Lecture 29:
Multinomial Logit & Intro to Deep Learning

Big Data and Machine Learning for Applied Economics
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

November 23, 2021

Announcements

- Thursday turn in your predictions
- Friday PS5 and PS6 are due
- Next week presentations

Recap: Text as Data

- Topic Models
- PCA Theory
- Latent Dirichlet Allocation (LDA): Example
- Word Embedings

Agenda

The multinomial logit model: Intuition

$X_{n \times p}$

$p >>> n$

- We've seen different techniques to use text in a regression
- So far all have been linear models,
- But what happens when we have to predict multiple outcomes

$y = \beta x + \lambda |\beta|$

The multinomial logit model: Intuition

- The MNLM can be thought of as simultaneously fitting binary logits for all comparisons among the alternatives.
- For example,
  - We have a categorical variable with the outcomes for Democrat, for Independent, and for Republican.
  - Assume that there is one independent variable measuring income in $1,000s$.
  - We can examine the effect of income on party by fitting three binary logits,

The multinomial logit model: Intuition

- The MNLM can be thought of as simultaneously fitting binary logits for all comparisons among the alternatives.

$$
\begin{aligned}
\ln \frac{Pr(D|X)}{Pr(I/X)} &= \beta_{0,D|I} + \beta_{1,D|I} Income \\
\ln \frac{Pr(R|X)}{Pr(I/X)} &= \beta_{0,R|I} + \beta_{1,R|I} Income \\
\ln \frac{Pr(D|X)}{Pr(R/X)} &= \beta_{0,D|R} + \beta_{1,D|R} Income
\end{aligned} \tag{1}
$$

- where the subscripts to the $\beta's$ indicate which comparison is being made.

The multinomial logit model: Intuition

- These logits include redundant info

$$\ln \frac{Pr(D|X)}{Pr(I/X)} - \ln \frac{Pr(R|X)}{Pr(I/X)} = \ln \frac{Pr(D|X)}{Pr(R/X)} \tag{2}$$

- which implies

$$\beta_{0,D|I} - \beta_{0,R|I} = \beta_{0,D|R} \tag{3}$$

$$\beta_{1,D|I} - \beta_{1,R|I} = \beta_{1,D|R} \tag{4}$$

Logits

- In general, with J alternatives, only $J-1$ binary ligts need to be fit (minimal set)

The multinomial logit model: Intuition

	Binary			Multinomial Logit		
	(1)	(2)	(3)			
VARIABLES	dem_ind	rep_ind	dem_rep	Democrat	Independent	Republican
income	-0.00249	0.0157***	-0.0184***	-0.00272		0.0152***
	(0.00355)	(0.00374)	(0.00230)	(0.00372)		(0.00366)
Constant	1.605***	0.659***	0.953***	1.613***		0.678***
	(0.149)	(0.162)	(0.105)	(0.153)		(0.160)
Observations	844	689	1,231	1,382	1,382	1,382

- Fitting the MNLM by fitting a series of binary logits is not optimal
  - Binary logit is based on a different sample.
  - It ignores the restricctions that are implicit in the definition of the MNLM

The multinomial logit model: formal statement

- Formally

$$\exp \quad \ln \Omega_{m|b}(X) = \ln \frac{Pr(y = m|X)}{Pr(y = b|X)} = X\beta_{m|b} \;\; for\, m = 1, \ldots, J \tag{5}$$

- were b is the base outcome (reference category)
- These J equations can be solved to compute the probabilities for each outcome

$$Pr(y = m|X) = \frac{exp(X\beta_{m|b})}{\sum_{j=1}^{J} exp(X\beta_{j|b})} \qquad \frac{\exp(X\beta)}{1+\exp(X\beta)} \tag{6}$$

Multinomial Inverse Regression

- A task that comes up often in social science is understanding how text connects to a set of related covariates.
- For example, you might want to connect the `we8there` reviews simultaneously to all five aspect ratings, allowing you to determine which content is predictive of ratings on, say, atmosphere separate from food or service.
- For such tasks, we can turn to multinomial inverse regression (MNIR) to link the text with observable covariates through a multinomial distribution.

