emaize 工作汇报

利用机器学习方法通过玉米基因型数据对表型进行预测

小组成员：陈旭鹏，徐隆琛

2017.7.31

我们两个人都认为，这一个月是我们人生中最充实、进步与收获最大的一个月。这一个月我们两个人合作非常紧密，互相讨论、帮助，共同解决一个又一个问题，因此我们也打算共同撰写这份的工作汇报。因为一直尝试到最后一天，时间紧迫，很多做的工作可能列的不完全或者有所疏漏，请老师见谅！

Acknowledgements

感谢史斌斌学长、胡博钦学姐给予的帮助。感谢鲁志老师提供的各种指导与帮助，使我们可以全身心地投入到课题中去，学习到非常多的知识、方法与技能。感谢 emaize 组委会的各位老师给予的帮助与指导。

1 介绍

1.1 背景

本次 e-Maize 挑战赛以玉米分子育种中“杂种优势”的遗传学问题为驱动，以大规模测交群体的“基因型”数据（高密度 SNP 标记）为输入，以多环境重复测量的“表型”数据为预测目标。大赛要求通过机器学习的方法，来建立一个预测效果足够好的模型，利用基因型精准预测表型，指导育种过程，从而大幅降低育种成本与耗时。大赛同时要求提交模型的 pcc 在 0.7 以上。

1.2 数据

本次竞赛涉及到一个玉米不完全双列杂交群体，共 6210 个杂交种。基因型数据为约 190 万有代表性的 SNP 标记，所有 6210 个杂交种的基因型数据都已给出，除去首行首列的索引，形成一个约 190 万行、6210 列的表格。表现型数据：目标性状为开花期、株高和产量三个性状，代表低度、中度和高度的杂种优势，这套杂交种表型 BLUP 数据，即为本次竞赛所用表型数据。其中公布了 4754 个杂交种，以用来供参赛选手建立模型，而剩下的 1456 个杂交种的表型数据则作为判断模型好坏的依据，并未公布出来。

1.3 目标

利用机器学习的方法，通过给出的 4754 个杂交种的基因型和表型数据，建立模型，并用剩下的 1456 个杂交种的基因型来预测其表型，希望得到一个跟真实表型足够接近的预测结果。

2 方法

2.1 研究问题

该项目的研究过程中我们遇到一系列需要解决的问题，主要概括如下：

A. 原始数据量很大，提供基因型数据约 190 万行，样本量为 6210 个，其中已知表型性状的样本量为 4754，需要预测的样本量为 1456。最大的挑战正是来自于大数据量造成的计算与时间开销。要寻找出合适的进行特征选择与数据降维的方法。

B. 各样本每个位点基因型分两个纯合型和一个杂合型，共 3 种，要对其进行向量化表示

C. 预测目标性状为连续值，机器学习模型种类众多，需要寻找合适的模型，寻找参数，预防过拟合。

2.2 工作流程概述

emaize 挑战赛的问题其实是一个非常有挑战的综合的问题，我们边学边做，遇到了很多问题，只有全部做完回过头看这一个月记录的充实的笔记，才能大概梳理清楚我们的工作轨迹。开始时我们只有一些基础的 linux 操作知识和 python 的基础知识，面对如此大数据量的问题，进展一度十分缓慢。但是我们选择多花时间，多读多试，从早上九点多到晚上一点，每天花费大约十四五个小时在实验室，等结果的时间不断学习和试验新方法，比如**原始数据较大**，直接处理时会遇到内存不足而报错 (Memory Error)，计算速度非常的慢，最初我们计算 ANOVA 第一折的值，就需要一整天的时间。我们通过搜索相关的解决方案，发现，基因

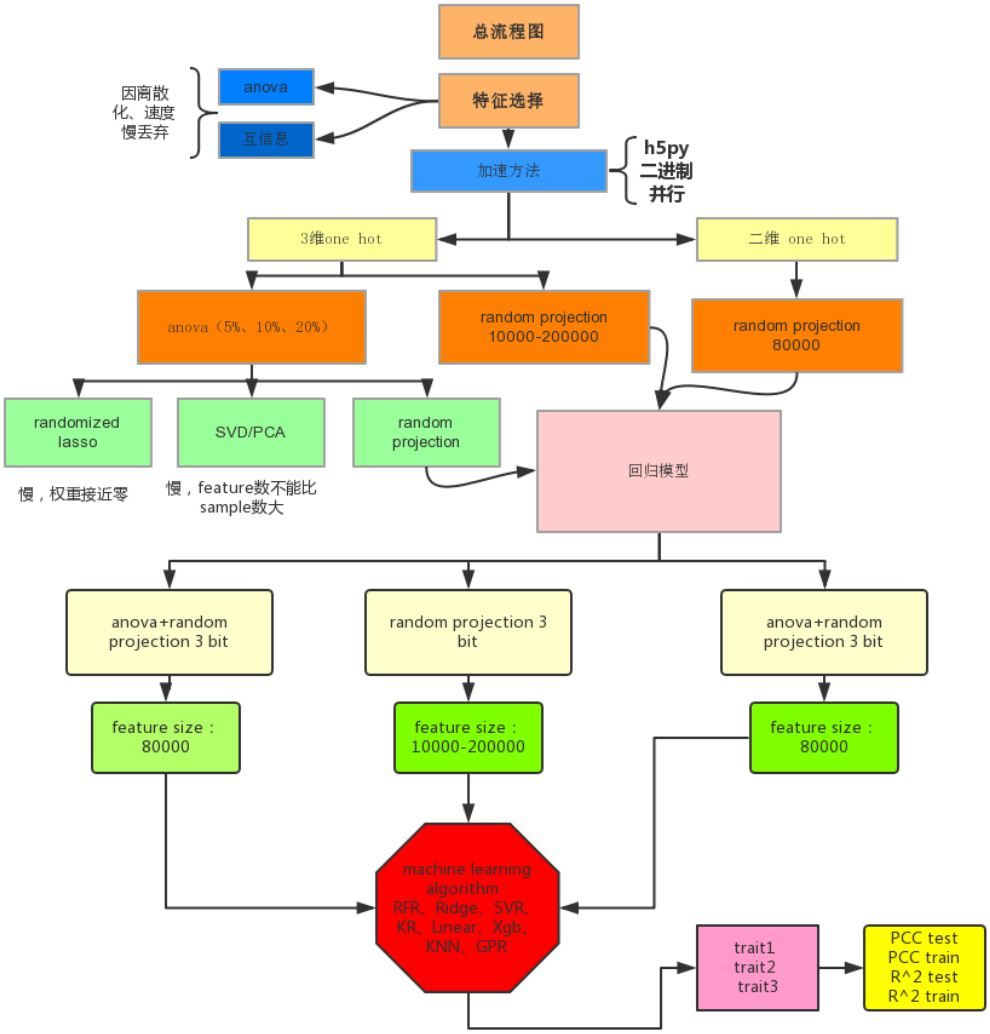
型数据的主体部分只有 0、1、2 三种，可以通过合理的二进制表示，将表示每个基因型的字节数压缩到 3bit 甚至 2bit，而不是一个字节，节省了内存使用量，同时找到了一种 python 的存储与读取文件格式 HDF5，彻底解决了时间花销的问题，将计算的速度提升了两百倍。我们还从斌斌学长那里要到了他做的一些项目的代码，学习如何规范化地书写代码，管理任务，并行计算，前期的大量的学习极大地提升了我们的效率，使我们后期可以大量的尝试各种方法，挑选出最好的结果。这道题目对于我们初学者来说，最大的困难就在于计算时间的漫长以及造成的多种方法尝试上的能力与时间问题，由于我们花费了大量的时间，而且在不断学习和尝试，因此虽然在解决问题的过程中，我们需要因为各种改变一次次地从头开始（比如更改数据结构，数据存储格式，更改计算阈值，更换完全不同的方法等），但是由于不断地学习，使得代码的体系化很好，使我们得以思路清晰，非常高效地解决遇到的各种困难，始终带着积极的心态勇敢地尝试新的模型和方法，取得了良好的结果。

不同问题的解决方案

研究问题	解决方案
数据量过大	数据降维，特征选择，优化数据存储格式
基因型表示方法	用单位正交向量来表示、包括二维与三维 one hot 向量
目标性状连续	采用多种机器学习回归模型
过拟合的预防及解决	10-fold cross validation，feature 量控制
筛选最优 feature 量	用不同 feature 量建立模型并预测，筛选出使得 pcc 最高的最低 feature 数量
筛选最佳模型	初步筛选出适合模型，再调参使得预测效果最优

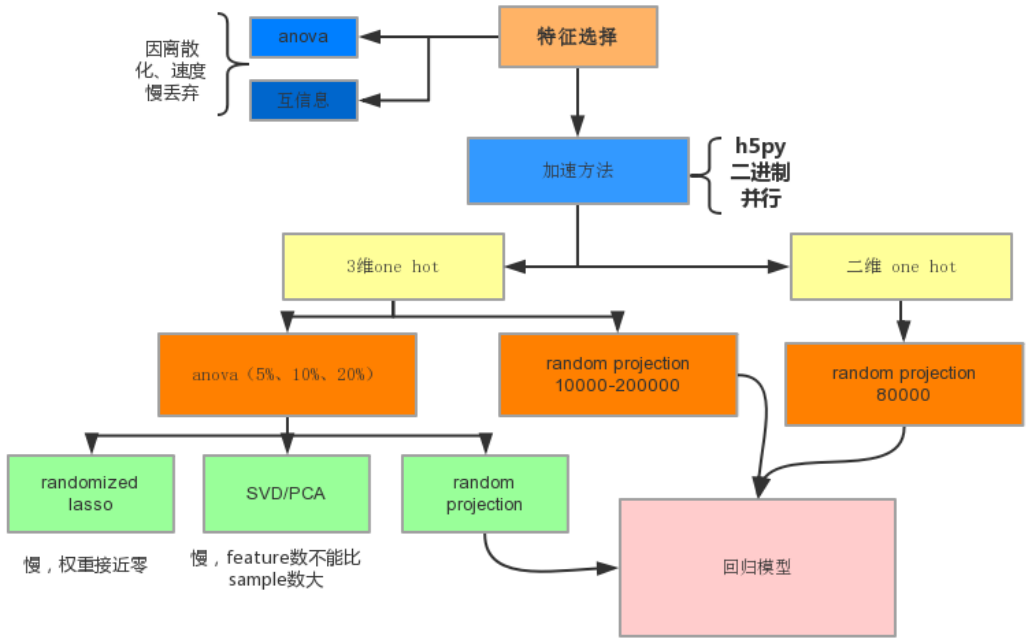
我们最终的工作流程分成了两个部分：特征选择、回归预测。下图是我们两

部分工作的基本流程，因为我们尝试了许多方法，进行甄别和挑选，最终选择出了通过二维向量化原始数据，随机映射（random projection）降维，局部敏感哈希删除冗余特征，高斯过程回归预测性状的方法。其他的一些回归模型依然可以在随机样本测试集上达到很好的效果。最后的两天我们又发现了一些有待以后解决的新的问题，这块儿内容会放在最后介绍。



3 具体流程

3.1 特征选择



A. 方差分析 ANOVA

1. 对于某 feature，基因型 0、1、2 对应变量Y_1、 Y_2、 Y_3进行分析
2. 原假设： Y_1、 Y_2、 Y_3同分布
3. p-value 小表示否认原假设，即说明Y_1、 Y_2、 Y_3有差异
4. **保留 p-value 较小的 feature**

我们分 trait1、trait2、trait3 对 190 万个 feature 分别做了 ANOVA，得到分布曲线如下。结果显示，trait1 和 trait2 的曲线很相似，有 20%的值接近于 0，trait3 的曲线则有所差别，ANOVA 的值更小，接近于 0 的 feature 量几乎达到 40%，这说明对于 trait3 而言，有效的 feature 含量可能更多。

