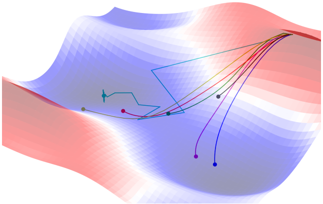


Optimization for deep learning

Vincent Barra
LIMOS, UMR 6158 CNRS, Université Clermont Auvergne

Introduction

Train a neural network $f_{\boldsymbol{w}}$ on a training set $Z \to$ define the optimal parameters $\boldsymbol{w}$ (weights, bias, filters).

What do we need to train ?

1. a loss function $\ell = \sum_{(\boldsymbol{x},y)\in Z} \mathcal{L}(f_{\boldsymbol{w}}(\boldsymbol{x}), y)$
2. an algorithm (backpropagation)
3. a way to perform a descent.

In deep learning the landscape of the loss function can exhibit a lot of local minima.

LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Optimizers - Gradient descent

Gradient descent:

$$\boldsymbol{g}_k = \frac{1}{|Z|} \sum_{(\boldsymbol{x},y)\in Z} \nabla_{\boldsymbol{w}} \mathcal{L}\left(f_{\boldsymbol{w_k}}(\boldsymbol{x}), y\right)$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \boldsymbol{g}_k$$

Observations

- "Real" gradient descent
- Computationaly expensive for large Z,
- Empirical estimation of the expected risk,
- Any partial sum is also an unbiased estimate of greater variance.

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

UNIVERSITÉ Clermont Auvergne

Optimizers - Gradient descent

Stochastic Gradient descent (SGD): Given an example $(\boldsymbol{x}, y) \in Z$:

$$\boldsymbol{g}_k = \nabla_{\boldsymbol{w}} \mathcal{L}\left(f_{\boldsymbol{w_k}}(\boldsymbol{x}), y\right)$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \boldsymbol{g}_k$$

Observations

- Helps evade local minima
- Sensitive to noise

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Optimizers - Gradient descent

mini-batch SGD: Given a batch of examples $(\boldsymbol{x}_i, y_i) \in Z, i \in [\![1 \cdots m]\!]$:

$$\boldsymbol{g}_k = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{w}} \mathcal{L}\left(f_{\boldsymbol{w_k}}(\boldsymbol{x}_i), y_i\right)$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \boldsymbol{g}_k$$

Observations

- Sequential or random sum
- The larger m, the lower the variance of the gradient estimates

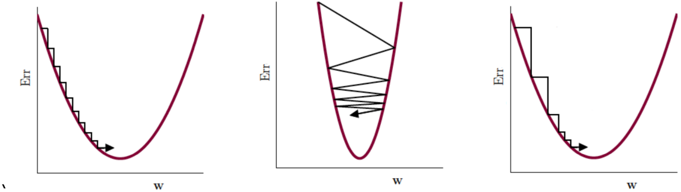
Result

Stochastic optimization algorithms like SGD yield the best generalization performance (in terms of excess error) despite being the worst optimization algorithms for minimizing the empirical risk (Bottou and Bousquet (2011)).

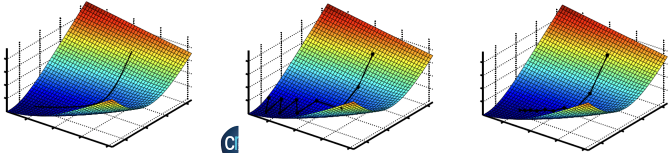
Learning rate

"The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyper parameter, tune the learning rate." (Goodfellow).

- η constant and too small: very slow gradient descent
- η constant and too large: erratic and oscillatory descent
- adaptative η as a function of the iteration



$\eta = 0.1$, 75 iterations $\eta = 2$, 10 iterations η_t, 10 iterations



Optimizers - Momentum

In valleys of the lanscape of the loss function (small but consistent gradients), the gradient descent is very slow.

Momentum:

$$\boldsymbol{u}_k = \alpha \boldsymbol{u}_{k-1} - \eta \boldsymbol{g}_k$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{u}_k$$

Observations

- $\boldsymbol{u}_k$: velocity = direction and speed by which the parameters move as the learning dynamics progresses
- Can jump local minima
- Accelerates if the gradient does not change much
- Dampens oscillations in narrow valleys
- α: how previous gradients affect the next descent (usually $\alpha \geq 0.8$).

