



华为云HERO高校联盟知识峰会

小白也能懂的迁移学习

—算法和案例分析



分享嘉宾：
李新春

目录

- ❑ 迁移学习概述
- ❑ 迁移框架、算法
- ❑ 案例展示（NAIE平台）

迁移学习概述

- 机器学习模型的训练需要大量标记数据，然而新场景或者某些场景本身就会遇到数据很难收集以及标注困难的问题
- 解决方案：构造实验数据、寻求以往相似场景模型等等 → 但是“相似”不意味着完全相同，数据之间仍然存在分布差异

现实场景



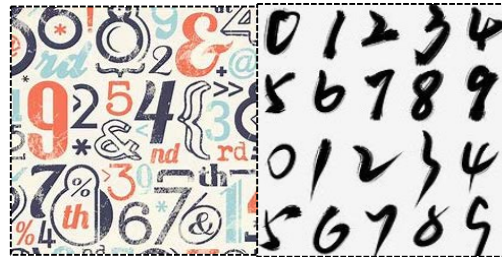
自动驾驶数据收集和标注困难

人类行为经验



人类学习开车时借鉴自行车的经验

知识迁移难点



手写数字的风格存在很大的差异

定义

作用

区别

分类

难点

应用

迁移学习概述-定义、作用

- 迁移学习主要是指将已有领域（**源域**）的知识迁移到新的场景（**目标域**）的过程，目的时辅助目标域快速有效地部署好的模型



源域



迁移学习

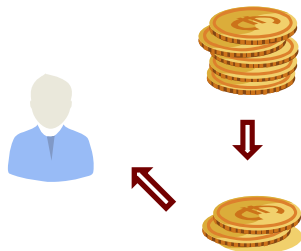


目标域

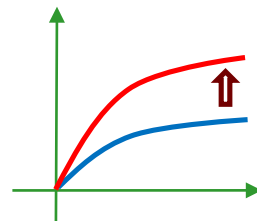
- 迁移学习具有节省时间成本、节省标注成本、提升模型性能等优点，可以解决目标域缺乏算力、缺乏有效标记数据等难点



