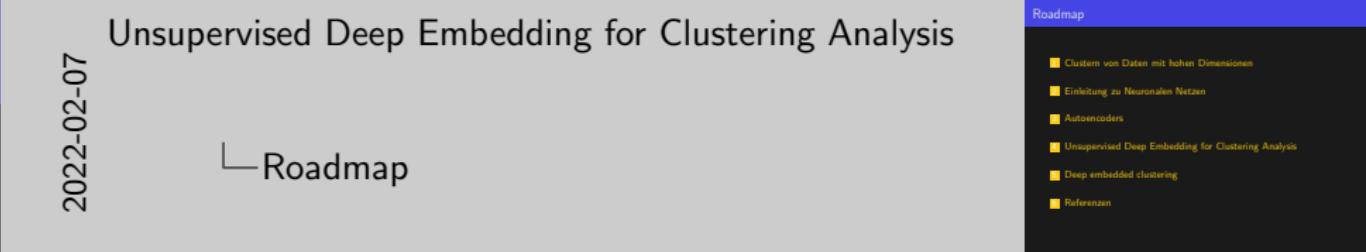
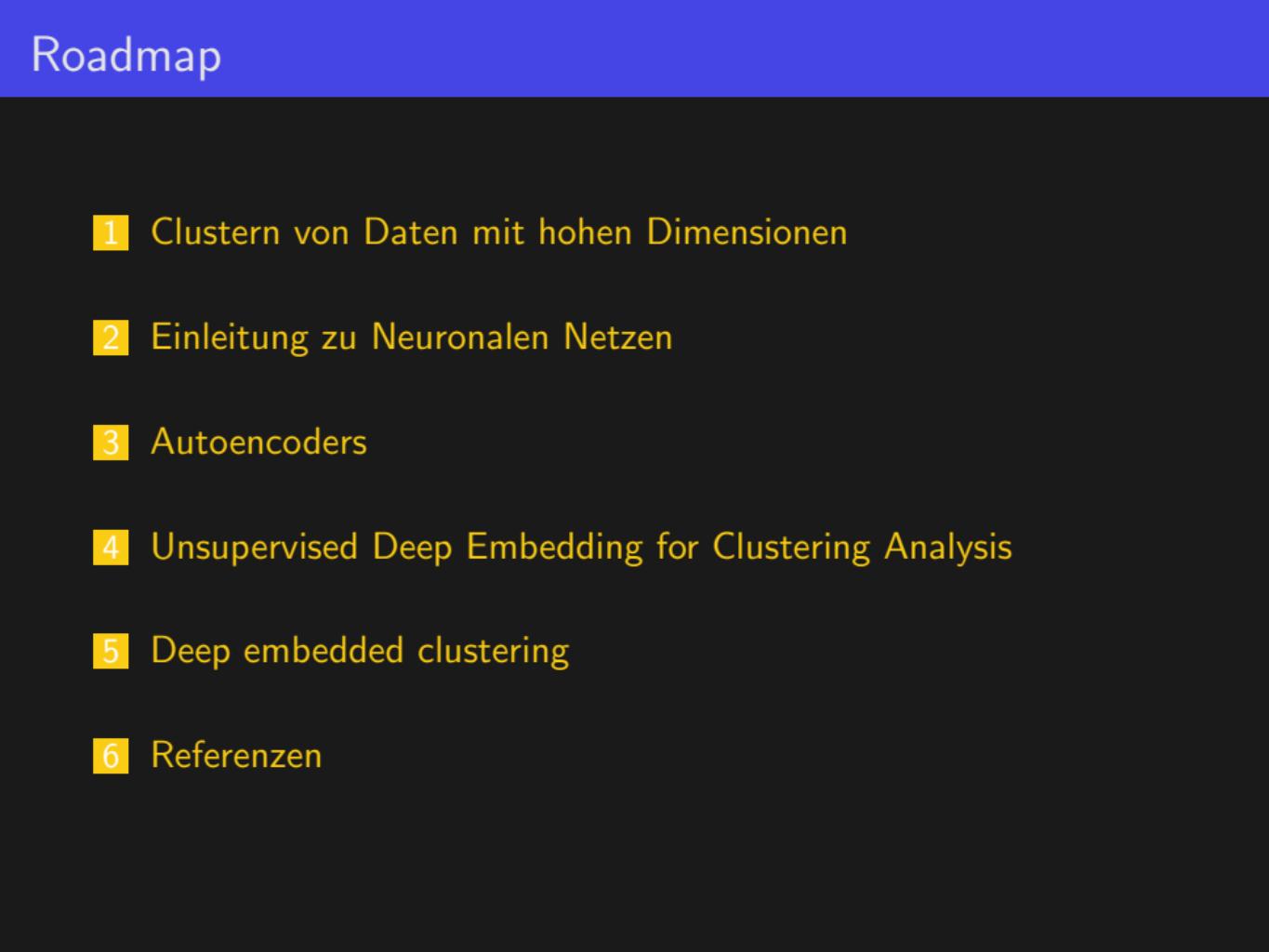


Unsupervised Deep Embedding for Clustering Analysis

Bachelorseminar Data Mining

Lukas Mahr

Ludwig-Maximilians-Universität München



■ Problem

■ Gaussian Mixture Models, KMeans

- Abstandsmetriken sind beschränkt auf den ursprünglichen Datendimensionen
- unwirksam wenn Datendimension hoch sind[1]

■ Variationen von KMeans für Daten mit hohen Dimensionen

- limitiert zu linearen Embeddings[2]

■ Spectral clustering

- Quadratische oder Super-quadratische Komplexität

■ Idee

■ Neuronalesnetzwerk zum reduzieren der Dimensionen

- nicht lineares mapping

■ Clustern der reduzierten Daten

- einfaches Clustering möglich, da Dimensionen reduziert

■ Verbessern des NN und der Cluster durch Backpropagation

└ Clustern von Daten mit hohen Dimensionen

└ Clustern von Daten mit hohen Dimensionen

2022-02-07

viele Daten Punkte viele Distanzen zu berechnen schwierig zu visualisieren
ohne die Dimensionen zu reduzieren Komplexität von Kmeans die exponentiell ansteigt

- Problem
 - Gaussian Mixture Models, KMeans
 - Abstandsmetriken sind beschränkt auf den ursprünglichen Dimensionen
 - unwirksam wenn Dimensionen hoch sind[1]
 - Variationen von KMeans für Daten mit hohen Dimensionen
 - limitiert zu linearen Embeddings[2]
 - Spectral clustering
 - Quadratische oder Super-quadratische Komplexität
- Idee
 - Neuronalesnetzwerk zum reduzieren der Dimensionen
 - nicht lineares mapping
 - Clustern der reduzierten Daten
 - einfaches Clustering möglich, da Dimensionen reduziert
 - Verbessern des NN und der Cluster durch Backpropagation

Neuronale Netze

Einleitung zu Neuronalen Netzen

Nicht-lineare statistische Modelle zur
Informationsverarbeitung

Informationsverarbeitung umfasst hierbei unter anderem

Klassifikation

Prognosenerstellung

Units der Neuronalen Netze angelehnt an Neuronen

Inputs zusammenfassen

Mit Schwellenwert vergleichen bzw. aktivieren

Verbindungen zwischen Units angelehnt an Synapsen

Gewichtung mit verstärkender oder schwächender Wirkung

Unsupervised Deep Embedding for Clustering Analysis

└ Einleitung zu Neuronalen Netzen

└ Idee

└ Einleitung zu Neuronalen Netzen

2022-02-07

Einleitung zu Neuronalen Netzen

Nicht-lineare statistische Modelle zur
Informationsverarbeitung
Informationsverarbeitung umfasst hierbei unter anderem
Klassifikation
Prognosenerstellung
Units der Neuronalen Netze angelehnt an Neuronen
Inputs zusammenfassen
Mit Schwellenwert vergleichen bzw. aktivieren
Verbindungen zwischen Units angelehnt an Synapsen
Gewichtung mit verstärkender oder schwächender Wirkung

wofür braucht man neuronale Netze

- Klassifizierung von Daten
- Prognose eines bestimmten Wertes
- Neuronen an Neuronen in unserem Gehirn angelehnt
- Verbindungen zwischen Künstlichen Neuronen an Synapsen angelehnt
- supervised learning = überwachtes lernen
- man besitzt Daten mit Beschriftungen (labels)
- unsupervised learning nicht überwachtes lernen
- unbeschriftete Daten, also es sind keine labels vorhanden

Einleitung zu Neuronalen Netzen

Künstlichen Neurons

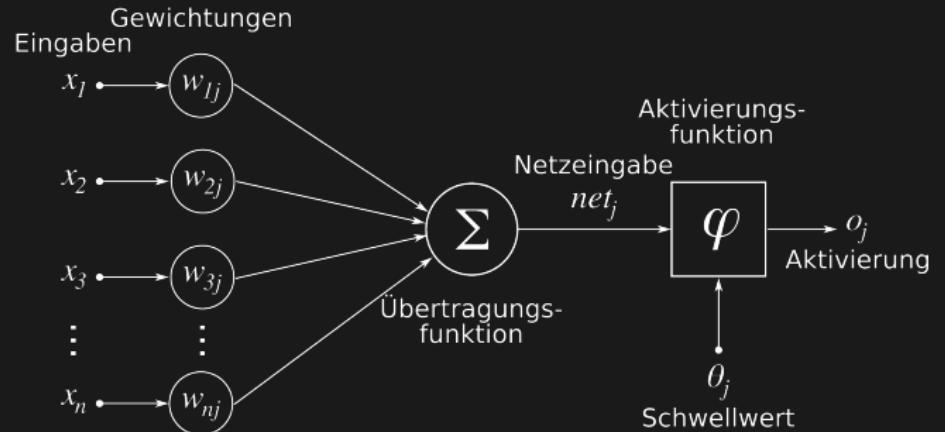


Figure: Darstellung eines künstlichen Neurons mit seinen Elementen

https://de.wikipedia.org/wiki/Datei:ArtificialNeuronModel_deutsch.png

Unsupervised Deep Embedding for Clustering Analysis

- └ Einleitung zu Neuronalen Netzen
 - └ Künstliches Neuron
 - └ Einleitung zu Neuronalen Netzen

2022-02-07



x_1, \dots, x_n sind die input variablen, jede der Eingabe variablen besitzt ein Gewicht, w_{1j}, \dots, w_{nj} . Diese werden Multipliziert und davon dann die summe berechnet. Hier die Übertragungsfunktion. Dazu wird ein Bias, in dem Fall der Schwellenwert gerechnet. Als letztes gibt es noch die Aktivierungsfunktion die meistens einen Wert zwischen 0 und 1 zurückgibt. Das ist dann der input für das nächste Neuron.

Einleitung zu Neuronalen Netzen

Layer/Schichten

The diagram illustrates a deep learning neural network structure. It consists of five layers: an input layer with 784 neurons, two hidden layers (Hidden Layer 1 with 128 neurons using the ReLU activation function, and Hidden Layer 2 with 64 neurons also using ReLU), an output layer with 10 neurons using softmax activation for classification, and a loss layer at the end using cross-entropy loss. The layers are fully connected to each other, with arrows indicating the flow of data from left to right. The input layer is shown with a grid of 28x28 neurons, representing a single image. The output layer is shown with 10 neurons, representing the predicted classes.

Figure: Deep learning Künstliches neuronales Netz maschinelles lernen
Apache MXNet - mehrschichtige PNG <https://de.cleanpng.com/png-x3zkr7/>

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis

- Einleitung zu Neuronalen Netzen
- Layer/Schicht
- Einleitung zu Neuronalen Netzen

Einleitung zu Neuronalen Netzen

Layer/Schichten

The diagram shows a neural network with five layers: Input Layer, Hidden Layer 1 (ReLU), Hidden Layer 2 (ReLU), Output Layer (softmax), and Loss Layer (cross-entropy). The Input Layer has 784 neurons. The Hidden Layers 1 and 2 each have 128 neurons and use the ReLU activation function. The Output Layer has 10 neurons and uses softmax activation. The Loss Layer is shown with a single neuron. The layers are fully connected, with arrows indicating the flow of data. The diagram is labeled "Figure: Deep learning Künstliches neuronales Netz maschinelles lernen Apache MXNet - mehrschichtige PNG" and "https://de.cleanpng.com/png-x3zkr7/".

Aktivierungsfunktionen



Figure: Rectifier-Aktivierungsfunktion

https://de.wikipedia.org/wiki/Datei:Activation_rectified_linear.svg

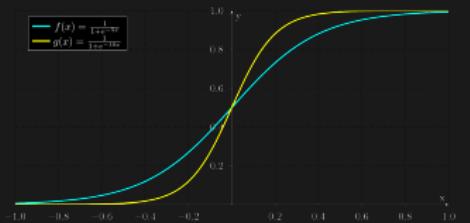
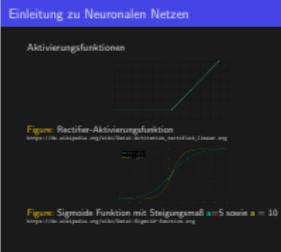


Figure: Sigmoid Funktion mit Steigungsmaß $a=5$ sowie $a = 10$

<https://de.wikipedia.org/wiki/Datei:Sigmoid-function.svg>

└ Einleitung zu Neuronalen Netzen
 └ Aktivierungsfunktion
 └ Einleitung zu Neuronalen Netzen

2022-02-07



alles negativ ist wird bei relu zu 0 während bei sigmoid, abhängig von der Steigung Werte zwischen -1 und 1 möglich sind

Loss/Kostenfunktion

Mean Squared Error

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Mean absolute error

$$MAE = \frac{\sum_{i=1}^n |\hat{y}_i - y_i|}{n}$$

Binary Cross-Entropy

$$H(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^n y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

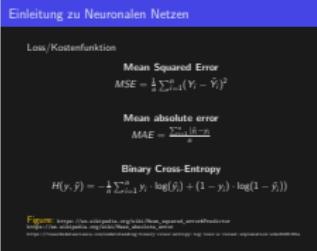
Figure: https://en.wikipedia.org/wiki/Mean_squared_error#Predictor

https://en.wikipedia.org/wiki/Mean_absolute_error

<https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>

2022-02-07

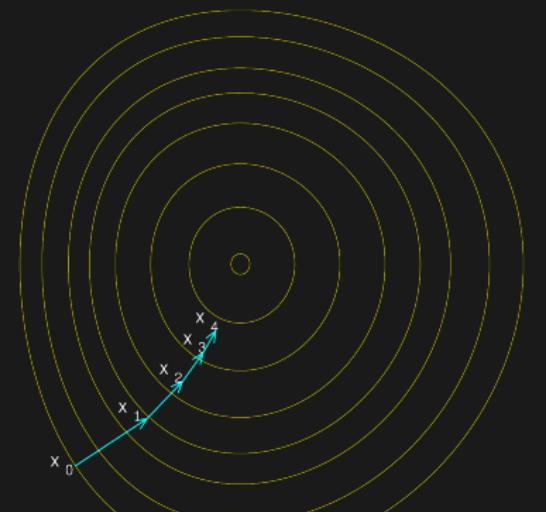
- └ Einleitung zu Neuronalen Netzen
 - └ Loss/Kostenfunktion
 - └ Einleitung zu Neuronalen Netzen



Man berechnet immer den Unterschied zwischen den wahren Labels und den predicteden Labels um zu erkennen wie weit diese auseinander liegen. Es wird immer versucht den Loss zu minimieren. Also ein Minimum der Kostenfunktion zu finden. Die Parameter der Funktion, welche angepasst werden müssen sind alle weights und biases der einzelnen Neuronen und den Layern.

