Gradient Tree Boosting for Binary Classification

Introduction

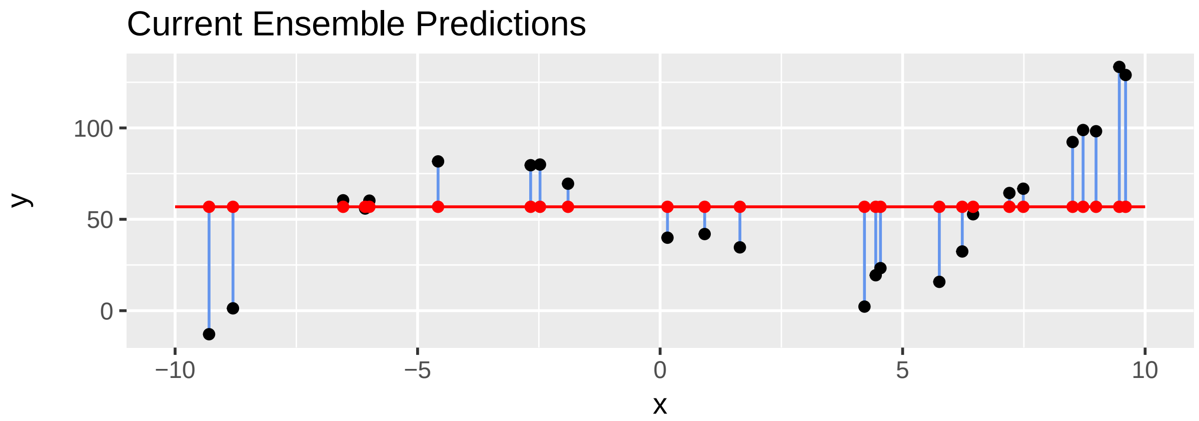
Reminder of gradient tree boosting for regression

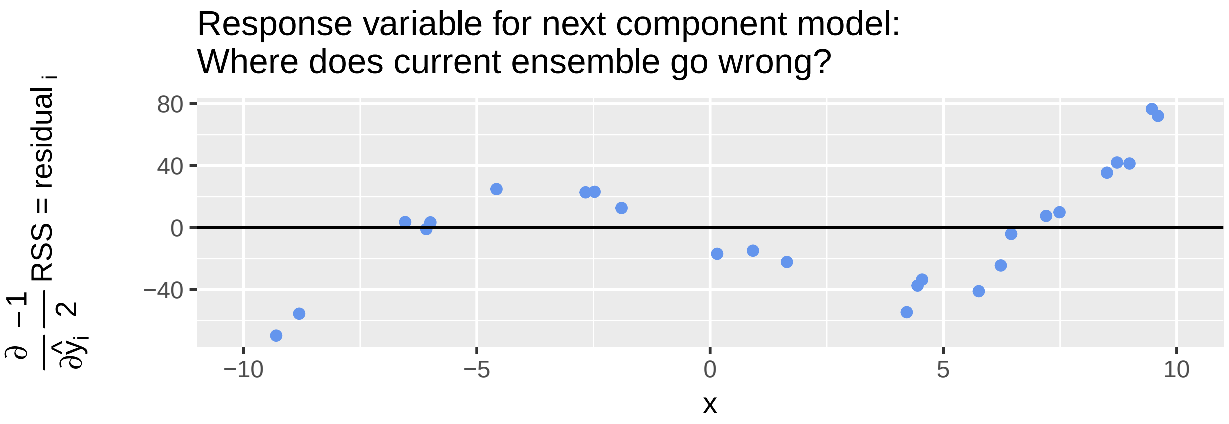
For regression, the gradient boosting procedure looks something like this (different implementations vary slightly):

1. Start with a simple initial model
    - For regression, might start by predicting the mean of the response variable
    - xgboost effectively starts by predicting 0 for all observations
2. Repeat the following:
    a. Fit a model that is specifically tuned to training set observations that the current ensemble does not predict well
    b. Update the ensemble by adding in this new model

We thought about step 2 a in two ways:

- **specific to regression**: fit the new model to the residuals from the current working ensemble
    - Adding the resulting model to the current working ensemble gives an updated ensemble with smaller residuals
- **a more generally applicable idea**: fit the new model to the gradient of $-\frac{1}{2}RSS$.
    - A better prediction has smaller RSS, larger $-\frac{1}{2}RSS$
    - The derivative $\frac{\partial}{\partial \hat{y}_i} - \frac{1}{2}RSS$ tells us how to change the current ensemble's prediction for observation i in order to increase $-\frac{1}{2}RSS$
        - If $\frac{\partial}{\partial \hat{y}_i} - \frac{1}{2}RSS > 0$, we should increase $\hat{y}_i$
        - If $\frac{\partial}{\partial \hat{y}_i} - \frac{1}{2}RSS < 0$, we should decrease $\hat{y}_i$
        - If $\frac{\partial}{\partial \hat{y}_i} - \frac{1}{2}RSS$ is large in magnitude, we should make a big change to $\hat{y}_i$
    - The gradient vector $\left(\frac{\partial}{\partial \hat{y}_1} - \frac{1}{2}RSS, \ldots, \frac{\partial}{\partial \hat{y}_1} - \frac{1}{2}RSS\right)$ collects this information together
    - If we're doing regression and optimizing RSS, this works out to exactly the vector of residuals
    - Fit next model using this gradient as the response.





Running example: Birthweight and bronchopulmonary dysplasia

Can we estimate probability of bronchopulmonary dysplasia (BPD, a lung disease that affects newborns) as a function of the baby's birth weight?

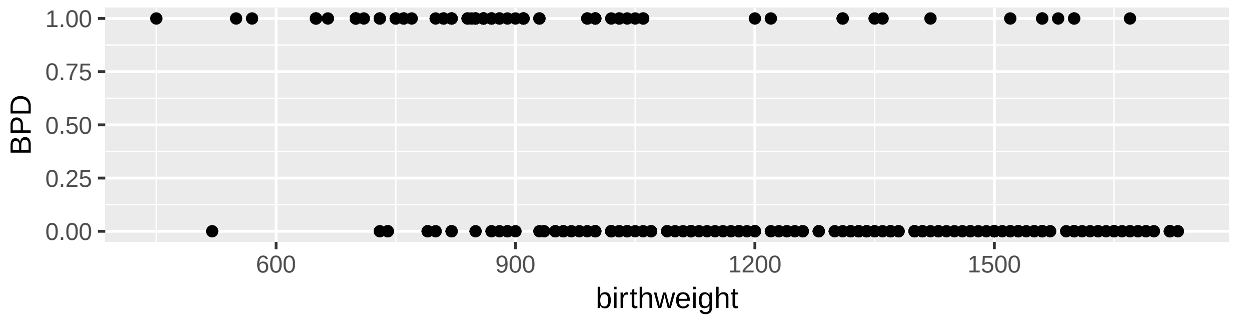
Data from Pagano, M. and Gauvreau, K. (1993). *Principles of Biostatistics.* Duxbury Press.

$$Y_i = \begin{cases} 1 & \text{if baby number } i \text{ has BPD} \\ 0 & \text{otherwise} \end{cases}$$
$$X_i = \text{ birth weight for baby number } i$$

```r
head(bpd)
```

```
## # A tibble: 6 x 2
##   birthweight   BPD
##         <int> <int>
## 1         850     1
## 2        1500     0
## 3        1360     1
## 4         960     0
## 5        1560     0
## 6        1120     0
```

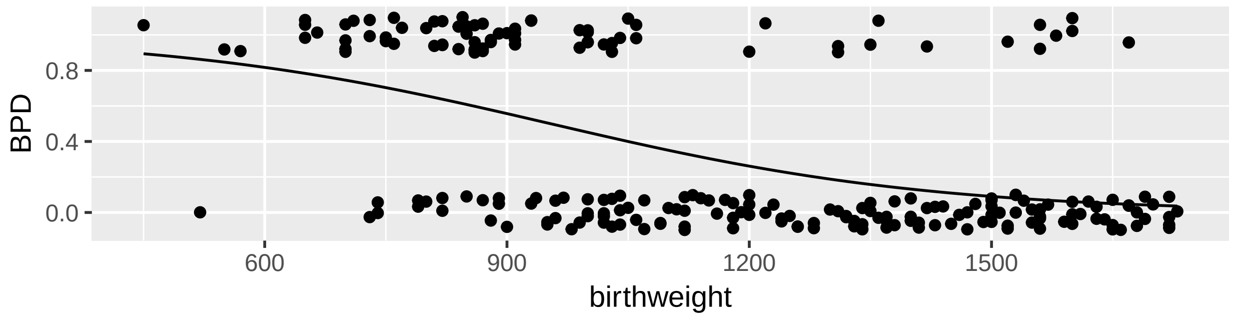
```r
ggplot(data = bpd, mapping = aes(x = birthweight, y = BPD)) +
  geom_point()
```



Reminder of logistic regression

$$P(Y_i = 1|X_i) = p(x_i) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

- Take something that looks like a linear regression model, pass it through the transformation $a \mapsto \frac{e^a}{1+e^a}$ to get probabilities between 0 and 1.

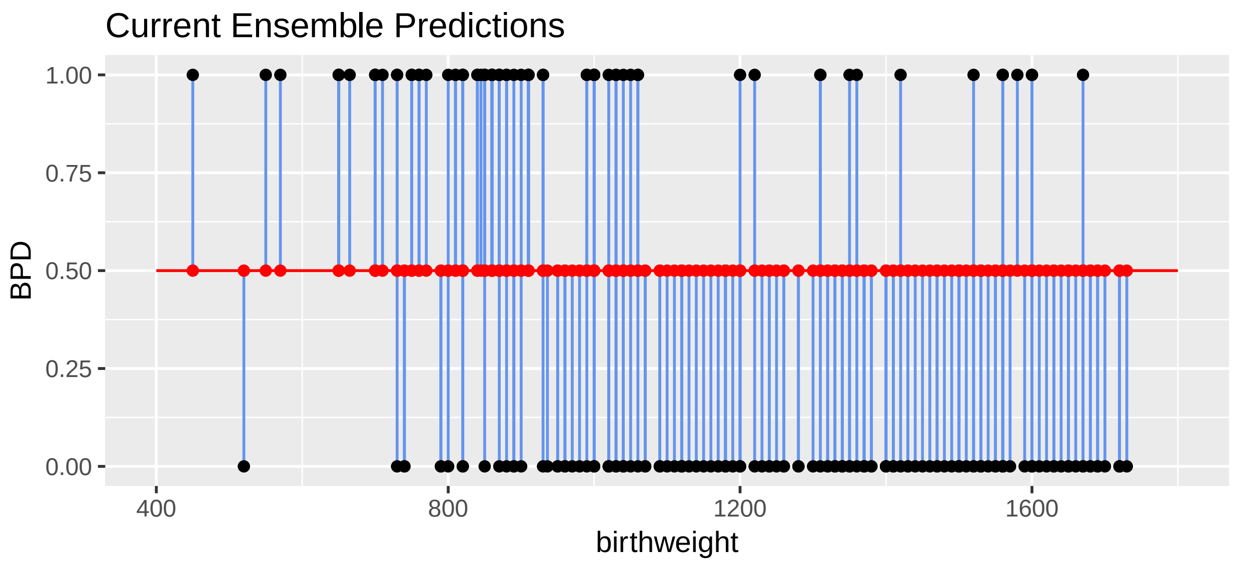


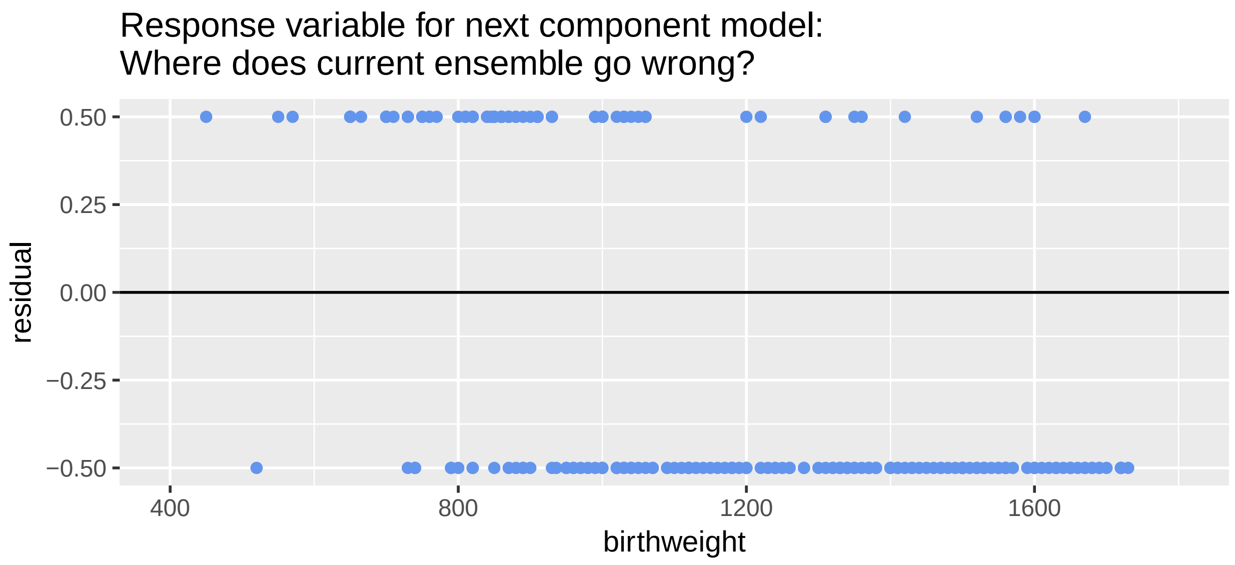
A big idea for today: instead of a linear regression model, we could pass a regression tree model through this transformation.

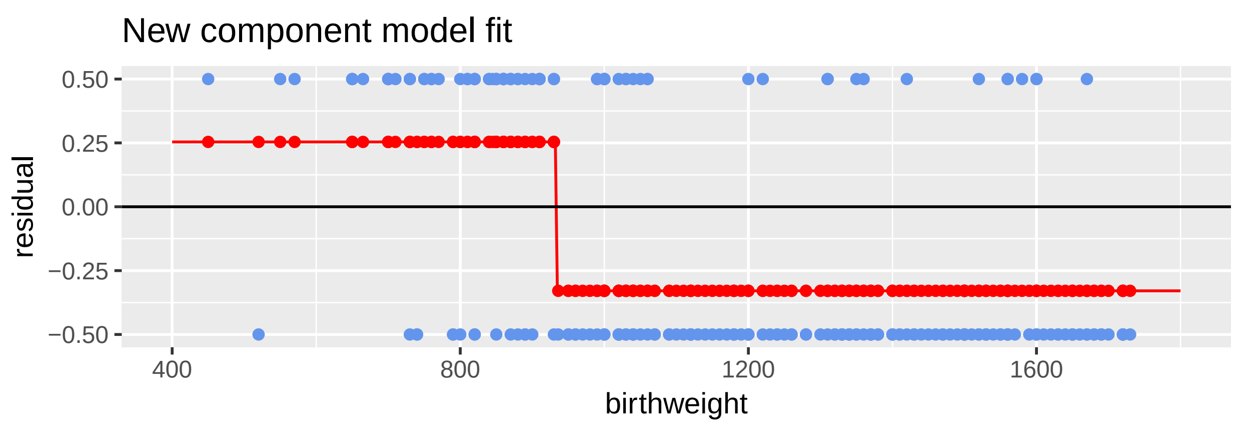
Motivating Pictures for Method:

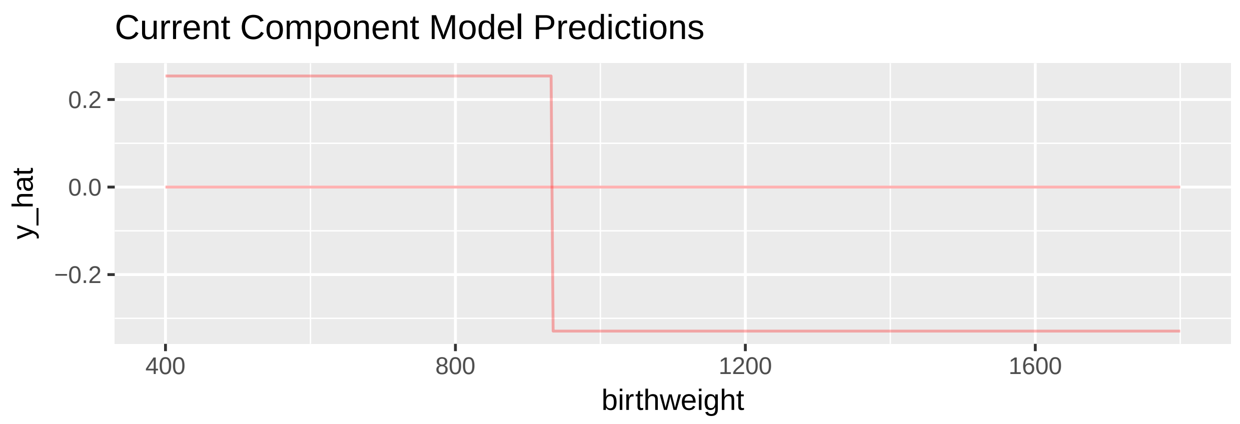
Start with a simple model:

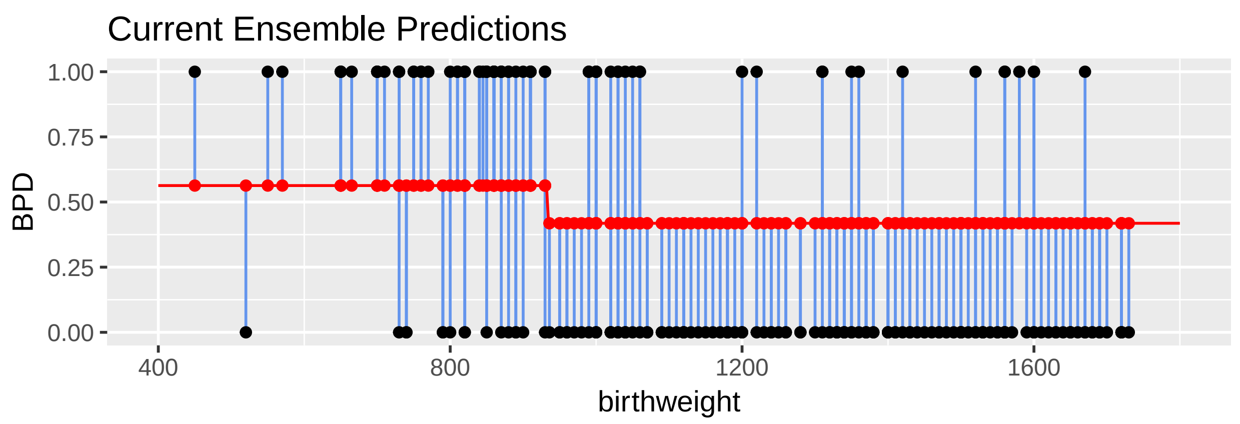
$p(x_i) = \widehat{P}(Y_i = 1|x_i) = \frac{e^0}{1+e^0} = 0.5$ for all values of x_i.

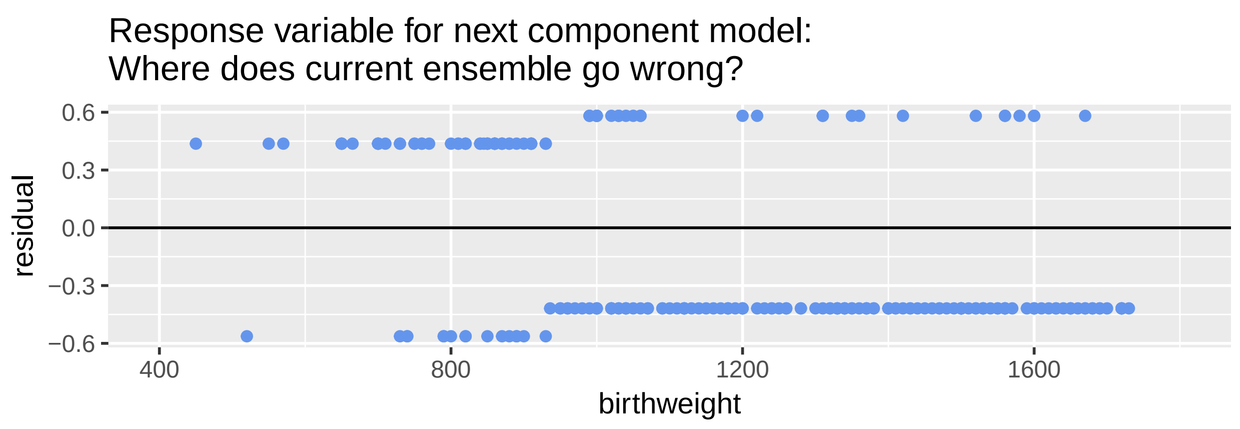


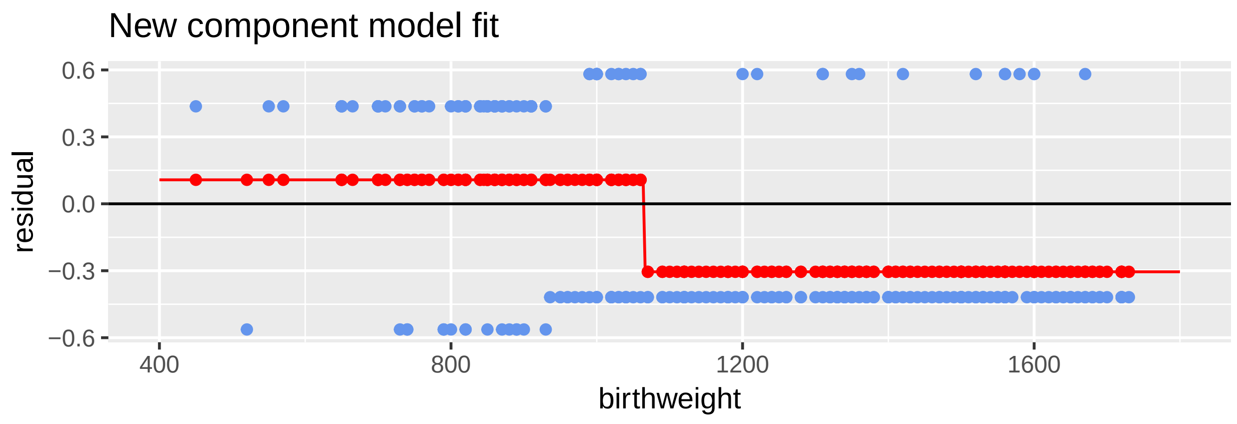


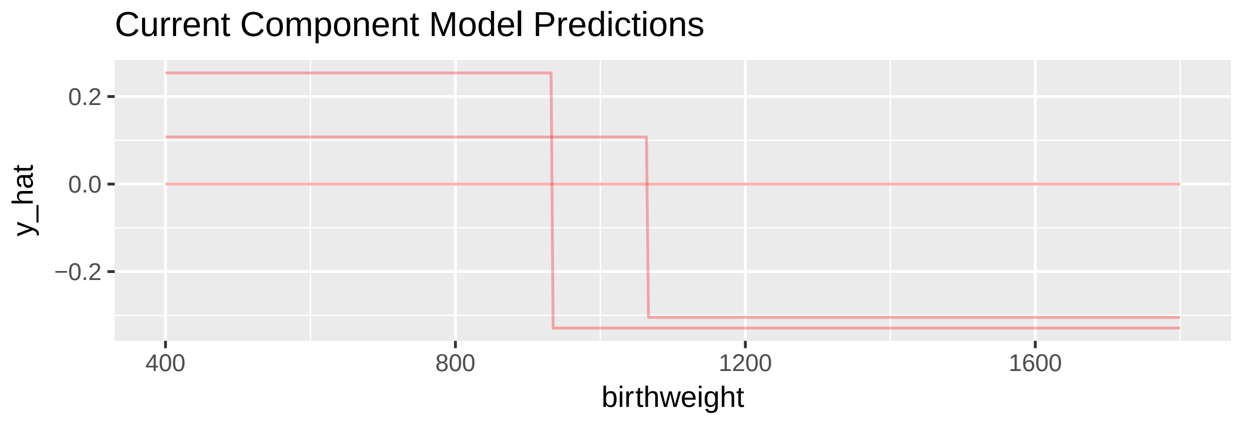

New component model fit
residual
0.50
0.25
0.00
−0.25
−0.50
400
800
1200
1600
birthweight

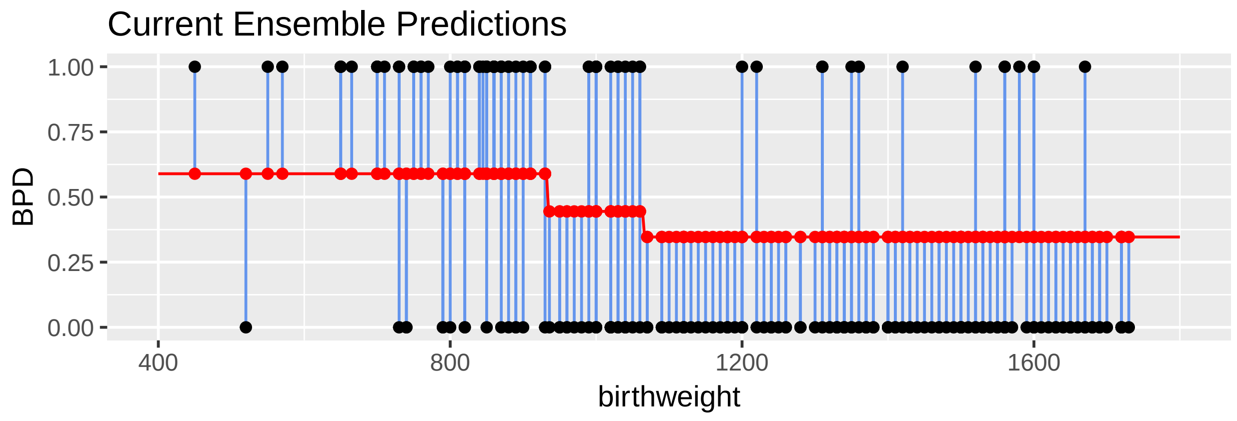

Current Component Model Predictions
y_hat
0.2
0.0
−0.2
400
800
1200
1600
birthweight

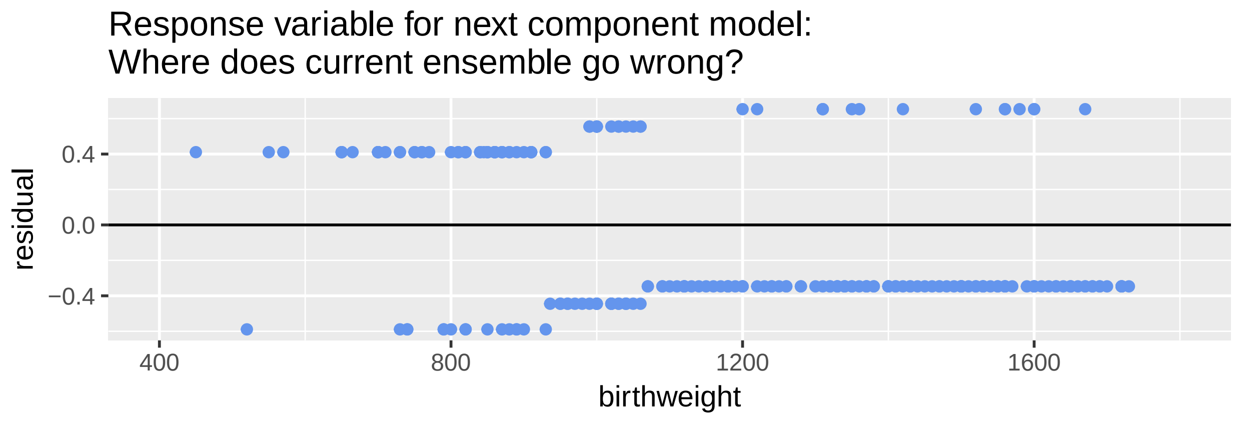

Current Ensemble Predictions
BPD
1.00
0.75
0.50
0.25
0.00
400
800
1200
1600
birthweight

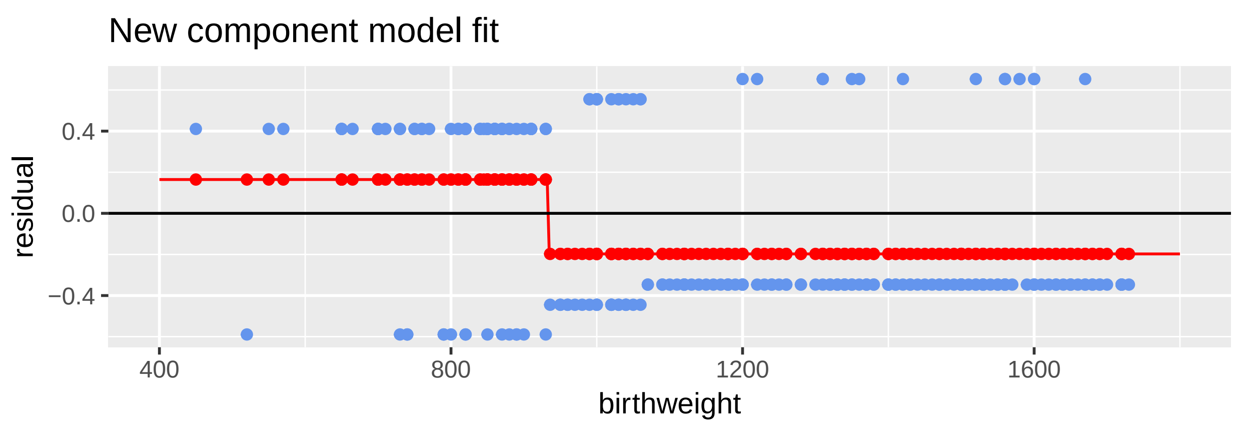

Response variable for next component model:
Where does current ensemble go wrong?
residual
0.6
0.3
0.0
−0.3
−0.6
400
800
1200
1600
birthweight

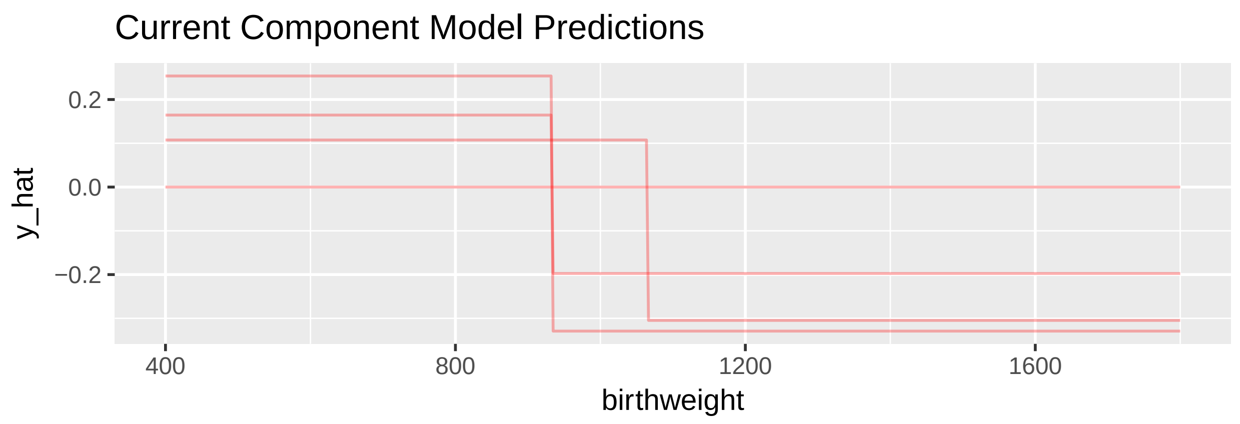

New component model fit
residual
0.6
0.3
0.0
−0.3
−0.6
400
800
1200
1600
birthweight

