

# Bayesian Factorization Machines with Stan and R

Adam M. Lauretig

12/18/2019

# Welcome

- ▶ Talk: Factorization machines:
  - ▶ What are they?
  - ▶ How do we fit them?
- ▶ Stan code + simulations!
  - ▶ Start with simple model
  - ▶ Next, hierarchical model

## Who I Am

- ▶ Senior Data Scientist, JUST Capital
- ▶ Previously, Ph.D. in political science (The Ohio State University)
- ▶ Word Embeddings, Causal Inference, Bayesian Statistics, Discrete Choice Models

## Why We Model

- ▶ Descriptive Inference: is there an association between  $X$  and  $y$ ?
- ▶ Prediction: with  $X$  and  $y$ , what is  $y$  for a new  $X$ ?
- ▶ Causal Inference: does changing  $X$ , change  $y$ ?
- ▶ Shameless plug: Book Chapter **insert link**

## Latent Variable Models/Unsupervised models

- ▶ Learning parameters to reconstruct observed data
- ▶ Ex: Principal Components Analysis, Factor Analysis, Matrix Factorization, etc
- ▶ Data  $X_{N \times J}$  is decomposed into two low rank matrices:  $\gamma_{N \times K}$  and  $\delta_{K \times J}$
- ▶ Various assumptions about the structure of  $\gamma$  and  $\delta$

## Factorization Machines

- ▶ Combine regression with a latent variable model on the residuals

But Why?

# Factorization Machines

- ▶ Regression Model, for one observation
- ▶ Categorical predictors  $\mathbf{x}_{n \in N}, \mathbf{x}_{j \in J}$
- ▶ Outcome  $y$
- ▶ Parameters  $\beta$

$$y_{nj} = \mathbf{x}_n \beta_1 + \mathbf{x}_j \beta_2 + \varepsilon_{nj}$$

```
m1 <- lm(y ~ factor(group_1) + factor(group_2))
```

- ▶ With Interactions

$$y_{nj} = \mathbf{x}_n \beta_1 + \mathbf{x}_j \beta_2 + \mathbf{x}_n \times \mathbf{x}_j \beta_3 + \varepsilon_{nj}$$

```
m1 <- lm(y ~ factor(group_1) + factor(group_2) + factor(group_1) * factor(group_2))
```

# Factorization Machines

## Problems:

- ▶ We can only estimate  $\beta_3$  for observed interactions
- ▶ As  $N$  and  $J$  grow,  $\beta_3$  increases with size  $N * J$

# Factorization Machines

Solution!

- ▶ Replace  $\beta_3$  with the dot product of low-rank latent factors:
- ▶  $\gamma_{N \times K}$
- ▶  $\delta_{J \times K}$
- ▶  $\beta_3$  is now  $\gamma_n \cdot \delta_j^\top$

# Factorization Machines

- ▶ Interaction model:

$$y_{nj} = \mathbf{x}_n \beta_1 + \mathbf{x}_j \beta_2 + \boldsymbol{\gamma}_n \cdot \boldsymbol{\delta}_j^\top + \varepsilon_{nj}$$

## Factorization Machines

Basic model - Each element of  $\delta_j$  and  $\gamma_n$  is distributed standard normal:  $\mathcal{N}(0, 1)$

$$y_{nj} \sim \mathcal{N}(\mathbf{x}_n \beta_1 + \mathbf{x}_j \beta_2 + \boldsymbol{\gamma}_n \cdot \boldsymbol{\delta}_j^\top, 1)$$

$$\boldsymbol{\beta} \sim \mathcal{N}(0, \sigma^2)$$

$$\gamma_{n,k} \sim \mathcal{N}(0, 1)$$

$$\delta_{j,k} \sim \mathcal{N}(0, 1)$$

# Simulating Data:

First, the regression component:

```
seed_to_use = 123
N = 1
J = 1
K = 1
set.seed(seed_to_use)
# number of levels for first covariate
N <- N
group_1 <- paste0("i", 1:N)
# number of levels for second covariate
J <- J
group_2 <- paste0("j", 1:J)
# number of latent dimensions
K <- K

# observed data ----
predictors <- expand.grid(group_1 = group_1, group_2 = group_2)
X_mat <- sparse.model.matrix(~ factor(group1) + factor(group_2) - 1, data = predictors)

# for sparsity, since here, we're assuming we have only dummies
# creating numeric values for each individual FE
predictors_as_numeric <- cbind(
  as.numeric(factor(predictors[, 1])), as.numeric(factor(predictors[, 2])))

# the regression part of the equation
betas <- matrix(rnorm(n = ncol(X_mat), 0, 2))
linear_predictor <- X_mat %*% betas
```

# Simulate a Factorization Machine:

Next, latent factors:

```
# group_1 factors are gammas
gammas <- mvrnorm(
  n = N,
  mu = rep(0, K),
  Sigma = diag(K))

# group 2 factors are deltas
deltas <- mvrnorm(
  n = J,
  mu = rep(0, K),
  Sigma = diag(K))

factor_terms <-
  matrix(NA, nrow = nrow(linear_predictor), ncol = 1)

for (i in 1:nrow(predictors)) {
  g1 <- as.character(predictors[i, 1])
  g1 <- as.numeric(substr(g1, 2, nchar(g1)))

  g2 <- as.character(predictors[i, 2])
  g2 <- as.numeric(substr(g2, 2, nchar(g2)))

  factor_terms[i,] <- matrix(gammas[g1,], nrow = 1) %*%
    matrix(deltas[g2,], ncol = 1)
}

y <- linear_predictor + factor_terms + rnorm(n = nrow(linear_predictor), 0, y_sigma)
```

# Factorization Machines in Stan

Use Stan to fit FMs:

- ▶ Data:

```
data{  
    int<lower = 0> N ; // number of group 1 observations  
    int<lower = 0> J ; // number of group 2 observations  
    int<lower = 0> K ; // number of latent dimensions  
    int X[(N*J), 2] ; // covariate matrix  
    vector[(N*J)] y ; // outcome  
    real<lower = 0> beta_sigma ; // sd on regression coefficients  
    real<lower = 0> y_sigma ; // sd on the outcome, y  
}
```

# Factorization Machines in Stan

Use Stan to fit FMs:

- ▶ Parameters:

```
parameters{
  vector[N] group_1_betas; // non-interacted coefficients
  vector[J] group_2_betas; // non-interacted coefficients
  matrix[N, K] gammas; // individual factors
  matrix[J, K] deltas; // group 2 factors
}

transformed parameters{
  real linear_predictor[(N*J)] ;
  for(i in 1:(N*J)){
    linear_predictor[i] =
      group_1_betas[X[i, 1]] + group_2_betas[X[i, 2]] +
      (gammas[X[i, 1], ] * deltas[ X[i, 2], ]');
  }
}
```

# Factorization Machines in Stan

Use Stan to fit FMs:

► Model:

```
model{

    // regression coefficients
    group_1_betas ~ normal(0, beta_sigma) ;
    group_2_betas ~ normal(0, beta_sigma) ;

    // latent factors
    for(n in 1:N){
        gammas[n, ] ~ normal(rep_vector(0, K), 1) ;
    }

    for(j in 1:J){
        deltas[j, ] ~ normal(rep_vector(0, K), 1) ;
    }

    // outcome
    y ~ normal(linear_predictor, y_sigma) ;
}
```

# Factorization Machines in Stan

Use Stan to fit FMs:

- ▶ Model Checking

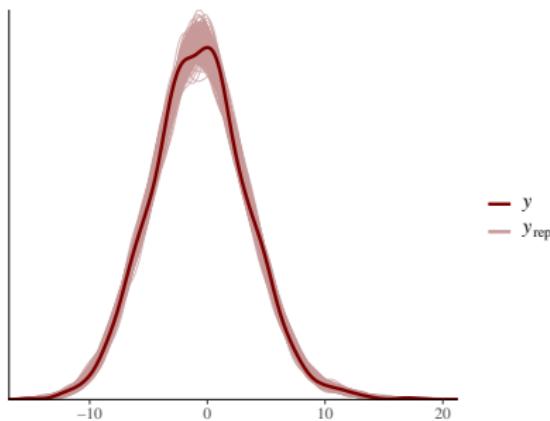
```
generated quantities{  
    real y_pred[(N*J)] ;  
    y_pred = normal_rng(linear_predictor, y_sigma) ;  
}
```

## Fitting a Model to Simulated Data

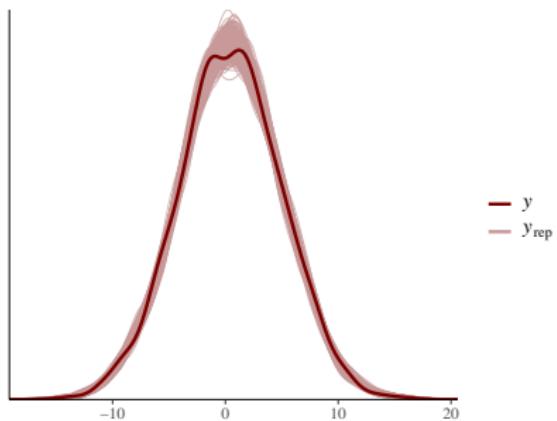
- ▶ Using `simulate_data` in `simulation_function.R`.
- ▶ Two versions:
- ▶ 100 members of group 1, 20 members of group 2
- ▶ 20 members of group 1, 100 members of group 2
- ▶ Fit model in `stan`, using NUTS

