浙江大学

本科生毕业设计
开题报告

学生姓名：陈威志

学生学号：3120000521

指导教师：吴健

年级与专业：计科 1202

所在学院：计算机科学与技术学院

一、题目：基于 SMO 算法 SVM 的试卷信息统计

二、指导教师对开题报告、外文翻译和中期报告的具体要求：

指导教师（签名）_____

年　　月　　日

毕业设计开题报告、外文翻译和中期报告考核

导师对开题报告、外文翻译和中期报告评语及成绩评定：

成绩比例	开题报告 占（20%）	外文翻译 占（10%）	中期报告 占（10%）
分 值			

导师签名_____

年　　月　　日

答辩小组对开题报告、外文翻译和中期报告评语及成绩评定：

成绩比例	开题报告 占（20%）	外文翻译 占（10%）	中期报告 占（10%）
分 值			

开题报告答辩小组负责人（签名）_____

年　　月　　日

-

目录

本科毕业设计开题报告

1. 项目背景

1.1 试卷的数字化管理对提高教学质量有着十分重要的作用。借助数字化试卷管理系统,可以有效分析学生对知识点的掌握情况。因此,设计一个数字化的试卷管理系统具有重要的应用价值。而试卷信息统计又是其中必不可少的一个步，在教师完成了阅卷之后，如何快速，准确地统计出试卷中的信息就显得至关重要。并且统计出的信息（试卷的难易度，正确率等）也会为后续的出题工作提供反馈性的信息，是试题库系统中重要的环节。关于试卷的导入，试题库的建立，国内以有人在这方面做过相关的工作，但是关于学生完成考试和教师批阅后的试卷信息统计和反馈，相关的工作并不是很多。

1.2 机器学习是从观测数据出发，寻找规律，对未来数据或无法直接观测的数据做出预测的学习方法。统计学和计算机能力是其重要基础。统计学习理论是由 Vapnik 建立的一套机器学习理论，而通过这套理论所引出的 SVM 对各个应用领域的发展都有极大的贡献。SVM[1]是监督学习[2]的一种，SVM 的目的是通过有限的样本信息，遵循结构风险最小化原则，找出一个超平面，将训练样本正确分类。目前广泛应用于医药学，图像识别，人脸检测，文本分类等领域。而 SVM 在试卷信息识别和统计上的应用缺不曾多见。

2. 目标和任务

项目目标：

建立试卷信息统计系统：用户将学生完成答题，教师批阅后的试卷扫描或拍照图片传入系统中。系统对图像进行预处理，分块，归一化等操作，将每道题的老师判卷结果(手写对勾√和错误×)交给训练好的 SVM 分类器进行识别，然后将信息汇总给出试卷的统计信息。

对 SVM 分类器进行测试，根据训练集（training set），调整参数，进行测试找到最优参数。与其他算法进行对比，分析性能，做出改进。

项目任务：

[1] 是一种监督式学习的方法，可广泛地应用于统计分类以及回归分析。

[2] 是一个机器学习中的方法，可以由训练资料中学到或建立一个模式（函数 / learning model），并依此模式推测新的实例。训练资料是由输入物件（通常是向量）和预期输出所组成。函数的输出可以是一个连续的值（称为回归分析），或是预测一个分类标签（称作分类）。

我的任务分为 3 个阶段：

1. 对图像进行处理，通过灰度化、二值化、去噪声、倾斜校正一系列预处理，得到利于识别的二值图像。然后利用投影法对所要提取的信息区域进行定位，在确定了待提取手写标识的位置后，通过分隔和归一化得到待辨识的手写对勾√或错误×。
2. 建立一个性能优秀的 SVM 分类器。首先通过搜集不同的手写对勾√或错误×，构成一定数量的训练集（900-1000 个实例），然后将图像矩阵转变为列向量作为 SVM 的特征向量。然后确定需要使用的核函数（如线性内核，RBF 内核，多项式内核等），分隔训练集，使用交叉验证的方式确定一个较优的分类器参数。最后确定我们试卷信息统计系统所使用的分类器。
3. 设计界面（可能是 app 或图形界面等形式，尚未确定），整合各个部分，使之成为一个完整的系统。并且对收集到的信息进行统计，如该份卷子的正确率，答题的变化曲线等。同时也要对系统进行测评，在待统计样本的数量，试卷的模糊程度，手写信息的规范程度不同的情况下，能否做到准确的统计，是否具有鲁棒性。与别的系统进行比较，分析性能的优劣所在，在什么地方可以有所改进。

3. 可行性分析

技术可行性：

图形预处理方面包括“试卷图像采集”“试卷图形预处理”“手写对勾√或错误×的识别”“手写信息的提取”等几个步骤，相关的技术已经较为成熟，也广泛地应用到了各个领域中，如医学，文字识别，图像识别等。：

- 试卷图像采集阶段可以用照相机或扫描仪获取到试卷信息，该过程简单易执行，同时也具有经济可行性。
- 试卷图形预处理阶段的技术包括灰度化，二值化，降噪，倾斜矫正等操作。在这方面已有不少相关的文献和项目可以提供有效可行的指导。只要查看相关的文献然后经过学习，理论联系实践就可以实际应用到项目中来。
- 手写对勾√或错误×的识别和提取所需要的技术包括定位，字符分隔，和归一化操作。在这方面的内容可以参考图像中手写字体的识别，汽车车牌照识别等相关文献。其中定位可以使用投影法。

手写字符识别方面需要使用 SMO 算法训练的 SVM 解决，大致分为以下阶段：特征向量的提取，SVM 训练，输出结果的反馈，统计及存储：

- 特征向量的提取目前一个可行的做法是将 2 维的图片信息二值化（即化为 0,1 矩阵），然后将其转化为列向量作为我们的特征向量。考虑到我们试卷当中的手写对勾√或错误×标记不会很大（包含的像素点很少），所以即使将其转化为列向量也不会产生数值困难，其次在解决 SVM 的 QP（二次规划）问题中，我们所使用的仅仅是两对向量的内积，而不需要将向量保存在内存中，所以初步考虑这样做或许是可行的。

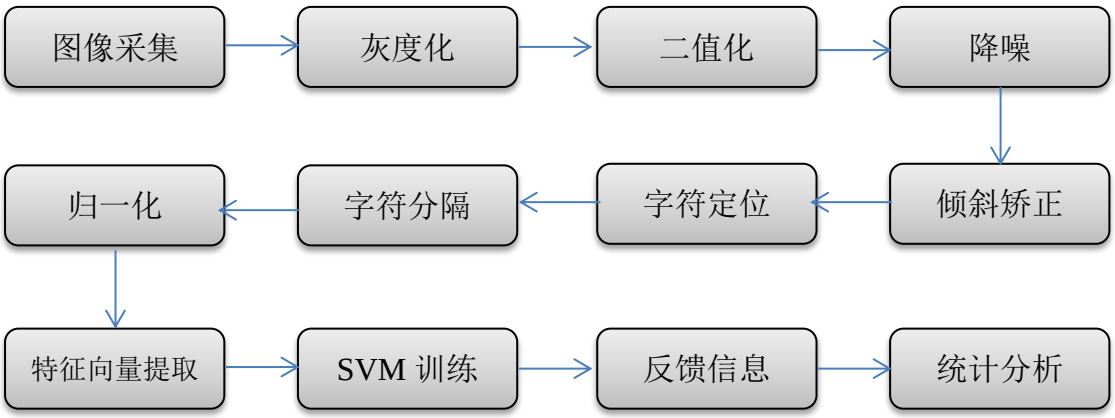
- 具前文所述 SVM 已经广泛地应用到了生活的各个方面，是一套非常完整的机器学习理论，虽然鲜有应用在试卷识别方向，但是可以参考 SVM 在图像识别，医学方面的相关文献。其次 SMO(序列最小优化算法)的发表在 SVM 界引起了轰动，先前的 SVM 算法必须求解复杂的二次规划问题，并且使用昂贵的二次规划问题工具。但是 SMO 算法较好地避免了这一问题。目前 SMO 算法被广泛应用于 SVM 训练中，并在通行的 SVM 库 libsvm 中得到实现。在降低算法复杂度的情况下，也提升了经济可行性。
- 最后对于输出结果的反馈，统计和存储就显得相对简单，初步来看不涉及太多的技术难点。随着批量处理的数量增加，可能会带来信息存储，及有效统计的问题。这方面也可以参考已有的大规模试题库系统的相关文献。所以在技术层面还是可行的。

