HoM: Chapter 1: The Machine Learning Landscape

ISLR: Chapter 1, 2: Statistical Learning: ToDo

What is ML?

1. A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. —Tom Mitchell, 1997
2. Simple and elegant definition
3. The three key aspects are:

   - Task (T): What we are looking to do, say, classification.

   - Experience (E): Every observation that is used to train a model and every mistake it makes.

   - Performance (P): Metric used to assess how well our model is achieving the desired task.

4. For a spam filter:

   - The examples that the system uses to learn are called the training set.
   - Each training example is called a training instance (or sample).
   - In this case, the task T is to flag spam for new emails, the experience E is the training data, and the performance measure P needs to be defined; for example, you can use the ratio of correctly classified emails (accuracy).

Traditional Approach: Rule-based Engine vs ML

Since the problem is difficult, your program will likely become a long list of complex rules—pretty hard to maintain. What if spammers notice that all their emails containing “4U” are blocked? They might start writing “For U” instead. A spam filter using traditional programming techniques would need to be updated to flag “For U” emails. If spammers keep working around your spam filter, you will need to keep writing new rules forever.

In contrast, a spam filter based on Machine Learning techniques automatically notices that “For U” has become unusually frequent in spam flagged by users, and it starts flagging them without your intervention.

Machine Learning tasks are about making predictions. This means that given a number of training examples, the system needs to be able to make good predictions for (generalize to) examples it has never seen before.

Other Use Cases:

1. Machine Learning shines for problems that either are too complex for traditional approaches or have no known algorithm. eg. Speech recognition - thousands of words spoken by millions of very different people in noisy environments and in dozens of languages
2. Interpretation and Important Predictors: Machine Learning can help humans learn. ML algorithms can be inspected to see what they have learned (although for some algorithms this can be tricky). Once a spam filter has been trained on enough spam, it can easily be inspected to reveal the list of words and combinations of words that it believes are the best predictors of spam. Sometimes this will reveal unsuspected correlations or new trends, and thereby lead to a better understanding of the problem.
3. Data Mining: Applying ML techniques to dig into large amounts of data can help discover patterns

Types of Machine Learning Systems:

1. Whether or not trained with with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)
2. Whether or not they can learn incrementally on the fly (online versus batch learning)
3. Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model-based learning)

Supervised Learning:

1. Supervised learning:

    - In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels.
    - A typical supervised learning task is classification. The spam filter is a good example of this: it is trained with many example emails along with their class (spam or ham), and it must learn how to classify new emails.
    - Another typical task is to predict a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc.) called predictors. This sort of task is called regression
2. Important supervised learning algorithms:

    - k-Nearest Neighbors
    - Linear Regression
    - Logistic Regression
    - Support Vector Machines
    - Decision Trees and Random Forests
    - Neural networks

Unsupervised Learning:

1. Unsupervised Learning: In unsupervised learning, as you might guess, the training data is unlabeled.
2. Important unsupervised learning algorithms:
    - Clustering

    - K-Means
    - DBSCAN
    - Hierarchical Cluster Analysis (HCA)
  - Anomaly detection and novelty detection: detecting unusual credit card transactions to prevent fraud, catching manufacturing defects, or automatically removing outliers from a dataset before feeding it to another learning algorithm
    - One-class SVM
    - Isolation Forest
  - Visualization and dimensionality reduction: goal is to simplify the data without losing too much information. One way to do this is to merge several correlated features into one. For example, a car's mileage may be strongly correlated with its age, so the dimensionality reduction algorithm will merge them into one feature that represents the car's wear and tear. This is called feature extraction.
    - Principal Component Analysis (PCA)
    - Kernel PCA
    - Locally Linear Embedding (LLE)
    - t-Distributed Stochastic Neighbor Embedding (t-SNE)
  - Association rule learning: goal is to dig into large amounts of data and discover interesting relations between attributes. eg: For a supermarket, who purchase barbecue sauce and potato chips also tend to buy chicken
    - Apriori
    - Eclat

Semi-supervised Learning:

1. Semisupervised learning: Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances. Some algorithms can deal with data that's partially labeled. This is called semisupervised learning

Reinforcement Learning

1. Reinforcement Learning is a a very different beast. The learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards.
2. It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation. eg: DeepMind's AlphaGo program

Batch Learning:

1. System incapable of learning incrementally: it must be trained using all the available data (lot of time and computing resources), so it is typically done offline.
2. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called offline learning.
3. If you want a batch learning system to know about new data (such as a new type of spam), you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data), then stop the old system and replace it with the new one.

