On the Equivalence between Neural Network and Support Vector Machine

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent research shows that the dynamics of an infinitely wide neural network (NN) trained by gradient descent can be characterized by Neural Tangent Kernel (NTK) [20]. Under the squared loss, the infinite-width NN trained by gradient descent with an infinitely small learning rate is equivalent to kernel regression with NTK [3]. While the equivalence between NN and kernel regression has been established, the equivalence between NN and support vector machine (SVM) remains unknown. In this work, we establish the equivalence between NN and SVM, specifically, the infinitely wide NN trained by soft margin loss and the standard 1-norm soft margin SVM with NTK trained by subgradient descent. We show such NN and SVM have the same dynamics, characterized by an inhomogeneous linear differential (difference) equation, and thus converge to the same solution. We further generalize our theory to general loss functions with ℓ_2 regularization and show the equivalence between NN and a family of ℓ_2 regularized kernel machines, which previous results cannot handle. Additionally, we show every finite-width NN trained by such regularized loss functions is approximately a kernel machine. Finally, we demonstrate two practical applications of our theory: (i) it enables *nontrivial* robustness certificate for the infinite-width NN while existing robustness verification methods would fail; (ii) we could obtain infinite-width NNs that are intrinsically more robust than those from previous kernel regression.

1 Introduction

Recent research has made some progress towards deep learning theory from the perspective of infinite-width NN. For a fully-trained neural network, it follows kernel gradient descent in the function space with respect to the neural tangent kernel (NTK) [20]. Under this linear regime and squared loss, it is rigorously proved that the fully-trained net is equivalent to kernel regression with NTK [3], which gives the generalization ability of such a model [4]. NTK helps us understand the optimization [20, 14] and generalization [4, 9] of NN through the perspective of kernels. It also inherits the expressive power of a deep neural network without having to actually train one. [5] reported strong performance of NTK on small-data tasks both for kernel regression and kernel SVM. However, kernel methods lack the scalability to large problems compared to NN. Ideally, we want both the theoretical advantages of kernel methods and the practical advantages of NN.

On the other hand, another classical machine learning paradigm before the prevalence of deep neural networks is the support vector machine (SVM) [7, 11] which allows learning linear classifiers in high dimensional feature spaces. SVM tackles the sample complexity challenge by searching for large margin separators and tackles the computational complexity challenge using the idea of kernels [29]. To learn an SVM model, it usually involves solving a dual problem which is cast as a convex quadratic programming problem. Recently, there are some algorithms using subgradient descent [30]

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

and coordinate descent [18] to further scale the SVM models to large datasets and high dimensional feature spaces.

We noticed that existing theoretical analysis mostly focused on connecting NN with kernel regression [20, 3, 22] but the connections between NN and SVM have not yet been explored. In this work, we establish the equivalence between NN and SVM for the first time to our best knowledge. More broadly, we show that our analysis can connect NNs with a family of ℓ_2 regularized kernel machines, including kernel ridge regression (KRR) and ℓ_2 regularized logistic regression, where previous results [20, 3, 22] cannot handle. Importantly, the equivalence between infinite-width NN and SVM (and other ℓ_2 regularized kernel machines) could help us understand NN better from the perspective of classical kernel methods [12, 31, 28, 33], especially towards understanding the training, generalization, and robustness of NN. We highlight our contributions as follows:

- We derive the continuous (gradient flow) and discrete dynamics of SVM trained by subgradient descent and the dynamics of NN trained by soft margin loss. We show the dynamics of SVM with NTK and NN are exactly the same in the infinite width limit because of the constancy of the tangent kernel and thus establish the equivalence. We also verify the equivalence by experiments of subgradient descent and stochastic subgradient descent on MNIST dataset [21].

- We further generalize our theory to general loss functions with ℓ_2 regularization and establish the equivalence between NN and a family of ℓ_2 regularized kernel machines as summarized in Table 1. Additionally, we extend the results in [13] and show that every finite-width neural network trained by a ℓ_2 regularized loss function is approximately a kernel machine.

- We show that our theory offers two practical benefits: (i) we can deliver *nontrivial* robustness certificate for the over-parameterized NN (with width $m \to \infty$) while existing robustness verification methods would give trivial robustness certificate due to bound propagations [17, 36, 38]. In particular, the certificate decreases at a rate of $O(1/\sqrt{m})$ as the width of NN increases; (ii) we show that the equivalent infinite-width NNs trained from our ℓ_2 regularized kernel machines are more robust than the equivalent NN trained from previous kernel regression [20, 3] (see Table 3), which is perhaps not too surprising as the regularization has a strong connection to robust machine learning.

2 Related Works and Background

2.1 Related Works

Neural Tangent Kernel and dynamics of neural networks. NTK was first introduced in [20] and extended to Convolutional NTK [3] and Graph NTK [16]. [5] reported strong performance of NTK on small-data tasks both for kernel regression and kernel SVM. A line of recent work [15, 1] proved the convergence of (convolutional) neural networks with large but finite width in a non-asymptotic way by showing the weights do not move far away from initialization in the optimization dynamics (trajectory). [22] showed the dynamics of wide neural networks are governed by a linear model of first-order Taylor expansion around its initial parameters. Recently, [24] argued that the transition to linearity of the model is equivalent to the constancy of tangent kernel, and both result from the scaling properties of the Hessian matrix's norm of the network as a function of the network width. They also argued that the tangent kernel is associated with the model itself and does not depend on the optimization algorithm or the choice of loss function.

Neural network and support vector machine. Prior works [35, 34, 25, 32, 23] have explored the benefits of encouraging large margin in the context of deep networks. [10] introduced a new family of positive-definite kernel functions that mimic the computation in multilayer neural nets and applied the kernels into SVM. [13] showed that neural networks trained by gradient flow are approximately kernel machines with a new conceptual kernel named path kernel. [30] proposed a subgradient algorithm to solve the primal problem of SVM, which can obtain a solution of accuracy ϵ in $\tilde{O}(1/\epsilon)$ iterations. In this paper, we also consider the SVM trained by subgradient descent and connect it with NN trained by subgradient descent.

2.2 Neural Networks and Tangent Kernel

We consider a general form of deep neural network f with a linear output layer as [24]. Let $[L] = \{1, ..., L\}, \forall l \in [L]$,

$$\alpha^{(0)}(w, x) = x,\ \alpha^{(l)}(w, x) = \phi_l(w^{(l)}, \alpha^{(l-1)}),\ f(w, x) = \frac{1}{\sqrt{m_L}} \left\langle w^{(L+1)}, \alpha^{(L)}(w, x) \right\rangle, \quad (1)$$

where each vector-valued function $\phi_l(w^{(l)}, \cdot) : \mathbb{R}^{m_{l-1}} \to \mathbb{R}^{m_l}$, with parameter $w^{(l)} \in \mathbb{R}^{p_l}$ (p_l is the number of parameters), is considered as a layer of the network. This definition includes the standard fully connected, convolutional (CNN), and residual (ResNet) neural networks as special cases. For a fully connected ReLU NN, $\alpha^{(l)}(w, x) = \sigma(\frac{1}{\sqrt{m_{l-1}}} w^{(l)} \alpha^{(l-1)})$ with $w^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}}$ and $\sigma(z) = \max(0, z)$.

Initialization and parameterization. In this paper, we consider the NTK parameterization [20], under which the constancy of the tangent kernel has been initially observed. Specifically, the parameters, $w := \{w^{(1)}; w^{(2)}; \cdots; w^{(L)}; w^{(L+1)}\}$ are drawn i.i.d. from a standard Gaussian, $\mathcal{N}(0, 1)$, at initialization, denoted as w_0. The factor $1/\sqrt{m_L}$ in the output layer is required by the NTK parameterization in order that the output f is of order $O(1)$. While we only consider NTK parameterization here, the results should be able to extend to general parameterization of kernel regime [37].

Definition 2.1 (tangent kernel)**.** The tangent kernel associated with function $f(w, x)$ at some parameter w is $\hat{\Theta}(w; x, x') = \langle \nabla_w f(w, x), \nabla_w f(w, x') \rangle$. Under certain conditions (usually infinite width limit and NTK parameterization), the tangent kernel at initialization converges in probability to a deterministic limit and keeps constant during training, $\hat{\Theta}(w; x, x') \to \Theta(x, x')$. This limiting kernel is called *Neural Tangent Kernel (NTK)*.

2.3 Kernel Machines

Kernel machine is a model of the form $g(\beta, x) = \varphi(\langle \beta, \Phi(x) \rangle + b)$, where β is the model parameter and Φ is a mapping from input space to feature space, $\Phi : \mathcal{X} \to \mathcal{F}$. φ is an optional nonlinear function, such as identity mapping for kernel regression and $sign(\cdot)$ for SVM and logistic regression. The kernel can be exploited whenever the weight vector can be expressed as a linear combination of the training points, $\beta = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$ for some value of α_i, $i \in [n]$, implying that we can express g as $g(x) = \varphi(\sum_{i=1}^{n} \alpha_i K(x, x_i) + b)$, where $K(x, x_i) = \langle \Phi(x), \Phi(x_i) \rangle$ is the kernel function. For a neural network in NTK regime, we have $f(w_t, x) \approx f(w_0, x) + \langle \nabla_w f(w_0, x), w_t - w_0 \rangle$, which makes the neural network linear in the gradient feature mapping $x \to \nabla_w f(w_0, x)$. Under squared loss, it is equivalent to kernel regression with a kernel machine that has $\Phi(x) = \nabla_w f(w_0, x)$ (or equivalently using NTK as the kernel), $\beta = w_t - w_0$ and φ identity mapping [3].

