

Advancing mathematics by guiding human intuition

with AI: [Nature (2021) 01 December 2021]

研究成果概括

本文作者： 李昀哲 Yunzhe Li 20123101

学校： 上海大学 Shanghai University

研究方向： 深度学习 Deep Learning, 计算机视觉 Computer Vision

作者简介： 上海大学 RoboMaster 战队视觉组成员，参与研发机器人移动预测击打，部署于移动开发平台（如：jetson tx2, agx 等），对计算机视觉、数据分析感兴趣且初步了解。

前言：

本文将对 2021 年 9 月 Nature 封面论文《Advancing mathematics by guiding human intuition with AI》进行解读。机器学习是人工智能浪潮中的热点话题，其应用和价值不仅在于“将楼筑高”、提升模型质量；同时也在于“将楼建广”，即，考虑如何将机器学习更好地应用于其他领域，帮助其他领域专家的工作。本文中，DeepMind 研究人员通过机器学习手段，帮助数学领域专家完成数学的研究工作，证明了两个整个数学界认可的定理。其中的机器学习模型并不复杂，亮点和价值在于将机器学习引入了一个新的领域。

论文地址： <https://www.nature.com/articles/s41586-021-04086-x>

源码地址： <https://github.com/deepmind>

论文作者： Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel

Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András

Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis & Pushmeet Kohli

研究背景

数学的研究往往包括：发现潜在的模式、关联，进而公式化、证明假设来得出定理。数学领域的研究和人工智能一般的研究方法不同：人工智能领域的学者，往往先获得一些数据，通过模型跑出一些结果，再从结果反推，观测是否存在可解释的点；而数学领域的研究，往往是先提出一些假设和猜想，再去想办法证明其正确性。1960 年以来，数学家们就通过计算机帮助他们发现模式和公式的猜想，如 BSD 猜想¹ (Birch and Swinnerton-Dyer conjecture)，但这里的计算机只是用来进行一些简单的计算、寻找反例¹⁰的工作或是构造一些符号解¹¹。计算机对于数学领域的帮助，在这篇论文发表前，是很少的。

不难看出，对于数学领域的研究（如：提出假设、猜想），都是需要一些“数学直觉”的推断，从特殊到一般的思想证明产生，但对于一些复杂的情况，这常会耗费很长的时间且并不容易。提出猜想往往是自然的，但很多猜想是找不到侧重点去证明的。2000 年 Clay 研究所提出的数学七大猜想，仅庞加莱猜想得以证明，因此，仅凭借人类的数学直觉和意识，证明数学问题是困难的。

DeepMind 就将机器学习³⁻⁵应用于此，通过人工智能，引导人类的直觉。简单来说，即，通过对所需研究的不变量的特征进行提取，抽象出他们之间的关联，列出潜在关联量之间的关联比率，引导数学家更快地找到证明方向，进而更快地证明定理。

研究成果

1 发现和证明了拓扑学中，结 (Knots) 的代数结构和几何结构之间一种新的联系

低维拓扑学 (Low-dimensional topology) 是数学研究中十分重要的领域，结 (Knots) 是低维拓扑学中比较关心的对象，主要的目标是对这些结进行分类，了解性质，从而建立和其他物体的关系。几何结构中的双曲线不变量 (hyperbolic invariant) 和代数结构中的

代数不变量 (algebraic invariant) 常作为切入点进行研究，成果 1 即为确立了二者之间的联系。

数学可以看作是一种很精确的描述世界的语言，又能细分为几何语言和代数语言，几何和代数之间的联系、对应关系是值得思考的，例如：中学阶段的证明题，往往可以通过几何解法和代数解法两个角度考虑，能否将两种角度中的每个概念一一对应，找到关系是数学中关心的问题。对于本节研究的拓扑学中的结的问题，先假设几何不变量和代数不变量之间存在联系，但具体是什么量，不得而知，如图 1 所示为双曲线结的不变量示例。因此需要机器学习的方法，选择出起决定性的量。



z: Knot	X(z): Geometric invariants				Y(z): Algebraic invariants		
	Volume	Chern-Simons	Meridional translation	...	Signature	Jones polynomial	...
	2.0299	0	i	...	0	$t^{-2} - t^{-1} + 1 - t + t^2$...
	2.8281	-0.1532	$0.7381 + 0.8831i$...	-2	$t - t^2 + 2t^3 - t^4 + t^5 - t^6$...
	3.1640	0.1560	$-0.7237 + 1.0160i$...	0	$t^{-2} - t^{-1} + 2 - 2t + t^2 - t^3 + t^4$...

