Theoretical aspects of neural networks optimization

Lénaïc Chizat*

July 24th 2019 - IFCAM summer school IISc - Bangalore

*CNRS and Université Paris-Sud

Overview

Introduction

Fully connected neural networks: basics

Infinitely wide static: Gaussian process

Infinitely wide dynamic I: Lazy Training

Infinitely wide nets II: non-linear asymptotic for two layers

Parametric supervised machine learning

- given training data $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i \in \{1, \ldots, n\}$
- build a function h such that $h(x) \approx y$ for unseen data (x, y)
- prediction of the form $h(w, x) \in \mathbb{R}$ with parameters $w \in \mathbb{R}^p$

Examples

- computer vision
- advertisement
- audio processing, natural language processing, medical imaging ...

Models

What are the types of models?

Linear (in the largest sense)

Linear regression, ad-hoc features, random features, kernel methods

$$h(w, x) = w \cdot \phi(x)$$

$\rightsquigarrow$ most of statistics and optimization theory

Models

What are the types of models?

Linear (in the largest sense)

Linear regression, ad-hoc features, random features, kernel methods

$$h(w, x) = w \cdot \phi(x)$$

$\leadsto$ most of statistics and optimization theory

Non-linear

$h(w, x)$ given by a *differentiable program*, such as a *computational graph* (including neural networks)

$\leadsto$ when last operation is linear: interpretation as learnt features

Parametric supervised machine learning

- given training data $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i \in \{1, \dots, n\}$
- build a function h such that $h(x) \approx y$ for unseen data (x, y)
- prediction of the form $h(w, x) \in \mathbb{R}$ with parameters $w \in \mathbb{R}^p$

The goal is to predict well, i.e. minimizing the **population loss**:

$$\min_{w \in \mathbb{R}^p} R(h(w)) := \mathbb{E}_{(x,y)} \mathrm{loss}(y, f(w, x))$$

Parametric supervised machine learning

- given training data $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i \in \{1, \ldots, n\}$
- build a function h such that $h(x) \approx y$ for unseen data (x, y)
- prediction of the form $h(w, x) \in \mathbb{R}$ with parameters $w \in \mathbb{R}^p$

The goal is to predict well, i.e. minimizing the **population loss**:

$$\min_{w \in \mathbb{R}^p} R(h(w)) := \mathbb{E}_{(x,y)} \mathrm{loss}(y, f(w, x))$$

Empirical risk minimization

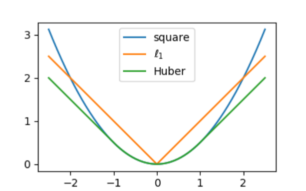
$$\min_{w \in \mathbb{R}^p} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \mathrm{loss}(y_i, h(w, x_i))}_{\text{Data fitting term } \hat{R}(h)} + \underbrace{\lambda \Omega(w)}_{\text{Regularizer}}$$

Losses

Regression ($y \in \mathbb{R}$)

Square loss

$$\mathrm{loss}(y, \bar{y}) = \frac{1}{2}(y - \bar{y})^2$$



$\mathrm{loss}(0, y)$

Losses

Regression ($y \in \mathbb{R}$)

Square loss

$$\mathrm{loss}(y, \bar{y}) = \frac{1}{2}(y - \bar{y})^2$$

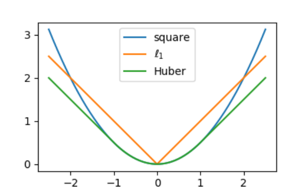
Classification ($y \in \{-1, 1\}$)

Logistic loss

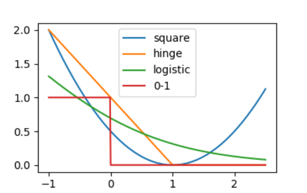
$$\mathrm{loss}(y, \bar{y}) = \log(1 + \exp(-y\bar{y}))$$

Hinge loss

$$\mathrm{loss}(y, \bar{y}) = \max\{0, 1 - y\bar{y}\}$$



$\mathrm{loss}(0, y)$



$\mathrm{loss}(1, y)$

Convexity in supervised machine learning

Convex loss $\circ$ linear predictor $=$ convex objective to minimize

Algorithms

- gradient descent
- stochastic gradient descent (finite sum, infinite sums)
- acceleration, variance reduction

Convexity in supervised machine learning

Convex loss $\circ$ linear predictor $=$ convex objective to minimize

Algorithms

- gradient descent
- stochastic gradient descent (finite sum, infinite sums)
- acceleration, variance reduction

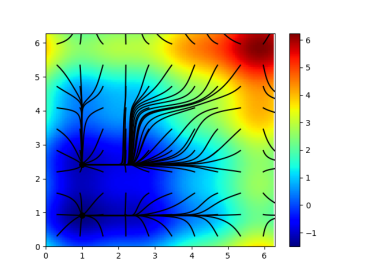
Theory

- global optimization with guaranteed complexity
- matching upper and lower bounds
- useful "black-box" theory

