Google Cloud

GCP ML Engineer Glossary of Terms

Softmax Activation

Softmax is **an activation function that scales numbers into probabilities**

> An analog to help understand the Softmax activation function is to imagine a multiclass competition, such as a race with multiple runners. Each runner represents a class, and the goal is to determine the probability of each runner winning the race.

The Softmax activation function has several important properties that make it well-suited for use in multiclass classification problems:

1. **Normalization:** The Softmax activation function normalizes the input vector into a probability distribution over multiple classes, with the sum of all the probabilities equal to 1.
2. **Monotonicity:** The Softmax activation function is monotonic, meaning that increasing the value of any element in the input vector will increase the corresponding probability in the output.
3. **Interpretability:** The Softmax activation function provides an interpretable output, with each element of the output vector representing the probability of the corresponding class.

Recall in softmax Activation:

Recall in the context of Softmax activation refers to the ability of a model to correctly identify all the positive instances of a class. In multiclass classification problems, recall is a measure of the model's ability to identify all positive instances of each class.

The Softmax activation function is used in multiclass classification problems to provide a probability distribution over multiple classes. In this context, recall can be used to evaluate the performance of the model for each class.

For example, if a model is trained to classify images of handwritten digits, recall for the digit "4" would be a measure of the model's ability to correctly identify all instances of the digit "4" in the test data. If the model incorrectly classifies some instances of the digit "4" as another digit, this would lower the recall for the digit "4".

In summary, recall in the context of Softmax activation refers to the ability of a model to correctly identify all positive instances of a class in a multiclass classification problem. The Softmax activation function is used to provide a probability distribution over multiple classes, and recall can be used to evaluate the performance of the model for each class.

- Recalls:

Precision = TruePositives / (TruePositives + FalsePositives) Recall = TruePositives / (TruePositives + FalseNegatives) A. Increase recall -> will decrease precision B. Decrease recall -> will increase precision C. Increase the false positives -> will decrease precision D. Decrease the false negatives -> will increase recall, reduce precision

'scale-tier' parameter:

Changing the scale tier does not impact performance–only speeds up training time. Epochs, Batch size, and learning rate all are hyperparameters that might impact model accuracy.

Feature Crossing:

In feature crossing,

- **you create a new feature by combining or interacting two or more existing features.** This new feature captures the relationship between the original features, and can provide more information to the model, allowing it to make better predictions.

Metrics Part1:

Optimization objectives for classification or regression models

When you train a model, Vertex AI selects a default optimization objective based on your model type and the data type used for your target column.

Classification models are best for:

Optimization objective	API value	Use this objective if you want to...
AUC ROC	maximize-au-roc	Maximize the area under the receiver operating characteristic (ROC) curve. Distinguishes between classes. Default value for binary classification.
Log loss	minimize-log-loss	Keep prediction probabilities as accurate as possible. Only supported objective for multi-class classification.
AUC PR	maximize-au-prc	Maximize the area under the precision-recall curve. Optimizes results for predictions for the less common class.

Google Cloud

Precision at Recall	maximize-precision-at-recall	Optimize precision at a specific recall value.
Recall at Precision	maximize-recall-at-precision	Optimize recall at a specific precision value.

Regression models are best for:

Optimization objective	API value	Use this objective if you want to...
RMSE	minimize-rmse	Minimize root-mean-squared error (RMSE). Captures more extreme values accurately. Default value.
MAE	minimize-mae	Minimize mean-absolute error (MAE). Views extreme values as outliers with less impact on model.
RMSLE	minimize-rmsle	Minimize root-mean-squared log error (RMSLE). Penalizes error on relative size rather than absolute value. Useful when both predicted and actual values can be quite large.

Dropout Layers

- The Dropout layers allows us to prevent a model from overfitting
- randomly setting the outgoing edges of hdden units to “0” at each update of the training phase

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

https://keras.io/api/layers/regularization_layers/dropout/

Features / Parameters

- increase / decrease Feature Parameters ? -
    - Example: “Age” as a Feature - Age range - 30-60 as a feature parameter
- increase / decrease regularization ?
    - done by adding a penalty term to the model's cost function. This penalty term, called the regularization term, discourages the model from assigning too much weight to any one feature
    - Example: too prevent Overfitting models:
        - L1 regularization adds a penalty term that is proportional to the absolute value of the weights

- L2 regularization adds a penalty term that is proportional to the square of the weights.

