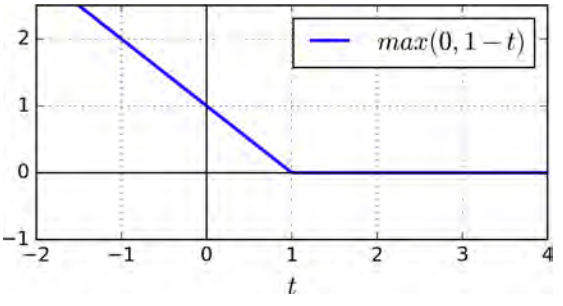

It is also possible to implement online kernelized SVMs—for example, using "Incremental and Decremental SVM Learning"[7] or "Fast Kernel Classifiers with Online and Active Learning."[8] However, these are implemented in Matlab and C++. For large-scale nonlinear problems, you may want to consider using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

want to tune the hyperparameters using small validation sets to speed up the process. What accuracy can you reach?

10. Train an SVM regressor on the California housing dataset.

Solutions to these exercises are available in Appendix A.

CHAPTER 6
Decision Trees

Like SVMs, *Decision Trees* are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks. They are very powerful algorithms, capable of fitting complex datasets. For example, in Chapter 2 you trained a `DecisionTreeRegressor` model on the California housing dataset, fitting it perfectly (actually overfitting it).

Decision Trees are also the fundamental components of Random Forests (see Chapter 7), which are among the most powerful Machine Learning algorithms available today.

In this chapter we will start by discussing how to train, visualize, and make predictions with Decision Trees. Then we will go through the CART training algorithm used by Scikit-Learn, and we will discuss how to regularize trees and use them for regression tasks. Finally, we will discuss some of the limitations of Decision Trees.

Training and Visualizing a Decision Tree

To understand Decision Trees, let's just build one and take a look at how it makes predictions. The following code trains a `DecisionTreeClassifier` on the iris dataset (see Chapter 4):

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
```