Deep Neural Networks for PDEs

Philipp Grohs

universität wien
Faculty of Mathematics

DL and Vis, September 2018

Short Reading List

1. Ian Goodfellow and Yoshua Bengio and Aaron Courville: Deep Learning; MIT Press, 2016
2. Aurelien Geron: Hands-On Machine Learning with Scikit-Learn and TensorFlow; O'Reilley, 2017
3. Brian Steele and John Chandler and Swarna Reddy: Algorithms for Data Science; Springer, 2017
4. Alan Jeffrey: Applied Partial Differential Equations – An Introduction; Academic Press, 2002

Syllabus

1. PDEs and the Curse of Dimensionality
2. A Crash Course in Statistical Learning Theory (including a Detour to Variational Autoencoders)
3. PDEs as Learning Problem
4. Solving linear Kolmogorov Equations by means of Neural Network Based Learning

PDEs and the Curse of Dimensionality

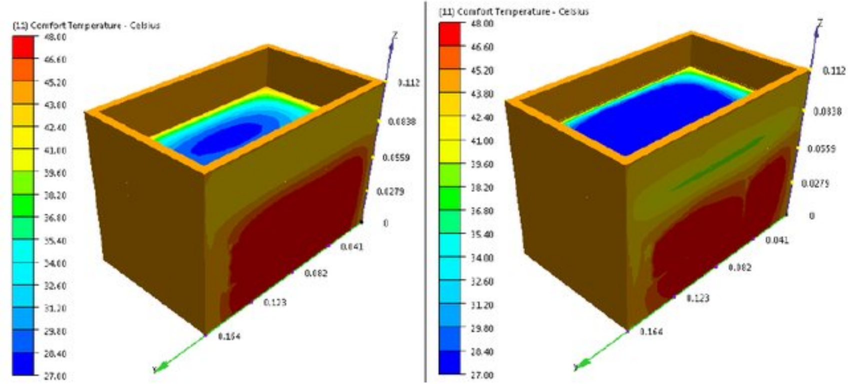
PDEs

A PDE for the function $u(x_1, \dots, x_d)$ is an equation of the form

$$\mathcal{F}\left(x_1, \dots, x_d, u, \frac{\partial u}{\partial x_1}, \dots \frac{\partial u}{\partial x_d}, \frac{\partial^2 u}{\partial x_1 \partial x_1}, \dots, \frac{\partial^2 u}{\partial x_1 \partial x_d}, \dots\right) = 0.$$

together with suitable boundary conditions.

Heat Equation



$$\frac{\partial u}{\partial t}(t,x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3} + g(t,x), \quad u(0,x) = \varphi(x)$$

$t \in (0,\infty), x \in \mathbb{R}^3;\ d = 4.$

Explicit Solution of Heat Equation if $g = 0$

Let $u(t, x)$ satisfy

$$\frac{\partial u}{\partial t}(t, x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3}, \quad u(0, x) = \varphi(x)$$

$t \in (0, \infty), x \in \mathbb{R}^3$; $d = 4$.

Explicit Solution of Heat Equation if $g = 0$

Let $u(t,x)$ satisfy

$$\frac{\partial u}{\partial t}(t,x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3}, \quad u(0,x) = \varphi(x)$$

$t \in (0,\infty), x \in \mathbb{R}^3$; $d = 4$.
Then

$$u(t,x) = \frac{1}{(4\pi t)^{3/2}} \int_{\mathbb{R}^3} \varphi(y) \exp(-|x-y|^2/4t) dy.$$

Fluid Dynamics

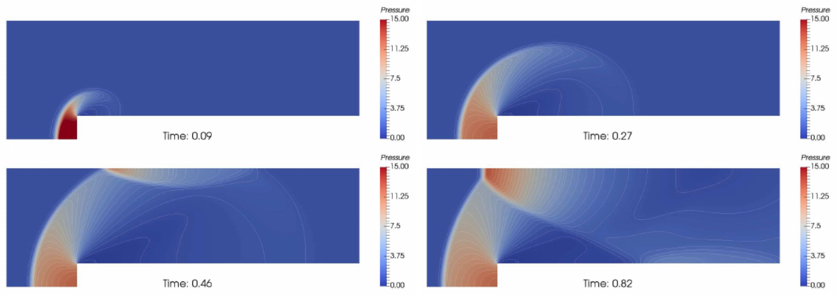


Figure 8: Mach 3 wind tunnel: Polynomial degree $K = 40$, $35k$ Vertices, Maxwellian molecules, $28.9M$ total DoFs. Coloring: pressure, contour lines: density. Computations were carried out on the Euler cluster of ETH Zurich (Xeon E5-2697 v2) with 360 cores.

$$\frac{\partial u}{\partial t}(t,x,v) + v \cdot \nabla u(t,x,v) = Qu(t,x,v)$$

$t \in (0,\infty), x, v \in \mathbb{R}^3$; $d = 7$.

Schrödinger Equation

Wave function of non-relativistic quantum mechanical system of N electons in a field of K nuclei of charge Z_ν and fixed position $R_\mu \in \mathbb{R}^3$

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N;t) = -\frac{1}{2}\sum_{\xi=1}^{N}\Delta_i\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N;t) -$$

$$\sum_{\xi=1}^{N}\sum_{\nu=1}^{K}\frac{Z_\nu}{|\mathbf{r}_\xi - R_\nu|}\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N;t) + \frac{1}{2}\sum_{\xi=1}^{N}\sum_{\eta=1}^{N}\frac{1-\delta_{\xi,\eta}}{|\mathbf{r}_\xi - \mathbf{r}_\eta|},$$

$t \in (0,\infty)$, $\mathbf{r}_1,\ldots,\mathbf{r}_N \in \mathbb{R}^3$; $d = 3N+1$.

Black-Scholes Equation

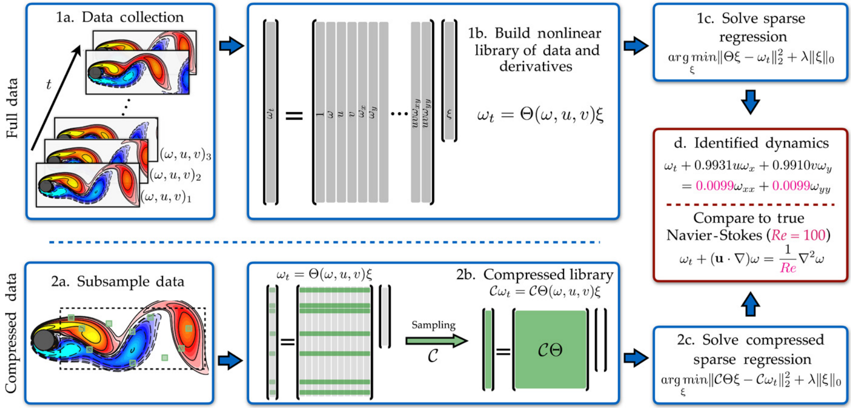
Pricing a portfolio of N financial derivatives

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{N} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^N} \left(\frac{\partial^2 u}{\partial x_i \partial x_j}\right)(t,x) + \sum_{i=1}^{N} \mu_i x_i \left(\frac{\partial u}{\partial x_i}\right)(t,x)$$

$$u(0,x) = \max\{K - \sum_{i=1}^{N} c_i x_i, 0\}$$

$t \in (0,\infty)$, $x \in \mathbb{R}^N$; $d = N+1$.

Learning the PDE [Rudy et.al. (2017)]



Finite Difference Approach

Want to approximate $u(x)$ for $x \in [0,1]^d$.

Finite Difference Approach

Want to approximate $u(x)$ for $x \in [0,1]^d$.

- Let

$$u_{i_1,\ldots,i_d} \sim u(i_1\epsilon,\ldots,i_d\epsilon), \quad (i_1,\ldots,i_d) \in \{0,\ldots,\lfloor \epsilon^{-1} \rfloor\}^d,$$

Finite Difference Approach

Want to approximate $u(x)$ for $x \in [0,1]^d$.

- Let

$$u_{i_1,\dots,i_d} \sim u(i_1\epsilon,\dots,i_d\epsilon), \quad (i_1,\dots,i_d) \in \{0,\dots,\lfloor \epsilon^{-1} \rfloor\}^d,$$

-

$$\frac{u_{i_1,\dots,i_l+1,\dots,i_d} - u_{i_1,\dots,i_l,\dots,i_d}}{\epsilon} \sim \frac{\partial}{\partial x_l} u(i_1\epsilon,\dots,i_d\epsilon),$$
$$(i_1,\dots,i_d) \in \{0,\dots,\lfloor \epsilon^{-1} \rfloor\}^d,$$

  and so on,

Finite Difference Approach

Want to approximate $u(x)$ for $x \in [0,1]^d$.

- Let

$$u_{i_1,\ldots,i_d} \sim u(i_1\epsilon, \ldots, i_d\epsilon), \quad (i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d,$$

-

$$\frac{u_{i_1,\ldots,i_l+1,\ldots,i_d} - u_{i_1,\ldots,i_l,\ldots,i_d}}{\epsilon} \sim \frac{\partial}{\partial x_l} u(i_1\epsilon, \ldots, i_d\epsilon),$$
$$(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d,$$

and so on,

- and solve the discrete system

$$\mathcal{F}\left(i_1\epsilon, \ldots, i_d\epsilon, u_{i_1,\ldots,i_d}, \frac{u_{i_1+1,\ldots,i_d} - u_{i_1,\ldots,i_d}}{\epsilon}, \ldots\right) = 0$$
$$(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d.$$

Curse of Dimensionality

The system

$$\mathcal{F}\left(i_1\epsilon, \ldots, i_d\epsilon, u_{i_1,\ldots,i_d}, \frac{u_{i_1+1,\ldots,i_d} - u_{i_1,\ldots,i_d}}{\epsilon}, \ldots\right) = 0$$

$$(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d.$$

requires us to solve an equation in $u_{i_1,\ldots,i_d}$ for $(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d$.

Curse of Dimensionality

The system

$$\mathcal{F}\left(i_1\epsilon, \ldots, i_d\epsilon, u_{i_1,\ldots,i_d}, \frac{u_{i_1+1,\ldots,i_d} - u_{i_1,\ldots,i_d}}{\epsilon}, \ldots\right) = 0$$
$$(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d.$$

requires us to solve an equation in $u_{i_1,\ldots,i_d}$ for $(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d$.

Exponential Dependence on the Dimension

Let $\epsilon = \frac{1}{2}$ (take two samples in each coordinate). Then these are 2^d unknowns.

Curse of Dimensionality

The system

$$\mathcal{F}\left(i_1\epsilon, \ldots, i_d\epsilon, u_{i_1,\ldots,i_d}, \frac{u_{i_1+1,\ldots,i_d} - u_{i_1,\ldots,i_d}}{\epsilon}, \ldots\right) = 0$$

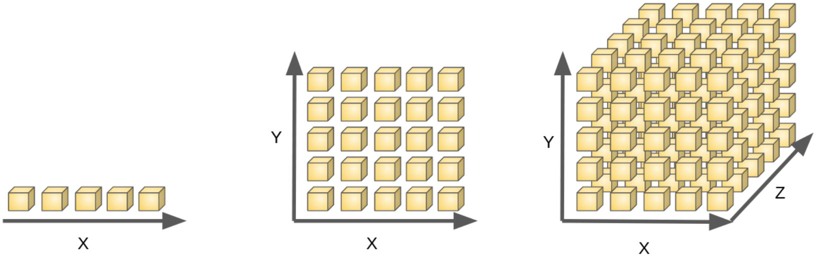
$$(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d.$$

requires us to solve an equation in $u_{i_1,\ldots,i_d}$ for $(i_1, \ldots, i_d) \in \{0, \ldots, \lfloor \epsilon^{-1} \rfloor\}^d$.

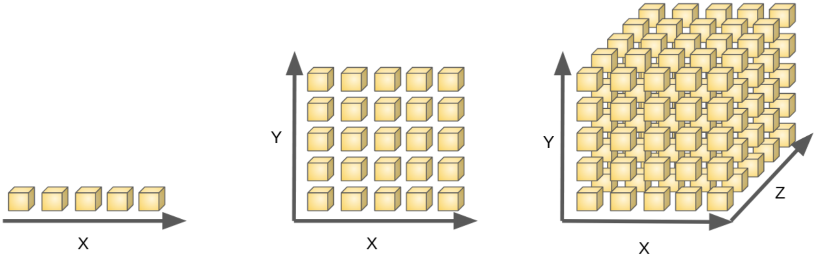
Exponential Dependence on the Dimension

Let $\epsilon = \frac{1}{2}$ (take two samples in each coordinate). Then these are 2^d unknowns. $\rightsquigarrow$ intractable for high-dimensional problems!

Curse of Dimensionality

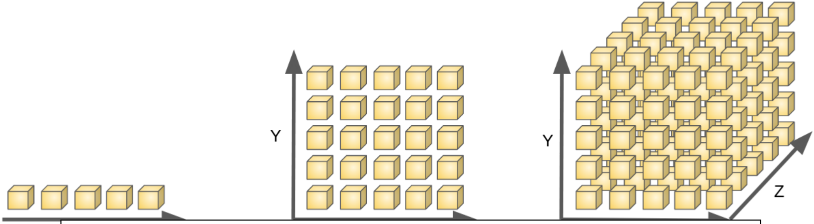


Curse of Dimensionality



The complexity of approximating a general d-dimensional function scales exponentially in d.

Curse of Dimensionality



The ... ction

Suppose we have a problem where we aim to approximate a d-dimensional function. An algorithm to solve the problem suffers from the curse of dimensionality if its computational complexity depends exponentially on the dimension d.

Black-Scholes Equation

- Pricing a portfolio of N financial derivatives

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{N} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^N} (\frac{\partial^2 u}{\partial x_i \partial x_j})(t,x) + \sum_{i=1}^{N} \mu_i x_i (\frac{\partial u}{\partial x_i})(t,x)$$

$$u(0,x) = \max\{K - \sum_{i=1}^{N} c_i x_i, 0\}$$

$t \in (0,\infty)$, $x \in \mathbb{R}^N$; $d = N+1$.

Black-Scholes Equation

- Pricing a portfolio of N financial derivatives

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{N} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^N} (\frac{\partial^2 u}{\partial x_i \partial x_j})(t,x) + \sum_{i=1}^{N} \mu_i x_i (\frac{\partial u}{\partial x_i})(t,x)$$

$$u(0,x) = \max\{K - \sum_{i=1}^{N} c_i x_i, 0\}$$

$t \in (0,\infty)$, $x \in \mathbb{R}^N$; $d = N + 1$.

- Realistic values: $d = 100 - 1000$.

Black-Scholes Equation

- Pricing a portfolio of N financial derivatives

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{N} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^N} \left(\frac{\partial^2 u}{\partial x_i \partial x_j}\right)(t,x) + \sum_{i=1}^{N} \mu_i x_i \left(\frac{\partial u}{\partial x_i}\right)(t,x)$$

$$u(0,x) = \max\{K - \sum_{i=1}^{N} c_i x_i, 0\}$$

  $t \in (0,\infty)$, $x \in \mathbb{R}^N$; $d = N + 1$.
- Realistic values: $d = 100 - 1000$.
- Complexity of finite difference method: $2^{100} - 2^{1000}$.

Black-Scholes Equation

- Pricing a portfolio of N financial derivatives

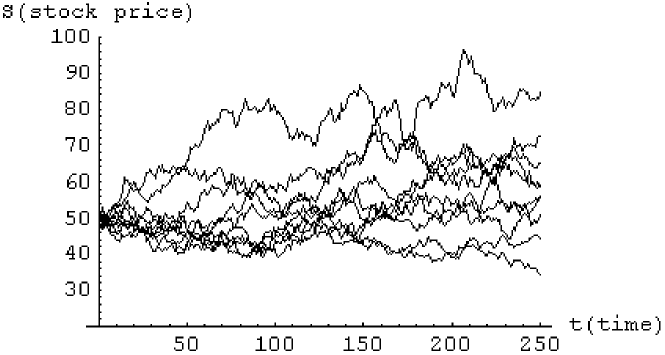
$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{N} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^N} \left(\frac{\partial^2 u}{\partial x_i \partial x_j}\right)(t,x) + \sum_{i=1}^{N} \mu_i x_i \left(\frac{\partial u}{\partial x_i}\right)(t,x)$$

$$u(0,x) = \max\{K - \sum_{i=1}^{N} c_i x_i, 0\}$$

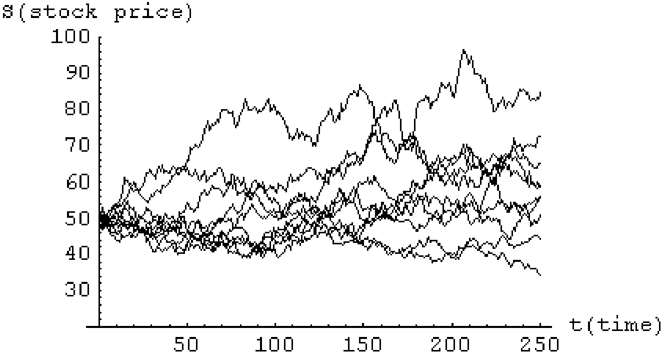
$t \in (0,\infty)$, $x \in \mathbb{R}^N$; $d = N + 1$.

- Realistic values: $d = 100 - 1000$.
- Complexity of finite difference method: $2^{100} - 2^{1000}$.
- Number of atoms in the universe: 2^{250}.

