



BST 261: Data Science II

Lecture 1

**Course Introduction, History of Deep Learning,
Neural Nets, Perceptrons and Backpropagation**

**Heather Mattie
Harvard T.H. Chan School of Public Health
Spring 2 2020**

Teaching Staff

Instructor

Heather Mattie

Instructor of Data Science

hemattie@hsph.harvard.edu

Building 1 Room 421A

(617) 432-5308

Office Hours: Tuesdays 1-2pm

Recipe of the Day

Cinnamon rolls



Teaching Staff - TAs

Kareem Carr

Biostatistics PhD Candidate

kareemcarr@g.harvard.edu

Gopal Kotecha

Biostatistics PhD Candidate

gkotecha@g.harvard.edu

Office Hours:

Mondays 12:45-2:45pm

Course Overview

Course content:

- ⦿ Computational and mathematical foundations of deep learning
- ⦿ Deep learning workflow
- ⦿ Bias/variance trade off
- ⦿ Feedforward networks (Multilayer perceptrons - MLPs)
- ⦿ Convolutional neural networks (CNNs)
- ⦿ Recursive/Recurrent neural networks (RNNs)
- ⦿ Deep learning research
- ⦿ Advanced topics in deep learning (GANs, VAEs, adversarial attacks)
- ⦿ Cloud computing with Google Cloud Platform (GCP)

Reading

- ◎ Most of the course content will come from these 2 books:
 - A lot of content will come from: Deep Learning. Ian Goodfellow, Yoshua Bengio, Aaron Courville. MIT Press, 2016. Book available at <http://www.deeplearningbook.org>
 - As well as: Deep Learning with Python. François Chollet. Manning Publications, 2017. Available at <https://www.manning.com/books/deep-learning-with-python>

◎ Other material will come from journal articles

Each student will present an article [from this list](#) during the course

Lectures

- ◎ Total of 16 lectures and 6-7 lab sessions; see syllabus for schedule
 - **All lectures, labs and office hours will be held via Zoom until further notice - and will be recorded**
- ◎ Lectures on Mondays & Wednesdays from 9:45 - 11:15am
- ◎ Slides will be available on the course website and [GitHub repo](#) before each lecture
- ◎ Lectures will be a mixture of theory and application
- ◎ All in-class coding examples will be in Python and use Keras with TensorFlow backend
- ◎ Labs will be held on Fridays 9:45 - 11:15am

Course Overview

Problem Sets:

- ◎ 2 problem sets
 - Problem Set 1 worth 20% of final grade
 - Problem Set 1 worth 30% of final grade
- ◎ Output is a Jupyter notebook consisting of code and text
- ◎ A Colab notebook template link will be sent out and available on the course Canvas site
- ◎ All assignments should be submitted on Canvas
- ◎ Problem Set #1 due Sunday **April 5** by 11:59pm
- ◎ May use 2 late days per problem set if needed

Course Overview

Paper Presentations:

- ◎ Each student is expected to read and present a paper to the class
- ◎ 5-7 minute presentations
- ◎ Slides are encouraged
 - Please email to Heather the night before or early the morning of your presentation
- ◎ 25% of final grade
- ◎ Sign-up list is [here](#)
- ◎ 2-3 students per lecture starting week 2

Course Overview

Group project proposal:

- ◎ Larger assignment that brings together different course themes
- ◎ Output is a Word doc or pdf file that should be submitted on Canvas
- ◎ Only one file needs to be submitted per group
- ◎ Groups may contain 1 - 4 students
- ◎ Due **May 15** by 11:59pm, 25% of grade
- ◎ See syllabus for details



Deep Learning

What is *Deep Learning*?

- ◎ Computers can solve problems that are intellectually difficult for humans
 - Ex: multiplication with large numbers and decimals, chess, etc.
 - Problems that can be described by a list of formal, mathematical rules
- ◎ Humans can solve problems that are intuitive, but difficult to describe formally and thus difficult for computers to solve
 - Ex: handwriting recognition, speech recognition, etc.

What is *Deep Learning*?

- ◎ Deep learning is a solution to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts
- ◎ The hierarchy of concepts builds on itself, producing a deep graph with many layers, leading to the concept of ***deep learning***

What is *Deep Learning*?

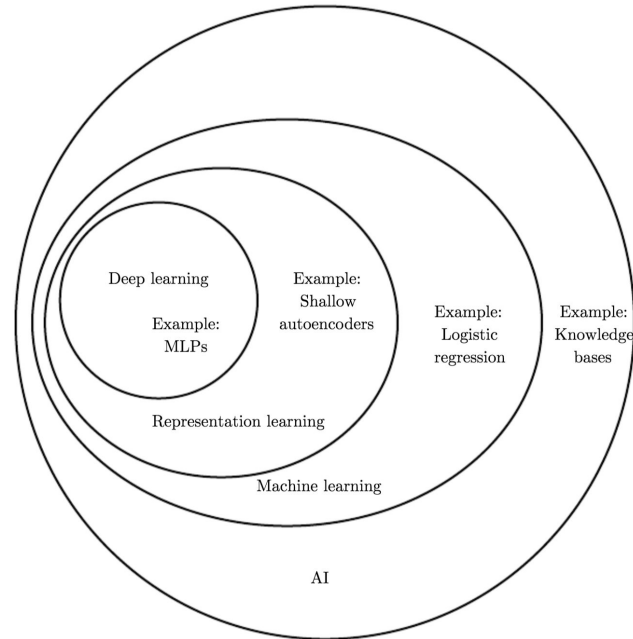
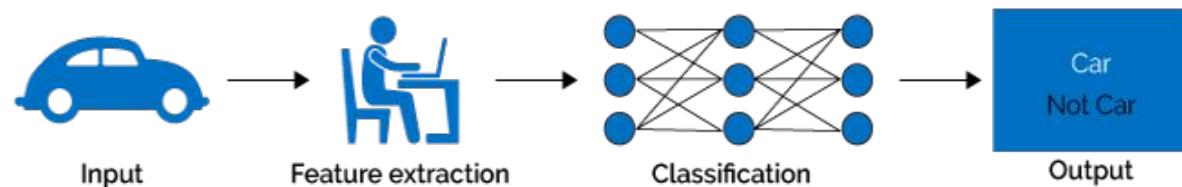
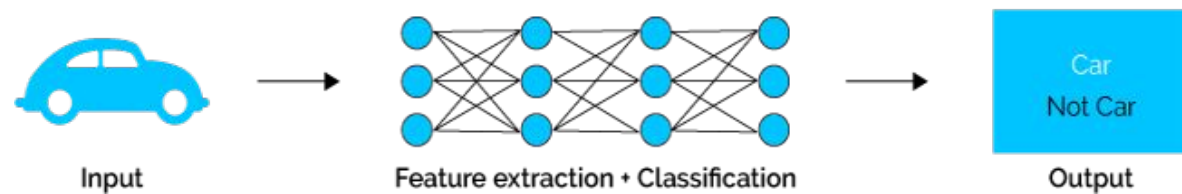


Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

Machine Learning



Deep Learning



ARTIFICIAL INTELLIGENCE

Any technique that enables computers to mimic human behavior



MACHINE LEARNING

Ability to learn without explicitly being programmed



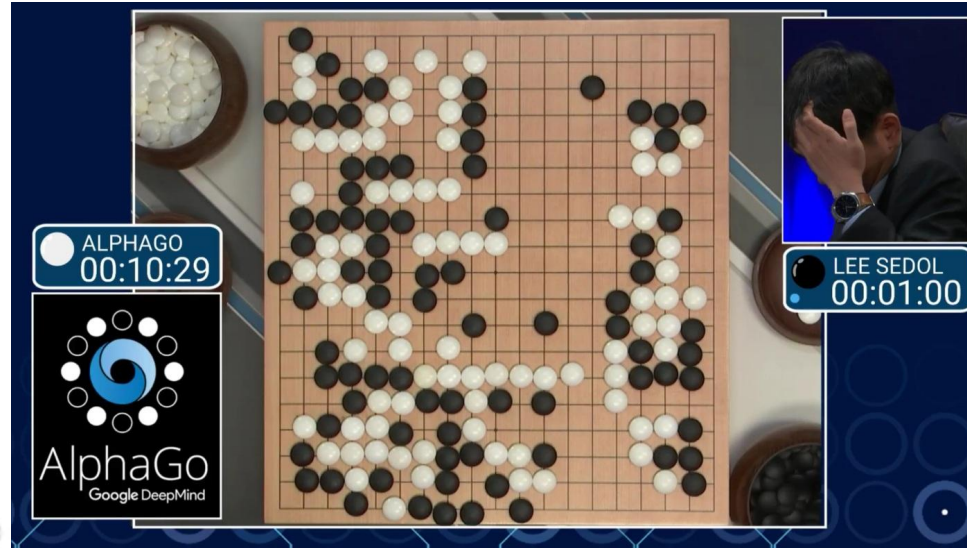
DEEP LEARNING

Extract patterns from data using neural networks

3 1 3 4 7 2
1 7 4 2 3 5

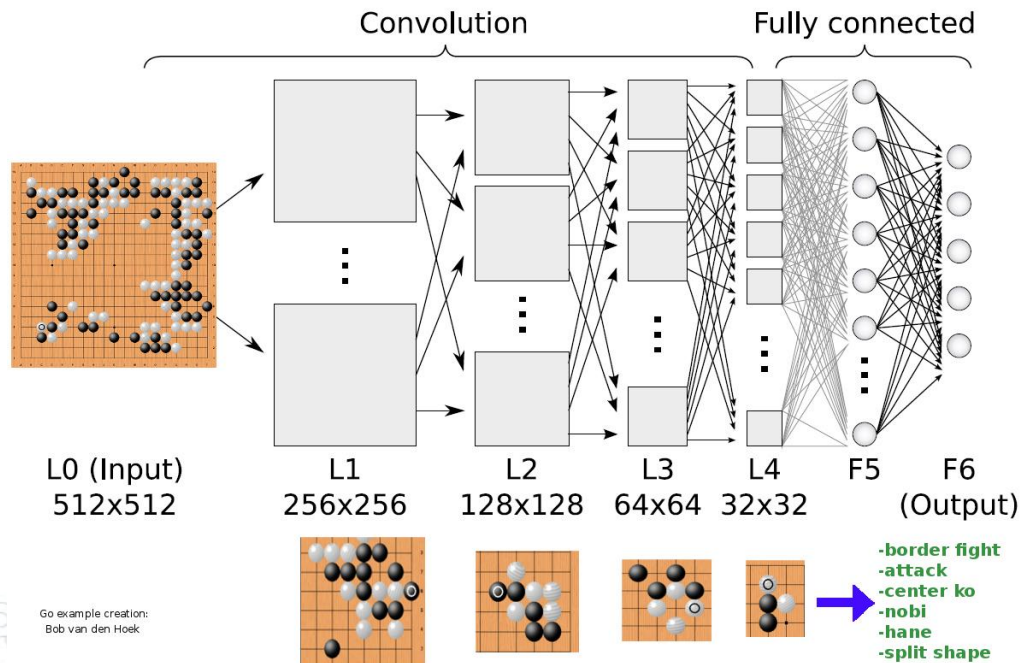
Deep Learning Successes

Games



Deep Learning Successes

Games



Deep Learning Successes

Art



Deep Learning Successes

Art

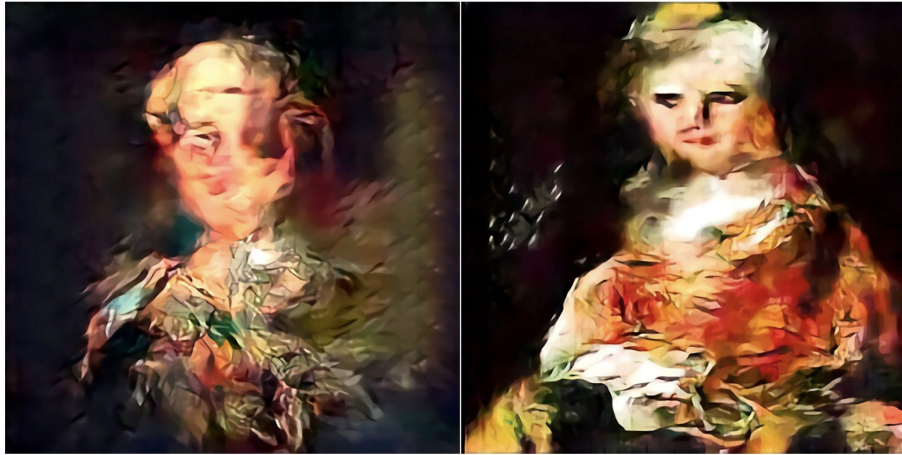


(a) *Input: a casual face photo*

(b) *Outputs: new headshots with the styles transferred from the examples. The insets show the examples.*

