斯坦福 ML 公开课笔记 11

本文对应 ML 公开课的第 11 个视频。前半部分仍然是讲学习理论的内容，后半部分主要讲述一些在具体应用中使用 ML 算法的经验。学习理论的内容包括贝叶斯统计和正则化(Bayesian statistics and Regularization)、在线学习(Online Learning)。ML 经验包括算法的诊断(Diagnostics for debugging learning algorithms)、误差分析(error analysis)、销蚀分析(ablative analysis)、过早优化(premature optimization)。

正则化

上一讲中讲到的特征选择方法是在简化模型，通过消除特征，减少模型的参数来降低过拟合的发生。正则化同样用于降低过拟合，但是它保留所有的参数。这是怎么做到的呢？

在之前的讲述中，求参数时一般使用极大似然估计(Maximum Likelihood)方法，即

$$\theta_{\mathrm{ML}} = arg\max_{\theta}\prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \tag{1}$$

在公示 1 中，我们认为θ是一个固定的未知值或向量，通过极大似然估计我们可以估计到θ的值。这是频率学派的看法。他们认为参数是是固定的，使用统计方法可以估计出这些值。

在统计学中，还存在另外一个学派，即贝叶斯派。同频率学派不同，他们认为θ是一个随机变量，服从某个先验分布。即θ~p(θ)。

在这种情况下，后验概率可以依下式来求：

$$\mathrm{p}(\theta|\mathrm{S}) = \frac{p(S|\theta)p(\theta)}{p(S)} = \frac{\left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)\right)p(\theta)}{\int\left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)\right)p(\theta)\,d\theta} \tag{2}$$

注意到公式 2 中与公式 1 中条件概率的差别，公式 2 中$p(y^{(i)}|x^{(i)},\theta)$中 x 与$\theta$之间以逗号分隔，表示$\theta$是随机变量。而公式 1 中$p(y^{(i)}|x^{(i)};\theta)$内 x 与$\theta$以分号隔开，表示$\theta$是一个具体的未知值。

公式 2 中，$p(y^{(i)}|x^{(i)},\theta)$的形式来源于所选模型，比如若使用贝叶斯逻辑斯蒂回归模型(bayesian logistic model,BLR)，那么$p\left(y^{(i)}\middle|x^{(i)},\theta\right) = h_\theta(x^{(i)})^{y^{(i)}}(1-h_\theta(x^{(i)}))^{1-y^{(i)}}$，其中$h_\theta\left(x^{(i)}\right) = 1/(1+\exp\{-\theta^T x^{(i)}\})$。

在对新样本进行预测时，使用如下公式计算后验概率：

$$\mathrm{p}(\mathrm{y}|\mathrm{x},\mathrm{S}) = \int p(y|x,\theta)p(\theta|S)\,d\theta \tag{3}$$

则预测公式如下：

$$\mathrm{E}[\mathrm{y}|\mathrm{x},\mathrm{S}] = \int y\mathrm{p}(\mathrm{y}|\mathrm{x},\mathrm{S})\,dy \tag{4}$$

但是，利用公式 3 和公式 4 的方法进行预测时，因为θ的积分使得计算量变得很大，当θ是高维向量时，往往是一个 NP 问题。所以在实际应用中，往往通过最大化后验概率来得到θ的估计，然后将θ代入到模型(比如上面的$h_\theta(x) = 1/(1+\exp\{-\theta^T\mathrm{x}\})$)中直接预测。$\theta$的估计公式如下。MAP 即为 maximum a posteriori。

$$\theta_{\mathrm{MAP}} = arg\max_{\theta}\left(\prod_{i=1}^{m} p\left(y^{(i)}\middle|x^{(i)};\theta\right)\right)p(\theta) \tag{5}$$

比较一下公式 5 与公式 1 的差别，仅仅是在末尾添加了先验概率 p(θ)。

在实际应用中，普遍选择正态分布作为θ的先验概率。即$\theta \sim N(0, \tau^2 I)$。实践表明，采MAP方法得到的参数比使用极大似然估计(ML)方法得到的参数更加不容易拟合，即降低了过拟合的概率。

在笔记3中，分析了最小二乘法与ML方法之间的解释关系。即最小二乘的概率解释是极大似然估计。那么MAP方法由解释了什么呢？

$$\min \sum_{i=1}^{m} ||y^{(i)} - \theta^T x^{(i)}||^2 \quad (ML) \tag{6}$$

$$\min \sum_{i=1}^{m} ||y^{(i)} - \theta^T x^{(i)}||^2 + \lambda||\theta||^2 \quad (MAP) \tag{7}$$

