Tox21 has more datasets than we will analyze here, so we need to remove the labels associated with these extra datasets (Example 4-2).

Example 4-2. Remove extra datasets from Tox21

```
# Remove extra tasks
train_y = train_y[:, 0]
valid_y = valid_y[:, 0]
test_y = test_y[:, 0]
train_w = train_w[:, 0]
valid_w = valid_w[:, 0]
test_w = test_w[:, 0]
```

Accepting Minibatches of Placeholders

In the previous chapters, we created placeholders that accepted arguments of fixed size. When dealing with minibatched data, it is often convenient to be able to feed batches of variable size. Suppose that a dataset has 947 elements. Then with a minibatch size of 50, the last batch will have 47 elements. This would cause the code in Chapter 3 to crash. Luckily, TensorFlow has a simple fix to the situation: using `None` as a dimensional argument to a placeholder allows the placeholder to accept tensors with arbitrary size in that dimension (Example 4-3).

Example 4-3. Defining placeholders that accept minibatches of different sizes

```
d = 1024
with tf.name_scope("placeholders"):
  x = tf.placeholder(tf.float32, (None, d))
  y = tf.placeholder(tf.float32, (None,))
```

Note `d` is 1024, the dimensionality of our feature vectors.

Implementing a Hidden Layer

The code to implement a hidden layer is very similar to code we've seen in the last chapter for implementing logistic regression, as shown in Example 4-4.

Example 4-4. Defining a hidden layer

```
with tf.name_scope("hidden-layer"):
  W = tf.Variable(tf.random_normal((d, n_hidden)))
  b = tf.Variable(tf.random_normal((n_hidden,)))
  x_hidden = tf.nn.relu(tf.matmul(x, W) + b)
```

We use a `tf.name_scope` to group together introduced variables. Note that we use the matricial form of the fully connected layer. We use the form xW instead of Wx in