Deep Learning Successes

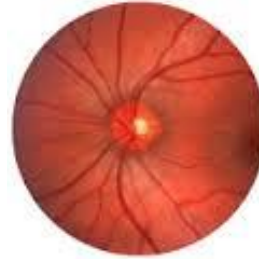
The AI-Art Gold Rush



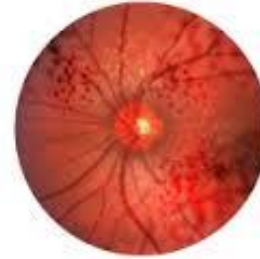
AI-generated "faceless portraits" by Ahmed Elgammal and AICAN. Photo: Artrendex Inc./The Atlantic

Deep Learning Successes

Medicine



Normal
Retina



Diabetic
Retina

JAMA | **Original Investigation** | INNOVATIONS IN HEALTH CARE DELIVERY

Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Varun Gulshan, PhD; Lily Peng, MD, PhD; Marc Coram, PhD; Martin C. Stumpe, PhD; Derek Wu, BS; Arunachalam Narayanaswamy, PhD; Subhashini Venugopalan, MS; Kasumi Widner, MS; Tom Madams, MEng; Jorge Cuadros, OD, PhD; Ramasamy Kim, OD, DNB; Rajiv Raman, MS, DNB; Philip C. Nelson, BS; Jessica L. Mega, MD, MPH; Dale R. Webster, PhD

Deep Learning Successes

CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning

Pranav Rajpurkar^{*1} Jeremy Irvin^{*1} Kaylie Zhu¹ Brandon Yang¹ Hershel Mehta¹
Tony Duan¹ Daisy Ding¹ Aarti Bagul¹ Robyn L. Ball² Curtis Langlotz³ Katie Shpanskaya³
Matthew P. Lungren³ Andrew Y. Ng¹

Abstract

We develop an algorithm that can detect pneumonia from chest X-rays at a level exceeding practicing radiologists. Our algorithm, CheXNet, is a 121-layer convolutional neural network trained on ChestX-ray14, currently the largest publicly available chest X-ray dataset, containing over 100,000 frontal-view X-ray images with 14 diseases. Four practicing academic radiologists annotate a test set, on which we compare the performance of CheXNet to that of radiologists. We find that CheXNet exceeds average radiologist performance on the F1 metric. We extend CheXNet to detect all 14 diseases in ChestX-ray14 and achieve state of the art results on all 14 diseases.



Input
Chest X-Ray Image

CheXNet
121-layer CNN

Output
Pneumonia Positive (85%)



1. Introduction

More than 1 million adults are hospitalized with pneumonia and around 50,000 die from the disease every year in the US alone (CDC, 2017). Chest X-rays

Deep Learning Successes

Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks

Pranav Rajpurkar*
Awni Y. Hannun*
Masoumeh Haghighpanahi
Codie Bourn
Andrew Y. Ng

PRANAVSR@CS.STANFORD.EDU
AWN1@CS.STANFORD.EDU
MHAGHPANAHI@IRHYTHMTECH.COM
CBOURN@IRHYTHMTECH.COM
ANG@CS.STANFORD.EDU

Abstract

We develop an algorithm which exceeds the performance of board certified cardiologists in detecting a wide range of heart arrhythmias from electrocardiograms recorded with a single-lead wearable monitor. We build a dataset with more than 500 times the number of unique patients than previously studied corpora. On this dataset, we train a 34-layer convolutional neural network which maps a sequence of ECG samples to a sequence of rhythm classes. Committees of board-certified cardiologists annotate a gold standard test set on which we compare the performance of our model to that of 6 other individual cardiologists. We exceed the average cardiologist performance in both recall (sensitivity) and precision (positive predictive value).

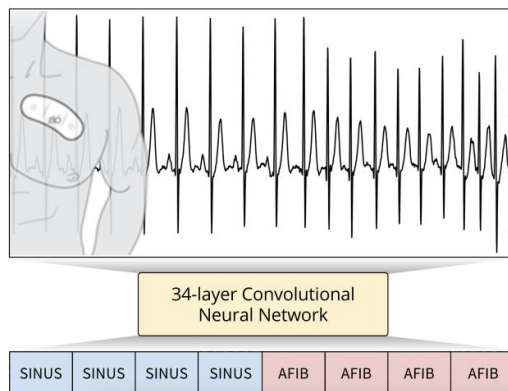
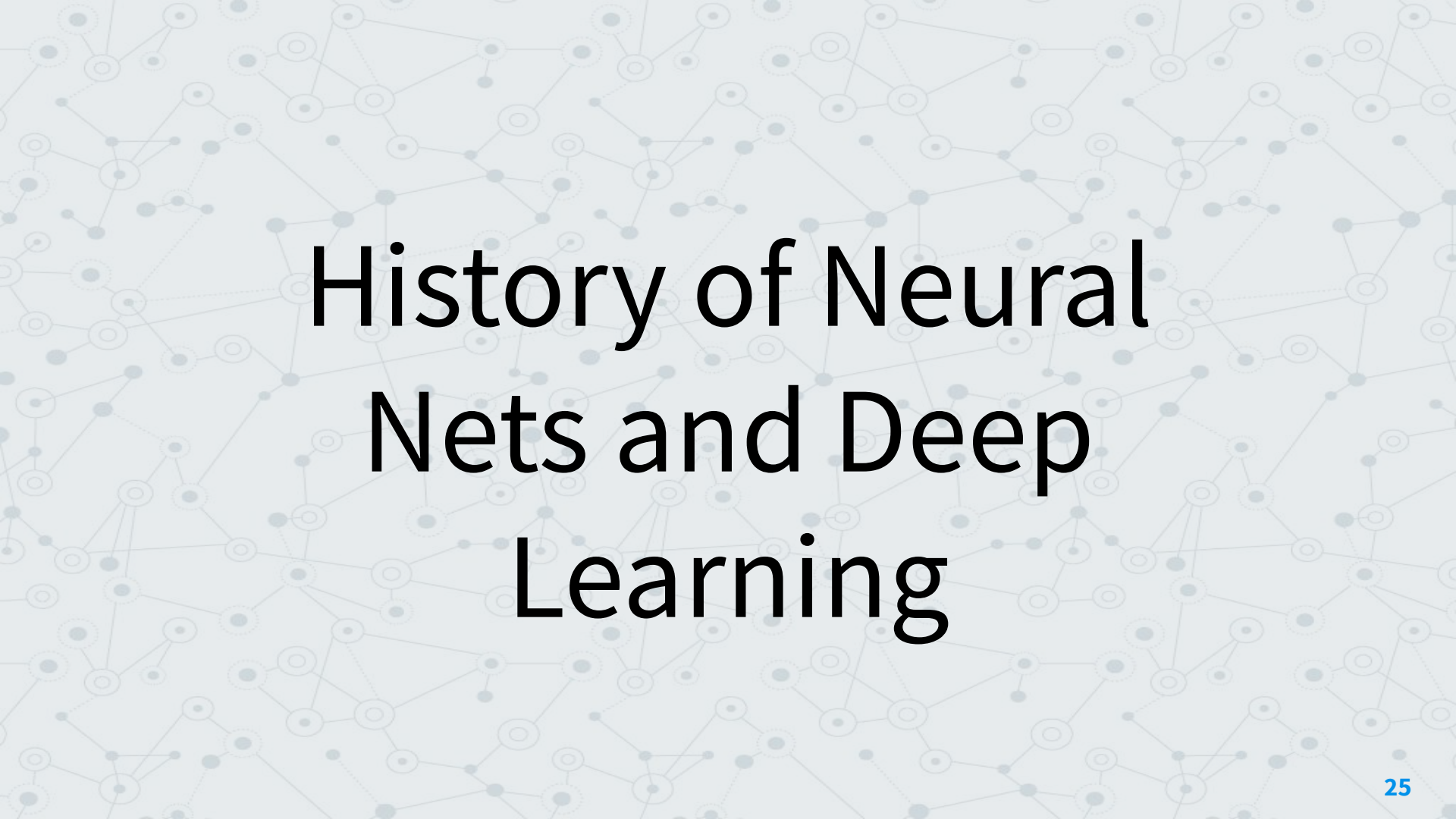