图 1 三种双曲线结中的不变量示例

前人的猜想为：几何不变量中，体积 (Volume) 可由代数不变量中的 Jones 多项式得出。而本文团队提出了一个先前未有的猜想：代数结构中的代数签名 (Signature) 和几何结构中的很多量都有关系。因此，该团队利用归因技术 (Attribution technique) 发现了三个特征比较重要，如图 2 a 所示，前三个特征显著大于之后的特征。使用这三个特征建立的模型和初始用所有特征建立的模型训练结果相似，如图 2 b 故确定这三个特征和代数签名 (Signature) 是存在关系的，将这种关系记为 f ，对 f 进行研究。

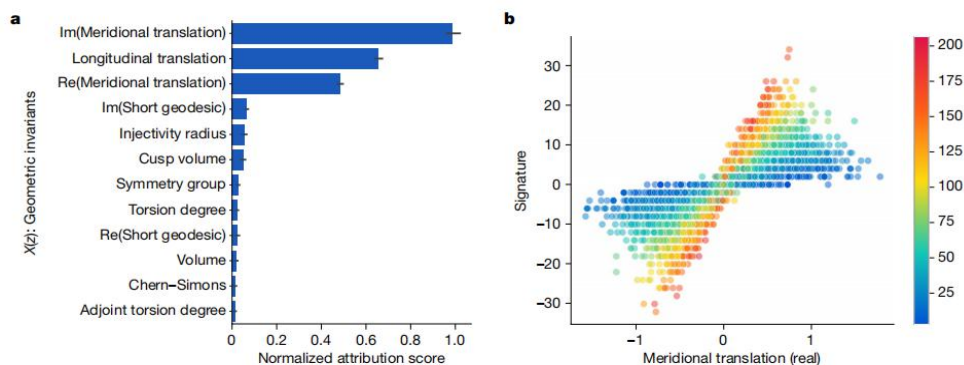


图 2 结理论 (Knot theory) 归因

将经向变换 (meridional translation) 记为 μ , 纵向变换 (longitudinal translation) 记为 λ , 引入一个斜度 (slope K) 为 $K = \text{Re}(\lambda/\mu)$, 得到了一个初步猜想:

$$|2\sigma(K) - \text{slope}(K)| < c_1 \text{vol}(K) + c_2$$

但在大量数据集的观测中, 发现了反例, 故引入了一个新的参数, 内射性半径 (injectivity radius), 记为 $\text{inj}()$, 从而得到定理: (证明见参考文献⁶)

$$|2\sigma(K) - \text{slope}(K)| \leq c \text{vol}(K) \text{inj}(K)^{-3}$$

这个领域已经研究了很多年, 但仍存在这样一种简单的关系, 这是非常让人意外的。

2 基于对称群⁷的组合不变猜想预测而来的一种候选算法

表示论 (Representation theory) 是一种线性对称性的理论, 该团队在该领域的研究内容为, 提出猜想和验证 KL 多项式能从区间图中直接计算得到, 如图 3 所示。

z: Pair of permutations	X(z): Unlabelled Bruhat interval	Y(z): KL polynomial
(03214), (34201)		$1 + q^2$
(021435), (240513)		$1 + 2q + q^2$

图 3 Bruhat 区间图数据集元素示例

这里的研究方式和上述拓扑学中类似，但这里的输入输出发生了变化，输入变为了一张图，输出变为了一个多项式，对于计算机而言，即，特征和标签发生了变化。最初发现用整张图预测 KL 多项式，可以得到不错的结果，现在欲知道到底图中的什么信息决定了预测的精度，受前人工作⁸启发，发现，输入图的一个子图就足够计算 KL 多项式。于是，再次使用归因技术，计算关联性，如图 4 所示。

