### Fingerprinting

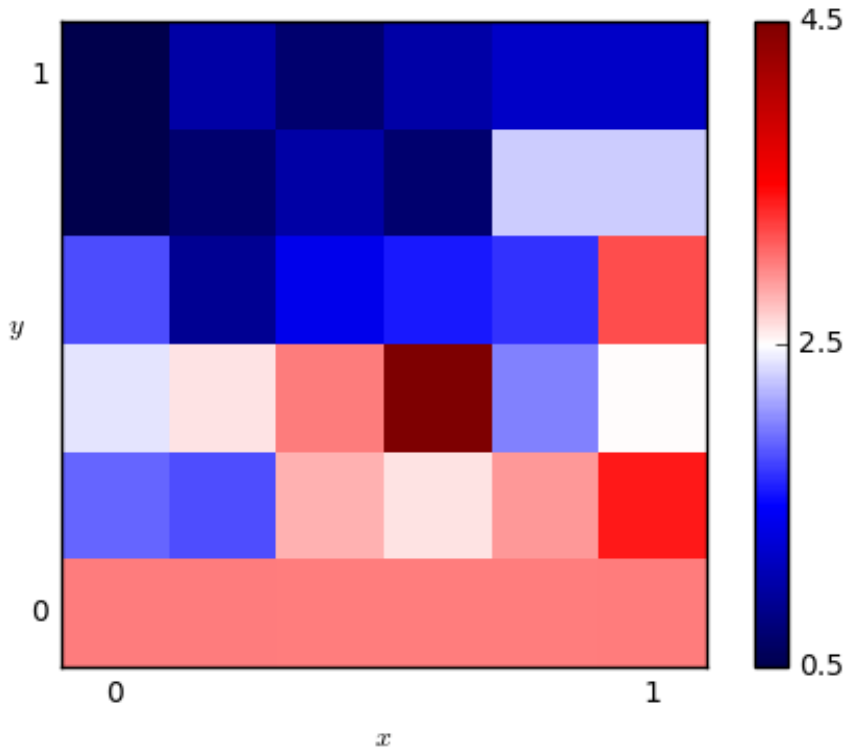
In *[Ashlock2008]*, *[Ashlock2009]* a methodology for obtaining visual representation of a strategy's behaviour is described. The basic method is to play the strategy against a probe strategy with varying noise parameters. These noise parameters are implemented through the `JossAnnTransformer`. The Joss-Ann of a strategy is a new strategy which has a probability `x` of cooperating, a probability `y` of defecting, and otherwise uses the response appropriate to the original strategy. We can then plot the expected score of the strategy against `x` and `y` and obtain a heat plot over the unit square. When `x + y >= 1` the `JossAnn` is created with parameters `(1-y, 1-x)` and plays against the Dual of the probe instead. A full definition and explanation is given in *[Ashlock2008]*, *[Ashlock2009]*.

Here is how to create a fingerprint of `WinStayLoseShift` using `TitForTat` as a probe:

```
>>> import axelrod as axl
>>> axl.seed(0)  # Fingerprinting is a random process
>>> strategy = axl.WinStayLoseShift
>>> probe = axl.TitForTat
>>> af = axl.AshlockFingerprint(strategy, probe)
>>> data = af.fingerprint(turns=10, repetitions=2, step=0.2)
>>> data
{...
>>> data[(0, 0)]
3.0
```

The `fingerprint` method returns a dictionary mapping coordinates of the form `(x, y)` to the mean score for the corresponding interactions. We can then plot the above to get:

```
>>> p = af.plot()
>>> p.show()
```

