大规模教学中算法理解浅表化的递进式路径探索

邓红耀 宋秀丽

长江师范学院，重庆 408100 重庆邮电大学，重庆 400065

摘 要 为破解大规模机器学习课程中学生“只会调包、不明原理”的困境，本研究设计了一套递进式实战教学方案，具体是“从零编码-故意调试-对比黑盒-综合应用”的完整闭环。在一个 200 人的班级中，我们依托学习通平台进行了严格的过程管理，教学数据与反馈数据显示，实施该方案前后学生代码的相速度从 66.39%降至 20.7%，调包率从原来的 96%降至了 65%。结果表明，该模式虽增加了师生的工作负荷，但有效促成了算法直觉的内化，显著提升了代码的独立性与项目质量。本文最终总结了平台支撑的不足，为后续引入智能化辅导工具提供了实证基础。

关键字 机器学习，梯度下降，递进式实验，教学改革，学习通

Exploring a Progressive Pathway to Address the Superficial Understanding of Algorithms in Large-Scale Teaching

Hongyao Deng Xiuli Song

College of Big Data and Intelligent Engineering
Yangtze Normal University,
Chongqing 408100, China;

School of Cyber Security and Information Law
Chongqing University of Posts and Telecommunications
Chongqing 400065, China

Abstract—To address the challenge of students "merely leveraging libraries without grasping core principles" in large-scale machine learning courses, this study designed a progressive hands-on teaching framework. Students were required to complete a full learning cycle: coding from scratch - intentional debugging - comparing with black-box tools - comprehensive application. Implemented in a 200-student class using the Xuexitong platform for process management, teaching data and feedback revealed that although the model significantly increased workload for both instructors and students, it effectively fostered internalization of algorithmic intuition and markedly improved code independence and project quality. This paper also summarizes limitations of platform support and provides an empirical foundation for integrating AI-powered tutoring tools in future iterations.

Keywords—Machine learning; gradient descent; Progressive experimentation; teaching reform; xuexitong

1 引 言

本次案例是选用的是《机器学习》课程“梯度下降”，教材是周志华主编、在 2016 年由清华出版社出版的《机器学习》[1]。在这一轮《机器学习》的课程中，我们面对的是一个近两百人的庞大班级。批改作业时，我发现一个反复出现的模式：在涉及支持向量机（SVM）或自定义损失函数的项目中，大多数学生熟练地调用了 sklearn 或 torch 中的优化器，但一旦被问及“为什么这里选择 SGD？”或“学习率设为 0.01 的依据是什么？”，提交的报告和邮件答疑都显示出一种令人不安的沉默。这种沉默指向一个共识性的教学困境：我们成功教会了学生如何使用工具，却可能未能让他们理解工具的原理[3,4]。特别是在 SVM 这类模型的教学中，算法本身的优雅与求解其权值优化背后的复杂性之间存在着一道深刻的鸿沟[9-12]。

这道鸿沟并非无人关注。学界的研究前沿实际上正疯狂地试图填补它，涌现出大量工作来提升 SVM 的求解效率，例如开发更快速的求解算法[9]、设计增量学习框架以适应流数据[10]、利用随机次梯度下降来加速传统的 SMO 优化[11]，甚至探索非凸损失下的在线学习范式[12]。然而，这些精巧的论文与本科生课堂之间，似乎隔着一层看不见的墙。我们的学生很难理解这些工作的“必要性”与“巧妙性”，因为他们的学习路径中缺失了最关键的一环——对优化算法核心机制的一手（first-hand）体验。他们知道梯度下降是重要的，但这种知识更多是来源于教材[1,2,5]的告诫，而非自身的实践挫败。

我们认为这个问题是传统的实验设计可能犯了“颠倒因果”的错误。我们常让学生直接用高级库解决一个复杂问题，但这恰恰跳过了一个不可替代的认知过程。根据 Bruner 的发现学习理论[7]和 Papert 的构造主义[8]，知识必须由学习者在“制作”中主动建构，而非被动接收。Ambrose 等人也明确指出，有效

