For a few of my commercial projects I have been in the seemingly strange place being asked to port a linear model from one data science system to another. Now I try to emphasize that it is better going forward to port procedures and build new models with training data. But sometimes that is not possible. Solving this problem for linear and logistic models is a fun mathematics exercise.

Our task is to take a linear model in one system and stand up a replicant of it in another system. We take the phrase “replicating a model” with an eye to the finance term “[replicating portfolio](https://en.wikipedia.org/wiki/Replicating_portfolio).” This different than the sense of the word in “reproducible research” (pertaining more the ability to re-run work reliably).

This technique can be useful in moving an existing model into a large data system such as [Apache Spark](https://spark.apache.org/) or [Google BigQuery](https://cloud.google.com/bigquery/).

Let’s work an example in [R](https://www.r-project.org/). Suppose we are working with a [linear regression model](https://en.wikipedia.org/wiki/Linear_model#Linear_regression_models) and from our donor system we have extracted the following representation of the model as “intercept” and “betas”.

intercept <- 3

betas <- c(weight = 2, height = 4)

Our goal is to build a linear regression model that has the above coefficients. The way we are going to do this is by building our own synthetic data set such that the regression fit through this data set yields these coefficients.

To do this we will build one data row for the intercept and one for each of the coefficients. Our data set will represent a dependent variable “y” based on the input variables “weight” and “height”.

n\_vars <- length(betas)

d <- as.data.frame(matrix(0,

nrow = n\_vars + 1,

ncol = n\_vars + 1))

colnames(d) <- c(names(betas), "y")

rownames(d) <- c("intercept", names(betas))

d["intercept", "y"] = intercept

This gives us the following partially filled out data.frame.

knitr::kable(d)

|  | **weight** | **height** | **y** |
| --- | --- | --- | --- |
| intercept | 0 | 0 | 3 |
| weight | 0 | 0 | 0 |
| height | 0 | 0 | 0 |

We then fill in the rows corresponding to betas with a 1 in the corresponding ni-th beta-column and “y” equal to intercept + beta[[ni]]. This is saying y = intercept + beta[[ni]].

for(ni in names(betas)) {

d[ni, ni] <- 1

d[ni, "y"] <- intercept + betas[[ni]]

}

This gives us the following finished synthetic data set.

knitr::kable(d)

|  | **weight** | **height** | **y** |
| --- | --- | --- | --- |
| intercept | 0 | 0 | 3 |
| weight | 1 | 0 | 5 |
| height | 0 | 1 | 7 |

The idea is: a linear model with our original intercept and coefficients equal to our betas is the unique linear model that matches the y-values we have specified. Let’s confirm this.

model <- lm(y ~ weight + height,

data = d)

print(coefficients(model))

## (Intercept) weight height

## 3 2 4

Notice this model has exactly the coefficient we started with. We have solved our linear model replication problem. For a follow-up problem: suppose we wanted a [logistic regrission](http://www.win-vector.com/blog/2011/09/the-simpler-derivation-of-logistic-regression/) model with the given coefficients. How would we go about that?

Our idea for that is as follows. First we need a data frame that has twice as many rows as our original did.

d2\_0 <- d

rownames(d2\_0) <- paste0(rownames(d2\_0), "\_0")

d2\_0$y <- 0

d2\_1 <- d

rownames(d2\_1) <- paste0(rownames(d2\_1), "\_1")

d2\_1$y <- 1

d2 <- rbind(d2\_0, d2\_1)

This gives us the following data set.

knitr::kable(d2)

|  | **weight** | **height** | **y** |
| --- | --- | --- | --- |
| intercept\_0 | 0 | 0 | 0 |
| weight\_0 | 1 | 0 | 0 |
| height\_0 | 0 | 1 | 0 |
| intercept\_1 | 0 | 0 | 1 |
| weight\_1 | 1 | 0 | 1 |
| height\_1 | 0 | 1 | 1 |

Now we are going use the weights to encode a relation in the data that forces a logistic regression to replicate the weights we want. The idea each row should be in our training set with a weight or mass proportional to to the logistic predicted probability for rows with y==1, and one minus the logistic predicted probability for rows with y==0.

What we are saying is: to replicate the logistic model we need evaluations of that model for these rows. Lets calculate those for our desired coefficients as follows.

d2$py <- intercept

for(ni in names(betas)) {

d2$py <- d2$py + betas[[ni]]\*d2[[ni]]

}

d2$py <- 1/(1+exp(-d2$py))

The last step is the traditional sigmoid transform that translates from link-space to probabilities for the logistic model. We said we wanted 1-py in the rows where y is zero, so we make that adjustment.

d2$py[d2$y==0] <- 1 - d2$py[d2$y==0]

This gives us the following data set.

knitr::kable(d2)

|  | **weight** | **height** | **y** | **py** |
| --- | --- | --- | --- | --- |
| intercept\_0 | 0 | 0 | 0 | 0.0474259 |
| weight\_0 | 1 | 0 | 0 | 0.0066929 |
| height\_0 | 0 | 1 | 0 | 0.0009111 |
| intercept\_1 | 0 | 0 | 1 | 0.9525741 |
| weight\_1 | 1 | 0 | 1 | 0.9933071 |
| height\_1 | 0 | 1 | 1 | 0.9990889 |

If we fit a logistic regression model on this data *with each row weighted by the py column* we recover the desired coefficients as we see below.

model2 <- glm(y ~ weight + height,

data = d2,

weights = d2$py,

family = binomial)

## Warning in eval(family$initialize): non-integer #successes in a binomial

## glm!

print(coefficients(model2))

## (Intercept) weight height

## 3 2 4

print(class(model2))

## [1] "glm" "lm"

model2 is now a logistic generalized linear model with the desired coefficients.

And that is how you replicate a model when you have access to the original model coefficients, but don’t have training data.