Einleitung zu Neuronalen Netzen

Backpropagation mit Gradient descent



The image shows a series of concentric circles on a black background, representing level sets of a function. A small white circle at the center represents the minimum of the function. A path of points x_0, x_1, x_2, x_3, x_4 is shown, with arrows indicating the direction of movement. The path starts at x_0 on the outermost circle and moves inward, illustrating the iterative steps of the gradient descent algorithm.

Figure: Illustration of gradient descent on a series of level sets
https://en.wikipedia.org/wiki/File:Gradient_descent.svg

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis

- └ Einleitung zu Neuronalen Netzen
- └ Backpropagation mit Gradient descent
- └ Einleitung zu Neuronalen Netzen

Einleitung zu Neuronalen Netzen

Backpropagation mit Gradient descent

Figure: Illustration of gradient descent on a series of level sets
https://en.wikipedia.org/wiki/File:Gradient_descent.svg

Über Backpropagation wird hier mit z.B Gradient Descent die Loass funktion minimiert. Der Gradient der Loss/ Kostenfunktion wird für alle weights and biases gleichzeitig berechnet. Man kann sich das vorstellen, wie eine Kugel die man einen in einer Hügellandschaft rollen lässt ein kleinen schritten und zwischen den schritten immer nach der Steigung des Abhanges schaut und dabei versucht die Kugel in das tiefste Tal zu bekommen.

Autoencoder

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis
└ Autoencoders
 └ Idee
 └ Autoencoders



Ein Autoencoder ist ein feedforward Neural network, was versucht den input zu Kopieren. Das hört sich im ersten Moment nutzlos an, hat aber doch ein paar Anwendungsmöglichkeiten. Dazu gehört z.B Denosing oder Reduktion des Features Spaces

Autoencoders

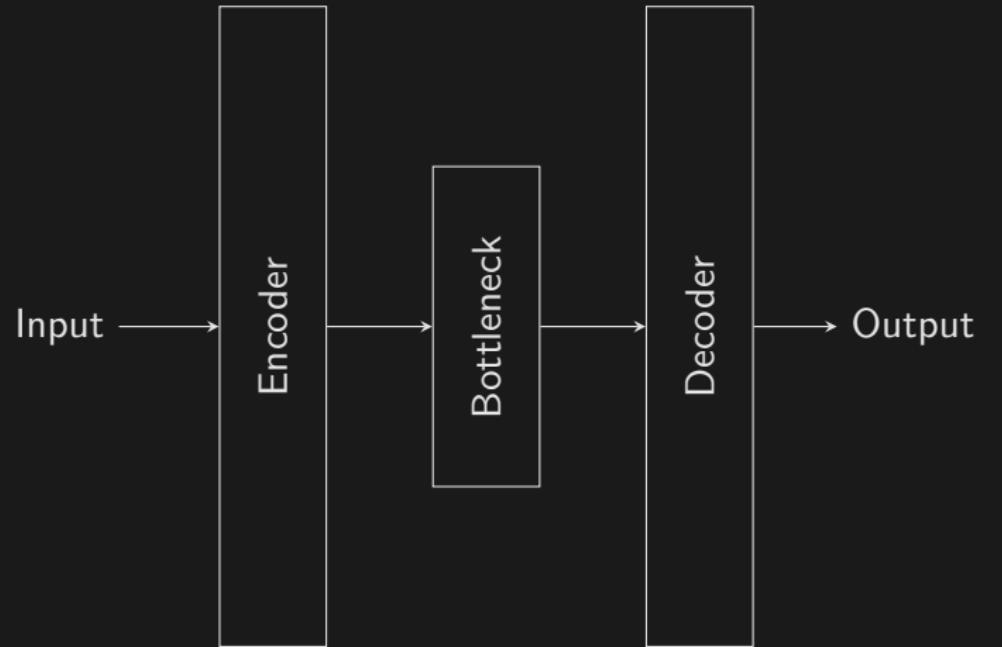


Figure: Einfaches Autoencoder Model

Unsupervised Deep Embedding for Clustering Analysis
└ Autoencoders
 └ Aufbau
 └ Autoencoders

2022-02-07

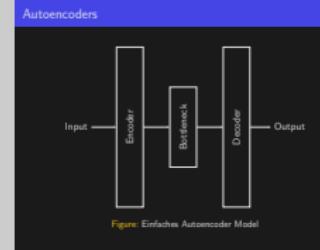


Figure: Einfaches Autoencoder Model

- Aufbau, Das Netzwerk besteht im Grunde aus 2 Teilen.
- Encoder und Decoder
- Encoder Transformiert die Input Daten in eine kleinere gewünschte Dimension
- Decoder Transformiert die Daten aus der Kleinen Dimension zurück in die Orginal Dimensionen.
- Hoffnung das der Encoder die Daten auf die Wichtigsten Features reduziert.

Autoencoders

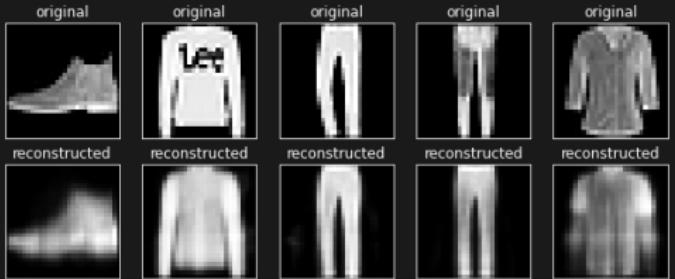


Figure: Orginal und Decoded Bilder

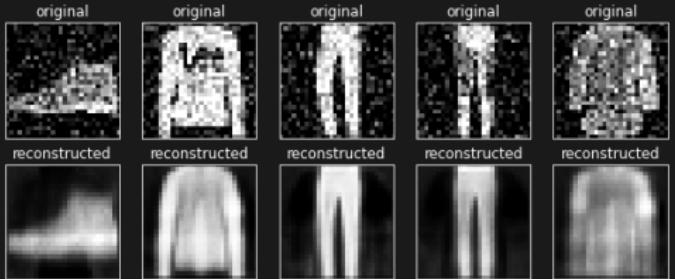


Figure: Noisy und Decoded Bilder

<https://github.com/Plutokekz/dec/blob/main/Autoencoders.ipynb>

Unsupervised Deep Embedding for Clustering Analysis

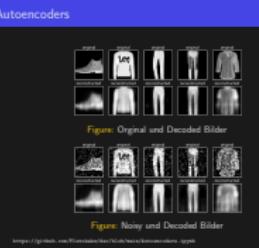
Autoencoders

Aufbau

Autoencoders

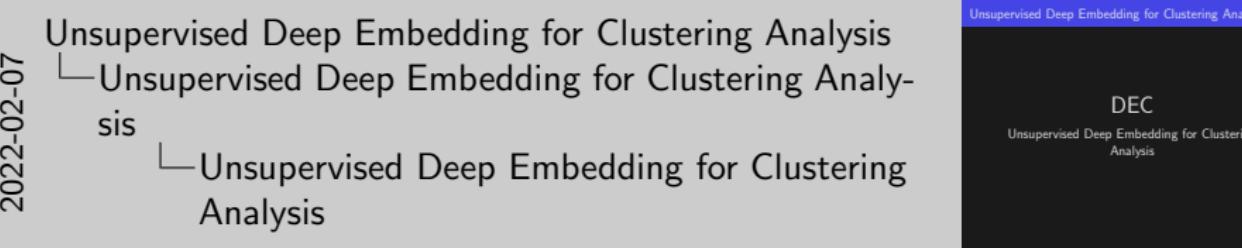
2022-02-07

- 784 input zu 10 zu 784 (28*28)
- Bottleneck size = 10 oder feature reduktion auf 10
- Random noise