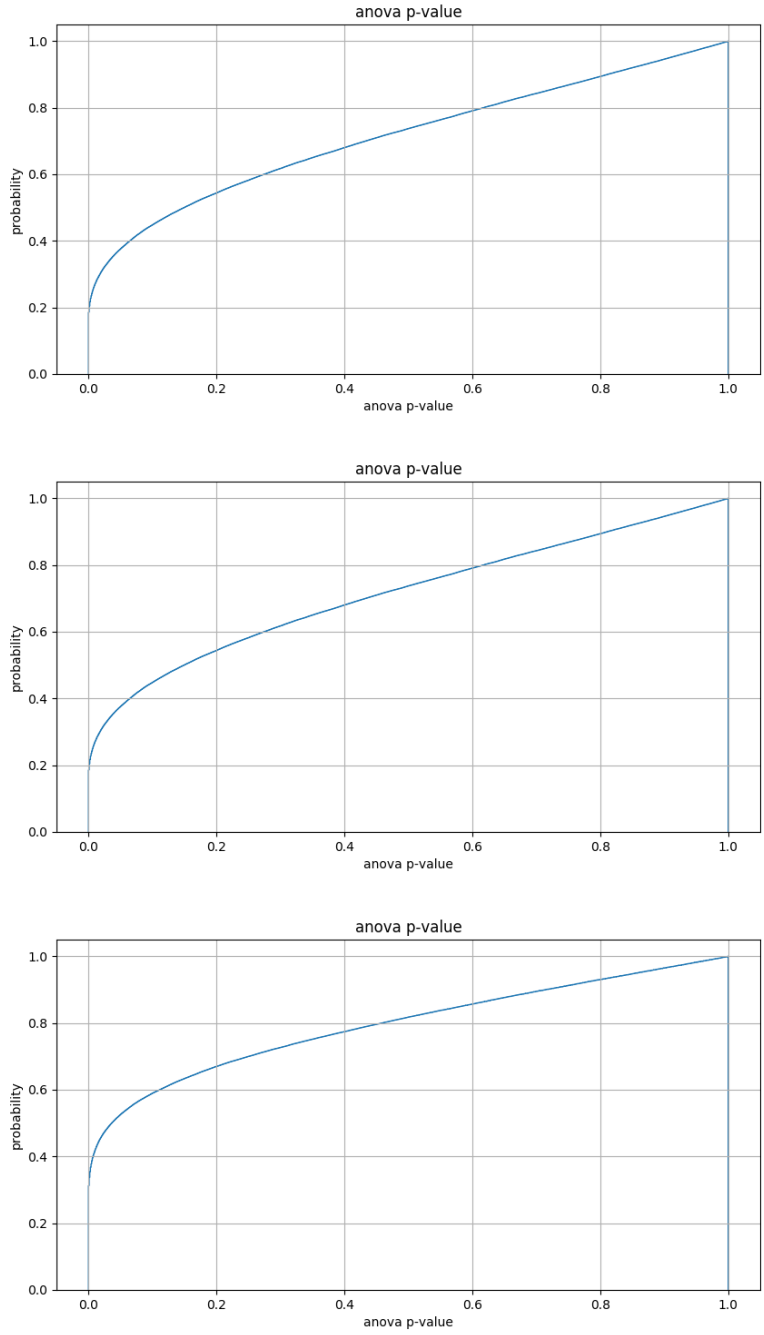


图 1.不同 trait 对应 ANOVA 的分布图：从上到下依次为 trait1, trait2, trait3.

B. 互信息 mutual information

1. 对性状连续值得到离散化 label
2. 计算某 feature 与 label 互信息
3. **保留适量互信息较大的 feature**

用互信息进行了初步处理，但考虑到需要预测的性状为连续值，而互信息需要先将性状人为而无足够依据地离散化，这一过程损失了很多信息，对预测不利。因此，我们后续步骤并未使用互信息的处理结果，而是选择了更有说服力的 ANOVA 所得到的结果来作为后续步骤的输入信息。

在开始时我们平行做了互信息和 ANOVA 两种方法，最终我们只使用了 ANOVA。同时我们在计算缓慢的时候思考如何加速的问题，通过搜索相关知识以及阅读斌斌学长其他项目的代码，我们发现了 h5py 存取格式，二进制处理以及并行化处理的方法，极大的提升了效率。因此在这里我们也介绍一下这几种方法

二进制：直接加载原始数据，经常出现内存不足报错的情况，通过分析 genotype 数据格式，我们发现每个 feature 对应的只有三种基因型，于是想到了对数据结构进行优化，将 AA、AT、TT 三种基因型分别表示为(1, 0, 0), (0, 1, 0), (0, 0, 1)的格式，并且将他们分别对应于 3bit 来进行存储，优化了大量内存，解决了初期经常出现的 MemoryError 问题；后来我们又发现，考虑到等位基因是否出现的思路更具有生物学意义，将 AA、AT、TT 分别表示为 10、11、01，同时得以用 2bit 来存放一个数据，进一步优化了数据存储结构。

h5py：最初我们用 txt 来存储信息，文件臃肿并且组织结构不佳，尤其是在有多层索引时表现不好，后来采用了 hdf5 格式存储文件，比较好的解决了这一问题。hdf5 文件为二进制格式，其存储类似于字典，能够方便索引到所需要的信息，并且可以针对读取，而不需要因其他信息而占用过多的内存。同时，通过导入 h5py 模块，能够很方便的对 hdf5 文件进行操作，节省了大量的时间和空间。

并行：初期我们的代码组织混乱，每次提交任务时都是写全整个代码，运行效率低下，当时仅计算第 0 折的 ANOVA 便耗时 1 天 。我们深感时间有限，通过向斌斌学长请教，学会了并行的方法，引入变量，通过文件导入任务列表，并行操作，后将计算所有 10 折的 ANOVA 所需的计算时间缩短到了 1 小时，效率提升 240 倍，这为我们之后能够尝试大量不同模型打下了很好的基础。

C. 广义线性模型 Lasso

1. 采用 L1 范数来规范化 Loss Function
2. 构造的模型是稀疏化的
3. 降维，可以用来进行特征筛选

我们筛选出 50%较小 ANOVA 值对应的 feature（约 290 万）来进行 Randomized Lasso，发现处理速度极慢，而且最终得到的结果不好，显示所有系数为 0；考虑到可能是由于预留的 feature 过多的缘故，又筛选出 5%最小 ANOVA 值对应 feature（约 29 万）来进行 Randomized Lasso，结果遇到了相同的情况。因此，我们认为 Lasso 可能并不适合该项研究，后续放弃了 Lasso，而选择其他进一步降维的方法。（后面的尝试发现，如果要使用 lasso，必须提前删除相似性高的 feature，但是 ANOVA 并没有这样的功能，而 random projection 可以做到）

3.2 数据降维

A. SVD/PCA

1. 将数据矩阵 A 变换为其奇异值矩阵 Λ 和两个矩阵的乘积 A=PΛ Q

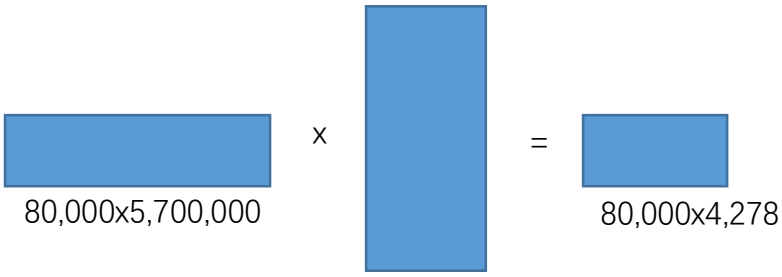
2. 通过取前 k 个奇异值得到 Λ 前 k 行 k 列构成的矩阵 Λ_k，来对 A 进行降维
3. 相当于将 A 映射到 k 维空间上，并且保存足够多的原始信息

对于 5%最小 ANOVA 值对应 feature 进行 SVD 处理，结果显示运行速度较慢，关键是由于 SVD 本身的原理，决定降维之后得到的数据矩阵的阶数（行数或列数）不能超过原数据矩阵的阶数。由于 training set 的样本数只有 4278，即数据矩阵的列数为 4278, 导致经过 SVD 处理后得到的 feature 数最多只能有 4278 个。我们认为 4278 个 feature 并不能很好地反映原始数据 570 万个 feature 的信息，因此去探索了一些其他类似的降维方法，如 PCA 和 Random Projection. 我们发现 PCA 和 SVD 原理一致，只是算法实现不同，并不能解决我们的问题，从而重点研究了 Random Projection.

B. 随机映射 Random Projection

1. 把高维空间的点线性映射到低维空间，可以自由选择低维空间维度
2. 降维前后两个样本点间距离几乎保持不变
3. 在处理稀疏数据时计算复杂度低
4. 通过局部敏感哈希(LSH)将相似的 feature 映射到同一个数，并剔除掉相似 feature。

Random Projection 降维示意图如下所示。



3.3 数据结构

A. 3 维 one-hot 向量

1. 把三种基因型分别表示为$(1,0,0)^T, (0,1,0)^T, (0,0,1)^T$
2. AA：$(1,0,0)^T$，AT：$(0,1,0)^T$，TT：$(0,0,1)^T$
3. 每个位点信息储存在 3bit 中，节省空间
4. 表示方法代表三种基因型重要性比 0、1、2 合理

B. 2 维 one-hot 向量

1. 根据等位基因是否出现，把三种基因型分别表示为 10，11，01
2. AA：$(1,0)^T$，AT：$(1,1)^T$，TT：$(0,1)^T$
3. 每个位点信息储存只需 2bit
4. 更加符合生物学意义，三种基因的距离关系可以更好地反应其生物学关系

在后期我们发现二维向量可以更好地表示基因型的相互关系，因此在后期我

们从头开始处理了基因型。将其首先向量化为二维 one-hot 向量。

预处理结果

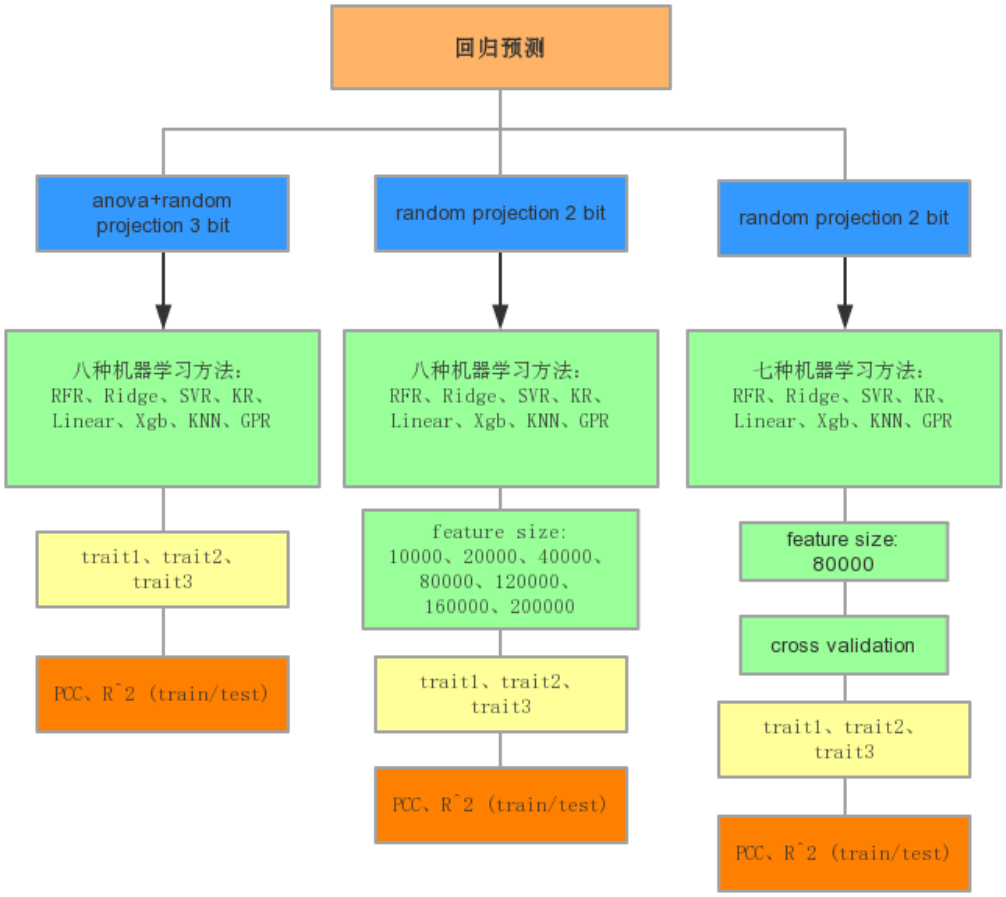
我们经过特征选择后的材料共有三种，在我们不断的尝试中，我们舍弃了一些方法（比如互信息、lasso、SVD 和 RFE、GBA 这样的无效方法），选择了更快也更优的方法获得预处理的数据：

