Using automatic derivatives to predict a ping pong match winner

Alex Fun
github.com/alexfun/AD_ping_pong

2018-10-10

Part 1: Optimisation


APPROVED
Mortgage
Application Form

CommonwealthBank

ANZ

Westpac

ANZ

wealthBank

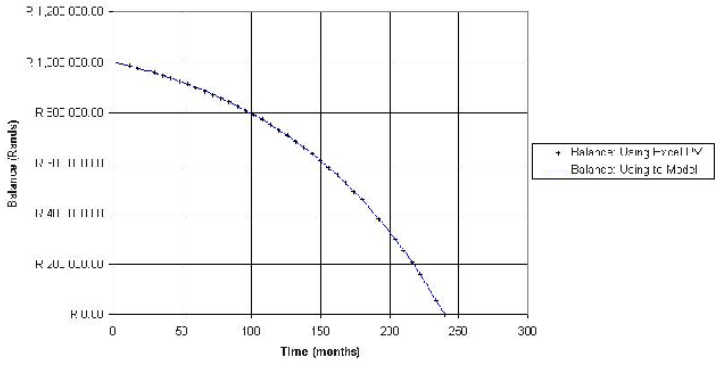
Westpac

ANZ

wealthBank

Westpac

Mortgage Balance Comparision



wealthBank

ANZ

Westpac

wealthBank

ANZ

Westpac

Optimisation

- Objective function (e.g. utility/happiness or log-likelihood) and decision variables (e.g. which bank/mortgage terms/etc or regression coefficients).
- Gradient based vs non-gradient based (not relevant here).

Optimisation: Newton's method

Suppose one wishes to find the value of (the decision variables) x such that $y = f(x)$ is maximised. Start with an initial guess x_0 for the optimal values, and iteratively build better guesses $x_1, x_2, \ldots$, via:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

Optimisation: Newton's method

Suppose one wishes to find the value of (the decision variables) x such that $y = f(x)$ is maximised. Start with an initial guess x_0 for the optimal values, and iteratively build better guesses $x_1, x_2, \ldots$, via:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

If we had a vector of decision variables $\mathbf{x}$ instead:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathbf{H}f(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n)$$

(https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization)

Optimisation: Newton's method

Nabla is the gradient and **H** is the hessian:

$$\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_m} \right)$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_m \, \partial x_1} & \frac{\partial^2 f}{\partial x_m \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_m^2} \end{bmatrix}$$

Optimisation: Newton's method

Nabla is the gradient and **H** is the hessian:

$$\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_m} \right)$$

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\, \partial x_m} \\ \dfrac{\partial^2 f}{\partial x_2\, \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\, \partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_m\, \partial x_1} & \dfrac{\partial^2 f}{\partial x_m\, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_m^2} \end{bmatrix}$$

Wow, much derivatives to find! (LaTeX from Wikipedia)

Calculating derivatives: Analytically

- Logistic regression:

$$\log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots$$

Calculating derivatives: Analytically

- Logistic regression:

$$\log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots$$

- Log-likelihood:

$$\ell = \sum_{i=1}^{n} y_i \log(\frac{p_i}{1-p_i}) + \sum_{i=1}^{n} \log(1-p_i)$$

Calculating derivatives: Analytically

- Logistic regression:

$$\log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots$$

- Log-likelihood:

$$\ell = \sum_{i=1}^{n} y_i \log(\frac{p_i}{1-p_i}) + \sum_{i=1}^{n} \log(1-p_i)$$

- After some algebra (or consultation of https://stats.stackexchange.com/a/16541):

$$\begin{aligned}\frac{\partial \ell}{\partial \beta_j} &= \dots \\ &= \sum_i y_i x_{ij} - \sum_i p_i x_{ij},\end{aligned}$$

$$\frac{\partial^2 \ell}{\partial \beta_j \partial \beta_k} = \dots$$

Calculating derivatives: Analytically

- Logistic regression:

$$\log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots$$

- Log-likelihood:

$$\ell = \sum_{i=1}^{n} y_i \log(\frac{p_i}{1-p_i}) + \sum_{i=1}^{n} \log(1-p_i)$$

- After some algebra (or consultation of https://stats.stackexchange.com/a/16541):

$$\begin{aligned}\frac{\partial \ell}{\partial \beta_j} &= \dots \\ &= \sum_i y_i x_{ij} - \sum_i p_i x_{ij}, \\ \frac{\partial^2 \ell}{\partial \beta_j \partial \beta_k} &= \dots\end{aligned}$$

(ellipses are left as an exercise for the reader.)

How can one calculate derivatives?

- By hand (aka analytic derivatives)

How can one calculate derivatives?

- By hand (aka analytic derivatives)
- Symbolic differentiation (Mathematica, Sage, other software packages)

How can one calculate derivatives?

- By hand (aka analytic derivatives)
- Symbolic differentiation (Mathematica, Sage, other software packages)
- Numerical differentiation (computational by picking a small value for Δx):

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

How can one calculate derivatives?

- By hand (aka analytic derivatives)
- Symbolic differentiation (Mathematica, Sage, other software packages)
- Numerical differentiation (computational by picking a small value for Δx):

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

- Automatic differentiation

Why automatic differentiation?

- It's magic! (since minimal additional code or thought is required to get derivatives)

Why automatic differentiation?

- It's magic! (since minimal additional code or thought is required to get derivatives)
- Exact to machine precision.

Why automatic differentiation?

- It's magic! (since minimal additional code or thought is required to get derivatives)
- Exact to machine precision.
- Efficient, e.g. for the TMB package: $\text{cost}(\nabla f) \leq 4\,\text{cost}(f)$.

Why automatic differentiation?

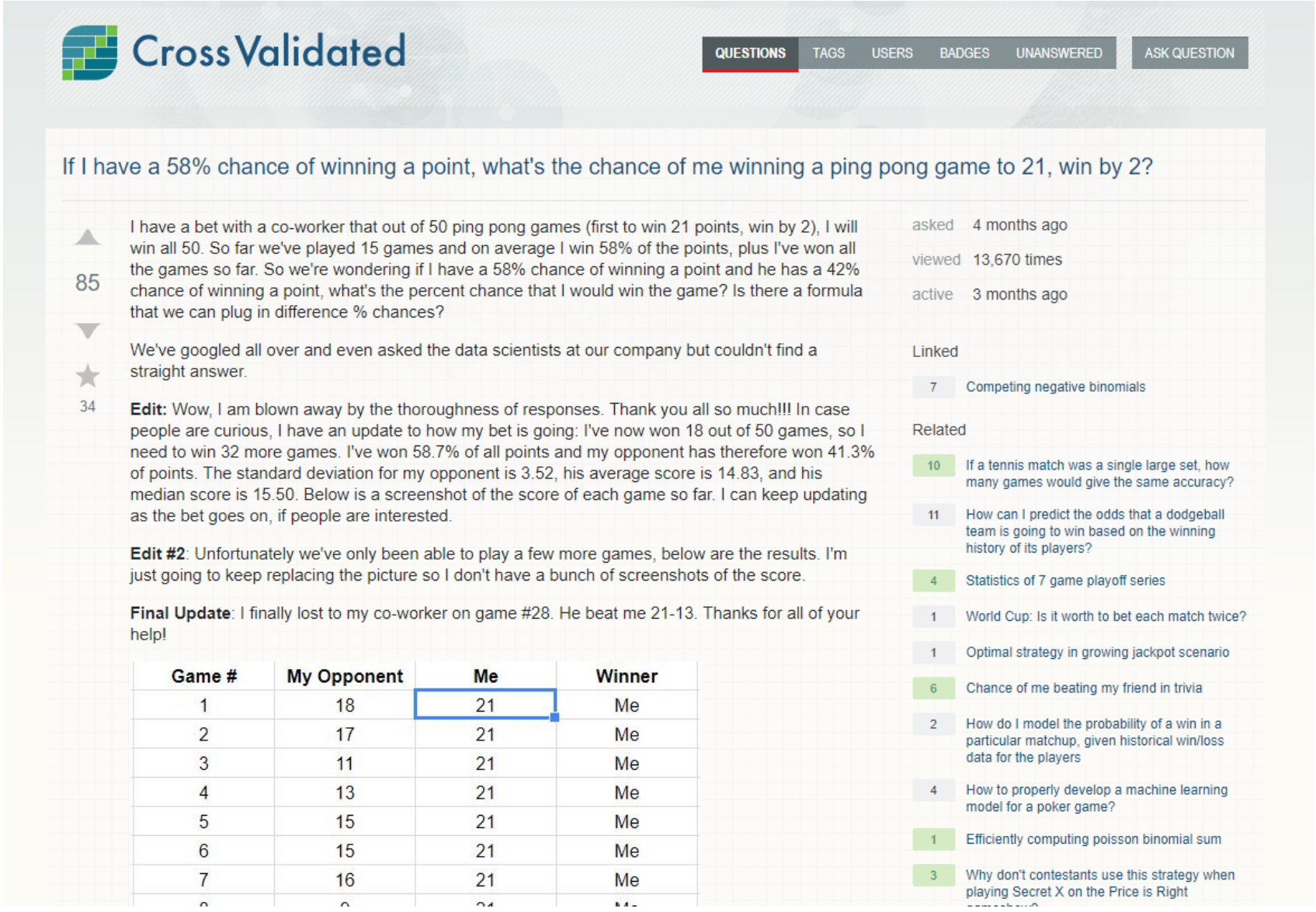
- It's magic! (since minimal additional code or thought is required to get derivatives)
- Exact to machine precision.
- Efficient, e.g. for the TMB package: $\text{cost}(\nabla f) \leq 4\,\text{cost}(f)$.
- https://justindomke.wordpress.com/2009/02/17/automatic-differentiation-the-most-criminally-underused-tool-in-the-p

Why automatic differentiation?

- It's magic! (since minimal additional code or thought is required to get derivatives)
- Exact to machine precision.
- Efficient, e.g. for the TMB package: $\text{cost}(\nabla f) \leq 4\,\text{cost}(f)$.
- https://justindomke.wordpress.com/2009/02/17/automatic-differentiation-the-most-criminally-underused-tool-in-the-p
- We'll come back to a more detailed description later.

Part 2: The stats.stackexchange (aka Cross Validated) problem

(adapted from www.xkcd.com/386)



(https://stats.stackexchange.com/questions/329521)

Qn:

If I have a 58% chance of winning a point, what's the chance of me winning a ping pong game to 21, win by 2?

85

34

I have a bet with a co-worker that out of 50 ping pong games (first to win 21 points, win by 2), I will win all 50. So far we've played 15 games and on average I win 58% of the points, plus I've won all the games so far. So we're wondering if I have a 58% chance of winning a point and he has a 42% chance of winning a point, what's the percent chance that I would win the game? Is there a formula that we can plug in difference % chances?

We've googled all over and even asked the data scientists at our company but couldn't find a straight answer.

Edit: Wow, I am blown away by the thoroughness of responses. Thank you all so much**!!!** In case people are curious, I have an update to how my bet is going: I've now won 18 out of 50 games, so I need to win 32 more games. I've won 58.7% of all points and my opponent has therefore won 41.3% of points. The standard deviation for my opponent is 3.52, his average score is 14.83, and his median score is 15.50. Below is a screenshot of the score of each game so far. I can keep updating as the bet goes on, if people are interested.

Edit #2: Unfortunately we've only been able to play a few more games, below are the results. I'm just going to keep replacing the picture so I don't have a bunch of screenshots of the score.

