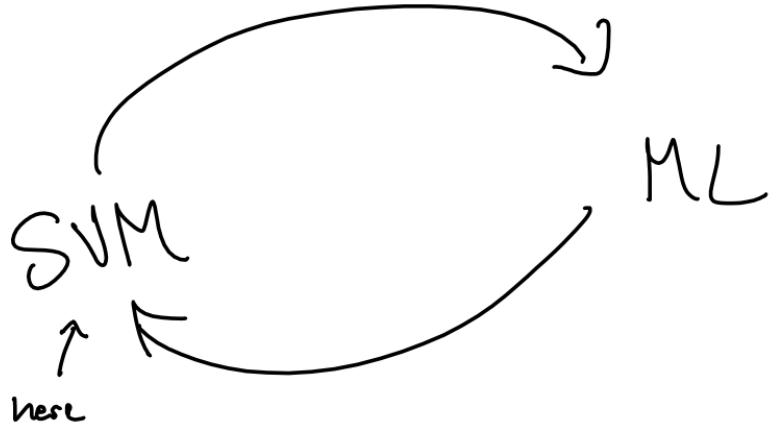
Langweiliger Teil

- Code zum Mitnehmen
- Wer kennt Python?
- Jupyter Notebook: Code und Text
- scikit-learn

Ziel



- Tief in Mathe
- Implementierung Gauss’scher Kernel
- Danach grundlegendes zu ML
- Gefühl, dass es machbar ist

Ziel (2)

- Tief in Mathe
- Implementierung Gauss’scher Kernel
- Danach grundlegendes zu ML

Training set

- Eine Matrix, ein Vektor
- Spalte, Zeile?
- 1. Tipp: Durcheinander-Würfeln
- Code erklären
- Überleitung!! (Minimierung)

Cost function

- Beste Lösung durch Minim.
- Summe, Iteration: langsam
- Lösung: Vektorisierung
- numpy Besonderheiten (func)

Feature Matrix

- B.U.N.T.-Plot
- Was fällt euch auf?
- Symmetrie, 1en auf Diag.
- Diag.: 0 im Zähler, exp(0)=1
- Sym.: Kernel symmetrisch

Minimieren (simplex)

- scipy.optimize: J() args
- Callback für Iterationsschritte (Fehler)
- J(theta) = Fehlerplot
- Minimale Lösung: Theta
- J(theta) schlecht plotbar, dims

Hypothese

- Umformen in Features (ker)
- B.U.N.T.-Plot
- Blau: y=0, Rot: y=1

Library

- Schlechte Performance
- Locker 100x schneller
- Code erklären

Linearer Kernel

- C groß, Algo sucht kleines Theta
- Kleines Theta: const. Underf
- C klein, groß Theta: overfitti.

MNIST

- 70000 handschriftliche Ziffern, 28x28
- X: 28x28=784
- Zufällige Zahlen plotten

Training/Test set

- 70% Training, 30% Test/CV
- Würfelt automatisch

Fit & Score

- Genau, wie vorher:
- Cost function, Features, Minimieren
- Score, andere Gruppen!!
- Random Predictions

Das gleiche mit Gauss’schem Kernel

Problem

- ~95% 1en haben keinen Strich
- Lösung: mehr Trainingsbsp. mit Strich
- Zahlen gleich groß, 20x20

Fehleranalyse

- Generator vom Anfang, 10000

Generalisierung

- Vereinfachte Kostenfnkt.
- groß C, Gauss-Berge klein
- 0.5 Grenze nicht erreicht
- Hälfte Trainingbsp. = Fehler
- festes Gamma, Plot

Lernkurven

- Trainings-Set-Größe plotten
- Auch mit C geplottet
- Je größer m, desto kleiner kann C werden
- Erkennbar: gutes m=10^2.5

Lernkurven (2)

- Rot-Grün Plots hohe Varian.
- Manuell vs. Library, Random Splits, Permutation Tests
- Gute Trainingsgröße: 10^2.5
- = ca. 300 Trainingsbsp.

Lessons learned

- Nicht selbst implementieren
- Zufall ist immer gut
- Gute Trainingsdaten
- Fehleranalyse, neg. Beispiele
- Immer mehrmals rechnen
- Klausurrelevant