As far as we know, there is no work establishing the equivalence between fully trained networks and SVM. [13] showed that neural networks trained by gradient flow are approximately kernel machines, but didn't discuss any specific kernel machine. In this work, we compare the dynamics of SVM and neural network trained by subgradient descent with soft margin loss and showed the equivalence between them in the infinite width limit.

2.4 Subgradient Optimization of Support Vector Machine

We first formally define the 1-norm soft margin SVM and then show how the subgradient descent can be applied to get an estimation of the SVM primal problem. For simplicity, we consider the homogenous model, $g(\beta, x) = \langle \beta, \Phi(x) \rangle$.[1]

Definition 2.2 (1-norm Soft Margin SVM)**.** Given labeled samples $\{(x_i, y_i)\}_{i=1}^{n}$ with $y_i \in \{-1, +1\}$, the hyperplane β^* that solves the optimization problem

$$\min_{\beta, \xi} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \xi_i, \quad s.t.\ y_i \langle \beta, \Phi(x_i) \rangle \geq 1 - \xi_i,\ \xi_i \geq 0,\ i \in [n],$$

realizes the 1-norm soft margin classifier with geometric margin $\gamma = 2/\|\beta^*\|$. This margin is also called the 1-norm soft margin.

[1] Note one can always deal with the bias term b by adding each sample with an additional dimension, $\Phi(x)^T \leftarrow [\Phi(x)^T, 1], \beta^T \leftarrow [\beta^T, 1]$.

Proposition 2.1. *The above primal problem of 1-norm soft margin SVM can be equivalently formulated as*

$$\min_{\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i \langle \beta, \Phi(x_i)\rangle), \tag{2}$$

where the second term is a hinge loss. This function is strongly convex for β.

From this, we see that the SVM technique is equivalent to empirical risk minimization with ℓ_2 regularization, where in this case the loss function is the nonsmooth hinge loss. The classical approaches usually consider the dual problem of SVM and solve it as a quadratic programming problem. Some recent algorithms, however, use subgradient descent [30] to optimize Eq. (2), which shows significant advantages when dealing with large datasets.

In this paper, we consider the 1-norm soft margin SVM trained by subgradient descent with loss function (2). Denote the objective function as $L(\beta)$. Define the subgradient as $\nabla_\beta L(\beta) = \beta - C\sum_{i=1}^{n} \mathbb{1}(y_i g(\beta, x_i) < 1) y_i \Phi(x_i)$, where $\mathbb{1}(\cdot)$ is the indicator function. The decision boundary of SVM is $g(\beta, x) = 0$ and $g(\beta, x) = \pm 1$ are the edges of the margin. The subgradient is composed by the samples on the wrong side of the margin. As proved in [30], we can find a solution of accuracy ϵ, i.e. $L(\beta) - L(\beta^*) \leq \epsilon$, in $\tilde{O}(1/\epsilon)$ iterations, where $\tilde{O}$ omits the logarithmic factors. And other works also give convergence guarantees for subgradient descent of convex functions [8, 6]. In the following analysis, we will assume the convergence of SVM trained by subgradient descent.

3 Main Theoretical Results

In this section, we describe our main results. We first derive the continuous (gradient flow) and discrete dynamics of SVM trained by subgradient descent (in Section 3.1) and the dynamics of NN trained by soft margin loss (in Section 3.2 and Section 3.3). we show that they have similar dynamics, characterized by an inhomogeneous linear differential (difference) equation. Next, we show that their dynamics are exactly the same in the infinite width limit because of the constancy of tangent kernel and thus establish the equivalence (Theorem 3.4). Furthermore, in Section 3.4, we generalize our theory to general loss functions with ℓ_2 regularization and establish the equivalence between NN and a family of ℓ_2 regularized kernel machines as summarized in Table 1.

3.1 Dynamics of 1-norm Soft Margin SVM

We consider the dynamics of 1-norm soft margin SVM trained by subgradient. For simpicity, we denote β_t as β at some time t and $g_t(x) = g(\beta_t, x)$. The proofs of the following two theorems are detailed in Appendix B.

Theorem 3.1 (Continuous Dynamics of SVM)**.** *Consider training 1-norm soft margin SVM by subgradient descent with infinite small learning rate (gradient flow [2]): $\frac{d\beta_t}{dt} = -\nabla_\beta L(\beta_t)$, the model $g_t(x)$ follows the below evolution:*

$$\frac{dg_t(x)}{dt} = -g_t(x) + C\sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i). \tag{3}$$

Denote $Q(t) = C\sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i)$, which changes over time until convergence. The model output $g_t(x)$ at some time T is

$$g_T(x) = e^{-T}\left(g_0(x) + \int_0^T Q(t)e^t\, dt\right), \quad \lim_{T\to\infty} g_T(x) = C\sum_{i=1}^{n} \mathbb{1}(y_i g_T(x_i) < 1) y_i K(x, x_i). \tag{4}$$

We see that the continuous dynamics of SVM trained by subgradient descent is described by an inhomogeneous linear differential equation (Eq. (3)). The solution is given by the general solution of the differential equation. As $T \to \infty$, the dynamics converge to the right solution. Later we will show NN has a similar dynamic as SVM and even the same in the case of constant tangent kernel.