# Fitting a Model to Simulated Data

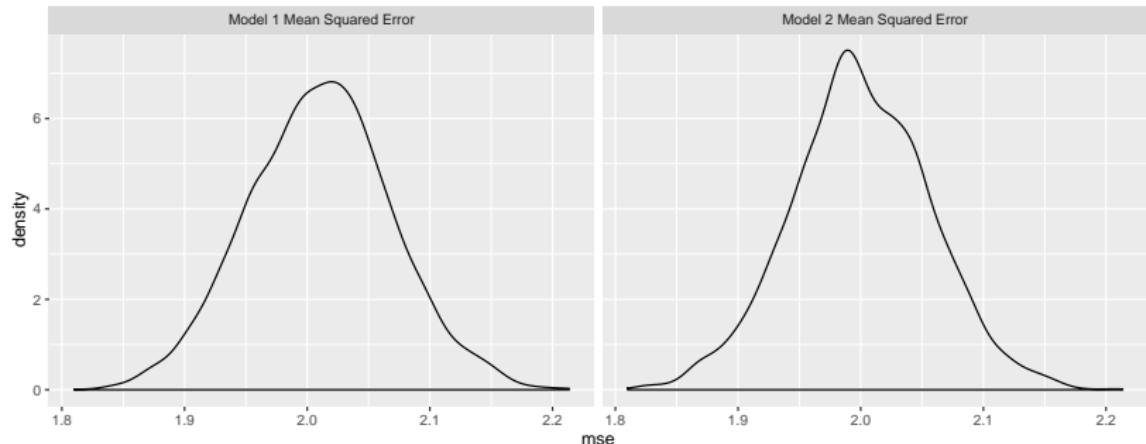
Model 1 Posterior Predictive



Model 2 Posterior Predictive



# Fitting a Model to Simulated Data



Now, Let's Get Hierarchical!

## Hierarchical Factorization Machines

- ▶ Basic FM implementation assumes all parameters are independent
- ▶ Hierarchical: Sharing parameters across groups
- ▶ Sharing parameters  $\$ = \$$  sharing information!

# Hierarchical Factorization Machines

$$y_{nj} \sim \mathcal{N}(\mathbf{x}_n \beta_1 + \mathbf{x}_j \beta_2 + \boldsymbol{\gamma}_n \cdot \boldsymbol{\delta}_j^\top, \sigma_y)$$

$$\boldsymbol{\beta} \sim \mathcal{N}(0, \sigma_\beta)$$

$$\boldsymbol{\gamma}_n \sim \mathcal{N}(\boldsymbol{\mu}_\gamma, \boldsymbol{\Sigma}_\gamma)$$

$$\boldsymbol{\delta}_j \sim \mathcal{N}(\boldsymbol{\mu}_\delta, \boldsymbol{\Sigma}_\delta)$$

$$\boldsymbol{\mu}_\gamma \sim \mathcal{N}(0, 1)$$

$$\boldsymbol{\mu}_\delta \sim \mathcal{N}(0, 1)$$

$$\boldsymbol{\Sigma}_\gamma = \sigma_\gamma \boldsymbol{\Omega}_\gamma$$

$$\boldsymbol{\Sigma}_\delta = \sigma_\delta \boldsymbol{\Omega}_\delta$$

$$\boldsymbol{\Omega}_\gamma \sim \mathcal{LKJ}(\Omega_0)$$

$$\boldsymbol{\Omega}_\delta \sim \mathcal{LKJ}(\Omega_0)$$

$$\sigma_\gamma \sim \text{Cauchy}(0, \sigma_0)$$

$$\sigma_\delta \sim \text{Cauchy}(0, \sigma_0)$$

# Hierarchical Factorization Machines in Stan

## ► Data

```
data{  
    int<lower = 0> N ; // number of group 1 observations  
    int<lower = 0> J ; // number of group 2 observations  
    int<lower = 0> K ; // number of latent dimensions  
    int X[(N*J), 2] ; // covariate matrix  
    vector[(N*J)] y ; // outcome  
    real<lower = 0> beta_sigma ; // sd on regression coefficients  
    real<lower = 0> y_sigma ; // sd on the outcome, y  
    real<lower = 0> gamma_sigma_prior ; //sd on gamma factors  
    real<lower = 0> delta_sigma_prior ; //sd on gamma factors  
}
```

