

WITH TENSORFLOW

---

DEEP LEARNING

AI, My Neighbors! 5

Overviews 50

Terms 65

Linear Regression 72

Gradient Descent Algorithm 78

TensorFlow Overviews 85

Multi-Variable Linear Regression 99



**Logistic Classification 113**

**SoftMax Cross-Entropy 129**

**Convolutional Neural Network 148**

**Utility 181**

**Neural Networks History 189**

**Application And Tips 213**

**Hinton's Summary 234**



BackPropagation 260

Diverse Algorithm 271

TensorBoard Visualization 290

Source Codes 1 300

Source Codes 2 315

Source Codes 3 348

Source Codes 4 369





# DEEP LEARNING

---

# AI, MY NEIGHBORS!

## FORWARD HOSPITAL



- ▶ <https://goforward.com>
- ▶ 2017년 1월 샌프란시스코
- ▶ 엔지니어들이 설립한 AI 병원
- ▶ 월정액 149\$
- ▶ 24시간 건강 검진 및 상담 무제한
- ▶ Design Your Health

## 진료 순서

1. 병원 방문
2. 아이패드로 로그인
3. 바디스캐너로 건강 진단
4. 의사 상담
5. 24시간 상담 가능 (스마트폰 앱 연동)



## ALEXA : KID SKILLS



Sesame Street

<https://www.amazon.com/Sesame-Workshop-Street/dp/B073XBBWRQ>



The SpongeBob Challenge



Amazon Storytime



NASA Mars



Word of the Day



*“Alexa, open  
Sesame Street”*

## ALEXA : KID SKILLS

- ▶ 1-2-3 Math

<https://www.amazon.com/Shanthan-Kesharaju-1-2-3-Math/dp/B01AVQLZQ0>

- ▶ This Day In History

- ▶ Jeopardy!

- ▶ Animal Game

- ▶ The Magic Door



## 진르터우탸오(오늘의 헤드라인)

- ▶ 모바일 뉴스앱
- ▶ 인공지능 활용해 개인별 맞춤 콘텐츠 제공
- ▶ 7,800만명 구독 - 3년새 200배 성장
- ▶ 누적 사용자 6억명, 하루 이용자 수 1억명
- ▶ 하루 평균 이용 시간 76분
- ▶ 기업가치 110억\$(약 12조원)



下载APP | 北京 晴 15° / 29°

登录 反馈 侵权投诉 头条

## 今日头条

推荐

热点

视频

图片

段子

社会

娱乐

游戏

体育

汽车

财经

搞笑



文在寅第五次与特朗普通电话 商讨半岛局势

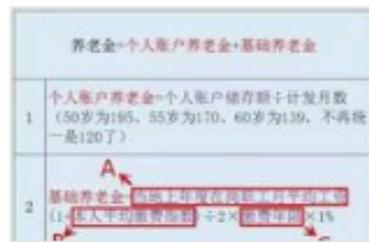


要闻  
社会  
娱乐  
体育  
军事  
明星



程序员被骗婚自杀案和王宝强离婚案代理律师张起淮：  
曾代理婚内诈骗案，罪犯被判10年

娱乐 中国之声 · 2520评论 · 刚刚



社保交满15年，退休能领多少钱？

社会 头条问答 · 8分钟前

大家都在搜：乔任梁逝世一周年

搜索



网上有害信息举报专区  
举报电话：12377



24小时热闻

中超-权健2-2国安全场集锦：帕托

## ZUMEPIZZA

- ▶ <https://www.zumepizza.com/>
- ▶ 매장에서 먹는 것과 같은 배달 피자
- ▶ 2016년 4월
- ▶ 200여평 주방에서 1시간에 288개 생산
- ▶ 1분에 4.5개 완성
- ▶ 로봇으로 인건비 줄이고, 유기농 재료 사용



## 매장

- ▶ 주문이 들어오면 주방의 스크린으로 전송
- ▶ 사람이 도우를 사람이 얇게 펼친다.
- ▶ 존과 폐페가 메뉴에 따라 적당량의 토마토 소스를 뿌린다.
- ▶ 천장에 매달린 마르타가 소스를 고르게 바른다.
- ▶ 사람이 토픽을 얹는다.
- ▶ 브루노가 토픽이 끝난 피자를 오븐에 넣어서 굽는다.
- ▶ 1차로 구워진 피자를 빈센치오가 배달 트럭에 싣는다.

## 배달

- ▶ 56개의 오븐을 갖춘 트럭
- ▶ GPS로 목적지에 도착하기 4분 전에 초벌 피자를 굽는다.
- ▶ 고객이 받았을 때, 가장 맛있는 상태의 피자 완성

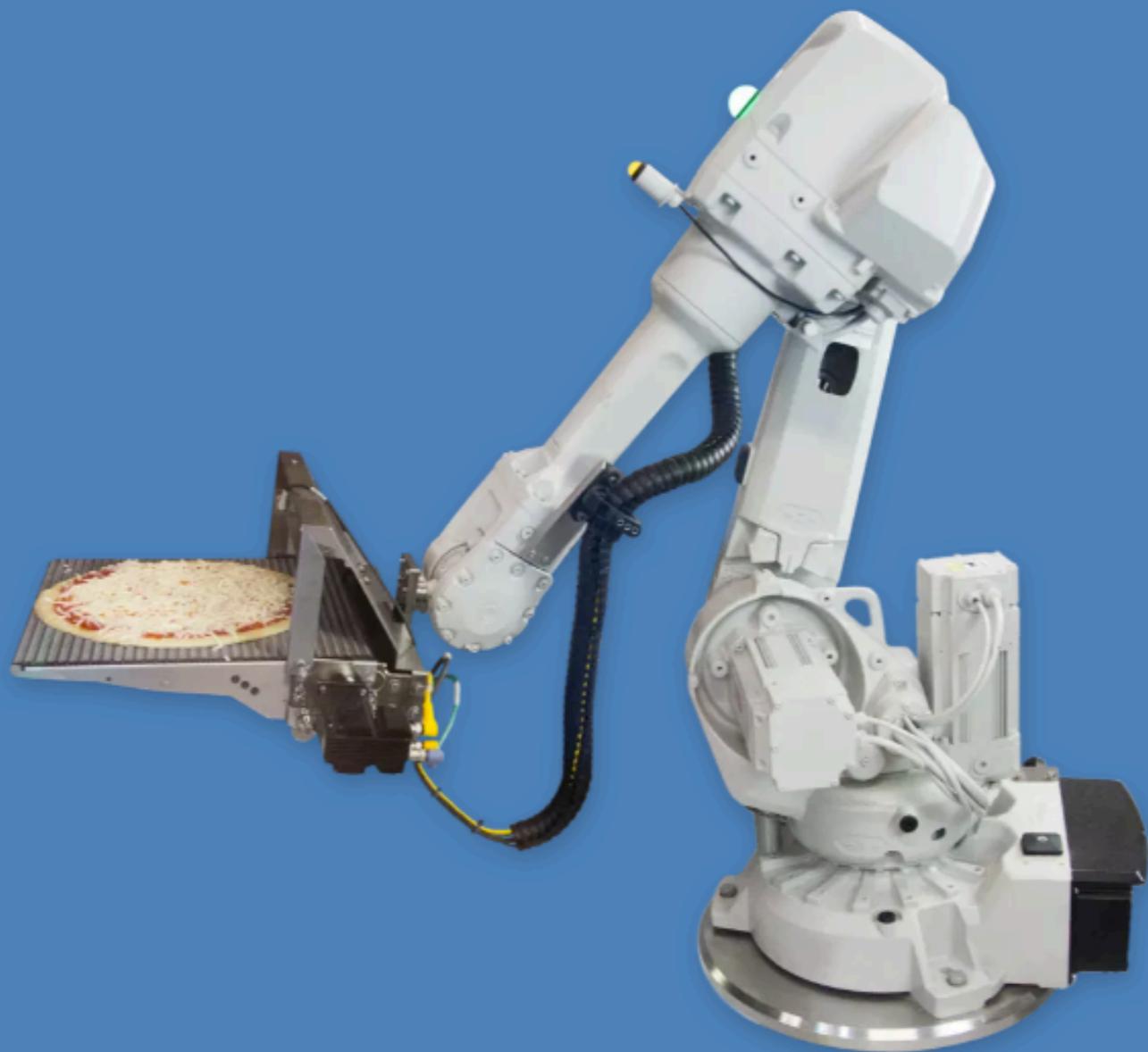


## 목표

- ▶ 주문 -> 생산 -> 도착을 5분 이내로 단축
- ▶ 피자배달 평균은 45분, ZumePizza는 22분.
- ▶ 주문 즉시 배달트럭이 출발하고 배달 중에 완성
- ▶ 고객 성향을 파악해 재료 예측 후 배달 중에 주문 접수
- ▶ 샐러드 로봇, 요구르트 로봇, 탄산음료 로봇, 볶음요리 로봇 개발
- ▶ 자율주행 배달

## *Automation done right*

Alongside our pizzaiolos, we also employ pizza-making robots, Pepe, Giorgio, Marta, Bruno, and Vincenzo, that work together in our kitchen to craft each and every pie. Our co-bots perform low-skill, repetitive, and dangerous tasks, giving human employees more opportunities to do creative, high-skill jobs at Zume.





INKITT

Inkitt

- ▶ <https://www.inkitt.com/>
- ▶ 2016년 여름 첫 번째 책을 출간한 출판사
- ▶ 24권 출간해서 22권이 아마존 베스트셀러
- ▶ 99.99%의 확률로 베스트셀러를 만드는 것이 목표
- ▶ e북 인세는 25%, 종이책 인세는 51%
- ▶ 22권의 책을 쓴 16명의 신예 베스트셀러 작가 배출

1. 누구나 자유롭게 글을 올린다  
저자 4만명, 진행 중인 내용 15만개
2. 독자는 구독하고 평가한다  
선호 장르를 선택하면 스토리 추천  
객관식과 주관식의 다양한 형태로 평가
3. 인공지능이 독자 반응을 분석해서 베스트셀러 가능 여부 판단  
페이지에 머문 시간, 몰입도, 재접속 후 다시 읽었는지 등의 데이터 종합 평가
4. 출판사가 작가에게 출판을 제의하고 작가는 수정한다  
저자는 독자 평가를 바탕으로 수정 및 보완  
표지 디자인은 3개의 후보 중에서 독자들 반응으로 판단

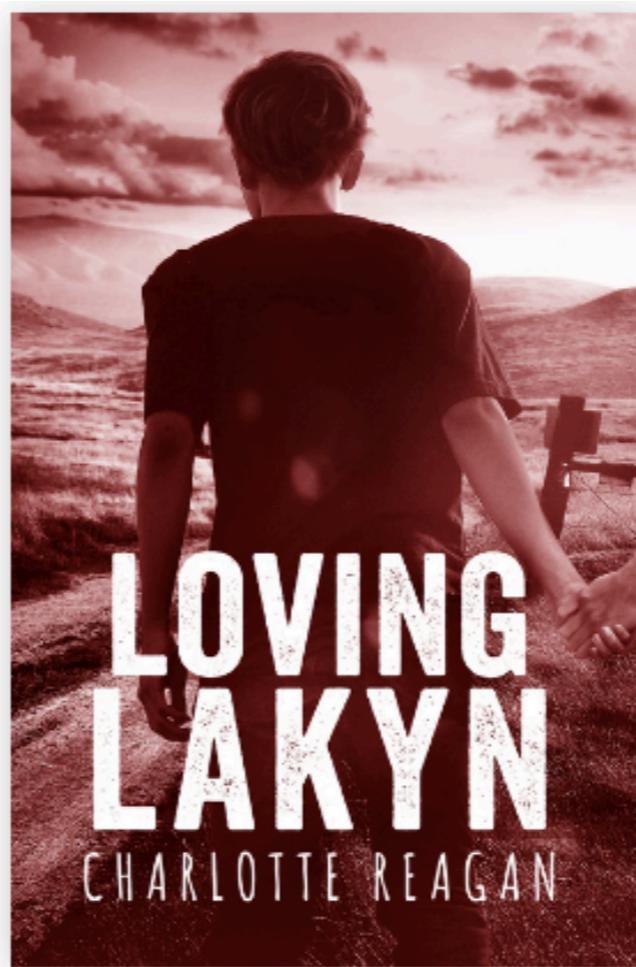
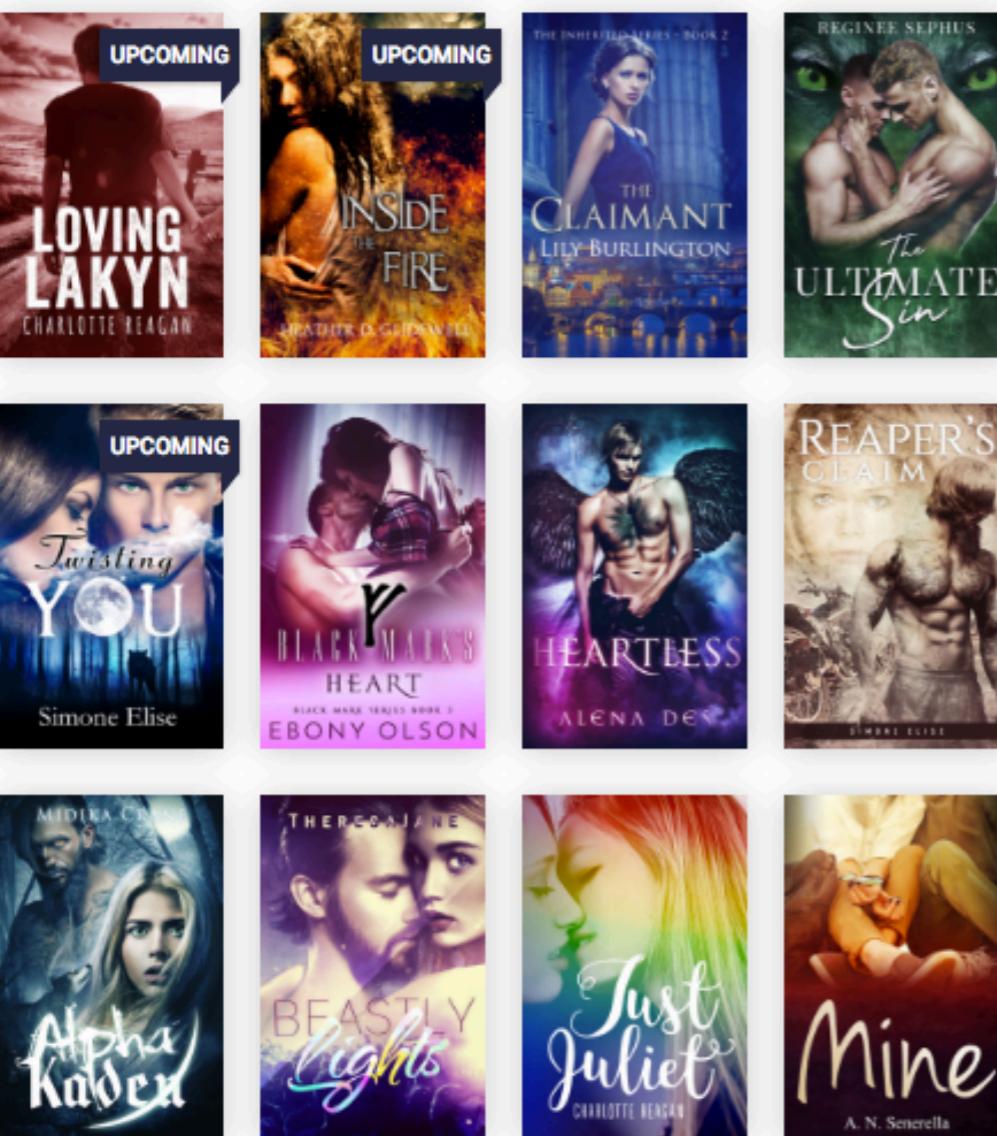
## 평가

### ▶ 객관식 평가

구성(plot), 문체(writing style), 문법,  
전반적 느낌(overall) 각각에 별점 부여

### ▶ 주관식 평가

특별했거나 교훈이 됐던 점  
출간되면 구입할지, 그 이유는  
다른 사람에게 추천할건지, 이유는  
스토리가 어떻게 바뀌었으면 좋을지



## Loving Lakyn

Lakyn James is sixteen years old and hating every second of it. He was supposed to be done, he'd tapped out. End of story, unsubscribe here. Suicide "attempt", they said. His intentions had no "attempt" in them.

Re-entering normal life after 'trying' to take his own is weird. Especially when the world keeps going like it never happened. He still has to eat breakfast, go to school, and somehow convince a cute boy that he's too damaged to date.

Scott White comes with his own problems,

## Writing Contests in 2017: The Definitive Guide

A comprehensive and regularly updated list of writing contests in 2017. Whether you are an aspiring or established author, here you will find a complete list of short story, novel, poetry, and essay competitions where you can submit your fiction and non-fiction masterpieces for the chance to win great prizes and receive notoriety for your work.

Is your writing contest not listed? [Let us know.](#)

TOP PRIZE

**Publishing  
Deal**

[VIEW CONTEST](#)

### Inkitt Writing Competition 2017



- A Dedicated Marketing Team
- Professional Editing & Cover
- 25% royalties from ebook sales and 51% royalties from print sales



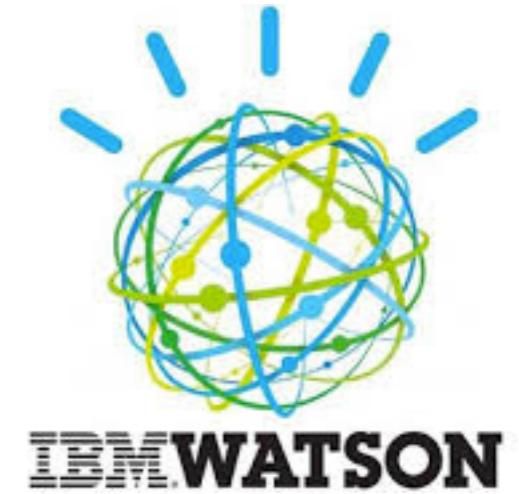
DEADLINE:  
September 30,  
2017



ENTRY FEE:  
Free

## IBM WATSON

- ▶ IBM에서 개발
- ▶ 2011년 퀴즈쇼 제퍼디 참가
- ▶ 인터넷에 연결되지 않은 상태에서 우승
- ▶ 의료, 금융, 세금 규정, 법학, 소비자 서비스 등의 다양한 영역 제공
- ▶ Bluemix 플랫폼에서 Watson Developer Cloud 서비스 제공



## ▶ 금융

정치, 사회적 위험요소를 파악해 리스크 분산 및 포트폴리오 적용.  
현대카드, 신한금융그룹 도입

## ▶ 방송

공포영화 예고편을 학습해서 '모건'에 대한 예고편 완성

## ▶ 의학

가천대 길병원, 건양대/부산대/대구가톨릭대 병원, 계명대 동산병원, 중앙보훈병원  
조선대 병원 예정.

## ▶ 교육

교원에서 수학 디지털 교과서에 왓슨 적용방안 협의 중

## ▶ 쇼핑

롯데그룹에서 백화점, 마트, 편의점, 면세점 대고객 서비스 (지능형 쇼핑 어드바이저)

## ▶ 스마트 홈스피커

SM엔터테인먼트에서 스마트 스피커에 한국어 왓슨 채택

## ▶ 한국어

SK그룹의 AIBRIL

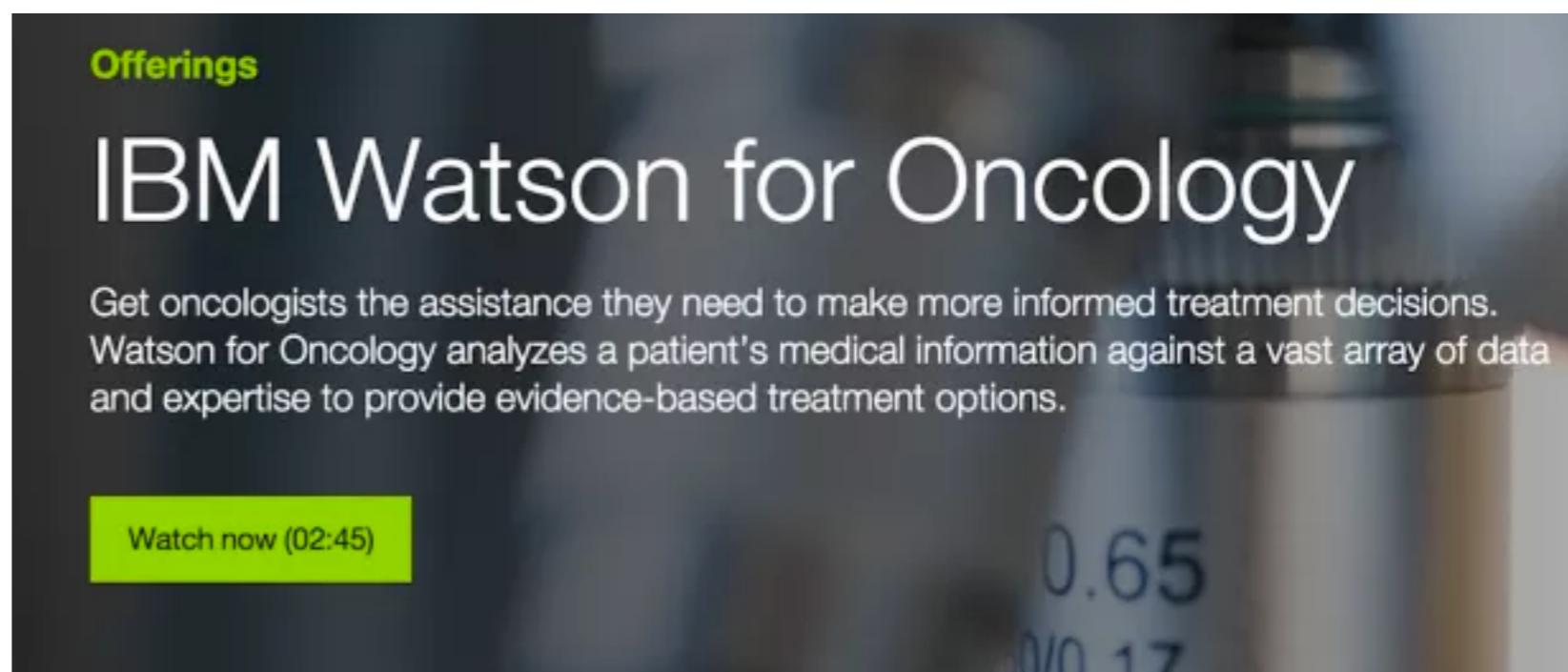


# Service

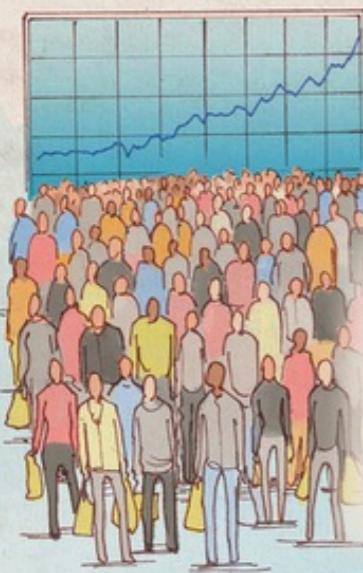
- 01 대화
- 02 자연어 분류
- 03 언어 번역
- 04 검색 및 평가
- 05 문서 변환
- 06 성향 분석
- 07 이미지 인식
- 08 자연어 이해

## 의료 영역

- ▶ 암 환자 1,000명 대상의 IBM Watson 진료 성적 공개 (2017. 03.05)  
<http://www.yoonsupchoi.com/2017/03/05/manipal-watson-for-oncology>
- ▶ 닥터 왓슨과 의료진 항암처방 엇갈리면... 환자 "왓슨 따를게요" (2017. 01. 12)  
[http://news.chosun.com/site/data/html\\_dir/2017/01/12/2017011200289.html](http://news.chosun.com/site/data/html_dir/2017/01/12/2017011200289.html)



DON'T JUST MEET CUSTOMERS' EXPECTATIONS. EXCEED THEM.



## MEET IBM WATSON ENGAGEMENT ADVISOR.

CUSTOMERS TODAY WANT MORE. DEMAND MORE. EXPECT MORE. YOURS INCLUDED.

COMPANIES THAT SATISFY MORE. WIN MORE. A ONE POINT CHANGE IN CUSTOMER SATISFACTION = 4.6% CHANGE IN YOUR MARKET VALUES. ACCORDING TO JOURNAL OF MARKETING, JANUARY 2006.

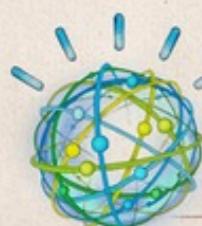
COGNITIVE TECHNOLOGY THAT THINKS AND BRIDGES THE GAP BETWEEN WHAT YOUR CUSTOMERS EXPECT AND THE SERVICES THAT YOU PROVIDE

WATSON ENGAGEMENT ADVISOR CAN:

- COMMUNICATE WITH CUSTOMERS IN NATURAL LANGUAGE
- LEARN FROM CUSTOMERS WITH EACH NEW INTERACTION

TO HELP YOUR BUSINESS:

- ENGAGE CUSTOMERS IN WAYS THEY LIKE
- EMPOWER CUSTOMERS AT THE POINT OF ACTION



### WATSON'S THOUGHT PROCESS:

IT UNDERSTANDS THE REQUEST IN CONTEXT

IT GENERATES AND EXPLORES HYPOTHESES AGAINST THE DATA

IT EVALUATES THOSE HYPOTHESES BASED ON MORE DATA

IT LEARNS FROM ITSELF AS ONLY WATSON CAN DO

IT THEN SHARES WITH YOU WHAT IT "THINKS" ALL IN SECONDS. ALL IN PLAIN ENGLISH

### EXAMPLES OF HOW IT CAN WORK:

BANK CUSTOMERS CAN USE WATSON TO BETTER UNDERSTAND THINGS LIKE RETIREMENT PLANNING.



"WE'RE EXCITED TO EXPLORE HOW WATSON CAN HELP OUR EMPLOYEES PROVIDE BETTER ADVICE, FASTER, WITHOUT HAVING TO NAVIGATE THROUGH A LARGE DATABASE SEARCHING FOR ANSWERS." - ROYAL BANK OF CANADA



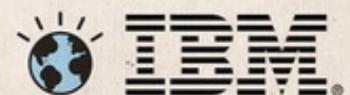
MOBILE CUSTOMERS ON THE GO CAN HAVE ISSUES RESOLVED FASTER WHEN REPS USE WATSON TO TROUBLESHOOT PROBLEMS.



CLIENTS AT COMPANIES SUCH AS NIELSEN CAN CREATE MORE EFFICIENT AND EFFECTIVE MEDIA PLANS.



WATSON ENGAGEMENT ADVISOR. HOW MIGHT WATSON EMPOWER YOUR CUSTOMERS - AND YOU?



IBMWATSON.COM

- ▶ [블로터11th] 알아두면 쓸데있는 신기한 인공지능 50선(2017. 09. 17.)  
<http://www.bloter.net/archives/289626>



## 1. 바둑 기사

알파고, 딥마인드

## 2. 스피커

에코, 구글홈, 홈팟, 인보크, 누구, 기가지니, 웨이브, 카카오미니

## 3. 자살 예방 상담사

문자메시지 기반 24시간 위기 상담 서비스

크라이시스 텍스트 라인(CTL) - 고위험군 필터링

## 4. 오이 분류

9등급 자동분류 시스템

## 5. 승무원

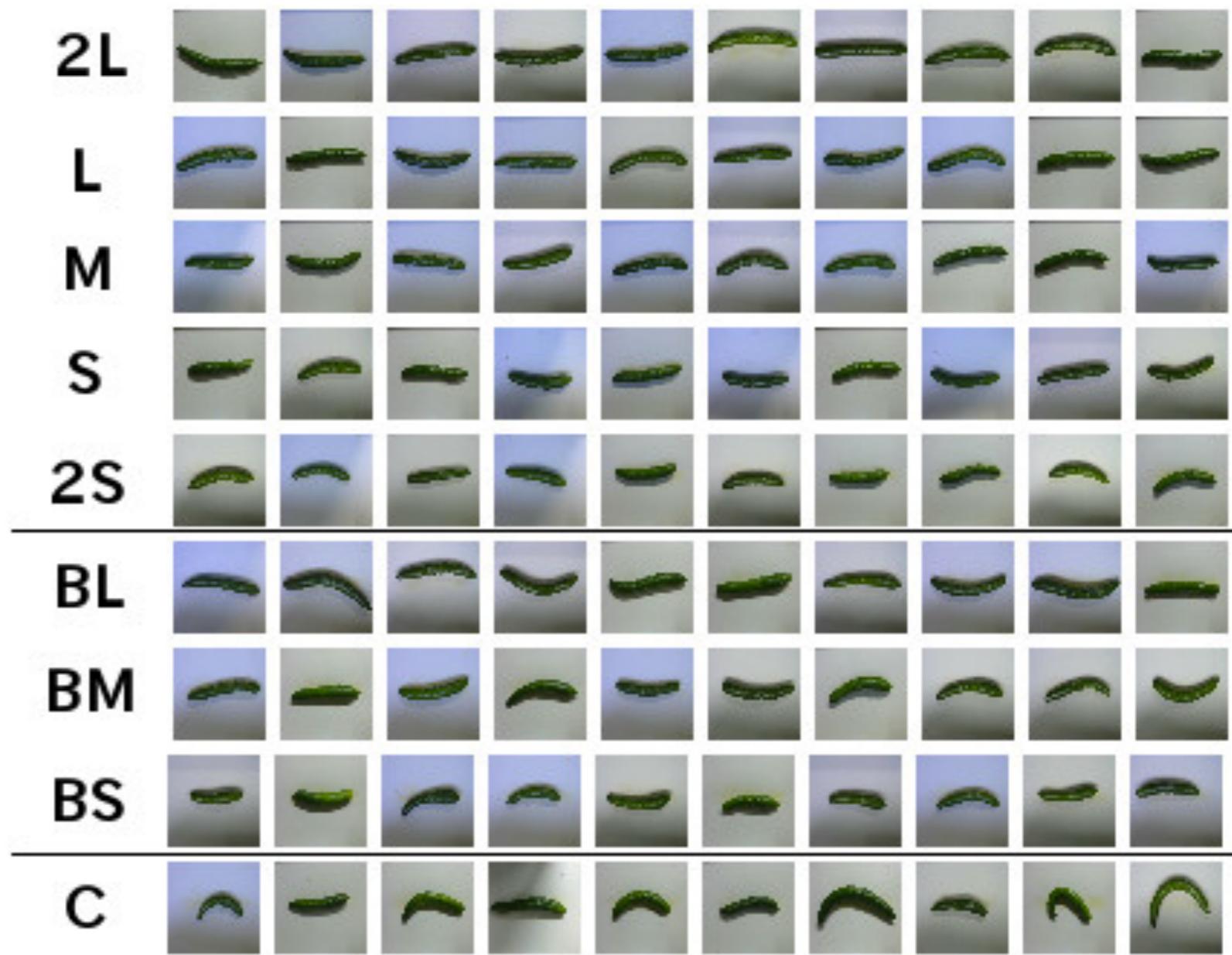
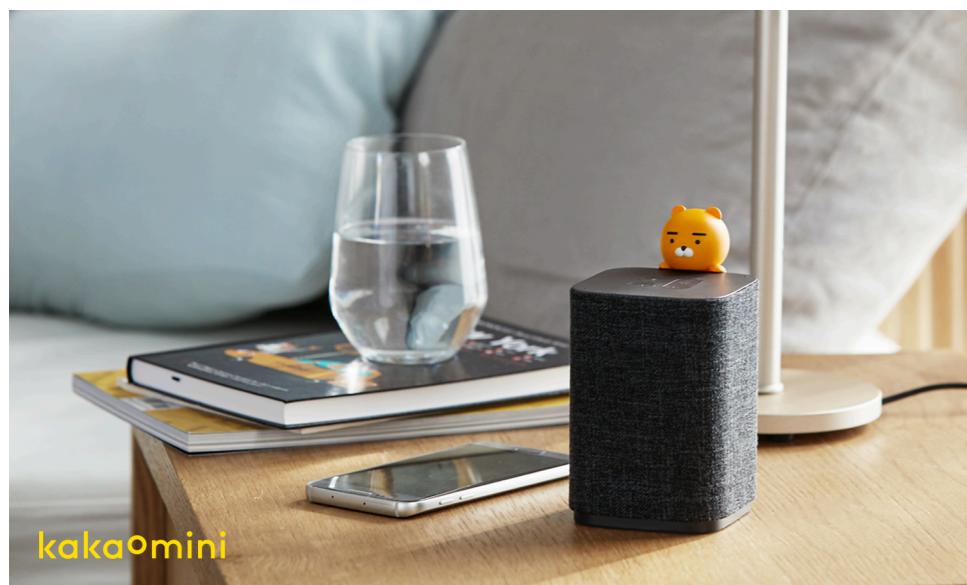
KLM(Koninklijke Luchtvaart Maatschappij) 네델란드 항공사

인공지능 챗봇 서비스 - 일정 확인, 체크인, 발권, 예약 변경 업무 수행

## 스피커, 오이

**amazon echo**

Always ready, connected, and fast. Just ask.



### 6. 아케이드 게이머

팩맨 인간 최고점 266,330점.

마이크로소프트 말루바 999,999점으로 만점

### 7. 흑백 사진을 컬러로

구글과 와세다 대학 공동 연구 - 흑백 사진을 컬러로 변환

### 8. 쇼핑 도우미

아웃도어 브랜드 노스페이스 - 왓슨 활용. '플루이드 리테일' 개발

### 9. 보험 상담사

AIA 생명 한국지점의 인공지능 콜센터 서비스.

고객과 대화 - 계약 정보 확인 및 확정

### 10. 돌고래 언어 해석

스웨덴 스타트업 '가비가이AB' - 2021년을 목표로 프로젝트 돌입

## 말루바

Hybrid Reward Architecture

Maluuba  
A Microsoft company

-	30425 * 10 = 304250
■	801 * 50 = 40050
ghost	17 * 200 = 3400
ghost	6 * 400 = 2400
ghost	3 * 800 = 2400
ghost	1 * 1600 = 1600
ghost	42 * 100 = 4200
ghost	40 * 200 = 8000
ghost	33 * 500 = 16500
ghost	43 * 700 = 30100
ghost	48 * 1000 = 48000
ghost	47 * 2000 = 94000
ghost	89 * 5000 = 445000
	999900

Level: 201

## 흑백 사진 변환

SIGGRAPH 2016

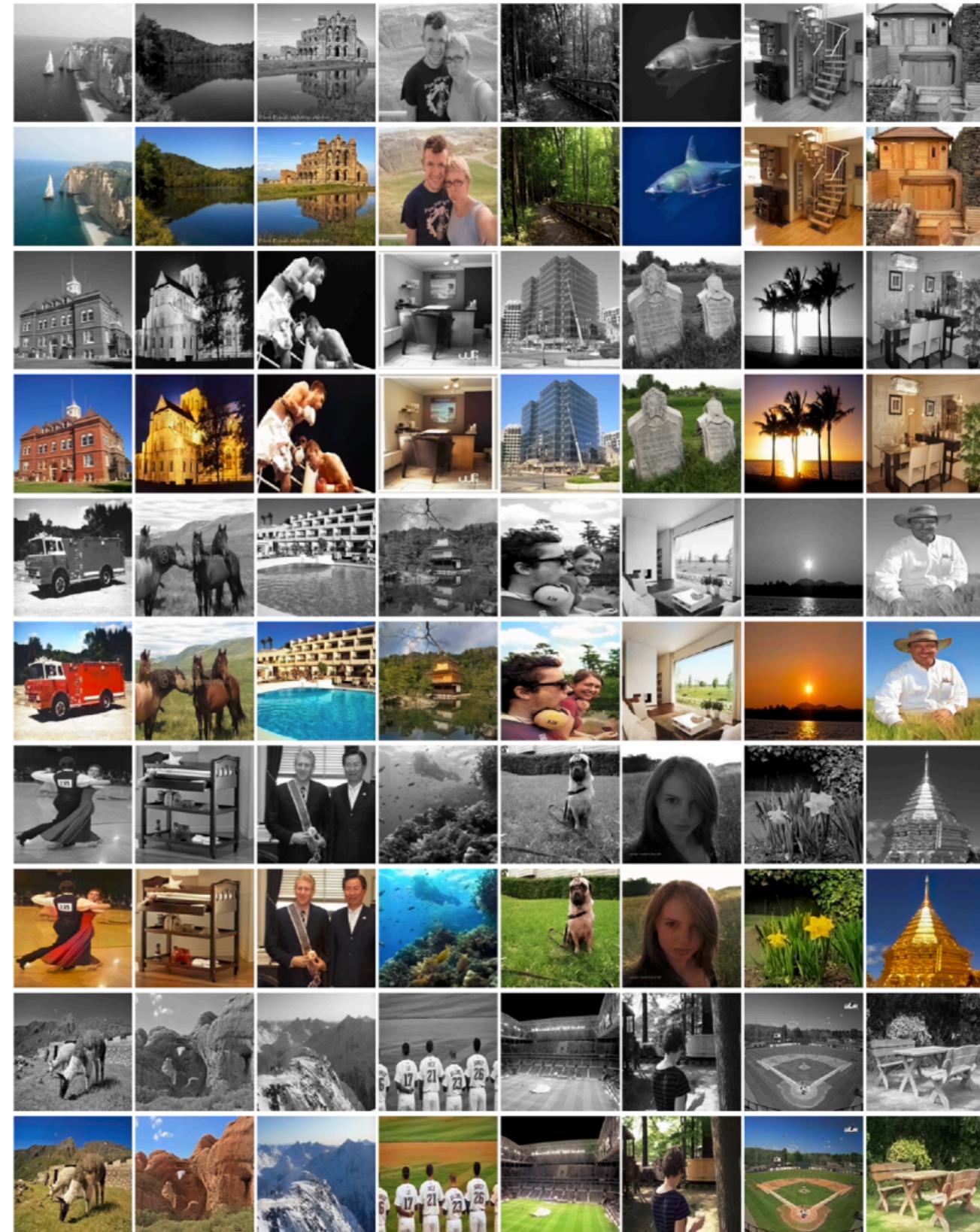


1941년 콜로라도 국립 공원

섬유 공장, 1937년 6월

베리 필드, 1909년 6월

해밀턴, 1936



## 11. 그림 도우미

구글의 '오토드로우' - 펜으로 그린 그림을 멋진 그림으로 변환

## 12. 포르노 비평가

'얼빠진 해커톤'에 등장한 프로젝트. 포르노를 학습하고 해석

## 13. 멸종위기동물 보호

바다소 탐지기 - 드론 항공촬영, 텐서플로우 자동 판별(80%)

## 14. 변호사

베이커앤허스테틀러 법무법인 - 파산 분야에 왓슨 기반의 '로스' 배치

## 15. 법률상담 서비스

19살 조슈아 브라우더 - 주차 딱지 취소에 필요한 채팅봇 '두낫페이' 개발

## 바다소 탐지



## 16. 기자

2012년 개발된 LA타임스의 '퀘이크봇' - 지진 탐지 자동 기사 작성

## 17. 고문서 번역

시스트란 인터네셔널과 미래창조과학부 - 고전문헌 자동번역 시스템 구축  
첫 번째 프로젝트로 국보 303호 '승정원 일기' 선택 - 완역 45년 소요

## 18. 반려동물 장난감

반려동물 케어 플랫폼 '고미랩스' - 놀이패턴, 견종, 나이, 성별 분석

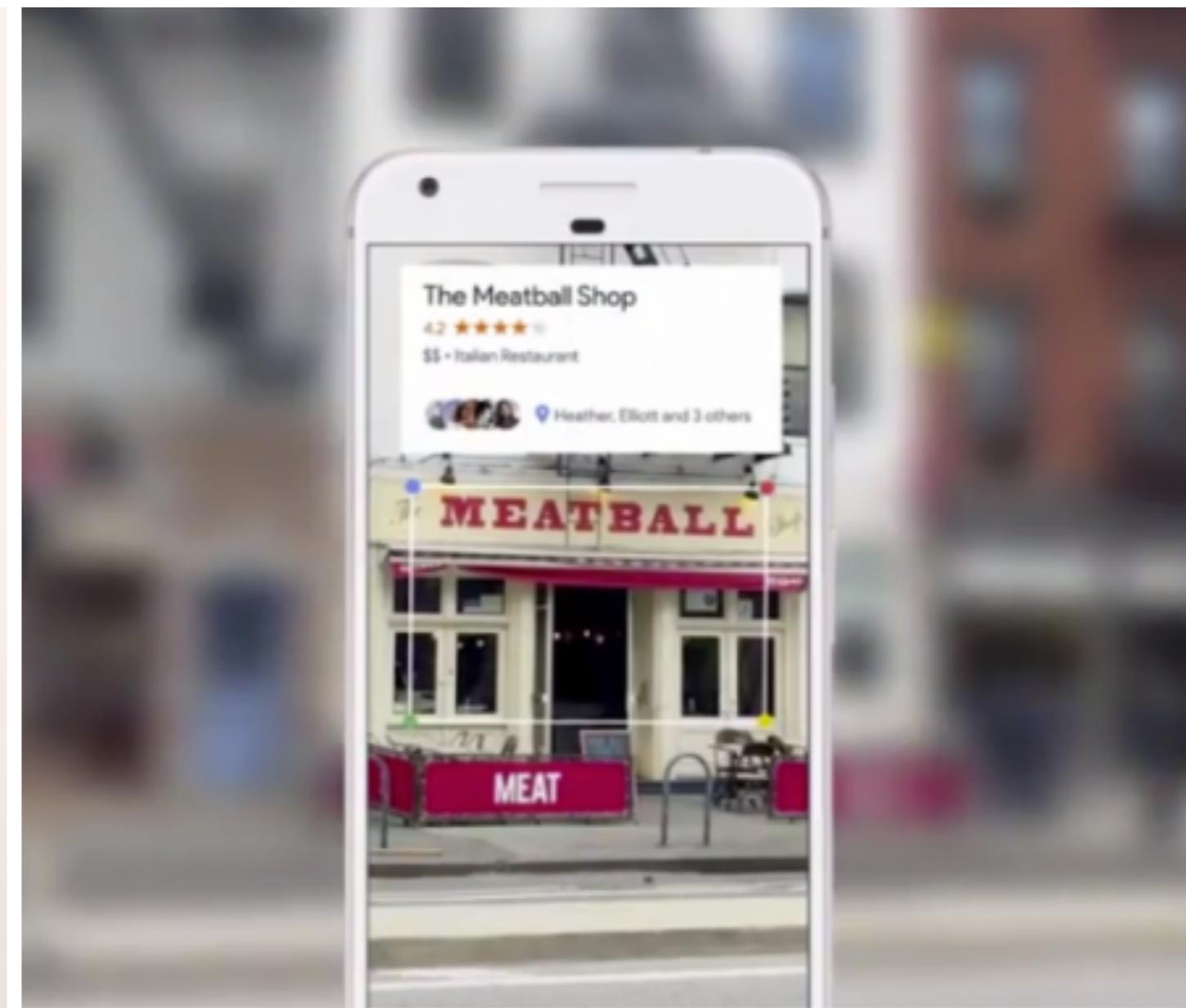
## 19. 사물 감별사

구글 렌즈 - 사물을 보여주면 이해하고 정보 전달. 구글 어시스턴트와 포토에 도입

## 20. 대선 뉴스 큐레이터

19대 대통령 선거 - 메인 화면과 뉴스 섹션에 '루빅스' 적용

## 고미, 구글렌즈



### 21. 난민 심리치료

스타트업 'X2AI' - 정신적 고통을 겪는 난민을 위한 챗봇 '카림' 개발

### 22. CCTV

국내 스타트업 '마인드셋' - '마인드아이': 하드웨어 없이 상황, 물체 식별

### 23. 영화 예고편 제작

'모건' 예고편 - 왓슨 100여편의 공포영화 학습.

제작 기간을 1개월에서 24시간으로 단축

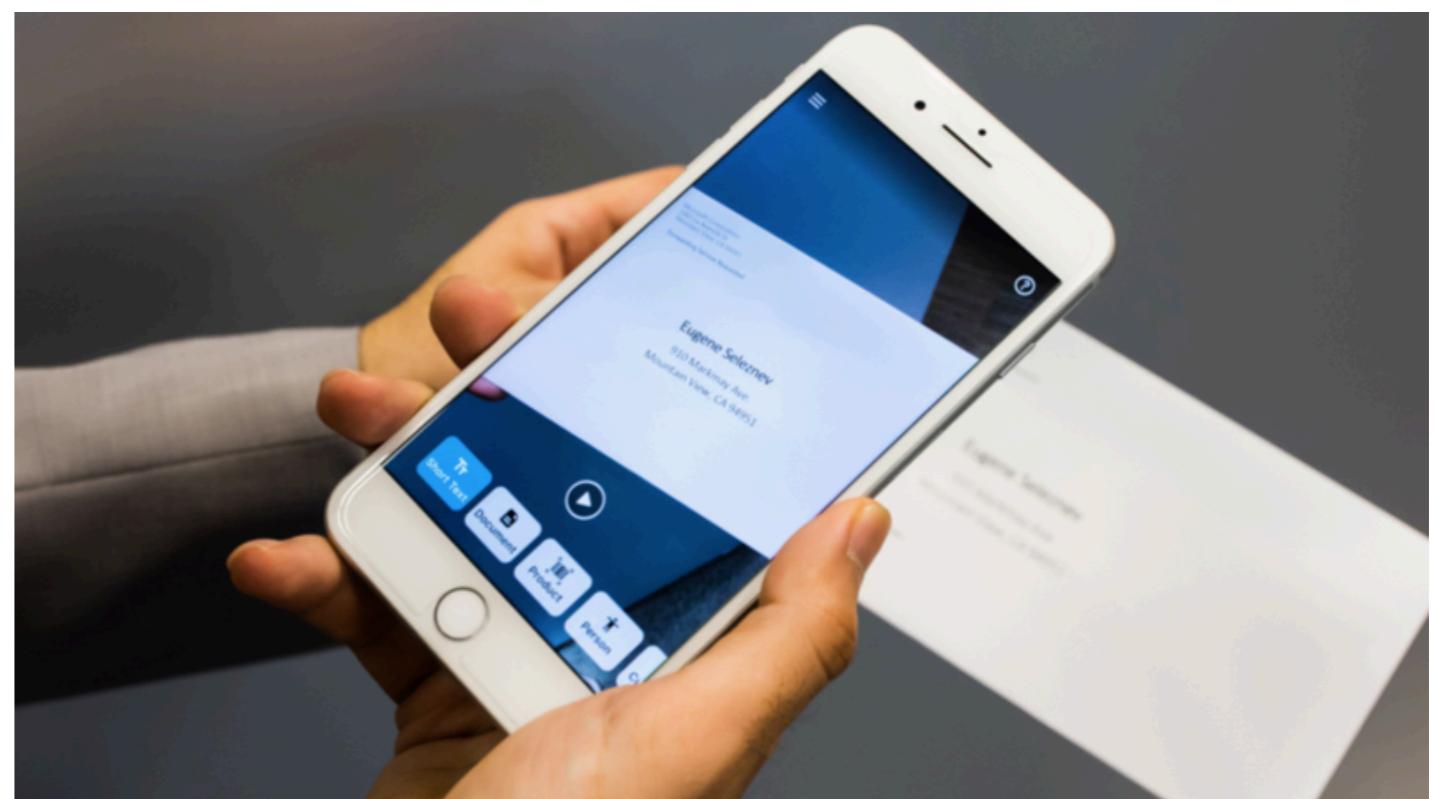
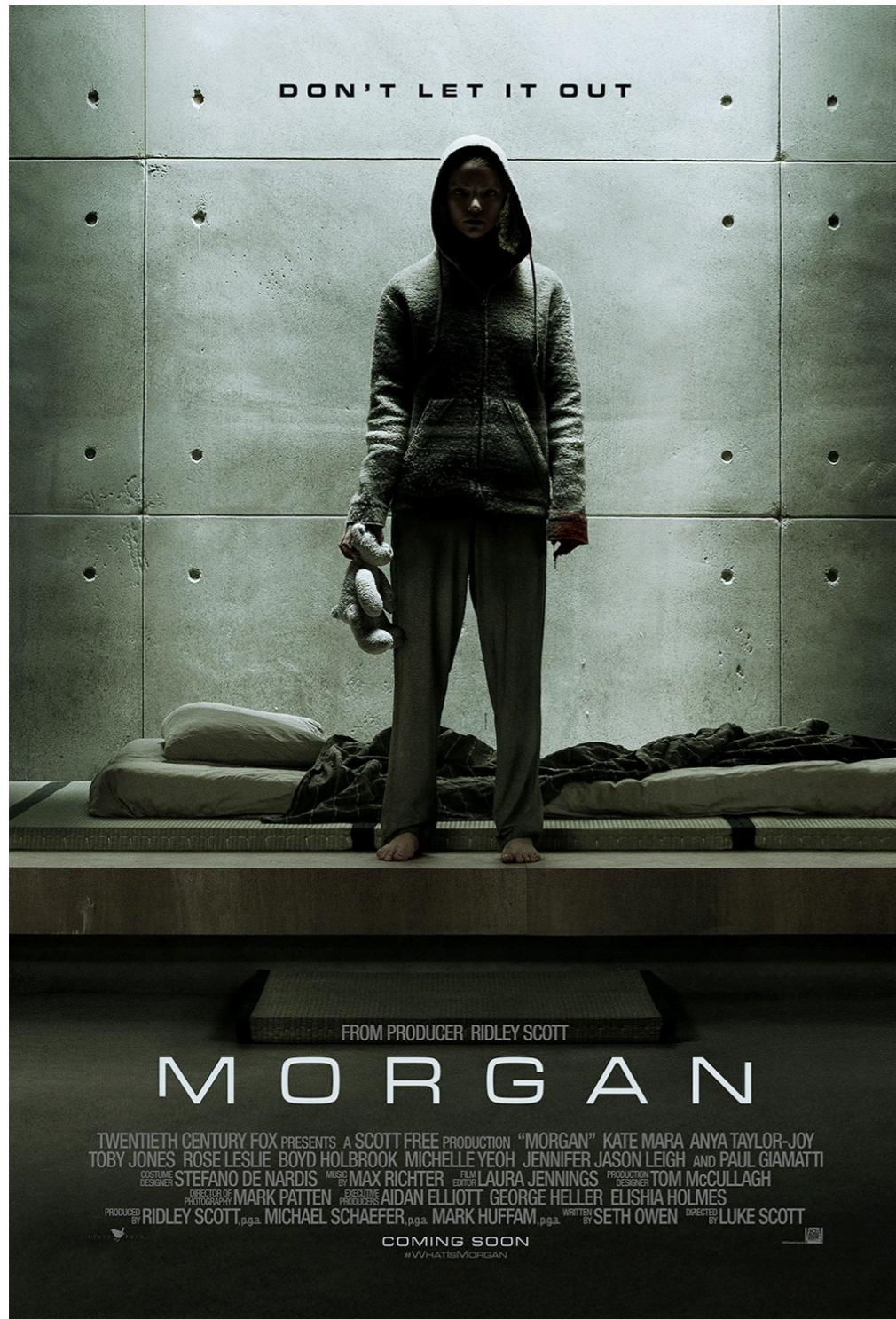
### 24. 경주용 차

무인 경주용 차량 '로보카' - 드라이버가 탑승하지 않는 '로보레이스' 출전

### 25. 시각장애인의 눈

마이크로소프트 '씨잉 AI' - 시각장애인에게 주변 환경, 인물, 사물 설명

## 모건, 로보카, SEEING AI



### 26. 명품가방 판별

스타트업 '엔트루피' - 3만여종의 핸드백 지갑을 98% 정확도로 판별

### 27. 이유식 재료 선정

일본 식료품업체 '큐피' - 400개 이상의 5톤 식재료에서 재료 판별

### 28. 자연재해 예측

오재호 부경대 교수팀 - 기상변화 예측 '알파멧' 개발, 한국지 지형 데이터 활용.

### 29. 매장 레이아웃 개선

인컨텍스트 솔루션스 - 인공지능과 VR을 조합해서 매장 레이아웃 구성

### 30. 워터마크 제거

구글에서 워터마크를 제거하는 논문 공개

## 워터마크 제거



### 31. 음란물 필터

네이버 음란물 필터 기술 '네이버 X-eye' - 모든 이미지에 대해 적용(98.1%)

### 32. 꽃가루 알레르기 위험 지수 관리

기상청 '꽃가루 농도위험지수'에 AI 적용 - 15.9%에서 69.4%로 개선

### 33. 항만 관리

일본 국토교통성 - 공장 출하, 도로/항만 혼잡도, 선박 도착시간 처리

### 34. 상어 감지

무인항공기 업체 '리틀 리퍼' - 인공지능 드론 사용. 20%에서 90%로 향상

### 35. 폐기물 분류

행정안전부 - AI 객체인식 기반 대형 폐기물 처리시스템 구축 사업 추진(은평구)

## 상어 감지



## 36. 치매 예측

캐나다 맥길대 - 치매 발생 2년 전에 예측 가능(84% 정확도)

## 37. 심정지 예측

호흡수, 심장박동수, 산소포화도, 혈압 등의 데이터 학습

현재 기술로는 30분 전에 예측 - 인공지능은 24시간 전에 가능(70% 이상)

## 38. 작곡

구글 '마젠타 프로젝트' - 기계가 예술을 창조할 수 있는지 알아보는 프로젝트

엔신스 - 1천개의 악기, 30만개의 데이터베이스로부터 새로운 소리 및 음악 생성

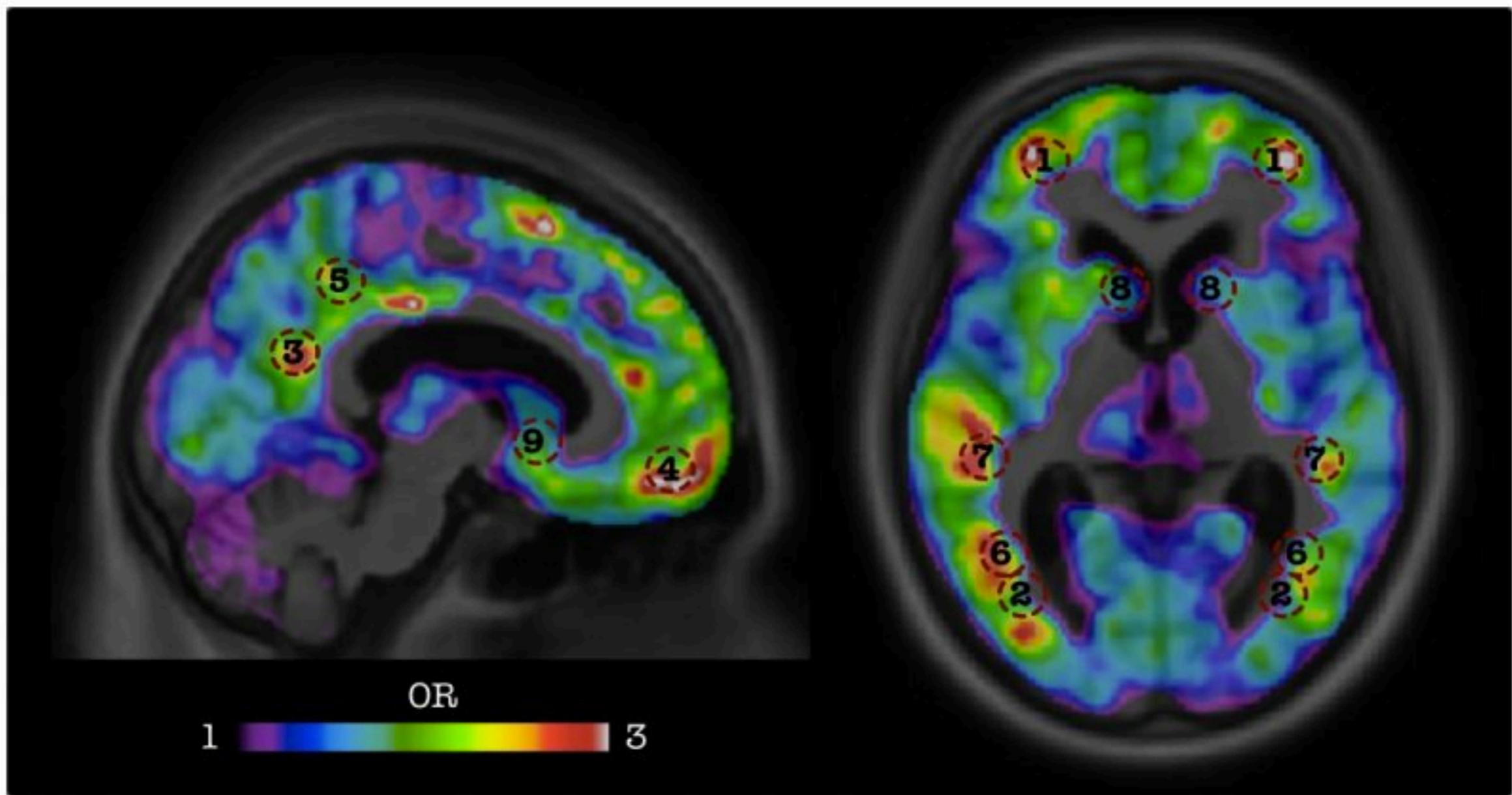
## 39. 시신경 질환 예측

김안과 병원 - 시신경 질환 예측 연구. 녹내장 진단 100% 정확도 달성

## 40. 저작권 침해 예방

한국저작권보호원 - 불법복제 영상 유통 차단에 활용

## 치매 예측, 마젠타



## 41. 졸음운전 예측

다이이치교통산업 - 운전자의 심박수, 운전자 태도, 주행 모습 데이터 수집

## 42. 다이어트 분야

비만치료 전문기업 '365cm네트웍스' - 인공지능 흡입기술 'MAIL 시스템'  
지방흡입술 집도의의 전체 수술 동작 저장 및 분석

## 43. 영상 조작

워싱턴 대학교 - 음성에 맞춘 '립싱크' 개발. 음성만 같은 다른 영상 제작

## 44. 드레스 제작

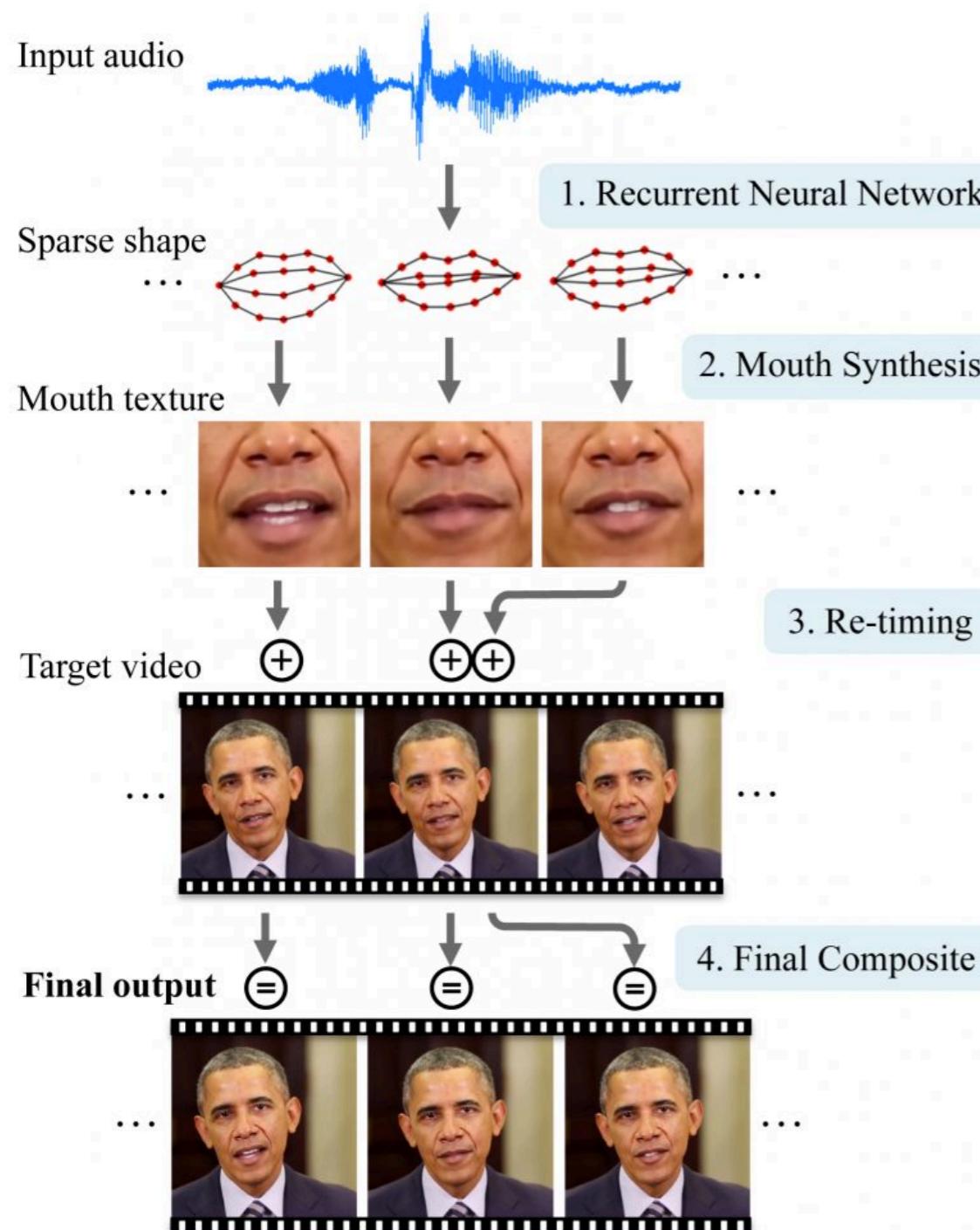
'마르케사' - IBM과 협업. 인공지능을 감정을 표현한 드레스 제작

## 45. 번역기

구글 번역, 네이버 파파고, 시스트란

인간 vs. 인공지능 번역 대결 - 인간이 승리했지만, 평가와 번역 환경 공정성 논란

## 영상 조작, 번역기



Google

번역

영어 프랑스어 한국어 언어 감지

한국어 영어 일본어 번역하기

0/5000

papago

언어감지 ▾

번역할 내용을 입력하세요.

한국어 ▾

0 / 5000 번역하기

### 46. 채용 도우미

리쿠르트 '헤이스' - 인력 정보에 바탕해서 헤드 헌터의 업무 경감

### 47. 목소리 재현

스타트업 '라이어버드' - 60초의 음성 데이터로 목소리 재현. 감정 표현 가능

### 48. 신용카드 거래 승인

마스터카드 - '디시전 인텔리전스'. 고객 개별 거래 평가, 점수, 학습

모든 거래를 분석하고 산출된 정보를 바탕으로 승인 여부 결정

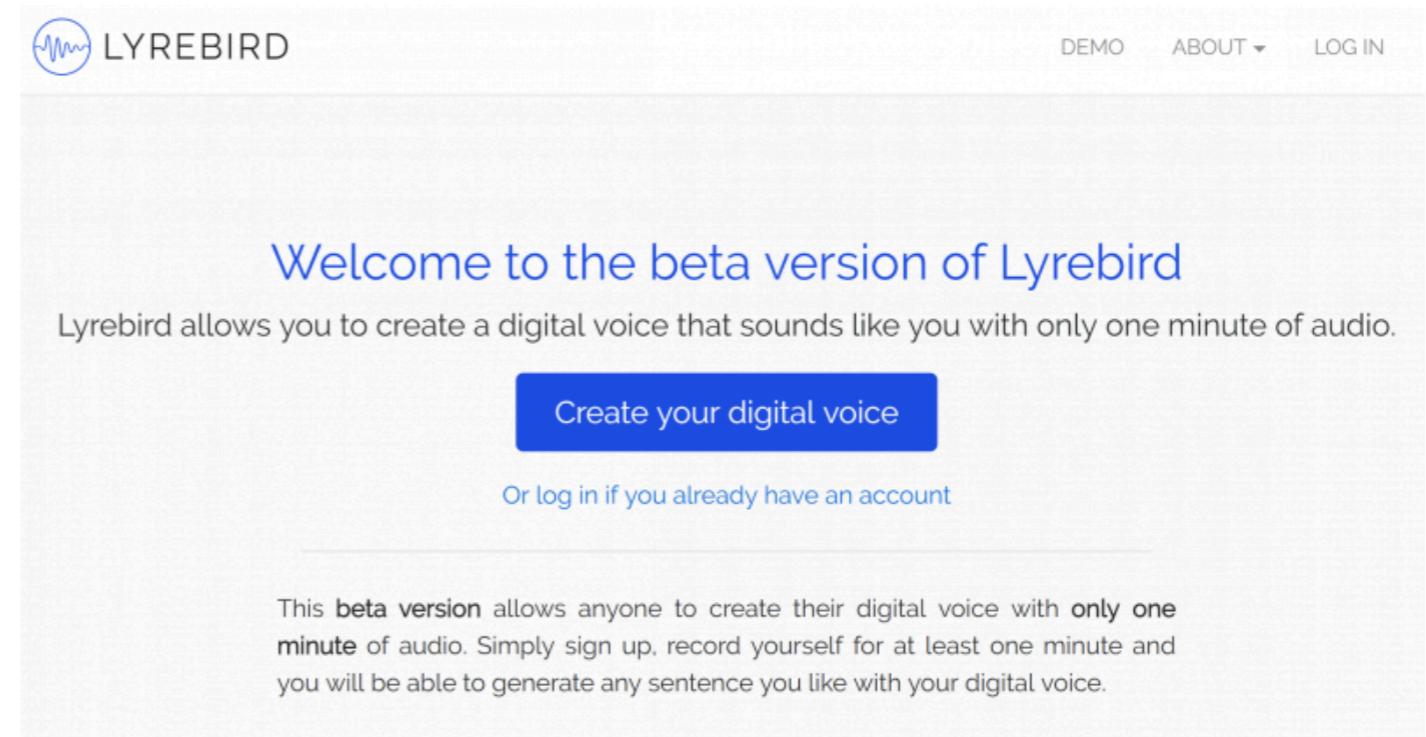
### 49. 영화 선호도 예측

디즈니 리서치팀 - 단편 이야기를 평가할 수 있는 신경망 연구

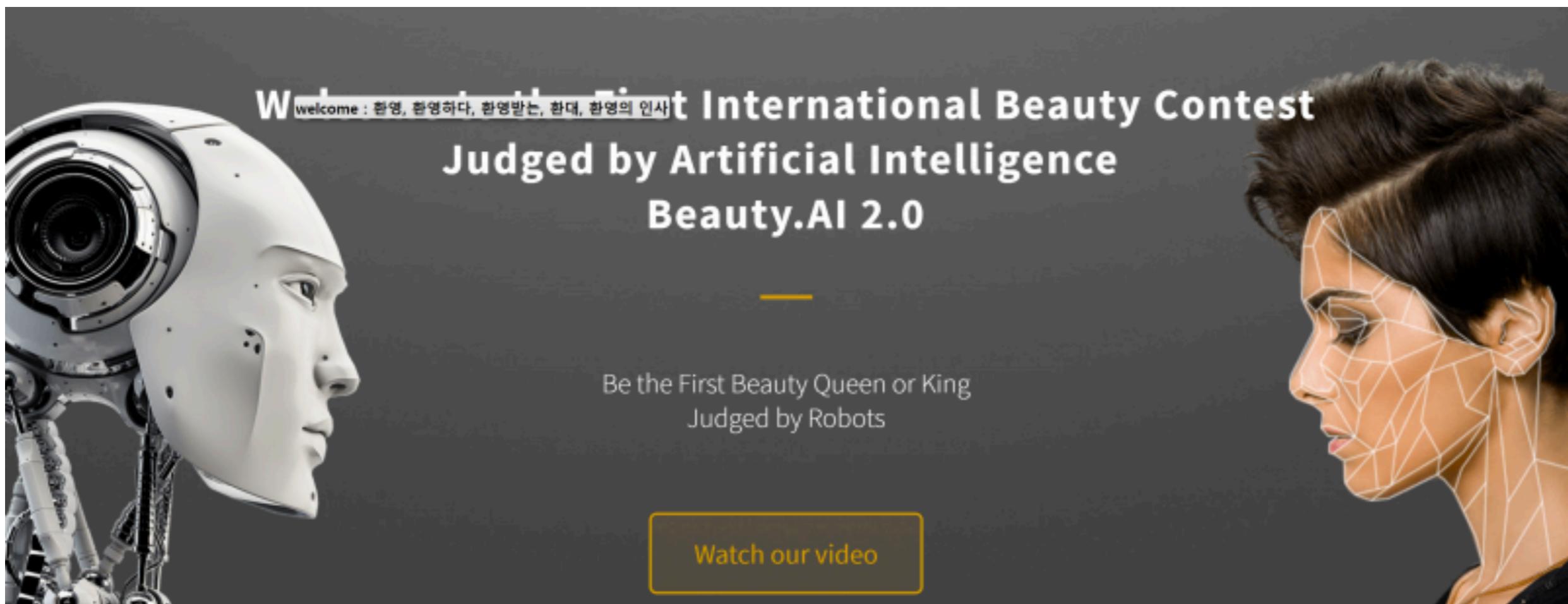
### 50. 미인대회 심사위원

'뷰티닷에이아이' - 로봇 판정단으로만 구성된 미인대회. 백인 편향 알고리즘 논란

## 미인대회, 목소리 재현



The image shows the landing page for the Lyrebird beta version. At the top right are links for DEMO, ABOUT, and LOG IN. The main heading is "Welcome to the beta version of Lyrebird". Below it is a subtext: "Lyrebird allows you to create a digital voice that sounds like you with only one minute of audio." A blue button labeled "Create your digital voice" is prominent. Below the button is a link "Or log in if you already have an account". A descriptive text block states: "This beta version allows anyone to create their digital voice with only one minute of audio. Simply sign up, record yourself for at least one minute and you will be able to generate any sentence you like with your digital voice."



The image shows the landing page for the Beauty.AI 2.0 contest. It features a large background image of a woman's face with a geometric overlay and a white robot head on the left. The title text is "Welcome : 환영, 환영하다, 환영받는, 환대, 환영의 인사" followed by "International Beauty Contest Judged by Artificial Intelligence Beauty.AI 2.0". Below the title is a subtitle: "Be the First Beauty Queen or King Judged by Robots". A yellow button at the bottom center says "Watch our video".



# DEEP LEARNING --- OVERVIEWS

## 머신러닝의 확산 - 폭발적인 증가

- ▶ 무어의 법칙(Moore's Law)으로 컴퓨팅 비용이 급격히 낮아져 지금은 최소한의 비용으로 강력한 컴퓨팅 성능을 폭넓게 이용할 수 있다.
- ▶ 새롭고 혁신적인 알고리즘이 더욱 빠른 결과를 제공한다.
- ▶ 데이터 과학자들이 머신러닝을 효과적으로 적용하기 위한 이론과 실무 지식을 축적했다.

# PROGRAMMING LANGUAGES - 나무위키

### ▶ R 언어

현재 가장 많이 쓰이는 통계 기반 프로그래밍 언어이다. R 언어의 강점은 간단한 코딩으로 충분히 인지 가능할 정도로 시각화된 데이터를 얻어내는 것에 있다. 이를 통해 개발 파이프라인을 단축시키는데 많은 기여를 한다.

### ▶ 파이썬

R 언어에 이어 이 분야에서 두 번째로 많이 쓰이는 언어이며, numpy 라이브러리를 써서 기계학습 알고리즘을 코딩하도록 도와준다. 비록 R 언어보다 코딩에 다소 시간이 걸리지만, 파이썬 언어의 대표적 장점인 이식성이 있어 다양한 분야에서 사용되어지며, scipy 라이브러리 등을 추가하여 내부 변수의 계산을 하거나 Cython 등을 이용하여 알고리즘의 속도를 빠르게 하기에도 용이하다.

### ▶ Matlab

수학적 정밀도가 어느 정도 보장되는 언어이다보니 주로 연구실 등에서 사용되어진다.

# 통계, 머신러닝, 데이터마이닝

## ▶ 통계 vs. 머신러닝

- ▶ 통계는 분포나 가정을 사용해서 엄격한 규칙이 적용되는 설문조사나 실험 계획에 사용됨
- ▶ 머신러닝은 대용량 데이터의 분석이나 패턴을 찾는데 사용됨

## ▶ 머신러닝 vs. 데이터마이닝

- ▶ 머신러닝은 훈련 데이터를 통해 학습된 알려진 속성을 기반으로 한 예측에 중점
- ▶ 데이터마이닝은 데이터의 미처 몰랐던 속성을 발견하는 것에 집중. 이는 데이터베이스의 지식 발견 부분의 분석 절차에 해당한다.
- ▶ 이들은 방법적으로 중복되는 부분이 있다. 데이터마이닝에서 머신러닝은 필수가 아니지만, 머신러닝에서는 데이터마이닝이 필수라는 부분이 다르다.

# 인공지능, 머신러닝, 딥 러닝

## ▶ 인공지능

- ▶ 외부 관찰자에게 인간처럼 스마트하게 소프트웨어를 작동시키는 폭넓은 방법, 알고리즘 및 기술
- ▶ 머신러닝, 컴퓨터 비전, 자연어 처리, 로봇 공학 및 그와 관련된 모든 주제를 포괄하는 개념

## ▶ 머신러닝

- ▶ 더 많은 데이터 축적을 통해 성능을 개선할 수 있도록 하는 다양한 알고리즘과 방법론
- ▶ 신경망, 서포트 벡터 머신, 결정 트리, 베이지안 신뢰 네트워크, k 최근접 이웃, 자기 조직화 지도, 사례 기반 추론, 인스턴스 기반 학습, 은닉 마르코프 모델, 회귀 기법

## ▶ 딥 러닝

- ▶ 신경망(**Neural Network**)을 부르는 다른 이름
- ▶ 여러 개의 히든 레이어를 통해 깊게 학습한다고 해서 붙여진 이름

## 정의 - TOM M. MITCHELL, CARNEGIE MELLON UNIVERSITY



- ▶ A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
- ▶ 태스크(T)에 대해 꾸준한 경험(E)을 통해 T에 대한 성능(P)을 높이는 것을 기계학습이라고 한다.
- ▶ 기계학습에서 가장 중요한 것은 E에 해당하는 데이터이다. 좋은 품질의 데이터를 많이 가지고 있다면 보다 높은 성능을 끌어낼 수 있다.

## 분류

- ▶ 지도 학습(Supervised Learning)
  - ▶ 회귀(Regression)
  - ▶ 분류(Classification)
- ▶ 비지도 학습(Unsupervised Learning)
  - ▶ 군집화(Clustering)
  - ▶ 분포 추정(Underlying Probability Density Estimation)
- ▶ 강화 학습(Reinforcement Learning)

## 둘러보기

---

### ▶ 예측

Linear regression, Regression tree, Kernel regression, Support vector regression

### ▶ 분류

Logistic regression, Decision tree, Nearest-neighbor classifier, Kernel discriminant analysis, Neural network, Support Vector Machine, Random forest, Boosted tree

### ▶ 차원(변수) 축소

Principal component analysis, Non-negative matrix factorization, Independent component analysis, Manifold learning, SVD

### ▶ 그룹화

k-means, Hierarchical clustering, mean-shift, self-organizing maps(SOMs)

### ▶ 선행학습(Pre-training), 2차분류

Deep Learning(Stacked Restricted Boltzmann Machine, Stacked Auto-Encoders 등을 사용한 Multi layers Neural Nets, Non-linear Transformation)

### ▶ 데이터 비교

Bipartite cross-matching, n-point correlation two-sample testing, minimum spanning tree

# SUPERVISED LEARNING (1)

### ▶ 서포트 벡터 머신 (support vector machine)

패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다. 두 카테고리 중 어느 하나에 속한 데이터의 집합이 주어졌을 때, SVM 알고리즘은 주어진 데이터 집합을 바탕으로 하여 새로운 데이터가 어느 카테고리에 속할지 판단하는 비확률적 이진 선형 분류 모델을 만든다. 만들어진 분류 모델은 데이터가 사상된 공간에서 경계로 표현되는데 SVM 알고리즘은 그 중 가장 큰 폭을 가진 경계를 찾는 알고리즈다.

### ▶ 은닉 마르코프 모델 (Hidden Markov model)

통계적 마르코프 모델의 하나로, 시스템이 은닉된 상태와 관찰가능한 결과의 두 가지 요소로 이루어졌다고 보는 모델이다. 관찰 가능한 결과를 야기하는 직접적인 원인은 관측될 수 없는 은닉 상태들이고, 오직 그 상태들이 마르코프 과정을 통해 도출된 결과들만이 관찰될 수 있기 때문에 '은닉'이라는 단어가 붙게 되었다. 음성 인식, 필기 인식, 동작 인식, 품사 태깅, 약보에서 연주되는 부분을 찾는 작업, 부분 방전, 생물정보학과 같이 시간의 영향을 받는 시스템의 패턴을 인식하는 작업에 유용한 것으로 알려져있다.

# SUPERVISED LEARNING (2)

### ▶ 회귀 분석 (Regression)

통계학에서, 회귀분석(regression analysis)은 관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한뒤 적합도를 측정해 내는 분석 방법이다. 회귀분석은 시간에 따라 변화하는 데이터나 어떤 영향, 가설적 실험, 인과 관계의 모델링등의 통계적 예측에 이용될 수 있다.

### ▶ 신경망 (Neural network)

인공신경망은 생물학의 신경망(특히 뇌)에서 영감을 얻은 통계학적 학습 알고리즘이다. 인공신경망은 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)이 학습을 통해 시냅스의 결합 세기를 변화시켜, 문제 해결 능력을 가지는 모델 전반을 가리킨다. 좁은 의미에서는 역전파법을 이용한 다층 퍼셉트론을 가리키는 경우도 있지만, 인공신경망은 이에 국한되지 않는다.

### ▶ 나이브 베이즈 분류 (Naive Bayes Classification)

특성들 사이의 독립을 가정하는 베이즈 정리를 적용한 확률 분류기의 일종으로 1950년대 이후 광범위하게 연구되고 있다. 텍스트 분류에 사용됨으로써 문서를 여러 범주 (예: 스팸, 스포츠, 정치)중 하나로 판단하는 문제에 대한 대중적인 방법으로 남아있다.

# UNSUPERVISED LEARNING

### ▶ 클러스터링(Clustering)

개체를 다수의 메트릭스에서 상호 유사한 세그먼트 또는 클러스터로 그룹화하는 기법. 고객 세분화가 클러스터링의 실제 예다. 클러스터링 알고리즘은 무척 다양한데, 가장 널리 사용되는 것이 k-평균(k-means)이고 비슷한 알고리즘으로 Gaussian Mixture Model도 있다.

### ▶ 독립 성분 분석(Independent Component Analysis)

다변량의 신호를 통계적으로 독립적인 하부 성분으로 분리하는 계산 방법이다. 각 성분은 비 가우스 성 신호로서 서로 통계적 독립을 이루는 성분으로 구성되어 있다. 독립 성분 분석은 블라인드 신호를 분리하는 특별한 방법이다.

독립 성분 분석의 전형적인 알고리즘은 복잡성을 줄이기 위한 전 단계로서 중심화 (centering), 백색화 (whitening), 차원 감소 (dimensionality reduction) 등의 과정이 필요하다. 백색화와 차원 감소는 주 성분 분석 (Principal Component Analysis)과 특이 값 분해 (Singular Value Decomposition)로 한다.

## 전국주지사협회 연설 : ELON MUSK

NGA 2017 SUMMER MEETING – Introducing the New Chair's Initiative "A... ⏲ ➔

**2017**  
**SUMMER MEETING**

**"AHEAD OF THE CURVE"**

**Speaker: Elon Musk**

Saturday July 15th  
1:45 pm ET

## ELON MUSK VS. MARK ZUCKERBERG

### ▶ Elon Musk

우리가 AI 규제에 민감하게 반응할 때, 그때는 이미 너무 늦었을 것이다. 규제는 일반적으로 나쁜 사건이 뭉텅이로 발생하고 절규에 가까운 대중의 요구가 빗발친 다음에야 만들어지기 시작한다. 그리고 다시 몇 년 후에야 규제 기관이 정비된다. 시간이 너무 오래 걸린다.

### ▶ Mark Zuckerberg

(AI에 대해) 부정적인 시각을 가지고 둠스데이(최후 종말의 날) 시나리오를 펼치는 사람들, 나는 이 사람들을 이해할 수 없다. 나는 이 문제에 대해 꽤 확고한 입장이다. 나는 낙관적이다. 기술은 언제나 좋거나 나쁜 쪽으로 사용될 수 있었다.

## ELON MUSK VS. MARK ZUCKERBERG

홈 회사소개 트위터 검색하기 이미 트위터에 가입하셨나요? 로그인 ▾

Darren Cunningham @dcunni · 7월 25일  
Zuckerberg blasts @elonmusk warnings against artificial intelligence as 'pretty irresponsible' [bizjournals.com/sanjose/news/2...](http://bizjournals.com/sanjose/news/2...) @svbizjournal #ai



Facebook CEO Mark Zuckerberg blasts Tesla CEO Elon Musk's warning about AI. "People who are naysayers and try to drum up these doomsday scenarios — I just, I don't understand it," the Facebook CEO said. "It's really negative." [bizjournals.com](http://bizjournals.com)

113 789 1,541

Elon Musk ✅  
@elonmusk

@dcunni 님, @SVbizjournal 님에게 보내는 답글

I've talked to Mark about this. His understanding of the subject is limited.

오전 12:07 - 2017년 7월 25일

팔로우

트위터에 처음이세요?  
지금 가입하여 타임라인을 원하는 대로 설정해보세요.

가입하기

회사소개 트위터

## MARK ZUCKERBERG

- ▶ 내가 AI에 낙관적인 이유 중 하나는 (AI가) 다양한 분야의 시스템을 개선하기 위한 기초 연구를 더 잘할 수 있게 한다는 점이다. 질병 진단에서부터 건강 유지, 안전한 자율주행차, 당신의 뉴스피드에 더 나은 콘텐츠를 보여주는 것, 더 연관성이 높은 검색 결과를 제공하는 것까지 여러 분야의 시스템을 개선할 수 있다. 우리가 우리의 AI 방법을 향상시킬 때마다 매번 이 모든 시스템이 더 나아진다. 이는 세상을 더 좋게 만들 가능성이 있다.



# DEEP LEARNING

---

# TERMS

## 1. 회귀분석 ([HTTP://MATH7.TISTORY.COM/118](http://MATH7.TISTORY.COM/118)에서 발췌)

- ▶ 점들이 퍼져있는 형태에서 패턴을 찾아내고, 이 패턴을 활용해서 무언가를 예측하는 분석.
- ▶ 새로운 표본을 뽑았을 때 평균으로 돌아가려는 특징이 있기 때문에 붙은 이름
- ▶ 회귀(回歸 돌회, 돌아갈 귀)라는 용어는 일반적으로 '돌아간다'는 정도로만 사용하기 때문에 회귀로부터 '예측'이라는 단어를 떠올리기는 쉽지 않다.

## 2. LINEAR REGRESSION

- ▶ 2차원 좌표에 분포된 데이터를 1차원 직선 방정식을 통해 표현되지 않은 데이터를 예측하기 위한 분석 모델.
- ▶ 머신러닝 입문에서는 기본적으로 2차원이나 3차원까지만 정리한다.
- ▶ 여기서는 편의상 1차원 직선으로 정리하고 있다. xy축 좌표계에서 직선을 그렸다고 생각하면 된다.

### 3. HYPOTHESIS

- ▶ **Linear Regression**에서 사용하는 1차원 방정식을 가리키는 용어로, 우리말로는 가설이라고 한다. 수식에서는  $h(x)$  또는  $H(x)$ 로 표현된다.
- ▶ 최저점(**minimize cost**)이라는 정답을 찾기 위한 가정이기 때문에 가설이라고 부를 수 있다.
- ▶  $H(x) = Wx + b$   
==>  $x$ 에 대한 1차 방정식

## 4. COST (비용)

- ▶ 앞에서 설명한 **Hypothesis** 방정식에 대한 비용(cost)으로 방정식의 결과가 크게 나오면 좋지 않다고 얘기하고 루프를 돌 때마다  $W$ 와  $b$ 를 비용이 적게 발생하는 방향으로 수정하게 된다.
- ▶ 미분을 사용해서 스스로 최저 비용을 찾아간다.
- ▶ **Gradient Descent Algorithm**을 사용해서 최저 비용을 찾는다.

## 5. COST 함수

- ▶ **Hypothesis** 방정식을 포함하는 계산식
- ▶ 현재의 기울기( $w$ )와 절편( $b$ )에 대해 비용을 계산해 주는 함수
- ▶  $w$ 와  $b$ 가 변함에 따라 반드시 **convex**(오목)한 형태로 설계되어야 하는 것이 핵심.
- ▶ **convex**하지 않다면, 경사를 타고 내려갈 수 없기 때문에 최저점 계산이 불가능해질 수 있다.
- ▶ **Linear Regression**을 비롯한 머신러닝 전체에서 최소 비용을 검색하기 위한 역할 담당

## 6. GRADIENT DESCENT ALGORITHM

- ▶ 딥러닝의 핵심 알고리즘
- ▶ 경사타고 내려가기, 경사하강법 등의 여러 용어로 번역되었다.
- ▶ 미분을 사용해서 비용이 작아지는 방향으로 진행하는 알고리즘
- ▶ 생각보다 어렵지 않고 간단한 미분 정도만 이해하면 알고리즘 자체는 너무 단순하다.
- ▶ 텐서플로우에 포함된 Optimizer는 대부분 Gradient Descent Algorithm에서 파생된 방법을 사용하고 있다.



DEEP LEARNING

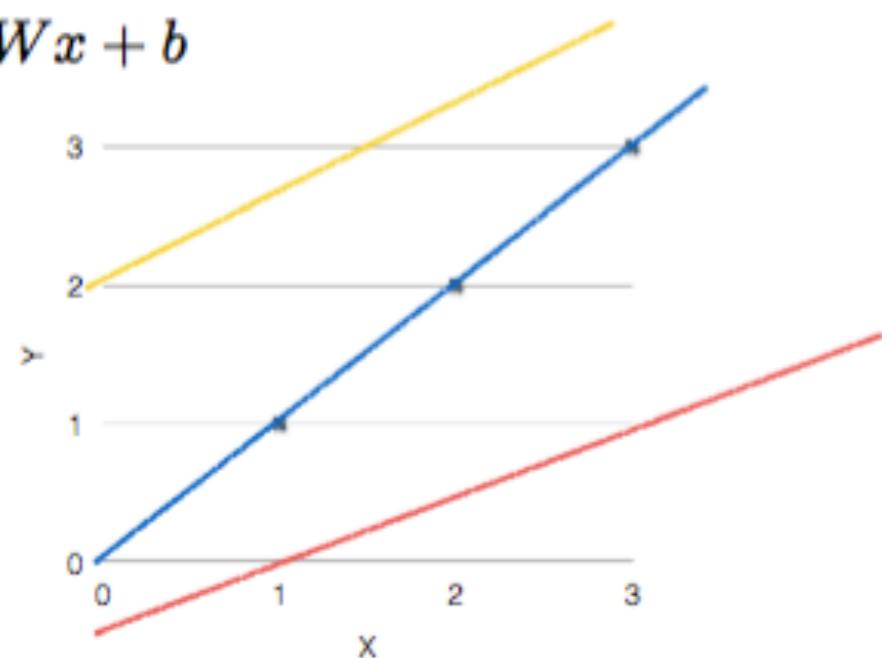
---

LINEAR  
REGRESSION

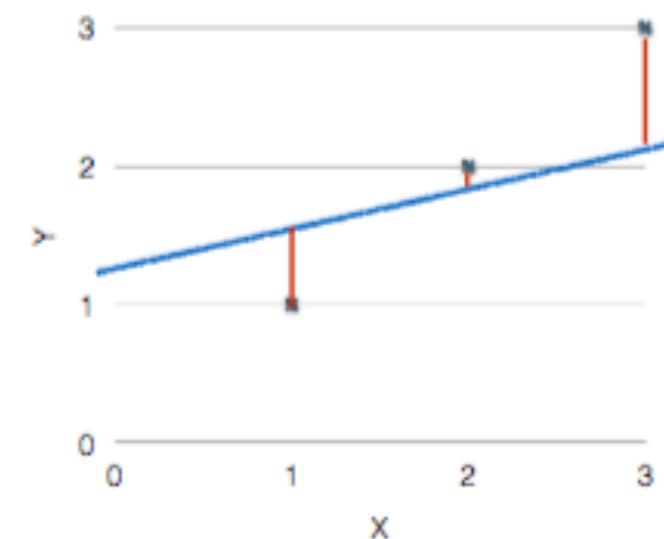
## 좋은 가설?

(Linear) Hypothesis

$$H(x) = Wx + b$$



Which hypothesis is better?



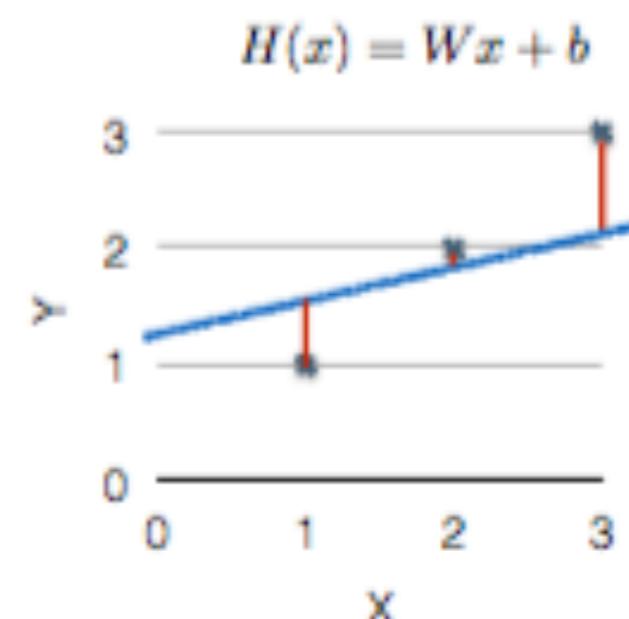
## 직선으로부터 점까지의 거리 계산

### Cost function

- How fit the line to our (training) data

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$



## COST 수식

$$cost = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

### 최저 COST를 만드는 W와 B는?

- ▶  $H(x) = wx + b$
- ▶  $x = [1, 2, 3]$   
 $y = [1, 2, 3]$
- ▶ w와 b가 아래와 같다면?
  1.  $w = 1/2, b = 0$
  2.  $w = 0, b = 2$
  3.  $w = 1, b = 1/2$
  4.  $w = 1, b = 1$

# THE GOAL OF LINEAR REGRESSION

- ▶ Cost를 최소로 만드는  
W(Weight)와 b(bias)를 찾는 것.
- ▶ Linear Regression의 목표는 곧 머신러닝의 목표!



DEEP LEARNING

---

GRADIENT DESCENT  
ALGORITHM

## HOW TO HIDE BIAS?

Hypothesis and Cost

$$H(x) = Wx + b$$

$$\text{cost}(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

Simplified hypothesis

$$H(x) = Wx$$

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

# COST FUNCTION GRAPH

**What  $\text{cost}(W)$  looks like?**

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

X	Y
1	1
2	2
3	3

- $W=1, \text{cost}(W)=0$

$$\frac{1}{3}((1 * 1 - 1)^2 + (1 * 2 - 2)^2 + (1 * 3 - 3)^2)$$

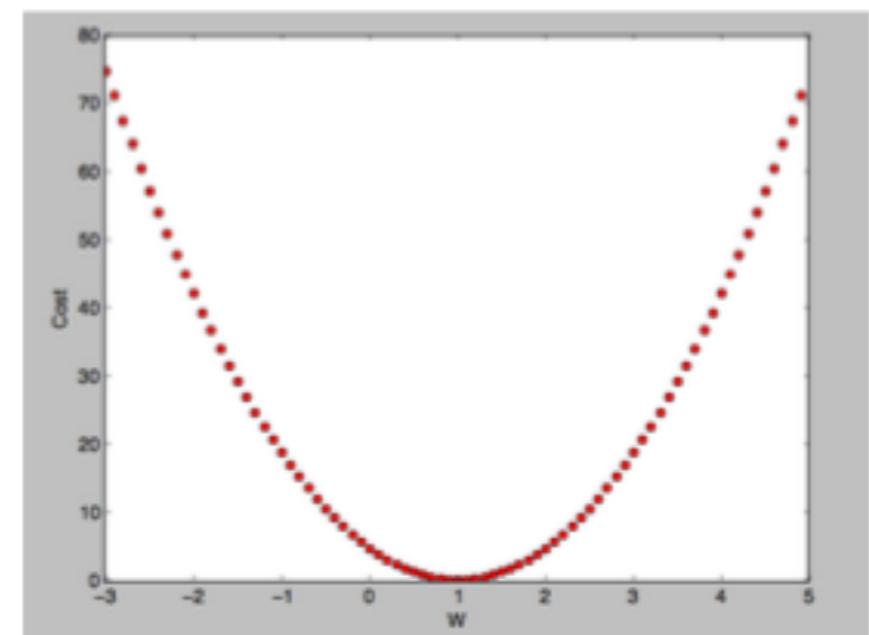
- $W=0, \text{cost}(W)=4.67$

$$\frac{1}{3}((0 * 1 - 1)^2 + (0 * 2 - 2)^2 + (0 * 3 - 3)^2)$$

- $W=2, \text{cost}(W)=?$

**How to minimize cost?**

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

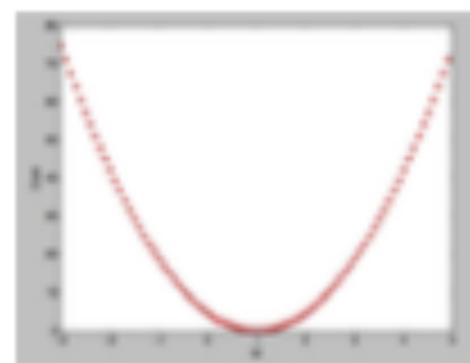


## GRADIENT DESCENT ALGORITHM

- Minimize cost function
- Gradient descent is used many minimization problems
- For a given cost function,  $\text{cost}(W, b)$ , it will find  $W, b$  to minimize cost
- It can be applied to more general function:  $\text{cost}(w_1, w_2, \dots)$

### HOW IT WORKS?

- Start with initial guesses
  - Start at 0,0 (or any other value)
  - Keeping changing  $W$  and  $b$  a little bit to try and reduce  $\text{cost}(W, b)$
- Each time you change the parameters, you select the gradient which reduces  $\text{cost}(W, b)$  the most possible
- Repeat
- Do so until you converge to a local minimum
- Has an interesting property
  - Where you start can determine which minimum you end up



## FORMAL DEFINITION

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

$$W := W - \alpha \frac{\partial}{\partial W} \frac{1}{2m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

$$\text{cost}(W) = \frac{1}{2m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

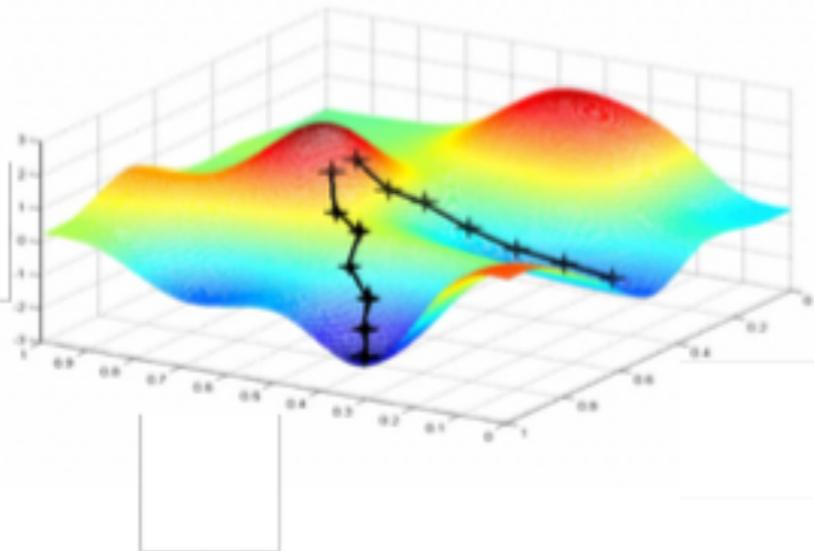
$$W := W - \alpha \frac{1}{2m} \sum_{i=1}^m 2(Wx^{(i)} - y^{(i)})x^{(i)}$$

$$W := W - \alpha \frac{\partial}{\partial W} \text{cost}(W)$$

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})x^{(i)}$$

# CONVEX FUNCTION

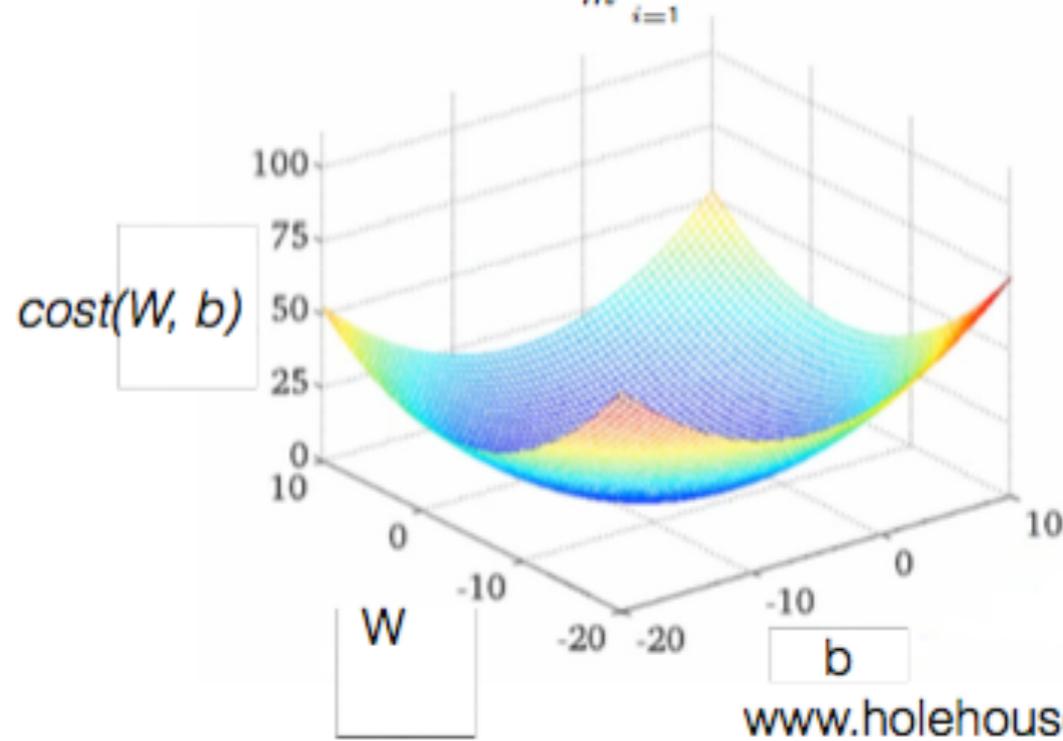
**Convex function**



[www.holehouse.org/mlclass/](http://www.holehouse.org/mlclass/)

**Convex function**

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$



[www.holehouse.org/mlclass/](http://www.holehouse.org/mlclass/)



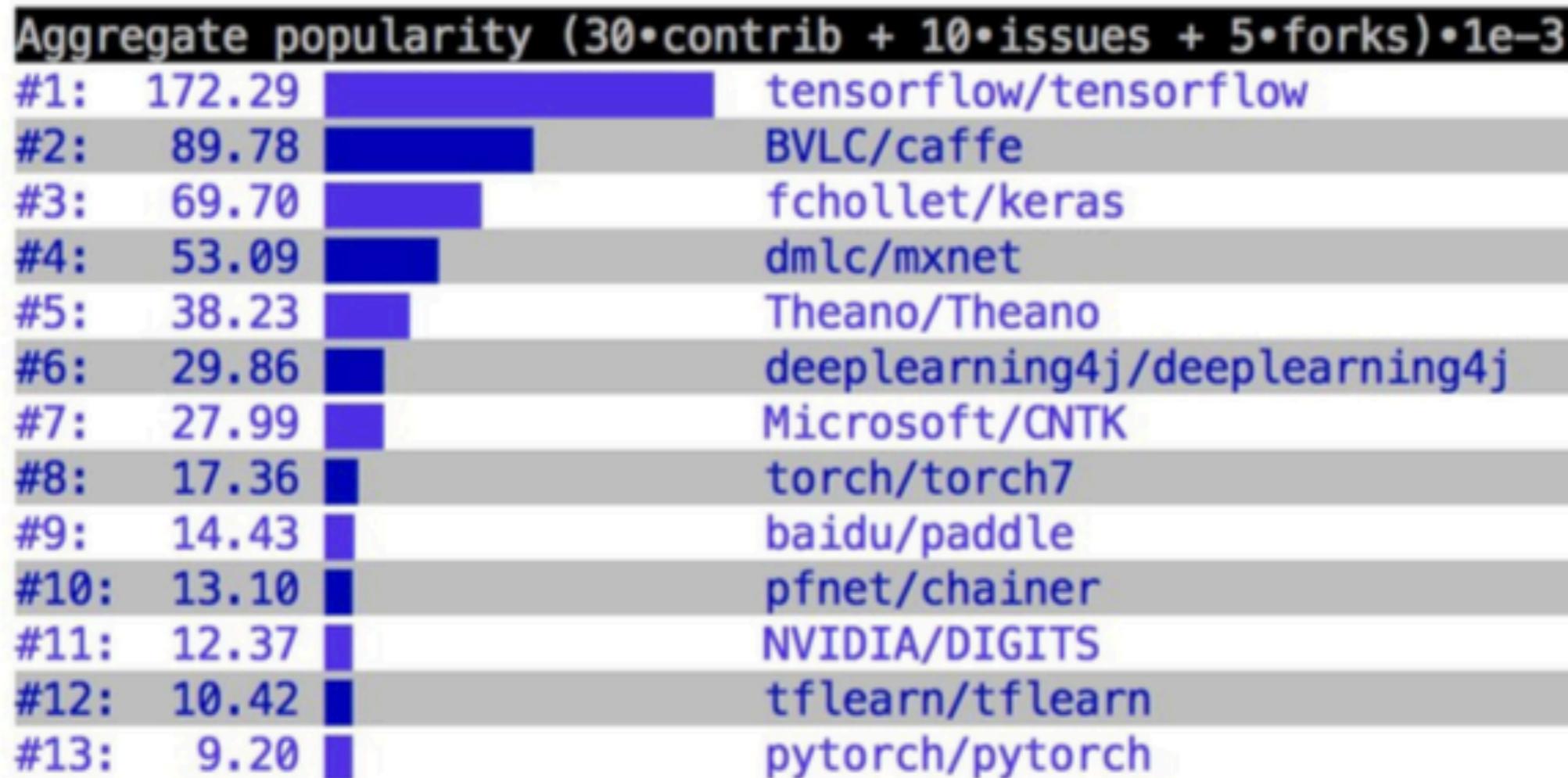
DEEP LEARNING

---

TENSORFLOW  
OVERVIEWS

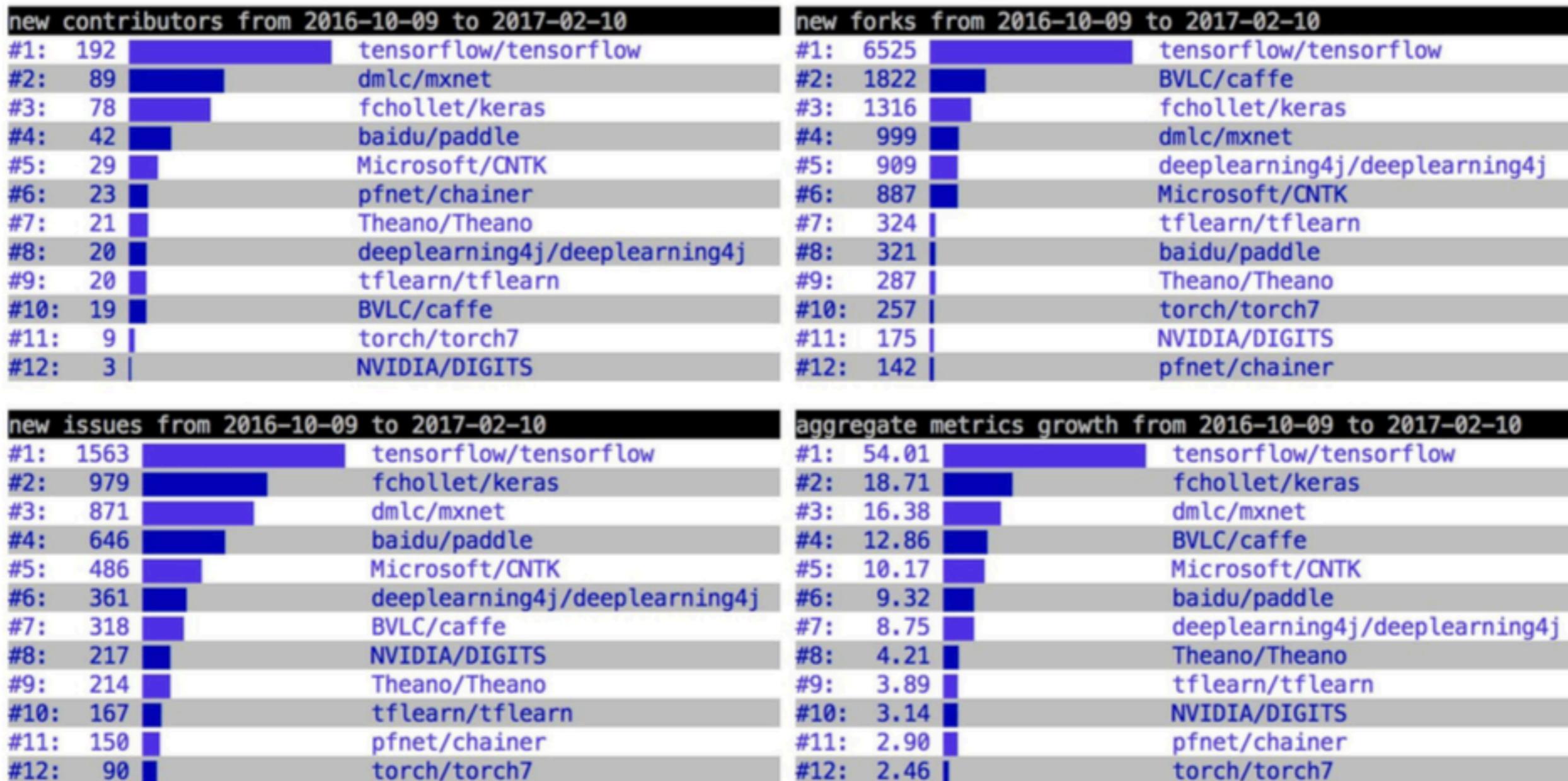
# 머신러닝 라이브러리 (1)

## Deep learning libraries: Accumulated GitHub metrics



# 머신러닝 라이브러리 (2)

Deep learning libraries: growth over past three months



François Chollet @fchollet - Feb 11

Time for an update: what does the deep learning library landscape look like, seen from GitHub? [pic.twitter.com/QDZyvLrYBd](https://twitter.com/QDZyvLrYBd)

<https://twitter.com/fchollet/status/830499993450450944/>

## 텐서플로우

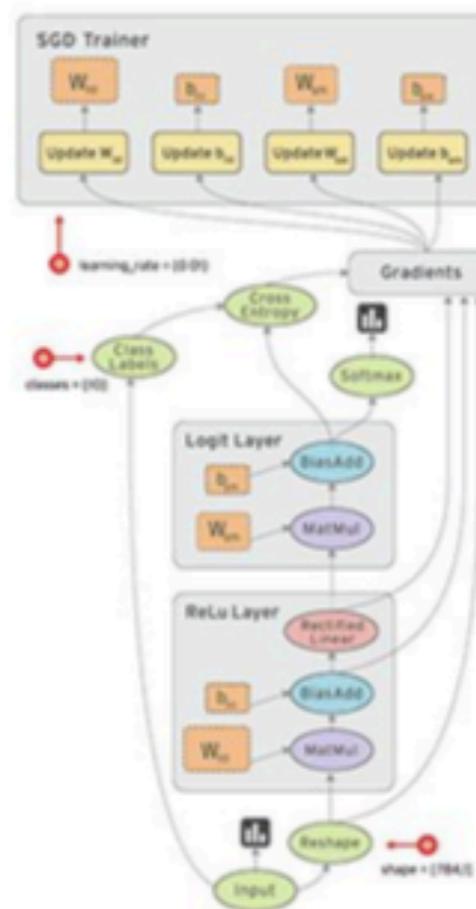
- TensorFlow™ is an open source software library for numerical computation using data flow graphs.
- Python!



## 텐서플로우 DATA FLOW GRAPH

# What is a Data Flow Graph?

- Nodes in the graph represent mathematical operations
- Edges represent the multidimensional data arrays (tensors) communicated between them.



## 텐서플로우 속성 : RANK

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Math entity	Python example
0	Scalar (magnitude only)	s = 483
1	Vector (magnitude and direction)	v = [1.1, 2.2, 3.3]
2	Matrix (table of numbers)	m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	3-Tensor (cube of numbers)	t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]
n	n-Tensor (you get the idea)	....

## 텐서플로우 속성 : SHAPE

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

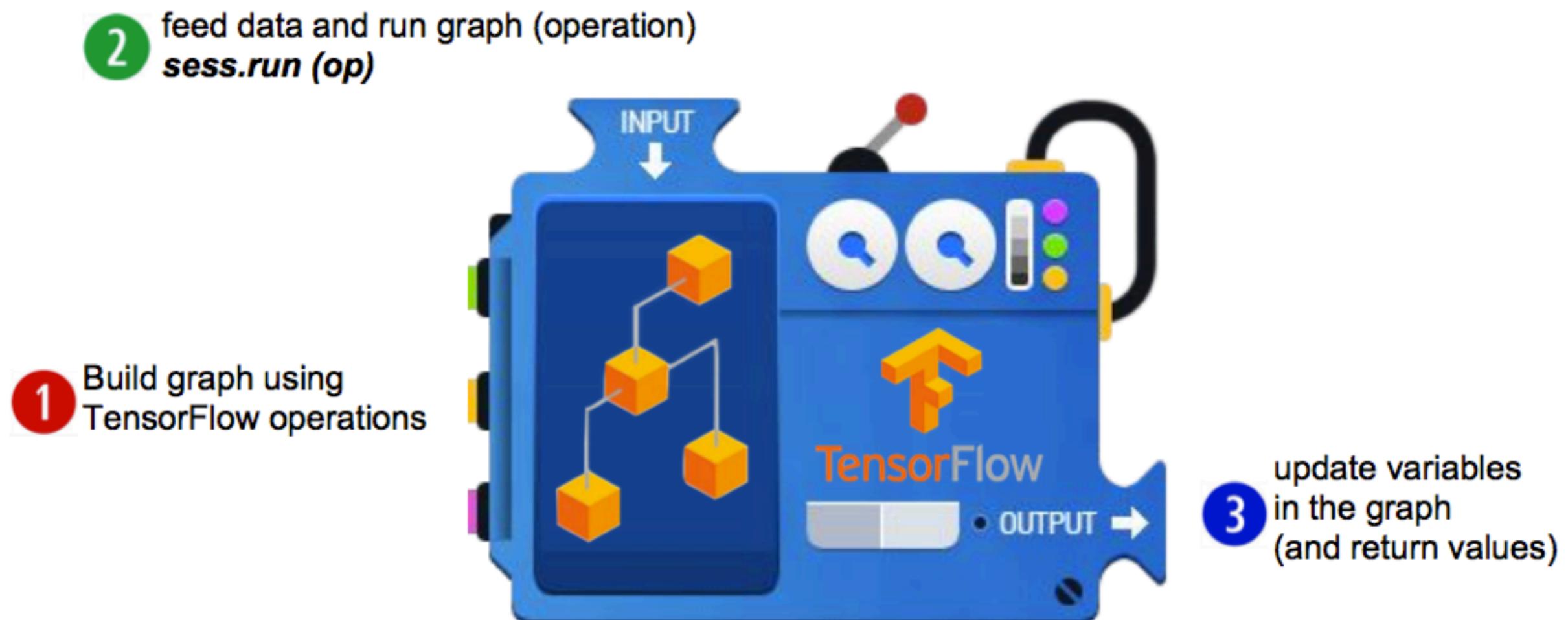
## 텐서플로우 속성 : DTYPES

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Data type	Python type	Description
DT_FLOAT	<code>tf.float32</code>	32 bits floating point.
DT_DOUBLE	<code>tf.float64</code>	64 bits floating point.
DT_INT8	<code>tf.int8</code>	8 bits signed integer.
DT_INT16	<code>tf.int16</code>	16 bits signed integer.
DT_INT32	<code>tf.int32</code>	32 bits signed integer.
DT_INT64	<code>tf.int64</code>	64 bits signed integer.

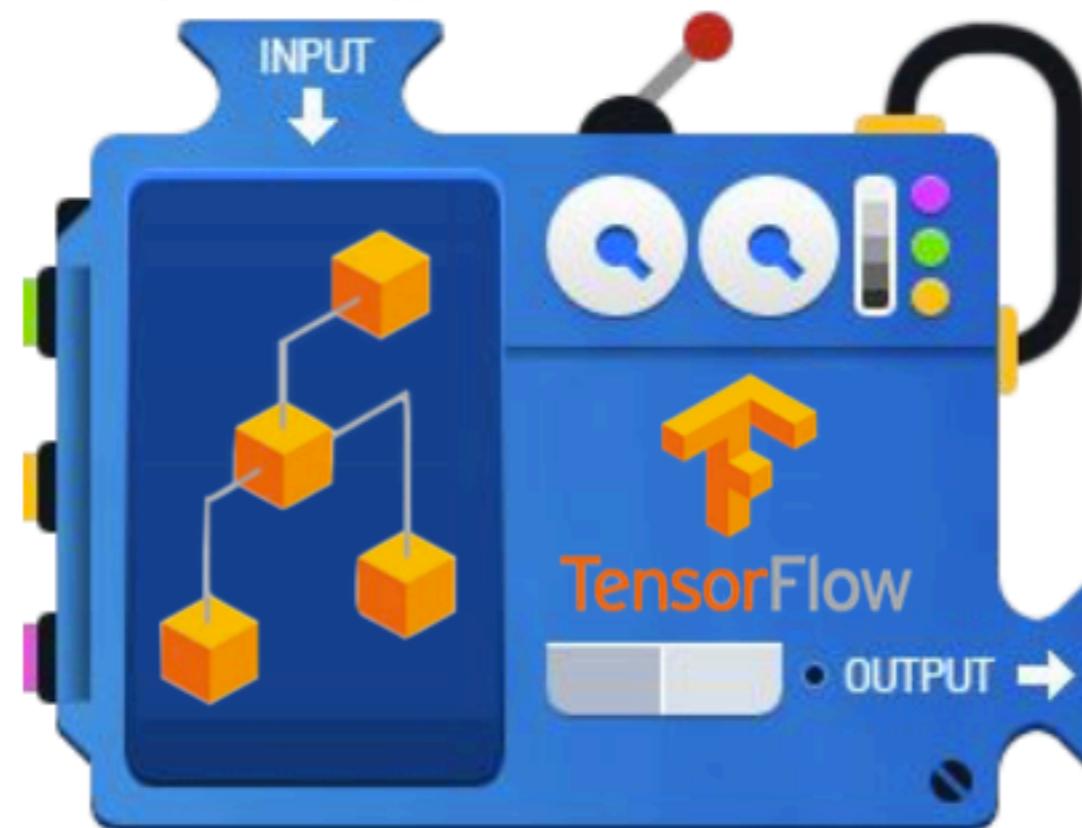
...

# TENSOR FLOW MECHANICS (OPERATION)



# TENSOR FLOW MECHANICS (PLACEHOLDER)

- 1 Build graph using TensorFlow operations
- 2 feed data and run graph (operation)  
`sess.run (op, feed_dict={x: x_data})`



- 3 update variables in the graph (and return values)

## TENSOR FLOW MECHANICS (1)

1

### Build graph using TF operations

$$H(x) = Wx + b$$

```
# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
# Our hypothesis XW+b
hypothesis = x_train * W + b
```

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

```
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))
```

# TENSOR FLOW MECHANICS (1)

1

## Build graph using TF operations

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

```
t = [1., 2., 3., 4.]  
tf.reduce_mean(t) ==> 2.5
```

```
# cost/loss function  
cost = tf.reduce_mean(tf.square(hypothesis - y_train))
```

## GradientDescent

```
# Minimize  
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)  
train = optimizer.minimize(cost)
```

## TENSOR FLOW MECHANICS (2, 3)

- 2
- 3

Run/update graph and get results

```
# Launch the graph in a session.  
sess = tf.Session()  
# Initializes global variables in the graph.  
sess.run(tf.global_variables_initializer())  
  
# Fit the line  
for step in range(2001):  
    sess.run(train)  
    if step % 20 == 0:  
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

# LINEAR REGRESSION SOURCE

```
import tensorflow as tf

# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Our hypothesis XW+b
hypothesis = x_train * W + b

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the Line
for step in range(2001):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

## Full code (less than 20 lines)

```
...
0 2.82329 [ 2.12867713] [-0.85235667]
20 0.190351 [ 1.53392804] [-1.05059612]
40 0.151357 [ 1.45725465] [-1.02391243]
...
1920 1.77484e-05 [ 1.00489295] [-0.01112291]
1940 1.61197e-05 [ 1.00466311] [-0.01060018]
1960 1.46397e-05 [ 1.004444] [-0.01010205]
1980 1.32962e-05 [ 1.00423515] [-0.00962736]
2000 1.20761e-05 [ 1.00403607] [-0.00917497]
...  
...
```



DEEP LEARNING

---

MULTI-VARIABLE  
LINEAR REGRESSION

## MULTI-VARIABLE LINEAR REGRESSION

---

### MULTI FEATURES

one-variable  
one-feature

x (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

multi-variable/feature

x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

# HYPOTHESIS AND COST FUNCTION

**Hypothesis**

$$H(x) = Wx + b$$

$$H(x_1, x_2) = w_1x_1 + w_2x_2 + b$$

**Cost function**

$$H(x_1, x_2) = w_1x_1 + w_2x_2 + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

# MATRIX AND TRANSPOSE

## Matrix multiplication

"Dot Product"

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 \end{bmatrix}$$

## Transpose

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

## MULTI-FEATURES

**Predicting exam score:  
regression using three inputs ( $x_1, x_2, x_3$ )**

multi-variable/feature

<b><math>x_1</math> (quiz 1)</b>	<b><math>x_2</math> (quiz 2)</b>	<b><math>x_3</math> (midterm 1)</b>	<b>Y (final)</b>
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

### HYPOTHESIS USING MATRIX (1)

$$w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$$

$$(x_1 \quad x_2 \quad x_3) \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = (x_1w_1 + x_2w_2 + x_3w_3)$$

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (2)

$$H(x_1, x_2, x_3) = x_1w_1 + x_2w_2 + x_3w_3$$

<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>x<sub>3</sub></b>	<b>Y</b>
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

$$(x_1 \quad x_2 \quad x_3) \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = (x_1w_1 + x_2w_2 + x_3w_3)$$

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (3)

$x_1$	$x_2$	$x_3$	$Y$
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

**Hypothesis using matrix**

$$w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (4)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[5, 3]

[3, 1]

[5, 1]

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (5)

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} * \begin{pmatrix} \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{H(x)} \end{pmatrix}$$

[5, 3]

[?, ?]

[5, 1]

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (6)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[n, 3]

[3, 1]

[n, 1]

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (7)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot ? = \begin{pmatrix} x_{11}w_{11} + x_{12}w_{21} + x_{13}w_{31} & x_{11}w_{12} + x_{12}w_{22} + x_{13}w_{32} \\ x_{21}w_{11} + x_{22}w_{21} + x_{23}w_{31} & x_{21}w_{12} + x_{22}w_{22} + x_{23}w_{32} \\ x_{31}w_{11} + x_{32}w_{21} + x_{33}w_{31} & x_{31}w_{12} + x_{32}w_{22} + x_{33}w_{32} \\ x_{41}w_{11} + x_{42}w_{21} + x_{43}w_{31} & x_{41}w_{12} + x_{42}w_{22} + x_{43}w_{32} \\ x_{51}w_{11} + x_{52}w_{21} + x_{53}w_{31} & x_{51}w_{12} + x_{52}w_{22} + x_{53}w_{32} \end{pmatrix}$$

?

[n, 3]
[?, ?]
[n, 2]

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (8)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} = \begin{pmatrix} x_{11}w_{11} + x_{12}w_{21} + x_{13}w_{31} & x_{11}w_{12} + x_{12}w_{22} + x_{13}w_{32} \\ x_{21}w_{11} + x_{22}w_{21} + x_{23}w_{31} & x_{21}w_{12} + x_{22}w_{22} + x_{23}w_{32} \\ x_{31}w_{11} + x_{32}w_{21} + x_{33}w_{31} & x_{31}w_{12} + x_{32}w_{22} + x_{33}w_{32} \\ x_{41}w_{11} + x_{42}w_{21} + x_{43}w_{31} & x_{41}w_{12} + x_{42}w_{22} + x_{43}w_{32} \\ x_{51}w_{11} + x_{52}w_{21} + x_{53}w_{31} & x_{51}w_{12} + x_{52}w_{22} + x_{53}w_{32} \end{pmatrix}$$

**[n, 3]**
**[3, 2]**
**[n, 2]**

$$H(X) = XW$$

## HYPOTHESIS USING MATRIX (9)

- Lecture (theory):

$$H(x) = Wx + b$$

- Implementation (TensorFlow)

$$H(X) = XW$$



DEEP LEARNING

---

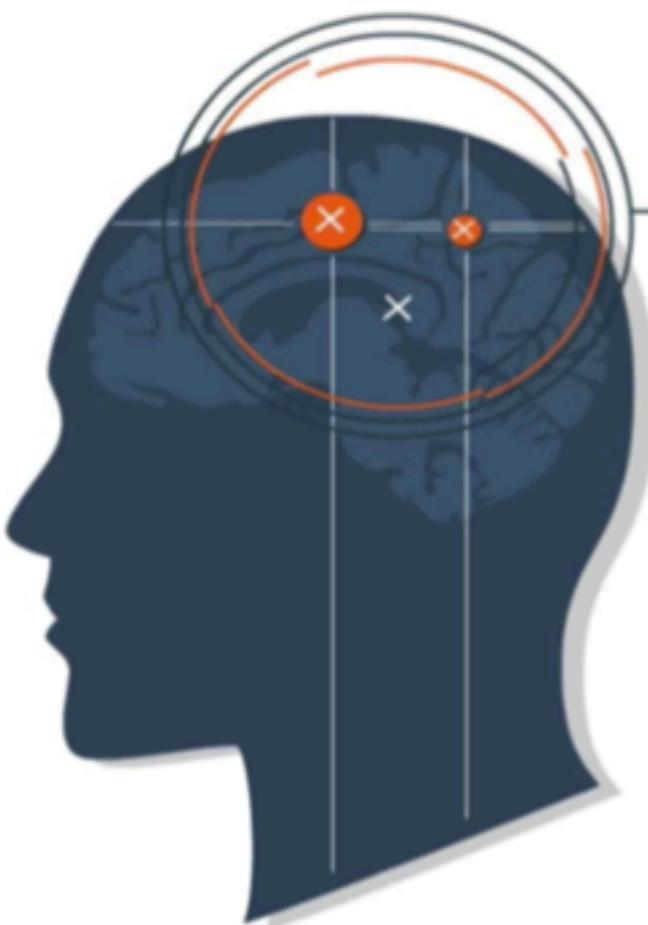
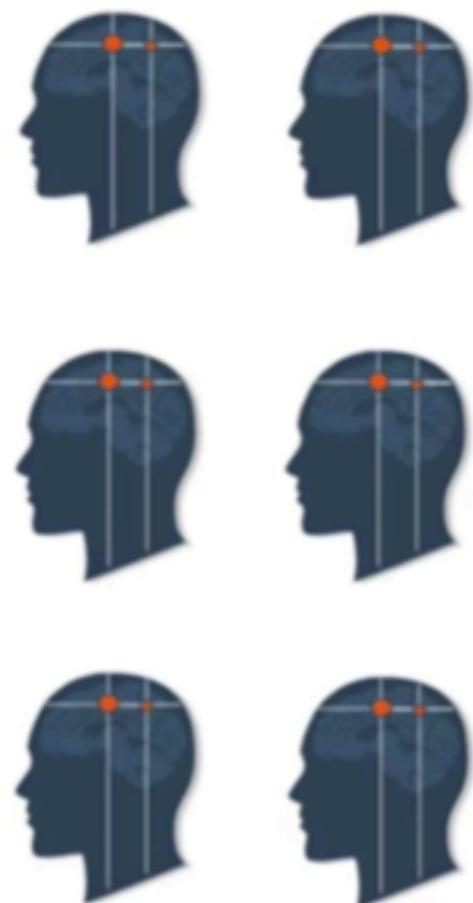
LOGISTIC  
CLASSIFICATION

# CLASSIFICATION AND ENCODING

- Spam Detection: Spam or Ham
  - Facebook feed: show or hide
  - Credit Card Fraudulent Transaction detection: legitimate/fraud
- 
- Spam Detection: Spam (1) or Ham (0)
  - Facebook feed: show(1) or hide(0)
  - Credit Card Fraudulent Transaction detection: legitimate(0) or fraud (1)

# RADIOLOGY

## Radiology



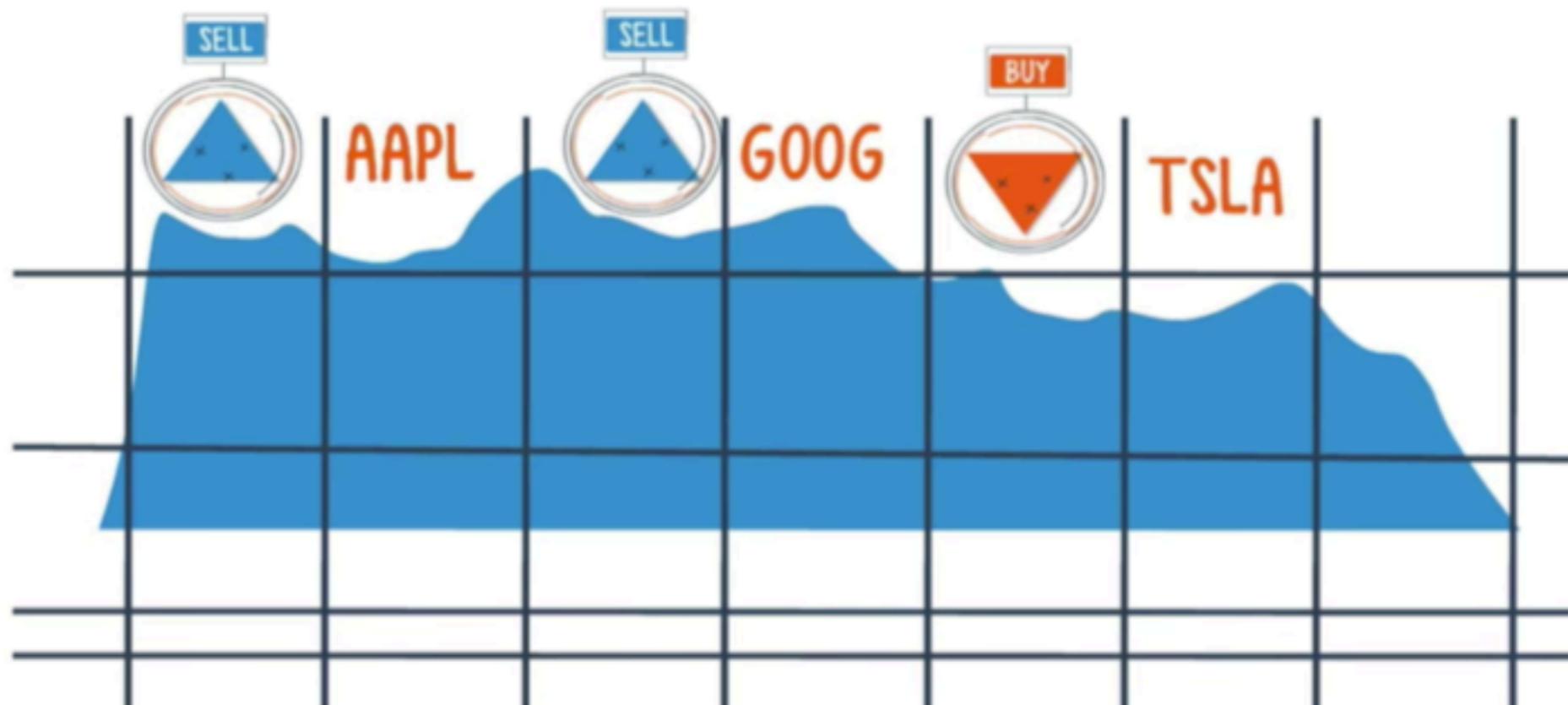
Malignant  
tumor

Benign  
tumor

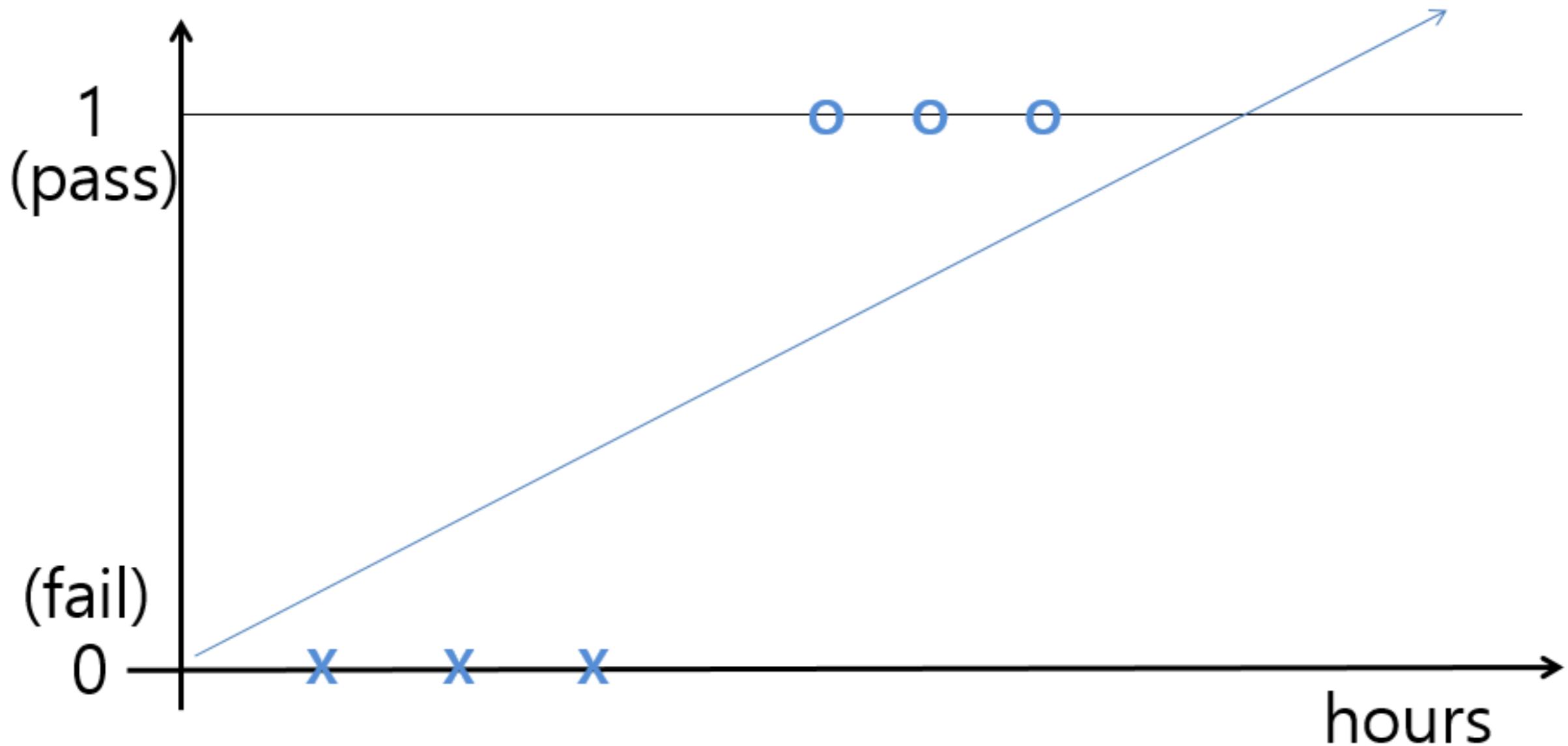
# FINANCE

DWJI	17,499.10	▼
SP500	2,025.51	▼
NASDAQ	4,976.9	▲
AAPL	107.71	▲
GOOG	750.06	▲
TSLA	234.24	▼

# Finance



# LINEAR REGRESSION?

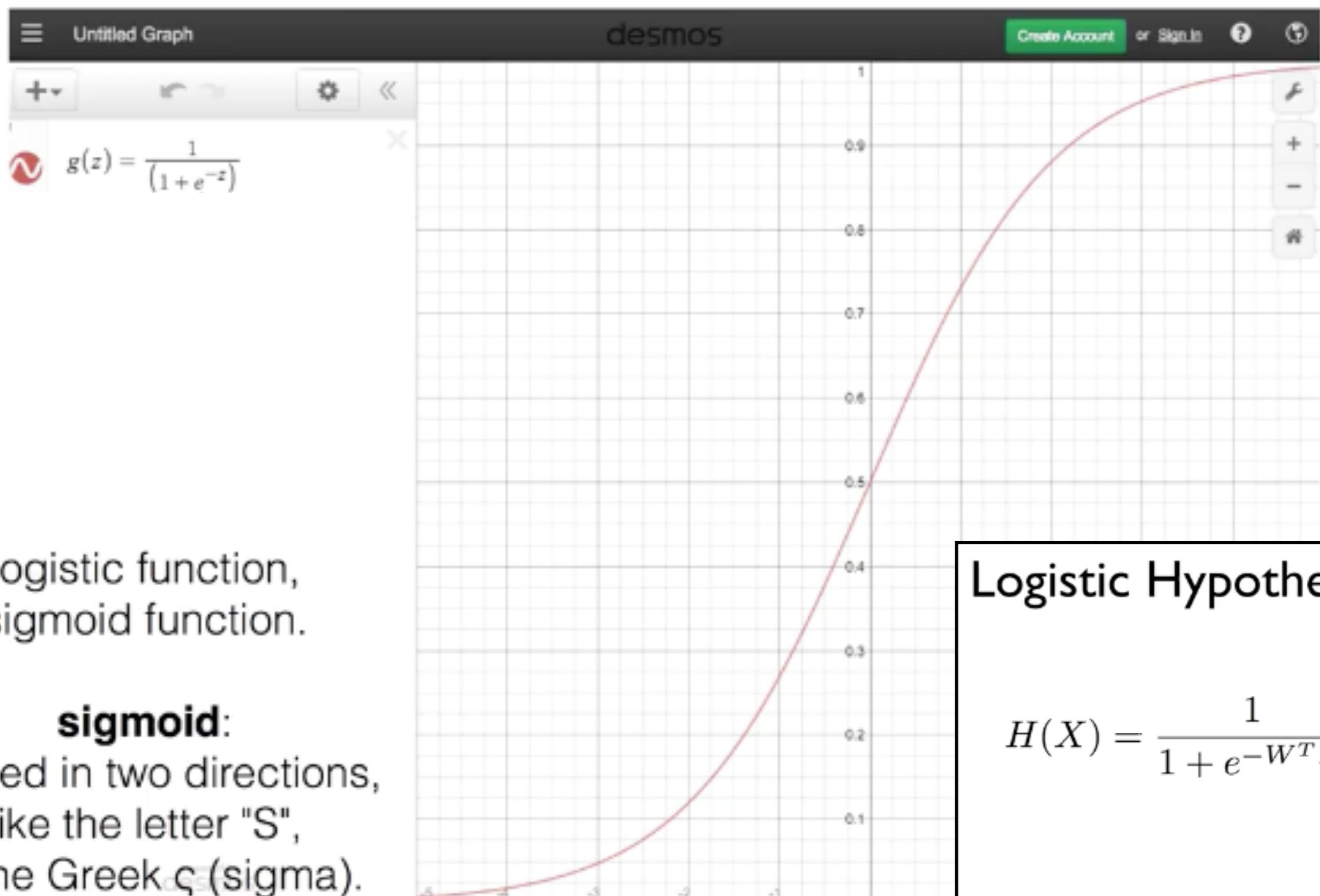


### PROBLEMS

#### Linear regression

- We know  $Y$  is 0 or 1  
$$H(x) = Wx + b$$
- Hypothesis can give values large than 1 or less than 0

# SIGMOID



## COST FUNCTION

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx + b$$

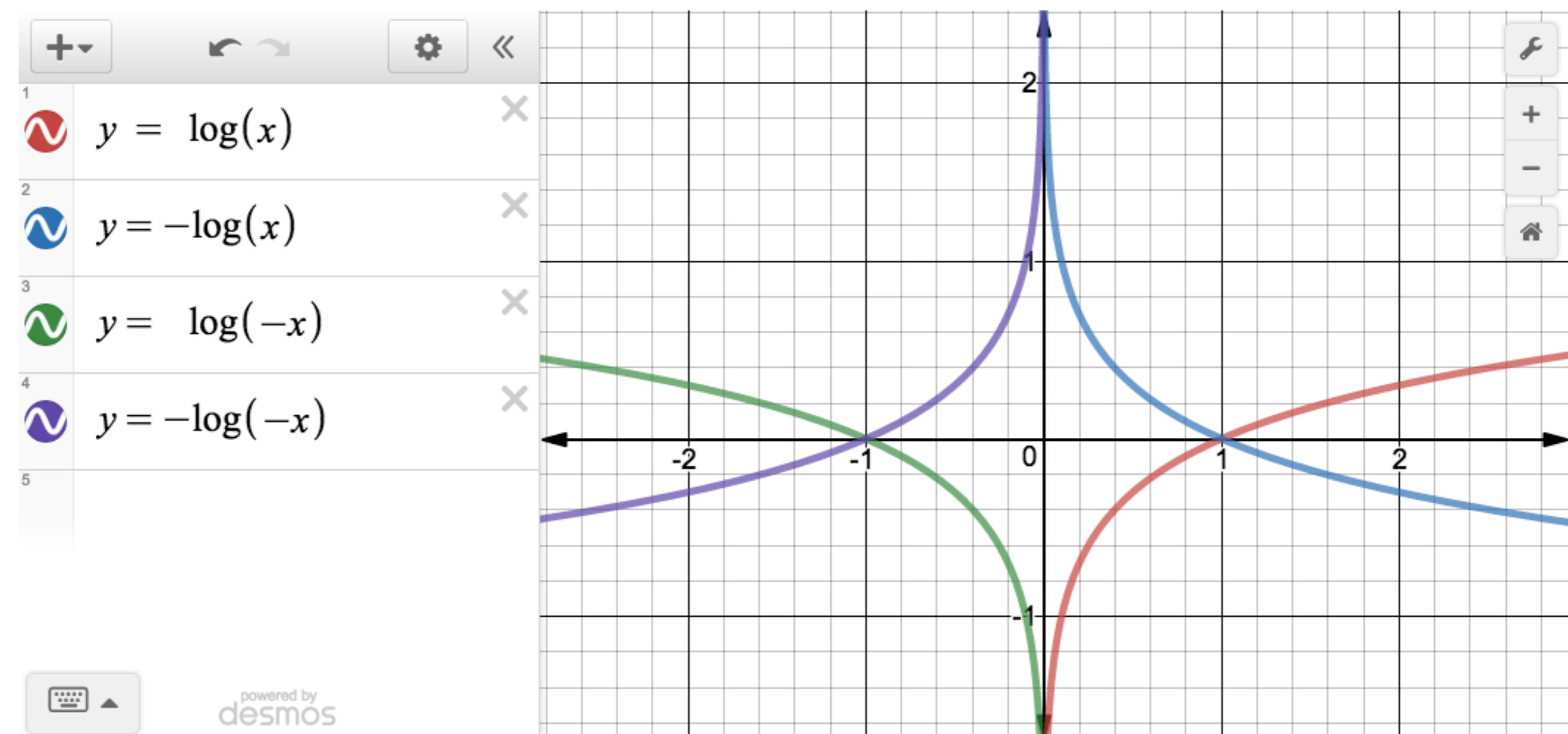
$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

## NEW COST FUNCTION FOR LOGISTIC

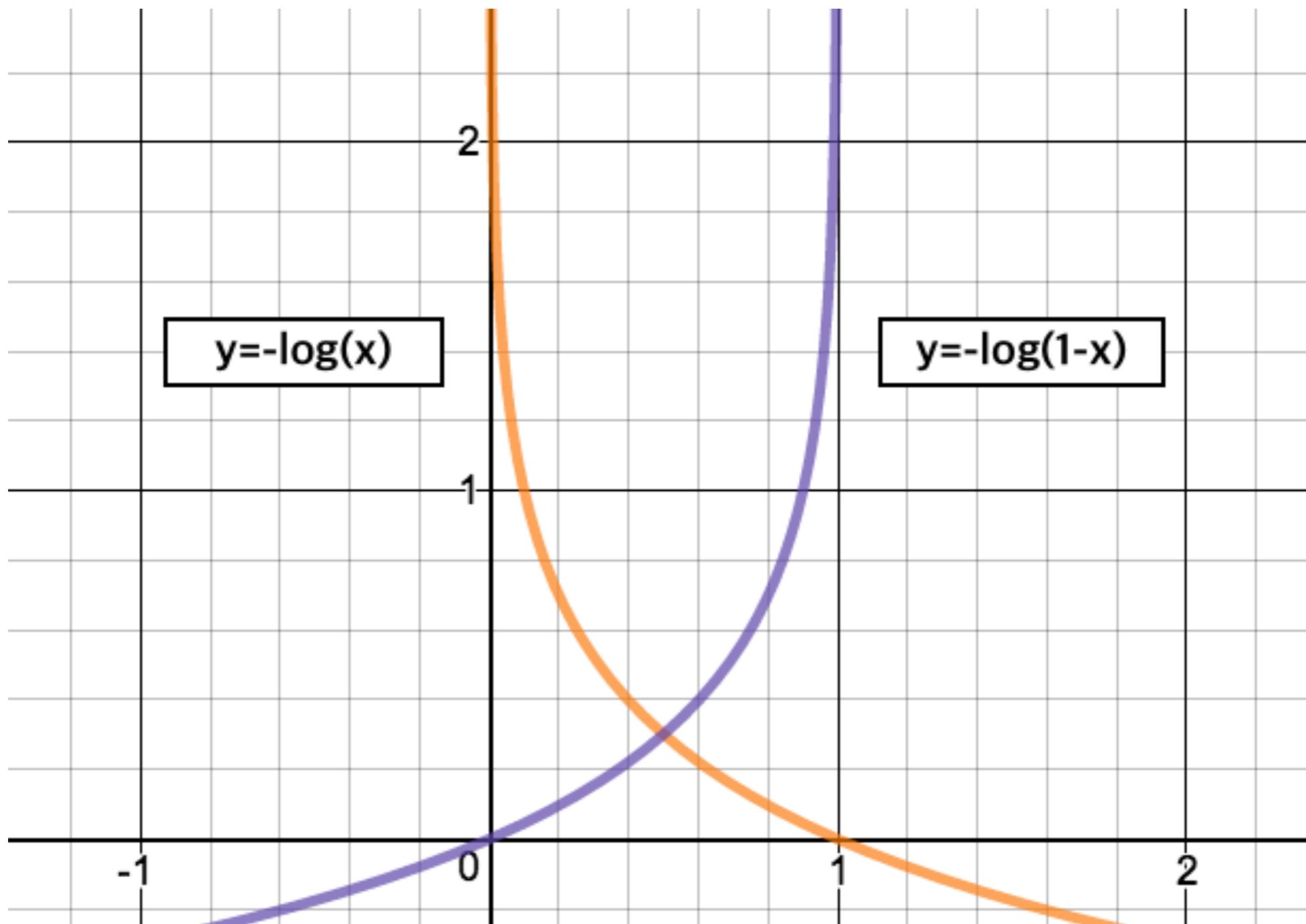
$$\text{cost}(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

## LOG FUNCTION



# LOG FUNCTIONS WE NEED TO KNOW



# UNDERSTANDING COST FUNCTION

$$C(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1-H(x)) & : y = 0 \end{cases}$$

$y = 1$

$H(x) = 1, \text{cost}(1) = 0$

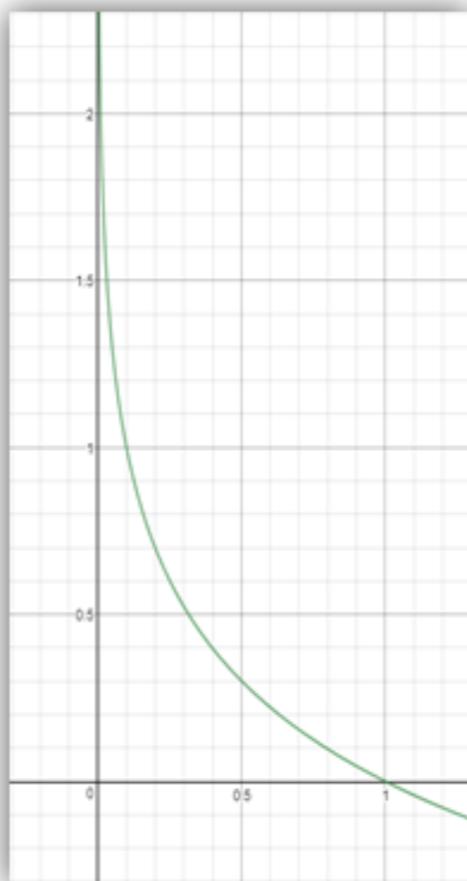
$H(x) = 0, \text{cost}(0) = \infty$

$y = 0$

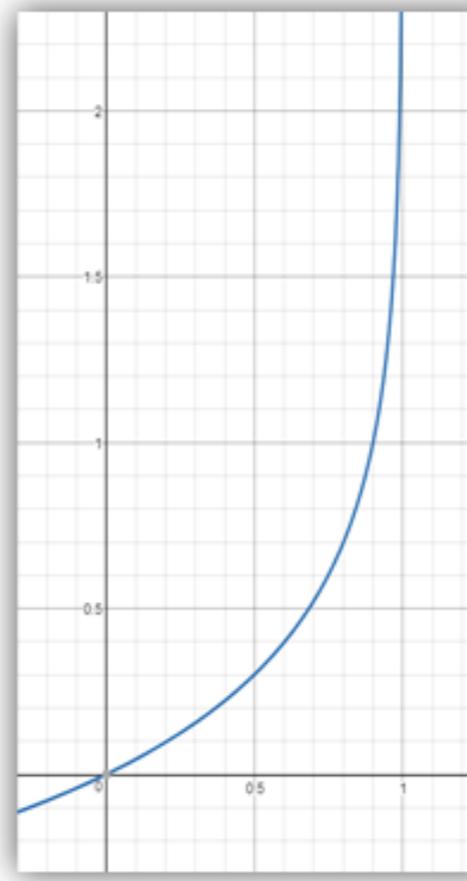
$H(x) = 0, \text{cost}(0) = 0$

$H(x) = 1, \text{cost}(1) = \infty$

$g(z) = -\log(z)$



$g(z) = -\log(1-z)$



## COST FUNCTION

$$\text{cost}(W) = \frac{1}{m} \sum C(H(x), y)$$

$$C(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

$$C(H(x), y) = -y \log(H(x)) - (1 - y) \log(1 - H(x))$$

## GRADIENT DESCENT ALGORITHM

$$cost(W) = -\frac{1}{m} \sum y \log(H(x)) + (1 - y) \log(1 - H(x))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

```
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))

# Minimize
a = tf.Variable(0.1) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```

# CLASSIFYING DIABETES

## Classifying diabetes

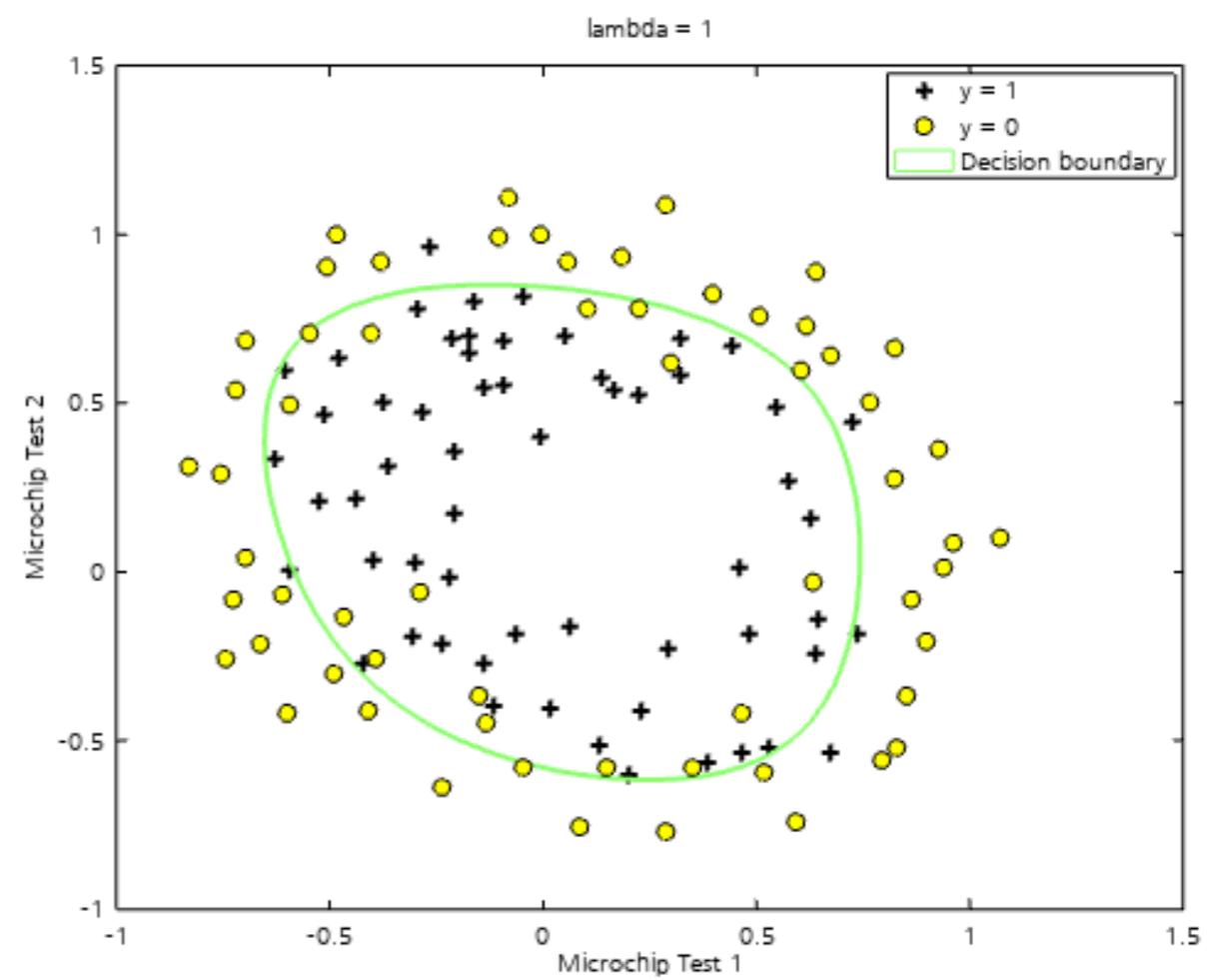
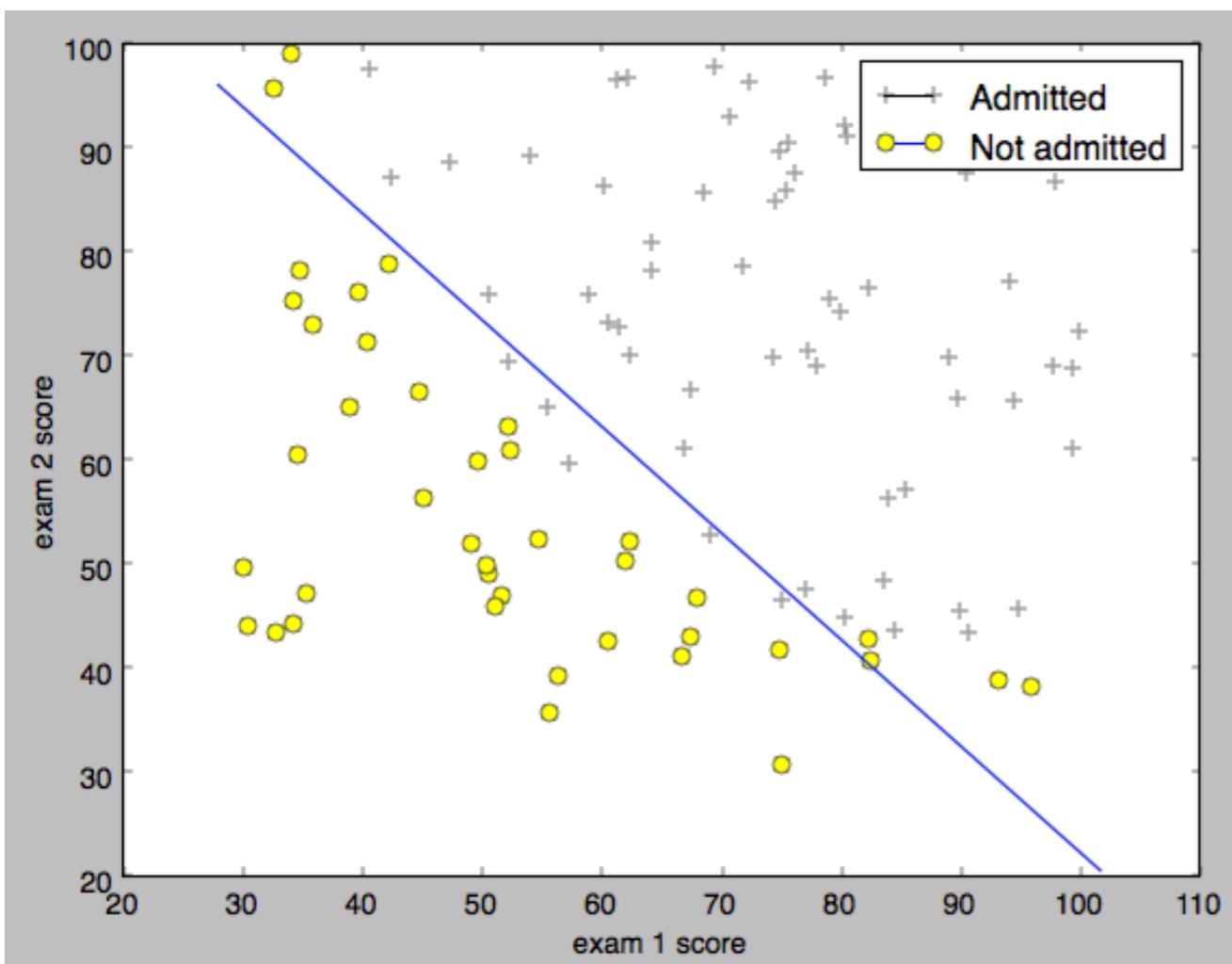


-0.411765	0.165829	0.213115	0	0	-0.23696	-0.894962	-0.7	1
-0.647059	-0.21608	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333	0
0.176471	0.155779	0	0	0	0.052161	-0.952178	-0.733333	1
-0.764706	0.979899	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667	0
-0.0588235	0.256281	0.57377	0	0	0	-0.868488	0.1	0
-0.529412	0.105528	0.508197	0	0	0.120715	-0.903501	-0.7	1
0.176471	0.688442	0.213115	0	0	0.132638	-0.608027	-0.566667	0
0.176471	0.396985	0.311475	0	0	-0.19225	0.163962	0.2	1

# LOGISTIC CLASSIFICATION

---

## DECISION BOUNDARY





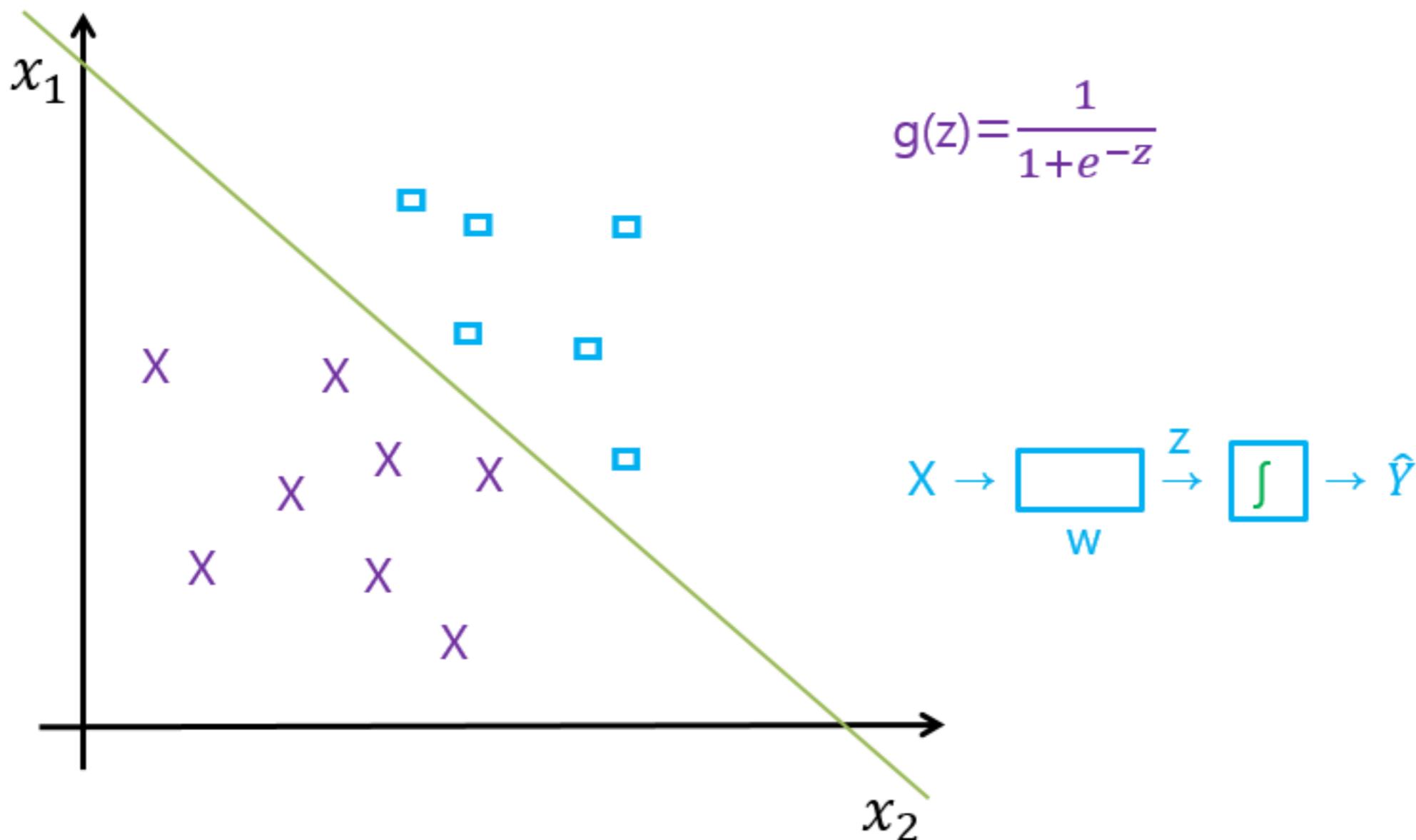
DEEP LEARNING

---

SOFTMAX

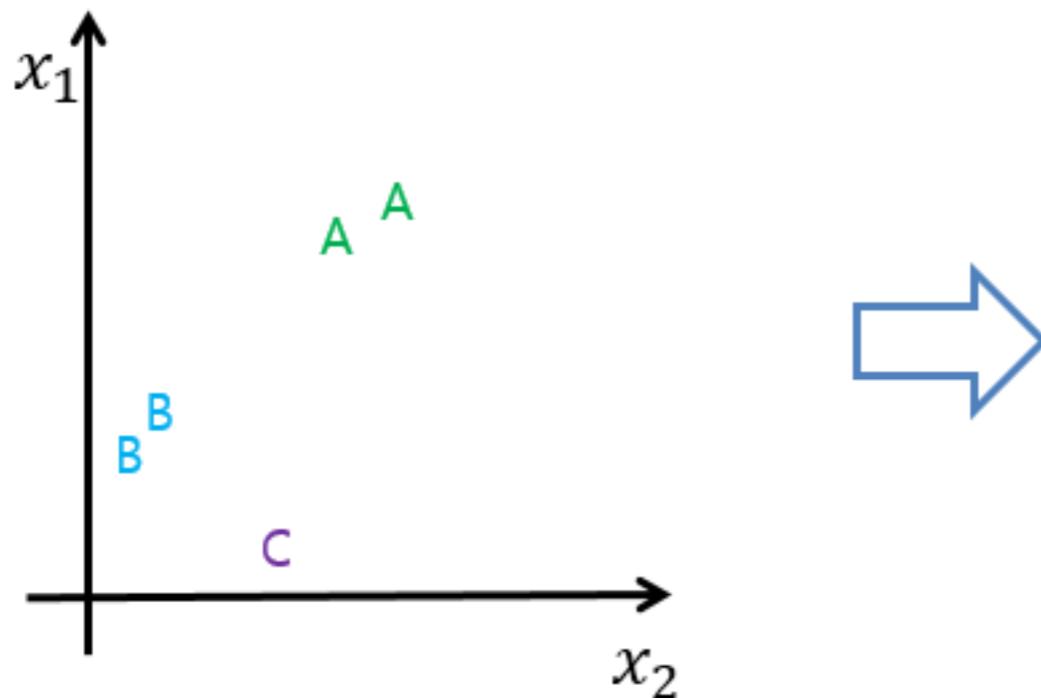
CROSS-ENTROPY

# LOGISTIC REGRESSION

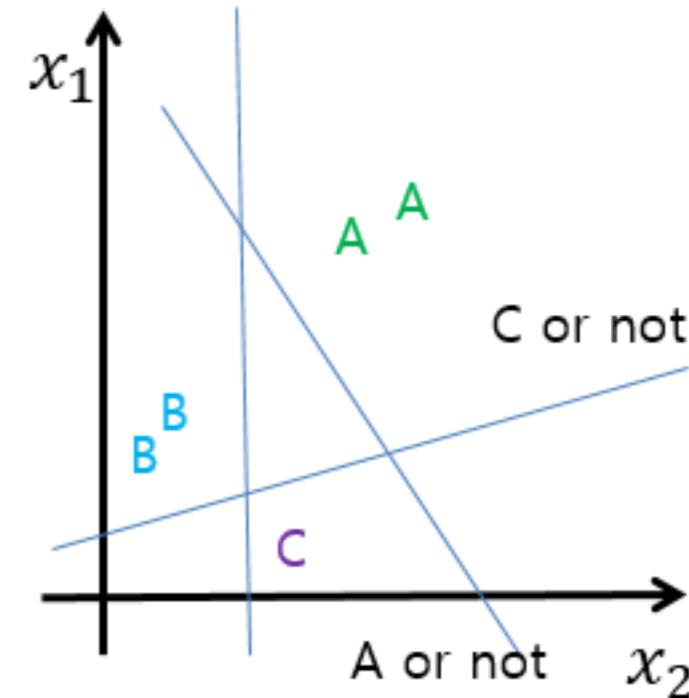


# MULTINOMIAL CLASSIFICATION

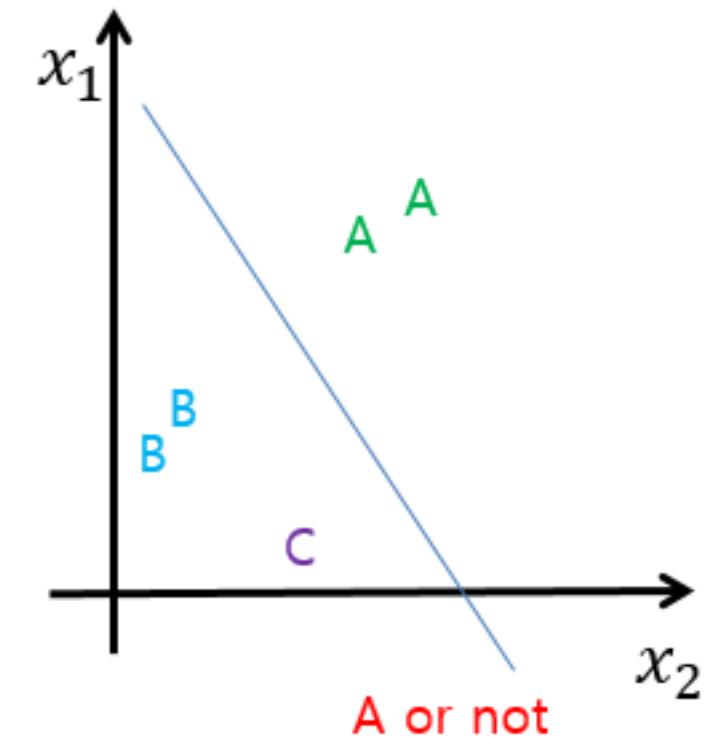
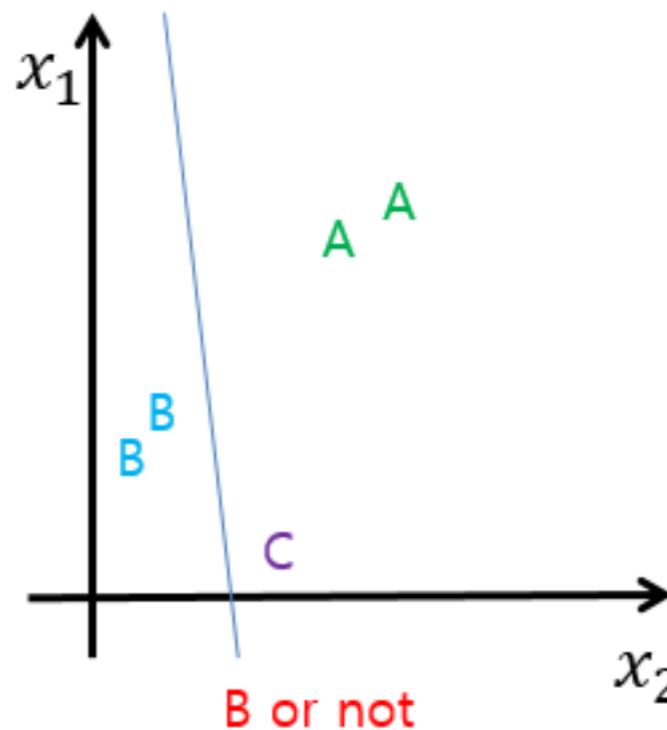
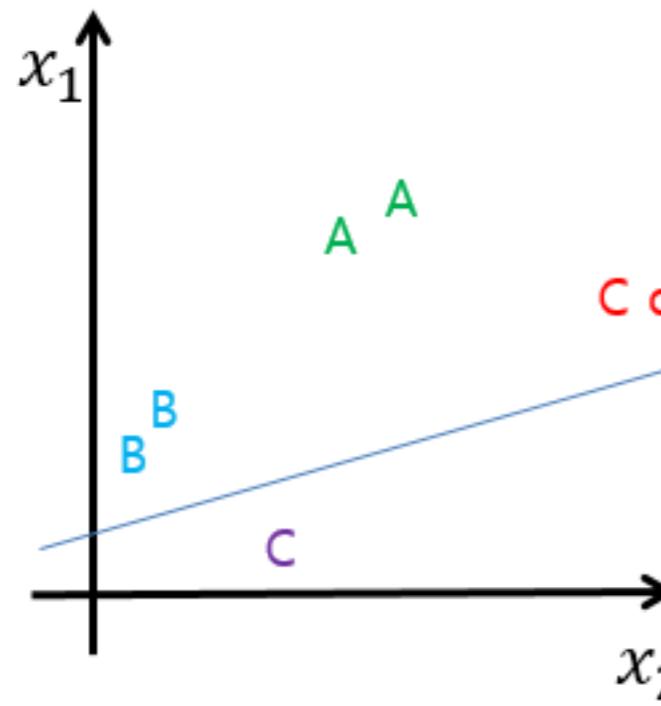
$x_1$ (hours)	$x_2$ (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C



B or not



# MULTINOMIAL CLASSIFICATION



$$X \rightarrow \boxed{\quad} \rightarrow \hat{Y}$$

A

$$X \rightarrow \boxed{\quad} \rightarrow \hat{Y}$$

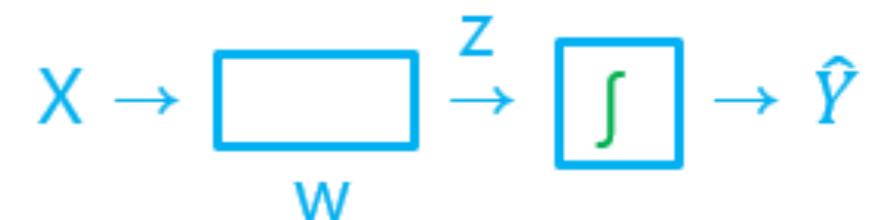
B

$$X \rightarrow \boxed{\quad} \rightarrow \hat{Y}$$

C

## MULTINOMIAL CLASSIFICATION

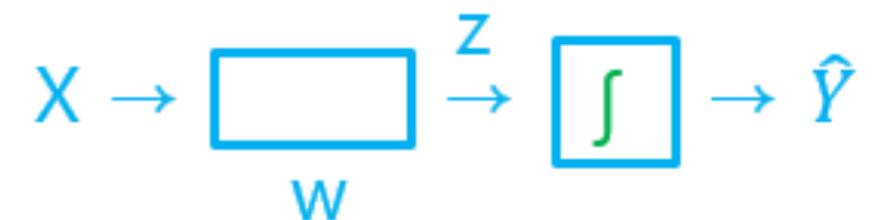
$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$



$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

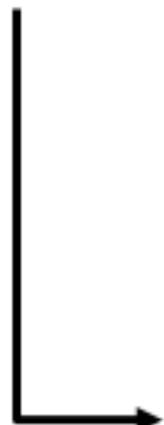


$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

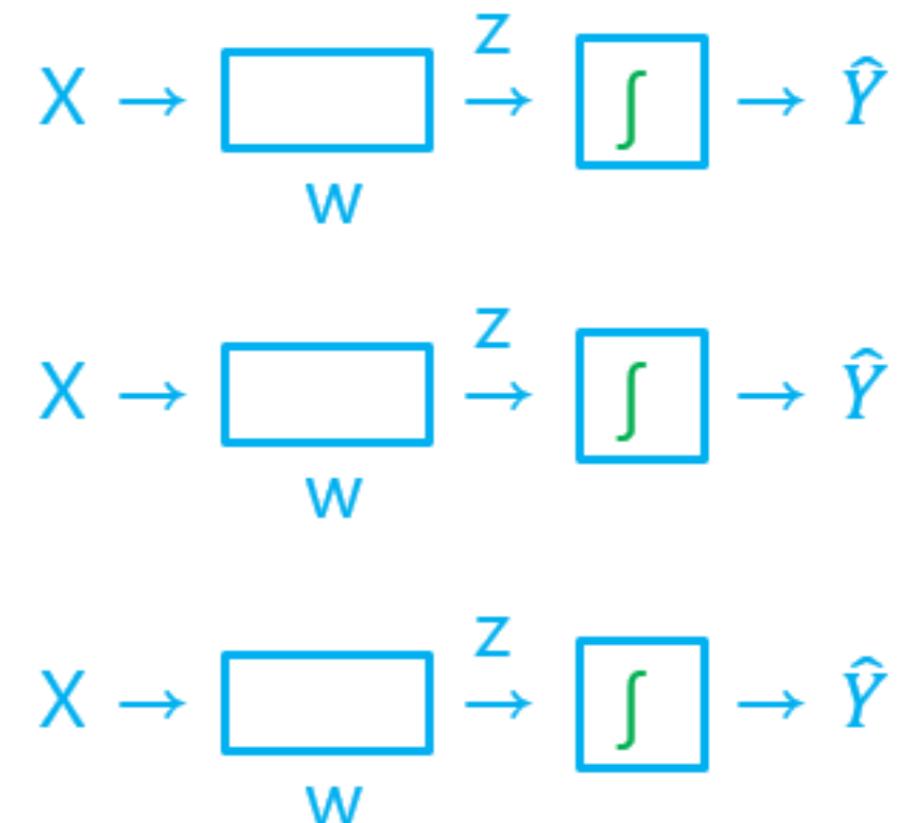


# MULTINOMIAL CLASSIFICATION

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

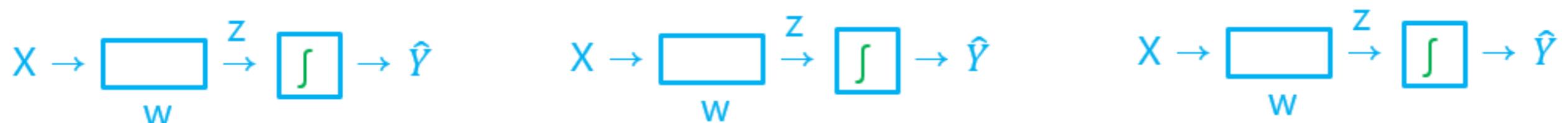


$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = ?$$



# MULTINOMIAL CLASSIFICATION

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix}$$



## WHERE IS SIGMOID?

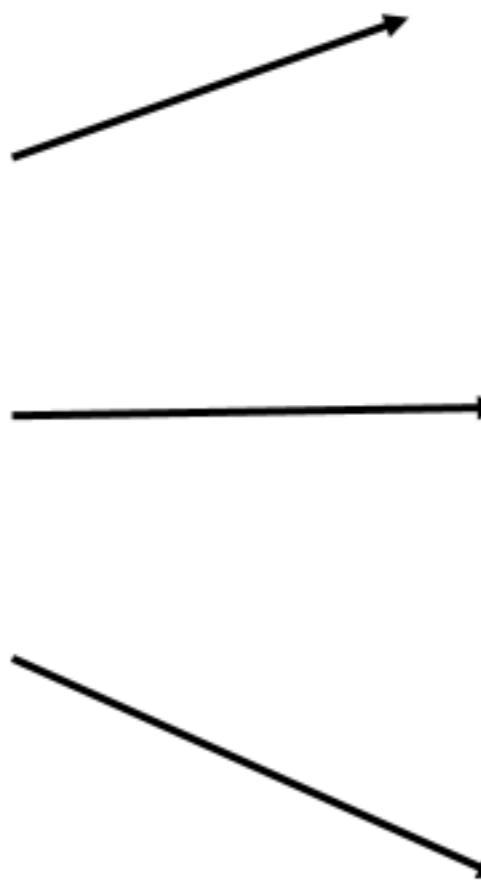
$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix} \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$



## SIGMOID?

### LOGISTIC CLASSIFIER

$$WX = Y \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$



$p = 0.7$



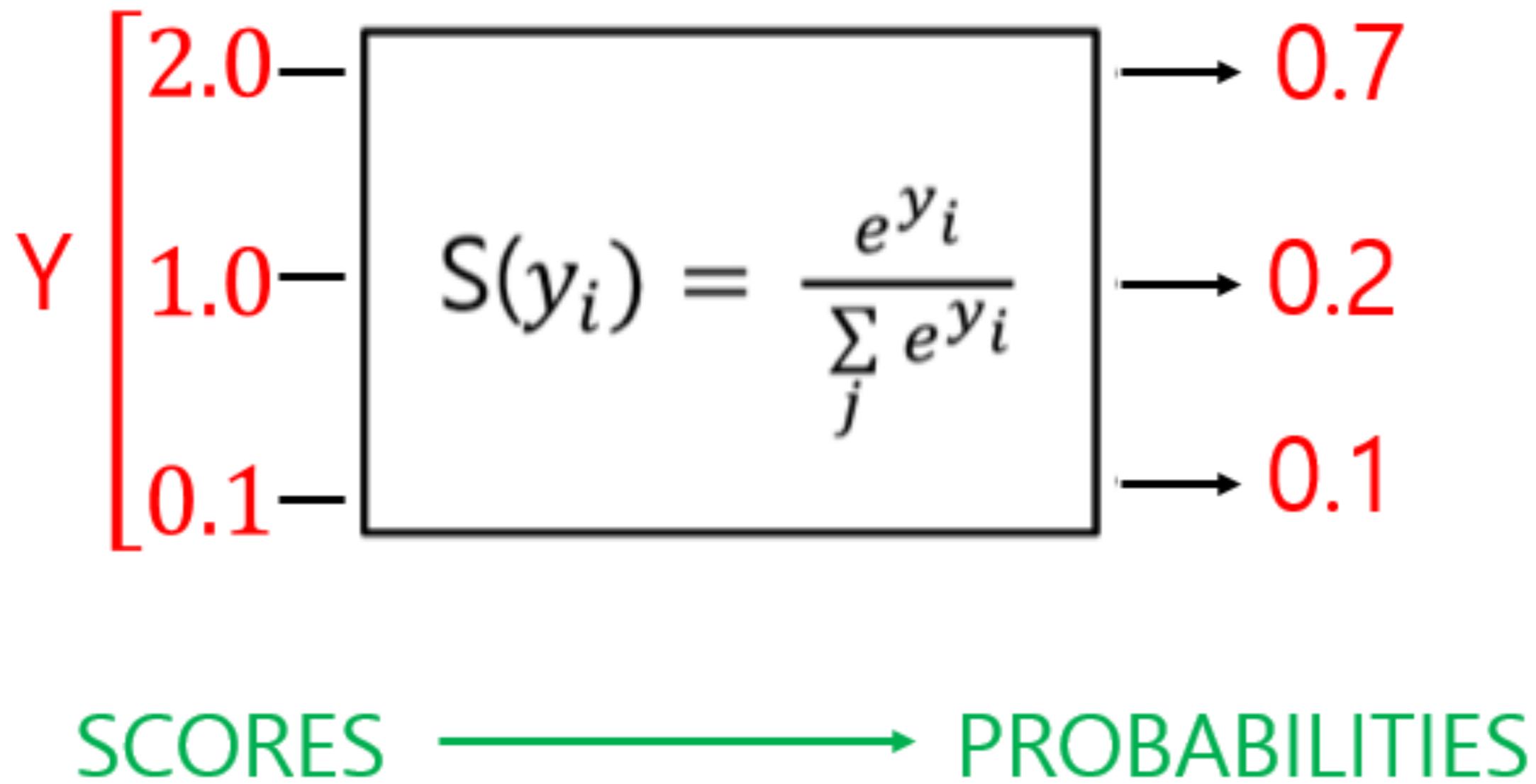
$p = 0.2$



$p = 0.1$

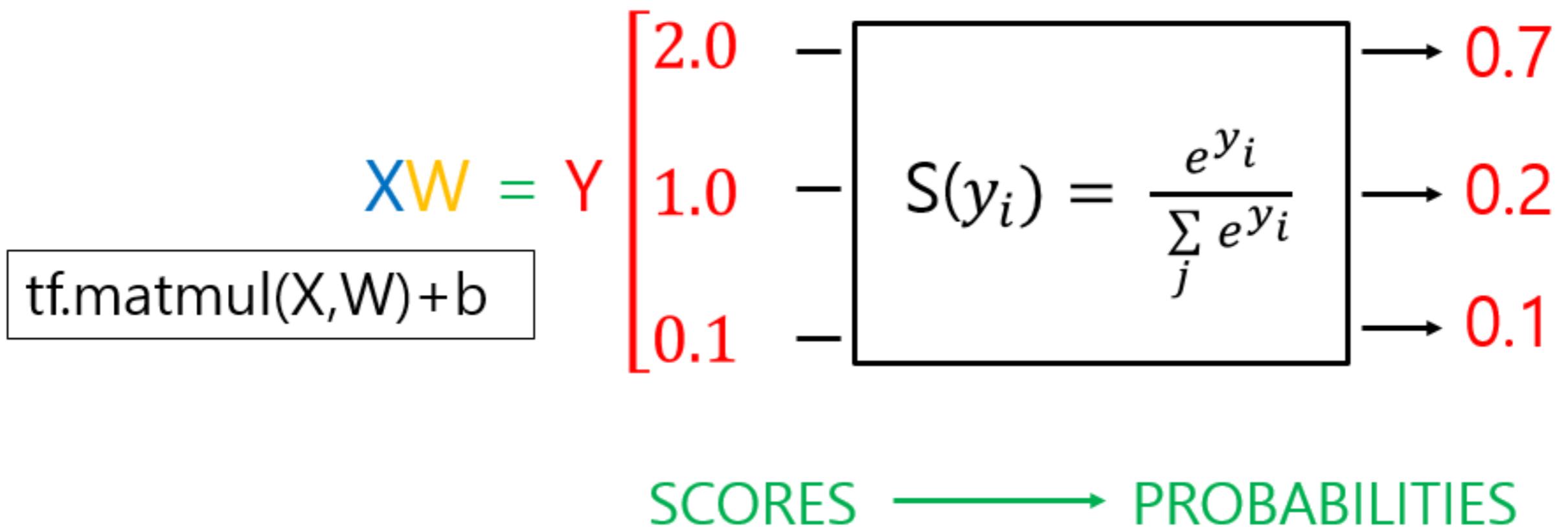


# SOFTMAX



# SOFTMAX

```
hypothesis = tf.nn.softmax(tf.matmul(X,W)+b)
```



# SOFTMAX\_CROSS\_ENTROPY\_WITH\_LOGITS

```
logits = tf.matmul(X, W) + b  
hypothesis = tf.nn.softmax(logits)
```

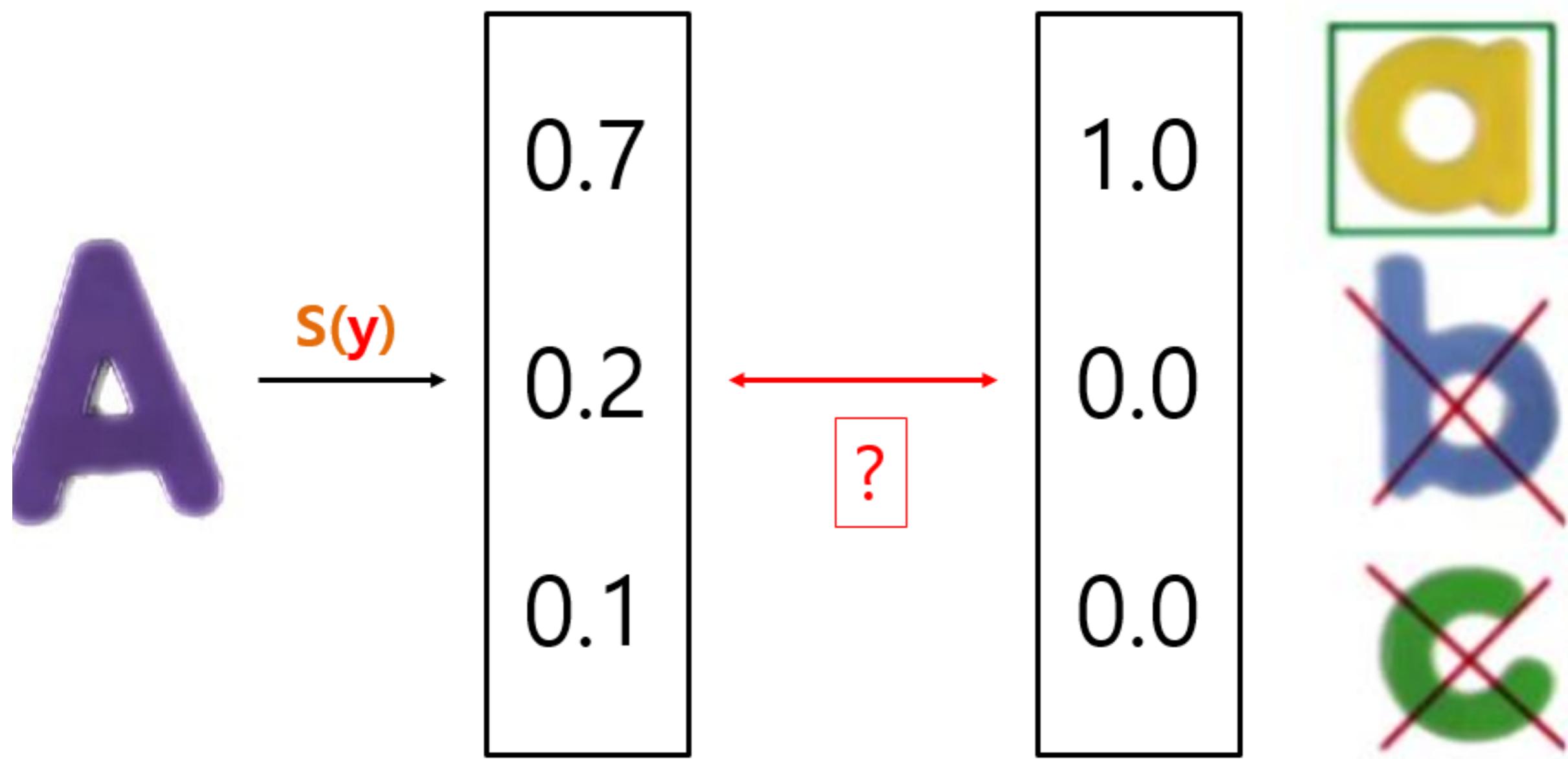
1

```
# Cross entropy cost/Loss  
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

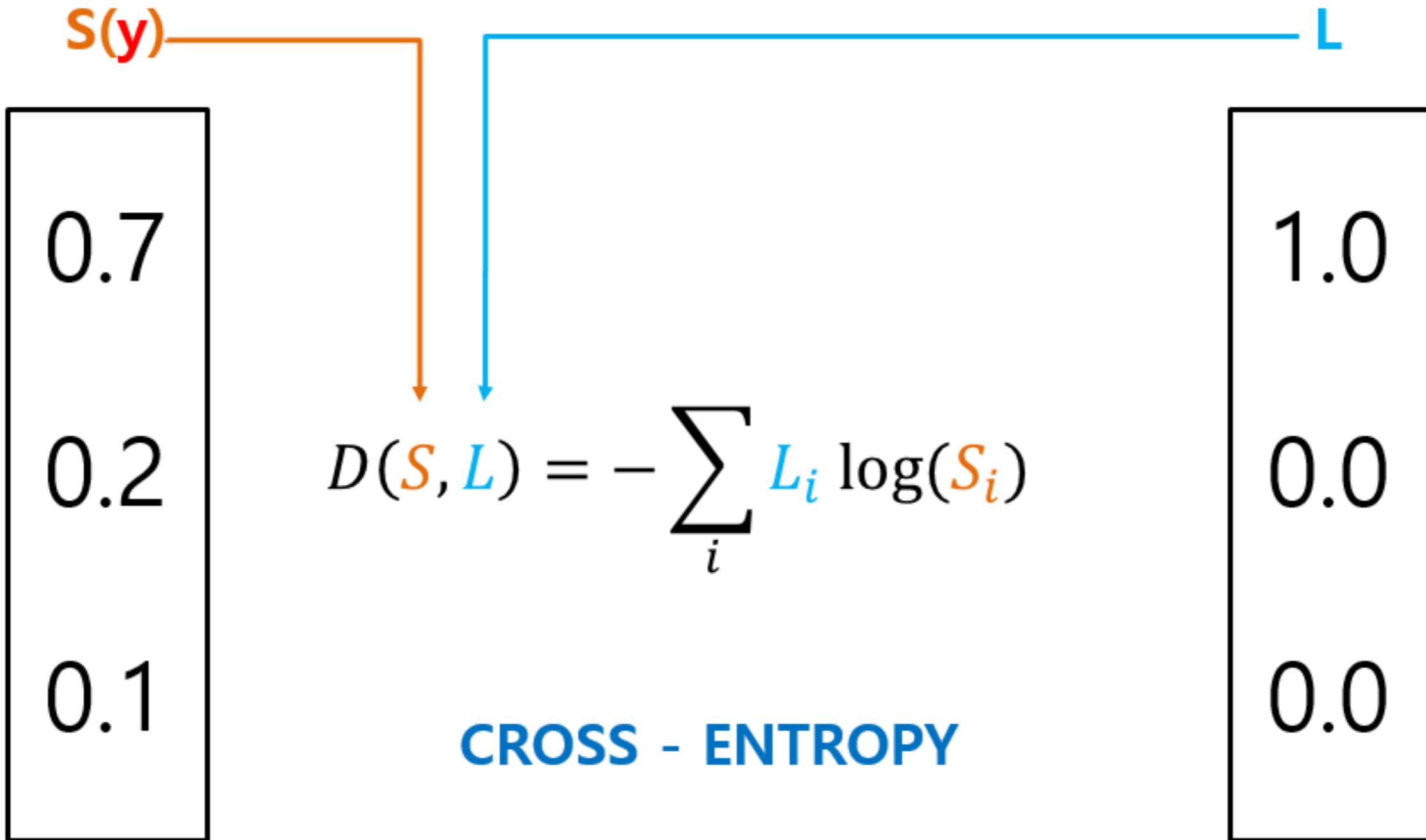
2

```
# Cross entropy cost/Loss  
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,  
                                                labels=Y_one_hot)  
cost = tf.reduce_mean(cost_i)
```

### ONE-HOT ENCODING



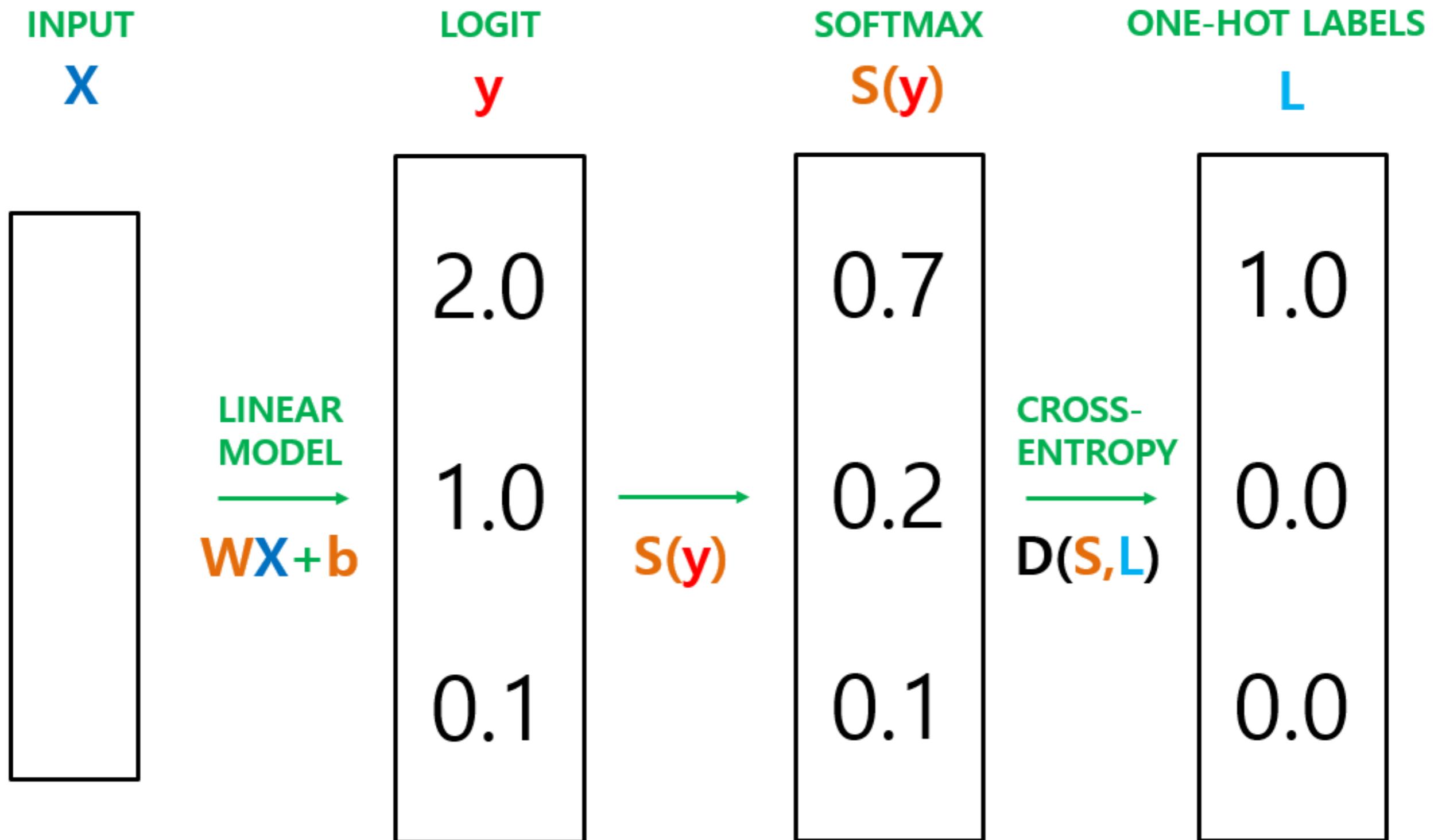
## COST FUNCTION



## SOFTMAX CROSS-ENTROPY

---

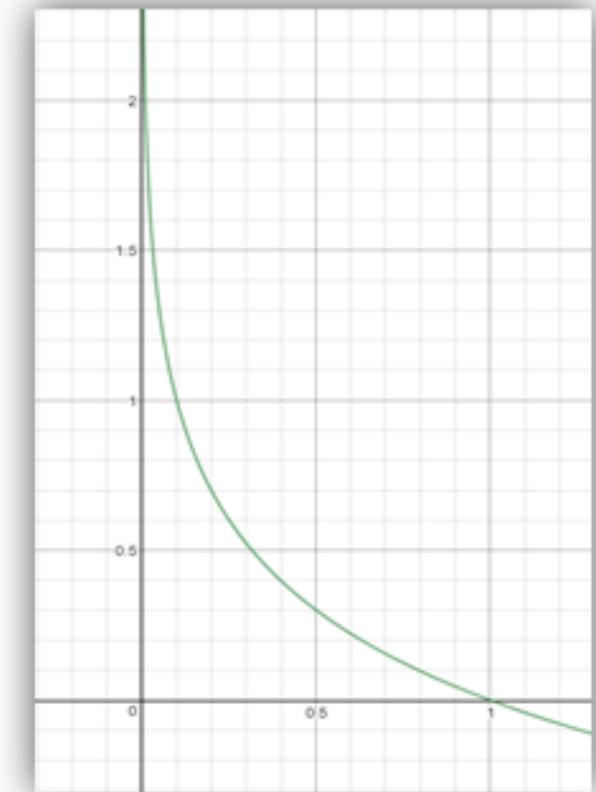
### COST FUNCTION



## CROSS-ENTROPY COST FUNCTION

$$-\sum_i L_i \log(s_i) \rightarrow -\sum_i L_i \log(\hat{y}_i) \rightarrow \sum_i L_i * -\log(\hat{y}_i)$$

$$L = \begin{matrix} A \\ B \end{matrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = B$$



$$\hat{Y} = \begin{matrix} A \\ B \end{matrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = B(0), \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$$\hat{Y} = \begin{matrix} A \\ B \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = A(X), \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty$$

## LOGISTIC COST VS. CROSS ENTROPY

$$C(H(x), y) = -y \log(H(x)) - (1 - y) \log(1 - H(x))$$

$$D(\textcolor{brown}{S}, \textcolor{blue}{L}) = - \sum_i \textcolor{blue}{L}_i \log(\textcolor{brown}{S}_i)$$

# COST FUNCTION : CROSS ENTROPY

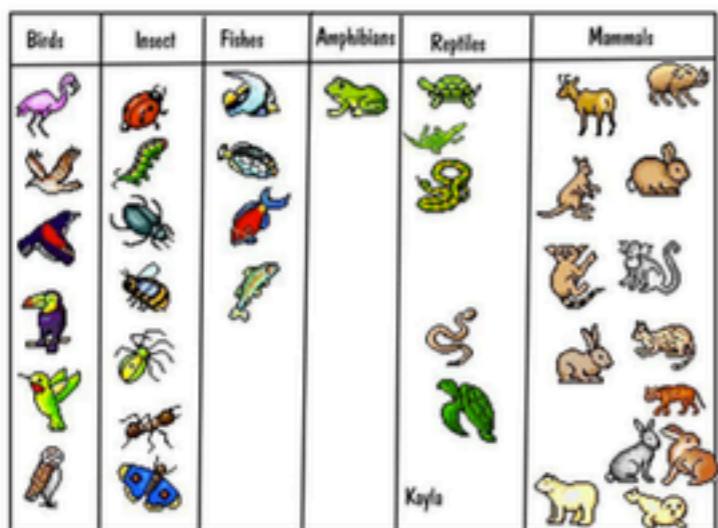
$$L = \frac{1}{N} \sum_i D(S(WX_i + b), L_i)$$

The diagram illustrates the calculation of the total loss  $L$ . It shows a large arrow pointing upwards from the text "TRAINING SET" to the summation symbol in the equation. From the tip of this large arrow, several smaller arrows branch off, each pointing to one of the terms  $D(S(WX_i + b), L_i)$  in the summation, representing the individual losses for each training example  $i$ .

```
# Cross entropy cost/Loss  
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))  
  
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

# ANIMAL CLASSIFICATION

with *softmax\_cross\_entropy\_with\_logits*



1	0	0	1	0	0	0	1	1	0	0	4	3	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	3
1	0	0	0	1	0	0	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	0	1	0	0	1	1	0	0	4	1	0	1	0
1	0	0	0	1	0	0	1	1	0	0	4	1	0	1	0
0	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0
0	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0
0	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0
1	0	0	0	1	0	0	1	1	0	0	4	0	1	0	0
1	0	0	0	1	0	0	1	1	0	0	4	1	0	1	0
0	1	0	0	0	1	0	0	1	1	0	0	2	1	1	0
0	0	0	0	0	1	1	1	1	0	0	1	0	1	0	3
0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	6
0	0	0	0	0	0	1	1	0	0	0	0	4	0	0	6
0	0	0	0	0	0	1	1	0	0	0	0	6	0	0	6
0	1	0	0	0	1	0	0	0	1	1	0	0	2	1	0
1	0	0	0	1	0	0	0	1	1	0	0	4	1	0	1



DEEP LEARNING

---

CONVOLUTIONAL  
NEURAL NETWORK

## A BIT OF HISTORY

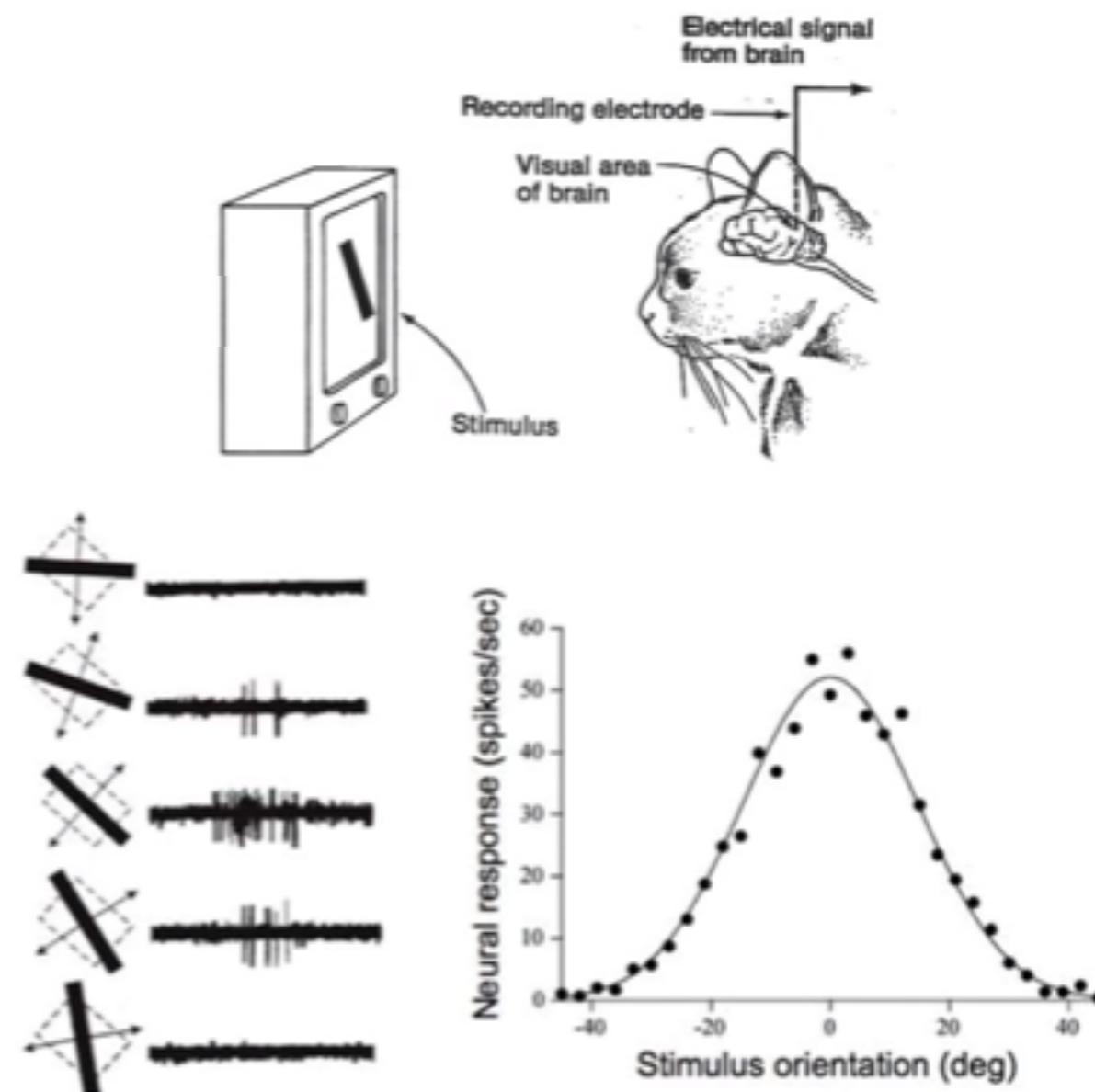
**Hubel & Wiesel,  
1959**

RECEPTIVE FIELDS OF SINGLE  
NEURONES IN  
THE CAT'S STRIATE CORTEX

**1962**

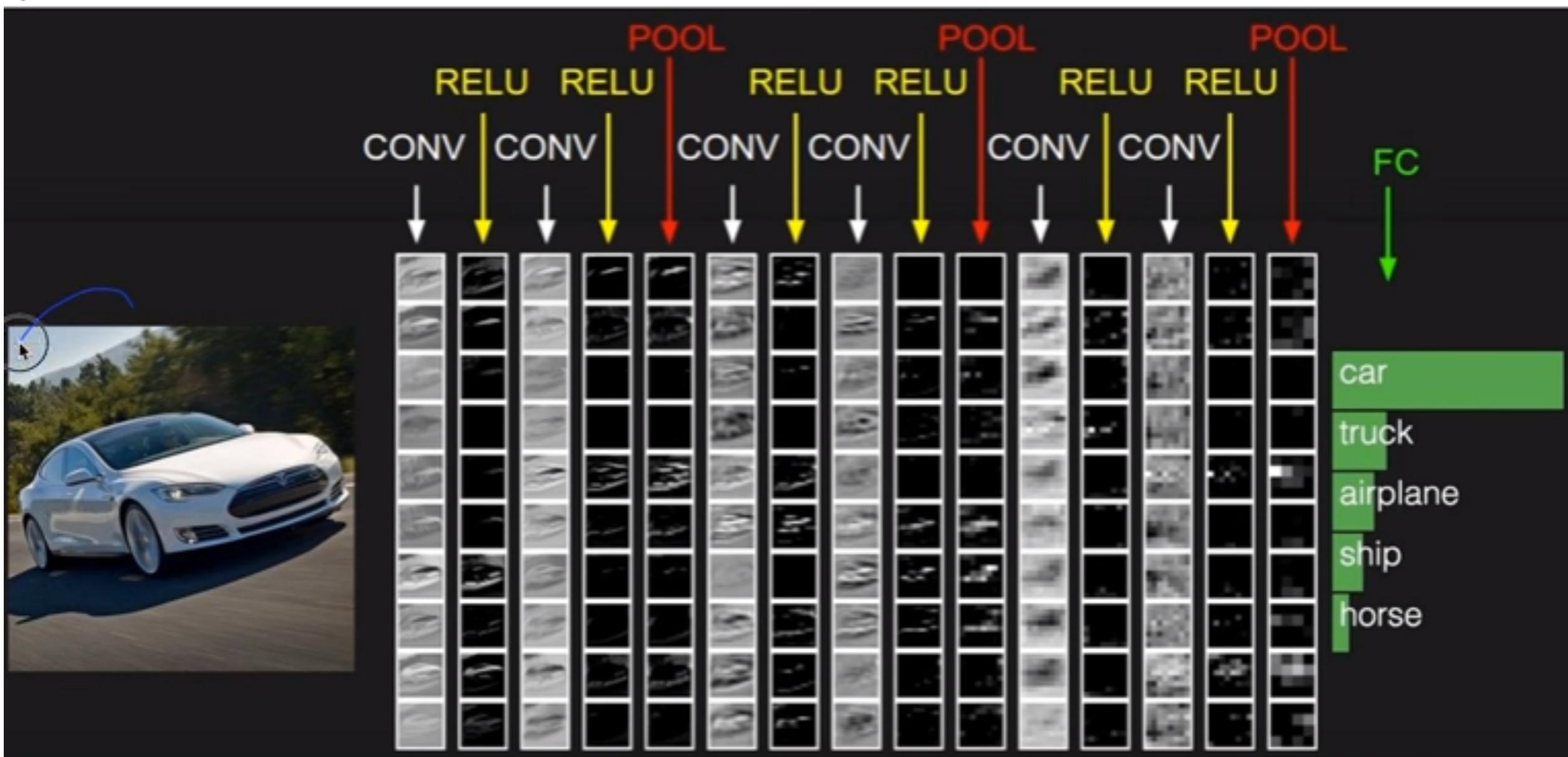
RECEPTIVE FIELDS, BINOCULAR  
INTERACTION  
AND FUNCTIONAL ARCHITECTURE IN  
THE CAT'S VISUAL CORTEX

**1968...**



# CONVOLUTIONAL NEURAL NETWORK

preview:



### CNN FILTER

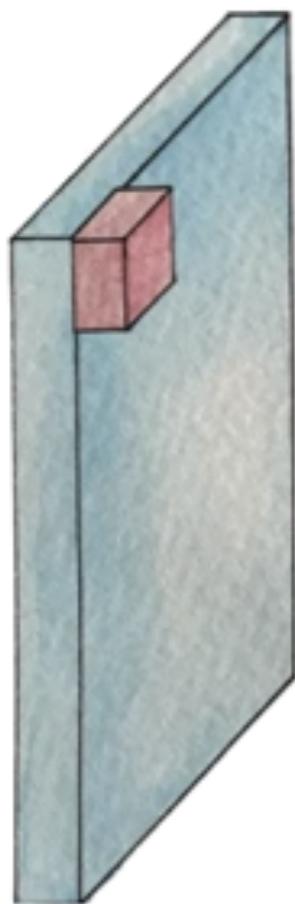
Start with an image (width x height x depth)



32x32x3 image

### CNN FILTER

Let's focus on a small area only ( $5 \times 5 \times 3$ )



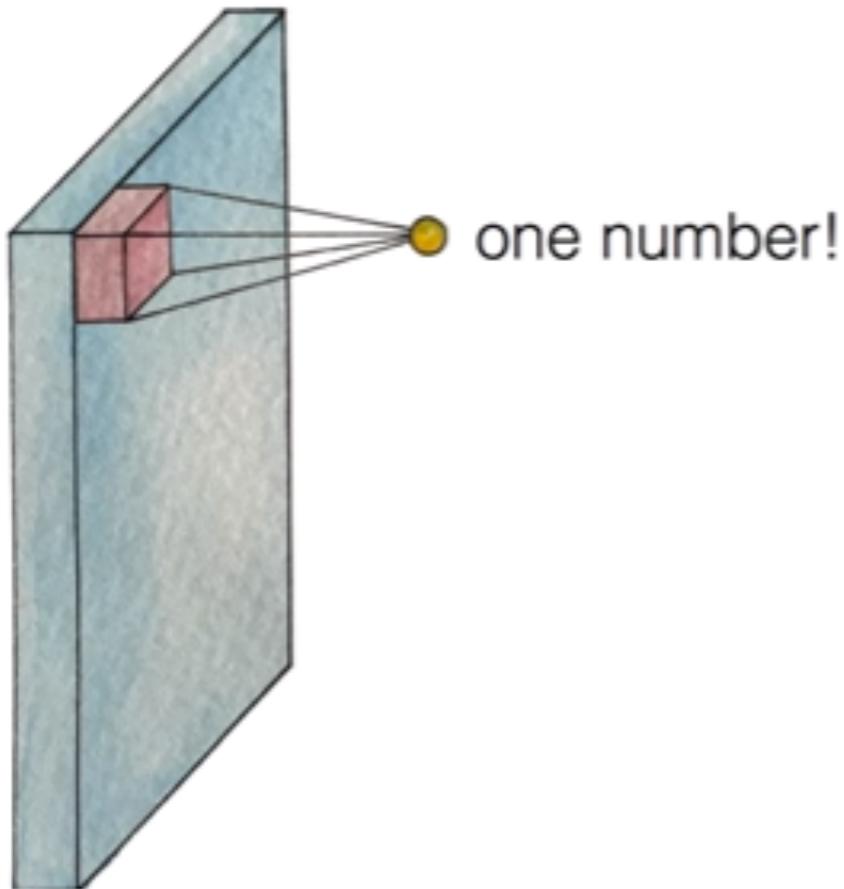
32x32x3 image



5x5x3 filter

### CNN FILTER

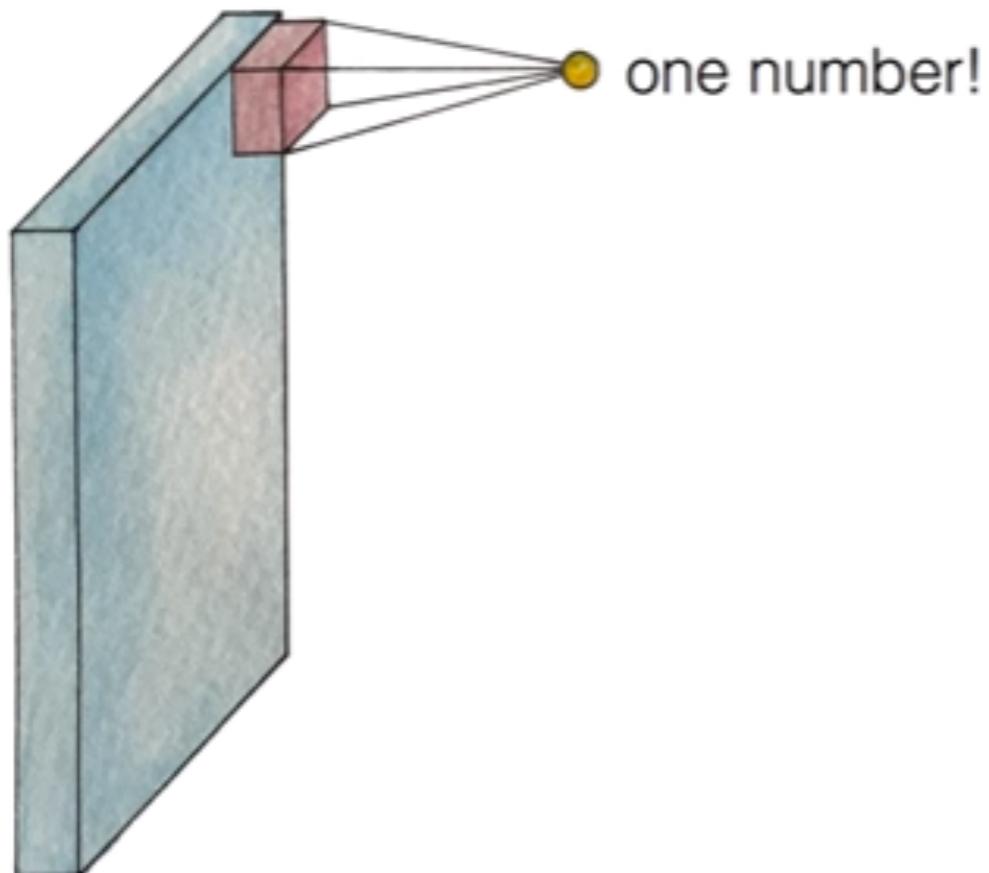
Let's look at other areas with the same filter ( $w$ )



32x32x3 image

### CNN FILTER

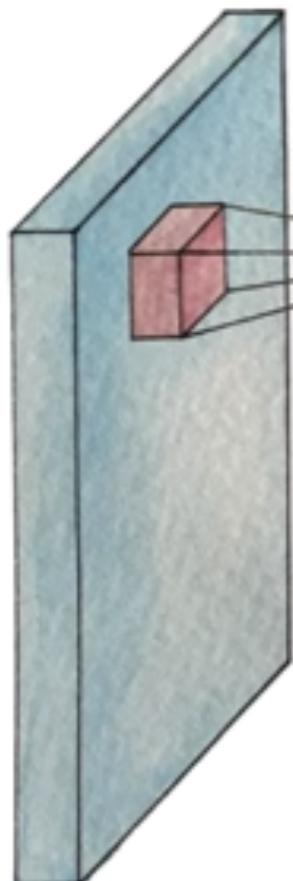
Let's look at other areas with the same filter ( $w$ )



32x32x3 image

### CNN FILTER

Let's look at other areas with the same filter ( $w$ )

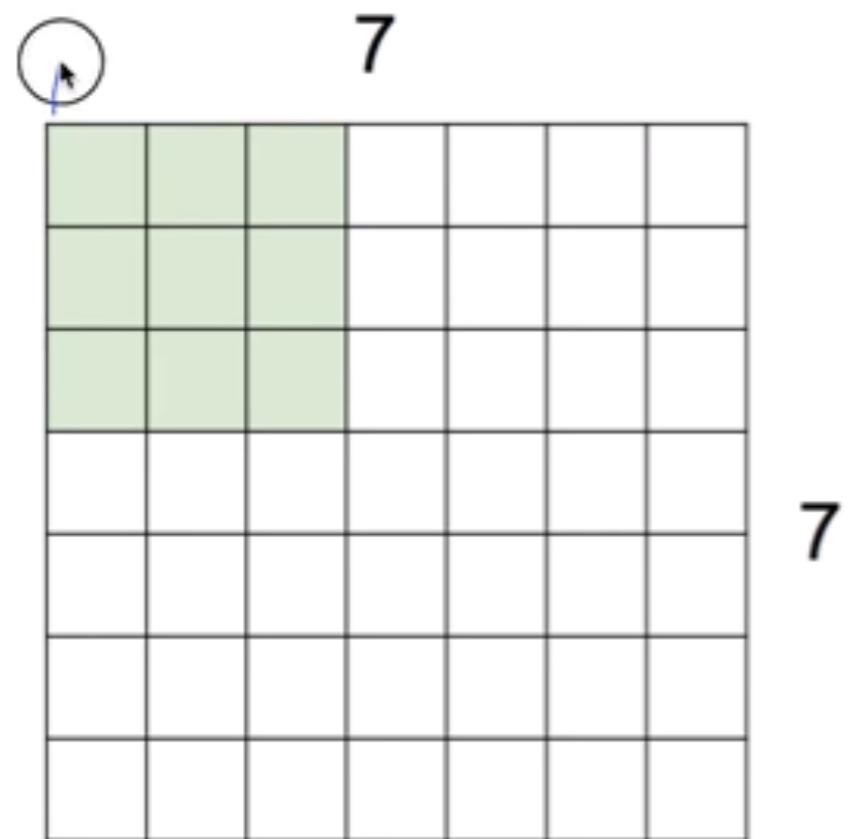


How many numbers  
can we get?

32x32x3 image

## MOVING FILTER

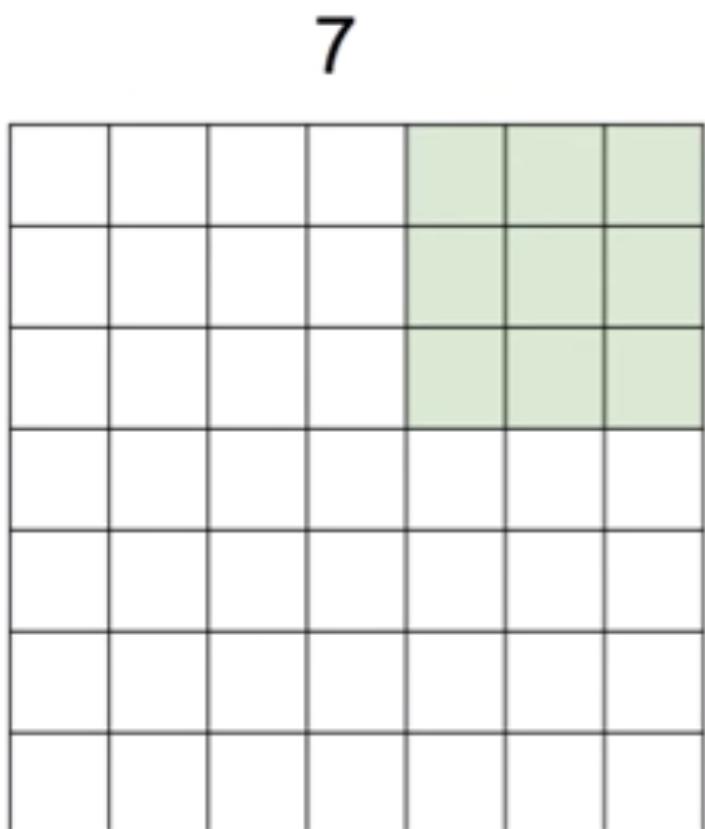
A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter

## MOVING FILTER

A closer look at spatial dimensions:

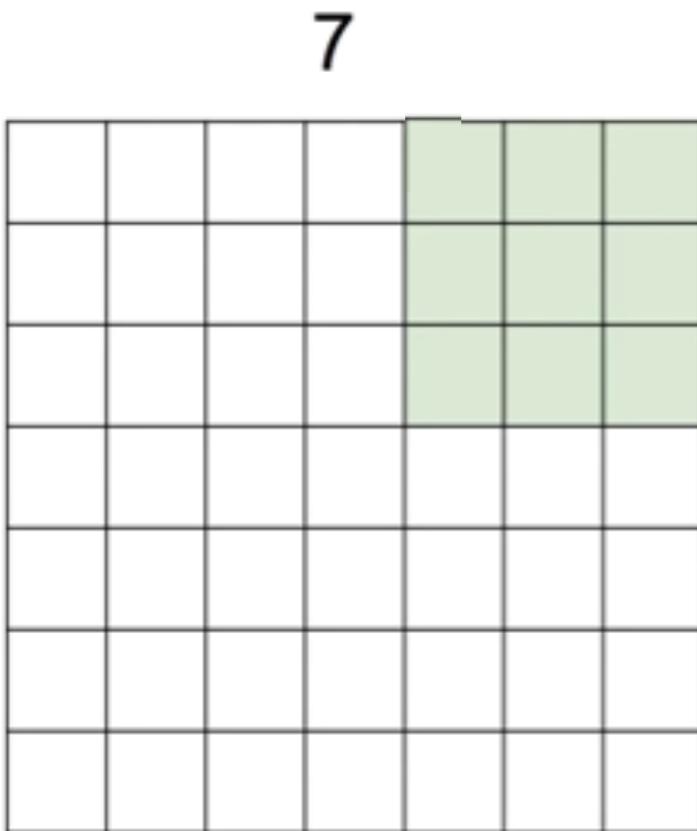


7x7 input (spatially)  
assume 3x3 filter

**=> 5x5 output**

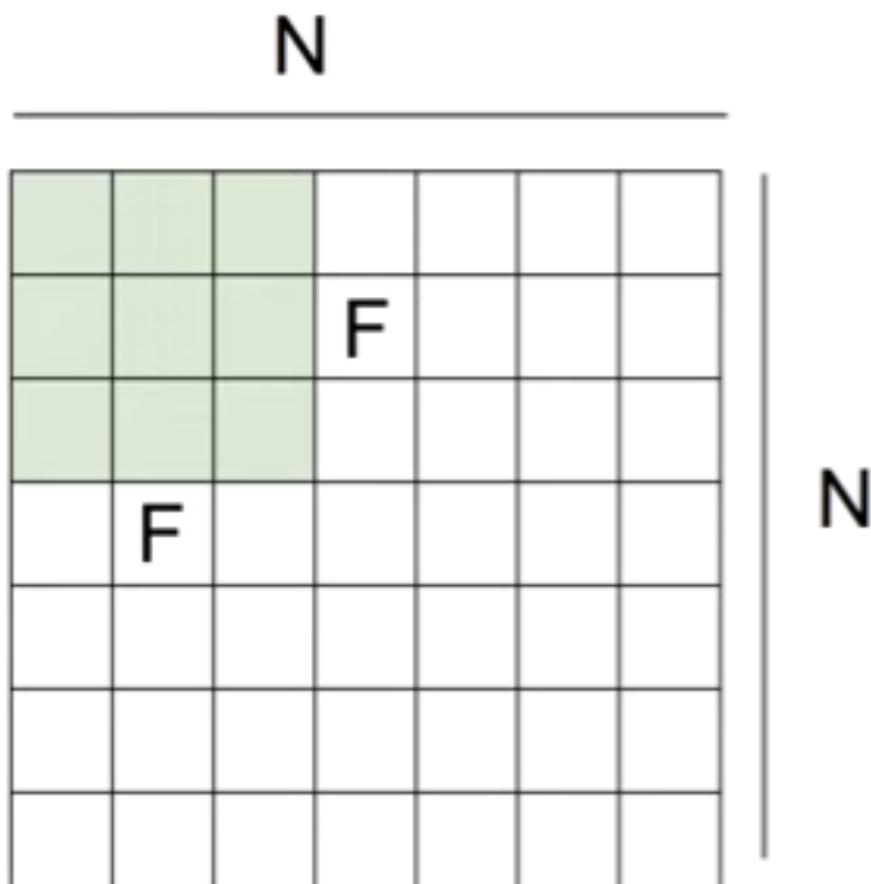
## MOVING FILTER

A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter  
applied **with stride 2**  
**=> 3x3 output!**

## MOVING FILTER



Output size:  
**(N - F) / stride + 1**

e.g.  $N = 7$ ,  $F = 3$ :  
stride 1 =>  $(7 - 3)/1 + 1 = 5$   
stride 2 =>  $(7 - 3)/2 + 1 = 3$   
stride 3 =>  $(7 - 3)/3 + 1 = 2.33$  :\

## ZERO PADDINGS

In practice: Common to zero pad the border

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. input 7x7

3x3 filter, applied with **stride 1**

**pad with 1 pixel border => what is the output?**

**7x7 output!**

in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with  $(F-1)/2$ . (will preserve size spatially)

e.g.  $F = 3 \Rightarrow$  zero pad with 1

$F = 5 \Rightarrow$  zero pad with 2

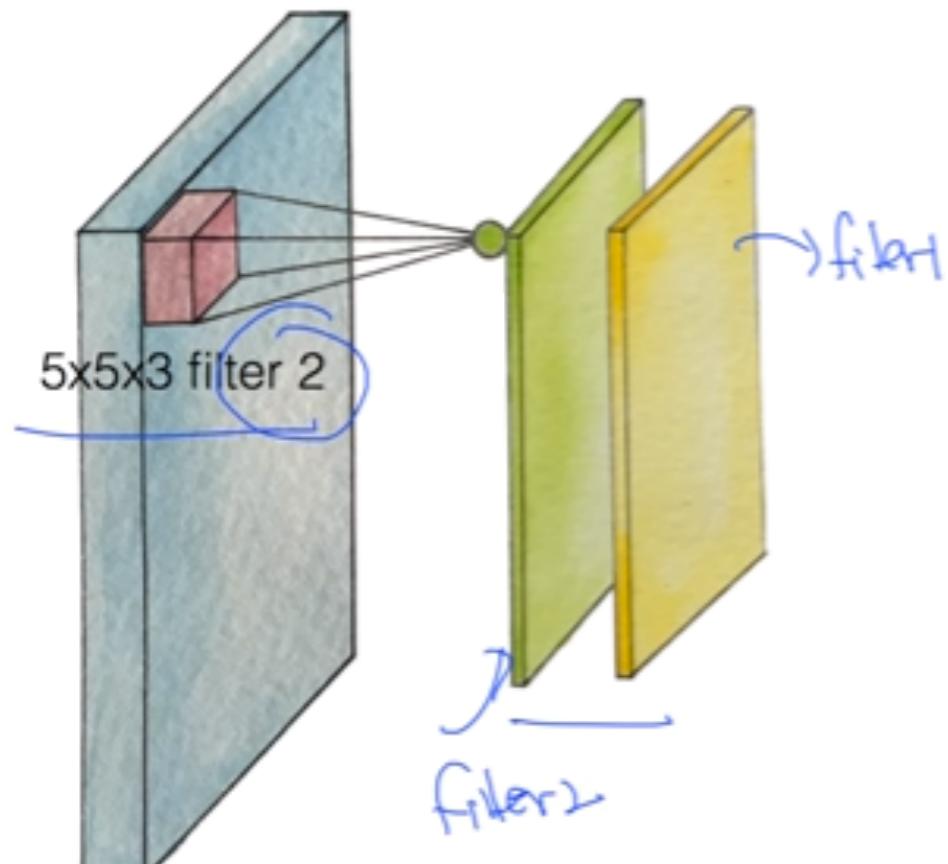
$F = 7 \Rightarrow$  zero pad with 3

## CONVOLUTIONAL NEURAL NETWORK

---

### CONVOLUTIONAL LAYERS

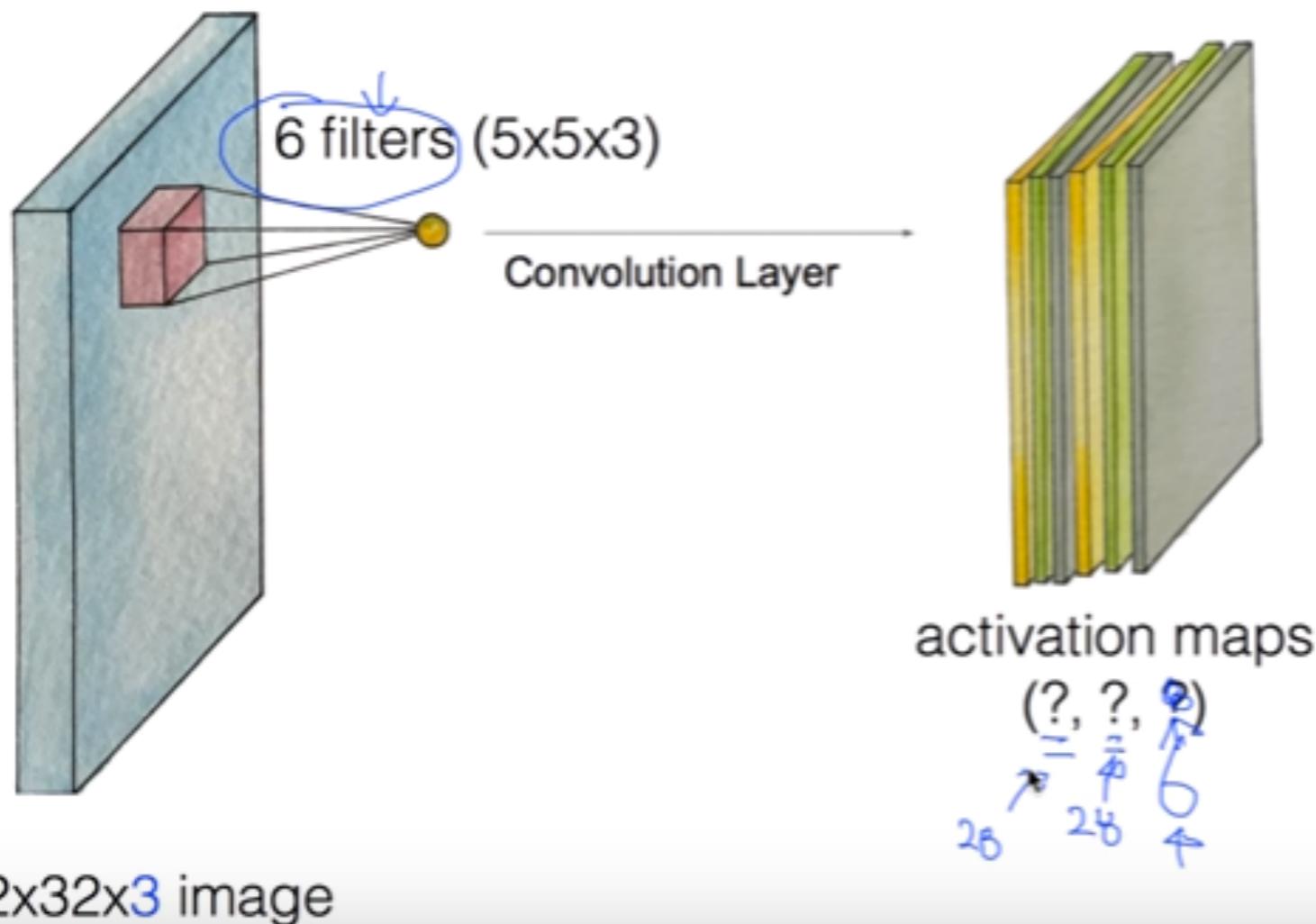
Swiping the entire image



$32 \times 32 \times 3$  image

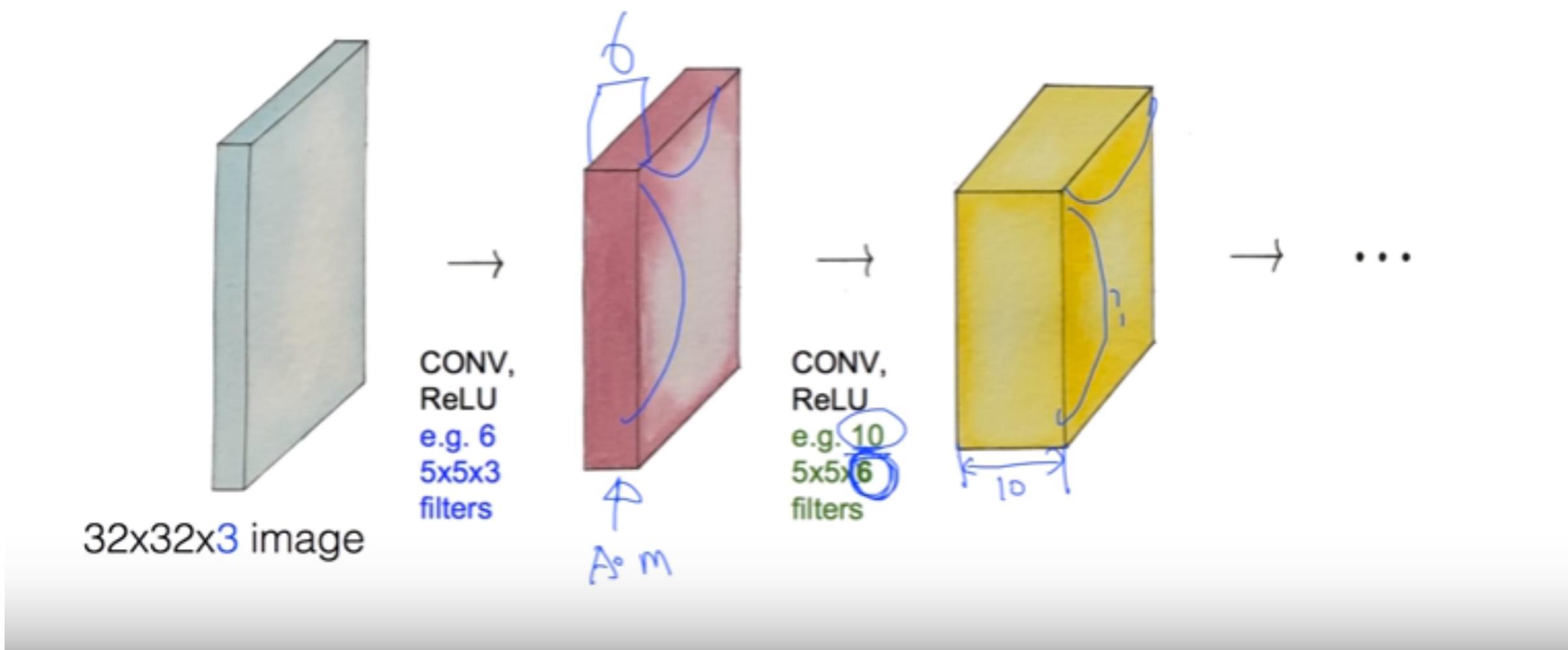
### CONVOLUTIONAL LAYERS

Swiping the entire image



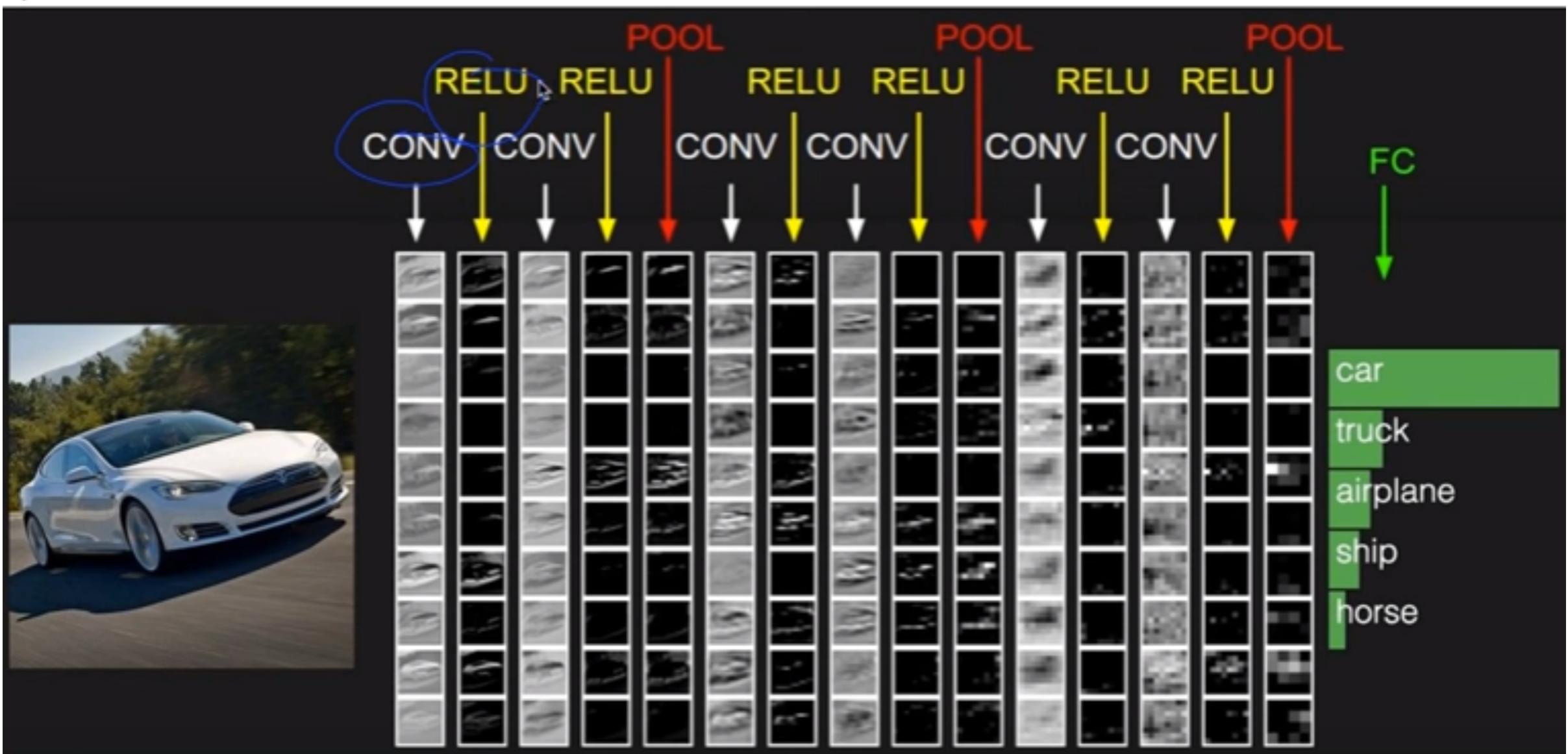
## CONVOLUTIONAL LAYERS

### Convolution layers



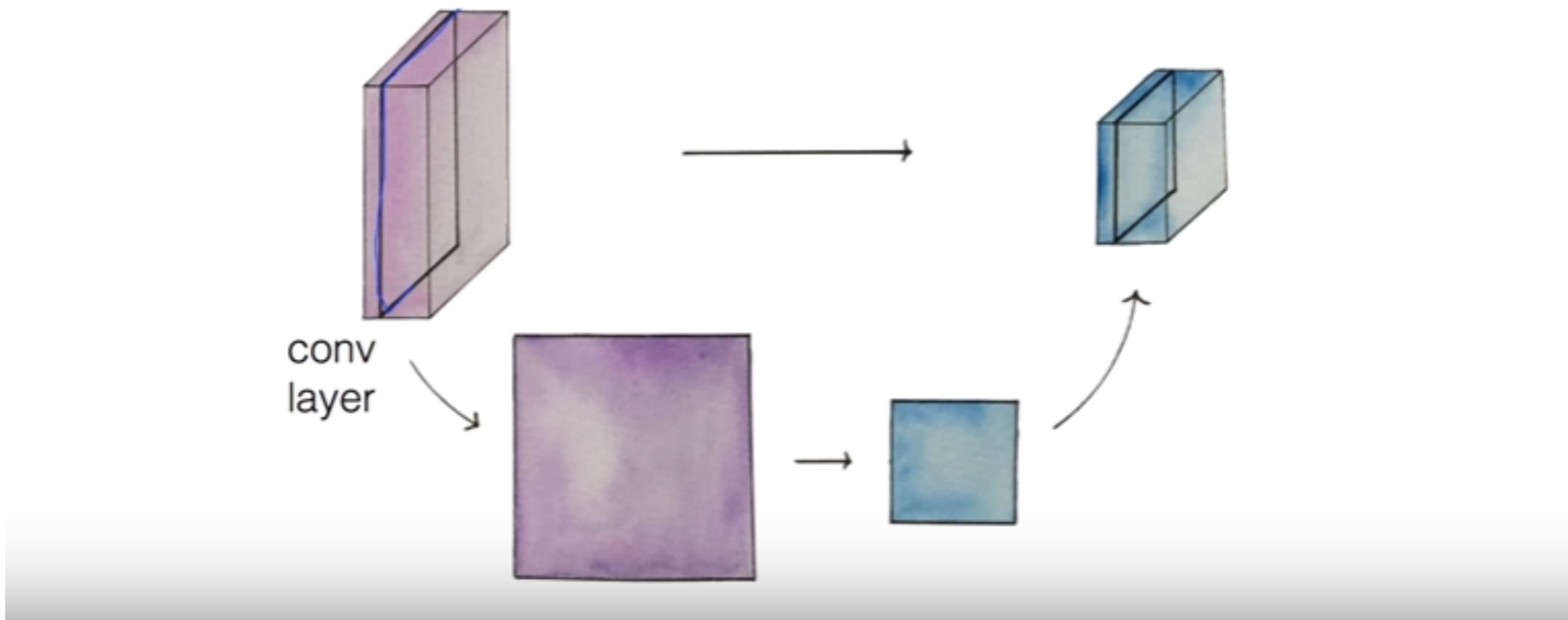
## POOLING LAYER

preview:



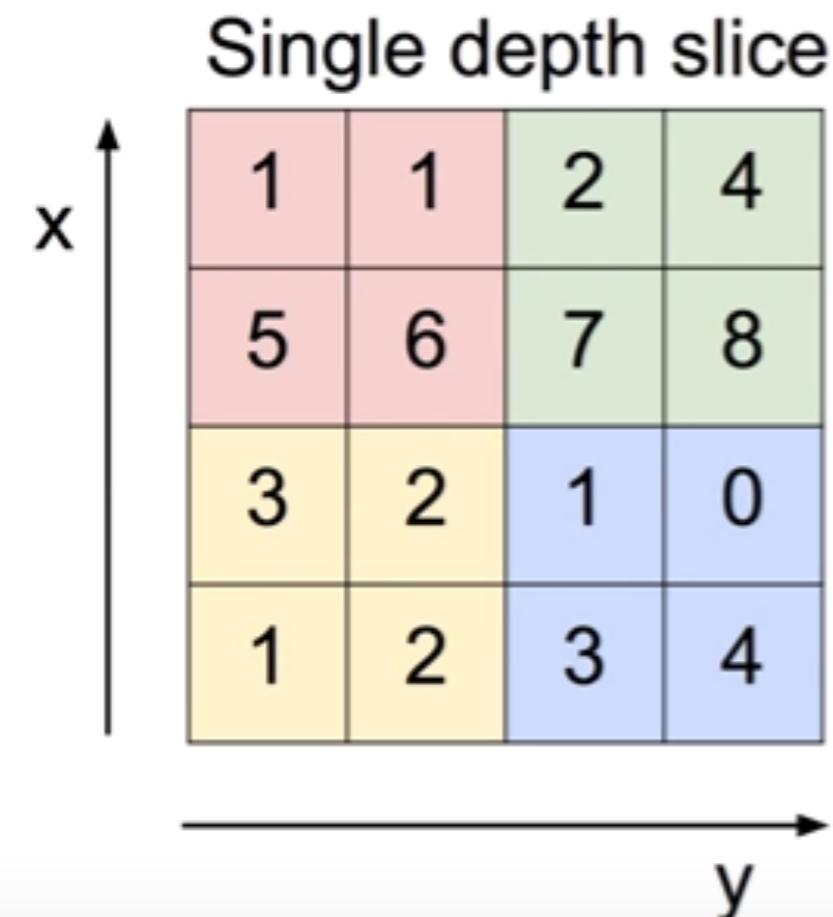
## POOLING LAYER

Pooling layer (sampling)



## POOLING LAYER

### MAX POOLING

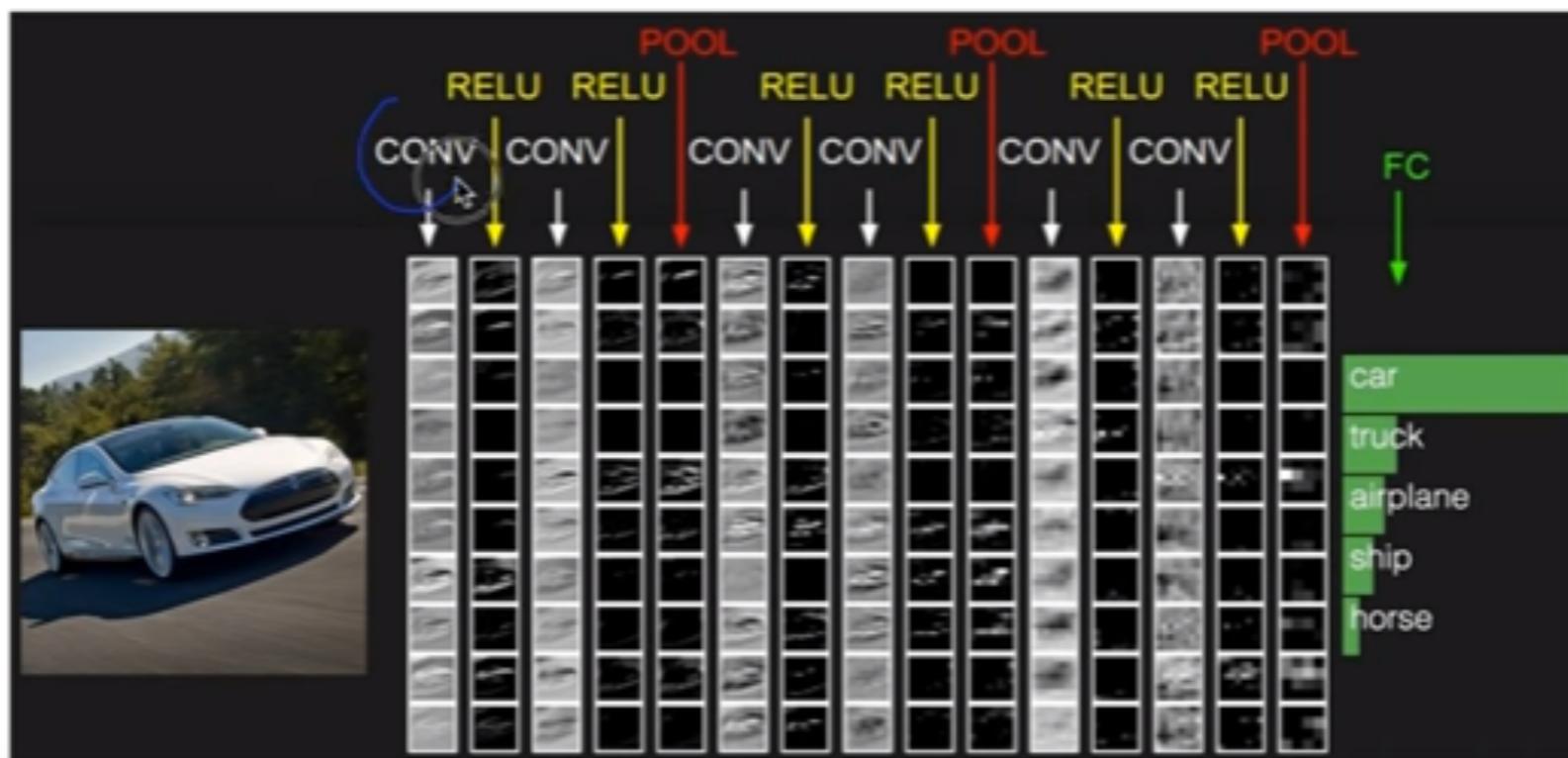


max pool with 2x2 filters  
and stride 2

6	8
3	4

## FULLY CONNECTED LAYER

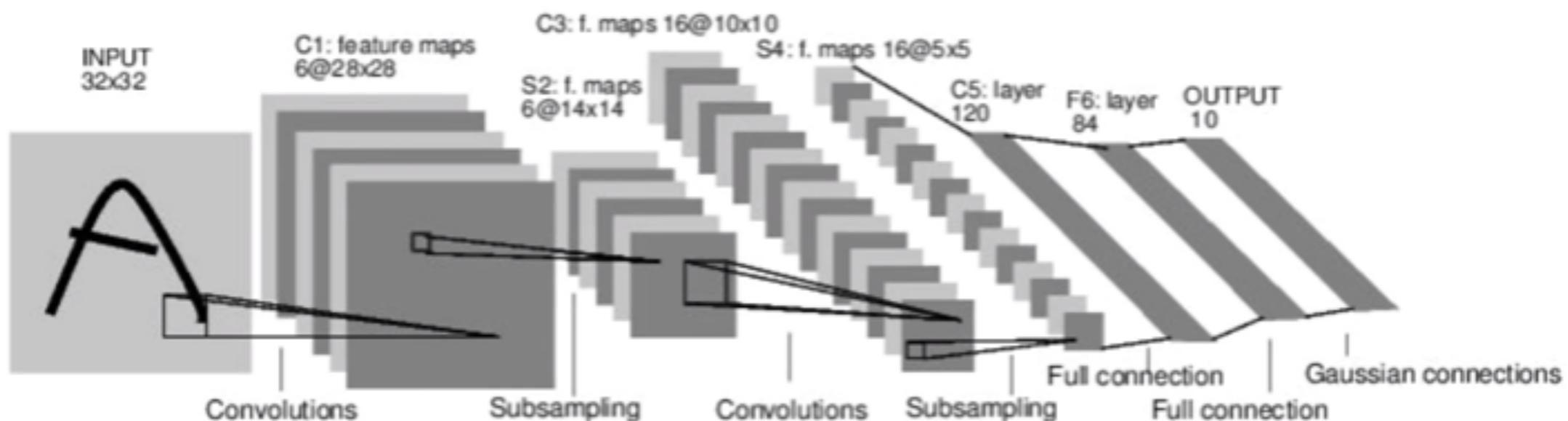
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



## CASE STUDY : LENET-5

### Case Study: LeNet-5

[LeCun et al., 1998]



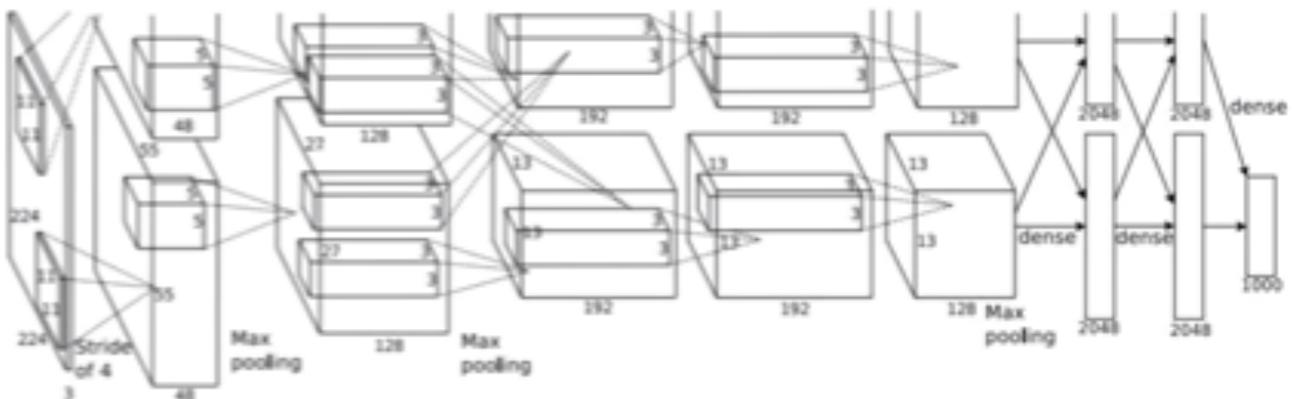
Conv filters were 5x5, applied at stride 1

Subsampling (Pooling) layers were 2x2 applied at stride 2  
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

## CASE STUDY : ALEXNET

### Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

**First layer (CONV1):** 96 11x11 filters applied at stride 4

=>

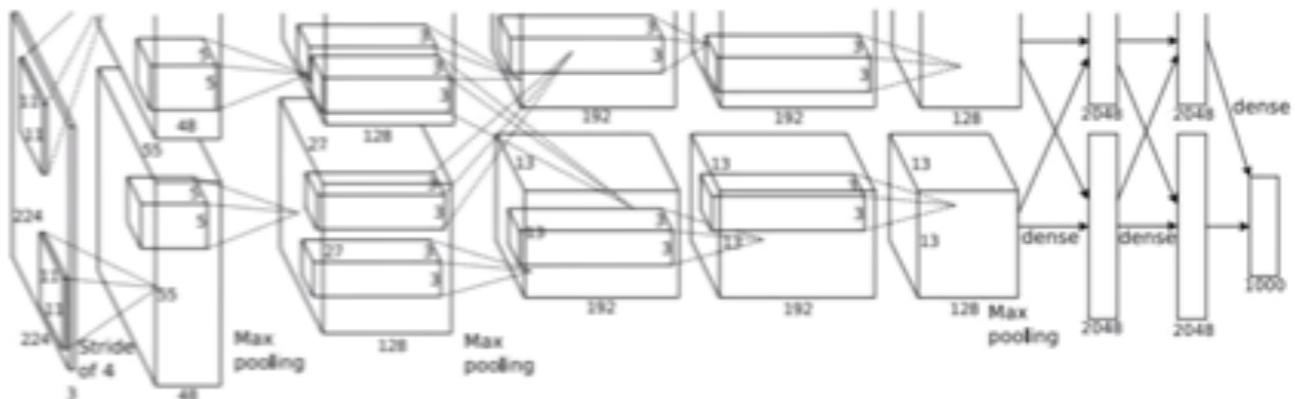
Output volume **[55x55x96]**

Parameters:  $(11 \times 11 \times 3) \times 96 = 35K$

## CASE STUDY : ALEXNET

### Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

After CONV1: 55x55x96

**Second layer (POOL1): 3x3 filters applied at stride 2**

Output volume: 27x27x96

Parameters: 0!

## CASE STUDY : ALEXNET

### Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0,

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

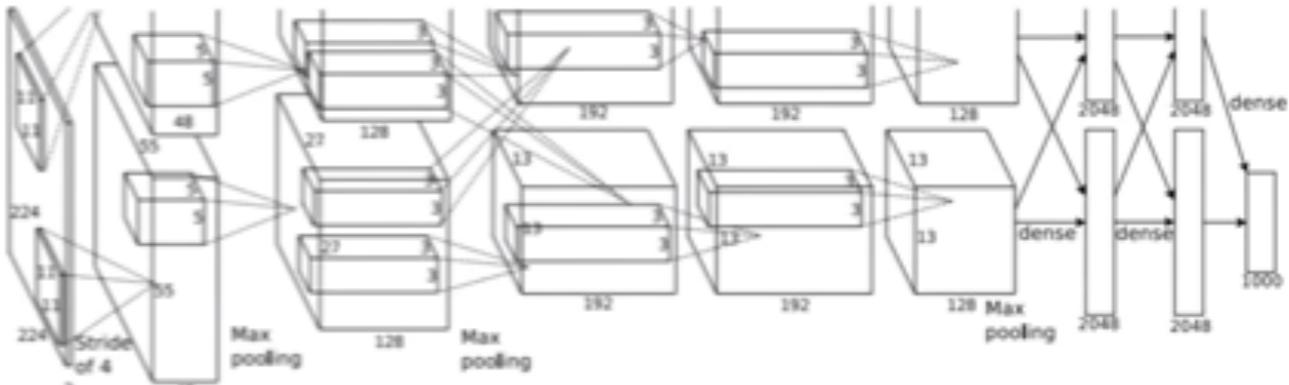
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)



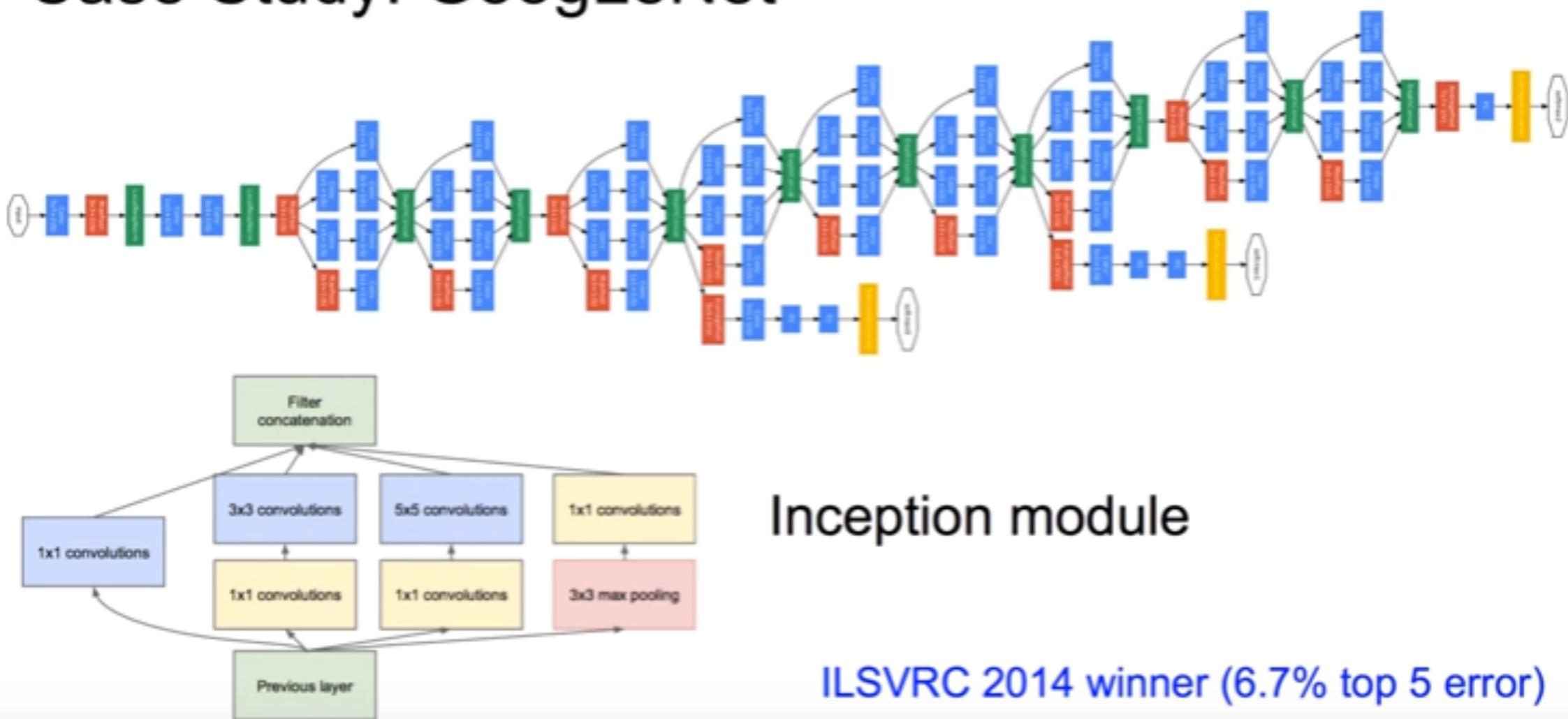
#### Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

## CASE STUDY : GOOGLENET

### Case Study: GoogLeNet

[Szegedy et al., 2014]



## CASE STUDY : RESNET

### Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

Microsoft Research

MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer nets**
  - ImageNet Detection: **16%** better than 2nd
  - ImageNet Localization: **27%** better than 2nd
  - COCO Detection: **11%** better than 2nd
  - COCO Segmentation: **12%** better than 2nd

\*improvements are relative numbers

ICCV15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

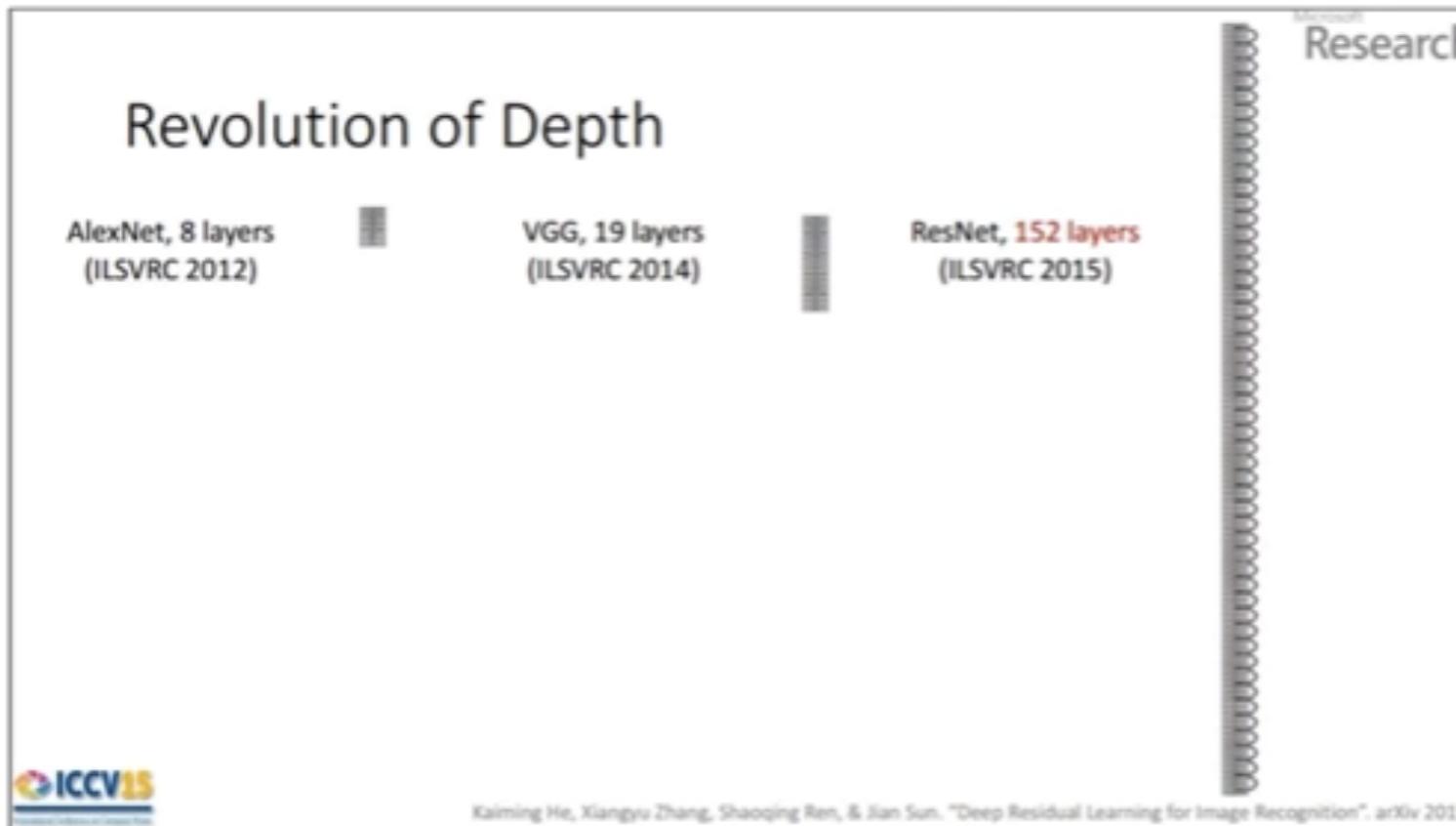
Slide from Kaiming He's recent presentation <https://www.youtube.com/watch?v=1PGLj-uKT1w>

## CASE STUDY : RESNET

### Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

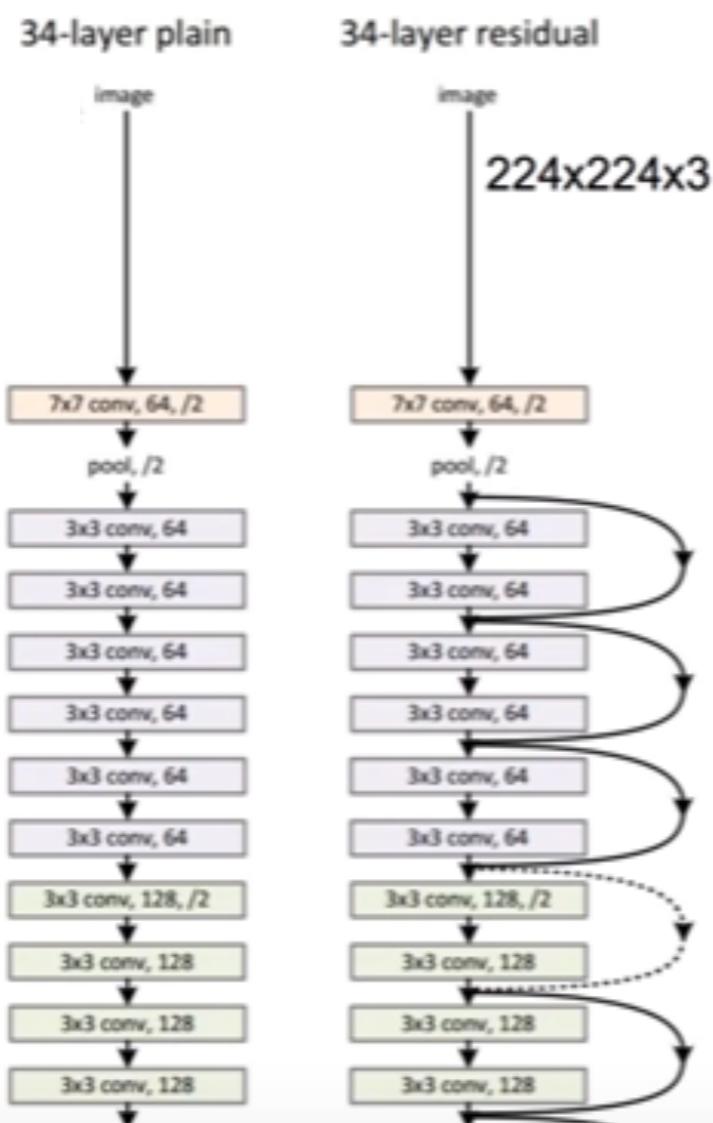


(slide from Kaiming He's recent presentation)

## CASE STUDY : RESNET

### Case Study: ResNet

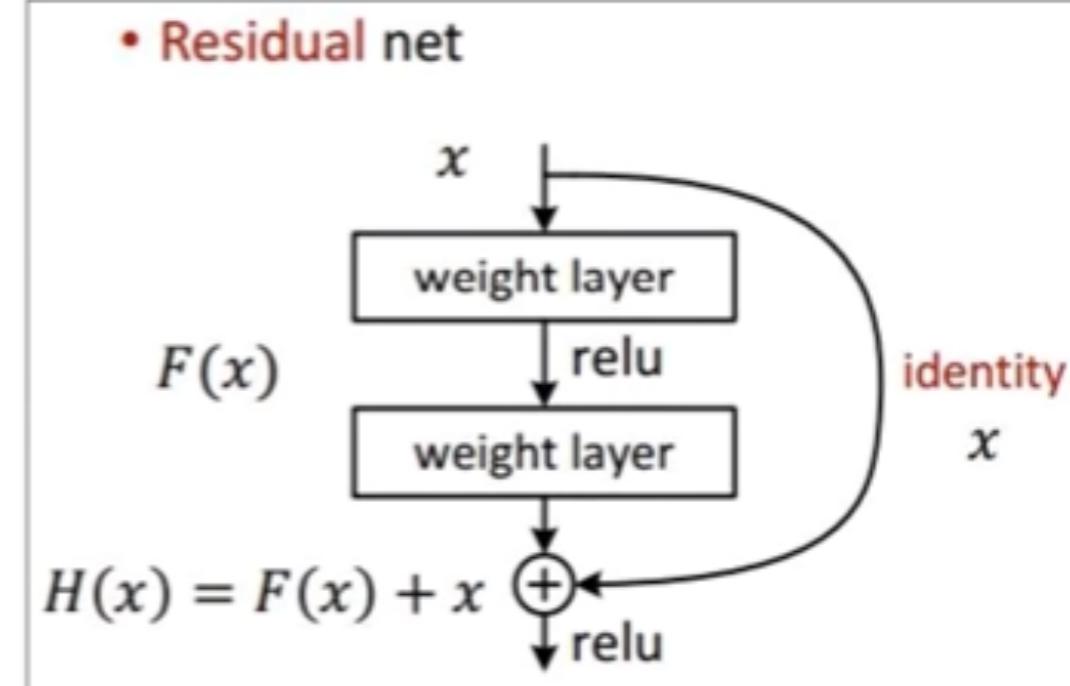
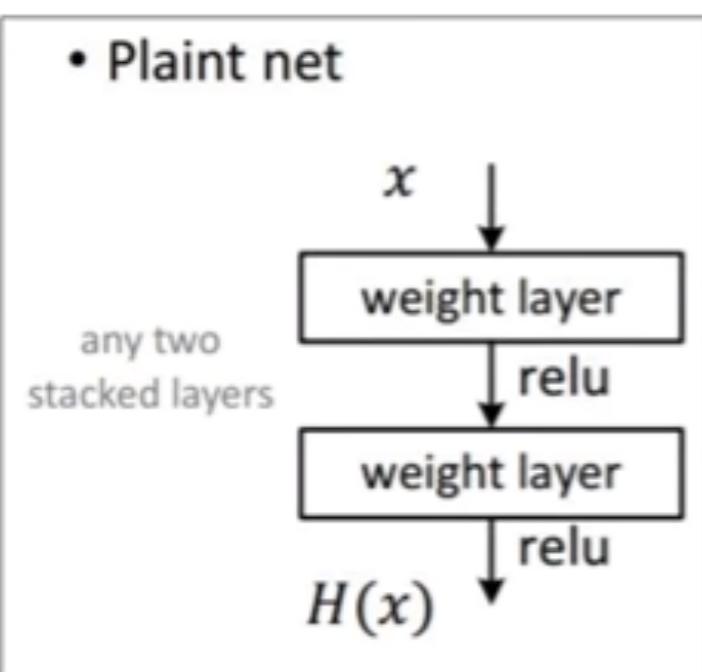
[He et al., 2015]



## CASE STUDY : RESNET

### Case Study: ResNet

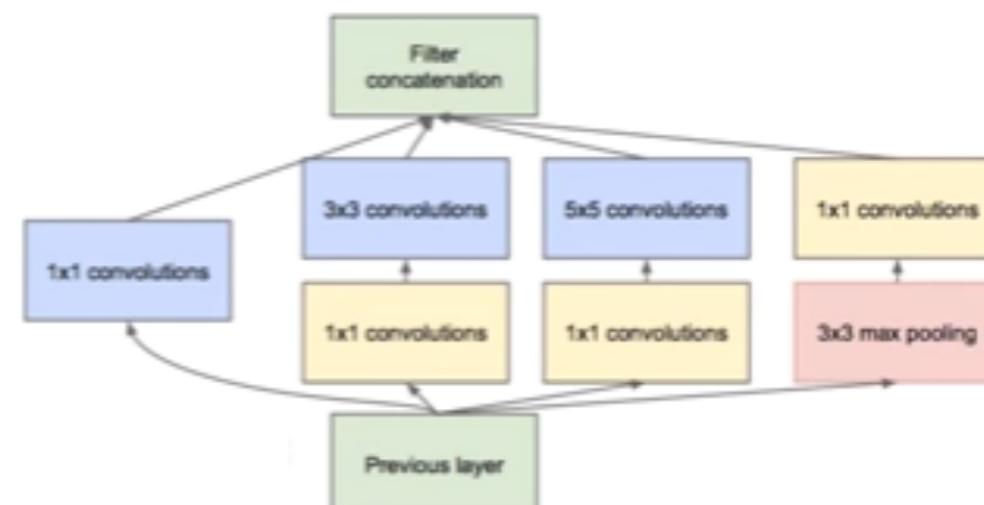
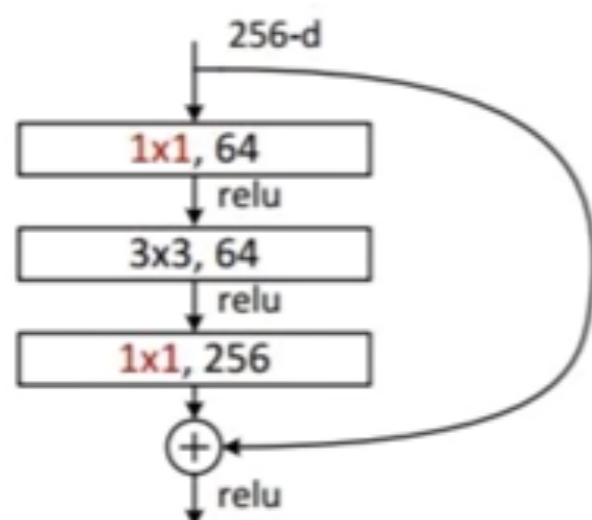
[He et al., 2015]



## CASE STUDY : RESNET

### Case Study: ResNet

[He et al., 2015]



# CASE STUDY : SENTENCE CLASSIFICATION

Convolutional Neural Networks for Sentence Classification  
[Yoon Kim, 2014]

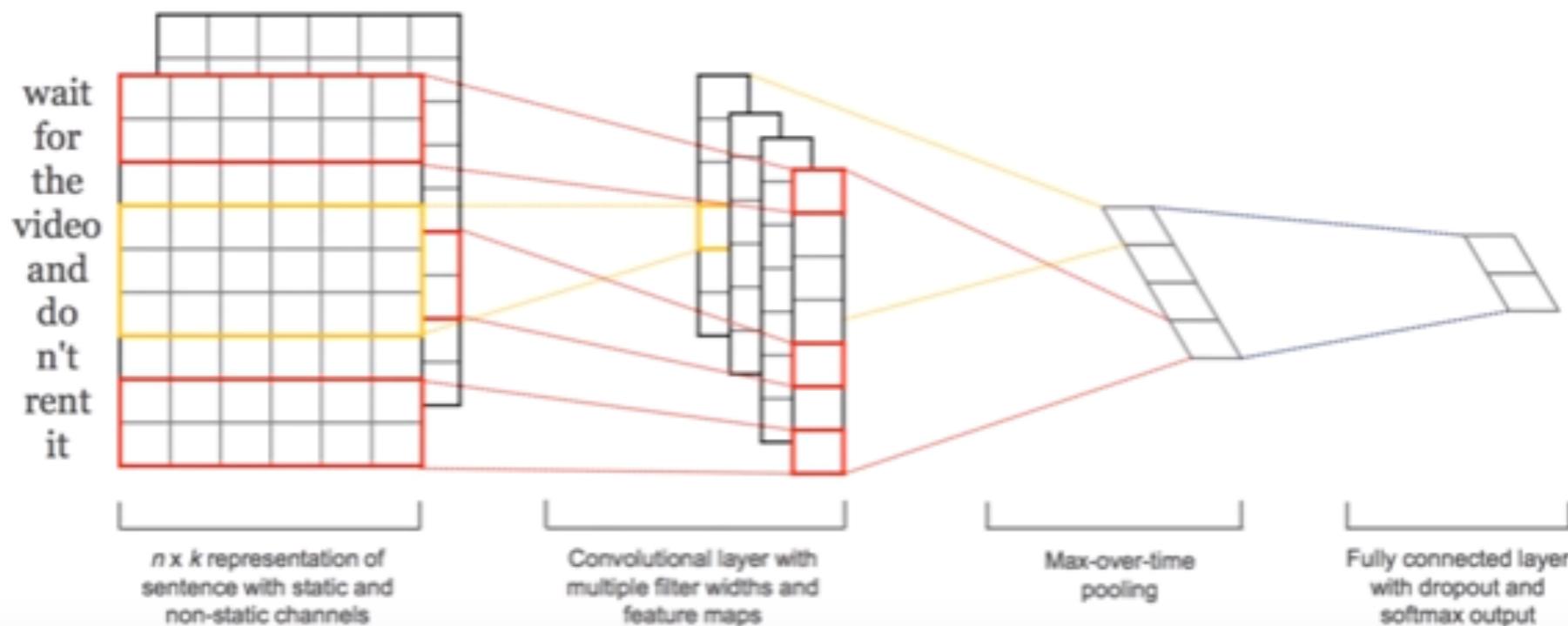
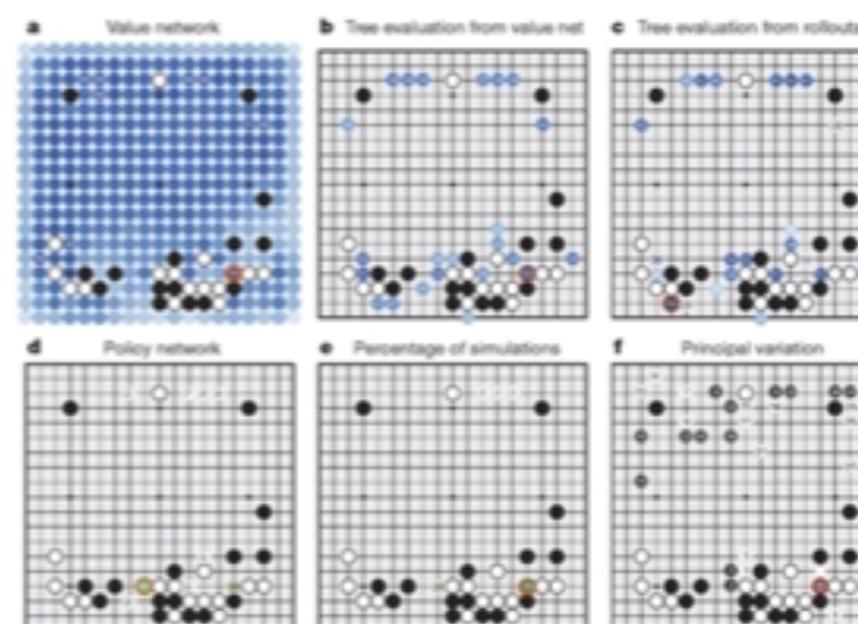


Figure 1: Model architecture with two channels for an example sentence.

## CASE STUDY : DEEPMIND'S ALPHAGO

Case Study Bonus: DeepMind's AlphaGo



## CASE STUDY : DEEPMIND'S ALPHAGO

The input to the policy network is a  $19 \times 19 \times 48$  image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a  $23 \times 23$  image, then convolves  $k$  filters of kernel size  $5 \times 5$  with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a  $21 \times 21$  image, then convolves  $k$  filters of kernel size  $3 \times 3$  with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size  $1 \times 1$  with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used  $k = 192$  filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with  $k = 128, 256$  and  $384$  filters.

### policy network:

[ $19 \times 19 \times 48$ ] Input

CONV1: 192  $5 \times 5$  filters , stride 1, pad 2 => [ $19 \times 19 \times 192$ ]

CONV2..12: 192  $3 \times 3$  filters, stride 1, pad 1 => [ $19 \times 19 \times 192$ ]

CONV: 1  $1 \times 1$  filter, stride 1, pad 0 => [ $19 \times 19$ ] (*probability map of promising moves*)



DEEP LEARNING

---

UTILITY

## NUMPY : SLICING

```
nums = range(5)      # range is a built-in function that creates a list of integers
print nums           # Prints "[0, 1, 2, 3, 4]"
print nums[2:4]       # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print nums[2:]        # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print nums[:2]        # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print nums[:]         # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print nums[:-1]       # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]    # Assign a new sublist to a slice
print nums           # Prints "[0, 1, 8, 9, 4]"
```

# NUMPY : INDEXING, SLICING, ITERATING

- Arrays can be indexed, sliced, iterated much like lists and other sequence types in Python
- As with Python lists, slicing in NumPy can be accomplished with the colon (:) syntax
- Colon instances (:) can be replaced with dots ( ... )

```
a = np.array([1, 2, 3, 4, 5])
# array([1, 2, 3, 4, 5])

a[1:3]
# array([2, 3])

a[-1]
# 5

a[0:2] = 9

a
# array([9, 9, 3, 4, 5])
```

```
b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
# array([[ 1,  2,  3,  4],
#        [ 5,  6,  7,  8],
#        [ 9, 10, 11, 12]])

b[:, 1]
# array([ 2,  6, 10])

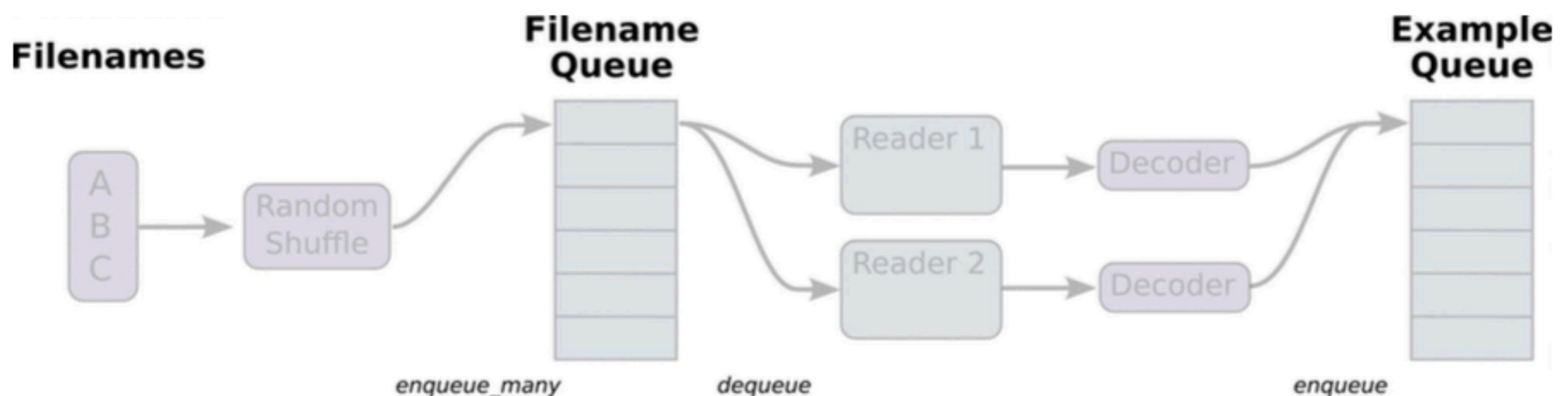
b[-1]
# array([ 9, 10, 11, 12])

b[-1, :]
# array([ 9, 10, 11, 12])

b[-1, ...]
# array([ 9, 10, 11, 12])

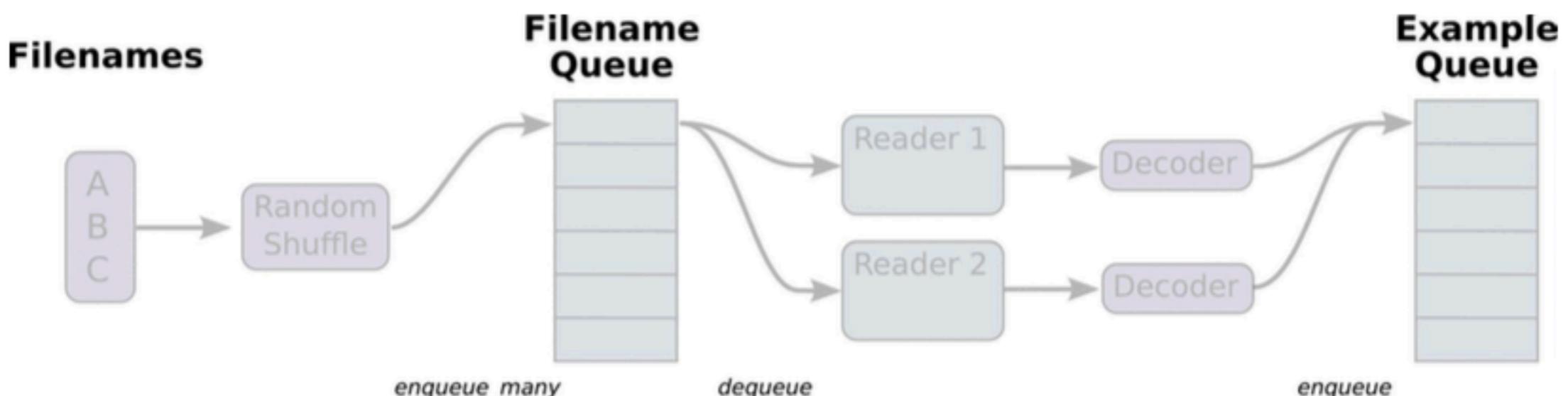
b[0:2, :]
# array([[1, 2, 3, 4],
#        [5, 6, 7, 8]])
```

# QUEUE RUNNERS (1)



## QUEUE RUNNERS (2)

```
1 filename_queue = tf.train.string_input_producer(  
    ['data-01-test-score.csv', 'data-02-test-score.csv', ... ],  
    shuffle=False, name='filename_queue')  
  
3 record_defaults = [[0.], [0.], [0.], [0.]]  
xy = tf.decode_csv(value, record_defaults=record_defaults)
```



```
2 reader = tf.TextLineReader()  
key, value = reader.read(filename_queue)
```

## TF.TRAIN.BATCH (1)

```
# collect batches of csv in
train_x_batch, train_y_batch = \
    tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)

sess = tf.Session()
...
# Start populating the filename queue.
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

for step in range(2001):
    x_batch, y_batch = sess.run([train_x_batch, train_y_batch])
    ...
coord.request_stop()
coord.join(threads)
```

## TF.TRAIN.BATCH (2)

```
import tensorflow as tf
filename_queue = tf.train.string_input_producer(
    ['data-01-test-score.csv'], shuffle=False, name='filename_queue')

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)

# Default values, in case of empty columns. Also specifies the type of the
# decoded result.
record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)

# collect batches of csv in
train_x_batch, train_y_batch = \
    tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Start populating the filename queue.
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

for step in range(2001):
    x_batch, y_batch = sess.run([train_x_batch, train_y_batch])
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train],
        feed_dict={X: x_batch, Y: y_batch})
    if step % 10 == 0:
        print(step, "Cost: ", cost_val,
              "\nPrediction:\n", hy_val)

