Deep BSDE algorithm

Ludovic Goudenège

March 3, 2020

Premia 22

This documentation is based on the article "Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning" from Han, Jentzen and E. and on the article from the same authors "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations".

1 Launch the python program

You can launch the computation by the command:
`python main.py -config_path=configs/XXXX.json`
where `XXX` should be one of the following problems of the section 2. All the problems are saved in the folder `./configs` but you can change the path in the previous command.

There are also two other commands:
`exp_name`: Name of numerical experiment, prefix of logging and output.
`log_dir`: Directory to write logging and output array.

2 Problems

Here is the list of problems detailed in the articles [1] and [2].

1. HJBLQ: Hamilton-Jacobi-Bellman (HJB) equation [1].
2. AllenCahn: Allen-Cahn equation with a cubic nonlinearity [1].
3. PricingDefaultRisk: Nonlinear Black-Scholes equation with default risk in consideration [1].
4. PricingDiffRate: Nonlinear Black-Scholes equation for the pricing of European financial derivatives with different interest rates for borrowing and lending [2].
5. BurgersType: Multidimensional Burgers-type PDEs with explicit solution [2].
6. QuadraticGradient: An example PDE with quadratically growing derivatives and an explicit solution [2].
7. ReactionDiffusion: Time-dependent reaction-diffusion-type example PDE with oscillating explicit solutions [2].

3 Adding problems

New problems can be added very easily. Inherit the class equation in equation.py and define the new problem. Note that the generator function and terminal function should be TensorFlow operations while the sample function can be python operation. A proper config file is needed as well.

4 Example of "config file"

This config file is from the problem 3 from the article [1].

```
{
  "eqn_config": {
    "_comment": "Nonlinear Black-Scholes equation with default risk in PNAS paper
     doi.org/10.1073/pnas.1718942115",
    "eqn_name": "PricingDefaultRisk",
    "total_time": 1.0,
    "dim": 100,
    "num_time_interval": 40
  },
  "net_config": {
    "y_init_range": [40, 50],
    "num_hiddens": [110, 110],
    "lr_values": [8e-3, 8e-3],
    "lr_boundaries": [3000],
    "num_iterations": 6000,
    "batch_size": 64,
    "valid_size": 256,
    "logging_frequency": 100,
    "dtype": "float64",
    "verbose": true
  }
}
```

The first parameters described the equation of the problem.

- `eqn_name` is the name of the equation described in the file equation.py.
- `total_time` is the maturity.
- `dim` is the dimension of the process.
- `num_time` is the number of time interval for the time discretization.

The following one described the neural network. You can change it to improve performances or topology of the network, but this shape works well.

References

[1] Han, J., Jentzen, A., and E, W. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning Proceedings of the National Academy of Sciences, 115(34), 8505-8510 (2018). https://doi.org/10.1073/pnas.1718942115 https://arxiv.org/abs/1707.02568 1, 2

[2] Han, J., Jentzen, A., and E, W. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations Communications in Mathematics and Statistics, 5, 349–380 (2017). https://doi.org/10.1007/s40304-017-0117-6 https://arxiv.org/abs/1706.04702 1