Advanced Deep Learning Architectures

COMP 5214 & ELEC 5680

Instructor: Dr. Qifeng Chen
https://cqf.io

Course

Website

- https://course.cse.ust.hk/comp5214

Location

- Zoom 960 8484 2280 (Passcode: 5214)

Time

- TuTh 12:00 PM - 01:20 PM

Who we are

Instructor: Qifeng Chen (cqf@ust.hk)

Office hours: 3 - 4pm Tue on Zoom 558 348 8283

TAs:

Chenyang Lei (cleiaa@connect.ust.hk)

Yue Wu (ywudg@connect.ust.hk)

Samuel Cahyawijaya (scahyawijaya@connect.ust.hk)

Benson Xu (jlxu@connect.ust.hk)

Logistics

- Three Assignments (30%)
  - First assignment will be out in about two weeks
- Midterm (35%)
  - The midterm will be in class
- Project Presentation (5%)
- Final Course Project (30%)
  - Proposal (5%) + Milestone (5%) + Final Report (20%)

Syllabus

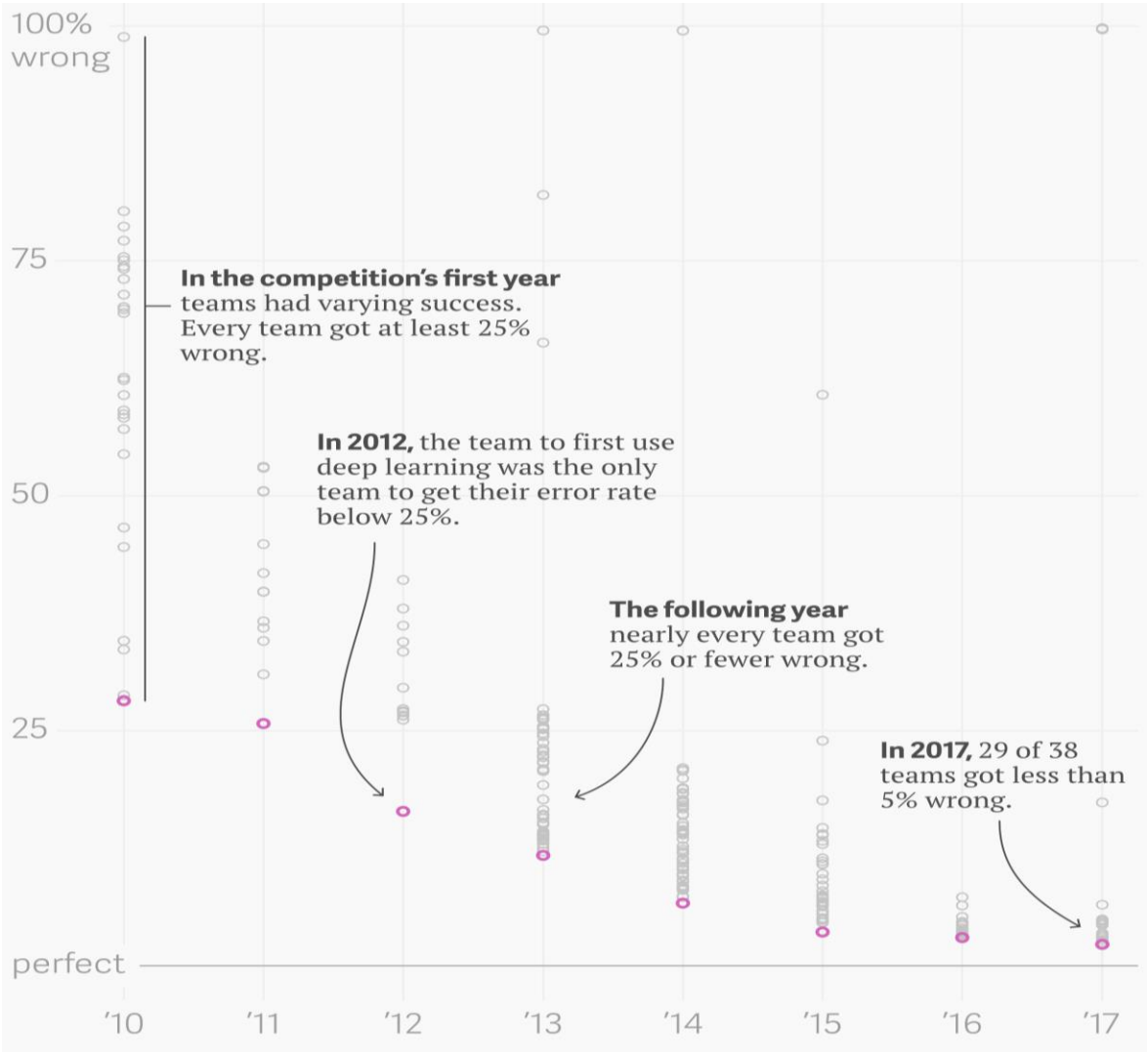
- Overview of deep learning: Basic architectures (CNN, RNN), Backpropagation, Loss functions
- Neural networks for image and video recognition tasks
- Neural networks for image and video processing tasks
- Deep 3D learning for point clouds, meshes, and volumetric data
- Deep 3D learning for stereo and multi-view data
- Graph neural networks for graph processing and analysis
- Sequential modeling and signal processing
- Deep generative models: Normalizing flow, GAN, Pix2pix, and CycleGAN
- Efficient neural networks
- Neural architecture search
- Final project presentation and project report submission


Deep Learning

Classify Images

http://www.image-net.org/

Classify Images



Yanofsky, Quartz
https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/

Detect and Segment Objects

https://github.com/matterport/Mask_RCNN

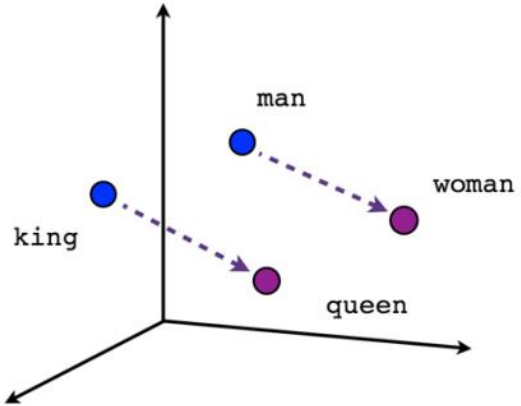
Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

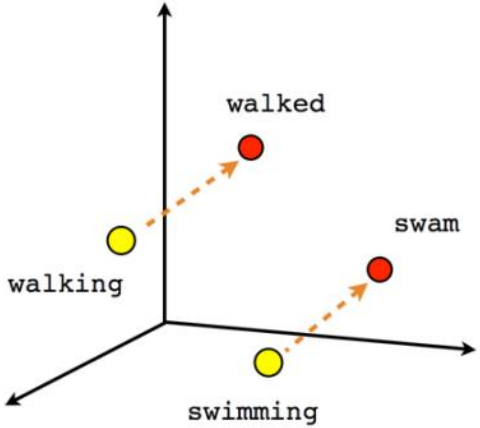
Synthesize Faces

Karras et al, ICLR 2018

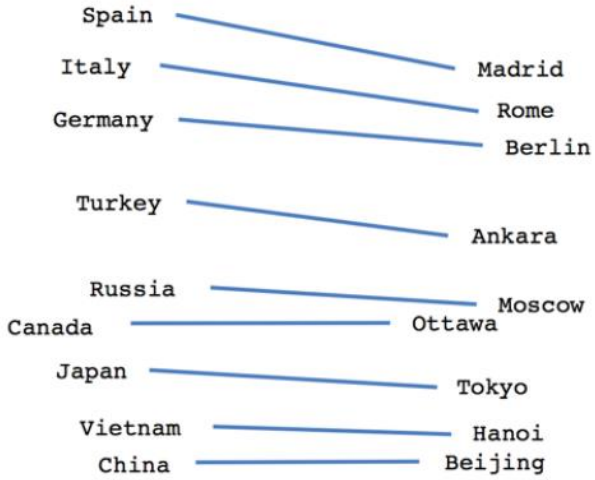
Analogies



Male-Female



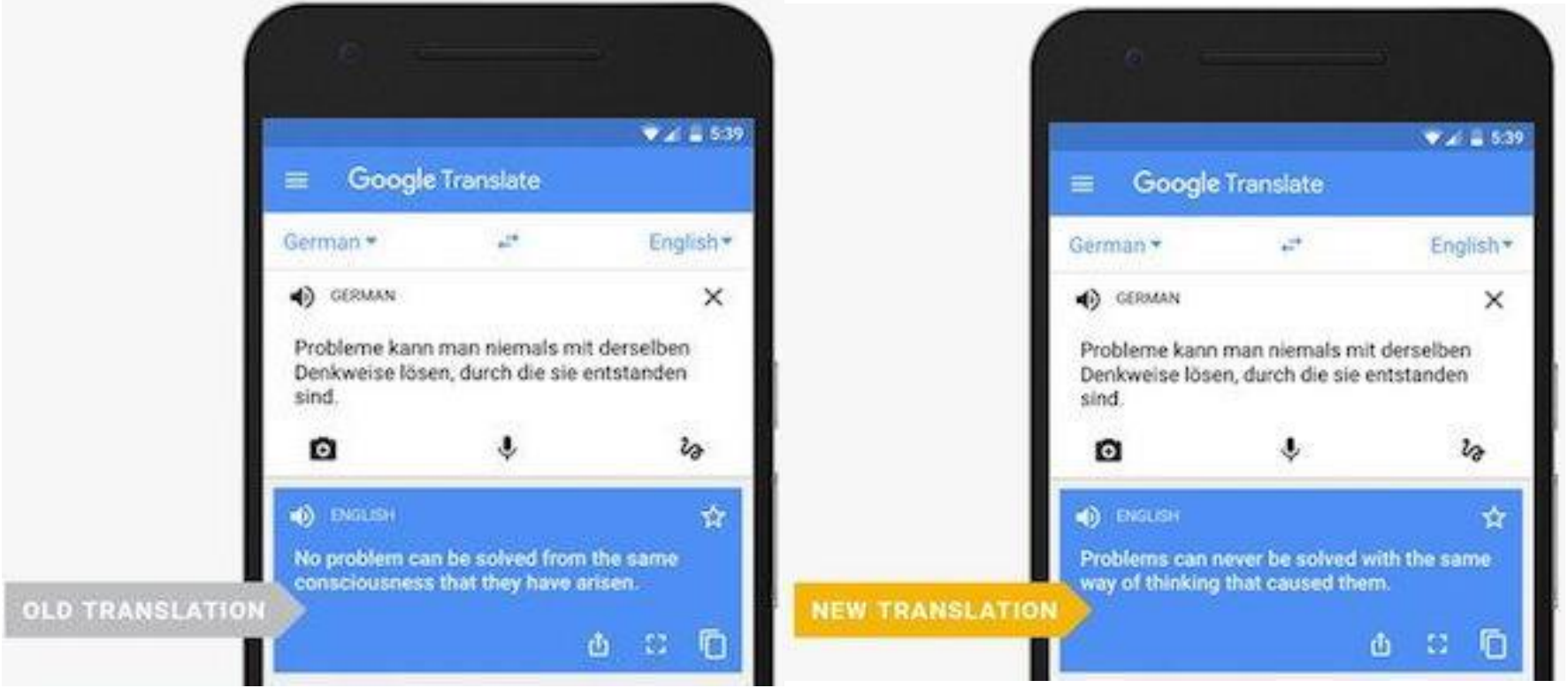
Verb tense



Country-Capital

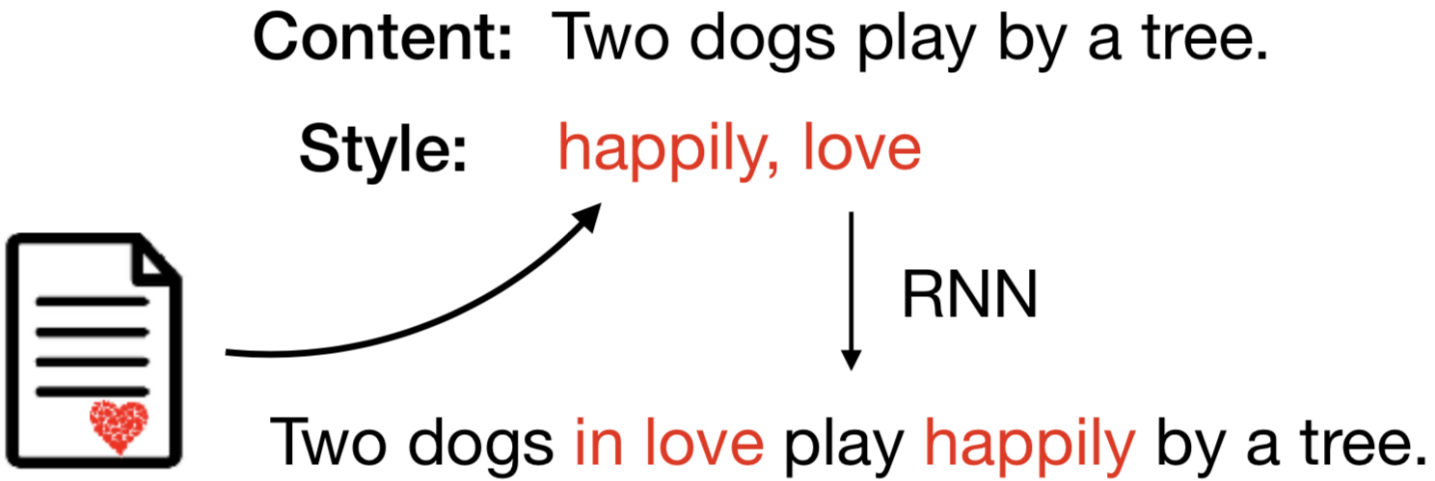
https://www.tensorflow.org/tutorials/word2vec

Machine Translation



https://www.pcmag.com/news/349610/google-expands-neural-networks-for-language-translation

Text synthesis



Li et al, NACCL, 2018

Question answering

Q: "What's her mustache made of?"

Vision Feature Extractor

Text Feature Extractor

Question Type: "Subordinate Object Recognition"

Question Type Guided Attention

Combine

Predictor

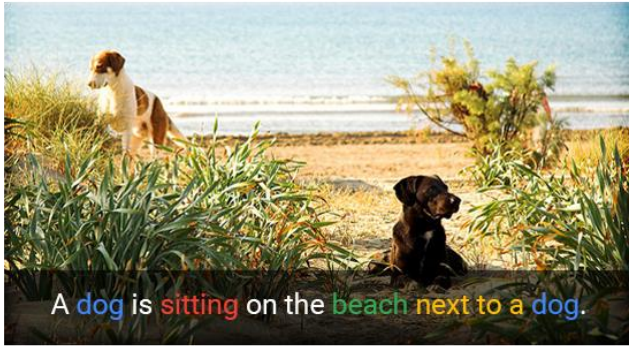
A: "Banana"

Shi et al, 2018, Arxiv

Image captioning

Human captions from the training set

Automatically captioned



Shallue et al, 2016
https://ai.googleblog.com/2016/09/show-and-tell-image-captioning-open.html

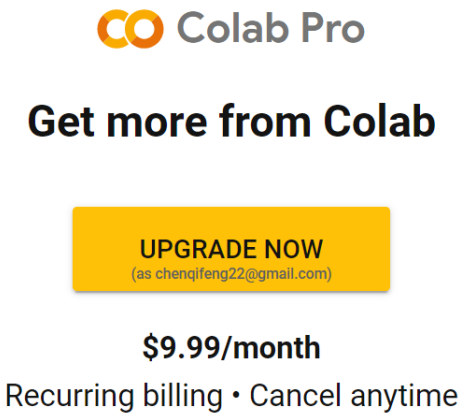

Software

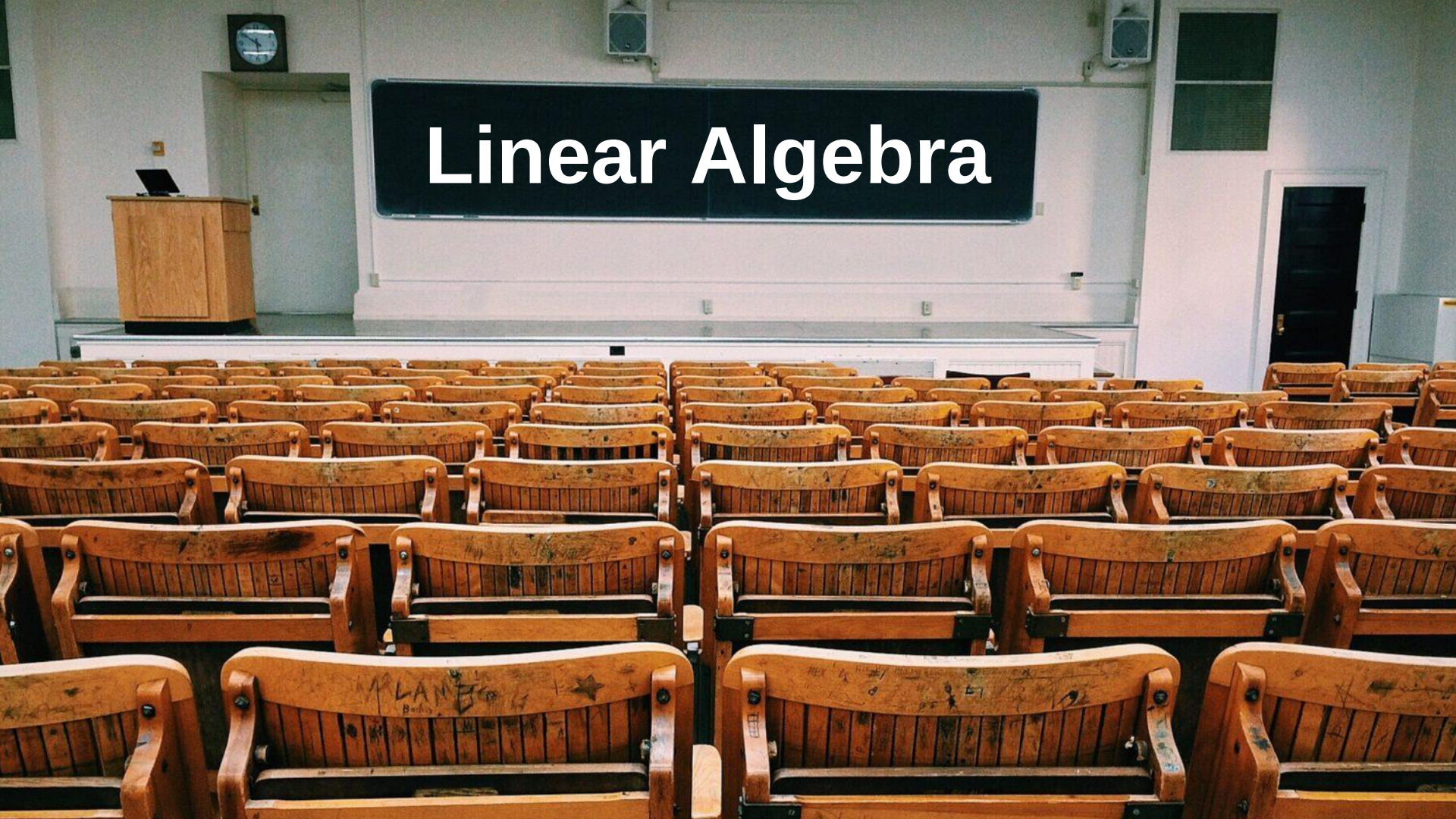
Tools

- **Python**
  - Everyone is using it in machine learning & data science
  - Conda package manager (for simplicity)
- **Jupyter**
  - So much easier to keep track of your experiments
  - Obviously you should put longer code into modules

Colab

- Go to colab.research.google.com
- Activate the GPU supported runtime
- Pro version: https://colab.research.google.com/signup




Linear Algebra

Scalars

- **Simple operations**

$$c = a + b$$

$$c = a \cdot b$$

$$c = \sin a$$

- **Length**

$$|a| = \begin{cases} a & \text{if} a > 0 \\ -a & \text{otherwise} \end{cases}$$

$$|a + b| \leq |a| + |b|$$

$$|a \cdot b| = |a| \cdot |b|$$

Vectors

- **Simple operations**

$$c = a + b \quad \text{where } c_i = a_i + b_i$$
$$c = \alpha \cdot b \quad \text{where } c_i = \alpha b_i$$
$$c = \sin a \quad \text{where } c_i = \sin a_i$$

Definition of a vector space

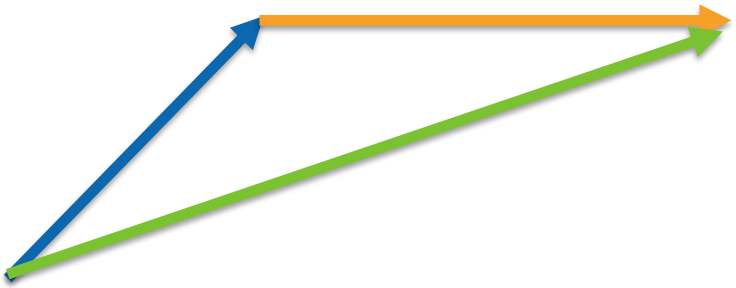
- **Length**

$$\|a\|_2 = \left[\sum_{i=1}^{m} a_i^2 \right]^{\frac{1}{2}}$$

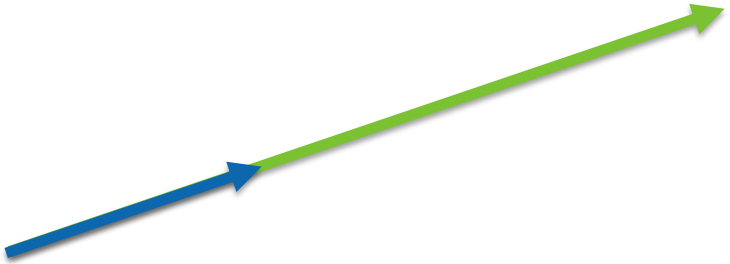
Definition of a norm

$$\|a\| \geq 0 \text{ for all } a$$
$$\|a + b\| \leq \|a\| + \|b\|$$
$$\|a \cdot b\| = |a| \cdot \|b\|$$

