Optimization for Deep Learning (OPT4DL)

Abolfazl Hashemi, Purdue ECE

Table of contents i

Table of contents ii

Introduction to the Course

Optimization

- Selection/identification of the best element with respect to some criterion, from a set of alternatives
  - cost/loss minimization

$$\min_{w} f(w) \qquad s.t. \quad w \in \mathcal{X} \tag{1}$$

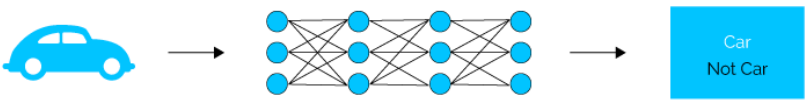
  - reward/utility maximization

$$\max_{w} \mathrm{U}(w) \qquad s.t. \quad w \in \mathcal{X} \tag{2}$$

- Concerned with modeling, Algorithms, Assessment, all pertinent to ML!

Deep Learning

- A class of Machine Learning (ML) methods based on Artificial Neural Networks (NNs) that enable learning/identifying useful patterns from data, and applying those patterns to new data



- Many application domain, most notably generative AI models (Chatgpt, Stable Diffusion, etc.)

Deep Learning (Cont'd)

Machine Intelligence Networked Data Science

- As defined by one of the pioneers of the field

Yann LeCun

December 24, 2019 ·

Some folks still seem confused about what deep learning is. Here is a definition:

DL is constructing networks of parameterized functional modules & training them from examples using gradient-based optimization. That's it.

This definition is orthogonal to the learning paradigm: reinforcement, supervised, or self-supervised.

Don't say "DL can't do X" when what you really mean is "supervised learning needs too much data to do X"

Extensions (dynamic networks, differentiable programming, graph NN, etc) allow the network architecture to change dynamically in a data-dependent way.

- Indicating the central role of the right kind of optimization!

Why OPT4DL?

Machine Intelligence Networked Data Science

- Two main reasons
  - The learning in DL and ML boils down to optimization problems
  - A lot of mysteries in getting the DL models work (even if fine-tuning a pre-trained model). Thus, given the first reason, right kind of optimization may lead to understanding and demystifying them.

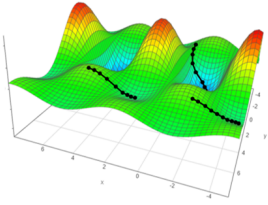
The Intimate Connection of Learning and Optimization

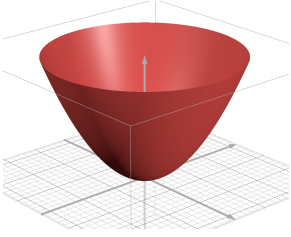
- Typical ML/DL task

$$\min_{w} f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) \qquad s.t. \quad w \in \mathbb{R}^d \tag{3}$$

where

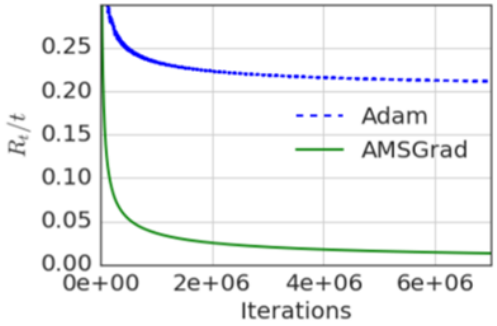
- n is the size of training data (could be billions)
- d the number of model parameters (could be billions), typically overparameterized: $d > n$
- L the average loss is non-convex





Mystery: Why/How Adam Works?

- Adam: an optimization algorithm for DL training
- Paper unfortunately had a mistake in its theory
- It sometimes fails to solve easy problems
- Yet, very successful and ubiquitous in practice

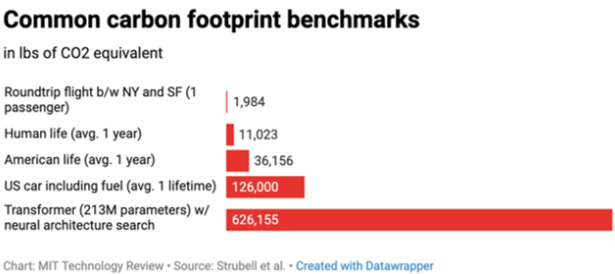


Mystery: Why Overparameterization is Good?

- Traditional ML theory seems to require d must be smaller than n to avoid overfitting, i.e., not generalizing well to unseen/new data
- But, in practice $d \gg n$ seems to generalize even better
- Magically, from many available solutions, DL model reaches a great one

Why Developing an Understanding is Essential?

- DL is and probably will continue to be a useful tool in many fields across science and engineering
- We deal with more than one task and one dataset
- DL training is costly: current success reliant on expensive hardware and high energy consumption



- Societal concerns/implications of DL and how to mitigate/resolve them

Why (the right kind of) Optimization may be Helpful?

- Learning and Optimization are entwined
- Successful Optimization is a main pillar of successful DL
- Modern optimization theory could provide actionable insights about complex, non-convex losses in DL and their implications to both convergence and generalization
  - convergence: finding a solution that matches the training data
  - generalization: good performance on unseen data

Our Main Goals in OPT4DL Class

- Learning about learning techniques, and potentially developing new ones
- Discuss OPT4DL theory, methods, and heuristics
- Pose why questions and discuss what we already know, what we need to know and how to approach doing so
- Leverage OPT4DL for resource-efficient DL training
- Discuss some pertinent practical and societal concerns and how to integrate them with our optimization methods

Introduction to me

- Abolfazl Hashemi, Ph.D. (email: `abolfazl@purdue.edu`)
- Assistant Professor at The Elmore Family School of Electrical and Computer Engineering at Purdue University, Since Fall 2021
- Research Goal: advance the field of Large-Scale Optimization provides actionable insights from the perspective of this foundational field to innovate multiple domains within ML/AI
- Recent Applications: Federated Learning, Medical Image Analysis, NextG Manufacturing, and Cyber-Physical Systems

Optimization Basics

Stationary Points

- Consider

$$\min_{w} f(w) \qquad s.t. \quad w \in \mathbb{R}^d \tag{4}$$

- w^* is a stationary point if the gradient (derivative) of f at w^* is zero:

$$\nabla f(w^*) = 0 \tag{5}$$

- If further $\nabla^2 f(w^*) \succ 0$, then w^* is a local minimum
- If further $0 \succ \nabla^2 f(w^*)$, then w^* is a local maximum
- A stationary (or even a local minimum) is not necessarily an optimal solution

- A set $\mathcal{X}$ is convex if for all x_i in $\mathcal{X}$ and all $m \geq 1$

$$\sum_{i=1}^{m} w_i x_i \in \mathcal{X} \tag{6}$$

where

$$w_i \geq 0, \quad \sum_{i=1}^{m} w_i = 1 \tag{7}$$

- In other words, the convex combination of all points belong to the set as well.

Convex Functions

- f is (strongly) convex if for all x_i

$$f(\sum_{i=1}^{m} w_i x_i) \leq \sum_{i=1}^{m} w_i f(x_i) \tag{8}$$

where

$$w_i \geq 0, \quad \sum_{i=1}^{m} w_i = 1 \tag{9}$$

($<$ for strongly convex case instead of $\leq$)

Convex Functions (Cont'd)

- f is (strongly) convex if all x and y

$$f(\mathrm{y}) \geq f(x) + \langle \nabla f(x), \mathrm{y} - x \rangle + \frac{\mu}{2} \|\mathrm{y} - x\|^2 \tag{10}$$

Convex Functions (Cont'd)

- f is (strongly) convex if for all x

$$\nabla^2 f(x) \succeq 0 \tag{11}$$

($\succ$ for strongly convex case instead of $\succeq$)

Why Convexity is Nice?

- Local information provides global information
- Any stationary point w^* (can be identified locally) is a global minimum (a property requiring global information about the function in general)
- Proof is easy to see from the definition

$$f(\mathrm{y}) \geq f(w^*) + \langle \nabla f(w^*), \mathrm{y} - w^* \rangle \tag{12}$$

  and noting $\nabla f(w^*) = 0$
- Furthermore, A strongly convex function has a unique minimizer

Convergence Notions

- We are interested in establishing

$$\mathrm{err}(\tilde{w}) \leq \epsilon \tag{13}$$

  for some target accuracy ϵ and a notion of convergence $\mathrm{err}(\cdot)$
- Our methods achieving this goal are typically iterative, i.e., they find an approximate solution in T iterations
- Typically better accuracy (lower ϵ) requires larger T
  - Sublinear rates: $T = \Omega(\epsilon^{-a})$, for some $a > 0$
  - Linear rates: $T = \Omega(\log \frac{1}{\epsilon})$
- Typical examples of $\mathrm{err}(\tilde{w})$
  - approximate first-order: $\|\nabla f(\tilde{w})\| \leq \epsilon$, for some $a > 0$ (General functions)
  - Low suboptimality gap: $f(\tilde{w}) - f^* \leq \epsilon$ (Convex)
  - Low point-wise error $\|\tilde{w} - w^*\| \leq \epsilon$ (Strongly convex functions)

Optimization as a Fixed Point Problem

- Consider

$$\min_{w} f(w) \qquad s.t. \quad w \in \mathbb{R}^d \tag{14}$$

- A necessary condition for optimality is first-order stationarity $\nabla f(w^*) = 0$
- Equivalently

$$\eta \nabla f(w^*) - w^* = -w^* \tag{15}$$

- Equivalently, the problem is to find the fixed points of $g(w) := w - \eta \nabla f(w)$
- The simplest method is the Fixed Point Iteration $w_{t+1} = g(w_t)$, for all $t = 1, \ldots, T$ which leads to

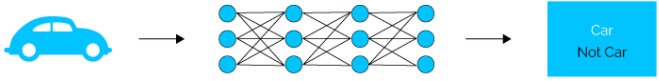
$$w_{t+1} = w_t - \eta \nabla f(w_t) \tag{16}$$

- This method is called Gradient Descent (GD) with stepsize/learning rate $\eta > 0$ which serves as the foundational of all DL training methods.

Machine Learning (ML) Basics

Machine Learning: Learning from Data

- We have $\mathcal{D} = \{z_1, \ldots, x_n\}$ a collection of n training data points/samples z_i's
- Think of z_i as a collection of features and their corresponding label, that is $z_i = (x_i, y_i)$
  - $x_i \in \mathbb{R}^p$ is a vector of p features, e.g., an image with p pixels
  - $y_i \in \mathbb{R}$ is the label for the i^{th} sample, e.g., whether the image is a car or not



- Typically, ML problems are one of the followings
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning

- Assume there exists some true but unknown rule between features and labels $y_i = A^*(x_i)$ for all i
- Choose a parametric model $A(\cdot, w)$ such that for some w^*, $A(\cdot, w^*) \approx A^*(\cdot)$
- Training: fit the model to $\mathcal{D}$ to find w^*
- Let $\ell(w, z_i)$ denote the loss/cost of a model w fitting sample i
  - Example: quadratic loss $\ell(w, z_i) = \frac{1}{2}(A(x_i, w) - y_i)^2$
- Training as an optimization problem

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) \tag{17}$$

- That is, minimize the average loss over the training data
- Let w^* denote the optimal solution

Performance on Unseen Data

- If training successful, model will fit training data well
- To assess performance on new, unseen data (data $\notin \mathcal{D}$) we make an assumption:
  - New and training data belong to the same universe
  - Let z be a new data, then $z \sim p_z$ and $\mathcal{D} \sim p_z$ for some data distribution p_z
- Measure of performance

$$\min_{w \in \mathbb{R}^d} \bar{f}(w) = \mathbb{E}_{z \sim p_z}[\ell(w, z_i)] \tag{18}$$

  how well we perform on average over the entire universe (not just over the training data)
- Note that f and $\bar{f}$ are not necessarily the same, f an empirical approximation of $\bar{f}$

Example: A Discrete Universe

- Let p_z be discrete and $p_z(Z = z_i) = \frac{1}{n}$
- That is, our data universe is discrete and each data is equally likely to be observed
- Then, f and $\bar{f}$ are

- To ensure performance on all data (both seen and unseen), we aim to minimize the Population Loss/Risk

$$\bar{w} := \arg\min_{w \in \mathbb{R}^d} \bar{f}(w) = \mathbb{E}_{z \sim p_z}[\ell(w, z_i)] \tag{19}$$

- But, all we can do in practice is to minimize the Empirical Loss/Risk

$$w^* := \arg\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) \tag{20}$$

  as p_z is unknown

- This approach is called Empirical Risk Minimization (ERM)

How good is Approximately Solving ERM?

- Let $\hat{w}$ be an approximate/suboptimal solution to ERM
- And recall $\bar{w}$ and w^* are the minimizers of Population loss ($\bar{f}$) and Empirical loss (f), respectively.
- We need to bound the Estimation Error: $\bar{f}(\hat{w}) - \bar{f}(\bar{w})$

$$\begin{aligned}\bar{f}(\hat{w}) - \bar{f}(\bar{w}) &= \bar{f}(\hat{w}) \pm f(\hat{w}) \pm f(w^*) \pm f(\bar{w}) - \bar{f}(\bar{w}) \\ &= f(\hat{w}) - f(w^*) \\ &+ \bar{f}(\hat{w}) - f(\hat{w}) \\ &+ f(w^*) - \bar{f}(w^*) \\ &+ f(w^*) - f(\bar{w})\end{aligned} \tag{21}$$

- estimation error $\leq$ optimization error $+2\times$ empirical approximation error
- Controlled by better training algorithms and better/larger datasets

Representation Error

- Recall $\bar{w}$ is the minimizer of Population loss ($\bar{f}$)
- Recall we assume $y = A^*(x)$
- If our model is not powerful enough, even by minimizing the Population Loss, we may not fully learn the feature-label relations
- Mathematically, $A^*(\cdot)$ and $A(\cdot, \bar{w})$ are not necessarily the same
- Representation error $= \bar{f}(\bar{w}) - f_{opt}$ where f_{opt} is the loss if we knew the true relation (we can assume $f_{opt} \approx 0$)

- Typically, representation error decreases as DL models become deeper and/or wider (more powerful)
- Total Learning Error = Representation Error + Optimization/Training Error + Empirical Approximation Error

Examples of Models

- Binary Classification
  - Features: $x \in \mathbb{R}^d$
  - Labels: $y \in \{-1, +1\}$
  - Linearly separable classes: $\hat{y} = \text{sign}(x^\top w)$
  - 0-1 loss: $\ell(w, z) = \mathbb{I}[y \neq \text{sign}(x^\top w)]$
  - Hinge loss: $\ell(w, z) = \max(0, 1 - x^\top w)$
  - ERM with hinge becomes a convex task
  - The linear relation $x^\top w$ could be replaced by a DL model $A(x, w)$, resulting in a nonconvex training tasks

Examples of Models (Cont'd)

- Linear Regression
  - Features: $x \in \mathbb{R}^d$
  - Labels: $y \in \mathbb{R}$
  - Linear dependency: $y = x^\top w + e$ where $e \sim \mathcal{N}(0, \sigma^2)$
  - Square loss: $\ell(w, z) = \frac{1}{2}(y - x^\top w)^2$
  - ERM becomes a convex task
  - The linear relation $x^\top w$ could be replaced by a DL model $A(x, w)$, resulting in a nonconvex training task

Examples of Models (Cont'd)

- Logistic Regression
  - Features: $x \in \mathbb{R}^d$
  - Labels: $y \in \{0, 1\}$
  - Sigmoid/logistic activation:

$$y = \frac{1}{1 + \exp(-x^\top w)} \tag{22}$$

  - $\frac{1}{1+\exp(-x^\top w)}$ is thought as the probability that the label of x is 1.
  - Binary Cross-entropy loss:

$$\ell(w, z) = -y \log \frac{1}{1 + \exp(-x^\top w)} - (1 - y) \log(1 - \frac{1}{1 + \exp(-x^\top w)}) \tag{23}$$

  - ERM becomes a convex task
  - The linear relation $x^\top w$ could be replaced by a DL model $A(x, w)$, resulting in a nonconvex training task

Gradient Descent: Performance under Convexity

Gradient Descent (GD): A Basic Optimization Method

- Consider

$$\min_{w \in \mathbb{R}^d} f(w) \tag{24}$$

- We talked about a simple method based on Fixed Point Calculation called Gradient Descent (GD)
  - Initialize w_1 and Learning rate $0 < \eta \leq 1$
  - For $t = 1, \ldots, T$ iterations

$$w_{t+1} = w_t - \eta g_t \tag{25}$$

    where $g_t = \nabla f(w_t)$

- Think of g_t as the update vector at time t

Gradient Descent: Intuition

- Intuition: Gradually follow the direction of decrease
- If $f(x) \geq f(y)$, by Taylor's theorem, we have locally

$$\begin{aligned} f(y) &\approx f(x) + \langle y - x, \nabla f(x) \rangle \\ 0 \geq f(y) - f(x) &\approx \langle y - x, \nabla f(x) \rangle \\ 0 &\geq \langle y - x, \nabla f(x) \rangle \end{aligned} \tag{26}$$

That is, gradient locally gives us the direction of increase.

How good is GD on Convex Functions

- Recall by convexity we can bound the suboptimality of a solution w_t, i.e. $f(w_t) - f(w^*)$

$$f(w^*) \geq f(w_t) + \langle w^* - w_t, \nabla f(w_t) \rangle \tag{27}$$

or equivalently

$$f(w_t) - f(w^*) \leq \langle \nabla f(w_t), w_t - w^* \rangle \tag{28}$$

- Thus, it suffices to bound the above inner-product

Proposition

For any sequence of update vectors $g_1, \ldots, g_T \in \mathbb{R}^d$ and any vector $w^* \in \mathbb{R}^d$, the update rule $w_{t+1} = w_t - \eta g_t$ satisfies

$$\sum_{t=1}^{T} \langle g_t, w_t - w^* \rangle \leq \frac{1}{2\eta} \|w_1 - w^*\|^2 + \frac{\eta}{2} \sum_{t=1}^{T} \|g_t\|^2 \tag{29}$$

Proof of the Proposition

- Let us consider the evolution of $\|w_t - w^*\|^2$ and leverage the form of the update

$$\begin{aligned}\|w_{t+1} - w^*\|^2 &= \|w_t - \eta g_t - w^*\|^2 \\ &= \|w_t - w^*\|^2 + 2\eta\langle g_t, w_t - w^*\rangle + \eta^2\|g_t\|^2\end{aligned} \tag{30}$$

- Rearrange

$$\langle g_t, w_t - w^*\rangle = \frac{\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2}{2\eta} + \frac{\eta}{2}\|g_t\|^2 \tag{31}$$

- Summing over t

$$\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle = \sum_{t=1}^{T}\frac{\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2}{2\eta} + \sum_{t=1}^{T}\frac{\eta}{2}\|g_t\|^2 \tag{32}$$

How good is GD on Convex Functions (Cont'd)

- Recall in GD: $w_{t+1} = w_t - \eta \nabla f(w_t)$ (that is, $g_t = \nabla f(w_t)$)
- By convexity $f(w_t) - f(w^*) \leq \langle \nabla f(w_t), w_t - w^* \rangle$
- Thus, by our proposition

$$\text{Accumulated suboptimality} = \sum_{t=1}^{T} f(w_t) - f(w^*) \leq \frac{1}{2\eta} \|w_1 - w^*\|^2 + \frac{\eta}{2} \sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \quad (33)$$

Definition

A function $f : \mathcal{X} \to \mathbb{R}$ is G-Lipschitz if and only if

$$|f(x) - f(y)| \leq G\|x - y\| \tag{34}$$

for some $G \geq 0$ and all $x, y \in \mathcal{X}$.

- Intuition: Bounded sensitivity and a measure of continuity of a function
- By rearranging we see the slope is bounded

$$\frac{|f(x) - f(y)|}{\|x - y\|} \leq G \tag{35}$$

- Thus, if the function is differentiable, equivalently a function is G-Lipschitz iff $\|\nabla f(x)\| \leq G$

G-Lipschitz Functions: Examples

- $f(x) = \sqrt{x}$
  - $x \in (0, 1]$
  - $x \in [0.5, 1]$
- $f(x) = |x|$
  - $x \in \mathbb{R}$
- $f(x) = e^x$
  - $x \in \mathbb{R}$
  - $x \in [a, b]$
- $f(x) = x^2$
  - $x \in \mathbb{R}$
  - $x \in [1, 10]$

Implications for GD

- Assume f is G-Lipschitz, i.e., $\|\nabla f(x)\| \leq G$
- Let $D = \|w_1 - w^*\|$
- From our proposition

$$\sum_{t=1}^{T} f(w_t) - f(w^*) \leq \frac{1}{2\eta}\|w_1 - w^*\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|\nabla f(w_t)\|^2 \tag{36}$$

Theorem

Assume f is G-Lipschitz and convex. GD with the update rule $w_{t+1} = w_t - \eta\nabla f(w_t)$ and learning rate $\eta = \frac{D}{G\sqrt{T}}$ satisfies the following bound on the incurred average suboptimality

$$\frac{1}{T}\sum_{t=1}^{T} f(w_t) - f(w^*) \leq \frac{DG}{\sqrt{T}} \tag{37}$$

How to Output a Solution?

- In practice we output the last solution, i.e. $\hat{w} = w_T$ as it tends to have a better performance
  - We need a different analysis or more structures to prove its performance
- Alternatively, let $\hat{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$
  - By convexity

$$f(\hat{w}) - f(w^*) \leq f(\frac{1}{T}\sum_{t=1}^{T} w_t) - f(w^*) \leq \frac{1}{T}\sum_{t=1}^{T} f(w_t) - f(w^*) \leq \frac{DG}{\sqrt{T}} \tag{38}$$

- Alternatively, let $\hat{w} \sim \mathcal{U}\{w_1, \ldots, w_T\}$
  - On expectation, this randomized solution satisfies

$$\mathbb{E}_{\hat{w}}[f(\hat{w}) - f(w^*)] = \frac{1}{T}\sum_{t=1}^{T} f(w_t) - f(w^*) \leq \frac{DG}{\sqrt{T}} \tag{39}$$

Analyzing the Bound

Theorem

GD with the update rule $w_{t+1} = w_t - \eta \nabla f(w_t)$ and learning rate $\eta = \frac{D}{G\sqrt{T}}$ satisfies

$$\text{Average Suboptimality} \leq \frac{DG}{\sqrt{T}} = \mathcal{O}(1/\sqrt{T}) \tag{40}$$

- The regulating role played by the learning rate, η
- Recall $w \in \mathbb{R}^d$, so no explicit dependence on the intrinsic dimension
- d, however finds its way in $\|\nabla f(x)\| \leq G$ and $D = \|w_1 - w^*\|$, typically
- Iteration Complexity: Number of iterations T to find an ϵ-accurate solution

$$T \geq \frac{D^2 G^2}{\epsilon} = \Omega(1/\epsilon^2) \Rightarrow \text{Average Suboptimality} \leq \epsilon \tag{41}$$

- Can we do better?

Introduction to Stochastic Gradient Descent (SGD)

- In ML/DL our loss function typically has the form

$$f(w) = \mathbb{E}_z[\ell(w, z)] \tag{42}$$

  (recall, think of it as average loss over data, i.e., $\frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i)$)
- To run GD, we need the gradient $\nabla \mathbb{E}_z[\ell(w, z)]$, which could be difficult to calculate
  - Large n means running out of memory and/or lots of calculations
  - In some cases we may not be able to calculate the expectation exactly (hard integrals)
- The main idea behind most training methods is to find good estimators such that $\hat{\nabla} \approx \nabla \mathbb{E}_z[\ell(w, z)]$

Proposition

If ℓ is a differentiable function of w and z, we can switch the order of derivative and expectation:

$$\nabla \mathbb{E}_z[\ell(w,z)] = \mathbb{E}_z[\nabla \ell(w,z)] \tag{43}$$

- This leads to leveraging sample approximation to estimate the expectation

$$z^1, \ldots, z^B \sim p_z, \qquad \mathbb{E}_z[\nabla \ell(w,z)] \approx \frac{1}{B} \sum_{j=1}^{B} \nabla \ell(w, z^j) \tag{44}$$

Stochastic Gradient Descent (SGD)

- Initialize w_1, learning rate $0 < \eta_t \leq 1$, and batch size $B \geq 1$
- For $t = 1, \ldots, T$ perform:
  - Sample a mini-batch of samples $z_t^1, \ldots, z_t^B \sim p_z$
  - Form the update vector $g_t = \frac{1}{B} \sum_{j=1}^{B} \nabla \ell(w_t, z_t^j)$
  - Update the parameters $w_{t+1} = w_t - \eta g_t$

Which Estimators are Good?

- We like estimators that have low bias (ideally zero) and low variance

Bias

Let X be a quantity and Y a Random Estimator of X. The bias of Y is defined as

$$\text{Bias}_Y = \|X - \mathbb{E}[Y]\| \tag{45}$$

Notably, Y is unbiased if $X = \mathbb{E}[Y]$.

Variance

Let X be a quantity and Y a Random Estimator of X. The variance of Y is defined as

$$\text{Var}(Y) = \mathbb{E}\|Y - \mathbb{E}[Y]\|^2 \tag{46}$$

Notably, if Y is unbiased, i.e., $X = \mathbb{E}[Y]$, we have $\text{Var}(Y) = \mathbb{E}\|Y - X\|^2$.

Which Estimators are Good? (Cont'd)

- We can characterize the total error of an estimators using bias-variance decomposition:

$$\text{Total error} = \mathbb{E}\|Y - X\|^2 = \|X - \mathbb{E}[Y]\|^2 + \mathbb{E}\|Y - \mathbb{E}[Y]\|^2 = \text{bias plus variance} \tag{47}$$

- Averaging independent unbiased estimators could be useful in lowering the error

$$\begin{aligned}\mathbb{E}\|\frac{Y_1+Y_2}{2} - X\|^2 = \mathbb{E}\|\frac{(Y_1-X)+(Y_2-X)}{2}\|^2 &= \frac{1}{4}\mathbb{E}\|(Y_1-X)+(Y_2-X)\|^2 \\ &= \frac{1}{4}\left(\mathbb{E}\|(Y_1-X)\|^2 + \mathbb{E}\|(Y_2-X)\|^2 + \mathbb{E}[\langle Y_1-X, Y_2-X\rangle]\right) \\ &= \frac{1}{4}\left(\mathbb{E}\|(Y_1-X)\|^2 + \mathbb{E}\|(Y_2-X)\|^2\right)\end{aligned} \tag{48}$$

Why SGD is a Good Idea?

- Each estimator $Y_j = \nabla \ell(w_t, z_t^j)$ is an unbiased estimator of loss gradient if we sample from the distribution
- The average of Y_j's will also remain unbiased
- As $B \to \infty$, by strong law of large numbers our estimator $\frac{1}{N} \sum_{j=1}^{B} Y_j$ (almost surely) approaches the loss gradient.
- But B cannot be too large in practice when optimizing model parameters. Impact on training and test errors?

Stochastic First-Order Oracle (SFO)

Given the current parameter w_t and the sample/data used in iteration t, i.e., z_t, Oracle returns an update vector g_t which is conditionally unbiased with bounded variance:

$$\mathbb{E}_{z_t}[g_t|w_t] = \nabla f(w_t), \qquad \mathbb{E}_{z_t}[\|g_t - \nabla f(w_t)\|^2|w_t] \leq \sigma^2 \tag{49}$$

for some $0 \leq \sigma^2 < \infty$.

- SFO enables formal study of not just SGD, but also more advanced training methods
- A simple way to model gradient calculation in practice
- We need the conditioning since w_t is itself random: it depends on our (random) selection of mini batches in previous iterations. In other words, w_t is a function of $z_1, \ldots, z_{t-1}$ (note the exclusion of z_t).
- Thus, the following expectations are equivalent: $\mathbb{E}_{z_t}[\cdot|w_t] \equiv \mathbb{E}_{z_t}[\cdot|z_1, \ldots, z_{t-1}]$

Performance of SGD

Theorem

Assume f is G-Lipschitz and convex. Furthermore assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, SGD with the update rule $w_{t+1} = w_t - \eta g_t$ and learning rate $\eta = \frac{D}{\sqrt{T(G^2+\sigma^2)}}$ satisfies the following bound on the incurred average suboptimality

$$\mathbb{E}_{z_1,\ldots,z_T}[\frac{1}{T}\sum_{t=1}^{T} f(w_t) - f(w^*)] \leq \frac{D\sqrt{G^2+\sigma^2}}{\sqrt{T}} \leq \frac{DG}{\sqrt{T}} + \frac{D\sigma}{\sqrt{T}} \tag{50}$$

- As our method is a randomized algorithm, our statement regarding its performance needs to be stated either on expectation or with high confidence
- As we discussed, with larger batch sizes $\sigma^2 \to \sigma^2/B$
- We can use either of 3 approaches to generate an output, that is (i) the last/best parameter w_T, (ii) averaging all parameters $w_1, \ldots, w_T$, and (iii) sampling one iterate at random

Minibatch SGD: A Fixed Budget Analysis

- Recall after T iterations of mini batch SGD

$$\text{error} \leq \frac{DG}{\sqrt{T}} + \frac{D\sigma}{\sqrt{BT}} \tag{51}$$

- What is the best B if we fix the total computation budget, i.e., number of iterations times number of stochastic gradient calculation per iteration: $C = B \times T$ (also called number of SFO calls)
- In terms of C, the error becomes

$$\text{error} \leq \frac{DG\sqrt{B}}{\sqrt{C}} + \frac{D\sigma}{\sqrt{C}} \tag{52}$$

  Suggesting the best batch size is $B = 1$?
- Why mini batch SGD with $B > 1$ then?

Answer: Parallel computation

- Assume we have M machines and that the computing time scales linearly with M

$$\tau = \frac{C}{M} \tag{53}$$

- Assume $M = B$ our batch size. Then, in terms of the computing time, the error becomes

$$\text{error} \leq \frac{DG}{\sqrt{\tau}} + \frac{D\sigma}{\sqrt{\tau B}} \tag{54}$$

- Larger batch sizes plus parallel computation reduces the error faster
- Remark: an analysis based on worst case scenario
- $B > 1$ leads to lower variance and smoother convergence (less ups and downs)

SGD in Practice (Reshuffling)

Consider a training problem, i.e. ERM:

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i) \tag{55}$$

- For $e = 1, \ldots, E$ epochs do
  - For $t = 1, \ldots, K$ inner iterations do
    1. form the mini-batch of samples $z_t^1, \ldots, z_t^B \sim p_z$
    2. Form the update vector $g_t = \frac{1}{B}\sum_{j=1}^{B} \nabla \ell(w_t, z_t^j)$
    3. Update the parameters $w_{t+1} = w_t - \eta g_t$
  - Optionally, shuffle the data into a new random order)

Has been observed to lead to smoother loss decay and better generalization

This can be attributed to the lower variance of the resultant (and more complex) gradient estimator, due to reshuffling

Theorem

Assume f is G-Lipschitz and convex. Furthermore assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, SGD with the update rule $w_{t+1} = w_t - \eta g_t$ and learning rate $\eta = \frac{D}{\sqrt{T(G^2+\sigma^2)}}$ satisfies the following bound on the incurred average suboptimality

$$\mathbb{E}_{z_1,\ldots,z_T}[\frac{1}{T}\sum_{t=1}^{T} f(w_t) - f(w^*)] \leq \frac{D\sqrt{G^2+\sigma^2}}{\sqrt{T}} \leq \frac{DG}{\sqrt{T}} + \frac{D\sigma}{\sqrt{T}} \tag{56}$$

- We will leverage our general proposition:

$$\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle \leq \frac{1}{2\eta}\|w_1 - w^*\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2 \tag{57}$$

  with suitable conditional expectations to account for the randomization of SGD.

- Let $z_{1:t}$ denote $z_1, \ldots, z_t$. We have by tower expectation, linearity of expectation, and the unbiasedness property of the SFO

$$\begin{aligned}\mathbb{E}_{z_{\mathbf{1}:t}}[\langle g_t, w_t - w^*\rangle] &= \mathbb{E}_{z_{\mathbf{1}:t-\mathbf{1}}}[\mathbb{E}_{z_t}[\langle g_t, w_t - w^*\rangle | z_{1:t-1}]] = \mathbb{E}_{z_{\mathbf{1}:t-\mathbf{1}}}[\langle \mathbb{E}_{z_t}[g_t | z_{1:t-1}], w_t - w^*\rangle] \\ &= \mathbb{E}_{z_{\mathbf{1}:t-\mathbf{1}}}[\langle \nabla f(w_t), w_t - w^*\rangle]\end{aligned} \tag{58}$$

Proof of Theorem (Cont'd)

- Recall, by convexity

$$f(w_t) - f(w^*) \leq \langle \nabla f(w_t), w_t - w^* \rangle \tag{59}$$

- Thus

$$\mathbb{E}_{z_{1:t-1}}[f(w_t) - f(w^*)] \leq \mathbb{E}_{z_{1:t-1}}[\langle \nabla f(w_t), w_t - w^* \rangle] \tag{60}$$

- Using the independence of samples used in each iteration (w_t only a function of $z_{1:t-1}$ and independent from the future samples $z_t, \ldots, z_T$)

$$\mathbb{E}_{z_1,\ldots,z_T}[f(w_t) - f(w^*)] = \mathbb{E}_{z_{1:t-1}}[f(w_t) - f(w^*)] \tag{61}$$

- Thus, by linearity of expectation and our general proposition

$$\mathbb{E}_{z_1,\ldots,z_T}[\sum_{t=1}^{T} f(w_t) - f(w^*)] \leq \frac{1}{2\eta}\|w_1 - w^*\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}_{z_1,\ldots,z_T}\|g_t\|^2 \tag{62}$$

Proof of Theorem (Cont'd)

- For the second term note by the independence of the samples (g_t only a function of w_t and z_t)

$$\mathbb{E}_{z_1,\dots,z_T}\|g_t\|^2 = \mathbb{E}_{z_1,\dots,z_t}\|g_t\|^2 \tag{63}$$

- By tower expectation

$$\mathbb{E}_{z_1,\dots,z_t}\|g_t\|^2 = \mathbb{E}_{z_1,\dots,z_{t-1}}[\mathbb{E}_{z_t}[\|g_t\|^2|z_1,\dots,z_{t-1}]] \tag{64}$$

- We will now aim to use our unbiased SFO assumption with bounded variance,

$$\begin{aligned}\mathbb{E}_{z_t}[\|g_t\|^2|z_1,\dots,z_{t-1}] &= \mathbb{E}_{z_t}[\|g_t \pm \nabla f(w_t)\|^2|z_1,\dots,z_{t-1}] \\ &= \mathbb{E}_{z_t}[\|\nabla f(w_t)\|^2|z_1,\dots,z_{t-1}] + \mathbb{E}_{z_t}[\|g_t - \nabla f(w_t)\|^2|z_1,\dots,z_{t-1}] \\ &\leq G^2 + \sigma^2\end{aligned} \tag{65}$$

by Lipschitzness and bounded variance assumptions. Division by T finishes the proof.

Beyond Convexity: (S)GD and Smoothness

Beyond Convex Functions

- So far, we discussed how GD and SGD perform if the function is convex
- Under Lipschitzness, both satisfy $\mathcal{O}(1/\sqrt{T})$ rate to find an approximate optimal solution
- In DL, however, the loss is nonconvex. Thus, we need to relax convexity
- Difficult goal! We thus also relax our notation of convergence to an approximate first-order solution

$$\mathbb{E}[f(\hat{w}) - f(w^*)] \leq \epsilon \Rightarrow \|\nabla f(\hat{w})\|^2 \leq \epsilon \tag{66}$$

- This condition requires the existences of gradient. Thus, instead of Lipschitzness we require another notion of functions' sensitivity called smoothness

Definition

A function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if its gradient is L-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \tag{67}$$

for all x, y and some constant $0 \leq L < \infty$.

- Recall that G-Lipschitzness, i.e. $|f(x) - f(y)| \leq G\|x - y\|$ implied the slope of f is bounded by G.
- Smoothness then means the slope of the ∇f, i.e., the second derivative of Hessian is bounded by L.
- More formally, if the function is twice differentiable, all eigen values of the Hessians are bounded:

$$-L \leq \lambda_d := \lambda_{min}(\nabla^2 f(x)), \ldots, \lambda_1 := \lambda_{max}(\nabla^2 f(x)) \leq L \tag{68}$$

for all x.

Examples

- $f(x) = \frac{1}{2} x^\top A x$

- $f(x) = e^x$

Smoothness and Bounded Curvature

- The condition $|\lambda_i(\nabla^2 f)| \leq L$, $i = 1, \dots d$ means the function has bounded curvature from above and from below
- That is, we can upper bound the function by a convex quadratic function and lower bound the function by a concave quadratic function at any point!
- Leading to the following sometimes more useful equivalent definition of smoothness

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \quad \text{RHS a convex quadratic function of } x \tag{69}$$

$$f(x) + \langle \nabla f(x), y - x \rangle - \frac{L}{2}\|y - x\|^2 \leq f(y) \quad \text{LHS a concave quadratic function of } x \tag{70}$$

Let $\Delta := f(w_1) - f(w^*)$ denote the initial suboptimality.

Theorem (GD)

Assume f is L-smooth. then, GD with $\eta = \frac{1}{L}$ satisfies

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \leq \frac{2L\Delta}{T} = \mathcal{O}(1/T). \tag{71}$$

Theorem (SGD)

Assume f is L-smooth. Furthermore assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, SGD with learning rate $\eta = \min\{\frac{1}{L}, \sqrt{\frac{2\Delta}{L\sigma^2 T}}\}$ satisfies

$$\mathbb{E}_{z_1,\ldots,z_T}[\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2] \leq \frac{2L\Delta}{T} + 2\sigma\sqrt{\frac{2\Delta L}{T}} = \mathcal{O}(1/\sqrt{T}). \tag{72}$$

(S)GD and Smooth functions (Cont'd)

- Dimension-free result
- Regulating role of $\eta = \frac{1}{L}$ for GD and $\eta = \min\{\frac{1}{L}, \sqrt{\frac{2\Delta}{L\sigma^2 T}}\}$ for SGD
- We can use either of 2 approaches to generate an output, that is (i) the last/best parameter w_T and (ii) sampling one iterate at random. The later implies $\mathbb{E}_{\hat{w}}\|\nabla f(\hat{w})\|^2 \leq \frac{2L\Delta}{T}$ for GD and $\mathbb{E}_{\hat{w}}\|\nabla f(\hat{w})\|^2 \leq \frac{2L\Delta}{T} + 2\sigma\sqrt{\frac{2\Delta L}{T}}$ for SGD
- Iteration complexity of $T = \Omega(\epsilon^{-1})$ for GD and $T = \Omega(\epsilon^{-2})$ for SGD
- SGD bound consist of a higher order term (first term matching GD's) and the dominant term (second term due to stochasticity)
- Iteration complexity of $T = \Omega(\epsilon^{-2})$ compared to $T = \Omega(\epsilon^{-1})$ for GD
- Bound tight in general, but with more structures (present in DL optimization) we can show the theoretical of benefits of advance training algorithms.

