Gradient Descent

Andrew L. Mackey

Artificial Intelligence
University of Arkansas – Fort Smith

Spring 2023

Topics

Terminology

- **Epoch** – a step that covers all the training examples
- **Learning Rate** (α) – also called the step size, represents a fixed constant or a value that decays over time as the learning process continues.
- **Loss Function** $\mathcal{L}(\mathbf{X}, \mathbf{y}, \hat{\mathbf{y}})$ – the amount of utility lost by predicting $h(x) = y$ when the correct answer is $f(x) = y$. We view this as a function that needs to be minimized.

Training Data $\mathbf{X}$

Let $\mathbf{X}$ be a matrix comprised of p number of features.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_p \end{bmatrix}$$

Each of the features would be represented as follows:

$$\mathbf{x}_1 = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ \vdots \\ x_{n1} \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ \vdots \\ x_{n2} \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ \vdots \\ x_{n3} \end{bmatrix} \quad \dots \quad \mathbf{x}_p = \begin{bmatrix} x_{1p} \\ x_{2p} \\ x_{3p} \\ \vdots \\ x_{np} \end{bmatrix}$$

Training Data $\mathbf{X}$ (cont.)

We can define $\mathbf{X}$ as the following set of observations:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x^{(1)}} \\ \mathbf{x^{(2)}} \\ \mathbf{x^{(3)}} \\ \vdots \\ \mathbf{x^{(n)}} \end{bmatrix} \qquad \text{where } \mathbf{x}^{(i)} = \begin{bmatrix} x_{i1} & x_{i2} & x_{i3} & \cdots & x_{ip} \end{bmatrix}$$

Training Data $\mathbf{X}$ (cont.)

Since $\mathbf{X}$ is defined as a matrix comprised of n rows (or observations) and p features, we can represent it as follows:

$$\underset{n \times p}{\mathbf{X}} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2p} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3p} \\ \vdots & & \ddots & & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{np} \end{bmatrix}$$

Training Data $\mathbf{X}$ (cont.)

We will add a column vector of 1's to the start of the matrix to represent the **bias term**.

$$\underset{n\times(p+1)}{\mathbf{X}} = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & x_{23} & \cdots & x_{2p} \\ 1 & x_{31} & x_{32} & x_{33} & \cdots & x_{3p} \\ \vdots & & & \ddots & & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \cdots & x_{np} \end{bmatrix}$$

Weights $\mathbf{w}$

For each of the p features and 1 bias term (the column vector of 1's), we will maintain a vector of weights $\mathbf{w}$. We will use w_0 to represent the weight for the bias term.

$$\underset{(p+1)\times 1}{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_p \end{bmatrix}$$

Functions

We will define the function $h(\cdot)$ to represent our model as it produces a hypothesis of the correct solution based on the $\mathbf{x}^{(i)}$ input data.

$$h : \mathbb{R}^{p+1} \rightarrow \mathbb{R}$$

Our function $h_{\mathbf{w}}(\cdot)$ will use the weights $\mathbf{w} = \{w_0, w_1, ..., w_p\}$ we generate to multiply by the input vector $\mathbf{x}^{(i)}$ (starting with a feature with the value of $x_0 = 1$ and values for the other p features).

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} = \mathbf{x}\mathbf{w} = \sum_{i=0}^{p} w_i x_i$$

Topics

Batch Gradient Descent (BGD)

Sums the loss over N training examples for every step.

Algorithm 1 Batch Gradient Descent

procedure GRADIENT-DESCENT($\mathbf{X}$, $\mathbf{y}$, α)
    $\mathbf{w} \leftarrow$ initialize to be any point in the parameter space
    $i = 0$
    while not converged **do**
        for $k = 1$ to p **do**
            $w_k = w_k - \alpha \frac{\partial}{\partial w_k} \left[\mathcal{L}\big(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{X}) \big) \right]$
        $b = b - \alpha \frac{\partial}{\partial b} \left[\mathcal{L}\big(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{X}) \big) \right]$
        $i \leftarrow i + 1$
        Update convergence values

Batch Gradient Descent (BGD)

- i – the current iteration number
- b – the bias weight
- $\mathbf{w}$ – the vector of weights
- $\mathbf{X}$ – the matrix of training data
- p – the number of features
- $\mathbf{y}$ – the vector of responses
- $\mathcal{L}(\cdot)$ – the loss function

Stochastic Gradient Descent (SGD)

Sums the loss over a randomly selected training example at each step.

Algorithm 2 Stochastic Gradient Descent

procedure STOCHASTIC-GRADIENT-DESCENT($\mathbf{X}$, $\mathbf{y}$, α)
    $\mathbf{w} \leftarrow$ initialize to be any point in the parameter space
    $i = 0$, $j = 0$
    while not converged **do**
        for $j = 1$ to n **do**
            Obtain record $\mathbf{x^{(j)}}$
            for $k = 1$ to p **do**
                $$w_k = w_k - \alpha \frac{\partial}{\partial w_k} \left[\mathcal{L}^{(j)}\left(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{x}^{(j)}) \right) \right]$$
            $$b = b - \alpha \frac{\partial}{\partial b} \left[\mathcal{L}^{(j)}\left(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{x}^{(j)}) \right) \right]$$
            $i \leftarrow i + 1$
        Update convergence values

Stochastic Gradient Descent (SGD)

- i – the current iteration number
- j – the current training record (the vector is $\mathbf{x}^{(j)}$)
- b – the bias weight
- $\mathbf{w}$ – the vector of weights
- $\mathbf{X}$ – the matrix of training data
- p – the number of features
- $\mathbf{y}$ – the vector of responses
- $\mathcal{L}(\cdot)$ – the loss function

Mini-batch Gradient Descent (MBGD)

Sums the loss over a batch of m examples out of N possible.

Algorithm 3 Mini-batch Gradient Descent

procedure MINI-BATCH-GRADIENT-DESCENT($\mathbf{X}$, $\mathbf{y}$, α)
  $\mathbf{w} \leftarrow$ initialize to be any point in the parameter space
  $i = 0$, $j = 0$
  while not converged **do**
    $\mathbf{B} \leftarrow$ create batches of size m from $\mathbf{X}$ where $|\mathbf{B}^{(i)}| = m$ training records
    for $j = 1$ to $|\mathbf{B}|$ batches **do**
      for $k = 1$ to p **do**
        $$w_k = w_k - \alpha \frac{\partial}{\partial w_k} \left[\mathcal{L}^{(j)} \left(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{x}^{(j)}) \right) \right]$$
      $$b = b - \alpha \frac{\partial}{\partial b} \left[\mathcal{L}^{(j)} \left(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{x}^{(j)}) \right) \right]$$
      $i \leftarrow i + 1$
    Update convergence values

Mini-batch Gradient Descent (MBGD)

- i – the current iteration number
- j – the current batch
- $\mathbf{B}$ – the collection of batches as $\mathbf{B}^{(1)}$, $\mathbf{B}^{(2)}$, ..., $\mathbf{B}^{(|\mathbf{B}|)}$ where $|\mathbf{B}|$ is the total number of batches.
- m – the number of records per batch
- b – the bias weight
- $\mathbf{w}$ – the vector of weights
- $\mathbf{X}$ – the matrix of training data
- p – the number of features
- $\mathbf{y}$ – the vector of responses
- $\mathcal{L}(\cdot)$ – the loss function

Topics

Algorithm Termination

- As the algorithms run, we need to determine when it is appropriate to terminate. We establish some **convergence criteria** for this.
- One approach is to calculate the percentage of change between iterations: $\Delta_{\%\text{cost}}$
- We then need to establish what is the threshold ϵ that the percentage of change must exceed to continue to justify adjusting the weights and iterating, thus we establish the following must remain true to continue:

$$\Delta_{\%\text{cost}} \geq \epsilon$$

Convergence Criteria

$$\Delta_{\%\ \text{cost}} = \frac{\mid \mathcal{L}_{i-1}\big(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{X})\big) - \mathcal{L}_{i}\big(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{X})\big) \mid \times 100}{\mathcal{L}_{i-1}\big(\mathbf{y}, h_{\mathbf{w},b}(\mathbf{X})\big)}$$

We will select the parameter ϵ and establish convergence once $\Delta_{\%\ \text{cost}} < \epsilon$.

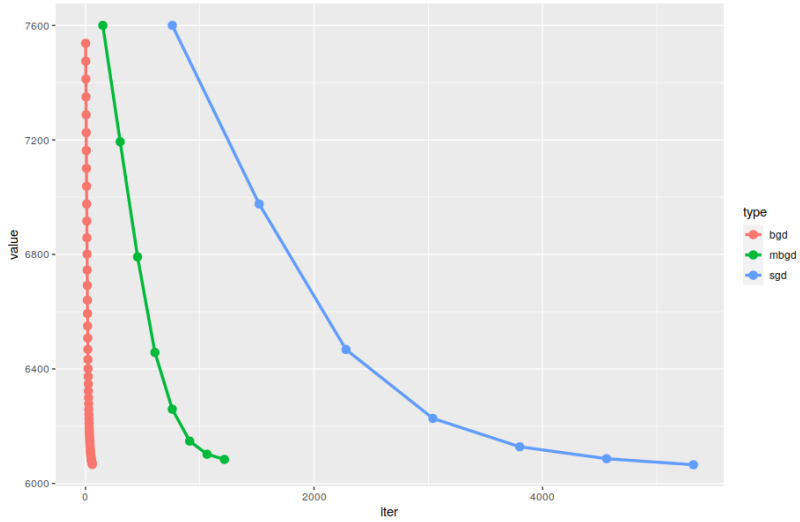
Convergence Criteria (cont.)

- $\mathcal{L}_i(\cdot)$ can be viewed as the loss function for the i^{th} iteration for N number of iterations.

$$\begin{bmatrix} \mathcal{L}_{i=1}(\cdot) & \mathcal{L}_{i=2}(\cdot) & \cdots & \mathcal{L}_{i=k}(\cdot) & \cdots & \mathcal{L}_{i=N-1}(\cdot) & \mathcal{L}_{i=N}(\cdot) \end{bmatrix}$$

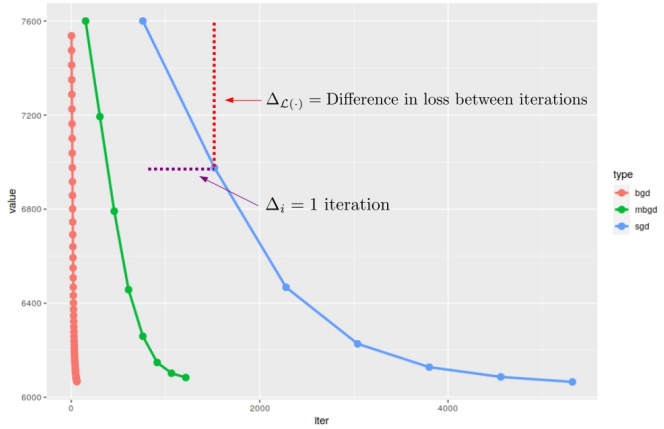
- $\mathcal{L}_i(\cdot)$ simply represents that the loss function we choose is calculated with $\mathbf{y}$ and $h_{\mathbf{w},b}(\mathbf{X})$.
  - $\mathbf{y}$ represents the true value for an observation
  - $\mathbf{w}$ and b represent the weights and bias nodes the model will use (some notations will represent b as w_0)
  - $\mathbf{X}$ represents the input (exercise caution as some notations will require all data to be passed into the loss function for calculation in matrix form $\mathbf{X}$ whereas some loss functions only require an input vector/observation, generally denoted as $\mathbf{x}$ or $\mathbf{x}^{(i)}$)
  - $h_{\mathbf{w},b}(\mathbf{X})$ is sometimes denoted as $\hat{\mathbf{y}}$

Loss Function Example



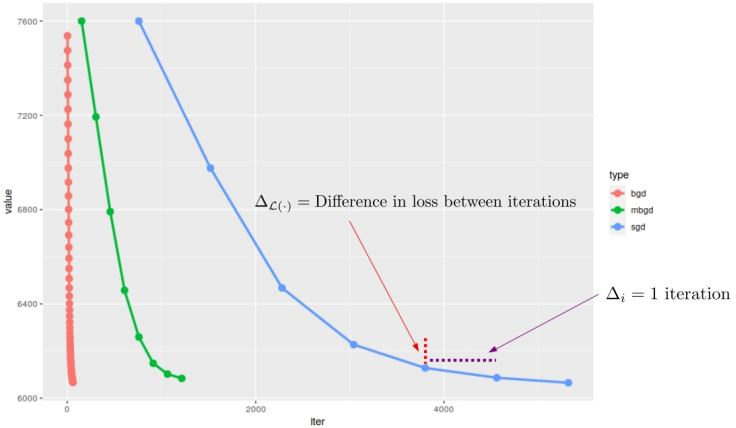
Loss Function Example

The loss function values will continue to decline as you adjust the weights.



Loss Function Example

Eventually, the weight adjustments will lead to diminishing reductions in the loss function. Once the percentage of overall difference in the loss function per iteration reaches ϵ, we use this as an early stopping criterion and terminate.



Topics

Regularization

Regularization is the process of explicitly penalizing complex models (or hypotheses).

This is similar to performing *feature selection* to remove attributes that do not add value or relevance to a model.

Regularization (cont.)

Regularization is a technique the helps algorithms avoid *overfitting*. It serves as a penalty term. We often measure this in the following form:

$$\text{Cost} = \text{Loss} + \lambda \cdot \text{Complexity}$$

- λ – a constant parameter we select for our model
- Complexity – we define some function to be added to our loss function that increases the cost with more complex models.
  1. L_1 regularization:

$$L_1(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_i |w_i|^1$$

  2. L_2 regularization:

$$L_2(\mathbf{w}) = ||\mathbf{w}||_2 = \mathbf{w}^T\mathbf{w} = \sum_i w_i^2$$

Regularization (cont.)

Suppose our loss function $\mathcal{L}(\mathbf{y}, h_{\mathbf{w}}(\mathbf{X}))$ is defined as follows:

$$\mathcal{L}(\mathbf{y}, h_{\mathbf{w}}(\mathbf{X})) = (\mathbf{y} - h_{\mathbf{w}}(\mathbf{X}))^T (\mathbf{y} - h_{\mathbf{w}}(\mathbf{X})) = \sum_i^n \left(y_i - h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^2$$

Thus, our loss function is:

$$\mathcal{L}(\mathbf{y}, h_{\mathbf{w}}(\mathbf{X})) = \sum_i^n \left(y_i - h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^2$$

We want to balance between an accurate solution and the complexity of the solution, we can employ the use of a q regularization penalty (i.e. L_1 or L_2) with some constant λ.

$$\mathcal{L}(\mathbf{y}, h_{\mathbf{w}}(\mathbf{X})) = \sum_i^n \left(y_i - h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^2 + \lambda \cdot L_q(\mathbf{w})$$

Regularization (cont.)

We commonly set q to be 1 or 2. For example, if we select $q = 2$, we will have the $L_2(\mathbf{w})$ cost added to our loss function:

$$\mathcal{L}(\,\mathbf{y}, h_{\mathbf{w}}(\mathbf{X})\,) = \sum_{i}^{n} \left(\, y_i - h_{\mathbf{w}}(\mathbf{x}^{(i)})\,\right)^2 + \lambda \cdot \sum_{k}^{p} w_k^2$$

- This penalty term biases the weight vector $\mathbf{w}$ to be closer to the origin.
- λ is the **regularization parameter** that trades off minimizing the loss on the training set and the magnitude of the parameters $\mathbf{w}$.
- When the magnitude of the parameter values become relatively large, this is generally an indication of overfitting.