Vectors



$$c = a + b$$



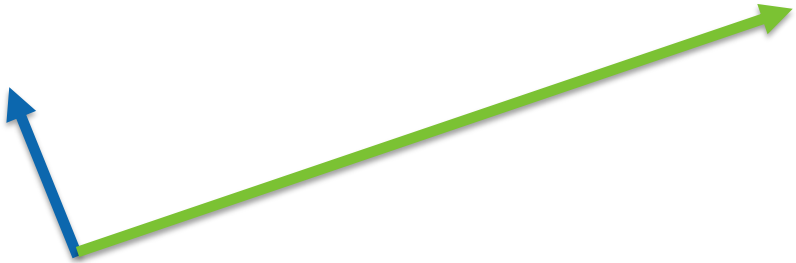
$$c = \alpha \cdot b$$

Mathematician's **'parallel for all do'**

Vectors

- **Dot product**

$$a^\top b = \sum_i a_i b_i$$

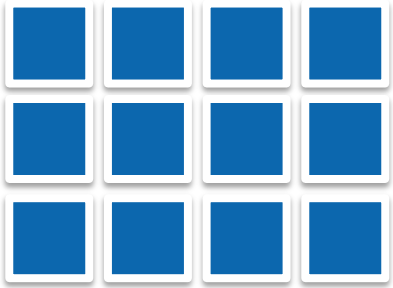


- **Orthogonality**

$$a^\top b = \sum_i a_i b_i = 0$$

(e.g. if we have two vectors that are orthogonal with a third, their linear combination is it, too)

Matrices



- **Simple operations**

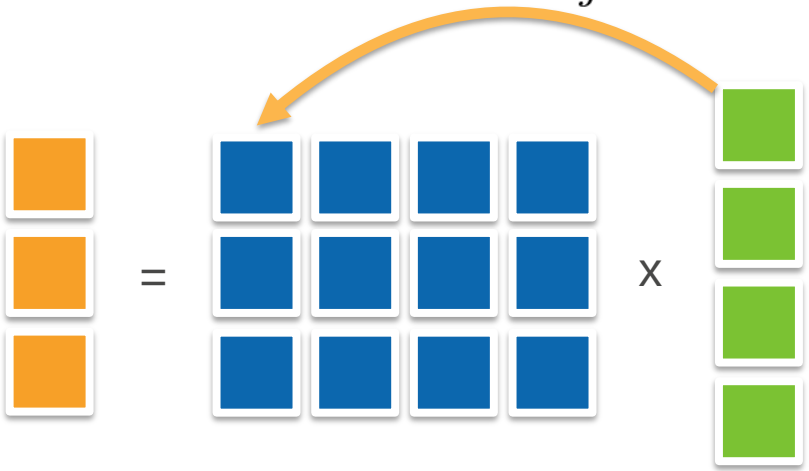
$$C = A + B \qquad \text{where } C_{ij} = A_{ij} + B_{ij}$$
$$C = \alpha \cdot B \qquad \text{where } C_{ij} = \alpha B_{ij}$$
$$C = \sin A \qquad \text{where } C_{ij} = \sin A_{ij}$$

- **Functional Analysis 101**
  vector = function, matrix = linear operator
  most theorems work sort-of in infinite dimensional spaces

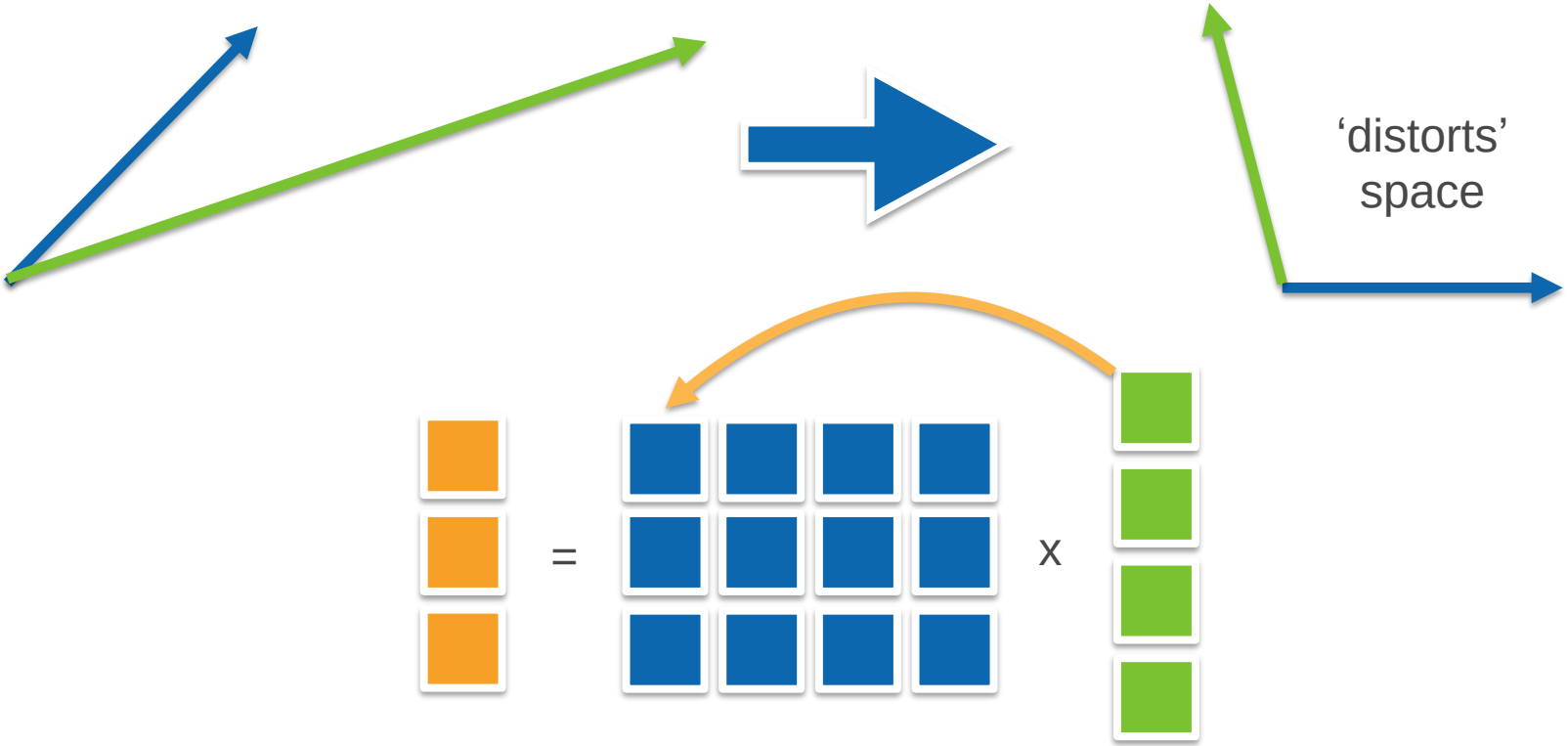
Matrices

- **Multiplications (matrix vector)**

$$c = Ab \text{ where } c_i = \sum_j A_{ij} b_j$$



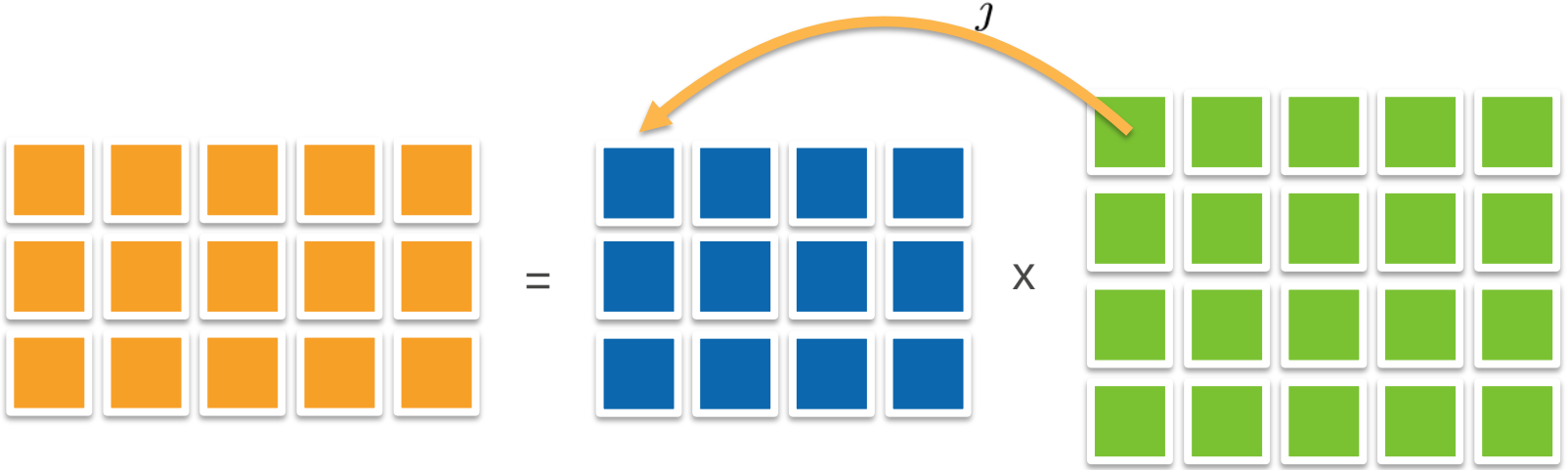
Matrices



Matrices

- **Multiplications (matrix matrix)**

$$C = AB \text{ where } C_{ik} = \sum_j A_{ij} B_{jk}$$



Matrices

- **Norms**

$$c = A \cdot b \text{ hence } \|c\| \leq \|A\| \cdot \|b\|$$

  - Choices depending on how to measure length of b and c
- **Popular norms**
  - Frobenius norm $\|A\|_{\text{Frob}} = \left[\sum_{ij} A_{ij}^2 \right]^{\frac{1}{2}}$
  - The $L_{2,1}$ is the sum of the Euclidean norms of the columns of the matrix

$$\|A\|_{2,1} = \sum_{j=1}^{n} \|a_j\|_2 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}|^2 \right)^{\frac{1}{2}}$$

Special Matrices

- **Symmetric, antisymmetric** $A_{ij} = A_{ji}$ and $A_{ij} = -A_{ji}$

- **Positive definite**

$$\|x\|^2 = x^\top x \geq 0 \text{ generalizes to } x^\top A x \geq 0$$

(all positive eigenvalues)

Special Matrices

- **Orthogonal Matrices**
  - All rows of the matrix are orthogonal to each other
  - All rows of the matrix have unit length

$$U \text{ with } \sum_j U_{ij} U_{kj} = \delta_{ik}$$

  - Rewrite in matrix form

$$UU^\top = \mathbf{1}$$

- **Permutation Matrices**

$$P \text{ where } P_{ij} = 1 \text{ if and only if } j = \pi(i)$$


GPUs love matrices and vectors
(they have many processors)


ndarray

N-dimensional Array Examples

N-dimensional array, short for ndarray, is the main data structure for machine learning and neural networks

0-d (scalar)

1.0

A class label

1-d (vector)

[1.0, 2.7, 3.4]

A feature vector

2-d (matrix)

[[1.0, 2.7, 3.4]
[5.0, 0.2, 4.6]
[4.3, 8.5, 0.2]]

A example-by-feature matrix

ND Array Examples, cont

3-d

```
[[[0.1, 2.7, 3.4]
  [5.0, 0.2, 4.6]
  [4.3, 8.5, 0.2]]
 [[3.2, 5.7, 3.4]
  [5.4, 6.2, 3.2]
  [4.1, 3.5, 6.2]]]
```

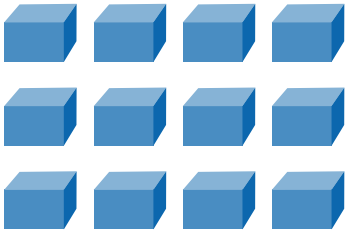
A RGB image (width x height x channels)

4-d

```
[[[[. . .
   . . .
   . . .]]]
```

A batch of RGB images (batch-size x width x height x channels)

5-d



```
[[[[. . .
   . . .
   . . .]]]
```

A batch of videos (batch-size x time x width x height x channels)

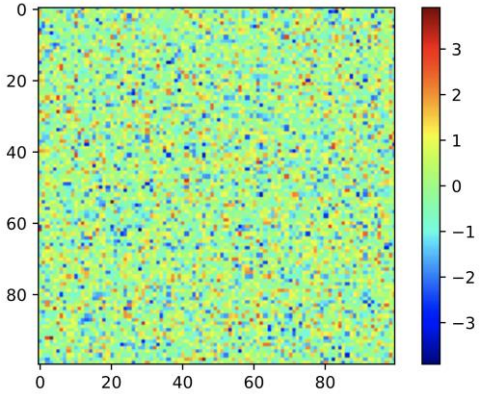
Create Arrays

Create arrays with

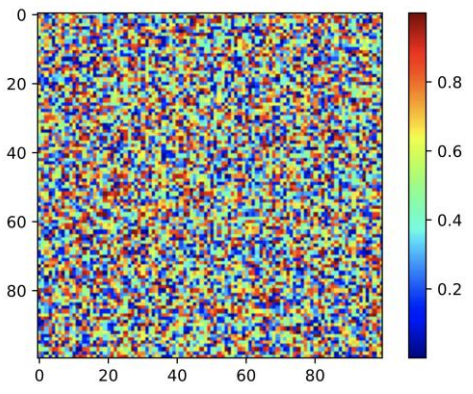
- A shape, e.g. 3-by-4 matrix
- Data type for each element, e.g. float
- Element values, e.g all 0s, or random values

100-by-100 matrix with elements generated from

A normal distribution

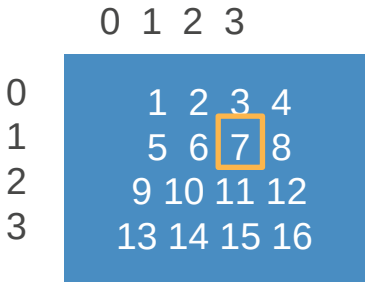


A uniform distribution

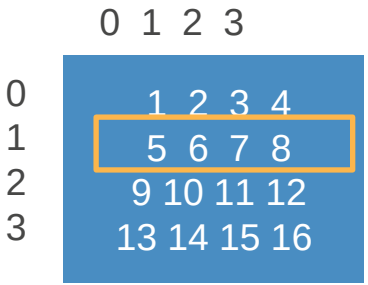


Access Elements

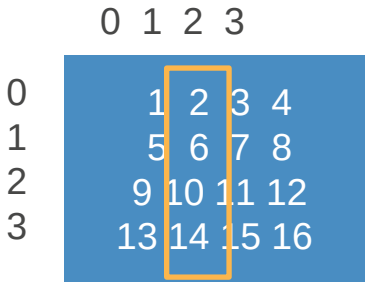
An element: [1, 2]



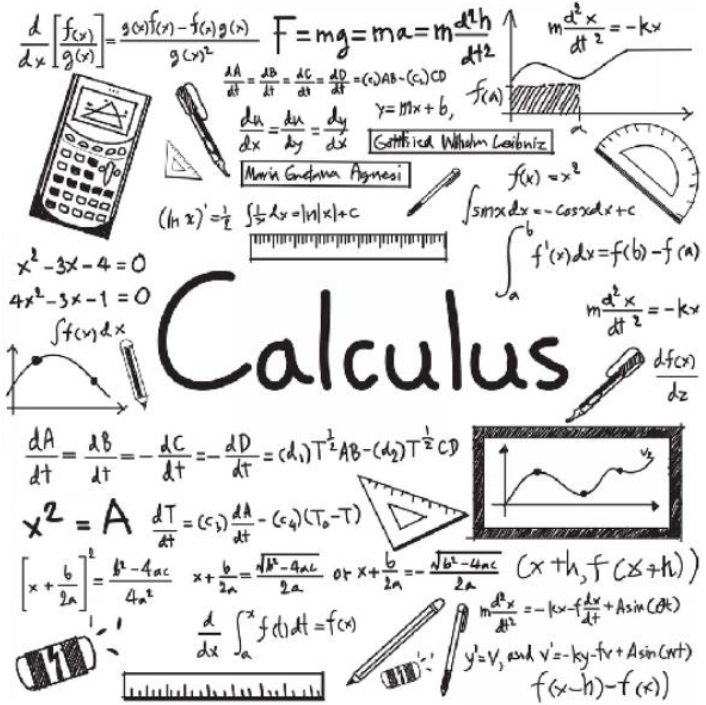
A row: [1, :]



A column: [:, 1]



Matrix



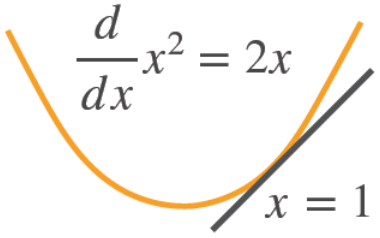
Review Scalar Derivative

y	a	x^n	$\exp(x)$	$\log(x)$	$\sin(x)$
$\frac{dy}{dx}$	0	nx^{n-1}	$\exp(x)$	$\frac{1}{x}$	$\cos(x)$

a is not a function of x

y	$u+v$	uv	$y=f(u),\ u=g(x)$
$\frac{dy}{dx}$	$\frac{du}{dx}+\frac{dv}{dx}$	$\frac{du}{dx}v+\frac{dv}{dx}u$	$\frac{dy}{du}\frac{du}{dx}$

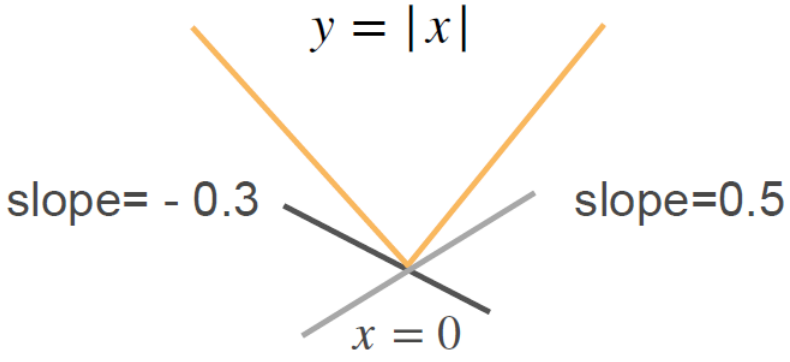
Derivative is the slope of the tangent line



The slope of the tangent line is 2

Subderivative

- Extend derivative to non-differentiable cases



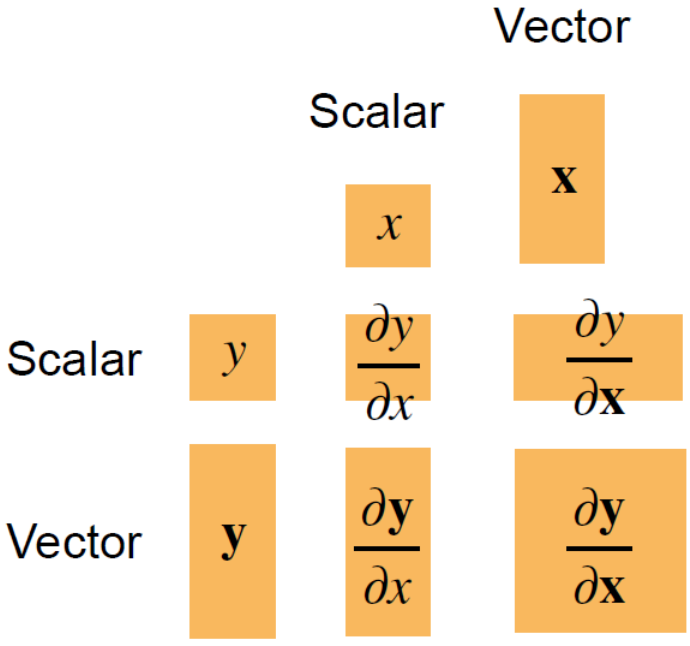
$$\frac{\partial |x|}{\partial x} = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ a & \text{if } x = 0, \quad a \in [-1,1] \end{cases}$$

Another example:

$$\frac{\partial}{\partial x} \max(x,0) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \\ a & \text{if } x = 0, \quad a \in [0,1] \end{cases}$$

Gradients

- Generalize derivatives into vectors



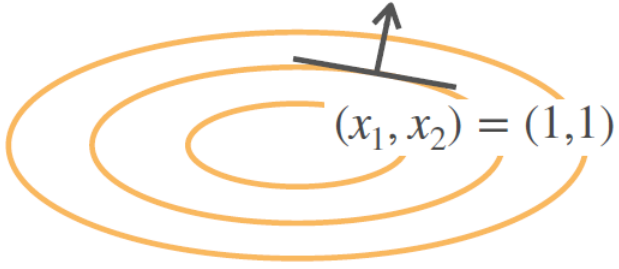
$\partial y / \partial \mathbf{x}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \frac{\partial y}{\partial \mathbf{x}} = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \cdots, \frac{\partial y}{\partial x_n} \right]$$

	x	$\mathbf{x}$
y	$\frac{\partial y}{\partial x}$	$\frac{\partial y}{\partial \mathbf{x}}$
$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

$$\frac{\partial}{\partial \mathbf{x}} x_1^2 + 2x_2^2 = [2x_1, 4x_2]$$

Direction (2, 4), perpendicular to the contour lines



Examples

y	a	au	$\text{sum}(\mathbf{x})$	$\|\mathbf{x}\|^2$
$\frac{\partial y}{\partial \mathbf{x}}$	$\mathbf{0}^T$	$a\frac{\partial u}{\partial \mathbf{x}}$	$\mathbf{1}^T$	$2\mathbf{x}^T$

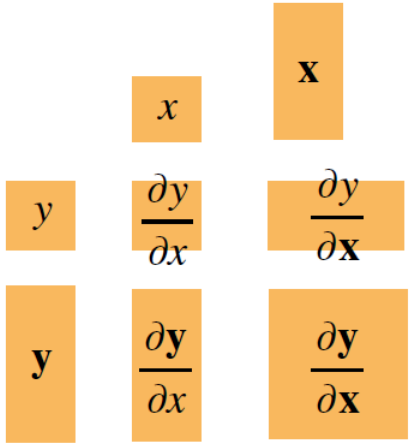
a is not a function of $\mathbf{x}$

$\mathbf{0}$ and $\mathbf{1}$ are vectors

y	$u + v$	uv	$\langle \mathbf{u}, \mathbf{v} \rangle$
$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial u}{\partial \mathbf{x}} + \frac{\partial v}{\partial \mathbf{x}}$	$\frac{\partial u}{\partial \mathbf{x}}v + \frac{\partial v}{\partial \mathbf{x}}u$	$\mathbf{u}^T\frac{\partial \mathbf{v}}{\partial \mathbf{x}} + \mathbf{v}^T\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$

$$\partial \mathbf{y} / \partial x$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \frac{\partial \mathbf{y}}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_m}{\partial x} \end{bmatrix}$$



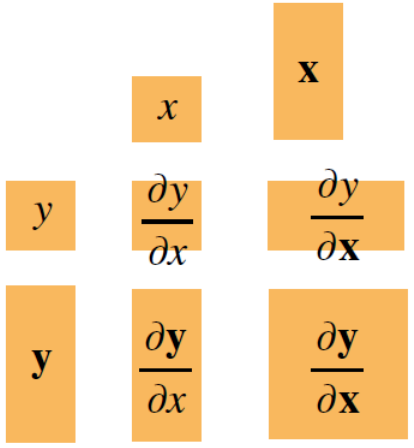
$\partial y / \partial \mathbf{x}$ is a row vector, while $\partial \mathbf{y} / \partial x$ is a column vector

It is called numerator-layout notation. The reversed version is called denominator-layout notation

$$\partial \mathbf{y} / \partial \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial \mathbf{x}} \\ \frac{\partial y_2}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial y_m}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1}, \frac{\partial y_1}{\partial x_2}, \cdots, \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1}, \frac{\partial y_2}{\partial x_2}, \cdots, \frac{\partial y_2}{\partial x_n} \\ \vdots \\ \frac{\partial y_m}{\partial x_1}, \frac{\partial y_m}{\partial x_2}, \cdots, \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$



Examples

$\mathbf{y}$	$\mathbf{a}$	$\mathbf{x}$	$\mathbf{Ax}$	$\mathbf{x}^T\mathbf{A}$
$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{A}$	$\mathbf{A}^T$

$\mathbf{y}$	$a\mathbf{u}$	$\mathbf{Au}$	$\mathbf{u} + \mathbf{v}$
$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$a\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\mathbf{A}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}$

$\mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{R}^m, \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathbb{R}^{m \times n}$

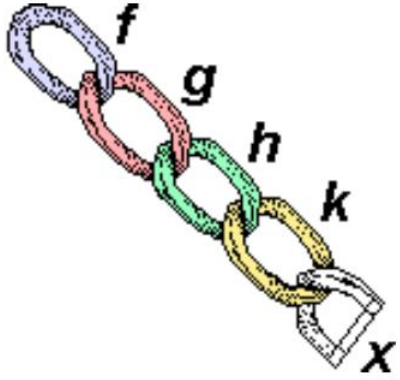
a, $\mathbf{a}$ and $\mathbf{A}$ are not functions of $\mathbf{x}$

$\mathbf{0}$ and $\mathbf{I}$ are matrices

Generalize to Matrices

	Scalar	Vector	Matrix
	x $(1,)$	$\mathbf{x}$ $(n,1)$	$\mathbf{X}$ (n,k)
Scalar	y $(1,)$	$\frac{\partial y}{\partial x}$ $(1,)$	$\frac{\partial y}{\partial \mathbf{x}}$ $(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}$ (k,n)
Vector	$\mathbf{y}$ $(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}$ $(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ (m,n)	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$ (m,k,n)
Matrix	$\mathbf{Y}$ (m,l)	$\frac{\partial \mathbf{Y}}{\partial x}$ (m,l)	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$ (m,l,n)	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$ (m,l,k,n)

Chain Rule



Generalize to Vectors

- Chain rule for scalars:

$$y = f(u),\ u = g(x) \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

- Generalize to vectors straightforwardly

$$\frac{\partial y}{\partial \mathbf{x}} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial \mathbf{x}} \qquad \frac{\partial y}{\partial \mathbf{x}} = \frac{\partial y}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \qquad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

$$(1,n)\quad (1,)\ (1,n) \qquad (1,n)\quad (1,k)\ \ (k,n) \qquad (m,n)\ \ (m,k)\ (k,n)$$

Example 1

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

Assume $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n, \quad y \in \mathbb{R}$

$$z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$$

Compute $\frac{\partial z}{\partial \mathbf{w}}$

Decompose

$$a = \langle \mathbf{x}, \mathbf{w} \rangle$$
$$b = a - y$$
$$z = b^2$$

$$\begin{aligned}
\frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial \mathbf{w}} \\
&= \frac{\partial b^2}{\partial b} \frac{\partial a - y}{\partial a} \frac{\partial \langle \mathbf{x}, \mathbf{w} \rangle}{\partial \mathbf{w}} \\
&= 2b \cdot 1 \cdot \mathbf{x}^T \\
&= 2\left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right) \mathbf{x}^T
\end{aligned}$$

Example 2

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

Assume $\mathbf{X} \in \mathbb{R}^{m \times n}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{R}^m$

$$z = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

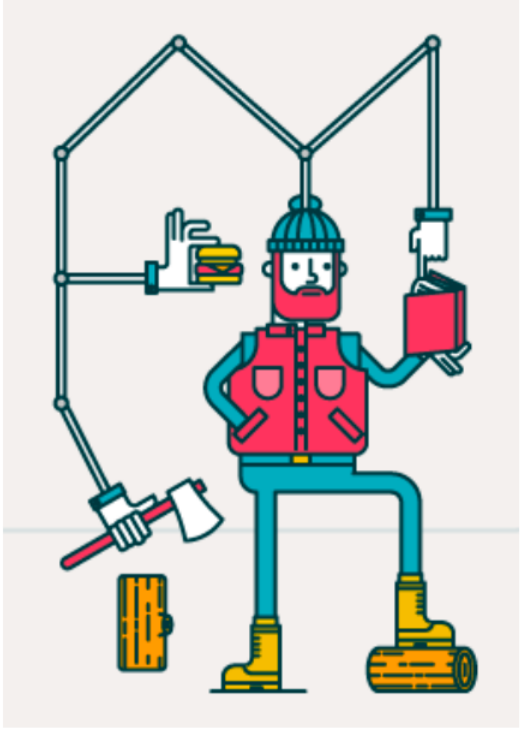
Compute $\frac{\partial z}{\partial \mathbf{w}}$

$$\begin{aligned}
\frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{w}} \\
&= \frac{\partial \|\mathbf{b}\|^2}{\partial \mathbf{b}} \frac{\partial \mathbf{a} - \mathbf{y}}{\partial \mathbf{a}} \frac{\partial \mathbf{X}\mathbf{w}}{\partial \mathbf{w}} \\
&= 2\mathbf{b}^T \times \mathbf{I} \times \mathbf{X} \\
&= 2\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^T \mathbf{X}
\end{aligned}$$

Decompose

$$\begin{aligned}
\mathbf{a} &= \mathbf{X}\mathbf{w} \\
\mathbf{b} &= \mathbf{a} - \mathbf{y} \\
z &= \|\mathbf{b}\|^2
\end{aligned}$$

Auto Differentiation



Auto Differentiation (AD)

- AD evaluates gradients of a function specified by a program at given values
- AD differs to
  - Symbolic differentiation

```
In[1]:= D[4 x^3 + x^2 + 3, x]

Out[1]= 2 x + 12 x^2
```

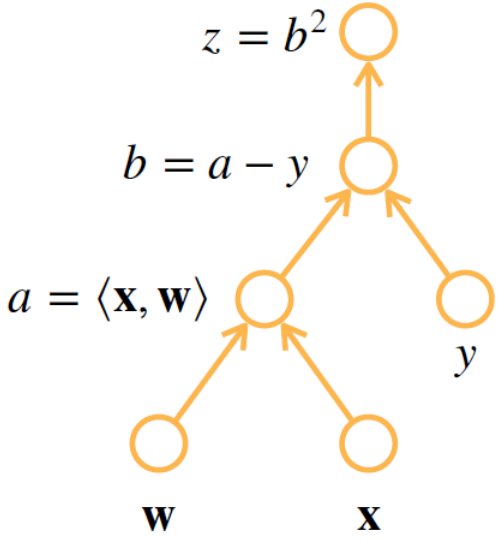
  - Numerical differentiation

$$\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Computation Graph

- Decompose into primitive operations
- Build a directed acyclic graph to present the computation

Assume $z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$



Computation Graph

- Decompose into primitive operations
- Build a directed acyclic graph to present the computation
- Build explicitly
  - Tensorflow/Theano/MXNet
- Build implicitly though tracing
  - PyTorch/MXNet

```
from mxnet import autograd, nd

with autograd.record():
    a = nd.ones((2,1))
    b = nd.ones((2,1)
    c = 2 * a + b
```

Two Modes

- By chain rule $$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\cdots\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}$$

- Forward accumulation

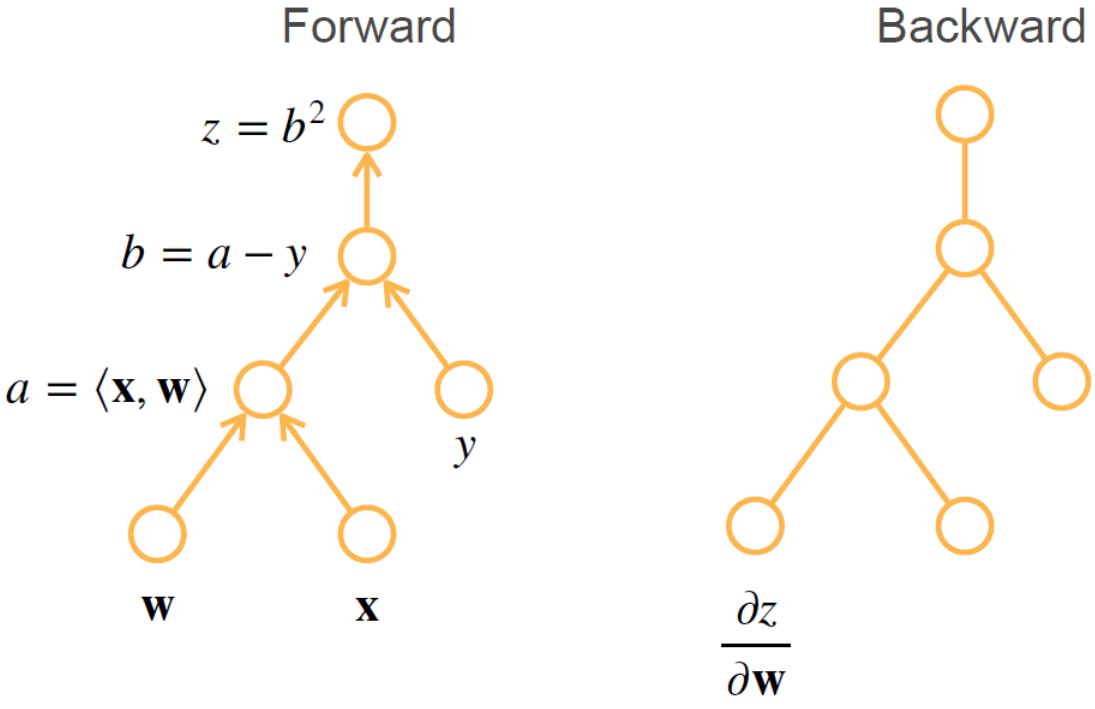
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u_n}\left(\frac{\partial u_n}{\partial u_{n-1}}\left(\cdots\left(\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}\right)\right)\right)$$

