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 [2])
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 nondifferentiable 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 errors in such networks. {Could add information on some of these Networks? See book chapter 20}

Goodfellow et al [1] presented a different approch 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 drscriminative 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.

2.1 Model Definition

The foundation for this 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 θ_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 θ_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.

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 $V(D, G)$ so that the descriminators payoff is calculated by:

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

[1, Equation (1)]

2.2 Training Process

Training is then done by maximizing the probablility of D assigning the correct label to the samples coming from either the training distribution p_d or the generated distribution p_g while simultaniously minimizing the probability of G generating samples to which D confidently assigns the training 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 immitations. 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 trainning distribution match such that $p_g = p_d$.
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:

for $numberoftrainingiterations$ do

for $ksteps$ do

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

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

end for

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

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

end for

[1, 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 exaxt 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. 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 maximizing $\log D(G(z))$ provides significantly higher gradients than minimizing $\log(1 - D(G(z)))$ without harming the algorithm's dynamics.

3 Theoretical Results and Boundries

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.