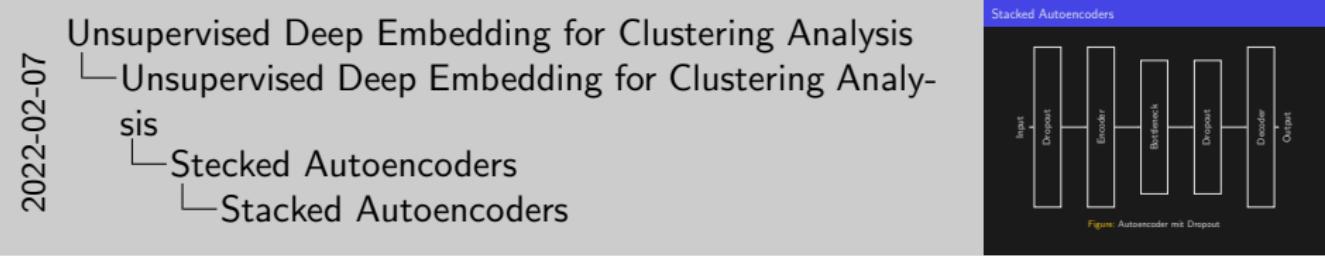
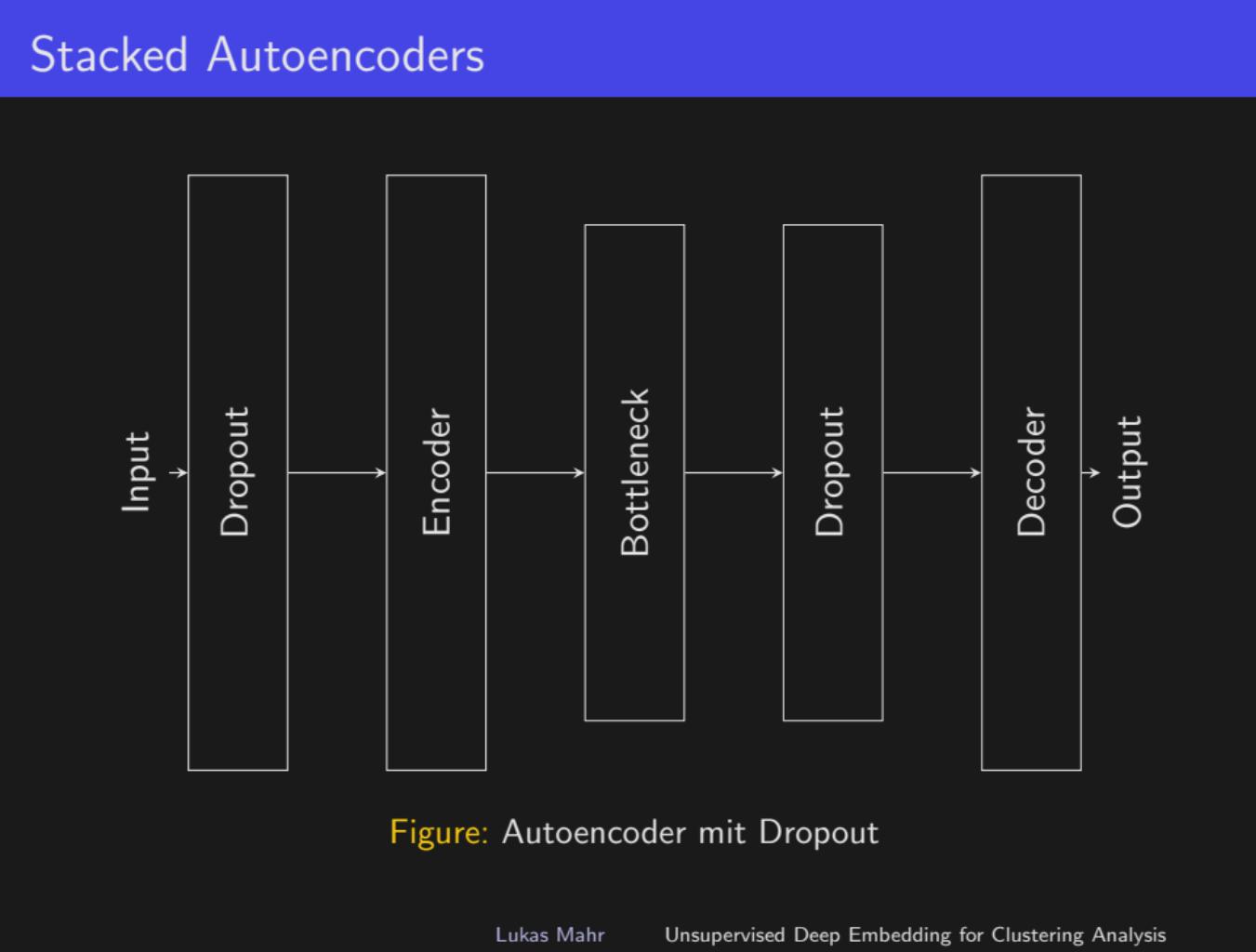
DEC

Unsupervised Deep Embedding for Clustering Analysis



DEC has two phases: (1) parameter initialization with a deep autoencode parameter optimization (i.e., clustering), where we iterate between computing an auxiliary target distribution and minimizing the Kullback–Leibler (KL) divergence to it.

- Besteht aus 2 teilen
- embedding mapping in den kleineren raum Z
- Stacked Autoencoders
- clustering
- Clustern der embddedten Daten — Embeddings verbessern so wie die Cluster anpassen



Stacked Autoencoders

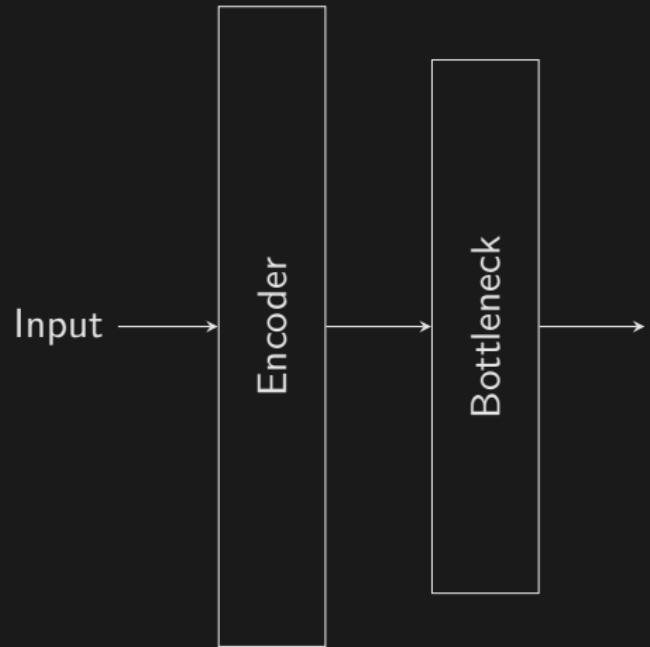
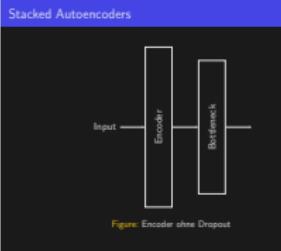


