# CSCI 1430 Final Project Report: Your project title

*Team name*: First member, second member, third member, fourth member.
Brown University
19th December 2017

## Abstract

*In this project, the application of variational inference for visual domain adaptation problem is explored. We leverage the power of deep neural network to learn features from data set, and construct a family of distribution for domain adaptation. We will first introduce the visual domain adaptation problem. Secondly, we will recap the basics of variational inference technique. Thirdly, this report will explore the use of natural gradient descent to improve variational inference.*

## 1. Domain Adaptation

Domain Adaptation is the task of adapting a model trained in one domain, called the *source* domain, to another domain, called the *target* domain. The source domain data is usually fully labeled but we only have access to data from target domain with no label (hence unsupervised). Although there are several different setup of the problem, we only focus here on the unsupervised domain adaptation with classification as objective.

Formally we have, a joint space of inputs and class labels, $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y} = \{1, \ldots, K\}$. Suppose we have two domains on this space, the **source (S)** and the **target (T)**, defined by unknown distributions $p_S(\mathbf{x}, y)$ and $p_T(\mathbf{x}, y)$. We are given source-domain training examples with labels $\mathcal{D}_S = \{\mathbf{x}_i^S, y_i\}_{i=1}^{N_S}$ and target data $\mathcal{D}_T = \{\mathbf{x}_I^T\}_{i=1}^{N_T}$ without labels. Assuming a **shared set of class labels** between the two domains. The goal is to assign the correct class labels to target data points.

This problem is tackled in a shared latent space framework, where we seek to learn an embedding function $G : \mathcal{X} \to \mathcal{Z}$ and a classifier $h : \mathcal{Z} \to \mathcal{Y}$. The functions $G(\cdot)$ and $h(\cdot)$ are shared across both domain and will be used to classify samples from target domain, i.e. $y = h(G(\mathbf{x}))$, where $\mathbf{x} \sim p_T$.

### 1.1. 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,

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

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

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

### 1.2. Deep Kernel Trick

Up to this point, the $\kappa_j$'s in (1) is only computable by limited choice of $\kappa_j$. We use the deep kernel trick to elevate this problem but modeling the $\kappa_j$ with 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, and the next section will explain a technique to approximate it.

## 2. Variational Inference

Variation Bayesian methods is a family of techniques for approximating intractable (posterior) distribution arising in Bayesian inference. Given an intractable posterior distribution $p(\mathbf{z}|\mathbf{x})$, where $\mathbf{z}$ is 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,

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

For the approximation we look at the KL-divergence,

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

The last line follows from that we are optimizing over $q(\mathbf{z})$ and $\log p(\mathbf{x})$ is a constant with respect $q(\mathbf{z})$. Replace $\mathbf{z}$ with $\mathbf{W}$ and $\mathbf{x}$ with $\mathcal{D}_S$; and this means minimizing $D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$ is equivalent to the following optimization,

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

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

## 3. Natural Gradient

Notice that (5) is an optimization in the distribution space, but steps taken in the parameter space. There could be two different parametrization of $q$ that yield the same results. Hence doing gradient descent with the parameters directly may yield wasteful update step and leading to slow convergence.

We will explore the use of natural gradients as introduced in [1] and [3]. It accounts for the non-Euclidean geometry of parameters of probability distributions by looking for optimal descent directions in KL-divergence balls.

### 3.1. Two Results from Statistics

The first result[1] is that the Hessian of KL-divergence between two distributions $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}_1)$, with respect to $\boldsymbol{\theta}_1$ evaluated at $(\boldsymbol{\theta})$ is Fisher information matrix.

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

[1]This is just a long derivation, there is nothing very deep, that is why it is skipped.

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

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

The intuition we get here is that $D_{KL}$ acts as a local distance measure for distribution nearby. Thus inducing a Riemannian manifold on $p(\boldsymbol{\theta})$ with $\mathbf{F}$ as metric.

### 3.2. 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,

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

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 descent on the distribution manifold with constant speed, without being slowed down by its curvature. This makes learning locally invariant to parametrizations 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):

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

Set the derivative with respect to $d$ to zero:

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

This tells us the update step should be taken as,

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

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. Experimental Result

After thinking about how to compute FIM for the problem and large system (the neural network), I stumbled across Vadam [2] which does exactly what I wanted to do. It is

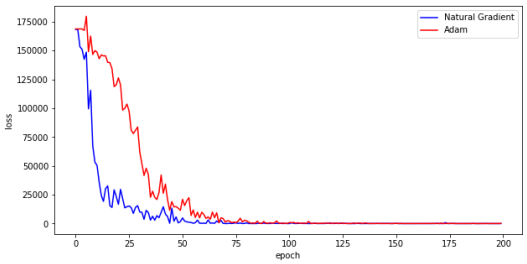


Figure 1. The graph shows the loss versus epoch iteration of two optimization algorithm, Vadam and Adam. The blue is Vadam which uses natural gradient for update clearly converges to a loss quicker than Adam in red.

implemented in the same framework; and I used it to empirically compare the convergence rate of two optimization algorithm, Vadam and Adam.

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. 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.

## 6. 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] 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. 2

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