减少新场景下模型训练的时间



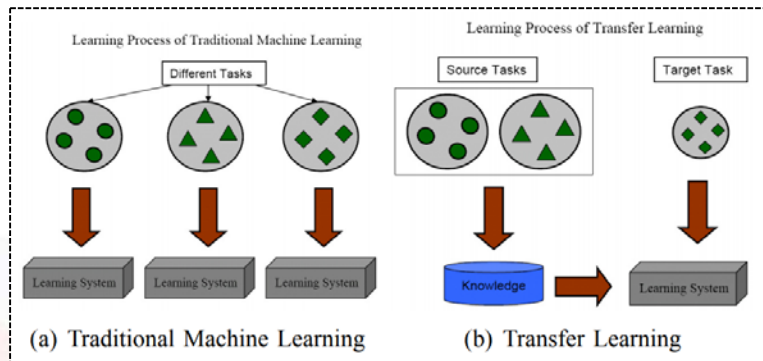
新场景标记样本不足，降低标注成本



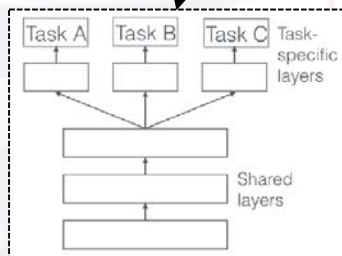
利用任务之间的相关性提升模型性能

迁移学习概述-区别

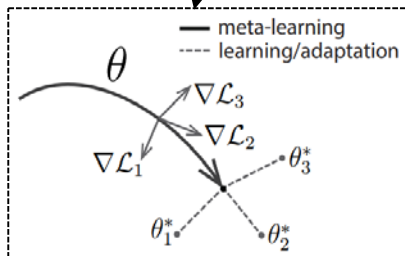
- 迁移学习与传统机器学习的区别、与其余研究领域的联系



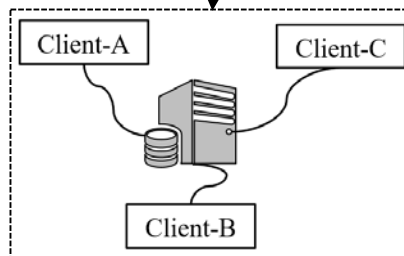
知识共享
知识传递



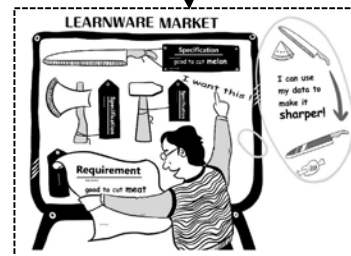
Multi-Task Learning
多任务学习



Meta Learning
元学习



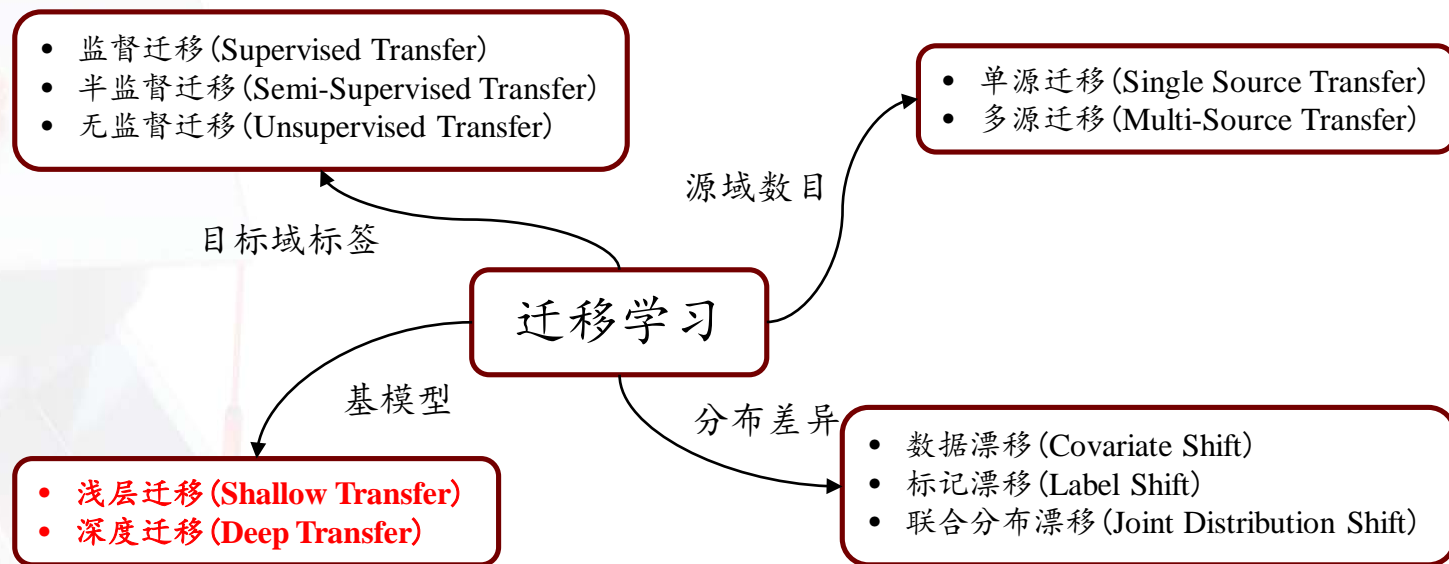
Federated Learning
联邦学习



Model Reuse
模型复用

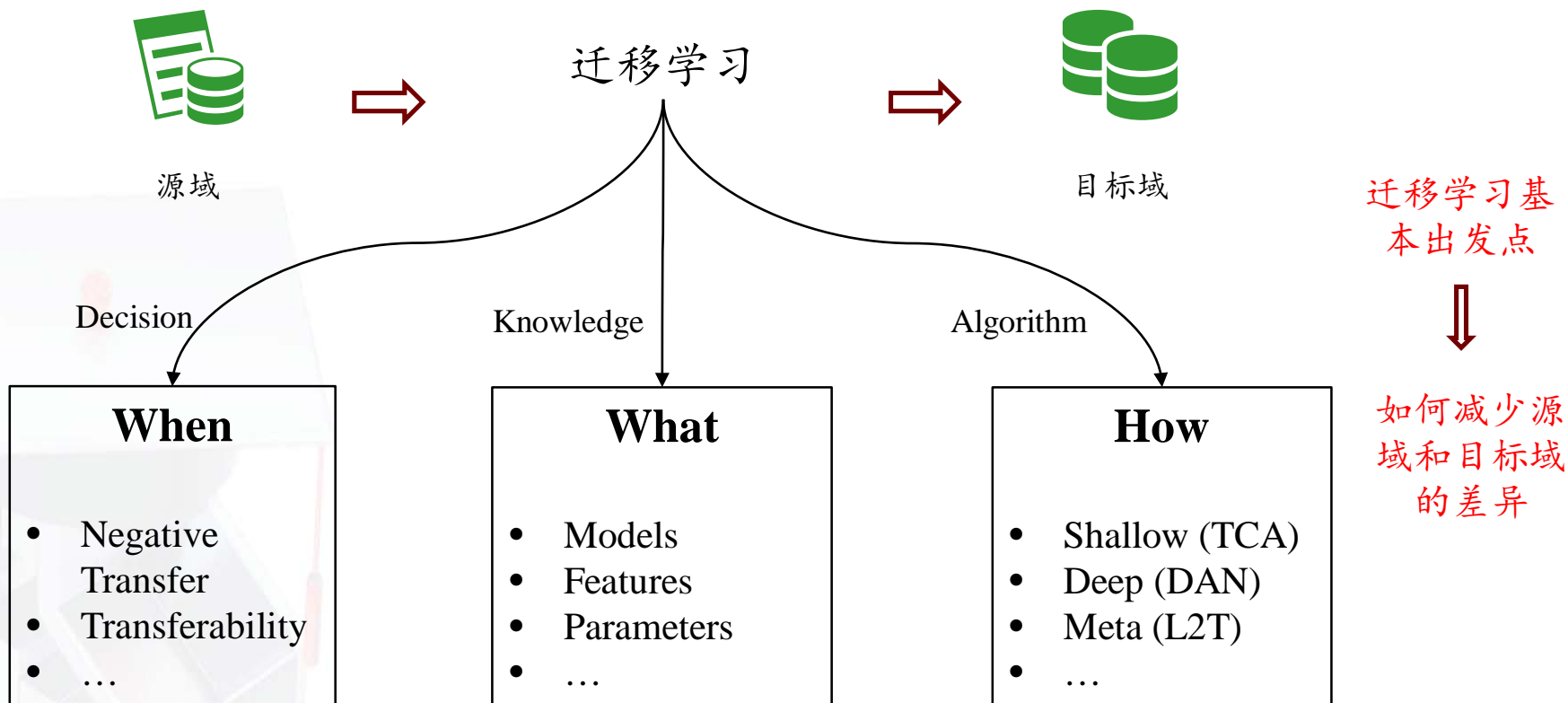
迁移学习概述-分类

- 迁移学习有很多子领域，这里给出主要的几种分类依据：
 - 目标域是否有标签：监督迁移、半监督迁移、无监督迁移等；
 - 源域目标域分布差异情况：数据漂移、标记漂移、联合分布漂移等；
 - 源域数目：单源迁移、多源迁移等；
 - 使用的基模型：浅层迁移、深度迁移等；



迁移学习概述-难点

- 迁移学习三大核心难题：是否要迁移（When）、迁移什么（What）、如何迁移（How）



Sinno Jialin Pan, Qiang Yang: A Survey on Transfer Learning. TKDE 2010.

迁移学习概述-应用

- 迁移学习在计算机视觉、自然语言处理、推荐系统、强化学习等领域得到了具体的应用

计算机视觉

- 医疗图像识别
- 安防监控识别
- 3D场景分析



自然语言处理

- 多语种、小语种
- 智能聊天对话
- 跨领域文本分类



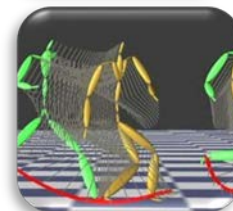
推荐系统

- 多用户推荐建模
- 跨平台推荐
- 季节性变化推荐



强化学习

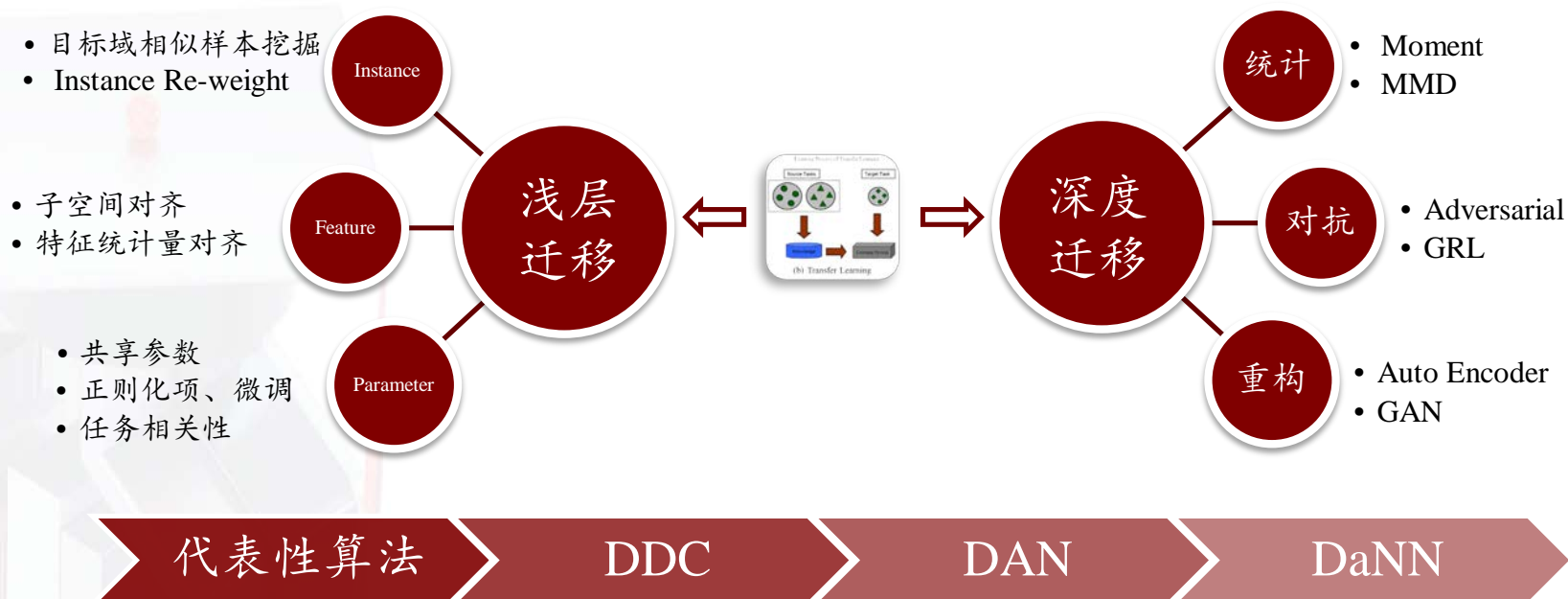
- 多场景路面检测
- 自动驾驶
- 仿真→真实环境



计算机网络：恶意软件分类；电信领域：跨区域流量预测、Wifi信号定位；地理学：遥感图像检测……

迁移框架、算法

- 迁移学习根据使用的基模型进行划分，可以分为浅层迁移算法和深度迁移算法
- 浅层迁移学习方法主要是指深度学习没有兴起之前的一些基于线性分类器、距离度量分类器等方法的迁移算法，主要包括基于**样本**、基于**特征**、基于**参数**的迁移等
- 深度迁移学习方法主要是指基于深度网络，可以使用梯度算法进行端到端优化训练的迁移算法，主要包括基于**统计**、基于**对抗**、基于**重构**的迁移等



迁移方法-代表性算法

浅层迁移算法举例

算法	出处	描述（优化目标）
KMM	NeurIPS2006	样本迁移、MMD距离
KLIEP	NeurIPS2006	样本迁移、KL距离
MMDE	AAAI2008	优化核函数矩阵使MMD最小
TCA	IJCAI2009	隐空间MMD最小
GFK	CVPR2012	子空间流形对齐
ITL	ICML2012	基于信息论的对齐
MSDA	ICML2012	基于简单重构的对齐
SA	ICCV2013	子空间对齐
GTL	TKDE2014	非负矩阵分解特征对齐
CORAL	AAAI2016	二阶协方差矩阵对齐

Sinno Jialin Pan, Qiang Yang: A Survey on Transfer Learning.
TKDE 2010.

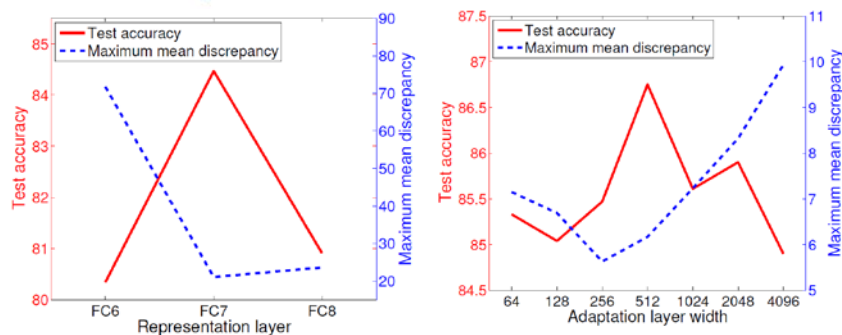
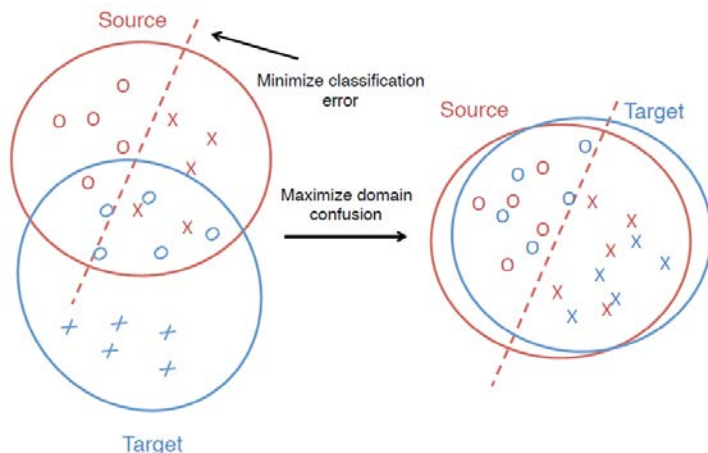
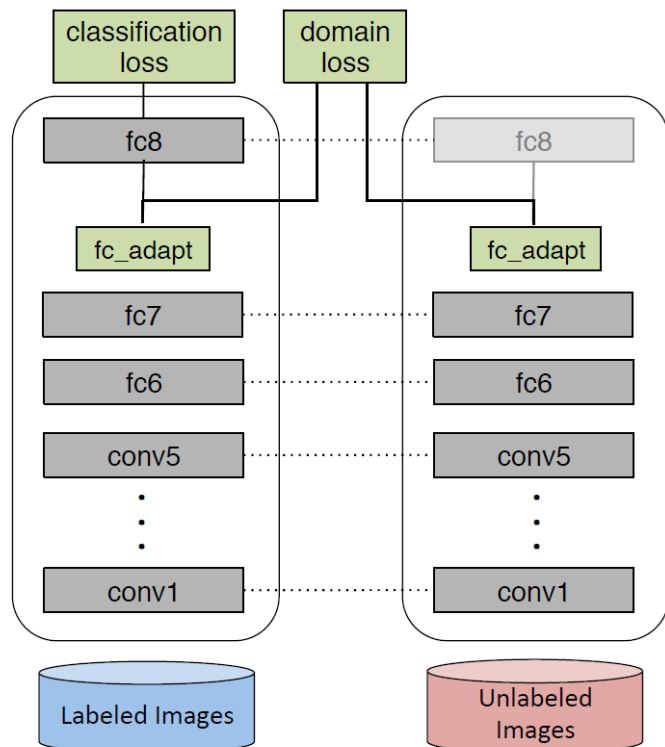
深度迁移算法举例

算法	出处	描述（优化目标）
DDC	CoRR2014	MMD损失
DAN	ICML2015	多层MK-MMD
DaNN	ICML2015	对抗训练和梯度反转层
DeepCoral	ECCV2016	二阶方差对齐损失
CoGAN	NeurIPS2016	两支半共享生成器
JAN	ICML2017	多隐层数据联合对齐
ADDA	CVPR2017	非共享判别式对抗训练
CDAN	NeurIPS2018	联合特征和分类器输出
MCD	CVPR2018	两个分类器取代域分类器
GTA	CVPR2018	通过生成网络进行对齐

Mei Wang, Weihong Deng: Deep visual domain adaptation: A
survey. Neurocomputing 2018.

深度迁移方法-DDC

- DDC (Deep Domain Confusion)引入了MMD损失进行对齐源域和目标域的隐层表示



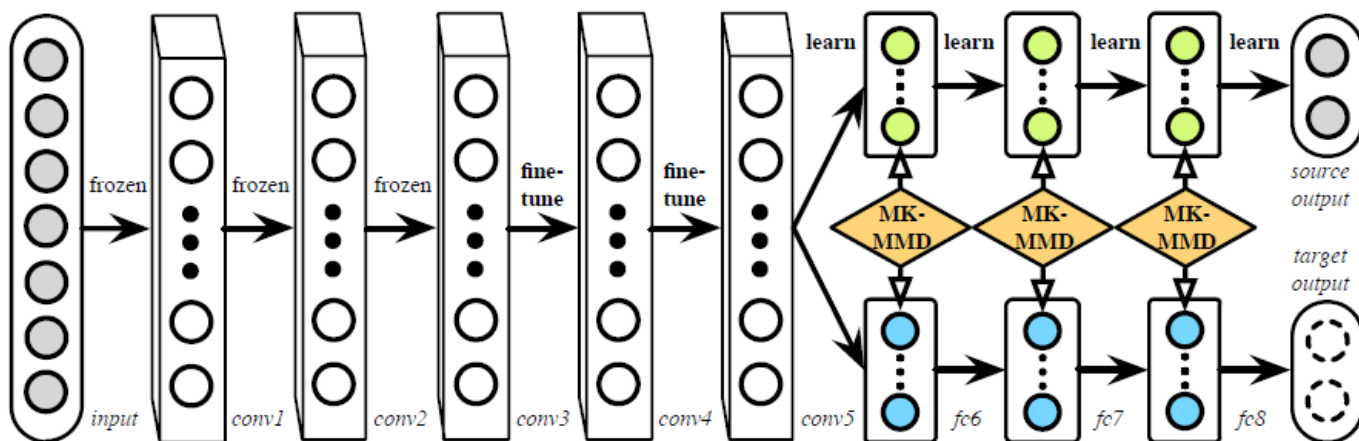
$$\text{mmd}^2(X_S, X_T) = \left\| \frac{1}{N_s} \sum_{i=1}^{N_s} \Phi(x_i^s) - \frac{1}{N_t} \sum_{i=1}^{N_t} \Phi(x_i^t) \right\|_{\mathcal{H}}^2$$

$$\text{loss} = \mathcal{L}(f(g(X_S)), y) + \lambda \cdot \text{mmd}^2(g(X_S), g(X_T))$$

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, Trevor Darrell: Deep Domain Confusion: Maximizing for Domain Invariance. CoRR 2014.

深度迁移方法-DAN

- DAN (Deep Adaptation Network)引入了多核MK-MMD损失进行对齐源域和目标域的隐层表示，并且在多个隐层进行对齐



$$\mathcal{K}(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|_2^2}{\sigma^2}\right)$$