Remark 3.1. From Eq. (4), we can see that the influence of initial model $g_0(x)$ deceases as time $T \to \infty$ and disappears at last. Also Interestingly, when Eq. (3) $\frac{dg_t(x)}{dt} = 0$, i.e. $g_t(x)$ does not change, we get the solution of $T \to \infty$ (Eq. (4)).

Theorem 3.2 (Discrete Dynamics of SVM)**.** *Let $\eta \in (0,1)$ be the learning rate. The dynamics of subgradient descent is*

$$g_{t+1}(x) - g_t(x) = -\eta g_t(x) + \eta C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i). \tag{5}$$

Denote $Q(t) = \eta C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i)$, which changes over time. The model output $g_t(x)$ at some time T is

$$g_T(x) = (1-\eta)^T \left(g_0(x) + \sum_{t=0}^{T-1} (1-\eta)^{-t-1} Q(t) \right), \lim_{T\to\infty} g_T(x) = C \sum_{i=1}^{n} \mathbb{1}(y_i g_T(x_i) < 1) y_i K(x, x_i).$$

The discrete dynamics are characterized by an inhomogeneous linear difference equation (Eq. (5)), and the discrete dynamics and solution of SVM have similar structures as the continuous case.

3.2 1-norm Soft Margin Neural Network

We first formally define the 1-norm soft margin neural network and then derive the dynamics of training a neural network by subgradient descent with soft margin loss. We will consider a neural network defined as Eq. (1). For convenience, we redefine $f(w,x) = \left\langle W^{(L+1)}, \alpha^{(L)}(w,x) \right\rangle$ with $W^{(L+1)} = \frac{1}{\sqrt{m_L}} w^{(L+1)}$ and $w := \{w^{(1)}; w^{(2)}; \cdots; w^{(L)}; W^{(L+1)}\}$.

Definition 3.1 (1-norm Soft Margin Neural Network)**.** Given samples $\{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{-1, +1\}$, the neural network defined as Eq. (1) that solves the following equivalent optimization problems

$$\min_{w,\xi} \frac{1}{2} \|W^{(L+1)}\|^2 + C \sum_{i=1}^{n} \xi_i, \quad s.t.\ y_i f(w, x_i) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i \in [n],$$

or equivalently

$$\min_{w} \frac{1}{2} \|W^{(L+1)}\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f(w, x_i)), \tag{6}$$

realizes the 1-norm soft margin classifier with geometric margin $\gamma = 2/\|W^{(L+1)}\|$. We denote Eq. (6) as $L(w)$ and call it soft margin loss.

This is generally a hard nonconvex optimization problem, but we can apply subgradient descent to optimize it heuristically. At initilization, $\|W_0^{(L+1)}\|^2 = \|w_0^{(L+1)}\|^2/m_L = O(1)$ and keeps $O(1)$ as $m \to \infty$. The derivative of $\|W^{(L+1)}\|^2/2$ for $w^{(L+1)}$ is $w^{(L+1)}/\sqrt{m_L} = O(1/\sqrt{m_L}) \to 0$.

Remark 3.2. For a fixed $\alpha^{(L)}(w,x)$, it is same as SVM with $\Phi(x) = \alpha^{(L)}(w,x)$, kernel $K(x,x') = \alpha^{(L)}(w,x) \cdot \alpha^{(L)}(w,x')$ and parameter $\beta = W^{(L+1)}$. But during training, $\alpha^{(L)}(w,x)$ is changing over time, which causes the kernel is changing over time.

3.3 Dynamics of Neural Network Trained by Soft Margin Loss

Denote the hinge loss in $L(w)$ as $L_h(y_i, f(w,x_i)) = C \sum_{i=1}^{n} \max(0, 1 - y_i f(w, x_i))$. Note the hinge loss is not differentiable at $y_i f(w, x_i) = 1$, but we can use the same subgradient as that for SVM, $L_h'(y_i, f(w,x_i)) = -C y_i \mathbb{1}(y_i f(w,x_i) < 1)$.

