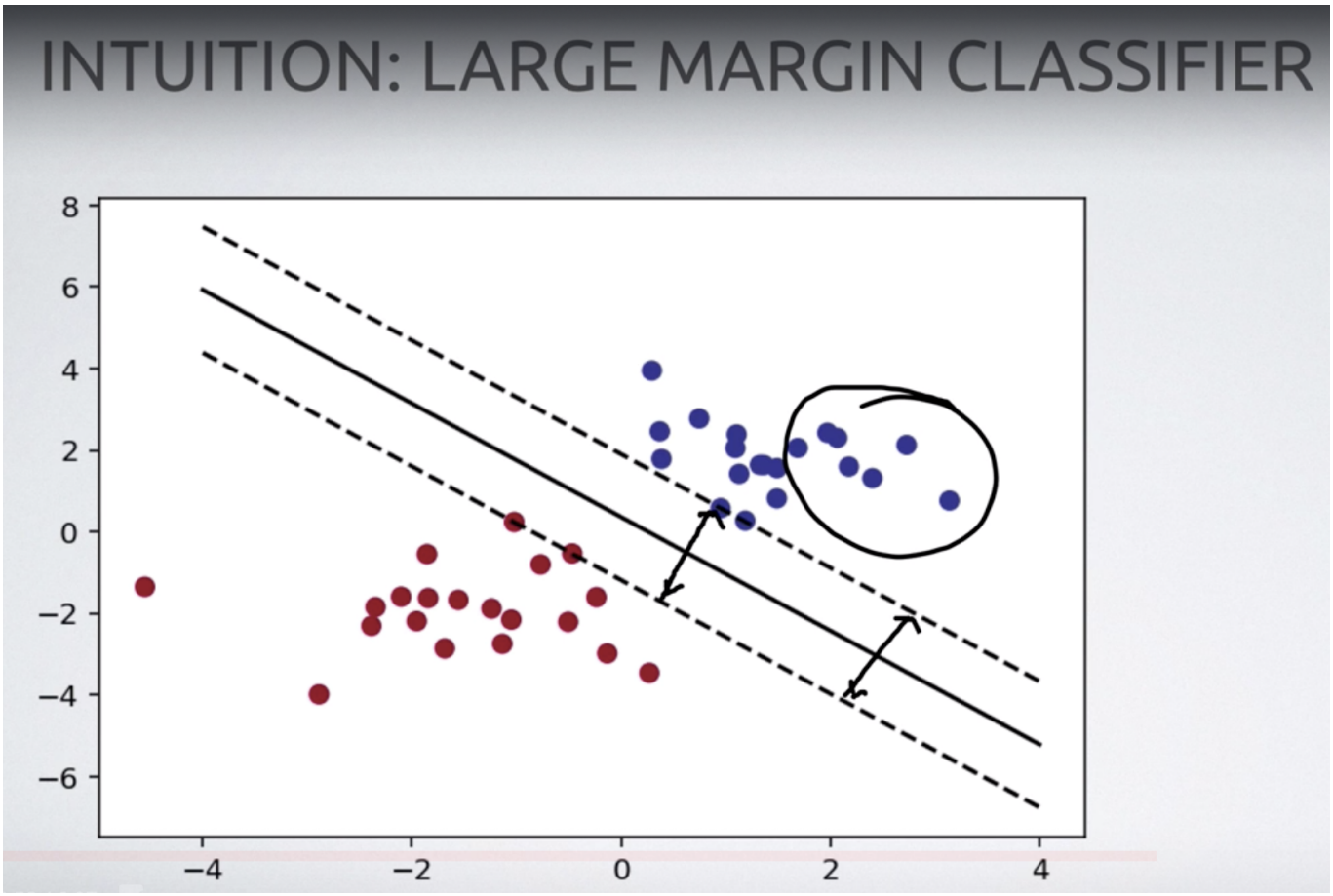
Support Vector Machines

Support Vector Machines (SVM) sind mächtige Klassifizierungsmodelle die einen etwas anderen Ansatz verfolgen als Deep Learning etc. Im Grunde geht es darum eine Trennlinie (`Hyperplane`) zwischen zwei verschiedenen Klassen an Daten zu ziehen. Dies kann natürlich auch im mehrdimensionalen Raum erfolgen. Ein Beispiel zeigt das folgende Bild:



Da es mehrere Möglichkeiten gibt diese Hyperplane zu ziehen, muss man beachten dass das Kriterium nach welchem diese Linie gezogen wird der maximimale Abstand zwischen den 'äußersten Punkten' der beiden Klassen ist. Man möchte also die Breite des Korridor in der Mitte maximieren.

