DZone

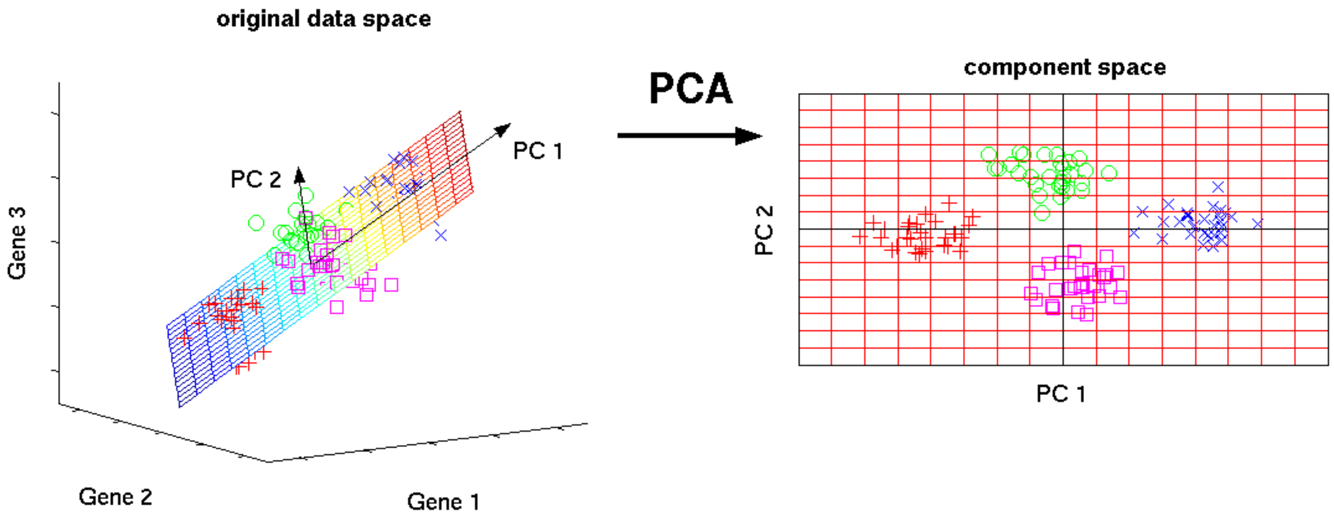
10 Machine Learning Algorithms You Should Know to Become a Data Scientist

by Shashank Gupta MVB · Mar. 25, 18 · AI Zone · Opinion

Machine learning practitioners have different personalities. While some of them are “I am an expert in X and X can train on any type of data,” where X = some algorithm, others are “right tool for the right job” people. A lot of them also subscribe to a “Jack of all trades, master of one” strategy, where they have one area of deep expertise and know slightly about different fields of machine learning. That said, no one can deny the fact that as practicing data scientists, we have to know basics of some common machine learning algorithms, which would help us engage with a new-domain problem we come across. This is a whirlwind tour of common machine learning algorithms and quick resources about them which can help you get started on them.

Principal Component Analysis (PCA)/SVD

PCA is an unsupervised method to understand global properties of a dataset consisting of vectors. Covariance Matrix of data points is analyzed here to understand what dimensions (mostly)/data points (sometimes) are more important (i.e. have high variance amongst themselves, but low covariance with others). One way to think of top PCs of a matrix is to think of its eigenvectors with highest eigenvalues. SVD is essentially a way to calculate ordered components too, but you don’t need to get the covariance matrix of points to get it.



This algorithm helps one fight curse of dimensionality by getting datapoints with reduced dimensions.

