Kernelizing Linear Classifiers

Jason D. M. Rennie
jrennie@csail.mit.edu

December 6, 2003

Abstract

It's common knowledge that the Support Vector Machine (SVM) can be kernelized. In other words, the SVM retains its convex objective even if points are projected into another space. What may be slightly less known is that other linear classification algorithms, such as logistic regression and least squares, can also be kernelized. In this paper, we review the mathematics behind the kernelization of the SVM and other classification algorithms.

1 The Support Vector Machine

For a matrix of training examples, $X = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}$, and a vector of their binary labels, $\mathbf{y}$, the Support Vector Machine[1] finds the vector of weights, $\mathbf{w}$, that minimizes

$$L_{\text{SVM}} = \text{sum}((1 - (X\mathbf{w} + b) \cdot \mathbf{y})_+) + \lambda \mathbf{w}^T \mathbf{w}. \tag{1}$$

The Representer Theorem (see Appendix B of [1]) guarantees us that we lose nothing by rewriting the weight vector, $\mathbf{w}$, in terms of weights on the examples, $\mathbf{c}$. Let $\mathbf{w} = X^T \mathbf{c}$ and $K = XX^T$.

$$L_{\text{SVM}} = \text{sum}((1 - (K\mathbf{c} + b) \cdot \mathbf{y})_+) + \lambda \mathbf{c}^T K \mathbf{c}. \tag{2}$$

The dual form is usually used for optimization since it yields a quadratic program in standard form. Let $Y = \text{diag}(\mathbf{y})$ and $\alpha = \{\alpha_1, \ldots, \alpha_n\}$. We can equivalently maximize[2]

$$L'_{\text{SVM}} = \text{sum}(\alpha) - \frac{1}{4\lambda} \alpha^T Y K Y \alpha. \tag{3}$$

[1] We follow the derivation of [1] except that we eliminate the $\frac{1}{l}$ term.
[2] λ should not be squared in Equation 2.23 of [1].

such that $\mathbf{y}^T\alpha = 0$ and $0 \le \alpha_i \le 1\ \forall i$. A new example, $\mathbf{x}$, is labeled

$$\text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}\right). \tag{4}$$

Note that nowhere do we need to represent individual examples. We need only compute dot (kernel) products between examples. This makes it easy to extend the SVM to non-linear decision boundaries. Let $\Phi : \mathbb{R}^d \to \mathcal{H}$ be a function that maps a data point to a higher-dimensional (possibly infinite) space. If $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$ is easily computable for every pair of examples, there is no need to explicitly store the higher-dimensional representation of example. All we need are the kernel products between pars of points. See §4 of [2] for additional discussion.

2 Regularized Least Squares Classification

Regularized Least Squares Classification[3] (RLSC) is similar to the SVM in that the objective minimizes a loss function plus a penalty term. The main difference is that the loss of an example is the square difference between the label and the classifier output. Unlike the SVM, overconfident outputs are penalized. For a discussion of the intuition behind RLSC, see §3.4 of [1]. Let X be our matrix of training examples; let $\mathbf{y}$ be the vector of labels. RLSC minimizes

$$L_{\text{RLSC}} = (X\mathbf{w} + b\mathbf{1} - \mathbf{y})^T(X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + \lambda\mathbf{w}^T\mathbf{w}. \tag{5}$$

Again, the Representer Theorem guarantees that we lose nothing by making the substitution $\mathbf{w} = X^T\mathbf{c}$,

$$L_{\text{RLSC}} = (K\mathbf{c} + b\mathbf{1} - \mathbf{y})^T(K\mathbf{c} + b\mathbf{1} - \mathbf{y}) + \lambda\mathbf{c}^TK\mathbf{c}. \tag{6}$$

Taking the gradient and solving for zero, we find that we minimize the objective by solving

$$(K + \lambda I)\mathbf{c} + b\mathbf{1} = \mathbf{y}. \tag{7}$$

As observed by [1], this can be minimized directly using, e.g. Conjugate Gradients. See §3 of [1] for additional discussion.

3 Regularized Logistic Regression

Like with RLSC, Regularized Logistic Regression (RLR) simply modifies the loss term. Let X be the set of training examples; let $\mathbf{y}$ be the vector of labels; let $Y = \text{diag}(\mathbf{y})$; let $g(z) = (1 + e^{-z})^{-1}$. RLR minimizes

$$L_{\text{RLR}} = -\text{sum}\left(\log g(Y(X\mathbf{w} + b\mathbf{1}))\right) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{8}$$

[3] Again, we follow the derivation of [1], removing the $\frac{1}{l}$ term

By the Representer Theorem, we lose nothing by making the substitution, $\mathbf{w} = X^T\mathbf{c}$. Again, we define $K = XX^T$.

$$L_{\text{RLR}} = -\text{sum}\left(\log g(Y(K\mathbf{c} + b\mathbf{1}))\right) + \frac{\lambda}{2}\mathbf{c}^T K\mathbf{c}. \tag{9}$$

As with RLSC, we use gradient descent to minimize the objective. This entails computing the gradient,

$$\frac{\partial L}{\partial \mathbf{c}} = -g(-Y(K\mathbf{c} + b\mathbf{1}))YK + \lambda K\mathbf{c}, \tag{10}$$

$$\frac{\partial L}{\partial b} = -g(-Y(K\mathbf{c} + b\mathbf{1}))\mathbf{y}. \tag{11}$$

4 Summary

We have shown that other classification algorithms besides the SVM can be kernelized to learn non-linear decision boundaries. However, we have not discussed the computational feasibility. [1] discusses such issues in §3.7.

References

[1] Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning.* PhD thesis, Massachusetts Institute of Technology, 2002.

[2] Christopher J. C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.