Mathematics of Deep Learning

Introduction & general overview

Lessons: Kevin Scaman
TDs: Mathieu Even

Dauphine UNIVERSITÉ PARIS | PSL

Practical details

Timeline

- **Dates:** 03/01/2023 - 21/02/2023 (13h45 - 17h)
- **Format:** 7 classes (1h30 class + 1h30 TDs), 1 Exam (21/02)
- **Room:** Salle 08 (Paris Santé campus)

Validation

- One **homework** on 24/01. **Deadline:** 07/02.
- One **exam** on the 21/02.

Contact

- **Email:** `kevin.scaman@ens.fr`

Class overview

1. **Introduction and general overview** 03/01
2. Non-convex optimization 10/01
3. Structure of ReLU networks and group invariances 17/01
4. Approximation guarantees 24/01
5. Stability and robustness 31/01
6. Infinite width limit of NNs 07/02
7. Generative models 14/02
8. Exam 21/02

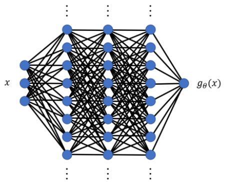
What is Deep Learning?

What is Deep Learning?

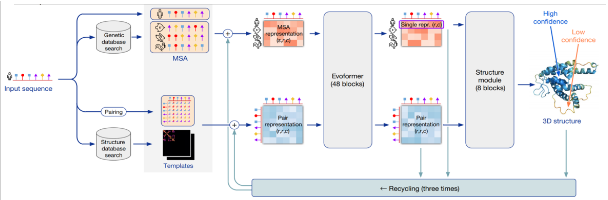
First, what are neural networks?

- The notion changed over the last 8 decades...!
- From early neural networks imitating real neurons...
- To highly complex architectures with multiple sub-modules.

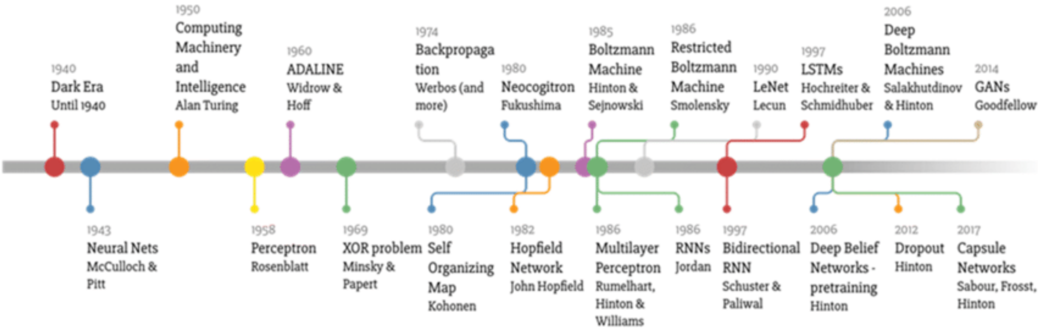
Multi-Layer Perceptron
(Rumelhart, Hinton, Williams, 75)



AlphaFold
(Jumper et.al., 2021)



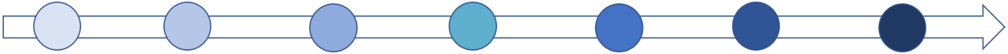
Timeline of Deep Learning

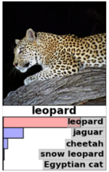


source: Mourtzis & Angelopoulos (2020)

Recent deep learning applications

Image classification	Image generation	Text generation	Board games (Go, chess)	Strategy games (Starcraft, Dota2)	Code generation	Protein folding
2012	...	...	...	...	...	2021

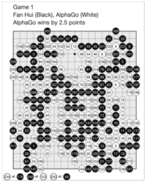




[Krizhevsky et.al., 2012]

[Goodfellow et.al., 2014]
[Karras et.al., 2018]
[news.artnet.com, 2018]

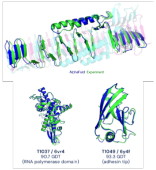
[Vaswani et.al., 2017]
[Hochreiter et.al., 2018]
[Devlin et.al., 2019]



[Silver et.al., 2016]
[Silver et.al., 2017]

[openai.com/five, 2018]
[Oriol et.al., 2019]

[Brown et.al., 2020]
[Chen et.al., 2021]



[Jumper et.al., 2021]

Most recent breakthrough: image generation (Dalle2, Stable diffusion, MidJourney, ...)

Images generated from prompts using MidJourney (https://www.midjourney.com/)

What is Deep Learning? (twitter wisdom)

Yann LeCun
@ylecun

Some folks still seem confused about what deep learning is. Here is a definition:

DL is constructing networks of parameterized functional modules & training them from examples using gradient-based optimization....
facebook.com/722677142/post...

Traduire le Tweet

4:32 PM · 24 déc. 2019 · Facebook

What is Deep Learning? (twitter wisdom)

Yann LeCun
@ylecun

Some folks still seem confused about what deep learning is. Here is a definition:

DL is constructing networks of parameterized functional modules & training them from examples using gradient-based optimization....
facebook.com/722677142/post...

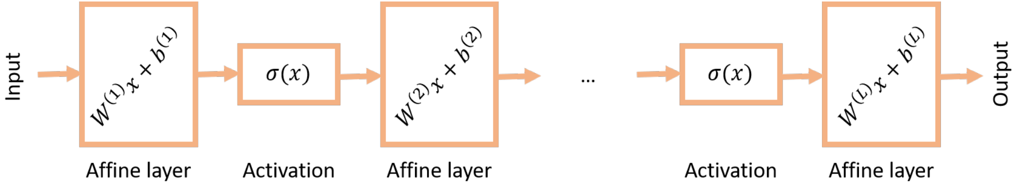
Traduire le Tweet

4:32 PM · 24 déc. 2019 · Facebook

Mathematical formulation

Recap of the ML training pipeline, NN formulation and loss functions

Multi-Layer Perceptron (Rumelhart, Hinton, Williams, 75)



Details

- We will denote as $L \geqslant 1$ the number of affine layers.
- The case $L = 1$ creates affine models.
- Activations are computed coordinate-wise $(\sigma(x)_i = \sigma(x_i))$.
- A *"neuron"* is a coordinate of the output of an activation layer.
- $W^{(l)}$ and $b^{(l)}$ are learnt during training.

Multi-Layer Perceptron: formal definition

Definition (MLP)

Let $L \geqslant 1$, $(d^{(l)})_{l\in[\![0,L]\!]} \in {\mathbb{N}^*}^{L+1}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ a non-linear activation function. A *Multi-Layer Perceptron* (MLP) of depth L, layer dimensions $(d^{(l)})_{l\in[\![0,L]\!]}$ and activation σ is a function $g_\theta : \mathbb{R}^{d^{(0)}} \to \mathbb{R}^{d^{(L)}}$ of the form:

$$g_\theta(x) = f^{(2L-1)} \circ f^{(2L-2)} \circ \cdots \circ f^{(2)} \circ f^{(1)}(x)$$

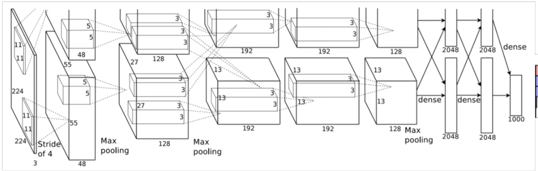
where $\forall l \in [\![1, L]\!]$, $f^{(2l-1)}(x) = W^{(l)}x + b^{(l)}$, $f^{(2l)}(x) = \sigma(x)$, $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$, $b^{(l)} \in \mathbb{R}^{d^{(l)}}$.

- Its parameter is $\theta = \left(W^{(l)}, b^{(l)}\right)_{l\in[\![1,L]\!]}$.
- We denote as $g_\theta^{(l)}(x) = f^{(l)} \circ \cdots \circ f^{(1)}(x)$ the intermediate output after layer $l \in [\![0, 2L-1]\!]$.

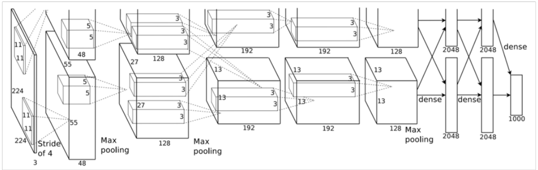
Typical Machine Learning setup

$(x_i)_{i \in [1,n]}$ $g_\theta : X \to Y$ $(y_i)_{i \in [1,n]}$

Training



Testing



?

AlexNet *(Krizhevsky et.al., 2012)*

Typical Machine Learning setup

Data distribution

Let $\mathcal{X}, \mathcal{Y}$ be an input and output space and $\mathcal{D}$ a distribution over $(\mathcal{X}, \mathcal{Y})$. Then, we denote our (test) input/output pair as

$$(X, Y) \sim \mathcal{D}$$

Typical Machine Learning setup

Data distribution

Let $\mathcal{X}, \mathcal{Y}$ be an input and output space and $\mathcal{D}$ a distribution over $(\mathcal{X}, \mathcal{Y})$. Then, we denote our (test) input/output pair as

$$(X, Y) \sim \mathcal{D}$$

Risk minimization (a.k.a. supervized ML)

The objective of *risk minimization* is to find a minimizer $\theta^* \in \mathbb{R}^p$ of the optimization problem

$$\min_{\theta \in \mathbb{R}^p} \mathbb{E}\big(\ell(g_\theta(X), Y)\big)$$

where $\ell : \mathcal{Y}^2 \to \mathbb{R}_+$ is a loss function and $g_\theta : \mathcal{X} \to \mathcal{Y}$ a model parameterized by $\theta \in \mathbb{R}^p$.

Typical Machine Learning setup

Data distribution

Let $\mathcal{X}, \mathcal{Y}$ be an input and output space and $\mathcal{D}$ a distribution over $(\mathcal{X}, \mathcal{Y})$. Then, we denote our (test) input/output pair as

$$(X, Y) \sim \mathcal{D}$$

Risk minimization (a.k.a. supervized ML)

The objective of *risk minimization* is to find a minimizer $\theta^* \in \mathbb{R}^p$ of the optimization problem

$$\min_{\theta \in \mathbb{R}^p} \mathbb{E}\big(\ell(g_\theta(X), Y)\big)$$

where $\ell : \mathcal{Y}^2 \to \mathbb{R}_+$ is a loss function and $g_\theta : \mathcal{X} \to \mathcal{Y}$ a model parameterized by $\theta \in \mathbb{R}^p$.

The target loss (e.g. accuracy) may be hard to train, and can thus be different from the one used as objective during training!

Loss functions

Mean Square Error vs. Cross Entropy

Typical Machine Learning setup

- For example, $\ell(y, y') = \mathbb{1}\{y \neq y'\}$ gives the **classification error** (i.e. 1 - accuracy).

Typical Machine Learning setup

- For example, $\ell(y, y') = \mathbb{1}\{y \neq y'\}$ gives the **classification error** (i.e. 1 - accuracy).
- For **classification** tasks, we usually use $\mathcal{Y} = \mathbb{R}^C$ where C is the number of classes, and
  - $\ell(y, y') = \mathbb{1}\{\mathrm{argmax}_i\, y'_i \neq \mathrm{argmax}_i\, y_i\}$ (top-1 classification error) or,
  - $\ell(y, y') = -\sum_i y'_i \ln\left(\exp(y_i)/\sum_j \exp(y_j)\right)$ (cross entropy).

Typical Machine Learning setup

- For example, $\ell(y, y') = \mathbb{1}\{y \neq y'\}$ gives the **classification error** (i.e. 1 - accuracy).
- For **classification** tasks, we usually use $\mathcal{Y} = \mathbb{R}^C$ where C is the number of classes, and
  - $\ell(y, y') = \mathbb{1}\{\text{argmax}_i\, y'_i \neq \text{argmax}_i\, y_i\}$ (top-1 classification error) or,
  - $\ell(y, y') = -\sum_i y'_i \ln\left(\exp(y_i)/\sum_j \exp(y_j)\right)$ (cross entropy).
- For **regression** tasks, we usually use $\mathcal{Y} = \mathbb{R}^d$ and
  - $\ell(y, y') = \|y - y'\|_2^2 = \sum_i (y_i - y'_i)^2$ (mean square error) or,
  - $\ell(y, y') = \|y - y'\|_1 = \sum_i |y_i - y'_i|$ (mean absolute error).

Mean square error (MSE): probabilistic interpretation

- **Definition:** $\ell(x, y) = \|x - y\|_2^2$.
- **Probabilistic model:** Assume that there is a $\theta \in \mathbb{R}^d$ such that

$$Y_i = g_\theta(X_i) + \varepsilon_i$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ are i.i.d. centered Gaussian random variables (mean 0 and variance σ^2), and X_i are i.i.d. and independent of θ.

Mean square error (MSE): probabilistic interpretation

- **Definition:** $\ell(x, y) = \|x - y\|_2^2$.
- **Probabilistic model:** Assume that there is a $\theta \in \mathbb{R}^d$ such that

$$Y_i = g_\theta(X_i) + \varepsilon_i$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ are i.i.d. centered Gaussian random variables (mean 0 and variance σ^2), and X_i are i.i.d. and independent of θ.

- **Maximum Likelihood Estimation:** The likelihood for the data to be drawn from a given θ is

$$\mathbb{P}_\theta((X_i, Y_i)) \;=\; \prod_i \mathbb{P}(X_i)\mathbb{P}_\theta(\varepsilon_i = Y_i - g_\theta(X_i)) \propto \exp\left(\frac{-\sum_i \|Y_i - g_\theta(X_i)\|_2^2}{2\sigma^2}\right)$$

Mean square error (MSE): probabilistic interpretation

- **Definition:** $\ell(x, y) = \|x - y\|_2^2$.
- **Probabilistic model:** Assume that there is a $\theta \in \mathbb{R}^d$ such that

$$Y_i = g_\theta(X_i) + \varepsilon_i$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ are i.i.d. centered Gaussian random variables (mean 0 and variance σ^2), and X_i are i.i.d. and independent of θ.

- **Maximum Likelihood Estimation:** The likelihood for the data to be drawn from a given θ is

$$\mathbb{P}_\theta((X_i, Y_i)) = \prod_i \mathbb{P}(X_i)\mathbb{P}_\theta(\varepsilon_i = Y_i - g_\theta(X_i)) \propto \exp\left(\frac{-\sum_i \|Y_i - g_\theta(X_i)\|_2^2}{2\sigma^2}\right)$$

- Maximizing the log-likelihood is equivalent to **minimizing the MSE**.

Cross entropy: probabilistic interpretation

- **Definition:** $\ell(x, y) = -\log\left(\frac{\exp(x_y)}{\sum_i \exp(x_i)}\right)$.
- **Probabilistic model:** Assume that there is a $\theta \in \mathbb{R}^d$ such that, for all classes $k \in [\![1, C]\!]$,

$$\log \mathbb{P}(Y_i = k \mid X_i) \propto g_\theta(X_i)_k$$

where X_i are i.i.d. and independent of θ.

Cross entropy: probabilistic interpretation

- **Definition:** $\ell(x, y) = -\log\left(\frac{\exp(x_y)}{\sum_i \exp(x_i)}\right)$.
- **Probabilistic model:** Assume that there is a $\theta \in \mathbb{R}^d$ such that, for all classes $k \in [\![1, C]\!]$,

$$\log \mathbb{P}(Y_i = k \mid X_i) \propto g_\theta(X_i)_k$$

  where X_i are i.i.d. and independent of θ.
- **Maximum Likelihood Estimation:** The likelihood for the data to be drawn from a given θ is

$$\mathbb{P}_\theta((X_i, Y_i)) = \prod_i \mathbb{P}(X_i)\mathbb{P}_\theta(Y_i \mid X_i) \propto \prod_i \frac{\exp(g_\theta(X_i)_{Y_i})}{\sum_k \exp(g_\theta(X_i)_k)}$$

Cross entropy: probabilistic interpretation

- **Definition:** $\ell(x, y) = -\log\left(\frac{\exp(x_y)}{\sum_i \exp(x_i)}\right)$.
- **Probabilistic model:** Assume that there is a $\theta \in \mathbb{R}^d$ such that, for all classes $k \in [\![1, C]\!]$,

$$\log \mathbb{P}(Y_i = k \mid X_i) \propto g_\theta(X_i)_k$$

  where X_i are i.i.d. and independent of θ.
- **Maximum Likelihood Estimation:** The likelihood for the data to be drawn from a given θ is

$$\mathbb{P}_\theta((X_i, Y_i)) = \prod_i \mathbb{P}(X_i)\mathbb{P}_\theta(Y_i \mid X_i) \propto \prod_i \frac{\exp(g_\theta(X_i)_{Y_i})}{\sum_k \exp(g_\theta(X_i)_k)}$$

- Maximizing the log-likelihood is equivalent to **minimizing the cross entropy**.

Generalization beyond the training samples

From train accuracy to test accuracy

Training objective

Empirical risk minimization

Let $(x_i, y_i)_{i\in[\![1,n]\!]}$ be a collection of n observations drawn independently according to $\mathcal{D}$.
Then, the objective of *empirical risk minimization* (ERM) is to find a minimizer $\hat{\theta}_n \in \mathbb{R}^p$ of

$$\min_{\theta\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} \ell(g_\theta(x_i), y_i)$$

Training objective

Empirical risk minimization

Let $(x_i, y_i)_{i \in [\![1,n]\!]}$ be a collection of n observations drawn independently according to $\mathcal{D}$. Then, the objective of *empirical risk minimization* (ERM) is to find a minimizer $\hat{\theta}_n \in \mathbb{R}^p$ of

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell(g_\theta(x_i), y_i)$$

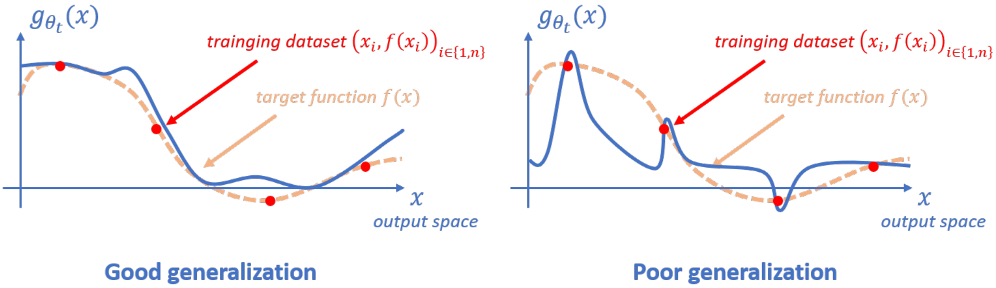
Test error

The *test error* of the ERM model is

$$\mathbb{E}\Big(\ell(g_{\hat{\theta}_n}(X), Y)\Big)$$

It is in general larger than the train error!

Beyond the training samples



- **Left model:** More regular, worst on the training set, better on the whole space.
- **Right model:** Less regular, better on the training set, worst on the whole space.
- How does the model behaves when the **test samples are different from the training samples**?

Beyond the training samples

Training objective and risk minimization

- Let $g_\theta : \mathcal{X} \to \mathcal{Y}$ be a model and $\mathcal{D}$ be a distribution of data points in $\mathcal{X} \times \mathcal{Y}$.

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}_\mathcal{D}(\theta) \triangleq \mathbb{E}_{(X,Y)\sim\mathcal{D}}(\ell(g_\theta(X), Y))$$

- During training we minimize $\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta)$ where $\widehat{\mathcal{D}}_n = \frac{1}{n}\sum_i \delta_{(x_i,y_i)}$ is the empirical distribution over the training dataset $(x_i, y_i)_{i\in[\![1,n]\!]}$.

Beyond the training samples

Training objective and risk minimization

- Let $g_\theta : \mathcal{X} \to \mathcal{Y}$ be a model and $\mathcal{D}$ be a distribution of data points in $\mathcal{X} \times \mathcal{Y}$.

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}_{\mathcal{D}}(\theta) \triangleq \mathbb{E}_{(X,Y)\sim\mathcal{D}}(\ell(g_\theta(X), Y))$$

- During training we minimize $\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta)$ where $\widehat{\mathcal{D}}_n = \frac{1}{n}\sum_i \delta_{(x_i,y_i)}$ is the empirical distribution over the training dataset $(x_i, y_i)_{i\in[\![1,n]\!]}$.

Other objectives

- Usually, our training set is not the final target: our objective is to provide a good model on another distribution $\mathcal{D}_{\text{test}}$.
- Multiple sub-problems, depending on the test distribution:
  - Generalization, out-of-distribution samples,
  - Robustness, interpolation, adversarial attacks, ...