Libraries

https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.svd.html

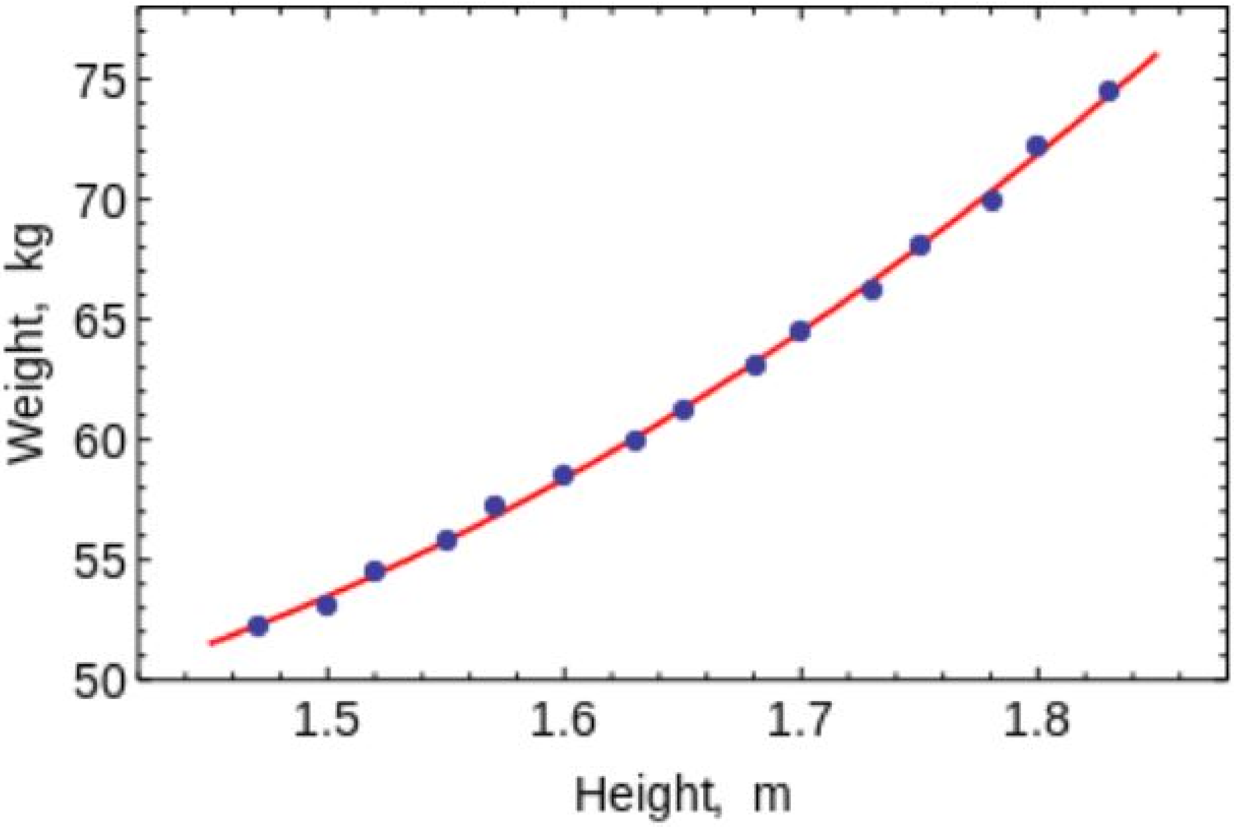
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Introductory Tutorial

https://arxiv.org/pdf/1404.1100.pdf

Least Squares and Polynomial Fitting

Remember your Numerical Analysis course in college, where you used to fit lines and curves to points to get an equation? You can use them to fit curves in machine learning for very small datasets with low dimensions. (For large data or datasets with many dimensions, you might just end up terribly overfitting, so don't bother.) OLS has a closed form solution, so you don't need to use complex optimization techniques.



As is obvious, use this algorithm to fit simple curves/regression.

Libraries

https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.lstsq.htmlhttps://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.polyfit.html

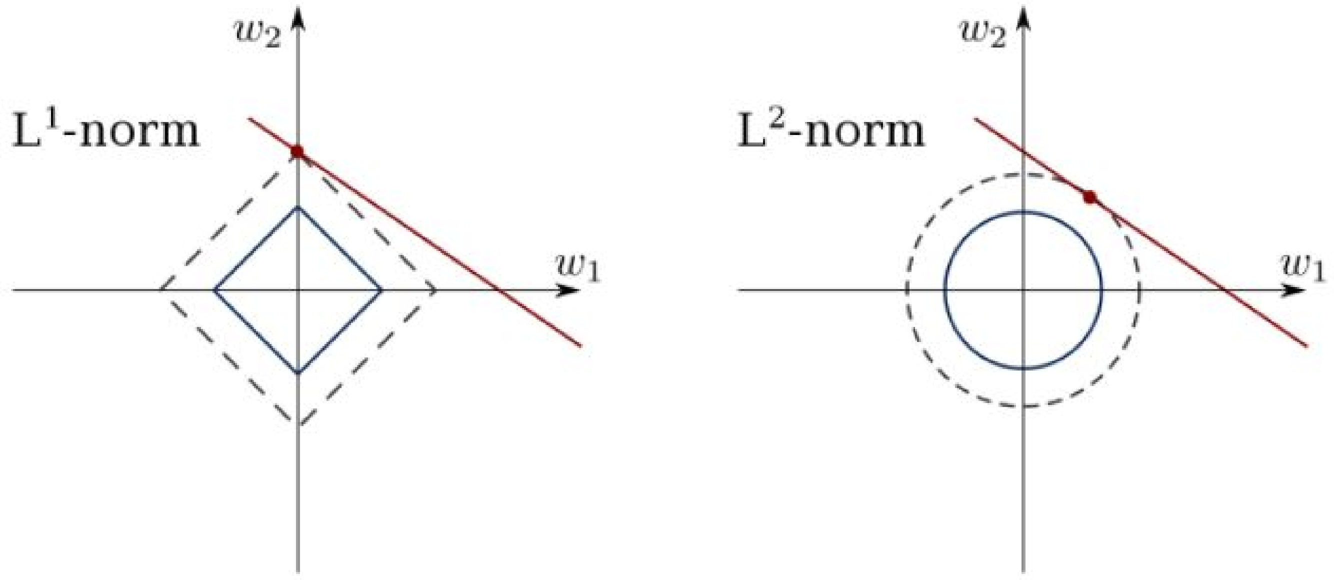
Introductory Tutorial

https://lagunita.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/linear_regression.pdf

Constrained Linear Regression

Constrained Linear Regression

Least Squares can get confused with outliers, spurious fields and noise in data. We thus need constraints to decrease the variance of the line we fit on a dataset. The right method to do it is to fit a linear regression model which will ensure that the weights do not misbehave. Models can have L1 norm (LASSO) or L2 (Ridge Regression) or both (elastic regression). Mean Squared Loss is optimized.