硬件可行性：

我们不需要存储整份的试卷信息，我们只是需要对试卷图像进行处理，然后提取相关的待识别的手写信息图像。假设我们的待识别的图片是以 256 色的 BMP 格式保存到，如果一个像素所占的位数是 1 个字节，所以 100*100 的图片需要 10KB 的空间存储。每份试卷有 50 道题目，1w 份试卷存储这些待识别字符的空间需要 4.7GB 的空间。但是这些信息只是需要暂时存储的，不需要永久性的保存，也不需要同时存在于内存中，而灰度化后的图片信息需要占用的颜色空间更少，需要的存储空间也就更少，所以在空间方面是可行的。

4. 初步技术方案和关键技术考虑

初步技术方案及项目流程：



图形提取及预处理技术简单分析：

4.1 灰度化：我们所采集的图片是 256 色的 BMP 格式保存，但是彩色图像不利于图像的处理，因此需要将彩色图像转换为灰度图进行处理。灰度图是指只包含亮度信息，不包含色彩信息的图像。亮度信息量化为 0-255 共 256 个级别，0 表示最暗，即黑色，255 表示最亮，即白色。此时每个像素的 RGB 值都是相同的，即 RGB 值从(0，0，0)，(1，1，1)一直 N(255，255，255)。

图像灰度处理的方法有：最大值法、平均值法和加权平均值法。一般常用加权平均值法对彩色图像进行转换，如下式：

$$灰度值 = 0.299R + 0.587G + 0.114B$$

其中 R 代表像素中红色分量，G 代表像素中绿色分量，B 代表像素中蓝色分量。在实际操作中考虑到我们最关心的是教师阅卷的红色部分，可以在 R 分量上设置较大的参数。

4.2 二值化：为了从包括了背景，字符还有噪声的试卷图像中取出目标物体，需要对图像进行二值化处理。二值化就是通过设定某一阈值 T，用 T 将图像的数据分成两大部分：大于 T 的像素群和小于 T 的像素群。

二值化的关键在于阈值 T 的选取，选择阈值的方法有很多：整体阈值如 Ostu 算法，局部阈值如 Bersen 算法。具体使用的算法需要在后续的工作中进行测试然后决定。

4.3 降噪：图像在形成、传输、接收和处理的过程中不可避免存在着外部干扰和内部干扰，如光电转换过程中敏感元件灵敏度的不均匀性、数字化过程的量化噪声、传输过程中的误差及人为因素等，均会存在着一定程度的噪声干扰。这些噪声一定程度上会对图像的识别产生影响，因此去除噪声也是图像处理中比较常见的预处理手段。消除图像噪声的工作称之为图像平滑或滤波。比较常用的有中值滤波、均值滤波。但是对于点、线等细节较多的图像，不适宜采用中值滤波的算法，因为在滤波的过程中很可能会将字符本身的像素也同时去掉。

这里我们初步打算使用是二值图像的黑白点噪声滤波。

4.4 倾斜矫正：由于人为操作等因素，在图像采集时可能造成图像的倾斜，如果产生倾斜会在后面的投影操作中一定程度上影响题目编号和位置的判断，所以需要对倾斜的图像进行矫正。

我们初步决定使用改进的 Hough 变化进行倾斜的矫正。具体方法在这里不做过多说明。

4.5 字符的分隔和提取，我们初步决定使用投影法，即提取出我们关心的手写对勾√或错误×字符。为此，我们可以先进行水平投影，在 x 轴中自左至右进行扫描遇到第一个黑像素点进行记录，这样可以记录出第一个字符的宽度，然后仅针对这个宽度内的字符进行 y 轴投影，接着在 y 轴扫描找出的黑色像素区域就是字符的高度，综合两个坐标轴反馈的信息，我们可以得出第一个字符的所在区域，进而进行提取。

4.6 由于手写字符的随意性与不确定性，采集得到的图像字符大小可能存在较大的差异，这样不利于字符的识别，因此为了提高字符识别的准确性和准确率，可以对字符进行归一化处理，即将原本各不相同的字符统一到同一尺寸。归一化的具体参数将根据后续的测试和反馈的结果进行调整。

基于 SMO 算法训练的 SVM 分类器：

4.6 SVM: 所谓 SVM 就是对于已知的训练集 T = {(x1,y1),⋯(xL ,yL)}∈ (Rn×Y)L 找出一个最优超平面(w • x)+b= 0，将上述训练集进行分类,并对新的输入，判断其属于哪类。其中 Rn 表示 n 维欧氏空间 Rn 上的分类问题。这里，我们只需用到 2 维欧氏空间，因此 n= 2，xi ∈ Rn 是输入，yi ∈Y = {1,-1}是输出，i= 1,2,⋯,l。

要找到最优超平面，也就是求两条支持直线之间的距离最大，即

$$MAX = \frac{2}{\|w\|}$$

为了方便推导，将其改为

$$\text{Min} = \frac{1}{2}\|w\|^2, y_i(w^T x_i + b) \geq 1 \quad i = 1, \ldots, n$$

对于线性可分问题，可直接用上述理论处理；对线性不可分问题，引入松弛变量 ξ 和惩罚参数 C。上式改为：

$$\text{Min} = （\frac{1}{2}\|w\|^2 + \text{C}\sum_{i=1}^{n}\xi_i）, y_i(w^T x_i + b) \geq 1 - \xi \quad i = 1, \dots, n$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

为了更方便地求解上式，引进 Lagrange 函数，并将原始问题改为 Dual Problem（对偶问题），即

$$\text{Min} = \left(\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j (x_i \cdot x_j)\alpha_i \alpha_j - \sum_{j=1}^{l}\alpha_j\right), \sum_{i=1}^{l} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, i = 1, \dots, n$$

这样，只需求出 Lagrange 乘子向量α即可。

4.7 SMO 算法应用：求解上式比较困难，需要昂贵的第三方二次规划工具。于是，产生了序列最小最优化算法(Sequential minimal optimization，SMO)，该算法能够便捷地求解这个凸二次规划问题。该算法由 Microsoft Research 的 JohnCPlatt 在 1998 年提出，并成为最快的二次规划优化算法，特别针对线性 SVM，并且数据稀疏时性能更优。它将工作集的规模减到最小，因而在算法中不需要求解凸二次规划问题的迭代过程，只用解析求解两个变量的最优化问题即可，其代价是增加其迭代次数。

每一次迭代选出两个分量α_i和α_j进行调整，其他分量则保持固定不变，通常将他们视为常数。通过对两个变量求解最优化问题，得到α_i^*和α_j^*的解，然后利用α_i^*和α_j^*来改进对应向量α的分量以及 b。

最后得到决策函数

$$f(x) = sgn\big(g(x)\big) = sgn(\sum_{i=1}^{l} y_i \alpha_i^* \big(x_i \cdot x_j\big) + b^*)$$

5. 预期工作结果

实现一个能够智能化识别教师批改痕迹的试卷信息统计系统。在系统中，能够快速统计试卷的批改信息，并且能够收集多张试卷的统计信息（如正确率等），供试卷的管理人员或审计人员进行处理。

6. 进度计划

3 月 30 号：查找相关文献 8-10 篇，学习相关技术，翻译其中一篇文献。构思项目的实施流程，撰写开题报告。

4 月 10 号：继续学习相关知识，简单了解图像处理内容，深度学习机器学习理论，打好机器学习的基础（可以参考 Stanford 机器学习公开课），掌握 SVM 的基础理论。

