

Support Vector Machines

Lecturer: Xiaojin Zhu

jerryzhu@cs.wisc.edu

Many NLP problems can be formulated as classification, e.g., word sense disambiguation, spam filtering, sentiment analysis, document retrieval, speech recognition, etc. There are in general 3 ways to do classification:

1. Create a generative model $p(y), p(x|y)$, and compute $p(y|x)$ with Bayes rule. Classify according to $p(y|x)$. For example Naive Bayes.
2. Create a discriminative model $p(y|x)$ directly. Classify according to $p(y|x)$. For example logistic regression.
3. Forget about probabilities. Create a discriminant function $f : \mathcal{X} \rightarrow \mathcal{Y}$, and classify according to $f(x)$. Support Vector Machine (SVM) is such an approach.

1 The Linearly Separable Case

We assume binary classification. The intuition of SVM is to put a hyperplane in the middle of the two classes, so that the distance to the nearest positive or negative example is maximized. Note this essentially ignores the class distribution $p(x|y)$, and is more similar to logistic regression.

The SVM discriminant function has the form

$$f(x) = w^\top x + b, \quad (1)$$

where w is the parameter vector, and b is the bias or offset scalar. The classification rule is $\text{sign}(f(x))$, and the linear decision boundary is specified by $f(x) = 0$. The labels $y \in \{-1, 1\}$.

If f separates the data, the geometric distance between a point x and the decision boundary is

$$\frac{yf(x)}{\|w\|}. \quad (2)$$

To see this, note $w^\top x$ is *not* the geometric distance between x 's projection on w and the origin: it must be normalized by the norm of w .

Given training data $\{(x, y)_{1:n}\}$, we want to find a decision boundary w, b such that to maximize the geometric distance of the closest point, i.e.

$$\max_{w, b} \min_{i=1}^n \frac{y_i(w^\top x_i + b)}{\|w\|}. \quad (3)$$

Note this is the key difference between SVM and logistic regression: they optimize different objectives.

The above objective is difficult to optimize directly. Here is a trick: notice for any \hat{w}, \hat{b} , the objective is the same for $\kappa\hat{w}, \kappa\hat{b}$ for any nonzero scaling factor κ . That is to say, the optimization (3) is actually over equivalence classes of w, b up to scaling. Therefore, we can reduce the redundancy by requiring the closest point to the decision boundary to satisfy:

$$yf(x) = y(w^\top x + b) = 1, \quad (4)$$

which implies all points to satisfy

$$yf(x) = y(w^\top x + b) \geq 1. \quad (5)$$

This converts the unconstrained but complex problem (3) into a constrained but simpler problem

$$\max_{w,b} \quad \frac{1}{\|w\|} \quad (6)$$

$$s.t. \quad y_i(w^\top x_i + b) \geq 1 \quad i = 1 \dots n. \quad (7)$$

Maximizing $\frac{1}{\|w\|}$ is equivalent to minimizing $\frac{1}{2}\|w\|^2$, but the latter will prove convenient later. Our problem now becomes

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \quad (8)$$

$$s.t. \quad y_i(w^\top x_i + b) \geq 1 \quad i = 1 \dots n. \quad (9)$$

This is known as a *quadratic programming* problem, where the objective is a quadratic function of the variable (in this case w), and there are linear inequality constraints. Standard optimization packages can solve such a problem (but often slowly for high dimensional x and large n). However, we will next derive the dual optimization problem. The dual problem has two advantages: 1. It illustrates the reason behind the name ‘support vector’; 2. It can use the powerful kernel trick.

