Data Analysis and Machine Learning: Support Vector Machines

Morten Hjorth-Jensen[1,2]

Department of Physics, University of Oslo[1]

Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University[2]

Nov 2, 2018

Support Vector Machines, overarching aims

A Support Vector Machine (SVM) is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression, and even outlier detection. It is one of the most popular models in Machine Learning, and anyone interested in Machine Learning should have it in their toolbox. SVMs are particularly well suited for classification of complex but small-sized or medium-sized datasets.

The basic mathematics relies on the definition of hyperplanes and the definition of a **margin** which separates classes (in case of classification problems) of variables. It is also used for regression problems.

With SVMs we distinguish between hard margin and soft margins. The latter introduces a so-called softening parameter to be discussed below. We distringuish also between linearn and non-linear approaches.

These notes will be updated shortly with more material

Strength and weakness

When we implement a linear support vector machine, the main parameter is the constant C. Small values of C mean simple models. These models are fast to train and also fast to predict and scale to very large data sets and work well with sparse data. Linear support vector machines make it easy to understand how a prediction is made, however it is often not easy to understand why coefficients are the way they are. These models work also well in higer dimensions.

Examples with kernels

```
from IPython.display import set_matplotlib_formats, display
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import mglearn
from cycler import cycler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.datasets import make_blobs


X, y = make_blobs(centers=4, random_state=8)
y = y % 2

mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.show()

from sklearn.svm import LinearSVC
linear_svm = LinearSVC().fit(X, y)

mglearn.plots.plot_2d_separator(linear_svm, X)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")

# add the squared first feature
```