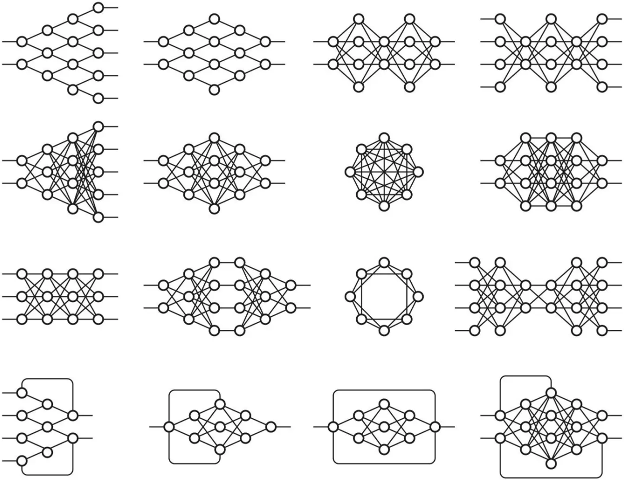
Bidirectionally Self-Normalizing Neural Networks

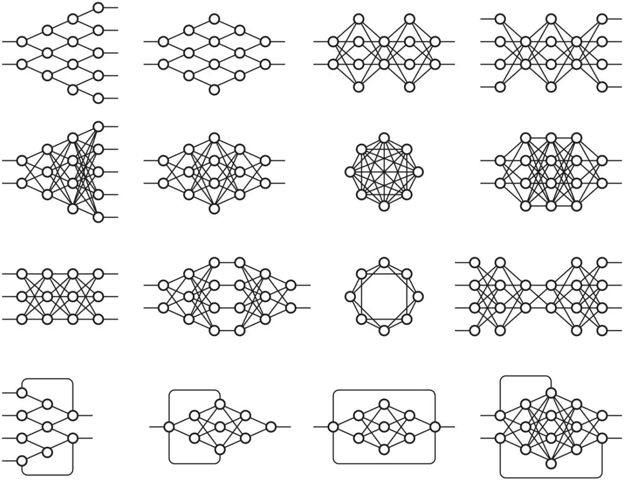
Yao Lu

Peking University & ANU & CSIRO/Data61

Neural Networks



Neural Networks

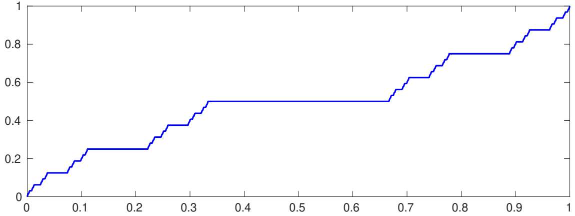


Universal Function Approximator

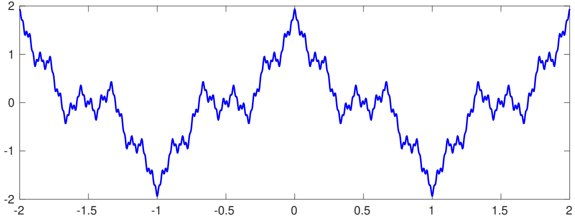
$F : \mathbb{R} \to \mathbb{R}$ is continuous

$F : \mathbb{R} \to \mathbb{R}$ is continuous

Cantor function



Weierstrass function



Function Approximation

$F : \mathbb{R} \to \mathbb{R}$ is continuous

$$F(x) \approx f_N(x) + f_{N-1}(x) + ... + f_0(x)$$

Function Approximation

$F : \mathbb{R} \to \mathbb{R}$ is continuous

$$F(x) \approx f_N(x) + f_{N-1}(x) + ... + f_0(x)$$

- $f_n(x) = a_n x^n$ Polynomial
- $f_n(x) = a_n \cos(nx) + b_n \sin(nx)$ Fourier series

Function Approximation

$F : \mathbb{R}^d \to \mathbb{R}$ is continuous

$$F(\mathbf{x}) \approx f_N(\mathbf{x}) + f_{N-1}(\mathbf{x}) + ... + f_0(\mathbf{x})$$

Function Approximation

$F : \mathbb{R}^d \to \mathbb{R}$ is continuous

$$F(\mathbf{x}) \approx f_N(\mathbf{x}) + f_{N-1}(\mathbf{x}) + ... + f_0(\mathbf{x})$$

- $f_n(\mathbf{x}) = a_n \phi(\|\mathbf{x} - \mathbf{x}_n\|)$ Radial basis function
- $f_n(\mathbf{x}) = a_n K(\mathbf{x}, \mathbf{x}_n)$ Kernel method

Superposition

$$F(\mathbf{x}) \approx f_N(\mathbf{x}) + f_{N-1}(\mathbf{x}) + ... + f_0(\mathbf{x})$$

Superposition

$$F(\mathbf{x}) \approx f_N(\mathbf{x}) + f_{N-1}(\mathbf{x}) + ... + f_0(\mathbf{x})$$

Composition

$$F(\mathbf{x}) \approx f_N \circ f_{N-1} \circ ... \circ f_0(\mathbf{x})$$

Superposition

$$F(\mathbf{x}) \approx f_N(\mathbf{x}) + f_{N-1}(\mathbf{x}) + \ldots + f_0(\mathbf{x})$$

Composition

$$F(\mathbf{x}) \approx f_N \circ f_{N-1} \circ \ldots \circ f_0(\mathbf{x})$$

Math difficulties

- Approximation
  What F and f? How deep? How wide? How accurate?

Superposition

$$F(\mathbf{x}) \approx f_N(\mathbf{x}) + f_{N-1}(\mathbf{x}) + ... + f_0(\mathbf{x})$$

Composition

$$F(\mathbf{x}) \approx f_N \circ f_{N-1} \circ ... \circ f_0(\mathbf{x})$$

Math difficulties

- Approximation
  What F and f? How deep? How wide? How accurate?

- Optimization
  How to choose $\boldsymbol{\theta}_n$ in $f_n(\mathbf{x}, \boldsymbol{\theta}_n)$?

Problem: Vanishing/Exploding Gradients

Problem: Vanishing/Exploding Gradients

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

where $\mathbf{x}^{(1)}$ is the input and $\mathbf{x}^{(L+1)}$ is the output

Problem: Vanishing/Exploding Gradients

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

where $\mathbf{x}^{(1)}$ is the input and $\mathbf{x}^{(L+1)}$ is the output

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}$$

Problem: Vanishing/Exploding Gradients

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

where $\mathbf{x}^{(1)}$ is the input and $\mathbf{x}^{(L+1)}$ is the output

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Problem: Vanishing/Exploding Gradients

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

where $\mathbf{x}^{(1)}$ is the input and $\mathbf{x}^{(L+1)}$ is the output

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Gradient

$$\frac{\partial E}{\partial \mathbf{W}^{(l)}} = \mathbf{y}^{(l)} \mathbf{x}^{(l)T}$$

Problem: Vanishing/Exploding Gradients

Problem: Vanishing/Exploding Gradients

A simple network of 20 layers of 500 units, $\phi(x) = \frac{1}{1+\exp(-x)}$

Problem: Vanishing/Exploding Gradients

A simple network of 20 layers of 500 units, $\phi(x) = \frac{1}{1+\exp(-x)}$

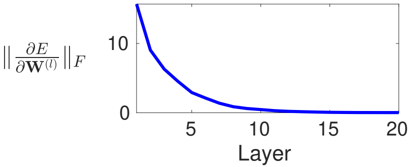
$\mathbf{x}^{(1)} \sim \mathcal{N}(0, \mathbf{I})$ and $\mathbf{t} \sim \mathcal{N}(0, \mathbf{I})$

Problem: Vanishing/Exploding Gradients

A simple network of 20 layers of 500 units, $\phi(x) = \frac{1}{1+\exp(-x)}$

$\mathbf{x}^{(1)} \sim \mathcal{N}(0, \mathbf{I})$ and $\mathbf{t} \sim \mathcal{N}(0, \mathbf{I})$

