sagemaker

Release 1.2.4

Jun 05, 2018

Contents

Amazon SageMaker Python SDK is an open source library for training and deploying machine-learned models on Amazon SageMaker.

With the SDK, you can train and deploy models using popular deep learning frameworks: **Apache MXNet** and **TensorFlow**. You can also train and deploy models with **algorithms provided by Amazon**, these are scalable implementations of core machine learning algorithms that are optimized for SageMaker and GPU training. If you have **your own algorithms** built into SageMaker-compatible Docker containers, you can train and host models using these as well.

Here you'll find API docs for SageMaker Python SDK. The project home-page is in Github: https://github.com/aws/sagemaker-python-sdk, there you can find the SDK source, installation instructions and a general overview of the library there.

CHAPTER 1

Overview

The SageMaker Python SDK consists of a few primary interfaces:

1.1 Estimators

A high level interface for SageMaker training

class sagemaker.estimator.**Estimator**(*image_name*, *role*, *train_instance_count*, *train_instance_type*, *train_volume_size=30*, *train_max_run=86400*, *input_mode='File'*, *output_path=None*, *output_kms_key=None*, *base_job_name=None*, *sagemaker_session=None*, *hyperparameters=None*)

Bases: `sagemaker.estimator.EstimatorBase`

A generic Estimator to train using any supplied algorithm. This class is designed for use with algorithms that don't have their own, custom class.

Initialize an `Estimator` instance.

Parameters

- **image_name** (*str*) – The container image to use for training.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.

- **train_volume_size** (*int*) – Size in GB of the EBS volume to use for storing input data during training (default: 30). Must be large enough to store training data if File Mode is used (which is the default).
- **train_max_run** (*int*) – Timeout in seconds for training (default: 24 * 60 * 60). After this amount of time Amazon SageMaker terminates the job regardless of its current status.
- **input_mode** (*str*) – The input mode that the algorithm supports (default: 'File'). Valid modes:
  - 'File' - Amazon SageMaker copies the training dataset from the S3 location to a local directory.
  - 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe.
- **output_path** (*str*) – S3 location for saving the trainig result (model artifacts and output files). If not specified, results are stored to a default bucket. If the bucket with the specific name does not exist, the estimator creates the bucket during the `fit()` method execution.
- **output_kms_key** (*str*) – Optional. KMS key ID for encrypting the training output (default: None).
- **base_job_name** (*str*) – Prefix for training job name when the `fit()` method launches. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.
- **sagemaker_session** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.
- **hyperparameters** (*dict*) – Dictionary containing the hyperparameters to initialize this estimator with.

train_image()

Returns the docker image to use for training.

The fit() method, that does the model training, calls this method to find the image to use for model training.

set_hyperparameters(***kwargs*)

hyperparameters()

Returns the hyperparameters as a dictionary to use for training.

The fit() method, that does the model training, calls this method to find the hyperparameters you specified.

create_model(*image=None*, *predictor_cls=None*, *serializer=None*, *deserializer=None*, *content_type=None*, *accept=None*, ***kwargs*)

Create a model to deploy.

Parameters

- **image** (*str*) – An container image to use for deploying the model. Defaults to the image used for training.
- **predictor_cls** (`RealTimePredictor`) – The predictor class to use when deploying the model.
- **serializer** (*callable*) – Should accept a single argument, the input data, and return a sequence of bytes. May provide a content_type attribute that defines the endpoint request content type

- **deserializer** (*callable*) – Should accept two arguments, the result data and the response content type, and return a sequence of bytes. May provide a content_type attribute that defines th endpoint response Accept content type.
- **content_type** (*str*) – The invocation ContentType, overriding any content_type from the serializer
- **accept** (*str*) – The invocation Accept, overriding any accept from the deserializer.
- **serializer, deserializer, content_type, and accept arguments are only used to define a default** (*The*) –
- **They are ignored if an explicit predictor class is passed in. Other arguments** (*RealTimePredictor.*) –
- **passed through to the Model class.** (*are*) –

Returns: a Model ready for deployment.

classmethod attach (*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **training_job_name** (*str*) – The name of the training job to attach to.
- **sagemaker_session** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

delete_endpoint ()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy (*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **initial_instance_count** (*int*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- ****kwargs** – Passed to invocation of `create_model()`. Implementations may customize `create_model()` to accept `**kwargs` to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a `predict()` method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit (*inputs*, *wait=True*, *logs=True*, *job_name=None*)

Train a model using the input training dataset.

The API calls the Amazon SageMaker CreateTrainingJob API to start model training. The API uses configuration you provided to create the estimator and the specified input training data to send the CreatingTrainingJob request to Amazon SageMaker.

This is a synchronous operation. After the model training successfully completes, you can call the `deploy()` method to host the model using the Amazon SageMaker hosting services.

Parameters

- **inputs** (*str or dict or sagemaker.session.s3_input*) – Information about the training data. This can be one of three types:
  - (str) the S3 location where training data is saved.
  - **(dict[str, str] or dict[str, sagemaker.session.s3_input]) If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - channel configuration for S3 data sources that can provide** additional information about the training dataset. See *sagemaker.session.s3_input()* for full details.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

model_data

str – The model location in S3. Only set if Estimator has been `fit()`.

training_job_analytics

Returns a TrainingJobAnalytics object for the current training job.

1.2 Predictors

Make real-time predictions against SageMaker endpoints with Python objects

class sagemaker.predictor.**RealTimePredictor**(*endpoint*, *sagemaker_session=None*, *serializer=None*, *deserializer=None*, *content_type=None*, *accept=None*)

Bases: `object`

Make prediction requests to an Amazon SageMaker endpoint.

Initialize a `RealTimePredictor`.

Behavior for serialization of input data and deserialization of result data can be configured through initializer arguments. If not specified, a sequence of bytes is expected and the API sends it in the request body without modifications. In response, the API returns the sequence of bytes from the prediction result without any modifications.

Parameters

- **endpoint** (*str*) – Name of the Amazon SageMaker endpoint to which requests are sent.
- **sagemaker_session** (`sagemaker.session.Session`) – A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.
- **serializer** (*callable*) – Accepts a single argument, the input data, and returns a sequence of bytes. It may provide a `content_type` attribute that defines the endpoint request content type. If not specified, a sequence of bytes is expected for the data.
- **deserializer** (*callable*) – Accepts two arguments, the result data and the response content type, and returns a sequence of bytes. It may provide a `content_type` attribute that defines the endpoint response's "Accept" content type. If not specified, a sequence of bytes is expected for the data.
- **content_type** (*str*) – The invocation's "ContentType", overriding any `content_type` from the serializer (default: None).
- **accept** (*str*) – The invocation's "Accept", overriding any accept from the deserializer (default: None).

predict(*data*)

Return the inference from the specified endpoint.

Parameters **data** (*object*) – Input data for which you want the model to provide inference. If a serializer was specified when creating the RealTimePredictor, the result of the serializer is sent as input data. Otherwise the data must be sequence of bytes, and the predict method then sends the bytes in the request body as is.

Returns

Inference for the given input. If a deserializer was specified when creating the RealTimePredictor, the result of the deserializer is returned. Otherwise the response returns the sequence of bytes as is.

Return type object

1.3 Session

class sagemaker.session.**LogState**

Bases: `object`

STARTING = 1

WAIT_IN_PROGRESS = 2

TAILING = 3

JOB_COMPLETE = 4

COMPLETE = 5

class sagemaker.session.**Session**(*boto_session=None*, *sagemaker_client=None*, *sagemaker_runtime_client=None*)

Bases: `object`

Manage interactions with the Amazon SageMaker APIs and any other AWS services needed.

This class provides convenient methods for manipulating entities and resources that Amazon SageMaker uses, such as training jobs, endpoints, and input datasets in S3.

AWS service calls are delegated to an underlying Boto3 session, which by default is initialized using the AWS configuration chain. When you make an Amazon SageMaker API call that accesses an S3 bucket location and one is not specified, the `Session` creates a default bucket based on a naming convention which includes the current AWS account ID.

Initialize a SageMaker `Session`.

