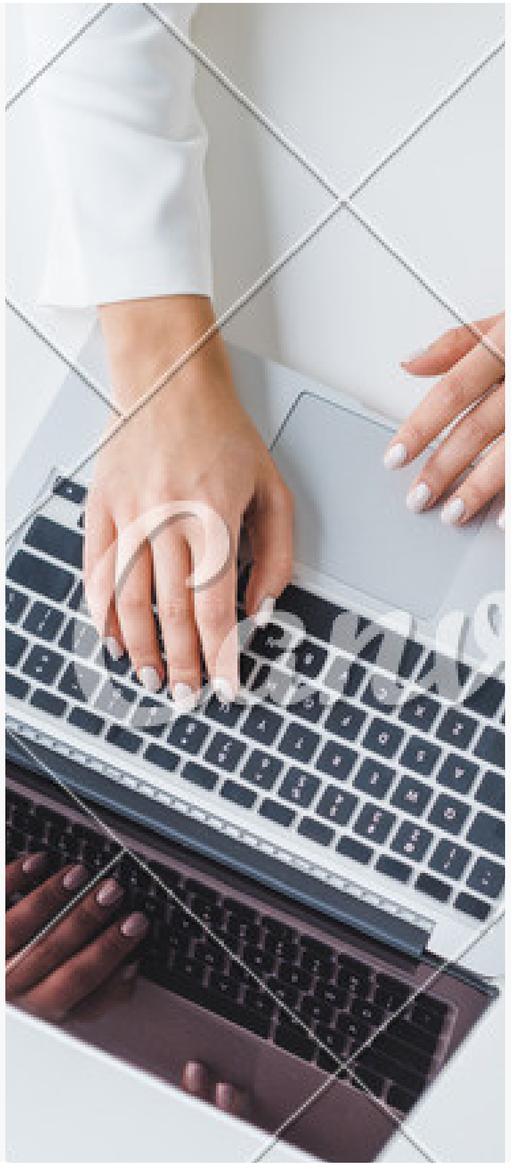
Optimizers in Artificial Neural Network Models

Instructor: Revendranath T

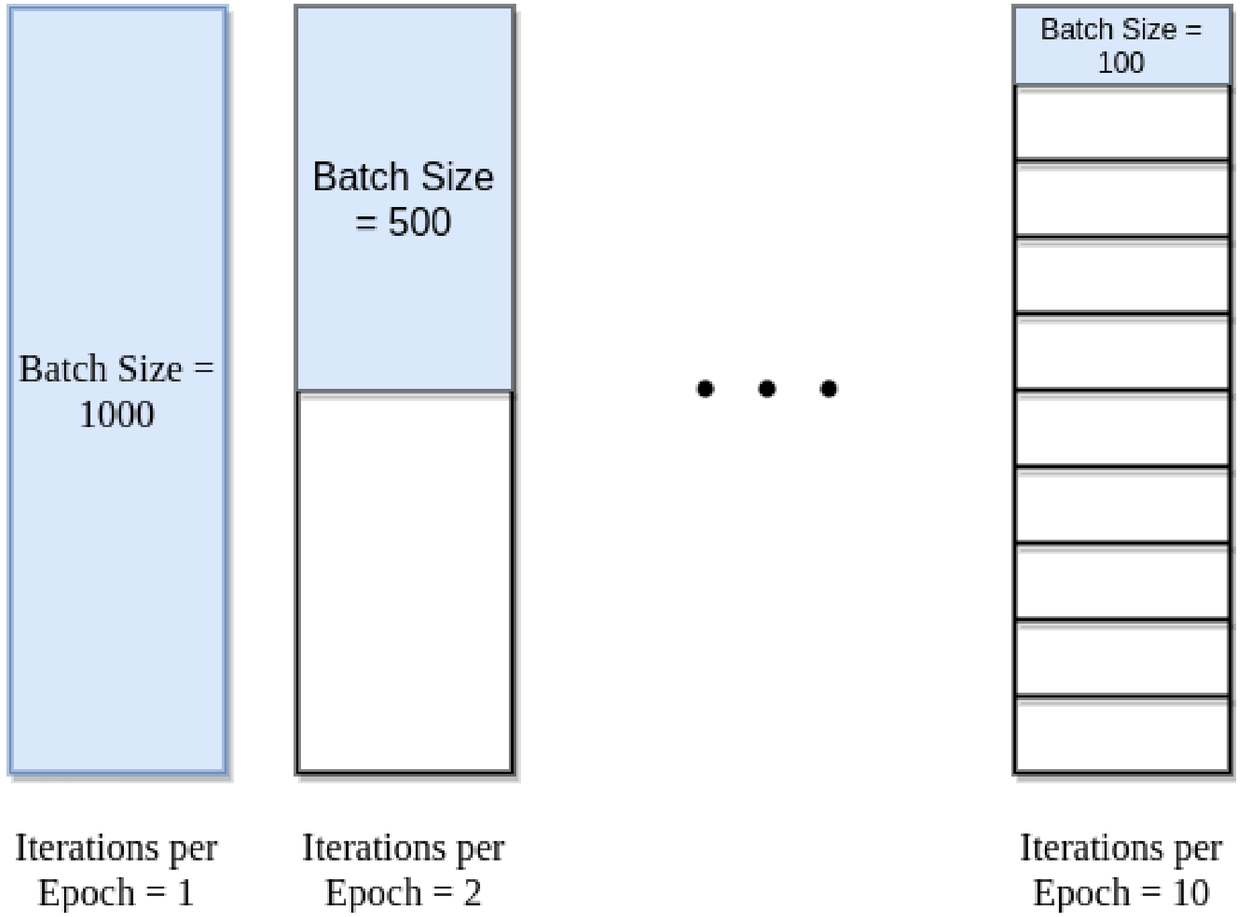


Goal of Optimizers

Make gradient descent run faster and avoid some of the problems that can cause it to get stuck

