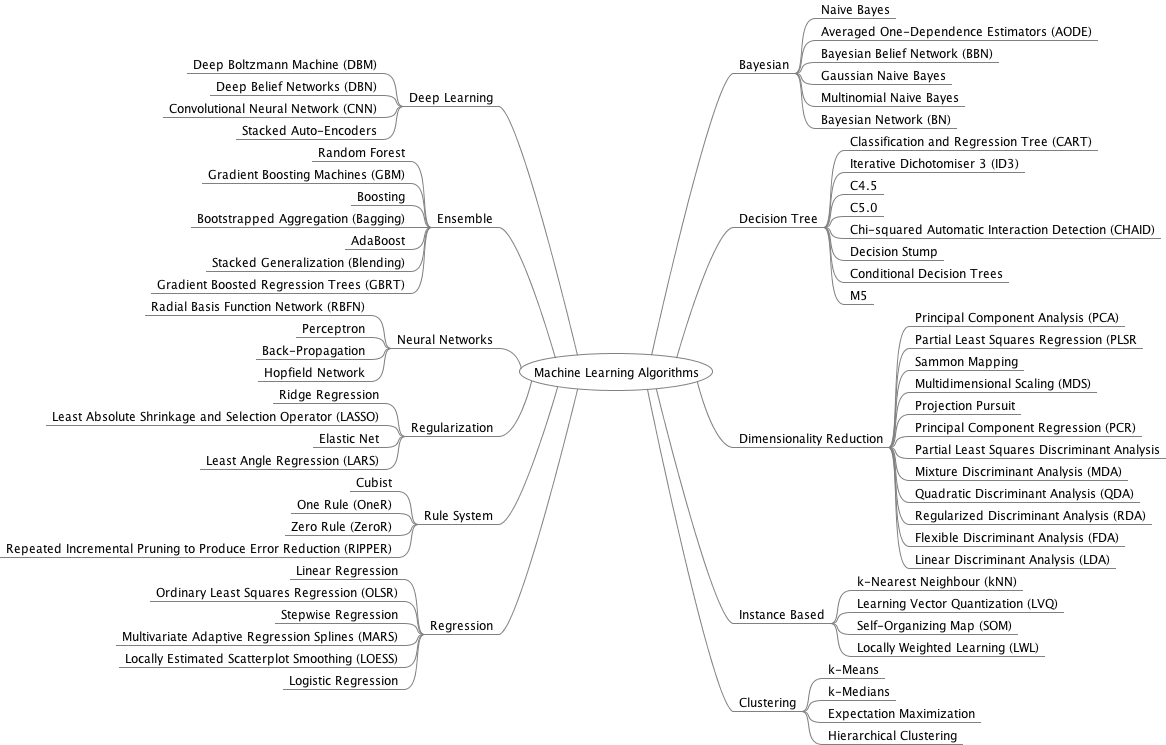


ml nOTES

Data Science and Machine Learning



Machine Learning Algorithms

Contents

[Machine Learning Algorithms 0](#_Toc26796638)

[Artificial Intelligence 3](#_Toc26796639)

[Types of ML Models: 3](#_Toc26796640)

[Learning Styles (ML Learning) 3](#_Toc26796641)

[Supervised Learning 3](#_Toc26796642)

[Example Problems: 3](#_Toc26796643)

[Example Algorithms: 3](#_Toc26796644)

[UnSupervised Learning 3](#_Toc26796645)

[Example Problems: 3](#_Toc26796646)

[Example Algorithms: 3](#_Toc26796647)

[Reinforcement (Semi-Supervised) Learning 3](#_Toc26796648)

[Example Problems: 3](#_Toc26796649)

[Example Algorithms: 3](#_Toc26796650)

[Algorithms 4](#_Toc26796651)

[Regression Algorithms 4](#_Toc26796652)

[Instance-Based Algorithms 4](#_Toc26796653)

[Regularization Algorithms 4](#_Toc26796654)

[Decision Tree Algorithms 4](#_Toc26796655)

[Bayesian Algorithms 5](#_Toc26796656)

[Clustering Algorithms 5](#_Toc26796657)

[Association Rule Learning Algorithms 5](#_Toc26796658)

[Artificial Neural Network Algorithms 5](#_Toc26796659)

[Deep Learning Algorithms 5](#_Toc26796660)

[Dimensionality Reduction Algorithms 6](#_Toc26796661)

[Ensemble Algorithms 6](#_Toc26796662)

[Other Algorithms 6](#_Toc26796663)

[Deep Learning 7](#_Toc26796664)

[Neural Networks 7](#_Toc26796665)

[Problems with Deep Neural Networks 8](#_Toc26796666)

[Neural Network types: 9](#_Toc26796667)

[Feed Forward Neural Network (FFNN) 9](#_Toc26796668)

[Recurrent Neural Network (RNN) 9](#_Toc26796669)

[Convolutional Neural Network (CNN): 9](#_Toc26796670)

[Pretrained Networks 10](#_Toc26796671)

[Computer Vision – Pretrained CNNs 10](#_Toc26796672)

[Text- sequence to sequence networks 10](#_Toc26796673)

[More information: 11](#_Toc26796674)

[Definitions 11](#_Toc26796675)

[Concepts 11](#_Toc26796676)

[Statics 15](#_Toc26796677)

[Probability 15](#_Toc26796678)

[Hypothesis and Inference 16](#_Toc26796679)

[Data Processing Steps 16](#_Toc26796680)

[Machine Learning Steps 16](#_Toc26796681)

[Machine Learning Packages 17](#_Toc26796682)

[Deep Learning 10 methods: 17](#_Toc26796683)

[Machine Learning Considerations 17](#_Toc26796684)

[ML Learning Models: 18](#_Toc26796685)

[Flash Cards 18](#_Toc26796686)

[Module installation and notes 19](#_Toc26796687)

[Deep Learning Frameworks: 19](#_Toc26796688)

[Data extractors & Parsers 19](#_Toc26796689)

[APIs 19](#_Toc26796690)

[Helper Functions 19](#_Toc26796691)

[Facial Recognition Packages 20](#_Toc26796692)

[ML Algorithmic Models: 20](#_Toc26796693)

[Data Sets and Portals 20](#_Toc26796694)

[Links: 21](#_Toc26796695)

[Math: 21](#_Toc26796696)

[Machine Learning: 22](#_Toc26796697)

[Git Repositories: 22](#_Toc26796698)

[Online Models 22](#_Toc26796699)

[Tutorials & Courses: 22](#_Toc26796700)

Artificial Intelligence **–** Two Approaches:

**Symbolic AI**: Uses mathematical symbols to represent objects and the relationships between them

**Connectionist AI**: Precursor to Deep Learning

# Types of ML Models:

**Classification:**

* ***Binary Classification***: Prediction is a yes or no where output is in the form of categories

Example algorithm: Logistic Regression

* ***Multi-Class Classification***: One output for more than two possible outcomes.

Ex algorithm: Multinomial Logistic Regression. Is a movie a comedy, documentary, or Thriller?

**Regression** (Linear or Non Linear)

* ***Simple Regression***: (1 input; 1 output) - Predict a numeric value where outcome is based on average of what has occurred. Example algorithm: Linear Regression. e.g. Home Value, Salary
* ***Multivariant Linear Regression*:** (Multiple input; 1 output)– Regression using various data features. i.e. how much will it rain today based on 1. How much it rained yesterday, 2. Is it summer or winter

# Learning Styles (ML Learning)

Supervised Learning– Answer is known, training exposes relationships or weights.

Use labeled training data to learn mapping function of input X to output Y: y = f(x)

Example Problems: Classification and Regression

Example Algorithms: Logistic Regression and Back Propagation Neural Network.

