# 开放式机器学习课程。专题10。梯度增强



大家好！

到目前为止，我们已经讨论了9个主题，从Python中的探索性数据分析到时间序列分析。今天我们来看看最流行和实用的机器学习算法之一：梯度增强。你可以在文章的格式中找到更详细的数学。

### 文章概要

1. boosting简介及历史
2. 梯度提升机算法
3. 损失函数
4. 作业#10
5. 有用资源

### 一。boosting简介及历史

几乎所有机器学习的人都听说过梯度增强。许多数据科学家将此算法包含在他们的数据科学家工具箱中，因为它在任何给定（未知）问题上都能产生良好的结果。

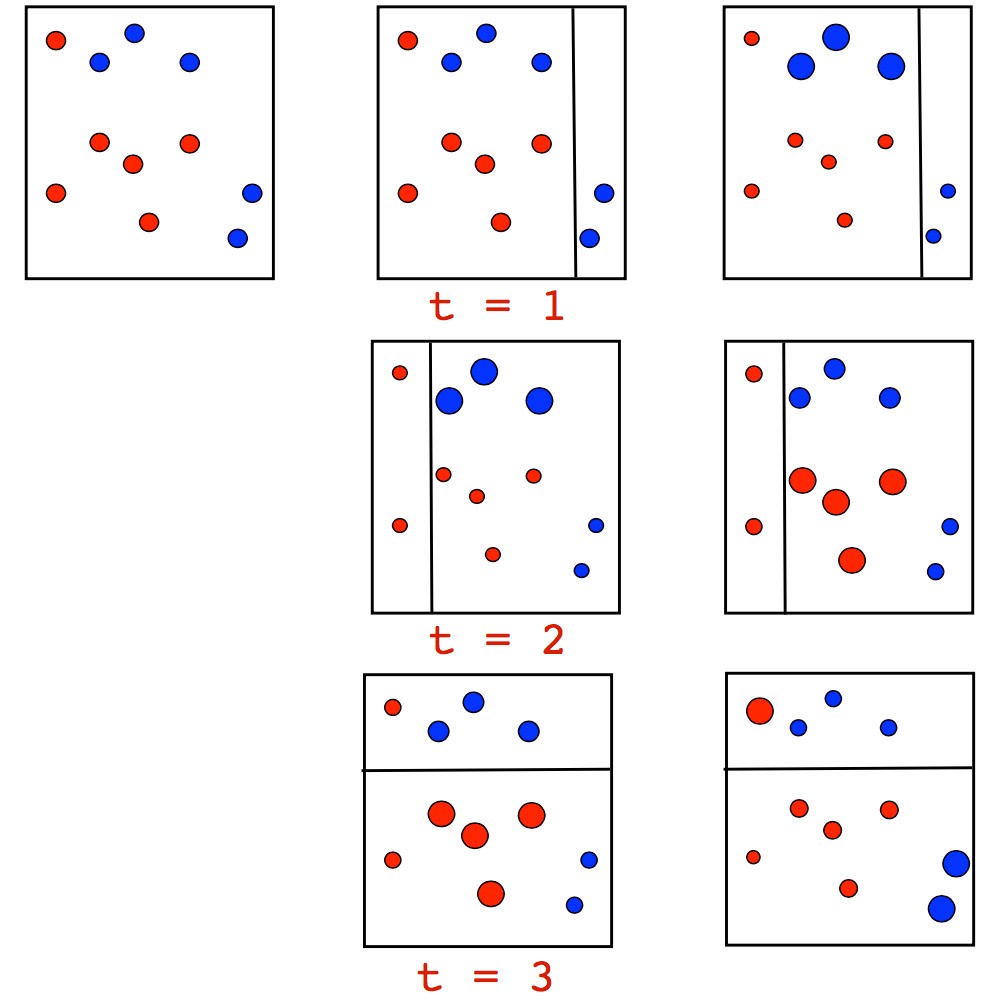
此外，XGBoost通常是的标准配方。它是如此的流行，以至于堆叠xgboots的想法已经成为一种模因。此外，助推是中的一个重要组成部分；有时，它甚至被认为是。让我们看看boosting的历史和发展。

Boosting是从大量相对薄弱和简单的模型中得到一个强大的模型的可能吗？所谓弱模型，不是指决策树等简单的基本模型，而是指精度性能差的模型，其中差的比随机的要好一点。

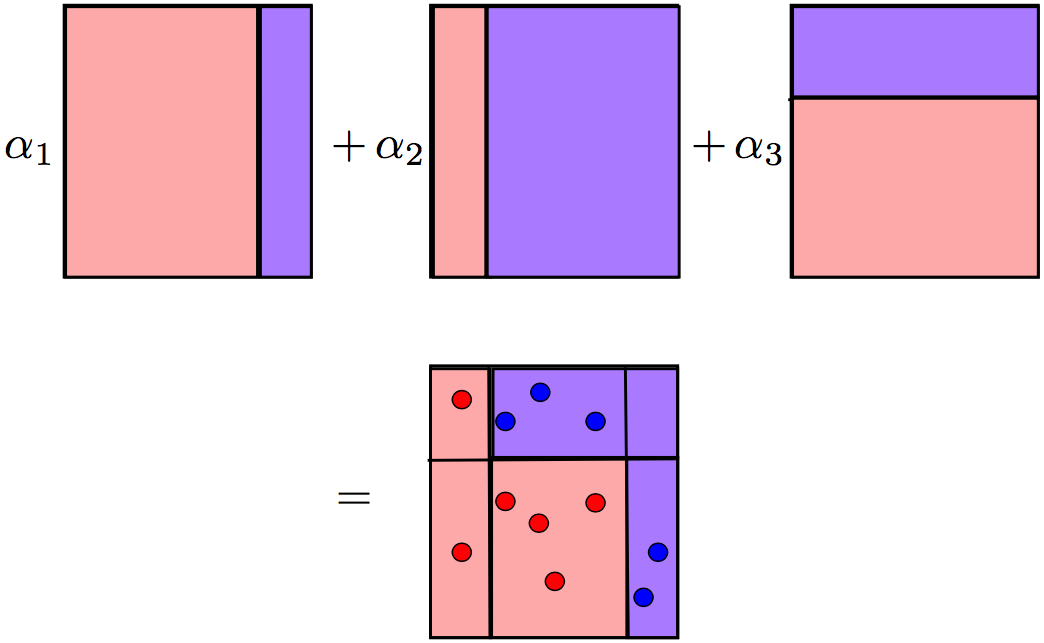
对于这个问题已经确定了，但是基于这个解决方案开发出功能完善的算法（例如AdaBoost）花了几年时间。这些算法采用贪婪的方法：首先，通过重新加权输入数据，建立简单模型（基本算法）的线性组合。然后，该模型（通常是一个决策树）建立在先前错误预测的对象上，这些对象现在被赋予更大的权重。

许多机器学习课程学习的是GBM（Gradient Boosting machine）的鼻祖AdaBoost。然而，自从AdaBoost与GBM合并以来，AdaBoost显然只是GBM的一个特殊变体。

算法本身具有非常清晰的视觉解释和定义权重的直观性。让我们看看下面的玩具分类问题，我们将在AdaBoost的每次迭代中在深度为1的树（也称为“树桩”）之间分割数据。对于前两个迭代，我们有以下图片：



点的大小与它的权重相对应，该权重是为错误的预测指定的。在每次迭代中，我们可以看到这些权重都在增长，树桩无法处理这个问题。不过，如果我们对这些树桩进行加权投票，我们将得到正确的分类：



这里有一个更详细的AdaBoost示例，当我们迭代时，我们可以看到权重增加，特别是在类之间的边界上。

AdaBoost工作得很好，但是对于算法为什么成功的解释却埋下了怀疑的种子。一些人认为这是一个超级算法，一个银弹，但其他人持怀疑态度，认为AdaBoost只是过于合适。

过度拟合问题确实存在，特别是当数据具有较强的异常值时。因此，在这些类型的问题中，AdaBoost是不稳定的。幸运的是，斯坦福大学统计系的几位教授开始研究这个算法，他们创建了套索、弹性网和随机森林。1999年，Jerome Friedman提出了boosting算法发展的一般化 - 梯度boosting（Machine），也称为GBM。通过这项工作，弗里德曼建立了许多算法的统计基础，提供了在函数空间中优化优化的一般方法。

CART、bootstrap和许多其他算法都起源于斯坦福的统计部门。在这样做的过程中，该系在今后的教科书中巩固了他们的名字。这些算法具有很强的实用性，最近的一些工作还没有被广泛采用。例如，退房。

弗里德曼的录像资料不多。尽管如此，40多年前，他对CART的创建以及如何解决统计问题（类似于今天的数据分析和数据科学）非常感兴趣。还有一个伟大的，回顾性的数据分析从一个创造者的方法，我们每天使用。

一般来说，已经从工程和算法研究过渡到建立和研究算法的成熟方法。从数学的角度来看，这并不是一个大的变化，我们仍然在添加（或增强）弱算法，并通过对模型不准确的部分数据的逐步改进来扩大集成。但是，这一次，下一个简单模型不只是建立在重加权对象上，而是改进了它对总体目标函数梯度的逼近。这个概念极大地打开了我们的算法的想象和扩展。

### GBM历史

GBM问世10多年后，才成为数据科学工具箱的重要组成部分。扩展了GBM以应用于不同的统计问题：GLMPUBD和GAMBUP，用于加强已有的GAM模型，COXBOOST用于生存曲线，RankBoost和LAMBARMAT用于排名。

GBM的许多实现也出现在不同的名称和平台上：随机GBM、GBDT（梯度增强决策树）、GBRT（梯度增强回归树）、MART（多加性回归树）等等。此外，ML社区是非常分割和分离的，这使得很难追踪到boosting的广泛程度。