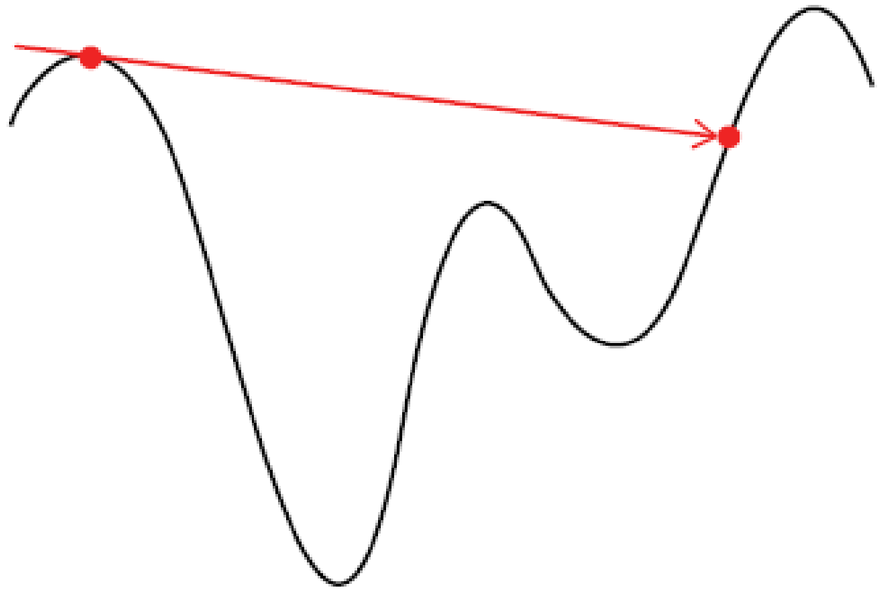
Automate activities

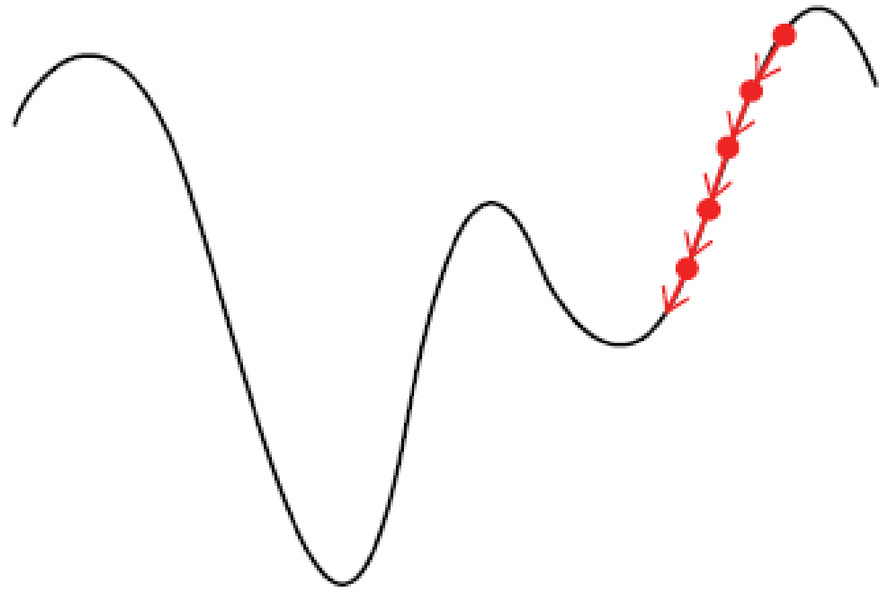
- to find the best learning rate
- to adjust learning rate automatically during training



Learning Rate (η)

- Values range 0.01 to 0.0001
- Smaller values of η (nearing 0 & positive) lead to slower learning and can find narrow valleys, but they can also get stuck in gentle valleys
- Larger values lead to faster learning, but they can lead us to miss valleys by jumping right over them.





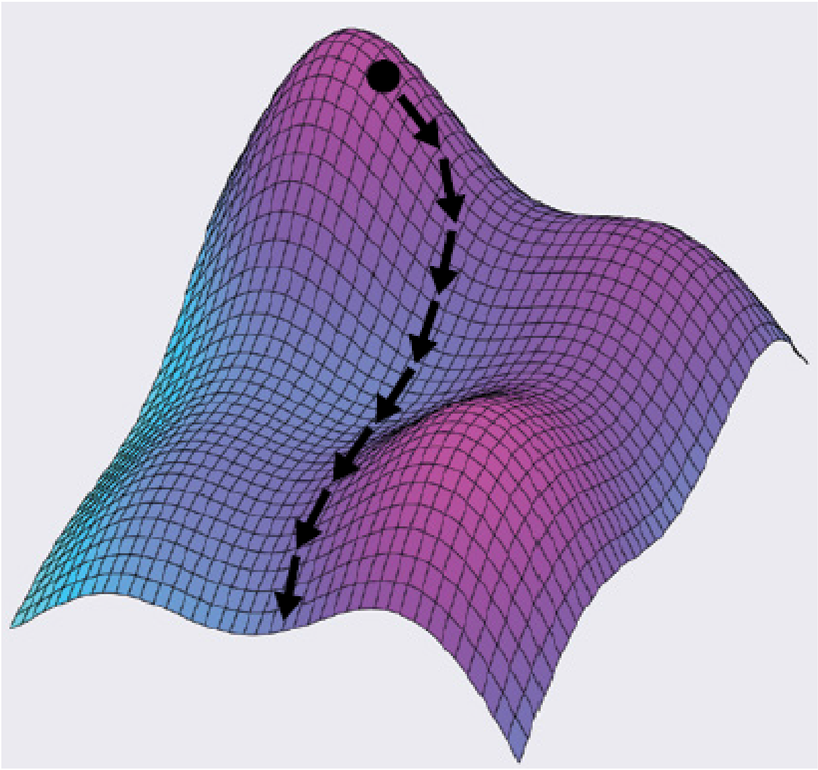
Optimizers

- The most popular algorithms:
  - momentum optimization,
  - Nesterov
  - Accelerated Gradient,
  - AdaGrad,
  - RMSProp, and
  - finally Adam and Nadam optimization.