如何选择合适的 σ

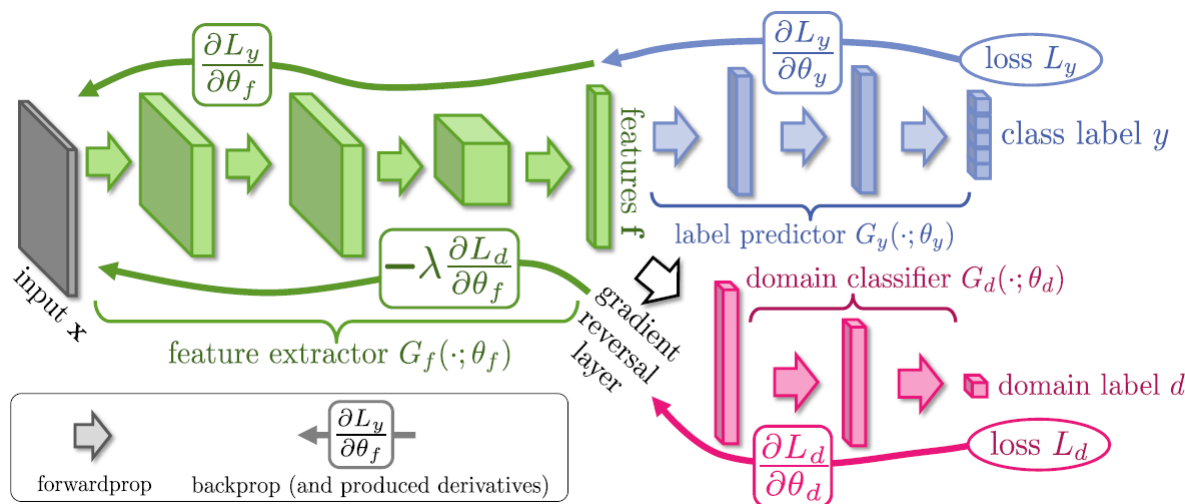


$$\mathcal{K} = \sum_{k=1}^K \beta_k \mathcal{K}_k, \sum_{k=1}^K \beta_k = 1$$

Mingsheng Long, Yue Cao, Jianmin Wang, Michael I. Jordan: Learning Transferable Features with Deep Adaptation Networks. ICML 2015.

深度迁移方法-DaNN

- DaNN (Domain Adversarial Neural Network)引入了GAN的思想对源域和目标域的数据进行对抗训练, Domain Classifier的作用是尽可能地区分源域和目标域的特征, Generator的作用则是提取域无关的特征, 使得Domain Classifier分不开



$$\theta_f^*, \theta_y^* = \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \theta_d)$$

$$\theta_d^* = \arg \max_{\theta_d} E(\theta_f^*, \theta_y^*, \theta_d)$$

- 注: 因为DaNN训练过程加入了梯度反转层, 因此又称为RevGrad(Reversal Gradient)或者GRL(Gradient Reversal Layer)

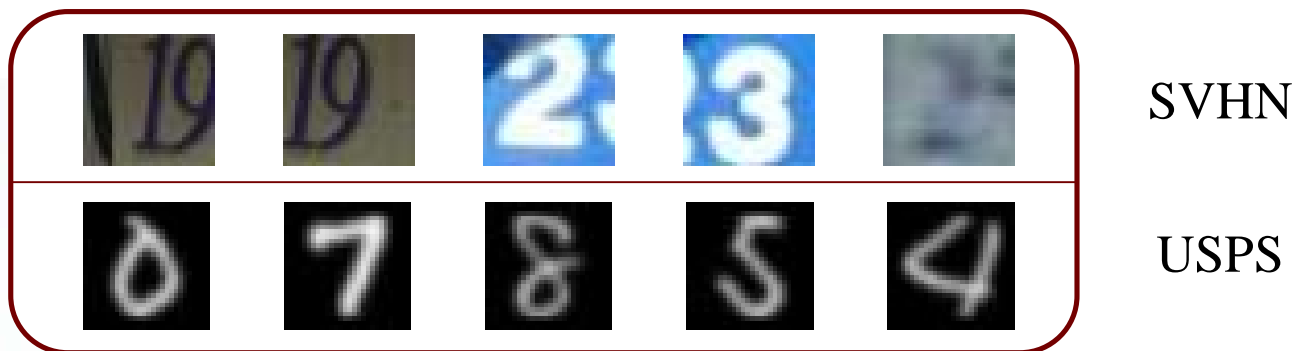
$$R_\lambda(\mathbf{x}) = \mathbf{x}$$

$$\frac{dR_\lambda}{d\mathbf{x}} = -\lambda \mathbf{I}$$

Yaroslav Ganin, Victor S. Lempitsky: Unsupervised Domain Adaptation by Backpropagation. ICML 2015.

案例分析

- 场景描述：有个新的手写数字集SVHN（谷歌街景门牌号码）或者USPS（美国邮政手写数字），需要使用机器学习模型识别其中的数字，但是这个数据集没有标注，不能进行训练，如何得到一个合适的模型？



解决方案

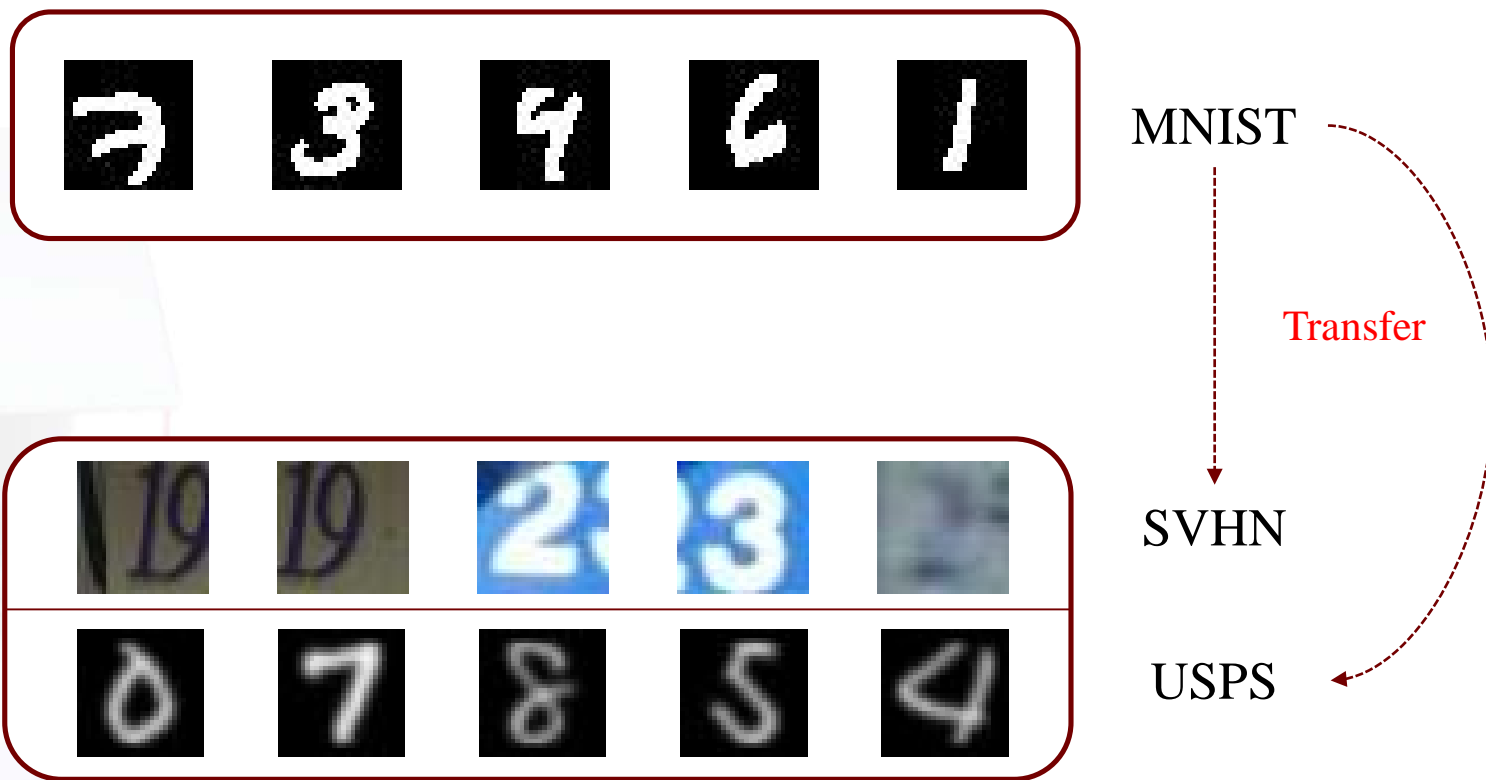
NAIE平台

操作步骤

SDK介绍

案例分析-解决方案

- 解决方案1: 数据集众包标注, 需要花费时间、人力和资金, 且标注质量如何保证?
- 解决方案2: 去Github寻找一个现成的模型 (很多情况下很难找到对应任务的模型)?
- 解决方案3: 寻找相关的数据集, 比如MNIST实验数据集, 来训练一个模型, 然后使用迁移学习。



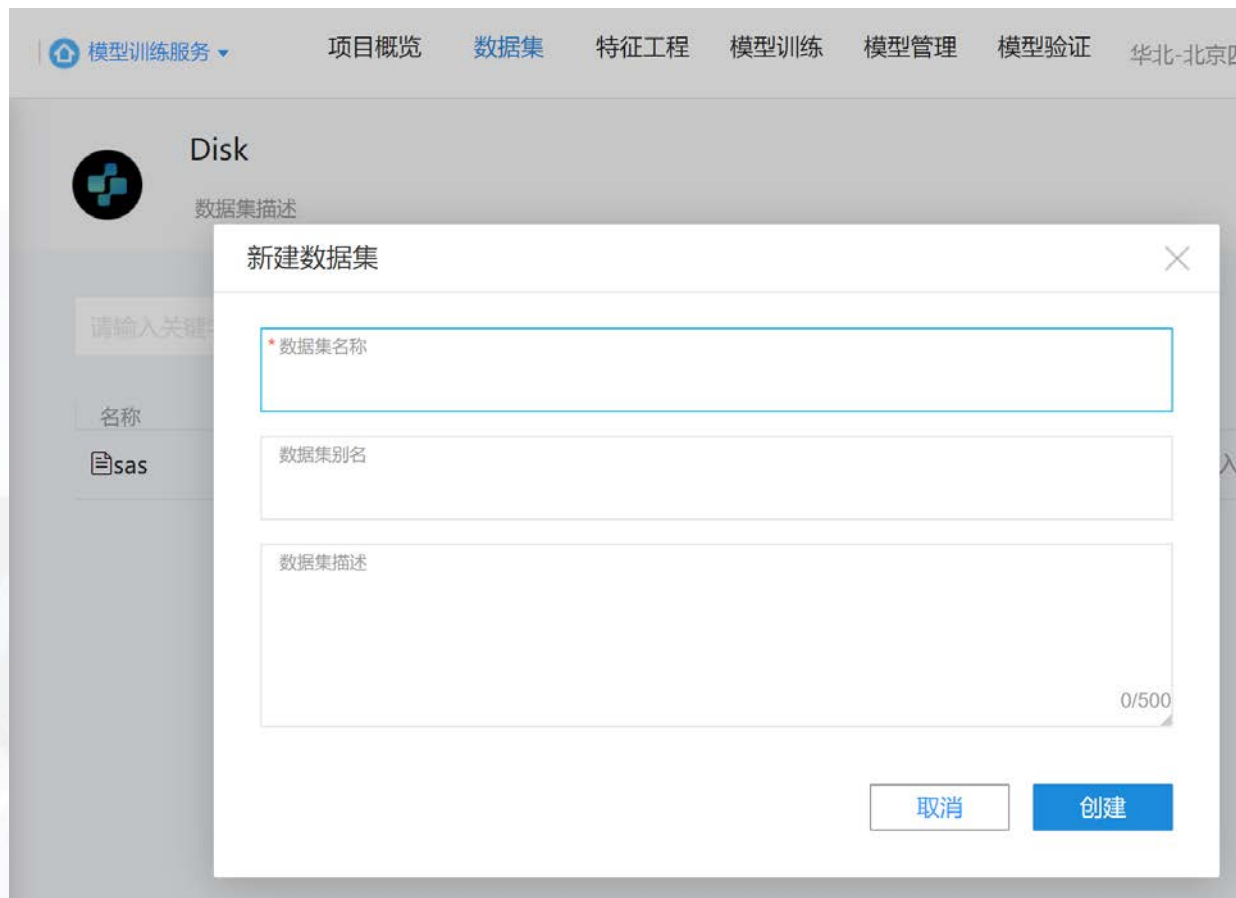
案例分析-NAIE平台

- NAIE平台官网: <https://console.huaweicloud.com/naie/>
- NAIE平台知乎官方账号: <https://www.zhihu.com/people/aigao-shi-qing>



案例分析-步骤

- 步骤一：数据上传，支持文本（json、txt）、图像（jpg、jpeg）及pickle数据，同时支持文件夹上传（10G以内）



The screenshot shows the '新建数据集' (New Dataset) dialog box in the Huawei Model Training Service interface. The dialog box is overlaid on a background showing a 'Disk' section with a '数据描述' (Data Description) tab. The dialog box contains the following fields and buttons:

- 数据集中名称** (Dataset Name): A required text input field, indicated by a red asterisk.
- 数据集别名** (Dataset Alias): A text input field.
- 数据集描述** (Dataset Description): A text area for describing the dataset, with a character count of 0/500.
- 取消** (Cancel): A button to close the dialog without saving.
- 创建** (Create): A blue button to create the new dataset.

