Optimization: A Critical Tool for sustainable AI [2]

Snehanshu Saha

Center for Learning Theory (COLT), Anuradha and Prashanth Palakurthi Centre for Artificial Intelligence Research (APPCAIR), BITS PILANI K K Birla Goa Campus

Papers: https://dblp.org/pid/130/3938.html
ML Blog: https://beginningwithml.wordpress.com/
Github: https://github.com/sahamath?tab=repositories
Youtube Lectures: https://tinyurl.com/yyn6e64q

[2] Dept. of Mathematics Colloquium:: Optimization

Highlights

A Fragmented Timeline

- Pattern Recognition in Small Data (SDPR)
- Big Data (BDPR) : Changed Landscape?
- Porting SDPR-What did we miss?
- The Elegance can't be ignored
- A (mild ?) critique of Deep Learning in Big Data
- Re-emergence of (Methodological) elegance in BDPR

Sustainable AI: cheaper H/W, carbon emission, speed up, native trainability, limited dependency on Cloud HPCs

Big Data Landscape

What's new:

Figure: V^3

Application

Volume, Veracity, Velocity: New Architectures as solutions?

AI in Big Data:

An Example

- Modern astronomical instruments record huge volumes of data in the form of images, catalogs, raw data and signals in different bandwidths.
- A new wave of pursuits in astronomy thus involve the use of **statistics**, **machine learning**, and **artificial intelligence** to solve problems. An emerging **interdisciplinary** area of study which calls for scientists to collaborate from the fields of:
  1. Astronomy and astrophysics
  2. Statistics
  3. Mathematics
  4. Computer and Information sciences.

Mentoring Information:

1. GALEX (The Galaxy Evolution Explorer) 30 TB
2. SDSS (The Sloan Digital Sky Survey) 40 TB
3. LSST (The Large Synoptic Survey Telescope) 200 PB expected

Big Data vs. Small Data

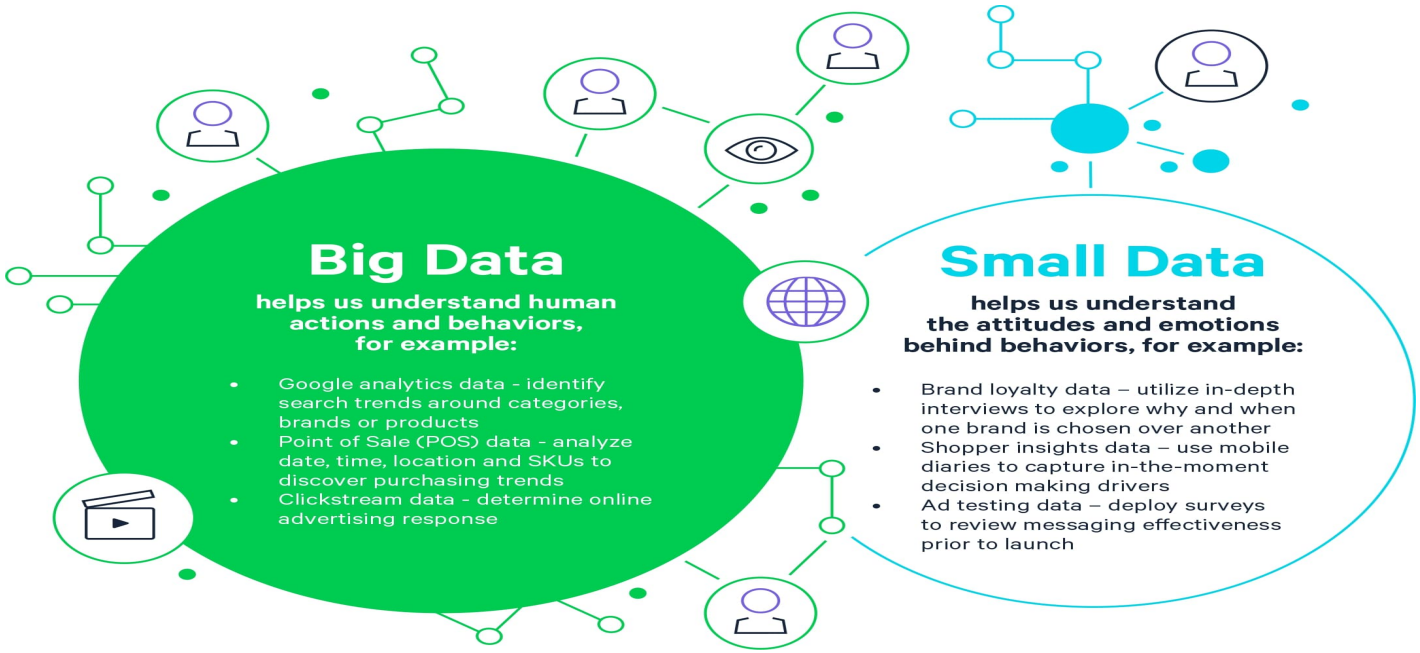


Figure: What Changes?

Are the methods still important? Yes!

- Over-reliance on Architecture necessary? No
- Is the elegance lost? No– LEARN FAST, LEARN DEEP

The Really Deep Neural Nets

Society is about to experience an epidemic of false positives coming out of big-data projects-Prof. Michael Jordan, UC Berkeley

Figure: A certain fool's gold element to our current obsession

Deep NNs leading to false, meaningless inferences...

The cost

Computing Setup

The Really Deep Neural Nets

Master Node 2 x Intel(R) Xeon(R) CPU E5-2620 six-core 48 GB Memory (4GB per core) • 900GB SAS 10K x 4 Compute Node : 10 Qty • 2 x Intel(R) Xeon(R) CPU E5-2650 v2 eight-core • 64 GB Memory (4GB per core) • 600GB SAS 10K x 1 GPU Node : 2 Qty • Nvidia Tesla K20Xm cards • 2 x Intel(R) Xeon(R) CPU E5-2650 v2 eight-core • 128 GB Memory (8GB per core) • 900GB SAS 10K x 4

Neural networks, commonly known as Artificial Neural network(ANN), is a system of interconnected units organized in layers, which processes information signals by responding dynamically to inputs. Layers of the network are oriented in such a way that inputs are fed at input layer and output layer receives output after being processed at neurons of one or more hidden layers.

The Really Deep Neural Nets

Approximate Budget: 75 Lacs!

The good..

The Really Deep Neural Nets

Resnet 18; Densenet 40; Resnet50: The Computer Vision Community
CNN, RNN, Attention Models
LSTM for time series data

- MNIST
- CIFAR10
- CIFAR 100

The Tools

Scala, Hadoop, Julia

Neural nets are extremely intensive computationally and consume a lot of time and resources while training; we tolerate the *"high maintenance kid"* because the accuracy and other performance metrics are amazing, mostly. Plus, it handles non-linear problems reasonably well.

The bad..

Training Time

Since deep neural networks are function approximators, a large corpus of data is usually required to accomplish reasonable approximation of the error function, which is nothing but the difference between target and predicted labels. While computing resources are available in abundance, this problem didn't receive the attention it deserves, until recently.

- MIT Technology review, by Karen Hao
- The process of deep learning has an outsize environmental impact
- The NLP Model

Building and testing a final paper-worthy model in NLP required training 4,789 models over a six-month period. Converted to CO2 equivalent, it emitted more than 78,000 pounds. **the process can emit more than 626,000 pounds of carbon dioxide equivalent–approximately five times the lifetime emissions of the average American car**.

MIT apologizes, permanently pulls offline huge dataset that taught AI systems to use racist, misogynistic slurs!

- LARGE IMAGE DATASETS: A PYRRHIC WIN FOR COMPUTER VISION? Vinay Uday Prabhu & Abeba Birhane, July 1, 2020
- The Word-2-vec, Wordnet: Mothers are homemakers!

Your Big Data ML models are as good as data; We're just processing the volume efficiently!

The dataset holds more than 79,300,000 images, scraped from Google Images, arranged in 75,000-odd categories.

The key problem is that the dataset includes, for example, pictures of Black people and monkeys labeled with the N-word; women in bikinis, or holding their children, labeled w****; parts of the anatomy labeled with crude terms...

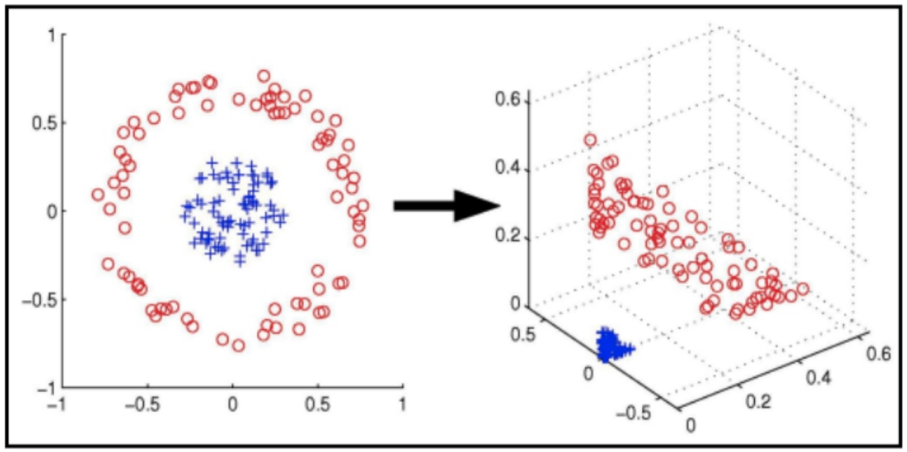
Non-Linearly Separable

what we deal with

Figure: The 2-class classifier: Redshift can't be used as feature–Violet: stars, yellow: quasars and blue: galaxies

False positives; detected by redshift

Kernel Embedding



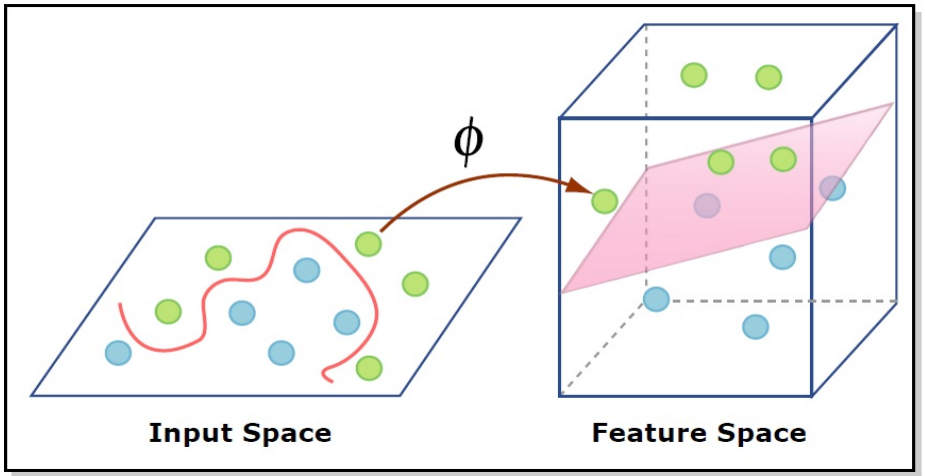


Figure: Non-linear decision boundary embedded to Linear via Mercer Theorem

Figure: A former student of mine training his DL model in NC State

- Optimize, optimize, optimize.....
- New methods in optimization for prediction error to hit rock bottom!
- Get rid of expensive architectures (too many hidden layers, too many neurons, too many weight updates)
- Single-shot learning

Tame the bandits i.e. hyperparameters

What we need

Advanced Calculus, Advanced Statistics, Linear Algebra
Need some geniuses and their theories: Hilbert, Banach, Cauchy, Lipschitz...
Functional Analysis, Convex optimization, Differential Equations, Chaos theory

What are the performance benchmarks we should be looking at

- Accuracy, loss etc.. (obvious)
- Epochs to converge to optima
- CPU/memory utilization
- Carbon footprint

What we should avoid?

- Inferences that make no sense
- Avoid ridiculous mistakes (redshift, Surf Temp estimation)
- ethical/social/racial misdemeanor

Optimization: Revisit the Neural classification

Devil's in the details: lr :: tune the bandits??

Optimise weights and biases by using back propagation on SBAF::
Initialize all weights w_{ij}**, biases** b_i**,** n_epochs**, lr, k and** α**;**

The forward Pass: Use appropriate function approximation

- ODE theory to the rescue.....for k, α

Adapative learning to the rescue.....for **lr**

"lr", the learning rate in weight update, recall?

back propagation & gradient descent: the squared loss function; Gradient Descent update Rule: REVISIT $\mathbf{w} := \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} f$

- Use Lipschitz Constant on the squared loss function
- Use MVT
- **Minimal assumption:** functions are Lipschitz continuous and differentiable up to first order only [a]
- $\alpha = \frac{1}{L}$, we force $\Delta \mathbf{w} \leq 1$, constraining the change in the weights.

[a] Note this is a weaker condition than assuming the gradient of the function being Lipschitz continuous. We exploit merely the boundedness of the gradient.

What is it?

For a function, the Lipschitz constant is the least positive constant L such that $\|f(\mathbf{w}_1) - f(\mathbf{w}_2)\| \leq L \|\mathbf{w}_1 - \mathbf{w}_2\|$

$$\begin{aligned} ||f(\mathbf{w}_1) - f(\mathbf{w}_2)|| &= ||\nabla_{\mathbf{w}} f(\mathbf{v})|| \;\; ||\mathbf{w}_1 - \mathbf{w}_2|| \\ &\leq \sup_{\mathbf{v}} ||\nabla_{\mathbf{w}} f(\mathbf{v})|| \;\; ||\mathbf{w}_1 - \mathbf{w}_2|| \end{aligned}$$

Thus, $\sup_{\mathbf{v}} ||\nabla_{\mathbf{w}} f(\mathbf{v})||$ is such an L. Since L is the least such constant, $L \leq \sup_{\mathbf{v}} ||\nabla_{\mathbf{w}} f(\mathbf{v})||$. We use $\max ||\nabla_{\mathbf{w}} f||$ to derive the Lipschitz constants.

What is it, really?

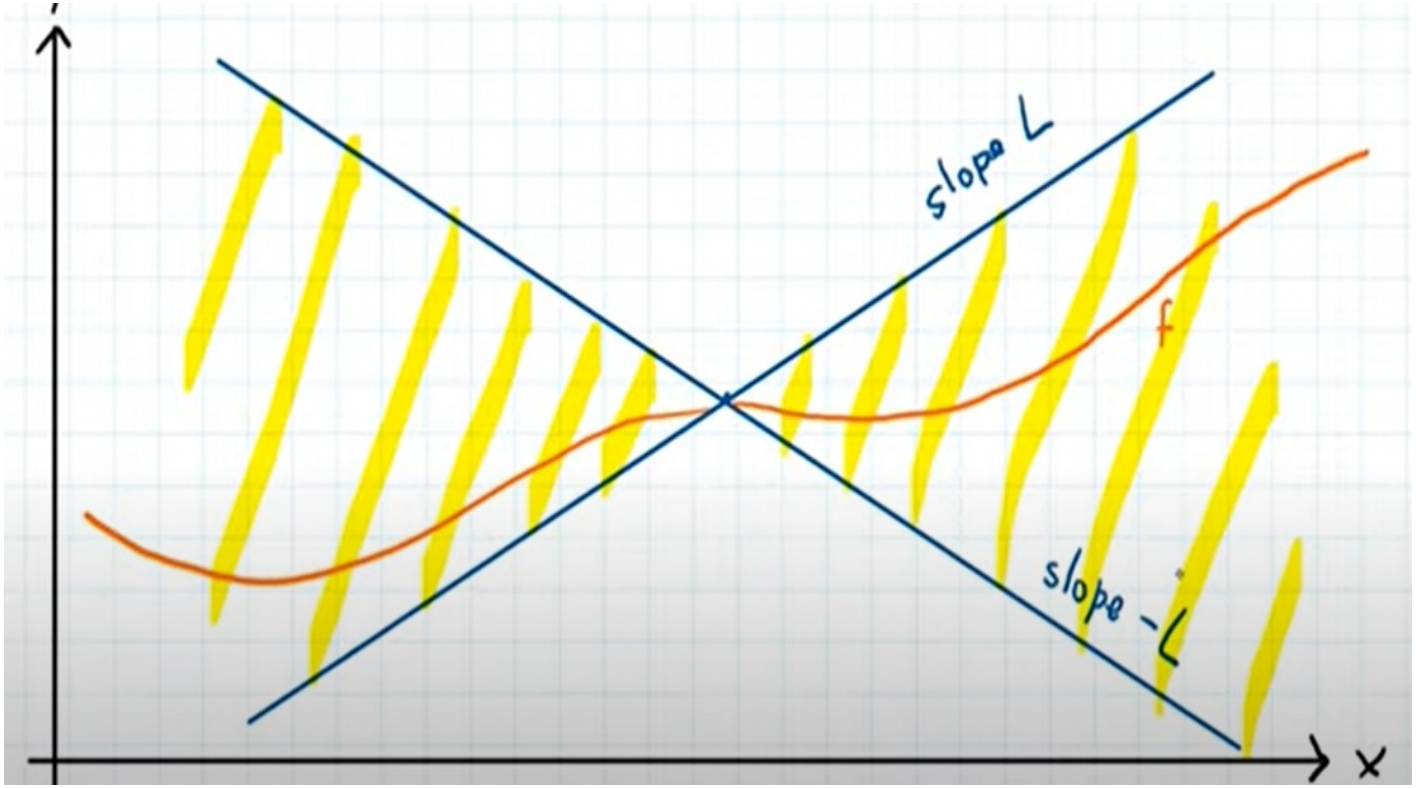


Figure: A lazy partner: Lipschitz function

Nicely behaved; Never gets out of the cone

LALR: knowing the "lr"

Minimal assumption: functions are Lipschitz continuous and differentiable up to first order only [a]

[a] Note this is a weaker condition than assuming the gradient of the function being Lipschitz continuous. We exploit merely the boundedness of the gradient.

$\alpha = \frac{1}{L}$, we force $\Delta \mathbf{w} \leq 1$, constraining the change in the weights. Stress: Not computing the Lipschitz constants of the *gradients* of the loss functions, but of the losses themselves. Assume: the loss is L-Lipschitz. Argument: set the learning rate to the reciprocal of the Lipschitz constant. Claim: supported by our experimental results.

Let the loss be given by $E(\mathbf{a}^{[L]}) = \frac{1}{2m}\left(\mathbf{a}^{[L]} - \mathbf{y}\right)^2$ *where the vectors contain the values for each training example.*

Separately compute the Lipschitz constant for a linear regression model and for neural networks;
Prove the equivalence of the two results, deriving the former as a special case of the latter.

$$g(\mathbf{w}) = \frac{1}{2m}\sum_{i=1}^{m}\left(\mathbf{x}^{(i)}\mathbf{w} - y^{(i)}\right)^2$$

$$
\begin{aligned}
g(\mathbf{w}) - g(\mathbf{v}) &= \frac{1}{2m} \sum_{i=1}^{m} \left(\mathbf{x}^{(i)} \mathbf{w} - y^{(i)} \right)^2 - \left(\mathbf{x}^{(i)} \mathbf{v} - y^{(i)} \right)^2 \\
&= \frac{1}{2m} \sum_{i=1}^{m} \left(\mathbf{x}^{(i)} (\mathbf{w} + \mathbf{v}) - 2y^{(i)} \right) \left(\mathbf{x}^{(i)} (\mathbf{w} - \mathbf{v}) \right) \\
&= \frac{1}{2m} \sum_{i=1}^{m} \left((\mathbf{w} + \mathbf{v})^T \mathbf{x}^{(i)T} - 2y^{(i)} \right) \left(\mathbf{x}^{(i)} (\mathbf{w} - \mathbf{v}) \right) \\
&= \frac{1}{2m} \sum_{i=1}^{m} \left((\mathbf{w} + \mathbf{v})^T \mathbf{x}^{(i)T} \mathbf{x}^{(i)} - 2y^{(i)} \mathbf{x}^{(i)} \right) (\mathbf{w} - \mathbf{v})
\end{aligned}
$$

Note: $(\mathbf{w} + \mathbf{v})^T \mathbf{x}^{(i)T}$ is a real number, whose transpose is itself.

Take the norm on both sides, and then assume that $\mathbf{w}$ and $\mathbf{v}$ are bounded such that $\|\mathbf{w}\|, \|\mathbf{v}\| \leq K$.

$$\boxed{\frac{\|g(\mathbf{w}) - g(\mathbf{v})\|}{\|\mathbf{w} - \mathbf{v}\|} \leq \frac{K}{m}\left\|\mathbf{X}^T\mathbf{X}\right\| + \frac{1}{m}\left\|\mathbf{y}^T\mathbf{X}\right\|}$$

Note: 1. We are forced to use separate norms because the matrix subtraction $2K\mathbf{X}^T\mathbf{X} - 2\mathbf{y}^T\mathbf{X}$ cannot be performed.
2. The RHS here is the Lipschitz constant.
3. Lipschitz constant changes if the cost function is considered with a factor other than $\frac{1}{2m}$.

Let the loss be given by $E(\mathbf{a}^{[L]}) = \frac{1}{2m}\left(\mathbf{a}^{[L]} - \mathbf{y}\right)^2$ *where the vectors contain the values for each training example.*

Note: Chain Rule in GD

$$\frac{\partial E}{\partial w_{ij}^{[L]}} = \frac{\partial E}{\partial a_j^{[L]}} \cdot \frac{\partial a_j^{[L]}}{\partial z_j^{[L]}} \cdot \frac{\partial z_j^{[L]}}{\partial w_{ij}^{[L]}}$$
$$= \frac{\partial E}{\partial a_j^{[L]}} \cdot \frac{\partial a_j^{[L]}}{\partial z_j^{[L]}} \cdot a_j^{[L-1]}$$

This gives us $\max_{i,j} \frac{\partial E}{\partial w_{ij}^{[L]}} = \max_j \frac{\partial E}{\partial a_j^{[L]}} \cdot \max_j \frac{\partial a_j^{[L]}}{\partial z_j^{[L]}} \cdot \max_j a_j^{[L-1]}$. The third part cannot be analytically computed; we denote it as K_z.

The formula!

Using the results above and other "tricks", we obtain:

$\max_{i,j} \frac{\partial E}{\partial w_{ij}^{[L]}} = \frac{K}{m}\mathbf{X}^T\mathbf{X} - \frac{1}{m}\mathbf{y}^T\mathbf{X}$; K is the upper bound on the weight vectors

Binary Classification-binary cross entropy loss: $\boxed{L = \frac{1}{2m}\mathbf{X}}$

Multiclass Classification-softmax regression loss: $\boxed{L = \frac{k-1}{km}\mathbf{X}}$

- L is lipschitz constant; **lr**, learning rate $= \frac{1}{L}$

Equivalence of the constants

Recall,

- $K_z = \max_j a_j^{[L-1]}$
- a linear regression model can be thought of as a neural network with no hidden layers and a linear activation (Why?)
- $\mathbf{a}^{[L-1]} = \mathbf{a}^0 = \mathbf{X} \rightarrow K_z = \max_j a_j^{[L-1]} = \mathbf{X}$

Note: K_a is the upper bound of the final layer activations. For a linear regression model, we have the "activations" as the outputs: $\hat{\mathbf{y}} = \mathbf{W}^T\mathbf{X}$. $\mathbf{W}$ has an upper bound $K \rightarrow$

$$K_a = \max \mathbf{a}^{[L]} = \max \mathbf{W}^T\mathbf{X} = \max \mathbf{W} \cdot \mathbf{X} = K\mathbf{X}$$

$$\max_{i,j} \frac{\partial E}{\partial w_{ij}^{[L]}} = \frac{1}{m}\left(K_a + \mathbf{y}\right) K_z = \frac{1}{m}\left(K\mathbf{X} + \mathbf{y}\right)\mathbf{X} = \frac{K}{m}\mathbf{X}^T\mathbf{X} + \frac{1}{m}\mathbf{y}^T\mathbf{X}$$

Such choice of Learning Rate?

Assumption: Gradients cannot change arbitrarily fast

A typical Gradient Descent algorithm is in the form of

$$\boldsymbol{w^{k+1}} = \boldsymbol{w^k} - \eta_k \nabla_w f \tag{1}$$

for $k \in \mathbb{Z}$ and stop if $\|\nabla_w f\| \leq \epsilon$
$\forall v, w, \exists L$ such that $\|f(\mathbf{v}) - f(\mathbf{w})\| \leq L \|\mathbf{v} - \mathbf{w}\|$. Also, $\nabla^2 f(w) \leq LI$

$$\boldsymbol{f(v)} = \boldsymbol{f(w)} + \nabla \boldsymbol{f(w)^T(v-w)} + \frac{1}{2}(\boldsymbol{v-w})^T \nabla^2 \boldsymbol{f(w)(v-w)}$$

$$\boldsymbol{f(v)} \leq \boldsymbol{f(w)} + \nabla \boldsymbol{f(w)^T(v-w)} + \frac{L}{2}\|\boldsymbol{v-w}\|^2 \tag{2}$$

- a convex quadratic upper bound which can be minimized using gradient descent with the learning rate $\eta_k = \frac{1}{L}$

Convergence

It follows, $f(w^{k+1}) \leq f(w^k) - \frac{1}{2L} \left\|\nabla f(w^k)\right\|^2$

Gradient Descent decreases $f(w)$ if $\eta_k = \frac{1}{L}$ and $\eta_k < \frac{2}{L}$. This proof also enables one to derive the rate of convergence with Lipschitz adaptive learning rate.

Let n represent the number of iterations. We know that,

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L} \left\|\nabla f(w^k)\right\|^2$$

Several steps after.... $n \geq \frac{2L(f(w^0) - f(w^*))}{\epsilon}$

Gradient Descent requires, at least, $n = O(\frac{1}{\epsilon})$ iterations to achieve the error bound: $\left\|\nabla f(w^k)\right\| \leq \epsilon$

Regularization

In particular, if an L_2 regularization term, $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ is added, it is trivial to show that the Lipschitz constant increases by λK, where K is the upper bound for $\|\mathbf{w}\|$. More generally, if a Tikhonov regularization term, $\|\mathbf{\Gamma w}\|_2^2$ term is added, then the increase in the Lipschitz constant can be computed as below.

$$\begin{aligned} L(\mathbf{w}_1) - L(\mathbf{w}_2) &= (\mathbf{\Gamma w}_1)^T(\mathbf{\Gamma w}_1) - (\mathbf{\Gamma w}_2)^T(\mathbf{\Gamma w}_2) \\ &= \mathbf{w}_1^T\mathbf{\Gamma}^2\mathbf{w}_1 - \mathbf{w}_2^T\mathbf{\Gamma}^2\mathbf{w}_2 \\ &= 2\mathbf{w}_2^T\mathbf{\Gamma}^2(\mathbf{w}_1 - \mathbf{w}_2) + (\mathbf{w}_1 - \mathbf{w}_2)^T\Gamma^2(\mathbf{w}_1 - \mathbf{w}_2) \\ \frac{\|L(\mathbf{w}_1) - L(\mathbf{w}_2)\|}{\|\mathbf{w}_1 - \mathbf{w}_2\|} &\leq 2\|\mathbf{w}_2\|\left\|\mathbf{\Gamma}^2\right\| + \|\mathbf{w}_1 - \mathbf{w}_2\|\left\|\mathbf{\Gamma}^2\right\| \end{aligned}$$

If $\mathbf{w}_1, \mathbf{w}_2$ are bounded by K,

$$\boxed{L = 2K\left\|\mathbf{\Gamma}^2\right\|}$$

Loss Function, NOT Activation

- For a neural network that uses the sigmoid, ReLU, or softmax activations, it is easily shown that the gradients get smaller towards the earlier layers in backpropagation. Because of this, the gradients at the last layer are the maximum among all the gradients computed during backpropagation.
- Our framework is independent of activation functions
- the framework is extensible to all loss function satisfying the *lipschitz condition*
- Thus, we set up a pipeline for classification problems-SYMNet.

- Economic theory is consistent with Our models: A new SVM Kernel and NN training (Activation functions derived from Lip Conditions on 1st Order DEs)
- We found the *suitable boy* (borrowing from Vikram Seth) LR $= 1/L$ and ..... implies not a **bandit** anymore!.

Progression in parameter handling: from tuning to selection

What's the big deal? Immensely cheaper computation! remarkable prediction performance!!

Recap: The beast conquered!

- Theoretical framework for computing an adaptive learning rate; this is also "adaptive" with respect to the data.
- "large" learning rates may not be harmful as once thought; remedy: guarded value of L_2 weight decay.

What did we gain?

- Novel Approximation functions used during forward pass with "little effort in parameter tuning, k, α; OUTCOME: Accuracy beating state-of-the-art
- Loss Function manipulation to devise learning rate formula; OUTCOME: NO Need to "handcraft" **lr**; performance as good as state of the art
- Metric gains: Time to converge, # of iterations to converge, CPU/RAM utilization

A pipeline with SBAF in forward pass + Adaptive **lr** in GD back-prop: SYMNET: Released v.1.0 ..https://github.com/sahamath/sym-netv1

Takeaway

So what have these new kids on the block done?

- saves number of iterations to converge
- accuracy, precision, recall and other performance metrics are better, at least on the data sets applied so far (13 different public data sets)
- Improvement in CPU and RAM utilization
- non-negligible improvement in time complexity

Parsimonious computing; *something I always wanted to do*

Adaptive Framework

- A theoretical framework for Lipschitz Adaptive Learning Rate(LALR) for regression-based tasks
- For Mean Absolute Error, LALR provides a 5x-20x increase in speed of convergence
- For quantile loss, an increase in speed of convergence by nearly 20x is observed
- Although the adaptive learning rate paradigm begins with large values of learning rate, a natural decay in the learning rate is observed

The Gene Expression Problem

Sustainable AI: Speeding up inference, cheaper H/W!!

- Gene expression profiling is used for studying gene expression patterns to determine genetic behaviour of cells.
- 22,000 genes across the entire human genome; approximately 1000 are earmarked as landmark genes remaining are target genes.
- Linear regression and Kernel techniques – sub-optimal performance.
- deep learning approach D-GEX – SOTA performance: Expensive H/W!

Problem Statement: Solve the Multi-Target multivariate regression problem of gene expression profiling under conditions of compute and model parsimony

Build upon this work; beat the previous SOTA but with a shallower neural network (a new activation function and.novel adaptive learning rate)

Challenges: LARGE GEO datase–training data > 3 GB, impossible to train with a laptop naively; Cheaper set up needed!!!

Use a clever learning rate for fast convergence:

$$\max_{i,j} \frac{\partial E}{\partial w_{ij}^{[L]}} = \frac{1}{m} \left(K_a + \mathbf{y}\right) K_z \tag{3}$$

Don't believe this? Please wait...

- D-GEX(D)3-layer architecture 9000×3; MAE= 0.3240; Us: D-GEX(L)(A-ReLU)2-layer architecture 9000×2; MAE= 0.3213
- Saha et. al ; LipGene: Lipschitz continuity guided adaptive learning rates for fast convergence on Microarray Expression Data Sets; IEEE/ACM Transactions on Computational Biology and Bioinformatics, Dec 2022, pp. 3553-3563, vol. 19

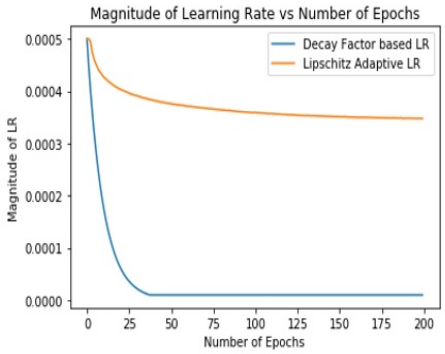
towards sustainable AI

Results and Discussion

Table 5: Comparison Of MAE on D-GEX(L) Vs D-GEX(D): Both models are trained on identical subsamples. Our model D-GEX(L) performs better

Epochs	D-GEX(D) 500	D-GEX(L) 500	D-GEX(D) 1500	D-GEX(L) 1500
100	0.4218	0.3807	0.3992	0.3672
200	0.38872	0.3780	0.3963	0.3646

Figure 1:LALR is always significantly higher than the Decay Based Learning Rate allowing much faster convergence



SPONSORS:

IEEE WCCI 2020 IEEE World Congress on Computational Intelligence Virtual Conference – July 19-24, 2020

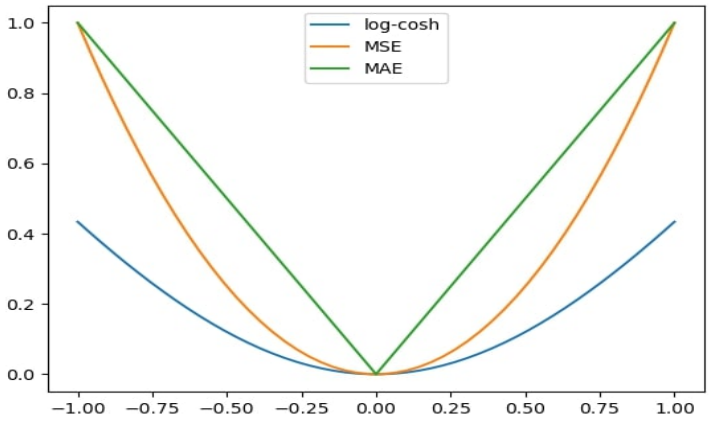
IEEE Advancing Technology for Humanity | IEEE Computational Intelligence Society | IET | The International Neural Network Society (INNS) | EPS Evolutionary Programming Society

Figure: Fast, accurate, cheaper: Our method for gene exp inference

Some Evidence

- Mohapatra, Saha et. al ; AdaSwarm: Augmenting gradient-based optimizers in Deep Learning with Swarm Intelligence, IEEE Transactions on Emerging Topics in Computational Intelligence; 6 (2), 329 - 340
- Saha et. al; Estimation and Applications of Quantiles in Deep Binary Classification; IEEE Transactions on Artificial Intelligence; 6 (2), 275-286, April 2022
- Saha et. al, Beginning with Machine Learning: A Comprehensive Primer; European Physical Journal-Special Topics, Springerhttps://doi.org/10.1140/epjs/s11734- 021-00209-7, 2021
- Saha et. al, LipschitzLR: Using theoretically computed adaptive learning rates for fast convergence; Applied Intelligence, Volume: 51 (3), 1460-1478, March 2021
- Saha et. al; ChaosNet: A Chaos based Artificial Neural Network Architecture for Classification; Chaos: An Interdisciplinary Journal of Nonlinear Science, 113-125, 29(11), June 2020

Best of both Worlds



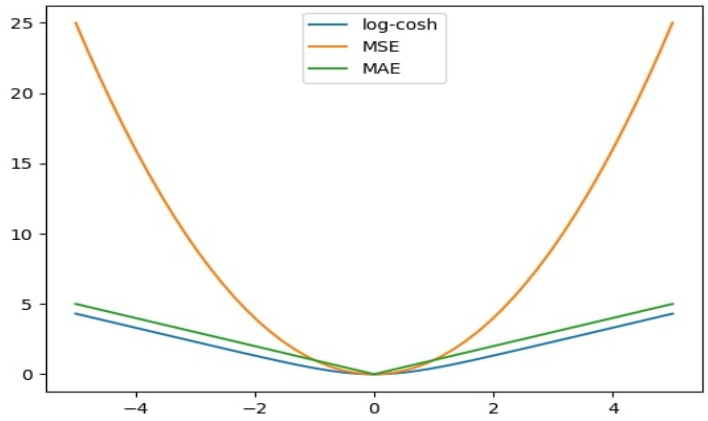


Figure: MSE-like behaviour of the log cosh ; MAE-like behaviour of the log cosh

$$\log(\cosh(x)) = \log(\frac{e^x + e^{-x}}{2})$$

$$= \begin{cases} |x| - \log(2) & \text{large x} \\ \frac{x^2}{2} & \text{small x} \end{cases}$$

log − cosh is *at least* 1-Lipschitz

[Tighter Lipschitz Constant of the log − cosh] log − cosh is Lipschitz, with a Lipschitz constant: $\frac{1}{m}\tanh(g(0) - \| y \|).\max_j a_j^{[L]}$

[log − cosh is convex] i.e. $J = \Sigma_{i=1}^{m} \log\cosh(y_i - \theta^T x_i)$ is convex.

Extension to the Binary-Classification setting:[Lipschitz constant for the *sBQC*] The Binary Smooth Quantile Classification Loss has the Lipschitz constant: $\frac{2}{\pi}\max\left(1, \frac{1-\tau}{\tau}, \frac{\tau}{\tau-1}\right)$

log − cosh is robust to label noise!

THANK YOU!

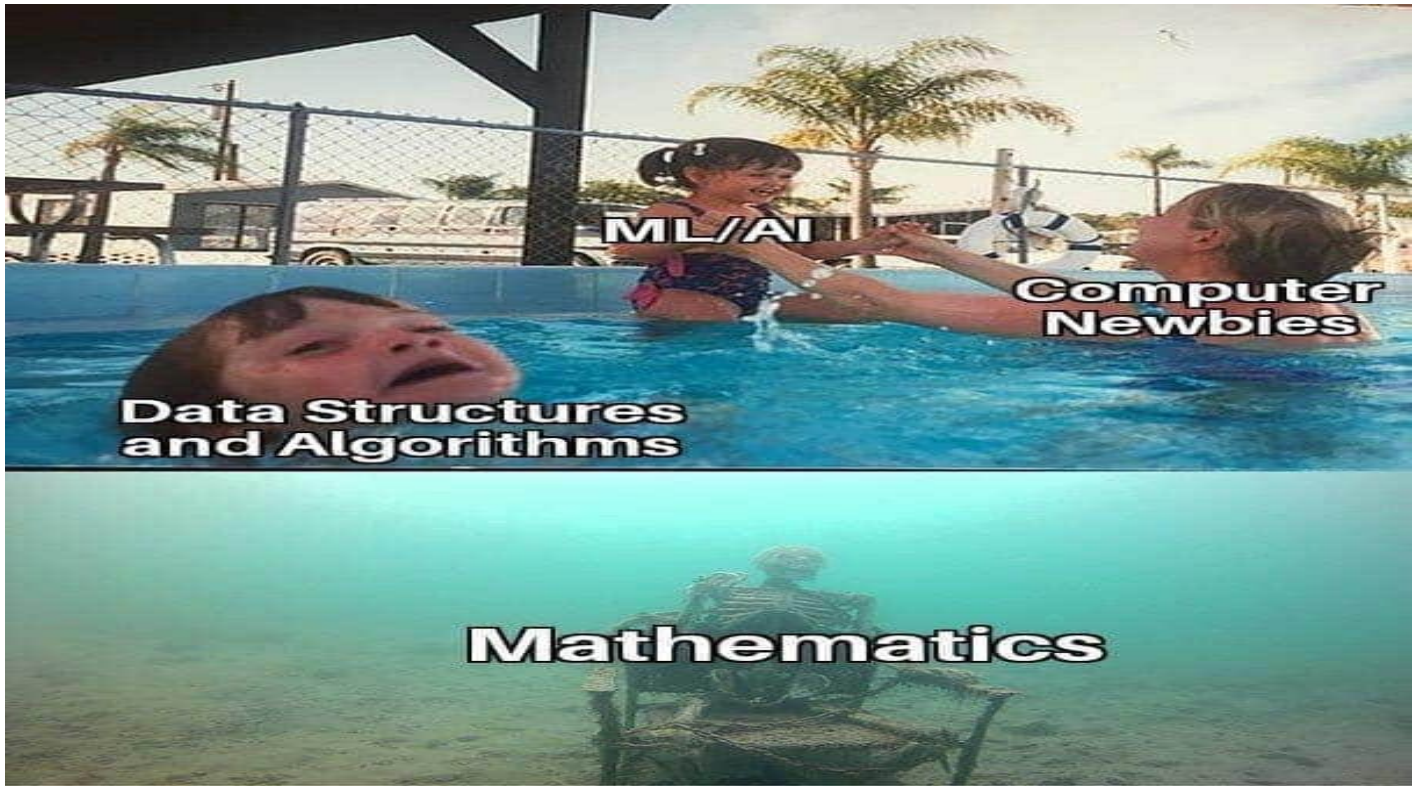


Figure: No Comment

I gratefully acknowledge multiple online sources for images used in the presentation.