Generalization

Setup

- Training samples are drawn iid according to the target distribution $(x_i, y_i) \sim \mathcal{D} = \mathcal{D}_{\text{test}}$.
- Let $\widehat{\theta}_n = \operatorname{argmin} \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta)$ be the parameter minimizing the training loss.
- Assume that the model is sufficiently expressive and $\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_n) = 0$. Is $\mathcal{L}_{\mathcal{D}}(\widehat{\theta}_n)$ small?

Generalization

Setup

- Training samples are drawn iid according to the target distribution $(x_i, y_i) \sim \mathcal{D} = \mathcal{D}_{\text{test}}$.
- Let $\widehat{\theta}_n = \operatorname{argmin} \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta)$ be the parameter minimizing the training loss.
- Assume that the model is sufficiently expressive and $\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_n) = 0$. Is $\mathcal{L}_{\mathcal{D}}(\widehat{\theta}_n)$ small?

Statistical error

- If $\theta \in \mathbb{R}^d$ is independent of the training samples, then, with probability $1 - \delta$,

$$\left| \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta) - \mathcal{L}_{\mathcal{D}}(\theta) \right| \leqslant \|\ell\|_\infty \sqrt{\frac{2 \ln (2/\delta)}{n}}$$

- Unfortunately, $\widehat{\theta}_n$ depends on the $\widehat{\mathcal{D}}_n$...

Decomposition of the error

Decomposition of the error

- Let $\widehat{\theta}_{n,t}$ be the parameters after t training steps and $\theta^* \in \operatorname{argmin}_\theta \mathcal{L}_\mathcal{D}(\theta)$. Then,

$$\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) = \underbrace{\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t})}_{\text{Generalization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*)}_{\text{Optimization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*) - \mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Statistical error}} + \underbrace{\mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Approx.}}$$

Decomposition of the error

Decomposition of the error

- Let $\widehat{\theta}_{n,t}$ be the parameters after t training steps and $\theta^* \in \operatorname{argmin}_\theta \mathcal{L}_\mathcal{D}(\theta)$. Then,

$$\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) = \underbrace{\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t})}_{\text{Generalization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*)}_{\text{Optimization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*) - \mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Statistical error}} + \underbrace{\mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Approx.}}$$

- **Approximation error:** by the universality of MLPs, is arbitrarily small. $d \searrow$

Decomposition of the error

Decomposition of the error

- Let $\widehat{\theta}_{n,t}$ be the parameters after t training steps and $\theta^* \in \operatorname{argmin}_\theta \mathcal{L}_\mathcal{D}(\theta)$. Then,

$$\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) = \underbrace{\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t})}_{\text{Generalization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*)}_{\text{Optimization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*) - \mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Statistical error}} + \underbrace{\mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Approx.}}$$

- **Approximation error:** by the universality of MLPs, is arbitrarily small. $d \searrow$
- **Statistical error:** Convergence in $O\left(\frac{1}{\sqrt{n}}\right)$ by Chebyshev's inequality. $n \searrow$

Decomposition of the error

Decomposition of the error

- Let $\widehat{\theta}_{n,t}$ be the parameters after t training steps and $\theta^* \in \text{argmin}_\theta \mathcal{L}_\mathcal{D}(\theta)$. Then,

$$\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) = \underbrace{\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t})}_{\text{Generalization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*)}_{\text{Optimization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*) - \mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Statistical error}} + \underbrace{\mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Approx.}}$$

- **Approximation error:** by the universality of MLPs, is arbitrarily small. $d \searrow$
- **Statistical error:** Convergence in $O\left(\frac{1}{\sqrt{n}}\right)$ by Chebyshev's inequality. $n \searrow$
- **Optimization error:** Convergence for SGD if function is sufficiently regular. $t \searrow$

Decomposition of the error

Decomposition of the error

- Let $\widehat{\theta}_{n,t}$ be the parameters after t training steps and $\theta^* \in \operatorname{argmin}_\theta \mathcal{L}_\mathcal{D}(\theta)$. Then,

$$\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) = \underbrace{\mathcal{L}_\mathcal{D}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t})}_{\text{Generalization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\widehat{\theta}_{n,t}) - \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*)}_{\text{Optimization error}} + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta^*) - \mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Statistical error}} + \underbrace{\mathcal{L}_\mathcal{D}(\theta^*)}_{\text{Approx.}}$$

- **Approximation error:** by the universality of MLPs, is arbitrarily small. $d \searrow$
- **Statistical error:** Convergence in $O\left(\frac{1}{\sqrt{n}}\right)$ by Chebyshev's inequality. $n \searrow$
- **Optimization error:** Convergence for SGD if function is sufficiently regular. $t \searrow$
- **Generalization error:** Difficult part. Depends on the model and opt. $d \nearrow, t \nearrow, n \searrow$

Overfitting in ML

Usual analysis

- Optimization error decreases
- Generalization error increases
- There is a trade-off

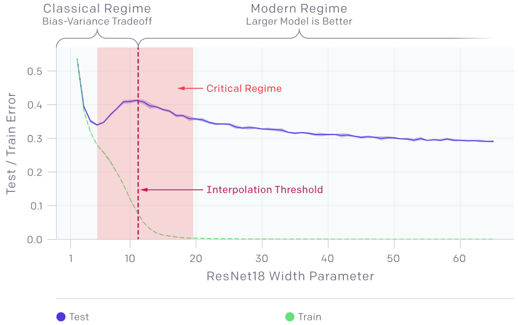
Usual mitigation strategies

- Early stopping
- Hyper-parameter selection via cross-validation
- Regularization: $\min_\theta \mathcal{L}_{\widehat{\mathcal{D}}_n}(\theta) + g(\theta)$ (usually $g(\theta) = \gamma \|\theta\|_2^2$).

But...Double descent!

Overfitting mitigated by over-parameterization

- After a certain model size, test error starts decreasing again.
- Over-parameterizing tends to create **implicit regularization**.

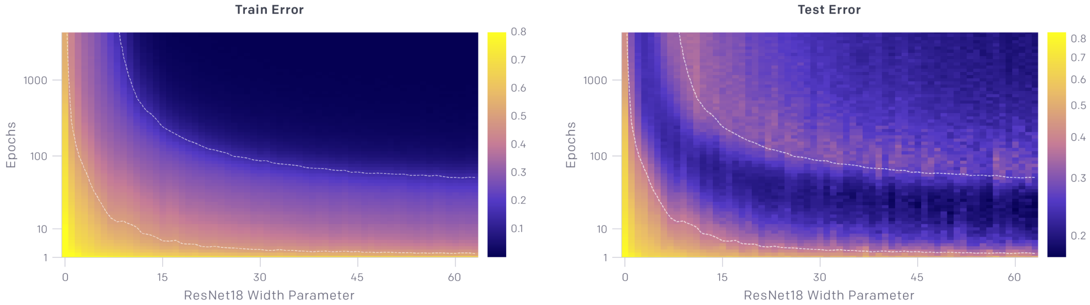


source: `https://openai.com/blog/deep-double-descent/`

But...Double descent!

Overfitting mitigated by over-parameterization

- After a certain model size, test error starts decreasing again.
- Over-parameterizing tends to create **implicit regularization**.

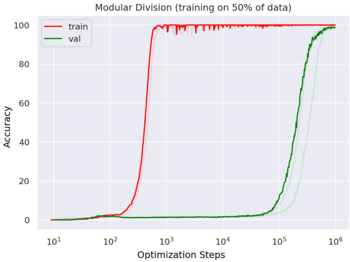


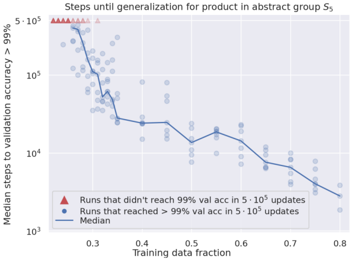
source: `https://openai.com/blog/deep-double-descent/`

But...Grokking!?

Generalization beyond overfitting

- All hope is lost... until you forget to turn your computer off during the holidays.
- Very (very) large plateaux during training.
- Still not a satisfactory explanation (don't do this at home. ;-)).





★	a	b	c	d	e
a	a	d	?	c	d
b	c	d	d	a	c
c	?	e	d	b	d
d	a	?	?	b	c
e	b	b	c	?	a

source: Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, Power et.al., 2022.

Class overview

1. **Introduction and general overview** 03/01
2. Non-convex optimization 10/01
3. Structure of ReLU networks and group invariances 17/01
4. Approximation guarantees 24/01
5. Stability and robustness 31/01
6. Infinite width limit of NNs 07/02
7. Generative models 14/02
8. Exam 21/02