Fisher GAN

Youssef Mroueh*, **Tom Sercu***
mroueh@us.ibm.com, tom.sercu1@ibm.com
∗ Equal Contribution
AI Foundations, IBM Research AI
IBM T.J Watson Research Center

Abstract

Generative Adversarial Networks (GANs) are powerful models for learning complex distributions. Stable training of GANs has been addressed in many recent works which explore different metrics between distributions. In this paper we introduce *Fisher GAN* which fits within the Integral Probability Metrics (IPM) framework for training GANs. Fisher GAN defines a critic with a data dependent constraint on its *second order moments*. We show in this paper that Fisher GAN allows for stable and time efficient training that does not compromise the capacity of the critic, and does not need data independent constraints such as weight clipping. We analyze our Fisher IPM theoretically and provide an algorithm based on Augmented Lagrangian for Fisher GAN. We validate our claims on both image sample generation and semi-supervised classification using Fisher GAN.

1 Introduction

Generative Adversarial Networks (GANs) [1] have recently become a prominent method to learn high-dimensional probability distributions. The basic framework consists of a generator neural network which learns to generate samples which approximate the distribution, while the discriminator measures the distance between the real data distribution, and this learned distribution that is referred to as *fake* distribution. The generator uses the gradients from the discriminator to minimize the distance with the real data distribution. The distance between these distributions was the object of study in [2], and highlighted the impact of the distance choice on the stability of the optimization. The original GAN formulation optimizes the Jensen-Shannon divergence, while later work generalized this to optimize f-divergences [3], KL [4], the Least Squares objective [5]. Closely related to our work, Wasserstein GAN (WGAN) [6] uses the earth mover distance, for which the discriminator function class needs to be constrained to be Lipschitz. To impose this Lipschitz constraint, WGAN proposes to use *weight clipping*, i.e. a data independent constraint, but this comes at the cost of reducing the capacity of the critic and high sensitivity to the choice of the clipping hyper-parameter. A recent development Improved Wasserstein GAN (WGAN-GP) [7] introduced a data dependent constraint namely a *gradient penalty* to enforce the Lipschitz constraint on the critic, which does not compromise the capacity of the critic but comes at a high computational cost.

weight clipping 不能保证严格的1阶Lipschitz约束

what is critic?

We build in this work on the Integral probability Metrics (IPM) framework for learning GAN of [8]. Intuitively the IPM defines a *critic* function f, that maximally discriminates between the real and fake distributions. We propose a theoretically sound and time efficient data dependent constraint on the critic of Wasserstein GAN, that allows a stable training of GAN and does not compromise the capacity of the critic. Where WGAN-GP uses a penalty on the gradients of the critic, *Fisher GAN* imposes a constraint on the *second order moments* of the critic. This extension to the IPM framework is inspired by the *Fisher Discriminant Analysis* method.

The main contributions of our paper are:

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1. We introduce in Section 2 the Fisher IPM, a scaling invariant distance between distributions. Fisher IPM introduces a data dependent constraint on the second order moments of the critic that discriminates between the two distributions. Such a constraint ensures the boundedness of the metric and the critic. We show in Section 2.2 that Fisher IPM when approximated with neural networks, corresponds to a discrepancy between *whitened* mean feature embeddings of the distributions. In other words a mean feature discrepancy that is measured with a Mahalanobis distance in the space computed by the neural network.

2. We show in Section 3 that Fisher IPM corresponds to the *Chi-squared* distance (χ_2) when the critic has unlimited capacity (the critic belongs to a universal hypothesis function class). Moreover we prove in Theorem 2 that even when the critic is parametrized by a neural network, it approximates the χ_2 distance with a factor which is a inner product between optimal and neural network critic. We finally derive generalization bounds of the learned critic from samples from the two distributions, assessing the statistical error and its convergence to the Chi-squared distance from finite sample size.

3. We use Fisher IPM as a GAN objective [1] and formulate an algorithm that combines desirable properties (Table 1): a stable and meaningful loss between distributions for GAN as in Wasserstein GAN [6], at a low computational cost similar to simple weight clipping, while not compromising the capacity of the critic via a data dependent constraint but at a much lower computational cost than [7]. Fisher GAN achieves strong semi-supervised learning results without need of batch normalization in the critic.

Table 1: Comparison between Fisher GAN and recent related approaches.

	Stability	Unconstrained capacity	Efficient Computation	Representation power (SSL)
Standard GAN [1, 9]	✗	✓	✓	✓
WGAN, McGan [6, 8]	✓	✗	✓	✗
WGAN-GP [7]	✓	✓	✗	?
Fisher Gan (Ours)	✓	✓	✓	✓

2 Learning GANs with Fisher IPM

2.1 Fisher IPM in an arbitrary function space: General framework

Integral Probability Metric (IPM). Intuitively an IPM defines a critic function f belonging to a function class $\mathscr{F}$, that maximally discriminates between two distributions. The function class $\mathscr{F}$ defines how f is bounded, which is crucial to define the metric. More formally, consider a compact space $\mathcal{X}$ in $\mathbb{R}^d$. Let $\mathscr{F}$ be a set of *measurable, symmetric and bounded* real valued functions on $\mathcal{X}$. Let $\mathscr{P}(\mathcal{X})$ be the set of measurable probability distributions on $\mathcal{X}$. Given two probability distributions $\mathbb{P}, \mathbb{Q} \in \mathscr{P}(\mathcal{X})$, the IPM indexed by a *symmetric* function space $\mathscr{F}$ is defined as follows [10]:

$$d_{\mathscr{F}}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathscr{F}} \Big\{ \mathop{\mathbb{E}}_{x \sim \mathbb{P}} f(x) - \mathop{\mathbb{E}}_{x \sim \mathbb{Q}} f(x) \Big\}. \tag{1}$$

It is easy to see that $d_{\mathscr{F}}$ defines a pseudo-metric over $\mathscr{P}(\mathcal{X})$. Note specifically that if $\mathscr{F}$ is not bounded, $\sup_f$ will scale f to be arbitrarily large. By choosing $\mathscr{F}$ appropriately [11], various distances between probability measures can be defined.

First formulation: Rayleigh Quotient. In order to define an IPM in the GAN context, [6, 8] impose the boundedness of the function space via a data independent constraint. This was achieved via restricting the norms of the weights parametrizing the function space to a ℓ_p ball. Imposing such a data independent constraint makes the training highly dependent on the constraint hyper-parameters and restricts the capacity of the learned network, limiting the usability of the learned critic in a semi-supervised learning task. Here we take a different angle and design the IPM to be scaling invariant as a *Rayleigh quotient*. Instead of measuring the discrepancy between means as in Equation (1), we measure a *standardized* discrepancy, so that the distance is **bounded by construction**. Standardizing this discrepancy introduces as we will see a *data dependent* constraint, that controls the growth of the weights of the critic f and ensures the stability of the training while maintaining the capacity of the critic. Given two distributions $\mathbb{P}, \mathbb{Q} \in \mathscr{P}(\mathcal{X})$ the Fisher IPM for a function space $\mathscr{F}$ is defined as follows:

$$d_{\mathscr{F}}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathscr{F}} \frac{\mathop{\mathbb{E}}_{x \sim \mathbb{P}}[f(x)] - \mathop{\mathbb{E}}_{x \sim \mathbb{Q}}[f(x)]}{\sqrt{1/2 \mathbb{E}_{x \sim \mathbb{P}} f^2(x) + 1/2 \mathbb{E}_{x \sim \mathbb{Q}} f^2(x)}}. \tag{2}$$

[1] Code is available at `https://github.com/tomsercu/FisherGAN`

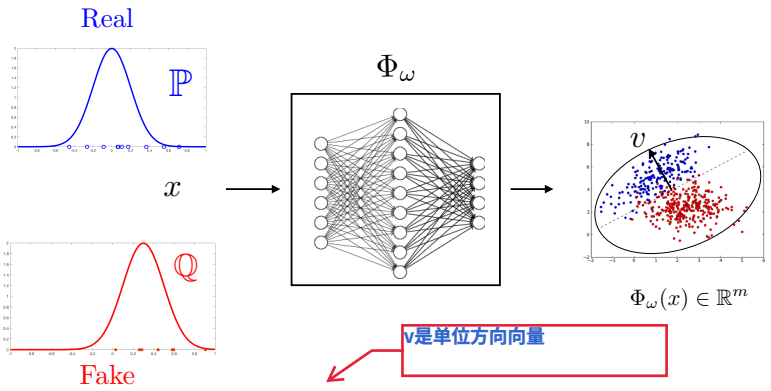

Figure 1: Illustration of Fisher IPM with Neural Networks. Φ_ω is a convolutional neural network which defines the embedding space. v is the direction in this embedding space with maximal mean separation $\langle v, \mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q})\rangle$, constrained by the hyperellipsoid $v^\top \Sigma_\omega(\mathbb{P};\mathbb{Q})\, v = 1$.

While a standard IPM (Equation (1)) maximizes the discrepancy between the means of a function under two different distributions, Fisher IPM looks for critic f that achieves a tradeoff between maximizing the discrepancy between the means under the two distributions (between *class* variance), and reducing the pooled second order moment (an upper bound on the intra-class variance).

Standardized discrepancies have a long history in statistics and the so-called *two-samples* hypothesis testing. For example the classic two samples Student's $t-$ test defines the student statistics as the ratio between means discrepancy and the sum of standard deviations. It is now well established that learning generative models has its roots in the two-samples hypothesis testing problem [12]. Non parametric two samples testing and model criticism from the kernel literature lead to the so called maximum kernel mean discrepancy (MMD) [13]. The MMD cost function and the mean matching IPM for a general function space has been recently used for training GAN [14, 15, 8].

Interestingly Harchaoui et al [16] proposed Kernel Fisher Discriminant Analysis for the two samples hypothesis testing problem, and showed its statistical consistency. The Standard Fisher discrepancy used in Linear Discriminant Analysis (LDA) or Kernel Fisher Discriminant Analysis (KFDA) can be written: $\sup_{f\in\mathscr{F}} \frac{\left(\mathbb{E}_{x\sim\mathbb{P}}[f(x)] - \mathbb{E}_{x\sim\mathbb{Q}}[f(x)]\right)^2}{\mathrm{Var}_{\mathrm{x}\sim\mathbb{P}}(\mathrm{f(x)}) + \mathrm{Var}_{\mathrm{x}\sim\mathbb{Q}}(\mathrm{f(x)})}$, where $\mathrm{Var}_{\mathrm{x}\sim\mathbb{P}}(\mathrm{f(x)}) = \mathbb{E}_{\mathrm{x}\sim\mathbb{P}}\mathrm{f}^2(\mathrm{x}) - (\mathbb{E}_{\mathrm{x}\sim\mathbb{P}}(\mathrm{f(x)}))^2$. Note that in LDA $\mathscr{F}$ is restricted to linear functions, in KFDA $\mathscr{F}$ is restricted to a Reproducing Kernel Hilbert Space (RKHS). Our Fisher IPM (Eq (2)) deviates from the standard Fisher discrepancy since the numerator is not squared, and we use in the denominator the second order moments instead of the variances. Moreover in our definition of Fisher IPM, $\mathscr{F}$ can be any symmetric function class.

Second formulation: Constrained form. Since the distance is scaling invariant, $d_{\mathscr{F}}$ can be written equivalently in the following constrained form:

伸缩不变性

$$d_{\mathscr{F}}(\mathbb{P},\mathbb{Q}) = \sup_{f\in\mathscr{F},\, \frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}} f^2(x) + \frac{1}{2}\mathbb{E}_{x\sim\mathbb{Q}} f^2(x) = 1} \mathscr{E}(f) := \mathop{\mathbb{E}}_{x\sim\mathbb{P}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Q}}[f(x)]. \tag{3}$$

Specifying $\mathbb{P}, \mathbb{Q}$: Learning GAN with Fisher IPM. We turn now to the problem of learning GAN with Fisher IPM. Given a distribution $\mathbb{P}_r \in \mathscr{P}(\mathcal{X})$, we learn a function $g_\theta : \mathcal{Z} \subset \mathbb{R}^{n_z} \to \mathcal{X}$, such that for $z \sim p_z$, the distribution of $g_\theta(z)$ is close to the real data distribution $\mathbb{P}_r$, where p_z is a fixed distribution on $\mathcal{Z}$ (for instance $z \sim \mathcal{N}(0, I_{n_z})$). Let $\mathbb{P}_\theta$ be the distribution of $g_\theta(z), z \sim p_z$. Using Fisher IPM (Equation (3)) indexed by a parametric function class $\mathscr{F}_p$, the generator minimizes the IPM: $\min_{g_\theta} d_{\mathscr{F}_p}(\mathbb{P}_r, \mathbb{P}_\theta)$. Given samples $\{x_i, 1 \dots N\}$ from $\mathbb{P}_r$ and samples $\{z_i, 1 \dots M\}$ from p_z we shall solve the following empirical problem:

$$\min_{g_\theta} \sup_{f_p\in\mathscr{F}_p} \hat{\mathscr{E}}(f_p, g_\theta) := \frac{1}{N}\sum_{i=1}^{N} f_p(x_i) - \frac{1}{M}\sum_{j=1}^{M} f_p(g_\theta(z_j)) \text{ Subject to } \hat{\Omega}(f_p, g_\theta) = 1, \tag{4}$$

where $\hat{\Omega}(f_p, g_\theta) = \frac{1}{2N}\sum_{i=1}^{N} f_p^2(x_i) + \frac{1}{2M}\sum_{j=1}^{M} f_p^2(g_\theta(z_j))$. For simplicity we will have $M = N$.

2.2 Fisher IPM with Neural Networks

We will specifically study the case where $\mathscr{F}$ is a finite dimensional Hilbert space induced by a neural network Φ_ω (see Figure 1 for an illustration). In this case, an IPM with data-independent constraint will be equivalent to mean matching [8]. We will now show that Fisher IPM will give rise to a *whitened* mean matching interpretation, or equivalently to *mean matching* with a Mahalanobis distance.