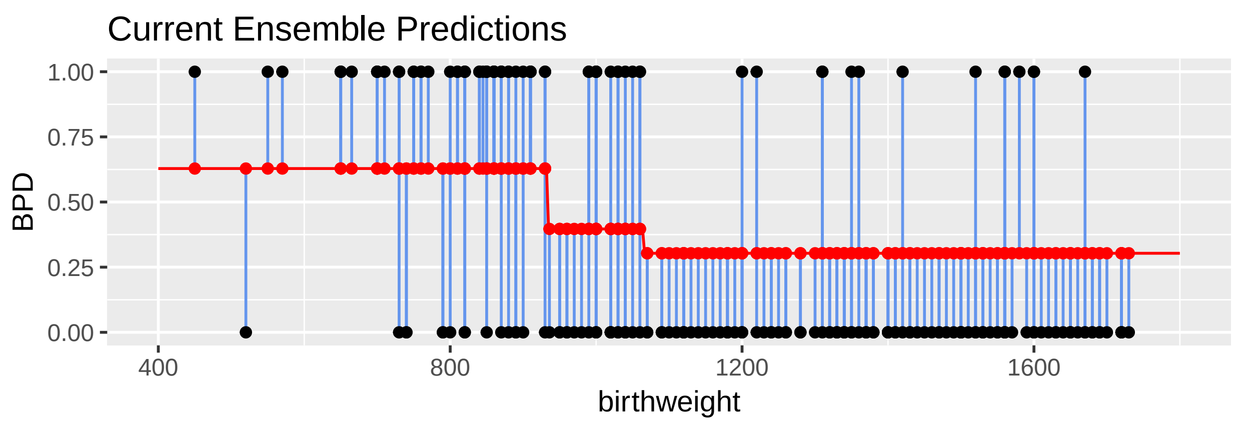

Current Component Model Predictions
y_hat
0.2
0.0
−0.2
400
800
1200
1600
birthweight

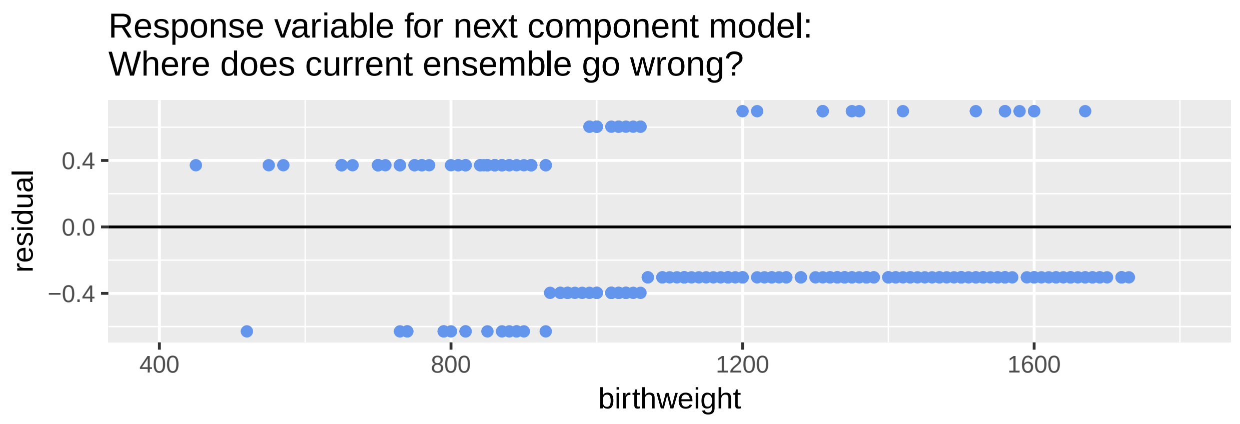

Current Ensemble Predictions
BPD
1.00
0.75
0.50
0.25
0.00
400
800
1200
1600
birthweight


Response variable for next component model:
Where does current ensemble go wrong?
residual
0.4
0.0
−0.4
400
800
1200
1600
birthweight

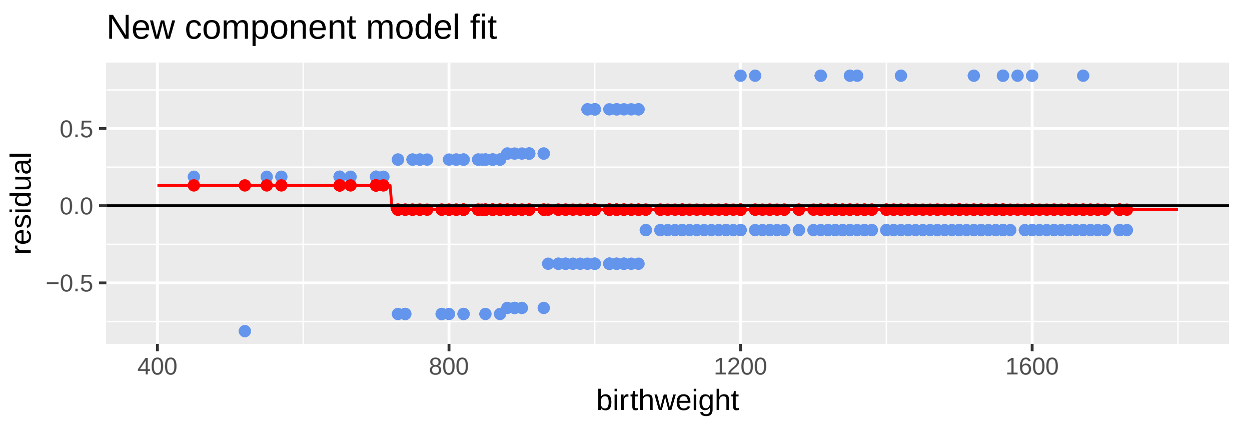

New component model fit
residual
0.4
0.0
−0.4
400
800
1200
1600
birthweight

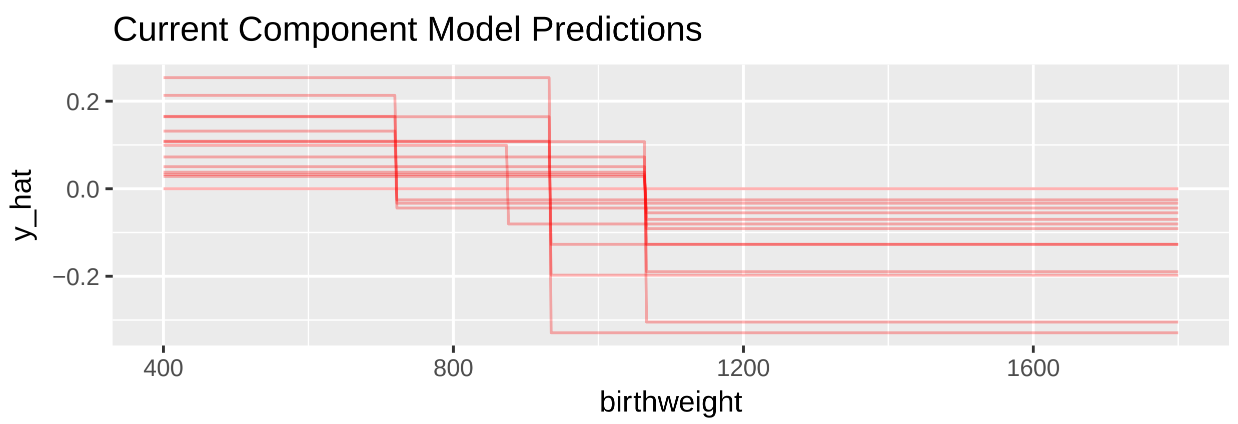

Current Component Model Predictions
y_hat
0.2
0.0
−0.2
400
800
1200
1600
birthweight

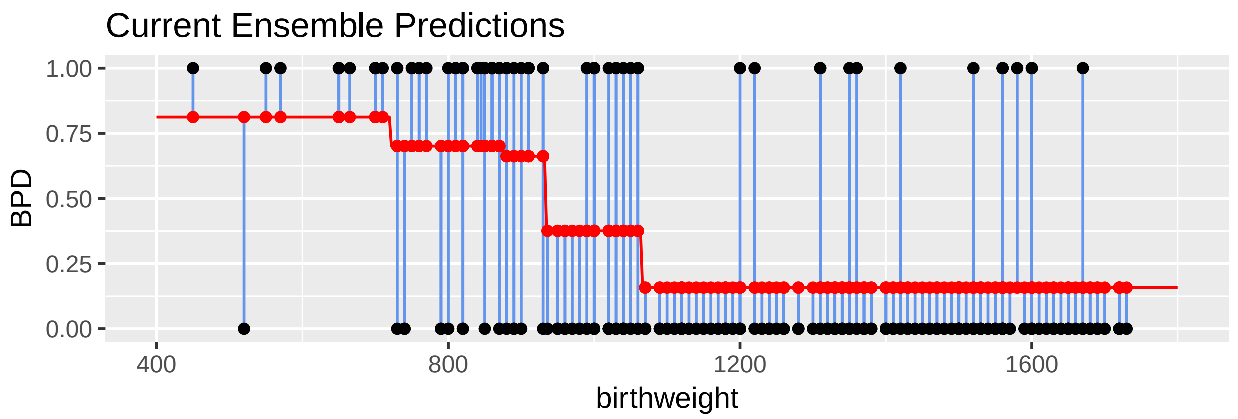

Current Ensemble Predictions
BPD
1.00
0.75
0.50
0.25
0.00
400
800
1200
1600
birthweight

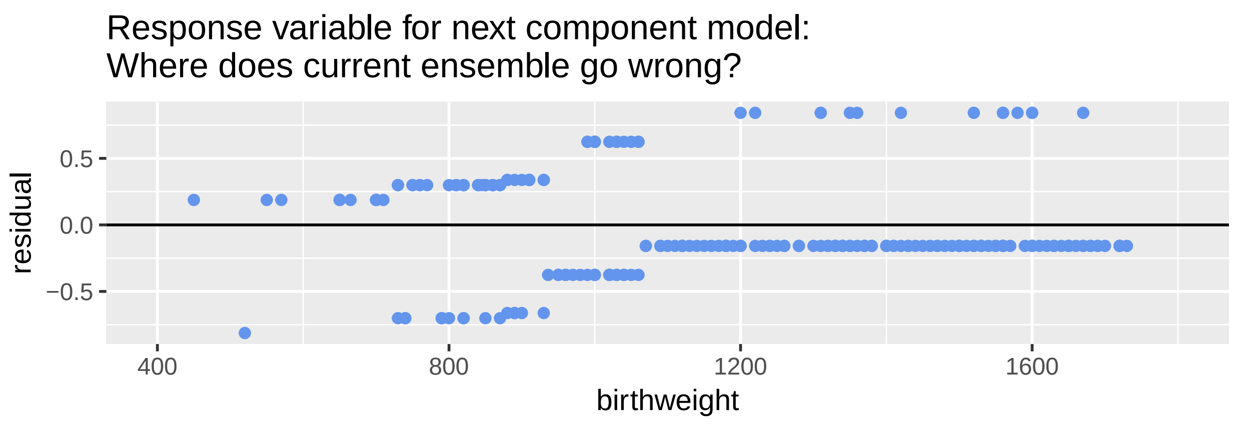

Response variable for next component model:
Where does current ensemble go wrong?
residual
0.4
0.0
−0.4
400
800
1200
1600
birthweight

. . . 10 more. . .


New component model fit
residual
0.5
0.0
−0.5
400
800
1200
1600
birthweight


Current Component Model Predictions
y_hat
0.2
0.0
−0.2
400
800
1200
1600
birthweight


Current Ensemble Predictions
BPD
1.00
0.75
0.50
0.25
0.00
400
800
1200
1600
birthweight


Response variable for next component model:
Where does current ensemble go wrong?
residual
0.5
0.0
−0.5
400
800
1200
1600
birthweight

Detour: Details of Estimation for Logistic Regression

Formal Model Statement

$$Y_i \sim \text{Bernoulli}(p(x_i)) \text{ where}$$
$$p(x_i) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}},$$

with each Y_i independent of the others.

This means that according to the model, $P(Y_i = 1) = p(x_i)$ and $P(Y_i = 0) = 1 - p(x_i)$.

Joint Probability of Observed Data

For a fixed value of β_0 and β_1, what is the probability assigned to the observed data $y_1, \ldots, y_n$?

$$\begin{aligned}
P(Y_1 = y_1, Y_2 = y_2, \ldots, Y_n = y_n | x_1, \ldots, x_n) &= P(Y_1 = y_1|x_1)P(Y_2 = y_2|x_2) \cdots P(Y_n = y_n|x_n) \\
&= \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} \{1 - p(x_i)\} \\
&= \prod_{i:y_i=1} \frac{e^{\beta_0+\beta_1 x_i}}{1 + e^{\beta_0+\beta_1 x_i}} \prod_{i:y_i=0} \frac{1}{1 + e^{\beta_0+\beta_1 x_i}}
\end{aligned}$$

For example, the parameter estimates for our model fit are $\hat{\beta}_0 = 4.03429128$ and $\hat{\beta}_1 = -0.00422914$.

The joint probability assigned to the data is:

```
bpd_augmented <- bpd %>%
  mutate(
    est_prob_Y_eq_1 =
      exp(4.03429128 - 0.00422914 * birthweight) / (1 + exp(4.03429128 - 0.00422914 * birthweight)),
    est_prob_Y_eq_y = ifelse(BPD == 1, est_prob_Y_eq_1, 1 - est_prob_Y_eq_1)
  )

head(bpd_augmented)
```

```
## # A tibble: 6 x 4
##   birthweight   BPD est_prob_Y_eq_1 est_prob_Y_eq_y
##         <int> <int>           <dbl>           <dbl>
## 1         850     1          0.608           0.608
## 2        1500     0          0.0903          0.910
## 3        1360     1          0.152           0.152
## 4         960     0          0.494           0.506
## 5        1560     0          0.0715          0.928
## 6        1120     0          0.331           0.669
```

```
nrow(bpd_augmented)
```

```
## [1] 223
```

```
prod(bpd_augmented$est_prob_Y_eq_y)
```

```
## [1] 2.628358e-49
```

```
0.5^nrow(bpd_augmented)
```

```
## [1] 7.418412e-68
```

Maximum likelihood estimation

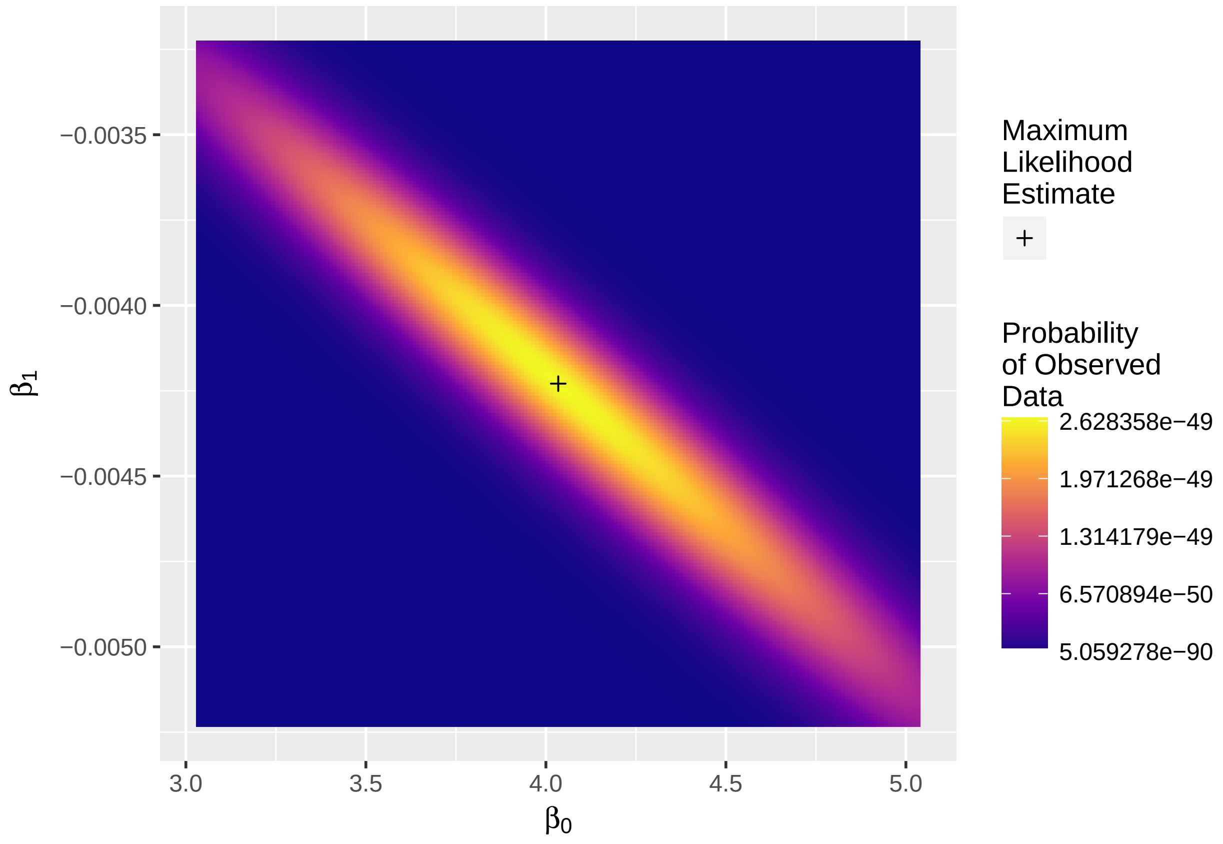
The best choice of β_0 and β_1 assigns highest probability to the observed data.

