



Diese Übung wird am 17.06.2021 um 13:30 Uhr besprochen und nicht bewertet.

Benötigte Dateien

Alle benötigten Datensätze und Skriptvorlagen finden Sie in unserem Moodle-Kurs:

<https://moodle.informatik.tu-darmstadt.de/course/view.php?id=1058>

8.1 Supportvektormaschinen

Supportvektormaschinen (engl. support vector machines) wurden 1995 von Cortes und Vapnik [1] eingeführt, obwohl ihre Idee schon früher diskutiert wurde. Nachfolgend werden ihre Prinzipien kurz erläutert. Um einfach zu beginnen, wird angenommen, dass die Eingabedaten aus zwei Klassen mit $y_i \in \{-1, 1\}$ bestehen und linear trennbar sind. Jeder Datenpunkt wird als ein Vektor im Eingabefeature-Raum gesehen. Die SVM-Methode zielt darauf ab, die Parameter \mathbf{w} und b für die lineare Entscheidungsfunktion zu finden

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

mit der Hyperebene $\mathbf{w}^T \mathbf{x} + b = 0$, die die Datenpunkte trennt:

$$\mathbf{w}^T \mathbf{x}_i^+ + b \geq 1 \quad (1)$$

$$\mathbf{w}^T \mathbf{x}_i^- + b \leq -1, \quad (2)$$

sodass alle positiven Beispiele \mathbf{x}_i^+ auf der einen Seite der Hyperebene liegen, während alle negativen Punkte \mathbf{x}_i^- auf der anderen Seite liegen. Der Abstand eines Punktes \mathbf{x}_i zur Hyperebene ist gegeben durch

$$\frac{y_i y(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|} \quad (3)$$

und die Spanne wird durch den orthogonalen Abstand zum nächstgelegenen Punkt \mathbf{x}_i definiert. Das Ziel ist es, \mathbf{w} und b so zu optimieren, dass die Spanne maximiert wird. Dies kann durch Lösen folgenden Optimierungsproblems erreicht werden

$$\operatorname{argmax}_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i [y_i (\mathbf{w}^T \mathbf{x}_i + b)] \right\} \quad (4)$$

w.r.t. $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$.

a) Datensatzvisualisierung

Beginnen wir mit einem einfachen Datensatz, der aus zwei Klassen-Clustern besteht.

Visualisieren Sie den Datensatz mittels eines Scatterplots in der Methode `plot_data`.

b) Lineare Kernel

Verwenden Sie das Modul `sklearn.svm.SVC` in der Funktion `fit_linear_svm`, um die Stützvektoren zu finden, die die Hyperebene in den Daten definieren, und die Spanne zu den nächstgelegenen Datenpunkten maximiert.

Verwenden Sie dabei keine Regularisierung ($C = 10^{10}$), sodass alle Stützvektoren so definiert, dass sie die Datenpunkte sind, die auf dem Rand der Spanne liegen und den exakt gleichen orthogonalen Abstand zur trennenden Hyperebene (den Randwert) haben.

c) Kernel-Trick

Wie Sie sehen können, trennt die Hyperebene die beiden Klassen perfekt voneinander. Wenn wir eine SVM mit linearem Kernel jedoch auf einem neuen Datensatz zweier konzentrischer Kreisen anwenden ist eine lineare Hyperebene unzureichend.

SVMs erlauben die Verwendung des sogenannten *Kernel-Tricks* für Probleme, die nicht wie oben angenommen linear trennbar sind, d.h. die Berechnung des Vektorprodukts in einem Merkmalsraum, in welchem die Daten erneut linear separierbar sind. Bekannte Kernel sind:

1.) der lineare Kernel:

$$k_{linear}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j, \quad (5)$$

2.) der d -Grad-Polynom-Kernel:

$$k_{poly}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d, \quad (6)$$

3.) der radiale Basisfunktionskernel (RBF):

$$k_{rbf}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2). \quad (7)$$

Wir müssen daher eine Hyperebene in einem anderen Kernelraum finden, in dem die Daten wieder trennbar werden. Verwenden wir nun den radialen Basisfunktion Kernel $k_{rbf}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ in der Methode `fit_rbf_svm` um unseren Merkmalsraum zu transferieren.

Geben Sie dabei $\frac{1}{n_{features}}$ als Kernel Koeffizient γ an.

d) Regularisierung

Ein weiterer wichtiger Hyperparameter für die SVM ist die Regularisierung. Die SVM-Optimierung versucht, Datenpunkte zu finden, die die Spanne definieren. Dies ist nicht immer die beste Lösung, da die Daten verrauscht sein können und die trennende Hyperebene das Rauschen überlagert. Daher ist es üblich, einige Datenpunkte in der Spanne zu erlauben, die als *Slack Variablen* bezeichnet werden. Diese Regularisierung wird mit dem C Parameter in `svm.SVC(...)` implementiert. Das sind die Kosten, die entstehen, wenn man zulässt, dass Supportvektoren innerhalb des Spannraums liegen. Die Zuweisung hoher Kosten (z.B. $C=1e10$) wird dazu führen, dass keine Datenpunkte innerhalb der Spanne liegen, während die Zuweisung niedriger Kosten (z.B. $C=1e-1$) zu einer weniger strengen Spanne führt.

