Recitation 1

Statistical Learning Theory
Intro to Gradient Descent

Colin

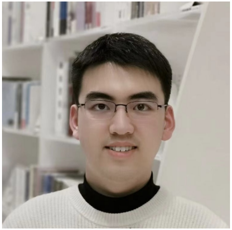
CDS

Jan. 25, 2023

Introduction

TAs for this course:
Colin Wan, Ying Wang, Yanlai Yang

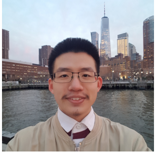


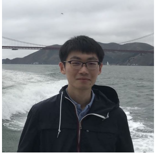


Office Hours:
Colin: Mon 5:00PM-6:00PM
Ying: Wed 6:00PM-7:00PM
Yanlai: Wed 1:00PM - 2:00PM

Introduction

Graders for this course:
Xiaojing Fan, Junze Li, Richard Lin, Ying Wang, Jerry Xue, Frances Yuan

Office Hours:
Lead Grader will host office hour the week after the grade of HWs are released

Logistics

- There will be 7 to 8 assignments and two tests
- Assignments will be released after each lab.
- You will have two weeks to complete the homework (except for hw1, which is only one week)
- The grades will be released after two weeks.
- All homeworks will be submitted through GradeScope. **DO NOT SUBMIT THROUGH BRIGHTSPACE.**

Logistics

- You are **strongly** encouraged to use LaTex, but we will accept scanned handwritten documents given the hand writing is eligible. **It is your responsibility to ensure it is in the correct orientation and matched to the correct questions.** Points will be takeoff otherwise.
- You will be able to submit regrade request on GradeScope after grades are released.
- When submitting regrade requests, clearly **express reasoning, cite supporting arguments**.
  - If you write "I think this deserves 2 points", we will disregard the regrade. Instead write something like "I proved XXX in line XX, showed XXX using XXX in line XX."

Motivation

In data science, we generally need to **Make a Decision** on a problem. To do this, we need to understand

- The setup of the problem
- The possible actions
- The effect of actions
- The evaluation of the results

How do we translate the problem into the language of DS/modeling?

Formalization

The Spaces

$\mathcal{X}$: input space $\mathcal{Y}$: outcome space $\mathcal{A}$: action space

Prediction Function

A **prediction function** f gets an input $x \in \mathcal{X}$ and produces an action $a \in \mathcal{A}$:

$$f : \mathcal{X} \mapsto \mathcal{A}$$

Loss Function

A **loss function** $\ell(a, y)$ evaluates an action $a \in \mathcal{A}$ in the context of an outcome $y \in \mathcal{Y}$:

$$\ell : \mathcal{A} \times \mathcal{Y} \mapsto \mathbb{R}$$

Risk Function

- Given a loss function ℓ, how can we evaluate the "average performance" of a prediction function f?
- To do so, we need to first assume that there is a **data generating distribution** $\mathcal{P}_{x,y}$.
- Then the expected loss of f on $\mathcal{P}_{x,y}$ will reflect the notion of "average preformance".

Definition

The **risk** of a prediction function $f : \mathcal{X} \mapsto \mathcal{A}$ is

$$R(f) = \mathbb{E}[\ell(f(x), y)]$$

It is the expected loss of f on a new sample (x, y) drawn from $\mathcal{P}_{X,Y}$.

Concepts of Learning

Types of Learning

- Supervised
- Unsupervised
- Semi-supervised

Processes of learning

- Modeling (setup)
- Learning (training)
- Inference (evaluation/understanding)

Finding 'best' function

Definition

$\mathfrak{F}$ is the family of functions we restrict our model to be.
Example: Linear, quadratic, decision tree, two layer neural-net...

Definition

$f_{\mathfrak{F}}$ is 'best' function one can obtain within $\mathfrak{F}$.

Definition

$\hat{f}_n$ is the 'best' function one can obtain using the data given.

Definition

$\tilde{f}_n$ is the function actually obtained using the data given.

The Bayes Prediction Function

Definition

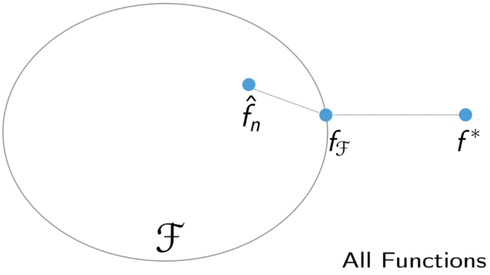
A **Bayes prediction function** $f^* : \mathcal{X} \mapsto \mathcal{Y}$ is a function that achieves the *minimal risk* among all possible functions:

$$f^* \in \arg\min_f R(f),$$

where the minimum is taken from all functions that maps from $\mathcal{X}$ to $\mathcal{A}$.

The risk of a Bayes function is called **Bayes risk**.

Error Decomposition



Example

Error Decomposition

- Approximation Error
  - Caused by the choice of family of functions or capacity of the model.
  - Expand the capacity of the model.
- Estimation Error
  - Caused by finite number of data
  - Obtain more data/add regularization
- Optimization Error
  - Caused by not able to find the best parameters
  - Try different optimization algorithms, learning rates, etc.

Gradient Descent

Motivation:

- Our goal is the find $\hat{f}_n$, the best possible model from given data
- Naive approach: Take gradient of loss function, solve for parameters that gives you 0.
  - Computationally intractable
  - Impossible to compute due to complex function structure
- The optimal parameters for LR: $\hat{\beta} = \left(X^\top X\right)^{-1} X^\top Y$
- When X's dimension reaches the millions, the inverse is essentially intractable.

But we do not need $\hat{f}_n$, a close $\tilde{f}_n$ is good enough for decision making. Therefore, instead of solving for the best parameters, we just need to approximate it well enough.

Gradient Descent

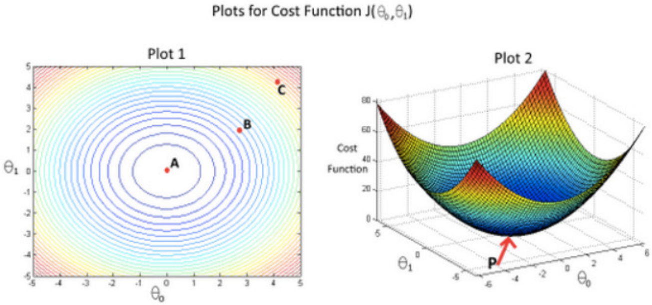
Idea:

- Given any starting parameters, the gradient indicates the direction of local maximal change.
- If we obtain new parameters by moving old parameter along its gradient, the new ones will give smaller loss (if we are careful).
- We can repeat this procedure until we are happy with the result.

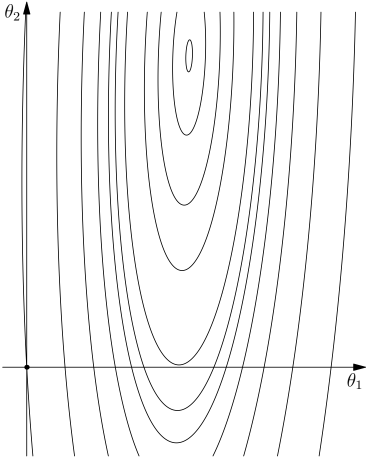
Contour Graphs

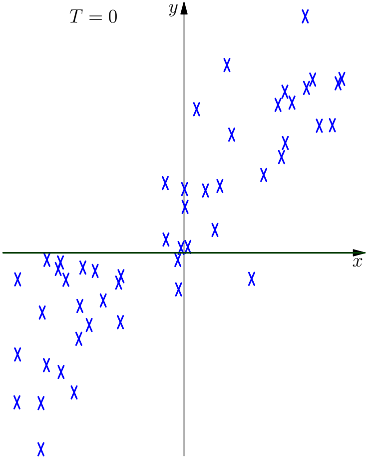
Imagine we are solving a simple linear regression problem: $y = \theta_0 + \theta_1 x$ with loss function:

$$J(\theta_0, \theta_1) = \sum^{n} (y_i - (\theta_0 + \theta_1 x_i))^2$$

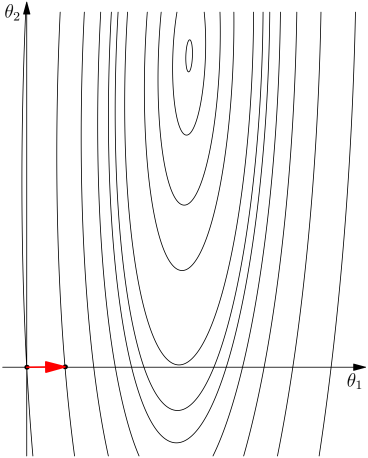


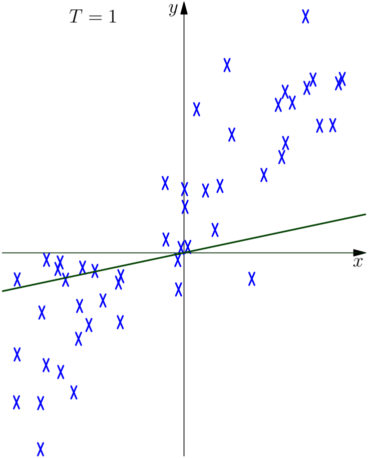
Negative Gradient Steps



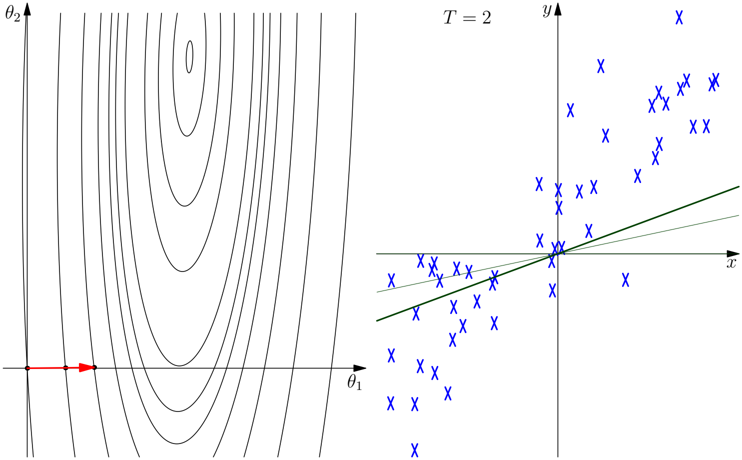


Negative Gradient Steps

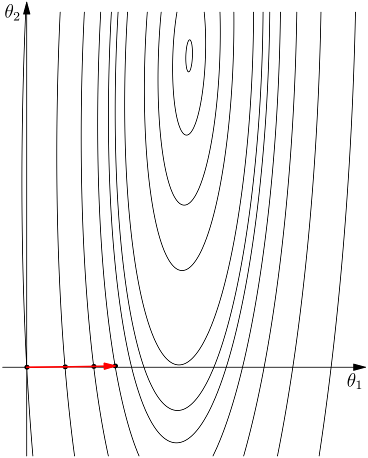


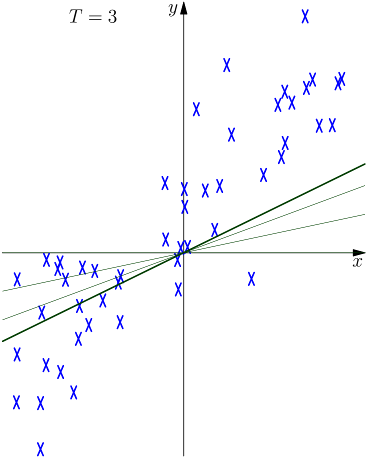


Negative Gradient Steps

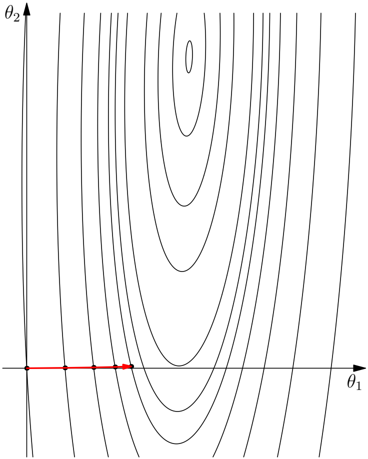


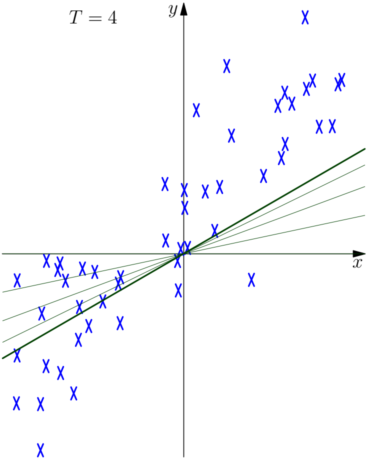
Negative Gradient Steps



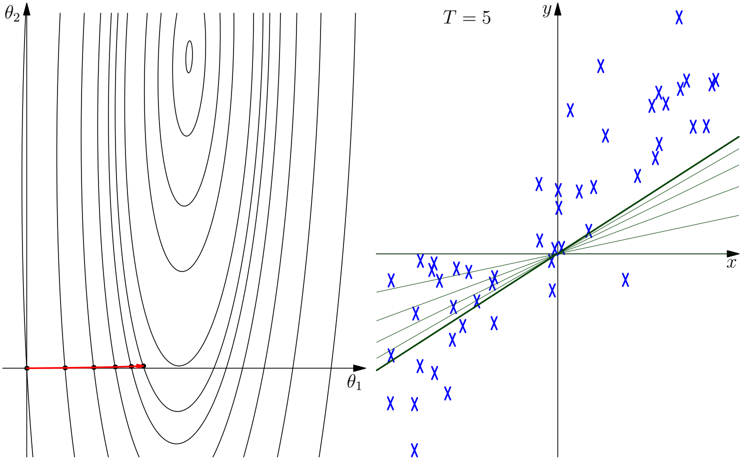


Negative Gradient Steps

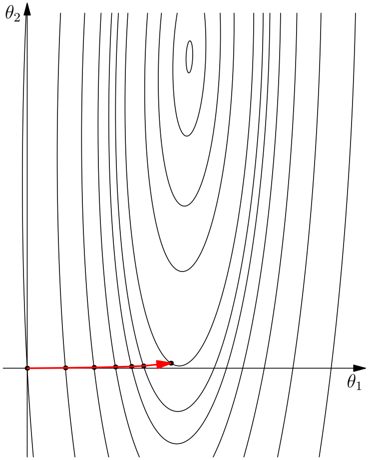


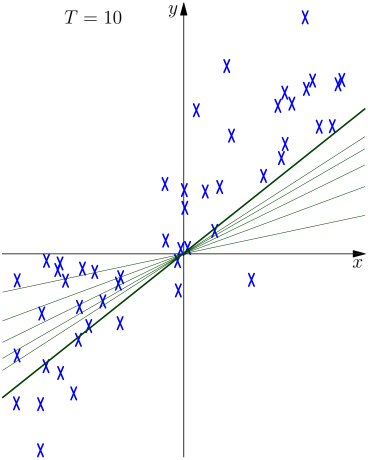


Negative Gradient Steps

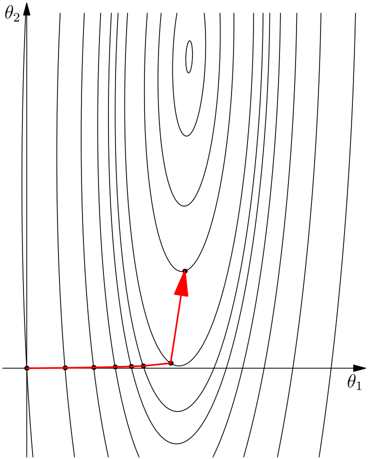


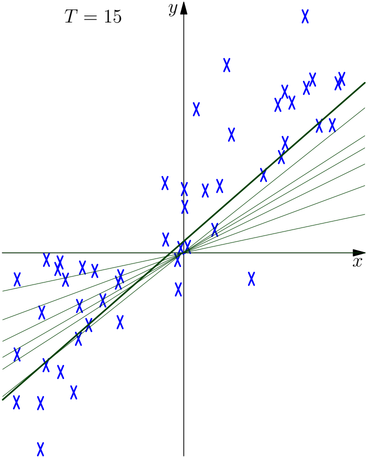
Negative Gradient Steps



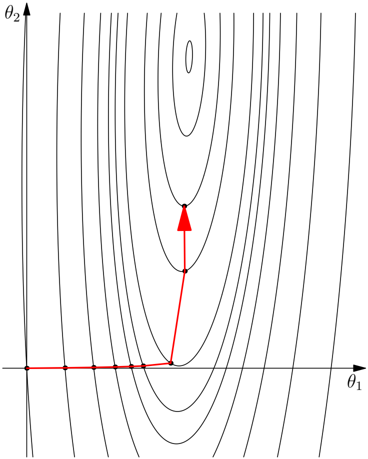


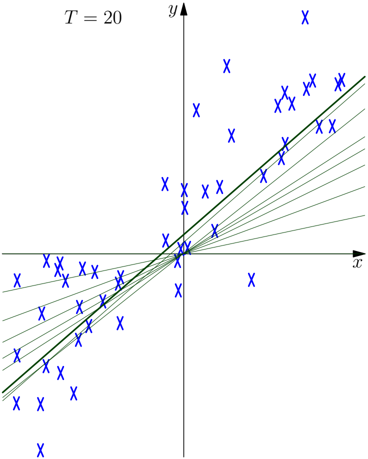
Negative Gradient Steps



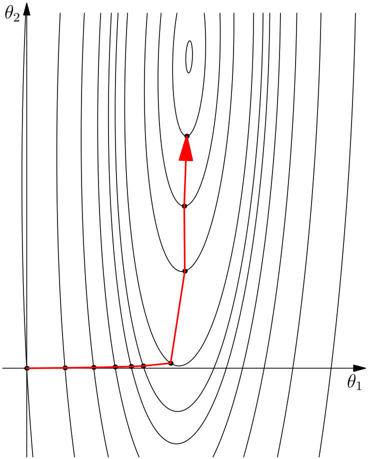


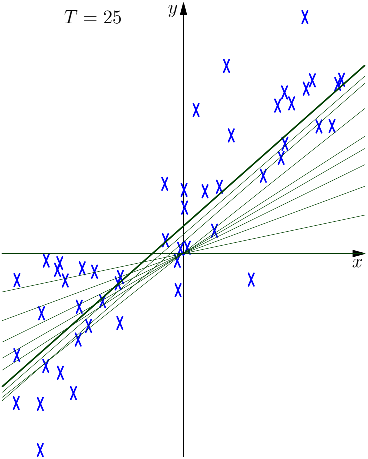
Negative Gradient Steps



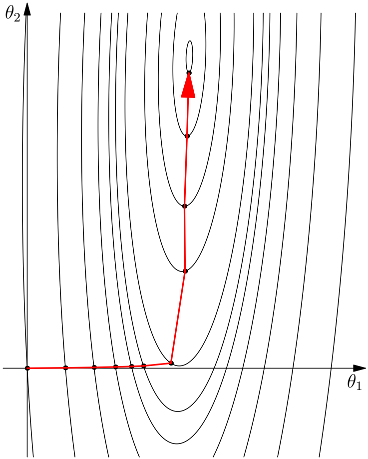


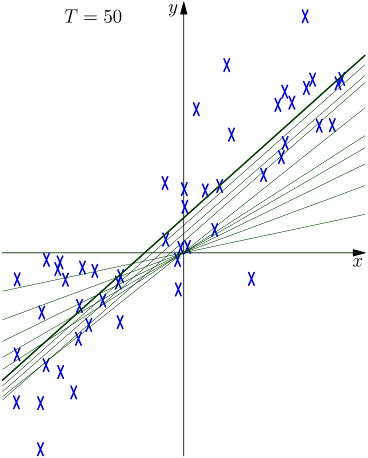
Negative Gradient Steps





Negative Gradient Steps





Gradient Descent

Gradient descent Algorithm

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

Things to review

- Calculus
  - Gradients, taking (partial) derivatives
- Linear Algebra
  - Matrix computation, matrix derivatives
  - Example: compute $\frac{\partial x^T A x}{\partial x}$, where A is a matrix and x is a vector