OPTIMIZATION METHODS FOR ENGINEERING PROBLEMS

TOPIC: *Stochastic Gradient*

ANTONIO CARLUCCI - 303327

ANAIS DELEPAUT – 294258

IMAN EBRAHIMI MEHR – 304089

OLIVIERO VOUCH - 293131

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv:1609.04747.

REGRESSION ON PARKINSON DATA

- Patients affected by **Parkinson**'s disease cannot exactly control their muscles. In particular they show tremor, they walk with difficulties and have problems in starting a movement.
- The severity of the illness is measured by neurologists according to the UPDRS (**Unified Parkinson's Disease Rating Scale**). The visit for the assignment of the UPDRS score takes a lot of time, and different neurologists may give slightly different scores.
- It would be useful to find an automatic way to give the patient an objective score, which can be measured several times during the day and help the neurologist to optimize medical treatment.

Given a dataset recording a set of features (medical parameters) for each patient collected over 6 months

↓

Regress UPDRS from the other features

PARKINSON DATASET

Patient ID	Age	Sex	Test time (days)	F1	F2	..	F18	UPDRS
1	72	M	5.643	0.0032	23.05	..	0.161	34.398
1	72	M	12.666	0.0015	22.98	..	0.183	34.894
2	74	M	3.866	0.0026	21.64	..	0.162	37.363
:	:	:	:	:	:	..	:	:
42	69	F	4.263	0.0043	23.42	..	0.195	32.495

N_T EXAMPLES

GROUND TRUTH Y

PATIENT INFORMATION

FEATURES **X** (medical parameters)

REGRESSAND (prediction $\hat{Y}$)

REGRESSION MODEL (linear)

$$\hat{Y}(\boldsymbol{\theta}) = \mathbf{m}^T\mathbf{X} + b$$
$$\boldsymbol{\theta} = (\mathbf{m}, b) \in \mathbb{R}^{18}$$

MSE COST FUNCTION (quadratic cost, <u>convex</u>)

$$J(\boldsymbol{\theta}) = \frac{1}{N_T}\sum_{i=1}^{N_T}\left(Y_i - \hat{Y}_i(\boldsymbol{\theta})\right)^2$$

GRADIENT DESCENT AT A GLANCE

Given the cost function $J(\boldsymbol{\theta})$ parametrized by model $\boldsymbol{\theta} \in \mathbb{R}^d$ with d degrees of freedom (DoF)

Minimize $J(\boldsymbol{\theta})$ by iteratively updating $\boldsymbol{\theta}$ in the opposite direction of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

BATCH GRADIENT DESCENT
(a.k.a. Vanilla Gradient Descent)

- Fix learning rate η
- Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ over the whole training dataset
- Single update of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

− Slow update

− NO *online* model update

STOCHASTIC GRADIENT DESCENT (SGD)

- Fix learning rate η
- Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, x^i, y^i)$ for each training example (x^i, y^i)
- Update of $\boldsymbol{\theta}$ for each (x^i, y^i):

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, x^i, y^i)$$

+ Fast update

+ *online* model update

− High-variance updates

MINI-BATCH GRADIENT DESCENT

- Fix learning rate η
- Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, x^i, y^i)$ over a batch of n training examples $(x^{(i:i+n)}, y^{(i:i+n)})$
- Update of $\boldsymbol{\theta}$ for each batch:

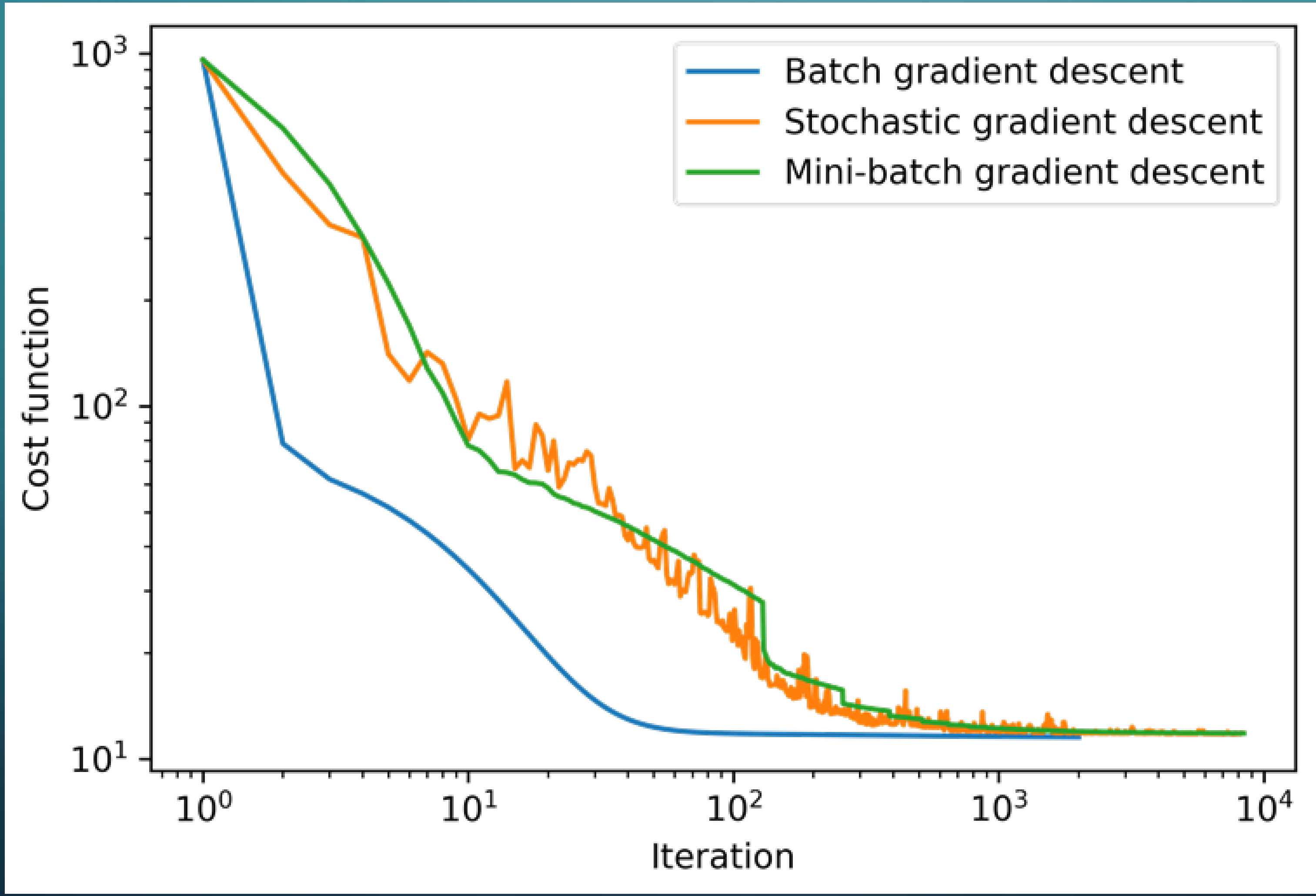
$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, x^{(i:i+n)}, y^{(i:i+n)})$$

+ Fast update

+ *online* model update

+ Low-variance updates

GRADIENT DESCENT REGRESSION



Initial cost: 953.436
Optimal cost: 11.442
Batch GD: 11.545
Stochastic GD: 11.825
Mini-batch GD: 11.826

GRADIENT DESCENT: CHALLENGES

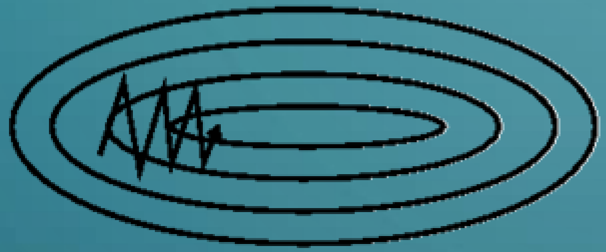
- How to fix the learning rate? If too small, slow convergence. If too large, loss function fluctuations or divergence
- Learning rate schedules to reduce η according to a pre-defined schedule or when parameters' change is below a threshold. But schedule or thresholds must be defined in advance (no adaptation)
- Same learning rate applies for all parameter updates. This is not good for sparse datasets or when features vary with different frequencies.
- For non-convex optimization problems, how to avoid getting trapped in sub-optimal local minima (i.e., saddle points)?

↓

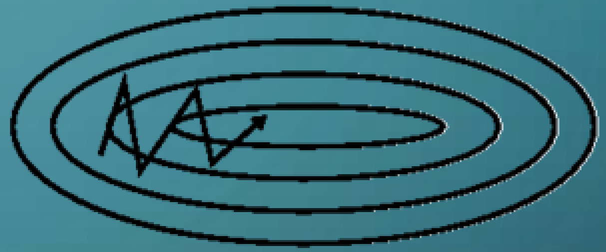
Momentum | **RMSprop** | **Adam**

MOMENTUM OPTIMIZATION

Accelerate gradient descent in the relevant optimal direction and dampen oscillations of the parameter updates around local optima



