CS6787: Advanced Machine Learning Systems

CS6787 Lecture 1 — Fall 2019
What’s missing in the basic stuff?

Efficiency!
Motivation:
Machine learning applications involve large amounts of data

More data $\rightarrow$ Better services

Better systems $\rightarrow$ More data
How do practitioners make their systems better?
How do we improve our systems?

• Build frameworks & software that make it easy to express, train, and evaluate a machine learning model.

• Use methods for accelerating convergence of learning algorithms — learn in fewer iterations.

• Automatically configure learning systems by using hyperparameter optimization

• Use methods for improving hardware efficiency — run each iteration faster.

• Use specialized ML hardware accelerators, and get our frameworks to do as much as possible automatically.

Course outline

Part 1

Part 2

Part 3

Part 4

Part 5
Course Format

One half

Traditional lectures
Broad description of techniques

One half

Important papers
Presentations by you
In-class discussions
Reviews of each paper
Prerequisites

• Basic ML knowledge (CS 4780)

• Math/statistics knowledge
  • At the level of the entrance exam for CS 4780
Grading

• Paper presentations
• Discussion participation
• Paper reviews
• Programming assignments
• Final project
Paper presentations

• Papers listed on the website
  • 20-minute presentation slot for each paper
  • Presenting in groups of two-to-three

• **Signups by Friday!**

• Learning goal
  • Practice digesting and talking about other people’s work
Discussion and Paper Reviews

• Each paper presentation will be followed by questions and breakout discussion

• Please read at least one of the papers before coming to class
  • And at least skim the other paper, so you know what to expect

• For each class period, submit a review of one of the two papers
  • Detailed instructions are on the course webpage
  • Learning goal: practice how to deeply read and critique a paper in context, and get some window into how peer review works.
Programming Assignments

• **New** by popular demand

• Two assignments in the first part of the semester only

• Learning goal: **become familiar with ML frameworks/tools**
  • …and the principles that underlie them
  • This will hopefully build skills for you to use in the final project
Final Project

• **Open-ended**: work on what you think is interesting!
  • Learning goal: *do a small bit of non-trivial research on your own*

• Groups of up to three

• Your proposed project must include:
  • The **implementation** of a machine learning system for some task
  • Exploring one or more of the *techniques discussed in the course*
  • To **empirically evaluate performance** and compare with a baseline.
Late Policy

• This is a graduate level course

• Two free late days for each of the paper reviews and programming assignments

• No late days on the final project
  • To make things easy on the graders

• No late days on the presentations (for obvious reasons)
Questions?
Today’s Topic

Stochastic Gradient Descent: The Workhorse of Machine Learning

CS6787 Lecture 1 — Fall 2019
But first... an icebreaker activity!

For each person in order:

- What is your name?
- What are you studying?
- What do you hope to learn from CS6787?

After everyone is done, discuss together:

Why do people use stochastic gradient descent?
Today’s Topic

Stochastic Gradient Descent: The Workhorse of Machine Learning

CS6787 Lecture 1 — Fall 2019
Optimization

• Much of machine learning can be written as an optimization problem

\[
\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f(w; x_i)
\]

• Example loss functions: logistic regression, linear regression, principle component analysis, neural network loss, empirical risk minimization
Types of Optimization

• Convex optimization
  • The easy case
  • Includes logistic regression, linear regression, SVM

• Non-convex optimization
  • NP-hard in general
  • Includes deep learning

A good strategy:
Build intuition about techniques from the convex case where we can prove things…
…and apply it to better understand more complicated systems.
An Abridged Introduction to Convex Functions
Convex Functions

∀α ∈ [0, 1], \( f(αx + (1 - α)y) \leq αf(x) + (1 - α)f(y) \)

\( f(x) = x^2 \)
Example: Quadratic

\[ f(x) = x^2 \]

\[
(\alpha x + (1 - \alpha)y)^2 = \alpha^2 x^2 + 2\alpha(1 - \alpha)xy + (1 - \alpha)^2 y^2 \\
= \alpha x^2 + (1 - \alpha)y^2 - \alpha(1 - \alpha)(x^2 + 2xy + y^2) \\
\leq \alpha x^2 + (1 - \alpha)y^2
\]
Example: Abs

\[ f(x) = |x| \]

\[ |\alpha x + (1 - \alpha)y| \leq |\alpha x| + |(1 - \alpha)y| \]
\[ = \alpha |x| + (1 - \alpha)|y| \]
Example: Exponential

\[ f(x) = e^x \]

\[ e^{\alpha x + (1-\alpha) y} = e^y e^{\alpha (x-y)} = e^y \sum_{n=0}^{\infty} \frac{1}{n!} \alpha^n (x - y)^n \]

\[ \leq e^y \left( 1 + \alpha \sum_{n=1}^{\infty} \frac{1}{n!} (x - y)^n \right) \quad \text{(if } x > y) \]

\[ = e^y \left( (1 - \alpha) + \alpha e^{x-y} \right) \]

\[ = (1 - \alpha)e^y + \alpha e^x \]
Properties of convex functions

• Any line segment we draw between two points lies above the curve

• Corollary: every local minimum is a global minimum
  • Why?

• This is what makes convex optimization easy
  • It suffices to find a local minimum, because we know it will be global
Properties of convex functions (continued)

• Non-negative combinations of convex functions are convex

\[ h(x) = af(x) + bg(x) \]

• Affine scalings of convex functions are convex

\[ h(x) = f(Ax + b) \]

• Compositions of convex functions are **NOT** generally convex
  • Neural nets are like this

\[ h(x) = f(g(x)) \]
Convex Functions: Alternative Definitions

• First-order condition

\[ \langle x - y, \nabla f(x) - \nabla f(y) \rangle \geq 0 \]

• Second-order condition

\[ \nabla^2 f(x) \succeq 0 \]

• This means that the matrix of second derivatives is positive semidefinite

\[ A \succeq 0 \iff \forall x, \langle x, Ax \rangle \geq 0 \]
Example: Quadratic

\[ f(x) = x^2 \]

\[ f''(x) = 2 \geq 0 \]
Example: Exponential

\[ f(x) = e^x \]

\[ f''(x) = e^x \geq 0 \]
Example: Logistic Loss

\[ f(x) = \log(1 + e^x) \]

\[ f'(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}} \]

\[ f''(x) = -\frac{-e^{-x}}{(1 + e^{-x})^2} = \frac{1}{(1 + e^x)(1 + e^{-x})} \geq 0. \]
Strongly Convex Functions

• Basically the easiest class of functions for optimization
  • First-order condition:
  \[ \langle x - y, \nabla f(x) - \nabla f(y) \rangle \geq \mu \| x - y \|^2 \]
  • Second-order condition:
  \[ \nabla^2 f(x) \succeq \mu I \]
  • Equivalently:
  \[ h(x) = f(x) - \frac{\mu}{2} \| x \|^2 \text{ is convex} \]
Which of the functions we’ve looked at are strongly convex?
Which of the functions we’ve looked at are strongly convex?
Concave functions

• A function is concave if its negation is convex

\[ f \text{ is convex } \iff h(x) = -f(x) \text{ is concave} \]

• Example: \( f(x) = \log(x) \)

\[ f''(x) = -\frac{1}{x^2} \leq 0 \]
Why care about convex functions?
Convex Optimization

• Goal is to minimize a convex function
Gradient Descent

\[ w \leftarrow w - \alpha \nabla f(w) \]
Gradient Descent Converges

• Iterative definition of gradient descent

\[ w_{t+1} = w_t - \alpha \nabla f(w_t) \]

• Assumptions/terminology:

Global optimum is \( x^* \)

Bounded second derivative \( \mu I \preceq \nabla^2 f(x) \preceq LI \)
Gradient Descent Converges (continued)

\[ w_{t+1} - w^* = w_t - w^* - \alpha (\nabla f(w_t) - \nabla f(w^*)) \]
\[ = w_t - w^* - \alpha \nabla^2 f(\zeta_t) (w_t - w^*) \]
\[ = (I - \alpha \nabla^2 f(\zeta_t)) (w_t - w^*) . \]

Taking the norm

\[ \| w_{t+1} - w^* \| \leq \| I - \alpha \nabla^2 f(\zeta_t) \|_2 \cdot \| w_t - w^* \| \]
\[ \leq \max(|1 - \alpha \mu|, |1 - \alpha L|) \cdot \| w_t - w^* \|. \]
Gradient Descent Converges (continued)

• So if we set $\alpha = \frac{2}{(L + \mu)}$ then

$$\|w_{t+1} - w^*\| \leq \frac{L - \mu}{L + \mu} \cdot \|w_t - w^*\|$$

• And recursively

$$\|w_K - w^*\| \leq \left(\frac{L - \mu}{L + \mu}\right)^K \cdot \|w_0 - w^*\|$$

• Called convergence at a linear rate or sometimes (confusingly) exponential rate
The Problem with Gradient Descent

• Large-scale optimization

\[ h(w) = \frac{1}{n} \sum_{i=1}^{n} f(w; x_i) \]

• Computing the gradient takes \( O(n) \) time

\[ \nabla h(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla f(w; x_i) \]
Gradient Descent with More Data

• Suppose we add more examples to our training set
  • For simplicity, imagine we just add an extra copy of every training example

\[ \nabla h(w) = \frac{1}{2n} \sum_{i=1}^{n} \nabla f(w; x_i) + \frac{1}{2n} \sum_{i=1}^{n} \nabla f(w; x_i) \]

• Same objective function
  • But gradients take 2x the time to compute (unless we cheat)

• We want to scale up to huge datasets, so how can we do this?
Stochastic Gradient Descent

- Idea: rather than using the full gradient, just use one training example
  - Super fast to compute

\[ w_{t+1} = w_t - \alpha \nabla f(w_t, x_{i_t}) \]

- In expectation, it’s just gradient descent:

\[
\mathbb{E}[w_{t+1}] = \mathbb{E}[w_t] - \alpha \cdot \mathbb{E}[\nabla f(w_t, x_{i_t})] \\
= \mathbb{E}[w_t] - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla f(w_t, x_i)
\]

This is an example selected uniformly at random from the dataset.
Stochastic Gradient Descent Convergence

• Can SGD converge using just one example to estimate the gradient?

\[ w_{t+1} - w^* = w_t - w^* - \alpha (\nabla h(w_t) - \nabla h(w^*)) - \alpha (\nabla f(w_t; x_{i_t}) - \nabla h(w_t)) \]

\[ = (I - \alpha \nabla^2 h(\zeta_t)) (w_t - w^*) - \alpha (\nabla f(w_t; x_{i_t}) - \nabla h(w_t)) \]

• How do we handle this extra noise term?

• Answer: bound it using the second moment!
Stochastic Gradient Descent Convergence

\[
E \left[ \|w_{t+1} - w^*\|^2 \right] = E \left[ \| (I - \alpha \nabla^2 h(\zeta_t)) (w_t - w^*) - \alpha (\nabla f(w_t; x_{i_t}) - \nabla h(w_t)) \|^2 \right] \\
= E \left[ \| (I - \alpha \nabla^2 h(\zeta_t)) (w_t - w^*) \|^2 \right] \\
- \alpha E \left[ (\nabla f(w_t; x_{i_t}) - \nabla h(w_t))^T (I - \alpha \nabla^2 h(\zeta_t)) (w_t - w^*) \right] \\
+ \alpha^2 E \left[ \| (\nabla f(w_t; x_{i_t}) - \nabla h(w_t)) \|^2 \right] \\
= E \left[ \| (I - \alpha \nabla^2 h(\zeta_t)) (w_t - w^*) \|^2 \right] + \alpha^2 E \left[ \| (\nabla f(w_t; x_{i_t}) - \nabla h(w_t)) \|^2 \right] \\
\leq (1 - \alpha \mu)^2 \cdot E \left[ \|w_t - w^*\|^2 \right] + \alpha^2 M
\]

assuming small enough \( \alpha \) and the bound \( E \left[ \| (\nabla f(w; x_i) - \nabla h(w)) \|^2 \right] \leq M. \)
Stochastic Gradient Descent Convergence

• Already we can see that this converges to a fixed point of

\[ \lim_{t \to \infty} E \left[ \| w_t - w^* \|^2 \right] \leq \frac{\alpha M}{2\mu - \alpha \mu^2} \]

• This phenomenon is called converging to a **noise ball**
  • Rather than approaching the optimum, SGD (with a constant step size) converges to a region of low variance around the optimum
  • This is okay for a lot of applications that **only need approximate solutions**
Demo
Stochastic gradient descent is super popular.
What Does SGD Power?

• Everything!
But how SGD is implemented in practice is not exactly what I’ve just shown you...

...and we’ll see how it’s different in the upcoming lectures.