基于深度神经网络的迁移学习系列一

ZZL

2017 年 11 月 28 日

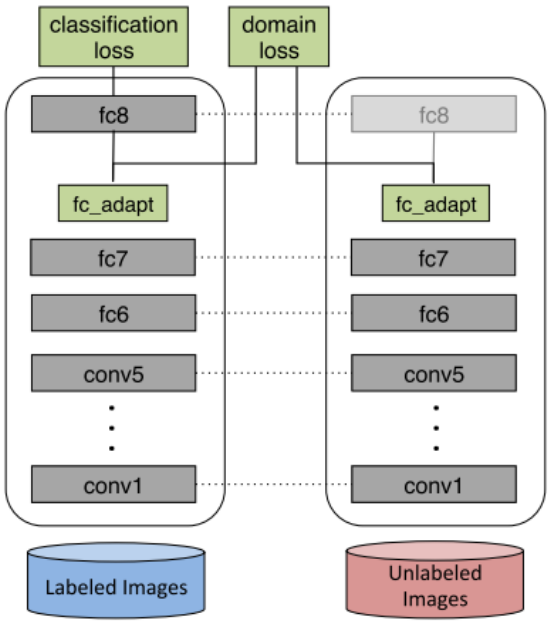
Introduction

本文介绍基于深度神经网络的迁移学习。按照优化目标可以分为基于 TCA 的深度网络实现和基于 JDA 的深度网络实现，按照网络结构可以分为权重共享与不共享两类。深度神经网络只是方法，优化目标在基于传统方法的文章中给出，不能舍本逐末。本文中介绍以下三篇文章《Deep Domain Confusion: Maximizing for Domain Invariance》,《Learning Transferable Features with Deep Adaptation Networks》,《Beyond SharingWeights for Deep Domain Adaptation》,《Deep Transfer Network: Unsupervised Domain Adaptation》。

一. Deep Domain Confusion: Maximizing for Domain Invariance

本文基于的思想是《How transferable are features in deep neural networks?》上讨论的内容，对于一个深度网络，随着网络层数的加深，网络越来越依赖于特定任务；而浅层相对来说只是学习一个大概的特征。不同任务的网络中，浅层的特征基本是通用的。这就启发我们，如果要适配一个网络，重点是要适配高层——那些 task-specific 的层。

本文针对于预训练的 AlexNet（8 层）网络，在第 7 层（也就是 feature 层，softmax 的上一层）根据 feature 来计算源域数据与目标域数据之间分布的距离（MMD 距离），并把该距离作为模型损失的一部分。模型的优化目标是，使得源域与目标域数据分布更接近并且使得模型对源域数据得分类效果最好。这个方法简称为 DDC。



网络结构

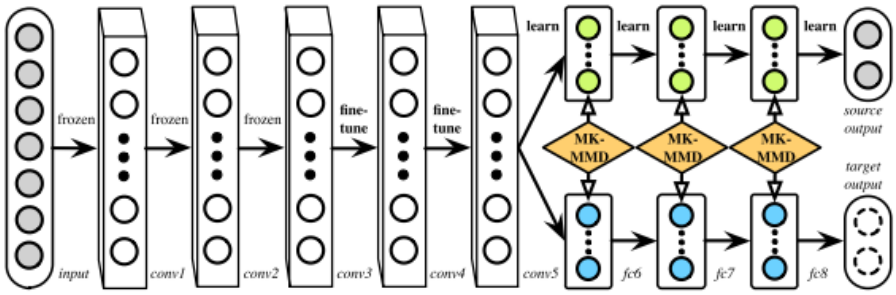
损失函数如下：$\mathrm{L} = L_C(X_L, y) + \lambda\, MMD^2(X_S, X_T)$。其中$MMD^2(X_S, X_T)$表示$X_S$、$X_T$经过核函数映射后的距离。

