```
weighted_score = accuracy_score(train_y, train_y_pred, sample_weight=train_w)
print("Weighted train Classification Accuracy: %f" % weighted_score)
weighted_score = accuracy_score(valid_y, valid_y_pred, sample_weight=valid_w)
print("Weighted valid Classification Accuracy: %f" % weighted_score)
weighted_score = accuracy_score(test_y, test_y_pred, sample_weight=test_w)
print("Weighted test Classification Accuracy: %f" % weighted_score)
```

Here `train_X`, `train_y`, and so on are the Tox21 datasets defined in the previous chapter. Recall that all these quantities are NumPy arrays. `n_estimators` refers to the number of decision trees in our forest. Setting 50 or 100 trees often provides decent performance. Scikit-learn offers a simple object-oriented API with `fit(X, y)` and `predict(X)` methods. This model achieves the following accuracy with respect to our weighted accuracy metric:

```
Weighted train Classification Accuracy: 0.989845
Weighted valid Classification Accuracy: 0.681413
```

Recall that the fully connected network from Chapter 4 achieved performance:

```
Train Weighted Classification Accuracy: 0.742045
Valid Weighted Classification Accuracy: 0.648828
```

It looks like our baseline gets greater accuracy than our deep learning model! Time to roll up our sleeves and get to work.

Graduate Student Descent

The simplest method to try good hyperparameters is to simply try a number of different hyperparameter variants manually to see what works. This strategy can be surprisingly effective and educational. A deep learning practitioner needs to build up intuition about the structure of deep networks. Given the very weak state of theory, empirical work is the best way to learn how to build deep learning models. We highly recommend trying many variants of the fully connected model yourself. Be systematic; record your choices and results in a spreadsheet and systematically explore the space. Try to understand the effects of various hyperparameters. Which make network training proceed faster and which slower? What ranges of settings completely break learning? (These are quite easy to find, unfortunately.)

There are a few software engineering tricks that can make this search easier. Make a function whose arguments are the hyperparameter you wish to explore and have it print out the accuracy. Then trying new hyperparameter combinations requires only a single function call. Example 5-2 shows what this function signature would look like for our fully connected network from the Tox21 case study.

Example 5-2. A function mapping hyperparameters to different Tox21 fully connected networks

```
def eval_tox21_hyperparams(n_hidden=50, n_layers=1, learning_rate=.001,
                           dropout_prob=0.5, n_epochs=45, batch_size=100,
                           weight_positives=True):
```

Let's walk through each of these hyperparameters. `n_hidden` controls the number of neurons in each hidden layer of the network. `n_layers` controls the number of hidden layers. `learning_rate` controls the learning rate used in gradient descent, and `dropout_prob` is the probability neurons are not dropped during training steps. `n_epochs` controls the number of passes through the total data and `batch_size` controls the number of datapoints in each batch.

`weight_positives` is the only new hyperparameter here. For unbalanced datasets, it can often be helpful to weight examples of both classes to have equal weight. For the Tox21 dataset, DeepChem provides weights for us to use. We simply multiply the per-example cross-entropy terms by the weights to perform this weighting (Example 5-3).

Example 5-3. Weighting positive samples for Tox21

```
entropy = tf.nn.sigmoid_cross_entropy_with_logits(logits=y_logit, labels=y_expand)
# Multiply by weights
if weight_positives:
  w_expand = tf.expand_dims(w, 1)
  entropy = w_expand * entropy
```

Why is the method of picking hyperparameter values called graduate student descent? Machine learning, until recently, has been a primarily academic field. The tried-and-true method for designing a new machine learning algorithm has been describing the method desired to a new graduate student, and asking them to work out the details. This process is a bit of a rite of passage, and often requires the student to painfully try many design alternatives. On the whole, this is a very educational experience, since the only way to gain design aesthetic is to build up a memory of settings that work and don't work.

Grid Search

After having tried a few manual settings for hyperparameters, the process will begin to feel very tedious. Experienced programmers will be tempted to simply write a `for` loop that iterates over the choices of hyperparameters desired. This process is more or less the grid-search method. For each hyperparameter, pick a list of values that might be good hyperparameters. Write a nested `for` loop that tries all combinations of these values to find their validation accuracies, and keep track of the best performers.