The basic idea is to form the Lagrangian, and maximize the Lagrangian wrt the Lagrange multipliers (called dual variables). To this end, we introduce $\alpha_{1:n} \geq 0$, and define the Lagrangian

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^n \alpha_i (y_i(w^\top x_i + b) - 1). \quad (10)$$

Setting $\partial L(w, b, \alpha)/\partial w = 0$ we obtain

$$w = \sum_{i=1}^n \alpha_i y_i x_i. \quad (11)$$

Setting $\partial L(w, b, \alpha)/\partial b = 0$ we obtain

$$\sum_{i=1}^n \alpha_i y_i = 0. \quad (12)$$

Putting these into the Lagrangian and we get the dual objective as a function of α only, which is to be maximized along with the following constraints,

$$\max_{\alpha} \quad -\frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^n \alpha_i \quad (13)$$

$$s.t. \quad \alpha_i \geq 0 \quad i = 1 \dots n \quad (14)$$

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (15)$$

This is again a constrained quadratic programming problem. We call (9) the primal problem, and (15) the dual problem. They are equivalent, but the primal has $D+1$ variables, where D is the number of dimensions of x . In contrast, the dual has n variables, where n is the number of training examples. In general, one should pick the smaller problem to solve. However, as we soon see, the dual form allows so called ‘kernel trick’.

If we solve the primal problem, our discriminant function is simply

$$f(x) = w^\top x + b. \quad (16)$$

If we solve the dual problem, from (11) we see that

$$f(x) = \sum_{i=1}^n \alpha_i y_i x_i^\top x + b. \quad (17)$$

Where is b in the dual problem? We have to go back to the constraints (9) and Lagrange multipliers α , and make use of the KKT condition, which states that at the solution, the primal and dual constraints hold, and a complementarity condition holds:

$$\alpha_i \geq 0 \quad (18)$$

$$y_i(w^\top x_i + b) - 1 \geq 0 \quad (19)$$

$$\alpha_i(y_i(w^\top x_i + b) - 1) = 0. \quad (20)$$

The complementarity condition implies that if either α_i or $y_i(w^\top x_i + b) - 1$ is strictly positive, the other must be zero.

This observation is significant. Define the two lines $f(x) = 1$ and $f(x) = -1$ to be the *margin* of the decision boundary. The complementarity condition states that only data points on the margin will have a non-zero α . Such points are called *support vectors*, because they define the decision boundary¹. All other points have $\alpha = 0$. They can be removed without affecting the solution. This is very different from logistic regression, which depends on all points. This property is called sparsity, which is quite desirable for computational reasons: f can be represented by support vectors (17), whose number is usually smaller than n .

With the support vectors, we can finally compute b . For any support vector x_i ($\alpha_i > 0$), the complementarity condition gives (note $1/y_i = y_i$)

$$b = y_i - w^\top x_i = y_i - \sum_{j=1}^n \alpha_j y_j x_j^\top x_i. \quad (21)$$

It is numerically more stable to average over all support vectors

$$b = \frac{1}{n_{s.v.}} \sum_{i \in s.v.} \left(y_i - \sum_{j=1}^n \alpha_j y_j x_j^\top x_i \right). \quad (22)$$

2 The Linearly Non-Separable Case

So far we assumed that the training data is linearly separable. However many real datasets are not linearly separable, and the previous problem (9) has no solution. To handle non-separable datasets, we relax the constraints by making the inequalities easier to satisfy. This is done with *slack variables* $\xi_i \geq 0$, one for each constraint:

$$y_i(w^\top x_i + b) \geq 1 - \xi_i \quad i = 1 \dots n. \quad (23)$$

Now a point x_i can satisfy the constraint even if it is on the wrong side of the decision boundary, as long as ξ_i is large enough. Of course all constraints can be trivially satisfied this way. To prevent this, we penalize the sum of ξ_i , and arrive at the new primal problem

$$\min_{w,b,\xi} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad (24)$$

$$s.t. \quad y_i(w^\top x_i + b) \geq 1 - \xi_i \quad i = 1 \dots n \quad (25)$$

$$\xi_i \geq 0, \quad (26)$$

¹This is for linear-separable datasets. For non-separable datasets, a support vector can be within the margin or even on the wrong side of the decision boundary.

where C is a weight parameter, which needs to be carefully set (e.g., by cross validation).

We can similarly look at the dual problem of (26) by introducing Lagrange multipliers. We arrive at

$$\max_{\alpha} \quad -\frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^n \alpha_i \quad (27)$$

$$s.t. \quad 0 \leq \alpha_i \leq C \quad i = 1 \dots n \quad (28)$$

$$\sum_{i=1}^n \alpha_i y_i = 0. \quad (29)$$

Note the only difference to the linear separable dual problem (15) is the upper bound C on the α 's. As before, when $\alpha = 0$ the point is not a support vector and can be ignored. When $0 < \alpha < C$, it can be shown using complementarity that $\xi = 0$, i.e., the point is on the margin. When $\alpha = C$, the point is inside the margin if $\xi \leq 1$, or on the wrong side of the decision boundary if $\xi > 1$.

