SYDE 671 Final Project Report: Unsupervised Visual Domain Adaptation: A Deep Max-Margin Gaussian Process Approach

JunYong Tong, Nick Torenvliet
University of Waterloo
December 12, 2019

Abstract

In this project we implement and adapt the Unsupervised Domain Adaptation(UDA) approach described in [6].

We begin with a brief description of the problem's context and introduce the various pieces of mathematical kit required to support the solution.

We go on to provide the basis for an adaptation of the method proposed in [6] by integrating the natural gradient method as given in [5].

We implement the approach in [6], as well as the natural gradient adaptation and then compare training outcomes of the two approaches across a grid of hyper-parameter selections. An ensuing discussion of results identifies the improvement our adaptation provides.

Finally we discuss potential applications for Domain Adaptation.

1. Introduction

In general deep learning requires well labelled training and test data that are independently and identically distributed (i.i.d). It is often difficult to meet the requirement for large labelled datasets, as the process of aquiring them can be time-consuming, costly, and laborious.

UDA is the task of training a model on labelled data from a source distribution so that it performs well on unlabelled data from a target distribution with common labels. Successful UDA application may provide good classification results while simultaneously reducing the burdensome requirement for labelled training data.

A common example might be the training of a text sentiment analysis system to classify email from clients to lawyers as either satisfied or unsatisfied. Utilizing UDA you could attempt to perform training using a labeled set of email from clients to financial advisors, and an unlabelled set of emails from clients to lawyers.

2. Related Work

The original account of domain adaptation is provided by Ben-David in [2], in which a theoretical upper-bound for error was provided.

There are three main approaches to tackling the domain adaptation problem.

The marginal matching strategy focuses on transferring deep neural network representations from a labelled source dataset to an unlabelled target dataset by matching the distributions of features between different domains, to extract domain-invariant features [11].

Generative modeling methods, [10] [4] [3], leverage an adversarial training strategy which allows the learning of feature representations to be discriminative for the labelled source domain data and indistinguishable between source and target domains.

A third approach utilizes a shared latent space to learn an embedding that are used for classification in the target domain. For example, [7] proposed unsupervised image-to-image translation framework based on the [8].

Kim et. al. in [6] provide the latest in a genre of literature related to the problem of Unsupervised Domain Adaptation using latent spaces. Their adaptation, the subject of this paper, utilizes Gaussian processes and variational inference to further refine the latent space based UDA approach. From this point on we use UDA to denote the method present in [6].

3. Unsupervised Domain Adaptation

To formalize the notion of latent space based UDA, consider a joint space of inputs and class labels, $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y} = \{1, \ldots, K\}$.

Consider a source **S** and target **T** domain on this space, with unknown distributions $p_S(\mathbf{x}, y)$ and $p_T(\mathbf{x}, y)$, respectively. Again consider a dataset consisting of labelled source-domain training examples $\mathcal{D}_S = \{\mathbf{x}_i^S, y_i\}_{i=1}^{N_S}$, and unlabelled target domain training examples $\mathcal{D}_T = \{\mathbf{x}_I^T\}_{i=1}^{N_T}$.

Assume a shared set of class labels between the source and target domains.

The goal of UDA is to assign correct class labels to the unlabelled data in $\mathbf{T}$. This assignment occurs via the support of a shared latent space $\mathcal{Z}$ with properties and structure imposed on it by the mathematical apparatus of UDA.

The structure of $\mathcal{Z}$ allows UDA to learn an embedding function $G : \mathcal{X} \to \mathcal{Z}$ and a classifier $h : \mathcal{Z} \to \mathcal{Y}$. $G(\cdot)$ and $h(\cdot)$ encode the information provided by the data in $\mathbf{S}$ to minimize error of the classifier h on the target domain $\mathbf{T}$.

3.1. Gaussian Process Classifier

We model K underlying latent functions $\mathbf{f} = \{f_i\}_{i=1}^K$ on shared space, $\mathcal{Z}$, as K-class Gaussian Process (GP) classifier,

$$p(\mathbf{f}) = \prod_{i=1}^{K} P(f_i),\ f_i \sim \mathcal{GP}\left(0, \kappa_j(\cdot,\cdot)\right) \tag{1}$$

where $\kappa_j(\cdot,\cdot)$ is the covariance function of the GP. Using the likelihood model,

$$P(y = i|\mathbf{f}(\mathbf{z})) = \frac{e^{f_i(\mathbf{z})}}{\sum_{j=1}^{K} e^{f_j(\mathbf{z})}},\ i = 1, \ldots, K \tag{2}$$

So that $\mathcal{D}_S$ induces a posterior distribution on $\mathbf{f}$,

$$p(\mathbf{f}|\mathcal{D}_S) \propto p(\mathbf{f}) \cdot \prod_{i=1}^{N_s} P(y_i^S|\mathbf{f}(\mathbf{z}_i^S)) \tag{3}$$

3.2. Deep Kernel Trick

The computation of the κ_j's in (1) is tractable only if considering a limited set of function classes for κ_j. To extend the capacity of the approach, the deep kernel trick is leveraged and the κ_j are modelled by a nonlinear feature map $\phi : \mathcal{Z} \to \mathbb{R}^d$ defined as an inner product in the feature space.

