Changelog:
-- 30 Oct 2016
-- 2 Nov 2016

Repetitions are on purpose; typos are not ☺

Lecture 2, Part 1

Machine learning is the study of computer systems that learn from data and experience. It is applied in an incredibly wide variety of application areas, from medicine to advertising, from military defence to pedestrian traffic. Any area in which you need to make sense of data is a potential customer of machine learning.

We will start Lecture 2 by introducing some basic concepts and basic terminology.
In this part of the lecture we will cover the following basic concepts: induction, generalization, data splittings, evaluation methods, evaluation measures, modelling, parameters and inductive bias. You might already have come across some of these notions, for example evaluation measures. We will reapet them here.

In this part of the course, we will focus on supervised learning that is also called inductive learning or indictive statistical learning. Inductive reasoning means that you look at some data, scientifically guess at a general hypothesis, and make statements (ie. predictions on test data) based on this hypothesis. Specifically, the supervised methods that we will study such as DTs, kNN - are all inductive learners. Inductive learning is the general theory behind supervised learning.
Stop and think: what does ”deductive learning” mean?
Deductive learning refers to logical reasoning from given knowledge → eg. automated theorem proving. It is the opposite of inductive learning.
You can imagine inductive learning as the effort of a system that tries to induce a ”general rule” from a set of observed examples. In other words, we can say that induction is the process of reaching a general conclusion from specific examples. Basically, with supervised learning, given a set of examples, we build a function that approximates the target concept. Induction is the process that allows us to generalize on specific examples and make a prediction on examples that have not been seen before. The goal of inductive ML is to use data to induce, to work out, a model. The goodness of this model will be evaluated on unseen data that has not been used during the model construction. If the model has a high performance on unseen data, we can say that the model generalizes well. Let’s take the spam classifier: we want to sort out spam and non-spam emails. We have input data that have been annotated with a label by humans (emails reciepients have classified emails as spam and non-spam). From this data, we try to figure out the best attributes/features for our classification problem (it might be words like ”viagra” or ”write your password here”). We extract the features that are indicative of our problem. Our goal is to predict the label of unlabelled data, i.e new emails we receive every day. We feed the labelled data into a machine leaning algorithm that induces a model from the input features. This induced model should be capable of predicting the right label (spam or non-spam) of new emails. If our classifier is reliable, it should guess correctly on the new emails. You can imagine a “model” as a learning algorithm that ”has learned” (or ”has been induced”) from some input data.
Input data usually take the form of a dataset. Say, that we want to predict the class of an iris flower. This is a classic ML problem. There are 3 species of iris flowers: iris setosa, iris verginica and iris vesicolor. We have a dataset containing attributes/features (sepal length,

sepal width, petal length, petal width), and we have feature values, ie measurements in centimeters and then we have also the correct class label. Our input data tells us that if an iris have sepal lenth 5.1 cm, sepal width 3.5 cm, petal length 1.4 cm, petal width 0.2, that iris will be an iris setosa. And so on. Based on these training examples, we want to induce a model that can give us a reliable prediction of the flower instance that has not been seen before, at training time. A learning algorithm should return the correct prediction for an iris flower that has not been "seen", ie that is not included in the input. In other words, the learning algorithm must generalize on labelled input data and guess the class of an "unseen" instance. A learning algorithm reads in training data and computes a function f. This function can then be applied on new data and automatically make predictions. Essentially, the goal of inductive machine learning is to take some training data and use it to induce a function f. The goodness of this function f will be evaluated on the test data. The machine learning algorithm has succeeded if its performance on the test data is high.
Remember the difference between supervised and unsupervised learning: **Supervised (inductive) learning →** Training data includes desired outputs; **Unsupervised learning**
In supervised learning, there are three components:
1. Class label (aka "label", denoted y)
2. Features (aka "attributes")
3. Feature values (aka "attribute values", denoted x) ⇒ Features can be binary, nominal or continuous. A labeled dataset is a collection of (x,y) pairs

The most central concept in machine learning is generalization: how to generalize beyond the examples that have been provided as input at "training time" to new examples that you see at "test time." The input data "train" the learning method. Then we have unseen examples to "test" what the learning algorithm has learned. We must divide our data in 2 sets: training set and test set. Learning is generalization not memorization. Generalization in ML refers to the ability of a learning algorithm to perform accurately on new/unseen data after having learned from existing examples.

In order to measure the performance of our classifier, we can use several techniques. One technique is to divide the data that we have into 2 parts: a training set and a test set. Suppose we have 1000 examples of iris flowers, we might select 800 of these as training data, and we set aside 200 as test data. We induce our model only from the 800 examples and then we run the induced model on the 200 examples separately and compute our test error on these 200 examples. -- Note that when we split up our data, the examples in test set have a class label, but these labels are not used to learn. They are used only to see if the predictions made by the induced model are correct. The performance of the model on these 200 examples is indicative of how well the model will do in the future on unseen non-labelled examples. Statistics tells us that if the sample is large enough, our data is representative and can get a reliable solution of our problem. Commonly used splits are 80% of the data used as training data and 20% as test data. Or 90% training and 10% test data; We can also see other proportions, for ex 50% and 50% which usually not recommended... There is no mathematical rule to decide about the split. It depends on your data. NEVER EVER MANIPULATE test data: if you do this, your results will be invalid. Remember also that the test data must belong to the same class distribution as the training data. You cannot just use data that have a different class distribution because the induced model can be confused.

Instead of using training data and test data, we can use a different splitting method called cross-validation. In 10-fold cross-validation you break you training data up into 10 equally-sized partitions. You train a learning algorithm on 9 of them and test it on the remaining 1. You do this 10 times, each holding out a different partition as the test data. 10-fold is the most common type of cross-validation. But other typical choices for n-fold are 2 and 5. Cross-validation is more reliable than using a single training set and a single test set. because by chance you can have a very good or a very bad combination of training set and test set, and the assessment of the performance of a classifier might not be realistic. We will come back to cross-validation later.

Leave-one-out cross-validation is a special case of cross-validation where the number of folds equals the number of instances in the data set. Each learning set is created by taking all the samples except one, the test set being the sample left out. Thus, the learning algorithm is applied once for each instance, using all other instances as a training set and using the selected instance as a single-item test set. This process is closely related to the statistical method of jack-knife estimation (see Daume').

Suppose we are training a classifier to predict which of 2 classes, C1 and C2, examples belong to. Suppose we have one sample randomly drawn from the original population. We divide the sample up into a training set and a test set. Suppose it turns out that most of the examples in the training set belong to c1 and most of those in the test set to C2. This is not good. We must ensure that the proportion of each class in the sets is the same as the proportion in the original sample. This is called: stratification
Stratification simply means that we randomly split the dataset so that each class is correctly represented in the resulting subsets — the training and the test set. For example
The Iris dataset consists of 50 Setosa, 50 Versicolor, and 50 Virginica flowers; the flower species are distributed uniformly: 33.3% Setosa; 33.3% Versicolor; 33.3% Virginia. Stratification means that in the training set and test set, or all the folds of cross-validations the 3 classes should be represented in this proportion to get a reliable evaluation of the classifier performance.

How are the errors counted? We use four names to refer to good and bad guesses of a classifier: TPs, TNs, FPs, and FNs.
A false positive is where you receive a positive result for a test, when you should have received a negative result. It's sometimes called a "false alarm" or "false positive error." Ex: Virus software on your computer incorrectly identifies a harmless program as a malicious one. n spam filters, for example, a false positive is a legitimate message mistakenly marked as UBE --unsolicited bulk email, as junk email is more formally known. Risk: Messages that are determined to be spam -- whether correctly or incorrectly -- may be rejected by a server or client-side spam filter and returned to the sender as bounce e-mail. and you might miss something important.
A false negative is where a negative test result is wrong. In other words, you get a negative test result, but you should have got a positive test result. Ex: Quality control in manufacturing; a false negative in this area means that a defective item passes through the cracks. Risk: a potentially dangerous situation may be missed. For example, a crippling computer virus can wreak havoc if not detected, or an individual with cancer may not receive timely treatment.
It is possible to tabulate correct and incorrect guesses in a confusion matrix. Confusion means "error", errors are caused because the classifier is confused. A confusion matrix is basically an error table. It is a useful table that presents both the class distribution in the data and the classifiers predicted class distribution with a breakdown of errors.

Usually, the rows are the observed/actual class labels and the columns the predicted class labels. Each cell contains the number of predictions made by the classifier that fall into that cell.

Multiclass confusion matrix. TPs for each class are the values along the main diagonal.
Say that we have a classification system has been trained to distinguish between cats, dogs and rabbits, a confusion matrix will summarize the results of testing the algorithm for further inspection. Assuming a sample of 27 animals — 8 cats, 6 dogs, and 13 rabbits, the resulting confusion matrix could look like the table on the screen. In this confusion matrix, out of the 8 actual cats, the system predicted that three were dogs, and out of the six dogs, it predicted that one was a rabbit and two were cats. We can see from the matrix that the system has troubles in distinguishing between cats and dogs, but it can distinguish pretty well between rabbits and other types of animals. As we said, all the TPs for each class are located in the diagonal of the table (the diagonal shows all the correct guesses), so it's easy to visually inspect the table for errors, as they will be represented by values outside the diagonal.
Stop and think: what and where are the TNs in a confusion matrix?
See Wikipedia (this example comes from there):
https://en.wikipedia.org/wiki/Confusion_matrix

Given these four numbers (TPs, TNs, FPs, and FNs), we can define the following measures: Accuracy, Precision, Recall, F-Measure

Accuracy is the percentage of correct results. Never trust accuracy when the proportions of the classes in your dataset are unbalanced, because accuracy is biassed towards the most frequent class. Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced (that is, when the number of samples in different classes vary greatly). For example, if there were 95 cats and only 5 dogs in the data set, the classifier could easily be biased into classifying all the samples as cats. The overall accuracy would be 95%, but in practice the classifier would have a 100% recognition rate for the cat class but a 0% recognition rate for the dog class.

In a classification task, the precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). In a classification task, a precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C). F-measure is the weighted harmonic mean of precision and recall. There are different types of F-measures.

The way in which we measure performance should depend on the problem we are trying to solve. There should be a strong relationship between the data that our algorithm sees at training time and the data it sees at test time.

Noise is random variation in the data. It is an anomaly that we can't explain. Noise refers to data points that for some reason are placed erratically. For ex, it might be a white iris flower

instead of purple iris flower (as expected). In the graphical representation noise is placed in an unexpected place. For ex, the blue noise point in the red points' area. Noise has a bearing on how an algortithm learns the data.

There are two notions that describe the distortion caused by how an algorithm has learned from data: underfitting and overfitting. Underfitting means that the model has not learned enough from the data. This means that it cannot generalize on unseen data provided at testing time. Overfitting means that the model has learned too much, even noise and the effect is the same: the model cannot generalize on unseen data provided at testing time.

Overfitting is the effect of a model that is too complicated and too specific to data that has been provided at training time. Underfitting is the result of an excessively simple model. ie the algorithm has not learned enough from input data. Overfitting and underfittin have the same effect on performance: poor predictions on unseen data.

A model can have many parameters. Parameters: The variables or setting that your algorithm is trying to tune to build an accurate model. Parameters are the things that we decided based on the input/training data. For ex, in a DT, parameters are the questions we want to ask to make the decisions; the order we want to ask these questions etc. (*See Daume' 1.9: Models, Parameters, Hyperparameters.*) Finding the best combination of parameters is not trivial and it is only marginally covered in this course.

There are additional "things" that we can adjust. These things are called hyperparameters. Hyperparmeters controls other parameters. Hyperparmeters cannot be learned directly from the data in the standard model training process and need to be predefined. An example of a hyperparametr is the maximum depth of a decision tree or the number of clusters in a k-means clustering.

How do we decide on hyperparmeters? Several approaches are possible. For ex, you can test parameters and hyperpameters via developent set. In order to do so, split your data into 70% training data, 10% development data and 20% test data. For each possible setting of the hyperparameters, train a model using that setting on the training data. Compute the model error rate on the development data. Test all the hyperparamenters on the development set, and from the collection of results, choose the one that achieve the lowest error rate on development data. Evaluate the model with chosen hyperparameters on the test data to estimate future performance.

Inductive bias is another important concept in machine learning. Inductive bias is the set of assumptions that a learning algorithm uses to predict unseen data. A bias is a preference. Each algorithm has its own preferences or personality, if you prefer. Inductive bias is something that is established before the learning algorithm see the data. In the absence of data that narrow down the concept that we want to learn, what kind of solutions are we more likely to prefer? For example, decision trees preferers smaller trees to larger trees, so it spontaneaousy behave accordingly.

Each algorithm has a different inductive bias. We will list them along the way. The inductive bias is another factor that affect the performance of a classifier. (*See Daume' 1.5: Inductive bias: what we know before the data arrives.*)

Not everything is learnable: Noise at feature level; Noise at class label level; Features are insufficiently representative; Labels are controversial; Inductive bias not appropriate for the kind of problem we try to learn.

The key point when we choose a learning algorithm is to see how the model perform on unseen data. Training data is important but the performance, the goodness of an ML model

is measured on data that the algorithm has not seen before, ie, during the training.

Lecture 2, Part 2: Weka

First we are going to see the difference btween the notions of concepts, instance and attribute. Then we will see what an arff file is, what it might contain and so. These concepts are thoroughly explained in the weka book, see the required reading for the lecture.

Let's focus on the input first, ie the trainging data.
A concept is the thing to be learned. Ex: different types of iris flowers.
Instances: the individual, independent examples of a concept. Ex: description of each iris flower.
Attributes: aspects of an instance. Ex: Sepal length, petal width, etc.
Attributes values are the actual measuraments. For ex: 3. 5 cm, 0.1 millimiters, etc.
In LT, attributes are often called "features".

Different styles of learning: We have already mentioned them in Lecture 1: classification, association, clustering, regression etc.

With classification, the aim is to predict a discrete (or categorical) class.
With association, we want to detect associations between different aspects.
With regression (often called "numeric prediction" in the weka book), we want to predict a numeric quantity, instead of a nominal label (as we do in "regular" classification).
Example problems that you will find in the weka book: weather data, contact lenses, irises, labor negotiations.

Classification learning can be *supervised.* Supervised means that the examples of the input come with their actual class. The outcome is a predicted *class* of unseen data (test data). The test data is also labeled by classes. But the model ignores these labels during the prediction phase. These class labels in the test data are used to check how good a model is. Basically, first a prediction is made, then we check if the model's prediction match the actual class in the test data.

Association learning can be applied if no class is specified and when any kind of structure is considered "interesting. Can predict not only classes but any kind of attributes of interest.

Clustering refers to finding groups of items that are similar. Clustering is *unsupervised:* The class of an example is not known. The ML methods must find sensible patterns/grouping by itself, with no indication of the real class.

Regression or numeric prediction is a variant of classification. The class is a numeric value. It is supervised learning.

The input to a machine learning scheme is a set of instances. In the standard scenario, each istance is an independent example of the concept to be learned. An instance contains the attribute value. For example, an instance is an iris flower represtented in

terms of petal and sepal measurements. In other scenarios, there can be multiple instances in an example (this refers to multi-instance classification), but we are not going to talk about that.

Importantly, the same concepts or the same problem can be represented by different attributes. The choice of attributes is a subjective choice. For example, a botanist might say that sepal and petal measuremets are not a good representations of iris flowers, and s/he might suggest that coulor is more indicative. The choice of attributes is normally based on the knowledge that we have of a field or a domain. There are several types of attributes that is possible to use: most commonly used: Nominal, ordinal, interval and ratio.

--- That's all for now.