同时，boosting在搜索排名中也得到了积极的应用。这个问题被重写为一个损失函数，它惩罚输出顺序中的错误，因此简单地将其插入GBM中变得方便。AltaVista是最早引入排名提升的公司之一。很快，这种想法就传播到了雅虎、Yandex、Bing等公司。一旦出现这种情况，boosting就成了主要的算法之一，不仅用于研究，也用于工业的核心技术。

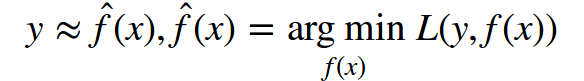


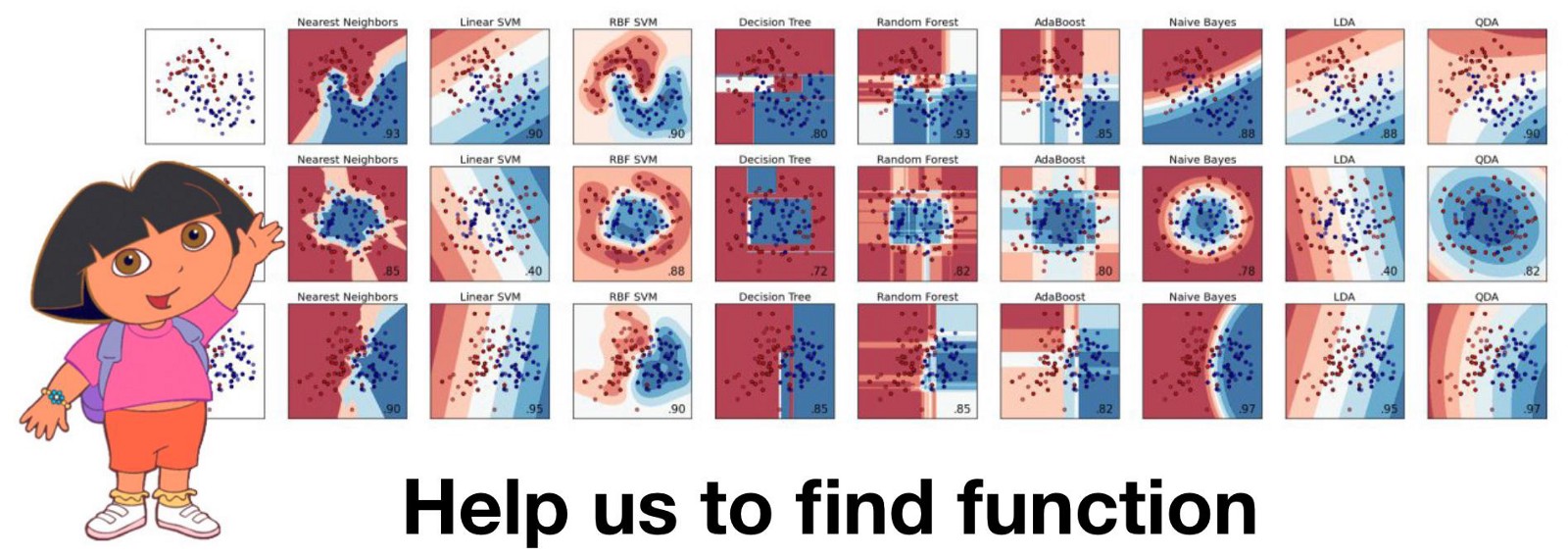
ML竞赛，特别是Kaggle竞赛，在促进普及方面发挥了重要作用。现在，研究人员有了一个共同的平台，他们可以在不同的数据科学问题上与来自世界各地的大量参与者竞争。有了Kaggle，人们可以在真实数据上测试新算法，给算法提供“闪光”的机会，并在跨竞争数据集共享模型性能结果方面提供完整的信息。这正是在使用boosting时发生的事情（查看从2011年开始对主要使用boosting的Kaggle获奖者的采访）。这座图书馆一出现就很快受到欢迎。XGBoost不是一个新的、独特的算法；它只是经典GBM的一个非常有效的实现，并带有额外的启发式。

该算法在ML算法出现多年后，经历了一条非常典型的发展道路：数学问题和算法工艺在实际应用中的成功和大规模采用。

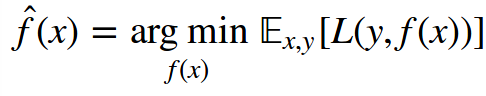
### 2。梯度提升机算法

我们将解决一般监督学习设置中的函数逼近问题。我们有一组特征X和目标变量y，用来恢复依赖性y=f（X）。我们通过近似f（x）来恢复依赖性，并且通过使用损失函数L（y，f）来理解哪个近似更好，我们想最小化：



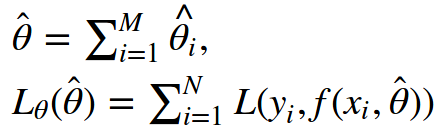


在这一时刻，我们不做任何假设有关的类型F（x），我们的近似模型，或目标变量的分布。我们只期望函数L（y，f）是可微的。我们的公式非常通用；让我们为具有总体平均值的特定数据集定义它。我们用于最小化数据丢失的表达式如下：

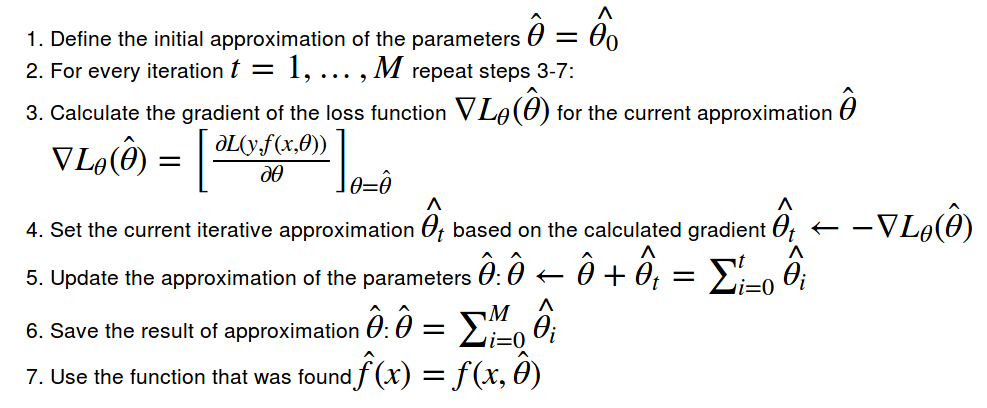


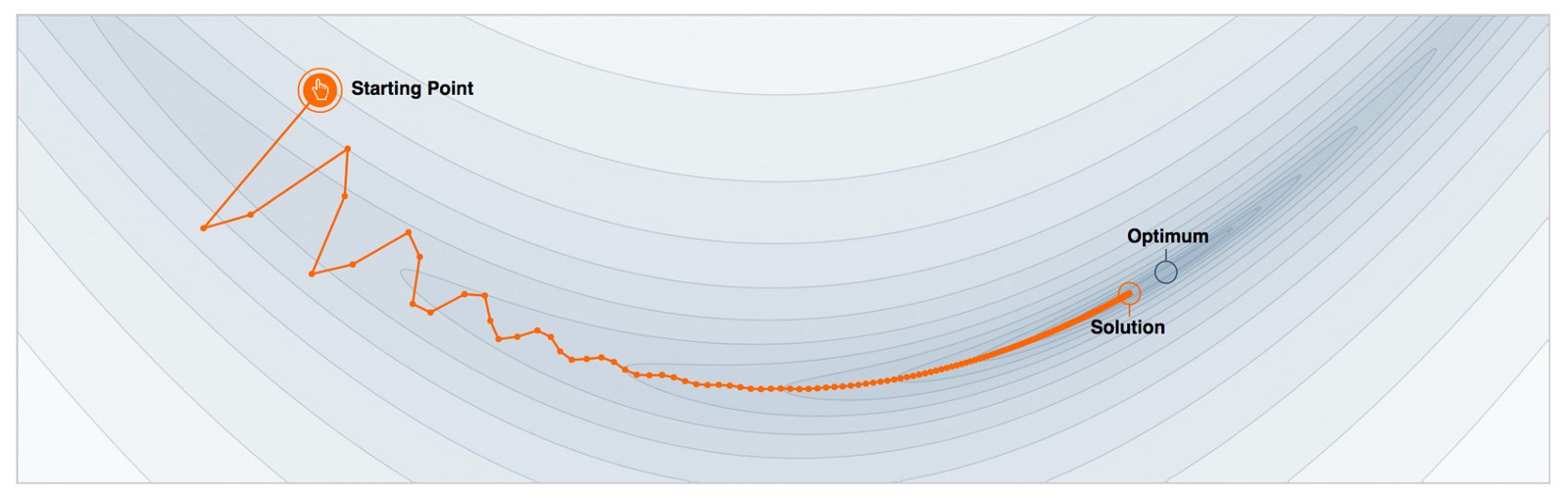
不幸的是，这些函数的数目不仅很大，其函数空间是无限维的。这就是为什么我们可以接受用一些函数族来限制搜索空间。这大大简化了目标，因为现在我们有一个参数值的可解优化。

寻找最优参数的简单解析解通常不存在，因此参数通常迭代地近似。首先，我们写下经验损失函数，这将允许我们使用我们的数据评估我们的参数。另外，让我们写出一些m次迭代的近似值和：



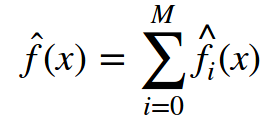
然后，唯一剩下的就是找到一个合适的迭代算法来最小化最后一个表达式。梯度下降是最简单和最常用的选择。我们定义梯度，并将迭代计算添加到梯度中（因为我们将损失最小化，所以添加了减号）。我们的最后一步是初始化我们的第一个近似并选择迭代次数M。让我们回顾一下这个低效和幼稚的算法的步骤：