1. **Classification Algorithms** (Linear Classifiers or Clustering algorithms)
2. **Classification & Regression** Algorithms
3. **Ensemble Learning**: A type of supervised learning that combines predictions from multiple models. Voting used for Classification; Average used for Regression

UnSupervised Learning – Finds relationships using known input variables (X) but no output variables.

Training exposes the prediction and outliers. e.g. “Women age 25 from the west coast prefer blue jeans”

Example Problems: Clustering, dimensionality reduction and associated rule learning.

Example Algorithms: Apriori algorithm and k-Means.

1. **Association** – Finds probability of co-occurrence i.e. Someone who buys bread is 80% likely to buy eggs.
2. **Clustering** – Finds similar Data Points and groups objects together
3. **Dimensionality Reduction** – Reduce data set variables while keeping important information.
4. **Visualization** **algorithms** – Input data is output into a 2D or 3D representation that can be easily plotted
5. **Feature Selection** (variable or attribute): Selects a subset of features for model construction
6. **Feature Extraction** – Data transformation: Moves high dimensional space to low dimensional space.
7. **Anomaly Detection -** Finds outliers in a collection of data. Ex: Banks look for fraudulent transactions

Reinforcement (Semi-Supervised) Learning– Feedback given only when learning achieves its goal.

Interacts with its environment learning through trial and error.

Example Problems: classification and regression

Example Algorithms: extensions to methods that make assumptions on how to model unlabeled data.

# Algorithms

Regression Algorithms **-** relationship between variables to make predictions using a model.

|  |  |
| --- | --- |
| * **Ordinary Least Squares Regression** (OLSR) * **Logistic Regression** – Predicts outcomes when you have at least 2 variables. E.g. Predict lung cancer using variables of height, weight, age and smoking history * **Linear Regression:** Y = ax + b (e.g. Rainfall in CM)   Alleviate collinearity amongst variables by using weight penalties.  L1 Norms (lasso regression) versus L2 Norms (ridge regression)   * **Stepwise Regression** * **Multivariate Adaptive Regression Splines** (MARS) * **Locally Estimated Scatterplot Smoothing** (LOESS) | Regression Algorithms |

Instance-Based Algorithms **–** useimportant instances or examples of training data to make the model.

|  |  |
| --- | --- |
| * **K Nearest Neighbor** (**KNN)** – Find K nearest instances similar to new instance.   Similarities are calculated using shortest distance i.e. Euclidean or Hamming  Output most frequent class or mode (**Classification**)  Output mean of outcomes (**Regression**)   * **Learning Vector Quantization** (**LVQ)** – Performs like a KNN with reduced memory requirement. Allows choice of how many training instances and creates a codebook of vectors used to make predictors. Best codebook (nearest neighbor) is found by calculating distance between codebook and the nearest data instance. * **Self-Organizing Map** (SOM) * **Locally Weighted Learning** (LWL) | Instance-based Algorithms |

Regularization Algorithms **-** penalizes models based on complexity, favoring simpler.

|  |  |
| --- | --- |
| * **LASSO Regression** (Least Absolute Shrinkage and Selection Operator)   Adds an absolute value Bias   * **Ridge Regression** - Adds a small squared Bias factor * **Elastic Net** * **Least-Angle Regression** (LARS) | Regularization Algorithms |
|  |  |

Decision Tree Algorithms **-** construct a model of decisions made based on values of attributes in the data.

|  |  |
| --- | --- |
| * *Conditiona***l Decision Trees** – Choose best feature to split upon   **CART** = **C**lassification **A**nd **R**egression **T**rees (i.e. ID3 or C4.5)  Terminal Nodes are Leaf Nodes – Output Variable (Y)  Non-terminal nodes: root node, input variable (X) and internal nodes.   * **Random Forest** – Random selection of feature to split. * **Iterative Dichotomiser 3** (ID3) * **C4.5 and C5.0** (different versions of a powerful approach) * **Chi-squared Automatic Interaction Detection** (CHAID) * **Decision Stump** * **M5** | Decision Tree Algorithms |

Bayesian Algorithms **-** apply Bayes’ Theorem for problems such as classification and regression.

|  |  |
| --- | --- |
| * **Naïve Bayes** –Probability an event occurs given a preceding event occurs. Naïve because it assumes all variables are independent. P(h|d) = (P(d|h) P(h)) / P(d) * **Gaussian Naive Bayes** * **Multinomial Naive Bayes** * **Averaged One-Dependence Estimators** (AODE) * **Bayesian Belief Network** (BBN) * **Bayesian Network** (BN) | Bayesian Algorithms |

Clustering Algorithms **-** Finds similar Data Points and groups objects together.

|  |  |
| --- | --- |
| Classification Algorithms (Linear Classifiers) (e.g. Speech, Handwriting recognition)   * **k-Means** – Groups similar data into clusters. Assigns a data point to the clusters having least distance between its centroid and the data point. * **k-Medians** * **Expectation Maximization** (EM) * **Hierarchical Cluster Analysis** (**HCA)** | Clustering Algorithms |

Association Rule Learning Algorithms **-** Finds probability of co-occurrence.

i.e. Someone who buys bread is 80% likely to buy eggs.

|  |  |
| --- | --- |
| * **Apriori** (from earlier): Generate associations between frequent dataset items X->Y   Principle is that if an itemset is frequent then all of its subsets are frequent.  /🡪 Support = frq(x,y) / N  Rule X->Y --🡪 Confidence = frq(x,y) /frq(x)  \🡪 Lift = Support/(supp(x) + supp(y))   * **Eclat** (**E**quivalence **CLA**ss **T**ransformation) – link([video](https://www.youtube.com/watch?v=oBiq8cMkTCU)) | Assoication Rule Learning Algorithms |

Artificial Neural Network Algorithms

|  |  |
| --- | --- |
| * **Perceptron -** Feedforward NN: Consists of at least 4 parts: 1. Input Layer,   2. Weights and Bias, 3. Net Sum, 4. Activation Function. Two types:   1. Single Layer: Doesn’t contain hidden layers and can learn linear functions. 2. Multi-Layer Perceptron – Has one or more hidden layers.  * **Back-Propagation** * **Hopfield Network** * **Radial Basis Function Network** (RBFN) (PBN) | Artificial Neural Network Algorithms |

Deep Learning Algorithms

|  |  |
| --- | --- |
| * **Neural Networks** – (*See Below*) * **Deep Boltzmann Machine** (DBM) - Precursor to Autoencoders * **Deep Belief Networks** (DBN) * **Convolutional Neural Network** (CNN) - Uses feature detectors also known as “Filters” or “Kernels”.   Used in Computer Vision, image classification, and signal processing.   * **Stacked Auto-Encoders** | Deep Learning Algorithms |

Dimensionality Reduction Algorithms

|  |  |
| --- | --- |
| * **Principal Component Analysis (PCA) -** Reduces the number of variables (“Principle Components”) in the data (Dimensions). * **Kernel PCA** – Uses Kernel Methods * **Principal Component Regression (PCR)** * **Locally Linear Embedding (LLE)** * **t-distributed Stochastic Neighbor Embedding (t-SNE)** * **Partial Least Squares Regression (PLSR)** * **Sammon Mapping** * **Multidimensional Scaling (MDS)** * **Projection Pursuit** * **Linear Discriminant Analysis (LDA)** * **Mixture Discriminant Analysis (MDA)** * **Quadratic Discriminant Analysis (QDA)** * **Flexible Discriminant Analysis (FDA)** | Dimensional Reduction Algorithms |

