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。

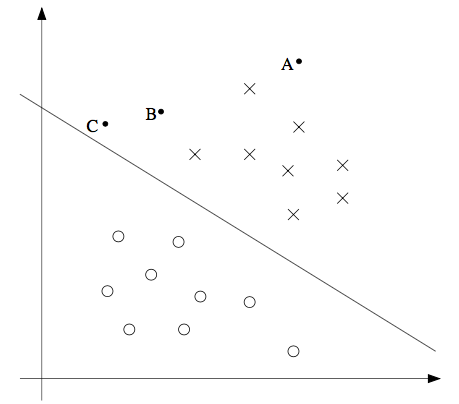
1 边界（Margins）:直觉（Intuition）

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。

Consider logistic regression, where the probability p(y = 1|x; θ) is modeled by hθ(x) = g(θTx). We would then predict “1” on an input x if and only if hθ(x) ≥ 0.5, or equivalently, if and only if θTx ≥ 0. Consider a positive training example (y = 1). The larger θTx is, the larger also is hθ (x) = p(y = 1|x; w, b), and thus also the higher our degree of “confidence” that the label is 1. Thus, informally we can think of our prediction as being a very confident one that y = 1 if θT x ≫ 0. Similarly, we think of logistic regression as making a very confident prediction of y = 0, if θT x ≪ 0. Given a training set, again informally it seems that we’d have found a good fit to the training data if we can find θ so that θT x(i) ≫ 0 whenever y(i) = 1, and θT x(i) ≪ 0 whenever y(i) = 0, since this would reflect a very confident (and correct) set of classifications for all the training examples. This seems to be a nice goal to aim for, and we’ll soon formalize this idea using the notion of functional margins.

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”。

For a different type of intuition, consider the following figure, in which x’s represent positive training examples, o’s denote negative training examples, a decision boundary (this is the line given by the equation θT x = 0, and is also called the separating hyperplane) is also shown, and three points have also been labeled A, B and C.



Notice that the point A is very far from the decision boundary. If we are asked to make a prediction for the value of y at A, it seems we should be quite confident that y = 1 there. Conversely, the point C is very close to the decision boundary, and while it’s on the side of the decision boundary on which we would predict y = 1, it seems likely that just a small change to the decision boundary could easily have caused our prediction to be y = 0. Hence, we’re much more confident about our prediction at A than at C. The point B lies in-between these two cases, and more broadly, we see that if a point is far from the separating hyperplane, then we may be significantly more confident in our predictions. Again, informally we think it’d be nice if, given a training set, we manage to find a decision boundary that allows us to make all correct and confident (meaning far from the decision boundary) predictions on the training examples. We’ll formalize this later using the notion of geometric margins.

2 Notation

To make our discussion of SVMs easier, we’ll first need to introduce a new notation for talking about classification. We will be considering a linear classifier for a binary classification problem with labels y and features x. From now, we’ll use y ∈ {−1, 1} (instead of {0, 1}) to denote the class labels. Also, rather than parameterizing our linear classifier with the vector θ, we will use parameters w, b, and write our classifier as



Here, g(z) = 1 if z ≥ 0, and g(z) = −1 otherwise. This “w,b” notation allows us to explicitly treat the intercept term b separately from the other parameters. (We also drop the convention we had previously of letting x0 = 1 be an extra coordinate in the input feature vector.) Thus, b takes the role of what was previously θ0, and w takes the role of [θ1 . . . θn]T .

Note also that, from our definition of g above, our classifier will directly predict either 1 or −1 (cf. the perceptron algorithm), without first going through the intermediate step of estimating the probability of y being 1 (which was what logistic regression did).

3 Functional and geometric margins

Let’s formalize the notions of the functional and geometric margins. Given a training example (x(i), y(i)), we define the functional margin of (w, b) with respect to the training example



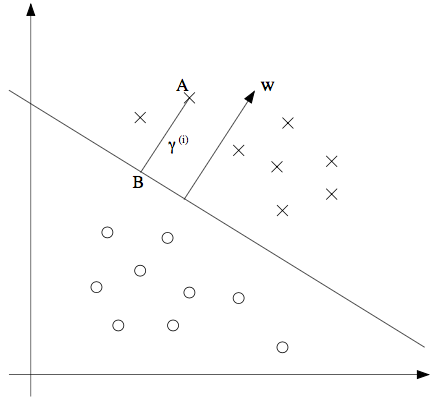
Note that if y(i) = 1, then for the functional margin to be large (i.e., for our prediction to be confident and correct), we need wT x + b to be a large positive number. Conversely, if y(i) = −1, then for the functional margin to be large, we need wT x + b to be a large negative number. Moreover, if y(i)(wT x + b) > 0, then our prediction on this example is correct. (Check this yourself.) Hence, a large functional margin represents a confident and a correct prediction.

For a linear classifier with the choice of g given above (taking values in {−1, 1}), there’s one property of the functional margin that makes it not a very good measure of confidence, however. Given our choice of g, we note that if we replace w with 2w and b with 2b, then since g(wTx+b) = g(2wTx+2b), this would not change hw,b(x) at all. I.e., g, and hence also hw,b(x), depends only on the sign, but not on the magnitude, of wT x + b. However, replacing (w,b) with (2w,2b) also results in multiplying our functional margin by a factor of 2. Thus, it seems that by exploiting our freedom to scale w and b, we can make the functional margin arbitrarily large without really changing anything meaningful. Intuitively, it might therefore make sense to impose some sort of normalization condition such as that ||w||2 = 1; i.e., we might replace (w,b) with (w/||w||2,b/||w||2), and instead consider the functional margin of (w/||w||2,b/||w||2). We’ll come back to this later.

Given a training set S = {(x(i),y(i));i = 1,...,m}, we also define the function margin of (w, b) with respect to S to be the smallest of the functional margins of the individual training examples. Denoted by γˆ, this can therefore be written:

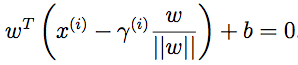


Next, let’s talk about geometric margins. Consider the picture below:

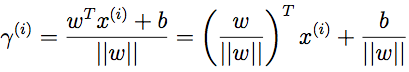


The decision boundary corresponding to (w,b) is shown, along with the vector w. Note that w is orthogonal (at 90◦) to the separating hyperplane. (You should convince yourself that this must be the case.) Consider the point at A, which represents the input x(i) of some training example with label y(i) = 1. Its distance to the decision boundary, γ(i), is given by the line segment AB.

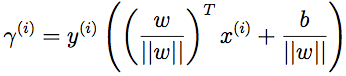
How can we find the value of γ(i)? Well, w/||w|| is a unit-length vector pointing in the same direction as w. Since A represents x(i), we therefore find that the point B is given by x(i) − γ(i) · w/||w||. But this point lies on the decision boundary, and all points x on the decision boundary satisfy the equation wT x + b = 0. Hence,



Solving for γ(i) yields



This was worked out for the case of a positive training example at A in the figure, where being on the “positive” side of the decision boundary is good. More generally, we define the geometric margin of (w,b) with respect to a training example (x(i), y(i)) to be



Note that if ||w|| = 1, then the functional margin equals the geometric margin—this thus gives us a way of relating these two different notions of margin. Also, the geometric margin is invariant to rescaling of the parame- ters; i.e., if we replace w with 2w and b with 2b, then the geometric margin does not change. This will in fact come in handy later. Specifically, because of this invariance to the scaling of the parameters, when trying to fit w and b to training data, we can impose an arbitrary scaling constraint on w without changing anything important; for instance, we can demand that ||w|| = 1, or |w1| = 5, or |w1 +b|+|w2| = 2, and any of these can be satisfied simply by rescaling w and b.

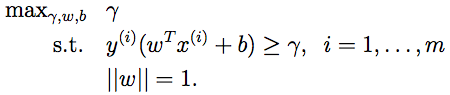
Finally, given a training set S = {(x(i), y(i)); i = 1, . . . , m}, we also define the geometric margin of (w,b) with respect to S to be the smallest of the geometric margins on the individual training examples:



4 The optimal margin classifier

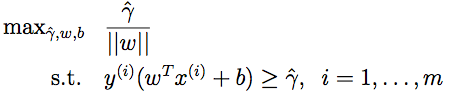
Given a training set, it seems from our previous discussion that a natural desideratum is to try to find a decision boundary that maximizes the (ge- ometric) margin, since this would reflect a very confident set of predictions on the training set and a good “fit” to the training data. Specifically, this will result in a classifier that separates the positive and the negative training examples with a “gap” (geometric margin).

For now, we will assume that we are given a training set that is linearly separable; i.e., that it is possible to separate the positive and negative ex- amples using some separating hyperplane. How we we find the one that achieves the maximum geometric margin? We can pose the following opti- mization problem:



I.e., we want to maximize γ, subject to each training example having func- tional 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:

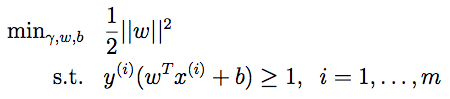


Here, we’re going to maximize γˆ/||w||, subject to the functional margins all being at least γˆ. Since the geometric and functional margins are related by γ = γˆ/||w|, this will give us the answer we want. Moreover, we’ve gotten rid of the constraint ||w|| = 1 that we didn’t like. The downside is that 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

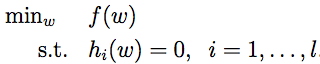
1You may be familiar with linear programming, which solves optimization problems that have linear objectives and linear constraints. QP software is also widely available, which allows convex quadratic objectives and linear constraints.

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

