UDACITY

‹ Return to "Machine Learning Engineer Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Machine Learning Capstone Project

REVIEW

CODE REVIEW

HISTORY

Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

This is a very cool analysis and a great read to a very real world and practical problem. You have demonstrated a full understanding of the entire machine learning pipeline and your report definitely gets the readers attention with the results you have achieved. You just have to expand in a couple of these sections, but will greatly improve your report. Please check out some of these other ideas presented here and we look forward in seeing your next submission!!

Definition

Student provides a high-level overview of the project in layman's terms. Background information such as the problem domain, the project origin, and related data sets or input data is given.

Nice work here with your opening section, as you have given good starting paragraphs to outline the project and have provided background information on the problem domain. Definitely a real world problem.

And you have provided good research to back your claims. It is always important to provide similiar research on such a topic.

The problem which needs to be solved is clearly defined. A strategy for solving the problem, including discussion of the expected solution, has been made.

> "In this project, I create a desktop application that capable of composing multiple services from different cloud providers (e.g. IBM, Google, Rackspace,). The application uses a predictor trained using QoS (quality of services) dataset1 to predict future provision values to accurately select providers."

Problem statement is clearly defined here. Would also recommend explicitly mentioning that this would be a regression problem in this section.

And very nice job mentioning your machine learning pipeline here, as this gives the reader some ideas in what is to come in your report and how you plan on solving this important task.

Metrics used to measure performance of a model or result are clearly defined. Metrics are justified based on the characteristics of the problem.

RMSE is an excellent metric to use for such a problem, but for this section please provide some justification for why RMSE has based chosen on this dataset and/or problem domain. For example, why wasn't something like MAE chosen?

Maybe think about how outliers are treated differently?

(https://www.quora.com/What-is-the-difference-between-squared-error-and-absolute-error)

Analysis

If a dataset is present, features and calculated statistics relevant to the problem have been reported and discussed, along with a sampling of the data. In lieu of a dataset, a thorough description of the input space or input data has been made. Abnormalities or characteristics about the data or input that need to be addressed have been identified.

Off to a good start here and glad that you the different features included in the data. Therefore lastly for this section, make sure you also show a **sample of your dataset**. As this allows the reader to get an understanding of the structure of the data you are working with.

Added: I see that you do this later, so will be fine.

A visualization has been provided that summarizes or extracts a relevant characteristic or feature about the dataset or input data with thorough discussion. Visual cues are clearly defined.

Good time series visuals!!! You might also look into using facebook's prophet library as well for some time series analysis.

You might also check out using some more advanced plotting libraries such as

- plot.ly: Modern Visualization for the Data Era. Where you can create really cool interactive visuals in jupyter notebooks and web apps!
- seaborn: statistical data visualization. Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. My favorite method in seaborn is catplot(), where you can plot categorical features effortlessly!

Algorithms and techniques used in the project are thoroughly discussed and properly justified based on the characteristics of the problem.

> "LSTMs are designed to avoid the long-term dependency problem which is in traditional RNN. Remembering information for long periods of time is practically their default behavior."

Can you please go into a bit more detail in these comments. Why does an LSTM cell handle long-term dependency better than a simple RNN? You should also mention the 'vanishing gradient' problem and go into a bit of detail in the different 'gates' for the model.

(http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Student clearly defines a benchmark result or threshold for comparing performances of solutions obtained.

Comparing against another paper is a fine idea and glad that you mention the results.

Benchmarking is the process of comparing your result to existing method or running a very simple machine learning model, just to confirm that your problem is actually 'solvable'.

Methodology

All preprocessing steps have been clearly documented. Abnormalities or characteristics about the data or input that needed to be addressed have been corrected. If no data preprocessing is necessary, it has been clearly justified.

Excellent job documenting all your pre-processing steps.

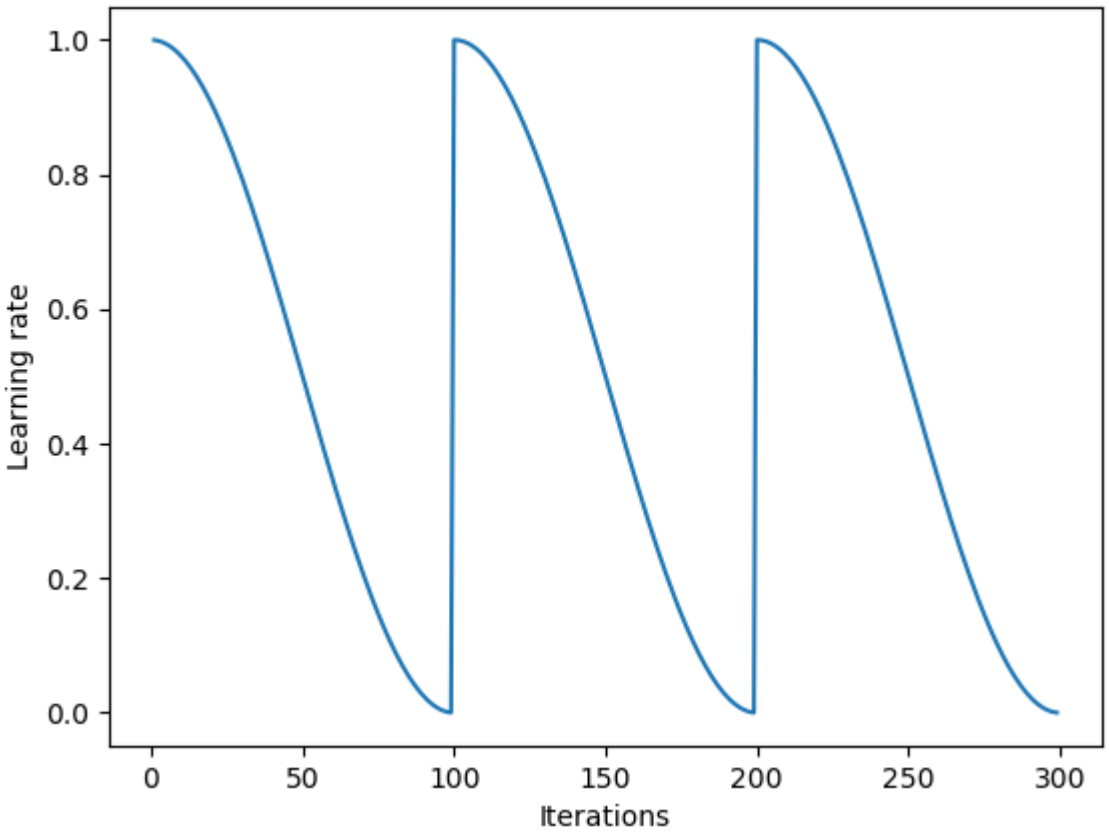
Another feature transformation idea, akin to standardization, for preparing continuous features for neural networks is with GaussRank. The purpose of this is to make the distribution of the transformed ranks Gaussian.

Can check how this is done here.
(https://github.com/zygmuntz/gaussrank/blob/master/gaussrank.py)

The process for which metrics, algorithms, and techniques were implemented with the given datasets or input data has been thoroughly documented. Complications that occurred during the coding process are discussed.

Very solid step by step process here, as it is quite clear in how you approached this problem. Your results would definitely be replicable.

Another idea would be to check out using Cyclical Learning Rates for Training Neural Networks. This is where we simply keep increasing the learning rate from a very small value, until the loss stops decreasing and then bump it up once more. We can plot the learning rate across batches to see what this looks like.



The process of improving upon the algorithms and techniques used is clearly documented. Both the initial and final solutions are reported, along with intermediate solutions, if necessary.

Nice work documenting your hyper-parameter tuning you performed here with your additional dropout later, as this is a great way to improve your model. And you have made it very clear in the parameter you tried and the results.

Could also look into something like gridSearch to tune hyper-parameters for your model, such as the learning rate, optimizers, batch_size, etc... As we can actually use Sklearn with Keras! Check out this example

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import Dense

def build_classifier(optimizer):
    classifier = Sequential()
    classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 're
lu', input_dim = 11))
    classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 're
lu'))
    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'si
gmoid'))
    classifier.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics
 = ['accuracy'])
    return classifier
classifier = KerasClassifier(build_fn = build_classifier)
parameters = {'batch_size': [25, 32],
              'epochs': [100, 500],
              'optimizer': ['adam', 'rmsprop']}
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 10)
grid_search = grid_search.fit(X_train, y_train)
best_parameters = grid_search.best_params_
best_accuracy = grid_search.best_score_
```

Results

The final model's qualities — such as parameters — are evaluated in detail. Some type of analysis is used to validate the robustness of the model's solution.

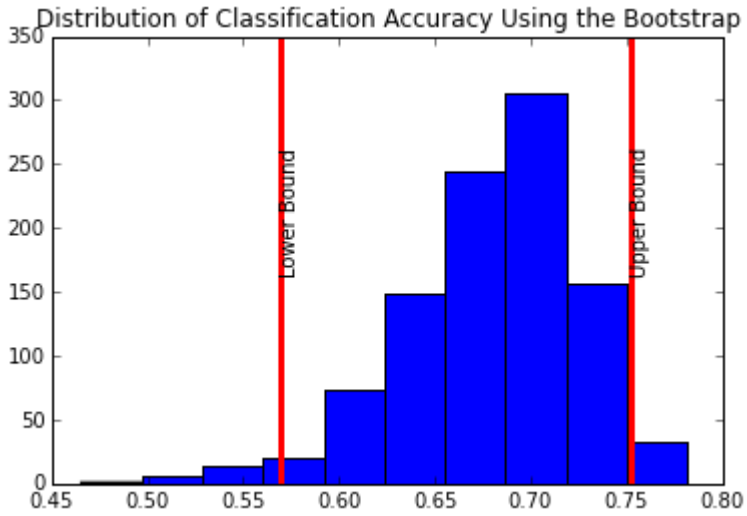
You have good analysis of your final models and excellent analysis to validate the robustness of the model's solution by showing your plots for predictions and your box plots.

X and Y labels in all of your plots would be nice to see.

Maybe one other idea would be to plot and 95% confidence interval to determine if the model is robust with bootstrapping. Here might be an example with the boston housing dataset.