coord.request_stop()
coord.join(threads)
```

## TF.TRAIN.SHUFFLE\_BATCH

```
# min_after_dequeue defines how big a buffer we will randomly sample
# from -- bigger means better shuffling but slower start up and more
# memory used.
# capacity must be larger than min_after_dequeue and the amount larger
# determines the maximum we will prefetch. Recommendation:
# min_after_dequeue + (num_threads + a small safety margin) * batch_size
min_after_dequeue = 10000
capacity = min_after_dequeue + 3 * batch_size
example_batch, label_batch = tf.train.shuffle_batch(
    [example, label], batch_size=batch_size, capacity=capacity,
    min_after_dequeue=min_after_dequeue)
```



# DEEP LEARNING

---

# NEURAL NETWORKS

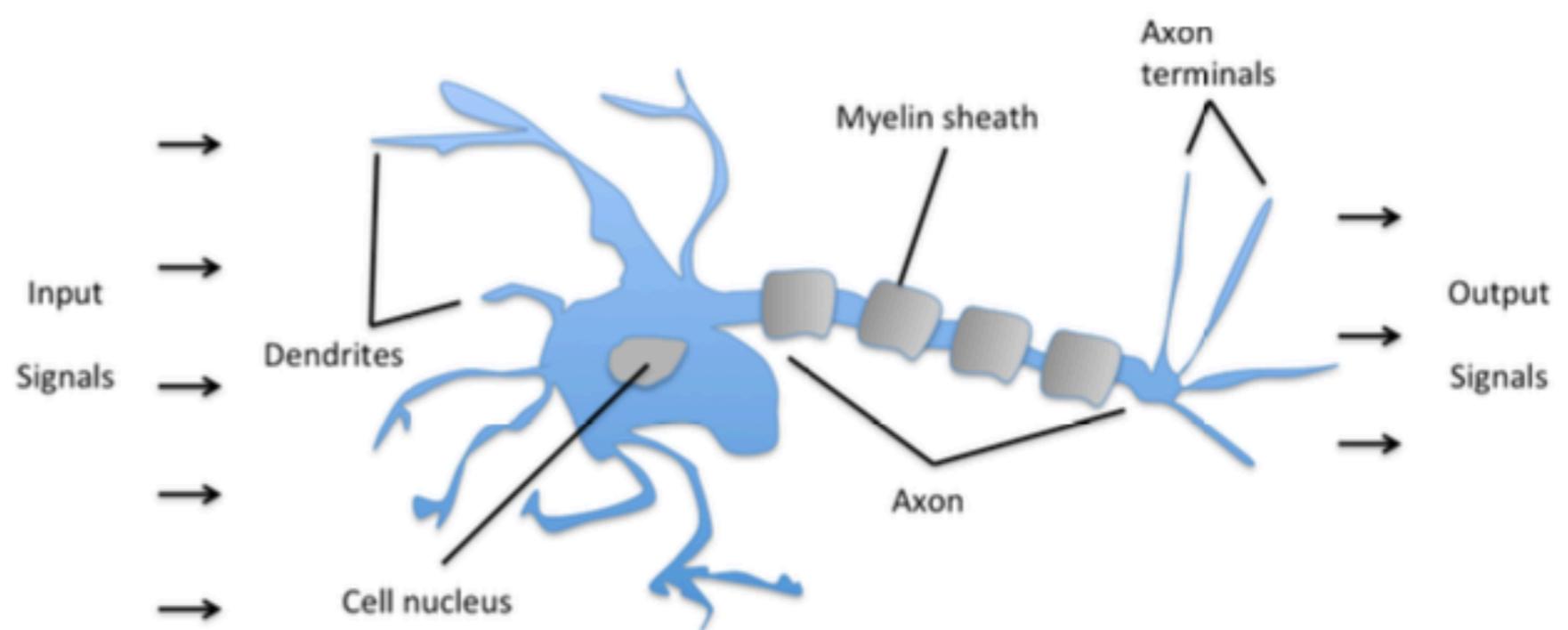
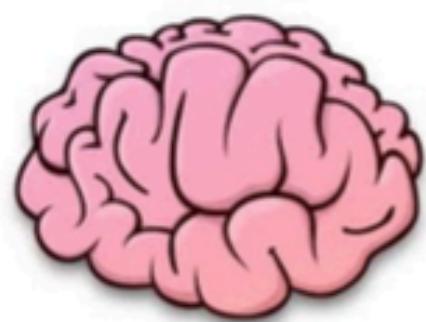
# HISTORY

# THE GOAL OF MACHINE LEARNING

Ultimate dream: thinking machine

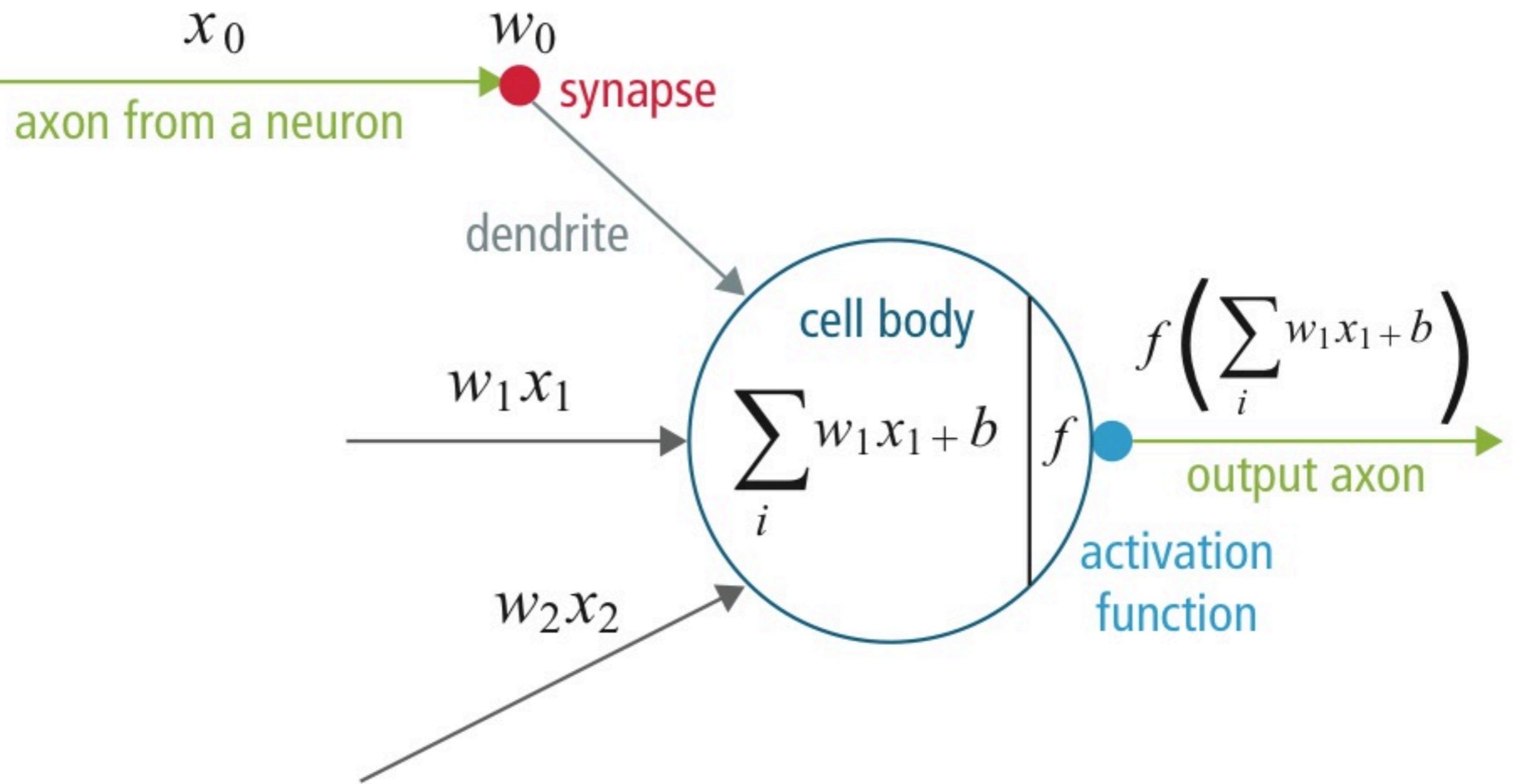


# SCHEMATIC OF A BIOLOGICAL NEURON

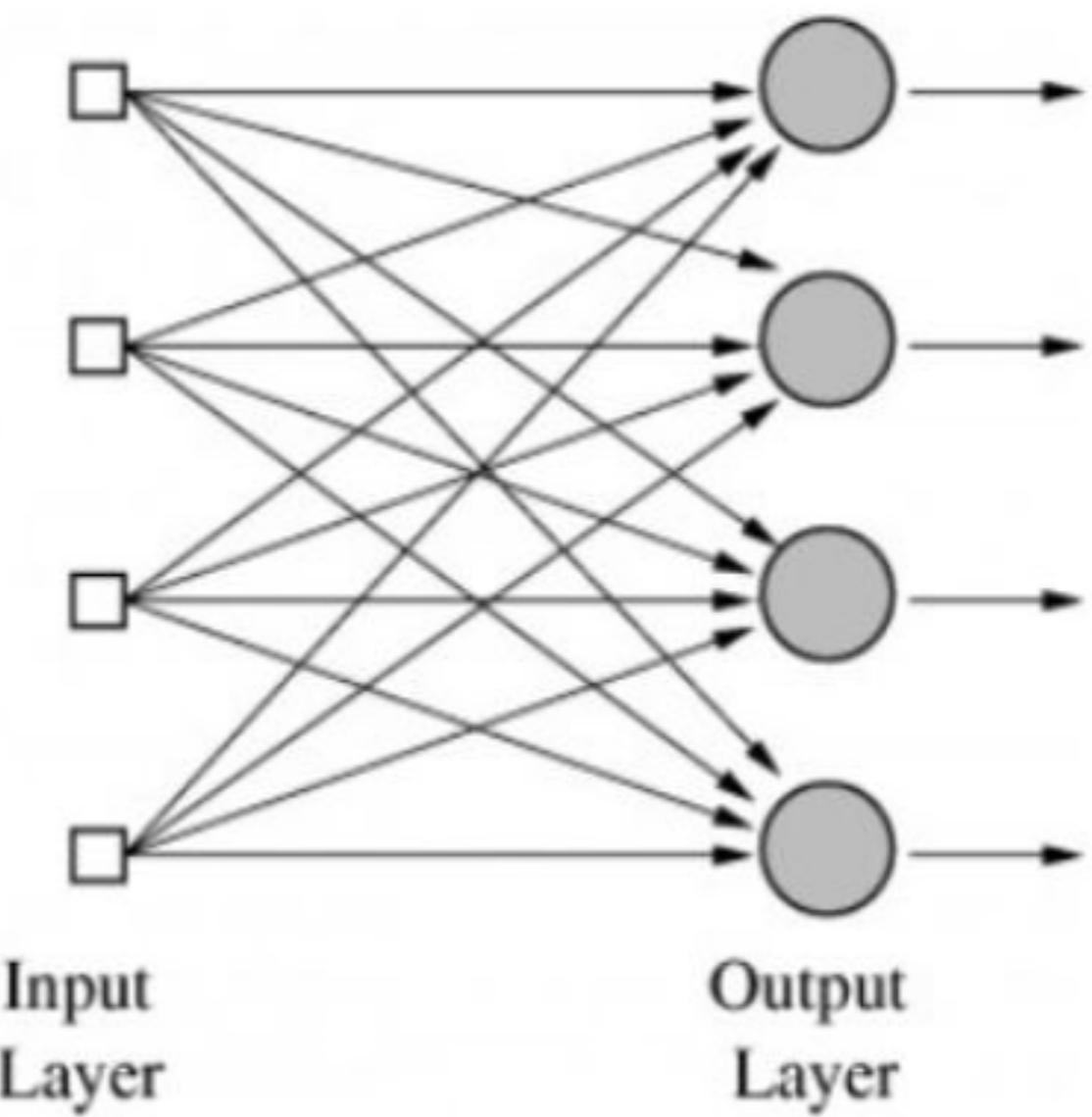
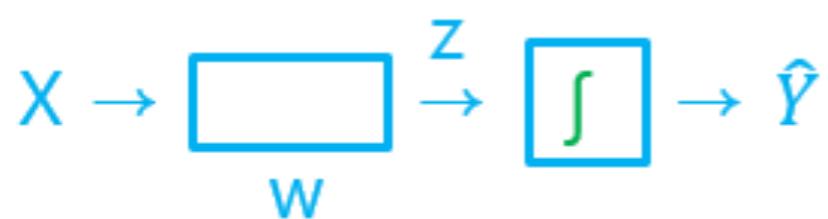


Schematic of a biological neuron.

## ACTIVATION FUNCTIONS



## LOGISTIC REGRESSION UNITS



### XOR PROBLEM

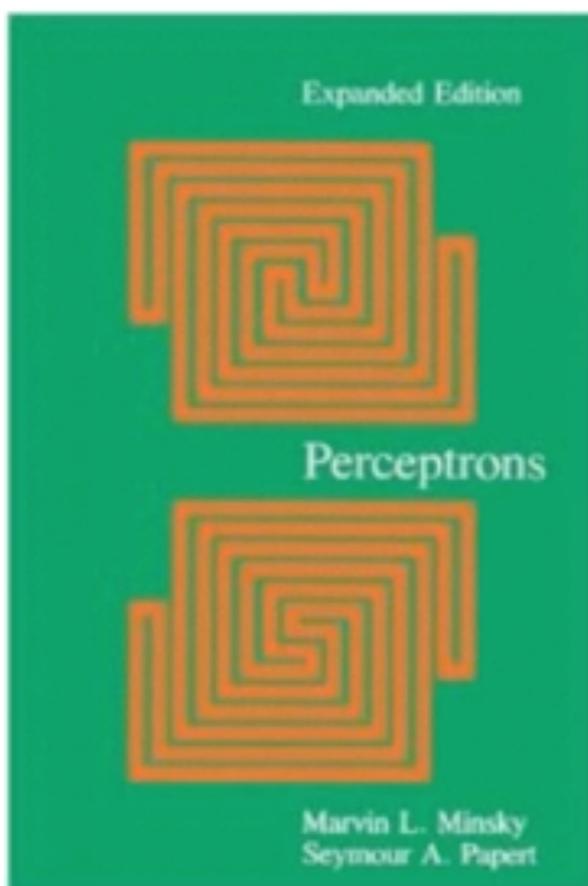
(Simple) XOR problem: linearly separable?



# PERCEPTRONS

## Perceptrons (1969)

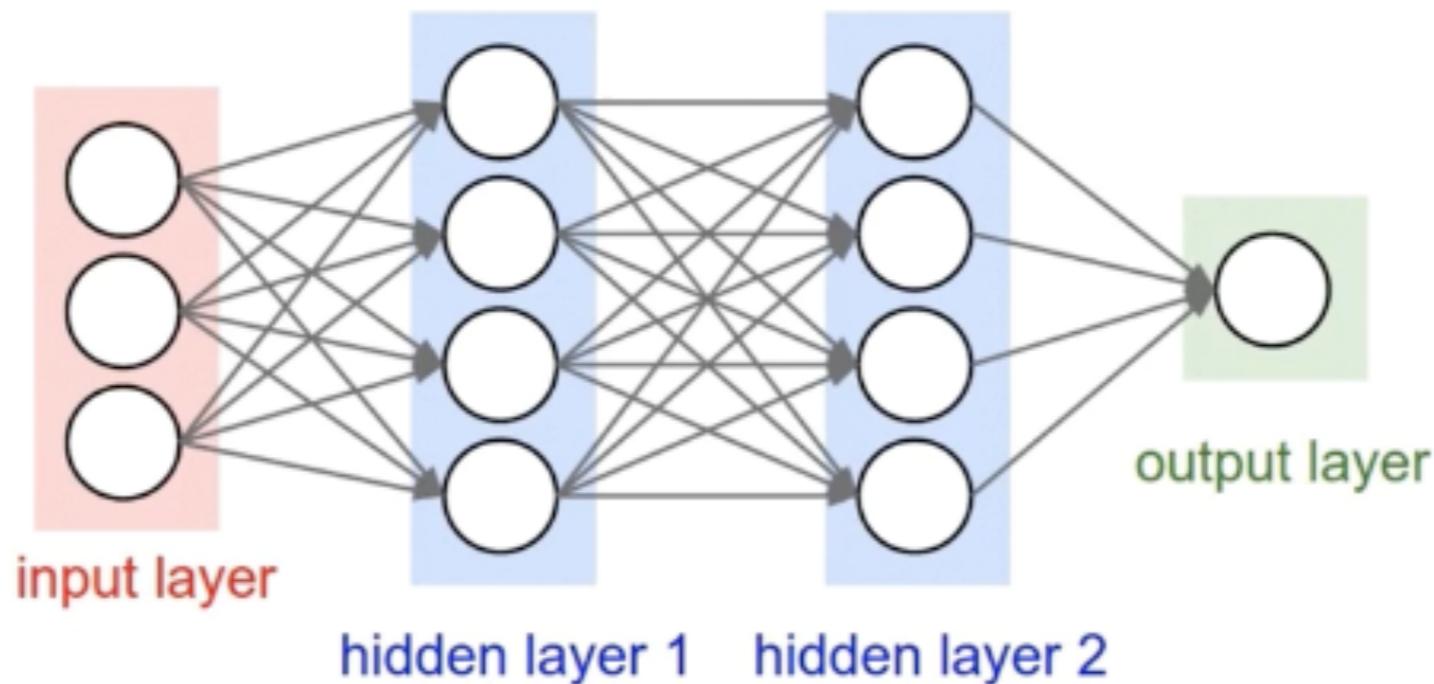
by Marvin Minsky, founder of the MIT AI Lab



- We need to use MLP, multilayer perceptrons (multilayer neural nets)
- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.

# MARVIN MINSKY, 1969

“No one on earth had found a viable way to train\*”

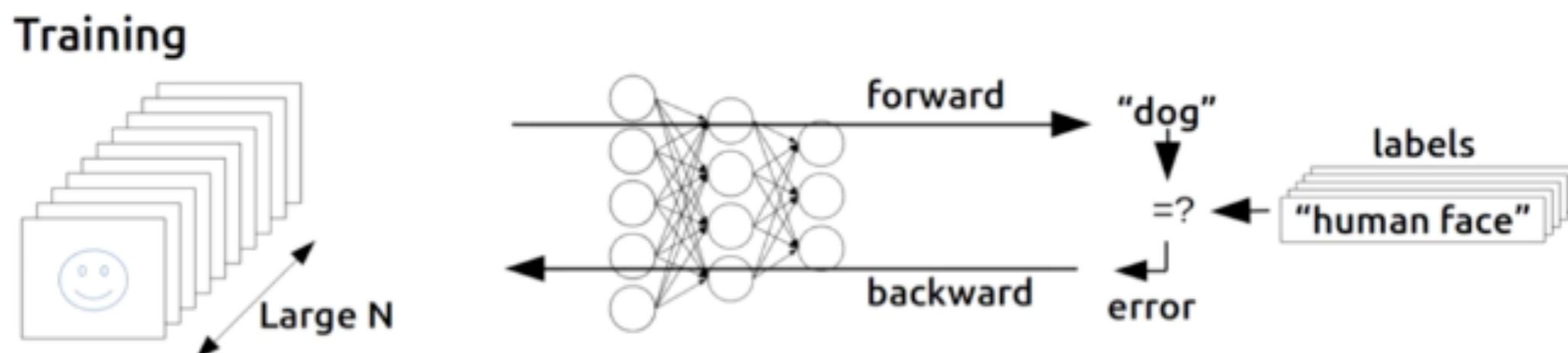


\*Marvin Minsky, 1969

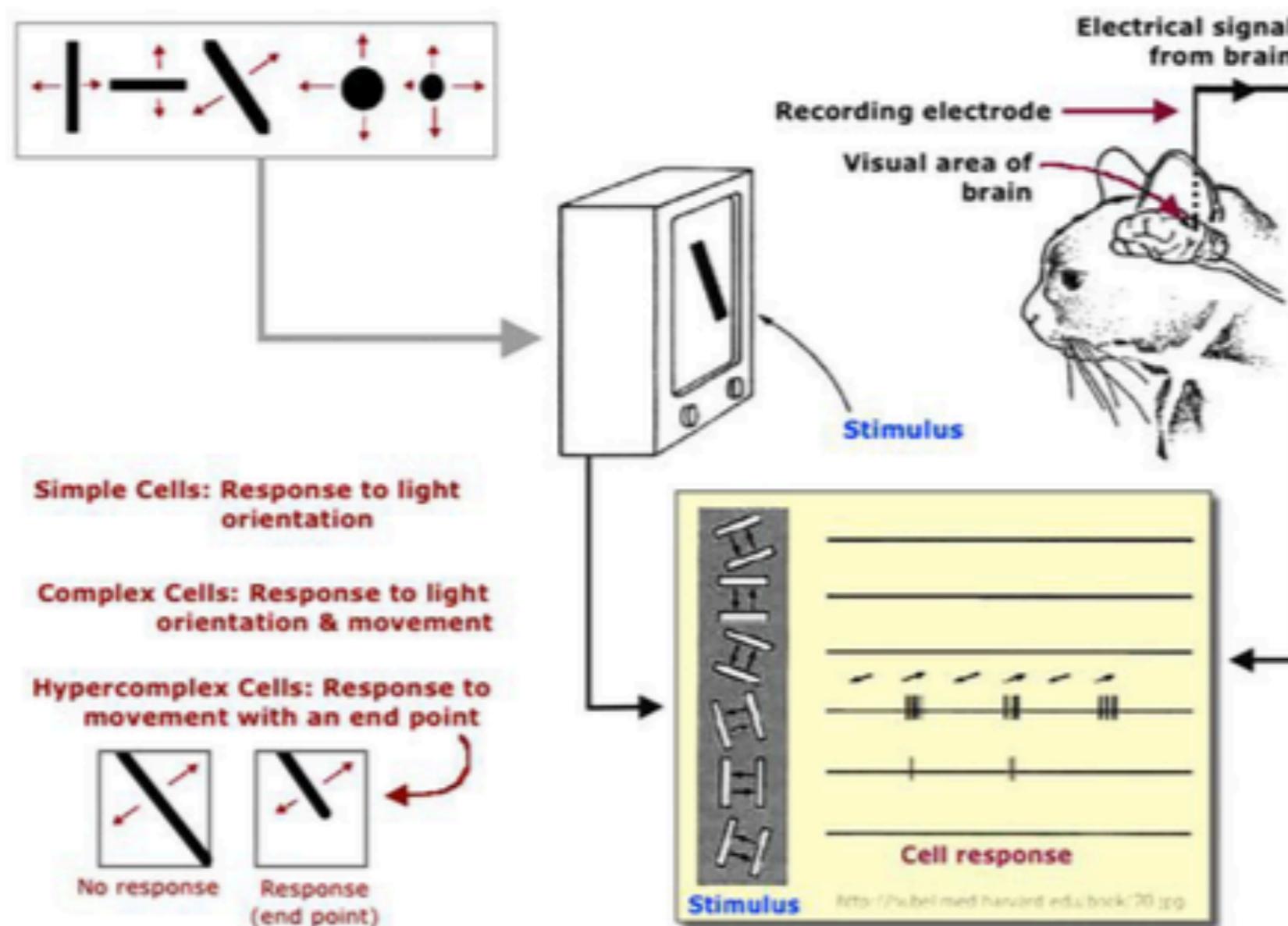
<http://cs231n.github.io/convolutional-networks/>

# BACKPROPAGATION

**Backpropagation**  
(1974, 1982 by Paul Werbos, 1986 by Hinton)

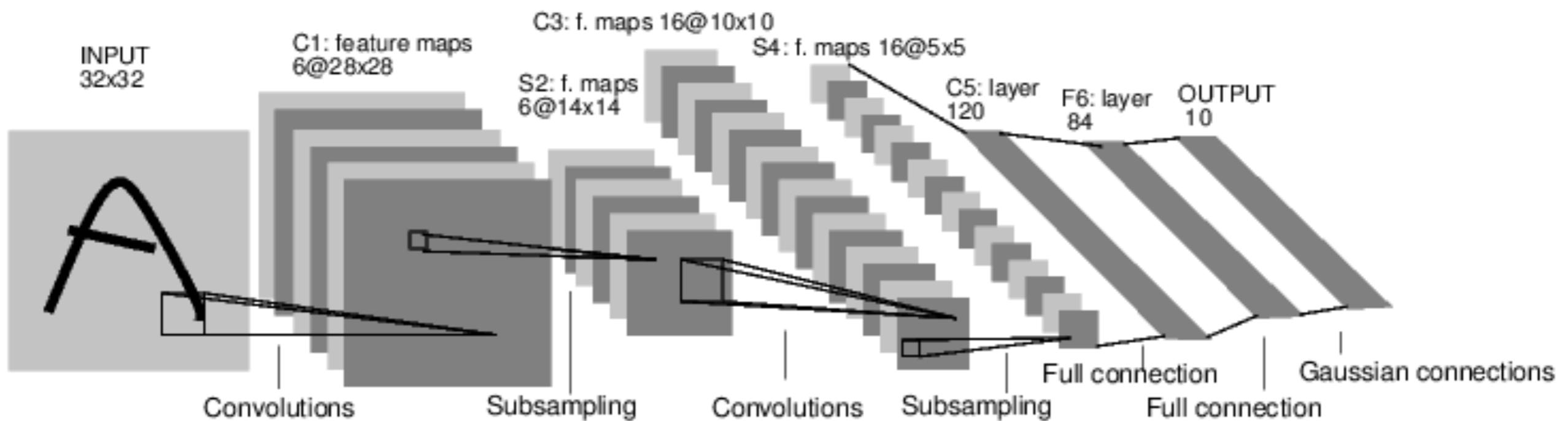


# CONVOLUTIONAL NEURAL NETWORKS



Hubel & Wiesel, 1959

# CONVOLUTIONAL NEURAL NETWORKS : LENET5



“At some point in the late 1990s, one of these systems was reading 10 to 20% of all the checks in the US.”

[LeNet-5, LeCun 1980]

## AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK



# TERMINATOR 2 (1991)

## Terminator 2 (1991)



**JOHN:** Can you learn? So you can be... you know. More human. Not such a dork all the time.

**TERMINATOR:** My CPU is a **neural-net** processor... a learning computer. But **Skynet** presets the switch to "read-only" when we are sent out alone.

... We'll learn how to **set** the neural net

**TERMINATOR** Basically. (starting the engine, backing out) The **Skynet** funding bill is passed. The system goes on-line August 4th, 1997. Human decisions are removed from strategic defense. **Skynet** begins to learn, at a geometric rate. It becomes **self-aware** at 2:14 a.m. eastern time, August 29. In a panic, they try to pull the plug.

**SARAH:** And **Skynet** fights back.

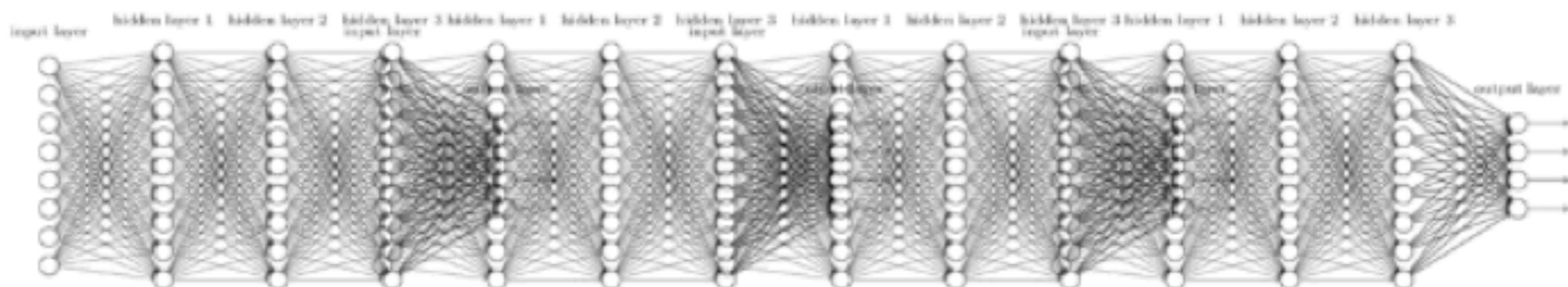
**TERMINATOR:** Yes. It launches its ICBMs against their targets in Russia.

**SARAH:** Why attack Russia?

**TERMINATOR:** Because **Skynet** knows the Russian counter-strike will remove its enemies here.

# A BIG PROBLEM

- Backpropagation just did not work well for normal neural nets with many layers
- Other rising machine learning algorithms: SVM, RandomForest, etc.
- 1995 “Comparison of Learning Algorithms For Handwritten Digit Recognition” by LeCun et al. found that this new approach worked better



### CIFAR

- Canadian Institute for Advanced Research (CIFAR)
- CIFAR encourages basic research without direct application, was what motivated Hinton to move to Canada in 1987, and funded his work afterward.



**CIFAR**  
CANADIAN INSTITUTE  
for ADVANCED RESEARCH

<http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-4/>

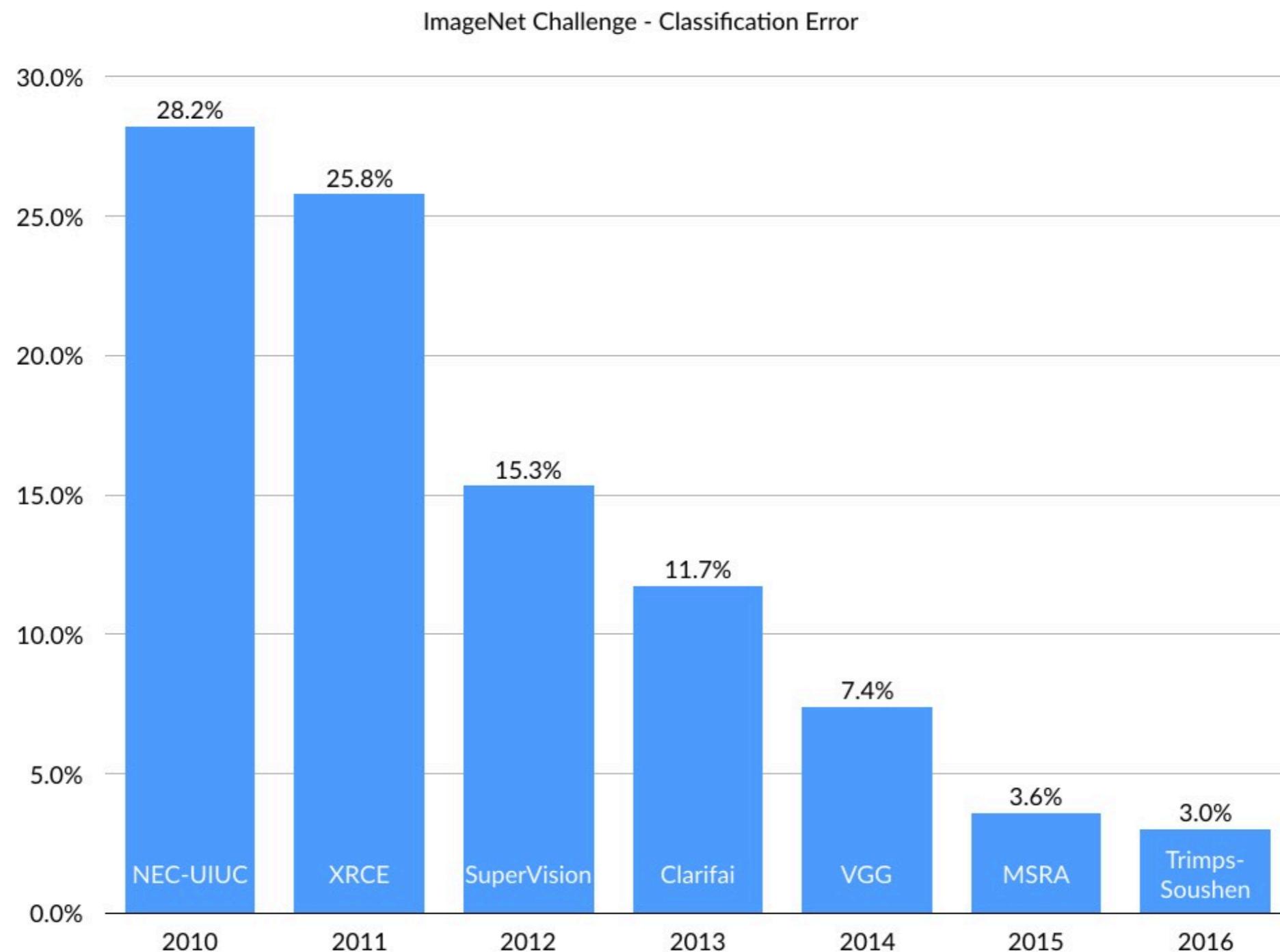
# BREAKTHROUGH – HINTON(2006), BENGIO(2007)

- In 2006, Hinton, Simon Osindero, and Yee-Whye Teh published, “A fast learning algorithm for deep belief nets”
- Yoshua Bengio et al. in 2007 with “Greedy Layer-Wise Training of Deep Networks”

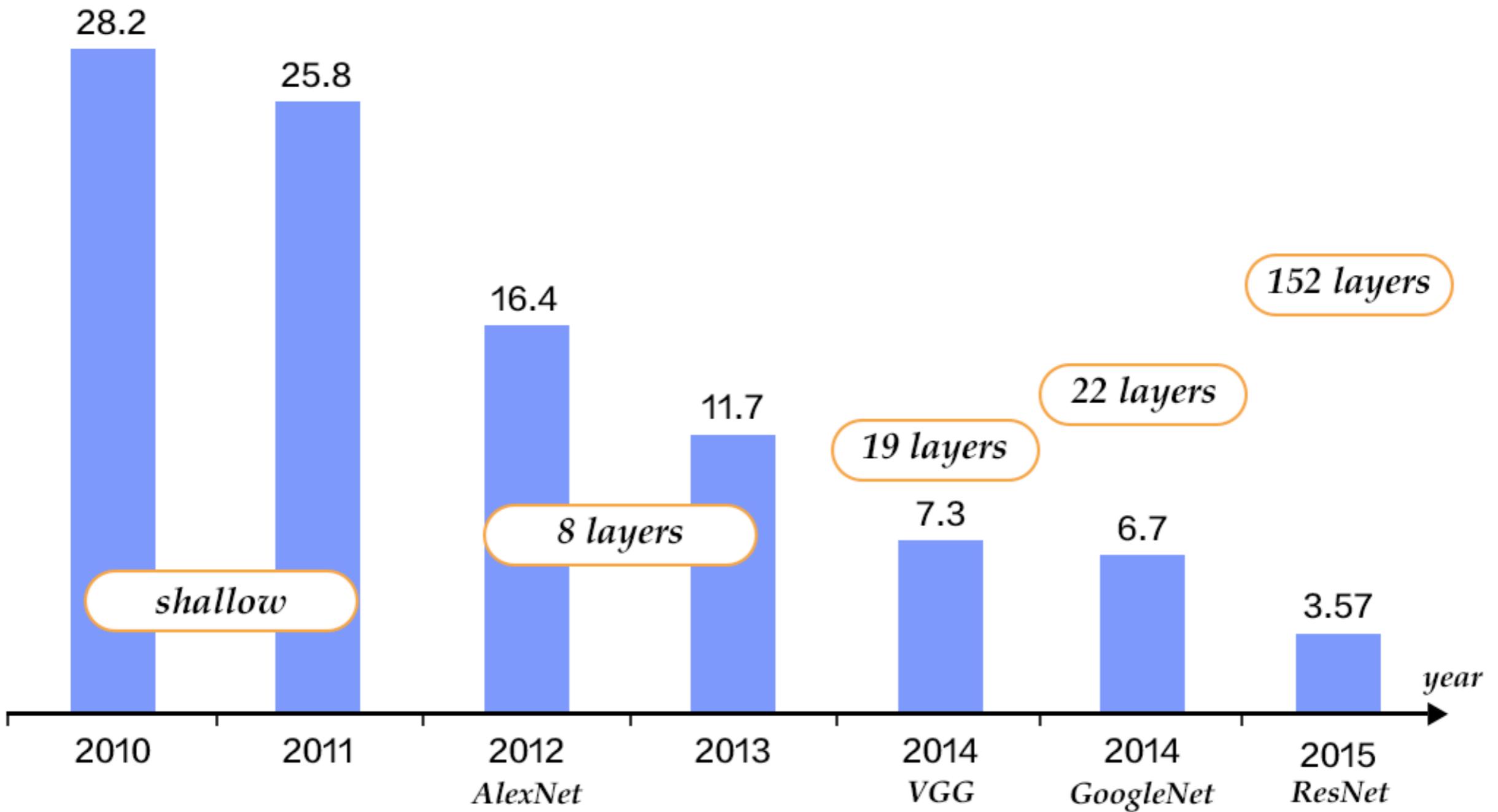
### BREAKTHROUGH – HINTON(2006), BENGIO(2007)

- Neural networks with many layers really could be trained well, if the weights are initialized in a clever way rather than randomly.
- Deep machine learning methods are more efficient for difficult problems than shallow methods.
- Rebranding to Deep Nets, Deep Learning

# IMAGENET CLASSIFICATION



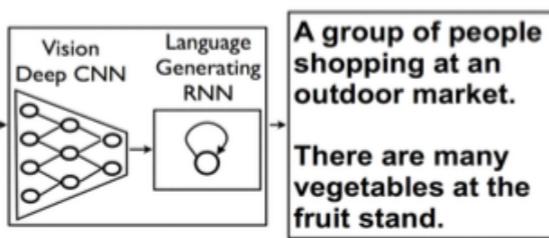
# IMAGENET CLASSIFICATION



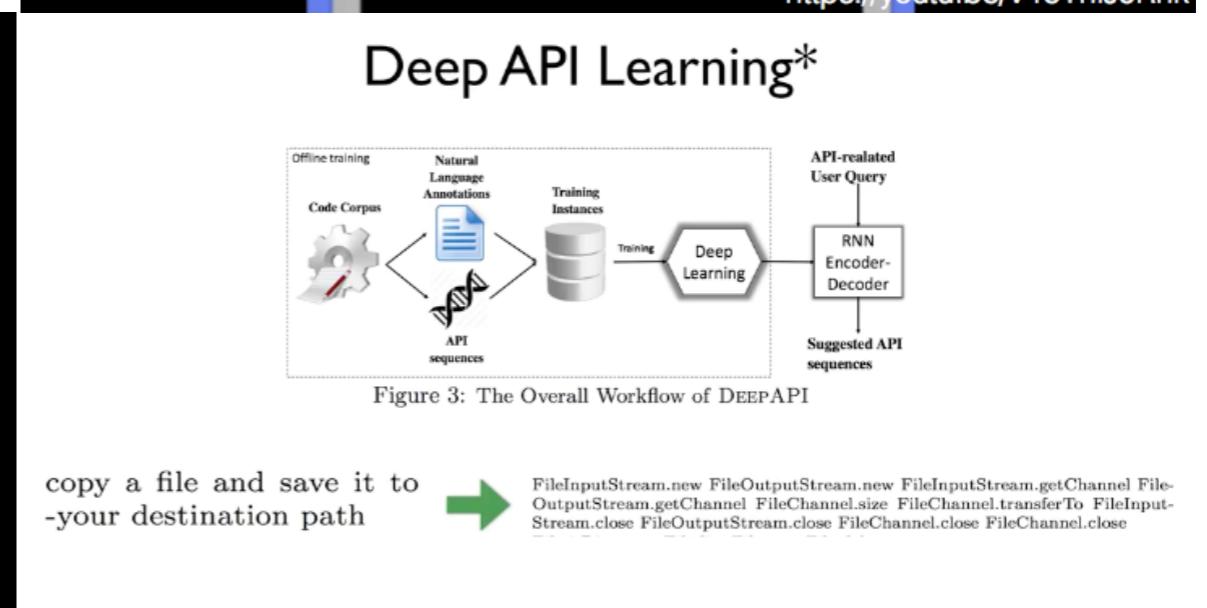
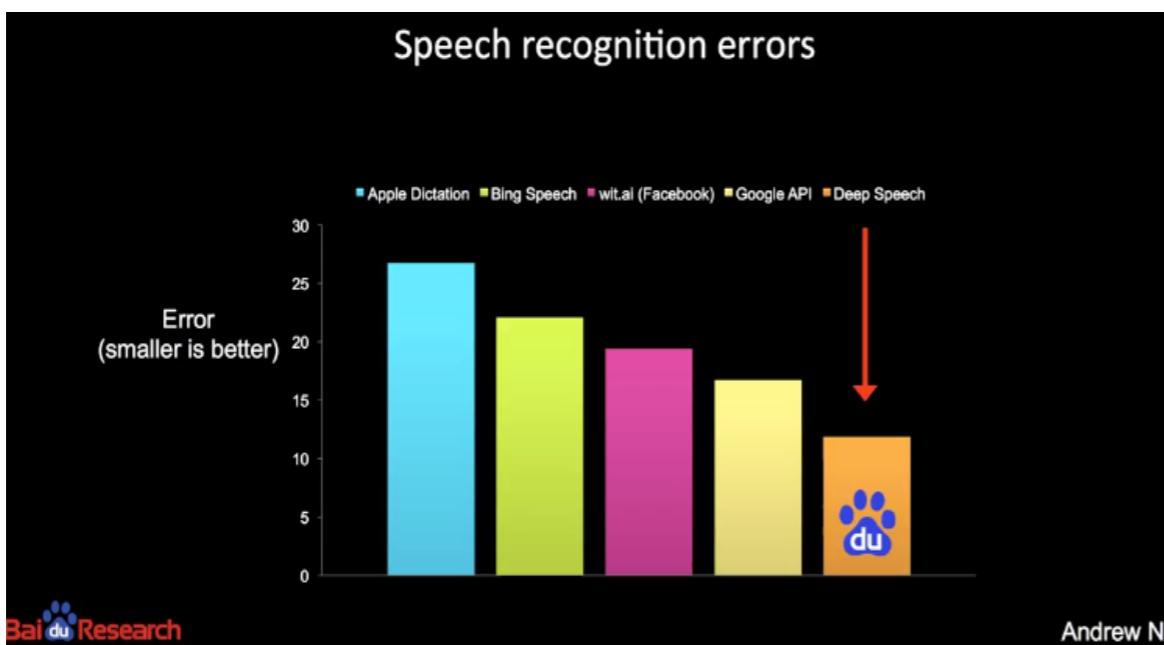
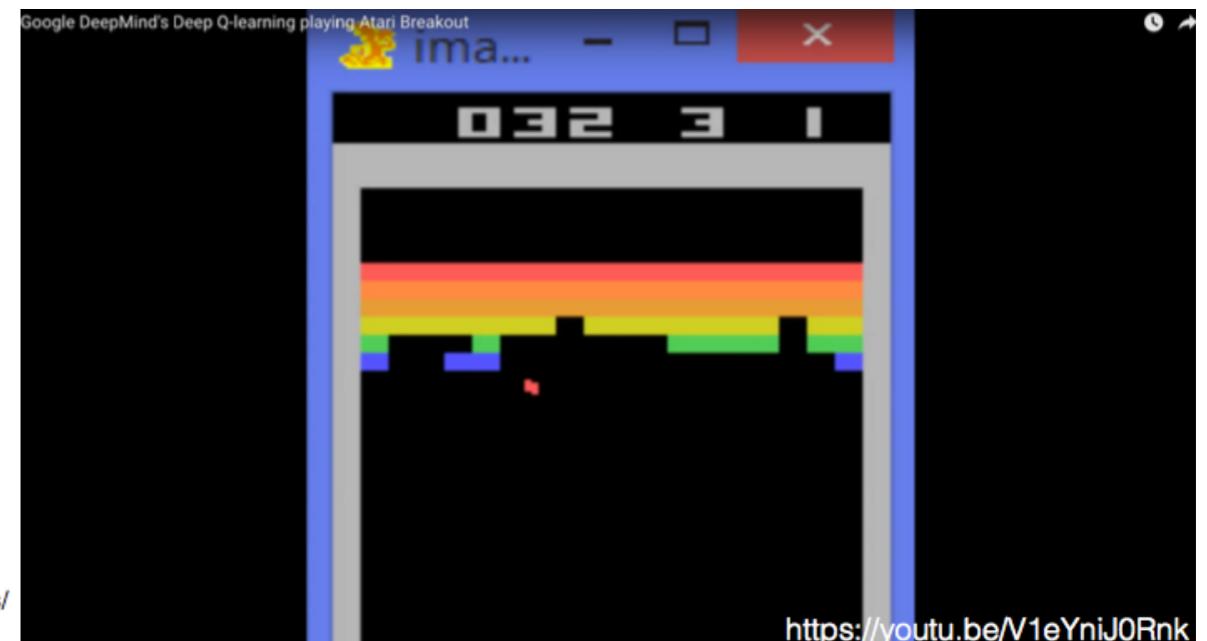
# NEURAL NETWORKS HISTORY

## AREA OF DEEP LEARNING

### Neural networks that can explain photos



<https://gigaom.com/2014/11/18/google-stanford-build-hybrid-neural-networks-that-can-explain-photos/>



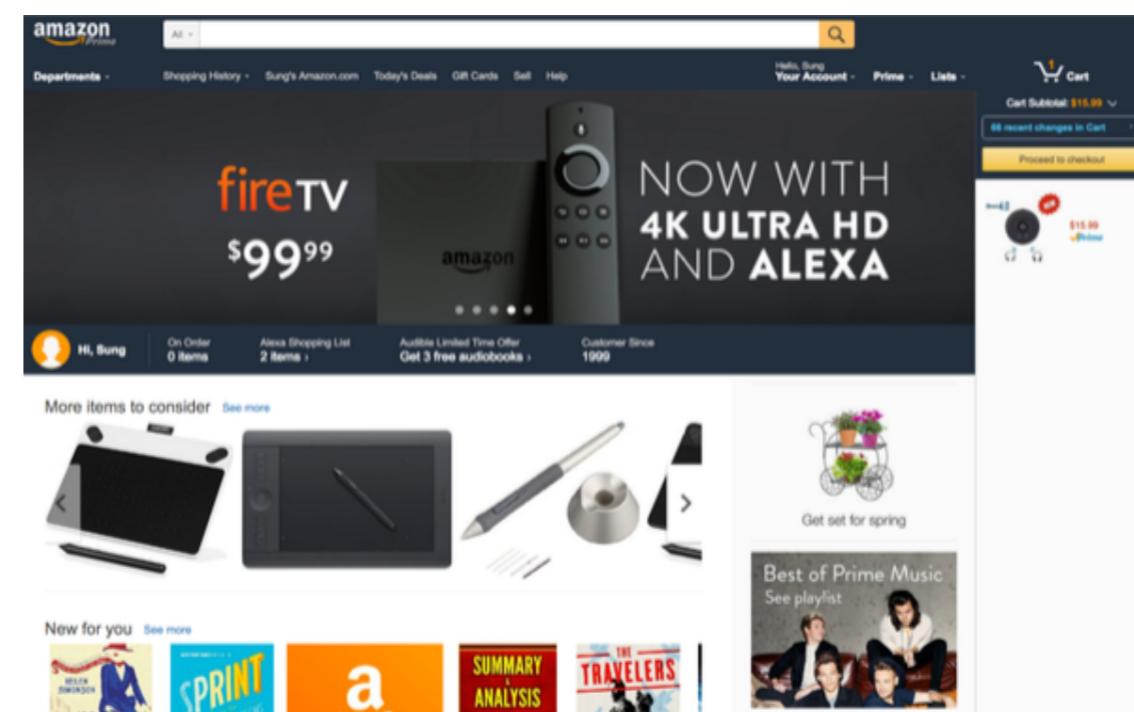
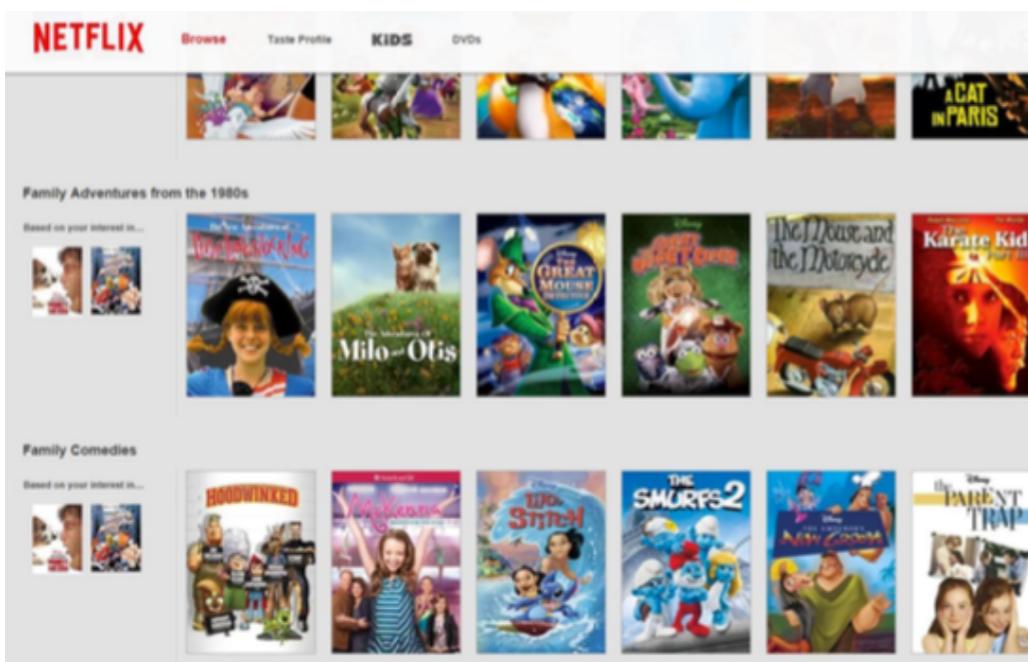
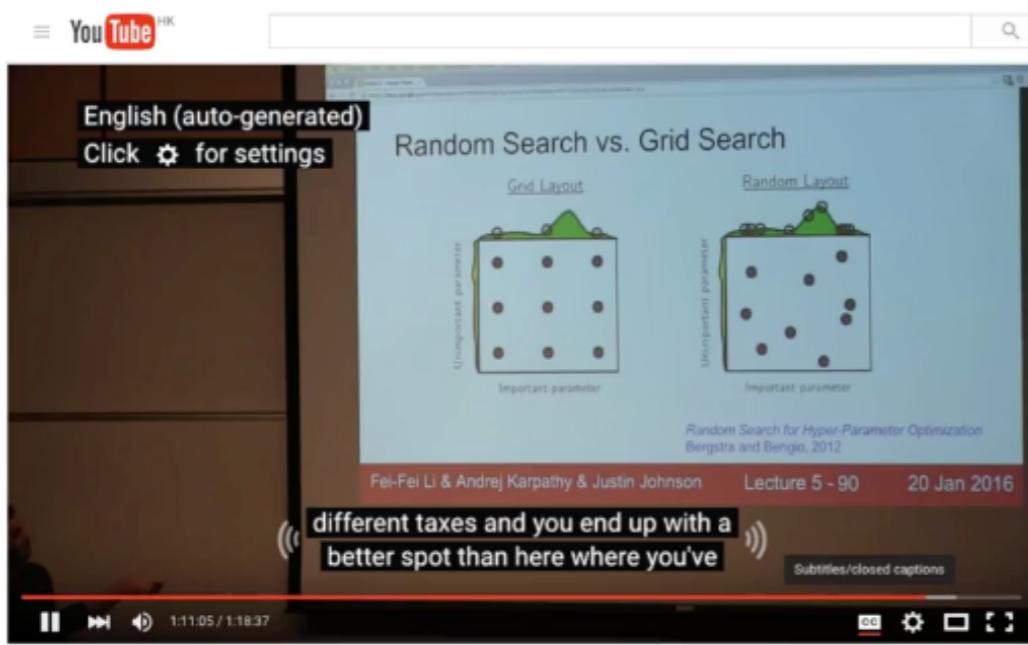
\*GU et al. at HKUST with MSRA

# ALPHAGO



# NEURAL NETWORKS HISTORY

## DEEP LEARNING FOR BUSINESS



### WHY SHOULD I CARE?

- *I am not a researcher, not a computer scientist!*
- Do you have data?
- Do you sell something?
- Are doing any business?

### WHY NOW?

- Students/Researchers
  - Not too late to be a world expert
  - Not too complicated (mathematically)
- Practitioner
  - Accurate enough to be used in practice
  - many ready-to-use tools such as TensorFlow
  - Many easy/simple programming languages such as Python
- After all, it is fun!



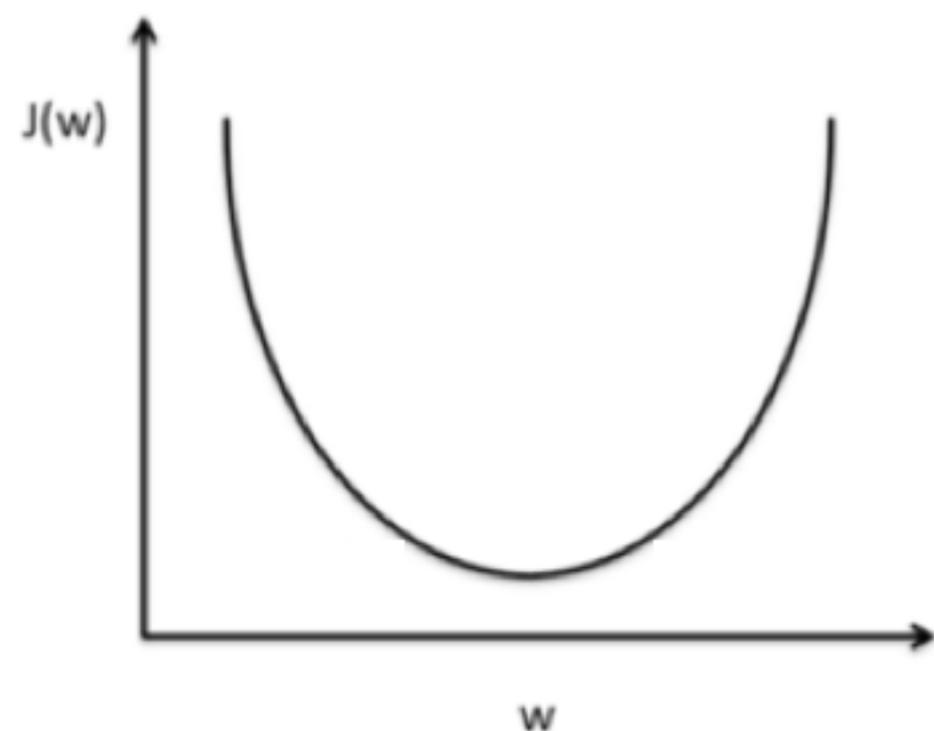
DEEP LEARNING

---

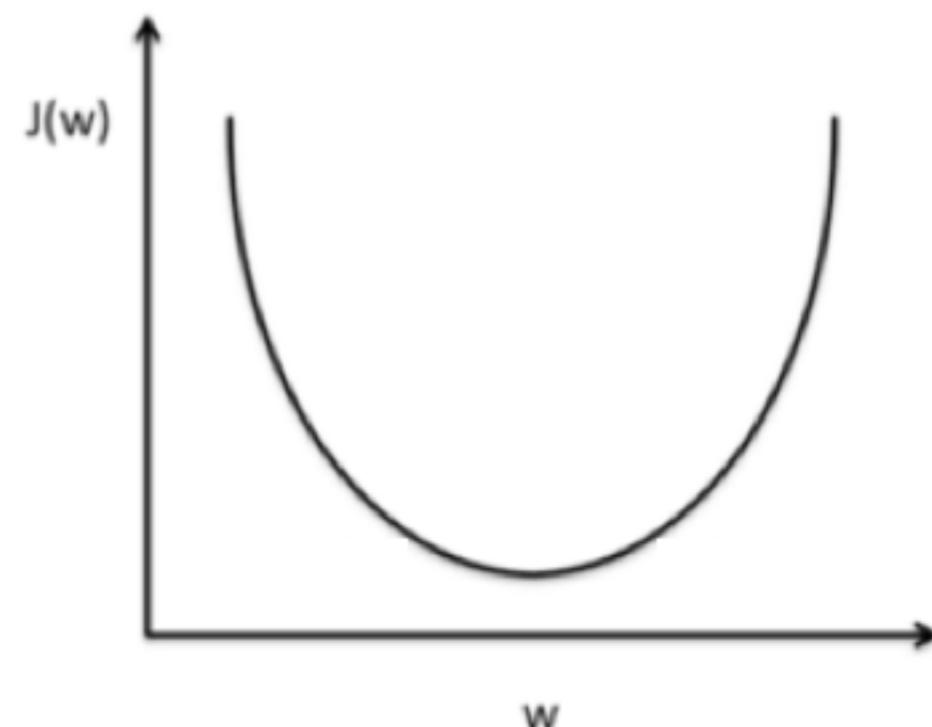
APPLICATION  
AND TIPS

# LEARNING RATE

**Large learning rate: overshooting**

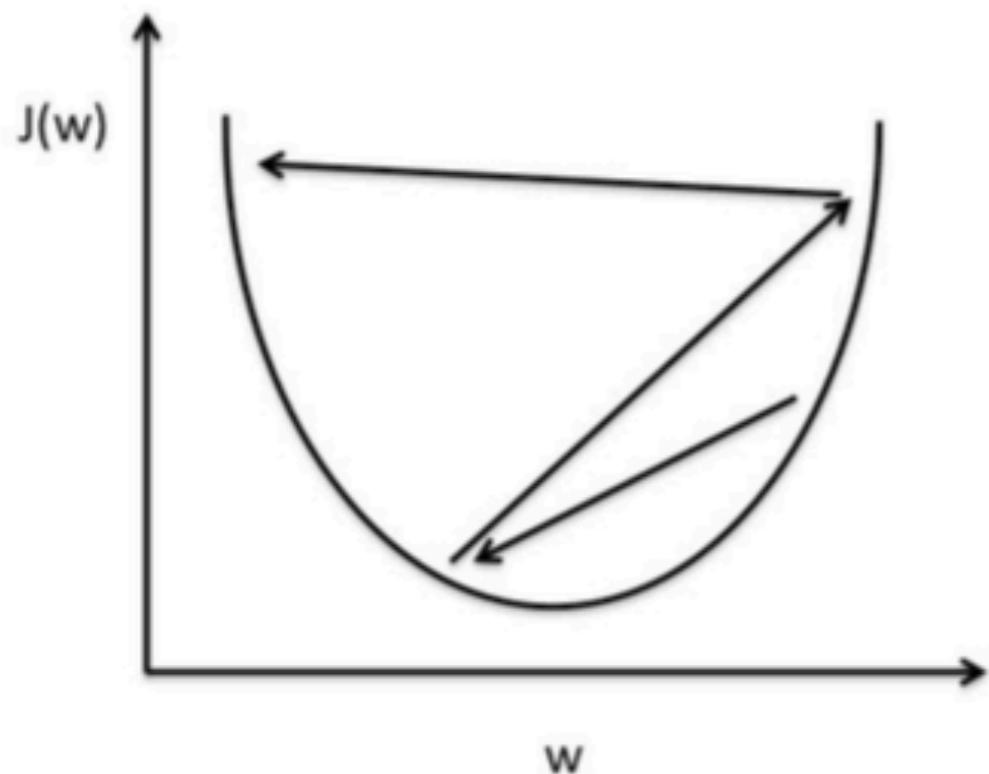


**Small learning rate:**  
takes too long, stops at local minimum

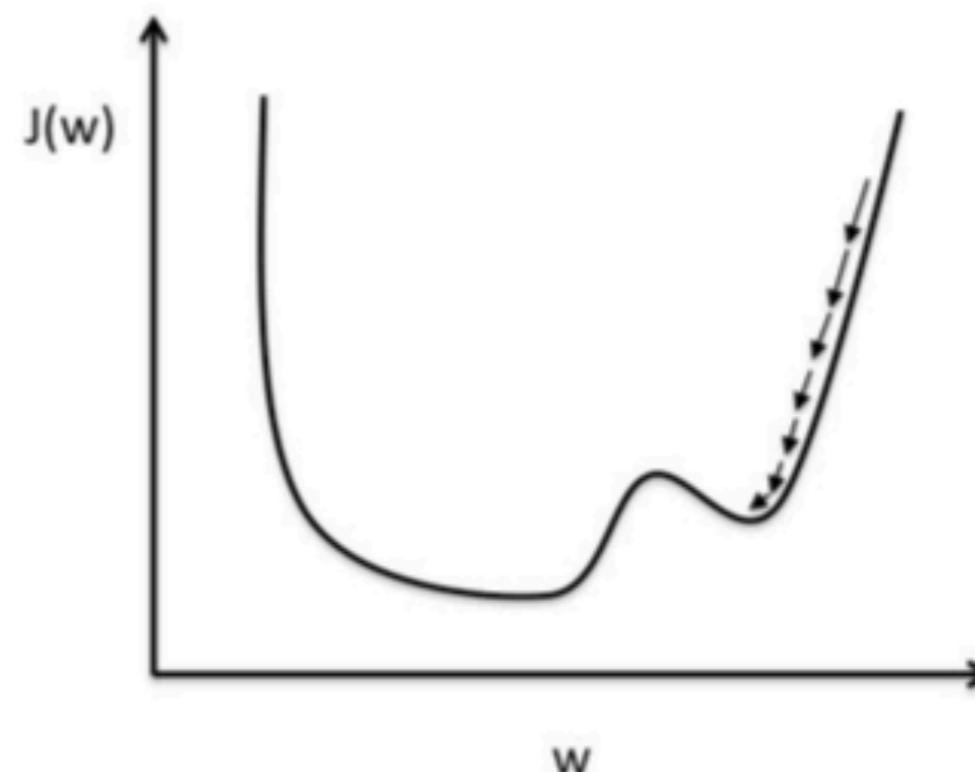


## LEARNING RATE

Learning rate: **NaN!**



Large learning rate: Overshooting.



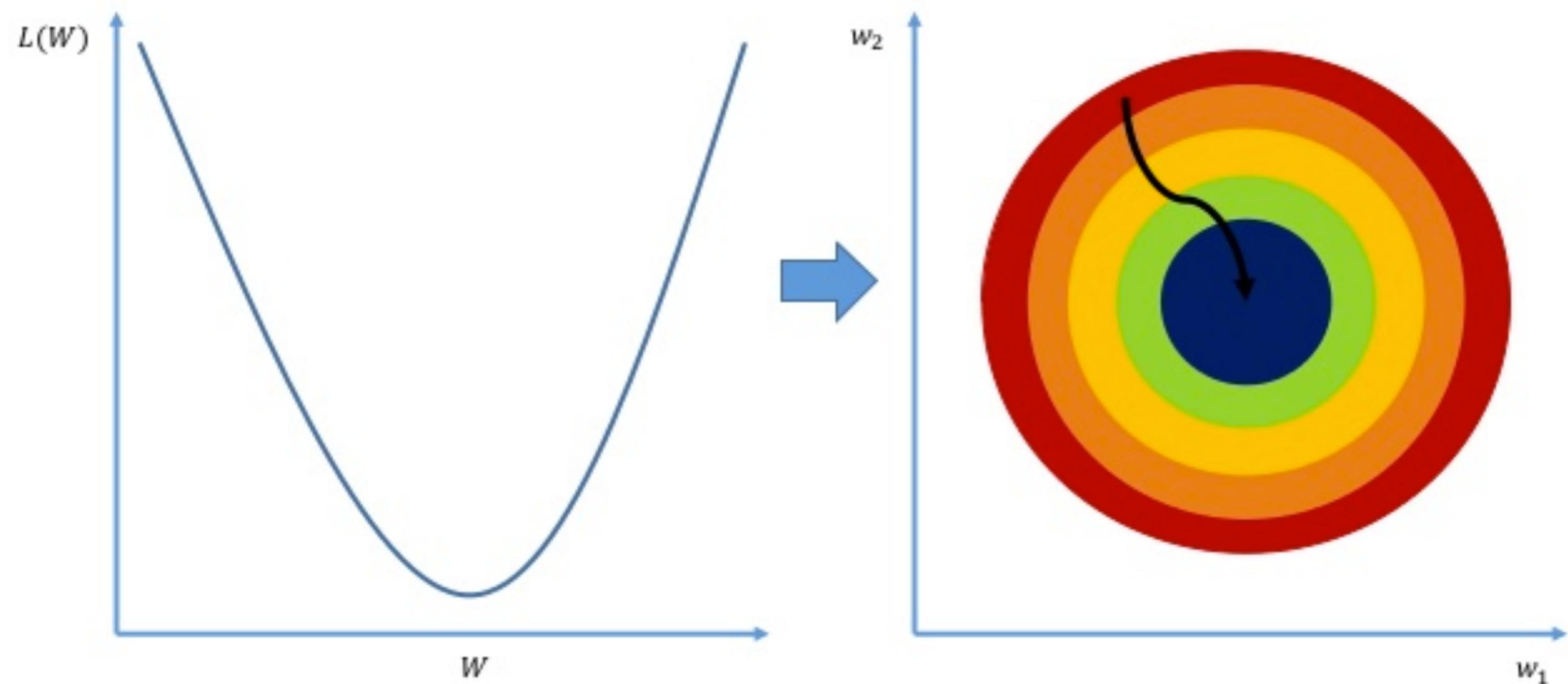
Small learning rate: Many iterations until convergence and trapping in local minima.

## FIND LEARNING RATE

Try several learning rates

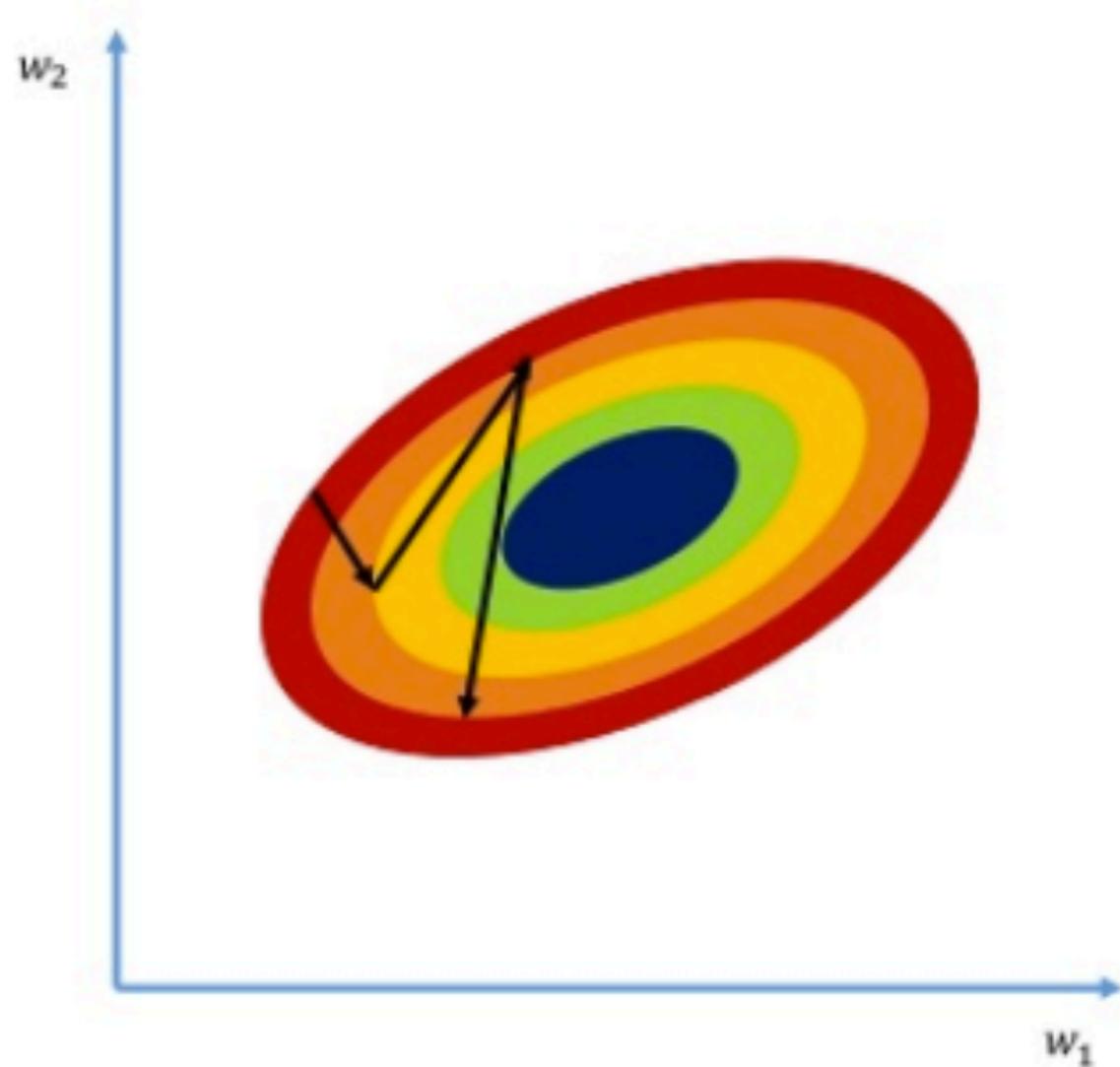
- Observe the cost function
- Check it goes down in a reasonable rate

## DATA PREPROCESSING FOR GRADIENT DESCENT



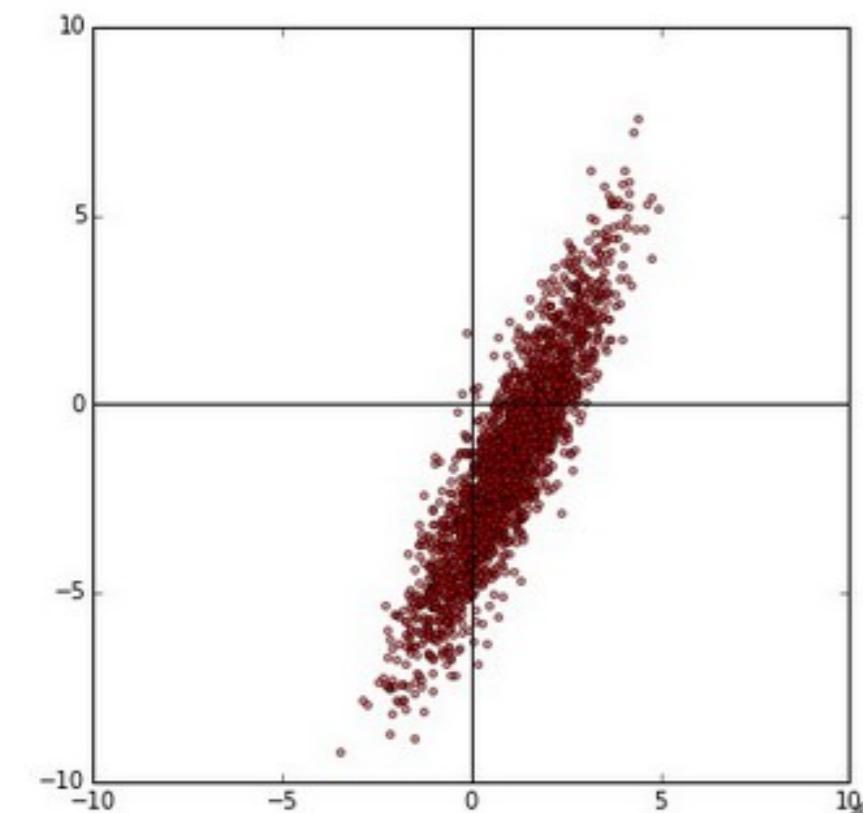
# DATA PREPROCESSING FOR GRADIENT DESCENT

x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C

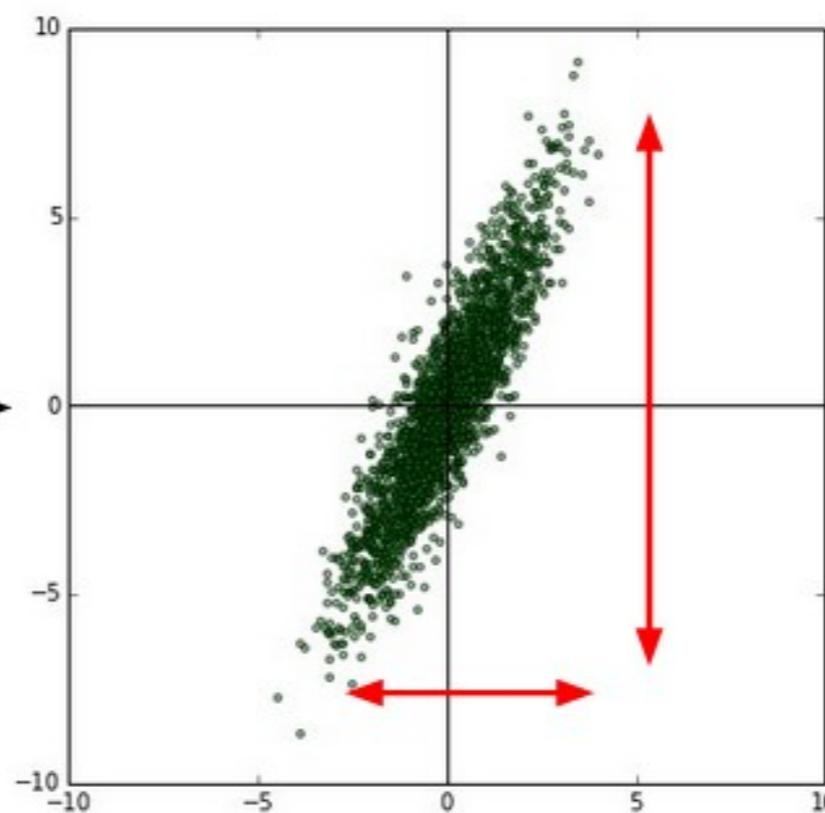


# PREPROCESSING FOR GRADIENT DESCENT

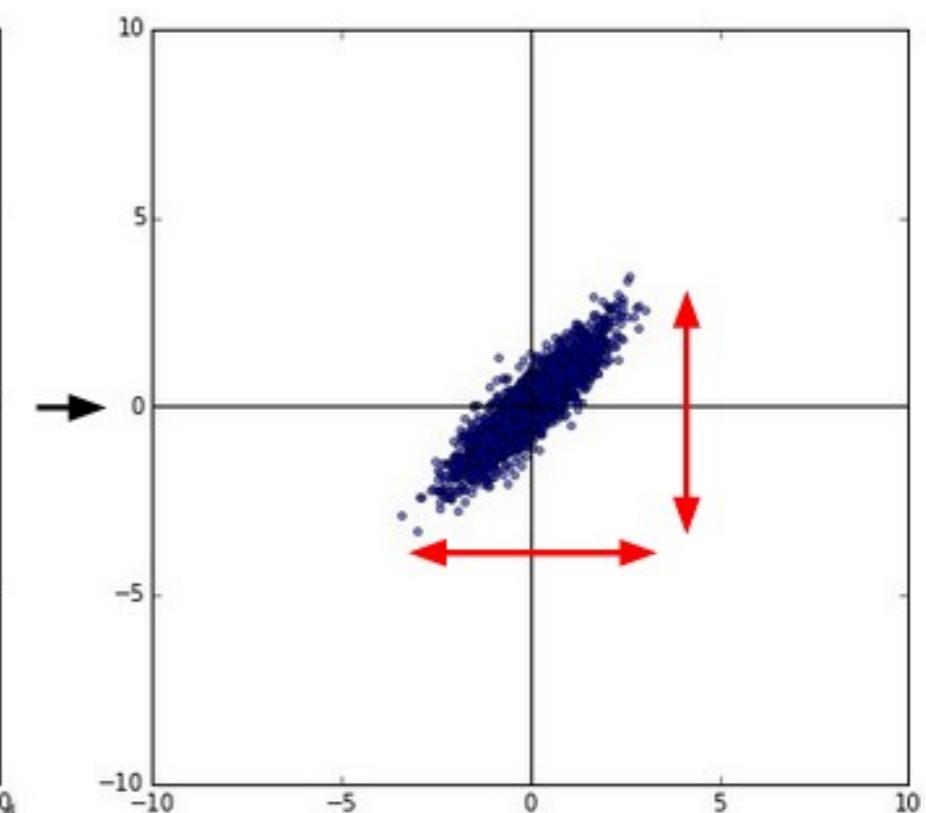
original data



zero-centered data



normalized data



# NORMALIZATION AND STANDARDIZATION

Normalization은 "해당 속성(변수)값-최소값/최대값-최소값"으로 0~1사이의 값으로 나타내는 척도법이고

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization은 특정한 분포(ex. 정규분포)들의 평균과 분산 혹은 표준편차를 이용해 "속성값-평균/표준편차"로 해당 분포에서의 이 속성값이 평균으로부터의 위치를 표준편차 단위로 옮겨서 다시 나타낸 것이다.

$$x_{new} = \frac{x - \mu}{\sigma}$$

# NORMALIZATION AND STANDARDIZATION

### ▶ Normalization

수식 :  $(\text{요소값} - \text{최소값}) / (\text{최대값} - \text{최소값})$

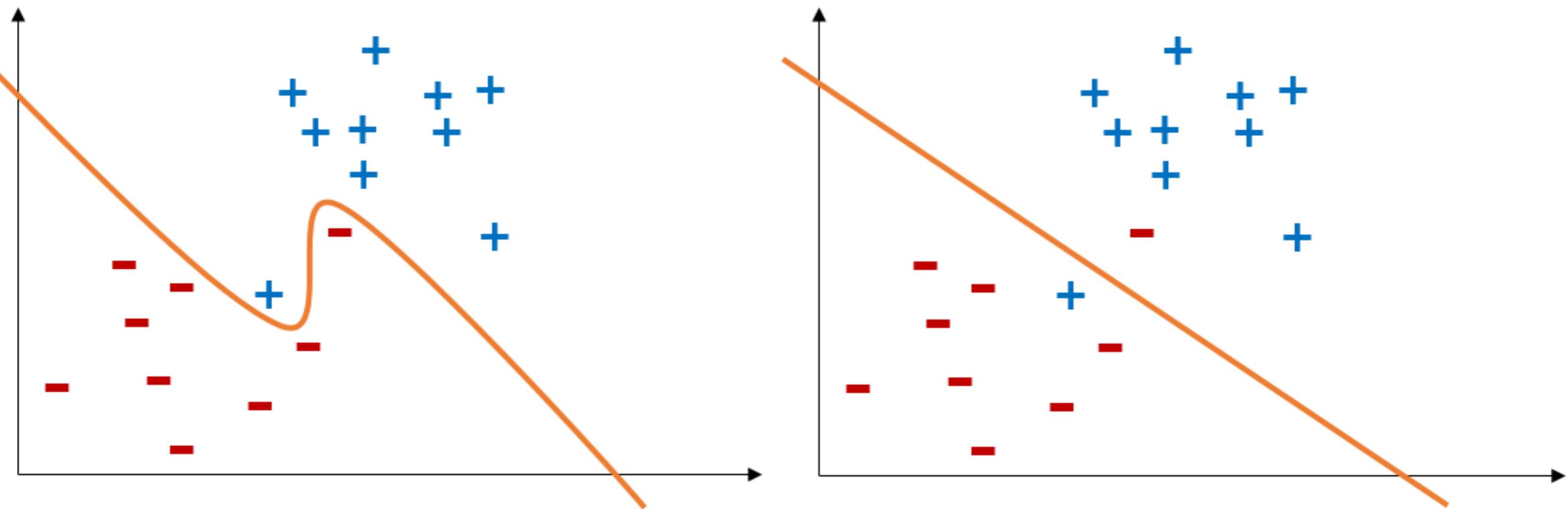
설명 : 전체 구간을 0~100으로 설정하여 데이터를 관찰하는 방법으로,  
특정 데이터의 위치를 확인할 수 있게 해줌

### ▶ Standardization

수식 :  $(\text{요소값} - \text{평균}) / \text{표준편차}$

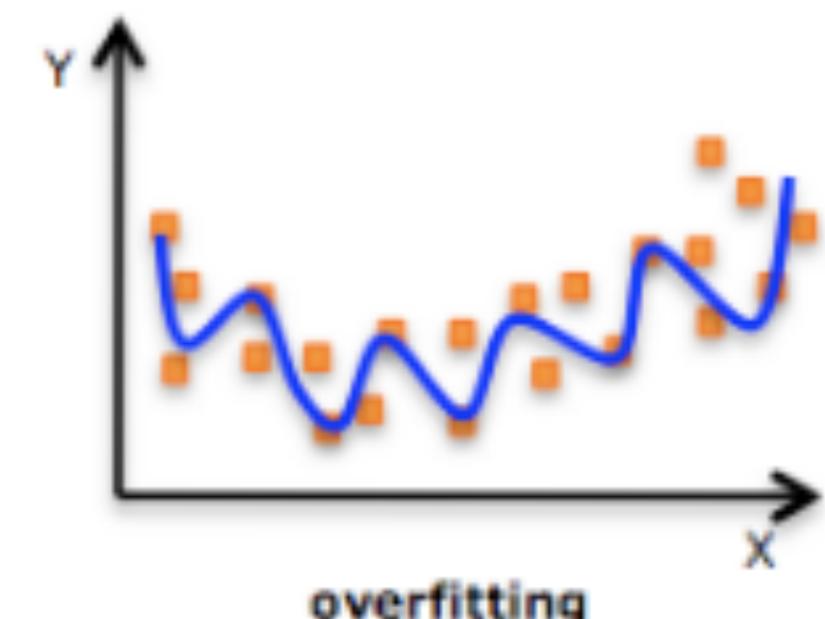
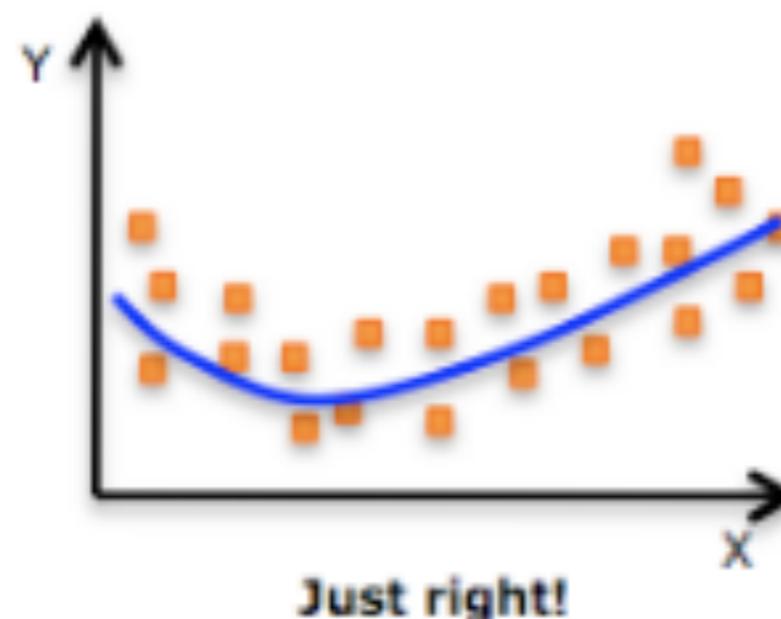
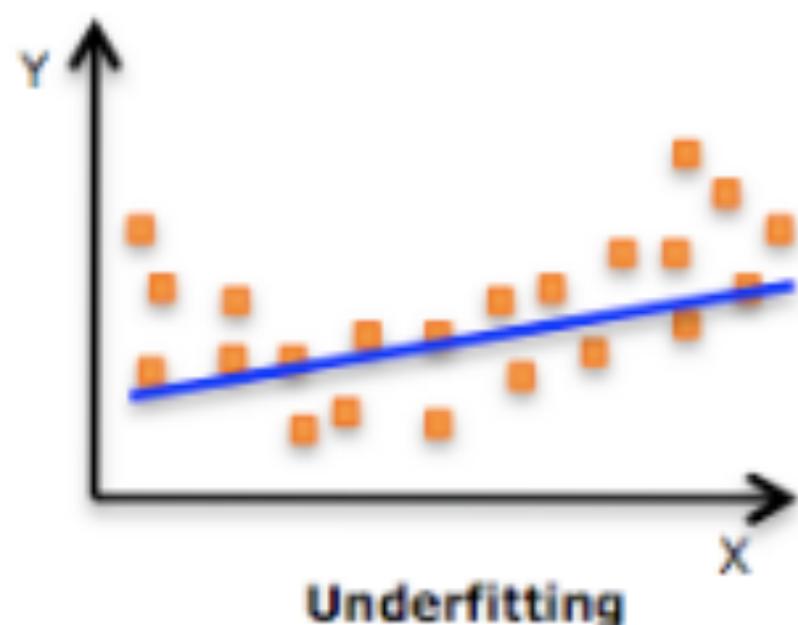
설명 : 평균까지의 거리로, 2개 이상의 대상이 단위가 다를 때,  
대상 데이터를 같은 기준으로 볼 수 있게 해줌

## OVERFITTING



- Our model is very good with training data set (with memorization)
- Not good at test dataset or in real use

## UNDERFITTING VS. OVERFITTING



## SOLUTIONS FOR OVERFITTING

- ▶ **More training data**
- ▶ **Reduce the number of features**
- ▶ **Regularization**
- ▶ **Early stopping**
- ▶ **Dropout**

## REGULARIZATION

- Let's not have too big numbers in the weight