- $\mathbf{W}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$

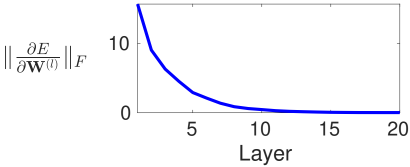


Problem: Vanishing/Exploding Gradients

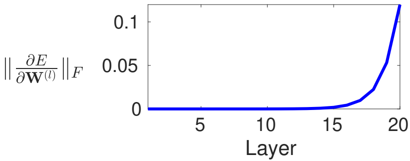
A simple network of 20 layers of 500 units, $\phi(x) = \frac{1}{1+\exp(-x)}$

$\mathbf{x}^{(1)} \sim \mathcal{N}(0, \mathbf{I})$ and $\mathbf{t} \sim \mathcal{N}(0, \mathbf{I})$

- $\mathbf{W}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$



- $\mathbf{W}^{(l)} \sim \mathcal{N}(0, 0.01\mathbf{I})$



Problem: Vanishing/Exploding Gradients

Gradients have the same scale $\rightarrow$ easy to solve

Problem: Vanishing/Exploding Gradients

Gradients have the same scale $\rightarrow$ easy to solve

Example

$$\min_{\boldsymbol{\theta}} \boldsymbol{\theta}^T \mathbf{Q} \boldsymbol{\theta}$$

where

$$\boldsymbol{\theta} = (\theta_1, \theta_2), \quad \mathbf{Q} = \begin{pmatrix} 0.01 & 0 \\ 0 & 1 \end{pmatrix}$$

Problem: Vanishing/Exploding Gradients

Gradients have the same scale $\rightarrow$ easy to solve

Example

$$\min_{\boldsymbol{\theta}} \boldsymbol{\theta}^T \mathbf{Q} \boldsymbol{\theta}$$

where

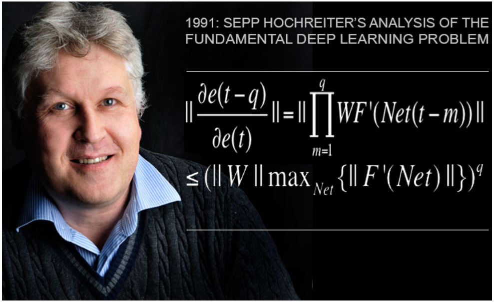
$$\boldsymbol{\theta} = (\theta_1, \theta_2), \quad \mathbf{Q} = \begin{pmatrix} 0.01 & 0 \\ 0 & 1 \end{pmatrix}$$

Gradient Descent

$$\theta_1 \leftarrow (1 - 0.01\eta)\theta_1$$
$$\theta_2 \leftarrow (1 - \eta)\theta_2$$

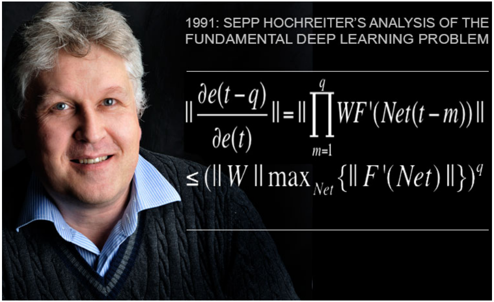
Problem: Vanishing/Exploding Gradients

Sepp Hochreiter (1991)



Problem: Vanishing/Exploding Gradients

Sepp Hochreiter (1991)



"**His work formally showed that deep neural networks are hard to train, because they suffer from the now famous problem of vanishing or exploding gradients**"

Sepp Hochreiter's Fundamental Deep Learning Problem

–Jürgen Schmidhuber

Problem: Vanishing/Exploding Gradients

A simple solution

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \frac{\partial E}{\partial \mathbf{W}^{(l)}} \Big/ \left\| \frac{\partial E}{\partial \mathbf{W}^{(l)}} \right\|_F$$

Problem: Vanishing/Exploding Gradients

A simple solution

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \frac{\partial E}{\partial \mathbf{W}^{(l)}} \Big/ \left\| \frac{\partial E}{\partial \mathbf{W}^{(l)}} \right\|_F$$

Drawbacks

- for fixed η, it does not converge

Problem: Vanishing/Exploding Gradients

A simple solution

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \frac{\partial E}{\partial \mathbf{W}^{(l)}} \Big/ \left\| \frac{\partial E}{\partial \mathbf{W}^{(l)}} \right\|_F$$

Drawbacks

- for fixed η, it does not converge
- for adaptive η, it is hard to tune learning rate schedule

Problem: Vanishing/Exploding Gradients

Tricks

- adaptive gradients (e.g., Adam)
- batch normalization
- gradient clipping
- shortcut connections

The Vanishing/Exploding Gradients problem is provably solved for deep nonlinear networks!

