# 调研一

姓名：张婷旖

学号：2016211524

学院：计算机

目录

[调研一 1](#_Toc5128707)

[1主动学习Active Learning 2](#_Toc5128708)

[1.1概念 2](#_Toc5128709)

[1.2主要算法及特点 2](#_Toc5128710)

[1.3是否能以及如何应用于训练模型的演化 5](#_Toc5128711)

[2增量学习 5](#_Toc5128712)

[2.1概念 5](#_Toc5128713)

[2.2主要算法及特点 6](#_Toc5128714)

[2.3是否能以及如何应用于训练模型的演化 10](#_Toc5128715)

[3模型迁移Transfer Learning 11](#_Toc5128716)

[3.1概念 11](#_Toc5128717)

[3.2主要算法及特点 17](#_Toc5128718)

[3.3是否能以及如何应用于训练模型的演化 19](#_Toc5128719)

[4强化学习reinforcement learning 20](#_Toc5128720)

[4.1概念 20](#_Toc5128721)

[4.2主要算法及特点 22](#_Toc5128722)

[4.3是否能以及如何应用于训练模型的演化 26](#_Toc5128723)

[5从深度学习、宽度学习出发了解一下代码的可解释性 27](#_Toc5128724)

[5.1一般的： 27](#_Toc5128725)

[5.2深度学习角度： 28](#_Toc5128726)

[5.3宽度学习Broad Learning System角度： 29](#_Toc5128727)

## 1主动学习Active Learning

### 1.1概念

在传统的监督学习中，要求用于训练学习模型的数据均是已标记的。一般认为，已标记的数据越多，标记越精准，由这些数据训练得到的模型也更加准确和高效。但现实的大数据往往是质量低下且未标注的，而人工标记成本很高且效率极低，于是我们寻求用一种办法：

* 从大量无标注数据中**挑选**出部分最有价值的数据进行**标记**，再将这些数据加入训练集中去训练我们的模型，这就是主动学习。

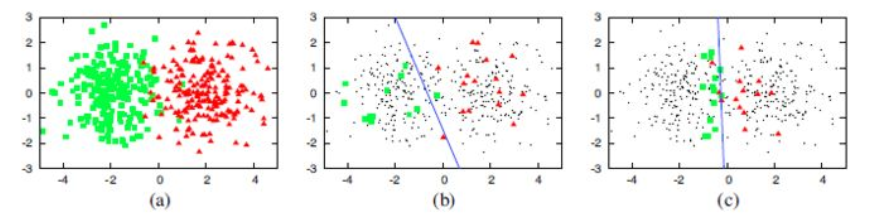
使用情况：

* 大量无标注数据
* 无任何可用的已标注数据来训练分类器
* 成本受限

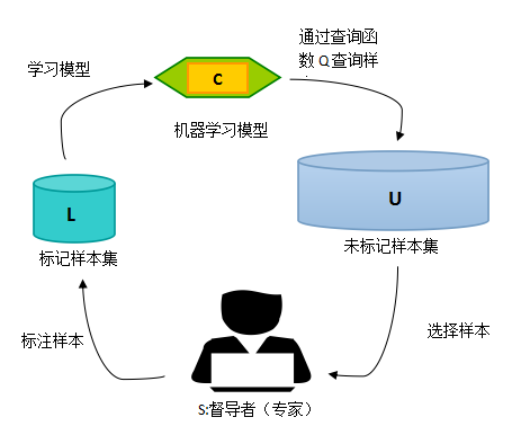
### 1.2主要算法及特点

主动学习是通过设计合理的查询函数（query function），从未标注的数据中挑出一部分后（一般挑选比较难分类的，“信息量”比较大的部分），加入训练集，重新训练模型，反复迭代

举个例子：

* 下图a代表数据分真实分布情况；b表示用随机抽样方法得到样本，然后人工打标，最后用打标后的数据训练出的分割线；c代表用主动学习算法得到的样本，然后打标，最后用打标后的数据训练出的分割线。显然主动学习很好的关注了两个类之间的边界
* 

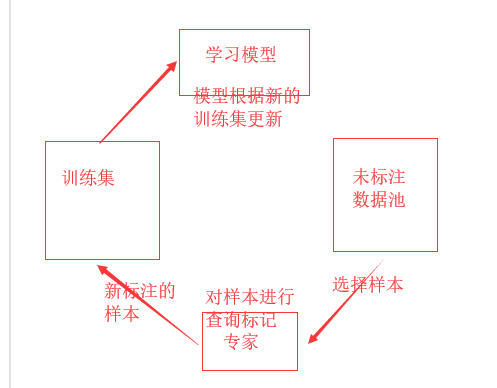
主动学习的模型：

* A=(C,Q,S,L,U)
* C 为一组或者一个分类器，L是用于训练已标注的样本。Q 是查询函数，用于从未标注样本池U中查询信息量大的信息，S是督导者，可以为U中样本标注正确的标签。
* 学习者通过少量初始标记样本L开始学习，通过一定的查询函数Q选择出一个或一批最有用的样本，并向督导者询问标签，然后利用获得的新知识来训练分类器和进行下一轮查询。不断循环，直至达到某一停止准则为止。
* 

#### 1.2.1基于数据池的

最常见，假设所有未标记数据已经给定，形成一个数据池，每次从中选择数据即可

主动学习算法迭代进行，每一次从未标记数据池中选择样本向专家查询标记，并将这些新标注的样本加入训练集，模型基于新的训练集进行更新，进而进入下一次迭代



##### 查询策略（样本选择策略）

主动学习的关键任务，按照一定的准则来选择被查询的样本。

* 基于信息量的（数据的不确定性）
  + 选择不确定性最高、信息熵最大的样本进行查询
  + 最常见
  + 只基于现有的已标记样本，忽略了大量的未标记样本中蕴含的数据分布信息，可能导致采样偏差问题
* 基于代表性的
  + 倾向于选择那些更能刻画数据整体分布的未标记数据进行标记查询(各个样本提供的信息不重复不冗余)
  + 往往通过聚类或密度估计等无监督技术来评估样本的代表性，由于忽略了已标记样本，因此整体性能也可能会依赖于聚类结果的好坏
* 基于差异性的
  + 所查询的样本提供的信息是全面的，各个样本提供的信息不重复不冗余，即样本之间具有一定的差异性。
* 综合多种准则的
  + 同时考虑选择样本的信息量和代表性
  + 能够有效避免采样偏差和依赖聚类结果的问题

#### 1.2.2基于数据流的

假设样本以流的形式一个一个到达，因此在某时刻当一个样本到达的时候，算法必须决定是否查询该样本的标记，不则丢弃

eg:数据流源源不断产生，而又无法保存下来所有数据时

#### 1.2.3基于合成样本查询的

不是从已有样本中选择来查询标记信息，而是直接从特征空间里合成出新的样本进行查询。新合成的样本可能是特征空间里任意取值组合产生的。

因此在某些应用问题中可能导致人类专家也无法标注这些合成样本。eg:在图像分类任务中，任意像素取值合成的一幅图片可能并不能呈现出清晰的语义。

### 1.3是否能以及如何应用于训练模型的演化

要提高机器学习模型性能，可从五个方面入手：

1. 数据预处理
2. 特征工程
3. 机器学习算法
4. 模型集成与融合
5. 数据增强

个人认为主动学习对训练模型的改进主要体现在数据预处理方面

* 获取数据、数据抽样（主动学习的关键任务，按照一定的准则来选择被查询的样本，如基于信息量准则、基于代表性准则等，选择样本时不仅要考虑其价值，还要考虑代价）
* 数据探索：质量分析（噪音，脏数据……）；特征分析
* 数据预处理与清洗：数据清理；集成；变换；归约（数据降噪后再进行选择和标记）

