推荐系统评价：NDCG方法概述

人工智能头条 2015-09-09

【编者按】在信息过剩的互联网时代，推荐系统的地位随着大数据的普及愈发重要。评估一个推荐模型的质量面临很多棘手的问题，我们常用的指标是直接的准确率、召回率，但准确率不一定具有很好的相关性。来自Zygmunt Z的这篇综述文章，把推荐当作是一个排名任务，提供了一种更注重相关性的视角来进行推荐系统的评估，颇具可读性。

如果你挖掘的信息较少，推荐的方法有很多。问题是，选择哪一个模型更合适。在这里，主要的决策因子是推荐质量。你可以通过验证来估计它，而推荐系统的验证可能会很棘手。你需要考虑一些东西，包括任务的制定，可用的反馈形式和一个优化指标。下面，我们来解决这些问题并呈现一个实例。

排名推荐

我们把推荐当作是一个排名任务，这表示我们主要感兴趣的是一个相对较少的项，我们认为这些项最相关并把它呈现给用户。这就是众所周知的Top-k推荐。

把它和评级预测做比较，如Netflix的竞赛。2007年，Yehuda Koren（此次比赛的一个胜出者）指出，人们对使用RMSE作为一个指标并赞成使用RMSE指标存有疑惑，我们应该使用一个特定的排名指标。

然而，在我们有限的实验中，我们发现RMSE指标并不适用于排名。对我们而言，当调整用户留存率时，用于RMSE的矩阵分解优化表现的相当不错，但当从所有的可用项选择推荐时，却彻底地失败了。

我们认为原因是训练会集中于评分较高的项，同时对于这些项产生一个很好的拟合结果。而对于评分较低的项，在损失影响方面没有太大的意义。结果，对他们的预测会不平衡，使得与实际得分相比，一些得分较高，一些得分较低。最后，靠前的条目将显示在热门推荐一栏中，因而破坏了推荐结果。

换句话说，RMSE指标不能辨别真实的内情，而且我们需要特定的排名指标。

排名指标

两个最受欢迎的排名指标是MAP和NDCG。我们在前段时间已经使用了平均精度均值（MAP）。NDCG表示归一化折损累积增益。两者之间的主要区别是，MAP认为是二元相关性（一个项是感兴趣的或者不感兴趣的），而NDCG允许以实数形式进行相关性打分。这种关系类似分类和回归的关系。

实际当中，很难直接地优化MAP或NDCG指标，因为他们是不连续的，所以不可微。幸运的是，排名学习中的排名指标和损失函数表明，用于排名学习的一对损失函数近似于这些指标。

NDCG

NDCG这个名字可能有点吓人，但其背后的思想却很简单。一个推荐系统返回一些项并形成一个列表，我们想要计算这个列表有多好。每一项都有一个相关的评分值，通常这些评分值是一个非负数。这就是*gain*（增益）。此外，对于这些没有用户反馈的项，我们通常设置其增益为0。

现在，我们把这些分数相加，也就是*Cumulative Gain*（累积增益）。我们更愿意看那些位于列表前面的最相关的项，因此，在把这些分数相加之前，我们将每项除以一个递增的数（通常是该项位置的对数值），也就是折损值，并得到DCG。

在用户与用户之间，DCGs没有直接的可比性，所以我们要对它们进行归一化处理。最糟糕的情况是，当使用非负相关评分时DCG为0。为了得到最好的，我们把测试集中所有的条目置放在理想的次序下，采取的是前K项并计算它们的DCG。然后将原DCG除以理想状态下的DCG并得到NDCG@K，它是一个0到1之间的数。

你可能已经注意到，我们使用*K*表示推荐列表的长度。这个数由专业人员指定。你可以把它想像成是一个用户可能会注意到的多少个项的一个估计值，如10或50这些比较常见的值。