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Gradient

$$\frac{\partial E}{\partial \mathbf{W}^{(l)}} = \mathbf{y}^{(l)}\mathbf{x}^{(l)T}$$

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Gradient

$$\frac{\partial E}{\partial \mathbf{W}^{(l)}} = \mathbf{y}^{(l)}\mathbf{x}^{(l)T}$$

Idea

Constrain $\mathbf{x}^{(l)}$ and $\mathbf{y}^{(l)}$

Definition (Bidirectional Self-Normalization)

$$\|\mathbf{x}^{(1)}\|_2 = \|\mathbf{x}^{(2)}\|_2 = \ldots = \|\mathbf{x}^{(L)}\|_2$$
$$\|\mathbf{y}^{(1)}\|_2 = \|\mathbf{y}^{(2)}\|_2 = \ldots = \|\mathbf{y}^{(L)}\|_2$$

Definition (Bidirectional Self-Normalization)

$$\|\mathbf{x}^{(1)}\|_2 = \|\mathbf{x}^{(2)}\|_2 = ... = \|\mathbf{x}^{(L)}\|_2$$
$$\|\mathbf{y}^{(1)}\|_2 = \|\mathbf{y}^{(2)}\|_2 = ... = \|\mathbf{y}^{(L)}\|_2$$

Proposition

If a neural network is bidirectionally self-normalizing, then

$$\left\|\frac{\partial E}{\partial \mathbf{W}^{(1)}}\right\|_F = \left\|\frac{\partial E}{\partial \mathbf{W}^{(2)}}\right\|_F = ... = \left\|\frac{\partial E}{\partial \mathbf{W}^{(L)}}\right\|_F$$

How to enforce the constraints?

$$\|\mathbf{x}^{(1)}\|_2 = \|\mathbf{x}^{(2)}\|_2 = \dots = \|\mathbf{x}^{(L)}\|_2$$

$$\|\mathbf{y}^{(1)}\|_2 = \|\mathbf{y}^{(2)}\|_2 = \dots = \|\mathbf{y}^{(L)}\|_2$$

If $\phi(x) = x$

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \mathbf{h}^{(l)}$$

Backward pass

$$\mathbf{y}^{(L)} = \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

If $\phi(x) = x$

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \mathbf{h}^{(l)}$$

Backward pass

$$\mathbf{y}^{(L)} = \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Then $\mathbf{W}^{(l)}$ is orthogonal

If $\phi(x)$ is nonlinear

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Can $\|\mathbf{x}^{(l)}\|_2$ and $\|\mathbf{y}^{(l)}\|_2$ be preserved?

If $\phi(x)$ is nonlinear

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Can $\|\mathbf{x}^{(l)}\|_2$ and $\|\mathbf{y}^{(l)}\|_2$ be preserved?

No, in general!

Mazur–Ulam Theorem

If V and W are normed space over $\mathbb{R}$ and the mapping

$$f : V \to W$$

is surjective isometry, then f is affine.

If $\phi(x)$ is nonlinear

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

Backward pass

$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Can $\|\mathbf{x}^{(l)}\|$ and $\|\mathbf{y}^{(l)}\|$ be preserved?

No, in general!

