Instacart Market Basket Analysis

My solution for the Instacart Market Basket Analysis competition hosted on Kaggle.

The Task

The dataset is an open-source dataset provided by Instacart (source)

> This anonymized dataset contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, we provide between 4 and 100 of their orders, with the sequence of products purchased in each order. We also provide the week and hour of day the order was placed, and a relative measure of time between orders.

Below is the full data schema (source)

> `orders` (3.4m rows, 206k users):
>
> - `order_id` : order identifier
> - `user_id` : customer identifier
> - `eval_set` : which evaluation set this order belongs in (see `SET` described below)
> - `order_number` : the order sequence number for this user (1 = first, n = nth)
> - `order_dow` : the day of the week the order was placed on
> - `order_hour_of_day` : the hour of the day the order was placed on
> - `days_since_prior` : days since the last order, capped at 30 (with NAs for `order_number` = 1)
>
> `products` (50k rows):
>
> - `product_id` : product identifier
> - `product_name` : name of the product
> - `aisle_id` : foreign key
> - `department_id` : foreign key
>
> `aisles` (134 rows):
>
> - `aisle_id` : aisle identifier
> - `aisle` : the name of the aisle
>
> `deptartments` (21 rows):
>
> - `department_id` : department identifier
> - `department` : the name of the department
>
> `order_products__SET` (30m+ rows):
>
> - `order_id` : foreign key
> - `product_id` : foreign key
> - `add_to_cart_order` : order in which each product was added to cart
> - `reordered` : 1 if this product has been ordered by this user in the past, 0 otherwise
>
> where `SET` is one of the four following evaluation sets (`eval_set` in `orders`):
>
> - `"prior"` : orders prior to that users most recent order (~3.2m orders)
> - `"train"` : training data supplied to participants (~131k orders)
> - `"test"` : test data reserved for machine learning competitions (~75k orders)

The task is to predict which products a user will reorder in their next order. The evaluation metric is the F1-score between the set of predicted products and the set of true products.

The Approach

The task was reformulated as a binary prediction task: Given a user, a product, and the user's prior purchase history, predict whether or not the given product will be reordered in the user's next order. In short, the approach was to fit a variety of generative models to the prior data and use the internal representations from these models as features to second-level models.

First-level models

The first-level models vary in their inputs, architectures, and objectives, resulting in a diverse set of representations.

- **Product RNN/CNN** (code): a combined RNN and CNN trained to predict the probability that a user will order a product at each timestep. The RNN is a single-layer LSTM and the CNN is a 6-layer causal CNN with dilated convolutions.
- **Aisle RNN** (code): an RNN similar to the first model, but trained at the aisle level (predict whether a user purchases any products from a given aisle at each timestep).
- **Department RNN** (code): an RNN trained at the department level.
- **Product RNN mixture model** (code): an RNN similar to the first model, but instead trained to maximize the likelihood of a bernoulli mixture model.
- **Order size RNN** (code): an RNN trained to predict the next order size, minimizing RMSE.
- **Order size RNN mixture model** (code): an RNN trained to predict the next order size, maximizing the likelihood of a gaussian mixture model.
- **Skip-Gram with Negative Sampling (SGNS)** (code): SGNS trained on sequences of ordered products.
- **Non-Negative Matrix Factorization (NNMF)** (code): NNMF trained on a matrix of user-product order counts.

Second-level models

The second-level models use the internal representations from the first-level models as features.

- **GBM** (code): a lightgbm model.
- **Feedforward NN** (code): a feedforward neural network.

The final reorder probabilities are a weighted average of the outputs from the second-level models. The final basket is chosen by using these probabilities and choosing the product subset with maximum expected F1-score.

Requirements

64 GB RAM and 12 GB GPU (recommended), Python 2.7

Python packages:

- lightgbm==2.0.4
- numpy==1.13.1
- pandas==0.19.2
- scikit-learn==0.18.1
- tensorflow==1.3.0