```
from sklearn.utils import resample
import matplotlib.pyplot as plt

data = pd.read_csv('housing.csv')
values = data.values
# configure bootstrap
n_iterations = 1000
n_size = int(len(data) * 0.50)

# run bootstrap
stats = []
for i in range(n_iterations):
    # prepare train and test sets
    train = resample(values, n_samples=n_size)
    test = np.array([x for x in values if x.tolist() not in train.tolist()])
    # model
    model = DecisionTreeRegressor(random_state=100)
    model.fit(train[:,:-1], train[:,-1])
    score = performance_metric(test[:,-1], model.predict(test[:,:-1]))
    stats.append(score)

# confidence intervals
alpha = 0.95
p = ((1.0 - alpha) / 2.0) * 100
lower = max(0.0, np.percentile(stats, p))
p = (alpha + ((1.0 - alpha) / 2.0)) * 100
upper = min(1.0, np.percentile(stats, p))

# plot
plt.hist(stats)
plt.axvline(lower, color='red', lw=3)
plt.text(lower, n_iterations // 4, 'Lower Bound', rotation=90)
plt.axvline(upper, color='red', lw=3)
plt.text(upper, n_iterations // 4, 'Upper Bound', rotation=90)
plt.title('Distribution of Classification Accuracy Using the Bootstrap')
plt.show()

print('%.1f confidence interval %.1f%% and %.1f%%' % (alpha*100, lower*100, upper*10
0))
```

(https://machinelearningmastery.com/calculate-bootstrap-confidence-intervals-machine-learning-results-python/)

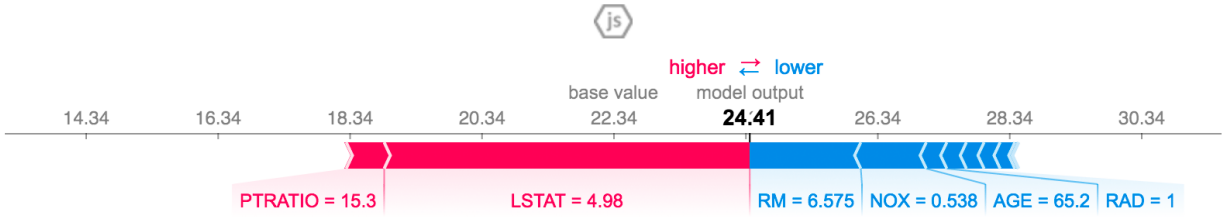


The final results are compared to the benchmark result or threshold with some type of statistical analysis. Justification is made as to whether the final model and solution is significant enough to have adequately solved the problem.

Conclusion

A visualization has been provided that emphasizes an important quality about the project with thorough discussion. Visual cues are clearly defined.

Another really cool idea would be to check out the SHAP library. SHAP (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations, uniting several previous methods and representing the only possible consistent and locally accurate additive feature attribution method based on expectations (see the SHAP NIPS paper for details). This is where you can visualize your machine learning model's predictions with visuals such as



Student adequately summarizes the end-to-end problem solution and discusses one or two particular aspects of the project they found interesting or difficult.

Nice work discussing your final end-to-end problem solution as this reads quite well. I can definitely tell that you have spent a long time on this project as it really shows.

Discussion is made as to how one aspect of the implementation could be improved. Potential solutions resulting from these improvements are considered and compared/contrasted to the current solution.

> "Use more capable hardware and use the tensorflow GPU version instead of CPU one."

Could also look into using floydhub or paperspace or vectordash(which have 1080Ti) to train your models in the cloud.

Quality

Project report follows a well-organized structure and would be readily understood by its intended audience. Each section is written in a clear, concise and specific manner. Few grammatical and spelling mistakes are present. All resources used to complete the project are cited and referenced.

Your writing is very clean and it is very easy to understand what you are saying. I personally thank you as this report is very easy to read :)

Code is formatted neatly with comments that effectively explain complex implementations. Output produces similar results and solutions as to those discussed in the project.

Code is formatted neatly with comments. You might also check out

- this post regarding Docstrings vs Comments.
- Google Style Python Docstrings
- This Best of the Best Practices" (BOBP) guide to developing in Python

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH