled data $\mathcal{Y}(\mathbf{x}, y, ?)$. The other half are sentiment- and domain-labelled data $\mathcal{F}(\mathbf{x}, y, d)$. We present a breakdown of the dataset in Table 1.[6]

For evaluation, we hold out four domains—namely books ("B"), dvds ("D"), electronics ("E"), and kitchen & housewares ("K")—for comparability with previous work (Blitzer et al., 2007). Each domain has 1k test instances, and we split this data into *dev* and *test* with ratio 4:6. The *dev* dataset is used for hyper-parameter tuning and early stopping,[7] and we report accuracy results on *test*.

3.1.1 Baselines and Comparisons

For comparison, we use 3 baselines. The first is a single channel CNN ("S-CNN"), which jointly over all data instances in a single model, without domain-specific parameters. The second baseline is a multi channel CNN ("M-CNN"), which expands the capacity of the S-CNN model (606k parameters) to match CSDA and DSDA (roughly 7.5m-8.3m parameters). Our third baseline is a multi-domain learning approach using adversarial learning for domain generation ("GEN"), the best-performing model of Li et al. (2018a) and state-of-the-art for unsupervised multi-domain adaptation over a comparable dataset.[8] We report results for their best performing GEN +d+g model.

3.1.2 Training Strategy

For the hyper-parameter setups, we provide the details in Appendix A.1. In terms of training, we simulate two scenarios using two experimental configurations, as discussed above: (a) domain supervision; and (2) domain semi-supervision. For domain supervised training, only $\mathcal{F}$ is used, which covers only 9 of the domains, and the test domain data is entirely unseen. For domain semi-supervised training, we use combinations of $\mathcal{F}$ and $\mathcal{Y}$, noting that both sub-corpora do not include data from the target domains, and none of which is explicitly labelled with sentiment, y, and domain, d. These simulate the setting where we have heterogenous data which includes a lot of relevant data, however its metadata is inconsistent, and thus cannot be easily modelled.

For λ in (5), according to the derivation of the ELBO it should be the case that $\lambda = 1$, however other settings are often justified in practice (Alemi et al., 2018). Accordingly, we tried both annealing and fixed schedules, but found no consistent differences in end performance. We performed a grid search for the fixed value, $\lambda = 10^a$, $a \in \{-3, -2, -1, 0, 1\}$, and selected $\lambda = 10^{-1}$, based on development performance. We provide further analysis in the form of a sensitivity plot in Section 3.2. The latent domain size k for DSDA is set to the true number of training domains $k = D = 9$. Note that, even for DSDA, we could use $k \neq D$, which we explore in the $\mathcal{F} + \mathcal{Y}$ supervision setting in Section 3.1.3. For CSDA we present the main results with $k = 13$, set to match the total number of domains in training and testing.

3.1.3 Results

Table 2 reports the performance of different models under two training configurations: (1) with $\mathcal{F} + \mathcal{Y}$ (domain semi-supervised learning); and (2) with $\mathcal{F}$ only (domain supervised learning). In each case, we report the standard deviation based on 10 runs with different random seeds.

Overall, domain B and D are more difficult than E and K, consistent with previous work. Comparing the two configurations, we see that when we use domain semi-supervised training (with the addition of $\mathcal{Y}$), all models perform bet-

[6] The dataset, along with the source code, can be found at https://github.com/lrank/Code_VariationalInference-Multidomain

[7] This confers light supervision in the target domain. However we would expect similar results were we to use disjoint held out domains for development wrt testing.

[8] The dataset used in Li et al. (2018a) differs slightly in that it is also based off Multi-Domain Sentiment Dataset v2.0, but uses slightly more training domains and a slightly different composition of training data. We retrain the model of the authors over our dataset, using their implementation.

