CS229 Lecture notes

原作者：[Andrew Ng](http://cs229.stanford.edu/)（[吴恩达](http://open.163.com/movie/2008/1/M/C/M6SGF6VB4_M6SGHFBMC.html)）

翻译：[CycleUser](https://zhuanlan.zhihu.com/python-kivy)

Part V

支持向量机（Support Vector Machines）

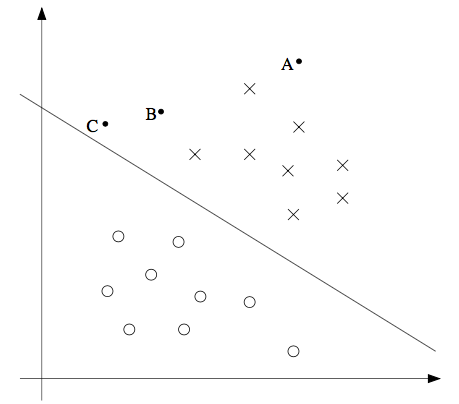
本章的讲义主要讲述的是 支持向量机( Support Vector Machine ，缩写为 SVM) 学习算法。SVM 算得上是现有的最好的现成的（“off-the-shelf”）监督学习算法之一，很多人实际上认为这里没有“之一”这两个字的必要，认为 SVM 就是最好的现成的监督学习算法。讲这个 SVM 的来龙去脉之前，我们需要先讲一些关于边界的内容，以及对数据进行分割成大的区块（gap）的思路。接下来，我们要讲一下最佳边界分割器（optimal margin classifier,），其中还会引入一些关于拉格朗日对偶（Lagrange duality）的内容。然后我们还会接触到核（Kernels），这提供了一种在非常高的维度（例如无穷维度）中进行 SVM 学习的高效率方法，最终本章结尾部分会讲 SMO 算法，也就是 SVM 算法的一个有效实例。

1 边界（Margins）:直觉（Intuition）

咱们这回讲 SVM 学习算法，从边界（margins）开始说起。这一节我们会给出关于边界的一些直观展示（intuitions），以及过对于我们做出的预测的信心（confidence）；在本章的第三节中，会对这些概念进行更正式化的表述。

考虑逻辑回归，其中的概率分布p(y = 1|x; θ) 是基于 hθ(x) = g(θTx) 而建立的模型。当且仅当 hθ(x) ≥ 0.5 ，也就是 θTx ≥ 0 的时候，我们才会预测出“1”。假如有一个正向（Positive）的训练样本（positive training example）（y = 1）。那么θTx 越大，hθ (x) = p(y = 1|x; w, b) 也就越大，我们对预测 Label 为 1 的“信心（confidence）”也就越强。所以如果 y = 1 且 θT x ≫ 0（远大于 0），那么我们就对这时候进行的预测非常有信心，当然这只是一种很不正式的粗略认识。与之类似，在逻辑回归中，如果有 y = 0 且 θT x ≪ 0（远小于 0），我们也对这时候给出的预测很有信心。所以还是以一种非常不正式的方式来说，对于一个给定的训练集，如果我们能找到一个 θ，满足当 y(i) = 1 的时候总有 θT x(i) ≫ 0，而 y(i) = 0 的时候则 θT x(i) ≪ 0，我们就说这个对训练数据的拟合很好，因为这就能对所有训练样本给出可靠（甚至正确）的分类。似乎这样就是咱们要实现的目标了，稍后我们就要使用**函数边界记号**（**notion of functional margins**）来用正规的语言来表达该思路。

还有另外一种的直观表示，例如下面这个图当中，画叉的点表示的是正向训练样本，而小圆圈的点表示的是负向训练样本，图中还画出了**分类边界**（**decision boundary**），这条线也就是通过等式 θT x = 0 来确定的，也叫做**分类超平面**（**separating hyperplane**）。图中还标出了三个点 A，B 和 C。