公式6即是ML和最小二乘法。公式7即是MAP和其对应的需要最小化的目标函数。在公式7中，有$\lambda = 1/\tau^2$。$\lambda||\theta||^2$也被成为正则化项，使用Laplace先验分布也可以达到类似的效果。λ值的选取可以采用笔记10中的交叉检验方法。

实际上，特征选择与正则化之间有一定的联系，可以这样考虑，当特征选择将一个特征删除的时候，等同于正则化时将该特征对应的θ分量设为0。

在文本分类问题上，特征数目有时会远远大于样本数目，这样就会产生过拟合。添加正则化项是解决该问题的一个不错的方法。

在线学习

之前学的算法都是批处理算法，即在训练集上得到模型后，再去对测试集或者训练集本身进行评测，得到训练误差和泛化误差。而在线学习并不这样，而是首先有一个初始的分类器，当第一个样本到来时，对该样本进行预测，得到预测结果，然后利用该样本的信息对分类器进行更新(比如，考虑感知器算法的更新规则，见笔记 1-2)；然后第二个样本到来时做同样的操作，以此类推。这样，我们就对 m 个样本都有一个预测值，只不过它们都是在训练的过程中得到的，对这些预测值进行统计，就得到了在线训练误差。这就是过程上在线学习与批处理的不同之处。

对于感知器算法来说，若正负样本线性可分，那么在线学习算法也是收敛的。

感知器在线学习收敛性证明

我们在笔记3的最后涉及了一点感知器的知识。其分类模型如下：

$$h_\theta(x) = g(\theta^T x) = \begin{cases} 0 & if(\theta^T x > 0) \\ 1 & otherwise \end{cases} \tag{8}$$

感知器算法的训练也比较简单，当一个新样本(x,y)到来时，若$h_\theta(x) = y$，则参数不发生变化，否则，使用如下公式进行参数更新：

$$\theta := \theta + yx \tag{9}$$

对于这样的感知器算法，可以证明，若正负样本线性可分，那么在线学习算法是可以收敛的。

首先，形式化的表述何为正负样本线性可分，样本可以用$(x^{(1)}, y^{(1)})$，$(x^{(2)}, y^{(2)})$，…，$(x^{(m)}, y^{(m)})$表示。假设对于所有的样本，均有$||x^{(i)}|| \le D$，线性可分的意义即为存在一个单位权重向量 u($||u|| = 1$)，使得对所有样本，均存在如下公式：

$$y^{(i)} * \left(u^T x^{(i)}\right) \ge \gamma$$

我们称公式9表示的是权重 u 至少以间隔γ将正负样本分开。

在这样的情况下，可以证明收敛定理：

在这个样本序列上的错误数不超过$(D/\gamma)^2$

证明如下：

令$\theta^{(k)}$为发生第 k 个错误时的权重值，则存在

$$y^{(i)} * \left(\left(\theta^{(k)}\right)^T x^{(i)}\right) \leq 0 \tag{10}$$

则有：

$$(\theta^{(k+1)})^T u = \left(\theta^{(k)}\right)^T u + y^{(i)}(x^{(i)})^T u \geq \left(\theta^{(k)}\right)^T u + \gamma \tag{11}$$

递推可以得到：

$$(\theta^{(k+1)})^T u \geq k\gamma \tag{12}$$

类似的，存在：

$$\begin{aligned} ||\theta^{(k+1)}||^2 &= ||\theta^{(k)} + y^{(i)}x^{(i)}||^2 = ||\theta^{(k)}||^2 + ||x^{(i)}||^2 + 2y^{(i)}(x^{(i)})^T\theta^{(k)} \\ &\leq ||\theta^{(k)}||^2 + ||x^{(i)}||^2 \leq ||\theta^{(k)}||^2 + D^2 \end{aligned} \tag{13}$$

递推可以得到：

$$||\theta^{(k+1)}||^2 \leq kD^2 \tag{14}$$

所以，最终有

$$\sqrt{k}D \geq ||\theta^{(k+1)}|| \geq (\theta^{(k+1)})^T u \geq k\gamma \tag{15}$$

在公式 15 中，第二个不等式源自

$$(\theta^{(k+1)})^T u = ||\theta^{(k+1)}|| * ||z|| * \cos\varphi \leq ||\theta^{(k+1)}|| * ||z|| = ||\theta^{(k+1)}|| \tag{16}$$

