# 随机梯度下降

刘新旺

https://xinwangliu.github.io/

国防科技大学 计算机学院

计算科学系人工智能与大数据教研室

2020 年 11 月 17 日

# Content

### 案例

回顾极值问题: 给定函数$f(k) = k^2 - 4k$, 求解该函数的极小值时，k的取值是多少?

## 案例

回顾极值问题: 给定函数$f(k) = k^2 - 4k$, 求解该函数的极小值时，k的取值是多少?

对$f(k)$求导, 然后令导数为0, 求解的$k$值即为所求:

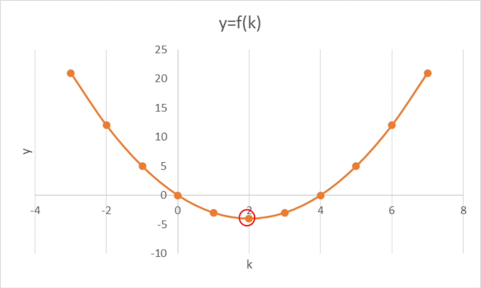
$\frac{\partial f(k)}{\partial k} = 2k - 4 = 0 \Longrightarrow k = 2$.

## 案例

回顾极值问题: 给定函数$f(k) = k^2 - 4k$, 求解该函数的极小值时，k的取值是多少?

对$f(k)$求导, 然后令导数为0, 求解的$k$值即为所求:

$\frac{\partial f(k)}{\partial k} = 2k - 4 = 0 \Longrightarrow k = 2.$

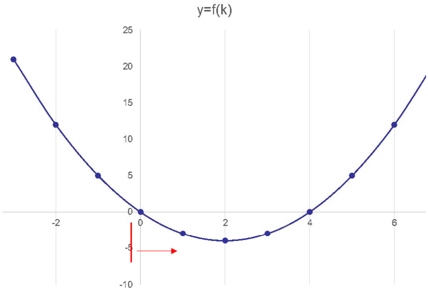


求导解法在复杂实际问题中很难计算. 迭代法通过从一个初始估计出发寻找一系列近似解来解决优化问题. 其基本形式如下:

$$k_{(t+1)} = k_t - \alpha * g(k_t), \tag{1}$$

其中$\alpha$被称为学习效率.

假设初始化$k_0 = 0$, 如何一步步迭代让$k_t$趋近最优解2?



要让$k_{t+1}$向最优值逼近, $g(k_t)$要满足两个条件:

- $g(k_t)$要能使$k_{t+1}$向最优解逼近(如何判定);
- 当$k_t$达到最优解时, $g(k_t)$要等于0.

当$k_t$达到最优解的时候, $k_{t+1}$要等于$k_t$, 即

$$k_{t+1} = k_t \Longrightarrow g(k_t) = 0. \tag{2}$$

核心问题: 即是寻找$g(k_t)$满足上述两个要求.

要让$k_{t+1}$向最优值逼近, $g(k_t)$要满足两个条件:

- $g(k_t)$要能使$k_{t+1}$向最优解逼近(如何判定);
- 当$k_t$达到最优解时, $g(k_t)$要等于0.

当$k_t$达到最优解的时候, $k_{t+1}$要等于$k_t$, 即

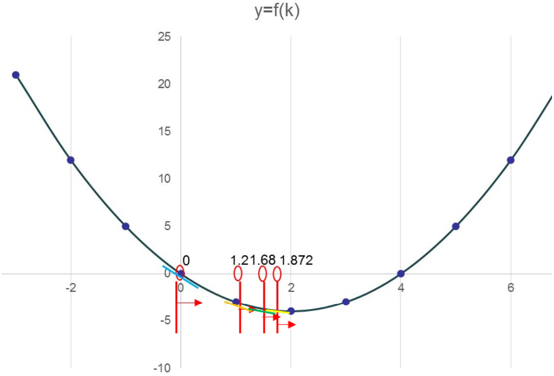
$$k_{t+1} = k_t \Longrightarrow g(k_t) = 0. \tag{2}$$

核心问题: 即是寻找$g(k_t)$满足上述两个要求.

函数$f(k)$的梯度满足上述两个要求!

令$g(k)$为$f(k)$的导数, 即$g(k)=2k-4$(设定学习速率为$\alpha=0.3$):

1. 当$k_0=0$: $k_1=k_0-0.3*g(k_0)=0-0.3*(-4)=1.2$
2. 当$k_1=1.2$: $k_2=k_1-0.3*g(k_1)=1.2-0.3*(2*1.2-4)=1.68$
3. 当$k_2=1.68$: $k_3=k_2-0.3*g(k_2)=1.68-0.3*(2*1.68-4)=1.872$
4. $\cdots$



- 随着迭代的不断进行, $g(k_t)$可以使$k_{t+1}$向最优值逼近. 而且, 当$k_{t+1}$离最优值越近时, $g(k_{t+1})$的绝对值越来越小. 当达到最优解时, $g(k_{t+1})$等于0.
- 学习速率$\alpha$的取值为什么是0.3?

> - 当$\alpha$取值较大时, 即梯度下降迭代的步长较大, 梯度下降迭代过程较快. 可以快速迭代到最优解附近, 但是可能一直在最优解附近徘徊,无法计算出最优解.
> - 当$\alpha$取值较小时, 即梯度下降迭代的步长较小, 梯度下降迭代过程较慢.

- 梯度优化：方向+步长

# 为什么需要随机梯度下降

- 基于梯度的优化在机器学习中已经被广泛应用
- 大规模梯度优化中的计算开销非常大，例如文本分类、自然语言处理等

当训练时间是瓶颈时，可以使用SGD.

**SGD的优点**

- 计算效率高.
- 容易实现.

# Content

## Gradient of a differentiable function

The gradient of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$, denoted as $\nabla f(\mathbf{w})$ is the vector of partial derivatives of $f$, namely,

$$\nabla f(\mathbf{w}) = \left( \frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \cdots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)^\top \tag{3}$$

Gradient descent is an iterative algorithm:

- Start with an initial value of $\mathbf{w}$ (say, $\mathbf{w}^1 = \mathbf{0}$);
- At each iteration, we take a step in the direction of the negative of the gradient at the current point, i.e.,

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla f(\mathbf{w}^{(t)}) \; (\eta > 0) \tag{4}$$

- Intuitively, the algorithm makes a small step in the opposite direction of the gradient points, thus decreasing the value of the function.
- After $T$ iterations, the algorithm outputs the last vector $\mathbf{w}^{(T)}$.
- The output could also be the averaged vector $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$. Taking the average turns out to be rather useful, especially when we generalize gradient descent to non-differentiable functions and to the stochastic case.

# 梯度下降算法的收敛速率

To analyze the convergence rate of the GD algorithm,

- we limit ourselves to the case of convex-Lipschitz functions.
- $\mathbf{w}^\star$ denote the minimizer of $f(\mathbf{w})$ with $\|\mathbf{w}^\star\| \leq B$.
- output $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$.
- bound $f(\hat{\mathbf{w}}) - f(\mathbf{w}^\star)$

To analyze the convergence rate of the GD algorithm,

- we limit ourselves to the case of convex-Lipschitz functions.
- $\mathbf{w}^\star$ denote the minimizer of $f(\mathbf{w})$ with $\|\mathbf{w}^\star\| \leq B$.
- output $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$.
- bound $f(\hat{\mathbf{w}}) - f(\mathbf{w}^\star)$

凸函数的性质：

$$
\begin{aligned}
f(\hat{\mathbf{w}}) - f(\mathbf{w}^\star) &\leq \frac{1}{T}\sum_{t=1}^{T}\left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star)\right) \\
f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star) &\leq \langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \nabla f(\mathbf{w}^{(t)})\rangle
\end{aligned} \tag{5}
$$

## Lemma 1

Let $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_T$ be an arbitrary sequence of vectors. Any algorithm with an initialization $\mathbf{w}^{(1)} = 0$ and an update rule of the form

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \mathbf{v}_t \ (\eta > 0) \tag{6}$$

satisfies

$$\sum_{t=1}^{T} \langle \mathbf{w}^t - \mathbf{w}^\star, \mathbf{v}_t \rangle \leq \frac{\|\mathbf{w}^\star\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|\mathbf{v}_t\|^2. \tag{7}$$

In particular, for every $B, \rho > 0$, if for all $t$ we have that $\|\mathbf{v}_t\| \leq \rho$ and if we set $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$, then for every $\mathbf{w}^\star$ with $\|\mathbf{w}^\star\| \leq B$, we have

$$\frac{1}{T} \sum_{t=1}^{T} \langle \mathbf{w}^t - \mathbf{w}^\star, \mathbf{v}_t \rangle \leq \frac{B\rho}{\sqrt{T}}. \tag{8}$$

By letting $\mathbf{v}_t = \nabla f(\mathbf{w}^{(t)})$ and apply the Lemma 1, we have the following corollary.

COROLLARY 14.2 *Let $f$ be a convex, $\rho$-Lipschitz function, and let* $\mathbf{w}^\star \in \operatorname{argmin}_{\{\mathbf{w}:\|\mathbf{w}\|\le B\}} f(\mathbf{w})$. *If we run the GD algorithm on $f$ for $T$ steps with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$, then the output vector $\bar{\mathbf{w}}$ satisfies*

$$f(\bar{\mathbf{w}}) - f(\mathbf{w}^\star) \ \le\ \frac{B\,\rho}{\sqrt{T}}.$$

*Furthermore, for every $\epsilon > 0$, to achieve $f(\bar{\mathbf{w}}) - f(\mathbf{w}^\star) \le \epsilon$, it suffices to run the GD algorithm for a number of iterations that satisfies*

$$T \ge \frac{B^2\rho^2}{\epsilon^2}.$$

Note that if $f$ is $\rho$-Lipschitz, then $\|\nabla f(\mathbf{w}^{(t)})\| \le \rho$.

有函数$f(x)$, 如果存在一个常量$\rho$, 使得对$f(x)$定义域上（可为实数也可以为复数）的任意两个值满足如下条件:

$|f(x_1) - f(x_2)| \leq |x_1 - x_2| * \rho$

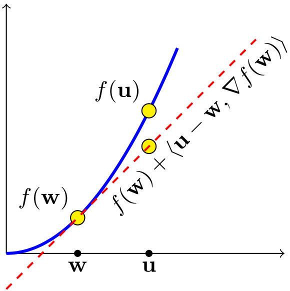
则称函数$f(x)$满足Lipschitz连续条件, 并称$\rho$为$f(x)$的Lipschitz 常数.

Lipschitz连续限制了函数的局部变动幅度不能超过某常量.

# Content

- The GD algorithm requires that the function $f$ be differentiable.
- The GD algorithm can be applied to non-differentiable functions by using a so-called subgradient(次梯度) of $f(\mathbf{w})$ at $\mathbf{w}^{(t)}$, instead of the gradient.

- For a convex function $f$, the gradient at $\mathbf{w}$ defines the slope of a tangent that lies below $f$




$$\forall\, \mathbf{u}, f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$$

## Definition of Subgradient

Let $\mathcal{S}$ be an open convex set. A function $f : \mathcal{S} \to \mathbb{R}$ is a convex function. A vector $\mathbf{v}$ that satisfies

$$\forall\, \mathbf{u}, f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \mathbf{v} \rangle \tag{9}$$

is called a subgradient of $f$ at $\mathbf{w}$. The set of subgradients of $f$ at $\mathbf{w}$ is called the differential set and denoted $\partial f(\mathbf{w})$.

How do we construct subgradients of a given convex function?

**Claim 1**

If $f$ is differentiable at $\mathbf{w}$ then $\partial f(\mathbf{w})$ contains a single element—the gradient of $f$ at $\mathbf{w}$, $\nabla f(\mathbf{w})$.

Example: The subgradient of $|x|$.

How do we construct subgradients of a given convex function?

**Claim 1**

If $f$ is differentiable at $\mathbf{w}$ then $\partial f(\mathbf{w})$ contains a single element—the gradient of $f$ at $\mathbf{w}$, $\nabla f(\mathbf{w})$.

Example: The subgradient of $|x|$.

**Claim 2**

Let $g(\mathbf{w}) = \max_{1\le i\le r} g_i(\mathbf{w})$ for $r$ convex differentiable functions $g_1, \cdots, g_r$. Given some $\mathbf{w}$, let $j \in \arg\max_{1\le i\le r} g_i(\mathbf{w})$, Then $\nabla g_j(\mathbf{w}) \in \partial g(\mathbf{w})$.

Example: A subgradient of Hinge loss $f(\mathbf{w}) = \max\{0, 1 - y\langle \mathbf{w}, x\rangle\}$.

- The gradient descent algorithm can be generalized to non-differentiable functions by using a subgradient of $f(\mathbf{w})$ at $\mathbf{w}^{(t)}$, instead of the gradient.
- The analysis of the convergence rate remains unchanged: Simply note that the following equation

$$f(\mathbf{w}^{(t)}) - f(\mathbf{w}^{\star}) \ \leq \langle \mathbf{w}^{(t)} - \mathbf{w}^{\star}, \partial f(\mathbf{w}^{(t)}) \rangle \tag{10}$$

true for subgradients as well.

# Main Idea

- In stochastic gradient descent, we do not require the update direction to be based exactly on the gradient.
- Instead, we allow the direction to be a **random vector** and require that its expected value will be a subgradient of the function at the current vector.

- In stochastic gradient descent, we do not require the update direction to be based exactly on the gradient.
- Instead, we allow the direction to be a **random vector** and require that its expected value will be a subgradient of the function at the current vector

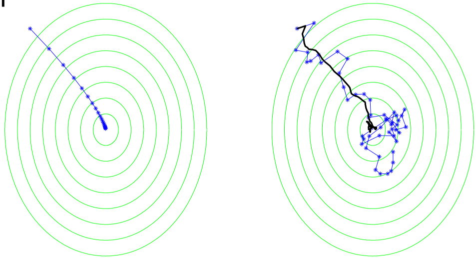


**Figure 14.3** An illustration of the gradient descent algorithm (left) and the stochastic gradient descent algorithm (right). The function to be minimized is $1.25(x+6)^2 + (y-8)^2$. For the stochastic case, the black line depicts the averaged value of $\mathbf{w}$.

```
Stochastic Gradient Descent (SGD) for minimizing
```

$$f(\mathbf{w})$$

**parameters:** Scalar $\eta > 0$, integer $T > 0$
**initialize:** $\mathbf{w}^{(1)} = \mathbf{0}$
**for** $t = 1, 2, \ldots, T$
  choose $\mathbf{v}_t$ at random from a distribution such that $\mathbb{E}[\mathbf{v}_t \mid \mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$
  update $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$
**output** $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}^{(t)}$

- In the context of learning problems, it is easy to find a random vector whose expectation is a subgradient of the risk function.
- For example, the gradient of the risk function at each sample.

### Objective function of soft-SVM

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \xi_i \quad s.t. \quad y_i\left(\mathbf{w}^\top \mathbf{x}_i\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i \tag{11}$$

To apply SGD, we have to transform the optimization problem in Eq.(14) into a unconstricted one.

### Equivalent formulation

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max\left\{0,\ 1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i\right)\right\} \tag{12}$$

## Objective function of soft-SVM

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\xi_i \quad s.t. \quad y_i\left(\mathbf{w}^\top \mathbf{x}_i\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i \qquad (11)$$

To apply SGD, we have to transform the optimization problem in Eq.(14) into a unconstricted one.

## Equivalent formulation

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\max\left\{0,\ 1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i\right)\right\} \qquad (12)$$

- How to find a random vector whose expectation is a subgradient of the risk function?

One subgradient of Eq.(12) is

$$\lambda \mathbf{w}^{(t)} + \mathbf{v}_t, \tag{13}$$

with $\mathbf{v}_t$ a subgradient of the (hinge) loss function at $\mathbf{w}^{(t)}$ on the random example chosen at iteration $t$.

**Updating Rules**

$$\begin{aligned} \mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \frac{1}{\lambda\, t}\left(\lambda \mathbf{w}^{(t)} + \mathbf{v}_t\right) \\ &= \frac{t-1}{t}\mathbf{w}^{(t)} - \frac{1}{\lambda\, t}\mathbf{v}_t \\ &= \frac{t-1}{t}\left(\frac{t-2}{t-1}\mathbf{w}^{(t-1)} - \frac{1}{\lambda\,(t-1)}\mathbf{v}_{t-1}\right) - \frac{1}{\lambda\, t}\mathbf{v}_t \\ &= -\frac{1}{\lambda\, t}\sum_{i=1}^{t}\mathbf{v}_i \end{aligned} \tag{14}$$

# Algorithm: Implementing SVM with SGD

**SGD for Solving Soft-SVM**

**goal:** Solve Equation (15.12)
**parameter:** $T$
**initialize:** $\boldsymbol{\theta}^{(1)} = \mathbf{0}$
**for** $t = 1, \ldots, T$
  Let $\mathbf{w}^{(t)} = \frac{1}{\lambda\, t}\boldsymbol{\theta}^{(t)}$
  Choose $i$ uniformly at random from $[m]$
  If $(y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle < 1)$
    Set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + y_i \mathbf{x}_i$
  Else
    Set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)}$
**output:** $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}^{(t)}$

where $\boldsymbol{\theta}^{(t)} = -\frac{1}{\lambda t}\sum_{i=1}^{t} \mathbf{v}_i$.

THEOREM 14.8 *Let $B, \rho > 0$. Let $f$ be a convex function and let $\mathbf{w}^\star \in \operatorname{argmin}_{\mathbf{w}:\|\mathbf{w}\| \le B} f(\mathbf{w})$. Assume that SGD is run for $T$ iterations with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$. Assume also that for all $t$, $\|\mathbf{v}_t\| \le \rho$ with probability $1$. Then,*

$$\mathbb{E}\left[f(\bar{\mathbf{w}})\right] - f(\mathbf{w}^\star) \le \frac{B\,\rho}{\sqrt{T}}.$$

*Therefore, for any $\epsilon > 0$, to achieve $\mathbb{E}[f(\bar{\mathbf{w}})] - f(\mathbf{w}^\star) \le \epsilon$, it suffices to run the SGD algorithm for a number of iterations that satisfies*

$$T \ge \frac{B^2 \rho^2}{\epsilon^2}.$$

Only Need to show that

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{i=1}^{T}\left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star)\right)\right] \le \mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t\rangle\right] \tag{15}$$

In the previous analyses of the GD and SGD algorithms, we required that the norm of $\mathbf{w}^\star$ will be at most $B$, which is equivalent to requiring that $\mathbf{w}^\star$ is in the set $\mathcal{H} = \{\mathbf{w} : \|\mathbf{w}\| \leq B\}$.

$$1..\ \ \mathbf{w}^{(t+\frac{1}{2})} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$$
$$2..\ \ \mathbf{w}^{(t+1)} = \operatorname{argmin}_{\mathbf{w}\in\mathcal{H}} \|\mathbf{w} - \mathbf{w}^{(t+\frac{1}{2})}\|$$

The projection step replaces the current value of $\mathbf{w}$ by the vector in $\mathcal{H}$ closest to it.

- Another variant of SGD is decreasing the step size as a function of $t$. For instance, we can set $\eta_t = \frac{B}{\rho t}$. The idea is that when we are closer to the minimum of the function, we take our steps more carefully.
- Other Averaging Techniques

# Content

- Gradient Descent
- Subgradient
- Stochastic Gradient Descent (SGD)

Next: Multiclass, Ranking and Complex Predication Problems