Non-convexity

Convex loss $\circ$ non-linear predictor $\rightsquigarrow$ a new world

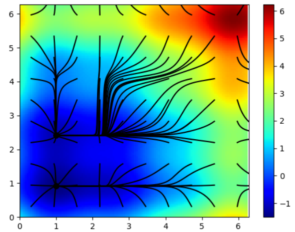
- local minima
- saddle points
- plateaux



Non-convexity

Convex loss $\circ$ non-linear predictor $\rightsquigarrow$ a new world

- local minima
- saddle points
- plateaux



Theory:

- in general, difficult problems (high dimensional exploration)
- some guarantees to escape stationary points
- need to look at the fine structure of the model
- *implicit bias*: optimization and statistics intertwinned

Why should we bother with non-linear models?

Empirical arguments

Optimized features beat fixed, random or hand-designed features

Model	Orig. Accuracy
`pnasnet_large_tf`	82.9 [82.5, 83.2]
`nasnetalarge`	82.5 [82.2, 82.8]
`resnet152`	78.3 [77.9, 78.7]
`inception_v3_tf`	78.0 [77.6, 78.3]
`densenet161`	77.1 [76.8, 77.5]
`vgg19_bn`	74.2 [73.8, 74.6]
`alexnet`	56.5 [56.1, 57.0]
`fv_64k`	35.1 [34.7, 35.5]

Test accuracy for ImageNet classification (Recht et al. 2019)

Why should we bother with non-linear models?

Empirical arguments

Optimized features beat fixed, random or hand-designed features

Model	Orig. Accuracy
pnasnet_large_tf	82.9 [82.5, 83.2]
nasnetalarge	82.5 [82.2, 82.8]
resnet152	78.3 [77.9, 78.7]
inception_v3_tf	78.0 [77.6, 78.3]
densenet161	77.1 [76.8, 77.5]
vgg19_bn	74.2 [73.8, 74.6]
alexnet	56.5 [56.1, 57.0]
fv_64k	35.1 [34.7, 35.5]

Test accuracy for ImageNet classification (Recht et al. 2019)

Theoretical arguments

Breaking the curse of dimensionality: from feature selection to feature construction

[Refs]:
Recht, Roelofs, Schmidt, Shankar (2019). *Do ImageNet Classifiers Generalize to ImageNet?*
Bach (2017). *Breaking the Curse of Dimensionality with Convex Neural Networks.*

Challenges for Theory

Need for new theoretical approaches

- optimization: non-convex (interactions, compositions)
- statistics: over-parameterized, implicit regularization

Challenges for Theory

Need for new theoretical approaches

- optimization: non-convex (interactions, compositions)
- statistics: over-parameterized, implicit regularization

How could theory be useful?

- effects of hyper-parameters
- insights on individual tools in a pipeline
- more robust, more efficient, more accessible models

Today's program

Note for the audience

- between literature survey, lecture and research talk
- we focus on broad ideas, mathematical details are in references
- rapidly evolving field, not mature yet very large

Today's program

Note for the audience

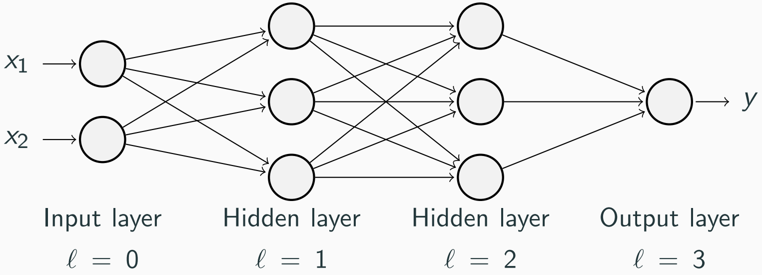
- between literature survey, lecture and research talk
- we focus on broad ideas, mathematical details are in references
- rapidly evolving field, not mature yet very large

Content

- neural nets, backpropagation, initialization
- large width asymptotic (static and dynamic)

Overview

Neural Networks



Fully connected architecture:

$$h(w, x) = W_L \sigma(W^{(L-1)} \sigma(\dots \sigma(W^{(1)} x + b^{(1)} \dots) + b^{(L-1)}) + b^{(L)}$$

- activation $\sigma : \mathbb{R} \to \mathbb{R}$ acts entry-wise
- parameters $w = ((W^{(1)}, b^{(1)}), \dots, (W^{(L)}, b^{(L)})$.

Activation functions

Universal approximation theorem

A 2-layer NN with continuous activation function σ can locally approximate any continuous function to any degree of accuracy if and only if σ is non-polynomial.

[Refs]:
Leshno, Lin, Pinkus, Schoken (1993). *Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function.*

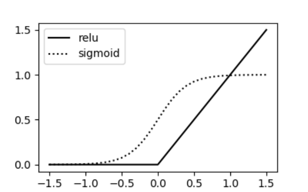
Activation functions

Universal approximation theorem

A 2-layer NN with continuous activation function σ can locally approximate any continuous function to any degree of accuracy if and only if σ is non-polynomial.

Common choices:

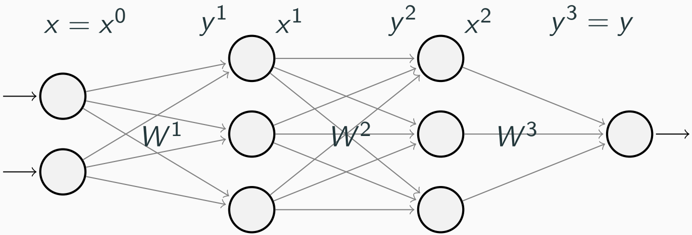
- sigmoid $\sigma(u) = (1 + \exp(-u))^{-1}$
- ReLU $\sigma(u) = \max\{u, 0\} = (u)_+$



[Refs]:

Leshno, Lin, Pinkus, Schoken (1993). *Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function.*

Notation (without bias)



Matrices $W^{(\ell)} \in \mathbb{R}^{m_{\ell-1} \times m_\ell}$ for $0 \leq \ell \leq L$

$$\begin{cases} x^{(\ell)} = h^{(\ell)}(w^{(\ell)}, x^{(\ell-1)}) = \sigma(y^{(\ell)}) \in \mathbb{R}^{m_\ell} \\ y_i^{(\ell)} = \displaystyle\sum_{j=1}^{m_{\ell-1}} W_{ij}^{(\ell)} x_j^{(\ell-1)} \in \mathbb{R}^{m_\ell} \end{cases}$$

where $h^\ell(w^\ell, x^{\ell-1}) = \sigma(W^\ell x^{\ell-1})$, except $h^{(L)} = W^{(L)} x^{(L-1)}$.

Backpropagation

Paradigm: Stochastic Gradient Descent with step-size $\eta > 0$

$$w(k) = w(k-1) - \eta \nabla_w [\mathrm{loss}(h(w(k-1), x(k)), y(k))]$$

Compute the gradient at a sample (x, y) with backpropagation [*blackboard*]

Backpropagation

Paradigm: Stochastic Gradient Descent with step-size $\eta > 0$

$$w(k) = w(k-1) - \eta \nabla_w [\mathrm{loss}(h(w(k-1), x(k)), y(k))]$$

Compute the gradient at a sample (x, y) with backpropagation [*blackboard*] Define $\delta x^{(L)} = \nabla_2 \mathrm{loss}(y, h(w, x))$ and

$$\begin{cases} \delta x_j^{\ell-1} = \displaystyle\sum_{i=1}^{m_\ell} \sigma'(y_i^\ell) \cdot W_{ij}^\ell \cdot \delta x_i^\ell \\ \delta W_{ij}^\ell = x_j^{\ell-1} \cdot \sigma'(y_i^\ell) \cdot \delta x_i^\ell \end{cases}$$

$\rightsquigarrow$ for general computational graphs: automatic differentiation

Initialization

Non convexity: initialization matters.

Pitfalls:

- saddle point at 0 if $L \geq 2$
- signal exploding/vanishing with depth
- gradient exploding/vanishing with depth

[*blackboard*]

Initialization

Non convexity: initialization matters.

Pitfalls:

- saddle point at 0 if $L \geq 2$
- signal exploding/vanishing with depth
- gradient exploding/vanishing with depth

[*blackboard*]

Solution:

Each layer $W_{ij}^{(\ell)}$ i.i.d with mean 0 and variance $2\tau_w^2/n_{\ell-1}$ (for deep ReLU NNs)

[To go deeper]:

Hanin (2018). *Which neural net architectures give rise to exploding and vanishing gradients?*

Hanin, Rolnick (2018). *How to Start Training: The Effect of Initialization and Architecture*

Overview

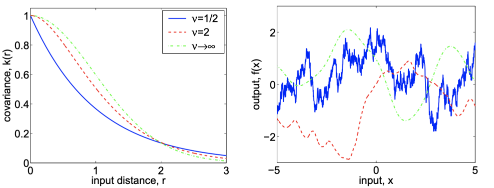
Gaussian process

Definition (Gaussian process)

A random function $f(x)$ is a Gaussian process if and only if for every finite set of points $x_1, x_2, \dots, x_n$ the random vector

$$(f(x_1), f(x_2), \dots, f(x_n))$$

is a multivariate Gaussian random variable. It is characterized by its mean $\mu(x) = \mathbb{E}[f(x)]$ and covariance $\Sigma(x, x') = \mathbb{E}[f(x)f(\bar{x})]$.

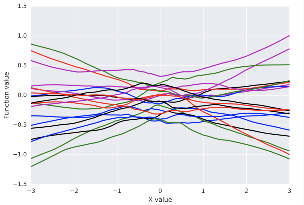


Radial covariances $\Sigma(x, x') = k(|x' - x|)$ and samples of $\mathcal{GP}(0, K)$ (Rasmussen and Williams, 2006)

Comments

[blackboard]

- for training, this randomness is not useful
- however, can be used as random features
- covariance depends on the architecture
- applications to Bayesian inference



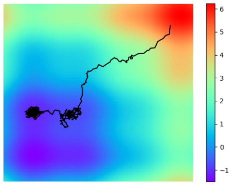
Realization for a relu NN ($L = 10$), (Lee et al. 2018)

[To go deeper]:
Neal (1994). *Priors for infinite networks.*
Lee, Bahri, Novak, Schoenholz, Pennington, Sohl-Dickstein (2018). *Deep neural networks as Gaussian processes.*
Matthews, Hron, Rowland, Turner, Ghahramani (2018). *Gaussian Process Behaviour in Wide Deep Neural Networks*

Overview

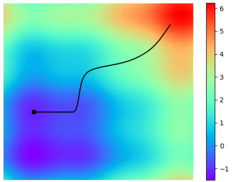
Gradient flow

Objective: $F(w) = \hat{R}(h(w)) = \frac{1}{n}\sum_{k=1}^{n} \mathrm{loss}(y_k, h(w, x_k))$.



SGD: take iid samples (x_k, y_k) and define

$$w(k+1) = w(k) - \eta \nabla_w (\mathrm{loss}(y_k, h(w(k), x_k)))$$



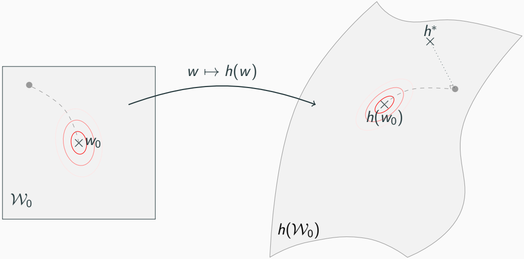
Gradient flow: small η and infinite samples limit

$$\begin{aligned} \frac{d}{dt} w(t) &= -\nabla_w [\hat{R}(h(w(t)))] \\ &= -Dh_{w(t)}^T \nabla \hat{R}(h(w(t))) \end{aligned}$$

$\rightsquigarrow$ gradient flow convenient to get qualitative insights

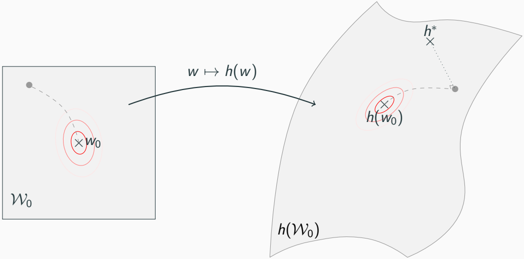
Dynamics of training

Let $h(w)$ be a differentiable model and w_0 an initialization.



Dynamics of training

Let $h(w)$ be a differentiable model and w_0 an initialization.



$$\frac{d}{dt}h(w(t)) = Dh_{w(t)}^T \frac{d}{dt}w(t) = -H_t \nabla \hat{R}(h(w(t)))$$

Definition (Tangent kernel)

$$K_t(x, x') = \langle \nabla h(w(t), x), \nabla h(w(t), x') \rangle \quad \text{and} \quad H_t = [K_t(x_i, x_j)]_{i,j}.$$

Vanilla NN with $W_{i,j}^{(\ell)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \tau_w^2/m_{\ell-1})$ without biases.

Model at initialization

As widths of layers diverge, $h(w_0) \sim \mathcal{GP}(0, \Sigma^L)$ where

$$\Sigma^{\ell+1}(x, x') = \tau_w^2 \cdot \mathbb{E}_{z^\ell \sim \mathcal{GP}(0,\Sigma^\ell)}[\sigma(z^\ell(x)) \cdot \sigma(z^\ell(x'))].$$

Large Neural Networks

Vanilla NN with $W_{i,j}^{(\ell)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \tau_w^2/m_{\ell-1})$ without biases.

Model at initialization

As widths of layers diverge, $h(w_0) \sim \mathcal{GP}(0, \Sigma^L)$ where

$$\Sigma^{\ell+1}(x, x') = \tau_w^2 \cdot \mathbb{E}_{z^\ell \sim \mathcal{GP}(0,\Sigma^\ell)}[\sigma(z^\ell(x)) \cdot \sigma(z^\ell(x'))].$$

Limit tangent kernel

In the same limit, $\langle \nabla_w h(w_0, x), \nabla_w h(w_0, x') \rangle \to K_{0,\infty}(x, x')$ where

$$K_{0,\infty}(x, x') = \sum_{\ell=1}^{L} \left(\Sigma^{(\ell-1)}(x, x') \prod_{\ell'=\ell}^{L} \dot{\Sigma}^{(\ell')}(x, x') \right)$$

and $\dot{\Sigma}^{\ell+1}(x, x') = \mathbb{E}_{z^\ell \sim \mathcal{GP}(0,\Sigma^\ell)}[\dot{\sigma}(z^\ell(x)) \cdot \dot{\sigma}(z^\ell(x'))]$.

[Refs]:
Arora, Do, Hu, Li (2019). *On Exact Computation with an Infinitely Wide Neural Network.*
Jacot, Gabriel, Hongler (2018). *Neural Tangent Kernel: Convergence and Generalization in Neural Networks.*

Infinitely wide dynamics

Theorem (Jacot et al. 2018, reformulated)

Assume that σ has a Lipschitz derivative and consider the gradient flow for the population loss. Then, for all $T > 0$, as the width of all layers grow unbounded, one has uniformly on $[0, T]$ and on compacts of the input space,

$$\langle \nabla_w h(w_t, x), \nabla_w h(w_t, x') \rangle \to K_{0,\infty}(x, x').$$

For the square loss $R(h) = \frac{1}{2}\|h - h^\star\|^2$, this gives in this limit,

$$\frac{d}{dt} h_t = -H_{0,\infty}(h_t - h^\star)$$

- not mentioned: some subtlety with layer-wise learning rate
- can be extended to cover ReLU activation

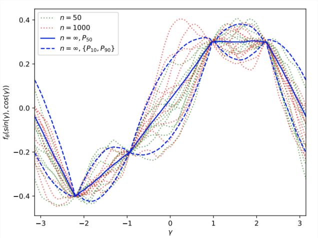
[Ref]:
Jacot, Gabriel, Hongler (2018). *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*
Arora, Do, Hu, Li (2019). *On Exact Computation with an Infinitely Wide Neural Net*

A word on quadratic gradient flow

- with our categorization: linear method (features not learnt)
- consider a p.s.d matrix $H \in \mathbb{R}^n \times \mathbb{R}^n$, and the gradient flow, with $h(0) \in \mathbb{R}^n$,

$$\frac{d}{dt}h(t) = -H(h(t) - h^\star)$$

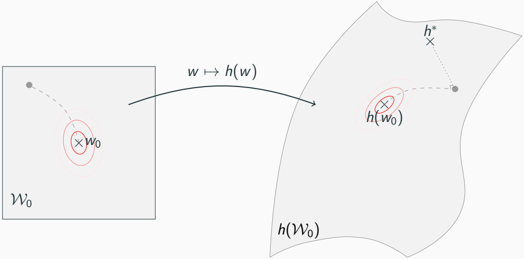
  has a closed form [*blackboard*]
- convergence to an interpolating function if H has full rank



Predictor learnt by a ReLU NN [Jacot et al. 2018]

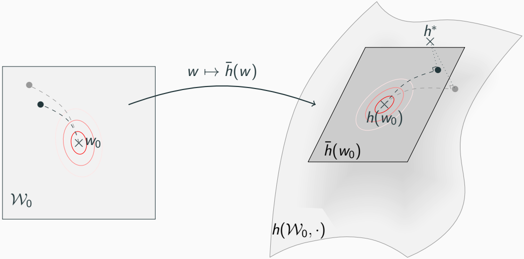
Tangent Model

Let $h(w, x)$ be a differentiable model and w_0 an initialization.



Tangent Model

Let $h(w, x)$ be a differentiable model and w_0 an initialization.



Tangent model

$$\bar{h}(w, x) = h(w_0, x) + (w - w_0) \cdot \nabla_w h(w_0, x)$$

Scaling the output by α makes the linearization more accurate.

Lazy Training Theorem

Theorem (Lazy training through rescaling)

Assume that $h(w_0, \cdot) = O(1/\alpha)$ and that the loss is quadratic. In the limit of a large scale α, the gradient flow of the non-linear model αh and on the tangent model $\bar{h}$ learn the same model, up to a $O(1/\alpha)$ remainder.

- instance of *implicit bias*: *lazy* because parameters hardly move
- in optimization: whenever $R(h(w))$ behaves like $R(\bar{h}(w))$
- linear models are better understood (training, generalization)
- recovers kernel ridgeless regression with offset $f(w_0, \cdot)$ and

$$K(x, x') = \langle \nabla_w h(w_0, x), \nabla_w h(w_0, x') \rangle$$

[Refs]:

Jacot, Gabriel, Hongler (2018). *Neural Tangent Kernel: Convergence and Generalization in Neural Networks.*
Du, Lee, Li, Wang, Zhai (2018). *Gradient Descent Finds Global Minima of Deep Neural Networks.*
Allen-Zhu, Li, Liang (2018). *Learning and Generalization in Overparameterized Neural Networks [...].*
Chizat, Bach (2018). *A Note on Lazy Training in Supervised Differentiable Programming.*

When does lazy training occurs?

Relative scale criterion

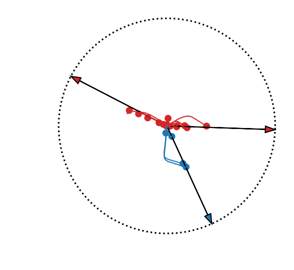
For the square loss $\frac{1}{2}\|y - y^\star\|^2$:

$$\kappa_h(w_0) := \frac{\|h(w_0) - h^\star\|}{\|\nabla h(w_0, \cdot)\|} \frac{\|\nabla^2 h(w_0, \cdot)\|}{\|\nabla h(w_0, \cdot)\|} \ll 1$$

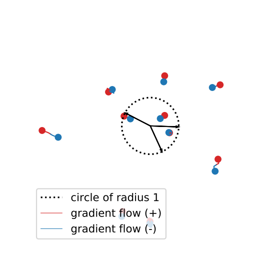
Examples

- *Homogeneous models with $h(w_0, \cdot) = 0$.*
  If for $\lambda > 0$, $h(\lambda w, x) = \lambda^L h(w, x)$, then $\kappa_h(w_0) \asymp 1/\|w_0\|^L$
- *Wide two-layer NNs with iid weights, $\mathbb{E}\Phi(w_i, \cdot) = 0$.*
  If $h(w, x) = \alpha(m) \sum_{i=1}^m \Phi(w_i, x)$, then $\kappa_h(w_0) \asymp (m\alpha(m))^{-1}$
- *Deep NNs with large layers.*
  Similar principle at play.

Numerical Illustrations (I)



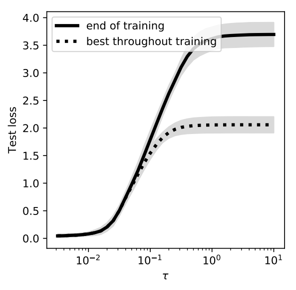
(a) Non-lazy



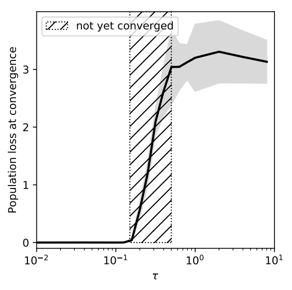
(b) Lazy

Figure 1: Training trajectories of a 2-layers ReLU NN in the teacher-student setting. Gradient descent on a finite data set.

Numerical experiment (I)



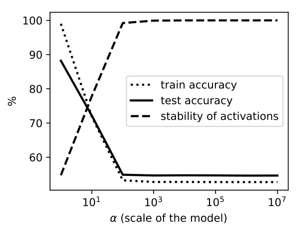
(a) Over-parameterized (GD)

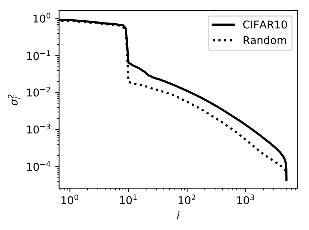


(b) Under-parameterized (SGD)

Training a 2-layers ReLU NN in the teacher-student setting generalization in 100-d vs init. scale τ. (a) finite data set, gradient descent (b) pure SGD (directly minimizes the population loss).

Numerical experiment (II)





(a) Accuracy vs scale **(b)** Spectrum of the tangent kernel

Training a (VGG) convolutional neural network on the CIFAR data set.

Lessons to be drawn

For practice

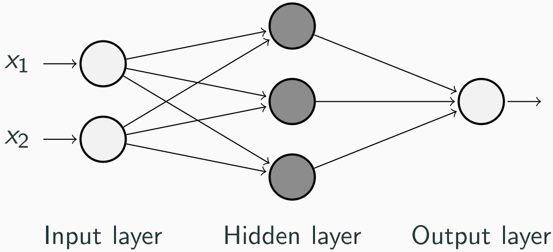
- very wide networks may lead to kernel (linear) methods
- non-linear training seems essential for state of the art NN

For theory

- in depth and quantitative analysis sometimes possible
- not just one theory for NNs training

Overview

Two Layers NNs



With activation σ, define $\phi(w_i, x) = W_i^{(2)} \sigma \left(\sum W_{ij}^{(1)} x_j + b_i \right)$ and

$$h(w, x) = \frac{1}{m} \sum_{i=1}^{m} \phi(w_i, x)$$

Hard problem: existence of spurious minima even with slight over-parameterization and good initialization

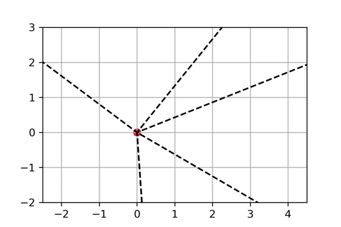
[Refs]:
Livni, Shalev-Shwartz, Shamir (2014). *On the Computational Efficiency of Training Neural Networks.*
Safran, Shamir (2018). *Spurious Local Minima are Common in Two-layer ReLU Neural Networks.*

Mean-Field Analysis

Many-particle limit

Training dynamics in the small step-size and infinite width limit:

$$\mu_{t,m} = \frac{1}{m}\sum_{i=1}^{m} \delta_{w_i(t)} \underset{m\to\infty}{\to} \mu_{t,\infty}$$



[Refs]:

Mei, Montanari, Nguyen (2018). *A Mean Field View of the Landscape of Two-Layers Neural Networks.*

Rotskoff, Vanden-Eijndem (2018). *Parameters as Interacting Particles [...].*

Sirignano, Spiliopoulos (2018). *Mean Field Analysis of Neural Networks.*

Chizat, Bach (2018) *On the Global Convergence of Gradient Descent for Over-parameterized Models [...]*

Global Convergence

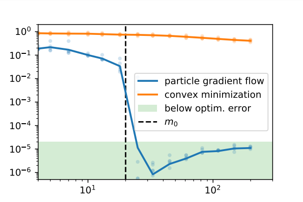
Theorem (Global convergence, informal)

In the limit of a large hidden layer, the gradient flow of a 2-layer NN initialized with "sufficient diversity" converges to a global minimizer.

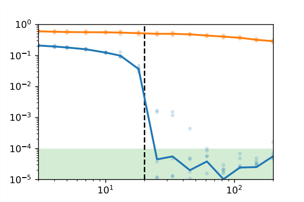
- diversity at initialization is key for success of training
- highly non-linear dynamics and regularization allowed
- for the population loss, this means: lowest test loss over all 2-layer NNs

[Refs]:
Chizat, Bach (2018). *On the Global Convergence of Gradient Descent for Over-parameterized Models [...].*

Numerical Illustrations

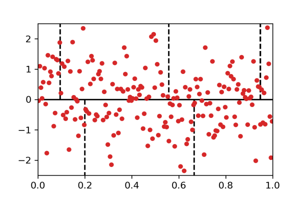


(a) ReLU



(b) Sigmoid

Population loss at convergence vs m for training a 2-layers NN in the teacher-student setting in 100-d.



This is a general principle for convex optimization on measures.

Wasserstein Gradient Flow

- parameterize the model with a probability measure μ:

$$h(\mu, x) = \int \phi(w, x)\mathrm{d}\mu(w), \quad \mu \in \mathcal{P}(\mathbb{R}^d)$$

Wasserstein Gradient Flow

- parameterize the model with a probability measure μ:

$$h(\mu, x) = \int \phi(w, x)\mathrm{d}\mu(w), \quad \mu \in \mathcal{P}(\mathbb{R}^d)$$

- consider the population loss over $\mathcal{P}(\mathbb{R}^d)$:

$$F(\mu) := R(h(\mu)) = \mathbb{E}_{(x,y)}\left[\mathrm{loss}\left(\int \phi(w, x)\mathrm{d}\mu(w), y\right)\right].$$

$\rightsquigarrow$ convex in linear geometry but non-convex in Wasserstein

Wasserstein Gradient Flow

- parameterize the model with a probability measure μ:

$$h(\mu, x) = \int \phi(w, x) \mathrm{d}\mu(w), \quad \mu \in \mathcal{P}(\mathbb{R}^d)$$

- consider the population loss over $\mathcal{P}(\mathbb{R}^d)$:

$$F(\mu) := R(h(\mu)) = \mathbb{E}_{(x,y)} \left[\mathrm{loss} \left(\int \phi(w, x) \mathrm{d}\mu(w), y \right) \right].$$

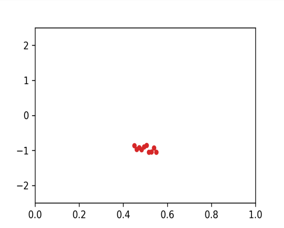
$\rightsquigarrow$ convex in linear geometry but non-convex in Wasserstein

- define the Wasserstein gradient flow :

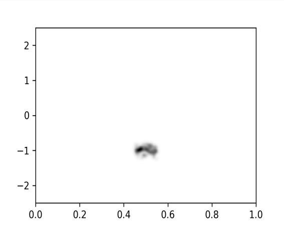
$$\mu_0 \in \mathcal{P}(\mathbb{R}^d), \qquad \frac{d}{dt}\mu_t = -\mathrm{div}(\mu_t v_t)$$

where $v_t(w) = -\nabla F'(\mu_t)$ is the Wasserstein gradient of F.