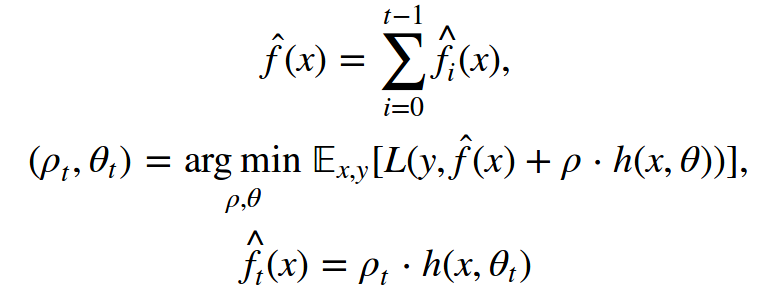
### 函数梯度下降

让我们想象一下，我们可以在函数空间中进行优化，并迭代地搜索函数本身。我们将我们的近似表示为增量改进的总和，每个都是一个函数。



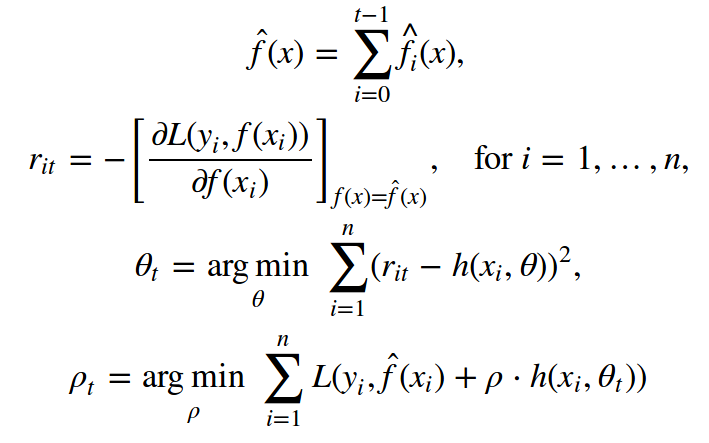
还没有发生什么事情，我们只是决定，我们将寻找我们的近似不是一个大的模型，具有大量的参数（作为一个例子，神经网络），但作为函数的总和，假装我们在功能空间中移动。

为了完成这项任务，我们需要用一些函数族来限制搜索。这里有几个问题 - 首先，模型的总和可能比这个家族中的任何模型都要复杂；其次，总的目标仍然在函数空间。让我们注意到，在每一步，我们都需要选择一个最佳系数。对于步骤t，问题如下：



这就是魔法发生的地方。我们已经用一般的术语定义了我们的所有目标，就好像我们可以为任何类型的损失函数训练任何类型的模型一样。在实践中，这是极其困难的，但幸运的是，有一个简单的方法来解决这项任务。

通过了解损失函数梯度的表达式，我们可以根据数据计算出它的值。所以，让我们训练模型，这样我们的预测将与这个梯度更相关（带负号）。换句话说，我们将使用最小二乘法用这些残差修正预测。对于分类、回归和排序任务，我们将最小化伪残差r和我们的预测之间的平方差。对于步骤t，最后一个问题如下所示：



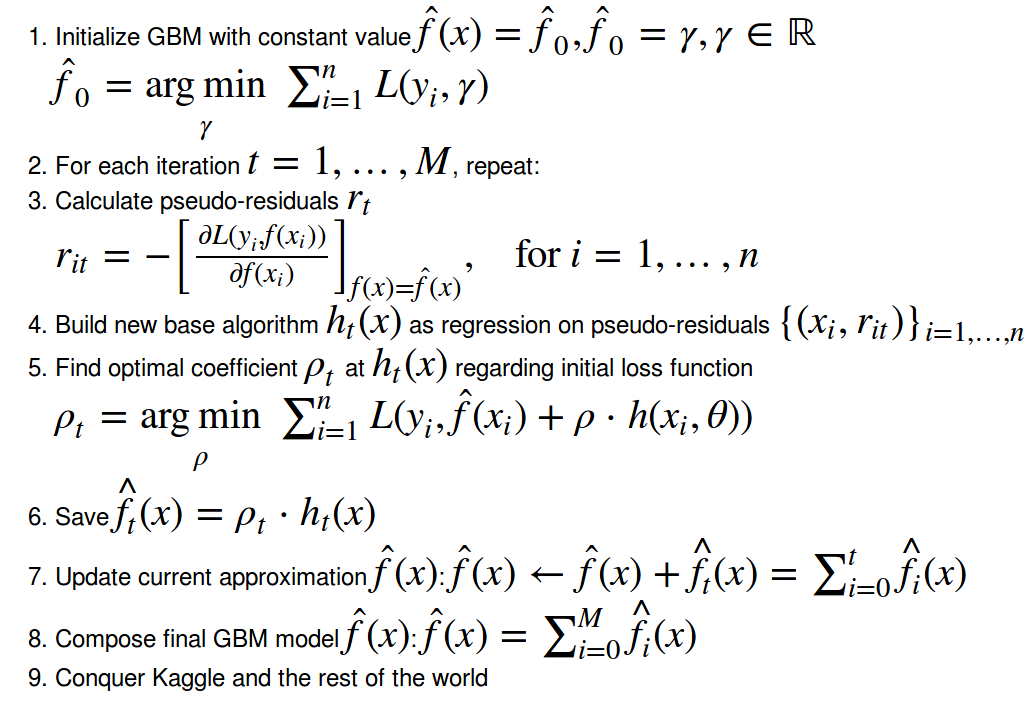


### Friedman经典GBM算法

我们现在可以定义Jerome Friedman在1999年提出的经典GBM算法。它是一个有监督的算法，包含以下组件：

* 数据集；
* 迭代次数M；
* 定义梯度损失函数的选择；
* 训练过程中基函数族的选择；
* 额外的超参数（例如，在决策树中，树的深度）；

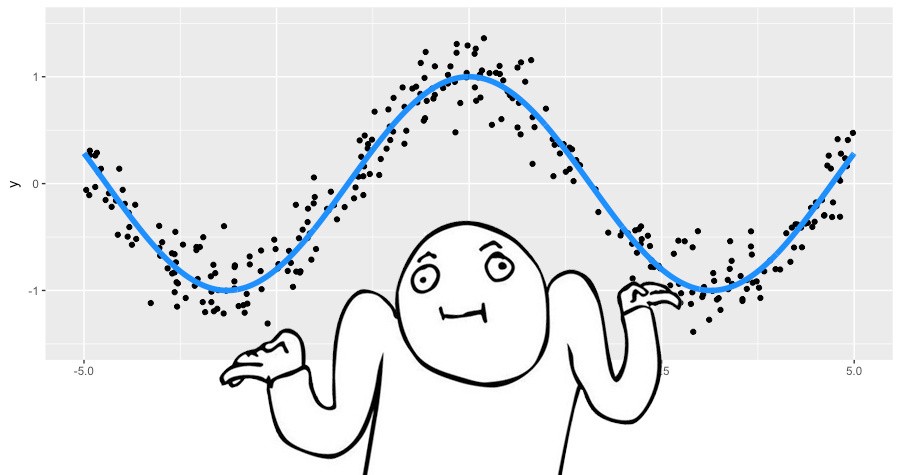
唯一剩下的就是初始近似值。为了简单起见，对于初始近似，使用恒定值伽玛。在初始损失函数（不是梯度）上，通过二进制搜索或另一行搜索算法来识别常数和最佳系数。因此，我们的GBM算法描述如下：