- Reverse accumulation (a.k.a Backpropagation)

$$\frac{\partial y}{\partial x} = \left(\left(\left(\frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\right)\cdots\right)\frac{\partial u_2}{\partial u_1}\right)\frac{\partial u_1}{\partial x}$$

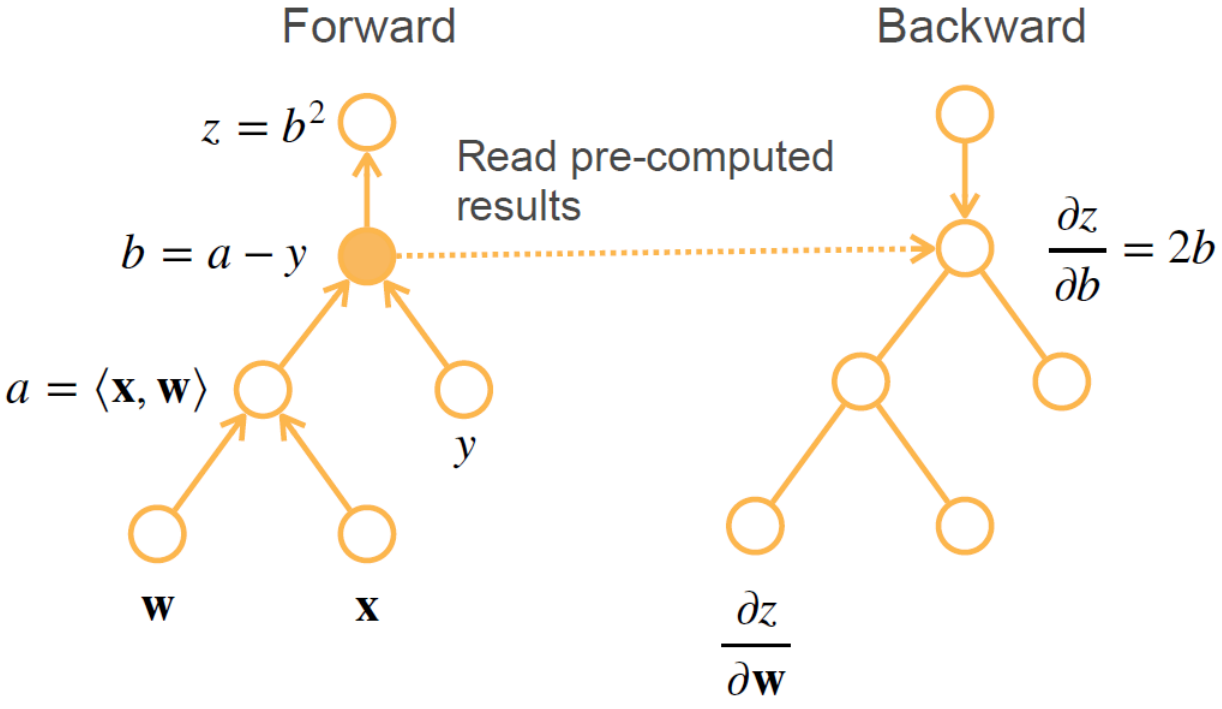
Reverse Accumulation

Assume $z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$



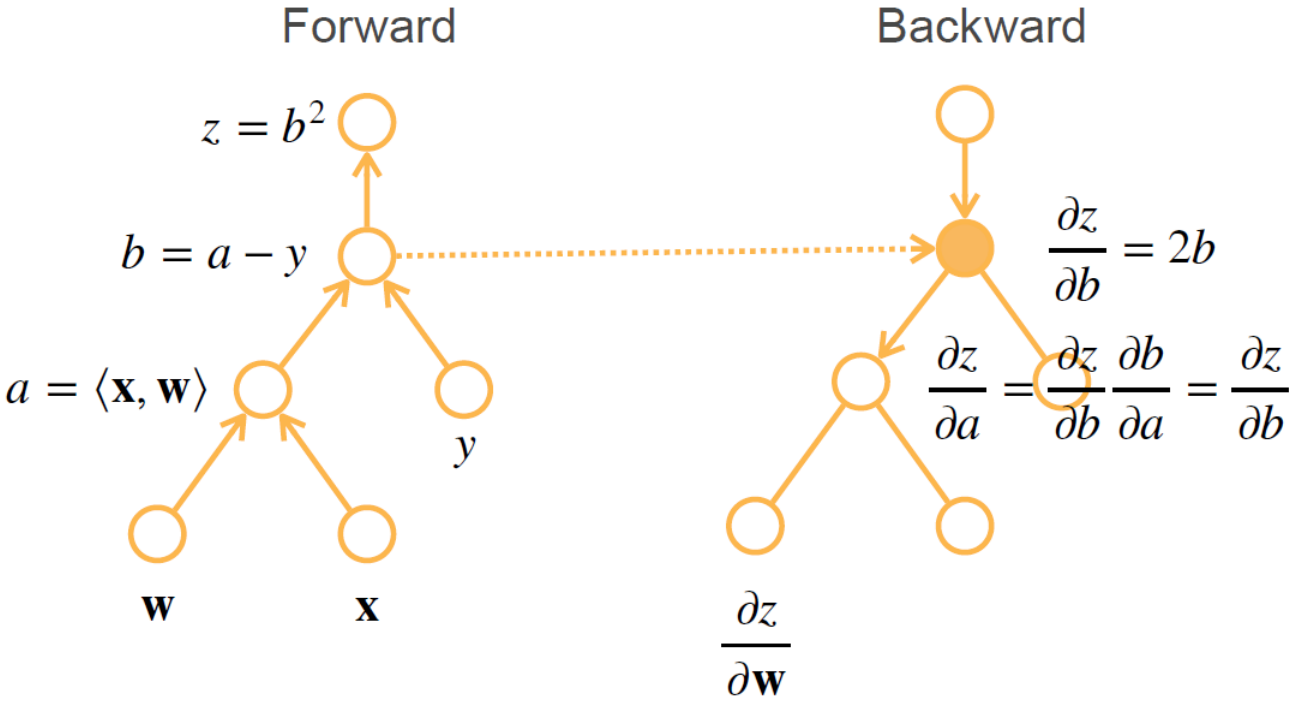
Reverse Accumulation

Assume $z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$



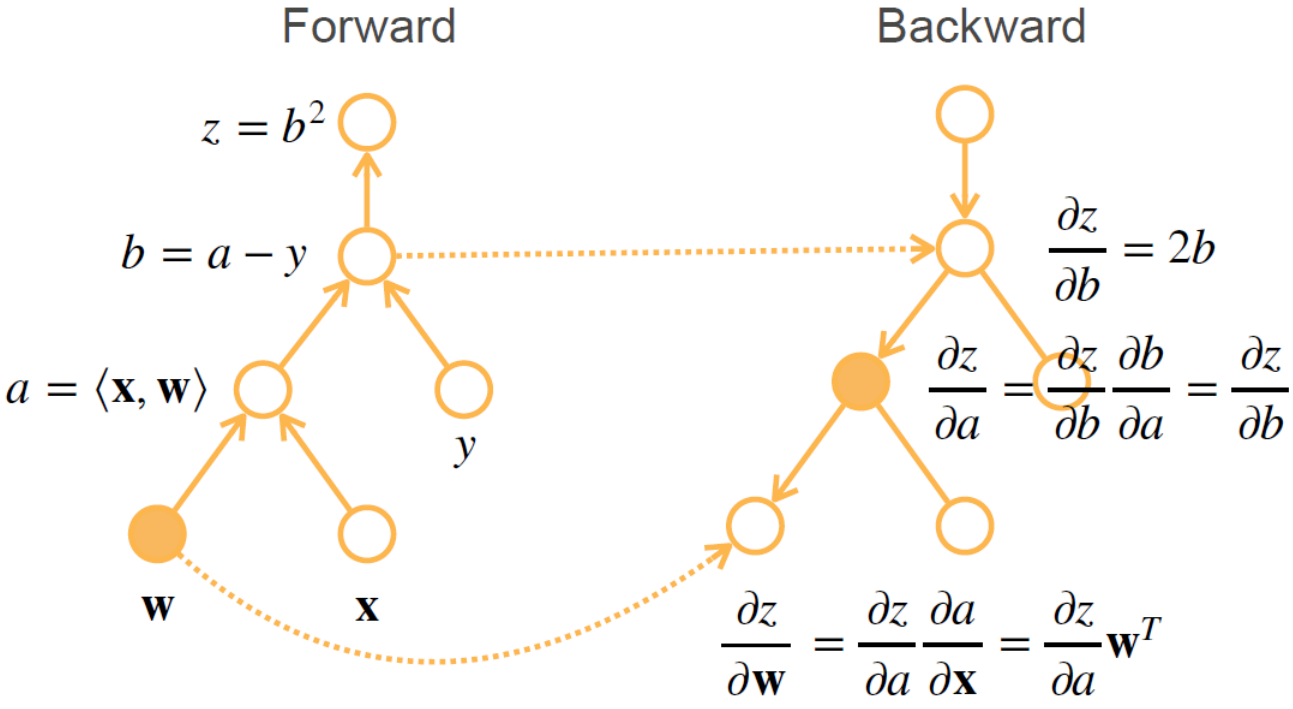
Reverse Accumulation

Assume $z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$



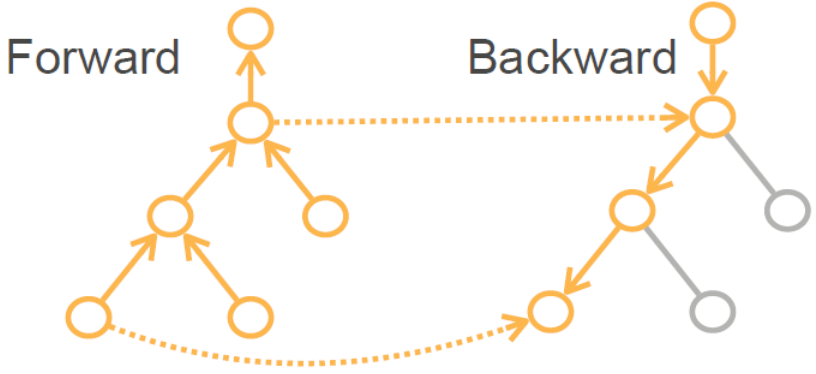
Reverse Accumulation

Assume $z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$

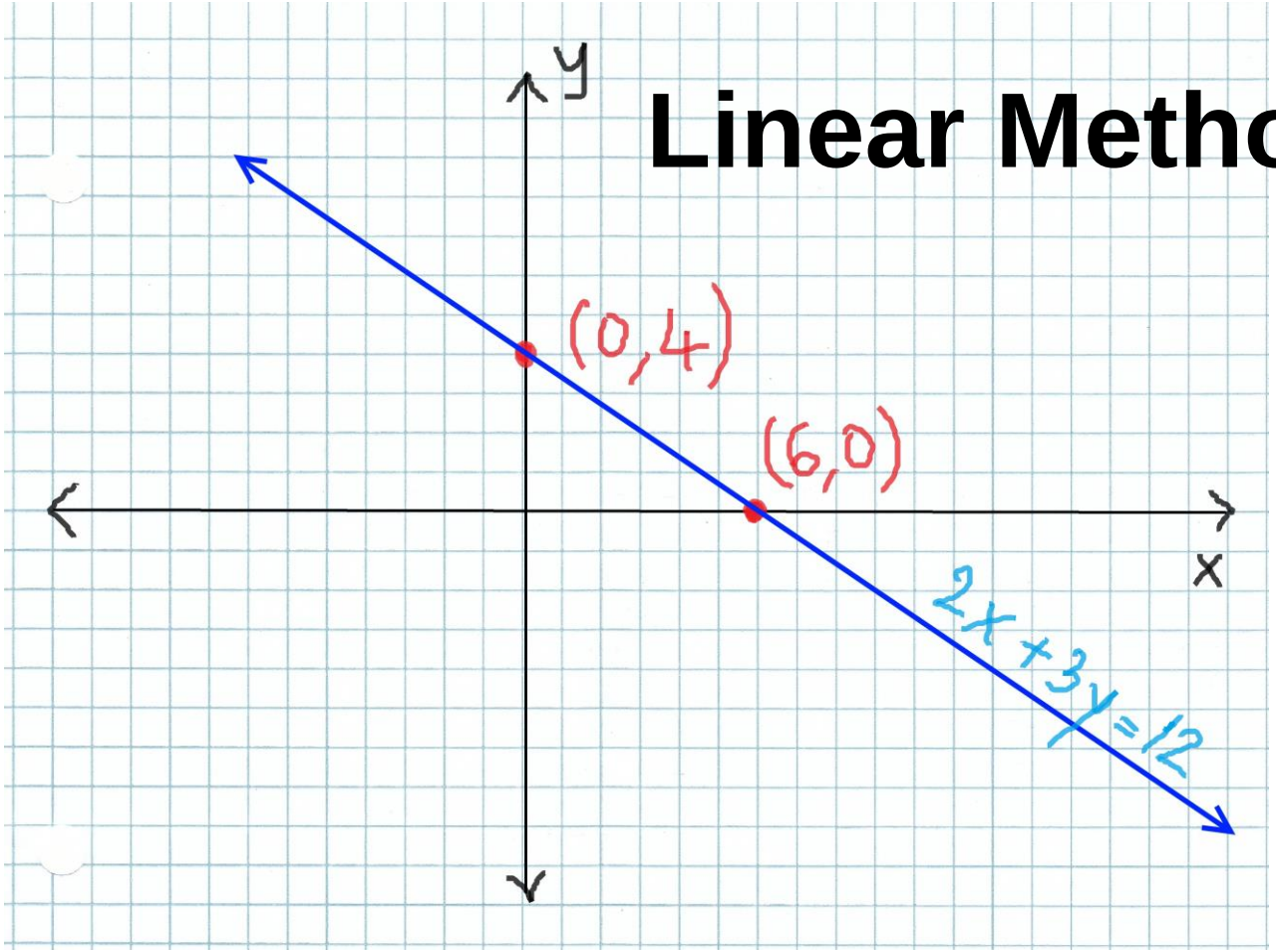


Reverse Accumulation Summary

- Build a computation graph
- Forward: Evaluate the graph, store intermediate results
- Backward: Evaluate the graph in a reversed order
  - Eliminate paths not needed



Linear Methods



House Buying 101

Pick a house, take a tour, and read facts

Estimate its price, bid



Listing price from agent

$5,498,000	7	5	4,865 Sq. Ft.
Price	Beds	Baths	$1130 / Sq. Ft.

Redfin Estimate: $5,390,037 On Redfin: 15 days

Predicted sale price

Virtual Tour
- Branded Virtual Tour
- Virtual Tour (External Link)

Parking Information
- Garage (Minimum): 2
- Garage (Maximum): 2
- Parking Description: Attached Garage, On Street
- Garage Spaces: 2

Multi-Unit Information
- # of Stories: 2

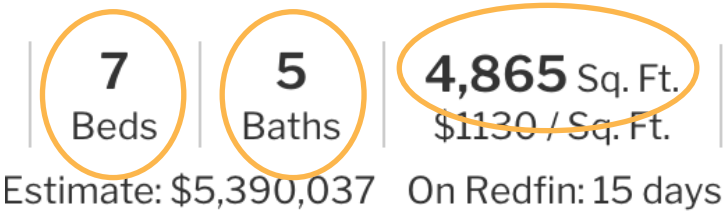
School Information
- Elementary School: El Carmelo Elementa
- Elementary School District: Palo Alto Uni
- Middle School: Jane Lathrop Stanford Mi
- High School: Palo Alto High
- High School District: Palo Alto Unified

Interior Features

Bedroom Information
- # of Bedrooms (Minimum): 7
- # of Bedrooms (Maximum): 7

- Kitchen Description: Countertop (Granite Dishwasher, Garbage Disposal, Hood Ove Island with Sink, Microwave, Oven Range

A Simplified Model



Assumption 1

The key factors impacting the prices are #Beds, #Baths, Living Sqft, denoted by x_1, x_2, x_3

Assumption 2

The sale price is a weighted sum over the key factors

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Weights and bias are determined later

Linear Model

Given *n*-dimensional inputs $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$

Linear model has a *n*-dimensional weight and a bias

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, b$$

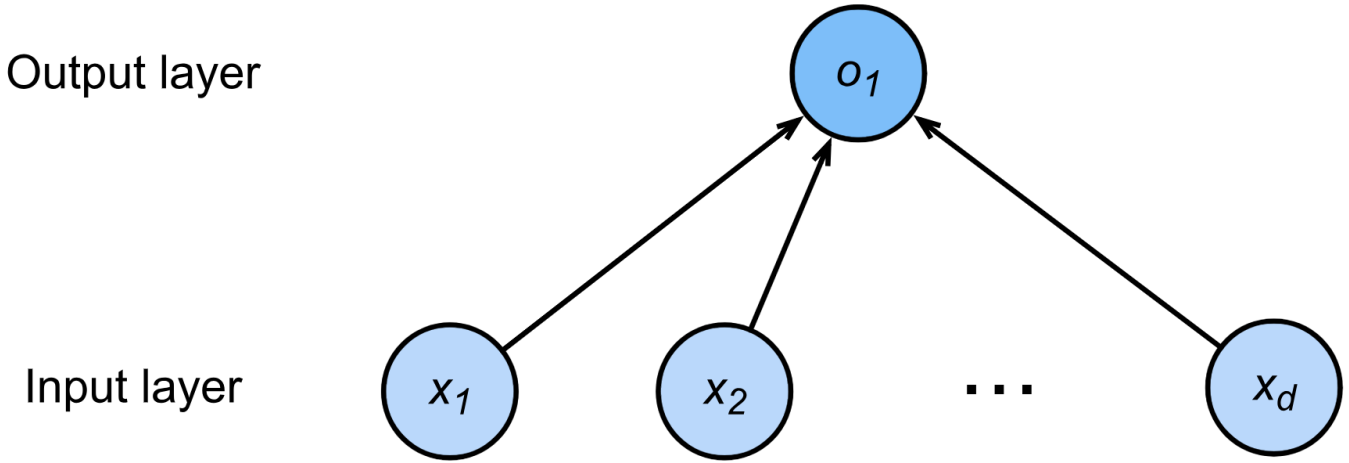
The output is a weighted sum of the inputs

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Vectorized version

$$y = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

Linear Model as a Single-layer Neural Network



We can stack multiple layers to get deep neural networks

Measure Estimation Quality

Compare the true value vs the estimated value
Real sale price vs estimated house price

Let y the true value, and $\hat{y}$ the estimated value, we can compare the loss

$$\ell(y, \hat{y}) = \left(y - \hat{y}\right)^2$$

It is called squared loss

Training Data

Collect multiple data points to fit parameters
Houses sold in the last 6 months

It is called the training data

The more the better

Assume *n* examples

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n]^T \quad \mathbf{y} = [y_0, y_1, \ldots, y_n]^T$$

Learn Parameters

Training loss

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b)^2 = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\mathbf{w} - b \|^2$$

Minimize loss to learn parameters

$$\mathbf{w}^*, \mathbf{b}^* = \arg\min_{\mathbf{w}, b} \ \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b)$$

Closed-form Solution

Add bias into weights by $\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}] \mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\mathbf{w} \|^2 \qquad \frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{2}{n} (\mathbf{y} - \mathbf{X}\mathbf{w})^T \mathbf{X}$$

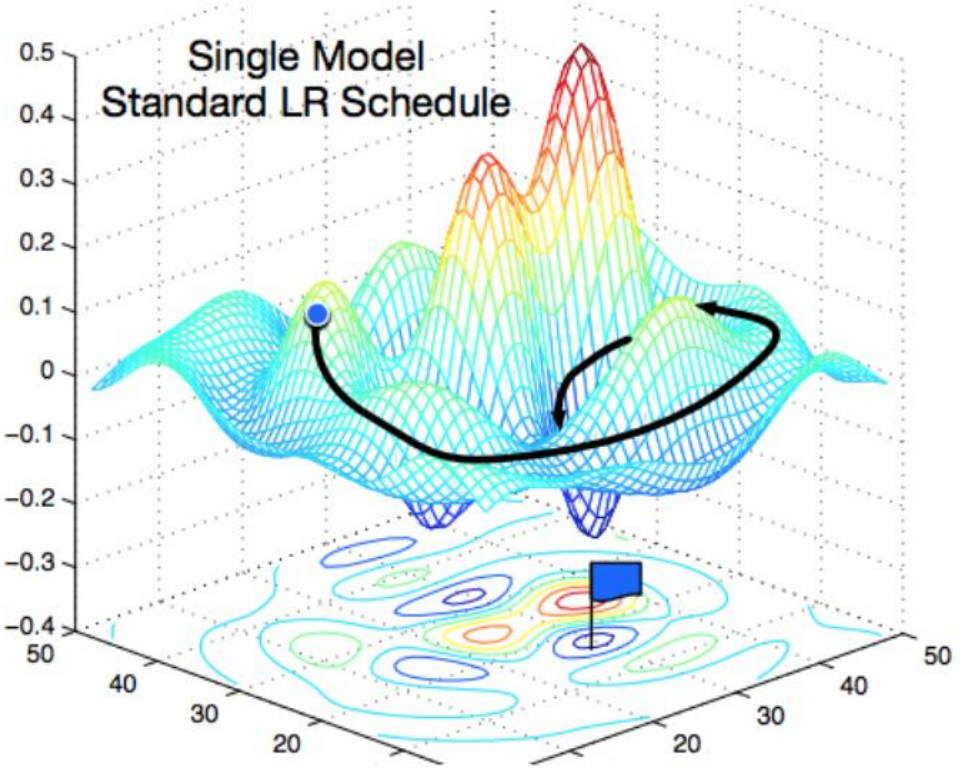
Loss is convex, so the optimal solutions satisfies

