Neural Network report thoughts

|  |
| --- |
| Neural network, are a broad category of mathematical models, under the umbrella of machine learning, that have some architecture where signals are sent between nodes. Supervised neural networks (SNN are used to determine relationship between a data, and parameter domain. Feedforward neural networks (FNN) are a type of SNN, where feedforward describes how the data flows in one direction from input to output[Bailer-Jones et al., 2001]. FNNs have an input layer where data flows in, some number of hidden layers and an output layer where some new altered data created by the neural network flows out. Each of the non-input layer nodes have 4 features: inputs, input weights, bias and activation function. A node takes inputs from some number of nodes in a layer preceding theirs, the weighting of an input is based on its importance relative to the other weight inputs to this node. The bias aids in optimizing the activation function of that node. The role of the activation function is to combine the other 3 features of the node to create an output. Through training, the neural network alters the weights of inputs and biases in order to sort itself into an algorithm that successfully accomplishes the given task. An example FNN is shown in figure 1.  Figure 1: Feedforward neural network architecture. “…each layer has an extra node that holds a constant value (and has no inputs) and provides an offset to the subsequent layers. These are necessary for the network model to function properly.” Figure by Bailer-Jones et al. (2001).  Previously neural networks have been implemented by Hendriks and Aerts (2019) to do forward modelling on coherent oscillations of intermediate mass stars, as well as Verma et al. (2016) who did backward modelling on solar like oscillations.  [Hendriks and Aerts 2019]  Deep Learning Applied to the Asteroseismic Modeling of Stars with Coherent Oscillation Modes  <https://arxiv.org/pdf/1811.03639.pdf>  [Verma et al. 2016]  Asteroseismic determination of fundamental parameters of sun-like stars using multi-layered neural networks  <https://arxiv.org/pdf/1602.00902.pdf>  [Bailer-Jones et al., 2001]  An introduction to artificial neural networks  <https://arxiv.org/pdf/astro-ph/0102224.pdf> |
| We will create a feedforward neural network, trained on a pre-calculated grid of models that maps fundamental stellar parameters onto observable properties (like those previously mentioned). The FNN will be validated by evaluating the loss function using the mean absolute error (MAE), because it is the preferred method when the data does not have noise, which is true of the grid models. Once properly trained the FNN should be able to do the same job as the grid model at a similar speed but without the systematic error sustained from interpolation between grid points and with better accuracy. |
| Hogg et al. (2016)  machine-learning algorithms to stellar abundances was also employed by da Silva et al. (2012), Ting et al. (2012), Jofré et al. (2017), Anders et al. (2018), Boesso & Rocha-Pinto (2018), and Price-Jones & Bovy (2019). |
| X-ray Study of Spatial Structures in Tycho's Supernova Remnant Using Unsupervised Deep Learning |
| **Deep Learning Applied to the Asteroseismic Modeling of Stars with Coherent**  **Oscillation Modes**  <https://iopscience-iop-org.ezproxye.bham.ac.uk/article/10.1088/1538-3873/aaeeec/pdf>  We develop a novel method based on machine-learning principles to achieve optimal initiation of CPU-intensive  computations for forward asteroseismic modeling in a multi-dimensional parameter space. A deep neural network is trained on a precomputed asteroseismology grid containing about 62 million coherent oscillation-mode frequencies derived from stellar evolution models. These models are representative of the core-hydrogen-burning stage of intermediate-mass and high-mass stars. The evolution models constitute a 6D parameter space and their predicted low-degree pressure- and gravity-mode oscillations are scanned using a genetic algorithm. A software  pipeline is created to find the best-fitting stellar parameters for a given set of observed oscillation frequencies. The proposed method finds the optimal regions in the 6D parameter space in less than a minute, hence providing the optimal starting point for further and more detailed forward asteroseismic modeling in a high-dimensional context.  Our method is open source and can be freely used by anyone  Comparison between asteroseismic data and stellar models is  done by computing the spectrum of oscillation modes predicted  by the theoretical models.  Statistical inference in asteroseismology can conveniently be done from forward modeling. Usually, one adopts a grid-based approach and considers millions of stellar structure models of different evolutionary stage. Applications to low-mass stars in the core-hydrogen-burning stage can be found in the pioneering papers by Miglio & Montalbán (2005) and Quirion et al. (2010), while Brassard et al. (2001) developed a methodology to treat coherent gravity modes in core-helium-burning subdwarf stars.  A prominent application of machine learning (ML) to lowmass star asteroseismology based on damped pressure modes was developed by Bellinger et al. (2016). It is based on the combined use of spectroscopic and asteroseismic input and relies on the principle of random forests.  