arXiv:2301.01383v1 [cs.LG] 3 Jan 2023

如何充分发挥孪生回归方法的作用

Sebastian J. Wetzel

滑铁卢大学，加拿大安大略省滑铁卢市 N2L 3G1；周界理论物理研究所，加拿大安大略省滑铁卢市 N2L 2Y5；Homes Plus Magazine Inc.，加拿大安大略省滑铁卢市 N2V 2B1

摘要。 孪生回归方法旨在解决原始回归问题的对偶问题，即预测回归目标之间的差异，而不是目标值本身。通过集成一个未知数据点与多个已知锚点数据之间的预测差异，可以得到原始回归问题的解。我们探讨孪生回归方法的不同方面：(1) 分解孪生回归算法的不同步骤，并考察它们对最终性能的贡献；(2) 考察其内在集成质量；(3) 将孪生神经网络回归与 K近邻回归结合，设计一种更准确、更高效的回归方法；(4) 开发一种简化的半监督回归方案。

关键词：人工神经网络、K近邻、随机森林、回归、半监督学习

1. 引言

回归是最通用、最常见的机器学习任务之一。在科学和工业的许多不同领域，专业人员都依赖各种方法，力求根据有限的训练数据，对新的数据点做出尽可能准确、可靠的预测。传统回归方法旨在推断输入特征与一个或多个目标变量之间的映射关系。孪生回归方法旨在解决一个对偶问题，即预测目标值之间的差异；随后，只需计算新未知数据点与多个锚点数据点之间的预测结果，即可得到原始回归问题的解。此过程通过单个训练后的模型生成预测集成，其准确度往往高于直接求解原始回归问题，并能以极低的成本实现半监督学习和不确定性估计 [1, 2, 3]。然而，其代价是难以扩展到大型数据集，因为有效训练数据集的大小会随原始训练数据集的大小呈二次方增长。

因此，在数据稀缺或获取成本高昂的领域，这些方法大有裨益。例如，房地产市场因城市而异，数据也很快会过时 [4, 5, 6]。另一个例子来自化学领域的计算：基于量子力学对化学体系进行模拟需要消耗大量计算资源 [7, 8]。

本文旨在为从业者提供一份使用 孪生回归 方法的指南，带领读者了解其优势与权衡，并尝试回答近年来尚未解决的大多数问题。

我们首先关注的问题是：为什么像求解对偶问题这样简单的技巧，能够比直接求解回归问题得到更准确的预测？虽然我们尚未能完全回答这一问题，但我们将孪生神经网络回归（TNNR）拆解为若干步骤，每一步都有望提升其相较于传统算法的性能。这些步骤包括通过配对训练数据点来扩大有效数据集，或对 TNNR 的预测采用不同的集成方法。此外，通过将 TNNR 的极端情形映射到 k近邻（k-NN）回归和普通人工神经网络（ANN），我们可以观察到孪生回归方法具有不同于传统回归的鲜明性能特征。

此外，我们还希望提高孪生回归方法的准确率。为此，我们基于对不同锚点的预测进行加权这一思路，对 TNNR 作出改进。由此，我们将 k近邻（kNN）与 TNNR 相结合，构建出一种准确率更高的回归方案。

文献 [2] 中提出的 TNNR 半监督回归框架专为基于神经网络的回归量身打造。该框架通过经修改的损失函数，对未知数据点施加一致性条件。本文末尾，我们将探讨如何把这一半监督学习方案的简化版本迁移到文献 [3]提出的随机森林（RF）孪生版本中。

在某些 项目中，应用具有强大记忆能力的神经网络非常重要 y

限制；小型芯片或自主系统中可能会出现这种情况 [9, 10]。在这些情况下，由于参数数量增加，存储机器学习模型集成通常非常低效。相比之下，使用孪生回归方法时，只需存储额外的锚点数据。

2. 先前工作

孪生回归方法所固有的成对比较思想，源自孪生神经网络。孪生神经网络旨在解决指纹识别或签名验证中出现的相似性分类问题 [11, 12]。孪生神经网络由两个结构相同且共享权重的神经网络组成，可将一对输入映射到潜在空间，并根据两者之间的距离确定其成对相似度。孪生回归方法同样接收一对输入，用于预测其标签之间的差值 [1]。

孪生神经网络回归 [1] 最初作为一种回归方法提出，用于解决预测一对输入数据点目标值之间成对差值这一对偶问题。与此同时，随机森林也独立发展出了相同的思路[3]。事实证明，这类回归框架具有多项优势：（1）能够非常高效地生成集成预测 [1, 3]。通常，在通过训练单个机器学习模型来构建集成的方法中，各项预测之间的相关性很强 [13, 14] ，因为它们经过微小扰动后便可相互转化。然而，在孪生回归方法中，各个集成成员之间的差异取决于输入数据点本身之间的距离。（2）在许多数据集上，孪生回归方法往往比底层基础算法更准确 [1, 3],；（3）一致性条件不仅可用于构建集成方差，还可用于构建不确定性估计器 [1, 3] ；（4）训练时可提供包含未标记数据点的环路，从而使该方法成为半监督回归算法 [2]。此外，（5）其内在的不确定性估计特性也适用于主动学习 [3]。

本文的一项核心贡献是将孪生神经网络回归与 k近邻回归相结合，以提高其准确率，优于标准的孪生神经网络回归。类似地，人工神经网络此前也曾在不同情境下与 k 近邻回归配合使用 [15, 16, 17]。

3. 回归问题的重新表述

回归问题可以表述如下：给定一个包含 n 个数据点 $X^{train} = (x_1^{train}, ..., x_n^{train})$ 及其对应目标值$Y^{train} = (y_1^{train}, ..., y_m^{train})$ 的带标签训练数据集，我们的任务是找到一个函数 f，使得对于数据流形上的所有数据点 x_i ， $f(x_i)$ 与 y_i 之间的偏差相对于预定义目标函数最小化。在本研究中，该目标函数为均方根误差 $L_{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(f(x_i) - y_i)^2}{n}}$。除非另有说明，所有性能度量

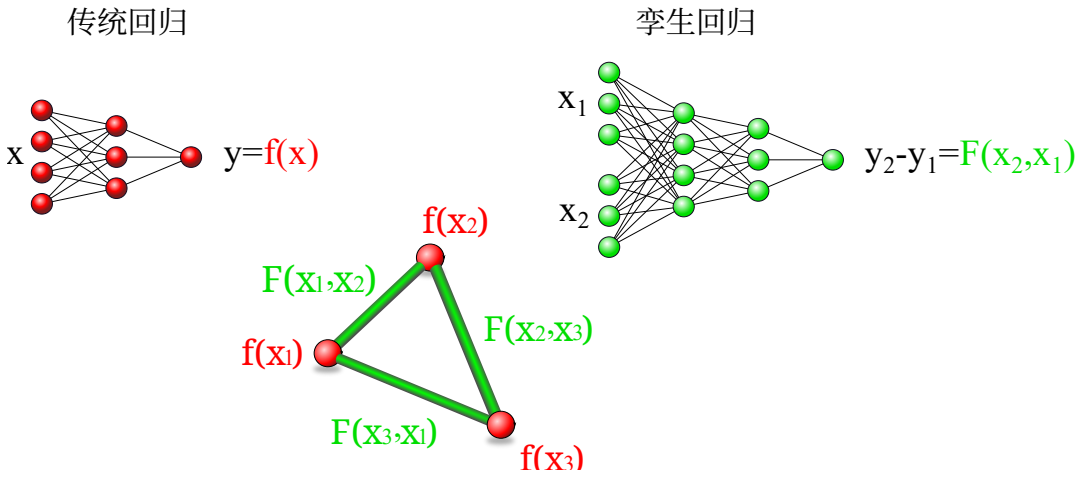


图 1： 回归问题的对偶形式：传统的回归问题解法是找到一个近似函数，将数据点 x 映射到其目标值 $f(x) = y$。孪生回归方法则解决将一对输入 x_1 和 x_2 映射到目标值之差 $F(x_2, x_1) = y_2 - y_1$的对偶问题。给定一个带标签锚点 (x_1, y_1)，所得函数便可用作原始回归问题 $y_2 = F(x_2, x_1) + y_1$ 的估计量。孪生回归方法必须满足环路一致性：沿每条环路的预测之和为零： $F(x_1, x_2) + F(x_2, x_3) + F(x_3, x_1) = 0$。