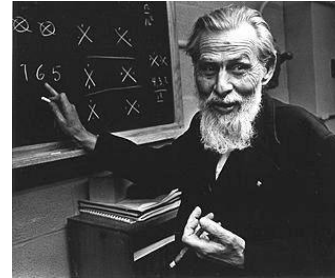
Figure 1. Our trained convolutional neural network correctly detecting the sinus rhythm (SINUS) and Atrial Fibrillation (AFIB) from this ECG recorded with a single-lead wearable heart monitor.



History of Neural Nets and Deep Learning

Neural Nets and Deep Learning Not New

- ◎ Date back to the 1940s
- ◎ Walter Pitts and Warren McCulloch
 - First notion of an artificial neuron
 - Designed to mimic the way a neuron was thought to work
 - [1943 paper](#)
- ◎ Frank Rosenblatt
 - “Perceptron” algorithm 1950s
 - Could recognize letters and numbers



AI Winter

- ◎ Many cycles of boom and bust
- ◎ Repeated promises of “true AI” that were unfulfilled and followed by “AI winters” - the first in 1969
- ◎ Marvin Minsky and Seymour Papert write book about shortcomings of perceptrons and effectively kill all research on neural nets

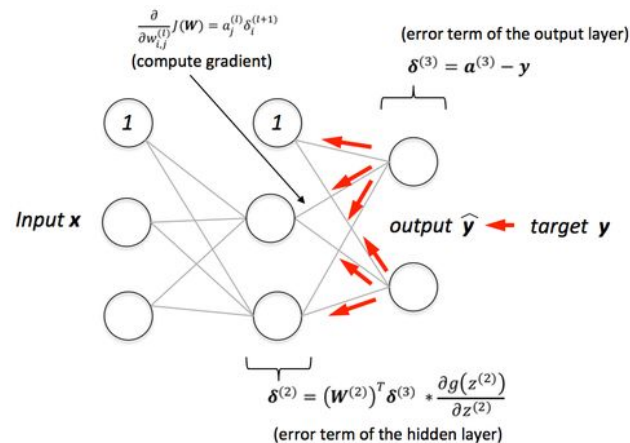
BRACE YOURSELVES

**AI WINTER IS
COMING**

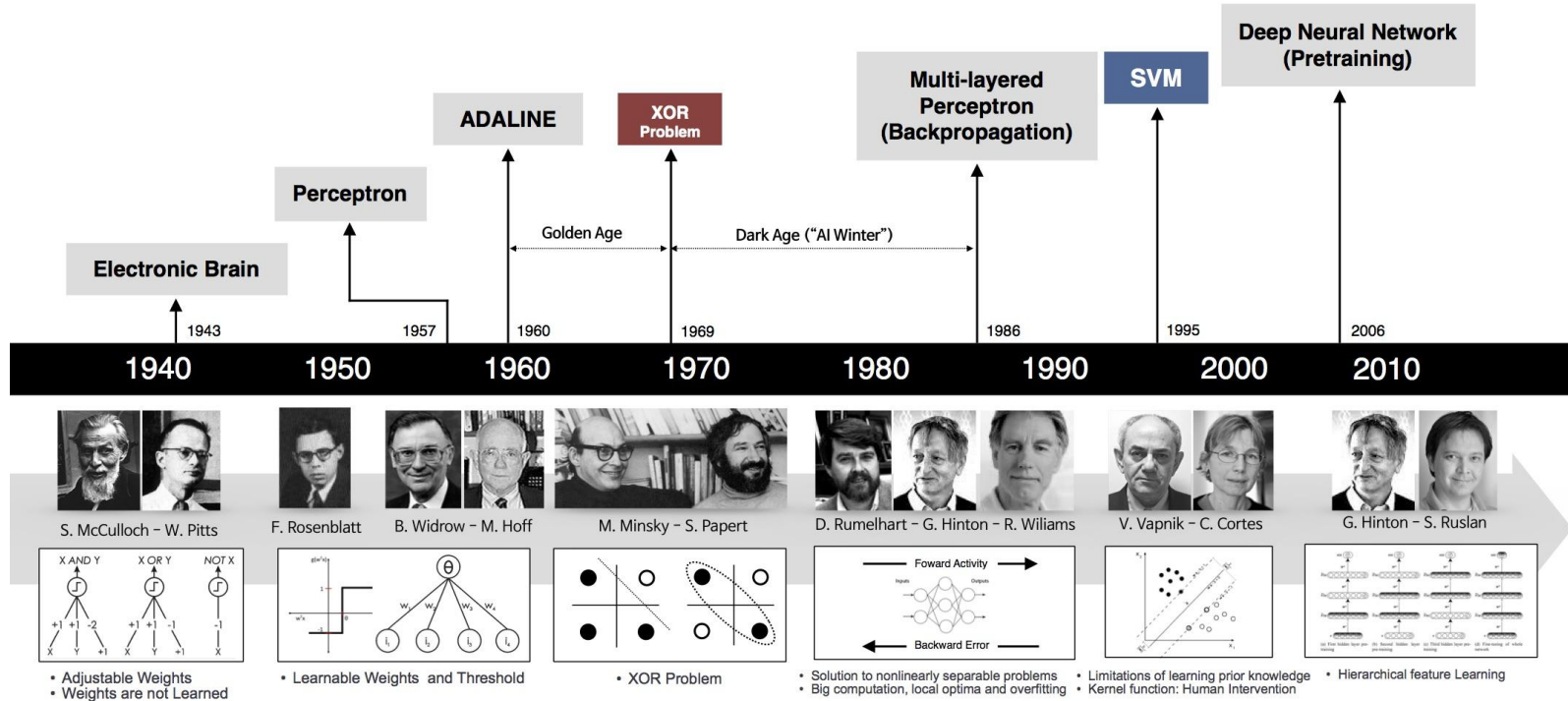
memegenerator.net

Return of the Neural Net

- Geoff Hinton, David Rumelhart and Ronald Williams discover back-propagation (1980s)
 - Allows neural nets to move past the limitations of perceptrons
 - Lead to convolutional neural nets (CNNs) and handwritten digits recognition
 - Problem: didn't scale \longrightarrow another 10-15 year AI winter
- Rebranding as “Deep Learning” (2006)
 - Unsupervised pretraining and deep belief networks
 - Could create “deeper” neural nets \longrightarrow “deep” learning
- Great AI Awakening (where we are now!)
 - Alexnet (2012)
 - Availability of GPUs (and TPUs) and larger data sets
 - Neural nets start surpassing humans



Deep Learning Timeline



Deep Learning Timeline

THE RISE OF AI

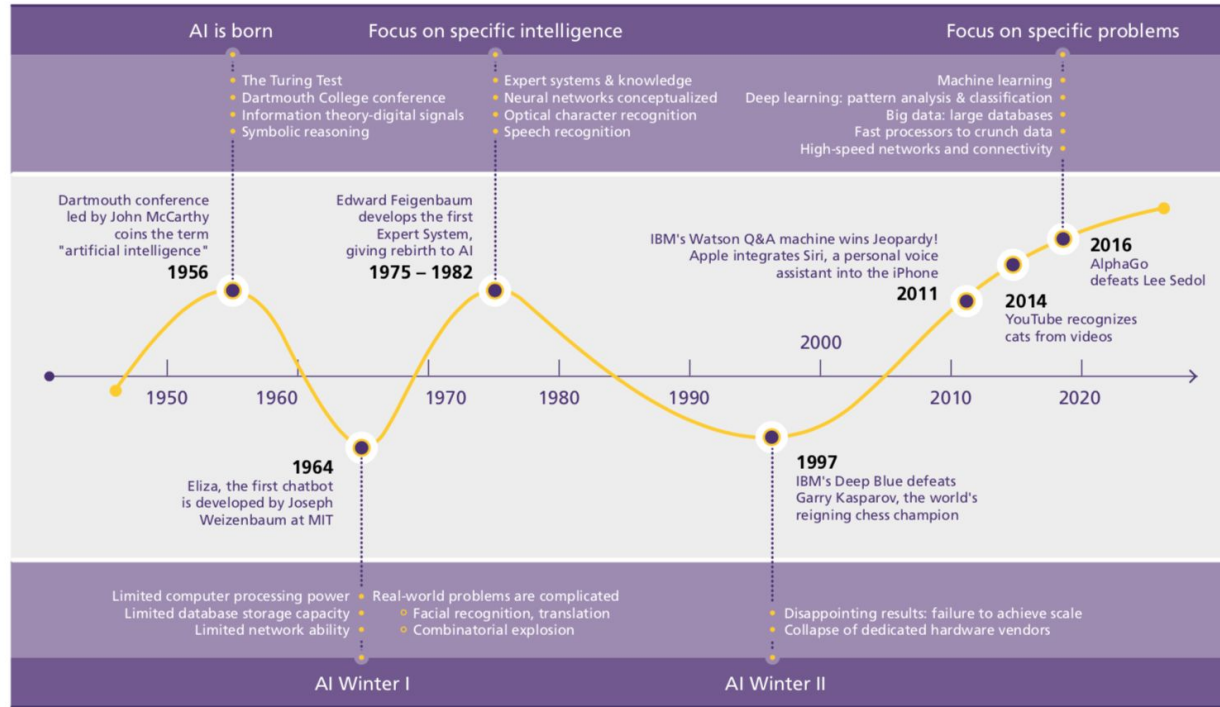
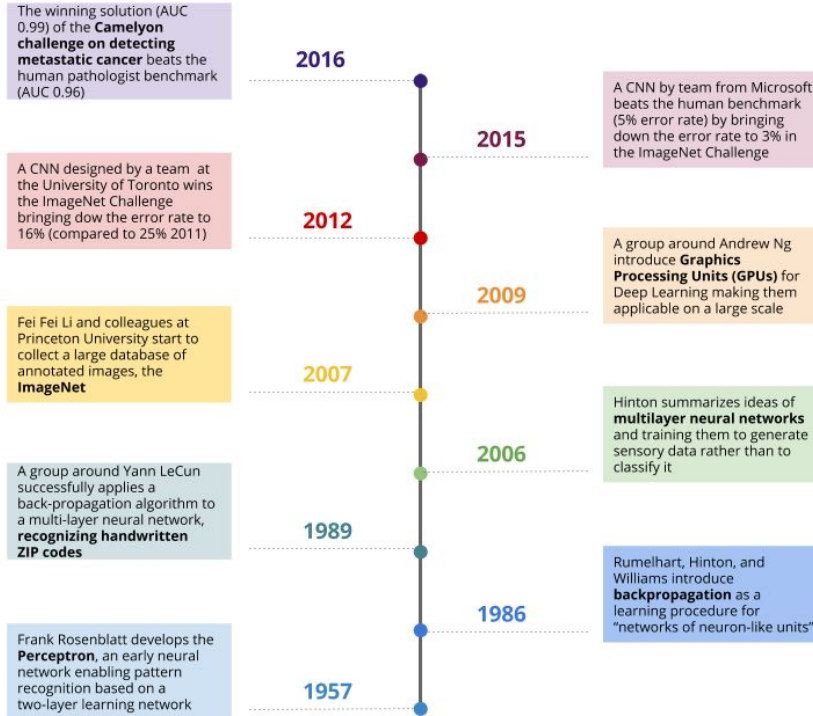


Figure 1: An AI timeline; Source: Lavenda, D./Marsden, P.

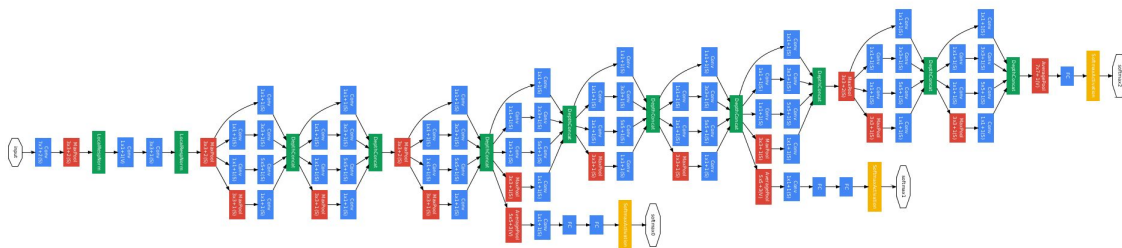
source dhl via @mikequindazzi

Deep Learning Timeline



AI winters are probably over

- ◎ We now have large, high-quality, labeled data sets
- ◎ GPUs and TPUs abound
 - Allows for deeper models and an increase in accuracy
- ◎ Improved functions needed for learning
 - ReLU
 - tanh
- ◎ Improved architectures
 - Resnets
 - Inception modules
- ◎ New regularization techniques
 - Dropout
 - Batch normalization
- ◎ Robust optimizers
- ◎ Software platforms
 - Tensorflow
 - Theano



How to stay current

- ◎ Advances in deep learning, and AI in general, are happening every day - it isn't possible to keep track of everything, but below are some good sources to check out
- ◎ Read papers on [arXiv](#)
- ◎ Subscribe to [Medium](#)
- ◎ [Google AI Blog](#)
- ◎ [Keras Blog](#)
- ◎ [OpenAI Blog](#)
- ◎ Twitter
 - Follow deep learning gods like Ian Goodfellow, Yann Lecun, Fei-Fei Li, Francois Chollet and our own HMS professor Andrew Beam
- ◎ [Talking machines podcast](#)

Takeaways

- ◎ Deep learning is real and probably here to stay
- ◎ Could potentially impact many fields → understand concepts so you have deep learning "insurance"
- ◎ Long history and connections to other models and fields
- ◎ Prereqs: Data (lots) + GPUs (more = better)
- ◎ Deep learning models are like legos, but you need to know what blocks you have and how they fit together
- ◎ Need to have a sense of sensible default parameter values to get started
- ◎ "Babysitting" the learning process is a skill

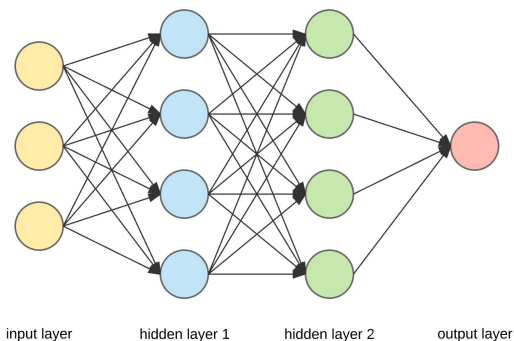


Neural Nets

What is a neural net?

A neural net is composed of 3 things:

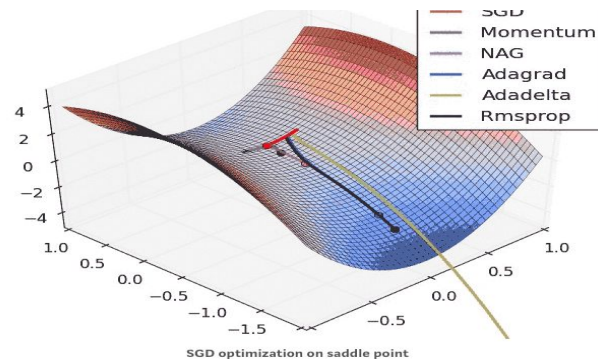
The network structure



The loss function

$$-y_i * \log(p_i) - (1 - y_i) * \log(1 - p_i)$$

The optimizer



<https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>

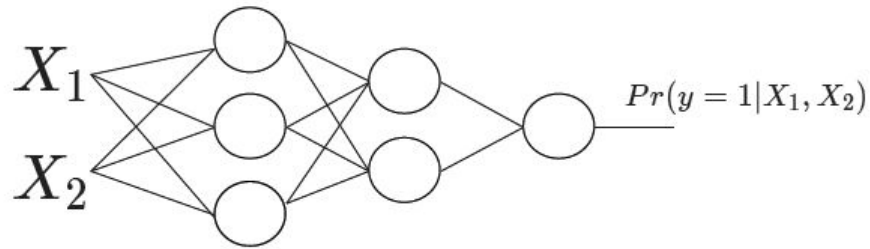
<https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6>

Neural Net Structure

- © A neural net is a modular way to build a classifier

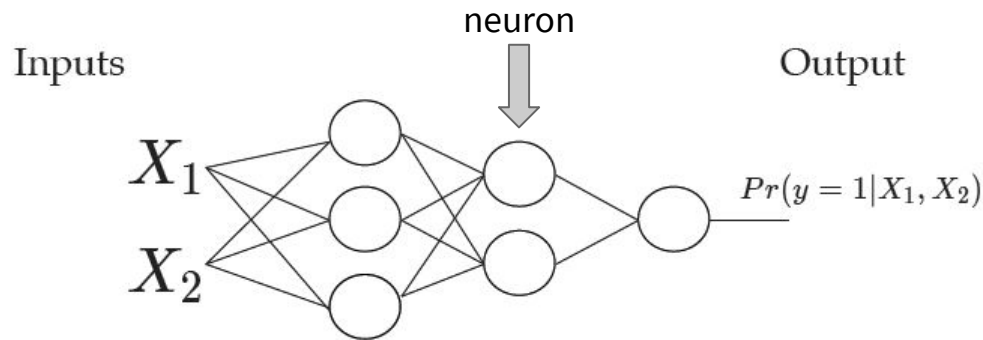
Inputs

Output



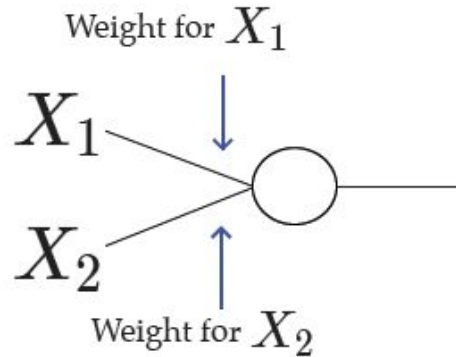
Neural Net Structure

- ⊙ A neural net is a modular way to build a classifier
- ⊙ The neuron is the basic functional unit in a neural network



