Introduction to machine learning alp

(medical diagnosis, quality control, emotion analysis, not safe for work, ...) Make big data decisions (sales forecasts, find crime sites, consumer patterns, ...) Recognize or identify objects or categories (text, faces, tags, language, ...) Automate estimation tasks (geolocation, gender, age, ...) Predict from historical data (estimates, recommendations, logistics, ...) Detecting irregularities in data (data security, fraud detection, system monitoring, ...) (database completion, coloring, depth reconstruction, ...) Automatically tags items in a large dataset (spam filtering, sentiment analysis, image rating,...) Build systems that automatically adapt (market conditions, consumer trends, continuous learning, ...) Identify clusters and trends in data (market segmentation, consumer trends, auto-visualization, ...) Create custom realistic imitations (speech synthesis, voice transfer, face transfer, computer graphics, ...) (handwriting synthesis, painting synthesis, music synthesis, painting from drawing, ...) (mathematics function approaches, stings fluid dynamics, ...) Create built-in intelligent systems (sensor loops, vision systems, ARM / Raspberry Pi, ...) Update classic algorithmic domains using machine learning (signal processing, image processing, computational chemistry, control theory) Standardize freeform or noisy data (autocorrect, sound denoising, image debacle, outer removal, ...) Learn about machine learning (industry, higher education, K-12) Research new machine learning methods and analyses (visualization, domain-specific methods, neural network architectures, ...) and much more... (mathematics function approaches, stings fluid dynamics, ...) Create built-in intelligent systems (sensor loops, vision systems, ARM / Raspberry Pi, ...) (autocorrect, sound denoising, image deblurring, outlier removal, ...) Learn about machine learning (industry, higher education, K-12) Research new machine learning methods and analyses (visualization, domain-specific methods, neural network architectures, ...) Update classic algorithmic domains using machine learning (signal processing, imaging, computational chemistry, control theory) Step 2: Targeted practices after sponge mode, you've probably already got a healthy dose of practice. Now it's time to take that exercise to the next level. Step 2: Targeted practice is all about using specific, conscious exercises to hone your skills. The goal of this step is threefold: Practice the entire machine learning workflow: Data collection, cleaning, and preprocessing. Model building, adjustment and evaluation. Practice real datasets: You start building intuition around the types of models that are appropriate for the types of challenges. Deep dive on individual topics: In step 1, for example, you learned about Algorithms. In step 2, use different types of cluster algorithms on datasets to see which ones work best. After this step, you will be ready to tackle larger projects without feeling overwhelmed. 2.1 - 9 Important topics Machine learning is a wide and rich field. There are applications for almost all industries. It's easy to get confused by all it is to learn. In addition, it is also easy to get lost in the weeds of individual models and lose sight of the big picture. Therefore, we have broken the essential into the following 9 topics. These are building block topics that together represent the simple value for machine learning: taking data and transforming it into something useful. Important ML theory, such as Bias-Variance trade-off. Algorithms to find the best parameters for a model. Handling of missing data, skewed distributions, outliers, etc. How to share your datasets to adjust parameters and avoid over-assembly. Learn from selected data using classification and regression models. Learn from unmarked data using factor and cluster analysis models. Make decisions based on different performance metrics. Combine multiple models for better performance. How machine learning can help different types of businesses. In this step, we strongly recommend that you start with implementations of the algorithm outside the box for two reasons. First, this is how most ML is carried out in the industry. Sure, there will be times when you need to examine original algorithms or develop them from scratch, but prototyping always starts with existing libraries. Second, you get the chance to practice the entire ML workflow without spending too much time on a part of it. This will give you an invaluable big picture intuition. Depending on the programming language you choose, you have 2 good options. Task: Complete the Quick Start Guide for one of the libraries below. Scikit-learn, or sklearn, is the gold standard Python library for general machine learning. It does almost everything, and it has implementations of all the usual algorithms. Scikit-Learn Tutorial, Wine Snob Edition Caret is love. Caret is life. Caret is a library that provides a unified interface for many different model packages in R. It also includes preprocessing, data splitting, and model evaluation features, making it a complete end-to-end solution. Quick Launch Webinar 2.3 - Practice Datasets For this step, you need datasets to practice building and adjusting models. Again, the point of step 2: Targeted practice is to take the theory that floats around in your mind after step 1: Sponge mode and put it in code. Much of the art of computer science and machine learning lies in dozens of micro-decisions you want to make to solve each problem. This is the perfect time to practice making these micro-decisions and evaluating consequences of each. Task: Select 5-10 datasets from the options below. We recommend that you start with the UCI Machine Learning Repository. For example, you can select 3 datasets each for regression, classification, and clusters. Task: For each dataset, you can try at least 3 different modeling approaches using Scikit-Learn or Caret. Consider the following questions: What types of preprocessing do you need to perform for each dataset? Need to reduce dimensions or perform feature choices? If so, what methods can you use? How should you try or share the dataset? How do you know if your model is over-adapted? What types of performance calculations should you use? How do different adjustment parameters affect your model results? Can you ensemble to get better results? (For clusters) Do your clusters look intuitive? We also have a select list of some of our favorite datasets for practices and projects. This is an incredible collection of over 350 different datasets specifically curated to practice machine learning. You can search for task (that is, regression, classification, or clusters), industry, dataset size, and more. (Go to the website) Kaggle.com is best known for hosting computer science competitions, but the site also houses over 180 community datasets for fun topics ranging from Pokemon data to European football matches. (Go to the website) If you're looking for social science or government data sets, look no further