Grundlagen der Künstlichen Intelligenz

Wintersemester 25/26

Vorlesung 8

Expectation-Maximization,
Gaussian Mixture Models und
Variational Autoencoder

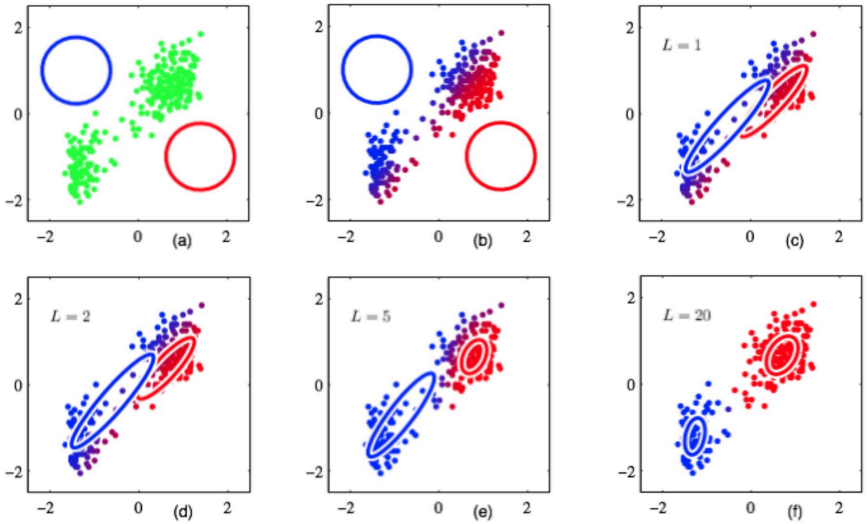
Prof. Dr. Pascal Friederich
T.T.-Prof. Dr. Peer Nowack

Lernergebnisse

Was werden wir heute lernen?

- Gaussian Mixture Models und Modelle mit latenten Variablen
- Expectation-Maximization Algorithmus (EM) für das Training von Mixture Models
- Herleitung von EM als Lösungsalgorithmus für allgemeine Modelle mit latenten Variablen
- Analyse des EM-Algorithmus
- Kurzer Einblick in Variational Autoencoder und deren Beziehung zu EM

Teil 1



1) Expectation Maximization und Gaussian Mixture Models

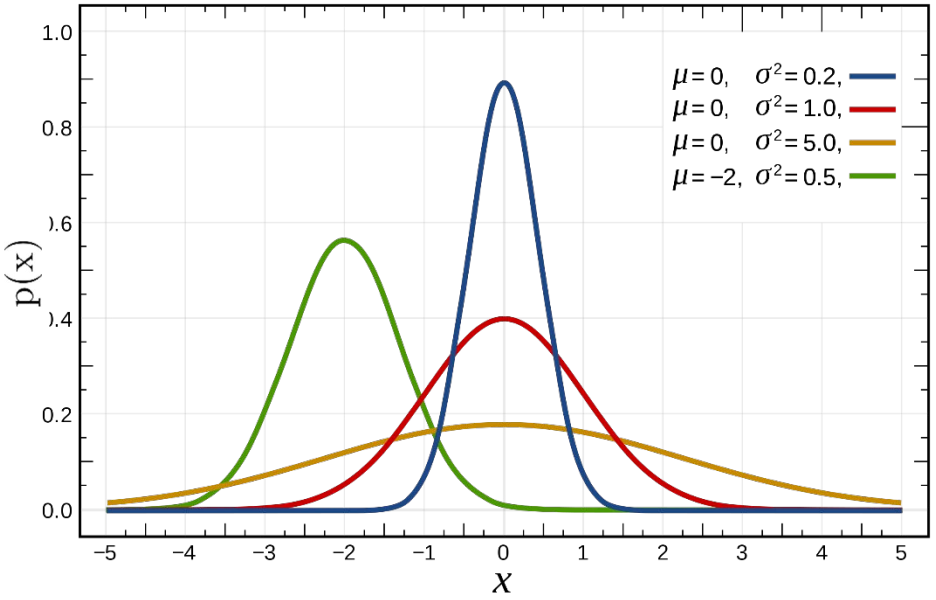
2) Verallgemeinerte Sicht auf EM

3) Variational Autoencoder

Gauss Verteilung

- Kontinuierliche Zufallsvariable: $X \in \mathbb{R}$
- Distribution is completely specified by mean μ and variance σ^2

$$p(x) = \mathcal{N}(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

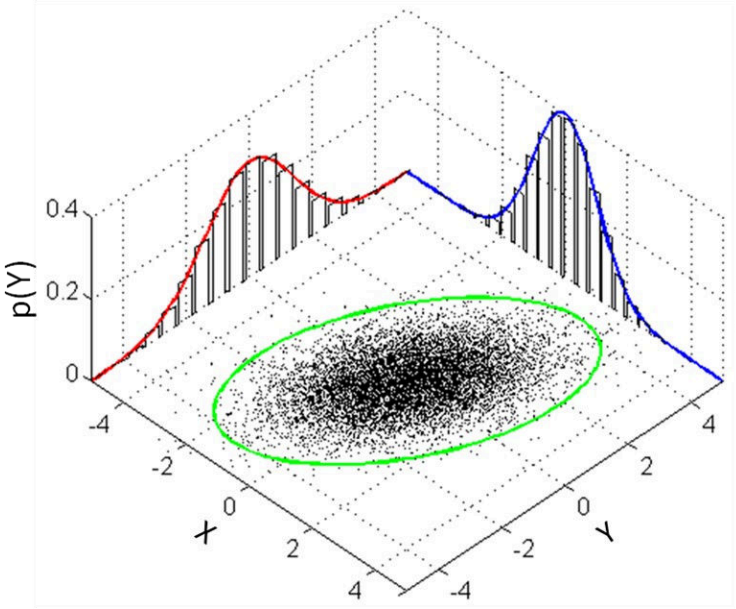


Multivariate Gauss Verteilung

- Kontinuierliche Zufallsvariable :

$$X \in \mathbb{R}^d$$

- Verteilung ist spezifiziert durch Mittelwertvektor $\boldsymbol{\mu}$ und Kovarianz-Matrix $\boldsymbol{\Sigma}$



$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right\}$$

Mixture Modelle

Eine Mixture-Verteilung ist die Summe der Einzelverteilungen:

Anzahl der Komponenten

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(k)p(\boldsymbol{x}|k)$$

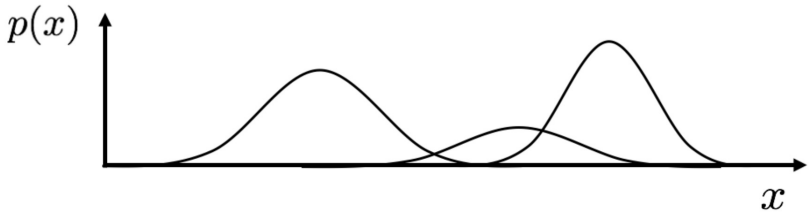
Mixture Koeffizienten

k-te Mixture Komponente

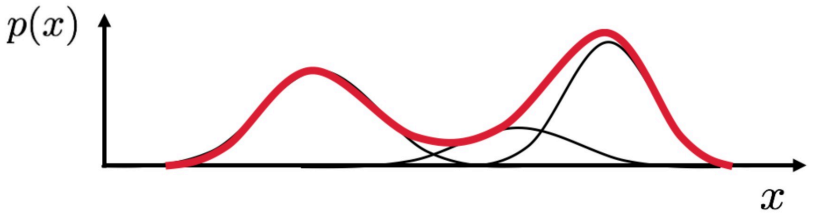
Im Grenzfall mit vielen/unendlichen Komponenten kann dies eine beliebige glatte Dichte approximieren

Gaussian Mixture Models (GMM)

- Einzelne Gauß Verteilungen



- Summe der Gaußschen Verteilungen



Gaußsche Mixture Modelle (GMMs)

- **Mixture Koeffizienten:**

$$p(k) = \pi_k, \text{ with } 0 \leq \pi_k \leq 1, \sum_k \pi_k = 1$$

- **Mixture Komponente:**

$$p(\boldsymbol{x}|k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- **Mixture Verteilung:**

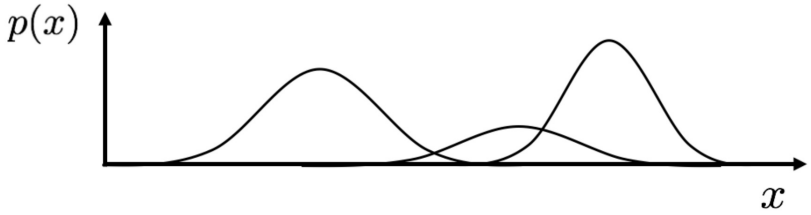
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  - Integriert immer zu 1
  - Parameter der Mixture

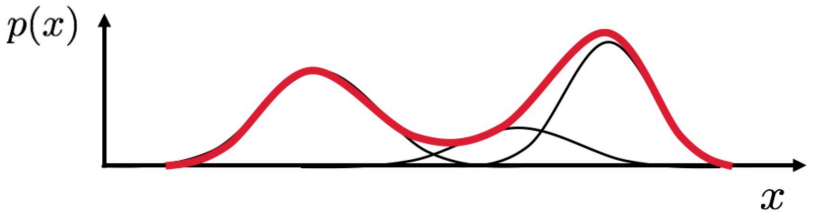
$$\boldsymbol{\theta} = \{\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \ldots, \pi_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}$$

Gaussian Mixture Models (GMM)

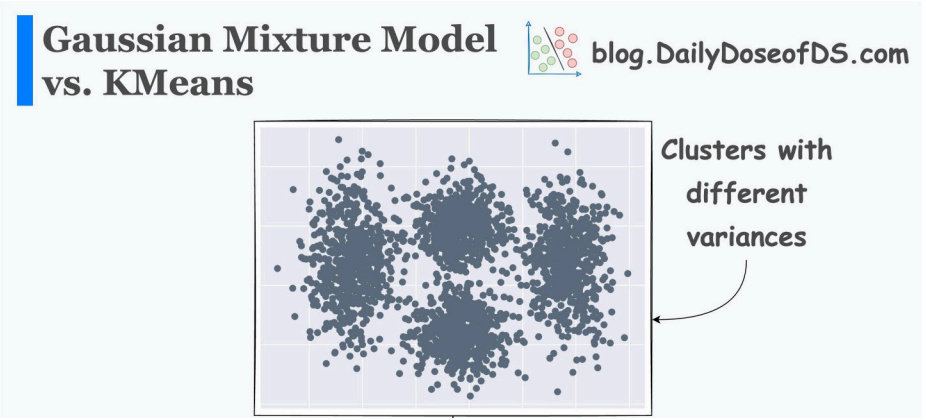
- Einzelne Gauß Verteilungen



- Summe der Gaußschen Verteilungen

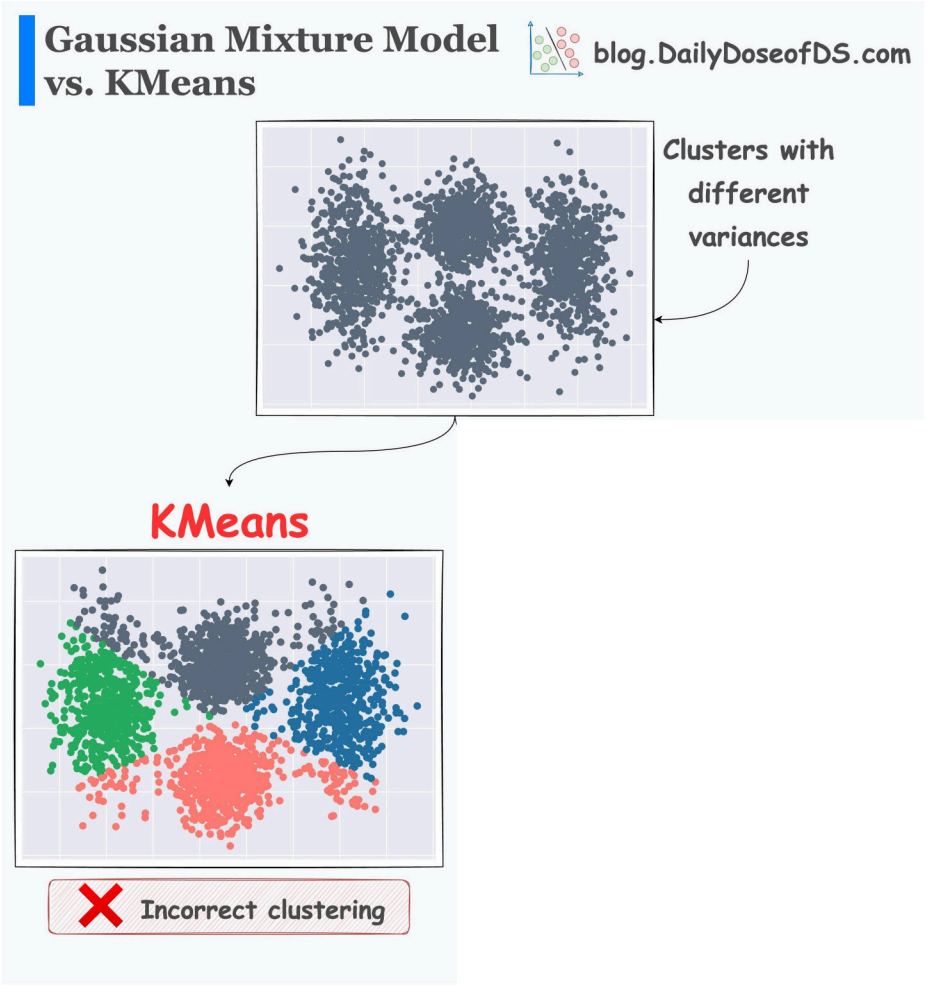


GMMs als Clustering Methode



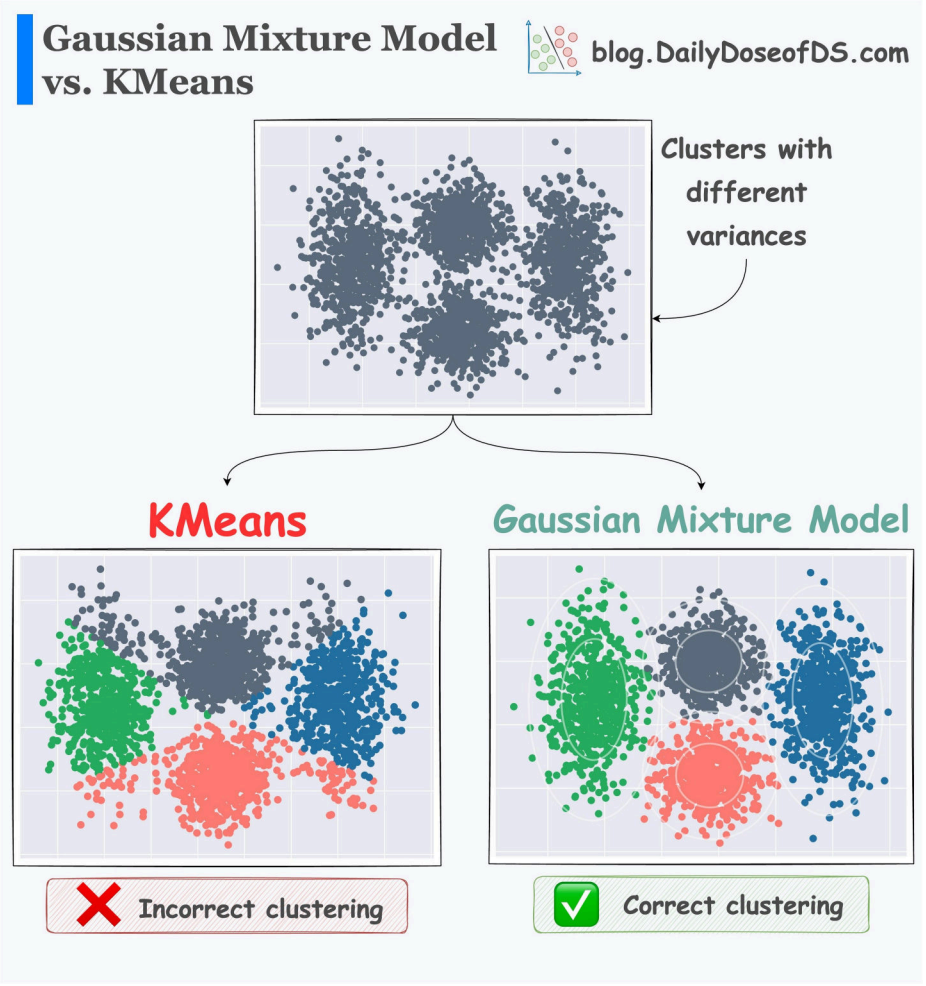
GMMs als Clustering Methode

- Die Kovarianz Matrix lernt die (elliptische) „Form“ des Clusters
- K-Means: „Form“ ist fixiert durch kürzesten Abstand



GMMs als Clustering Methode

- Die Kovarianz Matrix lernt die (elliptische) „Form“ des Clusters
- K-Means: „Form“ ist fixiert durch kürzesten Abstand



Maximum Likelihood einer Mixture Verteilung

- (Marginal-)Log-Likelihood mit *N* iid. Punkten

$$\mathcal{L} = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \underbrace{p(\boldsymbol{x}_i|\boldsymbol{\theta})}_{\text{marginal}} = \sum_{i=1}^{N} \log \underbrace{\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)}_{\text{non-exponential family}}$$

- F: Können wir einen Gradientenaufstieg durchführen?