Assuming $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then the $\mathcal{H}$ prior (3) becomes,

$$p(\mathbf{W}|\mathcal{D}_S) \propto \prod_{i=1}^{K} \mathcal{N}(\mathbf{w}_i; \mathbf{0}, \mathbf{I}) \cdot \prod_{i=1}^{N_s} P(y_i^S|\mathbf{W}\boldsymbol{\phi}(\mathbf{z}_i^S)) \tag{4}$$

Note that $p(\mathbf{W}|\mathcal{D}_S)$ is still intractable.

3.3. Variational Inference

Due to the intractability of (3) we estimate its value using Variational Inference. Variational Inference is a family of Bayesian techniques for approximating intractable (posterior) distributions that often arise in Bayesian inference.

Given an intractable posterior distribution $p(\mathbf{z}|\mathbf{x})$, where $\mathbf{z}$ is a latent variable and $\mathbf{x}$ is some data. We approximate $p(\mathbf{z}|\mathbf{x})$ with a variational distribution $q(\mathbf{z})$. The variational distribution is chosen to be simpler (tractable and factorizable) than the posterior.

We choose $q(\mathbf{z})$ to be,

$$q(\mathbf{z}) = \prod_{i=1}^{K} \mathcal{N}(\mathbf{w}_i; \mathbf{m}_i, \mathbf{S}_i)$$

To construct an approximation we employ the KL-divergence,

$$\begin{aligned} D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q(\mathbf{z})}\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right] \\ &= \mathbb{E}_{q(\mathbf{z})} \log q(\mathbf{z}) - \log p(\mathbf{z}|\mathbf{x}) \\ &= \mathbb{E}_{q(\mathbf{z})} \log q(\mathbf{z}) - \log \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})} \\ &= \mathbb{E}_{q(\mathbf{z})} \log q(\mathbf{z}) - \log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \\ &\quad + \log p(\mathbf{x}) \\ &= \mathbb{E}_{q(\mathbf{z})} \log \frac{q(\mathbf{z})}{p(\mathbf{z})} - \log p(\mathbf{x}|\mathbf{z}) \\ &\quad + \log p(\mathbf{x}) \end{aligned}$$

Then because we are optimizing over $q(\mathbf{z})$ and $\log p(\mathbf{x})$ is a constant with respect $q(\mathbf{z})$ we get,

$$D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) \propto D_{KL}(q(\mathbf{z})||p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z})} \log p(\mathbf{x}|\mathbf{z})$$

We replace $\mathbf{z}$ with $\mathbf{W}$ and $\mathbf{x}$ with $\mathcal{D}_S$ so that minimizing $D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$ is equivalent to finding,

$$\arg\min_{q} D_{KL}(q(\mathbf{W})||p(\mathbf{W})) - \sum_{i=1}^{N_S} \mathbb{E}_q \log p(y_i|\mathbf{W}) \tag{5}$$

This is also known as the evidence lower bound (ELBO).

4. Improving UDA via the Natural Gradient

Notice that (5) is an optimization in the distribution space with update steps taken in the parameter space. There could be two different parametrizations of q that yield the same results. Hence performing gradient descent with the parameters directly may lead wasteful update step and slow convergence.

To improve UDA by avoiding possible inefficiencies, we explore the use of natural gradients as introduced in [1] and [9]. Natural gradients account for the non-Euclidean geometry of parameters of probability distributions by looking for directions of optimal descent in KL-divergence balls.

4.1. Two Required Results from Statistics

The first required result[1] is that the Hessian of KL-divergence between two distributions $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}_1)$, with

[1] the lengthy derivation is ommitted with only the result included.

respect to $\boldsymbol{\theta}_1$ evaluated at $(\boldsymbol{\theta})$ is the Fisher information matrix.

$$\nabla^2_{\boldsymbol{\theta}_1} D_{KL}(p(\boldsymbol{\theta})||p(\boldsymbol{\theta}_1)) = \mathbf{F} \\ = \mathbb{E}_{p(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta})^\intercal \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta})]$$

The second required results is that the second-order Taylor approximation for $D_{KL}(p(\boldsymbol{\theta})||p(\boldsymbol{\theta}+\delta\boldsymbol{\theta}))$ can be approximated by (assuming $\delta\boldsymbol{\theta} \to 0$),

$$D_{KL}(p(\boldsymbol{\theta})||p(\boldsymbol{\theta}+\delta\boldsymbol{\theta})) \approx \frac{1}{2}\delta\boldsymbol{\theta}^\intercal \mathbf{F} \delta\boldsymbol{\theta} \tag{6}$$

Informally we understand D_{KL} acts as a local distance measure between distributions. With the Fisher result we conceive an induced Riemannian manifold over $p(\boldsymbol{\theta})$ with $\mathbf{F}$ as metric.