Use these algorithms to fit regression lines with constraints and avoiding overfitting and masking noise dimensions from the model.

Libraries

http://scikit-learn.org/stable/modules/linear_model.html

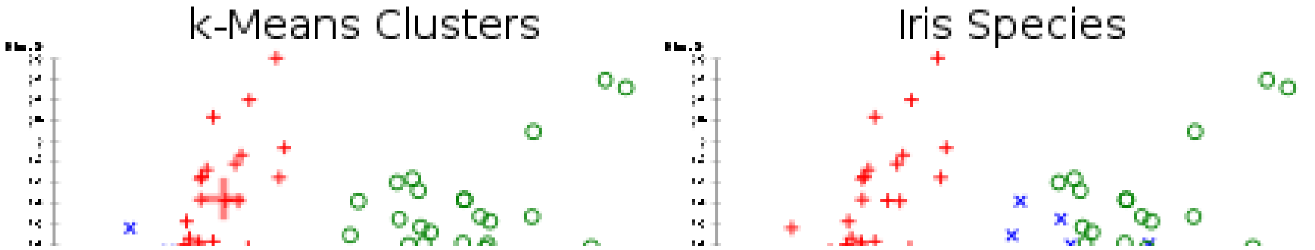
Introductory Tutorials

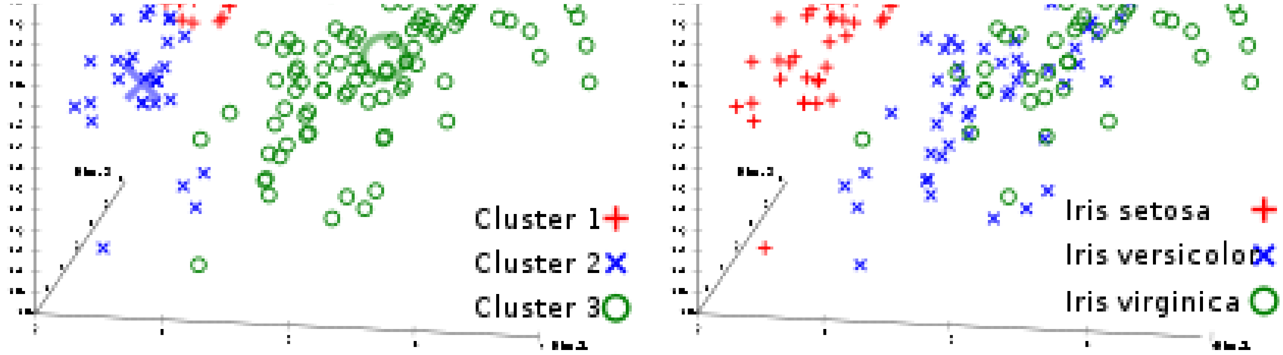
https://www.youtube.com/watch?v=5asL5Eq2x0A

https://www.youtube.com/watch?v=jbwSCwoT51M

K-Means Clustering

Everyone’s favorite unsupervised clustering algorithm. Given a set of data points in form of vectors, we can make clusters of points based on distances between them. It’s an Expectation Maximization algorithm that iteratively moves the centers of clusters and then clubs points with each cluster centers. The input the algorithm has taken is the number of clusters which are to be generated and the number of iterations in which it will try to converge clusters.





As is obvious from the name, you can use this algorithm to create K clusters in the dataset.

Library

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Introductory Tutorials

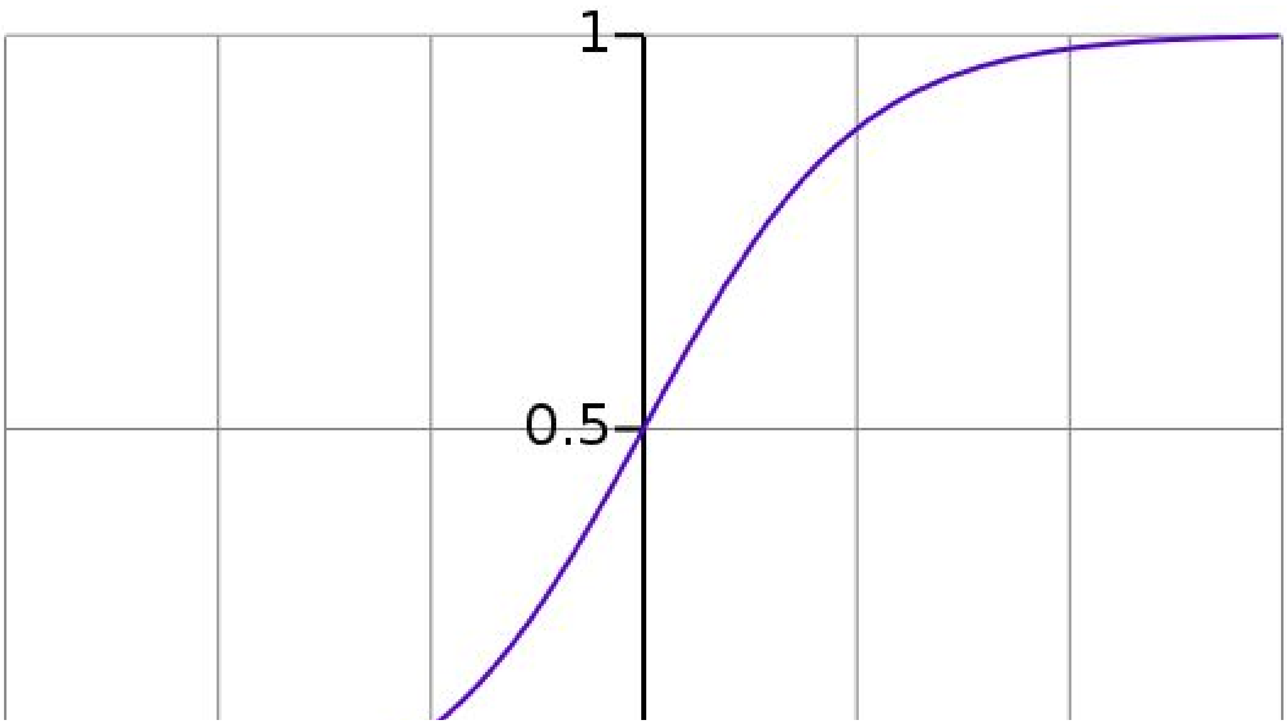
https://www.youtube.com/watch?v=hDmNF9JG3lo

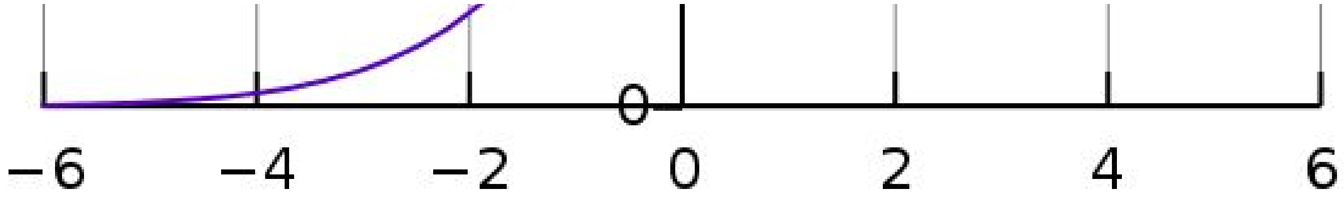
https://www.datascience.com/blog/k-means-clustering

Logistic Regression

Logistic Regression is constrained Linear Regression with a nonlinearity (sigmoid function is used mostly or you can use tanh too) application after weights are applied, hence restricting the outputs close to +/- classes (which is 1 and 0 in case of sigmoid). Cross-Entropy Loss functions are optimized using Gradient Descent. A note to beginners: Logistic Regression is used for classification, not regression. You can also think of Logistic regression as a one layered Neural Network. Logistic Regression is trained using optimization methods like Gradient Descent or L-BFGS. NLP people will often use it with the name of Maximum Entropy Classifier.

This is what a Sigmoid looks like:





Use LR to train simple, but very robust classifiers.

Library

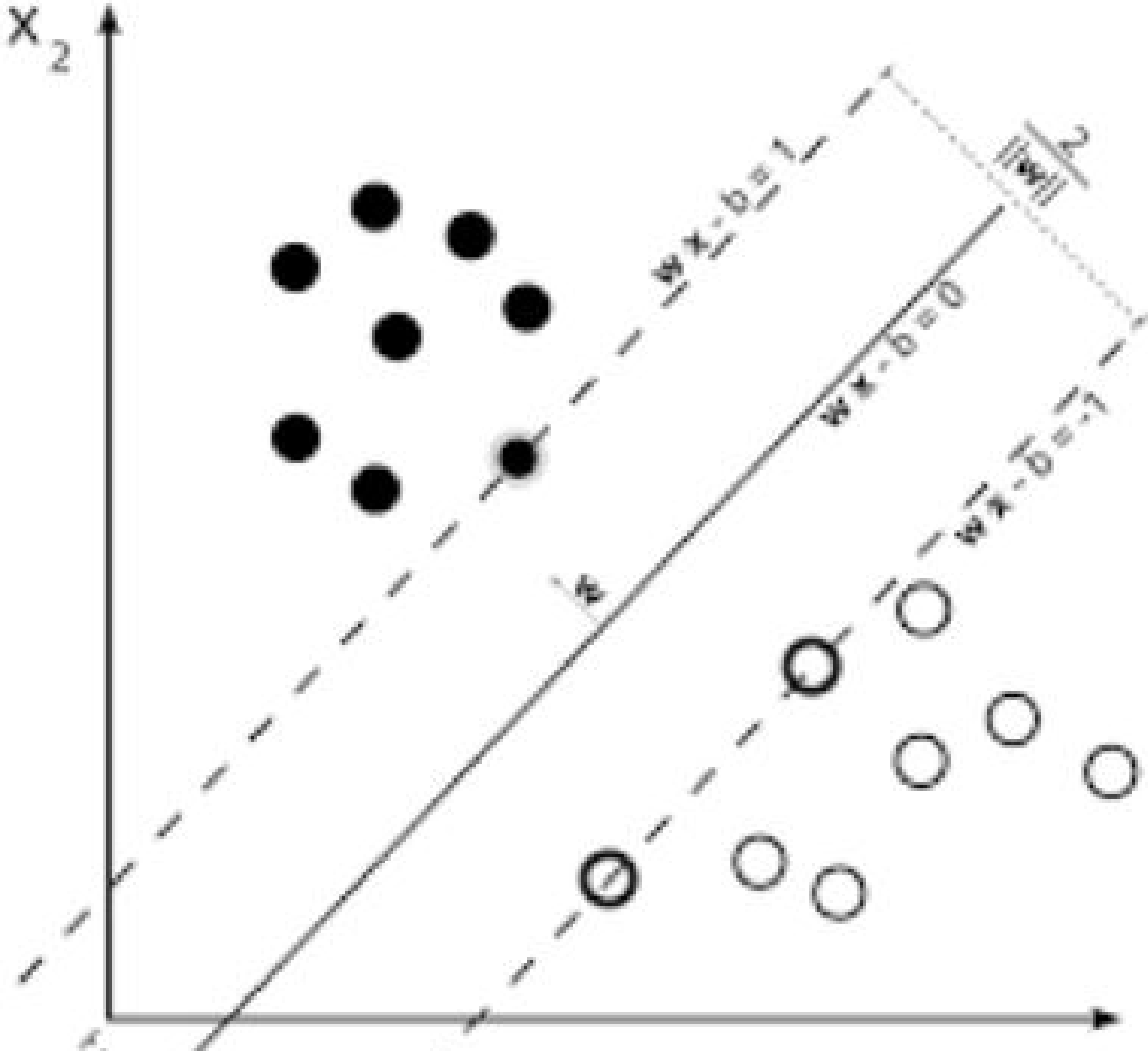
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Introductory Tutorial

https://www.youtube.com/watch?v=-la3q9d7AKQ

SVM (Support Vector Machines)

SVMs are linear models like Linear/Logistic Regression, the difference being that they have different margin-based loss function (the derivation of Support Vectors is one of the most beautiful mathematical results I have seen along with eigenvalue calculation). You can optimize the loss function using optimization methods like L-BFGS or even SGD.



Another innovation in SVMs is the usage of kernels on data to feature engineer. If you have good domain insight, you can replace the good-old RBF kernel with smarter ones and profit.

One unique thing that SVMs can do is learn one class classifiers.

SVMs can be used to train a classifier (even regressors).

