8.1: Machine Learning Basics

Machine learning

Machine learning [1,2] is an application of artificial intelligence (AI) that provides computer systems with the ability to automatically learn from data, identify patterns, and make predictions or decisions with minimal human intervention. It focuses on the development of computational models that perform a specific task without using explicit instructions. Machine learning algorithms are now used in a wide variety of applications in many areas. This chapter reviews commonly used machine learning algorithms for classification, which will be used with bioactivity data archived in PubChem to build a predictive model for bioactivity of small molecules in the assignment (link to the lecture 8 notebook). There are several articles that provide thorough reviews on the application of machine learning in drug discovery and development, including the following papers:

- Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery Yang, et al., Chem. Rev. 2019, 119, 10520−10594 doi:10.1021/acs.chemrev.8b00728
- Applications of machine learning in drug discovery and development, Vamathevan et al., Nat. Rev. Drug Discov. 2019, 18, 463-477. doi:10.1038/s41573-019-0024-5, PMCID: PMC6552674

Supervised and Unsupervised learning

There are two main categories of machine learning techniques: supervised learning and unsupervised learning.

Supervised learning

In supervised learning, a model is built from a training data set, which consists of a set of input data (represented with “descriptors” or “features”) and their known outputs (also called “labels” or “targets”). This model is used to predict an output for new input data (that are not included in the training data set). Simply put, supervised learning is about building a model, y=f(X), that predicts the value of y from input variables (X). [Note that X is in uppercase to reflect that input data are typically represented with a “vector” of multiple descriptors.] An example of supervised learning is to construct a model that predicts the binding affinity of small molecules against a given protein based on their molecular structures represented with molecular fingerprints. [Here, the molecular fingerprints correspond to the input and the binding affinity corresponds to the output.]

Supervised learning algorithms can be further divided into two categories (regression algorithms and classification algorithms), according to the type of the output data that supervised learning aims to predict.

- **Regression**
  Regression algorithms aim to build a mapping function from the input variable(s) (X) to the numerical or continuous output variable (y). The output variable can be an integer or a floating-point value and it usually represents a quantity, size, or strength. An example of regression problems is to predict the IC50 value of a compound against a target protein from its molecular structure.
- **Classification**
  Classification algorithms attempt to predict the “categorical” variable from the input variables. An example of classification problems is to predict whether a compound is agonistic, antagonistic, or inactive against a target protein.

Unsupervised learning

Unsupervised learning methods identify hidden patterns or intrinsic characteristics in the input data (X) and use them to cluster the data. Contrary to supervised learning, unsupervised learning does not use assigned labels to the input training data [that is, no output/target/label values (y) associated with the input data]. Therefore, it can be used to analyze unlabeled data. An example of the problems that unsupervised learning can handle is to group a set of compounds into small clusters according to their structural similarity (computed using molecular fingerprints) and identify structural features that characterize individual clusters. [For this task, the input data (X) is the molecular fingerprints for the compounds.]

Classification algorithms

Below are some commonly used machine learning algorithms for classification problems.

Naïve Bayes

Naïve Bayes classifiers [3,4] are a collection of supervised learning classifiers based on Bayes' theorem. It is called "Naïve" Bayes because it naively assumes that input features are conditionally independent of each other, which is *not* most likely true. Because the Naïve Bayes classifier is simple, yet effective, it has been commonly used in many applications, especially for text classification tasks (for example, spam mail detection). Read the following document about Naïve Bayes:

- Introduction to Naive Bayes Classification
  https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54

Decision tree

Decision Tree (DT) classifiers [5,6] are non-parametric supervised learning methods that predict the value of a target variable by learning simple decision rules inferred from the data features. While decision tree is easy to understand and interpret, it tends to result in overfitted models that do not generalize the data very well. Read the following web page about decision trees.

- Decision Trees in Machine Learning
  https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

Random forest

Random forests [7-9] are an example of ensemble learning methods (which use multiple learning algorithms to achieve predictive performance better than the performance of any of the constituent learning algorithms alone). A random forest model consists of multiple decision tree models, each of which is built using a sample drawn with replacement (often called a bootstrap sample) from the training set. In addition, a random subset of the original set of features is considered when partitioning at each node during decision tree construction. The resulting random forest model predicts the output class of a new input by letting each classifier vote for a single class or by averaging their probabilistic prediction. Random forest classifiers alleviate data outfitting issues of decision trees. Learn more about random forests from this web page:

- Understanding Random Forest
  https://towardsdatascience.com/understanding-random-forest-58381e0602d2

k-Nearest neighbors

k-nearest neighbors classification [10,11] does not construct a "general" model that predicts an output value. Instead, it simply remembers all instances of the training data. When a new input is provided to the kNN classifier, it finds a pre-defined number (k) of training samples closest to the input and predicts its label (output) from the labels of these k-nearest neighbors (through plurality voting). [For this reason, kNN is an example of instance-based learning or non-generalizing learning. In addition, it is also called a lazy learning because all computations are deferred until a new input is provided for label prediction.

Finding nearest neighbors of an input involves computation of the distances between the input and the training data, which can be evaluated using several distance metrics, including those discussed in Chapter 6 (e.g., Euclidean distance, Manhattan distance, Jaccard/Tanimoto, and so on). Read this introductory material about kNN classifiers.

- Machine Learning Basics with the K-Nearest Neighbors Algorithm
  https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Support vector machine

A support vector machine (SVM) [12-14] is a classification algorithm widely used in many fields. SVMs attempts to find an optimal hyperplane that separates classes by the largest possible margin in the feature space. SVMs use kernel functions that project the data from a low-dimensional space into a higher-dimensional space. This projection makes the data more widely scattered in higher-dimensional spaces, and therefore often more easily separable. In addition, because almost all real-world data is not cleanly separable, the SVM algorithm uses the concept of the soft margin, which allows for some misclassifications. Read this web page to learn about SVM.

- SVM (Support Vector Machine) — Theory
  https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

Neural network

Artificial neural networks (ANNs) [15,16] are biologically inspired computer algorithms designed to simulate the way in which the human brain processes information. An artificial neural network is based on a collection of interconnected artificial neurons (nonlinear information processing units that loosely model the neurons in a biological brain). The interconnections between these neurons are called synapses or weights. These neurons are normally arranged in layers.

An example of supervised neural network algorithms is multi-layer perceptron (MLP) that trains using back-propagation. In this supervised training, the neural network processes the inputs and compares the expected outputs with its actual outputs. Errors are then propagated back through the network and the weights between the neurons are adjusted with respect to the errors. This process is repeated until the errors are minimized. Read the following web page to learn about ANNs.

- Everything You Need to Know About Artificial Neural Networks
  https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1

Classification Performance Metrics

A confusion matrix is a tabular layout that summarizes prediction results from a classification model (often called classifier) and helps analyze the performance of the classifier. It shows the number of correct and incorrect predictions, broken down by each class. The information contained in a confusion matrix can be used to compute a wide variety of performance metrics as summarized in this Wikipedia document (https://en.Wikipedia.org/wiki/Confusion_matrix).

While a confusion matrix may be generated for multi-class problems, this chapter focuses on the one for a binary classification problem. Suppose that we want to identify active compounds against a given target protein, using a binary classifier that predicts whether a compound can change the activity of the target. The predictions may be “YES” (active) or “NO” (“inactive”). The confusion matrix for this classification problem is shown in this figure.

		Actual	
		NO	YES
Predicted	NO	True Negative (TN)	False Negative (FN)
	YES	False Positive (FP)	True Positive (TP)

Figure 8.1.1: Confusion Matrix

Let’s define the terms used in this confusion matrix.

- **True positive (TP)**:
  Cases in which active compounds are predicted to be active.
  (The model predicts “YES” for a compound and it is indeed “YES”)
- **True negative (TN)**:
  Cases in which inactive compounds are predicted to be inactive.
  (The model predicts “NO” for a compound and it is indeed “NO”).
- **False positive (FP)**:
  Cases in which inactive compounds are predicted to be active.
  (The model predicts “YES” for a compound but it is actually “NO”)
- **False negative (FN)**:
  Cases in which active compounds are predicted to be inactive.
  (The model predicted “NO” for a compound but it is actually “YES”)

Among the four cases presented in the confusion matrix, TP and TN are correct predictions and FP and FN are incorrect. Therefore, we can define the accuracy of the predictions using the following equation:

$$Accuracy(ACC) = \frac{\text{Correct Predictions}}{\text{All Predictions}} = \frac{TP+TN}{TP+TN+FP+FN} \tag{8.1.1}$$

This accuracy metric is commonly used as a performance measure for a classification model. However, in some cases, the accuracy alone is not enough to describe the performance of a classifier. For example, suppose that we have two models (A and B) that resulted in the prediction results summarized in the following confusion matrix.

