Generative Adversarial Networks

Marcel Canclini, marcel.canclini@gmail.com

23 April 2017

Abstract

2014 wurde mit Generative Adversarial Nets (Goodfellow u. a. 2014) ein neues Framework zur Erstellung von generativen Modellen vorgeschlagen. Dabei werden 2 sich konkurrenzierende Neuronale Netze trainiert, wobei das erste (der Generator) versucht das zweite (der Diskriminator) von der Echtheit seines erstellten Samples zu überzeugen. Das Trainieren der Modelle bringt einige Schwierigkeiten mit sich, welche mit angepassten Architekturen und Trainingsverfahren wie Wasserstein GAN (Arjovsky, Chintala und Bottou 2017) oder BEGAN (Berthelot, Schumm und Metz 2017) verbessert werden können.

Generative Adversarial Nets

Der verblüffend einfach zu verstehende Ansatz von (Goodfellow u. a. 2014) um generative Modelle zu erstellen, besteht darin jeweils einen Generator und einen Diskriminator mittels einem neuronalen Netz abzubilden und zu trainieren. Dadurch kann das Gesamtsystem mittels Backpropagation trainiert werden und benötigt weder Markov-Ketten noch approximative Inferenz während des Trainings oder der Generierung von Samples.

Die Aufgabe des Diskriminator D besteht in der Beurteilung ob ein Sample aus der Verteilung der echten, zu Trainingszwecken zur Verfügung gestellten, Daten, oder aus den durch den Generator G erstellten Fälschungen kommt. Der Generator hingegen versucht so gute Fälschungen zu erstellen, dass der Diskriminator diese als echt beurteilt.

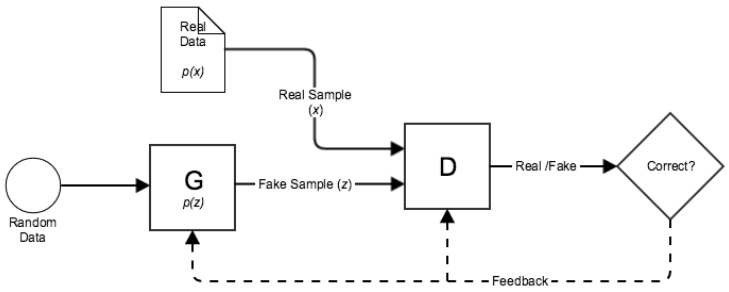

Figure 1: Vereinfachte Architektur eines Generative Adversarial Nets

Das Trainieren des Netz ist als minimax Spiel zu verstehen. Dabei wird D trainiert um die Wahrscheinlichkeit der richtigen Klassifizierung des Samples zu maximieren.

$$maxV(D) = log(D(x)) + log(1 - D(G(z)))$$

Gleichzeitig wird G trainiert um die korrekte Klassifizierung der durch G erstellen Samples zu minimieren.

$$minV(G) = log(1 - D(G(z)))$$

Das Trainieren des Gesamtsystems läuft über mehrere Iterationen, wobei die Optimierung des Diskriminators in einer inneren Schlaufe mit k Schritten durchgeführt wird. k ist ein Hyperparameter des Trainings und wird häufig kleiner 1 gewählt, damit der Generator besser trainiert wird. Das Vorgehen ist mit sehr klarem Pseudocode in (Goodfellow u. a. 2014, 4, Algorithm 1) beschrieben.

Schwachpunkte und Stärken

Das beschriebene System hat natürlich auch Nachteile. Ein wichtiger Punkt ist die Fragilität des Trainings. G und D müssen gut synchronisiert sein, wobei G gegenüber D nicht zu stark trainiert werden darf, da ansonsten G sich auf ein bestimmtes Sample einstellt. Gerade zu Beginn des Trainierens ist es für D zu einfach ein generiertes von einem echten Bild zu unterscheiden. Ein weiterer Nachteil ist die fehlende explizite Repräsentation von $p_g(x)$, also wie der Generator die ein Sample x erstellt. Dem gegenüber stehen die bereits erwähnten Vorteile der durchgehenden Ableitung und somit der Gewinnung der Gradienten durch Backpropagation. Das Training benötigt somit keine Markov-Ketten.

Weiterführende Arbeiten

Auf Basis von "Generative Adversarial Nets" (Goodfellow u. a. 2014) wurden weitere Arbeiten veröffentlicht, welche sich speziell mit der Verbesserung des Trainings auseinandersetzen. Diese liefern unterschiedliche Ansätze in der Definition der Loss Funktion und der Architektur. Im weiteren werden die zwei Ansätze **Wasserstein GAN** und **BEGAN** betrachtet.

