Lecture: Deep Learning in SAS Viya

Dr. Goutam Chakraborty

SAS® Professor of Marketing Analytics

Director of MS in Business Analytics and Data Science* (http://analytics.okstate.edu/mban/)
Director of Graduate Certificate in Business Data Mining (http://analytics.okstate.edu/certificate/grad-data-mining/)
Director of Graduate Certificate in Marketing Analytics (http://analytics.okstate.edu/certificate/grad-marketing-analytics/)

- *Name change pending internal approval.
- Note some of these slides are copyrighted by SAS® and used with permission. Reuse or redistribution is prohibited.
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

1

Outline

- CPU vs. GPU and why the latter is more appropriate for deep learning models
- Using SAS Studio to build DNN
  - Explain codes and options
- Important pointers:
  - Data for both demo and exercises will be loaded for you in SAS Viya so you can use it
    - You don't have to upload them again
  - Codes will be made variable so you can run them to replicate demos and then make appropriate modifications to do exercise
  - First log-in to SAS Viya: https://stwsasviya01.it.okstate.edu/SASDrive/
  - Then log in to SAS Studio in Viya: https://stwsasviya01.it.okstate.edu/SASStudio/

2

CPU Vs. GPU

Central Processing Unit (CPU)	Graphical Processing Unit (GPU)
Faster Clock Speed	**Slower** Clock Speed
Fewer Processing Units	**More** Processing Units
More Branching	**Less** Branching
Less Memory Bandwidth	**More** Memory Bandwidth

3

Deep Learning Actions

Deep Fully Connected Neural Network (DNN)

BuildModel / modeltable={name="< *Model table name* >"}
type = "**DNN**";

Convolutional Neural Network (CNN)

BuildModel / modeltable={name="< *Model table name* >"}
type = "**CNN**";

Recurrent Neural Network (RNN)

BuildModel / modeltable={name="< *Model table name* >"}
type = "**RNN**";

...

SAS Cloud Analytic Services Actions

```
PROC CAS;
    <CASL code>
QUIT;
```

- Example:

```
PROC CAS < exc > < noqueue >;
    BuildModel / modeltable={name="< Model table name >"}
                 type="DNN";
QUIT;
```

Building a Deep Neural Network

```
PROC CAS < exc > < noqueue >;
    BuildModel / modeltable={name="< model table name >"} type ="DNN";

AddLayer /
    modeltable="< model table name >"
    name="< name of layer >"
    layer={type="layer type"
           n="< number of hidden units >"
           act="< type of activation transformation >"
           init="< weight initialization method >"}
    srcLayers={"< previous layer name >"};
QUIT;
```

Example of Some Layers

Type	Details	Example
INPUT	Input Layer	layer={type='input'}
CONVO/CONVOLUTION	Convolutional Layer	layer={type='convolution' nFilters=32 width=5 height=5 stride=1 act='tanh' }
POOLING	Pooling Layer	layer={type='pooling' width=2 height=2 stride=2}
FC/FULLCONNECT	Fully connected Layer	layer={type='fullconnect' n=50 act='sigmoid' }
OUTPUT	Output Layer	layer={type='output' act='softmax' }
RESIDUAL	Residual Layer	layer={type='residual' }
BATCHNORM	Batch Normalization	layer={type='batchnorm' act='ELU'}
CONCAT	Concatination Layer	layer={type='concat'}
FCMP	FCMP Layer	layer={type='FCMP', forwardFunc='forward_prop', backwardFunc='back_prop', height=1, width=40, depth=1, nweights=1280}
RECURRENT	Recurrent Layer	layer={type='recurrent' n=50 act='sigmoid' rnnType='gru'}
TRANSCONVO	Transpose Convolution Layer	layer={type='transconvon' Filters=32 width=5 height=5 stride=2 act='tanh' }

7

Training a Deep Learning CAS Action Model

```
PROC CAS;
      dlTrain /
         table=' CAS-libref.data-table '
         modeltable= ' model name, specified in buildmodel action '
         bestweights={name= 'Name of output table containing best
                                    model weights '}
         inputs={' List of ALL input variables '}
         nominals={' List of nominal variables '}
         validtable={name= ' Name of validation data table '}
         target={'list of target variables '};
QUIT;
```

The parameters of the dlTrain action can be found at this link:
https://go.documentation.sas.com/?docsetId=casdlpg&docsetTarget=cas-deeplearn-dltrain.htm&docsetVersion=8.4&locale=en

Demo: Basics of Deep Learning
SAS Viya

Dr. Goutam Chakraborty

SAS® Professor of Marketing Analytics

Director of MS in Business Analytics and Data Science* (http://analytics.okstate.edu/mban/)
Director of Graduate Certificate in Business Data Mining (http://analytics.okstate.edu/certificate/grad-data-mining/)
Director of Graduate Certificate in Marketing Analytics (http://analytics.okstate.edu/certificate/grad-marketing-analytics/)

- *Name change pending internal approval.
- Note some of these slides are copyrighted by SAS® and used with permission. Reuse or redistribution is prohibited.
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

9

Outline

- Important pointers:
  - Data for both demo and exercises will be loaded for you in SAS Viya so you can use it
    - You don't have to upload them again
  - Codes will be made variable so you can run them to replicate demos and then make appropriate modifications to do exercise
  - First log-in to SAS Viya: https://stwsasviya01.it.okstate.edu/SASDrive/
  - Then log in to SAS Studio in Viya: https://stwsasviya01.it.okstate.edu/SASStudio/
- 3 different demos in SAS Viya

10

Demonstrations

- Data: Develop (split intro train, valid and test)
  - Data and variable descriptions on a separate handout
- Code files:
  - DLMS01D01a.sas
  - DLMS01D01b.sas
  - DLMS01D02a.sas
  - DLMS01D02b.sas
- Plan:
  - Run each code and explain results

11

Demo 1: Traditional NN with 7 Hidden Layers

- First open and run DLMS01D01a.sas (*modify local library details*, I used GC)
  - Promote 3 data sets to CAS (loading to memory)
- Second, open and run DLMS01D01b.sas (*no modifications needed*)
- Explore results

12

Demo2: Building and Training DL Network using CASL Code

- Open and run DLMS01D02a.sas (*no modifications needed*)
- Explore results

13

Demo3: Using batch Normalization, TanH instead of RELU, Xavier weight initialization

- Open and run DLMS01D02b.sas (*no modifications needed*)
- Explore results

14

Self Study Autoencoders

Dr. Goutam Chakraborty

SAS® Professor of Marketing Analytics

Director of MS in Business Analytics and Data Science* (http://analytics.okstate.edu/mban/)
Director of Graduate Certificate in Business Data Mining (http://analytics.okstate.edu/certificate/grad-data-mining/)
Director of Graduate Certificate in Marketing Analytics (http://analytics.okstate.edu/certificate/grad-marketing-analytics/)

- *Name change pending internal approval.
- Note some of these slides are copyrighted by SAS® and used with permission. Reuse or redistribution is prohibited.
- Some of the slides were developed by Mr. Sanjoy Dey, used with permission.

15

Outline

- Introduce autoencoders.
- Explain how to sparse and denoise autoencoders.

Building Autoencoders

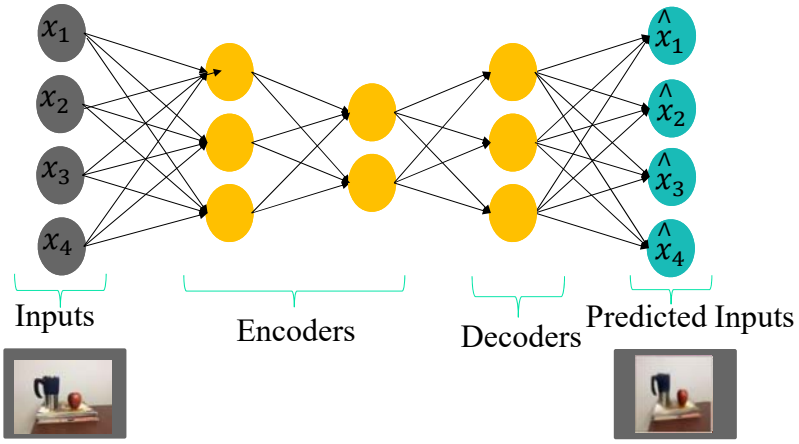
```
PROC CAS;
      dlTrain /
         table=' CAS-libref.data-table '
         modeltable= ' model name, specified in buildmodel action '
         bestweights={name= ' Name of output table containing best
                                    model weights' }
         inputs={' List of ALL input variables '}
         nominals={' List of nominal input variables '}
         validtable={name= ' Name of validation data table'}

RUN;
```