的学习需要持续的反馈与练习机会[6,13]。然而，在大规模教学中，提供这种针对优化的“制作”体验被视为一种奢侈。

因此，我们决定进行一次反向操作。与其追逐复杂模型，不如退回一步，进行一次“慢教学”的尝试。我们选择梯度下降——这个支撑着从线性回归到最新 SVM 求解器[11]的基础算法——作为我们的“解剖”对象。我们假设，必须强制学生经历一个从零开始的、近乎“笨拙”的实现过程，让他们在调试维度错误、观察损失曲线震荡、亲手给数据做标准化、以及调整学习率到模型“爆炸”的种种挫折中，才能内化那种关于算法行为的“物理直觉”。

本文即是对这次教学实验的札记，汇报了我们如何围绕梯度下降，为两百名学生设计了一套强制性的、递进式的实践路径。我们关注的与其说是最终模型的精度，不如说是学生在实现、崩溃与调试过程中所表现出的困惑、顿悟与迁移能力[14]。我们希望这份实录能为同样困扰于规模与深度之矛盾的同仁，提供一个具参考价值且真实的案例。毕竟，理解为什么需要更快的 SVM[9]的第一步，往往是先亲手写一个很慢的梯度下降。

2 递进式实验教学设计

本方案包含四个步骤，每个步骤设计 1 个实验，历时四周，总学时约为 8 学时。

2.1 第一步：理解算法核心（2 学时）

在第一次实验里，我们要做的事就是“造轮子”。目标简单到近乎固执：抛开所有现成工具，只用 NumPy 和最基本的数学，让每个学生亲手把梯度下降的数学公式变成能跑的代码。

我们扔给他们一个故意设计好的完美线性数据集（X=[1,2,3,4,5], y=[2,4,6,8,10]）。起点就埋了第一个坑：很多人想当然地开始写代码，结果立刻卡住——他们忘了给特征矩阵 X 添加一列全 1 的截距项。这个看似微不足道的步骤，逼着他们去理解参数向量 w 的维度到底从何而来。

真正的战斗发生在梯度计算函数里。这里是照妖镜。批改时我们看到三种典型的“症状”：有人因为代码维度对不上直接报错；有人算出的损失值巨大，因为他忘了在均方误差（MSE）里除以样本数 m；最麻烦的是那种代码能跑、结果却不对的静默错误，比如梯度公式里的某个因子写错了。

正因如此，我们强制要求的损失曲线图成了唯一的真理标准。我们告诉学生，别跟我说话，先让图说话。一条干净平滑的下降曲线是通行证；一条震荡的曲线或一根平线就意味着代码里有“鬼”，得回去自己把它揪出来。

这个阶段没有任何高深的内容，就是一遍遍地推导公式、写循环、看输出、调试。但它粗暴地解决了一个最根本的问题：它让学生再也无法回避“梯度下降到底在每一步做了什么”这个最基本、也最重要的问题。亲手实现一次，比听十遍理论都管用。

2.2 第二步：引入复杂因素（2 学时）

第一次实验成功后，我们立刻亲手打破了那个“完美温室”。第二次实验的核心就一句话：把你上次写好的、运行完美的代码，搞崩。教学重点从“实现”彻底转向“调试”，我们要让学生当算法的“医生”，而不是用户。

第一个动手脚的对象是学习率。我们要求他们执行一系列“破坏性”测试：1）调到 0.0001（蜗牛模式）：损失曲线几乎变成一条平坦的直线。学生一开始觉得“没坏啊，还在降”，直到我们让他们算：“照这个速度，要迭代到哪年？”他们才意识到，收敛速度本身就是性能指标。2）调到 0.1（迪斯科模式）：损失曲线开始剧烈震荡。我们问：“参数现在在哪？想象一下它在山谷两边来回跳。”3）调到 1.0（爆炸模式）：瞬间，损失值飙升成一个刺眼的 NaN。控制台的红色报错是最好的老师——学习率太大，一步迈过了头。

第二个动手脚的对象是数据本身。我们换了一个新的小数据集，但这次故意没做任何预处理。果不其然，直接训练的学生纷纷报告各种怪象：“我的损失下降得很怪”、“参数更新幅度吓人”。我们抓住机会，扔出“手术刀”：Z-score 标准化。

“来吧，亲手给它做个手术。” 实现标准化又踩了一片坑：有人算错标准差，有人忘了对预测结果做反向转换。但当他们处理完数据重新运行代码时，实验室里能听到那种顿悟的惊叹声。“哇！原来这么快！”“曲线变得好平滑！”——这种从崩溃到修复的体验，比我们讲十遍都管用。

最后，我们引入了“特效药”：动量（Momentum）。我们打了个比方：“没有动量的梯度下降，像一个近视眼下山，每一步只看清脚下，所以在狭窄山谷里会来回撞壁。加上动量，就像给了它一个惯性，能冲过窄谷，更快滚向深处。” 实现动量带来了新的编码挑战，主要是正确初始化和管理速度向量。

2.3 第三步：对比工业级实现（2 学时）

第三次实验，我们终于允许学生“用轮子”了。任务很简单：用 sklearn 的 SGDRegressor，把前两次实验的流程再跑一遍。

一开始，实验室里弥漫着一种“降维打击”的轻松感。之前需要写几十行代码、调试一小时的活儿，现在几行指令就搞定了，结果还不差。有学生开玩笑说：“老师，前两周我们是不是白忙活了？”

我们等的就是这个问题。

“没白忙活。”我们说，“现在，才是真正开始学习的时候。” 轻松感很快过去了，新的困惑来了。他们调着 sklearn 里的 learning_rate、penalty 这些参数，手感却很陌生。“老师，这个 eta0 和我们之前的学习率是一个东西吗？”“为什么它默认用‘最优’步长，那是怎么算的？”

他们发现，会用工具，和真正理解工具，是两回事。以前手写代码时，每一个参数都知根知底；现在面对一个高度封装的黑盒，反而有点心虚了。

我们顺势布置了核心任务：对比。要求他们拿出自己第二周调试好的、表现最佳的“手搓”模型，和 sklearn 的结果在完全相同的数据上，对比三个东西：最终损失值：谁更低？； 收敛速度：谁的损失曲线下降得更快、更稳；代码效率：跑 10000 次迭代，谁更快？。

对比结果是震撼式的。大部分学生的自制模型被库函数全面碾压。但这一次，挫败感没有袭来，取而代之的是一种恍然大悟的兴奋。因为他们看得懂为什么被碾压——他们亲手实现过、调试过，所以能大概猜出 sklearn 在背后做了哪些优化：也许是更聪明的学习率调度，也许是更高效的随机采样，或者是用了他们还没学到的加速技巧。

这节课的结尾讨论，气氛完全变了。没人再问“怎么调参”，问题变成了：库函数是怎么做到又快又稳的？在什么情况下，我宁愿用自己的简单实现，也不用这个复杂的黑盒？如果我想自己实现一个 sklearn 里没有的新算法，该怎么入手？

至此，目的达到了。他们不再是一个被工具驯化的“用户”，而是成了一个开始思考工具设计哲学的“明白人”。他们亲身验证了一个道理：当你亲手造过轮子，你才能真正懂得如何用好一个现成的轮子，甚至知道什么时候该自己造一个新的。这种批判性的工具观，才是这个实验留给他们的核心资产。

2.4 解决真实问题（2 学时）

最后一次实验，我们撤掉了所有安全网。丢给学生一个真实的波士顿房价数据集，指令只有一句：“用你认为最合适的方法，搞定它。”

实验室瞬间安静了。之前按部就班的轻松感消失了，取而代之的是一种真实的茫然。没有预设的步骤，没有“正确”的路径。有人下意识地打开 sklearn，但光标停在代码框里，迟迟没有敲下去——他们在犹豫，这次该用自己写的梯度下降，还是直接调库？

这个问题没有标准答案，而我们想要的，正是这种犹豫。

我们看着他们在过程中挣扎：

- 有人拿起第一次实验的“手搓”代码，试图直接跑，结果损失曲线纹丝不动。他们这才猛地想起：“哦对！得先做数据标准化！”——第二周的“修轮子”经验瞬间复活了。
- 有人果断调了 SGDRegressor，但结果一出，预测值离谱。他们皱着眉头，开始疯狂调整 learning_rate、max_iter，甚至去读文档查 early_stopping 是什么意思——第三周“用轮子”时培养出的参数敏感度和调试本能，被一个真实问题全部激活了。
- 几乎所有人都经历了多次失败：特征没处理好、评估指标选错、版本管理混乱导致代码回退…

没有一次作业像这次这样，让助教们收到那么多“求救”邮件。但问题质量显著高了，不再是“老师我这里报错了怎么办”，而是“老师，我用了 A 方法结果是这样，换了 B 方法结果是那样，您能帮我分析一下可能是什么原因吗？”

