Lecture 1: Introduction

Lan Xu
SIST, ShanghaiTech
Fall, 2023

Outline

- Course logistics
  - Overall objective
  - Grading policy
  - Pre-requisite / Syllabus
- Introduction to deep learning
- Machine learning review
- Artificial neurons

Course objectives

- Learning to use deep networks
  - How to write from scratch, debug and train neural networks
  - Toolboxes commonly used in practice
- Understanding deep models
  - Key concepts and principles
- State of the art
  - Some new topics from research field
  - Focusing on vision-related problems

Syllabus & Schedule

- Piazza:
  - https://piazza.com/shanghaitech.edu.cn/fall2023/cs280_2023
  - The schedule for the latter half of the semester may vary a bit
- Part I: Basic neural networks (1 weeks)
  - Linear models; Multiple layer networks; Gradient descent and BP
- Part II: CNN and RNN
- Part III: Transformer
- Part IV: Neural Prediction Applications
- Part V: Generative Models
- Part VI: Advanced Topics

Syllabus & Schedule

- Piazza:
  - https://piazza.com/shanghaitech.edu.cn/fall2023/cs280_2023
  - The schedule for the latter half of the semester may vary a bit
- Part I: Basic neural networks (1 weeks)
- Part II: CNN and RNN(2 weeks)
  - CNN basics; Understanding CNN; Training; Regularization
  - Sequence modeling; LSTM, GRU； Attention modeling
- Part III: Transformer
- Part IV: Neural Prediction Applications
- Part V: Generative Models
- Part VI: Advanced Topics

Syllabus & Schedule

- Piazza:
  - https://piazza.com/shanghaitech.edu.cn/fall2023/cs280_2023
  - The schedule for the latter half of the semester may vary a bit
- Part I: Basic neural networks (1 weeks)
- Part II: CNN and RNN(2 weeks)
- Part III: Transformer (2 weeks)
  - Transformer Architecture
  - Transformer Application
- Part IV: Neural Prediction Applications
- Part V: Generative Models
- Part VI: Advanced Topics

Syllabus & Schedule

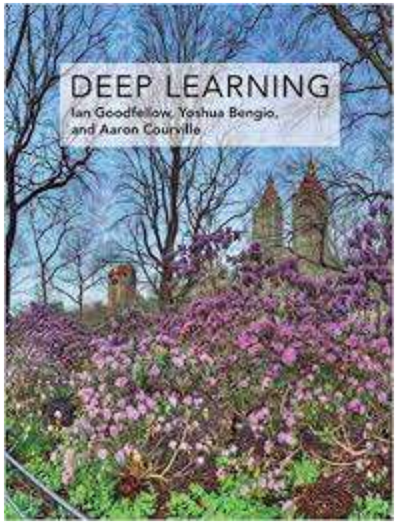
- Piazza:
  - https://piazza.com/shanghaitech.edu.cn/fall2023/cs280_2023
  - The schedule for the latter half of the semester may vary a bit
- Part I: Basic neural networks (1 weeks)
- Part II: CNN and RNN(2 weeks)
- Part III: Transformer (2 weeks)
- Part IV: Neural Prediction Applications (2 weeks)
  - Prediction task and VLM
  - Prediction task and LLM
- Part V: Generative Models
- Part VI: Advanced Topics
- Note: no lectures in the following weeks
  - Nov 7 ~ Nov 14 (CVPR)

Syllabus & Schedule

- Piazza:
  - https://piazza.com/shanghaitech.edu.cn/fall2023/cs280_2023
  - The schedule for the latter half of the semester may vary a bit
- Part I: Basic neural networks (1 weeks)
- Part II: CNN and RNN(2 weeks)
- Part III: Transformer (2 weeks)
- Part IV: Neural Prediction Applications (2 weeks)
- Part V: Generative Models (2.5 weeks)
  - Variational Auto Encoder (VAE); Generative deep nets (GAN)
  - Diffusion model; Multi-modal Generation
- Part V: Advanced Topics (1.5 weeks)
- Note: no lectures in the following weeks
  - Nov 1 ~ Nov 8 (CVPR)

Reference books and materials

- Deep learning:
  - http://www.deeplearningbook.org/
  - https://d2l.ai/



- Online deep learning courses:
  - Stanford: CS230, CS231n
  - CMU: 11-785
  - MIT: 6.S191

- Additional reading materials on Piazza
  - Survey papers, tutorials, etc.

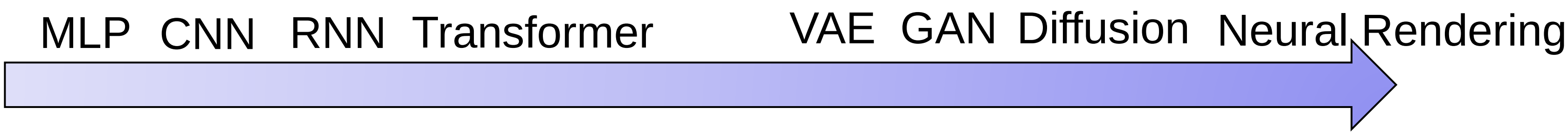
Instructor and TAs

- Instructor: Prof Lan Xu
  - xulan1@shanghaitech.edu.cn
  - SIST 1C-303D
- TAs:
  - Han Liang, Suyi Jiang, Chenfeng Zhao, Wenqian Zhang, Bo Yang
- Office hours: To be announced on Piazza
- We will use Piazza as the main communication platform

Grading policy

- 4 Problem sets: 10% x 3 = 30%
  - Write-up problem sets + Programming tasks
- Final course project: 50% (+5%)
  - Proposal
  - Final report (Conference format)
  - Presentation, Poster, Suplementary Materials
  - Bonus points for impressive and novel results: 5%
- 5 Quizzes (in class): 4% x 5 = 20%
- Late policy
  - A total of 7 free late (calendar) days to use, but no more than 4 late days can be used on any single assignment.
  - After that, 25% off per day late
  - Does not apply to Final course project/Quizzes
- Collaboration policy
  - Project team: 4~5 students
  - Grading according to each member's contribution

Course Roadmap



- Quiz (20%), HW (30%), Project (50%)
- More Important: relate the course to your research

上海科技大学2023-2024学年校历

	八月		九月				十月				十一月					十二月				一月			
星期一	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27	4	11	18	25	1 元旦	8	15	22
星期二	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23
星期三	23	30	6	13	20	27	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24
星期四	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25
星期五	25	1	8	15	22	29 中秋节	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12	19	26
星期六	26	2	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27
星期日	27	3	10	17	24	1 国庆节	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28
周数	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
学期	暑假					秋学期																	

Administrative Stuff

- Plagiarism
  - All assignments must be done individually
    - You may not look at solutions from any other source
    - You may not share solutions with any other students
    - Plagiarism detection software will be used on all the programming assignments
    - You may discuss together or help another student but you cannot give the exact solution
- Plagiarism punishment
  - When one student copies from another student, both students are responsible
  - Zero point on the assignment or exam in question
  - Repeated violation will result in an F grade for this course as well as further discipline at the school/university level

Pre-requisite

- Proficiency in Python
  - All class assignments will be in Python (and use numpy)
  - A Python tutorial available on Piazza
- Calculus, Linear Algebra, Probability and Statistics
  - Undergrad course level
- Equivalent knowledge of Andrew Ng's CS229 (Machine Learning)
  - Formulating cost functions
  - Taking derivatives
  - Performing optimization with gradient descent

Outline

- Course logistics
- Introduction to deep learning
  - What & Why deep learning?
- Machine Learning review
- Artificial neurons

Acknowledgement: Bhiksha Raj@CMU's course notes

Introduction

- Our goal: Build intelligent algorithms to make sense of data
  - Example: Recognizing objects in images

red panda (*Ailurus fulgens*)

  - Example: Predicting what would happen next

Vondrick et al. CVPR2016

Introduction

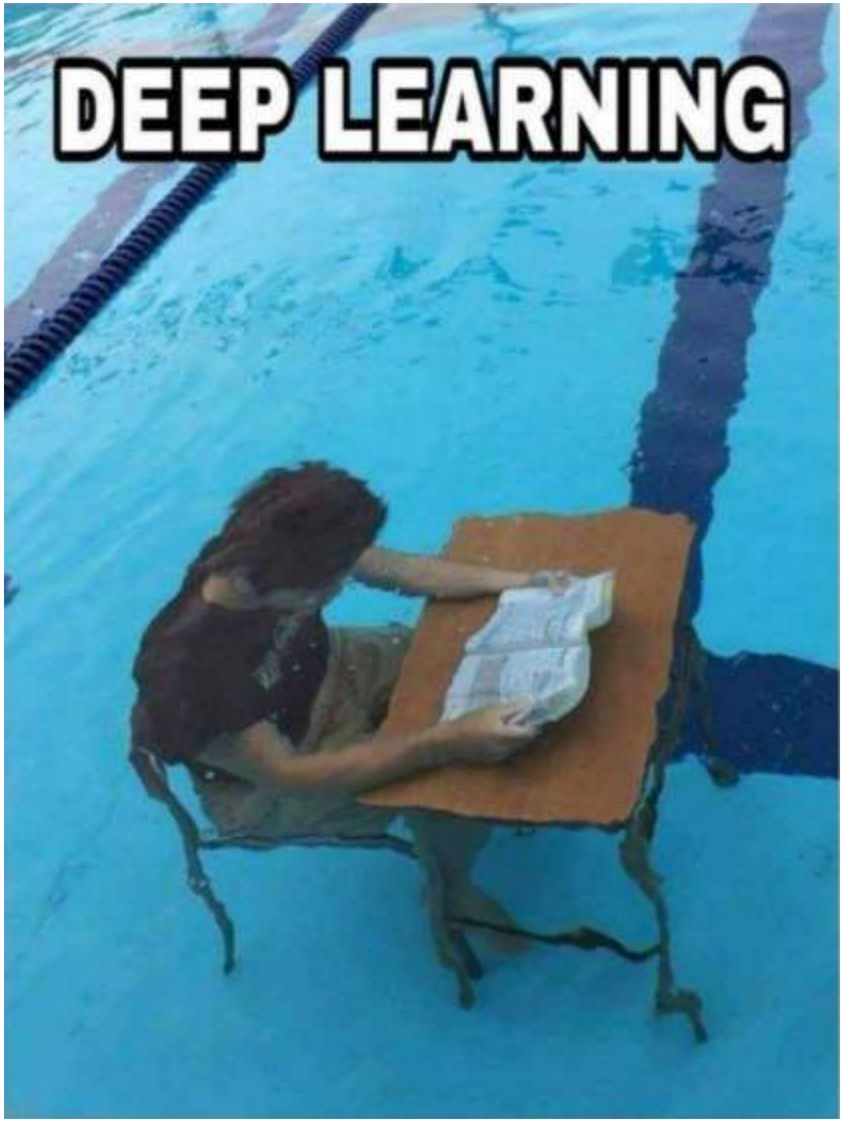
- Our goal: Build intelligent algorithms to make sense of data
  - Example: Recognizing objects in images
  - Example: Predicting what would happen next

Given an initial still frame,

Introduction

- A broad range of real-world applications
  - Speech recognition
    - Input: sound wave → Output: transcript
  - Language translation
    - Input: text in language A (Eng) → Output: text in language B (Chs)
  - Image classification
    - Input: images → Output: image category (cat, dog, car, house, etc.)
  - Autonomous driving
    - Input: sensory inputs → Output: actions (straight, left, right, stop, etc.)
- Main challenges: difficult to manually design the algorithms

A data-driven approach



A data-driven approach

A data-driven approach

- Each task as a mapping function (or a model)

Input data $\xrightarrow{\text{Mapping function}}$ Expected output

  - input data: images
  - expected output: object or action names

- Building such mapping functions from data

$\longrightarrow$ Mapping function

red panda (*Ailurus fulgens*)

A data-driven approach

- Building a mapping function (model)

$$y = f(x;\theta)$$

  - x: input data
  - y: expected output
  - θ: parameters to be estimated

- Learning the model from data
  - Given a dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$
  - Find the 'best' parameter $\hat{\theta}$, such that

$$y_n \simeq f(x_n;\hat{\theta}) \quad \forall n$$

  - And it can be generalized to unseen input data

What is deep learning?