If $\phi(x)$ is nonlinear

Forward pass

$$\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)}, \quad \mathbf{x}^{(l+1)} = \phi(\mathbf{h}^{(l)})$$

Backward pass

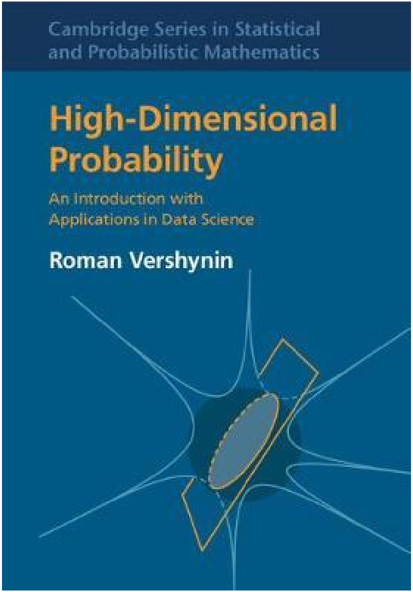
$$\mathbf{y}^{(L)} = \phi'(\mathbf{h}^{(L)}) \circ \frac{\partial E}{\partial \mathbf{x}^{(L+1)}}, \quad \mathbf{y}^{(l)} = \phi'(\mathbf{h}^{(l)}) \circ (\mathbf{W}^{(l+1)})^T \mathbf{y}^{(l+1)}$$

Can $\|\mathbf{x}^{(l)}\|$ and $\|\mathbf{y}^{(l)}\|$ be preserved?

No, in general!

Yes, roughly! $\|\mathbf{x}^{(l+1)}\|_2 \approx \|\mathbf{x}^{(l)}\|_2$ and $\|\mathbf{y}^{(l+1)}\|_2 \approx \|\mathbf{y}^{(l)}\|_2$.

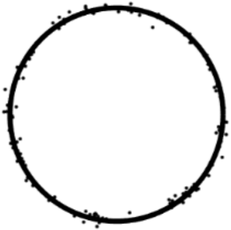
High-Dimensional Probability



High-Dimensional Probability

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$$

Low-dimensional



High-dimensional

Concentration of Measure

High-Dimensional Probability

Lemma

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz and* $\mathbb{E}_{z \sim \mathcal{N}(0,1)}[f(z)^2] = 1$

$$\|f(\mathbf{z})\|_2 \approx \|\mathbf{z}\|_2 \text{ as } d \to \infty$$

High-Dimensional Probability

Lemma

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz and* $\mathbb{E}_{z \sim \mathcal{N}(0,1)}[f(z)^2] = 1$

$$\|f(\mathbf{z})\|_2 \approx \|\mathbf{z}\|_2 \textit{ as } d \to \infty$$

```
z = torch.randn(10000)
f = 1.4674 * torch.tanh(z) + 0.3885
print(z.norm(), f.norm())
```

High-Dimensional Probability

Lemma

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz and* $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[f(z)^2] = 1$

$$\|f(\mathbf{z})\|_2 \approx \|\mathbf{z}\|_2 \text{ as } d \to \infty$$

```
z = torch.randn(10000)
f = 1.4674 * torch.tanh(z) + 0.3885
print(z.norm(), f.norm())

tensor(98.8555) tensor(99.8824)
tensor(99.2121) tensor(98.8777)
tensor(100.5818) tensor(99.9690)
```

High-Dimensional Probability

Lemma

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz and* $\mathbb{E}_{z \sim \mathcal{N}(0,1)}[f(z)^2] = 1$
- $\mathbf{x} \in \mathbb{R}^d$ *with bounded* $\|\mathbf{x}\|_\infty$

$$\|f(\mathbf{z}) \circ \mathbf{x}\|_2 \approx \|\mathbf{x}\|_2 \text{ as } d \to \infty$$

High-Dimensional Probability

Lemma

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz and* $\mathbb{E}_{z \sim \mathcal{N}(0,1)}[f(z)^2] = 1$
- $\mathbf{x} \in \mathbb{R}^d$ *with bounded* $\|\mathbf{x}\|_\infty$

$$\|f(\mathbf{z}) \circ \mathbf{x}\|_2 \approx \|\mathbf{x}\|_2 \text{ as } d \to \infty$$

```
z = torch.randn(10000)
f = 1.4674 * torch.tanh(z) + 0.3885
x = torch.rand(10000)
y = f * x
print(x.norm(), y.norm())
```