这里有一些计算NDCG的Python代码，非常简单。

要注意到，我们实验的测试集由训练集以外的所有项组成，包括那些没有用户排名的项（与上面RMSE讨论中提到的一样）。有时人们会对用户留存率设置测试限制，所以推荐系统的任务是减少调整那些相对较少的项。在实际情景当中并不如此。

现在，它的要点是，还有另一种DCG表述。你还可以使用负相关分数。在这种情况下，你可以计算出更糟糕情况下DCG的归一化（它将小于零），或者仍然使用零作为下限值，具体要视情况而定。

反馈形式

有两种类型的反馈形式：显性反馈和隐性反馈。显性反馈表示用户率项。另一方面，隐性反馈来自于用户行为的观察。大多数通常是二元的：用户点击了一个链接，观看了一个视频，浏览了一个产品，购买了一个产品。隐式反馈不常以计数的形式出现，例如用户听一首歌的次数是多少。MAP只是一种二元反馈指标，而NDCG可以在任何情况下使用，你可以对推荐项指定相关分数（二元、整数或是实数）。

弱泛化和强泛化

我们可以把用户（和项）分成两组：训练集的一组和非训练集的一组。第一组的验证分数对应于所谓的弱泛化，而第二组对应于强泛化。在弱泛化的情况下，每个用户都在训练集。我们采取一些评价用于训练，剩下的评价用于测试。在评估强泛化时，用户既可用在训练中，也可用在测试中。

事实上，我们主要感兴趣的是强泛化，因为在现实生活中，我们给用户推荐的条目并不存在于训练集。我们可以通过重新训练模型来解决这个问题，但这在实时推荐系统当中并不可行（除非我们的模型碰巧使用的是在线学习，这表明它可以使用实时获得的新数据进行更新）。我们假设将使用一个没有数据更新的预训练模型，因此我们需要一种方式来解释先前看不见的用户。

处理新用户

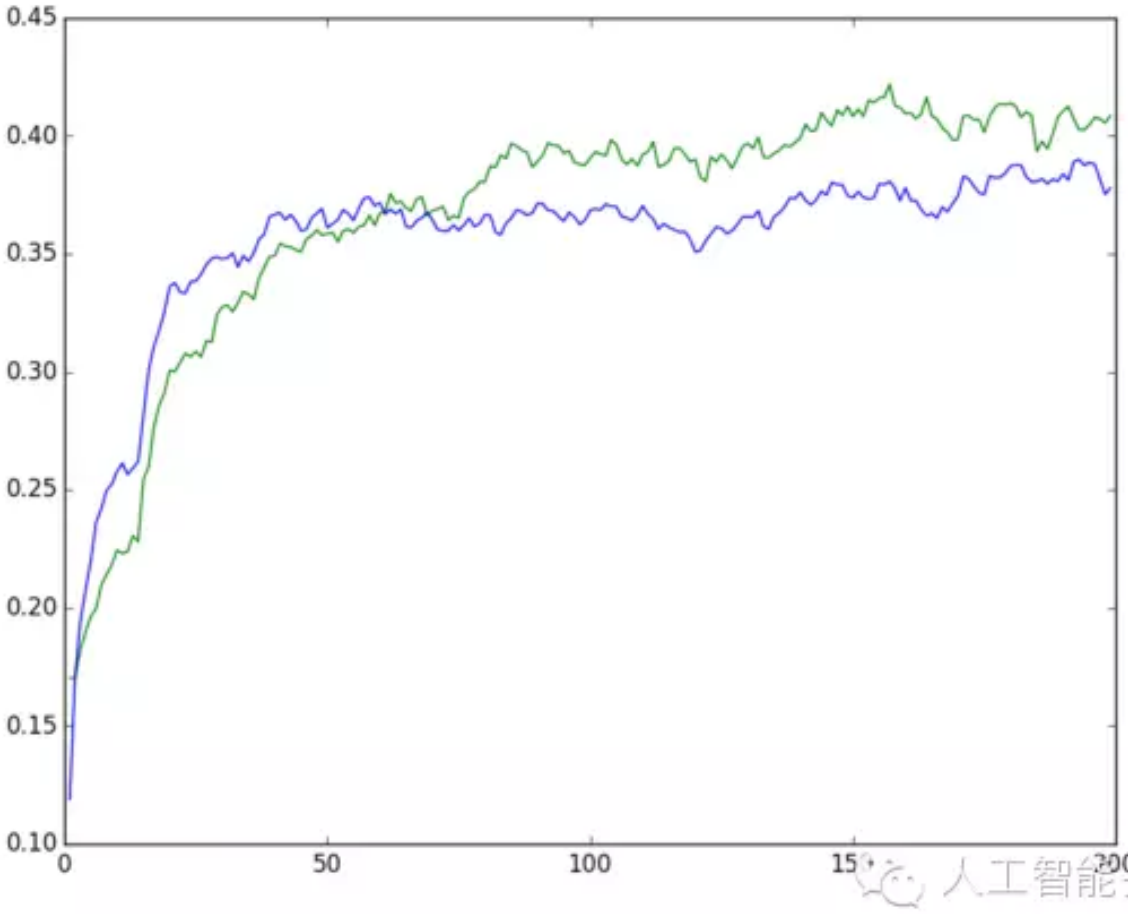
有些算法在这种场景中会更合适，而有些算法则会变得更糟糕。例如，人们可能会说矩阵分解模型不能为新用户提供推荐。这是不正确的。例如，使用交替最小二乘（ALS）。该方法通过在调整项因子时保持用户因子不变，并在调整用户因子时保持项因子不变，从而拟合模型，依次循环直至收敛。在测试时，当我们从一个新用户输入时，我们可以保持项因子不变来拟合用户因子，然后进行推荐。

一般来说，当预测率是用户和项因子之间的点积时，我们可以采取项因子和解决系统的线性方程组来估计用户因子。这相当于拟合一个线性回归模型。我们希望的比率（实例）数是大于因子的数，但即使不能如愿，我们也要感谢正则化。

缺乏实例是一个众所周知的*冷启动*问题：如果一个新的访问者没有评分，那么协同过滤对于推荐就没用。只有在我们有一些反馈之后，我们才能使用它开始工作。

越多越好

一般情况下，一个推荐系统得到的信息越多就会表现得越好，理想的情况下，当系统从给定用户中得到更多评价的时候，推荐的质量就会提高。在评价一个系统时，我们要考虑这个维度。为了完成这个，对于一个给定的用户，我们选择一个评价来训练，剩下的用来测试，然后选择两个评价进行训练，剩下的用来测试并依次下去，重复计算推荐和NDCG，直到达到某个特定数值或者测试集中没有剩余的评价为止。然后，我们绘制出如下结果图。



X轴是训练的评价数，Y轴是用户NDCG@50均值

当比较两个推荐系统的结果时，绘图将揭开它们的不同。要么一个比另一个更好，要么在曲线的某些点上相交。

该交叉点提供了使用两个系统组合的一种可能性。最初我们采用的是第一个系统，当获得的反馈大于阈值时，我们切换到另一个系统。在这里，当给出少许评价数时蓝色会表现的更好，但当评价数大约50个时就

会收敛。当提供更多的评价时，绿色则占据上风。

这个分数是在大约1000个用户组成的测试集中计算得到的，这个样本大小提供了可识别的模型，但是仍然有一些噪音，正如你从锯齿线上看到的那样。

事实上，我们需要的应该是一个数字而不是一个绘图，我们可以在训练中平均化等级数目之间的得分，我们称这个数为L。由此产生的指标是MANDCG：均值（用户之间）平均（1到L之间）NDCG。

本文的代码在GitHub上可以获得。要运行它，在你的推荐系统上需要提供的数据和插件。

最后，我们诚邀您来探索如何在MovieMood上使用更多的评价数来提升推荐系统的质量。

原文链接：Evaluating recommender systems（译者/刘帝伟 审校/刘翔宇、朱正贵、李子健 责编/周建丁）

阅读原文