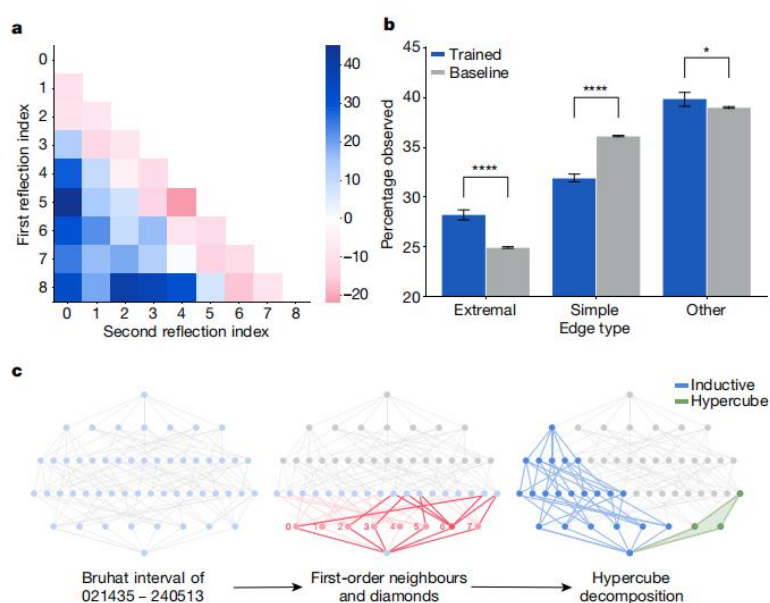


图 4 表示论归因图

发现可以把一个区间分为两个部分，会产生一个超立方体 (hypercube)，也足以计算 KL 多项式。得到结论：每个区间图，都可以得到一个 hypercube 的分解，从分解中可以计算 KL 多项式。结论和证明参见参考文献⁹。

代码验证及优化

1 网络定义

```
def net_forward(inp):
    return hk.Sequential([
        hk.Linear(300),
        jax.nn.sigmoid,
        hk.Linear(300),
        jax.nn.sigmoid,
        hk.Linear(300),
        jax.nn.sigmoid,
        hk.Linear(int((max_signature - min_signature) / 2)),
    ])(inp)

def softmax_cross_entropy(logits, labels):
    # labels are the true values of the signature
    one_hot = jax.nn.one_hot((labels - min_signature) / 2, logits.shape[-1])
    return -jnp.sum(jax.nn.log_softmax(logits) * one_hot, axis=-1)

# The cross-entropy loss is composed with the network predictions, to define
# "loss_fn" as a function on X and y.
def loss_fn(inputs, labels):
    return jnp.mean(softmax_cross_entropy(net_forward(inputs), labels))

# Haiku network transformation steps.
loss_fn_t = hk.without_apply_rng(hk.transform(loss_fn))
net_forward_t = hk.without_apply_rng(hk.transform(net_forward))

@jax.jit
def predict(params, data_X):
    return (np.argmax(net_forward_t.apply(params, data_X), axis=-1) * 2 +
            min_signature)

def normalize_features(df, cols, add_target=True):
    features = df[cols]
    sigma = features.std()
    if any(sigma == 0):
        print(sigma)
        raise RuntimeError(
            "A poor data stratification has led to no variation in one of the data "
            "splits for at least one feature (ie std=0). Restratify and try again.")
    mu = features.mean()
    normed_df = (features - mu) / sigma
    if add_target:
        normed_df[target] = df[target]
    return normed_df

def get_batch(df, cols, size=None):
    batch_df = df if size is None else df.sample(size)
    X = batch_df[cols].to_numpy()
    y = batch_df[target].to_numpy()
    return X, y

normed_train_df = normalize_features(train_df, column_names)
normed_validation_df = normalize_features(validation_df, column_names)
normed_test_df = normalize_features(test_df, column_names)
```

网络上就是一个简单的多层感知机，经过一些参数的定义，很快就能得到运行结果，如

图 5 所示，精度在 82.71%左右。

```
Step count: 0
Train loss: 2.3615565299987793
Validation loss: 2.3296875953674316
Step count: 100
Train loss: 0.9940038919448853
Validation loss: 0.9947787523269653
Step count: 200
Train loss: 0.5948654413223267
Validation loss: 0.5732844471931458
Step count: 300
Train loss: 0.4960586130619049
Validation loss: 0.48075395822525024
Step count: 400
Train loss: 0.5939966440200806
Validation loss: 0.4474546015262604
Step count: 500
Train loss: 0.6538435220718384
Validation loss: 0.4247085750102997
Step count: 600
Train loss: 0.31463027000427246
Validation loss: 0.40527692437171936
Step count: 700
Train loss: 0.42487049102783203
Validation loss: 0.4031493365764618
Step count: 800
Train loss: 0.3614102005958557
Validation loss: 0.3890964388847351
Step count: 900
Train loss: 0.40526723861694336
Validation loss: 0.4102986454963684
Validation loss increased. Stopping!
Test Accuracy: 0.8271358
```

图 5 运行结果展示

2 AutoGluon^{1 2}优化

参考李沐专家的讲解后，利用 AutoGluon 这样的 Auto ML 工具，可以达到利用更少的代码得到结果，且不需要调参。

通过以下代码进行预测，时间限制可以去除，此处为了更快的得到结果，故加以限制，但效果仍旧保持良好。

```
from autogluon.tabular import TabularPredictor
predictor = TabularPredictor(label=target).fit(
    train_df[column_names + [target]],
    tuning_data = validation_df[column_names + [target]],
    time_limit = 240)
```

查看模型结果：如图 6 所示，将三个模型融合后的精度更高，可以比原先 DeepMind 达到的 83 %的精度提升 10 个百分点，达到 97.23%的精度。

	model	score_test	score_val	pred_time_test	pred_time_val	fit_time
0	WeightedEnsemble_L2	0.972333	0.973480	7.760945	8.585636	235.275419
1	NeuralNetFastAI	0.966326	0.967013	0.631391	0.469818	168.253261
2	LightGBMXT	0.960911	0.960449	4.560817	4.472106	58.473782
3	KNeighborsDist	0.947488	0.948272	2.553903	3.638240	1.843957
4	KNeighborsUnif	0.934327	0.935373	2.146031	4.555897	1.910410

图 6 优化后的训练精度

精度并非最重要的，还可以显示训练后的关键特征是否正确：如图 7 所示，和论文中相符。

	importance	stddev	p_value	n	p99_high	p99_low
meridinal_translation_real	0.470546	0.000648	3.157433e-07	3	0.474258	0.466835
longitudinal_translation	0.391606	0.000476	2.462788e-07	3	0.394334	0.388878
meridinal_translation_imag	0.207794	0.000693	1.856123e-06	3	0.211768	0.203821
volume	0.112158	0.001982	5.206499e-05	3	0.123518	0.100798
cusp_volume	0.100823	0.001329	2.897310e-05	3	0.108440	0.093205
short_geodesic_imag_part	0.097157	0.000398	2.796269e-06	3	0.099438	0.094877
hyperbolic_torsion_degree	0.077879	0.000519	7.411128e-06	3	0.080855	0.074903
short_geodesic_real_part	0.021215	0.000810	2.425276e-04	3	0.025854	0.016576
injectivity_radius	0.006488	0.000887	3.086506e-03	3	0.011571	0.001405
Symmetry_Z/2 + Z/2	0.004256	0.000137	1.717317e-04	3	0.005039	0.003473
chern_simons	0.004136	0.000237	5.450050e-04	3	0.005492	0.002779
hyperbolic_adjoint_torsion_degree	0.003873	0.000098	1.076929e-04	3	0.004437	0.003309
Symmetry_0	0.001214	0.000292	9.348831e-03	3	0.002886	-0.000457
Symmetry_D4	0.000777	0.000068	1.284460e-03	3	0.001168	0.000385
Symmetry_D6	0.000503	0.000038	9.425079e-04	3	0.000720	0.000286
Symmetry_D3	0.000033	0.000000	5.000000e-01	3	0.000033	0.000033
Symmetry_D8	0.000000	0.000000	5.000000e-01	3	0.000000	0.000000

