CS 6316 Machine Learning

Gradient Descent

Yangfeng Ji

Information and Language Processing Lab
Department of Computer Science
University of Virginia

Overview

1. Gradient Descent
2. Stochastic Gradient Descent
3. SGD with Momentum
4. Adaptive Learning Rates

Gradient Descent

Learning as Optimization

As discussed before, learning can be viewed as optimization problem.

- Training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$
- Empirical risk

$$L(h_{\boldsymbol{\theta}}, S) = \frac{1}{m} \sum_{i=1}^{m} R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \tag{1}$$

  where R is the risk function

- Learning: minimize the empirical risk

$$\boldsymbol{\theta} \leftarrow \operatorname*{argmin}_{\boldsymbol{\theta}'} L_S(h_{\boldsymbol{\theta}'}, S) \tag{2}$$

Learning as Optimization (II)

Some examples of risk functions

- Logistic regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = -\log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \tag{3}$$

Learning as Optimization (II)

Some examples of risk functions

- Logistic regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = -\log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \tag{3}$$

- Linear regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \|h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\|_2^2 \tag{4}$$

Learning as Optimization (II)

Some examples of risk functions

- Logistic regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = -\log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \tag{3}$$

- Linear regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \|h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\|_2^2 \tag{4}$$

- Neural network

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \text{Cross-entropy}(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \tag{5}$$

Learning as Optimization (II)

Some examples of risk functions

- Logistic regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = -\log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \tag{3}$$

- Linear regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \|h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\|_2^2 \tag{4}$$

- Neural network

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \text{Cross-entropy}(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \tag{5}$$

- Percetpron and AdaBoost can also be viewed as minimizing certain loss functions

Constrained Optimization

The dual optimization problem for SVMs of the separable cases is

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{6}$$

$$\text{s.t.} \quad \alpha_i \geq 0 \tag{7}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \ \forall i \in [m] \tag{8}$$

Constrained Optimization

The dual optimization problem for SVMs of the separable cases is

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{6}$$

$$\text{s.t.} \quad \alpha_i \geq 0 \tag{7}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \ \forall i \in [m] \tag{8}$$

- Lagrange multiplier $\boldsymbol{\alpha}$ is also called dual variable
- This is an optimization problem only about $\boldsymbol{\alpha}$
- The dual problem is defined on the inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$

Optimization via Gradient Descent

The basic form of an optimization problem

$$\begin{aligned} &\min f(\boldsymbol{\theta}) \\ &\text{s.t.}\, \boldsymbol{\theta} \in B \end{aligned} \tag{9}$$

where $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ is the objective function and $B \subseteq \mathbb{R}^d$ is the constraint on $\boldsymbol{\theta}$, which usually can be formulated as a set of inequalities (e.g., SVM)

Optimization via Gradient Descent

The basic form of an optimization problem

$$\begin{aligned} &\min f(\boldsymbol{\theta}) \\ &\text{s.t.} \boldsymbol{\theta} \in B \end{aligned} \tag{9}$$

where $f(\boldsymbol{\theta}) : \mathbb{R}^d \rightarrow \mathbb{R}$ is the objective function and $B \subseteq \mathbb{R}^d$ is the constraint on $\boldsymbol{\theta}$, which usually can be formulated as a set of inequalities (e.g., SVM)

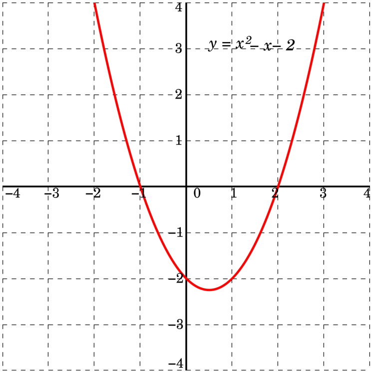
In this lecture

- we only focus on unconstrained optimization problem, in other words, $\boldsymbol{\theta} \in \mathbb{R}^d$
- assume f is convex and differentiable