A. **三维 one hot 向量表示+ANOVA + random projection**，我们首先将基因型通过三维向量表示，feature 数量约为 570 万，通过 ANOVA 筛选留下 5%的 feature，再通过 random projection 的方法取出冗余 feature，进一步降维至八万个 feature。

B. **三维 one hot 向量表示+ random projection**，通过前面的分析，我们发现通过 ANOVA 筛选必须留下大约 20%-40%的 feature 才有用，不但速度慢，而且效果可能没有 random projection 好。由于 random projection 同时具有删除相似特征以及降维的作用，效果应该会比 ANOVA 更好，而且删除相似特征的作用是 ANOVA 没有的，这个非常关键，之前没有使用 random projection，而使用 ANOVA + lasso 组合的时候，lasso 就会因为特征相似性太高导致过拟合，导致不得不放弃。后面的结果也证明 B 是比 A 更好的方法。

C. **二维 one hot 向量表示+ random projection**，由前面的分析，二维向量显然可以更好地表达 SNP 的相互关系，使得结果更好。后面的结果证明 C 是比 B 更好的方法。

3.4 回归预测



在花费大量的功夫寻找足够优异的、速度非常快的特征选择与降维方法后，我们现在有非常多的回归方法可供选择和比较。

我们进行了大量的尝试，学习、筛选出了多个适合做回归分析的模型，进行了多轮调参和交叉验证，同时我们在不同的预处理样本上进行了回归预测。得益于前期充分的准备工作，我们把回归部分的速度提升了非常多，每种方法回归预测的时间都只需要几分钟即可（测试数据：基于 random projection 方法的二维

化基因型数据，feature 数量为 80000）。

下面是我们的结果呈现，我们使用了三种不同的预处理结果，以及八种回归模型，最终我们选择了二维化基因型矩阵通过 random projection 方法筛选出 80000 个 feature 的材料作为回归的输入。由于运行和计算效率很高，我们尝试了各种组合，统计了各种情况下的 test 集以及 train 集上的 pcc 与 r^2。在正文部分只展示重要表格与结果，其余材料放在补充材料中。

A. 模型介绍

我们采用了八种回归模型来做预测，分别为：Linear Regression, XGBoost, Ridge, Kernal Ridge, KNN regression, Random Forest, Gaussian Process, SVR。其中 XGBoost 方法使用了专门的 package，其余七种方法均来自于基于 python 的 sklearn

1. Linear Regression

   通过建立线性模型来拟合数据，Loss Function 是 $\min\|X\omega - y\|_2^2$，优点是实现简单，回归速度快，在本题中作为一种基准，来衡量其他方法的好坏。但是结果表现一般，也无法调参。

2. Ridge

   是广义线性模型的一种，Loss Function 是 $\min\|X\omega - y\|_2^2 + \alpha\|\omega\|_2^2$，加入了正则化项预防过拟合，优点是可以控制权重大小，速度较快，稳定性比线性回归更高，缺点是不能稀疏化。

3. Kernel Ridge

加入了核函数的岭回归，通过 kernel trick，可以拟合更复杂的函数关系，在本题中，经过适当的选择，kernel ridge 可以达到非常高的 pcc。

4. Random forest

   使用了 bagging + decision trees 的方法，将决策树作为 base estimator，然后采用 bagging 技术训练一大堆小决策树，最后将这些小决策树组合起来，这样就得到了随机森林。缺点在于参数太多，调参比较麻烦，调参不当容易出现过拟合。

5. XGBoost

   基于 boosting 提升方法，GBDT(Gradient Boosting Decision Tree) ,是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终答案。它在被提出之初就和 SVM 一起被认为是泛化能力较强的算法。xgboost（eXtreme Gradient Boosting）可以说是提升方法的完全加强版本。xgboost 算法在各大比赛中展现了强大的威力,被广泛地应用于机器学习各种竞赛（如 kaggle 的机器学习竞赛）中。Xgboost 可以自行迭代，不断优化结果，速度很快，但是参数也比较多，调参比较繁琐。

6. SVR

   支持向量机是神经网络大规模应用前最有效的方法之一，在解决小样本、非线性及高维模式识别中表现许多特有优势。通过支撑向量与超平面实现分类与预测问题。通过核技巧可以将样本映射到更高维空间，并且包含正则化选项保证稀疏性以及预防过拟合。缺点是计算速度比较慢，在所有的八种方法中，只有 SVR 无法被优化到在个位数分钟内完成。并且由于本身已有较多的特征，SVR 容易出现过拟合现象，由于计算略显得慢，不适合调参。

7. KNN

K 最邻近算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN 算法本身简单有效，它是一种 lazy-learning 算法，分类器不需要使用训练集进行训练，训练时间复杂度为 O(n)。方法的缺点是提升空间较小，KNN 中只有一个可优化的参数 K，如果太小，则最近邻分类器容易受到训练数据的噪声而产生的过分拟合的影响；相反，如果 K 太大，最近分类器可能会误会分类测试样例，因为最近邻列表中可能包含远离其近邻的数据点。因此该方法的潜力并不是最大的。

8. Gaussian Process regression

高斯过程回归是基于贝叶斯理论和统计学习理论发展起来的一种机器学习方法,有着严格的统计学习理论基础,适于处理高维数、小样本和非线性等复杂回归问题与神经网络和支持向量机相比，该方法具有容易实现、超参数自适应获取以及输出具有概率意义等优点。如果一组随机变量都是服从高斯分布的，这组随机变量就是高斯过程高斯过程是在贝叶斯线性回归的基础上，把自变量空间通过核函数映射到高维空间实现的。其核函数和 SVM 类似，由于时间关系，我们使用 RBF 核函数时经过粗略的调参效果并不理想，但是使用较为简单的 DotProduct 核函数时，获得了最高的 pcc 值之一。

B. 模型结果

我们使用了多种材料，可以分为三大类别，进行了很多次的回归尝试。我们选取了两种指标，一种是题中要求的相关系数相关系数 pcc:

$$p = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

另一种是决定系数 r^2-score:

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}$$

同时我们不仅需要计算 test 集上的两个指标，还需要计算 trian 集上的两个指标，来考虑是否存在过拟合的问题，如果 trian 集上指标很高，而 test 集上指标较低，则存在过拟合的可能性。

1. 在三维 one hot 向量表示+ANOVA + random projection 数据上的结果

   feature 保留个数为 80000

		RFR	Ridge	SVR	KR	Linear	Xgb	KNN	GPR
Trait1	PCC test	0.917	0.937	0.955	0.937	0.937	0.938	0.859	0.957
	R2 test	0.839	0.876	0.909	0.876	0.876	0.878	0.799	0.915
	PCC train	0.986	1.000	0.959	1.000	1.000	0.967	0.932	1.000
	R2 train	0.969	1.000	0.919	1.000	1.000	0.933	0.902	1.000
Trait2	PCC test	0.864	0.918	0.936	0.918	0.918	0.913	0.794	0.939
	R2 test	0.724	0.843	0.872	0.843	0.842	0.828	0.692	0.882
	PCC train	0.978	1.000	0.948	1.000	1.000	0.956	0.896	1.000
	R2 train	0.951	1.000	0.896	1.000	1.000	0.910	0.836	1.000
Trait3	PCC test	0.709	0.758	0.809	0.758	0.757	0.797	0.687	0.821
	R2 test	0.501	0.557	0.651	0.557	0.556	0.633	0.440	0.673
	PCC train	0.961	1.000	0.844	1.000	1.000	0.895	0.830	1.000

	R2 train	0.911	1.000	0.707	1.000	1.000	0.789	0.683	1.000

可以看到各个模型都有非常好的表现结果，尤其是 GPR，在三种性状的预测上均取得了最佳结果。

2. 在三维 one hot 向量表示+ random projection 数据上的结果

这次我们剔除了 ANOVA，只是用 random projection，同时我们在考虑一个问题，究竟保留多少 feature 才是最好的，我们一直都认为 RFE 在此题中并不是一个可行的方法，因为严格意义上的 RFE 只有做到最后一步回归才可以用来返回筛选 feature，即使对于我们使用了多种优化方法与很快的模型后，也有极大的计算花销，实际上我们只需要在以万为变化幅度的基因型上进行测试，即可筛选出我们需要的最好的 feature 个数。我们在测试了 10000、20000、40000、80000、120000、160000 与 200000 个 feature 下的结果后，发现在 80000 时已经达到了非常好的结果，再继续增加 feature 数量，对于最后的表现的提升已经需要在小数点五位以后才可以看到（见补充材料）

feature num: 80000

		RFR	**Ridge**	**SVR**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait 1	PCC test	0.885	0.958	0.954	0.958	0.958	0.939	0.415	0.962
	R2 test	0.728	0.919	0.857	0.919	0.919	0.868	0.159	0.926
	PCC train	0.964	1.000	0.995	1.000	1.000	0.968	0.764	1.000
	R2 train	0.913	1.000	0.985	1.000	1.000	0.935	0.500	1.000
Trait 2	PCC test	0.856	0.946	0.934	0.946	0.946	0.924	0.623	0.947
	R2 test	0.696	0.894	0.829	0.894	0.894	0.843	0.387	0.897
	PCC train	0.973	1.000	0.995	1.000	1.000	0.959	0.802	1.000
	R2	0.933	1.000	0.984	1.000	1.000	0.917	0.632	1.000

	train								
Trait 3	PCC test	0.723	0.822	0.824	0.822	0.822	0.811	0.494	0.832
	R2 test	0.506	0.675	0.650	0.675	0.674	0.648	0.146	0.692
	PCC train	0.958	1.000	0.986	1.000	1.000	0.902	0.729	1.000
	R2 train	0.894	1.000	0.962	1.000	1.000	0.799	0.474	1.000

最优的方案是在 80000 个 feature 上进行优化，因此接下来我们都按照这个要求来筛选。不再尝试其他的 feature 数量了

3. 在二维 one hot 向量表示+ random projection 的结果

我们在 80000feature 上多次调参，将最终的十折交叉验证的平均值结果展示如下，具体的十折结果见附录。由于 SVR 的运算时间比较慢（几十分钟），因此此方法被我们舍弃。

