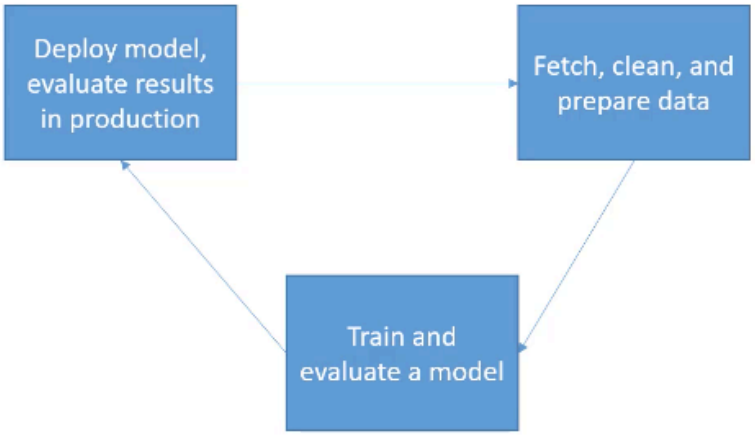
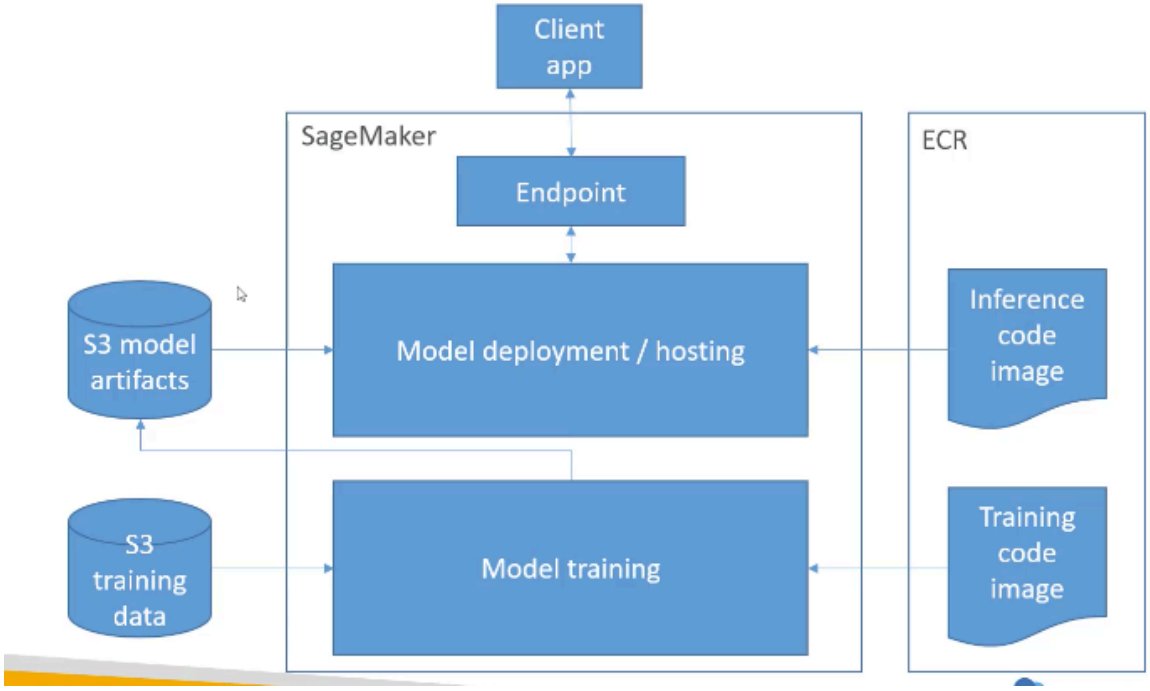
Udemy - 3 - Modeling - ML services - part 1

Introducing Amazon SageMaker

SageMaker is built to handle the entire machine learning workflow.



SageMaker Training & Deployment

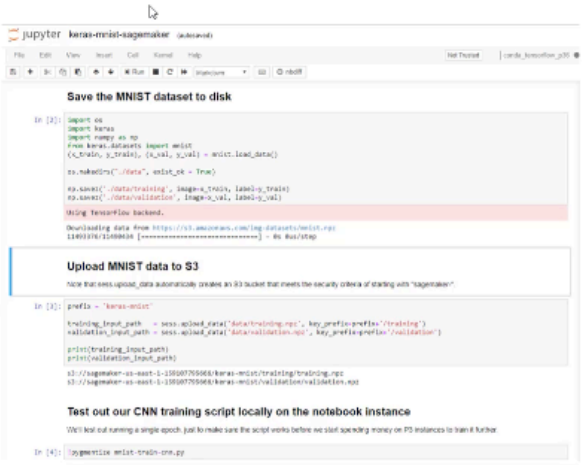


SM provisions a based of hosts to train, leveraging docker image from ECR, and

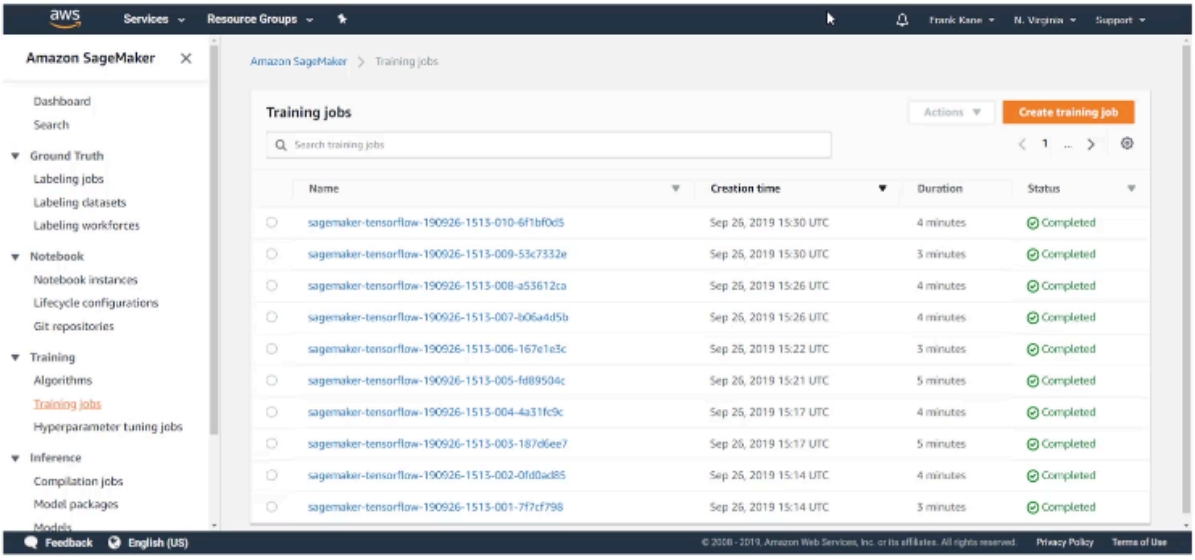
use training data in S3.
It saves the model artifact also in S3
We then have inference code (simple: take incoming requests and use saved model to make inferences)
Pull inference code from ECR, spins up hosts and end point

SageMaker Notebooks can direct the process

- Notebook Instances on EC2 are spun up from the console
  - S3 data access
  - Scikit_learn, Spark, Tensorflow
  - Wide variety of built-in models
  - Ability to spin up training instances
  - Abilty to deploy trained models for making predictions at scale



So can the SageMaker console



Ex: Kick off training job or hyper parameter job from notebook, then look auth the jobs in the console

Data prep on SageMaker

- Data must come from S3
  - Ideal format varies with algorithm – often it is RecordIO / Protobuf
- Apache Spark integrates with SageMaker
  - More on this later…
- Scikit_learn, numpy, pandas all at your disposal within a notebook

Training on SageMaker

- Create a training job
  - URL of S3 bucket with training data
  - ML compute resources
  - URL of S3 bucket for output
  - ECR path to training code
- Training options
  - Built-in training algorithms
  - Spark MLLib
  - Custom Python Tensorflow / MXNet code
  - Your own Docker image
  - Algorithm purchased from AWS marketplace

Deploying Trained Models

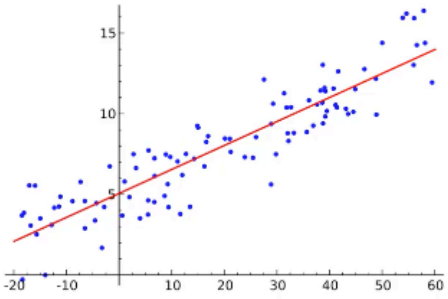
- Save your trained model to S3
- Can deploy two ways:
  - Persistent endpoint for making individual predictions on demand
  - SageMaker Batch Transform to get predictions for an entire dataset
- Lots of cool options
  - Inference Pipelines for more complex processing
  - SageMaker Neo for deploying to edge devices
  - Elastic Inference for accelerating deep learning models
  - Automatic scaling (increase # of endpoints as needed)

SAGEMAKER BUILT-IN ALGORITHMS

Linear Learner

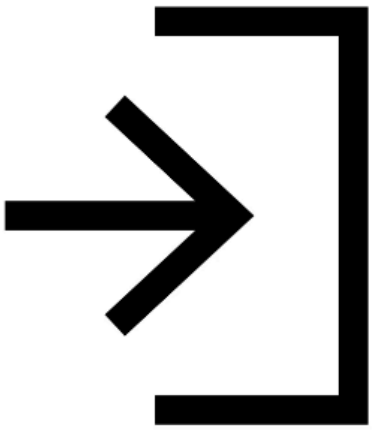
Linear Learner: What's it for?

- Linear regression
  - Fit a line to your training data
  - Predications based on that line
- Can handle both regression (numeric) predictions and classification predictions
  - For classification, a linear threshold function is used.
  - Can do binary or multi-class



Linear Learner: What training input does it expect?

- RecordIO-wrapped protobuf
  - Float32 data only!
- CSV
  - First column assumed to be the label
- File or Pipe mode both supported



=> If a model takes too long to get started to train, use PIPE mode (as it streams data from S3 to the fleet of training instances)

Linear Learner: How is it used?

- Preprocessing
  - Training data must be *normalized* (so all features are weighted the same)
  - Linear Learner can do this for you automatically
  - Input data should be *shuffled*
- Training
  - Uses stochastic gradient descent
  - Choose an optimization algorithm (Adam, AdaGrad, SGD, etc)
  - Multiple models are optimized in parallel
  - Tune L1, L2 regularization
- Validation
  - Most optimal model is selected

Reminder: L1 and L2 helps with overfitting (regularization). L1 does feature selection while L2 is weighting every feature more smoothly

Linear Learner: Important Hyperparameters

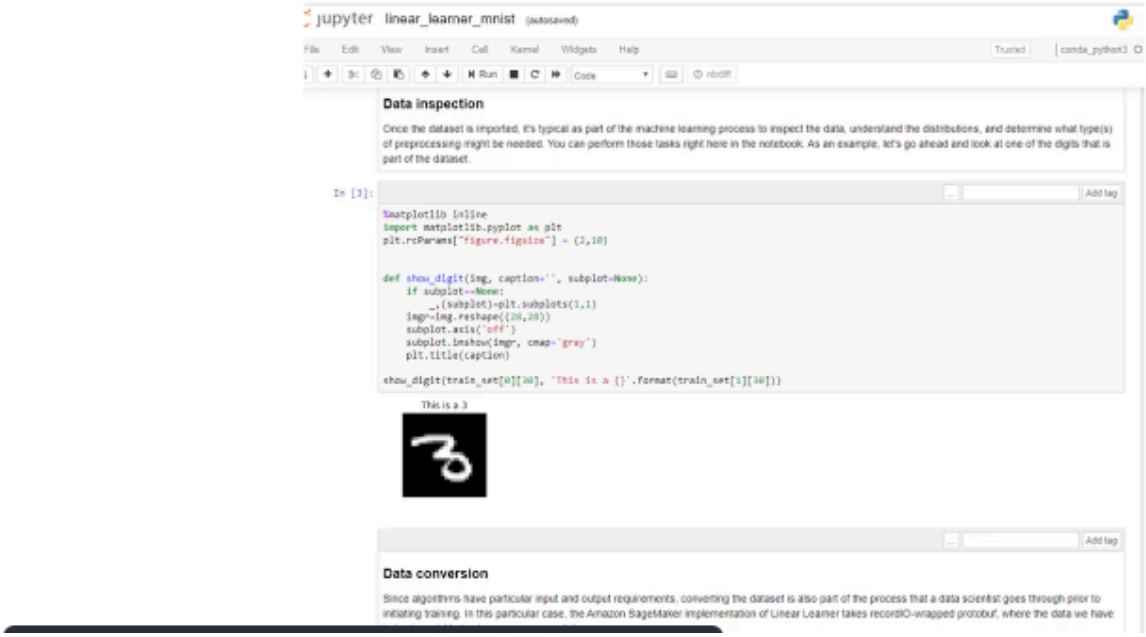
- Balance_multiclass_weights
  - Gives each class equal importance in loss functions
- Learning_rate, mini_batch_size
- L1
  - Regularization
- Wd
  - Weight decay (L2 regularization)

Linear Learner: Instance Types

- Training
  - Single or multi-machine CPU or GPU
  - Multi-GPU does not help

Example: using Linear Learner as a classifier for Hand digits recognition (even though it's not a deep neural net)

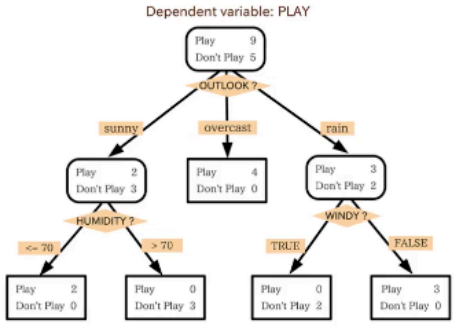
Example



XGBoost in Sagemaker

XGBoost: What's it for?

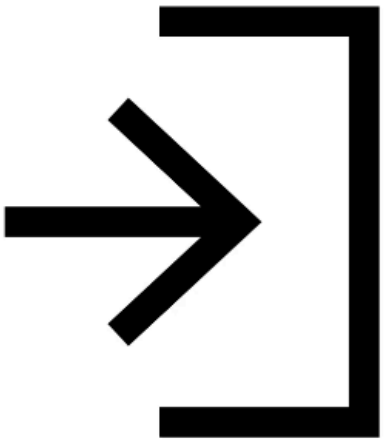
- eXtreme Gradient Boosting
  - Boosted group of decision trees
  - New trees made to correct the errors of previous trees
  - Uses gradient descent to minimize loss as new trees are added
- It's been winning a lot of Kaggle competitions
  - And it's fast, too
- Can be used for classification
- And also for regression
  - Using regression trees



While we think of XGboost for Classification, we can also use it for regression problems

XGBoost: What training input does it expect?

- XGBoost is weird, since it's not made for SageMaker. It's just open source XGBoost
- So, it takes CSV or libsvm input.
- AWS recently extended it to accept recordIO-protobuf and Parquet as well.



XGBoost: How is it used?

- Models are serialized/deserialized with Pickle
- Can use as a framework within notebooks
  - Sagemaker.xgboost
- Or as a built-in SageMaker algorithm

XGBoost: Important Hyperparameters

- There are a lot of them. A few:
- Subsample
  - Prevents overfitting
- Eta
  - Step size shrinkage, prevents overfitting
- Gamma
  - Minimum loss reduction to create a partition; larger = more conservative
- Alpha
  - L1 regularization term; larger = more conservative
- Lambda
  - L2 regularization term; larger = more conservative

XGBoost: Instance Types

- Uses CPU's only
- Is memory-bound, not compute-bound
- So, M4 is a good choice

=> don't bother using a P2 or P3 for that algorithm. Best instance type if a M4

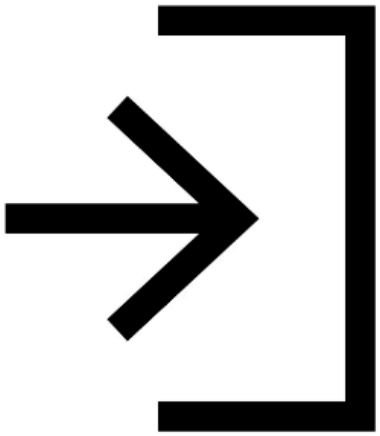
Seq2Seq in SageMaker

Seq2Seq: What's it for?

- Input is a sequence of tokens, output is a sequence of tokens
- Machine Translation
- Text summarization
- Speech to text
- Implemented with RNN's and CNN's with attention

Seq2Seq: What training input does it expect?

- RecordIO-Protobuf
  - Tokens must be integers (this is unusual, since most algorithms want floating point data.)
- Start with tokenized text files
- Convert to protobuf using sample code
  - Packs into integer tensors with vocabulary files
  - A lot like the TF/IDF lab we did earlier.
- Must provide training data, validation data, and vocabulary files.



CAUTION: Cannot pass a text file full or Word Document. Need to pass **tokenized** text file
AND convert everything to PROTOBUF (RecordIO)

Seq2Seq: How is it used?

- Training for machine translation can take days, even on SageMaker
- Pre-trained models are available
  - See the example notebook
- Public training datasets are available for specific translation tasks

Seq2Seq: Important Hyperparameters

- Batch_size
- Optimizer_type (adam, sgd, rmsprop)
- Learning_rate
- Num_layers_encoder
- Num_layers_decoder
- Can optimize on:
  - Accuracy
    - Vs. provided validation dataset
  - BLEU score
    - Compares against multiple reference translations
  - Perplexity
    - Cross-entropy

Seq2Seq: Instance Types

- Can only use GPU instance types (P3 for example)
- Can only use a single machine for training
  - But can use multi-GPU's on one machine

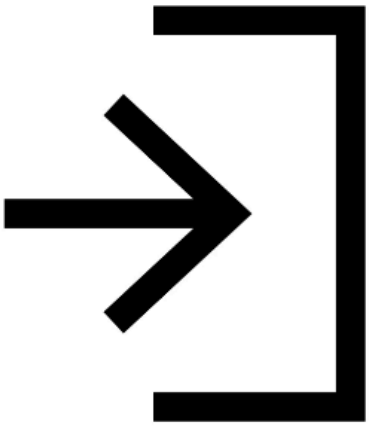
DeepAR in SageMaker

DeepAR: What's it for?

- Forecasting one-dimensional time series data
- Uses RNN's
- Allows you to train the same model over several related time series
- Finds frequencies and seasonality

DeepAR: What training input does it expect?

- JSON lines format
  - Gzip or Parquet
- Each record must contain:
  - Start: the starting time stamp
  - Target: the time series values
- Each record can contain:
  - Dynamic_feat: dynamic features (such as, was a promotion applied to a product in a time series product purchases)
  - Cat: categorical features



```
{"start": "2009-11-01 00:00:00", "target": [4.3, "NaN", 5.1, ...], "cat": [0, 1], "dynamic_feat": [[1.1, 1.2, 0.5, ...]]}
{"start": "2012-01-30 00:00:00", "target": [1.0, -5.0, ...], "cat": [2, 3], "dynamic_feat": [[1.1, 2.05, ...]]}
{"start": "1999-01-30 00:00:00", "target": [2.0, 1.0], "cat": [1, 4], "dynamic_feat": [[1.3, 0.4]]}
```

DeepAR: How is it used?

- Always include entire time series for training, testing, and inference
- Use entire dataset as test set, remove last time points for training. Evaluate on withheld values.
- Don't use very large values for prediction length (> 400)
- Train on many time series and not just one when possible

DeepAR: Important Hyperparameters

- Context_length
  - Number of time points the model sees before making a prediction
  - Can be smaller than seasonalities; the model will lag one year anyhow.
- Epochs
- mini_batch_size
- Learning_rate
- Num_cells

DeepAR: Instance Types

- Can use CPU or GPU
- Single or multi machine
- Start with CPU (C4.2xlarge, C4.4xlarge)
- Move up to GPU if necessary
  - Only helps with larger models
- CPU-only for inference
- May need larger instances for tuning

BlazingText in SageMaker

2 modes:

- **Text Classification:** Supervised learning where we pass sentences and labels.

IMPORTANT: **This is working against SENTENCES, not entire word documents**

- **Word2Vec:** creates word embeddings. Only works ON WORDS or SENTENCES, not ENTIRE documents.

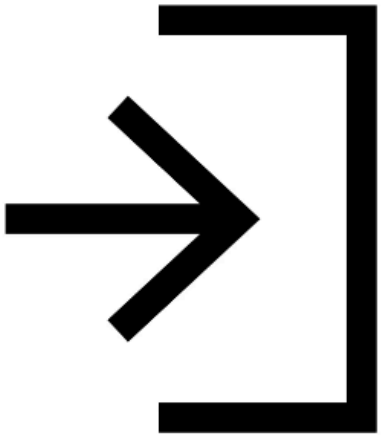
Useful for downstream NLP tasks such as sentiment analysis, machine translation, ...

BlazingText: What's it for?

- Text classification
  - Predict labels for a sentence
  - Useful in web searches, information retrieval
  - Supervised
- Word2vec
  - Creates a vector representation of words
  - Semantically similar words are represented by vectors close to each other
  - This is called a *word embedding*
  - It is useful for NLP, but is not an NLP algorithm in itself!
    - Used in machine translation, sentiment analysis
  - Remember it only works on individual words, not sentences or documents

BlazingText: What training input does it expect?

- For supervised mode (text classification):
  - One sentence per line
  - First "word" in the sentence is the string __label__ followed by the label
- Also, "augmented manifest text format"
- Word2vec just wants a text file with one training sentence per line.



__label__4 linux ready for prime time , intel says , despite all the linux hype , the open-source movement has yet to make a huge splash in the desktop market . that may be about to change , thanks to chipmaking giant intel corp .

label 2 bowled by the slower one again , kolkata , november 14 the past caught up with sourav ganguly as the indian skippers return to international cricket was short lived .

{"source":"linux ready for prime time , intel says , despite all the linux hype", "label":1}
{"source":"bowled by the slower one again , kolkata , november 14 the past caught up with sourav ganguly", "label":2}

BlazingText: How is it used?

- Word2vec has multiple modes
  - Cbow (Continuous Bag of Words)
  - Skip-gram
  - Batch skip-gram
    - Distributed computation over many CPU nodes

CBow: order does not matter (it's a bag of words)
Skip-gram (order does matter)

BlazingText: Important Hyperparameters

- Word2vec:
  - Mode (batch_skipgram, skipgram, cbow)
  - Learning_rate
  - Window_size
  - Vector_dim
  - Negative_samples
- Text classification:
  - Epochs
  - Learning_rate
  - Word_ngrams
  - Vector_dim

BlazingText: Instance Types

- For cbow and skipgram, recommend a single ml.p3.2xlarge
  - Any single CPU or single GPU instance will work
- For batch_skipgram, can use single or multiple CPU instances
- For text classification, C5 recommended if less than 2GB training data. For larger data sets, use a single GPU instance (ml.p2.xlarge or ml.p3.2xlarge)

Object2Vec
Like word2vec that can only work against words.
Except it can work against full documents or other objects

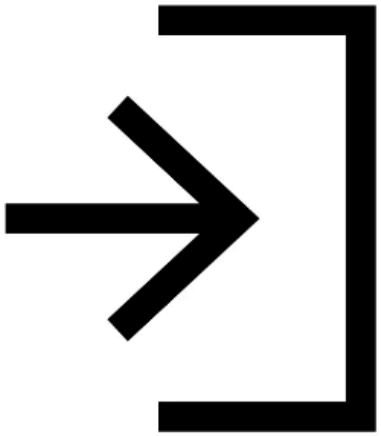
Idea is to find “objects” that are close one to the other.
-> great for recommendation (or clustering)

Object2Vec: What's it for?

- Remember word2vec from Blazing Text? It's like that, but arbitrary objects
- It creates low-dimensional dense embeddings of high-dimensional objects
- It is basically word2vec, generalized to handle things other than words.
- Compute nearest neighbors of objects
- Visualize clusters
- Genre prediction
- Recommendations (similar items or users)
- Unsupervised

Object2Vec: What training input does it expect?

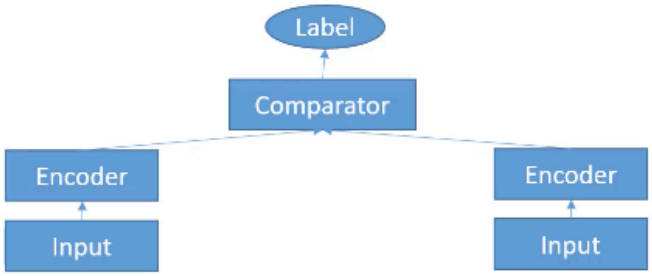
- Data must be tokenized into integers
- Training data consists of pairs of tokens and/or sequences of tokens
  - Sentence – sentence
  - Labels-sequence (genre to description?)
  - Customer-customer
  - Product-product
  - User-item



{"label": 0, "in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821, 4], "in1": [16, 21, 13, 45, 14, 9, 80, 59, 164, 4]}
{"label": 1, "in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4], "in1": [22, 32, 13, 25, 1016, 573, 3252, 4]}
{"label": 1, "in0": [774, 14, 21, 206], "in1": [21, 366, 125]}

Object2Vec: How is it used?

- Process data into JSON Lines and shuffle it
- Train with two input channels, two encoders, and a comparator
- Encoder choices:
  - Average-pooled embeddings
  - CNN's
  - Bidirectional LSTM
- Comparator is followed by a feed-forward neural network



Object2Vec: Important Hyperparameters

- The usual deep learning ones…
  - Dropout, early stopping, epochs, learning rate, batch size, layers, activation function, optimizer, weight decay
- Enc1_network, enc2_network
  - Choose hcnn, bilstm, pooled_embedding

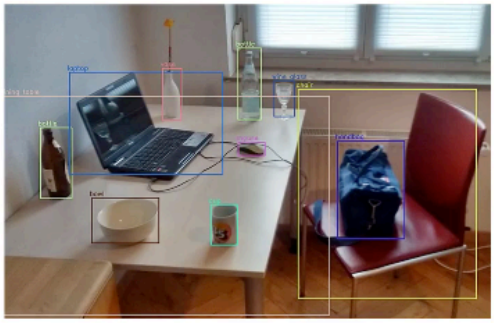
Object2Vec: Instance Types

- Can only train on a single machine (CPU or GPU, multi-GPU OK)
  - Ml.m5.2xlarge
  - Ml.p2.xlarge
  - If needed, go up to ml.m5.4xlarge or ml.m5.12xlarge
- Inference: use ml.p2.2xlarge
  - Use INFERENCE_PREFERRED_MODE environment variable to optimize for encoder embeddings rather than classification or regression.

Object Detection in SageMaker

Object Detection: What's it for?

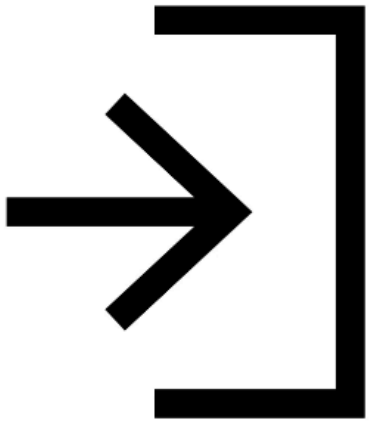
- Identify all objects in an image with bounding boxes
- Detects and classifies objects with a single deep neural network
- Classes are accompanied by confidence scores
- Can train from scratch, or use pre-trained models based on ImageNet



Object Detection: What training input does it expect?

- RecordIO or image format (jpg or png)
- With image format, supply a JSON file for annotation data for each image


```
{
  "file": "your_image_directory/sample_image1.jpg",
  "image_size": [
    {
      "width": 500,
      "height": 400,
      "depth": 3
    }
  ],
  "annotations": [
    {
      "class_id": 0,
      "left": 111,
      "top": 134,
      "width": 61,
      "height": 128
    },
  ],
  "categories": [
    {
      "class_id": 0,
      "name": "dog"
    },
  ]
}
```

Object Detection: How is it used?

- Takes an image as input, outputs all instances of objects in the image with categories and confidence scores
- Uses a CNN with the Single Shot multibox Detector (SSD) algorithm
  - The base CNN can be VGG-16 or ResNet-50
- Transfer learning mode / incremental training
  - Use a pre-trained model for the base network weights, instead of random initial weights
- Uses flip, rescale, and jitter internally to avoid overfitting

Object Detection: Important Hyperparameters

- Mini_batch_size
- Learning_rate
- Optimizer
  - Sgd, adam, rmsprop, adadelta

Object Detection: Instance Types

- Use GPU instances for training (multi-GPU and multi-machine OK)
  - Ml.p2.xlarge, ml.p2.8xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.8clarge, ml.p3.16xlarge
- Use CPU or CPU for inference
  - C5, M5, P2, P3 all OK

Image Classification in SageMaker

Image Classification: What's it for?

- Assign one or more labels to an image
- Doesn't tell you where objects are, just what objects are in the image

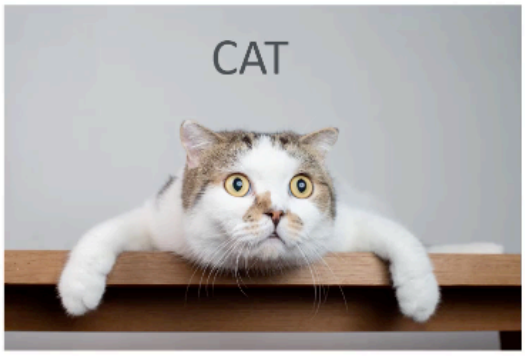
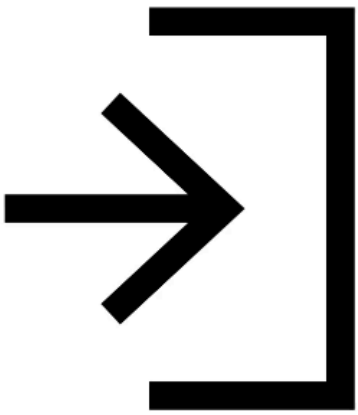


Image Classification: What training input does it expect?

- Apache MXNet RecordIO
  - Not protobuf!
  - This is for interoperability with other deep learning frameworks.
- Or, raw jpg or png images
- Image format requires .lst files to associate image index, class label, and path to the image
- Augmented Manifest Image Format enables Pipe mode

```
5 1 your_image_directory/train_img_dog1.jpg
1000 0 your_image_directory/train_img_cat1.jpg
22 1 your_image_directory/train_img_dog2.jpg
```



```
{"source-ref":"s3://image/filename1.jpg", "class":"0"}
{"source-ref":"s3://image/filename2.jpg", "class":"1", "class-metadata": {"class-name": "cat", "type" : "groundtruth/image-classification"}}
```

Image Classification: How is it used?

- ResNet CNN under the hood
- Full training mode
  - Network initialized with random weights
- Transfer learning mode
  - Initialized with pre-trained weights
  - The top fully-connected layer is initialized with random weights
  - Network is fine-tuned with new training data
- Default image size is 3-channel 224x224 (ImageNet's dataset)

Image Classification: Important Hyperparameters

- The usual suspects for deep learning
  - Batch size, learning rate, optimizer
- Optimizer-specific parameters
  - Weight decay, beta 1, beta 2, eps, gamma

Image Classification: Instance Types

- GPU instances for training (P2, P3) Multi-GPU and multi-machine OK.
- CPU or GPU for inference (C4, P2, P3)

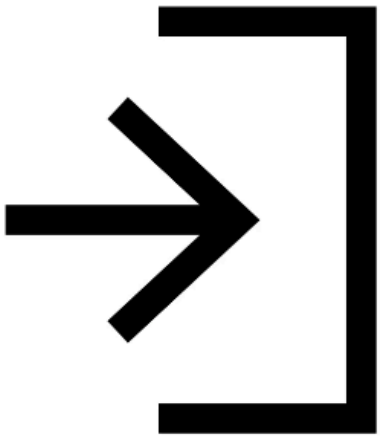
Semantic Segmentation in SageMaker

Semantic Segmentation: What's it for?

- Pixel-level object classification
- Different from image classification – that assigns labels to whole images
- Different from object detection – that assigns labels to bounding boxes
- Useful for self-driving vehicles, medical imaging diagnostics, robot sensing
- Produces a *segmentation mask*

Semantic Segmentation: What training input does it expect?

- JPG Images and PNG annotations
- For both training and validation
- Label maps to describe annotations
- Augmented manifest image format supported for Pipe mode.
- JPG images accepted for inference



Semantic Segmentation: How is it used?

- Built on MXNet Gluon and Gluon CV
- Choice of 3 algorithms:
  - Fully-Convolutional Network (FCN)
  - Pyramid Scene Parsing (PSP)
  - DeepLabV3
- Choice of backbones:
  - ResNet50
  - ResNet101
  - Both trained on ImageNet
- Incremental training, or training from scratch, supported too

Semantic Segmentation: Important Hyperparameters

- Epochs, learning rate, batch size, optimizer, etc
- Algorithm
- Backbone

Semantic Segmentation: Instance Types

- Only GPU supported for training (P2 or P3) on a single machine only
  - Specifically ml.p2.xlarge, ml.p2.8xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.8xlarge, or ml.p3.16xlarge
- Inference on CPU (C5 or M5) or GPU (P2 or P3)

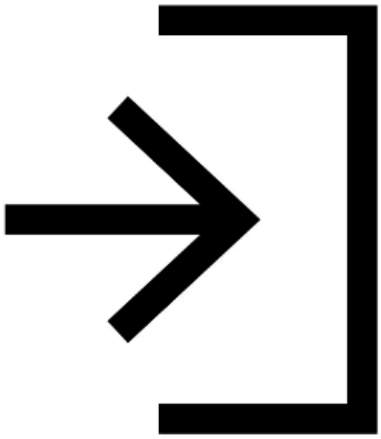
RCF (Random Cut Forest) in SageMaker

Random Cut Forest: What's it for?

- Anomaly detection
- Unsupervised
- Detect unexpected spikes in time series data
- Breaks in periodicity
- Unclassifiable data points
- Assigns an anomaly score to each data point
- Based on an algorithm developed by Amazon that they seem to be very proud of!

Random Cut Forest: What training input does it expect?

- RecordIO-protobuf or CSV
- Can use File or Pipe mode on either
- Optional test channel for computing accuracy, precision, recall, and F1 on labeled data (anomaly or not)



Random Cut Forest: How is it used?

- Creates a forest of trees where each tree is a partition of the training data; looks at expected change in complexity of the tree as a result of adding a point into it
- Data is sampled randomly
- Then trained
- RCF shows up in Kinesis Analytics as well; it can work on streaming data too.

Random Cut Forest: Important Hyperparameters

- Num_trees
  - Increasing reduces noise
- Num_samples_per_tree
  - Should be chosen such that 1/num_samples_per_tree approximates the ratio of anomalous to normal data

Random Cut Forest: Instance Types

- Does not take advantage of GPUs
- Use M4, C4, or C5 for training
- ml.c5.xl for inference

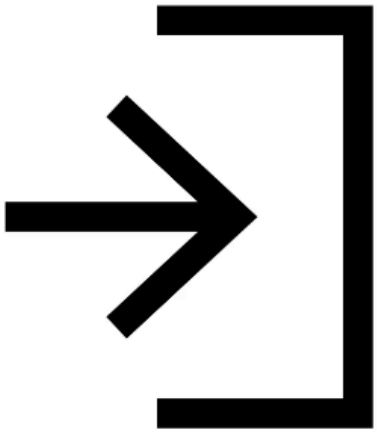
NTM (Neural Topic Modeling) in SageMaker

Neural Topic Model: What's it for?

- Organize documents into topics
- Classify or summarize documents based on topics
- It's not just TF/IDF
  - "bike", "car", "train", "mileage", and "speed" might classify a document as "transportation" for example (although it wouldn't know to call it that)
- Unsupervised
  - Algorithm is "Neural Variational Inference"

Neural Topic Model: What training input does it expect?

- Four data channels
  - "train" is required
  - "validation", "test", and "auxiliary" optional
- recordIO-protobuf or CSV
- Words must be tokenized into integers
  - Every document must contain a count for every word in the vocabulary in CSV
  - The "auxiliary" channel is for the vocabulary
- File or pipe mode



Neural Topic Model: How is it used?

- You define how many topics you want
- These topics are a latent representation based on top ranking words
- One of two topic modeling algorithms in SageMaker – you can try them both!

Neural Topic Model: Important Hyperparameters

- Lowering mini_batch_size and learning_rate can reduce validation loss
  - At expense of training time
- Num_topics

Neural Topic Model: Instance Types

- GPU or CPU
  - GPU recommended for training
  - CPU OK for inference
  - CPU is cheaper

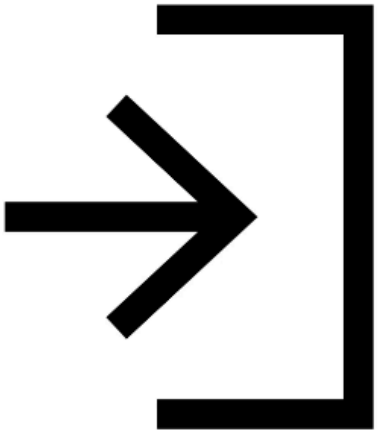
LDA (Latent Dirichlet Allocation) in SageMaker

LDA: What's it for?

- Latent Dirichlet Allocation
- Another topic modeling algorithm
  - Not deep learning
- Unsupervised
  - The topics themselves are unlabeled; they are just groupings of documents with a shared subset of words
- Can be used for things other than words
  - Cluster customers based on purchases
  - Harmonic analysis in music

LDA: What training input does it expect?

- Train channel, optional test channel
- recordIO-protobuf or CSV
- Each document has counts for every word in vocabulary (in CSV format)
- Pipe mode only supported with recordIO



LDA: How is it used?

- Unsupervised; generates however many topics you specify
- Optional test channel can be used for scoring results
  - Per-word log likelihood
- Functionally similar to NTM, but CPU-based
  - Therefore maybe cheaper / more efficient

LDA: Important Hyperparameters

- Num_topics
- Alpha0
  - Initial guess for concentration parameter
  - Smaller values generate sparse topic mixtures
  - Larger values (>1.0) produce uniform mixtures

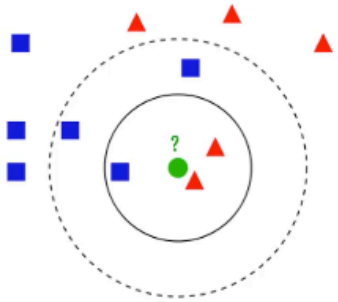
LDA: Instance Types

- Single-instance CPU training

KNN (K-Nearest Neighbors) in SageMaker
World simplest algorithm!
Being a **supervised classification algorithm**, K-nearest neighbors needs labelled data to train on.
With the given data, KNN can classify new, unlabelled data by analysis of the k number of the nearest data points. Thus, the variable k is considered to be a parameter that will be established by the machine learning engineer.

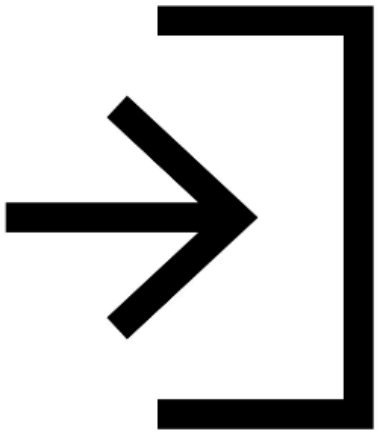
KNN: What's it for?

- K-Nearest-Neighbors
- Simple classification or regression algorithm
- Classification
  - Find the K closest points to a sample point and return the most frequent label
- Regression
  - Find the K closest points to a sample point and return the average value



KNN: What training input does it expect?

- Train channel contains your data
- Test channel emits accuracy or MSE
- recordIO-protobuf or CSV training
  - First column is label
- File or pipe mode on either



KNN: How is it used?

- Data is first sampled
- SageMaker includes a dimensionality reduction stage
  - Avoid sparse data ("curse of dimensionality")
  - At cost of noise / accuracy
  - "sign" or "fjlt" methods
- Build an index for looking up neighbors
- Serialize the model
- Query the model for a given K

KNN: Important Hyperparameters

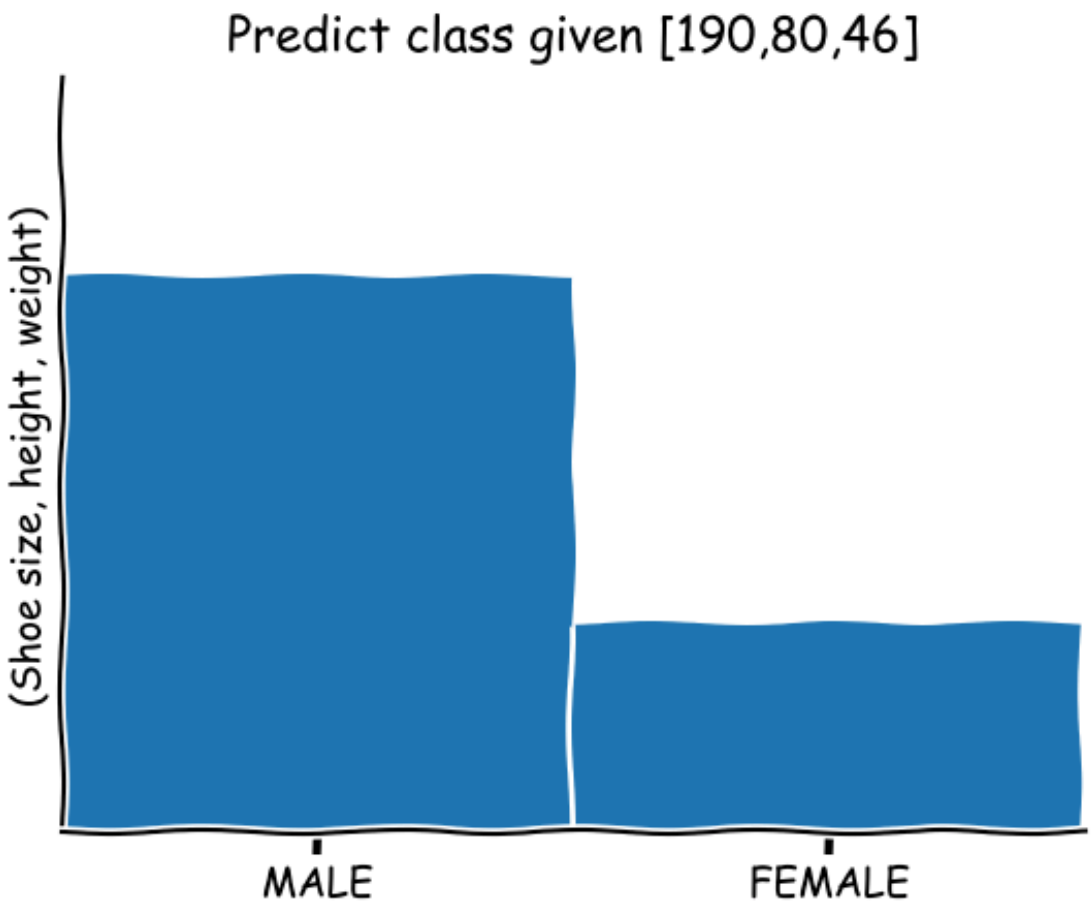
- K!
- Sample_size

K = how many neighbors to use

KNN: Instance Types

- Training on CPU or GPU
  - Ml.m5.2xlarge
  - Ml.p2.xlarge
- Inference
  - CPU for lower latency
  - GPU for higher throughput on large batches

Example where K-NN can be used

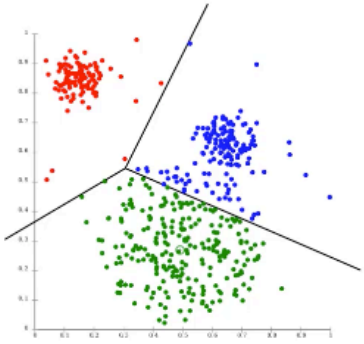


K-Means Clustering in SageMaker

K-means clustering represents an **unsupervised clustering algorithm** that needs unlabelled data to train.

K-Means: What's it for?

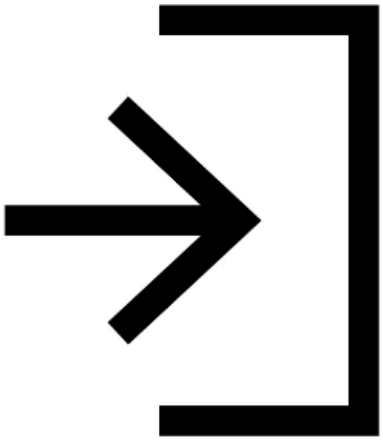
- Unsupervised clustering
- Divide data into K groups, where members of a group are as similar as possible to each other
  - You define what "similar" means
  - Measured by Euclidean distance
- Web-scale K-Means clustering



Different than K-NN that was supervised.
K-Means is UNSUPERVISED.

K-Means: What training input does it expect?

- Train channel, optional test
  - Train ShardedByS3Key, test FullyReplicated
- recordIO-protobuf or CSV
- File or Pipe on either



K-Means: How is it used?

- Every observation mapped to n-dimensional space (n = number of features)
- Works to optimize the center of K clusters
  - "extra cluster centers" may be specified to improve accuracy (which end up getting reduced to k)
  - K = k*x
- Algorithm:
  - Determine initial cluster centers
    - Random or k-means++ approach
    - K-means++ tries to make initial clusters far apart
  - Iterate over training data and calculate cluster centers
  - Reduce clusters from K to k
    - Using Lloyd's method with kmeans++

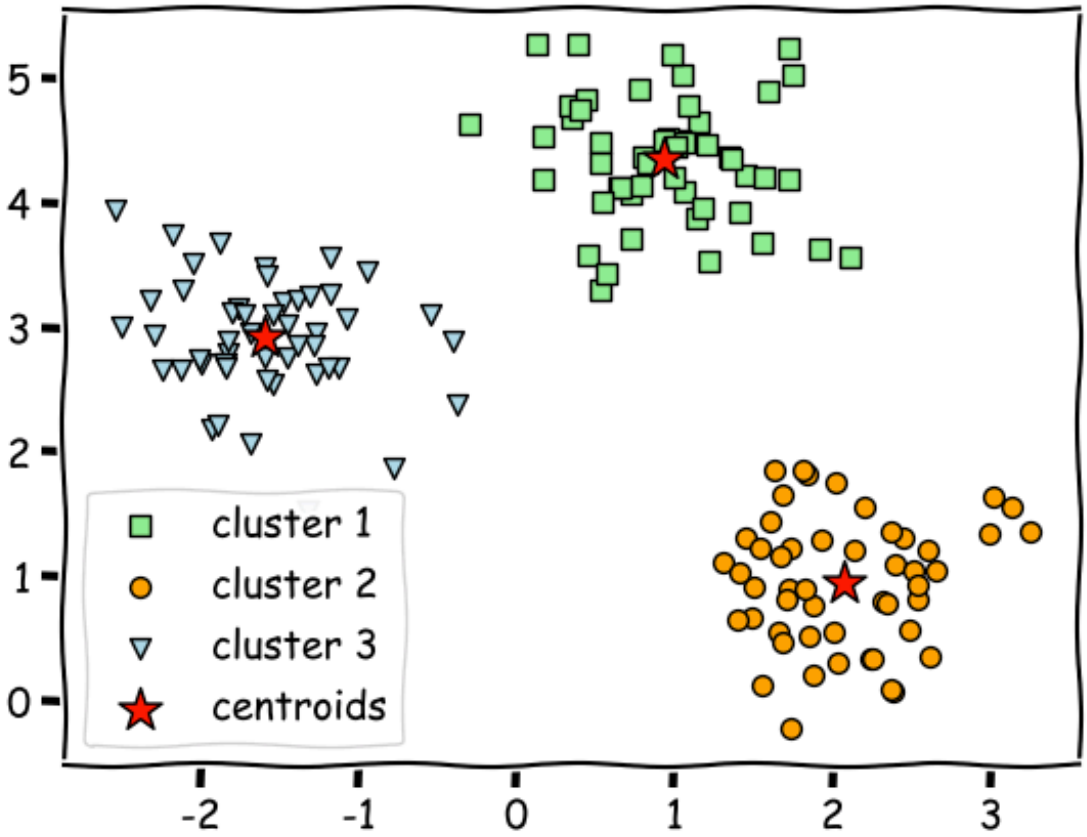
K-Means: Important Hyperparameters

- K!
  - Choosing K is tricky
  - Plot within-cluster sum of squares as function of K
  - Use "elbow method"
  - Basically optimize for tightness of clusters
- Mini_batch_size
- Extra_center_factor
- Init_method

K-Means: Instance Types

- CPU or GPU, but CPU recommended
  - Only one GPU per instance used on GPU
  - So use p*.xlarge if you're going to use GPU

Example:

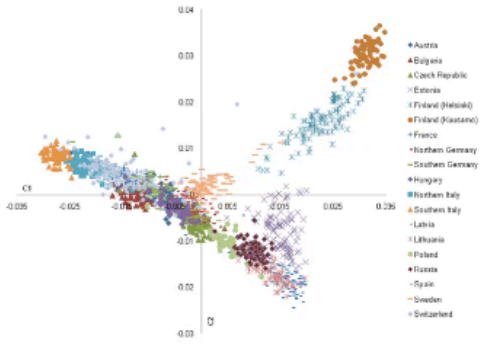


K-means clustering is able to gradually learn how to cluster the unlabelled points into groups by analysis of the mean distance of said points. In this case, the variable k depicts the number of clusters or different groups in which the data will be gathered. The algorithm functions by moving the data in such manner that error function is minimized.

PCA (Principal Component Analysis) in SageMaker

PCA: What's it for?

- Principal Component Analysis
- Dimensionality reduction
  - Project higher-dimensional data (lots of features) into lower-dimensional (like a 2D plot) while minimizing loss of information
  - The reduced dimensions are called components
    - First component has largest possible variability
    - Second component has the next largest…
- Unsupervised



PCA: What training input does it expect?

- recordIO-protobuf or CSV
- File or Pipe on either

PCA: How is it used?

- Covariance matrix is created, then singular value decomposition (SVD)
- Two modes
  - Regular
    - For sparse data and moderate number of observations and features
  - Randomized
    - For large number of observations and features
    - Uses approximation algorithm

PCA: Important Hyperparameters

- Algorithm_mode
- Subtract_mean
  - Unbias data

PCA: Instance Types

- GPU or CPU
  - It depends "on the specifics of the input data"

Factorization Machines in SageMaker
for Classification or regression with **SPARSE Data**
Typically used in Item **recommendation** (where we have a huge number of items).
We only know a few things about a user

Factorization Machines: What's it for?

- Dealing with sparse data
  - Click prediction
  - Item recommendations
  - Since an individual user doesn't interact with most pages / products the data is sparse
- Supervised
  - Classification or regression
- Limited to pair-wise interactions
  - User -> item for example

Factorization Machines: What training input does it expect?

- recordIO-protobuf with Float32
  - Sparse data means CSV isn't practical

Factorization Machines: How is it used?

- Finds factors we can use to predict a classification (click or not? Purchase or not?) or value (predicted rating?) given a matrix representing some pair of things (users & items?)
- Usually used in the context of recommender systems

Factorization Machines: Important Hyperparameters

- Initialization methods for bias, factors, and linear terms
  - Uniform, normal, or constant
  - Can tune properties of each method

Factorization Machines: Instance Types

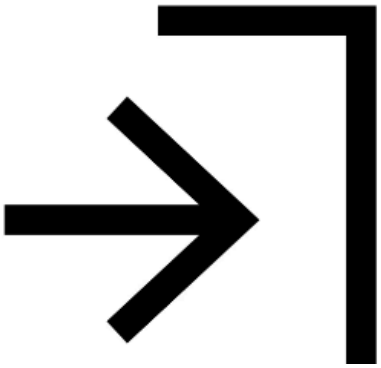
- CPU or GPU
  - CPU recommended
  - GPU only works with dense data

IP Insights: What's it for?

- Unsupervised learning of IP address usage patterns
- Identifies suspicious behavior from IP addresses
  - Identify logins from anomalous IP's
  - Identify accounts creating resources from anomalous IP's

IP Insights: What training input does it expect?

- User names, account ID's can be fed in directly; no need to pre-process
- Training channel, optional validation (computes AUC score)
- CSV only
  - Entity, IP



IP Insights: How is it used?

- Uses a neural network to learn latent vector representations of entities and IP addresses.
- Entities are hashed and embedded
  - Need sufficiently large hash size
- Automatically generates negative samples during training by randomly pairing entities and IP's

IP Insights: Important Hyperparameters

- Num_entity_vectors
  - Hash size
  - Set to twice the number of unique entity identifiers
- Vector_dim
  - Size of embedding vectors
  - Scales model size
  - Too large results in overfitting
- Epochs, learning rate, batch size, etc.

IP Insights: Instance Types

- CPU or GPU
  - GPU recommended
  - Ml.p3.2xlarge or higher
  - Can use multiple GPU's
  - Size of CPU instance depends on vector_dim and num_entity_vectors