Review: Gradient of a 1-D Function

Consider the gradient of this 1-dimensional function

$$y = f(x) = x^2 - x - 2 \tag{10}$$

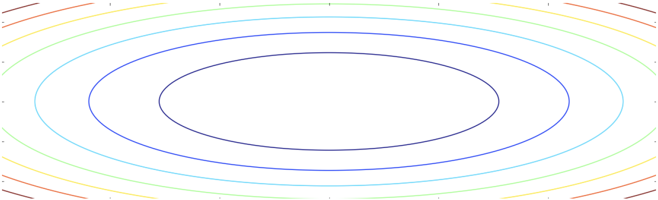


Review: Gradient of a 2-D Function

Now, consider a 2-dimensional function with $\boldsymbol{x} = (x_1, x_2)$

$$y = f(\boldsymbol{x}) = x_1^2 + 2x_2^2 \tag{11}$$

Here is the contour plot of this function



We are going to use this as our running example

Gradient Descent

To learn the parameter $\boldsymbol{\theta}$, the learning algorithm needs to update it iteratively using the following three steps

1. Choose an initial point $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$
2. Repeat
$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \tag{12}$$
   where η_t is the learning rate at time t
3. Go back step 1 until it converges

Gradient Descent

To learn the parameter $\boldsymbol{\theta}$, the learning algorithm needs to update it iteratively using the following three steps

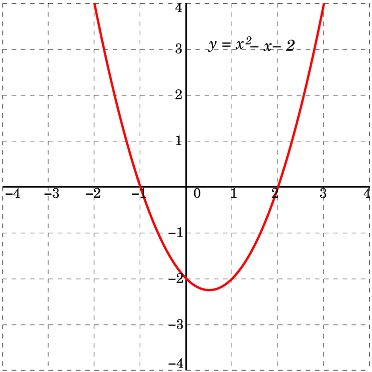
1. Choose an initial point $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$
2. Repeat
$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \tag{12}$$
   where η_t is the learning rate at time t
3. Go back step 1 until it converges

$\nabla f(\boldsymbol{\theta})$ is defined as

$$\nabla f(\boldsymbol{\theta}) = \left(\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_1}, \cdots, \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_d}\right) \tag{13}$$

Gradient Descent Interpretation

An intuitive justification of the gradient descent algorithm is to consider the following plot



The direction of the gradient is the direction that the function has the *"fastest increase"*.

Gradient Descent Interpretation (II)

Theoretical justification

- First-order Taylor approximation

$$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}) + \langle \Delta\boldsymbol{\theta}, \nabla f \rangle \big|_{\boldsymbol{\theta}} \tag{14}$$

Gradient Descent Interpretation (II)

Theoretical justification

- First-order Taylor approximation

$$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}) + \langle \Delta\boldsymbol{\theta}, \nabla f \rangle \big|_{\boldsymbol{\theta}} \tag{14}$$

- In gradient descent, $\Delta\boldsymbol{\theta} = -\eta \nabla f \big|_{\boldsymbol{\theta}}$

Gradient Descent Interpretation (II)

Theoretical justification

- First-order Taylor approximation

$$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}) + \langle \Delta\boldsymbol{\theta}, \nabla f \rangle \Big|_{\boldsymbol{\theta}} \tag{14}$$

- In gradient descent, $\Delta\boldsymbol{\theta} = -\eta \nabla f \Big|_{\boldsymbol{\theta}}$
- Therefore, we have

$$\begin{aligned} f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) &\approx f(\boldsymbol{\theta}) + \langle \Delta\boldsymbol{\theta}, \nabla f \rangle \Big|_{\boldsymbol{\theta}} \\ &= f(\boldsymbol{\theta}) - \langle \eta \nabla f, \nabla f \rangle \Big|_{\boldsymbol{\theta}} \\ &= f(\boldsymbol{\theta}) - \eta \|\nabla f\|_2^2 \Big|_{\boldsymbol{\theta}} \leq f(\boldsymbol{\theta}) \end{aligned} \tag{15}$$

Gradient Descent Interpretation (III)

Consider the second-order Taylor approximation of f

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}}\nabla^2 f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

Gradient Descent Interpretation (III)

Consider the second-order Taylor approximation of f

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}} \nabla^2 f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

- The quadratic approximation of f with the following

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2\eta}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}}(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

Gradient Descent Interpretation (III)

Consider the second-order Taylor approximation of f

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}} \nabla^2 f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

- The quadratic approximation of f with the following

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2\eta}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}}(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