Cross validation average

		RFR	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait 1	PCC test	0.882	0.953	0.953	0.953	0.932	0.390	0.956
	R2 test	0.964	1.000	1.000	1.000	0.968	0.763	1.000
	PCC train	0.745	0.907	0.907	0.907	0.862	0.139	0.913
	R2 train	0.912	1.000	1.000	1.000	0.934	0.500	1.000
Trait 2	PCC test	0.842	0.940	0.940	0.940	0.915	0.634	0.943
	R2 test	0.973	1.000	1.000	1.000	0.960	0.806	1.000
	PCC train	0.684	0.882	0.882	0.882	0.830	0.399	0.889
	R2 train	0.933	1.000	1.000	1.000	0.918	0.637	1.000
Trait 3	PCC test	0.727	0.820	0.820	0.820	0.798	0.471	0.829

	R2 test	0.959	1.000	1.000	1.000	0.903	0.727	1.000
	PCC train	0.509	0.670	0.670	0.670	0.628	0.085	0.686
	R2 train	0.895	1.000	1.000	1.000	0.801	0.468	1.000

我们确实在二维+80000feature 上面取得了最佳结果，因此到此为止，我们不但尝试了各种回归模型，而且完全确认了我们特征选择的有效性。之后可以看到，我们的回归模型是有一定的问题的，但是特征选择与降维部分的工作，可以说我们已经完满地解决了。

C. 调参

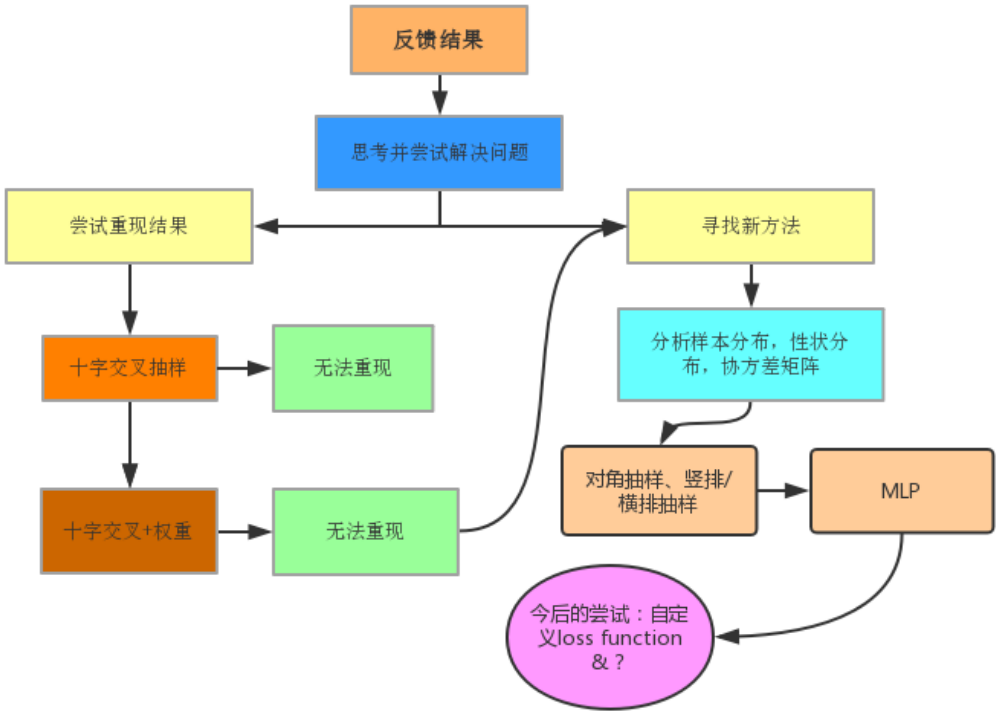
由于时间所限，没有使用非常规范的大规模调参方法，但是在调参部分也进行了相当多的学习和探索。

我们主要进行了高斯过程和 kernel ridge 的参数调整，进一步提升了结果。同时学习了 sklearn 的两种调参技巧：grid search 和 randomized search。我们又进一步地发现了一个强大的调节超参数专用的 package： hyperopt 开源库提供了超参数自动优化算法和软件架构，可应用于 DNN、CNN 和 Scikit-learn 开源机器学习库。hyperopt 提供了 Python 接口，使用简单，指定需要搜寻的参数空间、需要最小化的目标函数、优化时采用的优化算法和数据集即可。hyperopt. hyperopt 提供了 xgboost 的调参方法，并且通过 hpsklearn，可以更加方便地用来调节 sklearn 中的模型超参数。

3.5 最终探索

至此为止，可以说我们已经‘圆满’地完成了所有的工作，两个性状的 pcc 预测已经达到了 0.96，而最难预测的 trait3 的 pcc 也已经可以达到 0.82。可是当我们在 29 号获得反馈的时候，结果却让我们惊讶：我们的模型的预测结果相当的不好，与此同时，其他已经提交的组无一例外都拿到了很低的 pcc 的分数，没有一个组能够达到 0.7。

因此最后的三天时间，我们又推迟了撰写报告的时间，抓紧时间进行了大量的分析和探索，试图解决这几个问题。我们的分析和探索主要分为以下几个方面：



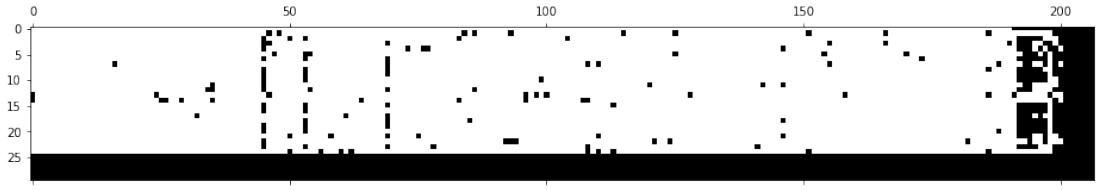
1. 分析样本分布、亲本母本与性状的关系、基因型的相关性等。
2. 通过分析样本的分布，创新地使用新的抽样方法重新进行 cross validation 来重现结果。
3. 为了彻底解决这次挑战赛的问题，我们使用了两种全新的模型，重新进行预测。

A. 进一步分析样本

如前所述，如果本次大赛提供的训练集与最终测试集是随机选择的，那么我们的回归模型已经可以给出一个相当完美的答案。但是本次大赛的样本划分相当特殊，造成了所有参赛人员的结果都非常不好。我们在获得结果反馈后，进行了两方面的准备：一是分析样本，用特殊的方法重现结果，与组织方确认是否是数据或者方法的问题。二是思考全新的能够彻底解决这个困难的方法。我们首先重新细致地分析了样本的情况。

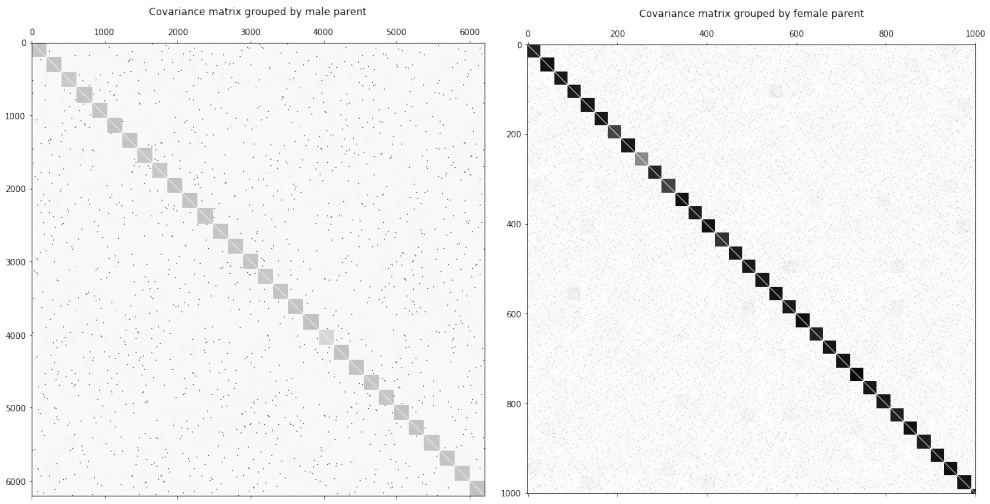
我们首先利用 matplotlib 进行了一些可视化的分析。

1. 样本分布图



我们利用父本和母本的杂交信息，重新绘制了样本的分布。通过这个样本分布图我们可以发现，组织方在划分训练集与测试集的时候，有意地选择了特殊的区域，这种非随机性是造成所有队伍结果均不理想的根源。

2. 父本母本基因型的协方差矩阵



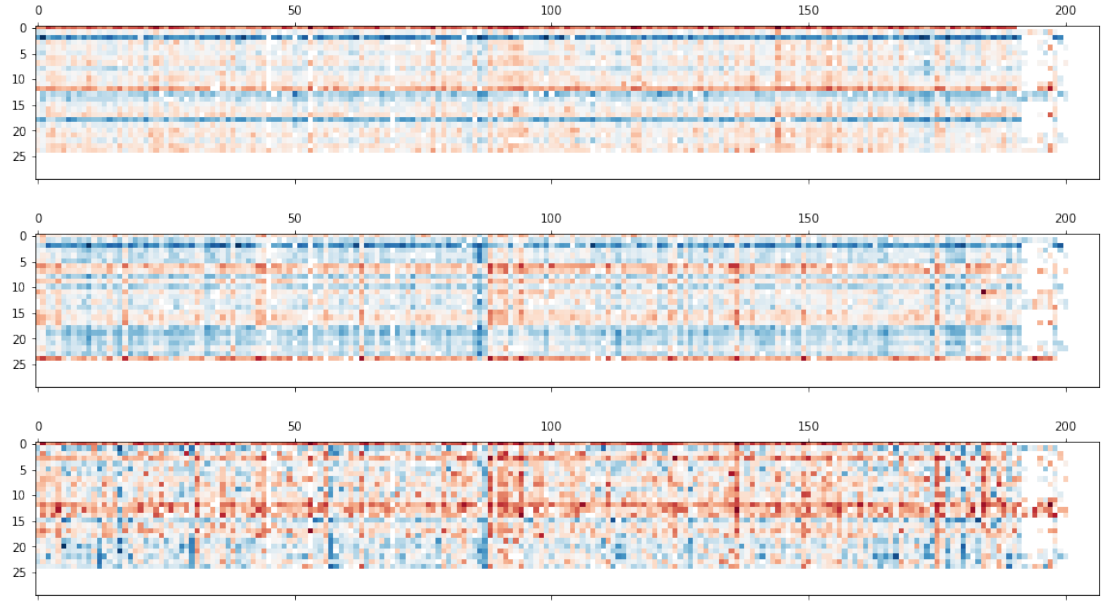
我们分析了基因型按照父本母本来表示的协方差矩阵，

协方差 correlation variance 可以用来判断相关性，公式如下，

$$Cov(X,Y) = E\big((X-\mu_X)(Y-\mu_Y)\big)$$

可以发现对角线，即来自同一父本或者母本的基因型经过预处理之后相关性非常高，说明用基因型可以比较准确地区分样本是否来自同一个父本或者母本

3. 性状数据分布



我们按照父本、母本对性状信息绘制了 heat map 图像，横坐标为母本，纵坐标为父本。发现对于 trait1 和 trait2，有比较明显的性状与父本的关联性，对

于 trait3，这种关系显得比较混乱。但是在反馈的结果中，所有组的 trait3 的预测结果反而比 trait1 和 trait2 好，这是相当反常的，因为在我们之前所有的回归模型中，trait3 的结果都是最难预测，表现最差的（trait3 有 0.82 左右，trait1 和 trait2 可以达到 0.96 左右）。

基于上面的分析，我们明确了两个方案，一是想办法在我们自己的 training set 上模拟组织方的划分方式进行交叉验证，而不是通常的机器学习模型所采用的随机划分十折交叉验证，来试图重现这个比较奇怪的结果，确认不是数据的问题。第二是思考新的解决问题的方法。

B. 采取新的抽样方式重现结果

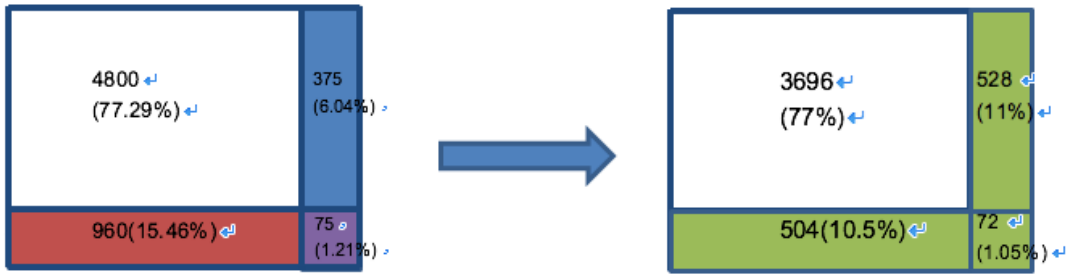
为了能够重现这种样本划分方式，我们决定在自己的训练集上重新进行交叉验证。为此我们设计了一种新的取样方式，十字交叉采样 (cross sampling)，示意图如下：

通过八次十字交叉采样，可以比较好地重现样本的分布特点，更好地模拟样本的划分。这次我们依然使用之前表现最好的高斯过程回归来进行交叉验证。结果依然可以达到 0.8 左右的 pcc，并不能重复出结果。

	0	1	2	3	4	5	6	7	平均
trait1	0.789	0.821	0.776	0.861	0.836	0.845	0.881	0.802	0.826
trait2	0.726	0.840	0.785	0.881	0.752	0.577	0.890	0.843	0.787
trait3	0.676	0.605	0.630	0.582	0.664	0.594	0.619	0.569	0.617

我们进一步研究发现，为了更加逼真地重现最终的预测，我们还要分析我们的十字交叉采样与最终的样本分布的比例问题。我们应该重新调整权重，避免在某一块儿预测很好而另一块儿预测很差导致的问题。

按照对角线抽样的方法，我们会将数据集分成所占比例如右的四个部分，$\begin{bmatrix} 0.77 & 0.11 \\ 0.105 & 0.015 \end{bmatrix}$，而按照组委会的划分方式，各部分所占权重是不同的，如右，$\begin{bmatrix} 0.81 & 0.028 \\ 0.16 & 0.006 \end{bmatrix}$。



因此我们重新计算了 pcc 的加权平均值。我们修改了代码，对三个区域分别预测，再求加权的八折验证的 pcc 平均值。分别为 0.802，0.752，0.627，依然重复不出来反馈的结果。通过详细的表格还可以看到，在同一个父本或者同一个母本的区域，pcc 并没有预测的较大差别，说明结果差也并不是因为模型在某个区域表现的比较差的原因。（f:蓝色区域，m:红色区域，fm:紫色区域）

trait1	f	0.838	0.809	0.83	0.89	0.86	0.864	0.779	0.853

trait1	fm	0.409	0.688	0.677	0.483	0.315	0.746	0.883	0.389
trait1	m	0.764	0.86	0.734	0.825	0.8	0.843	0.947	0.696
trait2	f	0.834	0.816	0.798	0.849	0.795	0.85	0.857	0.874
trait2	fm	0.576	0.543	0.588	0.761	0.538	0.149	0.821	0.397
trait2	m	0.738	0.894	0.74	0.906	0.764	0.196	0.939	0.833
trait3	f	0.624	0.559	0.593	0.621	0.677	0.658	0.644	0.639
trait3	fm	0.435	0.445	0.629	0.224	0.029	0.408	0.163	0.063
trait3	m	0.719	0.698	0.692	0.606	0.631	0.609	0.618	0.565

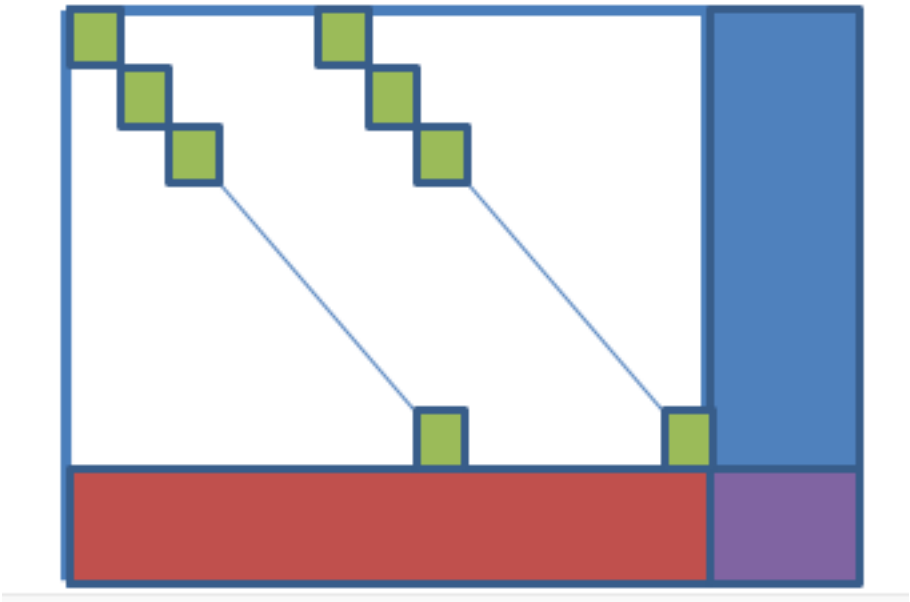
C. 寻找全新的方法解决问题

在和组委会多次交流并思考后，我们认为必须使用全新的模型来解决样本分布不随机的问题。从未来应用的角度来讲，这种新的方法也确实十分有必要。

本次大赛的根本宗旨是通过基因型数据来预测表型，但是显然之前所有人的失败的模型都没有真正地学习到这种规律。通过前面对基因型的协方差矩阵分析，**我们找到了问题的关键：模型是根据基因型的相似性来预测的，相似性比较好就可以预测准，如果父本母本都不一样，相似性差，预测的就差。**我们认为，之前的模型，通过基因型信息，将父本与母本的信息学习得太好了。而当模型进行预测，面对父本或者母本甚至双方基因型未知的情况时，表现就会很糟糕。为了解决模型学习信息过多而产生的在训练集表现很好，而在测试集表现不好的问题。我们设计了新的模型，使用新的抽样与训练方法。

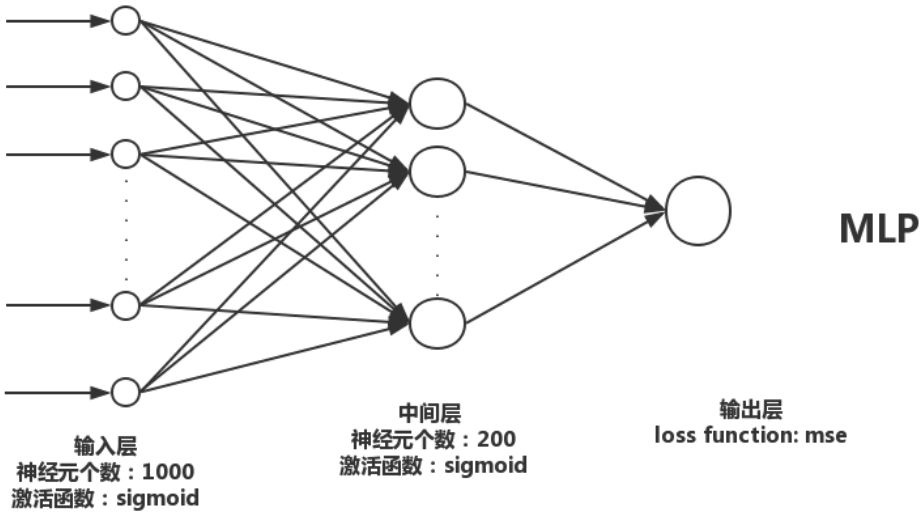
为了实现在苛刻的条件下训练模型的要求，我们设计了一种对角线抽样(diag samling)的方法，这种对角线抽样可以保证样本的父本与母本均不同，在这样的

情况下，如果模型可以表现得足够好，则可以说明模型可以实现完全基于基因型信息，不依赖父本、母本信息的要求。



但是对于整个训练集，由于只有 25*192 个样本，对角线抽样只能最多抽出 25 个父本母本完全不相干的样本。这样的样本集有 192 个，因此我们设想，可以不像普通的回归模型那样一次输入样本，而是通过很多轮迭代将样本分别送进去，每次送进去 25 个样本。这样的要求可以通过神经网络来实现。但是考虑到我们的样本数量非常少，显然不能应用大型的网络，而且之前在尝试多种回归模型时，我们发现最简单的线性回归已经取得了很好的效果，因此我们认为浅层（两层或者三层）的全连接网络已经足够。

我们尝试了两层、三层的 MLP 网络，尝试了不同的激活函数和正则化系数。



同时我们又提出了一种新的样本选取方式。对于本题而言，我们可以继续分开考虑测试集上的母本相同区域和父本相同区域，为了分别在这两个区域上达到很好的效果，我们可以采取多轮的竖排抽样与横排抽样，让模型学习这种父本相同或者母本相同的规律，来建立六个模型分别预测父本相同或者母本相同的区域。对角线与横排、竖排的两种样本选择方法我们都在平行地尝试。

我们直到 7.30 号才开始思考、尝试这种方法，在初步的模型搭建与调参后发现还有大量的工作要做。因为直接搭建 MLP，模型改动还不大，我们遇到了

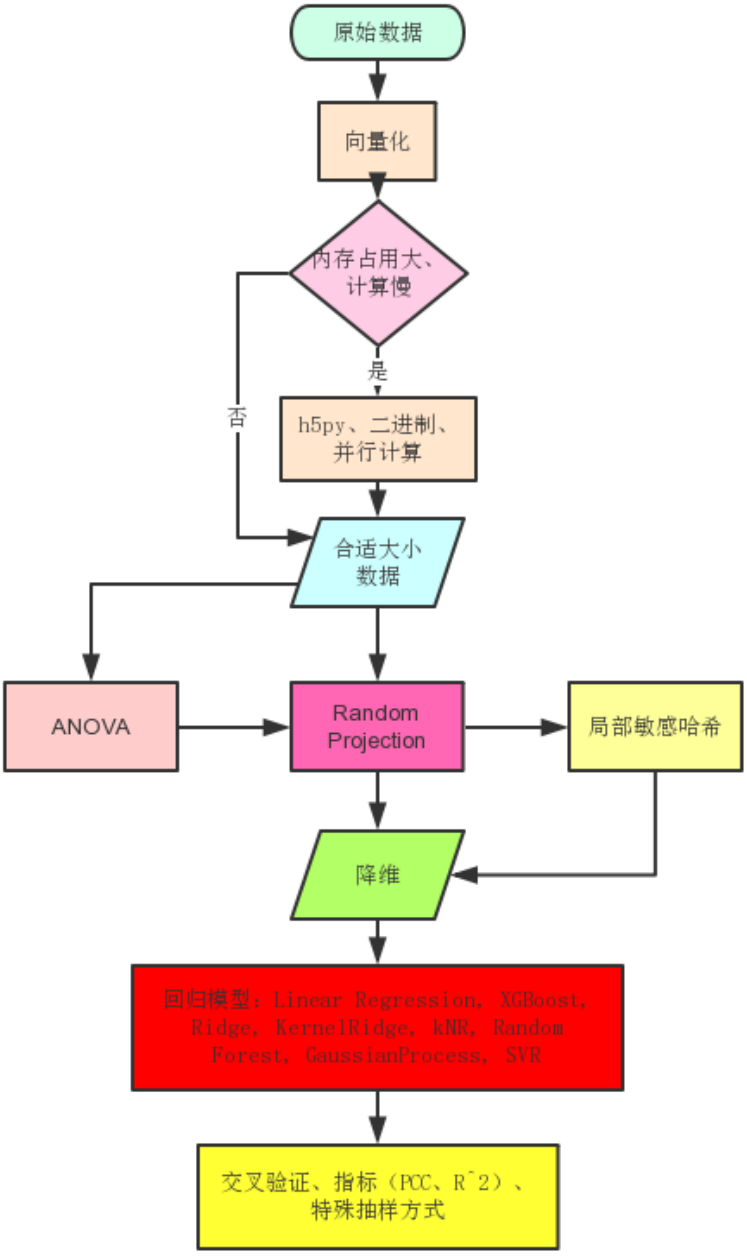
loss fuction（MSE 为指标）无法下降到接近零的问题，一直在 1 左右徘徊。说明我们预期模型可以学到的规律并没有那么简单被学到，因为性状值为方差为 1 的标准正态分布，模型会直接将所有的性状值预测为均值后，loss 就不再下降。之后我们要做几项工作，一是重新核查网络，检查每层的输出，二是尝试其他的方法，一个真正可能解决问题的方法是不能直接使用现成的目标函数，二是更改改目标函数，直接提现出亲本的不同。这是接下来如果继续工作的方向。

4 补充材料

4.1 Method

通过这次题目，我们总结出了一套较为有效的从特征选择、加速处理到回归预测的工作流程。其中如二进制、h5py 存取，random projection 用于特征选择和降维、xgboost、gaussian process 等方法和模型，对于解决我们的问题很有帮助，也值得借鉴和整理。为了彻底解决这个问题，我们还创新性地使用了一些特殊的抽样方式，训练方式，也可以作为解决特殊问题的借鉴。

我们整理了一个简单的流程图，以及所有的方法名称，各种方法的具体内容都可以在正文中找到。



方法名称：

互信息 （mutual information）

方差分析（ANOVA）

转化为二进制文件

h5py 存取 HDF5 格式文件

随机映射（random projection）

局部敏感哈希降维（LSH）

奇异值分解、主成分分析（SVD/PCA）

线性回归（Linear regression）

岭回归（Ridge）

核岭回归（Kernel Ridge）

随机森林（Random Forest）

XGBoost

K 最邻近法（KNN）

支持向量回归（SVR）

高斯过程（Gaussian Process）

sklearn 调：grid search 与 randomized search

hyperopt 与 hpsklearn 调参

协方差矩阵分析、热图分析 （covariance matrix、heat map）

对角抽样、横竖排抽样

全连接神经网络（MLP）

4.2 表格与图片

补充在正文中没有展示的表格和回归预测的散点图

表格

（1） 在三维 one hot 向量表示+ random projection 数据上的结果，用模型测试不同的 feature 数量下的结果，最终选择了 80000 作为 feature 数量。

10000

		RFR	Ridge	SVR	KR	Linear	Xgb	KNN	GPR
Trait 1	PCC test	0.875	0.927	0.332	0.927	0.927	0.931	0.411	0.960
	R2 test	0.724	0.857	0	0.857	0.857	0.852	0.146	0.922
	PCC train	0.977	1.000	0.944	1.000	1.000	0.962	0.742	1.000
	R2 train	0.941	1.000	0.774	1.000	1.000	0.920	0.485	1.000
Trait 2	PCC test	0.828	0.907	0.455	0.907	0.907	0.912	0.619	0.943
	R2 test	0.654	0.819	0.000	0.819	0.819	0.819	0.375	0.889
	PCC train	0.971	1.000	0.962	1.000	1.000	0.954	0.817	1.000
	R2 train	0.928	1.000	0.855	1.000	1.000	0.903	0.647	1.000
Trait 3	PCC test	0.699	0.737	0.513	0.737	0.737	0.792	0.434	0.831
	R2 test	0.471	0.493	0	0.493	0.493	0.614	0.057	0.690
	PCC train	0.963	1.000	0.964	1.000	1.000	0.894	0.702	1.000
	R2 train	0.903	1.000	0.843	1.000	1.000	0.779	0.433	1.000

20000

		RFR	Ridge	SVR	KR	Linear	Xgb	KNN	GPR
Trait 1	PCC test	0.889	0.951	0.342	0.951	0.951	0.939	0.398	0.962
	R2 test	0.750	0.904	0.006	0.904	0.904	0.866	0.125	0.925
	PCC train	0.978	1.000	0.945	1.000	1.000	0.963	0.767	1.000
	R2 train	0.944	1.000	0.776	1.000	1.000	0.924	0.508	1.000
Trait 2	PCC test	0.848	0.935	0.442	0.935	0.935	0.916	0.641	0.946
	R2 test	0.677	0.874	0.018	0.874	0.874	0.821	0.408	0.894

	PCC train	0.973	1.000	0.964	1.000	1.000	0.957	0.824	1.000
	R2 train	0.932	1.000	0.860	1.000	1.000	0.910	0.667	1.000
Trait 3	PCC test	0.721	0.778	0.499	0.778	0.778	0.803	0.498	0.827
	R2 test	0.501	0.590	0.019	0.590	0.590	0.628	0.180	0.684
	PCC train	0.962	1.000	0.966	1.000	1.000	0.897	0.742	1.000
	R2 train	0.905	1.000	0.850	1.000	1.000	0.787	0.511	1.000

40000

		RFR	**Ridge**	**SVR**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait 1	PCC test	0.884	0.954	0.586	0.954	0.954	0.932	0.341	0.961
	R2 test	0.740	0.911	0.174	0.911	0.911	0.857	0.084	0.923
	PCC train	0.978	1.000	0.954	1.000	1.000	0.964	0.738	1.000
	R2 train	0.946	1.000	0.812	1.000	1.000	0.927	0.479	1.000
Trait 2	PCC test	0.846	0.942	0.642	0.942	0.942	0.914	0.577	0.947
	R2 test	0.686	0.888	0.245	0.888	0.888	0.824	0.325	0.897
	PCC train	0.973	1.000	0.975	1.000	1.000	0.958	0.788	1.000
	R2 train	0.934	1.000	0.908	1.000	1.000	0.912	0.615	1.000
Trait 3	PCC test	0.757	0.811	0.621	0.811	0.811	0.810	0.555	0.830
	R2 test	0.549	0.655	0.250	0.655	0.655	0.645	0.269	0.689
	PCC train	0.963	1.000	0.978	1.000	1.000	0.899	0.754	1.000
	R2 train	0.907	1.000	0.906	1.000	1.000	0.792	0.534	1.000

80000

		RFR	Ridge	SVR	KR	Linear	Xgb	KNN	GPR
Trait 1	PCC test	0.885	0.958	0.954	0.958	0.958	0.939	0.415	0.962
	R2 test	0.728	0.919	0.857	0.919	0.919	0.868	0.159	0.926
	PCC train	0.964	1.000	0.995	1.000	1.000	0.968	0.764	1.000
	R2 train	0.913	1.000	0.985	1.000	1.000	0.935	0.500	1.000
Trait 2	PCC test	0.856	0.946	0.934	0.946	0.946	0.924	0.623	0.947
	R2 test	0.696	0.894	0.829	0.894	0.894	0.843	0.387	0.897
	PCC train	0.973	1.000	0.995	1.000	1.000	0.959	0.802	1.000
	R2 train	0.933	1.000	0.984	1.000	1.000	0.917	0.632	1.000
Trait 3	PCC test	0.723	0.822	0.824	0.822	0.822	0.811	0.494	0.832
	R2 test	0.506	0.675	0.650	0.675	0.674	0.648	0.146	0.692
	PCC train	0.958	1.000	0.986	1.000	1.000	0.902	0.729	1.000
	R2 train	0.894	1.000	0.962	1.000	1.000	0.799	0.474	1.000

120000

		RFR	Ridge	SVR	KR	Linear	Xgb	KNN	GPR
Trait 1	PCC test	0.875	0.961	0.959	0.961	0.961	0.936	0.353	0.963
	R2 test	0.721	0.922	0.909	0.922	0.922	0.868	0.098	0.926
	PCC train	0.968	1.000	0.994	1.000	1.000	0.968	0.771	1.000
	R2 train	0.921	1.000	0.987	1.000	1.000	0.934	0.495	1.000
Trait 2	PCC test	0.870	0.947	0.941	0.947	0.947	0.919	0.620	0.947
	R2 test	0.712	0.895	0.876	0.895	0.895	0.835	0.383	0.897
	PCC train	0.974	1.000	0.993	1.000	1.000	0.963	0.810	1.000

	R2 train	0.934	1.000	0.984	1.000	1.000	0.923	0.646	1.000
Trait 3	PCC test	0.727	0.824	0.828	0.824	0.824	0.803	0.490	0.832
	R2 test	0.513	0.678	0.679	0.678	0.678	0.636	0.130	0.692
	PCC train	0.960	1.000	0.975	1.000	1.000	0.907	0.733	1.000
	R2 train	0.900	1.000	0.942	1.000	1.000	0.808	0.481	1.000

160000

		RFR	**Ridge**	**SVR**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait 1	PCC test	0.875	0.961	0.959	0.961	0.961	0.936	0.353	0.963
	R2 test	0.721	0.922	0.909	0.922	0.922	0.868	0.098	0.926
	PCC train	0.968	1.000	0.994	1.000	1.000	0.968	0.771	1.000
	R2 train	0.921	1.000	0.987	1.000	1.000	0.934	0.495	1.000
Trait 2	PCC test	0.870	0.947	0.941	0.947	0.947	0.919	0.620	0.947
	R2 test	0.712	0.895	0.876	0.895	0.895	0.835	0.383	0.897
	PCC train	0.974	1.000	0.993	1.000	1.000	0.963	0.810	1.000
	R2 train	0.934	1.000	0.984	1.000	1.000	0.923	0.646	1.000
Trait 3	PCC test	0.727	0.824	0.828	0.824	0.824	0.803	0.490	0.832
	R2 test	0.513	0.678	0.679	0.678	0.678	0.636	0.130	0.692
	PCC train	0.960	1.000	0.975	1.000	1.000	0.907	0.733	1.000
	R2 train	0.900	1.000	0.942	1.000	1.000	0.808	0.481	1.000

200000

		RFR	**Ridge**	**SVR**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**

Trait 1	PCC test	0.874	0.961	0.961	0.961	0.961	0.943	0.398	0.963
	R2 test	0.729	0.924	0.919	0.924	0.924	0.881	0.142	0.927
	PCC train	0.961	1.000	0.991	1.000	1.000	0.970	0.770	1.000
	R2 train	0.907	1.000	0.981	1.000	1.000	0.939	0.503	1.000
Trait 2	PCC test	0.866	0.948	0.944	0.948	0.948	0.924	0.603	0.948
	R2 test	0.727	0.898	0.887	0.898	0.898	0.848	0.362	0.898
	PCC train	0.970	1.000	0.988	1.000	1.000	0.963	0.817	1.000
	R2 train	0.929	1.000	0.975	1.000	1.000	0.924	0.656	1.000
Trait 3	PCC test	0.713	0.823	0.827	0.823	0.823	0.799	0.480	0.831
	R2 test	0.499	0.677	0.683	0.677	0.677	0.632	0.134	0.691
	PCC train	0.962	1.000	0.948	1.000	1.000	0.905	0.725	1.000
	R2 train	0.903	1.000	0.889	1.000	1.000	0.806	0.472	1.000

（2） 在二维 one hot 向量表示+ random projection 数据上的结果，使用 80000 个 feature 进行 10 折交叉验证。

0 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.886	0.958	0.958	0.958	0.939	0.415	0.962
	R2 test	0.962	1.000	1.000	1.000	0.968	0.764	1.000
	PCC train	0.746	0.919	0.919	0.919	0.868	0.159	0.926
	R2 train	0.909	1.000	1.000	1.000	0.935	0.500	1.000
Trait2	PCC test	0.851	0.946	0.946	0.946	0.924	0.623	0.947
	R2 test	0.974	1.000	1.000	1.000	0.959	0.802	1.000

