Learnig rate decay

- Goal: Faster optimization – slowly reduce learning rate
- 1 epoch = 1 pass through data

- Learning decay: $\alpha = \frac{1}{1+decay.rate * epoch.num} * \alpha_0$
- Exponentially decay: $\alpha = 0{,}95^{epoch.num} * \alpha_0$
- $\alpha = \frac{K}{\sqrt{epoch.num}} * \alpha_0$
- Staire decay: For each epoch divide α by 2.
- Manual decay control of dataset.

Some Problems

- Local optima or Saddle point.

- Problem of plateaus: Derivative is close to 0 for a long time - make learning slow.

Build Mini-batch gradient descent

- Size of mini-batches = 2^n (n is natural number)
- Shuffle (melanger aléatoirement) training set

Good setting for hyperparameters

- The α : learning rate – most important
- Second importance:
  - Adam optimization: $\beta_1 = 0{,}9\ and\ \beta_2 = 0{,}9\ and\ \varepsilon = 10^{-8}$
  - Mini batch size 2^n
  - # hidden units
- Third importance:
  - # layers
  - Learning rate decays

How to choose hyperparameters

- We have space search that includes multiple hyperparameters (>= 2 dimensions)
- Not use grid search
- Use random search in space search = random values for hyperparameters.
- Use coarse sample: Find the better point and resample close to this point.

Use appropriate scale to pick hyperparameters

- Random choice is not uniformally distributed
- To uniform the random choice, we can do it on log scale

- $r = -4 \ * np.random.rand \ (r \ \in [a, b])$
- $\alpha = 10^r$

- For exponentially weighted average
- $\beta = 1 - 10^r$

Organize hyperparameter search

- Pandas approach: Changing hyperparameter each day

- Caviar approach: If we have good power of calculus. Training many models in parallel.