SageMaker ML Algorithms

Wednesday, 27 July 2022 14:33

	Algorithm	Input	Usage	Hyperparameter	Instance Types
Linear Learner	- **Linear regression - numeric quantity** - **Linear Threshold Fun: binary and multi-class categorical** classification - **Linear relationship**: x,y plot of feature and target = staright line diagonaly - Property price - number of rooms	- **RecordIO-wrapped protobuf (Float32)** - **CSV** (first column as label), - **Reader**: File or Pipe (performant as it stream from S3) mode	- **Pre-process:** Normalized and shuffled data - **Training**: stochastic gradient - **Optimization Algo:** SGD, Adam,etc - **Tune**: L1 & L2 regularization	- Balance_multiclass_weigths - Learning_rate - min_batch_size - Regularization (L1) - Weight Decay (L2)	- **Single or multi-** machine **CPU** - **Single GPU**
XGBoost	- **eXtreme Gradient boosting:** Boosted group of **decision trees** - **Corrects Decision trees errors from previous model to new model** - **Classification** - **Regression trees**	- **CSV or libsvm** - **AWS: recordIO-protobuf & Parquet**	- **Model**: Serialize/Deserialize models with **Pickle** - **Framework in SM :** Sagemaker.xgboost - **Built-in algorithm**	- **Prevent overfitting**: Subsample, Eta - **Step size shrink**: Eta - **Minimum loss reduction to create partition**: Gamma - **L1** : Alpha - **L2**: Lambda - **Optimize on AUC (area under curve), error, rmse**: Eval_metric - **Adjust positive and negative weights:** scale_pos_weight - **Max depth of tree:** max_depth	- **Multi-instance CPU** - **Memory-bound**: M5 - **XGBoost 1.2 Single GPU**: P3, tree_method_hyperparameter=gpu_hist (**Quick and cost effective**)
Seq2Seq (DL)	- **RNN's and CNN's** with attention - **Machine translation** - Text summarization - **Speech to Text**	- **tokenized text files** - **Convert to recordIO-protobuf (Integer** tokens) - training data, validation data - **vocabulary files**	- **Pre-trained models** in SM - Can take days to train new - **Public training dataset for translation**	- Batch_size - **Optimizer_type** = adam, sgd - Learning_rate - Num_layers_encoder - Num_layers_decoder - Optimize on: • **Accuracy** • **BLUE & Preplexity:** Machine translations	- **Only GPU** (P3) as DL Algo - **Single machine with multi-GPU**
DeepAR (DL)	- **RNN's** - **Forecast one dimensional time-series data** - Train same model over related time series - **Frequencies and Seasonality**	- **JSON lines** in file - **Gzip** or **Parquet** - **Record MUST**: **Start timestamp, target values** - **Record CAN**: **Dynamic**_features, **Categorical** feat	- **entire time series for Training, test and inference** - remove last timepoints from training - **train on many time series** - **Prediction length < 400**	- **Context_length**: **Number of time points before making prediction** - Epochs - Mini_batch_size - Learning_rate - Num_cells	- **Single or multi CPU** - **Single or multi GPU** - Start with CPU - **Inference**: **CPU** - **Tuning**: **Larger** instance
BlazingText (DL)	- **Supervised Text classification:** Label sentences, NOT DOCs - **Word2vector (word embedding)**: Machine **translation**, **sentiment analysis**	- **Text classification:** One sentence per line, first **word "_label_ followed by label"** - **Word2vec**: text file with one sentence per line	**Word2Vec input modes:** Cbow, Skkip-gram, Batch-skip gram (computation over many CPU nodes)	**Word2vec:** - Mode - Learning_rate, window_size, vector_dim, negative_samples **Text Classification:** - Epochs, Learning_rate, Word_grams, Vector_dim	- **Cbow & skinpgram:** Single CPU or GPU (P3) - **Batch_skipgram**: Single or multiple CPU - **Text classification**: CPU if training data < 2 GB otheriwse GPU (P2, P3)
Object2Vec (DL)	- **Object embedding, CNN** - Compute nearest neighbours of objects - **Recommender (similar items)** - Can be used for entire Docs instead of just objects - **Ex: sentence Embedding**	- **Tokenized into integers (Pre-process data)** - **Training data:** Pairs of tokens or sequence of tokens. Ex: User-item, sentence-sentence, **genre-description**	- **Shuffled JSON lines** - **2 input channels:** 2 encoders and one comparator - **Encoder types:** Average-pooled, CNN's, Bidirectional LSTM	- **Deep Learning:** dropout, epochs, learning_rate, batch_size, layers, optimizer, weight_decay, early_stopping - **Enc1_network, Enc2_network** • Hcnn, bilstm, pooled_embedding	- **Single Machine CPU** - **Single Machine with Multi-GPU** - **Inference:** **ml.**p2.2xlarge with INFERENCE_PREFERRED_MODE for encoding
Object Detection (WHERE?)	- **Single deep neural network, CNN with SSD** - Identify **image within bounding boxes** - **Labels to bounding boxes** - **classes with confidence** scores - new or pre-trained models from ImageNet	- **RecordIO or image format** - **JSON file for annotation** data for each image	- **Input:** Image - **Output:** all **instances of images** with **categories** and confidence scores - **Transfer learning**	- Mini_batch_size, Learning_rate - **Optimizer:** sgd, adam, rmsprop, adadelta	- **Training:** Multi-GPU or Multi-machine GPU (p2, p3) - **Inference:** CPU or GPU for (C5, M5, P2, P3)
Image Classification (WHAT?)	- **RestNet CNN** - **One or more labels to image** - **Tells what objects are**	- **MXNet RecordIO** - **raw jpg or png images** - **.lst files:** associate image to index, class label, path to image - **Reader**: Pipe mode to stream read images	- **Full training mode:** Initialize Network with random weights - **Transfer learning:** Initialized weights, network fine-tuned with new training data - **Default image size:** 3-channel 224x224	- batch_size, Learning_rate - **Optimizer**: weight decay, beta 1, beta 2, eps, gamma	- **Training:** Multi-GPU or Multi-machine GPU (p2, p3) - **Inference:** CPU or GPU (C4, P2, P3)
Semantic Segmentation (Pixel-level)	- **MXNet Gluon and Gluon CV** - **Pixel level image classification** - Produce segmentation mask - **Ex:** self-driving, medical	- **JPEG and PNG annotations** - Training and validation - Labels maps to describe - **Augmented manifest image format** with Pipe mode	- **3 algorithms:** FCN, PSP, DeepLabV3 - **Backbones:** ResNet50, ResNet101 trained on ImageNet - **Scratch or transfer learning**	- batch_size, Learning_rate, Optimizer, epochs - Algorithm - backbone	- **Training:** Only single machine GPU (P2 or P3) - **Inference:** CPU (C5, M5) or GPU (P2, P3)

	imaging, robot sensing	- **JPG** images accepted for **inference**			
Random Cut Forest	- **Forest of trees (each tree partition of training data)** - **Unsupervised Anomaly Detection** - Unexpected dip or spike in time series data - Assign anomaly score to each data point - Developed by Amazon	- **RecordIO-protobuf or CSV** - **Read**: File or Pipe - **Optional test channel**: computing accuracy, precision, recall and F1 on labelled data	- **Expected change in complexity of tree when adding a point** - Data is sampled randomly then trained, On Streaming data - Works on Stream and Batch data	- **Num_trees:** Increasing reduces noise - Num_samples_per_tree	- **Training:** CPU only (M4, C4, C5) - **Inference:** CPU (ml.C5.xl)
Neural Topic Model (Unsupervised for documents)	- **"Neural Variational Inference"** - **Unsupervised to classify or summarize documents** - Organize docs into topics - **Ex.**bike, car, train classify doc into transportation	- **RecordIO-protobuf or CSV** - **4 channels:** train (mandatory), validation, test, auxiliary. - **Words tokenized into integers:** Count of every word in doc in CSV vocabulary in auxiliary channel. - **Read:** File or Pipe	- **Define how many topics** - **Topics**: latent representation based of top ranking words	- **Lowering Mini_batch_size, Learning_rate:** reduce validation loss at expense of training time - Num_topics	- **Training:** CPU or GPU(recommended) - **Inference:** CPU
Latent Dirichlet Allocation (Unsupervised topic model, not DL)	- **Unsupervised Topic modelling with no DL** - Group docs with shared subset of words - **Ex**: Cluster customer based on purchase	- **RecordIO-protobuf or CSV** - **2 channels:** Train, optional test - Each doc with count of every word in CSV vocabulary - **Read:** File or pipe(only with recordIO)	- **Define how many topics** - **Optional Test channel:** scoring results - **Similar to NTM but CPU based (cheaper and efficient)**	- Num_topics, - **Alpha0** - Initial guess for concentration paramerer, small = sparse topics	- **Training:** Single-instance CPU
KNN - K-Nearest-Neighbors (Supervised Classification)	- **Supervised Classification or regression** - **Classification:** K closest points to sample point and return most frequent label - **Regression**: K closest points to sample point and return average	- **RecordIO-protobuf or CSV** - **Train channel:** Data - **Test channel:** Emit accuracy or MSE - **Read:** File or Pipe	- **Data is first sampled** - **Dimension reduction** - **Build index for looking up neighbors** - **Serialize model** - **Query model for a given K**	- K = how many neighbors to look at - Sample_size	- **Training:** CPU (C5) or GPU (P2) - **Inference:** CPU (Low Latency), GPU (High throughput on large batches)
K-Means Clustering (Unsupervised Segmentation, Grouping)	- **Unsupervised clustering classification** - Divide data into K groups - member of groups similar to each other - Web-scale K-means clustering	- **RecordIO-protobuf or CSV** - **Train channel:** Data ShardedbyS3Key - **Test channel:** FullyReplicated - **Read:** File or Pipe	- Every observation mapped to n-dimensional space (n = number of features) - **Optimize center of K clusters:** Determine initial, calculate from training data, reduce cluster	- **K:** Use "elbow method", optimize for tightness of clusters - Mini_batch_size - Extra_center_factor - Init_method	- **Single GPU per instance (p*.xlarge)** - **CPU(Recommended)**
Principal Component Analysis (PCA) (Unsupervised)	- **Unsupervised Dimensional reduction technique** - minimize loss of info - reduced dimensions called components: **1st comp with large variability,** second next largest - **Ex: Online surveys, census data**	- **RecordIO-protobuf or CSV** - **Read:** File or Pipe	- Covariance matrix created then SVD (singular value decomposition) - **Regular mode algo:** sparse data and moderate observations and features - **Randomized mode algo**: large observations and features and uses approximation algorithm	- Algorithm_mode - Subtract_mean = Unbias data	- **GPU or CPU:** depends on input data
t-Distributed Stochastic Neighbor Embedding (TSNE)	- **non-linear dimensionality reduction algorithm**				
Factorization Machines (Supervised regression)	- **Supervised Classification or regression of Sparse data** - Used in context of **recommender system, click rate pattern** - **Ex**: recommender system with less data from individual user - Limited to pair-wise interactions. User --> item example	- **Must be RecordIO-protobuf with Float32**	- **Find factors we can use to predict classification** - **Ex: click or not click? Predicted rating? Users and items?**	- bias, factors, linear terms - Uniform, normal or constant	- **CPU:** **recommended** - **GPU** only for dense data
Collaborative Filtering (Matrix Factorization)	- **Intelligent recommender** - filter out items that a user might like on the basis of reactions by similar users				
IP Insights (Unsupervised)	- **Unsupervised neural network - IP address usage patterns** - **identify suspicious behavior from given IP**: logins, accounts creating resources	- **CSV only (Entity,IP)** - Directly feed Usernames, AccountIDs - Training channel - optional validation (computes AUC score)	- Neural network to **learn latent vector representation of entities and IP add** - Hashed and embedded entities - **Auto-generate negative samples** during training - Random pairing entities and IP	- **Num_enity_vectors:** Hash size, 2x unique entity identifiers - **Vector_dim:** Size of embedding vectors - Epochs, learning_rate, batch_size.etc	- **CPU:** size depends on vector_dim and num_entity_vectors - **GPU:** recommended, P3, can use Multiple GPU's
Reinforcement Learning (DL)	- **DL framework with Tensorflow and MXNet** - **Agent that explores space** - learn state changes in			- Specific **based on implementation** - **Optimize using SM hyperp**arameters	- **DL so GPU** - **Multi-core, multi-instance.** - **Distributed training**

	different conditionsto predict subsequent behavior. - **Ex:** Pac-Man, Supply chain, Autonomous vehicles, Industrial robotics				**and/or environment rollout**
Q-Learning	- **Implementation of Reinforcement Learning** - **Q = state/action** (S: environmental states, a: possible actions in states) - Starts with zero then **penalty/reward based on learning space** - Randomly explore choices - **Use Q stored values to inform future choices**				