案例分析-步骤

• 步骤二：代码编辑

```
def gaussian_kernel(
    source, target, kernel_mul=2.0, kernel_num=5, fix_sigma=None):
    n_samples = int(source.size()[0]) + int(target.size()[0])
    total = torch.cat([source, target], dim=0)

    L2_distance = ((
        total.unsqueeze(dim=1) - total.unsqueeze(dim=0)
    ) ** 2).sum(2)

    if fix_sigma:
        bandwidth = fix_sigma
    else:
        bandwidth = torch.sum(L2_distance.data) / (n_samples * 2 - n_samples)
        bandwidth += 1e-8

    # print("Bandwidth:", bandwidth)
    bandwidth /= kernel_mul * (kernel_num // 2)
    bandwidth_list = [bandwidth * (kernel_mul ** i) for i in range(kernel_num)]
    kernel_val = [
        torch.exp(-L2_distance / band) for band in bandwidth_list
    ]
    return sum(kernel_val)

def mmd_rbf_noaccelerate(
    source, target, kernel_mul=2.0, kernel_num=5, fix_sigma=None):
    batch_size = int(source.size()[0])
    kernels = gaussian_kernel(
        source, target,
        kernel_mul=kernel_mul, kernel_num=kernel_num,
        fix_sigma=fix_sigma
    )
    XX = kernels[:batch_size, :batch_size]
    YY = kernels[batch_size:, batch_size:]
    XY = kernels[:batch_size, batch_size:]
    YX = kernels[batch_size:, :batch_size]
    loss = torch.mean(XX + YY - XY - YX)
    return loss
```

```
def train(model, optimizer, source_loader, target_loader, args):
    model.train()

    iter_source = iter(source_loader)
    iter_target = iter(target_loader)
    num_iter = len(source_loader)

    avg_ce_loss = Averager()
    avg_mmd_loss = Averager()
    avg_acc = Averager()
    for i in range(1, num_iter):
        sx, sy = iter_source.next()

        try:
            tx, _ = iter_target.next()
        except Exception:
            iter_target = iter(target_loader)
            tx, _ = iter_target.next()

        if args.cuda:
            sx, sy = sx.cuda(), sy.cuda()
            tx = tx.cuda()

        sh, logits = model(sx)
        th, _ = model(tx)

        criterion = nn.CrossEntropyLoss()
        ce_loss = criterion(logits, sy)

        mmd_loss = mmd_rbf_noaccelerate(sh, th)

        loss = ce_loss + args.lamb * mmd_loss

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        acc = count_acc(logits, sy)
        avg_acc.add(acc)
        avg_ce_loss.add(ce_loss.item())
        avg_mmd_loss.add(mmd_loss.item())

    ce_loss = avg_ce_loss.item()
    mmd_loss = avg_mmd_loss.item()
    acc = avg_acc.item()
    return ce_loss, mmd_loss, acc
```

案例分析-步骤

- 步骤三：运行和结果分析（Demo数据，原始数据的十分之一）

MNIST→SVHN No Adaptation

[Epoch:0] [TrLoss:0.84050] [TrAcc:0.737] [TeAcc:0.195]
 [Epoch:1] [TrLoss:0.17222] [TrAcc:0.949] [TeAcc:0.219]
 [Epoch:2] [TrLoss:0.11773] [TrAcc:0.964] [TeAcc:0.240]
 [Epoch:3] [TrLoss:0.09100] [TrAcc:0.969] [TeAcc:0.258]
 [Epoch:4] [TrLoss:0.06673] [TrAcc:0.979] [TeAcc:0.247]
 [Epoch:5] [TrLoss:0.02445] [TrAcc:0.992] [TeAcc:0.261]
 [Epoch:6] [TrLoss:0.01452] [TrAcc:0.996] [TeAcc:0.255]
 [Epoch:7] [TrLoss:0.01055] [TrAcc:0.998] [TeAcc:0.261]
 [Epoch:8] [TrLoss:0.00876] [TrAcc:0.999] [TeAcc:0.257]
 [Epoch:9] [TrLoss:0.00742] [TrAcc:0.999] [TeAcc:0.251]

+4%



MNIST→SVHN DAN

[Epoch:0] [TrLoss:0.86837] [MMD:2.63822] [TrAcc:0.721] [TeAcc:0.187]
 [Epoch:1] [TrLoss:0.21051] [MMD:2.41158] [TrAcc:0.936] [TeAcc:0.244]
 [Epoch:2] [TrLoss:0.13116] [MMD:2.41431] [TrAcc:0.958] [TeAcc:0.252]
 [Epoch:3] [TrLoss:0.09289] [MMD:2.41438] [TrAcc:0.971] [TeAcc:0.276]
 [Epoch:4] [TrLoss:0.07003] [MMD:2.41417] [TrAcc:0.979] [TeAcc:0.295]
 [Epoch:5] [TrLoss:0.02941] [MMD:2.37299] [TrAcc:0.993] [TeAcc:0.292]
 [Epoch:6] [TrLoss:0.01996] [MMD:2.38647] [TrAcc:0.994] [TeAcc:0.292]
 [Epoch:7] [TrLoss:0.01665] [MMD:2.39223] [TrAcc:0.995] [TeAcc:0.290]
 [Epoch:8] [TrLoss:0.01295] [MMD:2.38795] [TrAcc:0.997] [TeAcc:0.293]
 [Epoch:9] [TrLoss:0.01131] [MMD:2.39465] [TrAcc:0.997] [TeAcc:0.292]

MNIST→USPS No Adaptation

[Epoch:0] [TrLoss:0.83621] [TrAcc:0.732] [TeAcc:0.710]
 [Epoch:1] [TrLoss:0.19314] [TrAcc:0.942] [TeAcc:0.767]
 [Epoch:2] [TrLoss:0.11724] [TrAcc:0.962] [TeAcc:0.776]
 [Epoch:3] [TrLoss:0.09962] [TrAcc:0.969] [TeAcc:0.747]
 [Epoch:4] [TrLoss:0.05746] [TrAcc:0.981] [TeAcc:0.804]
 [Epoch:5] [TrLoss:0.02636] [TrAcc:0.990] [TeAcc:0.781]
 [Epoch:6] [TrLoss:0.01522] [TrAcc:0.996] [TeAcc:0.783]
 [Epoch:7] [TrLoss:0.01168] [TrAcc:0.997] [TeAcc:0.791]
 [Epoch:8] [TrLoss:0.00934] [TrAcc:0.998] [TeAcc:0.792]
 [Epoch:9] [TrLoss:0.00781] [TrAcc:0.999] [TeAcc:0.795]

+6%



MNIST→USPS DAN

[Epoch:0] [TrLoss:0.82244] [MMD:0.77220] [TrAcc:0.731] [TeAcc:0.831]
 [Epoch:1] [TrLoss:0.19474] [MMD:0.77927] [TrAcc:0.940] [TeAcc:0.832]
 [Epoch:2] [TrLoss:0.12379] [MMD:0.74995] [TrAcc:0.961] [TeAcc:0.822]
 [Epoch:3] [TrLoss:0.09187] [MMD:0.74013] [TrAcc:0.971] [TeAcc:0.823]
 [Epoch:4] [TrLoss:0.05910] [MMD:0.73806] [TrAcc:0.982] [TeAcc:0.850]
 [Epoch:5] [TrLoss:0.02590] [MMD:0.74846] [TrAcc:0.991] [TeAcc:0.853]
 [Epoch:6] [TrLoss:0.01578] [MMD:0.72747] [TrAcc:0.997] [TeAcc:0.869]
 [Epoch:7] [TrLoss:0.01379] [MMD:0.71482] [TrAcc:0.997] [TeAcc:0.855]
 [Epoch:8] [TrLoss:0.01140] [MMD:0.71238] [TrAcc:0.998] [TeAcc:0.864]
 [Epoch:9] [TrLoss:0.00984] [MMD:0.70754] [TrAcc:0.999] [TeAcc:0.856]

- SVHN和USPS上的识别准确率：左边是直接用MNIST的模型进行预测，右边是使用DAN进行迁移
- 基本上，新场景下使用迁移学习方法的性能比不使用的性能会有不定程度的提升，和任务相似程度有很大关系

案例分析-SDK说明

- NAIE平台官网: <https://console.huaweicloud.com/naie/>
- NAIE平台知乎官方账号: <https://www.zhihu.com/people/aigao-shi-qing>



The screenshot displays the iMaster NAIE console interface. At the top, there is a navigation bar with various tabs including '项目概览', '数据集', '特征工程', '模型训练', '模型管理', '模型验证', and a user profile section for 'dixinchunju'. Below the navigation bar, the main content area shows a project named 'transferdemo' with a user 'lixinchunju'. A '帮助中心' (Help Center) modal window is open, highlighting the 'SDK文档' (SDK Documentation) option. The background interface includes sections for '数据处理' (Data Processing) and '验证服务' (Validation Service), each with a circular progress indicator and a table of statistics.

状态	数量
运行中	0
成功	0
失败	0
正在训练	0
训练失败	0
训练准备	0

状态	数量
正在准备	0
正在训练	0
训练成功	0
训练失败	0
验证成功	0
验证失败	0

案例分析-SDK说明

- NAIE平台官网: <https://console.huaweicloud.com/naie/>
- NAIE平台知乎官方账号: <https://www.zhihu.com/people/aigao-shi-qing>



The screenshot shows the iMaster NAIE platform interface. The left sidebar contains a navigation menu with items like '总览', '本地算法迁移至平台', 'AutoML-自动机器学习', '分布式场景', '集成学习', '模型选择', '特征分析', '特征处理', '特征选择', '时序数据分析', '根因分析', and '迁移学习'. The '迁移学习' item is highlighted with a red arrow. The main content area displays the '迁移学习基础教程' (Transfer Learning Basic Tutorial) page, which includes an introduction to transfer learning, a list of transfer learning methods (EasyTransferability, MetaTrans), and a section on how to use the SDK.

迁移学习基础教程

近些年来, 机器学习包括深度学习的发展, 使模型的准确率越来越高。但是, 如果将已有的模型直接拿来处理同类型的问题, 结果却差强人意。迁移学习(Transfer Learning)提供了这一问题的解决方法。

关于迁移学习的介绍, 可以参考 <https://bbs.huaweicloud.com/forum/thread-21207-1-1.html>

NAIE平台提供了迁移学习SDK, 可以方便大家轻松使用多种常见的迁移学习方法, 解决不同任务之间迁移的问题。您只要将源域和目标域的数据准备好, 就可以轻松使用迁移学习SDK训练出一个更好的模型。

迁移学习SDK分为可迁移性判别方法、浅层迁移学习方法和深层迁移学习方法。集成了学术界和工业界常用的迁移学习能力, 并在可迁移性判别上引入了一套独有的判别方法。

迁移学习方法介绍

可迁移性判别方法

NAIE平台提供的可迁移性判别方法SDK, 可用于评估数据是否适用于迁移、哪种浅层迁移方法最有效。

- 迁移数据评估方法 (EasyTransferability)

评估数据是否适用于迁移。将源域和目标域的数据特征同时作为该方法的输入, 该方法会输出一个评估分数, 如果分数大于0.5, 说明数据适合迁移。

- 浅层迁移算法评估方法 (MetaTrans)

评估各浅层迁移学习方法迁移的效果。将源域的数据特征和标签、目标域的数据特征作为输入, 该方法会分别使用各浅层迁移学习方法SDK预置的历史Meta信息先训练出对应的SDK模型, 然后分别使用SDK模型来预测源域和目标域的数据迁移效果, 给出各SDK方法迁移的效果评估分数, 如果分数大于1, 说明使用对应的SDK来完成迁移学习是有效的, 最终会给出效果最好的浅层迁移学习方法。

研究难点讨论

- 如何判断某个迁移算法在某项任务上是否奏效？或者有何种方法来指导迁移算法的选择？
→ 如何度量可迁移性？



更多关于迁移学习

”



微信公众号和
官网



分享嘉宾知乎
主页



华为云HERO高校联盟知识峰会

Thanks !