### 分步示例：GBM的工作原理

让我们看看GBM是如何工作的。在这个玩具例子中，我们将恢复一个噪声函数

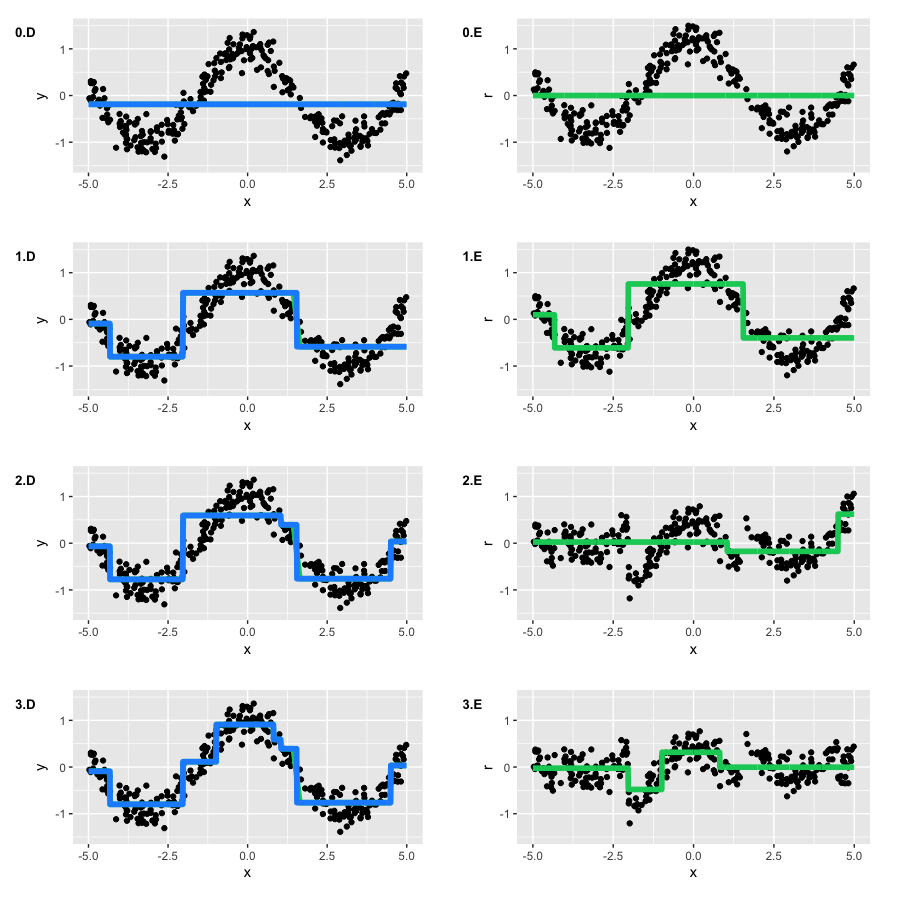




这是一个实值目标的回归问题，因此我们将选择使用均方误差损失函数。我们将产生300对观测，并用深度2的决策树来近似它们。让我们把使用GBM所需的所有东西放在一起：

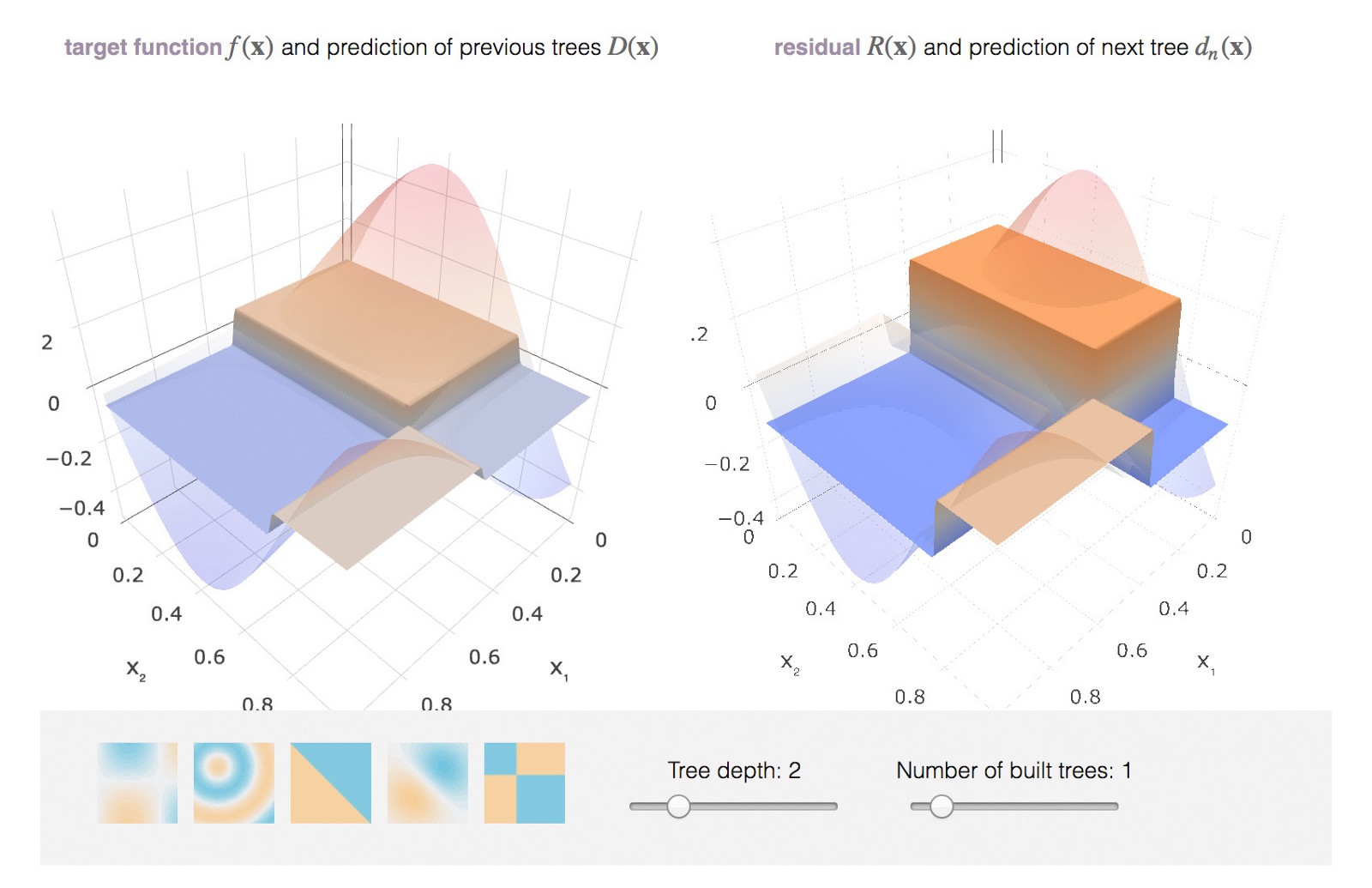
* 玩具数据
* 迭代次数M=3
* 均方误差损失函数L（y，f）=（y-f）2
* L（y，f）的梯度就是残差r=（y-f）
* 决策树作为基本算法
* 决策树的超参数：树深度等于2

我们将运行GBM，并绘制两种类型的图形：当前近似（蓝色图）和每个树上建立的伪残差（绿色图）。图的编号对应于迭代编号：



通过第二次迭代，我们的树恢复了函数的基本形式。然而，在第一次迭代中，我们看到算法只构建了函数的“左分支”。这是因为我们的树没有足够的深度来同时建立一个对称的分支，并且它集中在左分支上，误差较大。因此，右分支只在第二次迭代后出现。

该过程的其余部分如预期的那样——在每个步骤上，我们的伪残差减少，并且GBM在每次迭代中更好地逼近原始函数。然而，通过构造，树不能逼近连续函数，这意味着在这个例子中GBM不是理想的。要玩GBM函数近似，你可以使用这个博客中的令人敬畏的交互式演示：



### 三。损失函数

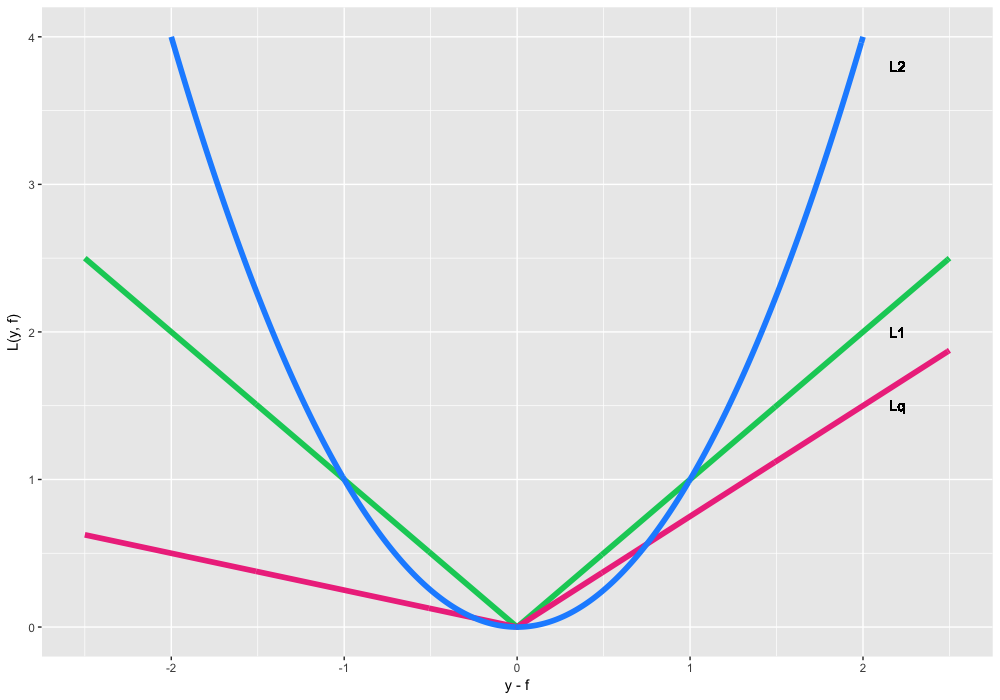
如果我们想解决分类问题而不是回归问题，会有什么变化？我们只需要选择一个合适的损失函数L（y，f）。这是最重要的高层时刻，它决定了我们将如何进行优化，以及我们可以在最终模型中预期的特性。