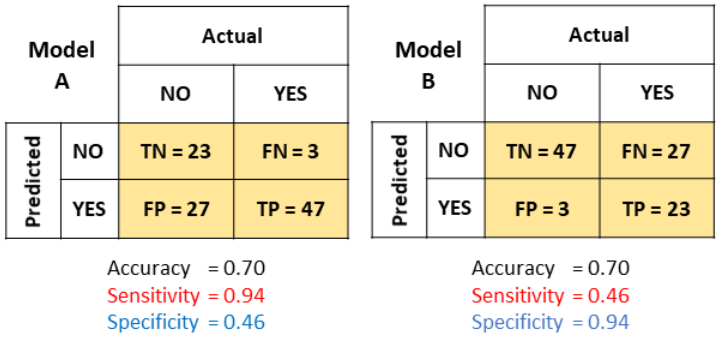

Figure 8.1.2: Accuracy, Sensitivity and specificity for two different confusions matrices representing models A and B

While both models gave the same accuracy (0.70), Model A works better for predicting active compounds (47 out of 50 active compounds were correctly predicted to be active) and Model B works better for predicting inactive compounds (47 out of 50 inactives were correctly predicted to be inactive). This difference can be described using sensitivity and specificity, which are given in these equations:

$$Sensitivity = \frac{\text{Number of Correct YES Predictions}}{\text{Actual number of "YES"}} = \frac{TP}{TP+FN} \tag{8.1.2}$$

$$Specificity = \frac{\text{Number of Correct NO Predictions}}{\text{Actual number of "NO"}} = \frac{TN}{TN+FP} \tag{8.1.3}$$

Sensitivity is the ability of a model to correctly identify "YES". A model with a high sensitivity (e.g., Model A) can correctly predict active compounds to be active. On the other hand, specificity is the model's ability to correctly identify "NO". If a model has a high specificity (e.g., Model B), it can correctly predict inactive compounds to be inactive.

Closely related to sensitivity and specificity are the true positive rate (TPR) and false positive rate (FPR).

$$\text{True Positive Rate} = \frac{\text{Number of Correct YES Predictions}}{\text{Actual number of "YES"}} = \frac{TP}{TP+FN} = Sensitivity \tag{8.1.4}$$

$$\text{False Positive Rate} = \frac{\text{Number of Incorrect YES Predictions}}{\text{Actual number of "NO"}}$$

$$= \frac{FP}{TN+FP} = \frac{FP+TN-TN}{TN+FP} = 1 - \frac{TN}{TN+FP}$$

$$= 1 - \text{Specificity} \tag{8.1.5}$$

The TPR and FPR are used to generate the receiver-operating characteristic (ROC) curve [17-19]. While ROC curve was developed during World War II to detect a sonar signal from an enemy submarine, it is now used as a way to visualize the performance of any predictive model. It is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at *various threshold settings*. The area under the ROC curve (AUC), which ranges from 0 to 1, represents the degree of separability between two classes. A good model has an AUC value close to 1, which indicates a good degree of separability. An AUC value of 0.5 indicates that the model cannot separate two classes from each other. If a model has an AUC score close to 0, it means that the model does the opposite of what is supposed to do (that is, it predicts all active compounds to be inactive and all inactive compounds to be active).

Several web sites provide interactive tools that help understand how the ROC curve is generated and a nice example is the one created by Oleg Alenkin and Alex Rogozhnikov, which is available at:

http://arogozhnikov.github.io/RocCurve.html

More detailed instruction of this tool can be found at http://arogozhnikov.github.io/2015/10/05/roc-curve.html.

The accuracy score often gives an incomplete picture of a model's performance, especially when the model aims to predict rare events (e.g., finding an airline passenger who has illegal firearms in his/her baggage at an airport, or finding a hit compound from a large compound library). Suppose that we construct two models (Models C and D) for bioactivity prediction and test them against a compound set containing 10 active and 990 inactive compounds and the resulting confusion matrices look like the following:

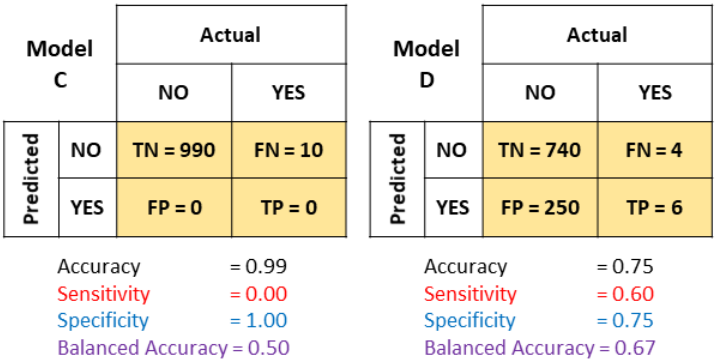

Figure 8.1.3:

Model C has a greater accuracy score (0.99) than Model D (0.75). However, what Model C actually does is to predict *all* compounds to be inactive and fail to find any active compounds at all. If the goal is to identify active compounds for subsequent (in vitro or in vivo) experiments, Model C is not very helpful although its accuracy is close to 100%. Model D may be considered as a better one because it can identify at least some active compounds (although it results in more false positives). For this reason, the balanced accuracy [20] is often used when dealing with imbalanced data.

$$\text{Balanced Accuracy (BACC)} = \frac{1}{2}\left(\frac{\text{Correct YES Predictions}}{\text{Actual YES Predictions}} + \frac{\text{Correct NO Predictions}}{\text{Actual NO Predictions}}\right)$$

$$= \frac{1}{2}(\text{Sensitivity} + \text{Specificity}) \tag{8.1.6}$$

That is, the balanced accuracy for binary classification is the average of sensitivity and specificity. Note that the balanced accuracy of Model D (0.67) is greater than that of Model C (0.50).

8.5. Further Reading

- Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery
  doi:10.1021/acs.chemrev.8b00728
- Applications of machine learning in drug discovery and development
  doi:10.1038/s41573-019-0024-5, PMCID: PMC6552674
- Introduction to Naive Bayes Classification
  https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54
- Decision Trees in Machine Learning
  https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
- Understanding Random Forest
  https://towardsdatascience.com/understanding-random-forest-58381e0602d2
- Machine Learning Basics with the K-Nearest Neighbors Algorithm
  https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
- Chapter 2 : SVM (Support Vector Machine) — Theory
  https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
- Everything You Need to Know About Artificial Neural Networks
  https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1

- Performance Metrics for Classification problems in Machine Learning
  https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b
- Understanding AUC - ROC Curve
  https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

References

1. Yang X, Wang YF, Byrne R, Schneider G, Yang SY: **Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery.** *Chem Rev* 2019, **119:**10520-10594.
2. Vamathevan J, Clark D, Czodrowski P, Dunham I, Ferran E, Lee G, Li B, Madabhushi A, Shah P, Spitzer M, Zhao SR: **Applications of machine learning in drug discovery and development.** *Nat Rev Drug Discov* 2019, **18:**463-477.
3. **Naive Bayes.** https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed Nov. 20, 2019.
4. **Introduction to Naive Bayes Classification.** https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54. Accessed Nov 20, 2019.
5. **Decision Trees.** https://scikit-learn.org/stable/modules/tree.html. Accessed Nov 19, 2019.
6. **Decision Trees in Machine Learning.** https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052. Accessed Nov. 20, 2019.
7. **Random Forests.** https://scikit-learn.org/stable/modules/ensemble.html#forest. Accessed Nov. 19, 2019.
8. **Understanding Random Forest.** https://towardsdatascience.com/understanding-random-forest-58381e0602d2. Accessed Nov. 20, 2019.
9. Breiman L: **Random forests.** *Mach Learn* 2001, **45:**5-32.
10. **Nearest Neighbors.** https://scikit-learn.org/stable/modules/neighbors.html. Accessed Nov. 20, 2019.
11. **Machine Learning Basics with the K-Nearest Neighbors Algorithm.** https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. Accessed Nov. 20, 2019.
12. **Support Vector Machine.** https://scikit-learn.org/stable/modules/svm.html. Accessed Nov. 20, 2019.
13. **Chapter 2 : SVM (Support Vector Machine) — Theory.** https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72. Accessed Nov. 20, 2019.
14. Boser BE, Guyon IM, Vapnik VN: **A training algorithm for optimal margin classifiers.** In *Book A training algorithm for optimal margin classifiers* (Editor ed.^eds.). pp. 144-152. City: ACM; 1992:144-152.
15. **Neural Network Models (Supervised).** https://scikit-learn.org/stable/modules/neural_networks_supervised.html. Accessed Nov 20, 2019.
16. **Everything You Need to Know About Artificial Neural Networks.** https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1. Accessed Nov 20, 2019.
17. Rao G: **What is an ROC curve?** *Journal of Family Practice* 2003, **52:**695-695.
18. Hoo ZH, Candlish J, Teare D: **What is an ROC curve?** *Emergency Medicine Journal* 2017, **34:**357-359.
19. Fawcett T: **An introduction to ROC analysis.** *Pattern Recognit Lett* 2006, **27:**861-874.
20. Brodersen KH, Ong CS, Stephan KE, Buhmann JM: **The Balanced Accuracy and Its Posterior Distribution.** In *2010 20th International Conference on Pattern Recognition*; *23-26 Aug. 2010*. 2010: 3121-3124.