Ensemble Algorithms **–** Used with supervised learning; combines predictions from multiple models.

|  |  |
| --- | --- |
| * **Bootstrapped Aggregation** (**Bagging**). Parallel learning – Simple Voting.   An algorithm designed to improve stability, reduce variance and accuracy   * **Random Forests** with Bagging.   + **Bagged decision tree**: One strong learner   + **Bagged Random Forest:** Multiple sub optimal learners   **Boot Strap Sampling:** Train multiple nodes on data subsets using same model   * Data generated from random subset of original data * Dataset uses repetition to keep dataset size the same as original * Original dataset used as a test dataset. | Ensemble Algorithms |
| * **Boosting (e.g. Boosted Trees)** – Sequential Learning with weighted Voting. Convert weak learners to strong learners by assigning greater weights to misclassified instances of previous models – Examples:   + **ADABoot** (Adaptative Boost) – Used with short decision trees. Weight each result from previously trained trees where hard to predict data gets more weight. Make predictions on each iteration and assess performance of each tree to determine how to weight each tree to improve accuracy.   + **CATBoost** (Categorial Gradient Boosting) | |
| * **Gradient Boosting Machines** (**GBM**) – **R gbm,**  **XGBoost** (extreme Gradient Boosting) * **Gradient Boosted Regression or Decision** Trees (**GBRT**) – Stochastic gradient boosting machines * **Stacked Generalization (aka Blending or Stacking)** – Combines multiple classifications or regression models via meta-classifiers or meta-regressors (models). Base models trained on complete dataset. Meta Models trained on Base Model as features. | |

# Other Algorithms

Computational intelligence (evolutionary algorithms, etc.)

Computer Vision (CV)

Natural Language Processing (NLP)

Recommender Systems

Reinforcement Learning

Graphical Models

**Adversarial Training** – Include bad data to provide a more robust model

* **Deep Convolutional Generative Adversarial Network (DCGAN)** – To generate physical models you must have an understanding of real-world components and relationships.

**Adversarial Network** – Two networks that fight each other which makes both stronger. 2 types:

1. Discriminator: Tries to make correct prediction from **input** data
2. Generator: Creates data to fool the Discriminator

|  |  |
| --- | --- |
| **Sequence to Sequence (Seq2Seq) Learning** –  Training exposes relationships between sequences. Examples: Language Translations, AI Chatbots |  |

**Support Vector Machines (SVM)** – Creates a separation of classes and finds a dividing line or hyperplane that separates classes

1. Two-step process
2. Create transformation such as W=X2 + Y2 to add a Z-axis where W=Z
3. Transform back into original plane
4. Tuning Parameters

* **Support Vector Regression (SVR) –** SVM used for classification

Deep Learning **–**

**Artificial Neural Network (ANN)** – Computer system with lots of connected nodes (Neurons) arranged in layers passing information from one layer to another.

* **Neural Networks** (see below)
* **Recursive Neural Networks** – Hierarchical Network – Input processed hierarchically in a tree fashion
* **Radical Basis Function Neural Network** – Used in Cybersecurity, healthcare and power restoration
* **Kohonen Self-Organizing Network** –Classification used in healthcare, Image & Speech recognition
* **Modular Neural Network** – Useful when you have several independent NNs working on same task.
* **Deep Neural Network** – Simply more hidden layers

## Neural Networks

Concepts:

* **Neuron** (aka **Node** or **Unit**) **-** The basic unit of computation in a neural network:

|  |  |  |
| --- | --- | --- |
|  | Y=f(w1\*X1 + w2\*X2 +b) | X = input  w = Weight  b = Bias  f = Activation Function |

* **Feature Scaling** – Normalize data during pre-processing
* **Activation Function (**aka **Transfer Function)** – Performs two goals.

1. Introduce non-linearity in output of a neuron. Allows neurons to learn nonlinear representations.
2. Normalizes or scales output to fit requirements for input node. e.g. (0,1) or (-1,1). Two types:
3. Linear – Simplest activation function where no transform is applied at all.

Cannot learn complex mapping functions however they are used in the output layer for networks that predict a quantity (e.g. regression problems).

Equation: f(x) = x, Range: (-α to α)

1. Non Linear – Preferred for training nodes when using complex structures in the data.
2. **Sigmoid** (aka logistic function) – Range: (0 to 1): б(X) = 1 / (1 + exp(-X))

*Sigmoid*refers to shape of the function**,** *Logistic*is the function. Used interchangeably

1. **tanh** or Hyperbolic Tangent function – Range: (-1 to 1): tanh(X) = 2б(2X) – 1

**Note:**

1. Sigmoid and tanh both suffer from the vanishing gradient problem:

In large, deep networks use of these functions fail to receive useful gradient information. The amount of error, used in back propagation, decreases dramatically with each additional layer, given the derivative of the chosen activation function.

1. Why choose Sigmoid over Step function? Because in order to train a neural network, we need to use calculus, such as the chain rule, which requires smooth and/or continuous functions such as Sigmoid. Sigmoid is a good approximation of a step function.
2. **ReLU** (**R**ectified **L**inear **U**nit) – Range: (0 to α): f(X) = max(0, X)

Often referred to as a piecewise linear function or a hinge function.

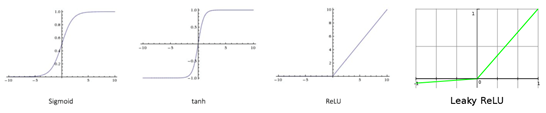
Overcomes the vanishing gradient problem

Outputs the input directly if is positive, otherwise, it will output zero. Code:

if input > 0: return input else: return 0 i.e. g(z) = max{0, z}

1. **Leaky ReLU** (LReLU or LReL) (if leak is not 0.01 then it is random) – Range: (-α to α)

Modifies the function to allow small negative values when the input is less than zero or when the unit is saturated and not active.



1. **Exponential Linear Unit** (ELU) - uses a parameterized exponential function to transition from the positive to small negative values.
2. **Parametric ReLU**, (PReLU) - Learns parameters that control the shape and leaky-ness.
3. **Maxout** - Returns the maximum of the inputs, designed to be used in conjunction with the dropout regularization technique.

* **Back Propagation** – (see Gradient Descent) Walk back error through hidden layers to adjust weights
* **Drop Outs** – Solution to overfitting – Simplify (thin) the network by “dropping” nodes forcing new paths.
* **SoftMax Function** – Range (0–1) to create percentages that add up to one for use in output predictions.
* **Convolutions** – example (input: image, output: edges). Two convolutions second output: lines

|  |  |
| --- | --- |
| Each convolution network looks for a specific pattern. Mathematical formula: |  |

Problems with Deep Neural Networks:

* **Vulnerable to Spoofing** – Adversarial attack
* **Opacity** – Not known how it makes decisions
* **Lack of common sense** – e.g. recognizes chairs & sofas but doesn’t know they are for sitting.

## Neural Network types:

Feed Forward Neural Network (FFNN) –Data only flows in one direction. Has a single input layer and a single output layer, it can have zero or multiple Hidden Layers. Example: Image & Speech recognition

|  |  |
| --- | --- |
| Screen Shot 2016-08-09 at 4.19.50 AM.png | **Input Nodes –** Provide information to the network. No computation is performed.  **Hidden Nodes – P**erform computations and transfer information to the output nodes.  **Output Nodes – A**re responsible for computations and transferring information to the outside world. |

Recurrent Neural Network (RNN) – Add state information into Neural Network. Make stateful

Example: Text to speech, text predictions

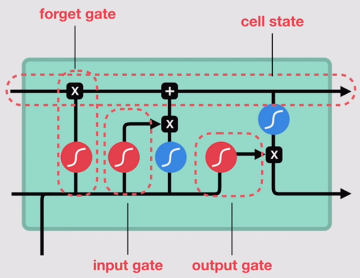
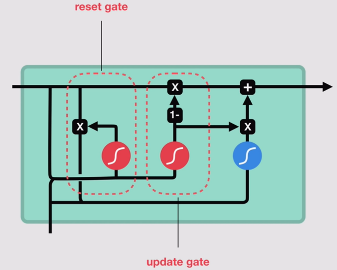
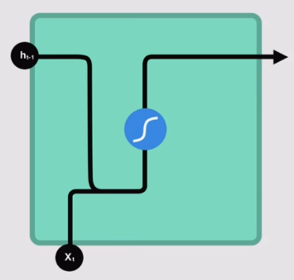
* **LSTM** – Long Short-Term Memory network – A special type of Recurrent Neural Network.

Simulates how the human brain can handle sudden contact switches based on input

* 1. Input Gate - Has control to decide when to let the input enter a neuron
  2. Forget Gate - Has control to decide when to remember what was computed in a previous step
  3. Output Gate - Has control to decide when to let output pass to the next time stamp

Uses relevant information which may be far from where it is needed.

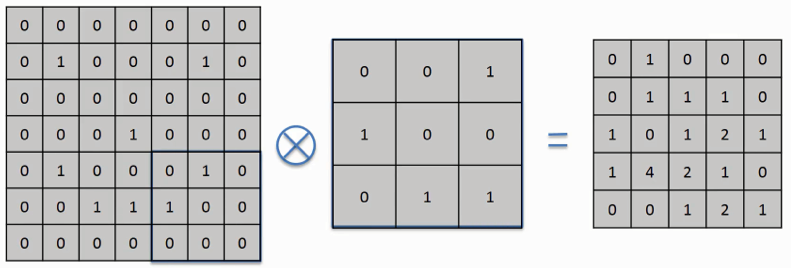
|  |  |
| --- | --- |
|  | Tanh function |
|  | Sigmoid Function |

  **. RNN GRU LSTM**

* Ex: I grew up in France… I speak fluent *French,* fluent; Label video frames. i.e. are characters

Convolutional Neural Network (CNN): Used in Computer Vision, image classification, signal processing

Uses feature detectors also known as “Filters” or “Kernels”



Input Image Feature Detector Feature Map

(Filter or Kernel) (Activation Map)

* 1. Convolutional layer: Use a model to make predictions on an N x N matrix (Usually 2 x 2)
  2. Layer that figures out the most important features from a dataset
  3. Max Pooling - Down sample algorithm. Keep the most interesting prediction of the N x N matrix

Also provides spatial variance needed by the network

* + 1. Down sample an input representation & reduce dimensionality
    2. Apply a max filter to usually nonoverlapping subregions of the initial representation

**Note**: There are multiple pooling types such as Mean, Max and Sum pooling. Max pooling is most common especially with convolutional networks

* 1. Fully Connected – Layer that compares new images to existing data to find make final prediction
* **Graph Convolutional Network**
* **Support Vector Machines** (**SVM**) – Creates separation of classes and finds a line/hyperplane that separates classes.

1. Two-step process
2. Create transformation such as W=X2 + Y2 to add a Z-axis where W=Z
3. Transform back into original plane
4. Tuning Parameters

* **Support Vector Regression (SVR) –** SVM used for classification

**Kernel** **Methods**– **Kernel Functions** (Similarity function): Given two objects a kernel function will output similarity scores

Linear Kernel – Prediction using the dot product between input X and each support vector xi. I.e. f(X) = B(0) + Sum (ai \* (x,xi))

* + Linear: (X1, X2) = X1\* X2
  + Polynomial: k(X1, X2) = (λ X1\* X2 + c)d
  + Gaussian or Radical Basis: k(X1, X2) = exp(λ ІІX1\* X2ІІ2)
  + Sigmoid: : k(X1, X2) = tanh(λ X1\* X2 + c)
* **Gamma –** How far the influence of a single training example reaches (Low meaning Far, High meaning Close)
* **Margin –** Separation of Line to the closest class points

## Pretrained Networks

## Computer Vision – Pretrained CNNs

* Pretrained CNNs:
  + **VGG Net** ([Visual Geometry Group](http://www.robots.ox.ac.uk/~vgg/research/very_deep/)) ([link2](https://arxiv.org/abs/1409.1556))- a deep convolutional network for object recognition. [VGG-16 & VGG-19](https://arxiv.org/pdf/1409.1556.pdf) ,VGG13
  + **AlexNet**
  + **ResNet** (Residual Network) Introduces Residual Learning using shortcut connections Note: ResNet-34 means a 34 layer deep neural Residual Network
  + **Inception**\_v3 - Google
  + **DenseNet**169
  + **Squeezenet1\_1**

### Text- sequence to sequence networks

* **Word2Vec** – Uses a predictive approach to create word embeddings
* **GloVe** (Global Vectors)– Uses a count based approach to create word embeddings

More information: Algorithms:

**Kernel** **Methods (Kernel** or **Similarity** functions): Given two objects a kernel function will output similarity scores

**Linear Kernel**: Prediction using dot product between input and each support vector. f(X) = B(0) + Sum (ai \* (x, xi))

* + Linear: (X1, X2) = X1\* X2
  + Polynomial: k(X1, X2) = (λ X1\* X2 + c)d
  + Gaussian or Radical Basis: k(X1, X2) = exp(λ ІІX1\* X2ІІ2)
  + Sigmoid: : k(X1, X2) = tanh(λ X1\* X2 + c)

**Gamma:** How far the influence of a single training example reaches (Low meaning Far, High meaning Close)

**Margin:** Separation of Line to the closest class points

* Feature selection algorithms
* Algorithm accuracy evaluation
* Performance measures
* Computational intelligence (evolutionary algorithms, etc.)
* Computer Vision (CV)
* Natural Language Processing (NLP)
* Recommender Systems
* Reinforcement Learning
* Graphical Models

# Definitions

* **Gradient** – An increase or decrease in the magnitude of a property – e.g. the slope of a line
* **A Priori -** "from before." If you know how many red, white, and blue gum balls are in the gum ball machine, this a priori knowledge can help you predict the color of the next ones to be dispensed.
* **Stochastic -** having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.
* **Euclidean Algorithm or Distance** - An efficient method for computing the greatest common divisor of two numbers. i.e. the "ordinary" straight-line distance between two points in Euclidean space
* **Hamming Distance** - Distance between two strings or numbers of equal length is the number of positions at which the corresponding symbols are different

# Concepts

* **Adversarial Images -** Images whose class category looks obvious to a human, but causes **massive** **failures** in a deep network.
* **Approximate differentiation** – using finite difference
* [Attention Mechanism](https://skymind.ai/wiki/attention-mechanism-memory-network) – Helps neural networks decide what data to prioritize
* **Autoencoder** ([video](https://www.youtube.com/watch?v=H1AllrJ-_30))– An unsupervised neural network where the target output is the input where it generates a more dense representation (i.e. compression) of the input data

Unlike a PCA algorithm, an autoencoder can learn non-linear transformations with a non-linear activation function and multiple layers.