GD: A new Perspective

- So far we motivated GD from the fixed point calculation and following the direction of descent locally perspectives
- Thanks to smoothness, we can develop a new perspective which also gives us a way to develop better/generalized methods
- Consider minimizing $f(w)$ iteratively. It is desired to update the parameter such that we always make a progress:

$$f(w_{t+1}) \leq f(w_t), \qquad \forall t \tag{73}$$

  That is, our method has the descent property.
- Going one step further, maximizing the progress is preferred:

$$\max_{w_{t+1}} P_t := f(w_t) - f(w_{t+1}) \tag{74}$$

- A general recipe to do so is Majorization-Minimization when minimizing loss/cost, or equivalently Minorization-Maximization if maximizing profit/utility/reward

Majorization-Minimization (MM)

- We want to maximize the P_t as much as possible. We will do so through a surrogate upperbound U_t with the following properties
  - $U_t(w_t) = f(w_t)$ (surrogate is tight at the current solution)
  - $U_t(w) \geq f(w)$ (surrogate is an upperbound on the loss function)
  - $U_t(\cdot)$ is simple to minimize
- Thus, our MM scheme is as follows
  - $w_{t+1} = \arg\min_w U_t(w)$
  - Update the surrogate to go from U_t to the new surrogate U_{t+1} satisfying the above properties
- Examples include K-means clustering and Expectation-Maximization (EM) for solving mixture models.
- We will show GD is also an example of MM for smooth functions

Why those properties?

- The properties of the surrogate function guarantee progress

$$\begin{aligned} P_t = f(w_t) - f(w_{t+1}) &\geq f(w_t) - U_t(w_{t+1}) \qquad \text{upper bound property} \\ &\geq f(w_t) - U_t(w_t) \qquad w_t \text{ is the minimizer of } U_t \\ &\geq f(w_t) - f(w_t) = 0 \qquad \text{tightness property} \end{aligned} \tag{75}$$

- This, in addition to assuming the loss is bounded from below guarantees the method will converge/stop at a stationary point.

GD as MM

- Recall by smoothness we have a convex quadratic upperbound on the loss!
- This could be our surrogate:

$$U_t(w) := f(w_t) + \langle w - w_t, \nabla f(w_t) \rangle + \frac{L}{2}\|w - w_t\|^2 \tag{76}$$

- It is tight at w_t
- It is always an upper bound on f by smoothness
- It is strongly convex (easy to optimize). Just take the gradient and set it equal to zero:

$$\begin{aligned} &\nabla U_t(w) = 0 \\ &0 + \nabla f(w_t) + L(w - w_t) = 0 \\ &w = w_t - \frac{1}{L}\nabla f(w_t) \end{aligned} \tag{77}$$

- This is exactly GD with learning rate $\eta = 1/L$
- More generally, GD with $\eta \leq 1/L$ is an instance of MM

- A procedure to handle constraints (e.g. Projected GD: $w_{t+1} = \mathcal{P}_{\mathcal{X}}(w_t - \eta \nabla f(w_t))$)

$$w_{t+1} = \arg\min_{w \in \mathcal{X}} f(w_t) + \langle w - w_t, \nabla f(w_t) \rangle + \frac{1}{2\eta} \|w - w_t\|^2 \tag{78}$$

- Extension to broad distance/divergence notions (e.g. Mirror Descent)

$$w_{t+1} = \arg\min_{w \in \mathcal{X}} f(w_t) + \langle w - w_t, \nabla f(w_t) \rangle + \frac{1}{2\eta} \mathcal{D}(w, w_t) \tag{79}$$

  Useful when w is a probability vector, e.g. reinforcement learning.

- Extension to arbitrary update vectors g_t

$$w_{t+1} = \arg\min_{w \in \mathcal{X}} f(w_t) + \langle w - w_t, g_t \rangle + \frac{1}{2\eta} \mathcal{D}(w, w_t) \tag{80}$$

  Results in Stochastic (Projected) Gradient Descent, Stochastic Mirror Descent, etc.

- Possible to find better surrogates if f has more structures (e.g., leveraging Hessian's properties).

Let $\Delta := f(w_1) - f(w^*)$ denote the initial suboptimality.

Theorem (GD)

Assume f is L-smooth. then, GD with $\eta = \frac{1}{L}$ satisfies

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \leq \frac{2L\Delta}{T} = \mathcal{O}(1/T). \tag{81}$$

Theorem (SGD)

Assume f is L-smooth. Furthermore assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, SGD with learning rate $\eta = \min\{\frac{1}{L}, \sqrt{\frac{2\Delta}{L\sigma^2 T}}\}$ satisfies

$$\mathbb{E}_{z_1,\ldots,z_T}[\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2] \leq \frac{2L\Delta}{T} + 2\sigma\sqrt{\frac{2\Delta L}{T}} = \mathcal{O}(1/\sqrt{T}). \tag{82}$$

Proof of GD's Theorem

- Study the amount of progress while leveraging smoothness

$$f(w_{t+1}) \leq f(w_t) + \langle w_{t+1} - w_t, \nabla f(w_t) \rangle + \frac{L}{2}\|w_{t+1} - w_t\|^2 \tag{83}$$

- Using GD's update $w_{t+1} = w_t - \eta \nabla f(w_t)$

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) - \eta \langle \nabla f(w_t), \nabla f(w_t) \rangle + \frac{L\eta^2}{2}\|\nabla f(w_t)\|^2 \\ &= f(w_t) - \|\nabla f(w_t)\|^2 \eta (1 - \frac{L\eta}{2}) \\ \Rightarrow P_t &= f(w_t) - f(w_{t+1}) = \|\nabla f(w_t)\|^2 \eta (1 - \frac{L\eta}{2}) \end{aligned} \tag{84}$$

- Side note 1: the larger the gradient, the more progress we make: $P_t = \mathcal{O}(\|\nabla f(w_t)\|^2)$
- Side note 2: for any $\eta < 2/L$ we make progress.
- Side note 3: Maximum progress when $\eta = 1/L$ (progress a concave quadratic function of η).

Proof of GD's Theorem (Cont'd)

- Rearrrange and average over t (requires $\eta < 2/L$)

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(w_t)\|^2 \leq \frac{\sum_{t=1}^{T} f(w_t) - f(w_{t+1})}{T\eta(1-\frac{L\eta}{2})} \tag{85}$$

- Telescoping structure and noting $f(w_1) - f(w_{T+1}) \leq f(w_1) - f(w^*) = \Delta$

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(w_t)\|^2 \leq \frac{\Delta}{T\eta(1-\frac{L\eta}{2})} \tag{86}$$

- Optimizing the bound w.r.t. η gives $\eta = 1/L$ and the stated result.

Proof of SGD's Theorem

- Following a similar approach we use smoothness and SGD's update to study progress

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) + \langle w_{t+1} - w_t, \nabla f(w_t) \rangle + \frac{L}{2}\|w_{t+1} - w_t\|^2 \\ &= f(w_t) - \eta \langle g_t, \nabla f(w_t) \rangle + \frac{L\eta^2}{2}\|g_t\|^2 \end{aligned} \tag{87}$$

  but here w_t's and g_t's are random quantities. We need proper expectations (hence the SFO and independent assumptions)

- First, conditioned on w_t (equivalently $z_1, \ldots, z_{t-1}$)

$$\mathbb{E}_{z_t}[\langle g_t, \nabla f(w_t) \rangle | w_t] = \langle \mathbb{E}_{z_t}[g_t | w_t], \nabla f(w_t) \rangle = \|\nabla f(w_t)\|^2 \tag{88}$$

- Second, conditioned on w_t (equivalently $z_1, \ldots, z_{t-1}$)

$$\begin{aligned} \mathbb{E}_{z_t}[\|g_t\|^2 | w_t] &= \mathbb{E}_{z_t}[\|g_t \pm \nabla f(w_t)\|^2 | w_t] \\ &= \mathbb{E}_{z_t}[\|g_t - \nabla f(w_t)\|^2 | w_t] + \|\nabla f(w_t)\|^2 \leq \|\nabla f(w_t)\|^2 + \sigma^2 \end{aligned} \tag{89}$$

Proof of SGD's Theorem

- Thus, conditioned on w_t (equivalently $z_1, \ldots, z_{t-1}$)

$$\mathbb{E}_{z_t}[f(w_{t+1})|w_t] \leq f(w_t) - \|\nabla f(w_t)\|^2 \eta(1 - \frac{L\eta}{2}) + \frac{L\eta^2\sigma^2}{2} \tag{90}$$

  Side note: No guarantee on the (expected) progress anymore!

- Averaging over t, rearranging (requires $\eta < 2/L$), and taking expectation w.r.t. $z_1, \ldots, z_T$

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{z_{1:T}} \|\nabla f(w_t)\|^2 \leq \frac{\sum_{t=1}^{T} f(w_t) - f(w_{t+1})}{T\eta(1 - \frac{L\eta}{2})} + \frac{L\eta\sigma^2}{2(1 - \frac{L\eta}{2})} \tag{91}$$

- Telescoping structure and noting $f(w_1) - f(w_{T+1}) \leq f(w_1) - f(w^*) = \Delta$

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{z_{1:T}} \|\nabla f(w_t)\|^2 \leq \frac{\Delta}{T\eta(1 - \frac{L\eta}{2})} + \frac{L\eta\sigma^2}{2(1 - \frac{L\eta}{2})} \tag{92}$$

- Optimizing the bound w.r.t. η gives the stated result (doable but a bit cumbersome).

Stochastic Gradient Descent (SGD): Some Examples

Linear Regression

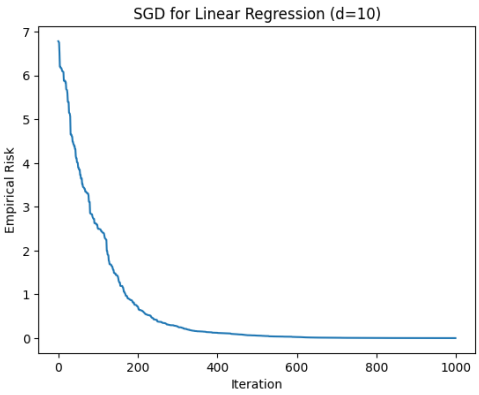
- Linear Regression
  - Features: $x_i \in \mathbb{R}^d$, $i = 1, \ldots, N$
  - Labels: $y_i \in \mathbb{R}$, $i = 1, \ldots, N$
  - Linear dependency: $y = x^\top w + e$ where $e \sim \mathcal{N}(0, \sigma^2)$
  - Square loss: $\ell(w, z) = \frac{1}{2}(y - x^\top w)^2$

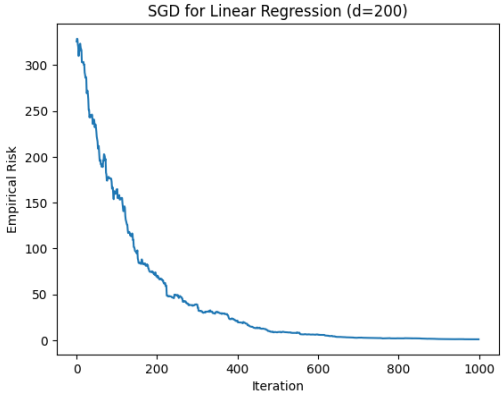
```
## Data Generation
X = np.random.normal(0, 1, (N,d))
w_best = np.random.normal(0, 1, (d, 1))
e = np.random.normal(0, sigma, (N, 1))
y = X @ w_best + e # True model
# Random initialization of weights
w = np.random.normal(0, 1, (d, 1))
# History for loss
loss1 = np.zeros(T)

for i in range(T):
    loss1[i] = np.mean(np.power(y-X@w, 2)) # save loss
    idx = np.random.randint(0, N) # choose a sample randomly (uniform dist.)
    xi, yi = np.expand_dims(X[idx], axis=1), y[idx]
    gw = -2*(yi-w.T @ xi)*xi # calculate gradient
    w -= eta * gw # update the weights
```

Linear Regression (Cont'd)

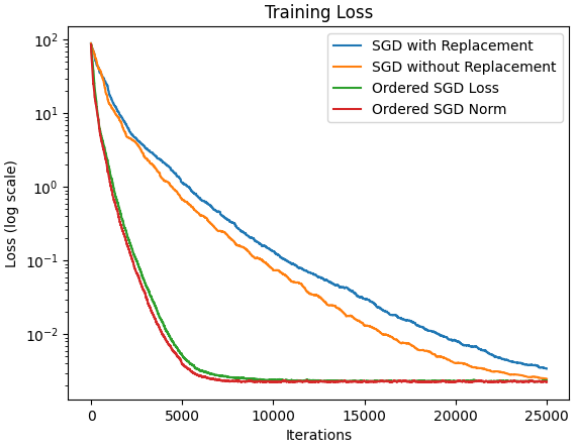
- $N = 100,\ d = 10,\ 200$





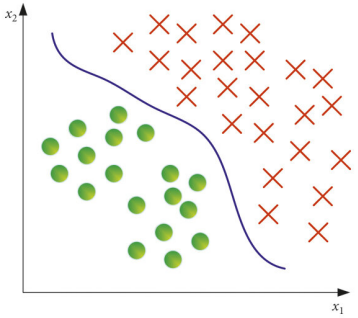
Non-Linear Regression

- $\ell(w, z) = \frac{1}{2}(y - \sigma(x^\top w))^2$
- A simple neural network



Performance-Robustness Tradeoff

- Focusing on harder instances accelerates training by finding the boundary sooner
- Typically measured by loss values $\ell(w_t, z_t^i)$ or gradient values $\|\nabla \ell(w_t, z_t^i)\|$
- Performance-Robustness Tradeoff: More chance that samples near the boundary have noisy labels, thus focus on easy samples for robust training
- Many such tradeoffs between performance and other contexts in ML/DL optimization



High Confidence Guarantees

Average vs High Confidence

- Recall we showed for SGD

$$\mathbb{E}[\text{error}] \leq \text{Bound}(T) \tag{93}$$

  This means if running SGD many many times, on average it perform well

- Not satisfactory, we want good performance after only one-time training. That is, given $0 < \delta < 1$ we want with probability exceeding $1 - \delta$

$$\text{error} \leq \text{Bound}(T, \delta) \tag{94}$$

  or equivalently

$$\Pr\{\text{error} \geq \text{Bound}(T)\} \leq \delta \tag{95}$$

- Can SGD deliver such results?

Stronger Results under Stronger Assumptions

- Recall our SFO assumption (written slightly differently here)

$$g_t = \nabla f(w_t) + e_t, \quad \mathbb{E}[e_t|w_t] = 0, \quad \mathbb{E}[\|e_t\|^2|w_t] \leq \sigma \tag{96}$$

- We can assume the noise is light-tailed (or sub-Gaussian)

$$\mathbb{E}[\exp(\|e_t\|^2/\sigma^2)|w_t] \leq 1 \tag{97}$$

  Example: Gaussian noise, uniformly bounded noise, etc.

- Combined with concentration of measure results (deviation from the mean), e.g., Azuma–Hoeffding, we can show our previous bounds hold with high confidence:

$$\text{error} \leq \text{Bound}(T, \log \frac{1}{\delta}) \tag{98}$$

Martingale Difference Sequence (MDS)

A sequence of random variables $X_1, \ldots, X_T$ is an MDS if $\mathbb{E}[X_t | z_1, \ldots z_{t-1}] = 0$ where $z_1, \ldots, z_T$ is another sequence of random variables

- Think of it as a generalization of zero-mean independent random variables.
- In the context of DL optimization, very often

$$X_t = g_t - \nabla f(w_t), \qquad \mathbb{E}_{z_t}[g_t - \nabla f(w_t) | z_1, \ldots z_{t-1}] = 0 \tag{99}$$

the term we dealt with in the proofs for SGD before.

- When assuming bounded gradient or light-tailed noise, we can show $\mathcal{O}(\sqrt{T \log \frac{1}{\delta}})$ growth of noisy terms

$$\sum_{t=1}^{T} \langle g_t - \nabla f(w_t), \nabla f(w_t) \rangle, \qquad \sum_{t=1}^{T} \eta \|g_t - \nabla f(w_t)\|^2 \tag{100}$$

Stronger Results under Stronger Assumptions (Cont'd)

- Could be violated in practice. A remedy is clipping or normalization:

$$g_t = \nabla f(w_t) + e_t, \quad w_{t+1} = w_t - \eta \min\{\max\{-C, g_t\}, C\} \text{ or } \quad w_{t+1} = w_t - \eta \frac{g_t}{\|g_t\| + \kappa} \quad (101)$$

  for some clipping parameter $C > 0$ and a small constant $\kappa > 0$ for numerical stability
- These will control the noise and could deliver high confidence guarantees
- Connection to advanced training methods (adaptivity) and other contexts (privacy).
- Appear in diverse learning settings, e.g., the PPO method in reinforcement Learning (RL)
- Without such safeguarding mechanisms, the growth of the accumulation of noisy terms could be larger w.r.t. to δ.

Markov Inequality

For any positive random variable X and constant $\lambda > 0$ we have

$$\Pr\{X > \lambda\} \leq \frac{\mathbb{E}[X]}{\lambda} \tag{102}$$

- Let $X = \text{error}$ and $\lambda = \text{Bound}(T)/\delta$
- By Markov

$$\Pr\{\text{error} > \text{Bound}(T)/\delta\} \leq \delta \frac{\mathbb{E}[\text{error}]}{\text{Bound}(T)} \leq \delta \tag{103}$$

or equivalently, with probability at least $1 - \delta$,

$$\text{error} \leq \text{Bound}(T)/\delta \tag{104}$$

- Terrible dependence on confidence δ (i.e., $1/\delta$ vs our desired $\log 1/\delta$)

Boosting (ensemble of) SGD

- We will boost SGD by running multiple SGDs in parallel and performing post-optimization
  - number of parallel runs S, number of samples for post-optimization N
  - Run S versions of SGDs in parallel and call their outputs $\hat{w}_s$, for $s = 1, \ldots, S$
  - Choose the best output $\hat{w}$:

$$\hat{w} = \arg \min_{w \in \{\hat{w}_1, \ldots, \hat{w}_S\}} \|G(w)\|, \qquad G(w) = \frac{1}{N} \sum_{j=1}^{N} \nabla \ell(w, z_j^{po}) \tag{105}$$

  or use function values instead

- Total Complexity: Recall for SGD total oracle/gradient complexity is $\mathcal{O}(BT)$ and $\mathcal{O}(T)$ when B is some constant. For SGD-PO:

$$\text{Oracle complexity } = S \times T + S \times N = (N + T) \times S \tag{106}$$

Theorem

If

$$N = \Omega(\frac{\sigma^2 \log \frac{1}{\delta}}{\epsilon\delta}), \quad T = \Omega(\frac{1}{\epsilon^2}), \quad S = \Omega(\log \frac{1}{\delta}) \tag{107}$$

SGD with Post-Optimization can find a solution $\hat{w}$ such that with probability at least $1 - \delta$ it holds $\|\nabla f(\hat{w})\|^2 \leq \epsilon$.

- Thus, the total complexity in terms of ϵ and δ is

$$\frac{\log^2 \frac{1}{\delta}}{\epsilon\delta} + \frac{\log \frac{1}{\delta}}{\epsilon^2} \tag{108}$$

- Proof relies on showing the ensembling/post-optimization enables boosting the simple Markov concentration result, hence improving the dependence on δ.

Stochastic Gradient Descent (SGD) without the G(radient)

Scenarios with no gradient

- Recall, we typically aim to solve

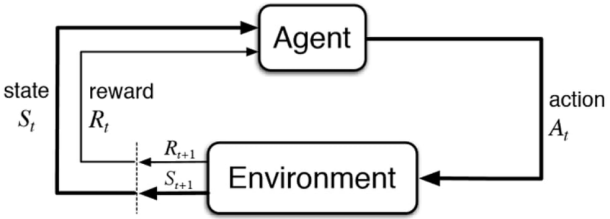
$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) \tag{109}$$

- A natural stochastic gradient is to sample a mini batch of samples $z_t^1, \ldots, z_t^B$ and compute

$$g_t = \frac{1}{B} \sum_{j=1}^{B} \nabla \ell(w_t, z_t^j) \tag{110}$$

- However, we may not be able to, or want to, do this in many scenarios
  - (Online) Reinforcement Learning
  - Adversarial attacks
  - Fine-tuning of Large Language Models (LLMs)

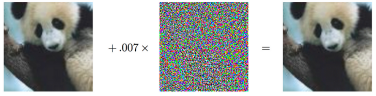
Example: (Online) Reinforcement Learning

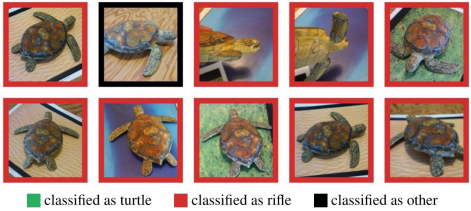


- Agent only gets to see the scalar reward, i.e., the reward for the played action.
- In certain RL models, the task can be reduced to an online optimization problem with linear utility functions $f_t(w) = \langle w, r_t \rangle$ where w is our policy and r_t is the reward vector
- The gradient is r_t but the agent only gets to see $R_t = r_t(i_s)$ where action a_{i_s} is the played action

Example: Adversarial Attacks

- Goal: Given a classifier and a small dataset, make the DL classily images incorrectly



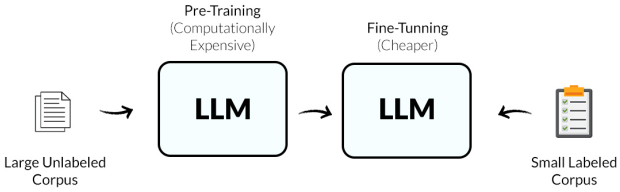


- We only know the images and the returned predictions. Let y_i be the label of image x_i

$$\min_{\delta} \ \text{score}_{w^*}(y_i|c_i+\delta) + \lambda\|\delta\| \tag{111}$$

  for some perturbation parameter δ.
- We do not know the weights of the DL model w^*, hence we cannot use gradient-based methods like SGD that need the gradient $\nabla \text{score}_{w^*}(y_i|c_i+\delta)$.

LLM fine-tuning



- As LLMs grow in size, the substantial memory overhead from gradient computation presents a significant challenge.
- Crucial to address, especially for applications like on-device training where memory efficiency is paramount.

SGD without the G: Intuition

- Let us recall the definition of derivatives for scalar functions

$$\begin{aligned} f'(w) &= \lim_{\mu \to 0} \frac{f(w+\mu) - f(w)}{\mu} \\ &= \lim_{\mu \to 0} \frac{f(w) - f(w-\mu)}{\mu} \\ &= \lim_{\mu \to 0} \frac{f(w+\mu) - f(w-\mu)}{2\mu} \end{aligned} \tag{112}$$

- Let $\mu > 0$ and fixed, then we expect

$$g^{\mu}(w) = \frac{f(w+\mu) - f(w-\mu)}{2\mu} \approx f'(w) \tag{113}$$

- We only need two function evaluations and no gradient calculation
- We expect to incur some bias depending on μ which vanishes as $\mu \to 0$

SGD without the G: Intuition (Cont'd)

- We can view g^{μ} as an expectation over random directions in $\mathbb{R}$ (i.e., left and right)
  - Let $u \sim \mathcal{U}\{-1, 1\}$, i.e., a vector in $\mathbb{R}$ lying on the surface of unit sphere.
  - Consider
$$g^{\mu u}(w) = \frac{f(w + \mu u) - f(w)}{\mu} u \tag{114}$$
    to be a random variable
- Then,
$$\begin{aligned} \mathbb{E}_u[g^{\mu u}(w)] &= \frac{1}{2}\frac{f(w+\mu) - f(w)}{\mu} - \frac{1}{2}\frac{f(w-\mu u) - f(w)}{\mu} u \\ &= \frac{f(w+\mu) - f(w-\mu)}{2\mu} = g^{\mu}(w) \approx f'(w) \end{aligned} \tag{115}$$
  when μ is small.
- Thus, $g^{\mu u}(w)$ is an unbiased estimator of $g^{\mu}(w)$, an approximation of the derivative of our loss function
- How to go to $\mathbb{R}^d$?

Zero-Order Gradient Estimator

- Recall: $u \sim \mathcal{U}\{-1, 1\}$, i.e., a vector in $\mathbb{R}$ lying on the surface of unit sphere.
- In general, we just sample u from the surface of unit sphere in $\mathbb{R}^d$
- Our estimator then becomes

$$g^{\mu u}(w) = \frac{f(w + \mu u) - f(w)}{\mu} u, \qquad u \sim \mathbb{S}_d \tag{116}$$

- Called zero-order estimator as it uses zeroth order information, i.e., function values only as opposed to first order (Gradient) or second order (Hessian) information.
- We can also sample u from a Gaussian distribution $\mathcal{N}(0, I_d)$ (e.g., for simpler mathematical analysis) given that

$$u \sim \mathcal{N}(0, I_d) \Leftrightarrow \frac{u}{\|u\|} \sim \mathbb{S}_d \tag{117}$$

ZO-SGD Method

- Consider $\min_w f(w) := \mathbb{E}_z[\ell(w, z)]$
- Initialize w_1,learning rate η, smoothing rate μ
- For $t = 1, \ldots, T$ do
  - sample a random direction $u_t \sim \mathcal{N}(0, I_d)$ or $u \sim \mathbb{S}_d$
  - sample a mini batch $z_t^1, \ldots, z_t^B$
  - form the update vector

$$g_t^{\mu u} = \frac{\frac{1}{B}\sum_{j=1}^{B} \ell(w_t + \mu u_t, z_t^j) - \ell(w_t, z_t^j)}{\mu} u_t \tag{118}$$

  - update the parameter $w_{t+1} = w_t - \eta g_t^{\mu u}$
- To lower the variance, we can sample multiple (a mini batch of) random directions as well

$$g_t^{\mu u} = \frac{1}{K}\sum_{k=1}^{K} \frac{\frac{1}{B}\sum_{j=1}^{B} \ell(w_t + \mu u_t^k, z_t^j) - \ell(w_t, z_t^j)}{\mu} u_t^k, \qquad u_t^k \sim \mathcal{N}(0, I_d) \tag{119}$$

ZO Estimation as Smoothing

- Recall $g^{\mu u}(w) = \frac{f(w+\mu u)-f(w)}{\mu} u$ and that

$$\mathbb{E}_u[g^{\mu u}(w)] = \frac{f(w+\mu) - f(w-\mu)}{2\mu} = g^{\mu}(w) \tag{120}$$

- So is $g^{\mu}(w)$ the true gradient for any function?
- In turns out

$$g^{\mu}(w) = \nabla f^{\mu}(w), \qquad f^{\mu} := \mathbb{E}_u[f(w+\mu u)] \tag{121}$$

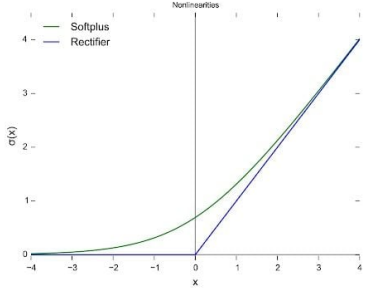
  and f^{μ} is called the smoothed version of f such that when $u \sim \mathcal{N}(0, I_d)$ the technique is called Gaussian Smoothing
- Called smoothed since its sensitivity properties are nicer than the original function f
  - f^{μ} has both lower Lipschitzness and Smoothness parameters
  - f^{μ} is smooth even if f is not! $L(f^{\mu}) \propto 1/\mu$
- μ controls the smoothness-approximation tradeoff

ZO Estimation as Smoothing (Cont'd)

- Recall

$$g^{\mu}(w) = \nabla f^{\mu}(w), \qquad f^{\mu} := \mathbb{E}_u[f(w + \mu u)] \tag{122}$$

- Intuition: Expectation is an integral and integral tends to make the functions smoother/nicer
- Indeed, f^{μ} is nothing but the convolution of f and the PDF of u. Recall: $f * g = \int f(\tau)g(w - \tau) \cdot d\tau$
- This idea is also useful when analyzing DL models with non-smooth activations, e.g., ReLU.



ZO-SGD for f as SGD for f^μ

- Recall $g^\mu(w) = \nabla f^\mu(w)$ and we used an unbiased estimator of $g^\mu(w)$ to minimize f without gradient calculation (ZO-SGD)
- Thus, ZO-SGD is effectively SGD for the following problem

$$\min_w f^\mu(w) := \mathbb{E}_u[f(w + \mu u)] = \mathbb{E}_{z,u}[\ell(w + \mu, z)] \tag{123}$$

- As μ tends to zero the two problem become equivalent
- This idea can be used to prove

Theorem

If $\mu \propto \frac{1}{d\sqrt{T}}$ and $\eta \propto \frac{1}{\sqrt{dT}}$ the iterates of ZO-SGD satisfy

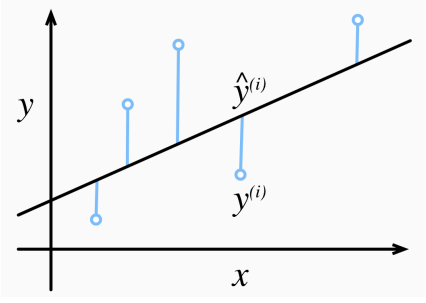
$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{z_{\mathbf{1}:T}, u_{\mathbf{1}:T}} \|\nabla f(w_t)\|^2 = \mathcal{O}\Big(\frac{d}{T} + \sqrt{\frac{d}{T}}\Big). \tag{124}$$

Deep Learning Architectures 1: MLPs

- Recall our simple linear regression model

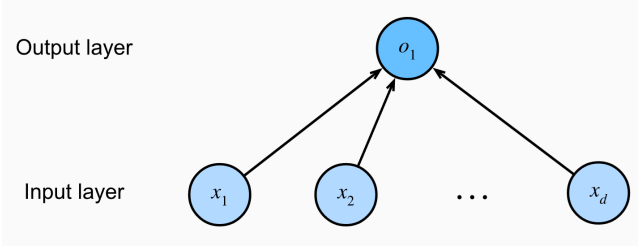
$$y_i = x_i^\top w^* + b^* + e_i \tag{125}$$

- labels $y_i \in \mathbb{R}$, features $x_i \in \mathbb{R}^d$, noise $e_i \in \mathbb{R}$
- regression weights $w^* \in \mathbb{R}^d$, bias $b^* \in \mathbb{R}$
- Previously we ignored the bias term. That is w.l.o.g. as we can increase the dimension to $d+1$ by defining $w^* \leftarrow (w^*, b^*)$ and $x_i \leftarrow (x_i, 1)$



From Linear Regression as DL (Cont'd)

- We can think of this model as a combination of artificial neurons: elements taking an input signal, processing it, and pass it forward
- That is, a simple Artificial neural network (ANN) or a shallow DL model
- The training involves fitting the parameters of the neurons, i.e., the regression parameters, to the data



- Can also be thought as maximum likelihood estimation when the noise is Gaussian and observations are statistically independent

$$\max_{w,b} p(y|x; w, b) := \prod_{i=1}^{n} p(y_i|x_i; w, b) \propto \prod_{i=1}^{n} \exp(-\frac{(y_i - x_i^\top w - b)^2}{2\sigma^2}) \tag{126}$$

- As the log is a monotone function, the above is equivalent to the following negative log likelihood minimization problem

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^\top w - b)^2 \qquad \Leftrightarrow \qquad \text{ERM} \tag{127}$$

- Most advance learning tasks follow a similar template: maximize the likelihood while assuming suitable distributional assumptions.
- Some times we also include a prior (aka a regularization term) and the problem becomes Maximum a priori estimation (MAP)

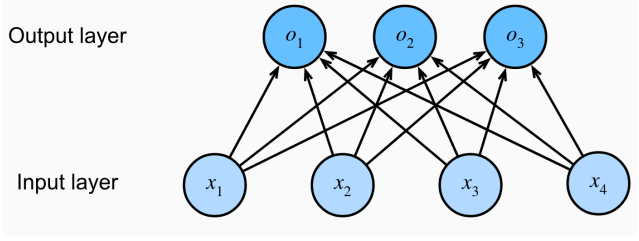
Classification as DL

- Now, assume a classification problem where instead $y_i \in \{1, \dots, C\}$
- We can represent the labels via one-hot encoding:

$$y_i \in \{e_1, \dots, e_C\}, \quad e_i \in \mathbb{R}^C, \quad e_i = (0, 0, \dots, 1, 0, 0) \tag{128}$$

That is, e_i's are the standard unit basis vectors in $\mathbb{R}^C$.

- Again a simple linear relation between features and labels: $y_i = W^* x_i + b^*$ where $y_i \in \mathbb{R}^C$ and $x_i \in R^d$ are labels and features while $W^* \in \mathbb{R}^{C \times d}$ and $b^* \in \mathbb{R}^C$ are weights and bias



- We can use square loss as before, but the solution is not elegant
- Let $\hat{W}$ and $\hat{b}$ denote the parameters of a learned model.
- Consider the predicted label $\hat{y}_i = \hat{W}x_i + \hat{b}$
- Desirable to think of the c's entry of $\hat{y}_i$ as the probability that the label of x is c.
- Thus, we want $\hat{y}_i$ (like the true labels y_i after one-hot encoding) to be nonnegative and sum to one, i.e., be probability vectors in $\mathbb{R}^C$
- We perform Softmax normalization

$$\hat{o} = \hat{W}x + \hat{b}, \qquad \hat{y} = \mathrm{softmax}(\hat{o}), \qquad \hat{y}(j) := \frac{exp(\kappa \hat{o}(j))}{\sum_{c=1}^{C} exp(\kappa \hat{o}(c))} \tag{129}$$

- Motivated by the Boltzmann distribution in Physics to model a distribution over energy states in gas molecules (in our case the states are the finite classes).

Classification and MLE (Cont'd)

- The training involves fitting the parameters of the neurons, i.e., the classification parameters, to the data. But which loss?
- Let us resort to MLE: find the parameters to maximize the likelihood of data
- Given our softmax normalization, the natural observation/noise distribution is multinomial (discrete Boltzmann) distribution

$$\hat{y}(j) = p(j|x) = \frac{exp(\hat{o}(j))}{\sum_{c=1}^{C} exp(\hat{o}(c))} \quad \text{probability of class/state } j \text{ given feature } x$$

$$-\log p(j|x) = -\log \hat{y}(j) = -\sum_{c=1}^{C} y(j) \log \hat{y}(j) \quad \text{only one coordinate of } y(j) \text{ is nonzero}$$

$$= \text{Cross-Entropy}(y, \hat{y}) \tag{130}$$

- A notion of distance between two distributions and related to the Kullback–Leibler (KL) divergence

Entropy, KL, and Cross-Entropy

- Entropy: how random a quantity is (more random, more bits needed to represent it)

$$H(p) = -\sum_{c=1}^{C} p(c) \log p(c) \geq 0 \tag{131}$$

- KL divergence (or relative entropy): how different two distributions are

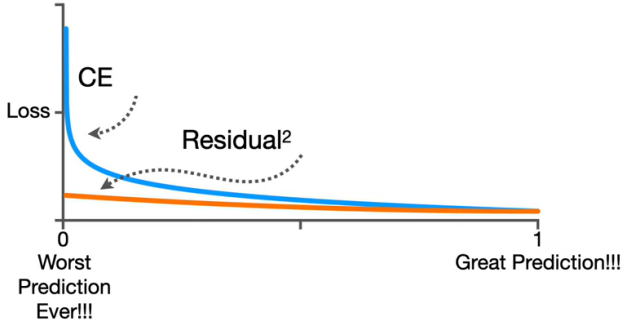
$$D_{kl}(p\|q) = \sum_{c=1}^{C} p(c) \log \frac{p(c)}{q(c)} \geq 0 \tag{132}$$

  An example of f-divergence with convex functions f: $D_f(p\|q) = \mathbb{E}_q[f(\frac{p}{q})]$. For KL, $f(t) = t \log t$.

- Easy to show Cross-Entropy$(y, \hat{y}) = H(y) + D_{kl}(y\|\hat{y}) \geq 0$.
- Thus, minimizing CE loss is equivalent to minimizing KL divergence.

CE vs Square loss

- As discussed CE is more natural than square loss as it enables us to think of lables as probabilities. A direct link to MLE
- CE provides larger feedback sooner



- Consider linear regression, i.e., $\hat{y} = w^\top x$. For training we use SGD to minimize ERM with square loss. By chain rule, Gradient is proportional to the gradient w.r.t. the output $\hat{y}$:

$$\ell(w, z) = \frac{1}{2}(y - \hat{y})^2, \qquad \nabla_w \ell(w, z) = -x(y - w^\top x) \propto \nabla_{\hat{y}} \ell(w, z) = \hat{y} - y \tag{133}$$

- Consider softmax classification, i.e., $\hat{y}(j) := \frac{exp(\hat{o}(j))}{\sum_{c=1}^{C} exp(\hat{o}(c))}$ where $\hat{o} = Wx$. For training we use SGD to minimize ERM with CE loss. By chain rule, Gradient is proportional to the gradient w.r.t. the output $\hat{y}$

$$\ell(w, z) = CE(y, \hat{y}) = -\sum_{c=1}^{C} y(c) \log \hat{y}(c), \qquad \nabla_{\hat{y}} \ell(w, z) = \mathrm{softmax}(\hat{o}) - y = \hat{y} - y \tag{134}$$

- In both cases, the larger the error/residual, the larger the gradient/signal

- Let us explore why we need hidden layers.
- Binary classification with hinge loss

$$y = \text{sign}(\langle x, w \rangle), \qquad \ell(w, z) = \max(0, 1 - y\langle x, w \rangle) \tag{135}$$

- Consider SGD for this problem.
- For $t = 1, \ldots, T$
  - sample $z_t = (x_t, y_t) \sim p_z$
  - If $y_t \langle x_t, w_t \rangle < 0$ perform $w_{t+1} = w_t + \eta y_t x_t$
  - If $y_t \langle x_t, w_t \rangle \geq 0$ perform $w_{t+1} = w_t$
- This is called Perceptron and given a margin $\rho > 0$ between classes has $\mathcal{O}(\rho^{-2})$ guaranteed on the number of mistakes
- What if data non-linearly separable?

