CHAPTER 10

Improving Models and Data Extraction 模型的改进以及数据提取

How do you go about improving upon a simple machine learning algorithm such as Naive Bayesian Classifiers, SVMs, or really any method? That is what we will delve into in this chapter, by talking about four major ways of improving models:

• Feature selection

• Feature transformation

• Ensemble learning

• Bootstrapping

关于如何改进一个已知的简单模型的机器学习算法，比如贝叶斯分类器、支持向量机(SVMs)或是其他算法，我们将在本章从以下四个方面进行如何改进的探讨：

* 特征选择
* 特征转换
* 集成学习
* 数据重抽样

I’ll outline the benefits of each of these methods but in general they reduce entanglement, overcome the curse of dimensionality, and reduce correction cascades and sensitivity to data changes.

我将分别概述这些优化方法的好处。总的来说，这些方法都是为了减少数据间的耦合相关性，降低数据维度，并减少校正级联和数据变化的敏感性。

They each have certain pros and cons and should be used when there is a purpose behind it. Sometimes problems are so sufficiently complex that tweaking and improvement are warranted at this level, other times they are not. That is a judgment people must make depending on the business context.

每个方法都有各自的优缺点，都有其特定的使用场景。如果是比较复杂的问题就需要进行必要的调整和改进，还需要根据每个商业案例的情况进行具体情况具体分析。

Debate Club

辩论俱乐部

I’m not sure if this is common throughout the world, but in the United States, debate club is a high school fixture. For those of you who haven’t heard of this, it’s a simple idea: high schoolers will take polarizing issues and debate their side. This serves as a great way for students who want to become lawyers to try out their skills arguing for a case.

我不确定辩论俱乐部在全世界其他高校是否都存在，但是在美国每个高校都有这样一个俱乐部。其实这只是一个非常简单的场所，就是提供给高中生们一个能够将他们对某些问题的看法坚持下来并且与他们意见不一致的同学进行辩论的场所。这也给那些将来立志当律师的学生尝们试就某些案例进行相关辩论的一个最好的实践方式。

The fascinating thing about this is just how rigorous and disciplined these kids are. Usually they study all kinds of facts to put together a dossier of important points to

make. Sometimes they argue for a side they don’t agree with but they do so with conviction.

比较有趣的地方是孩子们都会严格遵守相关的辩论规则通常他们会收集各种事件相关的事实案例并且规整成卷宗。有些时候其实他们的真实想法与他们辩论时所处的正反方并不一致，但是他们也会按照的辩论规则捍卫他们当前的正方(或者反方)的结论。

Why am I telling you this? These debate club skills are the key to making machine learning algorithms (and many cases any algorithm) work better:

• Collecting factual and important data

• Arguing different points of view in multiple ways

为什么我要告诉你辩论俱乐部的事情？事实上辩论俱乐部的核心技能就是我们解开机器学习算法的钥匙，即如何使大多数情况下算法更好的工作取决于下面两点：

* 收集相关事实和重要数据
* 争论不同的观点(以多种方式)

As you can imagine, if we could collect important or relevant data to feed into our models, and try different methods or approaches to the same problem, we will itera‐ tively get better as we find the best model combination.

你可以想象一下，假如我们能够收集到各种重要数据或者相关数据用以填充我们的模型，并且对于同样一个问题可以尝试各种不同的方法和解决途径进行多次迭代，那么我们终究可以找到一个最优模型解

This gets us into what we will be talking about: picking better data or arguing for solutions more effectively.

下面我们将分两方面继续进行深入探讨：如何选择更好的数据以及如何有效的解决争论性的问题

Picking Better Data

选择更好的数据

In this section we’ll be discussing how to pick better data. Basically we want to find the most compact, simplest amount of data that backs up what we are trying to solve. Some of that intuitively means that we want the data that supports our conclusion, which is a bit of cart before the horse; regardless, there are two great methods to improve the data one is using: feature selection and feature transformation algorithms.

在本节中我们将讨论如何选择更好的数据。基本原则是我们希望找到最紧凑最简单的数据集合用以支持我们所试图解决的问题。直观意义上就是指我们试图找到那些最能支持我们结论的那些数据，虽然这听上去似乎有些本末倒置(即有了结论以后反过来去找数据支撑)。尽管如此，我们将使用特征选择以及特征转换这两个伟大的方法来改善我们数据并依此做出选择。

This sounds like a great idea, but what is the motivation behind picking better data?

上述的方法听上去好像是个好主意，但是选取数据背后的动机究竟是什么？

Generally speaking, machine learning methods are better suited for smaller dimen‐ sions that are well correlated with the data. As we have discussed, data can become extremely overfit, entangled, or track improperly with many dimensions. We don’t want to under- or overfit our data, so finding the best set to map is the best use of our time.

通常来说，机器学习方法更适用于低维度并且数据之间依赖关系比较大的集合。正如我们之前所讨论的过，数据模型的构建中常常会出现过拟、或者关联了无关的冗余维度导致过拟等现象。我们并不希望出现欠拟或者过拟问题，因此需要找到最佳的数据集才是节省时间的最优方案。

Feature Selection

Let’s think about some data that doesn’t make a whole lot of sense. Say we want to measure weather data and want to be able to predict temperature given three variables: “Matt’s Coffee Consumption,” “Ice Cream Consumption,” and “Season” (see Table 10-1 and Figure 10-1).

特征选择

我们以一组看上去并没有太大意义的数据来讨论如何进行特征选择。假如我们想测量的对象是气象数据，而我们预测气温的依据基于给定的三个变量：Matt咖啡的消费、冰淇淋的消费、季节

Table 10-1. Weather data for Seattle

表10-1 西雅图天气数据

Obviously you can see that I generally drink about 4 cups of coffee a day. I tend to eat

more ice cream in the summertime and it’s generally hotter around that time.

通过数据表你显然可以看到我通常每天会喝掉大概4杯咖啡。如果天气变热或者是夏天的时候我会吃掉更多的冰淇淋。

But what can we do with this data? There are at most N choose K solutions to any

data set, so given N dimensions, we can find an enormous number of combinations

of various-sized subsets.

但是我们能用这些数据做什么呢？从N维数据中尝试找到K种解决方式，对于指定的N维数据，我们可以找到通过排列组合出巨量的包含各种长度子集。

At this point we want to reduce the amount of dimensions we are looking at but don’t

know where to start. In general we want to minimize the redundancy of our data

while maximizing the relevancy. As you can imagine this is a tradeoff: if we keep all

the data, then we’ll know 100% that we have relevant data whereas if we reduce some number of dimensions we might have redundancy—especially if we have lots and lots of dimensions.

基于这点，我们希望降低维度以方便我们更好的观测数据，但是我们并不知道如何开始。简而言之，我们希望在最大限度的减少冗余数据的同时最大限度的提高数据的相关性。你可以把这想象成是一种权衡：如果保留了全部的数据以及维度信息，我们确实能够100%的了解数据间的相关性，但是庞大的维度和巨量的数据会给我们模型计算带来困难。因此必须进行一些数据和维度上的裁剪，而且我们相信某些数据和维度存在冗余关系(这些冗余的数据和维度显然是可以被裁剪的)，特别是当维度非常非常大的时候。

We have talked about this before as being an entanglement problem with having too

many data points that point to the same thing.

我们之前已经谈论过这个问题，很多数据往往会指向同一个问题。

In general, redundancy and relevancy are calculated using the same metrics and on a spectrum:

•Correlation

•Mutual information

•Distance from some point (Euclidean distance from reference)

在一般情况下，冗余性和相关性通常从下面几个范围进行度量：

•相关性

•互信息

•点之间的距离(参考欧式距离)

So they actually end up measuring the same thing. How do we solve this?

其实依据这些度量标准所测量的都是同一个东西，我们如何解决这个问题？

Let’s first take a step back and think about what would happen if we just looked at all possibilities.

我们可以先退一步，首先想想所有发生的可能性是什么？

Exhaustive Search

穷举法搜索

Let’s imagine that in this case we want to find the best possible dimensions to train

on. We could realistically just search through all possibilities. In this case we have

three dimensions which would equate to seven models (123, 12, 13, 23, 1, 2, 3). From

here we could say that we want to find the model that has the highest accuracy

(Figure 10-2).

首先想象一下在下面这个例子中我们希望找到一个最佳的维度模型进行训练。我们可以实事求是的搜索所有的可能性（穷举法）。在这个情况下，我们有三个维度，这将等同于七个模型 (123, 12, 13, 23, 1, 2, 3）。基于这点，我们可以认为我们想找到的模型具有较高的精度。

Figure 10-2. Exhaustive search for best features

穷举法去搜索最佳的功能特性

This unfortunately doesn’t work as well as you go up in dimensions. If for instance

you have 10 dimensions, the possibilities from selecting 10 dimensions, to 1 dimen‐

sion would be 2^10– 1. This can be denoted in Pascal’s triangle (Figure 10-3) as a sum of combinations where:

不幸的消息是如果随着维度持续增加，穷举法将无法正常工作。例如你有10个维度，穷举选择10个维度的所有可能性需要2的10次方-1.这可以在帕斯卡三角(Pascal’s triangle)表示成一个组合

Figure 10-3. Pascal’s triangle

图10-3 帕斯卡三角

Pascal’s triangle shows all combinations for a given row. Since each row sums up to

2^n, all we need to do is subtract 1, so we don’t account for zero dimensions.

帕斯卡三角显示了给定行的所有组合。每行的总和是2的N次方，我们所需要做的只是在总和上减去1，这是因为我们不需要考虑零维的情况。

So as you add dimensions you would have to account for 2^n– 1 possible data sets. If

you had 3,000 dimensions (which would be a good reason to use feature selection),

you would have roughly a trecentillion (10^903) models to run through!

因此当你添加维度为N时，你需要计算所有数据集合就是2的N次方-1 。当我们有3000个维度时(这就是为何要使用特征选择的原因)你需要穷举的模型次数数量级达到了10的903次方！

译者注： 参考：<http://www.urbandictionary.com/define.php?term=trecentillion>

trecentillion的意思就是1后面有903个零

Surely there is a better way. We don’t need to try every model. Instead, what if we just randomly selected features?

肯定有比穷举更好的解决办法。例如，接下来章节我们将看到如果不穷举每一个模型，而是尝试随机选取一些特征会怎样呢？

Random Feature Selection

随机特征选择

A lot of the time random feature selection will be useful enough for our purposes.

Reducing the features by half or a certain amount is an excellent way of improving

data overfitting. The added benefit is that you really don’t have to think about it much

and instead try out a random feature selection of a certain percent.

毫无疑问，大量的时间随机特征选择有助于我们达到建立正确模型。减少一半或者一定数量的样本选取将有助于消除过拟现象。这样做额外的好处在于不需要你多加思考去纠结于那些特征该进行取舍，而是直接尝试去选取一个随机的特征并且依照某个百分比选取对应的数据集。

Say for instance you want to reduce the features by 25%. You could randomly see how

it performs for accuracy, precision, or recall. This is a simple way of selecting features,

but there is one major downside: what if training the model is slow? You are still

brute-forcing your way to finding features. This means that you are arbitrarily pick‐

ing a number and hoping for the best. Surely there is a better way.

比方说你希望减少25%的特征（或者说是维度）。你可以随机的选取25%的特征集进行删除，然后再按照模型对剩下的数据集进行精度预测，等所有操作都做完以后再召回这些删除的特征集合。这是最简单的特征选择方法。但是它有个致命的缺点，如训练模型一次所耗费的时间太长怎么办？如果此时你仍然执着于去选择模型最优解，那么你将耗费很长很长的时间。因此我们肯定还有更好的办法去解决这个问题。

A Better Feature Selection Algorithm

一个更好的特征选择算法

Instead of relying on random feature selection, let’s think a little more in terms of

what we want to improve with our model. We want to increase relevancy while reducing redundancy.

不依赖于随机特征选取，我们回过头来想一下我们优化模型的目的是什么？对了，我们的目的在于增加相关性的同时剔除冗余特征。

Relevancy is a measure of how relevant the dimension in question is versus the classification whereas redundancy is a measure of how redundant the dimension is compared to all the other dimensions.

关联性是指度量维度之间数据与分类的相关程度，冗余是指定维度与所有其它维度相比的冗余程度。

Usually for relevancy and redundancy you either use correlation or mutual information.

对于关联性和冗余度我们通常可以用相关性或者互信息来进行处理

Correlation is useful for data that is continuous in nature and not nominal.

By contrast, mutual information gives us a discrete measure of the mutual information shared between the two dimensions in question.

相关性对于自然界的连续性数据或者是非名词性属性的数据处理非常有用。与之相对立的，互信息则对处理两个维度之间的共享信息给出了一个数据分离方式。

Using our earlier example, correlation would look like the results in Table 10-2 for

relevancy and Table 10-3 for redundancy.

以我们之前的例子为例，表10-2表示的是维度与气温之间的相关性， 表10-3表示的是维度之间的冗余度

Table 10-2. Relevancy using correlation 维度与气温之间的相关性

Table 10-3. Redundancy using correlation 维度之间的冗余度

As you can see from these two tables, ice cream is highly correlated with temperature

and my coffee consumption is somewhat negatively correlated with temperature; the

month seems irrelevant. Intuitively we would think month would make a huge differ‐

ence, but since it runs on a modular clock it’s hard to model using linear approxima‐

tions. The redundancy is more interesting. Taken out of context my coffee consumption and month seem to have low redundancy, while coffee and ice cream

seem more redundant.

正如你在两张表中所看到的，冰淇淋的销量与气温是正相关性，但是咖啡则是负相关，月份与温度的相关性很低。直觉上我们认为月份会与气温有很大的相关性，但是由于我们采用的是线性相关的模型，所以这种月份如时钟一样的周期化循环过程很难采用线性过程进行模拟，所以此处的相关性很低。冗余度更有意思，从上下文来看，咖啡的消费和月份之间的冗余度很低，但是咖啡和冰淇淋之间的冗余度很高。

So what can we do with this data? Next I’m going to introduce a significant algorithm

that brings this all together

那么我们可以拿这些数据做什么呢？接下来我们将介绍一个如何使用这些数据的重要算法。

Minimum Redundancy Maximum Relevance Feature Selection

最小冗余最大相关性的特征选择

To bring all of these competing ideas together into one unified algorithm there is

minimum redundancy maximum relevance (mRMR) feature selection, which aims to

maximize relevancy while minimizing redundancy. We can do this using a maximiza‐

tion (minimization) problem using NumPy and SciPy.

把这些相互竞争的因素糅合在一起的统一算法称为mRMR(minimum redundancy maximum relevance)特征选择，目标是保持最大相关性的同时冗余度也最小。我们可以用python库中的Numpy和Scipy库来求解这个最大(最小)问题。

In this formulation we can just minimize the following function:

首先在公式中做如下简化：

More importantly in code we have:

更多的细节代码如下：

此处为Page 184页代码

This gives us almost exactly what we expected: my ice cream consumption models the

temperature quite well. For bonus points you could use an integer programming

method to get the values to be either 0 or 1, but for these purposes it’s obvious which

features should be selected to improve the model.

结果正如我们所期望的一样:我的冰淇淋消费与气温关系的模型成功建立了起来。作为额外的奖励，你也可以采用整数规划法将值变为0或者1，这样做的好处就是容易甄别出选择哪些特征可以有效的改善模型。

Feature Transformation and Matrix Factorization

特征变换与矩阵分解

We’ve actually already covered feature transformation quite well in the previous chapters. For instance, clustering and the kernel trick are both feature transformation methods, effectively taking a set of data and projecting it into a new space, whether it’s a cluster number or an expanded way of looking at the data. In this section,though, we’ll talk about another set of feature transformation algorithms that are rooted in linear algebra. These are generally used to factor a matrix down to a smaller size and generally can be used to improve models.

实际上我们已经在前面章节中覆盖了特征变换相关内容。例如，聚类和核变换都是特征变换里的方法：即有效地采样一组数据，并投射到一个新的空间，无论是簇数还是其他扩展模式都是分析数据的方式。在本节中，我们将讨论另外一种植根于线性代数的特征变换算法。它通常用于将矩阵维度降到更小的尺寸，并可以用来改进模型。

To understand feature transformation, let’s take a look at a few algorithms that transform a matrix into a new, more compressed or more verbose version of itself: principal component analysis and independent component analysis.

要理解特征变换，首先我们来看看这个新算法是如何将矩阵转换成一个压缩比更高而且更详细的新版本：主成分分析和独立成分分析。

Principal Component Analysis

主成分分析

Principal component analysis (PCA) has been around for a long time. This algorithm

simply looks at the direction with the most variance and then determines that as the

first principal component. This is very similar to how regression works in that it

determines the best direction to map data to. Imagine you have a noisy data set that looks like Figure 10-4.

主成分分析（PCA）这个算法由来已久。该算法可以简单地认为是找到方向与方差的最大，然后将其定为第一主成分。这与回归的原理非常相似，它决定了将数据映射到哪个方向最合适。假设你有一个噪声数据集如图10-4所示。

As you can see, the data has a definite direction: up and to the right. If we were to

determine the principal component, it would be that direction because the data is in

maximal variance that way. The second principal component would end up being

orthogonal to that, and then over iterations we would reduce our dimensions by

transforming them into these principal directions.

如上图所示，数据有一个明确的方向：向上和向右。我们需要确定的主成分肯定是数据的方向最大方差值。第二主成分将通过正交和迭代后转换到主要方向上面，这样我们就达到了减少维度的目的。

Another way of thinking about PCA is how it relates to faces. When you apply PCA

to a set of faces, an odd result happens known as the Eigenfaces (see

Figure 10-5).

关于PCA的另一种思考的例子就是它与面部的关系。当你应用PCA模型去处理一系列的面孔时，一个奇怪的结果将出现，我们称之为特征脸（见图10-5）。

Figure 10-5. Eigenfaces (Source: AT&T Laboratories)

图10-5 特征脸（图片来源:AT&T实验室）

While these look quite odd, it is fascinating that what comes out is really an average face summed up over all of the training data. Instead of implementing PCA now, we’ll wait until the next section where we implement an algorithm known as independent component analysis (ICA), which actually relies on PCA as well.

虽然这些脸咋看上去很奇怪，但其实这是真正分析了所有的训练数据而总结出来的迷人的平均脸谱。关于PCA的具体实现，我们将在下一节中进行详细讨论，并依赖于PCA实现一个称之为独立分量分析（ICA）的算法。

Independent Component Analysis

独立分量分析

Imagine you are at a party and your friend is coming over to talk to you. Near you is

someone you hate who won’t shut up, and on the other side of the room is a washing

machine that keeps making noise (see Figure 10-6)

想象你身处一个聚会上，你的朋友想过来和你说话。此时靠近你的是一个你讨厌的但是不会闭嘴的人，与此同时在房间的另一边是一个正在运行的洗衣机，不停的制造噪音（见图10-6）

Figure 10-6. Cocktail party example

图10-6。鸡尾酒会的例子

You want to know what your friend has been up to, so you listen to her closely. Being

human, you are adept at separating out sounds like the washing machine and that

loudmouth you hate. But how could we do that with data?

你想知道你的朋友在说什么，所以你得靠近她仔细倾听。作为人类，你能够分离出像洗衣机那样的噪声或是你不喜欢的声音。但是，我们如何能够做到让数据分离这些噪声？

Let’s say that instead of listening to your friend, you only had a recording and wanted

to filter out all of the noise in the background. How would you do something like

that? You’d use an algorithm called ICA.

还是以倾听朋友为例，假如你只有一个录音机但是需要滤除背景中的所有噪声，你会怎么做？你需要使用一个称为ICA的算法。

Technically, ICA minimizes mutual information, or the information shared between

the two variables. This makes intuitive sense: find me the signals in the aggregate that are different.

从技术上讲，ICA算法最大限度地减少互信息，或是共享的信息中的两个变量。这一种直观的感觉：总有不同的信号找到我。

Compared to our face recognition example in Figure 10-5, what does ICA extract?

Well, unlike Eigenfaces, it extracts features of a face, like noses, eyes, and hair.

PCA and ICA are useful for transforming data and can analyze information even bet‐

ter (see Figure 10-7). Then we can use this more succinct data to feed our models

more useful and relevant information, which will improve our models beyond just

cross-validation.

以图10-5中我们的人脸识别为例，什么称之为ICA提取？嗯，与PCA只提取出特征脸不一样，ICA算法需要提取脸、鼻子、眼睛和头发等各个特征。PCA和ICA都是转换数据的有效手段，甚至可以更好的分析信息(见图10-7）。然后，我们可以使用这个更简洁的数据来补充我们的模型，使之相关的信息更加完善，这也将使我们的模型比一般的交叉验证更有效。

Figure 10-7. ICA extraction example

图 10-7 ICA 提取样例

Now that we know about feature transformation and feature selection, let’s discuss

what we can do in terms of better arguing for a classiciation or regression point.

