Deep Learning Basics

Rachel Hu and Zhi Zhang

Amazon AI

Outline

- Installations
- Deep Learning Motivations
- DeepNumpy & Calculus
- Regression
- Optimization
- Softmax Regression
- Multilayer Perceptron (train MNIST)


Installations

Installations

- **Python**
  - Everyone is using it in machine learning
- **Miniconda**
  - Package manager (for simplicity)
- **Jupyter Notebook**
  - So much easier to keep track of your experiments

Installations

Detailed step-by-step instructions on **local (Mac or Linux)**:

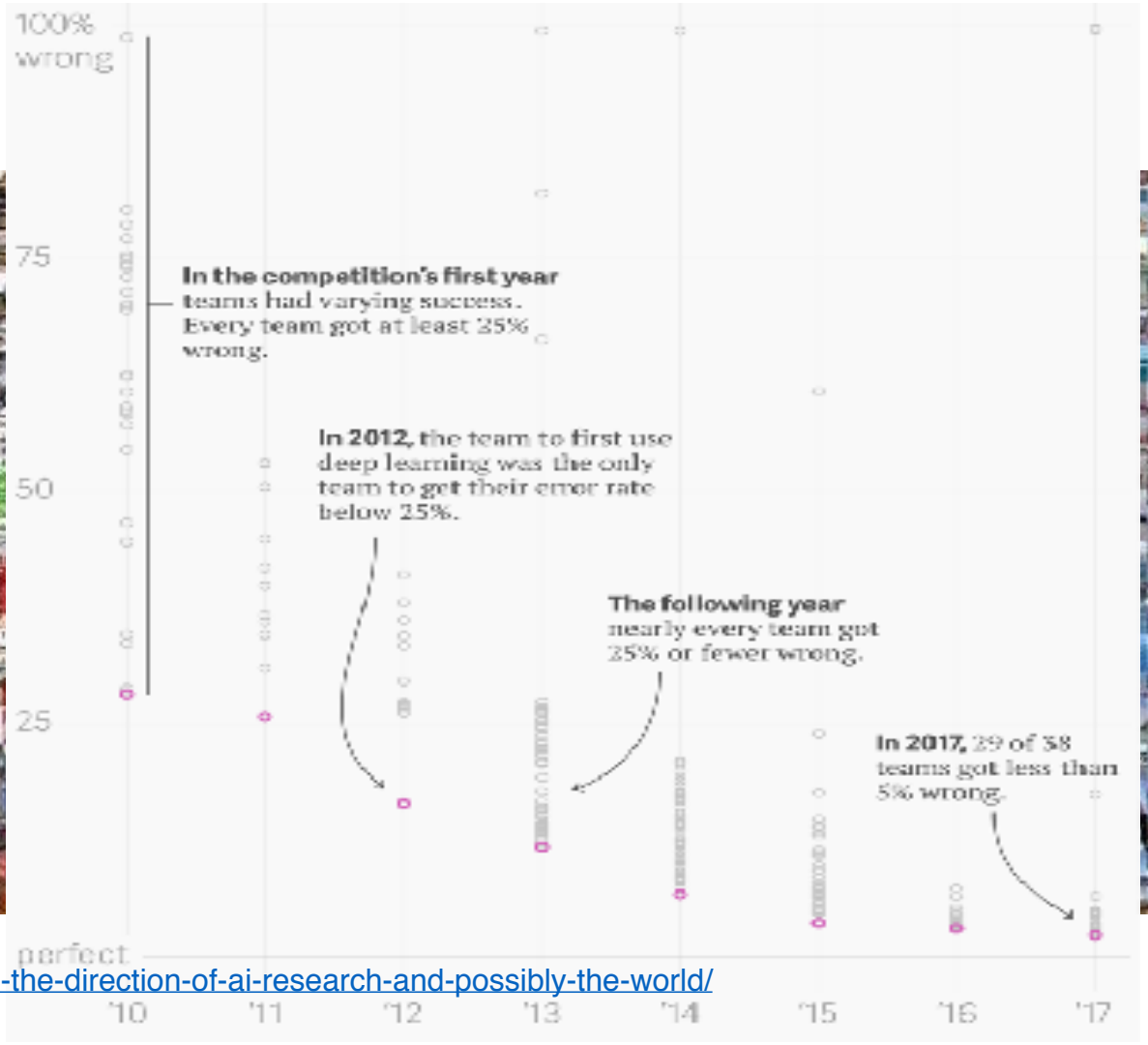
https://d2l.ai/chapter_install/install.html


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

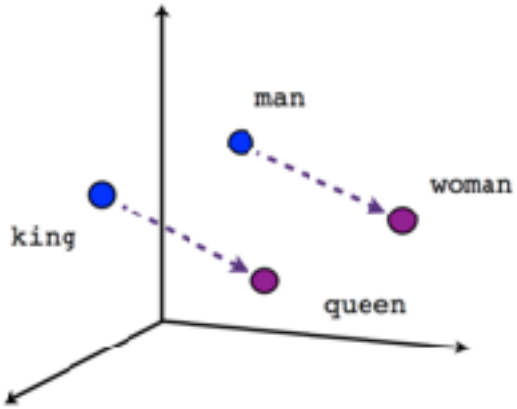
Style Transfer

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

Synthesize Faces

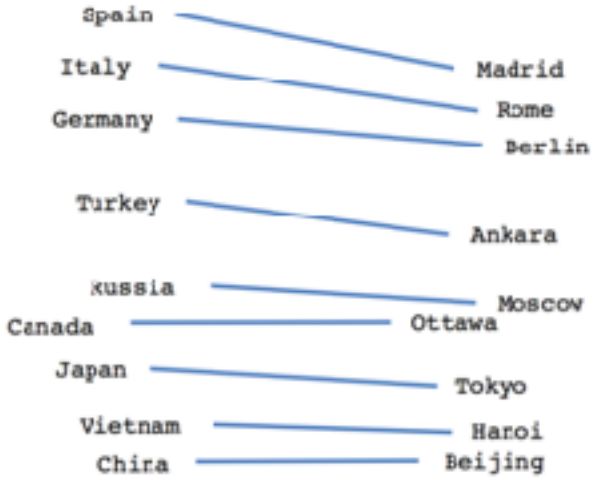
Karras et al, arXiv 2019

Analogies



Male-Female

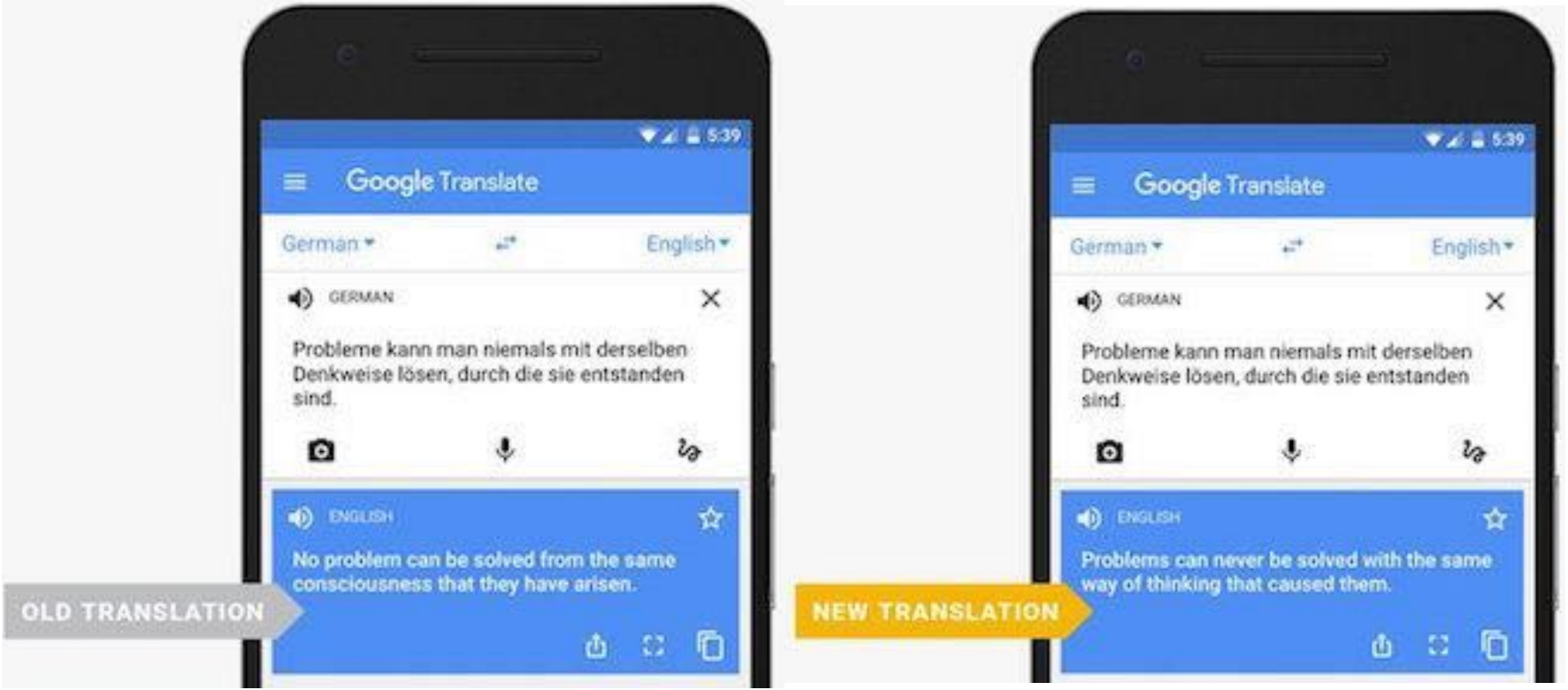
Verb tense



Country-Capital

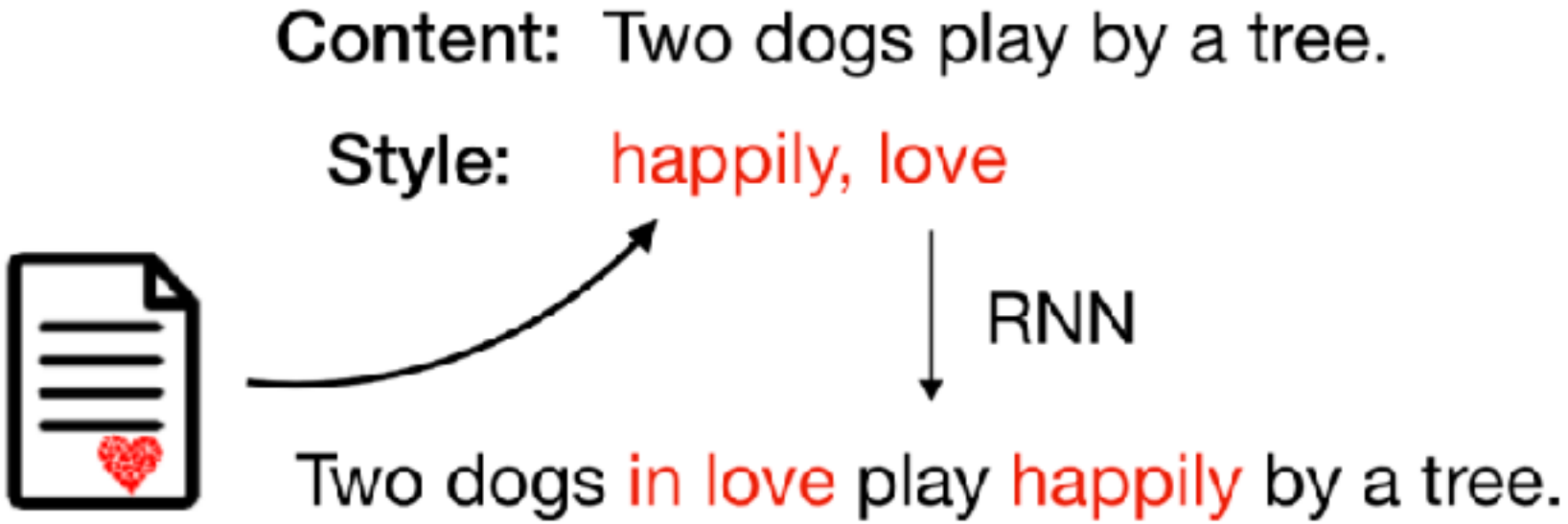
https://nlp.stanford.edu/projects/glove/

Machine Translation



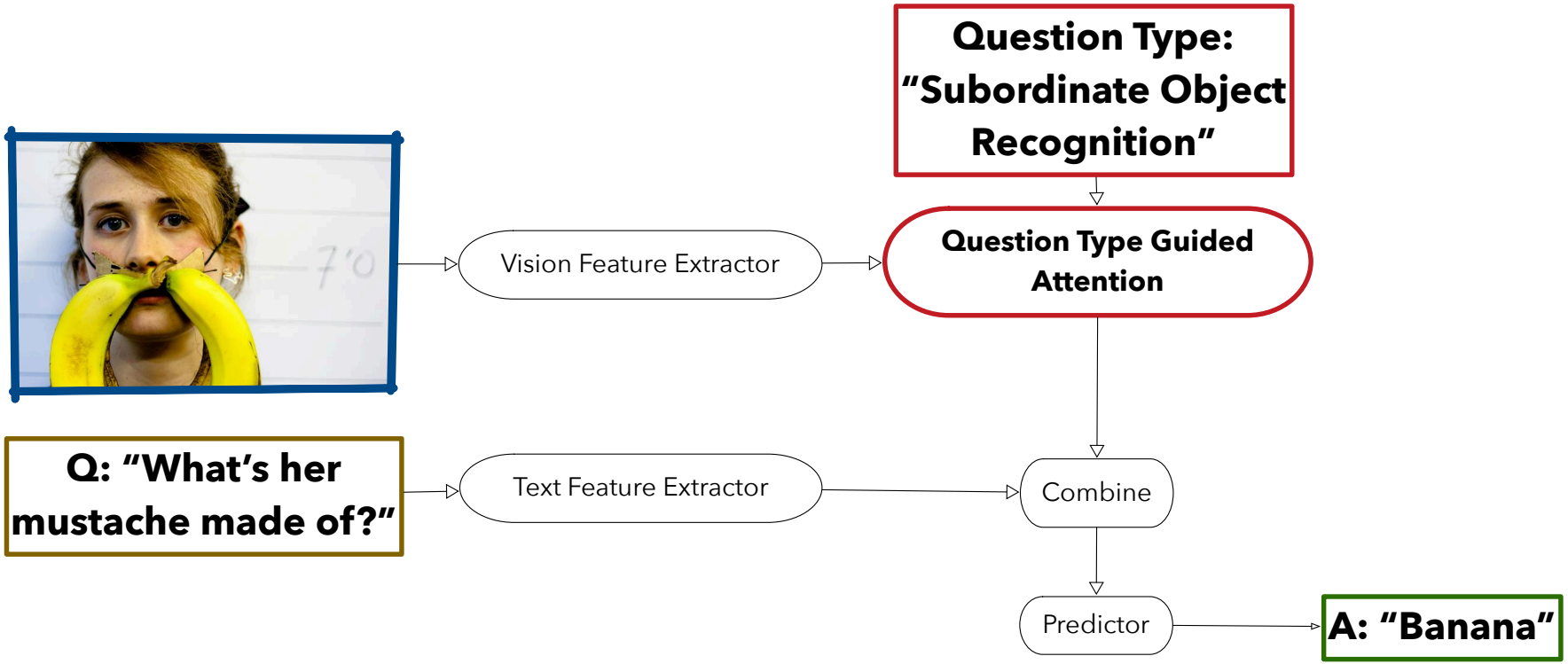
https://www.pcmag.com/news/349610/google-expands-neural-networks-for-language-translation

Text Synthesis



Li et al, NACCL, 2018

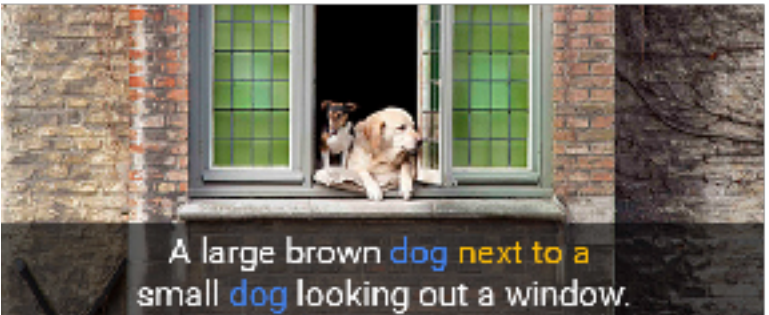
Question answering



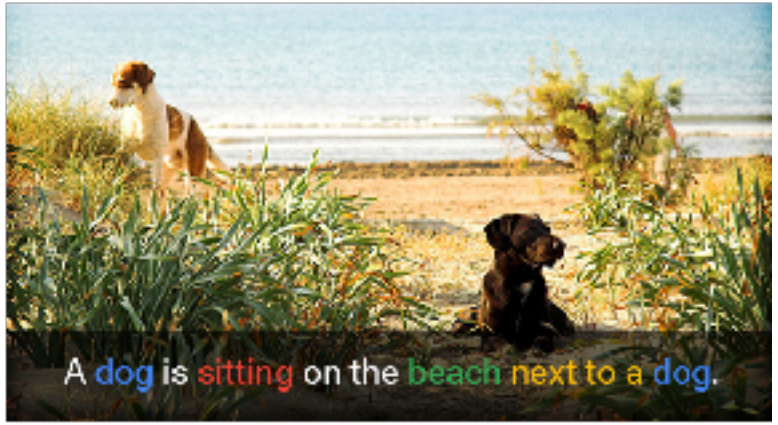
Shi et al, 2018, Arxiv

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

Problems we will solve: Classification

cat

dog

rabbit

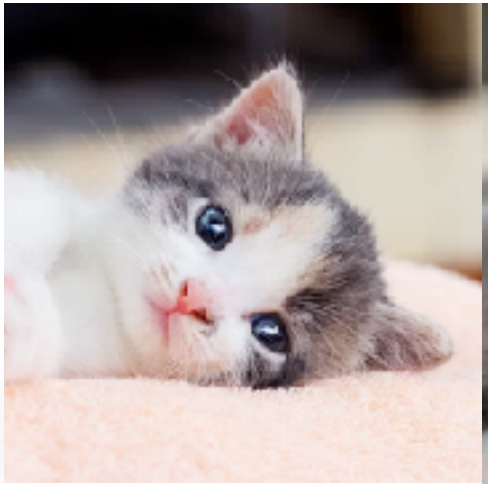
gerbil

Given image x estimate label y

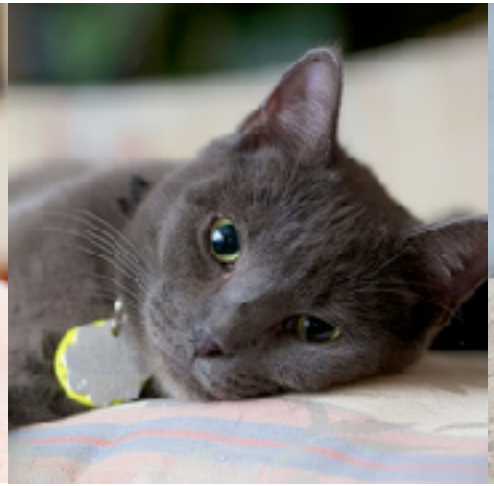
$$y = f(x) \text{ where } y \in \{1, \ldots N\}$$

Problems we will solve: Regression

0.4kg



2kg



4kg

10kg

Given image x estimate label y

$$y = f(x) \text{ where } y \in \mathbb{R}$$

Problems we will solve today: Sequence Models

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

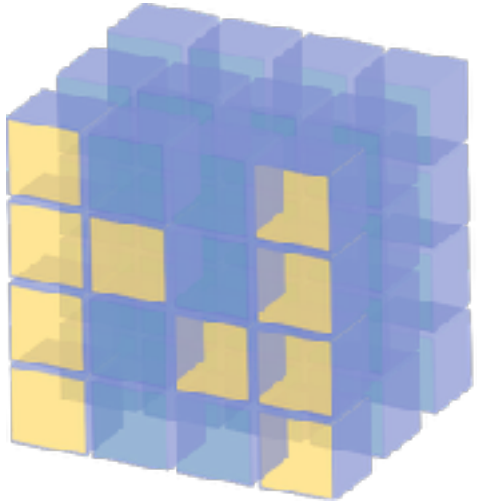
MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

"The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation," said Tom Hicks, the U.S. Energy Secretary, in a statement. "Our top priority is to secure the theft and ensure it doesn't happen again."

GPT2, 2019

Deep  NumPy

N-dimensional Arrays

N-dimensional arrays are the main data structure for machine learning and neural networks

0-d (scalar)	1-d (vector)	2-d (matrix)
		

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

A class label	A feature vector	An example-by-feature matrix

N-dimensional Arrays

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]]]
```

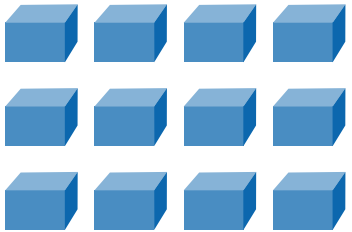
A RGB image (width x height x channels)

4-d

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

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

5-d

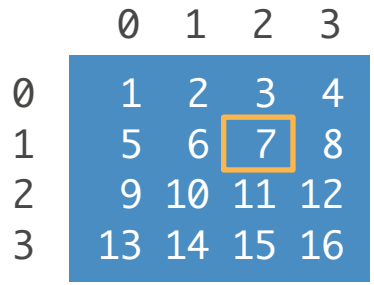


```
[[[[[. . .
    . . .
    . . .]]]]]
```

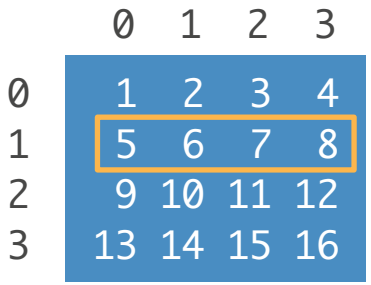
A batch of videos (batch-size x time x width x height x channels)

Element-wise access

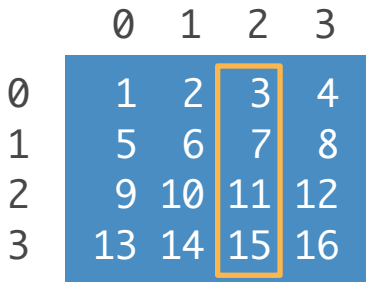
element: [1, 2]

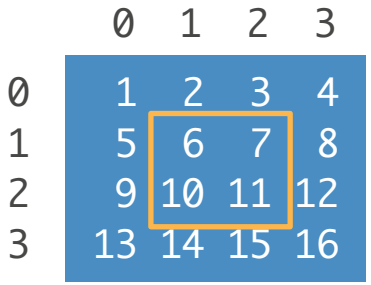


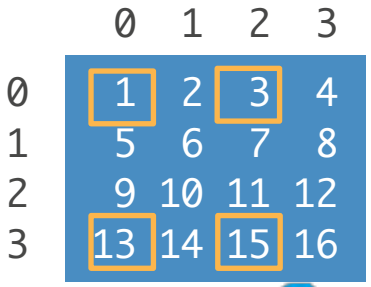
row: [1, :]

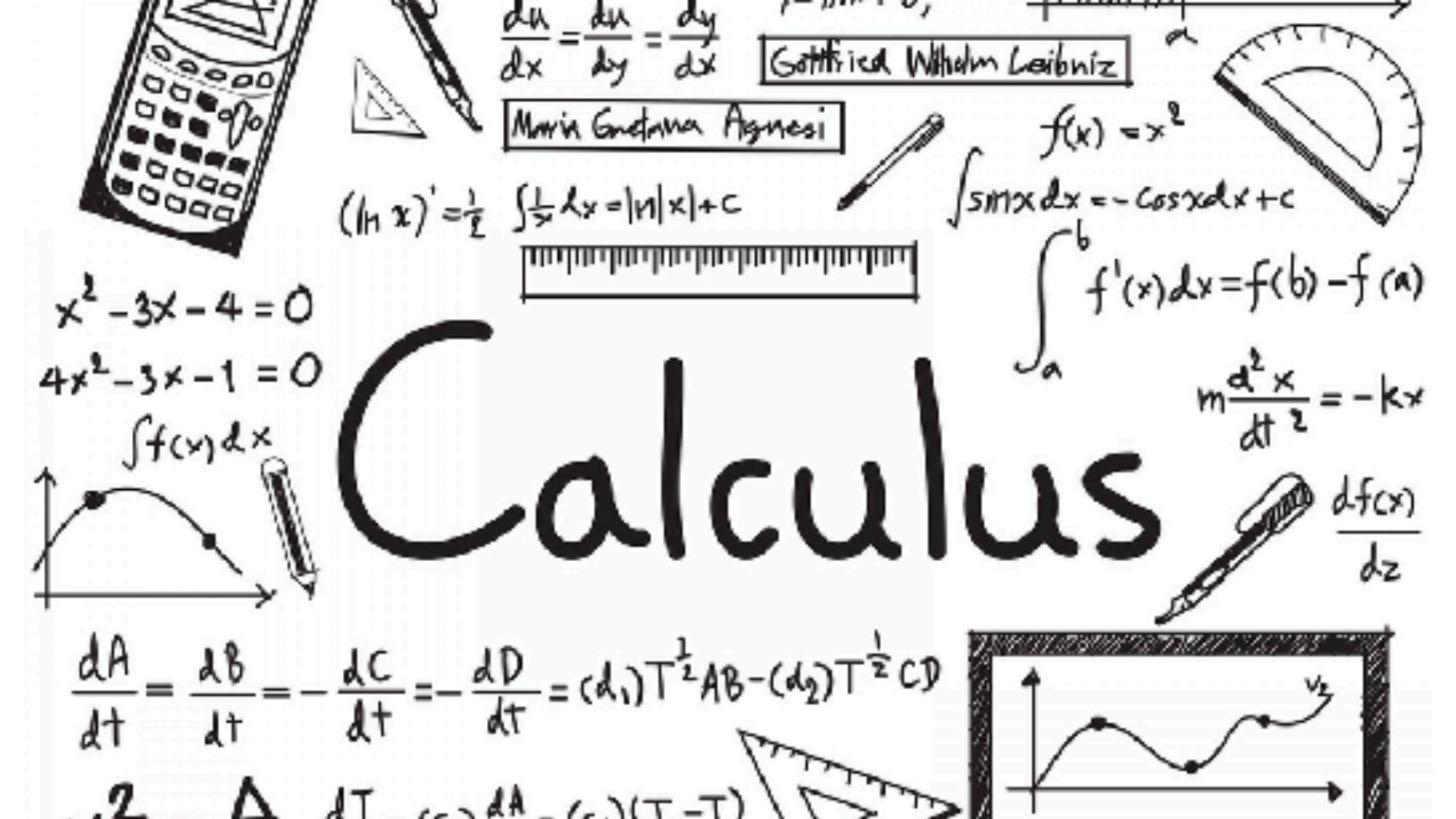


column: [1, :]








Calculus
Gottfried Wilhelm Leibniz
Maria Gaetana Agnesi
$\frac{dA}{dt} = \frac{dB}{dt} = -\frac{dC}{dt} = -\frac{dD}{dt} = (d_1)T^{\frac{1}{2}}AB - (d_2)T^{\frac{1}{2}}CD$
$x^2 - 3x - 4 = 0$
$4x^2 - 3x - 1 = 0$
$\int f(x)\,dx$
$\int_a^b f'(x)\,dx = f(b) - f(a)$
$m\frac{d^2x}{dt^2} = -kx$
$\frac{df(x)}{dz}$
$f(x) = x^2$
$\int \sin x\,dx = -\cos x\,dx + C$
$(\ln x)' = \frac{1}{x}$
$\int \frac{1}{x}\,dx = \ln|x| + C$
v_2

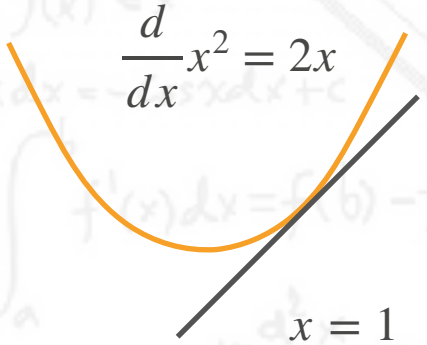
Calculus - Derivatives

Derivative measures the sensitivity to change of the output value with respect to a change in its input value.

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

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

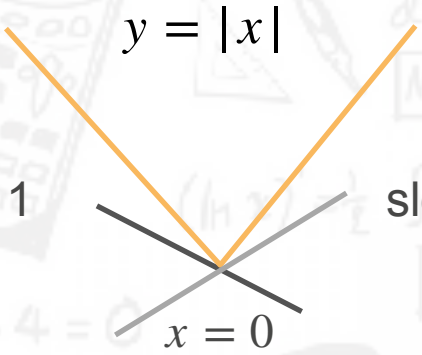
E.g. slope of tangent.



Calculus - Non-differentiable

Extend derivative to non-differentiable cases.

slope= - 1



slope=1

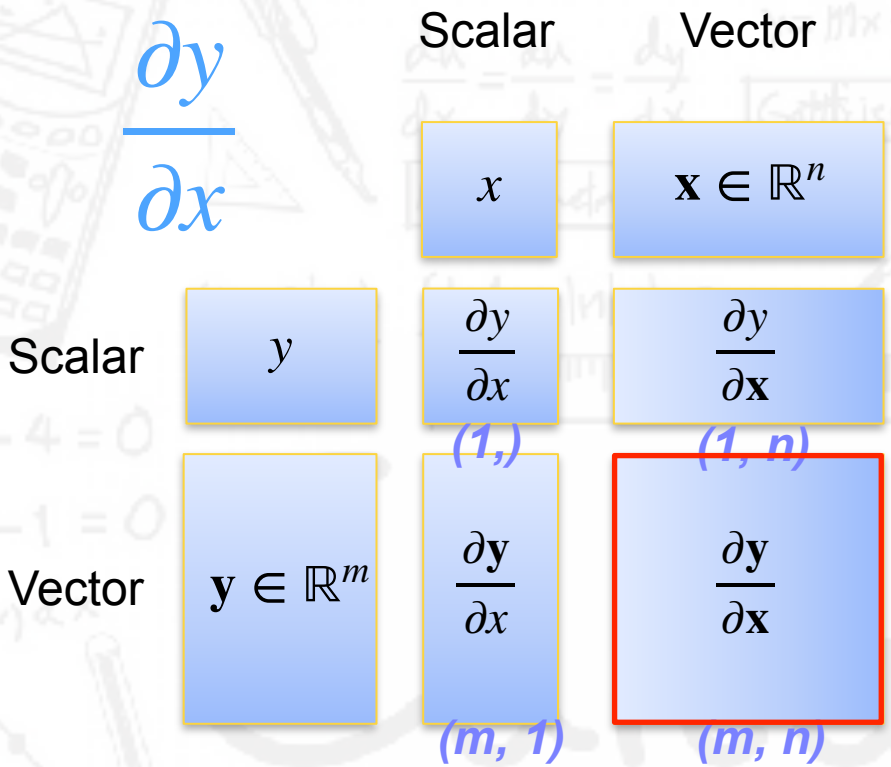
$$\frac{\partial |x|}{\partial x} = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ a & \text{if } x = 0, \quad a \in [-1,1] \end{cases}$$

Another example:

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

Calculus - Gradients

Gradient is a multi-variable generalization of the derivative.



Derivatives for vectors

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

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

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

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

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

$$\left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right]_{ij} = \frac{dy_i}{dx_j}$$

Derivatives for vectors

Let's do some exercise!

E.g.

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

$\mathbf{x} \in \mathbb{R}^n, \quad y \in \mathbb{R}, \quad \frac{\partial y}{\partial \mathbf{x}} \in \mathbb{R}^{1 \times n}$

0 and **1** are vectors

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

Derivatives for vectors

Let's do some exercise!

E.g.

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

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

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

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

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

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

Scalars

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

Vectors

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

Shapes: $(1,n)\ \ (1,)\ (1,n) \qquad (1,n)\ \ (1,k)\ \ (k,n) \qquad (m,n)\ \ (m,k)\ (k,n)$

Too many shapes to memory …

Automatic Differentiation

Computing derivatives by hand is HARD.

Chain rule (evaluate e.g. via backprop)

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

Compute graph:

- Build explicitly (TensorFlow, MXNet Symbol)
- Build implicitly by tracing (Chainer, PyTorch, DeepNumpy)

Automatic Differentiation

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

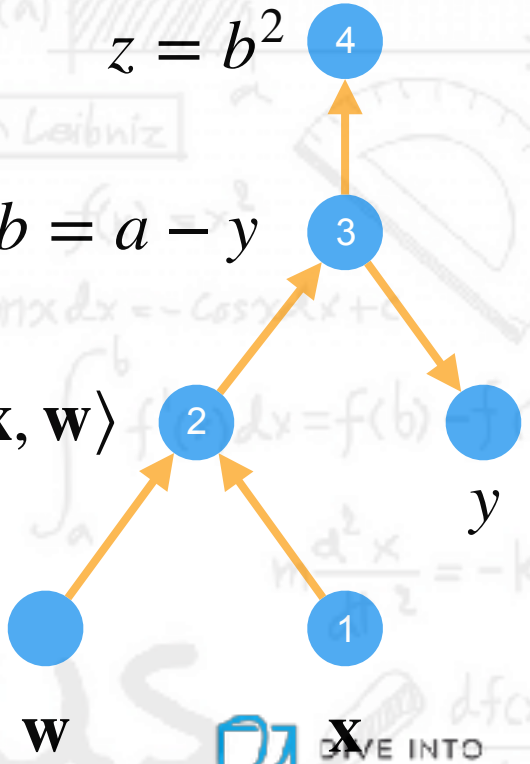
Computing derivatives by hand is HARD.

Chain rule (evaluate e.g. via backprop)

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

Compute graph:

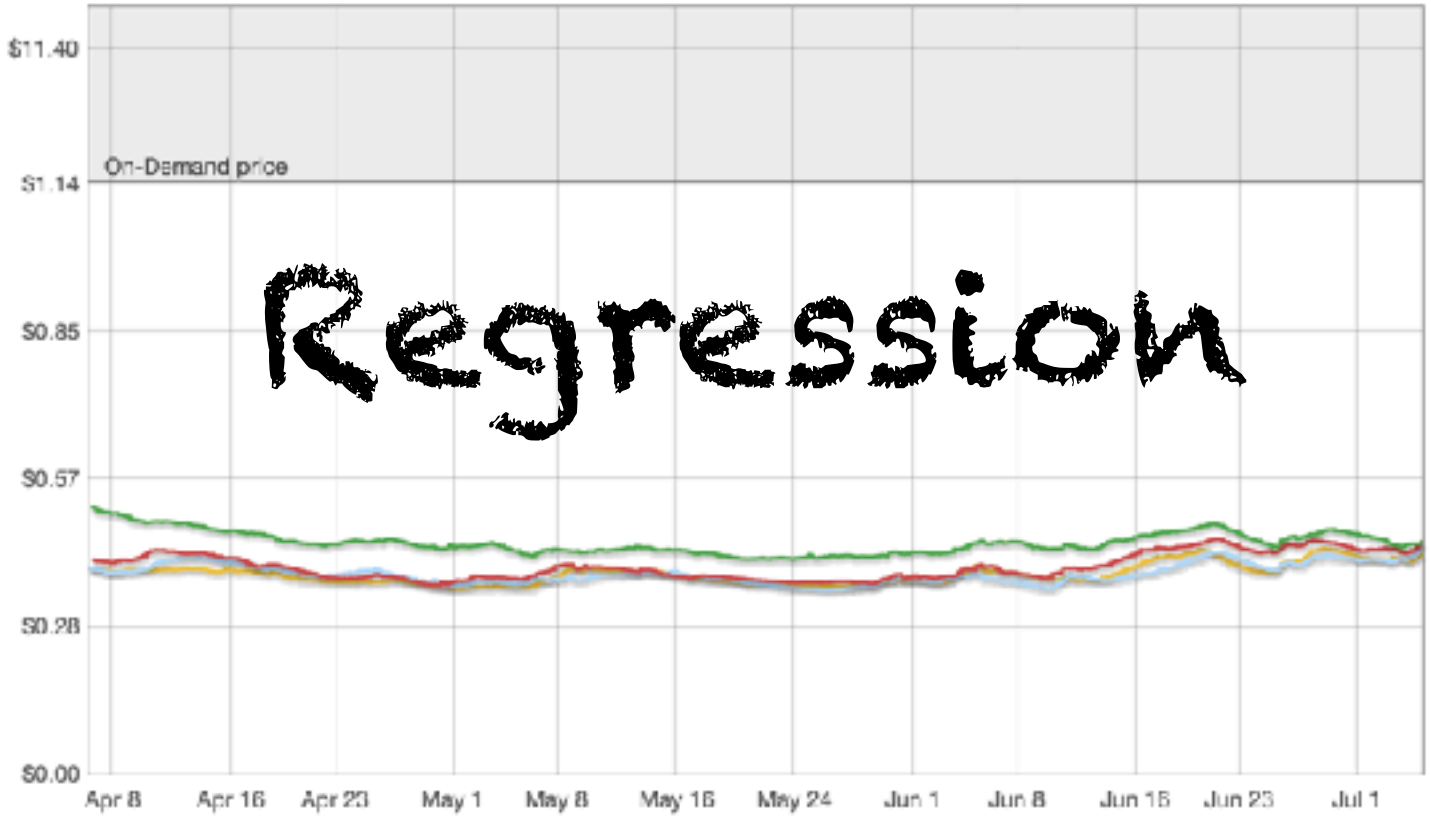
- Build explicitly (TensorFlow, MXNet Symbol)
- Build implicitly by tracing (Chainer, PyTorch, DeepNumpy)



NumPy & AutoGrad notebook

Spot Instance Pricing History

Product: Linux/UNIX Instance type: g3.4xlarge Date range: 3 months



Date

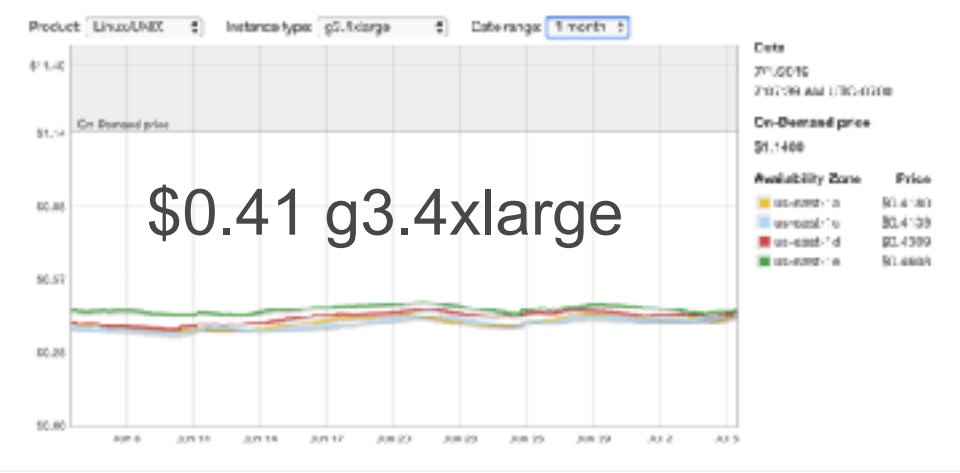
6/10/2019

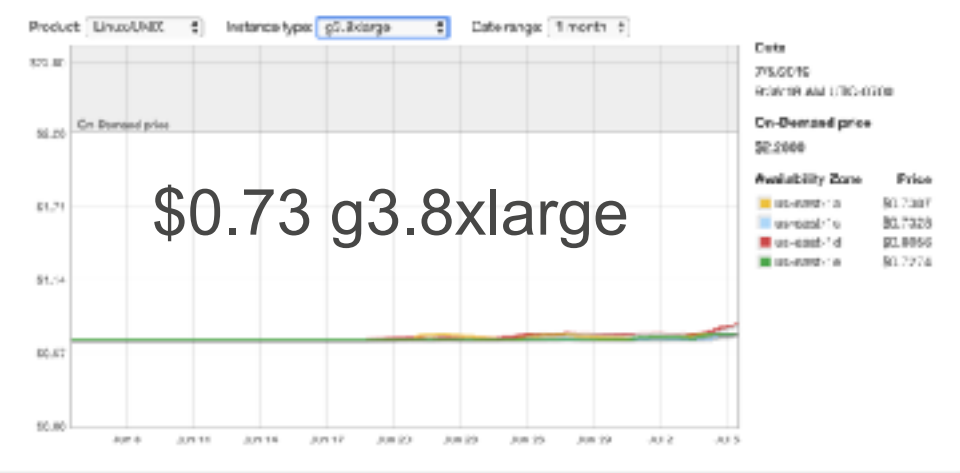
9:47:38 PM UTC-0700

On-Demand price

$1.1400

Availability Zone	Price
us-east-1a	$0.3741
us-east-1c	$0.3710
us-east-1d	$0.3693
us-east-1e	$0.4353





Can we estimate
prices (time, server, region)?

$$p = w_{\text{time}} \cdot t + w_{\text{server}} \cdot s + w_{\text{region}} \cdot r$$

Linear Model

- Basic version

$$\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

- Vectorized version

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

  - n-dimensional inputs $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$
  - Weights: $\mathbf{w} = [w_1, w_2, \ldots, w_n]^\top$
  - Bias, b

- Vectorized version (Closed form)
  - Add bias as an element in weights

$$\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}] \quad \mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \qquad \hat{y} = \langle \mathbf{w}, \mathbf{x} \rangle$$

l_2 Loss

- Basic version

$$\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

- Vectorized version

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

- Vectorized version (Closed form)

$$\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}] \quad \mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

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

- Basic version

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \left(\mathbf{y} - \hat{\mathbf{y}} \right)^2$$

- Vectorized version

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n} \left\| \mathbf{y} - \mathbf{X}\mathbf{w} - b \right\|^2$$

- Vectorized version (Closed form)

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

Objective Function

- Objective is to minimize training loss

$$\underset{w}{\text{argmin}}\, loss \Leftrightarrow \underset{w}{\text{argmin}}\, \frac{1}{n} \left\| \mathbf{y} - \mathbf{X}\mathbf{w} \right\|^2$$

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

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

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

- Basic version

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \left(\mathbf{y} - \hat{\mathbf{y}}\right)^2$$

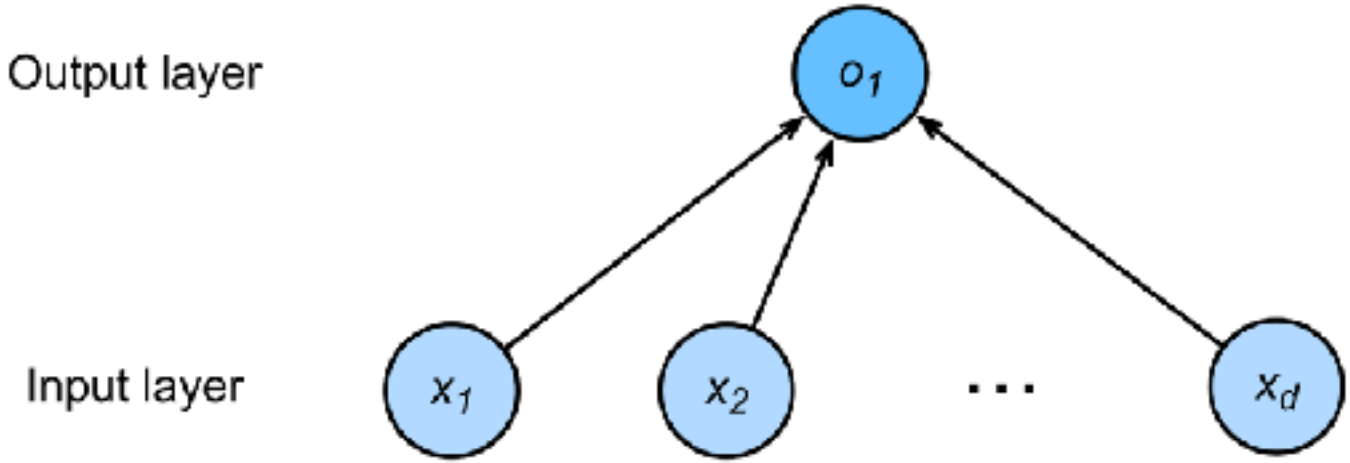
- Vectorized version

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n} \left\| \mathbf{y} - \mathbf{X}\mathbf{w} - b \right\|^2$$

- Vectorized version (Closed form)

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

Linear Model as a Single-layer Neural Network



We can stack multiple layers to get deep neural networks

Linear Regression notebook


Optimization
momentum
negative
gradient

Gradient Descent in 1D

Consider some continuously differentiable real-valued function f: $\mathbb{R} \to \mathbb{R}$. Using a Taylor expansion we obtain that:

$$f(x + \epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

Assume we pick a fixed step size (η>0) and choose $\epsilon = -\eta f'(x)$:

$$f(x - \eta f'(x)) = f(x) - \eta f'^2(x) + O(\eta^2 f'^2(x))$$

Gradient Descent in 1D

If the derivative $f'(x) \neq 0$ does not vanish, we make progress since $\eta f'^2(x) > 0$. Moreover, we can always choose η small enough for the higher order terms to become irrelevant. Hence we arrive at:

$$f(x - \eta f'(x)) \lessapprox f(x)$$

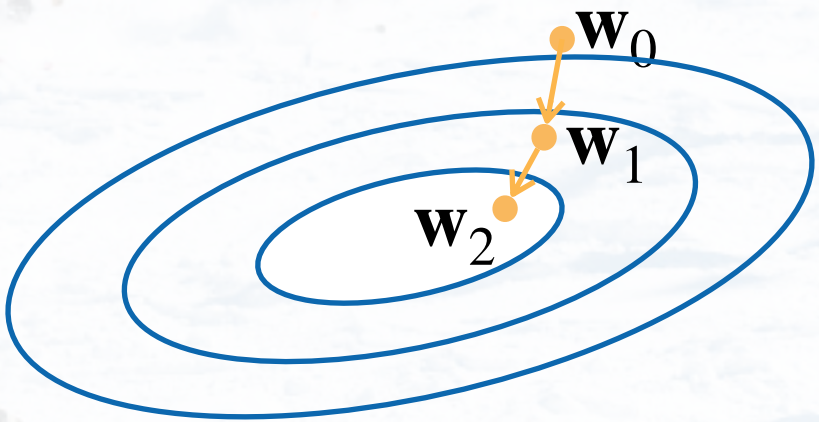
This means that, if we use $x \leftarrow x - \eta f'(x)$ to iterate x, the value of function $f(x)$ might decline.

Gradient Descent in General

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

Repeat to update weight

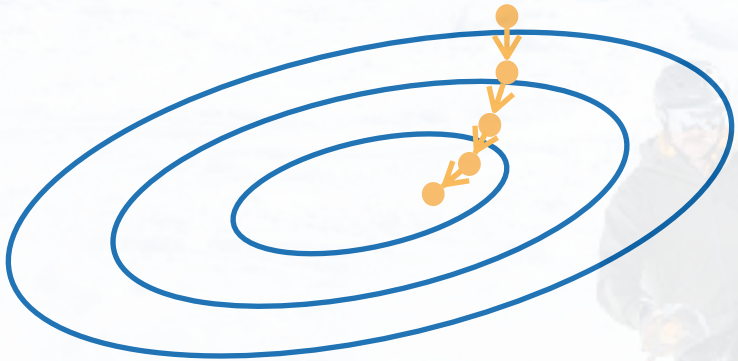
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \partial_w l(w_{t-1})$$



- Gradient direction indicates increase in value
- Learning rate adjusts step length

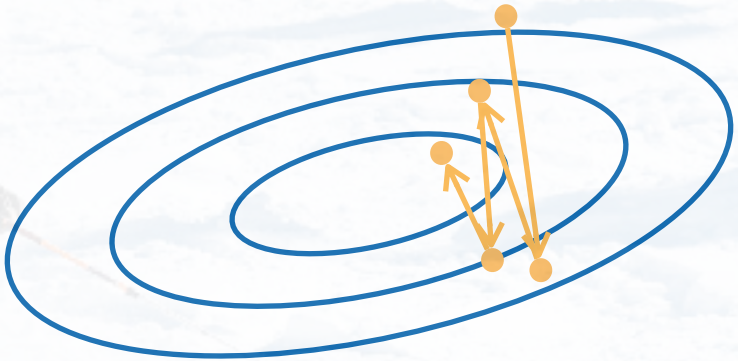
Goldilocks Learning Rate

Too small



More iterations

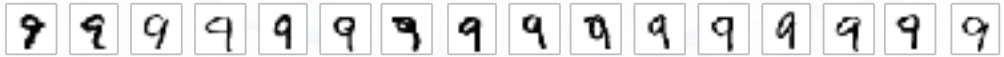
Too big



May diverge

Mini-batch Stochastic Gradient Descent (SGD)

- Computing the gradient over all data is too slow
- Redundancy in data (e.g. many similar digits)



- Single observation is not efficient on GPU

- Sample b examples $i_1, \ldots i_b$ to approximate loss/gradient

$$\frac{1}{b}\sum_{i \in I_b} l(\mathbf{x}_i, y_i, \mathbf{w}) \text{ and } \frac{1}{b}\sum_{i \in I_b} \partial_{\mathbf{w}} l(\mathbf{x}_i, y_i, \mathbf{w})$$

b is the mini-batch size (chosen for GPU efficiency)

Gradient Descent (SGD)

	Batch Size	Computation	Memory	Pros and Cons
GD	Training Size	Efficient	Inefficient	Parallel processing available stable gradient descent, but maybe bad (saddle point).
Mini-batch GD	n	Okay	Okay	A compromise that injects enough noise to each gradient update, while achieving a relative speedy convergence
SGD	1	Inefficient	Efficient	Can escape from saddle points or local minimal, but maybe very noisy

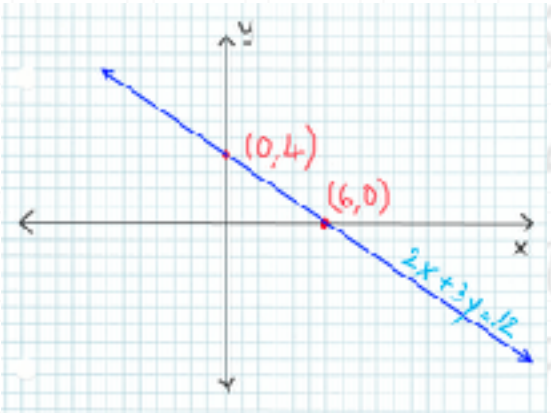

Softmax Regression

Regression vs. Classification

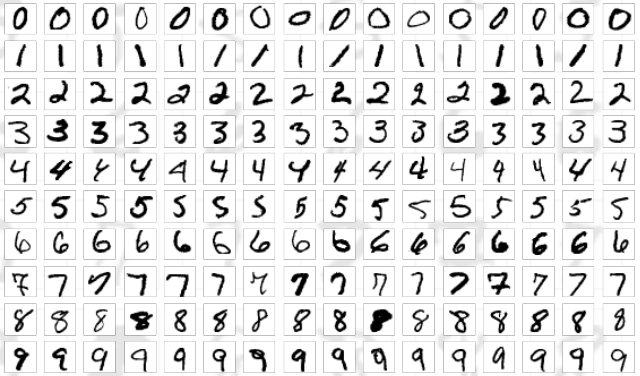
Regression estimates a continuous value

Classification predicts a discrete category

Housing price predictions



MNIST: classify hand-written digits (10 classes)

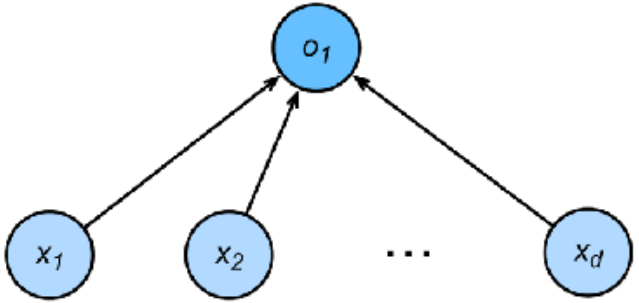


DIVE INTO DEEP LEARNING

From Regression to Multi-class Classification

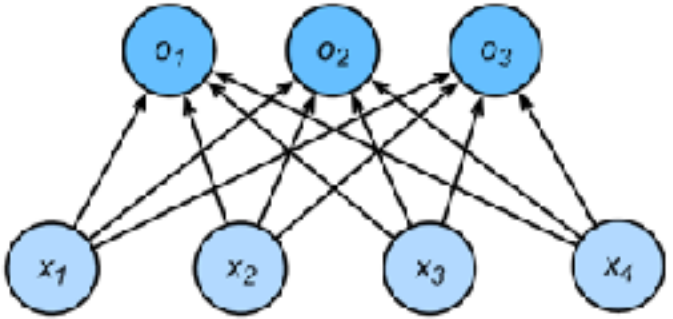
Regression

- Single continuous output
- Natural scale in $\mathbb{R}$
- One Loss given $y - f(x)$
  - e.g. in terms of difference



Multi-class Classification

