Brian Elinsky

Professor Lawrence Fulton, Ph.D.

2020SP_MSDS_422-DL_SEC55

12 May 2020

Assignment 6: Neural Networks

PROBLEM DESCRIPTION

Given an array of pixels and their associated degree of darkness, the objective is to classify each image as the correct digit. As an employee at a financial institution, I need to assess how accurate neural networks are at performing this task, along with recommending an architecture.

RESEARCH DESIGN AND MODELING METHODS

For this experiment, my plan was to fit the MNIST dataset with a neural network model. I would conduct a 2x2 factorial design, comparing 1 vs 2 hidden layers and 200 vs 400 neurons per layer. I didn't plan on conducting any automated hyperparameter tuning via a grid search.

DATA PREPARATION AND MODEL ARCHITECTURE

I used a MinMaxScaler to scale the features to the [0,1] range. This is necessary because I am training the neural network using Gradient Descent. I reshaped the array from (42000, 784) to (42000, 28, 28). Each of the four models are Keras sequential models. This is a fairly simple model. Each layer has one input tensor and one output tensor. The first layer is a Flatten layer. The second and third layers are Dense layers with relu activation. The final layer is a Dense layer with only 10 nodes, and softmax activation. Softmax

activation is necessary because the outputs should sum to 1 for a given training instance. I used a Sparse Categorical Cross-Entropy loss function because this is a classification problem and the dataset is sparse. In order to validate each model, I'm using a 10% validation split. I chose to train each model over 30 epochs.

RESULTS AND MODEL EVALUATION

After 30 epochs of training for each model, the models' validation accuracy continues to increase, and the validation loss continues to decrease. This indicates to me that there is still room to improve my model. Both models with 2 layers outperformed the 1 layer models. However, 400 nodes vs 200 nodes per layer seemed to have minimal impact, when controlling for number of layers. Future would could include adding more epochs of training, or adding more hidden layers. Kaggle scores for my four models are: 0.96942, 0.97, 0.96157, 0.96228 (https://www.kaggle.com/brianelinsky Account User ID 4810027). Additional stats for processing time, training set accuracy, and validation accuracy can be found in my lab notebook.

MANAGEMENT RECOMMENDATIONS

Based off on my research so far, neural networks slightly outperform random forests and PCAs on the MNIST dataset. If management considers a 97% accuracy rate is acceptable, this model can be deployed to production. However, if higher accuracy scores are necessary, further research will be needed to increase accuracy to state of the art scores around 99.8% accuracy.

✔ Selected submissions updated

>_ `kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"`

8 submissions for Brian Elinsky

Sort by Most recent

All Successful Selected

Submission and Description	Public Score	Use for Final Score
model_d_predictions.csv 10 hours ago by Brian Elinsky Neural Network 200 Nodes per Layer 1 Hidden Layer (model d)	0.96228	☐
model_c_predictions.csv 10 hours ago by Brian Elinsky Neural Network 400 Nodes per Layer 1 Hidden Layer (model c)	0.96157	☐
model_b_predictions.csv 10 hours ago by Brian Elinsky Neural Network 400 Nodes per Layer 2 Hidden Layers (model b)	0.97000	☑
model_a_predictions.csv 10 hours ago by Brian Elinsky Neural Network 200 Nodes per Layer 2 Hidden Layers (model a)	0.96942	☐

```
from lib_bre import *
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras

# Load train data
train_data_file = get_dataset_file_path('2020-05-04', 'train.csv')
train = pd.read_csv(train_data_file)

# Remove label
X_train = train.drop(columns='label')
y_train = train['label'].copy().to_numpy()

# Scale and reshape the data
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_train_processed = X_train_scaled.reshape((42000, 28, 28))

# Load test data
test_data_file = get_dataset_file_path('2020-05-04', 'test.csv')
X_test = pd.read_csv(test_data_file)

# Scale and reshape the test data
X_test_scaled = scaler.transform(X_test)
X_test_processed = X_test_scaled.reshape((28000, 28, 28))


def _log(message):
    print('[SimpleTimeTracker] {function_name} {total_time:.3f}'.format(**message))


@simple_time_tracker(_log)
def train_model(hidden_layers, nodes_per_layer, model_name):
    nn_clf = keras.models.Sequential()
    nn_clf.add(keras.layers.Flatten(input_shape=[28, 28]))

    for i in range(hidden_layers):
        nn_clf.add(keras.layers.Dense(nodes_per_layer, activation="relu"))

    nn_clf.add(keras.layers.Dense(10, activation="softmax"))
    nn_clf.compile(loss="sparse_categorical_crossentropy",
                   optimizer="sgd",
                   metrics=["accuracy"])

    history = nn_clf.fit(X_train_processed, y_train, epochs=30, validation_split=0.1)

    # Plot learning curves
    pd.DataFrame(history.history).plot(figsize=(8, 5))
    plt.grid(True)
    plt.gca().set_ylim(0, 1)
    plt.title(str(nodes_per_layer) + " Nodes per Layer, " + str(hidden_layers) + " Hidden Layers")
    plt.savefig(model_name)

    # Make final predictions on test set
    model_predictions = nn_clf.predict_classes(X_test_processed)
    model_predictions_series = pd.Series(model_predictions, name="Label")

    # Output predictions
    image_ids = pd.Series(data=range(1, 28001), name="ImageId")
    output = pd.concat([image_ids, model_predictions_series], axis=1)
    output.to_csv(model_name + "_predictions.csv", index=False)
```

```
train_model(2, 200, "model_a")
train_model(2, 400, "model_b")
train_model(1, 400, "model_c")
train_model(1, 200, "model_d")
```