Wasserstein GAN

Wasserstein GAN (Arjovsky, Chintala und Bottou 2017) bietet eine Alternative zur verwendeten Kullback-Leibler (KL) Distanz aus dem ursprünglichen Ansatz um folgendes Problem zu adressieren.

Wenn die wahre Verteilung $\mathbb{P}_r$ und die Modellverteilung $\mathbb{P}_\theta$ keine Überschneidung haben, kann die KL Distanz nicht definiert werden. Dem wird bei generativen Modellen entgegengewirkt, indem Zufallswerte aus einer Gaussverteilung mit hoher Bandbreite hinzugefügt wird. Solche Zufallswerte beeinflussen aber die Qualität der Resultate (Bilder). Diese wirken häufig verschwommen.

Als Alternative wird die Earth Mover (EM) Distanz (Wasserstein-1) vorgestellt. Diese definiert sich wie folgt.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\ \gamma}[||x - y||]$$

In Kapitel 2 (Arjovsky, Chintala und Bottou 2017, 3) werden verschiedene Distanzmasse anhand des Beispiels von Parallelen Linien miteinander verglichen. Berechnet werden jeweils die Jensen-Shannon (JS), die Kullback-Leibler (KL), Total Variation (TV) sowie die Earth Mover (EM) Distanz. Es zeigt, dass nur bei der EM Distanz die parametrische Verteilung $\mathbb{P}_\theta$ zu $\mathbb{P}_0$ konvergiert und somit Gradienten zum Trainieren eines Netz vorhanden sind.

Im Paper wird die EM Distanz anhand verschiedener beschriebener Theoreme optimiert und schlussendlich in einem WGAN Algorithmus (Arjovsky, Chintala und Bottou 2017, 8) als Pseudocode angewendet. Ein WGAN trainiert keinen Diskriminator, sondern den sogenannten *critic*. Dieser liefert dann die entsprechenden Gewichte zuhanden des, mittels Backpropagation trainierten, Generator.

Das 'klippen' der Gewichte w ist eine nötige aber unschöne Art um die Liptschitzfunktion zu erzwingen. Je grösser der Klipp-Hyperparameter c desto länger dauert das Trainieren. Bei zu kleinem c kann es zu verschwindenden Gradienten kommen.

Mit WGAN ist eine Alternative zum Trainieren von Generative Adversarial Nets vorhanden, welche auf einer stets ableitbaren Loss Funktion beruht und somit ein stabileres Trainieren zulässt, sowie keinen modalen Kollaps (nur ein Bild wird gelernt) hat.

Boundary Equilibrium Generative Adversarial Networks

Ganz aktuell (April 2017) wurde das BEGAN Framework vorgestellt. Diese adressiert die Probleme der Bildqualität sowie des instabilen Trainings.

Architektur

Wie bei Energy Based GAN (Zhao, Mathieu und LeCun 2016) verwendet BEGAN einen Auto-Encoder als Diskriminator. Die mit BEGAN vorgestellte Architektur (Berthelot, Schumm und Metz 2017, 5) basiert auf 2 einfachen Bausteinen. Ein Encoder, welcher auch gleichzeitig als Generator verwendet wird (jedoch mit anderen Gewichten) und ein Decoder. Beide bestehen nur aus Convolution Layern. Beide Architekturentscheide (Generator = Decoder, nur Convolution) wurden aus Gründen der Einfachheit getroffen. Der Generator besteht aus dem Decoder. Der Diskriminator setzt sich zusammen aus dem Encoder und dem Decoder. Die Loss-Funktion wird zwischen dem Eingabe- und Ausgabebild des Diskriminator berechnet.

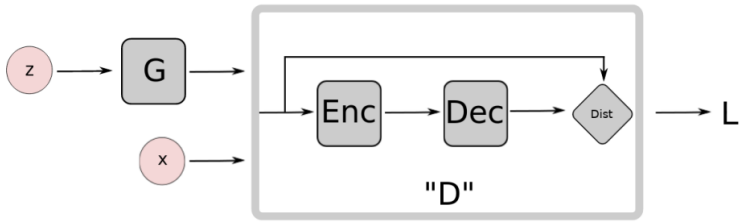

Figure 2: EBGAN /BEGAN Architektur,(Zhao, Mathieu und LeCun 2016, 4)

Loss Funktion

Statt die Distanz der Verteilungen $\mathbb{P}_r$ und $\mathbb{P}_g$ zu messen, werden mit BEGAN die Distanzen der Auto-Encoder Loss Verteilungen $\mathcal{L}()$ gemessen. Dazu wird ein Auto-Encoder auf dem realen und dem generierten Bild angewendet. Mittels der Wasserstein Distanz zwischen den Rekonstruktionsfehlern, welche anhand des zentralen Grenzwertsatzes einer Normalverteilung entsprechen, wird der wahre Fehler abgeleitet.

Dies führt zu folgenden Termen, welche durch D und G maximiert werden sollen:

$$\mathcal{L}_D = \mathcal{L}(x) - \mathcal{L}(G(x))$$

$$\mathcal{L}_G = \mathcal{L}(G(x))$$

Equilibrium Term

Das fragile Training wird bei BEGAN mittels des Equilibrium Term stabilisiert. Dazu werden die Fehler des Diskriminators und des Generators fortlaufend ausbalanciert. Da ein perfektes Equilibrium zu Instabilität führt, wird ein neuer Hyperparameter $\gamma \in [0, 1]$ definiert.

$$\gamma = \frac{\mathbb{E}[(\mathcal{L}(G(z))]}{\mathbb{E}[(\mathcal{L}(x))]}$$

Mit γ kann nun Bildqualität (kleines γ) gegenüber hoher Diversität (grosses γ) eingestellt werden. Um γ über die Zeit des Trainings stabil zu halten wird ein adaptiver Term $k_t \in [0, 1]$ eingeführt. Dieser wird anhand der "Proportional Control Theory" für jeden weitern Trainingsschritt angepasst.

$$\mathcal{L}_D = \mathcal{L}(x) - k_t * \mathcal{L}(G(x))$$

Die Loss Funktion zusammen mit dem Equilibrium Term führt zu folgender Definition eines BEGAN:

$$\mathcal{L}_D = \mathcal{L}(x) - k_t * \mathcal{L}(G(x))$$

$$\mathcal{L}_G = \mathcal{L}(G(x))$$

$$k_t + 1 = k_t + \lambda * (\gamma * \mathcal{L}(x) - \mathcal{L}(G(x)))$$

wobei λ ein Hyperparameter für die Lernrate ist (kleiner Wert. z.B. 0.001)

Diese BEGAN Definition lässt ein simultanes Training von G und D zu.

Schlussfolgerung

Rund um das Thema Generative Adversarial Nets läuft gerade einiges. Sowohl WGAN als auch BEGAN wurde in 2017 publiziert. BEGAN bringt die Möglichkeit ein Generatives Netz stabil und simultan zu trainieren und dabei auch noch einzustellen ob der Generator eher realistische oder Bilder mit hoher Diversität generieren soll. Die Resultate von BEGAN sind beeindruckend und der Kommentar, dass keine Brillen, nur wenige älter Menschen und mehr Frauen als Männer dargestellt werden, wird sicher noch einige Experimente nach sich ziehen. Lernt ein BEGAN das "Durchschnittliche Gesicht" oder entspricht dies einfach den Trainingsdaten?

Spannend wird auch sein, ob diese Art der Generierung auch den Einsatz in anderen Domänen wie Text oder Audio finden wird.

Interessant wäre ein qualitativer Vergleich eines GAN mit einem BEGAN. Man könnte separat ein GAN und ein BEGAN auf denselben Bildern trainieren. Anschliessend gibt man dem Diskriminator des GAN die erstellten Bilder des BEGAN Generators und misst wie viele der Bilder als echt beurteilt werden.

Referenzen

Arjovsky, Martin, Soumith Chintala und Léon Bottou. 2017. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]* (Januar). http://arxiv.org/abs/1701.07875 (zugegriffen: 14. April 2017).

Berthelot, David, Thomas Schumm und Luke Metz. 2017. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *arXiv:1703.10717 [cs, stat]* (März). http://arxiv.org/abs/1703.10717 (zugegriffen: 14. April 2017).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville und Yoshua Bengio. 2014. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]* (Juni). http://arxiv.org/abs/1406.2661 (zugegriffen: 14. April 2017).

Zhao, Junbo, Michael Mathieu und Yann LeCun. 2016. Energy-Based Generative Adversarial Network. *arXiv:1609.03126 [cs, stat]* (September). http://arxiv.org/abs/1609.03126 (zugegriffen: 14. April 2017).