	PCC train	0.693	0.894	0.894	0.894	0.843	0.387	0.897
	R2 train	0.936	1.000	1.000	1.000	0.917	0.632	1.000
Trait3	PCC test	0.720	0.822	0.822	0.822	0.811	0.494	0.832
	R2 test	0.960	1.000	1.000	1.000	0.902	0.729	1.000
	PCC train	0.503	0.675	0.675	0.674	0.648	0.146	0.692
	R2 train	0.897	1.000	1.000	1.000	0.799	0.474	1.000

1 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.887	0.948	0.948	0.948	0.930	0.395	0.951
	R2 test	0.962	1.000	1.000	1.000	0.969	0.764	1.000
	PCC train	0.741	0.899	0.899	0.899	0.856	0.143	0.905
	R2 train	0.907	1.000	1.000	1.000	0.936	0.504	1.000
Trait2	PCC test	0.851	0.943	0.943	0.943	0.918	0.621	0.946
	R2 test	0.973	1.000	1.000	1.000	0.959	0.807	1.000
	PCC train	0.697	0.888	0.888	0.888	0.837	0.385	0.895
	R2 train	0.934	1.000	1.000	1.000	0.916	0.640	1.000
Trait3	PCC test	0.712	0.798	0.798	0.798	0.779	0.443	0.812
	R2 test	0.959	1.000	1.000	1.000	0.904	0.730	1.000
	PCC train	0.494	0.634	0.634	0.634	0.602	0.058	0.659
	R2 train	0.895	1.000	1.000	1.000	0.802	0.472	1.000

2 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**

Trait1	PCC test	0.889	0.953	0.953	0.953	0.933	0.353	0.957
	R2 test	0.964	1.000	1.000	1.000	0.967	0.761	1.000
	PCC train	0.767	0.908	0.908	0.908	0.868	0.108	0.915
	R2 train	0.913	1.000	1.000	1.000	0.933	0.500	1.000
Trait2	PCC test	0.837	0.939	0.939	0.939	0.912	0.610	0.942
	R2 test	0.972	1.000	1.000	1.000	0.960	0.807	1.000
	PCC train	0.681	0.880	0.880	0.880	0.827	0.365	0.886
	R2 train	0.930	1.000	1.000	1.000	0.918	0.640	1.000
Trait3	PCC test	0.727	0.817	0.817	0.817	0.788	0.480	0.825
	R2 test	0.959	1.000	1.000	1.000	0.905	0.724	1.000
	PCC train	0.515	0.659	0.659	0.659	0.618	0.109	0.676
	R2 train	0.896	1.000	1.000	1.000	0.803	0.458	1.000

3 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.883	0.952	0.952	0.952	0.929	0.373	0.954
	R2 test	0.963	1.000	1.000	1.000	0.968	0.764	1.000
	PCC train	0.758	0.904	0.904	0.904	0.862	0.104	0.909
	R2 train	0.911	1.000	1.000	1.000	0.934	0.501	1.000
Trait2	PCC test	0.849	0.947	0.947	0.947	0.924	0.649	0.950
	R2 test	0.972	1.000	1.000	1.000	0.960	0.804	1.000
	PCC train	0.692	0.896	0.896	0.896	0.843	0.421	0.902
	R2 train	0.930	1.000	1.000	1.000	0.918	0.633	1.000

Trait3	PCC test	0.710	0.798	0.798	0.798	0.780	0.508	0.811
	R2 test	0.959	1.000	1.000	1.000	0.904	0.722	1.000
	PCC train	0.481	0.636	0.636	0.636	0.599	0.134	0.658
	R2 train	0.898	1.000	1.000	1.000	0.803	0.460	1.000

4 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.876	0.947	0.947	0.947	0.927	0.411	0.950
	R2 test	0.967	1.000	1.000	1.000	0.967	0.765	1.000
	PCC train	0.728	0.897	0.897	0.897	0.851	0.156	0.902
	R2 train	0.920	1.000	1.000	1.000	0.933	0.502	1.000
Trait2	PCC test	0.861	0.939	0.939	0.939	0.920	0.633	0.945
	R2 test	0.973	1.000	1.000	1.000	0.961	0.811	1.000
	PCC train	0.717	0.882	0.882	0.882	0.841	0.399	0.893
	R2 train	0.933	1.000	1.000	1.000	0.919	0.643	1.000
Trait3	PCC test	0.705	0.804	0.804	0.804	0.778	0.424	0.811
	R2 test	0.959	1.000	1.000	1.000	0.904	0.728	1.000
	PCC train	0.487	0.644	0.644	0.644	0.602	-0.011	0.658
	R2 train	0.896	1.000	1.000	1.000	0.803	0.468	1.000

5 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.880	0.958	0.958	0.958	0.940	0.361	0.960

	R2 test	0.963	1.000	1.000	1.000	0.968	0.764	1.000
	PCC train	0.738	0.916	0.916	0.916	0.874	0.126	0.922
	R2 train	0.912	1.000	1.000	1.000	0.934	0.501	1.000
Trait2	PCC test	0.831	0.933	0.933	0.933	0.904	0.604	0.937
	R2 test	0.972	1.000	1.000	1.000	0.960	0.808	1.000
	PCC train	0.672	0.869	0.869	0.869	0.808	0.358	0.878
	R2 train	0.931	1.000	1.000	1.000	0.918	0.640	1.000
Trait3	PCC test	0.753	0.833	0.833	0.833	0.814	0.441	0.842
	R2 test	0.958	1.000	1.000	1.000	0.903	0.728	1.000
	PCC train	0.541	0.686	0.687	0.686	0.656	0.004	0.705
	R2 train	0.893	1.000	1.000	1.000	0.801	0.471	1.000

6 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.878	0.951	0.951	0.951	0.919	0.394	0.955
	R2 test	0.964	1.000	1.000	1.000	0.967	0.766	1.000
	PCC train	0.747	0.905	0.905	0.905	0.840	0.139	0.912
	R2 train	0.915	1.000	1.000	1.000	0.933	0.499	1.000
Trait2	PCC test	0.859	0.942	0.942	0.942	0.922	0.659	0.947
	R2 test	0.974	1.000	1.000	1.000	0.960	0.803	1.000
	PCC train	0.702	0.887	0.887	0.887	0.841	0.433	0.896
	R2 train	0.935	1.000	1.000	1.000	0.917	0.632	1.000
Trait3	PCC test	0.705	0.817	0.817	0.817	0.788	0.519	0.824

	R2 test	0.958	1.000	1.000	1.000	0.903	0.725	1.000
	PCC train	0.485	0.663	0.663	0.663	0.615	0.131	0.678
	R2 train	0.892	1.000	1.000	1.000	0.800	0.463	1.000

7 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.888	0.955	0.955	0.955	0.936	0.351	0.958
	R2 test	0.966	1.000	1.000	1.000	0.967	0.757	1.000
	PCC train	0.749	0.912	0.912	0.912	0.865	0.113	0.917
	R2 train	0.916	1.000	1.000	1.000	0.932	0.500	1.000
Trait2	PCC test	0.823	0.942	0.942	0.942	0.916	0.605	0.943
	R2 test	0.973	1.000	1.000	1.000	0.961	0.807	1.000
	PCC train	0.665	0.886	0.886	0.886	0.835	0.360	0.890
	R2 train	0.933	1.000	1.000	1.000	0.920	0.639	1.000
Trait3	PCC test	0.727	0.836	0.836	0.836	0.809	0.483	0.842
	R2 test	0.959	1.000	1.000	1.000	0.903	0.726	1.000
	PCC train	0.497	0.698	0.698	0.698	0.641	0.128	0.708
	R2 train	0.895	1.000	1.000	1.000	0.801	0.463	1.000

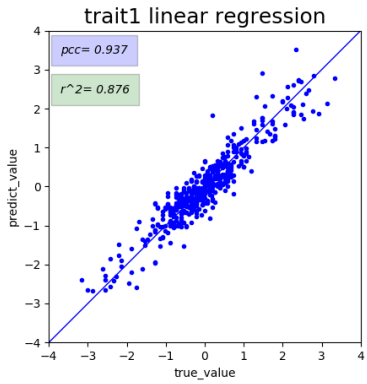
8 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.870	0.952	0.952	0.952	0.934	0.459	0.954
	R2 test	0.964	1.000	1.000	1.000	0.968	0.761	1.000
	PCC train	0.734	0.904	0.904	0.904	0.871	0.211	0.908

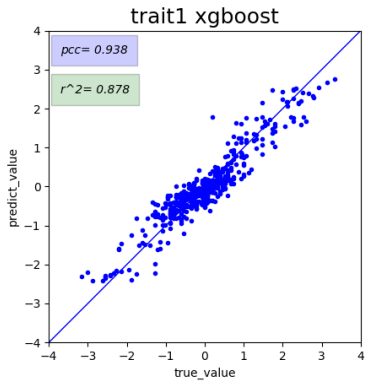
	R2 train	0.912	1.000	1.000	1.000	0.935	0.497	1.000
Trait2	PCC test	0.835	0.934	0.934	0.934	0.905	0.674	0.936
	R2 test	0.973	1.000	1.000	1.000	0.960	0.806	1.000
	PCC train	0.669	0.870	0.870	0.870	0.811	0.444	0.875
	R2 train	0.934	1.000	1.000	1.000	0.918	0.635	1.000
Trait3	PCC test	0.761	0.835	0.835	0.835	0.811	0.459	0.842
	R2 test	0.958	1.000	1.000	1.000	0.903	0.727	1.000
	PCC train	0.551	0.695	0.695	0.695	0.645	0.043	0.709
	R2 train	0.893	1.000	1.000	1.000	0.799	0.468	1.000

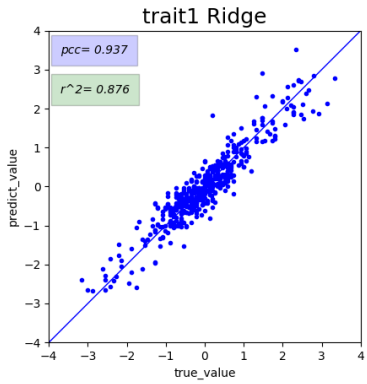
9 fold		**RFR**	**Ridge**	**KR**	**Linear**	**Xgb**	**KNN**	**GPR**
Trait1	PCC test	0.880	0.952	0.952	0.952	0.932	0.386	0.956
	R2 test	0.963	1.000	1.000	1.000	0.968	0.759	1.000
	PCC train	0.745	0.906	0.906	0.906	0.863	0.127	0.914
	R2 train	0.909	1.000	1.000	1.000	0.934	0.497	1.000
Trait2	PCC test	0.820	0.934	0.934	0.934	0.903	0.662	0.938
	R2 test	0.973	1.000	1.000	1.000	0.959	0.805	1.000
	PCC train	0.647	0.871	0.871	0.871	0.809	0.435	0.879
	R2 train	0.934	1.000	1.000	1.000	0.916	0.635	1.000
Trait3	PCC test	0.748	0.840	0.840	0.840	0.820	0.455	0.846
	R2 test	0.958	1.000	1.000	1.000	0.903	0.732	1.000
	PCC train	0.532	0.706	0.706	0.706	0.654	0.110	0.714

