Applied Statistical Programming - The EM Algorithm

Amaan, Alex, Patrick, Peter

4/13/2022

Write R and Rcpp code to answer the following questions. Write the code, and then show what the computer returns when that code is run. Thoroughly comment your solutions.

Complete this assignment before 10:00am on Wednesday, April 20. Submit the R implementation as an Rmarkdown and the knitted PDF to Canvas. Have one group member submit the activity with all group members listed at the top. The Rcpp portion will be given to you as your final assignment.

In-class Background: The Expectation-Maximization Algorithm

The goal of this in-class exercise is to implement an ensemble of models. You will combine forecasts of US presidential elections using ensemble Bayesian model averaging (EBMA). To do this, you must decide how to weight each component of the forecast in the prediction. The collection of these weighted forecasts form the ensemble, and you will use something called the EM (expectation-maximization) algorithm.

The task is to choose values w_k that maximize the following equation:

$$p(y|f_1^{s|t^\star}, ..., f_K^{s|t^\star}) = \sum_{k=1}^{N} w_k N(f_k^{t^\star}, \sigma^2) \tag{1}$$

For the remainder of this assignment, assume that the parameter σ^2 is known and that $\sigma^2 = 1$.

The first step of the EM algorithm is to estimate the latent quantity $\hat{z}_k^t$ that represents the probability that observation t was best predicted by model k.

$$\hat{z}_k^{(j+1)t} = \frac{\hat{w}_k^{(j)} N(y^t|f_k^t, 1)}{\sum_{k=1}^{N} \hat{w}_k^{(j)} N(y^t|f_k^t, 1)} \tag{2}$$

In this equation, j is the particular iteration of the EM algorithm, and $N(y^t|f_k^t, 1)$ is the normal cumulative distribution function evaluated at the observed election outcome (`dnorm(y, ftk, 1)`).

The second step of the EM algorithm is to estimate the expected value of the weights assuming that all $\hat{z}_k^t$ are correct.

$$\hat{w}_k^{(j+1)} = \frac{1}{n} \sum_t \hat{z}_k^{(j+1)t} \tag{3}$$

The estimation procedure is as follows:

1. Start with the assumption that all models are weighted equally.
2. Calculate $\hat{z}_k^{(j+1)t}$ for each model for each election.
3. Calculate $\hat{w}_k^{(j+1)}$ for each model.
4. Repeat steps 2-3 twenty times.

Complete the preceding tasks in `R` alone.

ANSWERS: R Section.

FIRST: find test data to use on our model.

- we use the Expectation-Maximization (EM) algorithm on datasets that combine multiple linear models. In this case, we combine different forecasts of US presidential elections using EBMA.
- To learn more about EBMA forecasts and (hopefully) find some toy data to use with our functions, let's install and investigate the `EBMAforecast` package.

```
## Install package:

# install.packages('EBMAforecast')
library(EBMAforecast)
```

```
## Warning: package 'EBMAforecast' was built under R version 4.1.3
```

```
## Read documentation:

# ?EBMAforecast

## Find data used in demo(PresForecast):

# demo(presForecast)
```

- Let's load the `presidentialForecast` data from the `EBMAforecast` package:

```
## Load `presidentialForecast` data from `EBMAforecast` package:
data("presidentialForecast")
# ?presidentialForecast

## OUTCOME VARIABLE: incumbent-party vote share in each presidential election.

df <- presidentialForecast
```

SECOND: for the remaining R code, let's first develop code for each of the steps referenced above before properly structuring them in a for-loop.

1. **Start with the assumption that all models are weighted equally**.

NOTE: recall from Montgomery, Hollenbach, and Ward (2012) that the $w_k \in [0, 1]$'s are model probabilities associated with each component model's predictive performance. In other words, the $w_k \in [0, 1]$'s are weights associated each each model such that $\sum_{k=1}^{K} w_k = 1$.

ASSUMED DATA STRUCTURE: dataframe with:

- `K` columns, which include `K-1` columns of models and `1` column of actual results.
- `n` rows of predictions (substantively, these are the predicted incumbent-party vote share for each presidential election).

```
## Find number of models:
K <- dim(df)[2] - 1

## Define w_hat with all models weighted equally.
w_hat <- replicate(K, 1/K)

## COMPLETE!
```

2. **Calculate $\hat{z}_k^{(j+1)t}$ for each model for each election**.
3. **Calculate $\hat{w}_k^{(j+1)}$ for each model**.

THIRD: perform step (4) above by creating a for-loop that repeats steps 2-3 twenty times.

```
# Eventually wrap everything in a loop, or apply function?

# Calculate zHat

# I'm not sure if the indices are in the right places. This is mostly because
# I'm not entirely sure what's going on
zHat[i + 1] <- (wHat[i] * dnorm(y[i], ftk[i], 1)) / sum(wHat * dnorm(y, ftk, 1))

# Calculate wHat

# Not sure for the same reasons as above.
wHat[i + 1] <- sum(zHat) / length(zHat)
```

Assignment: Rcpp Practice

1. Write an Rcpp function that will calculate the answer to Equation (2). The output will be a matrix.
2. Write an Rcpp function that will calculate the answer to Equation (3). The output will be a vector.
3. Write an Rcpp function that will complete the entire algorithm.