Supervised Learning

MInDS @ Mines

Supervised learning is the subset of machine learning that covers predicting a target label or class given labeled data. The two main subsets of supervised learning are classification and regression. Classification is supervised learning when the target class is discrete and regression is when it is continuous. There are many methods that can be applied to supervised learning. In this lecture, we will cover some metrics used to evaluate models as well as the k-Nearest Neighbors, Decision Trees and Random Forests methods.

Data is commonly available as features and *labels* or *targets*. Features are the observations that we detect about our data and labels are the useful values that we would like to be able to predict. For supervised learning, we start with data that includes both features and labels. We then learn a model that can predict the labels from the features.

If the labels of our data are continuous, we refer to this as *regression*. If the labels of our data are discrete, we refer to this as *classification*.

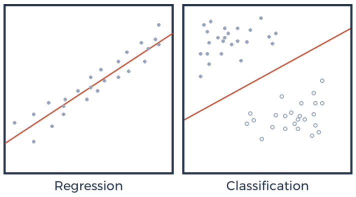

Figure 1: The two types of supervised learning

Metrics

Before we start considering methods that apply supervised learning, let's look at how we evaluate each of classification and regression problems. For each, we have a set of metrics that can be useful in gauging a model's performance.

Classification

The goal of a classifier is to accurately predict the correct label for each data point. Here, we will go over some commonly used metrics to evaluate a classifier.

For all the metrics for classification let's start by defining the notation we'll use. Based on the results of a model that is trained to classify data as one of n classes, c_i is the count of data from class i, and $c_{i,j}$ is the count of data from class i that was labeled as class j. Based on that definition we can create an $n \times n$ matrix that displays how the model performed. This matrix is called the *confusion matrix* and an example of that is,

$$\mathsf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \dots & c_{2n} \\ \vdots & \ddots & & & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \dots & c_{nn} \end{bmatrix}. \tag{1}$$

The simplest metric to use is *accuracy* which is the percentage of predictions that were correct.

The trace of a square matrix, is the sum of the diagonal of that matrix.

$$\mathrm{tr}(\mathsf{A}) = \sum_{i=1}^{n} a_{ii}$$

$$\text{Accuracy} = \frac{\sum_i c_{ii}}{\sum_i \sum_j c_{ij}} = \frac{\mathrm{tr(C)}}{\sum_i \sum_j c_{ij}}, \tag{2}$$

In other words, accuracy is the sum of samples where the class and prediction were the same divided by the sum of all samples.

Accuracy is useful at a high level but it actually doesn't tell us much about our model's performance for each class. For that we'll move on to more involved metrics that will be the basis for our summary metrics.[1] The true positive for a class i, TP_i ,is the count of that class that were labeled correctly. The true negative for a class i, TN_i, is the sum of all the other classes' true positives. The false positive for a class i, FP_i, is the sum of all the other classes' instances that were incorrectly labeled as class i. The false negative for a class i, FN_i, is the sum of all instances of class i that were incorrectly labeled as another class.

From those values, we can calculate *precision*, *recall* and the *F-1 score*. Precision, for a class i, tells us what percentage of data predicted as class i is in fact class i. Recall, for a class i, tells us what percentage of that class was actually predicted as class i. The F-1 score for a class i, is the harmonic mean of precision and recall.

TP_i	c_{ii}
TN_i	$\sum_{j \neq i, k \neq i} c_{jk}$
FP_i	$\sum_{j \neq i} c_{ji}$
FN_i	$\sum_{j \neq i} c_{ij}$
Pr_i	$\frac{TP_i}{TP_i + FP_i}$
Re_i	$\frac{TP_i}{TP_i + FN_i}$
$F1_i$	$\frac{2 * Pr_i * Re_i}{Pr_i + Re_i}$

Table 1: Commonly used metrics in determining classification performance for a particular class i.

These values are useful if we are only interested in a particular class. If we are interested in an overall summary score for the model then we can use a summary score by combining all the available classes' scores. The three most common ways to combine the scores result in the weighted, micro and macro scores for precision, recall and F-1. Micro aggregation use the values at the data level then calculates the average score, macro aggregation calculates each score and then averages them, and weighted aggregation calculates each score and then averages them weighted by the percentage of data in each label.

As an example, the various aggregation methods for precision are,

$$Pr_{micro} = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i)}, Pr_{macro} = \frac{\sum_i Pr_i}{n}, Pr_{weighted} = \sum_i \left(\frac{c_i}{\sum_j c_j} Pr_i\right). \tag{3}$$

Regression

The goal of a regression model is to accurately predict the correct continuous value for the target for each data point. Here, we will go over some commonly used metrics to evaluate a model.

