Bayes Theorem

$$P(z|x) = \frac{P(x,z)}{P(x)}$$

Problem: P(x) zu berechnen kann sich in der Praxis als schwierig erweisen[1]. Es ist also nicht möglich eine vollständige Enumeration durchzuführen um die fehlenden, versteckten Variablen z zu finden. Eine Alternative sind Monte Carlo Methoden wie Monte Carlo Markov Chain oder Variationale Methoden. Bei der Variationalen Methode wird versucht den Posterior P(z|x) mit einer Funktion q zu approximieren. Dieses Verfahren ist ähnlich zu dem Expectation Maximization Algorithmus und die fehlenden Variablen könnten Clusterzugehörigkeiten sein wie k-Means (hardclustering) oder auch eine Gaußsche Mischverteilungen in dem die einzelnen z Werte „Responsibilities" der Distribution darstellen.

Informationstheorie

Information: $I(x) = -\log P(x)$ Weniger wahrscheinliche Events haben einen höheren Informationsgehalt als Events mit einer hohen Wahrscheinlichkeit. (z.B. Trump tweets)

Entropy: $H = -\Sigma_{x \in X} P(x) * \log P(x) = \mathbb{E}_{x \sim P}[I(x)]$ Entropy liefert die Erwartete Menge an Informationen für die Distribution P

Kullback-Leibler-Abweichung:

$$D(P||Q) = KL(P,Q) = \Sigma_{x \in X} P(x) * \log \frac{P(x)}{Q(x)} = \mathbb{E}_{x \sim P}\left[\log \frac{P(x)}{Q(x)}\right]$$

$$= \Sigma_{x \in X} P(x) * (\log P(x) - \log Q(x))$$

Die KL-Abweichung gibt eine Distanz zwischen den beiden Distributionen P und Q an. Allerdings ist diese Distanz asymmetrisch da KL(P,Q) != KL(Q,P) ist.

Um die Abweichung zwischen den beiden Distributionen (Posterior und q) zu ermitteln benutzen wir die Kullback-Leibler-Abweichung (KL). Für sie gilt $KL \geq 0$ und die KL-Abweichung ist dann und nur dann gleich null, wenn die beiden Distributionen gleich sind.

$$q^*(z) = \arg_{q(z) \in D} \min KL(q(z)||\ P(z|x))$$

Per Definition der KL-Abweichung ergibt sich:

[1] Beispielsweise zeigt David Mackkay mit einer Rechnung, dass wenn die Dimension für z bei 1000 liegt und 2 diskrete Werte annehmen kann müssten wir bei der Enumeration sämtliche Kombinationen durchgehen und jeweils die Wahrscheinlichkeiten aufsummieren. Wenn jedes Elektron in dem Universum ein 1000 Gigahertz Computer wäre bräuchten wir immer noch 10^{190} „universe ages" um jeden Zustand einmal zu durchlaufen. Also eine lächerlich hohe Zahl die sich mit momentaner Technologie nicht berechnen lässt.
http://www.inference.org.uk/itprnn/book.pdf S.359

$$KL(q(z)||\,P(z|x)) = \mathbb{E}_{z\sim q(z)}\left[\log\left(\frac{q(z)}{P(z|x)}\right)\right] = \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big] - \mathbb{E}[\log \mathrm{P(z|x)}]$$

Durch Umformung der konditionalen Wahrscheinlichkeit und der Tatsache das $\mathbb{E}[\log(\mathrm{P(x)}] = \log P(x)$ ist, da sie unabhängig von dem Erwartungswert von z ist ergibt sich:

$$KL\big(q(z)||P(z|x)\big) = \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big] - \mathbb{E}\left[\log\left(\frac{P(z,x)}{P(x)}\right)\right]$$

$$= \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big] - \mathbb{E}[\log(\mathrm{P(z,x)})] + \log P(x)$$

Es ergibt sich folgende Gleichung:

$$KL(q(z)||\,P(z|x)) = \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big] - \mathbb{E}[\log(\mathrm{P(z,x)}] + \log P(x)$$

$$\log P(x) = \mathbb{E}[\log(\mathrm{P(z,x)}] - \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big] + KL(q(z)||\,P(z|x))$$

Unser Ziel ist es den Term $\log P(x) = \log(\sum_z P(x,z))$ zu maximieren. Dadurch, dass es sich um eine Summe innerhalb eines Logarithmus handelt kann die Maximum Likelihood Estimation nicht exakt durchgeführt und wir haben keinen Zugriff auf eine geschlossene Lösung mehr.

Wir können allerdings die beiden anderen Terme benutzen um $\log P(x)$ trotzdem zu optimieren. Sie bilden den sogenannten Evidence Lower Bound (ELBO) und die KL-Abweichung zwischen unserer Zielfunktion $\mathrm{q(z)}$ und $P(z|x)$. Da die KL-Abweichung nicht negativ sein kann und $\log P(x)$ eine Konstante darstellt gilt: ELBO ≤ log P(x). Wie gut unsere Approximation ist lässt sich also anhand der Formel ablesen:

$$\log P(x) = ELBO - KL$$

Durch Maximierung des ELBO wird die KL-Abweichung immer geringer und nähert sich immer weiter dem Wert von $\log P(x)$ an. Der ELBO dient in dem Sinne als Proxy für unsere eigene Optimierungsfunktion. Durch weitere Umformungen entsteht:

$$ELBO = \mathbb{E}[\log(\mathrm{P(z,x)}] - \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big] = \mathbb{E}\left[\log\left(\frac{P(z,x)}{q(z)}\right)\right]$$

$$= \mathbb{E}[\log(\mathrm{P(z)}] + \mathbb{E}[\log(\mathrm{P(x|z)}] - \mathbb{E}\big[\log\big(\mathrm{q(z)}\big)\big]$$

$$= \mathbb{E}[\log(\mathrm{P(x|z)}] - KL(q(z)||\,P(z))$$

Es ergibt sich die log-likelihood von $P(x|z)$ und die KL-Abweichung von unserem gewählten Prior und dem tatsächlichen Prior. Daran lässt sich schon ein Problem festmachen, nämlich dass sich unser Suchraum nur auf eine Dichtefunktionsfamilie beschränkt. Eine folge dessen ist, dass wir in der Praxis meist nie exakte Ergebnisse erhalten werden. Die KL-Abweichung bestraft in dem Sinne unsere ursprüngliche Annahme, mit der wir die Daten abbilden wollen, wenn sich diese Daten nicht, oder nicht gut, mit der Annahme abbilden lassen. Es ist in dem Sinne eine Beschränkung, dass wir verletzen können allerdings nur zu einem Preis.

Die Parallele zu dem Expectation Maximization (EM) Algorithmus ist nun das wir die Distributionen mit einem Set von Parametern θ beeinflussen können. Wir benutzen θ^{old} um einen neuen Wert für θ zu

ermitteln, der besser ist als die alten Parameter. Dazu setzen wir $q(z) = P(z|x, \theta^{old})$[2] also unserer momentanen besten Lösung für z und optimieren anschließend.

$$ELBO = \mathbb{E}\left[\log\left(\frac{P(z,x)}{q(z)}\right)\right] = \sum_z q(z) * \log\frac{P(z,x)}{q(z)} = \sum_z P(z|x,\theta^{old}) * \log\frac{P(z,x)}{P(z|x,\theta^{old})}$$

$$= \sum_z P(z|x,\theta^{old}) * \log P(z,x) - \sum_z P(z|x,\theta^{old}) * \log P(z|x,\theta^{old})$$

$$= \sum_z P(z|x,\theta^{old}) * \log P(z,x) + Const$$

Der Term ist die formale abstrakte Definition des EM Algorithmus[3]. Wenn wir für die gemeinsame Wahrscheinlichkeit einen neuen Parameter einführen θ können wir eine neue Funktion erstellen die sich wie folgt ergibt:

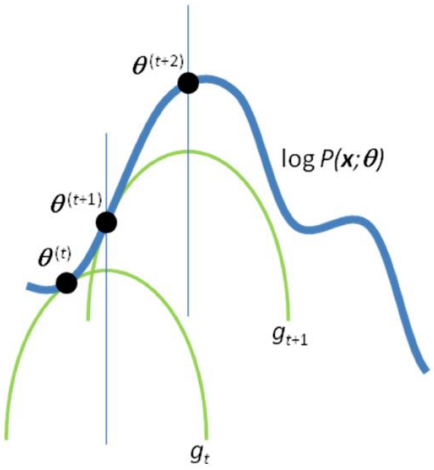
$$Q(\theta, \theta^{old}) = \sum_z P(z|x,\theta^{old}) * \log P(z,x|\theta)$$

Der neue Wert für Theta ergibt sich daraus aus dem „Maximization“ Schritt:

$$\theta^{new} = \arg_\theta \max Q(\theta, \theta^{old})$$

Die Formel wirkt als eine Art Zyklus, in dem wir die immer neuen Erwartungen berechnen und anschließend die Erwartungen maximieren (also das abwechselnde „einfrieren“ der Parameter).

Das folgende Bild zeigt dabei die Schrittweise Verbesserung des Parameters θ:



Supplementary Figure 1 Convergence of the EM algorithm. Starting from initial parameters $\theta^{(t)}$, the E-step of the EM algorithm constructs a function g_t that lower-bounds the objective function $\log P(x;\theta)$. In the M-step, $\theta^{(t+1)}$ is computed as the maximum of g_t. In the next E-step, a new lower-bound g_{t+1} is constructed; maximization of g_{t+1} in the next M-step gives $\theta^{(t+2)}$, etc.

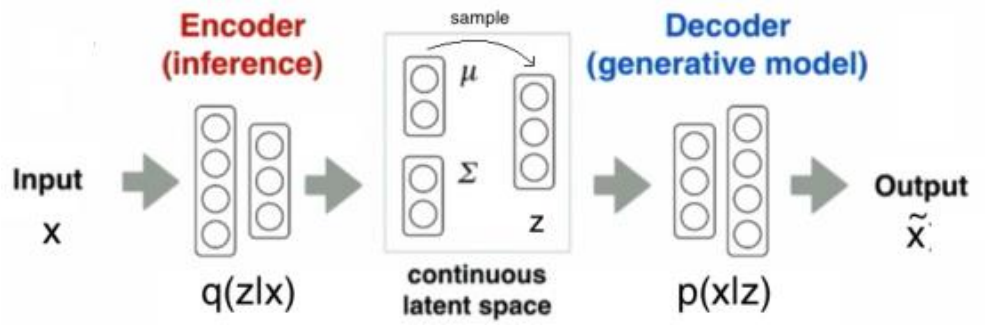
[2] Ein unwichtiges Detail: Wenn wir Variational inference durchführen wollen können wir beispielsweise für q folgendes einsetzten: $q(z) = \prod_i q_i(z_i)$ also das die Distributionen untereinander unabhängig sind das ist der sogenannte „Mean-field apporach“. Man optimiert die einzelnen Distributionen unabhängig indem man eine Distribution wählt zum optimieren und die anderen Distributionen „einfriert“. Dieser Ansatz wird häufig beim Topic Modeling (also Text Analyse) verwendet würde aber an dieser Stelle zu weit führen. Prinzipiell konvergiert Variational Inference schneller als Monte Carlo Methoden allerdings zu dem Preis von Genauigkeit.

[3] Nach Christopher Bishop - Pattern Recognition And Machine Learning - Springer 2006 S.440f.
http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf

Variational Autoencoder:

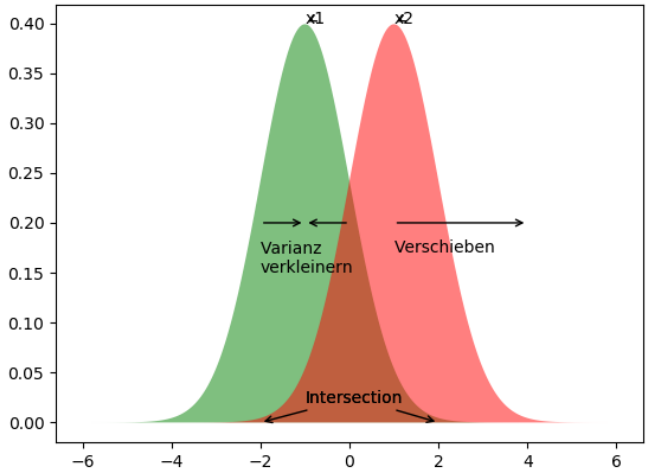
Dieser ELBO Term wird nun bei einem Variational Autoencoder (VAE) mit einem neuronalen Netz (bei Bildern mit einem konvolutionalen Netz) kombiniert.

q(z) wird zu einer Funktion q(z|x) und soll die versteckte Struktur z in Abhängigkeit von der Eingabe x finden die es dem generativen Model erlaubt den ursprünglichen Input x wiederherzustellen.



Dabei ist q(z|x) eine Kodierungsfunktion und p(x|z) eine Dekodierungsfunktion bzw. ein Generatives Model. Im Gegensatz zu herkömmlichen Autoencodern ist das ein probalistisches setting und kein deteministisches, da aus den vorhergesagten Distributionen jeweils gesampled wird.

Dadurch, dass Distributionen vorhergesagt werden entstehen Bereiche, in denen sich die Wahrscheinlichkeitsmasse überlappt. Das ist in dem Sinne ein Problem, da der VAE oftmals nicht unterscheiden kann zwischen zwei verschiedenen Eingaben und somit nicht exakt die Eingabe wiederherstellen kann. Der VAE kann das Problem auf prinzipiell auf drei verschiedene Vorgehensweisen lösen. Die erste Lösung wäre es, μ so anzupassen sodass sich möglichst wenig Wahrscheinlichkeitsmasse überlappt. Mit anderen Worten verschiebe die Distribution nach links bzw. nach rechts und verringere die überlappenden Wahrscheinlichkeiten. Allerdings verstößt dieses Verhalten gegen das weiche Constraint der KL-Abweichung. Dem Prior P(z) (beispielsweise einer Normalverteilung N(μ = 0, σ =1)) den wir vorab definieren. Das Gleiche gilt auch für den Ansatz der Reduzierung der Varianz, bzw. die Zufallskomponente zu entfernen, um möglichst deterministische Repräsentation zu erhalten, welches ebenfalls gegen die KL-Abweichung verstößt. Die letzte Möglichkeit ist es, ähnliche Objekte nahe beieinander anzuordnen. Diese Objekte unterscheiden sich beispielsweise in der x-y-Position, der Größe oder ähnlichem. Dieser Ansatz verstößt gegen keinerlei Constraint und sorgt dafür, dass (hoffentlich) plausible Repräsentationen gelernt werden. Die nachfolgende Abbildung zeigt diese Problemstellung samt den ersten beiden Lösungsansätze anhand von zwei Trainingsbeispielen.



Der Balanceakt zwischen dem ELBO Term und der KL-Abweichung stellt dabei im Wesentlichen die Aufgabe des VAE dar. Deswegen gibt es eine spezielle Variante von VAE die sogenannten Beta-VAE.

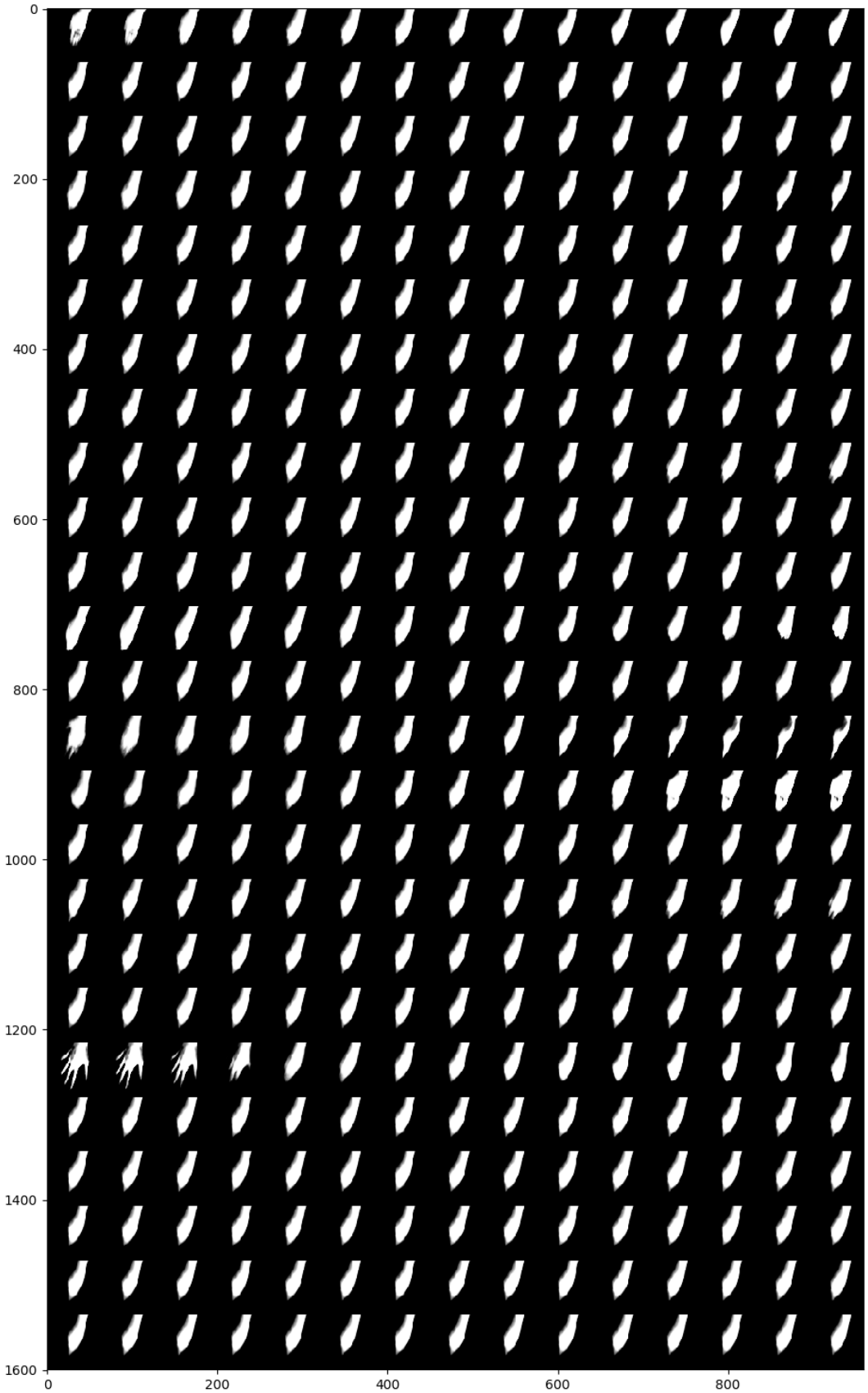
$$ELBO = \mathbb{E}[\log(\mathrm{P}(\mathrm{x}|\mathrm{z})] - \beta * KL(q(z)||\, P(z))$$

Diese Form forciert eine Präferenz auf die KL-Abweichung. Beta ist üblicherweise größer 1 und stellt dem Algorithmus die Frage ob es besser wäre den Rekonstruktionsfehler zu reduzieren oder das Constraint zu erfüllen. Mit anderen Worten die KL-Abweichung bekommt mehr Gewicht bei der Performancefunktion als der Rekonstruktionserror. Es ist für den Algorithmus erstrebenswerter größere Bereiche Wahrscheinlichkeitsmasse einzubeziehen und die Varianz ähnlich zu dem Prior zu machen. Gleichzeitig werden weniger Feature benutzt, da es von Nachteil ist von dem Prior zu weit abzuweichen. Ein Nachteil ist allerdings, dass der β – VAE multiple Distributionen benutzt um ein einzelnes Feature vorherzusagen, da mit jeder Verteilung ein Stück der Zufallskomponente entfernt wird und aus dem Zufallswerts ein Durchschnitt bzw. ein Erwartungswert wird.

In der Regel lernt der Beta-VAE Features, die eher unabhängig sind. Also die Features keinerlei oder wenig Korrelation aufweisen.

Die folgende Abbildung zeigt beispielsweise eine latente[4] Repräsentation mit 25 Dimensionen einer Hand. Die horizontalen Bilder zeigen die Veränderung, wenn die Komponenten um einen kleinen Wert verändert werden (von -3 bis +3 in 15 Schritten):

[4] Beispielsweise wenn wir den VAE als Graphisches Model interpretieren können wir nur x beobachten und die Struktur von dem Model ist: (z)→(x). Wir führen Inferenz durch um die fehlenden Daten z zu ermitteln.


0
200
400
600
800
1000
1200
1400
1600
0
200
400
600
800

Die Struktur die der VAE gelernt hat die „Mannifold“ der Daten: