Spring 2024

CMPE 258-01 Deep Learning

Dr. Kaikai Liu, Ph.D. Associate Professor

Department of Computer Engineering

San Jose State University

Email: kaikai.liu@sjsu.edu

Website: https://www.sjsu.edu/cmpe/faculty/tenure-line/kaikai-liu.php

Grading

Quiz (5%)
In-class mini-quiz

Homework&
Assignments (20%)

Midterm Exam (15%)+Final Exam (30%) = (45%)

Team Project (30%)

Grading policy for Homework Assignments

- Homework assignments will be assessed utilizing the EMRN rubrics, a four-level specification grading system.
  - "E" (excellent) and "M" (meets expectations) scores indicate successful completion (full marks). "R" (needs revision) and "N" (not assessable) scores correspond to lower marks, subject to assignment specifications.
  - Each homework assignment will have one revision period available.
  - Attaining an "E" does not affect the overall grade (full marks). However, students aiming for a final letter grade of "A" or "A+" must secure at least one "E" in homework assignments.
  - **Students got “E” in the homework, needs present their solution in the class.**

Determination of Grades

- The final grades will be calculated according to the final weighted score of all the assignments (exams, homework assignments, projects). The ABCDE grades will be determined by a set of thresholds that meets appropriate distribution and all required policies, which means what you perceive as a low score by summing up all your raw points will not necessarily prevent you from getting a good grade.
- No extra credit options.
- Assignment/exam re-evaluation request will not be considered unless the evaluation process is wrong.
- Peer ratings and comments will be considered for the project scores.
- The report will not be accepted when the Canvas assignment is closed. The assignment will receive zero grade.
- For turnitin-enabled assignment, the format of the submission should be acceptable to turnitin (such as WORD and PDF); otherwise the reports will be considered late and be penalized as above.

Classroom Protocol

- Students are responsible for lecture, book sections, project presentations, and any instructions given in the class.
- Avoid disturbing the class.
  - Students causing disruption in the class for other activities will be asked to leave the class and will be referred to the Judicial Affairs Officer of the University for disrupting the class after repeated offenses.
- Students should attend all meetings of the class.
  - Student Excused Absences: https://www.sjsu.edu/senate/docs/S22-2.pdf
  - Student shall notify the instructor in writing

The following are situations when an excused absence could become an incomplete or a course withdrawal. Students should consult with their instructor and advisor to determine the most suitable course of action.

- If the absence exceeds two consecutive weeks of class time.
- If the student returns to the class and attempts in good faith to complete the missing work but is overwhelmed and cannot finish.

Project Key Components

- Build a full pipeline of Deep Learning application with model training
  - Question/Problem Formulation: **propose** your own application and formulate the problem
  - Data Acquisition and Processing:
    - Identify a Suitable Dataset for Model Training and Evaluation
  - **State-of-the-Art Models**
    - Analyze the data, assess and compare various state-of-the-art open source models
  - **Model architecture change, Training, Evaluation, Fine-tunning**
    - Change the model architecture, tune parameters, and proceed with model training/fine-tunning, and evaluation. Gain the insights of the performance impact from the model architecture and parameters.
  - Inference, Optimization, and Real-time test
    - Create a comprehensive end-to-end deep learning application to enable inference using the trained model.
    - Depending on the chosen hardware platform, optimize model inference to enhance speed or reduce computational costs.
    - Perform evaluations and visualizations using real test data.

Common Datasets

- **Image Classification:**
  - **ImageNet** (https://www.image-net.org/): A large dataset with millions of labeled images across thousands of categories.
  - **CIFAR-10 and CIFAR-100**, **MNIST, FashionMNIST**
- **Object Detection and Segmentation:**
  - **COCO (Common Objects in Context):** A dataset with images containing objects labeled with bounding boxes and segmentation masks.
    - https://cocodataset.org/
  - **Autonomous driving related datasets**: Kitti, Waymo, Argo, nuScenes
- **Natural Language Processing:**
  - **IMDB Reviews:** A dataset of movie reviews classified as positive or negative sentiment.
  - **SQuAD (Stanford Question Answering Dataset):** A dataset for reading comprehension tasks, where models answer questions based on a given passage.
  - Machine Translation (WMT19).
- **Speech Recognition:**
  - **LibriSpeech:** A dataset of spoken words and sentences collected from audiobooks.
  - **Mozilla Common Voice:** A crowdsourced dataset of speech data for various languages.

Common Datasets

- **Time Series Analysis:**
  - **UCI Time Series Data Repository:** A collection of time series datasets for different applications, such as finance, health, and meteorology.
- **Video Analysis:**
  - **Kinetics:** A dataset for action recognition in videos with a wide range of human actions.
- **Medical Imaging:**
  - **MURA (Musculoskeletal Radiographs):** A dataset of bone X-rays for identifying musculoskeletal abnormalities.
  - **Chest X-Ray Images (Pneumonia):** A dataset for diagnosing pneumonia using chest X-ray images.
- **Autonomous Driving:**
  - **KITTI Vision Benchmark Suite** (https://www.cvlibs.net/datasets/kitti/)**:** A dataset for tasks like object detection, tracking, and scene understanding in autonomous driving scenarios.
  - Waymo open dataset: https://waymo.com/open/
  - Argo open dataset: https://www.argoverse.org/
  - NuScenes: https://www.nuscenes.org/

Project Requirement

- Embark on an engaging exploration of a topic that captivates your interest.
  - Limit one topic per group of four teams or fewer. While individual projects are encouraged, teams of 1-3 members are also welcomed. Should your group exceed 3 members, an additional project design and task distribution document (>1 page) must be submitted.
- The project encompasses three vital milestones:
  - A concise, one-page (single-spaced) project proposal.
  - An engaging presentation of your project to the class.
  - A comprehensive final project report, accompanied by code and a demonstration video.
- The assessment criteria encompass:
  - The importance of the problem addressed, the novelty of the solutions proposed, technical excellence, the degree of complexity, creativity, code clarity, presentation finesse, and documentation quality.
- Each project will be individually graded, ensuring a fair evaluation of efforts.

Additional Requirements

- It's advisable to avoid selecting a Basic Deep Learning tutorial as your project. There are numerous tutorials and online examples available for such projects.
- Copying existing tutorials or sample code is prohibited for the project. Your project must incorporate substantial modifications in terms of implementation, model selection, or inference optimization.
- Refrain from utilizing commercial APIs lacking open source code. While you can employ commercial APIs for comparative purposes, they shouldn't be your primary model (you need to identify open source alternatives).
- **Strive to delve into innovative solutions and engage with advanced models. Avoid relying solely on basic solutions to fulfill the project demonstration and meet minimum requirements**. If advanced approaches do not yield desired outcomes, rest assured there won't be penalties, provided you can effectively present your diligent efforts.