Figure: Encoder ohne Dropout

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis
└ Unsupervised Deep Embedding for Clustering Analysis
 └ Stecked Autoencoders
 └ Stacked Autoencoders

Encoder teil ohne noise/dropout und ohne dem decoder



Stacked Autoencoders

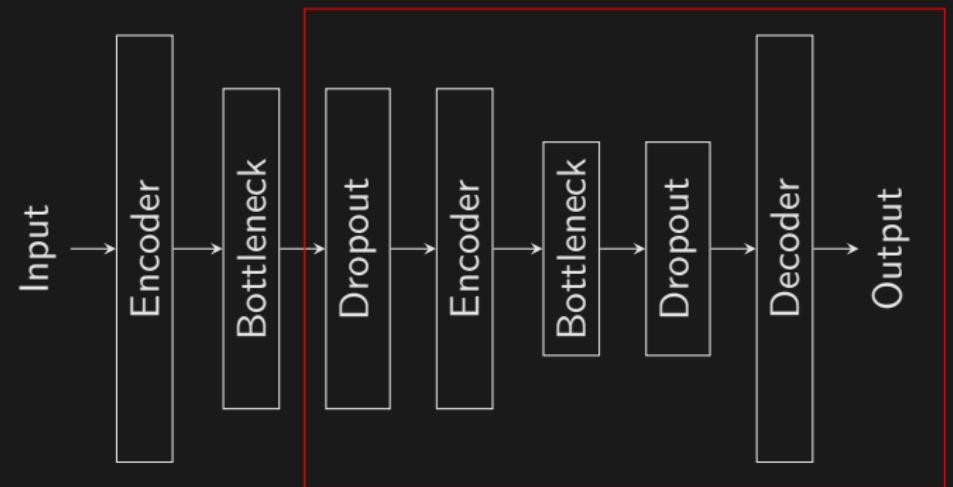
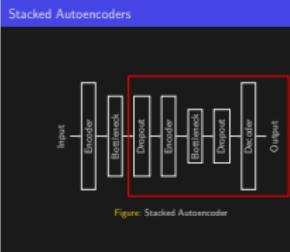


Figure: Stacked Autoencoder

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis
└ Unsupervised Deep Embedding for Clustering Analysis
 └ Stecked Autoencoders
 └ Stacked Autoencoders



Aktivierungsfunktion relu außer im letzten Decoder und Encoder Layer
sigmoid

für die zero-mean images

Zero-mean images - \bar{i} durchschnitt pro pixel über alle bilder im Datensatz
das bild wird dann von jedem bild abgezogen und damit liegt der durschnitt
bei den Bilder bei null deswegen zero mean images.

loss = least-square

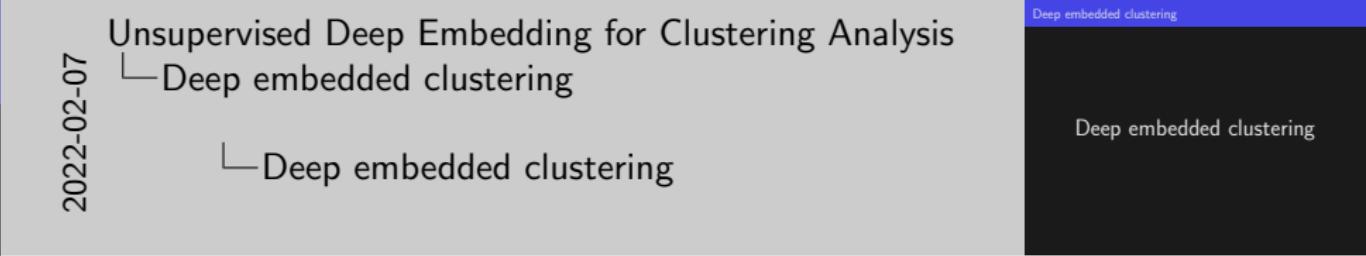
nach dem Training des Autoencoders wird der Encoder als nicht lineares
mapping für die daten genutzt die man Clustern möchte

Netzwerk Aufbau input-500-500-2000-10

50000 epochs pro layer

100000 ganzes Netz ohne dropout epochs fine tuning

256 batchsize



Deep embedded clustering

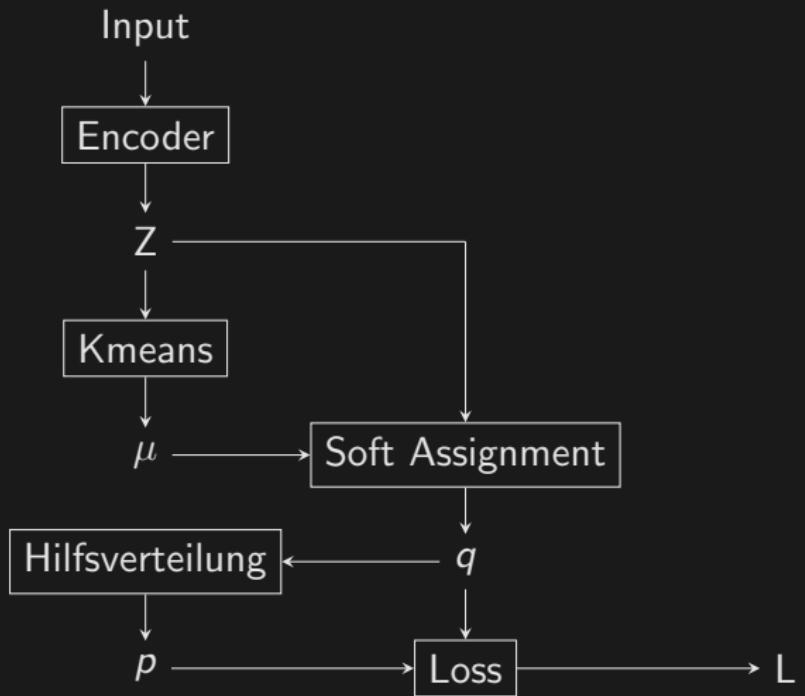


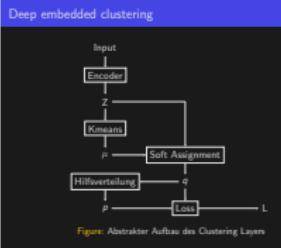
Figure: Abstrakter Aufbau des Clustering Layers

Unsupervised Deep Embedding for Clustering Analysis

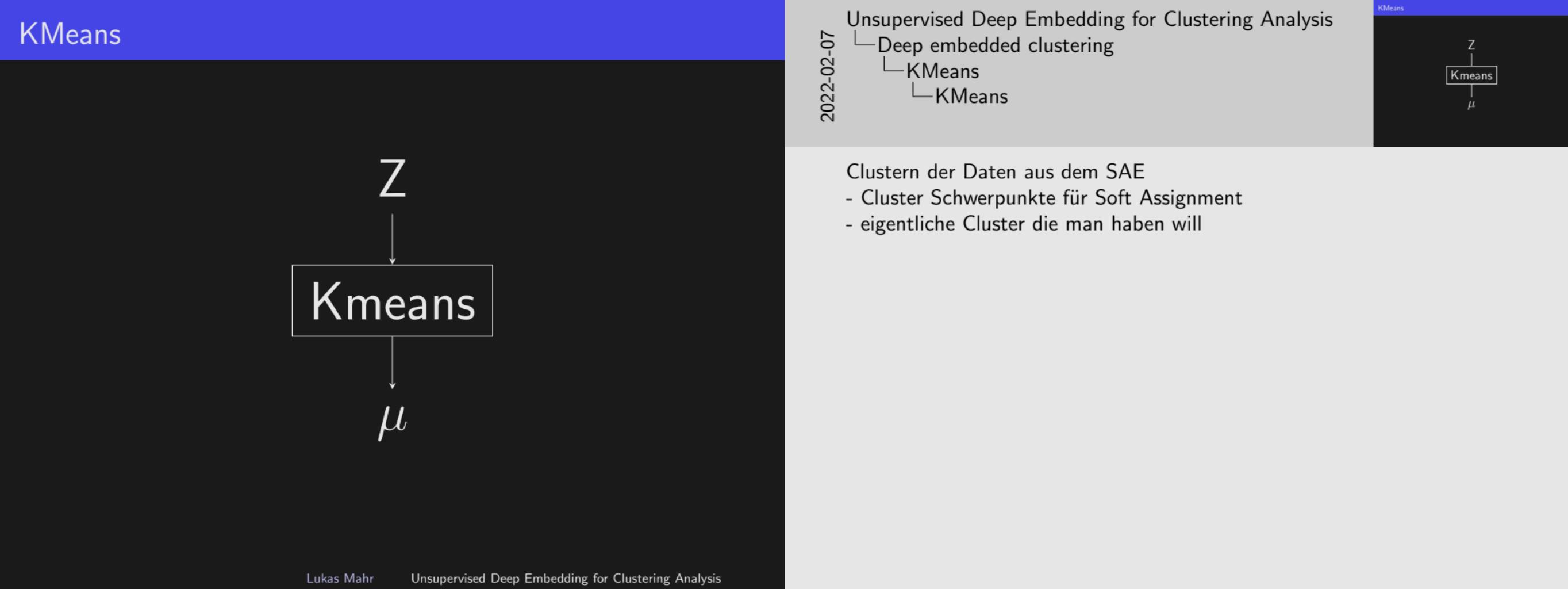
Deep embedded clustering

Deep embedded clustering

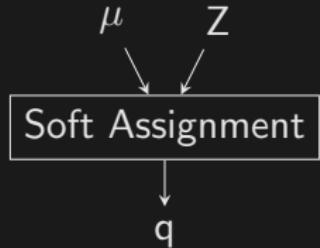
2022-02-07



- input daten durch sae werden auf kleinen feature raum Z abgebildet
- clustern der Daten Z es entstehen die Cluster Schwerpunkte μ
- soft assignments zwischen μ und z, wird zu q
- Hilfsverteilung aus q , wird zu p
- Loss zwischen q und p



Soft Assignments

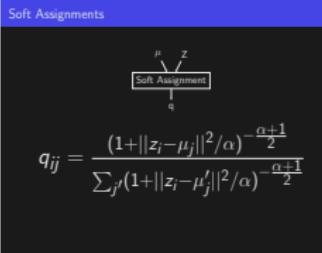


$$q_{ij} = \frac{(1+||z_i - \mu_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1+||z_i - \mu'_{j'}||^2/\alpha)^{-\frac{\alpha+1}{2}}}$$

Unsupervised Deep Embedding for Clustering Analysis

2022-02-07

- Deep embedded clustering
- Soft Assignments
- Soft Assignments



Student t-Verteilung um die Ähnlichkeit zwischen z_i und μ_j zu messen
Freiheitsgrad ist 1[3]

q_{ij} ist die Wahrscheinlichkeit die z_i dem Cluster q_j zuzuordnen

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}$$

$$f_j = \sum_j q_{ij}$$



2022-02-07

└ Deep embedded clustering
 └ Hilfsverteilung
 └ Hilfsverteilung



$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}$$
$$f_j = \sum_j q_{ij}$$

Eigenschaften der Hilfsverteilung

- Prognosen Verstärken
- mehr Gewicht auf genaue Datenpunkte
- loss Verteilung von jedem Cluster Schwerpunkt normalisieren um zu verhindern das der versteckte Merkmals Bereich nicht verzehrt wird.
 - man will das letzte Encoder Layer schützen.

f_j Soft-Cluster-Frequenzen

Erklärung wie die Eigenschaft erreicht werden im Fazit.

Loss

Kullback-Leibler-Divergenz


$$L = KL(P||Q) \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis

- Deep embedded clustering
- Loss
- Loss

Kullback-Leibler-Divergenz


$$L = KL(P||Q) \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

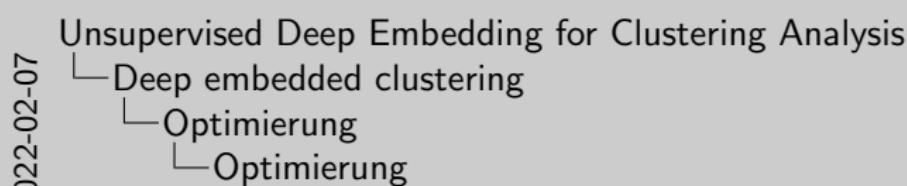
Kullback-Leibler-Divergenz oder Kullback-Leibler-Abstand

- maß für Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen
- hier als loss funktionen
- unterschied zwischen q und p
- der unterschied wird versucht zu minimieren.

$$\frac{\partial L}{\partial z_i} = \frac{\alpha+1}{\alpha} \sum_j \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j)$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha+1}{\alpha} \sum_i \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j)$$

Stochastic Gradient Descent



Optimierung

Stochastic Gradient Descent

$$\frac{\partial L}{\partial z_i} = \frac{\alpha+1}{\alpha} \sum_j \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j)$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha+1}{\alpha} \sum_i \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j)$$

Gradient von L wird berechnet in Abhängigkeit von q_i und μ_j
 $\frac{\partial L}{\partial z_i}$ wird dann für backpropagation genutzt.
wird Optimiert bis eine bestimmter Prozentsatz an Cluster punkten nicht mehr das Cluster wechselt, zwischen zwei fortlaufenden Iterationen.

MNIST



mnist

- 70k Handgeschriebene Zahlen
- von 0 - 9
- 28*28 pixel

STL-10

STL-10

- 10 Klassen
- 1300 Bilder pro Klasse

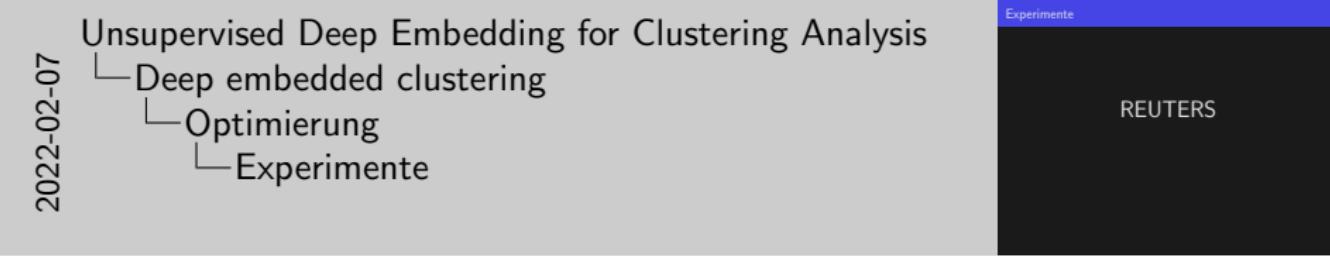
zusätzlich:

- 100000 Unbeschriftete Bilder
- haben beide benutzt
- aber gepreprocessed
- HOG features mit einem 8*8 filter

HOG:

- Histogram of Oriented Gradients
- Ausrichtung der Kanten und des Gradienten in 8*8 pixel blöcken

REUTERS



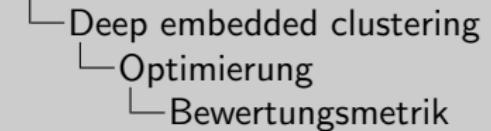
Reuters:

- 810000 English Nachrichtenberichte
- in Kategorie Bäume eingeteilt
- 4 Haupt Kategorien (corporate/industrial, government/social, markets, economics)
- alle die mehr Haupt Kategorien hatten sind raus geflogen
- 685071 übrig
- tf-idf von den 2000 häufigsten Wort Stämmen
- Term Frequency-Inverse Document Frequency
- wie wichtig ein Wort in einem Satz von einem Text ist.

$$ACC = \max_m \frac{\sum_{i=1}^n 1\{l_i=m(c_i)\}}{n}$$

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis



Bewertungsmetrik

$$ACC = \max_m \frac{\sum_{i=1}^n 1\{l_i=m(c_i)\}}{n}$$

- l_i das wahre label
 - c_i Cluster Zuordnung
 - m alle one to one Zuordnungen zwischen Clustern und Labels
- Findet das beste mapping zwischen Cluster und Label

Ergebnisse

Methode	MNIST	STL-HOG	REUTERS-10K	REUTERS
k-means	53.49%	28.39%	52.42%	53.29%
LDMGI	84.09%	33.08%	43.84%	N/A
SEC	80.37%	30.75%	60.08%	N/A
DEC w/o	79.82%	34.06%	70.05%	69.62%
DEC	84.30%	35.90%	72.17%	75.63%

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis

- Deep embedded clustering
 - Optimierung
 - Ergebnisse

Ergebnisse				
Methode	MNIST	STL-HOG	REUTERS-10K	REUTERS
k-means	53.49%	28.39%	52.42%	53.29%
LDMGI	84.09%	33.08%	43.84%	N/A
SEC	80.37%	30.75%	60.08%	N/A
DEC w/o	79.82%	34.06%	70.05%	69.62%
DEC	84.30%	35.90%	72.17%	75.63%

erklären der anderen Algorithmen und deren Start Bedingungen

Kmeans:

20 restart bestes Ergebnis wird genommen

LDMGI:

Image Clustering using Local Discriminant Models and Global Integration

Gleiche Parameter wie in deren Paper

SEC:

Spectral Clustering mit Linearen Embeddings

Gleiche Parameter wie in deren Paper

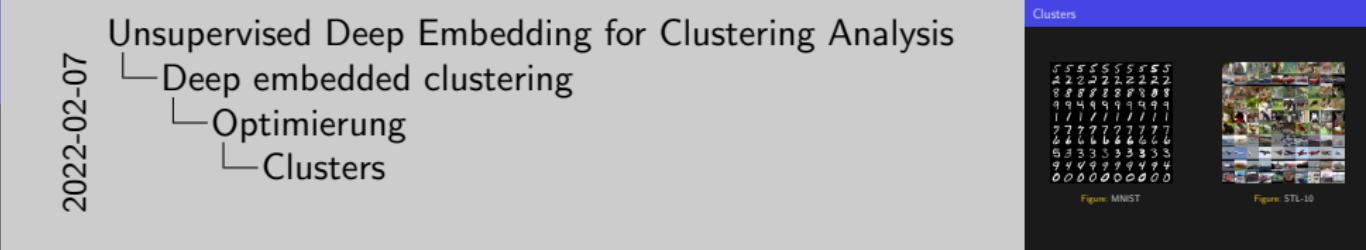
1 Hyperparameter für jeden Algorithmus mit 9 Auswahlmöglichkeiten
das beste Ergebnis wird genommen.

DEC wird λ variiert, Kontrolliert den annealing speed.

LDMGI und SEC konnten REUTERS nicht verarbeiten, weil zu viel RAM benötigt wurde.

Lukas Mahr

Unsupervised Deep Embedding for Clustering Analysis



reihe:

10 besten cluster Ergebnisse von einem Cluster

mnist:

4 und 9 landen im gleichen Cluster da sie so ähnlich aussehen.

stl:

meistens richtig: Flugzeuge, LKWs, Autos

Tiere wurden an Tiere Pose geclusterd und nicht der Art.

Liegt wahrscheinlich daran das sie HOG benutzt haben um features zu extrahieren

Autoencoder Features

Methode	MNIST	STL-HOG	REUTERS-10K	REUTERS
AE+kmeans	81.84%	33.92%	66.59%	71.97%
AE+LDMGI	83.98%	32.04%	42.92%	N/A
AE+SEC	81.56%	32.29%	61.86%	N/A
DEC	84.30%	35.90%	72.17%	75.63%

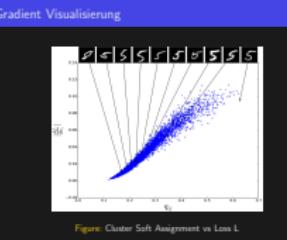
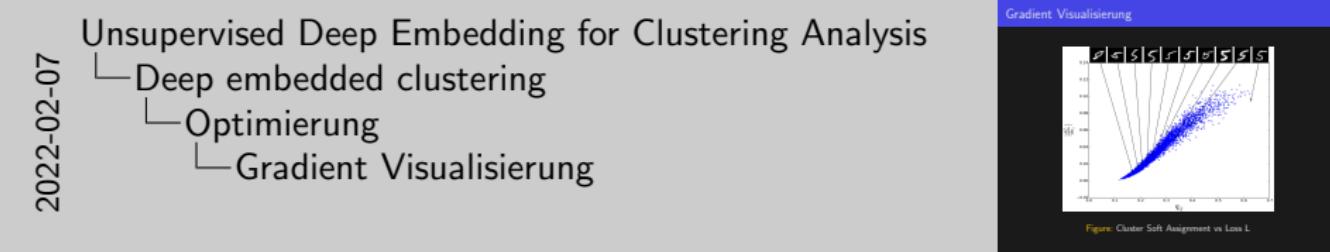
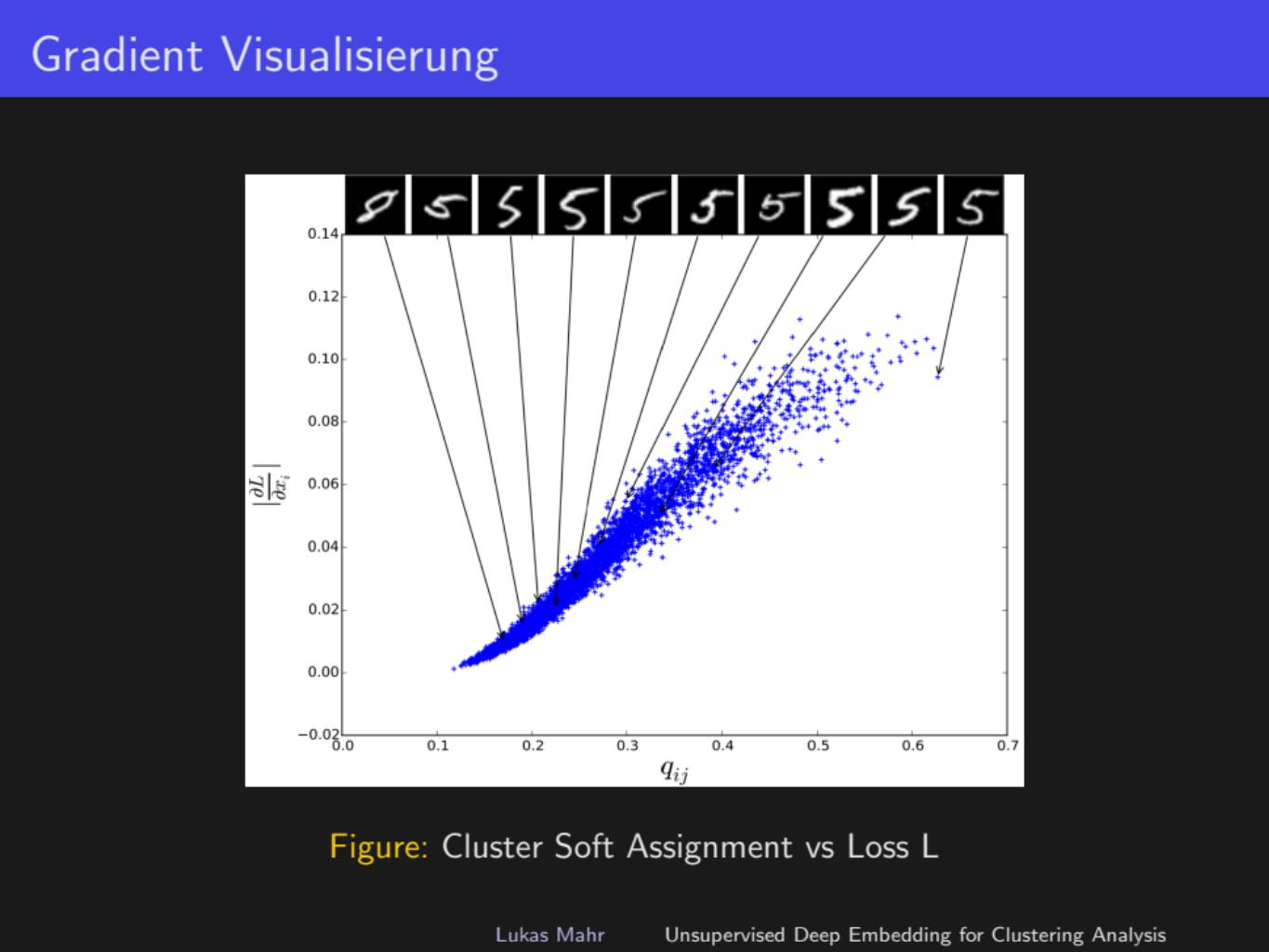
2022-02-07

