# Mathematics of Deep Learning
# Lecture 12: VC Dimension and Generalization, Implicit Bias of Gradient Descent, ResNets and ODEs

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: http://www.ee.columbia.edu/∼predrag

# Learning Infinite Hypothesis Classes

For most hypothesis classes $|\mathcal{H}| = \infty$: What can be done here?

Common measures of complexity of hypothesis classes

- Vapnik-Chervonenkis (VC) dimension
  Chapter 6, 28 & 20 in Shai's [ML2014] book
- Rademacher Complexity
  Chapter 26 in [ML2014] book; this was recently used in
  - A Priori Estimates For Two-layer Neural Networks, by Weinan et al., Jan 2019.
  - Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks, by Arora et al., Jan 2019.
- PAC - Bayes: Chapter 31 in [ML20014]
  used recently in several NN papers, e.g.: see Neyshabur et al.
- Compression Bounds: Chapter 30 in [ML2014]

# VC Dimension

- VC dimension: common characterization of sample complexity
- Introduced by Vapnik & Chervonenkis (VC)
- Can be used to characterize the sample complexity of NNs
- This is an example of bounding the generalization by $\hat{f}$ instead of the target function $f^*$.

# VC Dimension: Definition

- Let $C = \{x_1, \ldots, x_{|C|}\} \subset \mathcal{X}$
- Let $\mathcal{H}_C$ be the restriction of $\mathcal{H}$ to $C$, namely, $\mathcal{H}_C = \{h_C : h \in \mathcal{H}\}$ where $h_C : C \to \{0,1\}$ is s.t. $h_C(x_i) = h(x_i)$ for every $x_i \in C$
- Observe: we can represent each $h_C$ as the vector $(h(x_1), \ldots, h(x_{|C|})) \in \{\pm 1\}^{|C|}$
- Therefore: $|\mathcal{H}_C| \leq 2^{|C|}$
- We say that $\mathcal{H}$ shatters $C$ if $|\mathcal{H}_C| = 2^{|C|}$
- $\mathrm{VCdim}(\mathcal{H}) = \sup\{|C| \ : \ \mathcal{H} \text{ shatters } C\}$
- That is, the VC dimension is the maximal size of a set $C$ such that $\mathcal{H}$ gives no prior knowledge w.r.t. $C$

# VC dimension — Examples

To show that $\mathrm{VCdim}(\mathcal{H}) = d$ we need to show that:

1. There exists a set $C$ of size $d$ which is shattered by $\mathcal{H}$.
2. Every set $C$ of size $d+1$ is not shattered by $\mathcal{H}$.

# VC dimension — Examples

Threshold functions: $\mathcal{X} = \mathbb{R}$, $\mathcal{H} = \{x \mapsto \mathrm{sign}(x - \theta) : \theta \in \mathbb{R}\}$

- Show that $\{0\}$ (or any other one-point set) is shattered
- Show that any two points cannot be shattered

## VC dimension — Examples

Intervals: $\mathcal{X} = \mathbb{R}$, $\mathcal{H} = \{h_{a,b} : a < b \in \mathbb{R}\}$, where $h_{a,b}(x) = 1$ iff $x \in [a, b]$

- Show that $\{0, 1\}$ is shattered
- Show that any three points cannot be shattered

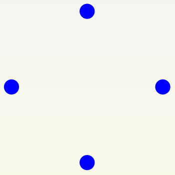
- Note that $\mathcal{H}$ is a 2-parameter class and $\mathrm{VCdim}(\mathcal{H}) = 2$

# VC dimension — Examples

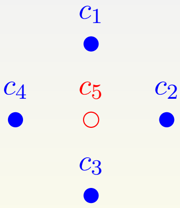
Axis aligned rectangles: $\mathcal{X} = \mathbb{R}^2$,
$\mathcal{H} = \{h_{(a_1,a_2,b_1,b_2)} : a_1 < a_2 \text{ and } b_1 < b_2\}$, where
$h_{(a_1,a_2,b_1,b_2)}(x_1, x_2) = 1$ iff $x_1 \in [a_1, a_2]$ and $x_2 \in [b_1, b_2]$

Show:

Shattered



Not Shattered




Note that $\mathcal{H}$ is a 4-parameter class and $\mathrm{VCdim}(\mathcal{H}) = 4$

# VC dimension — Examples

## Finite classes:

- Show that the VC dimension of a finite $\mathcal{H}$ is at most $\mathrm{VCdim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$
  - since $C$ cannot be shattered if $|\mathcal{H}| < 2^{|C|}$
- There can be arbitrary gap between $\mathrm{VCdim}(\mathcal{H})$ and $\log_2(|\mathcal{H}|)$
  - e.g., consider $\mathcal{X} = \{1, 2, \cdots, k\}$ and consider $\mathcal{H} = \{\text{step functions on } \mathcal{X}\}$
  - Then, $|\mathcal{H}| = k$, but $\mathrm{VCdim}(\mathcal{H}) = 1$

# VC dimension — Examples

Halfspaces: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{H} = \{\mathbf{x} \mapsto \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathbb{R}^d\}$

- Show that $\{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$ is shattered
- Show that any $d+1$ points cannot be shattered
- Hence, $\mathrm{VCdim}(\mathcal{H}) = d$

- Note again that $\mathcal{H}$ is a d-parameter class and $\mathrm{VCdim}(\mathcal{H}) = d$
- In general, one can expect that the VC dimension of a hypothesis class is equal to the number of parameters.

# The Fundamental Theorem of Statistical Learning

## Theorem (Theorem 6.8 in [ML] book)

*Let $\mathcal{H}$ be a hypothesis class of binary classifiers with $\mathrm{VCdim}(\mathcal{H}) = d$. Then, there are absolute constants $C_1, C_2$, s.t.*

1. *$\mathcal{H}$ is (agnostic) PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

2. *(Realizable case) $\mathcal{H}$ is PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

*Furthermore, this sample complexity is achieved by the ERM rule.*

- We gave a sketch of the proof of part 2: the realizable case.
- Part 1. will be covered today. It is based on Rademacher complexity: more specifically Massarat Lemma 26.8 (see Chapter 28 of the book).

# Sauer-Shelah-Perles Lemma

Let

$$\tau_{\mathcal{H}}(m) := \max_{C \in \mathcal{X} : |C| = m} |\mathcal{H}_C|$$

## Lemma (Sauer-Shelah-Perles)

*Let $\mathcal{H}$ be a hypothesis class with $\mathrm{VCdim}(\mathcal{H}) \leq d < \infty$. Then, for all $C \subset \mathcal{X}$ s.t. $|C| = m > d + 1$ we have*

$$\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i} \leq \left(\frac{em}{d}\right)^d$$

- The lemma shows that the maximum number of different vectors that our hypothesis class $\mathcal{H}$ can generate on a data set with $m > d + 1$ points grows polynomially, rather than exponentially in $m$.
- The proof is by induction in $m$, see p. 49 in the [ML] book.
- A more intuitive proof can be found in Bartlett

# Proof of the Upper Bound of the Agnostic Case

- Theorem 6.8-1
- By Sauer's lemma, if VCdim$(\mathcal{H}) = d$, then

$$|\{(h(x_1), \ldots, h(x_m)) : h \in \mathcal{H})\}| \leq \left(\frac{em}{d}\right)^d$$

- Denote

$$A = \{(1_{\{h(x_1) \neq y_1\}}, \ldots, 1_{\{h(x_m) \neq y_m\}}) : h \in \mathcal{H})\}$$

and observe

$$|A| \leq \left(\frac{em}{d}\right)^d$$

# Proof of the Upper Bound of the Agnostic Case

- Now, use the preceding estimate and recall the Massarat Lemma 26.8

$$
\begin{aligned}
R(A) &\leq \max_{\boldsymbol{a}\in A} \|\boldsymbol{a}\|_2 \frac{\sqrt{2\log(|A|)}}{m} \\
&\leq \sqrt{m}\frac{\sqrt{2d\log(em/d)}}{m} = \sqrt{\frac{2d\log(em/d)}{m}}
\end{aligned}
$$

  since $\|\boldsymbol{a}\|_2 \leq \sqrt{m}$

- Finally, the preceding bound and Theorem 26.5 yield, for any $h \in \mathcal{H}$ with probability at least $1-\delta$

$$
L(h) - \hat{L}_S(h) \leq \sqrt{\frac{8d\log(em/d)}{m}} + \sqrt{\frac{2\log(1/\delta)}{m}}
$$

# VC Dimension of Neural Networks

Recall the graph notation for NNs:

- A neural network is obtained by connecting many neurons together
- We focus on feedforward networks, formally defined by a directed acyclic graph $G = (V, E)$
- Input nodes: nodes with no incoming edges
- Output nodes: nodes without out going edges
- weights: $w : E \to \mathbb{R}$
- Calculation using breadth-first-search (BFS), where each neuron (node) receives as input:

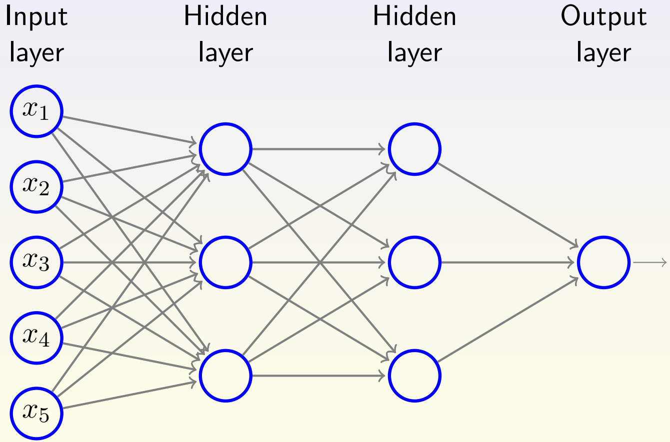
$$a[v] = \sum_{u \to v \in E} w[u \to v] o[u]$$

and output

$$o[v] = \sigma(a[v])$$

# Multilayer Neural Networks

- Neurons are organized in layers: $V = \dot\cup_{t=0}^{T} V_t$, and edges are only between adjacent layers
- Example of a multilayer neural network of depth $3$ and size $6$



# Neural Network Hypothesis Class

- Given a neural network $(V, E, \sigma, w)$, we obtain a hypothesis $h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$
- We refer to $(V, E, \sigma)$ as the architecture, and it defines a hypothesis class by

$$\mathcal{H}_{V,E,\sigma} \ = \ \{h_{V,E,\sigma,w} \ : \ w \text{ is a mapping from } E \text{ to } \mathbb{R}\} \ .$$

- The architecture is our "Prior knowledge" and the learning task is to find the weight function $w$

# Neural Network Sample Complexity

- **Theorem 1:** ($\sigma$ - step/sign function) The VC dimension of $\mathcal{H}_{V,E,\mathrm{sign}}$ is $O(|E|\log(|E|))$.
- **Theorem 2:** ($\sigma$ - any sigmoidal function) The VC dimension of $\mathcal{H}_{V,E,\sigma}$, for $\sigma$ being the sigmoidal function, is $\Omega(|E|^2)$.

- Representation trick: In practice, we only care about networks where each weight is represented using $O(1)$ bits, and therefore the VC dimension of such networks is $O(|E|)$, no matter what $\sigma$ is.

# Neural Network Sample Complexity

**Proof of Theorem 1**:

- Let $\tau_{\mathcal{H}}(m) = \max_{C \in \mathcal{X}:|C|=m} |\mathcal{H}_C|$, where $\mathcal{H}_C$ is the restriction to $C$ of binary valued functions in $\mathcal{H}$
- NN has $T$ layers: $0, 1, 2, \ldots, T$ with $V_t$ nodes at layer $t$.
- Then, $\mathcal{H}$ can be written as a composition

$$\mathcal{H} = \mathcal{H}^{(T)} \circ \cdots \circ \mathcal{H}^{(1)}$$

- Furthermore, each class $\mathcal{H}^{(t)}$ can be decomposed per each neuron

$$\mathcal{H}^{(t)} = \mathcal{H}^{(t,1)} \times \cdots \times \mathcal{H}^{t,|V_t|}$$

# Neural Network Sample Complexity

**Proof of Theorem 1**:

- Then
$$\tau_{\mathcal{H}^{(t)}}(m) \leq \prod_{i=1}^{|V_t|} \tau_{\mathcal{H}^{(t,i)}}(m)$$
- Let $d_{t,i}$ be the number of edges that are headed to the $i$th neuron of layer $t$.
- Since each neuron is a homogenous half-space hypothesis class and the VC dimension of the the half-spaces is the dimension of their input, by Sauer's lemma,
$$\tau_{\mathcal{H}^{(t,i)}}(m) \leq \left(\frac{em}{d_{t,i}}\right)^{d_{t,i}} \leq (em)^{d_{t,i}}$$
implying
$$\tau_{\mathcal{H}}(m) \leq (em)^{\sum_{i,t} d_{t,i}} = (em)^{|E|}$$

# Neural Network Sample Complexity

**Proof of Theorem 1**:

- Now, if we assume that $m$ points are shattered, we must have

$$2^m \leq (em)^{|E|}$$

implying

$$m \leq |E| \log(em) / \log(2)$$

resulting in

$$m \leq O(|E| \log(|E|)),$$

which concludes the proof.

# Generalization Bound for Unregularized NNs

**Theorem** Let $\mathcal{H} = (V, E, \sigma)$ be a hypothesis class of binary classifiers of multilayer NN with step function activation $\sigma$. Then, there are absolute constants $C_1, C_2$, s.t. $\mathcal{H}$ is (agnostic) PAC learnable with sample complexity

$$C_1 \frac{|E| \log(|E|) + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{|E| \log(|E|) + \log(1/\delta)}{\epsilon^2}$$

Furthermore, this sample complexity is achieved by the ERM rule.

- If $\sigma$ is any sigmoid, $|E| \log(|E|)$ should be replaced by $|E|^2$
- Hence, we need either regularization/shrinkage or prior knowledge on the target function to reduce the sample complexity: e.g., Weinan et. al (2019), Neyshabur et. al. (2015-)

# Recent Results on VC-dim of NNs With ReLUs

- Nearly-tight VC-dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks, Barttlet et al., 2019.
- Compute tight upper and lower bounds on the VC-dimension of deep neural networks with the ReLU activation function.
- Let $W$ be the number of weights and $L$ be the number of layers. Then, the paper
  - proves that the VC-dimension is $O(WL\log(W/L))$
  - provides examples with VC-dimension $\Omega(WL\log(W/L))$
- Roughly $\mathrm{VCdim}(\mathcal{H}) \approx WL$

# More Results on GD Convergence and Generalization

A Comparative Analysis of the Optimization and Generalization Property of Two-layer Neural Network and Random Feature Models Under GD, Weinan et al., Feb, 2020.

Key new results:

- Considers both lazy ($\beta = 1/\sqrt{m}$) and active ($\beta = 1/m$) training.

  **Theorem 3.2**: Relaxed the over-parametrisaton assumption $m = \Omega(n^6/(\delta^3\lambda_0^4)$ of Du et al. (2018) to $m = \Omega(n^2/(\delta\lambda_0^4)$

  **Theorem 3.3**: Show that in the over-parametrized regime, $m = \Omega(n^2/(\delta\lambda_0^4)$, the approximations of the network where $w$-s are trained and the one with random initial $w_0$-s are arbitrarily close, i.e.: training yields the same approximation as a fixed kernel method.

  **Theorem 4.1**: ($\beta = c/m$) Relax the over-parametrization, but assume Barron function, and show that early stopping solution, Corollary 4.3, can be close to optimal for mildly over-paramerised networks.

# Recall the Notation

- Training set: $S = \{(x_i, y_i)\}_{i=1}^n$; i.i.d. from a distribution $\rho_{x,y}$
- True (target) function: $f^*(x) = \mathbb{E}[y|x]$, where $y = f(x) + \xi$
- $f^*(x) : [-1, 1]^d \to [0, 1]$
- Two-layer neural network

$$f(x;\theta) = \sum_{k=1}^{m} a_k \sigma(w_k^\top x)$$

  where $w_k \in \mathbb{R}^d, a_k \in \mathbb{R}$ and $\theta = \{(a_k, w_k)\}_{k=1}^m$
  - Scaling: $\mathbb{P}[a_k(0) = \pm\beta] = 1/2$
  - $\beta$ can depend on $m$, e.g.: $\beta = 1/\sqrt{m}$, or $\beta = 1/m$.
- $\sigma(x) : \mathbb{R} \to \mathbb{R}$: activation function
  $\sigma(x)$ scale free: $\sigma(\alpha x) = \alpha\sigma(x), \alpha \geq 0, x \in \mathbb{R}$
  e.g., ReLU or Leaky ReLU

# Training

- Loss function: $\ell(y, y') = (y - y')^2/2$
- Ultimate goal: minimize the population (true) risk

$$L(\theta) = \mathbb{E}_{x,y}[\ell(f(x;\theta), y)]$$

- In practice: minimize the empirical risk

$$\hat{L}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i)$$

- Barron function: A function $f : \Omega \to \mathbb{R}$ is called a Barron if it admits the following representation

$$f(x) = \int_{S^d} a(w)\sigma(w^\top x)d\pi(w),$$

where $\pi$ is a probability distribution over $S^d = \{x : \|x\|_1 = 1\}$ and $a(\cdot)$ is a scalar function.

# Barron Space

- Barron norm: Let $f$ be a Barron function.
  - Denote by $\Theta_f$ all possible representations of $f$
  $$\Theta_f = \left\{ (a, \pi) : f(x) = \int_{S^d} a(w) \sigma(w^\top x) d\pi(w) \right\}$$
  - Barron norm $\gamma_p(f)$:
  $$\gamma_p(f) := \inf_{(a,\pi) \in \Theta_f} \left( \int_{S^d} |a(w)|^p d\pi(w) \right)^{1/p}$$
- Barron space
$$\mathcal{B}_p(\Omega) = \{ f(x) : \gamma_p(f) < \infty \}$$
- Since $\pi$ is a probability distribution, by Hölder's inequality
$$\gamma_p(f) \leq \gamma_q(f), \qquad \text{if } q \geq p > 0.$$
and thus
$$\mathcal{B}_\infty(\Omega) \subset \cdots \subset \mathcal{B}_2(\Omega) \subset \mathcal{B}_1(\Omega)$$

# New Results on GD Convergence of NNs

Relaxed the over-parametrization assumption

- $m = \Omega(n^6/(\delta^3\lambda_0^4)$ of Du et al. (2018) is relaxed to $m = \Omega(n^2/(\delta\lambda_0^4)$

**Theorem 3.2**: For any $0 < \delta < 1$, assume $m = \Omega(n^2 \ln(n^2/\delta)/(\delta\lambda_n^4)$, then, with probability at least $1-\delta$, over the random initialization, we have, for all $t \geq 0$,

$$\hat{L}_n(t) \leq e^{-m\lambda_n t}\hat{L}_n(0),$$

where $\lambda_n > 0$ is the assumption on the lower bound of the corresponding Gram matrices.

# Same as Fixed Kernel Method

- The over-parametrized GD training is the same as fixed kernel method
- **Definitions**:
  - Let $$\hat{f}_{ker}(x,t) := f_m(x; a_t, w_0),$$ where only $a_t$ is optimized and $w_0$ is left unchanged from its initial value.
  - and let $$\hat{f}(x,t) := f_m(x; a_t, w_t),$$ where both $w$ and linear weights, $a$, are optimized.

**Theorem 3.3**: Assume $m = \Omega(n^2 \ln(n^2/\delta)/(\delta\lambda_n^4))$ for $0 < \delta < 1$ and $\beta \leq 1$. Then with probability at least $1 - 6\delta$ we have

$$|\hat{f}(x,t) - \hat{f}_{ker}(x,t)| \lesssim \frac{c_\delta^2}{m}\left(\frac{1}{\sqrt{m}} + \beta + \sqrt{m}\beta^3\right)$$

where $c_\delta = 1 + \sqrt{\ln(1/\delta)}$

**Remark** If $\beta = o(m^{-1/6})$, then the right-hand side goes to zero

# Relaxed Over-parametrization and Early Stopping

- Remove the over-parametrization assumption
- Assume that target function, $f^*$, is Barron's
- and $\|f^*\|_\infty \leq 1$

**Theorem 4.1**: Let $\beta = c/m$, and assume $f^*$ is Barron function, with $\|f^*\|_\infty \leq 1$. then, for any $0 < \delta < 1$, with probability at least $1 - 4\delta$ we have

$$\hat{L}_n(t) \leq C\left(\frac{1}{m} + \frac{1}{mt} + \frac{1}{\sqrt{n}}\right)$$

**Corollary** (Generalization with early stopping) Assume $m > n$ and let $t = \sqrt{n}/m$. Then, under the assumption of Theorem 4.2,

$$L(t) \lesssim \left(\frac{1}{m} + \frac{1}{\sqrt{n}}\right)$$

# Implicit Bias and Margin Maximization of Gradient Descent

- Could it be that we are finding nice generalizable solutions due to GD optimization?
- For linear predictors with linearly separable data, Soudry, Hoffer, and Srebro (2017) show that GD on the cross-entropy loss is implicitly biased towards a maximum margin direction.
  - Bias of GD towards margin maximization means that gradient descent "prefers" a solution which is likely to generalize well, and not just achieve low empirical risk.
- The preceding work inspired many other results, e.g.: Ji and Telgarsky 2019+; Gunasekar et al. 2018; Lyu and Li 2019; Chizat and Bach 2020; Ji et al. 2020.
- In this lecture, I'll cover some of the results from Chapter 10 in Telgarsky, 2021 mongraph.

# Implicit Margin Maximization of Gradient Descent

- Consider $n$ data points: $(y_i, x_i), 1 \le i \le n, y_i \in \{-1, 1\}, x_i \in \mathbb{R}^d$ (For convenience, assume $x_i \equiv \tilde{x}_i = (1, x_{i1}, \ldots, x_{i,d-1})$)
- Assume that data points are *linearly separable*, i.e., there exists $w$, such that

$$\min_i y_i \langle w, x_i \rangle > 0$$

- Typically, there are many such $w$-s. Which one is the best in terms of generalization?
- Maximum margin classifier is given by $\hat{y}(x) = \operatorname{sign}\langle w^*, x \rangle$, where

$$w^* = \operatorname*{argmax}_{\|w\|_2 = 1} \min_i y_i \langle w, x_i \rangle$$

  and the margin is $\gamma := \min_i y_i \langle w^*, x_i \rangle$.
- What is the geometric interpretation of this classifier? Why do we expect it to generalize well? (This is the basis of Support Vector Machines.)
- It turns out that under the appropriate conditions, gradient descent converges to the maximum margin classifier.

# Implicit Margin Maximization of Gradient Descent

- Consider now replacing $\langle w, x_i \rangle$ with $f(x_i; w)$, and margin mapping

$$m_i(w) := y_i f(x_i; w),$$

where $f(x)$ is locally-Lipschitz and L-homogeneous, i.e., $f(cx) = c^L f(x), c \geq 0$.

- For $\ell(z) = e^{-z}$, let $\mathcal{L}$ be unnormalized loss (no division by $n$)

$$\mathcal{L}(w) := \sum_{i=1}^{n} \ell(m_i(w)) = \sum_{i=1}^{n} \ell(y_i f(x_i; w)).$$

- Next, we say that data is $m$-separable it there is a $w$, such that $\min_i m_i(w) > 0$.

- Now, define the (general) margin, maximum margin and smooth margin, respectively as

$$\gamma(w) := \min_i m_i(w/\|w\|) = \frac{\min_i m_i(w)}{\|w\|^L}, \ \bar{\gamma} := \max_{\|w\|=1} \gamma(w), \ \tilde{\gamma} := \frac{\ell^{-1}(\mathcal{L}(w))}{\|w\|^L}.$$

Motivation for smooth margin comes from (recall $\ell$ is exponential): $\ell^{-1}(\mathcal{L}(w)/n) \geq \min_i m_i(w) \geq \ell^{-1}(\max_i \ell(m_i(w))) \geq \ell^{-1}(\mathcal{L}(w))$

# Implicit Margin Maximization of Gradient Descent

Note: Gradient descent is biased towards larger margins, which guarantee good generalization.

**Theorem** (10.1 in Telgarsky, 2021) Consider the linear case, with linearly separable data, exponential loss, $\ell(z) = e^{-z}$, and $max_i \|x_i\| \leq 1$. Then, for gradient descent path $w_t$ with $w(0) = 0$,

$$\gamma(w_t) \geq \tilde{\gamma}(w_t) \geq \bar{\gamma} - \frac{\ln n}{\ln t + \ln(2n\gamma^2) - \ln\ln(2tne\gamma^2)}$$

**Proof**: Consider gradient flow path which satisfy

$$\dot{w}(t) = -\nabla \mathcal{L}(w(t)),$$

which, for $u(t) := \ell^{-1}(\mathcal{L}(w(t)))$, imply ($\ell' = -\ell$ and $\ell^{-1}(z) = -\ln z$)

$$\dot{u}(t) = \left\langle \frac{-\nabla \mathcal{L}(w(t))}{\mathcal{L}(w(t))}, \dot{w}(t) \right\rangle = \frac{\|\dot{w}(t)\|^2}{\mathcal{L}(w(t))}.$$

# Implicit Margin Maximization of Gradient Descent

**Proof**: Now we can lower bound $\gamma(w(t))$ as

$$\gamma(w(t)) \geq \tilde{\gamma}(w(t)) = \frac{u(t)}{v(t)} = \frac{u(0)}{v(t)} + \frac{\int_0^t \dot{u}(s)ds}{v(t)}, \tag{1}$$

where $v(t) := \|w(t)\|$ and $u(t)$ was previously defined. Now, we bound the second term in the preceding equation. To this end, note

$$\begin{aligned} \|\dot{w}(s)\| \geq \langle \dot{w}(s), w^* \rangle &= \left\langle -\sum_i x_i y_i \ell'(m_i(w(s))), w^* \right\rangle \\ &= \sum_i \ell(m_i(w(s))) \langle x_i y_i, w^* \rangle, \quad (\ell' = -\ell) \\ &\geq \gamma \mathcal{L}(w(s)) \end{aligned}$$

Also,

$$v(t) = \left\| \int_0^t \dot{w}(s)ds \right\| \leq \int_0^t \|\dot{w}(s)\| ds$$

# Implicit Margin Maximization of Gradient Descent

**Proof**: Combining the preceding, we lower bound the second term in (1)

$$\frac{\int_0^t \dot{u}(s)ds}{v(t)} = \frac{\int_0^t \frac{\|\dot{w}(s)\|^2}{\mathcal{L}(w(s))}ds}{v(t)} \geq \gamma \frac{\int_0^t \|\dot{w}(s)\|ds}{v(t)} \geq \gamma$$

Since the above inequality holds for any margin $\gamma$, it holds in for the maximum margin $\bar{\gamma}$.
Next, for the first term, $u(0)/v(t)$, in (1), note that $\mathcal{L}(w(0)) = n$, and thus $u(0) = -\ln n$, and it remains to prove that

$$\|w(t)\| \geq \ln(t) + \ln(2n\gamma^2) - 2\ln\ln(2tne\gamma^2).$$

In this regard, we first prove the following lemma.
**Lemma** For any convex loss function $\mathcal{L}$ and any $z \in \mathbb{R}^d$, along the gradient flow path $w(t)$, we have

$$\mathcal{L}(w(t)) \leq \mathcal{L}(z) + \frac{1}{2t}(\|w(0) - z\|_2^2 - \|w(t) - z\|_2^2).$$

# Implicit Margin Maximization of Gradient Descent

**Proof** of the lemma: note that

$$
\begin{aligned}
\frac{1}{2}(\|w(t)-z\|_2^2 - \|w(0)-z\|_2^2) &= \frac{1}{2}\int_0^t \frac{d}{ds}\|w(s)-z\|_2^2 ds \\
&= \int_0^t \left\langle \frac{dw}{ds}, w(s)-z \right\rangle ds \quad \left(\frac{dw}{ds} = -\nabla\mathcal{L}(w(s))\right) \\
&= \int_0^t \langle -\nabla\mathcal{L}(w(s)), w(s)-z \rangle ds \\
&\leq \int_0^t (\mathcal{L}(z) - \mathcal{L}(w(s)) ds \qquad \text{(convexity)} \\
&\leq t\mathcal{L}(z) - t\mathcal{L}(w(t)),
\end{aligned}
$$

where in the second to the last inequality we used the gradient flow assumption $dw(t)/dt = -\nabla\mathcal{L}(w(t))$, and the convexity assumption

$$\mathcal{L}(z) \geq \mathcal{L}(w) + \langle \nabla\mathcal{L}(w), z-w \rangle,$$

which concludes the proof of the lemma.

# Implicit Margin Maximization of Gradient Descent

**Proof**: Now, we complete the proof of the theorem, by applying the preceding lemma with $z = \ln(2tn\gamma^2)w^*/\gamma$, and recall $w(0) = 0$

$$\begin{aligned} n\ell(\|w(t)\|) &\le n \min_i \ell(y_i \langle w(t), x_i \rangle) \le \mathcal{L}(w(t)) \\ &\le \mathcal{L}(z) + \frac{1}{2t}(\|w(0) - z\|_2^2 - \|w(t) - z\|_2^2) \\ &\le \mathcal{L}(z) + \frac{\|z\|_2^2}{2t} \\ &\le \frac{n}{2tn\gamma^2} + \frac{(\ln(2tn\gamma^2))}{2t\gamma^2}, \end{aligned}$$

which, after dividing by $n$, taking $\ell^{-1}(z) = -\ln z$ on both sides, implies

$$\|w(t)\| \ge \ln(2tn\gamma^2) - \ln(1 + \ln(2tn\gamma^2)) = \ln(t) + \ln(2n\gamma^2) - \ln\ln(2tne\gamma^2),$$

which completes the proof of the theorem.

# Implicit Margin Maximization of Gradient Descent

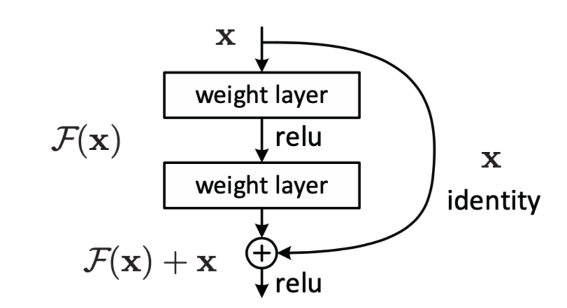
Few remarks:

- The rate of convergence is $1/\ln(t)$, which can be accelerated by rescaling time, Chizat and Bach (2020): see Theorem 10.2 in Telgarsky, 2021.
- Extension to nonlinear homogeneous functions can be found Theorem 10.3 in Telgarsky, 2021. The theorem is originally due to Lyu and Li (2019).
- Extensions to NNs under restrictions can be found in the aforementioned papers:
  - The Implicit Bias of Gradient Descent on Separable Data, Soudry et al., 2017.
  - Gradient Descent Maximizes the Margin of Homogeneous Neural Networks, Lyu, Kaifeng, and Jian Li, 2019.
  - Implicit Bias of Gradient Descent for Wide Two-Layer Neural Networks Trained with the Logistic Loss, Chizat and bach, 2020.
  - For additional references check the follow up references on Google Scholar, and Chapter 10 in Telgarsky 2021.

# Deep Residual Networks

Proposed by: Deep Residual Learning for Image Recognition, He et al., 2015.

- Easier to optimize at larger depth: scale to 100+ depth
- Reduces the problem of vanishing/exploding gradients
- basic building block



# Deep Residual Networks

- Hidden state transformations

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + f(\boldsymbol{h}_t, \theta_t), \tag{2}$$

  $t \in \{0, \dots, T\}$, $\boldsymbol{h}_t \in \mathbb{R}^d$

- $\boldsymbol{h}_0$ is the input layer and $\boldsymbol{h}_T$ is the output layer

$$\boldsymbol{h}_T = \boldsymbol{h}_0 + \sum_{t=0}^{T-1} f(\boldsymbol{h}_t, \theta_t)$$

  The function is additive and the gradient should behave better

# Continuous Approximation to ResNets

Neural Ordinary Differential Equations, by Chen et al., 2018.

- Approximate Equation (2) by a differential equation

$$\frac{d\boldsymbol{h}(t)}{dt} = f(\boldsymbol{h}(t), t, \theta)$$

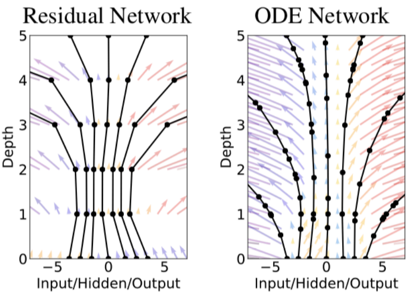



Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

# Continuous Approximation

Claimed advantages

- Can use standard ODE sovers
- Memory efficiency
- Solving ODEs well understood: over 120 years of experience
- Parameter efficiency
- Continuous transformations and change of variables
- Continuous time series models: can incorporate data that arrives at arbitrary times