Multinomial Inverse Regression

- The "inverse" in MNIR comes from the fact that, while text regression usually fits a single document attribute as a function of word counts, we are inverting the process by regressing the counts on any number of document attributes.
- Given document attributes v_i (author characteristics, date, beliefs, sentiment, etc.), MNIR follows the familiar generalized linear model framework.
- Each document x_i is modeled as arising from a multinomial with a logit link onto a linear function of v_i

Multinomial Inverse Regression

- Each document x_i is modeled as arising from a multinomial with a logit link onto a linear function of v_i

$$x_i \sim MN(q_i, m_i) \tag{7}$$

- with

$$q_{ij} = \frac{exp(\alpha_j + v_i'\phi_j)}{\sum_{l=1}^{p} exp(\alpha_l + v_i'\phi_l})) \tag{8}$$

- Now, the number of outcome categories is the number of tokens in our text vocabulary. This can be viewed as a natural extension of topic modeling: we are keeping the multinomial model for token counts but replacing unknown topics with known attributes.

Multinomial Inverse Regression: Example

- We have 6,166 reviews, with an average length of 90 words per review, `we8there.com`.
- A useful feature of these reviews is that they contain both text and a multidimensional rating on overall experience, atmosphere, food, service, and value.
- For example, one user submitted a glowing review for Waffle House #1258 in Bossier City, Louisiana:

I normally would not revue a Waffle House but this one deserves it. The workers, Amanda, Amy, Cherry, James and J.D. were the most pleasant crew I have seen. While it was only lunch, B.L.T. and chili, it was great. The best thing was the 50′ s rock and roll music, not to loud not to soft. This is a rare exception to what you all think a Waffle House is. Keep up the good work.
Overall: 5, Atmosphere: 5, Food: 5, Service: 5, Value: 5.

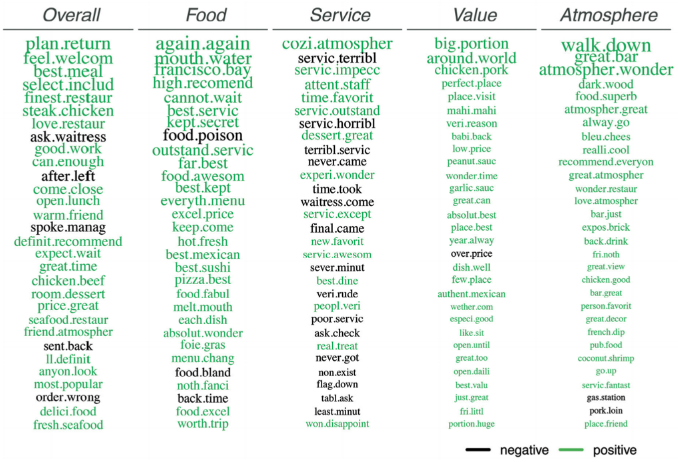
Multinomial Inverse Regression: Example

- Looking again at the `we8there` data, we can set v_i as the vector of five aspect ratings:
  1. overall
  2. atmosphere
  3. value
  4. food
  5. service

$X = ($overall, atm, value, food, serv$)$

- The multinomial response will be the vector of word counts for each review x_i, which implies 2640 outcome categories.

Multinomial Inverse Regression: Example



Multinomial Inverse Regression: Example

- This is truly a Big Data problem
- Traditional multiomial packages won't work
- In `R` you can use the package `distrom`
- It was designed to be efficient for these types of massive-response multinomials.
- It uses a Poisson distribution representation of the multinomial to distribute computation for each vocabulary element across multiple processors.

Deep Learning: Intro

Deep Learning: Intro

- Neural networks are simple models.
- Their strength lays in their simplicity because basic patterns facilitate fast training and computation.
- The model has linear combinations of inputs that are passed through nonlinear activation functions called nodes (or, in reference to the human brain, neurons).

Deep Learning: Intro

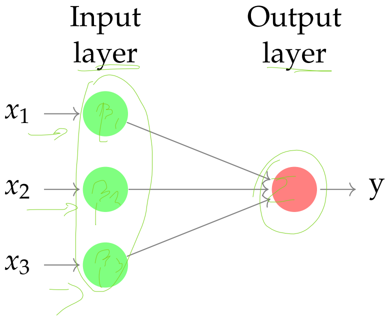
- Let's start with a familiar and simple model, the linear model

$$
\begin{aligned}
y &= f(X) + u \\
y &= \underbrace{\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3} + u
\end{aligned}
\tag{9}
$$

Deep Learning: Intro

- Let's start with a familiar and simple model, the linear model

$$
\begin{aligned}
y &= f(X) + u \\
y &= \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + u
\end{aligned}
\tag{9}
$$



Multilayer Perceptrons

$f \longrightarrow f^*$

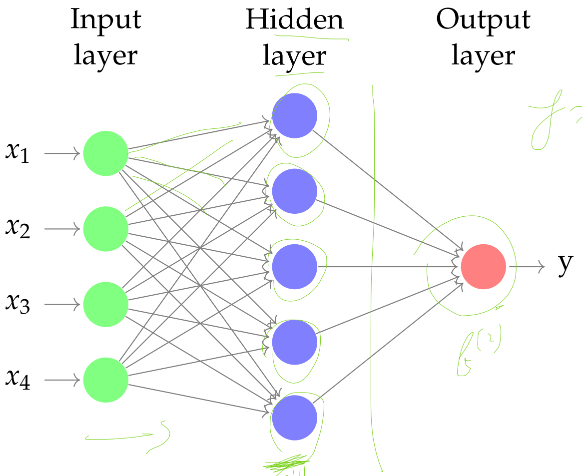
- Linear Models may be to simple, and miss the nonlinearities that best approximate $f^*(x)$
- We can overcome these limitations of linear models and handle a more general class of functions by incorporating one or more hidden layers.
- Deep feed forward networks, also called feed forward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models

Multilayer Perceptrons

- Feed forward neural networks are called networks because they are typically represented by composing together many different functions.
- For example, we might have two functions $f^{(2)}, f^{(1)}$ and connected in a chain to form $f(x) = f^{(2)}(f^{(1)}(x))$
- These chain structures are the most commonly used structures of neural networks.

Multilayer Perceptrons

ReLU



$f^{(2)}$

$y = f^{(2)}(f^{(1)}(x))$

Multilayer Perceptrons

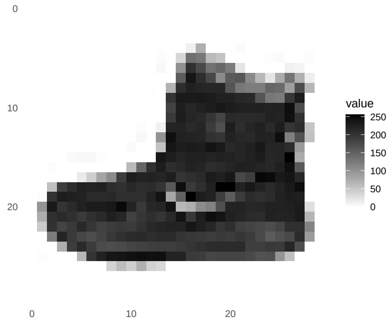
- The overall length of the chain gives the depth of the model. The name “deep learning” arose from this terminology.
- The final layer of a feed forward network is called the output layer
- During neural network training, we try to train $f(x)$ to match $f^*(x)$
- In the training data we observe the first layer, inputs (x), and the last layer, output (y)
- We do not observe the intermediate layers,they are then called hidden layers.
- Finally, these networks are called neural because they are loosely inspired by neuroscience.

Deep Learning: Demo

```r
library(keras)
fashion_mnist <- dataset_fashion_mnist()
```

10 tipos
?
4 p.

Deep Learning: Demo



Bota

28x28

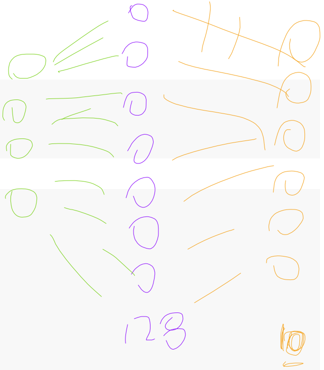
Deep Learning: Demo

```
train_images <- train_images / 255
test_images <- test_images / 255

model <- keras_model_sequential()
model %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)
model %>% fit(train_images, train_labels, epochs = 5, verbose = 2)
```

28x28

784

multinomial



Deep Learning: Demo

```
## Epoch 1/5
## 1875/1875 - 2s - loss: 0.5003 - accuracy: 0.8238
## Epoch 2/5
## 1875/1875 - 2s - loss: 0.3782 - accuracy: 0.8643
## Epoch 3/5
## 1875/1875 - 2s - loss: 0.3362 - accuracy: 0.8784
## Epoch 4/5
## 1875/1875 - 2s - loss: 0.3141 - accuracy: 0.8844
## Epoch 5/5
## 1875/1875 - 2s - loss: 0.2934 - accuracy: 0.8922
```

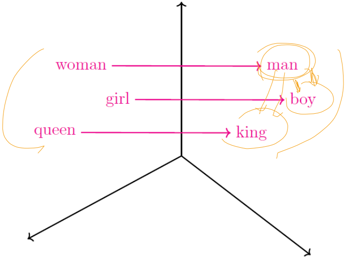
```
score <- model %>% evaluate(test_images, test_labels, verbose = 0)

cat('Test loss:', score[1], "\n")
```

```
## Test loss: 0.3377942

## Test accuracy: 0.8792
```

Word Embedding: Demo

Word Embedding: Demo

- In the original deep learning context, embedding layers replace each word with a vector value
  - for example, man becomes the location [1,2, 0.25] in a three-dimensional embedding space



[1, 2, 0,25]

[-0,5, 1, -0,5] =

[-0,5, 1, -0,25]

- Word embeddings preserve semantic relationships.
  - Words with similar meaning have similar representations.
  - Dimensions induced by word differences can be used to identify cultural concepts

Word Embedding: Demo

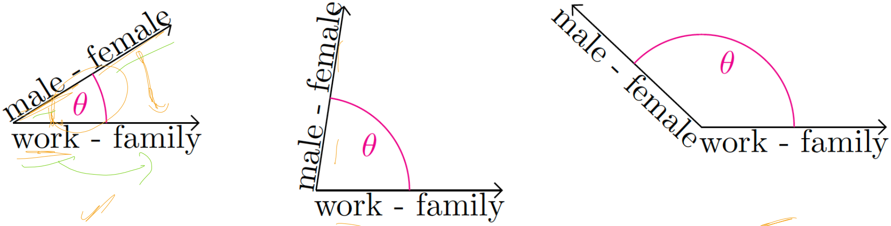
- The dimensions are useful because they produce quantitative measures of similarity between the associated concepts and specific words in the corpus.

Figure 2: Measuring Gender Stereotypes using Cosine Similarity

(a) Slant ≈ 1

(b) Slant ≈ 0

(c) Slant ≈ -1



Word Embedding: Demo

```
library(text2vec)
load('shakes_words_df_4text2vec.RData')
head(shakes_words)
```

```
##                  id        word
## 1 A_Lover_s_Complaint         nor
## 2 A_Lover_s_Complaint       gives
## 3 A_Lover_s_Complaint          it
## 4 A_Lover_s_Complaint satisfaction
## 5 A_Lover_s_Complaint          to
```

```
shakes_words_ls <- list(shakes_words$word)
it <- itoken(shakes_words_ls, progressbar = FALSE)
shakes_vocab <- create_vocabulary(it)
shakes_vocab <- prune_vocabulary(shakes_vocab, term_count_min= 5)
head(shakes_vocab)
```

```
## Number of docs: 1
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##          term term_count doc_count
## 1:     abbess          5         1
## 2:  abilities          5         1
## 3:  accessary          5         1
## 4:        ace          5         1
## 5:     adders          5         1
```