Here, we focus on intermediate- and high-mass corehydrogen-burning stars in the mass range  2-20 Solar masses with detected coherent oscillation modes. Our aim is to develop and provide an optimal deep-learning tool as a critical start-up ingredient for efficient forward asteroseismic modeling in a +6D parameter space. In particular, we wish to explore the performance of a deep-learning network in order to gain computational efforts, because the “manual” asteroseismic modeling applied to a few stars with coherent modes has so far required months of analysis work, with a high risk of missing local minima in parameter space (see Ausseloos et al. 2004 for an example of such a systematic effort).  This is not practical for future applications to samples of tens to hundreds of stars anticipated by future space missions, such as TESS (Ricker et al. 2016) and PLATO (Rauer et al. 2014).  Future work will consider deep-learning methods for asteroseismology of fast rotators.  Their grid was constructed in the framework of the European FP 7 SpaceInn Project.  It was computed with the open-source code MESA  The grid is based on the same input physics as used and described by Schmid & Aerts (2016); the MESA inlist is provided in that paper and therefore omitted here to avoid repetition.5 In brief, the models are based on the following input physics: solar metal mixture by Asplund et al. (2009), OPAL opacity tables by Iglesias & Rogers (1996), mixinglength theory of convection with a fixed mixing-length parameter of 1.8 (expressed in local pressure scale heights), and the Schwarzschild criterion for convective stability  For chosen time steps along the evolutionary track, or equivalently Xc values, the equilibrium models are perturbed in a linear framework to compute oscillation modes and their accompanying frequencies.  Here, we limit ourselves to frequencies of coherent low-degree (l) zonal (m = 0) gravity and pressure modes, computed for each of the ∼70 values of Xc between the ZAMS and the TAMS. These oscillation modes were computed with the public code GYRE (Townsend & Teitler 2013; Townsend et al. 2018), adopting the Cowling and adiabatic approximations as a good approach, as argued in Paper I.  The deep neural network (DNN) provided here is a first step to improving upon manual analyses in terms of computational cost; it will need future upgrades to deal with correlations among the parameters and to deliver appropriate error estimation in a Bayesian framework, just as for the manual analyses.  In this work, a DNN is trained on an asteroseismological grid of intermediate-mass and highmass stars to predict the frequencies of their coherent oscillation modes. This is a typical regression problem.  The neural network architecture used in this work is a fully connected or dense network. A schematic view of a dense network is shown in Figure 1.    The input vector in this example is six-dimensional and the output is one-dimensional. In between are two layers that are n-dimensional. The layers have 6, n, n, and 1 neurons, respectively, in this case. The value of each neuron, except the input neurons, is determined by the following function:    In order to add non-linearities to the neural network, the activation function must be non-linear. A typical choice is the Rectified Linear Unit (ReLU; Nair & Hinton 2010). This function is defined as f (x) = x if x > 0 and f (x) = 0 if x < 0. By starting from the input layer and calculating all values of all neurons layer-by-layer, the output is finally computed. Initially, the weights and biases are set to random numbers and during training the weights are adjusted to correctly predict a predetermined training set.  To validate that the network has correctly learned to predict the occurring features, the network is evaluated on a separate validation set. Typically, the validation set is 10% to 20% of the total data set.  For regression problems, this typically is a mean-squared error between the prediction by the neural network and the actual output belonging to a particular training sample. By backpropagating the error through the network, the weights and biases are updated to improve the cost function in the next iteration (Chauvin & Rumelhart 1995).  A typical caveat of training neural networks is overfitting. This means the neural network learns to replicate the exact results of the training set, instead of generalizing over the particular samples and learning the underlying correlations. This can be solved by adding a penalty term for high-valued weights, which is called regularization (Ng 2004). One can verify if a network is overfitted by comparing the training and validation set accuracy. If the training loss is decreasing but the validation loss is increasing, the network is overfitting. A more thorough introduction into deep learning and the validation approach we followed here is given in the Appendix of Caron et al. (2018), to which we refer for further details.  The aim of this work is to train a neural network that is able to predict the frequencies of the coherent pulsation modes of a given star with detected oscillation frequencies belonging to identified modes (l, npg). We want to achieve the capacity to quickly search the parameter space spanned by the stellar parameters to find the optimal values for given measurements of the oscillations, without relying on any other information. The choice of using deep neural networks for this problem has several reasons. First, the size of the data set is quite large. From the grid, around 62 million oscillation modes are extracted and the objective is to predict the frequency belonging to specific modes given by a wide range of stellar parameters. The aim is not to train on parameters of a particular star, but for all possible stellar parameters that are available in the data set in one go. This forces the ML algorithm to generalize over all parameters and learn the underlying structure. It also means that the network only has to be trained once, and then can be applied to all stars that reside somewhere inside the training grid. Although training can take some time (around six hours for this work), evaluation is typically very fast (for this work on the order of microseconds per inference) and the network size is not very large (the compressed network file of this work is 370 MB). Furthermore, by adding a batch normalization layer as the first layer inside the network, the issue of incorrectly weighting parameters as stated in Bellinger et al. (2016) is no longer present. The normalizing layer normalizes all data before it is fed to the trainable parameters. In Bellinger et al. (2016) it is also stated that deep-learningbased regression is unconstrained. In this work that is indeed the case. However if the network accuracy is high enough this does not have to be a problem. If it were, it is trivial to design a constrained activation function that is bounded between two values. For example the hyperbolic tangent activation function is bounded between ±1 and can trivially be scaled and translated to get a bound between two values.  A downside of deep learning compared to random forests is that it is fairly easy with random forests to get insights into why particular predictions are being made, while this information is not accessible from a DNN. Training a DNN is also typically more difficult due to the high number of hyperparameters that need to be set correctly to optimally train on the data. In this work, a DNN is trained using TensorFlow (Abadi et al. 2015) and TFLearn (Damien et al. 2016) on an asteroseismic grid of intermediate- and high-mass stars to predict the frequencies of their coherent oscillation modes. The neural network architecture is shown in Table 1, which has a total number of 53,777,033 trainable parameters. Different neural network architectures have been evaluated and the architecture described in Table 1 provides the best accuracy.  In addition, different hyperparameter combinations have been evaluated. The network is trained using the Adam optimizer (Kingma & Ba 2014) with a learning rate of 10−3 and a batch size of 2048.    1. start with a random sample of N points;  2. take the n points with the least error (lowest cost);  3. sample m points around the n points taken from the previous step, using a Gaussian prior with as mean the parameter values and standard deviation σ1 times the mean;  4. repeat steps 2 and 3 I times;  5. repeat steps 2, 3, and 4 also with a standard deviation of σ2 and σ3 times the mean.  The cost function mentioned above is different from the cost function of the neural network. In this step, the neural network is already trained and no longer optimized. In addition, the loss of the neural network is not propagated to the cost function of the PF. This is because the error of the neural network is constant over all predictions. In a future update the neural network output will also contain uncertainty information. |
| **FUNDAMENTAL PARAMETERS OF MAIN-SEQUENCE STARS IN AN INSTANT WITH MACHINE LEARNING**  <https://iopscience.iop.org/article/10.3847/0004-637X/830/1/31/pdf>  In this work, we consider the constrained multiple-regression problem of inferring fundamental stellar parameters from observable quantities. We construct a random forest of decision tree regressors to learn the relationships connecting observable quantities of main-sequence (MS) stars to their zero-age mainsequence (ZAMS) histories and current-age structural and chemical attributes.  We validate our technique by inferring the parameters of simulated stars in a hare-and-hound exercise, the Sun, and the well-studied stars 16 Cyg A and B. Finally, we conclude by applying our method on a catalog of Kepler objects-of-interest.  We explore various model physics by considering stellar evolutionary tracks that are varied not only in their initial mass and chemical composition, but also in their efficiency of convection, extent of convective overshooting, and strength of gravitational settling. We compare our results to the recent findings from GBM (Silva Aguirre et al. 2015), ISO (Metcalfe et al. 2015), interferometry (White et al. 2013), and asteroseismic glitch analyses (Verma et al. 2014) and find that we obtain similar estimates but with orders-of-magnitude speed-ups.  We seek a multiple-regression model capable of characterizing observed stars. To obtain such a model, we build a matrix of evolutionary simulations and use machine learning to discover relationships in the stellar models that connect observable quantities of stars to the model quantities that we wish to predict. The matrix is structured such that each column contains a different stellar quantity and each row contains a different stellar model.  Uses MESA  We train a random forest regressor on our matrix of evolutionary models to discover the relations that facilitate inference of stellar parameters from observed quantities.  A schematic representation of the topology of our random forest regressor can be seen in Fig 3.    **A random forest** is an ensemble regressor, meaning that it is composed of many individual components that each perform statistical regression, and the forest subsequently averages over the results from each component (Breiman 2001). The components of the ensemble are decision trees, each of which learns a set of decision rules for relating observable quantities to stellar parameters. An ensemble approach is preferred because using only a single decision tree that is able to see all of the training data may result in a regressor that has memorized the training data and is therefore unable to generalize to as yet unseen values. This undesirable phenomenon is known in machine learning as over-fitting, and is analogous to fitting n data points using a degree n polynomial: the fit will work perfectly on the data that were used for fitting, but fail badly on any unseen data. To avoid this, each decision tree in the forest is given a random subset of the evolutionary models and a random subset of the observable quantities from which to build a set of rules relating observed quantities to stellar parameters. This process, known as statistical bagging (Hastie et al. 2005, Section 8.7), prevents the collection of trees from becoming over-fit to the training data, and thus results in a regression model that is capable of generalizing the information it has learned and predicting values for data on which it has not been trained.  We choose random forests over any of the many other nonlinear regression routines (e.g., Gaussian processes, symbolic regression, neural networks, support vector regression, etc.) for several reasons. First, random forests perform constrained regression; that is, they only make predictions within the boundaries of the supplied training data (see e.g., Hastie et al. 2005, Section 9.2.1). This is in contrast to other methods like neural networks, which ordinarily perform unconstrained regression and are therefore not prevented from predicting non-physical quantities such as negative masses or from violating conservation requirements. Second, due to the decision rule process that is explained below, random forests are insensitive to the scale of the data. Unless care is taken, other regression methods will artificially weight some observable quantities like temperature as being more important than, say, luminosity, solely because temperatures are written using larger numbers (e.g., 5777 versus 1, see for example Section 11.5.3 of Hastie et al. 2005 for a discussion). Consequently, solutions obtained by other methods will change if they are run using features that are expressed using different units of measure. For example, other methods will produce different regressors if trained on luminosity values expressed in solar units verses values expressed in ergs, whereas random forests will not. Commonly, this problem is mitigated in other methods by means of variable standardization and through the use of Mahalabonis distances (Mahalanobis 1936). However, these transformations are arbitrary, and handling variables naturally without rescaling is thus preferred.  Thirdly, random forests take only seconds to train, which can be a great benefit if different stars have different features available. For example, some stars have luminosity information available whereas others do not, so a different regressor must be trained for each. In the extreme case, if one wanted to make predictions for stars using all of their respectively observed frequencies, one would need to train a new regressor for each star using the subset of simulated frequencies that correspond to the ones observed for that star. Ignoring the difficulties of surface-term corrections and mode identifications, such an approach would be well-handled by a random forest, suffering only a small hit to performance from its relatively small training cost. On the other hand, it would be infeasible to do this on a star-by-star basis with most other routines such as deep neural networks, because those methods can take days or even weeks to train. Finally, as we saw in the previous section, random forests provide the opportunity to extract insight about the actual regression being performed by examining the importance of each feature in making predictions.  Uncertainty: There are three separate sources of uncertainty in predicting stellar parameters. The first is the systematic uncertainty in the physics used to model stars. These uncertainties are unknown, however, and hence cannot be propagated. The second is the uncertainty belonging to the observations of the star. We propagate measurement uncertainties σ into the predictions by perturbing all measured quantities n = 10,000 times with normal noise having zero mean and standard deviation σ. We account for the covariance between asteroseismic separations and ratios by recalculating them upon each perturbation.  The final source is regression uncertainty. Fundamentally, each parameter can only be constrained to the extent that observations are able to bear information pertaining to that parameter. Even if observations were error-free, there still may exist a limit to which information gleaned from the surface may tell us about the physical qualities and evolutionary history of a star. We quantify those limits via cross-validation: we train the random forest on only a subset of the simulated evolutionary tracks and make predictions on a held-out validation set. We randomly hold out a different subset of the tracks 25 times to serve as different validation sets and obtain averaged accuracy scores. |
