STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

VAE for financial applications

Examples of financial applications with VAE

- generating synthetic data for simulations / testing trading algorithms
- fraud / anomaly detection
- price movement classification

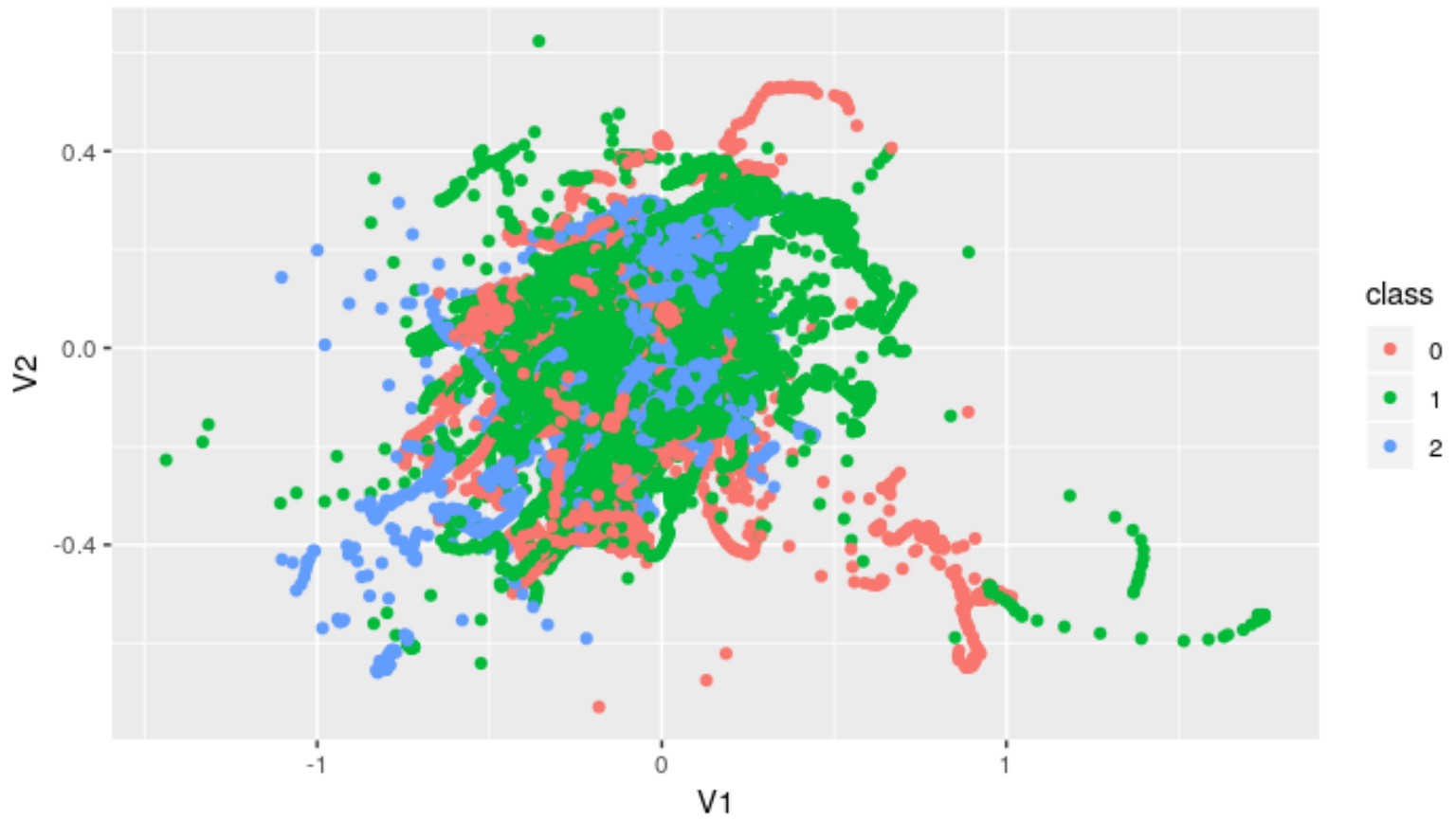
Sequence-to-sequence VAE

- useful for time series clustering / classification
- encoder and decoder are based on RNN
  - use an RNN to encode sequences into compressed latent space
  - choose a point from the latent space, and then **repeat** it for k times where k is the time step of the input sequence
    - same x from the latent space, but different states h's
  - use the k repeated points to reconstruct a sequence that is similar to the input sequences
- Example: VAE for limit order book data

- Try R

VAE for limit order book data

- Visualization of the latent space of a mildly attempted LOB example



- embedding input sequences into some latent distributions leads to self-learning classifications
  - further reading: Hennig et al. 2017.

Manage model runs with `tfruns`

- use R package `tfruns` to manage runs to tune parameters

```
library(tfruns)
training_run("program.R") # run whatever in `program.R`
tuning_run() # Run all combinations of the specified training flags

latest_run() # view the latest run result
ls_runs() # list all runs saved to local folders
View(ls_runs()) # a rendered version of ls_runs()
compare_runs(ls_runs(!is.na(metric_val_loss))) # compare different runs

clean_runs() # archives runs
purge_runs() # permanently delete runs
```

Training models with `tfruns`

- use `flags()` to organize hyper-parameters

```
FLAGS <- flags(
  flag_integer("intermediate_dim", 12), # default values, can be passed through training_run()
  flag_numeric("epsilon_std", 1.0)
)

tmp <- FLAGS$intermediate_dim
```

- use `training_run()` to run a specific combination of flagged parameters

```
training_run("program.R", flags = list(intermediate_dim = 16, epsilon_std = 0.5))
```

Hyper-parameter tuning with `tfruns`

- use `tuning_run()` to run all combinations of flagged parameters
- if only run a random sample of the combinations: `sample = 0.3`

```
tuning_run("program.R", sample = NULL,
           flags = list(intermediate_dim = c(12, 16),
                        epsilon_std = c(1.0, 0.5))
          )
```

- some tips for using `tfruns`
  - `tfruns` does not save learned weights, and we need to use callbacks in the R scripts
  - when `layer_lambda()` is used, Keras does not save models but weights
- other choices for training / hyper-parameter tuning
  - Google CloudML: https://tensorflow.rstudio.com/tools/cloudml
- Try R
- Back to Course Scheduler