- Using deep neural networks as the mapping function

- Model: Deep neural networks
  - A family of parametric models
  - Consisting of many ‘simple’ computational units
  - Constructing a multi-layer representation of input

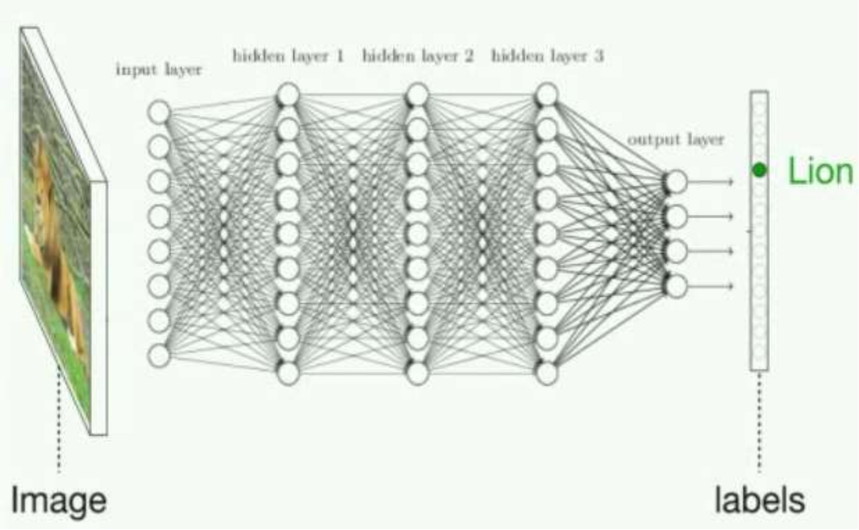
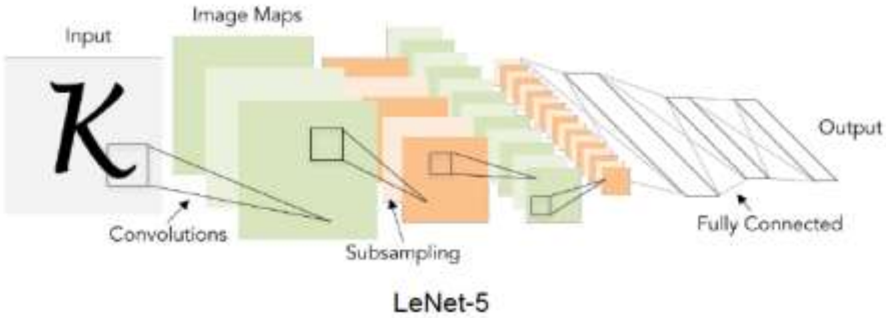


Image from Jeff Clune’s Deep Learning Overview

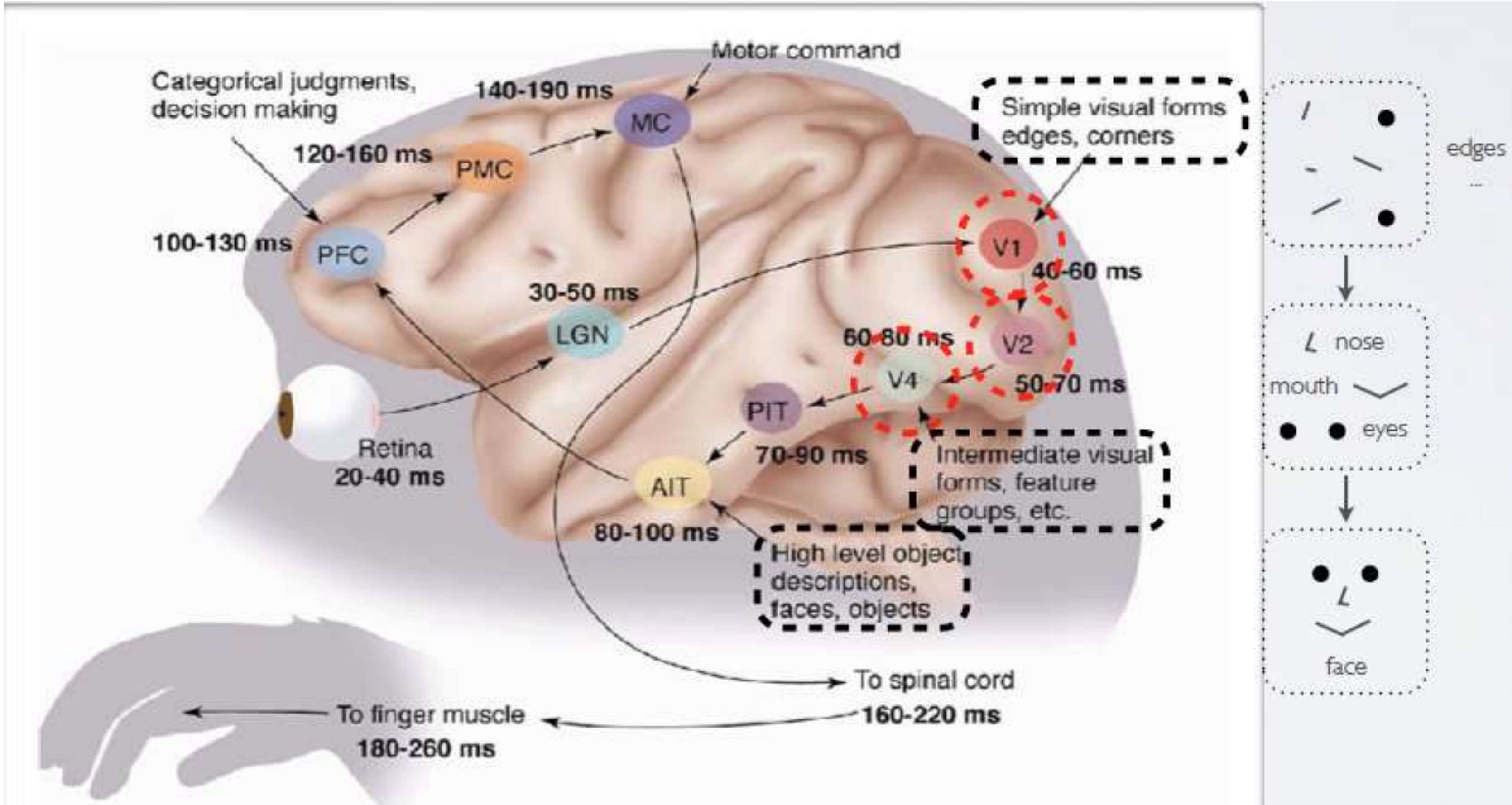
What is deep learning?

- Using deep neural networks as the mapping function
- Learning: Parameter estimation from data
  - Parameters: connection weights between units
  - Formulated as an optimization problem
  - Efficient algorithms for handling large-scale models & datasets



Why deep networks?

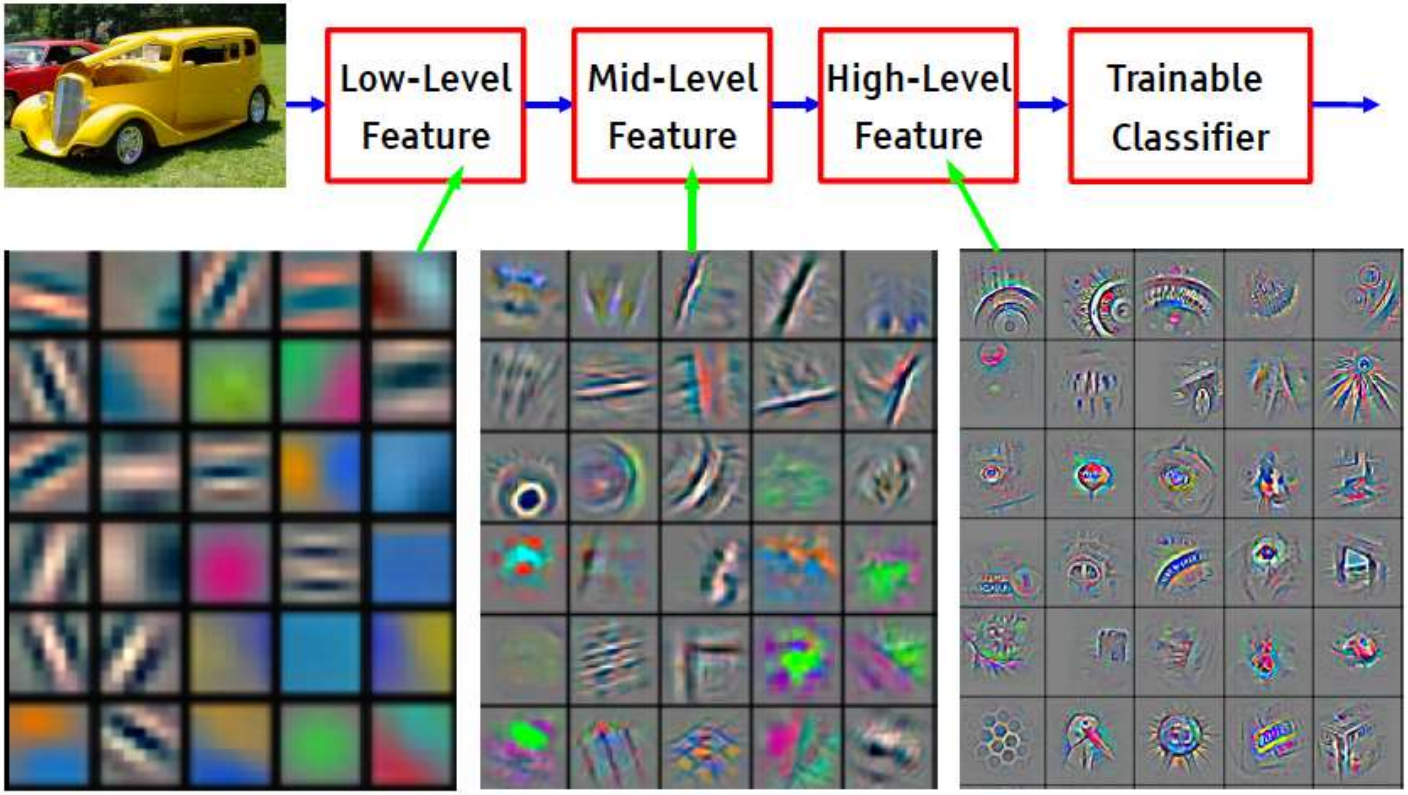
- Inspiration from visual cortex



[picture from Simon Thorpe]

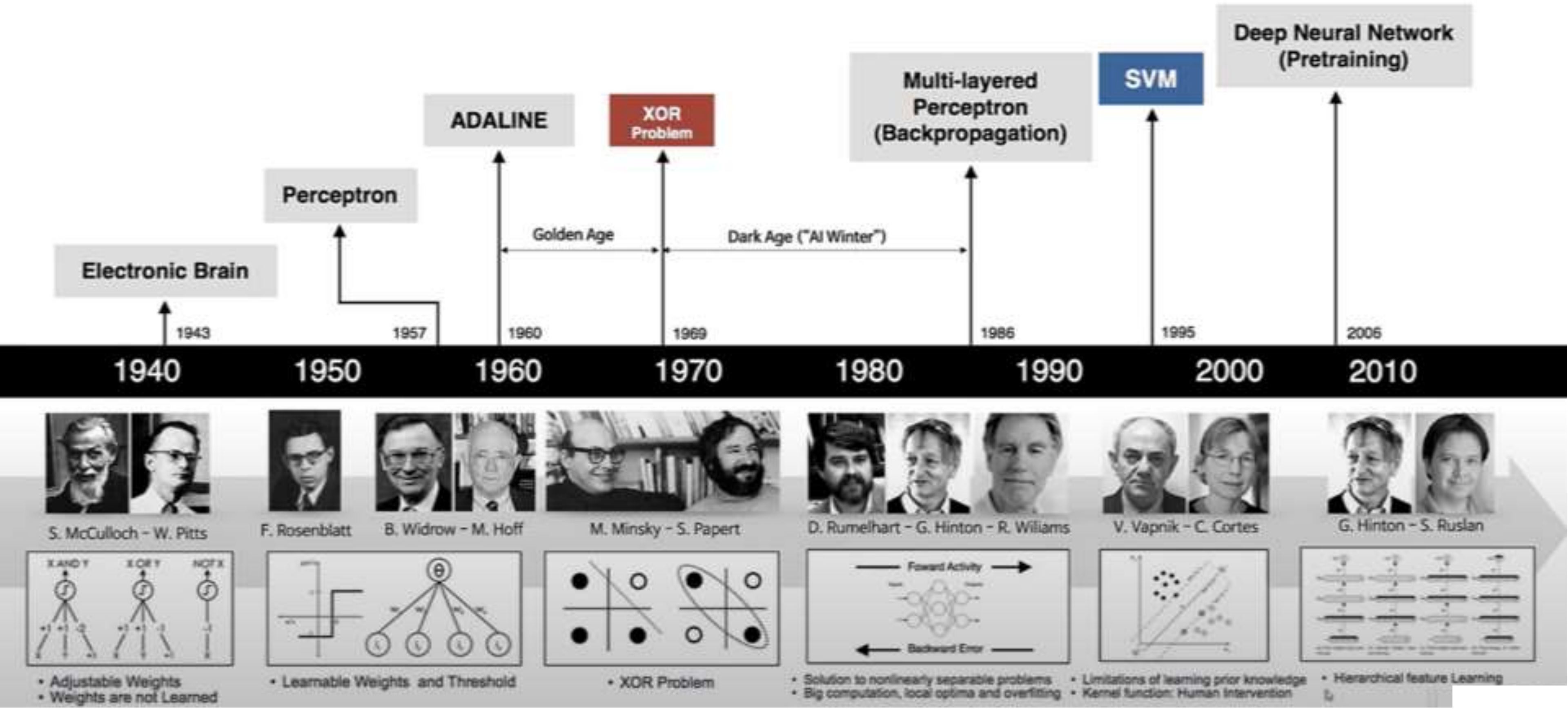
Why deep networks?

- A deep architecture can represent certain functions (exponentially) more compactly
- Learning a rich representation of input data



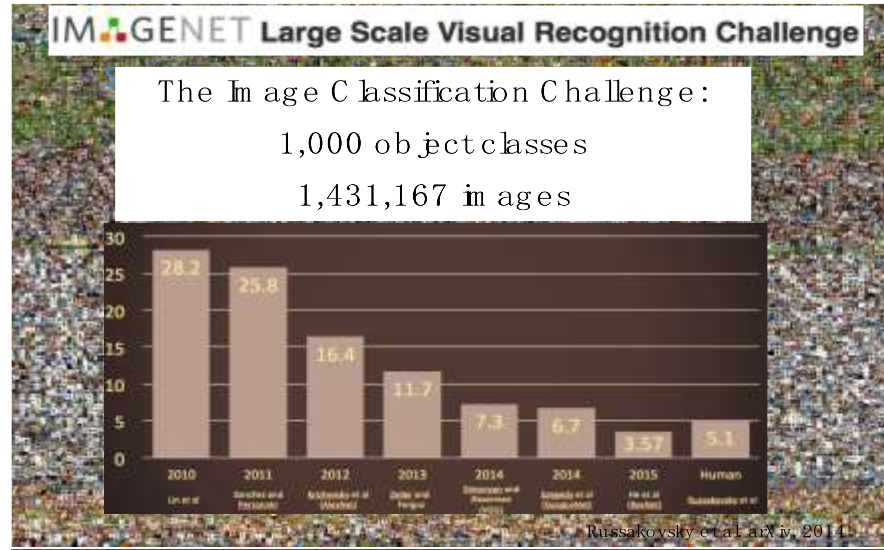
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Deep Learning History



Recent success with DL

- Some recent success with neural networks



Recent success with DL

- Some recent success with neural networks

ACM Turing Award 2019 (Nobel Prize of Computing)
Yann LeCun, Geoffrey Hinton, and Yoshua Bengio

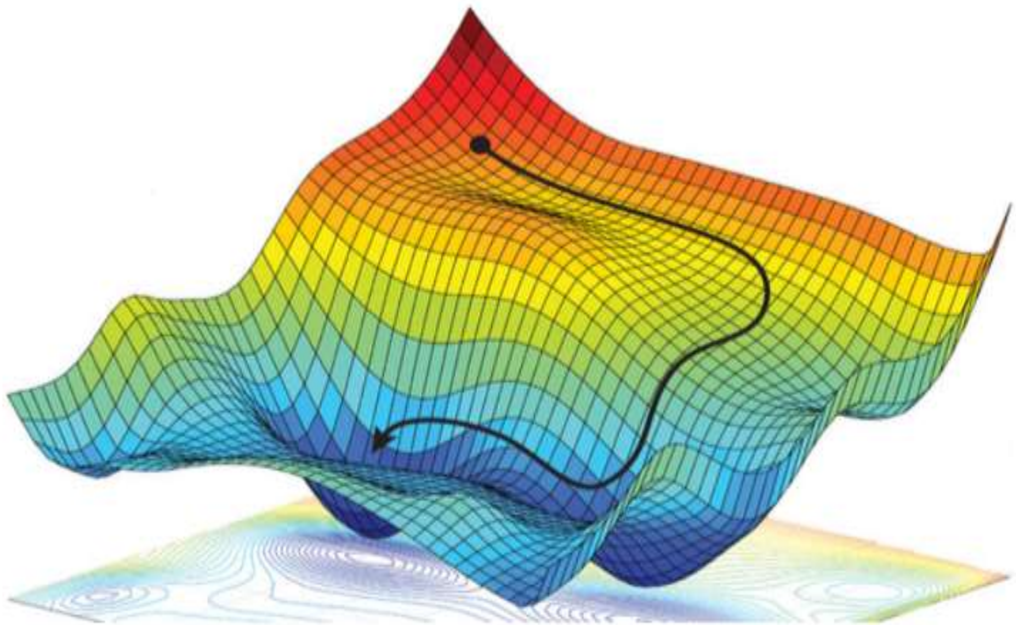
Recent success with DL

- Some recent success with neural networks

This guy didn't know about neural networks

This guy learned about neural networks

Is it alchemy?

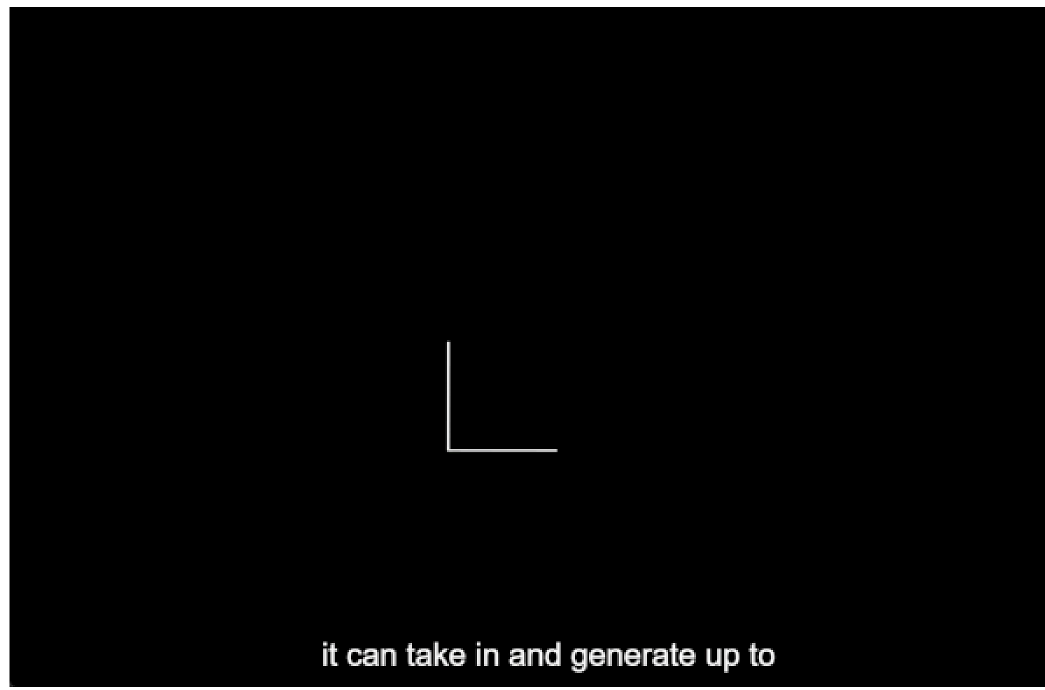


Gradient descent relies on trial and error to optimize an algorithm, aiming for minima in a 3D landscape.
ALEXANDER AMINI, DANIELA RUS. MASSACHUSETTS INSTITUTE OF TECHNOLOGY, ADAPTED BY M. ATAROD/*SCIENCE*

AI researchers allege that machine learning is alchemy

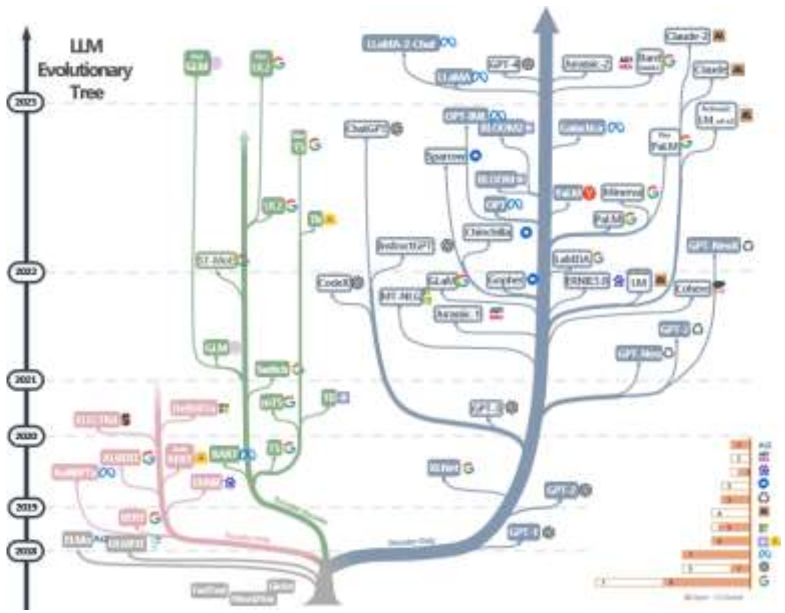
Recent success with “Modern” DL

- Modern Generative AI, e.g., GPT4
- High scores for SAT math, writing, etc.