Digit Recognizer Lab Notebook

2020-05-04

- Define the objective in business terms.
    - The objective is to classify as many digits correctly as possible.
- How will your solution be used?
    - This is not stated in the problem.
- What are the current solutions/workarounds (if any)?
    - Presumably the current solution is a manual one, and we are looking to automate it.
- How should you frame this problem (supervised/unsupervised, online/offline, etc.)?
    - This is an offline supervised learning problem.
- How should performance be measured?
    - Performance will be measured by categorization accuracy.
- Is the performance measure aligned with the business objective?
    - Not sure, as the business objective is not stated.
- What would be the minimum performance needed to reach the business objective?
    - I think above 90% would be pretty good. And above 97% would be very good. http://yann.lecun.com/exdb/mnist/
- What are comparable problems? Can you reuse experience or tools?
    - I haven't solved any comparable problems so far.
- Is human expertise available?
    - Yes, I am an expert (we all are) in handwriting recognition.
- How would you solve the problem manually?
    - I would solve this visually, not looking at the matrix of data. This indicates to me that the current matrix may not be the best way to represent the data. I may need to transform it before I train a model.
- List the assumptions you (or others) have made so far.
    - The data is labeled correctly.
    - The numbers are relatively centered.
- Verify assumptions if possible.

Experiment Plan

This week I've been given explicit directions to do the following:

Begin by fitting a random forest classifier using the full set of 784 explanatory variables and the model training set (train.csv). Record the time it takes to fit the model and then evaluate the model on the test.csv data by submitting to Kaggle.com. Provide your Kaggle.com score and user ID.

Experiment Results

RandomForestClassifier parameters:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini',
'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples':
None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None,
'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,
'n_estimators': 100, 'n_jobs': None, 'oob_score': True, 'random_state': 98,
'verbose': 0, 'warm_start': False}
```

OOB Score: 0.9609761904761904

Kaggle Score: 0.96342 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/)

Time: 32.018 seconds (without parallelization)

2020-05-05

Experiment Plan

My instructions are as follows:

Execute principal components analysis (PCA) on the combined training and test set data together, generating principal components that represent 95 percent of the variability in the explanatory variables. The number of principal components in the solution should be substantially fewer than the 784 explanatory variables. Record the time it takes to identify the principal components.

Using the identified principal components, use the train.csv to build another random forest classifier. Record the time it takes to fit the model and to evaluate the model on the test.csv data by submitting to Kaggle.com. ***Provide your Kaggle.com score and user ID.***

Experiment Results

The PCA reduced the number of features to 332.

RandomForestClassifier parameters:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini',
'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples':
None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None,
'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,
'n_estimators': 100, 'n_jobs': None, 'oob_score': True, 'random_state': 12,
'verbose': 0, 'warm_start': False}
0.9225238095238095
```

OOB Score: 0.9225238095238095

Kaggle Score: 0.93357 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/)

Time:

- 11.166 seconds to train the PCA
- 73.140 seconds (without parallelization) to train the Random Forest Classifier

The issue with this experiment is the information leakage in the PCA. The PCA was fit with both training data and test data. Hence, the final model included some information from the test data. This experiment should be replicated by fitting the PCA model with only training data, and using that model to transform the test data. There was also information leakage in the scaler.