Library

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

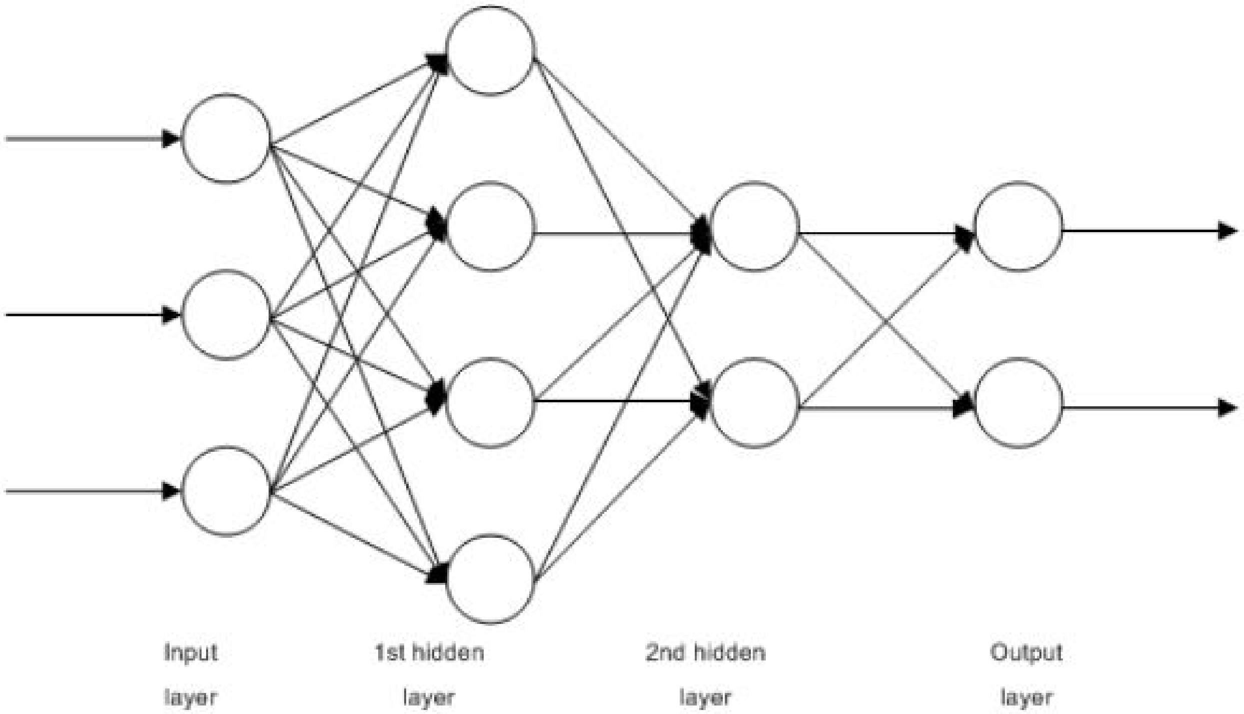
Introductory Tutorial

https://www.youtube.com/watch?v=eHsErlPJWUU

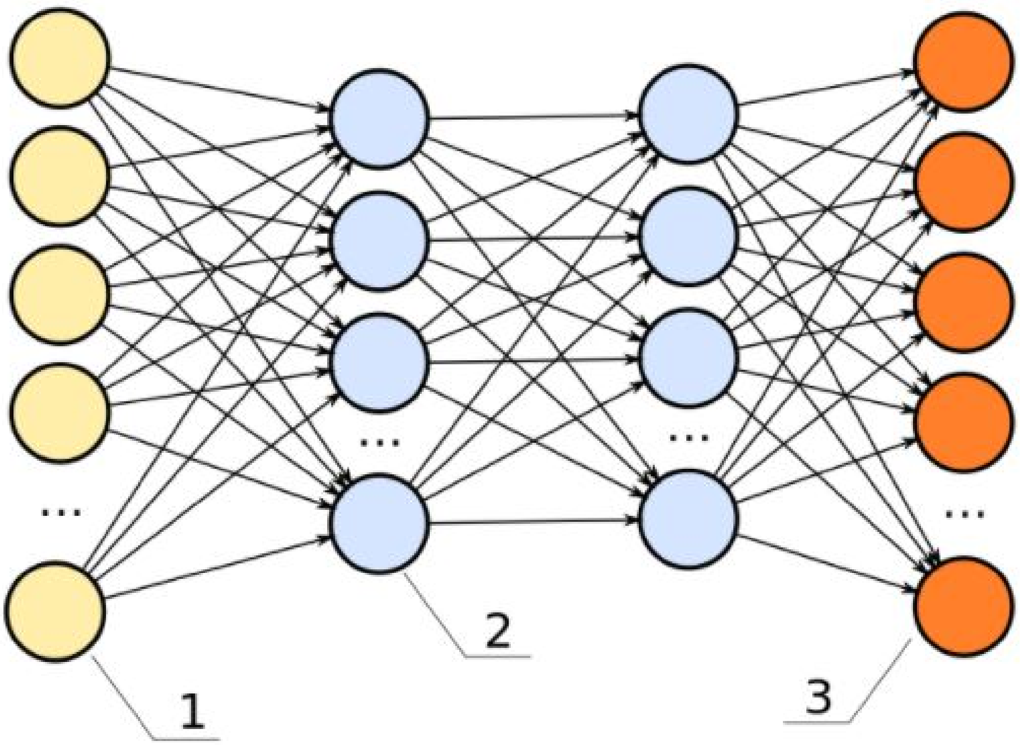
Note: SGD based training of both Logistic Regression and SVMs are found in SKLearn, which I often use as it lets me check both LR and SVM with a common interface. You can also train it on >RAM sized datasets using mini batches.

Feed-Forward Neural Networks

These are basically multilayered Logistic Regression classifiers. Many layers of weights separated by non-linearities (sigmoid, tanh, relu + softmax and the cool new selu). Another popular name for them is Multi-Layered Perceptrons. FFNNs can be used for classification and unsupervised feature learning as autoencoders.



Multi-layered perceptron



FFNN as an autoencoder

FFNNs can be used to train a classifier or extract features as autoencoders.

Libraries

http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

https://github.com/keras-team/keras/blob/master/examples/reuters_mlp_relu_vs_selu.py

Introductory Tutorials

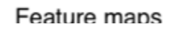
http://www.deeplearningbook.org/contents/mlp.html

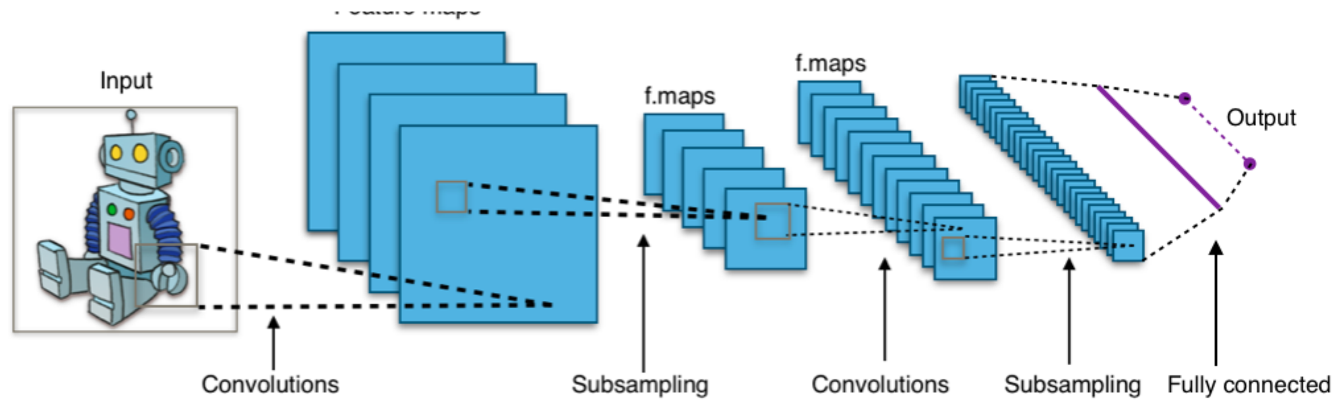
http://www.deeplearningbook.org/contents/autoencoders.html

http://www.deeplearningbook.org/contents/representation.html

Convolutional Neural Networks (Convnets)