| **AN IN-DEPTH STUDY OF GRID-BASED ASTEROSEISMIC ANALYSIS**  <https://iopscience.iop.org/article/10.1088/0004-637X/730/2/63/pdf>  GRID BASES MODELLING  As had been reported earlier, we find that it is relatively easy to get very precise values of stellar radii using grid-based techniques. However, we find that there are small, but significant, biases that can result because of the grid of models used. The biases can be minimized if metallicity is known. Masses cannot be determined as precisely as the radii and suffer from larger systematic effects. We also find that the errors in mass and radius are correlated. A positive consequence of this correlation is that log g can be determined both precisely and accurately with almost no systematic biases. Radii and log g can be determined with almost no model dependence to within 5% for realistic estimates of errors in asteroseismic and conventional observations. Errors in mass can be somewhat higher unless accurate metallicity estimates are available. Age estimates of individual stars are the most model dependent. The errors are larger, too. However, we find that for star clusters, it is possible to get a relatively precise age if one assumes that all stars in a given cluster have the same age.  We use three grids of models to test the model dependence of grid asteroseismology results. We use the Yale–Birmingham (YB) pipeline (Basu et al. 2010; described in Section 2.2) as the basis. We also estimate mass and radius directly from Δν, νmax, and Teff to determine whether or not grid methods are really needed to estimate these quantities. We test whether errors in the estimated mass and radius are correlated. And although determining ages of single stars is difficult (and model dependent), we examine whether seismic data allow us to do better than conventional fitting of evolutionary tracks. |
| **AUTOMATIC DETERMINATION OF STELLAR PARAMETERS VIA ASTEROSEISMOLOGY OF STOCHASTICALLY OSCILLATING STARS: COMPARISON WITH DIRECT MEASUREMENTS**  <https://iopscience.iop.org/article/10.1088/0004-637X/725/2/2176/pdf>  Grid based modelling  Here we present the SEEK package developed for the analysis of asteroseismic data from the Kepler mission. A central goal of the package is to obtain a fast and automatic determination of the stellar radius and other parameters in a form that is statistically well defined. The algorithms are tested by comparing the results of the analysis with independent measurements of stellar radius and mass for a sample of well-observed stars. We conclude that the SEEK package fixes stellar parameters with accuracy and precision |
| Metcalfe et al. (2009)  the authors combine a pipeline using a genetic algorithm to find the best model in the parameter space and a local analysis, based on linearization around this solution, to refine the fit and estimate the error in that solution.  Metcalfe et al. (2009) assumes that the solution is linear within the error bars and that no nearby local minima can be deep enough to contribute to the error analysis; this can lead to underestimation of the errors. In our view, only an error estimator, Bayesian or frequentist, looking at various models around the best solutions may provide fully reliable results. This is particularly true in a problem, as the present, where the valley of solution is extended. |
| **FUNDAMENTAL PARAMETERS OF MAIN-SEQUENCE STARS IN AN INSTANT WITH MACHINE LEARNING**  stellar ages cannot be measured directly; instead, they depend on indirect determinations via stellar modelling.  Traditionally, to determine the age of a star, procedures based on iterative optimization (hereinafter IO) seek the stellar model that best matches the available observations (Brown et al. 1994). IO is computationally intensive in that it demands the calculation of a large number of stellar models (see Metcalfe et al. 2009 for a discussion).  Several search strategies have been employed, including:  **grid-based modelling**, see Gai et al. 2011; Chaplin et al. 2014)  GBM by way of interpolation in a high-dimensional space, on the other hand, is sensitive to the resolution of each parameter and thus requires a very fine grid of models to search through (see e.g., Quirion et al. 2010, who use more than five million models that were varied in just four initial parameters). Additional dimensions such as efficiency parameters (e.g., overshooting or mixing length parameters) significantly impact on the number of models needed and hence the search times for these methods. As a consequence, these approaches typically use, for example, a solar-calibrated mixing length parameter or a fixed amount of convective overshooting. Since these values in other stars are unknown, keeping them fixed therefore results in underestimations of uncertainties. This is especially important in the case of atomic diffusion, which is essential when modelling the Sun (see e.g., Basu & Antia 1994), but is usually disabled for stars with M Mε > 1.4 because it leads to the unobserved consequence of a hydrogen-only surface (Morel & Thévenin 2002).  **in situ optimization (hereinafter ISO)** such as genetic algorithms (Metcalfe et al. 2014, Basu et al. (2010)), ISO requires that new stellar tracks are calculated for each target, as they do not know a priori all of the combinations of stellar parameter values that the optimizer will need for its search. They furthermore converge to local minima and therefore need to be run multiple times from different starting points to attain global coverage.  **Markov-chain Monte Carlo** (Bazot et al. 2012)  **downhill simplex algorithm** (Paxton et al. 2013;  see Silva Aguirre et al. 2015 for an extended discussion on the various methods of dating stars) |