学习 SMO 算法。

4 月 20 号：寻找（900-1000）个对勾√或错误×的手写样本建立训练集，导入成特征向量进行训练，对 1000 个训练集进行划分使用交叉验证和网格搜索的方式确定最优的参数。最终确定所使用的 SVM 分类器。

4 月 30 号：统合整个项目，对由图像预处理得到的测试集进行测试，修改图片归一化，SVM 惩罚项 C 等参数。对 SVM 分类器的性能和准确程度进行评估。

5 月 10 号：设计试卷信息统计系统，对批量处理的试卷反馈信息进行统计处理，给出统计信息，如某道题的正确率，某一份试卷的正确率，某份试卷的答题曲线等。然后进行测试，根据结果对系统做出改善。

5 月 20 号：完成毕业论文初稿

5 月 30 号：准备论文答辩

* 注：目录供参考

本科毕业设计外文翻译

支持向量机(SVM)实践手册

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin
Department of Computer Science
National Taiwan University, Taipei 106, Taiwan
http://www.csie.ntu.edu.tw/~cjlin
Initial version: 2003 Last updated: April 15, 2010

摘要

支持向量机（SVM）是一种流行的分类方法。然而，那些不熟悉 SVM 的初学者常常会因为忽略一些简单而重要的步骤而得不到满意的结果。在这本手册中，我们给出了一个简单的过程，通常会给出合理的结果。

1. 介绍

SVM（支持向量机）是数据分类的有用的技术。虽然 SVM 被认为是比神经网络更容易的算法，不熟悉使用的用户通常在第一次不会得到令人满意的结果。在这里，我们描述一个“如同食谱一样”的方法，这通常给出合理的结果。

注意到本指南不是提供给 SVM 研究者我们也不会保证你会达到最高的准确性。此外，我们也不打算解决具有挑战性的或困难的问题。我们的目的是给新手快速获取可接受结果的指导。

尽管用户不需要熟悉 SVM 背后的理论，我们也简单地介绍 SVM 所必须的基本原理。一个分类要求经常包括分隔数据到训练和测试集中。在训练集中的每个实例都包含一个“目标值”（比如类别标签）和多个“属性”（比如观察变量的特征值）。SVM 的目标就是引入一个模型（基于训练集）能基于测试集中的测试数据的属性预测目标值。

给定一个训练集中的实例下标对(x_i, y_i),i = 1⋯l， $x_i \in R^n$ 且 $y \in \{-1,1\}^l$, SVM(Boser et al., 1992; Cortes and Vapnik, 1995)需要求解以下问题的最优解:

$$\min_{w,b,\varepsilon} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

$$\text{Subject to } y_i(w^T \varphi(x_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0.$$

这里的训练向量x_i是被函数φ映射到高维空间中的。SVM 找到一个线性分隔的超平面并且在这个高维空间中具有最大间隔（margin）。C > 0 是错误项的惩罚参数。此外，$K(x_i, x_j) \equiv \varphi(x_i)^T \varphi(x_j)$被称为核函数。尽管有新的核函数被研究员提出，初学者需要

从 SVM 书籍中找到以下 4 种简单的核函数

- 线性：$K(x_i, x_j) = x_i{}^T x_j$.
- 多项式：$K(x_i, x_j) = (\gamma x_i{}^T x_j + \mathrm{r})^d, \gamma > 0$.
- 径向基函数神经（RBF）：$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0$.
- Sigmoid 函数：$K(x_i, x_j) = \tanh(\gamma x_i{}^T x_j + r)$.

在这里γ, r, d都是核参数

Applications	#training data	#testing data	#features	#classes	Accuracy by users	Accuracy by our procedure
Astroparticle[1]	3,089	4,000	4	2	75.2%	96.9%
Bioinformatics[2]	391	0[4]	20	3	36%	85.2%
Vehicle[3]	1,243	41	21	2	4.88%	87.8%

表 1：问题的特点和性能比较

1.1 实际生活中的例子

表 1 给出一些真实世界的例子。这些数据集是由我们的用户提供的,这些用户不能再开始的时候得到合理的准确度。详细信息在附录 A。

这些数据集是在 http://www.csie.ntu.edu.tw/~cjlin/papers/guide/data/

1.2 提出的方法

许多初学者现在使用如下方法:

- 以一个 SVM 包的格式变换数据
- 随机尝试一个新核函数和参数
- 测试

我们建议初学者首先尝试如下方法：

- 以一个 SVM 包的格式变换数据
- 对数据进行简单的缩放
- 考虑 RBF 核函数$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0$.
- 使用交叉验证，以找到最佳参数γ，C
- 用最优的参数γ，C 去训练整个训练集 [5]

1Courtesy of Jan Conrad from Uppsala University, Sweden.

2Courtesy of Cory Spencer from Simon Fraser University, Canada (Gardy et al., 2003).

3Courtesy of a user from Germany.

4As there are no testing data, cross-validation instead of testing accuracy is presented here.Details of cross-validation are in Section 3.2.

5最优参数可能受训练集大小影响，但是在实践中经过交叉验证以适应于整个训练集

- 测试

我们将在下面的章节中仔细讨论这个方法。

7. 2 数据预处理

2.1 分类特征

SVM 要求每个数据实例被表示为实数的一个向量。因此，如果有分类属性，我们首先必须将它们转换成数字数据。我们建议使用 m 个代表一个 m-类别属性。
m 个数中只有一个为 1，剩余为 0。例如，一个三类别属性如{红，绿，蓝}可以表示为（0,0,1），（0,1,0），和（1,0,0）。我们的经验表明，如果值的属性的数目不是太大，该编码可能比使用单数更稳定。

2.2 缩放

应用 SVM 之前缩放是非常重要的。Sarle 神经网络的第 2 部分常见问题 Sarle（1997）解释了这一点的重要性，大部分的考虑也适用支持向量机。缩放的主要优点是避免在更大的数字属性范围支配那些较小的数值范围。另一个优点是避免
在计算过程中的数值的困难。由于核函数的值通常依赖于特征向量的内积，例如线性内核和多项式内核，大的属性值可能导致数值问题。我们建议线性地缩放每个属性到范围[-1，+1]或[0，1]。

当然，我们必须使用同样的方法来缩放训练和测试数据。例如，假设我们缩放训练数据的第一属性从[-10，+10]至[-1，+1]。如果测试数据的第一属性在于在范围[-11，+ 8]，我们必须调整测试数据为[-1.1，+0.8]。请参阅附录 B 为一些实际的例子。

8. 3 模型选择

虽然如第 1 节所述只有四个常用内核，我们必须决定哪一个先试。然后选择惩罚参数 C 和内核参。

3.1 RBF 内核

在一般情况下，RBF 核是一个合理的第一选择。这个内核非线性地映射样本进入一个更高维空间，因此不同于线性内核，可以处理那些分类标签和属性不是线性关系（线性不可分）的情形。此外线性内核是 RBF Keerthi 和 Lin（2003）的一种特殊情形，因为有惩罚项 C 的线性内核与（γ，C）的 RBF 内核有一样的性能。此外具有一定参数的 sigmoid 内核可以展现和 RBF 相似的性能(Lin and Lin, 2003)。

第二个原因是影响复杂超参数的数目选型。多项式内核比 RBF 核有更多的超参数。

最后，RBF 内核具有较少的数值困难。一个关键点是$0 < K_{i,j} \leq 1$,相反多项式内核可能达到无穷$\gamma x_i{}^T x_j + \mathrm{r} > 1$或者接近于 0，$\gamma x_i{}^T x_j + \mathrm{r} < 1$当度数很大的时候。此外，我们必须注意的是，sigmoid 内核不是有效的（即，不是两个矢量的内积）根据一些参数（Vapnik，1995）。

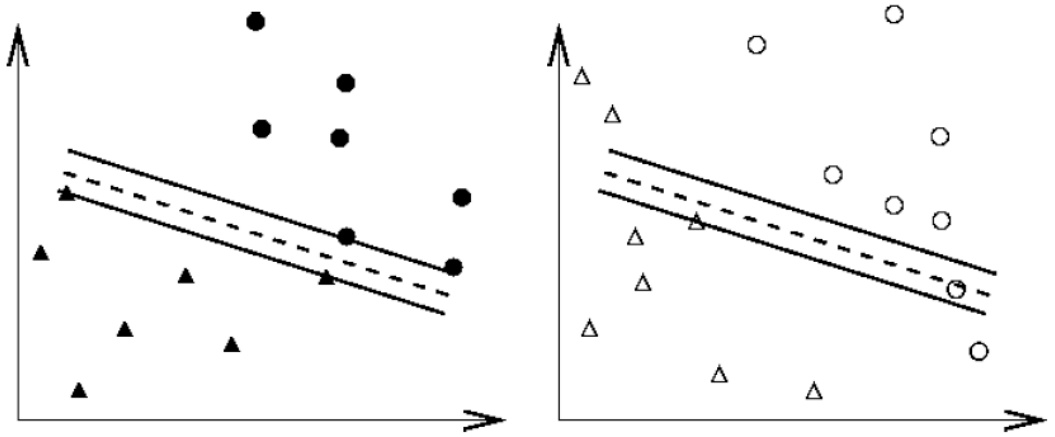
有些情况下 RBF 核是不合适的。尤其是当的特征的数量是非常大的，可以只使用线性核。我们在附录 C 讨论细节。

3.2 交叉验证和网格搜索

在 RBF 核中有两个参数：γ，C。我们事先是不知道哪个γ，C 最适合给定问题；因此必须做到某种模型选择（参数搜索）。目标是觉得一个好的（γ，C）使得分类器可以正确预测未知数据（即，测试集）。注意到可能对实现高精度训练不是很有帮助（即，一个准确预测训练集下标的分类器仍是未知的）。如上所述，一个通常的策略是将数据集分割成两部分，其中的一个被认为是未知的。如上所述，一个通常的策略是将数据集分割成两部分，其中的一个被认为是未知。从“未知”获得的预测精度得到更精确反映了分类独立数据集的性能。此过程的改进版被称为交叉验证。

在一个 v-折叠交叉验证中，我们首先将训练集分成 K 个相同大小的子集。接着一个子集作为测试集用于测试用 v-1 个子集作为训练集的分类器。因此，训练集中的每个实例都被验证了一遍，所以交叉验证的准确性就实例被正确分类的比例。

交叉验证可以防止过拟合问题。图 1 用一个二分类表现这个问题。实心圆和三角形是训练数据，而空心圆和三角形的测试数据。在图 1a 和 1b 中的分类器不好，因为过拟合了训练数据。如果我们把图 1a 和 1b 的训练和测试数据作为训练和交叉验证集，那么准确性是不好的。另一方面，在 1c 和 1d 中的分类器无过拟合问题且给出了更好的交叉验证和准确性。



(a) Training data and an overfitting classifier　(b) Applying an overfitting classifier on testing data

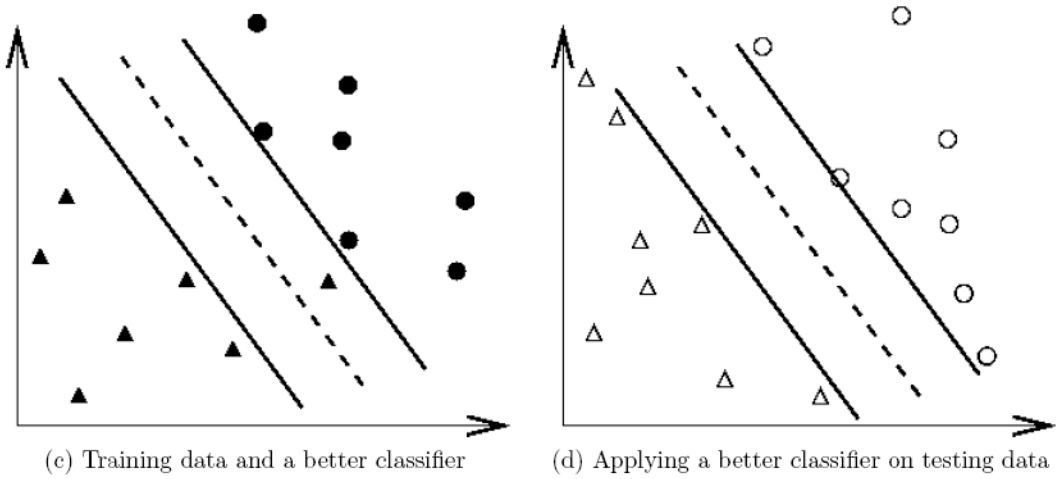


图 1：一个过拟合的分类器和一个较好的分类器

我们推荐一个“网格搜索”对于γ，C 使用交叉验证。多对（γ，C）值进行测试，然后选取交叉验证最准确的项。我们发现指数增长的尝试γ，C 序列，是一个好的实践方法。（例如，$C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$）。

网格搜索是简单的，但似乎朴素。实际上，有几个先进的方法可节省计算成本，例如，近似交叉验证率。然而有两个动机使我们使用简单的网格搜索。

其一，在心理上，我们不放心使用近似参数化简复杂的计算。其二，网格搜索找最优解的时间不比其他更先进算法慢，因为只有两个参数。此外，网格搜索可以被轻松地并行化，因为每个（γ，C）都是相互独立的。许多先进的方法都是迭代过程的，如沿某条路线行走，这是很难被并行化的。

因为这样一个完整的网格搜索仍可能费时，所以我们建议一上来先使用一个粗糙的网格。在确定一个更“好”的网格区域后，再设计一个更好的网格搜索。为了说明这一点，我们对于 Statlog collection（http://www.ncc.up.pt/liacc/ML/statlog/datasets.html）(Michie et al., 1994).的 german 问题做一个实验。在缩放了这个集合后，我们首先使用一个粗糙的网格（图 2）然后找到最优的（C，γ）是（2^3，2^{-5}）有着 77.5%的交叉验证率。接下我们设计一个更优的网格搜索在（2^3，2^{-5}）的附近（图 3）然后发现一个更好的交叉验证率 77.6%的（$2^{3.25}$，$2^{-5.25}$）。在最优的（C，γ）被找到后，整个训练集再一次被训练，生成一个更好的分类器。

上述方法非常适用于具有数千或更多的数据点的问题。对于非常大的数据集，一个可行的办法是随机选择的一个子集，然后设计网格搜索，然后对整个的数据集仅对更好的区域进行网格搜索。