4.2. Steepest Descent Directions in D_{KL} balls

Given a loss function to be minimized, in our case the negative log-likelihood parameterized by $\mathbf{W}$, denoted $\mathcal{L}(\mathbf{W})$.

Consider the minimization problem,

$$\arg\min_d \mathcal{L}(\mathbf{W}+d) \\ \text{s.t}\, D_{KL}(p(\mathbf{W})||p(\mathbf{W}+d)) = \epsilon$$

where $\epsilon > 0$ and $p(\mathbf{W})$ denotes a distribution parameterized by $\mathbf{W}$. The variable d denotes a small "pertubation" to $\mathbf{W}$.

The constraint is added to ensure the trivial solution is not in the solution set. More importantly using this constraint we ensure that we descend on the distribution manifold with constant speed, without being slowed down by its curvature. This makes learning locally invariant to parameterizations of the model.

Writing the optimization problem in Lagrangian form with $\mathcal{L}(\mathbf{W}+d)$ in its first-order Taylor approximation and $D_{KL}(p(\mathbf{W})||p(\mathbf{W}+d))$ with (6):

$$d^* = \arg\min_d \mathcal{L}(\mathbf{W}+d) + \lambda(D_{KL}(p(\mathbf{W})||p(\mathbf{W}+d)) + \epsilon) \\ \approx \arg\min_d \mathcal{L}(\mathbf{W}) + \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})^\intercal d + \frac{\lambda}{2} d^\intercal \mathbf{F} d$$

Set the derivative with respect to d to zero:

$$0 = \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}) + \lambda \mathbf{F} d \\ \implies d = -\frac{1}{\lambda}\mathbf{F}^{-1}\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})$$

This tells us the update step should be taken as,

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha_t\, \mathbf{F}^{-1}\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}_{t+1}) \tag{7}$$

This is update step that we will take. Here, $\mathbf{F}^{-1}$ act as a gradient correction according to local curvature from the KL-divergence surface.

4.3. As Found Natural Gradient

After thinking about how to compute $\mathbf{F}$ for the problem and large system (the neural network), I stumbled across Vadam [5] which does exactly what I wanted to do. It is implemented in the same framework; and I used it to empirically compare the convergence rate of two optimization algorithm, Vadam and Adam.

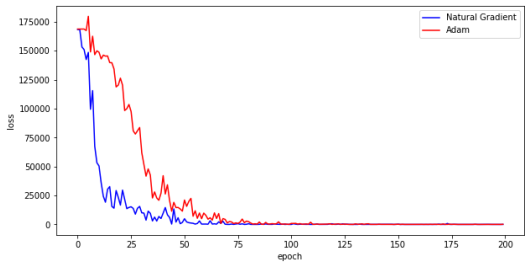

Figure 1. The graph shows the loss versus epoch iteration of two optimization algorithm, Vadam and Adam. The blue is Vadam which uses natural gradient for update clearly converges to a loss quicker than Adam in red.

We see from Figure 1 that we get a quicker convergence by using natural gradient (blue) to train our model than using first-order method (red).

5. Implementation and Results

6. Discussion

Note that Vadam works very well in this case because both Vadam and I have the same assumption about the form of variation distributions and its prior. Further, Vadam is developed exactly to adapt the Adam algorithm for variational inference through natural gradient.

Although we seen how using natural gradient helps us to achieve faster convergence rate, broader application of natural gradient descant for neural network training remains restricted by the problem domain. Another criticism of the use of natural gradient is that, it is unclear whether the faster convergence in epoch step is justified by the slow computation of FIM.

7. Conclusion

In this report we introduced the unsupervised domain adaptation problem and leveraged variational inference technique to learn a prior distribution for domain adaptation. In addition, we shown experimentally that using natural gradient results in quicker convergence in terms of epoch step for variational inference.

In conclusion, when the situation arises, we can consider using natural gradient for model training as it yields faster model convergence.

References

[1] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998. 2

[2] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 1

[3] S. Benaim and L. Wolf. One-sided unsupervised domain mapping. In *Advances in neural information processing systems*, pages 752–762, 2017. 1

[4] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739, 2017. 1

[5] M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018. 1, 3

[6] M. Kim, P. Sahu, B. Gholami, and V. Pavlovic. Unsupervised visual domain adaptation: A deep max-margin gaussian process approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4380–4390, 2019. 1

[7] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017. 1

[8] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016. 1

[9] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013. 2

[10] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017. 1

[11] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016. 1

Appendix

Team contributions

Pascale Walters As the only member of the group, I was responsible for all parts.