Theorem 3.3 (Continuous Dynamics of NN)**.** *Suppose an NN $f(w,x)$ defined as Eq. (1), with f a differentiable function of w, is learned from a training set $\{(x_i, y_i)\}_{i=1}^n$ by subgradient descent with soft margin loss (Eq. (6)) and learning rate $\eta \to 0$. Then the network has the following dynamics:*

$$\frac{df_t(x)}{dt} = -f_t(x) + C \sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i) < 1) y_i \hat{\Theta}(w_t; x, x_i).$$

See the full proof in Appendix C. The key observation is that when deriving the dynamics of $f_t(x)$, the $\frac{1}{2}\|W^{(L+1)}\|^2$ term in the loss function will produce a $f_t(x)$ term and the hinge loss will produce the tangent kernel term, which overall gives a similar dynamics to that of SVM. Comparing to the

Table 1: Summary of our theoretical results on the equivalence between infinite-width NNs and a family of kernel machines. Thanks to the representer theorem [28], our ℓ_2 regularized kernel machines can all apply kernel trick, meaning infinite NTK can be applied in these ℓ_2 regularized kernel machines.

λ	Loss $l(z, y_i)$	Kernel machine
$\lambda = 0$ ([20, 3])	$(y_i - z)^2$	Kernel regression
$\lambda \to 0$ (ours)	$\max(0, 1 - y_i z)$	Hard margin SVM
$\lambda > 0$ (ours)	$\max(0, 1 - y_i z)$	1-norm soft margin SVM
	$\max(0, 1 - y_i z)^2$	2-norm soft margin SVM
	$\max(0, \lvert y_i - z \rvert - \epsilon)$	Support vector regression
	$(y_i - z)^2$	Kernel ridge regression (KRR)
	$\log(1 + e^{-y_i z})$	Logistic regression with ℓ_2 regularization
	$\log(1 + \sum_{r \neq y_i} e^{z_r - z_{y_i}})$	Multiclass logistic regression with ℓ_2 regularization

previous continuous kernel gradient descent [20, 22], our result has an extra $-f_t(x)$ here because of the regularization term of the loss function.

If the tangent kernel $\hat{\Theta}(w_t; x, x_i)$ is fixed, $\hat{\Theta}(w_t; x, x_i) \to \Theta(x, x_i)$, the dynamics of NN is the same as that of SVM (Eq. (3)) with kernel $\Theta(x, x_i)$, assuming the neural network and SVM have same initial output $g_0(x) = f_0(x)$. [2] And this consistency of tangent kernel is the case for infinitely wide neural networks of common architectures, which does not depend on optimization algorithm and the choice of loss function, as discussed in [24].

Assumptions. We assume that (vector-valued) layer functions $\phi_l(w, \alpha), l \in [L]$ are L_ϕ-Lipschitz continuous and twice differentiable with respect to input α and parameters w. The assumptions serve for the following theorem to show the constancy of tangent kernel.

Theorem 3.4 (Equivalence between NN and SVM)**.** *As the minimum width of the NN, defined as Eq. (1), goes to infinity, the tangent kernel tends to be constant, $\hat{\Theta}(w_t; x, x_i) \to \Theta(x, x_i)$. Assume the NN and SVM have same initial output $g_0(x) = f_0(x)$. Then the infinitely wide NN trained by subgradient descent with soft margin loss has the same dynamics as SVM with constant NTK trained by subgradient descent:*

$$\frac{df_t(x)}{dt} = -f_t(x) + C \sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i) < 1) y_i \Theta(x, x_i).$$

And thus such NN and SVM converge to the same solution.

The proof is in Appendix D. We apply the results of [24] to show the constancy of tangent kernel in the infinite width limit. Then it is easy to check the dynamics of infinitely wide NN and SVM with NTK are the same. This theorem establishes the equivalence between infinitely wide NN and SVM for the first time. Previous theoretical results of SVM [12, 31, 28, 33] can be directly applied to understand the generalization of NN trained by soft margin loss. Given the tangent kernel is constant or equivalently the model is linear, we can also give the discrete dynamics of NN. See Appendix C.2 for the derivation. This discrete dynamics is also identical to that of SVM with NTK. compared with the previous discrete kernel gradient descent [22, 37], our result has an extra $-\eta f_t(x)$ term because of the regularization term of loss function.

$$f_{t+1}(x) - f_t(x) = -\eta f_t(x) + \eta C \sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i) < 1) y_i \Theta(x, x_i).$$

3.4 General Loss Functions

We note that above analysis does not have specific dependence on the hinge loss. Thus we can generalize our analysis to general loss functions $l(z, y_i)$, where z is the model output, as long as the

[2] This can be done by setting the initial values to be 0, i.e. $g_0(x) = f_0(x) = 0$.

loss function is differentiable (or has subgradients) with respect to z, such as squared loss and logistic loss. And we can scale the regularization term by a factor λ instead of scaling $l(z, y_i)$ with C as it for SVM, which are equivalent. Suppose the loss function for the kernel machine $g(\beta, x) = \langle \beta, \Phi(x) \rangle$ and $f(w, x) = \left\langle W^{(L+1)}, \alpha^{(L)}(w, x) \right\rangle$ are

$$L(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \sum_{i=1}^{n} l(g(\beta, x_i), y_i), \quad L(w) = \frac{\lambda}{2}\|W^{(L+1)}\|^2 + \sum_{i=1}^{n} l(f(w, x_i), y_i). \tag{7}$$

Then the continuous dynamics of $g_t(x)$ and $f_t(x)$ are

$$\frac{dg_t(x)}{dt} = -\lambda g_t(x) - \sum_{i=1}^{n} l'(g_t(x_i), y_i) K(x, x_i), \tag{8}$$

$$\frac{df_t(x)}{dt} = -\lambda f_t(x) - \sum_{i=1}^{n} l'(f_t(x_i), y_i) \hat{\Theta}(w_t; x, x_i), \tag{9}$$

where $l'(z, y_i) = \frac{\partial l(z, y_i)}{\partial z}$. In the situation of $\hat{\Theta}(w_t; x, x_i) \to \Theta(x, x_i)$ and $K(x, x_i) = \Theta(x, x_i)$, these two dynamics are the same (assuming $g_0(x) = f_0(x)$). When $\lambda = 0$, we recover the previous results of kernel regression. When $\lambda > 0$, we have our new results of ℓ_2 regularized loss functions. Table 1 lists the different loss functions and the corresponding kernel machines that infinite-width NNs are equivalent to. Note KRR is considered in [19] to analyze the generalization of NN trained with regularization term. However, they directly assume NN as a linear model and use it in KRR.

4 Discussion

In this section, we give some extensions and applications of our theory. We first extend the results in [13] and show that every finite-width neural network trained by a ℓ_2 regularized loss function is approximately a kernel machine in Section 4.1. Next, in Section 4.2, we show that our theory of equivalence (in Section 3.3) is useful to evaluating the robustness of over-parameterized NNs with infinite width. In particular, our theory allows us to deliver nontrivial robustness certificates for an infinite-width NN, while existing robustness verification methods [17, 36, 38] would become much looser (decrease at a rate of $O(1/\sqrt{m})$, where m is the width of NN) when the width of NN increases and trivial with infinite width (the experiment results are in Section 5 and Table 2).

4.1 Finite-width Neural Network Trained by ℓ_2 Regularized Loss

Inspired by [13], we can also show that every NN trained by (sub)gradient descent with loss function Eq. (7) is approximately a kernel machine without the assumption of infinite width limit.

Theorem 4.1. *Suppose an NN $f(w, x)$, is learned from a training set $\{(x_i, y_i)\}_{i=1}^n$ by (sub)gradient descent with loss function Eq. (7) and learning rate $\eta \to 0$. Then at some time T,*

$$\lim_{\eta \to 0} f_T(x) = \sum_{i=1}^{n} a_i K(x, x_i) + b, \quad K(x, x_i) = e^{-T} \int_0^T \hat{\Theta}(w_t; x, x_i) e^t \, dt,$$

$$a_i = -\frac{\int l'(y_i, f_t(x_i)) \hat{\Theta}(w_t; x, x_i) e^t \, dt}{\int \hat{\Theta}(w_t; x, x_i) e^t \, dt}, \qquad b = e^{-T} f_0(x).$$

See the proof in Appendix E. The proof is similar to [13] while we utilize the general solution of inhomogeneous linear differential equation instead of integrating both side of dynamics (Eq. (9)) directly. In [13], they introduced a new conceptual kernel, path kernel, $K(x, x_i) = \int_0^T \hat{\Theta}(w_t; x, x_i) \, dt$, integrated over the path taken by the parameters during gradient descent. But this path kernel may diverge as $T \to \infty$. Compared with their results, our a_i, b and $K(x, x_i)$ will stay bounded as long as $f_0(x)$, $l'(y_i, f_t(x_i))$ and $\hat{\Theta}(w_t; x, x_i)$ are bounded. Note $K(x, x_i) = e^{-T} \int_0^T \hat{\Theta}(w_t; x, x_i) e^t \, dt$ is a valid kernel since it is a nonnegative sum of positive semidefinite kernels. Interestingly, in the situation of $\hat{\Theta}(w_t; x, x_i) \to \Theta(x, x_i)$, $K(x, x_i) = e^{-T}(e^T - 1)\Theta(x, x_i) \to \Theta(x, x_i)$ as $T \to \infty$.

