Recitation 3

Gradient Descent

Artie Shen and Brett Bernstein

CDS

February 12, 2020

Announcement

- New TA office hour
- 5:30 pm - 6:30 pm, room 650, 60 5th Ave

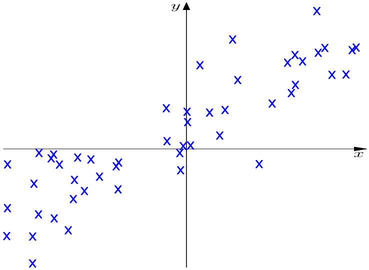
Agenda

- Motivation
- Calculating the gradient
- Gradient descent algorithm
- Coding exercise: gradient descent implementation

Intro Question

Question

We are given the data set $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We want to fit a **linear function** to this data by performing **empirical risk minimization**. More precisely, we are using the hypothesis space $\mathcal{F} = \{h_\theta(x) = \theta^T x \mid \theta \in \mathbb{R}^d\}$ and the loss function $\ell(a, y) = (a - y)^2$. Given an initial guess $\tilde{\theta}$ for the empirical risk minimizing parameter vector, how could we improve our guess?



Intro Solution

Solution

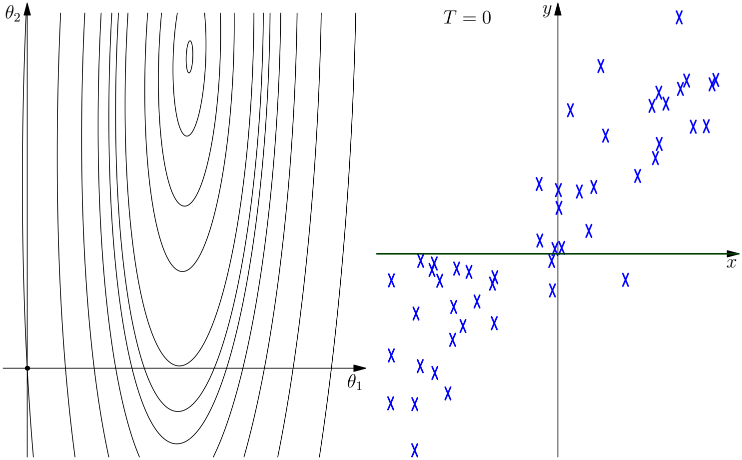
- The empirical risk is given by

$$J(\theta) := \hat{R}_n(h_\theta) = \frac{1}{n}\sum_{i=1}^{n}\ell(h_\theta(x_i), y_i) = \frac{1}{n}\sum_{i=1}^{n}(\theta^T x_i - y_i)^2 = \frac{1}{n}\|X\theta - y\|_2^2,$$

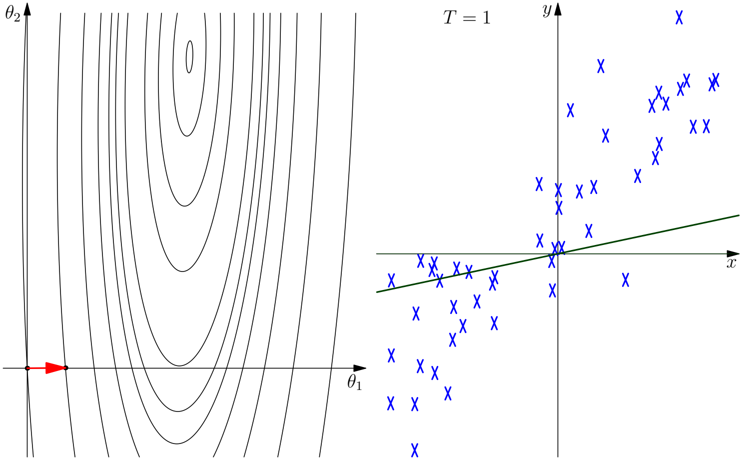
where $X \in \mathbb{R}^{n\times d}$ is the matrix whose ith row is given by x_i^T.

- Can improve a non-optimal guess $\tilde{\theta}$ by taking a small step in the direction of the negative gradient $-\nabla J(\theta)$.

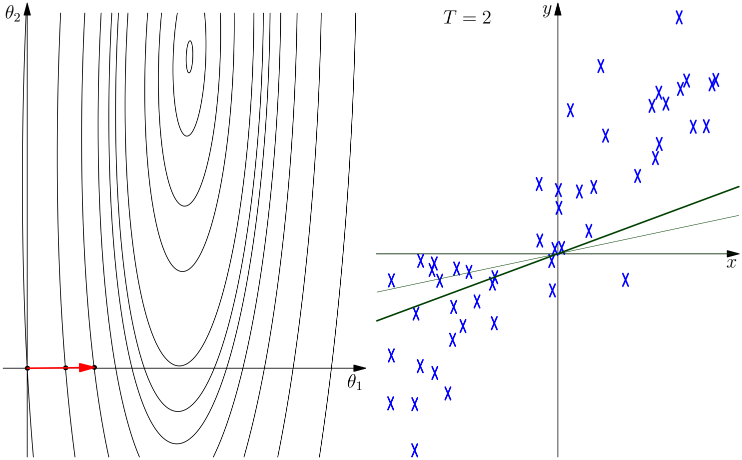
Negative Gradient Steps



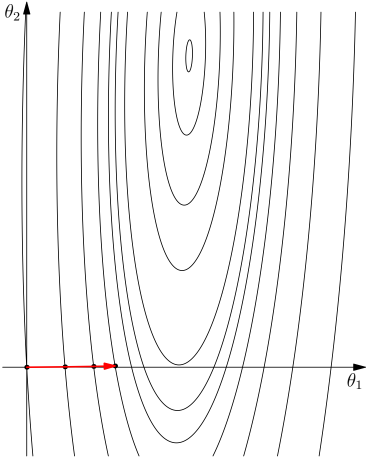
Negative Gradient Steps

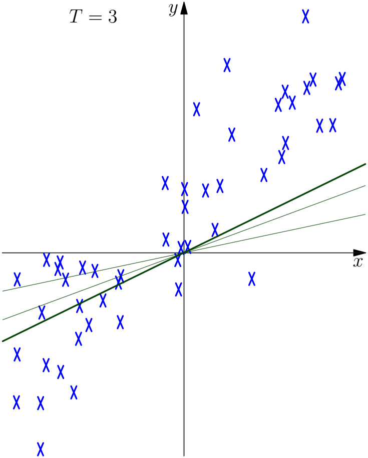


Negative Gradient Steps

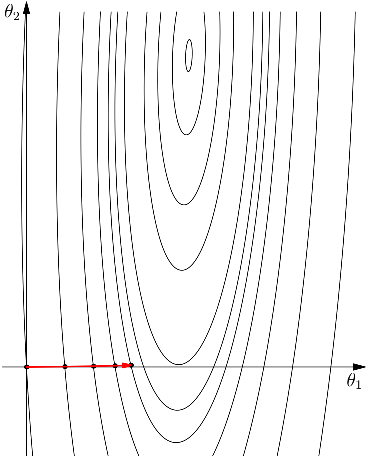


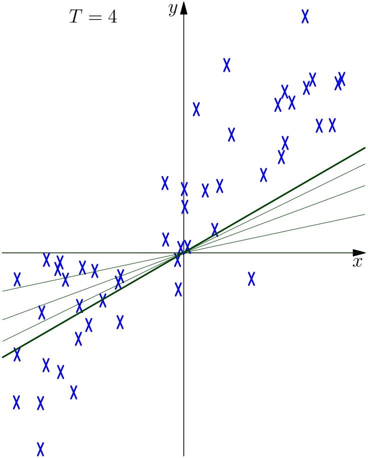
Negative Gradient Steps



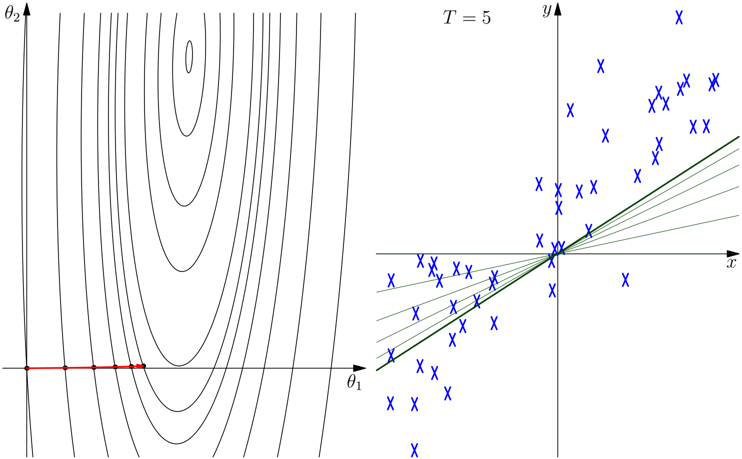


Negative Gradient Steps

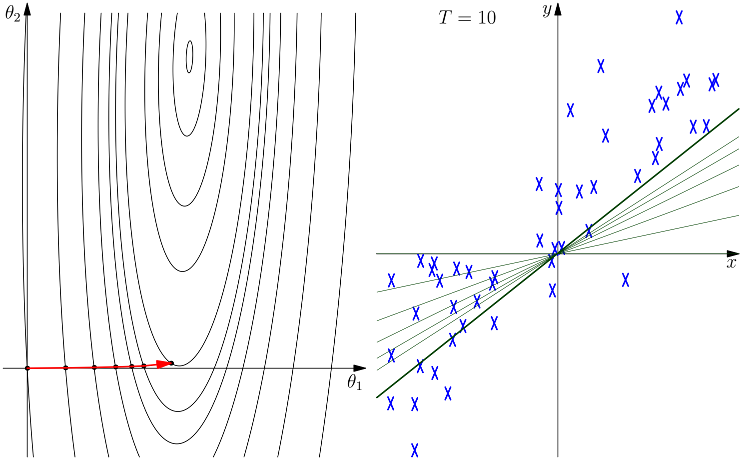




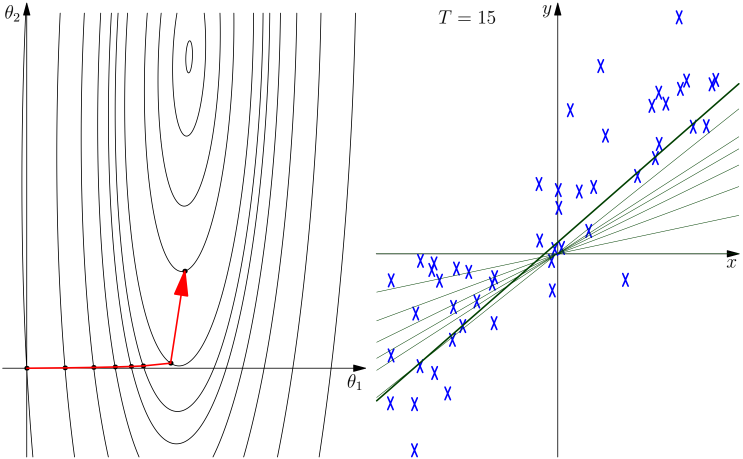
Negative Gradient Steps



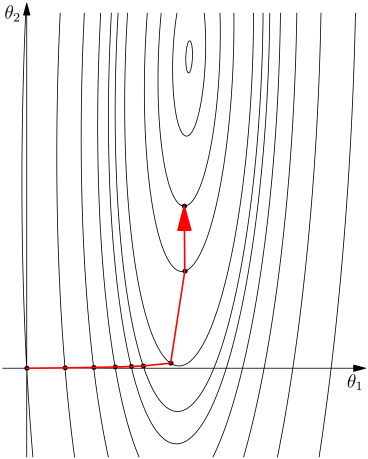
Negative Gradient Steps

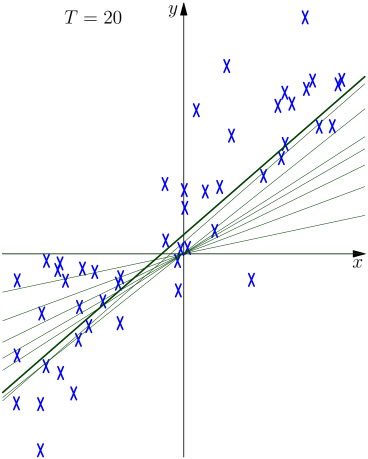


Negative Gradient Steps

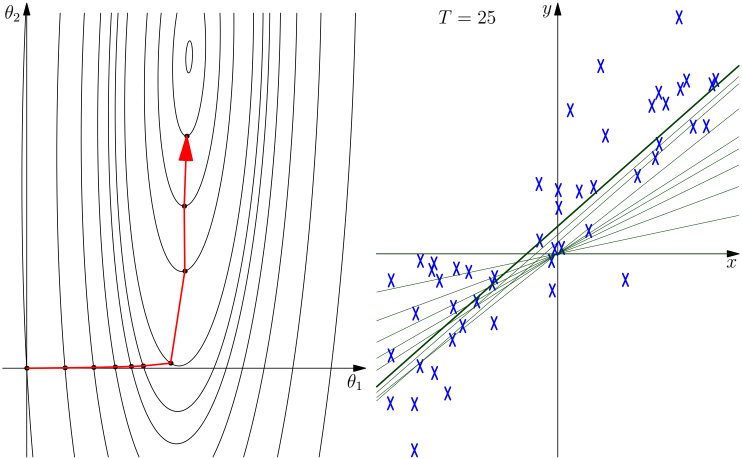


Negative Gradient Steps

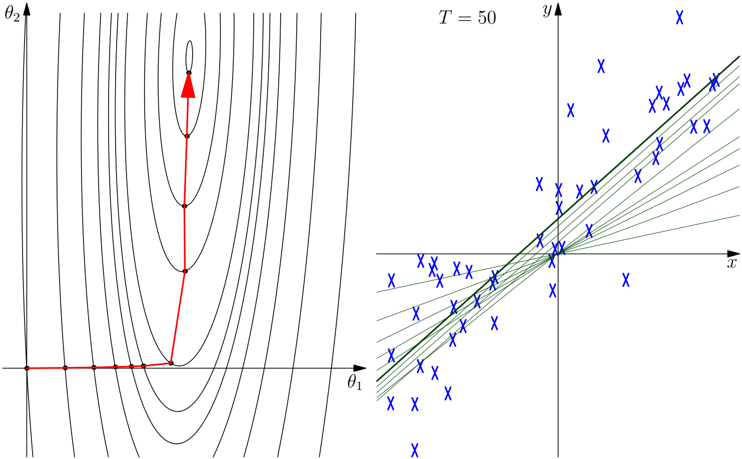




Negative Gradient Steps



Negative Gradient Steps



Intuition

- Why don't we directly compute the derivative $\frac{\partial}{\partial\theta}J(\theta, y)$ and solve for θ^* that makes $\frac{\partial}{\partial\theta}J(\theta^*, y) = 0$.
  - But what if the analytical solution for $\frac{\partial}{\partial\theta}J(\theta^*, y) = 0$ is computationally intractable?
- We can improve the an initial guess θ_0 by iteratively "updating" it using the gradient of the loss $\nabla J(\theta)$.
- This process will give us a "good enough" approximation for θ^*.
- To do so, we need to know how to calculate the gradient $J(\theta^*, y)$.

Directional Derivative

Definition

Let $f : \mathbb{R}^n \to \mathbb{R}$. The directional derivative $f'(x; u)$ of f at $x \in \mathbb{R}^n$ in the direction $u \in \mathbb{R}^n$ is given by

$$f'(x; u) = \lim_{h \to 0} \frac{f(x + hu) - f(x)}{h}.$$

- We say that u is a *descent direction* of f at x if $f'(x; u) < 0$.
- Taking a small enough step in a descent direction causes the function value decreases.

Partial Derivative

- Let $e_i = (\overbrace{0, 0, \ldots, 0}^{i-1}, 1, 0, \ldots, 0)$ denote the ith standard basis vector.
- The ith *partial derivative* is defined to be the directional derivative along e_i.
- It can be written many ways:

$$f'(x; e_i) = \frac{\partial}{\partial x_i} f(x) = \partial_{x_i} f(x) = \partial_i f(x).$$

- What is the intuitive meaning of $\partial_{x_i} f(x)$? For example, what does a large value of $\partial_{x_3} f(x)$ imply?

Differentiability and Gradients

- We say a function $f : \mathbb{R}^n \to \mathbb{R}$ is *differentiable* at $x \in \mathbb{R}^n$ if

$$\lim_{v \to 0} \frac{f(x+v) - f(x) - g^T v}{\|v\|_2} = 0,$$

  for some $g \in \mathbb{R}^n$.
- If it exists, this g is unique and is called the *gradient* of f at x with notation

$$g = \nabla f(x).$$

- It can be shown that

$$\nabla f(x) = \begin{pmatrix} \partial_{x_1} f(x) \\ \vdots \\ \partial_{x_n} f(x) \end{pmatrix}.$$

Computing Gradients

Question

Compute the gradient $\nabla f(\theta) \in \mathbb{R}^2$ for the following function:

$$f(\theta) = \theta_0^2 + 2\theta_0\theta_1^3$$

The gradient of this function can be computed as following:

$$\frac{\partial}{\partial \theta_0} = 2\theta_0 + 2\theta_1^3$$

$$\frac{\partial}{\partial \theta_1} = 6\theta_0\theta_1^2$$

$$\nabla f(\theta) = \begin{bmatrix} 2\theta_0 + 2\theta_1^3 \\ 6\theta_0\theta_1^2 \end{bmatrix}$$

Computing Gradients

Question

Compute the gradient $\nabla J(\theta) \in \mathbb{R}^d$ for

$$J(\theta) = \|X\theta - y\|_2^2$$

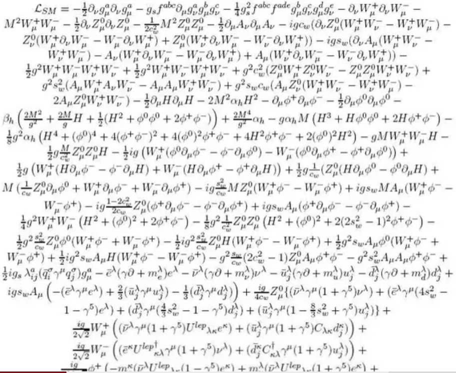
where $X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n, \theta \in \mathbb{R}^d$.

In homework2, we will show that

$$\nabla J(\theta) = 2(X^T X\theta - X^T y) = 2X^T(X\theta - y)$$

Gradient Checker

- So far we have worked with relatively simple functions where it is straight-forward to compute the gradient.
- For more complex functions, the gradient computation can be notoriously difficult to debug and get right. Example from `www.quora.com/What-is-the-most-complex-equation-in-the-world`.
- How can we test if our gradient computation is correct?



Gradient Checker

- Recall the mathematical definition of the derivative as:

$$\frac{\partial}{\partial \theta} f(\theta) = \lim_{\epsilon \to 0} \frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon}$$

- We can approximate the gradient (left hand side) using the equation on the right hand side by setting ϵ to a small constant, say $\epsilon = 10^{-4}$.
- Now let's expand this method to deal with vector input $\theta \in \mathbb{R}^d$. Let's say we want to verify out gradient at dimension i $(\nabla f(\theta))_i$. We can make use of one-hot vector e_i in which all dimension except the ith are 0 and the ith dimension has a value of 1: $e_i = [0, 0, ..., 1, ..., 0]^T$
- The gradient at ith dimension can be then approximated as

$$[\nabla f(\theta)]^{(i)} \approx \frac{f(\theta + \epsilon e_i) - f(\theta - \epsilon e_i)}{2\epsilon}$$

Recap

- To find a good decision function we will minimize the empirical loss on the training data.
- Having fixed a hypothesis space parameterized by θ, finding a good decision function means finding a good θ.
- Given a current guess for θ, we will use the gradient of the empirical loss (w.r.t. θ) to get a local linear approximation.
- If the gradient is non-zero, taking a small step in the direction of the negative gradient is guaranteed to decrease the empirical loss.
- This motivates the minimization algorithm called gradient descent.

Gradient Descent

Gradient descent Algorithm

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

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?

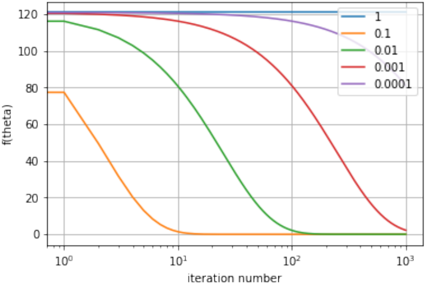
Learning Rate

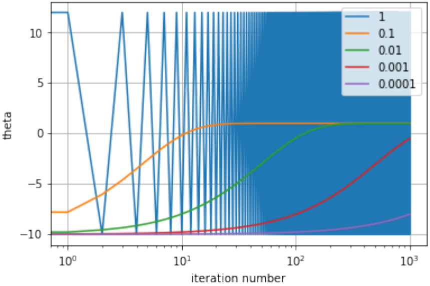
Application

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 $\theta^* = \text{argmin}_\theta f(\theta) = 1$.

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 $\delta \geq$ threshold:
    - we achieve a satisfactory reduction on $f(\theta)$
    - $\theta_i = \tilde{\theta}$
    - maybe we can consider increasing the learning rate for next iteration $\epsilon_i := 2\epsilon_{i-1}$
  - 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

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

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

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

Coding Exercise

In the provided notebook, we will use Python to:

- calculate the gradient for two example functions
- implement the gradient checker
- implement the gradient descent algorithm with Armijo's rule
- apply gradient descent on the UCI Automobile dataset