Final Update: I finally lost to my co-worker on game #28. He beat me 21-13. Thanks for all of your help!

Game #	My Opponent	Me	Winner
1	18	21	Me
2	17	21	Me
3	11	21	Me
4	13	21	Me
5	15	21	Me
6	15	21	Me
7	16	21	Me

asked 4 months ago

viewed 13,670 times

active 3 months ago

Linked

7 Competing negative binomials

Related

many games would give the same accuracy?

11 How can I predict the odds that a dodgeball team is going to win based on the winning history of its players?

4 Statistics of 7 game playoff series

1 World Cup: Is it worth to bet each match twice?

1 Optimal strategy in growing jackpot scenario

6 Chance of me beating my friend in trivia

2 How do I model the probability of a win in a particular matchup, given historical win/loss data for the players

4 How to properly develop a machine learning model for a poker game?

1 Efficiently computing poisson binomial sum

3 Why don't contestants use this strategy when playing Secret X on the Price is Right

Qn:

If I have a 58% chance of winning a point, what's the chance of me winning a ping pong game to 21, win by 2?

85

34

I have a bet with a co-worker that out of 50 ping pong games (first to win 21 points, win by 2), I will win all 50. So far we've played 15 games and on average I win 58% of the points, plus I've won all the games so far. So we're wondering if I have a 58% chance of winning a point and he has a 42% chance of winning a point, what's the percent chance that I would win the game? Is there a formula that we can plug in difference % chances?

We've googled all over and even asked the data scientists at our company but couldn't find a straight answer.

Edit: Wow, I am blown away by the thoroughness of responses. Thank you all so much**!!!** In case people are curious, I have an update to how my bet is going: I've now won 18 out of 50 games, so I need to win 32 more games. I've won 58.7% of all points and my opponent has therefore won 41.3% of points. The standard deviation for my opponent is 3.52, his average score is 14.83, and his median score is 15.50. Below is a screenshot of the score of each game so far. I can keep updating as the bet goes on, if people are interested.

Edit #2: Unfortunately we've only been able to play a few more games, below are the results. I'm just going to keep replacing the picture so I don't have a bunch of screenshots of the score.

Final Update: I finally lost to my co-worker on game #28. He beat me 21-13. Thanks for all of your help!

asked 4 months ago

viewed 13,670 times

active 3 months ago

Linked

7 Competing negative binomials

Related

[illegible] many games would give the same accuracy?

11 How can I predict the odds that a dodgeball team is going to win based on the winning history of its players?

4 Statistics of 7 game playoff series

1 World Cup: Is it worth to bet each match twice?

1 Optimal strategy in growing jackpot scenario

6 Chance of me beating my friend in trivia

2 How do I model the probability of a win in a particular matchup, given historical win/loss data for the players

4 How to properly develop a machine learning model for a poker game?

1 Efficiently computing poisson binomial sum

3 Why don't contestants use this strategy when playing Secret X on the Price is Right

Data:

Game #	My Opponent	Me	Winner
1	18	21	Me
2	17	21	Me
3	11	21	Me
4	13	21	Me
5	15	21	Me
6	15	21	Me
7	16	21	Me

If I have a 58% chance of winning a point, what's the chance of me winning a ping pong game to 21, win by 2?

Assume:

85

34

I have a bet with a co-worker that out of 50 ping pong games (first to win 21 points, win by 2), I will win all 50. So far we've played 15 games and on average I win 58% of the points, plus I've won all the games so far. So we're wondering if I have a 58% chance of winning a point and he has a 42% chance of winning a point, what's the percent chance that I would win the game? Is there a formula that we can plug in difference % chances?

We've googled all over and even asked the data scientists at our company but couldn't find a straight answer.

Edit: Wow, I am blown away by the thoroughness of responses. Thank you all so much**!!!** In case people are curious, I have an update to how my bet is going: I've now won 18 out of 50 games, so I need to win 32 more games. I've won 58.7% of all points and my opponent has therefore won 41.3% of points. The standard deviation for my opponent is 3.52, his average score is 14.83, and his median score is 15.50. Below is a screenshot of the score of each game so far. I can keep updating as the bet goes on, if people are interested.

Edit #2: Unfortunately we've only been able to play a few more games, below are the results. I'm just going to keep replacing the picture so I don't have a bunch of screenshots of the score.

Final Update: I finally lost to my co-worker on game #28. He beat me 21-13. Thanks for all of your help!

Game #	My Opponent	Me	Winner
1	18	21	Me
2	17	21	Me
3	11	21	Me
4	13	21	Me
5	15	21	Me
6	15	21	Me
7	16	21	Me

asked 4 months ago

viewed 13,670 times

active 3 months ago

Linked

7 Competing negative binomials

Related

10 If a tennis match was a single large set, how many games would give the same accuracy?

11 How can I predict the odds that a dodgeball team is going to win based on the winning history of its players?

4 Statistics of 7 game playoff series

1 World Cup: Is it worth to bet each match twice?

1 Optimal strategy in growing jackpot scenario

6 Chance of me beating my friend in trivia

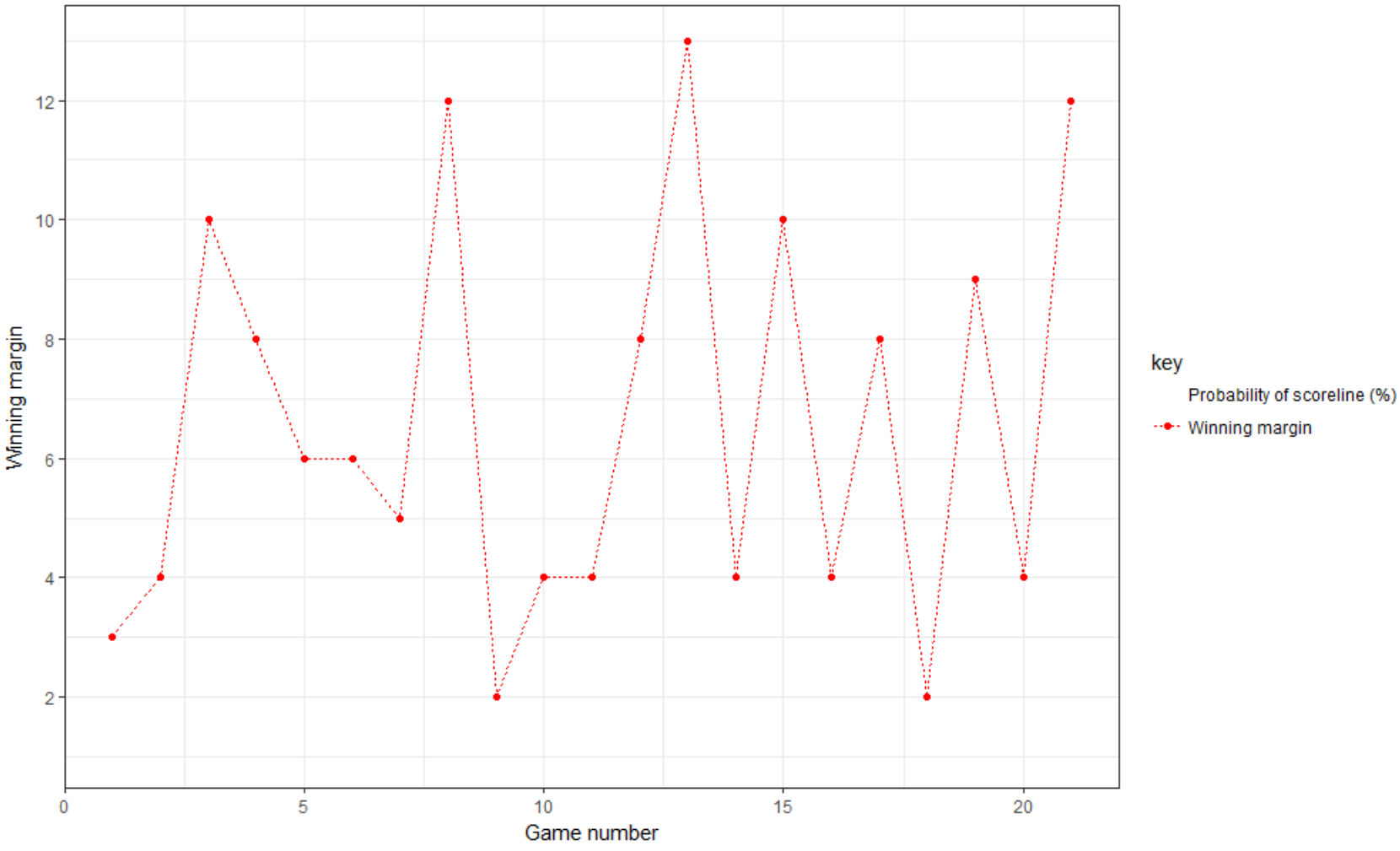
2 How do I model the probability of a win in a particular matchup, given historical win/loss data for the players

4 How to properly develop a machine learning model for a poker game?

1 Efficiently computing poisson binomial sum

3 Why don't contestants use this strategy when playing Secret X on the Price is Right

But what if the assumption is wrong?



But what if the assumption is wrong?

But what if the assumption is wrong?

- Postulate that in a truly competitive game, the original poster (OP) has a much higher (than 59%) chance of winning a point.

But what if the assumption is wrong?

- Postulate that in a truly competitive game, the original poster (OP) has a much higher (than 59%) chance of winning a point.
- However, to make the game fun, assume that the OP oscillates between playing really well and not as well.

But what if the assumption is wrong?

- Postulate that in a truly competitive game, the original poster (OP) has a much higher (than 59%) chance of winning a point.
- However, to make the game fun, assume that the OP oscillates between playing really well and not as well.
- Whenever a game is won by a tight margin, the OP plays better in the next game, and if a game is won by a huge margin, then they play worse.

But what if the assumption is wrong?

- Postulate that in a truly competitive game, the original poster (OP) has a much higher (than 59%) chance of winning a point.
- However, to make the game fun, assume that the OP oscillates between playing really well and not as well.
- Whenever a game is won by a tight margin, the OP plays better in the next game, and if a game is won by a huge margin, then they play worse.
- How can this be modelled mathematically?

Modelling: Probability to win a point

- Let s_i be the observed score difference, and p_i be the probability of the OP winning a point in the i-th game.