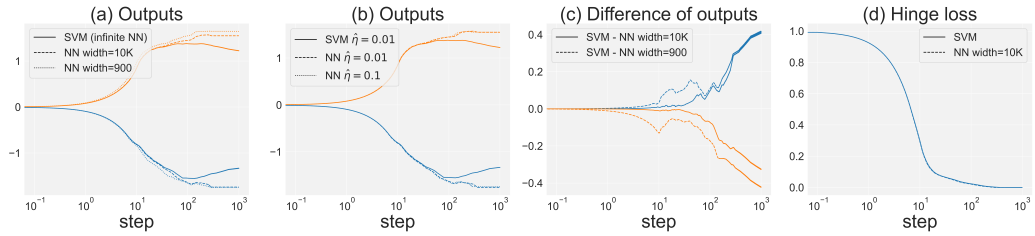

Figure 1: Training dynamics of neural network and SVM behave similarly. (a)(b) show dynamics of outputs for randomly selected two samples. (c) shows the difference between the outputs of SVM and NN. (d) shows the dynamics of hinge loss for SVM and NN. Without specification, the width of NN is 10K and $\hat{\eta} = 0.1$. The dynamics of SVM agrees better with wider NN.

4.2 Robustness of Infinite-width Neural Network

Our theory of equivalence allows us to deliver nontrivial robustness certificates for an infinite-width NN by considering the equivalent kernel machines. For an input $x_0 \in \mathbb{R}^d$, the objective of robustness is to find the largest ball such that no examples within this ball $x \in B(x_0, \delta)$ can change the classification result. Without loss of generality, we assume $g(x_0) > 0$. The robustness problem can be formulated as follows,

$$\max\ \delta, \quad \text{s.t. } g(x) > 0,\ \forall x \in B(x_0, \delta).$$

For an infinitely wide two-layer fully connected ReLU NN, $f(x) = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} v_j \sigma(\frac{1}{\sqrt{d}} w_j^T x)$, where $\sigma(z) = \max(0, z)$ is the ReLU activation. The NTK is

$$\Theta(x, x') = \frac{\langle x, x' \rangle}{d} \left(\frac{\pi - \arccos(u)}{\pi}\right) + \frac{\|x\|\|x'\|}{2\pi d} \sqrt{1 - u^2}$$

where $u = \frac{\langle x, x' \rangle}{\|x\|\|x'\|} \in [-1, 1]$. See the proof of the following theorem in Appendix F.1.

Theorem 4.2. *Consider the ℓ_∞ perturbation, for $x \in B_\infty(x_0, \delta) = \{x \in \mathbb{R}^d : \|x - x_0\|_\infty \leq \delta\}$, we can bound $\Theta(x, x')$ into some interval $[\Theta^L(x, x'), \Theta^U(x, x')]$. Suppose the $g(x) = \sum_{i=1}^{n} \alpha_i \Theta(x, x_i)$, where α_i are known after solving the kernel machine problems (e.g. SVM and KRR). Then we can lower bound $g(x)$ as follows.*

$$g(x) \geq \sum_{i=1, \alpha_i > 0}^{n} \alpha_i \Theta^L(x, x_i) + \sum_{i=1, \alpha_i < 0}^{n} \alpha_i \Theta^U(x, x_i).$$

5 Experiments

(I) Verification of the equivalence. The first experiment verifies the equivalence between 1-norm soft margin SVM with NTK trained by subgradient descent and NN trained by soft margin loss. We train the SVM and 3-layer fully connected ReLU NN for a binary MNIST [21] classification (0 and 1) with learning rate $\hat{\eta} = 0.1$ and $\hat{\eta} = 0.01$ with full batch subgradient descent on $n = 128$ samples, where $\hat{\eta}$ is the learning rate used in experiments. Figure 1 shows the dynamics of the outputs and loss for NN and SVM. Since the regularization terms in the loss of NN and SVM are different, we just plot the hinge loss. It can be seen that the dynamics of NN and SVM agree well. We also do a stochastic subgradient descent case for binary classification on full MNIST 0 and 1 data (12665 training and 2115 test) with learning rate $\hat{\eta} = 1$ and batch size 64, shown in Figure A.1. For more details, please see Appendix A.

(II) Robustness of over-parameterized neural network. Table 2 shows the robustness of two-layer overparameterized NNs with increasing width and SVM (which is equivalent to infinite-width two-layer ReLU NN) on binary classification of MNIST (0 and 1). We use the NN robustness verification algorithm (IBP) [17] to compute the robustness certificate for two-layer overparameterized NNs. The robustness certificate for SVM is computed using our method in Section 4.2. As demonstrated

in Table 2, the certificate of NN almost decrease at a rate of $O(1/\sqrt{m})$ and will decrease to 0 as $m \to \infty$, where m is the width of the hidden layer. We show that this is due to the bound propagation and our analysis is provided in Appendix F.2. Unfortunately, the decrease rate will be faster if the NN is deeper. The same problem will happen for LeCun initialization as well, which is used in PyTorch for fully connected layers by default. Notably, however, thanks to our theory, we could compute *nontrivial* robustness certificate for an infinite-width NN through the equivalent SVM as demonstrated.

Table 2: Robustness lower bounds of two-layer ReLU NN and SVM (infinite-width two-layer ReLU NN) tested on binary classification of MNIST (0 and 1). 100 test: randomly selected 100 test samples. Full test: full test data. Test only on data that classified correctly. std is computed over data samples. All models have test accuracy 99.95%. All values are mean of 5 experiments.

