Lecture 22: Tensorflow Introduction

COMP 343, Spring 2022

Victoria Manfredi

WESLEYAN
UNIVERSITY

Outline

Homework 7 and Project proposal

- Due today!

Homework 5 and 6

- about 10 homeworks still to grade for each …

Tensorflow

- Installation
- Simple neural network
- Re-implement Titanic neural network
- Other things to explore

What is Tensorflow (and Keras)?

- A machine learning library primarily for (deep) neural networks

- Makes constructing and training neural networks very easy. Can easily vary architectures and methods for training

- Neural networks implemented can be anything simple to complex state-of-the art models

Tensorflow installation

- You can install tensorflow locally

  pip install tensor flow

  or

  pip3 install tensorflow

- Or you can use Google colab which is already installed and you just need to import tensor flow

  https://colab.research.google.com/

- For details see https://www.tensorflow.org/install

Simple neural network

- Read through and set up neural network here

  https://www.tensorflow.org/tutorials/quickstart/beginner

- Epoch
  - One complete pass through all examples
- Batch
  - Examples divided into batches, weights updated after processing each batch
  - Batch size is typically less than number of examples
- Iteration
  - Number of iterations that an epoch runs
  - Given n examples and batch size of b then number of iterations is n / b

Tensorflow good-to-know

- A neural network is built using a sequential list
  - Each layer of the network is represented by a Dense(x,a) layer where x is the number of nodes and a is the activation function.

- The model.compile() step is necessary to set what metrics/loss functions are used.
  - It generalizes to both classification and regression problems, you just need to pick different metrics and loss functions depending on the problem.

- A classification problem has x output nodes for x categories unless it is a binary classification, in which case just 1 output node and use the Binary_Crossentropy() loss function.

- A regression problem like the ones we have seen also only has one output node . Multi-output regression, however, is also possible

- You can try to prevent overfitting with a combination of L2 regularization and dropout. Dropout "zeroes out" a certain percentage of random output features during training and at test time the output features are scaled down by this percentage to prevent overfitting.

Titanic predictions again

- Will need to change neural network settings
  - Loss is now binary cross entropy
  - Hidden layer activation: tanh
  - Output layer activation: sigmoid activation
- Try adding dropout rate of 0.2

Other things to explore

- Regression: https://www.tensorflow.org/tutorials/keras/regression
- Overfitting vs. underfitting: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit
- What can you vary?
  - Regularization
  - Number of hidden nodes
  - Loss function
  - Activation function
  - Optimization algorithm
  - …