Review of CS 260 Concepts

CS 360 Machine Learning

Week 1, Day 1

January 23, 2024

Contents

1 What is machine learning?

> "Machine Learning is the study of methods for programming computers to learn." - Tom Dietterich
>
> "Machine Learning is about predicting the future based on the past." - Hal Daume III
>
> "Machine learning seeks to answer the question: 'How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes' " - Tom Mitchell

From the above, the phrase "automatically improve with experience" is very central to machine learning, where usually we improve by using data.

Another view on machine learning considers that in traditional programming we would have data and a program and get the output, while in machine learning we have the data and the output and generate the program that can be reused in other settings.

Machine learning is related to statistics, data mining, and psychology of learning, but it is also importantly distinct. While statistics is more about understanding the phenomenon that generated the data, data mining is more about finding patterns and helping humans understand (as opposed to prediction), and psychology focuses on helping humans understand the mechanisms behind how humans learn, machine learning is a separate field.

Why would we want machine learning?

1. In cases where there are no human experts for a task, for example predicting failure points for new machines or other predictions on previously unseen data points.

2. When human experts can't explain their expertise, for example how to understand what a handwritten number look like, machine learning can help us automate this expertise even without the understanding behind it.

3. In cases where phenomena change rapidly, for example adapting to new conditions such as spammers that get better at beating a filter, we can use machine learning to keep up with this speed of change.

4. When the goal is to customize an experience for each user, for example programs that adapt to user's speech, machine learning can be trained to do this automatically.

Machine learning underlies much of the modern web, current innovations in science, and both important and everyday decisions about people.

2 Classification review

A traditional approach (see Figure 1-1 from the textbook) to studying a problem of creating a spam filter is that we look at a lot of emails that users have flagged as spam, we study those and try to find patterns (e.g., emails that use a lot of capitals) and create hand-written rules to detect spam. We would try out those rules, see if they work, and if they do we would launch our spam filter. One key problem with this approach is that the spammers can figure out what rules you've used, and they'll avoid those rules, leading the spam emails to get through the filter again. Additionally, the process of writing rules is tedious (has a large cost in programmer time) and there are likely cases of spam that the programmer might not catch with these hand-curated rules.

A more modern machine learning approach (textbook Figure 1-2) would collect data and train a model to better capture the previously seen data. However, there's still a concern that spammers can learn to adapt to these rules. An adaptive machine learning approach (Figure 1-3) would be able to handle even this last problem.

In the best case, we might additionally be able to learn something from what the machine learning system learns from the data (Figure 1-4). This is very true in biology, for example we might be able to correctly classify people based on their genetic risk for a disease using a machine learning classifier, but being able to better understand the mechanisms behind the classification might allow us to better understand the genuine mechanisms behind the disease.

Another view of a machine learning pipeline (Duame, Figure 1.1) is that you use your training data, pick a learning algorithm which gives you a function that goes from features to your labels, and then under your test examples you're able to predict those labels. Using these stages of this machine learning pipeline, we now unpack these terms further.

2.1 Terminology

Suppose that Alice takes a machine learning course for a semester, but the exam is on the history of pottery. Alternatively, suppose that all exam questions are exactly the same as homework question. Are either of these good tests of Alice's learning? No!

Generalization is a key to machine learning, where the test data is similar, but a bit different, from the training data. Transfer learning can allow you to tackle test datasets that are more different. Throughout, a general machine learning goal is to be able to apply your training data to your test data.

The process of *training* a machine learning model usually involves the program learning from many *examples*. In a *supervised* setting we know the "answer" or *label* and are using this to learn. The resulting output of the training process is known as a *model*. In the process of *testing*, a trained model predicts the output (label) for new examples without using their labels. Key to this process is the expectation that the test data is not examined during the training process - **never look at the test data!**. (One caveat to this view of machine learning is that there are some machine learning problems that do not decompose into training and testing.)

In *supervised learning* we have information about the output or response variable (label) that can be used to create (train) a model. This can be easier for the computer to learn the function between input and output. In *unsupervised learning* the data is unlabeled; there is no output or class information. There may not even be a function to learn!

One common machine learning task is *regression* – in this case the output or response variable is continuous (see textbook Figure 1-6). One example of a regression problem is modeling house price as a function of size, year, location, etc. Another common style of machine learning is *classification*, where the goal is to distinguish two or more *classes* or categories (this is the setting where there is a discrete output variable). For example, one classification problem is, "is the given photo a bagel or a dog?" There's a label you're trying to predict (bagel or dog) and there can be one or more classes (in this case, two). Often, the goal in classification is to distinguish between potentially similar objects.

The common subclass of classification problems when there are two classes is known as *binary classification*; these problems often have Boolean True and False values as the outcome possibilities. In binary classification there are two classes that may have different numbers of examples in the training data and in reality. When the sizes of these classes are roughly the same, these problems are known as *balanced* or *symmetric*. Problems where there are more examples of one class than another are known as *imbalanced* or *asymmetric*. Some examples of asymmetric problems include computing the probability that an email message is spam given the words of the email, determining the probability of Trisomy 21 (Down Syndrome) given the amount of sequencing of each chromosome, determining whether a credit card interaction is fraud given the timing and type of transactions, and deciding who to hire for a job or admit to college. In all of these problems most examples are not of a specific type, for example most emails are not spam, but a small number are flagged and identified as different, e.g., spam emails.

2.2 Evaluation metrics for binary classification

When we evaluate the testing results of trained binary classification models, a key way of representing the results which allows us to examine and compute evaluation metrics to determine whether the resulting model is good, is a confusion matrix. A confusion matrix displays the results of the testing process based on class type and is generally organized so that the rows represent the true class and the columns represent the predicted class of each data point. The cells of the matrix give the number of examples from the testing data that fell into each category (e.g., the number of examples that were from the positive class but were predicted as the negative class, known as the false negatives).

Table 1: General confusion matrix

		Predicted Class	
		Negative	Positive
True Class	Negative	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

Usually we have features $\vec{x}$ and labels $y \in \{0, 1\}$. We often use p for the number of features, i.e., $length(\vec{x}) = p$. The number of training examples is n, so our training matrix is $n \times p$. 1 is often used to indicate the flagging class, e.g., that the person was hired, the email was spam, etc., i.e., it's the thing with fewer training examples or a smaller class in the case of an asymmetric binary classification problem.

Suppose that we have 70 examples that are not spam and we predicted are not spam, we had 30 examples that were not spam and we said were spam, 10 where they were truly spam but we didn't catch them, and 50 that were spam and we correctly identified as spam. These are shown in the confusion matrix

Table 2: Example confusion matrix

		Predicted Spam	
		0	1
True Spam	0	70	30
	1	10	50

above. One thing to note about this confusion matrix is that the total number of examples, the sums of the rows, is fixed. Our algorithm could predict them in different ways, so the sum of the columns can vary based on where we set our thresholds for the classifier and other algorithmic choices. We can tune the algorithm to be more or less sensitive.

One overall evaluation of the algorithm we could make based on the above confusion matrix is its *accuracy*, defined as:

Definition 1 (Accuracy)**.**

$$accuracy = \frac{number\ correct}{total} = \frac{1}{n}\sum I(y_i = \hat{y}_i) \tag{1}$$

Where y_i is the true label for example i, $\hat{y}_i$ is the predicted label for example i, and I is the indicator function which returns 1 when the given value is True and 0 otherwise. In other words, $I(y_i = \hat{y}_i)$ returns 1, and is thus included in the summation, when the predicted and true values for an example i match. It allows you to count all the cases that are correct and skip the cases that are incorrect. Looking back at the confusion matrix, we can see that this is the sum of the diagonal terms (and this would remain true in a larger matrix); the error is all the off-diagonal terms.

If you have binary classification should you ever get a result where the accuracy is less than 50%? No! If you did, you could just flip your outcome.

We can normalize confusion matrices. If so, we want the rows to sum to 1. Why do this with the rows and not the columns? Because the rows have a fixed sum, and the columns do not. We can do this normalization by adding up the rows and dividing by the total for each row. So in our earlier example we get:

Table 3: Example normalized confusion matrix

	0	1
0	0.7	0.3
1	0.17	0.83

Which one do you like better? One thing that can be hidden in the normalized version is the total sample size, while that's very visible from the non-normalized confusion matrix. What else have you lost in the normalized version? You've lost visibility into the asymmetry; how different are the class sizes? But what have you gained? You can directly read off the false positive and false negative rates and it provides somewhat more interpretable metrics directly. There are pros and cons to both types of confusion matrices, and it's important not to hide information that's necessary in real world settings.

Looking back at our general confusion matrix, we also now consider the sums of the rows and columns: the rows indicate the total number of true examples of each class (with N indicating the total number of

true negatives and P indicating the total number of true positives), while the columns give the number of predictions of each class (with $N*$ indicating what we said was negative while $P*$ indicates what we said was positive or "flagged").

Table 4: General confusion matrix with row and column sums

		Predicted Class		
		Negative	Positive	
True Class	Negative	True negative (TN)	False positive (FP)	N
	Positive	False negative (FN)	True positive (TP)	P
		N*	P*	

In general, we want the numbers on the diagonal to be high and the off-diagonal error to be low. We can use this confusion matrix to calculate further important evaluation metrics to quantify this goal.

Definition 2 (Error)**.**

$$error = \frac{FN + FP}{TN + FP + FN + TP} = \frac{FN + FP}{N + P} \tag{2}$$

Note that accuracy $= 1 -$ error.

Definition 3 (Precision)**.**

$$precision = \frac{TP}{FP + TP} = \frac{TP}{P*} \tag{3}$$

Definition 4 (Recall (True Positive Rate))**.**

$$recall\ (true\ positive\ rate) = \frac{TP}{FN + TP} = \frac{TP}{P} \tag{4}$$

Definition 5 (False Positive Rate (FPR))**.**

$$false\ positive\ rate = \frac{FP}{TN + FP} = \frac{FP}{N} \tag{5}$$

In a ROC curve we have the false positive rate on the x-axis and the true positive rate on the y-axis. We want the true positive rate to be high and the false positive rate to be low.