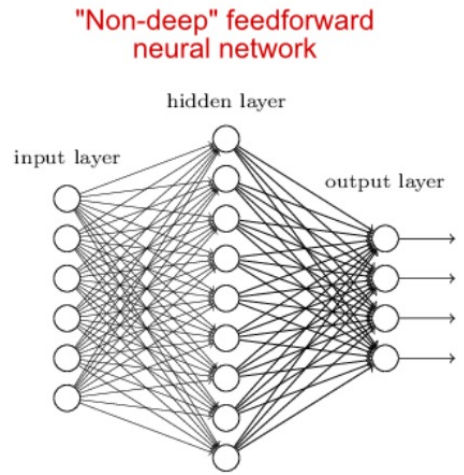
Deep Learning Thinking

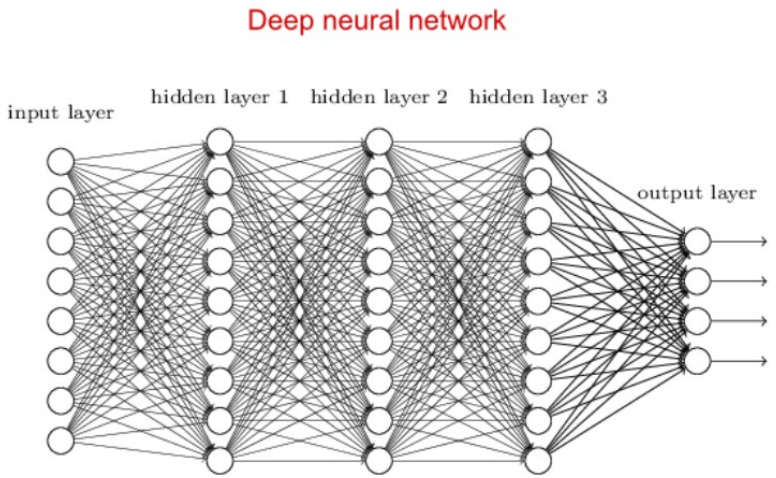
- Building the model is 'more similar to training a dog than to ordinary programming.'
  - Unlike ordinary software, our models are massive neural networks. Their behaviors are learned from a broad range of data, not programmed explicitly. Though not a perfect analogy, the process is more **similar to training a dog** than to ordinary programming. An initial "pre-training" phase comes first, in which the model learns to predict the next word in a sentence, informed by its exposure to lots of Internet text (and to a vast array of perspectives). This is followed by a second phase in which we "fine-tune" our models to narrow down system behavior. // OpenAI

Deep Learning

Neural Network

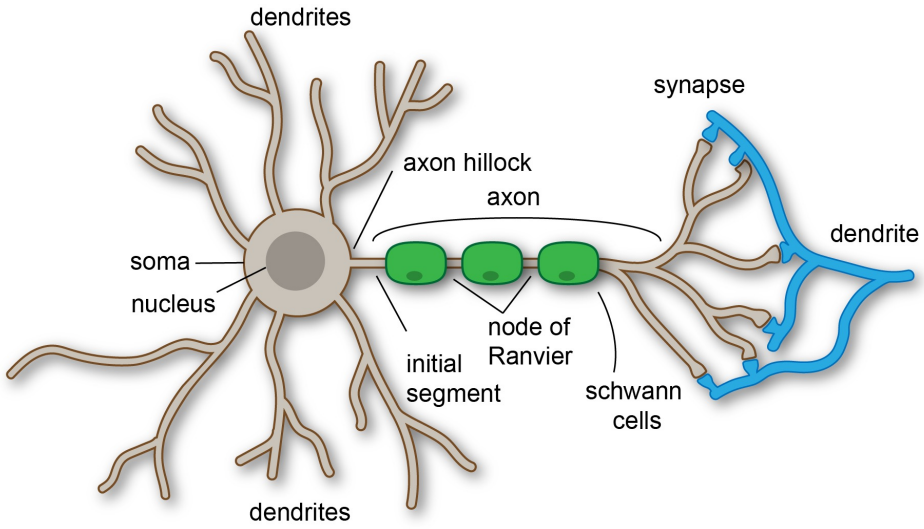
- Neural Network
  - Neural networks are a class of machine learning algorithms used to model complex patterns in datasets using multiple hidden layers and non-linear activation functions.
  - A neural network takes an input, passes it through multiple layers of hidden neurons (mini-functions with unique coefficients that must be learned), and outputs a prediction representing the combined input of all the neurons.

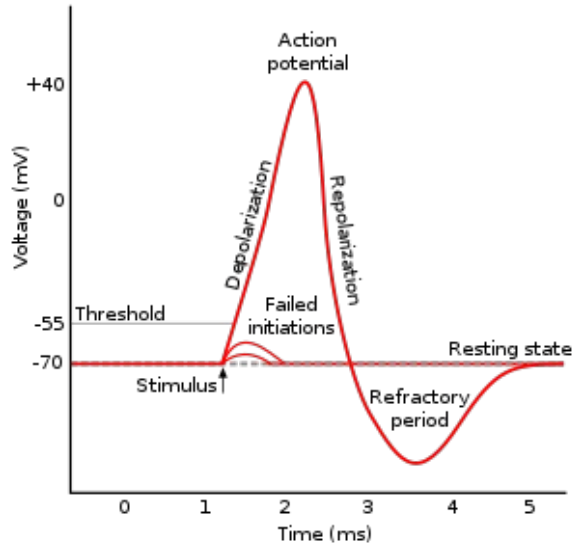




Real neuron

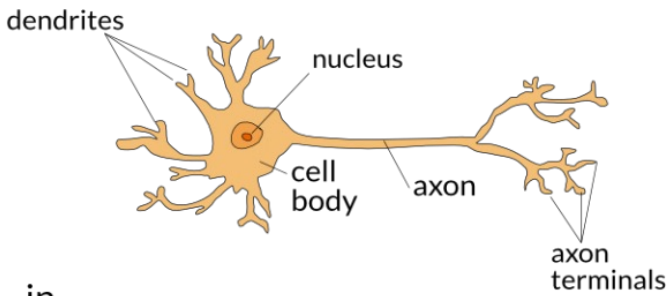
- Human brain is estimated to contain around 86,000,000,000 of such neurons. Each is connected to thousands of other neurons
  - **Dendrite**: is a branched protoplasmic extension of a nerve cell that propagates the electrochemical stimulation received from other neural cells to the cell body, or soma
  - **Soma**: or cell body is the bulbous, non-process portion of a neuron
  - **Axon**: or nerve fiber, is a long, slender projection of a nerve cell, or neuron that typically conducts electrical impulses known as action potentials away from the nerve cell body
  - **Action potentials** in neurons are also known as "nerve impulses" or "spikes", and the temporal sequence of action potentials generated by a neuron is called its "spike train". A neuron that emits an action potential, or nerve impulse, is often said to "fire".