Modelling: Probability to win a point

- Let s_i be the observed score difference, and p_i be the probability of the OP winning a point in the i-th game.
- One way to increase the OP's playing ability for the next game is to add a small amount to p_i:

$$\text{expit}(\text{logit}(p_i) + \beta)$$

Modelling: Probability to win a point

- Let s_i be the observed score difference, and p_i be the probability of the OP winning a point in the i-th game.
- One way to increase the OP's playing ability for the next game is to add a small amount to p_i:

$$\text{expit}(\text{logit}(p_i) + \beta)$$

- Let the probability of the original poster winning the $i + 1$-th game be:

$$p_{i+1} = \alpha p_0 + (1 - \alpha)(\text{expit}(\text{logit}(p_i) + \beta))$$

Modelling: Probability to win a point

- Let s_i be the observed score difference, and p_i be the probability of the OP winning a point in the i-th game.
- One way to increase the OP's playing ability for the next game is to add a small amount to p_i:

$$\text{expit}(\text{logit}(p_i) + \beta)$$

- Let the probability of the original poster winning the $i+1$-th game be:

$$p_{i+1} = \alpha p_0 + (1 - \alpha)(\text{expit}(\text{logit}(p_i) + \beta))$$

where α is a 'mixing' coefficient:

$$\alpha = \text{expit}(\alpha_0 + \alpha_1 s_i)$$

Regression

Have equations:

$$p_{i+1} = \alpha p_0 + (1 - \alpha)(\text{expit}(\text{logit}(p_i) + \beta))$$

for:

$$\alpha = \text{expit}(\alpha_0 + \alpha_1 s_i)$$

Regression

Have equations:

$$p_{i+1} = \alpha p_0 + (1 - \alpha)(\text{expit}(\text{logit}(p_i) + \beta))$$

for:

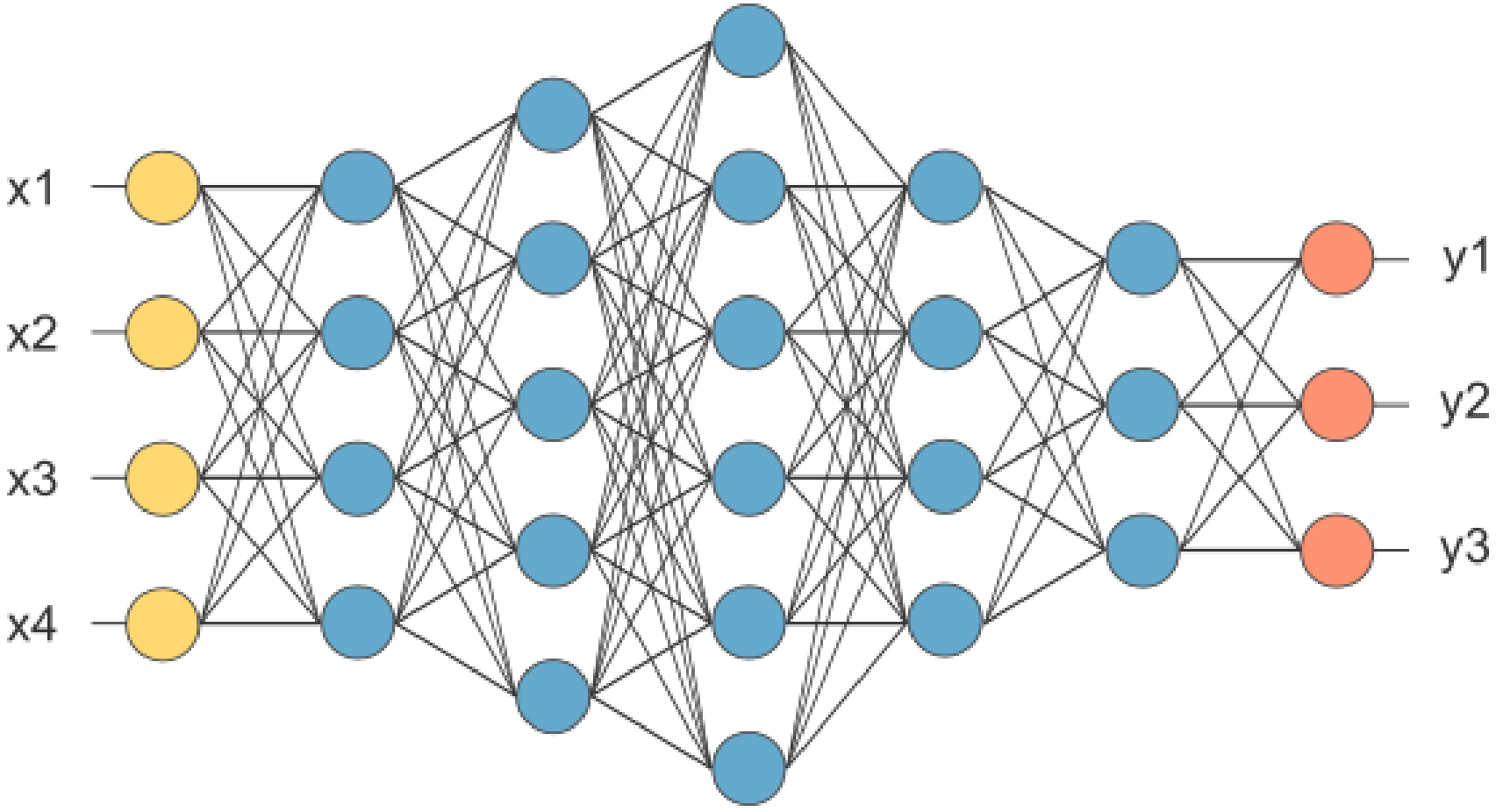
$$\alpha = \text{expit}(\alpha_0 + \alpha_1 s_i)$$

The aim is to find values for the following parameters that maximise the likelihood of observing the game scorelines:

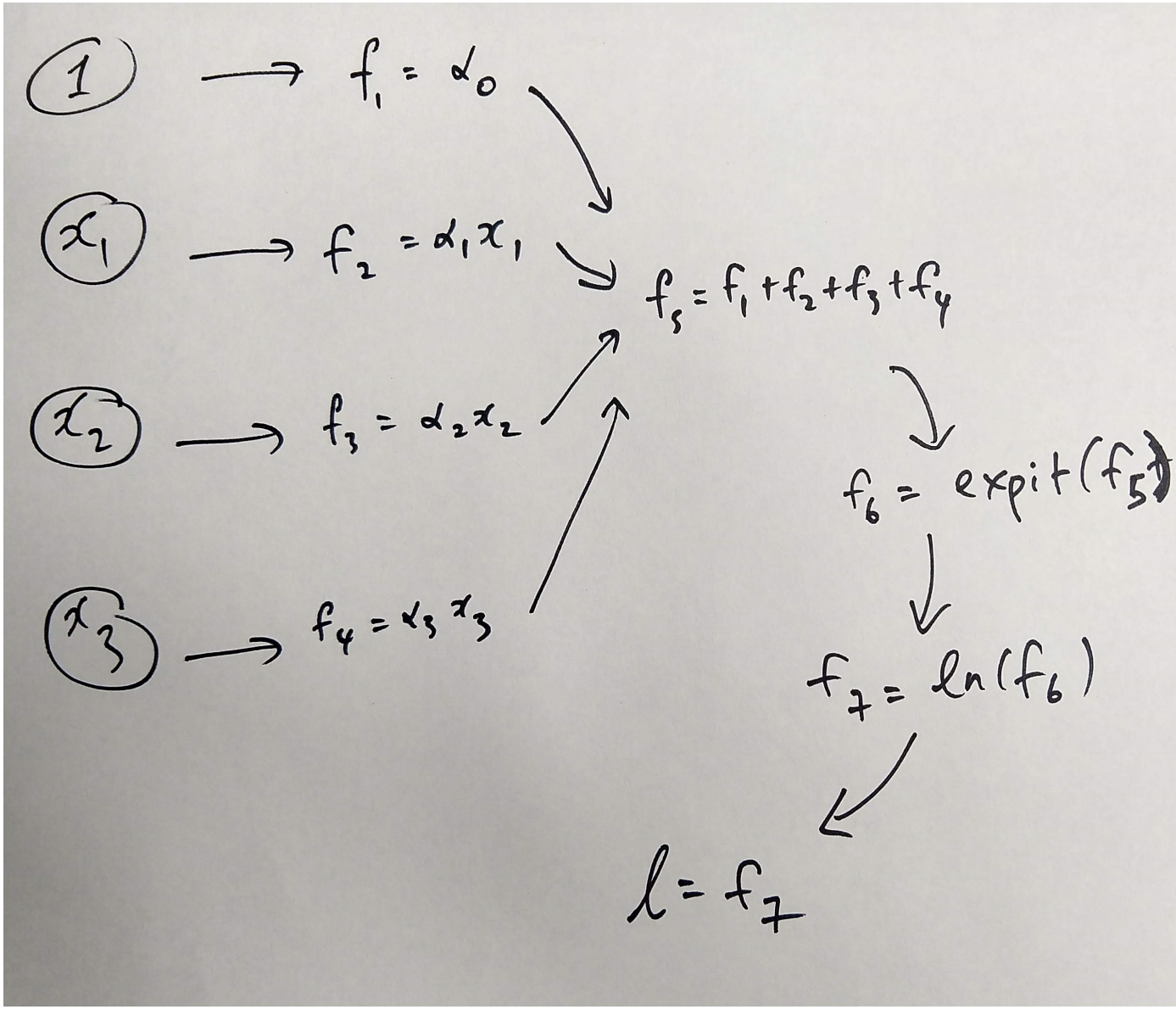
- p_0: probability of point win when original poster is not playing at their best.
- β: changes probability of winning in the next game depending on the scoreline this game.
- α_0 and α_1: coefficients governing the averaging between p_0 and p_i.