可以发现 A 点距离分界线很远。如果我们对 A 点的 y 值进行预测，估计我们会很有信心地认为在那个位置的 y = 1。与之相反的是 C，这个点距离边界线很近，而且虽然这个 C 点也在预测值 y = 1 的一侧，但看上去距离边界线的距离实在是很近的，所以也很可能会让我们对这个点的预测为 y = 0。因此，我们对 A 点的预测要比对 C 点的预测更有把握得多。B 点正好在上面两种极端情况之间，更广泛地说，如果一个点距离**分类超平面**（**separating hyperplane**）比较远，我们就可以对给出的预测很有信心。那么给定一个训练集，如果我们能够找到一个分类边界，利用这个边界我们可以对所有的训练样本给出正确并且有信心（也就是数据点距离分类边界要都很远）的预测，那这就是我们想要达到的状态了。当然上面这种说法还是很不正规，后面我们会使用**几何边界记号**（**notion of geometric margins**）来更正规地来表达。

2 记号（Notation）

在讨论 SVMs 的时候，出于简化的目的，我们先要引入一个新的记号，用来表示分类。假设我们要针对一个二值化分类的问题建立一个线性分类器，其中用来分类的标签（label）为 y，分类特征（feature）为 x。从此以后我们就用 y ∈ {−1, 1} (而不是之前的 {0, 1}) 来表示这个分类标签了。另外，以后咱们也不再使用向量 θ 来表示咱们这个线性分类器的参数了，而是使用参数 w 和 b，把分类器写成下面这样：



当 z ≥ 0，则 g(z) = 1；而反之若 z ＜ 0，则g(z) = -1。这里的这个 “w, b” 记号就可以让我们能把截距项（intercept term）b 与其他的参数区别开。（此外我们也不用再像早些时候那样要去设定 x0 = 1 这样的一个额外的输入特征向量了。）所以，这里的这个参数 b 扮演的角色就相当于之前的参数 θ0 ，而参数 w 则相当于 [θ1 . . . θn]T。

还要注意的是，从我们上面对函数 g 的定义，可以发现我们的分类器给出的预测是 1 或者 -1 （参考 感知器算法 perceptron algorithm），这样也就不需要先通过中间步骤（intermediate step）来估计 y 为 1 的概率（这就是逻辑回归中的步骤）。

3 函数边界和几何边界（Functional and geometric margins）

咱们来用正规语言来将函数边界和几何边界的记号的概念进行正规化。给定一个训练集 (x(i), y(i))，我们用下面的方法来定义对应该训练集的函数边界 (w, b)：



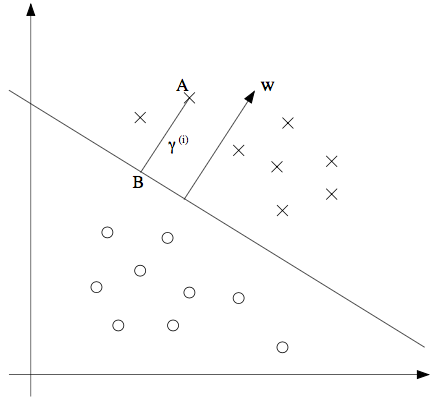
要注意，如果 y(i) = 1，那么为了让函数边界很大（也就是说，我们的预测很可信并且很正确），我们就需要 wT x + b 是一个很大的正数。与此相对，如果 y(i) = −1，那么为了让函数边界很大，我们就需要wT x + b 是一个很大（绝对值）的负数。而且，只要满足 y(i)(wT x + b) > 0，那我们针对这个样本的预测就是正确的。（自己检验证明吧。）因此，一个大的函数边界就表示了一个可信且正确的预测。

对于一个线性分类器，选择上面给定的函数 g （取值范围为{−1, 1}），函数边界的一个性质却使得这个分类器并不具有对置信度的良好量度。例如上面给定的这个函数 g，我们会发现，如果用 2w 替换掉 w，然后用 2b 替换 b，那么由于有 g(wTx + b) = g(2wTx + 2b)，所以这样改变也并不会影响 hw,b(x)。也就是说，函数 g 以及 hw,b(x) 只取决于 wT x + b 的正负符号（sign），而不受其大小（magnitude）的影响。然而，把(w, b) 翻倍成 (2w,2b) 还会导致函数距离也被放大了 2 倍。因此，这样看来就是只要随意去调整 w 和 b 的范围，我们就可以人为调整函数边界到足够大了，而不用去改变任何有实际意义的变量。直观地看，这就导致我们有必要引入某种归一化条件，例如使 ||w||2 = 1；也就是说，我们可以将 (w, b) 替换成 (w/||w||2,b/||w||2)，然后考虑对应 (w/||w||2,b/||w||2) 的函数边界。我们稍后再详细讨论这部分内容。

给定一个训练集 S = {(x(i),y(i)); i = 1, ..., m}，我们将对应 S 的函数边界 (w, b) 定义为每个训练样本的函数边界的最小值。记作 γˆ，可以写成：

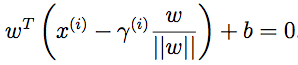


接下来，咱们要讲的是几何边界（geometric margins）。例如下图所示：

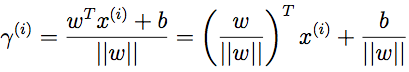


图中给出了对应 (w, b) 的分类边界，其倾斜方向（即法线方向）为向量 w 的方向。这里的向量 w 是与**分类超平面**垂直的（orthogonal，即夹角为 90°）。（你需要说服自己现实情况一定是这样的。）假设有图中所示的一个点 A，此点表示的是针对某训练样本的输入特征为 x(i) ，对应的标签（label）为 y(i) = 1。然后这个点到分类边界的距离 γ(i), 就通过 AB 之间的线段能够获得。

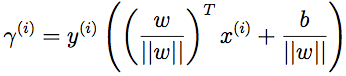
怎么找到的 γ(i) 值呢？这样，w/||w|| 是一个单位长度的向量，指向与 w 相同的方向。因为这里 A 点表示的是 x(i)，所以就能找到一个点 B，其位置为 x(i) − γ(i) · w/||w||。这个 B 点正好位于分类边界线上面，而这条线上的所有 x 都满足等式 wT x + b = 0 ，所以有：



通过上面的方程解出来的 γ(i) 为：



这个解是针对图中 A 处于训练样本中正向部分这种情况，这时候位于“正向（positive）”一侧就是很理想的情况。如果更泛化一下，就可以定义对应训练样本 (x(i), y(i)) 的几何边界 (w, b) 为：



这里要注意，如果 ||w|| = 1，那么函数边界（functional margin）就等于几何边界（geometric margin）——我们可以用这种方法来将两个边界记号联系起来。此外，几何边界是不受参数缩放的影响的；也就是说，如果我们把 w 改为 2w，b 改为 2b，那么几何边界并不会改变。稍后这个性质就会派上用场了。特别要注意的是，由于这个与参数缩放的无关性，当试图对某个数据集的 w 和 b 进行拟合的时候，我们就可以倒入一个任意设置的缩放参数来约束 w，而不会改变什么重要项；例如，我们可以设置 ||w|| = 1，或者 |w1| = 5，或者 |w1 +b|+|w2| = 2，等等都可以，这些都只需要对 w 和 b 进行缩放就可以满足了。

Finally, given a training set S = {(x(i), y(i)); i = 1, . . ., m}, we also define the geometric margin of (w, b) with respect to S to be the smallest of the geometric margins on the individual training examples:

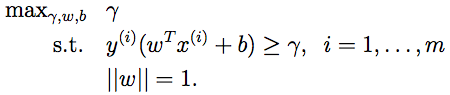
最后，给定一个训练集 S = {(x(i), y(i)); i = 1, ..., m}，我们也可以我们将对应 S 的几何边界 (w, b) 定义为每个训练样本的几何边界的最小值：



4 最优边界分类器（optimal margin classifier）

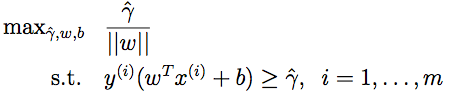
给定一个训练集，根据咱们前文的讨论，似乎很自然地第一要务就是要尝试着找出一个分类边界，使（几何）边界能够最大，因为这会反映出对训练集进行的一系列的置信度很高的分类预测，也是对训练数据的一个良好“拟合（fit）”。这样生成的一个分类器，能够把正向和负向的训练样本分隔开，中间有一个“空白区（gap）”，也就是几何边界。

到目前为止，我们都是假定给定的训练集是线性可分（linearly separable）的；也就是说，能够在正向和负向的样本之间用某种分类超平面来进行划分。那要怎样找到能够得到最大几何边界的那一组呢？我们可以提出下面的这样一个优化问题（optimization problem）：



I.e., we want to maximize γ, subject to each training example having functional margin at least γ. The ||w|| = 1 constraint moreover ensures that the functional margin equals to the geometric margin, so we are also guaranteed that all the geometric margins are at least γ. Thus, solving this problem will result in (w, b) with the largest possible geometric margin with respect to the training set.

If we could solve the optimization problem above, we’d be done. But the “||w|| = 1” constraint is a nasty (non-convex) one, and this problem certainly isn’t in any format that we can plug into standard optimization software to solve. So, let’s try transforming the problem into a nicer one. Consider:

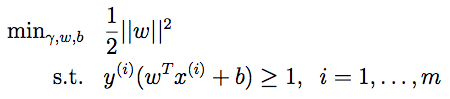


Here, we’re going to maximize γˆ/||w||, subject to the functional margins all being at least γˆ. Since the geometric and functional margins are related by γ = γˆ/||w|, this will give us the answer we want. Moreover, we’ve gotten rid of the constraint ||w|| = 1 that we didn’t like. The downside is that we now have a nasty (again, non-convex) objective γˆ function; and, we still don’t ||w|| have any off-the-shelf software that can solve this form of an optimization problem.

Let’s keep going. Recall our earlier discussion that we can add an arbi- trary scaling constraint on w and b without changing anything. This is the key idea we’ll use now. We will introduce the scaling constraint that the functional margin of w, b with respect to the training set must be 1:



Since multiplying w and b by some constant results in the functional margin being multiplied by that same constant, this is indeed a scaling constraint, and can be satisfied by rescaling w, b. Plugging this into our problem above, and noting that maximizing γˆ/||w|| = 1/||w|| is the same thing as minimizing ||w||2, we now have the following optimization problem:



We’ve now transformed the problem into a form that can be efficiently solved. The above is an optimization problem with a convex quadratic ob- jective and only linear constraints. Its solution gives us the optimal mar- gin classifier. This optimization problem can be solved using commercial quadratic programming (QP) code.1

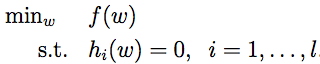
1You may be familiar with linear programming, which solves optimization problems that have linear objectives and linear constraints. QP software is also widely available, which allows convex quadratic objectives and linear constraints.

While we could call the problem solved here, what we will instead do is make a digression to talk about Lagrange duality. This will lead us to our optimization problem’s dual form, which will play a key role in allowing us to use kernels to get optimal margin classifiers to work efficiently in very high dimensional spaces. The dual form will also allow us to derive an efficient algorithm for solving the above optimization problem that will typically do much better than generic QP software.

5 Lagrange duality

Let’s temporarily put aside SVMs and maximum margin classifiers, and talk about solving constrained optimization problems.

Consider a problem of the following form:



Some of you may recall how the method of Lagrange multipliers can be used to solve it. (Don’t worry if you haven’t seen it before.) In this method, we define the Lagrangian to be

