# A Comprehensive Introduction to Gradient Descent

Recitation Notes for CPSC-381/581: Introduction to Machine Learning

## Contents



# 1 Introduction

Gradient descent is an iterative optimization algorithm used to find the minimum of a function. In machine learning, it is commonly applied to minimize loss functions, such as the mean squared error in linear regression or the negative log-likelihood in logistic regression. This document explains the theory behind gradient descent, provides detailed derivations for linear and logistic regression, and introduces the concept of backtracking line search for adapting the learning rate.

# 2 Gradient Descent Basics

## 2.1 The General Idea

Suppose we have a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ that we wish to minimize. The goal is to find

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}).$$

Gradient descent updates the parameter vector $\mathbf{w}$ iteratively by moving in the opposite direction of the gradient, since the gradient points in the direction of greatest increase. The basic update rule is:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla f(\mathbf{w}^{(t)}),$$

where:

- $\mathbf{w}^{(t)}$ is the parameter vector at iteration $t$,
- $\alpha > 0$ is the learning rate (step size),
- $\nabla f(\mathbf{w}^{(t)})$ is the gradient of $f$ evaluated at $\mathbf{w}^{(t)}$.

### 2.2 Why It Works: A Taylor Expansion Perspective

Assuming that $f$ is differentiable (and even twice differentiable), we can approximate $f$ near $\mathbf{w}^{(t)}$ using a Taylor expansion:

$$f(\mathbf{w}^{(t)} + \Delta \mathbf{w}) \approx f(\mathbf{w}^{(t)}) + \nabla f(\mathbf{w}^{(t)})^T \Delta \mathbf{w} + \frac{1}{2} \Delta \mathbf{w}^T H(\mathbf{w}^{(t)}) \Delta \mathbf{w},$$

where $H(\mathbf{w}^{(t)})$ is the Hessian matrix of second derivatives. If we choose $\Delta \mathbf{w} = -\alpha \nabla f(\mathbf{w}^{(t)})$, then the first-order term becomes

$$-\alpha \|\nabla f(\mathbf{w}^{(t)})\|^2,$$

which shows that, for sufficiently small $\alpha$, the function value decreases.

## 3 Gradient Descent for Linear Regression

### 3.1 Loss Function

In linear regression, our goal is to predict a target variable $y$ from input features $\mathbf{x}$ using a linear model:

$$\hat{y} = \mathbf{w}^T \mathbf{x},$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector.

The mean squared error (MSE) loss is given by:

$$\ell(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \Big( \mathbf{w}^T \mathbf{x}^{(n)} - y^{(n)} \Big)^2.$$

A common variation is to use the scaled loss:

$$\ell(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} \Big( \mathbf{w}^T \mathbf{x}^{(n)} - y^{(n)} \Big)^2,$$

which simplifies the gradient expressions by removing constant factors.

### 3.2 Derivation of the Gradient

For clarity, we begin with the unscaled loss:

$$\ell(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \Big( \mathbf{w}^T \mathbf{x}^{(n)} - y^{(n)} \Big)^2.$$

Let's assume that $\mathbf{w} = (w_0, w_1)^T$ and $\mathbf{x}^{(n)} = (x_0^{(n)}, x_1^{(n)})^T$.

**Partial Derivative with Respect to $w_0$**

Using the chain rule,

$$\begin{aligned}\frac{\partial}{\partial w_0}\ell(w_0,w_1) &= \frac{\partial}{\partial w_0}\left[\frac{1}{N}\sum_{n=1}^{N}\Big(w_0x_0^{(n)}+w_1x_1^{(n)}-y^{(n)}\Big)^2\right]\\ &= \frac{1}{N}\sum_{n=1}^{N}2\Big(w_0x_0^{(n)}+w_1x_1^{(n)}-y^{(n)}\Big)\frac{\partial}{\partial w_0}\Big(w_0x_0^{(n)}+w_1x_1^{(n)}\Big)\\ &= \frac{2}{N}\sum_{n=1}^{N}\Big(w_0x_0^{(n)}+w_1x_1^{(n)}-y^{(n)}\Big)x_0^{(n)}.\end{aligned}$$

Similarly, for $w_1$:

$$\frac{\partial}{\partial w_1}\ell(w_0,w_1)=\frac{2}{N}\sum_{n=1}^{N}\Big(w_0x_0^{(n)}+w_1x_1^{(n)}-y^{(n)}\Big)x_1^{(n)}.$$

Thus, in vector form:

$$\nabla\ell(\mathbf{w})=\frac{2}{N}\sum_{n=1}^{N}\Big(\mathbf{w}^T\mathbf{x}^{(n)}-y^{(n)}\Big)\mathbf{x}^{(n)}.$$

**Removing the Constant Factor**

By redefining the loss as:

$$\ell(\mathbf{w})=\frac{1}{2N}\sum_{n=1}^{N}\Big(\mathbf{w}^T\mathbf{x}^{(n)}-y^{(n)}\Big)^2,$$

the derivative becomes:

$$\nabla\ell(\mathbf{w})=\frac{1}{N}\sum_{n=1}^{N}\Big(\mathbf{w}^T\mathbf{x}^{(n)}-y^{(n)}\Big)\mathbf{x}^{(n)},$$

since the 2 in the derivative cancels with the $1/2$ in the loss definition.

## 3.3 Gradient Descent Update for Linear Regression

The update rule is given by:

$$\mathbf{w}^{(t+1)}=\mathbf{w}^{(t)}-\alpha\nabla\ell(\mathbf{w}^{(t)}).$$

For the unscaled loss, this is:

$$\mathbf{w}^{(t+1)}=\mathbf{w}^{(t)}-\alpha\frac{2}{N}\sum_{n=1}^{N}\Big(\mathbf{w}^{(t)T}\mathbf{x}^{(n)}-y^{(n)}\Big)\mathbf{x}^{(n)}.$$

When using the scaled loss, the update simplifies to:

$$\mathbf{w}^{(t+1)}=\mathbf{w}^{(t)}-\alpha\frac{1}{N}\sum_{n=1}^{N}\Big(\mathbf{w}^{(t)T}\mathbf{x}^{(n)}-y^{(n)}\Big)\mathbf{x}^{(n)}.$$

# 4 Gradient Descent for Logistic Regression

## 4.1 Loss Function

In logistic regression for binary classification, the model predicts:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}),$$

where the sigmoid function is defined by:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

The loss function, which is the negative log-likelihood, is:

$$\ell(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \log\Big(1 + \exp\big(-y^{(n)}\, \mathbf{w}^T \mathbf{x}^{(n)}\big)\Big),$$

with $y^{(n)} \in \{-1, +1\}$.

## 4.2 Gradient Derivation for Logistic Regression (Sketch)

Using the chain rule, the derivative with respect to a weight $w_i$ is given by:

$$\frac{\partial}{\partial w_i}\, \ell(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial w_i} \log\Big(1 + \exp\big(-y^{(n)}\, \mathbf{w}^T \mathbf{x}^{(n)}\big)\Big).$$

After some algebra (and a change of variables in the exponential term), this simplifies to:

$$\frac{\partial}{\partial w_i}\, \ell(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \frac{y^{(n)}\, x_i^{(n)}}{1 + \exp\big(y^{(n)}\, \mathbf{w}^T \mathbf{x}^{(n)}\big)}.$$

In vector form:

$$\nabla \ell(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \frac{y^{(n)}\, \mathbf{x}^{(n)}}{1 + \exp\big(y^{(n)}\, \mathbf{w}^T \mathbf{x}^{(n)}\big)}.$$

## 4.3 Gradient Descent Update for Logistic Regression

The update rule is the same in form as for linear regression:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla \ell(\mathbf{w}^{(t)}).$$

Thus, explicitly:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \frac{1}{N} \sum_{n=1}^{N} \frac{y^{(n)}\, \mathbf{x}^{(n)}}{1 + \exp\big(y^{(n)}\, \mathbf{w}^{(t)T} \mathbf{x}^{(n)}\big)}.$$

# 5 Learning Rate and Backtracking Line Search

## 5.1 The Role of the Learning Rate $\alpha$

The learning rate $\alpha$ determines how large a step we take in the direction of the negative gradient. If $\alpha$ is too large, the algorithm may overshoot the minimum and diverge. If $\alpha$ is too small, convergence will be slow.

### 5.2 Backtracking Line Search

Backtracking line search is a method to choose $\alpha$ adaptively to ensure that each update produces a sufficient decrease in the loss. The algorithm proceeds as follows:

1. **Initialization:** Choose an initial step size $\alpha_0$.
2. **Candidate Update:** Compute the candidate update:
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla \ell(\mathbf{w}^{(t)}).$$
3. **Armijo–Goldstein Condition:** Check if the following inequality holds:
$$\ell\Big(\mathbf{w}^{(t)} - \alpha \nabla \ell(\mathbf{w}^{(t)})\Big) \leq \ell(\mathbf{w}^{(t)}) - \beta \alpha \|\nabla \ell(\mathbf{w}^{(t)})\|^2,$$
where $\beta \in (0,1)$ is a constant (often set to a value such as 0.5 or 0.1).
4. **Adjust $\alpha$:** If the condition is not met, reduce $\alpha$ (e.g., set $\alpha \leftarrow \frac{1}{2}\alpha$) and repeat the check.

#### Proof Sketch of Sufficient Decrease

Assume that $\nabla \ell$ is Lipschitz continuous with Lipschitz constant $L$. Then, using Taylor's theorem, we have:

$$\ell\Big(\mathbf{w}^{(t)} - \alpha \nabla \ell(\mathbf{w}^{(t)})\Big) \leq \ell(\mathbf{w}^{(t)}) - \alpha \|\nabla \ell(\mathbf{w}^{(t)})\|^2 + \frac{L\alpha^2}{2}\|\nabla \ell(\mathbf{w}^{(t)})\|^2.$$

For sufficiently small $\alpha$, the term $\frac{L\alpha^2}{2}\|\nabla \ell(\mathbf{w}^{(t)})\|^2$ is dominated by $\alpha \|\nabla \ell(\mathbf{w}^{(t)})\|^2$, ensuring that the Armijo–Goldstein condition holds when

$$\frac{L\alpha}{2} \leq \beta.$$

Thus, there exists an $\alpha > 0$ satisfying the condition.

## 6 Key Points Summary

- **Core Update Rule:** The central update in gradient descent is
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla \ell(\mathbf{w}^{(t)}).$$
This rule applies universally to both linear and logistic regression.
- **Gradient Computation:** Gradients for both models are derived via the chain rule. For linear regression, explicit computation shows the emergence of the factor of 2, while for logistic regression the gradient involves the sigmoid function's derivative.
- **Adaptive Learning Rate:** Backtracking line search provides a systematic way to choose the learning rate $\alpha$ by ensuring that each update produces a sufficient decrease in the loss, as guaranteed by the Armijo–Goldstein condition.
- **Convergence Guarantees:** Under the assumption of Lipschitz continuity of the gradient, there is a theoretical guarantee that a suitable $\alpha$ exists such that the sufficient decrease condition holds, leading to convergence.