Alberto Andrés Valdés González.
Degree: Mathematical Engineer.
Work position: Data Scientist.
Mail: anvaldes@uc.cl/alberto.valdes.gonzalez.96@gmail.com
Location: Santiago, Chile.

Gradient Descent and its variants

When we want to solve the next convex optimization problem:

$$\min_{\theta} J(\theta)$$

With $J(\cdot)$ a convex function, we use the gradient descent algorithm.

$$\boxed{\theta_{k+1} = \theta_k - \eta \cdot \nabla J(\theta_k)}$$

1. Batch Gradient Descent:

In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that's just one step of gradient descent in one epoch.

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function , data , params)
  params = params - learning_rate * params_grad
```

2. Stochastic Gradient Descent:

Stochastic gradient descent (SGD) performs a parameter update for each training example $x^{(i)}$ and $y^{(i)}$.

Steps:

i. Take an example.

ii. Calculate it's gradient.

iii. Use the gradient to update weights.

iv. Repeat.

$$\boxed{\theta_{k+1} = \theta_k - \eta \cdot \nabla J(\theta_k, x^{[i]}, y^{[i]})}$$

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

---

3. Mini-batch Gradient Descent:

Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

$$\boxed{\theta_{k+1} = \theta_k - \eta \cdot \nabla J(\theta_k, x^{[i\cdot n:(i+1)\cdot n]}, y^{[i\cdot n:(i+1)\cdot n]})}$$

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data, batch_size=50):
    params_grad = evaluate_gradient(loss_function, batch, params)
    params = params - learning_rate * params_grad
```

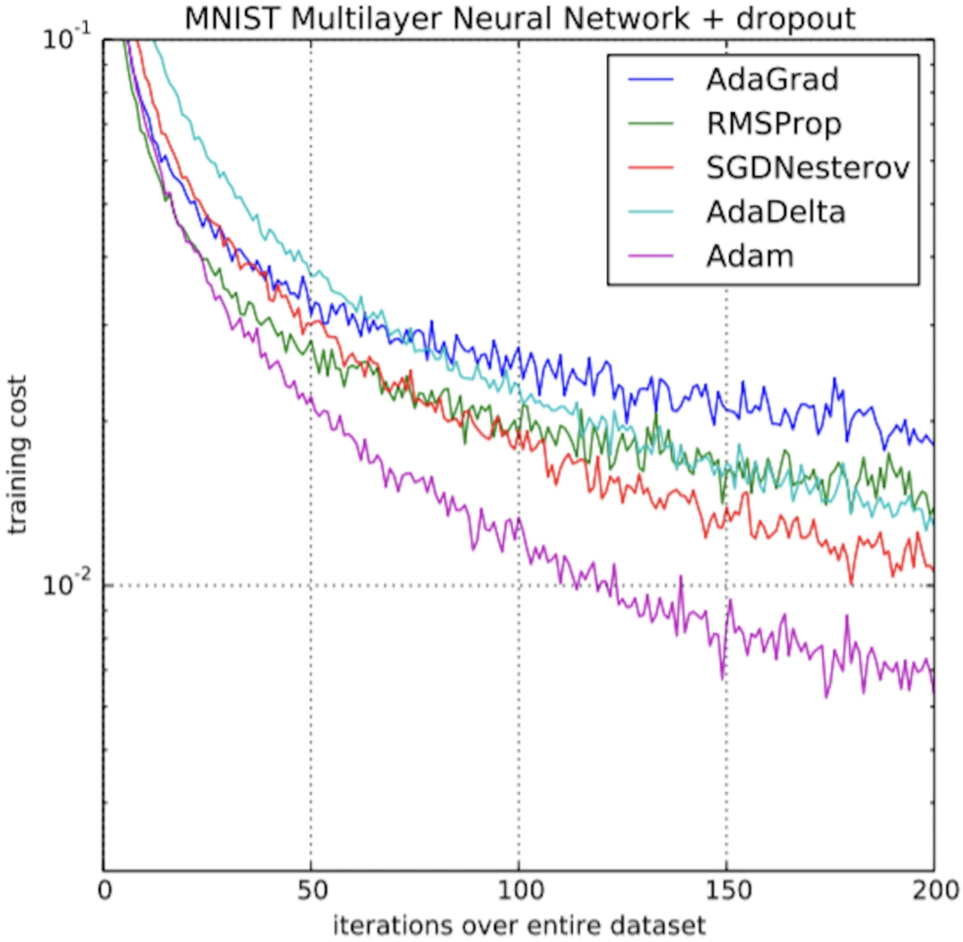
---

4. Other variants of gradient descent:

- Momentum.
- Nesterov.
- Adagrad.
- Adadelta.
- RMSprop.
- Adam.
- Adamax.
- AMSGrad.

Which optimizer choose?

Depends of the model.


MNIST Multilayer Neural Network + dropout
10^{-1}
10^{-2}
training cost
iterations over entire dataset
AdaGrad
RMSProp
SGDNesterov
AdaDelta
Adam