打分：

南京信息工程大学

课程名称： 气象统计实习报告

班级： XXXX 级专业名称 X 班

姓名：

学号：

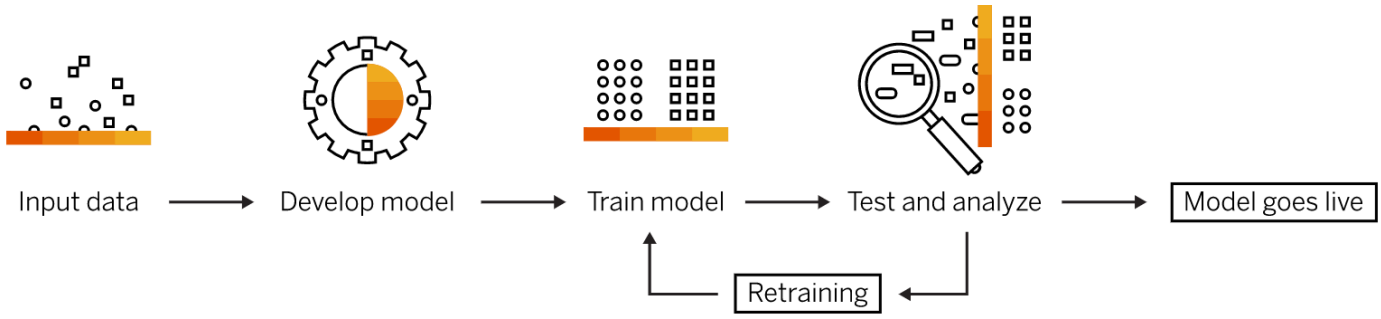
一、视频学习总结

1、机器学习的概念

机器学习是对非线性复杂模型的研究，它可以通过庞大的数据群以及不断地训练自动改进数据分析模型，通过不断微调模型中的参数，使模型能够越来越好地拟合样本数据，最后得到一套模型参数。这个训练过程，称之为机器学习。机器学习是人工智能的一个子集。这项技术的主要任务是指导计算机从数据中学习，然后利用经验来改善自身的性能。在机器学习中，算法会不断进行训练，从大型数据集中发现模式和相关性，然后根据数据分析结果做出最佳决策和预测。机器学习应用具有自我演进能力，它们获得的数据越多，准确性会越高。

机器学习包含多种使用不同算法的学习模型。根据数据的性质和期望的结果，可以将学习模型分成四种，分别是监督学习、无监督学习、半监督学习和强化学习。而根据使用的数据集和预期结果，每一种模型可以应用一种或多种算法。

机器学习的主要应用有对事物进行分类、发现模式、预测结果，以及制定明智的决策。算法一般一次只使用一种，但如果处理的数据非常复杂、难以预测，



也可以组合使用多种算法，以尽可能提高准确度。

图 1. 机器学习流程的工作原理

2、常用机器学习模型

1)分类模型：

①K 近邻(KNN)：是一种有监督学习方法。首先找到待测样本周围最近的已知样本点，接着统计周围已知样本点分布情况，再将待测样本类型归为优势方。其中定义数学上距离远近采用欧氏距离算法，先定义空间、距离公式，再录入

样本点信息，找最近的 k 个样本点（对 k 值非常敏感），找到最近的 k 个样本点中，哪个类别最多。过小的 k 容易造成过拟合，过大的 k 容易造成欠拟合

为保证合理的 k 值，采取交叉验证（cross-validation，来保证最优的估算）

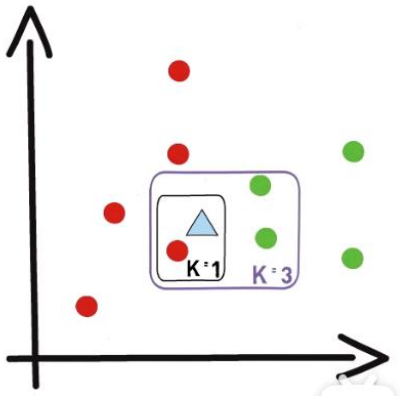


图 2. K 近邻算法(@五分钟机器学习)

②决策树(Decision Tree):其是机器学习中具有树状结构的算法，每个内部节点表示一个属性上的判断，每个分支代表一个判断结果的输出，每个叶节点代表一种分类的结果。决策树优势：直观，可视化；易于追溯和倒推。只适用于分类问题，树深太浅会导致欠拟合，树深太深会导致过拟合。若数据量很大，训练过程会很慢。

图 3. 决策树(@五分钟机器学习)

③随机森林：随机森林是一种典型的 Bagging 类算法。Bagging 算法是集成学习中一种典型的算法。这类算法可与其他分类、回归算法结合，提高其准确率、稳定性的同时，通过降低结果的方差，避免过拟合的发生。给定一个大小为 N 个样本 D 个特征(feature)的数据集 Dataset, Bagging 算法从中均匀、有放回地（即随机抽样法）选出 n 个样本且每个样本包含 d 个特征的子集，作为新的训练集。这样重复 m 次，对于每一次的随机采样，都在子训练集上使用分类、回

归等算法，则可得到 m 个模型。再通过取平均值、取众数等方法，即可得到 Bagging 的结果。假如我们的训练集包含了 N 个样本（Sample），每个样本被 D 个特征（feature）所描述，那训练一个随机森林主要分为以下几个步骤：首先预设模型的超参数，比如森林中有多少树 （Num of Learners）,每棵树最多几层深度（Max Depth）等。接着为了训练每个决策树，我们从完整的数据集(N 个样本，D 个 feature）中随机采样，选取 n 个样本，d 个 feature，从而保证每个决策树看问题的角度都不一样。然后根据每次采样，训练一个决策树。 当每个决策树都完成了训练，输入待测样本集（Test Set），我们再把最后每个树的测试结果整合在一起。

2)回归模型

①浅层神经网络(SNN)

浅层神经网络只有少量隐藏层的神经网络，浅神经网络只包含一到两层隐藏层。对浅神经网络的研究可以加强我们对深度神经网络内部运行机制的理解。下图所示是一个只包含一个隐藏层、一个输入层和一个输出层的浅神经网络。

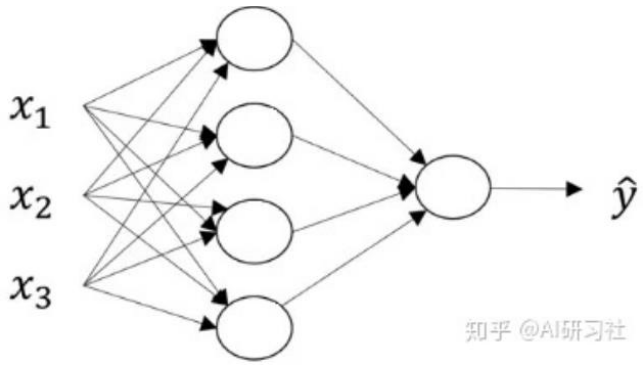


图 4. 浅层神经网络模型

输入层：输入预报因子x_1、x_2、x_3，它们被竖直地堆叠起来形成列，作为神经网络的输入。

隐藏层：上图中间的四个神经元。在训练神经网络时，我们只知道网络的输入值和输出值，无法得知隐藏层中各节点的具体数值。

输出层：如上图中最后一层的一个神经元，负责产生预测值。

②卷积神经网络模型(CNN)：

卷积神经网络通常由输入层→卷积层→池化层→全连接层→输出层组成，通过将这些层叠加起来，就可以构建一个完整的卷积神经网络。

输入层与传统神经网络/机器学习一样，模型需要输入的进行预处理操作，常见的 3 中预处理方式有：去均值、归一化、PCA/SVD 降维等。

卷积层是构建卷积神经网络的核心层，它产生了网络中大部分的计算量。以二维图像为例，通过二维离散卷积操作，把从全连接变成一个个小的局部连接，实现了高效的特征提取。卷积核具备空间平移不变的特性，无论图像大小如何，使用卷积操作，每个滑动区域都具有相同的共享权重。如果是 3*3 大小的卷积核，只需要学习 9 个参数，大大减少了要学习的参数数量。如果拥有 n 层卷积，就是深度卷积网络，层数越多，对复杂特征的表达能力越强。