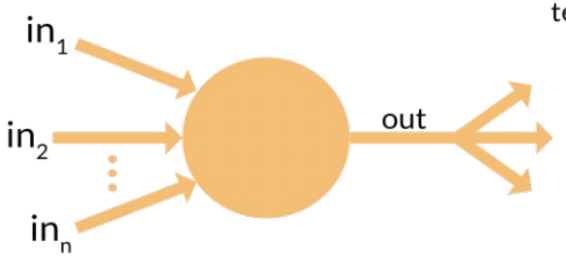




Artificial neuron

- The goal of simple artificial neurons models is to reflect some neurophysiological observations, not to reproduce their dynamics.
- Neurons in a layer are often called units. Parameters are often called weights.
  - Stateless wrt. Time, output real values

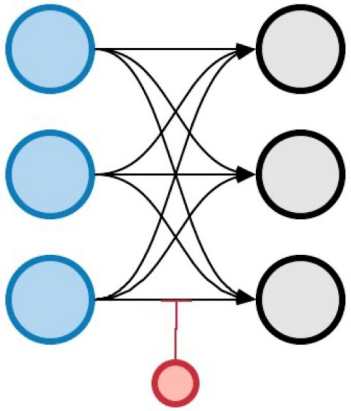




$$\sum_{i=1}^{d} \mathbf{w}_i \mathbf{x}_i + b$$

$$\sum_{i=0}^{d} \mathbf{w}_i \mathbf{x}_i \qquad \mathbf{x}_0 := 1$$

Collection of artificial neurons

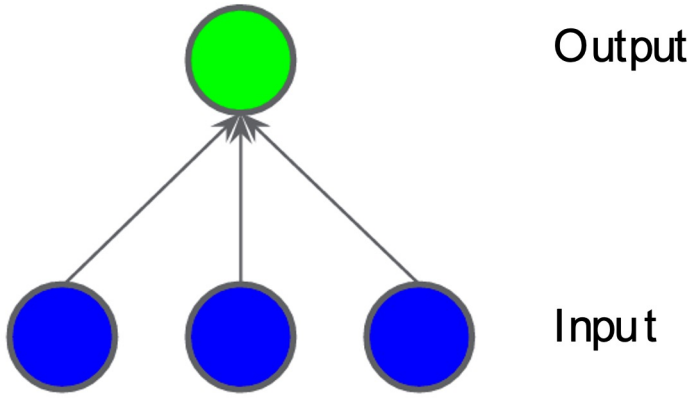


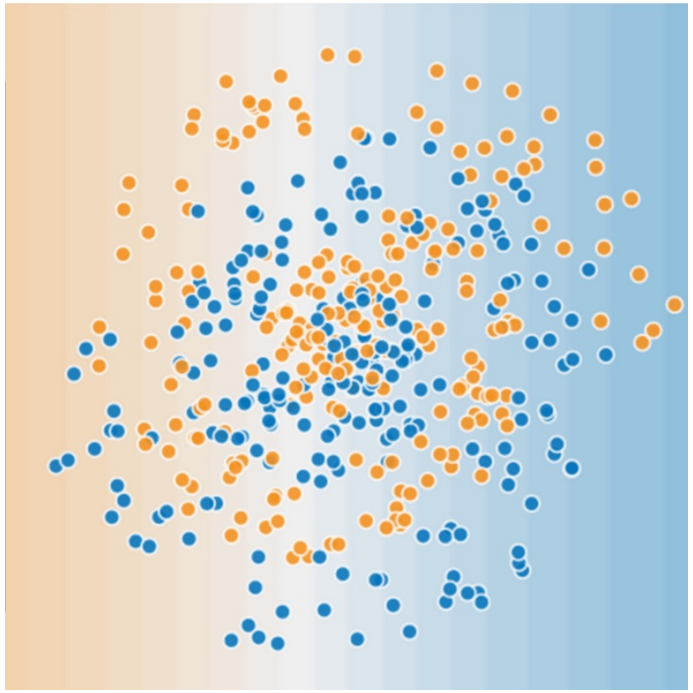
$$h(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

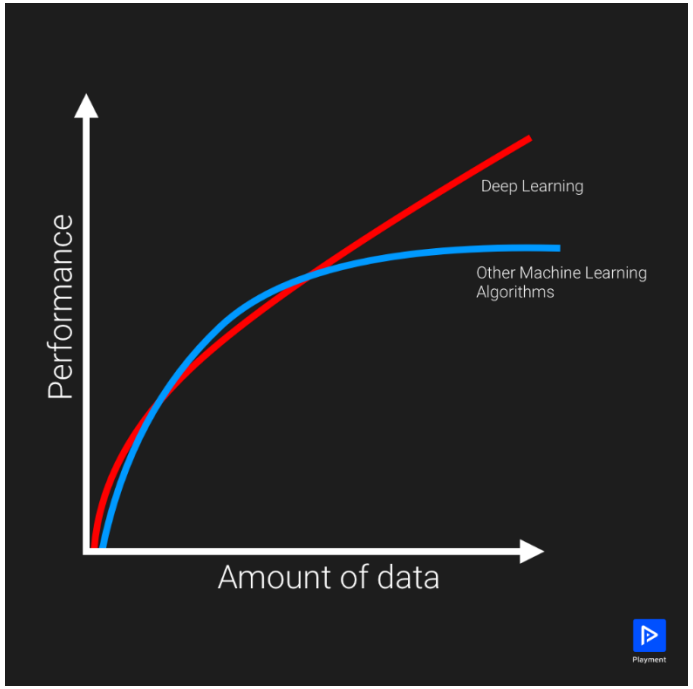
$$f_{\text{linear}}(\mathbf{x}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Nonlinear Classification Problem

- Nonlinear classification problem.
  - "Nonlinear" means that you can't accurately predict a label with a model of the form b+w1x1+w2x2 In other words, the "decision surface" is not a line.