(iid: Independent and identically distributed)

Maximum Likelihood einer Mixture Verteilung

- (Marginal-)Log-Likelihood mit *N* iid. Punkten

$$\mathcal{L} = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \underbrace{p(\boldsymbol{x}_i|\boldsymbol{\theta})}_{\text{marginal}} = \sum_{i=1}^{N} \log \underbrace{\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)}_{\text{non-exponential family}}$$

- F: Können wir einen Gradientenaufstieg durchführen?

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_j} = \sum_{i=1}^{N} \frac{\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)$$

$$= \sum_{i=1}^{N} \frac{p(j)p(\boldsymbol{x}_i|j)}{p(\boldsymbol{x}_i)} \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)$$

$$= \sum_{i=1}^{N} \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) p(j|\boldsymbol{x}_i)$$

(iid: Independent and identically distributed)

Maximum Likelihood einer Mixture Verteilung

- (Marginal-)Log-Likelihood mit *N* iid. Punkten

$$\mathcal{L} = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \underbrace{p(\boldsymbol{x}_i|\boldsymbol{\theta})}_{\text{marginal}} = \sum_{i=1}^{N} \log \underbrace{\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)}_{\text{non-exponential family}}$$

- F: Können wir einen Gradientenaufstieg durchführen?

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_j} = \sum_{i=1}^{N} \frac{\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)$$

$$= \sum_{i=1}^{N} \frac{p(j)p(\boldsymbol{x}_i|j)}{p(\boldsymbol{x}_i)} \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)$$

$$= \sum_{i=1}^{N} \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) p(j|\boldsymbol{x}_i)$$

- × Der Gradient hängt von allen anderen Komponenten ab (zyklische Abhängigkeit)
- × Keine geschlossene Lösung
- × Typischerweise sehr langsame Konvergenz
- ► A: Ja, aber die Summe (Marginalisierung) verträgt sich nicht gut mit dem Log

(iid: Independent and identically distributed)

Maximum Likelihood einer Mixture Verteilung

Gradient Descent ist möglich... aber sehr langsam!

- Können wir ein spezielleres Optimierungsverfahren für Mixture Modelle finden?
- Ja! **Expectation Maximization!**

Schätzung von Gaußschen Mixture Modellen

Warum also ist die Optimierung von $\mathcal{L} = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ **so schwierig?**

- Wir wissen nicht, welche Mixture-komponente k welchen Datenpunkt erzeugt hat
- Wenn wir Daten aus der gemeinsamen Verteilung $p(\boldsymbol{x}_i, k_i|\boldsymbol{\theta})$ hätten, dann wäre es einfach ...

In diesem Fall können wir einfach eine Maximum-Likelihood-Schätzung durchführen:

- Zuweisungen: $q_{ik} = p\left(k_i = k \mid \boldsymbol{x}_i\right)$ binär: $q_{ik} = \mathbb{I}(k, k_i)$ hat Wert 1, wenn die i-te Probe zur k-ten Komponente gehört, und 0, wenn nicht
- Koeffizienten: $\pi_k = \frac{\sum_i q_{ik}}{N}$
- Mittelwerte: $\boldsymbol{\mu}_k = \frac{\sum_i q_{ik} \boldsymbol{x}_i}{\sum_i q_{ik}}$
- Kovarianzen: $\boldsymbol{\Sigma}_k = \frac{\sum_i q_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i q_{ik}}$

Expectation-Maximization: Henne Ei...

Wir wissen jedoch nicht, welche Komponente zu welcher Probe gehört.

- Können wir das abschätzen? Bei einem gegebenen Mixture-Modell, ja!

Expectation-Maximization: Henne Ei...

Wir wissen jedoch nicht, welche Komponente zu welcher Probe gehört.

- Können wir das abschätzen? Bei einem gegebenen Mixture-Modell, ja!

Expectation-Step:

- Berechnung der Clusterwahrscheinlichkeiten aka Responsibilities für jedes Sample (Bayes-Regel)

$$q_{ik} = p(k_i = k|\boldsymbol{x}_i) = \frac{p(\boldsymbol{x}_i|k)p(k)}{\sum_j p(\boldsymbol{x}_i|j)p(j)} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- Die Responsibilities q_{ik} liegen zwischen 0 und 1
- Aber wir müssen die Gaußschen Komponenten kennen

Expectation-Maximization: Henne Ei...

Wir wissen jedoch nicht, welche Komponente zu welcher Probe gehört.

- Können wir das abschätzen? Bei einem aktuellen Mixture-Modell, ja!

Maximization-Step:

- Berechnung der (gewichteten) Maximum-Likelihood-Schätzung

$$\pi_k = \frac{\sum_i q_{ik}}{N} \qquad \boldsymbol{\mu}_k = \frac{\sum_i q_{ik}\boldsymbol{x}_i}{\sum_i q_{ik}} \qquad \boldsymbol{\Sigma}_k = \frac{\sum_i q_{ik}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i q_{ik}}$$

- Aber wir müssen die Responsibilities kennen

Algorithmus: EM für GMMs

- **Initialisieren:** Mixture-Komponenten + Mixture-Koeffizienten
  - z.B. Verwendung von k-means für die Komponentenmittelwerte und eine anfängliche Kovarianz

- **Wiederhole den Vorgang** bis zur Konvergenz**:**
  - **Expectation-Step:** Berechnung der Responsibilities

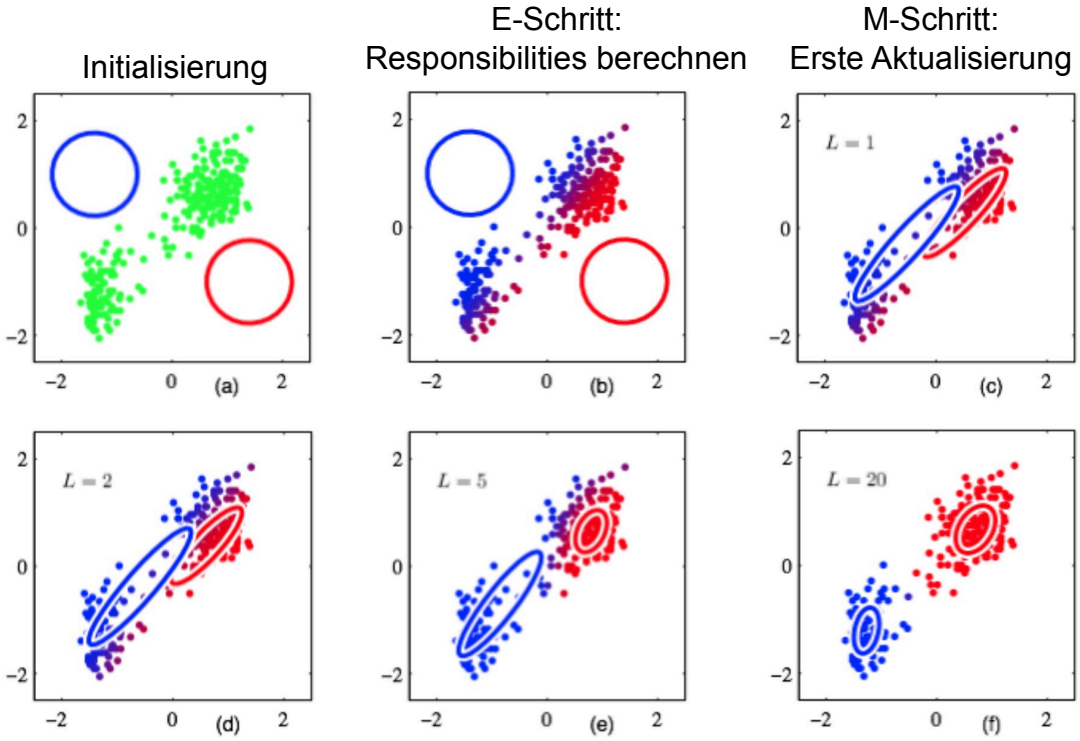
$$q_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

  - **Maximization-Step:** Aktualisierung der Koeffizienten, Komponentenmittelwerte und Komponentenvarianz

$$\pi_k = \frac{\sum_i q_{ik}}{N} \qquad \boldsymbol{\mu}_k = \frac{\sum_i q_{ik} \boldsymbol{x}_i}{\sum_i q_{ik}} \qquad \boldsymbol{\Sigma}_k = \frac{\sum_i q_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i q_{ik}}$$

Abbildung

- Jede Komponente steht für ein Cluster im Datensatz
- EM ist sehr empfindlich gegenüber der Initialisierung



Praktische Überlegungen...