一般来说，我们不需要自己发明这个——研究人员已经为我们做过了。今天，我们将探讨损失函数的两个最常见的目标：回归和二元分类。

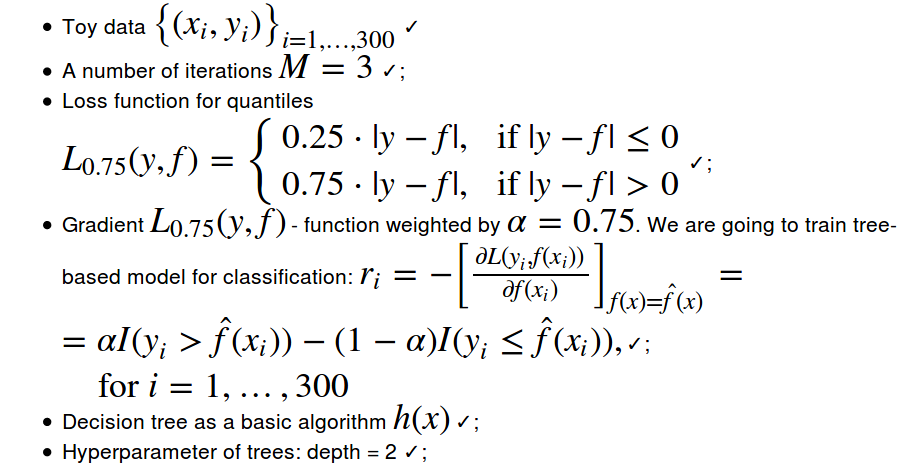
### 回归损失函数

让我们从y的回归问题开始，一个实数。为了选择合适的损失函数，我们需要考虑要恢复的条件分布（y | x）的哪些性质。最常见的选项是：

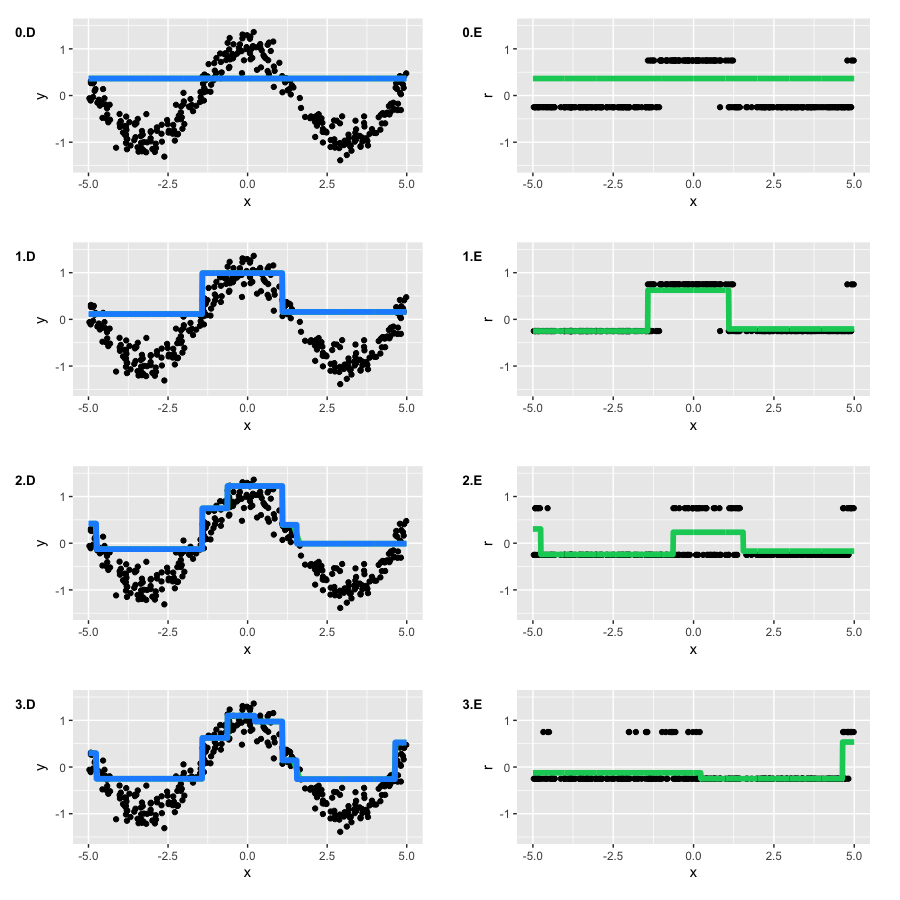
* L（y，f）=（y - f）a.k.a.L2损耗或高斯损耗。它是经典的条件平均，这是最简单和最常见的情况。如果我们对模型的鲁棒性没有任何额外的信息或要求，我们可以使用高斯损失。
* L（y，f）=| y - f | a.k.a.L1损失或拉普拉斯损失。乍一看，这个函数似乎不可微，但它实际上定义了条件中值。如我们所知，中值对异常值是稳健的，这就是为什么这种损失函数在某些情况下更好。对大变化的惩罚并不像在二语中那么重。
* Lq损失或分位数损失。它不使用中值，而是使用分位数。我们可以看到，这个函数是非对称的，它惩罚了位于定义分位数右侧的观测值。



我们用损失函数L\_q来计算数据。目标是恢复余弦的条件75%分位数。让我们把GBM的一切都放在一起：



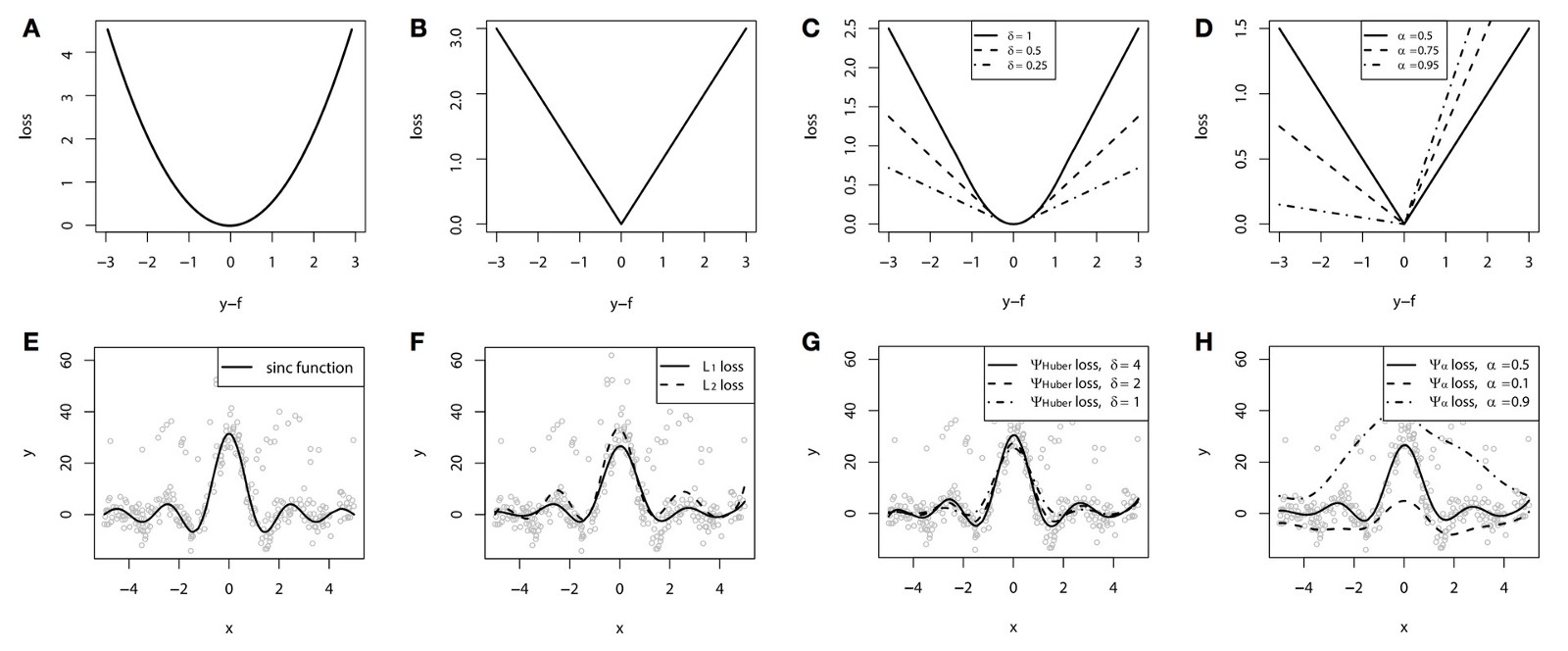
对于我们的初始近似，我们将采取Y的必要分位数。然而，我们对最优系数一无所知，所以我们将使用标准线搜索。结果如下：



具有分位数损失函数的GBM的总体结果与二次损失函数偏移量约为0.135的结果相同。但是如果我们使用90%的分位数，我们将没有足够的数据，因为类将变得不平衡。在处理非标准问题时，我们需要记住这一点。

对于回归任务，已经开发了许多损失函数，其中一些具有额外的属性。例如，它们可以像中的一样健壮。对于少量的异常值，损失函数作为L2工作，但是在定义了阈值之后，函数变为L1。这样可以减少异常值的影响，并集中于整体情况。

我们可以用下面的例子来说明这一点。数据是由函数y=sin（x）/x产生的，该函数带有来自正态分布和伯努利分布的混合噪声。我们给出了图A-D上的函数和F-H上的相关GBM（图E表示初始函数）：



.

在本例中，我们使用样条作为基本算法。你看，不一定非要用树来增强吗？

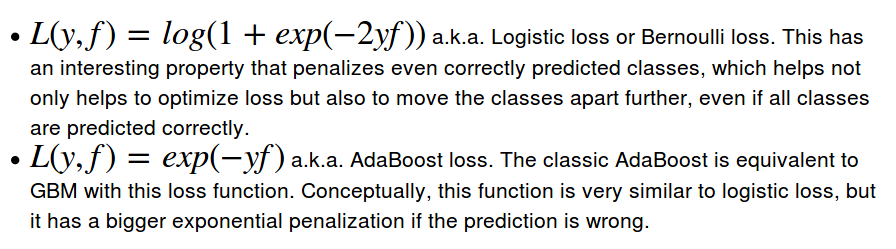
我们可以清楚地看到函数L2，L1和Huber损失之间的区别。如果我们选择最佳参数的胡贝尔损失，我们可以得到最佳的近似在我们所有的选项。这种差异在10%，50%和90%的分位数中也可以看到。