Data			B	D	E	K	Average
$\mathcal{F}+\mathcal{Y}$	S-CNN		78.9 ± 1.3	80.9 ± 1.5	82.4 ± 0.8	84.1 ± 1.8	81.6 ± 0.9
	M-CNN		79.0 ± 1.5	82.5 ± 1.3	84.1 ± 0.8	85.9 ± 0.8	82.9 ± 0.9
	GEN		78.4 ± 0.9	81.2 ± 1.0	83.9 ± 1.7	**87.5** ± 1.2	82.8 ± 1.1
	DSDA		76.8 ± 1.4	79.6 ± 1.7	83.1 ± 1.5	85.8 ± 2.0	81.3 ± 1.0
		+ semi-sup.	77.1 ± 1.6	79.9 ± 1.0	83.1 ± 1.7	85.4 ± 1.3	81.4 ± 0.6
	CSDA	w. Beta	78.4 ± 0.8	**84.4** ± 0.7	82.9 ± 1.1	87.2 ± 1.3	83.2 ± 0.9
		w. Dirichlet	**80.0** ± 1.4	84.3 ± 1.4	**86.2** ± 1.5	87.0 ± 0.3	**84.4** ± 0.9
$\mathcal{F}$ only	S-CNN		76.0 ± 1.8	77.0 ± 1.0	81.5 ± 1.3	82.8 ± 1.6	79.3 ± 0.7
	M-CNN		76.7 ± 1.8	79.2 ± 0.4	82.0 ± 1.2	83.1 ± 1.8	79.8 ± 1.3
	GEN		76.7 ± 2.0	79.1 ± 1.3	82.1 ± 1.6	84.0 ± 1.1	80.5 ± 0.7
	DSDA		74.3 ± 1.4	75.8 ± 2.2	80.5 ± 1.3	82.8 ± 1.4	78.4 ± 0.9
		+ unsup.	74.1 ± 2.0	75.6 ± 2.3	80.8 ± 1.3	83.0 ± 1.7	78.4 ± 0.6
	CSDA	w. Beta	**78.0** ± 1.9	80.5 ± 1.1	83.7 ± 1.3	85.7 ± 1.3	82.0 ± 1.1
		w. Dirichlet	77.9 ± 1.6	**80.6** ± 0.9	**84.4** ± 1.1	**86.5** ± 0.9	**82.3** ± 0.6
	IN DOMAIN ♣		80.4	82.4	84.4	87.7	83.7

Table 2: Accuracy [%] and standard deviation of different models under two data configurations: (1) using both $\mathcal{F}$ and $\mathcal{Y}$ (domain semi-supervised learning); and (2) using $\mathcal{F}$ only (domain supervised learning). In each case, we evaluate over the four held-out test domains (B, D, E and K), and also report the accuracy. Best results are indicated in **bold** in each configuration. Key: ♣ from Blitzer et al. (2007).

ter, demonstrating the utility of domain semi-supervised learning when annotated data is limited.

Comparing our discrete and continuous approaches (DSDA and DSDA, resp.), we see that CSDA consistently performs the best, outperforming the baselines by a substantial margin. In contrast DSDA is disappointing, underperforming the baselines, and moreover, shows no change in performance between domain supervision versus the semi-supervised or unsupervised settings. Among the CSDA based methods, all the distributions perform well, but the Dirichlet distribution performs the best overall, which we attribute to better modelling of the sparsity of domains, thus reducing the influence of uncertain and mixed domains. The best results are for domain semi-supervised learning ($\mathcal{F}+\mathcal{Y}$), which brings an increase in accuracy of about 2% over domain supervised learning ($\mathcal{F}$) consistently across the different types of model.

3.2 Analysis and Discussion

To better understand what the model learns, we focus on the CSDA model, using the Dirichlet distribution.