Modern DL and AI

- Huge family tree of LLM
- Impressive generation ability with multi-modality

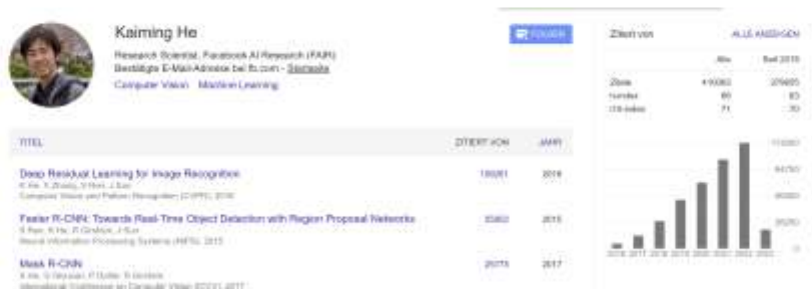


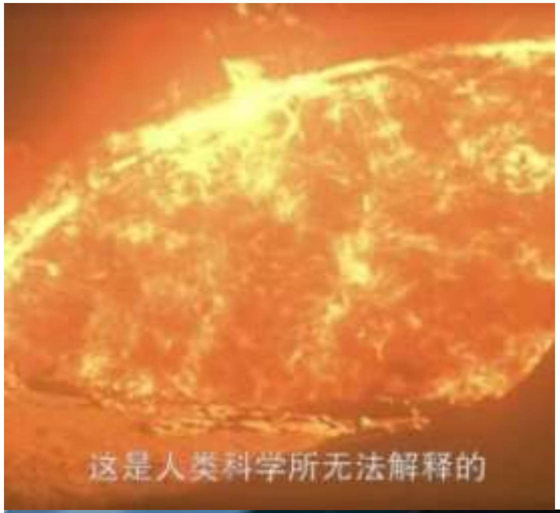
Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond

DALL-E 3 integrates with **ChatGPT** so users don't have to think of prompts anymore

Modern DL and AI

- AI for Science? Unified self-supervised X+AI





Modern DL and AI

- The party is over? NO!
- New paradigm; New opportunity; Super open-minded

……

Summary: Why deep learning?

- One of the major thrust areas recently in various pattern recognition, prediction and data analysis
  - Efficient representation of data and computation
  - Other key factors: large datasets and hardware

- The state of the art in many problems
  - Often exceeding previous benchmarks by large margins
  - Achieve better performances than human for certain “complex” tasks.

- But also somewhat controversial …
  - Lack of theoretical understanding
  - Sometimes difficult to make it work in practice

Questions to ask

- Understanding neural networks
  - What is different from traditional ML methods?
  - How it works for specific problems?
  - Why get great performance?
- Future development
  - Its limitation and weakness?
  - After more than 10 years, what is on-going or next?
  - The road to general-purpose AI?

Outline

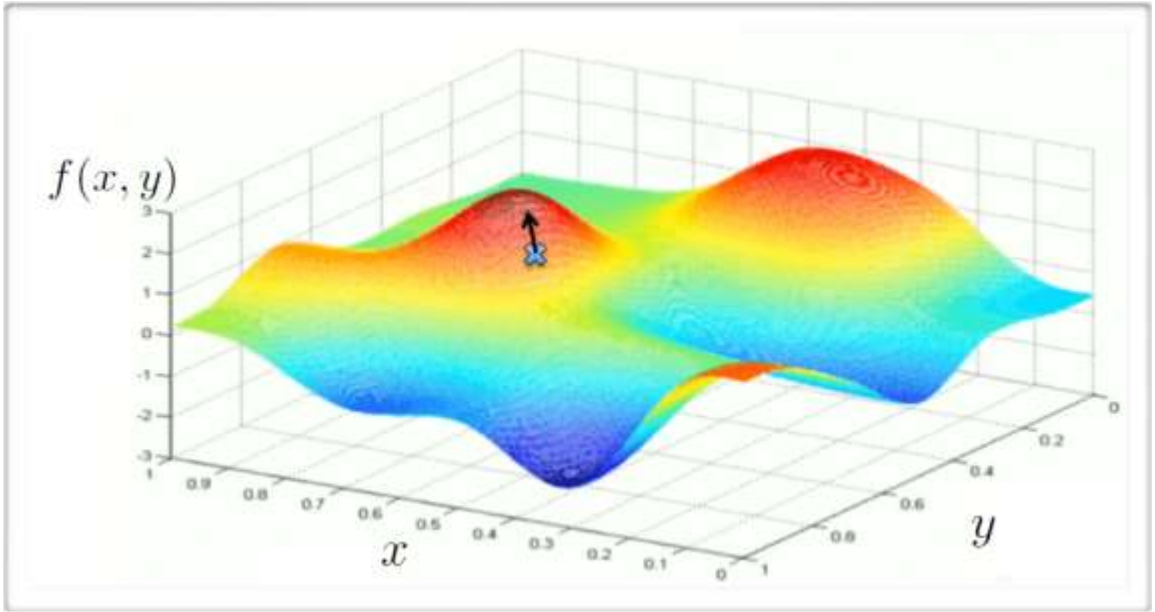
- Course logistics
- Introduction to deep learning
- Machine learning review
  - Math review
  - Supervised learning
- Artificial neurons

Acknowledgement: Hugo Larochelle's, Mehryar Mohri@NYU's & Yingyu Liang@Princeton's course notes

Math review – Calculus

- Gradient

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = [\frac{\partial}{\partial x_1} f(\mathbf{x}), \cdots, \frac{\partial}{\partial x_d} f(\mathbf{x})]^\intercal = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_d} f(\mathbf{x}) \end{bmatrix}$$



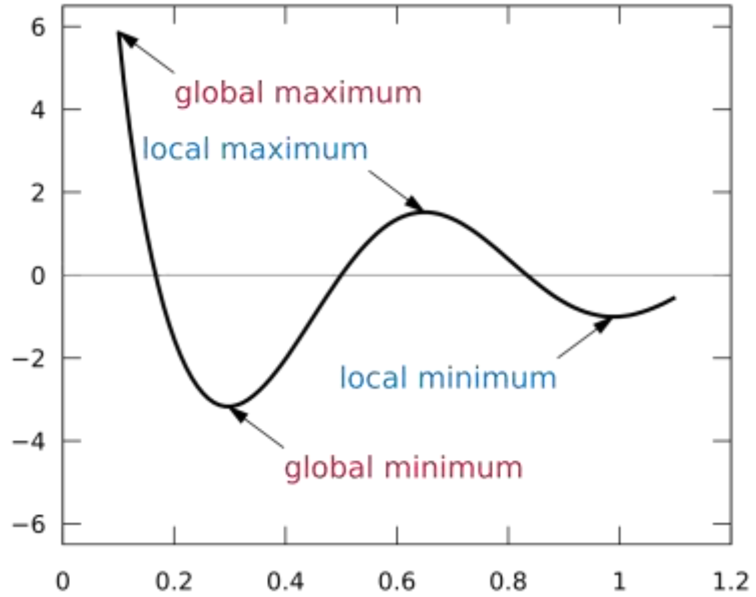
Math review – Calculus

- Local and global minima
  - Necessary condition

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

  - Sufficient condition
    - Hessian is positive definite

$$f(\mathbf{x}) \approx f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\intercal \nabla_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\intercal \nabla_{\mathbf{x}}^2 f(\mathbf{x})(\mathbf{x} - \mathbf{x}^*)$$



Math review – Probability

- Factorization

  - Probability chain rule: $p(s, o) = p(s|o)p(o) = p(o|s)p(s)$
    - in general:

$$p(\mathbf{x}) = \prod_i p(x_i | x_1, \ldots, x_{i-1})$$

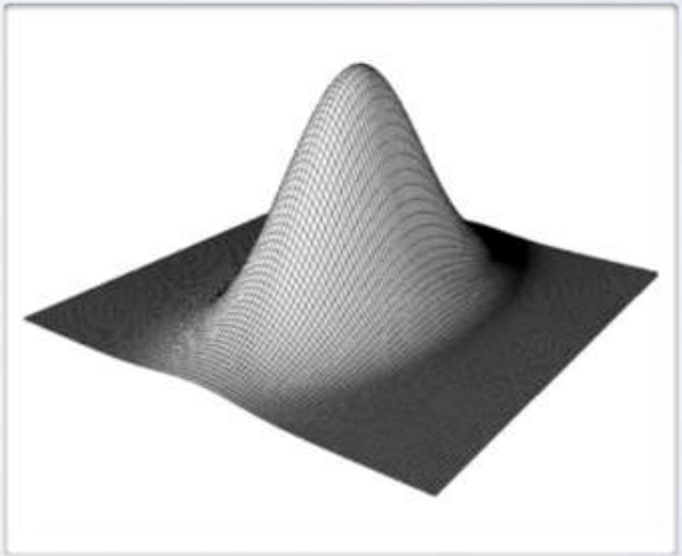
  - Bayes rule:

$$p(O = o | S = s) = \frac{p(S{=}s | O{=}o) p(O{=}o)}{\sum_{o'} p(S{=}s | O{=}o') p(O{=}o')}$$

Math review – Probability

- Common distributions

  - Gaussian variable: $\mathbf{X} \in \mathbb{R}^d$
    - $p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$
    - $\mathrm{E}[\mathbf{X}] = \boldsymbol{\mu}$
    - $\mathrm{Cov}[\mathbf{X}] = \Sigma$



Math review – Statistics

- Monte Carlo estimation
  - a method to approximate an expensive expectation

$$\mathrm{E}[f(\mathbf{X})] = \sum_{\mathbf{x}} f(\mathbf{x})\, p(\mathbf{x}) \approx \frac{1}{K} \sum_k f(\mathbf{x}^{(k)})$$

  - the $\mathbf{x}^{(k)}$ must be sampled from $p(\mathbf{x})$

- Maximum likelihood

$$\widehat{\theta} = \arg\max_{\theta} p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)})$$

  - Independent and identically distributed

$$p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}) = \prod_t p(\mathbf{x}^{(t)})$$

ML tasks

- Classification: assign a category to each item (e.g., document classification)
- Regression: predict a real value for each item (e.g., prediction of stock values, economic variables)
- Ranking: order items according to some criterion (e.g., relevant web pages returned by a search engine)
- Clustering: partition data into ‘homogenous’ regions (e.g., analysis of very large data sets)
- Dimensionality reduction: find lower-dimensional manifold preserving some properties of the data

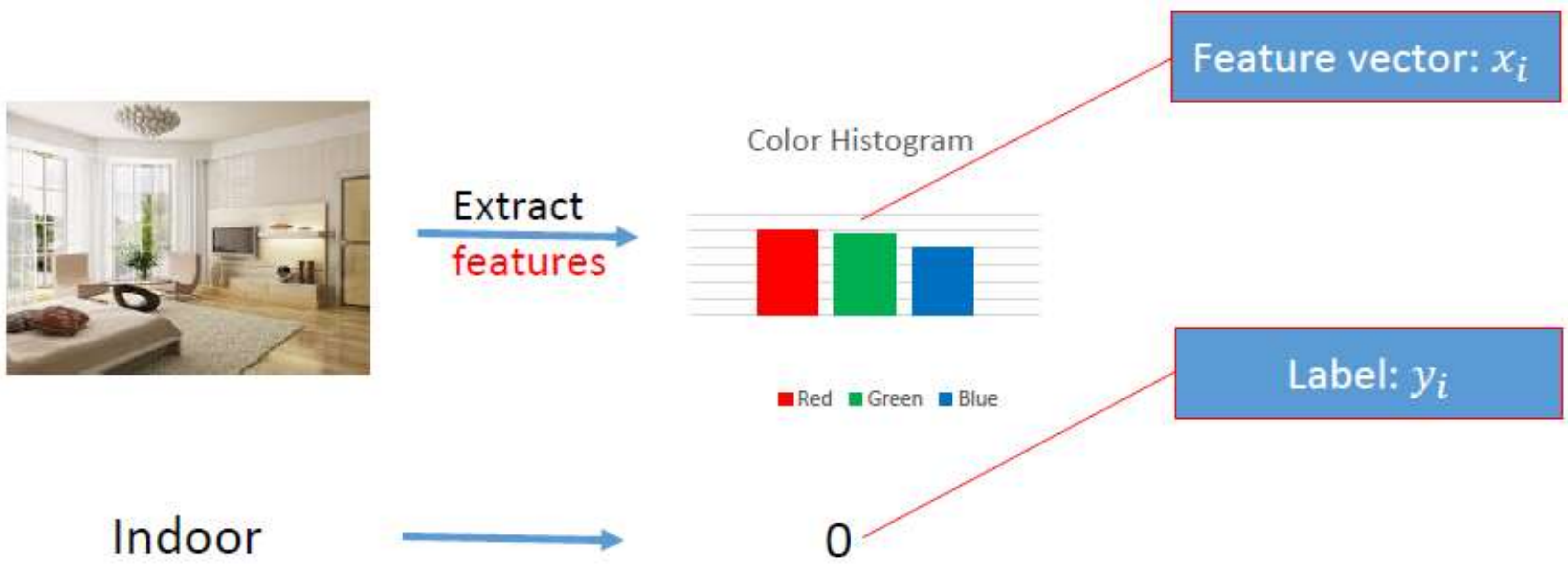
Standard learning scenarios

- Unsupervised learning: no labeled data
- Supervised learning: uses labeled data for prediction on unseen points
- Semi-supervised learning: uses labeled and unlabeled data for prediction on unseen points
- Reinforcement learning: uses reward to learn prediction on action policies.
- …

Supervised learning

- Task formulation

  - Learning example: $(\mathbf{x}, y)$
  - Task to solve: predict target y from input $\mathbf{x}$
    - classification: target is a class ID (from 0 to nb. of class - 1)
    - regression: target is a real number

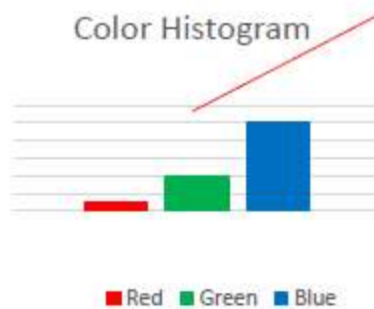


Supervised learning

- Task formulation

  - Learning example: $(\mathbf{x}, y)$
  - Task to solve: predict target y from input $\mathbf{x}$
    - classification: target is a class ID (from 0 to nb. of class - 1)
    - regression: target is a real number

Extract features



Feature vector: x_j

outdoor → 1

Label: y_j

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$
  - Find $y = f(x)$ using training data
  - s.t. f correct on test data

What kind of functions?

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. f correct on test data

Hypothesis class

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. f correct on test data

Connection between training data and test data?

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. f correct on test data i.i.d. from distribution D

They have the same distribution

i.i.d.: independently identically distributed

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \le i \le n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. f correct on test data i.i.d. from distribution D

What kind of performance measure?

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Various loss functions

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

  - Examples of loss functions:
    - 0-1 loss: $l(f, x, y) = \mathbb{I}[f(x) \neq y]$ and $L(f) = \Pr[f(x) \neq y]$
    - l_2 loss: $l(f, x, y) = [f(x) - y]^2$ and $L(f) = \mathbb{E}[f(x) - y]^2$

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ using training data
  - s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

How to use?

Learning problem

- Problem setup
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ that minimizes $\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} l(f, x_i, y_i)$
  - s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Empirical loss

Learning as iterative optimization

- Gradient descent
  - choose initial $w^{(0)}$, repeat

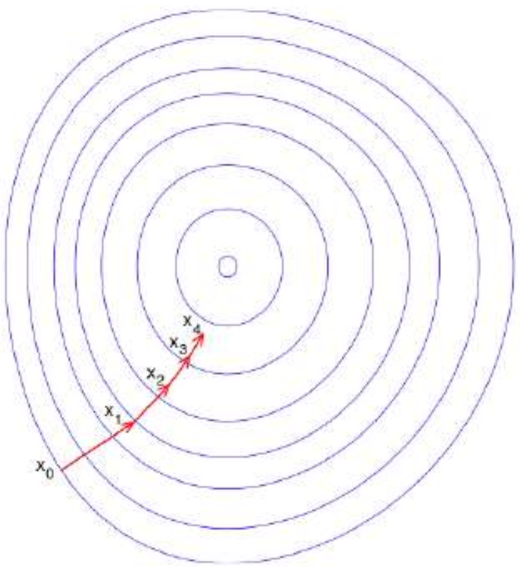
$$w^{(t+1)} = w^{(t)} - \eta_t \cdot \nabla \mathrm{L}\left(w^{(t)}\right)$$

    until stop
  - η_t is the learning rate, and

$$\nabla \mathrm{L}\left(w^{(t)}\right) = \frac{1}{n} \sum_i \nabla_w \mathrm{L_i}(w^{(t)}; y_i, x_i)$$

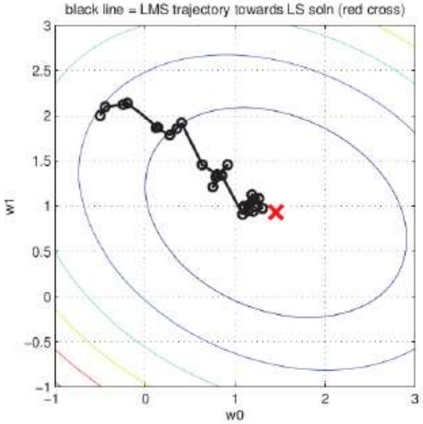
  - How to stop? $\|w^{(t+1)} - w^{(t)}\| \leq \epsilon$ or $\|\nabla \mathrm{L}(w^{(t)})\| \leq \epsilon$

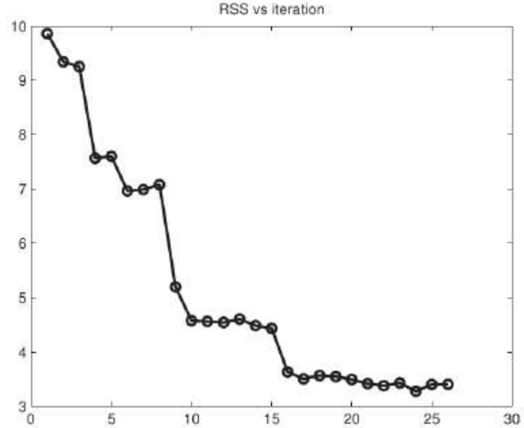
Two dimensional example:



Learning as iterative optimization

- Stochastic gradient descent (SGD)
  - Suppose data points arrive one by one
  - $\hat{L}(\mathrm{w}) = \frac{1}{n}\sum_{t=1}^{n} l(\mathrm{w}, x_t, y_t)$, but we only know $l(\mathrm{w}, x_t, y_t)$ at time t
  - Idea: simply do what you can based on local information
    - Initialize w_0
    - $\mathrm{w}_{t+1} = \mathrm{w}_t - \eta_t \nabla l(\mathrm{w}_t, x_t, y_t)$





Supervised learning pipeline

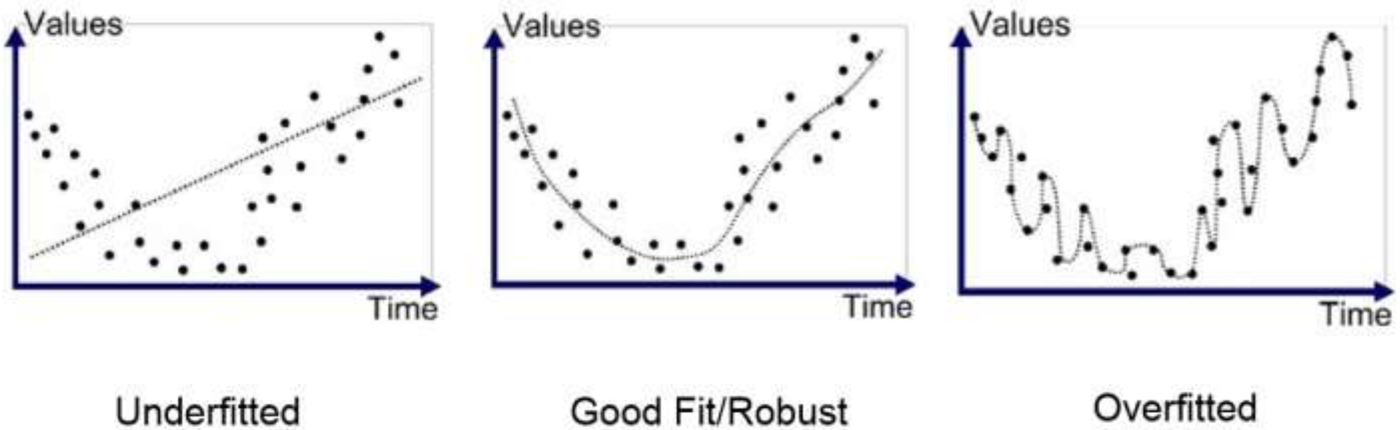
- Three steps
  - Collect data and extract features
  - Build model: choose hypothesis class $\mathcal{H}$ and loss function l
  - Optimization: minimize the empirical loss

- Datasets & hyper-parameters
  - Hyper-parameter: a parameter of a model that is not trained (specified before training)

  - Training set $\mathcal{D}^{\mathrm{train}}$ serves to train a model
  - Validation set $\mathcal{D}^{\mathrm{valid}}$ serves to select hyper-parameters
  - Test set $\mathcal{D}^{\mathrm{test}}$ serves to estimate the generalization performance (error)

Generalization

- Model selection for better generalization
  - Capacity: flexibility of a model
  - Underfitting: state of model which could improve generalization with more training or capacity
  - Overfitting: state of model which could improve generalization with less training or capacity
  - Model Selection: process of choosing the best hyper-parameters on validation set



Generalization

- Training/Validation curves

Questions

- Generalization
  - Interaction between training set size/capacity/training time and training error/generalization error
- If capacity increases:
  - Training error will ?
  - Generalization error will ?
- If training time increases:
  - Training error will ?
  - Generalization error will ?
- If training set size increases:
  - Generalization error will ?
  - Gap between the training and generalization error will ?

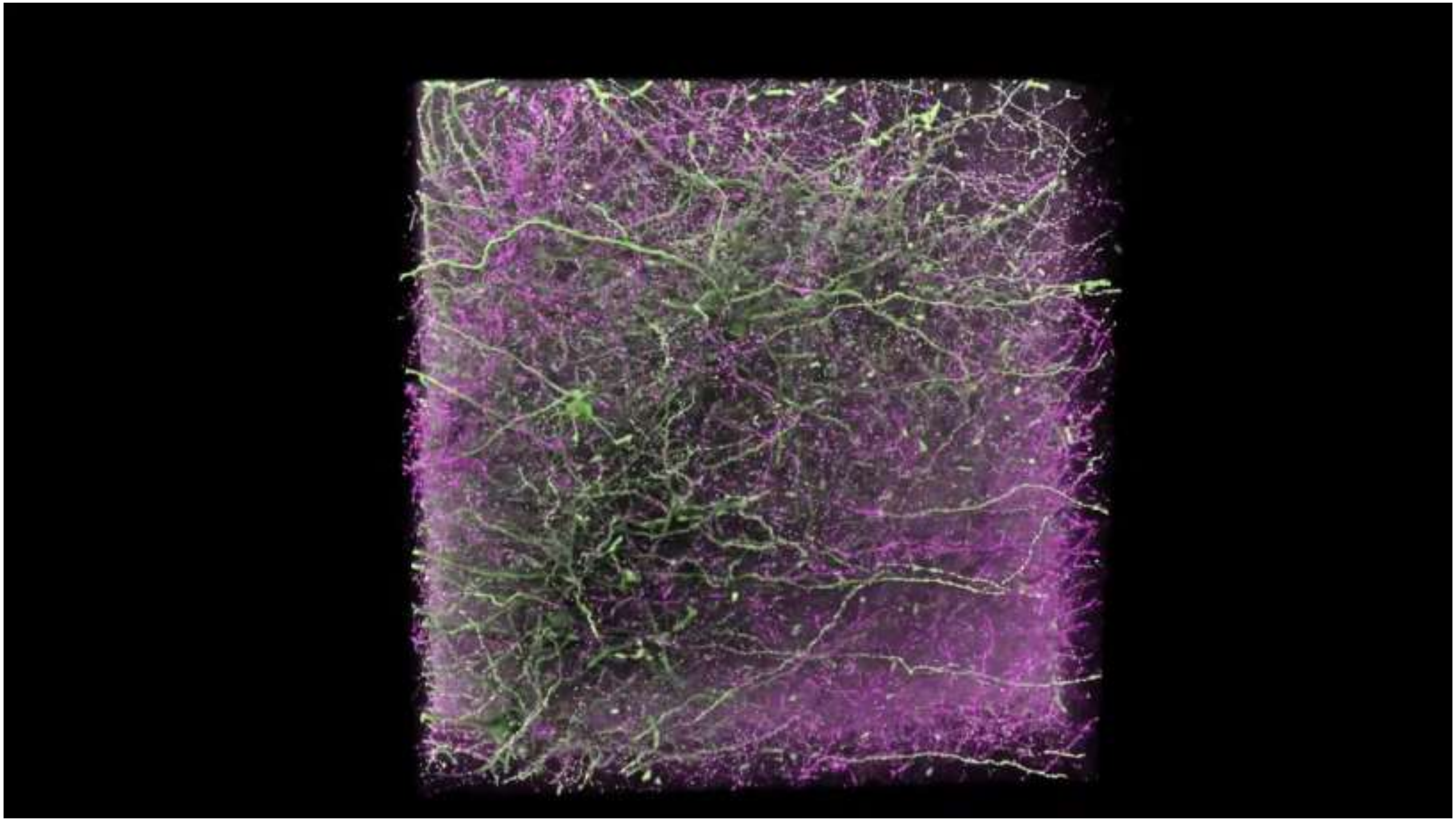
Outline

- Course logistics
- Introduction to deep learning
- Machine learning review
- Artificial neurons
  - Math model
  - Perceptron algorithm

Acknowledgement: Hugo Larochelle's, Mehryar Mohri@NYU's & Yingyu Liang@Princeton's course notes

Artificial Neuron

- Biological inspiration



https://www.youtube.com/watch?v=m0rHZ_RDdyQ

Artificial Neuron

- Biological inspiration
  - Our brain has $\sim 10^{11}$ neurons, each of which communicates (is connected) to $\sim 10^4$ other neurons

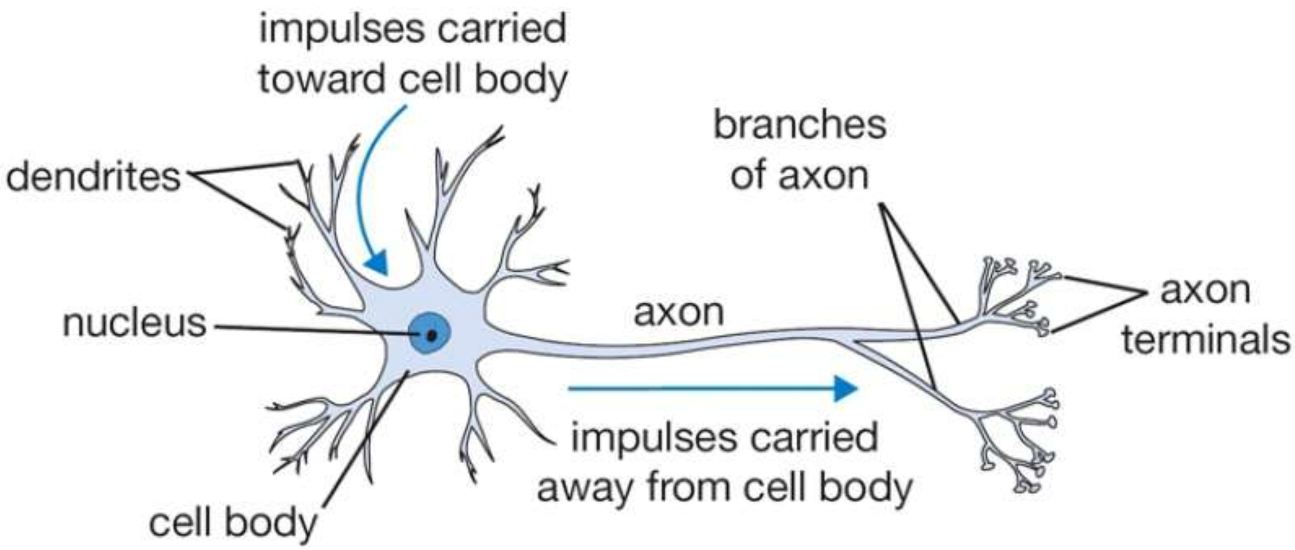
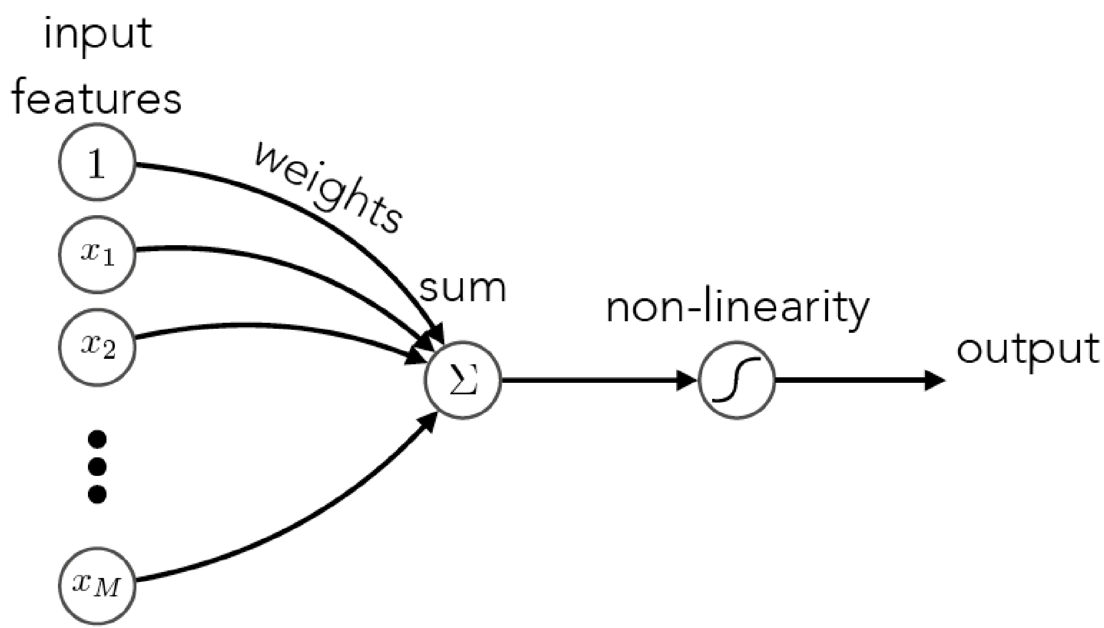


Figure : The basic computational unit of the brain: Neuron

Mathematical model of a neuron



artificial neuron: *weighted sum and non-linearity*

bias　　　　　　　　　　　　　　　　　　input features

$$s = b + w_1 x_1 + w_2 x_2 + \cdots + w_M x_M = \mathbf{w}^\mathsf{T}\mathbf{x}$$

sum　　　　weights

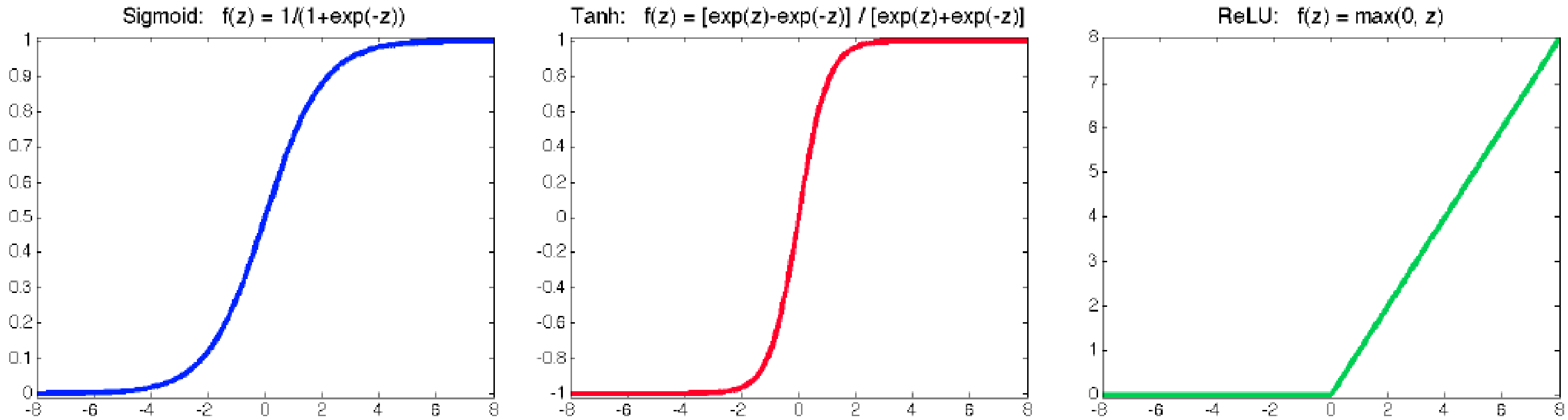
$$h = g(s)$$

output　　　non-linearity　　　sum

Activation functions

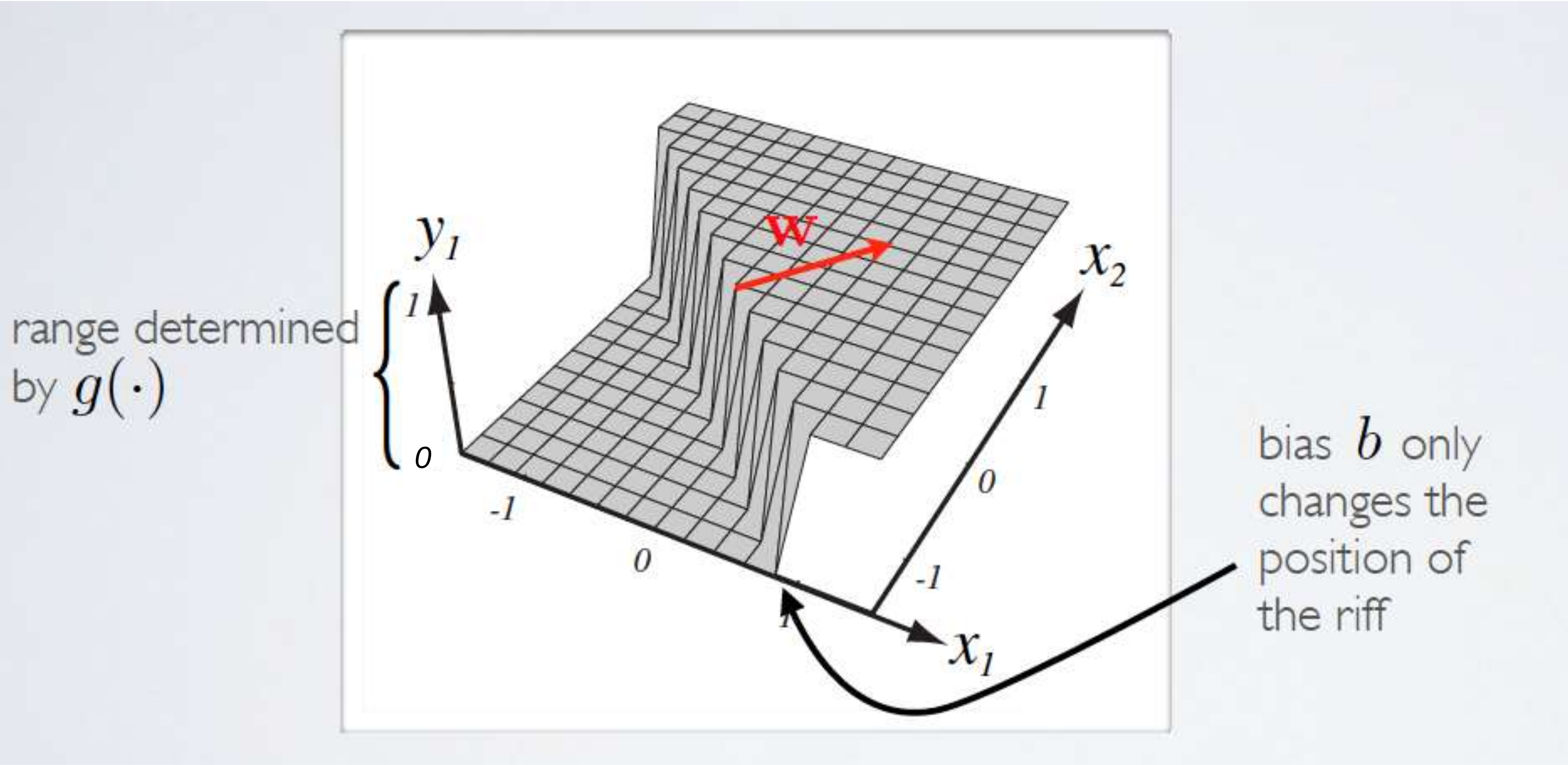
Most commonly used activation functions:

- Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$
- Tanh: $\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$
- ReLU (Rectified Linear Unit): $\mathrm{ReLU}(z) = \max(0, z)$



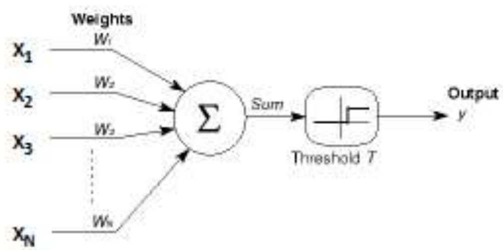
Capacity of single neuron

- Sigmoid activation function



What a single neuron does?

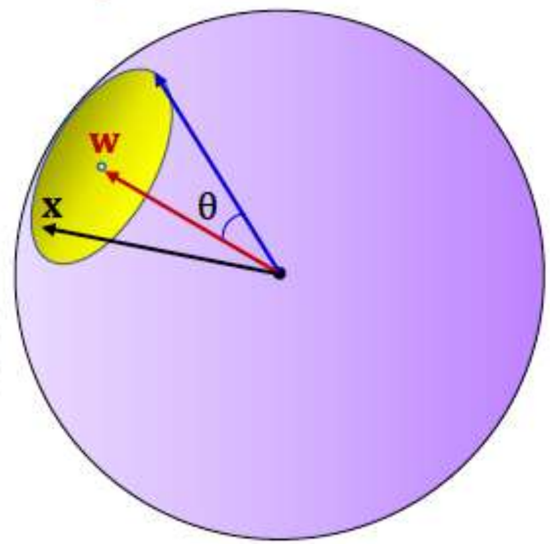
- A neuron (perceptron) fires if its input is within a specific angle of its weight
  - If the input pattern matches the weight pattern closely enough



$$\mathbf{x}^T\mathbf{w} > T$$

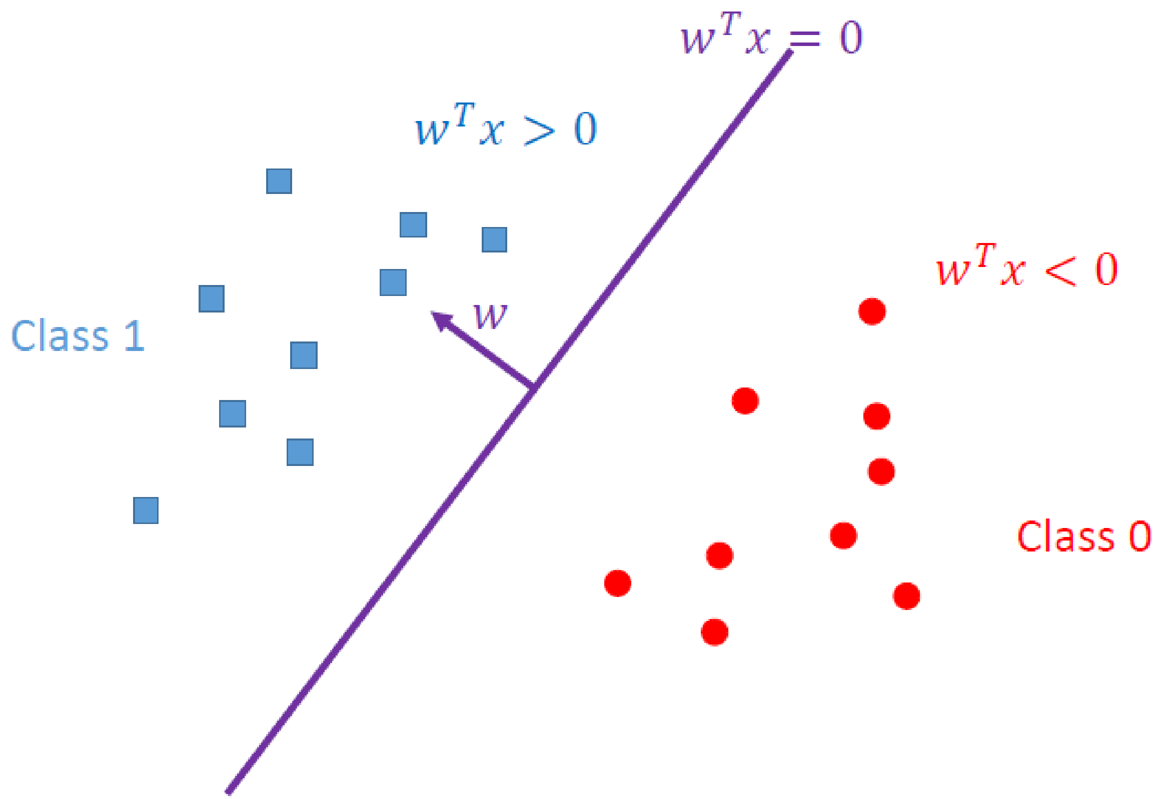
$$\Rightarrow \cos\theta > \frac{T}{|\mathbf{x}||\mathbf{w}|}$$

$$\Rightarrow \theta < \cos^{-1}\left(\frac{T}{|\mathbf{x}||\mathbf{w}|}\right)$$



Single neuron as a linear classifier

- Binary classification



How do we determine the weights?

- Learning problem
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Hypothesis $f_w(x) = w^T x$
    - $y = 1$ if $w^T x > 0$
    - $y = 0$ if $w^T x < 0$
  - Prediction: $y = \text{step}(f_w(x)) = \text{step}(w^T x)$

Linear model $\mathcal{H}$

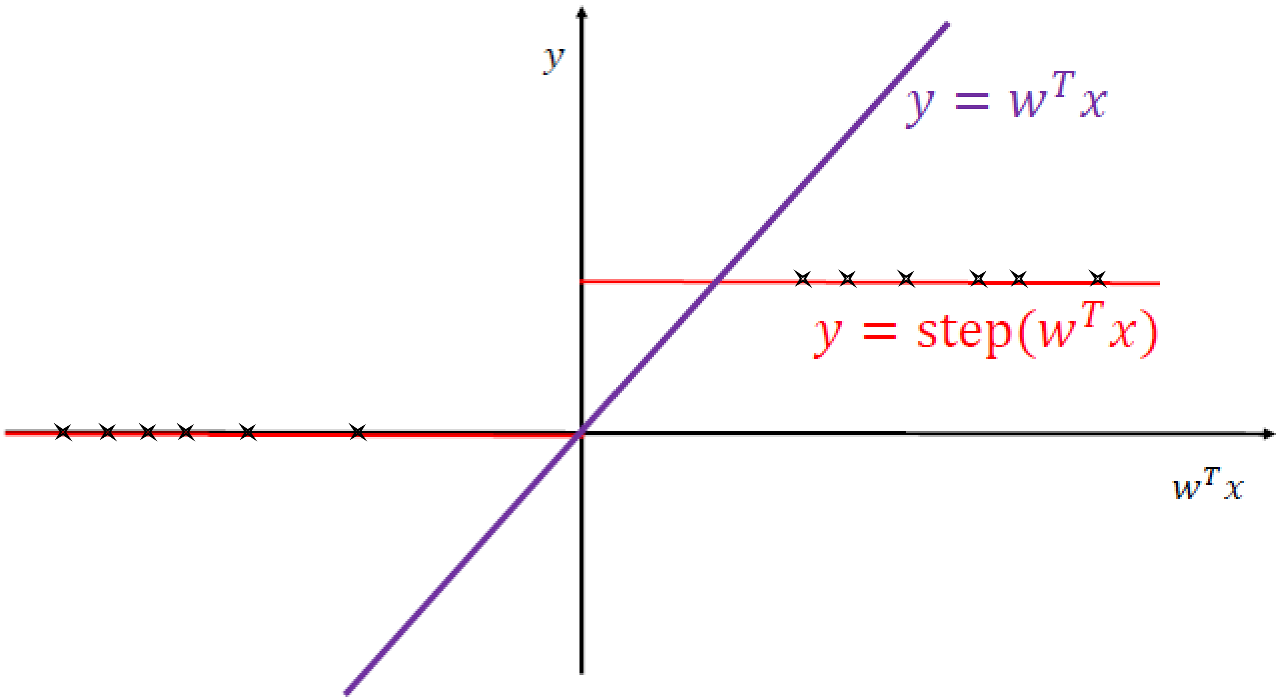
Linear classification

- Learning problem: simple approach
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $f_w(x) = w^T x$ that minimizes $\hat{L}(f_w) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$
  - Drawback: Sensitive to "outliers"