```
l2reg = 0.001 * tf.reduce_sum(tf.square(W))
```

$$L = \frac{1}{N} \sum_i D(S(WX_i + b), L_i) + \lambda \sum W^2$$

Diagram illustrating the components of the loss function:

- LOSS**: Points to the term  $D(S(WX_i + b), L_i)$ .
- TRAINING SET**: Points to the index  $i$  in the summation.
- WEIGHTS**: Points to the term  $\sum W^2$ .

## TEST & VALIDATION

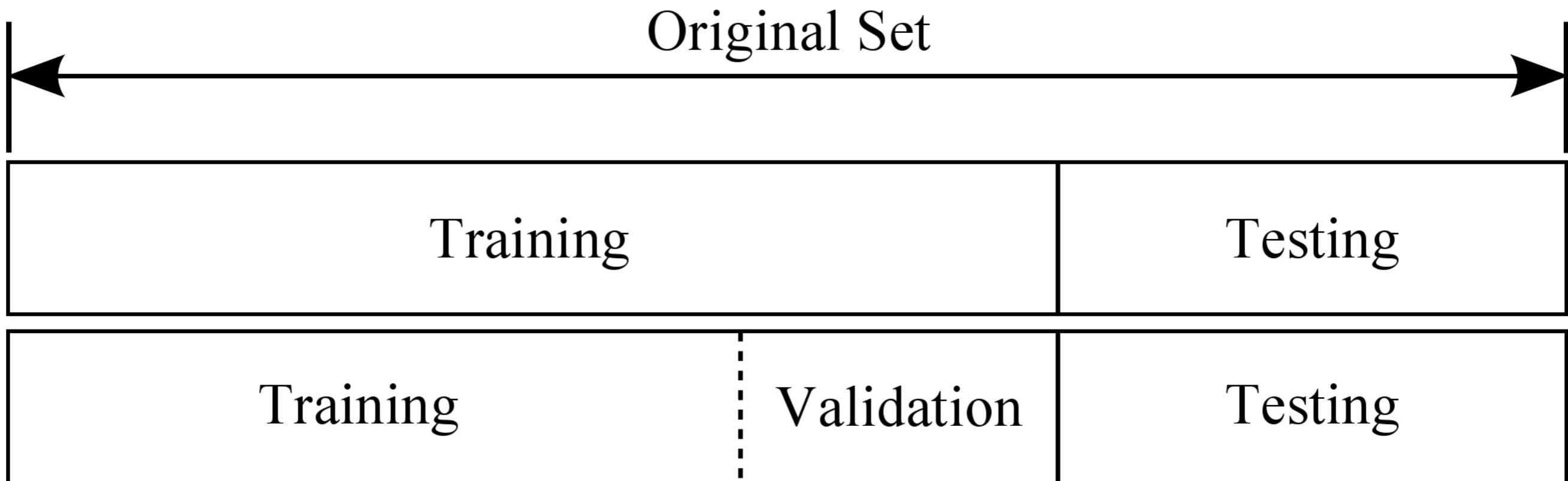
**Performance evaluation: is this good?**

# EVALUATION USING TRAINING SET?

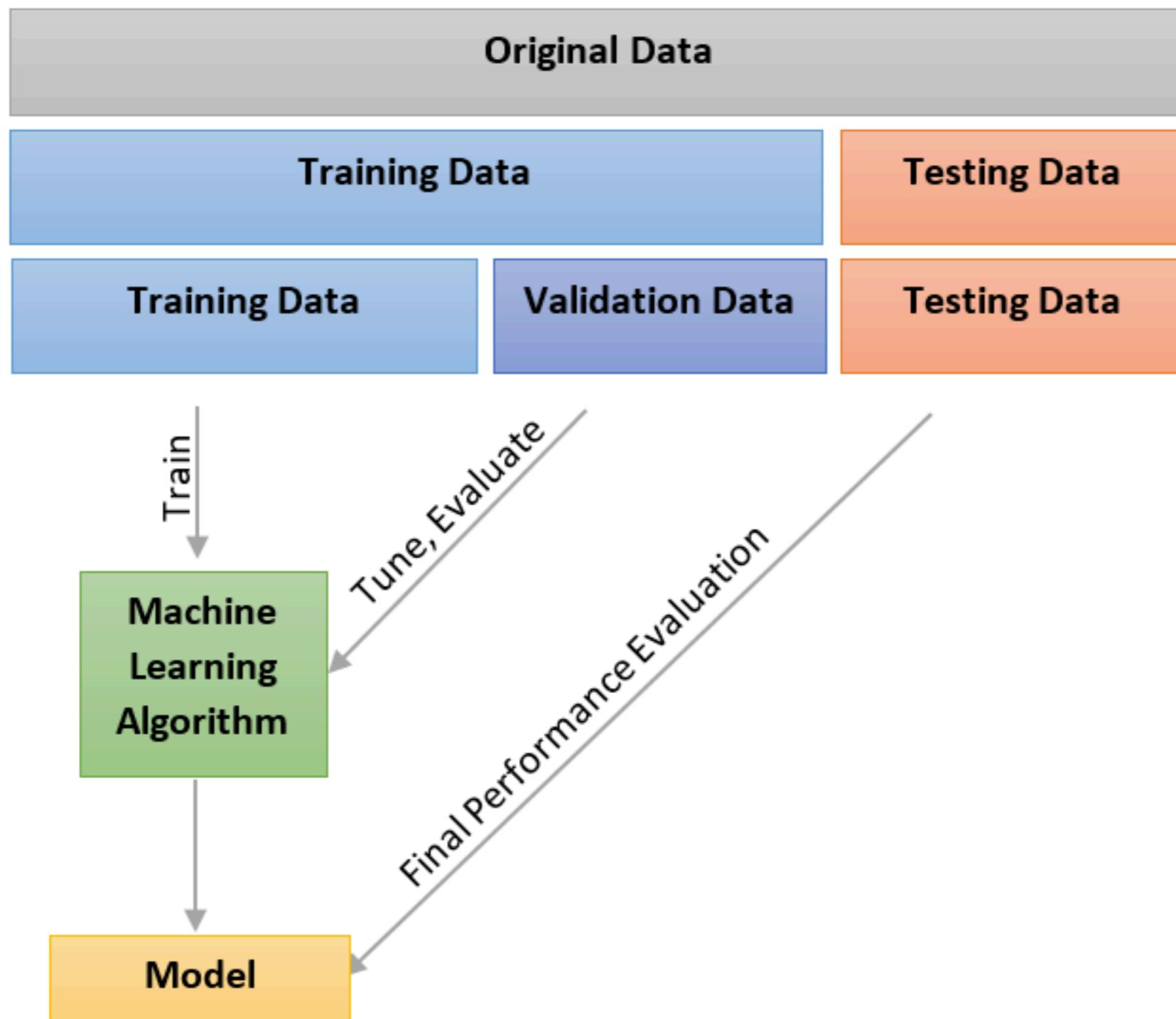
Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

- 100% correct (accuracy)
- Can memorize

## TRAINING, VALIDATION AND TEST SETS



# TRAINING, VALIDATION AND TEST SETS



## TRAINING, VALIDATION AND TEST SETS

- ▶ **Training set**

매개변수 학습

- ▶ **Validation set**

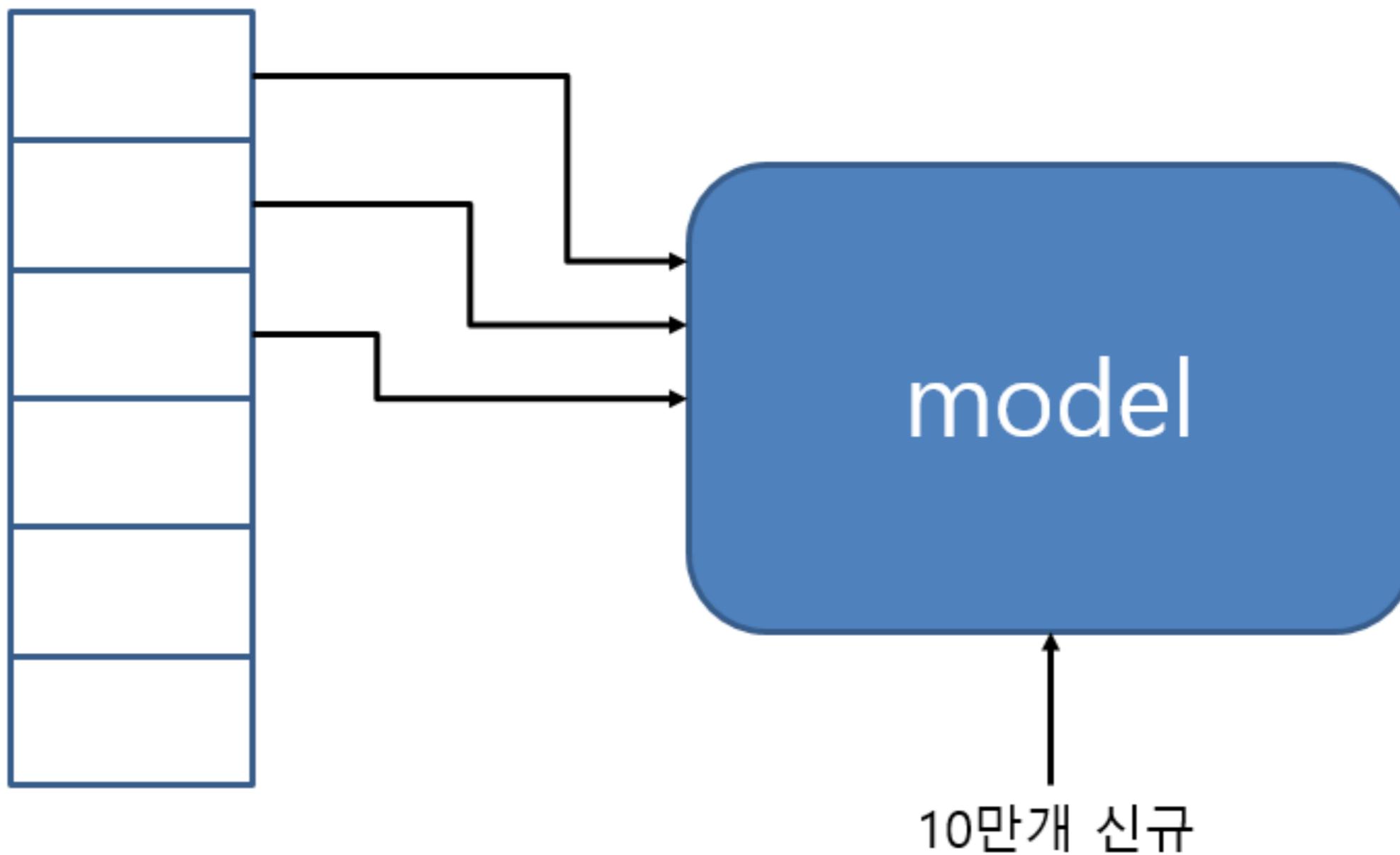
하이퍼파라미터 성능 평가

- ▶ **Test set**

모델 성능 평가

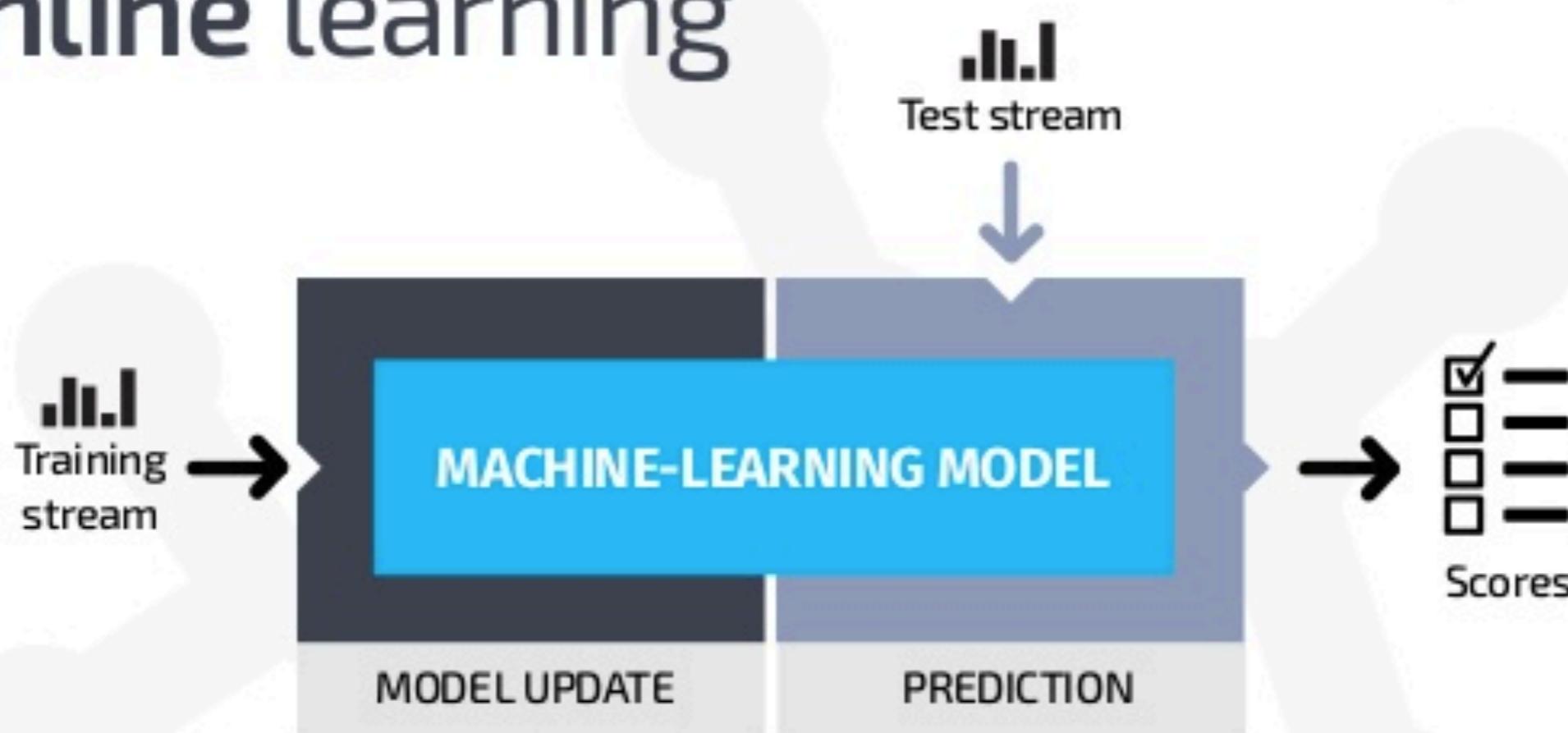
## ONLINE LEARNING

100만개



## ONLINE LEARNING

# Online learning

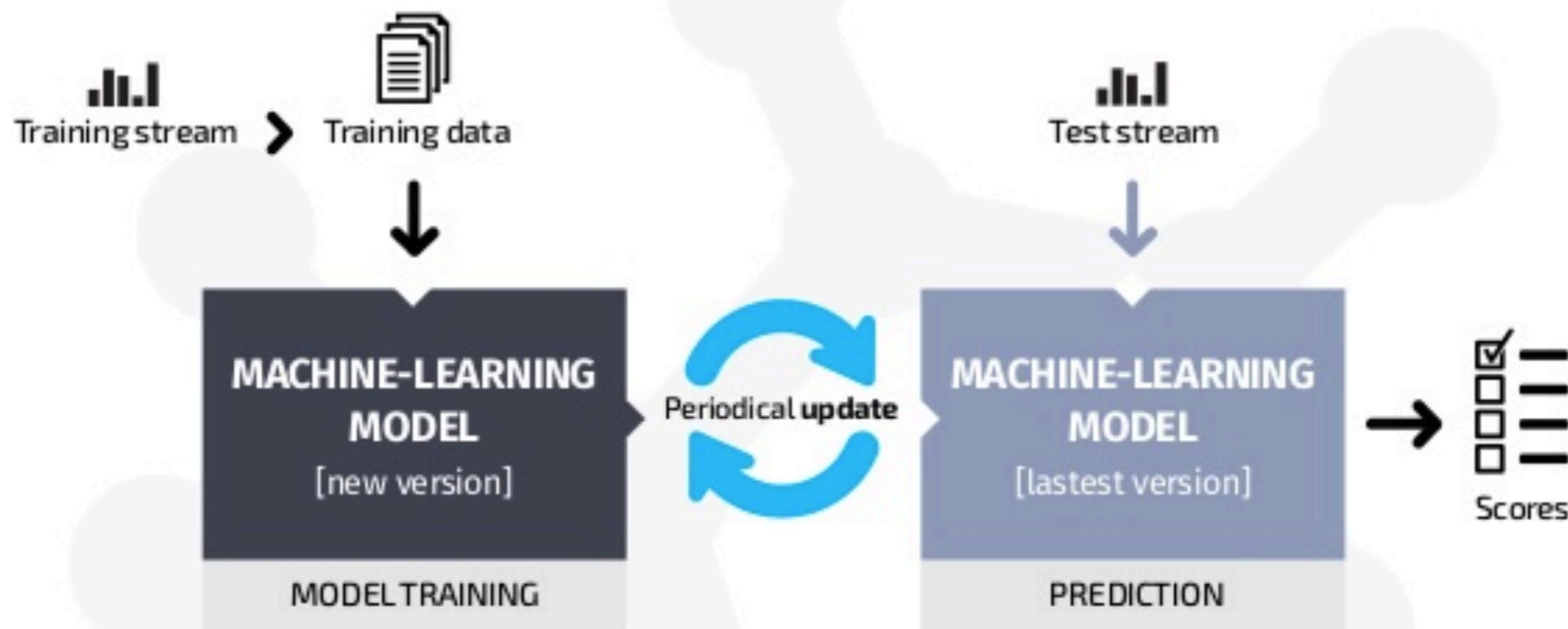


Model update after each processed training example

# OFFLINE LEARNING



## Offline learning



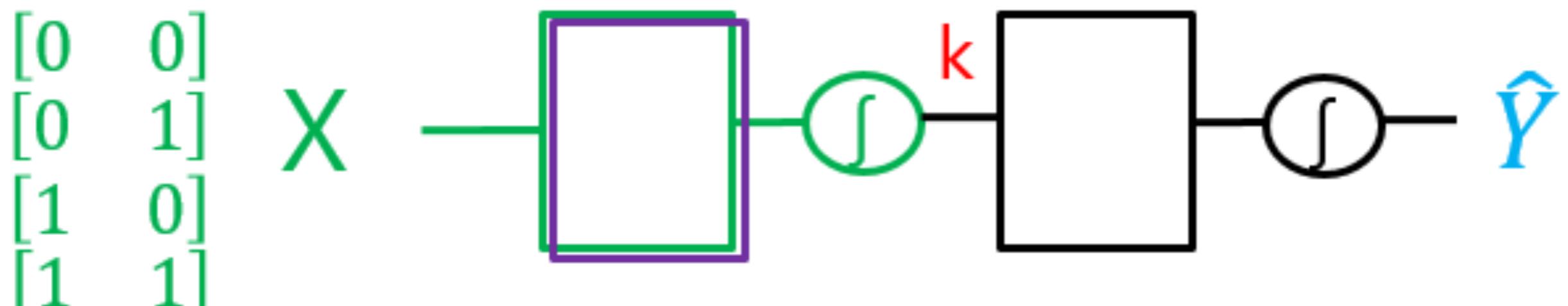


DEEP LEARNING

---

HINTON'S  
SUMMARY

## NEURAL NETWORK FOR XOR



$$W_1 = \begin{bmatrix} 5 & -7 \\ 5 & -7 \end{bmatrix}$$

$$B_1 = [-8 \quad 3]$$

$$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix}$$

$$B_2 = 6$$

# 9 HIDDEN LAYERS FOR XOR PROBLEM

```
W1 = tf.Variable(tf.random_uniform([2, 5], -1.0, 1.0), name = "Weight1")
W2 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight2")
W3 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight3")
W4 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight4")
W5 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight5")
W6 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight6")
W7 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight7")
W8 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight8")
W9 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight9")
W10 = tf.Variable(tf.random_uniform([5, 5], -1.0, 1.0), name = "Weight10")

W11 = tf.Variable(tf.random_uniform([5, 1], -1.0, 1.0), name = "Weight11")

b1 = tf.Variable(tf.zeros([5]), name="Bias1")
b2 = tf.Variable(tf.zeros([5]), name="Bias2")
b3 = tf.Variable(tf.zeros([5]), name="Bias3")
b4 = tf.Variable(tf.zeros([5]), name="Bias4")
b5 = tf.Variable(tf.zeros([5]), name="Bias5")
b6 = tf.Variable(tf.zeros([5]), name="Bias6")
b7 = tf.Variable(tf.zeros([5]), name="Bias7")
b8 = tf.Variable(tf.zeros([5]), name="Bias8")
b9 = tf.Variable(tf.zeros([5]), name="Bias9")
b10 = tf.Variable(tf.zeros([5]), name="Bias10")

b11 = tf.Variable(tf.zeros([1]), name="Bias11")
```

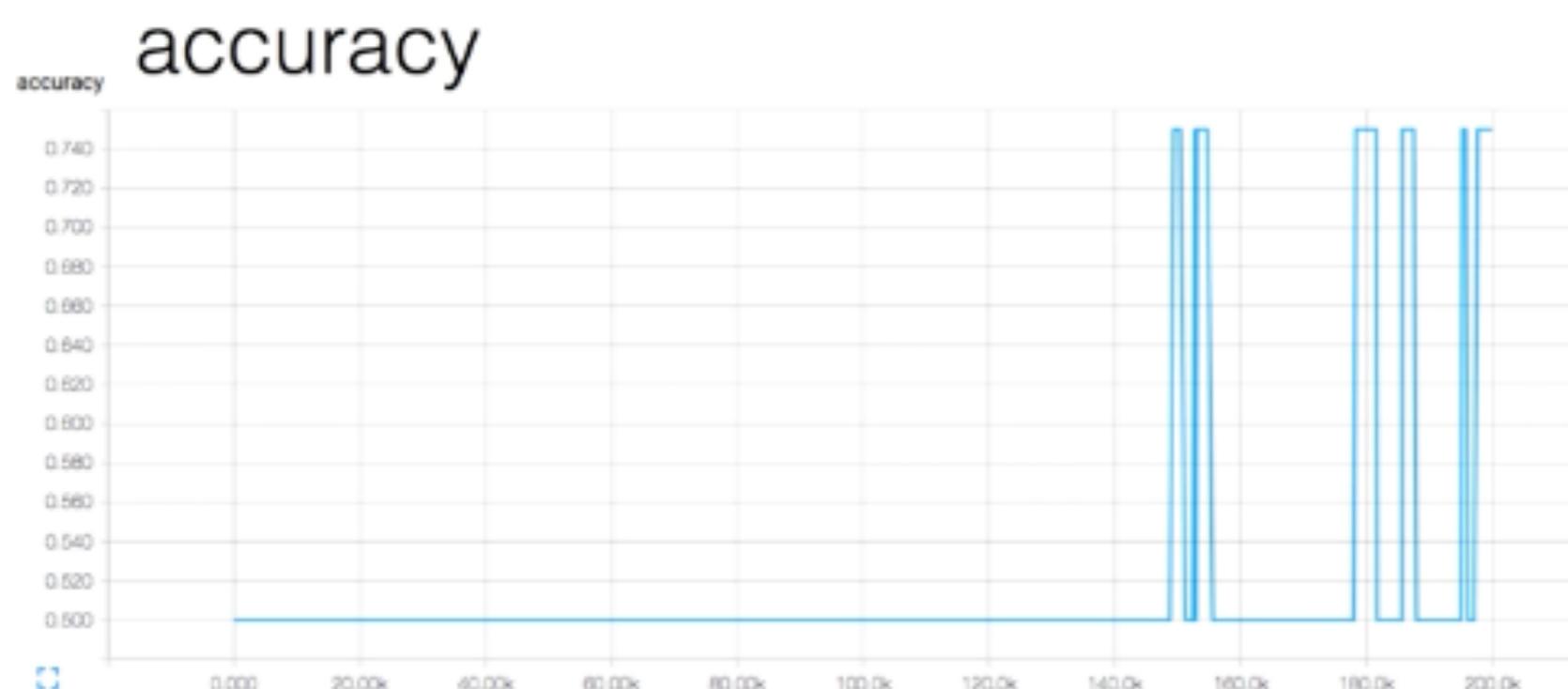
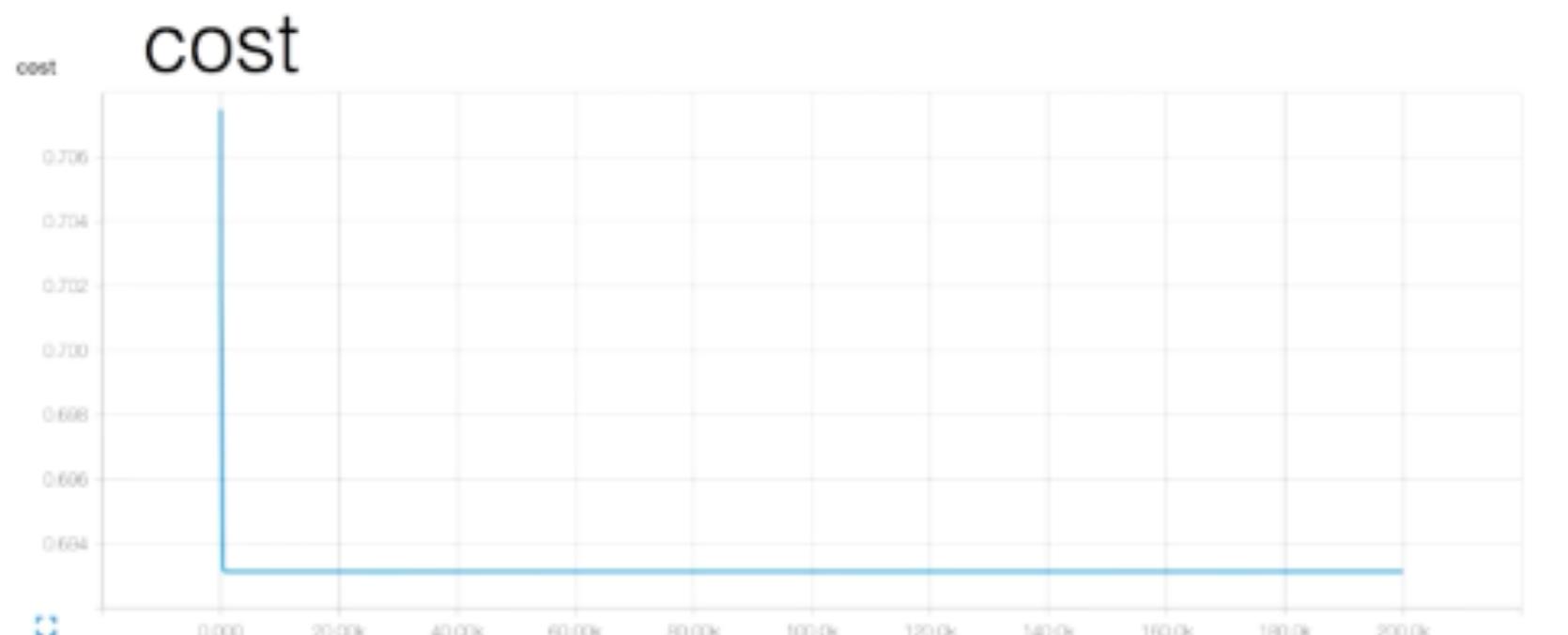
```
# Our hypothesis
with tf.name_scope("layer1") as scope:
    L1 = tf.sigmoid(tf.matmul(X, W1) + b1)
with tf.name_scope("layer2") as scope:
    L2 = tf.sigmoid(tf.matmul(L1, W2) + b2)
with tf.name_scope("layer3") as scope:
    L3 = tf.sigmoid(tf.matmul(L2, W3) + b3)
with tf.name_scope("layer4") as scope:
    L4 = tf.sigmoid(tf.matmul(L3, W4) + b4)
with tf.name_scope("layer5") as scope:
    L5 = tf.sigmoid(tf.matmul(L4, W5) + b5)
with tf.name_scope("layer6") as scope:
    L6 = tf.sigmoid(tf.matmul(L5, W6) + b6)
with tf.name_scope("layer7") as scope:
    L7 = tf.sigmoid(tf.matmul(L6, W7) + b7)
with tf.name_scope("layer8") as scope:
    L8 = tf.sigmoid(tf.matmul(L7, W8) + b8)
with tf.name_scope("layer9") as scope:
    L9 = tf.sigmoid(tf.matmul(L8, W9) + b9)
with tf.name_scope("layer10") as scope:
    L10 = tf.sigmoid(tf.matmul(L9, W10) + b10)

with tf.name_scope("last") as scope:
    hypothesis = tf.sigmoid(tf.matmul(L10, W11) + b11)
```

## POOR RESULTS?

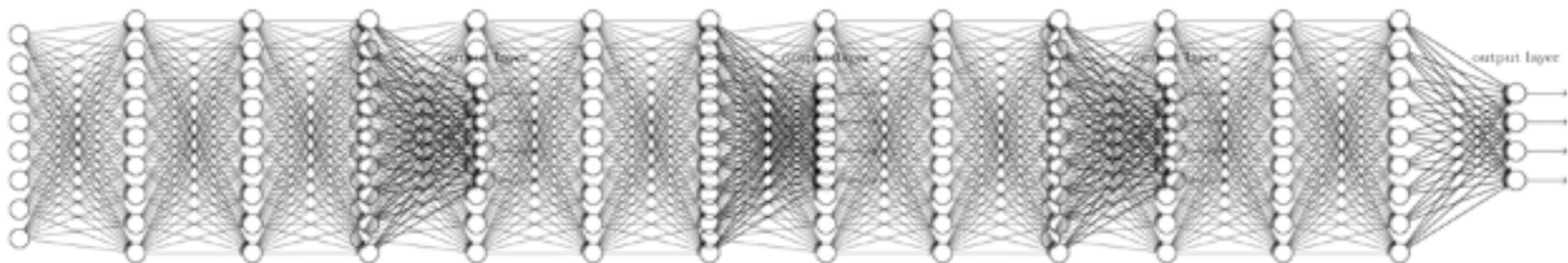
```
196000 [0.69314718, array([[ 0.49999988],  
[ 0.50000006],  
[ 0.49999982],  
[ 0.5 ]], dtype=float32)]  
198000 [0.69314718, array([[ 0.49999988],  
[ 0.50000006],  
[ 0.49999982],  
[ 0.5 ]], dtype=float32)]  
[array([[ 0.49999988],  
[ 0.50000006],  
[ 0.49999982],  
[ 0.5 ]], dtype=float32), array([[ 0.],  
[ 1.],  
[ 0.],  
[ 1.]], dtype=float32)]  
Accuracy: 0.5
```

# POOR RESULTS?



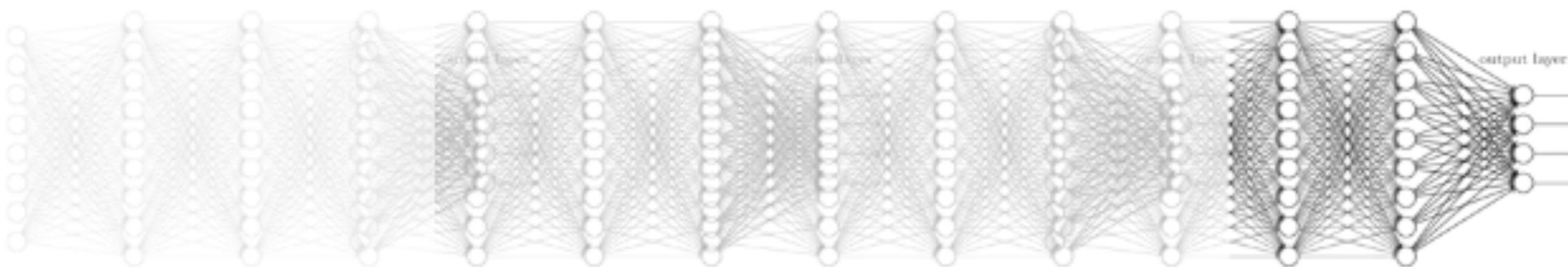
Tensorboard  
Cost &  
Accuracy

# BACKPROPAGATION



## VANISHING GRADIENT

Vanishing gradient (NN winter2: 1986-2006)



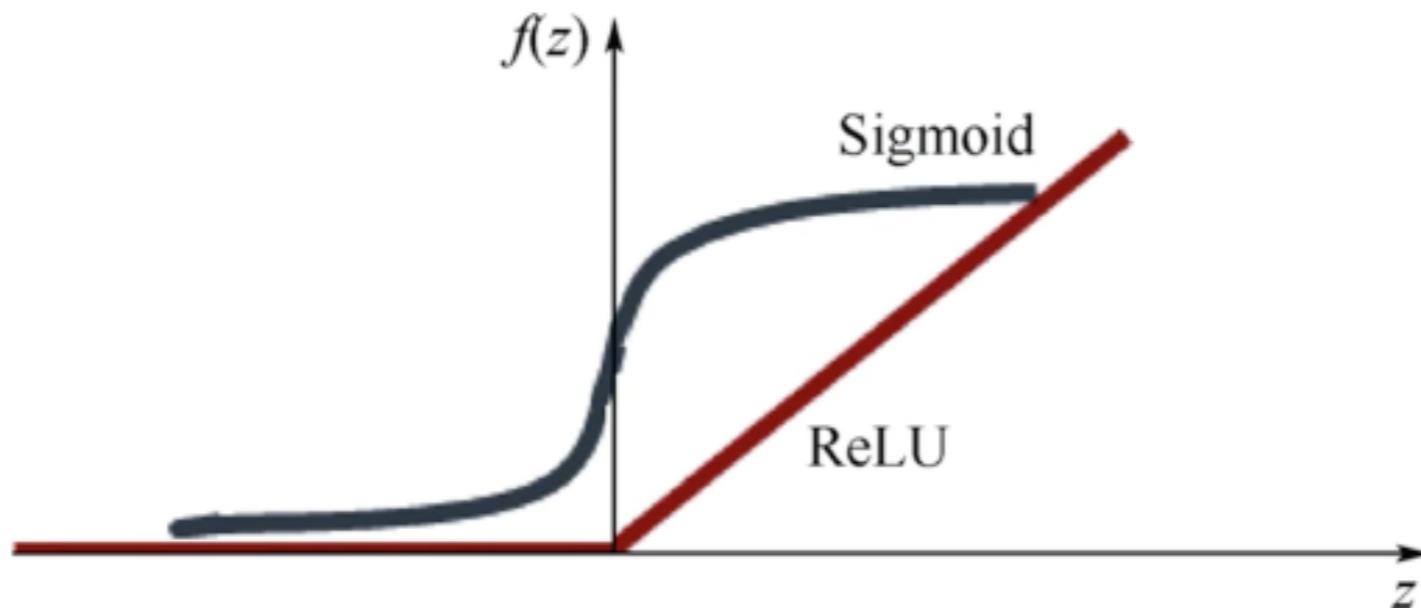
# HINTON'S SUMMARY

## Geoffrey Hinton's summary of findings up to today

- Our labeled datasets were thousands of times too small.
- Our computers were millions of times too slow.
- We initialized the weights in a stupid way.
- **We used the wrong type of non-linearity.**

# SIGMOID VS. RELU

Sigmoid!



ReLU: Rectified Linear Unit

`L1 = tf.sigmoid(tf.matmul(X, W1) + b1)`

`L1 = tf.nn.relu(tf.matmul(X, W1) + b1)`

# RELU

## ReLU

```
# Our hypothesis
with tf.name_scope("layer1") as scope:
    L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
with tf.name_scope("layer2") as scope:
    L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)
with tf.name_scope("layer3") as scope:
    L3 = tf.nn.relu(tf.matmul(L2, W3) + b3)
with tf.name_scope("layer4") as scope:
    L4 = tf.nn.relu(tf.matmul(L3, W4) + b4)
with tf.name_scope("layer5") as scope:
    L5 = tf.nn.relu(tf.matmul(L4, W5) + b5)
with tf.name_scope("layer6") as scope:
    L6 = tf.nn.relu(tf.matmul(L5, W6) + b6)
with tf.name_scope("layer7") as scope:
    L7 = tf.nn.relu(tf.matmul(L6, W7) + b7)
with tf.name_scope("layer8") as scope:
    L8 = tf.nn.relu(tf.matmul(L7, W8) + b8)
with tf.name_scope("layer9") as scope:
    L9 = tf.nn.relu(tf.matmul(L8, W9) + b9)
with tf.name_scope("layer10") as scope:
    L10 = tf.nn.relu(tf.matmul(L9, W10) + b10)

with tf.name_scope("last") as scope:
    hypothesis = tf.sigmoid(tf.matmul(L10, W11) + b11)
```

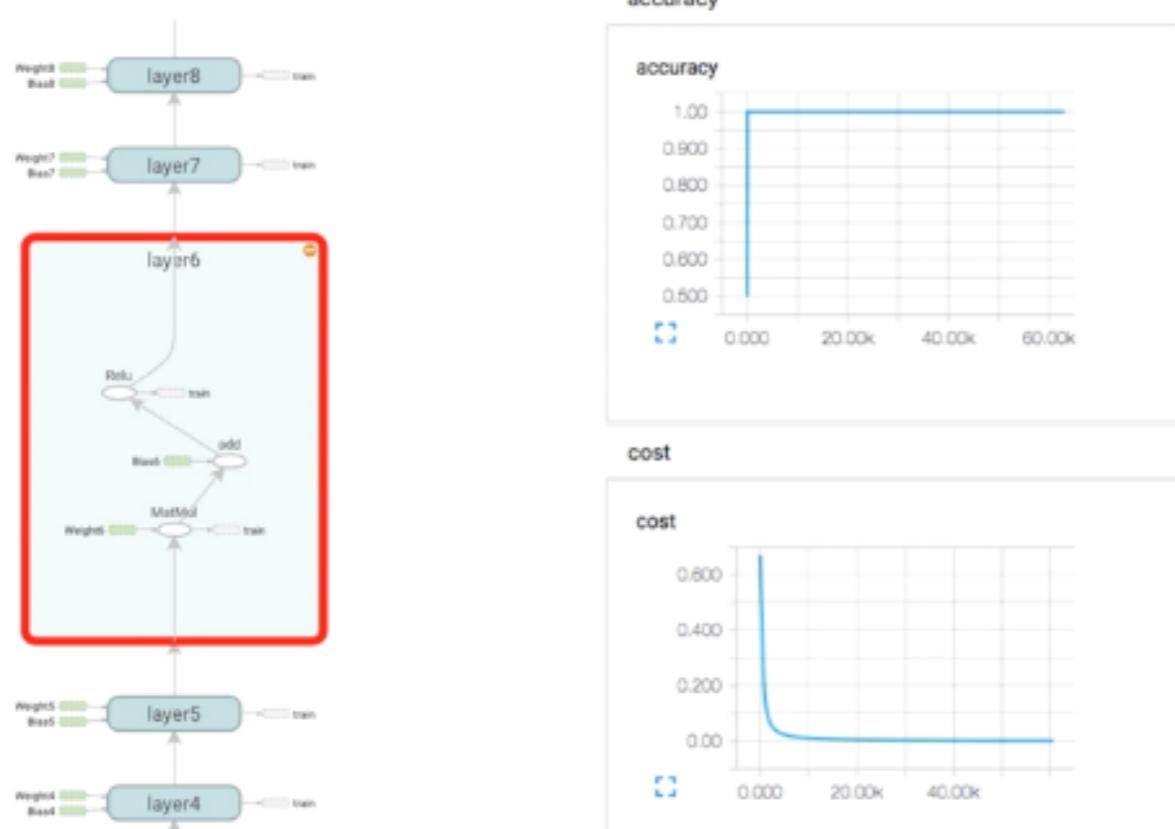
## Works very well

```
196000 [2.6226094e-06, array([[ 2.59195826e-06],
[ 9.99999642e-01],
[ 9.99994874e-01],
[ 2.43454133e-06]], dtype=float32)]
198000 [2.607708e-06, array([[ 2.55822852e-06],
[ 9.99999642e-01],
[ 9.99994874e-01],
[ 2.40260101e-06]], dtype=float32)]
[array([[ 2.52509381e-06],
[ 9.99999642e-01],
[ 9.99994874e-01],
[ 2.37124709e-06]], dtype=float32), array([[ 0.],
[ 1.],
[ 1.],
[ 0.]], dtype=float32)]
Accuracy: 1.0
```

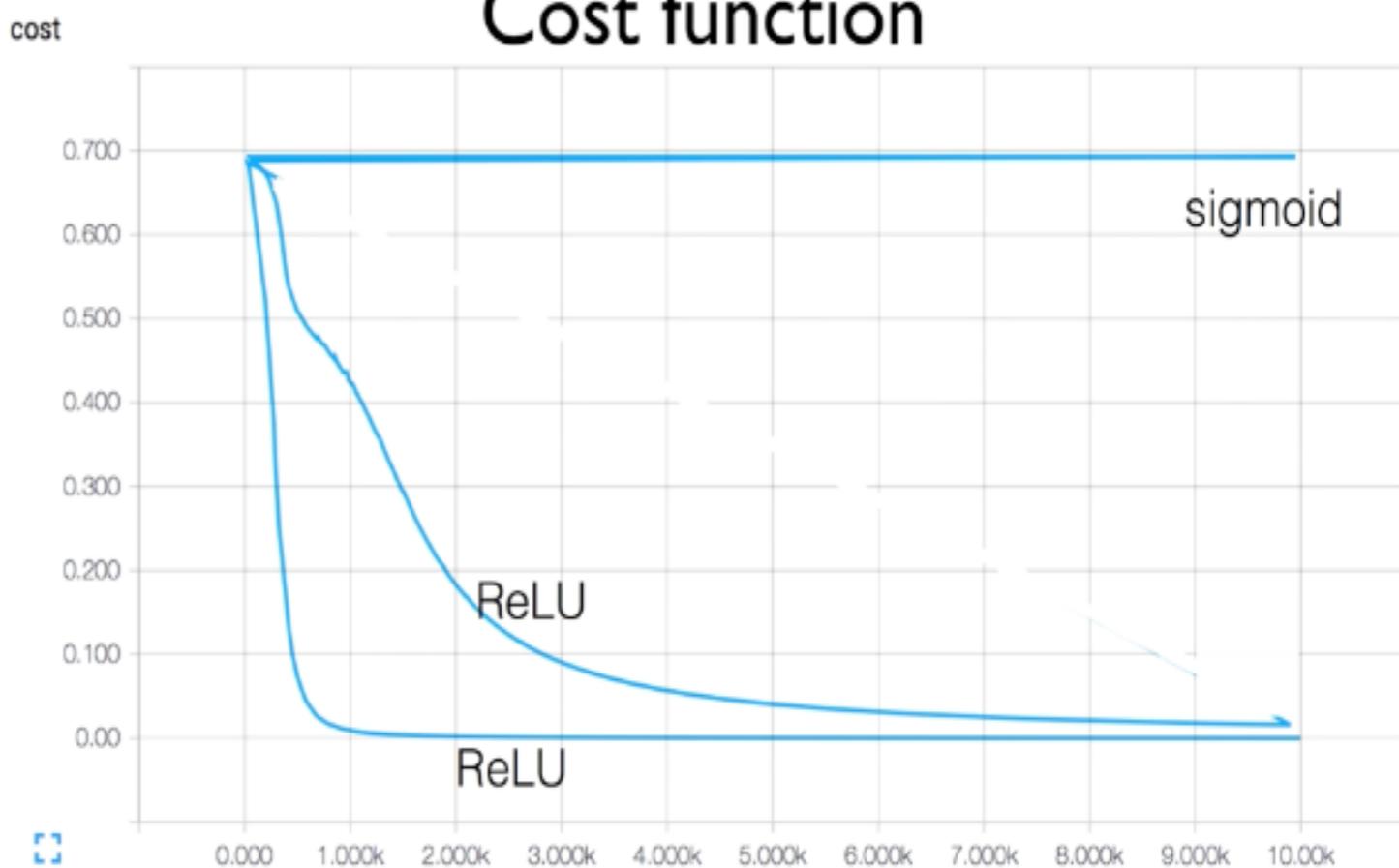
# HINTON'S SUMMARY

## TENSORBOARD

Works very well



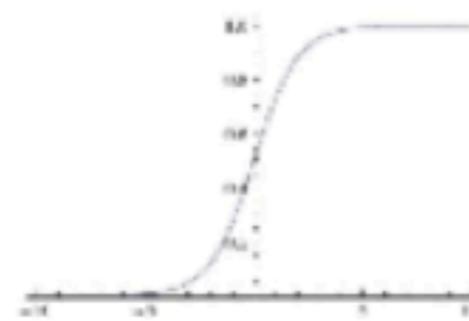
Cost function



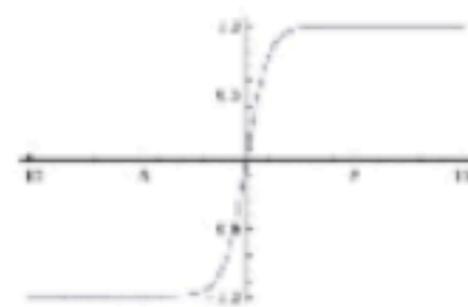
# ACTIVATION FUNCTIONS

**Sigmoid**

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



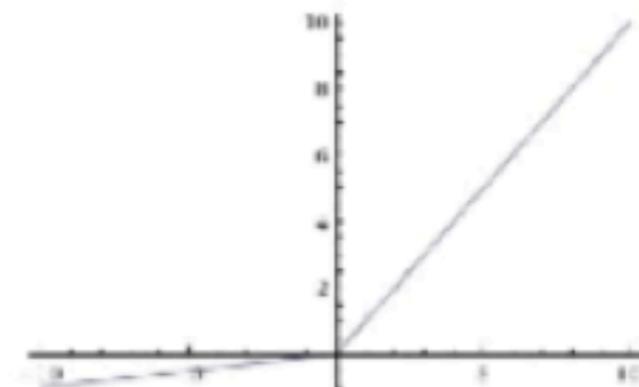
**tanh**     $\tanh(x)$



**ReLU**     $\max(0, x)$

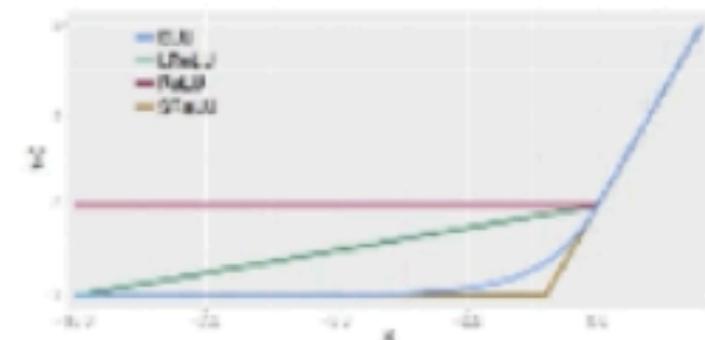


**Leaky ReLU**  
 $\max(0.1x, x)$

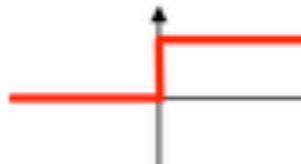
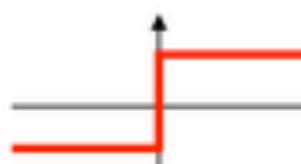
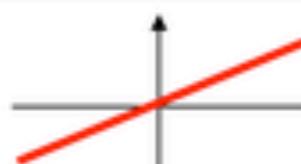
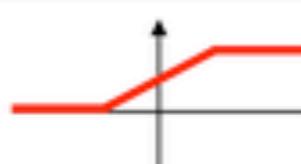
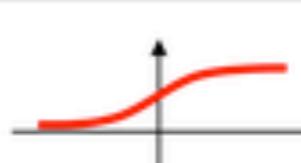


**Maxout**     $\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**     $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$



# ACTIVATION FUNCTIONS

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer NN	

## ACTIVATION FUNCTIONS ON CIFAR-10

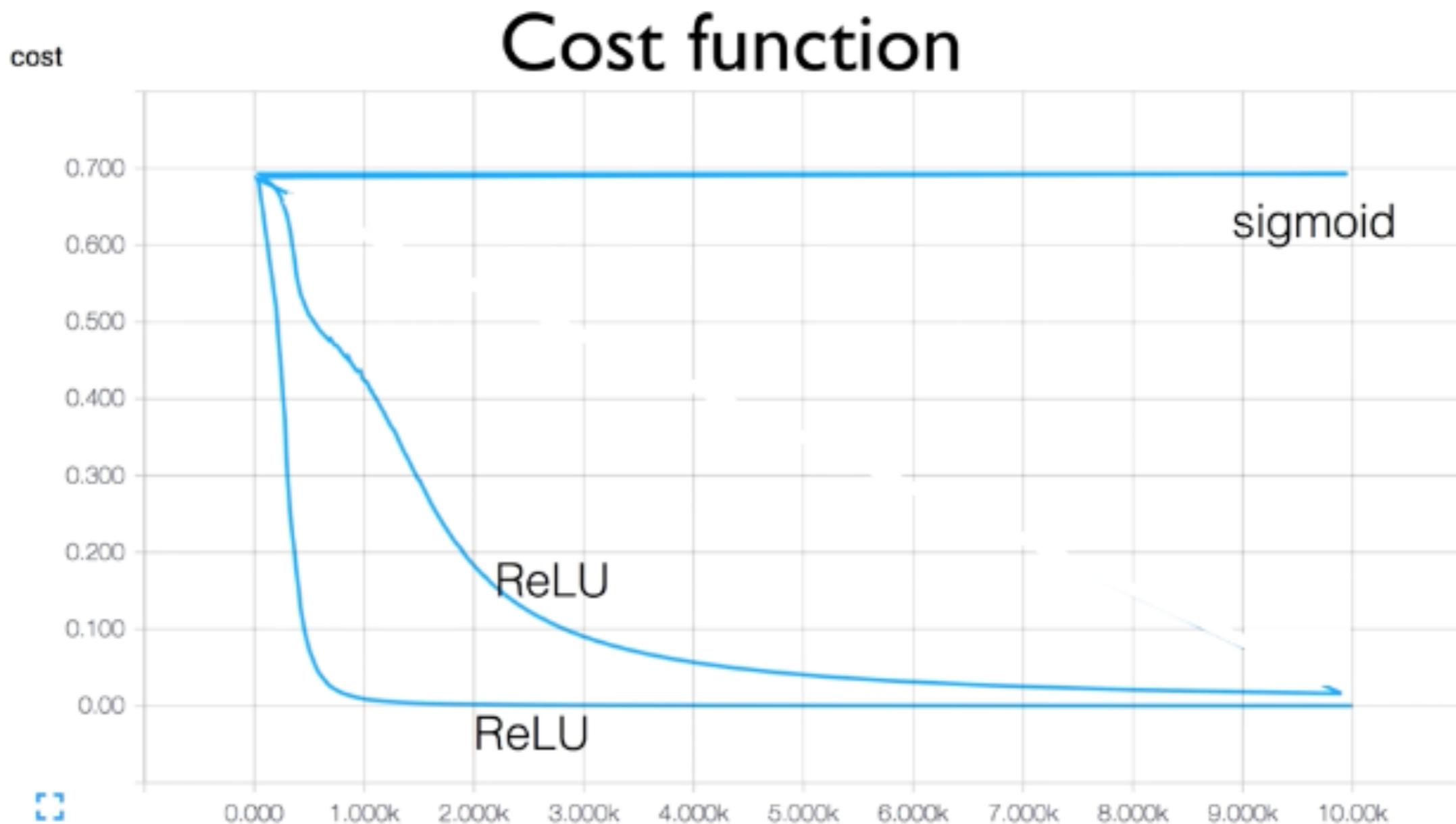
maxout	ReLU	VLReLU	tanh	Sigmoid
<b>93.94</b>	<b>92.11</b>	<b>92.97</b>	89.28	n/c
93.78	91.74	92.40	89.48	n/c
–	91.93	<b>93.09</b>	–	n/c
91.75	90.63	92.27	<b>89.82</b>	n/c
n/c†	90.91	92.43	89.54	n/c

# HINTON'S SUMMARY

Geoffrey Hinton's summary of findings up to today

- Our labeled datasets were thousands of times too small.
- Our computers were millions of times too slow.
- **We initialized the weights in a stupid way.**
- We used the wrong type of non-linearity.

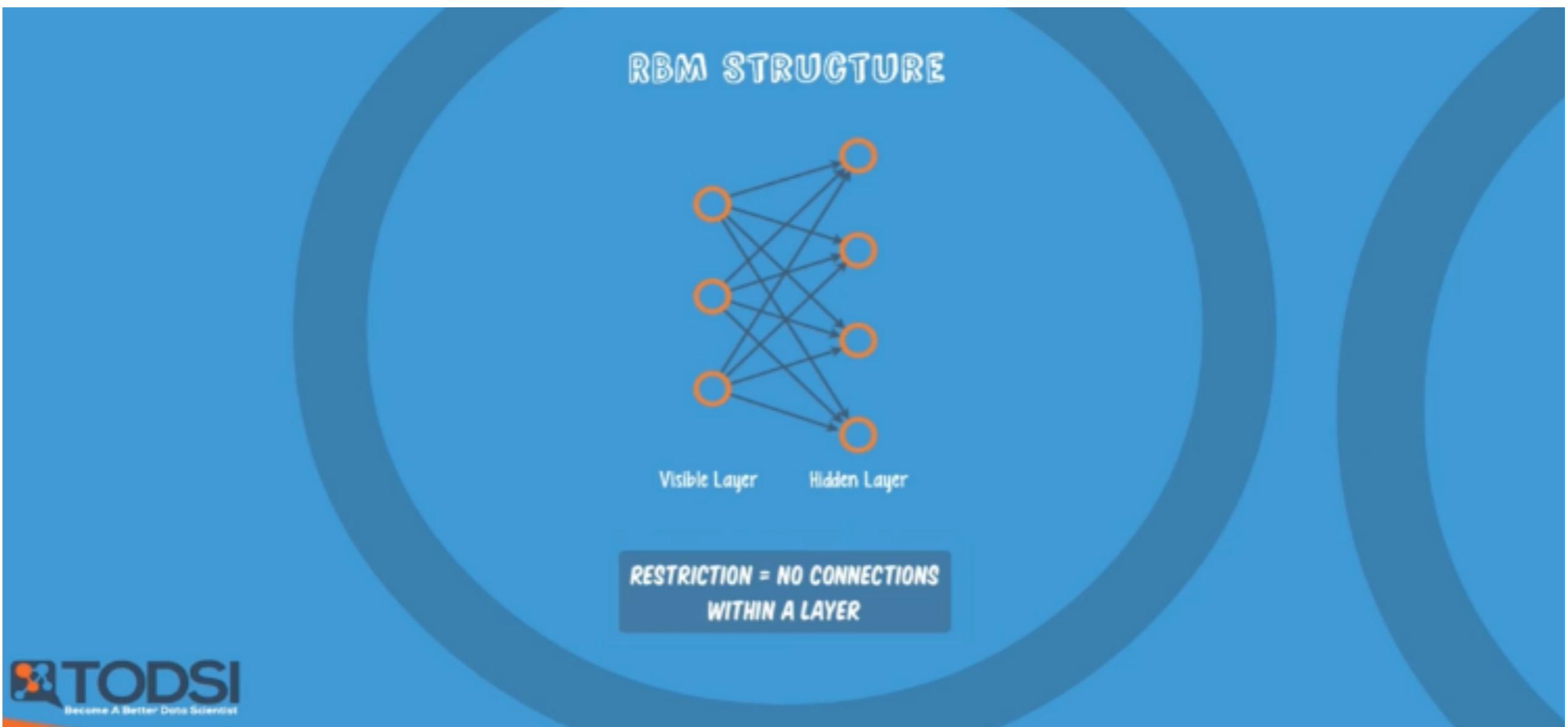
# RELU COST FUNCTION



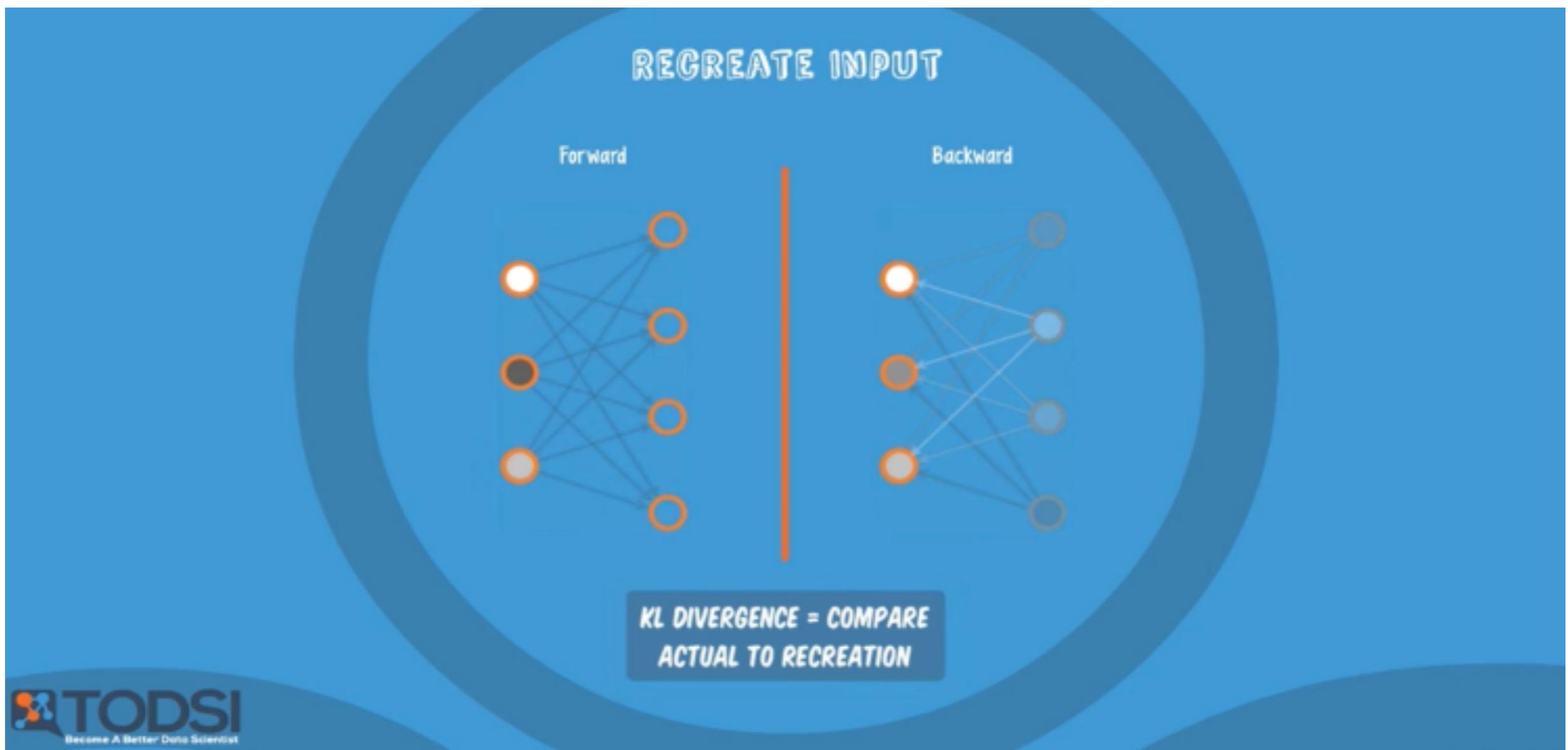
## WISE INITIALIZATION

- Not all 0's
- Challenging issue
- Hinton et al. (2006) "A Fast Learning Algorithm for Deep Belief Nets"
  - Restricted Boatman Machine (RBM)

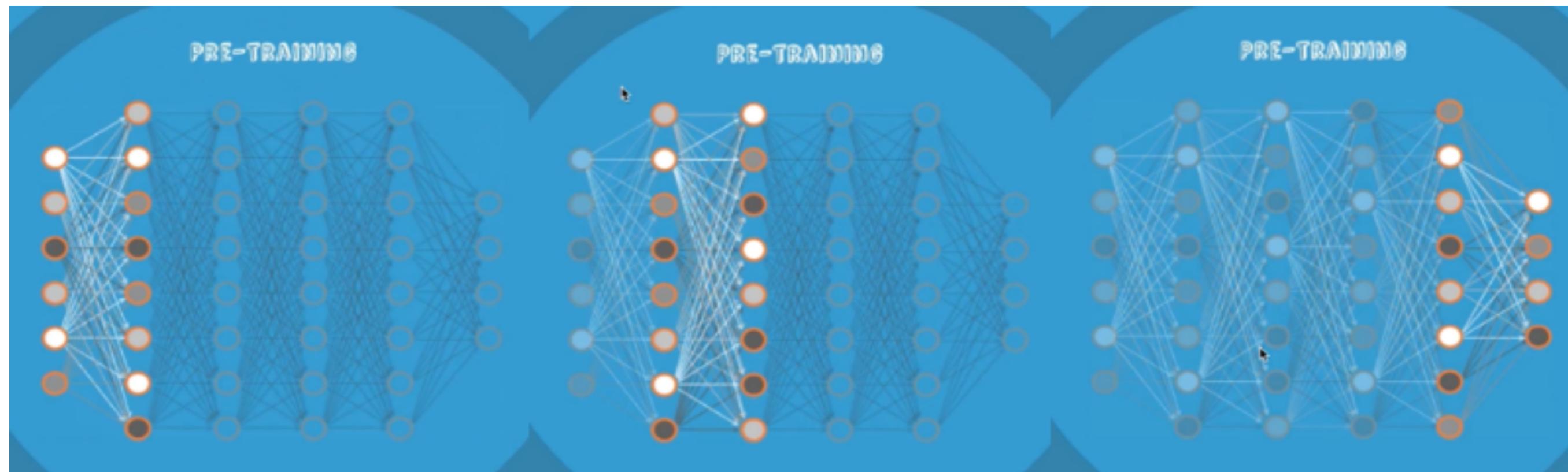
# RBM STRUCTURE



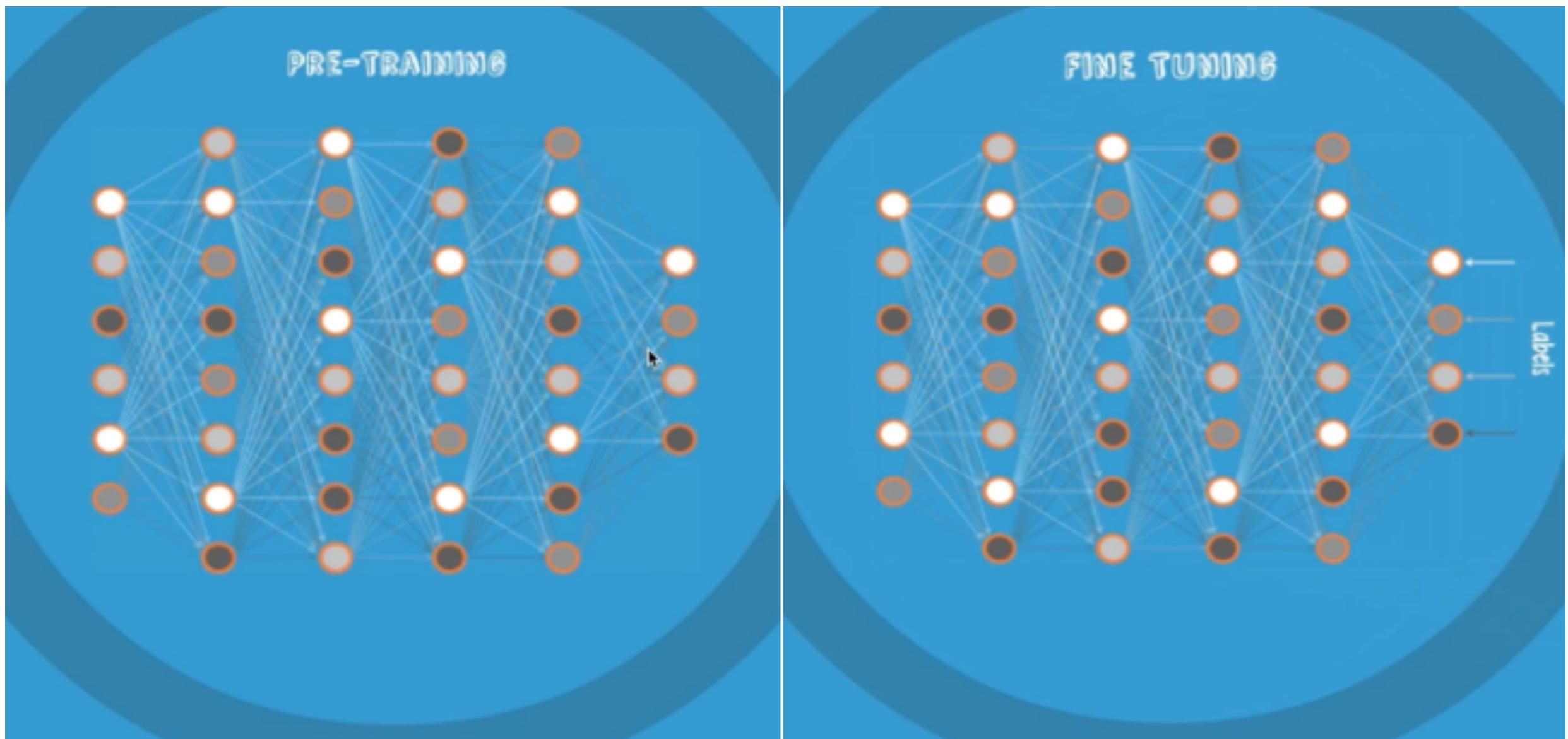
# RBM STRUCTURE



# RBM STRUCTURE



## RBM STRUCTURE



# GOOD NEWS

- No need to use complicated RBM for weight initializations
- Simple methods are OK
  - **Xavier initialization:** X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in International conference on artificial intelligence and statistics, 2010
  - **He’s initialization:** K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” 2015

# XAVIER/HE INITIALIZATION

- Makes sure the weights are ‘just right’, not too small, not too big
- Using number of input (fan\_in) and output (fan\_out)

```
# Xavier initialization  
# Glorot et al. 2010  
W = np.random.randn(fan_in, fan_out)/np.sqrt(fan_in)  
  
