ETH zürich

Übungslektion 14 – Machine Learning III

Informatik II

27. / 28. Mai 2025

Heutiges Programm

- Convolutional Neural Networks CNNs
- K-means Clustering
- Dimensionsreduktion

1. Recap: Convolutional Neural Networks (CNNs)

Warum CNNs für Bilder?

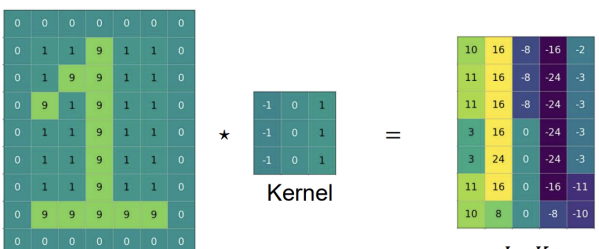
- Bilder besitzen spezielle Eigenschaften: **Lokalität**, **Invarianz** und **Hierarchie**.
- Lokale Bildbereiche enthalten relevante Merkmale (z.B. Kanten, Muster).
- Objekte im Bild bleiben auch bei Verschiebung erkennbar (Translation-Invarianz).
- Komplexe Strukturen entstehen aus einfachen lokalen Mustern.

Wie werden Bilder gespeichert?

- Bilder bestehen aus Pixelwerten, oft mit mehreren Farbkanälen (RGB).
- Ein Bild ist als mehrdimensionales Array im Computer gespeichert.
- Beispiel: Graustufenbild (1 Kanal), Farbbild (3 Kanäle).
- Bilddaten können groß sein – effiziente Verarbeitung ist wichtig!

Faltung (Convolution): Grundidee

- **Faltungsschichten** wenden kleine Filter (Kerne) lokal auf das Bild an.
- Jeder Filter erkennt ein bestimmtes Muster, z.B. eine Kante oder Textur.
- Das Resultat ist eine *Feature Map*, die zeigt, wo das Muster gefunden wurde.
- Mehrere Filter führen zu mehreren Feature Maps (Kanäle).



Mehrkanalige Faltung und Padding

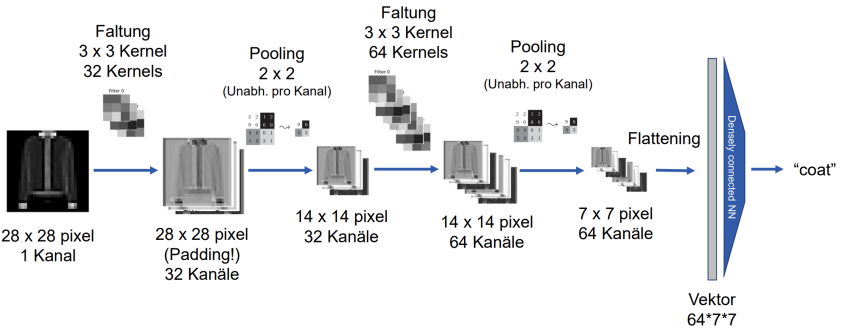
- Bei Bildern mit mehreren Kanälen wird für jeden Kanal ein separater Filter genutzt.
- Die Anzahl der Parameter bleibt niedrig, unabhängig von der Bildgröße.
- **Padding** wird oft verwendet, damit die Feature Map dieselbe Größe wie das Eingabebild behält (meist mit Nullen).

Pooling (Downsampling)

- Pooling reduziert die räumliche Größe der Feature Maps (z.B. Max-Pooling).
- Sorgt für kompaktere Darstellung, weniger Rechenaufwand und bessere Generalisierung.
- Max-Pooling wählt den größten Wert aus einem kleinen Fenster (z.B. 2x2).

Architektur eines CNNs: Beispiel

- Ein typisches CNN besteht aus mehreren Faltungs- und Pooling-Schichten, gefolgt von voll verbundenen Schichten.
- Beispiel (FashionMNIST):
  - 2 Faltungsschichten (z.B. 32 und 64 Filter)
  - 2 Max-Pooling-Schichten
  - Flattening, Dense-Layer (128 Neuronen), Ausgabe-Layer (z.B. Softmax)



Zusammenfassung: CNNs

- CNNs sind für Bilder besonders geeignet, weil sie Lokalität, Hierarchie und Invarianz ausnutzen.
- Faltung extrahiert lokale Merkmale; Pooling verdichtet Informationen.
- Parametereffizient durch lokale Filter.
- Geeignet für Aufgaben wie Bildklassifikation, Objekterkennung und mehr.