Parameters

- **boto_session** (*boto3.session.Session*) – The underlying Boto3 session which AWS service calls are delegated to (default: None). If not provided, one is created with default AWS configuration chain.
- **sagemaker_client** (*boto3.SageMaker.Client*) – Client which makes Amazon SageMaker service calls other than `InvokeEndpoint` (default: None). Estimators created using this `Session` use this client. If not provided, one will be created using this instance's `boto_session`.
- **sagemaker_runtime_client** (*boto3.SageMakerRuntime.Client*) – Client which makes `InvokeEndpoint` calls to Amazon SageMaker (default: None). Predictors created using this `Session` use this client. If not provided, one will be created using this instance's `boto_session`.

boto_region_name

upload_data(*path*, *bucket=None*, *key_prefix='data'*)

Upload local file or directory to S3.

If a single file is specified for upload, the resulting S3 object key is `{key_prefix}/{filename}` (filename does not include the local path, if any specified).

If a directory is specified for upload, the API uploads all content, recursively, preserving relative structure of subdirectories. The resulting object key names are: `{key_prefix}/{relative_subdirectory_path}/filename`.

Parameters

- **path** (*str*) – Path (absolute or relative) of local file or directory to upload.
- **bucket** (*str*) – Name of the S3 Bucket to upload to (default: None). If not specified, the default bucket of the `Session` is used. If the bucket does not exist, the `Session` creates the bucket.
- **key_prefix** (*str*) – Optional S3 object key name prefix (default: 'data'). S3 uses the prefix to create a directory structure for the bucket content that it display in the S3 console.

Returns

The S3 URI of the uploaded file(s). If a file is specified in the path argument, the URI format is: `s3://{bucket name}/{key_prefix}/{original_file_name}`. If a directory is specified in the path argument, the URI format is `s3://{bucket name}/{key_prefix}`.

Return type str

default_bucket()

Return the name of the default bucket to use in relevant Amazon SageMaker interactions.

Returns The name of the default bucket, which is of the form: `sagemaker-{region}-{AWS account ID}`.

Return type str

train(*image*, *input_mode*, *input_config*, *role*, *job_name*, *output_config*, *resource_config*, *hyperparameters*, *stop_condition*, *tags*)

Create an Amazon SageMaker training job.

Parameters

- **image** (*str*) – Docker image containing training code.
- **input_mode** (*str*) – The input mode that the algorithm supports. Valid modes:
  - **'File' - Amazon SageMaker copies the training dataset from the S3 location to** a directory in the Docker container.
  - 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe.
- **input_config** (*list*) – A list of Channel objects. Each channel is a named input source. Please refer to the format details described: https://botocore.readthedocs.io/en/latest/reference/services/sagemaker.html#SageMaker.Client.create_training_job
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. You must grant sufficient permissions to this role.
- **job_name** (*str*) – Name of the training job being created.
- **output_config** (*dict*) – The S3 URI where you want to store the training results and optional KMS key ID.
- **resource_config** (*dict*) – Contains values for ResourceConfig: * instance_count (int): Number of EC2 instances to use for training.

  The key in resource_config is 'InstanceCount'.

  - **instance_type (str): Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.** The key in resource_config is 'InstanceType'.
- **hyperparameters** (*dict*) – Hyperparameters for model training. The hyperparameters are made accessible as a dict[str, str] to the training code on SageMaker. For convenience, this accepts other types for keys and values, but `str()` will be called to convert them before training.
- **stop_condition** (*dict*) – Defines when training shall finish. Contains entries that can be understood by the service like `MaxRuntimeInSeconds`.
- **tags** (*list[dict]*) – List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

Returns ARN of the training job, if it is created.

Return type str

tune (*job_name*, *strategy*, *objective_type*, *objective_metric_name*, *max_jobs*, *max_parallel_jobs*, *parameter_ranges*, *static_hyperparameters*, *image*, *input_mode*, *metric_definitions*, *role*, *input_config*, *output_config*, *resource_config*, *stop_condition*, *tags*)

Create an Amazon SageMaker hyperparameter tuning job

Parameters

- **job_name** (*str*) – Name of the tuning job being created.
- **strategy** (*str*) – Strategy to be used.
- **objective_type** (*str*) – Minimize/Maximize
- **objective_metric_name** (*str*) – Name of the metric to use when evaluating training job.
- **max_jobs** (*int*) – Maximum total number of jobs to start.
- **max_parallel_jobs** (*int*) – Maximum number of parallel jobs to start.
- **parameter_ranges** (*dict*) – Parameter ranges in a dictionary of types: Continuous, Integer, Categorical
- **static_hyperparameters** (*dict*) – Hyperparameters for model training. The hyperparameters are made accessible as a dict[str, str] to the training code on SageMaker. For convenience, this accepts other types for keys and values, but `str()` will be called to convert them before training.
- **image** (*str*) – Docker image containing training code.
- **input_mode** (*str*) – The input mode that the algorithm supports. Valid modes:
  - **'File' - Amazon SageMaker copies the training dataset from the S3 location to** a directory in the Docker container.
  - 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe.
- **metric_definitions** (*list[dict]*) – Metrics definition with 'name' and 'regex' keys.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. You must grant sufficient permissions to this role.
- **input_config** (*list*) – A list of Channel objects. Each channel is a named input source. Please refer to the format details described: https://botocore.readthedocs.io/en/latest/reference/services/sagemaker.html#SageMaker.Client.create_training_job
- **output_config** (*dict*) – The S3 URI where you want to store the training results and optional KMS key ID.
- **resource_config** (*dict*) – Contains values for ResourceConfig:
- **instance_count** (*int*) – Number of EC2 instances to use for training.
- **instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **stop_condition** (*dict*) – Defines when training shall finish. Contains entries that can be understood by the service like `MaxRuntimeInSeconds`.

- **tags** (*list[dict]*) – List of tags for labeling the tuning job.

stop_tuning_job(*name*)

Attempts to stop tuning job on Amazon SageMaker with specified name.

Parameters **name** – Name of Amazon SageMaker tuning job.

create_model(*name*, *role*, *primary_container*)

Create an Amazon SageMaker `Model`.

Specify the S3 location of the model artifacts and Docker image containing the inference code. Amazon SageMaker uses this information to deploy the model in Amazon SageMaker.

Parameters

- **name** (*str*) – Name of the Amazon SageMaker `Model` to create.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. You must grant sufficient permissions to this role.
- **primary_container** (*str or dict[str, str]*) – Docker image which defines the inference code. You can also specify the return value of `sagemaker.container_def()`, which is used to create more advanced container configurations, including model containers which need artifacts from S3.

Returns Name of the Amazon SageMaker `Model` created.

Return type str

create_model_from_job(*training_job_name*, *name=None*, *role=None*, *primary_container_image=None*, *model_data_url=None*, *env={}*)

Create an Amazon SageMaker `Model` from a SageMaker Training Job.

Parameters

- **training_job_name** (*str*) – The Amazon SageMaker Training Job name.
- **name** (*str*) – The name of the SageMaker `Model` to create (default: None). If not specified, the training job name is used.
- **role** (*str*) – The `ExecutionRoleArn` IAM Role ARN for the `Model`, specified either by an IAM role name or role ARN. If None, the `RoleArn` from the SageMaker Training Job will be used.
- **primary_container_image** (*str*) – The Docker image reference (default: None). If None, it defaults to the Training Image in `training_job_name`.
- **model_data_url** (*str*) – S3 location of the model data (default: None). If None, defaults to the `ModelS3Artifacts` of `training_job_name`.
- **env** (*dict[string, string]*) – Model environment variables (default: {}).

Returns The name of the created `Model`.

Return type str

create_endpoint_config(*name*, *model_name*, *initial_instance_count*, *instance_type*)

Create an Amazon SageMaker endpoint configuration.

The endpoint configuration identifies the Amazon SageMaker model (created using the `CreateModel` API) and the hardware configuration on which to deploy the model. Provide this endpoint configuration to the `CreateEndpoint` API, which then launches the hardware and deploys the model.

Parameters

- **name** (*str*) – Name of the Amazon SageMaker endpoint configuration to create.
- **model_name** (*str*) – Name of the Amazon SageMaker `Model`.
- **initial_instance_count** (*int*) – Minimum number of EC2 instances to launch. The actual number of active instances for an endpoint at any given time varies due to autoscaling.
- **instance_type** (*str*) – Type of EC2 instance to launch, for example, 'ml.c4.xlarge'.

Returns Name of the endpoint point configuration created.

Return type str

create_endpoint(*endpoint_name*, *config_name*, *wait=True*)

Create an Amazon SageMaker `Endpoint` according to the endpoint configuration specified in the request.

Once the `Endpoint` is created, client applications can send requests to obtain inferences. The endpoint configuration is created using the `CreateEndpointConfig` API.

Parameters

- **endpoint_name** (*str*) – Name of the Amazon SageMaker `Endpoint` being created.
- **config_name** (*str*) – Name of the Amazon SageMaker endpoint configuration to deploy.
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).

Returns Name of the Amazon SageMaker `Endpoint` created.

Return type str

delete_endpoint(*endpoint_name*)

Delete an Amazon SageMaker `Endpoint`.

Parameters **endpoint_name** (*str*) – Name of the Amazon SageMaker `Endpoint` to delete.

wait_for_job(*job*, *poll=5*)

Wait for an Amazon SageMaker training job to complete.

Parameters

- **job** (*str*) – Name of the training job to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeTrainingJob` API.

Return type (dict)

Raises `ValueError` – If the training job fails.

wait_for_tuning_job(*job*, *poll=5*)

Wait for an Amazon SageMaker tuning job to complete.

Parameters

- **job** (*str*) – Name of the tuning job to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeHyperParameterTuningJob` API.

Return type (dict)

Raises `ValueError` – If the hyperparameter tuning job fails.

`wait_for_endpoint`(*endpoint*, *poll=5*)

Wait for an Amazon SageMaker endpoint deployment to complete.

Parameters

- **`endpoint`** (*`str`*) – Name of the `Endpoint` to wait for.
- **`poll`** (*`int`*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeEndpoint` API.

Return type dict

`endpoint_from_job`(*job_name*, *initial_instance_count*, *instance_type*, *deployment_image=None*, *name=None*, *role=None*, *wait=True*, *model_environment_vars=None*)

Create an `Endpoint` using the results of a successful training job.

Specify the job name, Docker image containing the inference code, and hardware configuration to deploy the model. Internally the API, creates an Amazon SageMaker model (that describes the model artifacts and the Docker image containing inference code), endpoint configuration (describing the hardware to deploy for hosting the model), and creates an `Endpoint` (launches the EC2 instances and deploys the model on them). In response, the API returns the endpoint name to which you can send requests for inferences.

Parameters

- **`job_name`** (*`str`*) – Name of the training job to deploy the results of.
- **`initial_instance_count`** (*`int`*) – Minimum number of EC2 instances to launch. The actual number of active instances for an endpoint at any given time varies due to autoscaling.
- **`instance_type`** (*`str`*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **`deployment_image`** (*`str`*) – The Docker image which defines the inference code to be used as the entry point for accepting prediction requests. If not specified, uses the image used for the training job.
- **`name`** (*`str`*) – Name of the `Endpoint` to create. If not specified, uses the training job name.
- **`role`** (*`str`*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. You must grant sufficient permissions to this role.
- **`wait`** (*`bool`*) – Whether to wait for the endpoint deployment to complete before returning (default: True).
- **`model_environment_vars`** (*`dict[str, str]`*) – Environment variables to set on the model container (default: None).

Returns Name of the `Endpoint` that is created.

Return type str

endpoint_from_model_data(*model_s3_location*, *deployment_image*, *initial_instance_count*, *instance_type*, *name=None*, *role=None*, *wait=True*, *model_environment_vars=None*)

Create and deploy to an `Endpoint` using existing model data stored in S3.

Parameters

- **model_s3_location** (*str*) – S3 URI of the model artifacts to use for the endpoint.
- **deployment_image** (*str*) – The Docker image which defines the runtime code to be used as the entry point for accepting prediction requests.
- **initial_instance_count** (*int*) – Minimum number of EC2 instances to launch. The actual number of active instances for an endpoint at any given time varies due to autoscaling.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, e.g. 'ml.c4.xlarge'.
- **name** (*str*) – Name of the `Endpoint` to create. If not specified, uses a name generated by combining the image name with a timestamp.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. You must grant sufficient permissions to this role.
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).
- **model_environment_vars** (*dict[str, str]*) – Environment variables to set on the model container (default: None).

Returns Name of the `Endpoint` that is created.

Return type str

endpoint_from_production_variants(*name*, *production_variants*, *wait=True*)

Create an SageMaker `Endpoint` from a list of production variants.

Parameters

- **name** (*str*) – The name of the `Endpoint` to create.
- **production_variants** (*list[dict[str, str]]*) – The list of production variants to deploy.
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).

Returns The name of the created `Endpoint`.

Return type str

expand_role(*role*)

Expand an IAM role name into an ARN.

If the role is already in the form of an ARN, then the role is simply returned. Otherwise we retrieve the full ARN and return it.

Parameters **role** (*str*) – An AWS IAM role (either name or full ARN).

Returns The corresponding AWS IAM role ARN.

Return type str

get_caller_identity_arn()

Returns the ARN user or role whose credentials are used to call the API. :returns: The ARN uer or role :rtype: (str)

logs_for_job(*job_name*, *wait=False*, *poll=10*)

Display the logs for a given training job, optionally tailing them until the job is complete. If the output is a tty or a Jupyter cell, it will be color-coded based on which instance the log entry is from.

Parameters

- **job_name** (*str*) – Name of the training job to display the logs for.
- **wait** (*bool*) – Whether to keep looking for new log entries until the job completes (default: False).
- **poll** (*int*) – The interval in seconds between polling for new log entries and job completion (default: 5).

Raises `ValueError` – If waiting and the training job fails.

sagemaker.session.**container_def**(*image*, *model_data_url=None*, *env=None*)

Create a definition for executing a container as part of a SageMaker model.

Parameters

- **image** (*str*) – Docker image to run for this container.
- **model_data_url** (*str*) – S3 URI of data required by this container, e.g. SageMaker training job model artifacts (default: None).
- **env** (*dict[str, str]*) – Environment variables to set inside the container (default: None).

Returns A complete container definition object usable with the CreateModel API.

Return type dict[str, str]

sagemaker.session.**production_variant**(*model_name*, *instance_type*, *initial_instance_count=1*, *variant_name='AllTraffic'*, *initial_weight=1*)

Create a production variant description suitable for use in a `ProductionVariant` list as part of a `CreateEndpointConfig` request.

Parameters

- **model_name** (*str*) – The name of the SageMaker model this production variant references.
- **instance_type** (*str*) – The EC2 instance type for this production variant. For example, 'ml.c4.8xlarge'.
- **initial_instance_count** (*int*) – The initial instance count for this production variant (default: 1).
- **variant_name** (*string*) – The `VariantName` of this production variant (default: 'AllTraffic').
- **initial_weight** (*int*) – The relative `InitialVariantWeight` of this production variant (default: 1).

Returns An SageMaker `ProductionVariant` description

Return type dict[str, str]

sagemaker.session.**get_execution_role**(*sagemaker_session=None*)

Return the role ARN whose credentials are used to call the API. Throws an exception if :param sagemaker_session: Current sagemaker session :type sagemaker_session: Session

Returns The role ARN

Return type (str)

class sagemaker.session.**s3_input**(*s3_data*, *distribution='FullyReplicated'*, *compression=None*, *content_type=None*, *record_wrapping=None*, *s3_data_type='S3Prefix'*)

Bases: `object`

Amazon SageMaker channel configurations for S3 data sources.

config

dict[str, dict] – A SageMaker `DataSource` referencing a SageMaker `S3DataSource`.

Create a definition for input data used by an SageMaker training job.

See AWS documentation on the `CreateTrainingJob` API for more details on the parameters.

Parameters

- **s3_data** (*str*) – Defines the location of s3 data to train on.
- **distribution** (*str*) – Valid values: 'FullyReplicated', 'ShardedByS3Key' (default: 'FullyReplicated').
- **compression** (*str*) – Valid values: 'Gzip', 'Bzip2', 'Lzop' (default: None).
- **content_type** (*str*) – MIME type of the input data (default: None).
- **record_wrapping** (*str*) – Valid values: 'RecordIO' (default: None).
- **s3_data_type** (*str*) – Value values: 'S3Prefix', 'ManifestFile'. If 'S3Prefix', `s3_data` defines a prefix of s3 objects to train on. All objects with s3 keys beginning with `s3_data` will be used to train. If 'ManifestFile', then `s3_data` defines a single s3 manifest file, listing each s3 object to train on. The Manifest file format is described in the SageMaker API documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/API_S3DataSource.html

1.4 Model

class sagemaker.model.**Model**(*model_data*, *image*, *role*, *predictor_cls=None*, *env=None*, *name=None*, *sagemaker_session=None*)

Bases: `object`

An SageMaker `Model` that can be deployed to an `Endpoint`.

Initialize an SageMaker `Model`.

Parameters

- **model_data** (*str*) – The S3 location of a SageMaker model data `.tar.gz` file.
- **image** (*str*) – A Docker image URI.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.
- **predictor_cls** (*callable[string, sagemaker.session.Session]*) – A function to call to create a predictor (default: None). If not None, `deploy` will return the result of invoking this function on the created endpoint name.

- **`env`** (*`dict[str, str]`*) – Environment variables to run with `image` when hosted in SageMaker (default: None).
- **`name`** (*`str`*) – The model name. If None, a default model name will be selected on each `deploy`.
- **`sagemaker_session`** (`sagemaker.session.Session`) – A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

`prepare_container_def`(*instance_type*)

Return a dict created by `sagemaker.container_def()` for deploying this model to a specified instance type.

Subclasses can override this to provide custom container definitions for deployment to a specific instance type. Called by `deploy()`.

Parameters **`instance_type`** (*`str`*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

Returns A container definition object usable with the CreateModel API.

Return type dict

`deploy`(*initial_instance_count*, *instance_type*, *endpoint_name=None*)

Deploy this `Model` to an `Endpoint` and optionally return a `Predictor`.

Create a SageMaker `Model` and `EndpointConfig`, and deploy an `Endpoint` from this `Model`. If `self.predictor_cls` is not None, this method returns a the result of invoking `self.predictor_cls` on the created endpoint name.

The name of the created endpoint is accessible in the `endpoint_name` field of this `Model` after deploy returns.

Parameters

- **`instance_type`** (*`str`*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.
- **`initial_instance_count`** (*`int`*) – The initial number of instances to run in the `Endpoint` created from this `Model`.
- **`endpoint_name`** (*`str`*) – The name of the endpoint to create (default: None). If not specified, a unique endpoint name will be created.

Returns

Invocation of `self.predictor_cls` on the created endpoint name, if `self.predictor_cls` is not None. Otherwise, return None.

Return type callable[string, *sagemaker.session.Session*] or None

CHAPTER 2

MXNet

A managed environment for MXNet training and hosting on Amazon SageMaker

2.1 MXNet

2.1.1 MXNet Estimator

class sagemaker.mxnet.estimator.**MXNet**(*entry_point*, *source_dir=None*, *hyperparameters=None*, *py_version='py2'*, *framework_version='1.1'*, ***kwargs*)

Bases: `sagemaker.estimator.Framework`

Handle end-to-end training and deployment of custom MXNet code.

This `Estimator` executes an MXNet script in a managed MXNet execution environment, within a SageMaker Training Job. The managed MXNet environment is an Amazon-built Docker container that executes functions defined in the supplied `entry_point` Python script.

Training is started by calling `fit()` on this Estimator. After training is complete, calling `deploy()` creates a hosted SageMaker endpoint and returns an `MXNetPredictor` instance that can be used to perform inference against the hosted model.

Technical documentation on preparing MXNet scripts for SageMaker training and using the MXNet Estimator is avaialble on the project home-page: https://github.com/aws/sagemaker-python-sdk

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to training. This should be compatible with either Python 2.7 or Python 3.5.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: None). Structure within this directory are preserved when training on Amazon SageMaker.

- **hyperparameters** (*dict*) – Hyperparameters that will be used for training (default: None). The hyperparameters are made accessible as a dict[str, str] to the training code on SageMaker. For convenience, this accepts other types for keys and values, but `str()` will be called to convert them before training.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2'). One of 'py2' or 'py3'.
- **framework_version** (*str*) – MXNet version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#mxnet-sagemaker-estimators
- ****kwargs** – Additional kwargs passed to the `Framework` constructor.

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

create_model(*model_server_workers=None*)

Create a SageMaker `MXNetModel` object that can be deployed to an `Endpoint`.

Parameters model_server_workers (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.

Returns

A SageMaker `MXNetModel` object. See *MXNetModel()* for full details.

Return type *sagemaker.mxnet.model.MXNetModel*

2.1.2 MXNet Model

class sagemaker.mxnet.model.**MXNetModel**(*model_data, role, entry_point, image=None, py_version='py2', framework_version='1.1', predictor_cls=<class 'sagemaker.mxnet.model.MXNetPredictor'>, model_server_workers=None, **kwargs*)

Bases: `sagemaker.model.FrameworkModel`

An MXNet SageMaker `Model` that can be deployed to a SageMaker `Endpoint`.

Initialize an MXNetModel.

Parameters

- **model_data** (*str*) – The S3 location of a SageMaker model data `.tar.gz` file.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.
- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to model hosting. This should be compatible with either Python 2.7 or Python 3.5.

- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for MXNet will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – MXNet version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*)

Return a container definition with framework configuration set in model environment variables.

Parameters instance_type (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

Returns A container definition object usable with the CreateModel API.

Return type dict[str, str]

2.1.3 MXNet Predictor

class sagemaker.mxnet.model.**MXNetPredictor**(*endpoint_name*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A RealTimePredictor for inference against MXNet Endpoints.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for MXNet inference.

Initialize an `MXNetPredictor`.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.
- **sagemaker_session** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

CHAPTER 3

TensorFlow

A managed environment for TensorFlow training and hosting on Amazon SageMaker

3.1 TensorFlow

3.1.1 TensorFlow Estimator

class sagemaker.tensorflow.estimator.**TensorFlow**(*training_steps=None*, *evaluation_steps=None*, *checkpoint_path=None*, *py_version='py2'*, *framework_version='1.6'*, *requirements_file="*, ***kwargs*)

Bases: sagemaker.estimator.Framework

Handle end-to-end training and deployment of user-provided TensorFlow code.

Initialize an TensorFlow estimator. :param training_steps: Perform this many steps of training. *None*, the default means train forever. :type training_steps: int :param evaluation_steps: Perform this many steps of evaluation. *None*, the default means that evaluation

> runs until input from eval_input_fn is exhausted (or another exception is raised).

Parameters

- **checkpoint_path** (*str*) – Identifies S3 location where checkpoint data during model training can be saved (default: None). For distributed model training, this parameter is required.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – TensorFlow version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#tensorflow-sagemaker-estimators

- **requirements_file** (*str*) – Path to a `requirements.txt` file (default: ‘’). The path should be within and relative to `source_dir`. Details on the format can be found in the Pip User Guide.
- ****kwargs** – Additional kwargs passed to the Framework constructor.

fit(*inputs*, *wait=True*, *logs=True*, *job_name=None*, *run_tensorboard_locally=False*)

Train a model using the input training dataset.

See `fit()` for more details.

Parameters

- **inputs** (*str or dict or sagemaker.session.s3_input*) – Information about the training data. This can be one of three types: (str) - the S3 location where training data is saved. (dict[str, str] or dict[str, sagemaker.session.s3_input]) - If using multiple channels for

  training data, you can specify a dict mapping channel names to strings or `s3_input()` objects.

  (sagemaker.session.s3_input) - channel configuration for S3 data sources that can provide
  additional information about the training dataset. See `sagemaker.session.s3_input()` for full details.

- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.
- **run_tensorboard_locally** (*bool*) – Whether to execute TensorBoard in a different process with downloaded checkpoint information (default: False). This is an experimental feature, and requires TensorBoard and AWS CLI to be installed. It terminates TensorBoard when execution ends.

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

create_model(*model_server_workers=None*)

Create a SageMaker `TensorFlowModel` object that can be deployed to an `Endpoint`.

Parameters **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.

Returns

A SageMaker TensorFlowModel object. See `TensorFlowModel()` for full details.

Return type *sagemaker.tensorflow.model.TensorFlowModel*

hyperparameters()

Return hyperparameters used by your custom TensorFlow code during model training.

3.1.2 TensorFlow Model

class sagemaker.tensorflow.model.**TensorFlowModel**(*model_data, role, entry_point, image=None, py_version='py2', framework_version='1.6', predictor_cls=<class 'sagemaker.tensorflow.model.TensorFlowPredictor'>, model_server_workers=None, **kwargs*)

Bases: `sagemaker.model.FrameworkModel`

Initialize an TensorFlowModel.

Parameters

- **model_data** (*str*) – The S3 location of a SageMaker model data `.tar.gz` file.
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.
- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to model hosting. This should be compatible with either Python 2.7 or Python 3.5.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for TensorFlow will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – TensorFlow version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*)

Return a container definition with framework configuration set in model environment variables.

This also uploads user-supplied code to S3.

Parameters **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

Returns A container definition object usable with the CreateModel API.

Return type dict[str, str]

3.1.3 TensorFlow Predictor

class sagemaker.tensorflow.model.**TensorFlowPredictor**(*endpoint_name, sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A `RealTimePredictor` for inference against TensorFlow ``Endpoint``s.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for MXNet inference

Initialize an `TensorFlowPredictor`.

Parameters

- **`endpoint_name`** (*`str`*) – The name of the endpoint to perform inference on.
- **`sagemaker_session`** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

CHAPTER 4

SageMaker First-Party Algorithms

Amazon provides implementations of some common machine learning algortithms optimized for GPU architecture and massive datasets.

4.1 K-means

The Amazon SageMaker K-means algorithm.

class sagemaker.**KMeans**(*role*, *train_instance_count*, *train_instance_type*, *k*, *init_method=None*, *max_iterations=None*, *tol=None*, *num_trials=None*, *local_init_method=None*, *half_life_time_size=None*, *epochs=None*, *center_factor=None*, *eval_metrics=None*, ***kwargs*)

Bases: `sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase`

A k-means clustering `AmazonAlgorithmEstimatorBase`. Finds k clusters of data in an unlabeled dataset.

This Estimator may be fit via calls to `fit_ndarray()` or `fit()`. The former allows a KMeans model to be fit on a 2-dimensional numpy array. The latter requires Amazon `Record` protobuf serialized data to be stored in S3.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://alpha-docs-aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, `deploy` returns a `KMeansPredictor` object that can be used to k-means cluster assignments, using the trained k-means model hosted in the SageMaker Endpoint.

KMeans Estimators can be configured by setting hyperparameters. The available hyperparameters for KMeans are documented below. For further information on the AWS KMeans algorithm, please consult AWS technical documentation: https://alpha-docs-aws.amazon.com/sagemaker/latest/dg/k-means.html

Parameters

- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if accessing AWS resource. For more information, see <link>???.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **k** (*int*) – The number of clusters to produce.
- **init_method** (*str*) – How to initialize cluster locations. One of 'random' or 'kmeans++'.
- **max_iterations** (*int*) – Maximum iterations for Lloyds EM procedure in the local kmeans used in finalize stage.
- **tol** (*float*) – Tolerance for change in ssd for early stopping in local kmeans.
- **num_trials** (*int*) – Local version is run multiple times and the one with the best loss is chosen. This determines how many times.
- **local_init_method** (*str*) – Initialization method for local version. One of 'random', 'kmeans++'
- **half_life_time_size** (*int*) – The points can have a decayed weight. When a point is observed its weight, with regard to the computation of the cluster mean is 1. This weight will decay exponentially as we observe more points. The exponent coefficient is chosen such that after observing `half_life_time_size` points after the mentioned point, its weight will become 1/2. If set to 0, there will be no decay.
- **epochs** (*int*) – Number of passes done over the training data.
- **center_factor** (*int*) – The algorithm will create `num_clusters * extra_center_factor` as it runs and reduce the number of centers to `k` when finalizing
- **eval_metrics** (*list*) – JSON list of metrics types to be used for reporting the score for the model. Allowed values are "msd" Means Square Error, "ssd": Sum of square distance. If test data is provided, the score shall be reported in terms of all requested metrics.
- ****kwargs** – base class keyword argument values.

repo_name = 'kmeans'

repo_version = 1

classmethod attach(*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **training_job_name** (*str*) – The name of the training job to attach to.
- **sagemaker_session** (sagemaker.session.Session) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **initial_instance_count** (*int*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- ****kwargs** – Passed to invocation of `create_model()`. Implementations may customize `create_model()` to accept `**kwargs` to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a predict() method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `RecordSet`. This defines a collection of S3 data files to train this `Estimator` on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **`records`** (`RecordSet`) – The records to train this `Estimator` on
- **`mini_batch_size`** (*`int or None`*) – The size of each mini-batch to use when training. If `None`, a default value will be used.
- **`wait`** (*`bool`*) – Whether the call should wait until the job completes (default: True).
- **`logs`** (*`bool`*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **`job_name`** (*`str`*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

`model_data`

str – The model location in S3. Only set if Estimator has been `fit()`.

`record_set`(*train*, *labels=None*, *channel='train'*)

Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **`train`** (*`numpy.ndarray`*) – A 2D numpy array of training data.
- **`labels`** (*`numpy.ndarray`*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **`channel`** (*`str`*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

`train_image`()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

`training_job_analytics`

Returns a TrainingJobAnalytics object for the current training job.

`eval_metrics`

An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

create_model()

Return a `KMeansModel` referencing the latest s3 model data produced by this Estimator.

hyperparameters()

Return the SageMaker hyperparameters for training this KMeans Estimator

class sagemaker.**KMeansModel**(*model_data*, *role*, *sagemaker_session=None*)

Bases: *sagemaker.model.Model*

Reference KMeans s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor to performs k-means cluster assignment.

class sagemaker.**KMeansPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Assigns input vectors to their closest cluster in a KMeans model.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The nearest cluster is stored in the `closest_cluster` key of the `Record.label` field.

4.2 PCA

The Amazon SageMaker PCA algorithm.

class sagemaker.**PCA**(*role*, *train_instance_count*, *train_instance_type*, *num_components*, *algorithm_mode=None*, *subtract_mean=None*, *extra_components=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

A Principal Components Analysis (PCA) *AmazonAlgorithmEstimatorBase*.

This Estimator may be fit via calls to `fit_ndarray()` or *fit()*. The former allows a PCA model to be fit on a 2-dimensional numpy array. The latter requires Amazon `Record` protobuf serialized data to be stored in S3.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `PCAPredictor` object that can be used to project input vectors to the learned lower-dimensional representation, using the trained PCA model hosted in the SageMaker Endpoint.

PCA Estimators can be configured by setting hyperparameters. The available hyperparameters for PCA are documented below. For further information on the AWS PCA algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/pca.html

This Estimator uses Amazon SageMaker PCA to perform training and host deployed models. To learn more about Amazon SageMaker PCA, please read: https://docs.aws.amazon.com/sagemaker/latest/dg/how-pca-works.html

Parameters

- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if accessing AWS resource.

- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_components** (*int*) – The number of principal components. Must be greater than zero.
- **algorithm_mode** (*str*) – Mode for computing the principal components. One of 'regular' or 'randomized'.
- **subtract_mean** (*bool*) – Whether the data should be unbiased both during train and at inference.
- **extra_components** (*int*) – As the value grows larger, the solution becomes more accurate but the runtime and memory consumption increase linearly. If this value is unset or set to -1, then a default value equal to the maximum of 10 and num_components will be used. Valid for randomized mode only.
- ****kwargs** – base class keyword argument values.

repo_name = 'pca'

repo_version = 1

DEFAULT_MINI_BATCH_SIZE = 500

create_model()

Return a `PCAModel` referencing the latest s3 model data produced by this Estimator.

classmethod attach(*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **training_job_name** (*str*) – The name of the training job to attach to.
- **sagemaker_session** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

data_location

delete_endpoint()

Delete an Amazon SageMaker Endpoint.

Raises ValueError – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a sagemaker.RealTimePredictor object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **initial_instance_count** (*int*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- ****kwargs** – Passed to invocation of create_model(). Implementations may customize create_model() to accept **kwargs to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a predict() method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

records should be an instance of RecordSet. This defines a collection of S3 data files to train this Estimator on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See record_set() to construct a RecordSet object from ndarray arrays.

Parameters

- **records** (RecordSet) – The records to train this Estimator on
- **mini_batch_size** (*int or None*) – The size of each mini-batch to use when training. If None, a default value will be used.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The `fit()` method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

model_data

str – The model location in S3. Only set if Estimator has been `fit()`.

record_set(*train*, *labels=None*, *channel='train'*)

Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **train** (*numpy.ndarray*) – A 2D numpy array of training data.
- **labels** (*numpy.ndarray*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **channel** (*str*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

training_job_analytics

Returns a TrainingJobAnalytics object for the current training job.

class sagemaker.**PCAModel**(*model_data*, *role*, *sagemaker_session=None*)

Bases: *sagemaker.model.Model*

Reference PCA s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor that transforms vectors to a lower-dimensional representation.

class sagemaker.**PCAPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Transforms input vectors to lower-dimesional representations.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

predict() returns a list of Record objects, one for each row in the input ndarray. The lower dimension vector result is stored in the projection key of the Record.label field.