High-Dimensional Probability

Lemma

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz and* $\mathbb{E}_{z \sim \mathcal{N}(0,1)}[f(z)^2] = 1$
- $\mathbf{x} \in \mathbb{R}^d$ *with bounded* $\|\mathbf{x}\|_\infty$

$$\|f(\mathbf{z}) \circ \mathbf{x}\|_2 \approx \|\mathbf{x}\|_2 \text{ as } d \to \infty$$

```
z = torch.randn(10000)
f = 1.4674 * torch.tanh(z) + 0.3885
x = torch.rand(10000)
y = f * x
print(x.norm(), y.norm())

tensor(57.6663) tensor(58.2298)
tensor(58.2302) tensor(58.2693)
tensor(57.5398) tensor(57.9497)
```

Lemma

If $\mathbf{W}$ *is orthogonal and uniformly distributed and* $\|\mathbf{x}\|_2 = \sqrt{d}$, *then*

$$\mathbf{W}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \text{ as } d \to \infty$$

Lemma

If $\mathbf{W}$ *is orthogonal and uniformly distributed and* $\|\mathbf{x}\|_2 = \sqrt{d}$, *then*

$$\mathbf{W}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \text{ as } d \to \infty$$

```
Z = torch.randn(5000, 5000)
Z = Z / Z.pow(2).sum(0, True).sqrt()
U, _, V = torch.svd(Z, compute_uv=True)
W = U @ V.t
```

Lemma

If $\mathbf{W}$ *is orthogonal and uniformly distributed and* $\|\mathbf{x}\|_2 = \sqrt{d}$, *then*

$$\mathbf{W}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \text{ as } d \to \infty$$

```
Z = torch.randn(5000, 5000)
Z = Z / Z.pow(2).sum(0, True).sqrt()
U, _, V = torch.svd(Z, compute_uv=True)
W = U @ V.t

x = torch.ones(5000, 1)
y = W @ x
```

Lemma

If $\mathbf{W}$ *is orthogonal and uniformly distributed and* $\|\mathbf{x}\|_2 = \sqrt{d}$, *then*

$$\mathbf{W}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \text{ as } d \to \infty$$

```
Z = torch.randn(5000, 5000)
Z = Z / Z.pow(2).sum(0, True).sqrt()
U, _, V = torch.svd(Z, compute_uv=True)
W = U @ V.t

x = torch.ones(5000, 1)
y = W @ x

plt.hist(y.numpy(), bins=100)
plt.show()
```

Lemma

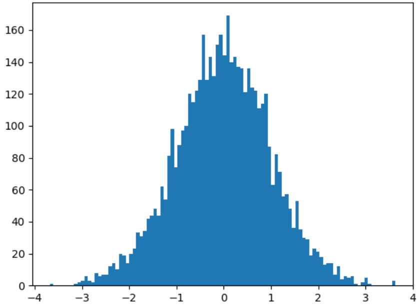
If $\mathbf{W}$ *is orthogonal and uniformly distributed and* $\|\mathbf{x}\|_2 = \sqrt{d}$, *then*

$$\mathbf{W}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \text{ as } d \to \infty$$

Lemma

If $\mathbf{W}$ is orthogonal and uniformly distributed and $\|\mathbf{x}\|_2 = \sqrt{d}$, then

$$\mathbf{W}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \text{ as } d \to \infty$$



- $\|\mathbf{x}\|_2 \approx \sqrt{d}$
- $\mathbf{W}$ is orthogonal and uniformly distributed
- ϕ and ϕ' are Lipschitz
- $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$

- $\|\mathbf{x}\|_2 \approx \sqrt{d}$
- $\mathbf{W}$ is orthogonal and uniformly distributed
- ϕ and ϕ' are Lipschitz
- $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$

Theorem (Forward Norm-Preservation)

Random vector

$$\|\phi(\mathbf{W}\mathbf{x})\|_2 \to \sqrt{d}$$

as $d \to \infty$.

- $\|\mathbf{x}\|_2 \approx \sqrt{d}$
- $\mathbf{W}$ is orthogonal and uniformly distributed
- ϕ and ϕ' are Lipschitz
- $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$

Theorem (Forward Norm-Preservation)

Random vector

$$\|\phi(\mathbf{W}\mathbf{x})\|_2 \to \sqrt{d}$$

as $d \to \infty$.