I was surprised that the OOB score and the Kaggle score were so close. I expected the Kaggle score to be lower due to the information leakage.

2020-05-06

Experiment Plan

Repeat the same experiment but without the information leakage.

The experiment we have proposed has a MAJOR design flaw. Identify the flaw. Fix it. Rerun the experiment in a way that is consistent with a training-and-test regimen, and submit this to Kaggle.com. Provide your Kaggle.com score and user ID

Experiment Results

The PCA reduced the number of features to 320.

RandomForestClassifier parameters:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini',
'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples':
None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None,
'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,
'n_estimators': 100, 'n_jobs': None, 'oob_score': True, 'random_state': 12,
'verbose': 0, 'warm_start': False}
```

OOB Score: 0.9203809523809524

Kaggle Score: 0.93114 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/)

Time:

- 8.834 seconds to train the PCA
- 66.2 seconds (without parallelization) to train the Random Forest Classifier

2020-05-11

Experiment Plan

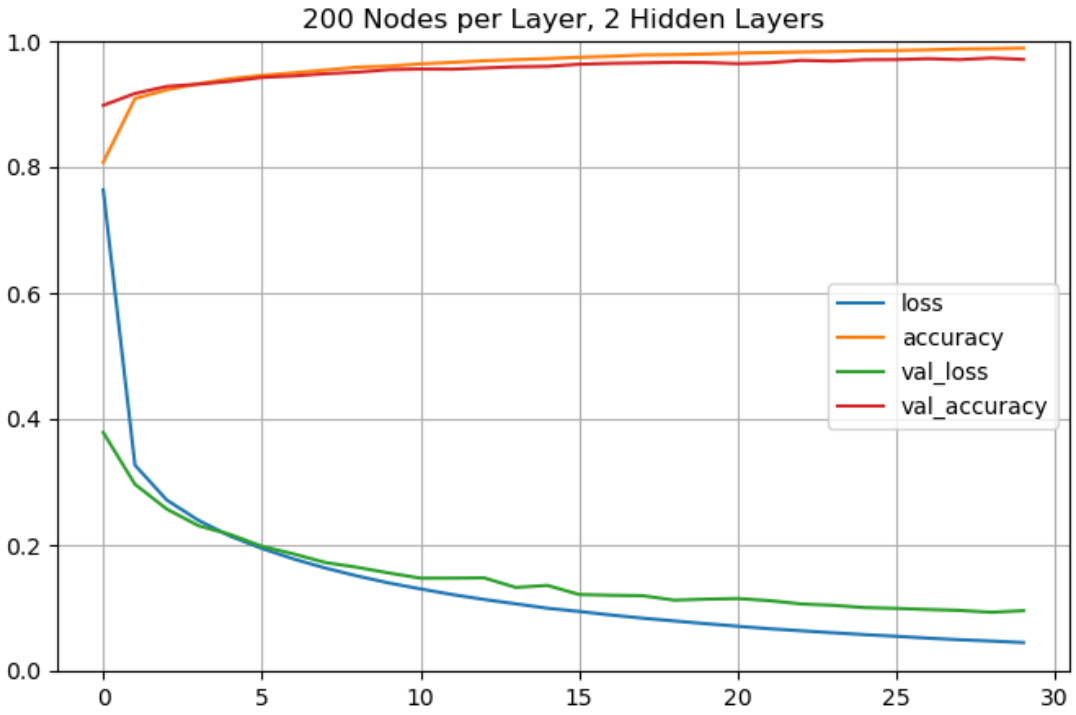