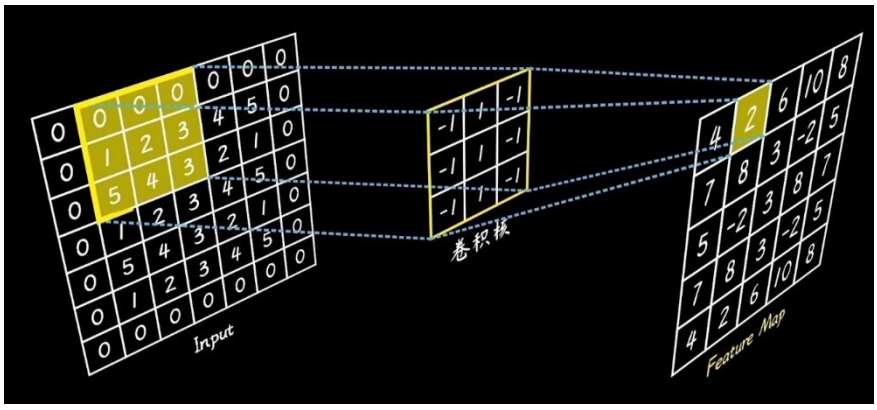
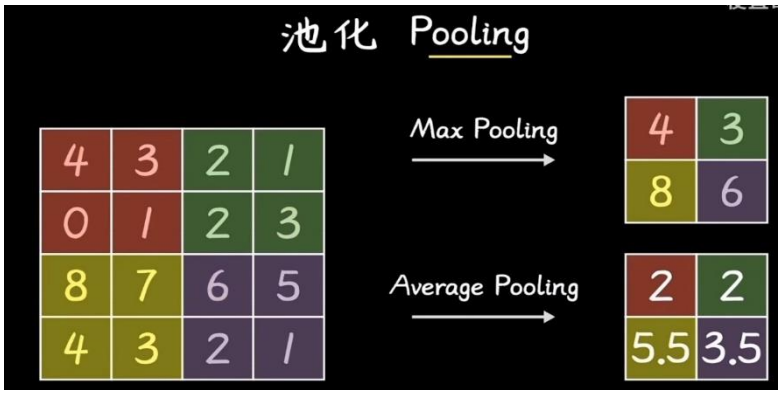


图 5. 卷积核工作原理

池化层（抓住主要矛盾，忽略次要因素）：将局部多个神经元的输出组合成下层单个神经元，来减少数据维度。该操作的目的是进一步放大主要特征，忽略掉几个像素的偏差，其意义是减少数据维度，减少训练参数，同时能避免过拟合。



卷积层加池化层，多层集连，对输入数据进行多尺度的特征提取和深度学习。

图 6. 池化层工作原理

全连接层：把相邻两层的神经元全部交叉相连，与传统神经网络相同（一般在最后层）和前面的卷积层相配合，形成先局部后整体的结构。当来到了全连接层之后，可以理解为一个简单的多分类神经网络（如：BP 神经网络），通过激活函数得到最终的输出。整个模型训练完毕。

③循环神经网络(RNN)

循环神经网络考虑了单个隐层在时间维度上的变化，沿着时序反复迭代的网络结构，实现对序列数据的学习。与在每层使用不同参数的传统深度神经网络不同，RNN 沿着时间轴重复，建立时序上的关联，假定不同时刻层级间共享一个 Ws（权重矩阵），这反映了我们在每个步骤执行相同任务的事实，只是使用不同的输入，这大大减少了我们需要学习的参数总数。由于时序上的层级结构，使 RNN 在输入输出关系上具有更大的灵活性，以便解决不同种类的问题。

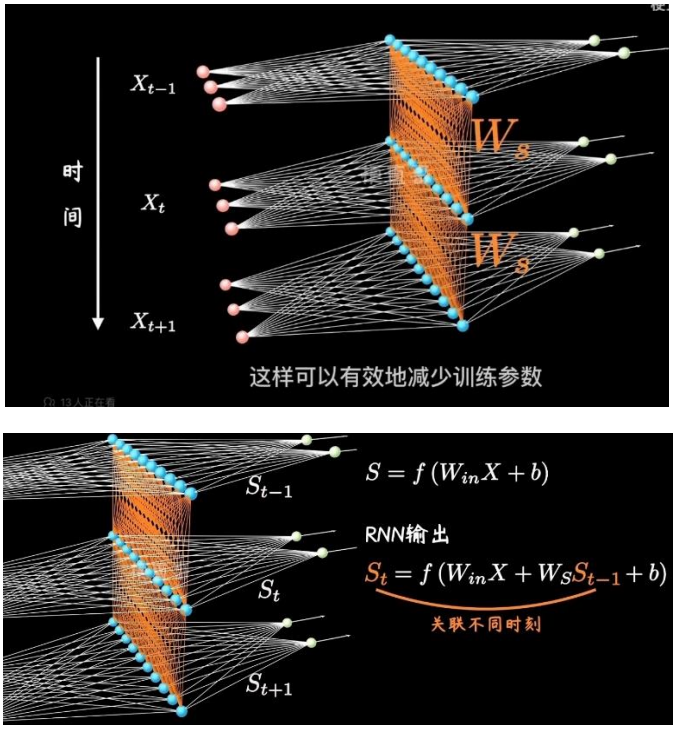


图 7. RNN 结构原理

3、训练集、测试集、验证集概念

如果给定的样本数据充足，我们通常使用均匀随机抽样的方式将数据集划分成 3 个部分——训练集、验证集和测试集，这三个集合不能有交集。

训练集用来训练模型→确定模型的权重和偏置这些参数，通常我们称这些参数为学习参数。

验证集用来对模型评估→选出得分最高的模型及对应超参 。验证集并不参与学习参数的确定，也就是验证集并没有参与梯度下降的过程。验证集只是为了选择超参数，比如网络层数、网络节点数、迭代次数、学习率这些都叫超参数。

测试集用来测试模型→评价模型泛化性能等的好坏。测试集只使用一次，即在训练完成后评价最终的模型时使用。它既不参与学习参数过程，也不参与超参数选择过程，而仅仅使用于模型的评价。值得注意的是，千万不能在训练过程中使用测试集，而后再用相同的测试集去测试模型。这样做其实是作弊，使得模型测试时准确率很高。

4、泛化能力、过拟合概念

泛化能力：指机器学习算法对新鲜样本的适应能力。学习的目的是学到隐含在数据背后的规律，对具有同一规律的训练集以外的数据，经过训练的网络也能给出合适的输出，该能力称为泛化能力。通俗来讲，泛化能力也就是举一反三的能力。对于深度学习或机器学习模型而言，我们不仅要求它对训练数据集有很好的拟合（训练误差），同时也希望它可以对未知数据集（测试集）有很好的拟合结果（泛化能力），所产生的测试误差被称为泛化误差。度量泛化能力的好坏，最直观的表现就是模型的过拟合（overfitting）和欠拟合（underfitting）。过拟合和欠拟合是用于描述模型在训练过程中的两种状态。