Visualisieren Sie die Entscheidungsregionen und Support-Vektoren für $C = [2^{-6}, 2^{-3}, 1, 2^3, 2^6, 2^{12}]$ in der Methode `plot_svm_C`. Sie können dabei auf die Funktion `utils.plot_support_vectors` zurückgreifen.

8.2 SVM Multi-Label Klassifikation

Wir werden den Zifferndatensatz verwenden, um die Genauigkeit der SVM in Bezug auf die verschiedenen Hyperparameter-Setups zu messen.

a) SVM Gridsearch

Support-Vektor-Maschinen sind leistungsfähige Klassifikatoren. Es ist wichtig, eine Menge von Hyperparametern zu finden, die gut zu den Daten passen. Dazu ist es üblich, eine Hyperparametersuche z.B. über eine Gridsearch

durchzuführen, das sich über mögliche Werte für verschiedene Parameter erstreckt. Für den RBF-Kernel können wir z.B. ein Gridsearch mit den Parametern C und γ abdecken, was effektiv jede Kombination beider Parameterbereiche aufbaut.

Führen Sie eine Gridsearch in der Methode `run_grid_search` für die Bereiche $2^{-13} \leq \gamma \leq 2^{-4}$, $2^{-3} \leq C \leq 2^3$, $1 \leq d \leq 6$ durch. Sie können dabei die Funktion `sklearn.model_selection.GridSearchCV` verwenden.

8.3 K-Means Clustering

Der K-Means-Algorithmus gruppiert Daten, indem er versucht, Proben in n Gruppen gleicher Varianz einzuteilen, wodurch ein Kriterium minimiert wird, bekannt als die Trägheit (engl. inertia) oder die innere Cluster-Quadratsumme (engl. *within-cluster sum-of-squares criterion*). Dieser Algorithmus erfordert eine gegebene Anzahl von Clustern. Er lässt sich gut auf eine große Anzahl von Proben skalieren und wurde über eine große Bandbreite von Anwendungsbereichen in vielen verschiedenen Bereichen, wie z.B. Suchmaschinen, Dokumenten-Clustering, Kundensegmentierung, Sensorvernetzung und vielem mehr angewendet.

K-Means teilt einen Satz N von Datenpunkten X in K disjunkte Cluster C , jedes beschrieben durch den Mittelwert μ_j der Proben im Cluster. Die Mittelwerte sind gemeinhin als *Zentroide* bezeichnet werden; beachten Sie, dass sie es im Allgemeinen nicht Punkte aus X sind, obwohl sie sich im gleichen Raum befinden.

Der K-Means-Algorithmus zielt darauf ab, Zentroide zu wählen, die die *Trägheiten* minimieren:

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_i - \mu_j||^2).$$

Trägheit kann als ein Maß dafür erkannt werden, wie intern kohärent Cluster sind. Diese Metrik leidet jedoch unter verschiedenen Nachteilen:

- Die Trägheit trifft die Annahme, dass Cluster konvex und isotrop sind, was nicht immer der Fall ist. Es reagiert schlecht auf langgestreckte Cluster oder Krümmungen mit unregelmäßigen Formen.
- Trägheit ist keine normalisierte Metrik: wir wissen einfach, dass niedrigere Werte besser sind und Null optimal. Aber in sehr hochdimensionalen Räumen neigen die euklidischen Abstände dazu, aufgeblasen zu sein (dies ist ein Beispiel für den so genannten "Fluch der Dimensionalität"). Ausführen eines Dimensionalitätsreduktionsalgorithmus wie z.B. *Principal Component Analysis* (PCA) vor der K-Means-Clusterbildung kann dieses Problem lindern und beschleunigen die Berechnungen.

a) Datensatzvisualisierung

Beginnen wir mit einem Beispieldatensatz, der aus drei getrennte Cluster besteht. Visualisieren Sie den Datensatz (`plot_cluster_data`) und wenden Sie K-Means Clustering an (`fit_kmeans`). Stellen Sie anschließend die gefundenen Cluster in der Methode `plot_clustering` dar.

b) Iterationen

K-Means hat erfolgreich drei separate Cluster gefunden.

Nun möchten wir die Bewegung der Cluster-Zentroiden über die Funktion `plot_kmeans_iterations` nach jeder Iteration visualisieren.

c) Problemepunkte

Bei der Anwendung von K-Means gibt es einige Problemstellen wie unten aufgeführt.

Erläutern Sie jeden dieser Aspekte:

1.) Initialisierung

2.) Korrekte Anzahl von Clustern

3.) Nicht-sphärische Cluster

4.) Cluster mit ungleicher Varianz

5.) Erzwungene Clusterbildung

Literatur

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.