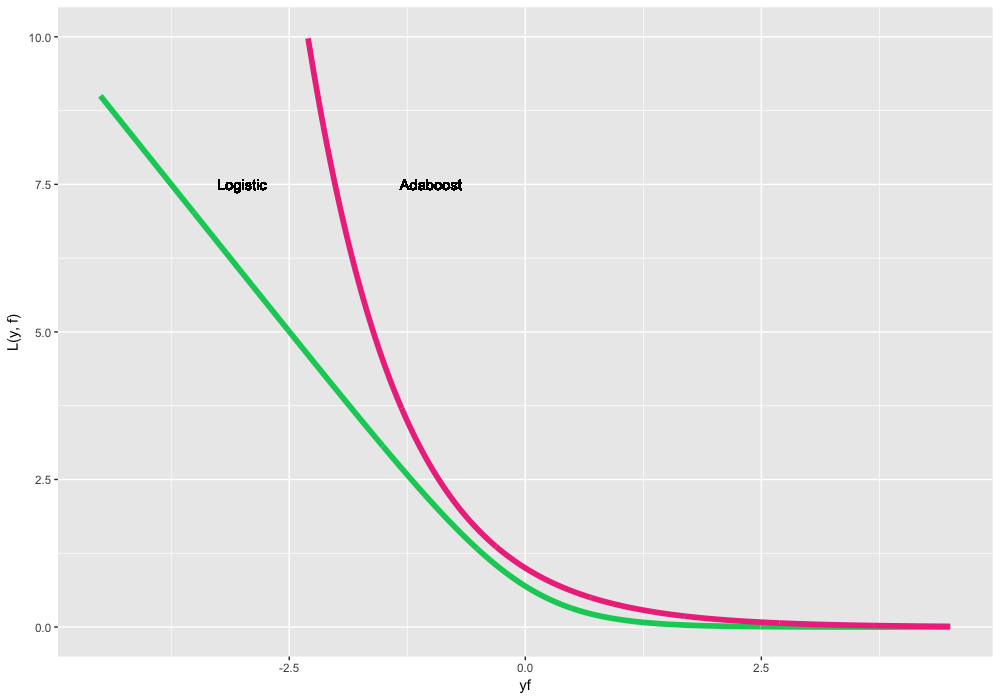
不幸的是，Huber loss函数仅由极少数流行的库/包支持；h2o支持它，但XGBoost不支持。它与其他更具异国情调的事物有关，但它可能仍然是有趣的知识。

### 分类损失函数

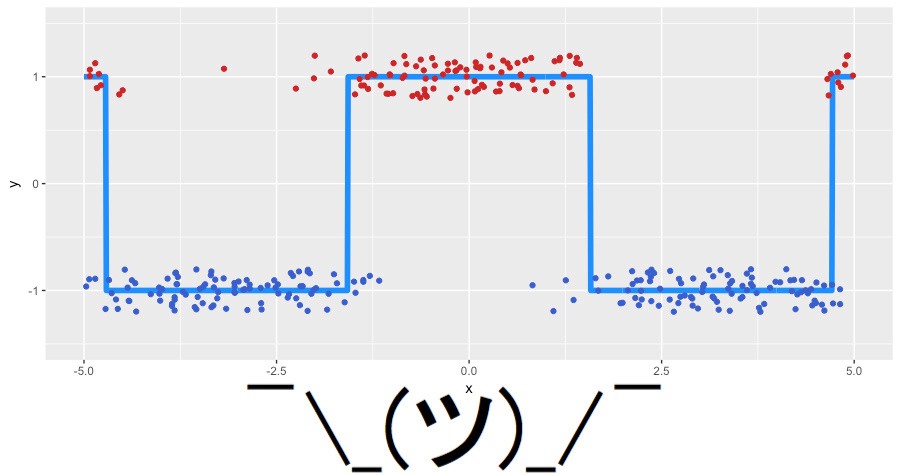
现在，让我们看看二进制分类问题。我们发现GBM甚至可以优化不可微损失函数。从技术上讲，用回归L2损失来解决这个问题是可能的，但这是不正确的。

目标变量的分布要求我们使用对数似然性，所以我们需要用不同的目标损失函数乘以它们的预测。最常见的选择如下：

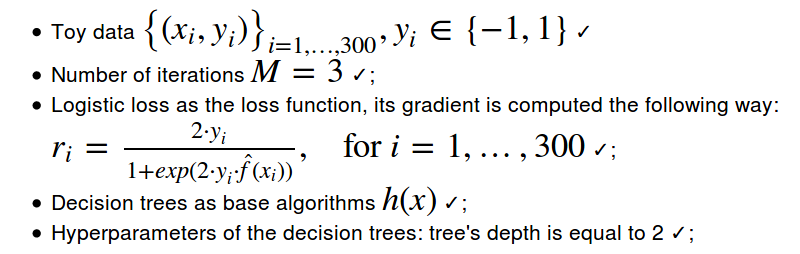




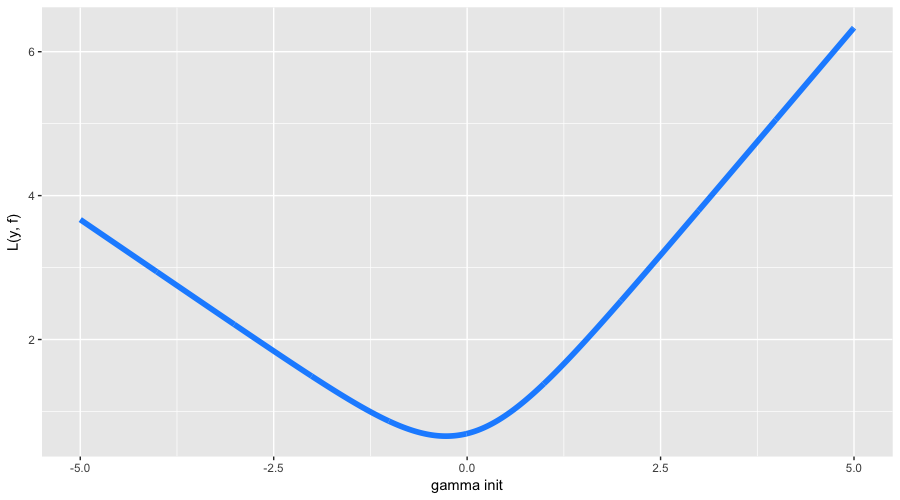
让我们为分类问题生成一些新的玩具数据。作为基础，我们将使用我们的噪声余弦，并将符号函数用于目标变量的类。我们的玩具数据如下（为了清晰起见，添加了抖动噪声）：



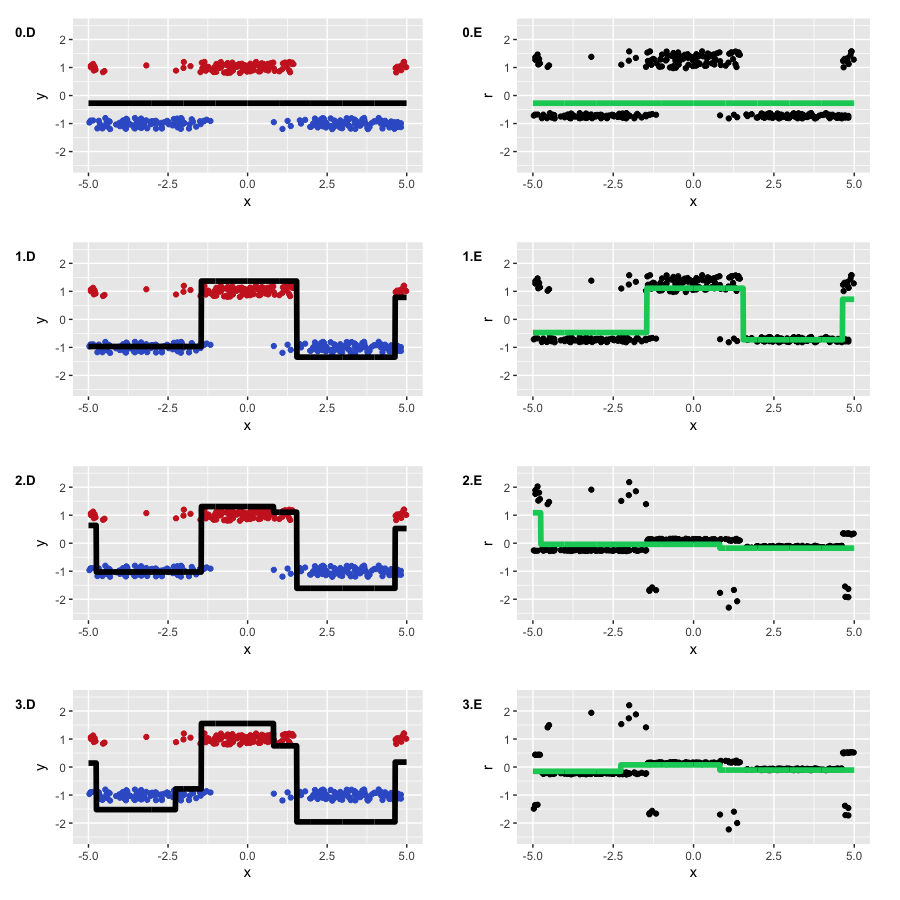
我们将利用物流损失来寻找我们真正的增长点。因此，我们再次总结了我们将用于GBM的内容：



这一次，算法的初始化有点困难。首先，我们的班级是不平衡的（63%对37%）。其次，我们的损失函数的初始化没有已知的分析公式，因此我们必须通过搜索来查找它：



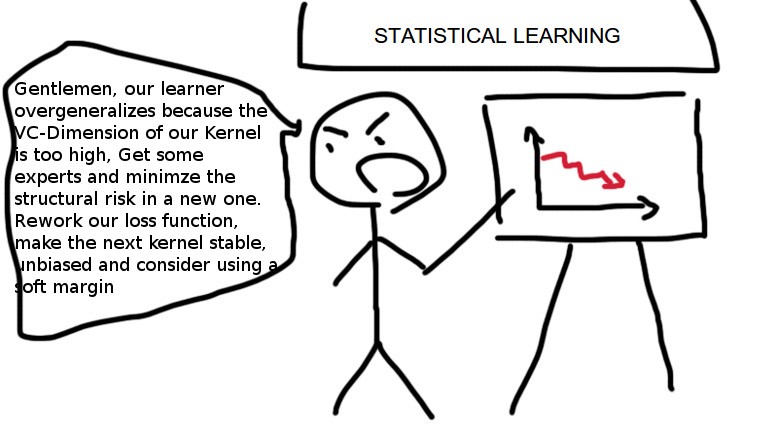
我们的最佳初始近似在-0.27 3之间。你可能会猜到它是负的，因为预测所有最受欢迎的类是更有利的，但是没有确切的值公式。现在让我们开始GBM，看看引擎盖下面到底发生了什么：