均在未知测试数据（X^{test}， Y^{test}）上进行评估。

孪生回归方法旨在解决原始回归问题的一种重新表述，如图 1 所示。对于每一对数据点 $(x_i^{train}, x_j^{train})$，我们训练一个回归模型来寻找一个函数 F，以预测差值

$$F(x_i, x_j) = y_i - y_j \quad . \tag{1}$$

这个函数 F 可用于通过 $y_i^{pred} = F(x_i, x_j) + y_j$ 构造原始回归问题的解，其中 (x_j, y_j) 是一个锚点，其目标值已知。每个训练数据点 $x_j^{train} \in X^{train}$ 都可以用作这样的锚点。通过对固定的未知数据点与不同锚点数据点之间的许多差值取平均，可以得到原始回归问题更准确的解估计

$$\begin{aligned} y_i^{pred} &= \frac{1}{n}\sum_{j=1}^{n}\left(F(x_i, x_j^{train}) + y_j^{train}\right) \\ &= \frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{2}F(x_i, x_j^{train}) - \frac{1}{2}F(x_j^{train}, x_i) + y_j^{train}\right) \quad . \end{aligned} \tag{2}$$

准确率的提升源于对不同锚点带来的噪声取平均，以及通过预测集成降低方差误差。先前的研究 [1, 3]建议将整个训练数据集用作锚点，因此，对于 y_i 的每一次预测，都会生成一个规模为训练集两倍的差值预测集成 $y_i - y_j$。

对偶形式的一大优势是可以通过包含多个数据点的环路来描述，如图 1 所示。与传统回归不同，

孪生回归方法的结果需要满足一致性条件。例如，对于任意三个数据点 x_1，x_2，x_3，沿闭环对预测值求和应为零： $F(x_1, x_2) + F(x_2, x_3) + F(x_3, x_1) = 0$。推理过程中，违反这些一致性条件会产生不确定性估计 [1, 3]。在训练阶段，对涉及未标记数据点的预测强制满足环路一致性，正是孪生回归方法成为半监督回归算法的关键。

虽然神经网络天生擅长学习线性函数，但其他算法（如随机森林）并非如此。因此， [3]提出用输入特征之差（x_i，x_j）→（x_i，x_j，$x_i - x_j$）来增强输入特征。有人可能会认为，这种改进与常见的数据增强类似；然而，传统机器学习算法无法使用这一特征，因为它需要两个不同的数据点。

4. 关于实验的说明

本文中的所有实验均在附录 A 所述的数据集上进行。由于只有基于神经网络的方法能够随着数据集大小的增加而良好扩展，因此这些方法使用完整数据集，其中 70% 用于训练，10% 用作验证集，20% 用作测试集。神经网络架构的详细信息见附录 C。随机森林，尤其是孪生随机森林，难以随着数据集大小的增加而扩展，因此仅从数据集中选取 100 个训练数据点，并以 100 个数据点组成测试集。随机森林不需要验证集，因为超=参数通过五折交叉验证进行优化。在附录 D 中可以找到随机森林实现的详细信息。所有实验均针对 25 种随机但固定的训练、测试以及（如适用）验证数据划分重复进行。

5. 集成性能

研究表明，孪生回归方法能够为回归问题提供准确的解 [1, 3], ，其性能可与当前其他最先进的算法相媲美，甚至更优，但在处理较大数据集时扩展性较差。这自然引出了一个问题：性能提升源自何处？将回归问题重新表述为预测目标值差异的对偶问题，可能有几个原因可以解释准确率的提升，包括有效训练集大小增加、预测的内部集成（详见附录 B 中的说明），或求解的是不同问题这一特性。下文将以 TNNR 为例考察这些原因；不过，我们认为所得结论同样适用于其他基线算法。

我们先来讨论不同类型的集成方法及其对准确率的影响。图2展示了若干项实验的结果，这些实验考察了不同类型的人工神经网络回归和 TNNR 的表现。对于每个数据集，基线结果由蓝色实心水平线表示，该线代表应用标准

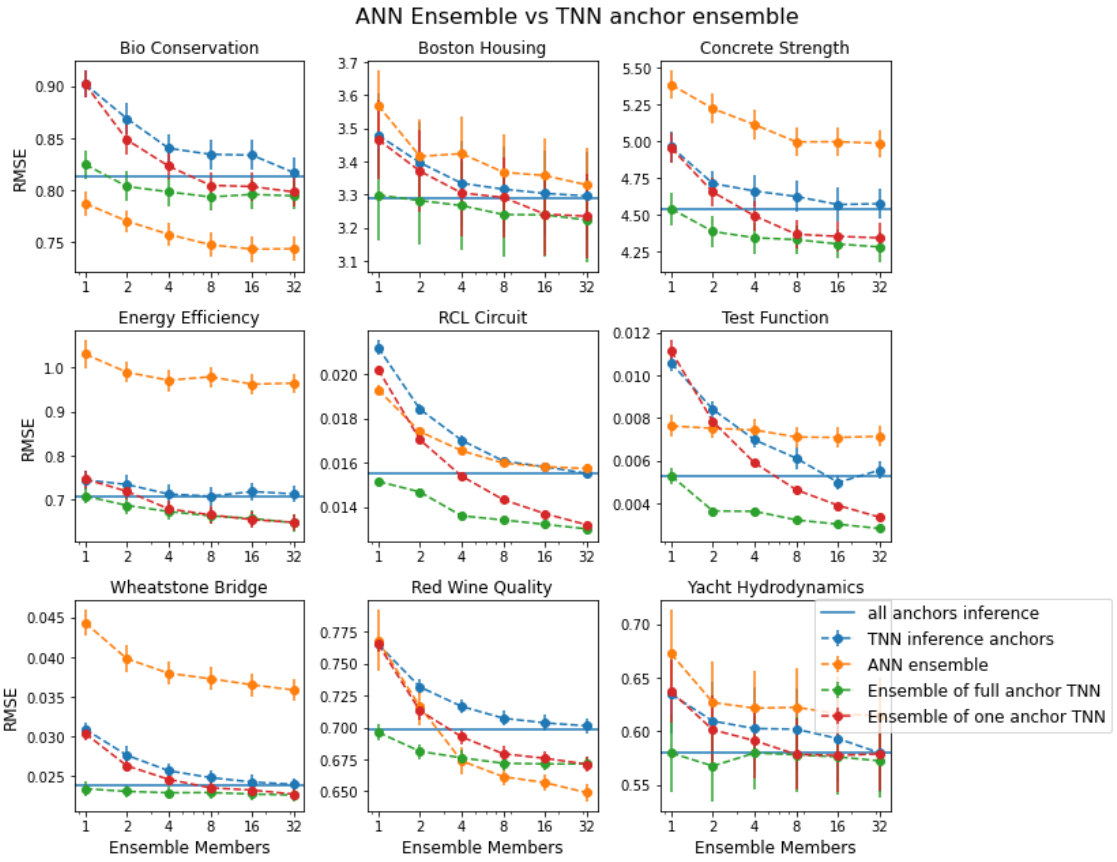


图 2：图中以均方根误差（RMSE）为指标，比较了不同人工神经网络集成与 TNNR 集成成员数量之间的关系（虚线所示）。集成可以通过训练多个模型创建，也可以通过对不同锚点进行 TNNR 推理来创建。蓝色实线表示在所有可能的训练配对上训练并评估一个 TNNR 模型，再使用所有可能的锚点进行预测。蓝色虚线表示在推理阶段改变锚点数量；随着推理锚点增加至整个训练集，该曲线趋近于蓝色实线。橙色虚线表示传统人工神经网络集成，其中多个人工神经网络分别独立训练。绿色虚线表示由独立训练的 TNNR 模型组成的集成；当集成大小为 1 时，这与蓝色实线等效。红色虚线表示独立训练的 TNNR 模型，每个模型在每次预测时仅使用一个推理锚点。

完整锚点 TNNR 后的测试均方根误差；橙色线最左侧的点则代表应用单个人工神经网络的结果，这证实了 TNNR 的均方根误差几乎总是低于人工神经网络回归。

为了比较传统人工神经网络与 TNNR，我们注意到，训练完成后，可以针对每个锚点将 TNNR 映射为一个人工神经网络，因为人工神经网络回归和 TNNR 使用˜相同的内部架构。对于每个固定的 $x_j \in X^{train}$，人工神经网络 f 定义如下

$$\tilde{f}_j(x_i) := F(x_i, x_j) + y_j \tag{3}$$

˜ x_j 的特征会改变 f_j 的权重，而 y_j 则被输出神经元˜吸收进其偏差中。此时，每个 f_j 与一个等效 ANN 之间唯一的区别，就是权重优化所采用的过程。

表1： 人工神经网络（ANN）回归与人工神经网络回归集成相比所获得的测试均方根误差最佳估计值。我们根据均方根误差的标准误来确定对其估计值的置信度。我们使用 70% 的可用数据进行训练，10% 的数据用于验证，20% 的数据用于测试。

	单个 ANN	32 个 ANN 集成	Gain
生物保护（BC）	0.7874±0.012	0.7438±0.0118	5.53%
波士顿房价(BH)	3.5695±0.1051	3.3291±0.1137	6.73%
混凝土强度（CS）	5.3829±0.0964	4.9842±0.0938	7.41%
能源效率（EE）	1.0310±0.0313	0.9649±0.0224	6.42%
RCL电路（RCL）	0.0193±0.0002	0.0157±0.0002	18.41%
测试函数（TF）	0.0076±0.0005	0.0071±0.0005	6.35%
惠斯通电桥（WSB）	0.0443±0.0016	0.0359±0.0014	19.1%
红葡萄酒质量(WN)	0.7681±0.024	0.6487±0.0068	15.54%
游艇水动力学(YH)	0.6723±0.0406	0.6143±0.0357	8.63%

表2： 孪生神经网络回归（TNNR）获得的测试均方根误差的最佳估计值。我们衡量了在推理阶段使用单个锚点与使用所有锚点之间的改进。此外，我们还将后者与完整锚点 TNNR 的集成进行了比较。

	1 个锚点 TNNR	所有锚点 TNNR	Gain	32 个 TNNR 的集成	Gain
BC	0.9021±0.0131	0.8140±0.0149	9.77%	0.7947±0.0134	2.37%
BH	3.4793±0.1262	3.2897±0.1293	5.45%	3.2232±0.1256	2.02%
CS	4.9602±0.1073	4.5385±0.1124	8.5%	4.2791±0.0995	5.71%
EE	0.7445±0.0224	0.7071±0.0229	5.02%	0.6468±0.0196	8.53%
RCL	0.0212±0.0003	0.0155±0.0002	26.6%	0.0130±0.0001	16.23%
TF	0.0106±0.0004	0.0053±0.0004	50.24%	0.0028±0.0001	46.41%
WSB	0.0309±0.0009	0.0239±0.0009	22.81%	0.0227±0.0008	5.15%
WN	0.7654±0.0059	0.6985±0.0064	8.75%	0.6713±0.0055	3.89%
YH	0.6344±0.0363	0.5798±0.0362	8.62%	0.5723±0.0336	1.29%

这使我们能够直接比较人工神经网络集成与 TNNR 在推理过程中使用多个锚点生成的隐式集成。我们通过图2中的橙色和蓝色虚线考察这两种模型的结果。随着集成大小增加，两条曲线的均方根误差都在下降，但令人清醒地认识到，TNNR 集成与人工神经网络集成并不等效：它们的斜率既不一致，也不会收敛到相近的均方根误差。

我们刚才对所有集成成员使用了同一个训练好的 TNNR 模型，同时从头开始训练每个人工神经网络模型。如果针对每个单独的锚点重新训练 TNNR，会发生什么？这些实验的结果以红色虚线表示。由于重新训练提高了集成多样性，红线始终位于蓝线下方

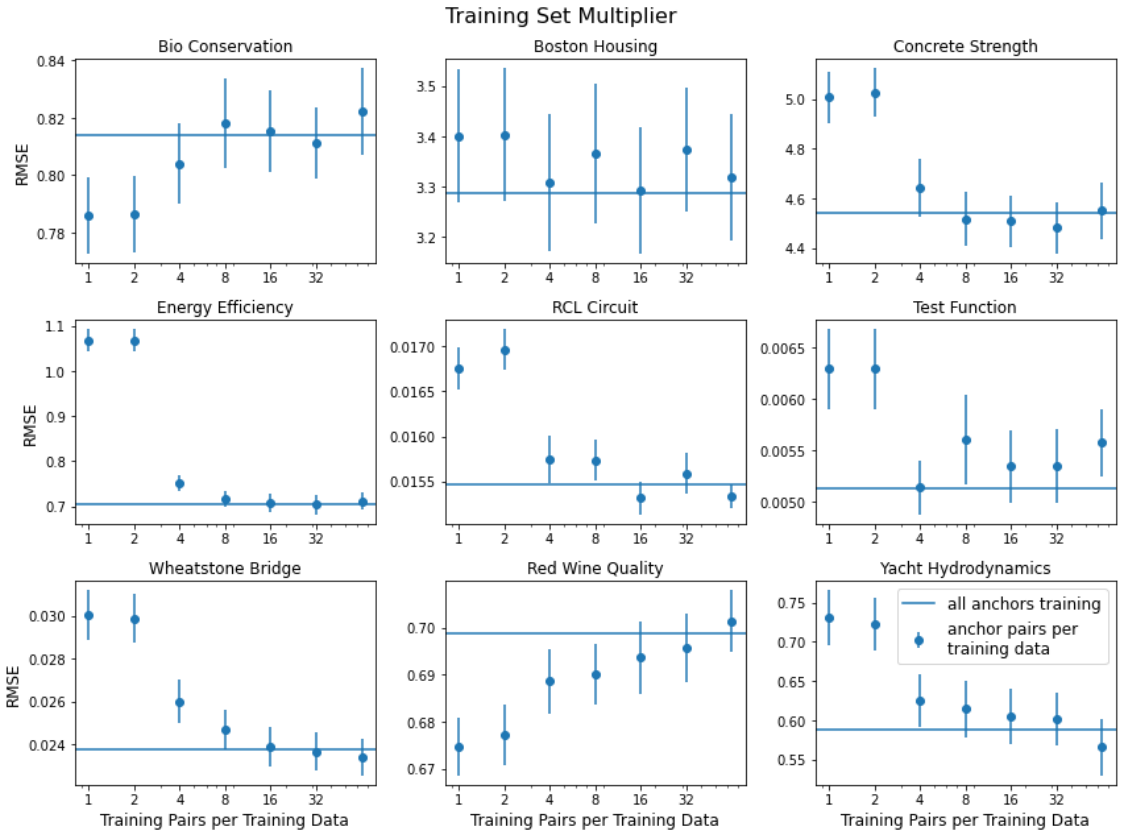


图3： 训练时使用的不同配对数量会影响 TNNR 的性能，性能以均方根误差衡量。蓝色实线表示使用所有可能的训练配对来训练 TNNR。我们将其与使用随机选取但固定的一组配对来训练 TNNR 的结果进行比较；后一种方法仍将所有训练数据点用作推理锚点。训练集乘数表示与原始未配对训练集相比，纳入了多少个配对。

。此外，我们可以看到，随着锚点数量增加，这些独立训练的 TNNR 性能提升得更快。如果资源充足，还可以构建由不同 TNNR 模型组成的集成，每个模型都能访问所有锚点，如绿色线所示。在九个数据集中，有七个数据集的结果明显最佳。我们注意到，对于重新训练的单锚点 TNNR（红线），当集成成员超过 32 个时，其表现会逐渐接近绿色线。这说明，通过多个锚点和多个模型实现的完整集成多样性，可以彼此独立地获得。

表1和表2更定量地展示了 TNNR 集成方法的优越性。由多个 TNN 组成的锚点+直接集成所带来的百分比提升幅度，远大于传统人工神经网络集成所带来的提升。

6. 有效训练集大小

相较于传统回归分析，另一项改进是训练集大小有所增加，这是通过将每个训练数据点与

表3： 采用最近邻孪生神经网络回归（NNTNNR）得到的测试均方根误差最佳估计值。

	TNNR	神经网络推理	Gain	神经网络训练+推理	Gain
BC	0.8234±0.0144	0.8162±0.0155	0.87%	0.8133±0.0152	1.23%
BH	3.3104±0.1202	3.2898±0.1202	0.62%	3.3563±0.1161	-1.39%
CS	4.4731±0.1242	4.4458±0.1091	0.61%	4.4290±0.1174	0.99%
EE	0.7156±0.0204	0.7056±0.0193	1.4%	0.6825±0.0216	4.63%
RCL	0.0158±0.0002	0.0151±0.0003	3.98%	0.0140±0.0002	11.33%
TF	0.0050±0.0001	0.0028±0.0002	43.65%	0.0021±0.0003	57.26%
WSB	0.0233±0.0006	0.0224±0.0009	4.06%	0.0236±0.0009	-1.01%
WN	0.6998±0.006	0.6951±0.0062	0.68%	0.6944±0.006	0.77%
YH	0.5977±0.0344	0.5184±0.0331	13.26%	0.5009±0.0333	16.19%

其他所有训练数据点配对来构建训练集实现的。这会将大小为 n 的训练集转变为大小为 n^2的成对训练集。在本节中，我们将衡量配对数量的增加是否会带来准确率的提升。

为了回答这个问题，我们来看图3中的几条曲线。该图比较了增加有效成对训练数据集对准确率的影响。为此，我们将训练集乘数定义为：从原始训练集中生成配对训练集时所创建的配对数量。训练集乘数为 1，意味着每个训练数据点只与另一个随机选取（不放回）、但固定的数据点配对（平均而言，这意味着每个训练数据点会被使用两次）。将训练集乘数增加到训练集的大小，就会逐渐趋近于孪生回归方法的标准形式。我们可以看到，除了两个数据集外，在所有数据集中，增加训练集乘数都会提升 TNNR 的表现。更准确地说，训练集乘数为 $\approx 8-16$ 似乎就足以达到标准 TNNR 的准确率。需要注意的是，在两个数据集——生物保护（BC）和红葡萄酒质量（WN）——上，增加训练集乘数反而会降低表现。这与其他算法胜过 TNNR 的情况一致（图2、图5），也表明 TNNR 可能不适用于这类回归任务。使用 TNNR 的数据科学家可以检查训练集乘数：如果发现随着乘数增加，准确率反而下降，就可以排除 TNNR 作为最优回归算法的可能性。

7. 最近邻TNNR

在这篇稿件中，我们提出了一种新的回归算法，该算法结合了K近邻回归和TNNR。当然，它也可以应用于各种基线孪生回归算法。在标准的孪生回归方法中，模型会学习预测任意两个数据点的目标值之差

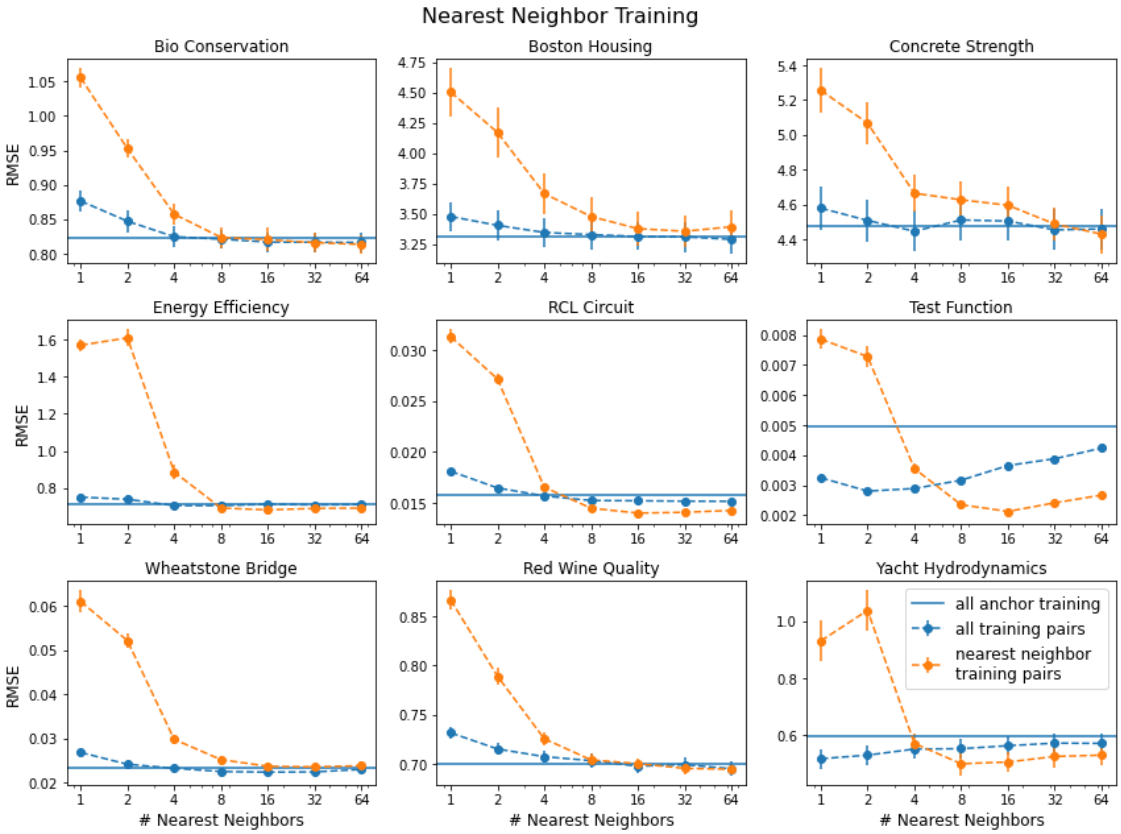


图4： 最近邻配对对 TNNR 的影响，以均方根误差衡量，并显示使用的最近邻数量（如适用）。蓝色实线表示原始TNNR的性能。蓝色虚线表示使用所有可能的配对进行训练、但推理时仅使用最近邻锚点的 TNNR 的结果。橙色虚线表示训练和推理时均仅使用最近邻配对的结果。

。然后，通过对新数据点与所有锚点数据之间目标值差的近似值取平均，利用该模型生成集成预测，见（2）。然而，并非所有这些锚点数据对预测都同等重要。因此，在本节中，我们将锚点限定为最近邻。为此，我们将 NN(i, m) 定义为训练集 $x_i \in X^{train}$中数据点 $x_i \in X$ 的 m 个最近邻的集合，并据此改写预测公式：

$$y_i^{pred} = \frac{1}{m} \sum_{j \in \mathrm{NN}(i,m)} \left(F(x_i, x_j^{train}) + y_j^{train} \right) \tag{4}$$

虽然我们已定义在推理阶段使用最近邻进行预测，但究竟是训练模型预测一般数据点的目标值之间的差异更好，还是预测推理阶段对应相同数量最近邻的相邻数据点之间的差异更好，仍是一个开放问题。图4比较了不同的训练版本，其中基线由标准

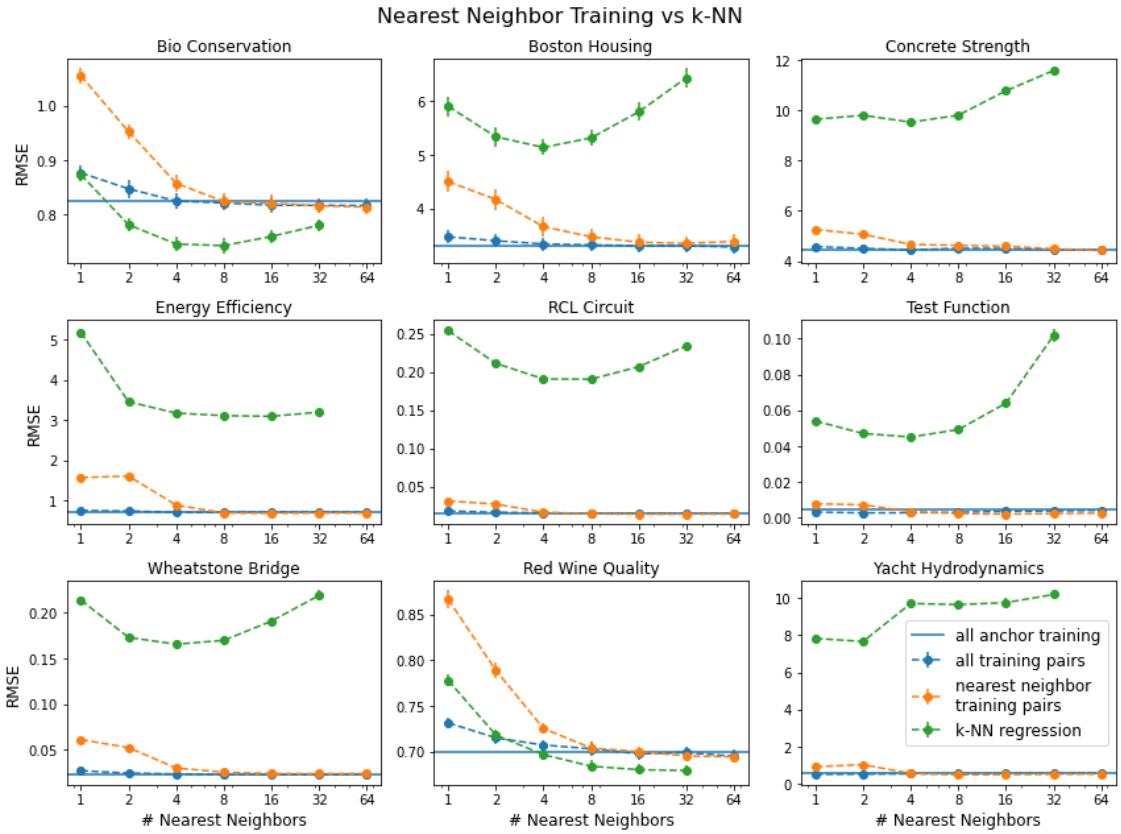


图5： 比较不同近邻训练对数量下的 k近邻回归与 TNNR：虚线表示均方根误差随邻居数量的变化。蓝色实线表示标准 TNNR，即训练和推理时均可使用所有可能的配对。蓝色虚线表示仅在推理时将配对限制为最近邻。橙色虚线表示在训练和推理时均将配对限制为最近邻。绿色虚线表示 k近邻。

TNNR确定。我们强调，这两个版本的训练在推理阶段选择最近邻时遵循相同的原则。推理时仅使用最近邻作为锚点时，可以看到，在9个数据集中有7个数据集的两种训练版本都不如传统 TNNR，而使用所有可能的数据点对进行训练，则优于仅使用相邻训练数据点进行训练。随着最近邻数量增加，情况会发生变化。在所有数据集中，随着最近邻数量增加至训练集大小，两种最近邻TNNR最终都会收敛到标准 TNNR。在9个数据集中有7个存在一个最佳点：使用最近邻训练的最近邻TNNR在大约16至64个邻居时表现更佳；其中3个数据集的最近邻训练优势非常明显，在TF数据集上均方根误差降低了 $\approx 60\%$，见表3。值得注意的是，该数据集的噪声为零。

8. TNNR 与 k近邻

一个自然的问题是，最近邻 TNNR（NNTNNR）与 k近邻回归有何关联。通过设置 $F(x_i, x_j) \equiv 0$，可以将最近邻 TNNR 与 k近邻回归联系起来，然后

$$y_i^{pred} = \frac{1}{m} \sum_{j \in NN(i,m)} \left(\underbrace{F(x_i, x_j^{train})}_{0} + y_j^{train} \right) = \frac{1}{m} \sum_{j \in \mathrm{NN}(i)}^{m} y_j^{train} \tag{5}$$

假设 $F(x_i, x_j)$ 对 k近邻回归的贡献微乎其微，那么我们会看到 NNTNNR 的性能与之大体相似。为检验这一说法，我们在图5中展示了 k近邻和 NNTNNR 的表现。从图中可以清楚地看到，在 9 个数据集中，有 7 个数据集上的 NNTNNR 表现远胜于 k近邻回归。不过，有两个数据集（即 BC 和 WN）是 k近邻表现更好。再次指出，这些数据集恰好也是预期的 TNNR 机制失效的地方（图3），同时也是人工神经网络集成表现优于 TNNR 的地方（图2）。此外，最优 TNNR 锚点的数量远多于 k近邻中最优的邻居数量。

9. 重新表述的益处

在讨论了集成和有效训练集大小增加所带来的影响之后，我们现在终于有了部分回答以下问题所需的工具：重新表述为对偶问题本身，是否有助于提高孪生回归方法的准确率。我们在（3）中将 TNNR 与普通人工神经网络联系起来，并在（5）中将 NNTNNR 与 k近邻回归联系起来。如果 TNNR 只是经过美化的人工神经网络或 k近邻回归，那么它的性能在定性上应与神经网络或 k近邻相似。然而，通过将 TNNR 与图2中的人工神经网络以及图5中的 k近邻进行比较，我们可以清楚地看到，TNNR 的性能特征截然不同：在相同的 9 个数据集中，有 7 个数据集上的表现优于人工神经网络和 k近邻，而在其余两个数据集上的表现则不如二者。正如我们从图3中对训练集大小增加所带来影响的测试结果中所知，这两个数据集正是增加数据集大小会对准确率产生不利影响的数据集，这表明 TNNR 机制失效，而人工神经网络和 k近邻仍能正常运行。所有这些事实都支持这样一个结论：在大多数数据集中，重新表述为对偶问题本身往往会对准确率产生积极影响。

10. 缩小高精度网络

通常需要将训练完毕的神经网络存储在内存限制严格的硬件上。这类硬件可能是允许对量子点进行自动调谐的芯片 [9]，也可能用于自动驾驶车辆 [10]。在这些情况下，需要权衡准确率和内存需求。神经网络集成

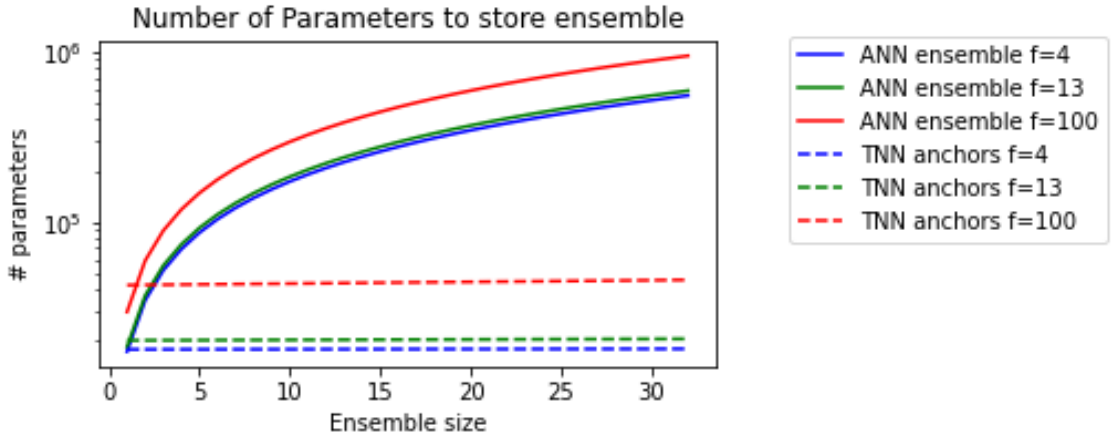


图6： 根据所存储的参数数量与集成大小，存储真实集成所需的内存需求。所使用的架构与本稿件中用于生成结果的架构相对应。不同的架构会导致定量上不同的图，但在定性上表现相似。实线表示针对描述每个数据实例的不同数量特征 f，存储由独立训练的人工神经网络模型组成的集成所需的内存需求。虚线表示通过存储一个TNNR模型和具有代表性数量的锚点来生成集成时所需的内存，这些锚点可以组合起来生成集成。

往往比单个神经网络更准确，但存储它们需要为每个集成成员额外增加线性增长的存储容量。TNNR 为此问题提供了一种巧妙的解决方案，因为要存储预测集成，只需存储一组训练好的权重和偏置，以及每个集成成员对应的一个锚点数据点。我们选择的神经网络架构能够在所考察的全部九个数据集上产生准确的结果。因此，我们选择了包含两个隐藏层的架构，每层有 128 个神经元。图 6 展示了在不同输入数据特征规模下，传统 ANN 集成和 TNNR 集成所需存储的参数数量。可以看出，在所有情况下，为达到与人工神经网络相近的集成大小，存储 TNNR 参数和锚点所需的容量都少得多。当然，在实际应用中，最优神经网络架构及其参数数量会因问题而异，这意味着我们对内存需求的定量分析未必适用于其他问题。不过，只要特征规模小于模型的参数数量，这一趋势在定性上仍然成立。

11. 使其他模型实现半监督

本节将探讨一个简单的框架，用于以半监督方式训练任意孪生回归方法。该思路基于 [2] 针对TNNR提出的半监督回归方法。正如图1所示，对偶形式要求机器学习模型预测差异 $F(x_i, x_j) = y_i - y_h$，即目标值之间的差异，而不是目标值 $f(x) = y$本身。这种形式的一个优势是，正确的

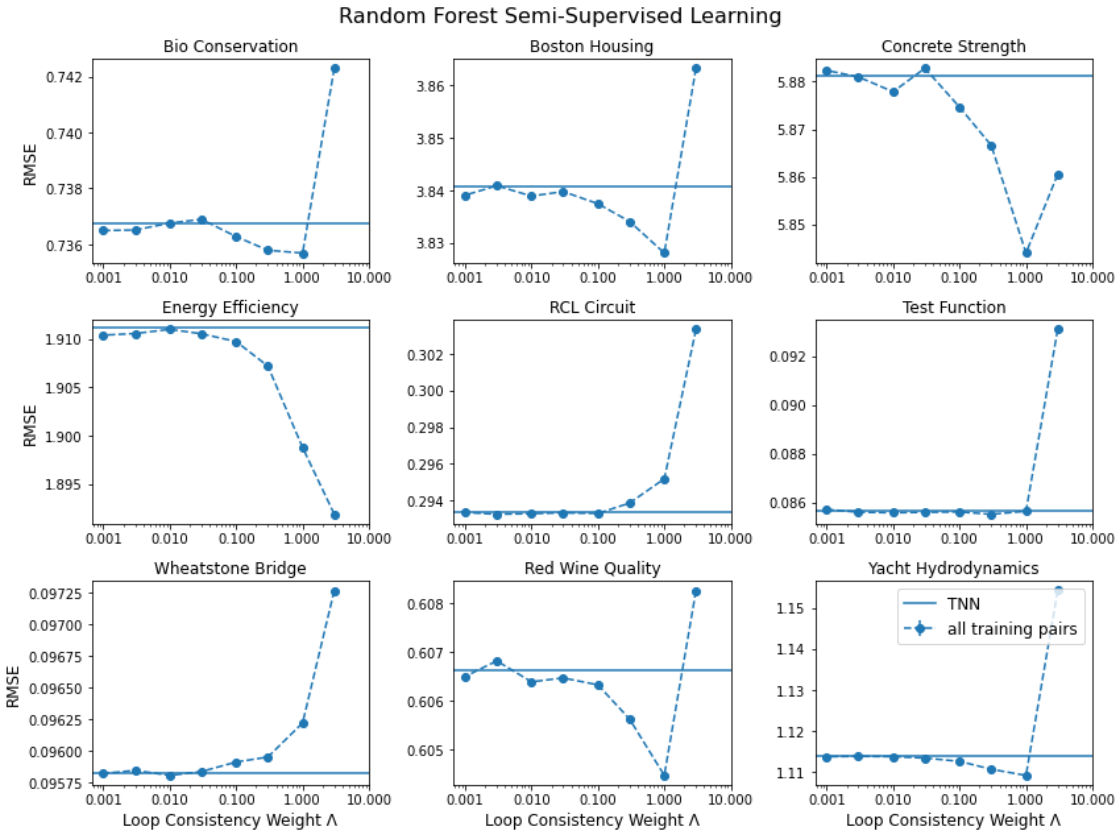


图7： 使用随机森林进行传导式半监督学习。随机森林中加入了包含未标记数据点的环路。环路对训练过程的影响程度由 Λ 衡量。与纯监督学习（蓝色实线）相比，调整 $\Lambda \approx 1$ 可降低几乎所有数据集上的均方根误差。

表4： 半监督随机森林得到的测试均方根误差最佳估计值。

	监督式随机森林	半监督随机森林	提升幅度
生物保护	0.7367±0.0081	0.7357±0.0078	0.14%
波士顿房价	3.8408±0.1322	3.8281±0.1356	0.33%
混凝土强度	5.8813±0.2242	5.8440±0.2276	0.63%
能源效率	1.9112±0.0417	1.8919±0.0411	1.01%
RCL电路	0.2934±0.0068	0.2932±0.0068	0.05%
测试函数	0.0857±0.0022	0.0855±0.0021	0.19%
惠斯通电桥	0.0958±0.0025	0.0958±0.0022	0.02%
红葡萄酒质量	0.6066±0.0083	0.6045±0.0084	0.35%
游艇水动力学	1.1138±0.0383	1.1091±0.0384	0.42%

sol解会满足环路一致性 $F(x_1,\ x_2) + F(x_2,\ x_3) + F(x_3,\ x_1) = 0$。

因此，我们提出了以下适用于所有孪生回归方法的算法：首先，我们使用有标签训练数据训练回归模型。随后，利用该模型预测从无标签数据集中随机抽取的环路上各目标值之间的差异。对于每个环路，使用一个调整量 a ，其定义为

$$a = F(x_1, x_2) + F(x_2, x_3) + F(x_3, x_1) \tag{6}$$

然后用它为环路中每个 $x_i,\ x_j$组合提出一个标签 $y_{ij} = F(x_i,\ x_j) - \Lambda \times a$。此处， Λ 是循环权重超参数。随后，将无标签数据集及其提出的标签加入有标签训练集，并据此重新训练模型。图 1 进一步展示了该算法。

我们将这一思路应用于 [3], 中提出的成对／孪生随机森林回归，该方法最初旨在解决化学领域小型数据集上的回归问题。由于随机森林在大型数据集上的扩展性不佳，我们将数据集限制为包含 100 个训练数据点和 100 个测试数据点。训练过程的细节见附录 D。

在应用半监督学习策略之前，我们先确认孪生随机森林回归适用于本文所用的测试基准，该基准由九个数据集（附录 A）组成。相应结果见表 D1。在三个数据集（BC、EE、WN）上，孪生随机森林的表现与传统随机森林相当，或略逊一筹；在四个数据集（BH、CS、RCL、YH）上，其表现有所提升；在另外两个数据集（TF、WSB）上，其表现大幅提升，均方根误差降低了 35% 以上。

确认孪生随机森林表现更优后，我们以传导式方式应用半监督学习框架。传导式意味着将测试数据用作无标签训练数据。这与归纳式半监督学习不同，后者会将无标签训练数据与最终测试数据分开。图 7 展示了循环权重 Λ 取不同值时的最终结果。我们可以清楚地看到，选择最优的$\Lambda \approx 1$ 可使九个数据集中的六个数据集的均方根误差降低。不过，如表4所示，半监督学习带来的相对提升幅度很小，且大多数情况下低于 1%。将这些结果与 [2], 中相同数据集上的其他半监督回归算法进行比较，可以发现，这种提升明显低于半监督 TNNR，略低于使用神经网络的协同训练。

12. 负面结果

本节简要讨论我们在实验过程中尝试过、但未能持续改进孪生回归方法的不同思路。

在探索孪生回归方法推理阶段锚点的理想权重时，一个直观的想法是尝试纳入内在

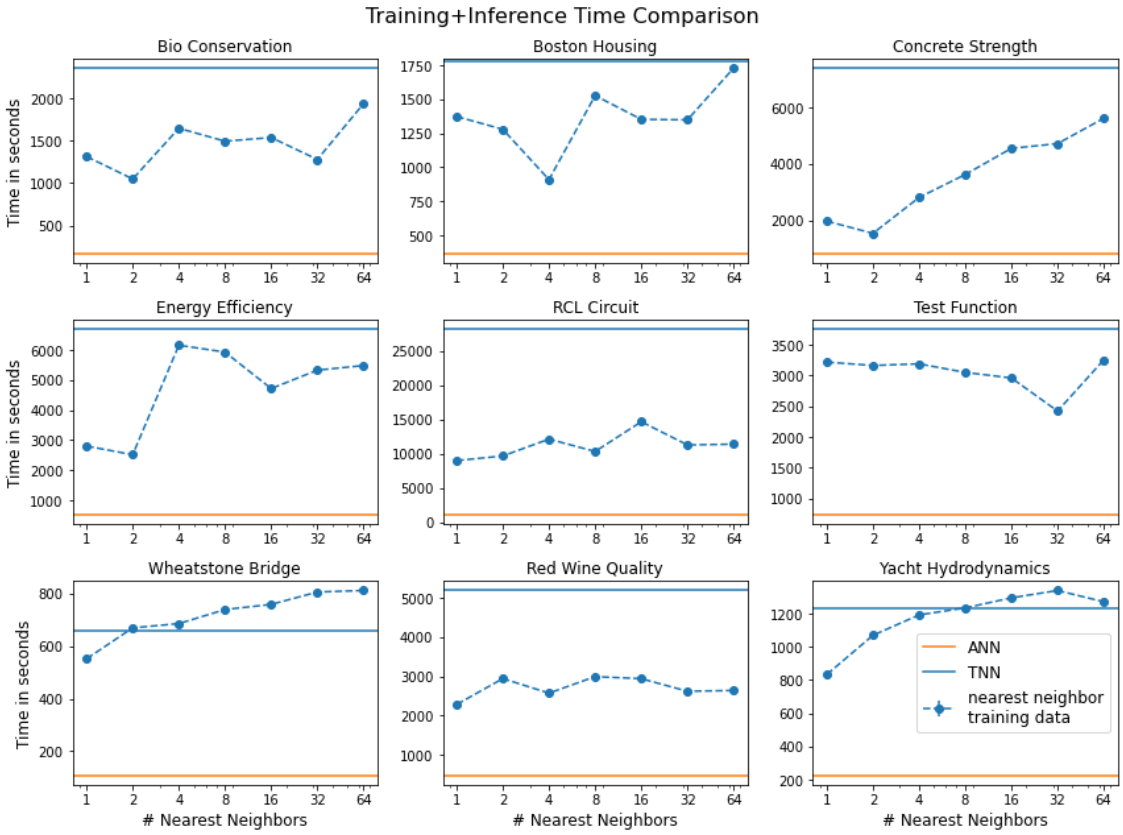


图8： TNNR 不同版本与人工神经网络回归的训练和推理时间（秒）对比。橙色实线表示人工神经网络的训练和推理时间。蓝色实线表示原始TNNR的相同时间。蓝色虚线表示将可能的训练/推理配对限制为最近邻时的情况，x 轴对应邻居数量。