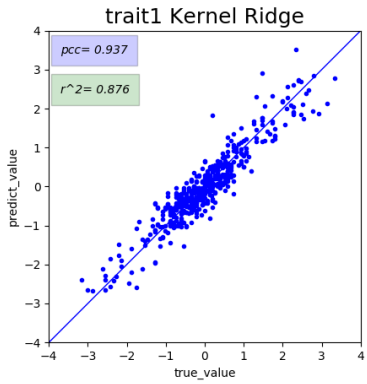
	R2 train	0.896	1.000	1.000	1.000	0.800	0.480	1.000

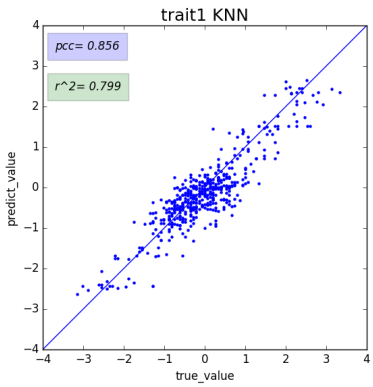
预测值与真值散点图

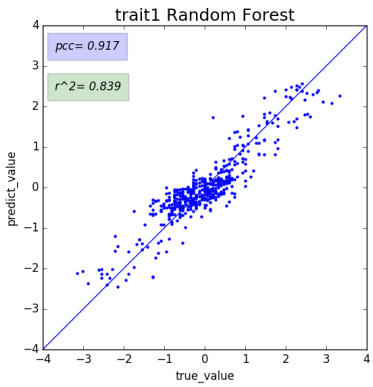


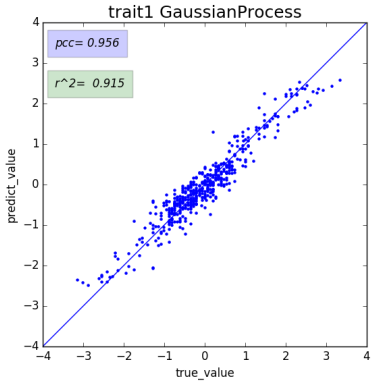


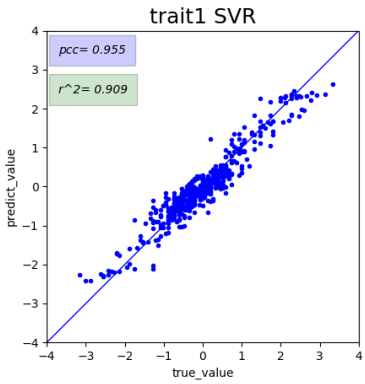


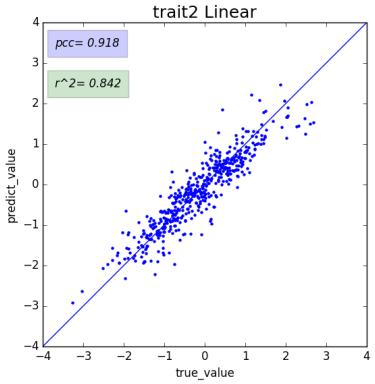


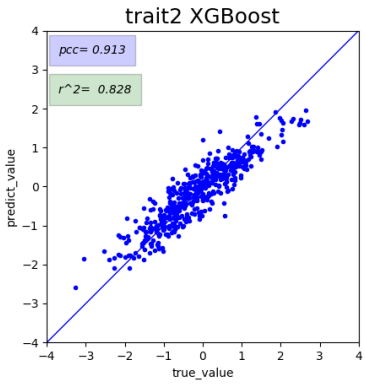


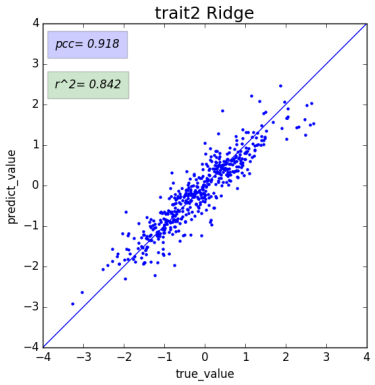


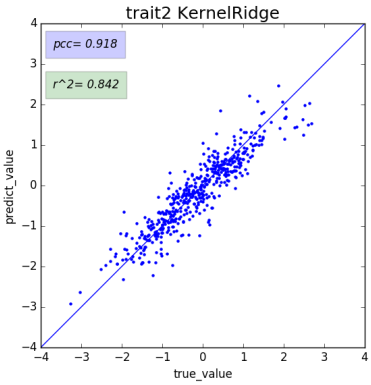

trait1 GaussianProcess
pcc= 0.956
r^2= 0.915
predict_value
true_value

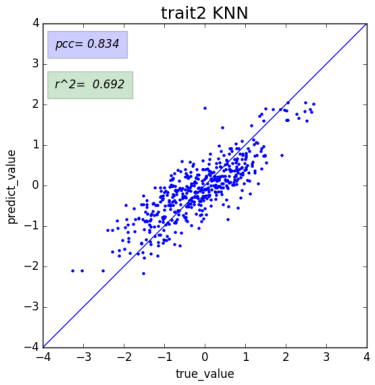

trait1 SVR
pcc= 0.955
r^2= 0.909
predict_value
true_value

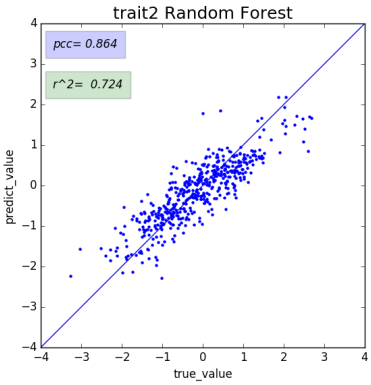

trait2 Linear
pcc= 0.918
r^2= 0.842
predict_value
true_value

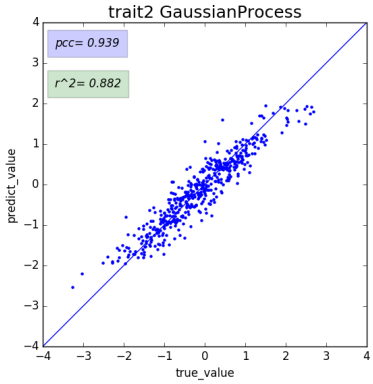

trait2 XGBoost
pcc= 0.913
r^2= 0.828
predict_value
true_value

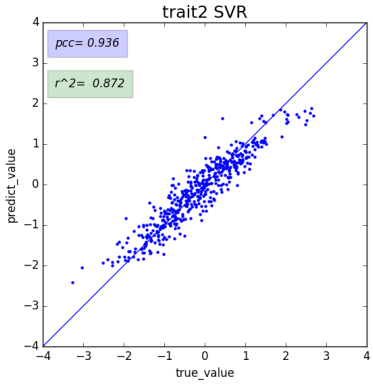

trait2 Ridge
pcc= 0.918
r^2= 0.842
predict_value
true_value

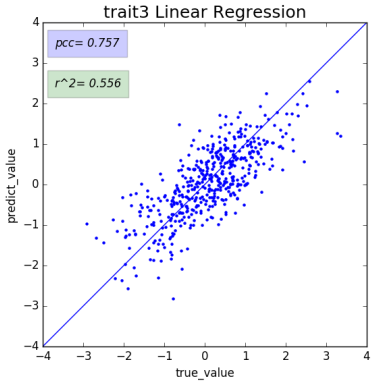

trait2 KernelRidge
pcc= 0.918
r^2= 0.842
predict_value
true_value

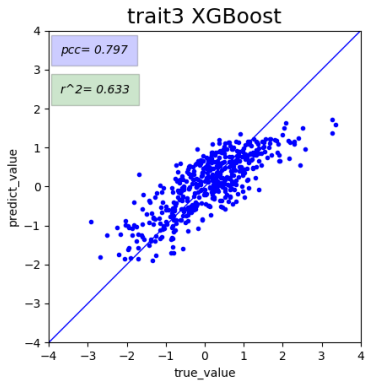

trait2 KNN
pcc= 0.834
r^2= 0.692
predict_value
true_value

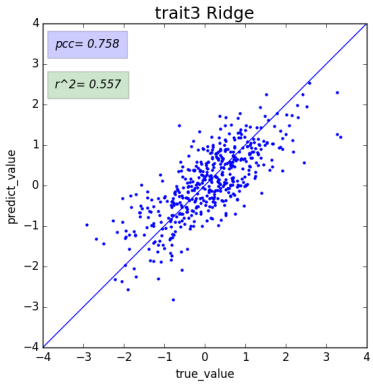

trait2 Random Forest
pcc= 0.864
r^2= 0.724
predict_value
true_value

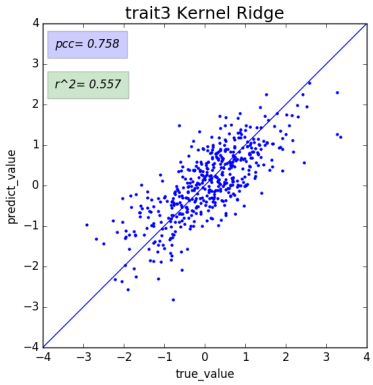

trait2 GaussianProcess
pcc= 0.939
r^2= 0.882
predict_value
true_value

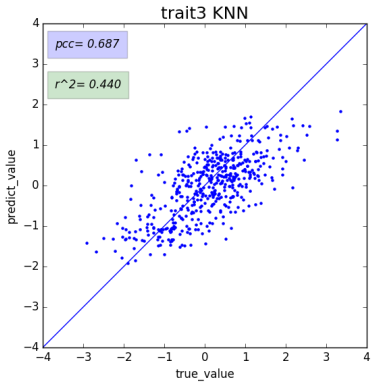

trait2 SVR
pcc= 0.936
r^2= 0.872
predict_value
true_value

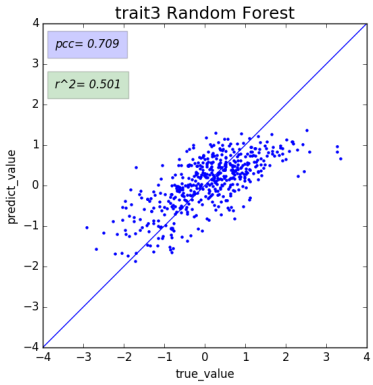

trait3 Linear Regression
pcc= 0.757
r^2= 0.556
predict_value
true_value

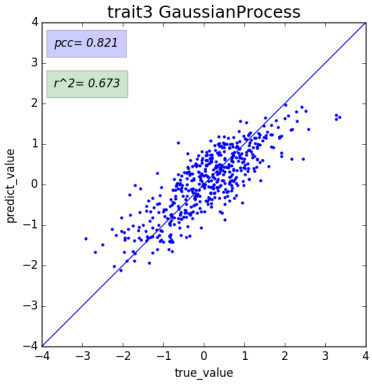

trait3 XGBoost
pcc= 0.797
r^2= 0.633
predict_value
true_value

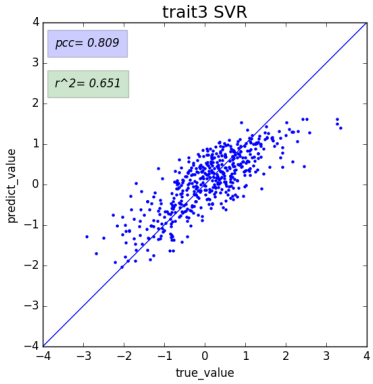

trait3 Ridge
pcc= 0.758
r^2= 0.557
predict_value
true_value


trait3 Kernel Ridge
pcc= 0.758
r^2= 0.557
predict_value
true_value


trait3 KNN
pcc= 0.687
r^2= 0.440
predict_value
true_value


trait3 Random Forest
pcc= 0.709
r^2= 0.501
predict_value
true_value


trait3 GaussianProcess
pcc= 0.821
r^2= 0.673
predict_value
true_value


trait3 SVR
pcc= 0.809
r^2= 0.651
predict_value
true_value