Momentum in language change: a model of self-actuating s-shaped curves

Language Dynamics and Change 6 (2): 171-198

Results and figures from the paper

Exponentially weighted moving average (EWMA) dynamics

```
# x is the old EWMA value, y the new datapoint, alpha the weight of new
# points
ewma <- function(x, y, alpha) alpha * y + (1 - alpha) * x
```

The weight of a datum point Y_{t-i} towards the current average X_t is $\alpha \cdot (1-\alpha)^{i-1}$. The weight omitted by stopping after k terms is $(1-\alpha)^k$.

```
ewmadev <- function(alpha, t, n0 = 0, target = 1, returnafter = Inf) {
    n0weight <- (1 - alpha)^t
    dev <- n0 * n0weight + target * (1 - n0weight)

    returning <- which(t >= returnafter)
    if (length(returning)) {
        ninterrupt <- n0 * (1 - alpha)^returnafter + target * (1 - (1 - alpha)^returnafter)
        dev[returning] <- ewmadev(alpha, t[returning] - returnafter, n0 = ninterrupt,
            target = n0)
    }
    return(dev)
}

# timestep/iteration at which the difference between two EWMAs with rates
# alpha and gamma and a constant target value is greatest
stepstomaxdifference <- function(alpha, gamma) log(alpha/gamma)/(alpha - gamma)

# calculate the maximum possible difference between the two EWMAs
expdecay <- function(alpha, t) exp(-alpha * t)

maxdecaydifference <- function(alpha, gamma) {
    t <- stepstomaxdifference(alpha, gamma)
    expdecay(alpha, t) - expdecay(gamma, t)
}

# normalise bias strength b according to the amplitude of the momentum term
normaliseb <- function(b, alpha, gamma) b/maxdecaydifference(alpha, gamma)

# auxiliary function: truncate values of x that exceed the [0,1] range
limit <- function(x) pmin(1, pmax(x, 0))
```

Figure 2

```r
ALPHA <- 2
GAMMA <- 3
MOMENTUM <- 1
MCOLOR <- gray(0.6)

addgammaanddifference <- function(alpha, gamma, glty = GAMMA, mlty = MOMENTUM,
    ...) {
    curve(ewmadev(gamma, x, ...), lty = glty, add = TRUE)
    curve(0.5 + (ewmadev(gamma, x, ...) - ewmadev(alpha, x, ...))/2, lty = mlty,
        col = MCOLOR, add = TRUE)
    # addmaxdifferenceinfo(alpha, gamma)
}
addalphaanddifference <- function(alpha, gamma, alty = ALPHA, mlty = MOMENTUM,
    ...) {
    curve(ewmadev(alpha, x, ...), lty = alty, add = TRUE)
    curve(0.5 + (ewmadev(gamma, x, ...) - ewmadev(alpha, x, ...))/2, lty = mlty,
        col = MCOLOR, add = TRUE)
    # addmaxdifferenceinfo(alpha, gamma)
}

decaydifferencecurve <- function(alpha, gamma, maxt = if (!add) 2 * stepstomaxdifference(alpha,
    gamma), add = FALSE, ylab = expression("EWMA value " ~ hat(n)[gamma]), alty = ALPHA,
    glty = GAMMA, mlty = MOMENTUM, ylim = 0:1, plotm = TRUE, mlab = expression(hat(n)[gamma] -
        hat(n)[0.15]), main = "", ...) {
    # 'difference between two EWMAs'
    curve(ewmadev(alpha, x, ...), lty = alty, to = maxt, xlab = "number of data points received",
        ylab = ylab, ylim = ylim, add = add, main = main)
    curve(ewmadev(gamma, x, ...), lty = glty, add = TRUE)
    if (plotm) {
        curve(0.5 + (ewmadev(gamma, x, ...) - ewmadev(alpha, x, ...))/2, lty = mlty,
            col = MCOLOR, add = TRUE)
        addmomentumaxis(col = MCOLOR, mlab = mlab)
    }
    # addmaxdifferenceinfo(alpha, gamma)
}

tightmargin <- function(...) par(mgp = c(2, 1, 0), mar = c(3, 3.3, 2, 0.8),
    font.main = 1, ...)  # b l t r

tightmargin(mfcol = c(2, 2), pty = "s", xaxs = "i", yaxs = "i")
alpha <- 0.01
gammas <- c(0.02, 0.05, 0.15)
ltys <- 5:3

decaydifferencecurve(alpha, gammas[1], alty = 1, glty = ltys[1], mlty = ltys[1],
    plotm = FALSE, main = "(a.i)")

for (i in 2:length(gammas)) {
    curve(ewmadev(gammas[i], x), lty = ltys[i], add = TRUE)
}
returnafter <- Inf
```

```r
for (i in 1:length(gammas)) {
    curve(ewmadev(gammas[i], x, returnafter = returnafter) - ewmadev(alpha,
        x, returnafter = returnafter), lty = ltys[i], to = 2 * stepstomaxdifference(alpha,
        gammas[1]), xlab = "number of data points received", ylab = expression("EWMA difference " ~
        hat(n)[gamma] - hat(n)[alpha]), ylim = c(-1, 1), add = i != 1, main = if (i ==
        1)
        "(b.i)")
}
abline(h = 0, lty = 1)

returnafter <- 60
decaydifferencecurve(alpha, gammas[1], returnafter = returnafter, alty = 1,
    glty = ltys[1], plotm = FALSE, main = "(a.ii)")
for (i in 2:length(gammas)) {
    curve(ewmadev(gammas[i], x, returnafter = returnafter), lty = ltys[i], add = TRUE)
    # addgammaanddifference(alpha, gammas[i], returnafter=returnafter,
    # glty=ltys[i], plotm=FALSE)
}

for (i in 1:length(gammas)) {
    curve(ewmadev(gammas[i], x, returnafter = returnafter) - ewmadev(alpha,
        x, returnafter = returnafter), lty = ltys[i], to = 2 * stepstomaxdifference(alpha,
        gammas[1]), xlab = "number of data points received", ylab = expression("EWMA difference " ~
        hat(n)[gamma] - hat(n)[alpha]), ylim = c(-1, 1), add = i != 1, main = if (i ==
        1)
        "(b.ii)")
}
abline(h = 0, lty = 1)
```