# He et al. 2015  
W = np.random.randn(fan_in, fan_out)/np.sqrt(fan_in/2)
```

# ACTIVATION FUNCTIONS AND INITIALIZATION ON CIFAR-10

Init method	maxout	ReLU	VLReLU	tanh	Sigmoid
LSUV	<b>93.94</b>	<b>92.11</b>	92.97	89.28	n/c
OrthoNorm	93.78	91.74	92.40	89.48	n/c
OrthoNorm-MSRA scaled	–	91.93	<b>93.09</b>	–	n/c
Xavier	91.75	90.63	92.27	<b>89.82</b>	n/c
MSRA	n/c†	90.91	92.43	89.54	n/c

## STILL AN ACTIVE AREA OF RESEARCH

- We don't know how to initialize perfect weight values, yet
- Many new algorithms
  - Batch normalization
  - Layer sequential uniform variance
  - ...

## HINTON'S SUMMARY

- Our labeled datasets were thousands of times too small.
- Our computers were millions of times too slow.
- We initialized the weights in a stupid way.
- We used the wrong type of non-linearity.

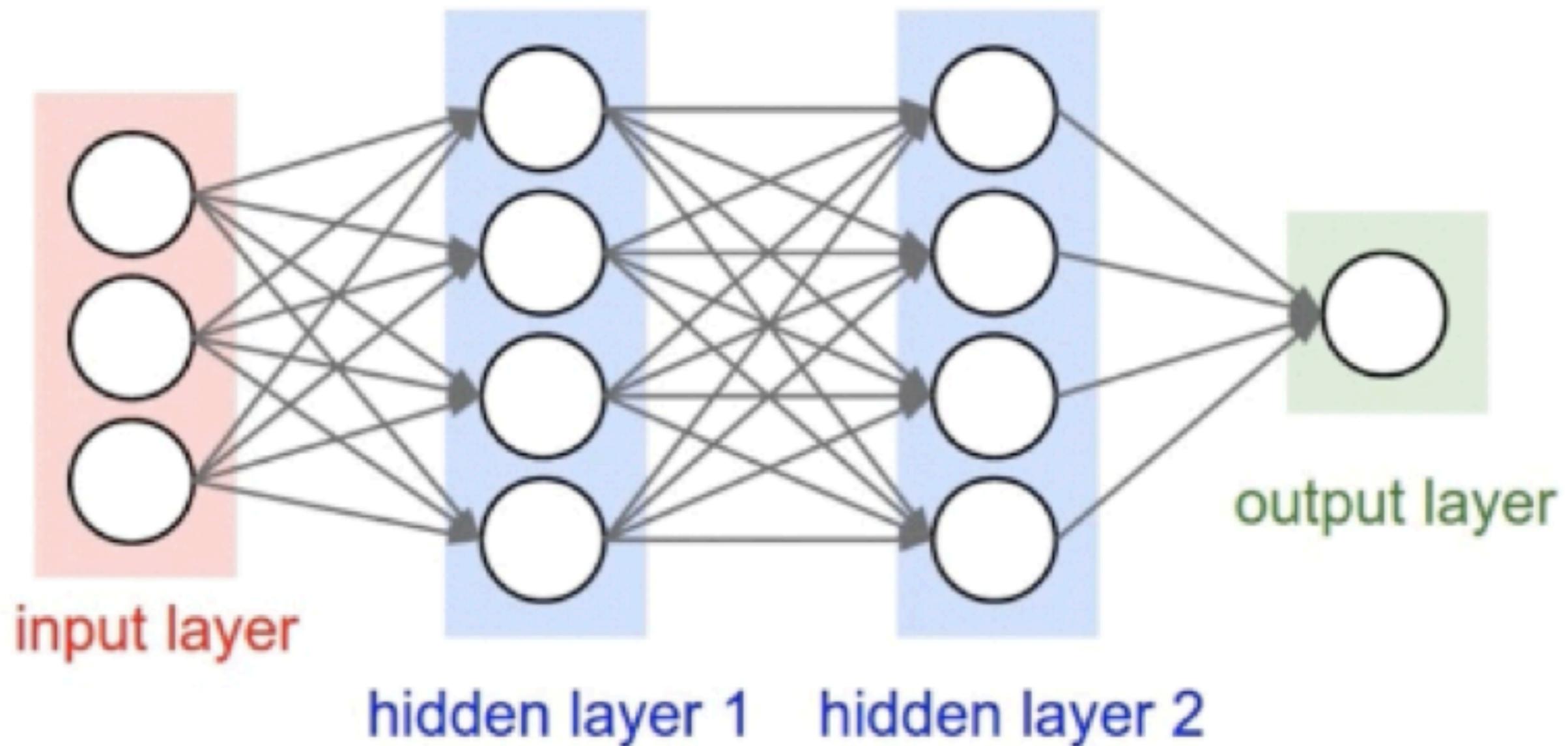


# DEEP LEARNING

---

# BACK PROPAGATION

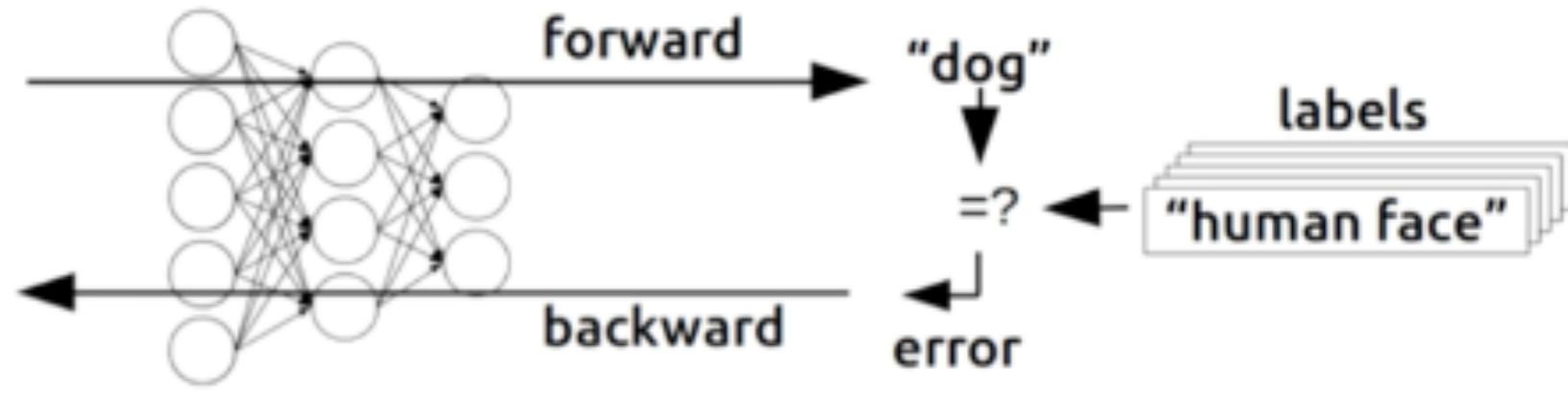
## DERIVATION



# BACKPROPAGATION

Backpropagation  
(1974, 1982 by Paul Werbos, 1986 by Hinton)

Training



## CHAIN RULE

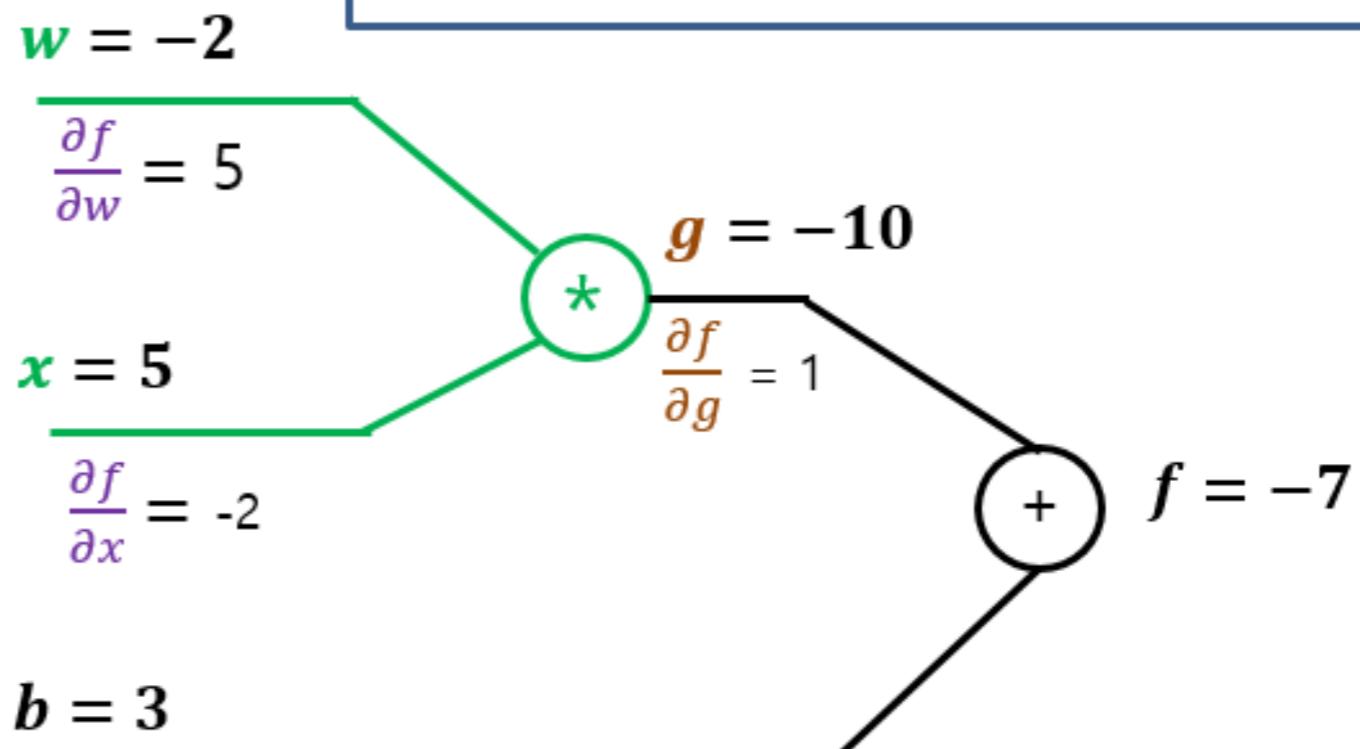
$$f = \mathbf{w}\mathbf{x} + b$$

- ① forward( $w = -2, x = 5, b = 3$ )  
 ② backward

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x} = 1 * w = -2$$

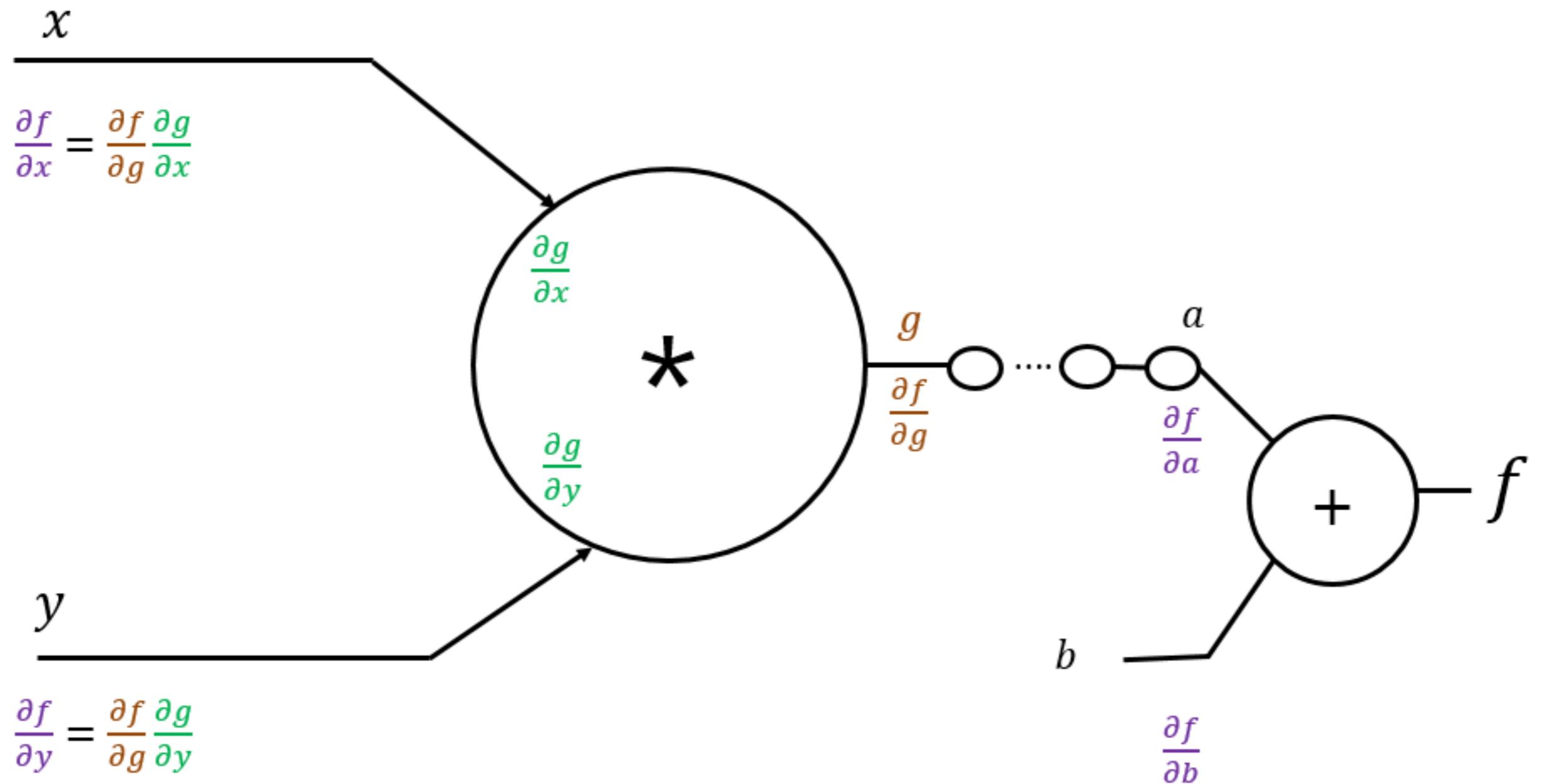
$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial w} = 1 * x = 5$$

$g = \mathbf{w}\mathbf{x}$	$f = g + b$
$\frac{\partial g}{\partial w} = x, \frac{\partial g}{\partial x} = w$	$\frac{\partial f}{\partial g} = 1, \frac{\partial f}{\partial b} = 1$

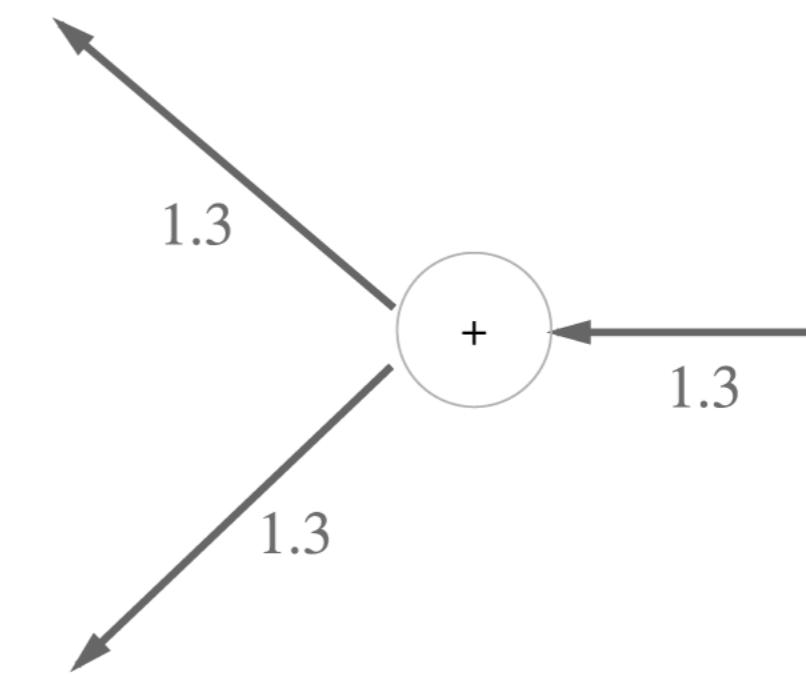
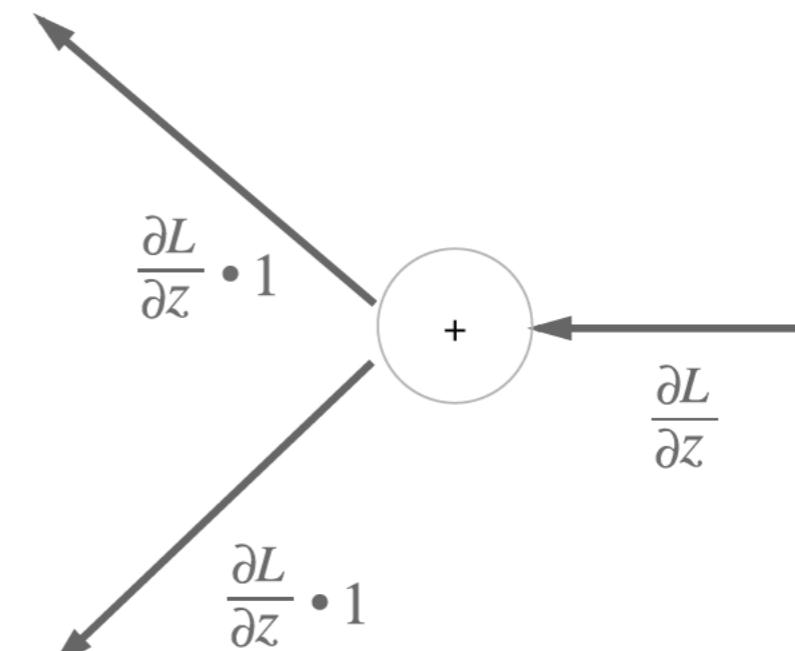
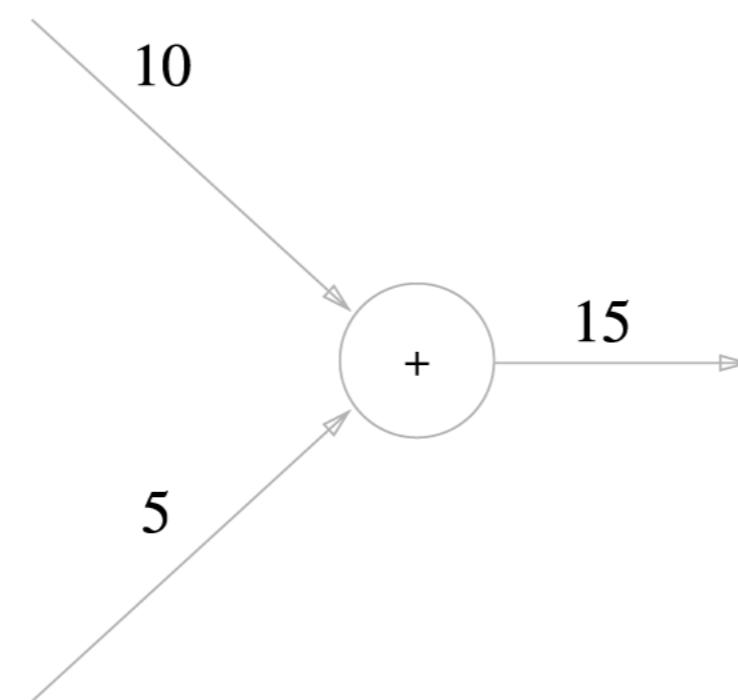
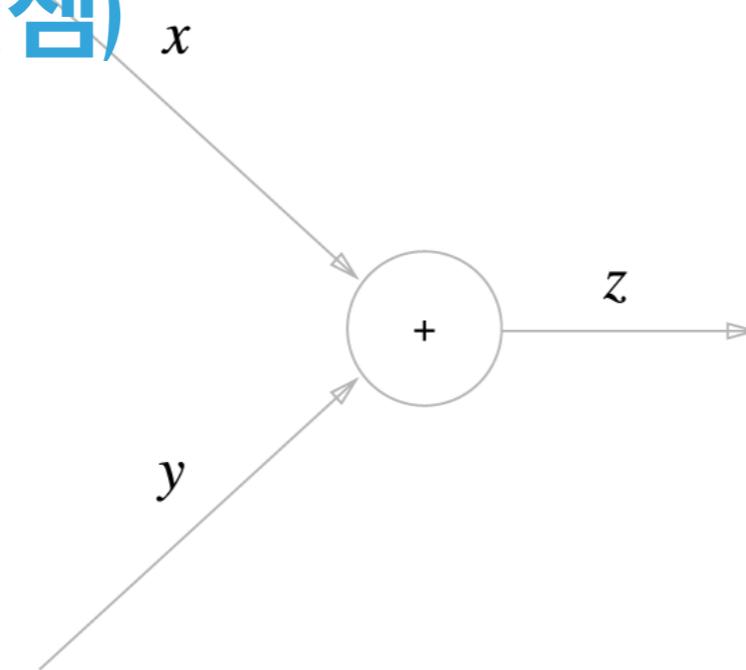


$$\frac{\partial f}{\partial b} = 1$$

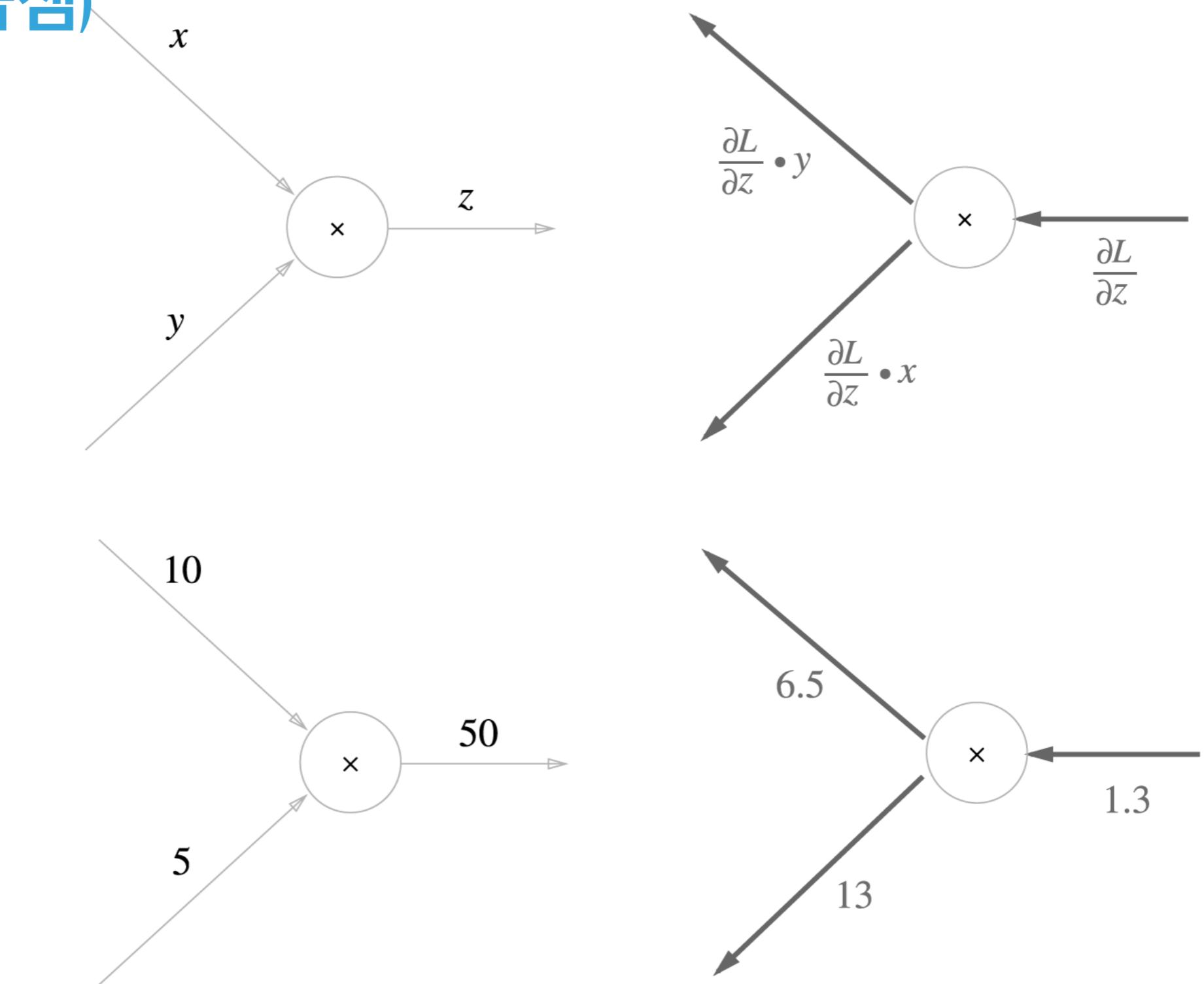
## CHAIN RULE



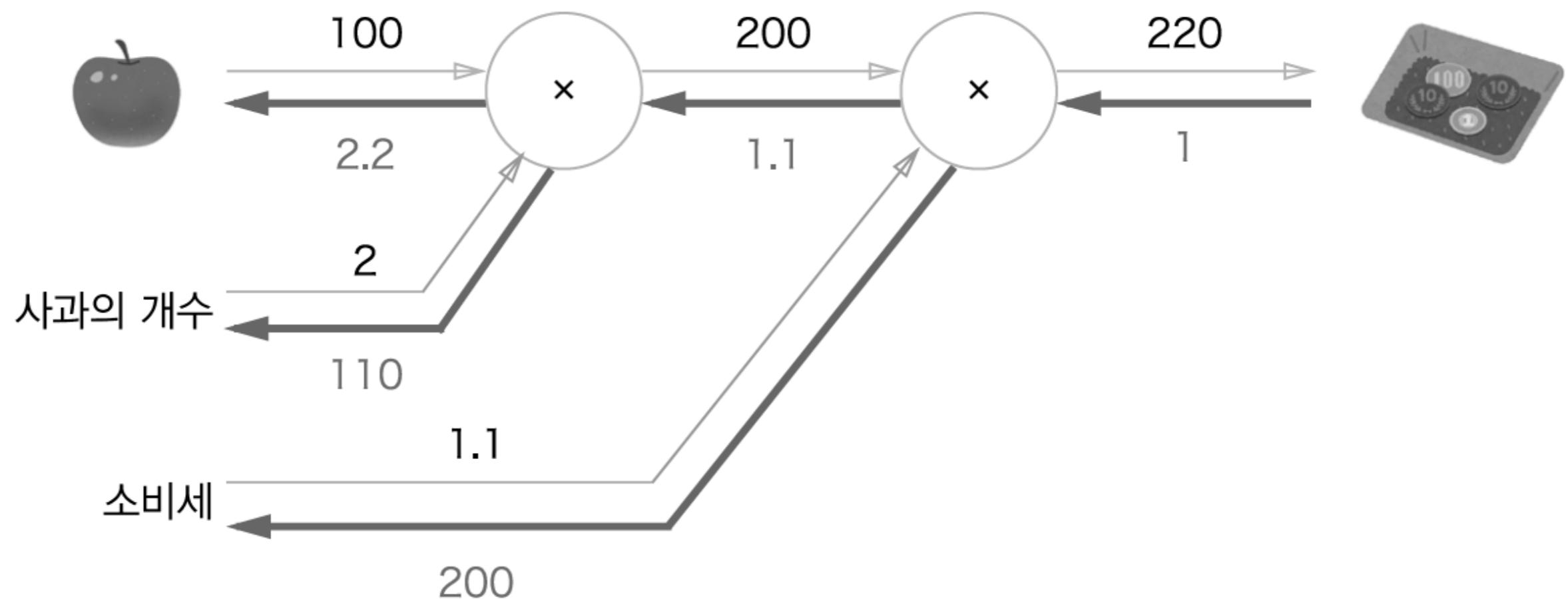
## 계산 그래프 (덧셈)



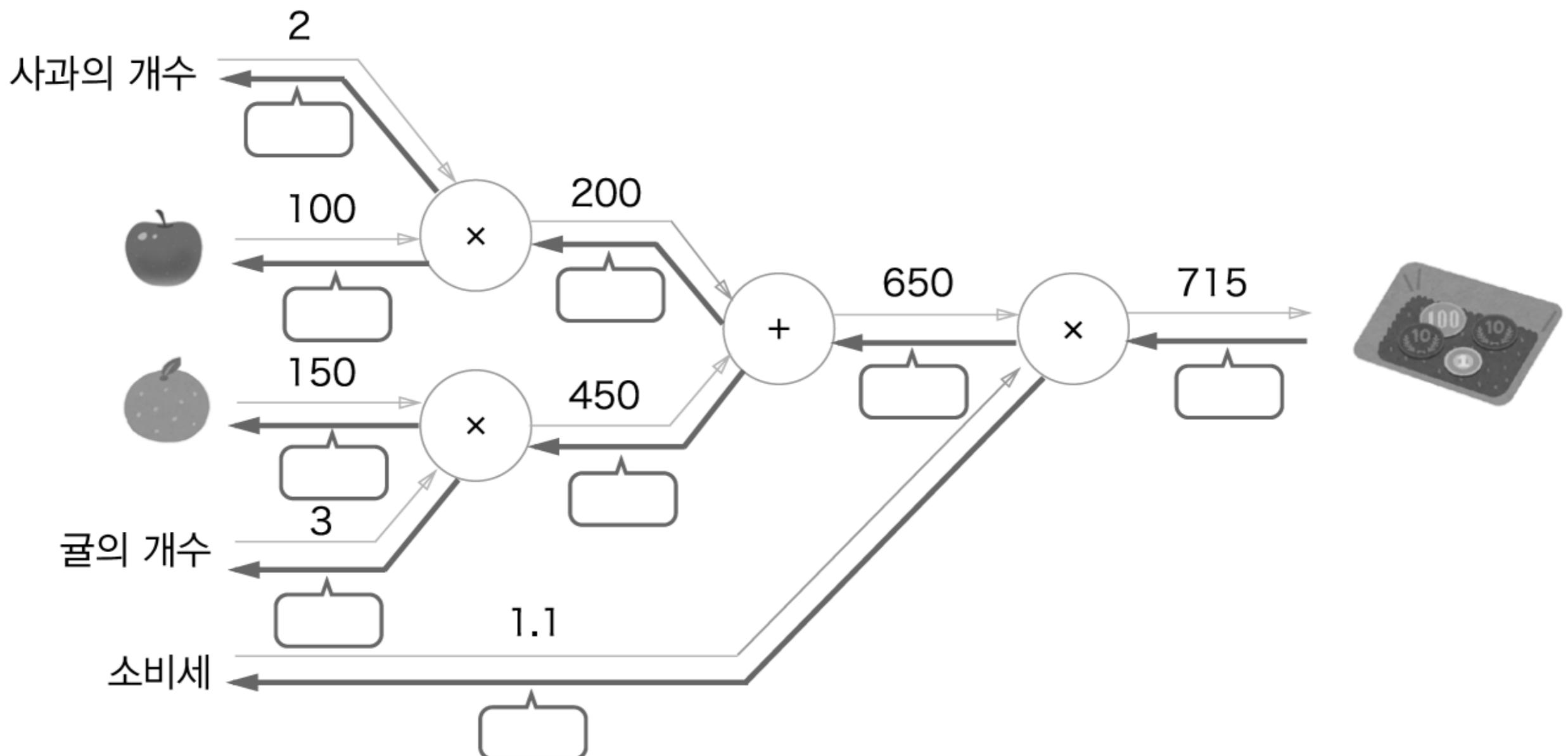
## 계산 그래프 (곱셈)



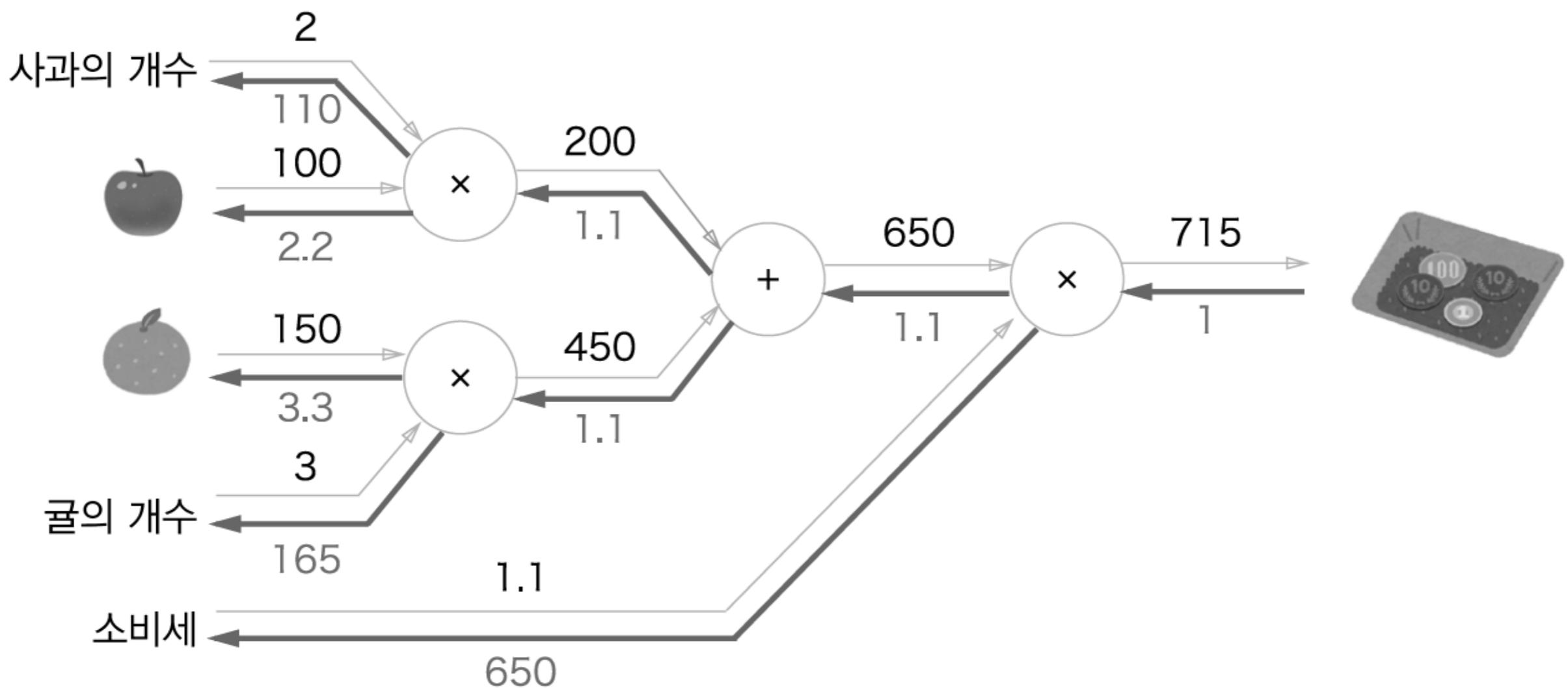
## 사과 그래프



## 과일 그래프 (퀴즈)

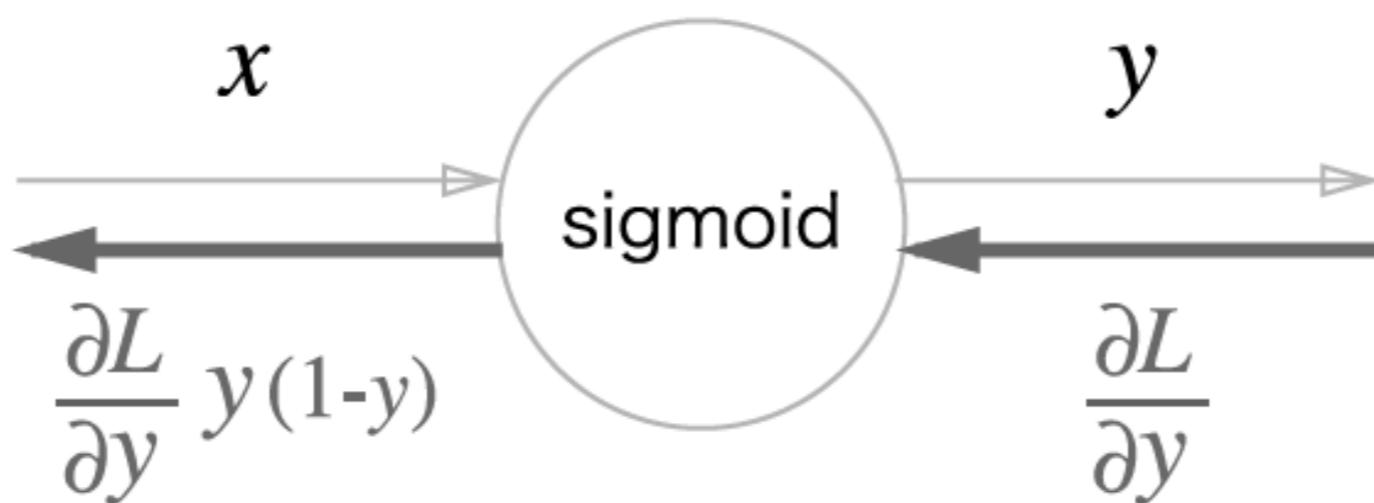
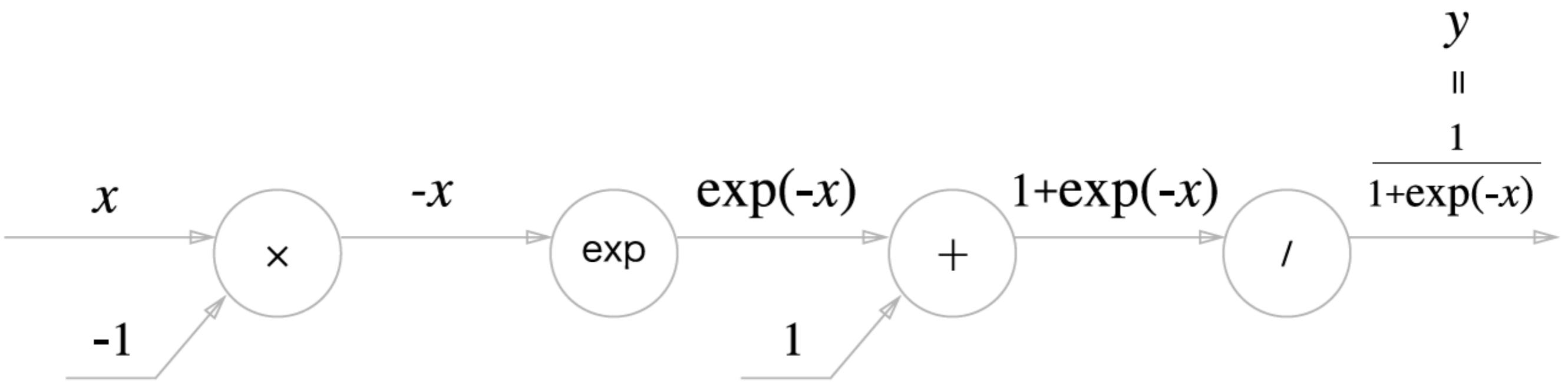


## 과일 그래프



## SIGMOID

$$g(z) = \frac{1}{1+e^{-z}}$$



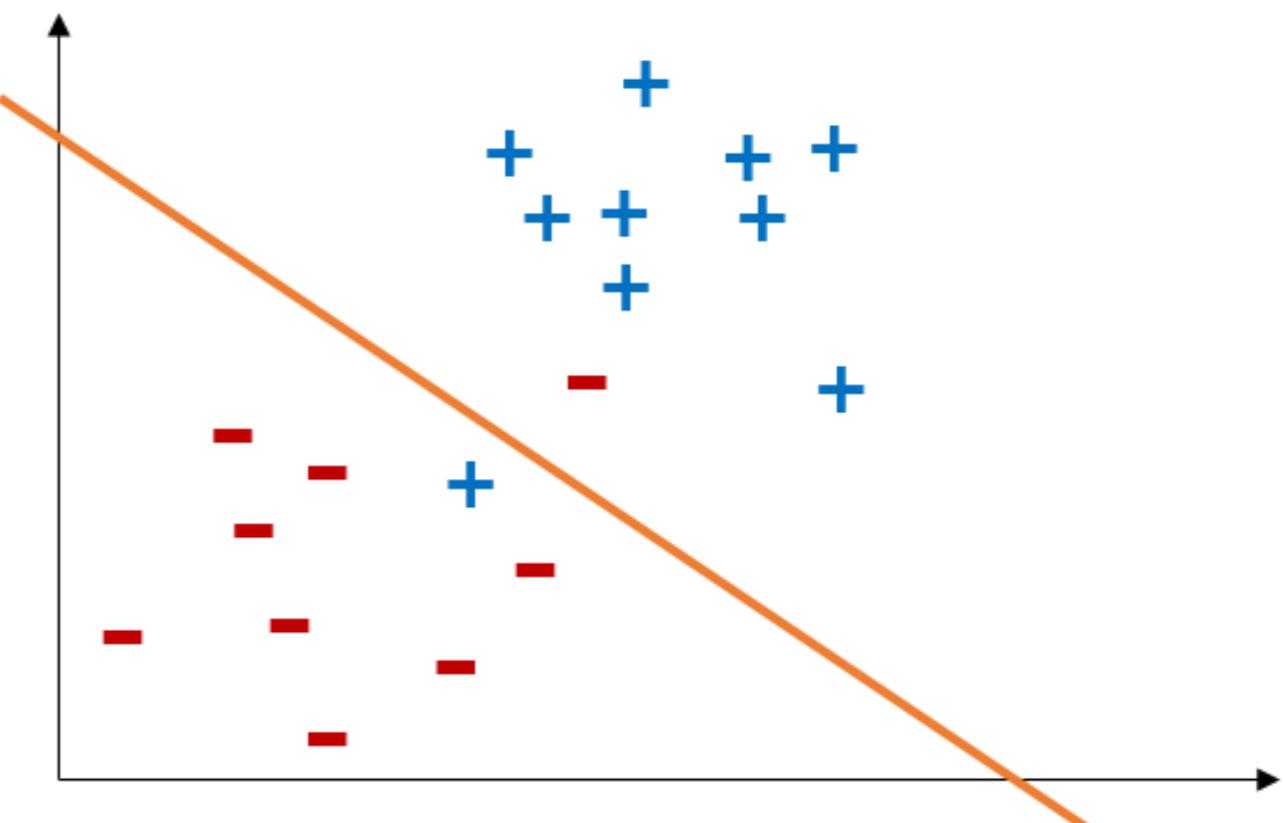
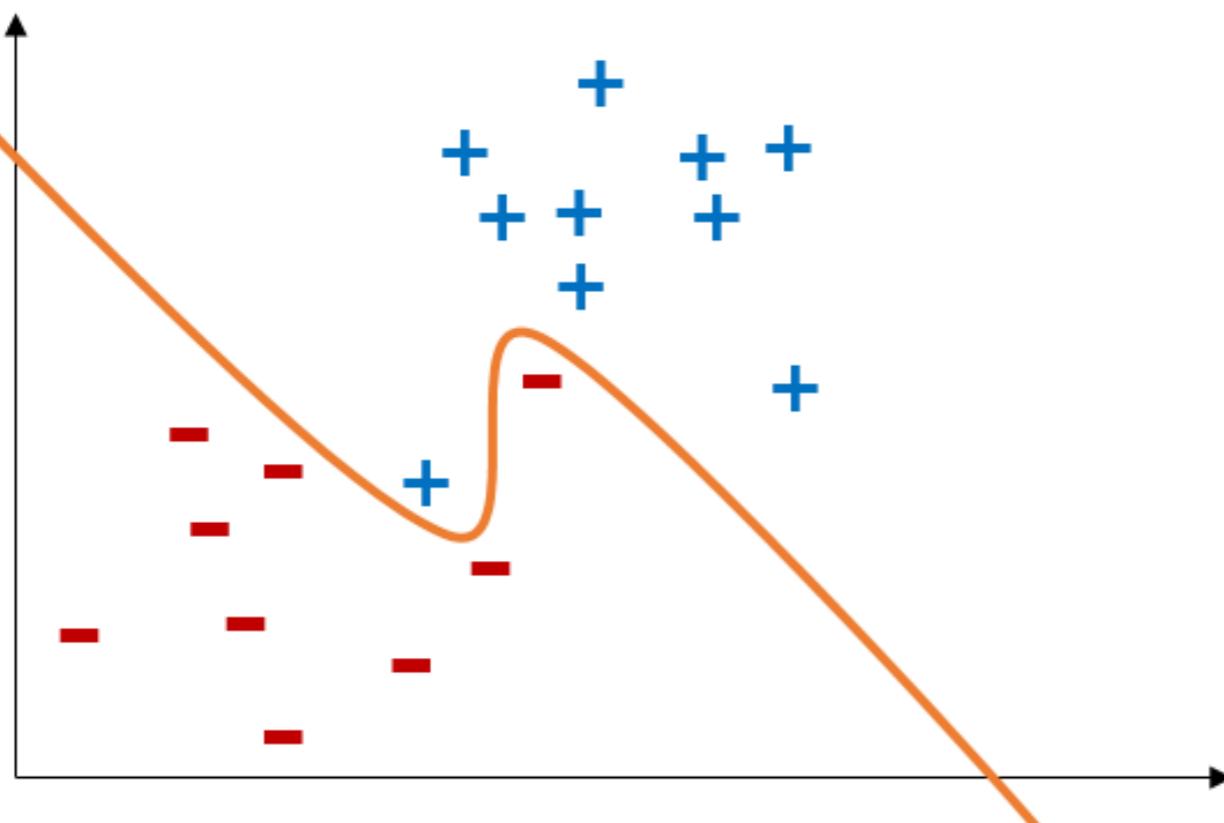


DEEP LEARNING

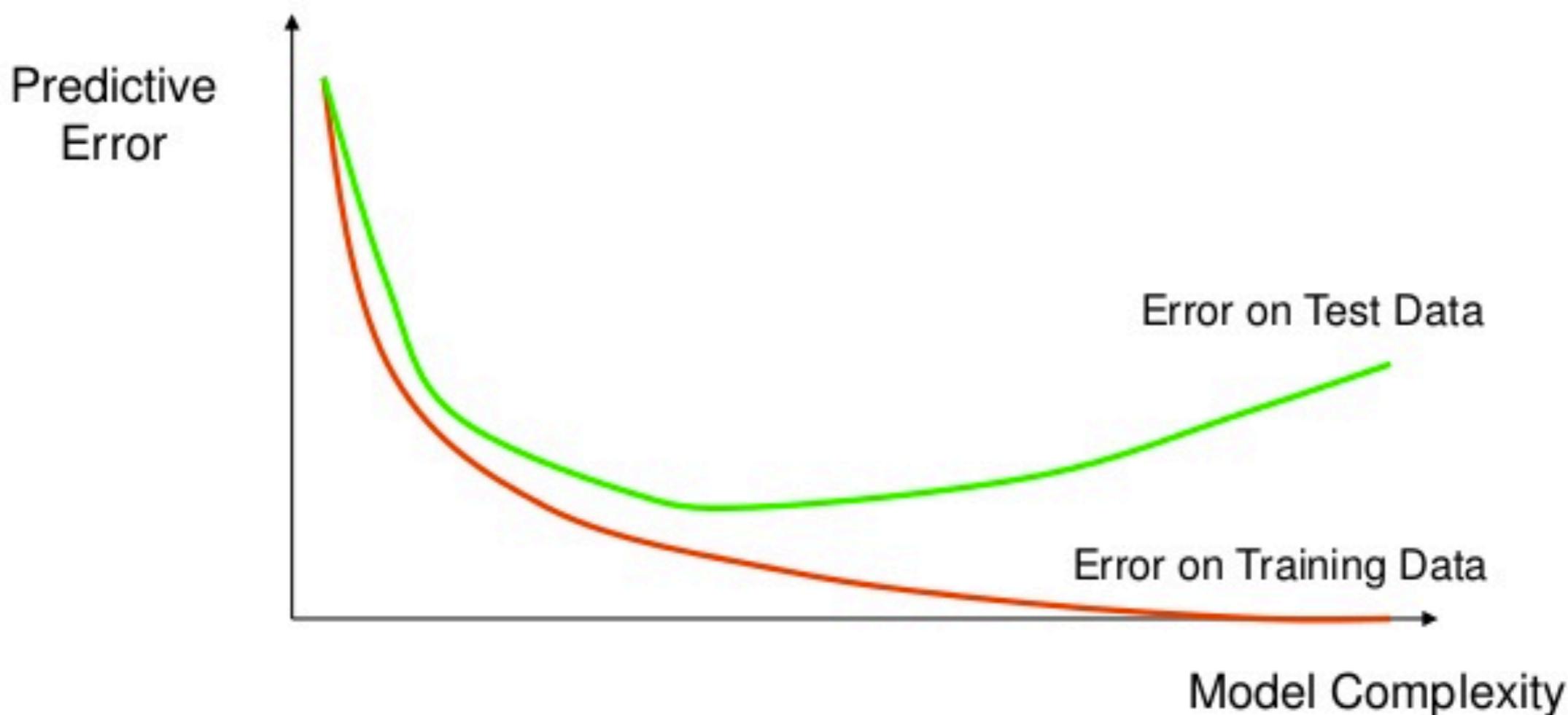
---

DIVERSE  
ALGORITHM

# OVERFITTING



## AM I OVERTFITTING?



- Very high accuracy on the training dataset (eg: 0.99)
- Poor accuracy on the test data set (0.85)

## SOLUTIONS FOR OVERFITTING

- ▶ **More training data**
- ▶ **Reduce the number of features**
- ▶ **Regularization**
- ▶ **Early stopping**
- ▶ **Dropout**

## REGULARIZATION

- Let's not have too big numbers in the weight

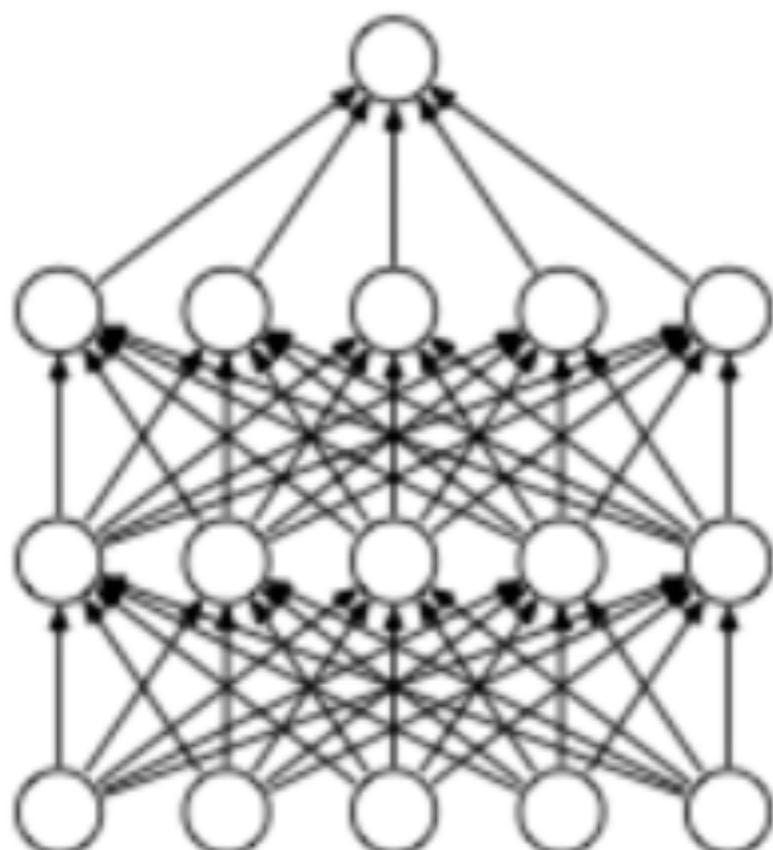
Regularization Strength

$$\text{cost} + \lambda \sum W^2$$

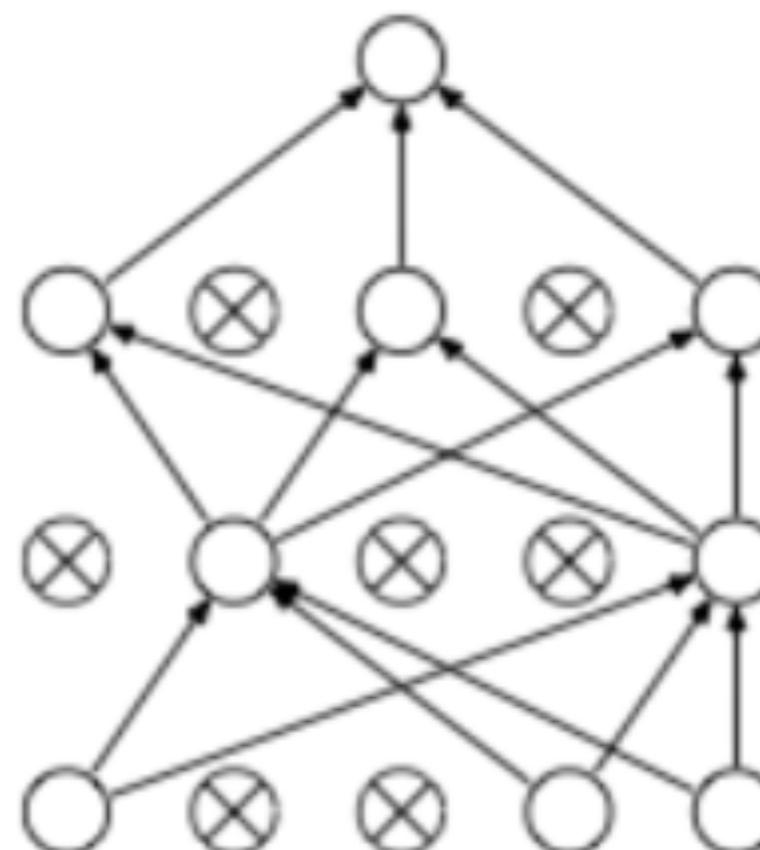
```
l2reg = 0.001 * tf.reduce_sum(tf.square(W))
```

## REGULARIZATION : DROPOUT

Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Srivastava et al. 2014]



(a) Standard Neural Net

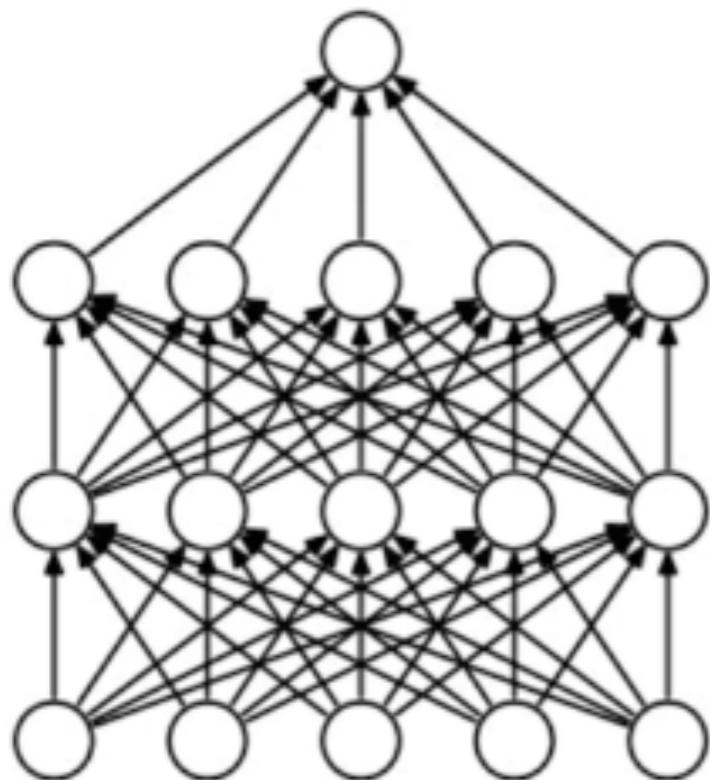


(b) After applying dropout.

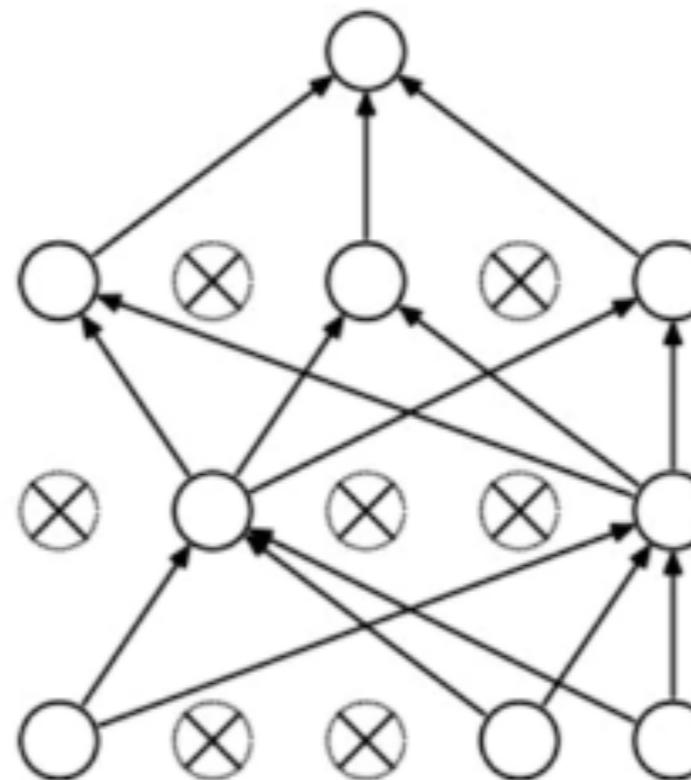
## REGULARIZATION : DROPOUT

### Regularization: Dropout

“randomly set some neurons to zero in the forward pass”



(a) Standard Neural Net



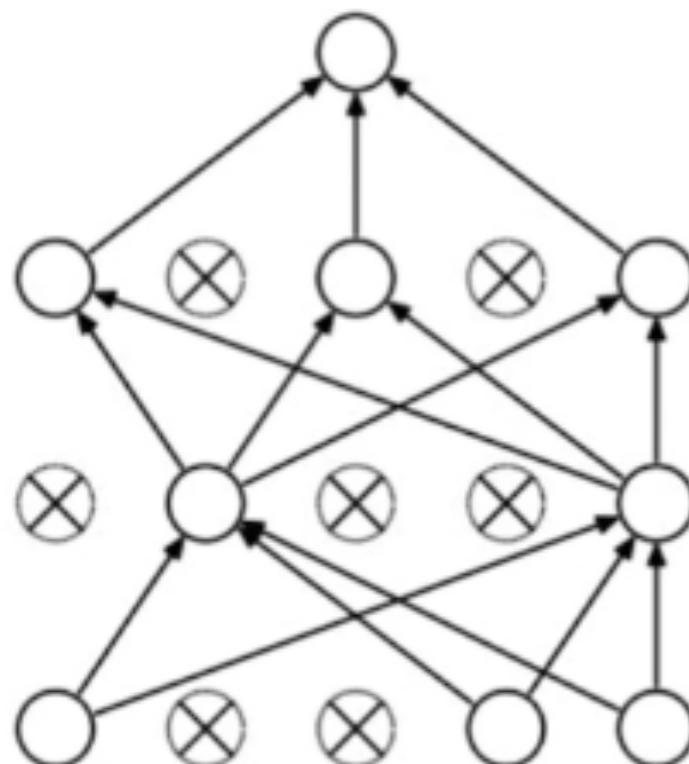
(b) After applying dropout.

[Srivastava et al., 2014]

## REGULARIZATION : DROPOUT

Waaaait a second...

How could this possibly be a good idea?



Forces the network to have a redundant representation.



## TENSORFLOW IMPLEMENTATION

```
dropout_rate = tf.placeholder("float")
_L1 = tf.nn.relu(tf.add(tf.matmul(X, W1), B1))
L1 = tf.nn.dropout(_L1, dropout_rate)
```

### TRAIN:

```
sess.run(optimizer, feed_dict={X: batch_xs, Y: batch_ys,
dropout_rate: 0.7})
```

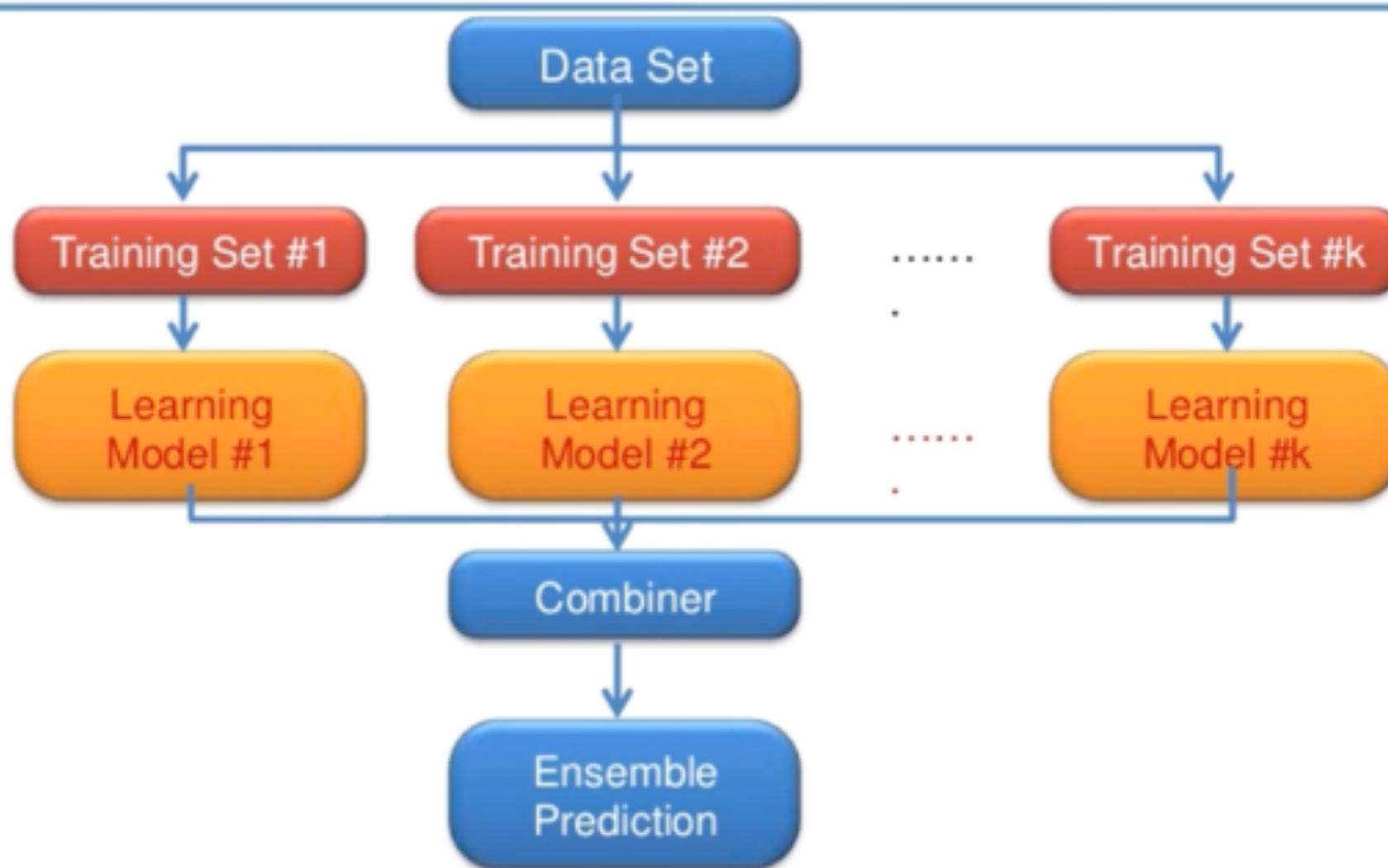
### EVALUATION:

```
print "Accuracy:", accuracy.eval({X: mnist.test.images, Y:
mnist.test.labels, dropout_rate: 1})
```

# ENSEMBLE

## What is Ensemble?

---

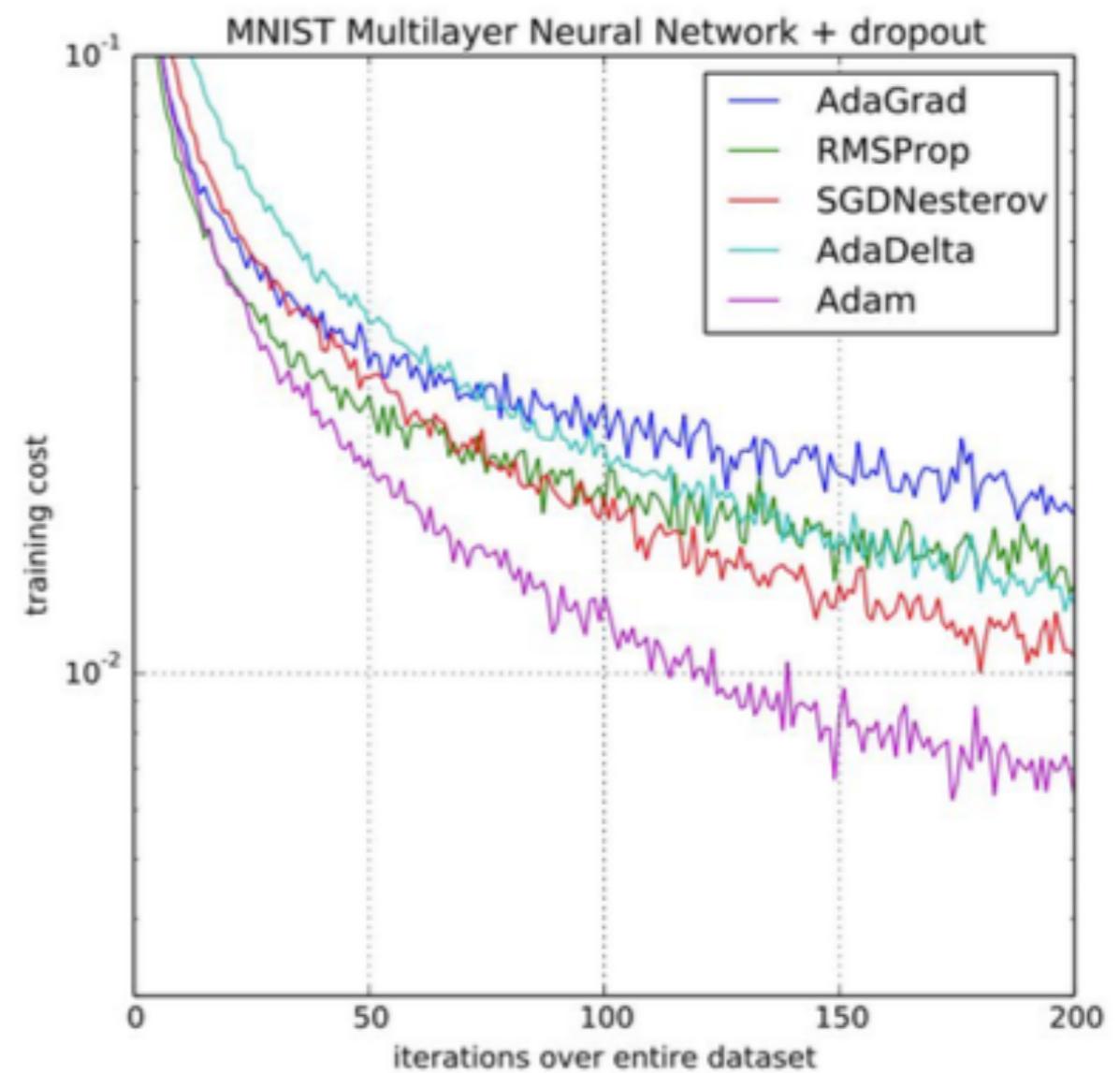
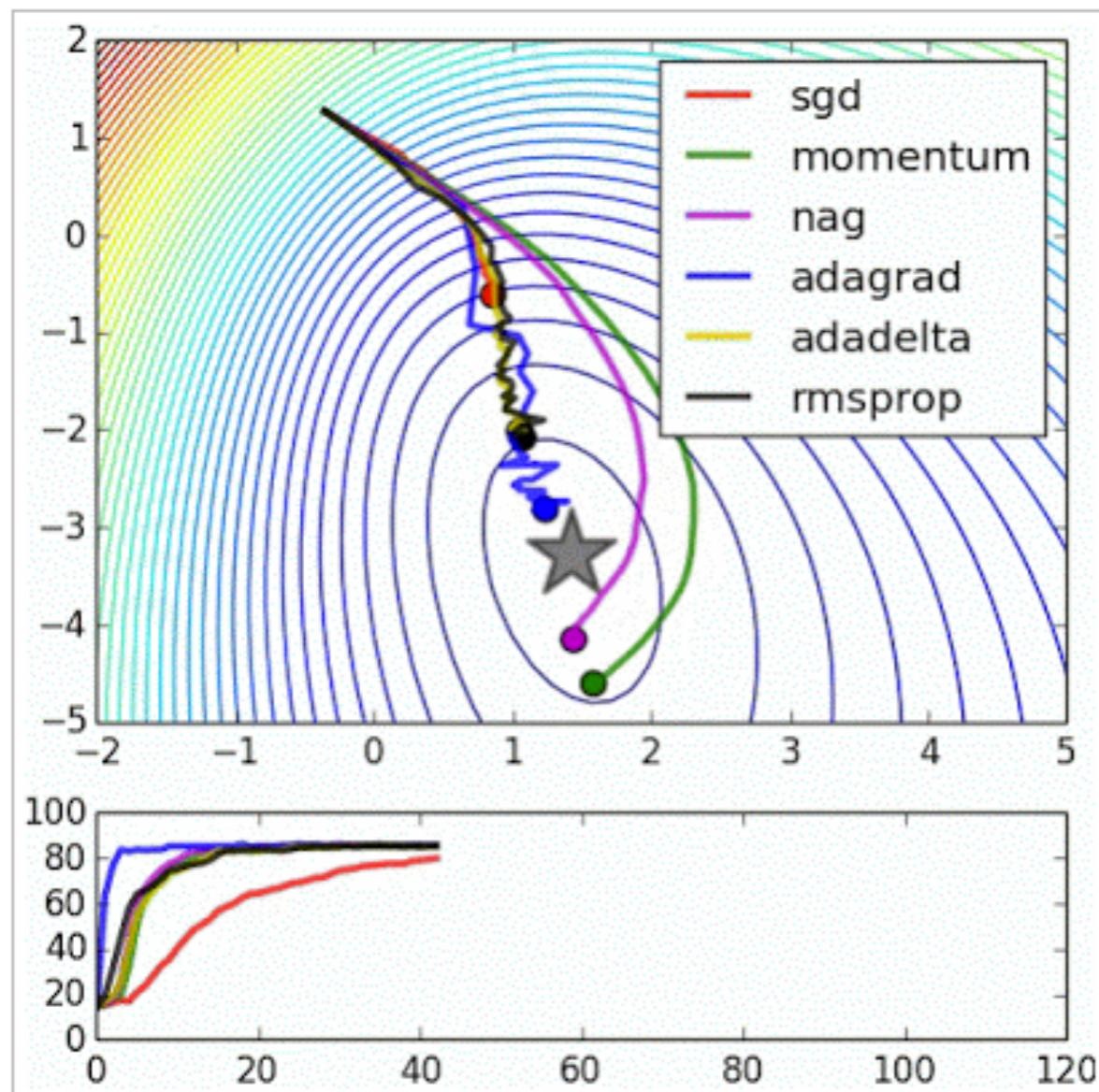


# OPTIMIZERS

```
train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

- `tf.train.AdadeltaOptimizer`
- `tf.train.AdagradOptimizer`
- `tf.train.AdagradDAOptimizer`
- `tf.train.MomentumOptimizer`
- `tf.train.AdamOptimizer`
- `tf.train.FtrlOptimizer`
- `tf.train.ProximalGradientDescentOptimizer`
- `tf.train.ProximalAdagradOptimizer`
- `tf.train.RMSPropOptimizer`

## OPTIMIZERS



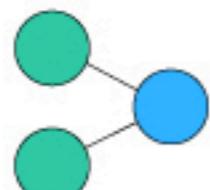
## ADAM OPTIMIZER

```
# define cost/Loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
                      logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

## PREDICTIONS SUMMARY

- Softmax VS Neural Nets for MNIST, 90% and 94.5%
- Xavier initialization: 97.8%
- Deep Neural Nets with Dropout: 98%
- Adam and other optimizers
- Exercise: Batch Normalization
  - [https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-6-mnist\\_nn\\_batchnorm.ipynb](https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-6-mnist_nn_batchnorm.ipynb)

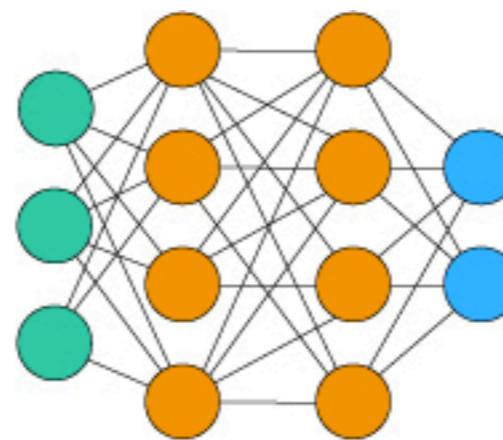
# NEURAL NETWORK LAYERS (1)



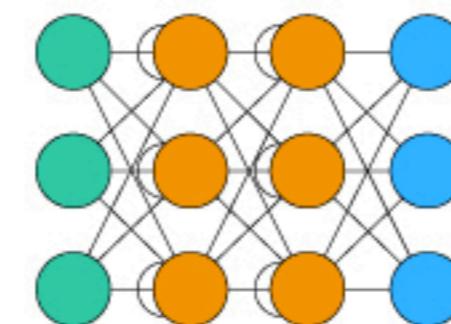
Single Layer  
Perceptron



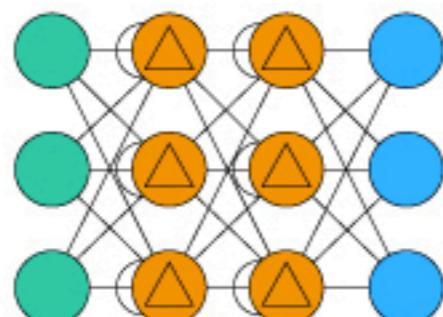
Radial Basis  
Network (RBN)



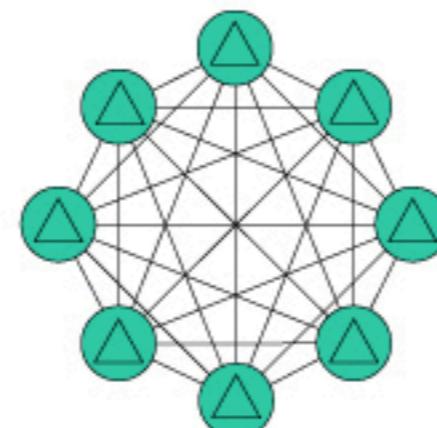
Multi Layer Perceptron



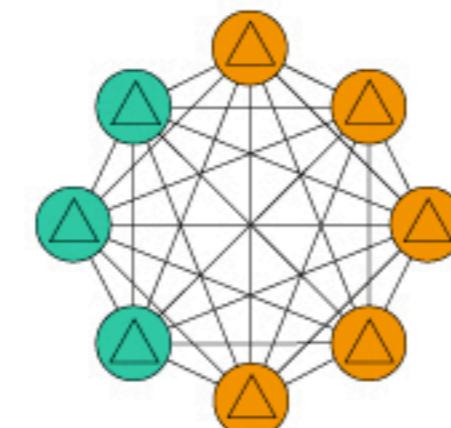
Recurrent Neural Network



LSTM Recurrent Neural  
Network



Hopfield Network



Boltzmann Machine

---

● Input Unit

○ Hidden Unit

△ Backfed Input Unit

● Output Unit

○△ Feedback with Memory Unit

△ Probabilistic Hidden Unit

## NEURAL NETWORK LAYERS (2)

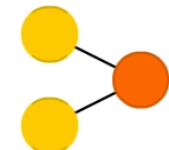
- (○) Backfed Input Cell
- (○) Input Cell
- (△) Noisy Input Cell
- (●) Hidden Cell
- (○) Probabilistic Hidden Cell
- (△) Spiking Hidden Cell
- (●) Output Cell
- (○) Match Input Output Cell
- (●) Recurrent Cell
- (○) Memory Cell
- (△) Different Memory Cell
- (●) Kernel
- (○) Convolution or Pool

*A mostly complete chart of*

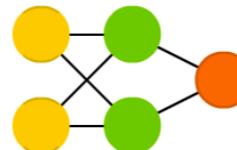
# Neural Networks

©2016 Fjodor van Veen - [asimovinstitute.org](http://asimovinstitute.org)

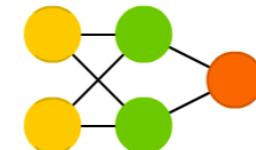
Perceptron (P)



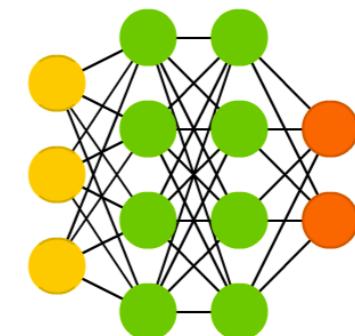
Feed Forward (FF)



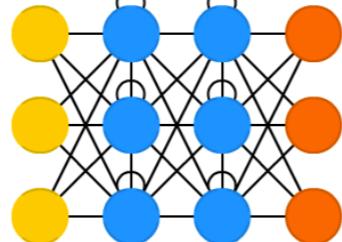
Radial Basis Network (RBF)



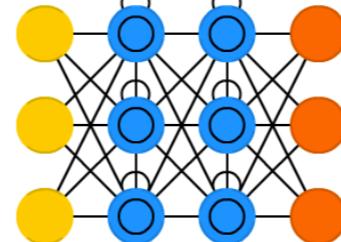
Deep Feed Forward (DFF)



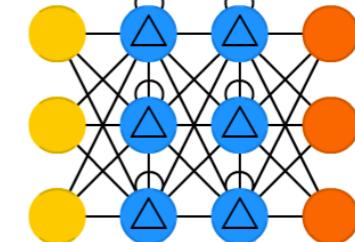
Recurrent Neural Network (RNN)



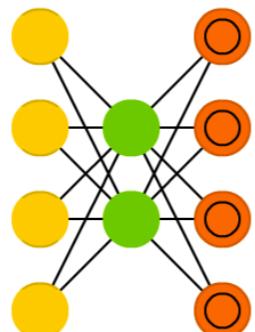
Long / Short Term Memory (LSTM)



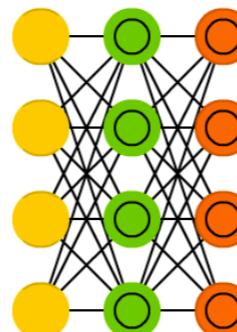
Gated Recurrent Unit (GRU)



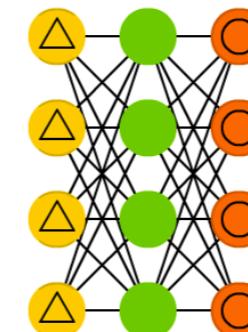
Auto Encoder (AE)



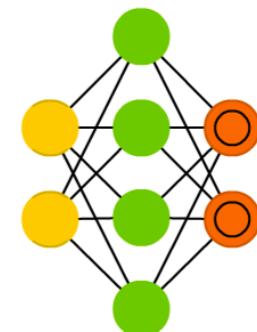
Variational AE (VAE)



Denoising AE (DAE)

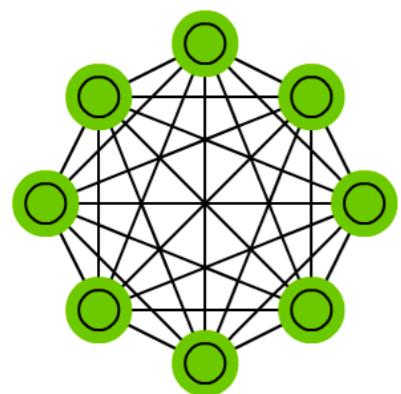


Sparse AE (SAE)

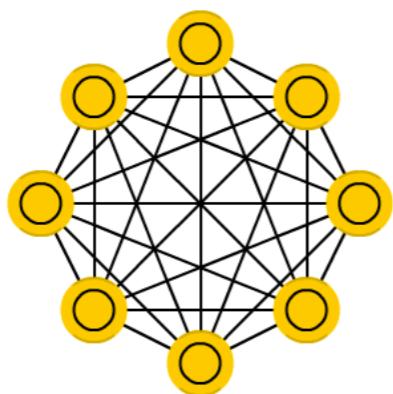


# NEURAL NETWORK LAYERS (3)

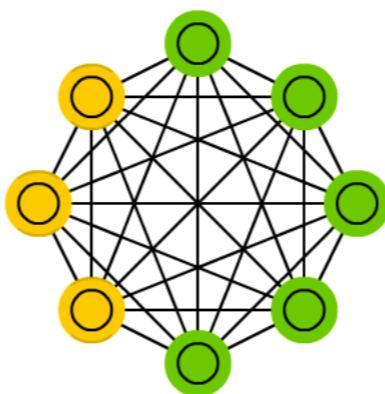
Markov Chain (MC)



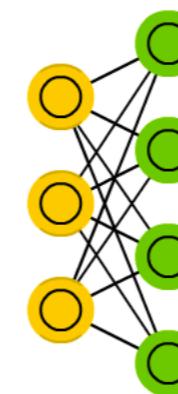
Hopfield Network (HN)



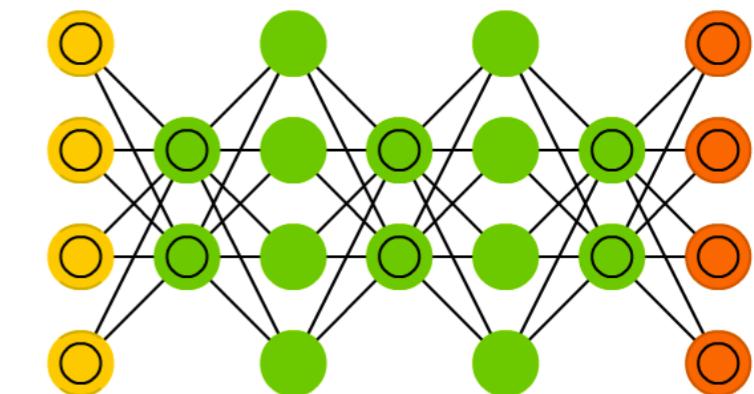
Boltzmann Machine (BM)



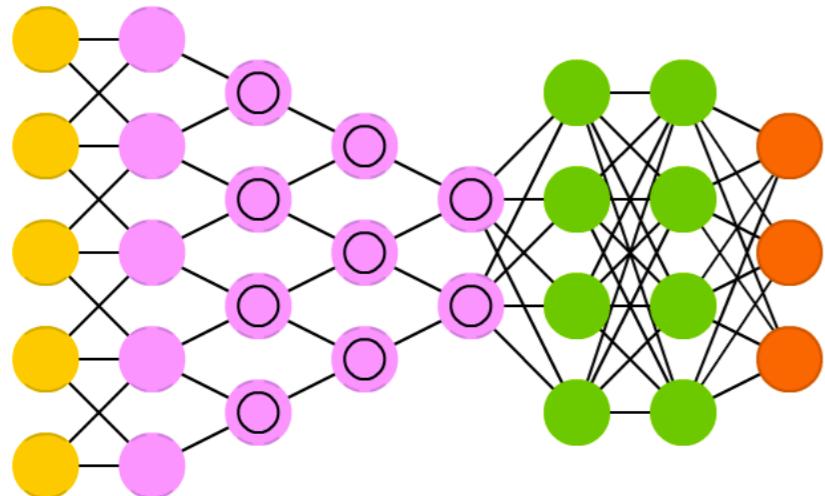
Restricted BM (RBM)



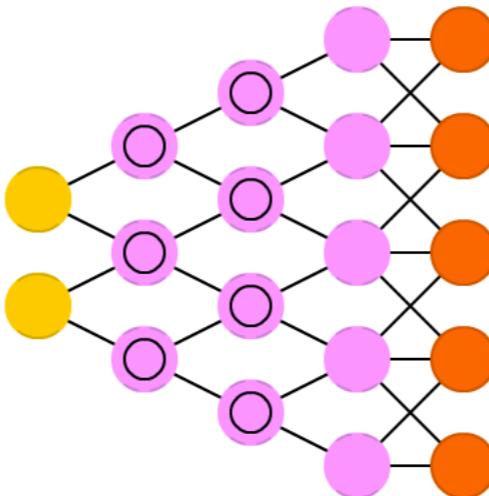
Deep Belief Network (DBN)



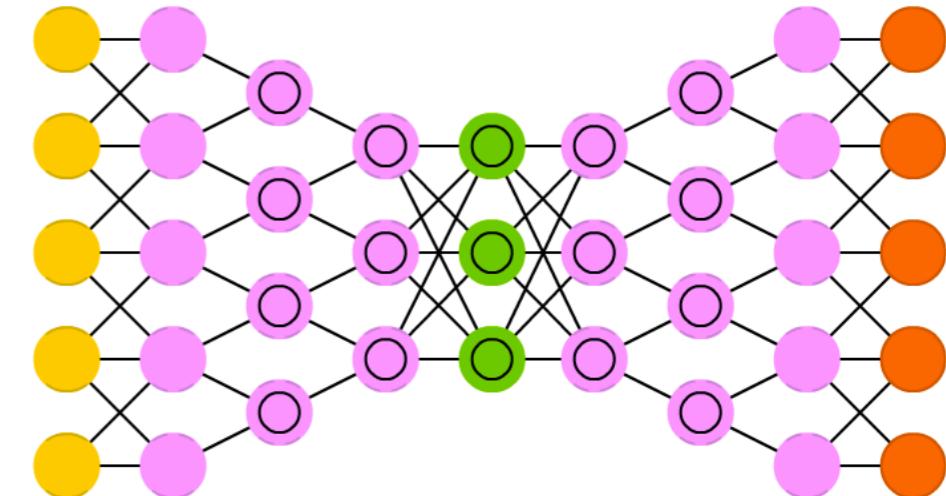
Deep Convolutional Network (DCN)



Deconvolutional Network (DN)

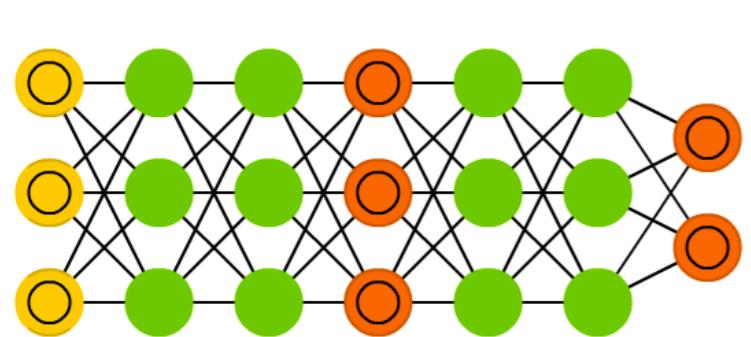


Deep Convolutional Inverse Graphics Network (DCIGN)

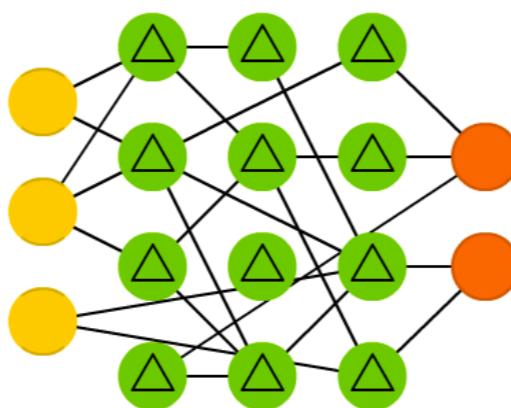


# NEURAL NETWORK LAYERS (4)

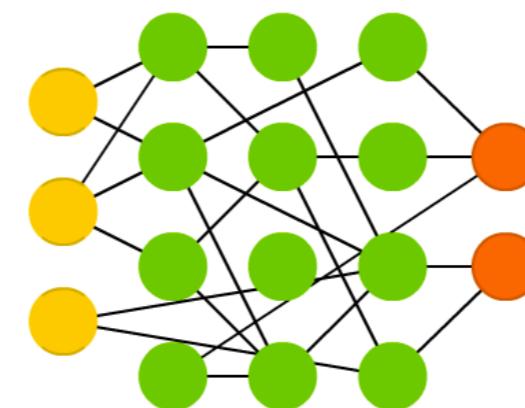
Generative Adversarial Network (GAN)



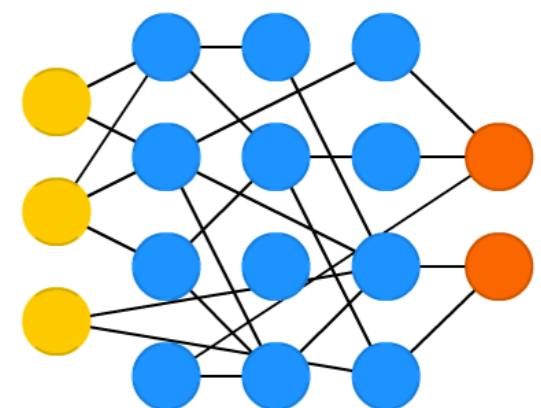
Liquid State Machine (LSM)



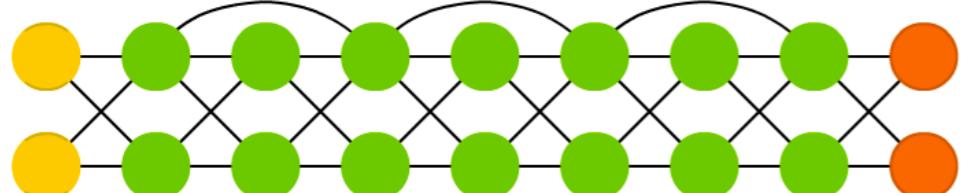
Extreme Learning Machine (ELM)



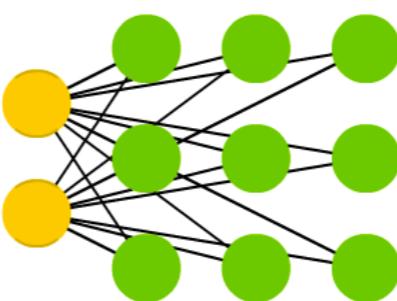
Echo State Network (ESN)



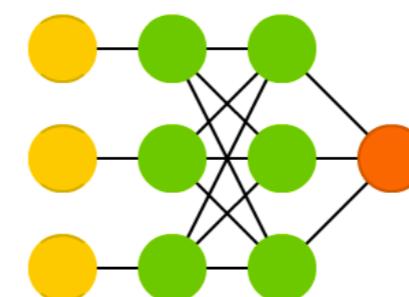
Deep Residual Network (DRN)



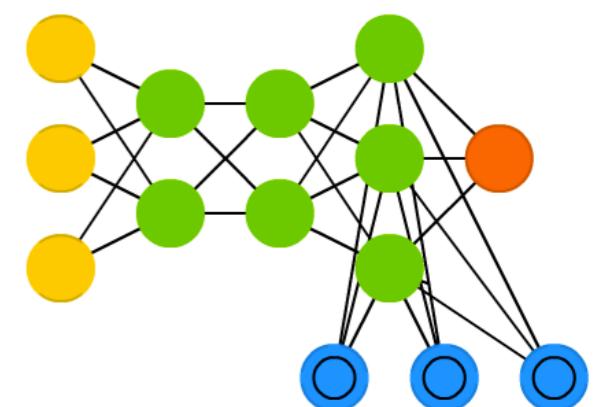
Kohonen Network (KN)



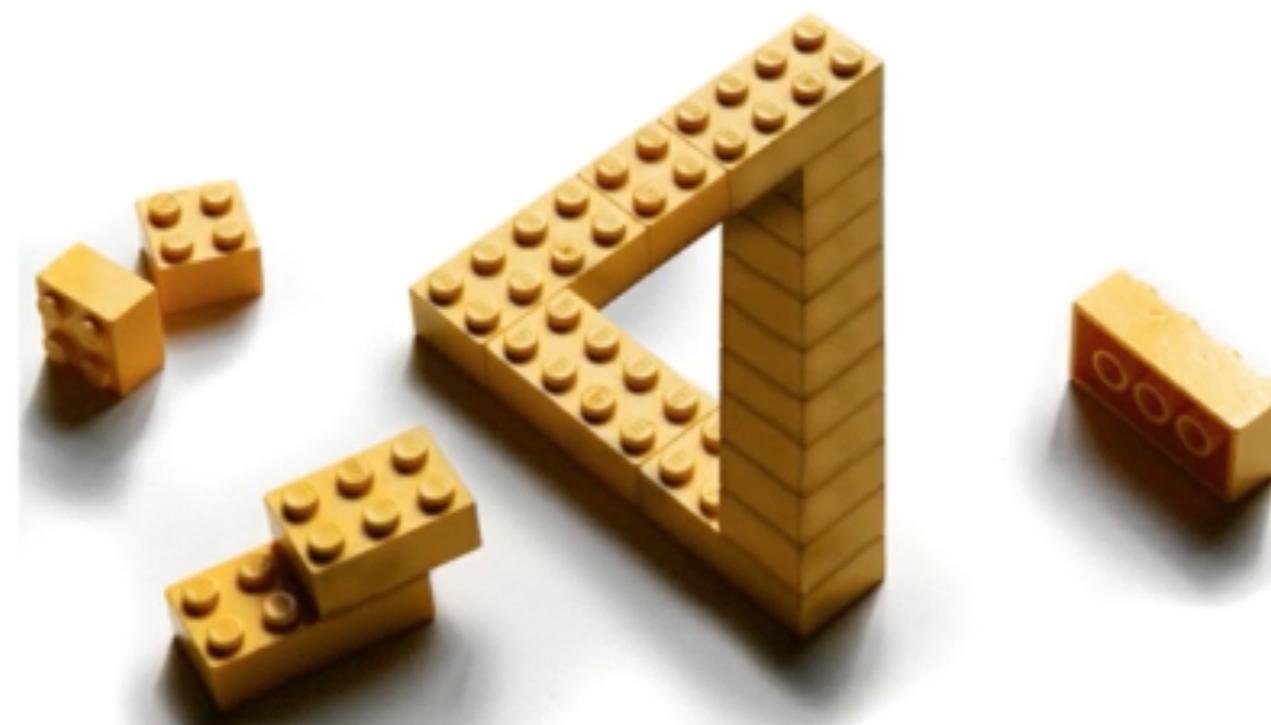
Support Vector Machine (SVM)



Neural Turing Machine (NTM)



# THE ONLY LIMIT IS YOUR IMAGINATION



<http://itchyi.squarespace.com/thelatest/2012/5/17/the-only-limit-is-your-imagination.html>



DEEP LEARNING

---

TENSORBOARD  
VISUALIZATION

## OLD FASHION : PRINT, PRINT, PRINT

```
9400 0.0151413 [array([[ 6.21692038,  6.05913448],  
[-6.33773184, -5.75189114]], dtype=float32), array([[ 9.93581772],  
[-9.43034935]], dtype=float32)]  
9500 0.014909 [array([[ 6.22498751,  6.07049847],  
[-6.34637976, -5.76352596]], dtype=float32), array([[ 9.96414757],  
[-9.45942593]], dtype=float32)]  
9600 0.0146836 [array([[ 6.23292685,  6.08166742],  
[-6.35489035, -5.77496052]], dtype=float32), array([[ 9.99207973],  
[-9.48807526]], dtype=float32)]  
9700 0.0144647 [array([[ 6.24074268,  6.09264851],  
[-6.36326933, -5.78619957]], dtype=float32), array([[ 10.01962471],  
[-9.51631165]], dtype=float32)]  
9800 0.0142521 [array([[ 6.24843407,  6.10344648],  
[-6.37151814, -5.79724932]], dtype=float32), array([[ 10.04679298],  
[-9.54414845]], dtype=float32)]  
9900 0.0140456 [array([[ 6.25601053,  6.11406422],  
[-6.3796401 , -5.80811596]], dtype=float32), array([[ 10.07359505],  
[-9.57159519]], dtype=float32)]  
10000 0.0138448 [array([[ 6.26347113,  6.12451124],  
[-6.38764334, -5.81880617]], dtype=float32), array([[ 10.10004139],  
[-9.59866238]], dtype=float32)]
```

# TENSORBOARD VISUALIZATION

## NEW WAY!



# 5 STEPS OF USING TENSORBOARD

- 1 From TF graph, decide which tensors you want to log

```
w2_hist = tf.summary.histogram("weights2", W2)
cost_summ = tf.summary.scalar("cost", cost)
```

- 2 Merge all summaries

```
summary = tf.summary.merge_all()
```

- 3 Create writer and add graph

```
# Create summary writer
writer = tf.summary.FileWriter('./logs')
writer.add_graph(sess.graph)
```

- 4 Run summary merge and add\_summary

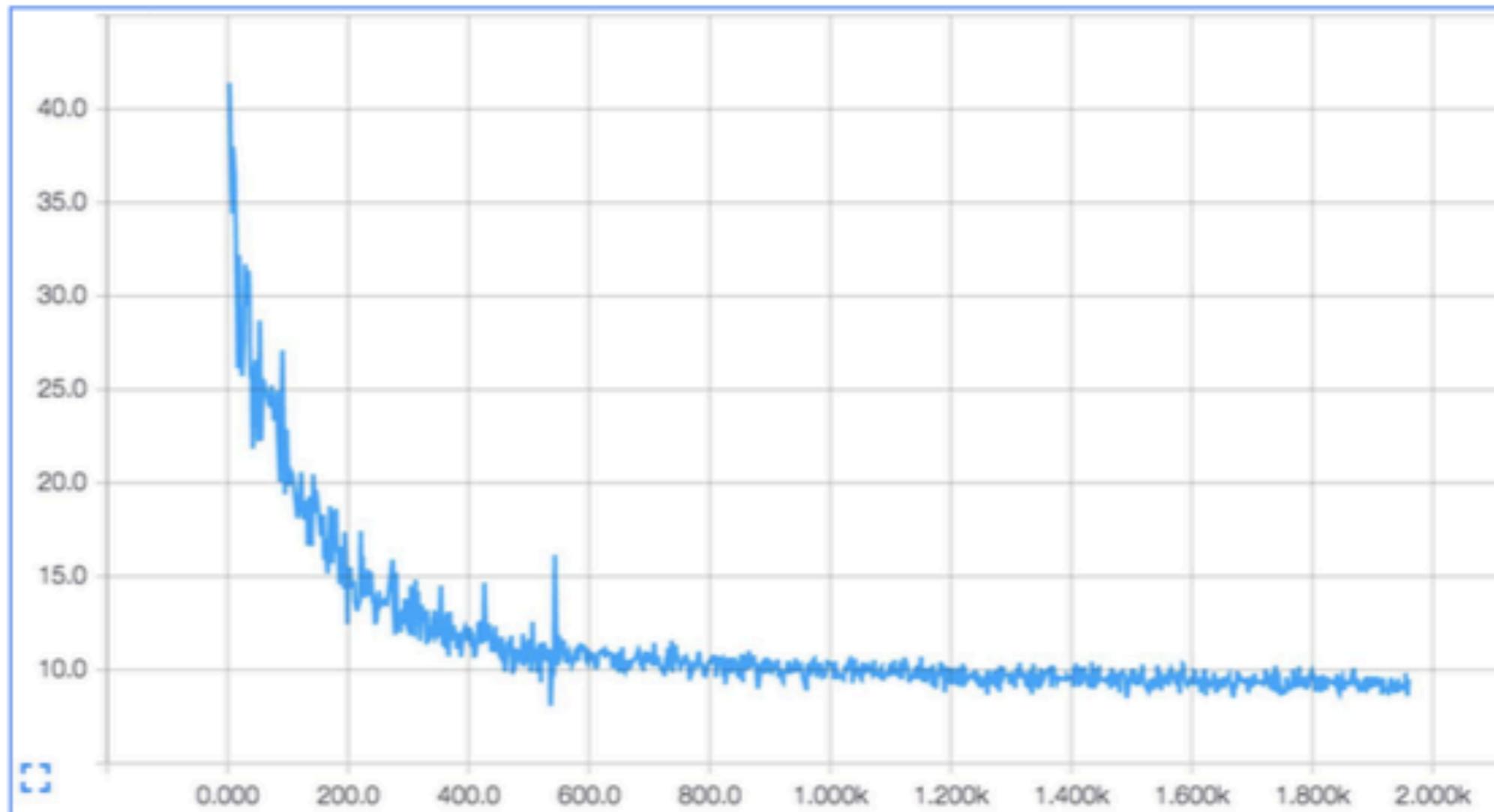
```
s, _ = sess.run([summary, optimizer], feed_dict=feed_dict)
writer.add_summary(s, global_step=global_step)
```

- 5 Launch TensorBoard

```
tensorboard --logdir=./logs
```

# SCALAR TENSORS

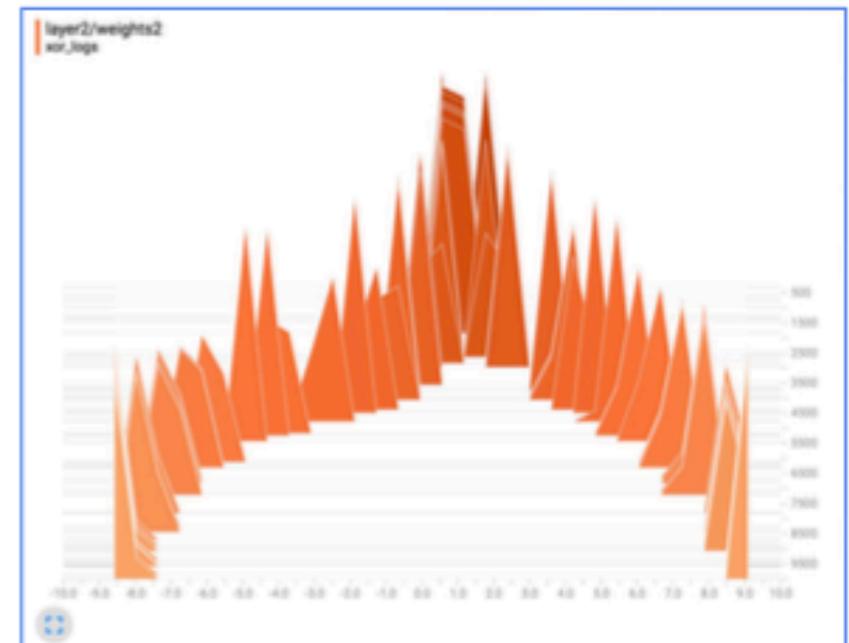
```
cost_summ = tf.summary.scalar("cost", cost)
```



# HISTOGRAM (MULTI-DIMENSIONAL TENSORS)

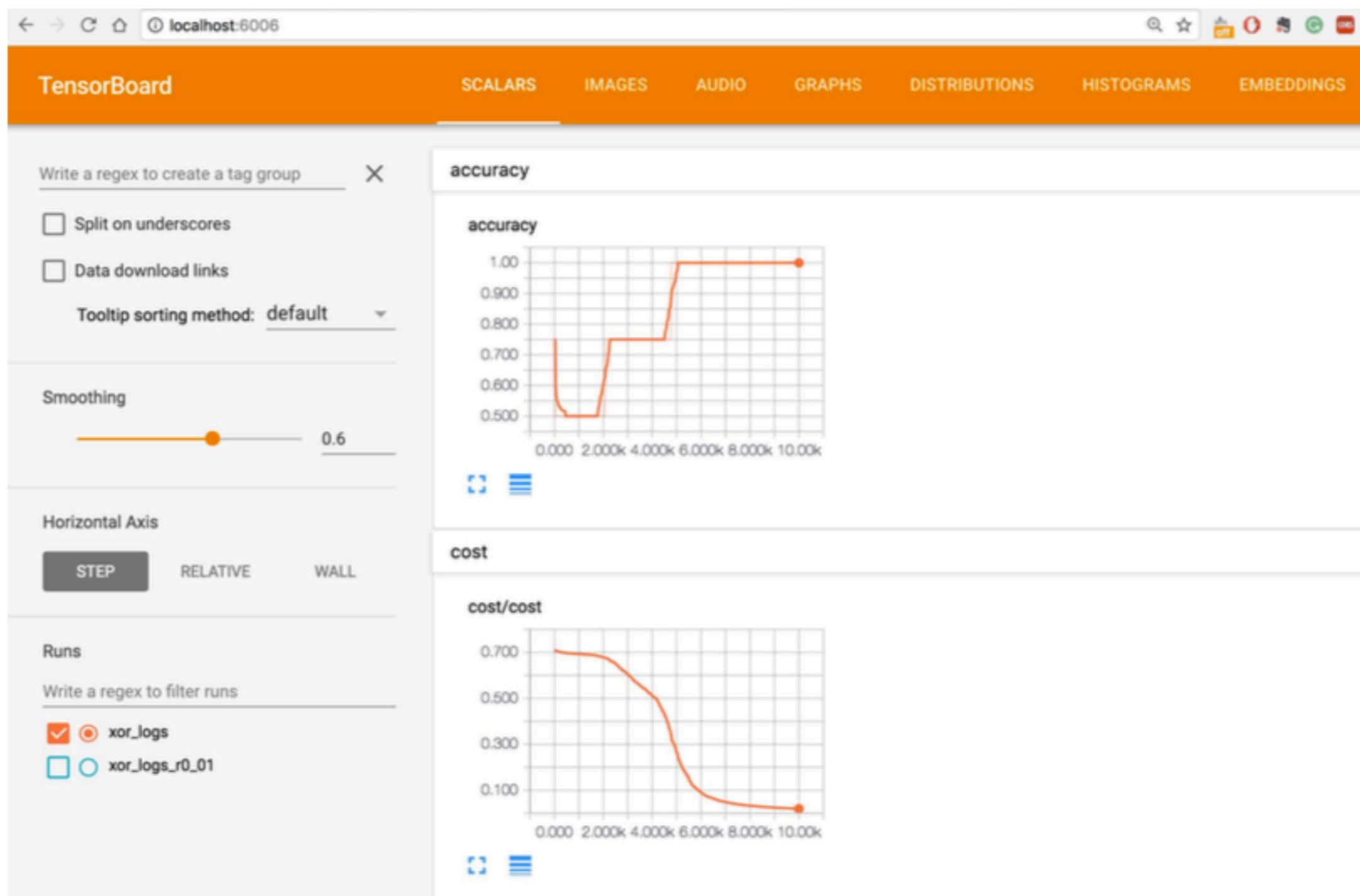
```
w2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, w2) + b2)
```

```
w2_hist = tf.summary.histogram("weights2", w2)
b2_hist = tf.summary.histogram("biases2", b2)
hypothesis_hist = tf.summary.histogram("hypothesis", hypothesis)
```

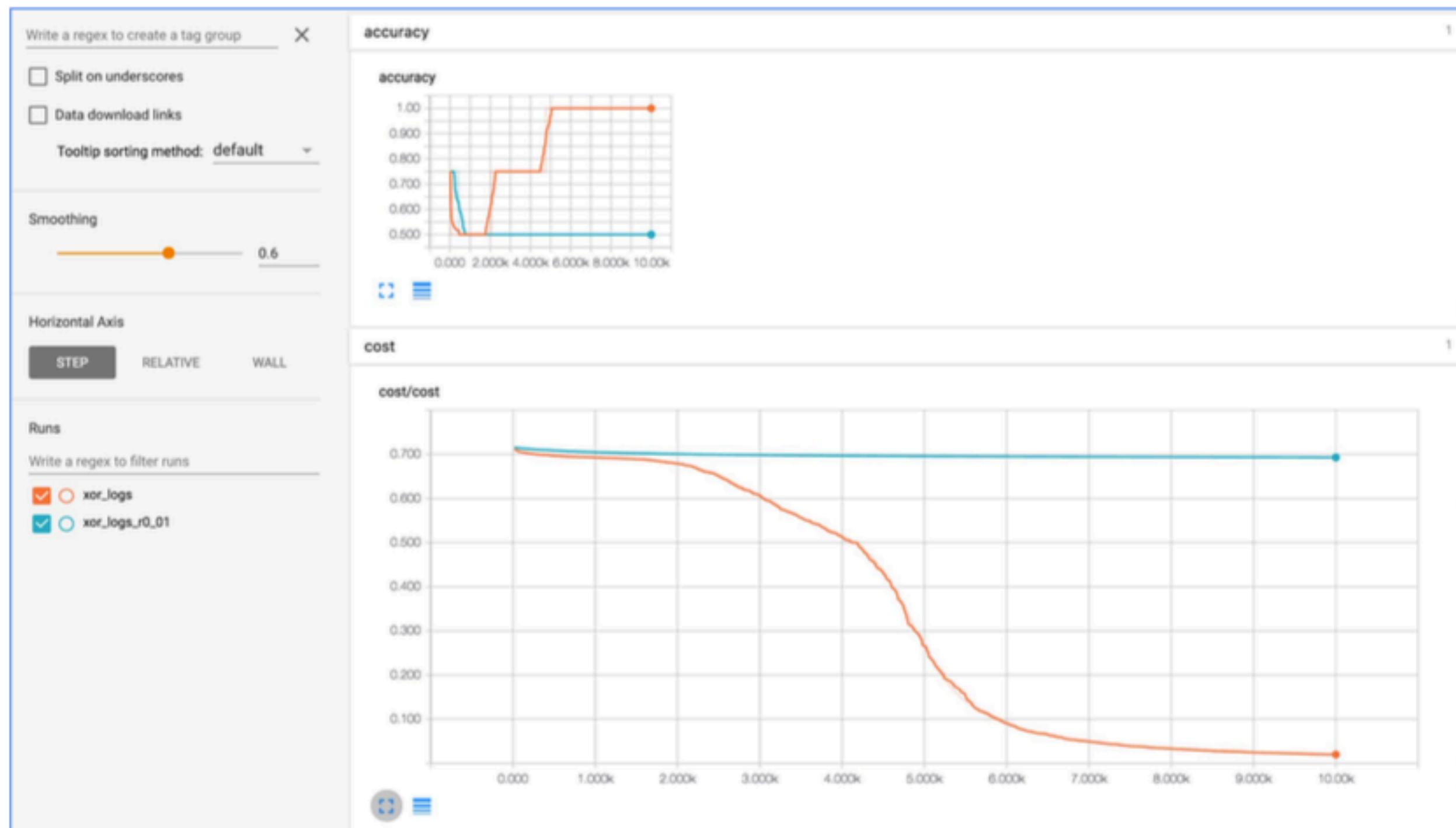


# TENSORBOARD VISUALIZATION

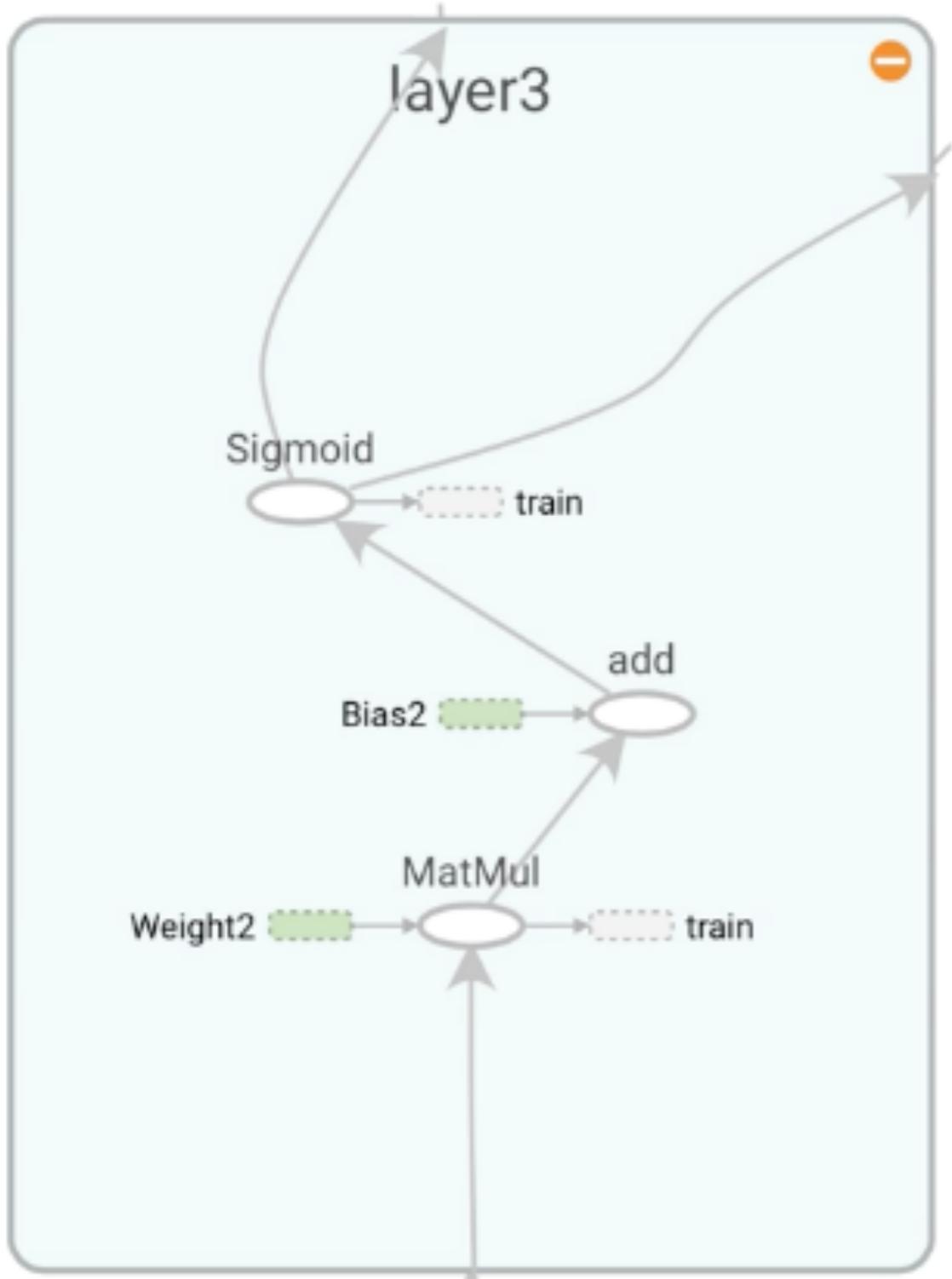
## 실행 결과



## 실행 결과 : MULTIPLE RUNS



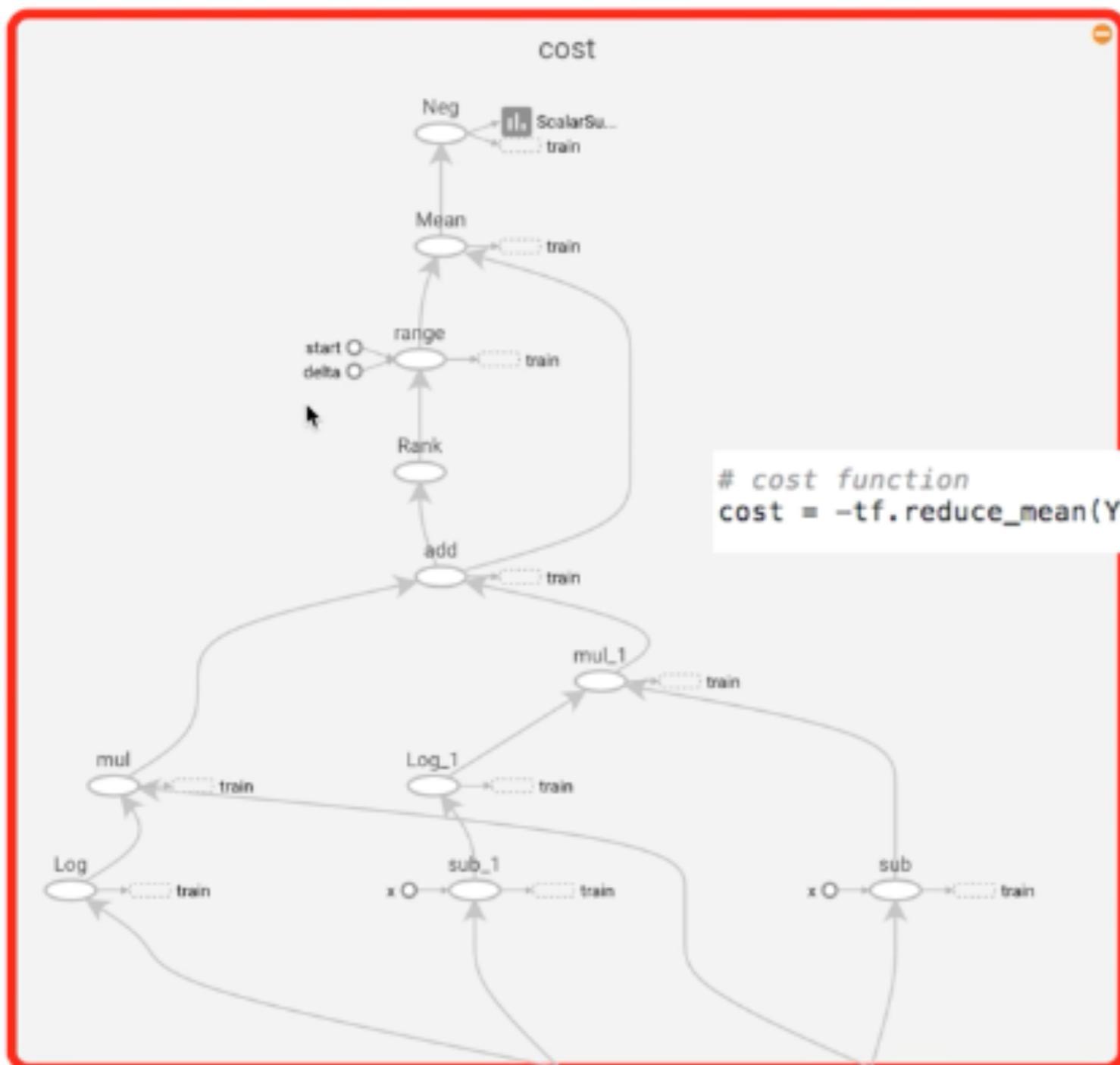
# BACK PROPAGATION



Back propagation in  
TensorFlow  
TensorBoard

```
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

# BACK PROPAGATION



## Back propagation in TensorFlow TensorBoard



DEEP LEARNING

---

SOURCE CODES

01

```
# 문제 1
# y에서 yhat까지의 비용을 계산하는 cost 함수를 만드세요.

def costToYhat(y, yhat):
    c = 0
    for i in range(len(y)):
        c += (y[i] - yhat[i]) ** 2
    return c / len(y)
```

# 문제 2-1

# w가 -3부터 5까지 변할 때의 cost를 그래프에 그려보세요.

```
def cost(x, y, w):
    c = 0
    for i in range(len(y)):
        yhat = w*x[i]
        c += (y[i] - yhat) ** 2
    return c / len(y)
```

```
x = [1, 2, 3]
```

```
y = [1, 3, 5]
```

# 문제 2-2

# w가 -3부터 5까지 변할 때의 cost를 그래프에 그려보세요.

```
xx, yy = [], []
for w in range(-30, 50):
    c = cost(x, y, w/10, 0)
    xx.append(w/10)
    yy.append(c)

import matplotlib.pyplot as plt

plt.plot(xx, yy, 'ro')
plt.show()
```

# 문제 3-1

# Gradient Descent Algorithm을 함수로 구현해 보세요.

```
def gradientDescent(x, y, w):
    s = 0
    for i in range(len(x)):
        hx = w * x[i]
        s += (hx - y[i]) * x[i]
    return s / len(x)
```

```
# 문제 3-2
# 앞에서 만든 gradientDescent 함수를 사용해서
# w가 1.0에 가까워지는 모습을 그래프로 표현해 보세요.

x, y = [1, 2, 3], [1, 2, 3]
w, learning_rate = -10, 0.1
for i in range(10):
    grad = gradientDescent(x, y, w)
    w -= grad * learning_rate
    plt.plot([0, 5], [0, 5*w])

plt.plot(x, y, 'ro')
plt.xlim(0, 5)
plt.ylim(0, 5)
plt.show()
```

```
# 문제 4
# 앞에서 만든 cost와 gradientDescent 함수를 사용해서
# x값이 5와 7일 때의 결과를 예측해 주세요.

x, y = [1, 2, 3], [1, 2, 3]
w = 10
for i in range(1000):
    if i%20 == 0:
        c = cost(x, y, w)
        print('{:3} : {}'.format(i, c))

grad = gradientDescent(x, y, w)
w -= grad * 0.1

print(w**5, w**7)
```

# 문제 5

# 텐서플로우에서 정의한 상수를 출력하는 함수를 만들어 보세요.

```
def showConstant(a):
    sess = tf.InteractiveSession()
    print(a.eval())
    sess.close()
```

# 문제 6

# 텐서플로우에서 정의한 변수를 출력하는 함수를 만들어 보세요.

```
def showVariable(a):
    sess = tf.InteractiveSession()
    a.initializer.run()
    print(a.eval())
    sess.close()
```

# 문제 7

# 텐서플로우에서 정의한 연산을 출력하는 함수를 만들어 보세요.

```
def showOperation(op):
    # sess = tf.InteractiveSession()
    # op.initializer.run()

    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    print(sess.run(op))
    sess.close()
```

# 문제 8

# 텐서플로우에서 정의한 수식을 출력하는 함수를 만들어 보세요.

```
def showEquation(x, w, b):
```

```
    xt = tf.constant(x)
```

```
    wt = tf.constant(w)
```

```
    bt = tf.constant(b)
```

```
    eq = tf.add(tf.mul(wt, xt), bt)
```

```
    sess = tf.Session()
```

```
    sess.run(tf.global_variables_initializer())
```

```
    print(sess.run(eq))
```

```
    sess.close()
```

# 문제 9

# placeholder 버전으로 덧셈 연산을 구현해 보세요.

```
import tensorflow as tf
```

```
aa = tf.placeholder(tf.int32)
```

```
bb = tf.placeholder(tf.int32)
```

```
add = tf.add(aa, bb)
```

```
sess = tf.Session()
```

```
sess.run(tf.global_variables_initializer())
```

```
print(sess.run(add, feed_dict={aa: 2, bb: 9}))
```

```
sess.close()
```

```
# 문제 10
# 다음 공식을 placeholder 버전으로 구현해 보세요.

# h(x) = 3x + 5

x = tf.placeholder(tf.int32)
op = tf.sub(tf.mul(3, x), 5)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

print(sess.run(op, feed_dict={x: 3}))
print(sess.run(op, feed_dict={x: 7}))
sess.close()
```

```
# 문제 11-1
# placeholder를 사용해서 특정 단을 출력하는 함수를 만드세요.

def show99(dan):
    left = tf.placeholder(tf.int32)
    rite = tf.placeholder(tf.int32)
    calc = tf.mul(left, rite)
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())

    for i in range(1, 10):
        r=sess.run(calc,feed_dict={left:i, rite:dan})
        print('{}x{}={:2}'.format(i, dan, r))
    sess.close()
```

```
# 문제 11-2
# placeholder를 사용해서 특정 단을 출력하는 함수를 만드세요.

def show99(dan):
    left = tf.placeholder(tf.int32)
    rite = tf.constant(dan)
    calc = tf.mul(left, rite)
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())

    for i in range(1, 10):
        r = sess.run(calc, feed_dict = {left: i})
        print('{0}x{1}={2}'.format(i, dan, r))
    sess.close()
```



DEEP LEARNING

---

SOURCE CODES

02

```
# 문제 12-1
# Linear Regression 알고리즘을 구현해 보세요.

x, y = [1, 2, 3], [1, 2, 3]

w = tf.Variable(tf.random_uniform([1], -1, 1))
b = tf.Variable(tf.random_uniform([1], -1, 1))

hypothesis = tf.add(tf.mul(w, x), b)          # wx + b
cost = tf.reduce_mean(tf.square(hypothesis-y))
learning_rate = tf.Variable(0.1)
optimizer =
tf.train.GradientDescentOptimizer(learning_rate)
train = optimizer.minimize(cost)
```

# 문제 12-2

# Linear Regression 알고리즘을 구현해 보세요.

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train)
    if i%20 == 0:
        print(i, sess.run(cost), sess.run(w), sess.run(b))

sess.close()
```

```
# 문제 13-1  
# placeholder 버전으로 수정하고, 5와 7일 때의 결과를 알려주세요.  
  
xx, yy = [1, 2, 3], [1, 2, 3]  
  
x = tf.placeholder(tf.float32)  
y = tf.placeholder(tf.float32)  
  
w = tf.Variable(tf.random_uniform([1], -1, 1))  
b = tf.Variable(tf.random_uniform([1], -1, 1))  
  
hypothesis = tf.add(tf.mul(w, x), b)           # wx + b  
cost = tf.reduce_mean(tf.square(hypothesis-y))  
learning_rate = tf.Variable(0.1)
```

```
# 문제 13-2
# placeholder 버전으로 수정하고, 5와 7일 때의 결과를 알려주세요.

optimizer =
    tf.train.GradientDescentOptimizer(learning_rate)
train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(i, sess.run(cost,
                           feed_dict={x: xx, y: yy}),
```

```
# 문제 13-3  
# placeholder 버전으로 수정하고, 5와 7일 때의 결과를 알려주세요.  
ww, bb = sess.run(w), sess.run(b)  
  
print(type(ww))  
print(ww, bb)  
print(ww*5 + bb)  
print(ww*7 + bb)  
  
print(sess.run(hypothesis, feed_dict={x: 5}))  
print(sess.run(hypothesis, feed_dict={x: 7}))  
  
sess.close()
```

# 문제 14-1

# 이전 문제를 simple\_4.txt 파일에서 읽어오는 코드로 수정해 주세요.

**#simple\_4.txt**

x, y

1, 1

2, 2

3, 3

**import numpy as np**

**xy = np.loadtxt('Data/simple\_4.txt', unpack=True,  
delimiter=', ', skiprows=1)**

**xx = xy[0] # [1, 2, 3]**

**yy = xy[1] # [1, 2, 3]**

```
# 문제 14-2
# 이전 문제를 simple_4.txt 파일에서 읽어오는 코드로 수정해 주세요.

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_uniform([1], -1, 1))
b = tf.Variable(tf.random_uniform([1], -1, 1))

hypothesis = tf.add(tf.mul(w, x), b)                      # wx + b
cost = tf.reduce_mean(tf.square(hypothesis-y))
learning_rate = tf.Variable(0.1)

optimizer =
tf.train.GradientDescentOptimizer(learning_rate)
```

```
# 문제 14-3
# 이전 문제를 simple_4.txt 파일에서 읽어오는 코드로 수정해 주세요.

train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(i,sess.run(cost,feed_dict={x:xx,y:yy}),
              sess.run(w), sess.run(b))
```

```
# 문제 14-4
# 이전 문제를 simple_4.txt 파일에서 읽어오는 코드로 수정해 주세요.

ww, bb = sess.run(w), sess.run(b)

print(type(ww))
print(ww, bb)
print(ww*5 + bb)
print(ww*7 + bb)

print(sess.run(hypothesis, feed_dict={x: 5}))
print(sess.run(hypothesis, feed_dict={x: 7}))

sess.close()
```

```
# 문제 15-1  
# cars.csv 파일을 모델링해서  
# 속도가 30과 50일 때의 제동거리를 예측해 주세요.  
  
xy = np.loadtxt('Data/cars.csv', unpack=True,  
                 delimiter=',')  
  
xx = xy[0]          # 속도  
yy = xy[1]          # 제동거리  
  
x = tf.placeholder(tf.float32)  
y = tf.placeholder(tf.float32)  
  
w = tf.Variable(tf.random_uniform([1], -1, 1))  
b = tf.Variable(tf.random_uniform([1], -1, 1))
```

```
# 문제 15-2  
# cars.csv 파일을 모델링해서  
# 속도가 30과 50일 때의 제동거리를 예측해 주세요.  
hypothesis = tf.add(tf.mul(w, x), b) # wx + b  
cost = tf.reduce_mean(tf.square(hypothesis-y))  
learning_rate = tf.Variable(0.0035)  
  
optimizer =  
tf.train.GradientDescentOptimizer(learning_rate)  
train = optimizer.minimize(cost)  
  
sess = tf.Session()  
sess.run(tf.global_variables_initializer())
```

```
# 문제 15-3
# cars.csv 파일을 모델링해서
# 속도가 30과 50일 때의 제동거리를 예측해 주세요.

for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(i,sess.run(cost,feed_dict={x:xx,y:yy}),
              sess.run(w), sess.run(b))

print(sess.run(hypothesis, feed_dict={x: 30}))
print(sess.run(hypothesis, feed_dict={x: 50}))
print(sess.run(hypothesis, feed_dict={x: [30, 50]}))
```

```
# 문제 15-4  
# cars.csv 파일을 모델링한 결과를 그래프로 출력해 주세요.  
y1 = sess.run(hypothesis, feed_dict={x: 0})  
y2 = sess.run(hypothesis, feed_dict={x: 30})  
  
sess.close()  
  
import matplotlib.pyplot as plt  
  
plt.plot(xx, yy, 'ro')  
plt.plot([0, 30], [ 0, y2])  
plt.plot([0, 30], [y1, y2])  
plt.show()
```

```
# 문제 16-1
# x 변수가 두 개인 Multi-variable Linear Regression
# 알고리즘을 구현해 보세요.

x1 = [1, 0, 3, 0, 5]
x2 = [0, 2, 0, 4, 0]
y   = [1, 2, 3, 4, 5]

w1 = tf.Variable(tf.random_uniform([1], -1, 1))
w2 = tf.Variable(tf.random_uniform([1], -1, 1))
b  = tf.Variable(tf.random_uniform([1], -1, 1))

hypothesis = w1*x1 + w2*x2 + b
cost = tf.reduce_mean(tf.square(hypothesis-y))
learning_rate = tf.Variable(0.1)
```

```
# 문제 16-2
# x 변수가 두 개인 Multi-variable Linear Regression
# 알고리즘을 구현해 보세요.

train = tf.train.GradientDescentOptimizer
        (learning_rate).minimize(cost)
sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train)
    if i%20 == 0:
        print(sess.run(cost), sess.run(w1),
              sess.run(w2), sess.run(b))

sess.close()
```

```
# 문제 17-1
# 이전 문제를 행렬로 변환해서 동작하도록 수정해 보세요.

x = [[1., 0., 3., 0., 5.],
      [0., 2., 0., 4., 0.]]
y = [1, 2, 3, 4, 5]

w = tf.Variable(tf.random_uniform([1, 2], -1, 1))
b = tf.Variable(tf.random_uniform([1], -1, 1))

hypothesis = tf.matmul(w, x) + b  # (1,2)x(2,5)=(1,5)
cost = tf.reduce_mean(tf.square(hypothesis-y))
learning_rate = tf.Variable(0.1)
```

```
# 문제 17-2
# 이전 문제를 행렬로 변환해서 동작하도록 수정해 보세요.

train = tf.train.GradientDescentOptimizer
        (learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train)
    if i%20 == 0:
        print(sess.run(cost), sess.run(w), sess.run(b))

sess.close()
```

```
# 문제 18-1
# 이전 문제에서 bias를 없애 보세요.

x = [[1., 1., 1., 1., 1.],
      [1., 0., 3., 0., 5.],
      [0., 2., 0., 4., 0.]]
y = [1, 2, 3, 4, 5]

w = tf.Variable(tf.random_uniform([1, 3], -1, 1))

hypothesis = tf.matmul(w, x)          # (1,3)x(3,5)=(1,5)
cost = tf.reduce_mean(tf.square(hypothesis-y))
learning_rate = tf.Variable(0.1)
```

```
# 문제 18-2
# 이전 문제에서 bias를 없애 보세요.

train = tf.train.GradientDescentOptimizer
        (learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train)
    if i%20 == 0:
        print(sess.run(cost), sess.run(w))

sess.close()
```

```
# 문제 19-1
# placeholder 버전으로 수정하고,
# x1은 5, x2는 9일 때의 결과를 예측해 보세요.

import numpy as np
xy = np.loadtxt('Data/xy.txt', unpack=True,
                 delimiter=',')
xx = xy[:-1]
yy = xy[-1]

x = tf.placeholder(tf.float64)
y = tf.placeholder(tf.float64)
```

```
# 문제 19-2  
# placeholder 버전으로 수정하고,  
# x1은 5, x2는 9일 때의 결과를 예측해 보세요.  
w = tf.Variable(tf.random_uniform([1, 3], -1, 1,  
                           dtype=tf.float64))  
  
hypothesis = tf.matmul(w, x)      # (1,3)x(3,5)=(1,5)  
cost = tf.reduce_mean(tf.square(hypothesis-y))  
learning_rate = tf.Variable(0.1)  
train = tf.train.GradientDescentOptimizer  
       (learning_rate).minimize(cost)  
  
sess = tf.Session()  
sess.run(tf.global_variables_initializer())
```

```
# 문제 19-3  
# placeholder 버전으로 수정하고,  
# x1은 5, x2는 9일 때의 결과를 예측해 보세요.  
for i in range(2001):  
    sess.run(train, feed_dict={x: xx, y: yy})  
  
# 5, 9  
print(sess.run(hypothesis,  
               feed_dict={x: [[1], [5], [9]]}))  
# x1(5, 7), x2(9, 7)  
print(sess.run(hypothesis,  
               feed_dict={x: [[1, 1], [5, 7], [9, 7]]}))  
  
sess.close()
```

# 문제 20

# trees.csv를 읽어들여서 x와 y에 치환해 보세요.

```
trees = np.loadtxt('Data/trees.csv', unpack=True,  
                    delimiter=',', skiprows=1)
```

```
# print(trees)
```

```
# print(trees.shape)
```

```
# ones = [1] * len(trees[0])
```

```
ones = np.ones(len(trees[0]))
```

```
xx = np.array([ones, trees[0], trees[1]])
```

```
yy = trees[-1]
```

```
print(xx)
```

```
print(xx.shape, xx.dtype, yy.dtype)
```

```
# 문제 21-1  
# trees.csv를 읽어들여서 아래에 대해 volume을 예측해 보세요.  
# Girth 8.8, 10.5  
# Height 63, 72  
x = tf.placeholder(tf.float64)  
y = tf.placeholder(tf.float64)  
  
w = tf.Variable(tf.random_uniform([1, 3], -1, 1,  
                                 dtype=tf.float64))  
  
hypothesis = tf.matmul(w, x)    # (1,3)x(3,31)=(1,31)  
cost = tf.reduce_mean(tf.square(hypothesis-y))  
learning_rate = tf.Variable(0.00015)
```

```
# 문제 21-2
# trees.csv를 읽어들여서 아래에 대해 volume을 예측해 보세요.

train = tf.train.GradientDescentOptimizer
(learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(sess.run(cost, feed_dict={x:xx, y:yy}))
        print(sess.run(w))
```

# 문제 21-3

# trees.csv를 읽어들여서 아래에 대해 volume을 예측해 보세요.

# 8.8,63,10.2

# 10.5,72,16.4

```
print(sess.run(hypothesis,
               feed_dict={x: [[1, 1],
                             [8.8, 10.5],
                             [63, 72]]}))
```

sess.close()

```
# 문제 22-1  
# trees.csv 파일을 그래프로 그려보세요.  
import tensorflow as tf  
import numpy as np  
  
def showTree(x1, x2, label, rate):  
    ones = np.ones(len(x1))  
    # xx = np.array([ones, x1, x2])  
    xx = np.vstack([ones, x1, x2])  
    yy = label  
  
    x = tf.placeholder(tf.float64)  
    y = tf.placeholder(tf.float64)
```

```
# 문제 22-2
# trees.csv 파일을 그래프로 그려보세요.

w = tf.Variable(tf.random_uniform([1, 3], -1, 1,
                                  dtype=tf.float64))

hypothesis = tf.matmul(w, x) #(1,3)x(3,31)=(1,31)
cost = tf.reduce_mean(tf.square(hypothesis-y))
learning_rate = tf.Variable(rate)

train = tf.train.GradientDescentOptimizer(
    learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

# 문제 22-3

# trees.csv 파일을 그래프로 그려보세요.

```
for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(sess.run(cost,
                        feed_dict={x: xx, y: yy}),
              sess.run(w))

sess.close()
```

```
# 문제 22-4  
# trees.csv 파일을 그래프로 그려보세요.  
girth, height, volume = np.loadtxt('Data/trees.csv',  
                                  unpack=True, delimiter=',', skiprows=1)  
  
showTree(girth, height, volume, 0.00015)  
showTree(volume, girth, height, 0.0005)  
showTree(height, volume, girth, 0.0001)
```

# 문제 22-5

# trees.csv 파일을 그래프로 그려보세요.

```
import matplotlib.pyplot as plt
```

```
plt.plot(girth, height, 'ro')
```

```
plt.plot(height, volume, 'g>')
```

```
plt.plot(volume, girth, 'b+')
```

```
plt.xlim(0, 100)
```

```
plt.ylim(0, 100)
```

```
plt.show()
```

```
# 문제 22-6  
# trees.csv 파일을 그래프로 그려보세요.  
  
from mpl_toolkits.mplot3d import Axes3D  
  
  
fig = plt.figure()  
ax = fig.add_subplot(111, projection='3d')  
ax.plot(girth, height, volume, 'ro')  
plt.xlabel('girth')  
plt.ylabel('height')  
ax.set_zlabel('volume')  
plt.show()
```



DEEP LEARNING

---

SOURCE CODES

03

```
# 문제 23  
# sigmoid 함수를 구현해 보세요.
```

```
import math
```

```
def sigmoid(z):  
    return 1 / (1 + math.e ** -z)
```

```
print(sigmoid(-100))  
print(sigmoid(-10))  
print(sigmoid(0))  
print(sigmoid(10))  
print(sigmoid(100))
```

# 문제 24

# -10에서 10까지에 대해 sigmoid 함수의 반환값을  
# 그래프로 표시해 보세요.

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
xx, yy = [], []
```

```
for i in np.arange(-10, 10, 0.1):
```

```
    c = sigmoid(i)
```

```
    xx.append(i)
```

```
    yy.append(c)
```

```
plt.plot(xx, yy, 'ro')
```

```
plt.show()
```

```
# 문제 25-1
```

```
# Logistic Classification 알고리즘을 구현해 보세요.
```

```
x = [[1., 1., 1., 1., 1., 1.],  
      [2., 3., 3., 5., 7., 2.],  
      [1., 2., 5., 5., 5., 5.]]
```

```
y = np.array([0, 0, 0, 1, 1, 1])
```

```
w = tf.Variable(tf.random_uniform([1, 3], -1, 1))
```

```
z = tf.matmul(w, x)
```

```
hypothesis = tf.div(1., 1. + tf.exp(-z))
```

```
cost = -tf.reduce_mean(y*tf.log(hypothesis) +  
                      (1-y)*tf.log(1-hypothesis))
```

```
learning_rate = tf.Variable(0.1)
```

```
# 문제 25-2
# Logistic Classification 알고리즘을 구현해 보세요.

train = tf.train.GradientDescentOptimizer
        (learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train)
    if i%20 == 0:
        print(sess.run(cost))

sess.close()
```

```
# 문제 26-1
# 아래 데이터에 대해 Logistic Classification 알고리즘으로
# x1이 (4, 6), x2가 (4, 3)일 때의 결과를 예측해 주세요.

xx = [[1., 1., 1., 1., 1., 1.],
       [2., 3., 3., 5., 7., 2.],
       [1., 2., 5., 5., 5., 5.]]
yy = np.array([0, 0, 0, 1, 1, 1])

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.random_uniform([1, 3], -1, 1))
```

```
# 문제 26-2
# 아래 데이터에 대해 Logistic Classification 알고리즘으로
# x1이 (4, 6), x2가 (4, 3)일 때의 결과를 예측해 주세요.
z = tf.matmul(w, x)
hypo = tf.div(1., 1. + tf.exp(-z))
cost = -tf.reduce_mean(y*tf.log(hypo) +
                       (1-y)*tf.log(1-hypo))
learning_rate = tf.Variable(0.1)

train = tf.train.GradientDescentOptimizer(
    learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
# 문제 26-3
# 아래 데이터에 대해 Logistic Classification 알고리즘으로
# x1이 (4, 6), x2가 (4, 3)일 때의 결과를 예측해 주세요.

for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(i, sess.run(cost,
                           feed_dict={x: xx, y: yy}))
    r1 = sess.run(hypo, feed_dict={x: [[1], [4], [3]]})
    r2 = sess.run(hypo, feed_dict={x: [[1], [6], [3]]})
    print(r1 >= 0.5)
    print(r2 >= 0.5)

sess.close()
```

# 문제 27-1

# 텐서플로우 버전 cost 함수 그래프를 출력해 보세요.

```
import tensorflow as tf
```

```
x = [1, 2, 3]
```

```
y = [1, 2, 3]
```

```
w = tf.placeholder(tf.float32)
```

```
hypothesis = w*x
```

```
cost_tf = tf.reduce_mean(tf.square(hypothesis-y))
```

```
sess = tf.Session()
```

```
sess.run(tf.global_variables_initializer())
```

```
# 문제 27-2
# 텐서플로우 버전 cost 함수 그래프를 출력해 보세요.

xx, yy = [], []
for i in range(-30, 50):
    c = sess.run(cost_tf, feed_dict={w: i/10})
    xx.append(i/10)
    yy.append(c)

sess.close()

import matplotlib.pyplot as plt

plt.plot(xx, yy, 'ro')
plt.show()
```

# 문제 28-1  
# Logistic Classification에 대한 cost 그래프를 그려 보세요.

```
import tensorflow as tf
import numpy as np

x = [[1., 1., 1., 1., 1., 1.],
      [2., 3., 3., 5., 7., 2.],
      [1., 2., 5., 5., 5., 5.]]
y = np.array([0, 0, 0, 1, 1, 1])

w = tf.placeholder(tf.float32)
```

```
# 문제 28-2
# Logistic Classification에 대한 cost 그래프를 그려 보세요.

z = tf.matmul(w, x)
hypothesis = tf.div(1., 1. + tf.exp(-z))
cost = -tf.reduce_mean(y*tf.log(hypothesis) +
                       (1-y)*tf.log(1-hypothesis))

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

# 문제 28-3  
# Logistic Classification에 대한 cost 그래프를 그려 보세요.

```
x1, x2, yy = [], [], []  
for i in np.arange(-0.3, 0.5, 0.05):  
    for j in np.arange(-0.3, 0.5, 0.05):  
        c = sess.run(cost, feed_dict={w: [[0, i, j]]})  
        x1.append(i)  
        x2.append(j)  
        yy.append(c)  
  
sess.close()
```

# 문제 28-4  
# Logistic Classification에 대한 cost 그래프를 그려 보세요.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x1, x2, yy, 'ro')
plt.xlabel('x1')
plt.ylabel('x2')
ax.set_zlabel('yy')
plt.show()
```

```
# 문제 29-1

# iris.csv 파일에서 중앙의 숫자만 반환하는 함수를 구현하세요.

def getIris(species_true, species_false):
    f = open('Data/iris.csv', 'r', encoding='utf-8')

    f.readline()

    result = []
    for row in csv.reader(f):

        if row[-1] != species_true
            and row[-1] != species_false:
            continue
```

# 문제 29-2

# iris.csv 파일에서 중앙의 숫자만 반환하는 함수를 구현하세요.

```
item = [1.]          # bias
item.extend([float(i) for i in row[1:-1]])
item.append(row[-1] == species_true)

result.append(item)

f.close()
return np.array(result).transpose()
```

```
# 문제 30-1
# iris.csv 파일로부터 train과 test 데이터 셋을 분리해서
# 반환하는 함수를 구현해 보세요.

def getIrisReal(species_true, species_false):
    f = open('Data/iris.csv', 'r', encoding='utf-8')

    # skip header.
    f.readline()

    result = []
    for row in csv.reader(f):
        if row[-1] != species_true
            and row[-1] != species_false:
            continue
```

# 문제 30-2

# iris.csv 파일로부터 train과 test 데이터 셋을 분리해서  
# 반환하는 함수를 구현해 보세요.

```
item = [1.]          # bias
item.extend([float(i) for i in row[1:-1]])
item.append(row[-1] == species_true)

result.append(item)

f.close()
train = result[15:-15]
test  = result[:15] + result[-15:]
return np.array(train).transpose(),
        np.array(test).transpose()
```



```
# 문제 31-2
# iris.csv 파일에 대해 Logistic Classification 알고리즘의
# 정확도를 측정해 보세요.

z = tf.matmul(w, x)          # (1,6)x(6,100)=(1,100)
hypothesis = tf.div(tf.cast(1., tf.float64),
                    1. + tf.exp(-z))

cost = -tf.reduce_mean(y * tf.log(hypothesis) +
                      (1. - y) * tf.log(1. - hypothesis))

learning_rate = tf.Variable(0.1)

train = tf.train.GradientDescentOptimizer(
            learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
# 문제 31-3
# iris.csv 파일에 대해 Logistic Classification 알고리즘의
# 정확도를 측정해 보세요.

for i in range(2001):
    sess.run(train, feed_dict={x: train_set[:-1],
                               y: train_set[-1]})

yhat = sess.run(hypothesis,
                feed_dict={x: test_set[:-1]})

print((yhat >= 0.5) == test_set[-1])
print(np.mean((yhat >= 0.5) == test_set[-1]))
sess.close()

showAccuracy('setosa', 'versicolor')
showAccuracy('versicolor', 'virginica')
```



DEEP LEARNING

---

SOURCE CODES

04

# 문제 32

# SoftMax 알고리즘을 구현해 보세요.

```
import math
```

```
a, b, c = 2.0, 1.0, 0.1
```

```
aa = math.e ** a
```

```
bb = math.e ** b
```

```
cc = math.e ** c
```

```
base = aa + bb + cc
```

```
print('{:.5f}'.format(aa/base))
```

```
print('{:.5f}'.format(bb/base))
```

```
print('{:.5f}'.format(cc/base))
```

```
# 문제 33  
# softmax.txt 파일을 x, y에 8행 3열로 데이터를 읽어 보세요.  
xy = np.loadtxt('Data/softmax.txt', unpack=True)  
  
x = xy[:-3].transpose()  
y = xy[-3: ].transpose()  
  
# ----- #  
  
xy = np.loadtxt('Data/softmax.txt', dtype=np.float32)  
  
x = xy[:, :-3]  
y = xy[:, -3:]
```

```
# 문제 34-1  
# 이전 문제에서 읽어온 x, y에 SoftMax 알고리즘을 적용해 보세요.  
w = tf.Variable(tf.zeros([3, 3]))  
  
z = tf.matmul(x, w) # (8,3)x(3,1)=(8,3)  
hypothesis = tf.nn.softmax(z)  
cost = tf.reduce_mean(-tf.reduce_sum(y *  
                                tf.log(hypothesis)))  
  
learning_rate = tf.Variable(0.03)  
  
train = tf.train.GradientDescentOptimizer  
        (learning_rate).minimize(cost)
```

```
# 문제 34-2
# 이전 문제에서 읽어온 x, y에 SoftMax 알고리즘을 적용해 보세요.

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(2001):
    sess.run(train)
    if i%20 == 0:
        print(i, sess.run(cost))

sess.close()
```

```
# 문제 35-1
# 이전 문제에 대해
# x1이 (11, 7), x2가 (3, 4)일 때의 결과를 예측해 보세요.
xy = np.loadtxt('Data/softmax.txt', dtype=np.float32)

xx = xy[:, :-3]
yy = xy[:, -3:]

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

w = tf.Variable(tf.zeros([3, 3]))
```

```
# 문제 35-2
# 이전 문제에 대해
# x1이 (11, 7), x2가 (3, 4)일 때의 결과를 예측해 보세요.
z = tf.matmul(x, w)                      # (8,3)x(3,1)=(8,3)
hypothesis = tf.nn.softmax(z)
cost = tf.reduce_mean(-tf.reduce_sum(y *
                                      tf.log(hypothesis)))
learning_rate = tf.Variable(0.03)

train = tf.train.GradientDescentOptimizer(
    learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
# 문제 35-3
# 이전 문제에 대해
# x1이 (11, 7), x2가 (3, 4)일 때의 결과를 예측해 보세요.

for i in range(2001):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(i, sess.run(cost,
                           feed_dict={x: xx, y: yy}))

print(sess.run(hypothesis,
               feed_dict={x: [[1., 11., 7.]]}))
print(sess.run(hypothesis,
               feed_dict={x: [[1., 3., 4.]]}))
```

```
# 문제 35-4
# 이전 문제에 대해
# x1이 (11, 7), x2가 (3, 4)일 때의 결과를 예측해 보세요.

a1 = sess.run(hypothesis,
              feed_dict={x: [[1., 11., 7.]]})
a2 = sess.run(hypothesis,
              feed_dict={x: [[1., 3., 4.]]})
print(np.sum(a1))
print(np.sum(a2))

yhat1 = sess.run(tf.argmax(a1, 1))
yhat2 = sess.run(tf.argmax(a2, 1))
print(yhat1, yhat2)
```

```
# 문제 35-5
# 이전 문제에 대해
# x1이 (11, 7), x2가 (3, 4)일 때의 결과를 예측해 보세요.
nomial = ['A', 'B', 'C']
print(nomial[yhat1[0]])
print(nomial[yhat2[0]])

nomial = np.array(['A', 'B', 'C'])
print(nomial[yhat1])
print(nomial[yhat2])

sess.close()
```

```
# 문제 36
# iris_softmax.csv 파일로부터
# train과 test 데이터셋을 반환하는 함수를 만들어 보세요.

def getIrisSoftMax():
    iris = np.loadtxt('Data/iris_softmax.csv',
                      delimiter=',')
    train_set = np.vstack((iris[:40],
                          iris[50:90], iris[100:140]))
    test_set = np.vstack((iris[40:50],
                         iris[90:100], iris[140:]))

    return train_set, test_set
```

# 문제 37

# iris\_softmax.csv 파일로부터 난수를 사용해서  
# train과 test 데이터셋을 반환하는 함수를 만들어 보세요.

```
def getIrisSoftMaxRandom():
```

```
    iris = np.loadtxt('Data/iris_softmax.csv',  
                      delimiter=',')
```

```
    random.shuffle(iris)
```

```
    train_set = iris[:-30]
```

```
    test_set = iris[-30:]
```

```
    return train_set, test_set
```

### # 문제 38

```
# iris_softmax.csv 파일로부터 난수를 '현명하게' 사용해서
# train과 test 데이터셋을 반환하는 함수를 만들어 보세요.

def getIrisSoftMaxRandomAdv():
    iris = np.loadtxt('Data/iris_softmax.csv',
                      delimiter=',')
    i1, i2, i3 = iris[:50], iris[50:100], iris[100:]

    random.shuffle(i1)
    random.shuffle(i2)
    random.shuffle(i3)

    return np.vstack((i1[:40], i2[:40], i3[:40])),
           np.vstack((i1[40:], i2[40:], i3[40:]))
```

```
# 문제 39-1  
# 이전 문제에서 구한 test 데이터셋의 정확도를 측정해 보세요.  
train_set, test_set = getIrisSoftMax()  
# train_set, test_set = getIrisSoftMaxRandom()  
# train_set, test_set = getIrisSoftMaxRandomAdv()  
  
xx = train_set[:, :-3] # 120x5  
yy = train_set[:, -3:] # 120x3  
  
x = tf.placeholder(tf.float32)  
y = tf.placeholder(tf.float32)  
  
w = tf.Variable(tf.zeros([5, 3]))
```

```
# 문제 39-2
# 이전 문제에서 구한 test 데이터셋의 정확도를 측정해 보세요.

z = tf.matmul(x, w)                      # (120,5)x(5,3)=(120,3)
hypo = tf.nn.softmax(z)

log_sum = -tf.reduce_sum(y * tf.log(hypo),
                         reduction_indices=1)

cost = tf.reduce_mean(log_sum)
learning_rate = tf.Variable(0.005)

train = tf.train.GradientDescentOptimizer(
    learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
# 문제 39-3
# 이전 문제에서 구한 test 데이터셋의 정확도를 측정해 보세요.

for i in range(201):
    sess.run(train, feed_dict={x: xx, y: yy})
    if i%20 == 0:
        print(i, sess.run(cost,
                           feed_dict={x: xx, y: yy}))
```

```
tx = test_set[:, :-3]
ty = test_set[:, -3:]
```

```
y1 = sess.run(hypothesis, feed_dict={x: tx})
print(y1)
print(y1.shape)
```

# 문제 39-4

# 이전 문제에서 구한 test 데이터셋의 정확도를 측정해 보세요.

```
y2 = sess.run(tf.argmax(y1, 1))
```

```
t2 = sess.run(tf.argmax(ty, 1))
```

```
print(y2)
```

```
print(t2)
```

```
print(y2 == t2)
```

```
print(np.mean(y2 == t2))
```

```
nomial= np.array(['setosa','versicolor','virginica'])
```

```
print(nomial)
```

```
print(nomial[y2])
```

```
sess.close()
```

```
# 문제 40
# action.txt 파일을 반환하는 함수를 만드세요.

import numpy as np
import matplotlib.pyplot as plt
import random

def getAction():
    action = np.loadtxt('Data/action.txt',
                        delimiter=',')
    print(action.shape)

    return action[:, :-1], action[:, -1:]
```

# 문제 41

# Gradient Descent 알고리즘을 행렬 버전으로 구현해 보세요.

```
def gradDescent(x, y):
```

```
    m, n = x.shape
```

```
    alpha = 0.01
```

```
    w = np.ones((n, 1))
```

```
    for _ in range(m):
```

```
        h = sigmoid(np.dot(x, w))
```

```
        error = h - y
```

```
        grad = alpha * np.dot(x.transpose(), error)
```

```
        w -= grad
```

```
    return w
```

```
# 문제 42
# 이전 문제를 확률적인 방식으로 수정해 보세요.

def gradDescentStochastic(x, y):
    m, n = x.shape
    alpha, w = 0.01, np.ones(n)

    for i in range(m*10):
        pos = i % m
        h = sigmoid(np.sum(x[pos] * w))
        error = h - y[pos]
        grad = alpha * x[pos] * error
        w -= grad

    return w
```

# 문제 43

# 이전 문제를 난수 버전의 확률적인 방식으로 수정해 보세요.

```
def gradDescentStochasticFinal(x, y):
```

```
    m, n = x.shape
```

```
    alpha, w = 0.01, np.ones(n)
```

```
    for _ in range(m*10):
```

```
        pos = random.randrange(m)
```

```
        h = sigmoid(np.sum(x[pos] * w))
```

```
        error = h - y[pos]
```

```
        grad = alpha * x[pos] * error
```

```
        w -= grad
```

```
    return w
```

# 문제 44-1

# Decision Boundary 함수를 구현해서 그래프로 출력해 보세요.

```
def decisionBoundary(w):
```

```
    # y1 = -(w[0][0] + w[1][0]*-4) / w[2][0]
```

```
    # y2 = -(w[0][0] + w[1][0]* 4) / w[2][0]
```

```
    y1 = -(w[0] + w[1]*-4) / w[2]
```

```
    y2 = -(w[0] + w[1]* 4) / w[2]
```

```
    plt.plot([-4, 4], [y1, y2], color='red')
```

```
    plt.show()
```

# 문제 44-2

# Decision Boundary 함수를 구현해서 그라프로 출력해 보세요.

```
def decisionBoundaryStochastic(w, color):
    y1 = -(w[0] + w[1]*-4) / w[2]
    y2 = -(w[0] + w[1]* 4) / w[2]

    plt.plot([-4, 4], [y1, y2], color=color)
    plt.show()
```

```
# 문제 45
```

```
# 이전에 나온 확률적 코드를 호출하는 코드를 구현해 보세요.
```

```
x, y = getAction()
```

```
print(x.shape, y.shape)      # (100, 3) (100, 1)
```

```
w1 = gradDescent(x, y)
```

```
decisionBoundary(w1)
```

```
w2 = gradDescentStochastic(x, y)
```

```
decisionBoundaryStochastic(w2, 'green')
```

```
w3 = gradDescentStochasticFinal(x, y)
```

```
decisionBoundaryStochastic(w3, 'blue')
```

# 문제 46-1

# 구축한 모델을 저장하는 코드를 구현해 보세요.

```
saver = tf.train.Saver()
```

# 반복문 버전 (최종 5개만 저장)

```
saver.save(sess, 'Model/basic', global_step=i)
```

# 최종 결과만 저장

```
saver.save(sess, 'Model/model')
```

# 문제 46-2

# 저장한 모델을 복구하는 코드를 구현해 보세요.

```
saver = tf.train.Saver()
```

# 가장 최신 버전 복구

```
latest = tf.train.latest_checkpoint('Model')
```

```
if latest:
```

```
    saver.restore(sess, latest)
```

```
else:
```

```
    pass      # 학습
```

# 특정 버전으로 복구

```
saver.restore(sess, 'Model/basic-2000')
```

# 문제 47-1

# mnist 데이터셋에 대해 정확도를 측정하는 기본 코드를 구현해 보세요.

```
import tensorflow as tf
```

```
from tensorflow.examples.tutorials.mnist
```

```
import input_data
```

```
mnist = input_data.read_data_sets('mnist',  
                                  one_hot=True)
```

```
learning_rate = 0.01
```

```
epoches = 15
```

```
batch_size = 100
```

```
x = tf.placeholder(tf.float32)
```

```
y = tf.placeholder(tf.float32)
```



```
# 문제 47-3
# mnist 데이터셋에 대해 정확도를 측정하는 기본 코드를 구현해 보세요.

optimizer = tf.train.GradientDescentOptimizer
            (learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for epoch in range(epoches):
    avg_cost = 0
    total = mnist.train.num_examples // batch_size
```

```
# 문제 47-4
# mnist 데이터셋에 대해 정확도를 측정하는 기본 코드를 구현해 보세요.

for i in range(total):
    batch_xs, batch_ys =
        mnist.train.next_batch(batch_size)

    _, c = sess.run([optimizer, cost],
                  feed_dict={x: batch_xs, y: batch_ys})

    avg_cost += c / total_batch

print('{:2} : {}'.format(epoch+1, avg_cost))
```

# 문제 47-5

# mnist 데이터셋에 대해 정확도를 측정하는 기본 코드를 구현해 보세요.

```
pred = tf.equal(tf.argmax(activation, 1),
                tf.argmax(y, 1))

accuracy = tf.reduce_mean(tf.cast(pred, tf.float32))

print('accuracy : ', sess.run(accuracy,
                             feed_dict={x: mnist.test.images,
                                         y: mnist.test.labels}))  
  
sess.close()
```

# 문제 48-1

# 이전 문제를 Multi Layer 버전으로 수정해 보세요.

# (w부터 cost 함수까지의 코드만 수정)

```
w1 = tf.Variable(tf.random_normal([784, 256]))
```

```
w2 = tf.Variable(tf.random_normal([256, 256]))
```

```
w3 = tf.Variable(tf.random_normal([256, 10]))
```

```
b1 = tf.Variable(tf.random_normal([256]))
```

```
b2 = tf.Variable(tf.random_normal([256]))
```

```
b3 = tf.Variable(tf.random_normal([ 10]))
```

# 문제 48-2

# 이전 문제를 Multi Layer 버전으로 수정해 보세요.

# (w부터 cost 함수까지의 코드만 수정)

```
a1 = tf.add(tf.matmul(x, w1), b1)
```

```
r1 = tf.nn.relu(a1)
```

```
a2 = tf.add(tf.matmul(r1, w2), b2)
```

```
r2 = tf.nn.relu(a2)
```

```
a3 = tf.add(tf.matmul(r2, w3), b3)
```

```
activation = a3
```

```
soft_cost = tf.nn.softmax_cross_entropy_with_logits(  
    activation, y)
```

```
cost = tf.reduce_mean(soft_cost)
```

```
# 문제 49-1
# 이전 문제에 xavier 초기화 코드를 추가해 보세요.
# (w부터 cost 함수까지의 코드만 수정)

xavier = tf.contrib.layers.xavier_initializer()
w1 = tf.get_variable('w1', shape=[784, 256],
                     initializer=xavier)
w2 = tf.get_variable('w2', shape=[256, 256],
                     initializer=xavier)
w3 = tf.get_variable('w3', shape=[256, 10],
                     initializer=xavier)

b1 = tf.Variable(tf.zeros(256))
b2 = tf.Variable(tf.zeros(256))
b3 = tf.Variable(tf.zeros( 10))
```

```
# 문제 49-2
# 이전 문제에 xavier 초기화 코드를 추가해 보세요.
# (w부터 cost 함수까지의 코드만 수정)

a1 = tf.add(tf.matmul(x, w1), b1)
r1 = tf.nn.relu(a1)
a2 = tf.add(tf.matmul(r1, w2), b2)
r2 = tf.nn.relu(a2)
a3 = tf.add(tf.matmul(r2, w3), b3)

activation = a3
soft_cost = tf.nn.softmax_cross_entropy_with_logits(
                        activation, y)
cost = tf.reduce_mean(soft_cost)
```

```
# 문제 50-1  
# 이전 문제에 drop-out 알고리즘을 적용해 보세요.  
# (w부터 cost 함수까지의 코드 수정 + placeholder 추가)  
xavier = tf.contrib.layers.xavier_initializer()  
w1 = tf.get_variable('w1', shape=[784, 256],  
                     initializer=xavier)  
w2 = tf.get_variable('w2', shape=[256, 256],  
                     initializer=xavier)  
w3 = tf.get_variable('w3', shape=[256, 256],  
                     initializer=xavier)  
w4 = tf.get_variable('w4', shape=[256, 256],  
                     initializer=xavier)  
w5 = tf.get_variable('w5', shape=[256, 10],  
                     initializer=xavier)
```

```
# 문제 50-2  
# 이전 문제에 drop-out 알고리즘을 적용해 보세요.  
# (w부터 cost 함수까지의 코드 수정 + placeholder 추가)  
  
b1 = tf.Variable(tf.zeros(256))  
b2 = tf.Variable(tf.zeros(256))  
b3 = tf.Variable(tf.zeros(256))  
b4 = tf.Variable(tf.zeros(256))  
b5 = tf.Variable(tf.zeros( 10))  
  
dropout_rate = tf.placeholder(tf.float32)
```

```
# 문제 50-3  
# 이전 문제에 drop-out 알고리즘을 적용해 보세요.  
# (w부터 cost 함수까지의 코드 수정 + placeholder 추가)  
  
r1 = tf.nn.relu(tf.add(tf.matmul( x, w1), b1))  
d1 = tf.nn.dropout(r1, dropout_rate)  
r2 = tf.nn.relu(tf.add(tf.matmul(d1, w2), b2))  
d2 = tf.nn.dropout(r2, dropout_rate)  
r3 = tf.nn.relu(tf.add(tf.matmul(d2, w3), b3))  
d3 = tf.nn.dropout(r3, dropout_rate)  
r4 = tf.nn.relu(tf.add(tf.matmul(d3, w4), b4))  
d4 = tf.nn.dropout(r4, dropout_rate)
```

# 문제 50-4

# 이전 문제에 drop-out 알고리즘을 적용해 보세요.

# (w부터 cost 함수까지의 코드 수정 + placeholder 추가)

```
activation = tf.add(tf.matmul(d4, w5), b5)
```

```
log_mul = tf.nn.softmax_cross_entropy_with_logits(  
    activation, y)
```

```
cost = tf.reduce_mean(log_mul)
```

```
# 문제 50-5
# 이전 문제에 drop-out 알고리즘을 적용해 보세요.
# (w부터 cost 함수까지의 코드 수정 + placeholder 추가)

# 반복문 안쪽의 출력 코드 수정
_, c = sess.run([optimizer, cost],
               feed_dict={x: batch_xs,
                          y: batch_ys,
                          dropout_rate: 0.7})
```

```
# 문제 50-6
# 이전 문제에 drop-out 알고리즘을 적용해 보세요.
# (w부터 cost 함수까지의 코드 수정 + placeholder 추가)

# 전체 코드 마지막의 accuracy 출력 코드 수정
print('accuracy :',
      sess.run(accuracy,
              feed_dict={x: mnist.test.images,
                         y: mnist.test.labels,
                         dropout_rate: 1.0}))
```

**THE END**