总结：

这个工作算是深度网络与迁移学习结合得很基础得工作，但他的思想有一定得借鉴价值。在以往的浅层模型中，被迁移的特征对于图像来说是将 n*n 的图像拉成一个 n*n 维的向量，对于文本来说则直接用词袋模型表示。这里是把网络当作一个提取特征得工具，相比于浅层模型优势在于迁移前的特征更加强大，效果会更好。但缺点也比较明显，对于传统浅层方法，他们的模型是适合图像和文本任务的，因为被迁移的特征足够简单，而现在我们用网络做特征提取，文本需要一层 CNN，图像需要 3 层 CNN，这就限制了模型的使用范围。

二．Learning Transferable Features with Deep Adaptation Networks

DAN 方法是在 DDC 方法的基础上发展起来的，它很好的解决了 DDC 的两个问题：一 是 DDC 只适配了一层网络，可能还是不够，因为 Jason 的工作中已经明确指出不同层都是可以迁移的。所以 DAN 就多适配几层；二是 DDC 是用了单一核的 MMD，单一固定的核可能不是最优的核。DAN 用了多核的 MMD（MK-MMD），效果比 DDC 更好。DAN 的网络结构如下：



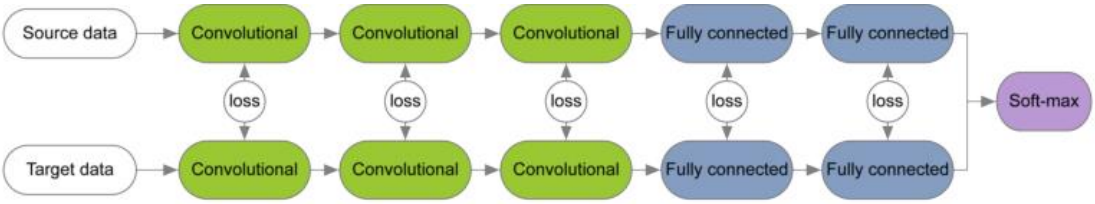
关于适配多层就不在叙述，下面介绍多核 MMD。

多核 MMD。这里的 MK-MMD 是衡量源域与目标域数据之间的差异，这种差异作为损失的一部分。Long 在文章中直接提到用多核来衡两域之间的距离。多核可以表示为 $\mathcal{K} \triangleq \left\{ k = \sum_{u=1}^{m} \beta_u k_u : \beta_u \geq 0, \forall u \right\}$，关于基于多核的 MMD 的无偏估计，以及对于权重系数的估计就不在这叙述。

我认为这里采用多核比单核好有以下原因：MMD 距离是在函数集中找到一个函数，使得两个分布经过这个函数映射后的距离最大，把这个最大的距离当作两个分布之间的距离，但不能精准的找到这么一个函数。在网络中，参数的更新会导致每一层上数据分布的变化，单一的核函数在每轮训练中对距离的表示的效果是会有变化的。而且采用单一的核函数，它的效果的好坏有些碰运气的成分。采用多核，可以通过调整每个核的权重来使得对距离衡量的效果尽可能好并且适合网络中数据分布的变化。

三．Beyond Sharing Weights for Deep Domain Adaptation

这篇文章与 DDC 相比，最大的区别在于他的网络层没有共享权重。DDC 图中虽然有两个网络，但他们的 conv1 到 fc8 层的权重都是一样的。在本文中，源域与目标域各自对应层的权重是相关但不同。网络结构如下：



网络结构

本文的核心在于两个网络对应的权重是相关但不同的，他们需要尽可能的接近。该网络的损失函数如下

$$L(\theta^s,\theta^t|\mathbf{X}^s,Y^s,\mathbf{X}^t,Y^t)=L_s+L_t+L_w+L_{MMD},$$

$$L_s=\frac{1}{N^s}\sum_{i=1}^{N^s}c(\theta^s|\mathbf{x}_i^s,y_i^s)$$

$$L_t=\frac{1}{N_l^t}\sum_{i=1}^{N_l^t}c(\theta^t|\mathbf{x}_i^t,y_i^t)$$

$$L_w=\lambda_w\sum_{j\in\Omega}r_w(\theta_j^s,\theta_j^t)$$

$$L_{MMD}=\lambda_u r_u(\theta^s,\theta^t|\mathbf{X}^s,\mathbf{X}^t)\ ,$$

L_s表示源域数据的分类误差；L_t表示目标域数据的分类误差（有极少量样本）L_w表示两个网络相关层之前权重的差异，这里Ω表示需要进行权重参数约束的层的集合；L_{MMD}表示域源与目标域之间的差异。