4.3 LinearLearner

The Amazon SageMaker LinearLearner algorithm.

class sagemaker.**LinearLearner**(*role*, *train_instance_count*, *train_instance_type*, *predictor_type*, *binary_classifier_model_selection_criteria=None*, *target_recall=None*, *target_precision=None*, *positive_example_weight_mult=None*, *epochs=None*, *use_bias=None*, *num_models=None*, *num_calibration_samples=None*, *init_method=None*, *init_scale=None*, *init_sigma=None*, *init_bias=None*, *optimizer=None*, *loss=None*, *wd=None*, *l1=None*, *momentum=None*, *learning_rate=None*, *beta_1=None*, *beta_2=None*, *bias_lr_mult=None*, *bias_wd_mult=None*, *use_lr_scheduler=None*, *lr_scheduler_step=None*, *lr_scheduler_factor=None*, *lr_scheduler_minimum_lr=None*, *normalize_data=None*, *normalize_label=None*, *unbias_data=None*, *unbias_label=None*, *num_point_for_scaler=None*, *margin=None*, *quantile=None*, *loss_insensitivity=None*, *huber_delta=None*, *early_stopping_patience=None*, *early_stopping_tolerance=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

An Estimator for binary classification and regression.

Amazon SageMaker Linear Learner provides a solution for both classification and regression problems, allowing for exploring different training objectives simultaneously and choosing the best solution from a validation set. It allows the user to explore a large number of models and choose the best, which optimizes either continuous objectives such as mean square error, cross entropy loss, absolute error, etc., or discrete objectives suited for classification such as F1 measure, precision@recall, accuracy. The implementation provides a significant speedup over naive hyperparameter optimization techniques and an added convenience, when compared with solutions providing a solution only to continuous objectives.

This Estimator may be fit via calls to fit_ndarray() or *fit()*. The former allows a LinearLearner model to be fit on a 2-dimensional numpy array. The latter requires Amazon Record protobuf serialized data to be stored in S3.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking deploy(). As well as deploying an Endpoint, deploy returns a LinearLearnerPredictor object that can be used to make class or regression predictions, using the trained model.

LinearLearner Estimators can be configured by setting hyperparameters. The available hyperparameters for LinearLearner are documented below. For further information on the AWS LinearLearner algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html

Parameters

- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if accessing AWS resource.

- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **predictor_type** (*str*) – The type of predictor to learn. Either "binary_classifier" or "regressor".
- **binary_classifier_model_selection_criteria** (*str*) – One of 'accuracy', 'f1', 'precision_at_target_recall',
- **'cross_entropy_loss'** (*'recall_at_target_precision',*) –
- **target_recall** (*float*) – Target recall. Only applicable if binary_classifier_model_selection_criteria is precision_at_target_recall.
- **target_precision** (*float*) – Target precision. Only applicable if binary_classifier_model_selection_criteria is recall_at_target_precision.
- **positive_example_weight_mult** (*float*) – The importance weight of positive examples is multiplied by this constant. Useful for skewed datasets. Only applies for classification tasks.
- **epochs** (*int*) – The maximum number of passes to make over the training data.
- **use_bias** (*bool*) – Whether to include a bias field
- **num_models** (*int*) – Number of models to train in parallel. If not set, the number of parallel models to train will be decided by the algorithm itself. One model will be trained according to the given training
- **parameter** (*regularization, optimizer, loss*) –
- **num_calibration_samples** (*int*) – Number of observations to use from validation dataset for doing model
- **calibration** (*finding the best threshold*) –
- **init_method** (*str*) – Function to use to set the initial model weights. One of "uniform" or "normal"
- **init_scale** (*float*) – For "uniform" init, the range of values.
- **init_sigma** (*float*) – For "normal" init, the standard-deviation.
- **init_bias** (*float*) – Initial weight for bias term
- **optimizer** (*str*) – One of 'sgd', 'adam' or 'auto'
- **loss** (*str*) – One of 'logistic', 'squared_loss', 'absolute_loss', 'hinge_loss',
- **'eps_insensitive_absolute_loss', 'quantile_loss', 'huber_loss' or 'auto'** (*'eps_insensitive_squared_loss',*) –
- **wd** (*float*) – L2 regularization parameter i.e. the weight decay parameter. Use 0 for no L2 regularization.
- **l1** (*float*) – L1 regularization parameter. Use 0 for no L1 regularization.
- **momentum** (*float*) – Momentum parameter of sgd optimizer.

- **learning_rate** (*float*) – The SGD learning rate
- **beta_1** (*float*) – Exponential decay rate for first moment estimates. Only applies for adam optimizer.
- **beta_2** (*float*) – Exponential decay rate for second moment estimates. Only applies for adam optimizer.
- **bias_lr_mult** (*float*) – Allows different learning rate for the bias term. The actual learning rate for the
- **is learning rate times bias_lr_mult.** (*bias*) –
- **bias_wd_mult** (*float*) – Allows different regularization for the bias term. The actual L2 regularization weight
- **the bias is wd times bias_wd_mult. By default there is no regularization on the bias term.** (*for*) –
- **use_lr_scheduler** (*bool*) – If true, we use a scheduler for the learning rate.
- **lr_scheduler_step** (*int*) – The number of steps between decreases of the learning rate. Only applies to learning rate scheduler.
- **lr_scheduler_factor** (*float*) – Every lr_scheduler_step the learning rate will decrease by this quantity. Only applies for learning rate scheduler.
- **lr_scheduler_minimum_lr** (*float*) – The learning rate will never decrease to a value lower than this.
- **lr_scheduler_minimum_lr** – Only applies for learning rate scheduler.
- **normalize_data** (*bool*) – Normalizes the features before training to have standard deviation of 1.0.
- **normalize_label** (*bool*) – Normalizes the regression label to have a standard deviation of 1.0. If set for classification, it will be ignored.
- **unbias_data** (*bool*) – If true, features are modified to have mean 0.0.
- **ubias_label** (*bool*) – If true, labels are modified to have mean 0.0.
- **num_point_for_scaler** (*int*) – The number of data points to use for calculating the normalizing and unbiasing terms.
- **margin** (*float*) – the margin for hinge_loss.
- **quantile** (*float*) – Quantile for quantile loss. For quantile q, the model will attempt to produce
- **such that true_label < prediction with probability q.** (*predictions*) –
- **loss_insensitivity** (*float*) – Parameter for epsilon insensitive loss type. During training and metric
- **any error smaller than this is considered to be zero.** (*evaluation,*) –
- **huber_delta** (*float*) – Parameter for Huber loss. During training and metric evaluation, compute L2 loss for
- **smaller than delta and L1 loss for errors larger than delta.** (*errors*) –

- **early_stopping_patience** (*int*) – the number of epochs to wait before ending training if no improvement is
- **The improvement is training loss if validation data is not provided, or else it is the validation** (*made.*) –
- **or the binary classification model selection criteria like accuracy, f1-score etc. To disable early** (*loss*) –
- **set early_stopping_patience to a value larger than epochs.** (*stopping,*) –
- **early_stopping_tolerance** (*float*) – Relative tolerance to measure an improvement in loss. If the ratio of
- **improvement in loss divided by the previous best loss is smaller than this value, early stopping will** (*the*) –
- **the improvement to be zero.** (*consider*) –
- ****kwargs** – base class keyword argument values.

repo_name = 'linear-learner'

repo_version = 1

DEFAULT_MINI_BATCH_SIZE = 1000

classmethod attach (*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **training_job_name** (*str*) – The name of the training job to attach to.
- **sagemaker_session** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **initial_instance_count** (*int*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- ****kwargs** – Passed to invocation of `create_model()`. Implementations may customize `create_model()` to accept `**kwargs` to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a `predict()` method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `RecordSet`. This defines a collection of S3 data files to train this `Estimator` on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`RecordSet`) – The records to train this `Estimator` on
- **mini_batch_size** (*int or None*) – The size of each mini-batch to use when training. If `None`, a default value will be used.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The `fit()` method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

`model_data`
str – The model location in S3. Only set if Estimator has been `fit()`.

`record_set`(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **`train`** (*`numpy.ndarray`*) – A 2D numpy array of training data.
- **`labels`** (*`numpy.ndarray`*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **`channel`** (*`str`*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

`train_image`()
Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

`training_job_analytics`
Returns a TrainingJobAnalytics object for the current training job.

`normalize_data`
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

`normalize_label`
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

`unbias_data`
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

`unbias_label`
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

`num_point_for_scaler`
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

`margin`
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

quantile
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

loss_insensitivity
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

huber_delta
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

early_stopping_patience
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

early_stopping_tolerance
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

create_model()
Return a `LinearLearnerModel` referencing the latest s3 model data produced by this Estimator.

class `sagemaker.`**LinearLearnerModel**(*model_data*, *role*, *sagemaker_session=None*)
Bases: *sagemaker.model.Model*

Reference LinearLearner s3 model data. Calling *deploy()* creates an Endpoint and returns a *LinearLearnerPredictor*

class `sagemaker.`**LinearLearnerPredictor**(*endpoint*, *sagemaker_session=None*)
Bases: *sagemaker.predictor.RealTimePredictor*

Performs binary-classification or regression prediction from input vectors.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The prediction is stored in the `"predicted_label"` key of the `Record.label` field.

4.4 Amazon Estimators

Base class for Amazon Estimator implementations

class `sagemaker.amazon.amazon_estimator.`**AmazonAlgorithmEstimatorBase**(*role*, *train_instance_count*, *train_instance_type*, *data_location=None*, ***kwargs*)
Bases: `sagemaker.estimator.EstimatorBase`

Base class for Amazon first-party Estimator implementations. This class isn't intended to be instantiated directly.

Initialize an AmazonAlgorithmEstimatorBase.

Parameters data_location (*str or None*) – The s3 prefix to upload RecordSet objects to, expressed as an S3 url. For example "s3://example-bucket/some-key-prefix/". Objects will be saved in a unique sub-directory of the specified location. If None, a default data location will be used.

feature_dim
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

mini_batch_size
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The `fit()` method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

data_location

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `RecordSet`. This defines a collection of S3 data files to train this `Estimator` on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`RecordSet`) – The records to train this `Estimator` on
- **mini_batch_size** (*int or None*) – The size of each mini-batch to use when training. If `None`, a default value will be used.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

record_set(*train*, *labels=None*, *channel='train'*)

Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **train** (*numpy.ndarray*) – A 2D numpy array of training data.
- **labels** (*numpy.ndarray*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **channel** (*str*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

4.5 FactorizationMachines

The Amazon SageMaker Factorization Machines algorithm.

class sagemaker.**FactorizationMachines**(*role*, *train_instance_count*, *train_instance_type*, *num_factors*, *predictor_type*, *epochs=None*, *clip_gradient=None*, *eps=None*, *rescale_grad=None*, *bias_lr=None*, *linear_lr=None*, *factors_lr=None*, *bias_wd=None*, *linear_wd=None*, *factors_wd=None*, *bias_init_method=None*, *bias_init_scale=None*, *bias_init_sigma=None*, *bias_init_value=None*, *linear_init_method=None*, *linear_init_scale=None*, *linear_init_sigma=None*, *linear_init_value=None*, *factors_init_method=None*, *factors_init_scale=None*, *factors_init_sigma=None*, *factors_init_value=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

Factorization Machines is `Estimator` for general-purpose supervised learning.

Amazon SageMaker Factorization Machines is a general-purpose supervised learning algorithm that you can use for both classification and regression tasks. It is an extension of a linear model that is designed to parsimoniously capture interactions between features within high dimensional sparse datasets.

This Estimator may be fit via calls to *fit()*. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility *record_set()* that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `FactorizationMachinesPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

FactorizationMachines Estimators can be configured by setting hyperparameters. The available hyperparameters for FactorizationMachines are documented below.

For further information on the AWS FactorizationMachines algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines.html

Parameters

- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if accessing AWS resource.

- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_factors** (*int*) – Dimensionality of factorization.
- **predictor_type** (*str*) – Type of predictor 'binary_classifier' or 'regressor'.
- **epochs** (*int*) – Number of training epochs to run.
- **clip_gradient** (*float*) – Optimizer parameter. Clip the gradient by projecting onto the box [-clip_gradient, +clip_gradient]
- **eps** (*float*) – Optimizer parameter. Small value to avoid division by 0.
- **rescale_grad** (*float*) – Optimizer parameter. If set, multiplies the gradient with rescale_grad before updating. Often choose to be 1.0/batch_size.
- **bias_lr** (*float*) – Non-negative learning rate for the bias term.
- **linear_lr** (*float*) – Non-negative learning rate for linear terms.
- **factors_lr** (*float*) – Noon-negative learning rate for factorization terms.
- **bias_wd** (*float*) – Non-negative weight decay for the bias term.
- **linear_wd** (*float*) – Non-negative weight decay for linear terms.
- **factors_wd** (*float*) – Non-negative weight decay for factorization terms.
- **bias_init_method** (*string*) – Initialization method for the bias term: 'normal', 'uniform' or 'constant'.
- **bias_init_scale** (*float*) – Non-negative range for initialization of the bias term that takes effect when bias_init_method parameter is 'uniform'
- **bias_init_sigma** (*float*) – Non-negative standard deviation for initialization of the bias term that takes effect when bias_init_method parameter is 'normal'.
- **bias_init_value** (*float*) – Initial value of the bias term that takes effect when bias_init_method parameter is 'constant'.
- **linear_init_method** (*string*) – Initialization method for linear term: 'normal', 'uniform' or 'constant'.
- **linear_init_scale** (*float*) – Non-negative range for initialization of linear terms that takes effect when linear_init_method parameter is 'uniform'.
- **linear_init_sigma** (*float*) – Non-negative standard deviation for initialization of linear terms that takes effect when linear_init_method parameter is 'normal'.
- **linear_init_value** (*float*) – Initial value of linear terms that takes effect when linear_init_method parameter is 'constant'.
- **factors_init_method** (*string*) – Initialization method for factorization term: 'normal', 'uniform' or 'constant'.
- **factors_init_scale** (*float*) – Non-negative range for initialization of factorization terms that takes effect when factors_init_method parameter is 'uniform'.

- **factors_init_sigma** (*float*) – Non-negative standard deviation for initialization of factorization terms that takes effect when factors_init_method parameter is 'normal'.
- **factors_init_value** (*float*) – Initial value of factorization terms that takes effect when factors_init_method parameter is 'constant'.
- ****kwargs** – base class keyword argument values.

repo_name = 'factorization-machines'

repo_version = 1

classmethod attach (*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **training_job_name** (*str*) – The name of the training job to attach to.
- **sagemaker_session** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

data_location

delete_endpoint ()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy (*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **initial_instance_count** (*int*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.

- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- **`**kwargs`** – Passed to invocation of `create_model()`. Implementations may customize `create_model()` to accept `**kwargs` to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a `predict()` method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `RecordSet`. This defines a collection of S3 data files to train this `Estimator` on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`RecordSet`) – The records to train this `Estimator` on
- **mini_batch_size** (*int or None*) – The size of each mini-batch to use when training. If `None`, a default value will be used.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The `fit()` method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

model_data

str – The model location in S3. Only set if Estimator has been `fit()`.

record_set(*train*, *labels=None*, *channel='train'*)

Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **train** (*numpy.ndarray*) – A 2D numpy array of training data.
- **labels** (*numpy.ndarray*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **channel** (*str*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

training_job_analytics

Returns a TrainingJobAnalytics object for the current training job.

create_model()

Return a `FactorizationMachinesModel` referencing the latest s3 model data produced by this Estimator.

class sagemaker.**FactorizationMachinesModel**(*model_data*, *role*, *sagemaker_session=None*)

Bases: *sagemaker.model.Model*

Reference S3 model data created by FactorizationMachines estimator. Calling *deploy()* creates an Endpoint and returns *FactorizationMachinesPredictor*.

class sagemaker.**FactorizationMachinesPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Performs binary-classification or regression prediction from input vectors.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The prediction is stored in the `"score"` key of the `Record.label` field. Please refer to the formats details described: https://docs.aws.amazon.com/sagemaker/latest/dg/fm-in-formats.html

4.6 LDA

The Amazon SageMaker LDA algorithm.

class sagemaker.**LDA**(*role*, *train_instance_type*, *num_topics*, *alpha0=None*, *max_restarts=None*, *max_iterations=None*, *tol=None*, ***kwargs*)

Bases: `sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase`

Latent Dirichlet Allocation (LDA) is `Estimator` used for unsupervised learning.

Amazon SageMaker Latent Dirichlet Allocation is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories. LDA is most commonly used to discover a user-specified number of topics shared by documents within a text corpus. Here each observation is a document, the features are the presence (or occurrence count) of each word, and the categories are the topics.

This Estimator may be fit via calls to `fit()`. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility `record_set()` that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `LDAPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

LDA Estimators can be configured by setting hyperparameters. The available hyperparameters for LDA are documented below.

For further information on the AWS LDA algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/lda.html

Parameters

- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if accessing AWS resource.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, ‘ml.c4.xlarge’.
- **num_topics** (*int*) – The number of topics for LDA to find within the data.
- **alpha0** (*float*) – Optional. Initial guess for the concentration parameter
- **max_restarts** (*int*) – Optional. The number of restarts to perform during the Alternating Least Squares (ALS) spectral decomposition phase of the algorithm.
- **max_iterations** (*int*) – Optional. The maximum number of iterations to perform during the ALS phase of the algorithm.
- **tol** (*float*) – Optional. Target error tolerance for the ALS phase of the algorithm.
- ****kwargs** – base class keyword argument values.

repo_name = 'lda'

repo_version = 1

create_model()

Return a `LDAModel` referencing the latest s3 model data produced by this Estimator.

classmethod attach(*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **`training_job_name`** (*`str`*) – The name of the training job to attach to.
- **`sagemaker_session`** (`sagemaker.session.Session`) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

`data_location`

`delete_endpoint``()`

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

`deploy``(`*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*`)`

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **`initial_instance_count`** (*`int`*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.
- **`instance_type`** (*`str`*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **`endpoint_name`** (*`str`*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- **`**kwargs`** – Passed to invocation of `create_model()`. Implementations may customize `create_model()` to accept `**kwargs` to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a `predict()` method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `RecordSet`. This defines a collection of S3 data files to train this `Estimator` on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`RecordSet`) – The records to train this `Estimator` on
- **mini_batch_size** (*int or None*) – The size of each mini-batch to use when training. If `None`, a default value will be used.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The `fit()` method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

model_data

str – The model location in S3. Only set if Estimator has been `fit()`.

record_set(*train*, *labels=None*, *channel='train'*)

Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **train** (*numpy.ndarray*) – A 2D numpy array of training data.
- **labels** (*numpy.ndarray*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **channel** (*str*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

training_job_analytics

Returns a TrainingJobAnalytics object for the current training job.

class sagemaker.**LDAModel**(*model_data*, *role*, *sagemaker_session=None*)

Bases: `sagemaker.model.Model`

Reference LDA s3 model data. Calling `deploy()` creates an Endpoint and return a Predictor that transforms vectors to a lower-dimensional representation.

class sagemaker.**LDAPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: `sagemaker.predictor.RealTimePredictor`

Transforms input vectors to lower-dimesional representations.

The implementation of `predict()` in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The lower dimension vector result is stored in the `projection` key of the `Record.label` field.

4.7 NTM

The Amazon SageMaker NTM algorithm.

class sagemaker.**NTM**(*role*, *train_instance_count*, *train_instance_type*, *num_topics*, *encoder_layers=None*, *epochs=None*, *encoder_layers_activation=None*, *optimizer=None*, *tolerance=None*, *num_patience_epochs=None*, *batch_norm=None*, *rescale_gradient=None*, *clip_gradient=None*, *weight_decay=None*, *learning_rate=None*, ***kwargs*)

Bases: `sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase`

Neural Topic Model (NTM) is `Estimator` used for unsupervised learning.

This Estimator may be fit via calls to `fit()`. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility `record_set()` that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `NTMPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

NTM Estimators can be configured by setting hyperparameters. The available hyperparameters for NTM are documented below.

For further information on the AWS NTM algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/ntm.html

Parameters

- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if accessing AWS resource.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_topics** (*int*) – Required. The number of topics for NTM to find within the data.
- **encoder_layers** (*list*) – Optional. Represents number of layers in the encoder and the output size of each layer.
- **epochs** (*int*) – Optional. Maximum number of passes over the training data.
- **encoder_layers_activation** (*str*) – Optional. Activation function to use in the encoder layers.
- **optimizer** (*str*) – Optional. Optimizer to use for training.
- **tolerance** (*float*) – Optional. Maximum relative change in the loss function within the last num_patience_epochs number of epochs below which early stopping is triggered.
- **num_patience_epochs** (*int*) – Optional. Number of successive epochs over which early stopping criterion is evaluated.
- **batch_norm** (*bool*) – Optional. Whether to use batch normalization during training.
- **rescale_gradient** (*float*) – Optional. Rescale factor for gradient.
- **clip_gradient** (*float*) – Optional. Maximum magnitude for each gradient component.
- **weight_decay** (*float*) – Optional. Weight decay coefficient. Adds L2 regularization.
- **learning_rate** (*float*) – Optional. Learning rate for the optimizer.
- ****kwargs** – base class keyword argument values.

repo_name = 'ntm'

repo_version = 1

classmethod attach (*training_job_name*, *sagemaker_session=None*)

Attach to an existing training job.

Create an Estimator bound to an existing training job, each subclass is responsible to implement `_prepare_init_params_from_job_description()` as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be `deploy()` ed to create a SageMaker Endpoint and return a `Predictor`.

If the training job is in progress, attach will block and display log messages from the training job, until the training job completes.

Parameters

- **training_job_name** (*str*) – The name of the training job to attach to.

- **sagemaker_session** (sagemaker.session.Session) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

Examples

```
>>> my_estimator.fit(wait=False)
>>> training_job_name = my_estimator.latest_training_job.name
Later on:
>>> attached_estimator = Estimator.attach(training_job_name)
>>> attached_estimator.deploy()
```

Returns Instance of the calling `Estimator` Class with the attached training job.

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *endpoint_name=None*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **initial_instance_count** (*int*) – Minimum number of EC2 instances to deploy to an endpoint for prediction.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- ****kwargs** – Passed to invocation of `create_model()`. Implementations may customize `create_model()` to accept `**kwargs` to customize model creation during deploy. For more, see the implementation docs.

Returns

A predictor that provides a `predict()` method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Return type *sagemaker.predictor.RealTimePredictor*

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `RecordSet`. This defines a collection of S3 data files to train this `Estimator` on.

Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label.

More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

See `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`RecordSet`) – The records to train this `Estimator` on
- **mini_batch_size** (*int or None*) – The size of each mini-batch to use when training. If `None`, a default value will be used.
- **wait** (*bool*) – Whether the call should wait until the job completes (default: True).
- **logs** (*bool*) – Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).
- **job_name** (*str*) – Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The `fit()` method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

model_data

str – The model location in S3. Only set if Estimator has been `fit()`.

record_set(*train*, *labels=None*, *channel='train'*)

Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `Record` object. The vector is stored in the "values" entry of the `features` property of each Record. If `labels` is not None, each corresponding label is assigned to the "values" entry of the `labels` property of each Record.

The collection of `Record` objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3.

The number of S3 objects created is controlled by the `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **train** (*numpy.ndarray*) – A 2D numpy array of training data.
- **labels** (*numpy.ndarray*) – A 1D numpy array of labels. Its length must be equal to the number of rows in `train`.
- **channel** (*str*) – The SageMaker TrainingJob channel this RecordSet should be assigned to.

Returns A RecordSet referencing the encoded, uploading training and label data.

Return type RecordSet

train_image()

Return the Docker image to use for training.

The `fit()` method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

training_job_analytics

Returns a TrainingJobAnalytics object for the current training job.

create_model()

Return a NTMModel referencing the latest s3 model data produced by this Estimator.

class sagemaker.**NTMModel**(*model_data*, *role*, *sagemaker_session=None*)

Bases: *sagemaker.model.Model*

Reference NTM s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor that transforms vectors to a lower-dimensional representation.

class sagemaker.**NTMPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Transforms input vectors to lower-dimesional representations.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy ndarray as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

predict() returns a list of Record objects, one for each row in the input ndarray. The lower dimension vector result is stored in the projection key of the Record.label field.

Python Module Index

Index