Equation 5-12. Computing the bias term using the kernel trick

$$\begin{aligned}\hat{b} &= \frac{1}{n_s}\sum_{\substack{i=1\\ \hat{\alpha}^{(i)}>0}}^{m}\left(1 - t^{(i)}\hat{\mathbf{w}}^T \cdot \phi\left(\mathbf{x}^{(i)}\right)\right) = \frac{1}{n_s}\sum_{\substack{i=1\\ \hat{\alpha}^{(i)}>0}}^{m}\left(1 - t^{(i)}\left(\sum_{j=1}^{m}\hat{\alpha}^{(j)}t^{(j)}\phi\left(\mathbf{x}^{(j)}\right)\right)^T \cdot \phi\left(\mathbf{x}^{(i)}\right)\right)\\ &= \frac{1}{n_s}\sum_{\substack{i=1\\ \hat{\alpha}^{(i)}>0}}^{m}\left(1 - t^{(i)}\sum_{\substack{j=1\\ \hat{\alpha}^{(j)}>0}}^{m}\hat{\alpha}^{(j)}t^{(j)}K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\right)\end{aligned}$$

If you are starting to get a headache, it's perfectly normal: it's an unfortunate side effects of the kernel trick.

Online SVMs

Before concluding this chapter, let's take a quick look at online SVM classifiers (recall that online learning means learning incrementally, typically as new instances arrive).

For linear SVM classifiers, one method is to use Gradient Descent (e.g., using `SGDClassifier`) to minimize the cost function in Equation 5-13, which is derived from the primal problem. Unfortunately it converges much more slowly than the methods based on QP.

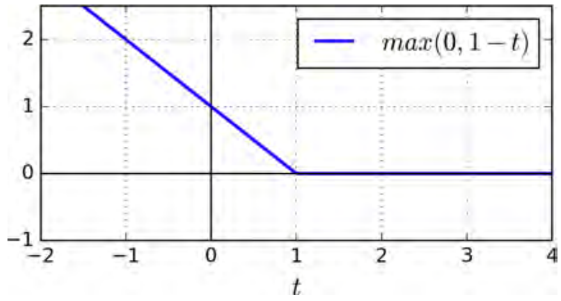
Equation 5-13. Linear SVM classifier cost function

$$J(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} \quad + \quad C\sum_{i=1}^{m} max\left(0, 1 - t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right)\right)$$

The first sum in the cost function will push the model to have a small weight vector **w**, leading to a larger margin. The second sum computes the total of all margin violations. An instance's margin violation is equal to 0 if it is located off the street and on the correct side, or else it is proportional to the distance to the correct side of the street. Minimizing this term ensures that the model makes the margin violations as small and as few as possible

Hinge Loss

The function $max(0, 1 – t)$ is called the *hinge loss* function (represented below). It is equal to 0 when $t \geq 1$. Its derivative (slope) is equal to –1 if $t < 1$ and 0 if $t > 1$. It is not differentiable at $t = 1$, but just like for Lasso Regression (see "Lasso Regression" on page 130) you can still use Gradient Descent using any *subderivative* at $t = 0$ (i.e., any value between –1 and 0).



It is also possible to implement online kernelized SVMs—for example, using "Incremental and Decremental SVM Learning"[7] or "Fast Kernel Classifiers with Online and Active Learning."[8] However, these are implemented in Matlab and C++. For large-scale nonlinear problems, you may want to consider using neural networks instead (see Part II).

Exercises

1. What is the fundamental idea behind Support Vector Machines?
2. What is a support vector?
3. Why is it important to scale the inputs when using SVMs?
4. Can an SVM classifier output a confidence score when it classifies an instance? What about a probability?
5. Should you use the primal or the dual form of the SVM problem to train a model on a training set with millions of instances and hundreds of features?
6. Say you trained an SVM classifier with an RBF kernel. It seems to underfit the training set: should you increase or decrease γ (`gamma`)? What about `C`?
7. How should you set the QP parameters (**H**, **f**, **A**, and **b**) to solve the soft margin linear SVM classifier problem using an off-the-shelf QP solver?
8. Train a `LinearSVC` on a linearly separable dataset. Then train an `SVC` and a `SGDClassifier` on the same dataset. See if you can get them to produce roughly the same model.
9. Train an SVM classifier on the MNIST dataset. Since SVM classifiers are binary classifiers, you will need to use one-versus-all to classify all 10 digits. You may

7 "Incremental and Decremental Support Vector Machine Learning," G. Cauwenberghs, T. Poggio (2001).

8 "Fast Kernel Classifiers with Online and Active Learning," A. Bordes, S. Ertekin, J. Weston, L. Bottou (2005).