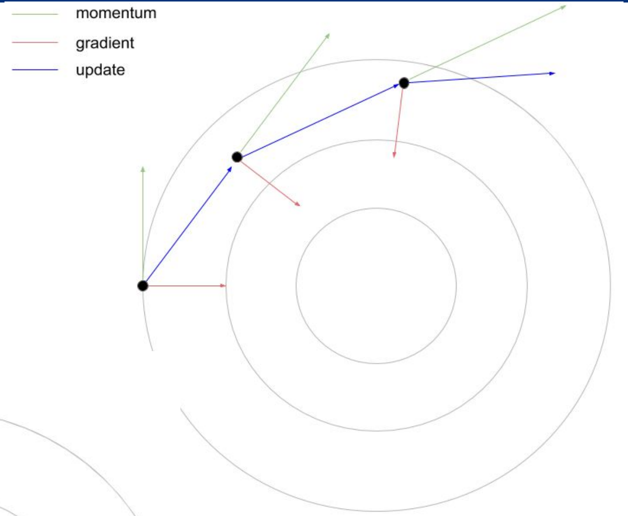
XNDL

Optimizers & regularization

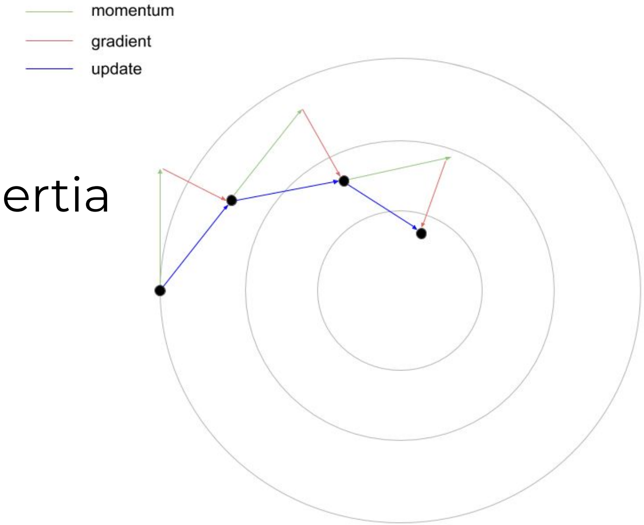
Dario Garcia Gasulla
dario.garcia@bsc.es

Inertia in Optimizers



Momentum:

- Add a fraction of the previous gradient. *Inertia*.
- Decaying weight parameter. *Friction*.
- Faster, smoother convergence



Nesterov:

- Gradient computed after inertia
- See the slope ahead
- Faster convergence

Adaptative LR Optimizers

Adagrad:

- Apply LR to parameter-wise gradients (adaptative)
- Considering all past updates
- High LR for infrequent ones. Low LR for frequent ones.
- Good for sparse data.

Issues:

- Requires initial global LR
- Vanishing LR (monotonically decreasing)

Adaptative LR Optimizers

Adadelta:

- Use effective LR instead (past param. update / current gradient)
- Set a max. window, implemented as a decay avg.
- Requires decay rate (0.9?)

Adam:

- Momentum (Decaying avg of past gradients, mean, beta1)
- Adadelta (Decaying avg of past squared gradients, variance, beta2)

Nadam:

- Adadelta + Nesterov

AMSGrad, AdaMax, AdamW, ...

Regularization

Norm based (L1, L2):

- ❖ Add penalties to the value of parameters or activations
- ❖ Layer-specific
- ❖ Added to loss

```
from tensorflow.keras import regularizers

layer = layers.Dense(
    units=64,
    kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
    bias_regularizer=regularizers.L2(1e-4),
    activity_regularizer=regularizers.L2(1e-5)
)
```

Optimizer Specific:

- ❖ Weight decay
- ❖ Gradient clipping (clipnorm, clipvalue)

```
tf.keras.optimizers.SGD(
    learning_rate=0.01,
    momentum=0.0,
    nesterov=False,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
```

Regularization

- Mean of activations approximates 0, deviation of activations apprx. 1
- Contain trainable params (depends on distribution!)

LayerNorm

- Transform activations sample-wise

```
tf.keras.layers.LayerNormalization(
```

BatchNorm

- Transform activations batch-wise.

```
tf.keras.layers.BatchNormalization(
```

Testing the environment

1. Download Fashion MNIST dataset:

https://github.com/zalandoresearch/fashion-mnist

2. Upload it to the cluster

scp *your_local_fashion_mnist.gz* nct...@dt01.bsc.es

3. Place them here: ~/.keras/datasets

create directories first!

Files

- lab7_code.py

  example of code for training a MLP on fashion dataset

- launcher.sh

  example of script for submitting a job to execute *code.py*

- Submit job:

  sbatch launcher.sh

Tasks

- Test different optimizers & hyperparameters
  - Convergence speed
- Try different normalization methods
  - Effect on overfitting
  - Convergence speed

Steps

- Test different optimizers & hyperparameters
  - Convergence speed

- Try different normalization methods
  - Effect on overfitting
  - Convergence speed

Baby steps

- Data nature, dimensionality, loss function, output layer + baseline
- Fix a batch size (stochasticity vs efficiency)

- Tune early stop, LR, activation funct., momentum while...
  - Start small **& underfit**
  - Grow **& overfit**
  - Regularize **& fit**

Dario Garcia-Gasulla (BSC)
dario.garcia@bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE