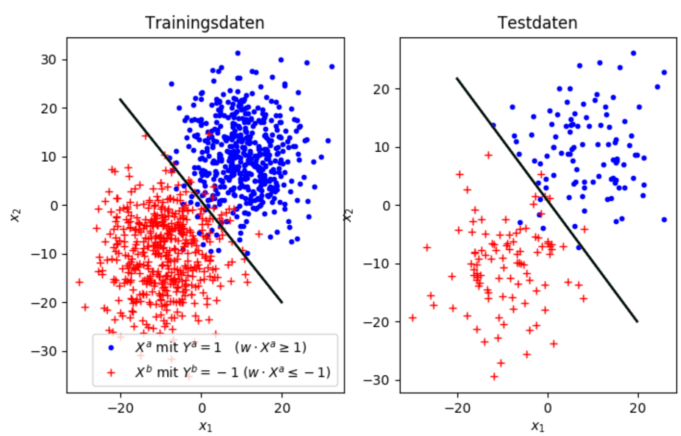
Computerpraktikum Mathematische Optimierung

2. Übungsblatt (Klassifikation mit SVMs durch ADMM)

Aufgabenstellung

Eine Support Vector Machine (SVM) ist eine mathematische Methode aus dem Gebiet des maschinellen Lernens zur Klassifikaton von Objekten in eine von zwei Klassen. Die SVM wird mit einer Anzahl von Trainingsvektoren, für welche die Klassenzugehörigkeit bekannt ist, initialisiert und berechnet für diese eine Hyperebene, die die Trainingsvektoren der beiden Klassen möglichst gut voneinander trennt. Getestet wird die SVM auf einem Testdatenset. Neue Datenvektor unbekannter Klasse werden klassifiziert, indem berechnet wird, auf welcher Seite der Hyperebene sie liegen (Anwendungsphase).



Die Daten bestehen aus einem Tupel $(\mathbf{x}^i, y^i)$ wobei $\mathbf{x}^i = (x_1^i, x_2^i) \in \mathbb{R}^2$ die Koordinaten des Datenpunktes und $y^i \in \{1, -1\}$ die Klasse bezeichnen. Zum Training liegen zwei bekannte Mengen von je $m/2$ Vektoren bereit, wobei die ersten $m/2$ Daten $\mathbf{x}^1, \dots, \mathbf{x}^{m/2}$ mit $y^i = 1$ und die zweiten $m/2$ Daten $\mathbf{x}^{m/2+1}, \dots, \mathbf{x}^m$ mit $y^i = -1$ gekennzeichnet sind. Diese sollen mittels einer Hyperebene $w^T \begin{pmatrix} 1 \\ \mathbf{x}^i \end{pmatrix} = w_1 + x_1^i w_2 + x_2^i w_3$ mit $w \in \mathbb{R}^3$ (d.h. eine Geradengleichung $x_2^i = m x_1^i + b$ mit Steigung m und Stützpunkt b), die den minimalen Abstand zu den Datenvektoren maximiert, getrennt werden. Mathematisch ausgedrückt wird dies durch eine L2-regularisierte Funktion:

$$\min_{w \in \mathbb{R}^3} \quad \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{m} \max(1 - y^i w^T \begin{pmatrix} 1 \\ \mathbf{x}^i \end{pmatrix}, 0)^2$$

Nachdem die trennende Hyperebene w berechnet wurde, können neue Vektoren durch Berechnen von $w^T \begin{pmatrix} 1 \\ \mathbf{x}^i \end{pmatrix} \begin{cases} \geq 1 & \Rightarrow \mathbf{x}^i \text{ gehört zur Klasse 1} \\ \leq -1 & \Rightarrow \mathbf{x}^i \text{ gehört zur Klasse -1} \end{cases}$ klassifiziert werden[1].

[1] Zum besseren Verständnis: www.svm-tutorial.com/2014/11/svm-understanding-math-part-2

Aufgaben

(a) Wenden Sie das ADMM-Verfahren auf die separierbare Zielfunktion

$$\begin{aligned} \min_{w\in\mathbb{R}^3, z\in\mathbb{R}^3} \quad & \frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{m} \max(1 - y^i w^T \begin{pmatrix} 1 \\ \mathbf{x}^i \end{pmatrix}, 0)^2 \\ \text{s.t.} \quad & w - z = 0 \end{aligned} \tag{1}$$

an. Der Parameter C modelliert die Wichtigkeit, die Ausreißern in den Daten bekommen.

ADMM-Algorithmus: wiederhole bis $(w_{k+1}, z_{k+1}, u_{k+1})$ konvergiert ist

1. $w_{k+1} = \text{argmin}_w\ L_\rho(w, z_k, u_k) = \text{argmin}_w \left(C\sum_{i=1}^{m} \max(1 - y^i w^T \begin{pmatrix} 1 \\ \mathbf{x}^i \end{pmatrix}, 0)^2 + \frac{\rho}{2}\|w - (z_k - u_k)\|_2^2 \right)$
2. $z_{k+1} = \text{argmin}_z\ L_\rho(w_{k+1}, z, u_k) = \text{argmin}_z\ \left(\|z\|_2^2 + \frac{\rho}{2}\|z - (w_{k+1} + u_k)\|_2^2 \right)$
3. Aufstiegsschritt der dualen Variablen: $u_{k+1} = u_k + x_{k+1} - z_{k+1}$

   (Hier haben wir die duale Variable zu $u = \frac{1}{\rho}y$ skaliert.)

(b) Implementieren Sie das ADMM-Verfahren in MATLAB, Python (mit `numpy` [2]) oder C.

Wählen Sie (zumindest für die Abgabe) $C = 0.05$ und $\rho = 1.0$. Verwenden Sie als Abbruchkriterium das Residuun $\|x_k - z_k\|_2 \leq \varepsilon$ mit $\varepsilon = 10^{-8}$.

(c) Wenden Sie Ihren Algorithmus auf die gegebenen Instanzen bestehend aus

- Zum Berechnen der Hyperebene w: Trainingsdaten $(X, Y) \in \mathbb{R}^{2\times m} \times \{0,1\}^{1\times m}$ mit $Y^1 = \ldots = Y^{m/2} = 1$ und $Y^{m/2+1} = \ldots = Y^m = -1$ wobei Y^i jeweils die Klasse von X^i definiert
- Zum Auswerten der Zielfunktion und Ergebnisplot: Testdaten $(X_{\text{test}}, Y_{\text{test}}) \in \mathbb{R}^{2\times m_{\text{test}}} \times \{0,1\}^{1\times m_{\text{test}}}$ mit $Y^1_{\text{test}} = \ldots = Y^{m_{\text{test}}/2}_{\text{test}} = 1$ und $Y^{m_{\text{test}}/2+1}_{\text{test}} = \ldots = Y^{m_{\text{test}}} = -1$

Zugriff auf die erforderlichen Daten erhalten Sie wie im 1. Übungsblatt, z.B. in Python durch

```
import numpy as np
data = np.load('uebung2_instanz.npz')
X = data['X'];Y = data['Y'];X_test = data['X_test'];Y_test = data['Y_test']
```

Geben Sie in jeder Iteration den Iterationszähler k, das Residuum sowie den Zielfunktionswert von (1) ausgewertet in jeder Iteration auf dem Testdatenset aus. Plotten[3] Sie außerdem den (i) Verlauf dieser beiden Größen und (ii) das Testdatenset und die berechnete Hyperebene w.