Support Vector Machine Notes

Ryan Gehring

Contents

Theory

Model Parametrization

Given a collection of n points, training examples of form $\{\mathbf{x_i}, y_i\}$, $\mathbf{x_i} \in \mathbb{R}^n, y_i \in \{-1, 1\}$, $i - 1 \in \mathbb{Z}^n$ we desire to separate the positive examples ($y_i = 1$) from the negative examples with a plane of form:

$$\mathbf{wx} - b = 0 \tag{1}$$

This is only possible if all the training examples are linearly separable, that is, a plane exists in $\mathbb{R}^n$ that separates the positive examples from the negative ones. A popular analogy is to think of this plane as the median line running down a street.

Objective Function

Continuing the analogy from before, the 'gutters' of the street are the parallel hyperplanes running through the nearest positive and negative examples:

$$\mathbf{wx} - b = \pm 1 \tag{2}$$

The shortest distance $d_{\pm 1}$ between the origin and one of the hyperplanes can be calculated by noting that the shortest distance vector falls along the normal vector $\mathbf{w}$ and intersects the plane.

$$\mathbf{w}\left(d_{\pm 1}\frac{\mathbf{w}}{||\mathbf{w}||}\right) = b \pm 1 \tag{3}$$

The distance between the two hyperplanes can then be calculated as the difference of the two distances from the origin.

$$d = ||d_{+1} - d_{-1}|| = \frac{2}{||\mathbf{w}||} \tag{4}$$

We will seek to find the plane which has the widest possible 'lanes'—that is, which maximizes the distance between the hyperplanes coincident with the nearest positive and negative examples. As we saw above this distance is inversely proportional to the magnitude of $\mathbf{w}$, So this is equivalent to minimizing $||\mathbf{w}||$ subject to the constraints that the positive and negative y_i are separated by the two hyperplanes. Finally, we elect to minimize the square of the above, $\frac{1}{2}||\mathbf{w}||^2$, because minimizing (squared) Euclidean distance will satisfy the KKT conditions we will later enumerate, enabling us to reformulate the function in terms of inner products of feature vectors and employ the 'kernel trick' an idea we will later develop.

Constraints

Formally the n constraints are:

$$y_i = 1 \implies \mathbf{wx_i} - b \geq 1$$
$$y_i = -1 \implies \mathbf{wx_i} - b \leq -1$$

Which boils down to:

$$1 - y_i\left(\mathbf{wx_i} - b\right) \leq 0 \tag{5}$$

The above constrained optimization problem can be solved via quadratic programming. However, for the reasons noted above we will explore solving a related optimization called a *Lagrangian dual problem.*

Theory continued on next page...

Lagrange Multipliers

For a general minimization problem of the function $f(\mathbf{w})$ subject to boundary conditions $g_i(\mathbf{w})$,

$$p^* = \min_{\mathbf{w}} f(\mathbf{w}), \forall i, g_i(\mathbf{w}) \leq c_i \tag{6}$$

The method of solution via Lagrange multipliers considers specifying several contour paths, one for the function to be minimized and one for each constraint. (Contour paths are points where the function value is held constant). For some set of choices of constants $c_1 ... c_n$, the curves will be tangent at the global minimum point, and the gradient of each curve will be a scalar multiple of each other.
(For any contour path, the gradient is normal to the contour.) (If it weren't, the value of the function would not be constant along the contour.) So at a tangent, where the contours are pointed in the same direction, the gradients for all paths will be coincident.

$$\nabla f(\mathbf{w}) = \sum_{i=1}^{n} \lambda_i \nabla g_i(\mathbf{w}) \tag{7}$$

With this insight in hand, we can make a clever transformation and come up with an *unconstrained* function, minimization of which represents the solution to the global minimization problem! Map the constraint $g_i(\mathbf{w}) \leq c_i$ to $h_i(\mathbf{w}) = g_i(\mathbf{w}) - c_i \leq 0$ and add to f to create the Lagrangian:

$$\mathcal{L}(\mathbf{w}, \lambda) = f(\mathbf{w}) + \sum_{i=1}^{n} \lambda_i h_i(\mathbf{w}) \tag{8}$$

Now if we seek to minimize $\mathcal{L}(\mathbf{w}, \lambda)$ taking the gradient with respect to $\mathbf{w}$ and setting to zero yields equation (7) — and taking the partial derivatives with respect to λ_i and setting to zero yields $h_i(\mathbf{w}) = 0$ — a candidate solution! Minimizing the lagrangian explicitly is called the 'primal' form of the optimization problem.
The primal problem is unconstrained—if we take λ to be positive, then by the fact that $h(\mathbf{w}) \leq 0$ we know that $f(\mathbf{w}) \geq \mathcal{L}(\mathbf{w}, \lambda)$. The primal form expressed formally is:

$$p^* = \min_{\mathbf{w}} \max_{\lambda \geq 0} \mathcal{L}(\mathbf{w}, \lambda) \tag{9}$$

Lagrangian Duality

Formally the dual problem results from interchanging the order of the max and min operations in the primal problem:

$$p_2^* = \max_{\lambda \geq 0} \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) \tag{10}$$