欠拟合：指模型不能在训练集上获得足够低的误差。换句换说，就是模型复杂度低，模型在训练集上就表现很差，没法学习到数据背后的规律。

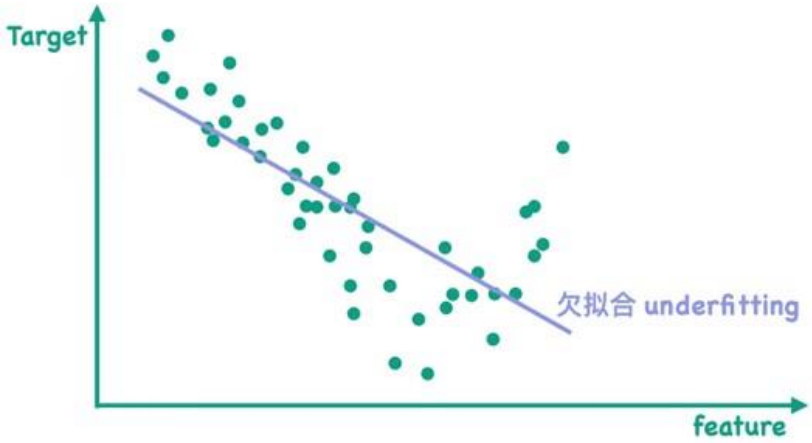


图 8. 欠拟合现象

过拟合：指训练误差和测试误差之间的差距太大。换句换说，就是模型复杂度高于实际问题，模型在训练集上表现很好，但在测试集上却表现很差。

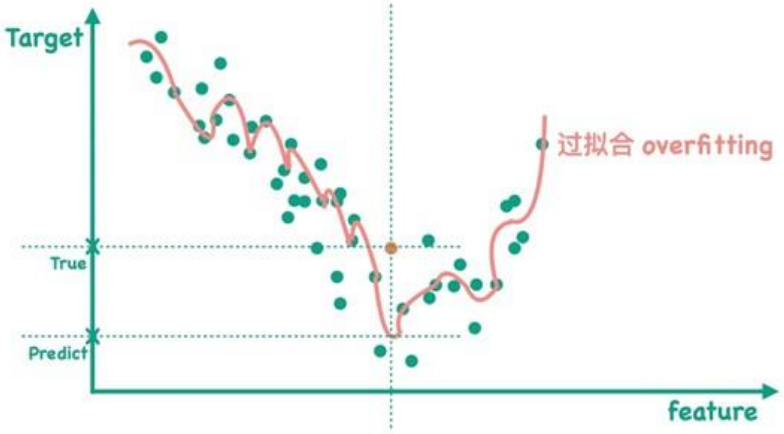


图 9. 过拟合现象

一般来说，训练过程会是如下所示的一个曲线图。

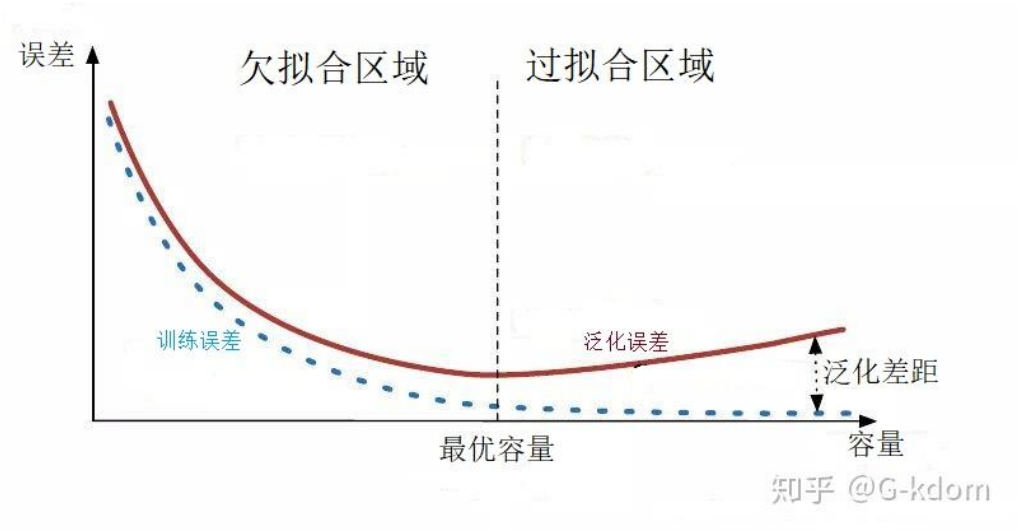


图 10. 欠拟合与过拟合

训练刚开始的时候，模型还在学习过程中，处于欠拟合区域。随着训练的进行，训练误差和测试误差都下降。在到达一个临界点之后，训练集的误差下降，测试集的误差上升了，这个时候就进入了过拟合区域——由于训练出来的网络过度拟合了训练集，对训练集以外的数据拟合效果不好。使用更多的数据是解决过拟合的根本方法。

5、气象领域使用 AI 技术将面临的哪些挑战？

不同与广泛使用机器学习方法的“大数据”领域（例如：图像识别）。AI 在气象领域应用将会面临两类特殊问题。第一类：气象观测样本数据不够多。气象观测也就百年左右。若分析气候问题，样本量十分有限。天气过程事件虽然相对较多些。但是，其中极端事件所占比例很少。因此，训练模型时往往能抓住一般天气事件的特征，而不能抓住极端事件的特征。然而，对极端事件的预报，才是社会公众最需要的气象服务。若仅使用极端事件训练模型，由于样本量少，往往导致模型过拟合、不稳定等问题。第二类：气象领域已经积累了很多理论研究，具有大量的气象背景知识。机器学习得到的模型不能违背已有气象背景知识的。若想将 AI 成功用于气象服务，一定要结合气象领域的这两类特殊问题进行有针对性的技术处理。针对上述问题的研究目前还很少见，无合适的相关资料。这里给出作者本人的研究分析。

在训练机器学习模型过程中，过拟合是一种普遍的现象。过拟合是指能够很好地拟合训练数据，而对除训练数据之外的未知数据却不能进行很好地拟合。例如：记录 10 个人近 10 天每人每天零点零分的心情指数和近 10 天某地区逐日气温。通过求解 10 元一次方程组，就可以得到用这 10 人心情指数完全拟合当天气温的多元线性回归方程。这个多元线性回归方程仅仅对这 10 天数据样本适用，丝毫不具备对今后气温预测的能力。模型训练的最终目的是提高其泛化能力，使其对未知数据也能够较好地拟合。如果训练样本量不足，训练样本代表的特征与总样本特征偏差较大，就会导致过拟合（Goodfellow et al., 2015）。这里采用机器学习网络教程中常用的手写数字识别数据集和相应模型展示过拟合现象。这套数据集共有 6 万个样本，通常随机抽取 5 万个作为训练数据、剩下的 1 万个作为测试数据。在 5 万个训练样本情况下，训练若干步后，模型对训练集的拟合精度和对测试集的拟合精度几乎相同（图略）。在“大数据”样本充足情况下，过拟合问题可以忽略。若从 6 万个样本中仅抽出 200 个样本作为训练数据。模型对训练集的拟合精度和对测试集的拟合精度呈现显著差异（图 1）。从图中可以看出，当训练迭代次数超过 100 步时，模型对训练集的识别精度几乎达到 100%。但是，对于测试集来说，识别精度不超过 80%。主要原因是模型拥有大量可调参数（大于 1 万），在训练数据偏少情况下，导致模型过多地去努力抓住训练样本独有的特征，而不是数据总体的共同特征。两个数据集上识别精度的明显差异说明过拟合问题不可忽略。总之，在样本量不足（样本量远小于模型可调参数个数）的情况下，机器学习方法训练得出的复杂模型（模型容量较大）往往会产生严重的过拟合现象。若不分析了解过拟合问题，往往会导致机器学习模型在汇报介绍时效果很好、但用于实际业务后效果下降。

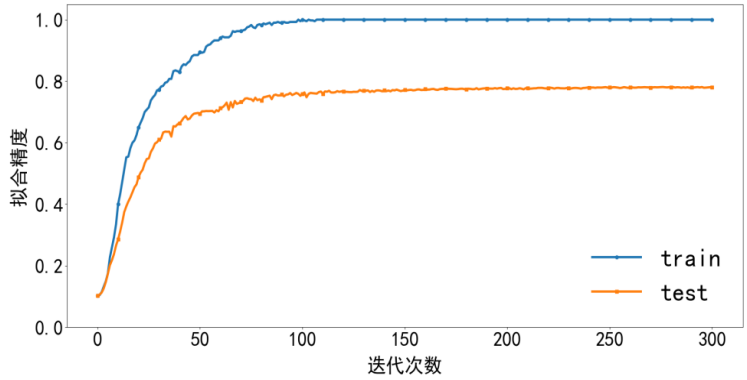


图 11. 拟合精度随迭代次数的变化。两条曲线分别为训练集(train，蓝色)和测试集(test，红色)。

机器学习模型在训练过程的参数初始化、样本使用顺序等方面常常具有一些随机性。因此，基于完全相同的训练样本，每次训练得到的模型参数不同。在“大数据”情况下，这种随机性有利于抑制模型过拟合、便于训练找到最佳模型参数。但是，在小样本情况下，每次训练得到的模型在测试样本上的计算输出结果往往存在较大偏差，即模型不稳定。这里采用机器学习网络教程中常用的房屋价格预测模型及其数据集来举例说明模型稳定性问题。这套数据共有 506 个样本。在小样本情况下，设置训练样本量为 40 个，测试样本 20 个。为对比小样本产生的影响，也开展正常的全样本试验（即大样本情况，除 20 个测试样本外的数据全部用于训练模型）。图 2 展示了小样本情况下模型预测结果的稳定性情况。预测结果的稳定性由 10 个集合成员（即 10 次训练得到的模型）的标准差来衡量（即图中的 error bar）。小样本情况下模型预测结果的标准差在 0.3 左右，远高于大样本情况下的标准差（0.05 左右）。小样本情况下模型预测技能（用预测值和实际值的相关系数 Cor 衡量）为 0.58±0.11，而大样本情况下模型预测技能为 0.77±0.02。由于过拟合现象，由小样本训练得出的模型预测技能明显低于大样本训练得出的模型。此外，由小样本训练得出的模型预测技能的稳定性（用 10 个成员计算得到 Cor 的标准差衡量，0.11）也明显弱于大样本训练得出的模型（0.02）。总之，由于样本量少，机器学习模型性能存在明显的不稳定性。一个不稳定的机器学习模型，肯定不能用于气象服务。

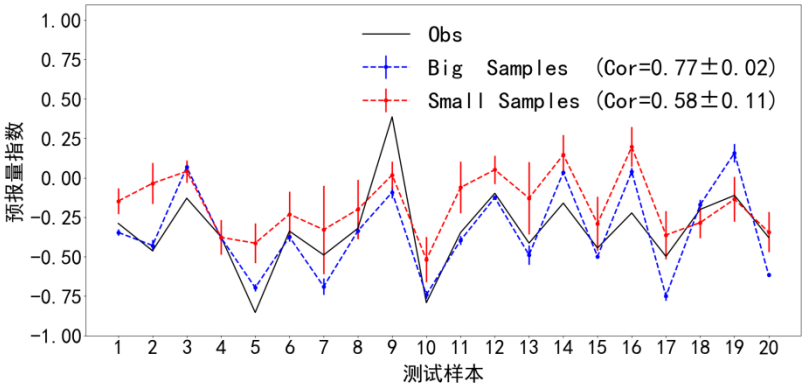


图 12. 基于测试样本得出的模型预测结果和相应实际值。黑色线为实际观测值；红色线为小样本下模型的预测结果；蓝色线为大样本下预测结果；竖线表示预测结果的不确定性范围（即 error bar，10 个成员的均方差）。

如何抑制小样本带来的上述困扰，还有待进一步深入研究。这里仅仅给出两点思路，以供读者参考。第一，人为扩充样本。仿照其他机器学习应用领域样本扩充技术，对气象事件样本进行一定程度的扩充，增加样本量。例如：借鉴图像识别领域中的图像数据扩充方法（如：翻转、旋转、变形、裁剪、添加噪声和像素缩放等），利用气象背景知识对气象事件样本数据进行合理扩充，即增加一些符合气象背景知识的人为制造的气象事件。第二，尽量使用可调参数较少（即模型容量低）的简单模型。图 3 展示了，小样本情况下，复杂神经网络和简单神经网络的模型性能差异。为模拟小样本情况，采用 60 个数据样本开展试验（训练样本 40 个，测试样本 20 个）。复杂模型所使用的神经网络模型有 8 个隐藏层，每层 50 个节点；简单模型为有 1 个隐藏层的浅层神经网络模型，节点数设为 20。相比复杂模型，简单模型预测结果的标准差明显减小。此外，简单模型的预测技能（0.60）略高于复杂模型（0.58）；预测技能的不确定性范围（0.09）也明显优于复杂模型（0.16）。

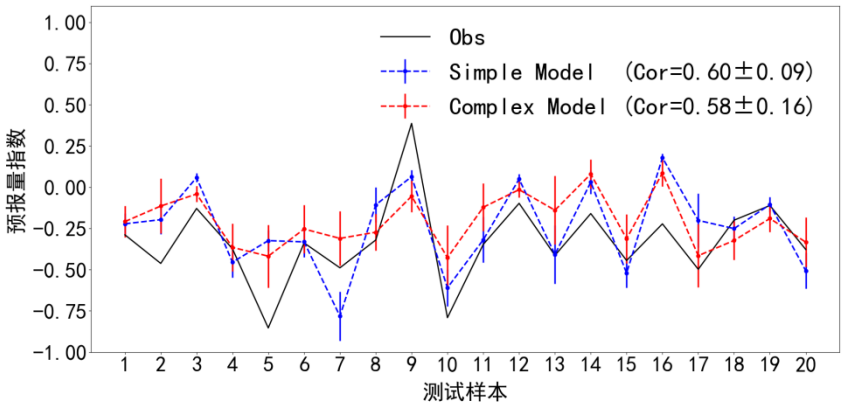


图 13. 同图 12，小样本情况下，复杂神经网络模型和简单神经网络模型对测试样本预测结果的对比。

机器学习方法在气象领域中的应用越来越多，正在成为一个强有力的工具。然而，机器学习模型通常被比喻为“黑箱子”。人们很难理解或者解释机器学习模型到底学到了什么。对模型结果无法解释，就难以根据模型结果进行决策。因此，分析评估模型可解释性十分重要。在分析机器学习模型可解释性的众多方法中，有一个几乎通用的方法——“梯度扰动”方法。该方法把统计模型当作“黑箱子”，通过给输入数据施加小扰动，分析模型输出结果对各个输入数据的敏感性。“梯度扰动”方法可以了解模型中各个输入量对输出结果的贡献。然后将之与气象背景知识进行对比，判断模型是否具有可解释性。凡是与气象背景知识相悖的模型，肯定不能使用。