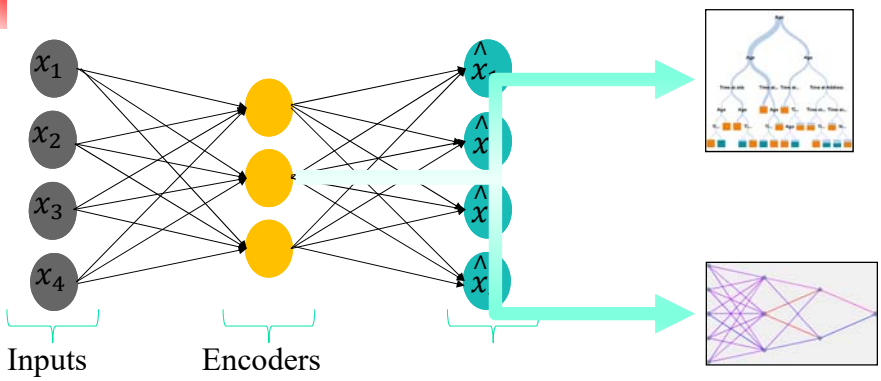
- Omit the TARGET parameter to train an autoencoder using dlTrain.

Autoencoders

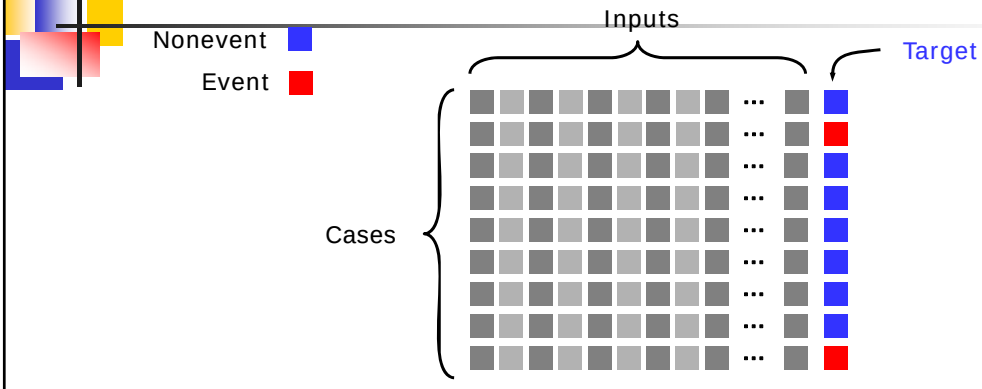
- Autoencoders use a feed-forward neural network.
- Unsupervised learning technique.
- Can be linear or nonlinear, but nonlinear are likely more useful.