Part 3: Automatic derivatives

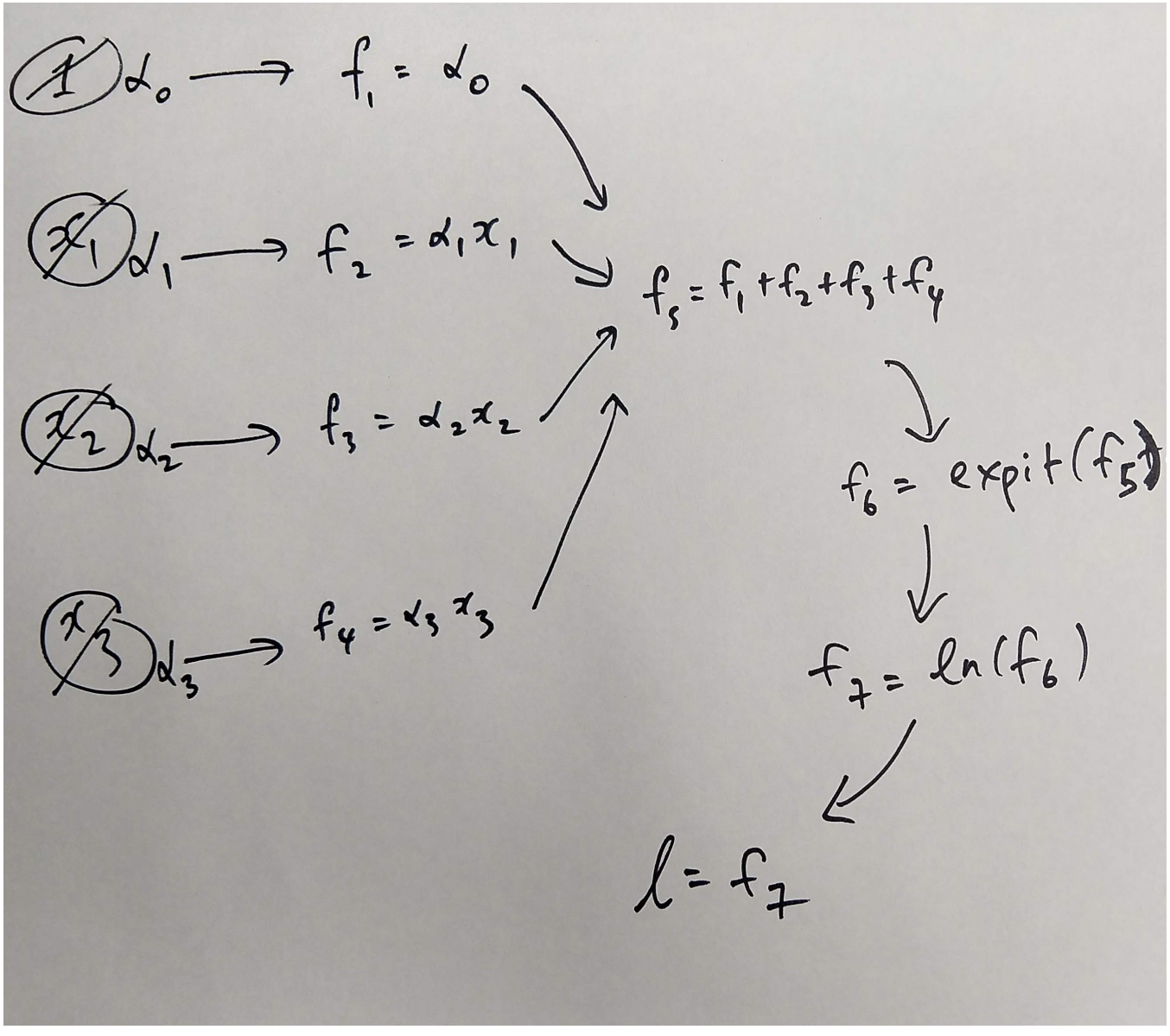
Neural networks



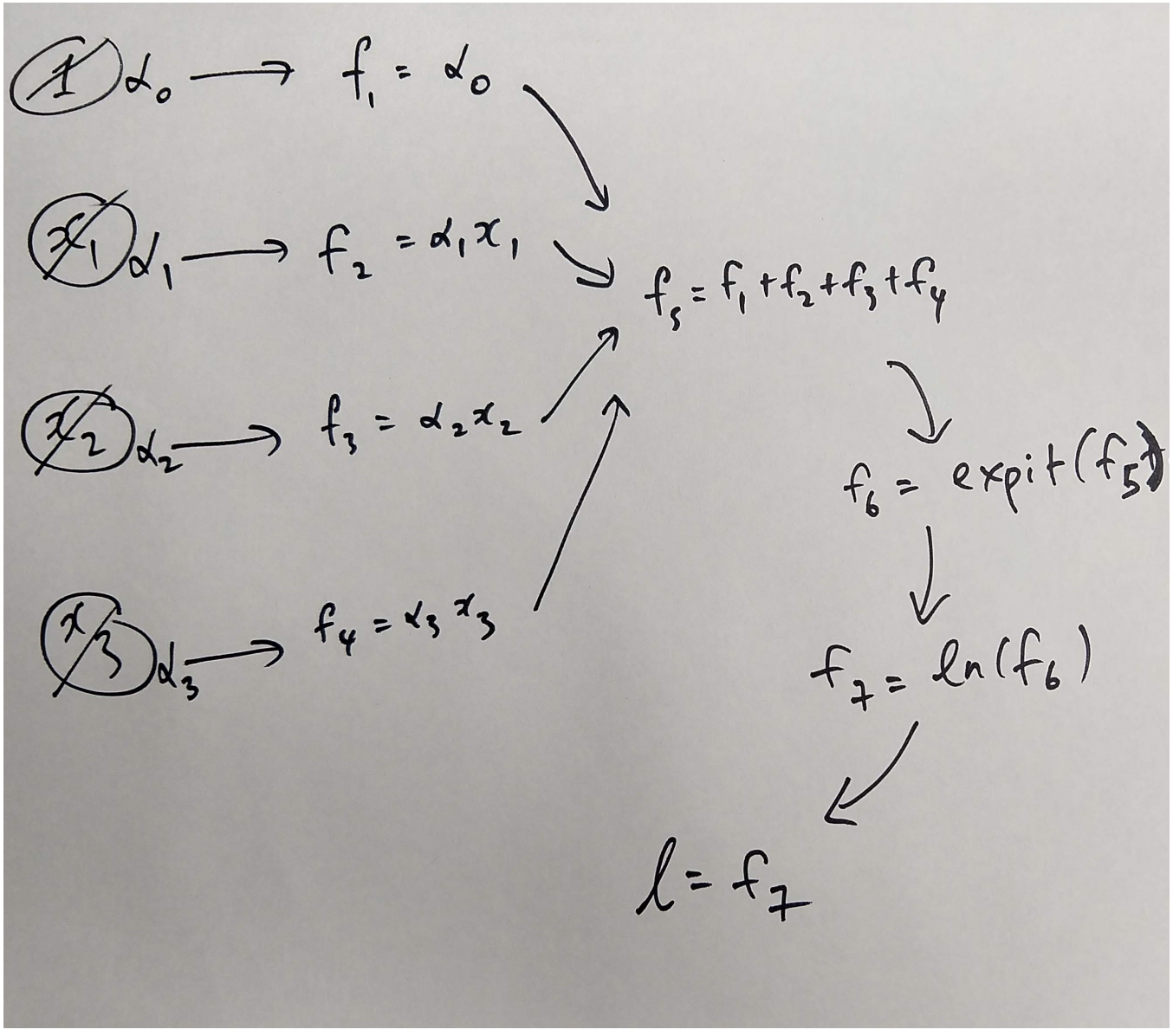
Neural networks



Neural networks



Neural networks



WTF $\frac{\partial \ell}{\partial \alpha_0}, \frac{\partial \ell}{\partial \alpha_1}, \frac{\partial \ell}{\partial \alpha_2}, \frac{\partial \ell}{\partial \alpha_3}$.

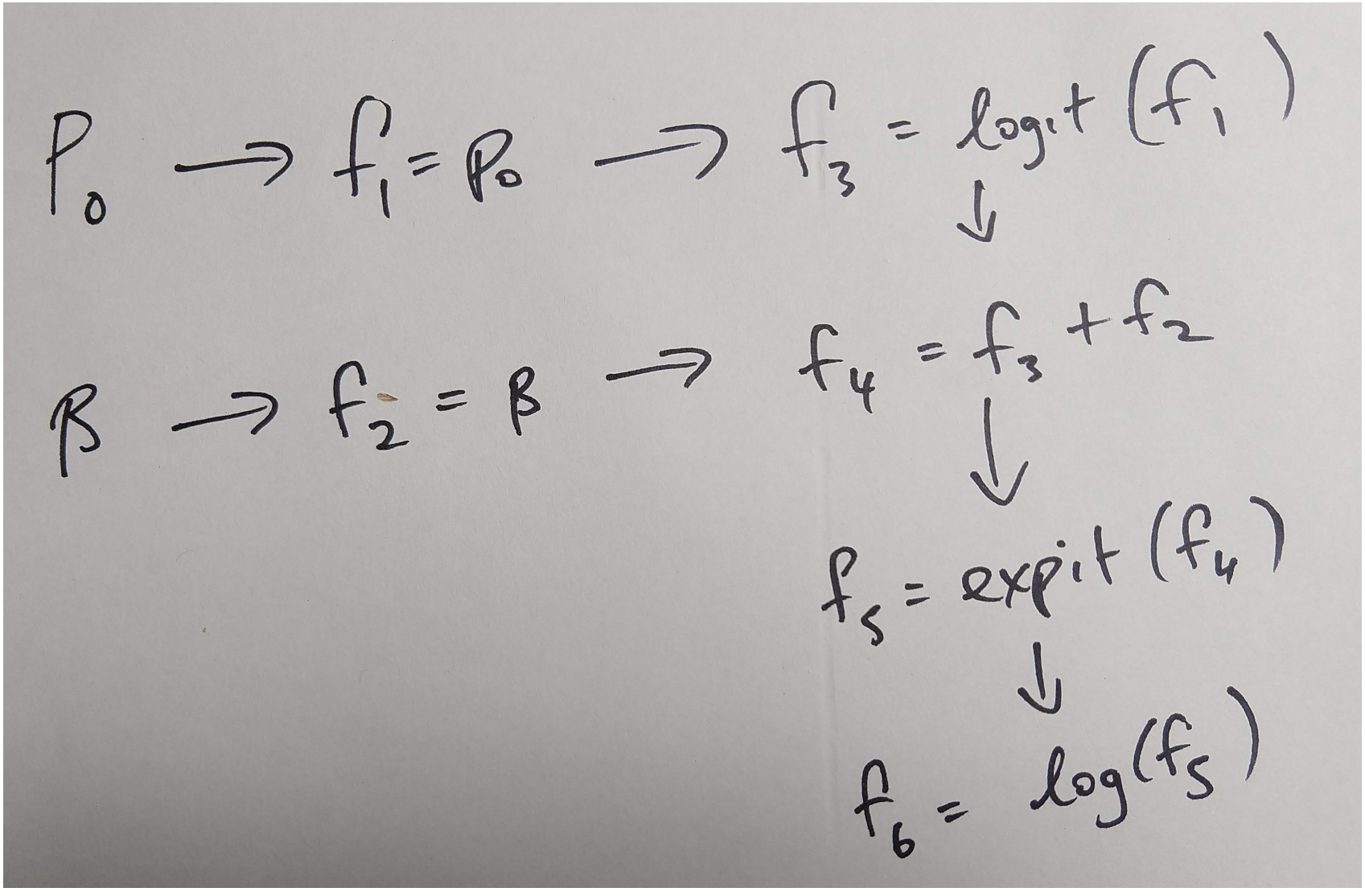
Optimisation: Gradient descent

Start with an initial guess $\alpha_j^{(0)}$ for the optimal values, and iteratively build better guesses $\alpha_j^{(1)}, \alpha_j^{(2)}, \ldots$, via:

$$\alpha_j^{(i+1)} = \alpha_j^{(i)} - \gamma \frac{\partial \ell}{\partial \alpha_j^{(i)}}$$

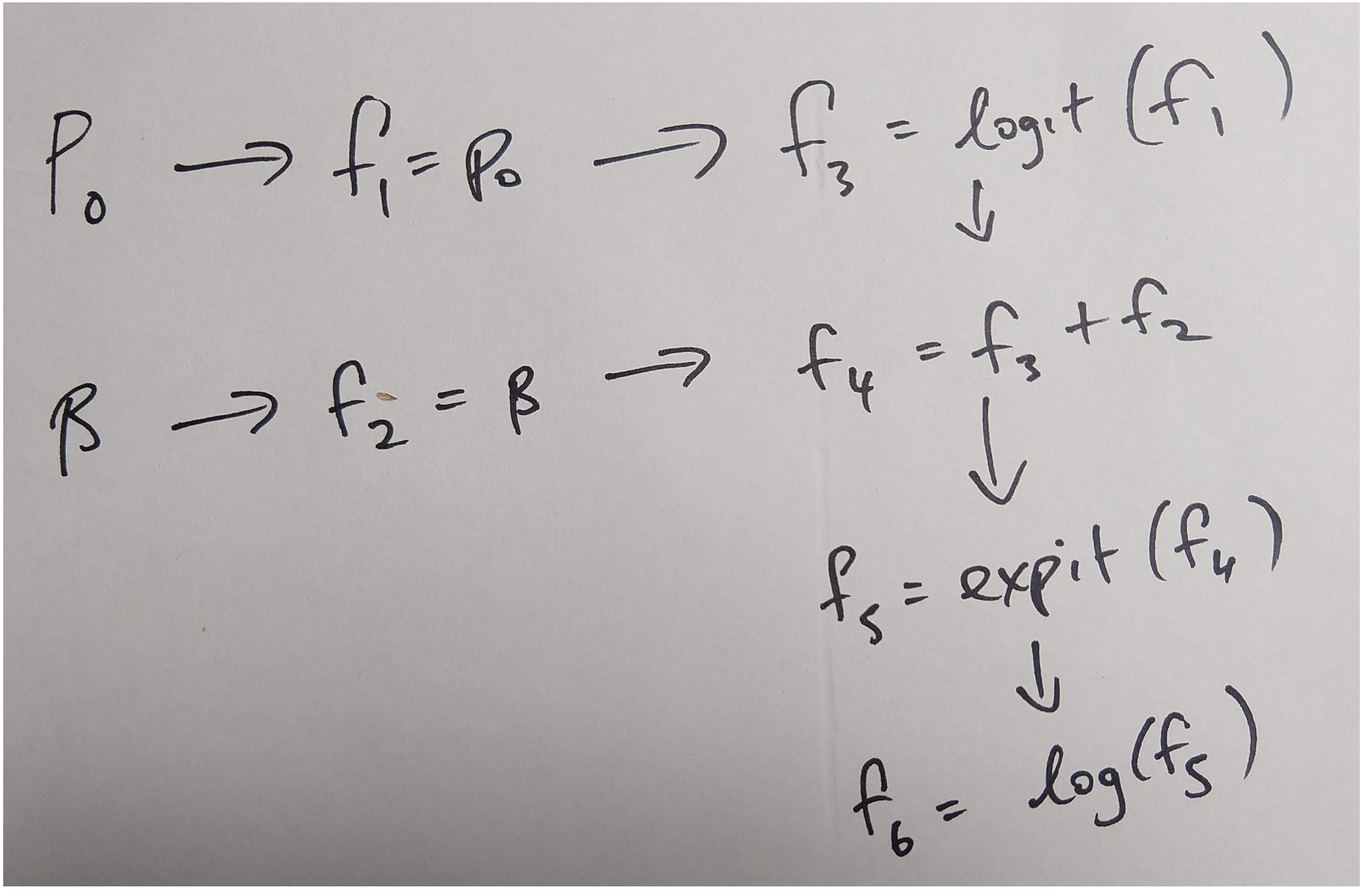
(γ is a learning rate parameter that does not concern us in this talk.)

Applied to expit(logit(p_0) + β)



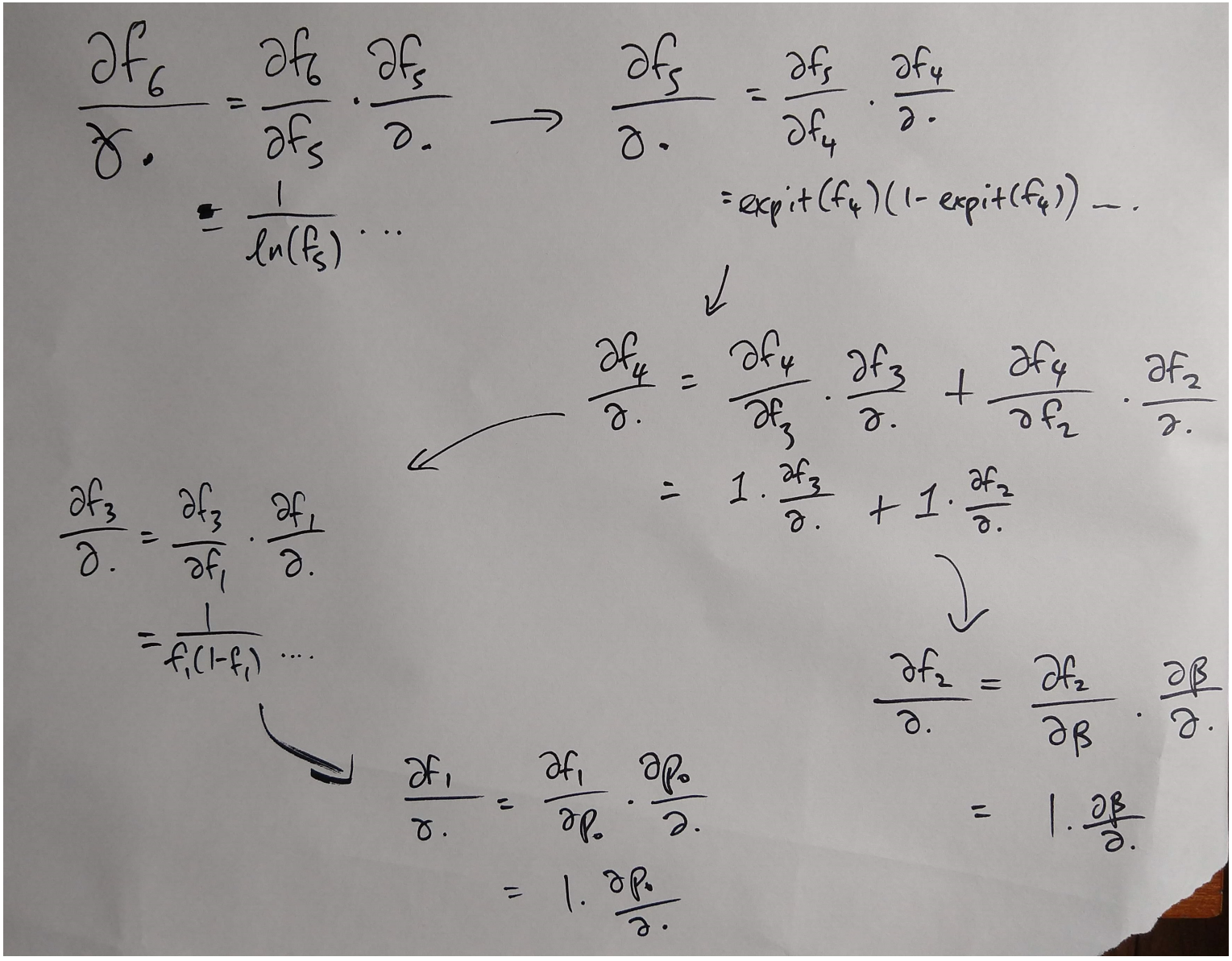
Forward propagation: follow arrows from start to finish

Calculate f_1, then f_2, then f_3, etc.

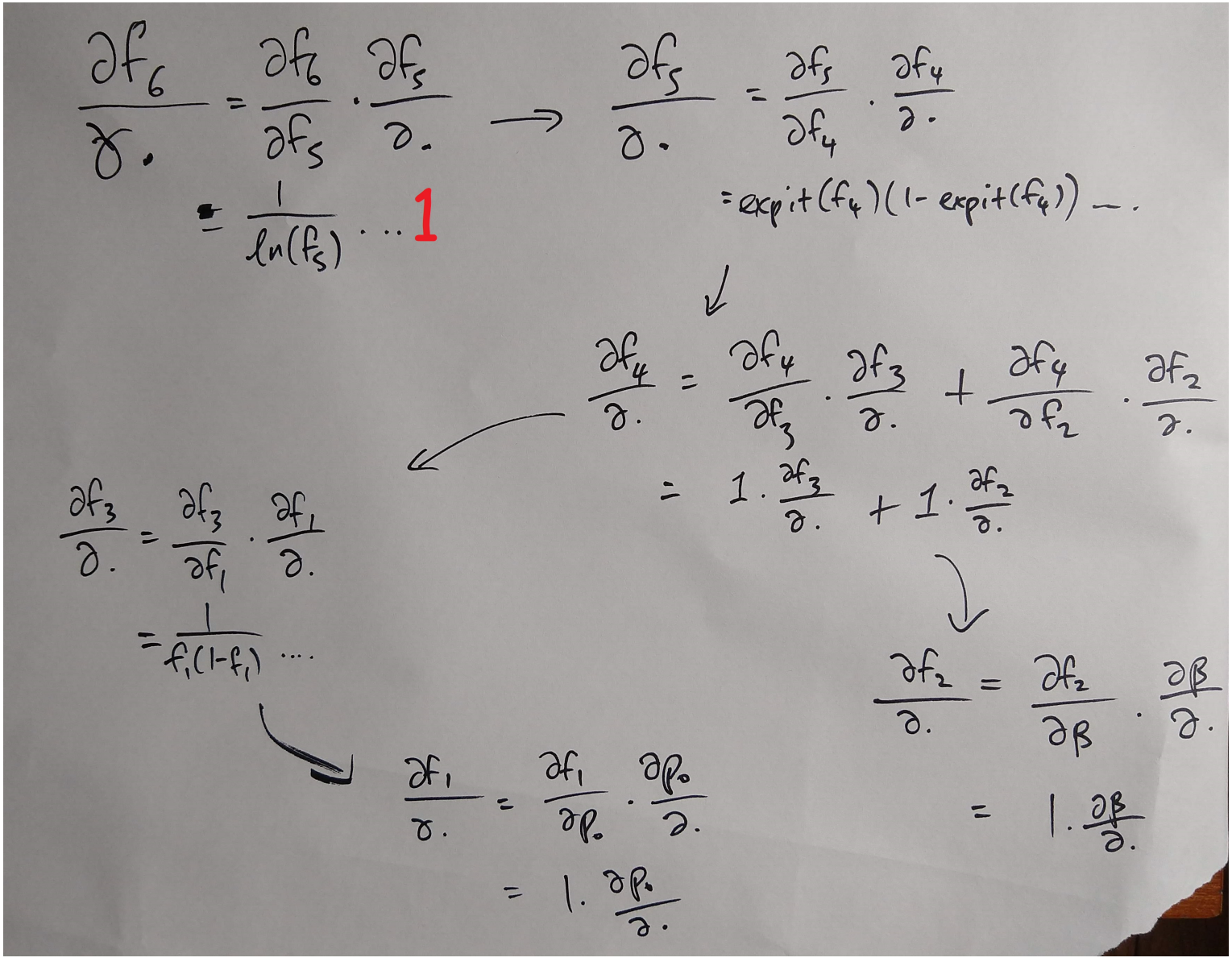


Backpropagation: reverse arrows and follow them

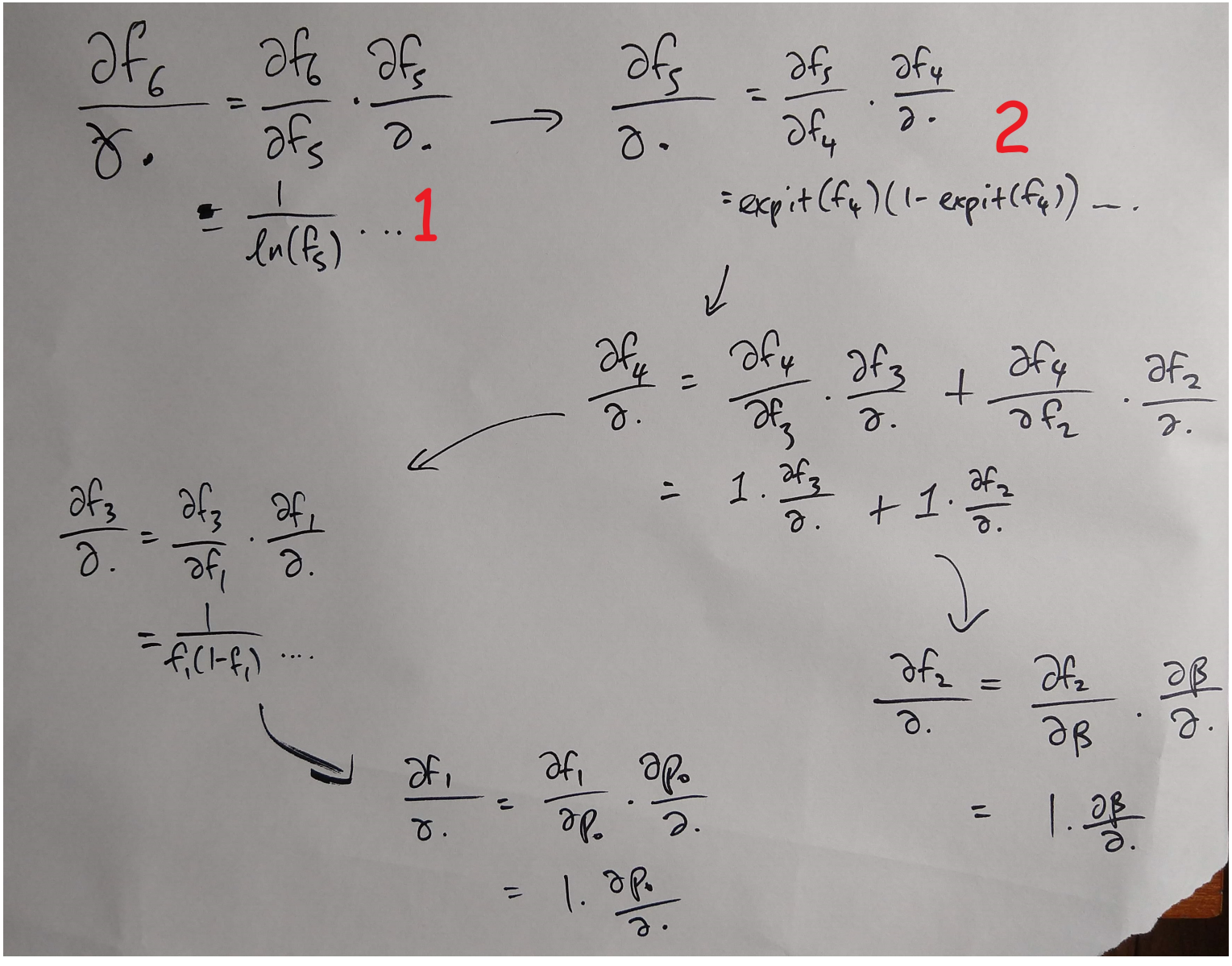
Calculate derivatives $\frac{\partial f_6}{\partial \cdot}$, $\frac{\partial f_5}{\partial \cdot}$, $\frac{\partial f_4}{\partial \cdot}$, etc