For all the metrics for regression, we will represent the real values of the target as $\mathbf{y}$ where y_i is the target value for the i^{th} item and $f(\mathbf{x})$ and $f(x_i)$ are the respective predicted values. A frequently used metric to assess model performance in regression is the mean squared error (MSE). MSE is the average of the difference between the real value and the predicted value, the error, squared which is,

$$\text{MSE} = \frac{\sum_i (y_i - f(x_i))^2}{n} = \frac{||\, \mathbf{y} - f(\mathbf{x}) \,||_2^2}{n}. \tag{4}$$

[1] If this is your first time learning of these metrics, this section is worth reading a couple times before moving on.

A common variation used of MSE is the root mean square error (RMSE) which is the square root of the MSE.

Another commonly used metric that you may have seen when using Excel to plot a trend line on a chart is the R^2 value. The R^2 value is one minus the sum of the squared errors divided by the sum of the squared distance to the mean. This gives us a value that is weighted by the variability within the sample data.

$$R^2 = 1 - \frac{\sum_i (y_i - f(x_i))^2}{\sum_i (y_i - \hat{y})^2}, \tag{5}$$

where $\hat{y}$ is the average of the real target values.

Methods

We've discussed some metrics to help us evaluate a model, now let's take a look at some methods that can actually solve the supervised learning problem. We'll look at some simple yet effective methods; namely k-Nearest Neighbors, decision trees and random forests.

k-Nearest Neighbors

k-Nearest Neighbors (kNN) is a simple method for applying supervised learning. When training a kNN model, the model practically memorizes all the locations of all the points and their values. When it is time to use the model for predictions, the model takes the input data and calculates the k nearest points to that input. In a classification problem, the majority class of the nearest points is the prediction and in regression, the average of the nearest points is the prediction. This approach may seem simplistic but with a large amount of data, it can be quite effective. When using kNN, there are a few things to consider:

- kNN delays the computational effort until the inference/prediction stage
- k is a hyperparameter that can affect the model's performance
- The way we calculate "nearest points" can significantly affect how our model performs. We can use an ℓ_p-norm for our distance metric and ℓ_1, and ℓ_2 are usually effective.

The Stanford Vision Lab has created a useful demo at `http://vision.stanford.edu/teaching/cs231n-demos/knn/` to illustrate how kNN works..

Decision Trees

You've probably come across decision trees several times before and it might come as a surprise to find out that they can be the basis for a supervised learning model. You've at least seen flow charts before.

Flow charts and decision trees are used to reach a conclusion or make a decision. In machine learning, this final decision is the prediction we are in search of.

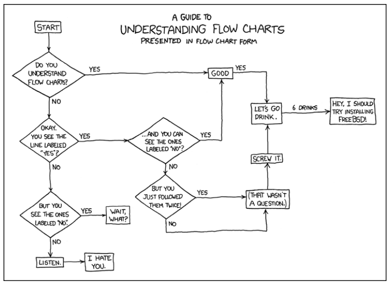
A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
NO
NO
SCREW IT.
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
NO
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
LISTEN.
I HATE YOU.

Figure 2: An example flow chart courtesy of https://xkcd.com/518

A decision tree starts at the root node with all the data and, based on a criterion, it splits the data into two nodes. Each of these nodes then repeats the procedure until we reach a stopping criterion. Stopping criteria include a maximum tree depth or a minimum number of samples in a node.

It is simple to follow a decision tree, but training one can be more complicated. A decision tree trains on a dataset by investigating the features and finding the points on those features where, if we were to split the data, we would get the most consistency in each individual set. The metric we use as a proxy for consistency is *impurity* and there are many ways to calculate it, two of which are Gini and cross entropy.

For a classification problem with k classes, a decision tree with n nodes, has c_m points in node m, and c_{mi} points in node m belonging to class i. We define the proportion of points of node m that belong to class i as $p_{mi} = \frac{c_{mi}}{c_m}$. Following that, the two popular impurity metrics to choose between are,

$$\text{Gini}_m = \sum_i p_{mi}(1 - p_{mi}), \tag{6}$$

$$\text{Cross-entropy}_m = -\sum_i p_{mi} \log p_{mi}. \tag{7}$$

For a regression problem, we can use the MSE of each set to represent its impurity. When using decision trees for regression, each node is an exact value that is the average of its points. This means that there are usually more superior methods for regression problems than decision trees. When using decision trees it is important to understand that:

- The splitting criterion or measure for impurity is a hyperparameter
- The stopping criterion for training is a critical hyperparameter to prevent overfitting of the data

Random Forests

In this section we will briefly introduce random forests. Random forests are an ensemble method[2] applied to decision trees that can improve the model's performance. Similar to a forest being a grouping of trees, a random forest model is an aggregation of several decision trees. Each decision tree is trained on a random subset of the data to create the random forest model. To use the model, the random forest aggregates the prediction from each of the trees. For classification, the model selects the most common class and for regression, it selects the average of the resulting predictions.

When using random forests, the same considerations from decision trees apply. In addition to those, the number of decision trees used to create the random forest model is a hyperparameter to consider.

[2] Ensemble methods are methods that combine several models to produce better predictions.