For this experiment, I will fit the MNIST dataset with a neural network model. I plan on conducting a 2x2 factorial design, comparing 1 vs 2 hidden layers and 200 vs 400 neurons per layer. I won't do any automated hyperparameter tuning via a grid search. I plan on submitting all 4 models to Kaggle.

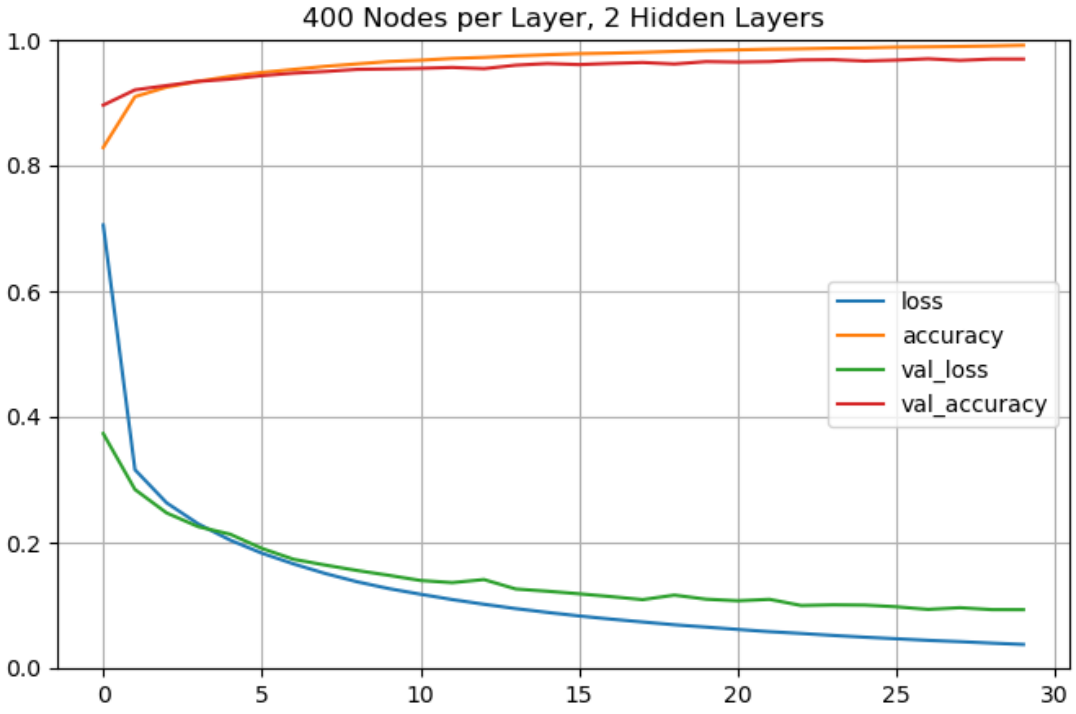
Pre-Processing and Model Decisions

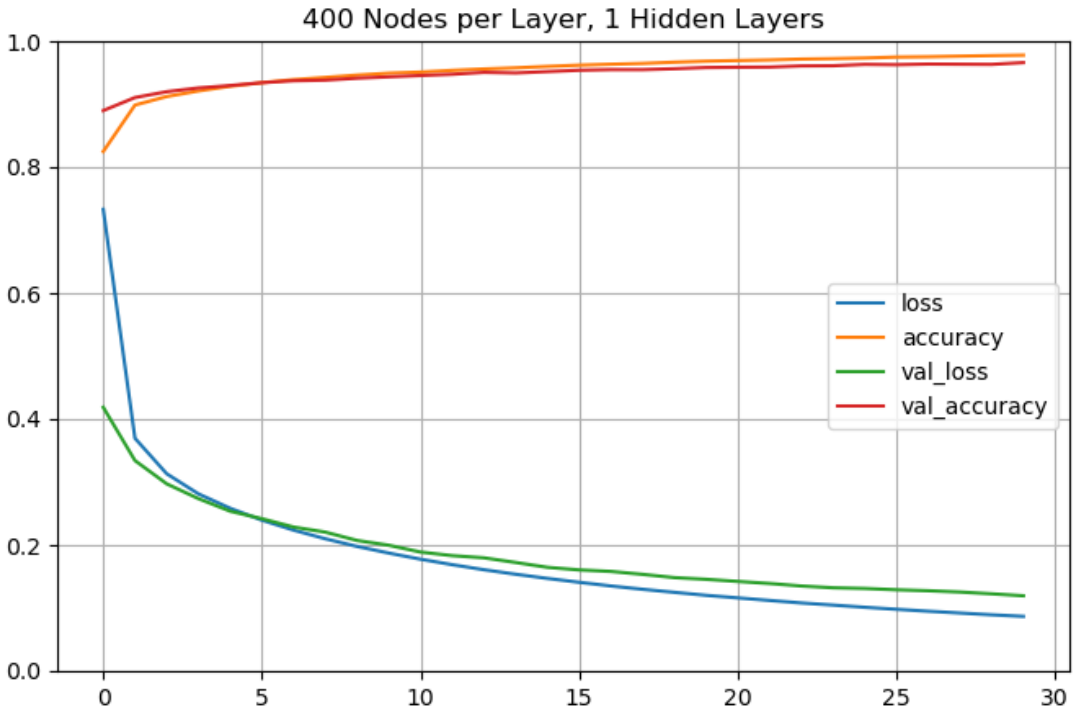
- I used a MinMaxScaler to scale the features to the [0,1] range. This is necessary because I am training the neural network using Gradient Descent.
- I reshaped the array from (42000, 784) to (42000, 28, 28).
- Each of the four models are Keras sequential models. This is a fairly simple model. Each layer has one input tensor and one output tensor.
- The first layer is a Flatten layer. The second and third layers are Dense layers with relu activation. The final layer is a Dense layer with only 10 nodes, and softmax activation. Softmax activation is necessary because the outputs should sum to 1 for a given training instance.
- I used a Sparse Categorical Cross-Entropy loss function because this is a classification problem and the dataset is sparse.
- In order to validate each model, I'm using a 10% validation split.
- I chose to train each model over 30 epochs.

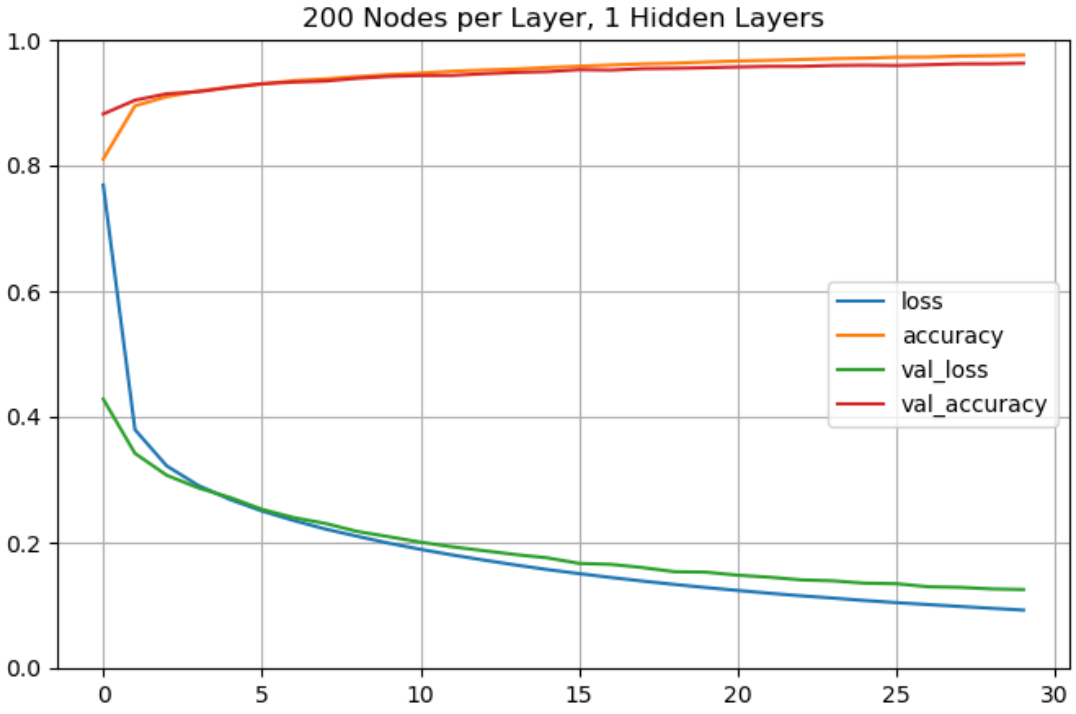
Results and Interpretation

Number of Layers	Nodes per Layer	Processing Time	Training Set Accuracy	Validation Accuracy	Kaggle Score
2	200	121.953	0.9889	0.971	0.96942
2	400	138.201	0.9915	0.9695	0.97
1	400	114.447	0.9778	0.966	0.96157
1	200	104.636	0.9759	0.9629	0.96228


200 Nodes per Layer, 2 Hidden Layers
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
loss
accuracy
val_loss
val_accuracy


400 Nodes per Layer, 2 Hidden Layers
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
loss
accuracy
val_loss
val_accuracy





- On all the models, the validation accuracy continues to increase, and the loss continues to decrease. There may be more I can juice out of this model by increasing the number of epochs.
- Both models with 2 layers performed better than the models with 1 layer. For the models with 2 hidden layers, they seemed to perform equally well, regardless of the number of nodes in the

layer.
- My model likely isn't overfitting. Either its fit well or there's some small amount of underfitting.

Next Steps

- I could try training the model with more epochs. I could also try adding more hidden layers.