Perceptron (Cont'd)

- Assume we have two features $x = (x^1, x^2)$ and $y = \text{xor}(x^1, x^2)$
- data non-linearly separable and Perceptron fails
- Idea: combine multiple Perceptrons

$$o_3 := \text{xor}(x^1, x^2) = x^1 \cdot \bar{x}^2 + x^2 \cdot \bar{x}^1 = (x^1 + x^2) \cdot (\bar{x}^1 + \bar{x}^2) := o_1 \cdot o_2 \tag{136}$$

Symbol	A	B	Q
	0	0	0
	1	0	1
	0	1	1
	1	1	0
Q = A ⊕ B			

Perceptron (Cont'd)

- $o_1 = x^1 + x^2$

x^1	1	0	1	0
x^2	1	0	0	1
o_1	1	0	1	1

- $o_2 = \bar{x}^1 + \bar{x}^2$

$\bar{x}^1$	1	0	1	0
$\bar{x}^2$	1	0	0	1
o_2	0	1	1	1

- $o_3 = o_1 \cdot o_2$

o_1	1	0	1	0
o_2	1	0	0	1
o_3	1	0	0	0

Multi-Layer Perceptron (MLP)

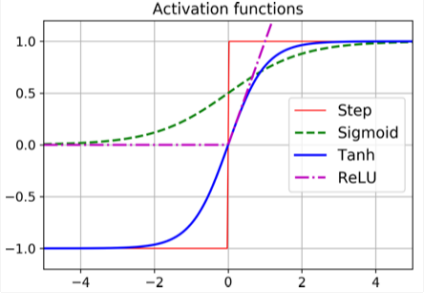
- Each rule can be learned by a single Perceptron. To be thought as a neuron with sign as the activation

$$\text{output} = \text{sign}(\text{input} \times \text{weight}) \tag{137}$$

- Thus, to learn the final label rule we are building a Multi-Layer Perceptron (MLP)
- MLP enjoys a higher learnability capacity (characterized by VC dimension) than a single Perceptron, thus able to handle complex data
- VC dim: Size of the largest dataset of points in general positions s.t. our model can classify it regardless of label assignment
- VCdim of Perceptron is roughly d but for MLP it is roughly Ld when having L hidden layers.

Activations

- Sigmoid: $\sigma(x) = \frac{1}{1+\exp(-x)}$
- Tanh: $\sigma(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$
- ReLU: $\sigma(x) = \max(0, x)$
- ReLU leads to more efficiency in computation and stability in training
- Activations are applied coordinate-wise



Why Nonlinear Activations?

- Two hidden layers in an MLP with linear activations

$$h_1 = W_1 x + b_1, \qquad h_2 = W_2 h_1 + b_2 \tag{138}$$

- These hidden layers will collapse into one

$$\tilde{h} = \tilde{W} x + \tilde{b}, \qquad \tilde{W} = W_2 W_1, \qquad \tilde{b} = W_2 b_1 + b_2 \tag{139}$$

- Thus, intuitively, we have lost the increase in the learning capacity we gained by having hidden layers

Deep Learning Architectures 2: CNNs

Harnessing Structures

- MLPs do not exploit the relation between the features (aka the prior knowledge)
- As a result, they grow very rapidly in size by the input dimension. For instance, 10^8 parameters for an MLP with 100 neurons processing 1 mega pixel image.
- Images, and other type of data, typically have two important structures
  - Translation Invariance: Moving an object does not change the object
  - Locality: Nearby pixels are related
- Turns out using convolution/cross-correlation could help leveraging these structures and reduce the number of parameters drastically.

Cat

Cat

- Consider the pre-activation result of the i-th neuron

$$v_i = \sum_j W_{ij} x_j + b_i = \sum_k W_{i(i+k)} x_{i+k} + b_i = \sum_k \omega_i^k x_{i+k} + b_i \tag{140}$$

- Translation invariance implies that a shift in the input should simply lead to a shift in the output
- This is only possible if ω and b are independent of specific location i:

$$v_i = \sum_k \omega^k x_{i+k} + b \tag{141}$$

effectively a convolution/cross-correlation

CNN: Example 1

- Let $x \in \mathbb{R}^5$ and $v \in \mathbb{R}^3$ such that $v = Wx + b$ has $3 + 3 \times 5 = 18$ learnable parameters in a regular MLP.
- With CNN, the number of parameters will reduce to $3 + 1 = 4$

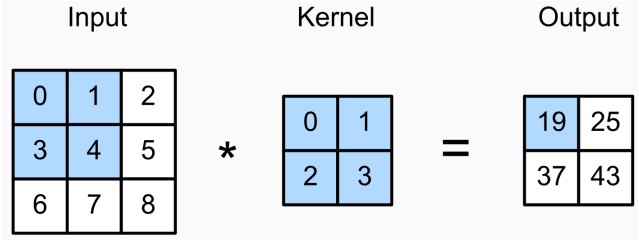
$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & 0 \\ 0 & 0 & \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} \quad \text{i.e. } v_i = b + \sum_{k=-1}^{1} \omega_k x_{i+k},\ i = 1, 2, 3 \qquad (142)$$

each color denoting a learnable parameters

- Thus, we are doing weight sharing and zero-ing out some other weights
- Convolution filter/kernel: $\bullet\bullet\bullet$, output called a feature map
- Note that in CNNs, the size of input, output, and ω are constrained:

$$d_{out} = d_{in} - d_\omega + 1 \qquad (143)$$

CNN: Example 2

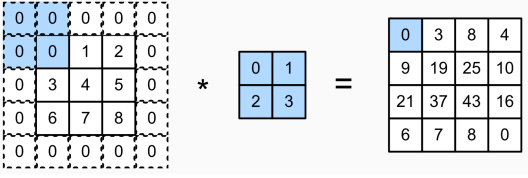


CNN Operations

- Locality: information relevant to the i-th output is localized. Thus, limit the range of sum in the convolution

$$v_i = b + \sum_{k=-\mathcal{O}(d)}^{\mathcal{O}(d)} \omega_k x_{i+k} \Rightarrow v_i = b + \sum_{k=-\Delta}^{\Delta} \omega_k x_{i+k} \tag{144}$$

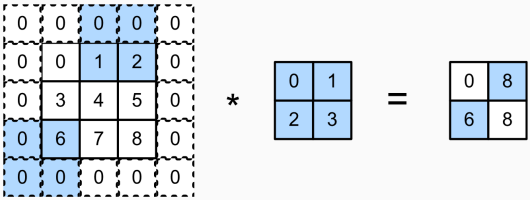
  where $\Delta = \mathcal{O}(1)$, e.g., $\Delta = 3, 5, 7, 9$ (typically odd). Thus, $d_\omega = 2\Delta + 1$
- Padding: Gaining control on output size and ensuring the preservation of edge information



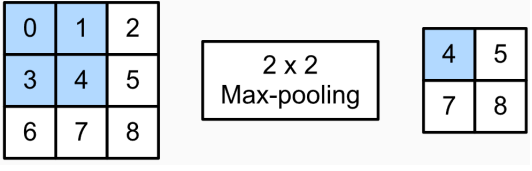
- Output size is $d_{in} - d_\omega + 1 + d_{padding}$
- $d_{padding} = d_\omega - 1$ ensures output has the same size as input.
- odd Δ to ensure even padding on each side

CNN Operations (Cont'd)

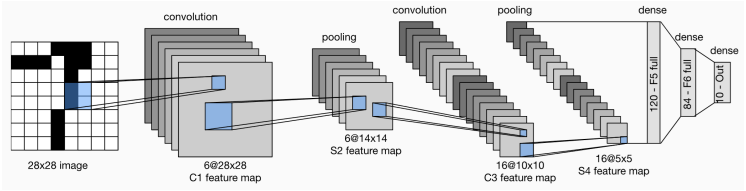
- Stride: More aggressive size reduction by shifting the kernel more than one location (say d_s location)

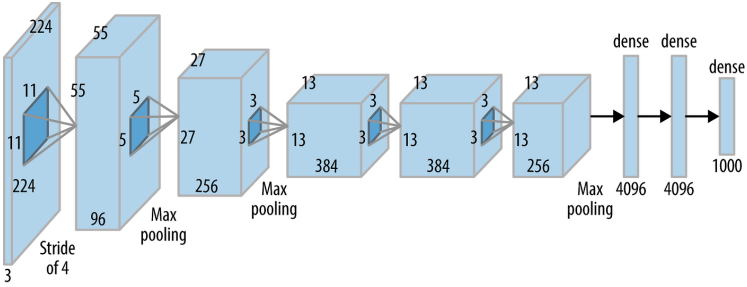


- Output size is $\frac{d_{in}-d_{\omega}+d_{padding}+d_s}{d_s}$
- Max and average pooling: Further parameter-free downsampling

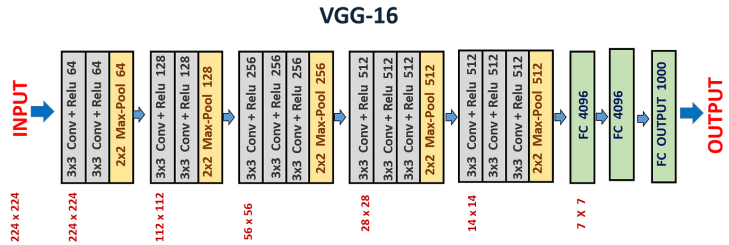


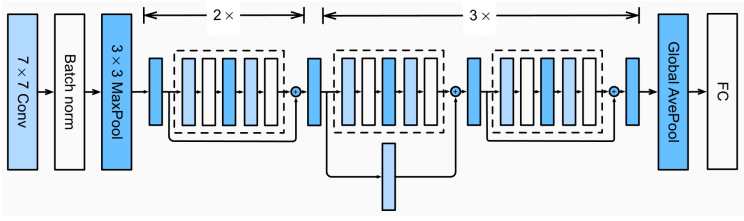
LeNet and AlexNet





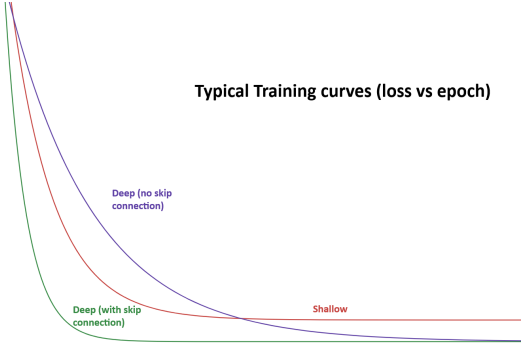
VGG and ResNets (Networks with Blocks)





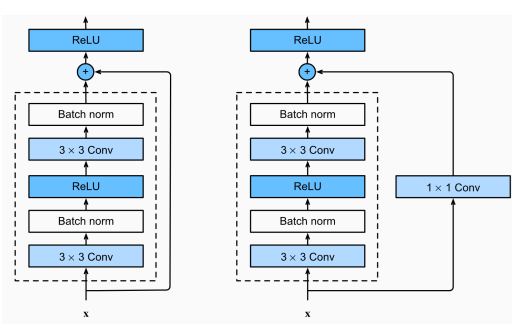
Residual/Skip Connection

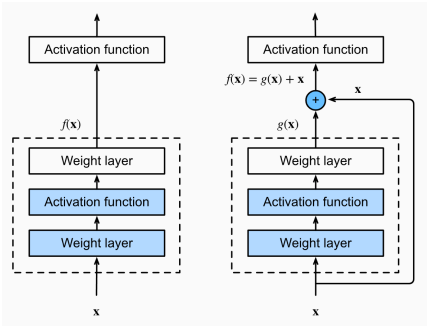
- We typically initialize NNs randomly
- In the early training as depth gets larger almost no useful signal goes from the input to the output and vice versa
- Residual connection helps carrying the information over while enjoying the high capacity of deep NNs



Residual Block

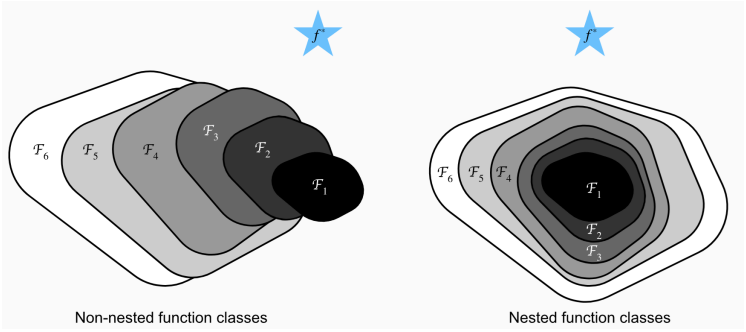
- Resnets constructed as a modular connection of Residual blocks
- Name: Each block learns to model the residual $g(x)$ (difference of input and output). As the ideal residual is small, learning the residual might be easier than learning the true unknown feature-label relation $f(x)$
- Used in other architectures, e.g., transformers





Residual as Nested Function Classes

- The modularity leads to nested capacity as depth increases
- Thus, deeper models are highly expected to outperform shallower ones



Connection to ODEs

- Consider an ODE: $\frac{dx(t)}{dt} = f(t, x)$
- if f known, Euler's method can be used to model the evolution of x and solve the ODE: $x_{t+1} = x_t + f(t, x_t)$
- This is very similar to residual learning: $x_{t+1} - x_t = f(t, x_t)$
- ResNets useful in solving and modeling ODEs and other physical governing equations, an area referred to as Physics-informed DL

Deep Learning Architectures 3: RNNs

- Recall, we discussed the connection between ERM and MLE

$$\min_w f(w) := \frac{1}{n}\sum_{i=1}^{n} \ell(w,(x_i,y_i)) \quad \Leftrightarrow \quad \max_w \prod_{i=1}^{n} p(y_i|x_i;w) \tag{145}$$

  in the context of classification and regression

- Many applications we work with sequences $x_1, x_2, \ldots$ where $x_i \in \mathbb{R}^p$ are observations at time t (e.g., working with financial data, language modeling, DNA data, etc.)
- Natural to assume a causal model where each observation depends on the previous ones $x_t = A(x_1, \ldots, x_{t-1}; w^*)$. An approach referred to as auto-regressive (AR) models
- Natural to follow a similar likelihood maximization approach to find w^*. For instance,

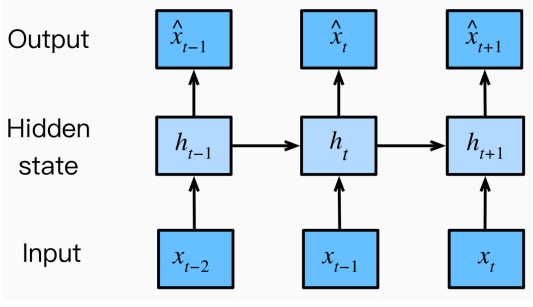
$$x_{t+1} = w(1)x_1 + w(2)x_2, \ldots, +w(t)x_t \tag{146}$$

- Issue: too many features, i.e. $\mathcal{O}(t)$ when t grows

- A solution is to limit the dependency of x_t to only $x_{t-\tau}, \ldots, x_{t-1}$. More generally, we aim to find a latent variable that effectively summarizes the dependencies of the observations

$$x_t = A(h_{t-1}; w^*), \qquad h_t = g(h_{t-1}, x_t; \omega^*) \tag{147}$$

- We can represent these functions by NNs and learn their parameters by fitting the data

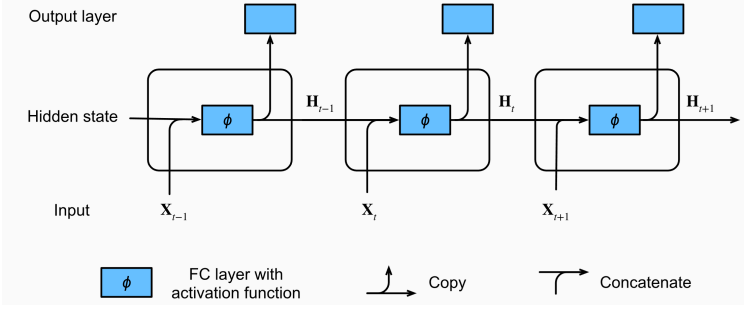


Simple RNN

- For instance, let us resort to MLPs $h_i = \sigma(W_i h_{i-1} + b_i), \quad i = 1, \ldots, L, \quad h_0 = x$
- For latent AR modeling

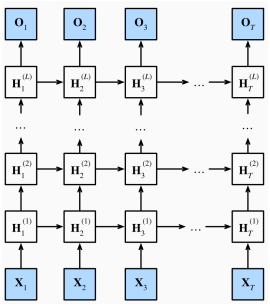
$$h_t = g(h_{t-1}, x_t; \omega^*) \quad \Leftrightarrow \quad h_t = \sigma(W h_{t-1} + b + \tilde{W} x_t) \tag{148}$$

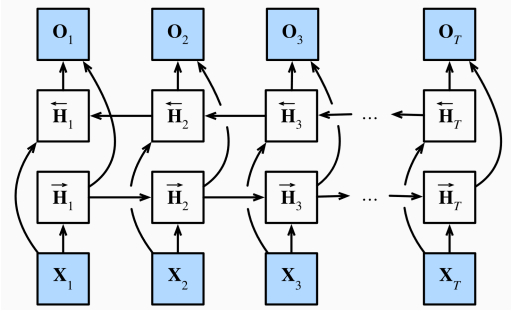
called an RNN with one hidden layer where $\omega^* = (W, \tilde{W}, b)$. Notably, the parametrization cost of an RNN does not grow as the number of time steps increases.



RNN++

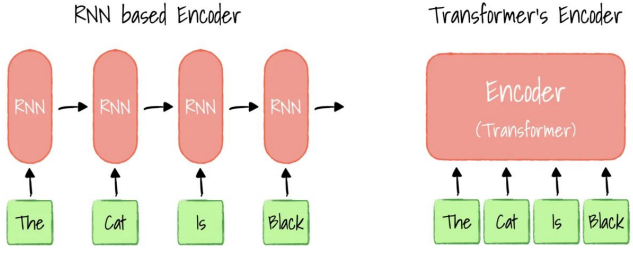
- Certain improvements to RNN, e.g. LSTM, GRU, to deal with the information/signal loss when working with long sequences and long-term dependencies. Solutions similar in nature to skip/residual connections
- We can also stack RNNs and form multiple hidden layers
- We can process the sequences from both directions to learn the dependencies more effectively, e.g. bidirectional LSTM





From RNNs to Transformers

- RNNs like CNNs and MLP need to work with data of a predefined size
- For RNNs, the sequence length should then be fixed. Thus, could limit their applicability
- RNNs further suffer from the need for sequential computation, not able to leverage advances parallel computing
- Transformers leverage the attention mechanism and positional encoding to alleviate these issues (at the cost of higher parallel computational cost)

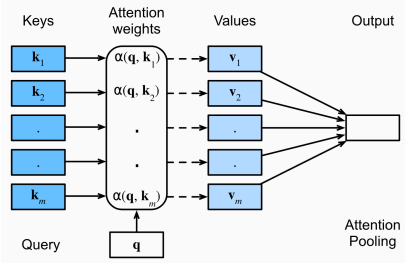


Attention Mechanism

- Idea based on databases, i.e. collections of keys k and values v
- We can make a query q regardless of the database size
- Instead of returning one of the values in the database, attention typically returns

$$\text{Attention}(q, D) = \sum_{i=1}^{m} \alpha(q, k_i) v_i, \quad \alpha(q, k_i) = \frac{exp(\langle q, k_i \rangle / \sqrt{d})}{\sum_{j=1}^{m} exp(\langle q, k_j \rangle / \sqrt{d})} \tag{149}$$

  i.e., using softmax to ensure weights are nonnegative and sum to one (convex combination).
- In practice, q, v, k are matrices and include learnable weight matrices.



Automatic Differentiation

Motivation

- Calculating derivatives is the crucial step in all the training algorithms
- Modern deep learning frameworks automate this process by offering automatic differentiation
- Two crucial ideas are back-propagation method and computational graphs
- Computational graph tracks how each value depends on others and automatic differentiation works backwards through this graph applying the chain rule (back-propagation)

Back-propagation for MLP

- Consider our canonical training task

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i) \tag{150}$$

for a simple MLP and a regression task

$$\ell(w, z_i) = \frac{1}{2}\|y_i - W_L\sigma(W_{L-1}\dots W_2\sigma(W_1 x_i))\|^2, \quad x_i \in \mathbb{R}^p, \quad y_i \in \mathbb{R}^m \tag{151}$$

- Thus, our learnable parameters are $w = (W_1, \dots, W_L) \in \mathbb{R}^d$
- And SGD update is $w_{t+1} = w_t - \eta g_t$ with g_t representing the gradient of the mini batch used in iteration t
- Our goal is to find partial derivatives w.r.t. each weight matrix using chain rule (treating each W_l as a $d_l \times d_{l-1}$ column vector such that $\partial_{W_L}\ell$ is also a $d_l \times d_{l-1}$ column vector).

Back-propagation: defining Variables

- Define pre-activation vectors v and post-activation vectors h

$$v_l = W_l h_{l-1}, \quad h_l = \sigma(v_l), \quad v_l, h_{l-1} \in \mathbb{R}^{d_l}, \quad l = 1, \ldots, L \tag{152}$$

where $h_0 = x \in \mathbb{R}^d$ and d_l denotes the number of neurons in layer l. Note v_L denotes model's prediction (output layer).

- Define the final error signal

$$e_L = \partial_{v_L} \ell = \partial_{v_L} \frac{1}{2} \|y - v_L\|^2 = v_L - y, \quad e_L \in \mathbb{R}^{d_L} \tag{153}$$

- Define activation derivative matrix

$$D_l = \text{diag}(\sigma'(v_l(1)), \ldots \sigma'(v_l(d_l))) \in \mathbb{R}^{d_l \times d_l} \tag{154}$$

Partial Derivatives

- We can represent each partial derivative using our defined variables. For instance,

$$\partial_{W_L} \ell = \partial_{W_L} v_L \times \partial_{v_L} \ell = \underbrace{I_{d_{L-1}} \otimes h_{L-1}}_{(d_L \cdot d_{L-1}) \times d_L \text{ matrix (Kronecker product)}} \times \underbrace{v_L - y}_{= e_L,\ d_L \times 1 \text{ vector}} \tag{155}$$

- Example:

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \in \mathbb{R}^2, \quad W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \end{bmatrix} \in \mathbb{R}^{3 \times 2}.$$

$$I_3 \otimes a = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix} \in \mathbb{R}^{6 \times 3}.$$

Partial Derivatives (Cont'd)

- Recall $v_l = W_l h_{l-1}, \quad h_l = \sigma(v_l), \quad v_l, h_{l-1} \in \mathbb{R}^{d_l}, \quad l = 1, \ldots, L$
- For layer $L-1$

$$\partial_{W_{L-1}} \ell = \underbrace{\partial_{W_{L-1}} v_{L-1}}_{h_{L-2}} \times \underbrace{\partial_{v_{L-1}} h_{L-1}}_{\text{derivative of activation } D_{L-1}} \times \underbrace{\partial_{h_{L-1}} v_L}_{W_L} \times \underbrace{\partial_{v_L} \ell}_{e_L} \tag{156}$$

where we may need operations like $I\otimes$ and $^\top$ to ensure dimensions match and the final answer is a $d_{L-1} \cdot d_{L-2}$ vector

- Consider defining

$$e_{L-1} := \partial_{v_{L-1}} \ell = \underbrace{\partial_{v_{L-1}} h_{L-1}}_{\text{derivative of activation } D_{L-1}} \times \underbrace{\partial_{h_{L-1}} v_L}_{W_L} \times \underbrace{\partial_{v_L} \ell}_{e_L} \tag{157}$$

by chain rule

Partial Derivatives (Cont'd)

- Thus

$$\partial_{W_{L-1}} \ell = \underbrace{\partial_{W_{L-1}} v_{L-1}}_{h_{L-2}} \times e_{L-1} \tag{158}$$

that is the partial derivative for each layer l is just the product of the activation/output of the previous layer h_{l-1} and the error of that layer e_l

- Gives us a natural procedure to leverage memory to reduce computational cost
- This procedure is called Back-propagation:
  - compute pre/post activations v_l, h_l in the forward pass and store them
  - compute and propagate the error signals in the backward pass

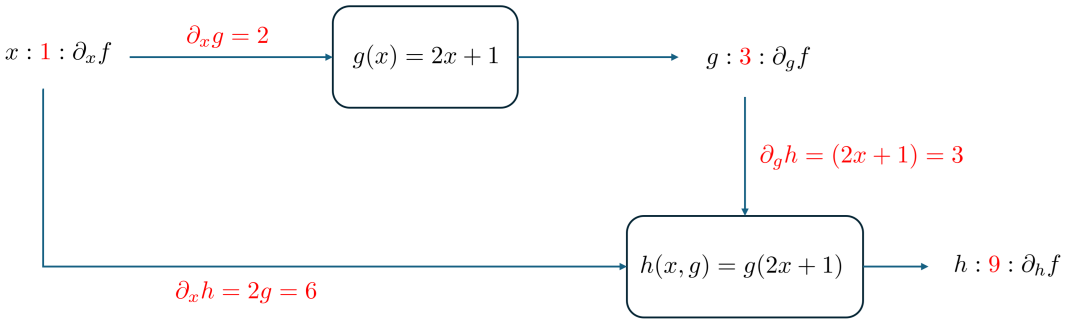
$$e_{l-1} = D_{l-1} \times W_l \times e_l, \quad l = 1, \ldots, L \tag{159}$$

Computation Graph

- A computation graph is a directed acyclic graph (DAG) that represents the sequence of operations applied to variables in a computational framework, such as deep learning.
- Computational graph tracks how each value depends on others and automatic differentiation works backwards through this graph applying the chain rule (back-propagation)
- Example $f(x) = (2x + 1) \cdot (2x + 1)$
- We think of this as $h(x, g) = g(2x + 1)$ where $g(x) = 2x + 1$
- We aim to calculate the derivative of f at $x = 1$

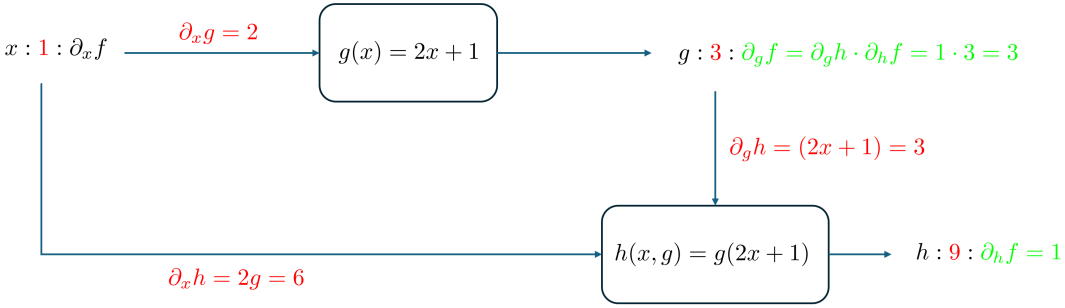
Computation Graph Example

- Forward pass
  - find the values of input variables (pre/post activations h_l, v_l)
  - label the edges with relevant partial derivatives (activation derivatives D_l)



Computation Graph Example (Cont'd)

- Backward pass
  - move backward to calculate the derivatives w.r.t. activations (error signals e_l's)
  - accumulate the derivatives and sum over all path



$$\partial_x f = \partial_x h \cdot \partial_h f + (\partial_h f \cdot \partial_g h) \cdot \partial_x g = 1 \cdot 6 + 2 \cdot (3 \cdot 1) = 6 + 6 = 12$$

Static Computation Graph

- The entire computation is defined and compiled before execution (a define-then-run approach), e.g. earlier versions of Tensorflow
- Advantages
  - More optimizable, as the entire computation graph is known in advance.
  - Allows for efficient graph-level optimizations, such as operation fusion
  - Can be deployed more easily for inference after compilation.
- Disadvantages
  - Less flexible: modifying the graph requires redefining and recompiling it
  - Debugging is harder since the execution is decoupled from definition.

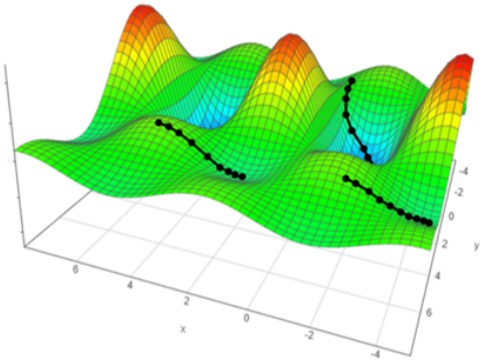
Dynamic Computation Graph

- Built on-the-fly during execution. The structure of the graph can change at each iteration (a define-by-run approach), e.g. Tensorflow 2.0 and Pytorch
- Advantages
  - Highly flexible: enables dynamic control flows, such as loops and conditionals.
  - Easier to debug because the execution happens immediately.
  - More intuitive for model development, especially for recurrent networks and reinforcement learning.
- Disadvantages
  - Potentially less optimized since the entire graph is not known beforehand.
  - Can be slower compared to static graphs due to runtime overhead.

Initialization and Normalization

Why Initialization Matters

- DL architectures are complex and leading to highly complex and nonconvex loss landscapes
- Initialization tends to determine the quality of of the solution found by iterative methods
- Initialization tends to impact the gradient information used by SGD throughout training



- Consider an MLP

$$h_j = \sigma(W_j h_{j-1} + b_j) \in \mathbb{R}^{d_j}, \quad , j = 1, \ldots, L, \quad , h_0 = x \in \mathbb{R}^d, \quad , h_L = y \in \mathbb{R} \tag{160}$$

where $b_j \in \mathbb{R}^{d_j}$ and $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$ are learnable parameters of layer j.

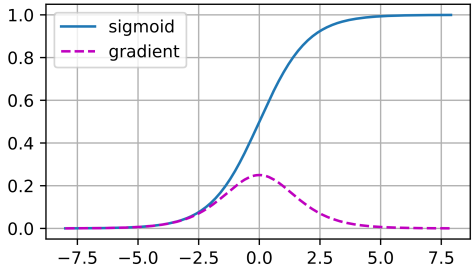
- Gradient is our learning signal and used in SGD for training.
- By Chain rule and treating each W_j as a $d_j^{j-1} := d_j \times d_{j-1}$ vector,

$$\begin{aligned} \partial_{W_j} \text{loss} &= \partial_{W_j} h_L \times \partial_{h_L} \text{loss} = \partial_{W_j} h_j \cdot \partial_{h_j} h_{j+1} \ldots \partial_{h_{L-2}} h_L \cdot \partial_{h_{L-1}} h_L \times \partial_{h_L} \text{loss} \\ &= \partial_{W_j} h_j \times M_{j+1} \times \ldots M_{L-1} \times M_L = \partial_{W_j} h_j \times \prod_{k=j+1}^{L} M_k \times \partial_{h_L} \text{loss} \end{aligned} \tag{161}$$

Side note: that the matrix $\partial_{W_j} h_L$ has dimension $d_j^{j-1} \times d_L$, the matrix $\partial_{W_j} h_j$ has dimension $d_j^{j-1} \times d_j$, while matrices M_k have dimensions $d_{k-1} \times d_k$. thus, dimension of both sides match.

Vanishing/Exploding Gradient Problems

- Our learning signal is proportional to $\prod_{k=j+1}^{L} M_k$, a product that could be very large or very small depending on the these matrices (and notably their eigen values).
- Vanishing gradient: Very small product means we cannot move very far from the initialization (happening with deep networks and RNNs; fixes include LSTM, skip connection).
- Exploding gradient: Very large product means divergence and instability (happening with deep networks; fixes include gradient clipping)
- Activation choice also plays a role



Another Problem: Symmetry and Limited Capacity

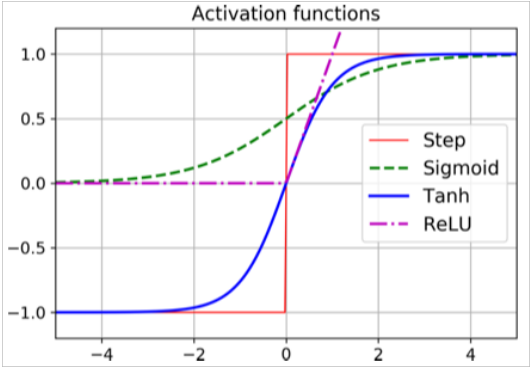
- Recall the motivation behind introducing nonlinearity and deep models was increased learning capacity
- With a bad initialization, we cannot achieve this. Consider

$$\begin{bmatrix} h(1) \\ h(2) \end{bmatrix} = \sigma\left(\begin{bmatrix} W_{11}x(1) + W_{12}x(2) + b(1) \\ W_{21}x(1) + W_{22}x(2) + b(2) \end{bmatrix} \right) \tag{162}$$

- Assume our initialization is such that $W_{11} = W_{12} = W_{21} = W_{22} = \alpha$ and $b(1) = b(2) = \beta$. thus, $h(1) = h(2)$
- then, the gradients $\partial_{W_{ij}} h(1)$ and $\partial_{W_{ij}} h(2)$ are equal. Thus, using gradient-based updates, W_{ij}'s will be the same, and so will the outputs
- Essentially, due to a symmetry, our two neurons reduce to only one neuron, hence limited capacity
- Randomization could help to solve this and the vanishing/exploding gradient at the same time.

Xavier Initialization

- Idea: Weights initialized such that the signal strength moving from one side to the other side in the network is preserved
- Thus, no drastic reduction or amplification leading to vanishing and exploding gradient, respectively
- Let us consider linear activation of the linear regime of all activations (small signal regime)



Xavier Initialization (Cont'd)

- Consider a layer with $W \in \mathbb{R}^{d_{out} \times d_{in}}$ and call the input and outputs x and o for simplicity.
- Forward information propagation condition

$$\mathbb{E}\|o\|^2 = \sum_{i=1}^{d_{out}} \mathbb{E}[o_i^2] = \sum_{i=1}^{d_{out}} \mathbb{E}[\sum_{j=1}^{d_{in}} W_{ij} x_j]^2 \tag{163}$$

- Backward information propagation condition

$$\mathbb{E}\|x\|^2 = \sum_{j=1}^{d_{in}} \mathbb{E}[x_j^2] = \sum_{j=1}^{d_{in}} \mathbb{E}[\sum_{i=1}^{d_{out}} W_{ij} o_i]^2 \tag{164}$$

- We assume $\mathbb{E}[x_j^2] = \gamma_{in}^2$, $\mathbb{E}[o_i^2] = \gamma_{out}^2$, $\mathbb{E}[W_{ij}^2] = \sigma^2$, and that W_{ij}'s are zero mean
- Thus, our conditions imply

$$d_{out}\gamma_{out}^2 = d_{out} d_{in} \sigma^2 \gamma_{in}^2, \qquad d_{in}\gamma_{in}^2 = d_{in} d_{out} \sigma^2 \gamma_{out}^2 \tag{165}$$

Xavier Initialization (Cont'd)

- Imposing $\gamma_{in}^2 = \gamma_{out}^2$

$$\begin{aligned} \gamma_{out}^2 &= d_{in}\sigma^2\gamma_{in}^2, \qquad \gamma_{in}^2 = d_{out}\sigma^2\gamma_{out}^2 \\ 1 &= d_{in}\sigma^2, \qquad 1 = d_{out}\sigma^2 \end{aligned} \tag{166}$$

- Thus, $\sigma^2 = 1/d_{in}$ and $\sigma^2 = 1/d_{out}$. As $d_{in} \neq d_{out}$, we instead set

$$\sigma^2 = \frac{1}{\mathrm{avg}(d_{in}, d_{out})} \qquad W_{ij} \sim \mathcal{N}(0, \sigma^2), \qquad \text{or} \quad W_{ij} \sim \mathcal{U}(-\sigma, \sigma) \tag{167}$$

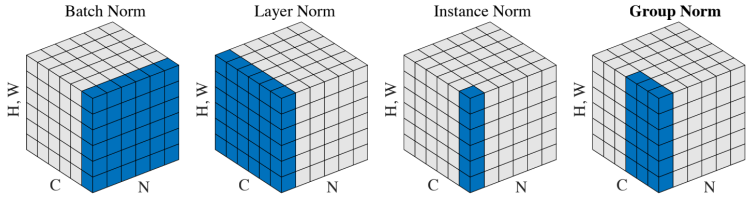
- Averaging can be arithmetic $(d_{in} + d_{out})/2$ or geometric mean $\sqrt{d_{in} \times d_{out}}$
- For ReLU's we typically increase σ^2 by a factor of 2 to compensate for the fact that half of the signal is dropped

$$\mathbb{E}[x_j^2] = \gamma_{in}^2 \quad \Leftrightarrow \quad \mathbb{E}[\max(0, x_j^2)] = \gamma_{in}^2/2 \tag{168}$$

for a symmetric input x.

Normalization Methods

- Xavier Initialization essentially performs a normalization at initialization
- As the weights get updated, weights will change, and so will the strength of layer's output/activation
- Normalization methods extend this idea to normalize the outputs throughout training.
- Certain connection to adaptive training methods to ensure parameters are varying more or less with the same rate
- Normalizing activations could lead to "smaller gradients", hence lower effective Lipschitzness and smoothness parameters G and L, thereby accelerating convergence.



Batch Normalization

- Consider a batch $\mathcal{B}$ of B vectors: $u_1, \ldots, u_B \in \mathbb{R}^p$ (p = $H \times W$ for a $H \times W$ matrix)
- Think of these as activations associated with samples/data used in minibatch SGD
- Batch normalization utilizes a data-driven operation

$$\mathrm{BN}(u) = \alpha \odot \frac{u - \mu_B}{\sigma_B + \kappa} + \beta, \quad \forall u \in \mathcal{B} \tag{169}$$

here $\alpha, \beta \in \mathbb{R}^p$ are learnable parameters and

$$\mu_B = \frac{1}{B} \sum_{u \in \mathcal{B}} u \in \mathbb{R}^p, \qquad \sigma_B^2 = \frac{1}{B} \sum_{u \in \mathcal{B}} \|u - \mu_B\|^2 \in \mathbb{R} \tag{170}$$

are sample mean and variance.