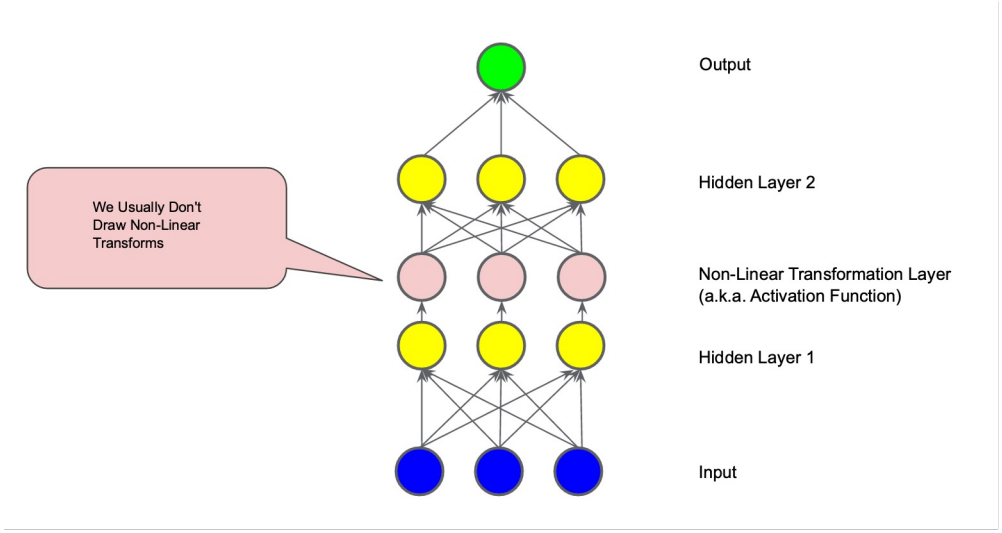


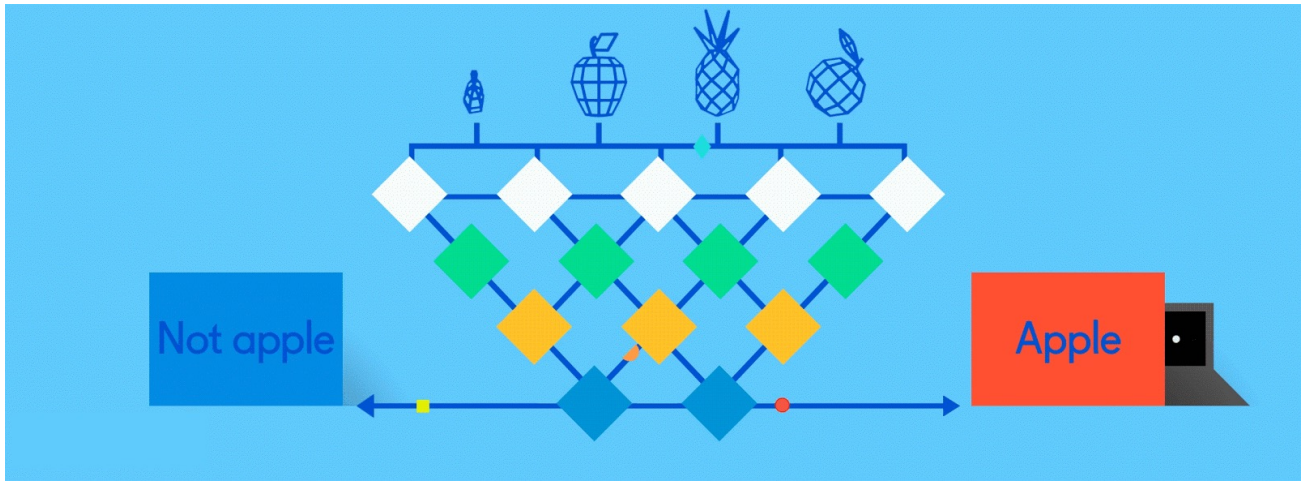




Neural Network

- Activation Functions
  - To model a nonlinear problem, we can directly introduce a nonlinearity. We can pipe each hidden layer node through a nonlinear function.
  - In the model represented by the following graph, the value of each node in Hidden Layer 1 is transformed by a nonlinear function before being passed on to the weighted sums of the next layer. This nonlinear function is called the activation function.





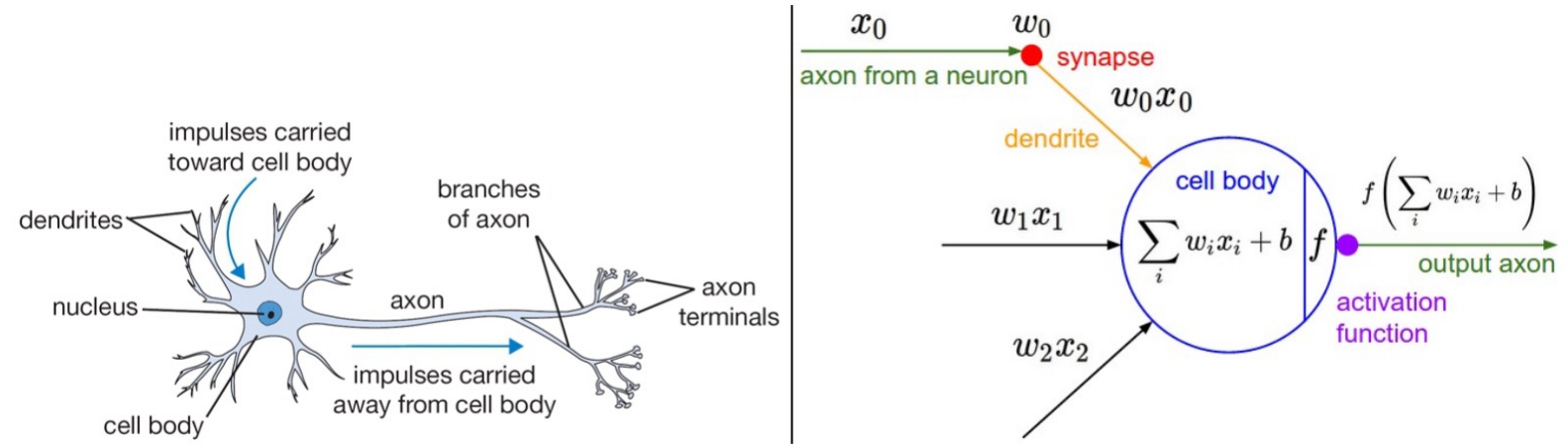
Activation functions

- ***Activation*** *functions* are really important for an Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.*They introduce non-linear properties to our Network*.
  - ***Their main purpose is to convert a input signal of a node in a A-NN to an output signal.*** That output signal now is used as a input in the next layer in the stack.
- Consider a neuron
  - the value of Y can be anything ranging from -inf to +inf. The neuron really doesn't know the bounds of the value. So how do we decide whether the neuron should fire or not
  - To check the Y value produced by a neuron and decide whether outside connections should consider this neuron as "fired" or not. Or rather let's say — "activated" or not.

$$Y = \sum (weight * input) + bias$$

Activation functions

- Activation function is a function used to transform the activation level of a unit (neuron) into an output signal
  - An activation function serves as a threshold, alternatively called classification or a partition. It essentially divides the original space into typically two partitions.
  - In its most general sense, a neural network layer performs a projection that is followed by a selection. Both projection and selection are necessary for the dynamics learning. The selection operation is enforces information irreversibility, an necessary criteria for learning.



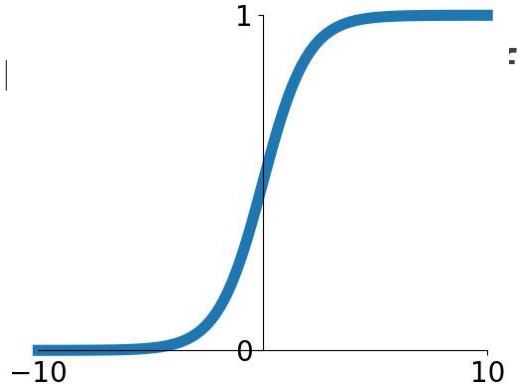
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Activation functions

- If we do not apply a Activation function then the output signal would simply be a simple ***linear function***.
  - A *linear function* is just a polynomial of **one degree.**
  - Now, a linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data. A Neural Network without Activation function would simply be a **Linear regression Model,** which has limited power and does not performs good most of the times. We want our Neural Network to not just learn and compute a linear function but something more complicated than that.
  - *Also without activation function our Neural network would not be able to learn and model other complicated kinds of data such as images, videos , audio , speech etc.*

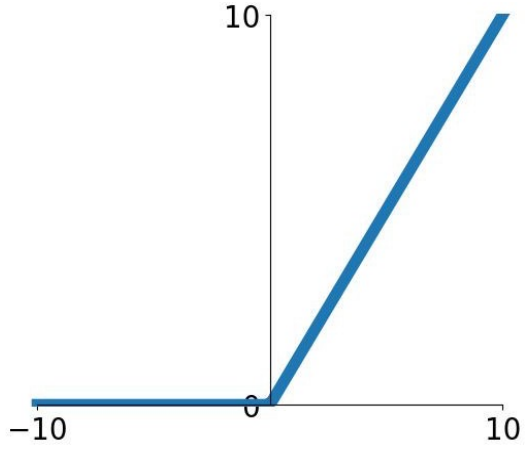
Activation functions

- Step and linear functions do not work well
- Sigmoid Function
  - Squashes numbers to range [0,1]
  - it is nonlinear in nature
  - It will give an analog activation unlike step function
  - unlike linear function, the output of the activation function is always going to be in range (0,1) compared to (-inf, inf) of linear function.
  - Historically popular since they have nice interpretation as a satu a neuron
- 3 problems:
  - Saturated neurons “kill” the gradients
    - “vanishing gradients”
  - Sigmoid outputs are not zero-centered
  - exp() is a bit compute expensive



$$\sigma(x) = 1/(1 + e^{-x})$$

ReLU

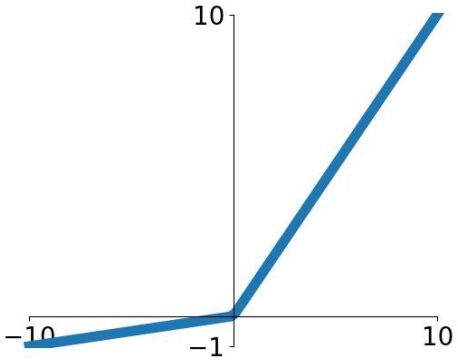


- ReLU (Rectified Linear Unit)
  - Computes f(x) = max(0,x)
  - Does not saturate (in +region)
  - Very computationally efficient
  - Converges much faster than sigmoid/tanh in practice (e.g. 6x)
  - ReLu give us this benefit: yields 0 activation because of the characteristic of ReLu (output 0 for negative values of x). This means a fewer neurons are firing (sparse activation) and the network is lighter.
- Problems
  - Not zero-centered output
  - gradient will be 0 because of which the weights will not get adjusted during descent. That means, those neurons which go into that state will stop responding to variations in error/ input (simply because gradient is 0, nothing changes). This is called dying ReLu problem.

Leaky ReLU

- Leaky ReLU
  - Does not saturate
  - Computationally efficient
  - Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
  - - will not “die”.

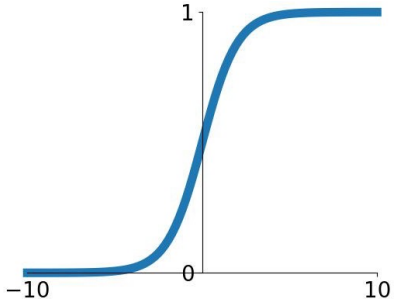
$$f(x) = \max(0.01x, x)$$



Activation Functions

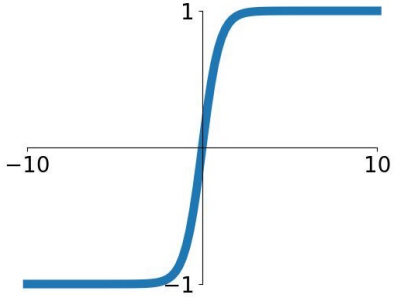
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



tanh

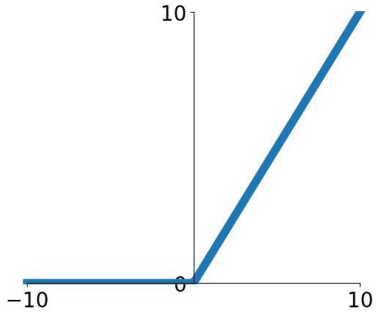
$$\tanh(x)$$



ReLU

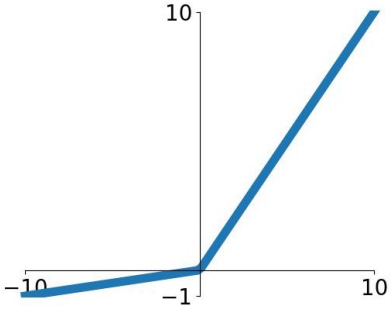
$$\max(0, x)$$

Good default choice



Leaky ReLU

$$\max(0.1x, x)$$

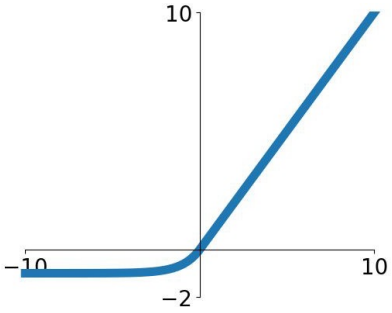


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

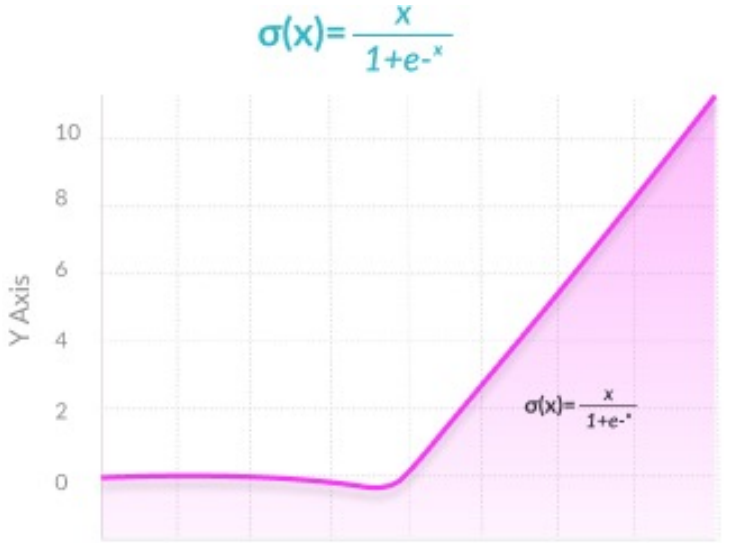
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



New Activation Functions

- Swish
  - Swish is a new, self-gated activation function discovered by researchers at Google.
  - Swish: a Self-Gated Activation Function: https://arxiv.org/abs/1710.05941v1

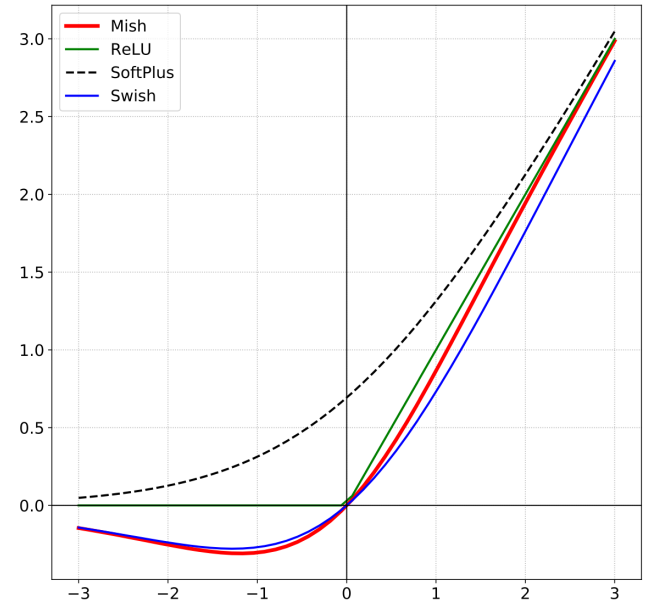
f(x)=x·sigmoid(x)



- Mish: A Self Regularized Non-Monotonic Activation Function
  - Paper (2019): https://arxiv.org/abs/1908.08681v3

$$f(x) = x \cdot \tanh \operatorname{softplus}(x)$$

$$\operatorname{softplus}(x) = \ln(1 + e^x)$$



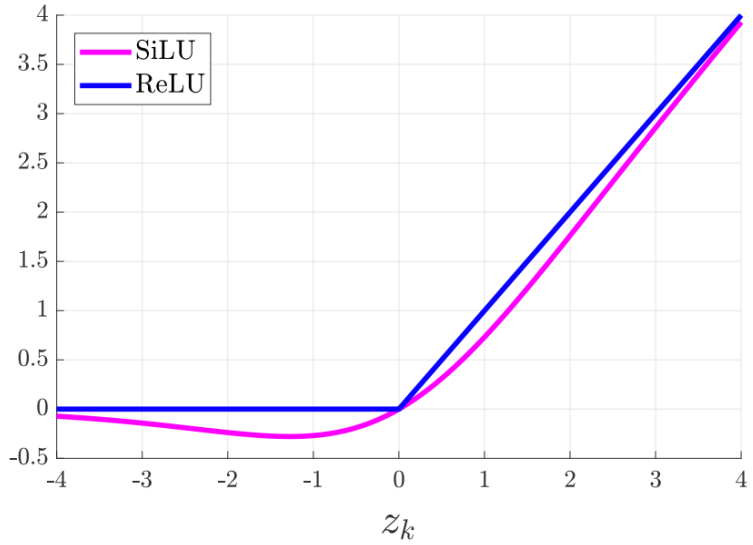
New Activation Functions

- Sigmoid Linear Units, or SiLUs
  - The activation of the SiLU is computed by the sigmoid function multiplied by its input
  - Paper (2017): https://arxiv.org/abs/1702.03118v3

$$x\sigma(x).$$

- Gaussian Error Linear Unit, or GELU
  - The GELU activation function is x times the standard Gaussian cumulative distribution function. The GELU nonlinearity weights inputs by their percentile
  - Paper (2016): https://arxiv.org/abs/1606.08415v5
  - GELUs are used in GPT-3, BERT, and most other Transformers.

$$\text{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}(x/\sqrt{2})\right],$$



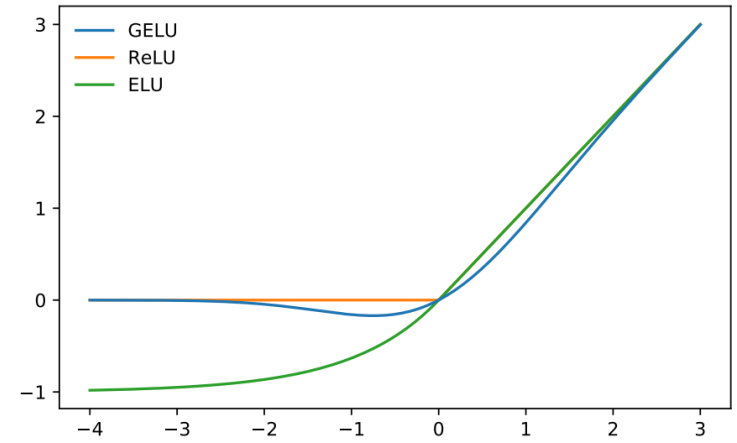
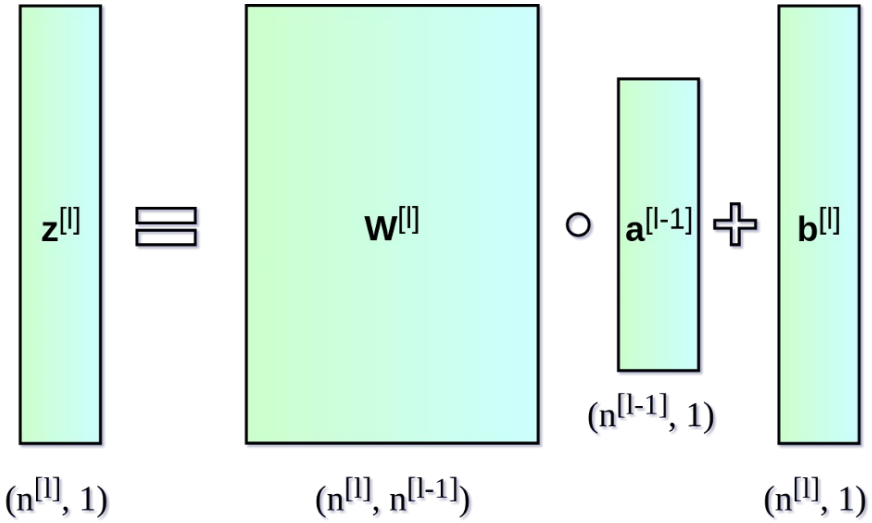


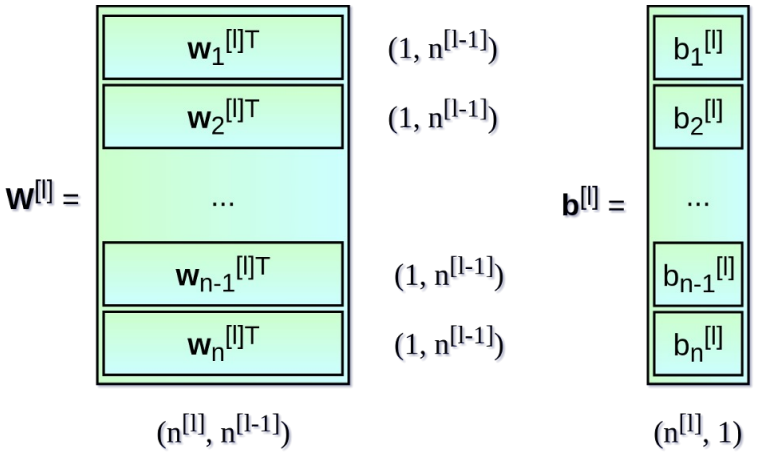
Figure 1: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU ($\alpha = 1$).