$\max_{\beta_0,\beta_1} \prod_{i:y_i=1} \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} \prod_{i:y_i=0} \frac{1}{1+e^{\beta_0+\beta_1 x_i}}$

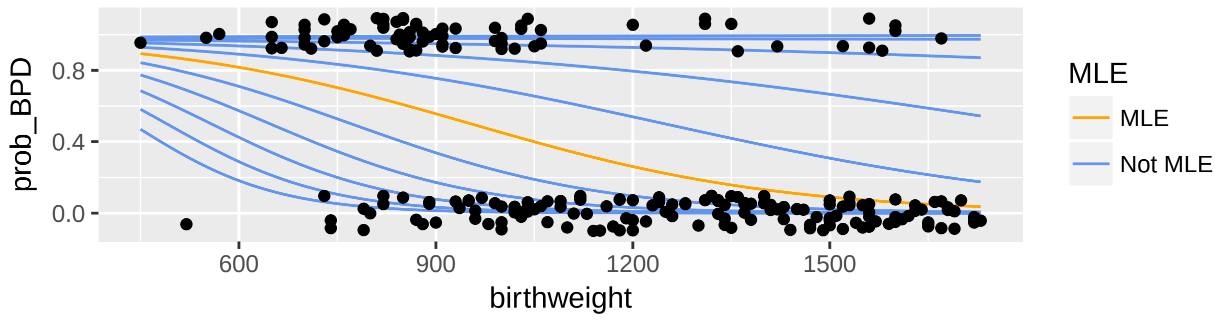
The function we are optimizing here is called the **likelihood function**:

$\text{Likelihood}(\beta_0, \beta_1) = \prod_{i:y_i=1} \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} \prod_{i:y_i=0} \frac{1}{1+e^{\beta_0+\beta_1 x_i}}$

Picture 1:



Picture 2:



If you take Stat 343, we'll talk more about exactly how to find the MLE's for logistic regression then.

Details of Estimation for Gradient Tree Boosting, Binary Classification

Motivation: Logistic transformation of (sum of) regression tree predictions

$$
\begin{aligned}
Y_i &\sim \text{Bernoulli}(p(x_i)) \text{ where} \\
p(x_i) &= \frac{e^{\text{prediction from a sum of regression trees}}}{1 + e^{\text{prediction from a sum of regression trees}}}
\end{aligned}
$$

Formulation in terms of "likelihood"

Let's denote the "prediction from a sum of regression trees" by $\hat{a}_i$

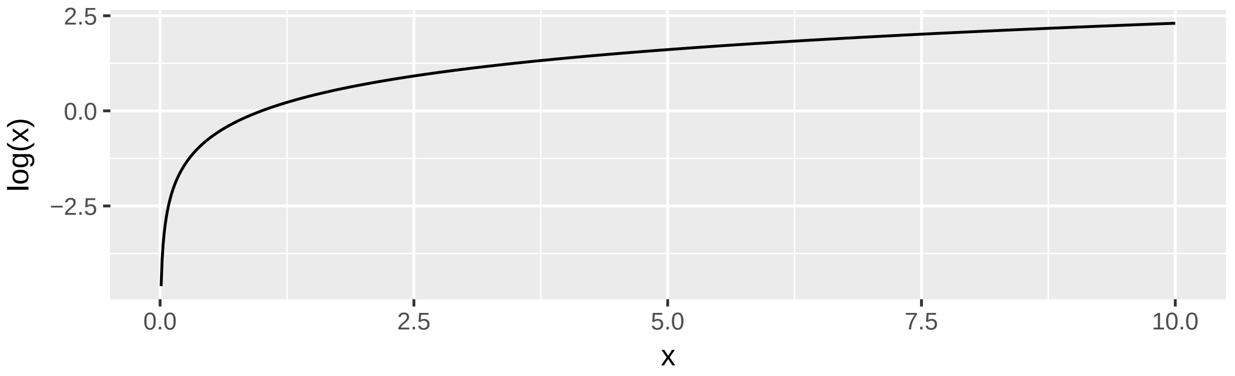
Consider the "likelihood" expressed in terms of the $\hat{a}_i$:

$$\prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} \{1 - p(x_i)\} = \prod_{i:y_i=1} \frac{e^{\hat{a}_i}}{1 + e^{\hat{a}_i}} \prod_{i:y_i=0} \frac{1}{1 + e^{\hat{a}_i}}$$

In each boosting iteration, our goal is to update the ensemble's predicted values $\hat{a}_i$ so as to increase this likelihood.

Formulation in terms of log-"likelihood"

- $\log(x)$ is an increasing function of x:



- That means that a choice of the $\hat{a}_i$ that maximizes the likelihood also maximizes the log of the likelihood.
  - Suppose $L(a) < L(\hat{a}_i)$ for all other values of a
  - Then $\log\{L(a)\} < \log\{L(\hat{a}_i)\}$ for all other values of a
  - (In the plot above, x matches up with $L(a)$)
- Instead of thinking about maximizing the likelihood, let's think about maximizing the log of the likelihood. Two reasons:
  1. Emotional convenience (we'll get answers that look reasonable, and then we'll feel happier)
  2. Computational stability (probabilities are very close to 0, and computers are bad at numbers very close to 0; the logarithms are easier for computers to deal with)

The log of the likelihood, expressed in terms of $\hat{a}_i$ is:

$$
\begin{aligned}
\log\left[\prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} \{1 - p(x_i)\}\right] &= \sum_{i:y_i=1} \log\left[\frac{e^{\hat{a}_i}}{1 + e^{\hat{a}_i}}\right] + \sum_{i:y_i=0} \log\left[\frac{1}{1 + e^{\hat{a}_i}}\right] \\
&= \sum_{i:y_i=1} \left\{\log\left(e^{\hat{a}_i}\right) - \log(1 + e^{\hat{a}_i})\right\} + \sum_{i:y_i=0} \left\{\log(1) - \log(1 + e^{\hat{a}_i})\right\} \\
&= \sum_{i:y_i=1} \left\{\hat{a}_i - \log(1 + e^{\hat{a}_i})\right\} + \sum_{i:y_i=0} \left\{-\log(1 + e^{\hat{a}_i})\right\}
\end{aligned}
$$

Derivatives of the log-likelihood

- Let's take the derivative of this with respect to $\hat{a}_{i^*}$, for a particular observation number i^*.
- This will tell us how to change our current ensemble prediction $\hat{a}_{i^*}$ for observation i^* to increase the probability assigned to that observation.

First, suppose that $y_{i^*} = 1$:

$$\begin{aligned}
&\frac{\partial}{\partial \hat{a}_{i^*}} \left[\sum_{i:y_i=1} \left\{ \hat{a}_i - \log(1 + e^{\hat{a}_i}) \right\} + \sum_{i:y_i=0} \left\{ -\log(1 + e^{\hat{a}_i}) \right\} \right] \\
&= \frac{\partial}{\partial \hat{a}_{i^*}} \left\{ \hat{a}_{i^*} - \log(1 + e^{\hat{a}_{i^*}}) \right\} \\
&= 1 - \frac{1}{1 + e^{\hat{a}_{i^*}}} e^{\hat{a}_{i^*}} \\
&= 1 - p(x_{i^*})
\end{aligned}$$

This is the "residual" $y_{i^*} - p(x_i)$!