Rayleigh Quotient. Consider the function space $\mathscr{F}_{v,\omega}$, defined as follows

$$\mathscr{F}_{v,\omega} = \{f(x) = \langle v, \Phi_\omega(x)\rangle \,|v \in \mathbb{R}^m, \Phi_\omega : \mathcal{X} \to \mathbb{R}^m\},$$

Φ_ω is typically parametrized with a multi-layer neural network. We define the mean and covariance (Gramian) feature embedding of a distribution as in McGan [8]:

非线性 $$\mu_\omega(\mathbb{P}) = \mathop{\mathbb{E}}_{x\sim\mathbb{P}}(\Phi_\omega(x)) \quad \text{and} \quad \Sigma_\omega(\mathbb{P}) = \mathop{\mathbb{E}}_{x\sim\mathbb{P}}\left(\Phi_\omega(x)\Phi_\omega(x)^\top\right),$$

Fisher IPM as defined in Equation (2) on $\mathscr{F}_{v,\omega}$ can be written as follows:

$$d_{\mathscr{F}_{v,\omega}}(\mathbb{P},\mathbb{Q}) = \max_\omega \max_v \frac{\langle v, \mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q})\rangle}{\sqrt{v^\top(\frac{1}{2}\Sigma_\omega(\mathbb{P}) + \frac{1}{2}\Sigma_\omega(\mathbb{Q}) + \gamma I_m)v}}, \tag{5}$$

where we added a regularization term ($\gamma > 0$) to avoid singularity of the covariances. Note that if Φ_ω was implemented with homogeneous non linearities such as RELU, if we swap (v,ω) with $(cv, c'\omega)$ for any constants $c, c' > 0$, the distance $d_{\mathscr{F}_{v,\omega}}$ remains unchanged, hence the scaling invariance.

Constrained Form. Since the Rayleigh Quotient is not amenable to optimization, we will consider Fisher IPM as a constrained optimization problem. By virtue of the scaling invariance and the constrained form of the Fisher IPM given in Equation (3), $d_{\mathscr{F}_{v,\omega}}$ can be written equivalently as:

$$d_{\mathscr{F}_{v,\omega}}(\mathbb{P},\mathbb{Q}) = \max_{\omega, v, v^\top(\frac{1}{2}\Sigma_\omega(\mathbb{P})+\frac{1}{2}\Sigma_\omega(\mathbb{Q})+\gamma I_m)v=1} \langle v, \mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q})\rangle \tag{6}$$

Define the pooled covariance: $\Sigma_\omega(\mathbb{P};\mathbb{Q}) = \frac{1}{2}\Sigma_\omega(\mathbb{P}) + \frac{1}{2}\Sigma_\omega(\mathbb{Q}) + \gamma I_m$. Doing a simple change of variable $u = (\Sigma_\omega(\mathbb{P};\mathbb{Q}))^{\frac{1}{2}} v$ we see that:

u是任意方向的单位向量，最大的内积为原向量的模

$$\begin{aligned} d_{\mathscr{F}_{u,\omega}}(\mathbb{P},\mathbb{Q}) &= \max_\omega \max_{u,\|u\|=1} \left\langle u, (\Sigma_\omega(\mathbb{P};\mathbb{Q}))^{-\frac{1}{2}}(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))\right\rangle \\ &= \max_\omega \left\|(\Sigma_\omega(\mathbb{P};\mathbb{Q}))^{-\frac{1}{2}}(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))\right\|, \end{aligned} \tag{7}$$

hence we see that fisher IPM corresponds to the worst case distance between *whitened means*. Since the means are white, we don't need to impose further constraints on ω as in [6, 8]. Another interpretation of the Fisher IPM stems from the fact that:

$$d_{\mathscr{F}_{v,\omega}}(\mathbb{P},\mathbb{Q}) = \max_\omega \sqrt{(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))^\top \Sigma_\omega^{-1}(\mathbb{P};\mathbb{Q})(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))},$$

from which we see that Fisher IPM is a Mahalanobis distance between the mean feature embeddings of the distributions. The Mahalanobis distance is defined by the positive definite matrix $\Sigma_w(\mathbb{P};\mathbb{Q})$. We show in Appendix A that the gradient penalty in Improved Wasserstein [7] gives rise to a similar Mahalanobis mean matching interpretation.

Learning GAN with Fisher IPM. Hence we see that learning GAN with Fisher IPM:

$$\min_{g_\theta} \max_\omega \max_{v, v^\top(\frac{1}{2}\Sigma_\omega(\mathbb{P}_r)+\frac{1}{2}\Sigma_\omega(\mathbb{P}_\theta)+\gamma I_m)v=1} \langle v, \mu_w(\mathbb{P}_r) - \mu_\omega(\mathbb{P}_\theta)\rangle$$

corresponds to a min-max game between a feature space and a generator. The feature space tries to maximize the Mahalanobis distance between the feature means embeddings of *real* and *fake* distributions. The generator tries to minimize the mean embedding distance.

3 Theory

We will start first by studying the Fisher IPM defined in Equation (2) when the function space has *full capacity* i.e when the critic belongs to $\mathscr{L}_2(\mathcal{X}, \frac{1}{2}(\mathbb{P}+\mathbb{Q}))$ meaning that $\int_\mathcal{X} f^2(x)\frac{(\mathbb{P}(x)+\mathbb{Q}(x))}{2}dx < \infty$. Theorem 1 shows that under this condition, the Fisher IPM corresponds to the *Chi-squared distance* between distributions, and gives a closed form expression of the *optimal critic* function f_χ (See Appendix B for its relation with the Pearson Divergence). Proofs are given in Appendix D.

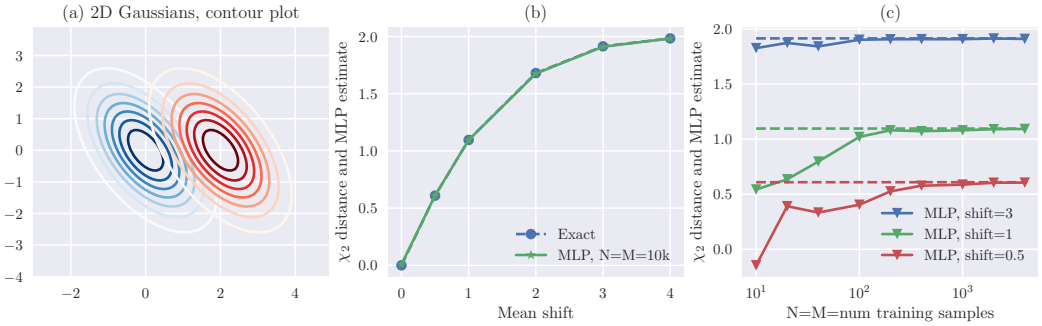

Figure 2: Example on 2D synthetic data, where both $\mathbb{P}$ and $\mathbb{Q}$ are fixed normal distributions with the same covariance and shifted means along the x-axis, see (a). Fig (b, c) show the exact χ_2 distance from numerically integrating Eq (8), together with the estimate obtained from training a 5-layer MLP with layer size = 16 and LeakyReLU nonlinearity on different training sample sizes. The MLP is trained using Algorithm 1, where sampling from the generator is replaced by sampling from $\mathbb{Q}$, and the χ_2 MLP estimate is computed with Equation (2) on a large number of samples (i.e. out of sample estimate). We see in (b) that for large enough sample size, the MLP estimate is extremely good. In (c) we see that for smaller sample sizes, the MLP approximation bounds the ground truth χ_2 from below (see Theorem 2) and converges to the ground truth roughly as $\mathcal{O}(\frac{1}{\sqrt{N}})$ (Theorem 3). We notice that when the distributions have small χ_2 distance, a larger training size is needed to get a better estimate - again this is in line with Theorem 3.

Theorem 1 (Chi-squared distance at full capacity)**.** *Consider the Fisher IPM for $\mathscr{F}$ being the space of all measurable functions endowed by $\frac{1}{2}(\mathbb{P}+\mathbb{Q})$, i.e. $\mathscr{F} := \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$. Define the Chi-squared distance between two distributions:*

$$\chi_2(\mathbb{P},\mathbb{Q}) = \sqrt{\int_{\mathcal{X}} \frac{(\mathbb{P}(x)-\mathbb{Q}(x))^2}{\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}} dx} \tag{8}$$

The following holds true for any $\mathbb{P}, \mathbb{Q}$, $\mathbb{P} \neq \mathbb{Q}$:

1) The Fisher IPM for $\mathscr{F} = \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$ is equal to the Chi-squared distance defined above: $d_{\mathscr{F}}(\mathbb{P},\mathbb{Q}) = \chi_2(\mathbb{P},\mathbb{Q})$.
2) The optimal critic of the Fisher IPM on $\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$ is :

$$f_\chi(x) = \frac{1}{\chi_2(\mathbb{P},\mathbb{Q})} \frac{\mathbb{P}(x)-\mathbb{Q}(x)}{\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}}.$$

We note here that LSGAN [5] at full capacity corresponds to a Chi-Squared divergence, with the main difference that LSGAN has different objectives for the generator and the discriminator (bilevel optimizaton), and hence does not optimize a single objective that is a distance between distributions. The Chi-squared divergence can also be achieved in the f-gan framework from [3]. We discuss the advantages of the Fisher formulation in Appendix C.

Optimizing over $\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$ is not tractable, hence we have to restrict our function class, to a hypothesis class $\mathscr{H}$, that enables tractable computations. Here are some typical choices of the space $\mathscr{H}$: Linear functions in the input features, RKHS, a non linear multilayer neural network with a linear last layer ($\mathscr{F}_{v,\omega}$). In this Section we don't make any assumptions about the function space and show in Theorem 2 how the Chi-squared distance is approximated in $\mathscr{H}$, and how this depends on the approximation error of the optimal critic f_χ in $\mathscr{H}$.

Theorem 2 (Approximating Chi-squared distance in an arbitrary function space $\mathscr{H}$)**.** *Let $\mathscr{H}$ be an arbitrary symmetric function space. We define the inner product $\langle f, f_\chi \rangle_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = \int_{\mathcal{X}} f(x) f_\chi(x) \frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2} dx$, which induces the Lebesgue norm. Let $\mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}$ be the unit sphere in $\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$: $\mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = \{f : \mathcal{X} \to \mathbb{R}, \|f\|_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = 1\}$. The fisher IPM defined on an arbitrary function space $\mathscr{H}$ $d_{\mathscr{H}}(\mathbb{P},\mathbb{Q})$, approximates the Chi-squared distance. The approximation*

quality depends on the cosine of the approximation of the optimal critic f_χ in $\mathscr{H}$. Since $\mathscr{H}$ is symmetric this cosine is always positive (otherwise the same equality holds with an absolute value)

$$d_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) = \chi_2(\mathbb{P},\mathbb{Q}) \sup_{f\in\mathscr{H}\cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}} \langle f, f_\chi\rangle_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})},$$

Equivalently we have following relative approximation error:

$$\frac{\chi_2(\mathbb{P},\mathbb{Q}) - d_{\mathscr{H}}(\mathbb{P},\mathbb{Q})}{\chi_2(\mathbb{P},\mathbb{Q})} = \frac{1}{2} \inf_{f\in\mathscr{H}\cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}} \|f - f_\chi\|^2_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}.$$

From Theorem 2, we know that we have always $d_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) \leq \chi_2(\mathbb{P},\mathbb{Q})$. Moreover if the space $\mathscr{H}$ was rich enough to provide a good approximation of the optimal critic f_χ, then $d_{\mathscr{H}}$ is a good approximation of the Chi-squared distance χ_2.

Generalization bounds for the sample quality of the estimated Fisher IPM from samples from $\mathbb{P}$ and $\mathbb{Q}$ can be done akin to [11], with the main difficulty that for Fisher IPM we have to bound the excess risk of a cost function with data dependent constraints on the function class. We give generalization bounds for learning the Fisher IPM in the supplementary material (**Theorem 3**, Appendix E). In a nutshell the generalization error of the critic learned in a hypothesis class $\mathscr{H}$ from samples of $\mathbb{P}$ and $\mathbb{Q}$, decomposes to the approximation error from Theorem 2 and a statistical error that is bounded using data dependent local Rademacher complexities [17] and scales like $O(\sqrt{1/n}), n = {}^{MN}/_{M+N}$. We illustrate in Figure 2 our main theoretical claims on a toy problem.

4 Fisher GAN Algorithm using ALM

For any choice of the parametric function class $\mathscr{F}_p$ (for example $\mathscr{F}_{v,\omega}$), note the constraint in Equation (4) by $\hat{\Omega}(f_p, g_\theta) = \frac{1}{2N}\sum_{i=1}^N f_p^2(x_i) + \frac{1}{2N}\sum_{j=1}^N f_p^2(g_\theta(z_j))$. Define the Augmented Lagrangian [18] corresponding to Fisher GAN objective and constraint given in Equation (4):

判别器critic 满足约束条件

$$\mathcal{L}_F(p,\theta,\lambda) = \hat{\mathscr{E}}(f_p, g_\theta) + \lambda(1 - \hat{\Omega}(f_p, g_\theta)) - \frac{\rho}{2}(\hat{\Omega}(f_p, g_\theta) - 1)^2 \tag{9}$$

where λ is the Lagrange multiplier and $\rho > 0$ is the quadratic penalty weight. We alternate between optimizing the critic and the generator. Similarly to [7] we impose the constraint when training the critic only. Given θ, for training the critic we solve $\max_p \min_\lambda \mathcal{L}_F(p,\theta,\lambda)$. Then given the critic parameters p we optimize the generator weights θ to minimize the objective $\min_\theta \hat{\mathscr{E}}(f_p, g_\theta)$. We give in Algorithm 1, an algorithm for Fisher GAN, note that we use ADAM [19] for optimizing the parameters of the critic and the generator. We use SGD for the Lagrange multiplier with learning rate ρ following practices in Augmented Lagrangian [18].

Algorithm 1 Fisher GAN

Input: ρ penalty weight, η Learning rate, n_c number of iterations for training the critic, N batch size
Initialize $p, \theta, \lambda = 0$
repeat
  for $j = 1$ **to** n_c **do**
    Sample a minibatch $x_i, i = 1 \dots N, x_i \sim \mathbb{P}_r$
    Sample a minibatch $z_i, i = 1 \dots N, z_i \sim p_z$
    $(g_p, g_\lambda) \leftarrow (\nabla_p \mathcal{L}_F, \nabla_\lambda \mathcal{L}_F)(p,\theta,\lambda)$
    $p \leftarrow p + \eta$ ADAM (p, g_p)
    $\lambda \leftarrow \lambda - \rho g_\lambda$ {SGD rule on λ with learning rate ρ}
  end for
  Sample $z_i, i = 1 \dots N, z_i \sim p_z$
  $d_\theta \leftarrow \nabla_\theta \hat{\mathscr{E}}(f_p, g_\theta) = -\nabla_\theta \frac{1}{N}\sum_{i=1}^N f_p(g_\theta(z_i))$
  $\theta \leftarrow \theta - \eta$ ADAM (θ, d_θ)
until θ converges

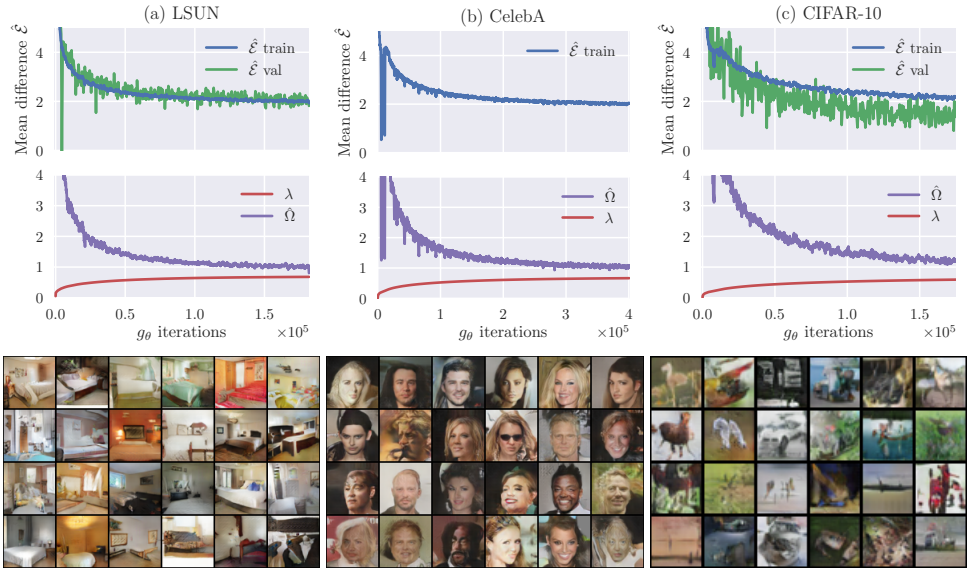

Figure 3: Samples and plots of the loss $\hat{\mathcal{E}}(.)$, lagrange multiplier λ, and constraint $\hat{\Omega}(.)$ on 3 benchmark datasets. We see that during training as λ grows slowly, the constraint becomes tight.

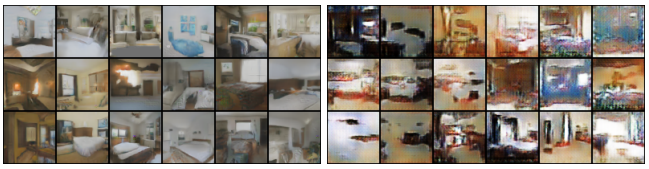

Figure 4: **No Batch Norm:** Training results from a critic f without batch normalization. Fisher GAN (left) produces decent samples, while WGAN with weight clipping (right) does not. We hypothesize that this is due to the implicit whitening that Fisher GAN provides. (Note that WGAN-GP does also succesfully converge without BN [7]). For both models the learning rate was appropriately reduced.

5 Experiments

We experimentally validate the proposed Fisher GAN. We claim three main results: (1) stable training with a meaningful and stable loss going down as training progresses and correlating with sample quality, similar to [6, 7]. (2) very fast convergence to good sample quality as measured by inception score. (3) competitive semi-supervised learning performance, on par with literature baselines, without requiring normalization of the critic.

We report results on three benchmark datasets: CIFAR-10 [20], LSUN [21] and CelebA [22]. We parametrize the generator g_θ and critic f with convolutional neural networks following the model design from DCGAN [23]. For 64×64 images (LSUN, CelebA) we use the model architecture in Appendix F.2, for CIFAR-10 we train at a 32×32 resolution using architecture in F.3 for experiments regarding sample quality (inception score), while for semi-supervised learning we use a better regularized discriminator similar to the Openai [9] and ALI [24] architectures, as given in F.4.We used Adam [19] as optimizer for all our experiments, hyper-parameters given in Appendix F.

Qualitative: Loss stability and sample quality. Figure 3 shows samples and plots during training. For LSUN we use a higher number of D updates ($n_c = 5$) , since we see similarly to WGAN that the loss shows large fluctuations with lower n_c values. For CIFAR-10 and CelebA we use reduced $n_c = 2$ with no negative impact on loss stability. CIFAR-10 here was trained without any label information. We show both train and validation loss on LSUN and CIFAR-10 showing, as can be expected, no overfitting on the large LSUN dataset and some overfitting on the small CIFAR-10 dataset. To back up our claim that Fisher GAN provides stable training, we trained both a Fisher Gan and WGAN where the batch normalization in the critic f was removed (Figure 4).

Quantitative analysis: Inception Score and Speed. It is agreed upon that evaluating generative models is hard [25]. We follow the literature in using "inception score" [9] as a metric for the quality

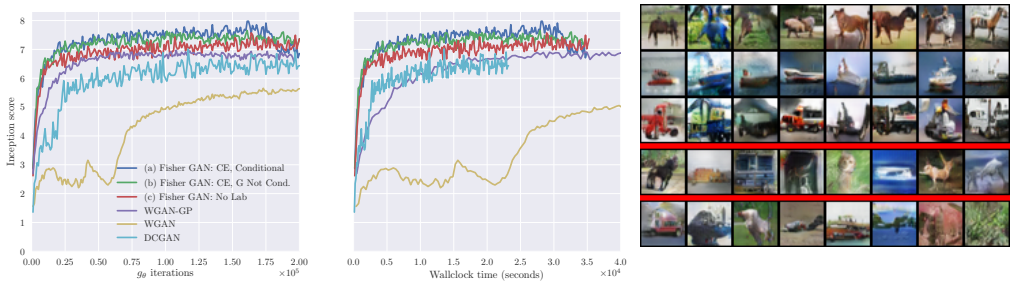

Figure 5: CIFAR-10 inception scores under 3 training conditions. Corresponding samples are given in rows from top to bottom (a,b,c). The inception score plots are mirroring Figure 3 from [7].
Note In v1 of this paper, the baseline inception scores were underestimated because they were computed using too few samples.
Note All inception scores are computed from the same tensorflow codebase, using the architecture described in appendix F.3, and with weight initialization from a normal distribution with stdev=0.02. In Appendix F.1 we show that these choices are also benefiting our WGAN-GP baseline.

of CIFAR-10 samples. Figure 5 shows the inception score as a function of number of g_θ updates and wallclock time. All timings are obtained by running on a single K40 GPU on the same cluster. We see from Figure 5, that Fisher GAN both produces better inception scores, and has a clear speed advantage over WGAN-GP.

Quantitative analysis: SSL. One of the main premises of unsupervised learning, is to learn features on a large corpus of unlabeled data in an unsupervised fashion, which are then transferable to other tasks. This provides a proper framework to measure the performance of our algorithm. This leads us to quantify the performance of Fisher GAN by semi-supervised learning (SSL) experiments on CIFAR-10. We do joint supervised and unsupervised training on CIFAR-10, by adding a cross-entropy term to the IPM objective, in conditional and unconditional generation.

Table 2: CIFAR-10 inception scores using resnet architecture and codebase from [7]. We used Layer Normalization [26] which outperformed unnormalized resnets. Apart from this, no additional hyperparameter tuning was done to get stable training of the resnets.

Method	Score	Method	Score
ALI [24]	$5.34 \pm .05$	SteinGan [31]	6.35
BEGAN [27]	5.62	DCGAN (with labels, in [31])	6.58
DCGAN [23] (in [28])	$6.16 \pm .07$	Improved GAN [9]	$8.09 \pm .07$
Improved GAN (-L+HA) [9]	$6.86 \pm .06$	*Fisher GAN ResNet (ours)*	$8.16 \pm .12$
EGAN-Ent-VI [29]	$7.07 \pm .10$	AC-GAN [32]	$8.25 \pm .07$
DFM [30]	$7.72 \pm .13$	SGAN-no-joint [28]	$8.37 \pm .08$
WGAN-GP ResNet [7]	$7.86 \pm .07$	WGAN-GP ResNet [7]	$8.42 \pm .10$
Fisher GAN ResNet (ours)	$7.90 \pm .05$	**SGAN [28]**	$8.59 \pm .12$
Unsupervised		Supervised	

Unconditional Generation with CE Regularization. We parametrize the critic f as in $\mathscr{F}_{v,\omega}$. While training the critic using the Fisher GAN objective $\mathcal{L}_F$ given in Equation (9), we train a linear classifier on the feature space Φ_ω of the critic, whenever labels are available (K labels). The linear classifier is trained with Cross-Entropy (CE) minimization. Then the critic loss becomes $\mathcal{L}_D = \mathcal{L}_F - \lambda_D \sum_{(x,y)\in \text{lab}} CE(x, y; S, \Phi_\omega)$, where $CE(x, y; S, \Phi_\omega) = -\log\left[\text{Softmax}(\langle \text{S}, \Phi_\omega(\text{x})\rangle)_\text{y}\right]$, where $S \in \mathbb{R}^{K\times m}$ is the linear classifier and $\langle S, \Phi_\omega\rangle \in \mathbb{R}^K$ with slight abuse of notation. λ_D is the regularization hyper-parameter. We now sample three minibatches for each critic update: one labeled batch from the small labeled dataset for the CE term, and an unlabeled batch + generated batch for the IPM.

Conditional Generation with CE Regularization. We also trained *conditional generator* models, conditioning the generator on y by concatenating the input noise with a 1-of-K embedding of the label: we now have $g_\theta(z, y)$. We parametrize the critic in $\mathscr{F}_{v,\omega}$ and modify the critic objective as above. We also add a cross-entropy term for the generator to minimize during its training step:

$\mathcal{L}_G = \hat{\mathscr{E}} + \lambda_G \sum_{z\sim p(z), y\sim p(y)} CE(g_\theta(z,y), y; S, \Phi_\omega)$. For generator updates we still need to sample only a single minibatch since we use the minibatch of samples from $g_\theta(z,y)$ to compute both the IPM loss $\hat{\mathscr{E}}$ and CE. The labels are sampled according to the prior $y \sim p(y)$, which defaults to the discrete uniform prior when there is no class imbalance. We found $\lambda_D = \lambda_G = 0.1$ to be optimal.

New Parametrization of the Critic: "$K+1$ SSL". One specific successful formulation of SSL in the standard GAN framework was provided in [9], where the discriminator classifies samples into $K+1$ categories: the K correct clases, and $K+1$ for fake samples. Intuitively this puts the real classes in competition with the fake class. In order to implement this idea in the Fisher framework, we define a new function class of the critic that puts in competition the K class directions of the classifier S_y, and another "K+1" direction v that indicates fake samples. Hence we propose the following parametrization for the critic: $f(x) = \sum_{y=1}^{K} p(y|x) \langle S_y, \Phi_\omega(x)\rangle - \langle v, \Phi_\omega(x)\rangle$, where $p(y|x) = \mathrm{Softmax}(\langle \mathrm{S}, \Phi_\omega(\mathrm{x})\rangle)_\mathrm{y}$ which is also optimized with Cross-Entropy. Note that this critic does not fall under the interpretation with whitened means from Section 2.2, but does fall under the general Fisher IPM framework from Section 2.1. We can use this critic with both conditional and unconditional generation in the same way as described above. In this setting we found $\lambda_D = 1.5$, $\lambda_G = 0.1$ to be optimal.

Layerwise normalization on the critic. For most GAN formulations following DCGAN design principles, batch normalization (BN) [33] in the critic is an essential ingredient. From our semi-supervised learning experiments however, it appears that batch normalization gives substantially worse performance than layer normalization (LN) [26] or even no layerwise normalization. We attribute this to the implicit whitening Fisher GAN provides.

Table 3 shows the SSL results on CIFAR-10. We show that Fisher GAN has competitive results, on par with state of the art literature baselines. When comparing to WGAN with weight clipping, it becomes clear that we recover the lost SSL performance. Results with the $K+1$ critic are better across the board, proving consistently the advantage of our proposed $K+1$ formulation. Conditional generation does not provide gains in the setting with layer normalization or without normalization.

Table 3: CIFAR-10 SSL results.
Note In v3, strong results were added using LN and no normalization.

Number of labeled examples	1000	2000	4000	8000
Model			Misclassification rate	
CatGAN [34]			19.58	
Improved GAN (FM) [9]	21.83 ± 2.01	19.61 ± 2.09	18.63 ± 2.32	17.72 ± 1.82
ALI [24]	19.98 ± 0.89	19.09 ± 0.44	17.99 ± 1.62	17.05 ± 1.49
WGAN (weight clipping) Uncond	69.01	56.48	40.85	30.56
WGAN (weight clipping) Cond	68.11	58.59	42.00	30.91
Fisher GAN BN Cond	36.37	32.03	27.42	22.85
Fisher GAN BN Uncond	36.42	33.49	27.36	22.82
Fisher GAN BN K+1 Cond	34.94	28.04	23.85	20.75
Fisher GAN BN K+1 Uncond	33.49	28.60	24.19	21.59
Fisher GAN LN Cond	26.78 ± 1.04	23.30 ± 0.39	20.56 ± 0.64	18.26 ± 0.25
Fisher GAN LN Uncond	24.39 ± 1.22	22.69 ± 1.27	19.53 ± 0.34	17.84 ± 0.15
Fisher GAN LN K+1 Cond	20.99 ± 0.66	19.01 ± 0.21	17.41 ± 0.38	15.50 ± 0.41
Fisher GAN LN K+1, Uncond	19.74 ± 0.21	17.87 ± 0.38	16.13 ± 0.53	14.81 ± 0.16
Fisher GAN No Norm K+1, Uncond	21.15 ± 0.54	18.21 ± 0.30	16.74 ± 0.19	14.80 ± 0.15

6 Conclusion

We have defined Fisher GAN, which provide a stable and fast way of training GANs. The Fisher GAN is based on a scale invariant IPM, by constraining the second order moments of the critic. We provide an interpretation as whitened (Mahalanobis) mean feature matching and χ_2 distance. We show graceful theoretical and empirical advantages of our proposed Fisher GAN.

Acknowledgments. The authors thank Steven J. Rennie for many helpful discussions and Martin Arjovsky for helpful clarifications and pointers.

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 *NIPS*, 2014.

[2] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

[3] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.

[4] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *ICLR*, 2017.

[5] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang. Least squares generative adversarial networks. *arXiv:1611.04076*, 2016.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ICML*, 2017.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv:1704.00028*, 2017.

[8] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. *arXiv:1702.08398 ICML*, 2017.

[9] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *NIPS*, 2016.

[10] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 1997.

[11] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 2012.

[12] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv:1610.03483*, 2016.

[13] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 2012.

[14] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. In *ICML*, 2015.

[15] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *UAI*, 2015.

[16] Zaïd Harchaoui, Francis R Bach, and Eric Moulines. Testing for homogeneity with kernel fisher discriminant analysis. In *NIPS*, 2008.

[17] Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. *Ann. Statist.*, 2005.

[18] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

[21] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv:1506.03365*, 2015.

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[24] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *ICLR*, 2017.

[25] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *ICLR*, 2016.

[26] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[27] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[28] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016.

[29] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017.

[30] D Warde-Farley and Y Bengio. Improving generative adversarial networks with denoising feature matching. *ICLR submissions*, 8, 2017.

[31] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

[32] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proc. ICML*, 2015.

[34] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv:1511.06390*, 2015.

[35] Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D. Lawrence. Metrics for probabilistic geometries. 2014.

[36] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Scholkopf, and Gert R. G. Lanckriet. On integral probability metrics, ϕ-divergences and binary classification. 2009.

[37] I. Ekeland and T. Turnbull. *Infinite-dimensional Optimization and Convexity*. The University of Chicago Press, 1983.

[38] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

Supplementary Material for Fisher GAN

Youssef Mroueh*, Tom Sercu*
IBM Research AI

A WGAN-GP versus Fisher GAN

Consider

$$\mathscr{F}_{v,\omega} = \{f(x) = \langle v, \Phi_w(x)\rangle, v \in \mathbb{R}^m, \Phi_\omega : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}^m\}$$

Let

$$J_{\Phi_\omega}(x) \in \mathbb{R}^{m\times d}, [J_{\Phi_\omega}(x)]_{i,j} = \frac{\partial \langle e_i, \Phi_\omega(x)\rangle}{\partial x_j}$$

be the Jacobian matrix of the $\Phi_\omega(.)$. It is easy to see that

$$\nabla_x f(x) = J^\top_{\Phi_\omega}(x)v \in \mathbb{R}^d,$$

and therefore

$$\|\nabla_x f(x)\|^2 = \left\langle v, J_{\Phi_\omega}(x)J^\top_{\Phi_\omega}(x)v\right\rangle,$$

Note that ,

$$J_{\Phi_\omega}(x)J^\top_{\Phi_\omega}(x)$$

is the so called *metric tensor* in information geometry (See for instance [35] and references there in). The gradient penalty for WGAN of [7] can be derived from a Rayleigh quotient principle as well, written in the constraint form:

$$d_{\mathscr{F}_{v,\omega}}(\mathbb{P},\mathbb{Q}) = \sup_{f\in\mathscr{F}_{v,\omega}, \mathbb{E}_{u\sim U[0,1]}\mathbb{E}_{x\sim u\mathbb{P}+(1-u)\mathbb{Q}}\|\nabla_x f(x)\|^2=1} \mathbb{E}_{x\sim\mathbb{P}}f(x) - \mathbb{E}_{x\sim\mathbb{Q}}f(x)$$

Using the special parametrization we can write:

$$\mathbb{E}_{u\sim U[0,1]}\mathbb{E}_{x\sim u\mathbb{P}+(1-u)\mathbb{Q}}\|\nabla_x f(x)\|^2 = v^\top\left(\mathbb{E}_{u\sim U[0,1]}\mathbb{E}_{x\sim u\mathbb{P}+(1-u)\mathbb{Q}}J_{\Phi_\omega}(x)J^\top_{\Phi_\omega}(x)\right)v$$

Let

$$\mathscr{M}_\omega(\mathbb{P};\mathbb{Q}) = \mathbb{E}_{u\sim U[0,1]}\mathbb{E}_{x\sim u\mathbb{P}+(1-u)\mathbb{Q}}J_{\Phi_\omega}(x)J^\top_{\Phi_\omega}(x) \in \mathbb{R}^{m\times m}$$

is the expected Riemannian metric tensor [35]. Hence we obtain:

$$\begin{aligned} d_{\mathscr{F}_{v,\omega}}(\mathbb{P},\mathbb{Q}) &= \max_w \max_{v, v^\top\mathscr{M}_\omega(\mathbb{P};\mathbb{Q})v=1} \langle v, \mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q})\rangle \\ &= \max_\omega \left\|\mathscr{M}_\omega^{-\frac{1}{2}}(\mathbb{P};\mathbb{Q})(\mu_\omega(\mathbb{P})) - \mu_\omega(\mathbb{Q})\right\| \end{aligned}$$

Hence Gradient penalty can be seen as well as mean matching in the metric defined by the expected metric tensor $\mathscr{M}_\omega$.

Improved WGAN [7] IPM can be written as follows :

$$\max_\omega \sqrt{(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))^\top \mathscr{M}_\omega^{-1}(\mathbb{P};\mathbb{Q})(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))}$$

to be contrasted with Fisher IPM:

$$\max_\omega \sqrt{(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))^\top \Sigma_\omega^{-1}(\mathbb{P};\mathbb{Q})(\mu_\omega(\mathbb{P}) - \mu_\omega(\mathbb{Q}))}$$

Both Improved WGAN are doing mean matching using different Mahalanobis distances! While improved WGAN uses an *expected metric tensor* $\mathscr{M}_\omega$ to compute this distance, Fisher IPM uses a simple pooled covariance Σ_ω to compute this metric. It is clear that Fisher GAN has a computational advantage!

B Chi-squared distance and Pearson Divergence

The definition of χ_2 distance:

$$\chi_2^2(\mathbb{P},\mathbb{Q}) = 2\int_{\mathcal{X}} \frac{(\mathbb{P}(x)-\mathbb{Q}(x))^2}{\mathbb{P}(x)+\mathbb{Q}(x)}dx.$$

The χ_2 Pearson divergence:

$$\chi_2^P(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} \frac{(\mathbb{P}(x)-\mathbb{Q}(x))^2}{\mathbb{Q}(x)}dx.$$

We have the following relation:

$$\chi_2^2(\mathbb{P},\mathbb{Q}) = \frac{1}{4}\chi_2^P\left(\mathbb{P}, \frac{\mathbb{P}+\mathbb{Q}}{2}\right).$$

C Fisher GAN and φ-divergence Based GANs

Since f-gan [3] also introduces a GAN formulation which recovers the Chi-squared divergence, we compare our approaches.

Let us recall here the definition of φ-divergence:

$$d_\varphi(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} \varphi\left(\frac{\mathbb{P}(x)}{\mathbb{Q}(x)}\right)\mathbb{Q}(x)dx,$$

where $\varphi : \mathbb{R}^+ \to \mathbb{R}$ is a convex, lower-semicontinuous function satisfying $\varphi(1) = 0$. Let φ^* the Fenchel conjugate of φ:

$$\varphi^*(t) = \sup_{u \in Dom_\varphi} ut - \varphi(u)$$

As shown in [3] and in [36], for any function space $\mathscr{F}$ we get the lower bound:

$$d_\varphi(\mathbb{P},\mathbb{Q}) \geq \sup_{f\in\mathscr{F}} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}}\varphi^*(f(x)),$$

For the particular case $\varphi(t) = (t-1)^2$ and $\varphi^*(t) = \frac{1}{4}t^2 + t$ we have the Pearson χ_2 divergence:

$$d_\varphi(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} \frac{(\mathbb{P}(x)-\mathbb{Q}(x))^2}{\mathbb{Q}(x)}dx = \chi_2^P(\mathbb{P},\mathbb{Q})$$

Hence to optimize the same cost function of Fisher GAN in the φ-GAN framework we have to consider:

$$\frac{1}{2}\sqrt{\chi_2^P\left(\mathbb{P}, \frac{\mathbb{P}+\mathbb{Q}}{2}\right)},$$

Fisher GAN gives an inequality for the symmetric Chi-squared and the φ-GAN gives a lower variational bound. i.e compare for φ-GAN:

$$\begin{aligned}\sup_{f\in\mathscr{F}} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\frac{\mathbb{P}+\mathbb{Q}}{2}}\varphi^*(f(x)) &= \sup_{f\in\mathscr{F}} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\frac{\mathbb{P}+\mathbb{Q}}{2}}\left(\frac{1}{4}f^2(x) + f(x)\right) \\ &= \sup_{f\in\mathscr{F}} \frac{1}{2}\left(\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x)\right) - \frac{1}{4}\mathbb{E}_{x\sim\frac{\mathbb{P}+\mathbb{Q}}{2}} f^2(x) \quad (10)\end{aligned}$$

and for Fisher GAN:

$$\sup_{f\in\mathscr{F}, \mathbb{E}_{x\sim\frac{\mathbb{P}+\mathbb{Q}}{2}} f^2(x)=1} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}}(f(x)) \quad (11)$$

while equivalent at the optimum those two formulations for the symmetric Chi-squared given in Equations (10), and (11) have different theoretical and practical properties. On the theory side:

1. While the formulation in (10) is a φ divergence, the formulation given by the Fisher criterium in (11) is an IPM with a data dependent constraint. This is a surprising result because φ-divergences and IPM exhibit different properties and the only known non trivial φ divergence that is also an IPM with data independent function class is the total variation distance [36]. When we allow the function class to be dependent on the distributions, the symmetric Chi-squared divergence (in fact general Chi-squared also) can be cast as an IPM! Hence in the context of GAN training we inherit the known stability of IPM based training for GANs.
2. Theorem 2 for the Fisher criterium gives us an approximation error when we change the function from the space of measurable functions to a hypothesis class. It is not clear how tight the lower bound in the φ-divergence will be as we relax the function class.

On the practical side:

1. Once we parametrize the critic f as a neural network with linear output activation, i.e. $f(x) = \langle v, \Phi_\omega(x)\rangle$, we see that the optimization is unconstrained for the φ-divergence formulation (10) and the weights updates can explode and have an unstable behavior. On the other hand in the Fisher formulation (11) the data dependent constraint that is imposed slowly through the lagrange multiplier, enforces a variance control that prevents the critic from blowing up and causing instabilities in the training. Note that in the Fisher case we have three players: the critic, the generator and the lagrange multiplier. The lagrange multiplier grows slowly to enforce the constraint and to approach the Chi-squared distance as training converges. Note that the φ-divergence formulation (10) can be seen as a Fisher GAN with fixed lagrange multiplier $\lambda = \frac{1}{2}$ that is indeed unstable in theory and in our experiments.

Remark 1. *Note that if the Neyman divergence is of interest, it can also be obtained as the following Fisher criterium:*

$$\sup_{f\in\mathscr{F}, \mathbb{E}_{x\sim\mathbb{P}} f^2(x)=1} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}}(f(x)), \tag{12}$$

this is equivalent at the optimum to:

$$\chi^2_N(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} \frac{(\mathbb{P}(x)-\mathbb{Q}(x))^2}{\mathbb{P}(x)} dx.$$

Using a neural network $f(x) = \langle v, \Phi_\omega(x)\rangle$, the Neyman divergence can be achieved with linear output activation and a data dependent constraint:

$$\sup_{v,\omega,\ \mathbb{E}_{x\sim\mathbb{P}}(\langle v,\Phi_\omega(x)\rangle)^2=1} \langle v, \mathbb{E}_{x\sim\mathbb{P}}\Phi_\omega(x) - \mathbb{E}_{x\sim\mathbb{Q}}\Phi_\omega(x)\rangle$$

To obtain the same divergence as a φ-divergence we need $\varphi(u) = \frac{(1-u)^2}{u}$, and $\varphi^*(u) = 2 - 2\sqrt{1-u},\ (u<1)$. Moreover exponential activation functions are used in [3], which most likely renders this formulation also unstable for GAN training.

D Proofs

Proof of Theorem 1. Consider the space of measurable functions,

$$\mathscr{F} = \left\{ f : \mathcal{X}\to\mathbb{R},\ f \text{ measurable such that } \int_{\mathcal{X}} f^2(x)\frac{(\mathbb{P}(x)+\mathbb{Q}(x))}{2}dx < \infty \right\}$$

meaning that $f \in \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$.

$$\begin{aligned}
d_{\mathscr{F}}(\mathbb{P},\mathbb{Q}) &= \sup_{f\in\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2}), f\neq 0} \frac{\mathop{\mathbb{E}}_{x\sim\mathbb{P}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Q}}[f(x)]}{\sqrt{\frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}} f^2(x) + \frac{1}{2}\mathbb{E}_{x\sim\mathbb{Q}} f^2(x)}} \\
&= \sup_{f\in\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2}), \|f\|_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}=1} \mathop{\mathbb{E}}_{x\sim\mathbb{P}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Q}}[f(x)] \\
&= \sup_{f\in\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2}), \|f\|_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}\leq 1} \mathop{\mathbb{E}}_{x\sim\mathbb{P}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Q}}[f(x)] \text{ (By convexity of the cost functional in } f) \\
&= \sup_{f\in\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})} \inf_{\lambda\geq 0} \mathcal{L}(f,\lambda),
\end{aligned}$$

where in the last equation we wrote the lagrangian of the Fisher IPM for this particular function class $\mathscr{F} := \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$:

$$\mathcal{L}(f, \lambda) = \int_{\mathcal{X}} f(x)(\mathbb{P}(x) - \mathbb{Q}(x))dx + \frac{\lambda}{2}\left(1 - \frac{1}{2}\int_{\mathcal{X}} f^2(x)(\mathbb{P}(x) + \mathbb{Q}(x))dx\right),$$

By convexity of the functional cost and constraints, and since $f \in \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$, we can minimize the inner loss to optimize this functional for each $x \in \mathcal{X}$ [37]. The first order conditions of optimality (KKT conditions) gives us for the optimum f_χ, λ_*:

$$(\mathbb{P}(x) - \mathbb{Q}(x)) - \frac{\lambda_*}{2} f_\chi(x)(\mathbb{P}(x) + \mathbb{Q}(x))) = 0,$$

$$f_\chi(x) = \frac{2}{\lambda_*} \frac{\mathbb{P}(x) - \mathbb{Q}(x)}{\mathbb{P}(x) + \mathbb{Q}(x)}.$$

Using the feasibility constraint: $\int_{\mathcal{X}} f_\chi^2(x) \left(\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}\right) = 1$, we get :

$$\int_{\mathcal{X}} \frac{4}{\lambda_*^2} \frac{(\mathbb{P}(x) - \mathbb{Q}(x))^2}{(\mathbb{P}(x) + \mathbb{Q}(x))^2} \left(\frac{\mathbb{P}(x) + \mathbb{Q}(x)}{2}\right) = 1,$$

which gives us the expression of λ_*:

$$\lambda_* = \sqrt{\int_{\mathcal{X}} \frac{(\mathbb{P}(x) - \mathbb{Q}(x))^2}{\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}} dx}.$$

Hence for $\mathscr{F} := \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$ we have:

$$d_{\mathscr{F}}(\mathbb{P}, \mathbb{Q}) = \int_{\mathcal{X}} f_\chi(x)(\mathbb{P}(x) - \mathbb{Q}(x))dx = \sqrt{\int_{\mathcal{X}} \frac{(\mathbb{P}(x) - \mathbb{Q}(x))^2}{\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}} dx} = \lambda_*$$

Define the following distance between two distributions:

$$\chi_2(\mathbb{P}, \mathbb{Q}) = \left\| \frac{d\mathbb{P}}{\frac{d\mathbb{P}+d\mathbb{Q}}{2}} - \frac{d\mathbb{Q}}{\frac{d\mathbb{P}+d\mathbb{Q}}{2}} \right\|_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})},$$

We refer to this distance as the χ_2 distance between two distributions. It is easy to see that :

$$d_{\mathscr{F}}(\mathbb{P}, \mathbb{Q}) = \chi_2(\mathbb{P}, \mathbb{Q})$$

and the optimal critic f_χ has the following expression:

$$f_\chi(x) = \frac{1}{\chi_2(\mathbb{P}, \mathbb{Q})} \frac{\mathbb{P}(x) - \mathbb{Q}(x)}{\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}}.$$

□

Proof of Theorem 2. Define the means difference functional $\mathscr{E}$:

$$\mathscr{E}(f; \mathbb{P}, \mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x)$$

Let

$$\mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = \{f : \mathcal{X} \to \mathbb{R}, \|f\|_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = 1\}$$

For a symmetric function class $\mathscr{H}$, the Fisher IPM has the following expression:

$$\begin{aligned} d_{\mathscr{H}}(\mathbb{P}, \mathbb{Q}) &= \sup_{f\in\mathscr{H},\ \|f\|_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}=1} \mathscr{E}(f; \mathbb{P}, \mathbb{Q}) \\ &= \sup_{f\in\mathscr{H}\cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}} \mathscr{E}(f; \mathbb{P}, \mathbb{Q}). \end{aligned}$$

Recall that for $\mathscr{H} = \mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$, the optimum $\chi_2(\mathbb{P}, \mathbb{Q})$ is achieved for :

$$f_\chi(x) = \frac{1}{\chi_2(\mathbb{P}, \mathbb{Q})} \frac{\mathbb{P}(x) - \mathbb{Q}(x)}{\frac{\mathbb{P}(x)+\mathbb{Q}(x)}{2}}, \forall x \in \mathcal{X} \text{ a.s.}$$

Let $f \in \mathscr{H}$ such that $\|f\|_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = 1$ we have the following:

$$\begin{aligned} \langle f, f_\chi \rangle_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} &= \int_{\mathcal{X}} f(x) f_\chi(x) \frac{(\mathbb{P}(x) + \mathbb{Q}(x))}{2} dx \\ &= \frac{1}{\chi_2(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X}} f(x)(\mathbb{P}(x) - \mathbb{Q}(x)) dx \\ &= \frac{\mathscr{E}(f; \mathbb{P}, \mathbb{Q})}{\chi_2(\mathbb{P}, \mathbb{Q})}. \end{aligned}$$

It follows that for any $f \in \mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}$ we have:

$$\mathscr{E}(f; \mathbb{P}, \mathbb{Q}) = \chi_2(\mathbb{P}, \mathbb{Q}) \langle f, f_\chi \rangle_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} \tag{13}$$

In particular taking the sup over $\mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}$ we have:

$$d_{\mathscr{H}}(\mathbb{P}, \mathbb{Q}) = \chi_2(\mathbb{P}, \mathbb{Q}) \sup_{f \in \mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}} \langle f, f_\chi \rangle_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}. \tag{14}$$

note that since $\mathscr{H}$ is symmetric all quantities are positive after taking the sup (if $\mathscr{H}$ was not symmetric one can take the absolute values, and similar results hold with absolute values.)

If $\mathscr{H}$ is rich enough so that we find, for $\varepsilon \in (0, 1)$, a $1 - \varepsilon$ approximation of f_χ in $\mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}$, i.e:

$$\sup_{f \in \mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}} \langle f, f_\chi \rangle_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} = 1 - \varepsilon$$

we have therefore that $d_{\mathscr{H}}$ is a $1 - \varepsilon$ approximation of $\chi_2(\mathbb{P}, \mathbb{Q})$:

$$d_{\mathscr{H}}(\mathbb{P}, \mathbb{Q}) = (1 - \varepsilon)\chi_2(\mathbb{P}, \mathbb{Q}).$$

Since f and f_χ are unit norm in $\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$ we have the following relative error:

$$\frac{\chi_2(\mathbb{P}, \mathbb{Q}) - d_{\mathscr{H}}(\mathbb{P}, \mathbb{Q})}{\chi_2(\mathbb{P}, \mathbb{Q})} = \frac{1}{2} \inf_{f \in \mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}} \|f - f_\chi\|^2_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q})}{2}}. \tag{15}$$

□

E Theorem 3: Generalization Bounds

Let $\mathscr{H}$ be a function space of real valued functions on $\mathcal{X}$. We assume that $\mathscr{H}$ is bounded, there exists $\nu > 0$, such that $\|f\|_\infty \leq \nu$. Since the second moments are bounded we can relax this assumption using Chebyshev's inequality, we have:

$$\mathbb{P}\{x \in \mathcal{X}, |f(x)| \leq \nu\} \leq \frac{\mathbb{E}_{x \sim \frac{\mathbb{P}+\mathbb{Q}}{2}} f^2(x)}{\nu^2} = \frac{1}{\nu^2},$$

hence we have boundedness with high probability. Define the expected mean discrepancy $\mathscr{E}(.)$ and the second order norm $\Omega(.)$:

$$\mathscr{E}(f) = \mathbb{E}_{x \sim \mathbb{P}} f(x) - \mathbb{E}_{x \sim \mathbb{Q}} f(x) \ , \Omega(f) = \frac{1}{2} \left(\mathbb{E}_{x \sim \mathbb{P}} f^2(x) + \mathbb{E}_{x \sim \mathbb{Q}} f^2(x) \right)$$

and their empirical counterparts, given N samples $\{x_i\}_{i=1}^N \sim \mathbb{P}$,$\{y_i\}_{i=1}^M \sim \mathbb{Q}$:

$$\hat{\mathscr{E}}(f) = \frac{1}{N} \sum_{i=1}^N f(x_i) - \frac{1}{M} \sum_{i=1}^M f(y_i), \ \hat{\Omega}(f) = \frac{1}{2N} \sum_{i=1}^N f^2(x_i) + \frac{1}{2M} \sum_{i=1}^M f^2(y_i),$$

Theorem 3. *Let $n = \frac{MN}{M+N}$. Let $\mathbb{P}, \mathbb{Q} \in \mathscr{P}(\mathcal{X}), \mathbb{P} \neq \mathbb{Q}$, and let $\chi_2(\mathbb{P}, \mathbb{Q})$ be their Chi-squared distance. Let $f^* \in \arg\max_{f \in \mathscr{H}, \Omega(f)=1} \mathscr{E}(f)$, and $\hat{f} \in \arg\max_{f \in \mathscr{H}, \hat{\Omega}(f)=1} \hat{\mathscr{E}}(f)$. Define the expected mean discrepancy of the optimal empirical critic $\hat{f}$:*

$$\hat{d}_{\mathscr{H}}(\mathbb{P}, \mathbb{Q}) = \mathscr{E}(\hat{f})$$

For $\tau > 0$. The following generalization bound on the estimation of the Chi-squared distance, with probability $1 - 12e^{-\tau}$:

$$\frac{\chi_2(\mathbb{P}, \mathbb{Q}) - \hat{d}_{\mathscr{H}}(\mathbb{P}, \mathbb{Q})}{\chi_2(\mathbb{P}, \mathbb{Q})} \leq \underbrace{\frac{1}{2} \inf_{f \in \mathscr{H} \cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}} \|f - f_\chi\|^2_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})}}_{\textit{approximation error}} + \underbrace{\frac{\varepsilon_n}{\chi_2(\mathbb{P}, \mathbb{Q})}}_{\textit{Statistical Error}} . \tag{16}$$

where

$$\varepsilon_n = c_3 \mathscr{R}_{M,N}(f; \{f \in \mathscr{H}, \hat{\Omega}(f) \leq 1 + \nu^2 + 2\eta_n\}, S)$$
$$+ c_4(1 + 2\nu\hat{\lambda}) \mathscr{R}_{M,N}(f; \{f \in \mathscr{H}, \hat{\Omega}(f) \leq 1 + \frac{\nu^2}{2} + \eta_n\}, S) + O(\frac{1}{\sqrt{n}})$$

and

$$\eta_n \geq c_1 \nu \mathscr{R}_{N,M}(f; f \in \mathscr{H}, S) + c_2 \frac{\nu^2 \tau}{n},$$

$\hat{\lambda}$ is the Lagrange multiplier, c_1, c_2, c_3, c_4 are numerical constants, and $\mathscr{R}_{M,N}$ is the rademacher complexity:

$$\mathscr{R}_{M,N}(f; \mathscr{F}, S) = E_\sigma \sup_{f \in \mathscr{F}} \left[\sum_{i=1}^{N+M} \sigma_i \tilde{Y}_i f(X_i) | S \right],$$

$\tilde{Y} = (\underbrace{\frac{1}{N}, \dots \frac{1}{N}}_{N}, \underbrace{\frac{-1}{M} \dots \frac{-1}{M}}_{M})$, $S = \{x_1 \dots x_N, y_1 \dots y_M\}$,$\sigma_i = \pm 1$ *with probability $\frac{1}{2}$, that are iids.*

For example:

$$\mathscr{H} = \{f(x) = \langle v, \Phi(x) \rangle, v \in \mathbb{R}^m\}$$

Note that for simplicity here we assume that the feature map is fixed $\Phi : \mathcal{X} \to \mathbb{R}^m$, and we parametrize the class function only with v.

$$\mathscr{R}_{M,N}(f; \{\mathscr{H}, \hat{\Omega}(f) \leq R\}, S)) \leq \sqrt{2R \frac{d(\gamma)}{n}},$$

where

$$d(\gamma) = \sum_{j=1}^{m} \frac{\sigma_j^2}{\sigma_j^2 + \gamma}$$

is the effective dimension ($d(\gamma) << m$). Hence we see that typically $\varepsilon_n = O(\frac{1}{\sqrt{n}})$.

Proof of Theorem 3. Let $\{x_i\}_{i=1}^N \sim \mathbb{P}$, $\{y_i\}_{i=1}^M \sim \mathbb{Q}$. Define the following functionals:

$$\mathscr{E}(f) = \mathbb{E}_{x \sim \mathbb{P}} f(x) - \mathbb{E}_{x \sim \mathbb{Q}} f(x) \ , \Omega(f) = \frac{1}{2} \left(\mathbb{E}_{x \sim \mathbb{P}} f^2(x) + \mathbb{E}_{x \sim \mathbb{Q}} f^2(x) \right)$$

and their empirical estimates:

$$\hat{\mathscr{E}}(f) = \frac{1}{N} \sum_{i=1}^{N} f(x_i) - \frac{1}{M} \sum_{i=1}^{M} f(y_i), \ \hat{\Omega}(f) = \frac{1}{2N} \sum_{i=1}^{N} f^2(x_i) + \frac{1}{2M} \sum_{i=1}^{M} f^2(y_i)$$

Define the following Lagrangians:

$$\mathcal{L}(f, \lambda) = \mathscr{E}(f) + \frac{\lambda}{2}(1 - \Omega(f)), \ \hat{\mathcal{L}}(f, \lambda) = \hat{\mathscr{E}}(f) + \frac{\lambda}{2}(1 - \hat{\Omega}(f))$$

Recall some definitions of the Fisher IPM:

$$d_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) = \sup_{f\in\mathscr{H}} \inf_{\lambda\geq 0} \mathcal{L}(f,\lambda) \text{ achieved at } (f_*,\lambda_*)$$

We assume that a saddle point for this problem exists and it is feasible. We assume also that $\hat{\lambda}$ is positive and bounded.

$$d_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) = \mathscr{E}(f_*) \text{ and } \Omega(f_*) = 1$$
$$\mathcal{L}(f,\lambda_*) \leq \mathcal{L}(f_*,\lambda_*) \leq \mathcal{L}(f_*,\lambda)$$

The fisher IPM empirical estimate is given by:

$$d_{\mathscr{H}}(\mathbb{P}_N,\mathbb{Q}_N) = \sup_{f\in\mathscr{H}} \inf_{\lambda\geq 0} \hat{\mathcal{L}}(f,\lambda), \text{ achieved at } (\hat{f},\hat{\lambda})$$

hence we have:

$$d_{\mathscr{H}}(\mathbb{P}_N,\mathbb{Q}_N) = \hat{\mathscr{E}}(\hat{f}) \text{ and } \hat{\Omega}(\hat{f}) = 1.$$

The Generalization error of the empirical critic $\hat{f}$ is the expected mean discrepancy $\mathscr{E}(\hat{f})$. We note $\hat{d}_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) = \mathscr{E}(\hat{f})$, the estimated distance using the critic $\hat{f}$, on out of samples:

$$\begin{aligned}\chi_2(\mathbb{P},\mathbb{Q}) - \hat{d}_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) &= \mathscr{E}(f_\chi) - \mathscr{E}(\hat{f}) \\ &= \underbrace{\mathscr{E}(f_\chi) - \mathscr{E}(f^*)}_{\text{Approximation Error}} + \underbrace{\mathscr{E}(f^*) - \mathscr{E}(\hat{f})}_{\text{Statistical Error}}\end{aligned}$$

Bounding the Approximation Error. By Theorem 2 we know that:

$$\mathscr{E}(f_\chi) - \mathscr{E}(f^*) = \chi_2(\mathbb{P},\mathbb{Q}) - d_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) = \frac{\chi_2(\mathbb{P},\mathbb{Q})}{2} \inf_{f\in\mathscr{H}\cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}} \|f - f_\chi\|^2_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}.$$

Hence we have for $\mathbb{P} \neq \mathbb{Q}$:

$$\frac{\chi_2(\mathbb{P},\mathbb{Q}) - \hat{d}_{\mathscr{H}}(\mathbb{P},\mathbb{Q})}{\chi_2(\mathbb{P},\mathbb{Q})} = \frac{1}{2} \inf_{f\in\mathscr{H}\cap \mathbb{S}_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}} \|f - f_\chi\|^2_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})} + \underbrace{\frac{\mathscr{E}(f^*) - \mathscr{E}(\hat{f})}{\chi_2(\mathbb{P},\mathbb{Q})}}_{\text{Statistical Error}} \tag{17}$$

Note that this equation tells us that the relative error depends on the approximation error of the the optimal critic f_χ, and the statistical error coming from using finite samples in approximating the distance. We note that the statistical error is divided by the Chi-squared distance, meaning that we need a bigger sample size when $\mathbb{P}$ and $\mathbb{Q}$ are close in the Chi-squared sense, in order to reduce the overall relative error.

Hence we are left with bounding the statistical error using empirical processes theory. Assume $\mathscr{H}$ is a space of bounded functions i.e $\|f\|_\infty \leq \nu$.

Bounding the Statistical Error. Note that we have: (i) $\hat{\mathcal{L}}(f^*,\hat{\lambda}) \leq \hat{\mathcal{L}}(\hat{f},\hat{\lambda})$ and (ii) $\Omega(f^*) = 1$.

$$\begin{aligned}\mathscr{E}(f^*) - \mathscr{E}(\hat{f}) &= \Big(\mathscr{E}(f^*) - \hat{\mathscr{E}}(f^*)\Big) + \underbrace{(\hat{\mathscr{E}}(f^*) + \frac{\hat{\lambda}}{2}(1-\hat{\Omega}(f^*)))}_{\hat{\mathcal{L}}(f^*,\hat{\lambda})} - \underbrace{\hat{\mathscr{E}}(\hat{f})}_{\hat{\mathcal{L}}(\hat{f},\hat{\lambda})}) + \Big(\hat{\mathscr{E}}(\hat{f}) - \mathscr{E}(\hat{f})\Big) + \frac{\hat{\lambda}}{2}\Big(\hat{\Omega}(f^*) - 1\Big) \\ &\leq \sup_{f\in\mathscr{H},\Omega(f)\leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)| + \sup_{f\in\mathscr{H},\hat{\Omega}(f)\leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)| + \frac{\hat{\lambda}}{2}\Big(\hat{\Omega}(f^*) - \Omega(f^*)\Big) \text{ Using (i) and (ii)} \\ &\leq \sup_{f\in\mathscr{H},\Omega(f)\leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)| + \sup_{f\in\mathscr{H},\hat{\Omega}(f)\leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)| + \frac{\hat{\lambda}}{2} \sup_{f\in\mathscr{H},\Omega(f)\leq 1} |\hat{\Omega}(f) - \Omega(f)|.\end{aligned}$$

Let $S = \{x_1 \dots x_N, y_1 \dots y_M\}$. Define the following quantities:

$Z_1(S) = \sup_{f\in\mathscr{H},\Omega(f)\leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)|$, Concentration of the cost on data distribution dependent constraint

$Z_2(S) = \sup_{f\in\mathscr{H},\hat{\Omega}(f)\leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)|$, Concentration of the cost on an empirical data dependent constraint

$Z_3(S) = \sup_{f\in\mathscr{H},\Omega(f)\leq 1} |\hat{\Omega}(f) - \Omega(f)|, \hat{\lambda} Z_3(S)$ is the sensitivity of the cost as the constraint set changes

We have:

$$\mathscr{E}(f^*) - \mathscr{E}(\hat{f}) \leq Z_1(S) + Z_2(S) + \hat{\lambda} Z_3(S), \tag{18}$$

Note that the sup in $Z_1(S)$ and $Z_3(S)$ is taken with respect to class function $\{f, \Omega(f) = \|f\|^2_{\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})} \leq 1\}$ hence we will bound $Z_1(S)$, and $Z_3(S)$ using local Rademacher complexity. In $Z_2(S)$ the sup is taken on a data dependent function class and can be bounded with local rademacher complexity as well but needs more careful work.

Bounding $Z_1(S)$, and $Z_3(S)$

Lemma 1 (Bounds with (Local) Rademacher Complexity [11, 17]). *Let $Z(S) = \sup_{f\in\mathscr{F}} \mathscr{E}(f) - \hat{\mathscr{E}}(f)$, Assume that $\|f\|_\infty \leq \nu$, for all $f \in \mathscr{F}$.*

- *For any $\alpha, \tau > 0$. Define variances $var_{\mathbb{P}}(f)$, and similarly $var_{\mathbb{Q}}(f)$. Assume $\max(var_{\mathbb{P}}(f), var_{\mathbb{Q}}(f)) \leq r$ for any $f \in \mathscr{F}$. We have with probability $1 - e^{-\tau}$:*

$$Z(S) \leq (1+\alpha) E_S Z(S) + \sqrt{\frac{2r\tau(M+N)}{MN}} + \frac{2\tau\nu(M+N)}{MN}\left(\frac{2}{3} + \frac{1}{\alpha}\right)$$

  The same result holds for : $Z(S) = \sup_{f\in\mathscr{F}} \hat{\mathscr{E}}(f) - \mathscr{E}(f)$.

- *By symmetrization we have: $\mathbb{E}_S Z(S) \leq 2E_S \mathscr{R}_{M,N}(f; \mathscr{F}, S)$ where $\mathscr{R}_{M,N}$ is the rademacher complexity:*

$$\mathscr{R}_{M,N}(f; \mathscr{F}, S) = E_\sigma \sup_{f\in\mathscr{F}} \left[\sum_{i=1}^{N+M} \sigma_i \tilde{Y}_i f(X_i) | S \right],$$

  $\tilde{Y} = (\underbrace{\frac{1}{N}, \dots \frac{1}{N}}_{N}, \underbrace{\frac{-1}{M} \dots \frac{-1}{M}}_{M})$, *$\sigma_i = \pm 1$ with probability $\frac{1}{2}$, that are iids.*

- *We have with probability $1 - e^{-\tau}$ for all $\delta \in (0,1)$:*

$$E_S \mathscr{R}_{M,N}(f; \mathscr{F}, S) \leq \frac{\mathscr{R}_{M,N}(f; \mathscr{F}, S)}{1-\delta} + \frac{\tau\nu(M+N)}{MN\delta(1-\delta)}.$$

Lemma 2 (Contraction Lemma [17]). *Let ϕ be a contraction, that is $|\phi(x) - \phi(y)| \leq L|x-y|$. Then, for every class $\mathscr{F}$,*

$$\mathscr{R}_{M,N}(f; \phi \circ \mathscr{F}, S) \leq L \mathscr{R}_{M,N}(f; \mathscr{F}, S),$$

$\phi \circ \mathscr{F} = \{\phi \circ f, f \in \mathscr{F}\}$.

Let $n = \frac{MN}{M+N}$. Applying Lemma 1 for $\mathscr{F} = \{f \in \mathscr{H}, \Omega(f) \leq 1\}$. Since $\Omega(f) \leq 1$, $var_{\mathbb{P}}(f) \leq \Omega(f) \leq 1$, and similarly for $var_{\mathbb{Q}}(f)$. Hence $\max(var_{\mathbb{P}}(f), var_{\mathbb{Q}}(f)) \leq 1$. Putting all together we obtain with probability $1 - 2e^{-\tau}$:

$$Z_1(S) \leq \frac{2(1+\alpha)}{1-\delta} \mathscr{R}_{M,N}(f; \{f \in \mathscr{H}, \Omega(f) \leq 1\}, S) + \sqrt{\frac{2\tau}{n}} + \frac{2\tau\nu}{n}\left(\frac{2}{3} + \frac{1}{\alpha} + \frac{1+\alpha}{\delta(1-\delta)}\right) \tag{19}$$

Now tuning to $Z_3(S)$ applying Lemma 1 for $\{f^2, f \in \mathscr{H}, \Omega(f) \leq 1\}$. Note that $Var(f^2) \leq \mathbb{E} f^4 \leq \Omega(f)\nu^2 \leq \nu^2$. We have that for $\alpha > 0$, $\delta \in (0,1)$ and with probability at least $1 - 2e^{-\tau}$:

$$Z_3(S) \leq \frac{2(1+\alpha)}{1-\delta} \mathscr{R}_{N,M}(f^2; \{f \in \mathscr{H}, \Omega(f) \leq 1\}, S) + \sqrt{\frac{2\tau\nu^2}{n}} + \frac{2\tau\nu^2}{n}\left(\frac{2}{3} + \frac{1}{\alpha} + \frac{1+\alpha}{\delta(1-\delta)}\right)$$

Note that applying the contraction Lemma for $\phi(x) = x^2$ (with lipchitz constant 2ν on $[-\nu, \nu]$) we have:

$$\mathscr{R}_{N,M}(f^2; \{f \in \mathscr{H}, \Omega(f) \leq 1\}, S) \leq 2\nu \mathscr{R}_{N,M}(f; \{f \in \mathscr{H}, \Omega(f) \leq 1\}, S),$$

Hence we have finally:

$$Z_3(S) \leq \frac{4(1+\alpha)\nu}{1-\delta} \mathscr{R}_{N,M}(f; \{f \in \mathscr{H}, \Omega(f) \leq 1\}, S) + \sqrt{\frac{2\tau\nu^2}{n}} + \frac{2\tau\nu^2}{n}\left(\frac{2}{3} + \frac{1}{\alpha} + \frac{1+\alpha}{\delta(1-\delta)}\right) \tag{20}$$

Note that the of complexity of $\mathscr{H}$, depends also upon the distributions $\mathbb{P}$ and $\mathbb{Q}$, since it is defined on the intersection of $\mathscr{H}$ and the unity ball in $\mathscr{L}_2(\mathcal{X}, \frac{\mathbb{P}+\mathbb{Q}}{2})$.

From Distributions to Data dependent Bounds. We study how the $\hat{\Omega}(f)$ concentrates uniformly on $\mathscr{H}$. Note that in this case to apply Lemma 1, we use $r \leq \mathbb{E}(f^4) \leq \nu^4$. We have with probability $1 - 2e^{-\tau}$:

$$\hat{\Omega}(f) \leq \Omega(f) + \frac{4(1+\alpha)\nu}{1-\delta} \mathscr{R}_{N,M}(f; f \in \mathscr{H}, S) + \sqrt{\frac{2\tau\nu^4 r}{n}} + \frac{2\tau\nu^2}{n}\left(\frac{2}{3} + \frac{1}{\alpha} + \frac{1+\alpha}{\delta(1-\delta)}\right)$$

Now using that for any $\alpha > 0$: $2\sqrt{uv} \leq \alpha u + \frac{v}{\alpha}$ we have for $\alpha = \frac{1}{2}$: $\sqrt{\frac{2\tau\nu^4}{n}} \leq \frac{\nu^2}{2} + \frac{4\tau\nu^2}{n}$. For some universal constants, c_1, c_2, let:

$$\eta_n \geq c_1 \nu \mathscr{R}_{N,M}(f; f \in \mathscr{H}, S) + c_2 \frac{\nu^2 \tau}{n},$$

we have therefore with probability $1 - 2e^{-\tau}$:

$$\hat{\Omega}(f) \leq \Omega(f) + \frac{\nu^2}{2} + \eta_n, \tag{21}$$

note that Typically $\eta_n = O(\frac{1}{\sqrt{n}})$.
The same inequality holds with the same probability:

$$\Omega(f) \leq \hat{\Omega}(f) + \frac{\nu^2}{2} + \eta_n, \tag{22}$$

Note that we have now the following inclusion using Equation (21):

$$\{f, f \in \mathscr{H}, \Omega(f) \leq 1\} \subset \left\{f, f \in \mathscr{H}, \hat{\Omega}(f) \leq 1 + \frac{\nu^2}{2} + \eta_n\right\}$$

Hence:

$$\mathscr{R}_{M,N}(f; \{f \in \mathscr{H}, \Omega(f) \leq 1\}, S) \leq \mathscr{R}_{M,N}(f; \{f \in \mathscr{H}, \hat{\Omega}(f) \leq 1 + \frac{\nu^2}{2} + \eta_n\}, S)$$

Hence we obtain a data dependent bound in Equations (19),(20) with a union bound with probability $1 - 6e^{-\tau}$.

