

Advanced Machine Learning Subsidiary Notes

Lecture 15: Wasserstein GANs

Adam Prügel-Bennett

January 28, 2021

1 Keywords

- GANs, Wasserstein distance, Duality, WGANs

2 Main Points

2.1 Generative Adversarial Networks GANs

- GANs are generative models
- They are often used for generating images or text
 - we will consider images just to be concrete but nothing really changes if we use text
- The task of the GAN is to generate images from the same distribution as those of a dataset \mathcal{D}
- GANs use two networks
 1. *A generator network*
 - It is fed a random vector $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - They are usually networks made with fully connected and deconvolution layers with weights w_G
 - They generate an "image" $\hat{x} = G(z, w_G)$
 - We train the weights to deceive the discriminator network
 2. *A discriminator network*
 - They receive either
 - * a random sample from \mathcal{D} or
 - * an image \hat{x} generated by the generator seeded with a random vector z
 - They are trained using backpropagation where the target output is
 - * 1 for the real image from \mathcal{D} or
 - * 0 if the input is from the generator
 - They are usually CNN networks
- The generator weights are also learned by back-propagation through both the discriminator and generator networks where the target is for the discriminator for output 1
 - the discriminator weights aren't changed
 - this is opposite to the loss for the discriminator
 - it is fed the information about how to fool the discriminator (i.e. how to change the elements of \hat{x} to maximise the output of the discriminator)

- Hopefully over time the generator produces image more like those from \mathcal{D}

- **Problems with GANs**

- GANs are notoriously hard to train
- The training of the discriminator and generator can become decoupled
- For example, the discriminator can become so good that any local change of \hat{x} doesn't fool the discriminator
 - * but this means there is no gradient to direct the learning of the generator
- To overcome this we consider a very different approach

2.2 Wasserstein Distance

- The Big Picture

- We consider minimising the distance between the distribution of images generated by the generator (that is, the distribution of $\hat{x} = G(z, w_G)$ where $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) and the distribution of real images (where we consider \mathcal{D} to be a set of samples drawn from this distribution)
- How do we measure distances between probability distributions?
- One of the most common methods is to use the KL-divergence

$$\text{KL}(p||q) = \int p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{y})}\right) d\mathbf{x}$$

- * Relatively nice to compute
- * Not a true distance (but that doesn't bother us)
- * Unfortunately it can get very large even when the probability distributions are relatively close together

- **Earth-Moving Distance**

- A very natural distance measure is the minimum distances you have to move the probability mass in one distribution $p(\mathbf{x})$ to make it identical to a second distribution $q(\mathbf{x})$
- This is also known as the *Wasserstein* distance
- Although conceptually straightforward it is a bit nasty to compute
- **Optimal Transport Policy**

- * We start from a transport policy $\gamma(\mathbf{x}, \mathbf{y})$ that tells us how much probability mass (or density) we need to move from probability distribution p at point \mathbf{x} to probability distribution q at point \mathbf{y}
- * As we start with a distribution $p(\mathbf{x})$ we need

$$\int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y} = p(\mathbf{x})$$

- * As we end with a distribution $q(\mathbf{x})$ we require

$$\int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} = q(\mathbf{y})$$

- * Note that $\gamma(\mathbf{x}, \mathbf{y})$ looks like a joint probability distribution
 - It is non-negative
 - Integrating over \mathbf{x} and \mathbf{y} we get 1

- * We denote the set of probability distributions that satisfy these constraints $\Lambda(p, q)$
- * The cost of a particular transport policy is

$$C(\gamma) = \iint d(\mathbf{x}, \mathbf{y}) \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} = \mathbb{E}_{\gamma}[d(\mathbf{x}, \mathbf{y})]$$

since $\gamma(\mathbf{x}, \mathbf{y})$ is the amount of probability mass we move and $d(\mathbf{x}, \mathbf{y})$ is the distance we move it

- * The optimal transport policy is the distribution $\gamma \in \Lambda(p, q)$ with the minimum cost
- * The cost of the optimal transport policy is the Wasserstein distance

$$W(p, q) = \min_{\gamma \in \Lambda(p, q)} \mathbb{E}_{\gamma}[d(\mathbf{x}, \mathbf{y})]$$

- * For high dimensional probability distributions finding the optimal transport policy using this definition is impractical