由公式 15，得到

$$k \leq (D/\gamma)^2 \tag{17}$$

得证。

机器学习算法应用建议

机器学习(ML，注意与极大似然估计区分)中有很多算法，那么在实际问题中应该怎样使用这些算法？当算法遇到瓶颈时该选择什么样的方向来对算法进行改进？本文的后半部分即是解决这样的问题。这些建议是针对应用已有算法到具体问题上，对发明新算法用处不大。

本文的后半部分主要分为三个主要内容：

1) 调试 ML 算法时如何进行诊断？
2) 误差分析与销蚀分析
3) 如何在一个 ML 问题上开始研究？

在对具体的内容分析之前，先介绍一个很有用的建议，即避免过早优化。

在我们进行项目开发或课题研究时，往往会遇到一些问题，在没有弄清楚问题之前，即没有明确的证据说明问题确实出在这里，我们往往想当然的对自己认为出问题的地方进行改进或者优化，运气好时能把问题解决，运气不好时则浪费了很多时间。

比如，在项目开发时，过早的优化不是瓶颈的代码段。虽然该段代码被优化到很快，但对系统的性能提高微乎其微，可以说是浪费了时间。而对于研究来说，过早的去做一些不能解决问题的事情浪费的时间可能会更多。因而，我们必须有一些能够判断问题所在的方法，这就引出了我们的下一节内容。

ML 算法诊断

设想这样的问题场景，对于垃圾邮件过滤问题，目下的研究现状如下：

- 在 50000 个特征中选择 100 个特征；

- 使用贝叶斯逻辑斯蒂回归(BLR)模型算法，其目标函数如公式 7 所示；
- 目前的误差率为 20%；

显然，20%的误差率难以接受。那么可能的解决方法如下：

1) 更多的数据；数据量变多往往有益于模型变好
2) 更少的特征；
3) 更多的特征；
4) 选择更好的特征；
5) 梯度下降方法多迭代几次；解决梯度下降算法没有收敛的问题
6) 尝试牛顿方法；牛顿方法可能更容易收敛
7) 调整公式 7 中λ的值；
8) 使用 svm 算法；

可能的方法数以百计，但我们只列出这么多吧。对于上述 8 种改进方法，如果一一尝试的话可能会非常耗时。比如，虽然增加数据一般会达到好的效果，但对一些应用来说，收集数据是很困难耗时的事情，如果你花了三个月的时间来收集数据，但最后发现增加数据后并没有使算法效果变好，那该是多悲催啊！

但面对这 8 种方法，如果选择改进的方向呢？只能靠拼 rp 么？当然不是，如果我们能找到几种标准，排除上面大部分的方向，只保留一两个，那么可能成功的概率会更大。

方差/偏差分析

第一个标准就是判定问题是出在高方差还是高偏差上。偏差(bias)/方差(variance)在笔记 9 中有介绍。一般说来，高方差针对的是过拟合问题，即训练误差很小但泛化误差却很大。而高偏差针对的模型本身不适合的情况，如特征数目过少等问题，表现即是训练误差和泛化误差都很大。看到下图可能会更清晰。

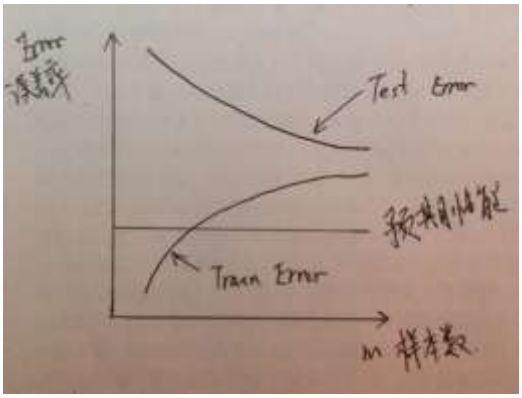

图 2 高偏差下的误差随样本变化图

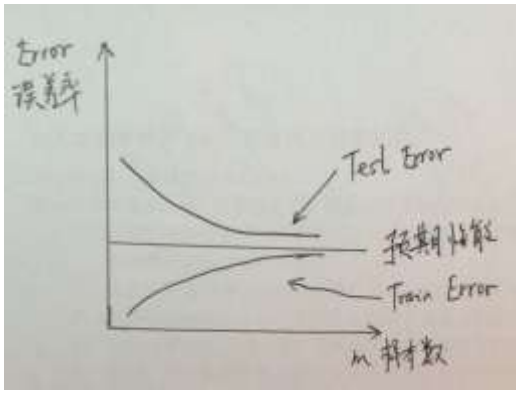

图 1 高方差下的误差随样本变化图

由图 1 可以看到，在高偏差下，训练误差和测试误差均会大于预期的性能。这个时候增加样本数目也是不能解决问题的。因为一般情况下，随着样本数的增加，训练误差会呈上升趋势。由图 2 可以看到，在高方差下可以比较训练误差和测试误差，如果它们之间相差很大，那么可以通过增加样本数目使得模型的过拟合程度减小，从而提高性能。