Wie viele Komponenten brauchen wir?

- Mehr Komponenten führen in der Regel zu einem bessereren Trainings-Likelihood
- Aber sind mehr Komponenten unbedingt besser? Nein, wegen der Overfitting!
- Es handelt sich wieder um ein Model-Selection Problem (verwende z.B. Cross-Validation)

EM versus k-means

K-means kann als Spezialfall von EM angesehen werden:

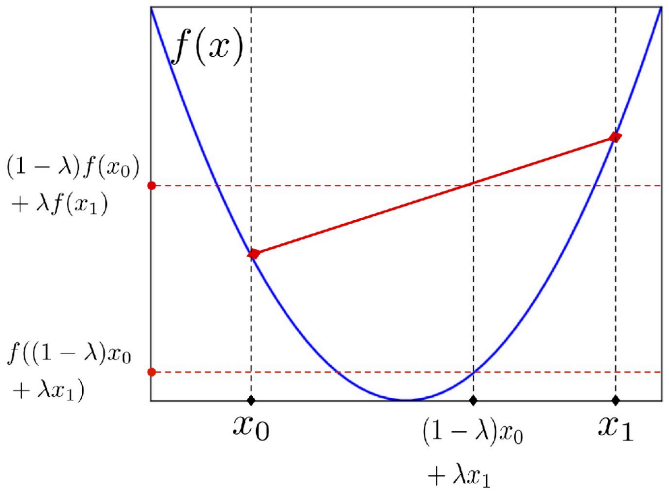
- Co-Varianzen werden immer auf 0 gesetzt (im Grenzfall)
- **E-Step / Zuweisungsschritt**:
  - Responsibilities q_{ik} des nächstgelegenen Clusters k werden auf 1 gesetzt, alle anderen Werte sind 0

- **M-Step / Anpassungsschritt:**
  - Die Aktualisierung für den Mittelwert ist die gleiche
  - Co-Varianzen werden ignoriert (auf nahe 0 gesetzt)

- EM ist schwieriger zu erlernen als k-means, aber es liefert auch Varianzen und Dichten
- Häufig wird k-means verwendet, um die Mittelwerte für EM zu initialisieren.

K-means ist dafür bekannt, dass es konvergiert. Konvergiert auch EM immer?

Teil 2



1) Expectation Maximization und Gaussian Mixture Models

2) Verallgemeinerte Sicht auf EM

3) Variational Autoencoder

Modelle für latente Variablen

Mixture-Modelle sind ein Beispiel für Modelle mit latenten Variablen

- Beobachtete Variablen: ***x***, latente Variablen: ***z*** (z. B. der Index der Mixture-Komponente)
- Parametrisches Modell: $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$
- Marginale Verteilung: $\underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{z} p(\boldsymbol{x}, z),}_{\text{discrete latent variable}} \quad \underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}}_{\text{continuous latent variable}}$

(Marginal) Log-Likelihood:

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left(\sum_{z_i} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i) \right)$$

... ist für alle Modelle mit latenten Variablen schwer zu optimieren (wegen des Logarithmus einer Summe)

Expectation-Maximization (EM)

Expectation-Maximization (EM) ist ein allgemeiner Algorithmus zur Schätzung von Modellen mit latenten Variablen

- **Häufigste Anwendung:** Gaußsche Mixture Modelle
- ... aber auch viele andere (tiefe) Modelle
- Seine Erweiterung wird Variational Bayes genannt, die dem Variational Autoencoder zugrunde liegt

EM kann auf 2 Arten abgeleitet werden:

- Jensen‘s Ungleichung
- Zerlegung in Untergrenze und Kullback-Leibler-Divergenz-Begriff
  (besserer, aber komplizierterer Weg... siehe ML-Vorlesung)

Jensens Ungleichung: Erinnerung

- Wenn eine Funktion konvex ist, dann

$$f\left(\sum_i \lambda_i \boldsymbol{x}_i\right) \leq \sum_i \lambda_i f(\boldsymbol{x}_i),$$

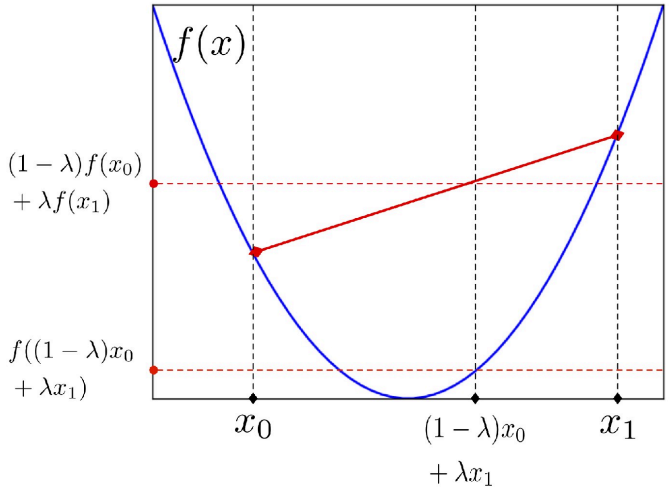
wobei $\{\lambda_i\}$ so sind, dass jedes $\lambda_i \geq 0$ und

$$\sum_i \lambda_i = 1$$

- Wenn wir die λ_i als die Parameter einer kategorialen Verteilung behandeln, d. h. $\lambda_i = p(\boldsymbol{x} = \boldsymbol{x}_i)$, kann dies wie folgt geschrieben werden

$$f\left(\mathbb{E}_p[\boldsymbol{x}]\right) \leq \mathbb{E}_p[f(\boldsymbol{x})].$$

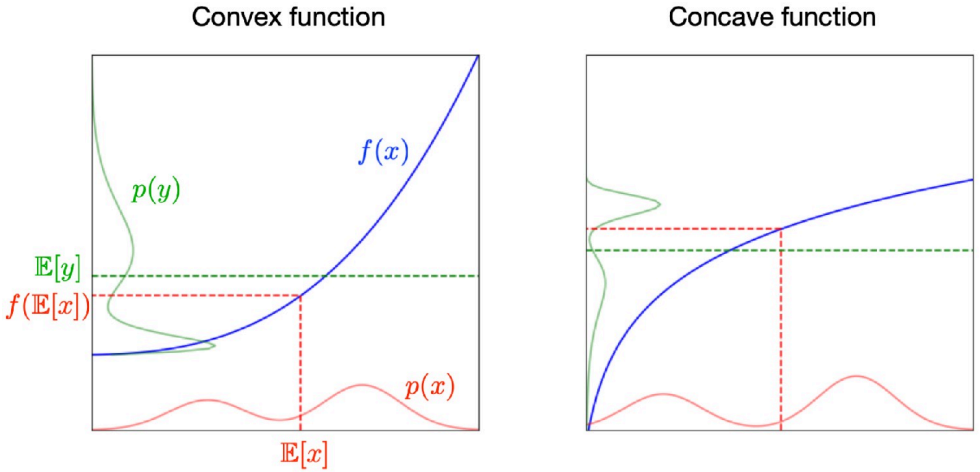
- Dies ist als Jensensche Ungleichung bekannt. Sie gilt auch für kontinuierliche Verteilungen.



Jensensche Inequality

- Eine Funktion *f(**x**)* ist konkav, wenn - *f(**x**)* konvex ist. In diesem Fall drehen wir die Jensensche Inequality um:

$$f\left(\mathbb{E}_p[\boldsymbol{x}]\right) \geq \mathbb{E}_p[f(\boldsymbol{x})].$$



Zurück zu Expectation-Maximization

Konkave Funktion!

(Marginale) Log-Likelihood:

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left(\sum_{z_i} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i) \right)$$

... die für alle Modelle mit latenten Variablen schwer zu optimieren ist (wegen des Logarithmus einer Summe)

Zurück zu Expectation-Maximization

Konkave Funktion!