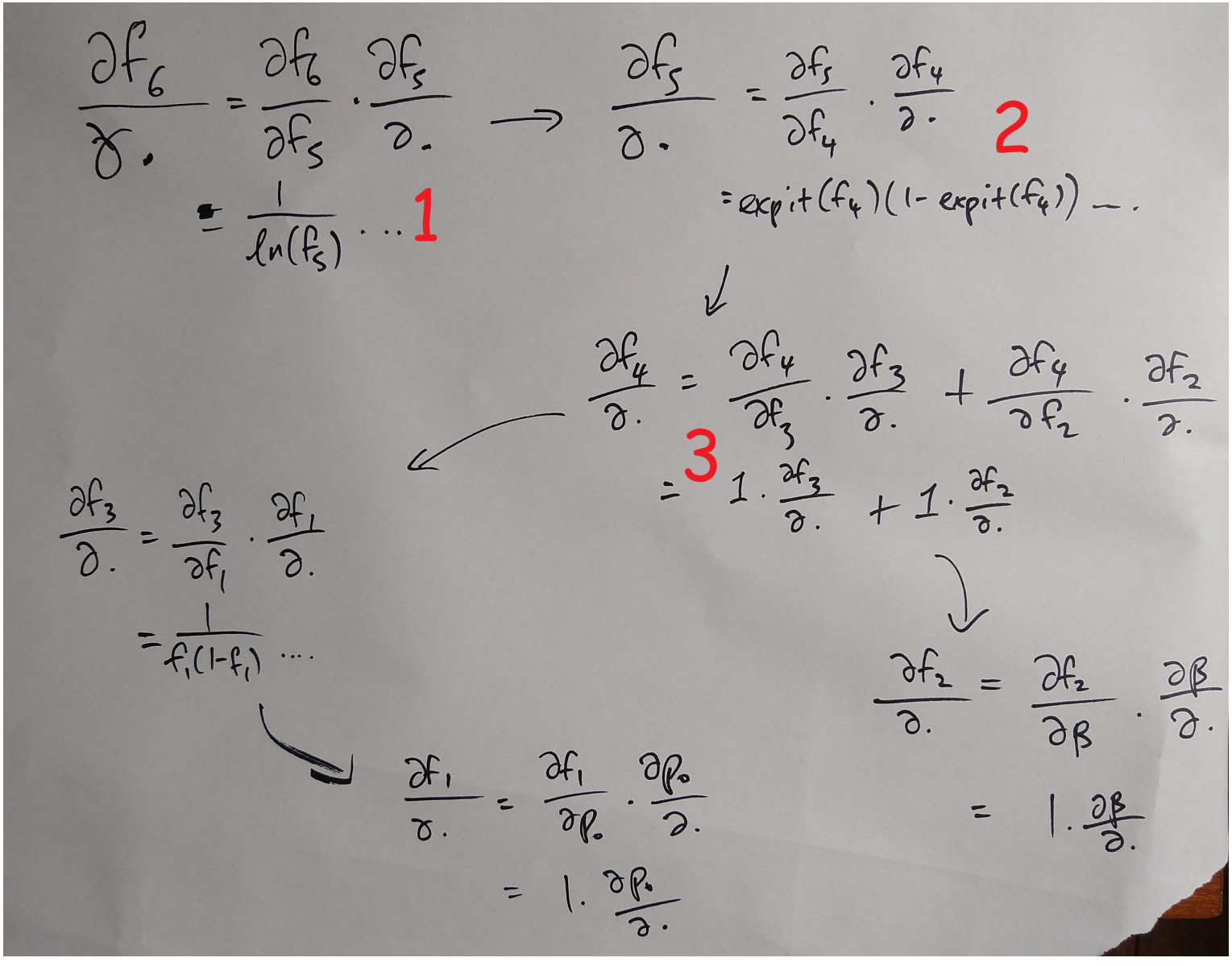
Backpropagation/Reverse mode AD



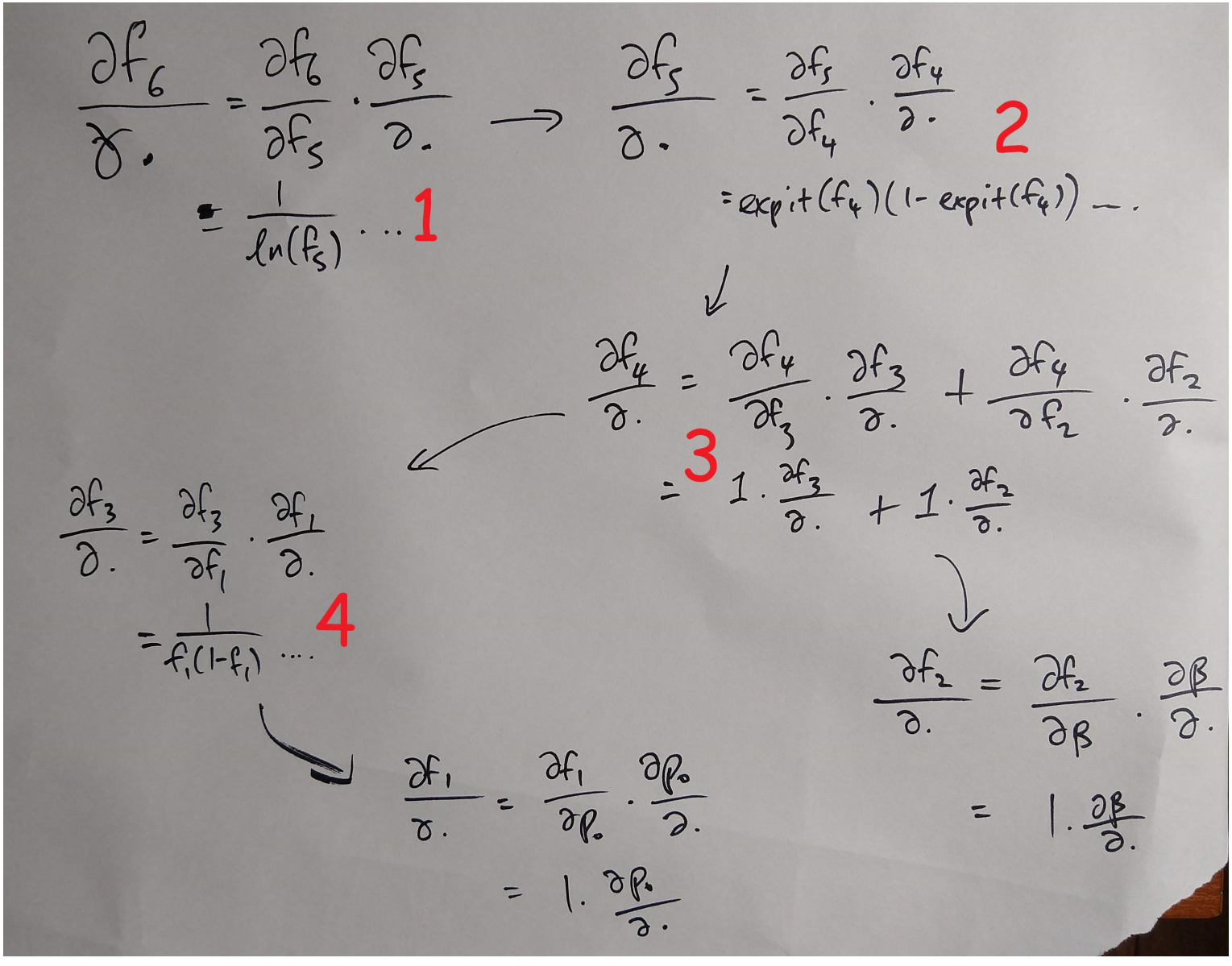
Backpropagation/Reverse mode AD



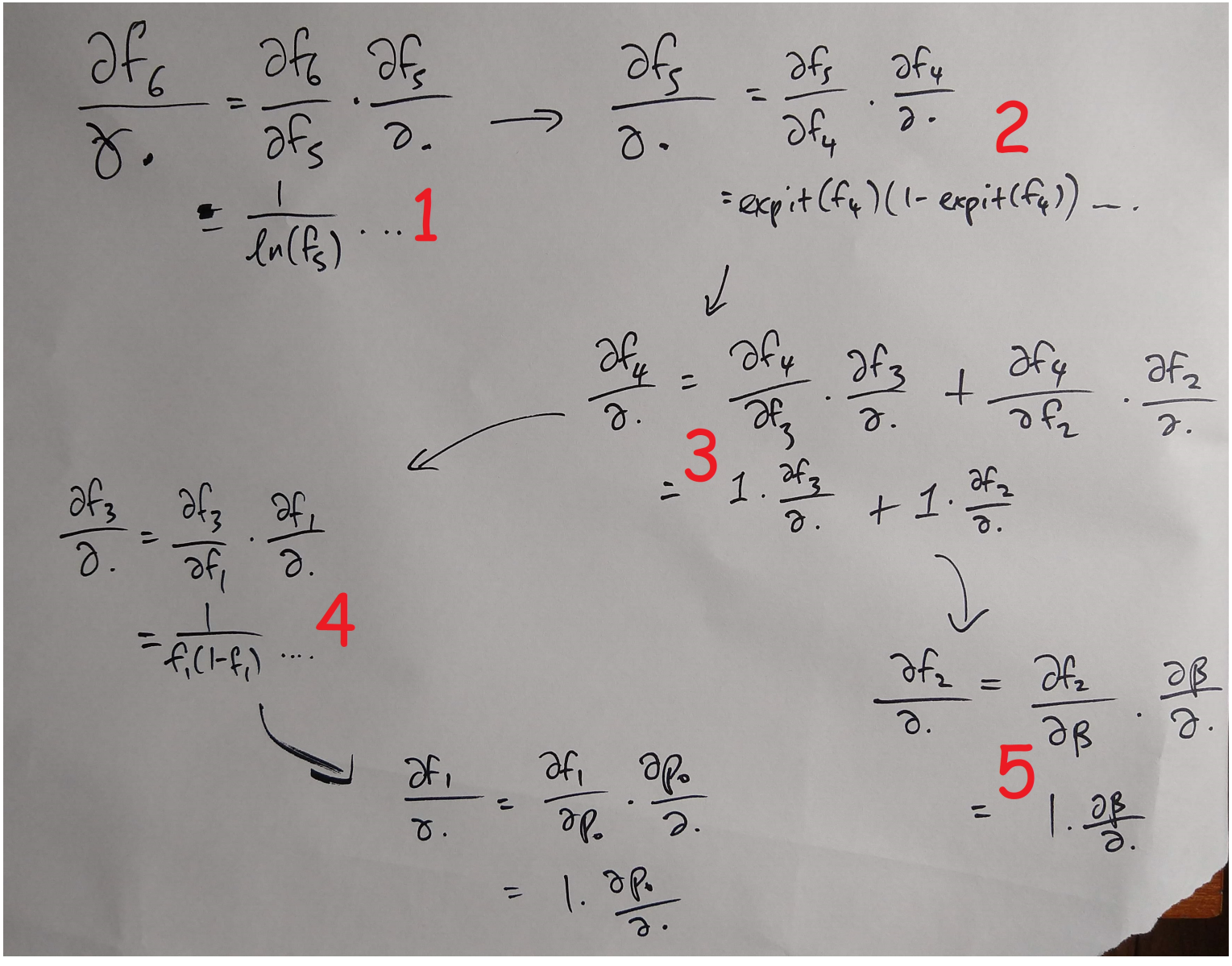
Backpropagation/Reverse mode AD



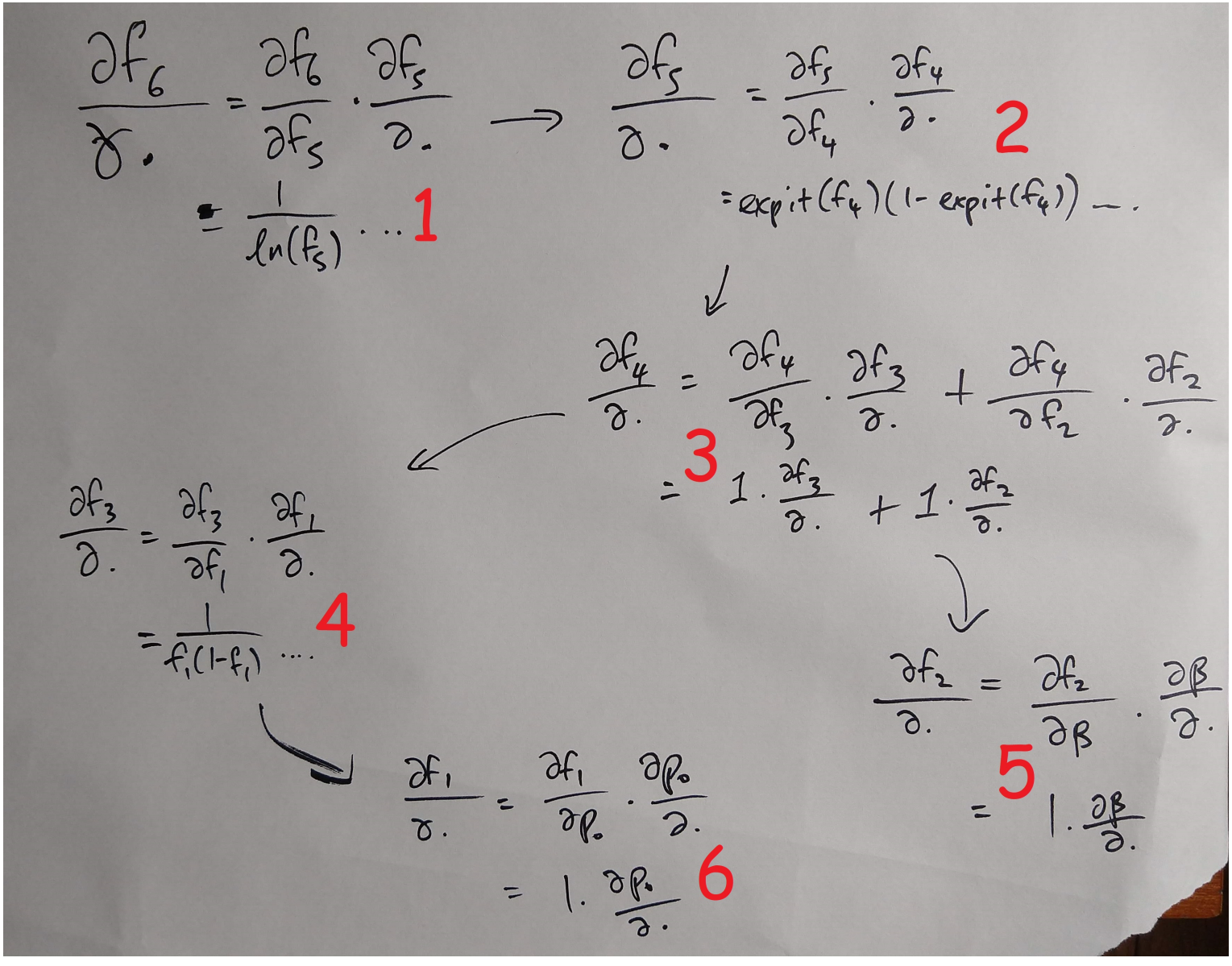
Backpropagation/Reverse mode AD



Backpropagation/Reverse mode AD



Backpropagation/Reverse mode AD



Part 4: Automatic derivatives in R

Three modelling contexts in R

Three modelling contexts in R

TMB - "Quickly implement complex random effect models. The package combines 'CppAD' (C++ automatic differentiation) ... to obtain efficient implementation of the applied Laplace approximation with exact derivatives."

Three modelling contexts in R

TMB - "Quickly implement complex random effect models. The package combines 'CppAD' (C++ automatic differentiation) ... to obtain efficient implementation of the applied Laplace approximation with exact derivatives."

rstan - "Implements full Bayesian statistical inference via Markov Chain Monte Carlo, ... automatic differentiation is used to quickly and accurately evaluate gradients without burdening the user with the need to derive the partial derivatives."

Three modelling contexts in R

TMB - "Quickly implement complex random effect models. The package combines 'CppAD' (C++ automatic differentiation) ... to obtain efficient implementation of the applied Laplace approximation with exact derivatives."

rstan - "Implements full Bayesian statistical inference via Markov Chain Monte Carlo, ... automatic differentiation is used to quickly and accurately evaluate gradients without burdening the user with the need to derive the partial derivatives."

tensorflow - "An open source software library for numerical computation using data flow graphs."

Implementing the objective function

- We want a function that calculates the log of the probability of the OP winning the i-th game, given their probability p_i of winning a point in that game:

$$p_{i+1} = \alpha p_0 + (1 - \alpha)(\text{expit}(\text{logit}(p_i) + \beta))$$

Implementing the objective function

- We want a function that calculates the log of the probability of the OP winning the i-th game, given their probability p_i of winning a point in that game:

$$p_{i+1} = \alpha p_0 + (1 - \alpha)(\text{expit}(\text{logit}(p_i) + \beta))$$

- The rest of this talk demonstrates implementations of this function in TMB, rstan, and tensorflow.

Modelling: Probability to win a game with a certain scoreline

- In a game, suppose player one wins points with probability p, and let x and y be the total number of points player one and two won, respectively.
- Suppose player one reaches 21 points first with a margin of two:

$$\Pr(21 : y) = \binom{20+y}{y} p^{21}(1-p)^{y}$$

Modelling: Probability to win a game with a certain scoreline

- In a game, suppose player one wins points with probability p, and let x and y be the total number of points player one and two won, respectively.
- Suppose player one reaches 21 points first with a margin of two:

$$\Pr(21 : y) = \binom{20+y}{y} p^{21}(1-p)^y$$

- Else:

$$\Pr(x : y) = 2^{y-20} \binom{40}{20} p^x (1-p)^y$$

(whuber: https://stats.stackexchange.com/a/329554)

Code: R

```r
get_prob_of_scoreline <- function(me, you, prob_win_point){

    # warning, not vectorised!
    stopifnot(all(length(me) == 1, length(you) == 1, length(prob_win_point) == 1))

    min_me_you <- min(me, you)


    if (you >= 20) {
        out_prob <- choose(40, 20) * 2^(min_me_you - 20) * prob_win_point^me * (1 - prob_win_point)^you

    } else {
        out_prob <- choose(20 + min_me_you, min_me_you) * prob_win_point^me * (1 - prob_win_point)^you

    }

    return(out_prob);
}
```

Code: R

```r
R_obj_fun_template <- function(par, me, you, score_diff) {

    p_0 <- par[1]
    alpha_0 <- par[2]
    alpha_1 <- par[3]
    beta <- par[4]

    n_obs <- length(score_diff)

    p <- p_0 # initialise initial model fitted probability
    nll <- 0 # initialise model negloglik

    for (i in seq_along(you)) {

        if (i > 1) {
            # mixing coefficient
            alpha <- plogis(alpha_0 + alpha_1 * score_diff[i - 1]);

            # update p with model
            p <-  alpha * p_0 + (1 - alpha) *  plogis(qlogis(p) + beta);


        }

        # increment nll
        nll <- nll - log(get_prob_of_scoreline(me[i], you[i], p));

    }



    return(nll)
}

optim_in_R$fn <- function(par) R_obj_fun_template(par, df$Me, df$Opponent, df$score_diff)
optim_in_R$par <- c(0.592246, 0, 0, 0)
# quasi-Newton optimisation with approximations to the Hessian
# i.e. second derivatives are never required
optim_in_R$method <- 'BFGS'
```

Code: Tensorflow

```r
## variables
p_0 <- tf$contrib$eager$Variable(0.592246)
beta <- tf$contrib$eager$Variable(0)
alpha_0 <- tf$contrib$eager$Variable(0)
alpha_1 <- tf$contrib$eager$Variable(0)

obj_fn_tf <- function(p_0, alpha_0, alpha_1, beta) {
    ## initialise
    me <- as.numeric(df$Me)
    you <- as.numeric(df$Opponent)
    p_new <- p_0
    nll <- -tf$log(get_prob_of_scoreline_tf(me[1], you[1], p_new))


    for (i in seq(2, nrow(df), 1)) {

        # calculate mixing coefficient
        sd <- me[i - 1] - you[i - 1]
        alpha <- tf$nn$sigmoid(alpha_0 + alpha_1 * sd)

        # calculate new p
        p_new <- alpha * p_0 + (1 - alpha) * tf$nn$sigmoid(tf$log(p_new/(1 - p_new)) + beta)


        # update nll
        nll <- nll - tf$log(get_prob_of_scoreline_tf(me[i], you[i], p_new))

    }

    return(nll)
}


# get automatic gradients
g_tf <- tf$contrib$eager$gradients_function(obj_fn_tf)

# but now need to convert it back into numeric (from tensor)
get_grad_tf <- function(...) {

    return(unlist(lapply(g_tf(...), as.numeric)))
}
```

