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 (at The Ohio State University)
- Word Embeddings, Causal Inference, Bayesian Statistics, Discrete Choice Models

Why We Model

- Descriptive Inference: is there an association between $\boldsymbol{X}$ and $\boldsymbol{y}$?
- Prediction: with $\boldsymbol{X}$ and $\boldsymbol{y}$, what is $\boldsymbol{y}$ for a new $\boldsymbol{X}$?
- Causal Inference: does changing $\boldsymbol{X}$, change $\boldsymbol{y}$?
- 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 $\boldsymbol{X}_{N \times J}$ is decomposed into two low rank matrices: $\boldsymbol{\gamma}_{N \times K}$ and $\boldsymbol{\delta}_{K \times J}$
- Various assumptions about the structure of $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$

Factorization Machines

- Combine regression with a latent variable model on the residuals

But Why?

Factorization Machines

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

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

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

- With Interactions

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

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

Factorization Machines

Problems:

- We can only estimate β_3 for observed interactions
- As N and J grow, β_3 increases with size $N * J$

Factorization Machines

Solution!

- Replace β_3 with the dot product of low-rank latent factors:
- $\boldsymbol{\gamma}_{N \times K}$
- $\boldsymbol{\delta}_{J \times K}$
- β_3 is now $\gamma_n \cdot \delta_j^\top$

Factorization Machines

- Interaction model:

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

- Depending on our assumptions about $\gamma_n \cdot \delta_j^\top$, we can now create FMs

Factorization Machines

Basic model - Each element of δ_j and γ_n is distributed standard normal: $\mathcal{N}(0,1)$ - Automatic Relevance Determination (ARD) prior to shrink the matrix rank

$$
\begin{aligned}
y_{nj} &\sim \mathcal{N}(\boldsymbol{x}_n\beta_1 + \boldsymbol{x}_j\beta_2 + \gamma_n \cdot \delta_j^\top, 1) \\
\boldsymbol{\beta} &\sim \mathcal{N}(0, \sigma^2) \\
\gamma_{n,k} &\sim \mathcal{N}(0,1) \\
\delta_{j,k} &\sim \mathcal{N}(0,1)
\end{aligned}
$$

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:

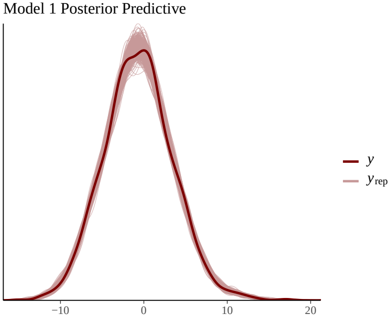
- Model Checking

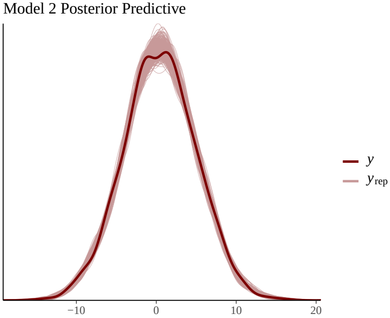
```
generated quantities{
  real y_pred[(N*J)]  ;
  y_pred = normal_rng(linear_predictor, y_sigma) ;
}
```

Fitting a Model to Simulated Data

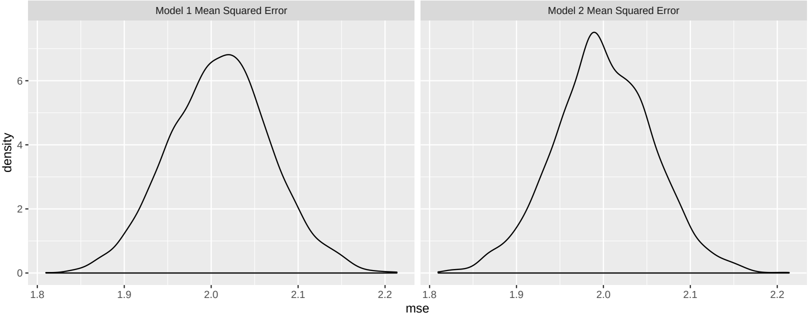
- Using `simulate_data` in `simulation_function.R`.
- Two versions:
- 100 members of group 1, 20 members of group 2
- 20 members of group 1, 100 members of group 2
- Fit model in `stan`, using `NUTS`

Fitting a Model to Simulated Data





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 infomation!

Hierarchical Factorization Machines

$$
\begin{aligned}
y_{nj} &\sim \mathcal{N}(\boldsymbol{x}_n\beta_1 + \boldsymbol{x}_j\beta_2 + \gamma_n \cdot \delta_j^\top, \sigma_y) \\
\boldsymbol{\beta} &\sim \mathcal{N}(0, \sigma_\beta) \\
\gamma_{n,} &\sim \mathcal{N}(\boldsymbol{\mu_\gamma}, \boldsymbol{\Sigma_\gamma}) \\
\delta_{j,} &\sim \mathcal{N}(\boldsymbol{\mu_\delta}, \boldsymbol{\Sigma_\delta}) \\
\mu_{\boldsymbol{\gamma}} &\sim \mathcal{N}(0, 1) \\
\mu_{\boldsymbol{\delta}} &\sim \mathcal{N}(0, 1) \\
\boldsymbol{\Sigma_\gamma} &= \boldsymbol{\sigma_\gamma \Omega_\gamma} \\
\boldsymbol{\Sigma_\delta} &= \boldsymbol{\sigma_\delta \Omega_\delta}
\end{aligned}
$$

Hierarchical Factorization Machines in Stan

```
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{
  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
  matrix[N, K] gamma_mu ; //gamma prior mean
  vector<lower=0, upper=pi()/2>[K] gamma_sigma_unif ; // 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 ; // reparameterized Cauchy
  cholesky_factor_corr[K] delta_omega; // correlation matrix
  matrix[K, J] delta_a ; // for non-centered parameterization
}

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