Exam Preparation
Machine Learning S. 5 Bachelor WS21/22

Jonas Weßner

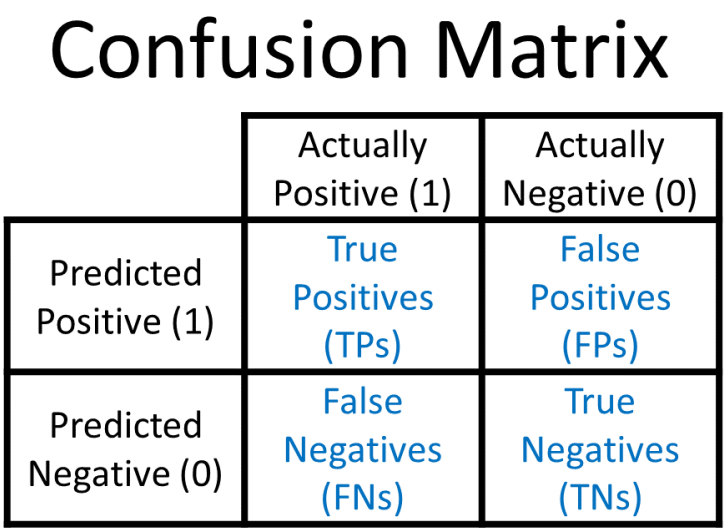
February 5, 2022

Contents

1 Metrics for evaluating predictions

The following metrics can be used to analyze the quality of a *classification model*.

1.1 Confusion Matrix



1.2 Accuracy

Accuracy answers the question "*What is the probability that a prediciton is correct?*".

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

It is only good, if the real distribution of positive and negatives in the data is close to symmetric.

1.3 Precision

Precision answers the question "*If we classify something as positive, how probable is it that it is actually positive?*".

$$Precision = \frac{TP}{TP + FP}$$

1.4 Recall

Recall a.k.a. sensitivity answers the question "*If a sample is positive, what is the probability we also label it as positive?*".

$$Recall = \frac{TP}{TP + FN}$$

1.5 F1 Score

The F1-score divides the true positives by the sum of the true positives and the mean of the false positives and false negatives. This a high F1-score requires the model to make not few false predictions in either direction. Therefore F1-score is better than accuracy if the real distribution of positive and negative values in the dataset is uneven.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2} \cdot (FP + FN)}$$

2 One-hot encoding

Many machine learning algorithms cannot have a categories as ouput values. Therefore if our goal is a categorization of samples we need to make a transformation of labels to numerical values.

Step 1: Integer Transformation

First we must convert all different variants of the category into distinct integer values, e.g.:

Dog	$\rightarrow$	1
Cat	$\rightarrow$	2
OtherAnimal	$\rightarrow$	3

We can now train a machine learning model on that data and it will return one output value. To convert the models output back into classes we could pick the class where the ouput is the closest to the corresponding integer value.

One-Hot Encoding

The problem with integer encoding is that we allow the model to that there is a defined order for the classes. E.g. in the above shown class mapping an ML model could assume that all other animals (3) are closer to cats (2) than to dogs (1). However, as this is not the case at all, using integer encoding might lead to poor predictions.
Instead we can use one-hot encoding resolving that issue. For each output class we create an output neuron / node with the value range of $[0, 1]$. The models predictions are converted back to the classes by taking the maximum of all values of the individual classes. The produces output for the individual classes can be seen as a probability that the sample is of instance of the corresponding class.

$$
\begin{array}{lcll}
\text{Dog} & \rightarrow & [0,1] & \longrightarrow \\
\text{Cat} & \rightarrow & [0,1] & \longrightarrow \quad max \\
\text{OtherAnimal} & \rightarrow & [0,1] & \longrightarrow
\end{array}
$$

