Introduction

For the classification of learning algorithms from the perspectives of **Generative vs discriminative** and **Linear vs nonlinear**, check the the notes "generative vs discriminative algorithms_GLM_GDA.pdf" under the same folder. Linearity and nonlinearity are discussed in the end of "Generalized linear models (GLMs) " of the PDF file.

Parametric vs nonparametric

Introduction

https://stats.stackexchange.com/questions/230044/what-are-real-life-examples-of-non-parametric-statistical-models (https://stats.stackexchange.com/questions/230044/what-are-real-life-examples-of-non-parametric-statistical-models)

- Non-parametric models differ from parametric models in that the model structure is not specified a priori but is instead determined from data. The term non-parametric is not meant to imply that such models completely lack parameters but that the number and nature of the parameters are flexible and not fixed in advance.
- So non-parametric methods are those if it makes no assumption on the **population distribution or sample size** to generate a model. For instance, K-means (parametric) assumes the following to develop a model: **All clusters are spherical (i.i.d. Gaussian). All axes have the same distribution and thus variance. All clusters are evenly sized.** As for k-NN(non-parametric), it uses the complete training set for prediction. It calculates the nearest neighbors from the test point for prediction. It assumes no distribution for creating a model.
- Note however, whether making population distribution should not be the key to distinguish parametric and non-parametric algorithm, even though parametric models often have population assumptions. The key is whether the number of parameter is fixed for whatever number of data samples.
- First even we assume a prior distribution, we can still have non-parametric algorithm. Second, we can have parametric model even we don't have a prior distribution. Consider the case of a generative model, where we assume the data is drawn from a fixed but unknown distribution. The learning problem here boils down to estimating the parameters of the distribution. If the distribution is characterized by fixed number of parameters then the generative model becomes a parametric model.

Application of parametric and non-parametric models

- As mentioned earlier, for each or a class of parametric models, there are assumptions to make the models eligible. If the data violates assumptions of a particularly parametric model, then we cannot use that model. Under such situation, a non-parametric model might be a better alternative. For example, if the model assumptions for typical linear regression are violated, then people can choose non-parametric regression. https://statistics.laerd.com/spss-tutorials/linear-regression-using-spss-statistics.php (https://statistics.laerd.com/spss-tutorials/linear-regression-using-spss-statistics.php)
- In general, conclusions drawn from non-parametric methods are not as powerful as the parametric ones. However, when the model assumptions for parametric models are violated, the corresponding results can skew the parameter estimates and ultimately increase the risk of invalid conclusion. Under such a situation, a nonparametric model is more robust to **outliers, nonlinear relationships, and does not depend on many population distribution assumptions**. Hence, it can provide more trust worthy results when trying to make inferences or predictions.
- If model assumptions are violated, we might still use parametric models if there are ways to fix the relevant violations. For example, we may eliminate the outliers, and bootstrap or obtain more data to satisfy the population distribution requirement. Otherwise, we may just use non-parametric models, where we don't need not care the violations such as outliers. For example, in KNN we only need the 'local' data to make predictions, the outliers elsewhere will not affect the results.
- **Is it true that non-parametric models tend to handle data with a 'local' way?** Check this out later.
- The robust features of non-parametric models mentioned earlier are usually achieved at the expenses of calculation efficiency, such as in kNN and LWA. So this **is usually one of the disadvantage of non-parametric models. Another disadvantage** is that non-parametric models are usually not as powerful as parametric model. There are of course exceptions such as SVM.

Examples of parametric and non-parametric algorithms

Parametric and non-parametric linear regression

- A parametric model
  There we try to predict a function parametrized in $w \in \mathbb{R}^d$
  $$f(x) = w^T x$$
  The dimensionality of w is independent of the number of observations, or the size of your data. In other words, **the number of parameter is fixed** for parametric models.
- A nonparametric model
  Instead kernel regression (not Ridge regression) tries to predict the following function:
  $$f(x) = \sum_{i}^{n} \alpha_i k(x_i, x)$$
  where we have n data points, α_i are the weights and $k(x_i, x)$ is the kernel function. Here the number of parameters α_i is dependent on the number of data points n. If we let $n \to \infty$, then $d \to \infty$.

