Stochastic gradient descent

Victor Kitov

v.v.kitov@yandex.ru

Table of Contents

Gradient

- For any function $f(x)$, depending from $x = (x_1, ...x_D)^T$ gradient

$$\nabla f(x) := \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \cdots \\ \frac{\partial f(x)}{\partial x_D} \end{pmatrix}$$

- If function $f(x, y)$ depends on other variables y gradient ∇_x considers only derivatives with respect to x:

$$\nabla_x f(x, y) := \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \cdots \\ \frac{\partial f(x)}{\partial x_D} \end{pmatrix}$$

Directional derivative

Definition 1

Consider differentiable function $f : \mathbb{R}^D \to \mathbb{R}$. A derivative along direction d, $\|d\| = 1$ is defined as

$$f'(x, d) = \lim_{\lambda \to 0} \frac{f(x + \lambda d) - f(x)}{\lambda}$$

Theorem 2

$f'(x, d) = \nabla f(x)^T d$

Proof. Using 1-st order Taylor expansion we have

$$f(x + \lambda d) = f(x) + \nabla f(x)^T (\lambda d) + o(\lambda)$$

$$\frac{f(x + \lambda d) - f(x)}{\lambda} = \nabla f(x)^T d + o(1) \xrightarrow{\lambda \to 0} \nabla f(x)^T d$$

□

Direction of maximal growth/decrease

Theorem 3

For differentiable function $f(x)$ locally at point x:

- $\frac{\nabla f(x)}{\|\nabla f(x)\|}$ *is the direction of maximum growth*
- $-\frac{\nabla f(x)}{\|\nabla f(x)\|}$ *is the direction of maximal decrease.*

Proof. 1-st order Taylor expansion

$$f(x+\lambda d) = f(x) + \nabla f(x)^T(\lambda d) + o(\lambda)$$

From Cauchi-Schwartz inequality, taking $\|d\| = 1$:

$$\left|\nabla f(x)^T d\right| \leq \|\nabla f(x)\| \, \|d\| = \|\nabla f(x)\|$$

Equality is achieved when $d \propto \nabla f(x)$, i.e.
$d = \pm \nabla f(x) / \|\nabla f(x)\|$. $\square$

Table of Contents

Comments

- Empirical risk minimization

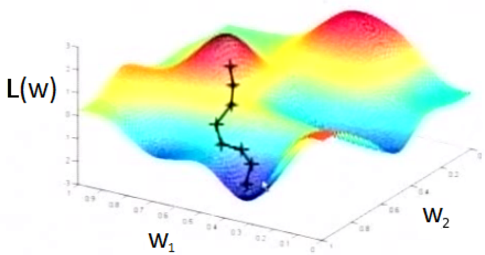
$$L(w) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(x_n, y_n, w) \to \min_w$$

- Regression: $\mathcal{L}(f_w(x_n) - y_n)$
- Classification $\mathcal{L}\left(\left(g_{y_n,w}(x_n) - g_{-y_n,w}(x_n)\right) y_n\right) = \mathcal{L}\left(g_w(x_n)y_n\right)$.
- Problems:
  - for general $\mathcal{L}, f(\cdot), g(\cdot)$ no analytical solution
  - $\widehat{\beta} = (X^TX)^{-1}X^TY$ - complexity $O(D^3)$ - high for big D.

Gradient descend optimization

- Gradient descend - iterative movement in steepest descent:

$$w := w - \nabla_w L(w)$$



- If $\mathcal{L}(u)$-convex => $L(w)$-convex => local optimum is global optimum.

Gradient descend optimization

INPUT:
* ε: parameter, controlling the speed of convergence
* stopping rule

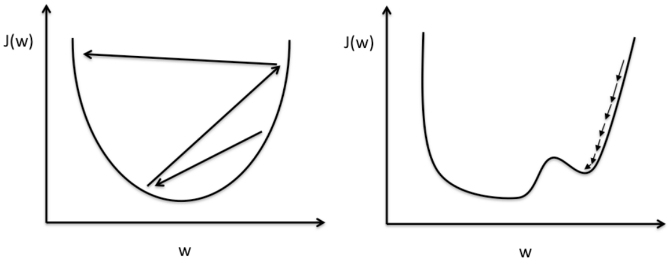
ALGORITHM:
initialize $t = 0$, w_0 randomly
WHILE stopping rule is not satisfied:
    $w_{t+1} := w_t - \varepsilon \nabla_w L(w_t)$
    $t := t + 1$

RETURN w_n

Stopping rules: $|L(w_t) - L(w_t)|$ or $\|w_t - w_{t-1}\|$ below threshold, or fixed #[iterations].

Learning rate selection[1]

ε should be selected carefully based on $L(w_t)$ dynamics.

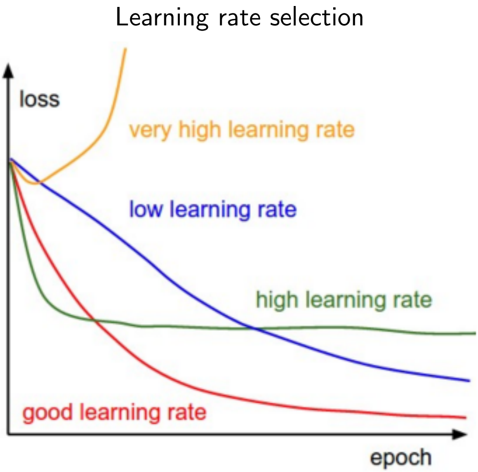


Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.

[1] Picture source.

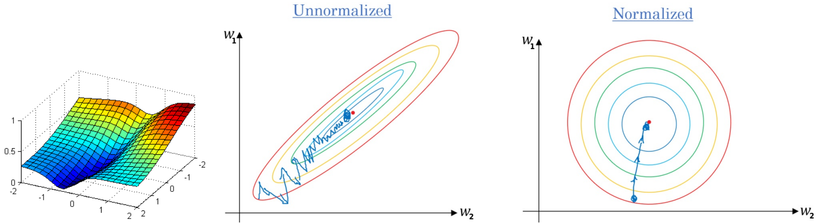
Learning rate selection



Feature normalization

Convergence is faster for normalized features:

- feature normalization solves the problem of «elongated valleys»



Problem of gradient descend (GD)

INPUT:
* ε_t: controls the speed of convergence
* stopping rule

ALGORITHM:
initialize $t = 0$, w_0 randomly
WHILE stopping rule is not satisfied:
$w_{t+1} := w_t - \varepsilon_t \frac{1}{N} \sum_{i=1}^{N} \nabla_w \mathcal{L}(x_i, y_i | w_n)$
$t := t + 1$

RETURN w_n

Gradient calculation requires $O(N)$ operations!

Stochastic gradient descent (SGD)

```
INPUT:
* ε_t: controls the speed of convergence
* stopping rule

ALGORITHM:
initialize t = 0, w_0 randomly
WHILE stopping rule is not satisfied:
    randomly sample I = {n_1, ...n_K} from {1, 2, ...N}
    w_{t+1} := w_t − ε_t (1/K) ∑_{n∈I} ∇_w L(x_n, y_n | w_t)
    t := t + 1

RETURN w_t
```

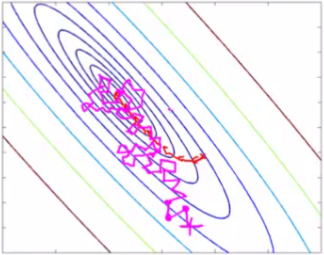
$$w_{t+1} := w_t - \varepsilon_t \frac{1}{K}\sum_{n\in I}\nabla_w \mathcal{L}(x_n, y_n|w_t)$$

Main idea: $\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(x_n, y_n|w) \approx \frac{1}{K}\sum_{n\in I}\mathcal{L}(x_n, y_n|w)$, one step takes $O(K)$, $K \ll N$, K-minibatch size.

SGD comments

- Indices generation: before each pass through the training set, it is randomly shuffled and then passed sequentially.
- Works even for $K = 1$.
- $\frac{1}{K}\sum_{i \in I} \nabla_w \mathcal{L}(x_i, y_i | w_n)$ can be computed in $O(1)$ for small K because processors internally perform vector arithmetics.

Convergence conditions



$$\sum_t \varepsilon_t = +\infty \qquad \text{SGD should reach any point}$$

$$\sum_t \varepsilon_t^2 < +\infty \qquad \varepsilon_t \text{ should converge to 0 fast}$$

In practice $\varepsilon_t = \frac{\alpha}{t+\beta}$ or constant which is reduced when criterion stops decreasing.

SGD reformulated

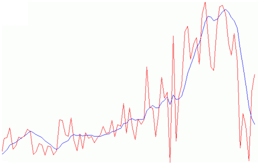
INPUT:
* ε_t: controls the speed of convergence
* stopping rule

ALGORITHM:
initialize $t = 0$, w_0 randomly, $\Delta w_0 = 0$
WHILE stopping rule is not satisfied:
    randomly sample $I = \{n_1, ... n_K\}$ from $\{1, 2, ... N\}$
    $\Delta w_{t+1} = -\frac{1}{K}\sum_{n\in I} \nabla_w \mathcal{L}(x_n, y_n | w_t)$
    $w_{t+1} := w_t + \varepsilon_t \Delta w_{t+1}$
    $t := t + 1$

RETURN w_n

Exponential smoothing

Example: original (red) and smoothed (blue) time series:



For series $z_1, ...z_N$ exponentially smoothed series is obtained by

$$\begin{cases} s_1 = z_1 & \alpha \in (0,1) \text{ - hyperparameter} \\ s_{n+1} = \alpha z_{n+1} + (1-\alpha)s_n & \text{recalculation takes } O(1) \end{cases}$$

SGD with momentum

INPUT:
∗ ε_t: controls the speed of convergence
∗ $\alpha \in (0,1]$: speed of direction change update
∗ stopping rule

ALGORITHM:
initialize $t=0$, w_0 randomly, $\Delta w_0 = 0$
WHILE stopping rule is not satisfied:
    randomly sample $I = \{n_1, ... n_K\}$ from $\{1, 2, ... N\}$
    $\Delta w_{t+1} = (1-\alpha)\Delta w_t + \alpha \frac{1}{K} \sum_{n \in I} \nabla_w \mathcal{L}(x_n, y_n | w_t)$
    $w_{t+1} := w_t + \varepsilon_t \Delta w_{t+1}$
    $t := t+1$

RETURN w_n

- Intuition: ↑speed by removing noisy gradients by aggregation over longer history.
- Typically $\alpha = 0.1$.

Other improvements

Other improvements of SGD exist:

- use 2nd order derivative
- Adam, RMSProp, AdaGrad, Adadelta
    - adjust ε_t for each dimension individually.
    - important dimensions get $\downarrow \varepsilon_t$
    - unimportant dimensions get $\uparrow \varepsilon_t$

Discussion of SGD

Advantages

- Simple
- Works online
- A small subset of learning objects may be sufficient for accurate estimation

Discussion of SGD

Advantages

- Simple
- Works online
- A small subset of learning objects may be sufficient for accurate estimation

Drawbacks

- Optimization using 2nd order derivatives converges faster.
- Needs selection of ε_t:
    - too big: divergence
    - too small: very slow convergence

- If $\mathcal{L}(\cdot)$ is convex => convergence to global min from any starting point.
- If $\mathcal{L}(\cdot)$ is non-convex => convergence to different local min, depending on starting point.

Table of Contents

L_1 regularization

- $||w||_1$ regularizer will do feature selection.
- Consider

$$\tilde{L}(w) = L(w) + \lambda \sum_{d=1}^{D} |w_d|$$

$$\frac{\partial \tilde{L}(w)}{\partial w_i} = \frac{\partial L(w)}{\partial w_i} + \lambda \, \mathrm{sign}\, w_i$$

$$\lambda \, \mathrm{sign}\, w_i \nrightarrow 0 \text{ when } w_i \to 0$$

- If $\lambda > \max_w \left| \frac{\partial L(w)}{\partial w_i} \right|$, then it becomes optimal to set $w_i = 0$
- For higher λ more weights become zero.

L_2 regularization

$$\tilde{L}(w) = L(w) + \lambda \sum_{d=1}^{D} w_d^2$$

$$\frac{\partial L(w)}{\partial w_i} = \frac{\partial L(w)}{\partial w_i} + 2\lambda w_i$$

$$2\lambda w_i \to 0 \text{ when } w_d \to 0$$

- Strength of regularization $\to 0$ as weights $\to 0$.
- So L_2 regularization will not set weights exactly to 0.

Summary

- Gradient descent iteratively optimizes $L(w)$ in the direction of maximum descent.
  - step takes $O(N)$
  - ε should be carefully chosen
- Stochastic gradient descent applies gradient descent to approximation of $L(w)$.
  - step takes $O(K)$
  - requires $\varepsilon_t \to 0$ for convergence.
- Feature normalization & momentum speeds up convergence.