Chapter 1 Exercises

David Piper

September 13, 2020

1 I would define machine learning as the study of algorithms that uses statistics to create programs that can solve problems without using explicit, exact algorithms for such problems. For example the sorting problem is well-known, and can easily be solved with many known algorithms so machine learning doesn't make sense there. In fact it would likely not be a good solution at all because it would be expensive to train an algorithm to solve the sorting problem and would likely not be correct 100% of the time, due to the inherent statistical natural of machine learning. However, if the problem to be solved is making a program to identify fraud then it is very hard to solve with traditional algorithms (as there is no known solution) so using machine learning in this case makes a lot of sense.

2 Machine learning is great for:

   1 Problems for which existing solutions require a lot of fine-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.

   2 Complex problems for which using a traditional approach yields no good solution: the best Machine Learning techniques can perhaps find a solution.

   3 Fluctuating environments: a Machine Learning system can adapt to new data.

   4 Getting insights about complex problems and large amounts of data.

3 A labeled training set is a special type of a training set which contains metadata about the items in the set. That is they are identified, for

example a set of pictures of food could have a description saying what type of food it is.

4 Two common supervised learning tasks are: classification (putting items into categories) and regression (predicting numerical values such as car price given features such as age).

5 Four common unsupervised tasks are

- Clustering
- Anomaly detection and novelty detection
- Visualization and dimensionality reduction
- Association rule learning

6 First I would use a dimensionality reduction algorithm to clean up the input and try to simplify the problem, as analyzing the environment data is very complex if we are using cameras. Then I would use a Reinforcement Learning algorithm to train the robot based on actions (like walking around and bumping into something is bad and walking on terrain that has no rocks/little obstacles is good).

7 Clustering algorithms are a good way to segment customers into multiple groups. It could simply look at the users and tries to pick one for each of them.

8 Spam is typically viewed as a supervised learning problem as it is a classification problem. Specifically it is very easy to gather examples of spam/ham emails and to train the system on them. That said, it is possible that some unsupervised algorithms such as clustering might be applicable here also and they might provide interesting results.

9 An online learning system is one which first trained on an existing dataset and then is trained on new items on-the-fly, rather than requiring completely retraining the system each time from scratch; which is what an offline/batch algorithm would require.

10 Out-of-core learning, or incremental learning, is an approach which is similar to online learning except the goal is not to avoid retraining necessarily, but rather to allow the system to learn in cases when the

dataset is so huge that it won't fit on the system all at once. So in this case the system is trained on small chunks of data from the overall dataset like in online learning until all of the dataset has been used.

11 Instance-based machine learning algorithms use similarity measures to identify new items based on existing data.

12 A model parameter is something that is learned about the data by a machine learning algorithm during the training process, whereas a hyperparameter is a part of the algorithm that is specified by the designer before the training starts and which affects the training. For example: a model parameter for spam detection might be length of email and a hyperparameter might be learning rate.

13 Model-based algorithms search for some general model that can be used to predict output given input, much like scientists. These models have some number of parameters which are then optimized based on some performance measure which tries to minimize something (a cost function) or maximize something (a utility function). So then you feed in the data and select the parameters that best match the data.

14 The four main challenges in Machine Learning are

1 Insufficient Quantity of Training Data

2 Nonrepresentative Training Data

3 Overfitting the Training Data

4 Underfitting the Training Data

15 If your model performs great on training data, but generalizes poorly to new instances you have most likely overfit your model to the training data. Some possible solutions are

- Simplify the model by selecting one with fewer parameters, by reducing the number of attributes in the training data, or by constraining the model.
- Gather more training data.
- Reduce the noise in the training data.

16 A test set is some portion of your overall dataset that you explicitly reserve from using to train your model on so that you have some empirical way to verify that your model generalizes well to new data and didn't overfit the training set.

17 A validation set is a portion of the training data that is used to ensure that you don't overfit the model's hyperparameters to the training set. By way of analogy, this is like the test set for regular parameters, but for hyperparameters instead.

18 A *train-dev* set is yet another testing set similar to a test set or a validation set, it is useful when you have some inherent differences in your data. For example. if 90% of your data comes from a low-quality source and 10% comes from a very reliable high-quality source then you want to make sure that you don't overfit or underfit the high-quality (more representative) data. So in this case you could use only high-quality data for the test set and validation set and then train your model on some combination of the low-quality and high-quality data. However if you do this then if you train your model on the training set and find that it performs poorly on the validation set, you won't be able to tell if your model has overfit the training set or if the problem is coming from the differences in the data between the low-quality data and the high-quality data.

The train-dev set helps you solve this problem. In this case you reserve some of the training data into a set called the train-dev set which you would use to evaluate the model before the validation set. So the flow would be: training set → train-dev set → validation set → test set → production. In this case if the model performs well on the training set, but poorly on the train-dev set then you have overfit the training data. On the other hand, if it performs well on both the training set *and* the train-dev set, but poorly on the validation set then you know that the problem must be coming from the low-quality vs high-quality data mismatch.

19 If you tune hyperparameters using the test set, rather than introducing an extra validation set then you can overfit the hyperparameters to the the test set. This is why a validation set is used to avoid this. That is, once you have set the hyperparameters using the training set and validation set you *don't* change them based on results from the test set.