Eine Besonderheit bei `SVM` ist, dass Datenpunkte die weit von der Hyperplane entfernt sind, gar nicht wirklich interessieren. Viel interessanter und wichtiger dagegen sind die Punkte im Grenzgebiet zwischen den beiden Klassen. Das führt auch zu effizienteren Berechnungen (ressourcenärmer)

Support Vector:

Es ist nochmal zu betonen, dass im die Punkte nahe an der jeweils anderen Klasse speziell gewichtet / betrachtet werden und weiter entfernte Punkte nicht wirklich ins Gewicht fallen.

Realisierung

Wichtig bei der realisierung ist auf jeden Fall die Standardisierung! Da Abstände berechnet werden müssen alle Datenpunkte den gleichen Maßstab haben.

```
from sklearn.model_selection import train_test_split

# Welche Spalten sollen zur Vorhersage verwendet werden
X = df[["", ""]].values
y = df[""].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state =
0, test_size = 0.25)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.svm import SVC

#Hier muss der Kernel angegeben werden!
model = SVC(kernel = "linear")
model.fit(X_train, y_train)

print(model.score(X_test, y_test))

#Visualisierung der Ergebnisse

from helper import plot_classifier

# Trainings-Daten plotten
plot_classifier(model, X_train, y_train, proba = False, xlabel = "",
ylabel = "")
```

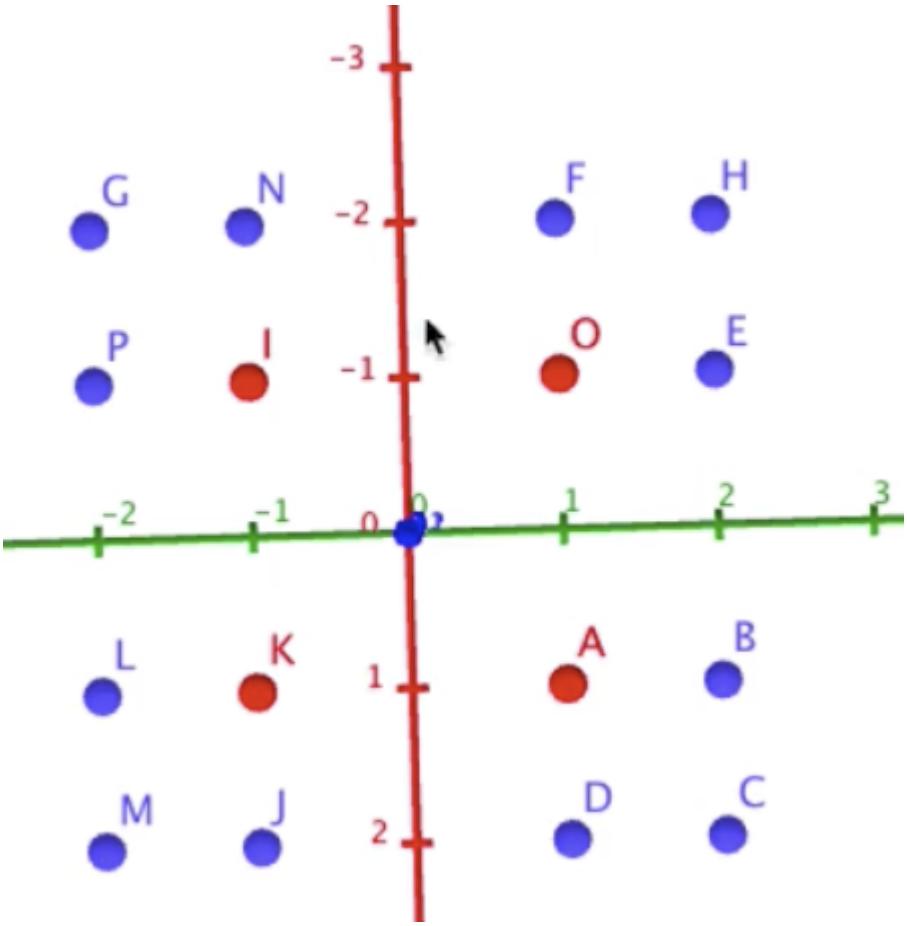
Innerhalb der Realisierung hat man die Möglichkeit den Parameter `C` anzugeben, dieser gibt an wie wichtig es ist alle Punkte möglichst richtig zu klassifizieren , mit Inkaufnahme eines nicht ganz so breiten Korridors. Standardmäßig ist dieser Parameter bei 1 kann aber verringert werden (Fehler werden eher erlaubt) oder vergrößert werden (z.B 100) wenn auf jeden fall KEIN Fehler gemacht werden soll.

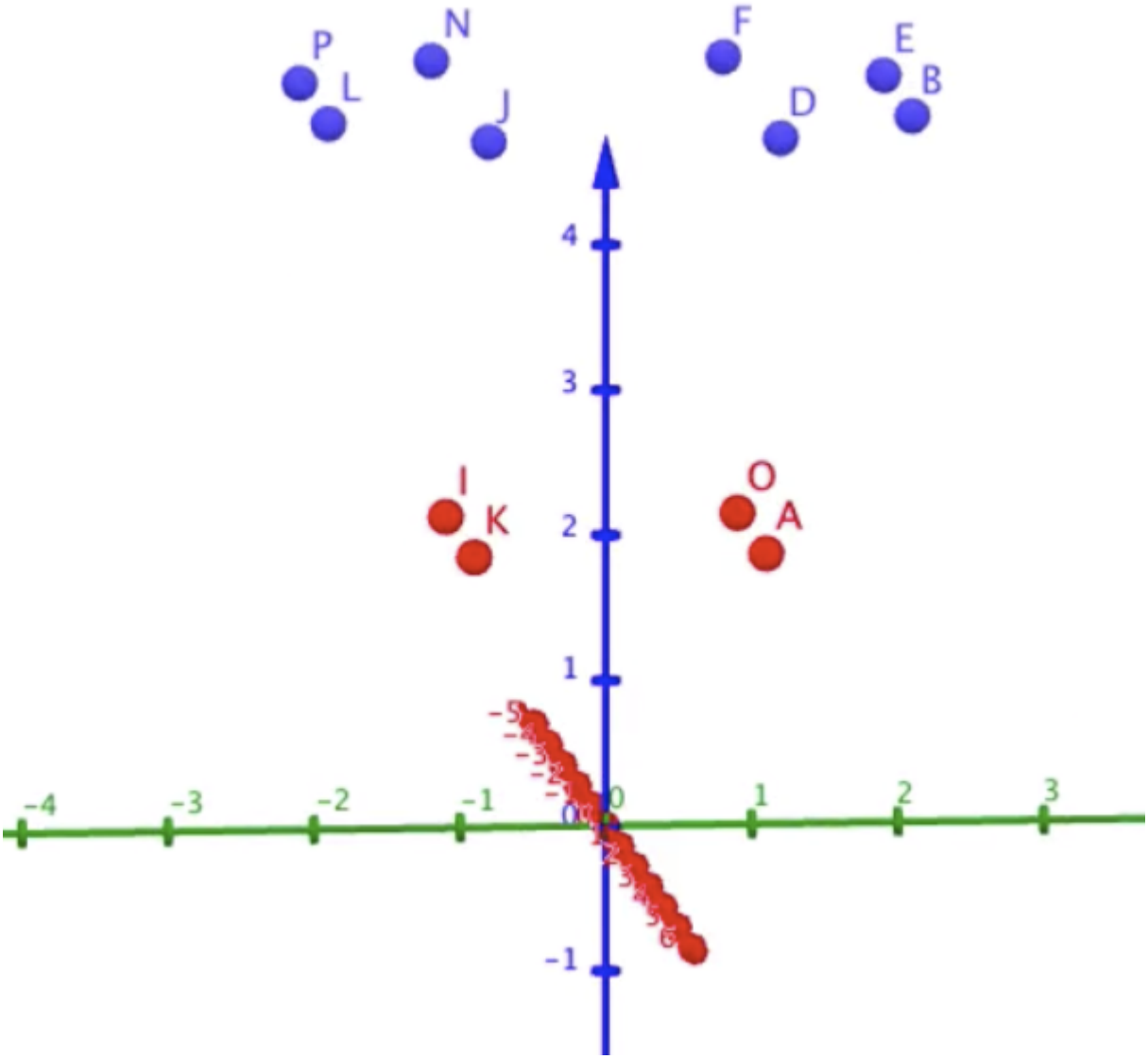
SVM mit Kernel

Man gerät schnell an seine Grenzen wenn die Daten nicht mit einer Geraden abrennbar sind. SVM `mit Kernel` machen es möglich machen zusätzliche Problemkategorien greifen zu können. SVM sind sehr wichtig.

Grundsätzlich werden Datenpunkte dabei einfach in einen höherdimensionalen Raum transferiert um zu ermöglich, dass eine Ebene die Daten besser abgrenzen kann. Beispiele für solche Transformationsfunktionen wären $f(x) = x^{2}$ oder $f(x) = x^{2} + y^{2}$.

Hier sieht man das Ergebnis der Tranformation:


-3
-2
-1
0
1
2
3
-2
-1
0
1
2
G
N
F
H
P
I
O
E
L
K
A
B
M
J
D
C



MIT Kurs

Die gesamte Vorlesung ist viel mathematischer aufgezogen.

Grundsätzlich gibt es eine `Decision Rule`. Diese kann wie folgt abgebildet werden:

$\vec{w} + \vec{u} >= 0$

Daraus entstehen folgende zwei Bedingungen für die zwei Klassen:

$\vec{w} * \vec{x}_{positiv} + b >= 1$ und

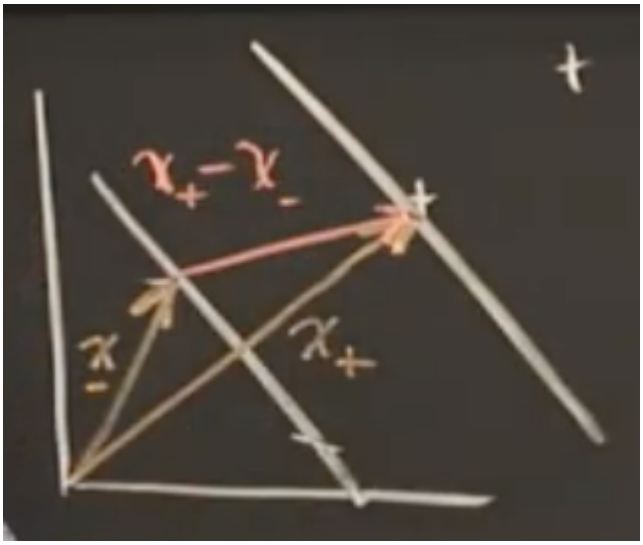
$\vec{w} * \vec{x}_{negative} + b >= -1$.

Um die Gleichung zu vereinfachen wird eine neue Variable eingeführt: `$y_{i}$`. Diese Variable ist `+1` für alle positiven Samples und `−1` für alle negativen.

Aus den zwei oben stehenden Gleichungen kann nun eine Gleichung mit `$y_{i}$` berechnet werden. (Eigentlich sind es zwei Gleichungen, diese sind aber für -1 und +1 gleich)

$y_{i} * (\vec{w} + \vec{x} + b >= 1) >= 1$ und $y_{i} * (\vec{w} + \vec{x} + b >= 1) >= 1$

BEIDE GLEICHUNGEN GLEICH! Daher entsteht daraus:
$y_{i} * (\vec{w} + \vec{x} + b) - 1 = 0$



Das Bild zeigt die zwei resultierenden Vektoren $\vec{x}*{-}$ und $*\vec{x}*{+}$. Außerdem ist der resultierende Abstandsvektor zu sehen, welcher passender Weise die breite des Korridors angibt. Juhu! Dieser wird berechnet aus

$\vec{x}*{+} - *\vec{x}*{-} = BreiteDesKorridors$.

Das Skalarprodukt dieser Formel mit dem Normalvektor ergibt den gesuchten Einheitsvektor:

$EV = (\vec{x}*{+} - *\vec{x}*{-}) * (\frac{\vec{w}}{||w||})$