- **Why non-parametric model is often robust to outliers and nonlinearity?**
  Similar to the kernel regression example, here we use locally weighted analysis (LWA) example with linear regression for discussion. Suppose a training set going most flat from left to right and then in the middle has a big bump. We will predict the value in the bump region. Using normal linear regression, there would be big error. With locally weighted algorithm, we put more weight on the training data only in the region closing to x. It is like we do a linear regression only with the data in the vicinity of x but not the whole region. So for such a bump problem, we improve our prediction accuracy and handle nicely the nonlinearity. For the same reason, the outliers not in this region will not contribute a lot to our results as small weight is on those outliers.

Parametric and non-parametric support vector machine (SVM)

- SVM without a kernel is a parametric model as the number of parameter is fixed, while SVM with a kernel is non-parametric as the number of parameter will grow as the number of data grows. Figure this out from the SVM notes (theory part).
- As a nonparametric algorithm, Kernel SVM should be robust to outliers and nonlinearity. For nonlinearity is for sure. But how about outliers. Figure this out in the future and check how the 'localized kernel' such as Gaussian kernel might help.

Tree-based models: nonparametric

https://www.quora.com/Why-is-a-decision-tree-considered-a-non-parametric-model (https://www.quora.com/Why-is-a-decision-tree-considered-a-non-parametric-model)
In decision trees, **the parameters of the model, which is the tree size, is certainly not fixed and usually changes with the size of the data**. For example, smaller datasets could be modeled with small size trees and larger datasets could require bigger trees. The important point is that we are not fixing the tree size beforehand.

Decision tree algorithms are usually robust to outliers. https://datascience.stackexchange.com/questions/37394/are-decision-trees-robust-to-outliers (https://datascience.stackexchange.com/questions/37394/are-decision-trees-robust-to-outliers). This is because decision trees divide items by lines, so it does not matter how far is a point from lines. Most likely outliers will have a negligible effect because the nodes are determined based on the sample proportions in each split region (and not on their absolute values).

However, different implementations to choose split points of continuous variables exist. Some consider all possible split points, others percentiles. But, in some poorly chosen cases (e.g. dividing the range between min and max in equidistant split points), outliers might lead to sub-optimal split points. But you shouldn't encounter these scenarios in popular implementations.

Under what situations algorithms usually robust or sensitive to outliers?

- The fundamental point is if the outliers are included in calculating other statistics such as sample mean, etc., then these outliers would affect the results. Regression, by definition usually means re-go-to-mean. See the formalism of Generalized linear models in the notes "generative vs discriminative algorithms_GLM_GDA", the hypothesis functions $h_\theta(x)$ for all models are always in the form of expectation values, i.e., $E(y \mid x, \theta)$. Thus outliers in regression algorithms can seriously affect the mean/expectation calculation, and eventually affect the validity of the results. Imagine how a few outliers can affect linear regression.
- However, not all regression algorithms are sensitive to outliers. For the locally weighted linear regression, or kernel linear regression, or KNN, due to the "local" action in the algorithm, the outliers outside the "local" region would not affect the results as they are with small weights. Of course, the outliers within a specific "local" region will affect the result in that region. **But generally 'local' action usually makes an algorithm robust to outliers, even though outliers are used in the calculation of mean, etc.**
- If an algorithm is with 'local' action whether by explicitly doing so (e.g. kNN) or by setting local weights (LWA, kernel SVM), then the algorithm will usually also robust to nonlinearity, or can handle nonlinearity well. Therefore, **it might be safe to say that if an algorithm can handle nonlinearity well, then it might also be robust to outliers.**
- In some algorithms, outliers are not included in calculating any statistics at all, as are true for most tree-based models, then these algorithms will completely robust to outliers. However, for some implementations, outliers might also be used to calculate quantities, and thus decision tree models sometimes might also be sensitive to outliers.

Lazy vs active/greedy learning

Introduction

https://en.wikipedia.org/wiki/Lazy_learning (https://en.wikipedia.org/wiki/Lazy_learning)
The primary motivation for employing lazy learning, as in the K-nearest neighbors algorithm, used by online recommendation systems ("people who viewed/purchased/listened to this movie/item/tune also ...") is that the data set is continuously updated with new entries (e.g., new items for sale at Amazon, new movies to view at Netflix, new clips at Youtube, new music at Spotify or Pandora). Because of the continuous update, the "training data" would be rendered obsolete in a relatively short time especially in areas like books and movies, where new best-sellers or hit movies/music are published/released continuously. Therefore, one cannot really talk of a "training phase".

Lazy classifiers are most useful for large, continuously changing datasets with few attributes that are commonly queried. Specifically, even if a large set of attributes exist - for example, books have a year of publication, author/s, publisher, title, edition, ISBN, selling price, etc. - recommendation queries rely on far fewer attributes - e.g., purchase or viewing co-occurrence data, and user ratings of items purchased/viewed.

Here are the key features of lazy algorithm:

- There is no training phase.

- Lazy algorithm considers only local data points when predicting (e.g. kNN) while greedy algorithm considers whole points first and then predicts later. **Verify this in the future.**

Compare k-NN and decision tree algorithm. Both belong to non-parametric algorithm, but k-NN is lazy while decision tree is greedy.

Advantages and disadvantages

The main advantage gained in employing a lazy learning method is that the target function will be approximated locally, such as in the k-nearest neighbor algorithm. Because the target function is approximated locally for each query to the system, lazy learning systems can simultaneously solve multiple problems and deal successfully with changes in the problem domain.

Theoretical disadvantages with lazy learning include:

- Particularly noisy training data increases the case base unnecessarily, because no abstraction is made during the training phase.
- Lazy learning methods are usually slower to evaluate.

Examples of Lazy Learning Methods

- K-nearest neighbors, which is a special case of instance-based learning.
- Lazy naive Bayes rules, which are extensively used in commercial spam detection software. This is not the normal naive Bayes as it does have training step.
- All the instance based learning techniques such as local weighted learning can be regarded as laze learning?

Supervised vs semi-supervised vs unsupervised vs reinforced

Supervised and unsupervised algorithms

See /algorithm folder for studied unsupervised learning algorithms such as clustering algorithms, principal component analysis, independent component analysis, factor analysis and linear discriminant analysis. All other algorithms in the same folder are for supervised learning.

Semi-supervised learning

https://towardsdatascience.com/simple-explanation-of-semi-supervised-learning-and-pseudo-labeling-c2218e8c769b (https://towardsdatascience.com/simple-explanation-of-semi-supervised-learning-and-pseudo-labeling-c2218e8c769b)
https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/ (https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/)

Reinforced learning

https://en.wikipedia.org/wiki/Reinforcement_learning (https://en.wikipedia.org/wiki/Reinforcement_learning)
Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Reinforcement learning is considered as one of three machine learning paradigms, alongside supervised learning and unsupervised learning.

It differs from supervised learning in that labeled input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected. Instead the focus is finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

The environment is typically formulated as a Markov decision process (MDP), as many reinforcement learning algorithms for this context utilize **dynamic programming** techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

Reinforcement learning, **due to its generality**, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, reinforcement learning is called **approximate dynamic programming, or neuro-dynamic programming.** The problems of interest in reinforcement learning have also been studied in the theory of optimal control, which is concerned mostly with the existence and characterization of optimal solutions, and algorithms for their exact computation, and less with learning or approximation, particularly in the absence of a mathematical model of the environment. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

For an introductory course, check the online course cs229.