Word Embedding: Demo

- The next step is to create the token co-occurrence matrix (TCM).
- The definition of whether two words occur together is arbitrary.

```
# maps words to indices
vectorizer <- vocab_vectorizer(shakes_vocab)

# use window of 10 for context words
shakes_tcm <- create_tcm(it, vectorizer, skip_grams_window = 10)
```

- Now we are ready to create the word vectors based on the GloVe model.

```
glove <- GlobalVectors$new(rank = 50, x_max = 10)
shakes_wv_main = glove$fit_transform(shakes_tcm, n_iter = 10, convergence_tol = 0.01, n_threads = 8)
```

```
## INFO  [16:55:06.317] epoch 1, loss 0.1242
## INFO  [16:55:08.764] epoch 2, loss 0.0844
## INFO  [16:55:11.249] epoch 3, loss 0.0762
## INFO  [16:55:13.680] epoch 4, loss 0.0707
## INFO  [16:55:16.109] epoch 5, loss 0.0666
## INFO  [16:55:18.540] epoch 6, loss 0.0634
## INFO  [16:55:20.980] epoch 7, loss 0.0609
## INFO  [16:55:23.419] epoch 8, loss 0.0589
## INFO  [16:55:25.849] epoch 9, loss 0.0572
## INFO  [16:55:28.288] epoch 10, loss 0.0558
```

Word Embedding: Demo

```r
dim(shakes_wv_main)
```

```
## [1] 9094   50
```

```r
shakes_wv_context <- glove$components

dim(shakes_wv_context)
```

```
## [1]   50 9094
```

```r
# Either word-vectors matrices could work, but the developers of the technique
# suggest the sum/mean may work better
shakes_word_vectors <- shakes_wv_main + t(shakes_wv_context)

rom <- shakes_word_vectors["romeo", , drop = F]

cos_sim_rom <- sim2(x = shakes_word_vectors, y = rom, method = "cosine", norm = "l2")
# head(sort(cos_sim_rom[,1], decreasing <- T), 10)
```

```
##     romeo    juliet    tybalt     nurse  benvolio  banished
## 1.0000000 0.7712391 0.7575977 0.6697068 0.6517349 0.6436404
```

Word Embedding: Demo

men − women +
work ?

```
test <- shakes_word_vectors["romeo", , drop = F] -
  shakes_word_vectors["mercutio", , drop = F] +
  shakes_word_vectors["nurse", , drop = F]

cos_sim_test <- sim2(x = shakes_word_vectors, y = test, method = "cosine", norm = "l2")
head(sort(cos_sim_test[,1], decreasing = T), 10)
```

```
##     nurse    juliet     romeo      lady    mother       bed         o      wife
## 0.8904362 0.7584004 0.7179267 0.6440354 0.6374490 0.5880860 0.5756074 0.5638571
##   capulet    dromio
## 0.5520459 0.5507196
```

Review & Next Steps

- Multinomial Regression
- Deep Learning: Intro and Demo
- Next class: More on Neural Nets
- Please fill the perception survey https://encuestacursosuniandes.com/login

Further Readings

- Ash, E., Chen, D. L., & Ornaghi, A. (2020). Stereotypes in High-Stakes Decisions: Evidence from US Circuit Courts (No. 1256). University of Warwick, Department of Economics.
- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola (2020) Dive into Deep Learning. Release 0.15.1. http://d2l.ai/index.html
- Long, J. S., & Freese, J. (2014). Regression models for categorical dependent variables using Stata. Stata press. Rstudio (2020). Tutorial TensorFlow https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_classification/
- Taddy, M. (2019). Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions. McGraw Hill Professional.
- Voigt, R., Camp, N. P., Prabhakaran, V., Hamilton, W. L., Hetey, R. C., Griffiths, C. M., ... & Eberhardt, J. L. (2017). Language from police body camera footage shows racial disparities in officer respect. Proceedings of the National Academy of Sciences, 114(25), 6521-6526.