图 7 关键特征的排序

基于 Auto ML 工具，不仅可以使代码实现非常简单，不用花时间去考虑用什么模型、怎么调参；且它会去试很多模型、超参数，结果会比较稳定。

研究成果总结

该团队在两个不一样的数学领域帮助数学家提出了猜想并给予了证明，机器学习并不是帮助你去证明一个猜想，而是去帮助分析数学物体之间属性的关系。可以具体地告诉研究者，哪些属性可以用来预测另外一些属性。得到这些信息后，数学家再去猜测这些属性间存在哪些数学关系，再去证明，这样的研究方法更自然。

该团队研究的领域都已经被研究过多年，但仍旧发现了一些基础性的关系，此框架还能用于帮助数学家发现一些大的物体之间的联系，这些关系仅凭借人类直觉和肉眼是很难发现的，例如表示论中的区间图，肉眼是很难直接区分的。

但这个方法仍旧存在两个局限性：首先需要生成一个比较大的数据集，这也是整个机器学习的局限性，对数学领域来讲，数据相对来说是较少的，需要数学家和其他专家合作，才能用代码生成数据集；其次，我们并不能总可以将一个问题表示为一个机器学习的问题有些内容是比较难学的。这篇论文对于将机器学习应用到数学领域的一大突破，可能不是第一个，但一定是目前为止结果最好的一个，这也将作为一篇奠基性的文章鼓励更多的机器学习专家和数学家合作，更快地推动数学的进展。

该团队的机器学习模型或许有人会觉得过于简单，也确实存在很大的提升框架，但本文的意义并非提出一个很好的机器学习模型，而是让数学家了解到机器学习是一个很好的工具，帮助推进数学研究，通过两个难易程度不一样的例子，证明了该方法的泛化性。也指导我们，研究问题时，不应该过度关心模型的调参之类的问题，这样会把我们的重心从问题本身上移开，转而去关注算法的问题。在解决问题之后，才应该关心提升算法的性能，因此在上述的末尾，参考李沐专家的方法，使用了 autogluon 进行优化。

参考文献

1. Carlson, J. et al. The Millennium Prize Problems (American Mathematical Soc., 2006)
2. Birch, B. J. & Swinnerton-Dyer, H. P. F. Notes on elliptic curves. II. *J. Reine Angew. Math.* 1965, 79–108 (1965)
3. MacKay, D. J. C. *Information Theory, Inference and Learning Algorithms* (Cambridge Univ. Press, 2003).
4. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436–444 (2015).
5. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, 2006).
6. Davies, A., Lackenby, M., Juhasz, A. & Tomašev, N. The signature and cusp geometry of hyperbolic knots. Preprint at arxiv.org (in the press).
7. Brenti, F. Kazhdan-Lusztig polynomials: history, problems, and combinatorial invariance. *Sémin. Lothar. Combin.* 49, B49b (2002).
8. Braden, T. & MacPherson, R. From moment graphs to intersection cohomology. *Math. Ann.* 321, 533–551 (2001).
9. Blundell, C., Buesing, L., Davies, A., Veličković, P. & Williamson, G. Towards combinatorial invariance for Kazhdan-Lusztig polynomials. Preprint at arxiv.org (in the press).
10. Wagner, A. Z. Constructions in combinatorics via neural networks. Preprint at <https://arxiv.org/abs/2104.14516> (2021).
11. Lample, G. & Charton, F. Deep learning for symbolic mathematics. Preprint at <https://arxiv.org/abs/1912.01412> (2019).
12. AutoGluon: AutoML for Text, Image, and Tabular Data — AutoGluon Documentation