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = 0$$

$$\Leftrightarrow \frac{2}{n} (\mathbf{y} - \mathbf{X}\mathbf{w})^T \mathbf{X} = 0$$

$$\Leftrightarrow \mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}\mathbf{y}$$

Basic Optimization

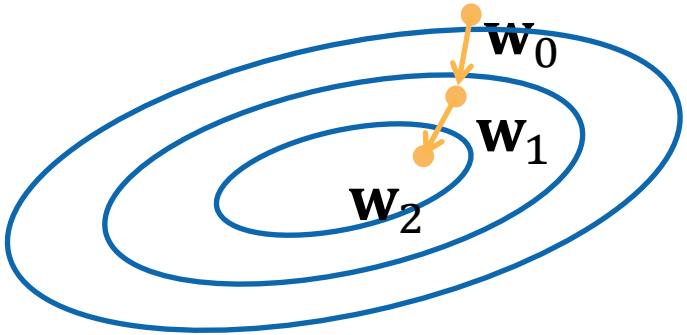


Gradient Descent

Choose a staring point $\mathbf{w}_0$

Repeat to update the weight *t=1,2,3*

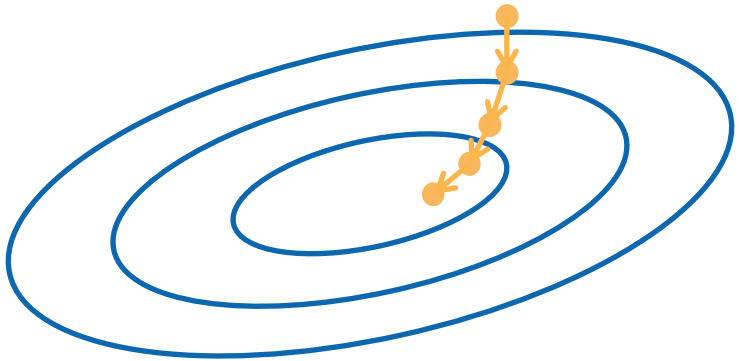
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial \ell}{\partial \mathbf{w}_{t-1}}$$



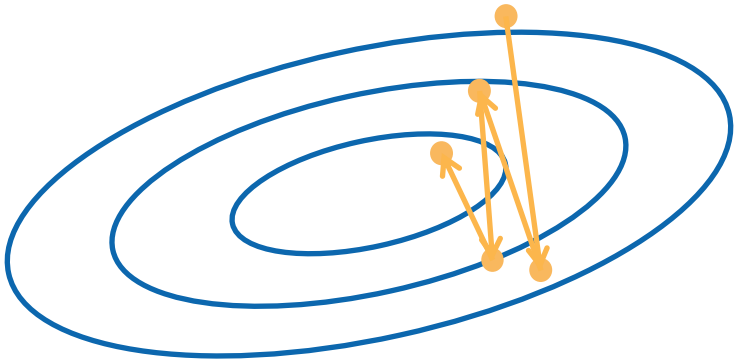
- Gradient: a direction that increases the value
- Learning rate: a hyper-parameter specifies the step length

Choose a Learning Rate

Not too small



Not too big



Mini-batch Stochastic Gradient Descent (SGD)

Computing the gradient over the whole training data is too expensive

- Takes minutes to hours for DNN models

Randomly sample *b* examples $i_1, i_2, \cdots, i_b$ to approximate the loss

$$\frac{1}{b} \sum_{i \in I_b} \ell(\mathbf{x}_i, y_i, \mathbf{w})$$

- *b* is the batch size, another important hyperparameters

Choose a Batch Size

Not too small

Workload is too small, hard to fully utilize computation resources

Not too big

Memory issue
Waste computation, e.g. when all x_i are identical

Summary

Problem: estimate a real value

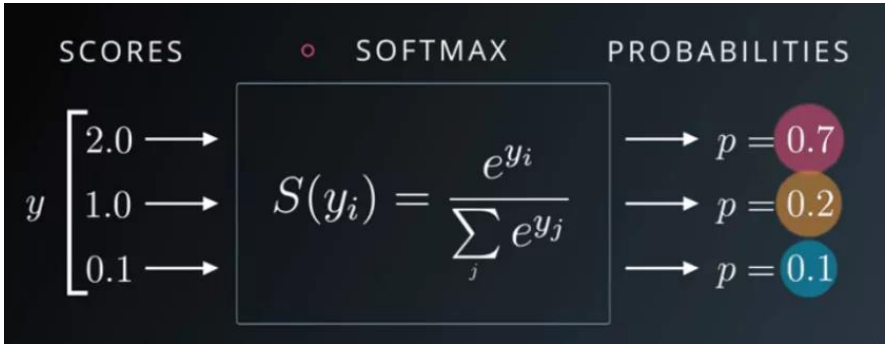
Model: $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$

Loss: square loss $\ell(y, \hat{y}) = \left(y - \hat{y}\right)^2$

Learn by mini-batch SGD

- Choose a starting point
- Repeat
  - Compute gradient
  - Update parameters

Softmax Regression



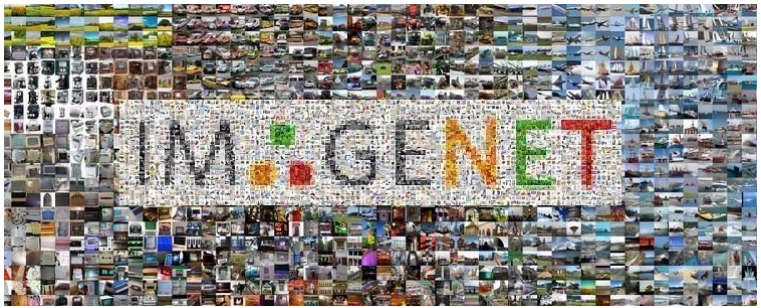
Regression vs Classification

Regression estimates a continuous value

Classification predicts a discrete category

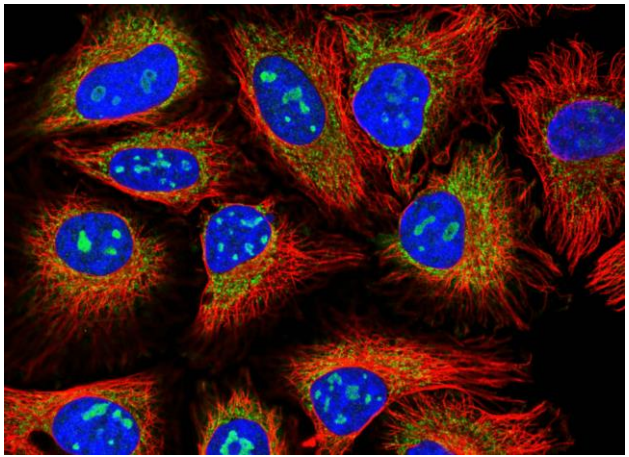
MNIST: classify hand-written digits (10 classes)

ImageNet: classify nature objects (1000 classes)



Various Classification Tasks at Kaggle

Classify human protein microscope images into 28 categories



0. Nucleoplasm
1. Nuclear membrane
2. Nucleoli
3. Nucleoli fibrillar
4. Nuclear speckles
5. Nuclear bodies
6. Endoplasmic reticu
7. Golgi apparatus
8. Peroxisomes
9. Endosomes
10. Lysosomes
11. Intermediate fila
12. Actin filaments
13. Focal adhesion si
14. Microtubules
15. Microtubule ends
16. Cytokinetic bridg

https://www.kaggle.com/c/human-protein-atlas-image-classification

Various Classification Tasks at Kaggle

Classify malware into 9 categories

https://www.kaggle.com/c/malware-classification

Various Classification Tasks at Kaggle

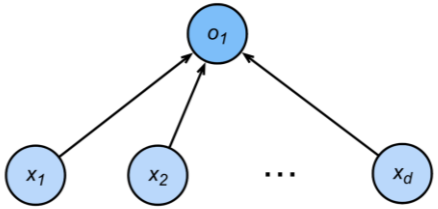
Classify toxic Wikipedia comments into 7 categories

comment_text	toxic	severe_toxic	obsc
Explanation\nWhy the edits made under my usern...	0	0	0
D'aww! He matches this background colour I'm s...	0	0	0
Hey man, I'm really not trying to edit war. It...	0	0	0
"\nMore\nI can't make any real suggestions on ...	0	0	0
You, sir, are my hero. Any chance you remember...	0	0	0

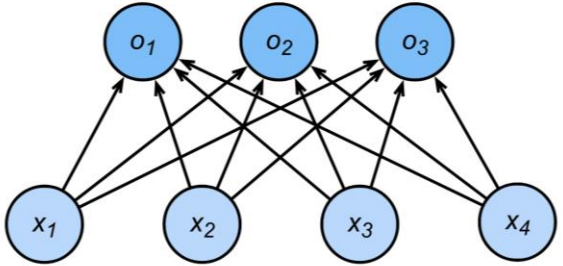
https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

From Regression to Multi-class Classification

Single continuous output



Output a confidence score for each category



- Predict the category

$$\arg\max_i (o_1, o_2, o_3)$$

- *max* is not differentiable
- define the loss function

Softmax

$$softmax([x_1, x_2, \dots, x_n]^T) = \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \dots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}}\right]$$

- Use exp to get positive numbers
- Divide by the sum to obtain a probability distribution

Example: given scores [1, -1, 2]

softmax: [0.26, 0.04, 0.7]

Softmax Gradient

$$softmax([x_1, x_2, \dots, x_n]^T) = \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \dots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}}\right]$$

$$\frac{\partial}{\partial \mathbf{x}} softmax(\mathbf{x})$$

https://aimatters.wordpress.com/2019/06/17/the-softmax-function-derivative/

Use Square Loss?

One-hot encoding of label *k*

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T, y_i = \begin{cases} 1 & if \;\; i = k \\ 0 & otherwise \end{cases}$$

We want the softmax output close to **y**

Use square loss for **y** = [0, 0, 1]

Softmax output	Loss
[0.3, 0, 0.7]	0.18
[0.17, 0.17, 0.66]	0.173

Cross Entropy Loss

Both one-hot encoding and softmax output are probability distribution

Cross entropy is commonly used to compare distributions

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

- If *i = k (ground truth)* , then

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\log(\hat{y}_i)$$

Linear Model vs Logistic Regression

Problem	single value regression	k-class classification
Model	$\langle \mathbf{w}, \mathbf{x} \rangle + b$ $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$	$softmax(\mathbf{W}\mathbf{x} + \mathbf{b})$ $\mathbf{W} \in \mathbb{R}^{k \times n}, \mathbf{b} \in \mathbb{R}^k$
Loss	Squared loss	Cross-entropy loss

The End