(Marginale) Log-Likelihood:

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left(\sum_{z_i} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i) \right)$$

... die für alle Modelle mit latenten Variablen schwer zu optimieren ist (wegen des Logarithmus einer Summe)

- Trick: Führe neue Verteilung $q_i(z_i)$ ein (wir werden gleich sehen, was das ist):

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{z_i} q_i(z_i) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)}{q_i(z_i)} \right) = \sum_{i=1}^{N} \log \mathbb{E}_{q_i(z_i)} \left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)}{q_i(z_i)} \right]$$

Zurück zu Expectation-Maximization

Konkave Funktion!

(Marginale) Log-Likelihood:

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left(\sum_{z_i} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i) \right)$$

... die für alle Modelle mit latenten Variablen schwer zu optimieren ist (wegen des Logarithmus einer Summe)

- Trick: Führe neue Verteilung $q_i(z_i)$ ein (wir werden gleich sehen, was das ist):

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{z_i} q_i(z_i) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)}{q_i(z_i)} \right) = \sum_{i=1}^{N} \log \mathbb{E}_{q_i(z_i)} \left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)}{q_i(z_i)} \right]$$

- Log ist eine konkave Funktion ist. Daher können wir die Jensensche Ungleichung verwenden, um den Logarithmus in den Erwartungswert zu schieben und so die untere Schranke zu erhalten:

$$\log \left(\mathbb{E}_p[\boldsymbol{x}] \right) \geq \mathbb{E}_p[\log(\boldsymbol{x})]$$

$$\mathrm{L}(\boldsymbol{\theta}) \geq \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)}{q_i(z_i)} \right] := \mathcal{L}(q, \boldsymbol{\theta})$$

Untere Variationsschranke

- Soeben wurde eine untere Schranke für die LogLikelihood abgeleitet

$$\mathrm{L}(\boldsymbol{\theta}) \geq \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)}{q_i(z_i)} \right] := \mathcal{L}(q, \boldsymbol{\theta})$$

- Vereinfachen:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i) \right] - \underbrace{\mathbb{E}_{q_i(z_i)} \left[\log q_i(z_i) \right]}_{\text{constant w.r.t } \boldsymbol{\theta}}$$

Anmerkungen:

- Die untere Schranke ist viel einfacher zu optimieren
- Oft gibt es geschlossene Formeln für die Herleitung von *q* gegeben $\boldsymbol{\theta}$ und umgekehrt
- Die untere Grenze gilt für jede Wahl von $q_i(z_i)$

Expectation-Maximization Schritte

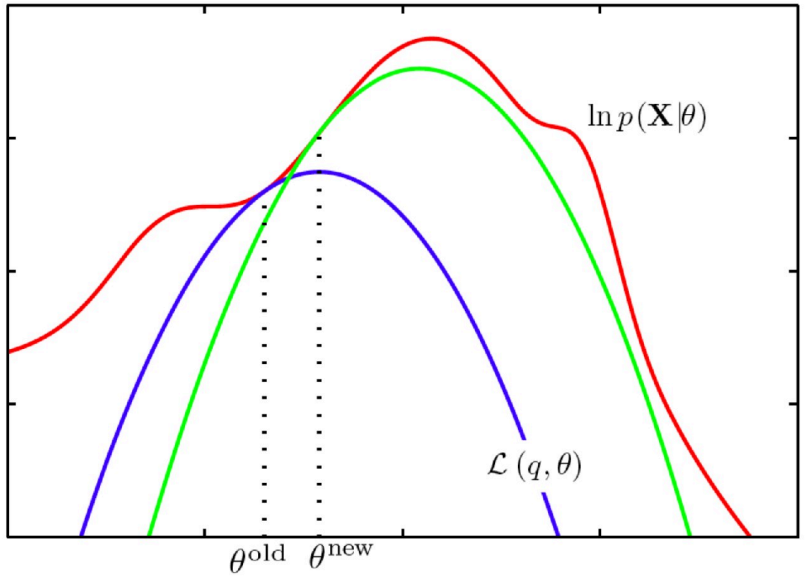
EM optimiert die untere Schranke durch die Anwendung von 2 Schritten:

- **(M)aximization-step** (finde Maximum von blau)**:**
  - Fixe Variationsverteilung $q = q_{\text{old}}$
  - Maximierung der unteren Schranke in Bezug auf $\boldsymbol{\theta}$

$$\boldsymbol{\theta}_{\text{new}} = \text{argmax}_{\boldsymbol{\theta}} \quad \mathcal{L}(q_{\text{old}}, \boldsymbol{\theta})$$

- **(E)xpectation-step** (blau → grün)**:**
  - Parameter festlegen $\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{new}}$
  - Maximierung der unteren Schranke in Bezug auf *q*

$$q_{\text{new}} = \text{argmax}_q \, \mathcal{L}\,(q, \boldsymbol{\theta}_{\text{new}}\,)$$



Expectation-Maximization Schritte

Expectation-Step:

- Finde $q(z)$ welches die untere Schranke maximiert:

$$q_{\text{new}} = \text{argmax}_q \quad \mathcal{L}(q, \boldsymbol{\theta}_{\text{old}})$$

  - Kann in geschlossener Form für diskrete latent Variablen z durchgeführt werden:
    - Z.B. Komponentenindizes für Mixture-Modelle
  - Das optimale q entspricht dem Posterior (hier nicht hergeleitet)

$$q_{\text{new}}(z_i) = p_{\boldsymbol{\theta}_{\text{old}}}(z_i|\boldsymbol{x}_i) = \frac{p_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{x}_i, z_i)}{p_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{x}_i)}$$

Expectation-Maximization Schritte

(M)aximierungs-Schritt:

- Maximierung der unteren Schranke bezüglich $\boldsymbol{\theta}$

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, z_i)\right] - \underbrace{\mathbb{E}_{q_i(z_i)} \left[\log q_i(z_i)\right]}_{\text{constant w.r.t } \boldsymbol{\theta}}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_i \sum_{z_i} q(z_i) \log p(\boldsymbol{x}_i, z_i | \boldsymbol{\theta}) + \text{const}$$

- Jeder mögliche Wert der “fehlenden Daten” wird mit der Wahrscheinlichkeit gewichtet, dass dieser Wert wahr ist, d.h.:

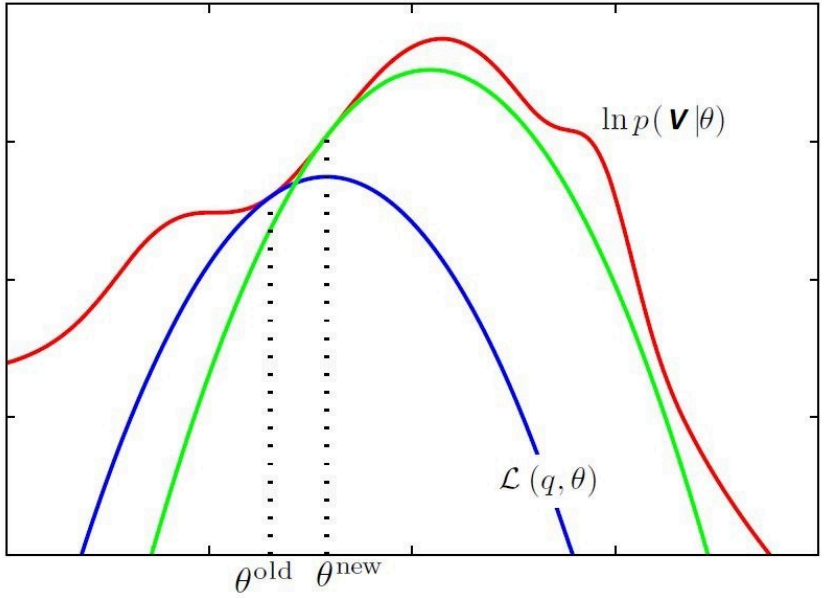
$$q(z_i) = p_{\boldsymbol{\theta}_{\text{old}}}(z_i | \boldsymbol{x}_i) = \frac{p_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{x}_i, z_i)}{p_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{x}_i)}$$

... erhalten im E-Schritt

Illustration von EM

Optimierung

- Die untere Schranke (blaue Kurve) ist eine konvexe Annäherung an den Marginal Likelihood (rote Kurve)
  - Das Maximum der unteren Schranke kann leicht ermittelt werden ($\boldsymbol{\theta}_{\text{new}}$)
  - Lösungen $\boldsymbol{\theta}_{\text{new}}$ in geschlossener Form verfügbar, kein Gradientenabstieg erforderlich
- Berechne eine neue untere Schranke (grüne Kurve)
- Aufgrund der lokalen Approximation der unteren Schranke kann EM nur lokale Optima finden



Konvergenz: EM konvergiert immer!

- EM verbessert die untere Grenze bei jedem Schritt
- Es kann gezeigt werden (siehe ML-Vorlesung), dass auch die logarithmische Grenzwahrscheinlichkeit immer erhöht wird (oder gleich bleibt)

EM für GMMs revisited

- **Mixture Koeffizienten:**

$$p(k) = \pi_k, \text{ with } 0 \leq \pi_k \leq 1, \sum_k \pi_k = 1$$

- **Mixture Komponenten:**

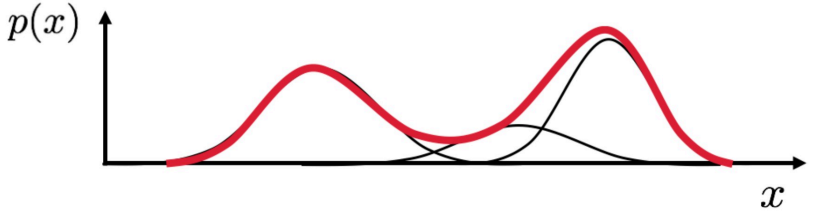
$$p(\boldsymbol{x}|k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- **Mixture Verteilung:**

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  - Integriert immer zu 1
  - Parameter des Mixtures

$$\boldsymbol{\theta} = \{\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \ldots, \pi_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}$$



EM für GMMs revisited

Folie 33 Bayes

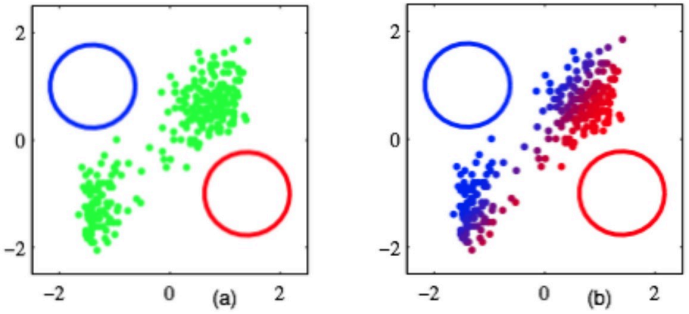
E-Step

- Berechne "Responsibilities":

$$q_{ik} = p(z_i = k|\boldsymbol{x}_i) = \frac{p(\boldsymbol{x}_i \mid z_i = k)p(z_i = k)}{p(\boldsymbol{x}_i)}$$

$$= \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}$$

- Wie viel die Komponente k zur Erzeugung von $\boldsymbol{x}_i$ beiträgt gemäß dem aktuellen Mixture-Modell



EM für GMMs revisited

M-Step:

Folie 34 → $\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \sum_i \sum_k q_{ik} \log p(k) p(\boldsymbol{x}_i \mid k)$ log Regeln → $= \arg\max_{\boldsymbol{\theta}} \sum_i \sum_k q_{ik} \log p(k) + \sum_k \sum_i q_{ik} \log p(\boldsymbol{x}_i \mid k)$

- Wir können die Aktualisierungen der einzelnen Komponenten und Koeffizienten trennen
  - nur additive Terme im Lower Bound
- **Koeffizienten aktualisieren:**

$$\overset{p(k)=\pi_k}{\boldsymbol{\pi}} = \arg\max_{\boldsymbol{\pi}} \sum_i \sum_k q_{ik} \log \pi_k$$

- **Komponenten aktualisieren:**

$$\overset{p(\boldsymbol{x}|k)=\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} = \arg\max_{\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k} \sum_i q_{ik} \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  - Jeder Datenpunkt wird mit q gewichtet$_{ik}$
  - Gewichtete Maximum-Likelihood-Schätzung

EM für Gaußsche Mixture-Modelle

Gewichtete Maximum-Likelihood Updates:

- Aktualisierungskoeffizienten: $\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \sum_i \sum_k q_{ik} \log \pi_k$

EM für Gaußsche Mixture-Modelle

Gewichtete Maximum-Likelihood Updates:

- Koeffizienten aktualisieren: $\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \sum_i \sum_k q_{ik} \log \pi_k$
  - Ergebnis: $\pi_k = \frac{\sum_i q_{ik}}{\sum_k \sum_i q_{ik}} = \frac{\sum_i q_{ik}}{N}$ (erfordert einige Schritte mit Lagrange Multiplier und Ableitung)

EM für Gaußsche Mixture-Modelle

Gewichtete Maximum-Likelihood Updates:

- Koeffizienten aktualisieren: $\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \sum_i \sum_k q_{ik} \log \pi_k$
  - Ergebnis: $\pi_k = \frac{\sum_i q_{ik}}{\sum_k \sum_i q_{ik}} = \frac{\sum_i q_{ik}}{N}$ (erfordert einige Schritte mit Lagrange Multiplier und Ableitung)
- Komponenten aktualisieren: $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k = \arg\max_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \sum_i q_{ik} \log \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

EM für Gaußsche Mixture-Modelle

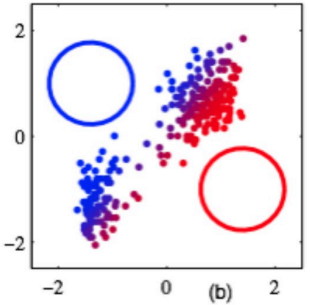
Gewichtete Maximum-Likelihood Updates:

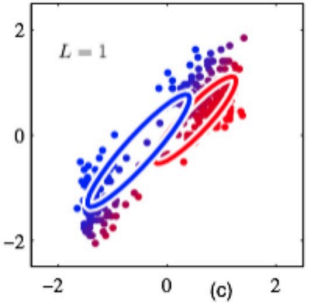
- Koeffizienten aktualisieren: $\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \sum_i \sum_k q_{ik} \log \pi_k$
  - Ergebnis: $\pi_k = \frac{\sum_i q_{ik}}{\sum_k \sum_i q_{ik}} = \frac{\sum_i q_{ik}}{N}$ (erfordert einige Schritte mit Lagrange Multiplier und Ableitung)
- Komponenten aktualisieren: $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k = \arg\max_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \sum_i q_{ik} \log \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
  - Mittelwert: $\boldsymbol{\mu}_k = \frac{\sum_i q_{ik} \boldsymbol{x}_i}{\sum_i q_{ik}}$
  - Kovarianz: $\boldsymbol{\Sigma}_k = \frac{\sum_i q_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i q_{ik}}$

EM für Gaußsche Mixture-Modelle

Gewichtete Maximum-Likelihood Updates:

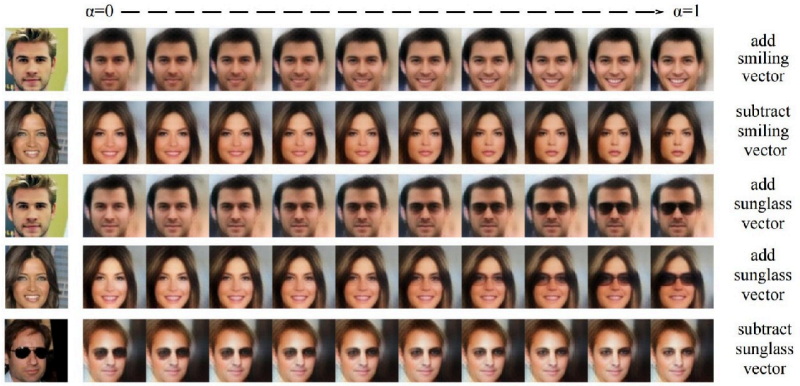
- Koeffizienten aktualisieren: $\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \sum_i \sum_k q_{ik} \log \pi_k$
  - Ergebnis: $\pi_k = \frac{\sum_i q_{ik}}{\sum_k \sum_i q_{ik}} = \frac{\sum_i q_{ik}}{N}$ (erfordert einige Schritte mit Lagrange Multiplier und Ableitung)
- Komponenten aktualisieren: $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k = \arg\max_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \sum_i q_{ik} \log \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
  - Mittelwert: $\boldsymbol{\mu}_k = \frac{\sum_i q_{ik} \boldsymbol{x}_i}{\sum_i q_{ik}}$
  - Kovarianz: $\boldsymbol{\Sigma}_k = \frac{\sum_i q_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i q_{ik}}$





Dieselben Gleichungen wie zuvor, aber jetzt haben wir sie hergeleitet!

Teil 3



1) Expectation Maximization und Gaussian Mixture Models

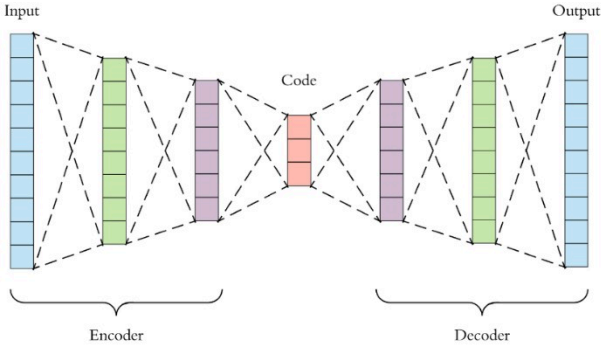
2) Verallgemeinerte Sicht auf EM

3) Variational Autoencoder

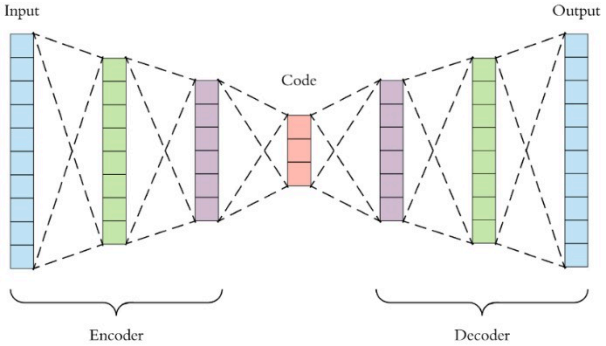
Variational-Autoencoder

Auto-Encoder

- Reduziert die Dimensionalität...
- ... aber es gibt keine Dichte
- ... und der AE kann keine neuen Daten generieren



Variational-Autoencoder

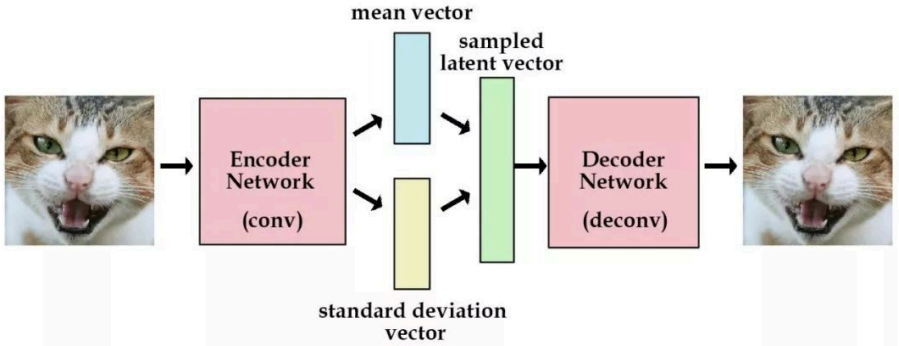


Auto-Encoder

- Reduziert die Dimensionalität...
- ... aber es gibt keine Dichte
- ... und der AE kann keine neuen Daten generieren

Variational-Autoencoder

- Definiert auch eine Verteilung im latenten Raum
- **Generatives Modell:** Kann Samples im latenten Raum erzeugen
- ... und projizieren sie in den High-D-Raum (z. B. Bilder)
- Liefert eine **Dichte** (auch wenn sie sehr schwer zu berechnen ist)



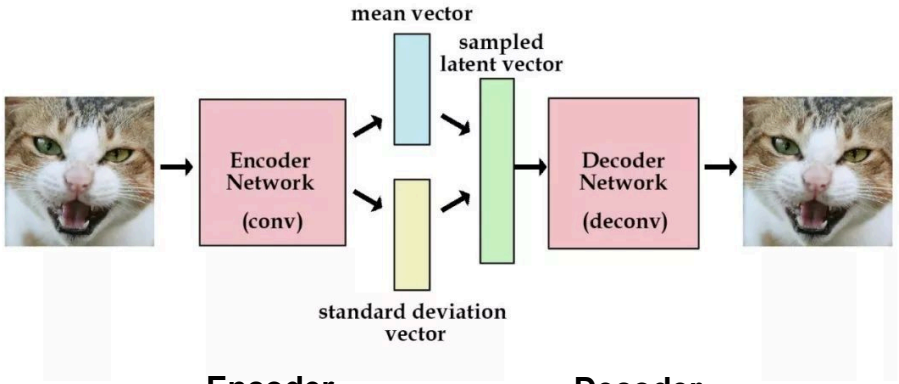
Encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

Decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$

Gelernt durch Variational Bayes:

- Verallgemeinerung von EM

Variational-Autoencoder: Einfache Erklärung



Encoder
$q_\phi(z|x)$

Decoder
$p_\theta(x|z)$

$q_\phi(z \mid x)$
gesampled aus encoder

Zwei Loss Terme

- Reconstruction: Genau wie bei “normalem” Autoencoder: Vergleich von $p_\theta(x \mid z)$ und x
- Vergleich der latenten Verteilung $q_\phi(z|x)$ mit einer Zielverteilung, normalerweise $\mathcal{N}(0,1)$

VAE Herleitung über amortisierte Variational Inference

Untere Schranke von EM: $$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i, z_i\right)\right] - \mathbb{E}_{q_i(z_i)} \left[\log q_i \left(z_i\right)\right]$$

VAE Herleitung über amortisierte Variational Inference

Untere Schranke von EM: $$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i, z_i\right)\right] - \mathbb{E}_{q_i(z_i)} \left[\log q_i \left(z_i\right)\right]$$

Anstatt eine individuelle Verteilung $q_i(\boldsymbol{z})$ pro Datenpunkt $\boldsymbol{x}_i$ zu verwenden, können wir eine "amortisierte" Verteilung $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)$ verwenden, die durch ein NN gegeben ist

VAE Herleitung über amortisierte Variational Inference

Untere Schranke von EM: $$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i, z_i\right)\right] - \mathbb{E}_{q_i(z_i)} \left[\log q_i \left(z_i\right)\right]$$

Anstatt eine individuelle Verteilung $q_i(z)$ pro Datenpunkt x_i zu verwenden, können wir eine "amortisierte" Verteilung $q_\phi(z|x_i)$ verwenden, die durch ein NN gegeben ist

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \mid \boldsymbol{z}\right) p(\boldsymbol{z})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log q_{\boldsymbol{\phi}} \left(\boldsymbol{z} \mid \boldsymbol{x}_i\right)\right]$$

VAE Herleitung über amortisierte Variational Inference

Untere Schranke von EM: $$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_i(z_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i, z_i\right)\right] - \mathbb{E}_{q_i(z_i)} \left[\log q_i \left(z_i\right)\right]$$

Anstatt eine individuelle Verteilung $q_i(z)$ pro Datenpunkt x_i zu verwenden, können wir eine "amortisierte" Verteilung $q_\phi(z|x_i)$ verwenden, die durch ein NN gegeben ist

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \mid \boldsymbol{z}\right) p(\boldsymbol{z})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log q_{\boldsymbol{\phi}} \left(\boldsymbol{z} \mid \boldsymbol{x}_i\right)\right]$$

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \mid \boldsymbol{z}\right)\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p(\boldsymbol{z})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log q_{\boldsymbol{\phi}} \left(\boldsymbol{z} \mid \boldsymbol{x}_i\right)\right]$$

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}} \left(\boldsymbol{x}_i \mid \boldsymbol{z}\right)\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}} \left(\boldsymbol{z} \mid \boldsymbol{x}_i\right)\right]$$

$\mathcal{N}(0, 1)$

Reconstruction loss — KL-divergence loss

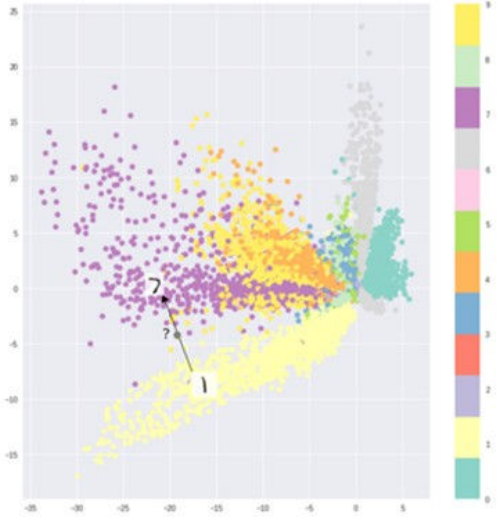
Dies ist das standard Objektive für Variational Auto-Encoder (VAE)

- Encoder $q_\phi(z|x)$
- Decoder $p_\theta(x|z)$

Visualisierung des Latenten Raums

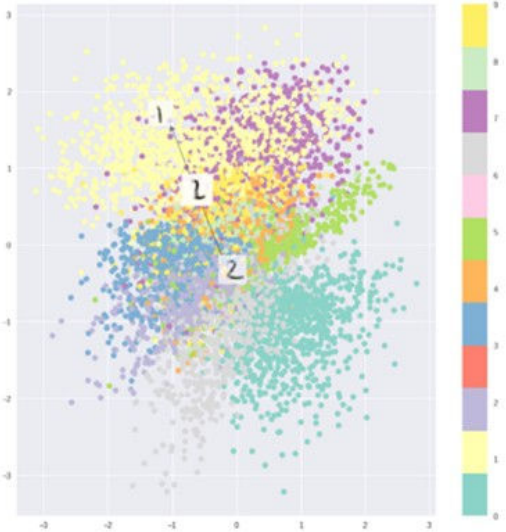
Auto-Encoder:

- Latenter Raum ist nicht zusammenhängend
- Keine sinnvolle Interpolation möglich



Variational Auto-Encoder:

- Probabilistischer latenter Raum wird besser „gefüllt“
- Ermöglicht Interpolation



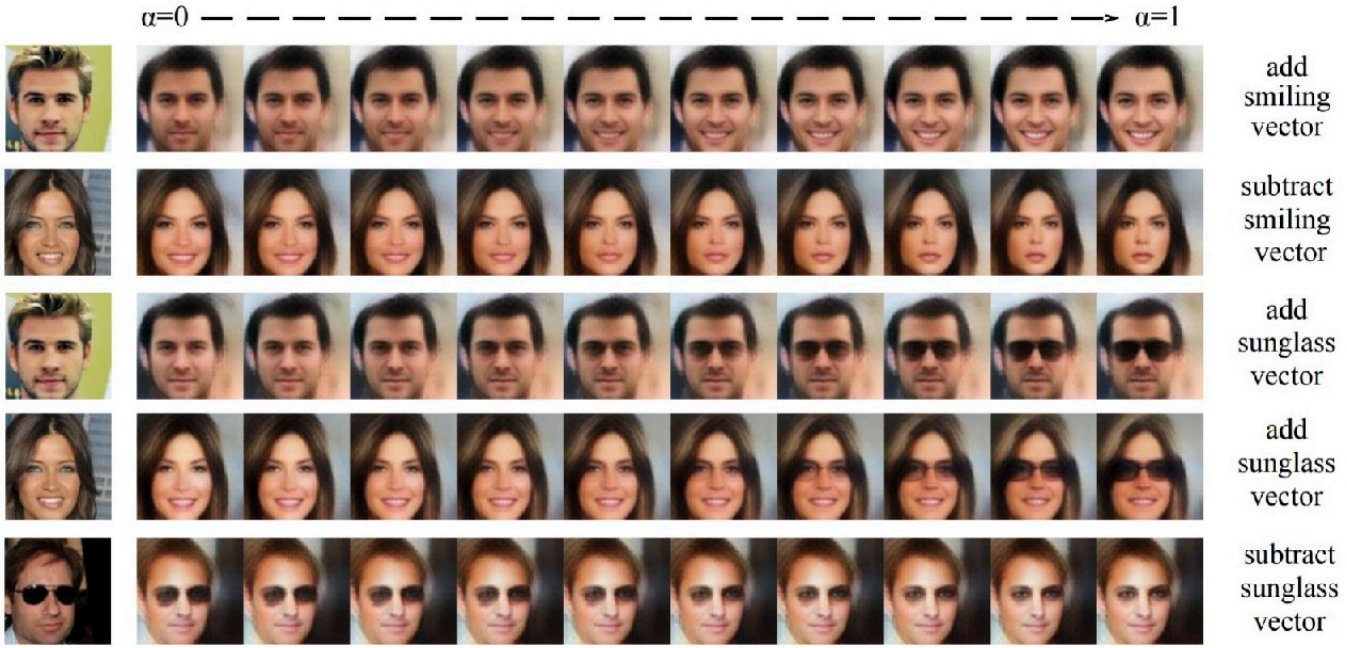
From: https://www.jeremyjordan.me/variational-autoencoders/

Interpolationen des latenten Raums

- Durch Interpolation zwischen zwei Vektoren im latenten Raum lassen sich oft interessante Ergebnisse erzielen:

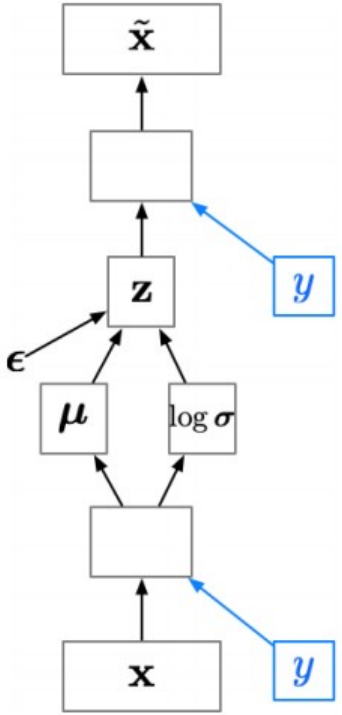
Ha and Eck, "A neural representation of sketch drawings"

Interpreting the latent space...



Klasse bedingte VAEs

- Bisher haben wir die Kennzeichnungen y nicht verwendet. Eine klassenbedingte VAE liefert die Kennzeichnungen sowohl an den Encoder als auch an den Decoder.
- Da der latente Code **z** nicht mehr die Bildkategorie modellieren muss, kann er sich auf die Modellierung der stilistischen Merkmale konzentrieren.
- Wenn wir Glück haben, können wir auf diese Weise Stil und Inhalt voneinander trennen. (Hinweis: Die Entflechtung ist immer noch eine dunkle Kunst.)
  - Siehe Kingma et al. "Semi-supervised learning with deep generative models".



Klassenbedingte VAE

- Indem man zwei latente Dimensionen (d.h. die Dimensionen von z) variiert, während man y konstant hält, kann man den latenten Raum visualisieren.

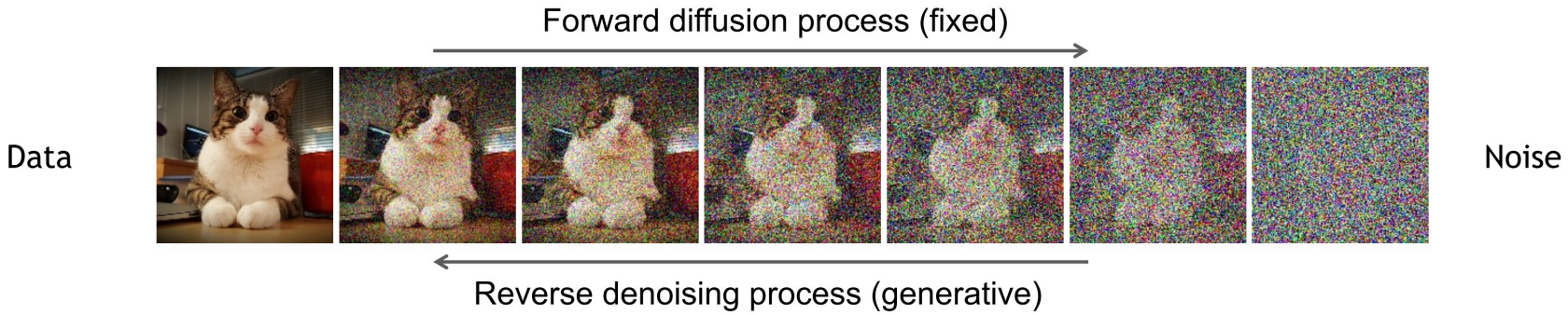
Klassenbedingte VAE

- Indem wir die Beschriftung y variieren, während wir **z** konstant halten, können wir Bildanalogien lösen

Outlook: Denoising Diffusion Probabilistic Models (DDPMs)

(Hierarchical) VAEs are the basis for diffusion models:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising
- SOTA in image generation
- Covered in Machine Learning Lecture…



Fragen zum Selbsttest

- Was sind Mixture-Modelle?
- Sollten Gradientenmethoden für das Training von Mixture-Modellen verwendet werden?
- Wie funktioniert der EM-Algorithmus?
- Was optimiert der EM Algorithmus?
- Warum verbessert EM immer die untere Schranke?
- Warum können wir jede Komponente unabhängig voneinander mit EM optimieren?