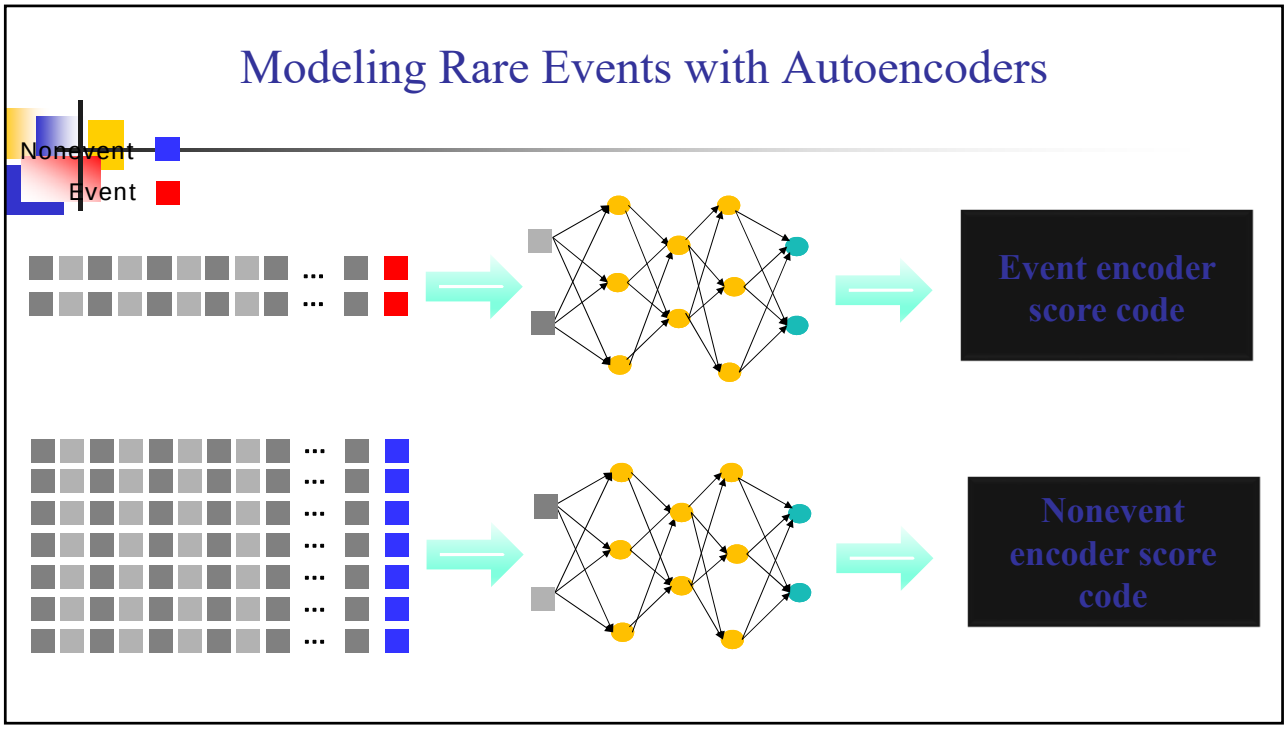


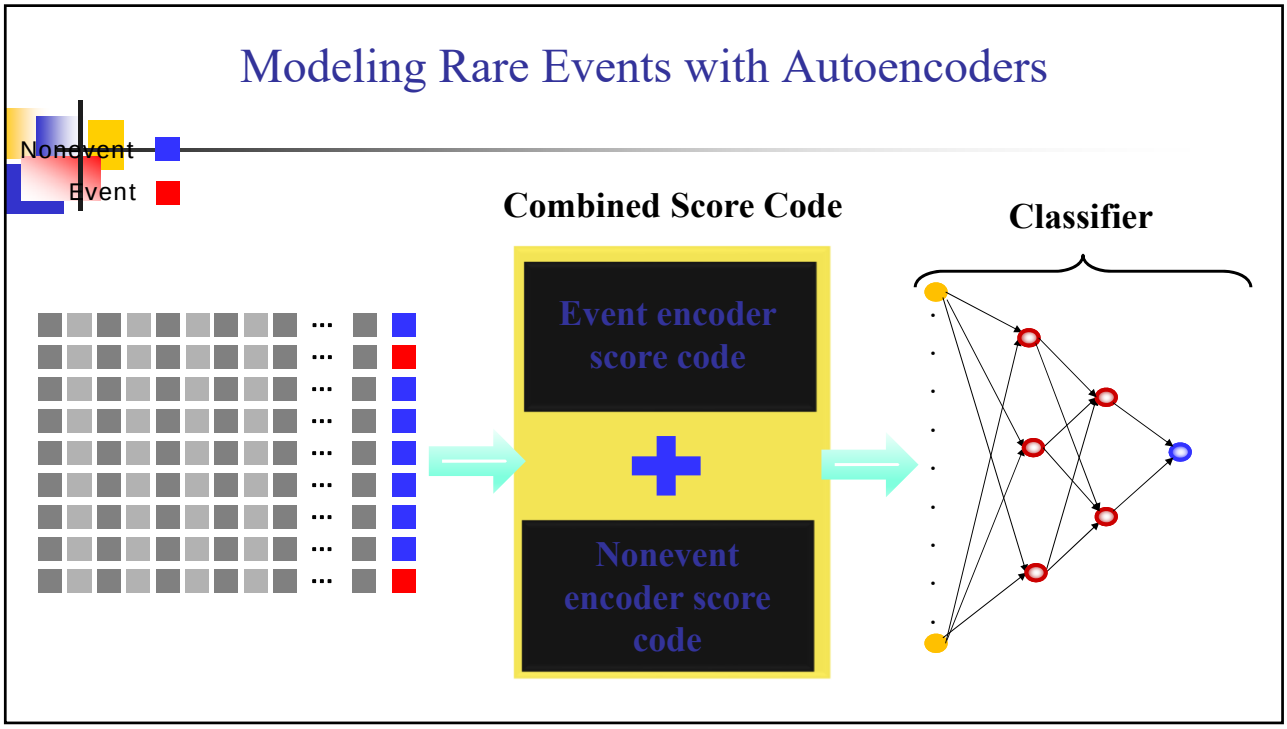
Modeling with Autoencoders



Modeling Rare Events with Autoencoders




Modeling Rare Events with Autoencoders
Nonevent
Event
Event encoder score code
Nonevent encoder score code


Modeling Rare Events with Autoencoders
Nonevent
Event
Combined Score Code
Classifier
Event encoder score code
Nonevent encoder score code