Almost any state-of-the-art vision-based machine learning result in the world today has been achieved using Convolutional Neural Networks. They can be used for Image classification, Object Detection or even segmentation of images. Invented by Yann Lecun in late 80s-early 90s, Convnets feature convolutional layers which act as hierarchical feature extractors. You can use them in text too (and even graphs).





Use convnets for state of the art image and text classification, object detection, image segmentation.

Libraries

https://developer.nvidia.com/digits

https://github.com/kuangliu/torchcv

https://github.com/chainer/chainercv

https://keras.io/applications/

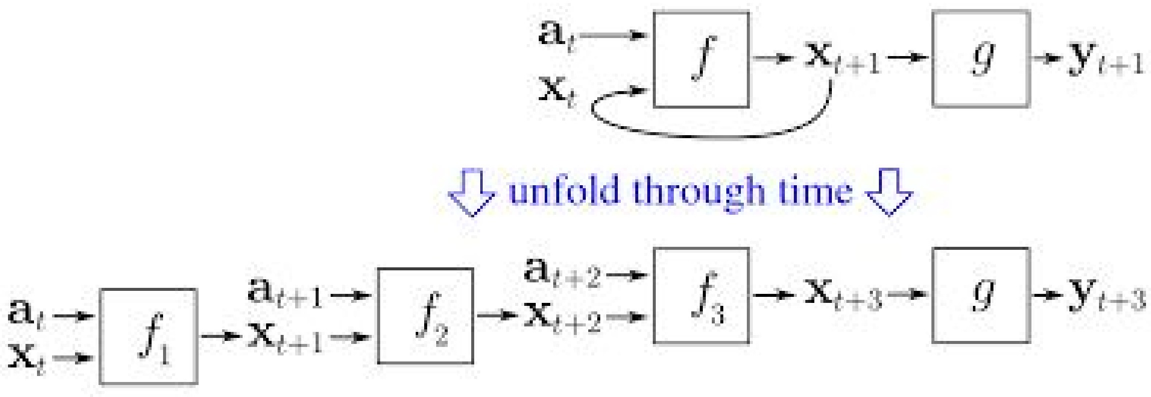
Introductory Tutorials

http://cs231n.github.io/

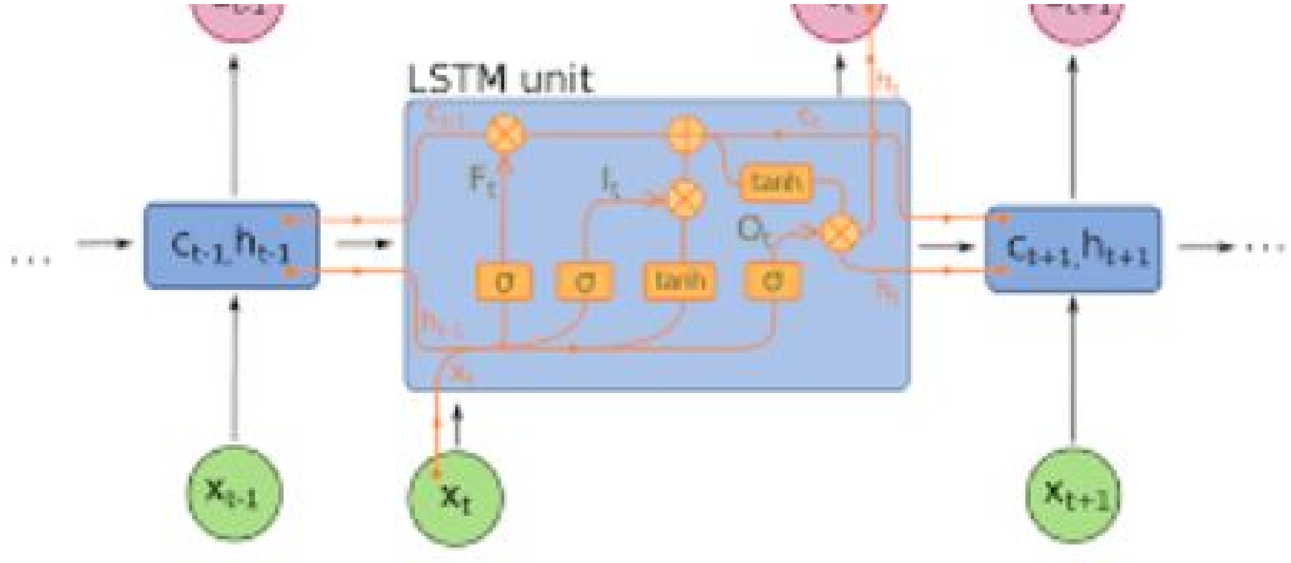
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Recurrent Neural Networks (RNNs)

RNNs model sequences by applying the same set of weights recursively on the aggregator state at a time t and input at a time t (Given a sequence has inputs at times 0..t..T, and have a hidden state at each time t which is output from t-1 step of RNN). Pure RNNs are rarely used now but its counterparts like LSTMs and GRUs are state of the art in most sequence modeling tasks.