根据上面的分析，我们就可以对 8 条解决方案的前 4 条进行分类。更多的数据解决的是高方差的问题；更少的特征也可以降低过拟合的程度，即解决高方差问题；更多的特征和更好的特征可以增加模型的复杂度，提高模型在数据上的拟合程度，因而对应解决高偏差问题。

收敛与目标函数是否正确的判断

考虑下面的情景，仍是针对垃圾邮件判断问题，

- 使用 BLR 模型可以达到正常邮件上 2%的错误率，垃圾邮件上 2%的错误率；

- 使用 SVM 模型可以达到正常邮件上 0.01%的错误率，垃圾邮件上 10%的错误率。

对于 BLR 来说，正常邮件上 2%的错误率不能容忍，因为人们一般情况下不会想错过朋友发来的邮件或者一些重要的邮件。显然，在这个场景下，svm 比 BLR 要好。

对于这种情况，有如下两个分析：

- BLR 模型是否收敛？
- BLR 模型的目标函数是否找对？

这两种情况都有可能造成上面的 svm 优于 BLR 的问题。如果 BLR 没有收敛，svm 比 BLR 收敛的更好，所以 svm 效果更好；如果模型的目标函数没有找对，那么在当前的目标函数上，svm 可能逊于 BLR，但是 svm 在真正的目标函数上却能达到更好的效果，也可能造成上述问题的发生。

对于函数是否收敛来说，我们可以画出迭代次数与目标函数的趋势图，但是这样的趋势图在通常情况下不能分辨出目标函数是否稳定的趋势，因为在训练的后期目标函数的每步优化往往都只提高一点，而这个一点的优化成果可能持续出现在很多次迭代中。

这里，介绍一种更加直观的方法。

对于上面 svm 优于 BLR 的的问题，我们根据实际问题找到实际的需要优化的函数：

$$\mathrm{a}(\theta) = \frac{1}{m}\sum_{i=1}^{m} w^{(i)} I\{h_\theta(x^{(i)}) \neq y^{(i)}\} \tag{18}$$

通过对正常样本和垃圾样本的错误的惩罚进行加权，我们可以更好的将目标锁定为在提高垃圾邮件识别率的同时更好的保护正常邮件不被错判。

那么，我们可以根据实验结果来计算出 BLR 和 svm 的 a 指标，显然，根据假设，有：

$$\mathrm{a}(\theta_{svm}) > \mathrm{a}(\theta_{BLR})$$

现在，我们可以通过判断在 BLR 的目标函数上，θ_{svm}和θ_{BLR}的表现，来判断到底发生了什么问题，BLR 的目标函数如公式 7。

可能会有两种情况发生，第一种情况为$\mathrm{a}(\theta_{svm}) > \mathrm{a}(\theta_{BLR})$与$\mathrm{J}(\theta_{svm}) > \mathrm{J}(\theta_{BLR})$，这就表明 BLR 算法的收敛程度没有 svm 高，造成了其在实际问题中表现差，这个时候就需要改进训练算法，使之收敛程度更高。第二种情况为$\mathrm{a}(\theta_{svm}) > \mathrm{a}(\theta_{BLR})$与$\mathrm{J}(\theta_{svm}) < \mathrm{J}(\theta_{BLR})$，这种情况表明 BLR 的目标函数 J 是个错误的函数，其不能真实的反应人们在该问题上的需要，此时应该改进目标函数。

因而，我们就可以对 8 种解决方案的后四条进行分类。梯度下降算法多迭代几次与尝试牛顿方法都是为了使收敛性更好，即解决收敛程度问题。调整λ的值与使用 svm 算法则是改进了目标函数。