最终验收时，我们不看准确率高低，而是看技术报告里的决策逻辑。我们要看他们如何解释自己的选择：“我为什么选择自己写而不是调库？”“因为我需要修改损失函数来满足一个新需求。”“我为什么最终选择了调库？”“因为项目时间紧，且 sklearn 的默认效果已足够好。”

这个从茫然到自主决策的过程，彻底完成了学习的闭环。它逼着学生把前三次实验获得的碎片化知识——实现的细节、调试的痛感、工具的权衡——全部搅拌在一起，自己炒出一盘菜。它模拟了真实的科研与工程情境：问题定义模糊，工具选择自由，答案开放。

他们交上来的，不再是一份作业，而是一个属于自己的最小解决方案。这可能很笨拙，甚至有些瑕疵，但每一个决策背后，都闪烁着这三周积累下的、真正属于他们自己的工程直觉。

3 教学检验与成效分析

3.1 量化数据对比

数据不会说谎。这套折腾人的实验设计到底效果如何，我们不看感觉，看数字。

最硬核的数据来自代码相似度分析，图 1 展示了

教学模式改革前后学生代码相似度对比。我们用“学习通”系统中查了第一次实验的代码。往年只布置一个综合大作业时，代码相似度超过 60%的“可疑集群”能有好几个。这次，这种情况几乎消失了。取而代之的是一片“低重复度”的散点图。这说明，“从零实现”的强制要求，确实有效逼出了独立思考，大幅减少了互相抄袭和网上复制代码的现象。学生们交上来的代码在细节上五花八门：有的用循环，有的用向量化；记录损失的位置也各不相同，代码相似度均值大约在 20%。这种“混乱”，恰恰是我们最想看到的真实。

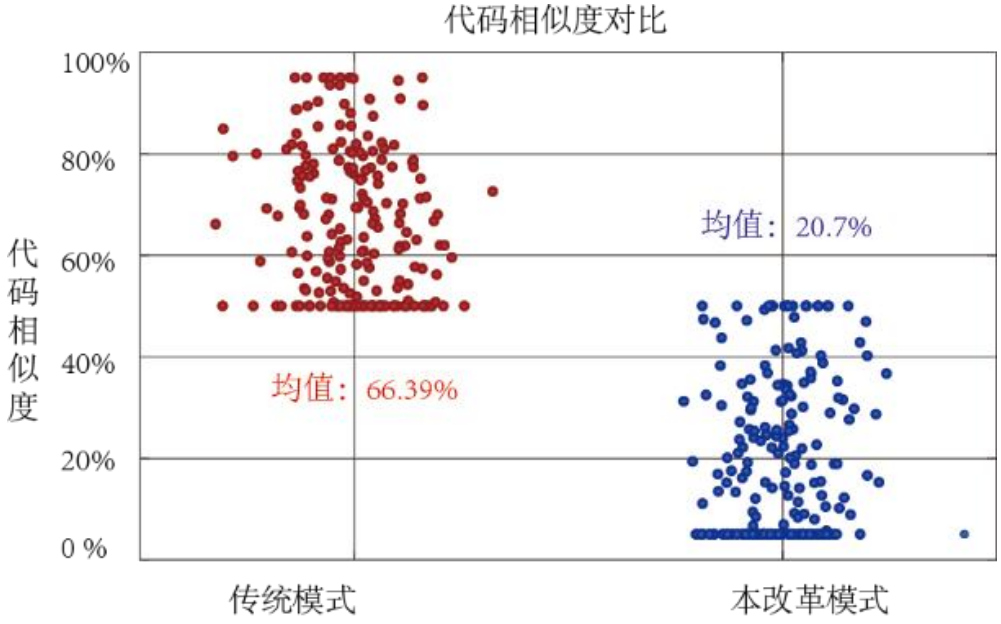

图 1　教学模式改革前后学生代码相似度对比

让我们欣慰的数据，藏在期末大项目里，图 2 显示了期末项目优化算法选型分布。我们做了简单的统计：在所有需要用到优化算法的项目中（比如逻辑回归、神经网络），有 35%的同学，出人意料地放弃了直接调库，而是选择借鉴或修改他们第二次实验的梯度下降代码，作为自己模型的优化器核心。