# Computing the Gradient

- Instead of backpropagation, compute the gradient by solving another (adjoint) ODE backward in time
- Consider optimizing a scalar loss $L(\cdot)$, whose input is the result of an ODE solver
$$L(\boldsymbol{z}(t_1)) = L\left(\boldsymbol{z}(t_0) + \int_{t_0}^{t_1} f(\boldsymbol{z}(t), t, \theta) dt\right)$$
- To optimize $L$, we need a gradient with respect to $\theta$
- So, we first compute the *adjoint*
$$\boldsymbol{a}(t) = \frac{\partial L}{\partial z(t)}$$
- $\boldsymbol{a}(t)$ dynamics is given by another ODE (an instantaneous analog of the chain rule)
$$\frac{d\boldsymbol{a}(t)}{dt} = -\boldsymbol{a}(t)^\top \frac{\partial f(z(t), t, \theta)}{\partial z(t)}$$

# Computing the Gradient

- $\boldsymbol{a}(t)$ dynamics is given by another ODE (an instantaneous analog of the chain rule)
$$\frac{d\boldsymbol{a}(t)}{dt} = -\boldsymbol{a}(t)^{\top} \frac{\partial f(z(t), t, \theta)}{\partial z(t)}$$
- We can compute $\boldsymbol{a}(t) = \partial L / \partial z(t)$ by solving the preceding equation backward in time starting with the initial value $\boldsymbol{a}(t_1) = \partial L / \partial z(t_1)$
- Complication: need to know $z(t)$ along its entire trajectory
  - Recompute $z(t)$ backward in time, together with the adjoint
- Finally, compute the gradient by evaluating the integral
$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \boldsymbol{a}(t)^{\top} \frac{\partial f(z(t), t, \theta)}{\partial z(t)} dt$$

# Reading

- PAC Learning and Generalization Theory - [ML2014] book: VCdim: Chapters 6&28; VCdim of NNs: Theorem 20.6 in Ch. 20.
- Generalization bounds
  - A Comparative Analysis of the Optimization and Generalization Property of Two-layer Neural Network and Random Feature Models Under GD, Weinan et al., Feb, 2020.
- Implicit bias of gradient descent
  - Chapter 15 in Telgarsky, 2021.
  - The Implicit Bias of Gradient Descent on Separable Data, Soudry et al., 2017.
  - Gradient Descent Maximizes the Margin of Homogeneous Neural Networks, Lyu, Kaifeng, and Jian Li, 2019.
  - Implicit Bias of Gradient Descent for Wide Two-Layer Neural Networks Trained with the Logistic Loss, Chizat and bach, 2020.
  - For additional references check the follow up references on Google Scholar, and Chapter 15 in Telgarsky 2021.
- Residual Networks
  - Deep Residual Learning for Image Recognition, He et al., 2015.
  - Neural Ordinary Differential Equations, by Chen et al., 2018.

# Final Project

The key difference from other courses and guiding questions:

- What did you learn about a neural network?
  - The focus should be on NN properties instead of applications.
- How do the changes in NN impact its performance?
  - The changes could be: architecture (e.g., width/depth), activation function, training method, normalization, dropout...
  - In class, we focused on plain vanilla feed-forward networks, but you could choose other types, e.g., ResNets.
- You could center your questions on one or more of the general themes we focused on in class:
  1. Approximation and interpolation theory and the impact of depth.
  2. Optimization landscape and global convergence.
  3. Generalization theory: conditions for small/bounded testing errors.
- Many of the problems we formulated in the context of wide/over-parametrized networks with two types of scaling: NTK/lazy training or mean-field/active training.

# Final Project

Rough Paper Outline, about 15 pages:

1. Introduction: e.g., describe the general problem area, DL and specific subtopic(s). Brief literature review, etc.
2. Detailed Problem Description: More detailed literature review for a selected problem(s), detailed description of the known results, theoretical or experimental, etc.
3. Some Reproduction: Theoretical or Experimental partial or full reproduction of the results. For example, run some simulations that illustrate main results.
4. New Results: Theoretical or Experimental Describe in detail your results. If experimental, describe the experiments and results. Explain clearly the graphs and tables from experiments, etc.
5. Discussion and conclusion: e.g., try to draw general inferences from your results. Compare to the known results from the literature, etc.

# Final Project

- **Deliverables**:
  - **Paper**: about 15 pages - the most important part.
  - **Presentations**: about 10min each, 10 slides
  - **Software**: Document your code well
- **First set presentations**: April 25, during the last class
  3% EC for those presenting on April 25
- Additional presentation slots during study/exam week: TBA
- **Project due**: During the exam week of May 5-12: TBA
- Academic Honesty - do not plagiarize; Turnitin will be used to check for originality

**Have Fun and GOOD LUCK!**