First, we consider the model capacity, in terms of the latent domain size, k. Figure 2 shows the impact of varying k. Note that the true number of domains is $D = 13$, comprising 9 training and 4 test domains. Setting k to roughly this value appears to be justified, in that the mean accuracy

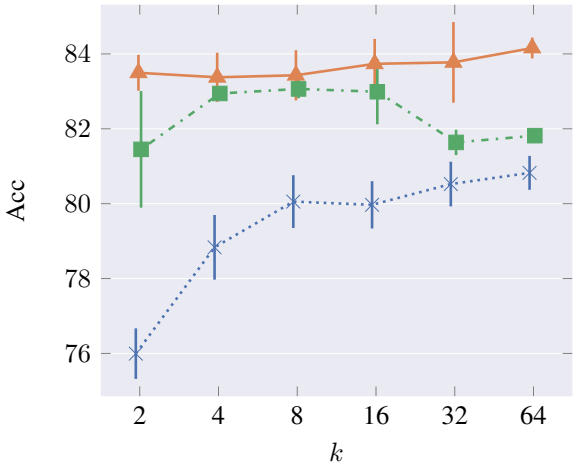


Figure 2: Performance with standard error (|) as latent domain size k is increased in $\log 2$ space with DSDA (×) and with three CSDA methods using Beta (■) and Dirichlet (▲) averaged accuracy, over $\mathcal{F}+\mathcal{Y}$.

increases with k, and plateaus around $k = 16$. Interestingly, when $k \geq 32$, the performance of CSDA with Beta drops, while performance for Dirichlet remains high—indeed Dirichlet is consistently superior even at the extreme value of $k = 2$, although it does show improvement as k increases. Also observe that DSDA requires a large latent state inventory, supporting our argument for the efficiency of continuous cf. discrete latent variables.

Next, we consider the impact of using different combinations of $\mathcal{F}$ and $\mathcal{Y}$. Table 3 shows the performance of difference configurations. Overall, $\mathcal{F}+\mathcal{Y}$ gives excellent performance. Interestingly,

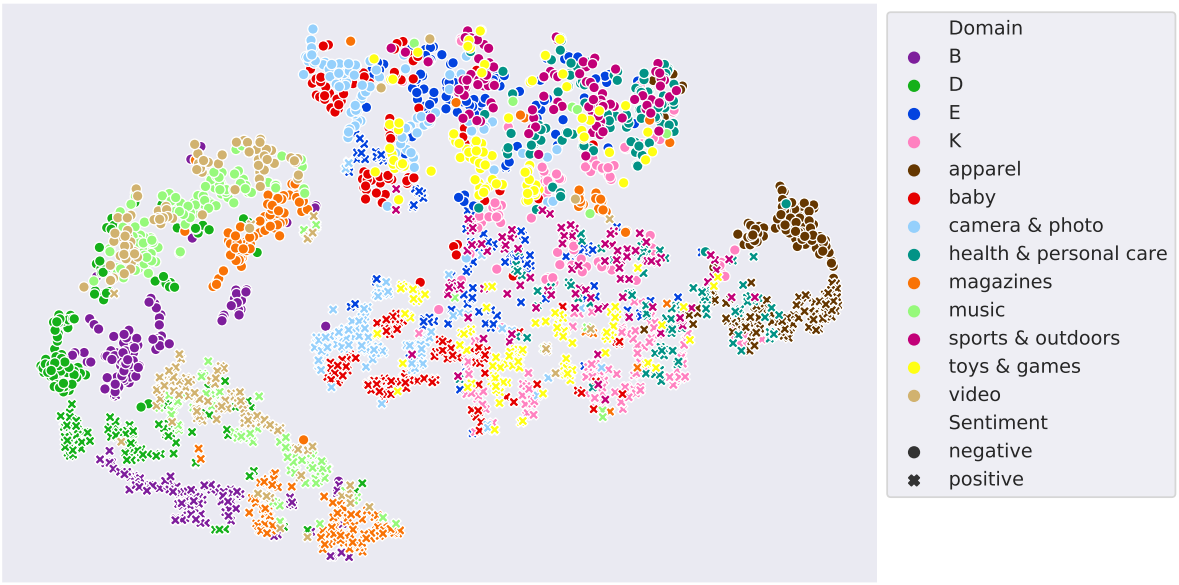


Figure 3: t-SNE of hidden representations in CSDA over all 13 domains, comprising 4 held-out testing domains (B, D, E and K), and the remaindering 9 domains are used only for training. Each point is a document, and the symbol indicates its gold sentiment label, using a filled circle for negative instances and cross for positive.