算法成功地恢复了类之间的分离。您可以看到“较低”区域是如何分离的，因为这些树对负类的正确预测以及混合类的两个步骤是如何形成的更有信心。很明显，我们有很多分类正确的观测数据和一些由于数据中的噪声而出现的误差较大的观测数据。

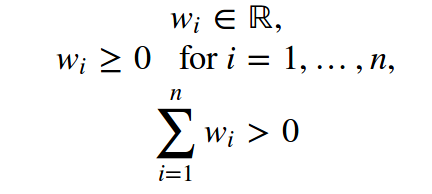
### 重量

有时，我们需要一个更具体的损失函数来解决我们的问题。例如，在金融时间序列中，我们可能希望对时间序列中的大变动给予更大的权重；对于客户流失预测，预测具有高LTV（或终身价值：客户未来将带来多少钱）的客户的流失更为有用。

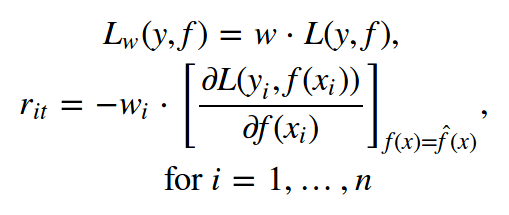


统计战士会发明自己的损失函数，写出它的梯度（为了更有效的训练，包括Hessian），并仔细检查这个函数是否满足所需的属性。然而，很有可能在某个地方犯错误，遇到计算困难，在研究上花费过多的时间。

取而代之的是，发明了一种非常简单的仪器（在实践中很少被记住）：测量观测值并分配权重函数。最简单的例子就是为班级平衡设置权重。一般来说，如果我们知道输入变量和目标变量中的某些数据子集对我们的模型具有更大的重要性，那么我们只需赋予它们更大的权重w（x，y）。主要目标是满足重量的一般要求：

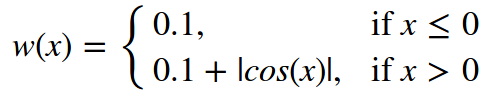


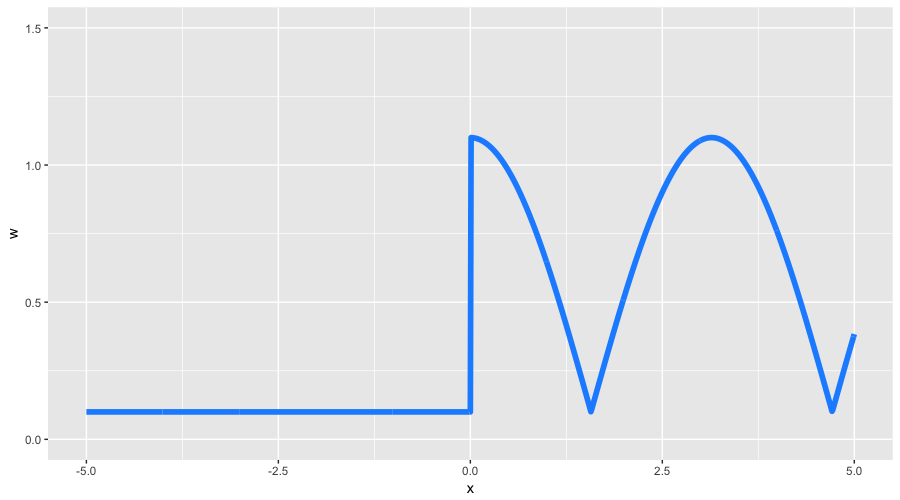
权重可以显著减少为我们正在解决的任务调整损失函数所花费的时间，还可以鼓励使用目标模型的属性进行实验。分配这些权重完全是创造力的作用。我们只需添加标量权重：



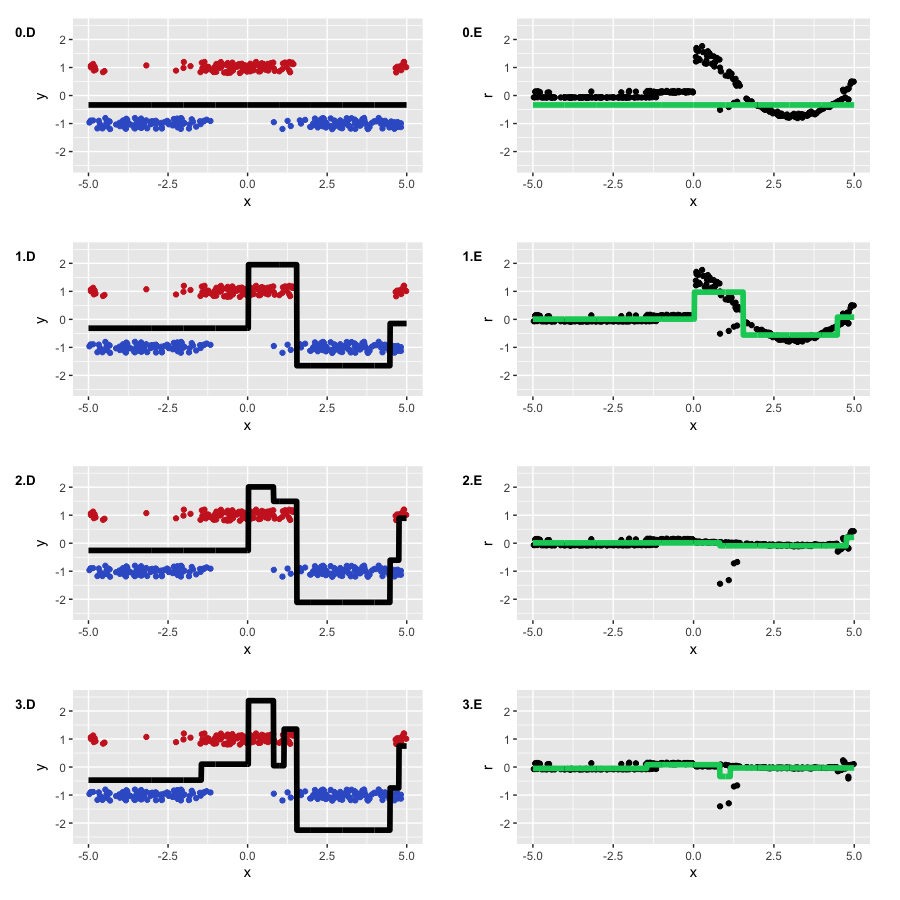
显然，对于任意权重，我们不知道模型的统计性质。通常，将权重链接到值y可能过于复杂。例如，在L1损失函数中使用与y成比例的权重不等于L2损失，因为梯度将不考虑预测本身的值。

我们提到这一切是为了更好地理解我们的可能性。让我们为玩具数据创建一些非常奇特的权重。我们将定义一个强不对称权函数如下：





利用这些权重，我们期望得到两个性质：X的负值的细节较少，函数的形式与初始余弦相似。我们从上一个例子中选取了另一个GBM的调谐，分类包括寻找最优系数的线搜索。让我们看看我们有什么：



我们达到了预期的结果。首先，我们可以看到伪残差的差别有多大；在初始迭代中，它们看起来几乎像原始余弦。其次，函数图的左边部分经常被忽略，而右边部分的权重更大。第三，我们在第三次迭代中得到的函数得到了足够的关注，并开始看起来与原始余弦相似（也开始有点过拟合）。

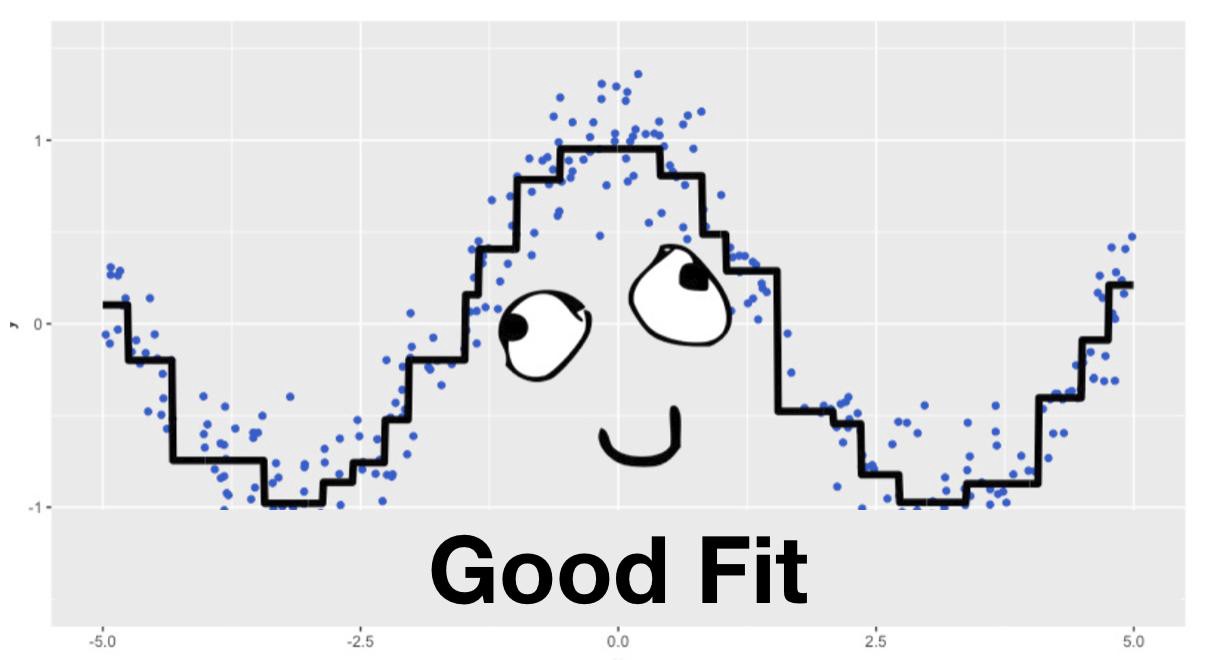
权重是一个强大但风险很大的工具，我们可以使用它来控制模型的属性。如果你想优化你的损失函数，首先尝试解决一个更简单的问题是值得的，但是要根据你的判断给观察值增加权重。

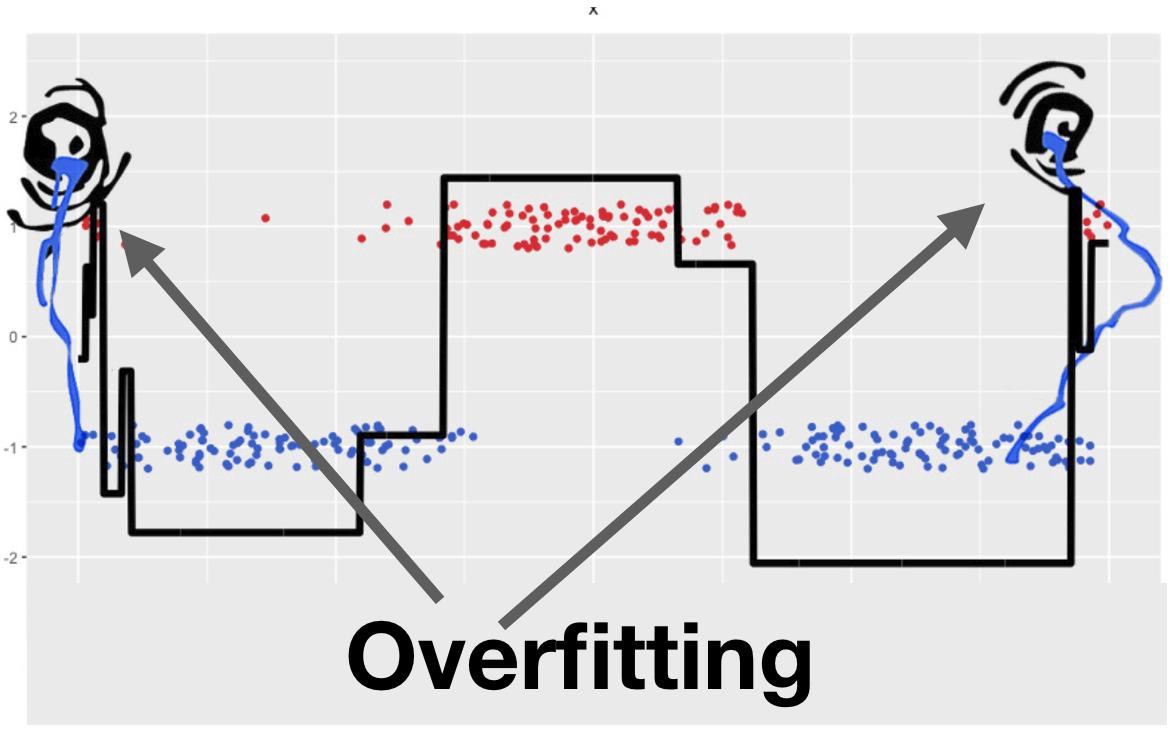
### 结论

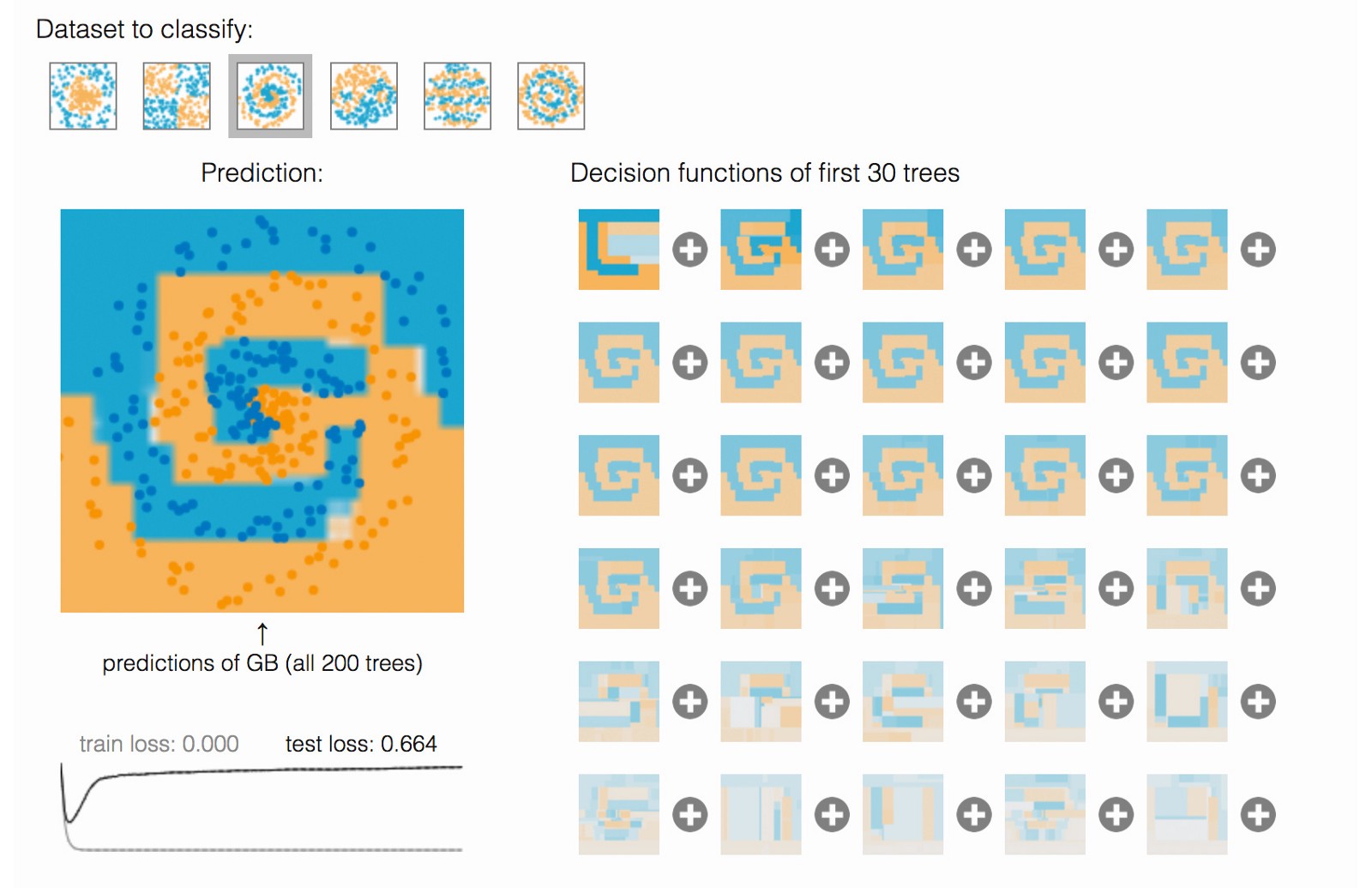
今天，我们学习了梯度增强背后的理论。GBM不仅是一些特定的算法，而且是构建模型集合的一种常用方法。此外，该方法具有足够的灵活性和可扩展性，可以训练大量的模型，考虑到具有不同加权函数的不同损失函数。

实践和ML竞赛表明，在标准问题（除了图像、音频和非常稀疏的数据）中，GBM通常是最有效的算法（更不用说堆叠和高级集成，其中GBM几乎总是其中的一部分）。此外，还有许多GBM的改编（Minecraft，ICML 2016）。顺便说一句，Viola-Jones算法，目前仍在计算机视觉中使用。

在这篇文章中，我们故意省略了有关GBM的正则化、随机性和超参数的问题。我们在整个过程中使用了少量迭代M=3并不是偶然的。如果我们使用30棵树而不是3棵树，并按照所述对GBM进行训练，结果将无法预测：







### 四。作业#10

课程每周（2018年10月1日）都会公布作业的完整版本。同时，您可以使用演示版本：。

### 5个。有用资源

* Jerome Friedman关于GBM
* Alexey Natekin和Alois Knoll的“梯度助推机器，教程”
* [Chapter in Elements of Statistical Learning](http://statweb.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf) from Hastie, Tibshirani, Friedman (page 337)
* [Wiki](https://en.wikipedia.org/wiki/Gradient_boosting) article about Gradient Boosting
* [Frontiers tutorial](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/) article about GBM
* [Video-lecture by Hastie](https://www.youtube.com/watch?v=wPqtzj5VZus) about GBM at h2o.ai conference

Author: [Alexey Natekin](https://www.linkedin.com/in/natekin/), OpenDataScience founder, Machine Learning Evangelist. Translated and edited by [Olga Daykhovskaya](https://www.linkedin.com/in/odaykhovskaya/), [Anastasia Manokhina](https://www.linkedin.com/in/anastasiamanokhina/), [Egor Polusmak](https://www.linkedin.com/in/egor-polusmak/), and [Yuanyuan Pao](https://www.linkedin.com/in/yuanyuanpao/).