EXPLAINATORY MACHINE LEARNING AND BASICS OF NEURAL NETWORK:MINI_BLOG_NOTES

KAGGLE COURSE: EXPLAINATORY MACHINE LEARNING AND INTRO TO DEEP LEARNING.

Source Code and Content License:(Apache 2.0 open source license). [Modified Form used here].

Person Alpha: Explain the Unique Feature of Sigmoid Function?
Person Beta: It outputs as 0 and 1 and used as a output layer in a neural network most of the time. Solves Binary Classification Problem with label such as 0 or 1 i.e., with only output in the form Yes or No.

Person Alpha: Accuracy = total_correct_count /total count, so it can't be use as loss function?
Person Beta: Accuracy is kind of a ratio, it generates the jump in the output, so we use cross entropy for loss function. Since mean square error calculates the distance between actual output and predicted output. Cross Entropy is the distance between actual probability distribution and calculated probability distribution and is used for the loss function calculation with sigmoid function in neural network.

```
PersonAlpha : For the following, write down the function for cross entropy?
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(4, activation='relu', input_shape=[33]),
    layers.Dense(4, activation='relu'),
    layers.Dense(1, activation='sigmoid'),
])
Person Beta:
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['binary_accuracy'],
)
```

We can have early stopping at selected threshold as:

```
early_stopping = keras.callbacks.EarlyStopping(
    patience=10,
    min_delta=0.001,
    restore_best_weights=True,
)

history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=512,
    epochs=1000,
    callbacks=[early_stopping],
    verbose=0, # hide the output because we have so many epochs
)
```

PersonAlpha: Write Concept and Code of Dropout?
PersonBeta: Often overfitting can be produced in neural network due to spurious pattern in the training data, but spurious patter often depends upon weights or conspiracy of the weights, so some of the input layers are removed during iterations of training randomly (internal layers). So, this entire procedure of reducing the spurious pattern is called Dropout.

```
keras.Sequential([

    # ...

    layers.Dropout(rate=0.3), # apply 30% dropout to the next layer

    layers.Dense(16),

    # ...

])
```

Person Alpha : Explain Batch Normalization with Code?
Person Beta: Batch Normalization is an internal process of neural network and usually optimize the neural network using mean and standard deviation of the neural network. And normalize the batch of data withing a scale. Model takes fewer epochs to complete with batch normalization lays

```
layers.Dense(16, activation='relu'),

layers.BatchNormalization(),
```

... or between a layer and its activation function:

```
layers.Dense(16),

layers.BatchNormalization(),

layers.Activation('relu'),
```

PersonAlpha: Code For Batch Normalization and Droput?
PersonBeta:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(1024, activation='relu', input_shape=[11]),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
```

```
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(1),
])
```

Person Alpha: Explain the capacity of the model?
Person Beta: The size and complexity of the pattern of data that model can learn is called the capacity of the model. A neural network can be make wider by adding more functionality(for effectively solving linear problems) and can be make more deep by adding more layers (for effectively solving nonlinear problems).

Person Alpha: Explain iterative steps of stochastic gradient descent?
Person Beta:
It is methods of optimizing the neural network. It is widely used with neural networks.

1. "Sample some training data and run it through the network to make predictions.
2. Measure the loss between the predictions and the true values.
3. Finally, adjust the weights in a direction that makes the loss smaller."

PA:
```
model.compile(

    optimizer="adam",

    loss="mae",

)
```

What is mae in the above code?
PB: mae is known as the Mean square error , is used as a loss function (mean square difference b/w actual and calculated output). It is often used in the compile part of Neural Network functionality which is the next step of Neural Network after defining layers.

Key Concept

"**What's In a Name?**
The gradient is a vector that tells us in what direction the weights need to go. More precisely, it tells us how to change the weights to make the loss change fastest. We call our process gradient descent because it uses the gradient to descent the loss curve towards a minimum. Stochastic means "determined by chance." Our training is stochastic as the minibatches are random samples from the dataset. And that's why it's called SGD!".

PA: Code for fitting the Model?
PB:
```
history = model.fit(
```

```
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=256,
    epochs=10,
)
```

PA: Code for a layer
PB:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    # the hidden ReLU layers
    layers.Dense(units=4, activation='relu', input_shape=[2]),
    layers.Dense(units=3, activation='relu'),
    # the linear output layer
    layers.Dense(units=1),
])
```

PA: Why are the insights from sophisticated machine learning models important:
PB:

They feature does a Machine Learning model predict and why are they important than the other aspect of the models, how they affect particular prediction and model's prediction is a big sense.

PA:
Code for Permutation?

PB:

After applying the model.

```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(my_model, random_state=1).fit(val_X, val_y)
eli5.show_weights(perm, feature_names = val_X.columns.tolist())
```

PA: Why Permutation is important?
PB:

Value in the top of output are the most important features and in the lower are the less important features. The number in each row shows the value of performance decrease with each random shuffling (similar to performance matrix).

Permutation Importance is a measure of feature important calculates most important models post model prediction.

PA :Code for Partial Plot?

PB:
```
from matplotlib import pyplot as plt
from pdpbox import pdp, get_dataset, info_plots

# Create the data that we will plot
pdp_goals = pdp.pdp_isolate(model=tree_model, dataset=val_X, model_features=fe
ature_names, feature='Goal Scored')

# plot it
pdp.pdp_plot(pdp_goals, 'Goal Scored')
plt.show()
```

PA: Functioning of Partial Plots?
PB:

Partial dependence plots are calculated after a model has been fit. The Partial Plot from the original fit model, separated out effect of each feature, by like considering a single row of data. Fitted model is used to predict the outcome. But a value of variable is changed repeatedly for one variable to make a series of predictions.

A few items are worth pointing out as you interpret this plot

Level of confidence is shown by blue shaded area. The Y axis shows the change in prediction from baseline or from the left of baseline.

PA: Application of SHAP Values?
PB: “

- A model says a bank shouldn't loan someone money, and the bank is legally required to explain the basis for each loan rejection
- A healthcare provider wants to identify what factors are driving each patient's risk of some disease so they can directly address those risk factors with targeted health interventions

”

PA: Code for shape Values?
PB :
```
import shap  # package used to calculate Shap values

# Create object that can calculate shap values
explainer = shap.TreeExplainer(my_model)

# Calculate Shap values
```

```
shap_values = explainer.shap_values(data_for_prediction)


shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], data_for_predicti
on)
```

PA: “Shape values SHAP values interpret the impact of having a certain value for a given feature in comparison to the prediction , make if that feature took some baseline value.”

- Instead of some baseline numbers some fixed number is used such .When for a team,number of goals = 3.
  SHAP values do this in a way that guarantees a nice property. Specifically, you decompose a prediction with the following equation:
- `sum(SHAP values for all features) = pred_for_team - pred_for_baseline_values`
- How do you interpret this?
- “It predicted 0.7, whereas the base_value is 0.4979. Feature values causing increased predictions are in pink, and their visual size shows the magnitude of the feature's effect. Feature values decreasing the prediction are in blue. The biggest impact comes from `Goal Scored` being 2. Though the ball possession value has a meaningful effect decreasing the prediction.”

If length of blue bars are substractted from the length of blue bars,then we get the distance b/w the base value and output starting from the base value

Output array[[0.29.0.7]] form

The team is 70% likely to have a player win the award.

“

- `shap.DeepExplainer` works with Deep Learning models.
- `shap.KernelExplainer` works with all models, though it is slower than other Explainers and it offers an approximation rather than exact Shap values. ”

```
# use Kernel SHAP to explain test set predictions
k_explainer = shap.KernelExplainer(my_model.predict_proba, train_X)
k_shap_values = k_explainer.shap_values(data_for_prediction)
shap.force_plot(k_explainer.expected_value[1], k_shap_values[1], data_for_pred
iction)
```

Advanced Plots:

Summary Plots:

```
import shap  # package used to calculate Shap values

# Create object that can calculate shap values
explainer = shap.TreeExplainer(my_model)

# calculate shap values. This is what we will plot.
# Calculate shap_values for all of val_X rather than a single row, to have mor
e data for plot.
shap_values = explainer.shap_values(val_X)
```

```
# Make plot. Index of [1] is explained in text below.
shap.summary_plot(shap_values[1], val_X)
```

“SHAP summary plots give us a birds-eye view of feature importance and what is driving it. This plot is made of many dots. Each dot has three characteristics:”

- “Vertical location shows what feature it is depicting
- Color shows whether that feature was high or low for that row of the dataset
- Horizontal location shows whether the effect of that value caused a higher or lower prediction.”

Dependence Contribution Plots in Code

```
import shap  # package used to calculate Shap values

# Create object that can calculate shap values
explainer = shap.TreeExplainer(my_model)

# calculate shap values. This is what we will plot.
shap_values = explainer.shap_values(X)

# make plot.
shap.dependence_plot('Ball Possession %', shap_values[1], X, interaction_index
="Goal Scored")
```

“Without an argument,for interaction_index,Shapely uses some logic to pick one that seems to be interesting to it accordingly.
The last line only seems to be different from the summary plot.
Each dot represents a row of the data. The horizontal location is the actual value from the dataset, and the vertical location shows what having that value did to the prediction. The fact this slopes upward says that the more you possess the ball, the higher the model's prediction is for winning the *Man of the Match* award.”

“The spread suggests that other features must interact with Ball Possession %. For example, here we have highlighted two points with similar ball possession values. That value caused one prediction to increase, and it caused the other prediction to decrease.”

“But there's a lot they don't show. For instance, what is the distribution of effects? Is the effect of having a certain value pretty constant, or does it vary a lot depending on the values of other feaures. SHAP dependence contribution plots provide a similar insight to PDP's, but they add a lot more detail.”