Machine Learning for Economists

Class 14: Artificial Neural Network Part II

葛雷

中国人民大学 - 数量经济

2025 年 5 月 14 日

RENMIN UNIVERSITY OF CHINA

Weights Initialization

Normalization

Regularization

AutoEncoder

Transfer Learning

Weights Initialization

Normalization

Regularization

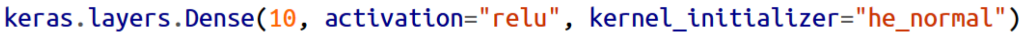
AutoEncoder

Transfer Learning

Weights Initialization

- We optimize the weights in the ANN by Gradient Descent
- How do we get the initial values of weights at t=0
- random draws from normal, truncated normal, or uniform distribution

Weights Initialization



```
keras.layers.Dense(10, activation="relu", kernel_initializer="he_normal")
```

- we have he_normal,glorot_normal,lecun_normal, he_uniform, glorot_uniform,lecun_uniform
- they are just uniform and normal distribution

note: He Kaiming contributes both to the He initialization and ResNet

Weights Initialization

- the choice of initialization should depend on your activation function
- sigmoid pairs well with uniform distribution. Why?
- ChatGPT using normal distribution pairs with non-saturated activation function GELU

Weights Initialization

Normalization

Regularization

AutoEncoder

Transfer Learning

Normalization

Normalization can help model:

- Improved Stability: reduces the risk of vanishing or exploding gradients
- Faster Training
- Regularization

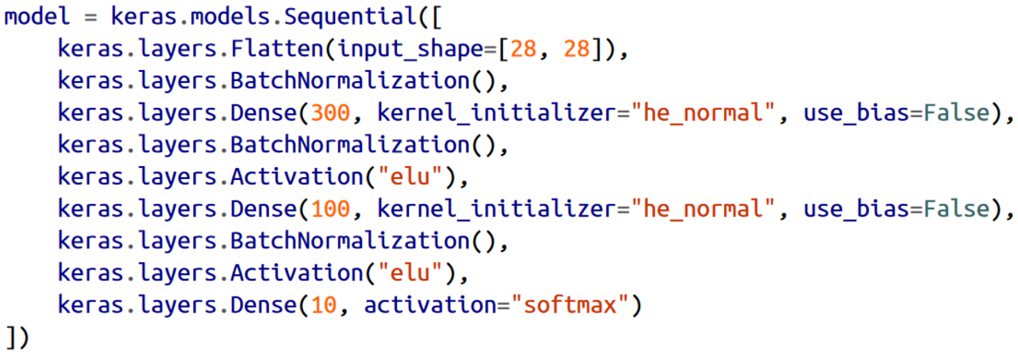
Input Data Normalization

- Input Data Normalization is same as requirement for Simple LASSO
- However, we should do for the data normalization in the deep layers, please turn to Batch Normalization and Layers Normalization

Batch Normalization

- Batch Normalization normalizes the inputs of each layer
- Actually, it is the normalization between layers
- For each batch

Batch Normalization

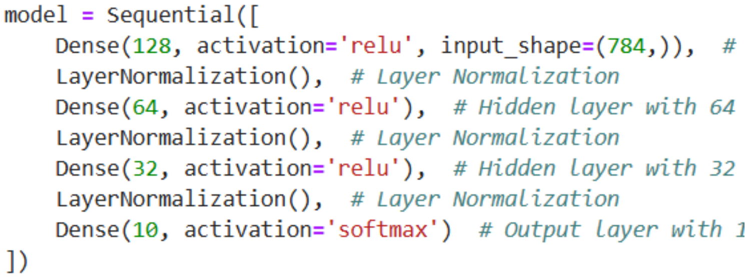


```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, kernel_initializer="he_normal", use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("elu"),
    keras.layers.Dense(100, kernel_initializer="he_normal", use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("elu"),
    keras.layers.Dense(10, activation="softmax")
])
```

Layer Normalization

- layer normalization normalizes across the features
- for each individual data sample

Layer Normalization



```
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),  #
    LayerNormalization(),  # Layer Normalization
    Dense(64, activation='relu'),  # Hidden layer with 64
    LayerNormalization(),  # Layer Normalization
    Dense(32, activation='relu'),  # Hidden layer with 32
    LayerNormalization(),  # Layer Normalization
    Dense(10, activation='softmax')  # Output layer with 1
])
```

Weights Initialization

Normalization

Regularization

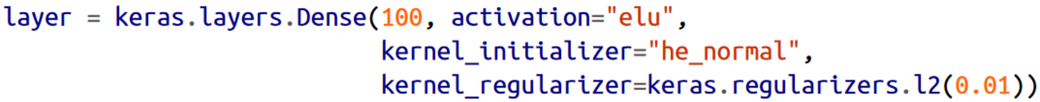
AutoEncoder

Transfer Learning

Regularization

- Regularization techniques help prevent the model from over fitting
- Make model more robust to new data (强所以可以抵御变化)

L1 and L2 Regularization



```
layer = keras.layers.Dense(100, activation="elu",
                           kernel_initializer="he_normal",
                           kernel_regularizer=keras.regularizers.l2(0.01))
```