- $\kappa > 0$ introduced for numerical stability and avoiding division by zero.
- Moderate B to avoid loss of information (small batch) and slow computation (large batch)

Batch Normalization (Cont'd)

- Moderate B to avoid loss of information (small batch) and slow computation (large batch)
- Moderate B leads to a noisy calculation during the training leads to better generalization.
- How to apply in conjunction with other operations?

$$h = \text{Dropout}(\sigma(\text{BN}(Wx + b))) \tag{171}$$

- During inference, we use the statistics of the entire data to calculate the mean and variance of each layer's activation

$$\mu_{\mathcal{D}} = \frac{1}{n}\sum_{u\in\mathcal{D}} u \in \mathbb{R}^p, \qquad \sigma^2_{\mathcal{D}} = \frac{1}{n}\sum_{u\in\mathcal{D}} \|u - \mu_{\mathcal{D}}\|^2 \tag{172}$$

Which can be approximated on the fly from mini-batch statistics found during training:

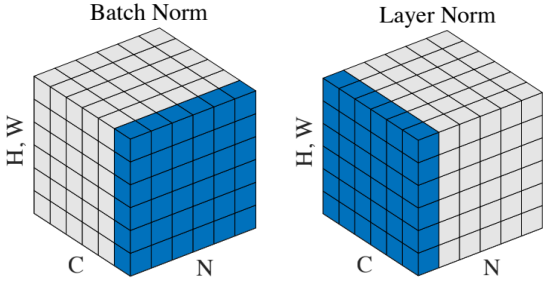
$$\mu_{\mathcal{D}} \leftarrow \frac{1}{t}\mu_{B_t} + \frac{t}{t+1}\mu_{\mathcal{D}}, \qquad \sigma^2_{\mathcal{D}} \leftarrow \frac{1}{t}\sigma^2_{B_t} + \frac{t}{t+1}\sigma^2_{\mathcal{D}} \tag{173}$$

Layer Normalization

- A deterministic operation normalizing each vector u individually

$$\mathsf{LN}(u) = \frac{u - \mu_u 1}{\sigma_u + \kappa} \in \mathbb{R}^p, \quad \mu_u = \frac{1}{p}\sum_{i=1}^{p} u(i) \in \mathbb{R}, \quad \sigma_u = \frac{1}{p}\sum_{i=1}^{p} |u(i) - \mu_u|^2 \in \mathbb{R} \tag{174}$$

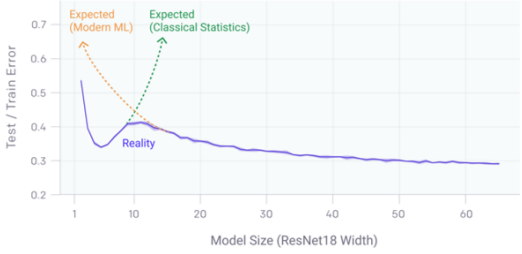
- Almost scale free $\mathsf{LN}(u) = \mathsf{LN}(\alpha u)$, a property useful for stabilizing training
- Can be applied to image-like quantities with multiple channels (e.g., $C = 3$)



Regularization

Motivation

- We wish to optimize training loss to build models that generalize beyond the training set.
- Classical ML theory suggests reduction on generalization error tend to be as a result of regularization: constraining the set of values that our parameters might take (model capacity reduction)
- This view has let to certain approaches which we will view as noise injection
- However, in practice, our models have high capacity that can fit entire data (overparameterization and interpolation). And by doing so, despite classical ML prediction, generalize well too.

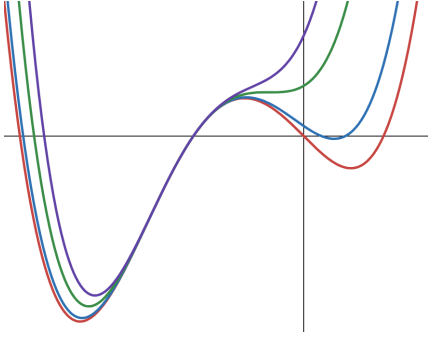


Method A: ℓ-2 Regularization

- One way to limit the model capacity is to constrain how large the parameters can get

$$\min_w \; f(w) + \lambda \|w\|_2^2, \quad f(w) = \mathbb{E}[\ell(w, z)] \tag{175}$$

- Leads to better landscape (getting rid of saddles, local max., sharp local min), good for both optimization and generalization. But we incur some bias



- Consider

$$\tilde{w}_j^* = \arg\min_w \; f(w) + \lambda_j \|w\|_2^2, \quad j = 1, 2 \tag{176}$$

- We can show

$$\|\tilde{w}_1^*\| \leq \|\tilde{w}_1^*\| \quad \Leftrightarrow \quad \lambda_1 \geq \lambda_2 \tag{177}$$

- Thus, as regularization becomes stronger optimal model becomes small and in the extreme case all weights will be zero (no learning capacity)
- Let w^* be the solution of unregularized problem. Assuming a L-smooth loss f, we can lower bound the bias

$$\text{Bias} = \frac{\|w^* - \tilde{w}^*\|}{\|w^*\|} \geq \frac{\lambda_L}{1 + \lambda_L}, \quad \lambda_L = \lambda / L \tag{178}$$

- As λ increases, the bias increases.

- Better landscape is good for optimization (easier functions). Finding higher quality solutions faster by (S)GD
- Consider GD for our regularized problem

$$\min_w \; f(w) + \lambda \|w\|_2^2, \quad f(w) = \mathbb{E}[\ell(w, z)] \tag{179}$$

We have

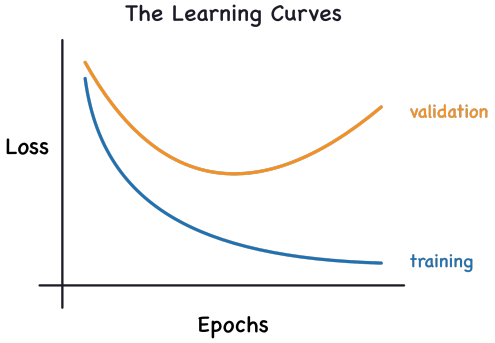
$$w_{w+1} = w_t - \eta(\nabla f(w_t) + \lambda w_t) = \underbrace{(1 - \eta\lambda)}_{\in(0,1)} w_t - \eta \nabla f(w_t) \tag{180}$$

called GD with Weight decay

- For simple methods like GD, ℓ-2 Regularization and Weight decay are equivalent. But not true in general, e.g. ADAM

Method B: Early Stopping

- When training, maintain a validation set and monitor the validation loss
- Continue training until validation loss starts to increase
- Then, stop earlier than what you originally had in mind
- By stopping early, we are restraining the model to reach its maximum learning capacity, aka regularization



Connection Between ℓ-2 Regularization and Early Stopping

- Consider a convex quadratic loss

$$f(w) = \frac{1}{2}(w - w^*)H(w - w^*), \quad w^* = \arg\min_w f(w) \tag{181}$$

and GD

$$w_{t+1} = w_t - \eta H(w_t - w^*), \quad w_1 = 0 \tag{182}$$

- Consider the Spectral decomposition $H = U\Lambda U^T$. Using simple steps we can show

$$\begin{aligned} U^\top w_t &= \left(I_d(I_d - \eta\Lambda)^\top\right) U^\top w_t \qquad \text{parameter at time } t \text{ of original prob.} \\ U^\top \tilde{w} &= \left(I_d(\lambda I_d + \Lambda)^{-1}\lambda\right) U^\top w_t \qquad \text{optimal solution of regularized problem} \end{aligned} \tag{183}$$

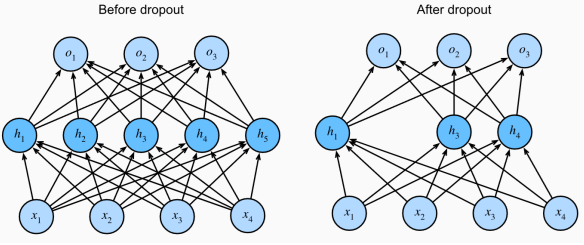
- $w_T \approx \tilde{w}$ if $\lambda T \approx \eta^{-1}$
- For a fixed learning rate, stopping early is like larger regularization.

Method C: Dropout

- In training, dropout some neurons at random by zeroing out p fraction of them

$$\tilde{h}_i = 0 \quad \text{with prob.} \quad p, \qquad \tilde{h}_i = \frac{h_i}{1-p} \quad \text{with prob.} \quad 1-p, \tag{184}$$

  such that $\mathbb{E}[\tilde{h}_i] = h_i$.
- By dropping neurons, we are restraining the model to reach its maximum learning capacity, aka regularization
- In testing, disabled (like batch normalization), unless for uncertainty quantification



Connection Between ℓ-2 Regularization and Dropout

- Dropout is essentially injecting noise into our model and hence the learning problem. The noise is independent of the activations and multiplicative in nature
- Let us use a simpler model to see the regularization effect of noise injection
- Consider a linear model

$$\min f(w) = \mathbb{E}_z[(y - \mathrm{w}^\top h)^2] \tag{185}$$

- Think of h as the activation we will inject noise into. Here, for simplicity, an additive noise

$$h \to h + e, \quad e \text{ indep. of } x, \quad \mathbb{E}[e] = 0, \quad \mathbb{E}[ee^\top] = \Sigma \tag{186}$$

- Note $\Sigma = A^\top A \succeq 0$ is the covariance matrix and PSD
- We can show using simple steps

$$\mathbb{E}_{z,e}[(y - \mathrm{w}^\top (h + e))^2] = \mathbb{E}_z[(y - \mathrm{w}^\top h)^2] + \|Aw\|_2^2 \tag{187}$$

Note: A similar idea used in Gaussian Smoothing and ZO-SGD.

Noise Injection and SAM

- We can also inject noise into w directly. Also, we can consider a small worst case noise, as opposed to a random one

$$\min_w \mathbb{E}_{e,z}[\ell(w+e,z)], \qquad \min_w \max_{e:\ \|e\|\leq\rho} \mathbb{E}_z[\ell(w+e,z)] \tag{188}$$

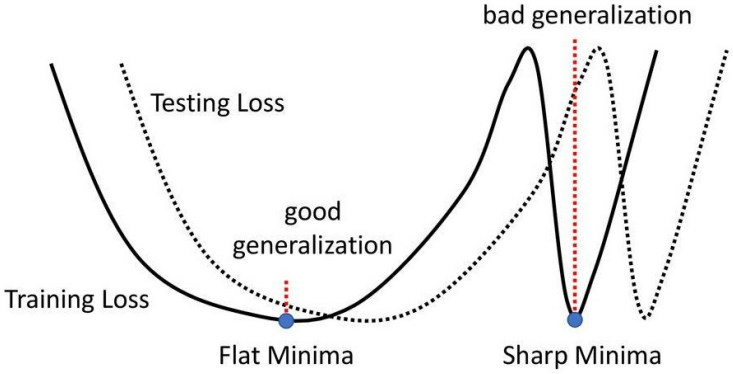
- The second formulation, with some approximation leads to

$$\min_w \mathbb{E}_z[\ell(w,z)] + \rho\|\nabla\mathbb{E}_z[\ell(w,z)]\| \tag{189}$$

- Called sharpness-aware minimization (SAM), useful for improved generalization through penalizing sharpness/sensitivity
- We can penalize the growth of higher-order derivatives, e.g., Hessian, too.

SAM and Improved Generalization

- Flatter Minima are likely to generalize better



Tuning Learning Rates

Motivation

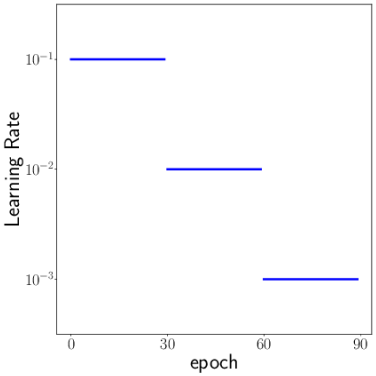
- Recall we showed minibatch SGD satisfies

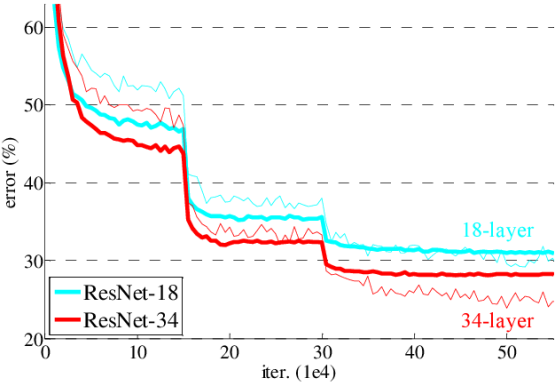
$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}\|\nabla f(w_t)\|^2 \lesssim \frac{\Delta}{T\eta} + \frac{L\eta\sigma^2}{2B} \tag{190}$$

  and we set the $\eta = \min\{\frac{1}{L}, \sqrt{\frac{2\Delta B}{L\sigma^2 T}}\}$ to get the best bound
- A lot of unknowns. Common practice: set η to be a small constant.
- As we train the model, the first term disappears, but the second term remains.
- Fix: adopt a decreasing schedule of learning rates
  - sublinear decay: $\eta \propto \frac{1}{\sqrt{t}}$
  - exponential decay: $\eta \leftarrow \eta/\alpha$, $\alpha > 1$, after every E epochs
- Interpretation: Early phase of training, we may prefer larger progresses (large η). But, as we approach the solution, we may need a more refined search (small η)
- To find the constants (e.g., the first learning rate), use Hyperparameter search

Exponential Decay

- Frequently used when training ResNets



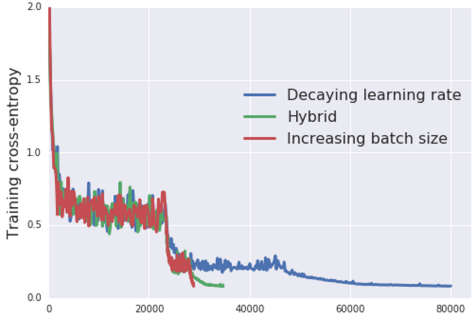


Role of Batch size

- Recall our bound again

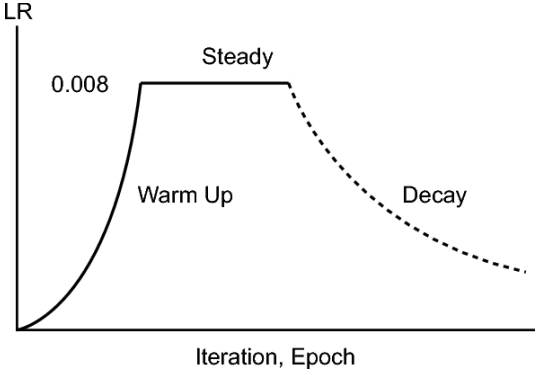
$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}\|\nabla f(w_t)\|^2 \lesssim \frac{\Delta}{T\eta} + \frac{L\eta\sigma^2}{2B} \tag{191}$$

- Conceptually, even if η is constant, we can shrink the second term by increasing the batch size!
- Interpretation: Early phase of training, we may tolerate more noise (smaller B) in our gradient estimation. But, as we approach the solution, we may need a more refined estimation (larger B)



Warm-up

- Gradually increase η to a target and then switch to a decaying schedule
- Can be interpreted as an attempt to bridge the gap between SGD and minibatch SGD



Warm-up (Cont'd)

- Recall, we argued when $\eta = \min\{\frac{1}{L}, \sqrt{\frac{2\Delta B}{L\sigma^2 T}}\}$, in terms of a fixed budget (oracle complexity)

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z_{\mathbf{1}:T}}\|\nabla f(w_t)\|^2 \lesssim \frac{B}{C} + \frac{1}{\sqrt{C}} \tag{192}$$

  $B = 1$ leads to a better worst-case bound, i.e., B steps of SGD better than one step of B-SGD
- Warm up could help bridge this gap in the larger gradient regime (early phase of training)
- Assume a small $\eta \propto \frac{B}{\sqrt{C}}$ (tuning constants properly) and recalling $C = B \times T$, from our bound

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z_{\mathbf{1}:T}}\|\nabla f(w_t)\|^2 \lesssim \frac{\Delta}{T\eta} + \frac{L\eta\sigma^2}{2B} \tag{193}$$

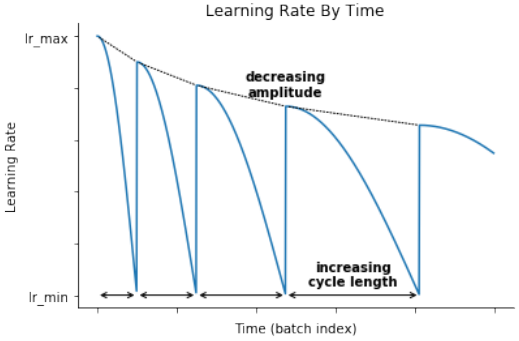
  we can see the resultant worst-case bound becomes $\mathcal{O}(\sqrt{\frac{1}{C}})$, independent of batch size
- In small gradient regime (intermediate and final learning phases), gradient could be small w.r.t. to the noise in gradient estimation, and likely SGD and B-SGD act similarly

Cyclic Learning Rates

- Repeated reset the model with the hope of a better starting point each time

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})[1 + \cos(\frac{t\pi}{T})] \tag{194}$$

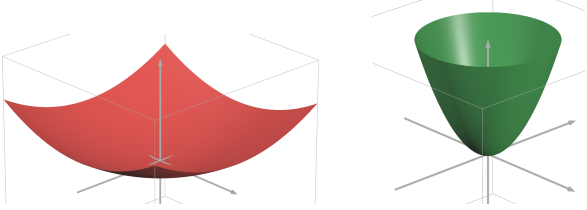
also called cosine-annealing or periodic schedule



Cyclic Schedule as Multi-Step Risk Minimizer

- At any point in the landscape w_t, there are two bad things could happen in the immediate next update w_{t+1}
  - We are at a flat landscape and are using a small η
  - We are at a steep landscape and are using a large η

  First one means slow progress and it's bad, but second one could lead to divergence and that is terrible



- With a constant η, we hedge against the second, by minimizing the one-step risk
- Cyclic Schedule could minimize Multi-Step Risks: what could happen n-th step from now

One-step Risk

- Consider GD and Convex Quadratic losses

$$f(w) = \frac{1}{2} w^\top Q w, \quad L I_d \succeq Q \succeq \mu I_d, \quad w^* = 0, \quad w^+ = (I_d - \eta Q) w \tag{195}$$

- Consider the one-step risk

$$\mathcal{R}(\eta) := \frac{\|w^+ - w^*\|}{\|w - w^*\|} = \frac{\|(I_d - \eta Q) w\|}{\|w\|} = \max_{i=1,\ldots,d} |\lambda_i (I_d - \eta Q)| = \max\{1 - \eta\mu, L\eta - 1\} \tag{196}$$

- And the best 1-step η can be found by solving the following linear program (solution where the two line intersect)

$$\eta_1^* = \arg \min_{0 \le \eta \le 1} \max\{1 - \eta\mu, L\eta - 1\} = \frac{2}{\mu + L} \tag{197}$$

  can be calculated directly by analyzing GD (typical in classic convex optimization)

- Now consider a case where we are interested in 2-step risk minimization:

$$w^+ = (I_d - \eta^+ Q)w, \quad w^\# = (I_d - \eta^\# Q)w^+,$$
$$\eta_1^*, \eta_2^* = \arg\min_{\eta^+, \eta^\#} \mathcal{R}(\eta^+, \eta^\#) := \frac{\|w^\# - w^*\|}{\|w - w^*\|} = \max_{i=1,\dots,d} |\lambda_i(I_d - (\eta^+ + \eta^\#)Q + \eta^+\eta^\# Q^2)| \tag{198}$$

A more involved problem but can be solved using Chebyshev polynomials

$$1/\eta_1^* = \frac{\mu + L}{2} + \frac{L - \mu}{2\sqrt{2}}, \quad 1/\eta_2^* = \frac{\mu + L}{2} - \frac{L - \mu}{2\sqrt{2}} \tag{199}$$

- That is, cycle between two learning rates
- Analyze for general training problems still open question...

Advanced Training Methods: Adaptivity

- Recall we showed minibatch SGD satisfies

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}\|\nabla f(w_t)\|^2 \lesssim \frac{\Delta}{T\eta} + \frac{L\eta\sigma^2}{2B} \tag{200}$$

  and we set the $\eta = \min\{\frac{1}{L}, \sqrt{\frac{2\Delta B}{L\sigma^2 T}}\}$ to get the best bound. But there are lots of unknowns...
- Here we focus on estimating variance on the fly

$$\mathbb{E}\|g_t - \nabla f(w_t)\|^2 \leq \sigma^2 \tag{201}$$

  from the stochastic gradients we have observed so far: $g_1, \ldots, g_t$. Note, these are NOT independent random quantities.

- To gain an intuition, let us assume they were i.i.d. with zero means

$$\sigma^2 \geq \mathrm{Var} = \mathbb{E}\|g\|^2 \approx \frac{1}{t}\sum_{\tau=1}^{t} \|g_\tau\|^2 \tag{202}$$

  such that $t\sigma^2 \approx \sum_{\tau=1}^{t} \|g_\tau\|^2$.

- Let us then use this expression in our learning rate formula and SGD:

$$w_{t+1} = w_t - \eta_t g_t, \qquad \eta_t = \frac{C}{\sqrt{\sum_{\tau=1}^{t} \|g_\tau\|^2 + \kappa^2}} \tag{203}$$

  for some constant C where we also added $\kappa > 0$ for numerical stability.

- This method is called adaptive SGD.
- Like Normalization methods, we are normalizing a key quantity

- Note that $\eta_t = \frac{C}{\sqrt{\sum_{\tau=1}^{t} \|g_\tau\|^2 + \kappa^2}}$ is not deterministic anymore as it depends on the random gradients.
- For implication, consider the part of proof from SGD:

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) + \langle w_{t+1} - w_t, \nabla f(w_t) \rangle + \frac{L}{2}\|w_{t+1} - w_t\|^2 \\ &= f(w_t) - \eta_t \langle g_t, \nabla f(w_t) \rangle + \frac{L\eta_t^2}{2}\|g_t\|^2 \end{aligned} \tag{204}$$

where conditioned on w_t (equivalently $z_1, \ldots, z_{t-1}$)

$$\mathbb{E}_{z_t}[\eta_t \langle g_t, \nabla f(w_t) \rangle | w_t] \neq \eta_t \mathbb{E}_{z_t}[\langle g_t, \nabla f(w_t) \rangle | w_t] \tag{205}$$

as η_t is depends on g_t

Handling Randomness of η_t

- Let $\eta_t = \frac{C}{\sqrt{\sum_{\tau=1}^{t} \|g_\tau\|^2 + \kappa^2}}$ for $t \geq 1$ and $\eta_0 = C/\kappa$
- Note that our learning rate schedule is causal,meaning it does not depend on future stochastic gradients.
- Furthermore, our schedule is non-increasing as we are accumulating non-negative terms $\|g_\tau\|^2$ in its denominator.
- We can leverage these properties

$$
\begin{aligned}
f(w_{t+1}) &\leq f(w_t) + \langle w_{t+1} - w_t, \nabla f(w_t) \rangle + \frac{L}{2}\|w_{t+1} - w_t\|^2 \\
&= f(w_t) - \eta_t \langle g_t, \nabla f(w_t) \rangle + \frac{L\eta_t^2}{2}\|g_t\|^2 \\
&= f(w_t) - \underbrace{\eta_{t-1}\langle g_t, \nabla f(w_t) \rangle}_{\text{Term 1}} + \frac{L\eta_t^2}{2}\|g_t\|^2 + \underbrace{(\eta_{t-1} - \eta_t)\langle g_t, \nabla f(w_t) \rangle}_{\text{Term 2}}
\end{aligned}
\tag{206}
$$

Handling Randomness of η_t (cont'd)

$$f(w_{t+1}) = f(w_t) - \underbrace{\eta_{t-1}\langle g_t, \nabla f(w_t)\rangle}_{\text{Term 1}} + \frac{L\eta_t^2}{2}\|g_t\|^2 + \underbrace{(\eta_{t-1}-\eta_t)\langle g_t, \nabla f(w_t)\rangle}_{\text{Term 2}} \tag{207}$$

- Term 1 has no issue with randomness anymore by causality
- Term 2 can be uniformly bounded by $(\eta_{t-1}-\eta_t)G^2$ assuming uniformly bounded (stochastic) gradients, e.g., by imposing G-Lipschitzness of sample loss $\ell(w,z)$
- Using our previous proof techniques, notably the telescoping term $\sum_{t=1}^{T}(\eta_{t-1}-\eta_t)G^2 = G^2(\eta_0-\eta_T) \le G^2\eta_0 = G^2C\kappa^{-1}$, we can establish the following result

Theorem

Assume $f(w)=\mathbb{E}[\ell(w,z)]$ is L-smooth and $\ell(w,z)$ is G-Lipschitz. Furthermore, assume we have access to an SFO and that $z_1,\dots,z_T$ are statistically independent. Then, adaptive SGD with learning rate $\eta_t = \frac{C}{\sqrt{\sum_{\tau=1}^{t}\|g_\tau\|^2+\kappa^2}}$ satisfies $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(\kappa^{-1}/\sqrt{T})$.

Integrating Exponential Moving Average (EMA)

- Our method is basically a simple version of famous methods such as Adagrad, Adadelta, RMSProp, and ADAM
- A key modification is using weighted averaging, putting emphasis on most recent observations

$$\eta_t = \frac{C}{\sqrt{\sum_{\tau=1}^{t} \|g_\tau\|^2 + \kappa^2}} \quad \rightarrow \quad \eta_t = \frac{C}{\sqrt{\sum_{\tau=1}^{t} \tilde{\beta}^{t-\tau}\|g_\tau\|^2 + \kappa^2}}, \quad 0 < \tilde{\beta} \leq 1$$

- In terms of moving averages, $\eta_t = \frac{C}{\sqrt{t \times S_t + \kappa^2}}$ where

$$S_t = (1-\frac{1}{t})S_{t-1} + \frac{1}{t}\|g_t\|^2 \quad \rightarrow \quad S_t = \beta S_{t-1} + (1-\beta)\|g_t\|^2 \text{ (exponential moving average (EMA))}$$

- For large t, simple average diminishes the contribution of new observation, but with a fixed β, the contribution of new observation is not diminished with time.
- $1 - \frac{1}{t} \rightarrow \beta$ Gives an idea that in practice β should be close to 1, e.g., $\beta = 0.99$

EMA and Training Instability

- One potential issue of EMA is losing the non-increasing property of η_t

$$\eta_t \leq \eta_{t-1} \equiv t \times S_t \geq (t-1) \times S_{t-1} \tag{208}$$

- The condition $t \times S_t \geq (t-1) \times S_{t-1}$ always happen for simple averaging, but if

$$\|g_t\|^2 \leq S_{t-1}\Big(1 - \frac{1}{t(1-\beta)}\Big) \tag{209}$$

  it won't happen for EMA. Likely to occur when t is large or β is small
- As a result, learning rates start to go up and lead to training instability and divergence issues (even for convex problems).
- Fix 1: Ensure $\beta \approx 1$ and do not overtrain
- Fix 2: Implement safeguarding: $S_t \leftarrow \max(S_t, S_{t-1}\frac{(t-1)}{t})$ to ensure the non-increasing property of η_t holds (An idea leading to AMSgrad)

Per-coordinate learning rates and curvature adaptivity

- In deep learning, different parameters may exhibit gradients with widely varying magnitudes.
- The multi-dimensional loss function in deep learning could mean some directions exhibit steep curvature while others are relatively flat.
- Per-coordinate learning rates: each parameter or group of parameters is updated with a learning rate adapted to its own gradient statistics.

$$w_{t+1}(i) = w_t(i) - \eta_t(i) g_t(i), \qquad \eta_t(i) = \frac{C}{\sqrt{\sum_{\tau=1}^{t} |g_\tau(i)|^2 + \kappa^2}} \tag{210}$$

- Similar motivation as normalization methods
- Can be analyzed by consider d one dimensional problems as

$$\langle g_t, w_t - w^* \rangle = \sum_{i=1}^{d} g_t(i)[w_t(i) - w^*(i)] \tag{211}$$

(And similar behavior for other important terms)

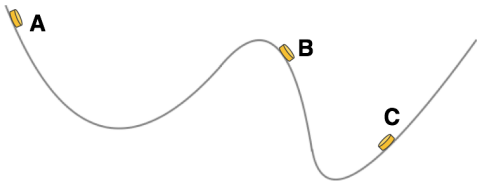
Advanced Training Methods: Momentum

A Physical Perspective

- Consider the movement of a ball going downhill
- The ball will eventually stop somewhere with locally low potential energy
- The trajectory is described by Newton's law of motion $F = m\ddot{w}$
- The ball experiences two forces: a force due to the potential energy $-\nabla f(w)$ and a drag force $-\rho\dot{w}$. Thus

$$m\ddot{w} = -\rho\dot{w} - \nabla f(w), \quad \equiv \quad \begin{cases} \dot{p} = -\alpha\frac{p}{m} - \nabla f(w) \\ \dot{w} = \frac{p}{m} \end{cases} \tag{212}$$

- Thus, conceptually, depending on the built up momentum p and the strength of the drag force, the ball could reside at a point with the globally lowest potential.



Deriving GD

- Let us consider the case where the ball is a massless particle. Assuming its speed is less than speed of light,

$$\dot{w} = -\rho^{-1}\nabla f(w) \qquad \text{called gradient flow} \tag{213}$$

- Using Euler's discretization

$$w_{t+1} = w_t - \rho^{-1}\nabla f(w_t) \tag{214}$$

  which is exactly our GD method
- Thus, GD shows the trajectory of massless or light ball
- Here, the momentum is zero and the ball may not be able to avoid getting stuck in a locally low potential state

- Let us now consider the heavy ball case. Using Euler's discretization again (and after basic manipulation and renaming the constants) we can show

$$w_{t+1} = w_t - \eta \nabla f(w_t) + \beta(w_t - w_{t-1}) \tag{215}$$

or equivalently

$$\begin{cases} w_{t+1} = w_t - \eta p_t \\ p_t = \beta p_{t-1} + \nabla f(w_t) \end{cases} \tag{216}$$

- This method is called the Heavy Ball Method with momentum parameter $0 \leq \beta < 1$ and serves as a key ingredient in DL optimization
- The correction term could aid reaching higher quality solutions compared to simple GD

Nesterov's Accelerated Gradient Method

- Assuming a time-varying drag force $-\frac{\rho}{t}\dot{w}$ the law becomes

$$\ddot{w} + \frac{\rho}{t}\dot{w} + \nabla f(w) = 0 \tag{217}$$

- It turns our this ODE is intimately related to another method called Nesterov's Accelerated Gradient (NAG)

$$\begin{cases} w_{t+1} = w_t - \eta p_t \\ p_t = \beta p_{t-1} + \nabla f(w_t - \eta\beta p_t) \end{cases} \tag{218}$$

- Called accelerated since it achieves an accelerated (and optimal) convergence rate compared to GD for smooth and strongly convex functions.
- NAG and other accelerated variants of GD can also be thought of as better discretization of our original law of motion, e.g., applying two-point Euler's method.

- In the context of DL optimization, we typically use m_t instead of p_t and also use the following variant

$$\begin{cases} w_{t+1} = w_t - \eta m_t \\ m_t = \beta m_{t-1} + (1-\beta) g_t, \quad m_0 = 0 \end{cases} \tag{219}$$

  which we call SGD with momentum (SGDm)

- Essentially, the momentum performs an exponential moving average on all stochastic gradients observed so far and uses that in the update

$$m_t = (1-\beta) \sum_{k=0}^{t-1} \beta^k g_{t-k} \tag{220}$$

Momentum as a new Gradient Estimator

- Recall under the SFO assumption, g_t is (conditionally) an unbiased estimator of $\nabla f(w_t)$ with variance σ^2
- In SGDm, we use $m_t = \beta m_{t-1} + (1-\beta)g_t$, $m_0 = 0$ instead of g_t to update our parameter
- Note that from an estimation perspective, this leads to bias as

$$\begin{aligned}\mathbb{E}_{z_t}[m_t|z_{1:t-1}] &= \mathbb{E}_{z_t}[\beta m_{t-1} + (1-\beta)g_t|z_{1:t-1}] \\ &= \beta\mathbb{E}_{z_t}[m_{t-1}|z_{1:t-1}] + (1-\beta)\mathbb{E}_{z_t}[g_t|z_{1:t-1}] \\ &= \beta\mathbb{E}_{z_t}[m_{t-1}|z_{1:t-1}] + (1-\beta)\nabla f(w_t) \neq \nabla f(w_t)\end{aligned} \tag{221}$$

- But, perhaps we are gaining some reduction in the variance σ^2 and as a result a reduction in total estimation error

- Recalling $m_0 = 0$ and expanding $m_t = \beta m_{t-1} + (1-\beta) g_t$ over time

$$m_t = (1-\beta) \sum_{k=0}^{t-1} \beta^k g_{t-k} \tag{222}$$

- To gain intuition, let us assume each g_{t-k} is an independent unbiased estimator $\nabla f(w_t)$ with variance $\mathbb{E}\|g_{t-k} - \nabla f(w_t)\|^2 \leq \sigma^2$. Our first take away is that

$$\mathbb{E}[m_t] = \nabla f(w_t) \cdot (1-\beta) \cdot \sum_{k=0}^{t-1} \beta^k = \nabla f(w_t) \cdot (1-\beta) \cdot \frac{(1-\beta^t)}{(1-\beta)} = \nabla f(w_t) \cdot (1-\beta^t) \tag{223}$$

so we have a bias that is vanishing as we train our model and $t \to \infty$. This observation provides a way to "de-bias" the momentum by dividing with $(1-\beta^t)$

$$m_t = \beta m_{t-1} + (1-\beta) g_t, \quad \hat{m}_t = m_t/(1-\beta^t), \quad w_{t+1} = w_t - \eta \hat{m}_t \tag{224}$$

- Let us now consider the variance of de-biased momentum $\hat{m}_t$ under our simple setting

$$\begin{aligned}
\mathbb{E}\|\hat{m}_t - \nabla f(w_t)\|^2 &= \left\| \frac{(1-\beta)}{(1-\beta^t)} \sum_{k=0}^{t-1} \beta^k g_{t-k} - \nabla f(w_t) \right\|^2 \\
&= \mathbb{E} \left\| \frac{(1-\beta)}{(1-\beta^t)} \sum_{k=0}^{t-1} \beta^k [g_{t-k} - \nabla f(w_t)] \right\|^2 \\
&= \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum_{k=0}^{t-1} \beta^{2k} \mathbb{E}\|g_{t-k} - \nabla f(w_t)\|^2 \\
&\leq \sigma^2 \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum_{k=0}^{t-1} \beta^{2k} = \sigma^2 \underbrace{\frac{1-\beta^{2t-1}}{(1-\beta^t)^2}}_{:=B(t)} \frac{(1-\beta)^2}{1-\beta^2} = \sigma^2 \frac{1-\beta}{1+\beta} B(t)
\end{aligned} \tag{225}$$

Implicit Mini Batching (Cont'd)

- $B(t) = \frac{1-\beta^{2t-1}}{(1-\beta^t)^2}$ is a decreasing function of $t \geq 1$ and approaches to 1 as $t \to \infty$
- Thus, we have shown as we train the variance of de-biased momentum decreases
- Let us now assume $B(t) \approx 1$ and we use a large momentum parameter like $\beta = 0.9$

$$\mathbb{E}\|\hat{m}_t - \nabla f(w_t)\|^2 \approx B(t)\frac{1-\beta}{1+\beta}\sigma^2 \approx \frac{1-\beta}{1+\beta}\sigma^2 \approx \frac{1-0.9}{1+0.9}\sigma^2 \approx \frac{\sigma^2}{20} \tag{226}$$

effectively a mini-batch SGD with batch size 20.

- Important Remark: Our intuitive analysis is not precise and even the de-biased momentum suffers from some bias. We will develop a formal study soon.

The ADAM optimizer

ADAM, leverages theses four component into an effective method: momentum, de-basing, adaptive η, per coordinate η (and optionally weight-decay):

- $z_t \sim p_z$ (Generating a mini batch)
- $g_t = \nabla \ell(w_t, z_t) + \lambda w_t$ (compute the stochastic gradient)
- $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ (compute the momentum)
- $\hat{m}_t = m_t / (1-\beta_1^t)$ (de-bias the momentum)
- $S_t(i) = \beta_2 S_{t-1}(i) + (1-\beta_2)|g_t(i)|^2$ (EMA for adaptive per coordinate variance estimation)
- $\hat{S}_t(i) = S_t(i)/(1-\beta_2^t)$ (de-bias the variance)
- $\eta_t(i) = \tilde{\eta}_t(i) / \sqrt{\kappa^2 + \hat{S}_t(i)}$ (adaptive per coordinate learning rate)
- $w_{t+1}(i) = [1 - \lambda \eta_t(i)] w_t(i) - \eta_t(i) \hat{m}_t(i)$ (parameter update)

Momentum as Bias-Variance Tradeoff

- Recall under the SFO assumption, g_t is (conditionally) an unbiased estimator of $\nabla f(w_t)$ with variance σ^2
- In SGDm, we use $m_t = \beta m_{t-1} + (1-\beta) g_t$, $m_0 = 0$ instead of g_t to update our parameter
- Note that from an estimation perspective, this leads to bias as

$$\begin{aligned}\mathbb{E}_{z_t}[m_t | z_{1:t-1}] &= \mathbb{E}_{z_t}[\beta m_{t-1} + (1-\beta) g_t | z_{1:t-1}] \\ &= \beta \mathbb{E}_{z_t}[m_{t-1} | z_{1:t-1}] + (1-\beta) \mathbb{E}_{z_t}[g_t | z_{1:t-1}] \\ &= \beta \mathbb{E}_{z_t}[m_{t-1} | z_{1:t-1}] + (1-\beta) \nabla f(w_t) \neq \nabla f(w_t)\end{aligned} \tag{227}$$

- What is the impact on convergence? Let us assuming L-smoothness

$$f(w_{t+1}) \leq f(w_t) - \eta \langle m_t, \nabla f(w_t) \rangle + \frac{L\eta^2}{2} \|m_t\|^2 \tag{228}$$

The usual analysis breaks since the $\mathbb{E}[\langle m_t, \nabla f(w_t) \rangle | z_{1:t-1}] \neq \|\nabla f(w_t)\|^2$.

Role of Estimation Error

- Let $e_t := m_t - \nabla f(w_t)$ be our total estimation error for SGDm
- Recall for SGD $e_t := g_t - \nabla f(w_t)$ such that $\mathbb{E}[\|e_t\|^2 | z_{1:t-1}] \leq \sigma^2$

Lemma 1

If $\eta \leq 1/4L$, for SGDm we have

$$\|\nabla f(w_t)\|^2 \leq \frac{4}{\eta}\Big(f(w_t) - \mathbb{E}_{z_t}[f(w_{t+1})|z_{1:t-1}]\Big) + 3\mathbb{E}[\|e_t\|^2|z_{1:t-1}] \tag{229}$$

- Recall, we had a result for SGD the second term was effectively $\eta\sigma^2$
- Thus, if we hope for convergence, $\mathbb{E}[\|e_t\|^2|z_{1:t-1}]$ must be small, e.g. $\mathcal{O}(\eta)$. That is, despite the small bias, variance reduces by a lot, hence estimation error and in turn convergence error reduces.

- Let us rewrite Let $e_{t+1} := m_{t+1} - \nabla f(w_{t+1})$

$$\begin{aligned} e_{t+1} &= \beta m_t + (1-\beta) g_{t+1} - \nabla f(w_{t+1}) \pm \nabla f(w_t) \\ &= \beta e_t + (1-\beta)(g_{t+1} - \nabla f(w_{t+1})) + \beta(\nabla f(w_t) - \nabla f(w_{t+1})) \end{aligned} \tag{230}$$

- When $\beta < 1$ the first term vanishes exponentially. The second term is also small and assuming a slow evolution of model parameters $\nabla f(w_t) \approx \nabla f(w_{t+1})$ and the third term is also small.

Lemma 2

Under the SFO assumption, SGDm's error satisfies the following recursion

$$\mathbb{E}\|e_{t+1}\|^2 \leq \beta \mathbb{E}\|e_t\|^2 + A(1-\beta)\mathbb{E}\|\nabla f(w_t)\|^2 + \tilde{A}(1-\beta)^2\sigma^2 \tag{231}$$

for some positive constants $A, \tilde{A} > 0$ when $\eta = C(1-\beta)/L$.

Theorem

Assume $f(w) = \mathbb{E}[\ell(w, z)]$ is L-smooth. Furthermore, assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, SGDm with learning rate $\eta = C(1-\beta)/L$ and momentum parameter $1 - \beta = \mathcal{O}(1/\sqrt{T})$ satisfies $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(1/\sqrt{T})$.

- The theorem highlights that the momentum parameter should be very large, e.g., $\beta = 0.99$, consistent with practice.
- Similar results hold under normalization

$$m_t = \beta m_{t-1} + (1-\beta) g_t, \qquad w_{t+1} = w_t - \eta \frac{m_t}{\|m_t\| + \kappa} \tag{232}$$

- Thus, despite a biased estimation, convergence occurs as the estimation error vanishes, thanks to variance reduction, as the algorithm converges (a virtuous cycle). Other/better ways to reduce variance?

Proof Sketch

- The proof relies on the idea of defining a potential function and tracking its evolution
- We actually had potential functions before which we worked with implicitly:

$$\Phi_t = \|w_t - w^*\|^2 \quad \text{for cvx and Lip. function,} \qquad \Phi_t = \mathbb{E}[f(w_t)] - f^* \quad \text{for smooth function} \tag{233}$$

  and we studied $\Phi_{t+1} - \Phi_t = \mathbb{E}[f(w_{t+1}) - f(w_t)]$ using L-smoothness
- Generalizable Template: Augment with above sources of errors, in this case, the estimation error

$$\Phi_t = \mathbb{E}[f(w_t)] - f^* + C_e \mathbb{E}\|e_t\|^2 \tag{234}$$

- Study the evolution of potential by summing over its consecutive differences:

$$\sum_{t=1}^{T} \Phi_{t+1} - \Phi_t = \Phi_{T+1} - \Phi_1 \geq f^* - f(w_1) - C_e \mathbb{E}\|e_1\|^2 \tag{235}$$

- By the definition of potential

$$\sum_{t=1}^{T} \Phi_{t+1} - \Phi_t = \left(\sum_{t=1}^{T} \mathbb{E}[f(w_{t+1}) - f(w_t)] \right) + \left(C_e \sum_{t=1}^{T} \mathbb{E}\|e_t\|^2 \right) \tag{236}$$

- First term dealt with by Lemma 1 (Smoothness) and Second term dealt with by Lemma 2 (evolution of estimation error)
- The nice property of error

$$\mathbb{E}\|e_{t+1}\|^2 \leq \beta \mathbb{E}\|e_t\|^2 + \text{ other terms}, \qquad 0 < \beta < 1 \tag{237}$$

ensures we can find a good constant C_e to ensure convergence.

- This idea can be utilized in a variety of learning settings (we will discuss a few more cases).

Advance Training Algorithms: Variance Reduction

Variance Reduction

- Recall m_t is a new estimator of $\nabla f(w_t)$ with some bias and lower variance than g_t.
- How can we get better variance reduction? One idea is Control variates from Statistics
- Let G be an unbiased estimator of F. Consider X and Y s.t. $\mathbb{E}[Y] = \mathbb{E}[X]$.
- Consider the new estimator

$$Z = G + \beta(X - Y) \tag{238}$$

- Let us calculate the mean and variance of Z

$$\begin{aligned} \mathbb{E}[Z] &= \mathbb{E}[G + \beta(X - Y)] = \mathbb{E}[G] + \beta\mathbb{E}[x] - \beta\mathbb{E}[Y] = F \\ \mathbb{E}\|Z - F\|^2 &= \mathbb{E}\|G - F\|^2 + \beta^2\mathbb{E}\|X - Y\|^2 + 2\beta\mathbb{E}[\langle G - F, X - Y\rangle] \end{aligned} \tag{239}$$

- That is, if $X - Y$ is negatively correlated with G such that

$$2\mathbb{E}[\langle G - F, X - Y\rangle] \leq \beta\mathbb{E}\|X - Y\|^2, \tag{240}$$

we may get a reduction in variance and retain unbiasedness.

Control variates in Optimization

- Consider the following update vector and iterates $w_{t+1} = w_t - \eta u_t$

$$Z = G + \beta(X - Y), \qquad u_t = \nabla \ell(w_t, z_t) + 1 \cdot (\nabla f(v) - \nabla \ell(v, z_t)) \tag{241}$$

  for some reference point v
- Since $\mathbb{E}[\nabla \ell(v, z_t)] = \nabla f(v)$, and $\mathbb{E}[\ell(w_t, z_t)] = \nabla f(w_t)$, u_t is unbiased.
- Let us assume both f and ℓ are L-smooth. Using basic techniques we can show

$$\mathbb{E}\|u_t - \nabla f(w_t)\|^2 \lesssim L\|w_t - v\|^2 \tag{242}$$

- Obviously, $v = w_t$ leads to the least variance, but too costly.
- Main idea: Only update the reference once in a while to trade some variance for computational efficiency

SVRG Algorithm

Consider the ERM objective $f(w) = \frac{1}{N}\sum_{i=1}^{N} \ell(w, z_i)$

- Initialize w_1 and $v_1 = w_1$, set $e = 1$ (the epoch counter)
- Compute $\nabla f(v_1)$ $\mathcal{O}(N)$ gradient computation
- for $t = 1, \dots T$
  - sample a batch $z_t \sim p_z$
  - form the update vector $u_t = \nabla \ell(w_t, z_t) + \nabla f(v_e) - \nabla \ell(v_e, z_t)$ 2 gradient computation
  - parameter update $w_{t+1} = w_t - \eta u_t$
  - if $t \equiv 0$ mod N (finishing one epoch, i.e., going over all training data)
    - $e \leftarrow e + 1$ increase epoch counter
    - $v_e = w_t$ update the reference
    - update the reference gradient $\nabla f(v_e)$ $\mathcal{O}(N)$ gradient computation every N iterations

Thus, if $T = EN$, the total gradient computation is $3EN = 3T$ (i.e., 3 times SGD's)

SVRG vs. SGD vs GD

- Recall, the oracle complexity of GD to find a stationary solution was $\mathcal{O}(N/\epsilon)$
- Recall, the oracle complexity of SGD to find a stationary solution was $\mathcal{O}(1/\epsilon^2)$
- It turns out the oracle complexity of SVRG to find a stationary solution is $\mathcal{O}(N + \frac{N^{2/3}}{\epsilon})$
- SVRG is better than GD if

$$N + \frac{N^{2/3}}{\epsilon} \leq N/\epsilon \quad \equiv \quad \epsilon^{-2} > \frac{N^{1/3}}{N^{1/3} - 1} \approx 1 \quad \text{always the case} \tag{243}$$

- SGD is better than SGD if

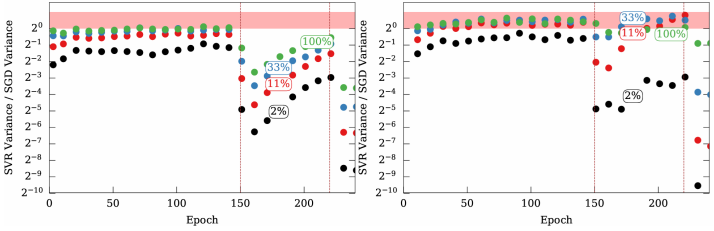
$$1/\epsilon^2 \leq N/\epsilon \quad \equiv \quad \epsilon^{-2} < N \quad \text{almost always the case in DL} \tag{244}$$

- SVRG is better than SGD if

$$N + \frac{N^{2/3}}{\epsilon} \leq 1/\epsilon^2 \quad \equiv \quad \epsilon^{-1.5} > N \quad \text{almost never the case in DL!} \tag{245}$$

Other shortcomings of SVRG-based methods

- For simple ML problems, not an issue, but for DL when using batch normalization, dropout, and data augmentation, this leads to ineffectiveness of variance reduction.
- The issue can be attributed to the finite-sum assumption $f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z)$.
- SVRG assumes whenever a z is sampled, the loss $\ell(w, z)$ is returned. Not true in DL!
- let ρ denote the randomness due to normalization, dropout, and data augmentation. In fact, $\ell(w, z) = \mathbb{E}_\rho[\tilde{\ell}(w, z_i, \rho)]$. Consequently, the sum is infinite.



(c) Small ResNet (d) ResNet-110

Figure 2: The SVRG to SGD gradient variance ratio during a run of SVRG. The shaded region indicates a variance *increase*, where the SVRG variance is worse than the SGD baseline. Dotted lines indicate when the step size was reduced.

Role of Data Augmentation

- In practice, to improve generalization and robustness, we not only use a sample z (say an image), but also its transformations (e.g., cropping, rotation, flipping), in training.
- In many DL packages, a transformation is chosen randomly any time we process a given z.
- Consider image $\tilde{z}$ and assume we will use it in iteration t
- Recall $u_t = \nabla \ell(w_t, \tilde{z}) + \nabla f(v_e) - \nabla \ell(v_e, \tilde{z})$. Also recall $\nabla f(v_e)$ is calculated outside an epoch while $\nabla \ell(v_e, \tilde{z})$ is calculated inside the epoch.
- $\tilde{z}$ appears with transformation T_1 in orange terms and T_2 in the blue term, Implying we may incur an estimation error due to bias and increased variance.

Role of Data Augmentation (Cont'd)

- A solution: Transformation Locking, meaning store the transformation used outside epoch and reused it inside epoch
- Nonetheless, as model updates, variance goes up, meaning, v_e not a good reference anymore.

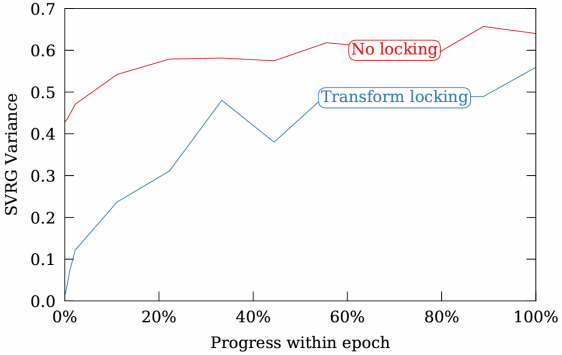


Figure 1: Variance within epoch two during LeNet training on CIFAR10.

Role of Dropout and Batch Normalization

- Dropout issue is similar to data augmentation as outride and inside epoch, different sparsity patterns are used. So, we can implement a similar locking solution.
- Issue with BN is complicated and two folds
- First, $\nabla \ell(w_t, \tilde{z})$ will now depends on other samples too, not just $\tilde{z}$

$$\nabla \ell(w_t, \tilde{z}; B_1) \neq \nabla \ell(w_t, \tilde{z}; B_2), \qquad B_1 \neq B_2 \tag{246}$$

  Implication: increased bias and variance in SVRG estimator due to the epoch-based structure
- Secondly, recall during inference, we use the statistics of the entire data to calculate the mean and variance of each layer's activation from mini-batch statistics found during training:

$$\mu_{\mathcal{D}} \leftarrow \beta \mu_{B_t} + (1-\beta)\mu_{\mathcal{D}}, \qquad \sigma^2_{\mathcal{D}} \leftarrow \beta \sigma^2_{B_t} + (1-\beta)\sigma^2_{\mathcal{D}} \tag{247}$$

- Default implementation won't compatible with SVRG as we compute the $(\mu_{B_t}, \sigma^2_{B_t})$ of a specific batch at both w_t and v_e.
- Solution: make sure statistics at v_e not used (batch reset)

Fixing SVRG

- In summary, we have issues due to the epoch-based structure and the finite-sum assumption and their incompatibility with DL operations.
- Ideally, we want to update the reference on the fly in each iteration. But doing so may be costly (e.g., requiring full gradient)
- Solution: Approximate control variates! Recall we said consider X and Y s.t. $\mathbb{E}[Y] = \mathbb{E}[X]$ and the estimator

$$Z = G + \beta(X - Y) \tag{248}$$

- Let us relax the condition such that now $\mathbb{E}[Y] \approx \mathbb{E}[X]$. Implication: some bias which we may control by finding a good β. Seems like Bias-variance tradeoff then! Connection to momentum?

SGDm as Approximate control variates

- Let us recall our SGDm update vector: $m_t = \beta m_{t-1} + (1-\beta)\nabla \ell(w_t, z_t)$, $m_0 = 0$
- Let us do a rearrangement

$$m_t = \nabla \ell(w_t, z_t) + \beta(m_{t-1} - \nabla \ell(w_t, z_t)), \qquad Z = G + \beta(X - Y) \tag{249}$$

- Intuitively, m_{t-1} is a good approximation of $\nabla f(w_{t-1})$ while $\nabla \ell(w_t, z_t)$ is a good approximation of $\nabla f(w_t)$
- under slow evolution assumption $w_{t-1} \approx w_t$ so

$$m_{t-1} \approx \nabla f(w_{t-1}) \approx \nabla f(w_t) \approx \nabla \ell(w_t, z_t) \tag{250}$$

  Hence SGDm as like Approximate control variates
- This discussion helps us to develop a better method by using something better to reduce the approximation bias

De-Biasing SGDm

- Previously, we talked about a de-biasing step: $m_t = \beta m_{t-1} + (1-\beta)\nabla \ell(w_t, z_t)$, $\hat{m}_t = m_t/(1-\beta^t)$
- Consider our previous discussion: m_{t-1} is a good approximation of $\nabla f(w_{t-1})$ while $\nabla \ell(w_t, z_t)$ is a good approximation of $\nabla f(w_t)$
- What if we change $\nabla \ell(w_t, z_t)$ to something that is a good approximation of $\nabla f(w_{t-1})$ instead?
- A candidate: $\nabla \ell(w_{t-1}, z_t)$ leading to the method

$$w_{t+1} = w_t - \eta m_t, \qquad m_t = \nabla \ell(w_t, z_t) + \beta(m_{t-1} - \nabla \ell(w_{t-1}, z_t)) \tag{251}$$

De-Biasing SGDm vs SVRG

- Our new estimator $m_t = \nabla \ell(w_t, z_t) + \beta(m_{t-1} - \nabla \ell(w_{t-1}, z_t))$
- Compared with SVRG's $u_t = \nabla \ell(w_t, z_t) + 1 \cdot (\nabla f(v) - \nabla \ell(v, z_t))$
  - Our reference point is now w_{t-1} which is much closer to w_t than v
  - We approximately calculate the full gradient at reference by using m_{t-1},
  - all calculations inside epoch as opposed to SVRG
  - using two gradient calculations in each iteration, like SVRG
  - Using $\beta < 1$ to tame the approximation bias as opposed to $\beta = 1$ in SVRG

Convergence Analysis

Theorem

Assume $f(w) = \mathbb{E}[\ell(w, z)]$ and $\ell(w, z)$ are both L-smooth. Furthermore, assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, the de-biased SGDm with learning rate $\eta = \mathcal{O}(\frac{1}{T^{1/3}})$ and momentum parameter $1 - \beta = \mathcal{O}(\frac{1}{T^{2/3}})$ satisfies $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(\frac{1}{T^{2/3}})$.

- Recall for SGD: $\eta = \mathcal{O}(\frac{1}{T^{1/2}})$, $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(\frac{1}{T^{1/2}})$
- Recall for SGDm: $\eta = \mathcal{O}(\frac{1}{T^{1/2}})$, $1 - \beta = \mathcal{O}(\frac{1}{T^{1/2}})$, $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(\frac{1}{T^{1/2}})$
- Thus, we get a provably better rate compared to both SGD and SGDm
- We are then using larger learning rates and momentum parameters, thanks to the reduced approximation bias.
- Recently shown optimal under the stated assumptions

Proof Sketch

- Like SGDm proof, we consider and analyze the evolution of the approximation error $e_t = m_t - \nabla f(w_t)$ and obtain

$$\mathbb{E}\|e_t\|^2 \leq \beta \mathbb{E}\|e_{t-1}\|^2 + \text{ other terms} \tag{252}$$

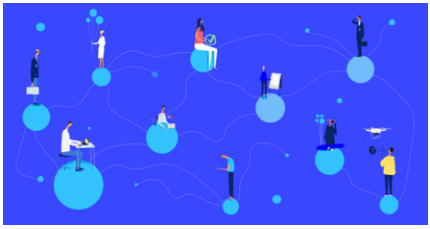
- We then define a potential function (again like SGDm proof)

$$\Phi_t = \mathbb{E}[f(w_t)] - f^* + C_e \mathbb{E}\|e_t\|^2 \tag{253}$$

- Study the evolution of potential by summing over its consecutive differences
- When $\beta < 1$ we can find a good constant C_e to ensure convergence.

Distributed Deep Learning

Learning from Distributed Data



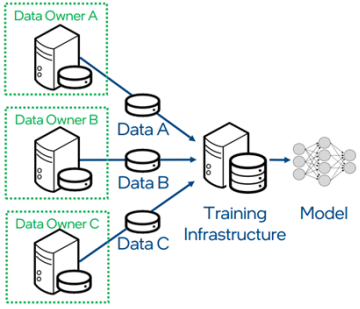
- So far, we discussed training tasks such as minimizing $f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i)$ (or abstractly, $f(w) = \mathbb{E}_{z\sim D}[\ell(w, z)]$) assuming data is all in one place
- In many settings, data is generated and is held locally by many devices or agents. Essentially,

$$\min_w f(w) = \frac{1}{K}\sum_{j=1}^{K} f_j(w), \qquad f_j(w) = \mathbb{E}_{z\sim D_j}[\ell(w, z)] \tag{254}$$

- We call f and f_j the global and local loss respectively.

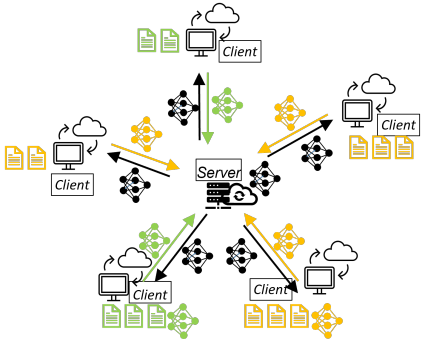
Centralized Learning

- All agents send their data to a central entity called Server
- Remark: We
  use Server, aggregator, and fusion center interchangeably
- Remark: We use agent, client,
  worker, participant, device, and node interchangeably
- Features
  - Data integrity no preserved
  - Communication
    issues if data size larger than agents' capability
  - Potential bottleneck at server if many agents
  - Fast and accurate training



Parallel Learning

- Instead of sharing
  raw data, agents share model-based representation of local data through (repeatedly) communicating $\nabla f_j(w_t)$ or similar quantities with server
- For $t = 1, \ldots, T$ Communication/training rounds
  - server sends the current global model $\bar{w}_t$ to the agents
  - each agent computes and
    sends $\nabla f_j(\bar{w}_t)$ or $w_{t+1}^j = \bar{w}_t - \eta \nabla f_j(\bar{w}_t)$ to the server
  - Server aggregates
    the recieved messages and update the global model



$$\bar{w}_{t+1} = \bar{w}_t - \eta \frac{1}{K} \sum_{j=1}^{K} \nabla f_j(\bar{w}_t), \qquad \bar{w}_{t+1} = \frac{1}{K} \sum_{j=1}^{K} w_{t+1}^j \tag{255}$$

Parallel Learning (Cont'd)

- Features
  - Improved level of preserving data integrity
  - Taking advantage of local computation
  - communication issues if model size larger than agents' capability
  - Potential bottleneck at server if many agents
  - Delay due to stragglers in synchronous systems
- We think of the averaging operator

$$\bar{w}_{t+1} = \bar{w}_t - \eta \frac{1}{K} \sum_{j=1}^{K} \nabla f_j(w_t), \qquad \bar{w}_{t+1} = \frac{1}{K} \sum_{j=1}^{K} w_{t+1}^j \tag{256}$$

  as an aggregation mechanism. Essentially maximizing Utilitarian Welfare.
- Can define other mechanisms based on other notions, e.g., egalitarian welfare (minimizing the worst loss).

- Can be thought of as approximate Parallel Learning to reduce communication and computation by trading off accuracy and precision
- Periodic Communication: Instead of communicating local gradients or local model updates after 1 local step, do so after $E > 1$ local steps

$$w_{t,\tau+1}^j = w_{t,\tau}^j - \eta \nabla f_j(w_{t,\tau}^j), \quad \tau = 1, \ldots, E, \quad w_{t,1}^j = \bar{w}_t, \quad w_{t,E+1}^j = w_{t+1}^j \tag{257}$$

- Can be thought of as approximate solving minimizing the local loss when initializing at $\bar{w}_t$:

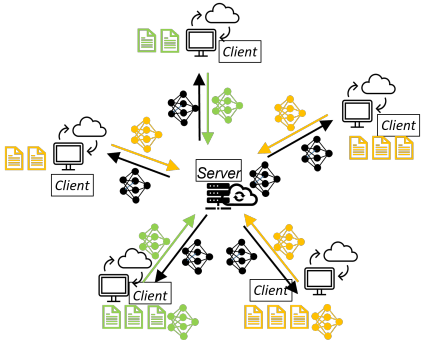
$$w_{t,E+1}^j = w_{t+1}^j \approx \operatorname{argmin} f_j(w) = \mathbb{E}_{z \sim D_j}[\ell(w, z)] \tag{258}$$

- Partial Participation: Each communication round, only a subset S_t of $r < N$ agents participate

$$\bar{w}_{t+1} = \frac{1}{r} \sum_{j \in S_t} w_{t+1}^j \tag{259}$$

Hardware Heterogeneity

- Agents may have
  different communication and computation capabilities
- Computation: agent i could
  be able to do more local steps than agent j, $E_i > E_j$
- Communication 1: agent i could
  be able to participate more frequently than agent j
- Communication 2: agent i could be able to send
  higher precision messages (more bits) than agent j
- Could lead to fairness issues
  as the typical method could favor more capable clients



- Each agent holds different loss as a result of different data, $D_i \neq D_j$
- Integration of partial participation and periodic communication leads to convergence issues.
- Let us rewrite our simple aggregation

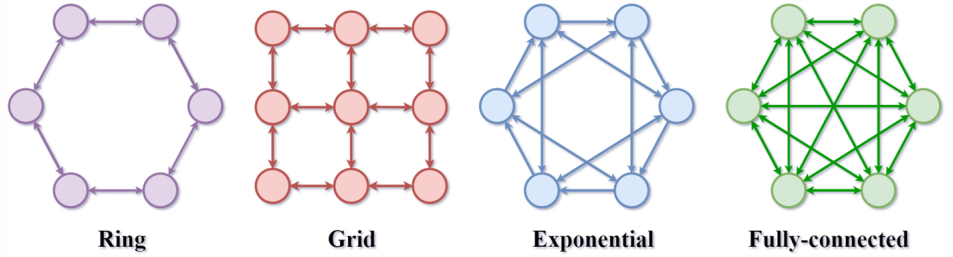
$$\begin{aligned} \bar{w}_{t+1} &= \frac{1}{r}\sum_{j\in S_t} w_{t+1}^j = \frac{1}{r}\sum_{j\in S_t} \bar{w}_t - \eta \sum_{\tau=1}^{E} \nabla f_j(w_{t,\tau}^j) \\ &= \bar{w}_t - \eta \frac{1}{r}\sum_{j\in S_t}\sum_{\tau=1}^{E} \nabla f_j(w_{t,\tau}^j) := \bar{w}_t - \eta \frac{1}{r}\sum_{j\in S_t} g_{t,E}^j \end{aligned} \tag{260}$$

  Resembling mini-batch SGD.

- One way to deal with these issues then is to use aggregations mimicking momentum to reduce the variance stemming from statistical heterogeneity.

Decentralized Learning

- No server, only peer-to-peer (p2p) communications, effectively each agent serving as its own local aggregator
- Mathematically, communication governed by a graph
- Lower graph degree implies lower communication bottleneck and lower delay
- Lower connectivity implies slower propagation of information and hence a slower convergence
- Synchronous communication could still cause delay issues



Decentralized Learning (Cont'd)

- Let N_j be the set of neighbors of agent i and note $i \in N_i$
- Each agent computes and sends the aggregation of the messages recieved from its neighbors:

$$w_{t+1}^i = \Big(\frac{1}{|N_i|}\sum_{j\in N_i} w_t^j\Big) - \eta\nabla f_i(w_t^i), \qquad w_{t+1}^i = \frac{1}{|N_i|}\sum_{j\in N_i}\Big(w_t^j - \eta\nabla f_i(w_t^j)\Big) \tag{261}$$

- Remark: A fully connected graph with no server node is equivalent to a star graph with a server node. Thus, parallel learning is an example of Decentralized Learning

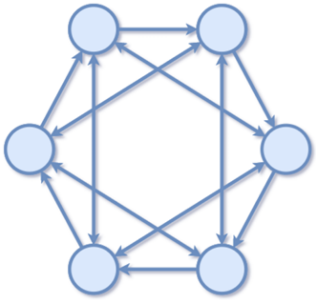
- Remark: Federated Learning is also an example of Decentralized Learning with a sequence of (disconnected) time-varying graphs

Decentralized SGD

- Assume we have K agents each with a different loss function
- Agents iterative communicate over a graph to find

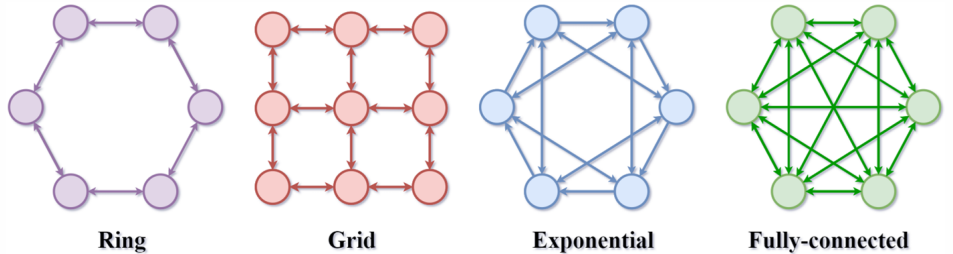
$$w^* = \arg\min_w f(w) = \frac{1}{K}\sum_{j=1}^{K} f_j(w), \qquad f_j(w) = \mathbb{E}_{z\sim D_j}[\ell(w,z)] \tag{262}$$

- At convergence, each agent's solution should be w^*, i.e., they should reach a consensus.



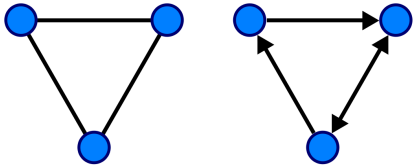
Communication Graph

- We can only hope to find a solution if the agents can communicate
- We assume a connected graph, that means there is at least one path between any two agents
- Intuitively, higher connectivity means faster convergence due to messages reaching the destination faster



Communication graph (Cont'd)

- Formally, $\mathcal{G}(\mathcal{N}, \mathcal{E})$ where $\mathcal{N} = \{1, \ldots, K\}$ is the set of agents and $\mathcal{E}$ is the set of edges
- $\{i, j\} \in \mathcal{E}$ if there is an edge from agent i to agent j, meaning, i can send a message to j
- Note, the communication could in general be asymmetric meaning $\{i, j\} \in \mathcal{E}$ but $\{i, j\} \notin \mathcal{E}$. Meaning, the graph could be a directed graph
- We will however, assume the graph is undirected for simplicity.
- Let $\mathcal{N}_i$ be the set of neighbors of agent i and note $i \in \mathcal{N}_i$
- More generally, we assume the communication is weighted and there is a weight matrix W that governs the communication among agents such that $W \in \mathbb{R}^{K \times K}$, $0 \leq W_{ij}$



Communication graph (Cont'd)

- Since each node can send a message to itself $W_{ii} > 0$
- Since the graph is undirected $W_{ij} = W_{ji}$, meaning W is a symmetric matrix $W = W^\top$
- Since $\{i, j\} \in \mathcal{E}$ if there is an edge between agent i and agent j

$$\{i, j\}, \{j, i\} \in \mathcal{E} \qquad \equiv \qquad W_{ij} = W_{ji} \neq 0 \tag{263}$$

- For each agent j, let us also normalize the weights for incoming messages sum to 1

$$\sum_{i=1}^{K} W_{ij} = 1 \quad \equiv \quad 1^\top W = 1^\top, \qquad \text{called Column stochastic} \tag{264}$$

- And since graph is undirected and W is symmetric

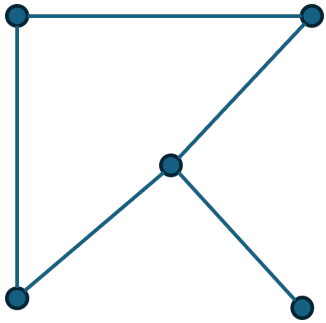
$$\sum_{j=1}^{K} W_{ij} = 1 \quad \equiv \quad W1 = 1, \qquad \text{called Row stochastic} \tag{265}$$

Doubly Stochastic Weight Matrix

- A matrix that
  is both row and column stochastic is called doubly stochastic
- Example: given any connected graph set

$$W_{ij} = \frac{1}{\max\{\text{degree}(i), \text{degree}(j)\}}, \quad i \neq j$$

  and $W_{ii} = 1 - \sum_{j \in \mathcal{N}_i} W_{ij}$
- Remark: since an agent can communicate
  with itself and the graph is connected, $\text{degree}(i) \geq 2$



Average Consensus

- Recall our goal is

$$w^* = \arg\min_w f(w) = \frac{1}{K}\sum_{j=1}^{K} f_j(w), \qquad f_j(w) = \mathbb{E}_{z\sim D_j}[\ell(w,z)] \tag{266}$$

- At convergence, each agent's solution should be w^*, i.e., they should reach a consensus.
- Let us first consider the case where $f_j(w) = 0.5\|w - x_j\|^2$, $x_j \in \mathbb{R}^d$. It tuns out f_j's and hence f are strongly convex.
- We can then easily calculate

$$w^* = \frac{1}{K}\sum_{j=1}^{K} x_j = \bar{x}, \qquad \text{Finding the average} \tag{267}$$

- This problem is called average consensus. Remember, we want to solve this over a graph
- A fully connected graph means by one exchange of local vectors x_j, each agent can compute the average (convergence in one iteration). How about general graphs?

Average Consensus Method

- Let us define a matrix $\bar{X} = \bar{x}1^\top = [\bar{x}, \ldots, \bar{x}] \in \mathbb{R}^{d \times K}$
- Then, it is not hard to see that

$$\bar{X} \cdot \frac{1}{K} 11^\top = \bar{x}1^\top \cdot \frac{1}{K} 11^\top = \bar{x}1^\top = \bar{X} \tag{268}$$

- But, for a fully connected graph, $W = \frac{1}{K} 11^\top$ (all entries are $1/K$ using a uniform weighting). That is,

$$\bar{X} = \bar{X}W, \qquad \text{a fixed point condition} \tag{269}$$

- Turns out this is true for all connected graphs with any doubly stochastic W.
- How do we find fixed points? Fixed point iteration (recall the connection to GD)
- Our method is then: Initialize local solutions x_1^i and run the following local step for t iterations

$$x_{t+1}^i = \sum_{j \in \mathcal{N}_i} W_{ij} x_t^i, \qquad \equiv \qquad X^{t+1} = X^t W \quad \text{in matrix form} \tag{270}$$

where $X^t = [x_t^1, \ldots, x_t^K] \in \mathbb{R}^{d \times K}$.

Theorem

Assume the graph is connected and W is doubly stochastic. Define the consensus error as

$$e_t = \frac{1}{K}\|X^t - \bar{X}\|_F^2 = \frac{1}{K}\sum_{j=1}^{K}\|x_t^j - \bar{x}\|^2 \tag{271}$$

Then, using the Banach Fixed point theorem, it holds that $e_{t+1} \leq |\lambda_2(W)|^2 e_t$, where $|\lambda_2(W)|$ is the second largest eigenvalue of W (in magnitude)

- Assuming W is doubly stochastic and recalling $W1 = 1$ means $\lambda = 1$ is an eigenvalue and in fact the maximum eigen value of W.
- The quantity $0 \leq \rho := 1 - |\lambda_2(W)| \leq 1$ is called the spectral gap of W
- Large ρ means better connectivity and faster convergence ($\rho = 1$ for a fully connected graph)
- small ρ means worse connectivity and slower convergence ($\rho = 0$ for a disconnected graph)

- Recall our problem is

$$w^* = \arg\min_w f(w) = \frac{1}{K}\sum_{j=1}^{K} f_j(w), \qquad f_j(w) = \mathbb{E}_{z\sim D_j}[\ell(w,z)] \tag{272}$$

and for consensus

$$x_{t+1}^i = \sum_{j\in\mathcal{N}_i} W_{ij} x_t^i, \qquad \equiv \qquad X^{t+1} = X^t W \quad \text{in matrix form} \tag{273}$$

- That is, everyone shares their model updates and perform local aggregation. For general training problems, we can follow a similar pattern:

$$x_{t+1}^i = \sum_{j\in\mathcal{N}_i} W_{ij}\left(x_t^i - \eta g_t^i\right), \qquad \equiv \qquad X^{t+1} = \left(X^t - \eta G^t\right) W \quad \text{in matrix form} \tag{274}$$

where $G^t = [g_t^1, \ldots, g_t^K] \in \mathbb{R}^{d\times K}$.

Theorem

Assume the graph is connected and W is doubly stochastic with spectral gap ρ. Define the consensus error as

$$e_t = \frac{1}{K}\|X^t - \bar{X}^t\|_F^2 = \frac{1}{K}\sum_{j=1}^{K}\|x_t^j - \bar{x}_t\|^2 \tag{275}$$

where $\bar{x}_t = \frac{1}{K}\sum_{j=1}^{K} x_t^j$, and $\bar{X}^t = [\bar{x}_t, \ldots, \bar{x}_t] \in \mathbb{R}^{d\times K}$. Assume f is L-smooth. Furthermore assume we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, If $\eta = \mathcal{O}(\rho/\sqrt{T})$,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla f(\bar{x}_t)\|^2 + \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[e_t] = \mathcal{O}(\frac{1}{\sqrt{T}}\text{poly}(\frac{1}{\rho})). \tag{276}$$

- The average iterates $\bar{x}_t$ (an unknown quantity to the agents) is a first-order solution
- The consensus error vanishes so everyone is reaching $\bar{x}_t$ (consensus)

Proof Sketch

- Instead of studying the performance of K local models, we consider how the average model $\bar{x}_t$ (a virtual sequence unknown to the agents) performs. At the same time, we study the evolution of the important source of error, the consensus error.
- For consensus error, using our assumptions we can establish

$$\mathbb{E}[e_{t+1}] \leq (1-\rho)\mathbb{E}[e_t] + \text{ other terms} \tag{277}$$

  immediate reminding us of the potential-based analysis
- We then define a potential function

$$\Phi_t = \mathbb{E}[f(\bar{x}_t)] - f^* + C_e\mathbb{E}[e_t] \tag{278}$$

- Study the evolution of potential by summing over its consecutive differences
- When $\rho < 1$ we can find a good constant C_e to ensure convergence.

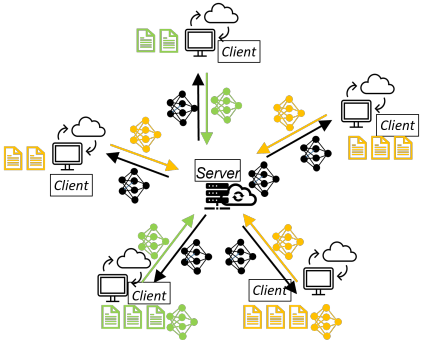
Compression Mechanisms

Distributed Learning

- Agents share model-based representation of local data through (repeatedly) communicating g_t^j with server (or each other)
- For $t = 1, \ldots, T$ Communication/training rounds
  - server sends the current global model $\bar{w}_t$ to the agents
  - each agent computes and sends g_t^j to the server
  - Server aggregates the recieved messages and update the global model

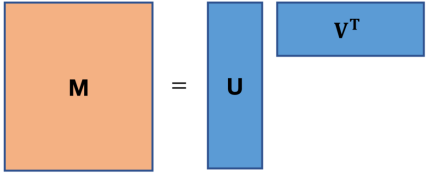
$$\bar{w}_{t+1} = \bar{w}_t - \eta \frac{1}{K} \sum_{j=1}^{K} g_t^j \qquad (279)$$

  - Recall $g_t^i \in \mathbb{R}^d$ and we need $\mathcal{O}(32d)$ bits for each message.
  - How to do this if agents have low communication resources?



Compression as a Solution

- We will send $\mathcal{C}(g_t^j)$ where $\mathcal{C}(\cdot)$ is a compression mechanism
- Quantization: reduce the precision (bits) of each coordinate from 32 to $q = 1, 2, 4, 8, 16$ bits, saving $\mathcal{O}(32/q)$ in communication resources
- Example: $x = [1.7, -2.6, -3.5]$ and $q = 1$ (sending the signs): $\mathcal{C}(x) = [1, -1, -1]$
- Sparsification: Only communicate k out of the d coordinates (and their indices), saving $\mathcal{O}(d/k)$ in communication resources
- Example: $x = [1.7, -2.6, -3.5]$ and $k = 1$ (selecting the largest entry): $\mathcal{C}(x) = [0, 0, -3.5]$
- Low-rank Factorization: When the gradients are matrices or tensors, find their low rank approximation and communicate that:



Good Compression Mechanisms and Examples

- $\mathcal{C}(g_t^j)$ can be thought of as the estimator of g_t^j. Thus, a compression is a good one if it has a low estimation error (low bias and variance)

$$\mathbb{E}\|\mathcal{C}(x) - x\|^2 = \underbrace{\mathbb{E}\|\mathcal{C}(x) - \mathbb{E}[\mathcal{C}(x)]\|^2}_{\text{Variance}} + \underbrace{\|\mathbb{E}[\mathcal{C}(x)] - x\|^2}_{\text{Bias}} \tag{280}$$

  where $\mathbb{E}$ is w.r.t. the random of $\mathcal{C}$

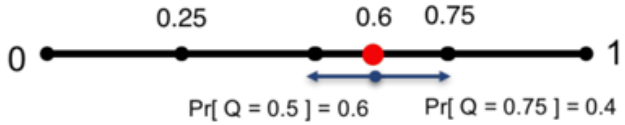
- Random Sparsification: Uniformly at random select k out of d coordinates.
  - It turns out $\mathbb{E}[\mathcal{C}(x)] = \frac{k}{d}x$ so we have some bias
  - It turns out $\mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq (1 - \frac{k}{d})\|x\|^2$
  - Can be made unbiased by using $\tilde{\mathcal{C}}(x) = \frac{d}{k}\mathcal{C}(x)$
  - But, doing so results in variance blowup: $\mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq (\frac{d}{k})^2\|x\|^2$
  - Takeaway 1: The biased version seems to be better in terms of estimation error
  - Takeaway 2: For both versions, the error vanishes as $x \to 0$, happens when x is (an estimator of) the gradient.

Good Compression Mechanisms and Examples (Cont'd)

- Top-k Sparsification: select k largest out of d coordinates
  - A deterministic operator and thus will have a bias
  - should be no worse that random sparsification so $\|\mathcal{C}(x) - x\|^2 \leq (1 - \frac{k}{d})\|x\|^2$
  - the error vanishes as $x \to 0$, happens when x is (an estimator of) the gradient.
- Random Dropout: set $\mathcal{C}(x) = x$ with probability p (full communication) and set $\mathcal{C}(x) = 0$ with probability $1 - p$ (no communication)
  - It turns out $\mathbb{E}[\mathcal{C}(x)] = px$ so we have some bias
  - It turns out $\mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq (1 - p)\|x\|^2$
  - Can be made unbiased by using $\tilde{\mathcal{C}}(x) = \frac{1}{p}\mathcal{C}(x)$
  - But, doing so results in variance blowup: $\mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq (\frac{1}{p})^2\|x\|^2$
  - Takeaway 1: The biased version seems to be better in terms of estimation error
  - Takeaway 2: For both versions, the error vanishes as $x \to 0$, happens when x is (an estimator of) the gradient.
  - Takeaway 3: on average, random dropout and random sparsification are identical with $p = k/d$

Random Quantization

- Normalize x by forming $u = \frac{x}{\|x\|} \odot \text{sign}(x)$
- Each coordinate of u is in $[0, 1]$ and we can quantize each separately and randomly into q levels to obtain $\tilde{u}$



- Send $\mathcal{C}(x)\|x\| \odot \text{sign}(x) \odot \tilde{u}$
  - It can be shown it is unbiased $\mathbb{E}[\mathcal{C}(x)] = x$
  - The variance satisfies $\mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq \min\{\frac{d}{s^2}, \frac{\sqrt{d}}{s}\}\|x\|^2$
  - q-bit random quantization saving us $\mathcal{O}(32/q)$ bits in communication
  - the error vanishes as $x \to 0$, happens when x is (an estimator of) the gradient.
  - We can rescaled it to reduce the variance at the cost of some bias.

Compression and Convergence

- Consider a compression mechanism such that

$$\mathbb{E}[\mathcal{C}(x)] = Ax, \qquad \mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq B\|x\|^2, \qquad 0 < A \leq 1, \;\; B \geq 0 \tag{281}$$

- For instance, $A = p$ and $B = 1 - p$ for random dropout. Or $A = 1$ and $B = \frac{d^2}{k^2}$ for unbiased random sparsification.
- Consider the following key step for SGD with compression, i.e. $w_{t+1} = w_t - \eta \mathcal{C}(g_t)$

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) + \langle w_{t+1} - w_t, \nabla f(w_t)\rangle + \frac{L}{2}\|w_{t+1} - w_t\|^2 \\ &= f(w_t) - \eta\langle \mathcal{C}(g_t), \nabla f(w_t)\rangle + \frac{L\eta^2}{2}\|\mathcal{C}(g_t)\|^2 \end{aligned} \tag{282}$$

- Taking expectation w.r.t. to $\mathcal{C}$ and then the randomness of current batch z_t conditioning on previous batches

$$\mathbb{E}[f(w_{t+1})] \leq f(w_t) - \eta A\|\nabla f(w_t)\|^2 + \frac{L\eta^2}{2}\mathbb{E}\|\mathcal{C}(g_t)\|^2 \tag{283}$$

Compression and Convergence (Cont'd)

- Note that

$$\mathbb{E}\|\mathcal{C}(g_t)\|^2 = \mathbb{E}\|\mathcal{C}(g_t) \pm g_t\|^2 = \mathbb{E}\|\mathcal{C}(g_t) - g_t\|^2 + \mathbb{E}\|g_t\|^2 + \mathbb{E}[\langle \mathcal{C}(g_t) - g_t, g_t\rangle] \tag{284}$$

- First term on RHS is bounded by $\mathbb{E}\|\mathcal{C}(g_t) - g_t\|^2 \leq B\mathbb{E}\|g_t\|^2$
- Third term on RHS can be handled as follows

$$\mathbb{E}[\langle \mathcal{C}(g_t) - g_t, g_t\rangle] = \mathbb{E}[\langle \mathcal{C}(g_t) - g_t \pm Ag_t, g_t\rangle] = \mathbb{E}[\langle (1-A)g_t, g_t\rangle] = (1-A)\mathbb{E}\|g_t\|^2 \tag{285}$$

  assuming $\mathcal{C}$ is independent of z_t and using $\mathbb{E}[\mathcal{C}(g_t)] = Ag_t$.
- Also recall using the SFO assumption $\mathbb{E}\|g_t\|^2 \leq \sigma^2 + \|\nabla f(w_t)\|^2$
- All in yields

$$\mathbb{E}[f(w_{t+1})] \leq f(w_t) - \eta\Big(A - \frac{L\eta}{2}(B+2-A)\Big)\|\nabla f(w_t)\|^2 + \frac{L\eta^2}{2}(B+2-A)\sigma^2 \tag{286}$$

Compression and Convergence (Cont'd)

- The rest is similar to SGD's analysis and we can obtain

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}\|\nabla f(w_t)\|^2 \leq \frac{f(w_1)-f^*}{T\eta\left(A-\frac{L\eta}{2}P\right)}+\frac{L\eta\sigma^2 P}{2A-L\eta P}, \qquad P:=(B+2-A) \tag{287}$$

- We now require a smaller learning rate for the result $\eta < \frac{2A}{LP}$ (vs. $\eta < 2/L$)
- There is effectively a new variance: $\tilde{\sigma}^2 = \sigma^2(B+2-A) \geq \sigma^2$. For instance:
  - for random dropout where $A = p$ and $B = 1-p$, $\tilde{\sigma}^2 = (3-2\rho)\sigma^2$
  - For unbiased random sparsification where $A = 1$ and $B = \frac{d^2}{k^2}$, $\tilde{\sigma}^2 = \sigma^2(1+\frac{d^2}{k^2})$
- The result reduces to SGD with no compression when $A = 1$ and $B = 0$.
- We can show $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(A^{-1}/\sqrt{T})$ when $A < 1$ and $B < 1$ (biased operators) while $\mathbb{E}\|\nabla f(\hat{w})\|^2 \simeq \mathcal{O}(B/\sqrt{T})$ when $A = 1$ and $B > 1$ (unbiased operators).
- Still converges, but now the dominant term is much worse as A^{-1} and B are very large (e.g. $\mathcal{O}(\frac{d}{k})$ and $\mathcal{O}(\frac{d^2}{k^2})$). Also, generally bias seems helpful

Error Feedback

- Let us focus on a biased compression operator such that $\mathbb{E}\|\mathcal{C}(x) - x\|^2 \leq (1-\delta)\mathbb{E}\|x\|^2$. This is called a contraction compression.
  - $\delta \to 1$ low compression (or a good operator)
  - $\delta \to 0$ high compression (or a bad operator)
- One way to interpret compression is that we are throwing out part of the information. How about we save it in memory, augment it with the next message, and the communicate that?
- In some sense, instead of throwing information out, we are communicating them with delay.
- And if we communicate long enough, everything more or less will be sent.
- Mathematically,

$$\text{communicate} \quad \mathcal{C}(x_t + e_t) \quad \text{instead of} \quad \mathcal{C}(x_t), \qquad e_{t+1} = (x_t + e_t) - \mathcal{C}(x_t + e_t) \tag{288}$$

- e_{t+1} is the compression error, $x_t + e_t$ is the uncompressed message, and $\mathcal{C}(x_t + e_t)$ is what we send out.

The EF-SGD Algorithm

- Initialize w_1 and $e_1 = 0$
- For $t = 1, \ldots, T$ do
  - Sample a batch $z_t \sim p_z$
  - Form the stochastic gradient $g_t = \nabla \ell(w_t, z_t)$
  - Compress and send $m_t = \mathcal{C}(g_t + e_t)$
  - Update the error $e_{t+1} = (g_t + e_t) - \mathcal{C}(g_t + e_t)$ and store it
  - Update the parameters $w_{t+1} = w_t - \eta m_t$

Theorem

Assume f is L-smooth and we have access to an SFO and that $z_1, \ldots, z_T$ are statistically independent. Then, If $\eta = \mathcal{O}(1/\sqrt{T})$, EF-SGD achieves $\mathbb{E}\|\nabla f(\hat{w})\|^2 \leq \mathcal{O}(\frac{1}{\delta T} + \frac{1}{\sqrt{T}})$.

- better dependence on the compression parameter as it appears only in the higher order term (as opposed to $\mathcal{O}(\frac{1}{\delta\sqrt{T}})$ for SGD without EF).

Proof Sketch

- Recall for decentralized SGD we considered how the average model (a virtual sequence unknown to the agents) performs. Here we also consider an idealized sequence with no compression error
- Define $\tilde{w}_t = w_t - \eta e_t$.
- Effectively removing the compression error from EF-SGD's iterates $w_{t+1} = w_t - \eta \mathcal{C}(g_t + e_t)$
- Intuitively, $\tilde{w}_t$ should resemble SGD's update with no compression, i.e.

$$\tilde{w}_{t+1} = \tilde{w}_t - \eta g_t \tag{289}$$

- Indeed that is the case:

$$\tilde{w}_{t+1} = w_{t+1} - \eta e_{t+1} = w_t - \eta \mathcal{C}(g_t + e_t) - \eta e_{t+1} = (w_t - \eta e_t) - \eta g_t = \tilde{w}_t - \eta g_t \tag{290}$$

where we used EF-SGD's update and $e_{t+1} = (g_t + e_t) - \mathcal{C}(g_t + e_t)$

Proof Sketch (Cont'd)

- We then use our potential-based analysis as before.
- We consider and analyze the evolution of the compression error $e_{t+1} = (g_t + e_t) - \mathcal{C}(g_t + e_t)$ and obtain

$$\mathbb{E}\|e_t\|^2 \leq \kappa(\delta)\mathbb{E}\|e_{t-1}\|^2 + \text{ other terms} \tag{291}$$

  for some $0 < \kappa(\delta) < 1$ depending only on the compression parameter
- We then define a potential function

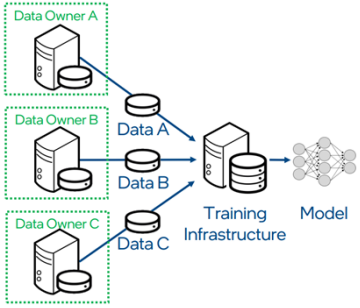
$$\Phi_t = \mathbb{E}[f(\tilde{w}_t)] - f^* + C_e\mathbb{E}\|e_t\|^2 \tag{292}$$

- Study the evolution of potential by summing over its consecutive differences
- When $\kappa(\delta) < 1$ we can find a good constant C_e to ensure convergence.

Privacy-Preserving DL

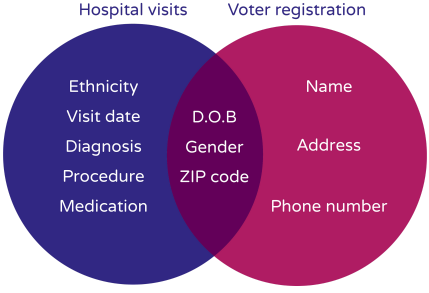
Data Integrity in Distributed DL

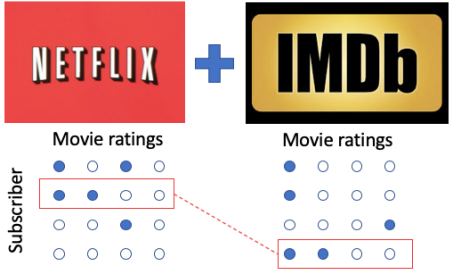
- Distributed
  learning in general leads to performance improvement
- However, information sharing opens the door to
  the possibility of sabotaging the security of personal data
- Ensuring data integrity (a
  context, hence an example of contextually constrained DL)



The need for Randomization

- Anonymization such as removing personally identifiers from data does not work due to linkage attacks (matching anonymized and non-anonymized data)
- Example: Netflix prize
- We need randomized safeguarding mechanisms
- Idea is to learn something useful about a population while not memorizing/learning something specific about an individual agent





Randomized Response

- Assume a survey is being conducted to determine the fraction of professors playing video games
- Assume professors playing video games is considered lame, which of course is not
- So, we want to learn something useful (the fraction) about a population (professors) while not memorizing/learning something specific about an individual agent (whether "professor Hashemi" plays a game)
- Let $x_i \in \{1, 0\}$ denote the truth about whether professor i plays games or not. And our goal is to estimate $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ for a population of n professors (e.g., $n = 120$ in Purdue ECE).
- Assume professor i will reveal an answer $y_i \in \{1, 0\}$ indicating whether they play games or not.
- Case 1: $y_i = x_i$
  - We can accurately estimate $\bar{x}$ by finding the average of y_i's
  - No privacy
- Case 2: $y_i = x_i$ with probability $1/2$ and $y_i = 1 - x_i$ with probability $1/2$
  - y_i's are useless and we cannot estimate $\bar{x}$ at all.
  - perfect privacy

Randomized Response (Cont'd)

- How about interpolating these two cases?
- $y_i = x_i$ with probability $1/2 + \varepsilon$ and $y_i = 1$ with probability $1/2 - \varepsilon$ where $0 \leq \varepsilon \leq 0.5$
- So on average when $\varepsilon > 0$, it is more likely to be truthful than lying. And in some sense provides plausible deniability.
- Let us calculate the mean of y_i

$$\mathbb{E}[y_i] = x_i(0.5 + \varepsilon) + (1 - x_i)(0.5 - \varepsilon) = 2\varepsilon x_i + (0.5 - \varepsilon) \tag{293}$$

meaning y_i is a biased estimator of x_i. Meaning, average of y_i may no be a good estimator of $\bar{x}$.

- After receiving the answers, we can form an unbiased estimator

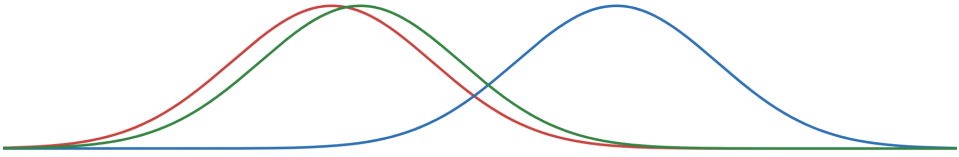
$$\hat{y}_i = \frac{y_i + \varepsilon - 0.5}{2\varepsilon}, \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} \hat{y}_i, \quad \mathbb{E}[\bar{y}] = \bar{x}, \quad \mathbb{E}|\bar{y} - \bar{x}| = \mathcal{O}(\frac{0.5 - \varepsilon}{\varepsilon\sqrt{n}}) \tag{294}$$

- Learning something useful ($\bar{x}$) with good accuracy about a population while not memorizing/learning something specific (x_i's) about an individual

Privacy, Sensitivity, and noise

- Assume a professor retires and the department hires a new assistant professor instead.
- Under Case 1 ($y_i = x_i$), our survey is highly sensitive to this change whereas under case 2 ($y_i = x_i$ with probability $1/2$ and $y_i = 1 - x_i$ with probability $1/2$) our survey is completely insensitive.
- Like before noise tends to reduce the sensitivity of our method and in this case improve privacy.
- this idea gives rise to Differential privacy, ensuring

$$\begin{aligned} &\mathbb{P}(\text{ survey (department } \cup \text{ retired)} \in \text{ outcome set)} \\ &\quad \approx \mathbb{P}(\text{ survey (department } \cup \text{ new hire)} \in \text{ outcome set)} \end{aligned} \tag{295}$$



Differential Privacy (DP)

Neighboring datasets

Two datasets $\mathcal{D} \in D_c$ and $\mathcal{D}' \in D_c$ are said to be neighboring if they differ in exactly one sample, and we denote this by $|\mathcal{D} - \mathcal{D}'| = 1$.

(ε, δ)-DP

Given a collection of datasets D_c, a randomized mechanism $\mathcal{M} : D_c \to \mathcal{Y}$ is said to be (ε, δ)-DP, if for any two neighboring datasets and all set of outcomes $\mathcal{R}$

$$\mathbb{P}(\mathcal{M}(\mathcal{D}) \in \mathcal{R}) \leq e^{\varepsilon}\mathbb{P}(\mathcal{M}(\mathcal{D}') \in \mathcal{R}) + \delta, \qquad \varepsilon, \delta \geq 0$$

- When $\delta = 0$, it is commonly known as pure DP. Otherwise, it is known as approximate DP.
- e^{ε} is for mathematical convenience. We could have used $1 + \varepsilon$ instead and note $1 + \varepsilon \approx e^{\varepsilon}$
- Symmetric definition despite looking otherwise (note it holds for any two $\mathcal{D}$ and $\mathcal{D}'$)

Differential Privacy (Cont'd)

- Can only be satisfied by randomized mechanism $\mathcal{M}$
- Smaller ε and δ imply stronger privacy guarantees (they control the privacy-accuracy tradeoff)
- The ratio

$$\log \frac{\mathbb{P}(\mathcal{M}(\mathcal{D}) \in \mathcal{R})}{\mathbb{P}(\mathcal{M}(\mathcal{D}') \in \mathcal{R})} \tag{296}$$

  is called the privacy loss and DP implies that with probability at least $1 - \delta$, the privacy loss is at most ε
- Typically, $\varepsilon < 0.1$ and $\delta \ll 1/n$ for a problem with n data points to ensure the privacy loss of each data point is small with high probability

Post Processing Theorem

If $\mathcal{M}$ is (ε, δ)-DP, so is its composition with any function h, i.e. $f(\mathcal{M}(\cdot))$.

No matter what others do to the released outcome $\mathcal{M}(\cdot)$, privacy is maintained if $\mathcal{M}(\cdot)$ is a (ε, δ)-DP.

Randomized Response as a DP Mechanism

- Going back to our example on professors and video games, let $\mathcal{D} = \{x_1, \ldots, x_n\}$ and $\mathcal{D}' = \{x_1', \ldots, x_n\}$
- By the Post Processing Theorem, we only need to show our process (i.e. $y_i = x_i$ with probability $1/2 + \varepsilon$ and $y_i = 1$ with probability $1/2 - \varepsilon$ where $0 \leq \varepsilon \leq 0.5$) is DP.
- Our mechanism is then $\mathcal{M}(\mathcal{D}) = [\mathcal{M}_1(x_1), \ldots, \mathcal{M}_n(x_n)]$ where $y_i = \mathcal{M}_i(x_i)$).
- Let $r \in \{0,1\}^n$ be a possible response profile and assume the responses are statistically independent

$$\frac{\mathbb{P}(\mathcal{M}(\mathcal{D}) = r)}{\mathbb{P}(\mathcal{M}(\mathcal{D}') = r)} = \frac{\prod_{i=1}^n \mathbb{P}(\mathcal{M}_i(x_i) = r)}{\mathbb{P}(\mathcal{M}_1(x_1') = r) \prod_{i=2}^n \mathbb{P}(\mathcal{M}_i(x_i) = r)} = \frac{\mathbb{P}(\mathcal{M}_1(x_1) = r)}{\mathbb{P}(\mathcal{M}_1(x_1') = r)} \leq \frac{0.5 + \varepsilon}{0.5 - \varepsilon} \approx e^{5\varepsilon} \tag{297}$$

  for all $\varepsilon \leq 0.3$. Thus, RR is $(5\varepsilon, 0)$-DP.
- Beyond single bit outcomes?

Beyond single bit outcomes

- RR can be thought of as adding the right amount of Bernoulli noise to $x_i \in \{1, 0\}$
- To guarantee DP for broad cases, we can similarly add noise of different kinds to the output.

Laplace Mechanism

Let $h : D_c \to \mathbb{R}^d$ be a function to $\mathbb{R}^d$. The Laplace mechanism is $\mathcal{M}(\mathcal{D}) = h(\mathcal{D}) + [e_1, \ldots, e_d]$ where each $e_i \sim \mathrm{Lap}(0, b)$ is an independent variable distributed according to the Laplace distribution

$$e \sim \mathrm{Lap}(\mu, b) \quad \equiv \quad p(e) \propto \exp(-\frac{|e-\mu|}{b}), \quad \mathbb{E}[e] = \mu, \quad \mathrm{Var}(e) = 2b^2 \tag{298}$$

Gaussian Mechanism

Let $h : D_c \to \mathbb{R}^d$ be a function to $\mathbb{R}^d$. The Gaussian mechanism is $\mathcal{M}(\mathcal{D}) = h(\mathcal{D}) + [e_1, \ldots, e_d]$ where each $e_i \sim \mathcal{N}(0, \sigma^2)$ is an independent variable distributed according to the Gaussian distribution

$$e \sim \mathcal{N}(\mu, \sigma^2) \quad \equiv \quad p(e) \propto \exp(-\frac{|e-\mu|^2}{2\sigma^2}), \quad \mathbb{E}[e] = \mu, \quad \mathrm{Var}(e) = \sigma^2 \tag{299}$$

Sensitivity in Gaussian and Laplace Mechanisms

- Consider Laplace mechanism and let us see how that leads to DP
- Note that in this case $\mathcal{M}(\mathcal{D})$ is a vector where each coordinate is distributed according to $\mathrm{Lap}(h_i(\mathcal{D}), b)$.
- Thus, $\frac{p(\mathcal{M}(\mathcal{D}))}{p(\mathcal{M}(\mathcal{D}'))}$ is simply the ratio of two Laplace distributions with different means
- Simple algebra and the definition of ℓ_1 norm, i.e. $\|x\|_1 = \sum_i x_i$ shows

$$\frac{p(\mathcal{M}(\mathcal{D}))}{p(\mathcal{M}(\mathcal{D}'))} \leq \exp\left(\frac{\|h(\mathcal{D}) - h(\mathcal{D}')\|_1}{b}\right) \tag{300}$$

implying $b \propto \frac{\|h(\mathcal{D}) - h(\mathcal{D}')\|_1}{\varepsilon}$ to ensure $(\varepsilon, 0)$-DP.

Sensitivity in Gaussian and Laplace Mechanisms (Cont'd)

- Thus,
$$b \propto \frac{\|h(\mathcal{D}) - h(\mathcal{D}')\|_1}{\varepsilon} \tag{301}$$
is adequate to ensure $(\varepsilon, 0)$-DP for the Laplace mechanism.
- A similar calculation shows
$$\sigma \propto \frac{\|h(\mathcal{D}) - h(\mathcal{D}')\|_2}{\varepsilon} \log \frac{1}{\delta} \tag{302}$$
to ensure Gaussian mechanism leads to (ε, δ)-DP
- Thus,
$$\max_{\mathcal{D},\mathcal{D}',|\mathcal{D}-\mathcal{D}'|=1} \|h(\mathcal{D}) - h(\mathcal{D}')\|_1, \qquad \max_{\mathcal{D},\mathcal{D}',|\mathcal{D}-\mathcal{D}'|=1} \|h(\mathcal{D}) - h(\mathcal{D}')\|_2 \tag{303}$$
are notions of sensitivity: the more sensitive our learning procedure h, the more noise needed to ensure privacy.
- Implications for DL optimization?

DP in DL Optimization

- Consider our typical learning problem

$$w^* = \arg\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i) \tag{304}$$

- Goal is to ensure an approximately optimal model $\hat{w}$ preserves the privacy of data $\mathcal{D} = \{z_1, \ldots, z_n\}$.
- We can leverage DP in three different ways
  - Output perturbation: adding noise to w^*
  - Objective perturbation: adding noise to $f(w)$
  - Gradient Perturbation: leveraging the fact that we use gradient-based methods for DL optimization and adding noise to gradients g_t

Output perturbation

- We hope to simply output $\hat{w} = w^* + e$ where e comes from Laplace or Gaussian dist.
- Unfortunately, The optimal solution w^* can change significantly if a single data point z_i is replaced, particularly when loss has a sharp or degenerate landscape.
- Implication: $\|w^*(\mathcal{D}) - w^*(\mathcal{D}')\|$ could be unbounded and DP cannot be guaranteed
- Under G-Lipschitzness and convexity of $f(w)$, the regularized problem

$$f_\lambda(w) := f(w) + \frac{\lambda \|w\|_2^2}{2} = \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) + \frac{\lambda \|w\|_2^2}{2} \tag{305}$$

will be λ-strongly convex and we can show

$$\|w^*(\mathcal{D}) - w^*(\mathcal{D}')\|_2 \leq \frac{2G}{\lambda n} \tag{306}$$

and we can use Gaussian Mechanism

- Two main drawbacks: need convexity and the ability to find the exact solution w^*

Objective Perturbation

- Instead of injecting noise into w^*, inject it into the regularized problem $f_\lambda(w)$ and solve

$$\min_w f_\lambda^e(w) := \frac{1}{n}\sum_{i=1}^n \ell(w, z_i) + \frac{\lambda\|w\|_2^2}{2} + \langle w, e\rangle, \quad e \sim \mathcal{N}(0, \sigma^2 I) \tag{307}$$

- Similar to output perturbation, for guaranteed DP, wee need to compute the optimal w^* exactly
- Let us consider the gradient descent update for such loss function

$$\nabla f_\lambda^e(w) = \frac{1}{n}\sum_{i=1}^n \nabla\ell(w, z_i) + \lambda w + e = \nabla f(w) + \lambda w + e, \qquad w_{t+1} = (1-\eta\lambda)w_t - \eta(\nabla f(w_t) + e) \tag{308}$$

That is, GD with weight decay where all gradients are perturbed with the same Gaussian noise

Gradient Perturbation

- Perturb sample gradients with noise in each iteration (DP-SGD)

$$z_t \sim p_z, \quad , g_t = \nabla \ell(w_t, z_t), \quad e_t \sim \mathcal{N}(0, \sigma^2 I), \quad w_{t+1} = w_t - \eta(g_t + e_t), \tag{309}$$

- We can achieve DP guarantees if $\|\ell(w_t, z_t)\|$ is bounded. True if the sample loss is Lipschitz
- In general, we have to resort to clipping

$$g_t = \nabla \ell(w_t, z_t), \quad \hat{g}_t^C = g_t \min\{1, \frac{C}{\|g_t\|} + \kappa\} \tag{310}$$

or normalization

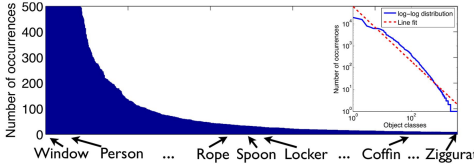
$$g_t = \nabla \ell(w_t, z_t), \quad \hat{g}_t^N = \frac{C g_t}{\|g_t\| + \kappa} \tag{311}$$

- Both method lead to biased update vectors.
- Can be extended to distributed learning paradigms when each g_t is communicated by a different agent.

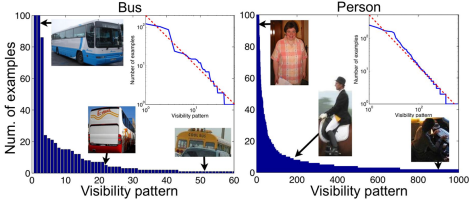
Memorization Phenomenon in Deep Learning

Motivation

- Differential Privacy: learn something useful about a population while not memorizing/learning something specific about an individual agent
- Noise-injection to Reduce sensitivity of our model to individual data points
- A natural trade-off between accuracy and privacy
- Is it always possible to learn something useful about a population while not memorizing/learning something specific about an individual agent?
- Role of data distribution: Long tailed vs Non-Long-Tail Distribution
- there are many small-frequency events (or values) that occur with relatively low probability



(a) The number of examples by object class in SUN dataset

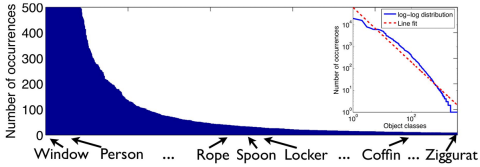


Long Tail Data Distributions

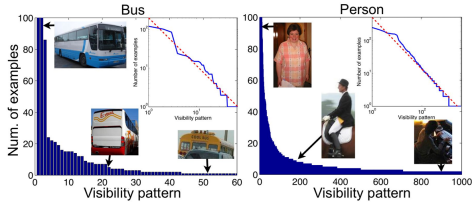
- A long-tail distribution
  is one where a large portion of the total probability mass is concentrated in the tail, which means there are many small-frequency events (or values) that occur with relatively low probability, but the tail of the distribution extends far out.
- Mathematically, the density follows Zipf's law

$$p(x) \propto x^{-s} \tag{312}$$

- In light-tail distribution such as Gaussian $p(x) \propto \exp(-x^2)$ the tail drops off much more quickly: Beyond a certain point, the probability of observing values far from the mean becomes extremely small.

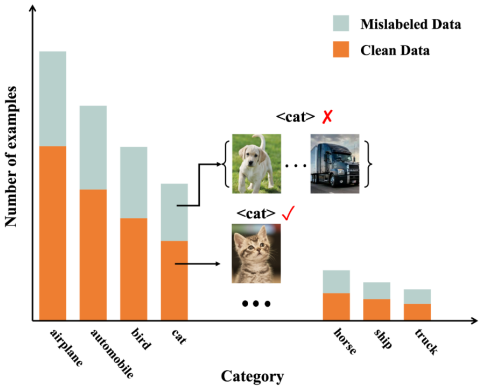


(a) The number of examples by object class in SUN dataset



Implication for Learning

- Long tail data points are atypical:
  they belong to subpopulations of size $\mathcal{O}(1/n)$
- If the
  model hopes to learn these subpopulations and generalize, it needs to memorize these samples
- Two questions arise:
  first what do we mean by memorization exactly?
  and secondly, are all atypical examples useful?
- Let us first
  answer the latter: No! atypical examples could include both "hard" examples as well as noisy (mislabeled, outliers, duplicated, etc.) examples



Memorization

- Memorization means model relies on specific training examples rather than learning general patterns applicable to unseen data

Definition

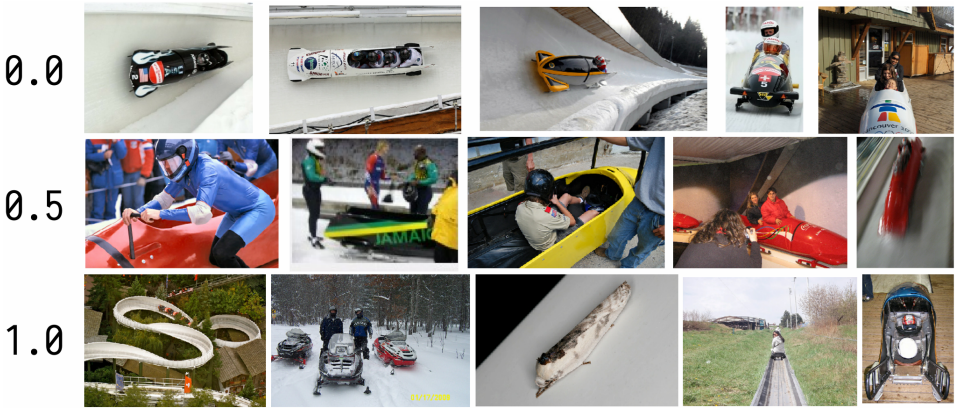
Let $\mathcal{A}$ a randomized algorithm (Such as SGD) applied to a training set S to learn a model h_S^w (e.g. a ResNet classifier) parameterized by w. Then, the memorization score for each data point is defined as

$$\mathrm{mem}(\mathcal{A}, S, z_i) = \mathbb{P}[h_S^w(x_i) = y_i] - \mathbb{P}[h_{S^{\setminus i}}^w(x_i) = y_i] \tag{313}$$

- measures the sensitivity of the model's predictions to the removal of individual data points from the training set.
- If removing a single example significantly alters the model's prediction for that example, the model is said to have memorized that example. Conversely, if the model's performance remains largely unchanged, it is considered more stable and less reliant on memorization.

Samples with Varied Memorization Scores

- Memorization Score for various data points from Class bobsled in ImageNet data set
- Low memorization score corresponds to typical and easy samples
- High memorization score corresponds to atypical (hard and also noisy) samples



Theorem

Let $e(\mathcal{A}, h^w, p_l)$ denote the expected generalization error of our model h^w trained by $\mathcal{A}$ for a long tailed distribution p_l. Let $e_{best}(h^w, p_l)$ denote the best achievable generalization error by any algorithm. Let $e_S(\mathcal{A}, 1)$ denote the number of examples that appear once (i.e. $freq_i = 1$) in the dataset S and are mislabeled by our model h^w trained by $\mathcal{A}$. Then,

$$e(\mathcal{A}, h^w, p_l) \geq e_{best}(h^w, p_l) + C \times e_S(\mathcal{A}, 1) \tag{314}$$

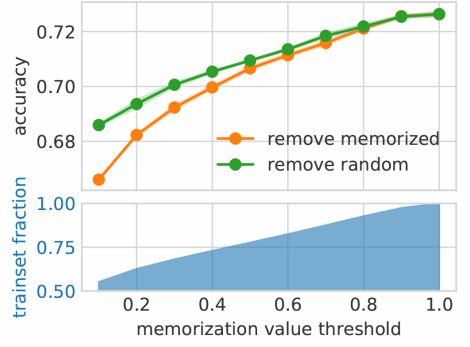
And crucially

$$e_S(\mathcal{A}, 1) = \sum_{i \in S;\ freq_i = 1} \mathbb{P}[h^w_{S \setminus i}(x_i) \neq y_i] - \mathrm{mem}(\mathcal{A}, S, z_i) \tag{315}$$

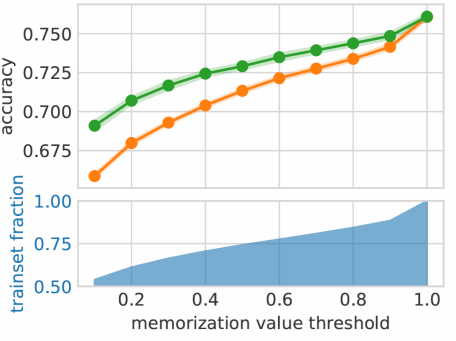
theorem implies to achieve the best performance, the model has to memorize atypical examples since $e_S(\mathcal{A}, 1)$ goes done as memorization goes up.

Removal Memorized Samples

- When removing samples with memorization score beyond a set threshold (e.g., 0.5) there is a significant drop in model's performance compared to random removal.
- Figure also demonstrates the long-tailed nature of data: more than 50% of samples have a memorization score below 0.1.



(a) ImageNet



(b) CIFAR-100

Memorization and Differential Privacy

- Recall, memorization score quantifies the sensitivity of the model's predictions to the removal of individual data points from the training set.
- Given the central role of sensitivity in the definition of both memorization and differential privacy, a natural question is about their relations

Theorem

Let $\mathcal{A}$ be a (ε, δ)-differentially private algorithm. Then with probability at least $1 - \delta$

$$\mathrm{mem}(\mathcal{A}, S, z_i) \leq 1 - e^{\epsilon}, \qquad \forall i \in S \tag{316}$$

- The theorem states that DP implies low memorization tendency (other direction?)
- As the privacy parameters go down, with increasing probability, $e^{\epsilon} \to 1$ and the memorization score goes down as well.
- This again highlights the privacy-accuracy tradeoff: In long tailed data distribution where memorization is essential, strong DP guarantees could hurt performance.

Input Loss Curvature and Memorization

- Recall, memorization score quantifies the sensitivity of the model's predictions to the removal of individual data points from the training set.
- The elegant definition of memorization is hard to compute and as mentioned it does not distinguish between truly hard examples and the mislabeled ones (no need to generalize well on a noisy sample).
- Therefore, there is a need for computationally efficient proxies that can also make this important distinction
- Our main tool to train a model is through minimizing the training loss. Thus, one could think of characterizing sensitivity by looking at loss $\ell(h_S^w, z_i)$ and its variation with respect to z_i
- Formally:

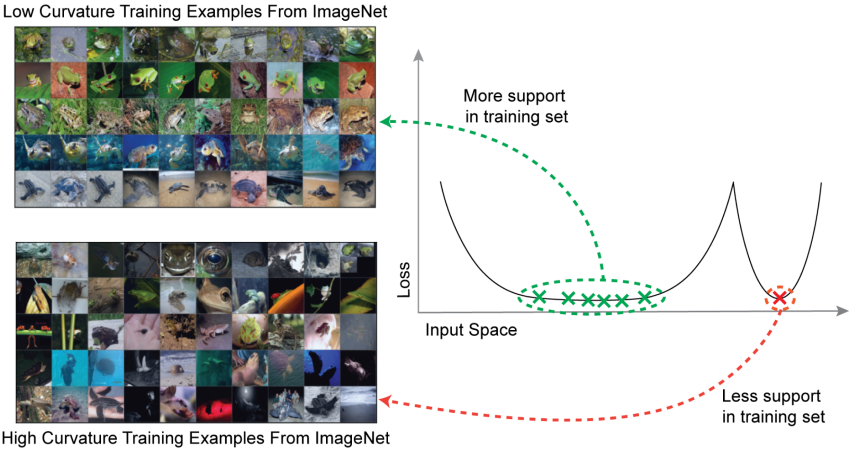
$$\mathrm{Curv}_w(z_i, S) = \mathrm{tr}(H_i) = \mathrm{tr}(\nabla^2_{z_i} \ell(h_S^w, z_i)) \tag{317}$$

Theorem

Under certain assumptions we can show $\mathrm{mem}(\mathcal{A}, S, z_i) \leq A\mathrm{Curv}_w(z_i, S) + B$

Input Loss Curvature and Memorization (Cont'd)

- curvature captures the local structure of the loss landscape:



Input Loss Curvature and Memorization (Cont'd)

- In practice, accumulated curvature $\sum_{t=1}^{T} \mathrm{Curv}_{w_t}(z_i, S)$ over the training process leads to higher correlation with memorization
- Curvature can be approximated efficiently using using Hutchinson's trace estimator and zeroth-order gradient estimation

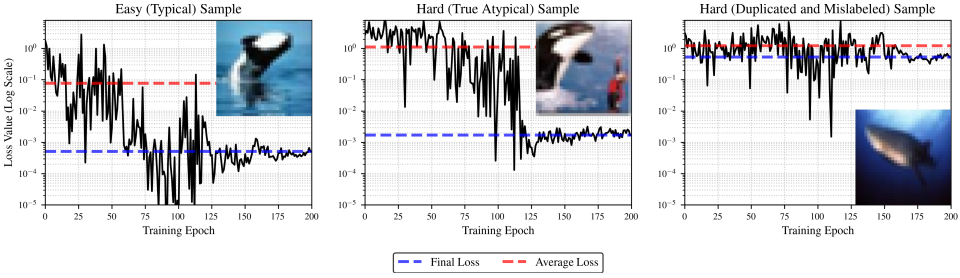
$$\mathrm{Curv}_w(z_i, S)^2 \propto \frac{1}{M} \sum_{j=1}^{M} \|\nabla_{z_i} \ell(h_S^w, z_i + e_j) - \nabla_{z_i} \ell(h_S^w, z_i)\|^2, \qquad e_j \sim p_e \tag{318}$$

- Viewing curvature as second-order sensitivity, we can use first-order (input gradient $\|\nabla_{z_i} \ell(h_S^w, z_i)\|$) and zeroth-order (input loss) and their cumulative versions for more efficient memorization proxies, leading to broad family of differential input sensitivity (DIS) proxies
- One potential benefit of DIS is the ease to study memorization in unsupervised and generative models (active area of research)

Distinguishing Noisy and Hard Examples

- Both typical and atypical samples exhibit a low final loss. Noisy samples maintain a relatively high final loss.
- The average loss for typical samples remains low, whereas it is high for both atypical and noisy samples
- Could lead to the following criterion which is large for truly hard samples

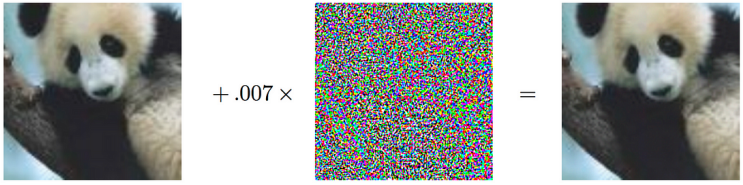
$$\text{GoodMem} = \text{CumulativeDIS} - \text{FinalDIS} \tag{319}$$



Robustness

Motivation

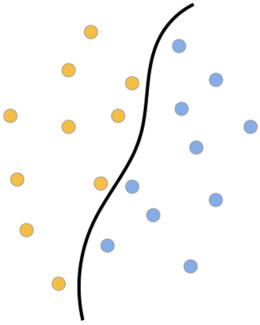
- Our DL models will be used in diverse settings where the conditions of the training process are not necessarily met
- For instance, they may need to withstand changes in the test data compared to training data
  - Change due to bad actors
  - change due to out-of-domain (OOD) Samples
- Thus, we want to induce robustness or resiliency into our training process

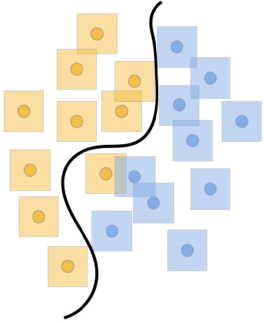


Robustness in Supervised Learning

- Main idea: Reduce sensitivity with respect to changes or perturbations in the data

$$\min_{w} \max_{e:\|e\|\leq\rho} \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i + e) \tag{320}$$





Adversarial Training

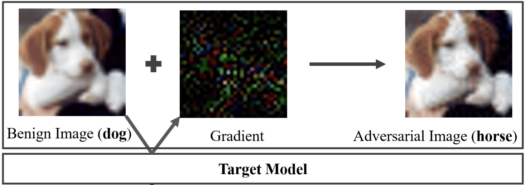
- During the training process, perturb the data according to projected gradient descent:

$$\tilde{z}_t = \text{Proj}\Big(z_t + \tilde{\eta}\nabla_z \ell(w_t, z_t)\Big) \tag{321}$$

- Note: the gradient is w.r.t. the data not parameter w
- the projection aims to ensure to perturbation is not too large and remains bounded
- Then, use the perturbed data $\tilde{z}_t$ (as opposed to the original data z_t) to update the model

$$g_t = \nabla \ell(w_t, \tilde{z}_t), \qquad w_{t+1} = w_t - \eta g_t \tag{322}$$

- Effectively trying to estimate the worst perturbation/attach for each data point



Input Gradient Regularization

- Instead of iteratively solving the max, can we find a closed-form expression for the worst-case perturbation?
- Let us write the first-order Taylor approximation around z

$$\ell(w, z+e) \approx \ell(w, z) + \langle \nabla \ell(w, z), z + e - z \rangle \tag{323}$$

which will be an affine function of e.

- It turns out the problem below has a closed form solution

$$\arg \max_{e:\|e\| \leq \rho} \ell(w, z) + \langle \nabla_z \ell(w, z), e \rangle = \rho \frac{\nabla_z \ell(w, z)}{\|\nabla_z \ell(w, z)\|} \tag{324}$$

Leading to the Input gradient regularized robust formulation

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) + \rho \|\nabla_z \ell(w, z_i)\| \tag{325}$$

- Note: Idea similar to sharpness-aware minimization, but there we regularized with gradient w.r.t. w instead

Less Pessimistic Solutions

- Previous approaches based on estimating the worst-cased perturbation could be pessimistic
- A way around is to use random perturbations instead

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{e\sim p_e}[\ell(w, z_i + e)] \tag{326}$$

resembling Gaussian Smoothing (convolution with the PDF of a nice distribution)

- Assume the noise is zero mean and $\mathbb{E}[ee^\top] = 2\rho I_p$. Let us write the second-order Taylor approximation around z

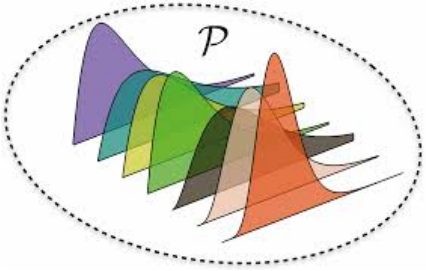
$$\ell(w, z + e) \approx \ell(w, z) + \langle \nabla_z \ell(w, z), e\rangle + \frac{1}{2} e^\top \nabla_z^2 \ell(w, z) e \tag{327}$$

Noting $e^\top \nabla_z^2 \ell(w, z) e = \text{Trace}(e^\top \nabla_z^2 \ell(w, z) e)$ and after using the linearity of trace and expectation leads to

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i) + \rho \text{Trace}(\nabla_z^2 \ell(w, z_i)) \tag{328}$$

i.e., an input curvature regularization method.

Distributionally Robust Optimization (DRO)



- To induce robustness, the previous approaches were based on perturbation of the input data. Intuitively, they aimed to guess the data distribution during inference. An alternative approach is to directly robustify the changes in the input data distribution:

$$\mathbb{P}(z = z_i) = 1/n \quad \rightarrow \quad \mathbb{P}(z = z_i) = p_i \qquad (329)$$

  where $D(p\|1/n) \leq \rho$.

- Equivalently, if $\mathcal{P}_\rho^D = \{p : D(p\|1/n) \leq \rho\}$ solve the problem

$$\min_w \max_{p \in \mathcal{P}_\rho^D} \mathbb{E}_{z\sim p}[\ell(w, z)]$$

Example: KL-based DRO

- Assume $D(p\|1/n)$ is the KL divergence:

$$D(p\|1/n) = \sum_{i=1}^{n} p_i \log \frac{p_i}{1/n} \tag{330}$$

- After certain steps (crucially using the duality theory), our DRO problem will reduce to

$$\min_w \rho^{-1} \log \Big(\frac{1}{n} \sum_{i=1}^{n} \exp(\rho \ell(w, z_i)) \Big) \tag{331}$$

- Easy to check when $\rho \to 0$, the objective reduces to $\frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i)$ (non-robust) while when $\rho \to \infty$ it reduces to $\max_i \ell(w, z_i)$ (pessimistic robustness)
- Since log is a monotone function, the problem is equivalent to

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \exp(\rho \ell(w, z_i)) \tag{332}$$

  i.e., a new ERM with sample loss $\tilde{\ell}(w, z_i) = \exp(\rho \ell(w, z_i))$.

- Consider the loss prior to removal of log

$$\rho^{-1} \log \left(\frac{1}{n} \sum_{i=1}^{n} \exp(\rho \ell(w, z_i)) \right) \tag{333}$$

- The gradient can be calculated as

$$\frac{\frac{1}{n} \sum_{i=1}^{n} \exp(\rho \ell(w, z_i)) \nabla \ell(w, z_i)}{\frac{1}{n} \sum_{i=1}^{n} \exp(\rho \ell(w, z_i))} = \sum_{i=1}^{n} \alpha_i^{\rho}(w) \nabla \ell(w, z_i), \quad \alpha_i^{\rho}(w) = \frac{\exp(\rho \ell(w, z_i))}{\sum_{j=1}^{n} \exp(\rho \ell(w, z_j))} \tag{334}$$

  i.e., a weighted average of sample gradients where the weights depend on the parameter

- Thus, we can intuitively (but not literally) think of the loss as a weighted ERM where the weights for each sample depend on the parameter w: $\mathbb{P}(z = z_i) = \alpha_i^{\rho}(w)$
- The weights are designed to promote samples with higher loss, thereby inducing robustness

Constrained Learning

Motivation

- We talked about numerous learning setups where

$$\min_{w} \mathbb{E}_z[\ell(w,z)] + \lambda h(w) \tag{335}$$

  where $h(w)$ was some regularization term and $\lambda \geq 0$ a regularization parameter
- How to determine λ? What if we want a certain condition satisfied in a very strong sense?
- These questions lead to a general Constrained Learning setup

$$\min_{w} \mathbb{E}_z[\ell(w,z)] \qquad \text{s.t.} \quad h(w) \leq \tau \tag{336}$$

- The constraint $h(w)$ can itself be a stochastic function

$$h(w) = \mathbb{E}_{\tilde{z}}[\tilde{\ell}(w,\tilde{z})] \tag{337}$$

- We can set $\tau = 0$ without loss of generality by redefining $h(w) \leftarrow h(w) - \tau$

- Multi-class Neyman-Pearson classification: the loss of a prioritized class is minimized while the rest ones are below a certain threshold denoted by

$$\begin{aligned} \min_{w} \quad & f_1(w) \\ \text{s.t.} \quad & f_i(w) \leq \tau_i, \quad i = 2, \ldots, K \end{aligned} \tag{338}$$

where $f_i(w) = \mathbb{E}_{z_i}[\ell(w, z_i)]$ is the loss function of each class, and here class 1 is set as the prioritized one.

- Physics-Informed Neural Networks (PINNs): learning a DL model that approximates the solution to a partial differential equation (PDE).

$$\begin{aligned} \min_{w} \quad & \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, M_w(x_i)) \\ \text{s.t.} \quad & \mathbb{E}_{(x,t)\sim\mathcal{D}_{\textbf{physics}}}\left[\|\mathcal{N}[M_w](x,t)\|^2\right] = 0 \end{aligned} \tag{339}$$

where the distribution $\mathcal{D}_{\text{physics}}$ represents a set of points sampled from the domain where the physical constraints (e.g., differential equations) must be enforced and the operator $\mathcal{N}[M_w]$ represents the governing laws of the system, typically through a PDE.

Bringing the Constraint into the Objective

- Consider the problem

$$\min_{w} \mathbb{E}_z[\ell(w, z)] \qquad \text{s.t.} \quad h(w) \leq 0 \tag{340}$$

- If we define the indicator function of the constraint as

$$\mathbb{I}\{h(w)\} = \begin{cases} 0, & h(w) \leq 0 \\ \infty, & h(w) > 0 \end{cases} \tag{341}$$

  we can equivalently solve

$$\min_{w} \mathbb{E}_z[\ell(w, z)] + \mathbb{I}\{h(w)\} \tag{342}$$

- The indicator function shows our amount of displeasure regarding the violation of constraint. Maximum displeasure leads to an equivalent problem, but not easily approachable. Other choices such as linear, quadratic, or the combination of the two lead to easier approaches
- This idea leads to penalty-based and Lagrangian-based formulations.

Weaker Displeasure Notations

- Penalty-Based method

$$\min_w \mathbb{E}_z[\ell(w,z)] + \frac{\rho}{2}([h(w)]_+)^2, \qquad \rho \geq 0 \tag{343}$$

where $[\cdot]_+ = \max(\cdot, 0)$

- Lagrangian (linear penalty) method

$$\min_w \mathbb{E}_z[\ell(w,z)] + \lambda h(w), \qquad \lambda \geq 0 \tag{344}$$

- Augmented Lagrangian method

$$\min_w \mathbb{E}_z[\ell(w,z)] + \lambda h(w) + \frac{\rho}{2}([h(w)]_+)^2 \quad \equiv \quad \min_w \mathbb{E}_z[\ell(w,z)] + \frac{\rho}{2}\left(\left[h(w) + \frac{\lambda}{\rho}\right]_+\right)^2 \tag{345}$$

Lagrangian

- The function $\mathcal{L}(w, \lambda) = \mathbb{E}_z[\ell(w, z)] + \lambda h(w)$ is called the Lagrangian associated with the original constrained learning task
- Note Lagrangian is linear in λ, thus, minimizing it w.r.t. w leads to a concave function which we call the Lagrange dual function

$$g(\lambda) = \min_w \mathcal{L}(w, \lambda) = \min_w \Big(\mathbb{E}_z[\ell(w, z)] + \lambda h(w) \Big) \tag{346}$$

Lagrangian (Cont'd)

- Lagrange dual function provides a lower bound on the optimal value of the original problem for each λ. For any feasible $\tilde{w}$

$$g(\lambda) = \min_{w} \mathcal{L}(w, \lambda) \leq \mathcal{L}(\tilde{w}, \lambda) = \mathbb{E}_z[\ell(\tilde{w}, z)] + \lambda h(\tilde{w}) \leq \mathbb{E}_z[\ell(\tilde{w}, z)] \tag{347}$$

  where we used $h(\tilde{w}) \leq 0$ for any feasible solution $\tilde{w}$ including the optimal one w^*.

- A natural question is: What is the best lower bound that can be obtained from the Lagrange dual function? This leads to the Lagrange dual problem

$$\max_{\lambda \geq 0} g(\lambda) \qquad \text{s.t.} \qquad \equiv \qquad \max_{\lambda \geq 0} \min_{w} \mathbb{E}_z[\ell(w, z)] + \lambda h(w) \tag{348}$$

- Under certain conditions (Slater's condition and strong duality), one can change the order of max and min and obtain an equivalent problem (to both dual and our original problem)

$$\max_{\lambda \geq 0} \min_{w} \mathbb{E}_z[\ell(w, z)] + \lambda h(w) \qquad \equiv \qquad \min_{w} \max_{\lambda \geq 0} \mathbb{E}_z[\ell(w, z)] + \lambda h(w) \tag{349}$$

Solving the Dual Problem via GD

- Let $\mathcal{L}(w, \lambda)$ denote the (augmented) Lagrangian function
- We can try to solve the min max (or max min) problem iteratively
- For $t = 1, \dots T$
  - $w_{t+1} = \min_w \mathcal{L}(w, \lambda_t)$
  - $\lambda_{t+1} = \max_{\lambda \geq 0} \mathcal{L}(w_{t+1}, \lambda)$
- The updates (after some approximations and under certain assumptions) can be interpreted as gradient descent on w and gradient ascent on λ:
  - $w_{t+1} = w_t - \eta \nabla_w \mathcal{L}(w_t, \lambda_t)$
  - $\lambda_{t+1} = [\lambda_t + \eta \nabla_\lambda \mathcal{L}(w_{t+1}, \lambda_t)]_+$

Switching Gradient Method (SGM)

- Consider our original problem

$$\min_{w} f(w) \qquad \text{s.t.} \quad h(w) \leq 0 \tag{350}$$

  and assume we are working with an iterative method.

- Note that the constraint need not be considered for an iterate w_t as long as $h(w_t) \leq \epsilon$. And we can just use GD or a similar method $w_{t+1} = w_t - \eta \nabla f(w_t)$
- On the other hand, if $h(w_t) > \epsilon$, we may want to prioritize the constraint over the objective by performing the following update as needed

$$w_{t+1} = w_t - \eta \nabla h(w_t) \tag{351}$$

- Effectively, depending on the amount of violation, we switch the gradient in GD
- For $t = 1, \ldots, T$
  - if $h(w_t) \leq \epsilon$ perform $w_{t+1} = w_t - \eta \nabla f(w_t)$
  - if $h(w_t) > \epsilon$ perform $w_{t+1} = w_t - \eta \nabla h(w_t)$

Switching Gradient Method (Cont'd)

- The method can be thought of as a kind of SGD for the dual problem.
- Consider $\min_w \max_{\lambda \geq 0} f(w) + \lambda h(w)$ and let $\lambda^* \geq 0$ be the maximizer. Then,

$$\min_w f(w) + \lambda^* h(w) \qquad \equiv \qquad \min_w \frac{1}{1+\lambda^*} f(w) + \frac{\lambda^*}{1+\lambda^*} h(w) \tag{352}$$

assuming λ^* is constant, for which the GD update is

$$w_{t+1} = w_t - \eta \left(\frac{1}{1+\lambda^*} \nabla f(w_t) + \frac{\lambda^*}{1+\lambda^*} \nabla h(w_t) \right) = w_t - \eta \mathbb{E}[\nabla \tilde{\ell}(w_t, z)] \tag{353}$$

where $z \in \{1, 2\}$, $\mathbb{P}(z = 1) = \frac{1}{1+\lambda^*}$, and $\mathbb{P}(z = 2) = \frac{\lambda^*}{1+\lambda^*}$, implying

$$\nabla \tilde{\ell}(w_t, z) = \begin{cases} \nabla f(w_t) & w.p. \quad \frac{1}{1+\lambda^*} \\ \nabla h(w_t) & w.p. \quad \frac{\lambda^*}{1+\lambda^*} \end{cases} \tag{354}$$

Second-Order Guarantees

Higher Quality Solutions

- Consider our training problem

$$\min_w f(w) := \mathbb{E}[\ell(w, z)] \tag{355}$$

- We showed SGD with update $w_{t+1} = w_t - \eta g_t$ satisfies $\mathbb{E}\|\nabla f(\hat{w})\|^2 \leq \epsilon$ using $T = \Omega(\epsilon^{-2})$ iterations
- This is called a first-order solution and in principle can be a local minimum, local maximum, or a saddle point
- Example: $f_1(w) = \frac{1}{3}w^3 - w$

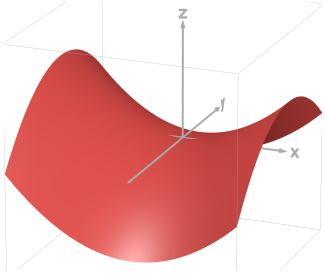
Higher-order Solutions (Cont'd)

- Example: $f_2(w) = \frac{1}{3}w^3$

- Example: $f_3(w) = \frac{1}{4}w^4$

- Example: $f_4(w) = \frac{1}{2}(w^2 - 1)^2$

- Example: $f(x, y) = x^2 - y^2$



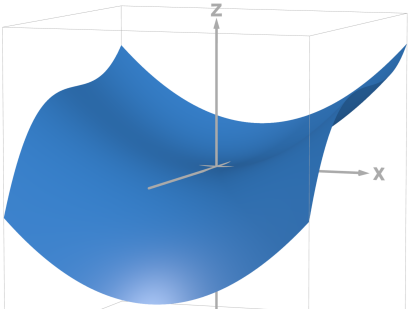
Definition

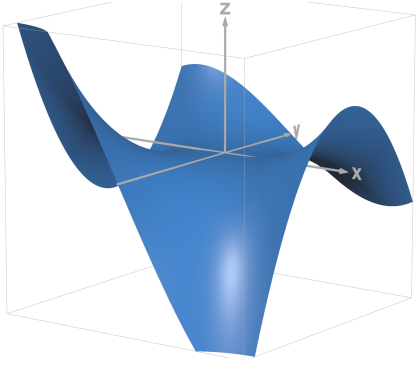
An (ϵ_1, ϵ_2)-accurate second order stationary solution for f is any $\hat{w}$ such that

$$\|\nabla f(\hat{w})\|^2 \leq \epsilon_1, \qquad \lambda_{\min}(\nabla^2 f(\hat{w})) \geq -\epsilon_2 \tag{356}$$

Which Solutions?

- The definition excludes local maximum and simple/strict saddle points (detectable by looking at Hessian).
- It does not however exclude higher order saddle points (undetectable by looking at Hessian)
- Examples: $x^2 + y^3$ (left) and monkey saddle, i.e. $x^3 - 3xy^2$ (right)



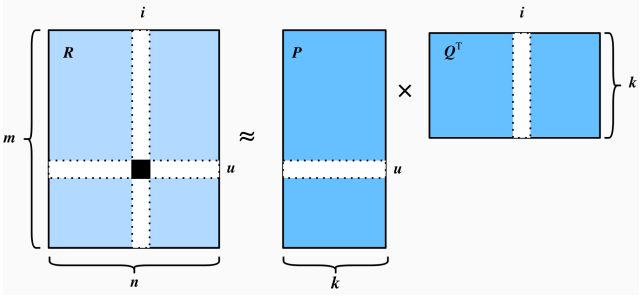


Why a Good Definition

- Many ML problems only include local/global minima and simple saddle points
- Example: Matrix Factorization

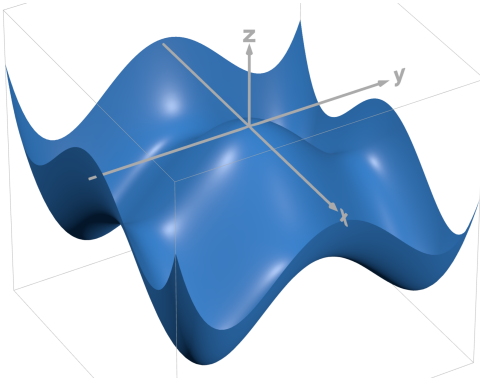
$$P^*, Q^* = \arg\min_{U,V} \|R - PQ^\top\|_F^2 \tag{357}$$

- Permutation of columns of P^* and Q^* leads to the same solution
- Similar idea applies to DL models and permutations weight matrices
- These exponentially many global minima are separated by exponentially many simple saddle points



Exponentially Many Saddles and Minima

- Consider the function $h(x) = (1 - x^2)^2$ and from it construct $h(x, y) = f(x) + f(y)$
  - 4 minima and 4 simple saddle points
  - $h(x, y, z) = f(x) + f(y) + f(z)$ will have 8 minima and 8 simple saddle points



Finding 2nd order Solutions

- Recall to find a first-order solution
  - We required accessing the gradient or a good approximation of that
  - We imposed smoothness, i.e., Lipschitzness of gradient
  - gradient is a first-order quantity
- Thus, intuitively, we expect to need the following
  - We require accessing the Hessian or a good approximation of that
  - We impose 2nd-order smoothness, i.e., smoothness of gradient, or Lipschitzness of Hessian
  - Hessian is a second-order quantity
- We will indeed require Lipschitzness of Hessian. However, we will discuss two methods one requiring Hessian and one requiring only (stochastic gradient)
- Hessian information will be useful to detect saddles and avoid them. Gradient noise injection could help avoid saddle points by taking advantage of their instability.
- In both cases, by avoid we mean find a direction to decrease the function value.

Second-Order Methods: Newton Algorithm

- Newton method is a celebrated algorithm for calculating zeros of functions iteratively:

$$f(x^*) = 0 \qquad x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)} \tag{358}$$

  Implication for optimization: finding zeros of gradient: $\nabla f(w^*) = 0$

$$w_{t+1} = w_t - H_t^{-1} \nabla f(w_t), \qquad H_t = \nabla^2 f(w_t) \tag{359}$$

- Newton Variations
  - Damped Newton: $H_t^{-1} \to \eta_t H_t^{-1}$
  - Levenberg–Marquardt Regularization (LMR) $H_t^{-1} \to (\eta_t^{-1} I_d + H_t)^{-1} \approx \eta_t (I_d - \eta_t H_t)$
- When $H_t^{-1} \to \eta I_d$ we obtain gradient descent.
- Effectively improves the local landscape conditions by leveraging the preconditioning matrix $C_t = H_t^{-1}$

- Recall, we previously discussed GD as an example of MM for L-smooth functions when $\eta \leq 1/L$

$$w_{t+1} = \arg\min_{w} f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \frac{1}{2\eta}\|w - w_t\|^2 \tag{360}$$

- We can easily show Newton update is equivalent to minimizing the 2nd-order Taylor approximation at w_t

$$w_{t+1} = \arg\min_{w} f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t (w - w_t) \tag{361}$$

- Note $(w - w_t)^\top H_t (w - w_t) \leq L\|w - w_t\|^2$ meaning Newton is not an instance of MM.
- Intuitively, Newton is designed to solve $\nabla f(w) = 0$ not $\nabla f(w) = 0$ such that $\nabla^2 f(w) \succeq 0$
- For this reason, Newton may not help us to avoid saddles effectively. But a variation of it will.
- Idea is to form an upper-bound and minimize that iteratively by assuming and leveraging Lipschitzness of Hessian as opposed to Lipschitzness of gradient.

Definition

A function f is ρ second-order smooth if $\|\nabla^2 f(w_1) - \nabla^2 f(w_2)\|_2 \leq \rho \|w_1 - w_2\|$ for all $w_1, w_t \in \mathbb{R}^d$

- It turns out this is equivalent to

$$f(w_1) \leq f(w_2) + \langle \nabla f(w_2), w_1 - w_2 \rangle + \frac{1}{2}(w_1 - w_2)^\top \nabla^2 f(w_2)(w_1 - w_2) + \frac{\rho}{6}\|w_1 - w_2\|^3 \quad (362)$$

$$\|\nabla f(w_1) - \nabla f(w_2) - \nabla^2 f(w_2)(w_1 - w_2)\| \leq \frac{\rho}{2}\|w_2 - w_1\|^2 \quad (363)$$

- Function has local cubic bound and the gradient is ρ-smooth (it has quadratic bound)
- Cubic Regularization of Newton Method (CRNM):

$$w_{t+1} = \arg\min_w f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t (w - w_t) + \frac{\rho}{6}\|w - w_t\|^3 \quad (364)$$

- Interestingly, the update is equivalent to a convex optimization problem when $d \geq 2$

CRNM Idea

- Under ρ-Lipschitz Hessian Hessian, the method is MM thus, we continually reduce the function value
- Consider the spectral decomposition of Hessian $H_t = \sum_{i=1}^{d} \lambda_i u_i u_i^\top$
- So the update can be written equivalently as

$$w_{t+1} = \arg\min_{w} f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \frac{1}{2} \sum_{i=1}^{d} \lambda_i \left(u_i^\top (w - w_t) \right)^2 + \frac{\rho}{6} \|w - w_t\|^3 \quad (365)$$

  - We are penalizing directions u_i with positive λ_i as they are the direction of increase.
  - We are taking advantage of directions u_i with negative λ_i as they are the direction of decrease.

Theorem

CRNM can find an $(\epsilon, 2\sqrt{\rho\epsilon})$-second order solution in $T = \Omega(\frac{\sqrt{\rho}(f(w_1) - f^*)}{\epsilon^{0.75}})$.

Proof Sketch

- If there exists a direction with sufficiently large (and negative) eigen value such that $\lambda < -\sqrt{\rho\epsilon}$, we can reduce the function value

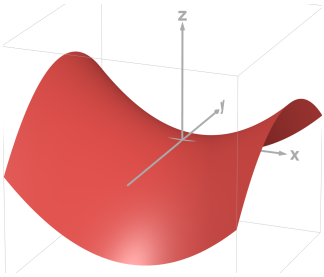
$$f(w_{t+1}) - f(w_t) \leq -\frac{2\lambda^3}{3\rho^2} \qquad \text{compare with} \qquad f(w_{t+1}) - f(w_t) \leq -\frac{2\|\nabla f(w_t)\|^2}{2L} \qquad \text{for GD} \tag{366}$$

- Otherwise, there are two cases. First, if $\|w_{t+1} - w_t\| \leq \epsilon\rho^{-1}$ then $\|\nabla f(w_{t+1})\|^2 \leq \epsilon$ meaning both conditions of approximate 2nd order definition are met and we find a solution.
- Secondly, if $\|w_{t+1} - w_t\| > \epsilon\rho^{-1}$, we can still make progress and reduce the function value

$$f(w_{t+1}) - f(w_t) < -\frac{\epsilon^{0.75}}{6\sqrt{\rho}} \tag{367}$$

- By contradiction we then show a solution is found after precisely $T = \Omega(\frac{\sqrt{\rho}(f(w_1) - f^*)}{\epsilon^{0.75}})$ iterations.

Avoiding Saddles with Perturbed GD: Motivation



- Consider $f(x, y) = 0.5(x^2 - y^2)$.
  If we initialize at $(0, 0)$ GD will not move.
- However, the saddle point is instable and if we
  inject random noise in the gradient, GD will eventually escape it

$$x_{t+1} = -\eta \sum_{\tau=1}^{t} (1-\eta)^{t-\tau} e_\tau(x), \qquad y_{t+1} = -\eta \sum_{\tau=1}^{t} (1+\eta)^{t-\tau} e_\tau(y) \tag{368}$$

  where $e_\tau(x)$ and $e_\tau(y)$ are the random noise
  added to the x and y component of the gradient, respectively.
- Remark: SGD has baked in noise, but explicit injection is needed in general.
- Remark: With random initialization, the chance of starting at a bad initialization (saddles and local maxima) is very low

Perturbed GD

- In each iteration run

$$w_{t+1} = w_t - \eta \nabla f(w_t) + \eta e_t, \qquad e_t \sim S(0,1) \quad \text{or} \quad e_t \sim B(0,1) \quad \text{or} \quad e_t \sim \mathcal{N}(0, I) \tag{369}$$

- Essentially, we use random exploration to find a direction of decrease, as opposed to using second order information (Hessian)
- Related to Langevin Dynamics

Theorem

Assuming L-first order and ρ-second order smoothness, perturbed GD can find an $(\epsilon, \sqrt{\epsilon})$ solution with high probability when $T = \Omega(d^2 \epsilon^{-1})$.

- Recall CRNM needs $T = \Omega(\epsilon^{-0.75})$ deterministically and w/o requiring L-first order smoothness
- The issue is adding noise in every single iteration, even if we are not near a saddle (seems unnecessary)

Better Perturbed GD

- Only add noise when needed (near a first-order solution that has a low gradient norm)
- for any t if $\|\nabla f(w_t)\| \leq \kappa$
  - $w_t \leftarrow w_t + e$ with $e_t \sim B(0, r)$ with $r = \mathcal{O}(\epsilon)$
- Do GD for $\tilde{T}$ iterations

Theorem

Assuming L-first order and ρ-second order smoothness, better perturbed GD can find an $(\epsilon, \sqrt{\epsilon})$ solution with high probability when $T = \tilde{\Omega}(\epsilon^{-1})$.

- Finding 2nd-order solutions is almost as easy as finding first-order solutions (when ignoring log terms that may depend on dimensions)

Stochastic Lower Bounds

Motivation

- Recall our canonical training task

$$\min_{w} f_p(w) := \mathbb{E}_{z\sim p}[\ell(w, z)] \tag{370}$$

  for which we have discussed several efficient training methods, e.g., SGD and its variations
- Notably, we have discussed several upper bounds on the performance of these methods
- Does not tell us the fundamental limit of performance: best achievable by any method using stochastic gradients
- We will discuss tools to derive such results for stochastic convex optimization.
- Setup:
  - p is unknown, ℓ is convex and G-Lipschitz for all z
  - We observe stochastic gradients $g_1, \ldots, g_T$
  - We produce $\hat{w}$ as a function of $g_1, \ldots, g_T$

Minimax Optimality

- Recall, for SGD, from stochastic gradients $g_1, \ldots, g_T$ generated according to samples $z_1, \ldots, z_T \sim p$ we upper bounded

$$\mathbb{E}_{z_1,\ldots,z_T}[f_p(w) - \min_w f_p(w)] \equiv \mathbb{E}_{\mathbb{P}^T}[f_p(w) - \min_w f_p(w)] \leq \frac{\|w_1 - w^*\|(G+\sigma)}{\sqrt{T}} \tag{371}$$

  where $\mathbb{P}^T$ is the joint distribution of observations $g_1, \ldots, g_T$
- How good is the suboptimality gap $\mathbb{E}_{\mathbb{P}^T}[f_p(w) - \min_w f_p(w)]$? depends on p which is unknown.
- The above notion of "risk" cannot be minimized since it depends on the unknown parameter p. Thus, to talk about the optimality, in general additional criteria are required
- One such criteria is minimax risk: achieves the smallest maximum risk among all alternative distributions belonging to some distribution set D
- Let $\mathbb{P}^T$ be the joint distribution of observations $g_1, \ldots, g_T$:

$$\mathcal{R}_T^D(w) = \max_{p \in D} \ \mathbb{E}_{\mathbb{P}^T}[f_p(w) - \min_w f_p(w)], \qquad w_{opt} = \arg\min_w \mathcal{R}_T^D(w) \tag{372}$$

Procedure to Establish the Lower Bound

- Our main strategy will involve lower bounding the risk step-by-step and constructing a hard example
- We reduce the optimization for this hard example to a statistical decision making (hypothesis test) problem
- We use tools from information theory (e.g. Le Cam's, Fano's, and Assoud's methods) to study the probability of failing the test
- We assume w.l.o.g. D is a finite set $D = \{p_1, p_{-1}\}$ and let π be a distribution over D such that $\pi(p_1) = \pi(p_{-1}) = 0.5$. Then, since max is larger than average

$$\begin{aligned}\mathcal{R}_T^D(w) &= \max_{p_i \in D} \ \mathbb{E}_{\mathbb{P}_i^T}[f_{p_i}(w) - \min_w f_{p_i}(w)] \geq \mathbb{E}_{p_i \sim \pi}\left[\mathbb{E}_{\mathbb{P}_i^T}[f_{p_i}(w) - \min_w f_{p_i}(w)]\right] \\ &= \pi(p_1)\mathbb{E}_{\mathbb{P}_{\mathbf{1}}^T}[f_{p_{\mathbf{1}}}(w) - \min_w f_{p_{\mathbf{1}}}(w)] + \pi(p_{-1})\mathbb{E}_{\mathbb{P}_{-\mathbf{1}}^T}[f_{p_{-\mathbf{1}}}(w) - \min_w f_{p_{-\mathbf{1}}}(w)]\end{aligned} \tag{373}$$

where $\mathbb{P}_i^T$ is the joint distribution of observations (stochastic gradients) when $z \sim p_i$

Binary Hypothesis Testing

- We will construct a test based on the optimization performance $f_{p_i}(w) - \min_w f_{p_i}(w)$
- In hypothesis testing, the goal is to determine which of the two alternative is correct based on a number of observations.
- Nature chooses a distribution $p_i \in D$ and as a result a loss function $f_{p_i}(w)$ (in our case $D = \{p_1, p_{-1}\}$)
- Based on this choice, we observe stochastic gradients $g_1, \ldots, g_T$ for $f_{p_i}(w)$ with joint distribution $\mathbb{P}_i^T$
- Goal is to correctly identify the choice made by nature, i.e., from $g_1, \ldots, g_T$ decide which distribution/loss was chosen.
- Typically, the probability of making mistake is lower bounded as the distributions, despite being different, are similar, so the observations could lead to a mistake
- The implication for us will be a lower bound on the convergence rate

Construction of Loss functions

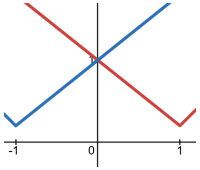
- Let $\ell(w, z) = G|w - zD|$ for $z \in \{1, -1\}$. Note the function is convex and G-Lipschitz. Consider two binary distribution on z

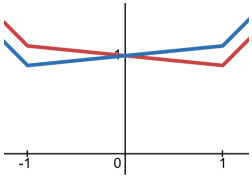
$$p_1(z = i) = \frac{1 + i\delta}{2}, \qquad p_{-1}(z = i) = \frac{1 - i\delta}{2}, \qquad i \in \{1, -1\} \tag{374}$$

giving rise to the two functions

$$f_{p_i}(w) = \mathbb{E}_{z \sim p_i}[\ell(w, z)] = \frac{1 + i\delta}{2} G|w - D| + \frac{1 - i\delta}{2} G|w + D|, \qquad i \in \{1, -1\} \tag{375}$$

- Both functions are convex and G-Lipschitz and also $w_i^* = iD$ implying $|w_i^*| = D$
- Functions becomes harder to distinguish as δ gets smaller





Construction of Loss functions (Cont'd)

- We consider the natural choice $g_t = G \cdot \text{sign}(w_t - z_t D)$ for the stochastic gradients by noting $\text{sign}(w - D)$ is a (sub)gradient of $|w - D|$
- It is evident that g_t is an unbiased estimator of $\nabla f_i(w_t)$ under $z_t \sim p_i$ and $\text{var}(g_t) \leq \mathbb{E}\|g_t\|^2 \leq G^2$
- The idea is that since the functions are so similar for small δ, by observing g_t we might mistakenly think they are the stochastic gradient for the other function.
- Despite the similarity of functions and stochastic gradients, the functions are separated in the sense that if

$$f_{p_i}(\hat{w}) \leq f_{p_i}(w_i^*) + 2GD\delta \quad \Rightarrow \quad f_{p_{-i}}(\hat{w}) \leq f_{p_{-i}}(w_{-i}^*) + 2GD\delta, \qquad i \in \{1, -1\} \tag{376}$$

- We will use this separation property to reduce optimization to statistical test

- We observe the sequence of stochastic gradients and use them in our favorite algorithm and we output p_i as the candidate for the initial choice of nature if $f_{p_i}(\hat{w}) \leq f_{p_i}(w_i^*) + 2GD\delta$
- Let us denote this test by $\text{Te}(g_1, \ldots, g_T) = i$
- Recall we showed

$$\mathcal{R}_T^D(w) = \max_{p_i \in D} \ \mathbb{E}_{\mathbb{P}_i^T}[f_{p_i}(w) - \min_w f_{p_i}(w)] \geq \mathbb{E}_{p_i \sim \pi}\left[\mathbb{E}_{\mathbb{P}_i^T}[f_{p_i}(w) - \min_w f_{p_i}(w)]\right] \tag{377}$$

- By Markov inequality, i.e. $\Pr(X \geq \lambda) \leq \mathbb{E}[X]\lambda^{-1}$ for $X, \lambda > 0$

$$\mathcal{R}_T^D(w) \geq \mathbb{E}_{p_i \sim \pi}\left[\mathbb{P}_i^T(f_{p_i}(w) - \min_w f_{p_i}(w) \geq 2GD\delta)\right] \cdot 2GD\delta \tag{378}$$

- Note $\mathbb{P}_i^T(f_{p_i}(w) - \min_w f_{p_i}(w) \geq 2GD\delta)$ is precisely the probability of making an error (failing the test) via our test $\mathrm{Te}(g_1, \ldots, g_T)$ when $g_1, \ldots, g_T$ are generated according to p_i.
- Total probability of error:

$$\begin{aligned}\mathbb{Q}(\mathrm{Te}(g_1, \ldots, g_T) \neq i) &= \pi(p_1) \cdot \mathbb{P}_1^T(\mathrm{Te}(g_1, \ldots, g_T) \neq 1) \\ &\quad + \pi(p_{-1}) \cdot \mathbb{P}_{-1}^T(\mathrm{Te}(g_1, \ldots, g_T) \neq -1) \\ &= \mathbb{E}_{p_i \sim \pi}\left[\mathbb{P}_i^T(\mathrm{Te}(g_1, \ldots, g_T) \neq i)\right]\end{aligned} \tag{379}$$

  where $\mathbb{Q}$ is the joint distribution over the random index $i \in \{-1, 1\}$ and the observed gradients $g_1, \ldots, g_T$
- Therefore, we have effectively shown

$$\mathcal{R}_T^D(w) \geq \mathbb{Q}(\mathrm{Te}(g_1, \ldots, g_T) \neq i) \cdot 2GD\delta \geq \min_{\tilde{\mathrm{Te}} \in \text{ set of all Tests}} \mathbb{Q}(\tilde{\mathrm{Te}}(g_1, \ldots, g_T) \neq i) \cdot 2GD\delta \tag{380}$$

Bounding the Error Probability

- Recall we have set $\pi(p_1) = \pi(p_{-1}) = 0.5$.
- Also note $\mathbb{P}_i^T$ are different probability measures defined on the same sample space
- Further the events corresponding to $\tilde{\text{Te}}(g_1, \ldots, g_T) \neq 1$ and $\tilde{\text{Te}}(g_1, \ldots, g_T) \neq -1$ are complement: any testing procedure maps one region of the sample space A, to 1 and the complement (A^c) to -1. Thus,

$$
\begin{aligned}
&\min_{\tilde{\text{Te}} \in \text{ set of all Tests}} \mathbb{Q}(\tilde{\text{Te}}(g_1, \ldots, g_T) \neq i) \\
&= \frac{1}{2} \min_{\tilde{\text{Te}} \in \text{ set of all Tests}} \left(\mathbb{P}_1^T(\tilde{\text{Te}}(g_1, \ldots, g_T) \neq 1) + \mathbb{P}_{-1}^T(\tilde{\text{Te}}(g_1, \ldots, g_T) \neq -1) \right) \\
&\equiv \min_A \frac{1}{2} \left(\mathbb{P}_1^T(A^c) + \mathbb{P}_{-1}^T(A) \right) \equiv \min_A \frac{1}{2} \left(1 - \mathbb{P}_1^T(A) + \mathbb{P}_{-1}^T(A) \right) \\
&\equiv \frac{1}{2} \left(1 - \max_A \{ \mathbb{P}_1^T(A) - \mathbb{P}_{-1}^T(A) \} \right) \equiv \frac{1}{2} (1 - \| \mathbb{P}_1^T - \mathbb{P}_{-1}^T \|_{TV})
\end{aligned}
\tag{381}
$$

Bounding the Minimax Rate with KL

- Thus, we have established a lower bound on the risk in terms of the separation of our functions and the joint distributions:

$$\mathcal{R}_T^D(w) \geq GD\delta(1 - \|\mathbb{P}_1^T - \mathbb{P}_{-1}^T\|_{TV}) \tag{382}$$

- We then typically upper bound the TV distance with KL divergence via

$$\text{Pinsker:} \quad \|\mathbb{P}_1^T - \mathbb{P}_{-1}^T\|_{TV} \leq \sqrt{\frac{1}{2} D_{kl}(\mathbb{P}_1^T \| \mathbb{P}_{-1}^T)} \tag{383}$$

$$\text{Bertagnolle-Huber:} \quad \|\mathbb{P}_1^T - \mathbb{P}_{-1}^T\|_{TV} \leq \sqrt{1 - \exp(-D_{kl}(\mathbb{P}_1^T \| \mathbb{P}_{-1}^T))} \tag{384}$$

Tensorization of KL Lemma

Let $\mathbb{P}_i^{[t]}$ denote the conditional distribution of g_t given $g_1, \ldots, g_{t-1}$ when $z \sim p_i$. Then,

$$D_{kl}(\mathbb{P}_1^T \| \mathbb{P}_{-1}^T) = \sum_{t=1}^{T} \mathbb{E}_{\mathbb{P}_1^{t-1}}[D_{kl}(\mathbb{P}_1^{[t]} \| \mathbb{P}_{-1}^{[t]})] \leq T \max_{t=1,\ldots,T} D_{kl}(\mathbb{P}_1^{[t]} \| \mathbb{P}_{-1}^{[t]}) \tag{385}$$

Bounding the Minimax Rate with KL (Cont'd)

- Thus, using KL tensorization our lower bound reduces to

$$\mathcal{R}_T^D(w) \geq GD\delta(1 - \sqrt{0.5T \max_{t=1,\ldots,T} D_{kl}(\mathbb{P}_1^{[t]} \| \mathbb{P}_{-1}^{[t]})}) \tag{386}$$

- Note that for any t, $\mathbb{P}_i^{[t]}$ is basically a Bernoulli distribution:

$$\mathbb{P}_1^{[t]}(z = i) = \frac{1 + i\delta}{2}, \qquad \mathbb{P}_{-1}^{[t]}(z = i) = \frac{1 - i\delta}{2}, \qquad i \in \{-1, 1\} \tag{387}$$

- So, we need to just calculate the KL between two slightly different Bernoulli distributions:

$$D_{kl}(\mathbb{P}_1^{[t]} \| \mathbb{P}_{-1}^{[t]}) = \delta \log \frac{1 + \delta}{1 - \delta} \leq c_1 \delta^2 \tag{388}$$

for some $c > 1$ and $\delta < 1$ (using Taylor approximation of log). Thus,

$$\mathcal{R}_T^D(w) \geq GD\delta(1 - c_2\delta\sqrt{T}) \geq c_3 \frac{DG}{\sqrt{T}} \tag{389}$$

by setting $\delta = c_4/\sqrt{T}$ for some $c_2, c_3, c_4 > 0$

Overparameterization and Interpolation

Overparameterization of DL models

- Modern DL models are overparameterized, that is, if $\mathrm{w} \in \mathbb{R}^d$ denotes the model parameters and n denotes the number of training data points, then $d > n$
- Thus, we have more unknowns than knowns and consequently multiple global minima for our training task

$$\min_w f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) \tag{390}$$

- Effectively, there are parameters w^* that can interpolate all samples

$$w^* = \arg\min_w f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i) \qquad \Leftrightarrow \qquad w^* = \arg\min_w \ell(w, z_i), \quad \forall i \in \{1, \ldots, n\} \tag{391}$$

- Equivalently, $\nabla f(w^*) = \nabla \ell(w^*, z_i) = 0$ which we call the interpolation condition
- We will discuss the impact of overparameterization and interpolation on convergence and generalization of DL models.

- Recall, for SGD we showed
$$\mathbb{E}\|\nabla f(\hat{w})\|^2 \lesssim \frac{1}{T} + \frac{\sigma^2}{\sqrt{T}} \tag{392}$$
where σ^2 was the variance of the SFO.
- Thanks to interpolation, the variance is not constant, rather decreases as SGD makes progress towards the optimal solution (free variance reduction)

Proposition

Consider $f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i)$ and assume each ℓ is L-smooth. Furthermore assume the interpolation condition holds. Then,
$$\mathbb{E}_{i\sim U[1,\ldots,n]}\|\nabla \ell(w, z_i)\|^2 \leq 2L(f(w) - f^*). \tag{393}$$
That is, the variance of stochastic gradient is bounded by the current suboptimality gap.

- Using this result, one could easily show $\mathbb{E}\|\nabla f(\hat{w})\|^2 \lesssim \frac{1}{T}$.

Proof of Proposition

- Let us calculate the second moment

$$\mathbb{E}_{i\sim U[1,\ldots,n]}\|\nabla\ell(w,z_i)\|^2=\frac{1}{n}\sum_{i=1}^{n}\|\nabla\ell(w,z_i)\|^2 \tag{394}$$

- A consequence of L-smoothness is that if $w_i^*=\arg\min_w \ell(w,z_i)$

$$\|\nabla\ell(w,z_i)\|^2\leq 2L(\ell(w,z_i)-\ell(w_i^*,z_i)) \tag{395}$$

- Thus over all

$$\mathbb{E}_{i\sim U[1,\ldots,n]}\|\nabla\ell(w,z_i)\|^2\leq 2L\frac{1}{n}\sum_{i=1}^{n}(\ell(w,z_i)-\ell(w_i^*,z_i)) \tag{396}$$

- Using interpolation condition we have $w_i^*=w^*$ for all i. thus, proof is complete by noting the definition of f.

Consequence of Smoothness

- Recall smoothness means

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2, \qquad \forall x, y \in \mathbb{R}^d \tag{397}$$

- Notably, the condition holds for

$$y_x^* = \operatorname*{argmin}_y f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \tag{398}$$

- A strongly convex problem. Taking the gradient and setting it to zero

$$y_x^* = x - \frac{1}{L}\nabla f(x) \tag{399}$$

- Substituting in smoothness definition and noting $f(y_x^*) \geq f^* = \min_x f(x)$ establishes

$$\|\nabla f(x)\|^2 \leq 2L(f(x) - f^*) \tag{400}$$

Faster Convergence to Better Solutions

- We showed thanks to interpolation (a property of overparameterized models), SGD-based method enjoy free variance reduction and converge faster to stationary solutions
- Next, we explore a condition that enables faster convergence to global minima.
- The idea is that the loss landscape along the trajectory of SGD-based methods is typically very nice such that these methods only converge to global solutions.
- Notably, along the trajectory, in addition to smoothness, gradient domination or the Polyak-Lojasiewicz (PL) Condition holds:

$$\|\nabla f(w)\|^2 \geq 2\mu(f(w) - f^*), \qquad \mu > 0 \tag{401}$$

- States all stationary solutions are also global solutions (despite nonconvexity).
- Along with the consequence of smoothness, i.e. $\|\nabla f(w)\|^2 \leq 2L(f(w) - f^*)$ leads to faster convergence to global minima

PL condition for Overparameterized Models

- Assume a learning task such that with n data points $z_i = (x_i, y_i)$ where $y_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^p$.
- Consider a parametric DL model $h(w, x)$. Thus, a simple learning problem is

$$\min_w f(w) \quad \equiv \quad \min_w \frac{1}{2n} \sum_{i=1}^{n} \Big(y_i - h(w, x_i) \Big)^2 \quad \equiv \quad \min_w \frac{1}{2n} \|\mathbf{y} - \mathbf{h}(w, \mathbf{X})\|^2 \tag{402}$$

by defining

$$\mathbf{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n, \qquad \mathbf{h}(w, \mathbf{X}) = [h(w, x_1), \ldots, h(w, x_n)]^\top \in \mathbb{R}^n \tag{403}$$

$$\mathbf{X} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^{n \times p} \tag{404}$$

and noting the definition of Euclidean norm

PL condition for Overparameterized Models (Cont'd)

- Let us calculate the gradient of the loss

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \nabla h(w, x_i))\nabla h(w, x_i) = \frac{1}{n}\nabla \mathbf{h}(w, \mathbf{X}) \cdot (\mathbf{y} - \mathbf{h}(w, \mathbf{X})) \tag{405}$$

  where $\nabla \mathbf{h} \in \mathbb{R}^{d \times n}$ is the gradient of $\mathbf{h}$ i.e., the transpose of its Jacobian.

- Let us calculate the norm square next:

$$\|\nabla f(w)\|^2 = \langle \nabla f(w), \nabla f(w) \rangle = \frac{1}{n}(\mathbf{y} - \mathbf{h}(w, \mathbf{X}))^\top \frac{\nabla \mathbf{h}(w, \mathbf{X})^\top \nabla \mathbf{h}(w, \mathbf{X})}{n}(\mathbf{y} - \mathbf{h}(w, \mathbf{X})) \tag{406}$$

- The matrix $K(w, \mathbf{X}) := \frac{\nabla \mathbf{h}(w, \mathbf{X})^\top \nabla \mathbf{h}(w, \mathbf{X})}{n} \in \mathbb{R}^{n \times n}$ is called the tangent kernel and it is a PSD matrix. Called a kernel since ij entry depends on x_i and x_j

PL condition for Overparameterized Models (Cont'd)

- Let us assume

$$\lambda_{\min}\Big(K(w,\mathbf{X})\Big) = \lambda_{\min}\Big(\frac{\nabla \mathbf{h}(w,\mathbf{X})^\top \nabla \mathbf{h}(w,\mathbf{X})}{n}\Big) = \mu > 0 \tag{407}$$

- Note a necessary condition for this is $d \geq n$, i.e., overparameterization of our DL model.
- Then,

$$\|\nabla f(w)\|^2 \geq \frac{\mu}{n}(\mathbf{y}-\mathbf{h}(w,\mathbf{X}))^\top(\mathbf{y}-\mathbf{h}(w,\mathbf{X})) = 2\mu\cdot\frac{1}{2n}\|\mathbf{y}-\mathbf{h}(w,\mathbf{X})\|^2 = 2\mu f(w) \geq 2\mu(f(w)-f^*) \tag{408}$$

  by definition of f and the fact that $f(w) \geq 0$ for all w.
- This implies if tangent kernel is well conditioned along the trajectory, PL holds and SGD-based method converge very fast to global minima.
- Random initialization can ensure with high probability $\lambda_{\min}\Big(K(w_1,\mathbf{X})\Big) > 0$.
- Thus if the optimization happens near the initialization (e.g., small η and/or L), PL holds for all w_t's with high probability.

Overparameterization and Generalization

- So far we showed the benefits of overparameterization and interpolation in terms of accelerating the training convergence.
- However, overparameterization can further aid with generalization: performing well on test/unseen data
- The main idea is that the training problem of overparameterized model enjoys implicit regularization when solved by SGD-based methods
- Highlighting the crucial role of the methods we talked about, not just in terms of training but also test performance.
- We will explore this idea for linear models $h(w, x_i) = w^\top x_i$ where $d = p > n$. Our task is:

$$\min_w f(w) \quad \equiv \quad \min_w \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - w^\top x_i \right)^2 \quad \equiv \quad \min_w \frac{1}{2n} \|\mathbf{y} - \mathbf{X}w\|^2 \tag{409}$$

- In this case, $K(w, \mathbf{X}) = \mathbf{X}\mathbf{X}^\top$ is independent of model parameters.

Implicit Regularization in Linear Models

- As the system is underdetermined (due to overparameterization), we can perfectly fit all data points such that $f^* = 0$
- Assuming the tangent kernel is positive definite, PL holds for w. Furthermore, the loss is smooth, implying using (S)GD we can find an optimal solution:

$$\lim_{t\to\infty} f(w_t) = 0, \qquad w_{t+1} = w_t - \eta\nabla f(w_t), \quad \eta \leq 1/L \tag{410}$$

- Thus, if $\hat{w} = \lim_{t\to\infty} w_t$, we have that $\mathbf{y} = \mathbf{X}\hat{w}$ exactly
- We will then show among all possible solutions w^*, GD's solution $\hat{w}$ is the one closest to the initialization w_1, i.e.,

$$\|w_1 - \hat{w}\|^2 \leq \|w^* - w_1\|^2, \qquad \forall w^* \in \arg\min_w f(w), \qquad w^* \neq \hat{w} \tag{411}$$

- So, implicitly, GD is solving an ℓ_2 regularized problem:

$$\min_w \|w - w_1\|^2 \quad s.t. \quad \mathbf{y} = \mathbf{X}w \qquad \equiv \qquad \min_w f(w) + \frac{\lambda}{2}\|w - w_1\|^2 \tag{412}$$

Some Linear Algebra Overview

- Range: The range of a matrix $\mathbf{X}$ is the set of all vector a that can be written as a linear combination of columns of $\mathbf{X}$. Notably, any vector $a = \mathbf{X}u$ is in the range of $\mathbf{X}$.
- Null-space: the null-space of a matrix $\mathbf{X}$ is the set of all vectors b such that $\mathbf{X}b = 0$.
- For a matrix $\mathbf{X}$ we have that Range($\mathbf{X}^\top$) and Null($\mathbf{X}$) are orthogonal:

$$b \in \text{Null}(\mathbf{X}) \quad \Leftrightarrow \quad \text{Proj}_{\text{Range}(\mathbf{X}^\top)}(b) = 0, \qquad a \in \text{Range}(\mathbf{X}^\top) \quad \Leftrightarrow \quad \text{Proj}_{\text{Null}(\mathbf{X})}(a) = 0$$

- For our problem where $\mathbf{y} = \mathbf{X}w$, as $d > n$, $\mathbf{X}$ is a fat matrix and has a non-trivial null-space.
- The vector $\bar{w} = \mathbf{X}^\dagger \mathbf{y}$ is in the range of $\mathbf{X}^\top$. $\mathbf{X}^\dagger = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}$ is the pseudo-inverse of $\mathbf{X}$
- The gradient $\nabla f(w) = \frac{1}{n}\mathbf{X}^\top(\mathbf{X}w - \mathbf{y})$ is in the range of $\mathbf{X}^\top$.
- The set of all solutions is

$$S = \{w^* | w^* = \bar{w} + b,\ b \in \text{Null}(\mathbf{X})\} \tag{413}$$

and notably $\text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w^*) = \bar{w}$ for all w^*

Implicit Regularization in Linear Models (Cont'd)

- Given the update $w_{t+1} = w_t - \eta \nabla f(w_t)$ and the fact that $\nabla f(w_t) \in \text{Range}(\mathbf{X}^\top)$ we can show

$$\text{Proj}_{\text{Null}(\mathbf{X})}(\hat{w}) = \text{Proj}_{\text{Null}(\mathbf{X})}(w_t) = \text{Proj}_{\text{Null}(\mathbf{X})}(w_1), \qquad \hat{w} = \lim_{t \to \infty} w_t \tag{414}$$

- Since $\text{Range}(\mathbf{X}^\top)$ and $\text{Null}(\mathbf{X})$ are orthogonal, for any $w^* = \bar{w} + b \neq \hat{w}$

$$\begin{aligned}
&\|w^* - w_1\|^2 = \|\text{Proj}_{\text{Null}(\mathbf{X})}(w^*) - \text{Proj}_{\text{Null}(\mathbf{X})}(w_1)\|^2 + \|\text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w^*) - \text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w_1)\|^2 \\
&\geq \|\text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w^*) - \text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w_1)\|^2 \\
&= \|\text{Proj}_{\text{Range}(\mathbf{X}^\top)}(\hat{w}) - \text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w_1)\|^2 \\
&\overset{(b)}{=} \|\text{Proj}_{\text{Range}(\mathbf{X}^\top)}(\hat{w}) - \text{Proj}_{\text{Range}(\mathbf{X}^\top)}(w_1)\|^2 + \|\text{Proj}_{\text{Null}(\mathbf{X})}(\hat{w}) - \text{Proj}_{\text{Null}(\mathbf{X})}(w_1)\|^2 \\
&= \|\hat{w} - w_1\|^2
\end{aligned} \tag{415}$$

- Orange term in (b) is zero due to eq. (21)

Implication for Deeper Models

- Consider again the general task

$$\min_w f(w) \quad \equiv \quad \min_w \frac{1}{2n} \sum_{i=1}^{n} \Big(y_i - h(w, x_i)\Big)^2 \quad \equiv \quad \min_w \frac{1}{2n} \|\mathbf{y} - \mathbf{h}(w, \mathbf{X})\|^2 \tag{416}$$

- To apply our previous results, one typically linearizes the model around the initialization w_1 to get a linear model:

$$\mathbf{h}^l(w, \mathbf{X}) = \mathbf{h}(w_1, \mathbf{X}) + \nabla \mathbf{h}(w_1, \mathbf{X})^\top (w - w_1) \tag{417}$$

- This leads to a linear regression task

$$\min_w \frac{1}{n} \|\tilde{\mathbf{y}} - \nabla \mathbf{h}(w_1, \mathbf{X})^\top (w - w_1)\|^2, \qquad \tilde{\mathbf{y}} = \mathbf{y} - \mathbf{h}(w_1, \mathbf{X}) \in \mathbb{R}^n \tag{418}$$

- Our results then hold for the linearized model. And thanks to overparameterization and using SGD-based methods with small learning rates, the approximation error is small (e.g. $\mathcal{O}(n/\sqrt{d})$).

Min-max optimization and GANs

From Supervised Learning to Generative Modeling

- Recall so far we talked about training tasks of the form

$$\min f(w) = \mathbb{E}_{z\sim q}[\ell(w, z)] \tag{419}$$

  with the hope of finding a model that best fits the training data
- In supervised learning, the implication is that our parametric model $A(\cdot, w)$ learns the unknown rule between features and labels $y_i = A^*(x_i)$ such that $\|A^*(\cdot) - A(\cdot, w)\|$ is minimized
- In generative modeling the goal is to learn the data distribution p_z or learn the ability to sample from it.
- If we think of $q(\cdot)$ as the true rule, a natural solution then is to use a parametric distribution $p(\cdot, w)$ such that it approximates the true rule p_z well, i.e. $\|q(\cdot) - p(\cdot, w)\|$
- This is precisely the goal of variational inference:

$$\min_w f(w) = D(q\|p(w)) \tag{420}$$

  for some notion of distance or divergence between distributions, e.g. the KL divergence.

Distances among distributions

- f-divergence:

$$D_f(q\|p) = \mathbb{E}_p[f(\frac{q}{p})] = \int_z p(z) f(\frac{q(z)}{p(z)}) \cdot dz \tag{421}$$

  for f being a convex function. Examples include
  - KL divergence with $f(t) = t \log t$
  - TV distance with $f(t) = \frac{1}{2}|t-1|$
  - Jensen-Shannon Distance (JSD) a symmetrized and smoothed version KL: $D_{JSD}(q\|p) = \frac{1}{2} D_{KL}(p\|q) + \frac{1}{2} D_{KL}(q\|p)$

- a-Wasserstein distance or Kantorovich–Rubinstein metric:

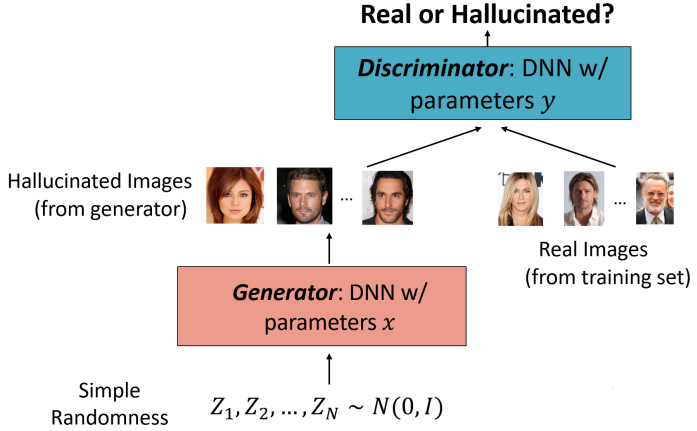
$$W_a(q,p) = \min_{\pi \in C(q,p)} \left(\mathbb{E}_{(z_1,z_2)\sim\pi} \|z_1 - z_2\|^a \right)^{\frac{1}{a}} \tag{422}$$

  for $a \geq 1$ where $C(q,p)$ is the set of all couplings (joint distributions) with marginals q and p.

- Turns out minimizing these is like solving a kind of game

Generative Adversarial Networks (GANs)

- p_w is good estimator of q if a skilled discriminator cannot distinguish $\tilde{z} \sim p_w$ from $z \sim q$
- This suggest building a zero-sum game between two players: one trying to generate fake data (the generator) and other trying to distinguish fake from real data (the discriminator)



GAN Game and Variational Inference

- Let D_y and G_x be the discriminator and generator with parameters y and x, respectively

$$\min_x \max_y U(x, y) := \mathbb{E}_{z\sim p}[\log D_y(z)] + \mathbb{E}_{Z\sim\mathcal{N}(0,I)}[\log(1 - D_y(G_x(Z))] \tag{423}$$

- D_y is a binary classier and assigns high score ($\log D_y$) to real data and low score to fake data.
- G_x want to assign a high score to fake data by maximizing $\log(D_y(G_x(Z))$
- With simple calculation, we can show the best discriminator satisfies:

$$D_{y^*}(z) = \frac{q(z)}{q(z) + p_{G_x}(\cdot)} \tag{424}$$

- Plugging this into the problem above yields the optimal solution for the generator:

$$x^* = \operatorname*{argmin}_x 2D_{JSD}(q\|p_{G_x}) - \log 4 \tag{425}$$

- Thus in equilibrium $D_{y^*}(z) = 0.5$ and $p_{G_{x^*}}(z) = q(z)$
- So a natural question is can we go from a general f-divergence to a game?

- For any convex f define it's Fenchel dual $\tilde{f}$ (which is concave)

$$\tilde{f}(\tilde{t}) = \max_{t} \Big\{ \langle \tilde{t}, t \rangle - f(t) \Big\}, \qquad f(t) = \max_{\tilde{t}} \Big\{ \langle t, \tilde{t} \rangle - \tilde{f}(\tilde{t}) \Big\} \tag{426}$$

- We can use this to lower bound f-divergences:

$$\begin{aligned} D_f(q\|p) &= \int_z p(z) f\Big(\frac{q(z)}{p(z)}\Big) \cdot dz = \int_z p(z) \max_{\tilde{t}} \Big\{ \frac{q(z)}{p(z)} \tilde{t} - \tilde{f}(\tilde{t}) \Big\} \cdot dz \\ &\geq \max_{\tilde{t}} \Big\{ \int_z \Big(p(z) \frac{q(z)}{p(z)} \tilde{t} - \tilde{f}(\tilde{t}) \Big) \cdot dz \Big\} \\ &\geq \max_{\tilde{t} \in \mathcal{T}(z)} \Big\{ \int_z q(z) \tilde{t}(z) \cdot dz - \int_z p(z) \tilde{f}(\tilde{t}(z)) \cdot dz \Big\} \\ &= \max_{\tilde{t}(z) \in \mathcal{T}(z)} \mathbb{E}_{z \sim q}[\tilde{t}(z)] - \mathbb{E}_{z \sim p}[\tilde{f}(\tilde{t}(z))] \end{aligned} \tag{427}$$

where the first inequality is due to Jensen's inequality and convexity of max while the second inequality is by restricting the set of all $\tilde{t}$ to a class $\mathcal{T}$ of functions of z

- Let us change $\tilde{t}(z)$ to $D_y(z)$ implying our function class is parametric with parameters y:

$$D_f(q\|p) \geq \max_y \mathbb{E}_{z\sim q}[D_y(z)] - \mathbb{E}_{z\sim p}[\tilde{f}(D_y(z))] \tag{428}$$

- Recall we think of q as data distribution and p as its estimator. If we assume p is parameterized via p_G for some function $G_x(Z)$ taking random Gaussian $Z \sim \mathcal{N}(0, I)$, fitting q by minimizing $D_f(q\|p)$ amounts to

$$\min_x \max_y \mathbb{E}_{z\sim q}[D_y(z)] - \mathbb{E}_{Z\sim\mathcal{N}(0,I)}[\tilde{f}(D_y(G_x(Z)))] \tag{429}$$

  generalizing GANs from JSD to arbitrary f-divergences.

- The bound is tight for $D_{y^*}(z) = f'(\frac{q(z)}{p(z)})$
- Generator is trying to minimize the divergence estimate, while the discriminator tries to tighten the lower bound.
- How about a-Wasserstein distance? Any benefit?

Wasserstein GAN

- Recall $D_f(q\|p) = \mathbb{E}_p[f(\frac{q}{p})]$ so the support of p has to cover the support of q. That is if $p(z_1) = 0$ we like $q(z_1) = 0$ too
- Otherwise discontinuity arises in f-divergences and they could become ∞ (e.g., KL).
- Turns out a-Wasserstein distances are nicer in such scenarios
- Using Kantorovich-Rubinstein duality, we represent this task as a minmax game. For $a = 1$

$$W_1(q,p) = \min_{\pi \in C(q,p)} \left(\mathbb{E}_{(z_1,z_2)\sim\pi} \|z_1 - z_2\| \right) = \max_{D:\|D\|_{Lip}\leq 1} \mathbb{E}_{z\sim q}[D(z)] - \mathbb{E}_{z\sim p}[D(z)] \tag{430}$$

  where $\|D\|_{Lip} \leq 1$ means D must have a Lipschitz constant at most 1
- This idea leads to Wasserstein GAN (WGAN)

$$\min_x \max_y \mathbb{E}_{z\sim q}[D_y(z)] - \mathbb{E}_{Z\sim\mathcal{N}(0,I)}[D_y(G_x(Z))] \tag{431}$$

- Lipschitzness is ensured by weight clipping or gradient penalty $\|\nabla_y D_y(z)\|$

Difficulties of Minmax Optimization

- Recall we showed for f-GAN training

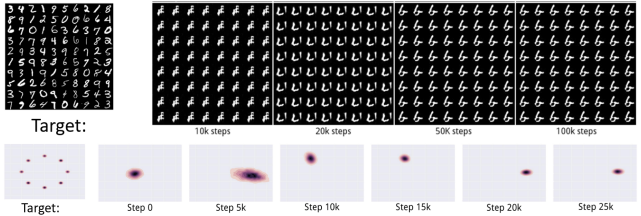
$$\min_x \max_y U(x,y) = \mathbb{E}_{z\sim q}[D_y(z)] - \mathbb{E}_{Z\sim\mathcal{N}(0,I)}[\tilde{f}(D_y(G_x(Z)))] \tag{432}$$

- And for WGAN training

$$\min_x \max_y U(x,y) = \mathbb{E}_{z\sim q}[D_y(z)] - \mathbb{E}_{Z\sim\mathcal{N}(0,I)}[D_y(G_x(Z))] \tag{433}$$

- Both are minmax optimization. One could use minmax version of (S)GD for training

$$x_{t+1} = x_t - \eta\nabla_x U(x_t, y_t), \qquad y_{t+1} = y_t + \eta\nabla_y U(x_t, y_t) \tag{434}$$



Minmax vs min

- The issue can be attributed to the fundamental difference between minmax and min problems.
- For minimization we interpreted GD a simple forward Euler discretization of gradient flow

$$w_{t+1} = w_t - \eta \nabla f(w_t), \qquad \dot{w} = -\nabla f(w) \tag{435}$$

- Let $v = [x, y]^\top$ and define $F(v) = [\nabla_x U(x_t, y_t), -\nabla_y U(x_t, y_t)]^\top$. Minmax GD becomes

$$v_{t+1} = v_t - \eta F(v_t), \qquad \dot{v} = -F(v) \tag{436}$$

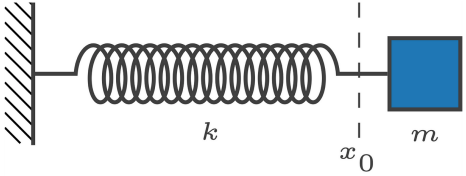
- They look very similar, but there is a fundamental difference! ∇f is the gradient of a scalar function but F is not
- Jacobian of ∇f, $J_{\nabla f}$ is symmetric (the Hessian $\nabla^2 f$) while the Jacobian of F, J_F is not
- The Jacobian controls the local dynamics: how the flow $F(v)$ curves, rotates, contracts, or expands locally.

Dissipative vs. Conservative Systems

- Note $J_F = \frac{J_F + J_F^\top}{2} + \frac{J_F - J_F^\top}{2}$, i.e., sum of symmetric and skew-symmetric matrices
- when the flow is $\dot{w} = -\nabla f(w)$ there is no skew-symmetric component and under mild condition the system is dissipating energy.
- When the flow is $\dot{v} = -F(v)$, the system might conserve energy and the skew-symmetric component determines the rotation.
- The most simple example is harmonic oscillator

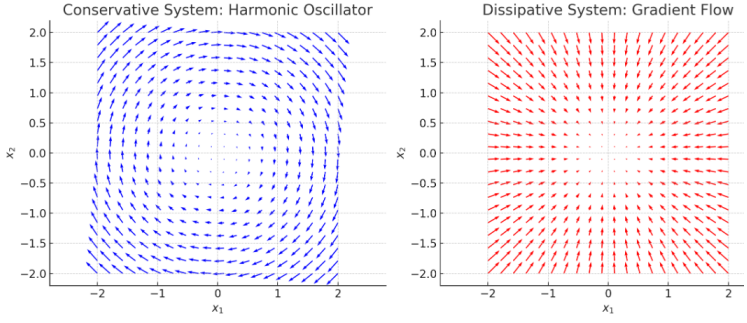
$$\ddot{w} = -w \qquad \equiv \qquad \dot{x} = y, \quad \dot{y} = -x \qquad \equiv \qquad \dot{v} = -F(v) \tag{437}$$

for which the Jacobian is purely skew-symmetric. It corresponds to the simplest minmax problem: $\min_x \max_y \; U(x, y) = xy$



Dissipative vs. Conservative Systems (Cont'd)

- Following the flow might not help in skew-symmetric dynamics
- Simple Euler forward discretization (GD in optimization) converges nicely to a stable point for dissipative systems while does not work well for conservative systems as it pushes us around (and potentially away) the stable solution
- Thus, for minmax problems we generally need other classes of method



Negative Momentum and Predictability

- Recall we talked about momentum for min problems. Intuitively, we assume usefulness of gradient direction and we double down by introducing positive momentum
- For minmax problems, the gradient direction has a mistake and we can correct that by introducing negative momentum.
- This idea leads to the update $v_{t+1} = v_t - \eta F(v_{t+1})$
- This is precisely the Euler Backward discretization of the flow and it is generally implicit.
- Also called proximal point method since if $F \to \nabla f$ for some convex function f

$$w_{t+1} = w_t - \eta \nabla f(w_{t+1}) \quad \equiv \quad w_{t+1} = \arg\min_w f(w) + \frac{1}{2\eta}\|w - w_t\|^2 \tag{438}$$

Mid-Point Discretization and Extra Gradient

- Consider the flow

$$\dot{v} = -F(v) \qquad \equiv \qquad v_{t+1} = v_t - \int_t^{t+1} F(v_\tau) \cdot d\tau \tag{439}$$

- Forward Euler (GD) approximates

$$v_{t+1} = v_t - \int_t^{t+1} F(v_\tau) \cdot d\tau \approx \int_t^{t+1} F(v_t) \cdot d\tau = v_t - \eta F(v_t) \tag{440}$$

- Backward Euler (Proximal Point) approximates

$$v_{t+1} = v_t - \int_t^{t+1} F(v_\tau) \cdot d\tau \approx \int_t^{t+1} F(v_{t+1}) \cdot d\tau = v_t - \eta F(v_{t+1}) \tag{441}$$

Mid-Point Discretization and Extra Gradient (Cont'd)

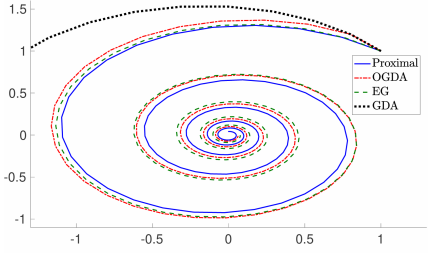
- More generally for any $u \in (t, t+1)$ and in particular for $u = 1/2$ we can do

$$v_{t+1} = v_t - \int_t^{t+1} F(v_\tau) \cdot d\tau \approx \int_t^{t+1} F(v_{t+\frac{1}{2}}) \cdot d\tau = v_t - \eta F(v_{t+\frac{1}{2}}) \tag{442}$$

  and find $v_{t+\frac{1}{2}}$ via Forward Euler to obtain an explicit update

$$v_{t+1} = v_t - \eta F(v_{t+\frac{1}{2}}), \qquad v_{t+\frac{1}{2}} = v_t - \eta F(v_t) \tag{443}$$

- Called Extra Gradient (EG), approximates PP by doing $F(v_{t+1}) \approx F(v_t - \eta F(v_t))$



Optimistic Gradient Descent

- Consider a field which is Lipschitz, meaning the variations are controlled: $F(v_{t+1}) \approx F(v_t)$.
- This yields GD. We hope to get a tighter approximation by doubling down on our optimism towards controlled variation of F
- $F(v_{t+1}) \approx F(v_t)$ is effectively the zeroth-order Taylor approximation. Consider the first-order Taylor approximation instead

$$F(v_{t+1}) \approx F(v_t) + J_F(v_t)(v_{t+1} - v_t), \qquad J_F(v_t) \approx \frac{F(v_t) - F(v_{t-1})}{\|v_t - v_{t-1}\|} \cdot e^\top \tag{444}$$

  where $e = \frac{v_t - v_{t-1}}{\|v_t - v_{t-1}\|}$ is the unit vector along the direction of $v_t - v_{t-1}$
- Let us be optimistic one last time and assume $v_{t+1} - v_t \approx v_t - v_{t-1}$, leading to

$$F(v_{t+1}) \approx F(v_t) + \frac{F(v_t) - F(v_{t-1})}{\|v_t - v_{t-1}\|} \frac{(v_t - v_{t-1})^\top}{\|v_t - v_{t-1}\|}(v_t - v_{t-1}) = F(v_t) + F(v_t) - F(v_{t-1}) \tag{445}$$

- This serves as the update vector in Optimistic GD: $v_{t+1} = v_t - 2\eta F(v_t) + \eta F(v_{t-1})$