- Minimize $f(\boldsymbol{\theta}')$ wrt $\boldsymbol{\theta}'$

$$\begin{aligned} \frac{\partial f(\boldsymbol{\theta}')}{\partial \boldsymbol{\theta}'} &\approx \nabla f(\boldsymbol{\theta}) + \frac{1}{2\eta}(\boldsymbol{\theta}' - \boldsymbol{\theta}) = 0 \\ &\Rightarrow \boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \cdot \nabla f(\boldsymbol{\theta}) \end{aligned} \tag{16}$$

Gradient Descent Interpretation (III)

Consider the second-order Taylor approximation of f

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}} \nabla^2 f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

- The quadratic approximation of f with the following

$$f(\boldsymbol{\theta}') \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) + \frac{1}{2\eta}(\boldsymbol{\theta}' - \boldsymbol{\theta})^{\mathsf{T}}(\boldsymbol{\theta}' - \boldsymbol{\theta})$$

- Minimize $f(\boldsymbol{\theta}')$ wrt $\boldsymbol{\theta}'$

$$\begin{aligned} \frac{\partial f(\boldsymbol{\theta}')}{\partial \boldsymbol{\theta}'} &\approx \nabla f(\boldsymbol{\theta}) + \frac{1}{2\eta}(\boldsymbol{\theta}' - \boldsymbol{\theta}) = 0 \\ &\Rightarrow \boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \cdot \nabla f(\boldsymbol{\theta}) \end{aligned} \tag{16}$$

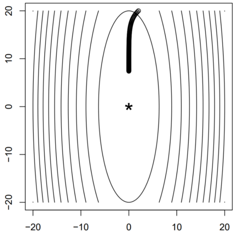
- Gradient descent chooses the next point $\boldsymbol{\theta}'$ to minimize the function

Step size

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \tag{17}$$

If choose fixed step size $\eta_t = \eta_0$, consider the following function

$$f(\boldsymbol{\theta}) = (10\theta_1^2 + \theta_2^2)/2$$



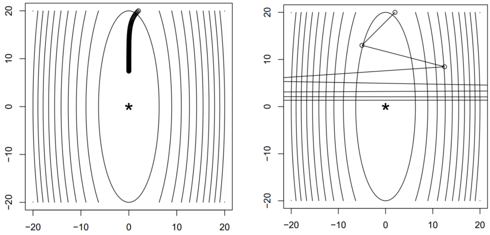
(a) Too small

Step size

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \cdot \frac{\partial f(\theta)}{\partial \theta}\Big|_{\theta=\theta^{(t)}} \tag{17}$$

If choose fixed step size $\eta_t = \eta_0$, consider the following function

$$f(\theta) = (10\theta_1^2 + \theta_2^2)/2$$



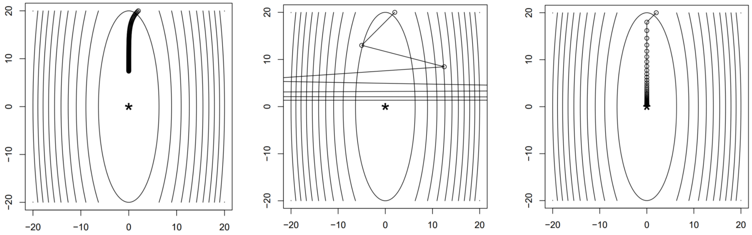
(d) Too small (e) Too large

Step size

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \tag{17}$$

If choose fixed step size $\eta_t = \eta_0$, consider the following function

$$f(\boldsymbol{\theta}) = (10\theta_1^2 + \theta_2^2)/2$$



(g) Too small (h) Too large (i) Just right

Optimal Step Sizes

- **Exact Line Search** Solve this one-dimensional subproblem

$$t \leftarrow \operatorname*{argmin}_{s \geq 0} f(\boldsymbol{\theta} - s\nabla f(\boldsymbol{\theta})) \tag{18}$$

Optimal Step Sizes

- **Exact Line Search** Solve this one-dimensional subproblem

$$t \leftarrow \operatorname*{argmin}_{s \geq 0} f(\boldsymbol{\theta} - s \nabla f(\boldsymbol{\theta})) \tag{18}$$

- **Backtracking Line Search**: with parameters $0 < \beta < 1$, $0 < \alpha \leq 1/2$, and large initial value η_t, if

$$f(\boldsymbol{\theta} - \eta \nabla f(\boldsymbol{\theta})) > f(\boldsymbol{\theta}) - \alpha \eta_t \|\nabla f(\boldsymbol{\theta})\|_2^2 \tag{19}$$

shrink $\eta_t \leftarrow \beta \eta_t$

Optimal Step Sizes

- **Exact Line Search** Solve this one-dimensional subproblem

$$t \leftarrow \operatorname*{argmin}_{s \geq 0} f(\boldsymbol{\theta} - s\nabla f(\boldsymbol{\theta})) \tag{18}$$

- **Backtracking Line Search**: with parameters $0 < \beta < 1$, $0 < \alpha \leq 1/2$, and large initial value η_t, if

$$f(\boldsymbol{\theta} - \eta\nabla f(\boldsymbol{\theta})) > f(\boldsymbol{\theta}) - \alpha\eta_t \|\nabla f(\boldsymbol{\theta})\|_2^2 \tag{19}$$

  shrink $\eta_t \leftarrow \beta\eta_t$

- Usually, this is not worth the effort, since the computational complexity may be too high (e.g., f is a neural network)

Convergence Analysis

- f is convex and differentiable, additionally

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta}')\|_2 \leq L \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 \tag{20}$$

for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$ and L is a fixed positive value

Convergence Analysis

- f is convex and differentiable, additionally

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta}')\|_2 \leq L \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 \tag{20}$$

  for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$ and L is a fixed positive value

- **Theorem**: Gradient descent with fixed step size $\eta_0 \leq 1/L$ satisfies

$$f(\boldsymbol{\theta}^{(t)}) - f^* \leq \frac{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|_2^2}{2\eta_0 t} \tag{21}$$

  where f^* is the optimal value and $\boldsymbol{\theta}^*$ is the optimal parameter

Convergence Analysis

- f is convex and differentiable, additionally

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta}')\|_2 \leq L \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 \tag{20}$$

  for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$ and L is a fixed positive value
- **Theorem**: Gradient descent with fixed step size $\eta_0 \leq 1/L$ satisfies

$$f(\boldsymbol{\theta}^{(t)}) - f^* \leq \frac{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|_2^2}{2\eta_0 t} \tag{21}$$

  where f^* is the optimal value and $\boldsymbol{\theta}^*$ is the optimal parameter
- Same result holds for backtracking with η_0 replaced by β/L

Stochastic Gradient Descent

Gradient Descent

Given a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, the loss function is defined as

$$L(h_{\boldsymbol{\theta}}, S) = \frac{1}{m} \sum_{i=1}^{m} R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \tag{22}$$

where R is the risk function

Examples:

- Logistic regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = -\log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \tag{23}$$

- Linear regression

$$R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \|h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\|_2^2 \tag{24}$$

Gradient Descent (II)

- Consider the gradient of loss function $\nabla L(h_{\boldsymbol{\theta}}, S)$

$$\nabla L(h_{\boldsymbol{\theta}}, S) = \frac{1}{m} \sum_{i=1}^{m} \nabla R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \tag{25}$$

Gradient Descent (II)

- Consider the gradient of loss function $\nabla L(h_{\boldsymbol{\theta}}, S)$

$$\nabla L(h_{\boldsymbol{\theta}}, S) = \frac{1}{m} \sum_{i=1}^{m} \nabla R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \tag{25}$$

- To simplify the notation, let $f_i(\boldsymbol{\theta}) = R(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i)$ and $f(\boldsymbol{\theta}) = L(h_{\boldsymbol{\theta}}, S)$, then

$$\nabla f(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(\boldsymbol{\theta}) \tag{26}$$

Stochastic Gradient Descent

To learn the parameter $\boldsymbol{\theta}$, we can compute the gradient with one training example $(\boldsymbol{x}_i, y_i)$ each time step and update the parameter as

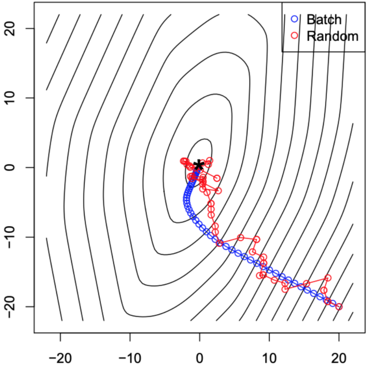
$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_t \cdot \nabla f_i(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t)}} \qquad (27)$$

where

- t: time step
- $\nabla f_i(\boldsymbol{\theta}^{(t)})$ is the gradient of the single-example loss L
- η_t is the learning rate (step size)

Stochastic?

Compare gradient descent and *stochastic* gradient descent



As each step SGD only uses the gradient from one training example, it can be viewed as a gradient descent method with some randomness

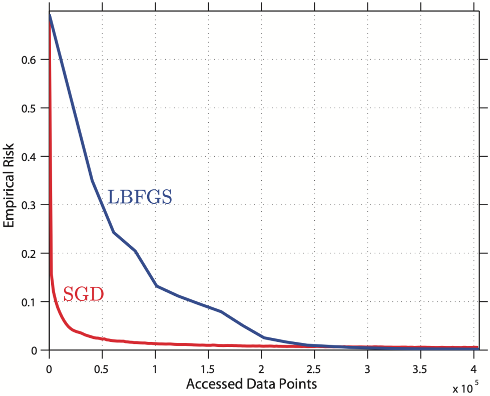
Motivation

There are at least two motivations of using SGD

- SGD can be a big savings in terms of memory usage
  - learning with large-scale data
  - models with lots of parameters
- The iteration cost of SGD is independent of sample size m

Motivation (II)

An empirical comparison between SGD and a batch optimization method (L-BFGS) on a binary classification problem with logistic regression [Bottou et al., 2018]



How to Choose an Example

- **Cyclic Rule**: choose $i \in (1, 2, \ldots, m)$ in order

How to Choose an Example

- **Cyclic Rule**: choose $i \in (1, 2, \ldots, m)$ in order
- **Randomized Rule**: Every iteration, choose $i \in [m]$ uniformly at random
  - In practice, randomized rule is more common, since we have

    $$E\left[\nabla f_i(\boldsymbol{\theta})\right] \approx \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) \tag{28}$$

    as an unbiased estimate of $\nabla f(\boldsymbol{\theta})$
  - Alternatively, shuffle the training example at the end of each training epoch

Convergence of SGD

The convergence of SGD usually requires **diminishing step sizes**

- The usual conditions on the learning rates are

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \sum_{t=1}^{\infty} \eta_t^2 \leq \infty \tag{29}$$

[Bottou et al., 1998]

Convergence of SGD

The convergence of SGD usually requires **diminishing step sizes**

- The usual conditions on the learning rates are

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \sum_{t=1}^{\infty} \eta_t^2 \leq \infty \tag{29}$$

- A simplest function that satisfies these conditions is

$$\eta_t = \frac{1}{t} \tag{30}$$

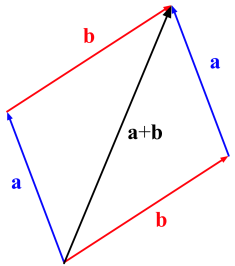
[Bottou et al., 1998]

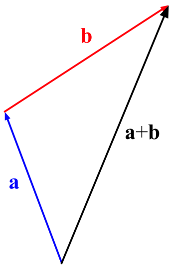
SGD with Momentum

Review: Vector Addition

The parallelogram law of vector addition

$$c = a + b \tag{31}$$





SGD with Momentum

Given the loss function $f(\boldsymbol{\theta})$ to be minimized, SGD with momentum is given by

$$\boldsymbol{v}^{(t)} = \mu \boldsymbol{v}^{(t-1)} + \nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t-1)}} \quad (32)$$

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta_t \boldsymbol{v}^{(t)} \quad (33)$$

where

- η_t is still the learning rate
- $\mu \in [0, 1]$ is the momentum coefficient. Usually, $\mu = 0.99$ or 0.999.

Intuitive Explanation

(Note: the arrow show the opposite direction of the gradient)

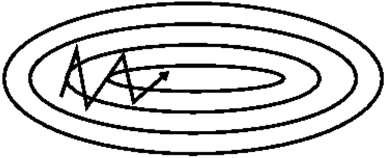
(a) SGD without momentum

Figure: The effect of momentum in SGD: reduce the fluctuation (Credit: Genevieve B. Orr)

Intuitive Explanation

(Note: the arrow show the opposite direction of the gradient)

(a) SGD without momentum



(b) SGD with momentum

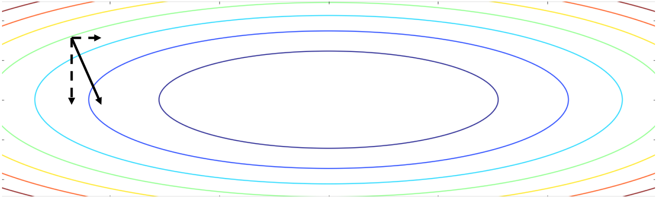
Figure: The effect of momentum in SGD: reduce the fluctuation (Credit: Genevieve B. Orr)

Another Example with Contour Plot

Consider the following problem

$$y = x_1^2 + 10x_2^2 \tag{34}$$

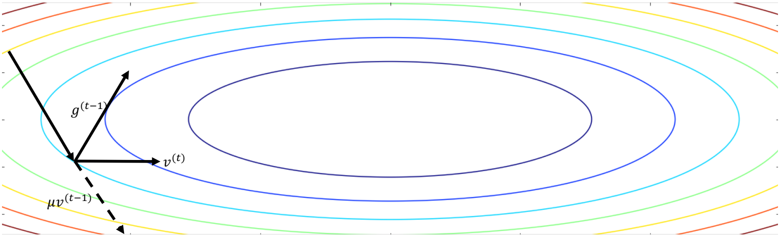
$$\frac{\partial y}{\partial x_1} = 2x_1 \qquad \frac{\partial y}{\partial x_2} = 20x_2 \tag{35}$$



Note: the arrow show the opposite direction of the gradient

Add the previous gradient reduce the fluctuation of stochastic gradients

$$\boldsymbol{v}^{(t)} = \mu\boldsymbol{v}^{(t-1)} + \boldsymbol{g}^{(t-1)} \tag{36}$$



Note: the arrow show the opposite direction of the gradient

Adaptive Learning Rates

Basic Idea

The basic idea of using adaptive learning rates is to make sure that

all θ_k's converge roughly at the same speed

For neural networks, the motivation of picking a different learning rate for each θ_k (the k-th component of parameter θ) is not new [LeCun et al., 2012] (the article was originally published in 1998).

AdaGrad

The basic idea of **AdaGrad** [Duchi et al., 2011] is to modify the learning rate η for $\boldsymbol{\theta}_k$ by using the history of the gradients

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{37}$$

AdaGrad

The basic idea of **AdaGrad** [Duchi et al., 2011] is to modify the learning rate η for $\boldsymbol{\theta}_k$ by using the history of the gradients

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{37}$$

- $g_k^{(t-1)} = [\nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t-1)}}]_k$ is the k-th component of $\nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t-1)}}$
- $G_{k,k}^{(t-1)} = \sum_{i=1}^{t-1}(g_k^{(i)})^2$

AdaGrad

The basic idea of **AdaGrad** [Duchi et al., 2011] is to modify the learning rate η for $\boldsymbol{\theta}_k$ by using the history of the gradients

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{37}$$

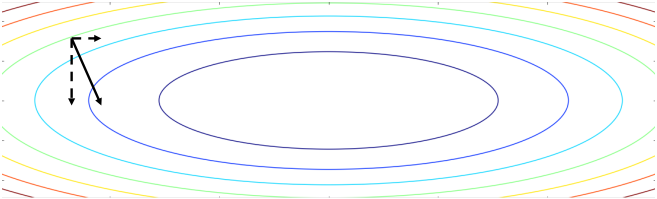
- $g_k^{(t-1)} = [\nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t-1)}}]_k$ is the k-th component of $\nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t-1)}}$
- $G_{k,k}^{(t-1)} = \sum_{i=1}^{t-1}(g_k^{(i)})^2$
- η_0 is the initial learning rate
- ϵ is a smoothing parameter usually with order 10^{-6}

AdaGrad: Intuitive Explanation

Consider the gradient of a 2-dimensional optimization problem with $\boldsymbol{\theta} = (\theta_1, \theta_2)$

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{38}$$

The magnitude of gradient along θ_2 is often larger then θ_1

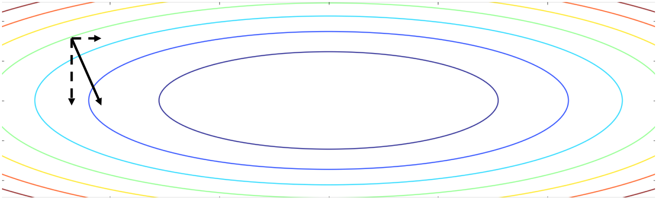


AdaGrad: Intuitive Explanation

Consider the gradient of a 2-dimensional optimization problem with $\boldsymbol{\theta} = (\theta_1, \theta_2)$

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{G_{k,k}^{(t-1)} + \epsilon}} g_k^{(t-1)} \tag{38}$$

The magnitude of gradient along θ_2 is often larger then θ_1



AdaGrad helps shrink step sizes along θ_2 that allows the procedure converges roughly at the same speed

RMSProp

RMSProp (Root Mean Square Propagation) uses a moving average over the past gradients

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \frac{\eta_0}{\sqrt{\boldsymbol{r}_k^{(t)} + \epsilon}} g_k^{(t-1)} \tag{39}$$

where

$$r_k^{(t)} = \rho r_k^{(t-1)} + (1-\rho)[g_k^{(t-1)}]^2 \tag{40}$$

and $\rho \in (0, 1)$, k is the dimension index, and t is the time stemp

[Hinton, 2012]

Adam

The Adam algorithm [Kingma and Ba, 2014] is proposed to combine the idea of SGD with moment and RMSProp

Adam

The Adam algorithm [Kingma and Ba, 2014] is proposed to combine the idea of SGD with moment and RMSProp

$$v_k^{(t)} = \mu v_k^{(t-1)} + (1-\mu) g_k^{(t-1)} \quad (41)$$

$$r_k^{(t)} = \rho r_k^{(t-1)} + (1-\rho)[g_k^{(t-1)}]^2 \quad (42)$$

$$\hat{v}_k^{(t)} = \frac{v_k^{(t)}}{1-\mu^t} \quad (43)$$

$$\hat{r}_k^{(t)} = \frac{r_k^{(t)}}{1-\rho^t} \quad (44)$$

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \eta_0 \frac{\hat{v}_k^{(t)}}{\sqrt{\hat{r}_k^{(t)}} + \epsilon} \quad (45)$$

The default values of μ and ρ are 0.9 and 0.999 respectively.

Adam

The Adam algorithm [Kingma and Ba, 2014] is proposed to combine the idea of SGD with moment and RMSProp

$$v_k^{(t)} = \mu v_k^{(t-1)} + (1-\mu) g_k^{(t-1)} \quad (41)$$

$$r_k^{(t)} = \rho r_k^{(t-1)} + (1-\rho)[g_k^{(t-1)}]^2 \quad (42)$$

$$(43)$$

$$(44)$$

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \eta_0 \frac{\hat{v}_k^{(t)}}{\sqrt{\hat{r}_k^{(t)} + \epsilon}} \quad (45)$$

The default values of μ and ρ are 0.9 and 0.999 respectively.

How to Choose a Optimization Algorithm?

Summary of learning methods for neural networks

- For small datasets (e.g. 10,000 cases) or bigger datasets without much redundancy, use a full-batch method.
  - Conjugate gradient, LBFGS ...
  - adaptive learning rates, rprop ...
- For big, redundant datasets use mini-batches.
  - Try gradient descent with momentum.
  - Try rmsprop (with momentum ?)
  - Try LeCun's latest recipe.

- Why there is no simple recipe:

  Neural nets differ a lot:
  - Very deep nets (especially ones with narrow bottlenecks).
  - Recurrent nets.
  - Wide shallow nets.

  Tasks differ a lot:
  - Some require very accurate weights, some don't.
  - Some have many very rare cases (e.g. words).

[Hinton, 2012, Lecture Notes in 2012]

Reference

Bottou, L., Curtis, F. E., and Nocedal, J. (2018).
Optimization methods for large-scale machine learning.
Siam Review, 60(2):223–311.

Bottou, L. et al. (1998).
Online learning and stochastic approximations.
On-line learning in neural networks, 17(9):142.

Duchi, J., Hazan, E., and Singer, Y. (2011).
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of machine learning research, 12(Jul):2121–2159.

Hinton, G. (2012).
Neural networks for machine learning.
Lecture notes.

Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012).
Efficient backprop.
In *Neural networks: Tricks of the trade*, pages 9–48. Springer.