Neurons

- ◎ A neuron does 2 things:



- 1) Weighted sum of inputs

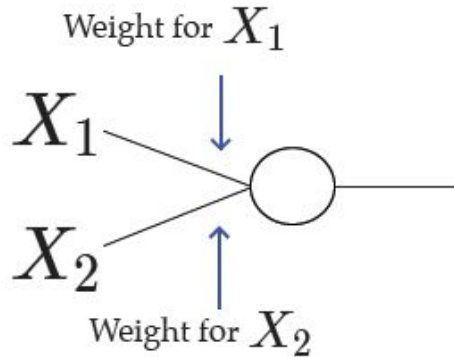
$$w_1 * X_1 + w_2 * X_2$$

- 2) Nonlinear transformation

$$\phi(w_1 * X_1 + w_2 * X_2)$$

Neurons

- ◎ A neuron does 2 things:



- 1) Weighted sum of inputs

$$w_1 * X_1 + w_2 * X_2$$

- 2) Nonlinear transformation

$$\phi(w_1 * X_1 + w_2 * X_2)$$

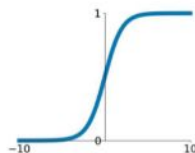


Activation function - a
non-linear transformation

Activation Functions

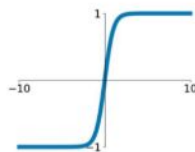
Sigmoid

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



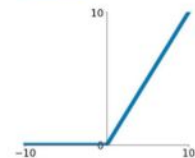
tanh

$$\tanh(x)$$



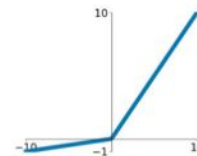
ReLU

$$\max(0, x)$$



Leaky ReLU

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

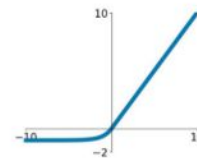


Maxout

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

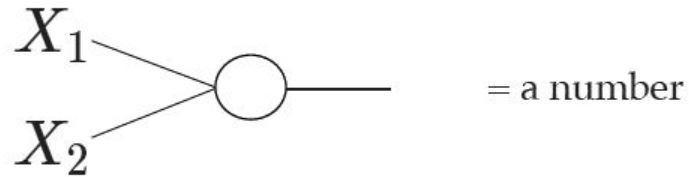
ELU

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



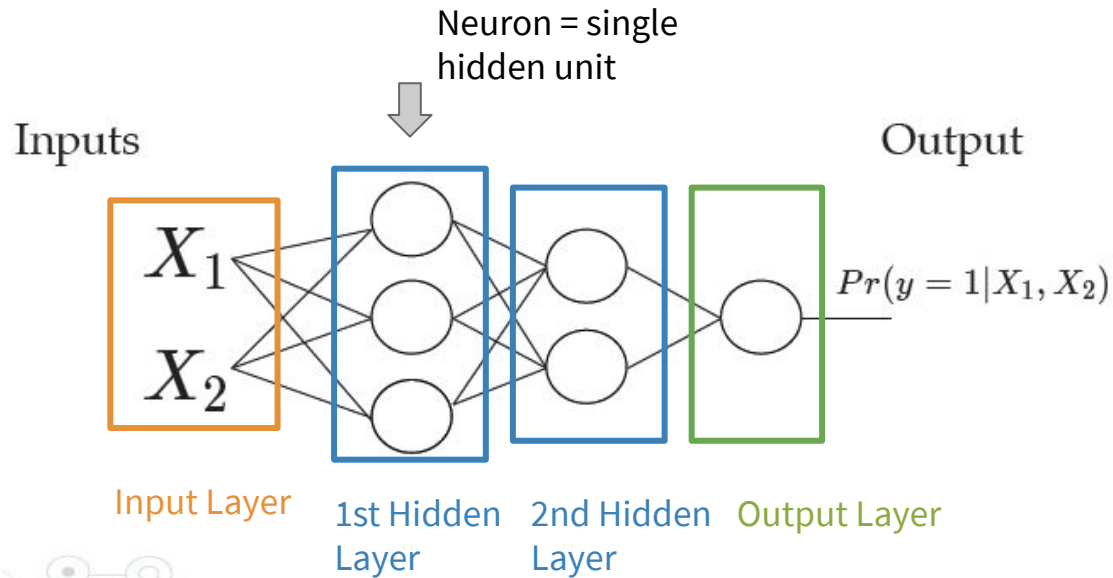
Neuron

- ◎ A neuron produces a single number that is a nonlinear transformation of its input connections



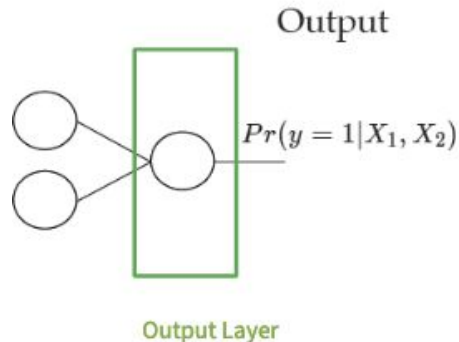
Neural Net Structure

- Neural nets are organized into **layers**



Loss Functions

- ◎ We need a way to measure how well the network is performing
 - Is it making good predictions?
- ◎ A **loss function** is a function that returns a single number which indicates how closely a prediction matches the ground truth, i.e. the actual label
 - We want to minimize the loss to achieve more accurate predictions
 - Also known as the objective function, cost function, loss, etc.