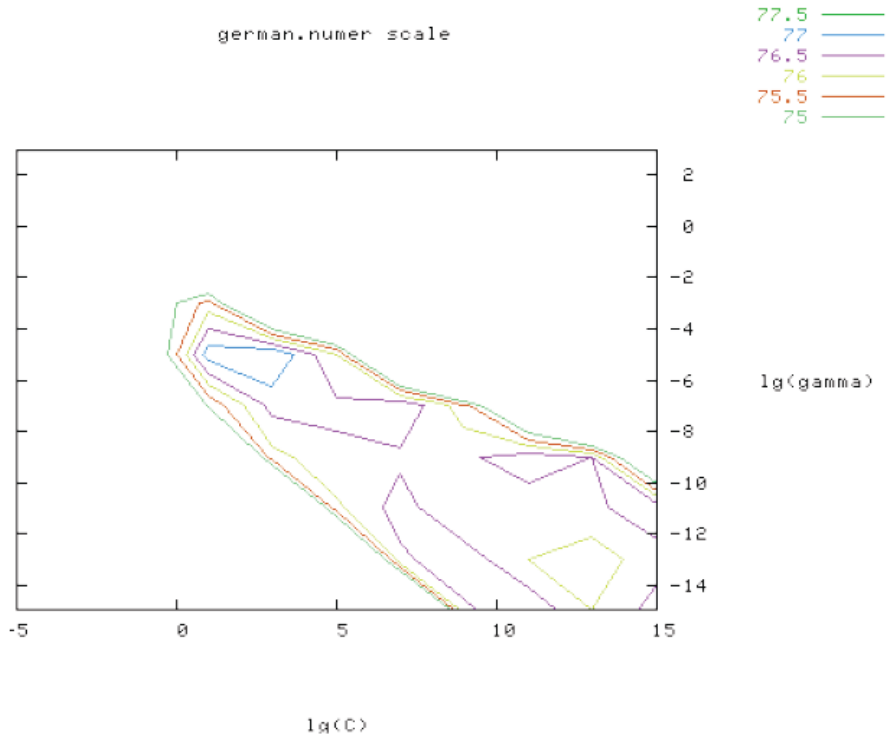


图 2：宽松网格搜索在$C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$

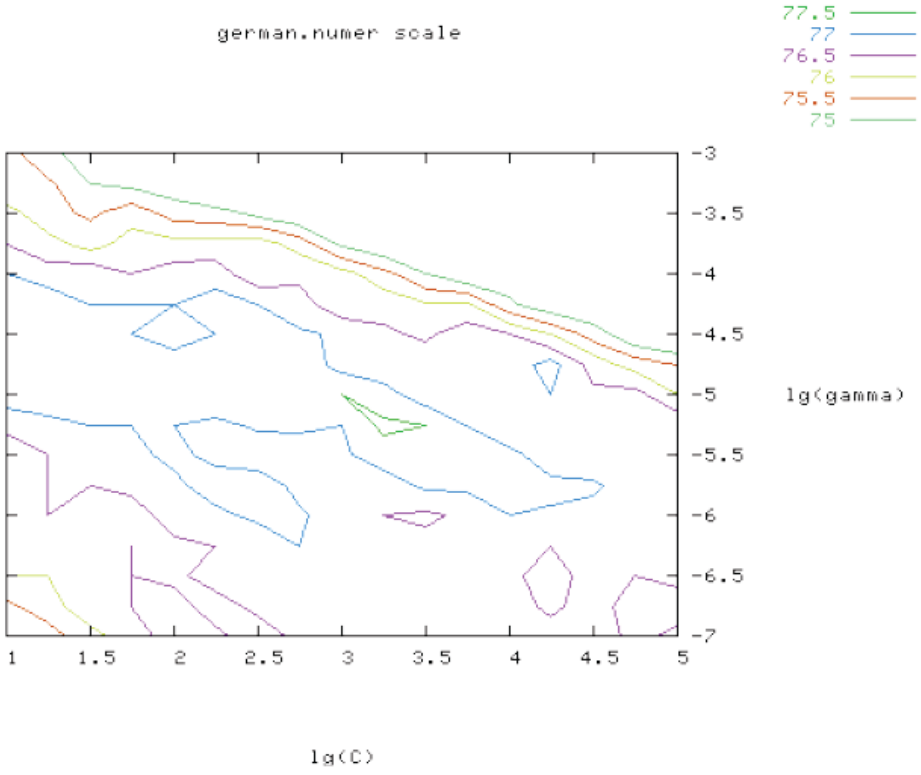


图 3：优化网格搜索在$C = 2^{1}, 2^{1.25}, \ldots, 2^{5}, \gamma = 2^{-7}, 2^{-6.75}, \ldots, 2^{-3}$

9. 4 讨论

在某些情况下，上述建议的程序是不够好，所以其他技术如特征选择可能是必要的。这些问题超出范围本指南。我们的经验表明，这种方法非常适用于特征不多的数据。如果有成千上万的属性，可能有必要选取特征中的一个子集交给 SVM。

10. 答谢

我们感谢那些使用我们 SVM 软件 LIBSVM 和 BSVM 的用户，他们帮助我们指出了初学者可能遇到的困难。我们也感谢校验这篇论文的用户（特别地，Robert Campbell）。

11. 附录

A 指出方法的一个实例(略)

B 缩放训练和测试集的常见问题(略)

C 什么时候使用线性内核而不是 RBF 内核(略)

12. 参考文献

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pages 144–152. ACM Press, 1992.

C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.C. Cortes and V. Vapnik. Support-vector network. Machine Learning, 20:273–297,1995.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9:1871–1874, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf.

J. L. Gardy, C. Spencer, K. Wang, M. Ester, G. E. Tusnady, I. Simon, S. Hua,K. deFays, C. Lambert, K. Nakai, and F. S. Brinkman. PSORT-B: improving protein subcellular localization prediction for gram-negative bacteria. Nucleic Acids Research, 31(13):3613–3617, 2003.

S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation, 15(7):1667–1689, 2003.

H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, Department of ComputerScience, National Taiwan University, 2003. URL http://www.csie.ntu.edu.tw/
~cjlin/papers/tanh.pdf.

D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, editors. Machinelearning, neural and statistical classification. Ellis Horwood, Upper Saddle River,NJ, USA, 1994. ISBN 0-13-106360-X. Data available at http://archive.ics.
uci.edu/ml/machine-learning-databases/statlog/.

W. S. Sarle. Neural Network FAQ, 1997. URL ftp://ftp.sas.com/pub/neural/FAQ.html. Periodic posting to the Usenet newsgroup comp.ai.neural-nets.

V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York,NY, 1995.

A Practical Guide to Support Vector Classification

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin

Department of Computer Science

National Taiwan University, Taipei 106, Taiwan

http://www.csie.ntu.edu.tw/~cjlin

Initial version: 2003 Last updated: April 15, 2010

Abstract

The support vector machine (SVM) is a popular classification technique. However, beginners who are not familiar with SVM often get unsatisfactory results since they miss some easy but significant steps. In this guide, we propose a simple procedure which usually gives reasonable results.

1 Introduction

SVMs (Support Vector Machines) are a useful technique for data classification. Although SVM is considered easier to use than Neural Networks, users not familiar with it often get unsatisfactory results at first. Here we outline a "cookbook" approach which usually gives reasonable results.

Note that this guide is not for SVM researchers nor do we guarantee you will achieve the highest accuracy. Also, we do not intend to solve challenging or difficult problems. Our purpose is to give SVM novices a recipe for rapidly obtaining acceptable results.

Although users do not need to understand the underlying theory behind SVM, we briefly introduce the basics necessary for explaining our procedure. A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one "target value" (i.e. the class labels) and several "attributes" (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

Given a training set of instance-label pairs $(\mathbf{x}_i, y_i), i = 1, \ldots, l$ where $\mathbf{x}_i \in R^n$ and $\mathbf{y} \in \{1, -1\}^l$, the support vector machines (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) require the solution of the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{1}$$

Table 1: Problem characteristics and performance comparisons.

Applications	#training data	#testing data	#features	#classes	Accuracy by users	Accuracy by our procedure
Astroparticle[1]	3,089	4,000	4	2	75.2%	96.9%
Bioinformatics[2]	391	0[4]	20	3	36%	85.2%
Vehicle[3]	1,243	41	21	2	4.88%	87.8%

Here training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function ϕ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is called the kernel function. Though new kernels are being proposed by researchers, beginners may find in SVM books the following four basic kernels:

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r)^d$, $\gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r)$.

Here, γ, r, and d are kernel parameters.

1.1 Real-World Examples

Table 1 presents some real-world examples. These data sets are supplied by our users who could not obtain reasonable accuracy in the beginning. Using the procedure illustrated in this guide, we help them to achieve better performance. Details are in Appendix A.

These data sets are at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/data/`

[1] Courtesy of Jan Conrad from Uppsala University, Sweden.

[2] Courtesy of Cory Spencer from Simon Fraser University, Canada (Gardy et al., 2003).

[3] Courtesy of a user from Germany.

[4] As there are no testing data, cross-validation instead of testing accuracy is presented here. Details of cross-validation are in Section 3.2.