Without Momentum



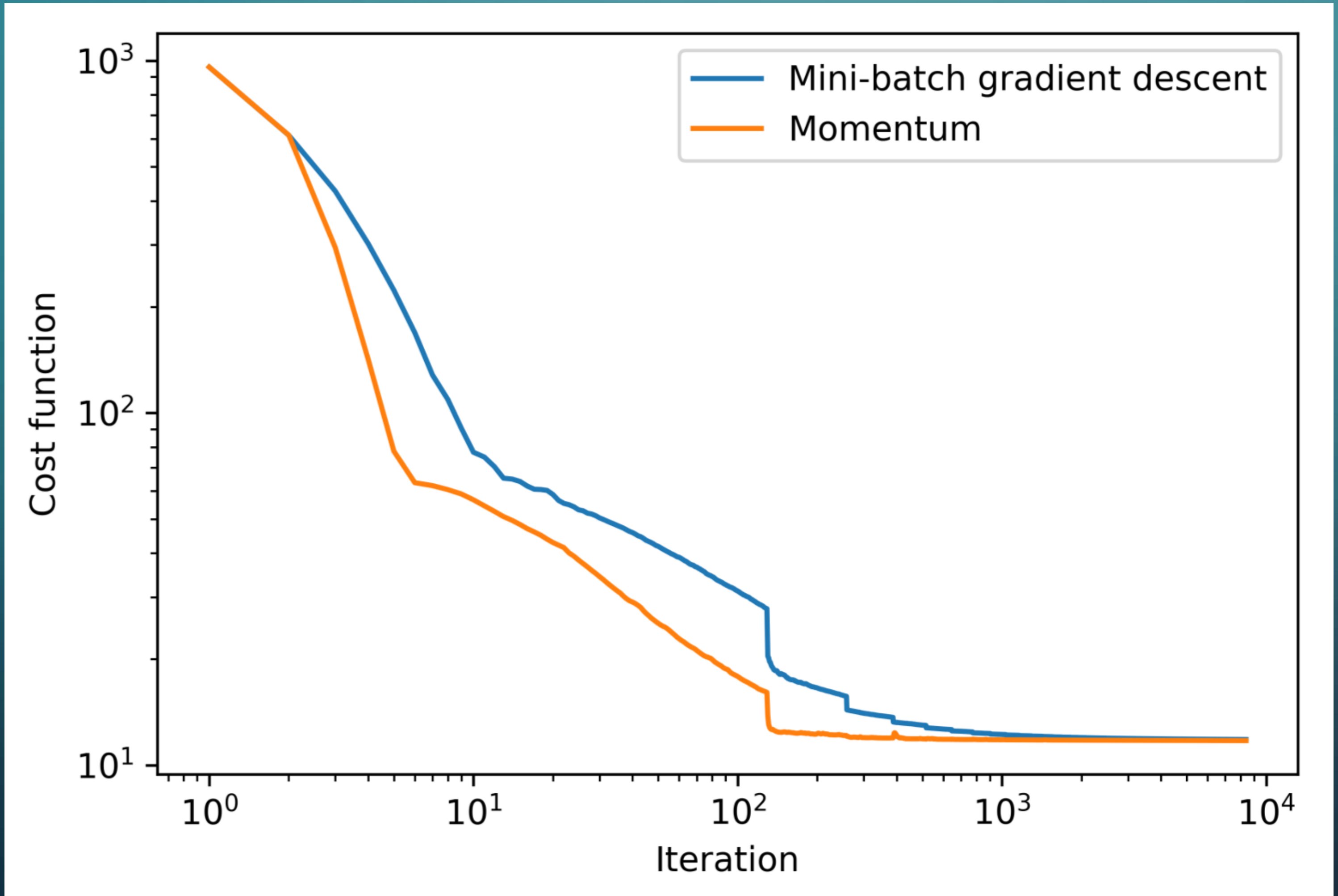
With Momentum

Calculate the update vector by adding a fraction γ of the previous update to the gradient

$$v_t = \gamma v_{t-1} + \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} - v_t$$

GRADIENT DESCENT & MOMENTUM



Initial cost: 953.436
Mini-batch GD: 11.826
Momentum: 11.740

RMSPROP OPTIMIZATION

Adapt the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent parameters

θ_k

- $g_k^i = \nabla_{\theta^i} J(\theta_k^i)$ → Gradient of θ_k
- $\mathbb{E}_{w,i}[\boldsymbol{g}_k^2]$ → Exponentially decaying average over w