Bounding $Z_2(S)$. Note that concentration inequalities don't apply to $Z_2(S)$ since the cost function and the function class are data dependent. We need to turn the constraint to a data independent constraint i.e does not depend on the training set. For $f, \hat{\Omega}(f) \leq 1$, by Equation (22) we have with probability $1 - 2e^{-\tau}$:

$$\Omega(f) \leq 1 + \frac{\nu^2}{2} + \eta_n,$$

we have therefore the following inclusion with probability $1 - 2e^{-\tau}$:

$$\{f \in \mathscr{H}, \hat{\Omega}(f) \leq 1\} \subset \{f \in \mathscr{H}, \Omega(f) \leq 1 + \frac{\nu^2}{2} + \eta_n\}$$

Recall that:

$$Z_2(S) = \sup_{f \in \mathscr{H}, \hat{\Omega}(f) \leq 1} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)|$$

Hence with probability $1-2e^{-\tau}$:

$$Z_2(S) \leq \tilde{Z}_2(S) = \sup_{f, f\in\mathscr{H}, \Omega(f)\leq 1+\frac{\nu^2}{2}+\eta_n} |\hat{\mathscr{E}}(f) - \mathscr{E}(f)|$$

Applying again Lemma 1 on $\tilde{Z}_2(S)$ we have with probability $1-4e^{-\tau}$:

$$Z_2(S) \leq \tilde{Z}_2(S) \leq \frac{2(1+\alpha)}{1-\delta}\mathscr{R}_{M,N}(f; \{f\in\mathscr{H}, \Omega(f)\leq 1+\frac{\nu^2}{2}+\eta_n\}, S) + \sqrt{\frac{2\tau(1+\frac{\nu^2}{2}+\eta_n)}{n}} + \frac{2\tau\nu}{n}\left(\frac{2}{3}+\frac{1}{\alpha}+\frac{1+\alpha}{\delta(1-\delta)}\right).$$

Now reapplying the inclusion using Equation (21), we get the following bound on the local rademacher complexity with probability $1-2e^{-\tau}$:

$$\mathscr{R}_{M,N}(f; \{f\in\mathscr{H}, \Omega(f)\leq 1+\frac{\nu^2}{2}+\eta_n\}, S) \leq \mathscr{R}_{M,N}(f; \{f\in\mathscr{H}, \hat{\Omega}(f)\leq 1+\nu^2+2\eta_n\}, S)$$

Hence with probability $1-6e^{-\tau}$ we have:

$$Z_2(S) \leq \frac{2(1+\alpha)}{1-\delta}\mathscr{R}_{M,N}(f; \{f\in\mathscr{H}, \hat{\Omega}(f)\leq 1+\nu^2+2\eta_n\}, S) + \sqrt{\frac{2\tau(1+\frac{\nu^2}{2}+\eta_n)}{n}} + \frac{2\tau\nu}{n}\left(\frac{2}{3}+\frac{1}{\alpha}+\frac{1+\alpha}{\delta(1-\delta)}\right).$$

Putting all together. We have with probability at least $1-12e^{-\tau}$, for universal constants c_1, c_2, c_3, c_4

$$\eta_n \geq c_1\nu\mathscr{R}_{N,M}(f; f\in\mathscr{H}, S) + c_2\frac{\nu^2\tau}{n},$$

$$\begin{aligned}
\mathscr{E}(f^*) - \mathscr{E}(\hat{f}) &\leq Z_1(S) + Z_2(S) + \hat{\lambda}Z_3(S) \\
&\leq \varepsilon_n \\
&= c_3\mathscr{R}_{M,N}(f; \{f\in\mathscr{H}, \hat{\Omega}(f)\leq 1+\nu^2+2\eta_n\}, S) \\
&+ c_4(1+2\nu\hat{\lambda})\mathscr{R}_{M,N}(f; \{f\in\mathscr{H}, \hat{\Omega}(f)\leq 1+\frac{\nu^2}{2}+\eta_n\}, S) \\
&+ O(\frac{1}{\sqrt{n}}).
\end{aligned}$$

Note that typically $\varepsilon_n = O(\frac{1}{\sqrt{n}})$. Hence it follows that:

$$\frac{\chi_2(\mathbb{P},\mathbb{Q}) - \hat{d}_{\mathscr{H}}(\mathbb{P},\mathbb{Q})}{\chi_2(\mathbb{P},\mathbb{Q})} \leq \underbrace{\frac{1}{2}\inf_{f\in\mathscr{H}\cap\mathbb{S}_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}} \|f-f_\chi\|^2_{\mathscr{L}_2(\mathcal{X},\frac{\mathbb{P}+\mathbb{Q}}{2})}}_{\text{approximation error}} + \underbrace{\frac{\varepsilon_n}{\chi_2(\mathbb{P},\mathbb{Q})}}_{\text{Statistical Error}}. \tag{23}$$

If $\mathbb{P}$ and $\mathbb{Q}$ are close we need more samples to estimate the χ_2 distance and reduce the relative error.

Example: Bounding local complexity for a simple linear function class.

$$\mathscr{H} = \{f(x) = \langle v, \Phi(x)\rangle, v\in\mathbb{R}^m\}$$

Note that for simplicity here we assume that the feature map is fixed $\Phi : \mathcal{X} \to \mathbb{R}^m$, and we parametrize the class function only with v. Note that

$\sup_{v, v^\top(\Sigma(\mathbb{P}_N)+\Sigma(\mathbb{Q}_M)+\gamma I_m)v \le 2R} \left\langle v, \sum_{i=1}^N \sigma_i \tilde{Y}_i \Phi(X_i) \right\rangle$

$$= \sup_{v, \|v\| \le 1} \left\langle v, \left(\frac{\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m}{2R} \right)^{-\frac{1}{2}} \sum_{i=1}^N \sigma_i \tilde{Y}_i \Phi(X_i) \right\rangle$$

$$= \left\| \left(\frac{\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m}{2R} \right)^{-\frac{1}{2}} \sum_{i=1}^{N+M} \sigma_i \tilde{Y}_i \Phi(X_i) \right\|$$

$$= \sqrt{2R} \sqrt{\sum_{i,j=1}^{N+M} \sigma_i \sigma_j \tilde{Y}_i \tilde{Y}_j \Phi(X_i)^\top \left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m \right)^{-1} \Phi(X_j)}$$

It follows by Jensen inequality that $\mathbb{E}_\sigma \sup_{v, v^\top(\Sigma(\mathbb{P}_N)+\Sigma(\mathbb{Q}_M)+\gamma I_m)v \le \sqrt{2R}} \left\langle v, \sum_{i=1}^N \sigma_i \tilde{Y}_i \Phi(X_i) \right\rangle$

$$\le \sqrt{2R} \sqrt{\mathbb{E}_\sigma \sum_{i,j=1}^{N+M} \sigma_i \sigma_j \tilde{Y}_i \tilde{Y}_j \Phi(X_i)^\top \left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m \right)^{-1} \Phi(X_j)}$$

$$= \sqrt{2R} \sqrt{\sum_{i=1}^{N+M} \tilde{Y}_i^2 \Phi(X_i)^\top \left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m \right)^{-1} \Phi(X_i)}$$

$$= \sqrt{2R} \sqrt{Tr\left(\left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m \right)^{-1} \left(\frac{1}{N} \Sigma(\mathbb{P}_N) + \frac{1}{M} \Sigma(\mathbb{Q}_M) \right) \right)}$$

$$\le \sqrt{2R \frac{M+N}{MN}} \sqrt{Tr\left(\left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m \right)^{-1} \left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) \right) \right)}$$

Let

$$d(\gamma) = Tr\left(\left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) + \gamma I_m \right)^{-1} \left(\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M) \right) \right),$$

$d(\gamma)$ is the so called effective dimension in regression problems. Let Σ be the singular values of $\Sigma(\mathbb{P}_N) + \Sigma(\mathbb{Q}_M)$,

$$d(\gamma) = \sum_{j=1}^m \frac{\sigma_j^2}{\sigma_j^2 + \gamma}$$

Hence we obtain the following bound on the local rademacher complexity:

$$\mathscr{R}_{M,N}(f; \{\mathscr{H}, \hat{\Omega}(f) \le R\}, S)) \le \sqrt{2R \frac{(M+N) d(\gamma)}{MN}} = \sqrt{2R \frac{d(\gamma)}{n}}$$

Note that without the local constraint the effective dimension $d(\gamma)$ (typically $d(\gamma) << m$) is replaced by the ambient dimension m. □

F Hyper-parameters and Architectures of Discriminator and Generators

For CIFAR-10 we use adam learning rate $\eta = 2\text{e}{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and penalty weight $\rho = 3\text{e}{-7}$, for LSUN and CelebA we use $\eta = 5\text{e}{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and $\rho = 1\text{e}{-6}$. We found the optimization to be stable with very similar performance in the range $\eta \in [1\text{e}{-4}, 1\text{e}{-3}]$ and $\rho \in [1\text{e}{-7}, 1\text{e}{-5}]$ across our experiments. We found weight initialization from a normal distribution with stdev=0.02 to perform better than Glorot [38] or He [39] initialization for both Fisher GAN and WGAN-GP. This initialization is the default in pytorch, while in the WGAN-GP codebase He init [39] is used. Specifically the initialization of the generator is more important.

We used some L2 weight decay: $1\text{e}{-6}$ on ω (i.e. all layers except last) and $1\text{e}{-3}$ weight decay on the last layer v.

F.1 Inception score WGAN-GP baselines: comparison of architecture and weight initialization

As noted in Figure 5 and in above paragraph, we used intialization from a normal distribution with stdev=0.02 for the inception score experiments for both Fisher GAN and WGAN-GP. For transparency, and to show that our architecture and initialization benefits both Fisher GAN and WGAN-GP, we provide plots of different combinations below (Figure 6). Architecture-wise, F64 refers to the architecture described in Appendix F.3 with 64 feature maps after the first convolutional layer. F128 is the architecture from the WGAN-GP codebase [7], which has double the number of feature maps (128 fmaps) and does not have the two extra layers in G and D (D layers 2-7, G layers 9-14). The result reported in the WGAN-GP paper [7] corresponds to `WGAN-GP F128 He init`. For WGAN (Figure 7) the 64-fmap architecture gives some initial instability but catches up to the same level as the 128-fmap architecture.

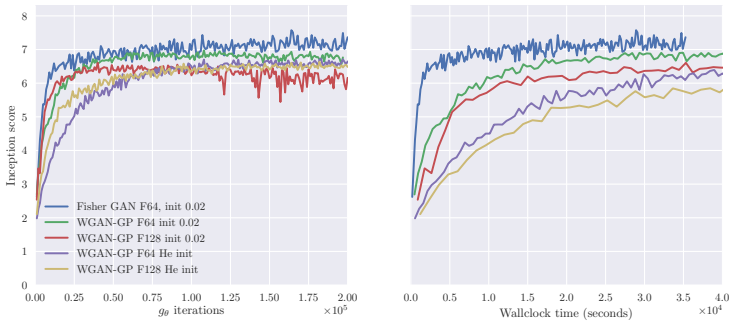

Figure 6: Architecture and initialization variations, trained with WGAN-GP. Fisher included for comparison. In the main text (Figure 5) we only compare against the best architecture F64 init 0.02.

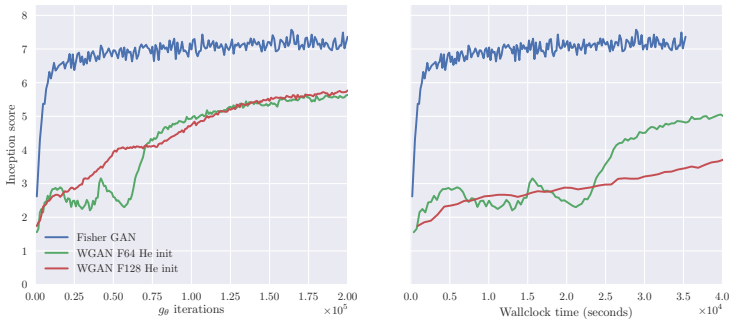

Figure 7: Architecture variations, trained with WGAN. Fisher included for comparison.

F.2 LSUN and CelebA.

```
### LSUN and CelebA: 64x64 dcgan with G_extra_layers=2 and
    D_extra_layers=0
G (
  (main): Sequential (
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1),
       bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU (inplace)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2),
       padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (5): ReLU (inplace)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2),
       padding=(1, 1), bias=False)
```

```
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (8): ReLU (inplace)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2),
       padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (11): ReLU (inplace)
    (12): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
       1), bias=False)
    (13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (14): ReLU (inplace)
    (15): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
       1), bias=False)
    (16): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (17): ReLU (inplace)
    (18): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2),
       padding=(1, 1), bias=False)
    (19): Tanh ()
  )
)
D (
  (main): Sequential (
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
       bias=False)
    (1): LeakyReLU (0.2, inplace)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1,
       1), bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (4): LeakyReLU (0.2, inplace)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1,
       1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (7): LeakyReLU (0.2, inplace)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1,
       1), bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (10): LeakyReLU (0.2, inplace)
  )
  (V): Linear (8192 -> 1)
)
```

F.3 CIFAR-10: Sample Quality and Inceptions Scores Experiments

```
### CIFAR-10: 32x32 dcgan with G_extra_layers=2 and D_extra_layers=2.
    For samples and inception.
G (
  (main): Sequential (
    (0): ConvTranspose2d(100, 256, kernel_size=(4, 4), stride=(1, 1),
       bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU (inplace)
    (3): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2),
       padding=(1, 1), bias=False)
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (5): ReLU (inplace)
    (6): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2),
       padding=(1, 1), bias=False)
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (8): ReLU (inplace)
    (9): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
       1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (11): ReLU (inplace)
```

```
    (12): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1), bias=False)
    (13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (14): ReLU (inplace)
    (15): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2),
        padding=(1, 1), bias=False)
    (16): Tanh ()
  )
)
D (
  (main): Sequential (
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
        bias=False)
    (1): LeakyReLU (0.2, inplace)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1), bias=False)
    (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (4): LeakyReLU (0.2, inplace)
    (5): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1), bias=False)
    (6): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (7): LeakyReLU (0.2, inplace)
    (8): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1,
        1), bias=False)
    (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (10): LeakyReLU (0.2, inplace)
    (11): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1,
        1), bias=False)
    (12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (13): LeakyReLU (0.2, inplace)
  )
  (V): Linear (4096 -> 1)
  (S): Linear (6144 -> 10)
)
```

F.4 CIFAR-10: SSL Experiments

```
### CIFAR-10: 32x32 D is in the flavor OpenAI Improved GAN, ALI.
G same as above.

D (
  (main): Sequential (
    (0): Dropout (p = 0.2)
    (1): Conv2d(3, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (2): LeakyReLU (0.2, inplace)
    (3): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1), bias=False)
    (4): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True)
    (5): LeakyReLU (0.2, inplace)
    (6): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1,
        1), bias=False)
    (7): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True)
    (8): LeakyReLU (0.2, inplace)
    (9): Dropout (p = 0.5)
    (10): Conv2d(96, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1), bias=False)
    (11): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True)
    (12): LeakyReLU (0.2, inplace)
    (13): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1), bias=False)
    (14): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True)
    (15): LeakyReLU (0.2, inplace)
```

```
    (16): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), padding=(1,
        1), bias=False)
    (17): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True)
    (18): LeakyReLU (0.2, inplace)
    (19): Dropout (p = 0.5)
    (20): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (21): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True)
    (22): LeakyReLU (0.2, inplace)
    (23): Dropout (p = 0.5)
    (24): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (25): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True)
    (26): LeakyReLU (0.2, inplace)
    (27): Dropout (p = 0.5)
    (28): Conv2d(384, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (29): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True)
    (30): LeakyReLU (0.2, inplace)
    (31): Dropout (p = 0.5)
  )
  (V): Linear (6144 -> 1)
  (S): Linear (6144 -> 10)
)
```

G Sample implementation in PyTorch

This minimalistic sample code is based on `https://github.com/martinarjovsky/WassersteinGAN` at commit d92c503.

Some elements that could be added are:

- Validation loop
- Monitoring of weights and activations
- Separate weight decay for last layer v (we trained with $1\mathrm{e}{-3}$ weight decay on v).
- Adding Cross-Entropy objective and class-conditioned generator.

G.1 Main loop

First note the essential change in the critic's forward pass definition:

```
-       output = output.mean(0)
-       return output.view(1)
+       return output.view(-1)
```

Then the main training loop becomes:

```
gen_iterations = 0
for epoch in range(opt.niter):
   data_iter = iter(dataloader)
   i = 0
   while i < len(dataloader):
      ############################
      # (1) Update D network
      ###########################
      for p in netD.parameters(): # reset requires_grad
         p.requires_grad = True # they are set to False below in netG update

      # train the discriminator Diters times
      if opt.hiDiterStart and (gen_iterations < 25 or gen_iterations % 500 == 0):
         Diters = 100
      else:
         Diters = opt.Diters
      j = 0
      while j < Diters and i < len(dataloader):
         j += 1

         data = data_iter.next()
         i += 1

         # train with real
```

```
        real_cpu, _ = data
        netD.zero_grad()
        batch_size = real_cpu.size(0)

        if opt.cuda:
            real_cpu = real_cpu.cuda()
        input.resize_as_(real_cpu).copy_(real_cpu)
        inputv = Variable(input)

        vphi_real = netD(inputv)

        # train with fake
        noise.resize_(opt.batchSize, nz, 1, 1).normal_(0, 1)
        noisev = Variable(noise, volatile = True) # totally freeze netG
        fake = Variable(netG(noisev).data)
        inputv = fake

        vphi_fake = netD(inputv)
        # NOTE here f = <v,phi> , but with modified f the below two lines are the
        # only ones that need change. E_P and E_Q refer to Expectation over real and fake.
        E_P_f, E_Q_f = vphi_real.mean(), vphi_fake.mean()
        E_P_f2, E_Q_f2 = (vphi_real**2).mean(), (vphi_fake**2).mean()
        constraint = (1 - (0.5*E_P_f2 + 0.5*E_Q_f2))
        # See Equation (9)
        obj_D = E_P_f - E_Q_f + alpha * constraint - opt.rho/2 * constraint**2
        # max_w min_alpha obj_D. Compute negative gradients, apply updates with negative sign.
        obj_D.backward(mone)
        optimizerD.step()
        # artisanal sgd. We minimze alpha so a <- a + lr * (-grad)
        alpha.data += opt.rho * alpha.grad.data
        alpha.grad.data.zero_()

    ############################
    # (2) Update G network
    ############################
    for p in netD.parameters():
        p.requires_grad = False # to avoid computation
    netG.zero_grad()
    # in case our last batch was the tail batch of the dataloader,
    # make sure we feed a full batch of noise
    noise.resize_(opt.batchSize, nz, 1, 1).normal_(0, 1)
    noisev = Variable(noise)
    fake = netG(noisev)
    vphi_fake = netD(fake)
    obj_G = -vphi_fake.mean() # Just minimize mean difference
    obj_G.backward() # G: min_theta
    optimizerG.step()
    gen_iterations += 1
```

G.2 Full diff from reference

Note that from the arXiv LaTeX source, the file `diff.txt` could be used in combination with `git apply`.

```
diff --git a/main.py b/main.py
index 7c3e638..e0cae42 100644
--- a/main.py
+++ b/main.py
@@ -34,15 +34,17 @@ parser.add_argument('--cuda' , action='store_true', help='enables cuda')
 parser.add_argument('--ngpu' , type=int, default=1, help='number of GPUs to use')
 parser.add_argument('--netG', default='', help="path to netG (to continue training)")
 parser.add_argument('--netD', default='', help="path to netD (to continue training)")
-parser.add_argument('--clamp_lower', type=float, default=-0.01)
-parser.add_argument('--clamp_upper', type=float, default=0.01)
+parser.add_argument('--wdecay', type=float, default=0.000, help='wdecay value for Phi')
 parser.add_argument('--Diters', type=int, default=5, help='number of D iters per each G iter')
+parser.add_argument('--hiDiterStart' , action='store_true', help='do many D iters at start')
 parser.add_argument('--noBN', action='store_true', help='use batchnorm or not (only for DCGAN)')
 parser.add_argument('--mlp_G', action='store_true', help='use MLP for G')
 parser.add_argument('--mlp_D', action='store_true', help='use MLP for D')
-parser.add_argument('--n_extra_layers', type=int, default=0, help='Number of extra layers on gen and disc')
+parser.add_argument('--G_extra_layers', type=int, default=0, help='Number of extra layers on gen and disc')
+parser.add_argument('--D_extra_layers', type=int, default=0, help='Number of extra layers on gen and disc')
 parser.add_argument('--experiment', default=None, help='Where to store samples and models')
 parser.add_argument('--adam', action='store_true', help='Whether to use adam (default is rmsprop)')
+parser.add_argument('--rho', type=float, default=1e-6, help='Weight on the penalty term for (sigmas -1)**2')
 opt = parser.parse_args()
 print(opt)

@@ -60,7 +62,7 @@ cudnn.benchmark = True
 if torch.cuda.is_available() and not opt.cuda:
     print("WARNING: You have a CUDA device, so you should probably run with --cuda")

-if opt.dataset in ['imagenet', 'folder', 'lfw']:
+if opt.dataset in ['imagenet', 'folder', 'lfw', 'celeba']:
     # folder dataset
     dataset = dset.ImageFolder(root=opt.dataroot,
```

```
                        transform=transforms.Compose([
@@ -94,7 +96,6 @@ nz = int(opt.nz)
 ngf = int(opt.ngf)
 ndf = int(opt.ndf)
 nc = int(opt.nc)
-n_extra_layers = int(opt.n_extra_layers)

 # custom weights initialization called on netG and netD
 def weights_init(m):
@@ -106,11 +107,11 @@ def weights_init(m):
        m.bias.data.fill_(0)

 if opt.noBN:
-    netG = dcgan.DCGAN_G_nobn(opt.imageSize, nz, nc, ngf, ngpu, n_extra_layers)
+    netG = dcgan.DCGAN_G_nobn(opt.imageSize, nz, nc, ngf, ngpu, opt.G_extra_layers)
 elif opt.mlp_G:
     netG = mlp.MLP_G(opt.imageSize, nz, nc, ngf, ngpu)
 else:
-    netG = dcgan.DCGAN_G(opt.imageSize, nz, nc, ngf, ngpu, n_extra_layers)
+    netG = dcgan.DCGAN_G(opt.imageSize, nz, nc, ngf, ngpu, opt.G_extra_layers)

 netG.apply(weights_init)
 if opt.netG != '': # load checkpoint if needed
@@ -120,7 +121,7 @@ print(netG)
 if opt.mlp_D:
     netD = mlp.MLP_D(opt.imageSize, nz, nc, ndf, ngpu)
 else:
-    netD = dcgan.DCGAN_D(opt.imageSize, nz, nc, ndf, ngpu, n_extra_layers)
+    netD = dcgan.DCGAN_D(opt.imageSize, nz, nc, ndf, ngpu, opt.D_extra_layers)
     netD.apply(weights_init)

 if opt.netD != '':
@@ -132,6 +133,7 @@ noise = torch.FloatTensor(opt.batchSize, nz, 1, 1)
 fixed_noise = torch.FloatTensor(opt.batchSize, nz, 1, 1).normal_(0, 1)
 one = torch.FloatTensor([1])
 mone = one * -1
+alpha = torch.FloatTensor([0]) # lagrange multipliers

 if opt.cuda:
     netD.cuda()
@@ -139,14 +141,16 @@ if opt.cuda:
     input = input.cuda()
     one, mone = one.cuda(), mone.cuda()
     noise, fixed_noise = noise.cuda(), fixed_noise.cuda()
+    alpha = alpha.cuda()
+alpha = Variable(alpha, requires_grad=True)

 # setup optimizer
 if opt.adam:
-    optimizerD = optim.Adam(netD.parameters(), lr=opt.lrD, betas=(opt.beta1, 0.999))
-    optimizerG = optim.Adam(netG.parameters(), lr=opt.lrG, betas=(opt.beta1, 0.999))
+    optimizerD = optim.Adam(netD.parameters(), lr=opt.lrD, betas=(opt.beta1, 0.999), weight_decay=opt.wdecay)
+    optimizerG = optim.Adam(netG.parameters(), lr=opt.lrG, betas=(opt.beta1, 0.999), weight_decay=opt.wdecay)
 else:
-    optimizerD = optim.RMSprop(netD.parameters(), lr = opt.lrD)
-    optimizerG = optim.RMSprop(netG.parameters(), lr = opt.lrG)
+    optimizerD = optim.RMSprop(netD.parameters(), lr = opt.lrD, weight_decay=opt.wdecay)
+    optimizerG = optim.RMSprop(netG.parameters(), lr = opt.lrG, weight_decay=opt.wdecay)

 gen_iterations = 0
 for epoch in range(opt.niter):
@@ -160,7 +164,7 @@ for epoch in range(opt.niter):
         p.requires_grad = True # they are set to False below in netG update

     # train the discriminator Diters times
-    if gen_iterations < 25 or gen_iterations % 500 == 0:
+    if opt.hiDiterStart and (gen_iterations < 25 or gen_iterations % 500 == 0):
         Diters = 100
     else:
         Diters = opt.Diters
@@ -168,10 +172,6 @@ for epoch in range(opt.niter):
     while j < Diters and i < len(dataloader):
         j += 1

-        # clamp parameters to a cube
-        for p in netD.parameters():
-            p.data.clamp_(opt.clamp_lower, opt.clamp_upper)
-
         data = data_iter.next()
         i += 1

@@ -185,18 +185,28 @@ for epoch in range(opt.niter):
         input.resize_as_(real_cpu).copy_(real_cpu)
         inputv = Variable(input)

-        errD_real = netD(inputv)
-        errD_real.backward(one)
+        vphi_real = netD(inputv)

         # train with fake
         noise.resize_(opt.batchSize, nz, 1, 1).normal_(0, 1)
         noisev = Variable(noise, volatile = True) # totally freeze netG
```

```
          fake = Variable(netG(noisev).data)
          inputv = fake
-         errD_fake = netD(inputv)
-         errD_fake.backward(mone)
-         errD = errD_real - errD_fake
+
+         vphi_fake = netD(inputv)
+         # NOTE here f = <v,phi> , but with modified f the below two lines are the
+         # only ones that need change. E_P and E_Q refer to Expectation over real and fake.
+         E_P_f, E_Q_f = vphi_real.mean(), vphi_fake.mean()
+         E_P_f2, E_Q_f2 = (vphi_real**2).mean(), (vphi_fake**2).mean()
+         constraint = (1 - (0.5*E_P_f2 + 0.5*E_Q_f2))
+         # See Equation (9)
+         obj_D = E_P_f - E_Q_f + alpha * constraint - opt.rho/2 * constraint**2
+         # max_w min_alpha obj_D. Compute negative gradients, apply updates with negative sign.
+         obj_D.backward(mone)
          optimizerD.step()
+         # artisanal sgd. We minimze alpha so a <- a + lr * (-grad)
+         alpha.data += opt.rho * alpha.grad.data
+         alpha.grad.data.zero_()

       ############################
       # (2) Update G network
@@ -209,14 +219,20 @@ for epoch in range(opt.niter):
       noise.resize_(opt.batchSize, nz, 1, 1).normal_(0, 1)
       noisev = Variable(noise)
       fake = netG(noisev)
-      errG = netD(fake)
-      errG.backward(one)
+      vphi_fake = netD(fake)
+      obj_G = -vphi_fake.mean() # Just minimize mean difference
+      obj_G.backward() # G: min_theta
       optimizerG.step()
       gen_iterations += 1

-      print('[%d/%d][%d/%d][%d] Loss_D: %f Loss_G: %f Loss_D_real: %f Loss_D_fake %f'
+      IPM_enum = E_P_f.data[0] - E_Q_f.data[0]
+      IPM_denom = (0.5*E_P_f2.data[0] + 0.5*E_Q_f2.data[0]) ** 0.5
+      IPM_ratio = IPM_enum / IPM_denom
+      print(('[%d/%d][%d/%d][%d] IPM_enum: %.4f IPM_denom: %.4f IPM_ratio: %.4f '
+           'E_P_f: %.4f E_Q_f: %.4f E_P_(f^2): %.4f E_Q_(f^2): %.4f')
          % (epoch, opt.niter, i, len(dataloader), gen_iterations,
-         errD.data[0], errG.data[0], errD_real.data[0], errD_fake.data[0]))
+         IPM_enum, IPM_denom, IPM_ratio,
+         E_P_f.data[0], E_Q_f.data[0], E_P_f2.data[0], E_Q_f2.data[0]))
       if gen_iterations % 500 == 0:
          real_cpu = real_cpu.mul(0.5).add(0.5)
          vutils.save_image(real_cpu, '{0}/real_samples.png'.format(opt.experiment))
diff --git a/models/dcgan.py b/models/dcgan.py
index 1dd8dbf..ea86a94 100644
--- a/models/dcgan.py
+++ b/models/dcgan.py
@@ -48,9 +48,7 @@ class DCGAN_D(nn.Module):
          output = nn.parallel.data_parallel(self.main, input, range(self.ngpu))
       else:
          output = self.main(input)
-
-      output = output.mean(0)
-      return output.view(1)
+      return output.view(-1)

 class DCGAN_G(nn.Module):
    def __init__(self, isize, nz, nc, ngf, ngpu, n_extra_layers=0):
@@ -148,9 +146,7 @@ class DCGAN_D_nobn(nn.Module):
          output = nn.parallel.data_parallel(self.main, input, range(self.ngpu))
       else:
          output = self.main(input)
-
-      output = output.mean(0)
-      return output.view(1)
+      return output.view(-1)

 class DCGAN_G_nobn(nn.Module):
    def __init__(self, isize, nz, nc, ngf, ngpu, n_extra_layers=0):
```