CSCE 790: Neural Networks and Their Applications

AIISC and Dept. Computer Science and Engineering
Email: vignar@sc.edu

October 10, 2023

Generalization?

- The out-of-sample error E_{out} measures how well our training on D has generalized to data that we have not seen before
- E_{out} is based on the performance over the entire input space X
- Intuitively, if we want to estimate the value of E_{out} using a sample of data points, these points must be 'fresh' test points that have not been used for training
- The in sample error E_{in}, by contrast, is based on data points that have been used for training

Generalization

- The value of E_{in} does not always generalize to a similar value of E_{out}
- Generalization is a key issue in learning
- One can define the generalization error as the discrepancy between E_{in} and E_{out}
- Universal approximation theorem warrants $E_{in} \rightarrow 0$ as number of neurons in the hidden layers $\rightarrow \infty$
- For more, check: Learning from Data: A Short Course, by Hsuan-Tien Lin, Malik Magdon-Ismail, and Yaser Abu-Mostafa

Convex Set

Definition

A subset $C \subset \mathbb{R}^n$ is a convex set if $\alpha x + (1-\alpha)y \in C$, $\forall x, y \in C$, $\forall \alpha \in [0,1]$.

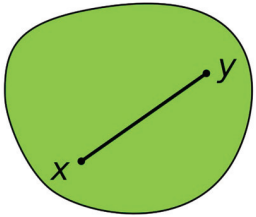


Figure: Convex Set

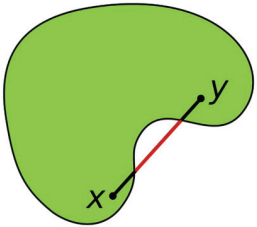


Figure: Non-Convex Set

Figure: (a) Illustration of a convex set which looks somewhat like a deformed circle.

Figure: (b) Illustration of a non-convex set which looks somewhat like a boomerang.

Convex Sets - Example

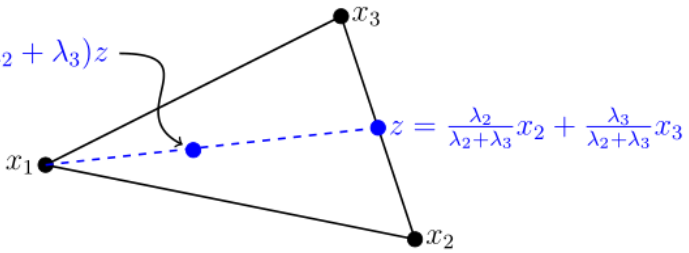
Example (Convex sets)

1. A hyperplane $H = \{x \in \mathbb{R}^n : p^T x = c\}$ for some $p \in \mathbb{R}^n$, $p \neq 0$, and $c \in \mathbb{R}$, for example, a plane in $\mathbb{R}^3$, $p_1 x + p_2 y + p_3 z = 1$.
2. Half space $H^+ = \{x \in \mathbb{R}^n : p^T x \geq c\}$ or $H^- = \{x \in \mathbb{R}^n : p^T x \leq c\}$.

Building Blocks of Hulls

The set of all convex combinations $\sum_{i=1}^{3} \lambda_i x_i$ of $x_1, x_2, x_3 \in \mathbb{R}^n$ with $\lambda_1 + \lambda_2 + \lambda_3 = 1$ is the triangular region determined by x_1, x_2, x_3 (formed between vertices x_1, x_2, x_3).

$$\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 = \lambda_1 x_1 + (\lambda_2 + \lambda_3) z$$



Convex Hull

More generally, the set of all convex combinations $\sum_{i=1}^{k} \lambda_i x_i$ of k vectors $x_1, \ldots, x_k \in \mathbb{R}^n$ is the convex polyhedral region determined by $x_1, \ldots, x_k$ (the so-called convex Hull, the intersection of all convex sets containing x_i, $i = 1, \ldots, k$).

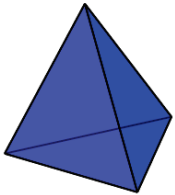


Figure: Illustration of a tetrahedron that is a convex combination of four vectors.

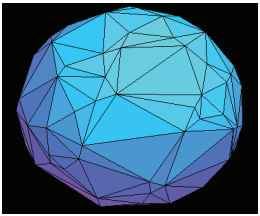
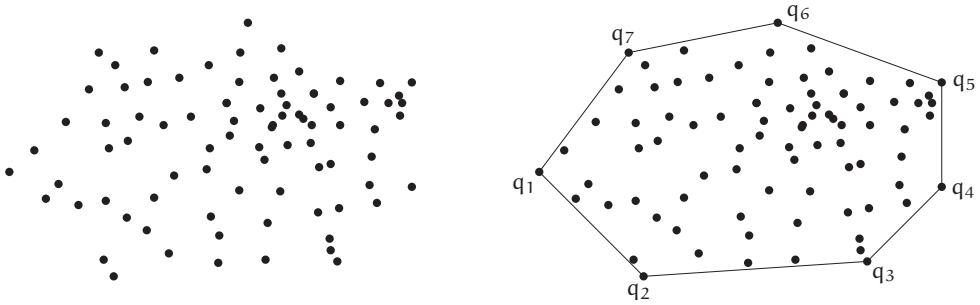


Figure: A convex hull of 100 random uniform points on a sphere.

Convex Hull



(a) Input. (b) Output.

Figure: Convex Hull of a set of points in $\mathbb{R}^2$.

Convex Hull

Given an arbitrary set S in $\mathbb{R}^n$, different convex sets can be generated from S. In particular, we discuss below the convex hull of S.

Definition (Convex hull)

Let S be an arbitrary set in $\mathbb{R}^n$. The convex hull of S, denoted $\text{conv}(S)$, is the collection of all convex combinations of S. In other words, $x \in \text{conv}(S)$ if and only if x can be represented as

$$x = \sum_{j=1}^{k} \lambda_j x_j, \quad \sum_{j=1}^{k} \lambda_j = 1,$$
$$\lambda_j \geq 0 \quad \text{for} \quad j = 1, \ldots, k,$$

where k is a positive integer and $x_1, \ldots, x_k \in S$.

Interpolation

Definition

Interpolation occurs for a sample x whenever this sample belongs to the convex hull of a set of samples $X \triangleq \{x_1, \ldots, x_N\}$, if not, extrapolation occurs.

Theorem

Given a $d-$dimensional dataset $X \triangleq \{x_1, \ldots, x_N\}$ with i.i.d. samples uniformly drawn from an hyperball, the probability that a new sample x is in interpolation regime has the following asymptotic behavior

$$\lim_{d \to \infty} \underbrace{p(x \in Hull(X))}_{Interpolation} = \begin{cases} 1 \iff N > d^{-1} 2^{\frac{d}{2}} \\ 0 \iff N < d^{-1} 2^{\frac{d}{2}} \end{cases} \tag{1}$$

Learning in High Dimension Always Amounts to Extrapolation

Convex Optimization (for completeness)

- Convex cost function
- Constraints represented by convex functions
- Convex constraints $\implies$ constraint set is convex
- $\rightarrow$ Convex optimization problem

Automatic Differentiation

Computational methods for calculating derivatives

- Manual derivation and coding them
- Numerical differentiation using finite difference approximation
- Symbolic differentiation using expression manipulation
- Automatic or Algorithmic differentiation

Automatic Differentiation (AD)

- Manual differentiation
  - Time consuming
  - Prone to error
- Numerical differentiation
  - Prone to round-off and truncation errors
  - Simple to implement
  - Scales poorly for gradients
- Symbolic differentiation
  - Addresses weaknesses of manual and numerical differentiation
  - Often results in complex and cryptic expressions
  - "Expression swell" is a problem
- Manual and symbolic differentiation require closed-form expressions

Autodiff

- AD as a technical term refers to a specific family of techniques that compute derivatives through accumulation of values during code execution to generate numerical derivative evaluations rather than derivative expressions
- Back-propagation algorithm is a specialized version of autodiff
- AD is partly symbolic and partly numerical
  - AD provide numerical values of derivatives (as opposed to derivative expressions) and
  - AD uses symbolic rules of differentiation (but keeps track of derivative values as opposed to the resulting expressions)

Illustration of Backpropagation

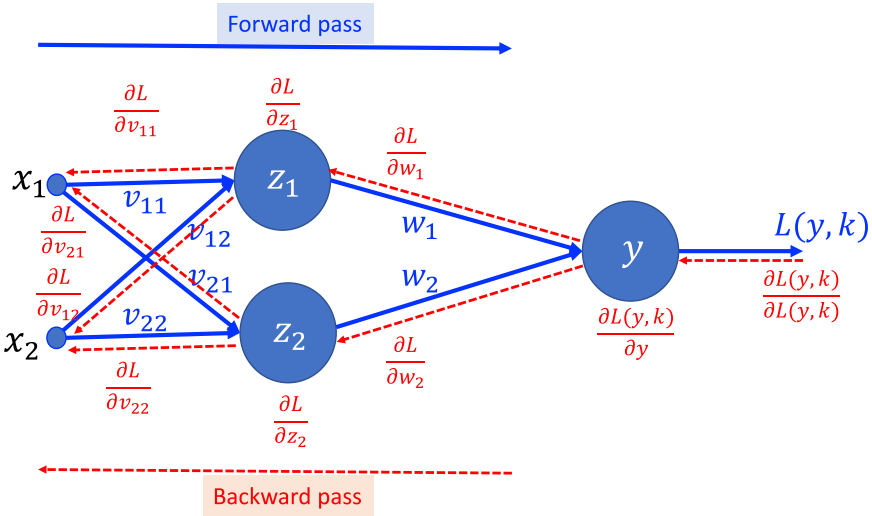


Figure: Illustration of backpropagation

Numerical Differentiation

- Numerical differentiation is the finite difference approximation of derivatives using values of the original function evaluated at some sample points
- Numerical approximations of derivatives are inherently ill-conditioned and unstable

Symbolic Differentiation

- Symbolic differentiation is the automatic manipulation of expressions for obtaining derivative expressions
- Symbolic derivatives do not lend themselves to efficient runtime calculation of derivative values
- Careless symbolic differentiation can easily produce exponentially large symbolic expressions (*expression swell*) - long time to evaluate

Expression Swell

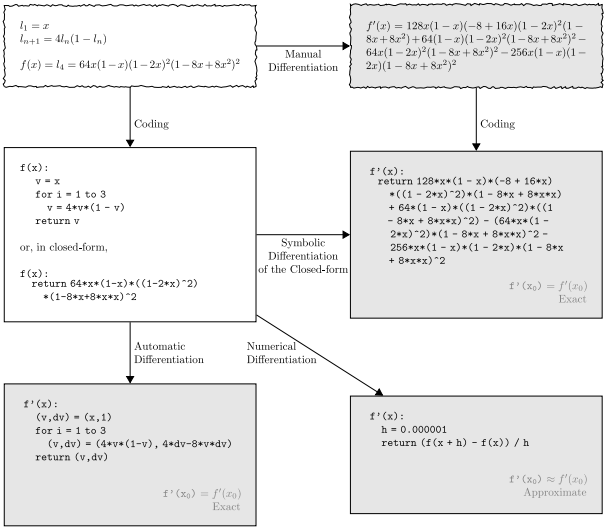
n	l_n	$\frac{d}{dx}l_n$	$\frac{d}{dx}l_n$ (Simplified form)
1	x	1	1
2	$4x(1-x)$	$4(1-x)-4x$	$4-8x$
3	$16x(1-x)(1-2x)^2$	$16(1-x)(1-2x)^2-16x(1-2x)^2-64x(1-x)(1-2x)$	$16(1-10x+24x^2-16x^3)$
4	$64x(1-x)(1-2x)^2(1-8x+8x^2)^2$	$128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+64(1-x)(1-2x)^2(1-8x+8x^2)^2-64x(1-2x)^2(1-8x+8x^2)^2-256x(1-x)(1-2x)(1-8x+8x^2)^2$	$64(1-42x+504x^2-2640x^3+7040x^4-9984x^5+7168x^6-2048x^7)$

Figure: Illustration of expression swell using logistic map $l_{n+1} = 4l_n(1-l_n)$, $l_1 = x$

Numerical-Symbolic Differentiation to AD

- When we are concerned with the accurate numerical evaluation of derivatives and not so much with their actual symbolic form, it is in principle possible to significantly simplify computations by storing only the values of intermediate sub-expressions in memory
- Moreover, for further efficiency, we can interleave as much as possible the differentiation and simplification steps
- This interleaving idea forms the basis of AD

Illustration (Contents related to AD are from Baydin, et al., 2018)



The range of approaches for differentiating mathematical expressions and computer code, looking at the example of a truncated logistic map (upper left). Symbolic differentiation (center right) gives exact results but requires closed-form input and suffers from expression swell; numerical differentiation (lower right) has problems of accuracy due to round-off and truncation errors; automatic differentiation (lower left) is as accurate as symbolic differentiation with only a constant factor of overhead and support for control flow.

Automatic Differentiation

Autodiff - Baydin, et al., 2018