The discriminant function is again

$$f(x) = \sum_{i=1}^n \alpha_i y_i x_i^\top x + b. \quad (30)$$

The offset b can be computed on support vectors with $0 < \alpha < C$.

3 The Kernel Trick

The dual problem (29) only involves the dot product $x_i^\top x_j$ of examples, not the example themselves. So does the discriminant function (30). This allows SVM to be *kernelized*.

Consider the dataset $\{(x, y)_{1:3}\} = \{(-1, 1), (0, -1), (1, 1)\}$, where $x \in \mathbb{R}$. This is not a linearly separable dataset. However, if we map x to a three dimensional vector

$$\phi(x) = (1, \sqrt{2}x, x^2)^\top, \quad (31)$$

the dataset becomes linearly separable in the three dimensional space (equivalently, we have a non-linear decision boundary in the original space). The map does not actually increase the intrinsic dimensionality of x : $\phi(x)$ lies on a one dimensional manifold in the 3D space. Nonetheless, this suggests a general way to handle linearly non-separable data: map x to some $\phi(x)$. This is complimentary to the slack variables, so that we can simply replace all x with $\phi(x)$ in (29) and (30).

If $\phi(x)$ is very high dimensional, representing it and computing the inner product becomes an issue. Here is when the kernel kicks in. Note the dual problem and its solution (29) and (30) involves inner product of feature vectors $\phi(x_i)^\top \phi(x_j)$ only. Thus it might be possible to use a feature representation $\phi(x)$ without explicitly representing it, as long as we can compute the inner product. For example, the inner product of (31) can be computed as

$$k(x_i, x_j) = \phi(x_i)^\top \phi(x_j) = (1 + x_i x_j)^2. \quad (32)$$

The computational saving is much bigger for such *polynomial kernels* $k(x_i, x_j) = (1 + x_i x_j)^n$ with larger n , where the explicit feature vector has many more dimensions. For the so-called *Radial Basis Function* (RBF) kernel

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \quad (33)$$

the corresponding feature vector is infinite dimensional.

Thus the kernel trick is to replace $\phi(x_i)^\top \phi(x_j)$ with a kernel function $k(x_i, x_j)$ in (29) and (30). What functions are valid kernels that correspond to some feature vector $\phi(x)$? They must be so-called Mercer kernels $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, where

1. k is continuous,
2. k is symmetric,
3. k is positive definite, i.e., for any m points $x_{1:m}$, the $m \times m$ Gram matrix $K = k(x_{1:m}, x_{1:m})$ is positive semi-definite.

4 Odds and Ends

An equivalent formulation to the SVM constrained optimization problem (26) is the *unconstrained* problem

$$\min_{w,b} \sum_{i=1}^n \max(1 - y_i(w^\top x_i + b), 0) + \lambda \|w\|^2. \quad (34)$$

If we call $y_i(w^\top x_i + b)$ the margin of x_i , the term $\max(1 - y_i(w^\top x_i + b), 0)$ wants the margin of any training point to be larger than 1, i.e., having a confident prediction. The term is known as the *hinge loss* function. Note the above objective is very similar to L2-regularized logistic regress, just with a different loss function (the latter uses negative log likelihood loss).

There is *no probabilistic interpretation* of the margin of a point. There are heuristics to convert margin into probability $p(y|x)$, which works well in practice, but is not justified in theory.

There are many ways to extend binary SVM to multiclass classification. A heuristic method is 1-vs-rest. For a K class problem, create K binary classification subproblems: class 1 vs. $(2-K)$, class 2 vs. $(1,3-K)$, and so on. Solve each subproblem with a binary SVM. Classify x_i to the class for which it has the largest positive margin.

SVM can be extended to regression, by replacing the hinge loss with an ϵ -insensitive loss.