

# Generative Adversarial Networks for Text Generation

Sven Vaupel

xx.xx.2017

## 1 Overview

Giving a basic view over what is planned. Especially what I plan to describe before I tackle GANs!

1. Notation ("copy" from [1])
2. Neural Networks
  - What are Neural Networks and how do they work. A brief introduction.
  - (a) Use cases
    - Giving a quick overview of Regression and Classification using simple examples.
  - (b) Important terms
    - Giving a quick overview over the terminology used specifically in machine learning
    - i. Hyperparameter
      - The Hyperparameters' values are set prior to the actual training process.
    - ii. Stochastic gradient
    - iii. Over- / Underfitting
  - (c) Important concepts
    - Concepts which are important for the explanation of later topics

- i. Backpropagation  
Why is it Important and what are the requirements.
- ii. Minibatch Training
- iii. Dropout
- iv. Latent space?  
Could be of interest for some of the papers. What Information can be drawn from it. Showing some examples.

### 3. GAN

## 2 Generative Adversarial Networks

Appart from classification and data prognosis there is the field of data generation. The different networks existing for this task mostly consist of non-differentiable functions, thus the function's gradient has to be estimated to use backpropagation during the network's training process. This estimation is one of the main causes for over- and underfitting in such networks [2]. {Could add information on some of these Networks? See book chapter 20}

Goodfellow et al. [2] presented a different approach to tackle the generation of data according to a set of examples. The method called generative adversarial networks (GAN) connects the output of a differentiable data generating network to the input of a driscriminative multilayer perceptron. Connecting the networks' in- and outputs in such fashion, it is now possible to train the generator by minimizing the probability that the attached discriminator is able to correctly distinguish between example- and artificially generated data. The fact that both connected networks are based on differentiable functions makes the learning phase feasible for backpropagation. Therefore GANs are independent of error-prone gradient estimations.

### 2.1 Model Definition

The foundation for the training is a game theoretic scenario where the generator  $G$  tries to beat the adversary  $D$  in a zero sum game. Based on the set of data  $x$ , by defining a noise prior  $p_z(z)$  it is possible to provide  $G(z; \theta_g)$  which represents a mapping from the noise variable  $z$  to the given data space  $x$  where  $\theta_d$  are the parameters of a multiplayer perceptron representing the

differentiable function  $G$ . As obvious  $G$  serves as the generating network which creates data space samples  $x$  from a random noise  $z$ . The discriminator on the other hand takes any samples  $x$  from the data space and assigns them a scalar value according to the probability whether the sample originates from the original training data or the data generated by  $G$ . Similar to  $G$ 's definition, we construct a differentiable function  $D(x; \theta_d)$  with underlying perceptron parameters  $\theta_d$  representing the discriminator.  $D$  then calculates the odds for  $x$  originating from the training distribution  $p_d$  rather than the generated distribution  $p_g$  [2, 1].

Regarding the goal of learning the generator's distribution  $p_g$  over the data space, the GAN's learning process can be described as a competition of  $G$  and  $D$  in a two player minimax game using the value function  $V(D, G)$  such that the discriminators payoff is calculated by:

$$\min_g \max_d V(G, D) = \mathbb{E}_{x \sim p_d} [\log D(x; \theta_d)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z; \theta_g); \theta_d))] \quad (1)$$

[2, Equation (1)]

## 2.2 Training Process

Training is then done by maximizing the probability of  $D$  assigning the correct label to the samples coming from both the training distribution  $p_d$  and the generated distribution  $p_g$  while simultaneously minimizing the probability of  $G$  generating samples to which  $D$  confidently assigns the generated data as origin. In other words we train the discriminator to be able to sort out artificial samples while at the same time training the generator to fool the discriminator with its imitations. This process results in an equilibrium where even a maximally trained discriminator can not distinguish generated from original samples thus assigning  $D(x) = 1/2$  to every input. When this happens we reached the point where the generated and the training distribution match such that  $p_g = p_d$  [2].

Implementation wise it is only possible to approach the training procedure with an iterative numerical process. This approach is shown in Algorithm 1.

---

**Algorithm 1** Training process of generative adversarial networks.

---

- Applying minibatch training for data sampling.
  - Using stochastic gradient descent to optimize the discriminator and generator.
  - Applying hyperparameter  $k$ . Resulting in a relation of  $k : 1$  training steps between the discriminator and the generator.
- 

for *number of training iterations* do

for *k steps* do

1. Sample minibatch of  $m$  noise samples  $\{z_1, \dots, z_m\}$  from noise prior  $p_g(z)$
2. Sample minibatch of  $m$  noise samples  $\{x_1, \dots, x_m\}$  from data generation distribution  $p_d(x)$
3. Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \sum_{i=1}^m [\log D(x_i) + \log(1 - D(G(z_i)))]$$

end for

1. Sample minibatch of  $m$  noise samples  $\{z_1, \dots, z_m\}$  from noise prior  $p_g(z)$
2. Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \sum_{i=1}^m [\log(1 - D(G(z_i)))]$$

end for

[2, Algorithm (1)]

---

As stated in Algorithm 1 the inner loop does not train the discriminator to completion but limits the learning process by applying a constraining hyperparameter  $k$ . A completely trained perceptron  $D$  would result in overfitting on the training distribution  $p_d$  focusing the generator on creating exact copies of the training examples. Therefore the algorithm keeps  $D$  near its optimum by alternating between training  $D$  for  $k$  steps and training  $G$  for one step [2]. Regarding the network’s early training phase,  $\log(1 - D(G(z)))$  could saturate in cases where  $D$  confidently assigns correct labels to all data samples. Thus early on maximizing  $\log D(G(z))$  provides significantly higher gradients than minimizing  $\log(1 - D(G(z)))$  without harming the algorithm’s dynamics. In this case the training process will maximize the probability that the discriminator makes a mistake rather than minimizing its probability of assigning correct values [2].

### 2.3 Theoretical Results and Boundries

When optimizing in the space of probability distributions Goodfellow et al. [2] showed that Algorithm 1 maximizes Equation (1) such that  $p_g$  converges to  $p_g$ . If the distributions are represented by parameterized neural networks however the convergence of  $p_g$  is not guaranteed. The generated distribution’s representation  $G(x; \theta_g)$  limits the resulting  $p_g$  to a restricted family of probability distributions such that  $\max_d (V(D, G))$  is not guaranteed to be convex.

Possible equilibria in such a scenario are saddle points of  $V(D, G)$  where a local maximum in the first dimension and a local minimum in the second is reached. For those points the alternation seen in Algorithm 1 could result in a loop where the gradient based optimization circulates around a saddle without converging to it [1]. Goodfellow et al. [3] identified this behaviour as one possible source of underfitting in existing GAN implementations. Even if multilayer perceptrons restrict the space of possible distributions and therefore do not give any theoretical guarantees for convergence, the performance of approaches using multilayer perceptrons suggests that they are a legitimate model to use. [2]

## References

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org>
- [2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>
- [3] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks,” *ArXiv e-prints*, Dec. 2013. [Online]. Available: <http://adsabs.harvard.edu/abs/2013arXiv1312.6211G>