不确定性度量 [1]。这些度量包括集成标准差和环路一致性违例。应赋予不确定性度量较低的锚点更高的权重，而不确定性度量较高的锚点则应获得较低的权重。尽管我们观察到了一些益处，但无法持续证明这一过程能以具有统计显著性的方式提高准确率。我们认为，与内在一致性度量无关的其他不确定性度量或许更合适，例如高斯过程中的相关度量。

我们最初在制定将半监督学习框架从 [2] 推广到其他算法的策略时，计划采用一种迭代算法。训练底层的孪生回归算法后，模型将对未知数据点预测标签。随机抽取包含这些数据点的环路，就可以检查环路一致性。随后，将这些未知数据点及其标签加入训练数据集；这些标签以原始预测为基础，沿着满足回路条件的方向稍作调整。我们的想法是重复这一过程，迭代优化标签。然而，事实证明，这一过程往往无法在 $\Lambda > 1/3$ 次迭代内收敛，或者最终收敛到次优解，其效果甚至不如最初的监督版本。

将 k近邻回归与 TNNR 结合，也旨在减少

计算时间。正如 [1], 所探讨的那样，随着数据集规模增大，孪生回归方法的训练时间会显著变差，这主要是由于配对数据点导致有效数据集大小增加。对于许多基线算法而言，数据集大小与训练时间之间的关系很明确；但对于神经网络，这种关系则不太明了。由于神经网络的训练时间随训练集规模增加而变化得非常有利，我们重点研究 TNNR，以测试在训练阶段仅使用最近邻配对能否缩短训练时间。从图 8 可以看出，在大多数数据集上，计算成本都有降低的趋势。然而，训练时间的变化幅度太小且不够稳定，因此不能仅凭这一点就认为应当使用 NNTNNR 而非传统 TNNR。

13. 结论

孪生回归是一种简单且灵活的框架，可通过内在集成和半监督学习提升小型至中型数据集上的性能。在本文中，我们回答了有关孪生回归框架性质的若干问题。此外，我们还提出了多项改进措施，进一步提升孪生回归方法已达到的最先进性能。

我们比较了传统人工神经网络与 TNNR 的集成行为。为此，我们在推理阶段将单锚点 TNNR 映射为等效的人工神经网络模型。通过在图2中可视化这些考察结果，我们可以看到，单锚点 TNNR 集成的性能在集成成员数达到约 32 个时，逐渐趋近于全锚点 TNNR 集成。增加锚点数量不会带来额外增益。这表明，TNNR 的内在集成多样性是通过针对每个锚点重新训练网络所能实现的多样性的子集。表1和表2显示，结合锚点+直接集成多个 TNNR 所带来的提升，远大于传统人工神经网络的集成。这解释了 TNNR 在人工神经网络回归方面优于 ANN 的一个原因。

此外，我们还考察了通过配对数据点来增大训练集大小对 TNN 准确率的影响，见图3。总体而言，每个训练数据点的配对数越多，均方根误差越低。然而，令我们意外的是，并非所有数据集都能从这种配对中受益：在两个数据集（BC、WN）上，如果每个训练数据点只允许配对一次，TNNR 的均方根误差最低。将这一异常现象与图2和图5中的结果进行比较，可以发现，它恰好出现在 k近邻回归和人工神经网络集成优于 TNNR 的数据集上，这表明 TNNR 的性能提升因素失效了。观察这些数据集的属性后，我们发现，孪生回归方法似乎在连续型数据集上表现最佳，其中标签可以通过确定性函数近似得出。

我们还指出了 TNNR 的另一项优势：当无法存储大量参数，但又希望保留集成方法的优势时，TNNR 只需存储

一个模型和一些锚点，就能生成预测集成；这通常比存储多个人工神经网络模型所需的空间小得多，见图 6。

在探索 TNNR 推理阶段锚点的理想加权方式时，我们发现，最近邻预测往往能得到最准确的结果。因此，我们开发了最近邻 TNNR（NNTNNR），将 k 近邻算法与 TNNR 相结合。NNTNNR 有两个版本：一个在训练和推理阶段都遵循最近邻，另一个仅在推理阶段限制为最近邻。限制训练时使用的邻居为最近邻，通常能带来略好的结果，见图4。两个版本的表现都优于标准 TNNR，尤其是在低噪声数据集上，见表3。需要注意的是，NNTNNR 并非只是对 k近邻回归的小幅改进，因为当锚点数量或最近邻数量变化时，它呈现出截然不同的性能特征，如图5所示。

我们设计了一种基于强制环路一致性的半监督回归框架，可应用于任何孪生回归算法；不过，我们只针对随机森林进行了测试。从图7可以看出，与监督式对应方法相比，该方法带来了明显的改进。然而，均方根误差的降低幅度相对较小，几乎总是不到1%，见表4。将这些结果与 [2], 中相同数据集上的其他半监督回归算法进行比较，可以看出，这种改进明显不及半监督 TNNR，且略逊于采用神经网络的协同训练。

支持本文的代码可在 [18] 获取。

14. 致谢

感谢 Zurab Jashi 对随机森林代码提供的帮助。本研究由 Mitacs 和 Homes Plus Magazine Inc. 通过 Mitacs Accelerate 计划提供支持。我们还感谢 Compute Canada 提供计算资源。感谢加拿大国家研究委员会与 Perimeter 合作开展 PIQuIL 项目。Perimeter Institute 的研究部分由加拿大政府通过加拿大创新、科学和经济发展部提供支持，并由安大略省通过经济发展、就业创造与贸易部提供支持。

附录 A. 数据集

表 A1： 数据集

Name	Key	Size	特征	Type
生物浓度	BC	779	14	离散 ，连续
波士顿房价	BH	506	13	离散，连续
混凝土强度	CS	1030	8	连续
能源效率	EF	768	8	离散，连续
RCL电路电流	RCL	4000	6	连续
测试函数	TF	1000	2	连续
红葡萄酒质量	WN	1599	11	离散、连续
惠斯通电桥电压	WSB	200	4	连续
游艇水动力学	YH	308	6	离散

由该方程创建的测试函数（TF）数据集

$$F(x_1, x_2) = x_1^3 + x_1^2 - x_1 - 1 + x_1 x_2 + \sin(x_2) \tag{A.1}$$

以及零噪声。

RCL 电路电流数据集 (RCL) 的输出是流经 RCL 电路的电流，其模型由以下方程给出

$$I_0 = V_0 \cos(\omega t)/\sqrt{R^2 + (\omega L - 1/(\omega C))^2} \tag{A.2}$$

并添加了均值为 0、标准差为 0.1 的高斯噪声。

惠斯通电桥电压 (WSB) 的输出是测得的电压，其方程如下

$$V = U(R_2/(R_1 + R_2) - R_3/(R_2 + R_3)) \tag{A.3}$$

并添加了均值为 0、标准差为 0.1 的高斯噪声。

附录 B. 偏差-方差权衡与集成方法

在回归问题中，我们的任务是通过估计函数 $f(x) = y$，找出尚未标记数据点的真实标签。给定有限的训练数据集 D，我们将这个近似记为 $f(x; D)$。预期均方误差可分解为三个误差来源：偏差误差 $\text{Bias}_D[f(x; D)]$,、方差误差 $\text{Var}_D\left[\hat{f}(x; D)\right]$，以及数据集的固有误差 σ。

$$\text{MSE} = \text{E}_x \left\{ \text{Bias}_D[\hat{f}(x; D)]^2 + \text{Var}_D\left[\hat{f}(x; D)\right] \right\} + \sigma^2. \tag{B.1}$$

