On the Equivalence between Neural Network and Support Vector Machine

Yilan Chen
Computer Science and Engineering
University of California San Diego
La Jolla, CA
yilan@ucsd.edu

Wei Huang
Engineering and Information Technology
University of Technology Sydney
Ultimo, Australia
weihuang.uts@gmail.com

Lam M. Nguyen
IBM Research
Thomas J. Watson Research Center
Yorktown Heights, NY
LamNguyen.MLTD@ibm.com

Tsui-Wei Weng
Halıcıoğlu Data Science Institute
University of California San Diego
La Jolla, CA
lweng@ucsd.edu

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) [27]. 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 [4]. However, the equivalence is only known for ridge regression currently [6], while the equivalence between NN and other kernel machines (KMs), e.g. support vector machine (SVM), remains unknown. Therefore, in this work, we propose to establish the equivalence between NN and SVM, and specifically, the infinitely wide NN trained by soft margin loss and the standard soft margin SVM with NTK trained by subgradient descent. Our main theoretical results include establishing the equivalence between NN and a broad family of ℓ_2 regularized KMs with finite-width bounds, which cannot be handled by prior work, and showing that every finite-width NN trained by such regularized loss functions is approximately a KM. Furthermore, we demonstrate our theory can enable three practical applications, including (i) *non-vacuous* generalization bound of NN via the corresponding KM; (ii) *nontrivial* robustness certificate for the infinite-width NN (while existing robustness verification methods would provide vacuous bounds); (iii) intrinsically more robust infinite-width NNs 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 NTK [27]. Under this linear regime and squared loss, it is rigorously proved that the fully-trained net is equivalent to kernel regression with NTK [4], which gives the generalization ability of such a model [5]. NTK helps us understand the optimization [27, 18] and generalization [5, 12] of NN through the perspective of kernels. However, existing theories about NTK [27, 30, 4, 13] usually assume the loss is a function of the model output, which does not include the case of regularization. Besides, they usually consider the squared loss which corresponds to a kernel regression, which may have limited insights to understand classification problems since squared loss and kernel regression are usually used for regression problems.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

On the other hand, another popular machine learning paradigm with solid theoretical foundation before the prevalence of deep neural networks is the support vector machine (SVM) [10, 15], 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 [43]. 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 [44] and coordinate descent [23] 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 [27, 4, 30] 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 KMs, including kernel ridge regression (KRR), support vector regression (SVR) and ℓ_2 regularized logistic regression, where previous results [27, 4, 30] cannot handle. These are the equivalences beyond ridge regression for the first time. Importantly, the equivalence between infinite-width NN and these ℓ_2 regularized KMs may shed light on the understanding of NN from these new equivalent KMs [16, 45, 42, 48], especially towards understanding the training, generalization, and robustness of NN for classification problems. Besides, regularization plays an important role in machine learning to restrict the complexity of models. This equivalence may shed light on the understanding of the regularization for 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 show same linear convergence rate of SVM and NN under reasonable assumption. We verify the equivalence by experiments of subgradient descent and stochastic subgradient descent on MNIST dataset [28].
- 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 KMs as summarized in Table 1. We prove the difference between the outputs of SVM and NN sacles as $O(\ln m/\lambda\sqrt{m})$, where λ is the coefficient of the regularization and m is the width of the NN. Additionally, we show every finite-width neural network trained by a ℓ_2 regularized loss function is approximately a KM.
- We show that our theory offers three practical benefits: (i) computing *non-vacuous* generalization bound of NN via the corresponding KM; (ii) 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 propagation [22, 52, 55]. In particular, the certificate decreases at a rate of $O(1/\sqrt{m})$ as the width of NN increases; (iii) we show that the equivalent infinite-width NNs trained from our ℓ_2 regularized KMs are more robust than the equivalent NN trained from previous kernel regression [27, 4] (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 [27] and extended to Convolutional NTK [4] and Graph NTK [20]. [26] studied the NTK of orthogonal initialization. [6] reported strong performance of NTK on small-data tasks both for kernel regression and kernel SVM. However, the equivalence is only known for ridge regression currently, but not for SVM and other KMs. A line of recent work [19, 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). [30] showed the dynamics of wide neural networks are governed by a linear model of first-order Taylor expansion around its initial parameters. However, existing theory about NTK [27, 30, 4] usually assume the loss is a function of the model output, which does not include the case of regularization. Besides, they usually consider the squared loss which corresponds to a kernel regression, which may have limited insights to understand classification problems since squared loss and kernel regression are usually used for regression problems. In this paper, we study the regularized loss functions and establish the equivalence with KMs beyond kernel regression and regression problems.

Besides, we studied the robustness of NTK models. [24] studied the label noise (the labels are generated by a ground truth function plus a Gaussian noise) while we consider the robustness of input perturbation. They study the convergence rate of NN trained by ℓ_2 regularized squared loss to an underlying true function, while we give explicit robustness certificates for NNs. Our robustness certificate enables us to compare different models and show the equivalent infinite-width NNs trained from our ℓ_2 regularized KMs are more robust than the equivalent NN trained from previous kernel regression.

Neural network and support vector machine. Prior works [50, 49, 34, 47, 31] have explored the benefits of encouraging large margin in the context of deep networks. [14] introduced a new family of positive-definite kernel functions that mimic the computation in multilayer neural nets and applied the kernels into SVM. [17] showed that neural networks trained by gradient flow are approximately KMs with a new conceptual kernel named path kernel. [44] 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, where $\tilde{O}$ omits the logarithmic factors. In this paper, we also consider the SVM trained by subgradient descent and connect it with NN trained by subgradient descent. [46, 3] studied the connection between SVM and regularization neural network [41], one-hidden layer NN that has very similar structures with that of KMs and is not widely used in practice. NNs used in practice now (e.g. fully connected ReLU NN, CNN, ResNet) do not have such structures. [40] analyzed NN trained by two-layer NN trained by hinge loss without regularization on linearly separable dataset. Note for SVM, it must have a regularization term such that it can achieve max-margin solution.

2.2 Neural Networks and Tangent Kernel

We consider a general form of deep neural network f with a linear output layer as [32]. Let $[L] = \{1, ..., L\}, \forall l \in [L]$,

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

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 [27], 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 [53].

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, $\hat{\Theta}(w; x, x') \to \Theta_\infty(x, x')$. This limiting kernel is called *Neural Tangent Kernel (NTK).*

2.3 Kernel Machines

Kernel machine (KM) 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 some 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 $\Phi(x) = \nabla_w f(w_0, x)$ (or equivalently using NTK as the kernel), $\beta = w_t - w_0$ and φ identity mapping [4].

As far as we know, there is no work establishing the equivalence between fully trained networks and SVM. [17] showed that neural networks trained by gradient flow are approximately KMs, but didn't discuss any specific KM. In this work, we compare the dynamics of SVM and neural network trained by subgradient descent with soft margin loss and show the equivalence between them in the infinite width limit.

2.4 Subgradient Optimization of Support Vector Machine

We first formally define the standard 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 (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 below optimization problem realizes the soft margin classifier with geometric margin $\gamma = 2/\|\beta^*\|$.

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

Proposition 2.1. *The above primal problem of 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. Denote this function as $L(\beta)$, which is strongly convex in 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 [44] to optimize Eq. (2), which shows significant advantages when dealing with large datasets.

In this paper, we consider the soft margin SVM trained by subgradient descent with $L(\beta)$. We use the subgradient $\nabla_\beta L(\beta) = \beta - C\sum_{i=1}^n \mathbb{1}(y_i g(\beta, x_i) < 1) y_i \Phi(x_i)$, where $\mathbb{1}(\cdot)$ is the indicator function. As proved in [44], we can find a solution of accuracy ϵ, i.e. $L(\beta) - L(\beta^*) \leq \epsilon$, in $\tilde{O}(1/\epsilon)$ iterations. Other works also give convergence guarantees for subgradient descent of convex functions [11, 9]. In the following analysis, we will generally 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, and have the same convergence rate under reasonable assumption. 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 KMs as summarized in Table 1.

3.1 Dynamics of Soft Margin SVM

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 C.

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

Theorem 3.1 (Continuous Dynamics and Convergence Rate of SVM)**.** *Consider training 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:*

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

and has a linear convergence rate:

$$L(\beta_t) - L(\beta^*) \leq e^{-2t}\left(L(\beta_0) - L(\beta^*)\right).$$

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

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

The continuous dynamics of SVM is described by an inhomogeneous linear differential equation (Eq. (3)), which gives an analytical solution. 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.

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

3.2 Soft Margin Neural Network

We first formally define the 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 $f(w, x) = \langle W^{(L+1)}, \alpha^{(L)}(w, x)\rangle$ 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 (Soft Margin Neural Network)**.** Given samples $\{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{-1, +1\}$, the neural network w^* defined as Eq. (1) that solves the following two equivalent optimization problems

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

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

realizes the soft margin classifier with geometric margin $\gamma = 2/\|W_*^{(L+1)}\|$. 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, $\|W_0^{(L+1)}\|^2 = O(1)$. The derivative of the regularization for $w^{(L+1)}$ is $w^{(L+1)}/\sqrt{m_L} = O(1/\sqrt{m_L}) \to 0$. For a fixed $\alpha^{(L)}(w, x)$, this problem 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 $\beta = W^{(L+1)}$. If we only train the last layer of NN, it corresponds to an SVM with a NNGP kernel [29, 36]. But for a fully-trained NN, $\alpha^{(L)}(w, x)$ is changing over time.