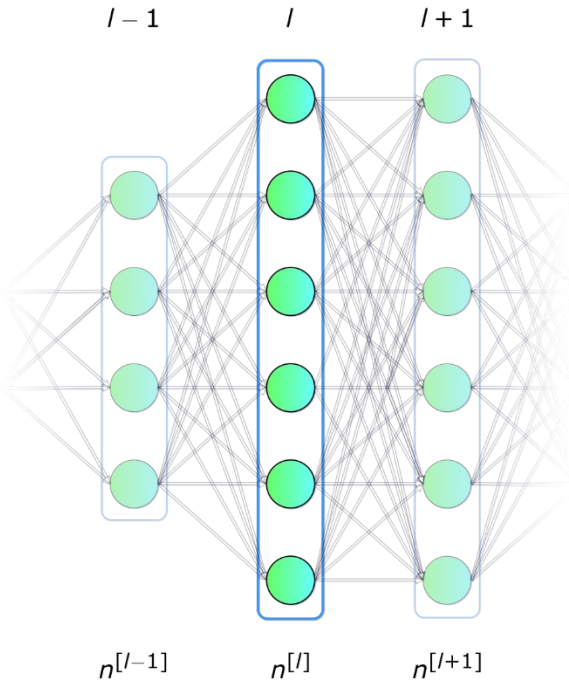
Neural Network

- To unify the notation, the equations will be written for the selected layer *[l]*. By the way, subscript *i* mark the index of a neuron in that layer.
- Each neuron in the layer performs a similar calculation according to the following equations:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{a}^{[l-1]} + \mathbf{b}^{[l]} \qquad \mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

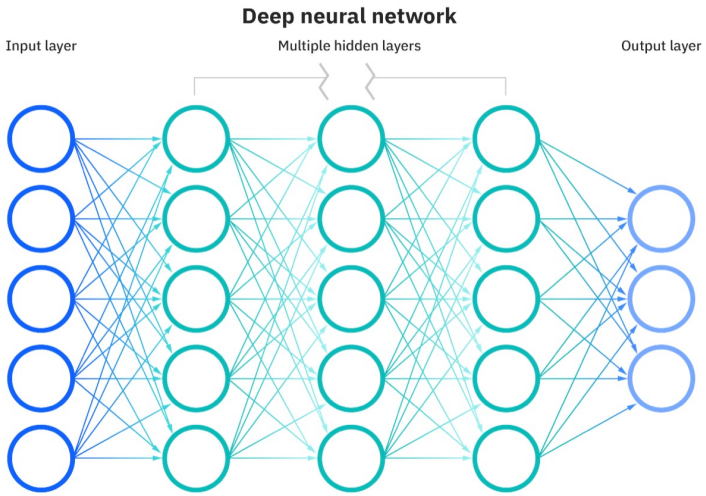


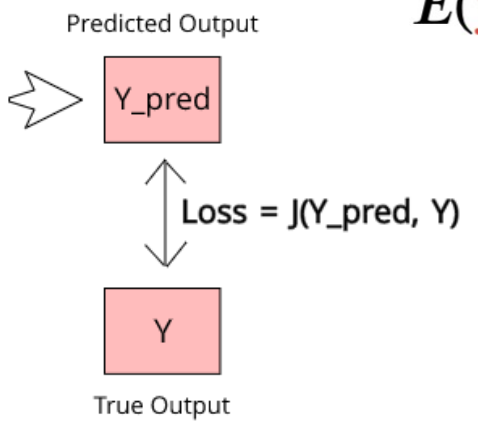




Neural Network Regression

- Neural Network for Regression:
  - Similar to Linear Regression that it also uses a set of inputs and weights to produce an output.
  - Regression Loss Functions: Mean Squared Error
  - Backpropagation allows us to calculate and attribute the error associated with each neuron, allowing us to adjust and fit the parameters of the model(s) appropriately.





$$E(y_{output}, y_{target}) = \frac{1}{2}(y_{output} - y_{target})^2$$

$$w_{ij} = w_{ij} - \alpha \frac{dE}{dw_{ij}}$$

Gradient descent

- Gradient descent is an optimization algorithm for minimizing the loss of a predictive model with regard to a training dataset.
  - Which step in this algorithm is most time consuming?
  - Typically the loss function is really the **average loss** over a **large dataset**.
  - **Loading** and **computing** on all the data is **expensive**

Gradient Descent Algorithm

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

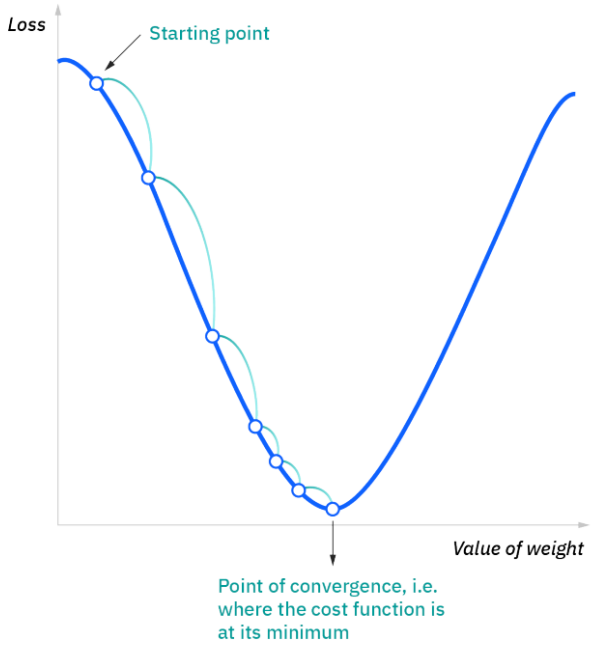
For τ from 0 to convergence:

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left(\nabla_\theta \mathbf{L}(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$$

(Evaluated at $\theta=\theta^{(\tau)}$)

$$\nabla_\theta \mathbf{L}(\theta) \Big|_{\theta=\theta^{(\tau)}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \text{loss}(y, f_\theta(x)) \Big|_{\theta=\theta^{(\tau)}}$$

This update is simultaneously done for all the weights.



Stochastic gradient descent

- Batch gradient descent computes the gradient using the whole dataset.
  - This is great for convex, or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution, either local or global.
- Stochastic gradient descent (SGD) computes the gradient using random sample.
  - Most applications of SGD actually use a minibatch of several samples.
  - SGD works well for error manifolds that have lots of local maxima/minima. In this case, the somewhat noisier gradient calculated using the reduced number of samples tends to jerk the model out of local minima into a region that hopefully is more optimal.
  - Single samples are really noisy, while **minibatches** tend to average a little of the noise out. Thus, the amount of jerk is reduced when using minibatches.