than Data.gov, a collection of the U.S. government's open data. You can search over 190,000 datasets. (Go to the website) Artificial intelligence has become widespread recently. People across different disciplines try to use AI to make their tasks much easier. For example, economists use AI to predict future market prices to make money, doctors use AI to classify whether a tumor is malignant or benign, meteorologists use AI to predict the weather, HR recruiters use AI to check the resume of applicants to verify whether the applicant meets the minimum criteria for the job, etcetera. The driving force behind such ubiquitous use of AI is machine learning algorithms. For anyone who wants to learn ML algorithms but hasn't got their feet wet yet, you're in the right place. The rudimental algorithm that every Machine Learning enthusiast starts with is a linear regression algorithm. Therefore, we will do the same as it provides a base for us to build on and learn other ML algorithms. What is linear regression? Before we know what is linear regression, let's get used to regression. Regression is a method of modeling a target value based on independent predictors. This method is mostly used for forecasting and figuring out the cause and effect ratio of variables. Regression techniques vary mostly based on the number of independent variables and the type of relationship between the independent Variables. Linear regressionSimple linear regression is a type of regression analysis in which the number of independent variables is one, and there is a linear relationship between the independent(x) and dependent(y) variable. The red line in the chart above is called the best fit straight line. Based on the given data points, we try to plot a line that models the points best. The line can be modeled based on the linear equation shown below. The motive of the linear regression algorithm is to find the best values for a_0 and a_1. Before moving on to the algorithm, let's take a look at two important concepts you need to know to better understand linear regression. Cost functionThe cost function helps us figure out the best possible values for a_0 and a_1 that will provide the best fit line for the data points. Since we want the best values for a_0 and a_1, we convert this search issue into a minimizing issue where we want to minimize the error between the expected value and the actual value. Minimize and cost functionWe select the function above to minimize. The difference between the expected values and the basic truth measures the error difference. We square the error difference and sum over all data points and divide this value by the total number of data points. This gives the average squared error across all data points. Therefore, this cost function is also called the Average Squared Error (MSE) function. Now, by using this MSE function, we are going to change the values of the a_0 a_1 so that the MSE value settles on the minima. Gradient DescentThe next important concept needed to understand linear regression is gradient descent. Gradient descent is a method of a_0 and a_1 to reduce the cost function (MSE). The idea is that we start with some values for a_0 and a_1, and then we change those values iteratively to reduce costs. Gradient deseeing helps us change the values. Gradient DescentTo draw an analogy, imagine a pit in your form, and you are at the top of the pit and your goal is to reach the bottom of the pit. There is a catch, you can only take a discreet number of steps to reach the bottom. If you decide to take one step at a time you will eventually reach the bottom of the pit, but this would take longer. If you choose to take longer steps each time, you will reach sooner, but there is a chance that you can overshoot the bottom of the pit and not exactly at the bottom. In the gradient descent algorithm, the number of steps you take is the learning frequency. This determines how quickly the algorithm converges to minima. Convex vs Non-convex functionSometimes, the cost function can be a non-convex function where you can settle on a local minima, but for linear regression there is always a convex function. You may be wondering how to use gradient path to update and a_1. If you a_0 and a_1, we'll take gradients from the cost feature. To find these gradients, we partially take derivatives with respect a_0 and a_1. Now, to understand how the partial derivatives are found below, you will require some calculus, but if you do not, that's fine. You can take it as it is. The partial derivatives are the gradients, and they are used to update the values of a_0 and a_1. Alpha is the learning frequency that is a hyperparameter that you must specify. A smaller learning rate can get you closer to minima, but takes more time to reach the minima, a greater learning rate converges earlier, but there's a chance you might overshoot the minima. CodeLet comes to the code. We have two choices, we can either use scikit learning library to import linear regression model and use it directly, or we can write our own regression model based on the equations above. Instead of choosing one among the two, let's do both :)There are many datasets available online for linear regression. I used it from this link. Let's visualize the training and test data. Training(left) and Testing(right) dataLet its start with the simplest of the two methods, that is, to use scikit learning library to build our linear regression model. We use pandas library to read the train and test files. We retrieve the independent (x) and dependent (y) variables, and since we only have one function (x) we transform them so that we can feed them into our linear regression model. We use scikit learn to import linear regression model. we fit the model on the training data and predict the values of the test data. We use R2 score to measure the accuracy of our model. R2 scores on testing dataNow, let's build our own linear regression model from the equations above. We will only use numpy library for calculations and R2 scores for calculations. We initialize the value 0.0 for a_0 and a_1. For 1,000 eras, we calculate the cost, and using the cost we calculate the gradients, and using the gradients we update the values of a_0 and a_1. After 1000 eras, we would have got the best values for a_0 and a_1 and thus we can formulate the best fit straight line. The test kit contains 300 samples, which is why we need to reshape a_0 and a_1 700x1 to 300x1. Now we can only use the equation to predict values in the test set and get the R2 score. R2 score on testing dataWe can observe the same R2 score as the previous method. We also plot the regression line along with the test data points to get a better visual understanding of how good the algorithm works. Regression line – Test dataConclusionLinear regression is an algorithm that all Machine Learning enthusiasts need to know, and it's also the right place to start for people who also want to learn machine learning. It really is a simple but useful algorithm. I hope this article was helpful to you. You.