这里以一个基于前期海温预测副高指数的神经网络模型进行讲述。图 4 展示了模型输入量（前期秋、冬海温）对模型输出量（6 月西太副高指数）的贡献。通过相关系数可以初步了解前期秋、冬海温对 6 月西太副高指数的影响（图 4 上）。前人大量研究工作已经对这种线性统计关系给出了合理的物理解释，并得到广泛认可。在这里，将其当作气象背景知识。与线性相关性不同，非线性统计模型（神经网络模型）在某区域（如北太平洋）的敏感性会受到其他区域（如东太平洋）海温状况的影响。因此，由“梯度扰动”方法计算得出的神经网络模型敏感性空间分布图具有年份依赖性。图 4 给出了四个样例年（1962、1974、1983 和 2004 年）的输出结果对输入数据的敏感性空间分布图。这 4 年神经网络模型敏感性空间分布特征与气象背景知识（线性相关性）基本一致。这四年实际观测的副高指数分别为：-0.15（正常）、0.70（高），-0.58（低）和 1.00（高）；模型预测的副高指数为：-0.04（正常）、0.51（高），-0.56（低）和 -0.38（正常）。1962

年、1974 年和 1983 年的副高指数预测基本正确，而 2004 年的预测显然错误。相对于预测正确的年份，预测失误年份（如 2004 年）模型输出结果对输入数据的敏感性整体相对较弱，其空间特征与线性相关性的差异较大。大多数预测错误年份都有这种特征（图略）。也就是说，通过分析模型可解释性还可以在一定程度上了解当年预测的可信度。若敏感性整体偏弱、空间分布与气象背景知识差异较大通常预示该年份的预测可信度较低。

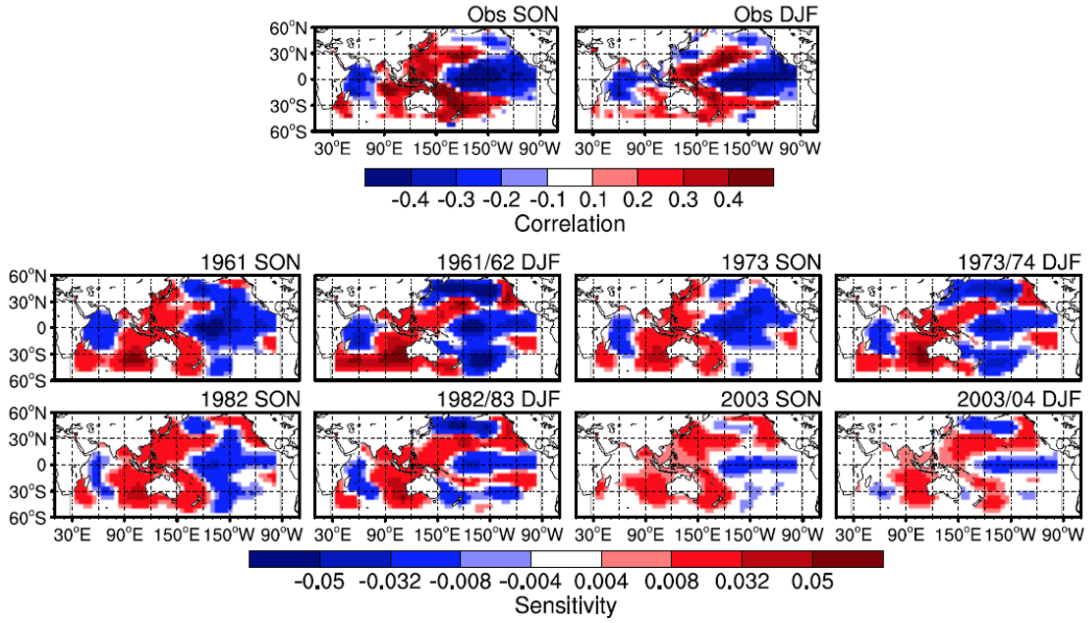


图 14. 模型输入量（前期秋、冬海温）对模型输出量（6 月西太副高指数）的贡献。上面板为基于观测数据得出的输入量与输出量的线性相关；下面板为神经网络模型基于某年样本计算得出的输出量对输入量的敏感性。

二、神经网络试验

1、浅层神经网络的不稳定性试验设计

试验设计思路：Xobs 为生成-1~1 之间的随机数，Yobs 根据设置的神经网络结构生成，如下图。不卡死设置初始权重的种子，将训练集上训练所得参数应用于测试集，计算得出测试集 Yobs 与通过该神经网络计算得到的 YML 的差值，将运行 10 次得到的结果进行集合分析，可发现这 10 个集合成员的标准差较大，模型每次训练所的结果存在较大波动，即表现出了小样本时浅层神经网络的不稳定性。

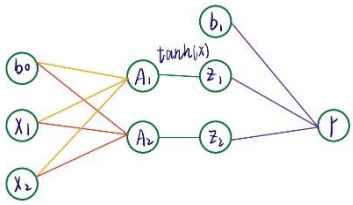


图 15. 浅层神经网络结构

试验步骤：

1) 修改路径：

../bld/makefile 文件中修改路径

```
FC      = ifort
###FFLAGS = -r8 -i4 -C
FFLAGS = -check bounds -r8 -i4 -C
OBJS    = main.o \
          statistic_subroutine.o ran_mod.o \
          pra_data_mod.o NNbase.o types_mod.o \
          preliminary.o
#
ROOT_DIR = /home/FCtest/caiyw/sxbg/NNtest/
EXEC     = $(ROOT_DIR)/test
VPATH    = $(ROOT_DIR)/code
#---------------------------------------
```

../code/pra_data_mod.f90 文件中修改路径

```
character(len = *), parameter :: rootdir="/home/FCtest/caiyw/sxbg/NNtest/"  ! root dir include code directory
 character(len = *), parameter :: ObsDatadir=trim(rootdir)//"ObsData/"
 character(len = *), parameter :: Samplesdir=trim(rootdir)//"samples/"
```

2) 修改神经网络超参数

../code/pra_data_mod.f90 文件中设置自己设计的神经网络超参数

```
integer,parameter :: Nobs=50
integer,parameter :: Np=2
integer,parameter :: Ntrain=30
integer,parameter :: Nbatch=10
integer,parameter :: Nvalidate=10
logical, parameter :: ORTestSet = .True.
integer,parameter :: Ntest=10
!logical, parameter :: ORTestSet = .False.
real,parameter :: LossThreshold = 0.35
integer,parameter :: Niters = 5000
real,parameter :: LearningRate  = 0.1
integer,parameter ::  SeedNNstate=0!431495
integer,parameter ::  SeedNNsamples=45
integer,parameter ::  SeedNNbatch=20
```