如果我们用由两个函数组成的集成 $f(x;\ D)\ =\ 1/2 f_A(x;\ D)\ +\hat{}1/2 f_B(x;D)$ 替换估计量，并且这两个函数的偏差和方差均与原估计量相同，那么我们可以将均方误差分解为

$$\mathrm{MSE} = \mathrm{E}_x \left\{ \mathrm{Bias}_D[1/2\hat{f}_A(x;D) + 1/2\hat{f}_B(x;D)]^2 + \mathrm{Var}_D\left[1/2\hat{f}_A(x;D) + 1/2\hat{f}_B(x;D)\right] \right\} + \sigma^2 \tag{B.2}$$

$$= \mathrm{E}_x \left\{ \mathrm{Bias}_D[\hat{f}(x;D)]^2 + \mathrm{Var}_D\left[1/2\hat{f}_A(x;D)\right] + \mathrm{Var}_D\left[1/2\hat{f}_B(x;D)\right] \right. \tag{B.3}$$

$$\left. + 2\,\mathrm{Cov}_D\left[1/2\hat{f}_A(x;D), 1/2\hat{f}_B(x;D)\right] \right\} + \sigma^2 \tag{B.4}$$

$$= \mathrm{E}_x \left\{ \mathrm{Bias}_D[\hat{f}(x;D)]^2 + 1/2\,\mathrm{Var}_D\left[\hat{f}_A(x;D)\right] + 1/2\,\mathrm{Cov}_D\left[\hat{f}_A(x;D), \hat{f}_B(x;D)\right] \right\} + \sigma^2 \tag{B.5}$$

$\hat{}\hat{}\ \ f_A(x;D)$ 和 $f_B(x;D)$ 的相关性越低，方差与协方差之比就越小。因此，由相关性较弱的集成成员组成的集成能够绕过偏差-方差权衡，从而降低均方误差。通过归纳，这一论证可以推广到规模更大的集成。

附录 C. 神经网络架构

我们的传统神经网络回归方法和孪生神经网络回归方法均采用相同的架构，并使用 TensorFlow 库 [19] 构建。它们由两个隐藏层组成，每层各有 128 个神经元，并使用 ReLU 激活函数。最终层包含一个不带激活函数的神经元。我们使用 Adadelta 优化器训练神经网络，并使用学习率回调和提前停止回调：前者将学习率降低 50%，后者则在损失不再下降时停止训练。因此，只需将训练轮数设得足够大，确保始终触发提前停止即可；在我们的案例中，ANN 的训练轮数为 2000，TNNR 的训练轮数为 10000。两种情况下的批次大小均为 16。

附录 D. 随机森林

本文中的随机森林使用 scikit-learn 库 [20]。我们在所有数据集的一个子集上训练模型：从每个数据集中随机抽取 100 个训练数据点和 100 个测试数据点。我们使用五折交叉验证来优化随机森林的以下超参数：“最大深度” $\in [4, 8, 16, 32, 64]$、“最大特征数” $\in [0.33, 0.667, 1.0]$、“叶节点最小样本数” $\in [1, 2, 5]$、“划分最小样本数” $\in [2, 4, 8]$、“估计器数量” $\in [100, 300, 600]$。传统随机森林和孪生随机森林都从同一个参数池中选择最优超参数。需要注意的是，对于半监督学习，超参数仅在

表 D1： 随机森林与孪生随机森林测试均方根误差的最佳估计值

	随机森林	孪生随机森林	改进
生物保护	0.7407± 0.0087	0.7427± 0.0081	-0.27%
波士顿房价	4.0019± 0.1394	3.8301± 0.1322	4.29%
混凝土强度	6.3763± 0.2136	5.8519± 0.2242	8.22%
能源效率	1.8773± 0.0422	1.8906± 0.0417	-0.71%
RCL电路	0.3168± 0.0071	0.2958± 0.0068	6.63%
测试函数	0.1402± 0.0044	0.0874± 0.0023	37.66%
惠斯通电桥	0.1461± 0.0039	0.0942± 0.0025	35.52%
红葡萄酒质量	0.5989± 0.0085	0.6068± 0.0083	-1.32%
游艇水动力学	1.1917± 0.029	1.1117± 0.0383	6.71%

初始监督学习步骤中进行优化，随后将最优参数沿用到半监督学习过程中。

算法1： 通过环路一致性进行半监督学习

数据： 带标签数据集 $D_L = (X_{train,\ L},\ Y_{train,\ L})$

无标签数据集 $D_U = (X_{train,\ U})$

输入： 循环权重 Λ

循环数 n_l =length(D_L)/3

创建 $\tilde{D}_L = [((x_i, x_j, x_i - x_j)\ y_{ij} = y_i - y_j)$ 对于 $x_i \in X_{train,L}$ 对于 $x_j \in X_{train,L}]$

初始化机器学习模型 M

在 M 上训练 $\tilde{D}_L$

n_l个样本 环路 L=$[(x_i, x_j, x_k)$，其中 $x_i \in X_{train,L}, x_j, x_k \in X_{train,U}]$

对于 $(x_i, x_j, x_k) \in L$ **执行**

  预测 $M(x_i, x_j)\ M(x_j, x_k)\ M(x_k, x_i)$

  $a = M(x_i,\ x_j) + M(x_j,\ x_k) + M(x_k,\ x_i)$

  $(y_{ij},\ y_{jk},\ y_{ki}) = (M(x_i,\ x_j) - \Lambda a,\ M(x_j,\ x_k) - \Lambda a,\ M(x_k,\ x_i) - \Lambda a)$

  添加 $((x_i, x_j, x_i - x_j)\ y_{ij})$,$((x_j, x_k, x_j - x_k)\ y_{jk})$,$((x_k, x_i, x_k - x_i)\ y_{ki})$ 到 $\tilde{D}_L$

训练 M 上 $\tilde{D}_L$

输出： 训练后的模型 M

[1] Wetzel S J, Ryczko K, Melko R G and Tamblyn I 2022 Applied AI Letters e78[2] Wetzel S J, Melko R G and Tamblyn I 2022 Machine Learning: Science and Technology 3 045007[3] Tynes M, Gao W, Burrill D J, Batista E R, Perez D, Yang P and Lubbers N 2021 Journal of Chemical Information and Modeling 61 3846–3857[4] Baldominos A, Blanco I, Moreno A J, Iturrarte R, Bern´ardez ´O and Afonso C 2018 Applied sciences 8 2321[5] Rafiei M H and Adeli H 2016 Journal of Construction Engineering and Management 142 04015066[6] Yu Y, Lu J, Shen D and Chen B 2021 Neural Computing and Applications 33 3925–3937[7] Ryczko K, Wetzel S J, Melko R G and Tamblyn I 2022 Journal of Chemical Theory and Computation 18 1122–1128[8] Avula N V, Veesam S K, Behera S and Balasubramanian S 2022 Machine Learning: Science and Technology[9] Czischek S, Yon V, Genest M A, Roux M A, Rochette S, Lemyre J C, Moras M, Pioro-Ladri`ere M, Drouin D, Beilliard Y et al. 2021 Machine Learning: Science and Technology 3 015001[10] Lechner M, Hasani R M and Grosu R 2018 arXiv preprint arXiv:1803.08554[11] Bromley J, Guyon I, LeCun Y, S¨ackinger E and Shah R 1993 Advances in neural information processing systems 6 737–744[12] Baldi P and Chauvin Y 1993 neural computation 5 402–418[13] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 The journal of machine learning research 15 1929–1958[14] Wan L, Zeiler M, Zhang S, Le Cun Y and Fergus R 2013 使用 dropconnect 的神经网络正则化 国际计算机学习会议 (PMLR) pp 1058–1066[15]Wu J 2009 一种基于 k 近邻非参数估计回归函数的全新人工神经网络集成模型及其在降雨预报中的应用 2009 国际计算科学与优化联合会议 第 2 卷 (IEEE) pp 44–48[16] Bensaci R, Khaldi B, Aiadi O and Benchabana A 2021 Applied Sciences 11 10176[17] Liu Z, Guo J, Cao J, Wei Y and Huang W 2018 Promet-Traffic&Transportation 30 445–456[18] Wetzel S 2023 公开 GitHub 代码库 URL https://github.com/sjwetzel/PublicGetMostOutOfTNNR[19] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado G S, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz, Kaiser L, Kudlur, Levenberg, Man´e D, Monga, Moore, Murray, Olah, Schuster, Shlens, Steiner, Sutskever, Talwar, Tucker, Vanhoucke, Vasudevan, Vi´egas F, Vinyals, Warden, Wattenberg, Wicke, Yu and Zheng 2015 TensorFlow：大型异构系统机器学习软件，可从 tensorflow.org 获取 URL https://www.tensorflow.org/[20] Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot and Duchesnay 2011 Journal of Machine Learning Research 12 2825–2830