Optimizers - Momentum

Nesterov acceleration:

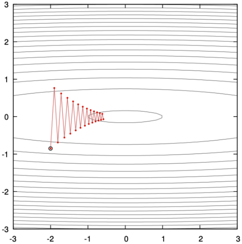
$$\boldsymbol{g}_k = \frac{1}{|Z|} \sum_{(\boldsymbol{x},y)\in Z} \nabla_{\boldsymbol{w}} \mathcal{L}\left(f_{\boldsymbol{w}_k + \alpha \boldsymbol{u}_{k-1}}(\boldsymbol{x}), y\right)$$

$$\boldsymbol{u}_k = \alpha \boldsymbol{u}_{k-1} - \eta \boldsymbol{g}_k$$

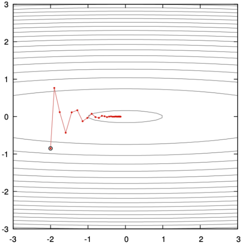
$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{u}_k$$

Observations

- Move in the direction of $\boldsymbol{u}_{k-1}$ and apply momentum.



$\eta = 5.10^{-2}, \mu = 0$



$\eta = 5.10^{-2}, \mu = 0.5$

Optimizers - Adaptive methods

Previous methods apply the same η to all parameters. AdaGrad uses an adaptive η, depending on the sparse property of parameters.

$$t_k = t_{k-1} + \boldsymbol{g}_k \odot \boldsymbol{g}_k$$
$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\eta}{\sqrt{t_k} + \epsilon} \odot \boldsymbol{g}_k$$

Observations

- $\eta \approx 0.01$, and automatic and adaptive computation of the learning rate
- Fine when the landscape is quasi convex
- t_k unboundedly increases during training $\rightarrow$ the update may become null.

Optimizers - Adaptive methods

RMSProp: to correct the unbounded behavior of t_k and adapt to non convex landscapes

$$t_k = \rho t_{k-1} + (1-\rho)\boldsymbol{g}_k \odot \boldsymbol{g}_k$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\eta}{\sqrt{t_k} + \epsilon} \odot \boldsymbol{g}_k$$

Adam: RMSProp + bias correction the the first two moments.

$$\boldsymbol{m}_k = \beta_1 \boldsymbol{m}_{k-1} + (1-\beta_1)\boldsymbol{g}_k$$

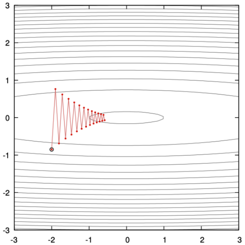
$$v_k = \beta_2 v_{k-1} + (1-\beta_2)\boldsymbol{g}_k \odot \boldsymbol{g}_k$$

$$b_k = \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k}$$

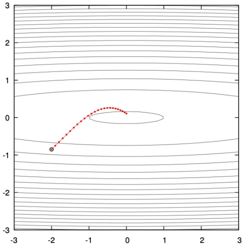
$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k b_k \frac{\boldsymbol{m}_k}{\sqrt{v_k} + \epsilon}$$

Comparisons

SGD/Adam comparison, strongly convex function.



$\eta = 5.10^{-2}, \mu = 0$



$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1.10^{-8}$

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

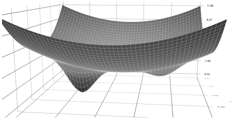
cnrs

UCA UNIVERSITÉ Clermont Auvergne

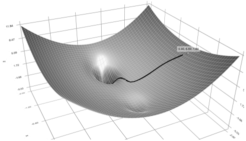
Comparisons

$$z = -2exp(-((x-1)^2 + y^2)/.2) - 6exp(-((x+1)^2 + y^2)/.2) + x^2 + y^2$$

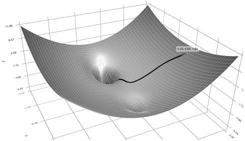
z has 2 minima. The number of iterations is indicated in brackets.



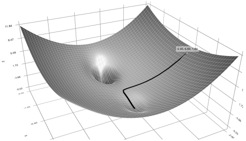
Surface $z = f(x, y)$



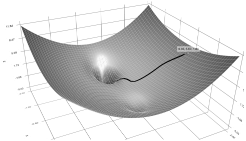
Adagrad (230)



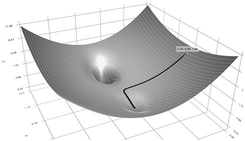
Adam (56)



Momentum (46)



RMSProp (70)



SGD (127)

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

UNIVERSITÉ Clermont Auvergne