Online learning:

1. In online learning, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called mini- batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.
2. Online learning is great for systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously. It is also a good option if you have limited computing resources.
3. Learning Rate: how fast they should adapt to changing data. If you set a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data (you don't want a spam filter to flag only the latest kinds of spam it was shown).

Instance-based Learning:

1. Instance-based learning: the system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them). eg: measure of similarity between two emails - A (very basic) similarity measure between two emails could be to count the number of words they have in common. The system would flag an email as spam if it has many words in common with a known spam email.

Model-based Learning:

1. Model-based learning: Another way to generalize from a set of examples is to build a model of these examples and then use that model to make predictions. This is called model-based learning
2. Sample Linear Equation: This model has two model parameters, θ0 and θ1.
3. Cost Function: For Linear Regression problems, people typically use a cost function that measures the distance between the linear model's predictions and the training examples to evaluate how badly model is performing; the objective is to minimize this distance.
4. This is where the Linear Regression algorithm comes in: you feed it your training examples, and it finds the parameters that make the linear model fit best to your data. This is called training the model.

Main Challenges in ML:

1. Poor Quality Data
   - If some instances are clearly outliers, it may help to simply discard them or try to fix the errors manually.
   - If some instances are missing a few features (e.g., 5% of your customers did not specify their age), you must decide whether you want to ignore this attribute altogether, ignore these instances, fill in the missing values (e.g., with the median age), or train one model with the feature and one model without it.
2. Nonrepresentative Training Data: It is crucial that your training data be representative of the new cases you want to generalize to. This is true whether you use instance-based learning or model-based learning.

3. If the sample is too small, you will have sampling noise (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called sampling bias.
4. Irrelevant Features:
   - A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called feature engineering, involves the following steps:
     - Feature selection (selecting the most useful features to train on among existing features)
     - Feature extraction (combining existing features to produce a more useful one—as we saw earlier, dimensionality reduction algorithms can help)
     - Creating new features by gathering new data
5. Overfitting the Training Data: the model performs well on the training data, but it does not generalize well
   - if the training set is noisy, or if it is too small (which introduces sampling noise), then the model is likely to detect patterns in the noise itself. Obviously these patterns will not generalize to new instances
6. Underfitting the Training Data: Opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data.

Fixing Overfitting:

1. Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. Here are possible solutions:
   A. Simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data, or by constraining the model.
      a. Constraining a model to make it simpler and reduce the risk of overfitting is called regularization. For example, the linear model we defined earlier has two parameters, θ0 and θ1. This gives the learning algorithm two degrees of freedom to adapt the model to the training data: it can tweak both the height (θ0) and the slope (θ1) of the line.
      b. A hyperparameter is a parameter of a learning algorithm (not of the model). As such, it is not affected by the learning algorithm itself; it must be set prior to training and remains constant during training. Tuning hyperparameters is an important part of building a Machine Learning system.
   B. Gather more training data.
   C. Reduce the noise in the training data (e.g., fix data errors and remove outliers).

Fixing Underfitting:

1. Select a more powerful model, with more parameters.
2. Feed better features to the learning algorithm (feature engineering).
3. Reduce the constraints on the model (e.g., reduce the regularization hyperparameter).

Testing and Validating Model:

1. Holdout validation: you simply hold out part of the training set to evaluate several candidate models and select the best one. The new held-out set is called the validation set (or sometimes the development set, or dev set). More specifically, you train multiple models with various hyperparameters on the reduced training set (i.e., the full training set minus the validation set), and you select the model that performs best on the validation set. After this holdout validation process, you train the best model on the full training set (including the validation set), and this gives you the final model. Lastly, you evaluate this final model on the test set to get an estimate of the generalization error.
2. Repeated cross-validation, using many small validation sets: Each model is evaluated once per validation set after it is trained on the rest of the data. By averaging out all the evaluations of a model, you get a much more accurate measure of its performance. There is a drawback, however: the training time is multiplied by the number of validation sets.
3. the validation set and the test set must be as representative as possible of the data you expect to use in production

Exercise:

1. How would you define Machine Learning?
2. Can you name four types of problems where it shines?
3. What is a labeled training set?
4. What are the two most common supervised tasks?
5. Can you name four common unsupervised tasks?
6. What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?
7. What type of algorithm would you use to segment your customers into multiple groups?
8. Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?
9. What is an online learning system?
10. What is out-of-core learning?
11. What type of learning algorithm relies on a similarity measure to make predictions?
12. What is the difference between a model parameter and a learning algorithm’s hyperparameter?
13. What do model-based learning algorithms search for? What is the most common strategy they use to succeed? How do they make predictions?
14. Can you name four of the main challenges in Machine Learning?
15. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?
16. What is a test set, and why would you want to use it?
17. What is the purpose of a validation set?
18. What is the train-dev set, when do you need it, and how do you use it?
19. What can go wrong if you tune hyperparameters using the test set?

Solution:

1. Machine Learning is about building systems that can learn from data. Learning means getting better at some task, given some performance measure.

2. Machine Learning is great for complex problems for which we have no algorithmic solution, to replace long lists of hand-tuned rules, to build systems that adapt to fluctuating environments, and finally to help humans learn (e.g., data mining).
3. A labeled training set is a training set that contains the desired solution (a.k.a. a label) for each instance.
4. The two most common supervised tasks are regression and classification.
5. Common unsupervised tasks include clustering, visualization, dimensionality reduction, and association rule learning.
6. Reinforcement Learning is likely to perform best if we want a robot to learn to walk in various unknown terrains, since this is typically the type of problem that Reinforcement Learning tackles. It might be possible to express the problem as a supervised or semisupervised learning problem, but it would be less natural.
7. If you don't know how to define the groups, then you can use a clustering algorithm (unsupervised learning) to segment your customers into clusters of similar customers. However, if you know what groups you would like to have, then you can feed many examples of each group to a classification algorithm (supervised learning), and it will classify all your customers into these groups.
8. Spam detection is a typical supervised learning problem: the algorithm is fed many emails along with their labels (spam or not spam).
   A. An online learning system can learn incrementally, as opposed to a batch learning system. This makes it capable of adapting rapidly to both changing data and autonomous systems, and of training on very large quantities of data.
9. Out-of-core algorithms can handle vast quantities of data that cannot fit in a computer's main memory. An out-of-core learning algorithm chops the data into mini-batches and uses online learning techniques to learn from these mini-batches.
10. An instance-based learning system learns the training data by heart; then, when given a new instance, it uses a similarity measure to find the most similar learned instances and uses them to make predictions.
11. A model has one or more model parameters that determine what it will predict given a new instance (e.g., the slope of a linear model). A learning algorithm tries to find optimal values for these parameters such that the model generalizes well to new instances. A hyperparameter is a parameter of the learning algorithm itself, not of the model (e.g., the amount of regularization to apply).
12. Model-based learning algorithms search for an optimal value for the model parameters such that the model will generalize well to new instances. We usually train such systems by minimizing a cost function that measures how bad the system is at making predictions on the training data, plus a penalty for model complexity if the model is regularized. To make predictions, we feed the new instance's features into the model's prediction function, using the parameter values found by the learning algorithm.
13. Some of the main challenges in Machine Learning are the lack of data, poor data quality, nonrepresentative data, uninformative features, excessively simple models that underfit the training data, and excessively complex models that overfit the data.

14. If a model performs great on the training data but generalizes poorly to new instances, the model is likely overfitting the training data (or we got extremely lucky on the training data). Possible solutions to overfitting are getting more data, simplifying the model (selecting a simpler algorithm, reducing the number of parameters or features used, or regularizing the model), or reducing the noise in the training data.

15. A test set is used to estimate the generalization error that a model will make on new instances, before the model is launched in production.
16. A validation set is used to compare models. It makes it possible to select the best model and tune the hyperparameters.
17. The train-dev set is used when there is a risk of mismatch between the training data and the data used in the validation and test datasets (which should always be as close as possible to the data used once the model is in production). The train-dev set is a part of the training set that’s held out (the model is not trained on it). The model is trained on the rest of the training set, and evaluated on both the train-dev set and the validation set. If the model performs well on the training set but not on the train-dev set, then the model is likely overfitting the training set. If it performs well on both the training set and the train-dev set, but not on the validation set, then there is probably a significant data mismatch between the training data and the validation + test data, and you should try to improve the training data to make it look more like the validation + test data.
18. If you tune hyperparameters using the test set, you risk overfitting the test set, and the generalization error you measure will be optimistic (you may launch a model that performs worse than you expect).

```
In [ ]:
```