诊断实例

Ng 以他的某些学生正在做的一个自动驾驶直升机飞行的项目作为例子来分析如何找到问题所在。首先，对于一个能够自动驾驶直升机的程序来说，要经过如下步骤：

1) 建立一个精确的模拟器；
2) 选择一个损失函数 J。
3) 使用强化学习算法(Reinforcement Learning, RL)来对损失函数进行优化，输出参数θ_{RL}。

那么，可能的问题分析会是这样。如果算法能在模拟器上正常的驾驶直升机，而在实际中不能，那么问题出在模拟器与现实的差距太大，这时需要改进模拟器的精确性。让真实的人来驾驶直升飞机，得到参数θ_{human}，判断算法学到的参数的损失与真实人的参数损失的差别，若$\mathrm{J}(\theta_{\mathrm{RL}}) > \mathrm{J}(\theta_{\mathrm{human}})$，则需要改进优化算法，使算法更加收敛；若$\mathrm{J}(\theta_{\mathrm{RL}}) < \mathrm{J}(\theta_{\mathrm{human}})$，则问题出在损失函数的设计上，这时需要改进损失函数形式。

误差分析

以上的 ML 算法诊断在算法没有问题的时候也可以使用，因为这有利于搞清楚问题的本质，使你能更好的阐明问题与论点，从而写出更加具有研究型的论文。

在诊断中，我们假设只有一个模型，但实际中，一个系统可能有多个部件组成。比如一个基于人脸的性别识别系统，可能由如下几个部件组成，按流程处理的先后顺序如下：

1) 数据采集
2) 数据预处理——背景消除
3) 人脸识别
4) 人脸器官识别——眼睛、鼻子、嘴
5) 将器官识别结果作为属性输入到分类器模型如 LR、SVM 中
6) 得到最后的分类结果

由上可知，从第 2 步到第 5 步是真正的算法处理过程。

所谓的误差分析，做法就是对每个部件用基准值代替算法的输出，然后观察最后结果的变化。对于上面的系统，可能的误差分析结果如下：

表 1 性别分类系统的误差分析

用基准值替代的部件	准确率
None(系统真实输出)	85%
背景消除	85.1%
人脸识别	91%
眼睛识别	94%
鼻子识别	95%
嘴识别	96%
分类结果	100%

需要注意的是，上表中用基准值替代的部件是累积的，比如当眼睛识别用基准值替代时，其上的背景消除与人脸识别仍然是用基准值替代。

经过这样的分析，我们可以的多啊，对于这个系统，提高较多的部件是人脸识别和眼睛识别。当然，器官识别中不同不同的基准值替代顺序可能会有不同的分析结果，所以要较为完备的进行比较，找到真正的瓶颈所在，避免过早优化问题。

销蚀分析

销蚀分析与误差分析不同，误差分析是一步步的用基准值替代算法输出，比较的是系统与最高性能的差别。而销蚀分析考虑的是与底线系统之间的性能差异。

比如，对于垃圾邮件分类器来说，先构建一个初始分类器，然后考虑一些比较高级的特征，比如邮件的语法风格、邮件的主机信息、邮件标题等。先将所有特征全加入到分类器中，然后逐个剔除，观察性能的下降幅度，将那些使性能没有下降或者下降很少的特征去除掉，找到真正有用的特征。

当然，与误差分析一样，该方法可能对顺序敏感，所以需要多次试验才行。

如何开始 ML 问题的解决

在 ML 上开始研究一般有两种做法。

第一种是仔细构建法。对问题进行深入的分析，提取正确的特征，收集需要的徐局，设计正确的算法结构。这样往往能一次就得到较好的具有可扩展性的算法。

第二种是创建-修改法。这种做法一般是先建立一个较差的系统，然后利用上面的诊断

法对系统进行修改，从而最终得到较好的系统。这种做法比较广泛应用，因为互联网上的成功产品，往往不是最好的，而是最早的。

如果在 ML 上开始做解决方案的话，一个较好的建议是先对数据进行分析，比如为什么数据中的这些属性值是负的。当找出数据中的规律或者数据中的错误的时候，往往会发现系统性能差不是需要更复杂的算法，而是更强大的预处理。

在做研究的过程中，要把注意力集中到关键问题上，不要轻易的相信某些理论对算法有用而花大量的时间在那些理论上。比如如果要检测出图片中的物体的话，可能会有人说三维微分几何对这个问题会有用，但是在你确认这个的确有用之前，不要浪费精力在这个上面。

Ng 谈到自己的经验时说，一般来说，花在设计、诊断上的时间通常都有价值，一般来说，三分之一的时间花在诊断的设计上都是可以接受的。