* + variational autoencoder – Can perform functions of a generative model by learning a latent variable of its input data. Alternative to Generative adversarial network
* **Bayesian Network** (BaysNet) –
* **BIAS** – Shifts activation function up or down
* *Restricted* **Boltzmann Machine (RBM)** – Precursor to Autoencoders, an early NN which consists of a shallow 2-layer net: input layer and hidden layer
* **Bucketing** and **Padding** – For Seq2Seq networks –
  + Split sequence into fixed length buckets,
  + Pad smaller sequences to fit fixed size buckets
* **Chain Rule** - is a [formula](https://en.wikipedia.org/wiki/Formula) for computing the [derivative](https://en.wikipedia.org/wiki/Derivative) of the [composition](https://en.wikipedia.org/wiki/Function_composition) of two or more [functions](https://en.wikipedia.org/wiki/Function_(mathematics)).

For example, from Calculus,: output \* (1- output) \* (output – target)

* **Collinearity** - A condition in which some of the independent variables are highly correlated.
* **Defensive Distillation** – Create second model to mimic first model to defeat adversarial attacks
* **Distance Function (aka Distance Metric) -** 

|  |  |
| --- | --- |
| * **Dot Product – Multiply rows by columns to achieve result** |  |

* [Eigenvalue](https://medium.com/fintechexplained/what-are-eigenvalues-and-eigenvectors-a-must-know-concept-for-machine-learning-80d0fd330e47) **-** The scalar that is used to transform (stretch) an Eigenvector.
* [Eigenvector](https://medium.com/fintechexplained/what-are-eigenvalues-and-eigenvectors-a-must-know-concept-for-machine-learning-80d0fd330e47) - Vectors that, when a linear transformation (i.e. multiplying by a scalar) is performed, does not change its direction. Every vector (list of numbers) has a direction when it is plotted on a XY chart.
* **Encodings** (aka **Embeddings**) – Transform an object into a numerical representation
* **Epoch –** One training cycle through dataset. One complete presentation of the dataset to be learned
* **Fourier Transform** – Break apart complex sound waves into component frequencies
* **Gated Recurrent Unit Neural Network** (GRU) –
* **Generative Query Network Architecture –** Uses two different networks to learn its way around complex virtual environments. Two Networks:

1. Representation Network: Use standard image recognition to identify what is visible
2. Generative Network: Uses Network 1 output to produce a 3D model of the environment to predict objects it can’t see

Example: AI sees a table with one leg occluded and generates a 3D representation of table with missing leg.

* **GNB** (Gaussian Naïve Bayes)
* **Gradient Descent** (called with Neural Networks **Back propagation**) – A method to calculate a partial derivative (or gradient of a function to determine its slope. Two common formulas:
  1. **Analytic differentiation** – computes derivatives using the chain rule
  2. **Approximate differentiation** – using finite difference

|  |  |  |
| --- | --- | --- |
|  | M Gradient |  |
| B Gradient |

* **Graph Learning** –
* **Graph Network training** – A Neural network that takes Graph Relational Database information as input for training to introduce inductive biasing into the neural network.

Induces a BIAS toward representing things as objects and relationships

Example: Paws, tails, whiskers *“is a part of”* A cat

Ball A & Block B *“is next to”* One another

The earth *“in in orbit around”* The Sun

* **He weight initialization (**[**link**](https://arxiv.org/pdf/1502.01852.pdf)**)**: Most used for initializing neural network weights for use with a ReLU activation function. Xavier weight initialization is used with Sigmoid and Tanh activation functions

Formula: F(n) = +/- 2/sqrt(n) where n is the number of nodes in the prior layer.

* **HyperParameter**: Examples: Learning Rate
* **Image processing**:
* **HOG** – Histogram of Oriented Graphics – Face detection using black and white images looking at pixel brightness to create vectors (arrows called gradients) with slope from brighter to darker
* **Face Landmark Estimation** – Measure distances to a number of landmarks on a face
* **Affine Transformation** – Transform rotation and scale preserving parallel lines
* **Loss Functions** ([link](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#kullback-leibler))– Error - How far off is the prediction
  + [**Cross-Entropy**](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy)(aka Negative log likelihood)

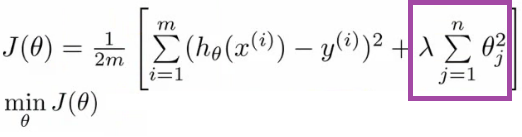
|  |  |
| --- | --- |
| * + **Sum of square Errors** – for each point compute distance from predicted and actual values, then square them, add squares together, and divide by number of points |  |

* + [Hinge](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#hinge)
  + [Huber](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#huber)
  + Kullback-Leibler
  + Mean Absolute Error (MAE) (L1)
  + Mean Square Error (MSE) (L2)
* **Markov Model** - It is assumed that future states depend only on the current state, not on the events that occurred before it (that is, it assumes the [Markov property](https://en.wikipedia.org/wiki/Markov_property))
* **MobileNets** - an architecture for running deep networks on mobile devices. They use a different style of convolutions to reduce both memory consumption and inference time.
* **Model Compression** – Typically accomplished using Teacher-Student Training since student models can have near teacher-level performance, even while using 1-2 orders of magnitude fewer parameters
* **Multi-task Cascaded Convolutional Network** (MTCNN) –
* **Multiple Linear Regression (MLR) -** a statistical technique that looks at how multiple independent variables are related to a dependent variable by modeling the linear relationship between independent variables and a dependent variable using several variables to predict the outcome.
* **Normalization** –

**Min max scaling –** Normalized to values between 0 – 1: z = (x – min(x)) / (max(x) – min(x))

**Batch Normalization** - Eliminate or reduce the effect of outliers. Two Problems:

* 1. Weights Problem – Initial weights may contain many outliers
  2. Learning Weights Problem – Gradients from outliers affect learning weights
* **One-Shot Learning -** aims to learn information about object categories from one, or only a few, training images. Often done using Siamese Network.
* **Optimizers** ([link](https://keras.io/optimizers/)) - Shape and mold a model into most accurate possible form by modifying the weights.
  + Rmsprop – Gradient Descent
  + Adam
* **Overfitting** – 100% prediction on the training data, poor prediction on real data
  + **Regularization** – Adds a penalty as model complexity increases. You can also penalize the loss function by adding more data



|  |  |
| --- | --- |
| * Lasso – Sum the absolute values of weights   (L1 Regularization) | https://cdn-images-1.medium.com/max/1200/1*jgWOhDiGjVp-NCSPa5abmg.png |
| * Ridge – Sum the squared values of weights   (L2 Regularization) |  |

The **key difference** between these techniques is that Lasso shrinks the less important feature’s coefficient to zero thus, removing some feature altogether.

* + **Cross-Validation** – Train on subset of data to validate with rest of data
    - Holdout – Verify predictions using untrained data
    - K Fold – Train multiple models using subsets and perform holdout on each model
  + **Dropout** – Disable nodes in a Neural Network to force different paths during training
* **Performance Metrics**. Consists of:
  + ***Accuracy*** – Percentage of time guess was true: (TP+TN)/(TP+TN+FN+FP)
  + ***Precision*** = True positives / All positive guesses: TP/(TP+FP)
  + ***Recall*** (**Sensitivity**) = True positives / Total positives: TP/(TP+FN)
  + ***Specificity*** = True negatives / Total negatives: (TN/(TN+FP)
  + ***F1 score*** = 2 x (precision x recall)/(precision+recall): 2((p\*r)/(p+r)) = (2\*TP) / (2(TP) + FP + FN)

**TP** = True Positive**, TN** = True Negative**, FP** = False Positive**, FN** = False Negative

* **Pipeline** – A machine learning workflow. Can be automated
* **Siamese Network –** An ANN that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. Often one of the output vectors are precomputed, thus forming a baseline the other output vector are compared against using a distance function.
* [**Sigmoid Function**](https://en.wikipedia.org/wiki/Sigmoid_function) **–** Converts a sum to a probability between 0 and 1 using a mathematical function having a characteristic "S"-shaped or **sigmoid curve**.

[Logistic Function](https://en.wikipedia.org/wiki/Logistic_function): S(x) = 1 / (1 + e-x) = ex / (ex + 1)

[Hyberbolic tangent](https://en.wikipedia.org/wiki/Hyperbolic_function): S(x) = tanh x = (ex - e-x) / (ex + e-x) (shifted & scaled version of the logistic function)

* **Teacher-Student Training -** Teacher and student are trained on dataset. Student it trained using *output of the teacher* (using “soft labels”rather than “hard labels”). Teacher is outputting *class probabilities.*
* **Tensor**: N-Dimensional matrix
  + 0-Dimensional – Scalar
  + 1-Dimensional – Vector
  + 2-Dimenstional - Matrix
* **Transfer Learning** – a method where a model developed for a task is reused as the starting point for a model on a second task. For example, initialized weight values are transferred from a previous model
* **Xavier weight initialization** - Used for initializing neural network weights with Sigmoid or Tanh activation functions. He weight initialization is most common when using a ReLU activation function.

Formula: F(n) = +/- 1/sqrt(n) where n is the number of nodes in the prior layer.

* **Weights** - How much importance or strength to apply to a particular node.
* **Learning Rate Decay** – (aka **Adaptive Learning Rate** or **Learning Rate Annealing**) –

Make large changes at the beginning when large learning rates are used and decrease learning rates later for smaller updates. i.e. training vs tuning. Two popular learning rate decay options:

* 1. Decrease learning rate gradually based on the epoch
  2. Decrease learning rate using punctuated large drops at specific epochs
* **SGD** - (Stochastic Gradient Descent) **–** Pits are called local minima
* **Residual Learning:** Create learning layers using the residual of previous layers. i.e. use the subtraction of features learned from input of that layer.
* **Shortcut connections:** Directly connecting input of nth layer to some (n+x)th layer.
* **Performance Measure** – for regression problems

**RMSE** – (Root Mean Square Error) -

**MAE** – (Mean Absolute Error) -

Statics

**Central Tendencies –** an estimation of where the data is centered. Two methods:

1. **Mean** = average: Sum of the data divided by its count

**Note**: Mean is very sensitive to outliers

1. **Median =** Middle most value if odd # of data points or average of the 2 middle most values if even.
2. **Quantile =** value under which a certain percent of the data lies

(Median represents the value under which 50% of the data lies)

1. **Mode =** Most common value(s)

**Dispersion –** How spread out is the data: (values near zero is not spread out, large values is very spread out)

1. **Range** = Difference between largest and smallest elements
2. **Variance** = Average squared deviation from the mean
3. **Standard Deviation** = Square root of the Variance to bring it back into the same unit as the Range

**Correlation –** Relationship between two metrics

1. **Covariance** = Measures how two variables vary in tandem from their means
2. **Correlation** = Divides Covariance by the Standard Deviation to resolve problems related to units therefore it is unitless and lies between **-1** (perfect anticorrelation) and **1** (perfect correlation)

**Note**: Correlation can be very sensitive to outliers. Note: Correlation is not causation

1. **Simpson’s Paradox** – Correlations assume “all else being equal”, paradox can occur when this assumption is false.

Probability

**Dependent** – two variables are dependent if knowing something about whether variable A happens gives us information about whether variable B happens or vice versa.

**Independent** – There is no relationship between the occurrence of variable A and variable B

**Distributions** -

1. **Discrete distributions** – Associates a positive probability with a discrete outcome. Ex a coin flip
2. **Continuous Distributions** – Distribution across a continuum of outcomes for example:

*Uniform Distribution*– puts equal weight on all data usually between 0 and 1

* 1. **Probability density function (PDF)** – probability of a value being within a certain interval equals the integral of the density function over the interval
  2. **Cumulative distribution function (CDF)** – probability that a random variable is less than or equal to a certain value.

1. **Normal Distribution –** Bell shaped curve with parameters mean µ(mu) and standard deviation σ(sigma)
   1. **Mean** = Where the bell curve is centered
   2. **Standard Deviation** = Width of the bell curve
   3. **Standard Normal distribution** – when mean = 0 and standard deviation = 1
2. **Central Limit Theorem:** A random variable defined as the average of a large number of independent and identically distributed random variables is itself approximately normally distributed.

Hypothesis and Inference

**Null Hypothesis (H0) –** Default position. Ex: Coin toss lands heads 50% of the time (p = 0.5).

**Alternative Hypothesis (H1)** – Position to test. Ex: Coin toss does not land on heads 50% of the time (p ≠ 0.5).

**Type 1 error** (aka **significance or False Positive**) = We reject H0 even though it’s true. Set to 5% or 1%

**Type 2 error** (aka **power or** **False Negative**) = We fail to reject H0 even though it’s false.

**Confidence Interval** = Range around a threshold. i.e. if threshold is 5% then acceptable range around 100 is (95 – 105)

**Bernoulli Trial** (aka **Binomial** (n, p) **trial**): A random experiment with exactly two possible outcomes in which the probability of success is the same every time the experiment is conducted

**p-Value:** Probability we see a value as least as extreme as one we actually observed

**Continuity Correction:** An adjustment that is made when a discrete distribution is approximated by a continuous distribution. Provide a median value between a lower and upper bound of the defined target

**p-hacking**: Testing enough hypotheses against a dataset will find one significant result creating an artificial p-value. For example removing the right outliers can result in the desired p-value.

# Data Processing Steps

1. **Imports –** Preprocessing libraries and frameworks
2. **Read Datasets –** Pandas is most commonly used tool
3. **Create a matrix of variables –** Make compatible with Numpy – can be done at various points in the process. Examples: (Pandas, Numpy)

a**.** Create independent variable matrix (X)

b. Create dependent variable matrix (Y)

1. **Remove or insert missing data –** Examples: (SciKit learn: imputer, Pandas: fillna)
2. **Convert categorical data –** Encode to numeric data. Examples: (SciKit Learn: LabelEncoder, TFLearn: Vocabulary Processor, Keras: Tokenizer
3. **Create Dummy Variables (Labels)** – e.g. Convert resultant categorical numeric data into array of dummy variables. Examples: Keras: ToCategorical, SciKit Learn: HotEncoder
4. **Split data into Training, Test and Validation datasets** – Examples: SciKit Learn: train\_test\_split
5. **Feature scale data** – Example: SciKit Learn: Standard Scalar

**Note**: Without data scaling, the weights of the neural network can grow large, making the network unstable and increasing the generalization error.

1. **Pad Sequence Data** – Examples: TFLearn: Vocabulary Processor, Keras: pad\_sequences
2. **Remove unneeded data (i**.e. Columns) – Examples: Pandas: dataframe.drop([Labels])

# Machine Learning Steps

1. **Problem Definition –** What are we trying to accomplish? What do we want ML task to learn?
2. **Evaluation –** Determine business case and specify how learning task is evaluated
3. **Data Collection –** Find Dataset and properly label input & output data
4. **Data Analysis** (Cleaning and Preprocessing) **–** Format data and decode parameters (features) to use
5. **Training, Validation & Hyperparameter Tuning –** Select proper ML Algorithm & Hyperparameter values.
6. **Testing & Prediction** – Determine effectiveness and accuracy of ML model
7. **Deployment** – Deploy to production

# Machine Learning Packages

* **Numpy** – Mathematical and data handling operations
* **Pandas** – Machine Learning data structures

Panda Series (one dimensional)

Panda Dataframe (two dimensional)

* **Matplotlib** – Visualization Library
* **Seaborn** – (Built on Matplotlib) – High level visualizations
* **SciKit Learn** – ML Modeling
* **SciPy** – Scientific library for advanced operations
* **Keras** – (Biased toward deep learning) Higher level ML API. Can run lower level APIs such as TensorFlow, Theano or CNTK.
* [Theano](http://deeplearning.net/software/theano/) - Define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays.
* [TensorFlow](https://www.tensorflow.org/) – (Biased toward deep learning) Lower level ML API and open source library

# Deep Learning 10 methods:

1. Back Propagation
2. Stochastic Gradient Descent

Stochastic meaning: randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

1. Learning Rate Decay
2. Dropout
3. Max Pooling
4. Batch Normalization
5. Long Short-Term Memory
6. Skip-gram
7. Continuous Bag Of Words
8. Transfer Learning

Machine Learning Considerations

1. Iterate, iterate, iterate – build prototypes to test algorithms, ideas and concepts
2. Use a single evaluation metrics – makes comparisons easier between iterations
3. Error Analysis is critical – find out why you algorithm failed
4. Define an optimal error rate – (Bayes error rate) – Know when you are done

Detect when an algorithm has a high bias or variance

1. Work on problems humans can do well – e.g. speech or image recognition, classification or object detection
2. Training and datasets are split into three groups. Ratios depend on data set size.

Examples 60:20:20 or 50:25:25 or 70:15:15

1. How to split dataset:
   1. Training Dataset – Model is trained on this subset
   2. Development (or Validation) Dataset
      1. Development – Used to tune or optimize the model
      2. Validation – Used to check accuracy of trained model or validate changes
   3. Testing Dataset – Assess final model performance before deployment

# ML Learning Models:

* Supervised Learning Model
* Unsupervised Learning Model
* Semi-supervised Learning Model
* Reinforcement Learning Model
* Nature inspired Learning Model
* Bio-inspired Learning Model
* Statistical Learning Model

# Flash Cards

|  |  |
| --- | --- |
| AI Approaches   1. Symbolic AI 2. Connectionist AI | Unsupervised Learning   1. Associations – comes after 2. Clustering 3. Dimensionality Reduction |
| Adversarial Network   1. Generator   Discriminator | Unsupervised Learning: Associations   1. Apriori – Comes after 2. Eclat – Equivalence Class transformations |
| Ensemble Learning Algorithms   1. Bagging – Parallel Learning   Simple Voting   1. Boosting – Sequential Learning   Weighted Voting   1. Stacking – Uses Meta Classifiers or Meta regressors | Unsupervised Learning: Dimensionality Reduction   1. PCA Reduce number of variables (Principle Components) in the data 2. Kernal PCA 3. LLA – Locally Linear Reduction 4. t-SNE – t-distributed Stochastic Neighbor Embedded |
| Perceptron: Parts   1. Input Layer 2. Weights and Bias 3. NetSum 4. Activation Function | Unsupervised Learning: Clustering   1. K-Means – Group similar data in clusters 2. HCA – Hierarchical Cluster Analysis 3. Expectation Maximizations |
| Activation Function:   1. Sigmoid – Range (0 to 1) 2. Tanh – Hyperbolic Tangent.   Range (-1 to 1)   1. ReLU – Rectified Linear Unit   Range (0 to 1) Neg values = 0  or Range (0 to α)   1. Leakly ReLU – Range (-α to α) 2. ELU – Exponential Linear Unit   Range (-α to α) | LSTM - Long Short Term Memory: decide when to   1. Let input enter a neuron 2. Remember what was computed in previous step 3. Pass output to next time stamp |
| Overfitting   1. L2 Regularization 2. Dropout 3. Early Stopping 4. Cross Validation | Convolutional Neural Network Layers:   1. Convolution Layer 2. Max Pool Layer 3. Fully Connected Layer 4. SoftMax Layer |

# Module installation and notes

1. To view installed modules and versions using command line: pip freeze
2. To view installed modules within Jupyter Notebook: !pip freeze
3. To Install from command line example: python -m pip install beautifulsoup4 requests html5lib
4. To upgrade an existing package using pip: pip install –upgrade tqdm
5. If using anaconda, install using conda install lxml

# Deep Learning Frameworks:

1. [TensorFlow](https://www.tensorflow.org/) w/ TF Learn API – Python – Google
2. Pytorch – Python - Facebook
3. Caffe – C++ - Facebook
4. Sonnet – Google + Tensor Flow
5. Keras – Python (for beginners) + Tensor Flow or CNTK
6. MxNet w/ Gluon– Amazon
7. CNTK (Cognitive Tool Kit) – Microsoft
8. Chainer – Open Source
9. DL4J (Deep Learning 4 Java) – Java
10. CoreML – Apple
11. ONNX – Facebook & Microsoft

# Data extractors & Parsers

* [BeautifulSoup](https://pypi.org/project/beautifulsoup4/) – v4 Python library to extract data from html pages. Documentation: [Beautiful Soup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)
* [Requests Library](https://2.python-requests.org/en/latest/)– v2.22.0 an elegant and simple HTTP library for Python, built for human beings
* [html5lib](https://pypi.org/project/html5lib/) – v1.0.1 HTML parser based on the WHATWG HTML specification
* [pandas-datareader](https://pypi.org/project/pandas-datareader/) - v0.8.1 Remote data access for pandas
* [lxml](https://pypi.org/project/lxml/) - v4.4.2 Powerful XML processing library combining libxml2/libxslt with the ElementTree API.
* [PyTables (tables)](https://pypi.org/project/tables/) – v3.6.1 Hierarchical datasets for Python
* [xlrd](https://pypi.org/project/xlrd/) – v1.2.0 Library for developers to extract data from Microsoft Excel (tm) spreadsheet files
* [openpyxl](https://pypi.org/project/openpyxl/) – 3.0.3 A Python library to read/write Excel 2010 xlsx/xlsm files
* [sqlalchemy](https://pypi.org/project/SQLAlchemy/) – SQL Database Abstraction Library

# APIs

* [GitHub](https://developer.github.com/v3/) – v3 GitHub REST API v3
* [Twython](https://twython.readthedocs.io/en/latest/) - Twitter library. Twitter API [Docs](https://developer.twitter.com/en/docs)

# Helper Functions

* [dateutil](https://dateutil.readthedocs.io/en/stable/) – v2.8.0 - Powerful extensions to datetime
* [tqdm](https://pypi.org/project/tqdm/2.2.3/) – v2.2.3 - A Fast, Extensible Progress Meter

# Facial Recognition Packages

[OpenFace](https://cmusatyalab.github.io/openface/) (system)

[FaceNet](https://arxiv.org/pdf/1503.03832.pdf) (system)

[OpenCV](https://opencv.org/) (library)

# ML Algorithmic Models:

* Linear Regression
* Logistic Regression
* Decision Tree
* SVM
* Naive Bayes
* kNN
* K-Means
* Random Forest

# Data Sets and Portals

* ImageNet
* AlexNet
* [MNIST DATABASE](http://yann.lecun.com/exdb/mnist/) of handwritten digits
* [CIFAR-10](https://www.cs.toronto.edu/~kriz/cifar.html) dataset
* [Caltech-UCSD Birds-200-2011](http://www.vision.caltech.edu/visipedia/CUB-200-2011.html)
* [UC Irvine Machine Learning Repository](http://archive.ics.uci.edu/ml/index.php)
* [Amazon AWS Datasets](https://registry.opendata.aws/)
* [Caltech 101](http://www.vision.caltech.edu/Image_Datasets/Caltech101/): Pictures of objects belonging to 101 categories
* [Caltech 256](http://www.vision.caltech.edu/Image_Datasets/Caltech256/): Collection of all 30607 images 256 categories
* [DAGS](http://dags.stanford.edu/projects/scenedataset.html): stanford background database
* [New York Metropolitan Transportation Authority (MTA)](https://new.mta.info/developers/open-data)