The motivation for so doing is interesting and not immediately obvious so I will digress briefly about it. Suppose you are seeking to purchase a schedule of items $\mathbf{w}$ whose cost is $f(\mathbf{w})$ subject to shopping list constraints— that is, you want a certain range of quantities of apples, bananas, etc. The primal form of the solution fixes the schedule of items, and then finds parameters $\lambda_1 ... \lambda_n$ which minimize the 'cost' of any boundary condition violations $\lambda_i h(\mathbf{w})$. In the primal form, this cost is either infinity (constraint violated) or zero (no constraint violated). By considering a number of possible schedules of purchases, you arrive at the cheapest solution which satisfies your constraints.
The dual form results from considering the problem above from the perspective of the merchant rather than the customer. Here we first fix the 'unit cost' vector λ of boundary violations (producing too many or too few items), then compute the worst-case, minimum economic outcome from an adversarial customer's shopping choices. By following this line of reasoning for many choices of cost structures, we arrive at an outcome that maximizes profit to the merchant. It so happens that this is a lower bound on the primal solution—f

Theory continued on next page...

(revenue) is always greater than or equal to $\mathcal{L}$ (profit), and so the merchant will always earn less money than the customer pays.
Under strong duality (f is convex, solution is strictly feasible), it turns out that the the dual solution is both a lower bound and an upper bound on the primal solution—in other words, for some problems, they are equal. If you have convexity + feasibility then the KKT conditions listed below are also true. KKT conditions basically say that the solution has to be within the boundary and have derivative zero with respect to all free parameters, and one more thing: $\lambda_i > 0 \implies h_i = 0$

$$\nabla_{\mathbf{w}}\mathcal{L} = 0 \tag{11}$$
$$\nabla_{\lambda}\mathcal{L} = 0 \tag{12}$$
$$\forall i, h_i(\mathbf{w}) \leq 0 \tag{13}$$
$$\forall i, \lambda_i \geq 0 \tag{14}$$
$$\forall i, \lambda_i h_i(\mathbf{w}) = 0 \tag{15}$$

The dual problem for SVM Classifiers

We can now solve the deal problem, which will have the interesting property that it is written entirely in terms of inner products of feature vectors, which will allow us to employ the kernel trick to turn SVM into a nonlinear classifier by projecting our data into high dimensional space.
Recalling (5) we can write the Lagrangian as

$$\mathcal{L}(\mathbf{w}, \lambda) = \frac{1}{2}\mathbf{w}\mathbf{w} + \sum_{i=1}^{n} \lambda_i \left(y_i \left(\mathbf{w}\mathbf{x_i} - b\right) - 1\right) \tag{16}$$

Recall from the KKT criterion that λ_i is only greater than 0 for cases where $g = 0$, that is, when the training point is right on the margin. Usually this only holds for a few points out of all the training data—these points are called the **support vectors**, and they contain all the information needed to train this type of classifier! (You could actually throw away most of the data and get the same result!)
Now that we have the primal form, we can find the dual form. So we must minimize first over $\mathbf{w}$ and b. We note KKT (11) is satisfied for a minimum $\mathbf{w}$:

$$\nabla_{\mathbf{w}}\mathcal{L} = 0 = \mathbf{w} + \sum_{i=1}^{n} \lambda_i y_i \mathbf{x_i} \tag{17}$$

Taking the derivative with respect to b yields:

$$0 = -\sum_{i=1}^{n} \lambda_i y_i \tag{18}$$

Reducing (16) using above 2 relations yields the dual optimization problem:

$$\mathcal{D}(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}\right) \tag{19}$$

Kernel Trick

Kernel's are the dot product of transformations called *feature mappings* of an input vector. Given a feature mapping ϕ,

$$K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T \phi(\mathbf{x_j}) \tag{20}$$

Theory continued on next page...

K is a valid kernel if the kernel matrix (matrix giving the kernel result between all pairs of n items) is symmetric and positive semi-definite. (Basically, just the properties of the dot product).
Benefits of kernel function - by mapping to high dimensional space you can build a nonlinear classifier.

Non-separable data

In the case that the data is linearly inseparable, we can alter the loss function to a hinge function as follows:

$$f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}|| + C\sum_i \epsilon_i \tag{21}$$

where ϵ_i is the *slack variable* for each training case, the amount by which the margin misclassifies the case. The dual form of this optimization problem is given the same equation as before, but there are altered constraints:

$$\mathcal{D}(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}\right) \tag{22}$$

$$0 \le \lambda_i \le C \tag{23}$$

$$\sum \lambda_i y_i = 0 \tag{24}$$

Solving the Dual Problem via SMO

Coordinate ascent is an algorithm which seeks to maximize a function of several variables by iteratively maximizing the function by each variable until convergence. IE, if you have a function of x and y, coordinate ascent would start at a given point, maximize in the x axis, then y, then x, then y again, until convergence.
Sequential minimzal optimization is a simple idea - we are trying to maximize our dual function over λ but our constraint shows that λ_i is determined by the other λ's. So, rather than arg-maxing over one parameter at a time, we do it over two, and repeat with a new pair until convergence.
The update step is derived as follows:

$$\lambda_i y_i + \lambda_j y_j = \eta \tag{25}$$

Where η is a constant equal derived from the constraint (24). And if you substitute this into the dual loss function for λ_i and keep the other λ's constant, you have a quadratic equation in λ_i which is trivial to solve. Then use that value of λ_i to determine the maximizing value of $lambda_j$. Repeat until KKT is satisfied within a tolerance.

Application

Java Implementation

See the Github Repo for an implementation.