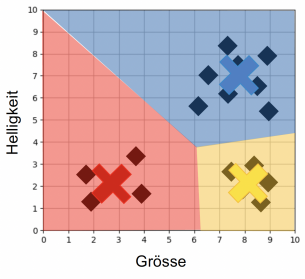
2. Recap: K-Means Clustering

Was ist K-Means?

- Ziel: Teile n Datenpunkte $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$ in K Cluster ein.
- Jedem Cluster k ist ein Zentrum $\boldsymbol{\mu}_k$ zugeordnet.
- Jeder Punkt wird dem nächstgelegenen Zentrum zugewiesen:

$$c_i = \arg \min_{k=1,...,K} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

- K ist ein Hyperparameter.

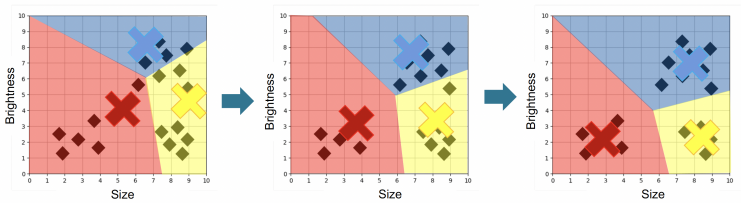


K-Means Training

1. Initialisiere Clusterzentren $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K$ zufällig.
2. **Zuweisungsschritt:** Weise jedem Punkt das nächste Zentrum zu.
3. **Update-Schritt:** Berechne für jedes Cluster das neue Zentrum als Mittelwert:

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{i:c_i=k} \mathbf{x}_i$$

4. Wiederhole Schritt 2 und 3 bis zur Konvergenz.



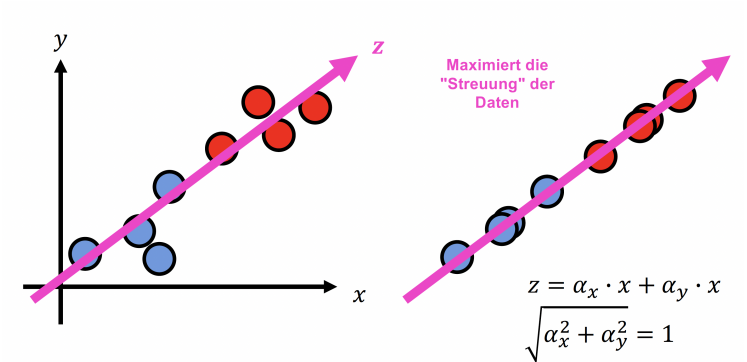
Eigenschaften und Herausforderungen von K-Means

- K-Means bestimmt nicht automatisch die beste Anzahl K.
- Sensitiv gegenüber der Wahl der Anfangszentren.
- Kann zu Überanpassung führen, wenn K zu groß gewählt wird.
- Clusterzentren müssen keine echten Datenpunkte sein.

3. Recap: Dimensionsreduktion

Was ist Dimensionsreduktion?

- Ziel: D-dimensionale Daten $\mathbf{x}_i \in \mathbb{R}^D$ auf $d < D$ Dimensionen projizieren.
- Formell: Finde Projektion $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$, $\mathbf{W} \in \mathbb{R}^{D \times d}$.
- **Vorteile:** Kompakt, schnelleres Training, bessere Visualisierung, weniger Überanpassung.



Lineare Dimensionsreduktion

Lineare Methoden:

- **Prinzip:** Finden eine lineare Projektion des Datensatzes in einen Raum niedrigerer Dimension.
- **Beispiele:** Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA).
- **Eigenschaften:**
  - Nur lineare Beziehungen zwischen Variablen werden abgebildet.
  - Schnell und effizient, aber limitiert, wenn die Daten auf gekrümmten Mannigfaltigkeiten liegen.

Nichtlineare Dimensionsreduktion

Nichtlineare Methoden:

- **Prinzip:** Abbildung der Daten auf eine niedrigdimensionale Mannigfaltigkeit, die auch nichtlineare Beziehungen erfasst.
- **Beispiele:** t-SNE, UMAP, Isomap, Locally Linear Embedding (LLE).
- **Eigenschaften:**
  - Erkennen komplexe, gekrümmte Strukturen im Datensatz.
  - Oft rechenintensiver und schwieriger zu interpretieren.

Wann welche Methode verwenden?

Lineare Methoden:

- Gut geeignet, wenn die wichtigsten Strukturen in den Daten linear sind.
- Schnelle Vorverarbeitung für Visualisierung, Kompression oder maschinelles Lernen.

Nichtlineare Methoden:

- Besser, wenn komplexe Strukturen oder Cluster erwartet werden, die durch lineare Methoden nicht erfasst werden.
- Häufig in der explorativen Datenanalyse und Visualisierung (z.B. t-SNE für High-Dimensional Data).

Praxis:

- Oft werden zuerst lineare Methoden wie PCA ausprobiert; bei Bedarf folgen nichtlineare Methoden.
- Interpretierbarkeit und Rechenaufwand sollten mitbedacht werden.

Principal Component Analysis (PCA): Mathematik

- Finde Richtung $\mathbf{u}_1$ mit maximaler Varianz:

$$\mathbf{u}_1 = \arg\max_{\|\mathbf{u}\|=1} \mathrm{Var}(\mathbf{u}^T\mathbf{x})$$

- Maximiert wird $\mathbf{u}^T S\mathbf{u}$, wobei S die Kovarianzmatrix ist:

$$S = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$$

- $\mathbf{u}_1$ ist der Eigenvektor von S mit dem größten Eigenwert.

PCA: Projektion und Rekonstruktion

- Projektion eines Punktes $\mathbf{x}_i$ auf die erste Hauptkomponente:

$$z_i = \mathbf{u}_1^T(\mathbf{x}_i - \overline{\mathbf{x}})$$

- Für mehrere Komponenten: $\mathbf{z}_i = \mathbf{U}^T(\mathbf{x}_i - \overline{\mathbf{x}})$, wobei $\mathbf{U}$ die Matrix der ersten d Eigenvektoren ist.
- Die projizierten Daten können für Visualisierung und weiteres Lernen genutzt werden.

Eigenschaften und Grenzen von PCA

- PCA kann Dimensionen ohne signifikanten Informationsverlust reduzieren.
- Die wichtigsten Hauptkomponenten sind zueinander orthogonal.
- Funktioniert am besten, wenn die Merkmale linear korreliert sind.
- Nachteile: Nur lineare Zusammenhänge werden abgebildet, Skalierung der Daten kann wichtig sein.

4. Übung

Übung: CNN-Architektur und Parameteranzahl

Betrachten Sie folgendes Convolutional Neural Network (CNN) für die Klassifikation von FashionMNIST-Bildern (Architektur wie in der Vorlesung):

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 28, 28]	320
MaxPool2d-2	[-1, 32, 14, 14]	0
Conv2d-3	[-1, 64, 14, 14]	18,496
MaxPool2d-4	[-1, 64, 7, 7]	0
Linear-5	[-1, 128]	401,536
Linear-6	[-1, 10]	1,290
Total params:		421,642

Vergleichen Sie dazu ein neuronales Netz, das nur aus einer Flattening-Schicht und dann zwei voll verbundenen Schichten mit 14×14 und 7×7 Neuronen besteht. **Berechnen Sie die Gesamtanzahl der Parameter dieses Netzes (ohne Convolution), und vergleichen Sie diesen Wert mit dem oben für das CNN erhaltenen.**

Lösung: CNN-Architektur und Parameteranzahl

Voll verbundenes Netz:

- **Flatten:** $28 \times 28 = 784$ Inputs
- **Erste Schicht:** $784 \to 196$ (14×14)
  Parameter: $784 \times 196 + 196 = 153,900$
- **Zweite Schicht:** $196 \to 49$ (7×7)
  Parameter: $196 \times 49 + 49 = 9,653$
- **Ausgabe-Schicht:** $49 \to 10$
  Parameter: $49 \times 10 + 10 = 500$

Gesamt: $153,900 + 9,653 + 500 = 164,053$ Parameter

Vergleich: Das CNN oben hat $421,642$ Parameter, das vollverbundene Netz (ohne Convolution) nur $164,053$. Das CNN ist deutlich ausdrucksstärker, bleibt aber parameter-effizient bei größeren Bildern.

Übung: CNN-Kanäle und Padding

Betrachten Sie ein RGB-Bild mit Breite $W = 320$, Höhe $H = 240$ und $C = 3$ Kanälen (RGB). Das Bild ist der Input für eine Convolutional-Schicht mit 5×5-Kern (inkl. Bias), Schrittweite 1.