Momentum Optimization

- Momentum optimizer will start out slowly
- It will quickly pick up momentum until it eventually reaches terminal velocity (if there is some friction or air resistance)
- Cares about the gradient values in the last iteration
- Due to the momentum, the optimizer may overshoot a bit, then come back, overshoot again, and oscillate like this many times before stabilizing at the minimum.
- It is good to have a bit of friction in the system to get rid of these oscillations and thus speeds up convergence.
- Implementation in keras: optimizer = **momentum**

Momentum Optimization



Momentum algorithm

1. $\mathbf{m} \leftarrow \beta\mathbf{m} - \eta\nabla_{\theta}J(\theta)$
2. $\theta \leftarrow \theta + \mathbf{m}$

Nesterov Optimization

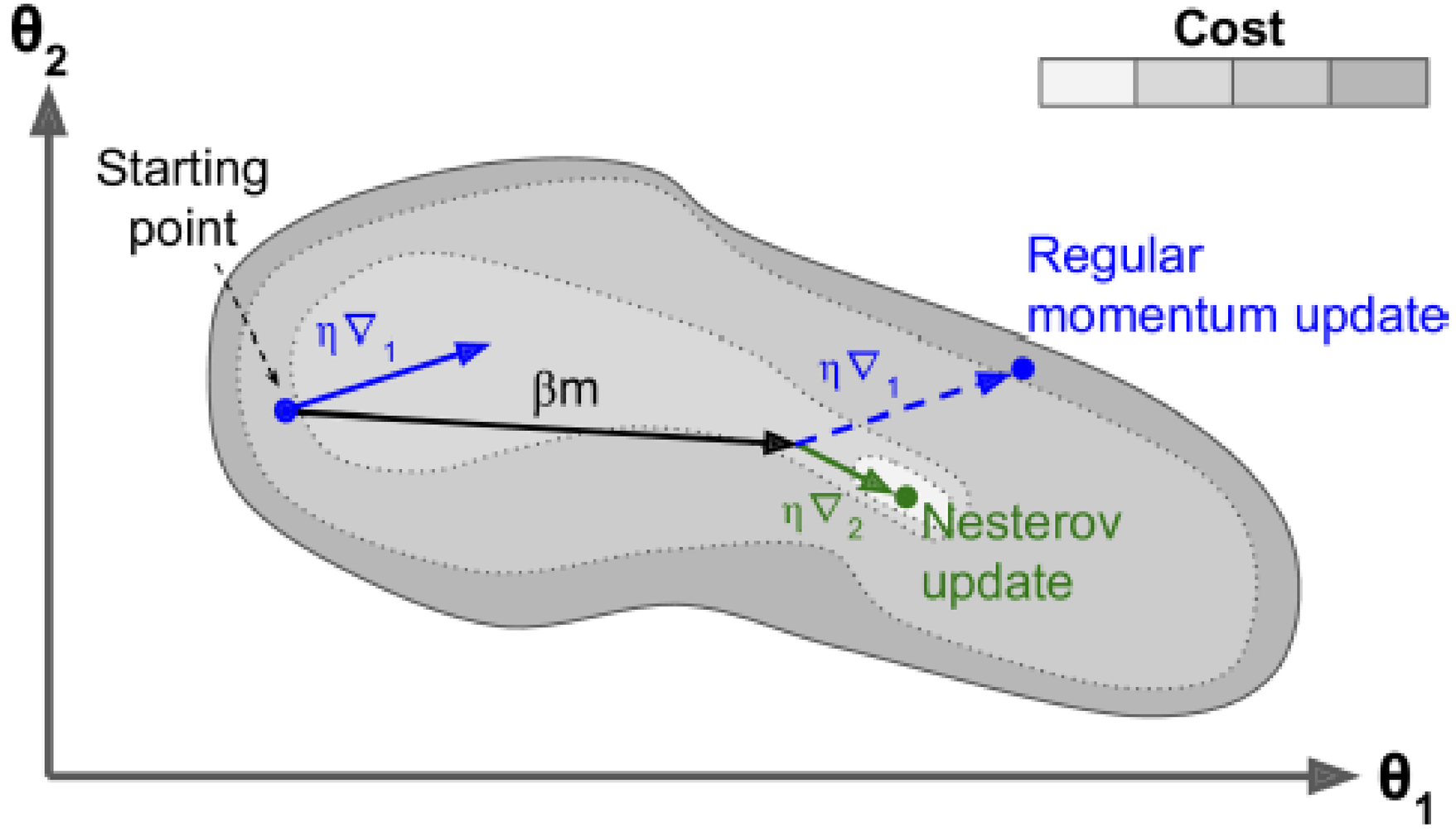
- Nesterov optimizer measures the gradient of the cost function, not at the local position, but slightly ahead
- This tweak points optimizer towards right direction
- NAG is generally faster than regular momentum optimization
- Implementation in keras:

optimizer = keras.optimizers.SGD(lr=0.001, momentum=0.9, nesterov=True)

Nesterov Accelerated Gradient algorithm

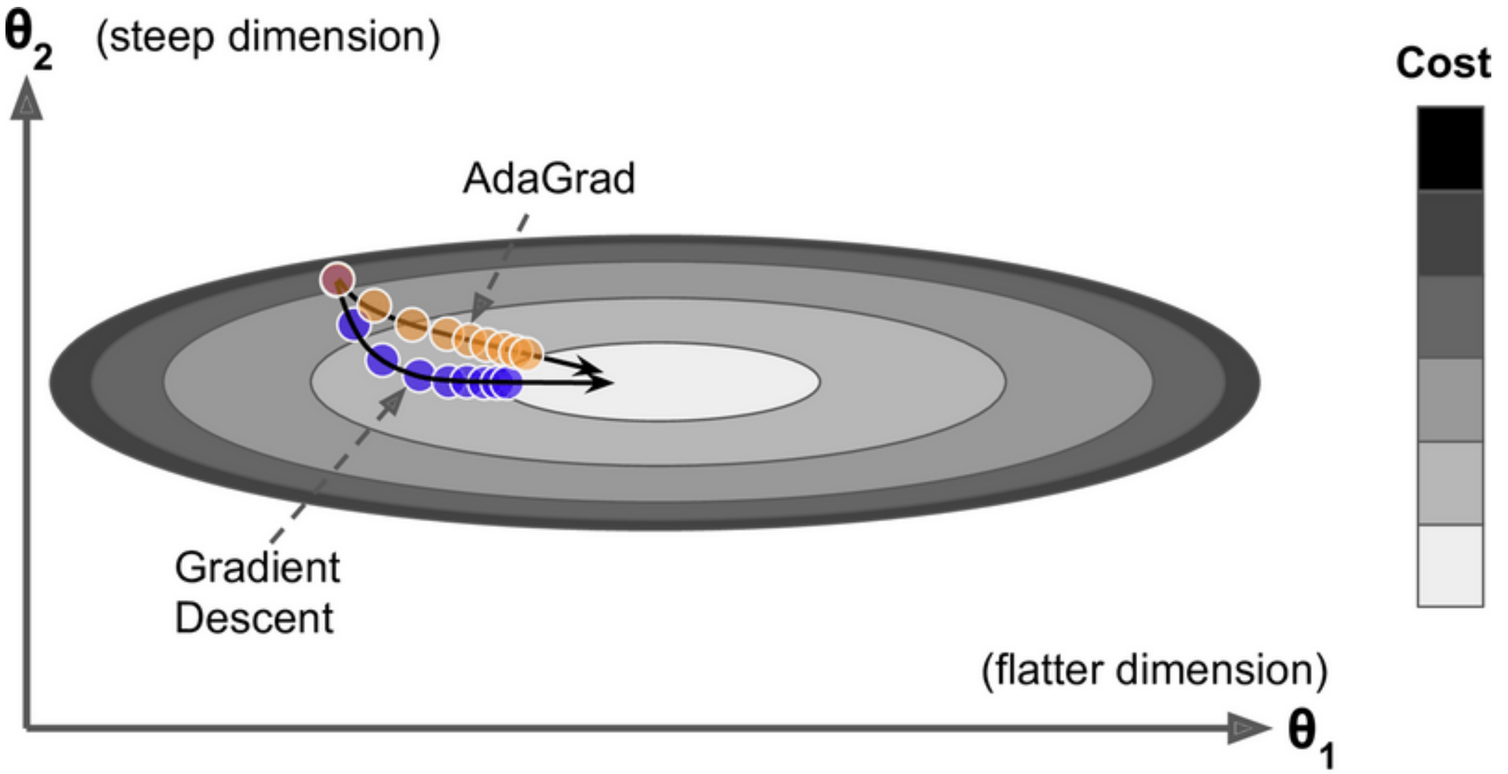
1. $\mathbf{m} \leftarrow \beta\mathbf{m} - \eta\nabla_{\theta}J(\theta + \beta\mathbf{m})$
2. $\theta \leftarrow \theta + \mathbf{m}$