一些例子：

1. 多标记学习任务中，一个样本可以同时具有多个标记，这时查询方式（即以何种方式查询所选样本的监督信息）对主动学习性能非常关键。【多标记任务中查询策略的选定】
2. 在一些任务中，提供标记信息的不再是一个专家，而是一群可能提供错误信息的用户，这时如何从带有噪音的数据中获取正确的标记信息变得非常重要。【数据降噪后再进行选择和标记】
3. 还有一些任务中，可能标注每个样本的代价不一样，这使得主动学习算法在选择样本的时候不仅要考虑样本可能带来的价值，还要考虑标注它可能花费的代价。【选择样本时不仅要考虑其价值，还要考虑代价】

## 2增量学习

### 2.1概念

一个学习系统能不断地从新样本中学习新的知识，并能保存大部分以前已经学习到的知识。非常类似于人类自身的学习模式。因为人在成长过程中，每天学习和接收新的事物，学习是逐步进行的，而且，对已经学习到的知识，人类一般是不会遗忘的。

对于增量学习，每当新增数据时，并不需要重建所有的知识库，而是在原有知识库的基础上，仅对由于新增数据所引起的变化进行更新。

批量学习的局限性：

* 对于大多数机器学习算法，我们都归纳成**批量学习**（batch learning）模式，即假设在训练之前，所有训练样本一次都可以得到，学习这些样本之后,学习过程就终止了,不再学习新的知识。
* 然而在实际应用中,训练样本通常不可能一次全部得到,而是随着时间逐步得到的,并且样本反映的信息也可能随着时间产生了变化。如果新样本到达后要重新学习全部数据,需要消耗大量时间和空间,因此批量学习的算法不能满足这种需求。

增量学习的优点：

* 增量学习算法可以渐进的进行知识更新,且能修正和加强以前的知识,使得更新后的知识能适应新到达的数据,而不必重新对全部数据进行学习。增量学习降低了对时间和空间的需求,更能满足实际要求。
* 一方面由于其无需保存历史数据，从而减少存储空间的占用；另一方面增量学习在当前的样本训练中充分利用了历史的训练结果，从而显著地减少了后续训练的时间
* 特点：

1)可以从新数据中学习新知识;

2)以前已经处理过的数据不需要重复处理;

3)每次只有一个训练观测样本被看到和学习;

4)学习新知识的同时能保存以前学习到的大部分知识;

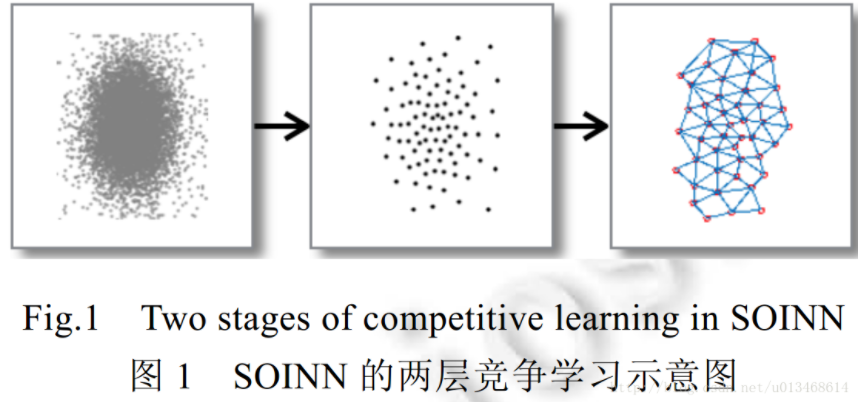
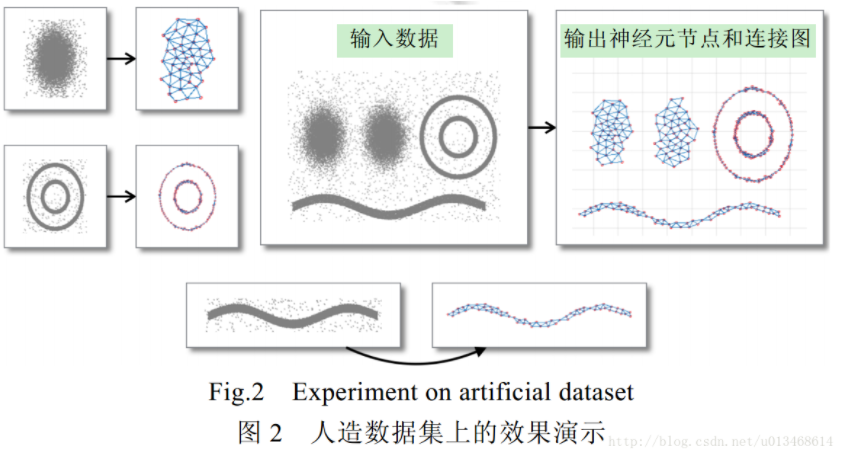
5)—旦学习完成后训练观测样本被丢弃;

6)学习系统没有关于整个训练样本的先验知识;

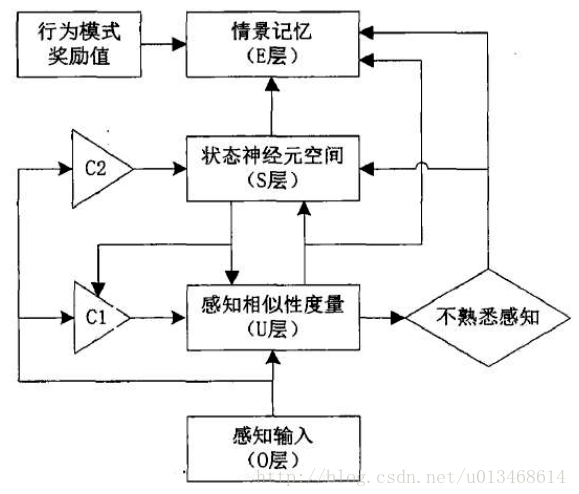
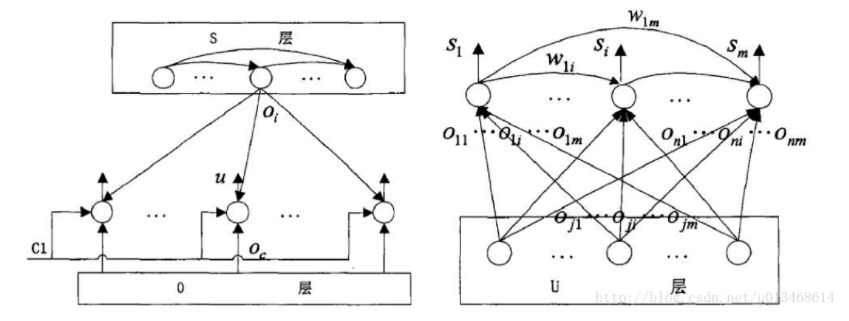
### 2.2主要算法及特点