Black-Scholes Equation

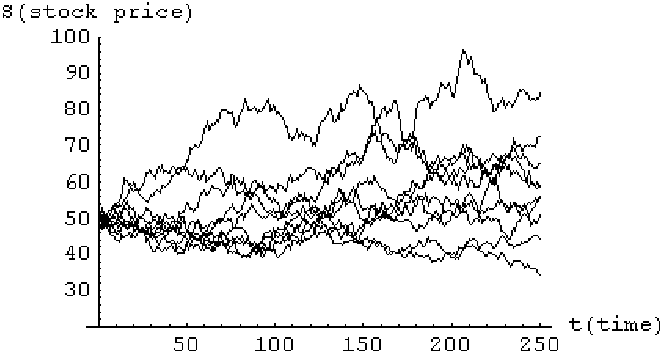


Black-Scholes Equation



- Option pricing is extremely relevant and has to be done every day in the financial industry

Black-Scholes Equation



- Option pricing is extremely relevant and has to be done every day in the financial industry
- All algorithms for the solution of the Black-Scholes equation suffer from the curse of dimensionality!

MNIST

MNIST Database for handwritten digit recognition http://yann.lecun.com/exdb/mnist/

MNIST

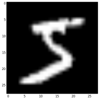
MNIST Database for hand-written digit recognition http://yann.lecun.com/exdb/mnist/

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:

MNIST

MNIST Database for handwritten digit recognition http://yann.lecun.com/exdb/mnist/

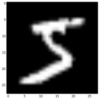
- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:



MNIST

MNIST Database for hand-written digit recognition http://yann.lecun.com/exdb/mnist/

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:

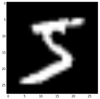


- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit

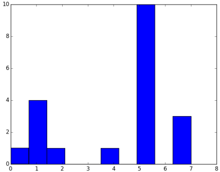
MNIST

MNIST Database for hand-written digit recognition http://yann.lecun.com/exdb/mnist/

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:

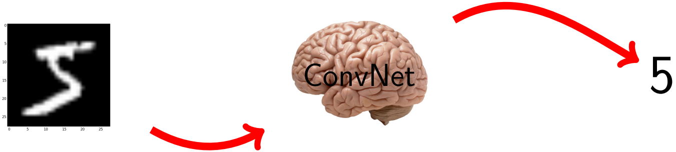


- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit

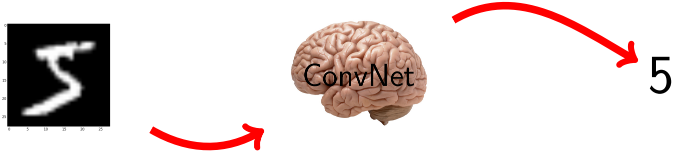


MNIST

MNIST

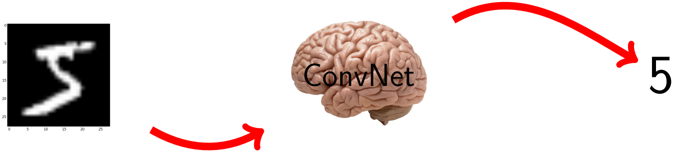


MNIST



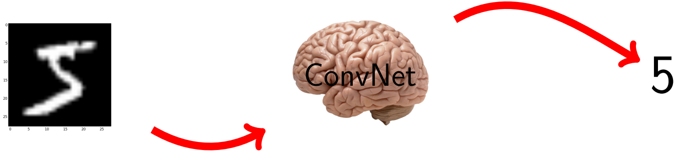
- This is a 784-dimensional function

MNIST



- This is a 784-dimensional function
- Apparently, deep learning does not suffer from the curse of dimensionality for certain classification problems!

MNIST



- This is a 784-dimensional function
- Apparently, deep learning does not suffer from the curse of dimensionality for certain classification problems!

Can this also be used for the solution of PDEs?

A Crash Course in Statistical Learning Theory

Data Generating Distribution

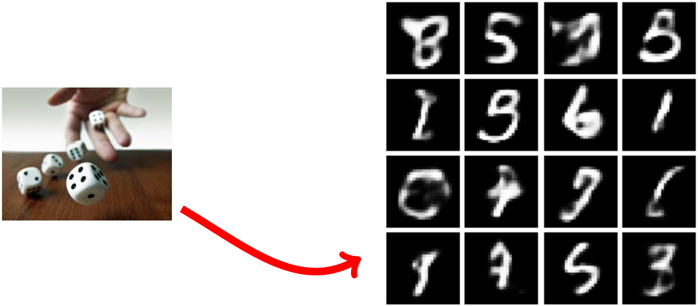
Suppose that there exists a probability distribution on $\mathbb{R}^{784}$ that randomly generates handwritten digits.

Data Generating Distribution

Suppose that there exists a probability distribution on $\mathbb{R}^{784}$ that randomly generates handwritten digits.

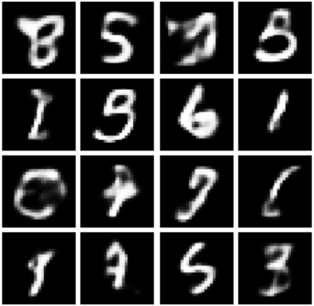
Data Generating Distribution

Suppose that there exists a probability distribution on $\mathbb{R}^{784}$ that randomly generates handwritten digits.



Data Generating Distribution

Suppose that there exists a probability distribution on $\mathbb{R}^{784}$ that randomly generates handwritten digits.



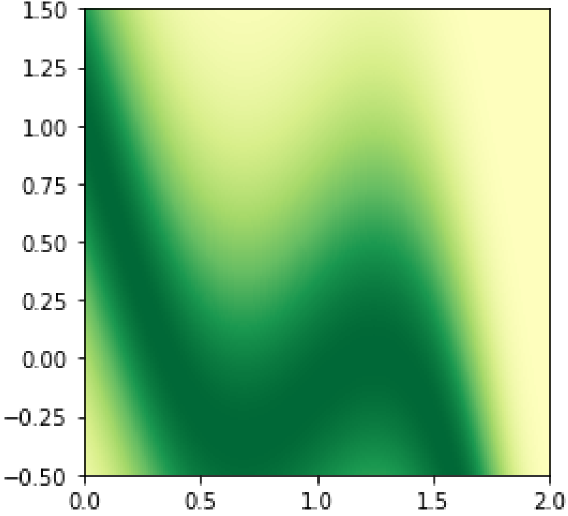
$\boxed{\rightsquigarrow \textbf{Variational Autoencoder Demo}}$

A New Look

Suppose that our training data consists of samples according to a given data distribution (X, Y)

A New Look

Suppose that our training data consists of samples according to a given data distribution (X, Y)

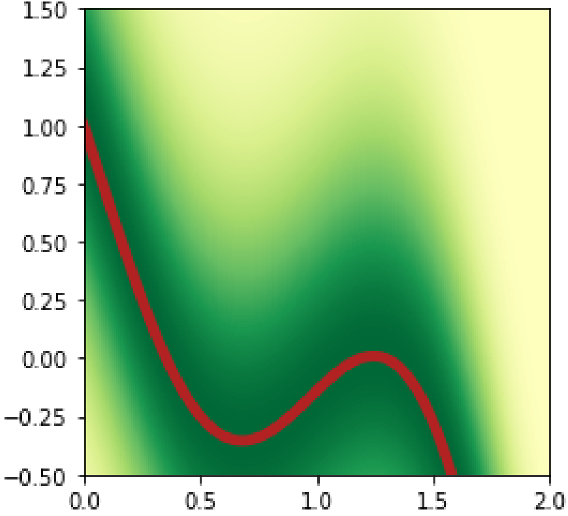


A New Look

If we knew the data distribution (X, Y), the best functional relation between X and Y would simply be $\mathbb{E}[Y|X = x]$!

A New Look

If we knew the data distribution (X, Y), the best functional relation between X and Y would simply be $\mathbb{E}[Y|X = x]$!

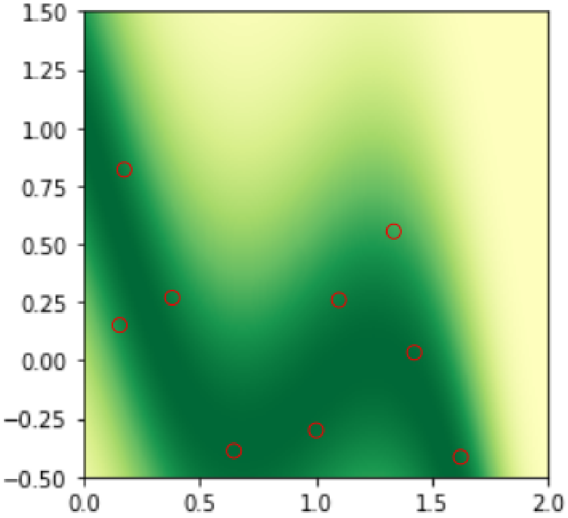


A New Look

But we only have samples and do not know the distribution (X, Y)

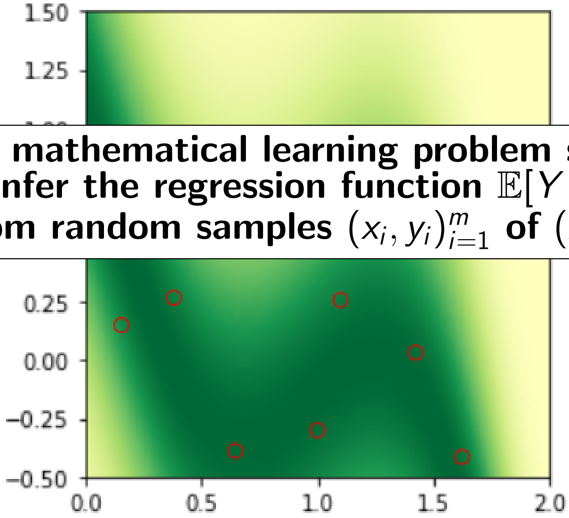
A New Look

But we only have samples and do not know the distribution (X, Y)



A New Look

But we only have samples and do not know the distribution (X, Y)



A mathematical learning problem seeks to infer the regression function $\mathbb{E}[Y|X = x]$ from random samples $(x_i, y_i)_{i=1}^m$ of (X, Y).

Mathematical Formulation

Mathematical Formulation

- Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $X : \Omega \to \mathbb{R}^d$ and $Y : \Omega \to \mathbb{R}^n$ be random vectors. Find the best functional relationship $\hat{U} : \mathbb{R}^d \to \mathbb{R}^n$ between these vectors in the sense that

$$\begin{aligned}
\hat{U} &= \underset{U:\mathbb{R}^d \to \mathbb{R}^n}{\operatorname{argmin}} \int_\Omega |U(X(\omega)) - Y(\omega)|^2 d\mathbb{P}(\omega) \\
&= \underset{U:\mathbb{R}^d \to \mathbb{R}^n}{\operatorname{argmin}} \mathbb{E}\left[|U(X) - Y|^2\right].
\end{aligned}$$

Mathematical Formulation

- Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $X : \Omega \to \mathbb{R}^d$ and $Y : \Omega \to \mathbb{R}^n$ be random vectors. Find the best functional relationship $\hat{U} : \mathbb{R}^d \to \mathbb{R}^n$ between these vectors in the sense that

$$\begin{aligned}\hat{U} &= \underset{U:\mathbb{R}^d \to \mathbb{R}^n}{\operatorname{argmin}} \int_\Omega |U(X(\omega)) - Y(\omega)|^2 d\mathbb{P}(\omega) \\ &= \underset{U:\mathbb{R}^d \to \mathbb{R}^n}{\operatorname{argmin}} \mathbb{E}\left[|U(X) - Y|^2\right].\end{aligned}$$

- We have

$$\hat{U}(x) = \mathbb{E}\left[Y|X = x\right].$$

Mathematical Formulation

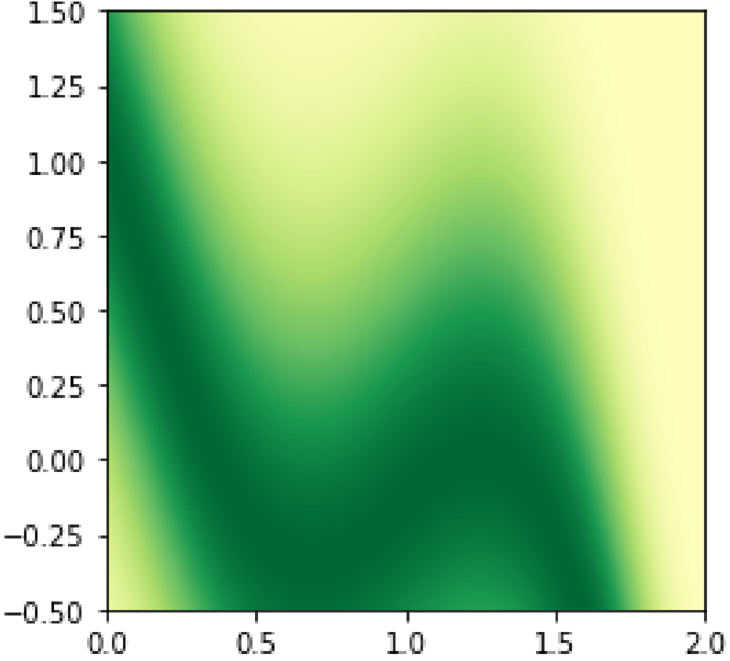
- Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $X : \Omega \to \mathbb{R}^d$ and $Y : \Omega \to \mathbb{R}^n$ be random vectors. Find the best functional relationship $\hat{U} : \mathbb{R}^d \to \mathbb{R}^n$ between these vectors in the sense that

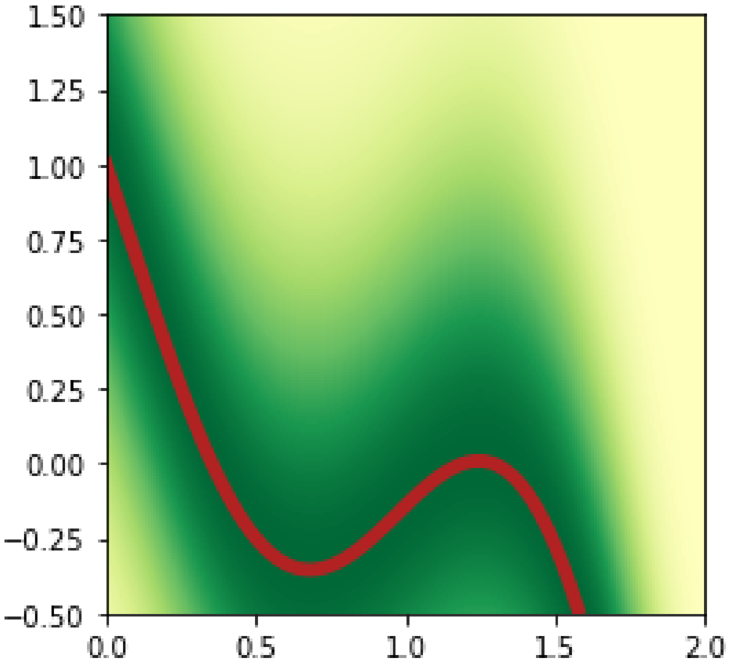
$$\begin{aligned}\hat{U} &= \underset{U:\mathbb{R}^d \to \mathbb{R}^n}{\operatorname{argmin}} \int_\Omega |U(X(\omega)) - Y(\omega)|^2 d\mathbb{P}(\omega) \\ &= \underset{U:\mathbb{R}^d \to \mathbb{R}^n}{\operatorname{argmin}} \mathbb{E}\left[|U(X) - Y|^2\right].\end{aligned}$$

- We have

$$\hat{U}(x) = \mathbb{E}\left[Y | X = x\right].$$

$\hat{U}$ is called the *regression function*.


1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0.0
0.5
1.0
1.5
2.0



1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0.0
0.5
1.0
1.5
2.0

Statistical Learning Theory

- Let $z = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$ be m realizations of samples independently drawn according to (X, Y). For a function $U\colon \mathbb{R}^d \to \mathbb{R}^k$ define the *empirical risk* of U by

$$\mathcal{E}_z(U) = \frac{1}{m} \sum_{i=1}^{m} |U(x_i) - y_i|^2.$$

Statistical Learning Theory

- Let $z = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$ be m realizations of samples independently drawn according to (X, Y). For a function $U \colon \mathbb{R}^d \to \mathbb{R}^k$ define the *empirical risk* of U by

$$\mathcal{E}_z(U) = \frac{1}{m} \sum_{i=1}^{m} |U(x_i) - y_i|^2.$$

- *Empirical Risk Minimization (ERM)* picks a hypothesis class $\mathcal{H} \subset C(\mathbb{R}^d, \mathbb{R}^k)$ and computes the *empirical regression function*

$$\hat{U}_{\mathcal{H},z} \in \operatorname*{argmin}_{U \in \mathcal{H}} \mathcal{E}_z(U).$$

Statistical Learning Theory

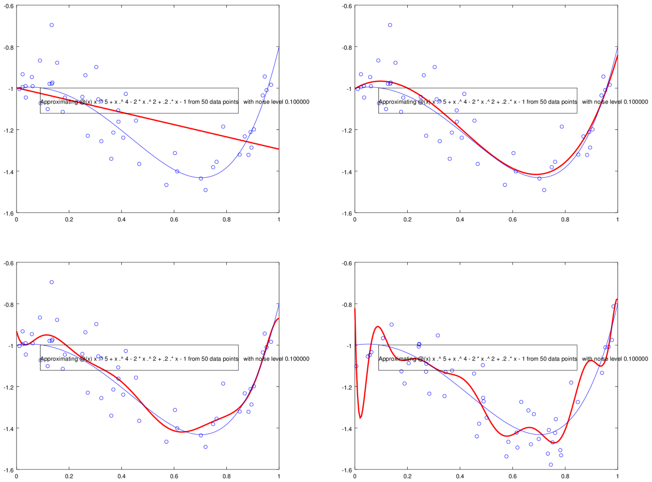
- Let $z = \big((x_1, y_1), \dots, (x_m, y_m)\big)$ be m realizations of samples independently drawn according to (X, Y). For a function $U\colon \mathbb{R}^d \to \mathbb{R}^k$ define the *empirical risk* of U by

$$\mathcal{E}_z(U) = \frac{1}{m} \sum_{i=1}^{m} |U(x_i) - y_i|^2.$$

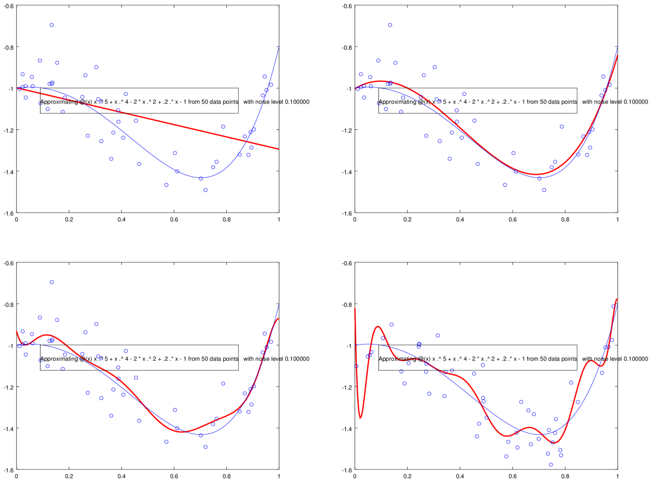
- *Empirical Risk Minimization (ERM)* picks a hypothesis class $\mathcal{H} \subset C(\mathbb{R}^d, \mathbb{R}^k)$ and computes the *empirical regression function*

$$\hat{U}_{\mathcal{H},z} \in \operatorname*{argmin}_{U \in \mathcal{H}} \mathcal{E}_z(U).$$

- Example $\mathcal{H} = \{\text{Polynomials of degree} \leq p\}$.



Approximating @(x) x .^ 5 + x .^ 4 - 2 * x .^ 2 + .2 .* x - 1 from 50 data points with noise level 0.100000
-0.6
-0.8
-1
-1.2
-1.4
-1.6
0
0.2
0.4
0.6
0.8
1



Degree too low: underfitting. Degree to high: overfitting!

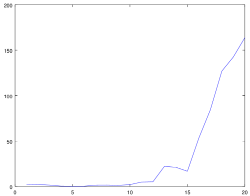


Figure: Error with Polynomial Degree

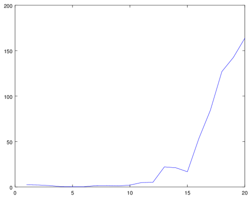


Figure: Error with Polynomial Degree

Bias-Variance-Problem

"Capacity" of the hypothesis space has to be adapted to the complexity of the target function and the sample size!

Bias-Variance Decomposition

Let (X, Y) data generating r.v.'s and $\hat{U}$ the regression function. Let $\mathbf{z} = (x_i, y_i)_{i=1}^m$ i.i.d. samples, $\mathcal{H}$ a hypothesis class and $\hat{U}_{\mathcal{H},\mathbf{z}}$ the empirical regression function. We seek to understand the error

$$\epsilon := \mathcal{E}(\hat{U}_{\mathcal{H},\mathbf{z}}) - \mathcal{E}(\hat{U}) = \mathbb{E}|\hat{U}_{\mathcal{H},\mathbf{z}}(X) - \hat{U}(X)|^2$$

Bias-Variance Decomposition

Let (X, Y) data generating r.v.'s and $\hat{U}$ the regression function. Let $\mathbf{z} = (x_i, y_i)_{i=1}^m$ i.i.d. samples, $\mathcal{H}$ a hypothesis class and $\hat{U}_{\mathcal{H},\mathbf{z}}$ the empirical regression function. We seek to understand the error

$$\epsilon := \mathcal{E}(\hat{U}_{\mathcal{H},\mathbf{z}}) - \mathcal{E}(\hat{U}) = \mathbb{E}|\hat{U}_{\mathcal{H},\mathbf{z}}(X) - \hat{U}(X)|^2$$

Bias-Variance Decomposition

Let $U_{\mathcal{H}} := \operatorname{argmin}_{U\in\mathcal{H}} \mathbb{E}|U(X) - \hat{U}(X)|^2$,
$\epsilon_{approx} := \mathbb{E}|U_{\mathcal{H}}(X) - \hat{U}(X)|^2$ the *approximation error* and
$\epsilon_{generalize} := \mathcal{E}(U_{\mathcal{H},\mathbf{z}}) - \mathcal{E}(U_{\mathcal{H}})$ the *generalization error*. Then
$\epsilon = \epsilon_{approx} + \epsilon_{generalize}$.

Bias-Variance Decomposition

Let (X, Y) data generating r.v.'s and $\hat{U}$ the regression function. Let $\mathbf{z} = (x_i, y_i)_{i=1}^m$ i.i.d. samples, $\mathcal{H}$ a hypothesis class and $\hat{U}_{\mathcal{H},\mathbf{z}}$ the empirical regression function. We seek to understand the error

$$\epsilon := \mathcal{E}(\hat{U}_{\mathcal{H},\mathbf{z}}) - \mathcal{E}(\hat{U}) = \mathbb{E}|\hat{U}_{\mathcal{H},\mathbf{z}}(X) - \hat{U}(X)|^2$$

Bias-Variance Decomposition

Let $U_{\mathcal{H}} := \operatorname{argmin}_{U \in \mathcal{H}} \mathbb{E}|U(X) - \hat{U}(X)|^2$,
$\epsilon_{approx} := \mathbb{E}|U_{\mathcal{H}}(X) - \hat{U}(X)|^2$ the *approximation error* and
$\epsilon_{generalize} := \mathcal{E}(U_{\mathcal{H},\mathbf{z}}) - \mathcal{E}(U_{\mathcal{H}})$ the *generalization error*. Then
$\epsilon = \epsilon_{approx} + \epsilon_{generalize}$.

Main Theorem [e.g., Cucker-Zhou (2007)]

If $m \gtrsim \frac{\ln(\mathcal{N}(\mathcal{H}, c\cdot\eta))}{\eta^2}$ (and very strong conditions hold), then $\epsilon_{generalize} \leq \eta$ w.h.p. where $\mathcal{N}(\mathcal{H}, s)$ is the s-covering number of $\mathcal{H}$ w.r.t. L^∞ .

Bias-Variance Decomposition

Let (X, Y) data generating r.v.'s and $\hat{U}$ the regression function. Let $\mathbf{z} = (x_i, y_i)_{i=1}^m$ i.i.d. samples, $\mathcal{H}$ a hypothesis class and $\hat{U}_{\mathcal{H},\mathbf{z}}$ the empirical regression function. We seek to understand the error

$$\epsilon := \mathcal{E}(\hat{U}_{\mathcal{H},\mathbf{z}}) - \mathcal{E}(\hat{U}) = \mathbb{E}|\hat{U}_{\mathcal{H},\mathbf{z}}(X) - \hat{U}(X)|^2$$

Bias-Variance Decomposition

Let $U_{\mathcal{H}} := \operatorname{argmin}_{U \in \mathcal{H}} \mathbb{E}|U(X) - \hat{U}(X)|^2$,
$\epsilon_{approx} := \mathbb{E}|U_{\mathcal{H}}(X) - \hat{U}(X)|^2$ the *approximation error* and

Problems for Data Science Applications:

- **Assumption that data is iid is debatable**
- **Different asymptotic regime in deep learning (where often $\#$DOFs $>>$ $\#$training samples)**
- **Without knowing $\mathbb{P}_{(X,Y)}$ it is impossible to control the approximation error.**

w.r.t. L^∞ .

PDEs as Learning Problems

Explicit Solution of Heat Equation if $g = 0$

Let $u(t, x)$ satisfy

$$\frac{\partial u}{\partial t}(t, x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3}, \quad u(0, x) = \varphi(x)$$

$t \in (0, \infty), x \in \mathbb{R}^3$; $d = 4$.

Explicit Solution of Heat Equation if $g = 0$

Let $u(t,x)$ satisfy

$$\frac{\partial u}{\partial t}(t,x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3}, \quad u(0,x) = \varphi(x)$$

$t \in (0,\infty), x \in \mathbb{R}^3$; $d = 4$.
Then

$$u(t,x) = \int_{\mathbb{R}^3} \varphi(y) \frac{1}{(4\pi t)^{3/2}} \exp(-|x-y|^2/4t) dy.$$

Explicit Solution of Heat Equation if $g = 0$

Let $u(t, x)$ satisfy

$$\frac{\partial u}{\partial t}(t, x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3}, \quad u(0, x) = \varphi(x)$$

$t \in (0, \infty), x \in \mathbb{R}^3$; $d = 4$.
Then

$$u(t, x) = \int_{\mathbb{R}^3} \varphi(y) \frac{1}{(4\pi t)^{3/2}} \exp(-|x - y|^2/4t) dy.$$

In other words

$$u(t, x) = \mathbb{E}\left[\varphi(Z_t^x)\right], \quad Z_t^x \sim \mathcal{N}(x, t^{1/2} I).$$

Explicit Solution of Heat Equation if $g = 0$

Let $u(t, x)$ satisfy

$$\frac{\partial u}{\partial t}(t, x) = \frac{\partial^2 u}{\partial x_1 \partial x_1} + \frac{\partial^2 u}{\partial x_2 \partial x_2} + \frac{\partial^2 u}{\partial x_3 \partial x_3}, \quad u(0, x) = \varphi(x)$$

$t \in (0, \infty), x \in \mathbb{R}^3$; $d = 4$.
Then

$$u(t, x) = \int_{\mathbb{R}^3} \varphi(y) \frac{1}{(4\pi t)^{3/2}} \exp(-|x - y|^2 / 4t) dy.$$

In other words

$$u(t, x) = \mathbb{E}\left[\varphi(Z_t^x)\right], \quad Z_t^x \sim \mathcal{N}(x, t^{1/2} I).$$

In other words, for $x \in [u, v]^3$ and $X \sim \mathcal{U}[u, v]^3$ and $Y = \varphi\left(Z_t^X\right)$ we have

$$u(t, x) = \mathbb{E}\left[Y | X = x\right].$$

The solution $u(t,x)$ of the PDE can be interpreted as solution to the learning problem with data distribution (X,Y), where $X \sim \mathcal{U}[u,v]^3$ and $Y = \varphi(Z_t^X)$ and $Z_t^X \sim \mathcal{N}(x, t^{1/2} I)$!

$t \in (0,\infty), x \in \mathbb{R}^3$; $d = 4$.

Then

$$u(t,x) = \int_{\mathbb{R}^3} \varphi(y) \frac{1}{(4\pi t)^{3/2}} \exp(-|x-y|^2/4t) dy.$$

In other words

$$u(t,x) = \mathbb{E}\left[\varphi(Z_t^x)\right], \quad Z_t^x \sim \mathcal{N}(x, t^{1/2} I).$$

In other words, for $x \in [u,v]^3$ and $X \sim \mathcal{U}[u,v]^3$ and $Y = \varphi\left(Z_t^X\right)$ we have

$$u(t,x) = \mathbb{E}\left[Y | X = x\right].$$

Ex

The solution $u(t,x)$ of the PDE can be interpreted as solution to the learning problem with data distribution (X,Y), where $X \sim \mathcal{U}[u,v]^3$ and $Y = \varphi(Z_t^X)$ and $Z_t^X \sim \mathcal{N}(x, t^{1/2} I)$!

Contrary to conventional ML problems, the data distribution is now explicitly known – we can simulate as much training data as we want!

In other words, for $x \in [u,v]^3$ and $X \sim \mathcal{U}[u,v]^3$ and $Y = \varphi\left(Z_t^X\right)$ we have

$$u(t,x) = \mathbb{E}\left[Y|X=x\right].$$

Ex

The solution $u(t,x)$ of the PDE can be interpreted as solution to the learning problem with data distribution (X,Y), where $X \sim \mathcal{U}[u,v]^3$ and $Y = \varphi(Z_t^X)$ and $Z_t^X \sim \mathcal{N}(x, t^{1/2}I)$!

Contrary to conventional ML problems, the data distribution is now explicitly known – we can simulate as much training data as we want!

We will see in a minute that similar properties hold for a much more general class of PDEs!

Linear Kolmogorov Equations

Given $\Sigma : \mathbb{R}^d \to \mathbb{R}^{d \times d}$, $\mu : \mathbb{R}^d \to \mathbb{R}^d$ and initial value $\varphi : \mathbb{R}^d \to \mathbb{R}$, find $u : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ with

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\left(\Sigma(x)\Sigma^T(x)\mathrm{Hess}_x u(t,x)\right) + \mu(x) \cdot \nabla_x u(t,x),$$
$$(t,x) \in [0,T] \times \mathbb{R}^d, \quad u(0,x) = \varphi(x).$$

Linear Kolmogorov Equations

Given $\Sigma : \mathbb{R}^d \to \mathbb{R}^{d \times d}$, $\mu : \mathbb{R}^d \to \mathbb{R}^d$ and initial value $\varphi : \mathbb{R}^d \to \mathbb{R}$, find $u : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ with

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\left(\Sigma(x)\Sigma^T(x)\mathrm{Hess}_x u(t,x)\right) + \mu(x) \cdot \nabla_x u(t,x),$$
$$(t,x) \in [0, T] \times \mathbb{R}^d, \quad u(0,x) = \varphi(x).$$

- Examples include convection-diffusion equations and Black-Scholes Equation.

Linear Kolmogorov Equations

Given $\Sigma : \mathbb{R}^d \to \mathbb{R}^{d \times d}$, $\mu : \mathbb{R}^d \to \mathbb{R}^d$ and initial value $\varphi : \mathbb{R}^d \to \mathbb{R}$, find $u : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ with

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\left(\Sigma(x)\Sigma^T(x)\mathrm{Hess}_x u(t,x)\right) + \mu(x) \cdot \nabla_x u(t,x),$$
$$(t,x) \in [0,T] \times \mathbb{R}^d, \quad u(0,x) = \varphi(x).$$

- Examples include convection-diffusion equations and Black-Scholes Equation.
- Standard methods such as sparse grid methods, sparse tensor product methods, spectral methods, finite element methods or finite difference methods are incapable of solving such equations in high dimensions ($d = 100$)!

Special Case: Pricing of Financial Derivatives

Special Case: Pricing of Financial Derivatives

- Given a portfolio consisting of d assets with value $(x_i(t))_{i=1}^d$.

Special Case: Pricing of Financial Derivatives

- Given a portfolio consisting of d assets with value $(x_i(t))_{i=1}^d$.
- *European Max Option:* At time T, exercise option and receive

$$G(x) := \max\left\{\max_{i=1}^{d}(x_i - K_i), 0\right\}$$

Special Case: Pricing of Financial Derivatives

- Given a portfolio consisting of d assets with value $(x_i(t))_{i=1}^d$.
- *European Max Option:* At time T, exercise option and receive

$$G(x) := \max\left\{\max_{i=1}^{d}(x_i - K_i), 0\right\}$$

- (Black-Scholes (1973)): in the absence of correlations the portfolio-value $u(t,x)$ satisfies

$$\left(\frac{\partial}{\partial t}\right)u(t,x) + \frac{\mu}{2}\sum_{i=1}^{d} x_i \left(\frac{\partial}{\partial x_i}u(t,x)\right) + \frac{\sigma^2}{2}\sum_{i=1}^{d}|x_i|^2\left(\frac{\partial^2}{\partial x_i^2}u(t,x)\right) = 0,$$

$$u(T,x) = G(x).$$

Special Case: Pricing of Financial Derivatives

- Given a portfolio consisting of d assets with value $(x_i(t))_{i=1}^d$.
- *European Max Option:* At time T, exercise option and receive

$$G(x) := \max\left\{\max_{i=1}^{d}(x_i - K_i), 0\right\}$$

- (Black-Scholes (1973)): in the absence of correlations the portfolio-value $u(t,x)$ satisfies

$$\left(\frac{\partial}{\partial t}\right)u(t,x) + \frac{\mu}{2}\sum_{i=1}^{d} x_i\left(\frac{\partial}{\partial x_i}u(t,x)\right) + \frac{\sigma^2}{2}\sum_{i=1}^{d}|x_i|^2\left(\frac{\partial^2}{\partial x_i^2}u(t,x)\right) = 0,$$

$$u(T,x) = G(x).$$

- **Pricing Problem:** $u(0,x) = ??$.

Kolmogorov PDEs as Learning Problems

Kolmogorov PDEs as Learning Problems

For $x \in \mathbb{R}^d$ and $t \in \mathbb{R}_+$ let

$$Z_t^x := x + \int_0^t \mu(Z_s^x) ds + \int_0^t \Sigma(Z_s^x) dW_s.$$

Then (Feynman-Kac)

$$u(T, x) = \mathbb{E}(\varphi(Z_T^x)).$$

Kolmogorov PDEs as Learning Problems

For $x \in \mathbb{R}^d$ and $t \in \mathbb{R}_+$ let

$$Z_t^x := x + \int_0^t \mu(Z_s^x)ds + \int_0^t \Sigma(Z_s^x)dW_s.$$

Then (Feynman-Kac)

$$u(T, x) = \mathbb{E}(\varphi(Z_T^x)).$$

Lemma (Beck-Becker-G-Jafaari-Jentzen (2018))

Let $X \sim \mathcal{U}_{[a,b]^d}$ and let $Y = \varphi(Z_X^T)$. The solution $\hat{U}$ of the mathematical learning problem with data distribution (X, Y) is given by

$$\hat{U}(x) = u(T, x), \quad x \in [a, b]^d,$$

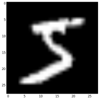
where u solves the corresponding Kolmogorov equation.

