**Texas Tech University**

**Department of Computer Science**

**Course:** Introduction to Artificial Intelligence **Group:** 1

**Instructor:** Dr. Juan Carlos Rojas **Email:** [Juan-Carlos.Rojas@ttu.edu](mailto:Juan-Carlos.Rojas@ttu.edu)

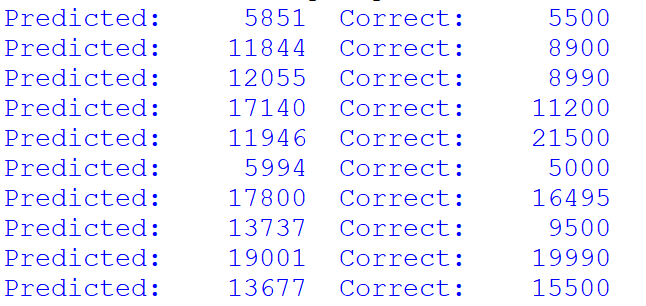
**Hours:** 8:00 – 12:00 (Saturdays) **Room:** 320

# Homework 5

Due Saturday, June 29 at 8:00am.

## Practice 1

* Use Tensorflow to solve the Normal equations of a linear regression model of vehicle prices
  + See VehiclePrice\_LinearRegression\_tf\_1.py
* Add TensorFlow instructions to compute the prediction values against the test dataset
  + Hints:
    - You need to declare a vector for the test data
    - Multiply it by W
* Print the first 10 prediction values
* Compare with the first 10 correct label values



## Practice 2

* Use Tensorflow to to solve a linear regression model for vehicle prices using gradient descent
  + You can base it on VehiclePrice\_LinearRegression\_tf\_4.py
* Add Tensorflow instructions to compute the RMSE of the test predictions against the test labels
* Hints:
  + Compute the predictions against the test data as Y\_pred\_test = X\_test \* W + b
  + Compute the test MSE as test\_mse = tf.reduce\_mean(tf.square(Y\_pred\_test - Y\_true\_test))
  + At the end, run a TensorFlow session to compute the value of test\_mse



## Optional Problem 2b

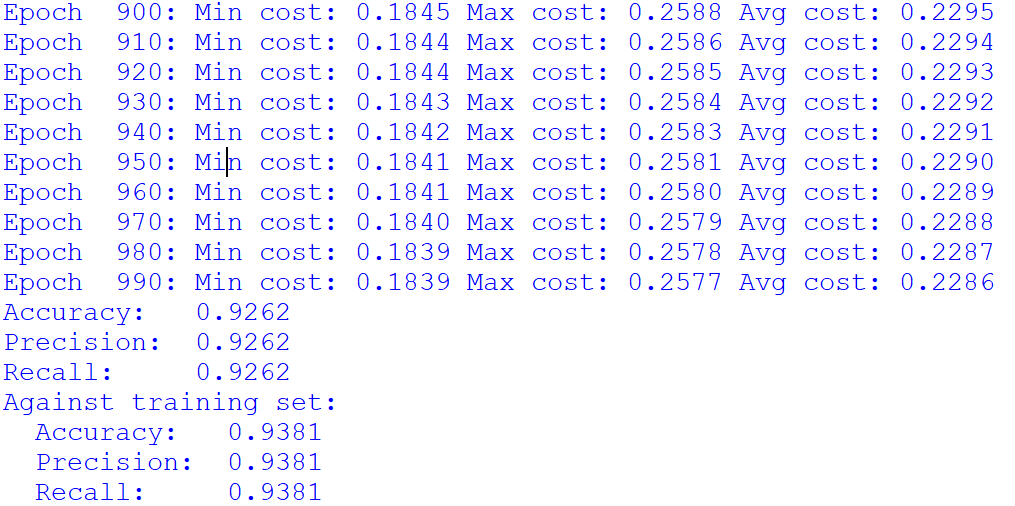
* Adjust the learning rate in the gradient descent solution of the vehicle price linear regression
  + What happens if the learning rate is very large?
  + What happens if it is very small?

## Optional Problem 2c

* Replicate the gradient descent solution for softmax regression of MNIST images
  + Use Mnist\_SoftmaxRegression\_tf\_2.py as a base
* Adjust the learning rate to make it converge faster
  + Can you get to the same result?

## Practice 3

* Consider the mini-batch gradient descent solution of softmax regression of MNIST images
  + From Mnist\_SoftmaxRegression\_tf\_3.py
* In the update loop when the epoch matches the print step, compute and print the average cost for all the mini-batches in this epoch
  + Also compute the minimum and maximum cost for any mini-batch in this epoch
* Hints:
  + Store the cost values in a list of length n\_batches
  + Move the if (epoch % print\_step == 0) outside the batch loop



## Practice 4

* Consider the 2-layer neural network for MNIST classification
  + From Mnist\_NN\_1.py
* Modify the number of nodes in the hidden layer
  + Try values like 10, 50, 100, 500, 1000, etc.
* For each case, let it train and measure the accuracy against the test set, and also against the training set
  + Comment on your results

## Optional Problem 4b

* Repeat the Practice 4 but using different activation functions for the hidden layer
  + You can try ReLU & ELU
* Are your results different?
* Is it any faster/slower?

## Practice 5

* Create a 2-layer neural network for the classification of credit defaults
* Suggestions:
  + For the cost function, use sigmoid\_cross\_entropy\_with\_logits
  + Compute the prediction probabilities with tf.nn.sigmoid(Y\_L2\_linear)
  + Suggested training parameters:
    - batch\_size = 60
    - learning\_rate = .02
    - n\_epochs = 500
    - n\_nodes\_l1 = 10
    - Activation function: ReLU
* Compute the AUC score against the test data set and against the training data set

