

Recitation 2

Gradient Descent and Stochastic Gradient Descent

Yanlai Yang

CDS

February 15, 2023

Agenda

- Gradient Descent
 - Adaptive Learning Rate
- Stochastic Gradient Descent
- Application
 - Linear Regression
 - Logistic Regression

Gradient Descent Recap

Gradient Descent

- Initialize $x = 0$
- Repeat:
 - $x \leftarrow x - \underbrace{\eta}_{\text{step size}} \nabla f(x)$
- Until stopping criterion satisfied

Gradient Descent Recap

Gradient Descent

- Initialize $x = 0$

- Repeat:

- $x \leftarrow x - \underbrace{\eta}_{\text{step size}} \nabla f(x)$

- Until stopping criterion satisfied

- Choosing the step size is the key in gradient descent
- A fixed step size will work, eventually, as long as it's small enough

Gradient Descent

Gradient Descent Algorithm

- Goal: find $\theta^* = \arg \min_{\theta} J(\theta)$
- $\theta^0 :=$ [initial condition]
- $i := 0$
- while not [termination condition]:
 - compute $\nabla J(\theta_i)$
 - $\epsilon_i :=$ [choose learning rate at iteration i]
 - $\theta^{i+1} := \theta^i - \epsilon_i \nabla J(\theta_i)$
 - $i = i + 1$
- return θ^i

Gradient Descent

- How to initialize θ^0 ?

Gradient Descent

- How to initialize θ^0 ?
 - sample from some distribution
 - compose θ^0 using some heuristics

Gradient Descent

- How to initialize θ^0 ?
 - sample from some distribution
 - compose θ^0 using some heuristics
- How to choose termination conditions?

Gradient Descent

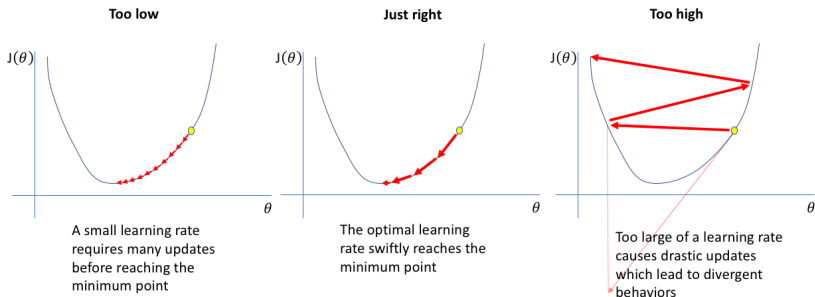
- How to initialize θ^0 ?
 - sample from some distribution
 - compose θ^0 using some heuristics
- How to choose termination conditions?
 - run for a fixed number of iteration
 - the value of $f(\theta)$ stabilizes
 - θ^i converges

Gradient Descent

- How to initialize θ^0 ?
 - sample from some distribution
 - compose θ^0 using some heuristics
- How to choose termination conditions?
 - run for a fixed number of iteration
 - the value of $f(\theta)$ stabilizes
 - θ^i converges
- What is a good learning rate?

Gradient Descent

- How to initialize θ^0 ?
 - sample from some distribution
 - compose θ^0 using some heuristics
- How to choose termination conditions?
 - run for a fixed number of iteration
 - the value of $f(\theta)$ stabilizes
 - θ^i converges
- What is a good learning rate?



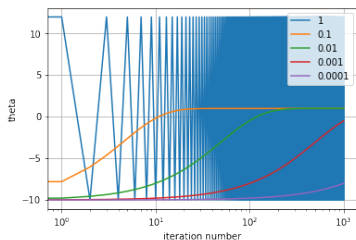
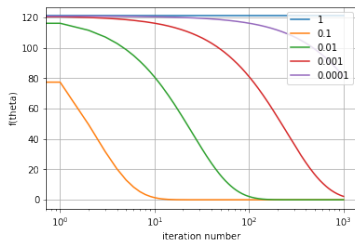
Learning Rate

Application

Suppose we would like to find $\theta^* \in \mathbb{R}$ that minimizes $f(\theta) = \theta^2 - 2\theta + 1$. The gradient (in this case, the derivative) $\nabla f(\theta) = 2\theta - 2$. We can easily see that $\theta^* = \operatorname{argmin}_{\theta} f(\theta) = 1$.

Learning Rate

- We applied gradient descent for 1000 iterations on $f(\theta) = \theta^2 - 2\theta + 1$ with varying learning rate $\epsilon \in \{1, 0.1, 0.01, 0.001, 0.0001\}$
- When the learning rate is too large ($\epsilon = 1$), $f(\theta)$ does not decrease through iterations. The value of θ_i at each iteration significantly fluctuates.
- When the learning rate is too small ($\epsilon = 0.0001$), $f(\theta)$ decreases very slowly.



Adaptive Learning Rate

- Instead of using a fixed learning rate through all iterations, we can adjust our learning rate in each iteration using a simple algorithm.
- At each iteration i :
 - $\tilde{\theta} := \theta_{i-1} - \epsilon_{i-1} \nabla f(\theta_{i-1})$
 - $\delta := f(\theta_{i-1}) - f(\tilde{\theta})$
 - if $\delta \geq \text{threshold}$:
 - we achieve a satisfactory reduction on $f(\theta)$
 - $\theta_i = \tilde{\theta}$
 - maybe we can consider increasing the learning rate for next iteration
 $\epsilon_j := 2\epsilon_{i-1}$
 - else:
 - the reduction is unsatisfactory
 - $\theta_i = \theta_{i-1}$
 - the learning rate is too large, so we reduce the learning rate
 - $\epsilon_i := \frac{1}{2}\epsilon_{i-1}$

Adaptive Learning Rate

How to decide a proper threshold for $f(\theta_{i-1}) - f(\tilde{\theta})$

Armijo rule

If learning rate ϵ satisfies

$$f(\theta_{i-1}) - f(\tilde{\theta}) \geq \frac{1}{2}\epsilon \|\nabla f(\theta_{i-1})\|^2$$

then $f(\theta)$ is guaranteed to converge to a (local) minimum under certain technical assumptions.

You can find more details at **this link**

Stochastic Gradient Descent

Stochastic Gradient Descent

- Initialize $w = 0$
 - Repeat:
 - randomly choose training point $(x_i, y_i) \in \mathcal{D}_n$
 - $w \leftarrow w - \eta \underbrace{\nabla_w \ell(f_w(x_i), y_i)}_{\text{Grad(Loss on i'th example)}}$
 - Until stopping criterion satisfied
-
- Equivalent to Minibatch Gradient Descent with batch size $N = 1$.
 - Use a single randomly chosen point to determine step direction.

Minibatch Gradient Descent

Minibatch Gradient Descent (minibatch size N)

- Initialize $w = 0$
 - Repeat:
 - randomly choose N points $\{(x_i, y_i)\}_{i=1}^N \subset \mathcal{D}_n$
 - $w \leftarrow w - \eta \left[\frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_i), y_i) \right]$
 - Until stopping criterion satisfied
-
- Minibatch gradient is an unbiased estimate of full-batch gradient:
$$\mathbb{E} \left[\nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$
 - Use a random subset of size N to determine step direction
 - Bigger N: Better estimate of the gradient, but slower (more data to touch)
 - Smaller N: Worse estimate of the gradient, but faster

Gradient Descent for Linear Regression

Linear Least Squares Regression Setup

- **Data:** Inputs are feature vectors of dimension d . Outputs are continuous scalars.

$$\mathcal{D} = \left\{ \mathbf{x}^{(i)}, y^{(i)} \right\}_{i=1}^n \quad \text{where } \mathbf{x} \in \mathbb{R}^d \text{ and } y \in \mathbb{R}$$

- **Hypothesis Space:** $\mathcal{F} = \{ f : \mathbb{R}^d \rightarrow \mathbb{R} \mid f(\mathbf{x}) = \theta^T \mathbf{x}, \theta \in \mathbb{R}^d \}$
- **Action:** Our prediction is a linear function of the inputs

$$\begin{aligned} \hat{y} &= f_{\theta}(\mathbf{x}) = \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d \\ \hat{y} &= f_{\theta}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} \end{aligned} \quad (\text{We assume } x_1 \text{ is } 1)$$

- **Loss:** $\ell(\hat{y}, y) = (y - \hat{y})^2$

Gradient Descent for Linear Regression

- **Goal:** Finding the set of parameters that minimize the empirical risk:

$$\hat{R}_n(\theta) = \frac{1}{n} \sum_{i=1}^n \left(\theta^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

where $\theta \in R^d$ parameterizes the hypothesis space \mathcal{F}

- Set our cost function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^n \left(\theta^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

Three Approach to solving $\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$

- **Approach 1:** Closed Form Solution (set derivatives equal to zero and solve for parameters)
 - pros: one shot algorithm!
 - cons: does not scale to large datasets (matrix inverse is bottleneck)
- **Approach 2:** Gradient Descent (take larger, more certain steps toward the negative gradient)
 - pros: conceptually simple, guaranteed convergence
 - cons: batch, often slow to converge
- **Approach 3:** Stochastic Gradient Descent (take many small, quick steps opposite the gradient)
 - pros: memory efficient, fast convergence, less prone to local optima
 - cons: convergence in practice requires tuning and fancier variants

Approach 1: Close-Form Solution

- Transform the cost function in matrix form

$$\begin{aligned} J(\theta) &= \frac{1}{2} \sum_{i=1}^n (\mathbf{x}_i^T \theta - y_i)^2 \\ &= \frac{1}{2} (\mathbf{X}\theta - \bar{y})^T (\mathbf{X}\theta - \bar{y}) = \frac{1}{2} \|\mathbf{X}\theta - y\|_2^2 \end{aligned}$$

- To minimize $J(\theta)$, take derivative and set to zero:

$$\begin{aligned} \nabla J(\theta) &= (\mathbf{X}^T \mathbf{X} \theta - \mathbf{X}^T y) = \mathbf{X}^T (\mathbf{X} \theta - y) = 0 \\ \Rightarrow \hat{\theta} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y \end{aligned}$$

- Ensure invertibility of $\mathbf{X}^T \mathbf{X}$
- What if \mathbf{X} has less than full column rank?

Approach 2: Iterative Method GD

Gradient Descent Algorithm

- $\theta^0 :=$ [initial condition]
- $i := 0$
- while not [termination condition]:
 - $\theta^{i+1} := \theta^i - \epsilon_i \nabla_{\theta} J(\theta)$
 - $i = i + 1$
- return θ^i

Recall:

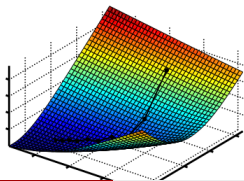
$$\nabla_{\theta} J(\theta) = \begin{bmatrix} \frac{d}{d\theta_1} J(\theta) \\ \frac{d}{d\theta_2} J(\theta) \\ \vdots \\ \frac{d}{d\theta_d} J(\theta) \end{bmatrix}$$

Approach 3: Iterative Method SGD

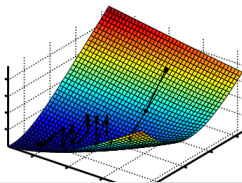
Stochastic Gradient Descent Algorithm

- $\theta^0 :=$ [initial condition]
- $i := 0$
- while not [termination condition]:
 - For each training pair (x^j, y^j) (in random order)
 - $\theta^{i+1} := \theta^i - \epsilon_i \nabla_{\theta} J^{(j)}(\theta)$ {with $J^{(j)}(\theta) = \frac{1}{2} (\theta^T \mathbf{x}^{(j)} - y^{(j)})^2$ }
 - $i = i + 1$
- return θ^i

GD



SGD



Gradient Descent for Logistic Regression

Binary Classification Setup for Logistic Regression

- **Data:** Inputs are feature vectors of dimension d . Targets are class labels. $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^n$ where $\mathbf{x} \in \mathbb{R}^d$ and $y \in \{0, 1\}$
- **Action:** Our prediction is the probability of class label given linear signals

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad \text{with} \quad g(z) = \frac{1}{1+e^{-z}}$$

- Sigmoid Function $g(z)$: takes a real-valued number and maps it into the range $[0,1]$ (Probability Interpretation)
- Assume

$$\begin{cases} P(y = 1 \mid x; \theta) = h_{\theta}(x) \\ P(y = 0 \mid x; \theta) = 1 - h_{\theta}(x) \end{cases}$$

- More Compactly: $p(y \mid x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$

Logistic Regression

- Assumption: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ are independently generated
- Likelihood:** The probability of getting the y_1, \dots, y_n in \mathcal{D} from the corresponding x_1, \dots, x_n

$$\begin{aligned}
 P(y_1, \dots, y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n) &= \prod_{i=1}^n p(y^{(i)} \mid x^{(i)}; \theta) \\
 &= \prod_{i=1}^n (h_{\theta}(x^{(i)}))^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}}
 \end{aligned}$$

- Goal:** maximize the log likelihood (Easier)

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^n y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))$$

- Equivalent to minimize the objective function with logistic loss:

$$J(\theta) = \sum_{i=1}^n \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

$$\text{where } \ell(h(x), y) = -y \log(h_{\theta}(x)) - (1 - y) \log(1 - h_{\theta}(x))$$

Logistic Regression

- Analytic solution won't work
- Find optimum using iterative methods: Gradient Ascent or Stochastic Gradient Ascent
 - Gradient ascent rule: $\theta := \theta + \alpha \nabla_{\theta} \ell(\theta)$
 - Stochastic gradient ascent rule: $\theta := \theta + \alpha \nabla_{\theta} \ell^{(i)}(\theta)$ for random training pair (x^i, y^i)
- For one training example (x, y) , the partial derivative of log likelihood $\ell(\theta)$:

$$\begin{aligned}
 &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
 &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x) (1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\
 &= (y(1 - g(\theta^T x)) - (1 - y)g(\theta^T x)) x_j \\
 &= (y - h_{\theta}(x)) \cdot x_j
 \end{aligned}$$

Logistic Regression

References

- DS-GA 1003 Machine Learning Spring 2021 & 2022
- Stanford CS229 Note1
- CMU 10-701 Linear Regression Slide