- Linear Programming

- * Computing the Wasserstein distance is a linear programming problem
- * We want to choose $\gamma(\mathbf{x}, \mathbf{y})$ to minimise a linear objective function $C(\gamma)$ subject to linear constraints
- * We can write this as a Lagrange problem

$$\begin{aligned} \mathcal{L} = \iint d(\mathbf{x}, \mathbf{y}) \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} &- \int \alpha(\mathbf{x}) \left(\int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y} - p(\mathbf{x}) \right) d\mathbf{x} \\ &- \int \beta(\mathbf{y}) \left(\int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} - q(\mathbf{y}) \right) d\mathbf{y} \end{aligned}$$

subject to $\gamma(\mathbf{x}, \mathbf{y}) \geq 0$

- $\alpha(\mathbf{x})$ and $\beta(\mathbf{y})$ are Lagrange multiplier functions
- This looks strange because we are used to optimise vectors in Linear programming but here we optimise functions
- We can discretise the function and we would get a vector
- But functions form a vector space so we can define a linear programme for functions

* Dual Form

- We can rearrange the Lagrangian as

$$\mathcal{L} = \int \alpha(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} + \int \beta(\mathbf{y}) q(\mathbf{y}) d\mathbf{y} - \iint \gamma(\mathbf{x}, \mathbf{y}) (\alpha(\mathbf{x}) + \beta(\mathbf{y}) - d(\mathbf{x}, \mathbf{y})) d\mathbf{x} d\mathbf{y}$$

- Now we can interpret $\gamma(\mathbf{x}, \mathbf{y})$ as a Lagrange multiplier function so that the dual problem is

$$\max_{\alpha(\mathbf{x}), \beta(\mathbf{x})} \int \alpha(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} + \int \beta(\mathbf{y}) q(\mathbf{y}) d\mathbf{y}$$

subject to

$$\alpha(\mathbf{x}) + \beta(\mathbf{y}) \leq d(\mathbf{x}, \mathbf{y})$$

- note this is an inequality constraint because $\gamma(\mathbf{x}, \mathbf{y}) \geq 0$
- But this has to be true when $\mathbf{x} = \mathbf{y}$ so

$$\alpha(\mathbf{x}) + \beta(\mathbf{x}) \leq d(\mathbf{x}, \mathbf{x}) = 0$$

- Thus $\beta(\mathbf{x}) = -\alpha(\mathbf{x}) - \epsilon(\mathbf{x})$ where $\epsilon(\mathbf{x}) \geq 0$
- Our objective function becomes

$$\int \alpha(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} + \int \beta(\mathbf{y}) q(\mathbf{y}) d\mathbf{y} = \int \alpha(\mathbf{x}) (p(\mathbf{x}) - q(\mathbf{x})) d\mathbf{x} - \int q(\mathbf{x}) \epsilon(\mathbf{x}) d\mathbf{x}$$

- But this is clearly maximised when $\epsilon(\mathbf{x}) = 0$ therefore $\beta(\mathbf{x}) = -\alpha(\mathbf{x})$
- The problem simplifies to

$$\max_{\alpha(\mathbf{x})} \int \alpha(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} - \int \alpha(\mathbf{y}) q(\mathbf{y}) d\mathbf{y} = \max_{\alpha(\mathbf{x})} (\mathbb{E}_p[\alpha(\mathbf{x})] - \mathbb{E}_q[\alpha(\mathbf{x})])$$

subject to

$$\alpha(\mathbf{x}) - \alpha(\mathbf{y}) \leq d(\mathbf{x}, \mathbf{y})$$

- functions $\alpha(\mathbf{x})$ that satisfy this constraint are known as *Lipschitz-1 functions*
- An equivalent condition is that

$$\|\nabla_{\mathbf{x}} \alpha(\mathbf{x})\| \leq 1$$

- this is a continuity condition saying the output has to change slowly as we change the input

2.3 Wasserstein GANs

- In our Wasserstein GAN we train a generator to minimise the Wasserstein distance between the distribution of images from the generator and the true distribution
- We use mini-batches to approximate the expectations

$$\mathbb{E}_p[\alpha(\mathbf{x})] \approx \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \alpha(\mathbf{x}), \quad \mathbb{E}_q[\alpha(\mathbf{x})] \approx \frac{1}{n} \sum_{i=1}^n \alpha(G(\mathbf{z}_i, \mathbf{w}_G))$$

- We need to find the function $\alpha(\mathbf{x})$ that maximises the difference between these expectations
- We make $\alpha(\mathbf{x})$ a neural network called the *critic*
 - this plays the same role as the discriminator in a normal GAN
 - Again we make this a CNN
 - The difference is it has to be Lipschitz-1
 - This is difficult to achieve and is usually bodged (you can read the literature if you are interested)
- Wasserstein GANs claim to solve many of the problems of normal GANs
 - They are not perfect because they only approximate the Lipschitz-1
- They are for me one of the elegant solutions in machine learning of the last few years