CSDA	B	D	E	K	Average
$\mathcal{F}$	77.9	80.6	84.4	86.5	82.3
$\mathcal{F}+\mathcal{Y}$	80.0	84.3	86.2	87.0	84.4
$\mathcal{Y}$	77.6	81.5	83.7	85.2	82.0

Table 3: Accuracy [%] of CSDA w. Dirichlet trained with different configurations of $\mathcal{F}$ and $\mathcal{Y}$.

$\mathcal{Y}$ on its own is only a little worse than only $\mathcal{F}$, showing that target labels y are more important for learning than the domain d. The $\mathcal{Y}$ configuration fully domain unsupervised training still results in decent performance, boding well for application to very messy and heterogenous datasets with no domain metadata.

Finally, we consider what is being learned by the model, in terms of how it learns to use the k dimensional latent variables for different types of data. We visualise the learned representations, showing points for each domain plotted in a 2d t-SNE plot (Maaten and Hinton, 2008) in Figure 3. Notice that each domain is split into two clusters, representing positive (×) and negative (•) instances within that domain. Among the test domains, B (books) and D (dvds) are clustered close together but are still clearly separated, which is encouraging given the close relation between these two media. The other two, E (electronics) and K (kitchen & housewares) are mixed together and intermingled with other domains. Overall across all domains, the APPAREL cluster is quite distinct,

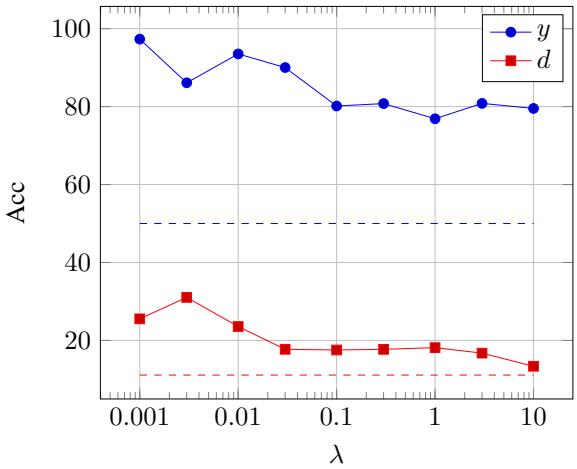


Figure 4: Diagostic classifier accuracy [%] over $\mathbf{z}$ to predict the sentiment label y and domain label d, with respect to different λ, shown on a log scale. Dashed horizontal lines show chance accuracy for both outputs.

while VIDEO and MUSIC are highly associated with D, and part of the cluster for MAGAZINES is close to B; all of these make sense intuitively, given similarities between the respective products. E is related to CAMERA and GAMES, while K is most closely connected to HEALTH and SPORTS.

To obtain a better understanding of what is being encoded in the latent variable, and how this is effected by the setting of λ, we learn simple diagnostic classifiers to predict sentiment label y and domain label d, given only $\mathbf{z}$ as input. To do so, we first train our model over the training set, and

Data			EUROGOV	TCL	WIKIPEDIA	EMEA	EUROPARL	TBE	TSC	Average
$\mathcal{Y}$	S-CNN		98.8	92.3	85.9	98.5	92.3	79.3	91.7	91.3
	M-CNN		**98.9**	**93.6**	86.2	99.2	96.0	88.3	91.7	93.4
	DSDA		98.3	91.9	86.3	97.8	95.2	86.0	79.0	90.6
	CSDA	w. Beta	98.7	93.0	**89.0**	**99.3**	96.8	93.1	**95.2**	**95.0**
		w. Dirichlet	**98.9**	93.0	**89.0**	99.2	**96.7**	**93.2**	94.5	94.9
$\mathcal{F}$	DSDA		98.0	91.8	85.7	97.7	95.3	85.4	78.1	90.3
	CSDA	w. Beta	**99.3**	**93.7**	89.1	99.2	**96.9**	**93.6**	93.9	95.1
		w. Dirichlet	99.0	**93.7**	**89.3**	**99.3**	**96.9**	93.3	**96.1**	**95.4**
	GEN		99.9	93.1	88.7	92.5	97.1	91.2	96.1	94.1
	LANGID.PY		98.7	90.4	91.3	93.4	97.4	94.1	92.7	94.0