Solving linear Kolmogorov Equations by means of Neural Network Based Learning

The Vanilla DL Paradigm

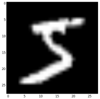
The Vanilla DL Paradigm

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:

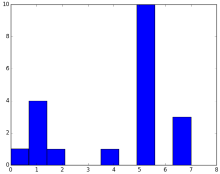


The Vanilla DL Paradigm

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:

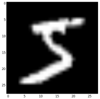


- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit

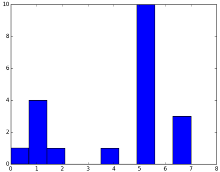


The Vanilla DL Paradigm

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:



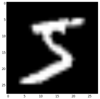
- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit



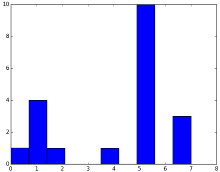
- Given labeled *training data* $(x_i, y_i)_{i=1}^m \subset \mathbb{R}^{784} \times \mathbb{R}^{10}$.

The Vanilla DL Paradigm

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:



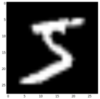
- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit



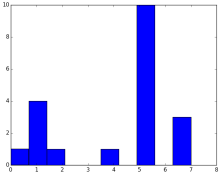
- Given labeled *training data* $(x_i, y_i)_{i=1}^{m} \subset \mathbb{R}^{784} \times \mathbb{R}^{10}$.
- Fix network architecture, e.g., number of layers (for example $L = 3$) and numbers of neurons ($N_1 = 30,\ N_2 = 30$).

The Vanilla DL Paradigm

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^{784}$:



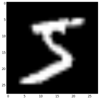
- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit



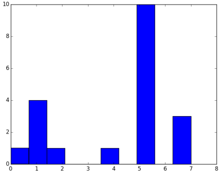
- Given labeled *training data* $(x_i, y_i)_{i=1}^{m} \subset \mathbb{R}^{784} \times \mathbb{R}^{10}$.
- Fix network architecture, e.g., number of layers (for example $L = 3$) and numbers of neurons $(N_1 = 30,\ N_2 = 30)$.
- The learning goal is to find the empirical regression function $f_{\mathbf{z}} \in \mathcal{H}^{\sigma}_{(784,30,30,10)}$.

The Vanilla DL Paradigm

- Every image is given as a 28×28 matrix $x \in \mathbb{R}^{28 \times 28} \sim \mathbb{R}^{784}$:



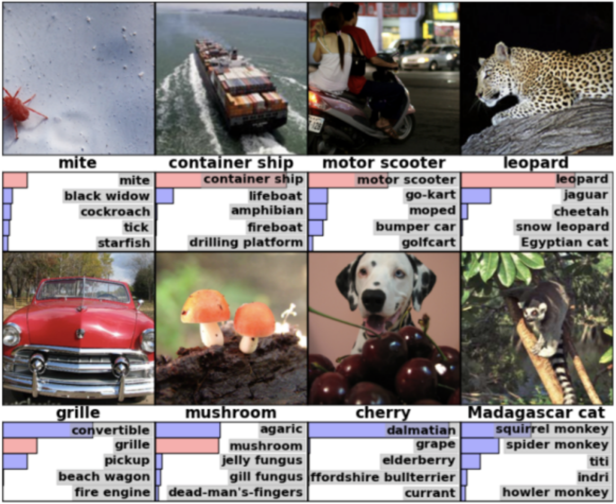
- Every label is given as a 10-dim vector $y \in \mathbb{R}^{10}$ describing the 'probability' of each digit



- Given labeled *training data* $(x_i, y_i)_{i=1}^{m} \subset \mathbb{R}^{784} \times \mathbb{R}^{10}$.
- Fix network architecture, e.g., number of layers (for example $L = 3$) and numbers of neurons ($N_1 = 30,\ N_2 = 30$).
- The learning goal is to find the empirical regression function $f_{\mathbf{z}} \in \mathcal{H}^{\sigma}_{(784,30,30,10)}$.
- Typically solved by stochastic first order optimization methods.

Description of Image Content

ImageNet Challenge



Deep Learning Algorithm

Deep Learning Algorithm

1. Generate training data $\mathbf{z} = (x_i, y_i)_{i=1}^m \overset{iid}{\sim} (X, \varphi(Z_X^T))$ by simulating Z_X^T with the Euler-Maruyama scheme.

Deep Learning Algorithm

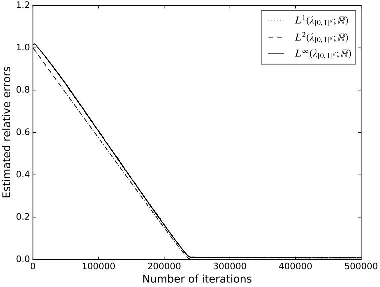
1. Generate training data $\mathbf{z} = (x_i, y_i)_{i=1}^m \overset{iid}{\sim} (X, \varphi(Z_X^T))$ by simulating Z_X^T with the Euler-Maruyama scheme.
2. Apply the Deep Learning Paradigm to this training data

Deep Learning Algorithm

1. Generate training data $\mathbf{z} = (x_i, y_i)_{i=1}^m \overset{iid}{\sim} (X, \varphi(Z_X^T))$ by simulating Z_X^T with the Euler-Maruyama scheme.
2. Apply the Deep Learning Paradigm to this training data ...meaning that
   (i) we pick a network architecture $(N_0 = d, N_1, \ldots, N_L = 1)$, and let $\mathcal{H} = \mathcal{H}^\sigma_{(N_0,\ldots,N_L)}$ and
   (ii) attempt to approximately compute

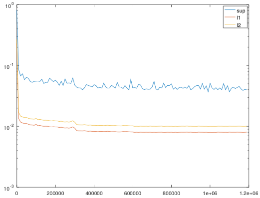
$$\hat{U}_{\mathcal{H},\mathbf{z}} = \operatorname*{argmin}_{U \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m (U(x_i) - y_i)^2$$

   in Tensorflow.



Number of steps	Relative $L^1(\lambda_{[0,1]^d};\mathcal{R})$-error	Relative $L^2(\lambda_{[0,1]^d};\mathcal{R})$-error	Relative $L^\infty(\lambda_{[0,1]^d};\mathcal{R})$-error	Runtime in seconds
0	0.998253	0.998254	1.003524	0.5
10000	0.957464	0.957536	0.993083	44.6
50000	0.786743	0.786806	0.828184	220.8
100000	0.574013	0.574060	0.605283	440.8
150000	0.361564	0.361594	0.384105	661.0
200000	0.001419	0.001784	0.010423	880.8
500000	0.001419	0.001784	0.010423	2200.7
750000	0.001419	0.001784	0.010423	3300.6

Figure: Estimated errors associated to the solution $u(1,\cdot)$ of the 100-dimensional parabolic PDE $\frac{\partial u}{\partial t}(t,x) = \Delta_x u(t,x)$, $u(0,x) = |x|^2$, $x \in [0,1]^{100}$.



Number of steps	Relative $L^1(\lambda_{[90,110]^d};\mathcal{R})$-error	Relative $L^2(\lambda_{[90,110]^d};\mathcal{R})$-error	Relative $L^\infty(\lambda_{[90,110]^d};\mathcal{R})$-error	Runtime in seconds
0	1.004285	1.004286	1.009524	1
25000	0.842938	0.843021	0.87884	110.2
50000	0.684955	0.685021	0.719826	219.5
100000	0.371515	0.371551	0.387978	437.9
150000	0.064605	0.064628	0.072259	656.2
250000	0.001220	0.001538	0.010039	1092.6
500000	0.000949	0.001187	0.005105	2183.8
750000	0.000902	0.001129	0.006028	3275.1

Figure: Estimated errors associated to the solution $u(T,\cdot)$ of the 100-dimensional uncorrelated Black Scholes PDE $\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i=1}^{d} |\sigma_i x_i|^2 (\frac{\partial^2 u}{\partial x_i^2})(t,x) + \sum_{i=1}^{d} \mu_i x_i (\frac{\partial u}{\partial x_i})(t,x)$, $u(0,x) = \exp(-rT)\max\{\max_{i\in\{1,2,\ldots,d\}} x_i - 100, 0\}$, $x \in [90,110]^{100}$.