3.3 Dynamics of Neural Network Trained by Soft Margin Loss

Denote the hinge loss in $L(w)$ as $L_h(y_i, f(w, x_i)) = C \max(0, 1 - y_i f(w, x_i))$. We 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 and Convergence Rate 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 $L(w)$ and gradient flow. Then the network has the following dynamics:*

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

Let $\hat{\Theta}(w_t) \in \mathbb{R}^{n \times n}$ be the tangent kernel evaluated on the training set and $\lambda_{min}(\hat{\Theta}(w_t))$ be its minimum eigenvalue. Assume $\lambda_{min}(\hat{\Theta}(w_t)) \geq \frac{2}{C}$, then NN has at least a linear convergence rate, same as SVM:

$$L(w_t) - L(w^*) \leq e^{-2t} \left(L(w_0) - L(w^*) \right).$$

The proof is in Appendix D. 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 previous continuous-time gradient descent [27, 30], our result has an extra $-f_t(x)$ here because of the regularization term of the loss function. The convergence rate is proved based on a sufficient condition for the PL inequality. The assumption of $\lambda_{min}(\hat{\Theta}(w_t)) \geq \frac{2}{C}$ can be guaranteed in a parameter ball when $\lambda_{min}(\hat{\Theta}(w_0)) > \frac{2}{C}$, by using a sufficiently wide NN [33].

If the tangent kernel $\hat{\Theta}(w_t; x, x_i)$ is fixed, $\hat{\Theta}(w_t; x, x_i) \to \hat{\Theta}(w_0; x, x_i)$, the dynamics of NN is the same as that of SVM (Eq. (3)) with kernel $\hat{\Theta}(w_0; x, x_i)$, 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 [32].

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, $m = \min_{l \in [L]} m_l$, goes to infinity, the tangent kernel tends to be constant, $\hat{\Theta}(w_t; x, x_i) \to \hat{\Theta}(w_0; x, x_i)$. Assume $g_0(x) = f_0(x)$. Then the infinitely wide NN trained by subgradient descent with soft margin loss has the same dynamics as SVM with $\hat{\Theta}(w_0; x, x_i)$ trained by subgradient descent:*

$$\frac{df_t(x)}{dt} = -f_t(x) + C \sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i) < 1) y_i \hat{\Theta}(w_0; x, x_i).$$

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

The proof is in Appendix E. We apply the results of [32] 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. We give a finite-width bound for general loss functions in the next section. This theorem establishes the equivalence between infinitely wide NN and SVM for the first time. Previous theoretical results of SVM [16, 45, 42, 48] 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 (Appendix D.4), which is identical to that of SVM. Compared with the previous discrete-time gradient descent [30, 53], our result has an extra $-\eta f_t(x)$ term because of the regularization term of loss function.

$$f_{t+1}(x) - f_t(x) = -\eta f_t(x) + \eta C \sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i) < 1) y_i \hat{\Theta}(w_0; x, x_i).$$

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

Table 1: Summary of our theoretical results on the equivalence between infinite-width NNs and a family of KMs. Thanks to the representer theorem [42], our ℓ_2 regularized KMs can all apply kernel trick, meaning infinite NTK can be applied in these ℓ_2 regularized KMs.

λ	Loss $l(z, y_i)$	Kernel machine
$\lambda = 0$([27, 4])	$(y_i - z)^2$	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

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 loss function is differentiable (or has subgradients) with respect to z, such as squared loss and logistic loss. Besides, 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 KM and NN 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 $\hat{\Theta}(w_t; x, x_i) \to \hat{\Theta}(w_0; x, x_i)$ and $K(x, x_i) = \hat{\Theta}(w_0; x, x_i)$, 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 KMs that infinite-width NNs are equivalent to. KRR is considered in [25] to analyze the generalization of NN. However, they directly assume NN as a linear model and use it in KRR. Below we give finite-width bounds on the difference between the outputs of NN and the corresponding KM. The proof is in F.

Theorem 3.5 (Bounds on the difference between NN and KM). *Assume $g_0(x) = f_0(x), \forall x$ and $K(x, x_i) = \hat{\Theta}(w_0; x, x_i)$ [3]. Suppose the KM and NN are trained with losses (7) and gradient flow. Suppose l is ρ-lipschitz and β_l-smooth for the first argument (i.e. the model output). Given any $w_T \in B(w_0; R) := \{w : \|w - w_0\| \leq R\}$ for some fixed $R > 0$, for training data $X \in \mathbb{R}^{d \times n}$ and a test point $x \in \mathbb{R}^d$, with high probability over the initialization,*

$$\|f_T(X) - g_T(X)\| = O\left(\frac{e^{\beta_l \|\hat{\Theta}(w_0)\|} R^{3L+1} \rho n^{\frac{3}{2}} \ln m}{\lambda \sqrt{m}}\right),$$

$$\|f_T(x) - g_T(x)\| = O\left(\frac{e^{\beta_l \|\hat{\Theta}(w_0; X, x)\|} R^{3L+1} \rho n \ln m}{\lambda \sqrt{m}}\right).$$

where $f_T(X), g_T(X) \in \mathbb{R}^n$ are the outputs of the training data and $\hat{\Theta}(w_0; X, x) \in \mathbb{R}^n$ is the tangent kernel evaluated between training data and test point.

[3] Linearized NN is a special case of such g.

4 Discussion

In this section, we give some extensions and applications of our theory. We first show that every finite-width neural network trained by a ℓ_2 regularized loss function is approximately a KM in Section 4.1, which enables us to compute non-vacuous generalization bound of NN vis the corresponding KM. 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 infinite-width NNs, while existing robustness verification methods [22, 52, 55] would become much looser (decrease at a rate of $O(1/\sqrt{m})$) as 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 [17], we can also show that every NN trained by (sub)gradient descent with loss function (7) is approximately a KM without the assumption of infinite width.

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 (7) and gradient flow. Assume sign$(l'(y_i, f_t(x_i)))$ = sign$(l'(y_i, f_0(x_i))), \forall t \in [0, T]$.*[4] *Then at some time $T > 0$,*

$$f_T(x) = \sum_{i=1}^{n} a_i K(x, x_i) + b, \quad \textit{with} \quad K(x, x_i) = e^{-\lambda T} \int_0^T |l'(f_t(x_i), y_i)| \hat{\Theta}(w_t; x, x_i) e^{\lambda t} \, dt,$$

and $a_i = -$sign$(l'(f_0(x_i), y_i))$, $b = e^{-\lambda T} f_0(x)$.

See the proof in Appendix G, which utilizes the solution of inhomogeneous linear differential equation instead of integrating both side of dynamics (Eq. (9)) directly [17]. Note in Theorem 4.1, a_i is deterministic and independent with x, different with [17] that has a_i depends on x. Deterministic a_i makes the function class simpler. Combing Theorem 4.1 with a bound of the Rademacher complexity of the KM [7] and a standard generalization using Rademacher complexity [37], we can compute the generalization bound of NN via the corresponding KM. See Appendix B for more background and experiments. The generalization bound we get will depend on a_i, which depends on the label y_i. This differs from traditional complexity measures that cannot explain the random label phenomenon [54].

4.2 Robustness of Infinite-width Neural Network

Our theory of equivalence allows us to deliver nontrivial robustness certificates for infinite-width NNs by considering the equivalent KMs. 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,

$$\max \ \delta, \quad \text{s.t. } g(x) > 0, \ \forall x \in B(x_0, \delta). \tag{10}$$

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

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

where $u = \frac{\langle x, x' \rangle}{\|x\| \|x'\|} \in [-1, 1]$. See the proof of the following theorem in Appendix H.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 $g(x) = \sum_{i=1}^n \alpha_i \Theta(x, x_i)$, where α_i are known after solving the KM problems (e.g. SVM and KRR). Then we can lower bound $g(x)$ as follows.*

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

Using a simple binary search and above theorem, we can find a lower bound for (10). Because of the equivalence between the infinite-width NN and KM, the lower bound we get for the KM is equivalently a robustness lower bound for the corresponding infinite-width NN.

[4] This is the case for hinge loss.

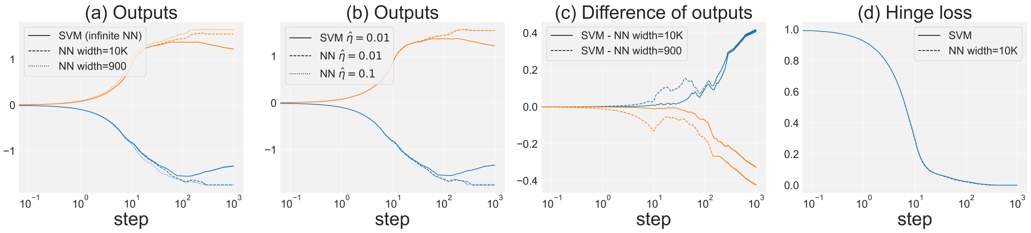

Figure 1: Training dynamics of neural network and SVM behave similarly. (a)(b) show dynamics of outputs for randomly selected two samples. (c) shows the difference between the outputs of SVM and NN. The dynamics of SVM agrees better with wider NN. (d) shows the dynamics of hinge loss for SVM and NN.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

5 Experiments

(I) Verification of the equivalence. The first experiment verifies the equivalence between 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 [28] 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 very 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) [22] 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 in Appendix H.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 ± std) $\times 10^{-3}$	
Model	Width	100 test	Full test
NN	10^3	7.4485 ± 2.5667	7.2708 ± 2.1427
NN	10^4	2.9861 ± 1.0730	2.9367 ± 0.89807
NN	10^5	0.99098 ± 0.35775	0.97410 ± 0.29997
NN	10^6	0.31539 ± 0.11380	0.30997 ± 0.095467
SVM	∞	8.0541 ± 2.5827	7.9733 ± 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
$\lambda = 0$([27, 4])	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	0.064	99.95%	0.008054	243.91X

6 Conclusion and Future Works

In this paper, we establish the equivalence between SVM with NTK and the NN 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 KM. Finally, we demonstrate our theory is useful to compute *non-vacuous* generalization bound for NN, *non-trivial* robustness certificate for infinite-width NN while existing neural network robustness verification algorithm cannot handle, 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 KMs) with NTK has been established, it would be very interesting to understand the generalization and robustness of NN from the perspective of these KMs. 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.

7 Acknowledgement

We thank the anonymous reviewers for useful suggestions to improve the paper. We thank Libin Zhu for helpful discussions. We thank San Diego Supercomputer Center and MIT-IBM Watson AI Lab for computing resources. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) [51], which is supported by National Science Foundation grant number ACI-1548562. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) *Expanse* at San Diego Supercomputer Center through allocation TG-ASC150024 and ddp390. T.-W. Weng is partially supported by National Science Foundation under Grant No. 2107189.

References

[1] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via overparameterization. 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] P. Andras. The equivalence of support vector machine and regularization neural networks. *Neural Processing Letters*, 15(2):97–104, 2002.

[4] 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.

[5] 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.

[6] 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.

[7] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[8] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*, 20(1):2285–2301, 2019.

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

[10] 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.

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

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

[13] L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

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

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

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

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

[18] 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.

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

[20] 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.

[21] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[22] 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.

[23] 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.

[24] T. Hu, W. Wang, C. Lin, and G. Cheng. Regularization matters: A nonparametric perspective on overparametrized neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 829–837. PMLR, 2021.

[25] 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.

[26] W. Huang, W. Du, and R. Y. Da Xu. On the neural tangent kernel of deep networks with orthogonal initialization. *arXiv preprint arXiv:2004.05867*, 2020.

[27] 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.

[28] 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.

[29] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[30] 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.

[31] 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.

[32] 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.

[33] C. Liu, L. Zhu, and M. Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *arXiv preprint arXiv:2003.00307*, 2020.

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

[35] P. M. Long and H. Sedghi. Generalization bounds for deep convolutional neural networks. *arXiv preprint arXiv:1905.12600*, 2019.

[36] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[37] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[38] 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`.

[39] 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.

[40] F. Pellegrini and G. Biroli. An analytic theory of shallow networks dynamics for hinge loss classification. *Advances in Neural Information Processing Systems*, 33, 2020.

[41] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

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

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

[44] 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.

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

[46] A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural networks*, 11(4):637–649, 1998.

[47] 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.

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

[49] 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.

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

[51] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, Sept.-Oct. 2014. ISSN 1521-9615. doi: 10.1109/MCSE.2014.80. URL `doi.ieeecomputersociety.org/10.1109/MCSE.2014.80`.

[52] 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.

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

[54] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[55] 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.

Appendices

A Experiment Details

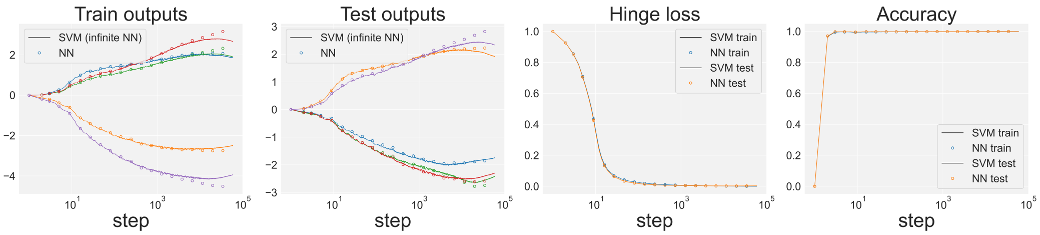

Figure A.1: SVM and NN trained by stochastic subgradient descent for binary MNIST classification task on full 0 and 1 data with learning rate $\hat{\eta} = 1$ and batch size 64. The width of NN is 10K.

A.1 SVM Training

We use the following loss to train the SVM,

$$L(\beta) = \frac{\lambda}{2} \|\beta\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \langle \beta, \Phi(x_i) \rangle). \tag{11}$$

Let $\hat{\eta}$ be the learning rate for this loss in experiments. Then the dynamics of subgradient descent is

$$g_{t+1}(x) = (1 - \hat{\eta}\lambda) g_t(x) + \frac{\hat{\eta}}{n} \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i). \tag{12}$$

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

$$g_T(x) = (1 - \hat{\eta}\lambda)^T \left(g_0(x) + \frac{\hat{\eta}}{n} \sum_{t=0}^{T-1} (1 - \hat{\eta}\lambda)^{-t-1} Q(t) \right). \tag{13}$$

If we set $g_0(x) = 0$, we have

$$g_T(x) = \sum_{t=0}^{T-1} (1 - \hat{\eta}\lambda)^{T-1-t} Q(t). \tag{14}$$

We see that $g_T(x)$ is always a linear combination of kernel values $K(x, x_i)$ for $i = 1, \ldots, n$. Since $K(x, x_i)$ are fixed, we just need to store and update the weights of the kernel values. Let $\alpha_t \in \mathbb{R}^n$ be the weights at time t, that is

$$g_t(x) = \sum_{i=1}^{n} \alpha_{t,i} K(x, x_i). \tag{15}$$

Then according to Eq. (12), we update α at each subgradient descent step as follows.

$$\alpha_{t+1,i} = (1 - \hat{\eta}\lambda) \alpha_{t,i} + \frac{\hat{\eta}}{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i, \quad \forall i \in \{1, \ldots, n\}. \tag{16}$$

For the SGD case, we sample $S_t \subseteq \{1, \ldots, n\}$ at step t and update the weights of this subset while keep the other weights unchanged.

$$\alpha_{t+1,i} = (1 - \hat{\eta}\lambda) \alpha_{t,i} + \frac{\hat{\eta}}{|S_t|} \mathbb{1}(y_i g_t(x_i) < 1) y_i, \quad \forall i \in S_t,$$
$$\alpha_{t+1,i} = \alpha_{t,i}, \quad \forall i \notin S_t.$$

The kernelized implementation of Pegasos [44] set $\hat{\eta}_t = \frac{1}{\lambda t}$ for proving the convergence of the algorithm. In our experiments, we use constant $\hat{\eta}$.

A.2 More Details

(I) Verification of the equivalence. The first experiment illustrates the equivalence between soft margin SVM with NTK trained by subgradient descent and NN trained by soft margin loss. We initialize 3-layer fully connected ReLU neural networks of width 10000 and 900, with NTK parameterization and make sure $f_0(x) = 0$ by subtracting the initial values from NN's outputs. We initialize the parameter of SVM with $\beta_0 = 0$, and this automatically makes sure $g_0(x) = 0$. SVM is trained by directly update the weights of kernel values [44] and more details can be found in Appendix A. We set the regularization parameter as $\lambda = 0.001$ and take the average of the hinge loss instead of sum.[5] We train the NN and SVM for a binary MNIST [28] classification task (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 (see Appendix A). Figure 1 shows the dynamics of the outputs and loss for NN and SVM. Since the regularization term in the loss of NN and SVM are different, we just plot the hinge loss. We see the dynamics of NN and SVM agree well. We also do a stochastic subgradient descent case for binary MNIST classification task on full 0 and 1 data (12665 train data and 2115 test data) with learning rate $\hat{\eta} = 1$ and batch size 64, shown in Figure A.1.

Experiments are implemented with PyTorch [39] and the NTK of infinite-width NN is computed using Neural Tangents [38]. We do our experiments on 16G V100 GPU.

B Computing Non-vacuous Generalization Bounds via Corresponding Kernel Machines

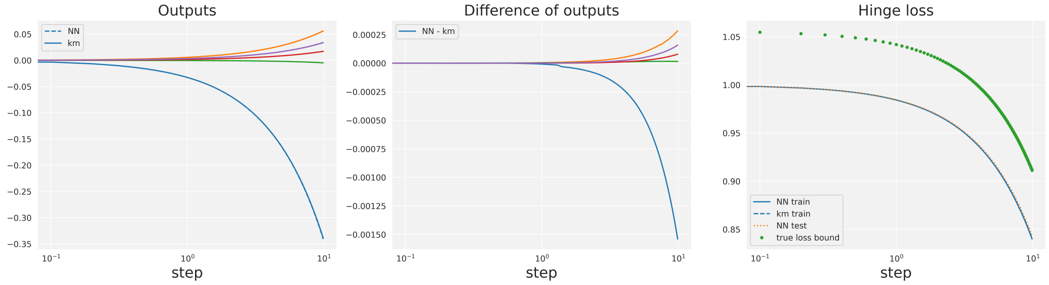

Figure B.2: Computing non-vacuous generalization bounds via corresponding kernel machines. Two-layer NN with 100 hidden nodes trained by full-batch subgradient descent for binary MNIST classification task on full 0 and 1 data with learning rate $\hat{\eta} = 0.1$. The kernel machine (KM) approximates NN very well. And we get a tight bound of the true loss by computing its Rademacher complexity. The confidence parameter is set as $1 - \delta = 0.99$.

Using Theorem 4.1, we can numerically compute the kernel machine that the NN is equivalent to, i.e. we can compute the kernel matrix and the weights at any time during the training. Then one can apply a generalization bound of kernel machines to give an generalization bound for this kernel machine (equivalently for this NN). Let $\mathcal{H}$ be the reproducing kernel Hilbert space (RKHS) corresponding to the kernel $K(\cdot, \cdot)$. The RKHS norm of a function $f(x) = \sum_{i=1}^n a_i K(x, x_i)$ is [6]

$$\|f\|_{\mathcal{H}} = \left\| \sum_{i=1}^n a_i \Phi(x_i) \right\| = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_i a_j K(x_i, x_j)}$$

Lemma B.1 (Lemma 22 in [7]). *For a function class $\mathcal{F}_B = \{f(x) = \sum_{i=1}^n a_i K(x, x_i) : \|f\|_{\mathcal{H}} \leq B\} \subseteq \{x \to \langle \beta, \Phi(x) \rangle : \|\beta\| \leq B\}$, its empirical Rademacher complexity can be bounded as*

$$\hat{\mathcal{R}}_S(\mathcal{F}_B) = \frac{1}{n} \mathop{\mathbb{E}}_{\sigma_i \sim \{\pm 1\}^n} \left[\sup_{f \in \mathcal{F}_B} \sum_{i=1}^n \sigma_i f(x_i) \right] \leq \frac{B}{n} \sqrt{\sum_{i=1}^n K(x_i, x_i)}$$

[5] This is equivalent to use $\lambda = 0.001 \times n$ in Eq. (7).

[6] Assume $f_0(x) = 0$.

Assume the data is sampled i.i.d. from some distribution D and the population loss is $L_D(f) = \mathbb{E}_{(x,y)\sim D}\left[l(f(x), y)\right]$. The experical loss is $L_S(f) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i)$. Combing with a standard generalization bound using Rademacher complexity blow [37], we can get a bound of the population loss $L_D(f)$ for the kernel machine (equivalently for this NN).

Lemma B.2. *Suppose the loss $\ell(\cdot,\cdot)$ is bounded in $[0, c]$, and is ρ-Lipschitz in the first argument. Then with probability at least $1-\delta$ over the sample S of size n,*

$$\sup_{f\in\mathcal{F}}\{L_D(f) - L_S(f)\} \le 2\rho\hat{\mathcal{R}}_S(\mathcal{F}) + 3c\sqrt{\frac{\log(2/\delta)}{2n}}$$

Most of the existing generalization bounds of NN [8, 35] are vacacous since they have a dependence on the number of parameters. Compared to those, the bound for kernel machines does not have a dependence on the number of NN's parameters, making it non-vacuous and promising. Moreover, we can even apply this generalization bound to optimize NN directly like PAC-Bayes bound [21], which gives NN with guaranteed generalization ability.

C Dynamics of Support Vector Machine

In this section, we derive the continuous and discrete dynamics of soft margin SVM trained by subgradient with the following loss function

$$L(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i\langle\beta, \Phi(x_i)\rangle), \tag{17}$$

and the subgradient

$$\nabla_\beta L(\beta_t) = \beta_t - C\sum_{i=1}^{n}\mathbb{1}(y_i g_t(x_i) < 1) y_i \Phi(x_i). \tag{18}$$

Lemma C.1. *$L(\beta)$ satisfies the Polyak- Lojasiewicz (PL) inequality,*

$$L(\beta_t) - L(\beta^*) \le \frac{1}{2}\|\nabla_\beta L(\beta_t)\|^2 \quad \forall\, \beta_t. \tag{19}$$

where $\beta^* = \arg\min_\beta L(\beta)$.

Proof. Since $L(\beta)$ is 1-strongly convex, by the definition of strong convexity and subgradient

$$L(\beta) \ge L(\beta_t) + \langle\nabla_\beta L(\beta_t), \beta - \beta_t\rangle + \frac{1}{2}\|\beta - \beta_t\|^2 \tag{20}$$

The right hand side is a convex quadratic function of β (for fixed β_t). Setting the gradient with respect to β equal to 0, we find that $\tilde{\beta} = \beta_t - \nabla_\beta L(\beta_t)$ minimize right hand side. Therefore we have

$$\begin{aligned} L(\beta) &\ge L(\beta_t) + \langle\nabla_\beta L(\beta_t), \beta - \beta_t\rangle + \frac{1}{2}\|\beta - \beta_t\|^2 \\ &\ge L(\beta_t) + \left\langle\nabla_\beta L(\beta_t), \tilde{\beta} - \beta_t\right\rangle + \frac{1}{2}\left\|\tilde{\beta} - \beta_t\right\|^2 \\ &= L(\beta_t) - \frac{1}{2}\|\nabla_\beta L(\beta_t)\|^2. \end{aligned} \tag{21}$$

Since this holds for any β, we have

$$L(\beta^*) \ge L(\beta_t) - \frac{1}{2}\|\nabla_\beta L(\beta_t)\|^2. \tag{22}$$

□

C.1 Continuous Dynamics of SVM

Here we give the detailed derivation of the dynamics of soft margin SVM trained by subgradient. In the learning rate $\eta \to 0$ limit, the subgradient descent equation, which can also be written as

$$\frac{\beta_{t+1} - \beta_t}{\eta} = -\nabla_\beta L(\beta_t), \tag{23}$$

becomes a differential equation

$$\frac{d\beta_t}{dt} = -\nabla_\beta L(\beta_t). \tag{24}$$

This is known as gradient flow [2]. And we have defined the subgradient as

$$\nabla_\beta L(\beta_t) = \beta_t - C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i \Phi(x_i). \tag{25}$$

Applying the chain rule, the dynamics of $g_t(x) = \langle \beta_t, \Phi(x) \rangle$ is

$$\begin{aligned} \frac{dg_t(x)}{dt} &= \frac{\partial g_t(x)}{\partial \beta_t} \frac{d\beta_t}{dt} \\ &= \left\langle \Phi(x), -\beta_t + C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i \Phi(x_i) \right\rangle \\ &= -g_t(x) + C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i). \end{aligned} \tag{26}$$

Denoting $Q(t) = C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i)$, the equation becomes

$$\frac{dg_t(x)}{dt} + g_t(x) = Q(t). \tag{27}$$

Note this is a first-order inhomogeneous differential equation. The general solution at some time T is given by

$$g_T(x) = e^{-T} \left(g_0(x) + \int_0^T Q(t) e^t \, dt \right). \tag{28}$$

As we already know that the loss function is strongly convex, β will converge to the global optimizer in this infinite small learning rate setting. This can be seen by

$$\frac{d\left(L(\beta_t) - L(\beta^*)\right)}{dt} = \frac{dL(\beta_t)}{dt} = \frac{\partial L(\beta_t)}{\partial \beta_t} \frac{d\beta_t}{dt} = \langle \nabla_\beta L(\beta_t), -\nabla_\beta L(\beta_t) \rangle = -\left\| \nabla_\beta L(\beta_t) \right\|^2 . \tag{29}$$

We see that $L(\beta_t)$ is always decreasing. Since $L(\beta)$ is strongly convex and thus bounded from below, by monotone convergence theorem, $L(\beta_t)$ will always converge. By Lemma C.1, we have the Polyak-Lojasiewicz (PL) inequality,

$$L(\beta_t) - L(\beta^*) \le \frac{1}{2} \left\| \nabla_\beta L(\beta_t) \right\|^2 \tag{30}$$

Combining with above, we have

$$\frac{d\left(L(\beta_t) - L(\beta^*)\right)}{dt} \le -2\left(L(\beta_t) - L(\beta^*)\right). \tag{31}$$

Solving the equation, we get

$$L(\beta_t) - L(\beta^*) \le e^{-2t}\left(L(\beta_0) - L(\beta^*)\right). \tag{32}$$

Thus we have a linear convergence rate.

Now, let us assume $g_T(x)$ will converge and see what is $g_T(x)$ as $T \to \infty$. As time increases $T \to \infty$, $e^{-T} g_0(x) \to 0$.

$$g_T(x) \to e^{-T} \int_0^T Q(t) e^t \, dt \tag{33}$$

$Q(t)$ is changing over time due to $g_t(x)$ is changing. Suppose $Q(t)$ keeps changing until some time T_1 and keeps constant, $Q(t) = Q$, after T_1,

$$\lim_{T\to\infty} g_T(x) = e^{-T} \int_0^{T_1} Q(t)e^t \, dt + e^{-T} \int_{T_1}^{T} Qe^t \, dt. \tag{34}$$

As $T \to \infty$, the first part of right hand side converges to 0.

$$\begin{aligned}
\lim_{T\to\infty} g_T(x) &\to e^{-T} \int_{T_1}^{T} Qe^t \, dt \\
&= e^{-T} \int_{T_1}^{T} e^t \, dt \cdot Q \\
&= e^{-T}(e^T - e^{T_1}) \cdot Q \\
&\to Q \\
&= C \sum_{i=1}^{n} \mathbb{1}(y_i g_T(x_i) < 1) \cdot y_i K(x, x_i).
\end{aligned} \tag{35}$$

C.2 Discrete Dynamics of SVM

Let $\eta \in (0, 1)$ be the learning rate. The equation of subgradient descent update at some time t is

$$\beta_{t+1} - \beta_t = -\eta \nabla_\beta L(\beta_t). \tag{36}$$

The dynamics of $g_t(x)$ is

$$\begin{aligned}
g_{t+1}(x) - g_t(x) &= \langle \beta_{t+1} - \beta_t, \Phi(x) \rangle \\
&= \left\langle -\eta\beta_t + \eta C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i \Phi(x_i), \Phi(x) \right\rangle \\
&= -\eta g_t(x) + \eta C \sum_{i=1}^{n} \mathbb{1}(y_i g_t(x_i) < 1) y_i K(x, x_i).
\end{aligned} \tag{37}$$

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

$$\begin{aligned}
g_T(x) &= (1-\eta) g_{T-1}(x) + Q(T-1) \\
&= (1-\eta)\Big((1-\eta) g_{T-2}(x) + Q(T-2)\Big) + Q(T-1) \\
&= (1-\eta)^T g_0(x) + \sum_{t=0}^{T-1} (1-\eta)^{T-1-t} Q(t) \\
&= (1-\eta)^T \left(g_0(x) + \sum_{t=0}^{T-1} (1-\eta)^{-t-1} Q(t) \right).
\end{aligned} \tag{38}$$

The convergence of subgradient descent usually requires additional assumption that the norm of the subgradient is bounded. We refer readers to [44, 11, 9] for some proofs. Here let us assume the subgradient descent converges to the global optimizer and $Q(t)$ keeps changing until some time T_1

and keeps constant, $Q(t) = Q$, after T_1. As $T \to \infty$,

$$\begin{aligned} g_T(x) &\to \sum_{t=0}^{T-1} (1-\eta)^{T-1-t} Q(t) \\ &= \sum_{t=0}^{T_1-1} (1-\eta)^{T-1-t} Q(t) + \sum_{t=T_1}^{T-1} (1-\eta)^{T-1-t} Q \\ &\to \sum_{t=T_1}^{T-1} (1-\eta)^{T-1-t} Q \\ &= \sum_{t=T_1}^{T-1} (1-\eta)^{T-1-t} Q \\ &= \frac{-(1-\eta)^{T-T_1} + 1}{\eta} Q. \end{aligned} \tag{39}$$

As $\eta \in (0, 1)$, $-(1-\eta)^{T-T_1} \to 0$.

$$\begin{aligned} g_T(x) &\to \frac{1}{\eta} Q \\ &= C \sum_{i=1}^{n} \mathbb{1}(y_i g_T(x_i) < 1) y_i K(x, x_i). \end{aligned} \tag{40}$$

D Dynamics and Convergence Rate of Neural Network Trained by Soft Margin Loss

D.1 Continuous Dynamics of NN

In the learning rate $\eta \to 0$ limit, the subgradient descent equation, which can also be written as

$$\frac{w_{t+1} - w_t}{\eta} = -\nabla_w L(w_t), \tag{41}$$

becomes a differential equation

$$\frac{dw_t}{dt} = -\nabla_w L(w_t). \tag{42}$$

This is known as gradient flow [2]. Then for any differentiable function $f_t(x)$,

$$\frac{df_t(x)}{dt} = \sum_{j=1}^{p} \frac{\partial f_t(x)}{\partial w_j} \frac{dw_j}{dt}, \tag{43}$$

where p is the number of parameters. Replacing $\frac{dw_j}{dt}$ by its subgradient descent expression:

$$\frac{df_t(x)}{dt} = \sum_{j=1}^{p} \frac{\partial f_t(x)}{\partial w_j} (-\frac{\partial L(w_t)}{\partial w_j}). \tag{44}$$

And we know

$$\frac{\partial L(w_t)}{\partial w_j} = w_j \mathbb{1}(w_j \in W^{(L+1)}) + \sum_{i=1}^{n} \frac{\partial L_h}{\partial f_t(x_i)} \frac{\partial f_t(x_i)}{\partial w_j}. \tag{45}$$

where $\mathbb{1}(w_j \in W^{(L+1)})$ equals to 1 if the parameter w_j is in the last layer $W^{(L+1)}$ else 0. Combining above together,

$$\frac{df_t(x)}{dt} = \sum_{j=1}^{p} \frac{\partial f_t(x)}{\partial w_j} \Big(-w_j \mathbb{1}(w_j \in W^{(L+1)}) - \sum_{i=1}^{n} \frac{\partial L_h}{\partial f_t(x_i)} \frac{\partial f_t(x_i)}{\partial w_j} \Big). \tag{46}$$

Rearranging terms:

$$\frac{df_t(x)}{dt} = -\sum_{k=1}^{p_{L+1}} \frac{\partial f_t(x)}{\partial W_k^{(L+1)}} W_k^{(L+1)} - \sum_{i=1}^{n} \frac{\partial L_h}{\partial f_t(x_i)} \sum_{j=1}^{p} \frac{\partial f_t(x)}{\partial w_j} \frac{\partial f_t(x_i)}{\partial w_j}, \tag{47}$$

where p_{L+1} is the number of parameters of the last layer ($L+1$ layer). The first part of the right hand side is

$$\sum_{k=1}^{p_{L+1}} \frac{\partial f_t(x)}{\partial W_k^{(L+1)}} W_k^{(L+1)} = \left\langle \frac{\partial f_t(x)}{\partial W^{(L+1)}}, W^{(L+1)} \right\rangle = \left\langle \alpha_t^{(L)}(x), W^{(L+1)} \right\rangle = f_t(x). \tag{48}$$

Applying $L_h'(y_i, f_t(x_i)) = \frac{\partial L_h}{\partial f_t(x_i)}$, the subgradient of hinge loss, and the definition of tangent kernel (2.1), the second part is

$$-\sum_{i=1}^{n} \frac{\partial L_h}{\partial f_t(x_i)} \sum_{j=1}^{p} \frac{\partial f_t(x)}{\partial w_j} \frac{\partial f_t(x_i)}{\partial w_j} = -\sum_{i=1}^{n} L_h'(y_i, f_t(x_i)) \hat{\Theta}(w_t; x, x_i). \tag{49}$$

Thus the equation becomes

$$\frac{df_t(x)}{dt} = -f_t(x) - \sum_{i=1}^{n} L_h'(y_i, f_t(x_i)) \hat{\Theta}(w_t; x, x_i). \tag{50}$$

Take $L_h'(y_i, f_t(x_i)) = -Cy_i \mathbb{1}(y_i f_t(x_i) < 1)$ in

$$\frac{df_t(x)}{dt} = -f_t(x) + C \sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i) < 1) y_i \hat{\Theta}(w_t; x, x_i). \tag{51}$$

D.2 Additional Notations

Denote $X \in \mathbb{R}^{d\times n}$ as the training data. Denote $f_t = f_t(X) \in \mathbb{R}^n$ and $g_t = g_t(X) \in \mathbb{R}^n$ as the outputs of NN and SVM on the training data. Denote $\hat{\Theta}(w_t) = \hat{\Theta}(w_t; X, X) \in \mathbb{R}^{n\times n}$ as the tangent kernel evaluated on the training data at time t, and $l'(f_t) \in \mathbb{R}^n$ as the derivative of the loss function w.r.t. f_t. Denote $\nabla_w f_t \in \mathbb{R}^{n\times p}$ as the Jacobian and we have $\hat{\Theta}(w_t) = \nabla_w f_t \nabla_w f_t^T$. Denote $\lambda_0 = \lambda_{min}\left(\hat{\Theta}(w_t)\right)$ as the smallest eigenvalue of $\hat{\Theta}(w_t)$. Then we can write the dynamics of NN as

$$\frac{d}{dt} f_t = -f_t - \hat{\Theta}(w_t) l'(f_t).$$

Let $v \in \mathbb{R}^p$ with $v_j = \mathbb{1}(w_j \in W^{(L+1)})$. We can write the gradient as

$$\nabla_w L(w_t) = w_t \odot v + \nabla_w f_t^T l'(f_t).$$

D.3 Convergence of NN

The loss of NN is

$$L(w_t) = \frac{1}{2} \left\| W_t^{(L+1)} \right\|^2 + \sum_{i}^{n} l(f_t(x_i), y_i),$$

where $l(f, y) = C \max(0, 1 - yf)$. The dynamic of the loss is

$$\frac{dL(w_t)}{dt} = \frac{\partial L(w_t)}{\partial w_t} \frac{dw_t}{dt} = \langle \nabla_w L(w_t), -\nabla_w L(w_t) \rangle = -\left\| \nabla_w L(w_t) \right\|^2.$$

Since $L(w_t) \geq 0$ is bounded from below, by monotone convergence theorem, $L(w_t)$ will always converge to a stationary point. Applying Lemma D.1, we have

$$\frac{d\left(L(w_t) - L(w^*)\right)}{dt} = -\left\| \nabla_w L(w_t) \right\|^2 \leq -2\left(L(w_t) - L(w^*)\right).$$

Thus we have a linear convergence, same as SVM.

$$L(w_t) - L(w^*) \leq e^{-2t}\left(L(w_0) - L(w^*)\right).$$

Lemma D.1 (PL inequality of NN for soft margin loss)**.** *Assume $\lambda_0 \geq \frac{2}{C}$, then $L(w_t)$ satisfies the PL condition*

$$\left\|\nabla_w L(w_t)\right\|^2 \geq 2\left(L(w_t) - L(w^*)\right).$$

Proof.

$$\begin{aligned}
\left\|\nabla_w L(w_t)\right\|^2 &= \left\langle w_t \odot v + \nabla_w f_t^T l'(f_t), w_t \odot v + \nabla_w f_t^T l'(f_t)\right\rangle \\
&= \langle w_t \odot v, w_t \odot v\rangle + \left\langle \nabla_w f_t^T l'(f_t), \nabla_w f_t^T l'(f_t)\right\rangle + 2\left\langle w_t \odot v, \nabla_w f_t^T l'(f_t)\right\rangle \\
&= \left\|W_t^{(L+1)}\right\|^2 + l'(f_t)^T \hat{\Theta}(w_t) l'(f_t) + 2\left\langle W_t^{(L+1)}, \nabla_{W^{(L+1)}} f_t^T l'(f_t)\right\rangle \\
&= \left\|W_t^{(L+1)}\right\|^2 + l'(f_t)^T \hat{\Theta}(w_t) l'(f_t) + 2 f_t^T l'(f_t).
\end{aligned}$$

We want the loss satisfies the PL condition $\left\|\nabla_w L(w_t)\right\|^2 \geq 2\left(L(w_t) - L(w^*)\right)$.

$$\begin{aligned}
&\left\|\nabla_w L(w_t)\right\|^2 - 2\left(L(w_t) - L(w^*)\right) \\
&= \left\|\nabla_w L(w_t)\right\|^2 - 2L(w_t) + 2L(w^*) \\
&= l'(f_t)^T \hat{\Theta}(w_t) l'(f_t) + 2 f_t^T l'(f_t) - 2\sum_i^n l(f_t(x_i), y_i) + 2L(w^*) \\
&\geq \lambda_0 \left\|l'(f_t)\right\|^2 + 2 f_t^T l'(f_t) - 2\sum_i^n l(f_t(x_i), y_i) + 2L(w^*),
\end{aligned}$$

where the last inequality is the inequality of quadratic form. For hinge loss $l(f, y) = C\max(0, 1 - yf) = C(1 - yf)\mathbb{1}(1 - yf > 0)$ and $l'(f, y) = -Cy\mathbb{1}(1 - yf > 0)$,

$$\begin{aligned}
&\left\|\nabla_w L(w_t)\right\|^2 - 2\left(L(w_t) - L(w^*)\right) \\
&\geq \lambda_0 \left\|l'(f_t)\right\|^2 + 2 f_t^T l'(f_t) - 2\sum_i^n l(f_t(x_i), y_i) + 2L(w^*) \\
&= \lambda_0 \sum_i^n l'(f_t(x_i), y_i)^2 + 2\sum_i^n f_t(x_i) l'(f_t(x_i), y_i) - 2\sum_i^n l(f_t(x_i), y_i) + 2L(w^*) \\
&= \lambda_0 \sum_i^n C^2 \mathbb{1}(1 - y_i f_t(x_i) > 0) - 2\sum_i^n C y_i f_t(x_i) \mathbb{1}(1 - y_i f_t(x_i) > 0) \\
&\quad - 2\sum_i^n C(1 - y_i f_t(x_i)) \mathbb{1}(1 - y_i f_t(x_i) > 0) + 2L(w^*) \\
&= C\sum_i^n \mathbb{1}(1 - y_i f_t(x_i) > 0)\left(C\lambda_0 - 2\right) + 2L(w^*).
\end{aligned}$$

Since $L(w^*) > 0$, as long as $\lambda_0 \geq 2/C$, the loss $L(w_t)$ satisfies the PL condition $\left\|\nabla_w L(w_t)\right\|^2 \geq 2\left(L(w_t) - L(w^*)\right)$. $\lambda_0 \geq 2/C$ can be guaranteed in a parameter ball when $\frac{2}{C} < \lambda_{min}\left(\hat{\Theta}(w_0)\right)$ by using a sufficiently wide NN [33]. □

D.4 Discrete Dynamics of NN

The subgradient descent update is

$$w_{t+1} - w_t = -\eta \nabla_w L(w_t). \tag{52}$$

We consider the situation of constant NTK, $\hat{\Theta}(w_t; x, x_i) \to \hat{\Theta}(w_0; x, x_i)$, or equivalently linear model. As proved by Proposition 2.2 in [32], the tangent kernel of a differentiable function $f(w, x)$

is constant if and only if $f(w,x)$ is linear in w. Take the Taylor expansion of $f(w_{t+1},x)$ at w_t,

$$\begin{aligned}
& f(w_{t+1},x)-f(w_t,x)\\
&= f(w_t,x)+\langle \nabla_w f(w_t,x), w_{t+1}-w_t\rangle - f(w_t,x)\\
&= \langle \nabla_w f(w_t,x), -\eta\nabla_w L(w_t)\rangle\\
&= \left\langle \nabla_w f(w_t,x), -\eta\Big(wv+\sum_{i=1}^{n} L_h'(y_i,f_t(x_i))\nabla_w f_t(x_i)\Big)\right\rangle\\
&= -\eta f_t(x)+\eta\sum_{i=1}^{n} L_h'(y_i,f_t(x_i))\hat{\Theta}(w_t;x,x_i) \qquad (53)\\
&= -\eta f_t(x)+\eta C\sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i)<1)y_i\hat{\Theta}(w_t;x,x_i)\\
&\to -\eta f_t(x)+\eta C\sum_{i=1}^{n} \mathbb{1}(y_i f_t(x_i)<1)y_i\hat{\Theta}(w_0;x,x_i).
\end{aligned}$$

E Proof of Theorem 3.4

Proof. We prove the constancy of tangent kernel by adopting the results of [34].

Lemma E.1 (Theorem 3.3 in [32]; Hessian norm is controlled by the minimum hidden layer width)**.** *Consider a general neural network $f(w,x)$ of the form Eq. (1), which can be a fully connected network, CNN, ResNet or a mixture of these types. Let m be the minimum of the hidden layer widths, i.e., $m=\min_{l\in[L]} m_l$. Given any fixed $R>0$, and any $w\in B(w_0;R):=\{w:\|w-w_0\|\le R\}$, with high probability over the initialization, the Hessian spectral norm satisfies the following:*

$$\|H(w)\| = O(\frac{R^{3L}\ln m}{\sqrt{m}}). \qquad (54)$$

Lemma E.2 (Proposition 2.3 in [32]; Small Hessian norm ⇒ Small change of tangent kernel)**.** *Given a point $w_0\in\mathbb{R}^p$ and a ball $B(w_0;R):=\{w:\|w-w_0\|\le R\}$ with fixed radius $R>0$, if the Hessian matrix satisfies $\|H(w)\|<\epsilon$, where $\epsilon>0$, for all $w\in B(w_0,R)$, then the tangent kernel $\hat{\Theta}(w;x,x')$ of the model, as a function of w, satisfies*

$$\left|\hat{\Theta}(w;x,x')-\hat{\Theta}(w_0;x,x')\right| = O(\epsilon R), \quad \forall w\in B(w_0;R),\ \forall x,x'\in\mathbb{R}^d. \qquad (55)$$

Applying above two lemmas, we can see that in the limit of $m\to\infty$, the spectral norm of Hessian converge to 0 and the tangent kernel keeps constant in the ball $B(w_0;R)$.

Corollary E.2.1 (Consistancy of tangent kernel)**.** *Consider a general neural network $f(w,x)$ of the form Eq. (1). Given a point $w_0\in\mathbb{R}^p$ and a ball $B(w_0;R):=\{w:\|w-w_0\|\le R\}$ with fixed radius $R>0$, in the infinite width limit, $m\to\infty$,*

$$\lim_{m\to\infty}\hat{\Theta}(w;x,x')\to\hat{\Theta}(w_0;x,x_i), \quad \forall w\in B(w_0;R),\ \forall x,x'\in\mathbb{R}^d. \qquad (56)$$

Thus we prove the constancy of tangent kernel in infinite width limit. Then it is easy to check the dynamics of infinitely wide NN is the same with the dynamics of SVM with constant NTK.

□

F Bound the difference between SVM and NN

Assume the loss l is ρ-lipschitz and β_l-smooth for the first argument (i.e. the model output). Assume $f_0(x)=g_0(x)$ for any x.

F.1 Bound the difference on the Training Data

The dynamics of the NN and SVM are

$$\frac{d}{dt}f_t = -\lambda f_t - \hat{\Theta}(w_t)l'(f_t)$$
$$\frac{d}{dt}g_t = -\lambda g_t - \hat{\Theta}(w_0)l'(g_t)$$

The dynamics of the difference between them is

$$\frac{d}{dt}\left(f_t - g_t\right) = -\lambda\left(f_t - g_t\right) - \left(\hat{\Theta}(w_t)l'(f_t) - \hat{\Theta}(w_0)l'(g_t)\right)$$

The solution of the above differential equation at time T is

$$\begin{aligned} f_T - g_T &= e^{-\lambda T}\left(f_0 - g_0\right) - e^{-\lambda T}\int_0^T \left(\hat{\Theta}(w_t)l'(f_t) - \hat{\Theta}(w_0)l'(g_t)\right)e^{\lambda t}dt \\ &= e^{-\lambda T}\int_0^T \left(\hat{\Theta}(w_0)l'(g_t) - \hat{\Theta}(w_t)l'(f_t)\right)e^{\lambda t}dt \end{aligned}$$

using $f_0 = g_0$. Thus

$$\|f_T - g_T\| \leq e^{-\lambda T}\int_0^T \left\|\hat{\Theta}(w_0)l'(g_t) - \hat{\Theta}(w_t)l'(f_t)\right\| e^{\lambda t}dt$$

Since l is β_l smooth,

$$\begin{aligned} \left\|\hat{\Theta}(w_0)l'(g_t) - \hat{\Theta}(w_t)l'(f_t)\right\| &= \left\|\hat{\Theta}(w_0)l'(g_t) - \hat{\Theta}(w_0)l'(f_t) + \hat{\Theta}(w_0)l'(f_t) - \hat{\Theta}(w_t)l'(f_t)\right\| \\ &= \left\|\hat{\Theta}(w_0)\left(l'(g_t) - l'(f_t)\right) + \left(\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right)l'(f_t)\right\| \\ &\leq \left\|\hat{\Theta}(w_0)\left(l'(g_t) - l'(f_t)\right)\right\| + \left\|\left(\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right)l'(f_t)\right\| \\ &\leq \beta_l\left\|\hat{\Theta}(w_0)\right\|\|g_t - f_t\| + \rho\sqrt{n}\left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\| \end{aligned}$$

where $\|l'(f_t)\| \leq \rho\sqrt{n}$. Thus we have

$$\|f_T - g_T\| \leq e^{-\lambda T}\beta_l\left\|\hat{\Theta}(w_0)\right\|\int_0^T \|g_t - f_t\| e^{\lambda t}dt + e^{-\lambda T}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\| e^{\lambda t}dt$$

Applying the Grönwall's inequality,

$$\begin{aligned} \|f_T - g_T\| &\leq e^{-\lambda T}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\| e^{\lambda t}dt \cdot e^{e^{-\lambda T}\beta_l\left\|\hat{\Theta}(w_0)\right\|\int_0^T e^{\lambda t}dt} \\ &= e^{-\lambda T}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\| e^{\lambda t}dt \cdot e^{\frac{1}{\lambda}(1-e^{-\lambda T})\beta_l\left\|\hat{\Theta}(w_0)\right\|} \\ &= e^{-\lambda T}e^{\frac{1}{\lambda}(1-e^{-\lambda T})\beta_l\left\|\hat{\Theta}(w_0)\right\|}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\| e^{\lambda t}dt \end{aligned}$$

By Lemma E.1 and Lemma E.2, in a parameter ball $B(w_0; R) = \{w : \|w - w_0\| \leq R\}$, with high probability, $\left|\hat{\Theta}(w; x, x') - \hat{\Theta}(w_0; x, x')\right| = O(R^{3L+1}\ln m/\sqrt{m})$ w.r.t. m. Then we have

$$\left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\| \leq \left\|\hat{\Theta}(w_0) - \hat{\Theta}(w_t)\right\|_F = O(\frac{R^{3L+1}n\ln m}{\sqrt{m}})$$

Thus we have

$$\begin{aligned} \|f_T - g_T\| &\leq \frac{1}{\lambda}(1 - e^{-\lambda T})e^{(1-e^{-T})\beta_l\left\|\hat{\Theta}(w_0)\right\|}\rho\sqrt{n}\cdot O(\frac{R^{3L+1}n\ln m}{\sqrt{m}}) \\ &= O(\frac{e^{\beta_l\left\|\hat{\Theta}(w_0)\right\|}R^{3L+1}\rho n^{\frac{3}{2}}\ln m}{\lambda\sqrt{m}}) \end{aligned}$$

F.2 Bound on the Test Data

For a test data x, the prove is similar to the training case. Denote $\hat{\Theta}(w_t; X, x) \in \mathbb{R}^n$ as the tangent kernel evaluate between the training data and a test data x. Recall

$$\begin{aligned}
\frac{df_t(x)}{dt} &= -\lambda f_t(x) - \hat{\Theta}(w_t; X, x)^T l'(f_t) \\
\frac{dg_t(x)}{dt} &= -\lambda g_t(x) - \hat{\Theta}(w_0; X, x)^T l'(g_t) \\
\frac{d}{dt}\left(f_t(x) - g_t(x)\right) &= -\lambda\left(f_t(x) - g_t(x)\right) - \left(\hat{\Theta}(w_t; X, x)^T l'(f_t) - \hat{\Theta}(w_0; X, x)^T l'(g_t)\right)
\end{aligned}$$

The solution of the above differential equation is

$$\begin{aligned}
f_T(x) - g_T(x) &= e^{-\lambda T}\left(f_0 - g_0\right) - e^{-\lambda T}\int_0^T \left(\hat{\Theta}(w_t; X, x)^T l'(f_t) - \hat{\Theta}(w_0; X, x)^T l'(g_t)\right) e^{\lambda t} dt \\
&= e^{-\lambda T}\int_0^T \left(\hat{\Theta}(w_0; X, x)^T l'(g_t) - \hat{\Theta}(w_t; X, x)^T l'(f_t)\right) e^{\lambda t} dt
\end{aligned}$$

using $f_0 = g_0$. Thus

$$\|f_T(x) - g_T(x)\| \le e^{-\lambda T}\int_0^T \left\|\hat{\Theta}(w_0; X, x)^T l'(g_t) - \hat{\Theta}(w_t; X, x)^T l'(f_t)\right\| e^{\lambda t} dt$$

Since l is β_l smooth,

$$\begin{aligned}
&\left\|\hat{\Theta}(w_0; X, x)^T l'(g_t) - \hat{\Theta}(w_t; X, x)^T l'(f_t)\right\| \\
&= \left\|\hat{\Theta}(w_0; X, x)^T l'(g_t) - \hat{\Theta}(w_0; X, x)^T l'(f_t) + \hat{\Theta}(w_0; X, x)^T l'(f_t) - \hat{\Theta}(w_t; X, x)^T l'(f_t)\right\| \\
&= \left\|\hat{\Theta}(w_0; X, x)^T\left(l'(g_t) - l'(f_t)\right) + \left(\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right) l'(f_t)\right\| \\
&\le \left\|\hat{\Theta}(w_0; X, x)^T\left(l'(g_t) - l'(f_t)\right)\right\| + \left\|\left(\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right) l'(f_t)\right\| \\
&\le \beta_l \left\|\hat{\Theta}(w_0; X, x)\right\| \|g_t - f_t\| + \rho\sqrt{n}\left\|\left(\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right)\right\|
\end{aligned}$$

where $\|l'(f_t)\| \le \rho\sqrt{n}$. Thus we have

$$\begin{aligned}
&\|f_T(x) - g_T(x)\| \\
&\le e^{-\lambda T}\beta_l \left\|\hat{\Theta}(w_0; X, x)\right\| \int_0^T \|g_t - f_t\| e^{\lambda t} dt + e^{-\lambda T}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right\| e^{\lambda t} dt
\end{aligned}$$

Applying the Grönwall's inequality,

$$\begin{aligned}
&\|f_T(x) - g_T(x)\| \\
&\le e^{-\lambda T}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right\| e^{\lambda t} dt \cdot e^{e^{-\lambda T}\beta_l \left\|\hat{\Theta}(w_0; X, x)\right\| \int_0^T e^{\lambda t} dt} \\
&= e^{-\lambda T}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right\| e^{\lambda t} dt \cdot e^{\frac{1}{\lambda}(1 - e^{-\lambda T})\beta_l \left\|\hat{\Theta}(w_0; X, x)\right\|} \\
&= e^{-\lambda T} e^{\frac{1}{\lambda}(1 - e^{-\lambda T})\beta_l \left\|\hat{\Theta}(w_0; X, x)\right\|}\rho\sqrt{n}\int_0^T \left\|\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right\| e^{\lambda t} dt
\end{aligned}$$

By Lemma E.1 and Lemma E.2, in a parameter ball $B(w_0; R) = \{w : \|w - w_0\| \le R\}$, with high probability, $\left|\hat{\Theta}(w; x, x') - \hat{\Theta}(w_0; x, x')\right| = O(R^{3L+1}\ln m/\sqrt{m})$. Then we have

$$\left\|\hat{\Theta}(w_0; X, x)^T - \hat{\Theta}(w_t; X, x)^T\right\| = O(\frac{R^{3L+1}\sqrt{n}\ln m}{\sqrt{m}})$$

Thus we have

$$
\begin{aligned}
\|f_T(x) - g_T(x)\| &\leq \frac{1}{\lambda}(1 - e^{-\lambda T})e^{(1-e^{-T})\beta_l \left\|\hat{\Theta}(w_0;X,x)\right\|}\rho\sqrt{n} \cdot O(\frac{R^{3L+1}\sqrt{n}\ln m}{\sqrt{m}}) \\
&= O(\frac{e^{\beta_l \left\|\hat{\Theta}(w_0;X,x)\right\|}R^{3L+1}\rho n \ln m}{\lambda\sqrt{m}})
\end{aligned}
$$

G Finite-width Neural Networks are Kernel Machines

Inspired by [17], we can also show that every neural network trained by (sub)gradient descent with loss function in the form (7) is approximately a kernel machine without the assumption of infinite width limit.

Theorem G.1. *Suppose a neural network $f(w, x)$, with f a differentiable function of w, is learned from a training set $\{(x_i, y_i)\}_{i=1}^n$ by (sub)gradient descent with loss function $L(w) = \frac{\lambda}{2}\left\|W^{(L+1)}\right\|^2 + \sum_{i=1}^n l(y_i, f(w, x_i))$ and gradient flow. Assume sign$(l'(y_i, f_t(x_i))) =$ sign$(l'(y_i, f_0(x_i))), \forall t \in [0, T]$, keeps unchanged during training. Then at some time T,*

$$
f_T(x) = \sum_{i=1}^n a_i K(x, x_i) + b, \tag{57}
$$

where

$$
\begin{gathered}
a_i = -\textit{sign}(l'(y_i, f_0(x_i))), \qquad b = e^{-\lambda T} f_0(x), \\
K(x, x_i) = e^{-\lambda T}\int_0^T |l'(y_i, f_t(x_i))|\,\hat{\Theta}(w_t; x, x_i)e^{\lambda t}\,dt\,dt
\end{gathered}
$$

Proof. As we have derived, the neural network follows the dynamics of Eq. (9):

$$
\frac{df_t(x)}{dt} = -\lambda f_t(x) - \sum_{i=1}^n l'(y_i, f_t(x_i))\hat{\Theta}(w_t; x, x_i). \tag{58}
$$

Note this is a first-order inhomogeneous linear differential equation with the functions depended on t. Denote $Q(t) = -\sum_{i=1}^n l'(y_i, f_t(x_i))\hat{\Theta}(w_t; x, x_i)$,

$$
\frac{df_t(x)}{dt} + \lambda f_t(x) = Q(t). \tag{59}
$$

Let $f_0(x)$ be the initial model, prior to gradient descent. The solution is given by

$$
f_T(x) = e^{-\lambda T}\left(f_0(x) + \int_0^T Q(t)e^{\lambda t}\,dt\right). \tag{60}
$$

Then

$$
\begin{aligned}
f_T(x) &= e^{-\lambda T}\left(f_0(x) - \sum_{i=1}^n \int_0^T l'(y_i, f_t(x_i))\hat{\Theta}(w_t; x, x_i)e^{\lambda t}\,dt\right) \\
&= e^{-\lambda T} f_0(x) - \sum_{i=1}^n e^{-\lambda T}\int_0^T l'(y_i, f_t(x_i))\hat{\Theta}(w_t; x, x_i)e^{\lambda t}\,dt \\
&= e^{-\lambda T} f_0(x) - \sum_{i=1}^n e^{-\lambda T}\int_0^T \text{sign}(l'(y_i, f_t(x_i))) \cdot |l'(y_i, f_t(x_i))|\,\hat{\Theta}(w_t; x, x_i)e^{\lambda t}\,dt \\
&= e^{-\lambda T} f_0(x) - \sum_{i=1}^n \text{sign}(l'(y_i, f_0(x_i))) \cdot e^{-\lambda T}\int_0^T |l'(y_i, f_t(x_i))|\,\hat{\Theta}(w_t; x, x_i)e^{\lambda t}\,dt.
\end{aligned} \tag{61}
$$

where the last equality uses the assumption $\text{sign}(l'(y_i, f_t(x_i))) = \text{sign}(l'(y_i, f_0(x_i))), \forall t \in [0, T]$. Thus

$$f_T(x) = \sum_{i=1}^{n} a_i K(x, x_i) + b, \tag{62}$$

with

$$a_i = -\text{sign}(l'(y_i, f_0(x_i))), \qquad b = e^{-\lambda T} f_0(x),$$
$$K(x, x_i) = e^{-\lambda T} \int_0^T |l'(y_i, f_t(x_i))| \, \hat{\Theta}(w_t; x, x_i) e^{\lambda t} \, dt$$

□

$K(x, x_i) = e^{-\lambda T} \int_0^T |l'(y_i, f_t(x_i))| \, \hat{\Theta}(w_t; x, x_i) e^{\lambda t} \, dt$ is a valid kernel since it is a nonnegative sum of positive definite kernels. 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.

H Robustness of Over-parameterized Neural Network

H.1 Robustness Verification of NTK

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

$$\Theta(x, x') = \frac{\langle x, x' \rangle}{d} (\frac{\pi - \arccos(u)}{\pi}) + \frac{\|x\| \, \|x'\|}{2\pi d} \sqrt{1 - u^2} = \frac{\|x\| \, \|x'\|}{2\pi d} h(u), \tag{63}$$

$$h(u) = 2u(\pi - \arccos(u)) + \sqrt{1 - u^2}. \tag{64}$$

where $u = \frac{\langle x, x' \rangle}{\|x\| \|x'\|} \in [-1, 1]$. 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 $\|x\|$ in the interval $[\|x\|^L, \|x\|^U]$ as follows.

$$\|x\| = \|x_0 + \Delta\| \leq \|x_0\| + \|\Delta\| \leq \|x_0\| + \sqrt{d}\delta = \|x\|^U,$$
$$\|x\| = \|x_0 + \Delta\| \geq |\|x_0\| - \|\Delta\|| \geq \max(\|x_0\| - \sqrt{d}\delta, 0) = \|x\|^L.$$

Then we can also bound u in $[u^L, u^U]$.

$$\langle x, x' \rangle = \langle x_0 + \Delta, x' \rangle \in \left[\langle x_0, x' \rangle - \sqrt{d}\delta \, \|x'\|, \langle x_0, x' \rangle + \sqrt{d}\delta \, \|x'\| \right],$$

$$u^L = \frac{\langle x_0, x' \rangle - \sqrt{d}\delta \, \|x'\|}{\|x\|^U \, \|x'\|} \quad \text{if } \langle x_0, x' \rangle - \sqrt{d}\delta \, \|x'\| \geq 0 \quad \text{else} \quad \frac{\langle x_0, x' \rangle - \sqrt{d}\delta \, \|x'\|}{\|x\|^L \, \|x'\|},$$

$$u^U = \frac{\langle x_0, x' \rangle + \sqrt{d}\delta \, \|x'\|}{\|x\|^L \, \|x'\|} \quad \text{if } \langle x_0, x' \rangle + \sqrt{d}\delta \, \|x'\| \geq 0 \quad \text{else} \quad \frac{\langle x_0, x' \rangle + \sqrt{d}\delta \, \|x'\|}{\|x\|^U \, \|x'\|},$$

$$u^U = \min(u^U, 1).$$

where $\Delta \in B_\infty(0, \delta)$. $h(u)$ is a bow shaped function so it is easy to get its interval $[h^L(u), h^U(u)]$. Then we can get the interval of $\Theta(x, x')$, denote as $[\Theta^L(x, x'), \Theta^U(x, x')]$.

$$\Theta^L(x, x') = \frac{\|x\|^L \, \|x'\|}{2\pi d} h^L(u) \quad \text{if } h^L(u) \geq 0 \quad \text{else} \quad \frac{\|x\|^U \, \|x'\|}{2\pi d} h^L(u),$$

$$\Theta^U(x, x') = \frac{\|x\|^U \, \|x'\|}{2\pi d} h^U(u) \quad \text{if } h^U(u) \geq 0 \quad \text{else} \quad \frac{\|x\|^L \, \|x'\|}{2\pi d} h^U(u).$$

Suppose the $g(x) = \sum_{i=1}^{n} \alpha_i \Theta(x, x_i)$, α_i are known after solving the kernel machine problem. Then we can lower bound and upper bound $g(x)$ as follows.

$$g(x) \geq \sum_{i=1, \alpha_i > 0}^{n} \alpha_i \Theta^L(x, x_i) + \sum_{i=1, \alpha_i < 0}^{n} \alpha_i \Theta^U(x, x_i), \tag{65}$$

$$g(x) \leq \sum_{i=1, \alpha_i < 0}^{n} \alpha_i \Theta^L(x, x_i) + \sum_{i=1, \alpha_i > 0}^{n} \alpha_i \Theta^U(x, x_i). \tag{66}$$

H.2 IBP for Two-layer Neural Network

See the computation of IBP in [22]. For affine layers of NTK parameterization, the IBP bounds are computed as follows.

$$\begin{aligned}
\mu_{k-1} &= \frac{\overline{z}_{k-1} + \underline{z}_{k-1}}{2} \\
r_{k-1} &= \frac{\overline{z}_{k-1} - \underline{z}_{k-1}}{2} \\
\mu_k &= \frac{1}{\sqrt{m}} W \mu_{k-1} + b \\
r_k &= \frac{1}{\sqrt{m}} \left|W\right| r_{k-1} \\
\underline{z}_k &= \mu_k - r_k \\
\overline{z}_k &= \mu_k + r_k
\end{aligned} \tag{67}$$

where m is the input dimension of that layer. At initialization, W, μ_{k-1} and b are independent. Since $\mathbb{E}\left[W\right] = 0$ and $\mathbb{E}\left[b\right] = 0$,

$$\mathbb{E}\left[\mu_k\right] = \frac{1}{\sqrt{m}} \mathbb{E}\left[W\right] \mathbb{E}\left[\mu_{k-1}\right] + \mathbb{E}\left[b\right] = 0 \tag{68}$$

Since $|W|$ follows a folded normal distribution (absolute value of normal distribution) and $r_{k-1} \geq 0$, $|W| \geq 0$, $\mathbb{E}\left[|W|\right] \mathbb{E}\left[r_{k-1}\right] = O(m)$,

$$\mathbb{E}\left[r_k\right] = \frac{1}{\sqrt{m}} \mathbb{E}\left[|W|\right] \mathbb{E}\left[r_{k-1}\right] = O(\sqrt{m}) \tag{69}$$

Thus

$$-\mathbb{E}\left[\underline{z}_k\right] = -\mathbb{E}\left[\mu_k\right] + \mathbb{E}\left[r_k\right] = O(\sqrt{m}) \tag{70}$$

$$\mathbb{E}\left[\overline{z}_k\right] = \mathbb{E}\left[\mu_k\right] + \mathbb{E}\left[r_k\right] = O(\sqrt{m}) \tag{71}$$

And this will cause the robustness lower bound to decrease at a rate of $O(1/\sqrt{m})$. The same results hold for LeCun initialization, which is used in PyTorch for fully connected layers by default.