Deep Learning for Natural Language Processing

Week 1 Introduction to PyTorch

Fast campus

17 Nov. 2018

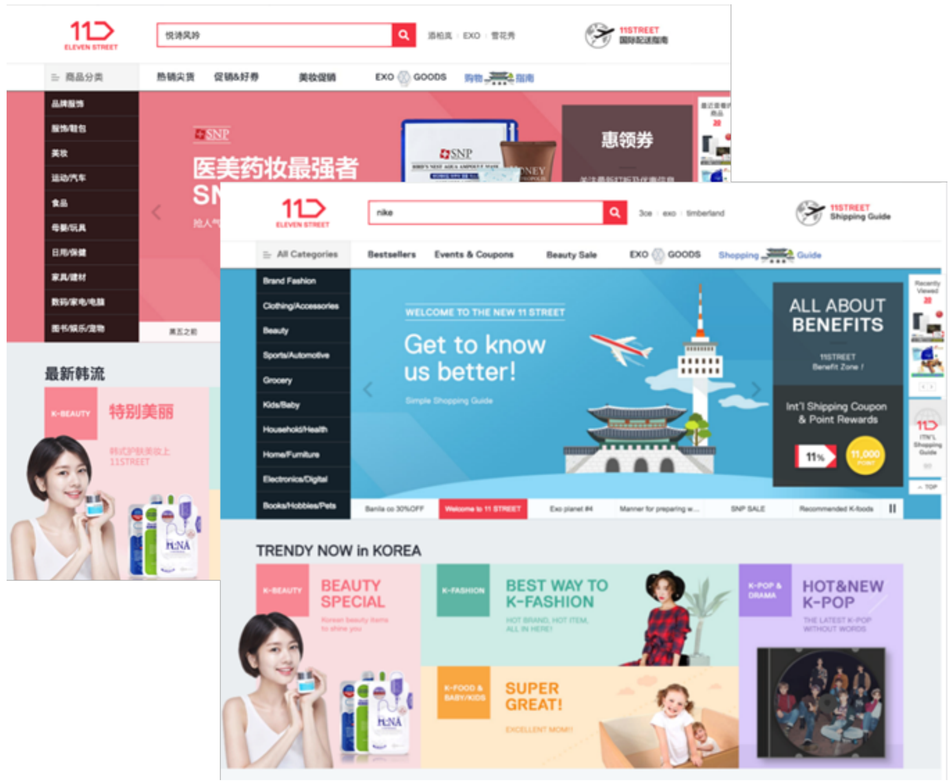
https://www.fastcampus.co.kr/data_camp_nlpbasic/

Course introduction

00

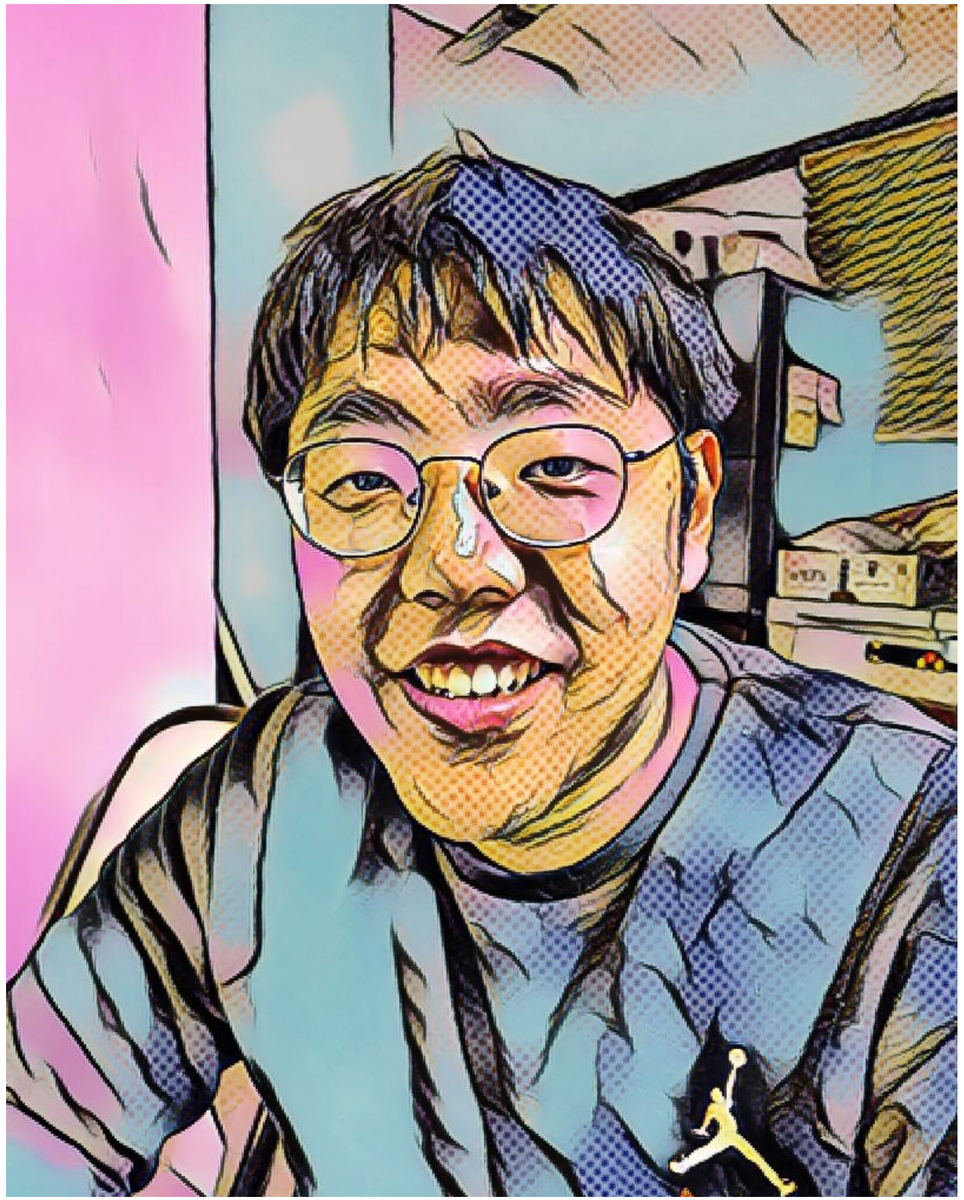
Ki Hyun Kim

- Machine Learning Engineer, MakinaRocks
  - Generative Learning, Unsupervised Learning, Anomaly Detection
- Machine Learning Researcher, SKPlanet
  - Neural Machine Translation
- Researcher, ETRI
  - Language Modeling for Speech Recognition, Automatic Speech Translation



Ki Hyun Kim

- LinkedIn: https://www.linkedin.com/in/ki-hyun-kim/
- GitHub: https://github.com/kh-kim/
- Email: pointzz.ki@gmail.com



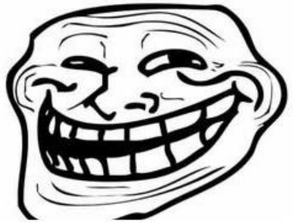
오상준

- Deep Learning Engineer, Deep Bio Inc.
- Biomedical Lab 랩장, 모두의연구소
- Docker Contents Creator, 모두를 위한 딥러닝 Season 2
- Co-founder and Research Engineer, QuantumSurf
- GitHub: https://github.com/juneoh
- Email: me@juneoh.net

A typical DL project workflow

1. Obtain and cleanse data
2. Choose a baseline model — e.g. ResNet, DenseNet, MobileNet, NASNet, MNASNet
3. Train to overfit
4. Tune and optimize — hard example mining, model pruning, custom loss functions, etc.
5. Deploy — e.g. using Open Neural Network Exchange Format (ONNX)

Machine Learning in Research	**Machine Learning in Engineering**
we spend months trying hard to design an algorithm complex enough to gain 0.3% improvement over previous best models	we use a simple algorithm and spend one week annotating more data to gain 3% improvement over previous week's result.



Definitions

- baseline: reference benchmark
- Proof-of-Concept (POC): a quick working prototype for an idea
- binary classification: A or not A
- multi-class classification: A, B, or C
- one-hot encoding: A as (1, 0, 0), B as (0, 1, 0), C as (0, 0, 1)

Who this course is for

In order to follow the hands-on sessions, you are expected to:

- know basic Python programming (data structures, OOP, etc.)
  - e.g. watched 무작정 따라하는 파이썬 프로그래밍 (YouTube)
- have heard of the basics of machine learning and/or deep learning
  - e.g. watched 모두를 위한 딥러닝/머신러닝 or PyTorch Zero to All

Who this course is for

And preferably, to fully catch the theory:

- understand the basic concepts of related topics in mathematics (statistics, probability, calculus, and linear algebra)
  - e.g. watched Machine Learning by Andrew Ng (Coursera)

But even if not, **just ask!**

What this course does NOT offer

- Basic programming and mathematics
  - (왕초보) 프로그래밍 첫걸음 시작하기 (Fast campus)
  - (초보) 파이썬으로 개발 시작하기 (Fast campus)
  - (중고급) 파이썬 완전 정복 CAMP (Fast campus)
  - (초급) 딥러닝을 위한 최적화와 수치해석 CAMP (Fast campus)

What this course does NOT offer

- *Everything* deep learning can do
  - (초급) PyTorch로 시작하는 딥러닝 입문 CAMP (Fast campus)
- Deeper topics of NLP: Sequence-to-Sequence, Neural Machine Translation
  - (고급) PyTorch를 활용한 자연어처리 심화 CAMP (Fast campus)

What this course DOES offer

- A rapid understanding of core concepts in deep NLP for intuition and insight
- Hands-on sessions with working project codebase to freely exploit
- Real-life tips and thorough support from field experts
- A comprehensive guide of how you should study further

The goal

A rapid, focused deep NLP bootcamp:

the journey from a beginner in deep learning to a junior NLP researcher, capable of building business-level text classification models.

Syllabus

Week 1 Introduction to PyTorch - 오상준

- (Theory) Introduction to Deep Learning
- (Practice) Hello PyTorch!

Week 2 Introduction to NLP - 김기현

- (Theory) Deeper Introduction to Deep Learning
- (Theory) Introduction to Natural Language Processing

Syllabus

Week 3 Word Sense Disambiguation - 김기현

- (Theory) Word Sense Disambiguation
- (Practice) Word Sense Disambiguation Excercise

Week 4 Word Vector Embedding - 오상준

- (Theory) Geometry of Deep Learning
- (Practice) Word Vector Embedding

Syllabus

Week 5 Deep Artificial Neural Networks - 오상준

- (Theory) Deep Neural Networks
- (Practice) Cryptocurrency Price Prediction

Week 6 Text Classification with PyTorch - 김기현

- (Theory) Naïve Bayes
- (Practice) Text Classification Excercise

Course mechanics

- Materials
  - By e-mail and GitHub
- Questions
  - Any time: during, before, or after lectures
  - In person, by email, or via Facebook (TBD)
- Fast campus regulations
  - Absences up to 2 times
  - 3 e-mail surveys: 1st, 3rd, 6th week

Complementary materials

Intermediate

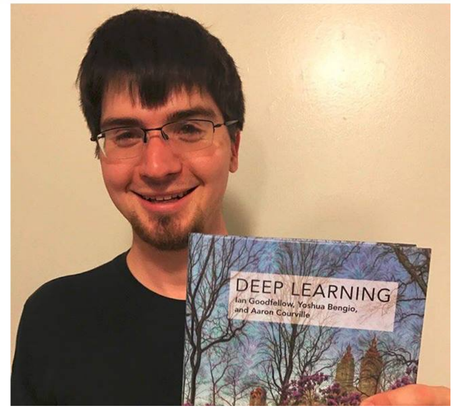
The materials below cover the theoretical parts of this course. They are recommended for further commitment.

- Deep Learning Book by Ian Goodfellow et al.
  - A thorough look at the basics, from linear algebra and calculus
- CS231n: Convolutional Neural Networks for Visual Recognition at Stanford

one taught me
love

one taught me
patience

one taught me
pain



Complementary materials

Intermediate

- CS224d: Deep Learning for Natural Language Processing at Stanford
- Oxford Deep NLP 2017
- Google 머신러닝 단기집중과정

Complementary materials

Advanced

- Pattern Recognition and Machine Learning 우리말 정리

Other

- 라온피플 머신러닝 아카데미
- colah's blog

Complementary materials

Tips and trends

- The official PyTorch forums
- PyTorch KR (Facebook)
- TensorFlow KR (Facebook)
- PR12 딥러닝 논문읽기 모임 (YouTube)
- Two Minute Papers (YouTube)
- Machine Learning (Reddit)
- Google AI Blog

Introduction to Deep Learning

- Genealogy
- Timeline
- Neural networks
- Activation functions and non-linearity
- Loss functions
- Regularization methods

01


SK Telecom
이젠, 삐삐라고 하지않는다!
초고속 인공지능, 스피드012 - 삐삐라고 하기엔 너무 똑똑하다!
파워 VMS로 더 강해져!
지역번경안내는 자동으로!
세계표준시각별 위성으로!
모든 서비스를 하나로!
건전지 수명 최대 4배까지!
미수신 메시지는 저장!
삐삐를 바꿔도 바꾸는그대로!
초고속 인공지능
SPEED 012

when someone says Sophia is an AI

Giedrius Trump

when someone says Sophia is an AI

32

1h **Haha** **Reply**

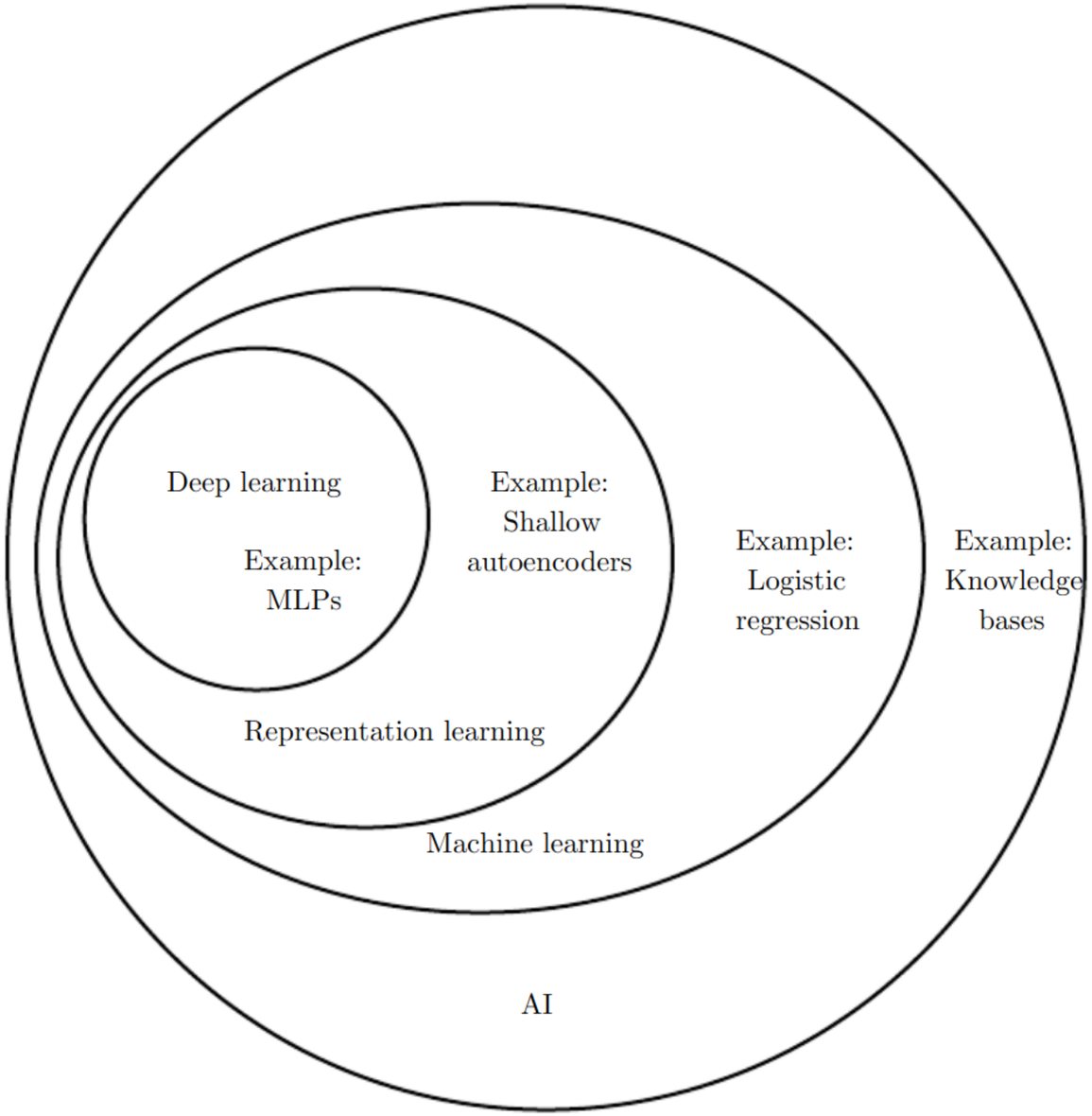
Yann LeCun

Exactly, except that Geoff and Andrew would push me into the fight.

76

1h **Like** **Reply**

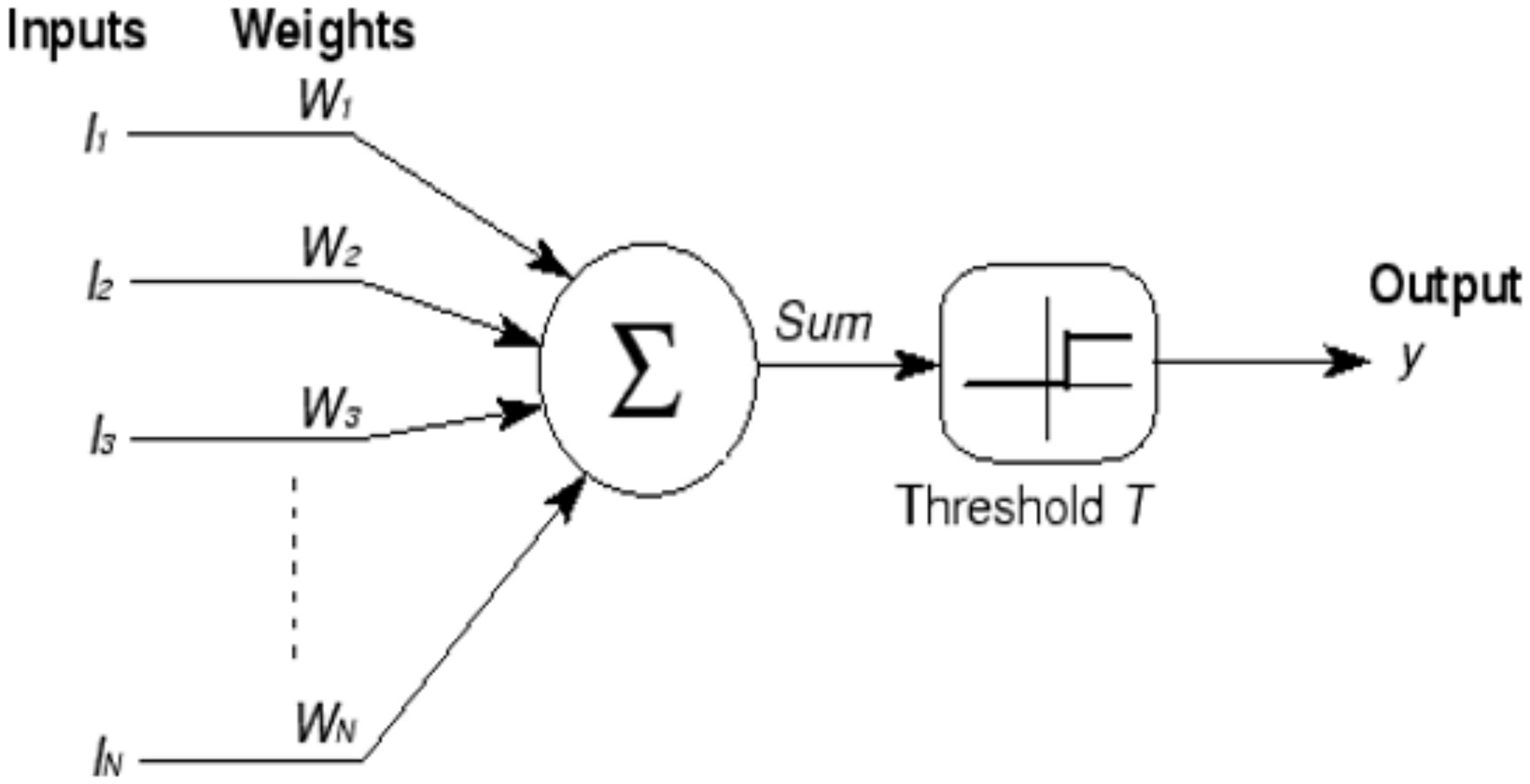
https://www.facebook.com/yann.lecun/posts/10155635039762143


Deep learning
Example:
MLPs
Example:
Shallow
autoencoders
Example:
Logistic
regression
Example:
Knowledge
bases
Representation learning
Machine learning
AI

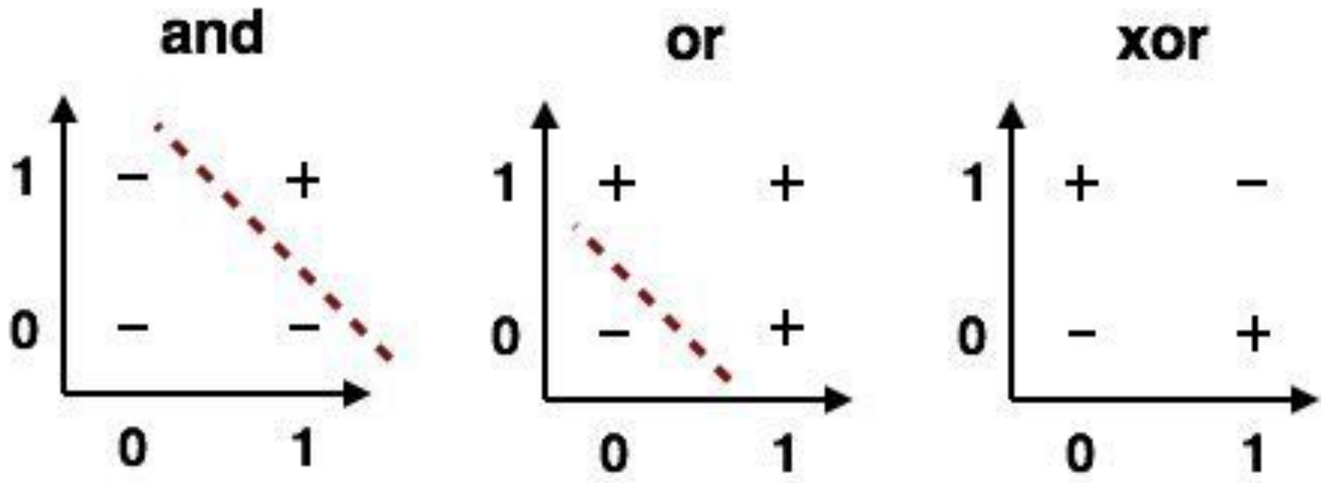
Timeline

- **Cybernetics** 1940s-1960s
  - McCulloch-Pltts neuron
    - McCulloch and Pitts, 1942. A Logical Calculus of the Ideas Immanent in Nervous Activity
  - Hebbian Learning
    - Hebb, 1949. The Organization of Behaviour.
  - Perceptron
    - Rosenblatt, 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization

Timeline


Inputs
Weights
W_1
I_1
W_2
I_2
W_3
I_3
W_N
I_N
Σ
Sum
Threshold T
Output
y

Timeline


and
1
0
−
+
−
−
0
1
or
1
0
+
+
−
+
0
1
xor
1
0
+
−
−
+
0
1


BRACE YOURSELVES
AI WINTER IS COMING
memegenerator.net

Timeline

- **Connectionism** 1980s-1990s
  - Backpropagation
    - Rumelhart et al., 1986. Learning Representations by Back-propagating Errors.
  - Convolutional Neural Networks
    - Fukushima, 1980. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.

Timeline

- **Deep Learning** 2006-
  - Deep Neural Networks
    - Hinton et al., 2006. A Fast Learning Algorithm for Deep Belief Nets.
  - Rectified Linear Units
    - Golorot et al., 2011. Deep Sparse Rectifier Neural Networks.
  - AlexNet
    - Krizhevsky et al, 2012. ImageNet Classification with Deep Convolutional Neural Networks.

And God said,

"Let there be light!"

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

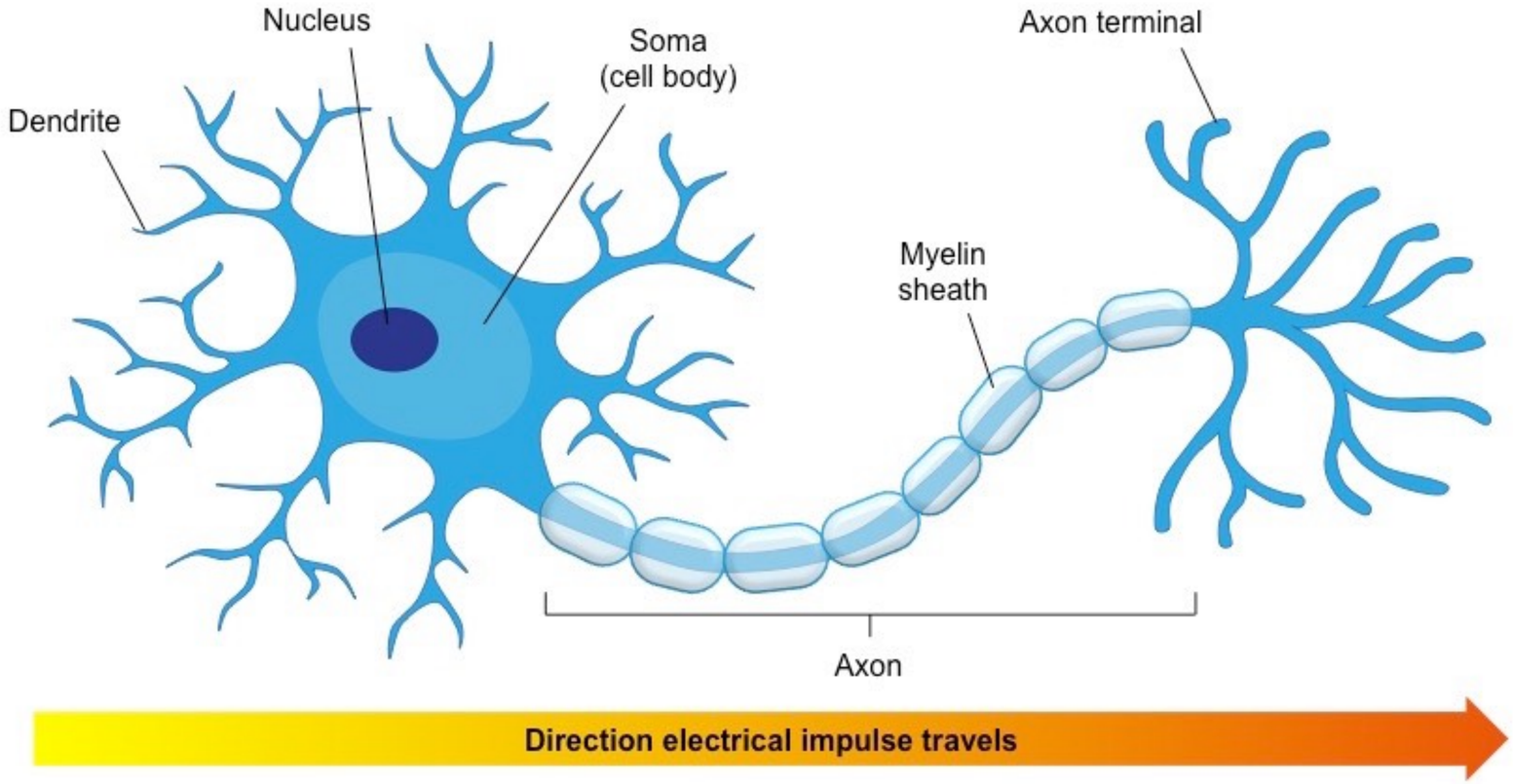
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-

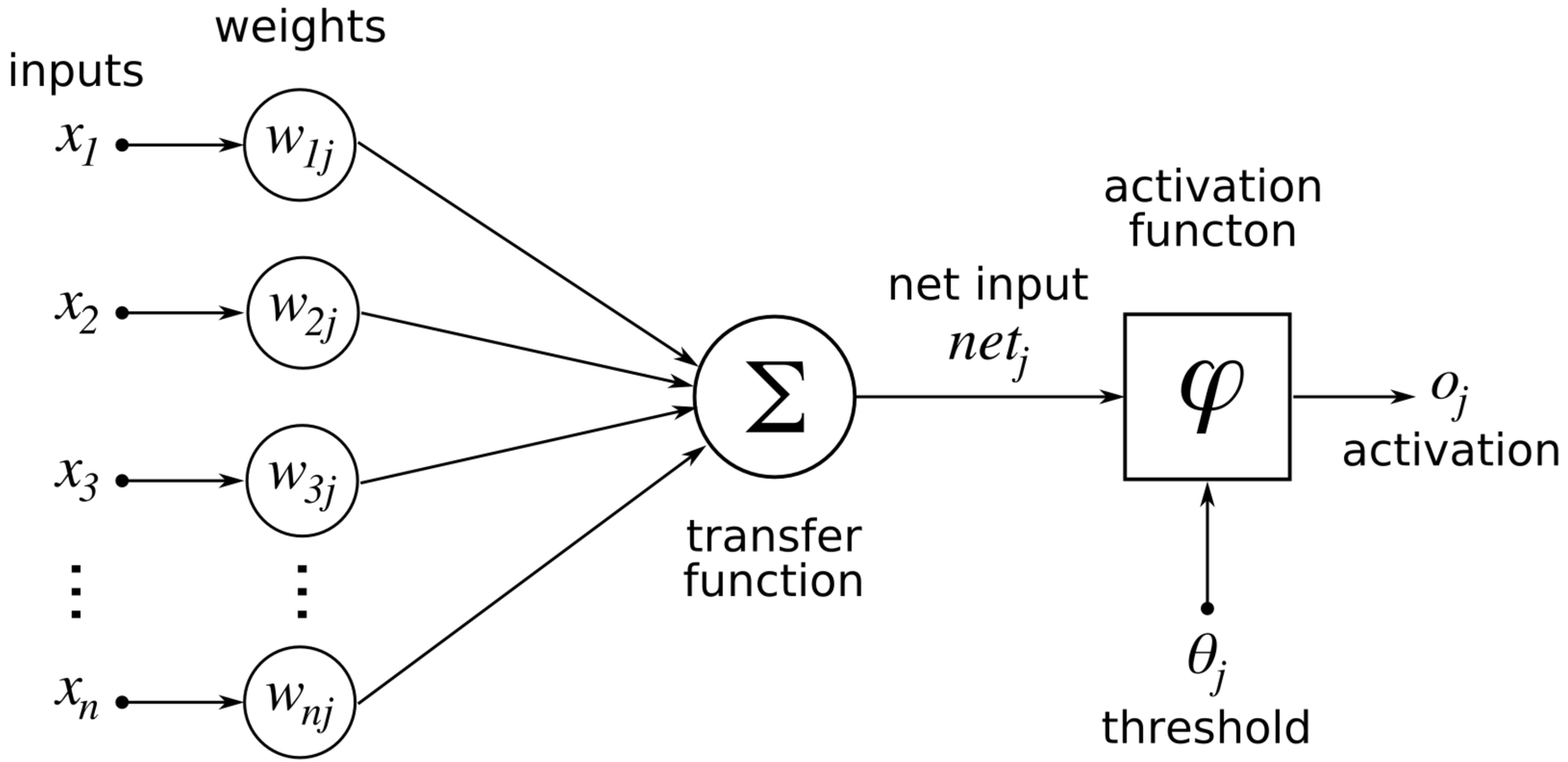
Neural networks

- Feed-forward networks



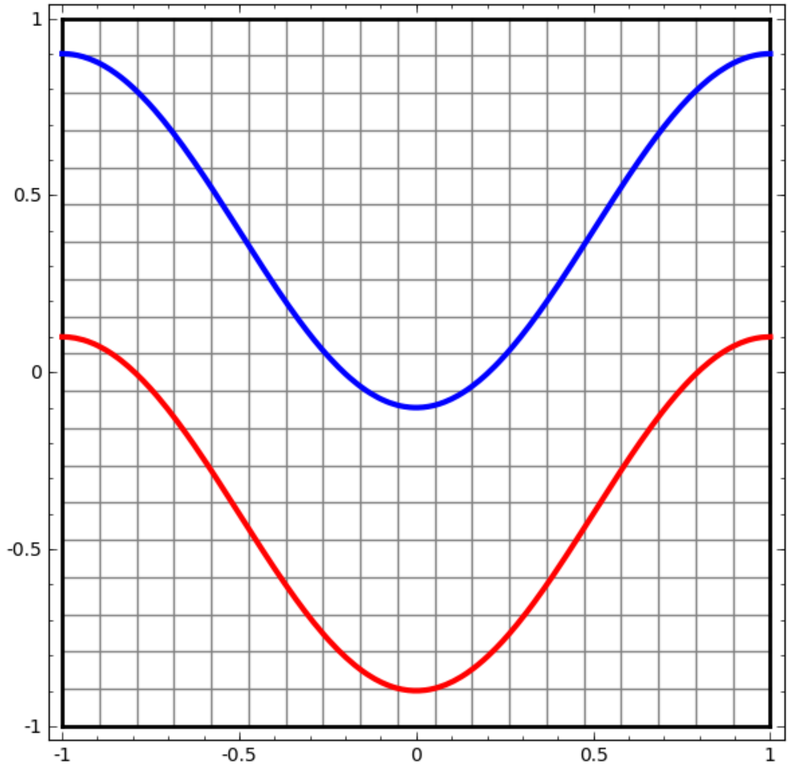
Neural networks

- Feed-forward networks



Neural networks

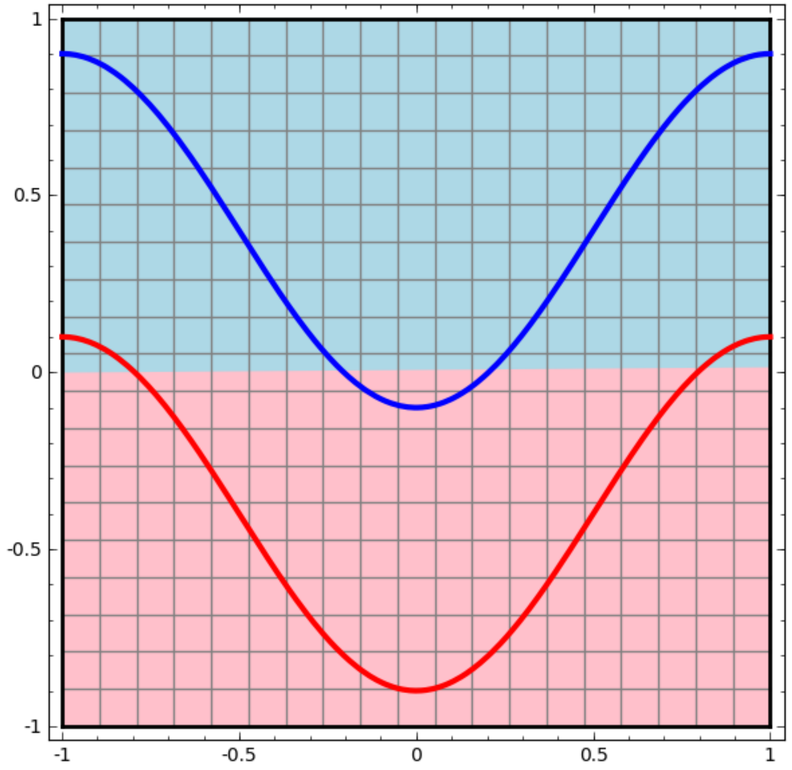
Problem: draw a single straight line to separate colors.



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Neural networks

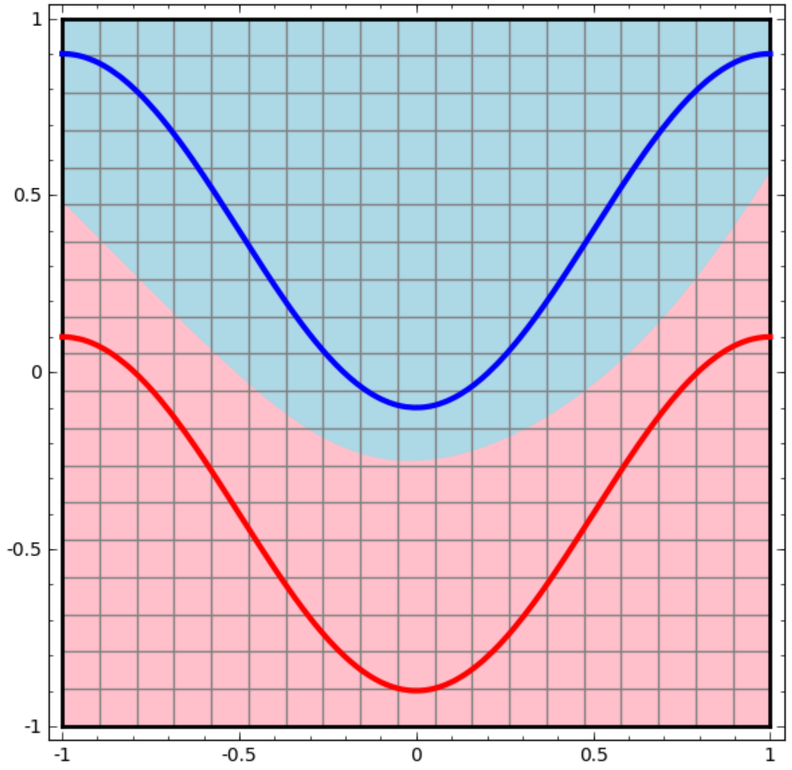
Problem: draw a single straight line to separate colors.



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Neural networks

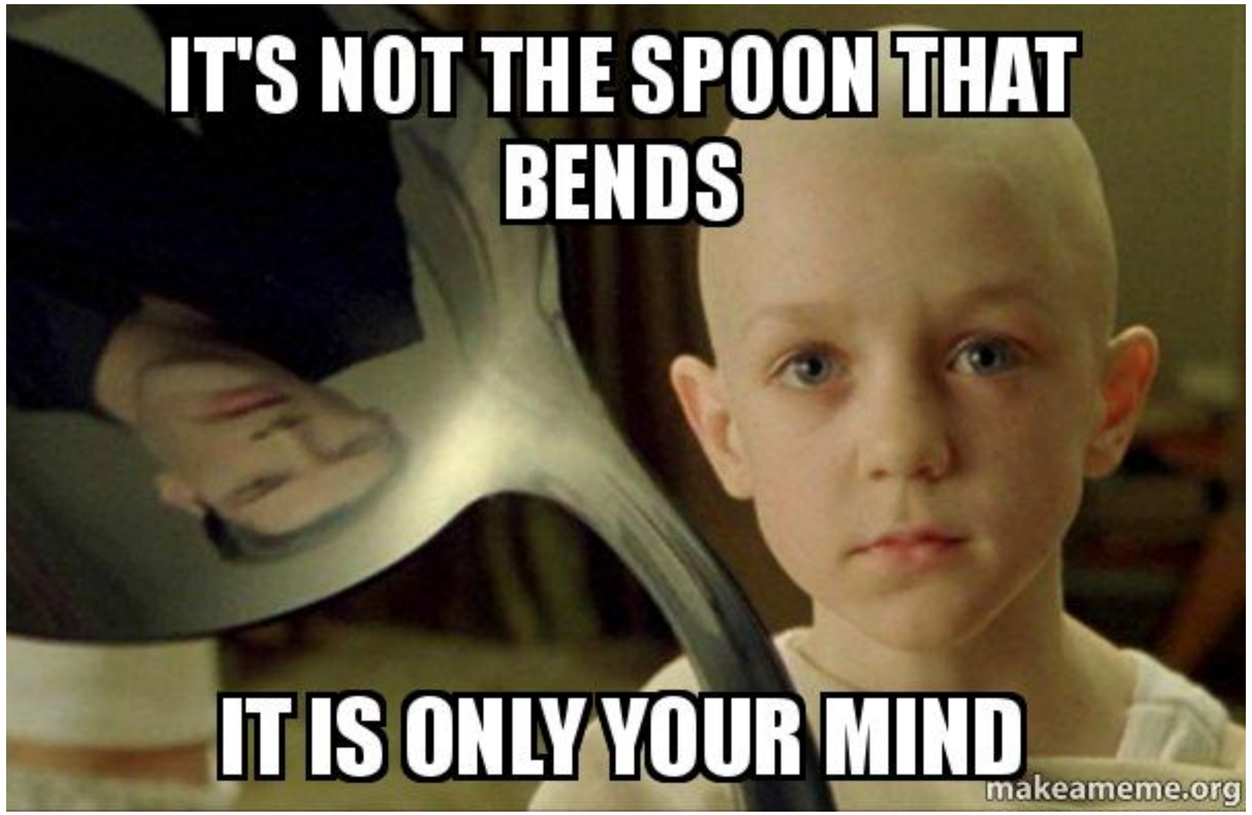
Problem: draw a single straight line to separate colors.



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

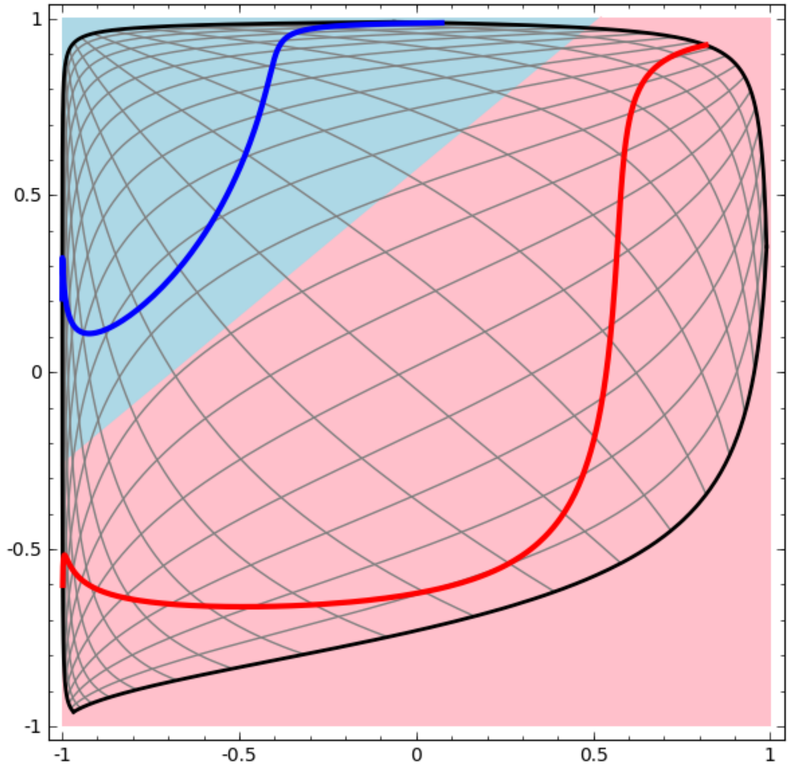
Neural networks

Problem: draw a single straight line to separate colors.

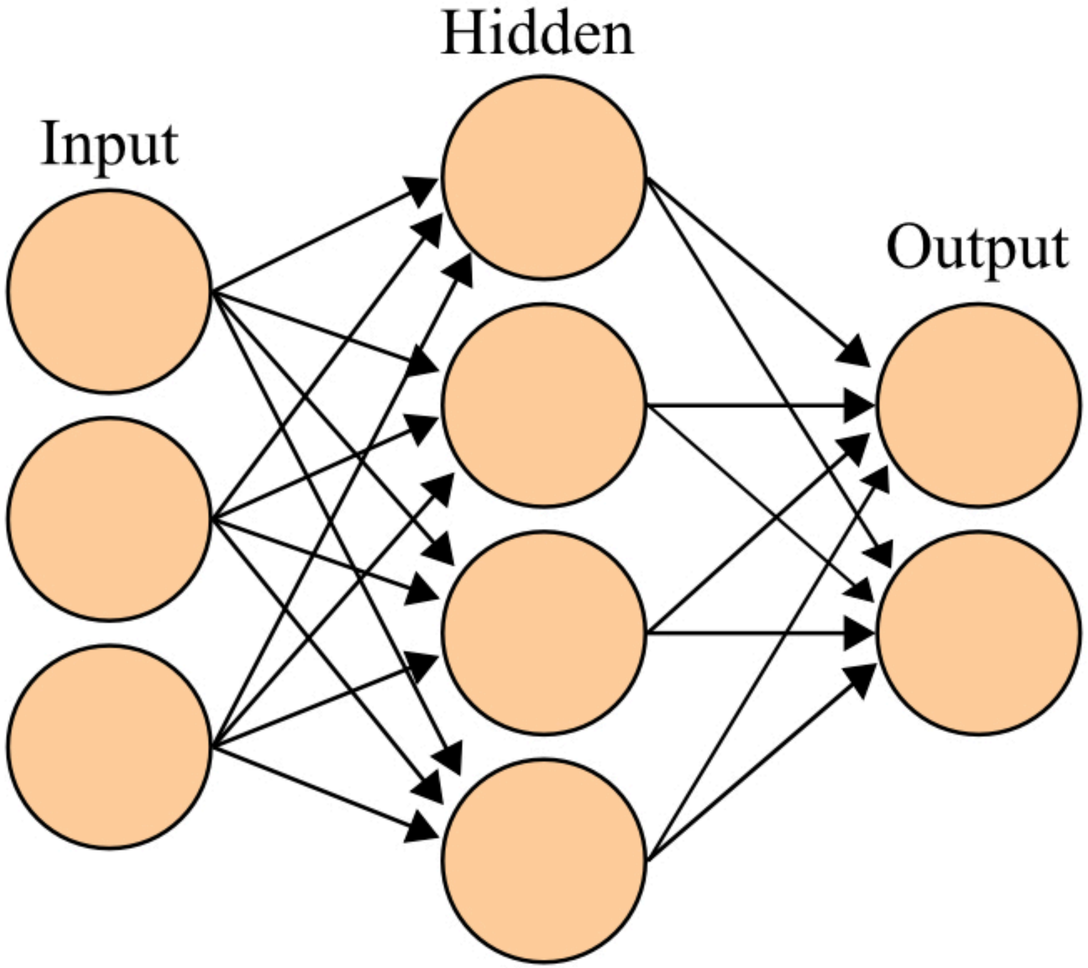


Neural networks

Problem: draw a single straight line to separate colors.



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

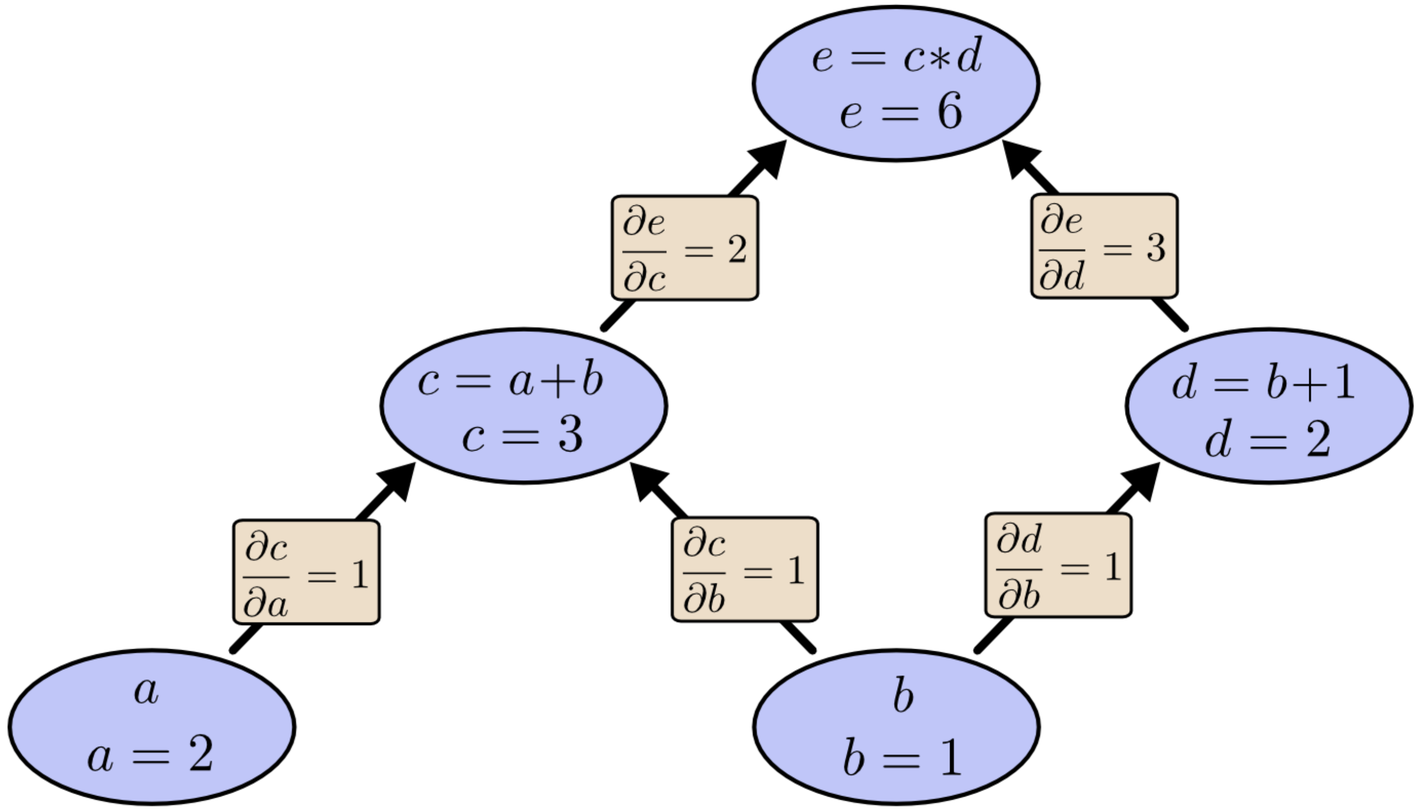


The hidden layer learns a **representation**, so that the data is linearly separable.

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Neural networks

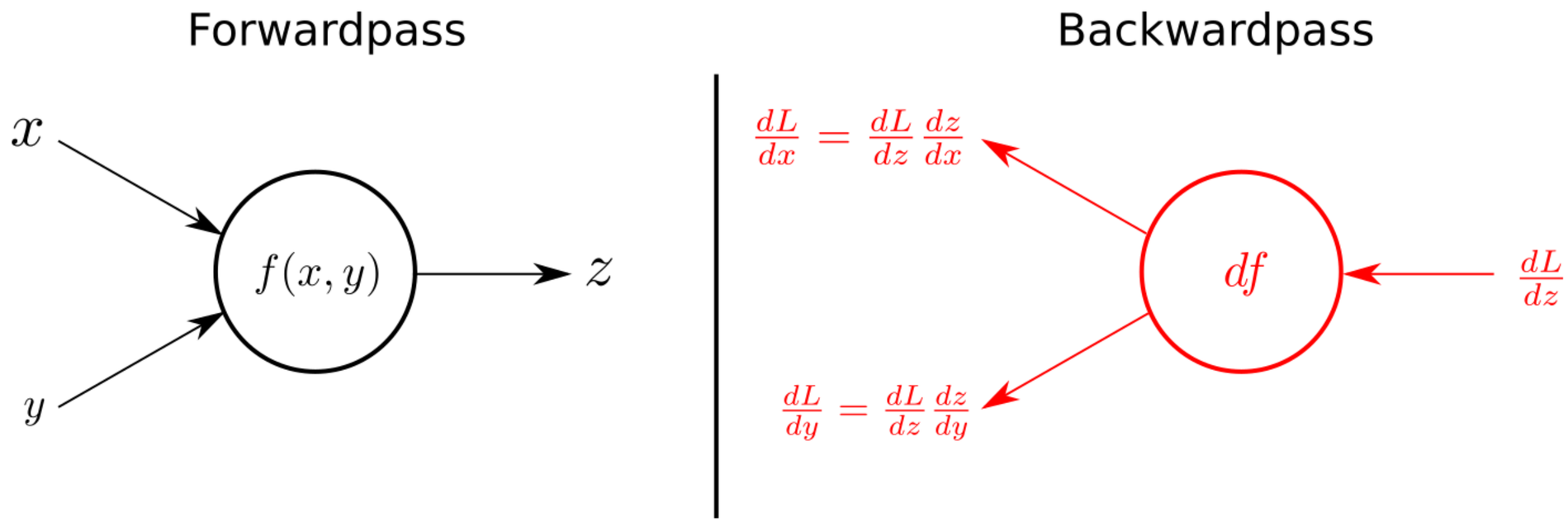
- Backpropagation



http://colah.github.io/posts/2015-08-Backprop/

Neural Networks

- Backpropagation

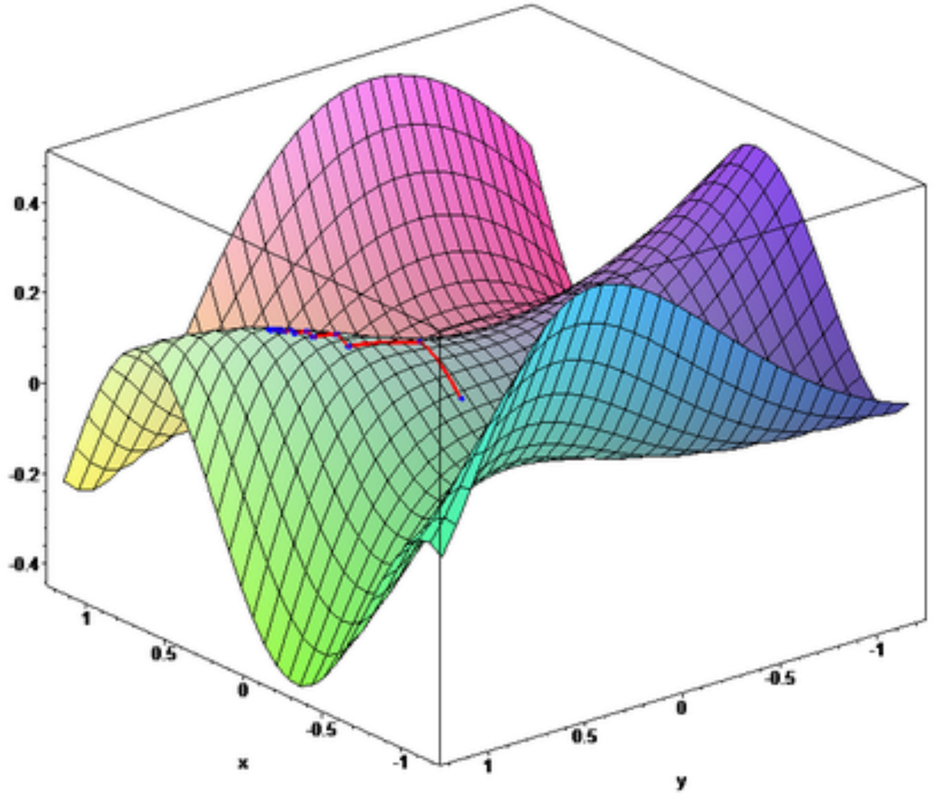


https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html


IT IS FRIDAY, ALL MY FRIENDS ARE AT PARTY
BUT, I'M HERE LEARNING BACK PROPAGATION
memegenerator.net

Neural networks

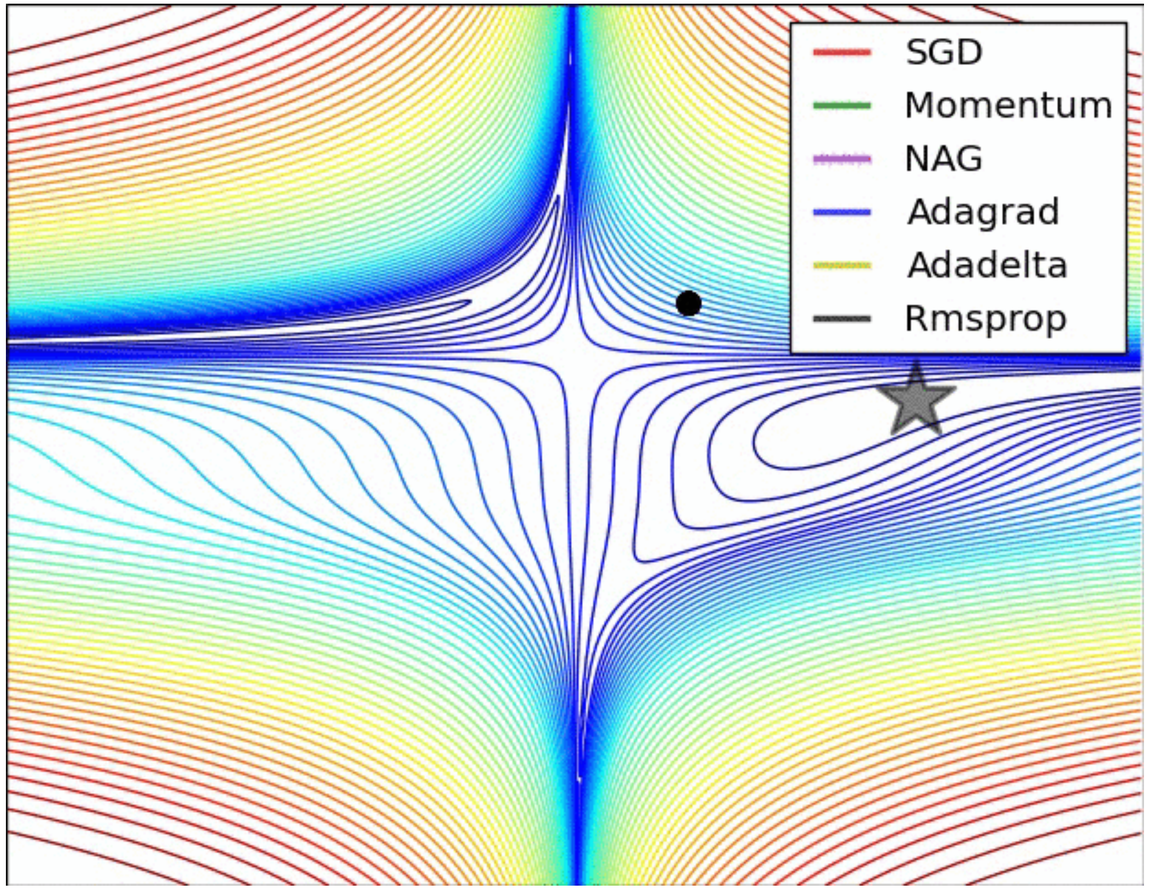
- Gradient descent



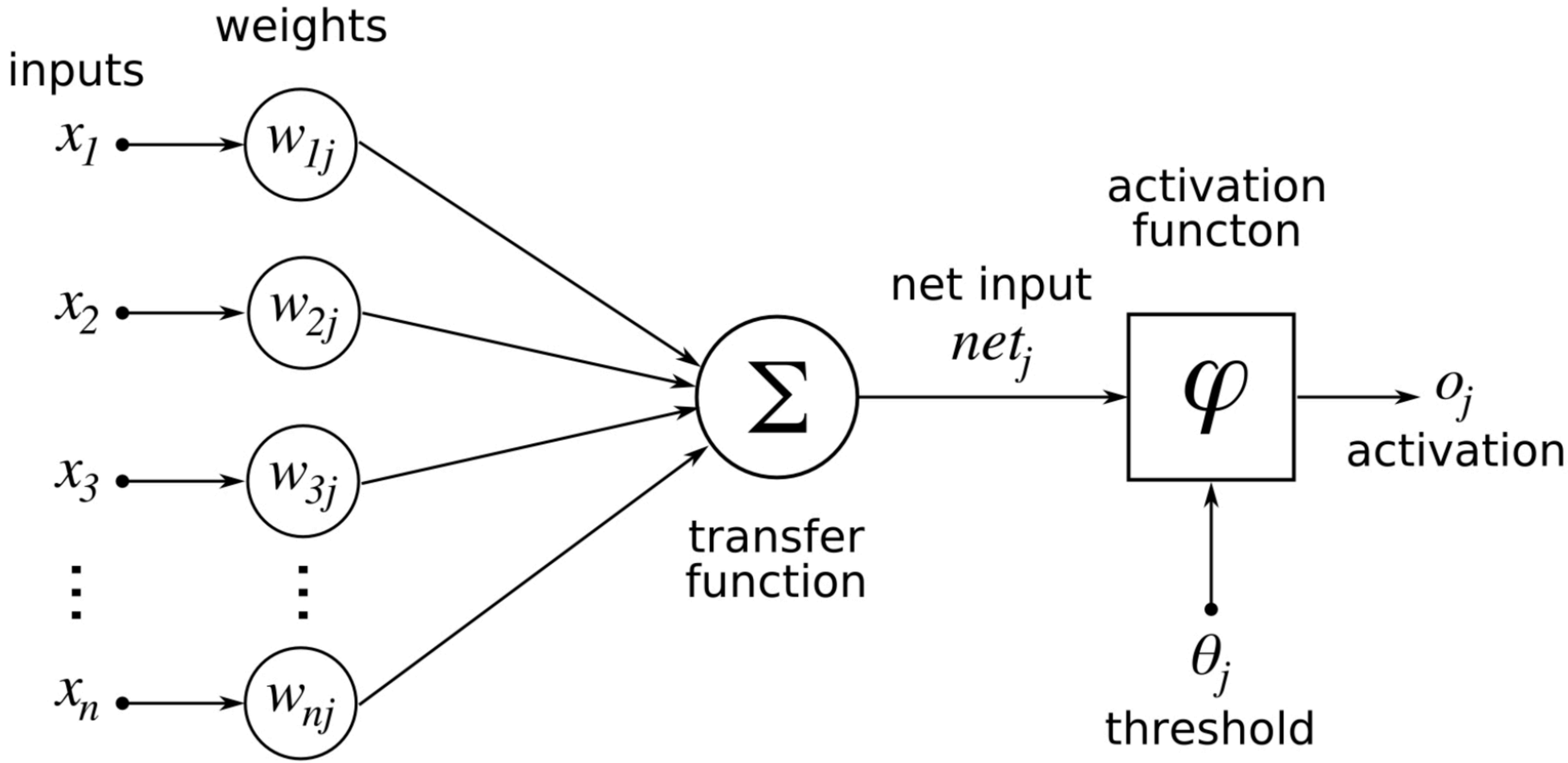

```
def updated_weights(weights, gradients, learning_rate):
    return weights - learning_rate * (gradients - weights)
```

Neural networks

- Gradient descent
  - **Stochastic Gradient Descent (SGD)** is steady and stable.
  - **Adam** is fast, but sometimes wacky.

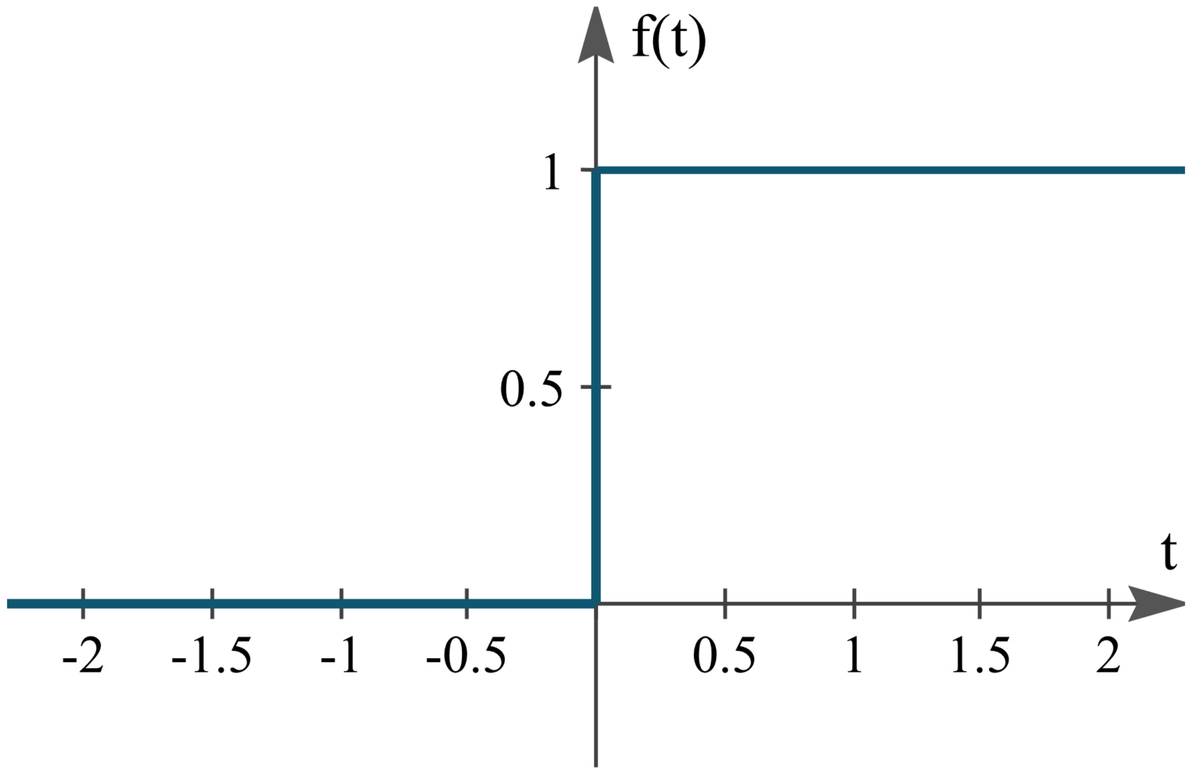


Activation functions and non-linearity



Activation functions and non-linearity

- Step function

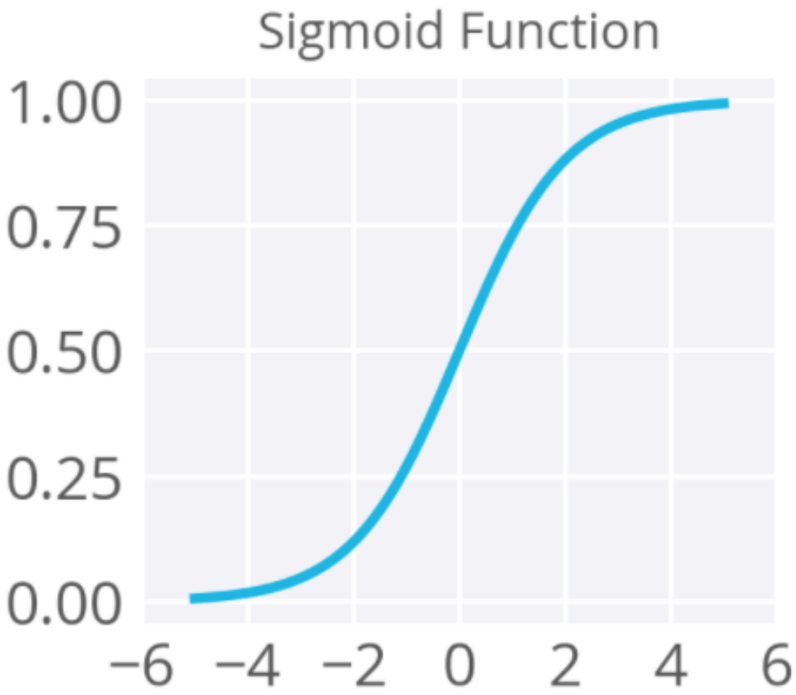


Activation functions and non-linearity

Activation functions and non-linearity

- Sigmoid function

$$\frac{1}{1 + e^{-x}}$$

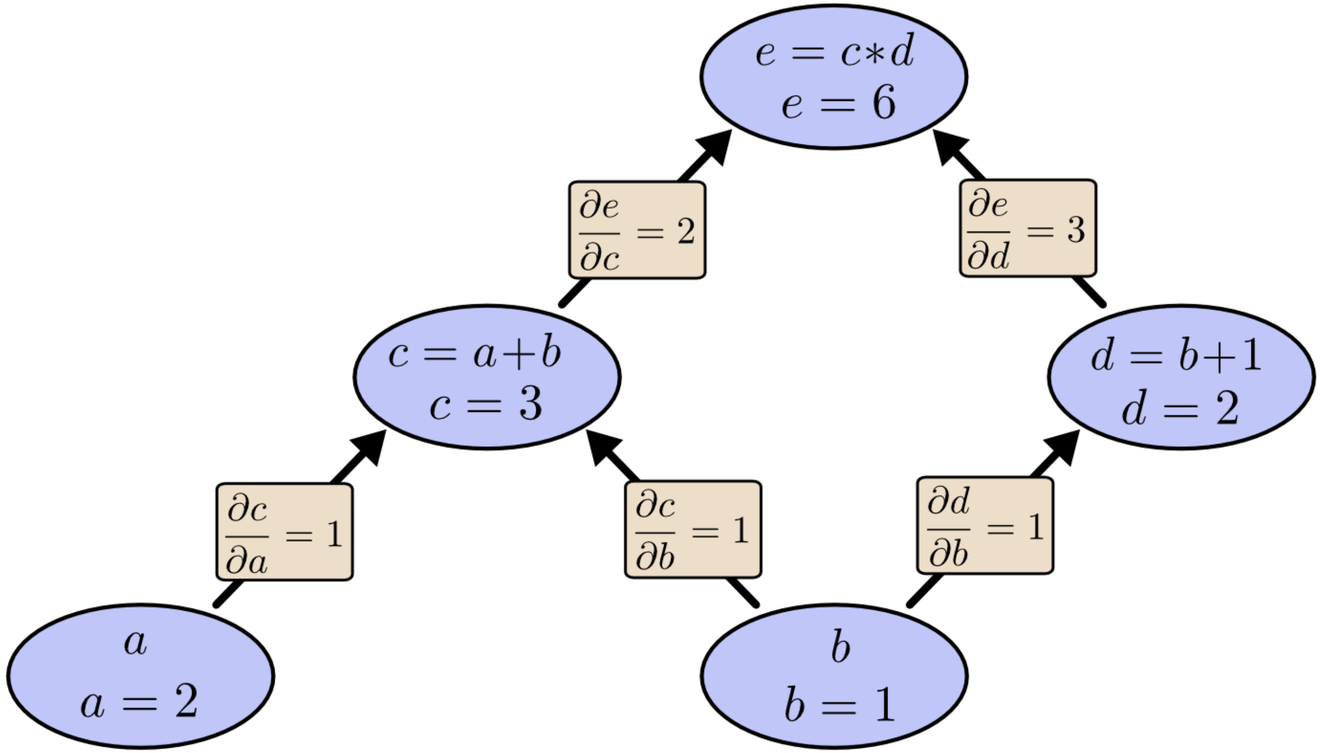



```
def sigmoid(inputs):
    return 1.0 / (1.0 + exp(-inputs))
```

https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions

Activation functions and non-linearity

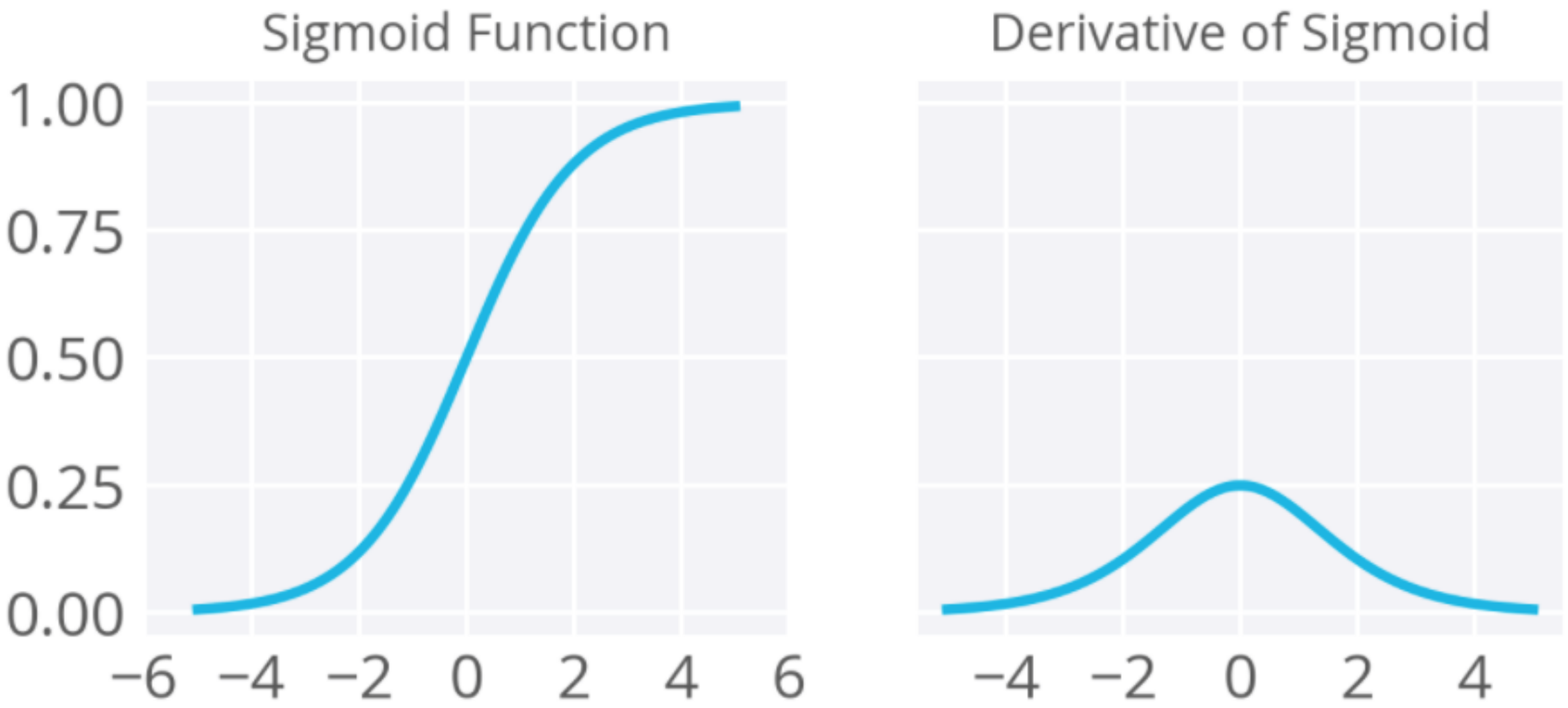
- Sigmoid function



http://colah.github.io/posts/2015-08-Backprop/

Activation functions and non-linearity

- Sigmoid function

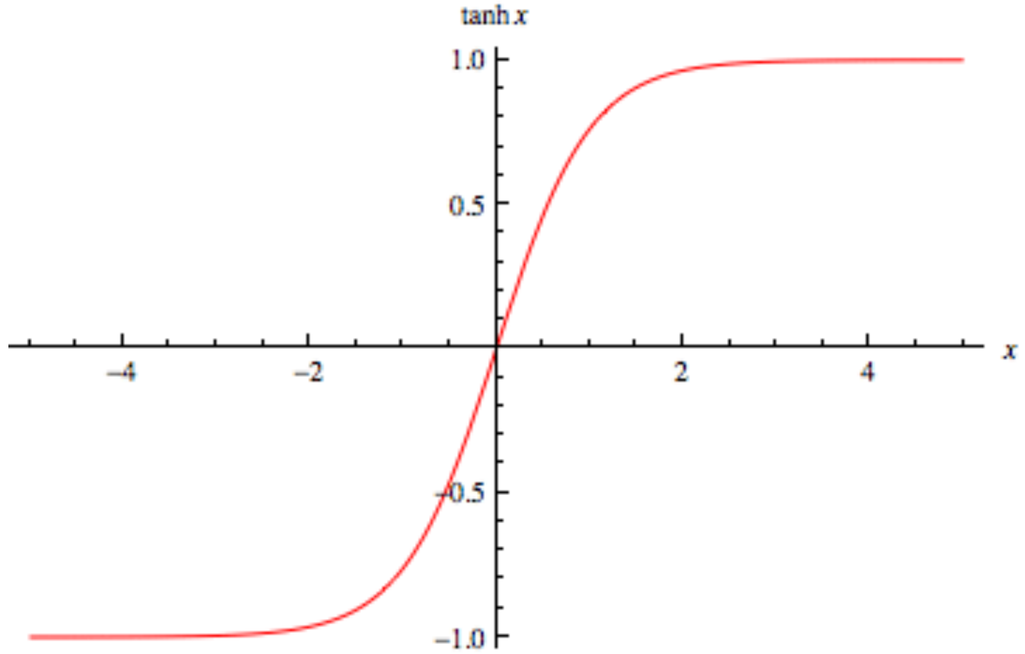


https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions

Activation functions and non-linearity

- Hyperbolic Tangent (tanh)

$$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

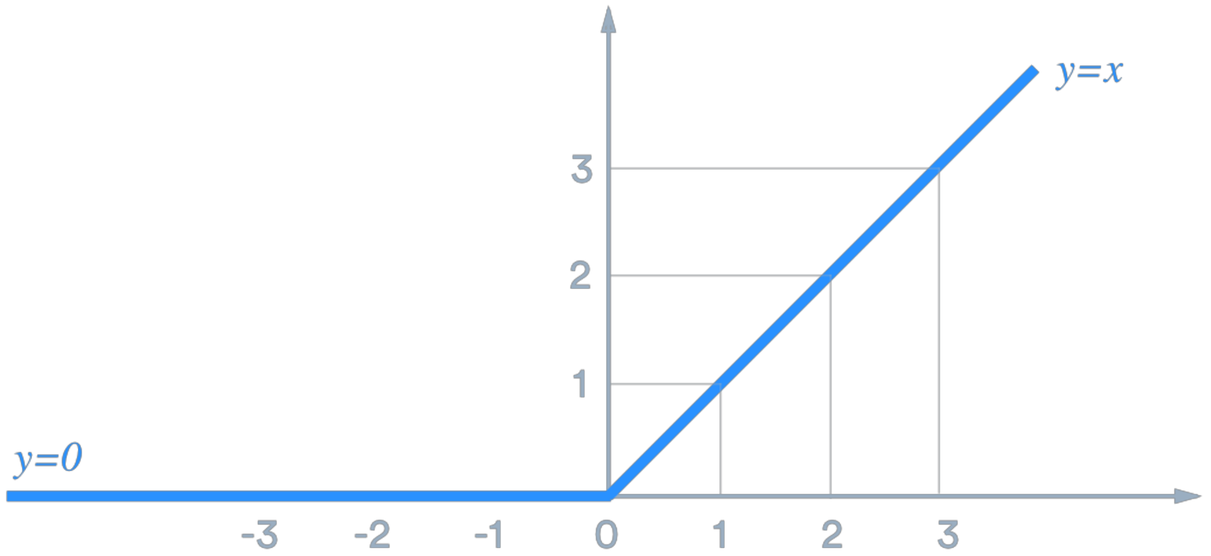



```
def tanh(inputs):
    return (exp(inputs) - exp(-inputs)) / (exp(inputs) + exp(-inputs))
```

Activation functions and non-linearity

- Rectified Linear Units (ReLU)

$$\max(0, x)$$




```
def relu(inputs):
    return max(0, inputs)
```

Activation functions and non-linearity

- Softmax function



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

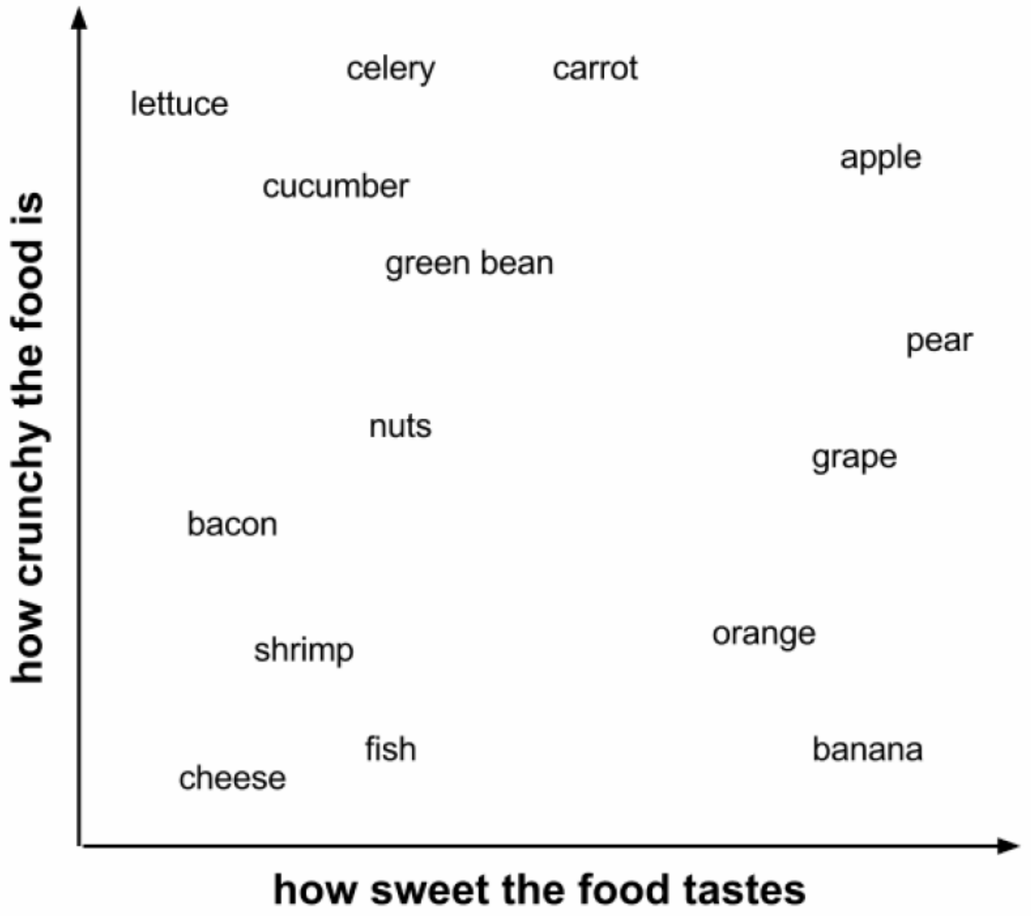
$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \rightarrow \text{Softmax} \rightarrow \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix}$$

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture03.pdf

https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions

Activation functions and non-linearity

- Where activation functions are used:
  - in-between layers: **ReLU** and its variants
  - after the final layer: **sigmoid** (binary) or **softmax** (multiclass)



K-Nearest Neighbors

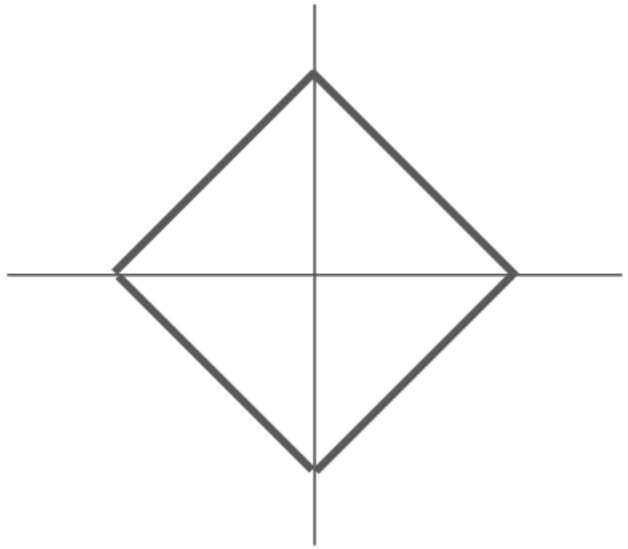
Loss functions

- L1 Loss

$$\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



```
def l1_loss(targets, outputs):
    return sum(abs(targets - outputs))
```

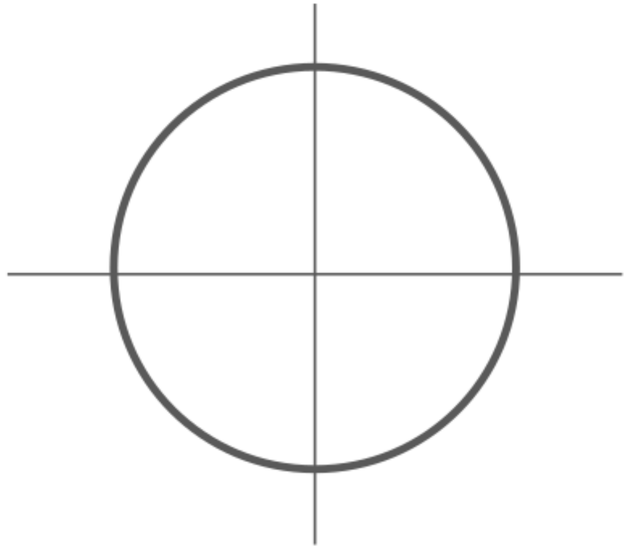
Loss functions

- L2 Loss

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



```
def l2_loss(targets, outputs):
    return sum(sqrt((targets - outputs)**2))
```

Loss functions

- Mean squared error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

	상세 점수											
	야구순위	1	2	3	4	5	6	7	8	9	10	
순위	현재	두	슼	한	넥	기	삼	롯	엘	클	엔	
1		기	두	엔	롯	슼	엘	삼	넥	클	한	32
1		기	엔	두	엘	넥	슼	삼	롯	클	한	32
5		엔	기	롯	두	엘	넥	슼	삼	한	클	38
1		기	두	엔	엘	슼	롯	넥	삼	한	클	32
4		기	두	엔	엘	롯	슼	넥	삼	클	한	34

```
def mean_square_error(targets, outputs):
    return mean(sqrt((targets - outputs)**2))
```

Loss functions

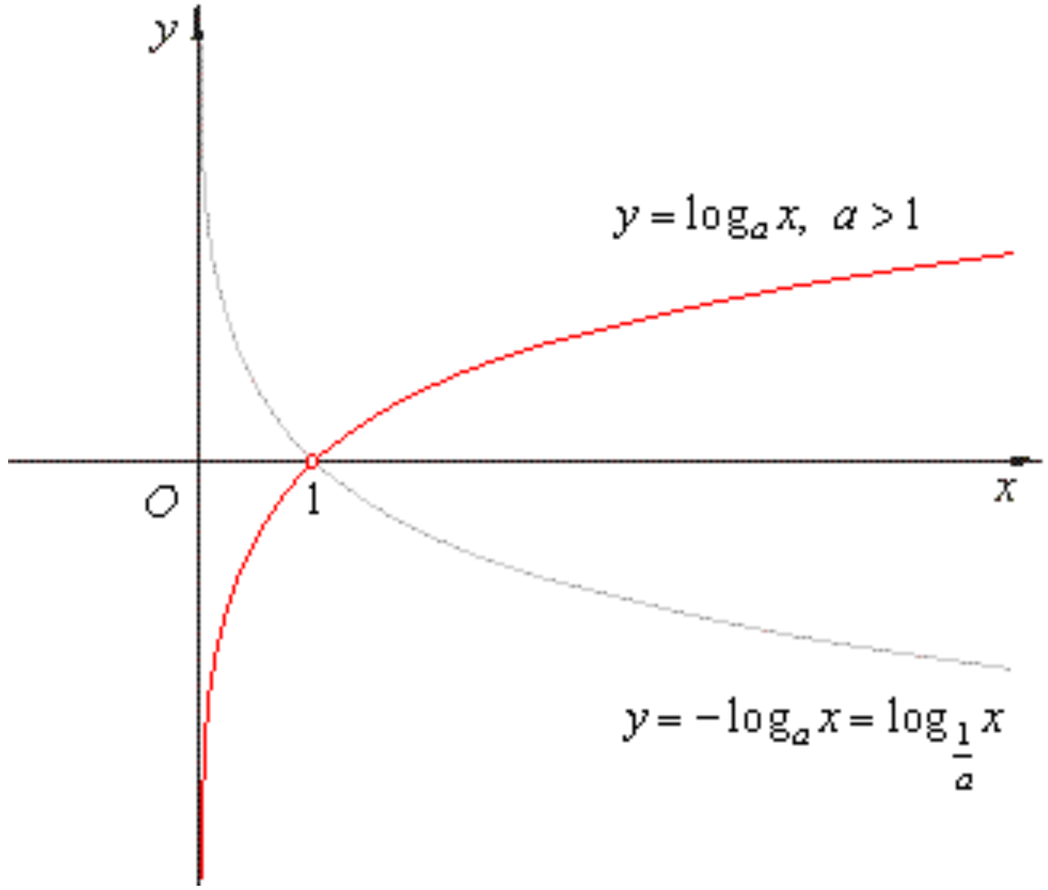
- Cross Entropy

Entropy (in information theory)

= amount of information in an event

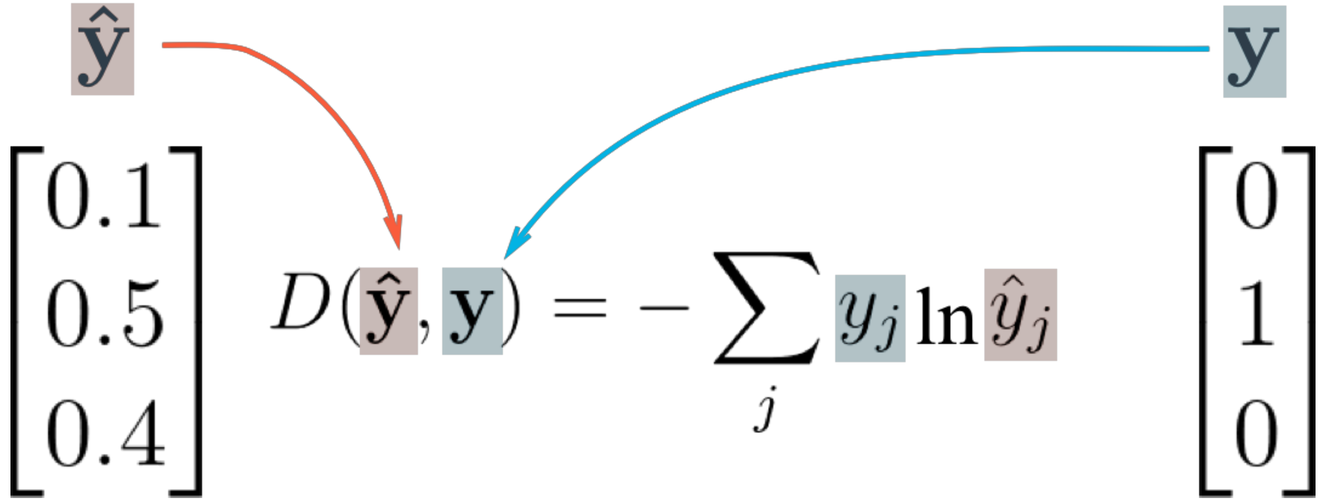
= amount of surprise

$$h[x] = \mathbb{E}[-\log P(x)]$$



Loss functions

- Cross Entropy




```
def cross_entropy_loss(targets, outputs):
    return -sum(targets * log(outputs))
```

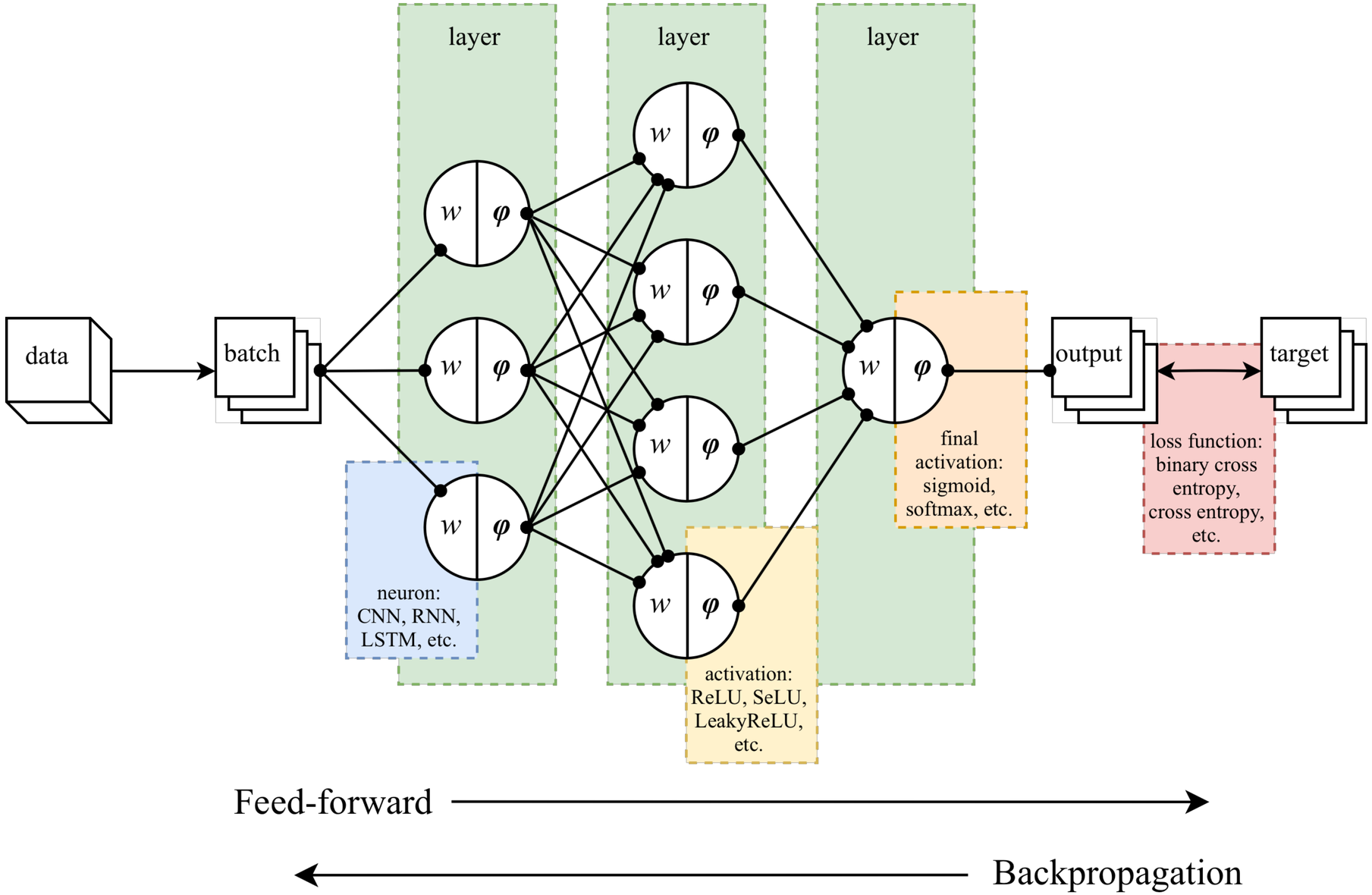

https://towardsdatascience.com/deep-learning-concepts-part-1-ea0b14b234c8

Loss functions

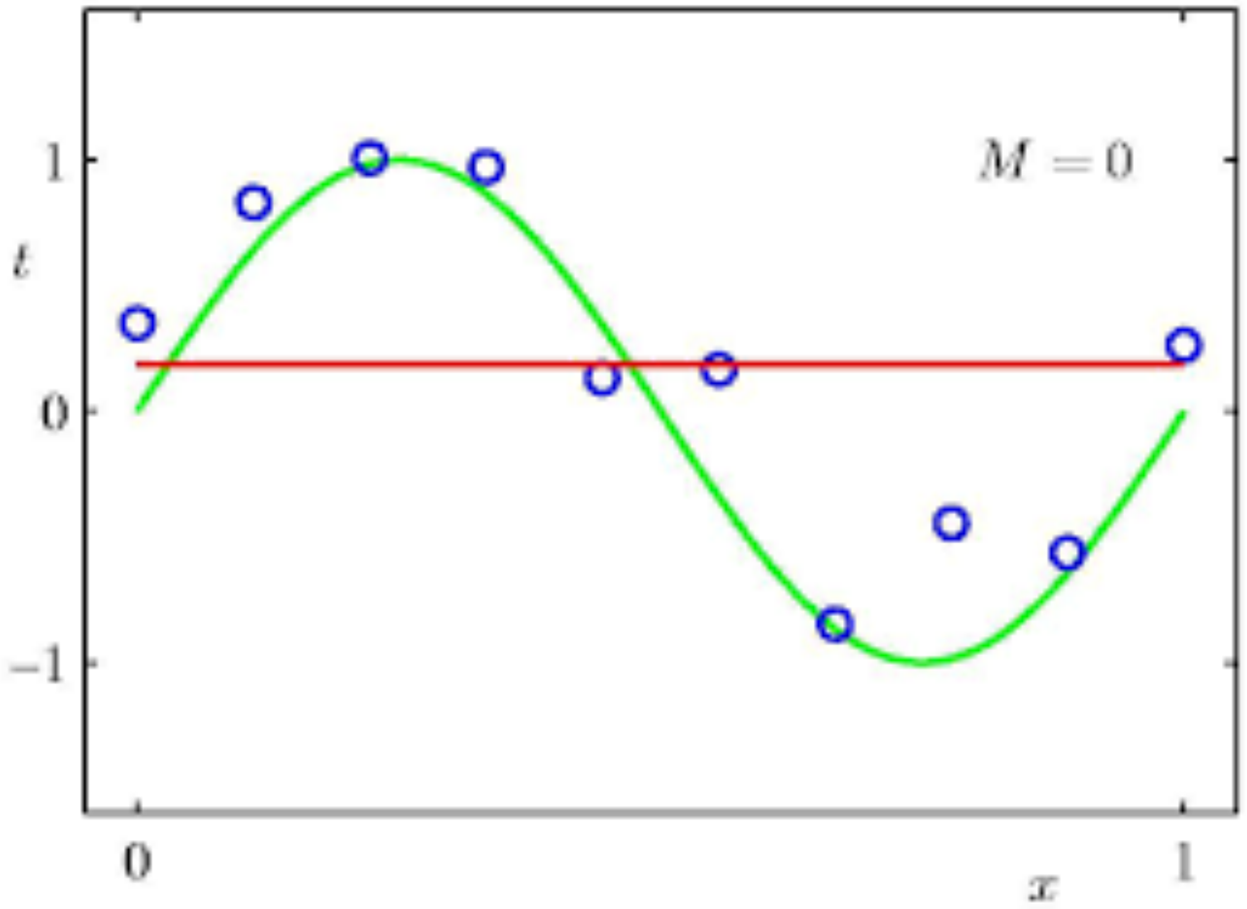
- Binary Cross Entropy

$$-\frac{1}{n}\sum_{i=1}^{n}[y_i ln(\hat{y}_i) + (1-y_i) ln(1-\hat{y}_i)]$$

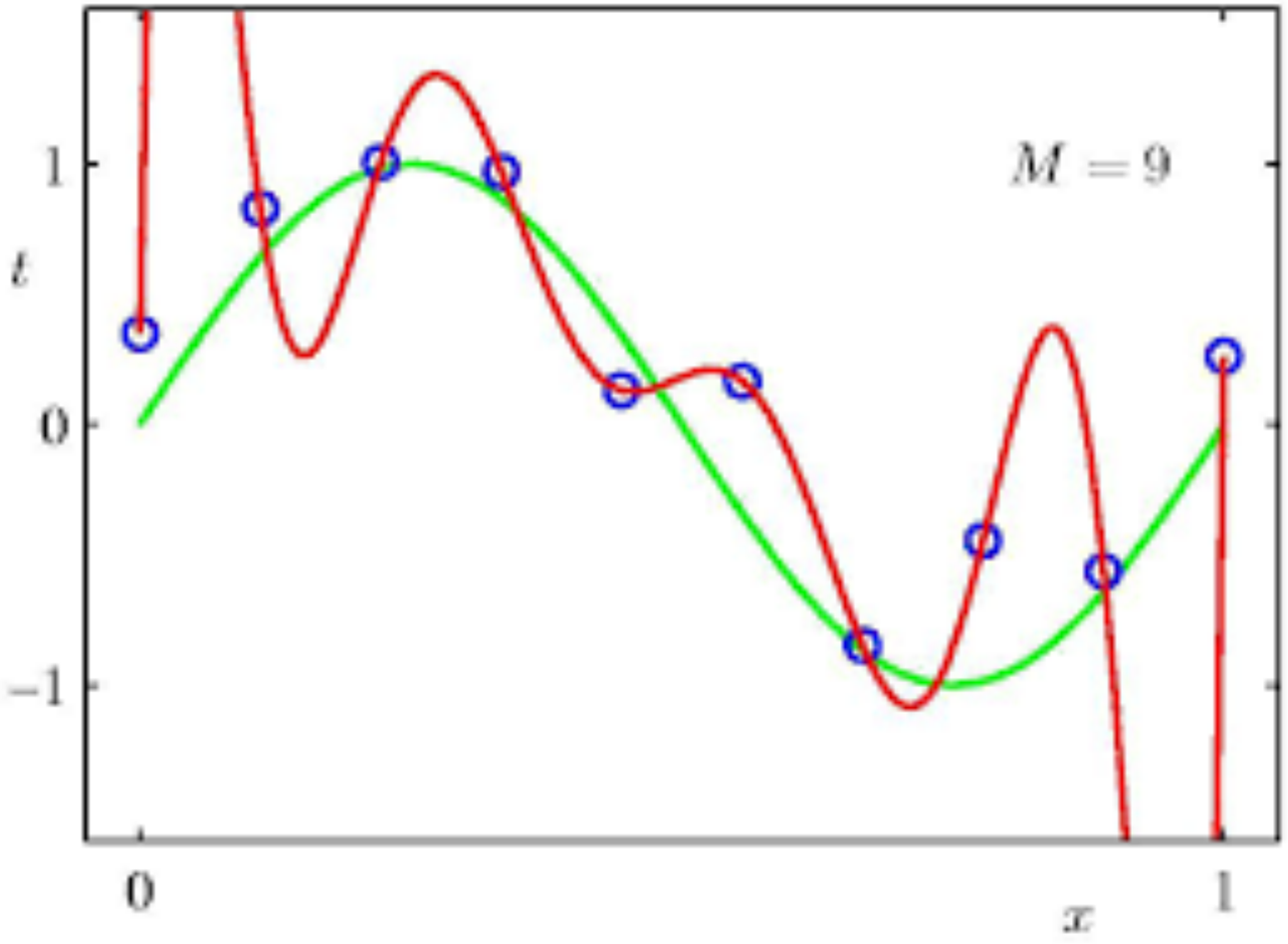
```
def binary_cross_entropy_loss(targets, outputs):
    return -mean(targets * log(outputs) + (1 - targets) * log(1 - outputs))
```


layer
layer
layer
data
batch
w
φ
output
target
neuron:
CNN, RNN,
LSTM, etc.
activation:
ReLU, SeLU,
LeakyReLU,
etc.
final
activation:
sigmoid,
softmax, etc.
loss function:
binary cross
entropy,
cross entropy,
etc.
Feed-forward
Backpropagation

Regularization methods



Regularization methods



Regularization methods

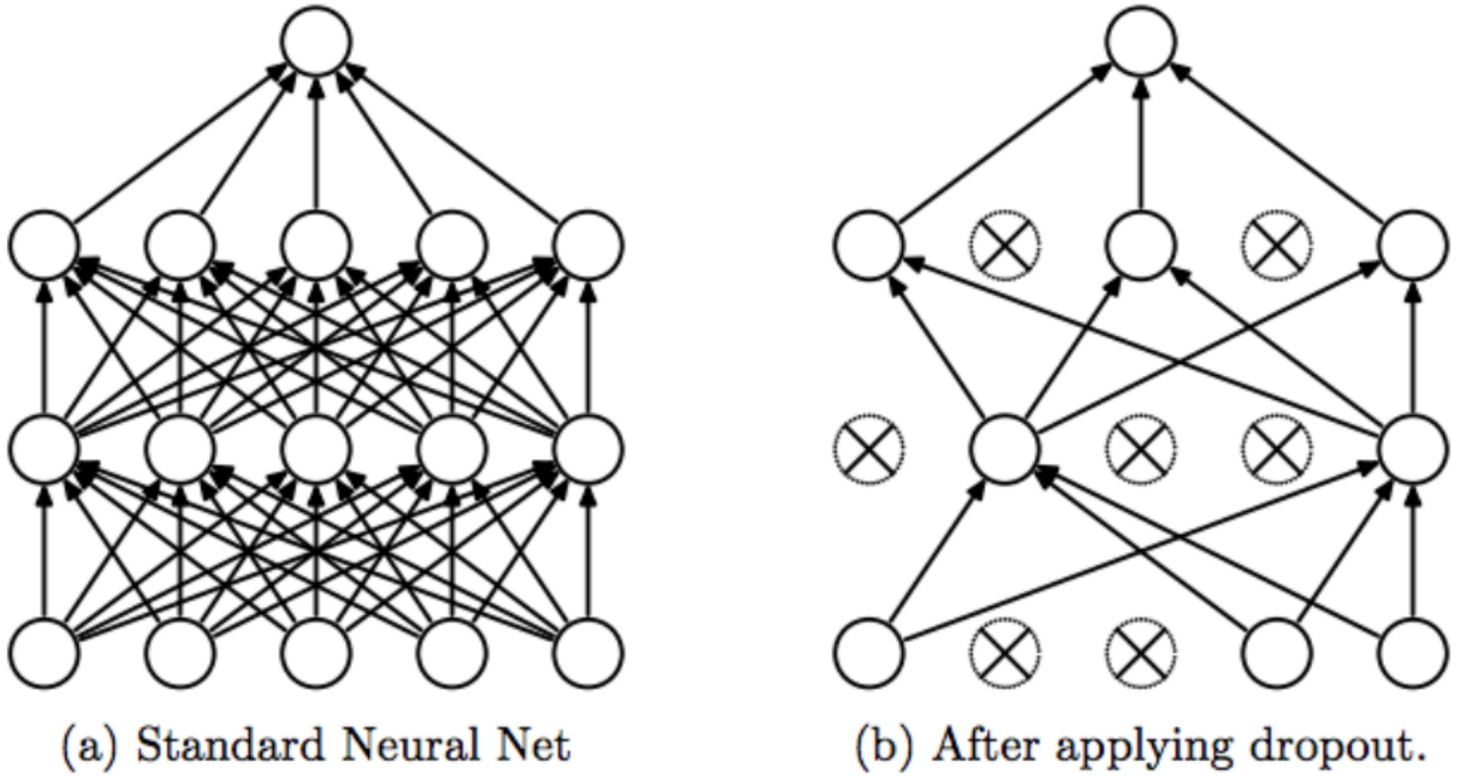
- Weight decay

$$W \leftarrow W - \lambda(\frac{\partial L}{\partial W} + \gamma \| W \|)$$

```
def update_weights(weights, gradients, learning_rate, weight_decay):
    weight_penalty = weight_decay * sum(sqrt(weights ** 2))
    return weights - learning_rate * (gradients @ weights + weight_penalty)
```

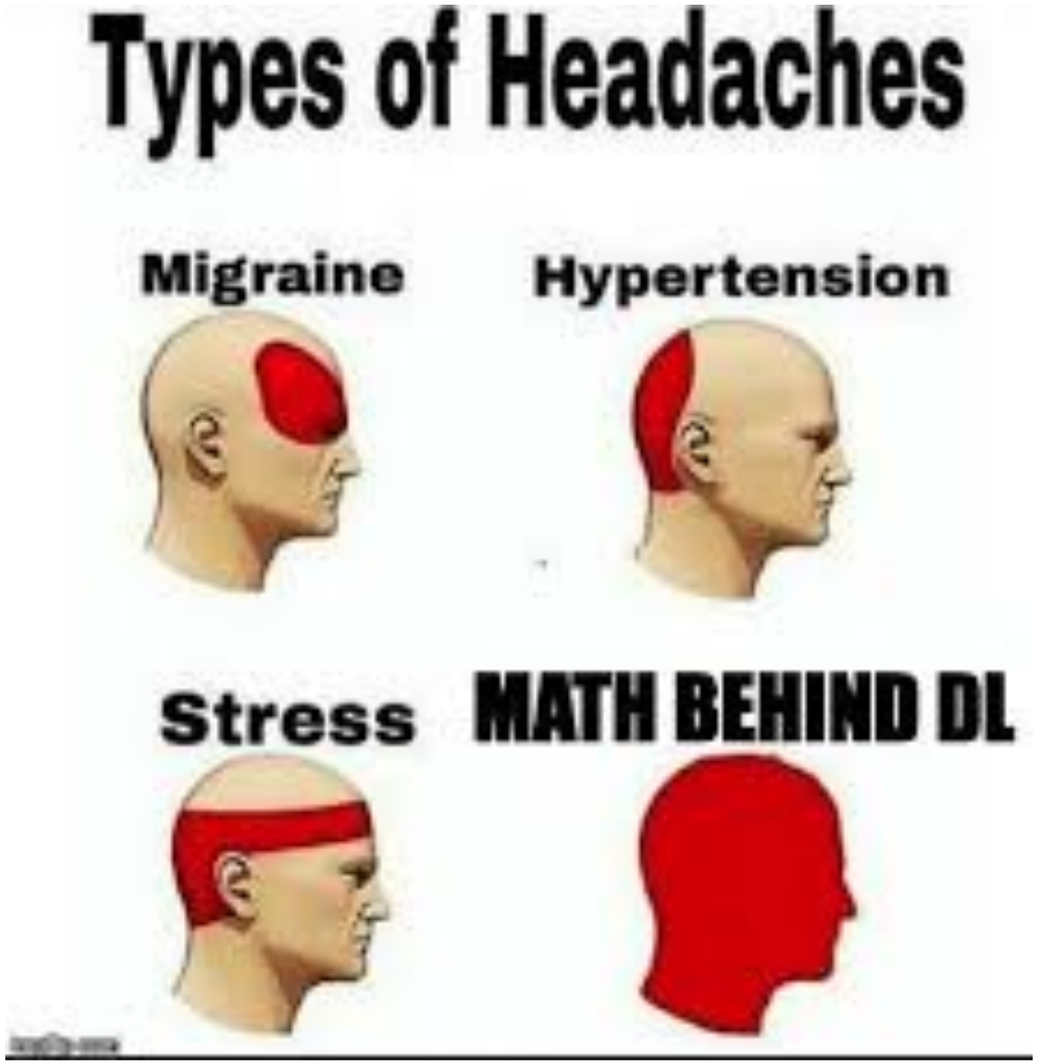
Regularization methods

- Dropout



(a) Standard Neural Net

(b) After applying dropout.


Types of Headaches
Migraine
Hypertension
Stress
MATH BEHIND DL

Hello PyTorch!

PyTorch, Anaconda, Jupyter Notebook

Installation

PyTorch source code structure

Tensors

Modules

Image classification with PyTorch

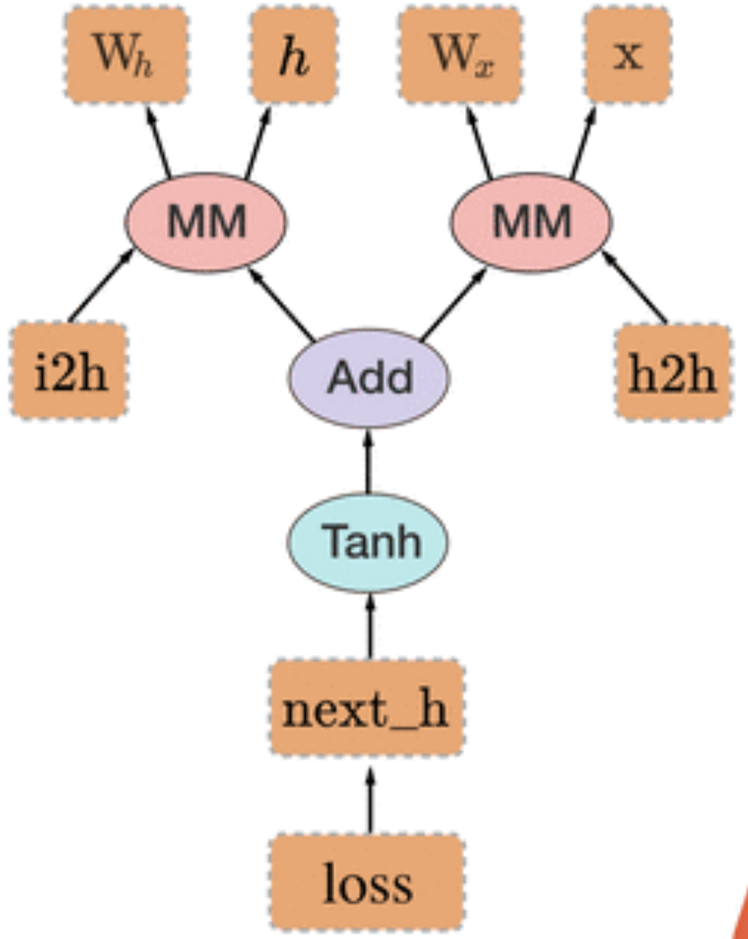
02

PyTorch

- Deep learning framework
  - TensorFlow, Keras, Torch, Chainer, MXNet
- Python-native, NumPy-friendly
- Dynamic graphs
- Official website: https://pytorch.org/
- Documentation: https://pytorch.org/docs/stable/index.html
- Source code: https://github.com/pytorch/pytorch

Back-propagation
uses the dynamically created graph

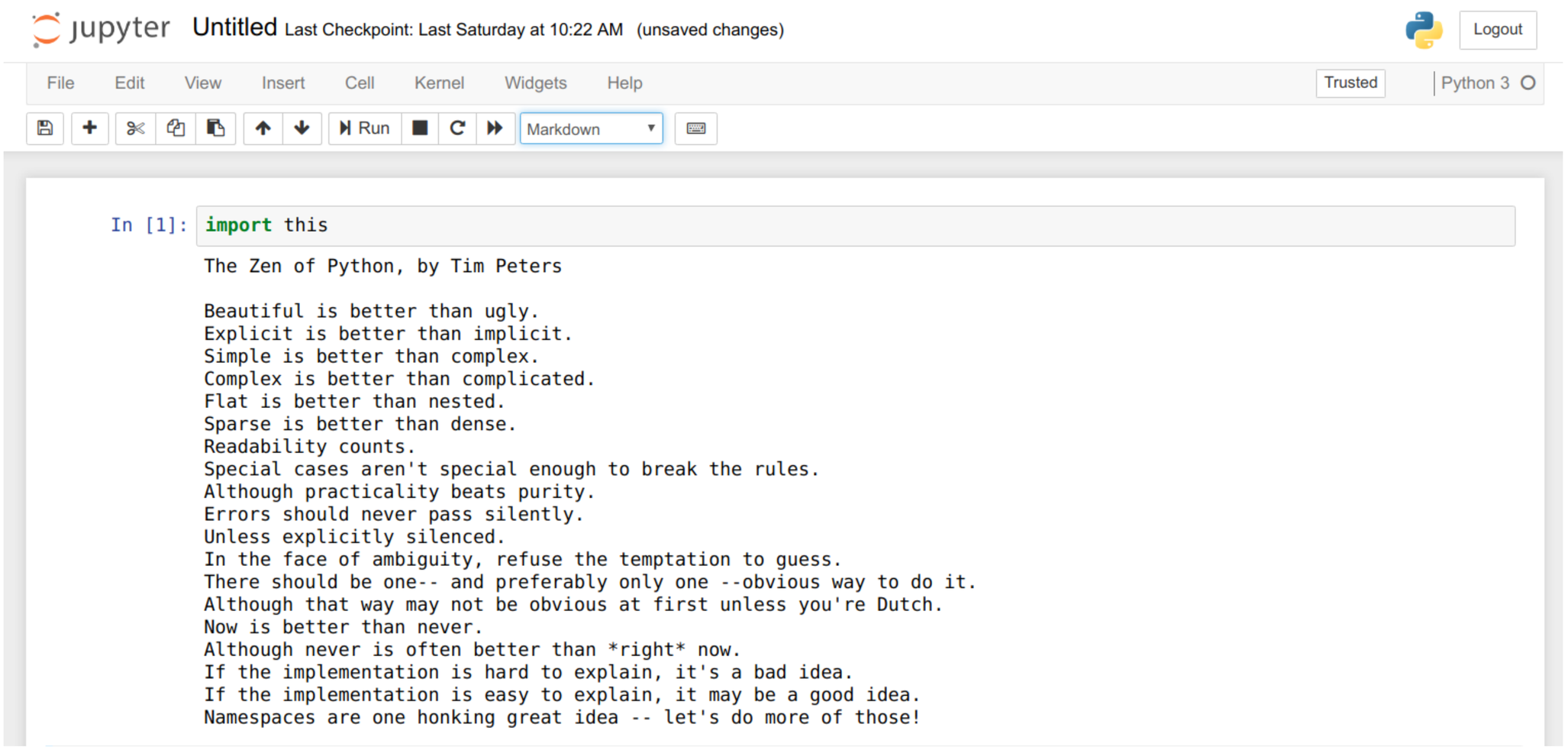
```
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
W_h = torch.randn(20, 20)
W_x = torch.randn(20, 10)

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()

loss = next_h.sum()
loss.backward()  # compute gradients!
```



Anaconda

- Python package manager + virtual environment manager
- Accessed via the terminal (Ubuntu and macOS) or Anaconda Prompt (Windows)
- https://www.anaconda.com/download/

Jupyter Notebook



Installation

1. Install Anaconda from https://www.anaconda.com/download/
2. Run the installation commands:
   - Ubuntu:

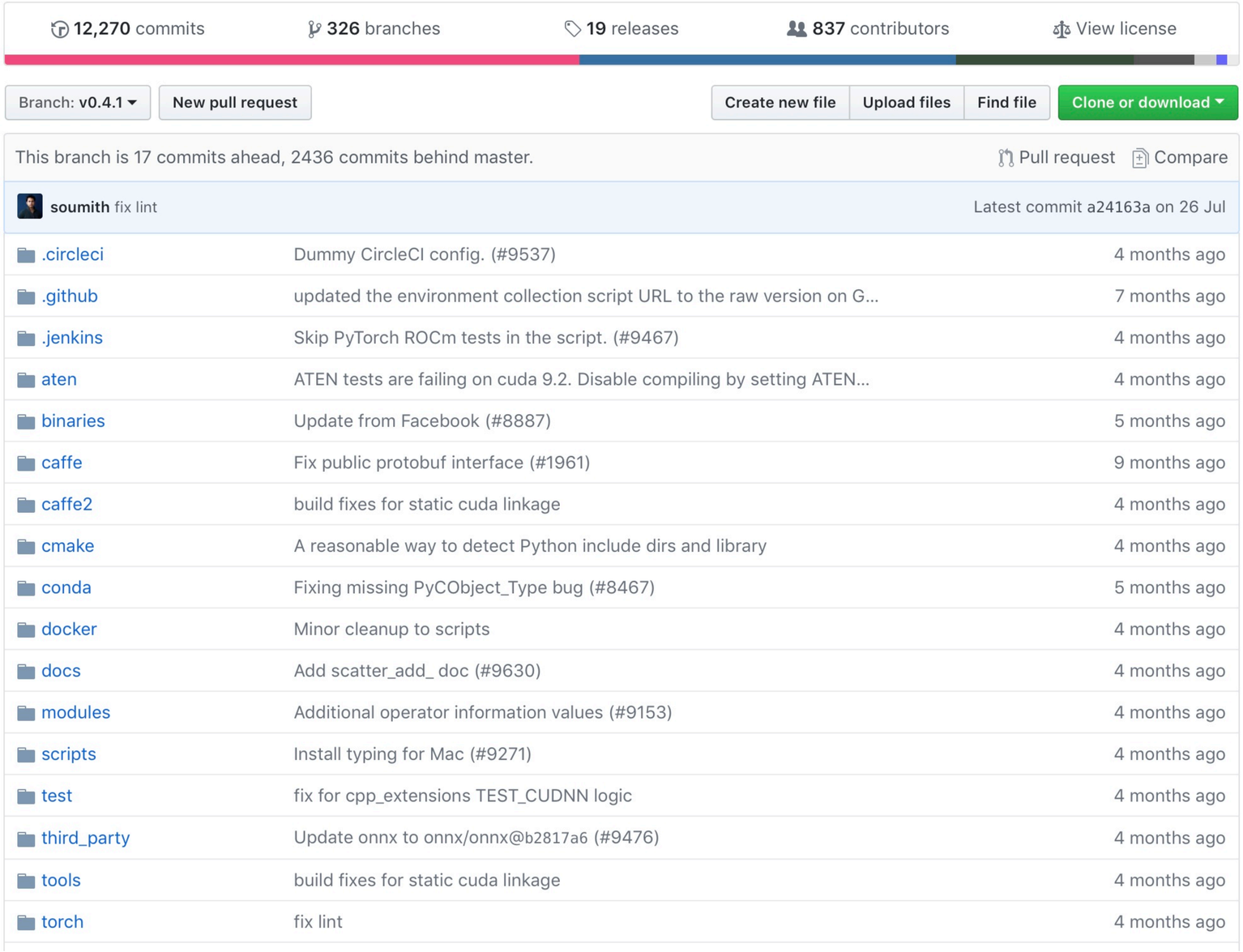
```
conda install -y pytorch-cpu torchvision-cpu -c pytorch
```

   - macOS:

```
conda install -y pytorch torchvision -c pytorch
```

   - Windows:

```
conda install -y pytorch-cpu -c pytorch
pip3 install torchvision
```


12,270 commits
326 branches
19 releases
837 contributors
View license
Branch: v0.4.1
New pull request
Create new file
Upload files
Find file
Clone or download
This branch is 17 commits ahead, 2436 commits behind master.
Pull request
Compare
soumith fix lint
Latest commit a24163a on 26 Jul
.circleci Dummy CircleCI config. (#9537) 4 months ago
.github updated the environment collection script URL to the raw version on G... 7 months ago
.jenkins Skip PyTorch ROCm tests in the script. (#9467) 4 months ago
aten ATEN tests are failing on cuda 9.2. Disable compiling by setting ATEN... 4 months ago
binaries Update from Facebook (#8887) 5 months ago
caffe Fix public protobuf interface (#1961) 9 months ago
caffe2 build fixes for static cuda linkage 4 months ago
cmake A reasonable way to detect Python include dirs and library 4 months ago
conda Fixing missing PyCObject_Type bug (#8467) 5 months ago
docker Minor cleanup to scripts 4 months ago
docs Add scatter_add_ doc (#9630) 4 months ago
modules Additional operator information values (#9153) 4 months ago
scripts Install typing for Mac (#9271) 4 months ago
test fix for cpp_extensions TEST_CUDNN logic 4 months ago
third_party Update onnx to onnx/onnx@b2817a6 (#9476) 4 months ago
tools build fixes for static cuda linkage 4 months ago
torch fix lint 4 months ago

Branch: v0.4.1 ▾ **pytorch** / **torch** /

Create new file | Upload files | Find file | History

This branch is 17 commits ahead, 2436 commits behind master. Pull request Compare

 **soumith** fix lint — Latest commit a24163a on 26 Jul

..		
_thnn	Update from Facebook (#8887)	5 months ago
autograd	Fix segmentation fault in grad_fn (#9292)	4 months ago
backends	Move _cudnn_init_dropout_state to TensorOptions and enable cuDNN drop...	5 months ago
contrib	Add code for TensorBoard visualization of JIT GraphExecutors (#8050)	6 months ago
csrc	fix small literals being flushed to 0 by std::to_string	4 months ago
cuda	Move nccl scatter and gather to C++ (#9117)	4 months ago
distributed	Distributed Data Parallel Module Implementation (#8584)	5 months ago
distributions	cherry pick #9500 and #9590 into 0.4.1 (#9599)	4 months ago
for_onnx	Codemod Toffee -> ONNX, toffee -> onnx. Change file names to match	a year ago
jit	move batchop import to init to avoid debugging confusions (#9425)	4 months ago
legacy	cherry pick #9500 and #9590 into 0.4.1 (#9599)	4 months ago
lib	Delete flag from THTensor. (#9494)	4 months ago
multiprocessing	Eliminate storage views. (#9466)	4 months ago
nn	Fixed a missing '=' in LPPoolNd repr function (#9629)	4 months ago
onnx	Add support for .norm() pytorch onnx export and ReduceL1/ReduceL2 caf...	4 months ago
optim	cherry pick #9500 and #9590 into 0.4.1 (#9599)	4 months ago
sparse	Delete dead Tensor code paths (#5417)	9 months ago
testing	Codemod to update our codebase to 0.4 standard (#6641)	7 months ago
utils	fix lint	4 months ago

Branch: v0.4.1 ▾ **pytorch** / torch / **nn** /

Create new file | Upload files | Find file | History

This branch is 17 commits ahead, 2436 commits behind master. Pull request Compare

 **vmirly** and **SsnL** Fixed a missing '=' in LPPoolNd repr function (#9629) ··· Latest commit 76c16a5 on 21 Jul

..		
_functions	Implement 2D and 3D alpha_dropout (#9073)	4 months ago
backends	Implement MarginRankingLoss as native function and add reduce=True ar...	8 months ago
modules	Fixed a missing '=' in LPPoolNd repr function (#9629)	4 months ago
parallel	Distributed Data Parallel Module Implementation (#8584)	5 months ago
utils	cherry pick #9500 and #9590 into 0.4.1 (#9599)	4 months ago
__init__.py	Add weight normalization implementation (#1945)	a year ago
functional.py	Implement 2D and 3D alpha_dropout (#9073)	4 months ago
grad.py	Codemod to update our codebase to 0.4 standard (#6641)	7 months ago
init.py	Fix #8692 (#8699)	5 months ago
parameter.py	Fix serialization for Parameters (#8633)	5 months ago

Branch: v0.4.1 ▾ **pytorch** / torch / nn / **modules** /

Create new file | Upload files | Find file | History

This branch is 17 commits ahead, 2436 commits behind master. Pull request Compare

 **vmirly** and **SsnL** Fixed a missing '=' in LPPoolNd repr function (#9629) ··· Latest commit 76c16a5 on 21 Jul

..		
__init__.py	Implement 2D and 3D alpha_dropout (#9073)	4 months ago
activation.py	copy paste documentation error fixed in Softmin (#7324)	6 months ago
adaptive.py	Add utf-8 header to Python file with Unicode. (#8131)	5 months ago
batchnorm.py	Fix loading 0.4 BN checkpoints (#9004)	5 months ago
container.py	Add ModuleDict and ParameterDict containers (#8463)	4 months ago
conv.py	Update extension docs, fix Fold/Unfold docs (#9239)	4 months ago
distance.py	implement TripletMarginLoss as a native function (#5680)	8 months ago
dropout.py	Implement 2D and 3D alpha_dropout (#9073)	4 months ago
fold.py	Update extension docs, fix Fold/Unfold docs (#9239)	4 months ago
instancenorm.py	Fix loading 0.4 BN checkpoints (#9004)	5 months ago
linear.py	Fix the comments: code and comments dimensions mis-match (#9070)	5 months ago
loss.py	docs fixes (#9607)	4 months ago
module.py	Fix loading 0.4 BN checkpoints (#9004)	5 months ago
normalization.py	Update extension docs, fix Fold/Unfold docs (#9239)	4 months ago

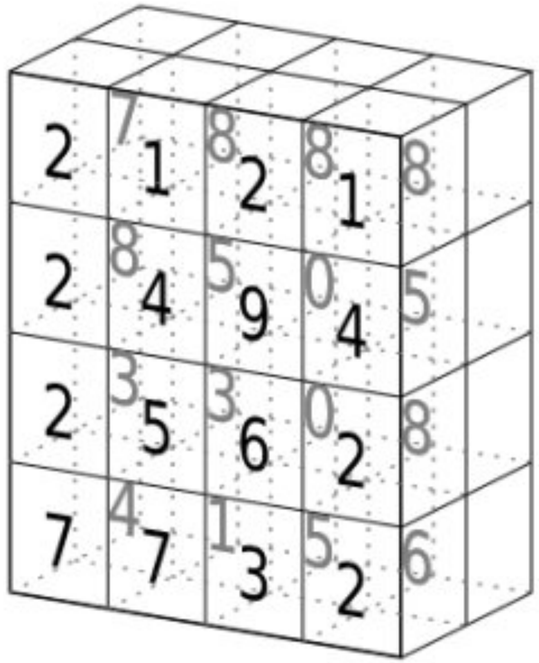
Tensors: PyTorch's core data structure

't'
'e'
'n'
's'
'o'
'r'

tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]

https://www.kdnuggets.com/2018/05/pytorch-tensor-basics.html

Tensors: PyTorch's core data structure

- Creating tensors
  - with pre-existing data: `torch.tensor()`
  - with specific size: `torch.ones()`, `torch.zeros()`, `torch.rand()`

```
import torch

tensor = torch.tensor([[1, 2, 3], [4, 5, 6]])
print(tensor)
zeros = torch.zeros(2, 3)
print(zeros)
```

Tensors: PyTorch's core data structure

torch.tensor(data, dtype=None, device=None, requires_grad=False) → Tensor

Constructs a tensor with data.

- data (array_like) – Initial data for the tensor. Can be a list, tuple, NumPy ndarray, scalar, and other types.
- dtype (torch.dtype, optional) – the desired data type of returned tensor. Default: if None, infers data type from data.
- device (torch.device, optional) – the desired device of returned tensor. Default: if None, uses the current device for the default tensor type. device will be the CPU for CPU tensor types and the current CUDA device for CUDA tensor types.

Tensors: PyTorch's core data structure

torch.tensor(data, dtype=None, device=None, requires_grad=False) → Tensor

- requires_grad (bool, optional) – If autograd should record operations on the returned tensor. Default: False.

Tensors: PyTorch's core data structure

Data type	dtype	CPU tensor
32-bit floating point	`torch.float32` or `torch.float`	`torch.FloatTensor`
64-bit floating point	`torch.float64` or `torch.double`	`torch.DoubleTensor`
16-bit floating point	`torch.float16` or `torch.half`	`torch.HalfTensor`
8-bit integer (unsigned)	`torch.uint8`	`torch.ByteTensor`
8-bit integer (signed)	`torch.int8`	`torch.CharTensor`
16-bit integer (signed)	`torch.int16` or `torch.short`	`torch.ShortTensor`
32-bit integer (signed)	`torch.int32` or `torch.int`	`torch.IntTensor`
64-bit integer (signed)	`torch.int64` or `torch.long`	`torch.LongTensor`

Tensors: PyTorch's core data structure

torch.zeros(*sizes, out=None, dtype=None, layout=torch.strided, device=None, requires_grad=False) → Tensor

Returns a tensor filled with the scalar value 0, with the shape defined by the variable argument sizes.

- sizes (int...) – a sequence of integers defining the shape of the output tensor. Can be a variable number of arguments or a collection like a list or tuple.
- out (Tensor, optional) – the output tensor
- dtype (torch.dtype, optional) – the desired data type of returned tensor. Default: if None, uses a global default.

Tensors: PyTorch's core data structure

torch.zeros(*sizes, out=None, dtype=None, layout=torch.strided, device=None, requires_grad=False) → Tensor

- layout (torch.layout, optional) – the desired layout of returned Tensor. Default: torch.strided.
- device (torch.device, optional) – the desired device of returned tensor. Default: if None, uses the current device for the default tensor type (see torch.set_default_tensor_type()). device will be the CPU for CPU tensor types and the current CUDA device for CUDA tensor types.
- requires_grad (bool, optional) – If autograd should record operations on the returned tensor. Default: False.

Tensors: PyTorch's core data structure

- Transposing tensors: Tensor.t()

```
tensor = torch.tensor([[1, 2, 3], [4, 5, 6]])

transposed = tensor.t()
print(transposed)
```

Tensors: PyTorch's core data structure

- Reshaping tensors

```
tensor = torch.Tensor([[1, 2, 3], [4, 5, 6]])

reshaped = tensor.view(3, 2)
print(reshaped)

reshaped = tensor.view(6, 1)
print(reshaped)

reshaped = tensor.view(2, -1)
print(reshaped)
```

Tensors: PyTorch's core data structure

Tensor.t() → Tensor

Expects self to be a matrix (2-D tensor) and transposes dimensions 0 and 1.

Tensor.view(*args) → Tensor

Returns a new tensor with the same data as the self tensor but of a different size.

- args (torch.Size or int...) – the desired size

Tensors: PyTorch's core data structure

- Broadcasting

```
tensor = torch.Tensor([[1, 2, 3], [4, 5, 6]])

changed = tensor + 1
print(changed)
```

Tensors: PyTorch's core data structure

- In-place operations (marked by an underscore)

```
tensor = torch.Tensor([[1, 2, 3], [4, 5, 6]])

changed = tensor - 1
tensor.add_(1)

print(changed)
print(tensor)
```

Modules: the building block of PyTorch

torch.nn.Module

- Base class for all neural network modules
- The Python magic method __call__() executes forward() method
- If backward() method is not defined, the backward is automatically created by forward()
- Submodules can be assigned as regular attributes

Modules: the building block of PyTorch

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
       x = F.relu(self.conv1(x))
       return F.relu(self.conv2(x))
```

Image classification with PyTorch

- FashionMNIST
  - Zalando's clothing product images
  - 28-pixel-square grayscale images
  - 60k examples for training, 10k samples for testing
  - 10 classes
  - https://github.com/zalandoresearch/fashion-mnist

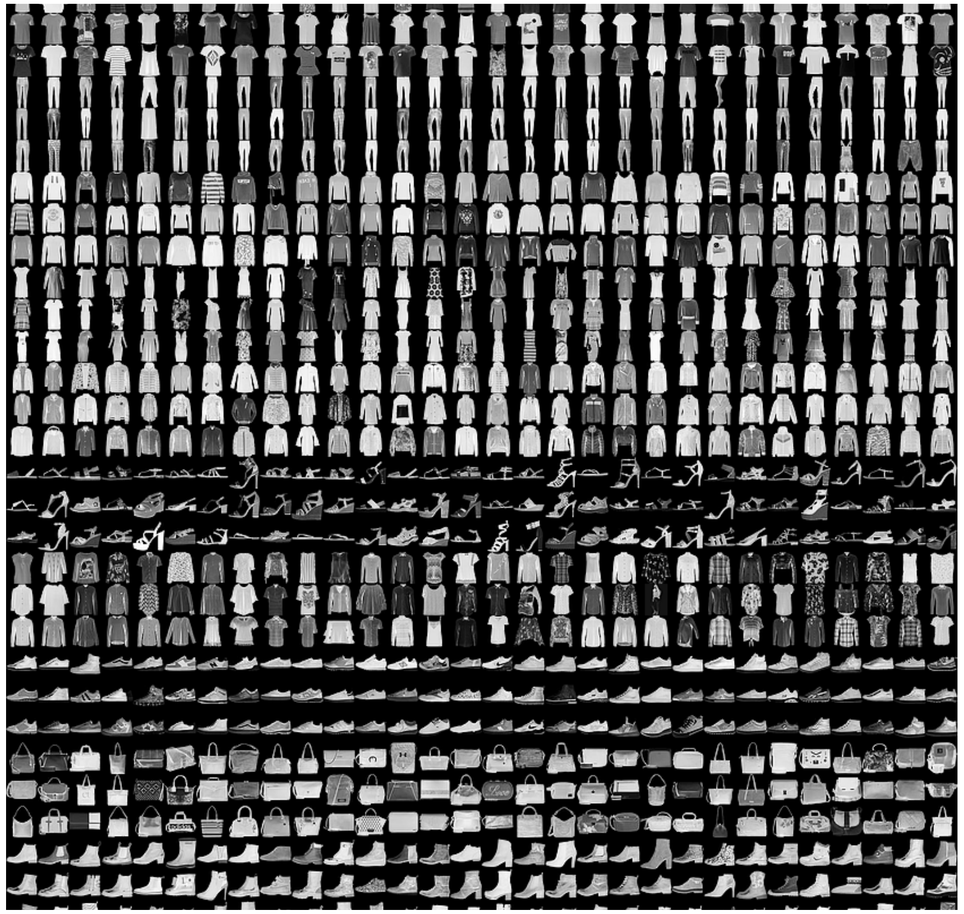


Image classification with PyTorch

- ResNet
  - He et al., Deep Residual Learning for Image Recognition. 2015.
  - torchvision.models.resnet
    - The PyTorch reference implementation of the ResNet
    - A great example of building PyTorch models with Python OOP

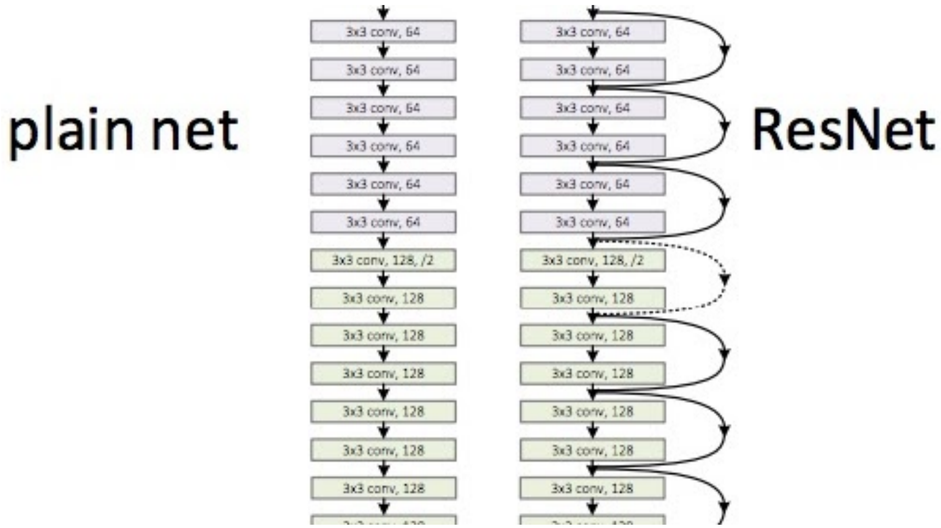


Image classification with PyTorch

git clone or download https://github.com/juneoh/sample_pytorch_project

- Dockerfile — if you want to use Docker.
- README.md — the repository description.
- main.py — the main code.
- requirements.txt — the package requirements to run this example, for pip.

Image classification with PyTorch

- The process
    1. Prepare the data: training, validation, test.
    2. Create the model and the loss function.
    3. Create the optimizer and attach it to the model.
    4. For each epoch, train, evaluate and save model.
    5. Finally, evaluate the model on the test dataset.

Image classification with PyTorch

- The process
  - 🤔 Why not use cross-validation?

Yoshua Bengio, My lab has been one of the three that started the deep learning approach, back in 2006, along with Hinton's...

Answered Jan 20, 2016 · Upvoted by Naran Bayanbat, MSCS with focus in machine learning and Boxun Zhang, Data Scientist at Spotify; PhD in Computer Science · Author has **172** answers and **3.6m** answer views

We mostly have large datasets when it is not worth the trouble to do something like k-fold cross-validation. We just use a train/valid/test split. Cross-validation becomes useful when the dataset is tiny (like hundreds of examples), but then you can't typically learn a complex model.

36.6k Views · View Upvoters

Code review

Thank you!