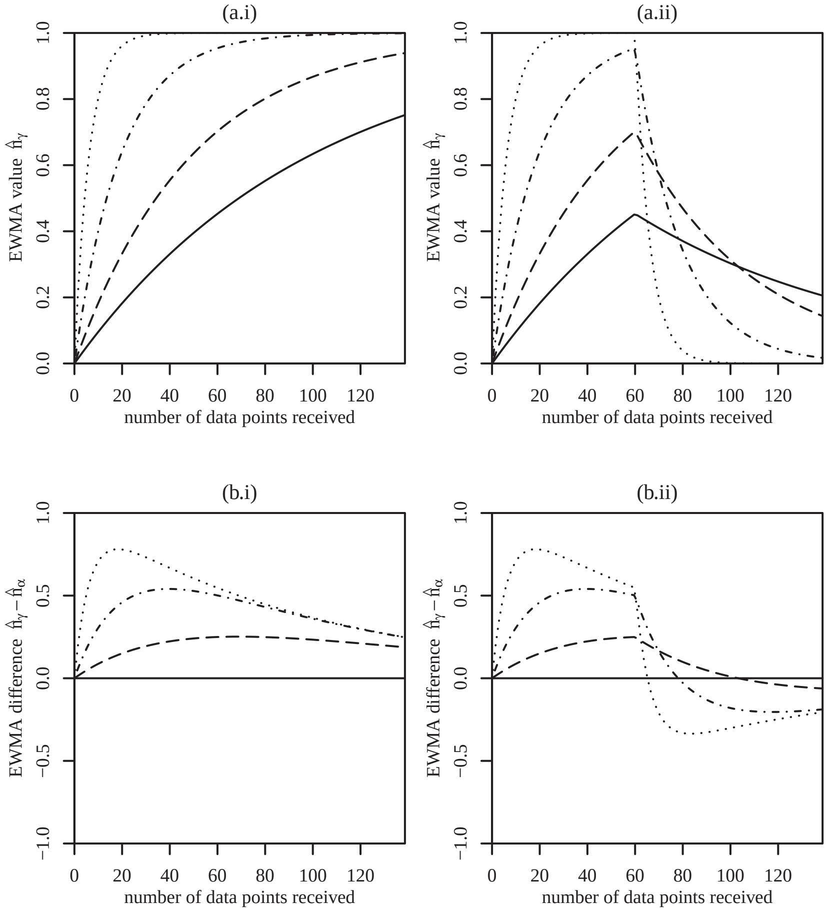
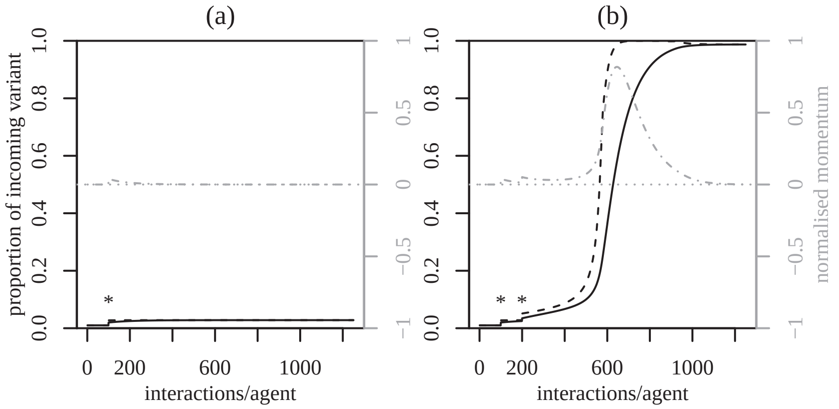