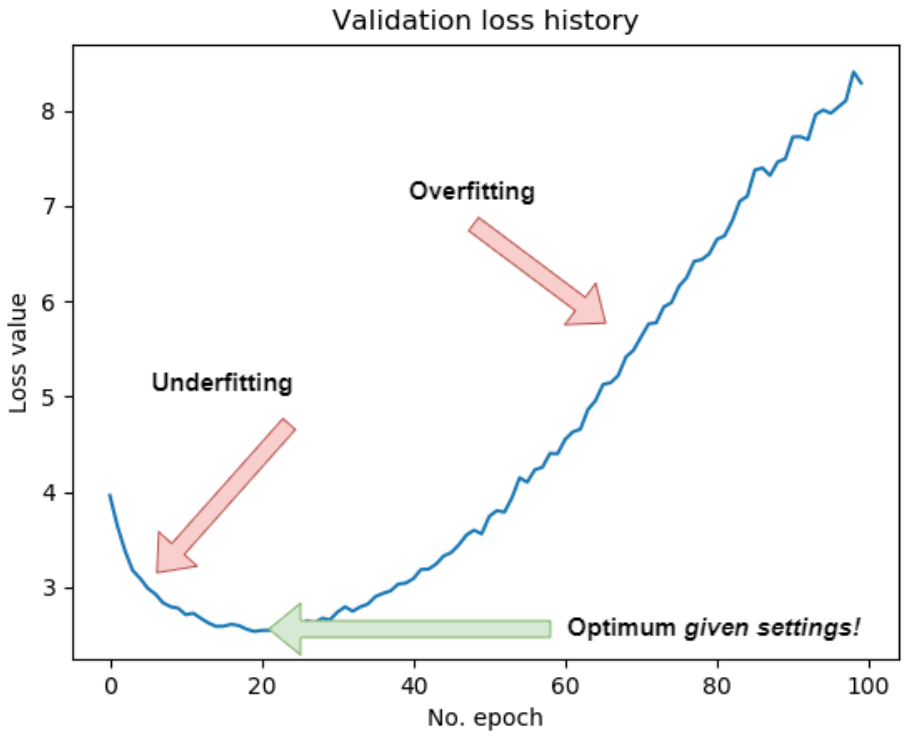
3 Overfitting and Underfitting

In supervised machine learning we usually have a function that determines a specific target parameter (e.g. if a passenger on the titanic survives). However, the function may not be accurate for all samples since the data may include errors (e.g. measurement errors). As we do not know the real underlying function we try to approximate it with supervised machine learning.
If a model is underfitted, it does not take all parameters of the real underlying function into account and therefore is not accurate.
If a model is overfitted, it takes parameters into account that do not apply to the underlying function but where given in the training samples. Overfitting leads to bad results on unseen data as parameters have been learned that are no general indicators for an event and can not be used on other data than the training samples. Overfitting may also occur if the paramters are tuned in a too detailed way that is only applicable to errors or inaccuracy of measurements in the training data.

3.1 How can it be detected?

To detect underfitting and overfitting we can validate the model after each training epoch by letting it make predictions on unseen data and calculating a chosen metric (in the figure we calculate a loss) over that data. Now we can make the following interpretations of the graph:

1. As long as the loss is decreasing the model is still underfitted and shall be trained for more epochs.
2. If the los is increasing the model is becoming more and more overfitted and training can be stopped.
3. If the loss remains unchanged the model has reached the global or a local optimum.



3.2 Possible solutions

1. **Stop early:**
   Stop training to avoid overfitting if your score on the validation set does not increase anymore.
2. **k-fold-cross-validation:**
   Split dataset into k groups and train k-times on k-1 groups while using the remaining group as validation data. This way after each epoch we train on data that has not been seen in that epoch while training more than one epoch.
3. **Increase dataset size:**
   This has positive influence on underfitting and overfitting.
4. **Data augmentation:**
   Create more data by applying some tranformations to the samples. E.g. flip and crop images. This helps preventing that the model learns too much details about individual samples.
5. **Reduce Complexity:**
   Use pooling layers and less neurons to decrease overfitting.
6. **Regularizaiton:**
   Penalizing big numbers of coefficients.
7. **Use Ensembling:**
   Combine predictions of multiple ML-models.

4 PCA - principal component analysis

4.1 Reasons for using PCA

If we have data with n variables/features we can use principal component analysis to reduce the amount of features to k with $k < n$ features while keeping the most important features of the data and eliminating the less important features. Reduction of features is useful to plot the data (because plots with more than 3 features are non-trivial) and also for machine-learning models to increase computation speed.

4.2 Algorithm

1. Calculate mean for each variable. Doing that we also get the center of the data as $(mean(x_1), mean(x_2), ...mean(x_n))$.
2. Now we want to shift the data so that its center is at the center of the coordinate system. We can subtract each individual mean from the corresponding variable to do that.
3. We now compute the principle component 1 PC1. We are searching for a vector (straight line) that fits the data in the x_1 axis best. That means we search for a line where the squared distances of the data points to this line are minimal. It is important that this is equivalent to finding a line where the data is spread out the most if we project it onto the line.
4. For each other variable v_2 to v_n we draw a line that is orthogonal to all preceding lines and rotate it until it represents the data best for the given variable in the same way as with the first line. These lines are PC_2 to PC_n.
5. As said, the data points of the features are spread out as much as possible along the PCs now. This means that the variance and therefore also the amount of information is maximized for the specific variable if we project the data onto that PC. Now we can take the k PCs with the greatest variance (also called eigenvalues here) and use them to represent our data.

5 Python Basics

5.1 Slicing

```
a[start:stop]       # items start through stop-1
a[start:]           # items start through the rest of the array
a[:stop]            # items from the beginning through stop-1
a[:]                # a copy of the whole array
a[start:stop:step]  # start through not past stop, by step
```

5.2 Data Extraction with Pandas

```
df.head(5)                          # show first 5 lines
df.tail(3)                          # show last 3 lines
df.columns
df.describe()                       # statistic summary of data
df["Survived"]                      # get survived column as pandas.Series
df[0:3]                             # get the first 3 rows with all columns
df.loc["2013-01-03"]                # Select a row by the value of the index
column
df.loc["2013-01-03", "name"]        # Select the value of the ‘name‘ column of
that row
df.iloc[5]                          # Select the 5th row by its index
df[df["income"] > 1000]             # selecting rows via a boolean array
df.dropna(how="any")                # drop all rows that contain null values
df.mean()                           # calculates mean of each column and returns a
 Series
df1.fillna(value=5)                 # replace all NaN values with 5
df.apply(lambda x: x.max() - 10)    # apply a function to each data point
```

6 Regularization

6.1 What is regularization

Regularization is the method of penalizing complex model e.g. models with a lot of parameters. It is used to avoid overfitting.

A common loss function for evaluating a model is the residual sum of squares (RSS).

$$RSS = \sum_{i=1}^{n} (y_i - y_i')^2$$

When evaluating the model the loss function shall be minimized.
But we can also use other loss functions that include regularization terms as will be shown in the following.

6.2 Ridge

Ridge adds the sum of the squares of coefficients w to the RSS which makes the loss function prefer smaller coefficients. However coefficients will never be zeroed out using Ridge.

$$Ridge = RSS + \lambda \sum_{i=0}^{n} w_i^2$$

6.3 Lasso

Lasso adds the sum of the absolutes of coefficients w to the RSS which makes the loss function prefer smaller coefficients and possibly drive some coefficients to zero i.e. eliminating them.

$$Ridge = RSS + \lambda \sum_{i=0}^{n} |w_i|$$

6.4 Dropout

As opposed to Ridge and Lasso, which are used mainly used with linear models, dropout is a regularization method used for neural networks. Using dropout means that the output values of some randomly chosen nodes of a layer (which may not be the output-layer) are ignored. This way the net is meant to be more robust to noise in the training data, because it the dropped out nodes will be different in every epoch and therefore the training experience will also slightly differ.

7 Machine Learning Tasks

7.1 Classification

7.2 Regression

7.3 Clustering

8 MLP - Multi-Layer-Perceptron

8.1 What is MPL?

8.2 Calculation of a number of parameters with and without bias

9 Feature map calculation in convolutional NN

10 Input and output sizes in Neural networks

Describe here: Size of inputs and outputs in MLP and convolutional NN calculated from image size and the number of output classes.

11 Activation functions

11.1 Softmax

11.2 Sigmoid

11.3 RELU

12 Solving non-linear problems with NNs

Use example of logical function XOR here.