The shift fallacy

The shift fallacy is as follows. We fit two models `m` and `m_shift` with data-weights `one` (the all ones vector) and `a * (one - y) + b * y` (`y` being the dependent variable). We are re-sampling according to outcome, a (not always advisable) technique popular with some for un-balanced classification problems (note: we think this technique is popular due to the common error of using classification rules for classification problems) . Then the fallacy is to (falsely) believed the two models differ only in the intercept term.

This is easy to disprove in R.

```
library(wrapr)

# build our example data
# modeling y as a function of x1 and x2 (plus intercept)

d <- wrapr::build_frame(
  "x1"  , "x2", "y" |
    0   , 0   , 0   |
    0   , 0   , 0   |
    0   , 1   , 1   |
    1   , 0   , 0   |
    1   , 0   , 0   |
    1   , 0   , 1   |
    1   , 1   , 0   )

knitr::kable(d)
```

x1	x2	y
0	0	0
0	0	0
0	1	1
1	0	0
1	0	0
1	0	1
1	1	0

First we fit the model with each data-row having the same weight.

```
m <- glm(
  y ~ x1 + x2,
  data = d,
  family = binomial())

m$coefficients
```

```
## (Intercept)          x1          x2
##  -1.2055937  -0.3129307   1.3620590
```

Now we build a balanced weighting. We are up-sampling both classes so we don't have any

fractional weights (fractional weights are fine, but they trigger a warning in `glm()`).

```
w <- ifelse(d$y == 1, sum(1 - d$y), sum(d$y))
w
```

```
## [1] 2 2 5 2 2 5 2
```

```
# confirm prevalence is 0.5 under this weighting
sum(w * d$y) / sum(w)
```

```
## [1] 0.5
```

Now we fit the model for the balanced data situation.

```
m_shift <- glm(
  y ~ x1 + x2,
  data = d,
  family = binomial(),
  weights = w)

m_shift$coefficients
```

```
## (Intercept)          x1          x2
##  -0.5512784   0.1168985   1.4347723
```

Notice that all of the coefficients changed, not just the intercept term. And we have thus demonstrated the shift fallacy.

The balance fallacy

An additional point is: the simple model without re-weighting is the better model on this training data. There appears to be an industry belief that to work with unbalanced classes one *must* re-balance the data. In fact moving to "balanced data" doesn't magically improve the model quality, what it *does* is helps hide *some* of the bad consequences of using classification rules instead of probability models (please see here for some discussion).

For instance our original model has the following statistical deviance (lower is better):

```
deviance <- function(prediction, truth) {
  -2 * sum(truth * log(prediction) + (1 - truth) * log(1 - prediction))
}

deviance(
  prediction = predict(m, newdata = d, type = 'response'),
  truth = d$y)
```

```
## [1] 7.745254
```

And our balanced model has a worse deviance.

```
deviance(
  prediction = predict(m_shift, newdata = d, type = 'response'),
  truth = d$y)
```

```
## [1] 9.004022
```

Part of this issue is that the balanced model is scaled wrong. It's average prediction is, by design, inflated.

```
mean(predict(m_shift, newdata = d, type = 'response'))
```

```
## [1] 0.4784371
```

Whereas, the original model average to the same as the average of the truth values (a property of logistic regression).

```
mean(predict(m, newdata = d, type = 'response'))
```

```
## [1] 0.2857143
```

```
mean(d$y)
```

```
## [1] 0.2857143
```

So let's adjust the balanced predictions back to the correct expected value (essentially Platt scaling).

```
d$balanced_pred <- predict(m_shift, newdata = d, type = 'link')

m_scale <- glm(
  y ~ balanced_pred,
  data = d,
  family = binomial())

corrected_balanced_pred <- predict(m_scale, newdata = d, type =
'response')

mean(corrected_balanced_pred)
```

```
## [1] 0.2857143
```

We now have a prediction with the correct expected value. However, notice this deviance is *still* larger than the simple un-weighted original model.

```
deviance(
  prediction = corrected_balanced_pred,
  truth = d$y)
```

```
## [1] 7.803104
```

Our opinion is: re-weighting or re-sampling data for a logistic regression is pointless. The fitting procedure deals with un-balanced data quite well, and doesn't need any attempt at help. We think this sort of re-weighting and re-sampling introduces complexity, the possibility of data-leaks with up-sampling, and a loss of statistical efficiency with down-sampling. Likely the re-sampling fallacy is driven by a need to move model scores to near `0.5` when using `0.5` as a default *classification rule* threshold (which we argue against in "Don't Use Classification Rules for Classification Problems"). This is a problem that is more easily avoided by insisting on a probability model over a classification rule.

Conclusion

Some tools, such as logistic regression, work best on training data that accurately represents

the distributions facts of problem, and do not require artificially balanced training data. Also, re-balancing training data is a bit more involved than one might think, as we see more than just the intercept term changes when we re-balance data.

Take logistic regression as the entry level probability model for classification problems. If it doesn't need data re-balancing then other any tool claiming to be *universally better* than it *should* also not need artificial re-balancing (though if they are internally using classification rule metrics, some hyper-parameters or internal procedures may need to be adjusted).

Prevalence re-balancing *is* working around mere operational issues: such as using classification rules (instead of probability models), using sub-optimal metrics (such as accuracy). However, there operational issues are better directly corrected than worked around. A lot of the complexity we see in modern machine learning pipelines is patches patching unwanted effects of previous patches.