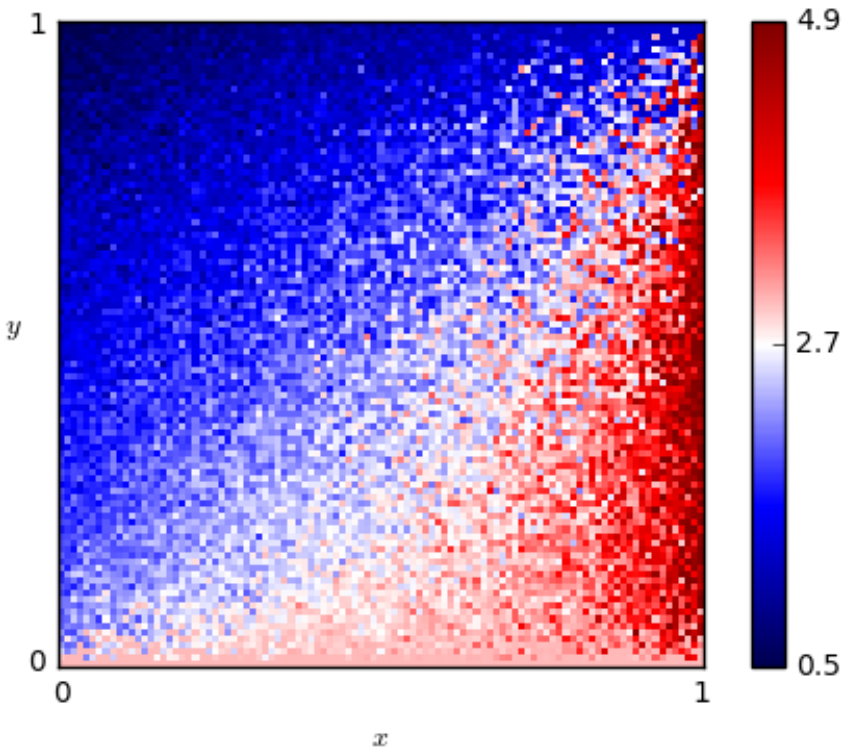



In reality we would need much more detail to make this plot useful.

Running the above with the following parameters:

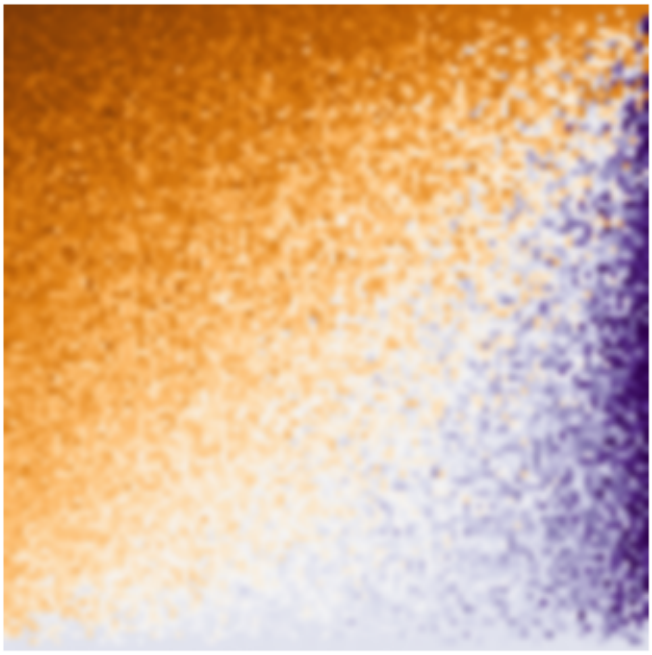
```
>>> af.fingerprint(turns=50, repetitions=2, step=0.01)
```

We get the plot:




We are also able to specify a matplotlib colour map, interpolation and can remove the colorbar and axis labels:

```
>>> p = af.plot(col_map='PuOr', interpolation='bicubic', colorbar=False, labels=False)
>>> p.show()
```



Note that it is also possible to pass a player instance to be fingerprinted and/or as a probe. This allows for the fingerprinting of parametrized strategies:

```
>>> axl.seed(0)
>>> player = axl.Random(.1)
>>> probe = axl.GTFT(.9)
>>> af = axl.AshlockFingerprint(player, probe)
>>> data = af.fingerprint(turns=10, repetitions=2, step=0.2)
>>> data
{...
>>> data[(0, 0)]
4.4...
```

Ashlock's fingerprint is currently the only fingerprint implemented in the library.

## Further capabilities in the library

This section shows some of the more intricate capabilities of the library.

Contents:

### Accessing strategies

All of the strategies are accessible from the main name space of the library. For example:

```
>>> import axelrod as axl
>>> axl.TitForTat()
Tit For Tat
```