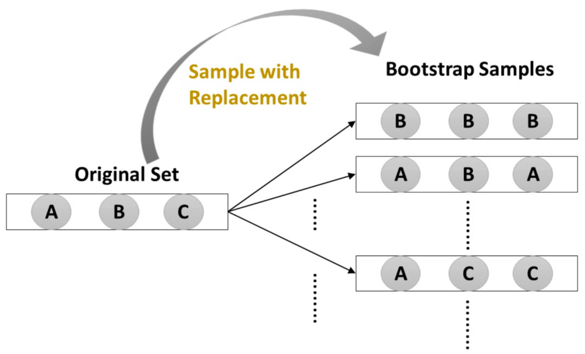
202~~0~~1/10/~~28~~20

Ensemble Methods: Bagging

One way to get a diverse set of classifiers is to use very different training algorithms, as we just described in the rationale of ensemble methods. Another approach is to use **the same training algorithm** for every feature, but to train them on different random subsets of the training set. This approach is called **bagging** (short for **bootstrap** (重複抽樣) **aggregating**) based on the **bootstrap**, a widely applicable and extremely powerful statistical tool.

The basic idea of **bootstrap** or **bootstrapping** is that inference about a population from sample data (sample → population) can be modelled by resampling the sample data and performing inference about a sample from resampled data (resampled → sample). In short, **bootstrap resamples the data with replacement as illustrated in the figure below**. When doing properly, we will have:

sample → population $\approx$ resampled → sample



1. Theoretical Minimum for Bagging

Bagging is a general-purpose procedure for **reducing the variance** of a statistical learning method. Basically, we resample the data with replacement and then train a classifier on the newly sampled data. Then, we combine the outputs of each of the individual classifiers using a majority-voting scheme or other similar schemes.

Let us consider a given a set of *n* **independent** (or uncorrelated) observations $Z_1, Z_2, \cdots, Z_n$, each with variance σ^2. The variance of the mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{n}$. **In other words, averaging a set of observations reduces variance.**

Hence a natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we could calculate $g^{(1)}(\mathbf{x})$, $g^{(2)}(\mathbf{x})$, …, $g^{(B)}(\mathbf{x})$ using B separate training sets, and average them in order to obtain a single low-variance statistical learning model,

$$g_{\text{avg}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} g^{(b)}(\mathbf{x})$$

Remark: $g_{\text{avg}}(\mathbf{x}) \approx \bar{g}(\mathbf{x})$ as we discussed in the note on bias and variance.

Fun Time: Is this (to take many training sets from the population) practical? (1) Yes (2) No

Instead, we can **bootstrap**, by taking repeated samples from the single training data set. In this approach we generate B different bootstrapped training data sets. We then train our method on the b^{th} bootstrapped training set in order to get $g^{(*b)}(x)$, and finally average all the predictions to obtain the *bootstrapped aggregated estimator* or *bagged estimator*:

$$g_{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} g^{(*b)}(\mathbf{x})$$

Remarks:

1. It is important to remember that g_{bag} is not exactly equal to g_{avg}. Thus g_{bag} might perform worse than the original estimator g.
2. The bagging procedure can be further enhanced by introducing random forests, which will be discussed next.

2. Python Example

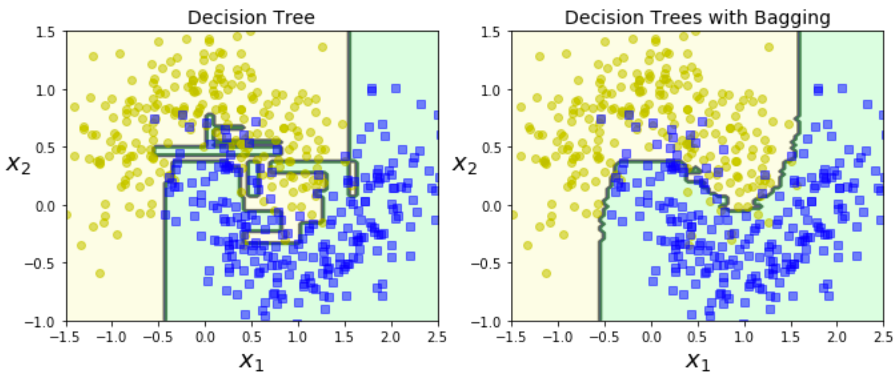
`Scikit-Learn` offers a simple API for bagging with the `BaggingClassifier` class (or `BaggingRegressor` for regression). The following code trains an ensemble of 500 Decision Tree classifiers, each trained on 100 training instances, randomly sampled from the 500 data generated from `make_moon` with replacement. The `n_jobs` parameter tells `Scikit-Learn` the number of CPU cores to use for training and predictions (`-1` tells `Scikit-Learn` to use all available cores):

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(random_state=42), n_estimators=500,
```

```
    max_samples=100, bootstrap=True, n_jobs=-1, random_state=42)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

Figure below compares the decision boundary of a single Decision Tree with the decision boundary of a bagging ensemble of 500 trees (from the preceding code).



Q: what have you observed?

A:

As you can see, the decision boundary from the bagging ensemble is much smooth (low variance) and ensemble's predictions will likely generalize much better than the single Decision Tree's predictions (`0.904 vs 0.856`).

You can download the complete source code `Bagging.ipynb` from the course website.