Loss Functions

One of the simplest loss functions is **binary cross-entropy** which is used for binary classification

y_i is the true label

$$p_i = P(y_i = 1 | X_1, X_2)$$

$$l(y_i, p_i) = -y_i * \log(p_i) - (1 - y_i) * \log(1 - p_i)$$

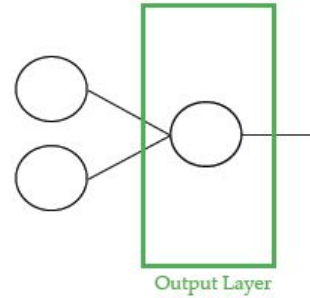
y	p	Loss
0	0.1	0.1
0	0.9	2.3
1	0.1	2.3
1	0.9	0.1

Output Layer and Loss

- ◎ The output layer needs to “match” the loss function
- ◎ Correct shape
- ◎ Correct scale
- ◎ For binary cross-entropy, the network needs to produce a single probability:

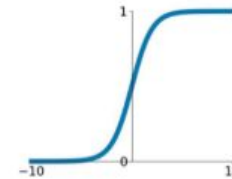
$$p_i = P(y_i = 1 | X_1, X_2)$$

- ◎ One unit in the output layer represents this probability
- ◎ Activation function must “squash” the output to be between 0 and 1



Sigmoid

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



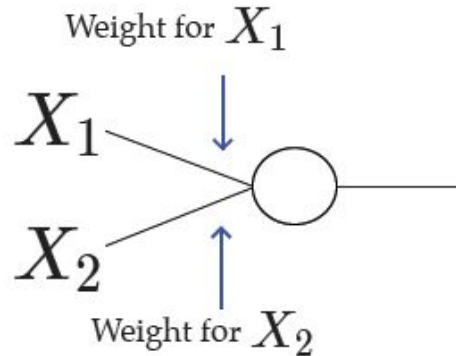
Output Layer and Loss

- ◎ We can change the output layer and loss to model many different kinds of data

Task	Last-layer activation	Loss function
Binary classification	sigmoid	Binary cross-entropy
Multiclass, single-label classification	softmax	Categorical cross-entropy
Multiclass, multilabel classification	sigmoid	Binary cross-entropy
Regression to arbitrary values	None	Mean square error (MSE)
Regression to values between 0 and 1	sigmoid	MSE or binary cross-entropy

The Optimizer

Remember this?



1) Weighted sum of inputs

$$w_1 * X_1 + w_2 * X_2$$

2) Nonlinear transformation

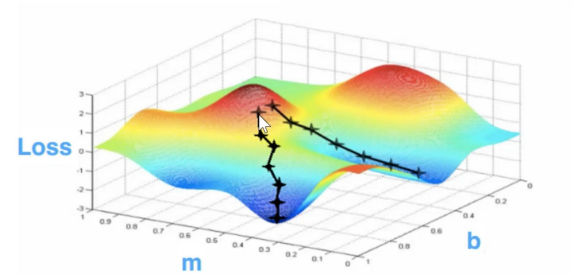
$$\phi(w_1 * X_1 + w_2 * X_2)$$

We have specified the network and the loss function, but what about values for the weights? We want weights that minimize the loss function - how do we calculate them?

The Optimizer

Gradient Descent

$f(x) = \text{nonlinear function of } x$



Q: How do we minimize the loss function?

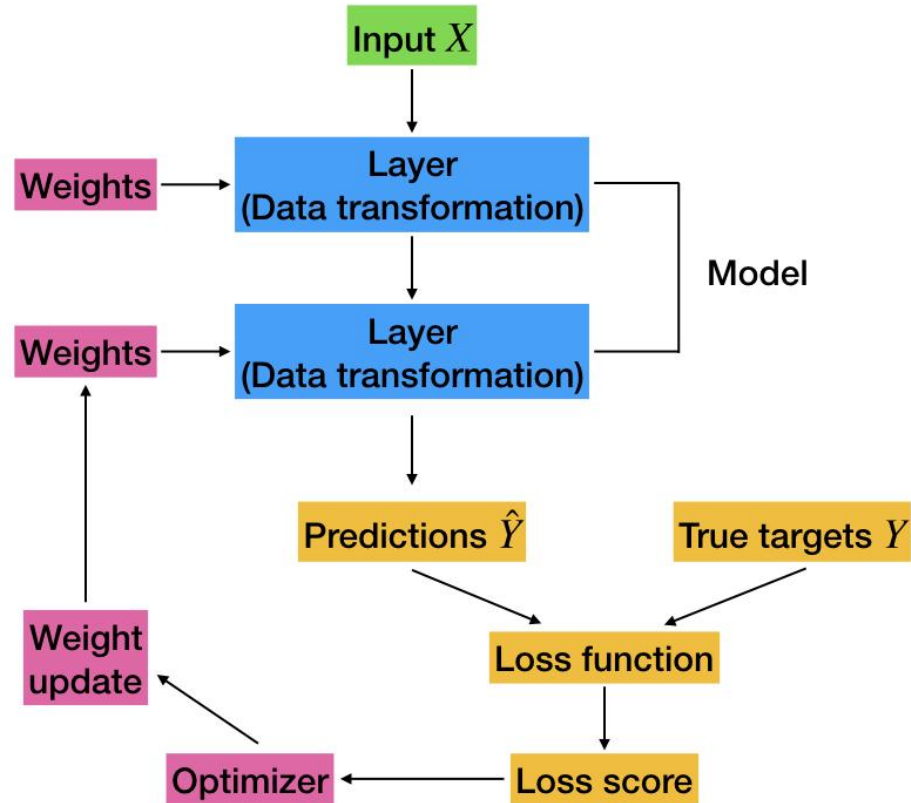
A: Stochastic Gradient Descent (SGD)

1. Give weights random initial values
2. Evaluate the partial derivative of each weight with respect to the negative log-likelihood at the current weight value on a mini-batch
3. Take a step in the direction opposite the gradient
4. Repeat

The Optimizer

- ◎ Many variations on the basic idea of SGD are available
 - Rmsprop
 - Adagrad
 - Adadelata
 - Momentum
 - NAG
 - etc.

Workflow



Action Items

- ◎ Review linear algebra and Python review slides if you're feeling rusty
 - Available on the course Canvas and GitHub repo
 - Reviewing Calculus would also be useful if you're feeling rusty about how to derive partial derivatives (they will show up in Problem Set #1)
- ◎ Sign up for a paper to present
- ◎ Find group members for the group project proposal
 - Can also work individually