Number of steps	Relative $L^1(\lambda_{[90,110]^d};\mathcal{R})$-error	Relative $L^2(\lambda_{[90,110]^d};\mathcal{R})$-error	Relative $L^\infty(\lambda_{[90,110]^d};\mathcal{R})$-error	Runtime in seconds
0	1.003383	1.003385	1.011662	0.8
25000	0.631420	0.631429	0.640633	112.1
50000	0.269053	0.269058	0.275114	223.3
100000	0.000752	0.000948	0.00553	445.8
150000	0.000694	0.00087	0.004662	668.2
250000	0.000604	0.000758	0.006483	1119.3
500000	0.000493	0.000615	0.002774	2292.8
750000	0.000471	0.00059	0.002862	3466.8

Figure: Estimated errors associated to the solution $u(T,\cdot)$ of the 100-dimensional correlated Black Scholes PDE $\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{d} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^d} \big(\frac{\partial^2 u}{\partial x_i \partial x_j}\big)(t,x) + \sum_{i=1}^{d} \mu_i x_i \big(\frac{\partial u}{\partial x_i}\big)(t,x)$, $u(0,x) = \exp(-\mu T) \max\big\{110 - \min_{i\in\{1,2,\dots,d\}}\{x_i\}, 0\big\}$, $x \in [90,110]^{100}$.

Number of steps	Relative $L^1(\lambda_{[90,110]^d};\mathcal{R})$-error	Relative $L^2(\lambda_{[90,110]^d};\mathcal{R})$-error	Relative $L^\infty(\lambda_{[90,110]^d};\mathcal{R})$-error	Runtime in seconds
0	1.003383	1.003385	1.011662	0.8
25000	0.631420	0.631429	0.640633	112.1
50000	0.269053	0.269058	0.275114	223.3
100000	0.000752	0.000948	0.00553	445.8
150000	0.000694	0.00087	0.004662	668.2
250000	0.000604	0.000758	0.006483	1119.3
500000	0.000493	0.000615	0.002774	2292.8
750000	0.000471	0.00059	0.002862	3466.8

Figure: Estimated errors associated to the solution $u(T,\cdot)$ of the 100-dimensional correlated Black Scholes PDE $\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\sum_{i,j=1}^{d} x_i x_j \beta_i \beta_j \langle \varsigma_i, \varsigma_j \rangle_{\mathbb{R}^d} (\frac{\partial^2 u}{\partial x_i \partial x_j})(t,x) + \sum_{i=1}^{d} \mu_i x_i (\frac{\partial u}{\partial x_i})(t,x)$, $u(0,x) = \exp(-\mu T)\max\{110 - \min_{i\in\{1,2,\dots,d\}}\{x_i\}, 0\}$, $x \in [90,110]^{100}$.

All computations were performed in single precision (float32) on a NVIDIA GeForce GTX 1080 GPU with 1974 MHz core clock and 8 GB GDDR5X memory with 1809.5 MHz clock rate. The underlying system consisted of an Intel Core i7-6800K CPU with 64 GB DDR4-2133 memory running Tensorflow 1.5 on Ubuntu 16.04.

Some Theoretical Results

Linear Affine Kolmogorov Equations

Given $\Sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, $\mu : \mathbb{R}^d \to \mathbb{R}^d$ affine and initial value $\varphi : \mathbb{R}^d \to \mathbb{R}$, find $u : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ with

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\left(\Sigma(x)\Sigma^T(x)\mathrm{Hess}_x u(t,x)\right) + \mu(x)\cdot\nabla_x u(t,x),$$
$$(t,x) \in [0,T]\times\mathbb{R}^d, \quad u(0,x) = \varphi(x).$$

Linear Affine Kolmogorov Equations

Given $\Sigma : \mathbb{R}^d \to \mathbb{R}^{d \times d}$, $\mu : \mathbb{R}^d \to \mathbb{R}^d$ affine and initial value $\varphi : \mathbb{R}^d \to \mathbb{R}$, find $u : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ with

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\left(\Sigma(x)\Sigma^T(x)\mathrm{Hess}_x u(t,x)\right) + \mu(x) \cdot \nabla_x u(t,x),$$
$$(t,x) \in [0,T] \times \mathbb{R}^d, \quad u(0,x) = \varphi(x).$$

Includes Black-Scholes Equation with correlations!

Linear Affine Kolmogorov Equations

Given $\Sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, $\mu : \mathbb{R}^d \to \mathbb{R}^d$ affine and initial value $\varphi : \mathbb{R}^d \to \mathbb{R}$, find $u : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ with

$$\frac{\partial u}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\left(\Sigma(x)\Sigma^T(x)\mathrm{Hess}_x u(t,x)\right) + \mu(x)\cdot\nabla_x u(t,x),$$
$$(t,x) \in [0,T]\times\mathbb{R}^d, \quad u(0,x) = \varphi(x).$$

Includes Black-Scholes Equation with correlations!

Theorem [G-Hornung-Jentzen-Von Wurstenberger (2018)], simplified version

Suppose that $\varphi \in \mathcal{H}^\sigma_{(N_0,\dots,N_L)}$ (or can be well approximated by NNs). Then for all $\epsilon > 0$ there is Φ_ϵ with $\mathrm{size}(\Phi_\epsilon) \lesssim \mathrm{size}(\varphi)\cdot\epsilon^{-2}$ and

$$\sup_{x\in[a,b]^d} |u(T,x) - R_\sigma(\Phi_\epsilon)(x)| \le \epsilon.$$

The implicit constant depends at most polynomially on the dimension $d = N_0$.

Option Pricing without Curse of Dimensionality

Theorem [Berner-G-Jentzen (2018)], very special case

Let $\varphi(x) = \min\{\max\{\max(x_i - K_i), 0\}, R\}$ or $\varphi(x) = \min\{\max\{\sum_{i=1}^{d} x_i - K, 0\}, R\}$ (or any typical option). Then for all $\epsilon > 0$ there is $\Phi_\epsilon \in \mathcal{H}^{ReLU}_{(N_0,\ldots,N_L)}$ with $\mathrm{size}(\Phi_\epsilon) = \mathcal{O}(\epsilon^{-2})$ and

$$\frac{1}{(b-a)^{d/2}} \left(\int_{[a,b]^d} |u(T,x) - R_\sigma(\Phi_\epsilon)(x)|^2 dx \right)^{1/2} \leq \epsilon.$$

Such networks can be found by solving the ERM problem with $m \sim \epsilon^{-4}$ samples. The implicit constants depend at most polynomially on the dimension $d = N_0$!

Option Pricing without Curse of Dimensionality

Theorem [Berner-G-Jentzen (2018)], very special case

Let $\varphi(x) = \min\{\max\{\max(x_i - K_i), 0\}, R\}$ or $\varphi(x) = \min\{\max\{\sum_{i=1}^{d} x_i - K, 0\}, R\}$ (or any typical option). Then for all $\epsilon > 0$ there is $\Phi_\epsilon \in \mathcal{H}^{ReLU}_{(N_0,\ldots,N_L)}$ with $\mathrm{size}(\Phi_\epsilon) = \mathcal{O}(\epsilon^{-2})$ and

$$\frac{1}{(b-a)^{d/2}} \left(\int_{[a,b]^d} |u(T,x) - R_\sigma(\Phi_\epsilon)(x)|^2 dx \right)^{1/2} \leq \epsilon.$$

Such networks can be found by solving the ERM problem with $m \sim \epsilon^{-4}$ samples. The implicit constants depend at most polynomially on the dimension $d = N_0$!

Due to compositional structure of NNs, all results hold also for options operating on options...

Wrap Up

Wrap Up

- Several PDEs can be reformulated as learning problem

Wrap Up

- Several PDEs can be reformulated as learning problem
- Neural network based numerical solution of high-dimensional PDEs is extremely promising both empirically and mathematically – and it is possible to prove real theorems!

Wrap Up

- Several PDEs can be reformulated as learning problem
- Neural network based numerical solution of high-dimensional PDEs is extremely promising both empirically and mathematically – and it is possible to prove real theorems!
- Specifically, we can prove that these methods are capable of overcoming the curse of dimensionality for an important class of PDEs arising in computational finance.

Wrap Up

- Several PDEs can be reformulated as learning problem
- Neural network based numerical solution of high-dimensional PDEs is extremely promising both empirically and mathematically – and it is possible to prove real theorems!
- Specifically, we can prove that these methods are capable of overcoming the curse of dimensionality for an important class of PDEs arising in computational finance.
- We can observe these properties in simulations.

Thank You!

Questions?

Literature

- Beck, Becker, G, Jaafari, Jentzen. *Solving Stochastic Differential Equations and Kolmogorov Equations by Means of Deep Learning*. ArXiv 1806.00421.
- Elbrächter, G, Jentzen, Schwab. *DNN Expression Rate Analysis of High Dimensional PDEs: Applications in Option Pricing*. ArXiv 1806.xxxxx.
- Perekreshtenko, G, Elbrächter, Bölcskei. *The universal approximation power of finite-width deep ReLU networks*. ArXiv 1806.01528.
- G, Hornung, Jentzen, Von Wurstemberger. *A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations*. ArXiv 1809.xxxxx.
- Berner, G, Jentzen. *Empirical risk minimization over deep neural network hypothesis classes breaks the curse of dimensionality for the numerical approximatoin of Black-Scholes partial differential equations*. ArXiv 1809.xxxxx.