1.2 Proposed Procedure

Many beginners use the following procedure now:

- Transform data to the format of an SVM package
- Randomly try a few kernels and parameters
- Test

We propose that beginners try the following procedure first:

- Transform data to the format of an SVM package
- Conduct simple scaling on the data
- Consider the RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$
- Use cross-validation to find the best parameter C and γ
- Use the best parameter C and γ to train the whole training set[5]
- Test

We discuss this procedure in detail in the following sections.

2 Data Preprocessing

2.1 Categorical Feature

SVM requires that each data instance is represented as a vector of real numbers. Hence, if there are categorical attributes, we first have to convert them into numeric data. We recommend using m numbers to represent an m-category attribute. Only one of the m numbers is one, and others are zero. For example, a three-category attribute such as {red, green, blue} can be represented as (0,0,1), (0,1,0), and (1,0,0). Our experience indicates that if the number of values in an attribute is not too large, this coding might be more stable than using a single number.

[5]The best parameter might be affected by the size of data set but in practice the one obtained from cross-validation is already suitable for the whole training set.

2.2 Scaling

Scaling before applying SVM is very important. Part 2 of Sarle's Neural Networks FAQ Sarle (1997) explains the importance of this and most of considerations also apply to SVM. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. We recommend linearly scaling each attribute to the range $[-1, +1]$ or $[0, 1]$.

Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from $[-10, +10]$ to $[-1, +1]$. If the first attribute of testing data lies in the range $[-11, +8]$, we must scale the testing data to $[-1.1, +0.8]$. See Appendix B for some real examples.

3 Model Selection

Though there are only four common kernels mentioned in Section 1, we must decide which one to try first. Then the penalty parameter C and kernel parameters are chosen.

3.1 RBF Kernel

In general, the RBF kernel is a reasonable first choice. This kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF Keerthi and Lin (2003) since the linear kernel with a penalty parameter $\tilde{C}$ has the same performance as the RBF kernel with some parameters (C, γ). In addition, the sigmoid kernel behaves like RBF for certain parameters (Lin and Lin, 2003).

The second reason is the number of hyperparameters which influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel.

Finally, the RBF kernel has fewer numerical difficulties. One key point is $0 < K_{ij} \leq 1$ in contrast to polynomial kernels of which kernel values may go to infinity $(\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r > 1)$ or zero $(\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r < 1)$ while the degree is large. Moreover, we must note that the sigmoid kernel is not valid (i.e. not the inner product of two

vectors) under some parameters (Vapnik, 1995).

There are some situations where the RBF kernel is not suitable. In particular, when the number of features is very large, one may just use the linear kernel. We discuss details in Appendix C.

3.2 Cross-validation and Grid-search

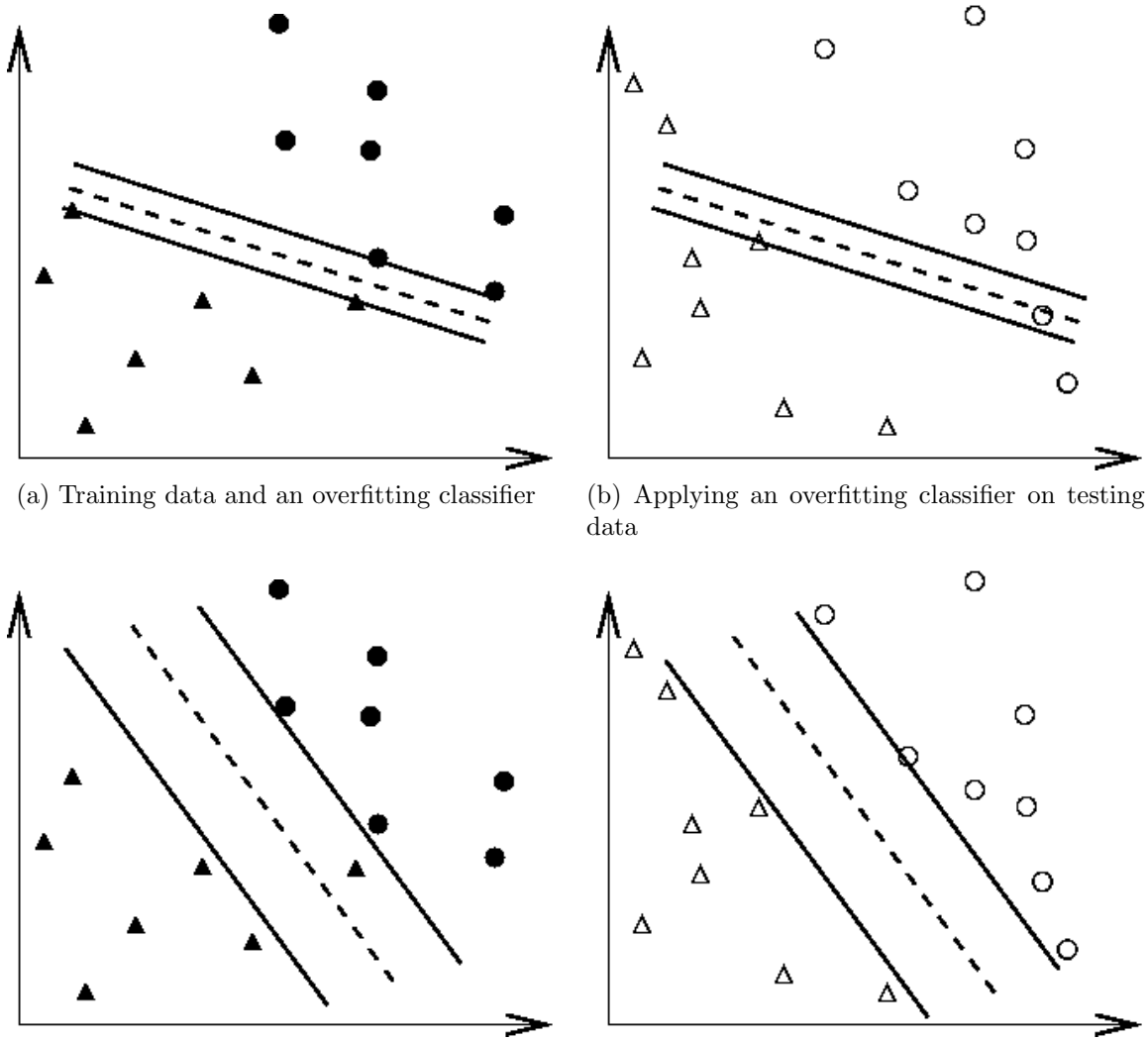
There are two parameters for an RBF kernel: C and γ. It is not known beforehand which C and γ are best for a given problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C, γ) so that the classifier can accurately predict unknown data (i.e. testing data). Note that it may not be useful to achieve high training accuracy (i.e. a classifier which accurately predicts training data whose class labels are indeed known). As discussed above, a common strategy is to separate the data set into two parts, of which one is considered unknown. The prediction accuracy obtained from the "unknown" set more precisely reflects the performance on classifying an independent data set. An improved version of this procedure is known as cross-validation.

In v-fold cross-validation, we first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining $v - 1$ subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified.

The cross-validation procedure can prevent the overfitting problem. Figure 1 represents a binary classification problem to illustrate this issue. Filled circles and triangles are the training data while hollow circles and triangles are the testing data. The testing accuracy of the classifier in Figures 1a and 1b is not good since it overfits the training data. If we think of the training and testing data in Figure 1a and 1b as the training and validation sets in cross-validation, the accuracy is not good. On the other hand, the classifier in 1c and 1d does not overfit the training data and gives better cross-validation as well as testing accuracy.

We recommend a "grid-search" on C and γ using cross-validation. Various pairs of (C, γ) values are tried and the one with the best cross-validation accuracy is picked. We found that trying exponentially growing sequences of C and γ is a practical method to identify good parameters (for example, $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$).

The grid-search is straightforward but seems naive. In fact, there are several advanced methods which can save computational cost by, for example, approximating the cross-validation rate. However, there are two motivations why we prefer the simple



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

Figure 1: An overfitting classifier and a better classifier (● and ▲: training data; ○ and △: testing data).

grid-search approach.

One is that, psychologically, we may not feel safe to use methods which avoid doing an exhaustive parameter search by approximations or heuristics. The other reason is that the computational time required to find good parameters by grid-search is not much more than that by advanced methods since there are only two parameters. Furthermore, the grid-search can be easily parallelized because each (C, γ) is independent. Many of advanced methods are iterative processes, e.g. walking along a path, which can be hard to parallelize.

Since doing a complete grid-search may still be time-consuming, we recommend

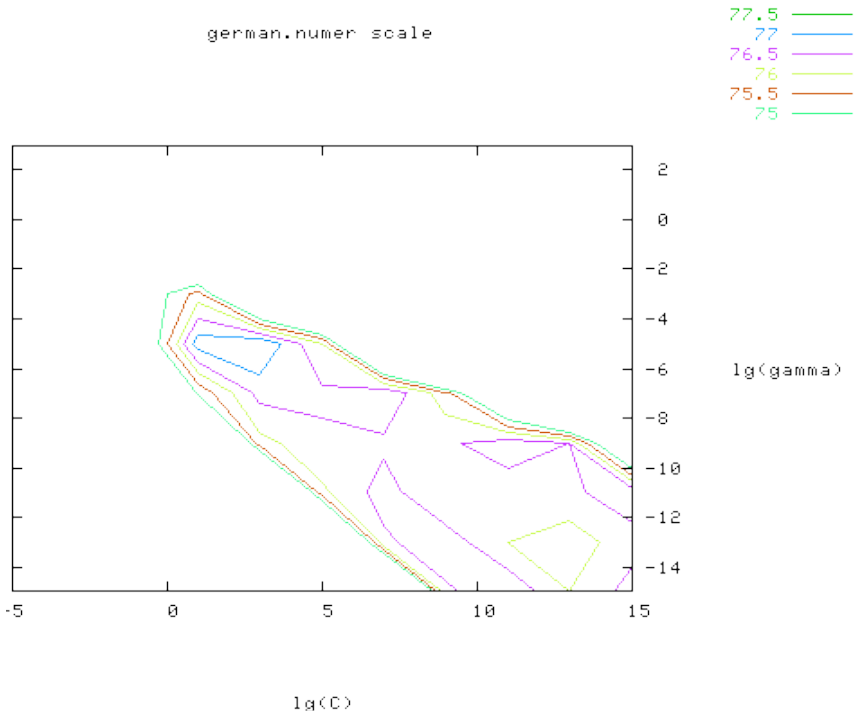


Figure 2: Loose grid search on $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$.

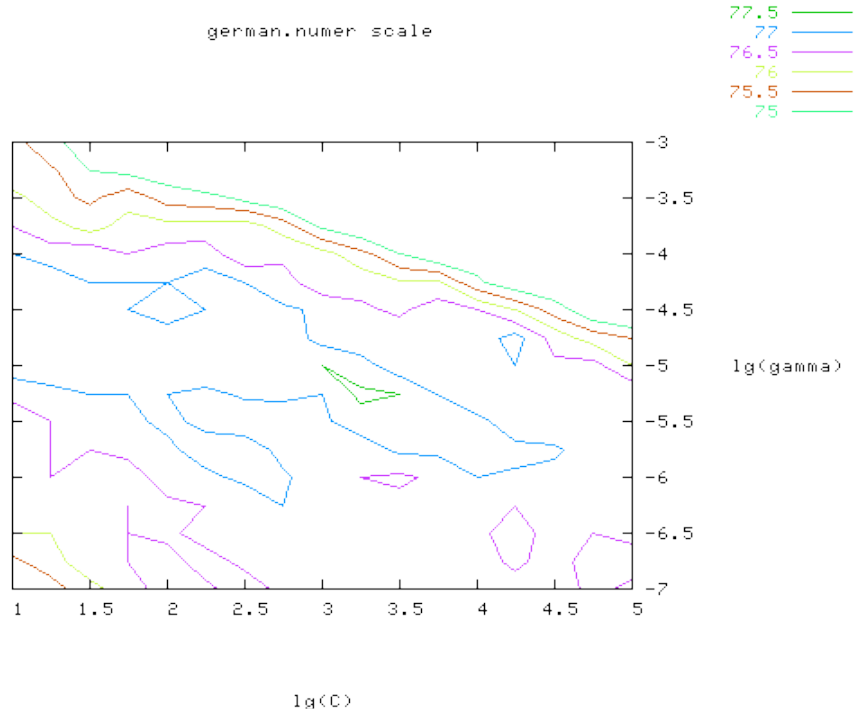


Figure 3: Fine grid-search on $C = 2^{1}, 2^{1.25}, \ldots, 2^{5}$ and $\gamma = 2^{-7}, 2^{-6.75}, \ldots, 2^{-3}$.

using a coarse grid first. After identifying a "better" region on the grid, a finer grid search on that region can be conducted. To illustrate this, we do an experiment on the problem `german` from the Statlog collection (Michie et al., 1994). After scaling this set, we first use a coarse grid (Figure 2) and find that the best (C, γ) is $(2^3, 2^{-5})$ with the cross-validation rate 77.5%. Next we conduct a finer grid search on the neighborhood of $(2^3, 2^{-5})$ (Figure 3) and obtain a better cross-validation rate 77.6% at $(2^{3.25}, 2^{-5.25})$. After the best (C, γ) is found, the whole training set is trained again to generate the final classifier.

The above approach works well for problems with thousands or more data points. For very large data sets a feasible approach is to randomly choose a subset of the data set, conduct grid-search on them, and then do a better-region-only grid-search on the complete data set.

4 Discussion

In some situations the above proposed procedure is not good enough, so other techniques such as feature selection may be needed. These issues are beyond the scope of this guide. Our experience indicates that the procedure works well for data which do not have many features. If there are thousands of attributes, there may be a need to choose a subset of them before giving the data to SVM.

Acknowledgments

We thank all users of our SVM software LIBSVM and BSVM, who helped us to identify possible difficulties encountered by beginners. We also thank some users (in particular, Robert Campbell) for proofreading the paper.

A Examples of the Proposed Procedure

In this appendix we compare accuracy by the proposed procedure with that often used by general beginners. Experiments are on the three problems mentioned in Table 1 by using the software LIBSVM (Chang and Lin, 2001). For each problem, we first list the accuracy by direct training and testing. Secondly, we show the difference in accuracy with and without scaling. From what has been discussed in Section 2.2, the range of training set attributes must be saved so that we are able to restore them while scaling the testing set. Thirdly, the accuracy by the proposed procedure

(scaling and then model selection) is presented. Finally, we demonstrate the use of a tool in LIBSVM which does the whole procedure automatically. Note that a similar parameter selection tool like the `grid.py` presented below is available in the R-LIBSVM interface (see the function `tune`).

A.1 Astroparticle Physics

- Original sets with default parameters

```
$ ./svm-train svmguide1
$ ./svm-predict svmguide1.t svmguide1.model svmguide1.t.predict
 → Accuracy = 66.925%
```

- Scaled sets with default parameters

```
$ ./svm-scale -l -1 -u 1 -s range1 svmguide1 > svmguide1.scale
$ ./svm-scale -r range1 svmguide1.t > svmguide1.t.scale
$ ./svm-train svmguide1.scale
$ ./svm-predict svmguide1.t.scale svmguide1.scale.model svmguide1.t.predict
 → Accuracy = 96.15%
```

- Scaled sets with parameter selection (change to the directory `tools`, which contains `grid.py`)

```
$ python grid.py svmguide1.scale
...
2.0 2.0 96.8922
```

(Best C=2.0, γ=2.0 with five-fold cross-validation rate=96.8922%)

```
$ ./svm-train -c 2 -g 2 svmguide1.scale
$ ./svm-predict svmguide1.t.scale svmguide1.scale.model svmguide1.t.predict
 → Accuracy = 96.875%
```

- Using an automatic script

```
$ python easy.py svmguide1 svmguide1.t
Scaling training data...
Cross validation...
```

```
Best c=2.0, g=2.0
Training...
Scaling testing data...
Testing...
Accuracy = 96.875% (3875/4000) (classification)
```

A.2 Bioinformatics

- Original sets with default parameters

```
$ ./svm-train -v 5 svmguide2
 → Cross Validation Accuracy = 56.5217%
```

- Scaled sets with default parameters

```
$ ./svm-scale -l -1 -u 1 svmguide2 > svmguide2.scale
$ ./svm-train -v 5 svmguide2.scale
 → Cross Validation Accuracy = 78.5166%
```

- Scaled sets with parameter selection

```
$ python grid.py svmguide2.scale
...
2.0 0.5 85.1662
 → Cross Validation Accuracy = 85.1662%
```

(Best C=2.0, γ=0.5 with five fold cross-validation rate=85.1662%)

- Using an automatic script

```
$ python easy.py svmguide2
Scaling training data...
Cross validation...
Best c=2.0, g=0.5
Training...
```

A.3 Vehicle

- Original sets with default parameters

```
$ ./svm-train svmguide3
$ ./svm-predict svmguide3.t svmguide3.model svmguide3.t.predict
 → Accuracy = 2.43902%
```

- Scaled sets with default parameters

```
$ ./svm-scale -l -1 -u 1 -s range3 svmguide3 > svmguide3.scale
$ ./svm-scale -r range3 svmguide3.t > svmguide3.t.scale
$ ./svm-train svmguide3.scale
$ ./svm-predict svmguide3.t.scale svmguide3.scale.model svmguide3.t.predict
 → Accuracy = 12.1951%
```

- Scaled sets with parameter selection

```
$ python grid.py svmguide3.scale
...
128.0 0.125 84.8753
```

(Best C=128.0, γ=0.125 with five-fold cross-validation rate=84.8753%)

```
$ ./svm-train -c 128 -g 0.125 svmguide3.scale
$ ./svm-predict svmguide3.t.scale svmguide3.scale.model svmguide3.t.predict
 → Accuracy = 87.8049%
```

- Using an automatic script

```
$ python easy.py svmguide3 svmguide3.t
Scaling training data...
Cross validation...
Best c=128.0, g=0.125
Training...
Scaling testing data...
Testing...
Accuracy = 87.8049% (36/41) (classification)
```

B Common Mistakes in Scaling Training and Testing Data

Section 2.2 stresses the importance of using the same scaling factors for training and testing sets. We give a real example on classifying traffic light signals (courtesy of an anonymous user) It is available at LIBSVM Data Sets.

If training and testing sets are separately scaled to $[0, 1]$, the resulting accuracy is lower than 70%.

```
$ ../svm-scale -l 0 svmguide4 > svmguide4.scale
$ ../svm-scale -l 0 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 69.2308% (216/312) (classification)
```

Using the same scaling factors for training and testing sets, we obtain much better accuracy.

```
$ ../svm-scale -l 0 -s range4 svmguide4 > svmguide4.scale
$ ../svm-scale -r range4 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 89.4231% (279/312) (classification)
```

With the correct setting, the 10 features in `svmguide4.t.scale` have the following maximal values:

0.7402, 0.4421, 0.6291, 0.8583, 0.5385, 0.7407, 0.3982, 1.0000, 0.8218, 0.9874

Clearly, the earlier way to scale the testing set to $[0, 1]$ generates an erroneous set.

C When to Use Linear but not RBF Kernel

If the number of features is large, one may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance. Using the linear kernel is good enough, and one only searches for the parameter C. While Section 3.1 describes that RBF is at least as good as linear, the statement is true only after searching the (C, γ) space.

Next, we split our discussion to three parts:

C.1 Number of instances $\ll$ number of features

Many microarray data in bioinformatics are of this type. We consider the Leukemia data from the LIBSVM data sets (`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`). The training and testing sets have 38 and 34 instances, respectively. The number of features is 7,129, much larger than the number of instances. We merge the two files and compare the cross validation accuracy of using the RBF and the linear kernels:

- RBF kernel with parameter selection

  ```
  $ cat leu leu.t > leu.combined
  $ python grid.py leu.combined
  ...
  8.0 3.0517578125e-05 97.2222
  ```

  (Best C=8.0, $\gamma = 0.000030518$ with five-fold cross-validation rate=97.2222%)

- Linear kernel with parameter selection

  ```
  $ python grid.py -log2c -1,2,1 -log2g 1,1,1 -t 0 leu.combined
   ...
  0.5 2.0 98.6111
  ```

  (Best C=0.5 with five-fold cross-validation rate=98.61111%)

  Though `grid.py` was designed for the RBF kernel, the above way checks various C using the linear kernel (`-log2g 1,1,1` sets a dummy γ).

The cross-validation accuracy of using the linear kernel is comparable to that of using the RBF kernel. Apparently, when the number of features is very large, one may not need to map the data.

In addition to LIBSVM, the LIBLINEAR software mentioned below is also effective for data in this case.

C.2 Both numbers of instances and features are large

Such data often occur in document classification. LIBSVM is not particularly good for this type of problems. Fortunately, we have another software LIBLINEAR (Fan et al., 2008), which is very suitable for such data. We illustrate the difference between

LIBSVM and LIBLINEAR using a document problem `rcv1_train.binary` from the LIBSVM data sets. The numbers of instances and features are 20,242 and 47,236, respectively.

```
$ time libsvm-2.85/svm-train -c 4 -t 0 -e 0.1 -m 800 -v 5 rcv1_train.binary
Cross Validation Accuracy = 96.8136%
345.569s
$ time liblinear-1.21/train -c 4 -e 0.1 -v 5 rcv1_train.binary
Cross Validation Accuracy = 97.0161%
2.944s
```

For five-fold cross validation, LIBSVM takes around 350 seconds, but LIBLINEAR uses only 3. Moreover, LIBSVM consumes more memory as we allocate some spaces to store recently used kernel elements (see `-m 800`). Clearly, LIBLINEAR is much faster than LIBSVM to obtain a model with comparable accuracy.

LIBLINEAR is efficient for large-scale document classification. Let us consider a large set `rcv1_test.binary` with 677,399 instances.

```
$ time liblinear-1.21/train -c 0.25 -v 5 rcv1_test.binary
Cross Validation Accuracy = 97.8538%
68.84s
```

Note that reading the data takes most of the time. The training of each training/validation split is *less than four seconds*.

C.3 Number of instances $\gg$ number of features

As the number of features is small, one often maps data to *higher dimensional spaces* (i.e., using nonlinear kernels). However, if you really would like to use the linear kernel, you may use LIBLINEAR with the option `-s 2`. When the number of features is small, it is often faster than the default `-s 1`. Consider the data `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/covtype.libsvm.binary.scale.bz2`. The number of instances 581,012 is much larger than the number of features 54. We run LIBLINEAR with `-s 1` (default) and `-s 2`.

```
$ time liblinear-1.21/train -c 4 -v 5 -s 2 covtype.libsvm.binary.scale
Cross Validation Accuracy = 75.67%
67.224s
$ time liblinear-1.21/train -c 4 -v 5 -s 1 covtype.libsvm.binary.scale
```

```
Cross Validation Accuracy = 75.6711%
452.736s
```

Clearly, using `-s 2` leads to shorter training time.

References

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

J. L. Gardy, C. Spencer, K. Wang, M. Ester, G. E. Tusnady, I. Simon, S. Hua, K. deFays, C. Lambert, K. Nakai, and F. S. Brinkman. PSORT-B: improving protein subcellular localization prediction for gram-negative bacteria. *Nucleic Acids Research*, 31(13):3613–3617, 2003.

S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, Department of Computer Science, National Taiwan University, 2003. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf`.

D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, editors. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994. ISBN 0-13-106360-X. Data available at `http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/`.

W. S. Sarle. Neural Network FAQ, 1997. URL `ftp://ftp.sas.com/pub/neural/FAQ.html`. Periodic posting to the Usenet newsgroup comp.ai.neural-nets.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1995.