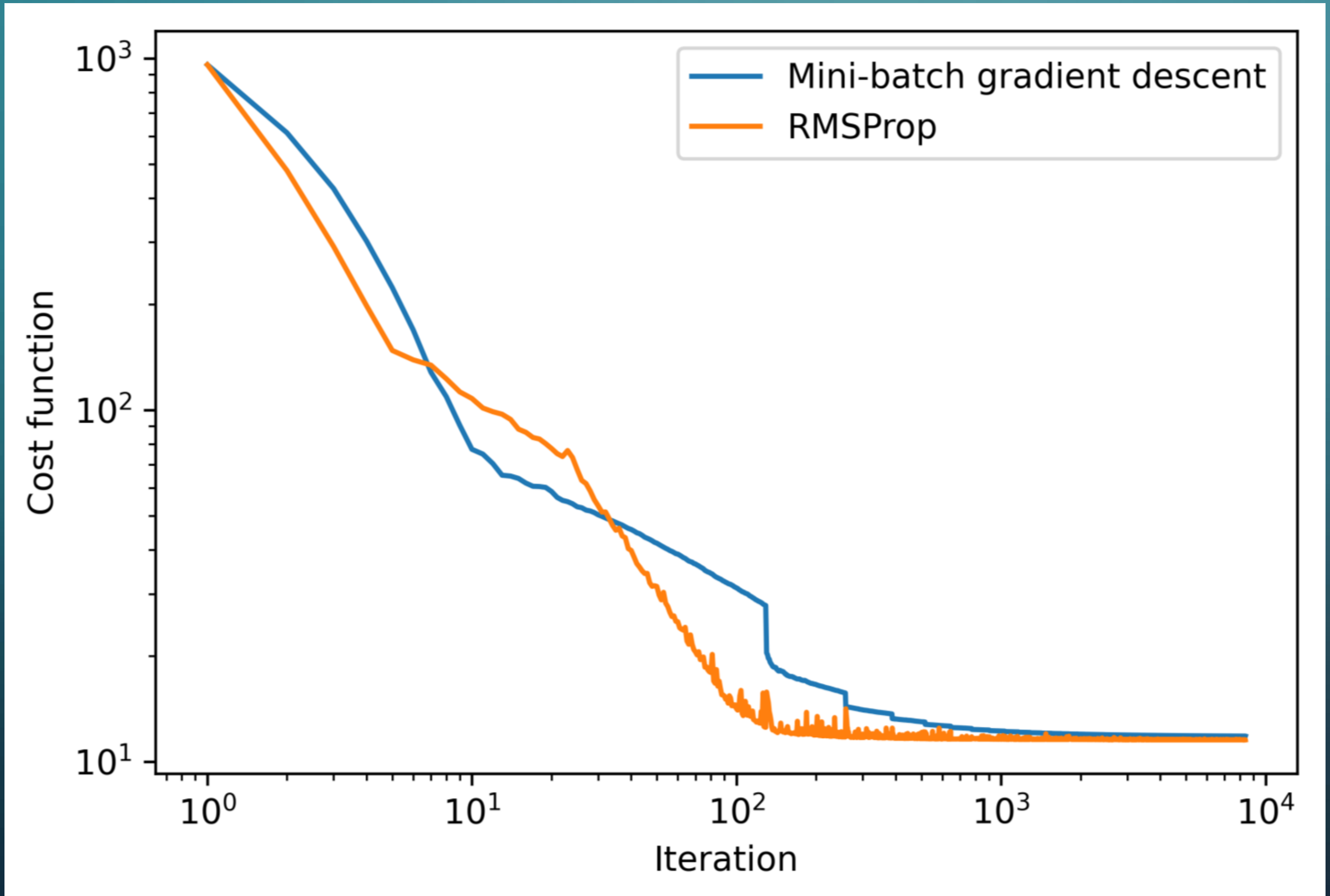
Calculate the update vector for parameter θ_k based on the window of accumulated past gradients and the current gradient

$$\mathbb{E}_{w,i}[\boldsymbol{g}_k^2] = 0.9\mathbb{E}_{w,i-1}[\boldsymbol{g}_k^2] + 0.1 {g_k^i}^2$$

$$\theta_k^{i+1} = \theta_k^i - \frac{\eta}{\sqrt{\mathbb{E}_{w,i}[\boldsymbol{g}_k^2] + \epsilon}} g_k^i$$

ϵ: SMOOTHING TERM

GRADIENT DESCENT & RMSPROP



Initial cost: 953.436
Mini-batch GD: 11.826
RMSprop: 11.539

ADAM OPTIMIZATION

Adapt the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent parameters

$$\theta_k \begin{cases} m_k^i = \beta_1 m_k^{i-1} + (1-\beta_1) g_k^i \\ v_k^i = \beta_2 v_k^{i-1} + (1-\beta_2) g_k^{i^2} \end{cases}$$

m_k^i → Exponentially decaying average of past gradients (Momentum)

v_k^i → Exponentially decaying average of past squared gradients (RMSprop)

DECAY RATES (β_1, β_2)