Metrics Part2:

VertexAI Metrics:

1. **Accuracy:** This metric measures the proportion of correct predictions made by the model.
2. Precision: This metric measures the proportion of positive predictions that are actually positive.
3. **Recall (Sensitivity):** This metric measures the proportion of actual positive cases that are correctly identified as positive by the model.
4. **F1-score:** This metric is the harmonic mean of precision and recall, and provides a single metric that balances both.
5. **AUC-ROC:** This metric measures the ability of a binary classifier to distinguish between positive and negative classes.
6. **Mean Squared Error (MSE):** This metric measures the average squared difference between the predicted values and the true values.
7. **Mean Absolute Error (MAE):** This metric measures the average absolute difference between the predicted values and the true values.
8. **Confusion Matrix:** This matrix summarizes the true positive, true negative, false positive, and false negative predictions made by the model.
9. **Log Loss:** This metric measures the performance of a classifier where the prediction input is a probability value between 0 and 1.
10. **R-squared:** This metric measures the proportion of variance in the target variable that is explained by the model.

DATAFLOW:

you are building a data pipeline using google cloud dataflow sdk, this pipeline is going to perform operations of data using conditional and for loops creating a branch pipeline

Which of the following concepts should be used to achieve this?

A) pardo B) PCollection C) Transform D) Pipeline

A) pardo (short for "Parallel Do") should be used to perform operations on data using loops in a Google Cloud Dataflow pipeline. Pardo allows you to apply a user-defined function to each element of a PCollection in parallel.

B) PCollection (short for "Parallel Collection") is a fundamental concept in Dataflow, representing a distributed and parallel collection of data. PCollection is used as input and output of various Transforms.

C) Transform is a fundamental concept in Dataflow, representing a computation that transforms one or more PCollections into one or more other PCollections.

D) Pipeline is the top-level concept in Dataflow, representing a directed acyclic graph (DAG) of Transforms that are executed together to perform a data processing pipeline.

Reguralization, L1 vs L2

- Regularization = to prevent overfitting in a machine learning model.

What is the difference between L1 and L2 ridge regression in machine learning in very simplistic terms?

The main difference between L1 and L2 Ridge Regression is the type of regularization used. L1 regularization adds "absolute values of magnitude" of coefficient as a penalty term to the loss function, while L2 regularization adds "squared magnitude" of coefficient as a penalty term.

In simpler terms,

- L1 regularization will force the model to use only a subset of the features by shrinking some of the coefficients to zero. This can be useful for feature selection. On the other hand,
- L2 regularization will not shrink any coefficients to zero, but it will keep all of them small and close to zero, which can help to prevent overfitting.

Coefficient

What is a coefficient in machine learning?

A coefficient in machine learning is a value that represents the strength of the relationship between a particular feature or input variable and the output variable. It is used in models such as linear regression, where the coefficients are used to calculate the predicted output value based on the input values. In simpler terms, a coefficient is a number that tells us how important a specific feature is in determining the outcome.

- A higher coefficient means that the feature has a stronger effect on the outcome
- lower coefficient means that the feature has a weaker effect.

L2 ridge regression

L2 Ridge Regression is a variation of linear regression that uses "L2 regularization" to prevent overfitting. In simpler terms, it's a way of adding a small amount of bias to the model to stop it

from becoming too complex, and to improve its generalization. The regularization term is the sum of the squares of the coefficients (except the intercept) multiplied by a scalar value lambda. This forces the coefficients to be close to zero but not exactly zero. This approach helps to reduce the model complexity and multicollinearity.

How to study for (and pass) Google's Professional Machine Learning Engineer certification

AUCPR vs AUC ROC

AUC PR Optimize results for predictions for the less common class. it is straightforward to answer, you just have to capture key word to get the right way. (Almost banlanced Or Imbalanced)
https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

When to Use ROC vs. Precision-Recall Curves? Generally, the use of ROC curves and precision-recall curves are as follows:

- **ROC curves should be used when there are roughly equal numbers of observations for each class.**
- **Precision-Recall curves should be used when there is a moderate to large class imbalance.**

Terminology

TensorFlow:

TFRecords

Dataset for large Datasets (that wont fit in Memory)

so that they can be loaded in small portions (batches) and processed on-the-fly during training.

TensorFlow Functions for Pipelines:

Here are 10 important TensorFlow functions for machine learning pipelines:

1. `**tf.data.Dataset**`: This function creates a dataset object that can be used to efficiently load and preprocess large datasets for training and evaluation.
2. `map`: This function applies a given function to each element of a dataset, allowing for data preprocessing and augmentation.
3. `batch`: This function combines elements of a dataset into batches, which can be used for training and evaluation.

4. `shuffle`: This function randomly shuffles elements in a dataset, which is important for training deep learning models.
5. `repeat`: This function repeats a dataset a specified number of times, which is useful for training models on large datasets.
6. `tf.keras.Model`: This function creates a model in TensorFlow using the Keras API. The model can be trained and evaluated using the `fit` method.
7. `compile`: This function compiles a Keras model by specifying the optimizer, loss function, and metrics used for training and evaluation.
8. `fit`: This function trains a Keras model on a dataset. It can be used with `Dataset` objects or with arrays of data.
9. `predict`: This function makes predictions on new data using a trained model.
10. `evaluate`: This function evaluates a trained model on a dataset, computing metrics such as accuracy, precision, recall, and F1-score.

TENSORFLOW METRICS:

The `tf.keras.metrics` module in TensorFlow provides a variety of metrics for evaluating machine learning models. Here are some of the most commonly used functions in this module:

1. `tf.keras.metrics.Accuracy`: Computes the accuracy between the predictions and the true labels.
2. `tf.keras.metrics.Precision`: Computes the precision between the predictions and the true labels.
3. `tf.keras.metrics.Recall`: Computes the recall between the predictions and the true labels.
4. `tf.keras.metrics.F1Score`: Computes the F1 score between the predictions and the true labels.
5. `tf.keras.metrics.AUC`: Computes the area under the receiver operating characteristic (ROC) curve.
6. `tf.keras.metrics.MeanSquaredError`: Computes the mean squared error between the predictions and the true labels.
7. `tf.keras.metrics.MeanAbsoluteError`: Computes the mean absolute error between the predictions and the true labels.
8. `tf.keras.metrics.BinaryCrossentropy`: Computes the binary cross-entropy between the predictions and the true labels.
9. `tf.keras.metrics.CategoricalCrossentropy`: Computes the categorical cross-entropy between the predictions and the true labels.
10. `tf.keras.metrics.ConfusionMatrix`: Computes the confusion matrix between the predictions and the true labels.

KUBEFLOW Pipeline Functions:

Google Cloud

Here are 10 important functions in Kubeflow Pipelines, managing end-to-end ML workflows on Kubernetes:

1. `kfp.components`: This function is used to create reusable components in a Kubeflow pipeline. Components are self-contained ML operations that can be executed as part of a pipeline.
2. `kfp.dsl`: This function provides a domain-specific language (DSL) for creating pipelines in Kubeflow. The DSL makes it easy to create pipelines and perform common ML operations.
3. `kfp.Client`: This function creates a client that can be used to interact with a Kubeflow pipeline deployment.
4. `kfp.Client().create_run_from_pipeline_func`: This function creates a pipeline run from a pipeline function defined using the DSL.
5. `kfp.Client().run_pipeline`: This function runs a pipeline on a Kubeflow deployment.
6. `kfp.Client().list_runs`: This function lists the runs of a pipeline, including the run ID, status, and start time.
7. `kfp.Client().wait_for_run_completion`: This function waits for a pipeline run to complete.
8. `kfp.Client().download_pipeline`: This function downloads a pipeline definition as a YAML file.
9. `kfp.Client().upload_pipeline`: This function uploads a pipeline definition from a YAML file.
10. `kfp.Client().delete_run`: This function deletes a pipeline run.