Momentum vs Nesterov Optimization



AdaGrad

- Implements **adaptive learning rate**: decays the learning rate faster for steep dimensions than for dimensions with gentler slopes.
- Points the resulting updates more directly toward the global optimum.
- Requires much less tuning of the learning rate hyperparameter η.

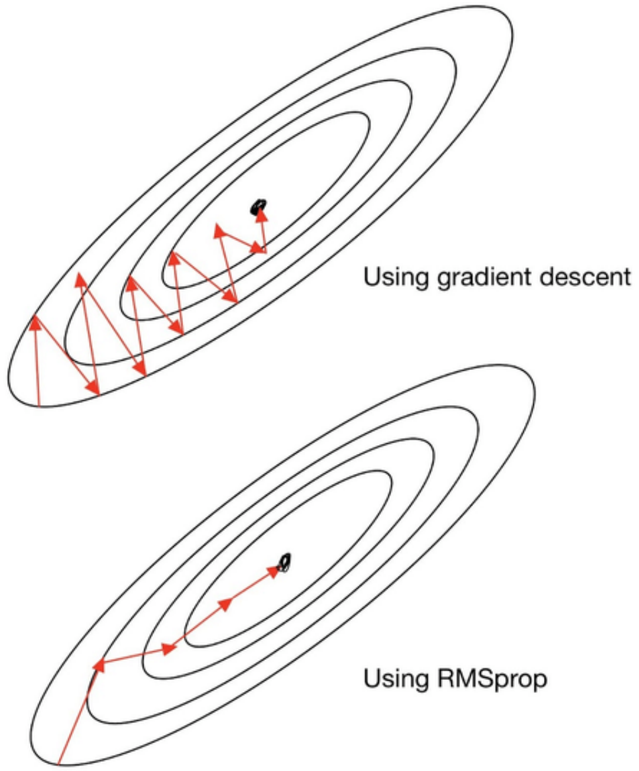


AdaGrad: Limitations

- AdaGrad frequently performs well for simple quadratic problems, but it often stops too early when training neural networks.
- The learning rate gets scaled down so much that the algorithm ends up stopping entirely before reaching the global optimum.
- Use Adagrad optimizer for for simpler tasks such as Linear Regression and should not use it to train deep neural networks .
- Understanding AdaGrad is helpful to grasp the other adaptive learning rate optimizers

RMS Prop Optimizer

- The RMSProp algorithm fixes the limitations of AdaGrad optimizer by accumulating only the gradients from the most recent iterations using exponential decay
- Performs much better than AdaGrad.
- Been a first choice optimization algorithm of many researchers until Adam optimization came around.
- Keras Implementation:




```
optimizer = keras.optimizers.RMSprop(lr=0.001, rho=0.9)
```

Adam Optimizer

- Adam: stands for adaptive moment estimation, combines the ideas of momentum optimization and RMSProp
- Just like momentum optimization, it keeps track of an exponentially decaying average of past gradients; and
- just like RMSProp, it keeps track of an exponentially decaying average of past squared gradients
- Use the default value $\eta = 0.001$
- Keras Implementation:

```
optimizer = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
```

Nadam Optimizer

- Nadam optimization is Adam optimization plus the Nesterov trick to converge faster,

Optimizers Comparison