L_w 一般的想法是取L_2范数，但作者认为L_2范数对微小的扰动太敏感，于是作者在这里运用了指数损失 $r_w(\theta_j^s,\theta_j^t)=\exp\left(\|\theta_j^s-\theta_j^t\|^2\right)-1$。这样的指数损失函数的目标是使得两者权重更加接近，而这样对于不同分布的域来说太苛刻了，于是作者采用 $r_w(\theta_j^s,\theta_j^t)=\exp\left(\|a_j\theta_j^s+b_j-\theta_j^t\|^2\right)-1$。这里a、b也是网络要学习的参数。

$\mathbf{f}_i^s=\mathbf{f}_i^s(\theta^s,\mathbf{x}_i^s)$表示源域网络最后一层的特征表示，$f_i^t$表示目标域网络最后一层的特征表示。文本用MMD的平方来衡量两者之间的距离 $\mathrm{MMD}^2(\{\mathbf{f}_i^s\},\{\mathbf{f}_j^t\})=\left\|\frac{1}{N^s}\sum_{i=1}^{N^s}\phi(\mathbf{f}_i^s)-\frac{1}{N^t}\sum_{j=1}^{N^t}\phi(\mathbf{f}_j^t)\right\|^2$。

$\emptyset(\cdot)$ 表示映射函数，展开上式 $\mathrm{MMD}^2(\{\mathbf{f}_i^s\},\{\mathbf{f}_i^t\})=\sum_{i,i'}\frac{\phi(\mathbf{f}_i^s)^T\phi(\mathbf{f}_{i'}^s)}{(N^s)^2}-2\sum_{i,j}\frac{\phi(\mathbf{f}_i^s)^T\phi(\mathbf{f}_j^t)}{N^sN^t}+\sum_{j,j'}\frac{\phi(\mathbf{f}_j^t)^T\phi(\mathbf{f}_{j'}^t)}{(N^t)^2}$。

这里写作核函数的形式为 $r_u(\theta^s,\theta^t|\mathbf{X}^s,\mathbf{X}^t)=\sum_{i,i'}\frac{k(\mathbf{f}_i^s,\mathbf{f}_{i'}^s)}{(N^s)^2}-2\sum_{i,j}\frac{k(\mathbf{f}_i^s,\mathbf{f}_j^t)}{N^sN^t}+\sum_{j,j'}\frac{k(\mathbf{f}_j^t,\mathbf{f}_{j'}^t)}{(N^t)^2}$。本文采用径向基核函数$\mathrm{K(u,v)}=\exp(-\|u-v\|^2/\sigma_2)$，实验中$\sigma_2$设为1。

本文先训练源域网络，作为一个单纯的分类任务，然后再用源域网络初始化目标域网络，接着两个网络联合训练。对于$\mathrm{j}\in\Omega$，初始化$a_j=1$、$b_j=0$。

四. Deep Transfer Network: Unsupervised Domain Adaptation

这篇文章可以看作是 JDA 的深度网络实现版，这篇文章中网络的输出是属于每个类的概率，我们得到的是 P(Y|X)，因此我们不必像 JDA 原文中用 P(X|Y)来代替 P(Y|X)。

Deep Transfer Network（DTN）包含两种类型的网络层，一种是基于共享权重的特征提取层，为了学习到一个子空间能够使得源域和目标域之间的边缘分布更近；另一种是分类层，是使得两个域的条件概率分布更近。

令 f 表示激活函数，h 表示下一层的激活值，W 表示权重，则有 h=f(Wx)。一个 L 层的网络，前 L-1 层为特征提取层，最后一层为分类层，本文在分类层之间来适配两个域的边缘分布。$\mathbf{H}^s(l-1)=[\mathbf{h}_1^s(l-1),\ldots,\mathbf{h}_{n^s}^s(l-1)]\in\mathbb{R}^{k\times n^s}$表示源域的特征矩阵，$\mathbf{H}^t(l-1)$表示目标域的特征矩阵。两域之间的距离表示为：

$$\begin{aligned}\mathrm{MMD}_{mar} &= \| \frac{1}{n^s}\sum_{i=1}^{n^s}\mathbf{h}_i^s(l-1)-\frac{1}{n^t}\sum_{j=1}^{n^t}\mathbf{h}_j^t(l-1)\ \|_2^2 \\ &= \mathrm{Tr}(\mathbf{H}(l-1)\mathbf{M}\mathbf{H}^T(l-1)),\end{aligned}$$

MMD 作为网络的损失之一（这里直接求数据之间的距离，没有经过映射函数的处理）。

本文用逻辑回归作为网络的分类层，具体公式如下 $P(y=c|\mathbf{x})=\mathrm{softmax}_c(\mathbf{w}_c^T\mathbf{x})=\frac{e^{\mathbf{w}_c^T\mathbf{x}}}{\sum_j e^{\mathbf{w}_j^T\mathbf{x}}}$.

这里得到了 P(Y|X)，不需要在用 P(X|Y)近似 P(Y|X)。联合概率分布距离如下：

$$\begin{aligned}&\mathrm{MMD}_{con} \\ &=\sum_{c=1}^{C}\| \frac{1}{n^s}\sum_{i=1}^{n^s}P(y^s=c|\mathbf{x}_i^s)-\frac{1}{n^t}\sum_{j=1}^{n^t}P(y^t=c|\mathbf{x}_j^t)\ \|^2 \\ &=\sum_{c=1}^{C}\mathbf{q}_c^T\mathbf{M}\mathbf{q}_c,\end{aligned}$$

最终的损失函数如下：$\mathrm{J(W)}=-\mathrm{L(W)}+\alpha MMD_{mar}+\mu MMD_{con}$。其中 $-\mathrm{L(W)}=-\sum_{i=1}^{n}\log(P(y=y_i|x_i,W))$表示分类的损失。

Algorithm 1 Optimization of the Deep Transfer Network

Input: Source data with label $\mathbf{X}^s, Y^s$ and target data $\mathbf{X}^t$.
Output: Parameters of deep transfer network $\mathbf{W}$ and predicted labels of the target samples $\bar{Y}^t$

1: **begin:**
2: Set $i=0$. Get $\bar{Y}_0^t$ by training a baseline neural network with $\mathbf{X}^s, Y^s$ and testing with $\mathbf{X}^t$.
3: **repeat**
4: $i=i+1$.
5: Make mini-batches.
6: Get $\mathbf{W}$ by optimizing Eq.8 using source data $\mathbf{X}^s, Y^s$ and target data $\mathbf{X}^t, \bar{Y}_{i-1}^t$.
7: Predict $\mathbf{X}^t$ with the network to get $\bar{Y}_i^t$.
8: **until** $\bar{Y}_i^t=\bar{Y}_{i-1}^t$ **or** $i>T$.
9: **return** $\mathbf{W}, \bar{Y}^t=\bar{Y}_i^t$.

JDA 算法需要迭代处理，而神经网络的计算中也有这种特性。

参考文献

[1] Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. Deep domain confusion: Maximizing for domain invariance. Technical report, arXiv:1412.3474, 2014

[2]Long M, Cao Y, Wang J, et al. Learning transferable features with deep adaptation networks[C]//International Conference on Machine Learning. 2015: 97-105.

[3] https://zhuanlan.zhihu.com/p/27657910

[4] Rozantsev A, Salzmann M, Fua P. Beyond sharing weights for deep domain adaptation[J]. arXiv preprint arXiv:1603.06432, 2016

[5] Zhang X, Yu F X, Chang S F, et al. Deep transfer network: Unsupervised domain adaptation[J]. arXiv preprint arXiv:1503.00591, 2015.

[6] https://zhuanlan.zhihu.com/p/27657910

[7] Gretton A, Borgwardt K M, Rasch M J, et al. A kernel two-sample test