[Refs]:
Bach (2017). *Breaking the Curse of Dimensionality with Convex Neural Networks.*
Ambrosio, Gigli, Savaré (2008). *Gradient Flows in Metric Spaces and in the Space of Probability Measures.*



$$\xrightarrow[m\to\infty]{}$$



Quadratic loss

When $R(h) = \frac{1}{2}\|h - h^*\|^2$ for some $h^* \in \mathcal{F}$, interpretation as a system of charged particles with *varying* charge and interaction

$$k((r_1, \theta_1), (r_2, \theta_2)) = r_1 r_2 \langle \Phi(\theta_1), \Phi(\theta_2) \rangle_{\mathcal{F}}.$$

[Refs:]
Ambrosio, Gigli, Savaré (2008). *Gradient flows in metric spaces and in the space of probability measures.*
Panigrahy, Rahimi, Sachdeva, Zhang (2017). *Convergence results for neural networks via electrodynamics.*

Mean-Field Limit for SGD

Now consider the actual training trajectory ((x_k, y_k) i.i.d):

$$\begin{cases} w(k) = w(k-1) - \eta m \nabla_w [\mathrm{loss}(h(w(k-1), x(k)), y(k))] \\ \hat{\mu}_m(k) = \dfrac{1}{m} \displaystyle\sum_{i=1}^{m} \delta_{w_i(k)} \end{cases}$$

Theorem (Mei, Montanari, Nguyen '18)

Under regularity assumptions, if $w_1(0), w_2(0), \ldots$ are drawn independently accordingly to μ_0 then with probability $1 - e^{-z}$,

$$\|\hat{\mu}_m^{(\lfloor t/\eta \rfloor)} - \mu_t\|_{BL}^2 \lesssim e^{Ct} \max\left\{\eta, \frac{1}{m}\right\} \left(z + d + \log \frac{m}{\eta}\right)$$

[Refs]:

Mei, Montanari, Nguyen (2018). *A Mean Field View of the Landscape of Two-Layers Neural Networks.*

Global Convergence (more formal)

Theorem (Homogeneous case)

Assume that μ_0 is supported on a centered sphere or ball, that ϕ is 2-homogeneous in the weights and some regularity. If μ_t converges in Wasserstein distance to μ_∞ then μ_∞ is a global minimizer of F. In particular, if $w_1(0), w_2(0), \ldots$ are drawn accordingly to μ_0 then

$$\lim_{m,t\to\infty} F(\mu_{t,m}) = \min F.$$

- applies to 2-layers ReLU NNs
- different statement for sigmoid NNs

[Refs]:
Chizat, Bach (2018). *On the Global Convergence of Gradient Descent for Over-parameterized Models [...].*

Remark on the scaling

Change of parameterization/initialization $\Rightarrow$ change of behavior.

		Mean field	Lazy
model	$h(w,x)$	$\frac{1}{m}\sum \phi(w_i,x)$	$\frac{1}{\sqrt{m}}\sum \phi(w_i,x)$
init. predictor	$\|h(w_0,\cdot)\|$	$O(1/\sqrt{m})$	$O(1)$
scale	κ_0^{-1}	$O(1)$	$O(\sqrt{m})$
displacement	$\|w_\infty - w_0\|$	$O(1)$	$O(1/\sqrt{m})$

NB: the $1/m$ scaling cannot be used with i.i.d. initialization and $L > 2$.

Generalization

Through regularization

In regression tasks, adaptivity to subspace when minimizing

$$\min_{\mu\in\mathcal{P}(\mathbb{R}^d)} \frac{1}{n}\sum_{i=1}^{n}\left|\int \phi(w,x_i)\mathrm{d}\mu(w) - y_i\right|^2 + \int V(w)\mathrm{d}\mu(w)$$

where ϕ is ReLU activation and V a ℓ_1-type regularizer.

$\rightsquigarrow$ explicit sample complexity bounds for regression

$\rightsquigarrow$ also some bounds under separability assumptions

[Refs]:
Bach (2017). *Breaking the Curse of Dimensionality with Convex Neural Networks.*
Wei, Lee, Liu, Ma (2018). *Regularization Matters: Generalization and Optimization of Neural Nets v.s. their Induced Kernel.*

Lessons to be drawn

For practice

- over-parameterization/random init. yields global convergence
- choice of scalings is crucial

For theory

- strong generalization guaranties need neurons that move
- non-quantitative technics still leads to insights

What I did not talk about

Focus was on gradient-based training in “realistic” settings.

Wide range of other approaches

- loss landscape analysis
- linear neural networks
- phase transition/computational barriers
- tensor decomposition
- ...

[Refs]:

Arora, Cohen, Golowich, Hu (2018). *Convergence Analysis of Gradient Descent for Deep Linear Neural Networks*

Aubin, Maillard, Barbier, Krzakala, Macris, Zdeborová (2018). *The Committee Machine: Computational to Statistical Gaps in Learning a Two-layers Neural Network.*

Zhang, Yu, Wang, Gu (2018). *Learning One-hidden-layer ReLU Networks via Gradient Descent.*

Conclusion

- **several regimes, several theories**
- **calls for new tools from mathematics**

Perspectives for research

- **how deep NN optim. works is mostly open**
- **optimization of compositional models?**