		Robustness certificate δ (mean $\pm$ std) $\times 10^{-3}$	
Model	Width	100 test	Full test
NN	10^3	7.4485 $\pm$ 2.5667	7.2708 $\pm$ 2.1427
NN	10^4	2.9861 $\pm$ 1.0730	2.9367 $\pm$ 0.89807
NN	10^5	0.99098 $\pm$ 0.35775	0.97410 $\pm$ 0.29997
NN	10^6	0.31539 $\pm$ 0.11380	0.30997 $\pm$ 0.095467
SVM	∞	8.0541 $\pm$ 2.5827	7.9733 $\pm$ 2.1396

(III) Comparison with kernel regression. Table 3 compares our ℓ_2 regularized models (KRR and SVM with NTK) with the previous kernel regression model ($\lambda = 0$ for KRR). All the robustness lower bounds are computed using our method in Section 4.2. While the accuracies of different models are similar, as the regularization increases, the robustness of KRR increases. The robustness of SVM outperforms the KRR with same regularization magnitude a lot. Our theory enables us to train an equivalent infinite-width NN through SVM and KRR, which is intrinsically more robust than the previous kernel regression model.

Table 3: Robustness of equivalent infinite-width NN models with different loss functions (see Table 1) on binary classification of MNIST (0 and 1). λ is the parameter in Eq. (7).

	Model	λ	Test accuracy	Robustness certificate δ	Robustness improvement
$\lambda = 0$ [20, 3]	KRR	0	99.95%	3.30202×10^{-5}	-
$\lambda > 0$ (ours)	KRR	0.001	99.95%	3.756122×10^{-5}	1.14X
	KRR	0.01	99.95%	6.505500×10^{-5}	1.97X
	KRR	0.1	99.95%	2.229960×10^{-4}	6.75X
	KRR	1	99.95%	0.001005	30.43X
	KRR	10	99.91%	0.005181	156.90X
	KRR	100	99.86%	0.020456	619.50X
	KRR	1000	99.76%	0.026088	790.06X
	SVM	0.032	99.95%	0.008054	243.91X

6 Conclusion and Future Works

In this paper, we establish the equivalence between SVM with NTK and the neural network trained by soft margin loss with subgradient descent in the infinite width limit, and we show that they have the same dynamics and solution. We also extend our analysis to general ℓ_2 regularized loss functions and show every neural network trained by such loss functions is approximately a kernel machine. Finally, we demonstrate our theory is useful to compute *non-trivial* robustness certificate for infinite-width NN while existing neural network robustness verification algorithm cannot, and with our theory, the resulting infinite-width NN from our ℓ_2 regularized models is intrinsically more robust than that from the previous NTK kernel regression. For future research, since the equivalence between NN and SVM (and other ℓ_2 regularized kernel machines) with NTK has been established, it would be very interesting to understand the generalization and robustness of NN from the perspective of these kernel machines. Our main results are currently still limited in the linear regime. It would be interesting to extend the results to the mean field setting or consider its connection with the implicit bias of NN.

References

[1] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[2] L. Ambrosio, N. Gigli, and G. Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[3] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150, 2019.

[4] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

[5] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. *arXiv preprint arXiv:1910.01663*, 2019.

[6] D. P. Bertsekas and A. Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[7] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[8] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

[9] Y. Cao and Q. Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *arXiv preprint arXiv:1905.13210*, 2019.

[10] Y. Cho. *Kernel methods for deep learning*. PhD thesis, UC San Diego, 2012.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[12] N. Cristianini, J. Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[13] P. Domingos. Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*, 2020.

[14] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[15] S. S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[16] S. S. Du, K. Hou, B. Póczos, R. Salakhutdinov, R. Wang, and K. Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *arXiv preprint arXiv:1905.13192*, 2019.

[17] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[18] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415, 2008.

[19] W. Hu, Z. Li, and D. Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. *arXiv preprint arXiv:1905.11368*, 2019.

[20] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8572–8583, 2019.

[23] X. Liang, X. Wang, Z. Lei, S. Liao, and S. Z. Li. Soft-margin softmax for deep classification. In *International Conference on Neural Information Processing*, pages 413–421. Springer, 2017.

[24] C. Liu, L. Zhu, and M. Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33, 2020.

[25] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7, 2016.

[26] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://github.com/google/neural-tangents`.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[28] B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[29] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[30] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[31] J. Shawe-Taylor, N. Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[32] J. Sokolić, R. Giryes, G. Sapiro, and M. R. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

[33] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[34] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu. On the depth of deep neural networks: A theoretical view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[35] Y. Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.

[36] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018.

[37] G. Yang and E. J. Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

[38] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. *arXiv preprint arXiv:1811.00866*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See for example Section 3.3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See the proofs in Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 5 and Appendix A.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5. We report mean value of multiple experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]