Table 4: Accuracy [%] over 7 LangID benchmarks, as well as the averaged score, for different models under two data configurations: (1) using domain unsupervised learning ($\mathcal{Y}$); and (2) using domain supervised learning ($\mathcal{F}$). The best results are indicated in **bold** in each configuration. Note that the training data for GEN and LANGID.PY is slightly different from that used in the original papers.

record samples of $\mathbf{z}$ from the inference network. We then partition the training set, using 70% to learn linear logistic regression classifiers to predict y and d, and use the remaining 30% for evaluation. Figure 4 shows the prediction accuracy, based on averaging over three runs, each with different $\mathbf{z}$ samples. Clearly very small $\lambda \leq 10^{-2}$, leads to almost perfect sentiment label accuracy which is evidence of overfitting by using the latent variable to encode the response variable. For $\lambda \geq 10^{-1}$ the sentiment accuracy is still above chance, as expected, but is more stable. For the domain label d, the predictive accuracy is also above chance, albeit to a lesser extent, and shows a similar downward trend. At the setting $\lambda = 0.1$, used in the earlier experiments, this shows that the latent variable encodes captures substantial sentiment, and some domain knowledge, as observed in Figure 3.

In terms of the time required for training, a single epoch of training took about 25min for the CSDA method, using the default settings, and a similar time for DSDA and M-CNN. The runtime increases sub-linearly with increasing latent size k.

3.3 Language Identification

To further demonstrate our approaches, we then evaluate our models with the second task, language identification (LangID: Jauhiainen et al. (2018)).

For data processing, we use 5 training sets from 5 different domains with 97 language, following the setup of Lui and Baldwin (2011). We evaluate accuracy over 7 holdout benchmarks: EUROGOV, TCL, WIKIPEDIA from Baldwin and Lui (2010), EMEA (Tiedemann, 2009), EUROPARL (Koehn, 2005), TBE (Tromp and Pechenizkiy, 2011) and TSC (Carter et al., 2013). Differently from sentiment tasks, here, we evaluate our methods using the full dataset, but with two configurations: (1) domain unsupervised, where all instance have only labels but no domain (denoted $\mathcal{Y}$); and (2) domain supervised learning, where all instances have labels and domain ($\mathcal{F}$).

3.3.1 Results

Table 4 shows the performance of different models over 7 holdout benchmarks and the averaged scores. We also report the results of GEN, the best model from Li et al. (2018a), and one state-of-the-art off-the-shelf LangID tool: LANGID.PY (Lui and Baldwin, 2012). Note that, both S-CNN and M-CNN are domain unsupervised methods. In terms of results, overall, both of our CSDA models consistently outperform all other baseline models. Comparing the different CSDA variants, Beta vs. Dirichlet, both perform closely across the LangID tasks. Furthermore, CSDA out-performs the state-of-the-art in terms of average scores. Interestingly the two training configurations show that domain knowledge $\mathcal{F}$ provides a small performance boost for CSDA, but not does help for DSDA. Above all, the LangID results confirm the effectiveness of our proposed approaches.

4 Related Work

Domain adaptation ("DA") typically involves one or more training domains and a single target domain. Among DA approaches, single-domain adaptation is the most common scenario, where a model is trained over one domain and then transferred to a single target domain using prior

knowledge of the target domain (Blitzer et al., 2007; Glorot et al., 2011). Adversarial learning methods have been proposed for learning robust domain-independent representations, which can capture domain knowledge through semi-supervised learning (Ganin et al., 2016).

Multi-domain adaptation uses training data from more than one training domain. Approaches include feature augmentation methods (Daumé III, 2007), and analagous neural models (Joshi et al., 2012; Kim et al., 2016), as well as attention-based and hierarchical methods (Li et al., 2018b). These works assume the 'oracle' source domain is known when transferring, however we do not require an oracle in this paper. Adversarial training methods have been employed to learn robust domain-generalised representations (Liu et al., 2016). Li et al. (2018a) considered the case of the model having no access to the target domain, and using adversarial learning to generate domain-generation representations by cross-comparison between source domains.

The other important component of this work is Variational Inference ("VI"), a method from machine learning that approximates probability densities through optimisation (Blei et al., 2017; Kucukelbir et al., 2017). The idea of a variational auto-encoder has been applied to language generation (Bowman et al., 2016; Kim et al., 2018; Miao et al., 2017; Zhou and Neubig, 2017; Zhang et al., 2016) and machine translation (Shah and Barber, 2018; Eikema and Aziz, 2018), but not in the context of semi-supervised domain adaptation.

5 Conclusion

In this paper, we have proposed two models—DSDA and CSDA—for multi-domain learning, which use a graphical model with a latent variable to represent the domain. We propose models with a discrete latent variable, and a continuous vector-valued latent variable, which we model with Beta or Dirichlet priors. For training, we adopt a variational inference technique based on the variational autoencoder. In empirical evaluation over a multi-domain sentiment dataset and seven language identification benchmarks, our models outperform strong baselines, across varying data conditions, including a setting where no target domain data is provided. Our proposed models have broad utility across NLP applications on heterogenous corpora.

Acknowledgements

This work was supported by an Amazon Research Award. We thank the anonymous reviewers for their helpful feedback and suggestions.

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 229–237.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, 47(1):195–215.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the International Conference on Learning Representations*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Bryan Eikema and Wilker Aziz. 2018. Auto-encoding variational neural machine translation. *arXiv preprint arXiv:1807.10564*.

Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. 2018. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems 31*, pages 439–450.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:59:1–59:35.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*, pages 513–520.

Peter W Glynn and Donald L Iglehart. 1989. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392.

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2018. Automatic language identification in texts: A survey. *CoRR*, abs/1804.08186.

Mahesh Joshi, Mark Dredze, William W. Cohen, and Carolyn Penstein Rosé. 2012. Multi-domain learning: When do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1302–1312.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2678–2687.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 387–396.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3581–3589.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit 2005*, pages 79–86.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. 2017. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(1):430–474.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018a. What's in a domain? learning domain-robust text representations using adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 474–479.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018b. Hierarchical attention transfer network for cross-domain sentiment classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep multi-task learning with shared memory for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 118–127.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Fifth International Joint Conference on Natural Language Processing*, pages 553–561.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of ACL 2012 System Demonstrations*, pages 25–30.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2410–2419.

Harshil Shah and David Barber. 2018. Generative neural machine translation. In *Advances in Neural Information Processing Systems*, pages 1346–1355.

Jörg Tiedemann. 2009. News from OPUS – a collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248.

Erik Tromp and Mykola Pechenizkiy. 2011. Graph-based n-gram language identification on short texts.

In *Proceedings of the 20th Machine Learning Conference of Belgium and The Netherlands*, pages 27–34.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320.

A Appendices

A.1 Base Model Architecture

For the sentiment task, all the hidden representations are learned by convolutional neural networks (CNN), following Kim (2014). All documents are lower-cased and truncated to maximum 256 tokens, and then each word is mapped into a 300 dimensional vector representation using randomly-initialised word embeddings. In each CNN channel, filter windows are set to $\{3, 4, 5\}$, with 128 filters for each. Then, ReLU and pooling are applied after the filtering, generating 384-d ($128 * 3$) hidden representations. Dropout is applied to the hidden $\mathbf{h}$, at a rate of 0.5. For simplicity, we use the same CNN architecture to encode the functions f used in the prior q and in the inference networks p, in each case with different parameters. Specifically, in prior q, the embedding sizes of domain and label are set to 16 and 4, respectively. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ share the same CNN but with different output projections. After gating using $\mathbf{z}$, the final hidden goes through a one-hidden MLPwith hidden size 300. We use the Adam optimiser (Kingma and Ba, 2015) throughout, with the learning rate set to 10^{-4} and a batch size of 32, optimising the loss functions (1) or (5), for DSDA and CSDA, respectively.

For the language identification task, all documents are tokenized as a byte sequence, truncated or padded to a length of 1k bytes. We use the same CNN architecture and hyper-parameter configurations as for the sentiment task.

A.2 Implicit Reparameterisation Gradient

In this section, we outline the implicit reparameterisation gradient method of Figurnov et al. (2018). First, we review some background on variational inference. We start by defining a differentiable and invertible standardization function as

$$S_\sigma(\mathbf{z}) = \epsilon \sim q(\epsilon)\,, \tag{6a}$$

which describes a mapping between points drawn from a specific distribution function and a standard distribution, q. For example, for a Gaussian distribution $z \sim \mathcal{N}(\mu, \psi)$, we can define $S_{\mu,\psi}(z) = (z - \mu)/\psi \sim \mathcal{N}(0, 1)$ to map to the standard Normal. We aim to compute the gradient of the expectation of a objective function $f(\mathbf{z})$,

$$\nabla_\sigma \mathop{\mathbb{E}}_{q_\sigma(\mathbf{z})}[f(\mathbf{z})] = \mathop{\mathbb{E}}_{q(\epsilon)}[\nabla_\sigma f(S^{-1}(\epsilon))]\,, \tag{6b}$$

where in ELBO (5) in our case, $f(\mathbf{z}) = p_\theta(y|\mathbf{z}, \mathbf{x})$ is the likelihood function.

The implicit reparameterisation gradient technique is a way of computing the reparameterisation without the need for inversion of the standardization function. This works by applying $\nabla_\sigma S^{-1}(\epsilon) = \nabla_\sigma \mathbf{z}$,

$$\nabla_\sigma \mathop{\mathbb{E}}_{q_\sigma(\mathbf{z})}[f(\mathbf{z})] = \mathop{\mathbb{E}}_{q_\sigma(\mathbf{z})}[\nabla_\mathbf{z} f(\mathbf{z}) \nabla_\sigma \mathbf{z}]\,. \tag{6c}$$

However, we still need to calculate $\nabla_\sigma \mathbf{z}$. The key insight here is that we can compute $\nabla_\sigma \mathbf{z}$ by *implicit differentiation*. We apply the total gradient $\nabla^{\text{TD}}_\sigma$ over (6a),

$$\nabla^{\text{TD}}_\sigma S_\sigma(\mathbf{z}) = \nabla^{\text{TD}}_\sigma \epsilon\,. \tag{6d}$$

From the definition of a standardization function, the noise ϵ is independent of σ, and we apply the multi-variable chain rule over left side of (6d),

$$\frac{\partial S_\sigma(\mathbf{z})}{\partial \mathbf{z}} \nabla_\sigma \mathbf{z} + \frac{\partial S_\sigma(\mathbf{z})}{\partial \sigma} = \mathbf{0}\,. \tag{6e}$$

Therefore, the key of the implicit gradient calculation in this process can be summarised as

$$\nabla_\sigma \mathbf{z} = -(\nabla_\mathbf{z} S_\sigma(\mathbf{z}))^{-1} \nabla_\sigma S_\sigma(\mathbf{z})\,. \tag{6f}$$

This expression allows for computation of (6c), which can be applied to a range of distribution families. We refer the reader to Figurnov et al. (2018) for further details.