- includes train and bus schedules, service status, elevator and escalator status

* [PlantVillage Dataset](https://drive.google.com/file/d/0B_voCy5O5sXMTFByemhpZllYREU/view)
* [Federal Deposit Insurance Corporation](https://www.fdic.gov/bank/individual/failed/banklist.html) – Failed Bank database
* [ProPublica](https://www.propublica.org/datastore/) ([link](https://github.com/propublica/compas-analysis/blob/master/Compas%20Analysis.ipynb))
* [Beginner Python Machine Learning Handbook](https://drive.google.com/drive/folders/1l2DBl7UaHR3mF1dh3sF88EdC08WO-ora) datasets and notebooks
* [Kaggle Public Datasets](https://www.kaggle.com/datasets)
* [Tensor sample Datasets](https://www.tensorflow.org/api_docs/python/tf/keras/datasets)
* [Wikipedia List of datasets for machine learning research](https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research)
* [Quora List of public datasets](https://www.quora.com/Where-can-I-find-large-datasets-open-to-the-public)
* [Reddit datasets](https://www.reddit.com/r/datasets) (request data)
* [Google dataset search](https://toolbox.google.com/datasetsearch)
* [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/index.php?fbclid=IwAR09F5grBOTCr1SS4v8gONEYeqk0DqqWpPdt1blmYF9ucZkhsQeM5T0E7ew),
* <http://dataportals.org/>
* <https://opendatamonitor.eu/frontend/web/index.php?r=dashboard%2Findex>
* <https://www.quandl.com/>
* Anne Bonner (bonn0062) - [GitHub](https://github.com/bonn0062?tab=repositories)
* Matthew Stewart, PhD Researcher – [articles on ML](https://medium.com/@matthew_stewart)

Machine Learning Model Image: Username: deeplearning PWD: deeplearning

* Python 3.5
* Open CV 3.2 w/ Python Bindings
* dlib 19.9
* TensorFlow 1.5 for Python 3
* Keras 2 for Python 3
* Face Recognition for Python 3
* PyCharm community edition

Additional development environment

* Python 3.7.2
* Atom
* Jupyter Notebook
* Matplotlib - is the library you’ll want if you’re going to make charts.
* Numpy - is the library you’ll need for all things mathematical.
* Pandas - is the best tool available for importing and managing datasets.
* Scipy
* Scikit-learn

Beginner – Keres

Production – Tensor Flow

Research – Pytorch or Sonnet

Interoperability – ONNX

IOS - CoreML

AWS – MxNET

Azure – CNTK

Java – DL4J

Machine Learning Training Notes:

Vocabulary Processor

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Links:

## Math:

* Math for Machine Learning (cheatsheets)

1. [Linear Algebra](http://www.souravsengupta.com/cds2016/lectures/Savov_Notes.pdf)
2. [Calculus](http://tutorial.math.lamar.edu/pdf/Calculus_Cheat_Sheet_All.pdf):
3. [Statistics](http://web.mit.edu/~csvoss/Public/usabo/stats_handout.pdf)
4. [Mathematical Symbols](https://www.rapidtables.com/math/symbols/Basic_Math_Symbols.html)

* Books

1. [Mathematical Notation: A Guide for Engineers and Scientists](https://www.amazon.com/Mathematical-Notation-Guide-Engineers-Scientists/dp/1466230525/ref=as_li_ss_tl?ie=UTF8&linkCode=sl1&tag=authordanjeff-20&linkId=66199c1d62fae34e987ea2ab774fb065)
2. [Mathematics: A Very Short Introduction](https://www.amazon.com/Mathematics-Short-Introduction-Timothy-Gowers/dp/0192853619/ref=as_li_ss_tl?ie=UTF8&qid=1483654040&sr=8-1&keywords=mathematics+a+very+short+introduction&linkCode=sl1&tag=authordanjeff-20&linkId=e6b596ba7a39e142147f59b13b25635d)
3. [Mathematics for the Nonmathemetician](https://www.amazon.com/Mathematics-Nonmathematician-Morris-Kline-dp-0486248232/dp/0486248232/ref=mt_paperback?_encoding=UTF8&me=&qid=1483654414)
4. [No bullshit guide to math and physics](https://www.amazon.com/No-bullshit-guide-math-physics/dp/0992001005/ref=as_li_ss_tl?ie=UTF8&qid=1483654535&sr=8-1&keywords=no+bullshit+guide+to+math&linkCode=sl1&tag=authordanjeff-20&linkId=ba9cb39fd47965260d737c3841d5495b): 5th Edition
5. [Algebra Unplugged](https://www.amazon.com/Algebra-Unplugged-Jim-Loats/dp/0962781576/ref=as_li_ss_tl?ie=UTF8&qid=1483654774&sr=8-1&keywords=algebra+unplugged&linkCode=sl1&tag=authordanjeff-20&linkId=ac455a97e93943e3db410a1381017cd1): 1st Edition

* eBook

1. [Deep Learning](http://www.deeplearningbook.org/)

## Machine Learning:

* Neural Networks

1. [Make Your Own Neural Network](https://www.amazon.com/Make-Your-Own-Neural-Network/dp/1530826608/ref=as_li_ss_tl?ie=UTF8&qid=1483655005&sr=8-1&keywords=make+your+own+neural+network&linkCode=sl1&tag=authordanjeff-20&linkId=1d5697a82a1251d8c8247b8d34838347): 1st Edition ([GitHub Link](https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork)) ([Blog](https://makeyourownneuralnetwork.blogspot.com/))

## Git Repositories:

* [data-science-from-scratch](https://github.com/joelgrus/data-science-from-scratch)
* [how\_to\_make\_an\_image\_classifier](https://github.com/llSourcell/how_to_make_an_image_classifier/blob/master/demo.ipynb)
* [MLAlgorithm/Challenge/LinearRegression](https://github.com/amanullahtariq/MLAlgorithm/tree/eca367287f7874e08a790ce0b0c21567e0b38a22/Challenge/LinearRegression)

# Online Models

* [Google numeric online tool](http://scs.ryerson.ca/~aharley/vis/conv/flat.html)

# Tutorials & Courses:

* [Pandas: Short hands-on challenges to perfect your data manipulation skills](https://www.kaggle.com/learn/pandas)
* [Machine Learning Explainability](https://www.kaggle.com/learn/machine-learning-explainability)
* [Tensorflow for Poets](https://petewarden.com/2016/02/28/tensorflow-for-poets/)
* [Recurrent Neural Networks for Artists](http://blog.otoro.net/2017/01/01/recurrent-neural-network-artist/)
* [Learning AI if You Suck at Math](https://hackernoon.com/learning-ai-if-you-suck-at-math-8bdfb4b79037?gi=594c7ab826bd)
* [How to Load, Convert, and Save Images With the Keras API](https://machinelearningmastery.com/how-to-load-convert-and-save-images-with-the-keras-api/)
* [How to Develop a Convolutional Neural Network to Classify Photos of Dogs and Cats (with 97% accuracy)](https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/)
* [A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)
* [How to Fix the Vanishing Gradients Problem Using the Rectified Linear Unit (ReLU)](https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/)
* [How to Implement the Backpropagation Algorithm From Scratch In Python](https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/)
* [Gentle Introduction to Vector Norms in Machine Learning](https://machinelearningmastery.com/vector-norms-machine-learning/)
* [Convolutional Neural Networks (CNN) - superdatascience](https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-plan-of-attack)