Intro to Machine Learning for Beginners

[[Link](https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08)]

3 reasons for ML to blow up

* Increase in cheap computation power
* Development of better algorithms
* High amount of data availability

Supervised vs Unsupervised Learning

[[Link](https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d)]

# Supervised Learning:

* Logistic Regression
* Naïve Bayes
* Support Vector Machines
* Artificial Neural Networks
* Random Forests

Main Considerations:

* Model complexity
  + Opt for low-complexity model if:
    - small amount of data, or
    - data is not uniformly spread throughout the different possible scenarios
  + High-complexity model tend to overfit if used on a small number of data points
* Bias-variance tradeoff
  + Balance between bias and variance
    - bias: constant error term
    - variance: amount by which the error may vary between different training sets
  + High bias, low variance
    - consistently wrong 20% of the time
  + Low bias, high variance
    - can be wrong anywhere from 5% - 50% of the time
      * depending on the data used to train it
  + Generally, increasing bias (and decreasing variance) results in models with relatively guaranteed baseline levels of performance
* To produce **models that generalize well**,
  + the variance of your model should scale with
    - **the size** and
    - **complexity** of your *training data*
* Small, simple datasets should usually be learned with low-variance models, and
* Large, complex datasets will require higher-variance models to fully learn the structure of the data.
* Note: Both (model complexity and bias-variance tradeoff) are interrelated.

# Unsupervised Learning:

* Most common tasks
  + Clustering
  + Association
  + Representation Learning
  + Density Estimation
* Algorithms:
  + k-means clustering
  + principal component analysis
  + autoencoders
* No specific way to compare model performance
* Use-cases for unsupervised learning
  + EDA, and
  + Dimensionality Reduction
    - refers to the methods used to represent data using less columns or features.
    - In representation learning
      * we wish to learn relationships between individual features, allowing us to represent our data using the latent features that interrelate our initial features.
      * this sparse latent structure is often represented using far fewer features than we started with, so it can make further data processing much less intensive, and can eliminate redundant features

|  |  |  |
| --- | --- | --- |
|  | Supervised Learning | Unsupervised Learning |
| Discrete | Classification or Categorization | Clustering |
| Continuous | Regression | Dimensionality Reduction |

Batch Learning vs Online Learning

[[Link](https://www.springboard.com/archeio/download/804bb55c63e4426785c9c9d79723f3fe/)]

Batch Learning:

* System cannot learn incrementally, and must be trained using all available data
* The learning process occurs offline (this is known as “offline learning”)
* Data scientists can automate the training, evaluation, and launch of ML systems that use Batch Learning
  + But there are shortcomings to an automated Batch approach

Online Learning:

* Trains the system by breaking the data up into small groups and feeding the system those groups over a longer period.
* The learning is broken up into individually cheap and fast steps, which allows for receiving large amounts of data in real-time.
* Must set the learning rate (i.e., the speed with which the system adapts to changing data)
  + Too high will result in system quickly adapting to the new data at the expense of learning done on previous data
  + Too low will result in system learning new data too slowly to be effective
* With Batch learning, you have relatively high amount of control over the quality of the data your system learns on
* With Online learning, you snooze, you lose

What is Online Machine Learning?

[[Link](https://www.springboard.com/archeio/download/804bb55c63e4426785c9c9d79723f3fe/)]

* Online Learning is also known as **incremental** or **out-of-core learning.**
* Purpose-built incremental learning libraries
  + Vowpal Wabbit
  + crème (scikit-inspired API)
* To react to new data, and make AI that learns over time, ML practitioners typically do one of two things:
  + They manually train on newer data, and deploy the resulting model once they are happy with its performance
  + They schedule training on new data to take place, say, once a week and automatically deploy the resulting model
* There is one glaring problem to this method
  + You want a modelthat **can learn from new examples in something close to real time**

ML Algorithms:   
One SD (σ) – Instance – based Algorithms

[[Link](https://towardsdatascience.com/ml-algorithms-one-sd-%CF%83-instance-based-algorithms-4349224ed4f3)]

* Which algorithm is better for a specific task, depends on the following factors:
  + the size, quality, and nature of data
  + available computational time
  + urgency of the task
  + what do you want to do with the data?
* Good article by same author: ML Algorithms: One SD(σ) [[Link](https://towardsdatascience.com/ml-algorithms-one-sd-%CF%83-74bcb28fafb6)]

Instance-based Algorithms:

* These algorithms do not perform explicit generalization
* They compare new problem instances with instances seen in training, which have been stored in memory
* K-Nearest Neighbor (KNN)
  + **Supervised** Learning Algorithm
  + **Classification** and **Regression** problems
  + KNN stores all available cases and classifies new cases by a majority vote of its K neighbors
  + Predictions are made for a new data point by searching through the entire training set for K most similar instances (the neighbors) and summarizing the output variable for those K instances.
  + Uses Euclidean distances (by default) to calculate “closeness” to determine neighbors
* Learning Vector Quantization (LVQ)
  + **Supervised** Learning Algorithm
  + **Classification** problems
    - both binary and multi-class classification
  + **Artificial Neural Network** Algorithm
  + For K-Nearest Neighbors, you need to hang on to your entire training dataset
    - For LVQ, we can choose how many training instances to hang onto and learn exactly how those instances should look like
    - The number of instances is optimized during the learning process
    - IF KNN gives good results, try using LVQ to reduce the memory requirements
* Self-Organizing Map (SOM)
  + **Unsupervised** Learning Algorithm
  + **Deep Learning Model**
  + Mostly used for :
    - feature detection
    - dimensionality reduction
  + SOM differ from other artificial neural networks
    - as it applies competitive learning as opposed to error-correction learning (like backpropagation with gradient descent)
    - They use a neighborhood function to preserve the topological properties of the input space.
  + SOM performs a topologically ordered mapping from high dimensional space onto two-dimensional space.
    - It produces a two-dimensional representation of the input space of the set of training samples.
  + More confusing stuff, check article
* Locally Weighted Learning (LWL)
  + Instead of building a global model for the whole function space, for each point of interest a local model is created based on neighboring data of the query point
  + In general, data points which are in the close neighborhood to the current query point are receiving a higher weight than data points which are far away
  + LWL methods are **non-parametric**

LeetCode  
/Basic Concepts in ML  
/ Machine Learning 101

[[Link](https://leetcode.com/explore/featured/card/machine-learning-101/)]

# Machine Learning – What

# Machine Learning – How

Underfitting vs. Overfitting

* For supervised learning algorithms, there are two common cases where the generated model does not fit well: underfitting and overfitting
* **Generalization** is an important measure for supervised learning algorithms
  + It measures how well a model derived from the training data can predict the desired attribute of the unseen data.
* For overfitting cases, we can try out another algorithm that generates a simpler model.
  + More often, one stays with the original algorithm that generated the overfitted model, but adds a **regularization** term to the algorithm, i.e., penalizing the model that is over-complicated so that the algorithm is steered to generate a less complicated model while fitting the data.

# Bias vs Variance

* Bias is a learner’s tendency to consistently learn the same wrong thing
* Variance is the tendency to learn random things, unrelated to the real signal

Definitions

## Loss function:

* + *L*(*F*(*xi*​​),*ti*​) = (*F*(*xi*​​) - *ti*)
    - where,
      * *F(xi​​)* is the prediction
      * *ti* is the true value
* The larger the difference, the larger the loss
* If the target attribute *ti* is of **numerical value**, a common loss function would be a **square error**
  + *L*(*F*(*xi*​​),*ti*​) = (*F*(*xi*​​) - *ti*)2

## Main Prediction:

* Given a loss function *L* and sets of training set S {s1, s2… sn}, the main prediction for a learner is defined as
  + *ym = argminy’ES(L(y, y’))*
* For each training set *si*, we train a model *Fi* with a given learner. For a given training sample, we then produce a set of predictions *Y* = {*y1, y2 … yn*} with *yi* corresponding to the result produced by the model *Fi.* The ***main prediction*** *ym* is the prediction y’ whose average loss with regards to all predictions in whose average loss with regards to all predictions in *Y* is minimum (i.e., it is the prediction that “differs least” from all prediction in *Y* according to L)
* More confusion, refer to article

# Machine Learning – Why