Grid Cross Validation

GridSearchCV is a way of systematically working through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance. The beauty is that it can work through many combinations in only a couple extra lines of code.

Here's an example from the sklearn documentation, which can be found here:

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svr = svm.SVC()
clf = grid_search.GridSearchCV(svr, parameters)
clf.fit(iris.data, iris.target)
```

Let's break this down line by line.

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
```

A dictionary of the parameters, and the possible values you want to try for them. In this case, they're playing around with the kernel (possible choices are 'linear' and 'rbf'), and C (possible choices are 1 and 10).

Then all the following (kernel, C) combinations are automatically generated: [('rbf', 1), ('rbf', 10), ('linear', 1), ('linear', 10)]. Each is used to train an SVM, and the performance is then assessed using cross-validation.

```
svr = svm.SVC()
```

This looks kind of like creating a classifier, just like we've been doing since the first lesson. But note that the "clf" isn't made until the next line--this is just saying what kind of algorithm to use. Another way to think about this is that the "classifier" isn't just the algorithm in this case, it's algorithm plus parameter values. Note that there's no monkeying around with the kernel or C; all that is handled in the next line.

```
clf = grid_search.GridSearchCV(svr, parameters)
```

This is where the first bit of magic happens; the classifier is being created. We pass the algorithm (*svr*) and the dictionary of parameters to try (*parameters*) and it generates a grid of parameter combinations to try.

```
clf.fit(iris.data, iris.target)
```

And the second bit of magic. The fit function now tries all the parameter combinations, and returns a fitted classifier that's automatically tuned to the optimal parameter combination. You can now access the parameter values via

```
clf.best_estimator_
```