常见的增量式学习框架：

#### 2.2.1自组织增量学习神经网络SOINN

* 一种基于竞争学习的两层神经网络（不包括输入层）
  + 第1层网络接受原始数据的输入，以在线的方式自适应地生成原型神经元来表示输入数据。这些节点和它们之间的连接反映了原始数据的分布情况；
  + 第2层根据第1层网络的结果估计出原始数据的类间距离与类内距离，并以此作为参数，把第1层生成的神经元作为输入再运行一次SOINN算法，以稳定学习结果。
  + 
  + 
* 关键是**动态调整**（<u>调整神经元的权值向量</u>和<u>网络的拓扑结构</u>以优化对输入数据的表达精度。）
* <u>适时增加神经元</u>自适应地确定神经元的数量以**满足一定的量化误差约束**，在不影响之前学习结果的情况下**适应之前没有学习过的输入模式**。（两个目的）
  + 类内节点插入（为了自适应地减小神经元的量化误差,尽可能准确地近似原始数据的分布。SOINN 在运行过程中会记录每个神经元的累积量化误差,每学习一段固定的时间之后,找出所有节点中累积量化误差最大的两个节点,然后在它们的中间插入一个新的节点,以插值的方式更新它们的累计量化误差值.考虑到并非每次插入操作都是有必要的,如果不进行一些限制的话,那么随着算法的进行,节点的数量会不断地增加.因此,SOINN 在每次类内的节点插入操作后都会再判断该次插入操作是否显著降低了量化误差:如果没有,则取消本次插入操作。）
  + 类间节点插入（发生在新输入的数据与之前学习过的数据差异性较大的时候。SOINN 通过为每一个神经元i设置一个相似度阈值(similarity threshold)参数Ti来判断新来的数据样本是否有可能属于一个新的类别，如果该数据点与之前学习得到神经元差异性较大,就在该数据点的位置上生成一个新的节点来代表这个可能的模式）
  + 类间节点插入是SOINN 实现增量学习的关键,节点插入的时机对于最终的结果有较大影响；每个节点的相似度阈值参数T又是决定插入操作的关键.
  + 如果T值过小,则每个数据都会被认为是一个新的模式而生成一个节点
  + T值过大,则会导致节点个数过少,此时量化误差增大,而且不能准确反映数据的分布
  + 理想情况下,该参数应大于平均的类内距离同时小于平均的类间距离
  + SOINN在这个问题上采用了一种自适应的方式不断更新Ti 的值，使得能够适应不断变化的输入模式。
* 可以用于各类非监督学习中
* SOINN的评判阈值是自动动态调整的

#### 2.2.2情景记忆马尔可夫决策过程EM-MDP

* EM-MDP是一套完整的人工智能方案（简化版）。这个框架中包括对情景的认知、增量学习、短期与长期记忆模型，其中增量学习是重点。
* 
* 框架中U层与S层都具有增量式学习的能力，U层与S层的结构：
* 
* U层节点个数等于输入感知的维数，每个节点的输出由3个信号采用“多数表决2/3”原则共同确定：（1）当前环境的感知输入Oc ;（2）控制信号C1；（3）由S层反馈的获胜状态神经元的映射感知。
* 情景网络接受来自环境的感知输入，通过检查当前感知输入与所有存储感知向量之间的匹配程度，确定新感知及其相关事件是否已存在机器人的情景记忆当中。按照预先设定的激活阈值来考察相似性度量，决定对新输入的感知采取何种处理方式。在网络每一次接受新的感知输入时，都需要经过一次匹配过程。相似性度量存在两种情况：
  + 相似度超过设定阈值：选择该邻近状态集为当前输入感知的代表状态神经元集合。**实际上是对情景记忆中的映射感知进行重新编码，以稳定已经被学习了的熟悉事件。**
  + 相似度不超过设定阈值：需要在S层新增一个代表新输入感知的状态神经元并存储当前感知为该新增状态的映射感知，以便参加之后的匹配过程。同时建立与该状态神经元相连的权值，以存储该类感知和参与以后的匹配过程。**实际上是对不熟悉事件建立新的表达编码。**
* EM-MDP中的阈值是预先设定的一个定值

#### 2.2.3两种主要算法的区别

* SOINN被应用的相对较多
* M-MDP结还未对其中增量式学习部分进行有效的框架分离，所以结构相对作为单独框架提出来的SOINN不是很清晰
* 将一个输判定是否为新知识，需要度量新输入与旧知识之间的“距离”，SOINN的评判阈值是自动动态调整的，而EM-MDP中的阈值是预先设定的一个定值

\*现有的增量学习算法大多采用决策树和神经网络算法实现的，缺点:

* 一方面由于缺乏对整个样本集期望风险的控制，算法易于对训练数据产生overfitting；
* 另一方面,由于缺乏对训练数据有选择的遗忘淘汰机制,在很大程度上影响了分类精度。
* 改进：
  + 增量学习存储的所有数据允许重新训练

### 2.3是否能以及如何应用于训练模型的演化

可以用于训练模型的演化，因为在实际运用中，数据往往不是一开始就固定下来放在数据池中的，可能是流水形式到来甚至无法保存的，或者是不断有新数据加入到原始数据池中的，模型对不同的数据模拟训练结果往往不同，但要对所有数据都进行重新训练，显然是开销巨大的，有的情境下甚至无法重新训练（有时数据无法保存），增量模型便是为了解决这种情况出现的。

增量模型只对新到来的数据进行重新训练，根据新数据和旧数据之间的“距离”去判断其属于一个旧“类”还是另起一个新“类”。占用更少的内存，花费更少的时间，实现对模型的不断修正

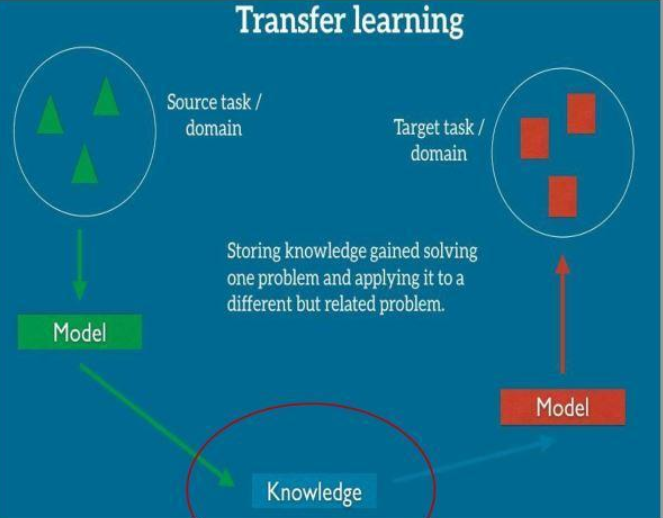
## 3模型迁移Transfer Learning

### 3.1概念

#### 3.1.1迁移学习：

从以前的任务当中去学习知识（knowledge）或经验，并应用于新的任务当中。换句话说，迁移学习目的是从一个或多个源任务（source tasks）中抽取知识、经验，然后应用于一个不同但是相关的目标领域（target domain）当中去。（比如说，人类在学会骑自行车后，再骑摩托车就很容易了，这其实是一种适应的能力）

**迁移学习的难点在于找出不变量**

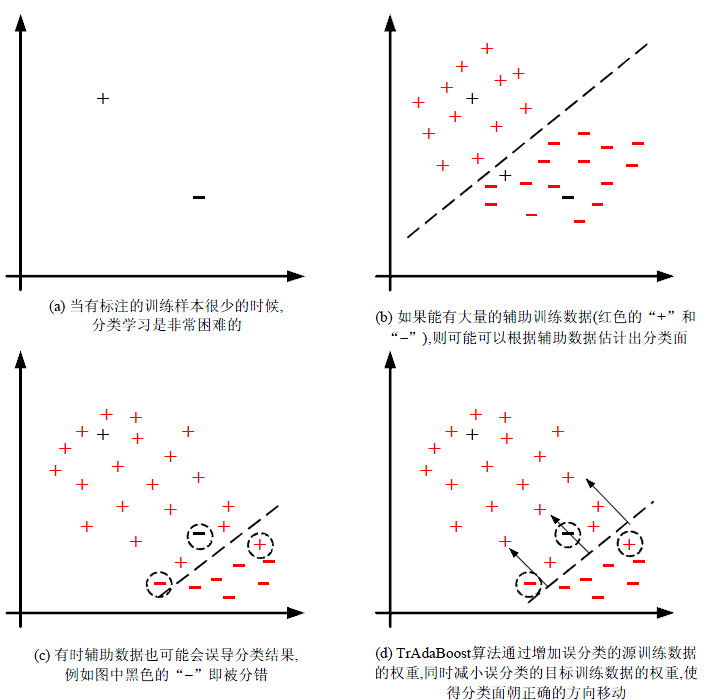
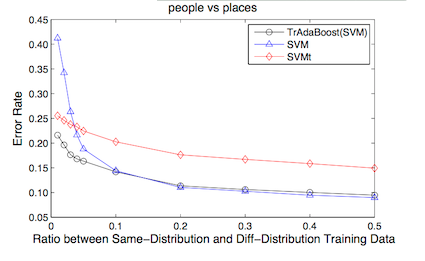


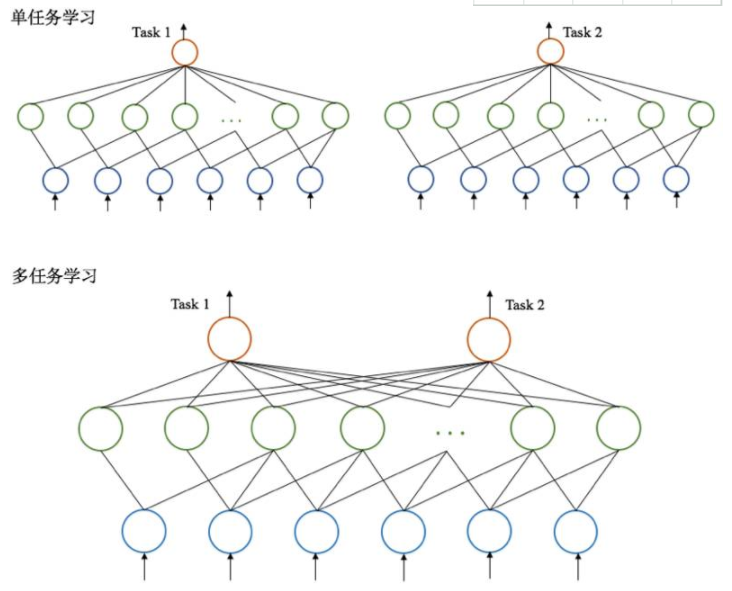
普通的监督学习需要海量数据的支撑，而人工智能的发展越来越趋向于不要求海量数据的也能达到应用的精准要求，因此，小数据学习正在成为新的热点（比如，迁移学习）



迁移学习适合有标签的应用域,分类和回归问题是比较适合做迁移学习的场景

#### 3.1.2迁移学习的四类算法：

* 基于样本的迁移学习
  + 从源领域中，选取对目标领域建模有用的样本，和目标领域的样本一起使用，来实现迁移学习的效果。
    - 通过提升（boosting）算法，对样本设置不同权重；
    - 通过一定的过滤规则，只选取和目标领域相近的样本。
  + 典型方法：
    - TrAdaBoost：
      * 从源 Domain 数据中 筛选有效数据，过滤掉与目标 Domain 不match的数据，通过 Boosting方法建立一种权重调整机制，增加有效数据权重，降低无效数据权重（从过期数据里面找出和目标数据最接近的样本数据）
      * 
      * 权重的更新方式：
        + 对于辅助样本来讲，预测值和标签越接近，权重越大；
        + 而对于目标数据则是相反，预测值和标签差异越大，权重越大
        + 原因：我们想找到辅助样本中 和 目标数据分布最接近的样本，同时放大目标样本Loss的影响
      * 
      * 实验发现，当 同分布数据（目标数据）占比当低于0.1时，算法效果明显，当比例超过 0.1时，TrBoost 退化为 SVM 的效果。
        + 我们认为大于0.1时，仅仅依靠 目前数据就足够完成样本训练，这种情况下，辅助样本的贡献可以忽略。
        + 另外，当 目标数据 和 辅助数据 差别比较大时，该方法是不 Work的
    - 从 PU-Learning 中借鉴而来，记为 SPY。
      * 为解决 PU-learning 中只有正样本和无标签样本，而没有负样本问题，在 ICML 2002 上提出的一种从无标签样本中寻找最可能的负例，以便有监督学习的技术。
      * 流程：样本重标记：将 Ds 和 Dt 对应的数据重新标记，即 Ds 中的样本标记为负样本，Dt 中的样本标记为正样本；数据集切分：将重新标记后的 Ds 和 Dt 放在一起，并随机切分为训练集、测试集；分类器训练：选择一个可以输出概率预测结果的基分类器，并在训练集上训练该分类器；概率预测：将训练好的模型在测试集上预测，输出测试集中每条样本属于正样本的概率；样本筛选：将测试集中样本按概率降序排序，设置阈值 k，记不小于第 k 个正样本对应的概率为 p，选择 Ds 中概率不小于 p 的样本，放入 Dt 中。
* 基于特征的迁移学习
  + 利用源领域 Ds（和目标领域 Dt）的信息，寻找合适的特征表示空间，作为所要迁移知识的载体，来增强目标领域的学习效果。关键在于如何找到合适的特征表示空间。
  + 典型方法：Self-taught learning，multi-task structure learning
* 基于参数的迁移学习
  + 通过在不同领域间共享参数，来实现迁移学习的效果
  + 典型方法：
    - Learning to learn
    - Regularized multi-task learning多任务学习，使得不同的学习任务能相互促进。一般是通过共享特征来实现共享参数的功能。



* 基于相关性的迁移学习
  + 建立源Domain和目标Domain之间的相关知识映射
  + 典型方法：Mapping 方法

优势：

* 适应小数据：迁移学习能够将大数据所训练的学习器迁移到只有小数据的领域。
* 提升可靠性：迁移学习所训练的模型具有适应性，可以迁移到多个领域而不产生显著的性能下降。
* 满足个性化：其实也是适应性的体现。

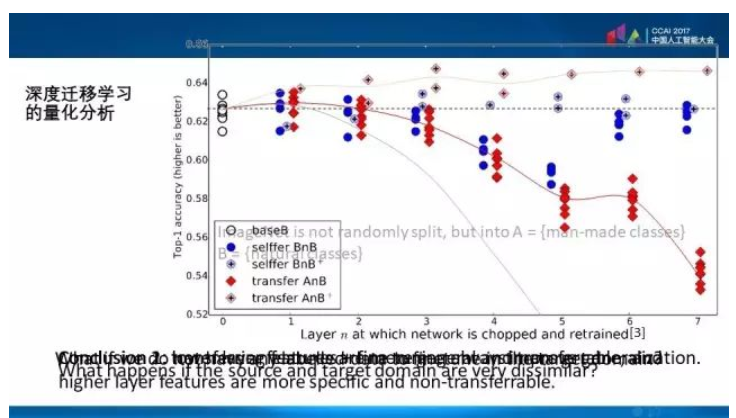
#### 3.1.3模型迁移

模型迁移就是迁移学习的一种：

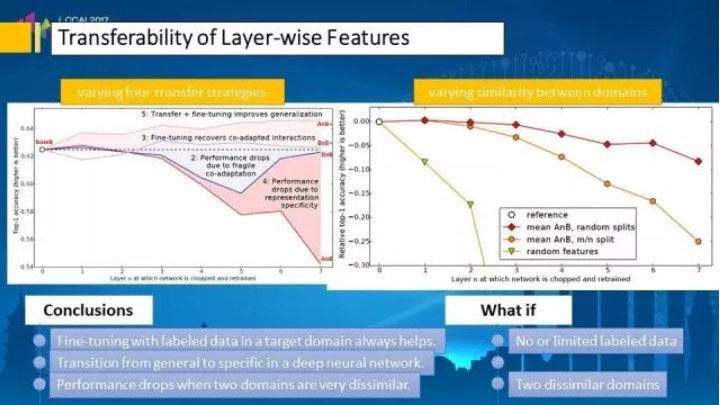
* eg：模型迁移利用上千万的图象训练一个图象识别系统，当我们遇到一个新的图象领域，就不用再去找几千万个图象来训练了，可以原来的图像识别系统迁移到新的领域，所以在新的领域只用几万张图片同样能够获取相同的效果。
* 模型迁移的一个好处是可以和深度学习结合起来，我们可以区分不同层次可迁移的度，相似度比较高的那些层次他们被迁移的可能性就大一些。



* 对神经网络来说，从一个领域迁移到另外一个领域，在下层比较容易迁移，在上层就不容易。



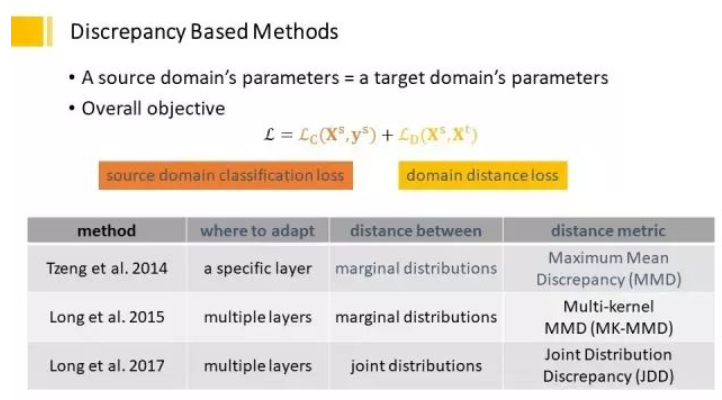
我们可以看到代表迁移能力的红色是逐层下降的。我们在当中也可以做各种各样的演算和变化，比如说我们可以让迁移过来的参数，让它在之后的领域里面再得到重新的训练，这样就会得到上面的这条线，我们这个线放得越宽迁移能力就会变得越强，如果两个领域之间相差太大了，那么这个迁移是没办法进行的，就像下面的这条蓝线直线下降，这个能力就不行。



* 关于模型迁移，我们关心两点。
  + 不同模型里面的一个层次，它的迁移量化
    - 我们在两个领域完全都没有标注时，可以利用刚才的知识把下层固定住，直接迁移。在比较靠上层就可以放开，利用的方法是可以来减少两个领域之间的距离，这个距离虽然我们没有标注，我们还是可以把它衡量出来的
  + 两个领域的距离有没有一个很好的衡量
* 这里我们引入预训练模型(pre-trained model)，在解决问题的时候，不用从零开始训练一个新模型，可以从在类似问题中训练过的模型入手。使用之前在大数据集上经过训练的预训练模型，我们可以直接使用相应的结构和权重，将它们应用到我们正在面对的问题上。
* Keras库中有许多训练得不错的预训练模型，在进行模型迁移学习时，我们对这些模型进行微调即可

### 3.2主要算法及特点

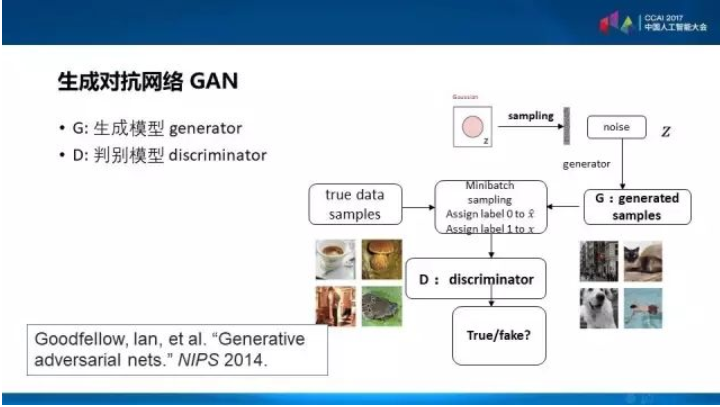
#### 3.2.1基于差异的方法

* 
* 在哪里做迁移的转换、自适应。可以在某个层次来做，也可以在多个层次来做，如何来衡量两个领域之间的距离，另外还可以将源领域的信息加进来，建立一个更准确的Loss。
* eg：把自然语言的深度处理分成三个层次，即E、H、O，如果我们锁定不变，而其他的层次有可能变化，也有可能不变化，这样可以发现，先迁移后微调的效果最好。
* 在目标领域我们能够让它的再生后期做自适应的学习，就是开锁的这种，效果是最好的

#### 3.2.2传递式迁移学习

* 过去的迁移学习都是给一个源领域，一个目标领域，从源到迁移目标，现在我们可以多步迁移，从A到B，然后从B到C
* 传递式的迁移学习也可以利用深度学习的分层这个概念进行。我们右边这里有三类数据，有一个是源领域的数据，比如图像，有一个是目标领域的数据，比如说文字，中间有很多文字和图像混杂在一起的领域。我们一个做法就是在中间领域挑一些样本，让它更好地帮助源领域迁移到目标领域

#### 3.2.3生成对抗网络GAN

* 
* 它的思想来自两个模型交互，一个是生成模型，可以生成样板，就是右边的G模块，它一开始是非常粗糙的。还有是D模型，比较生成和真实的样本，它可以告诉你有没有被发现，你生成模型生成的样本够不够好，经过他们两者之间的博弈，最后就产生了学习的跃进，使得G和D两者都得到了改进。
* Cycle GAN的思想也非常有趣，一个源领域可以迁移到一个目标领域的话，我们同样也应该让一个目标领域的数据、知识迁移到源领域。如果是回来的话，如果还能返回原来的样本，那么就是说明迁移已经学得靠谱了。

#### 3.2.4对预训练模型进行微调的方法

##### 特征提取

* 将预训练模型当做特征提取装置来使用
* 具体的做法是，将输出层去掉，然后将剩下的整个网络当做一个固定的特征提取机，从而应用到新的数据集中。

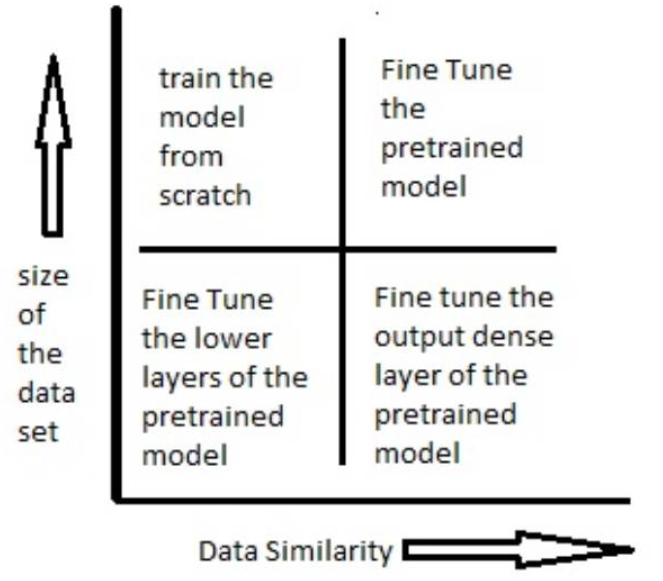
##### 采用预训练模型的结构

* 将预训练模型所有的权重随机化，然后依据自己的数据集进行训练

##### 训练特定层，冻结其他层

* 对预训练模型进行部分训练，将模型起始的一些层的权重保持不变，重新训练后面的层，得到新的权重。在这个过程中，我们可以多次进行尝试，从而能够依据结果找到frozen layers和retrain layers之间的最佳搭配。

##### 在各种情况下应该如何使用预训练模型：

* 
* 数据集小，数据相似度高：不需要重新训练模型。我们只需要将输出层改制成符合问题情境下的结构。使用预处理模型作为模式提取器。
* 数据集小，数据相似度不高：冻结预训练模型中的前k个层中的权重，然后重新训练后面的n-k个层，当然最后一层也需要根据相应的输出格式来进行修改。数据的相似度不高，重新训练的过程就变得非常关键。而新数据集大小的不足，则是通过冻结预训练模型的前k层进行弥补。
* 数据集大，数据相似度不高：采用预训练模型将不会是一种高效的方式。最好的方法还是将预处理模型中的权重全都初始化后在新数据集的基础上重头开始训练
* 数据集大，数据相似度高：最理想的情况，采用预训练模型会变得非常高效。保持模型原有的结构和初始权重不变，随后在新数据集的基础上重新训练。

### 3.3是否能以及如何应用于训练模型的演化

可以用于训练模型的演化

将已经训练好的系统模型做一些微调，迁移运用到相关的新领域中去，这样一来，新模型的训练会更加高效，也节省了大量的开销，对预先保有的模型进行：特征提取、直接使用、分层训练等操作，实际上是进行一些参数上的和输入输出维度的调整。特别是现有的许多图像分类的预模型已经已经训练得非常好，只需加入少量个性化特征，简单调整某些参数，就可以很快训练出自己需要的图片分类模型，准确度也很高

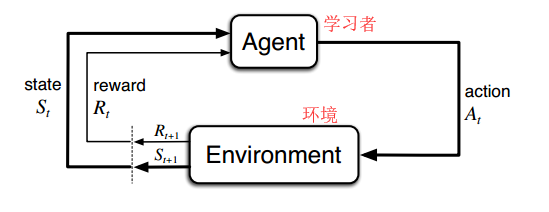
## 4强化学习reinforcement learning

### 4.1概念

机器学习可以分为三类：

* 监督学习supervised learning
* 非监督学习unsupervised learning
* 强化学习reinforcement learning

强化学习：

* 例子：
  + 当还是一个调皮的孩子不愿意做作业，父母就会在孩子不愿意做作业的时候就会说：“做完作业带你去麦当劳”。这时候，小孩子眼睛闪着金光，于是调皮的孩子就会为了去麦当劳乖乖地去写作业，久而久之，就会明白只有努力写作业才能获得去麦当劳的奖励。
  + 然而事情不是总这么简单，父母对于作业完成的顺序可能会有要求，假如父母特别希望看到孩子先做完数学，然后再做语文、英语作业，如果按照父母的意愿先做完数学再做其他作业，那么就不仅能吃上炸鸡，还可以加一个雪糕。于是小孩就会学聪明点，为了吃到更多麦当劳食品，就会按照父母的意愿去先完成数学作业，再做其他作业。最后小孩不仅知道努力做作业可以获得奖励，并且为了吃到更多的麦当劳食品，改变做作业的顺序，这就相当于找到一个好的策略，能够使小孩获得最大累积奖励。
  + 在上面的这个例子中，调皮的孩子就是智能体，父母代表环境，麦当劳的炸鸡和雪糕分别代表不同的奖励信号，小孩选择不做作业、做作业、做作业的顺序就是动作，当前作业的完成情况可以类比为状态。父母（环境）会根据孩子的作业的完成情况（当前状态）给予不同的奖励，对于不同奖励我们会采取不同的方式去做作业（选择动作），做作业并且先做数学作业就是最优策略
  + 强化学习就是不断地根据环境的反馈信息进行试错学习，进而调整优化自身的状态信息，其目的是为了找到最优策略、或者找到最大奖励的过程。
* 
* 重点：在已有当前模型的情况下，如何选择下一步的行动才对完善当前的模型最有利
  + 探索（exploration）：选择之前未执行过的actions，从而探索更多的可能性
  + 开发（exploitation）：选择已执行过的actions，从而对已知的actions的模型进行完善。
* 强化学习最重要的3个特点是：
  + 基本是以一种闭环的形式；
  + 不会直接指示选择哪种行动（actions）；
  + 一系列的actions和奖励信号（reward signals）都会影响之后较长的时间。
  + 反馈信号有时间延迟；
* 组成：智能体（Agent）、环境（Environment）、状态（State）、动作（Action）、奖励（Reward）
* 过程：
  + 智能体执行了某个动作后，环境将会转换到一个新的状态
  + 对于该新的状态环境会给出奖励信号（正奖励或者负奖励）。
  + 随后，智能体根据新的状态和环境反馈的奖励，按照一定的策略执行新的动作。
  + 智能体通过强化学习，可以知道自己在什么状态下，应该采取什么样的动作使得自身获得最大奖励

### 4.2主要算法及特点

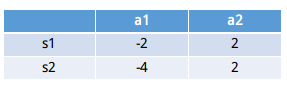
#### 4.2.1基于值的：

根据自己认为的高价值选择行为

在所有动作中计算值，然后选择值最高的那个行为。

##### Q-learning的算法

例子：

* 假设我们现在处于状态s1（在写作业），有俩个行为，分别是a1（看电视），a2（写作业）。根据经验，在这种状态下，选择a2（写作业）的值（value，可理解为分数）要比选择a1（看电视）要高。这里的state（状态）、action（动作）、值（value）都可以用一个有关s、a和value的Q表格代替。
* 这个例子中，在状态s1下选择a1的值为 Q(s1,a1)=-2, 选择a2的值为 Q(s2,a2)=2, 所以我们选择a2这个动作然后到达状态s2。重复上面的过程，我们可以得到Q(s2,a1), Q(s2,a2) 的值（value），并比较他们的大小，从而确定下一个action（动作）达到状态 s3，并不断的重复这个过程，这就是Q表，如下图所示。
* 

算法思想：

* QLearning是强化学习算法中值迭代的算法，Q即为Q（s,a）就是在某一时刻的 s 状态下(s∈S)，采取 a (a∈A)动作能够获得收益的期望，环境会根据agent的动作反馈相应的回报reward r，所以算法的主要思想就是将State与Action构建成一张Q-table来存储Q值，然后根据Q值来选取动作获得较大的收益

可以将问题抽象成一个马尔科夫决策过程，每个格子都算是一个状态，目的是找到一条能够到达终点获得最大奖赏的策略

Q-learning的训练公式：

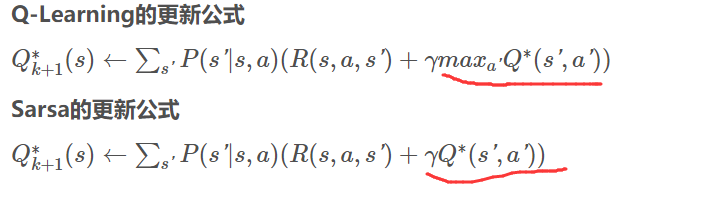
* 
* α为学习速率（learning rate），γ为折扣因子（discount factor）
* 学习速率α越大，保留之前训练的效果就越少。折扣因子γ越大，所起到的作用就越大。
* γ越大，小鸟就会越重视以往经验，越小，小鸟只重视眼前利益（R）

##### Sarsa

Sarsa算法的的决策部分与Qlearning相同，都是通过Q表的形式进行决策，在 Q 表中挑选值较大的动作值施加在环境中来换取奖惩

两者不同的是行为更新准则是有差异的。Sarsa不会去选取他估计出来的最大Q估计值，而是<u>直接选取</u>估计出来的Q值。

* Sarsa 是 on-policy型, 在线学习, 下一个 state*和action* 将会变成他真正采取的 action 和 state 。Sarsa 是一种保守的算法, 他在乎每一步决策, 以及下一步需要的stage和action。如果你比较在乎机器的损害, 用Sarsa 算法, 在训练时就能减少损坏的次数
* Q learning 是Off-policy型,离线学习.state*和 action* 在算法更新的时候都还是不确定的。Qlearning保证了一个stage下一次的value最大化



##### Sarsa- lambda

Sarsa 的一种提速方法

Sarsa 是一种单步更新法, 也就是 Sarsa(0), 他等走完这一步以后直接更新行为准则. 如果延续这种想法, 走完这步, 再走一步, 然后再更新, 我们可以叫他 Sarsa(1)。这回合我们走了 n 步, 那我们就叫 Sarsa(n). 为了统一这样的流程, 我们就有了一个 lambda 值来代替我们想要选择的步数。 这也就是 Sarsa(λ)的由来

Sarsa 和 Qlearning 都是每次获取到奖励reward后只更新获取到 reward 的前一步，那么Sarsa(λ)就是更新获取到 reward 的前 λ 步. λ 在 [0, 1] 之间取值,

* 当 lambda = 0, Sarsa-lambda 就是 Sarsa单步更新, 只更新获取到 reward 前经历的最后一步。
* 如果 lambda = 1, Sarsa-lambda就变成了回合更新，更新的是获取到 reward 前所有经历的步，对所有步更新的力度都是一样.
* 当 lambda 在 0 和 1 之间, 取值越大, 获得奖励大的步更新力度越大

##### DQN (Deep Q Network)

在Q-Learning中，我们提到了用Q表来存储当前状态s1下采取的动作action的值（value，在Q表中也称为Q值）。但是在实际过程中，s2可能有很多不同的情况，这将会导致Q表存储的值会很多，占内存，且十分耗时的

通常做法是把Q-Table的更新问题变成一个函数拟合问题，相近的状态得到相近的输出动作。如下式，通过更新参数 θ使Q函数逼近最优Q值：



DRL是将深度学习（DL）与强化学习（RL）结合，直接从高维原始数据学习控制策略。而DQN是DRL的其中一种算法，它要做的就是将卷积神经网络（CNN）和Q-Learning结合起来，CNN的输入是原始图像数据（作为状态State），输出则是每个动作Action对应的价值评估Value Function（Q值）。

DL与RL结合的问题：

1. DL需要大量带标签的样本进行监督学习；RL只有reward返回值，而且伴随着噪声，延迟（过了几十毫秒才返回），稀疏（很多State的reward是0）等问题；
2. DL通常假设数据样本独立同分布；RL的训练样本的状态是一个序列，而且通常状态之间高度相关。
3. DL目标分布固定；RL的分布一直变化，比如你玩一个游戏，一个关卡和下一个关卡的状态分布是不同的，所以训练好了前一个关卡，下一个关卡又要重新训练；
4. 过往的研究表明，使用非线性网络表示值函数时出现不稳定等问题。

DQN解决问题方法：

* 通过Q-Learning使用reward来构造标签（对应问题1）
* 通过experience replay（经验池）的方法来解决相关性及非静态分布问题（对应问题2、3）
  + 使用ξ-greedy来选择动作，把每个时间步agent与环境交互得到的转移样本 (st,at,rt,st+1)储存到回放记忆单元，要训练时就随机拿出一些（minibatch）来训练（其实就是将游戏的过程打成碎片存储，训练时随机抽取就避免了相关性问题），通过Q-learning对模型进行参数更新。
  + 由于处理变长数据对神经网络来说有困难，所以DQN使用固定长度的历史数据来表征状态，具体来说，就是过去三帧或四帧图像的融合。
  + 这是<u>DQN的最关键技术——经验回放</u>：在经验回放中随机均匀采样，打破了训练样本之间的相关性；同时，采用过去的多个样本做平均，也平滑了训练样本分布，减缓了样本分布变化的问题。
* 使用一个CNN（MainNet）产生当前Q值，使用另外一个CNN（Target）产生Target Q值（对应问题4）

优点：

1. 算法通用性；
2. End-to-End 训练方式；
3. 可生产大量样本供监督学习。
4. 每一步的数据都可以被多次采样，极大的提高了数据效率。
5. 直接从连续的数据里学习效率低下，因为前后的样本之间高度相关。而随机采样打破了这些相关性，因此降低了参数更新的方差
6. 通过经验回放，数据分布会在过去多个状态下被平均，从而平滑了训练过程，避免了训练发散。由于使用了经验回放，算法必然是off-policy的，因为产生数据样本（动作选择）的网络参数与当前被训练的网络参数是不同的，也就是生成动作的policy与算法学习的policy是不同的。 实际操作中，经验回放仅存储最近的N个样本点，当参数更新时均匀的从这些样本点中采样。

缺点：

1. 无法应用于连续动作控制；
2. 只能处理只需短时记忆问题，无法处理需长时记忆问题，经验池的样本也会不断的被最近的数据所覆盖掉。
3. CNN不一定收敛，需精良调参。
4. 经验池不能区分哪些样本点更重要
5. 均匀采样也赋予了每个样本同样的重要性，更聪明的采样策略应该是对那些能够学到更多的样本点赋予更多的权重

#### 4.2.2直接输出的：

不通过分析奖励值，直接输出行为的方法

能够在一个连续区间内挑选动作

##### Policy Gradients

使用神经网络，直接根据状态输出动作或者动作的概率

如果一个动作得到的reward多，那么我们就使其出现的概率增加，如果一个动作得到的reward少，我们就使其出现的概率减小。根据这个思想，我们构造如下的损失函数：loss= -log(p)\*vt

* log(p)表示在状态s对所选动作a的吃惊度, 概率越小, -log(p) 越大. vt代表的是当前状态s下采取动作a所能得到的奖励。如果在p很小的情况下, 得到了一个大的Reward, 也就是大的vt, 那么-log(p)\*vt就更大, 表示更吃惊, (我选了一个不常选的动作, 却发现原来它能得到了一个好的 reward, 那我就得对我这次的参数进行一个大幅修改)。

##### Actor-Critic

基本思想和Policy Grandients相同，但是Actor Critic可以进行单步更新/实时更新，比传统的Policy Gradient更新速度快。

但比较难收敛

### 4.3是否能以及如何应用于训练模型的演化

能应用于训练模型的演化

强化学习与监督学习的不同主要表现在强化信号上，强化学习中由环境提供的强化信号是对产生动作的好坏作一种评价，而不是告诉强化学习系统RLS如何去产生正确的动作。

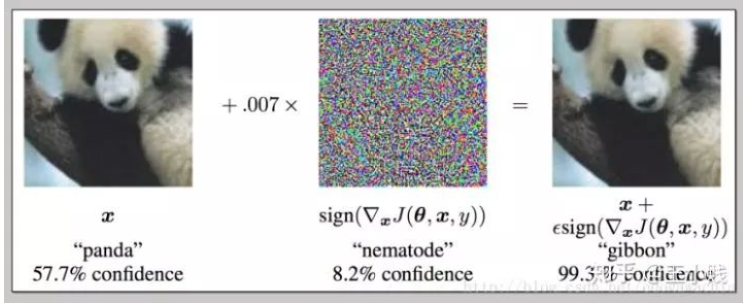
如果Agent的某个行为策略导致环境正的奖赏(强化信号)，那么Agent以后产生这个行为策略的趋势便会加强。Agent的目标是在每个离散状态发现最优策略以使期望的折扣奖赏和最大。

强化学习把学习看作试探评价过程，Agent选择一个动作用于环境，环境接受该动作后状态发生变化，同时产生一个强化信号(奖或惩)反馈给Agent，Agent根据强化信号和环境当前状态再选择下一个动作，选择的原则是使受到正强化(奖)的概率增大。选择的动作不仅影响立即强化值，而且影响环境下一时刻的状态及最终的强化值。

强化学习系统学习的目标是动态地调整参数，以达到强化信号（反馈奖励）最大。

## 5从深度学习、宽度学习出发了解一下代码的可解释性

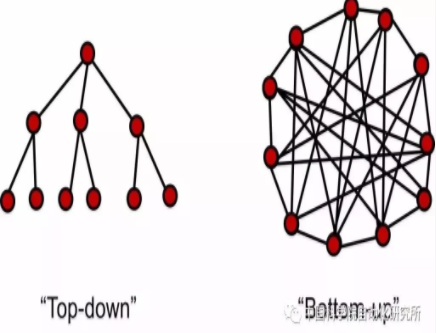
### 5.1一般的：

* 可解释性：我们需要了解或解决一件事情的时候，我们可以获得我们所需要的足够的可以理解的信息。
* eg：潮涨潮落->月球
* 不可解释性：如果在一些情境中我们无法得到相应的足够的信息，那么这些事情对我们来说都是不可解释的
* eg：对于一个CNN模型，在熊猫的图片中添加了一些噪声之后却以99.3%的概率被判定为长臂猿：
* 
* 我的理解：通常来说，我们希望对各种问题和任务等，都得到一个正确的答案和与之相符合的合理解释，但在如今的机器学习中，运用神经网络等方法，我们往往会得到准确度极高但很难去解释的模型结果，很多人把这种方法称为“黑箱”，黑箱代表未知，未知意味着潜在的危险，如何去权衡准确度极高的结果和潜在的危险是一个重要的问题。当然，可解释性始终是一个非常好的性质，我们总能从模型的可解释性中受益，从而验证并改进工作
* 数据和机器学习模型的可解释性是在数据科学的 “有用性”中至关重要的方面之一，它确保模型与您想要解决的问题保持一致

可解释性的用处：

* 判别并减轻偏差
* 考虑问题的上下文
* 改进泛化能力和性能
* 道德和法律原因

通常，随着模型复杂性的增加，模型可解释性按照同样的速度降低。特征重要性是解释模型的一种基本方法。即使对于深度学习等黑盒模型，也存在提高可解释性的技术。



* 传统建模时，我们使用一种自上而下的数据科学方法
* 深度学习中，我们使用自下而上的数据科学方法，通常将手动和部分困难任务自动化。

### 5.2深度学习角度：

个人理解的深度学习代码的可解释性主要是是指建模时的，代码其实就是模型的思想方法的数学化公式化表达，我们用一些可解释的方法去建立可解释的模型，那么代码也就是可以解释的，不过这种做法要求和限定很高，因为在机器学习中，对于那些数据量很大很复杂的问题，往往我们会考虑使用神经网络这种类似“黑箱”的模型，由于参数的数量以及提取和组合特征的方法复杂，我们是很难去解释他其中每一步的原理和结果产生的原因的，解读模型最终结果当然也就更加困难，这也就加大了我们进一步训练学习模型的难度。可以说，可解释性差是目前深度学习最大的缺陷之一

深度学习的黑箱性主要来源于其高度非线性性质，每个神经元都是由上一层的线性组合再加上一个非线性函数的得到，我们无法像理解线性回归的参数那样通过非常solid的统计学基础假设来理解神经网络中的参数含义及其重要程度、波动范围。

实际上我们是知道这些参数的具体值以及整个训练过程的，所以神经网络模型本身其实并不是一个黑箱，其黑箱性在于我们没办法用人类可以理解的方式理解模型的具体含义和行为，而神经网络的一个非常好的性质在于神经元的分层组合形式，这让我们可以用物质组成的视角来理解神经网络的运作方式。

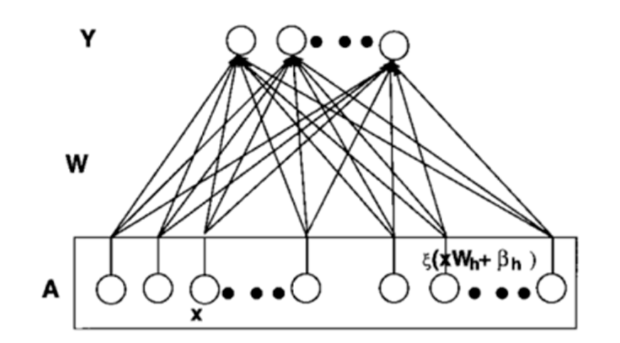
### 5.3宽度学习Broad Learning System角度：

#### 5.3.1宽度学习：

增加了从输入层到输出层的直接连接。网络的第一层也叫**输入层**，第二层改名了，叫做**增强层**，第三层是**输出层**。具体来看，网络中有三种连接，分别是

* （输入层 => 增强层）加权后有非线性变换
* （增强层 => 输出层）只有线性变换
* （输入层 => 输出层）只有线性变换

##### RVFLNN：



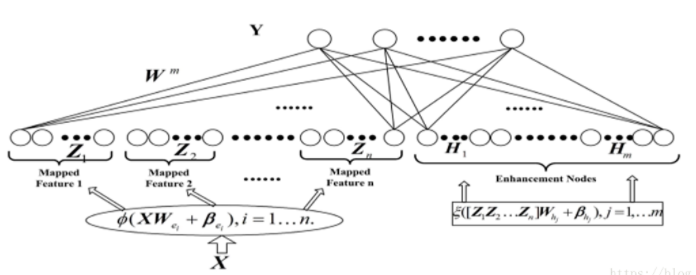
我们把增强层和输入层排成一行时，将它们视为一体，那网络就成了由 A（输入层+增强层）到 Y 的线性变换了！线性变换对应的权重矩阵 W 就是 输入层加增强层 到 输出层 之间的线性连接。

这时你可能要问：那输入层到增强层之间的连接怎么处理/优化？我们的回答是：不管它。我们给这些连接随机初始化，固定不变

如果我们固定输入层到增强层之间的权重，那么对整个网络的训练就是求出 A 到 Y 之间的变换 W，而 W 的确定非常简单：  
 W=A−1Y

输入 X 已知，就可以求出增强层 A；训练数据的标签已知，就知道了 Y。

##### BLS：



相对于RVFLNN，BLS对输入层做了一点改进，就是不直接用原始数据作为输入层，而是先对数据做了一些变换，相当于特征提取，将变化后的特征作为原RVFLNN的输入层。在我们不把第一层叫做输入层，而是叫它**特征层**。

当给定了特征 Z，直接计算增强层 H，将特征层和增强层合并成 A=[Z|H]，竖线表示合并成一行。由于训练数据的标签 Y 已知，计算权重 W=A−1Y 即可。即当数据固定，模型结构固定，可以直接找到最优的参数 **W**

但数据一般是不固定的，模型也是不固定的，我们需要调整数据的维数，比如增加新的特征。

宽度学习的核心在其增量学习算法，因为当数据量上亿时，相当于矩阵 **Z** 或 **X** 有上亿行，每次更新权重都对一个上一行的矩阵求*伪逆* 是不现实的。(利用上一次的计算结果，和新加入的数据，只需少量计算就能得进而得到更新的权重。)

相比深度学习的反复训练，时常陷入局部最优无法自拔，宽度学习的优势非常明显。

#### 5.3.2代码的可解释性：

与深度学习的神经网络不同，去年提出的宽度学习BLS，不采用深度的结构，基于单隐层神经网络构建，可以用易懂的数学推导来做增量学习，最初是为了解决深度学习参数数量过多，运行速度过慢且消耗大量资源的问题而提出的。

在特征数量不那么庞大时，研究证明，不断增加非线性网络层数，增加模型的复杂度，并不是优化学习模型所必须的，只要增强层单元数量足够多，也能够很好的逼近我们希望学习到的非线性函数。

相对于深度学习模型，宽度学习模型的网络层数减少，复杂度降低，参数数量减少，算法嵌套减少，代码可解释性自然更强，这也算是未来发展的一个趋势，随着其他缺陷不断被弥补（比如数据量，运算能力等），寻求可解释性的提升已经成为模型研究的一个重要方向。不过，就目前来看，宽度学习的训练结果精度不高，要推广还需要深入探索更好的学习算法