# Hierarchical Factorization Machines in Stan

## ► Parameters

```
parameters{  
    vector[N] group_1_betas; // non-interacted coefficients  
    vector[J] group_2_betas; // non-interacted coefficients  
  
    matrix[N, K] gamma_mu ; //gamma prior mean  
    vector<lower=0, upper=pi()/2>[K] gamma_sigma_unif ; // for reparameterized Cauchy  
    cholesky_factor_corr[K] gamma_omega; // correlation matrix  
    matrix[K, N] gamma_a ; // for non-centered parameterization  
  
    matrix[J , K] delta_mu ; //delta prior mean  
    vector<lower=0, upper=pi()/2>[K] delta_sigma_unif ; // for reparameterized Cauchy  
    cholesky_factor_corr[K] delta_omega; // correlation matrix  
    matrix[K, J] delta_a ; // for non-centered parameterization  
}
```

# Hierarchical Factorization Machines in Stan

## ► Transformed Parameters

```
transformed parameters{
  real linear_predictor[(N*J)] ;
  vector<lower=0>[K] gamma_sigma ;
  matrix[N, K] gammas ;
  vector<lower=0>[K] delta_sigma;
  matrix[J, K] deltas ;

  // reparameterized cauchy
  for (k in 1:K) {
    gamma_sigma[k] = gamma_sigma_prior .* tan(gamma_sigma_unif[k]) ;
  }
  for (k in 1:K) {
    delta_sigma[k] = delta_sigma_prior .* tan(delta_sigma_unif[k]) ;
  }

  // non-centered parameterization
  gammas = gamma_mu + (diag_pre_multiply(gamma_sigma, gamma_omega) * gamma_a)' ;
  deltas = delta_mu + (diag_pre_multiply(delta_sigma, delta_omega) * delta_a)' ;

  for(i in 1:(N*J)){
    linear_predictor[i] =
      group_1_betas[X[i, 1]] + group_2_betas[X[i, 2]] +
      (gammas[X[i, 1], ] * deltas[X[i, 2], ])';
  }
}
```

# Hierarchical Factorization Machines in Stan

## ► Model

```
model{

    // regression coefficients
    group_1_betas ~ normal(0, beta_sigma) ;
    group_2_betas ~ normal(0, beta_sigma) ;

    // correlation matrices
    gamma_omega ~ lkj_corr_cholesky(5) ;
    delta_omega ~ lkj_corr_cholesky(5) ;

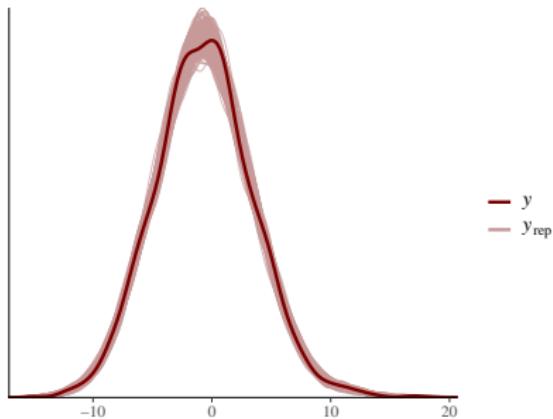
    // for non-centered parameterization
    to_vector(gamma_a) ~ std_normal() ;
    to_vector(delta_a) ~ std_normal() ;

    // hierarchical means
    for(n in 1:N){
        gamma_mu[n, ] ~ normal(0, 1) ;
    }
    for(j in 1:J){
        delta_mu[j, ] ~ normal(0, 1) ;
    }

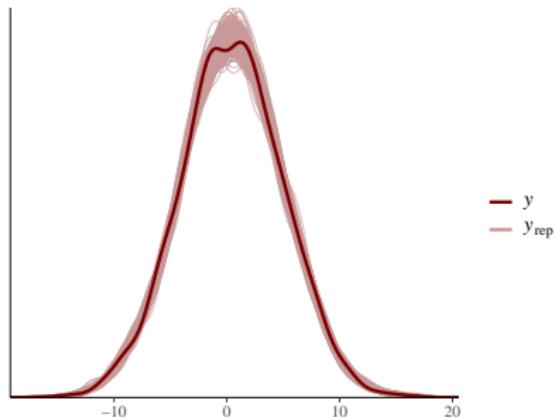
    // outcome
    y ~ normal(linear_predictor, y_sigma) ;
}
```

# Fitting a (Hierarchical) Model to Simulated Data

Hierarchical Model 1 Posterior Predictive



Hierarchical Model 2 Posterior Predictive



# Fitting a (Hierarchical) Model to Simulated Data