1. Das Output-Bild hat Breite $W = 318$ und Höhe $H = 238$. **Wie groß ist das Padding der Schicht?**
2. Die folgende Frage betrifft die Anzahl der Ausgabekanäle: Angenommen, diese Schicht hat insgesamt 304 lernbare Parameter. **Wie viele Ausgabekanäle hat diese Schicht?**

Lösung: CNN-Kanäle und Padding

1. **Padding:**
   Output width = W − Kernel $+ 1 = 320 - 5 + 1 = 316$ (if no padding).
   Aber Output = 318, also wurde **Padding = 1** Pixel an jeder Seite verwendet.
2. **Ausgabekanäle:**
   Für jeden Ausgabekanal: $5 \times 5 \times 3$ (RGB) + 1 Bias = 76 Parameter
   $304/76 = 4$
   $\rightarrow$ **Es gibt 4 Ausgabekanäle.**

Coding-Aufgabe: PCA auf MNIST

Aufgabe:

- Wenden Sie Principal Component Analysis (PCA) auf die MNIST-Bilddaten an (X_train).
- Bestimmen Sie den minimalen Wert von k, sodass mindestens 90% der Gesamtvarianz erklärt werden.
- Stellen Sie den kumulierten Anteil der erklärten Varianz als Plot dar.

Hinweis: Nutzen Sie `sklearn.decomposition.PCA` und denken Sie daran, die Daten vorab zu normalisieren (z.B. mit `StandardScaler`).

Lösungsansatz: PCA auf MNIST

1. Daten normalisieren (z.B. mit `StandardScaler`).
2. PCA-Modell auf `X_train` fitten (ohne Begrenzung der Komponentenanzahl).
3. Die kumulierte Summe der erklärten Varianz (`explained_variance_ratio_`) berechnen.
4. Den kleinsten Wert k bestimmen, für den die kumulierte Varianz > 0.9 ist.
5. Plot mit k gegen kumulierte erklärte Varianz.

Coding-Aufgabe: PCA in 2D und Visualisierung

Aufgabe:

- Reduzieren Sie die MNIST-Daten mit PCA auf 2 Dimensionen.
- Stellen Sie die transformierten Daten als Streudiagramm dar, wobei jede Ziffer eine eigene Farbe erhält.
- Diskutieren Sie, ob sich die Klassen im 2D-Raum gut trennen lassen.

Hinweis: Nutzen Sie `plt.scatter` und färben Sie die Punkte nach den Labeln (`y_train`).

Lösungsansatz: PCA in 2D und Visualisierung

1. Setzen Sie `n_components=2` in PCA.
2. Transformieren Sie `X_train` zu 2D-Koordinaten.
3. Erstellen Sie einen Streudiagramm-Plot, wobei jede Ziffer/Label eine andere Farbe hat.
4. Diskutieren: Die Ziffern sind meist nicht linear separierbar, aber einige Cluster sind sichtbar.

Coding-Aufgabe: t-SNE Embedding

Aufgabe:

- Wenden Sie t-SNE auf eine Stichprobe der MNIST-Daten (z.B. 2000 zufällige Punkte) an und projizieren Sie diese in 2 Dimensionen.
- Visualisieren Sie die eingebetteten Punkte als Streudiagramm, gefärbt nach Ziffer.
- Vergleichen Sie die Trennschärfe mit dem PCA-Plot.

Hinweis: t-SNE ist rechenintensiv—wählen Sie daher nur einen Teil der Daten aus!

Lösungsansatz: t-SNE Embedding

1. Ziehen Sie eine Zufallsstichprobe von z.B. 2000 Punkten aus `X_train`.
2. Normalisieren Sie die Daten (optional, aber empfohlen).
3. Führen Sie t-SNE (`sklearn.manifold.TSNE`) mit `n_components=2` aus.
4. Erstellen Sie einen Streudiagramm-Plot, jede Ziffer anders eingefärbt.
5. Vergleich: t-SNE trennt die Ziffern oft besser als PCA, aber die Achsen sind weniger interpretierbar.

5. Hausaufgaben

Exercise 12: Intro ML III

Auf https://expert.ethz.ch/enrolled/SS25/mavt2/exercises

- K-means
- Grid Search for Polynomials
- Circles
- Dimensionality Reduction
- CNNs

Abgabedatum: Freitag 06.06.2025, 20:00 MEZ