RNN (If there is a densely connected unit and a nonlinearity, nowadays f is generally LSTMs or GRUs). LSTM unit which is used instead of a plain dense layer in a pure RNN.



Use RNNs for any sequence modeling task especially text classification, machine translation, and language modeling.

Library:

https://github.com/tensorflow/models (many cool NLP research papers from Google are here)

https://github.com/wabyking/TextClassificationBenchmark

http://opennmt.net/

Introductory Tutorials

http://cs224d.stanford.edu/

http://www.wildml.com/category/neural-networks/recurrent-neural-networks/

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Conditional Random Fields (CRFs)

CRFs are probably the most frequently used models from the family of Probabilistic Graphical Models (PGMs). They are used for sequence modeling like RNNs and can be used in combination with RNNs too. Before Neural Machine Translation systems came in CRFs were the state of the art and in many sequence tagging tasks with small datasets, they will still learn better than RNNs which require a larger amount of data to generalize. They can also be used in other structured prediction tasks like Image Segmentation etc. CRF models each element of the sequence (say a sentence) such that neighbors affect a label of a component in a sequence instead of all labels being independent of each other.

Use CRFs to tag sequences (in Text, Image, Time Series, DNA etc.).

Library:

https://sklearn-crfsuite.readthedocs.io/en/latest/

Introductory Tutorials

Introductory Tutorials

http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/

https://www.youtube.com/watch?v=GF3iSJkgPbA

Decision Trees

Let’s say I am given an Excel sheet with data about various fruits and I have to tell which look like Apples. What I will do is ask a question “Which fruits are red and round ?” and divide all fruits which answer yes and no to the question. Now, All Red and Round fruits might not be apples and all apples won’t be red and round. So I will ask a question “Which fruits have red or yellow color hints on them? ” on red and round fruits and will ask “Which fruits are green and round ?” on not red and round fruits. Based on these questions I can tell with considerable accuracy which are apples. This cascade of questions is what a decision tree is. However, this is a decision tree based on my intuition. Intuition cannot work on high dimensional and complex data. We have to come up with the cascade of questions automatically by looking at tagged data. That is what Machine Learning based decision trees do. Earlier versions like CART trees were once used for simple data, but with bigger and larger dataset, the bias-variance tradeoff needs to solved with better algorithms. The two common decision trees algorithms used nowadays are Random Forests (which build different classifiers on a random subset of attributes and combine them for output) and Boosting Trees (which train a cascade of trees one on top of others, correcting the mistakes of ones below them).

Decision Trees can be used to classify data points (and even regression).

Libraries

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

http://xgboost.readthedocs.io/en/latest/

https://catboost.yandex/

Introductory Tutorials

http://xgboost.readthedocs.io/en/latest/model.html

https://arxiv.org/abs/1511.05741

https://arxiv.org/abs/1407.7502

http://education.parrotprediction.teachable.com/p/practical-xgboost-in-python

TD Algorithms (Good to Have)

If you are still wondering how can any of the above methods solve tasks like defeating Go world champion like DeepMind did, they cannot. All the 10 type of algorithms we talked about before this was Pattern Recognition, not strategy learners. To learn strategy to solve a multi-step problem like winning a game of chess or playing Atari console, we need to let an agent-free in the world and learn from the rewards/penalties it faces. This type of Machine Learning is called Reinforcement Learning. A lot (not all) of recent successes in the field is a result of combining perception abilities of a Convnet or LSTM to a set of algorithms called Temporal Difference Learning. These include Q-Learning, SARSA and some other variants. These algorithms are a smart play on Bellman’s equations to get a loss function that can be trained with rewards an agent gets from the environment.

These algorithms are used to automatically play games mostly, also other applications in language generation and object detection.

Libraries

https://github.com/keras-rl/keras-rl

https://github.com/tensorflow/minigo

Introductory Tutorials

https://web2.qatar.cmu.edu/~gdicaro/15381/additional/SuttonBarto-RL-5Nov17.pdf

https://www.youtube.com/watch?v=2pWv7GOvuf0

These are the ten machine learning algorithms that you can learn to become a data scientist.

You can also read about machine learning libraries here.

Like This Article? Read More From DZone

Artificial Intelligence Will Automate Business Processes

An Intuitive Approach to Deep Learning

Robust Algorithms for Machine Learning

Free DZone Refcard

Introduction to TensorFlow

Topics: MACHINE LEARNING , DATA SCIENCE , AI , ALGORITHM , TUTORIAL

Opinions expressed by DZone contributors are their own.