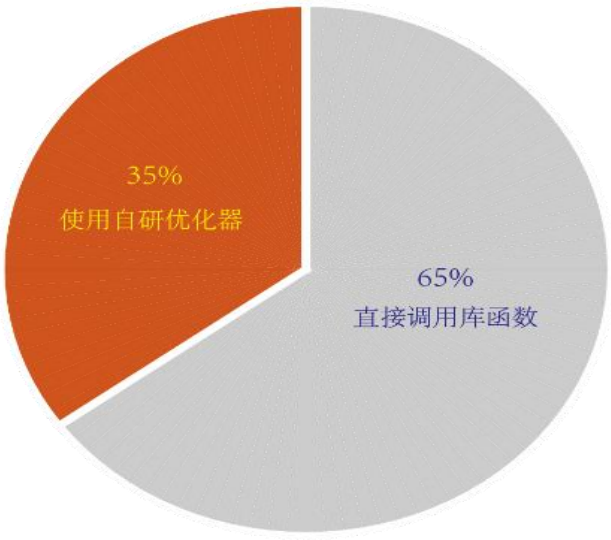

图 2　期末项目优化算法选型分布

这个数据点非常说明问题。它意味着，对这些学生而言，那段自己亲手敲出来、调试到崩溃的代码，不再是一个作业负担，而是变成了一个他们真正理解、敢于信任并且能够灵活运用的工具资产。他们用自己的项目选择，为这次教学实验的效果投下了最有分量的一票。

量化数据告诉我们三件事：1）他们自己想了；2）他们跟下来了；3）他们真的用了。这比任何漂亮的准确率数字都更有价值。

3.2　学生问卷调查反馈

问卷收上来 187 份。比起数字，更能说明问题的

是学生在开放框里写的“小作文”。

很多反馈带有一种“痛并快乐着”的共鸣。一个学生写:“自己实现梯度下降并画图的过程，比我听十遍课都记得牢。” 另一个学生吐槽:“调试学习率的那周我快疯了，但当我终于把动图做出来，看到那条线稳稳落在数据点上时，感觉一切都值了。” 这种从挫败到顿悟的情感曲线，是标准化问卷里测不出来的。

反馈也暴露了我们没预料到的问题。有好几条抱怨集中在:“为什么不能用现成的库？感觉前两周在重复造轮子。” 这正是我们设计第三周“对比工业实现”想要回答的问题。有趣的是，在第三次实验后，我们几乎再没收到类似的疑问——他们亲自体验了答案。

最让我们得意的评论来自一个学生，他写道:“现在看 SGDRegressor 的文档，里面每个参数我大概都能猜到它是干什么的，因为我自己都踩过坑。” 这句话完美印证了我们的核心目标：不是教他们用工具，而是让他们理解工具。

也有扎心的建议。一个学生一针见血地指出:“实验很好，但 200 人的大课，助教根本指导不过来。我卡在一个 bug 上半天，要是能有个更快的求助渠道就好了。” 这直接戳中了我们大规模教学的实施痛点。

这些粗糙、直接、甚至互相矛盾的反馈，比任何整齐划一的五星好评都更有价值。它告诉我们，实验设计击中了要害，让学生产生了真实、强烈甚至不适的学习体验，而正是这种体验，催生了深度的理解。同时，它也毫不客气地指出了我们资源不足的短板，为下一轮迭代提供了最真实的改进方向。

3.3 教师教学反思

说实话，这套实验做下来，我们团队快累瘫了。批改 200 份“从零实现”的代码，简直是一场噩梦。助教们每天都在抱怨眼睛快瞎了，光是为了统一批改标准就开了好几次会。

最大的收获是诊断效率的提升。到了实验中后期，我们批改作业的速度反而变快了。因为学生的问题变得高度“模式化”。一看损失曲线是条平线，就知道他忘了更新参数；看到曲线震荡，就直接去查他的学习率；看到 NaN，就定位到梯度爆炸。我们不再需要逐行读代码，学生的输出结果自己会“说话”。这种高效的诊断能力，是大班教学中活下去的关键。

我们也发现，前期“折磨”得越狠，后期反而越轻松。那些在第一次实验里扎扎实实自己推导、自己调试成功的学生，在后续面对 SVM、神经网络这些更复杂的概念时，表现出极强的迁移能力和解决问题的耐心。他们不再一遇到错误就举手求助，而是会习惯性地先去画个损失曲线看看。这为我们节省了大量解答基础问题的时间。