+

$$\hat{m}_k^i = m_k^i / 1 - \beta_1$$

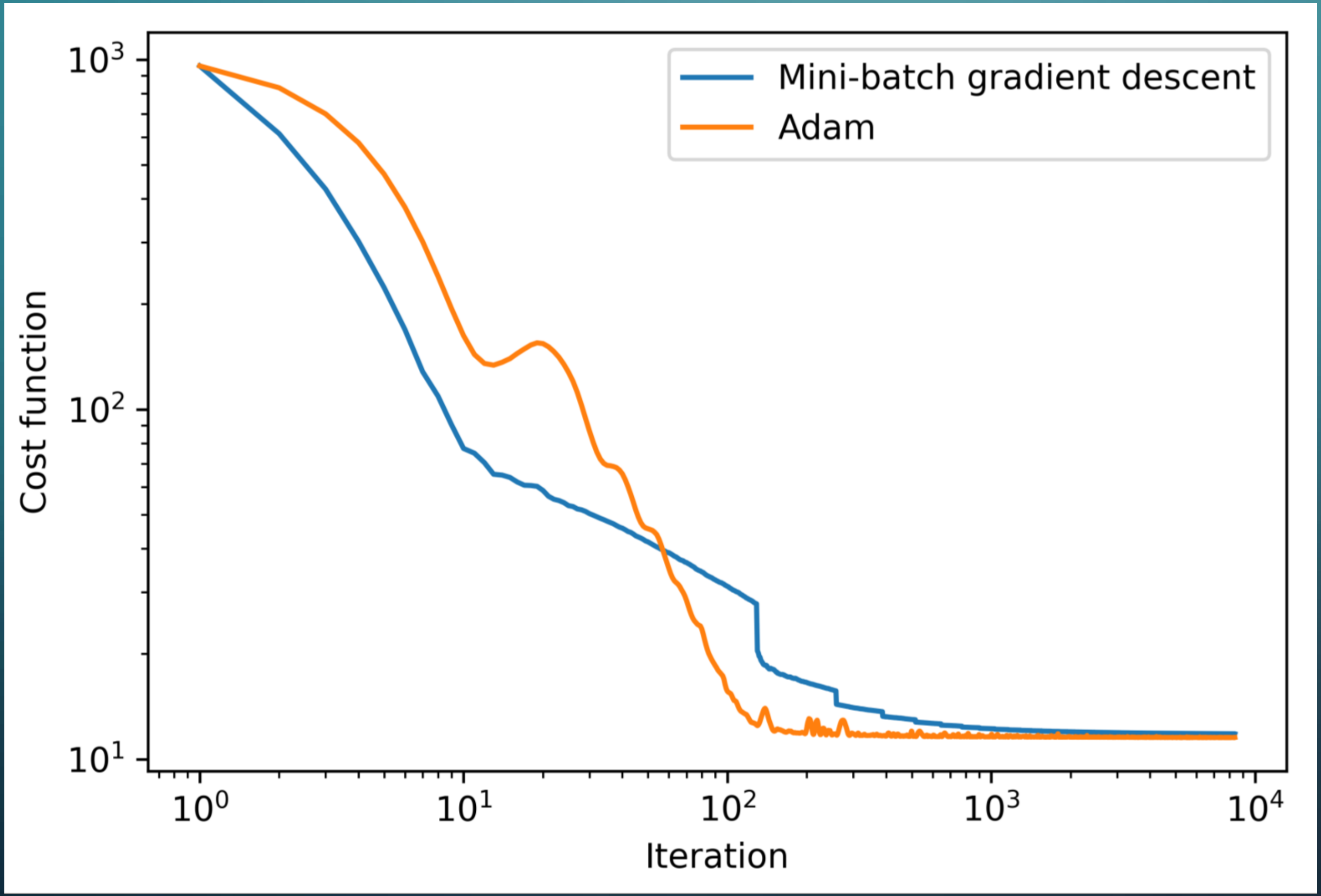
$$\hat{v}_k^i = v_k^i / 1 - \beta_2$$

BIAS CORRECTION

Calculate the update vector for parameter θ_k based on the estimates of the first moment m_k^i and the second moment v_k^i of the gradient g_k

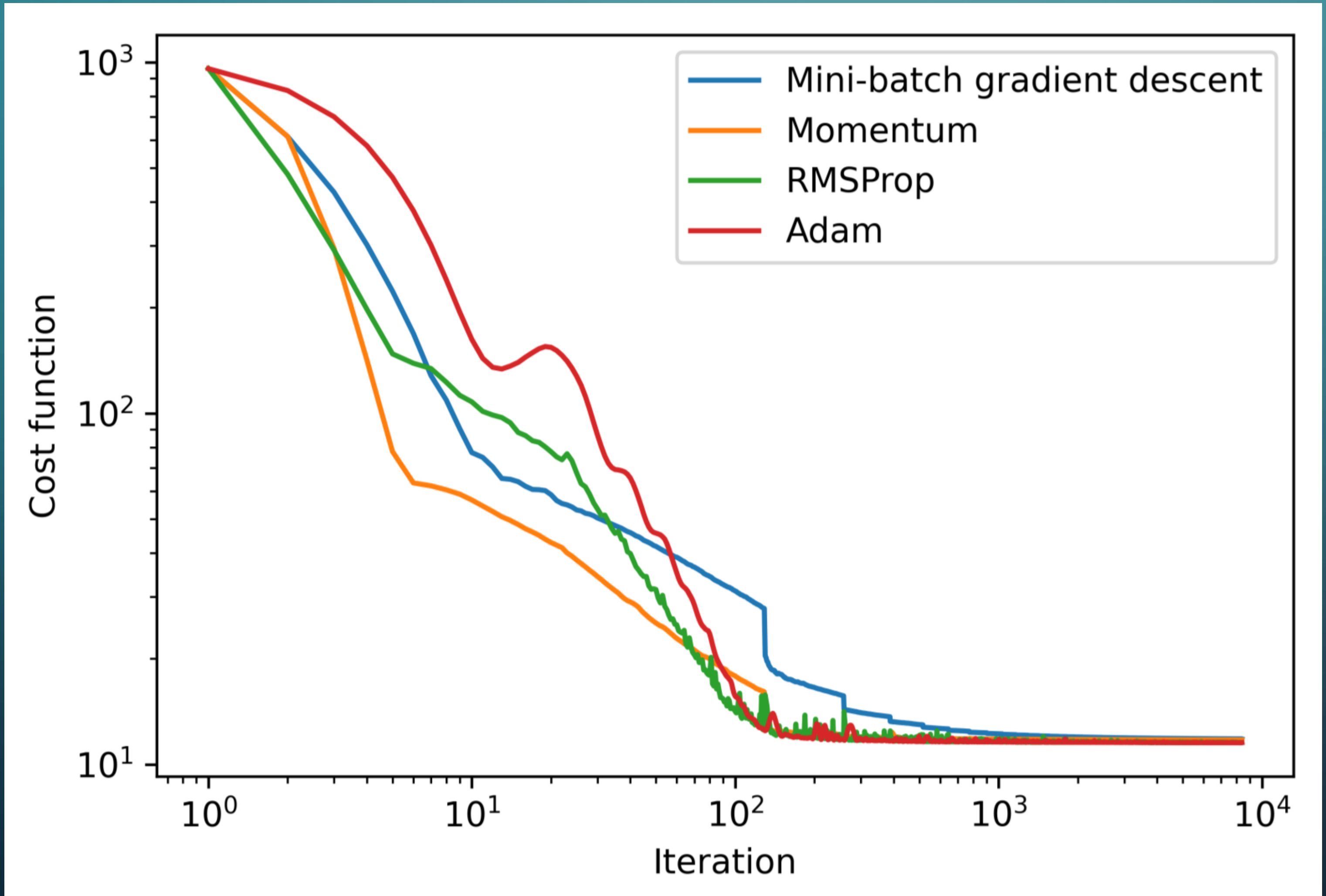
$$\theta_k^{i+1} = \theta_k^i - \frac{\eta}{\sqrt{\hat{v}_k^i} + \epsilon} \hat{m}_k^i$$

GRADIENT DESCENT & ADAM



Initial cost: 953.436
Mini-batch GD: 11.826
Adam: 11.535

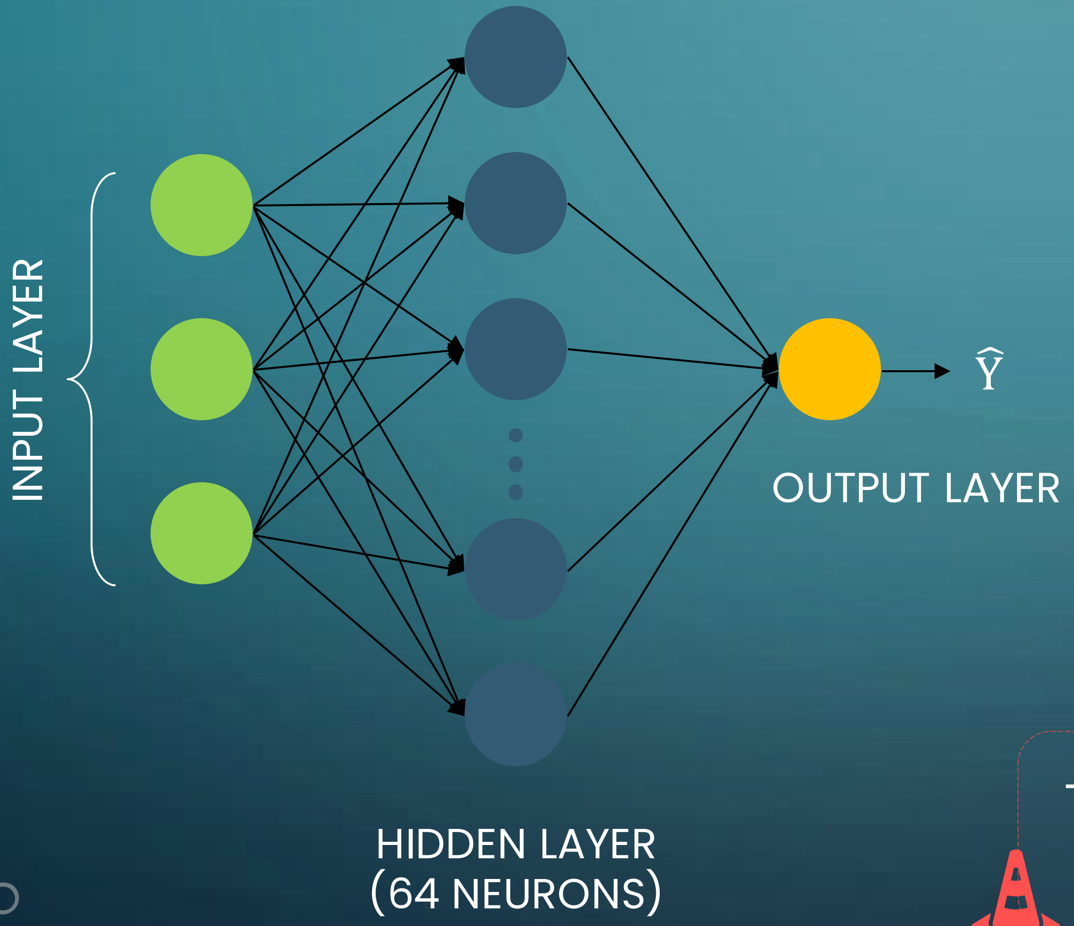
GRADIENT DESCENT OPTIMIZATIONS



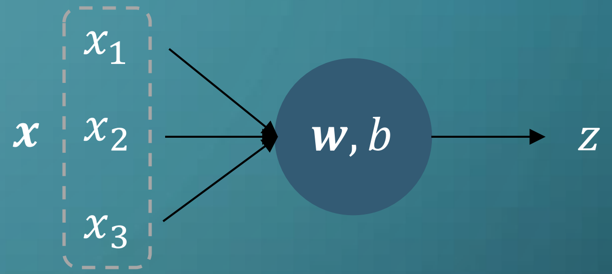
Initial cost: 953.436
Mini-batch GD: 11.826
Momentum: 11.740
RMSprop: 11.539
Adam: 11.535

A MORE COMPLEX MODEL

Shallow Neural Network (NN)



ReLU Activation (linear)

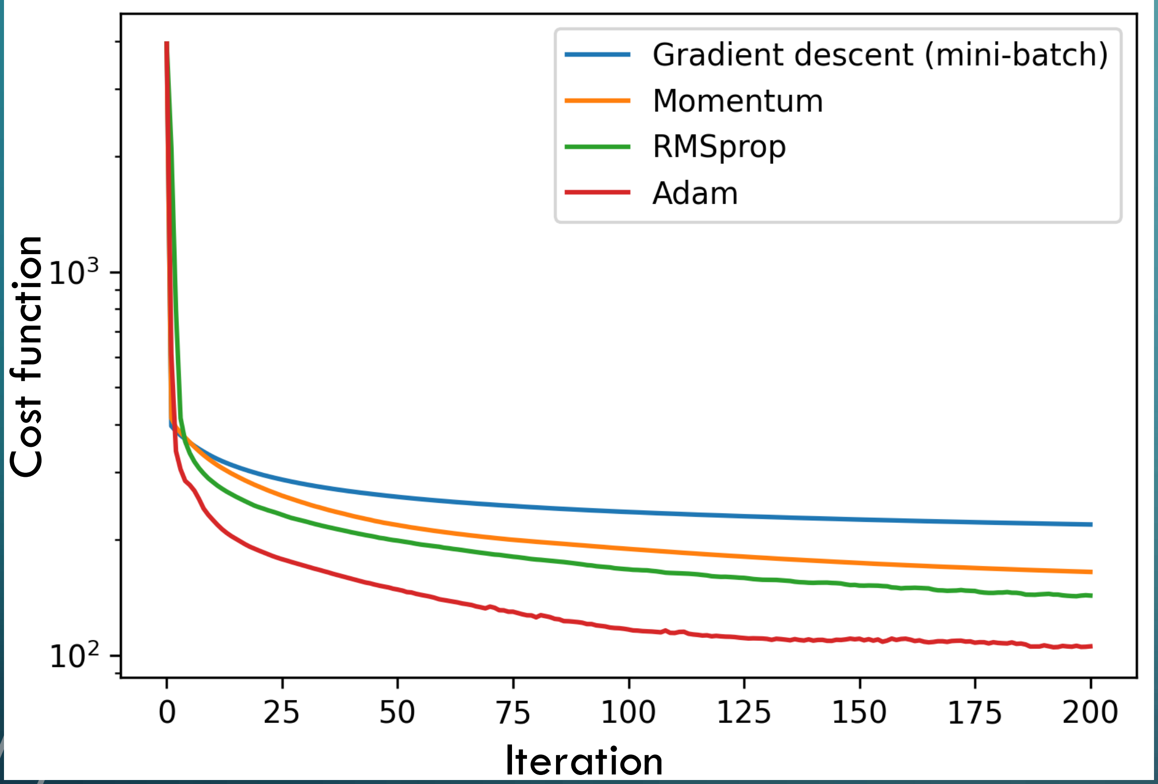


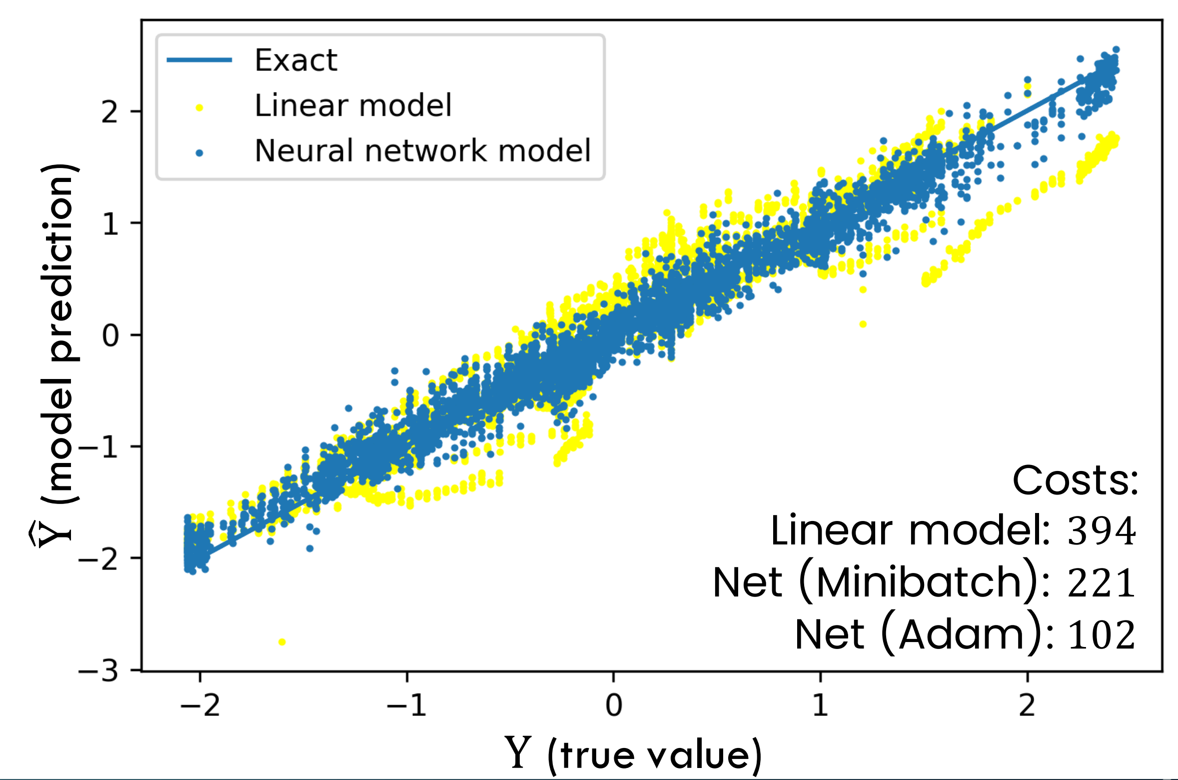
$$\tilde{z} = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$z = max(0, \tilde{z})$$

The cost function is now non-convex and with many local minima

SHALLOW NN: REGRESSION PERFORMANCE





- Adam achieves a better minimum than mini-batch GD
- NN model is better than linear model

CONCLUSIONS

GD variants: Batch, mini-batch & stochastic

GD optimizations: Momentum, RMSprop and ADAM

LINEAR REGRESSION MODEL (convex)

Momentum, RMSprop and Adam

↓

better convergence rate

NN REGRESSION MODEL (non-convex)

mini-batch GD with Adam optimizations

↓

better minimum
faster convergence rate