现在我们已经了解了特征变换和特征选择，让我们讨论一下基于如何利用它们来改善我们的分类和回归模型

Ensemble Learning

集成学习

Up until this point we have discussed selecting dimensions as well as transforming

dimensions into new ones. Both of these approaches can be quite useful when

improving models or the data we are using. But there is yet another way of improving

our models: ensemble learning.

至此为止，我们已经讨论了维度选择以及维度变换。这两种方法对于改进模型或是正使用的数据集都是非常有用的。但还有另一种改进我们模型的方法：集成学习。

Ensemble learning is a simple concept: build multiple models and aggregate them

together. We have already encountered this with random forests in Chapter 5.

集成学习的概念非常简单：建立多个模型并将它们聚合在一起。其实我们在第5章讨论的随机森林就是集成学习的一种方式。

A common example of ensemble learning is actually weather. When you hear a forecast for the next week, you are most likely hearing an aggregation of multiple weather models. For instance, the European model (ECMWF) might predict rain and the US model (GFS) might not. Meterologists take both of these models and determine which one is most likely to hit and deliver that information during the evening news.

集成学习的一个常见例子是天气预报。当你听到下周的天气预报时，你很可能会听到多种天气模式的聚集结果。例如，欧洲模型（ECMWF）可以预测下周是否下雨而美国国家环境预报中心的全球预报系统模型（GFS）则可能做不到。气象学家综合考虑了这些模型后的预测结果，确定那些天气状况是最有可能出现的以后，就会在晚间新闻传递出来综合的信息。

译者注： ECMWF模型关注于未来24小时到7天内天气情况，GFS模型关注于未来1周到2周内的天气情况。

When aggregating multiple models, there are two general methods of ensemble learning: bagging, a naive method; and boosting, a more elegant one.

当聚合多个模型时，通常有两种集成学习的方法：自抽样法（Bagging），一个简单的方法；boosting，一个更优雅的方法。

注：boosting暂时用

Bagging

自抽样法

Bagging or bootstrap aggregation has been a very useful technique. The idea is simple: take a training set and generate new training sets off of it.

自抽样法(bagging,也是bootstrap aggregating的缩写)是一个非常有用的技术。这个想法很简单：每轮新的训练集从初始的训练集中随机生成出来。

Let’s say we have a training set of data that is 1,000 items long and we split that into 50 training sets of 100 a piece. (Because we sample with replacement, these 50 training sets will overlap, which is okay as long as they are unique.) From here we could feed this into 50 different models.

假设总的训练集包含1000个项目数据，我们把它分成50轮训练集，每组100个数据。（因为我们允许样本数据在某个训练集中出现多次，因此这50个训练集将可能交叉使用部分数据，但是这不会影响我们的结果，只要每个训练集之间是独一无二的即可）然后将这些训练集供给50个不同的模型。

Now at this point we have 50 different models telling us 50 different answers. Like the weather report just mentioned, we can either find the one we like the most or do something simpler, like average all of them.

现在，我们就拥有有50种不同的模型以及50个不同的答案。就像刚才提到的天气报告一样，我们从中挑取出最喜欢的那个答案或者简单的在所有答案中求取均值。

This is what bootstrap aggregating does: it averages all of the models to yield the average result off of the same training set. The amazing thing about bagging is that in practice it ends up improving models substantially because it has a tendency to remove some of the outliers.

这就是自抽样法所做的事情：它平均了所有的模型所产生的结果，这些训练数据都来自同一个训练集。关于自抽样法的神奇之处在于模型最终得到了改进，因为它去除了一些异常值倾向的数据。

But should we stop here? Bagging seems like a bit of a lucky trick and also not very elegant. Another ensemble learning tool is even more powerful: boosting.

但我们的讨论该到此为止吗？自抽样法似乎有点幸运的把戏，而且也不是很优雅。下面我们来介绍另一个更强大的集成学习工具：促进法（boosting）。

Boosting

促进法

Instead of splitting training data into multiple data models, we can use another

method like boosting to optimize the best weighting scheme for a training set.

与之前将训练数据分割成多个数据模型不同，我们可以使用另一个称之为促进法(Boosting)的方案来优化训练集中的最佳加权方案。

Given a binary classification model like SVMs, decision trees, Naive Bayesian Classifiers, or others, we can boost the training data to actually improve the results.

给定一个二进制的分类模型，如支持向量机(SVMs)，决策树，朴素贝叶斯分类器，或其它的模型，我们都可以提高训练数据的质量并改善其结果。

Assuming that you have a similar training set to what we just described with 1,000 data points, we usually operate under the premise that all data points are important or that they are of equal importance. Boosting takes the same idea and starts with the assumption that all data points are equal. But we intuitively know that not all training points are the same. What if we were able to optimally weight each input based on what is most relevant?

假设你有一个训练集，正如我们刚才所描述的1000个数据点。我们通常操作的假设前提是：所有的数据点都很重要，或他们是同等重要的。基于同样的想法，并开始假设所有的数据点都是平等的。但是直觉告诉我们：并非所有的训练点都是同等重要的。如果我们能够根据每一个训练点的相关权重来优化每一次输入会怎样呢？

That is what boosting aims to do. Many algorithms can do boosting but the most popular is AdaBoost.

这正是促进法的目标。许多算法都能达到这个目的，但目前最流行的是AdaBoost。

To use AdaBoost we first need to fix up the training data just a bit. There is a requirement that all training data answers are either 1 or –1. So, for instance, with spam classification we would say that spam is 1 and not spam is –1. Once we have changed our data to reflect that, we can introduce a special error function:

要使用AdaBoost，我们首先需要修改一下训练数据集。因为这个方法的前提就是要求所有的训练数据答案要么是1要么是- 1。因此，以垃圾邮件分类为例，我们会说，垃圾邮件是1，而不是垃圾邮件是- 1。一旦我们修改了数据集之后，就可以引入一个特殊的误差函数：

注:此处有一公式

This function is quite interesting. Table 10-4 shows all four cases.

这个公式很有意思。表10-4显示了所有的4种用例

Table 10-4. Error function in all cases

表10-4 误差公式的所有用例

As you can see, when f(x) and y equal, the error rate is minimal, but when they are

not the same it is much higher.

正如你所看到的，当f（x）和y相等时，错误率是最小的，当它们不一样时，错误率就高得多。

From here we can iterate through a number of iterations and descend on a better

weighting scheme using this algorithm:

因此，我们可以使用该算法通过一系列的迭代和梯度下降获得一个更好的加权方案：

Choose a hypothesis function (either SVMs, Naive Bayesian Classifiers, or some thing else)

选择一个假设函数（无论是支持向量机SVMs，朴素贝叶斯分类器，或是其他的）

—Using that hypothesis, sum up the weights of points that were miss classified:

-基于该假设函数，加和所有的错误分类的权重点：

注:此处有一公式

—Choose a learning rate based on the error rate:

-基于错误率计算学习率：

Add to the ensemble:

添加到全局函数：

注:此处有一公式

Update weights:

更新权重：

注:此处有一公式

for all weights

此处应更新所有的权重

Renormalize weights by making sure they add up to 1

归一化权重并确定所有的权重总和为1

What this does is converge on the best possible weighting scheme for the training

data. It can be shown that this is a minimization problem over a convex set of functions.

上面所做的是对训练数据收敛的最佳加权方案。它也可以被理解成是一个最小化的凸函数组求解的问题。

This meta-heuristic can be excellent at improving results that are mediocre from any

weak classifier like Naive Bayesian Classification or others like decision trees.

这种元启发式可以很好的改善任何弱分类器并将它们组合成一个强分类器，如朴素贝叶斯分类或其他决策树等。

Conclusion

You’ve learned a few different tricks of the trade with improving existing models: feature selection, feature transformation, ensemble learning, and bagging. In one big

graphic it looks something like Figure 10-8.

结论

至此我们已经学会了各种不同的技巧用于改善现有的模型：特征选择、特征转换、集成学习和自抽样法。把它们都集成在了一张总图如图10-8所示。

Figure 10-8. Feature improvement in one model

图10-8. 特征改进的总图

As you can see, ensemble learning and bagging mostly focus on building many models and trying out different ideas, while feature selection and feature transformation

are about modifying and studying the training data.

可以看出，集成学习和自抽样法主要关注于多模型的建立和各种尝试，而特征选择和特征变换则更加关注于训练数据集的修改和研究。