twiml

AI Enterprise Workflow Study Group

Course 4, Week 2

4/25/2020

Agenda

- Check in
- Discussion
- Next steps

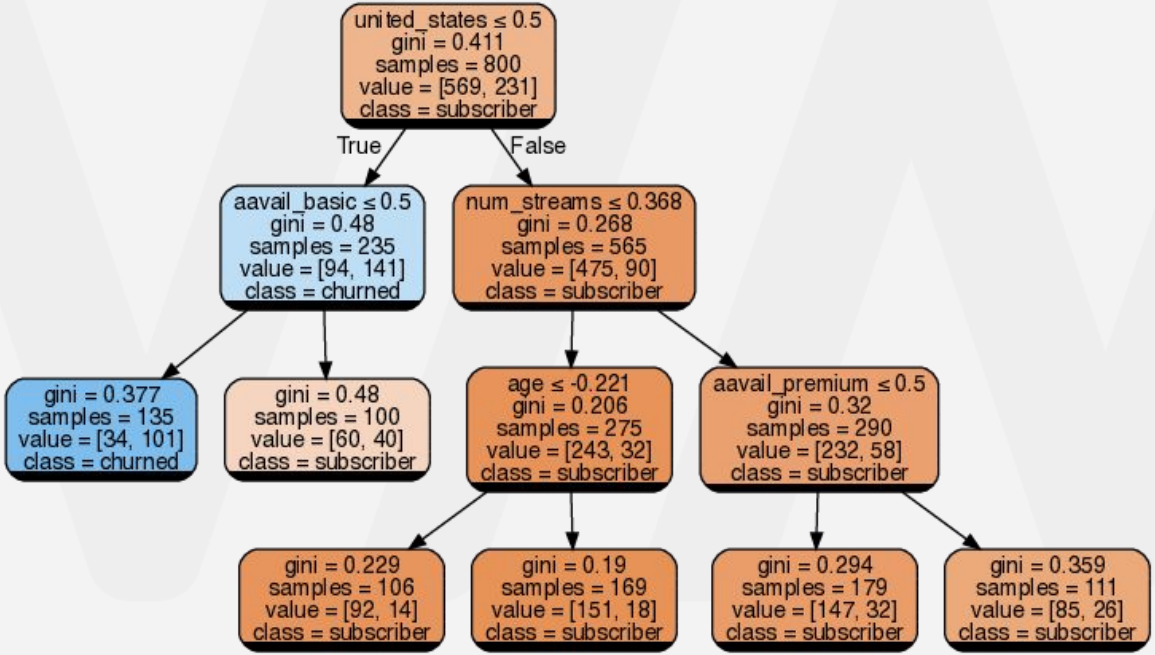
Course & Study Group Schedule

AI Enterprise Workflow Study Group		
Session	**Topic**	**Date**
Overview Webinar	Webinar with instructor, Ray Lopez	15-Feb
Course 1 Week 1	Course intro	22-Feb
Course 1 Week 2	Data ingestion, cleaning, parsing, assembly	29-Feb
Course 2 Week 1	Exploratory data analysis & visualization	7-Mar
Course 2 Week 2	Estimation and NHT	14-Mar
Course 3 Week 1	Data transformation and feature engineering	21-Mar
Course 3 Week 2	Pattern recognition and data mining best practices	28-Mar
Course 4 Week 1	Model evaluation and performance metrics	18-Apr
Course 4 Week 2	Building machine learning and deep learning models	25-Apr
Course 5 Week 1	Deploying models	2-May
Course 5 Week 2	Deploying models using Spark	9-May
Course 6 Week 1	Feedback loops and monitoring	16-May
Course 6 Week 2	Hands on with OpenScale and Kubernetes	23-May
Course 6 Week 3	Captsone project week 1	30-May
Course 6 Week 4	Captsone project week 2	6-Jun

Course 4 Week 1 learning objectives

1. Explain the use of tree-based algorithms in supervised learning applications
2. Explain the use of neural networks in supervised learning applications
3. Discuss the major variants of neural networks and recent advances
4. Create and test an instance of Watson Visual Recognition
5. Create a neural net model in Tensorflow

Tree-Based Methods



Learning Decision Trees

The decision tree learning algorithm recursively learns the tree as follows:

1. Assign all training data to the root of the tree. Set current node to root node.
2. For each feature
    1. Partition all data instances at the node by the feature value.
    2. Compute the *impurity* from the partitioning.
3. Identify feature that minimizes the impurity. Set this feature to be the splitting criterion at the current node.
4. Partition all instances according to attribute value of the best feature.
5. Denote each partition as a child node of the current node.
6. For each child node:
    1. If the child node is “pure” (has instances from only one class), or we’ve reached maximum depth or minimum samples per node, tag it as a leaf and return.
    2. If not set the child node as the current node and recurse to step 2.

Adapted from: https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/

Impurity Measures

Impurity Measure is cost function for tree construction

1. Classification

Gini = Impurity Measure for classification. Gini Impurity is the probability of incorrectly classifying a randomly chosen element in the dataset if it were randomly labeled according to the class distribution in the dataset.

Basically tells you how mixed the classes are (impure) in each of the groups created by a split.

Gini Impurity >= 0, the smaller the better

1. Regression

Use RMSE or MAE for regression problems.

Decision Trees: Advantages

Advantages:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

Decision Trees: Disadvantages

Disadvantages:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Ensemble Methods: Bagging

Bagging = Bootstrap Aggregation

A Bagging classifier/regressor is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Random Forests = Bagging applied to decision trees where test set and features are randomized.

Ensemble Methods: Boosting

Boosting = Learning weak classifiers with respect to a distribution and adding them to a final strong classifier

Gradient boosting = algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction

Bagging (RFs) vs Boosting

If we use decision trees as our base model in boosting, they tend to be very simple with very few splits. This is in contrast to random forests, which tend to have fully grown trees as base models. The other important distinction between random forests and boosting approaches is that boosting builds its ensemble sequentially, where random forests can train the base models in parallel.

Decision Trees: In Practice

- Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit.
- Consider performing dimensionality reduction (PCA, ICA, or Feature selection) beforehand to give your tree a better chance of finding features that are discriminative.
- Understanding the decision tree structure will help in gaining more insights about how the decision tree makes predictions, which is important for understanding the important features in the data.
- Visualise your tree as you are training by using the `export` function. Use `max_depth=3` as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.
- Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use `max_depth` to control the size of the tree to prevent overfitting.
- Use `min_samples_split` or `min_samples_leaf` to ensure that multiple samples inform every decision in the tree, by controlling which splits will be considered. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data. Try `min_samples_leaf=5` as an initial value. If the sample size varies greatly, a float number can be used as percentage in these two parameters. While `min_samples_split` can create arbitrarily small leaves, `min_samples_leaf` guarantees that each leaf has a minimum size, avoiding low-variance, over-fit leaf nodes in regression problems. For classification with few classes, `min_samples_leaf=1` is often the best choice.

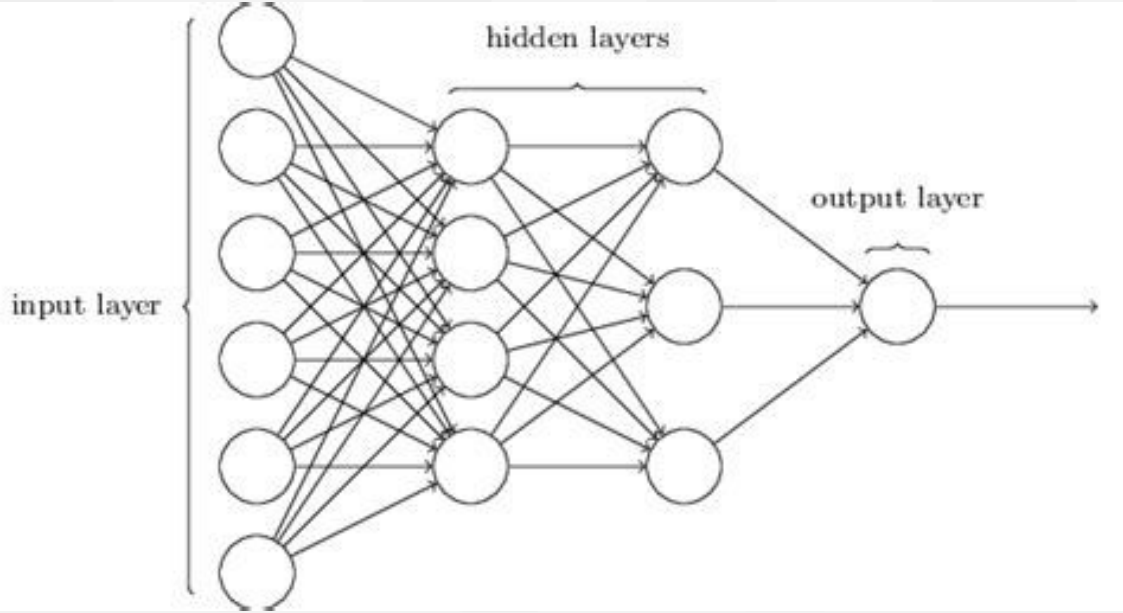
Decision Trees: In Practice, cont'd

- Balance your dataset before training to prevent the tree from being biased toward the classes that are dominant. Class balancing can be done by sampling an equal number of samples from each class, or preferably by normalizing the sum of the sample weights (`sample_weight`) for each class to the same value. Also note that weight-based pre-pruning criteria, such as `min_weight_fraction_leaf`, will then be less biased toward dominant classes than criteria that are not aware of the sample weights, like `min_samples_leaf`.
- If the samples are weighted, it will be easier to optimize the tree structure using weight-based pre-pruning criterion such as `min_weight_fraction_leaf`, which ensure that leaf nodes contain at least a fraction of the overall sum of the sample weights.
- All decision trees use `np.float32` arrays internally. If training data is not in this format, a copy of the dataset will be made.
- If the input matrix X is very sparse, it is recommended to convert to sparse `csc_matrix` before calling fit and sparse `csr_matrix` before calling predict. Training time can be orders of magnitude faster for a sparse matrix input compared to a dense matrix when features have zero values in most of the samples.

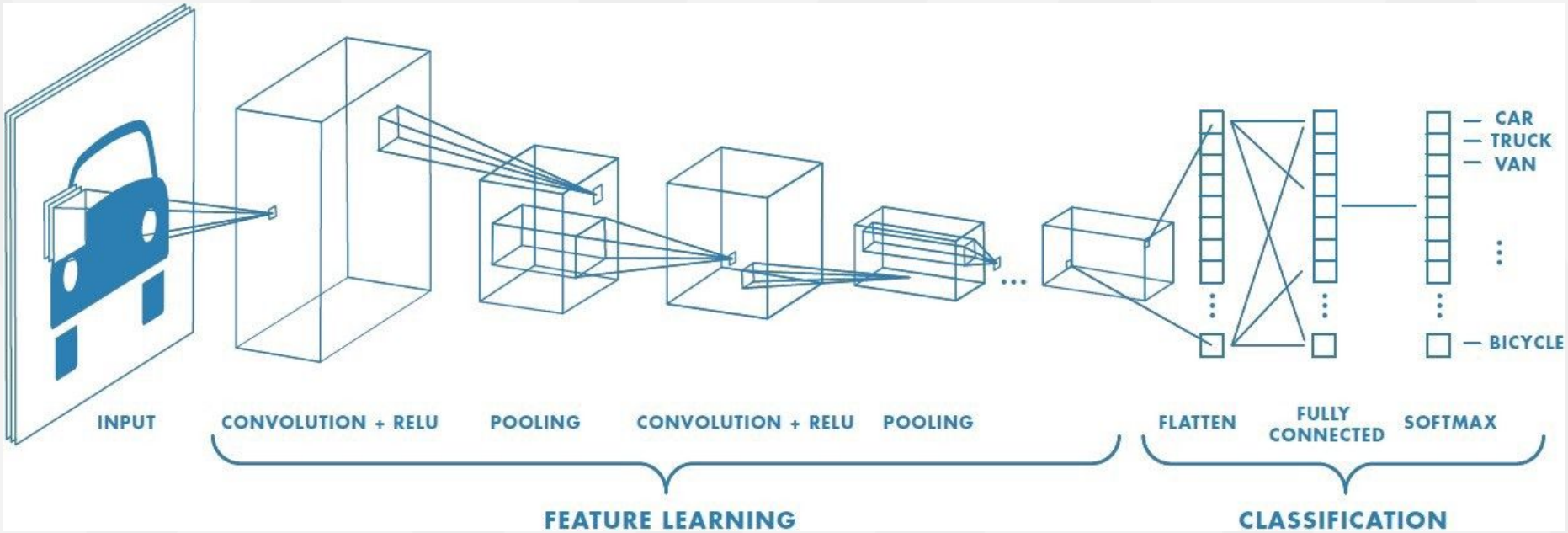
Neural Networks

- Multi-layer perceptron. Feed-forward neural network used for supervised learning. Often uses multiple layers and non-linear activation functions. Can be applied to most supervised learning tasks.
- Autoencoders. It is a feed forward neural network that can be used to predict the feature matrix. Autoencoders may be thought of as being a special case of feedforward network used for unsupervised learning tasks. Example applications include dimension reduction and de-noising.
- Convolutional neural network. A convolutional neural network (CNN, or ConvNet) uses image filters or kernels to learn patterns in a feature matrix. Commonly used to detect patterns in images and video.
- Recurrent neural network. In RNNs connections between nodes can be cyclical, giving the network memory. Used for sequences: handwriting, speech recognition, time series. One commonly used RNN architecture is the long short-term memory.

Multilayer Perceptron (MLP)



CNNs


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Additional Discussion

What did you learn?

What stumbling blocks did you run into?

How do these lessons relate to your experience?

What did you learn/find interesting in this week's lesson?

What are you doing as homework?

What interesting resources have you found?

Other?