Stochastic Gradient Descent

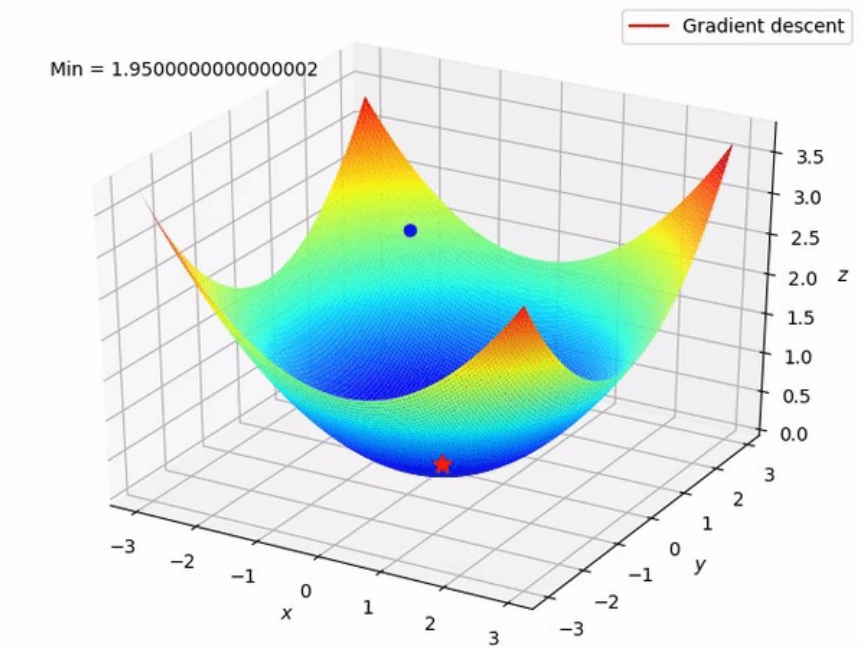
- Draw a simple random sample of data indices
  - Often called a **batch** or **mini-batch**
  - Choice of **batch size** trade-off **gradient quality** and **speed**
- Compute **gradient estimate** and uses as **gradient**

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

$\mathcal{B} \sim$ Random subset of indices

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \mathbf{L}_i(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$$



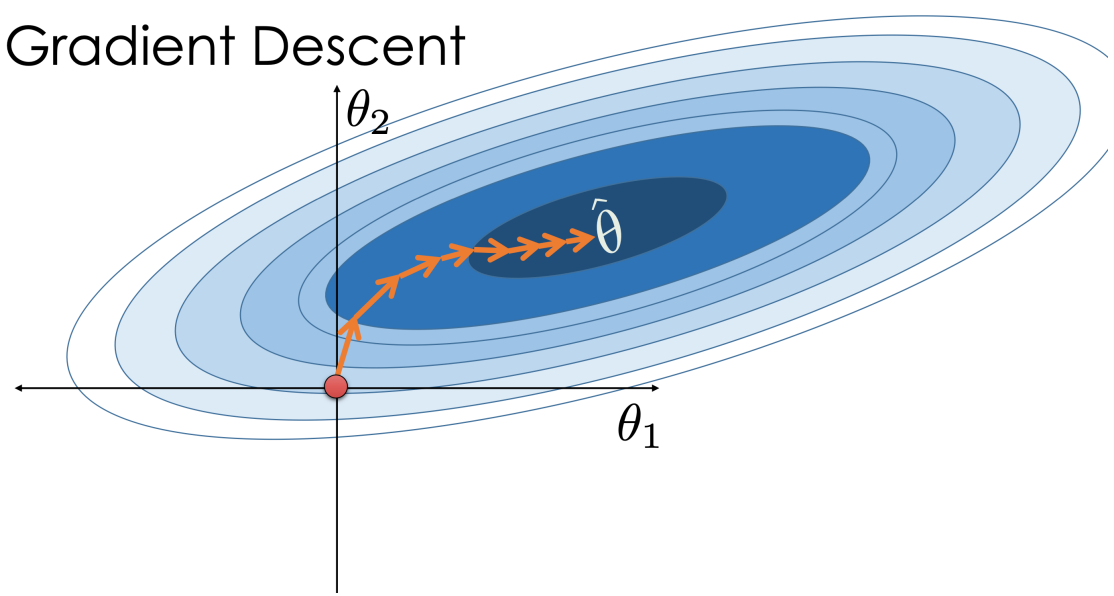
- Loss can be written as a sum of the loss on each record.

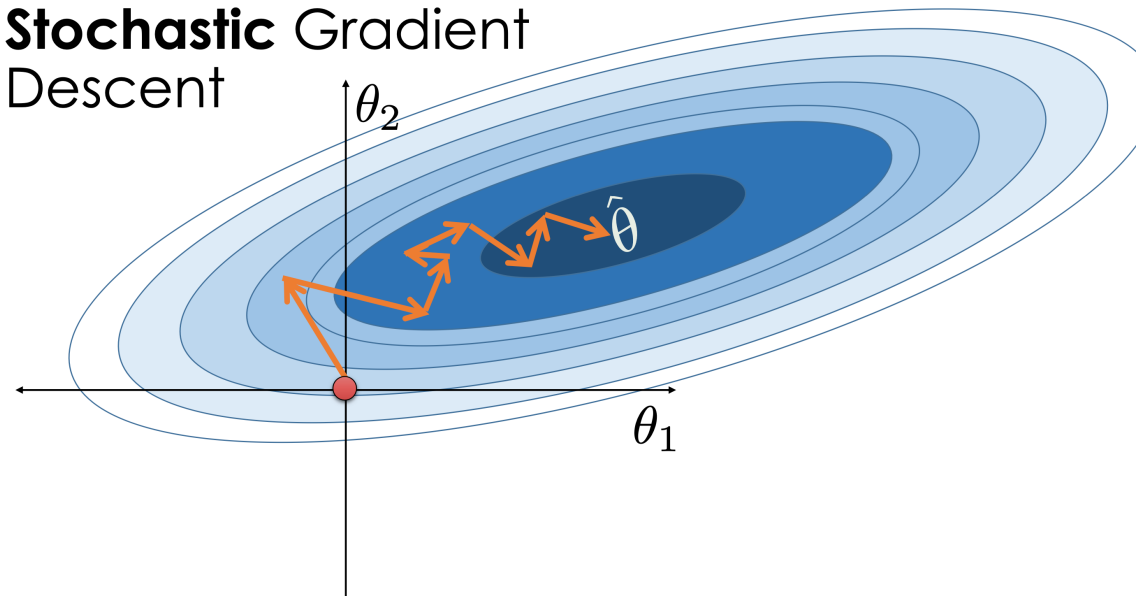
Decomposable Loss

$$\mathbf{L}(\theta) = \sum_{i=1}^{n} \mathbf{L}_i(\theta) = \sum_{i=1}^{n} \mathbf{L}(\theta, x_i, y_i)$$

Stochastic Gradient Descent

- **Dominant algorithm** in modern machine learning and AI
  - Some important **variations**
    - **Momentum** terms (SGD with Momentum)
    - **Automatic learning rates** (ADAM)
  - Core algorithm driving progress in **deep learning**
    - TensorFlow and Pytorch designed around SGD





Thank You

Address:
ENG257, SJSU

Email Address:
Kaikai.liu@sjsu.edu