String Theory meets Machine Learning - Hyperparameter Optimization

Robin Schneider

Uppsala University

November 2020

Performance

How can we improve performance?

Performance

How can we improve performance?

- Longer training time
- More data
- Data augmentation
- Feature engineering
- Changing architecture

Performance

How can we improve performance?

- Longer training time
- More data
- Data augmentation
- Feature engineering
- Changing architecture

Changing all kinds of hyperparameters.

(Hyper-)Parameters

Here are some hyperparameters that influence the success of our experiments:

(Hyper-)Parameters

Here are some hyperparameters that influence the success of our experiments:

- activation function
- number of hidden units and layers
- regularization parameters: l1, l2, dropout rate
- learning rate, momentum, batch size
- kernel size, stride and padding
- weight initialization

(Hyper-)Parameters

Here are some hyperparameters that influence the success of our experiments:

- activation function
- number of hidden units and layers
- regularization parameters: l1, l2, dropout rate
- learning rate, momentum, batch size
- kernel size, stride and padding
- weight initialization

Other parameters:

- problem formulation
- data used (e.g. ratio of train to val split, discarding of direct products)
- seed (this might be bit controversial)
- computational package (this too)

Hyperparameter optimization

How do we find the best values for our hyperparameters?

Hyperparameter optimization

How do we find the best values for our hyperparameters?
Trial and error :(

Hyperparameter optimization

How do we find the best values for our hyperparameters?
Trial and error :(

- Human intuition
- Grid Search
- Random Search
- Bayesian Optimization (GP, TPE)
- Hyperband and BOHB
- (Genetic Algorithms)

CNNs and MNIST

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 32)          9248
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 10)                5130
=================================================================
Total params: 23,946
Trainable params: 23,946
Non-trainable params: 0
```

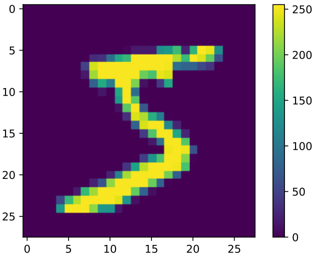

Figure: *Data sample of MNIST set and CNN to tackle the classification problem.*

MNIST data set. (60.000, 10.000) train and test images. 28x28 with one color channel in the range of $\{0, ..., 255\}$.

Grid Search

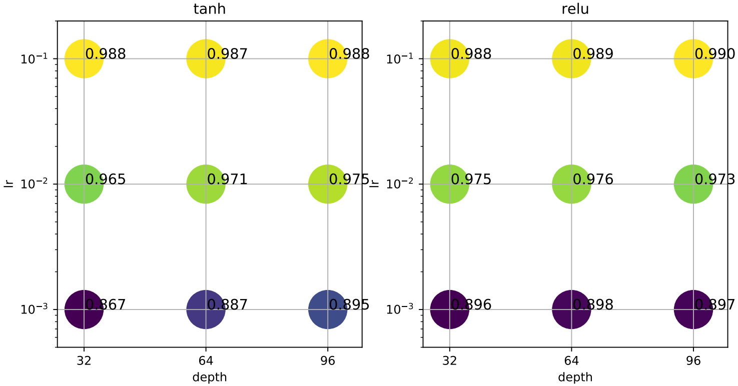

Figure: *Grid search over activation, depth and learning rate.*

Random Search

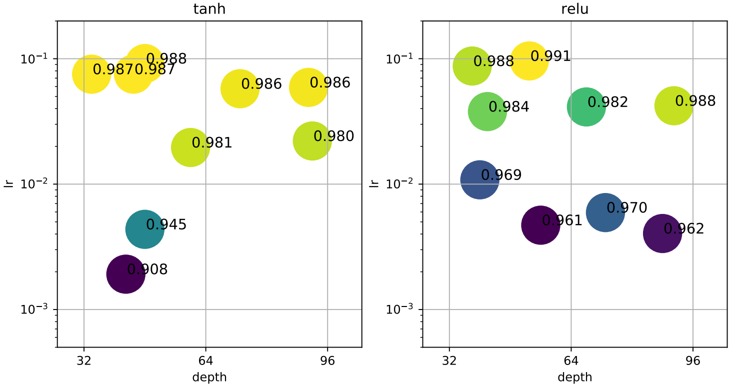

Figure: *Random search over activation, depth and learning rate.*

Exploration and Exploitation

Multi armed bandit

- Maximize gain with finite number of resources
- We want to minimize regret

$$\rho = T \cdot \mu^* - \sum_t \hat{r}_t \tag{1}$$

  with T number of round/configurations, μ^* maximal reward and $\hat{r}_t$ the reward in round t.

- Problem of reinforcement learning

Exploration and Exploitation

Multi armed bandit

- Maximize gain with finite number of resources
- We want to minimize regret

$$\rho = T \cdot \mu^* - \sum_t \hat{r}_t \tag{1}$$

with T number of round/configurations, μ^* maximal reward and $\hat{r}_t$ the reward in round t.

- Problem of reinforcement learning

In practice

- Many hyperparameters are correlated
- Should utilize prior knowledge

Bayesian Optimization

Utilizing prior results to get better estimates of future configurations.

- We model $p(f(\lambda)|D_n)$, where $\lambda \in D_n$ is a data collection of hyperparameters.
- Use **acquisitions** function $a : \lambda \rightarrow \mathbb{R}^+$ which trades exploration vs exploitation which will be maximized

Bayesian Optimization

Utilizing prior results to get better estimates of future configurations.

- We model $p(f(\lambda)|D_n)$, where $\lambda \in D_n$ is a data collection of hyperparameters.
- Use **acquisitions** function $a : \lambda \to \mathbb{R}^+$ which trades exploration vs exploitation which will be maximized
- Select new parameters: $\lambda_{n+1} = \arg\max_\lambda a(\lambda)$, add new data point $\{f_{n+1}, \lambda_{n+1}\}$ to collection D_{n+1} improve posterior $p(f|D_{n+1})$
- Common acquisitions function is **Expected Improvement**

$$a(\lambda_i) = \mathrm{EI}(\lambda_i) = \mathbb{E}_{p(f|D)}[\max(f(\lambda) - f(\lambda^+))] \tag{2}$$

  others are Upper Confident Bound or (Maximum) Probability of Improvement.

Gaussian optimization

Gaussian optimization has been shown to consistently outperform random/grid search [1206.2944].

- Assume f has some Gaussian process prior given by

$$f \sim \mathcal{GP}(m(\lambda), \Sigma_\theta(\lambda, \lambda')) = \mathcal{N}(y|m(\lambda), \Sigma_\theta(\lambda, \lambda')) \tag{3}$$

  where $\lambda \in D_n$ and Σ_θ is some covariance matrix
- Take for example the following kernel

$$\Sigma_\theta(\lambda_i, \lambda_j) = \theta_1 \exp\left[-\frac{1}{2\theta_2}||\lambda_i - \lambda_j||^2\right] \tag{4}$$

Gaussian optimization

Gaussian optimization has been shown to consistently outperform random/grid search [1206.2944].

- Assume f has some Gaussian process prior given by

$$f \sim \mathcal{GP}(m(\lambda), \Sigma_\theta(\lambda, \lambda')) = \mathcal{N}(y|m(\lambda), \Sigma_\theta(\lambda, \lambda')) \tag{3}$$

  where $\lambda \in D_n$ and Σ_θ is some covariance matrix
- Take for example the following kernel

$$\Sigma_\theta(\lambda_i, \lambda_j) = \theta_1 \exp\left[-\frac{1}{2\theta_2}||\lambda_i - \lambda_j||^2\right] \tag{4}$$

- Get posterior $p(f_{n+1}|D_n, \lambda_{n+1}) = \mathcal{N}(y|\mu_n(\lambda_{n+1}), \sigma_n(\lambda_{n+1}))$. Select new parameters according to Expected Improvement with

$$\mathrm{EI}(\lambda) = \begin{cases} (\mu(\lambda) - f^+ - \xi)P(\gamma > f^+) + \sigma(\lambda)\phi(\gamma) & \\ 0 & \text{if } \sigma(\lambda) = 0 \end{cases} \tag{5}$$

  with $\gamma = \frac{\mu(\lambda) - f^+ - \xi}{\sigma(\lambda)}$ and $\phi(\gamma)$ being the PDF.

Gaussian in a picture

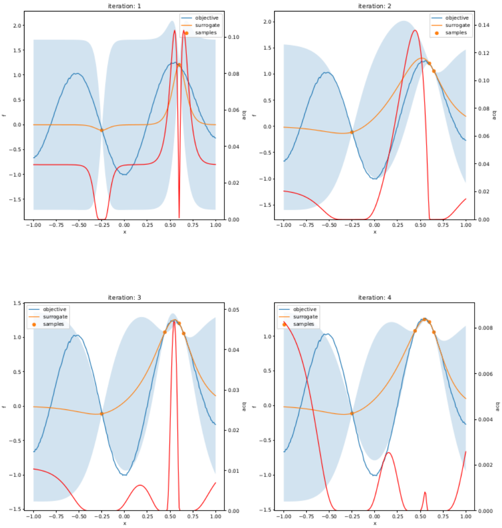

Figure: *Gaussian process on 1D problem.*

Gaussian in a picture

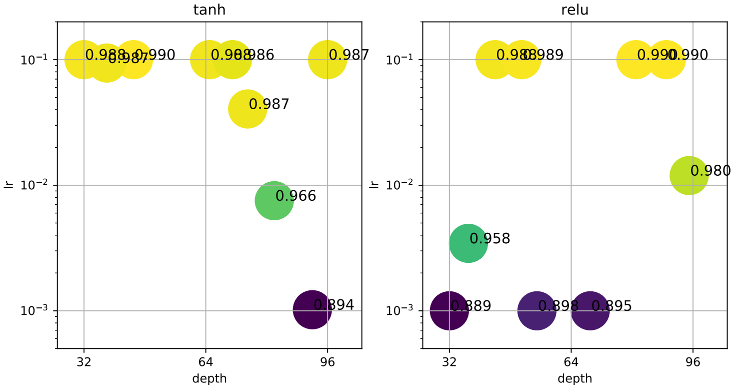

Figure: *Bayesian optimization with GP over activation, depth and learning rate using [github.com/fmfn/BayesianOptimization].*

General thoughts

Possible improvements? What to keep in mind? Possible problems?

General thoughts

Possible improvements? What to keep in mind? Possible problems?

- We do not have an infinite budget.
- We might have access to a cluster for parallel distribution.
- Bad configuration should be stopped early.
- Simplicity and adaptability.

Hyperband

Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization [1603.06560].

- Solution to infinite-armed bandit in non-stochastic setting and comes within log factors of stochastic setting
- Improve performance via adaptive resource allocation and early stopping
- Outperforms standard Bayesian Optimization (up to factor of 30)

Hyperband

Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization [1603.06560].

- Solution to infinite-armed bandit in non-stochastic setting and comes within log factors of stochastic setting
- Improve performance via adaptive resource allocation and early stopping
- Outperforms standard Bayesian Optimization (up to factor of 30)
- Builds up on **successive halving** with finite budget B and number of test configurations n. Example: $B = 80, n = 16$, then goes through iterations $i.(n_i, r_i)$: $1.(16, 1), 2.(8, 2), 3.(4, 4), 4.(2, 8), 5.(1, 16)$.
- Two Hyperparameters: R maximum budget to a single computation, η controls discarded configurations. Introduce $s_{\max} + 1$ with $s_{\max} \approx \log_\eta(R)$ brackets of successive halving. Each bracket uses up to B resources for a total of about $(s_{\max} + 1)B$ compute.

Hyperband II

Table: Hyperband brackets with $\eta = 3$ and budget $R = 81$. On the left we have successive halving with $B = 5 \cdot 81$ and $n = 81$ while to the right $n = 5$.

	s =	4	s =	3	s =	2	s =	1	s =	0
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1								

Hyperband II

Table: Hyperband brackets with $\eta = 3$ and budget $R = 81$. On the left we have successive halving with $B = 5 \cdot 81$ and $n = 81$ while to the right $n = 5$.

	s =	4	s =	3	s =	2	s =	1	s =	0
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3						

Hyperband II

Table: Hyperband brackets with $\eta = 3$ and budget $R = 81$. On the left we have successive halving with $B = 5 \cdot 81$ and $n = 81$ while to the right $n = 5$.

	s =	4	s =	3	s =	2	s =	1	s =	0
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9				

Hyperband II

Table: Hyperband brackets with $\eta = 3$ and budget $R = 81$. On the left we have successive halving with $B = 5 \cdot 81$ and $n = 81$ while to the right $n = 5$.

	s =	4	s =	3	s =	2	s =	1	s =	0
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27		

Hyperband II

Table: Hyperband brackets with $\eta = 3$ and budget $R = 81$. On the left we have successive halving with $B = 5 \cdot 81$ and $n = 81$ while to the right $n = 5$.

	s =	4	s =	3	s =	2	s =	1	s =	0
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1										

Hyperband II

Table: Hyperband brackets with $\eta = 3$ and budget $R = 81$. On the left we have successive halving with $B = 5 \cdot 81$ and $n = 81$ while to the right $n = 5$.

	s =	4	s =	3	s =	2	s =	1	s =	0
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

BOHB

We start from Hyperband, but instead of random picking all the way, we use Bayesian optimization [1807.01774].

- Uses Tree Parzen Estimator rather than Gaussian process.
- Combines the best of both worlds. Fast convergence + better exploitation.
- Strong anytime performance.
- Strong final performance.
- Tested in various different settings: Kernel methods, deep learning, deep reinforcement learning.
- Available as a python package.

BOHB in a picture

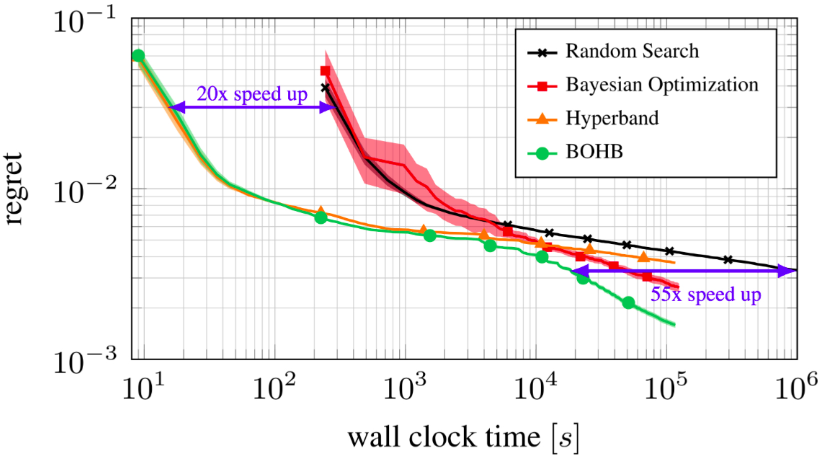

Figure: *BOHB for a neural network over six hyperparameters, taken from [automl.org/blog_bohb/].*

Application: Learning CY metrics

Are there any numerical problems in string theory? Any problems where we do not require 100 % accuracy?

- We will learn how to construct NNs, which approximate CY metrics.
- 'Classical' Donaldson algorithm is *very* expensive.
- Dataset: 100.000 points, evaluation of holomorphic volume form and integration weights on Fermat quintic.

Application: Learning CY metrics

Are there any numerical problems in string theory? Any problems where we do not require 100 % accuracy?

- We will learn how to construct NNs, which approximate CY metrics.
- 'Classical' Donaldson algorithm is *very* expensive.
- Dataset: 100.000 points, evaluation of holomorphic volume form and integration weights on Fermat quintic.
- There are scalar quantities that measure the deviation from CY metric.
- Write custom loss and optimize.

Application: Learning CY metrics

We have

$$\mathrm{Vol}_{\mathrm{CY}} = \frac{1}{N}\sum_{i}^{N} w_i, \qquad \mathrm{Vol}_{\mathrm{K}} = \frac{1}{N}\sum_{i}^{N} \frac{\omega^3(p_i)}{\Omega(p_i)\wedge\bar{\Omega}(p_i)} w_i \tag{6}$$

- Sigma measure

$$\sigma = \frac{1}{N_t \mathrm{Vol}_{\mathrm{CY}}}\sum_{i=1}^{N_t} \left|1 - \frac{\omega^3(p_i)/\mathrm{Vol}_{\mathrm{K}}}{\Omega(p_i)\wedge\Omega(\bar{p}_i)/\mathrm{Vol}_{\mathrm{CY}}}\right| w_i \tag{7}$$

- Ricci measure

$$||R|| = \frac{1}{N_t \mathrm{Vol}_{\mathrm{K}}^{2/3}}\sum_{i=1}^{N_t} \frac{\omega^3(p_i)}{\Omega(p_i)\wedge\Omega(\bar{p}_i)} |R(p_i)| w_i \tag{8}$$