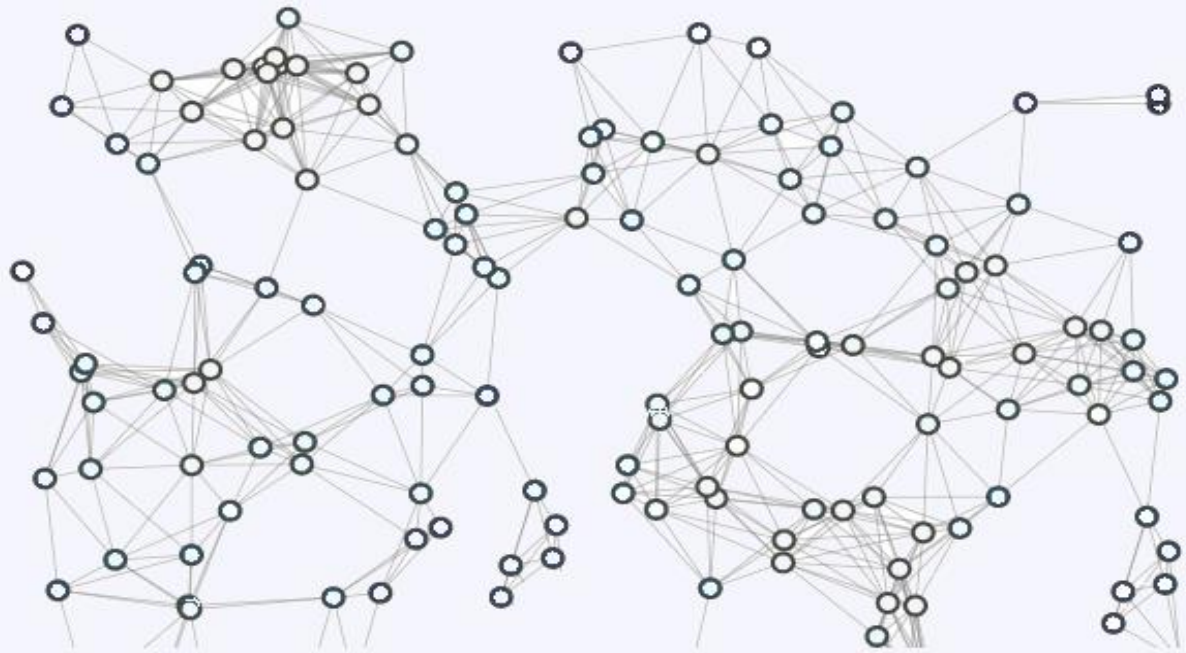
Model Training



Rodolfo C. Raga Jr.

September 23-24, 2019

San Beda College

jundy.raga@gmail.com

Topic Outline

Day 1	Day 2
1) Intro to Machine Learning and Tensorflow 2) Data Preprocessing 3) Learning Models (Regression) 3.1. NN Simple Linear 3.2. NN Multiple Linear **4) Model Training**	1) Learning Models (Classification) 1.1. NN Logistic Regression 1.2. CNN Deep Learning 2) Model Testing 3) Model Evaluation / Validation Performance 4) Data Visualization

Model Training: Issues and Problems

- You have already built your prediction model and its already producing initial prediction values using the training data.
- However, it still fails to produce good performance when you run it on the validation dataset

Model Training: Selection and Assessment

Model Selection: Estimating performances of different models to choose the best one (produces the minimum of the ***test error***) (Using validation dataset)

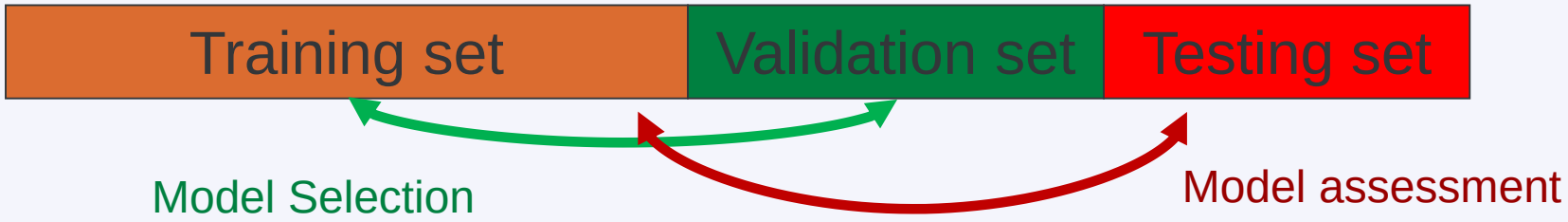
Model Assessment: Having chosen a model, estimating the ***prediction error*** on new data (Using testing dataset)

There are three datasets to be familiar with

- **Training set:** a set of examples used to fit the hyperparameters of the Neural Network (learning), we would use the training set to find the “optimal” weights .

- **Validation set**: a set of examples used to tune the parameters of a classifier, e.g., to find the “optimal” number of hidden units or determine a stopping point for the back-propagation algorithm

- **Testing set**: a set of examples used only to assess the performance of a fully-trained classifier. It is used to estimate the error rate after we have chosen the final model. After assessing the final model on the test set, YOU MUST NOT tune the model any further!



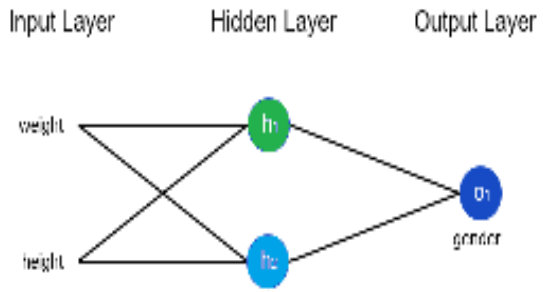
Model Hyperparameter

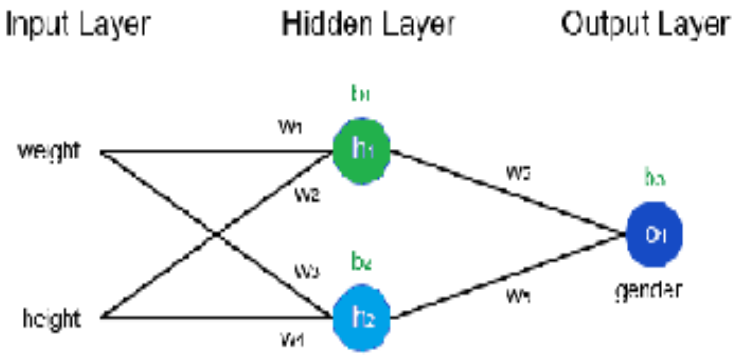
- **Model hyperparameters** : parameters that are external to the model and whose value cannot be estimated from data and must be specified before training.
  - The best value for a model on a given problem is unknown.
  - Can use rules of thumb, copy values used on other issues, or search for the best value using heuristics.

Model Parameters

- In contrast, **model parameters** are parameters which internal to the model and whose values are learned from data during training
  - The weights in an artificial neural network.
  - The support vectors in a support vector machine.
  - The coefficients in a linear regression or logistic regression.

Model vs Hyper Parameters





(Model Design + Hyperparameters) → Model Parameters

The building blocks:

- # Layers
- Activations
- Optimizers

...

The knobs that you can turn:

- Learning Rate
- Dropout

...

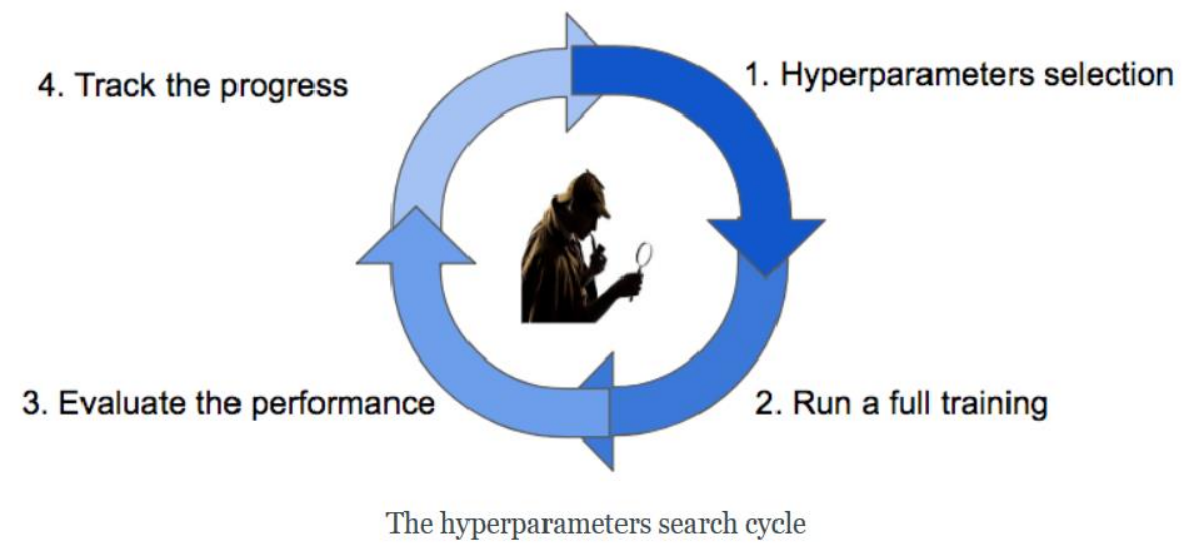
The variables learned from the data:

- weights

...

Hyperparameter Tuning

- The process of **hyperparameter tuning** (also called **hyperparameter optimization**) means finding the combination of hyperparameter values for a machine learning model that performs the best - as measured on a validation dataset - for a problem.



The hyperparameters search cycle

Hyperparameter Tuning strategies

1. **Baby Sitting**: select hyperparameters based on intuition/experience/guessing, train the model with the hyperparameters, and score on the validation data. Repeat process until you run out of patience or are satisfied with the results.
2. **Grid Search**: set up a grid of hyperparameter values and for each combination, train a model and score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient!
3. **Random search**: set up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources.

Grid Search Tuning

Grid Search: The Grid Search is a naive approach whose strategy is to simply trying every possible configuration. Steps are summarized as follows:

1. Define a grid of n dimensions, where each dimension corresponds to a hyperparameter. e.g. n = 3, {learning_rate, dropout_rate, batch_size}
2. For each dimension, define a range of possible values:

   e.g. batch_size = [4, 8, 16, 32, 64, 128, 256]
3. Implement every possible combination of values for this grid and wait for the results to establish the best one:

   e.g. C1 = (0.1, 0.3, 4) -> acc = 92%,

   C2 = (0.1, 0.35, 4) -> acc = 92.3%, etc..

Sample Grid Search Space

P1 \ p2	0.1	0.2	0.3	0.4
0.5	0.565	0.565	0.778	0.257
0.4	0.257	0.987	0.125	0.288
0.3	0.744	0.387	0.877	0.441
0.2	0.977	0.298	0.247	0.801
0.1	0.559	0.542	0.899	0.979

Grid Search Coding

```
from sklearn.model_selection import GridSearchCV
 model = GradientBoostingRegressor()
parameters = {'learning_rate': [0.01,0.02,0.03], 'subsample'     : [0.9, 0.5, 0.2],
                  'n_estimators' : [100,500,1000], 'max_depth'     : [4,6,8] }
grid = GridSearchCV(estimator=model, param_grid = parameters, cv = 2, n_jobs=-1)
grid.fit(X_train, y_train)

print("\n The best estimator across ALL searched params:\n", grid.best_estimator_)
print("\n The best score across ALL searched params:\n", grid.best_score_)
 print("\n The best parameters across ALL searched params:\n", grid.best_params_)
```

Random Search Tuning

Random Search: The Random Search is similar to Grid Search except that it picks the parameter values randomly from the configuration space. Random search goes as follows:

1. Define a grid of n dimensions, where each dimension corresponds to a hyperparameter. e.g. n = 3, {learning_rate, dropout_rate, batch_size}
2. For each dimension, specify the distributions to sample from:

   e.g. 'learn_rate': sp_randint (0.001, 0.01),
3. Randomly pick a combination of values for this grid, execute, and wait for the best performance:

   e.g. C1 = (0.1, 0.3, 4) -> acc = 92%,

   C2 = (0.1, 0.35, 4) -> acc = 92.3%, etc..

RandomSearch Implementation

```
from sklearn.model_selection import RandomizedSearchCV
 param_dist = {"max_depth": [3, 5],
               "max_features": sp_randint(1, 11),
               "min_samples_split": sp_randint(2, 11),
               "min_samples_leaf": sp_randint(1, 11),
               "bootstrap": [True, False]
              }
# run randomized search
n_iter_search = 20
random_search = RandomizedSearchCV(model2, param_distributions=param_dist,
                                   n_iter=n_iter_search)
```

Demo Time !!!

Activity 3:

Experiment on using your chosen strategy to optimize the hyperparameters of the Neural Network for multiple regression and report on the highest performance that you were able to generate.