Code: Rcpp

```
R_obj_fun_template_Rcpp <- cppFunction(
    "double R_obj_fun_templatep_Rcpp(NumericVector par, IntegerVector me, IntegerVector you, IntegerVector score_diff) {



    // parameters:
    double p_0 = par[0];
    double alpha_0 = par[1];
    double alpha_1 = par[2];
    double beta = par[3];

    int n_obs = me.size();

    double p = p_0; // initialise initial model fitted probability
    double nll = 0; // initialise model negloglik


    for (int i = 0; i < n_obs; i++) {

        if (i > 0) {
            // mixing coefficient
            double alpha = Rf_plogis(alpha_0 + alpha_1 * score_diff[i - 1], 0.0, 1.0, 1, 0);

            // update p with model

            p =  alpha * p_0 + (1 - alpha) *  Rf_plogis(Rf_qlogis(p, 0.0, 1.0, 1, 0) + beta, 0.0, 1.0, 1, 0);


        }
        nll += -log(get_prob_of_scoreline_Rcpp(me[i], you[i], p));


    }


    return nll;

}",
    includes = "double get_prob_of_scoreline_Rcpp(int me, int you, double prob_win_point) {
    double out_prob;
```

Code: Stan

```
// [[Rcpp::depends(rstan)]]


#include <Rcpp.h>
#include <stan/math.hpp>

using namespace Rcpp;

// [[Rcpp::export]]
NumericVector stan_obj_fun(NumericVector par,
                           IntegerVector me,
                           IntegerVector you,
                           IntegerVector score_diff) {


    // parameters:
    stan::math::var p_0 = par[0];
    stan::math::var alpha_0 = par[1];
    stan::math::var alpha_1 = par[2];
    stan::math::var beta = par[3];

    int n_obs = me.size();

    NumericVector ret(5); // initialise return object for obj fn and gradient
    stan::math::var p = p_0; // initialise initial model fitted probability
    stan::math::var nll = 0; // initialise log-lik

    for (int i = 0; i < n_obs; i++) {

        if (i > 0) {
            // mixing coefficient
            stan::math::var alpha = inv_logit(alpha_0 + alpha_1 * score_diff[i - 1]);
            // update p with model
            p =  alpha * p_0 + (1 - alpha) *  inv_logit(logit(p) + beta);
        }
        {}


    }

    // Compute Function Value
    ret[0] = nll.val();

    // Compute Gradient
    nll.grad();
    ret[1] = p_0.adj();
    ret[2] = alpha_0.adj();
    ret[3] = alpha_1.adj();
    ret[4] = beta.adj();

    // Memory is allocated on a global stack
    stan::math::recover_memory();
    stan::math::ChainableStack::memalloc_.free_all();

    return ret;
}
```

Code: TMB

```
template<class Type>
Type objective_function<Type>::operator() () {


    DATA_VECTOR(you);
    DATA_VECTOR(me);
    DATA_VECTOR(score_diff);


    // parameters:
    PARAMETER(p_0);
    PARAMETER(alpha_0);
    PARAMETER(alpha_1);
    PARAMETER(beta);

    int n_obs = me.size();

    Type p = p_0; // initialise initial model fitted probability
    Type nll = 0; // initialise model negloglik
    vector<Type> p_vec_out(n_obs);

    for (int i = 0; i < n_obs; i++) {

        if (i > 0) {
            // mixing coefficient
            Type alpha = invlogit(alpha_0 + alpha_1 * score_diff[i - 1]);

            // update p with model
            p =  alpha * p_0 + (1 - alpha) *  invlogit(logit(p) + beta);


        }
        nll += -log(get_prob_of_scoreline(me[i], you[i], p));
        // store into vector to be returned to R
        p_vec_out[i] = p;
    }


    REPORT(p_vec_out);
    return nll;

}
```

Usage with optim

```
Rcpp::sourceCpp('cpp_files/reg_20181001_stan_example.cpp')
stan_obj_fun(c(0.592246, 0, 0, 0), df$Me, df$Opponent, df$Me - df$Opponent)
# [1]  5.924815e+01  3.312757e-05  0.000000e+00  0.000000e+00 -2.264476e+00


optim_in_stan <- list()
optim_in_stan$fn <- function(par) stan_obj_fun(par, df$Me, df$Opponent, df$Me - df$Opponent)[1]
optim_in_stan$gr <- function(par) stan_obj_fun(par, df$Me, df$Opponent, df$Me - df$Opponent)[2:5]
optim_in_stan$par <- c(0.592246, 0, 0, 0)
optim_in_stan$method <- 'BFGS'


(optim_out_stan <- do.call(optim, optim_in_stan))
```

Usage with optim

```
Rcpp::sourceCpp('cpp_files/reg_20181001_stan_example.cpp')
stan_obj_fun(c(0.592246, 0, 0, 0), df$Me, df$Opponent, df$Me - df$Opponent)
# [1]  5.924815e+01  3.312757e-05  0.000000e+00  0.000000e+00 -2.264476e+00


optim_in_stan <- list()
optim_in_stan$fn <- function(par) stan_obj_fun(par, df$Me, df$Opponent, df$Me - df$Opponent)[1]
optim_in_stan$gr <- function(par) stan_obj_fun(par, df$Me, df$Opponent, df$Me - df$Opponent)[2:5]
optim_in_stan$par <- c(0.592246, 0, 0, 0)
optim_in_stan$method <- 'BFGS'


(optim_out_stan <- do.call(optim, optim_in_stan))
```

```
> ## check automatic derivatives match numerical
> optim_in_stan$gr(c(0.592246, 0, 0, 0))
[1]  3.312757e-05  0.000000e+00  0.000000e+00 -2.264476e+00
> numDeriv::grad(func = optim_in_stan$fn, x = c(0.592246, 0, 0, 0))
[1]  3.312477e-05  0.000000e+00  0.000000e+00 -2.264476e+00
>
```

Results: Speed of optimisation

```
> microbenchmark(TMB = do.call(optim, optim_in),
+                TMB_numerical = do.call(optim, optim_in_no_jac),
+                Rcpp = do.call(optim, optim_in_Rcpp),
+                stan = do.call(optim, optim_in_stan),
+                # tensorflow = do.call(optim, optim_in_tf),
+                base_R = do.call(optim, optim_in_R))
Unit: milliseconds
          expr        min         lq       mean     median         uq        max neval
           TMB   4.062817   4.389928   7.258069   8.290176   8.428645   13.02282   100
 TMB_numerical  16.188451  22.926824  30.435070  33.284754  35.344212   43.55281   100
          Rcpp  15.311019  18.743908  27.760031  30.739962  32.245148   46.23528   100
          stan   5.886817   7.776003  10.872992  12.075659  12.248499   18.17798   100
        base_R 548.053155 612.511254 919.357057 1051.173354 1068.660768 1309.77592   100
```

Results: Coefficients

```
> (optim_out <- do.call(optim, optim_in))
$`par`
        p_0     alpha_0     alpha_1        beta
  0.5461695 -33.9765827   3.7675724   0.1189786

$value
[1] 57.25596

$counts
function gradient
     161       68
```

Results: Coefficients

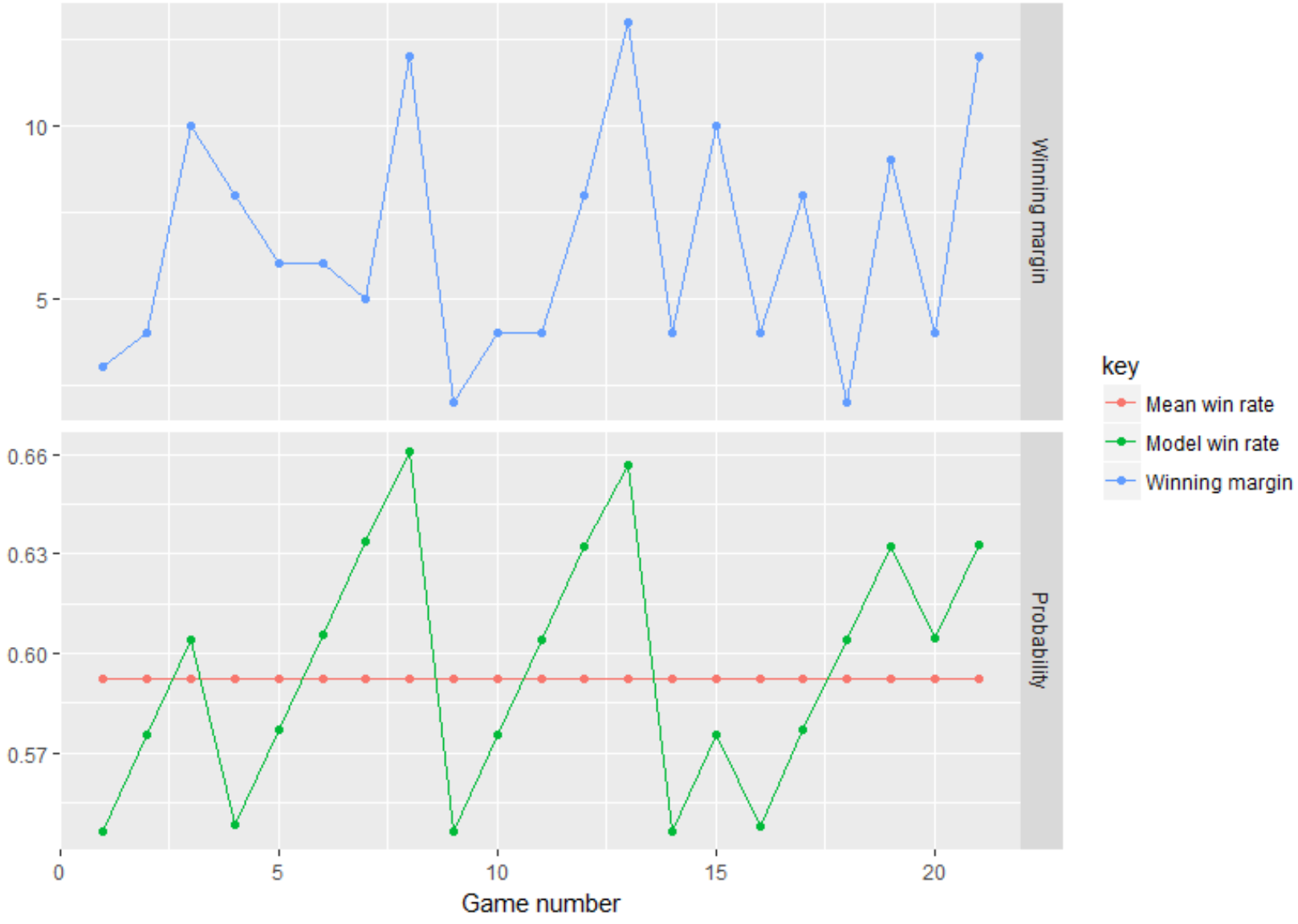
- Recall that s_i is the observed score difference in the i-th game.
- The probability of the original poster winning the $i+1$-th game, given by the model, is:

$$p_{i+1} = 0.546\alpha + (1-\alpha)(\text{expit}(\text{logit}(p_i) + 0.119))$$

where the mixing coefficient α is:

$$\alpha = \text{expit}(-34 + 3.77 s_i)$$

Results: fitted game win probs



The last word

The last word

Your thoughts on whether points and games are independent? Not to mention stationary over time (cumulated games and experience playing each other)? – Mark L. Stone Mar 21 at 15:58

1 I don't think the points and games were independent. The more I won, the more complacent I got and my opponent took advantage of that. – richard Mar 21 at 19:28