Reduce to linear regression; ignore the fact $y \in \{0,1\}$

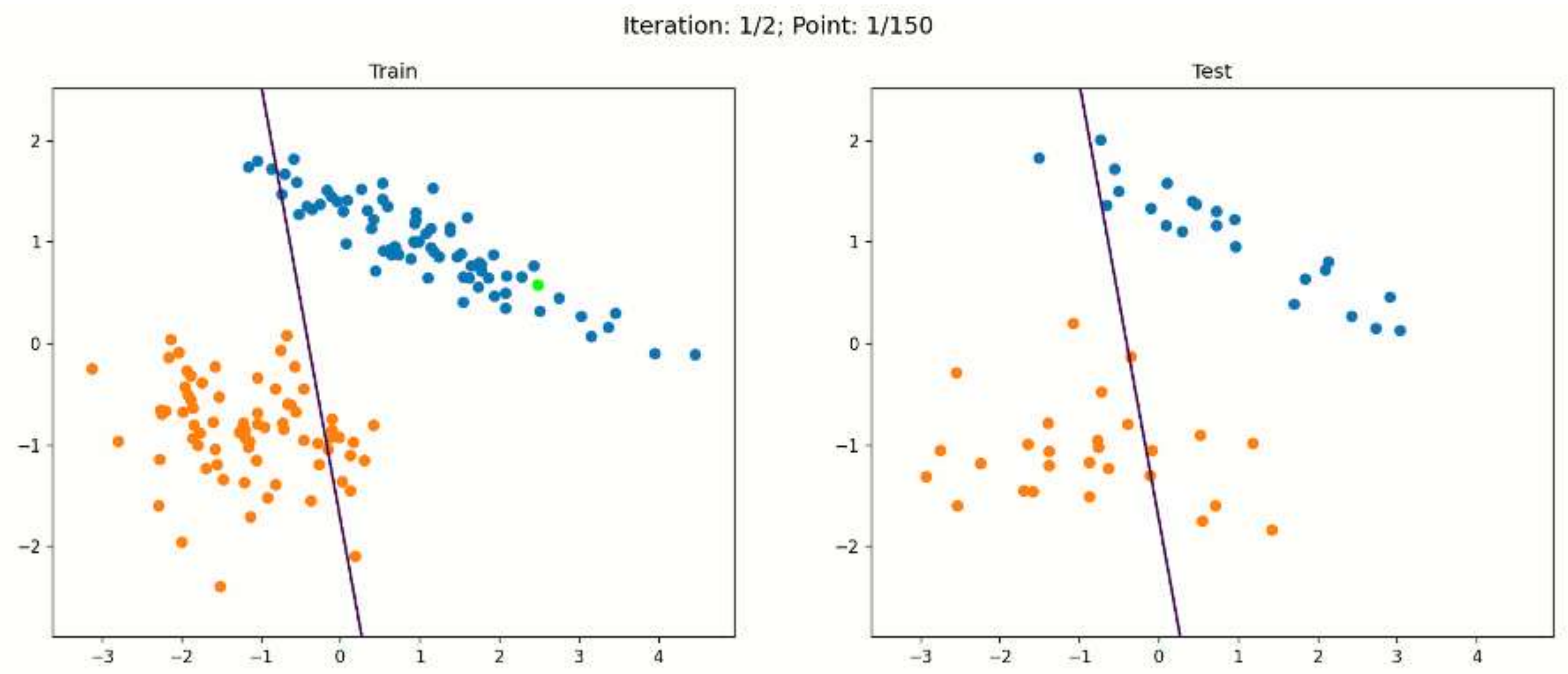
1D Example

- Compare two predictors



Perceptron algorithm

- Learn a single neuron for binary classification



https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-3c8e76b4e2d1

Perceptron algorithm

- Learn a single neuron for binary classification

- Task formulation
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Hypothesis $f_w(x) = w^T x$
    - $y = +1$ if $w^T x > 0$
    - $y = -1$ if $w^T x < 0$
  - Prediction: $y = \text{sign}(f_w(x)) = \text{sign}(w^T x)$

  - Goal: minimize classification error

Perceptron algorithm

- Algorithm outline

- Assume for simplicity: all x_i has length 1

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.

Perceptron: figure from the lecture note of Nina Balcan

Perceptron algorithm

- Intuition: correct the current mistake

  - If mistake on a positive example

$$w_{t+1}^T x = (w_t + x)^T x = w_t^T x + x^T x = w_t^T x + 1$$

  - If mistake on a negative example

$$w_{t+1}^T x = (w_t - x)^T x = w_t^T x - x^T x = w_t^T x - 1$$

Perceptron algorithm

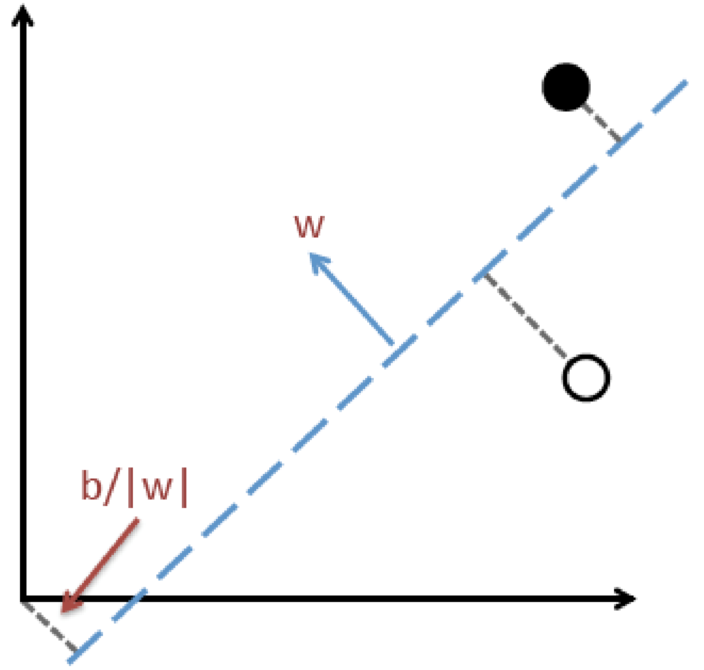
- The Perceptron theorem
  - Suppose there exists w^* that correctly classifies $\{(x_i, y_i)\}$
  - W.L.O.G., all x_i and w^* have length 1, so the minimum distance of any example to the decision boundary is

$$\gamma = \min_i |(w^*)^T x_i|$$

  - Then Perceptron makes at most $\left(\frac{1}{\gamma}\right)^2$ mistakes

Hyperplane Distance

- Line is a 1D, Plane is 2D
- Hyperplane is many D
  - Includes Line and Plane
- Defined by (w,b)
- Distance: $\dfrac{|w^T x - b|}{\|w\|}$
- Signed Distance: $\dfrac{w^T x - b}{\|w\|}$



Linear Model = un-normalized signed distance!

Perceptron algorithm

- The Perceptron theorem: proof
  - First look at the quantity $w_t^T w^*$
  - Claim 1: $w_{t+1}^T w^* \geq w_t^T w^* + \gamma$
  - Proof: If mistake on a positive example x

$$w_{t+1}^T w^* = (w_t + x)^T w^* = w_t^T w^* + x^T w^* \geq w_t^T w^* + \gamma$$

  - If mistake on a negative example

$$w_{t+1}^T w^* = (w_t - x)^T w^* = w_t^T w^* - x^T w^* \geq w_t^T w^* + \gamma$$

Perceptron algorithm

- The Perceptron theorem: proof

  - Next look at the quantity $||w_t||$

  - Claim 2: $||w_{t+1}||^2 \leq ||w_t||^2 + 1$
  - Proof: If mistake on a positive example x

$$||w_{t+1}||^2 = ||w_t + x||^2 = ||w_t||^2 + ||x||^2 + 2w_t^T x$$

Negative since we made a mistake on x

Perceptron algorithm

- The Perceptron theorem: proof intuition
  - Claim 1: $w_{t+1}^T w^* \geq w_t^T w^* + \gamma$
  - Claim 2: $||w_{t+1}||^2 \leq ||w_t||^2 + 1$

The correlation gets larger. Could be:

1. w_{t+1} gets closer to w^*
2. w_{t+1} gets much longer

Rules out the bad case "2. w_{t+1} gets much longer"

Perceptron algorithm

- The Perceptron theorem: proof

  - Claim 1: $w_{t+1}^T w^* \geq w_t^T w^* + \gamma$
  - Claim 2: $\left|\left|w_{t+1}\right|\right|^2 \leq \left|\left|w_t\right|\right|^2 + 1$

  After M mistakes:

  - $w_{M+1}^T w^* \geq \gamma M$
  - $\left|\left|w_{M+1}\right|\right| \leq \sqrt{M}$
  - $w_{M+1}^T w^* \leq \left|\left|w_{M+1}\right|\right|$

  So $\gamma M \leq \sqrt{M}$, and thus $M \leq \left(\frac{1}{\gamma}\right)^2$

Perceptron algorithm

The Perceptron theorem

- Suppose there exists w^* that correctly classifies $\{(x_i, y_i)\}$
- W.L.O.G., all x_i and w^* have length 1, so the minimum distance of any example to the decision boundary is

$$\gamma = \min_i |(w^*)^T x_i|$$

Need not be i.i.d. !

- Then Perceptron makes at most $\left(\frac{1}{\gamma}\right)^2$ mistakes

Do not depend on n, the length of the data sequence!

Perceptron algorithm

- What loss function is minimized?
  - Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  - Find $y = f(x) \in \mathcal{H}$ that minimizes $\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} l(f, x_i, y_i)$
  - s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Perceptron algorithm

- What loss function is minimized?
  - Hypothesis: $y = \text{sign}(w^T x)$
  - Define hinge loss

$$l(w, x_t, y_t) = -y_t w^T x_t \, \mathbb{I}[\text{mistake on } x_t]$$

$$\hat{L}(w) = -\sum_t y_t w^T x_t \, \mathbb{I}[\text{mistake on } x_t]$$

$$w_{t+1} = w_t - \eta_t \nabla l(w_t, x_t, y_t) = w_t + \eta_t y_t x_t \, \mathbb{I}[\text{mistake on } x_t]$$

Perceptron algorithm

- What loss function is minimized?
  - Hypothesis: $y = \text{sign}(w^T x)$

$$w_{t+1} = w_t - \eta_t \nabla l(w_t, x_t, y_t) = w_t + \eta_t y_t x_t \, \mathbb{I}[\text{mistake on } x_t]$$

  - Set $\eta_t = 1$. If mistake on a positive example

$$w_{t+1} = w_t + y_t x_t = w_t + x$$

  - If mistake on a negative example

$$w_{t+1} = w_t + y_t x_t = w_t - x$$

Summary

- Introduction to deep learning
- Course logistics
- Review of basic math & ML
- Artificial neurons

- Next time
  - Basic neural networks