Unsupervised Deep Embedding for Clustering Analysis

- Deep embedded clustering
- Optimierung
- Autoencoder Features

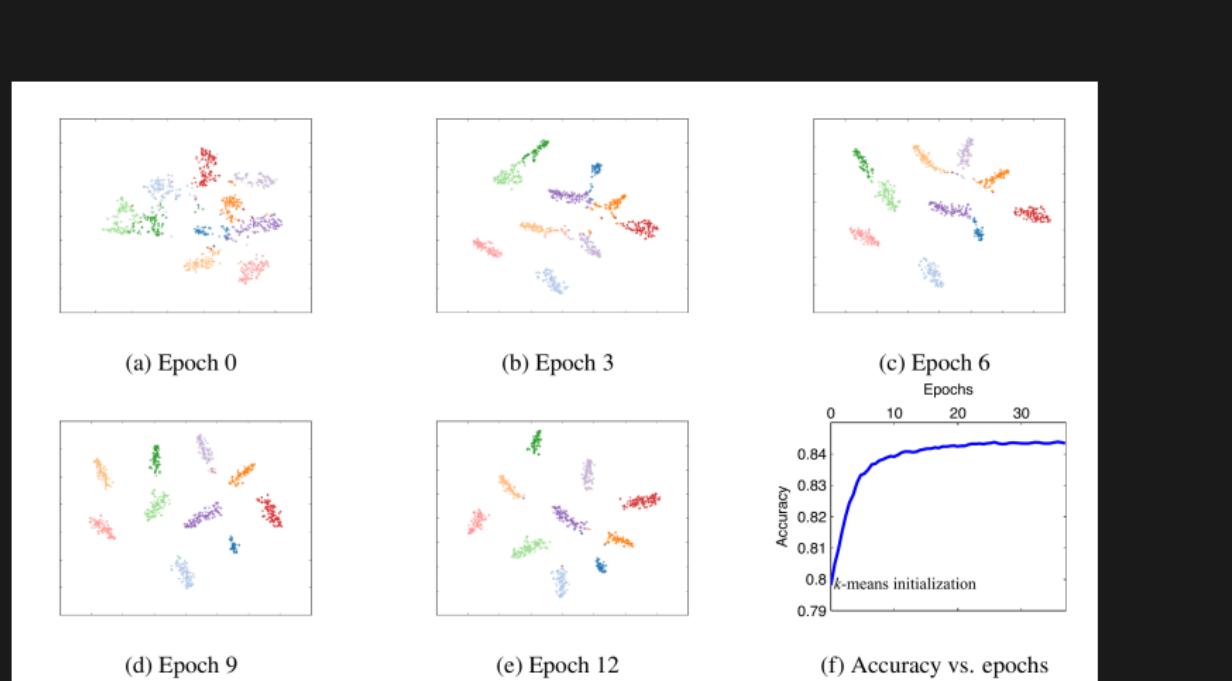
Autoencoder Features				
Methode	MNIST	STL-HOG	REUTERS-10K	REUTERS
AE+kmeans	81.84%	33.92%	66.59%	71.97%
AE+LDMGI	83.98%	32.04%	42.92%	N/A
AE+SEC	81.56%	32.29%	61.86%	N/A
DEC	84.30%	35.90%	72.17%	75.63%

Alles ist besser mit einen Autoencoder.
keine angaben zum Autoencoder
kein stacked autoencoeder
ist dec wirklich soooo viel besser ??



Richtigkeit von Hilfsverteilung P und Punkte die nähere am Cluster Mittelpunkt sind haben mehr Einfluss auf den Gradienten.
punkte die weiter weg sind, weniger Einfluss, und 5 wird zur 8.

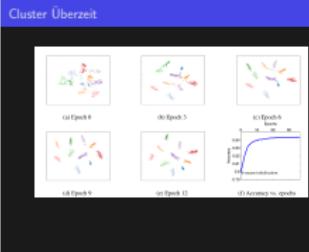
Cluster Überzeit



2022-02-07

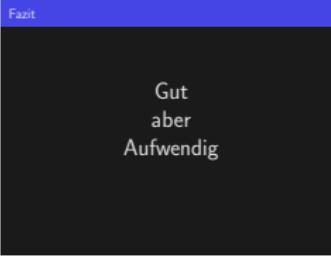
Unsupervised Deep Embedding for Clustering Analysis

- Deep embedded clustering
- Optimierung
- Cluster Überzeit



Man kann beobachten das die Cluster sich über zeit verbessern.
genauigkeit geht hoch.
nach wenigen Epochs sichtbare Verbesserung.
30 Epochs sind nicht viele
spricht für DEC.

Gut
aber
Aufwendig



sehr aufwendig um etwas zu Clustern, wegen SAE
Weiterentwicklung von Kmeans, da eigentlich nur SAE weiter verbessert wird.
eigentliches Clustering Kmeans.



Von wem ist das Paper

Junyuan Xie
University of Washington

Ross Girshick
Facebook AI Research (FAIR)

Ali Farhadi
University of Washington

Referenzen

Steinbach, Michael, Ertöz, Levent, and Kumar, Vipin. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, pp. 273–309. Springer, 2004.

Ye, Jieping, Zhao, Zheng, and Wu, Mingrui. Discriminative k-means for clustering. In *NIPS*, 2008.

van der Maaten, Laurens. Learning a parametric embedding by preserving local structure. In *International Conference on Artificial Intelligence and Statistics*, 2009.

2022-02-07

Unsupervised Deep Embedding for Clustering Analysis

Referenzen

Steinbach, Michael, Ertöz, Levent, and Kumar, Vipin. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, pp. 273–309. Springer, 2004.

Ye, Jieping, Zhao, Zheng, and Wu, Mingrui. Discriminative k-means for clustering. In *NIPS*, 2008.

van der Maaten, Laurens. Learning a parametric embedding by preserving local structure. In *International Conference on Artificial Intelligence and Statistics*, 2009.