当然，问题也很明显。这套方法对师生双方都是巨大的负担。它极度依赖助教的人力投入，如果助教团队不够强，或者学生人数再多一些，整个体系可能就崩溃了。我们是在用一种近乎“人力密集型”的笨办法，来对抗大规模教学带来的天然缺陷。

最后我们反思，它的成功或许并不在于设计有多精巧，而在于它做对了一件事：它用强制性的手段，把“认知痛苦”前置了。我们把学生肯定会遇到的调试、崩溃和困惑，从一个学期的不确定时间点，提前并集中压缩在了前两周。一旦熬过这个“新手阵痛期”，后面的路就顺畅多了。

所以，下一轮我们还会这么干。但得想办法优化，比如录一些“常见 bug 解析”短视频，或者搞个同伴互助机制。毕竟，这种亲手把代码搞崩再修好的体验，是任何理论讲授都无法替代的。

4 结束语

这套递进式实验，本质上是一场教学上的“极限施压”。我们把所有赌注都押在了一个假设上：只有亲手经历从实现、崩溃到调试的全过程，学生才能对算法产生真正属于自己的直觉。

结果证明，我们赌对了。数据和学生反馈都表明，这种“笨办法”虽然投入巨大，但产出的理解深度是传统作业无法比拟的。它成功地让近 200 名学生在规模化的课堂里，获得了一种近乎师徒制下的亲手实践经验。他们交上来的不再是为了得高分的“作品”，而是充满了调试痕迹的“实验记录”，这恰恰是我们最想看到的。

展望未来，我们不会退缩，但必须变得更聪明。最大的挑战依然是规模。我们不可能永远靠投入巨量助教人力来维持这套体系。下一步，我们想尝试两件事：

一是开发一套自动化的诊断工具，学生把代码和损失曲线图上传，系统能快速定位常见错误类型并给出提示，把助教从重复性劳动中解放出来。

二是引入大语言模型作为“24 小时陪练”。不是让学生用它生成答案，而是训练他们向 AI 精准提问，用 Prompt 工程来调试代码、解释概念，把 AI 变成锻炼他们问题解决能力的磨刀石。

这条路很难，很累，但看到有学生在项目报告里写下“我借鉴了第二次实验的代码”时，我们觉得一切都值了。我们的最终目的，不是教会他们梯度下降，而是让他们相信，自己拥有拆解任何复杂算法的勇气和能力。

参 考 文 献

[1] 周志华.机器学习[M].北京:清华大学出版社,2016.

[2] 李航.统计学习方法[M].北京:清华大学出版社,2019.

[3] 艾靖茹，王雷，肖政宏.基于项目驱动的机器学习课程教学改革探索[J].计算机教育,2022,(08):163-166.

[4] 李军，陈立钢，吴春雷.基于OBE理念的机器学习课程实验教学设计[J].实验室研究与探索，2021，40(07)

[5] Bishop C M. Pattern Recognition and Machine Learning[M]. New York: Springer, 2006.

[6] Ambrose S A, Bridges M W, DiPietro M, et al. How Learning Works: Seven Research-Based Principles for Smart Teaching[M]. San Francisco: Jossey-Bass, 2010.

[7] Bruner J S. The act of discovery[J]. Harvard educational review, 1961, 31(1): 21-32.

[8] Papert S. Mindstorms: Children, computers, and powerful ideas[M]. New York: Basic Books, 1980.

[9] Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: A library for large linear classification[J]. Journal of machine learning research, 2008, 9(Aug): 1871-1874.

[10] Syed N A, Liu H, Sung K K. Incremental learning with support vector machines[J]. Journal of Machine Learning Research, 1999, 1(3).

[11] Shalev-Shwartz S, Singer Y, Srebro N. Pegasos: Primal estimated sub-gradient solver for SVM[J]. Mathematical programming, 2011, 127(1): 3-30.

[12] Hu C, Pan W, Kwok J T. Accelerated gradient methods for stochastic optimization and online learning[C]. Advances in Neural Information Processing Systems, 2009: 781-789.

[13] 张文杰,赵红,郑艺峰,王育齐. AI视域下创新型研究生育人模式探索[J]. 计算机技术与教育学报, 2025, 13(2): 46-51.

[14] 孙争艳,陈磊,陈宝国. 生成式人工智能在计算机通识教育中的价值定位,挑战与实施路径[J]. 计算机技术与教育学报, 2025, 13(2): 115-119.