这套数据共有 50 个样本，预报因子为 2 个。设置训练集样本量为 30 个，其中 10 个样本为一个 batch，验证集为 10 个，测试集为 10 个。

```
data w0given(1,1:Nmid) / 2.5,2.5/
data w0given(2,1:Nmid) /5.0,-5.0/
data b0given(:) /0.0,0.0/
data w1given(1:Nmid) /1.0,1.0/
data b1given /0.0/
```

```
!!!!<<<<<<<<Two Hidden Layers>>>>>>>>>
    !integer,parameter :: Nhid=2
    !integer,parameter :: N0=2,N1=6,N2=4    !N0-input layer;N1-first hidden layer; N2-second hidden layer
    !!!!<<<<<<<<One Hidden Layer>>>>>>>>>
    integer,parameter :: Nhid=1
    integer,parameter :: N0=2,N1=2,N2=0   ! Note,the N2 must be defined 0.

    !<<<<<Type of Trainning>>>>>>
    integer,parameter :: NNtraintype=0     ! 0-basetrain;
    !integer,parameter :: NNtraintype=1    ! 1-lootrain
    integer,parameter :: Nallseeds=200      ! lootrain, seeds number begin from SeedNNstate
    integer,parameter :: Ngoodseeds=10      ! lootrain, how mang good seeds


    character(len = *), parameter :: rootdir="/home/FCtest/caiyw/sxbg/NNtest/"    ! root dir include code directory
    character(len = *), parameter :: ObsDatadir=trim(rootdir)//"ObsData/"
    character(len = *), parameter :: Samplesdir=trim(rootdir)//"samples/"

    !logical, parameter :: ORProduceData = .False.    ! False,Read Obs Data or Produced Data. <house data Nobs=5
    logical, parameter :: ORProduceData = .True.    ! Ture,Producing Obs Samples based on a given equation
```

该部分对应所设计的神经网络结构，输入层为 2 个神经元，含有一层隐藏层，隐藏层的神经元数为 2，输出层为 1 个神经元。第一支为 A1=2.5x1+5.0x2,第二支为 A2=2.5x1-5.0x2。

3) 编写生成样本代码

../code/preliminary.f90 文件中生成样本并写入数据文件

```
    subroutine ProduceObsData
      implicit none
      integer ip,iobs,i
      real Yrandom(Nobs),Rantemp(Nobs),Xmid(Nmid,Nobs),Xobs0(Np,Nobs),Xmid_1(Nmid,Nobs)
      ...
    !!!生成均匀分布的随机样本Xobs
      do iobs=1,Nobs
        Yobs(iobs)=0.
        do ip=1,Np
          call random_number(Rrandom)
          Xobs(ip,iobs)=Rrandom*2.0-1.0
        enddo
      enddo

      !!!根据浅层神经网络生成预报量样本Yobs
       do iobs=1,Nobs
         do i=1,Nmid
           Xmid_1(i,iobs)=b0given(i)
           do ip=1,Np
             Xmid_1(i,iobs)=Xmid_1(i,iobs)+Xobs(ip,iobs)*w0given(ip,i)
           enddo
!          Xmid(i,iobs)=1./(1.+exp(-Xmid_1(i,iobs)))          !sigmoid
           Xmid(i,iobs)=2./(1.+exp(-2.*Xmid_1(i,iobs)))-1  !tanh
         enddo
       enddo
       do iobs=1,Nobs
!        Ygiven(iobs)=b1given
         Yobs(iobs)=b1given
         do i=1,Nmid
!          Ygiven(iobs)=Ygiven(iobs)+w1given(i)*Xmid(i,iobs)
           Yobs(iobs)=Yobs(iobs)+w1given(i)*Xmid(i,iobs)
         enddo
       enddo

      open(22,file=ObsDatadir//"OBS_samples.txt",form="formatted")
      do iobs=1,Nobs
        write(22,('(i4,2x,<Np>f8.4,2x,f20.4)')) iobs,Xobs(1:Np,iobs),Yobs(iobs)
      enddo
      close(22)
    endsubroutine ProduceObsData
```

4) 运行结果：

在 NNtest 目录下输入指令./run,即可得到如下结果，多次运行会发现每次所得 loss 和参数大小不同，即该模型不稳定。下图为其中 2 次运行结果。

```
[FCtest@node1 NNtest]$ ./run
rm -fr *.mod *.o /home/FCtest/caiyw/sxbg/NNtest//test
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/pra_data_
mod.f90 -o pra_data_mod.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/ran_mod.f
90 -o ran_mod.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/types_mod
.f90 -o types_mod.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/statistic_subroutine.f90 -o statistic_subroutine.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/NNbase.f90 -o NNbase.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/preliminary.f90 -o preliminary.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/main.f90 -o main.o
ifort  -check bounds -r8 -i4 -C main.o statistic_subroutine.o ran_mod.o pra_data_mod.o NNbase.o types_mod.o preliminary.o -o /home/FCtest/caiyw/sxbg/NNte
st//test
EXEC have compiled
 One Hidden Layer
 useful:  0.340000000000000        33.0000000000000
 useful:  0.520000000000000        24.0000000000000
 -0.9147 -0.7638
  0.1678 -0.8376
   5   lossTrain            0.48858    corrTrain            0.7788    lossTest            0.66647    corrTest            0.8737
  10   lossTrain            0.43996    corrTrain            0.8140    lossTest            0.62358    corrTest            0.9147
  15   lossTrain            0.37551    corrTrain            0.8570    lossTest            0.46651    corrTest            0.9484
  20   lossTrain            0.30145    corrTrain            0.9121    lossTest            0.37897    corrTest            0.9771
  25   lossTrain            0.22933    corrTrain            0.9502    lossTest            0.25426    corrTest            0.9864
  30   lossTrain            0.16939    corrTrain            0.9730    lossTest            0.19699    corrTest            0.9919
  35   lossTrain            0.12719    corrTrain            0.9849    lossTest            0.15010    corrTest            0.9948
  40   lossTrain            0.09949    corrTrain            0.9908    lossTest            0.12592    corrTest            0.9963
  45   lossTrain            0.08187    corrTrain            0.9937    lossTest            0.11589    corrTest            0.9972
  50   lossTrain            0.07501    corrTrain            0.9950    lossTest            0.12190    corrTest            0.9975
  55   lossTrain            0.06741    corrTrain            0.9958    lossTest            0.11136    corrTest            0.9979
  60   lossTrain            0.06577    corrTrain            0.9961    lossTest            0.11940    corrTest            0.9979
  65   lossTrain            0.06257    corrTrain            0.9963    lossTest            0.11326    corrTest            0.9981
 countup:           1
  70   lossTrain            0.06043    corrTrain            0.9965    lossTest            0.11121    corrTest            0.9982
  75   lossTrain            0.05926    corrTrain            0.9967    lossTest            0.11623    corrTest            0.9979
 countup:           1
 min lossTest:  0.112232614913660
  75   lossValidate         0.12223    corrValidate             0.9918
    state1%W0    -1.075   3.515   1.248   3.242
    state1%B0     0.035  -0.078
    state1%W1    -1.476   1.474
    state1%B1     0.026
```

```
[FCtest@node1 NNtest]$ ./run
rm -fr *.mod *.o /home/FCtest/caiyw/sxbg/NNtest//test
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/pra_data_mod.f90 -o pra_data_mod.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/ran_mod.f90 -o ran_mod.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/types_mod.f90 -o types_mod.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/statistic_subroutine.f90 -o statistic_subroutine.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/NNbase.f90 -o NNbase.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/preliminary.f90 -o preliminary.o
ifort  -c -check bounds -r8 -i4 -C /home/FCtest/caiyw/sxbg/NNtest//code/main.f90 -o main.o
ifort  -check bounds -r8 -i4 -C main.o statistic_subroutine.o ran_mod.o pra_data_mod.o NNbase.o types_mod.o preliminary.o -o /home/FCtest/caiyw/sxbg/NNtest//test
EXEC have compiled
 One Hidden Layer
 useful:  0.340000000000000        33.0000000000000
 useful:  0.520000000000000        24.0000000000000
  0.2390 -0.8468
 -0.2727  0.9958
   5   lossTrain            0.51506    corrTrain            0.7080    lossTest            0.67515    corrTest            0.8335
  10   lossTrain            0.48903    corrTrain            0.7419    lossTest            0.63836    corrTest            0.8639
  15   lossTrain            0.45536    corrTrain            0.7814    lossTest            0.60214    corrTest            0.8757
  20   lossTrain            0.44121    corrTrain            0.7953    lossTest            0.56586    corrTest            0.8788
  25   lossTrain            0.43639    corrTrain            0.8002    lossTest            0.57264    corrTest            0.8787
  30   lossTrain            0.43339    corrTrain            0.8031    lossTest            0.56599    corrTest            0.8780
 countup:           1
  35   lossTrain            0.43078    corrTrain            0.8060    lossTest            0.57939    corrTest            0.8776
 countup:           2
 min lossTest:  0.569259421987706
  35   lossValidate         0.60171    corrValidate             0.7379
    state1%W0     1.486   0.565  -1.571   0.608
    state1%B0    -0.271   0.807
    state1%W1     0.237  -0.951
    state1%B1     0.635
```

5)绘制 10 次运行结果集合：图中红实线为利用训练集运行 10 次所得的模型在测试集上测试算得的 YML 与理想情况下 Yobs 值的差的平均值，粉色区域为预测结果的不确定范围。

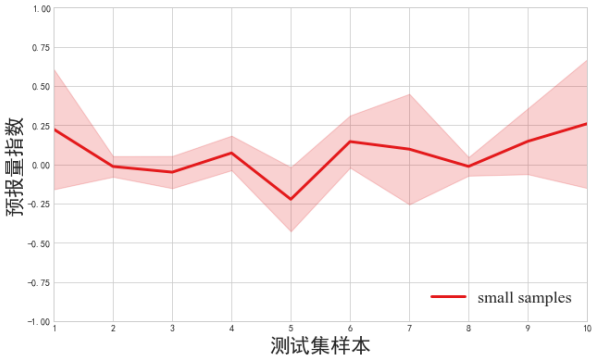


图 16. 基于测试样本得出的模型预测结果。红色实线为 10 次预测结果 YML 与

实际观测值 Yobs 差的均值；粉色区域表示预测结果的不确定范围(即 error bar，10 个成员的均方差)

2、通过线性回归模型展示过拟合

本试验使用多元线性逐步回归算法来展示过拟合现象。逐步回归的基本思想是从诸多因子中挑选一批关系显著的作为预报因子。将因子一个一个引入，条件是该因子的方差贡献显著，同时每引入一个新的因子，要对已经引入的因子逐个进行显著性检验，将方差贡献变为不显著的因子剔除。

首先在 pra_data_mod.f90 文件中修改样本相关的参数。设置样本量 Nt 为 1000 个，Xsamples 是输入的预报因子样本，共有 Nx——6 个，Xsamples 是根据平均值为 0，标准差为 1.0 生成的正态分布随机数，每个预报因子对应的多元线性回归系数 Bgiven 分别为 2.0，1.0，-1.0，0.5，-0.1，0.1，可以代表相应预报因子的重要程度；Ygiven 是预报量中的可解释部分，由 Xsamples 乘上其对应的系数生成；Yrandom 是预报量中的随机不可解释的部分，和 Ygiven 相互正交；Ysamples 等于 Ygiven 和 Yrandom 二者之和。接下来设置 Fgiven，代表自己预设的可解释部分的比例，即预报因子 x 对预报量 y 的方差的线性关系程度。Fgiven 值越大，代表可解释部分所占比例越高，该参数范围在（0.0, 1.0）之间。在此设 Fgiven 的值为 0.3。

```
integer,parameter :: Nt=1000                          ! the size of samples, suggest <99999
integer,parameter :: Nx=6                            ! the size of all input predictors, > Np
integer,parameter :: Ntrain=50                         ! Training Set. <=Nt <999
integer,parameter :: Ntest=50                          ! RollingForecast Test Set. <=Training

character(len = *), parameter :: rootdir="/nuist/scratch/meso/meso_shixj/temp/OverFitting/"    ! root dir include code directory
!character(len = *), parameter :: rootdir="/nuist/u/home/shixj/shixiangjun/StatisticMethods/OverFitting/"    ! root dir include code directory
character(len = *), parameter :: Samplesdir=trim(rootdir)//"samples/"

integer,parameter :: Np=6                              ! the size of usefull predictors, < Nx
real Bgiven(Np)
data Bgiven(1:Np) / 2.0, 1.0,-1.0, 0.5,-0.1, 0.1 /
!data Bgiven(1:Np) / 5.0,-3.0, 3.0, 0.000001,0.000001, 0.000001 /
!data Bgiven(1:Np) / 5.0,-3.0, 3.0 /
!data Bgiven(1:Np) / 1.0 /
real, parameter :: Fgiven=0.3                          ! the fraction of prediction is known by given equation
!logical, parameter :: ORProduceData = .False.    ! False,Read Produced Data.
logical, parameter :: ORProduceData = .True.    ! Ture,Producing Samples based on a given equation
integer,parameter ::  SeedData=4582591              ! used for the producing data
real Xsamples(Nx,Nt),Ysample(Nt),Ygiven(Nt),Yrandom(Nt)
real Xadd(Nx,Ntrain),Yadd(Ntrain),YsampleA(Nt),Yreg(Nt)        ! add-added sample; A-unnormal samples were amplified.
real Yhigh,Ylow                                    ! the threshould from Ysample
real,parameter ::  Ythreshould=1.2
integer XsamplesRank(Nx,Nt),YsampleRank(Nt),YgivenRank(Nt),YregRank(Nt)      ! 0=normal 25%-75%; 1=high; -1=low
integer,parameter ::  YgivenRankMethod=1              ! 0---based on sorting
                                                      ! 1---based on the threshould from Ysample
```

图 17. pra_data_mod.f90 文件修改参数

接下来在主程序中按照 call LoadData→call DataAnalyze→call DataExtend→call ForeCastAnalyze2 的次序运行，用到的统计方法是多元线性逐步回归算法，

输出的结果如下：

```
 All samples          1000
Syy,U,Q,U/Syy,cor:   20900.000    6270.000   14630.000       0.3000  0.5477
 Y--X 1       2.000     0.43748   0.19139
 Y--X 2       1.000     0.21874   0.04785
 Y--X 3      -1.000    -0.21874   0.04785
 Y--X 4       0.500     0.10937   0.01196
 Y--X 5      -0.100    -0.02187   0.00048
 Y--X 6       0.100     0.02187   0.00048
      sum of Bgiven(i)*Sxy/Syy    0.30000
 Yhigh,Ylow:   3.01762283557420       -2.99097984846241
  YgivenHigh   108 YRank L N H    3   33   72
  YgivenHigh%   10 YRank L N H    2   30   66
   YgivenLow   123 YRank L N H   67   52    4
   YgivenLow%   12 YRank L N H   54   42    3
YsampleHigh  250Ygiven L N H    4  174   72
YsampleLow   250Ygiven L N H   67  180    3
```

图 18. 样本分析结果

其中输出内容的第 1 行 All samples 代表总的样本量；第 2 行 Syy 代表样本 Ysample 的方差；U 代表样本可解释部分 Ygiven 的方差；Q 代表样本随机扰动部分 Yrandom 的方差；U/Syy 代表样本可解释方差，即 Fgiven；cor 代表给定部分 Ygiven 和样本 Ysample 的相关系数。第 3~8 行的第 2 列代表预测因子的给定系数 Bgiven；第 3、4 列代表单个因子的相关系数和解释方差。第 9 行 sum of Bgiven(i)*Sxy/Syy 代表所有预报因子的总解释方差，即可解释比例 0.3。第 10 行 Yhigh、Ylow 分别代表经排序得出的样本的高值阈值和低值阈值。YgivenHigh 代表在总样本中有 108 个样本属于高值样本，同一行的 YRank L N H 代表 Ygiven 高值时，样本值低、中、高分别出现的次数。第 11 行的 YgivenHigh%代表高值 Ygiven 在总样本中所占的比例，同一行的 YRank 代表 Ygiven 值高时，YsampleLow 是低、中、高分别出现的比例。第 12~13 行的 YgivenLow 和 YgivenLow%同理。最后两行的 YsampleHigh 和 YsampleLow 代表将总样本降序排列后划分的前 1/4 与后 1/4，即分别有 250 个高值和低值，同一行的 Ygiven L N H 代表 Ysample 高值时，Ygiven 是低、中、高分别出现的次数。

当回归方程中的可解释的 Y 的比例大于设置的理论模型中的可解释的 Y 的比例时，就会出现过拟合现象。下图是使用线性回归预测的结果，第 1~2 列代表过拟合系数，其中过拟合系数的定义为：回归方程中可解释 Y 的比例 / 理论模型中 Y 的可解释比例。因此，过拟合系数越高，代表模型的过拟合程度越严重。第 3 列代表不同过拟合系数出现的次数所占的百分比。

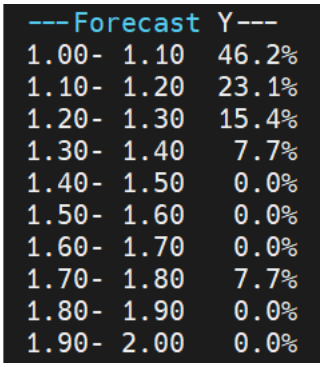


图 19.　Fgiven 值为 0.3 的过拟合结果

为了减弱过拟合现象，本试验对异常高和低的样本（即降序排列后，前 25%和后 25%的样本）进行扩充。右图为加了扩充异常样本后的试验结果，显而易见的，过拟合系数低的所占比例增加，过拟合系数高的所占比例减少，证明该试验扩充样本后，可以使过拟合程度有效降低。

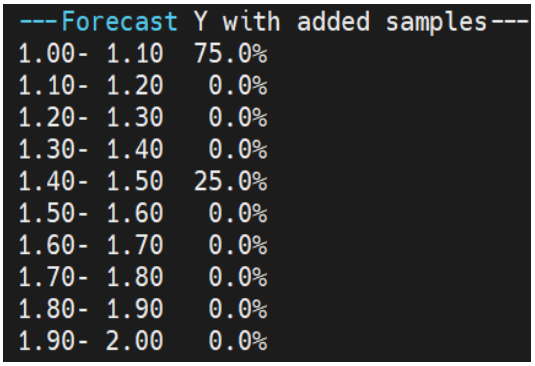
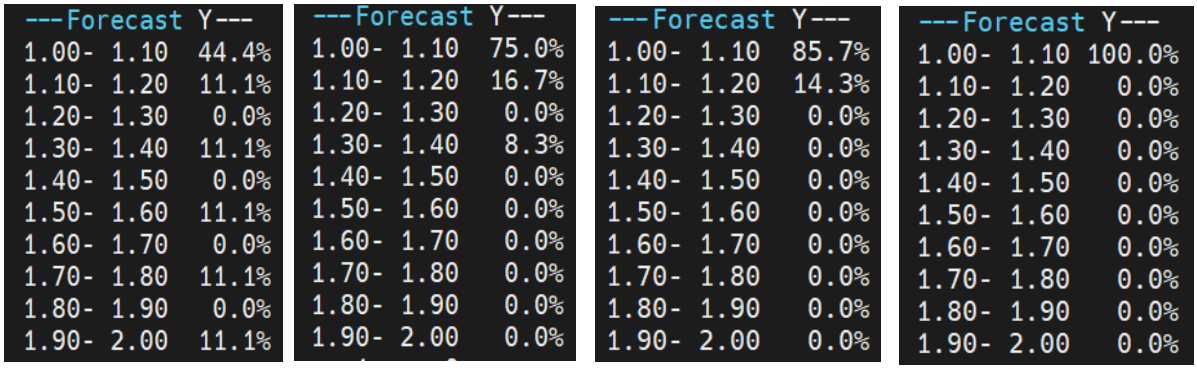


图 20.　加扩充样本后的过拟合结果

接下来改变 Fgiven 的值，重复上述试验。下组图从左到右分别为 Fgiven 值设置为 0.1、0.5、0.7、1.0 时的过拟合结果。可见随着可解释部分比例增加，过拟合程度显著降低，当 Fgiven 设置为 1.0 时，即可解释部分比例占 100%，Yrandom 为 0，预报量 Y 可以被精确地表示出来，就不再出现过拟合现象。但在实际样本的情况中，Fgiven 的值通常小于等于 0.3，即预报因子和预报量的相关系数为 0.5 左右，Fgiven 过高不符合实际情况。



---Forecast Y---	
1.00- 1.10	44.4%
1.10- 1.20	11.1%
1.20- 1.30	0.0%
1.30- 1.40	11.1%
1.40- 1.50	0.0%
1.50- 1.60	11.1%
1.60- 1.70	0.0%
1.70- 1.80	11.1%
1.80- 1.90	0.0%
1.90- 2.00	11.1%

---Forecast Y---	
1.00- 1.10	75.0%
1.10- 1.20	16.7%
1.20- 1.30	0.0%
1.30- 1.40	8.3%
1.40- 1.50	0.0%
1.50- 1.60	0.0%
1.60- 1.70	0.0%
1.70- 1.80	0.0%
1.80- 1.90	0.0%
1.90- 2.00	0.0%

---Forecast Y---	
1.00- 1.10	85.7%
1.10- 1.20	14.3%
1.20- 1.30	0.0%
1.30- 1.40	0.0%
1.40- 1.50	0.0%
1.50- 1.60	0.0%
1.60- 1.70	0.0%
1.70- 1.80	0.0%
1.80- 1.90	0.0%
1.90- 2.00	0.0%

---Forecast Y---	
1.00- 1.10	100.0%
1.10- 1.20	0.0%
1.20- 1.30	0.0%
1.30- 1.40	0.0%
1.40- 1.50	0.0%
1.50- 1.60	0.0%
1.60- 1.70	0.0%
1.70- 1.80	0.0%
1.80- 1.90	0.0%
1.90- 2.00	0.0%

图 21.　Fgiven 设为不同值时的过拟合结果