Using Testing to Iteratively Improve Training

Peter Stone **Manuela Veloso**
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
{pstone,veloso}@cs.cmu.edu

Abstract

We present a general Active Learning paradigm and illustrate it within the domain of robotic soccer. We then discuss the measurement of confidence in what has been learned using Active Learning.

Introduction

Machine Learning experiments often involve a training phase and a testing phase. During the training phase, a learning algorithm is (traditionally) applied to a randomly distributed set of examples in the *instance space*. Then during the testing phase, a test set is drawn from the same distribution to measure generalization performance. Learning is deemed to be successful if performance after training is better than performance before training.

However, especially in complex and hostile environments, researchers report successful learning results even though performance on the test set indicates that there is still much room for improvement. For example, if a reasonable algorithm can only classify 30% of the test set correctly, then a learner that can achieve 60% correct classification is considered wildly successful. But perhaps more carefully chosen training data would have led to still better results.

We have recently been working in the domain of robotic soccer. This is a domain with difficult learning tasks in which the training data can have a huge effect on the robot's performance. First we present a general Active Learning paradigm which can be applied to different learning techniques and different domains. We then illustrate this paradigm with a particular case study within our domain. Finally, we discuss the implications of the paradigm with respect to the concept of confidence in learning.

A General Active Learning Paradigm

Although the testing phase is generally the end of a Machine Learning experiment, from an Active Learning perspective it can actually be seen as the beginning. Especially when there is so much room for improvement, appropriately altering the training set could potentially improve performance. Assuming that certain regions of the instance space are more easily classified than others, at least relative to the original training set (perhaps the training set did not have enough examples in a certain region of the space), the testing results can be used to construct a new training set that will allow the same learning algorithm to cover the instance space more completely. Now the testing phase can be viewed as a "what-do-I-need-most-help-with?" phase in an "I-can-get-better!" process.

The paradigm we propose is as follows:

1. Gather training data.
2. Use a Machine Learning algorithm to learn from the training data.
3. Gather testing data.
4. Test the performance of the learner.
5. If generalization performance is adequate, continue testing (Goto Step 3) or stop.
6. Using the results of testing, analyze which regions of the instance space are covered least well by the learner.
7. Gather additional training data from a skewed distribution of the instance space, favoring those regions discovered in Step 6.
8. Combine both sets of training data.
9. Goto Step 2.

Notice that this paradigm can be used with any type of Machine Learning that uses a pre-specified training set. For example, it can be used either with Neural Networks or with Decision Tree Learning. A requirement for Step 7 (and Step 1) is that the instance space can be queried for training examples.

Step 6 is the key step in our paradigm. It requires that the there be some notion of a topology on the instance space so that it makes sense to think of certain *regions* that require more training examples than others. The incorrectly classified test examples can indicate where these regions lie, thus favoring additional training on those regions that are not yet learned well, but that *do* occur during testing (i.e. if part of the instance space is rarely queried, then there is no need to train for it). Then examples from these regions can be combined with (part of) the original training set to produce a more useful training set. The original training set should be retained so as to increase the chance that the *good* learning results will be retained and repeated.

Robot Soccer: A Case Study

As an illustrative example, consider the domain of robotic soccer in which we have been working. We have been using a simulator based on a real-world system to study the learning possibilities in a complex, real-time domain. In particular, we have been studying the task of learning to shoot a moving ball into a goal. The ball passes between the robot and the goal with some speed and heading; the robot must learn when to accelerate so as to redirect the ball into the goal.

This instance space has a clear topology. If the robot is good at scoring when it starts at a particular point, it can probably also score from nearby points. Similarly, if the robot is good at scoring when the ball is moving at a certain speed or in a certain direction, it is probably also good for similar speeds and directions. On the other hand, if the robot is not good at scoring for a given position, speed, and/or direction, it will also have problems at nearby points.

In this soccer domain, our paradigm works as follows:

1. Collect a training set by randomly (within ranges) selecting starting positions for the robot and directions and speeds for the ball. Especially in a simulator, it is possible to set the starting position for a trial as desired. Then the training examples can consist of haphazard attempts to score from these starting positions.
2. Use a neural network to learn from the training data.
3. Gather a random testing set as in Step 1.
4. Test the performance of the neural net on the test set: how often can the robot score when using the output of the neural net as its decision criterion?
5. If performance is adequate, stop. After the first iteration, performance is not likely to be adequate since this is a noisy, complex environment.
6. Use another neural net to learn from the test data whether a particular starting position and ball speed and angle will lead to a successful trial (a goal) or an unsuccessful one (a miss).
7. Use this second neural net to filter possible training examples, choosing only examples that would probably not be correctly classified test examples. Construct an additional training set half the size of the original one by again trying to score haphazardly (or as learned) from these new examples.
8. Combine this training set with a randomly chosen 50% of the original training set.
9. Goto Step 2.

Intuitively, this sequence can help learning because there are neighborhoods of starting positions from which it is more difficult to score. For instance, when the ball is moving quickly, the window of opportunity in which to intercept the ball is smaller. Consequently, the neural net in Step 6 learns that more training examples with the ball moving quickly are needed. Similarly, there may be regions of robot positions (such as off to the side of the goal) which require further training. On the other hand, if the original training set was sufficient for the robot to learn how to shoot a slow-moving ball into the goal, then further such training examples would not be included in the revised training set. Note that our implementation of the Active Learning paradigm keeps the training set at a constant size. However, if the learning algorithm can handle larger training sets without degrading performance, all of the original training examples can be retained. Continuing this Active Learning process iteratively helps the robot to become an expert shooter.

Confidence in Active Learning Results

Notice that our Active Learning paradigm contains an implicit estimate of confidence in the results produced. Step 4 of the general paradigm says *test the performance* of the learner. But what does this statement mean? How does one measure one's confidence in past learning?

The obvious interpretation is a straightforward measurement of how well the system performs on a testing set after learning compared with how it performed before learning. But the key here is the testing set. Although the training data evolves throughout the iterative learning process, the testing data must remain independent and identically distributed (i.i.d.). As long as the testing data is gathered from the same distribution each time, it can provide an accurate measurement of confidence in what has been learned.

The applied version of our paradigm makes this notion of consistency in data-gathering more explicit. Step 3 says Gather a random testing set *as in Step 1*. Step 1 is only visited once to construct the initial training set, but the testing set is constructed in the same way every time. Then Step 4 tests the learning system's performance on the randomly gathered testing set. Confidence is directly measured by way of actual performance.

One important thing to keep in mind is that the testing set need not come from an even distribution. Ideally, it comes from a realistic real-world distribution that reflects the situations in which the learner is intended to be applied. Only then is it apparent how performance has improved *practically*, and only then is it possible to get a meaningful measurement of confidence in learning.

Conclusion

We have presented a general active learning paradigm and illustrated its application to a specific domain. The paradigm relies on repetitive testing with data that is i.i.d. Meanwhile, the training set evolves to better suit both the problem and the learning algorithm. The paradigm can be used in conjunction with any learning algorithm that uses a prespecified training set. Searching in the space of training sets, it iteratively moves towards more efficient and successful learning.