Figure 3

```
# idealised deterministic feedback loop
feedbackloopfiniteT <- function(alpha = 0.01, gamma = 0.02, T = 5, b = 2, noutliers = 1,
    x0 = 0.01, maxt = 1250) {
    normalisedb <- normaliseb(b, alpha, gamma)
    # row 1 = x, row 2 = ewma, row 3 = y (biased target)
    pop <- matrix(nrow = 3, ncol = maxt)
    pop[, 1] <- c(x0, x0, x0)
    for (i in 2:maxt) {
        # fabricate several outliers 100 timesteps in if (i >= 100 && i <
        # 100+noutliers) { fabricate outliers every 100 timesteps
        if (i%%100 == 0 && i%/%100 <= noutliers) {
            pop[3, i] <- NA
        } else {
            # calculate analytical mean of biased self-samples
            pop[3, i] <- weighted.mean(c(0, limit(1:(T - 1)/T + normalisedb *
                (pop[2, i - 1] - pop[1, i - 1])), 1), dbinom(0:T, T, pop[1,
                i - 1]))
        }
        pop[1, i] <- ewma(pop[1, i - 1], if (is.na(pop[3, i]))
            1 else pop[3, i], alpha)
        pop[2, i] <- ewma(pop[2, i - 1], if (is.na(pop[3, i]))
            1 else pop[3, i], gamma)
    }
    # calculate normalised momentum
    pop[2, ] <- (pop[2, ] - pop[1, ])/maxdecaydifference(alpha, gamma)
    pop
}

plotfeedbackloop <- function(data, ylab = "proportion of incoming variant",
    col = MCOLOR, mlab = if (is.na(ylab)) "normalised momentum" else NA, ...) {

    plot(data[1, ], type = "l", yaxs = "i", ylim = 0:1, xlab = "interactions/agent",
        ylab = ylab, ...)

    # plot (biased) target
    if (dim(data)[1] > 2)
        lines(data[3, ], lty = 2)
    # mark where fabricated datapoints were added
    points(which(is.na(data[3, ])), rep(0.1, length(which(is.na(data[3, ])))),
        pch = "*")

    # plot dotted line to show momentum zero point
    abline(h = 0.5, lty = 3, col = col)
    par(new = T)

    plot(data[2, ], type = "l", col = col, yaxs = "i", lty = 4, axes = F, xlab = NA,
        ylab = NA, ylim = c(-1, 1))
    axis(4, c(-1, -0.5, 0, 0.5, 1), c(-1, -0.5, 0, 0.5, 1), col = col, col.axis = col)
    if (!is.null(mlab))
        mtext(mlab, side = 4, line = 2, col = MCOLOR, cex = par("cex"))
}

# adjust outer margin to fit in 2nd y axis label
doublemargin <- function(...) tightmargin(oma = c(0, 0, 0, 2), ...)  # b l t r

doublemargin(pty = "s", mfrow = 1:2)
plotfeedbackloop(feedbackloopfiniteT(noutliers = 1), main = "(a)")
plotfeedbackloop(feedbackloopfiniteT(noutliers = 2), main = "(b)", ylab = NA)
```


(a)
(b)
proportion of incoming variant
*
*
*
interactions/agent
normalised momentum