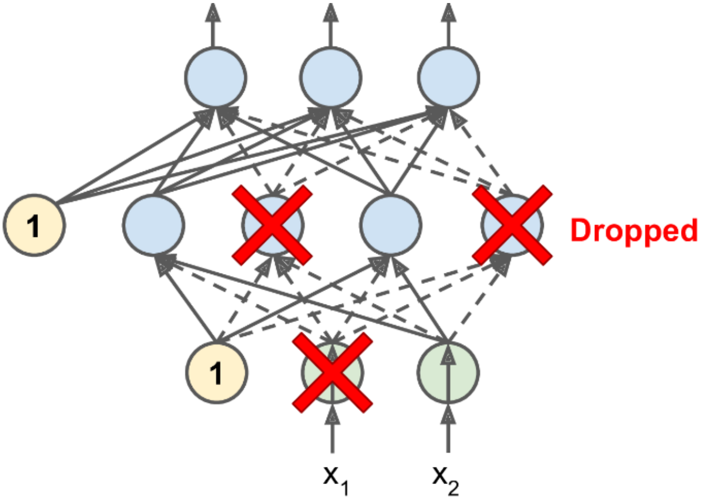
L1 and L2 Regularization

Question: where does the l1 l2 apply to the neural network?

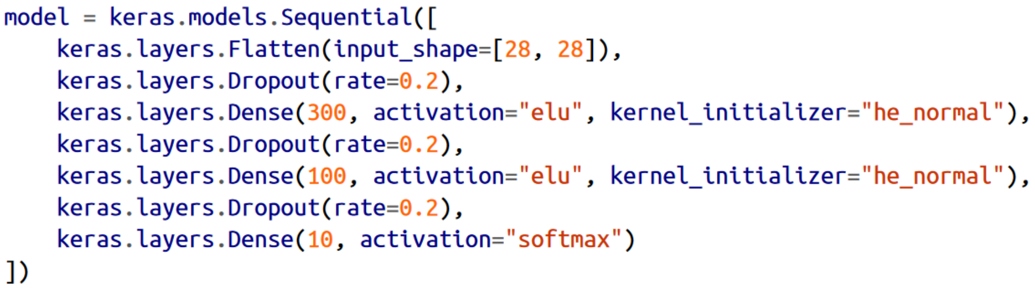
Dropout layer

- we randomly “dropped out” neurons during training (in each batch)
- these neurons (x) is replaced with 0 at this iteration
- only in the training session not in the prediction session

Dropout layer



Dropout layer

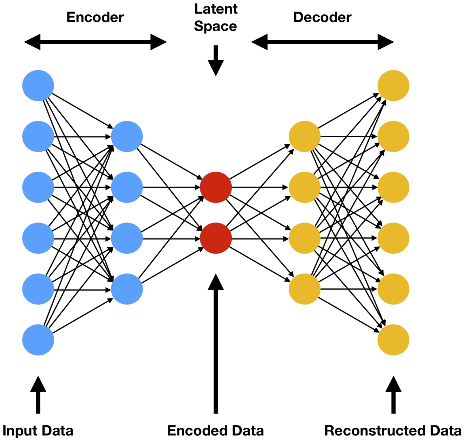


```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(300, activation="elu", kernel_initializer="he_normal"),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(100, activation="elu", kernel_initializer="he_normal"),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(10, activation="softmax")
])
```

AutoEncoder

1. Traditional PCA (linear dimension reduction by using linear function)
2. AutoEncoder (non-linear dimension reduction by using ANN)
3. Both PCA and AutoEncoder are popular unsupervised machine learning algorithms for the dimension reduction

AutoEncoder



AutoEncoder in Stock Quant

- Example in Stock Model: ROA, ROE, RPS are highly correlated. Leverage ratio, quick ratio, cash ratio are highly correlated.
- Question: why the autoencoder is popular in stock quant?

Weights Initialization

Normalization

Regularization

AutoEncoder

Transfer Learning

Transfer Learning

- I have a **small dataset**, but I want to use a large model
- Yes. you should turn to **transfer learning**
- Use your small data to adjust the pre-trained large model (large model is trained by large dataset)

Transfer Learning: Textbook Figure 11-4

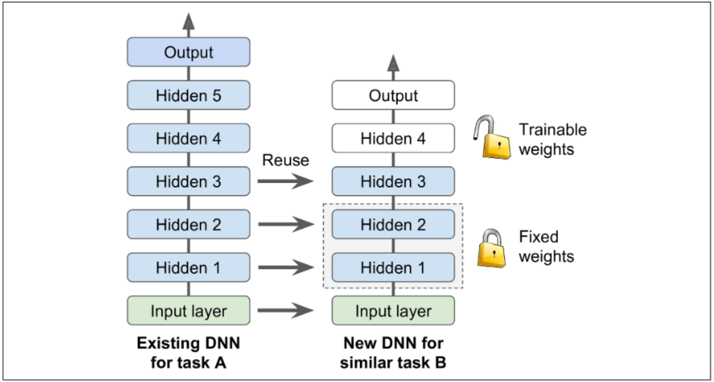


Figure 11-4. Reusing pretrained layers

Transfer Learning: Yolo Examples

- Large model: Yolo model (118,000 images; around 70 million parameters)
    - Transfer learning example 1: detecting diseases from X-rays, MRIs, or CT scans by using small sample medical images
    - Transfer learning example 2: detecting objects in self-driving by using small sample driving images

Transfer Learning: LLM Examples

- Large model: Llama (15.6 trillion tokens dataset), Deepseek, Bert, Roberta
    - Sentiment Analysis
    - Textual Classification
    - Important information extraction
    - Report generations

Reference

1. Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd edition)
2. Wikipedia
3. geeksforgeeks
4. Kaggle
5. Wikipedia
6. ChatGPT
7. DeepSeek