Theorem (Backward Norm-Preservation)

Let $\mathbf{D} = \mathrm{diag}(\phi'(\mathbf{w}_1^T\mathbf{x}), \ldots, \phi'(\mathbf{w}_d^T\mathbf{x}))$ and $\mathbf{y} \in \mathbb{R}^d$ be a fixed vector with bounded $\|\mathbf{y}\|_\infty$. Then

$$\|\mathbf{D}\mathbf{y}\|_2^2 \to \|\mathbf{y}\|_2^2$$

as $d \to \infty$.

Gaussian-Poincaré Normalization

$$\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$$

Gaussian-Poincaré Normalization

$$\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$$

Proposition

For almost any φ, there exist two constants a and b that

$$\phi(x) = a\varphi(x) + b$$

such that

$$\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$$

Gaussian-Poincaré Normalization

$$\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)^2] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi'(z)^2] = 1$$

	Tanh	ReLU	LeakyReLU	ELU	SELU	GELU
a	1.4674	1.4142	1.4141	1.2234	0.9660	1.4915
b	0.3885	0.0000	0.0000	0.0742	0.2585	-0.9097

Experiments

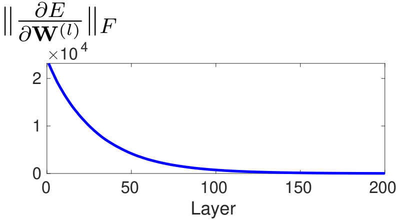
Simple network of 200 layer of 500 units with orthogonal $\mathbf{W}^{(l)}$

$$\mathbf{x}^{(1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

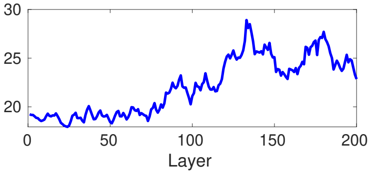
Experiments

Simple network of 200 layer of 500 units with orthogonal $\mathbf{W}^{(l)}$

$$\mathbf{x}^{(1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



SELU



SELU-GPN

Experiments

	MNIST		CIFAR-10	
	Train	Test	Train	Test
Tanh	99.05 (87.39)	**96.57** (89.32)	80.84 (27.90)	**42.71** (29.32)
Tanh-GPN	**99.81** (84.93)	95.54 (87.11)	**96.39** (25.13)	40.95 (26.58)
ReLU	11.24 (11.24)	11.35 (11.42)	10.00 (10.00)	10.00 (10.00)
ReLU-GPN	**33.28** (11.42)	**28.13** (11.34)	**46.60** (10.09)	**34.96** (9.96)
LeakyReLU	11.24 (11.24)	11.35 (11.63)	10.00 (10.21)	10.00 (10.06)
LeakyReLU-GPN	**43.17** (11.19)	**49.28** (11.66)	**51.85** (9.89)	**39.38** (10.00)
ELU	99.06 (98.24)	95.41 (**97.48**)	80.73 (42.39)	**45.76** (44.16)
ELU-GPN	**100.00** (97.86)	96.56 (96.69)	**99.37** (43.35)	43.12 (44.36)
SELU	99.86 (97.82)	97.33 (97.38)	29.23 (46.47)	29.55 (45.88)
SELU-GPN	**99.92** (97.91)	96.97 (**97.39**)	**98.24** (47.74)	**45.90** (45.52)
GELU	11.24 (12.70)	11.35 (10.28)	10.00 (10.43)	10.00 (10.00)
GELU-GPN	**97.67** (11.22)	**95.82** (9.74)	**90.51** (10.00)	**36.94** (10.00)

Table 1: Accuracy (percentage) of neural networks of depth 200 and width 500 with different activation functions on real-world data. The numbers in parenthesis denote the results when batch normalization is applied before the activation function.

Summary

We theoretically solved the vanishing/exploding gradients problem!

Summary

We theoretically solved the vanishing/exploding gradients problem!

Limitations

- Assumptions holds only at initialization
- Constraining the networks too much
- Only for MLP