| FUNDAMENTAL PARAMETERS OF MAIN-SEQUENCE STARS IN AN INSTANT WITH MACHINE LEARNING = created grid of evolutionary tracks and used MESA, random trees, tested by blind Hare and Hound  “We seek a multiple-regression model capable of characterizing observed stars. To obtain such a model, we build a matrix of evolutionary simulations and use machine learning to discover relationships in the stellar models that connect observable quantities of stars to the model quantities that we wish to predict.” |
| A Very Brief Introduction to Machine Learning With Applications to Communication Systems  <https://arxiv.org/pdf/1808.02342.pdf>  What is machine learning  The problem of interest is studied in detail, producing a mathematical model that capture the physics of the set-up under study. Based on the model, an optimized algorithm is produced that offers performance guarantees under the assumption that the given physics-based model is an accurate representation of reality.  In contrast, in its most basic form, the machine learning approach substitutes the step of acquiring domain knowledge with the potentially easier task of collecting a sufficiently large number of examples of desired behaviour for the algorithm of interest. These examples constitute the training set. |
| Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs  <https://arxiv.org/pdf/astro-ph/0102224.pdf>  “Artificial neural networks” is a relatively loose term referring to mathematical models which have some kind of distributed architecture, that is, consist of processing nodes (analogous to neurons) with multiple connections (analogous to dendrites and axons). These connections generally have adaptable parameters which modify the signals which pass along them. There are numerous types of artificial neural networks for addressing many different types of problems, such as modelling memory, performing pattern recognition, and predicting the evolution of dynamical systems. Most networks therefore perform some kind of data modelling, and they may be split into two broad classes: supervised and unsupervised. The former refers to networks which attempt to learn the relationship between a data and a parameter domain, while the latter refers to networks used to find “natural” groupings with a data set independently of external constraints. What they have in common is the idea of learning about a problem through relationships intrinsically present in data, rather that through a set of predetermined rules.  feedforward multilayer perceptron. The term perceptron is historical, and refers to the function performed by the nodes. Feedforward means that there is a definite input and output, and a flow of data in one direction. This is in contrast to recurrent neural networks in which data flows in a loop  The weights, w, appearing in equations 1–3 are the free parameters of the network. Clearly, in order for the network to yield appropriate outputs for given inputs, the weights must be set to suitable values. This is done by training the network on a set of input vectors for which the ideal outputs (or targets) are already known. This is supervised learning. |
| Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks  <https://www.amazon.com/Neural-Smithing-Supervised-Feedforward-Artificial/dp/0262527014/ref=as_li_ss_tl?s=books&ie=UTF8&qid=1523836940&sr=1-1&keywords=%22neural+smithing%22&linkCode=sl1&tag=inspiredalgor-20&linkId=70977b12ae60ed55d5e64a2480c5697c>  Input Layer: Input variables, sometimes called the visible layer.  Hidden Layers: Layers of nodes between the input and output layers. There may be one or more of these layers.  Output Layer: A layer of nodes that produce the output variables.  Specifically, the universal approximation theorem states that a feedforward network with a linear output layer and at least one hidden layer with any “squashing” activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units.  A node, also called a neuron or Perceptron, is a computational unit that has one or more weighted input connections, a transfer function that combines the inputs in some way, and an output connection. |
| Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function f (defined below) to the weighted sum of its inputs    The output Y from the neuron is computed as shown in the Figure 1. The function f is non-linear and is called the Activation Function. The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real world data is non linear and we want neurons to learn these non linear representations.  <https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/> |
|  |
| “ML was born from pattern recognition and it is essentially based on the premise that intelligent machines should be able to learn from and adapt to their environment through experience [19]–[24]. Due to the ever growing volumes of generated data”  “training enables the ML framework to discover potential relationships between the input data and the output data of this machine learning framework.”  “Supervised learning algorithms are trained using labelled data [65]. When dealing with labeled data, both the input data and its desired output data are known to the system”  <https://arxiv.org/pdf/1710.02913.pdf> |
|  |
|  |
|  |
|  |
|  |
|  |