- Always positive (if $y_{i^*} = 1$, always want a higher $\hat{P}(Y_i = 1|x_i)$)
- Larger in magnitude the more "incorrect" the predictive probability is.

Second, suppose that $y_{i^*} = 0$:

$$\begin{aligned}
&\frac{\partial}{\partial \hat{a}_{i^*}} \left[\sum_{i:y_i=1} \left\{ \hat{a}_i - \log(1 + e^{\hat{a}_i}) \right\} + \sum_{i:y_i=0} \left\{ -\log(1 + e^{\hat{a}_{i^*}}) \right\} \right] \\
&= \frac{\partial}{\partial \hat{a}_{i^*}} \left\{ -\log(1 + e^{\hat{a}_{i^*}}) \right\} \\
&= -\frac{1}{1 + e^{\hat{a}_{i^*}}} e^{\hat{a}_{i^*}} \\
&= -p(x_{i^*})
\end{aligned}$$

Again, this is the "residual" $y_{i^*} - p(x_{i^*})$!

- Always negative (if $y_{i^*} = 0$, always want a smaller $\hat{P}(Y_{i^*} = 1|x_{i^*})$)
- Larger in magnitude the more "incorrect" the predictive probability is.

Final Procedure: Gradient Tree Boosting

Our ensemble procedure builds up a sum of regression trees $\hat{a}(x)$, which is passed through the logistic transformation to find the estimated probability of being in class 1:

$\hat{P}(Y_i = 1|x_i) = \frac{e^{\hat{a}(x_i)}}{1+e^{\hat{a}(x_i)}}$

Algorithm:

1. Initialize $\hat{a}^{(0)}(x) = 0$ for all x.
2. For $b = 1, \ldots, B$:
   a. Compute the predicted probability of being in class 1 for each observation i, based on the ensemble that was created after the previous step: $\hat{P}^{(b-1)}(Y_i = 1|x_i) = \frac{e^{\hat{a}^{(b-1)}(x_i)}}{1+e^{\hat{a}^{(b-1)}(x_i)}}$
   b. Fit a regression tree $\hat{g}^{(b)}(x)$ using the residuals $Y_i - \hat{P}^{(b-1)}(Y_i = 1|x_i)$ as the response.
   c. Update the ensemble by adding in this new model: $\hat{a}^{(b)}(x) = \hat{a}^{(b-1)}(x) + \hat{g}^{(b)}(x)$

Note:

- The above procedure can be thought of as using a linear (first-order Taylor series) approximation to the log-likelihood
- xgboost uses a quadratic (second-order Taylor series) approximation

Fit with xgboost

```
set.seed(34592)
class(bpd$BPD)
```

```
## [1] "integer"
```

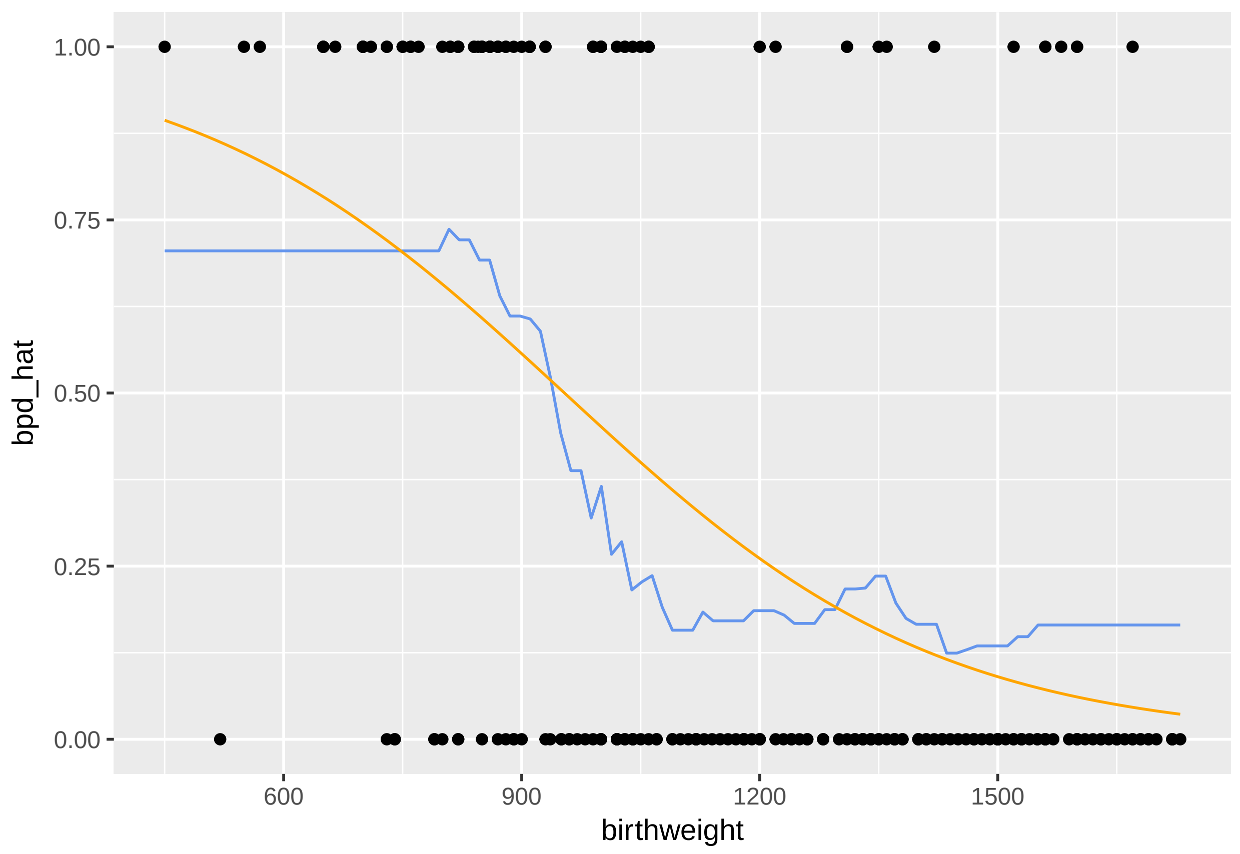
```
# Convert BPD variable in bpd data frame to a factor so xgboost knows to
# treat this as a classification problem.
bpd <- bpd %>% mutate(BPD = factor(BPD))
class(bpd$BPD)
```

```
## [1] "factor"
```

```
library(caret)
xgb_fit <- train(
  BPD ~ birthweight,
  data = bpd,
  method = "xgbTree",
  trControl = trainControl(method = "cv", number = 10, returnResamp = "all"),
  tuneGrid = expand.grid(
    nrounds = c(5, 10, 20, 30, 40, 50, 100),
    eta = c(0.001, 0.01, 0.05, 0.1), # learning rate; 0.3 is the default
    gamma = 0, # minimum loss reduction to make a split; 0 is the default
    max_depth = 1:3, # how deep are our trees?
    subsample = c(0.05, 0.1, 0.25, 0.5, 1), # proportion of observations to use in growing each tree
    colsample_bytree = 1, # proportion of explanatory variables used in each tree
    min_child_weight = 1 # think of this as how many observations must be in each leaf node
  )
)

xgb_fit$results %>% filter(Accuracy == max(Accuracy))
```

```
##   eta max_depth gamma colsample_bytree min_child_weight subsample nrounds
## 1 0.1         2     0                1                1       0.1      50
##    Accuracy     Kappa AccuracySD   KappaSD
## 1 0.8065688 0.5592598 0.05803928 0.1454808
```

```
birthweight_grid <- data.frame(birthweight = seq(from = min(bpd$birthweight), to = max(bpd$birthweight), leng

plot_df <- birthweight_grid %>%
  mutate(
    bpd_hat = predict(xgb_fit, type = "prob", newdata = birthweight_grid)[, 2]
  )

ggplot() +
  geom_line(data = plot_df,
    mapping = aes(x = birthweight, y = bpd_hat),
    color = "cornflowerblue") +
  geom_line(data = logistic_preds %>% filter(MLE == "MLE"),
    mapping = aes(x = birthweight, y = prob_BPD, group = factor(beta_1)),
    color = "orange") +
  geom_point(data = bpd, mapping = aes(x = birthweight, y = as.numeric(as.character(BPD))))
```


1.00
0.75
0.50
0.25
0.00
bpd_hat
600
900
1200
1500
birthweight