- Multiple outputs, one for each class
- Multiple Loss given
- Outputs should reflect confidence



From Regression to Multi-class Classification

- One hot encoding per class

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$$

$$y_i = \begin{cases} 1 \text{ if } i = y \\ 0 \text{ otherwise} \end{cases}$$

- Train with squared loss
- Largest output wins
  - $\hat{y} = \underset{i}{\operatorname{argmax}}\, o_i$
  - *max* is not differentiable

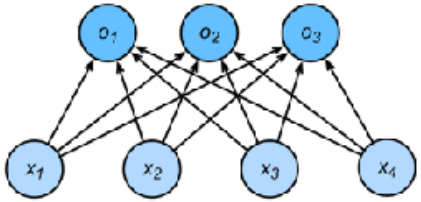
Use Square Loss?

Example:

Given y = [0, 0, 1]

Predict $\hat{y}$ = [0.3, 0, 7000]

Losses are not in the same scale!



From Regression to Multi-class Classification

- One hot encoding per class

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$$

$$y_i = \begin{cases} 1 \text{ if } i = y \\ 0 \text{ otherwise} \end{cases}$$

- Train with squared loss
- Largest output wins
  - $\hat{y} = \operatorname*{argmax}_i o_i$
  - *max* is not differentiable

Softmax Regression

- Probability indicates confidence (nonnegative, sums to 1)

$$\text{softmax}([o_1, o_2, \ldots, o_n]^T) = \left[\frac{e^{o_1}}{\sum_{i=1}^n e^{o_i}}, \frac{e^{o_2}}{\sum_{i=1}^n e^{o_i}}, \ldots, \frac{e^{o_n}}{\sum_{i=1}^n e^{o_i}} \right]$$

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

softmax: [0.26, 0.04, 0.7]

Cross-Entropy Loss

- (element-wise) Negative log-likelihood (for given class $y \in \{0,1\}$)

$$-\log p(y \mid o) = \log \sum_i \exp(o_i) - o_y$$

- (vector-wise) Cross-Entropy Loss (for probability distribution $y_i \in [0,1]$)

$$l(\mathbf{y}, \mathbf{o}) = \log \sum_i \exp(o_i) - \mathbf{y}^\top \mathbf{o}$$

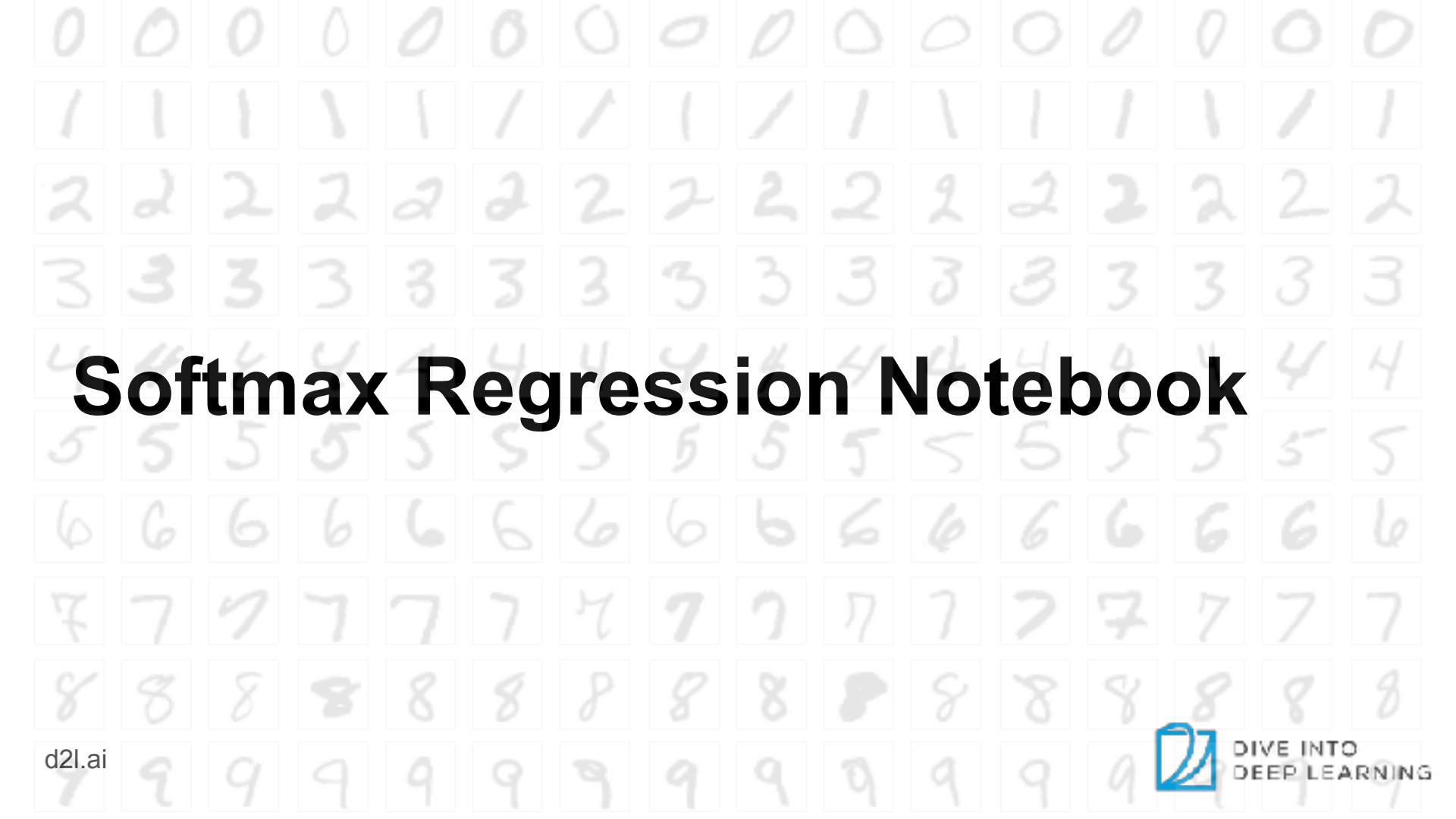
- Gradient

Difference between true and estimated probability

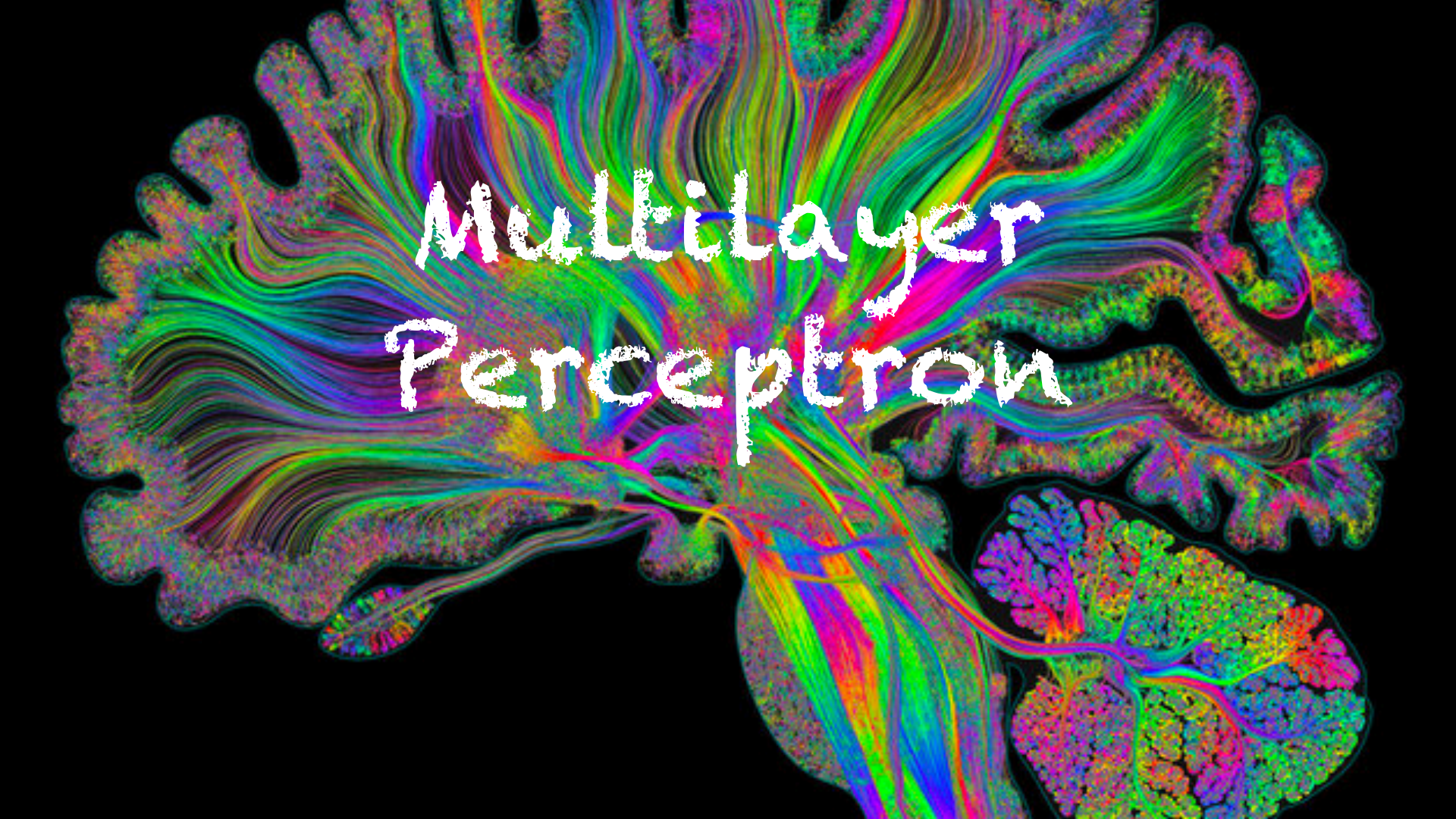
$$\partial_{\mathbf{o}} l(\mathbf{y}, \mathbf{o}) = \frac{\exp(\mathbf{o})}{\sum_i \exp(o_i)} - \mathbf{y}$$

Regression vs Softmax Regression

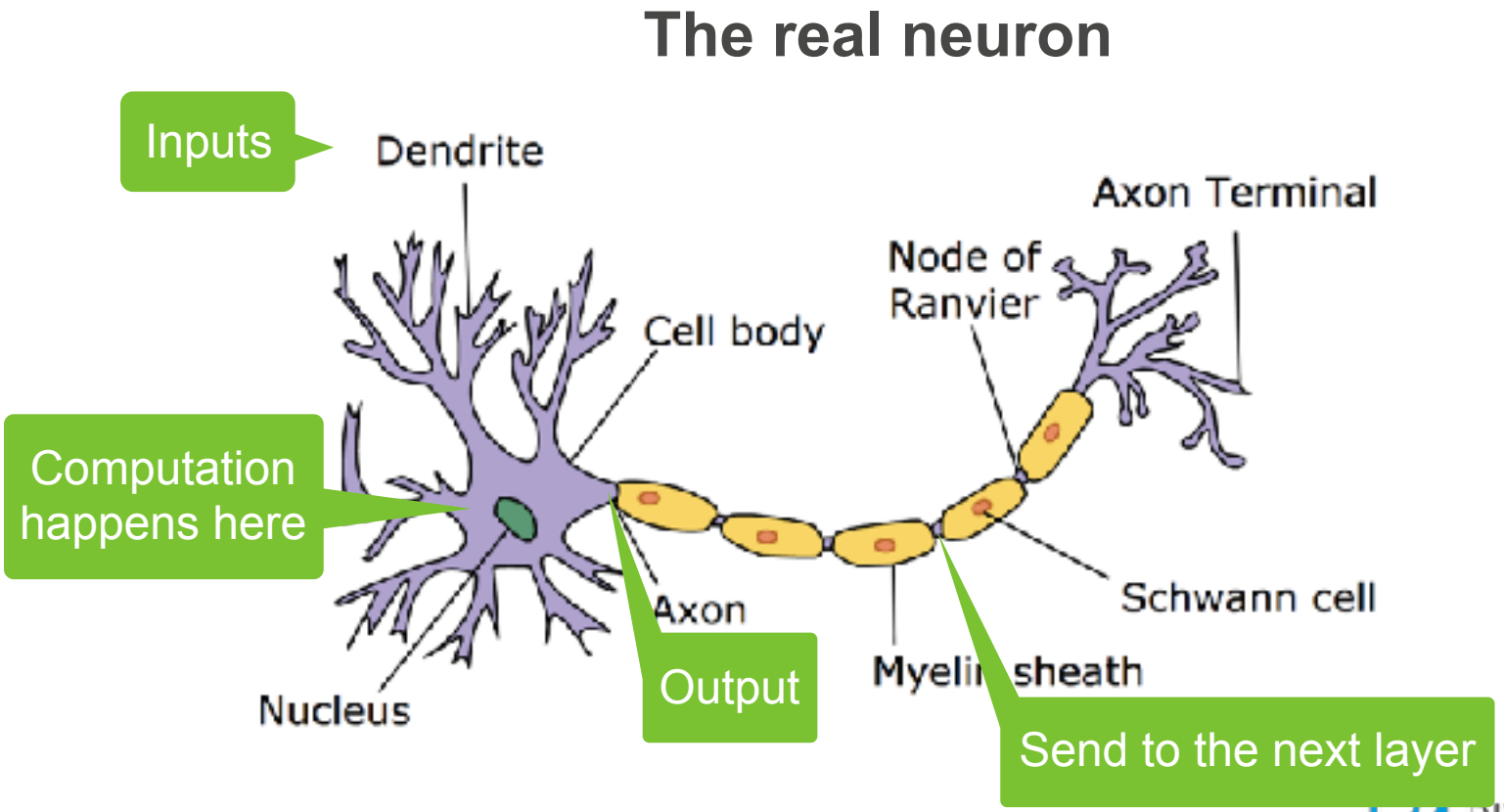
Problem	Regression (Linear Model)	Softmax Regression (k-class classification)
Model	$\langle \mathbf{w}, \mathbf{x} \rangle + b$ $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$	$\text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$ $\mathbf{W} \in \mathbb{R}^{k \times n}, \mathbf{b} \in \mathbb{R}^k$
Loss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



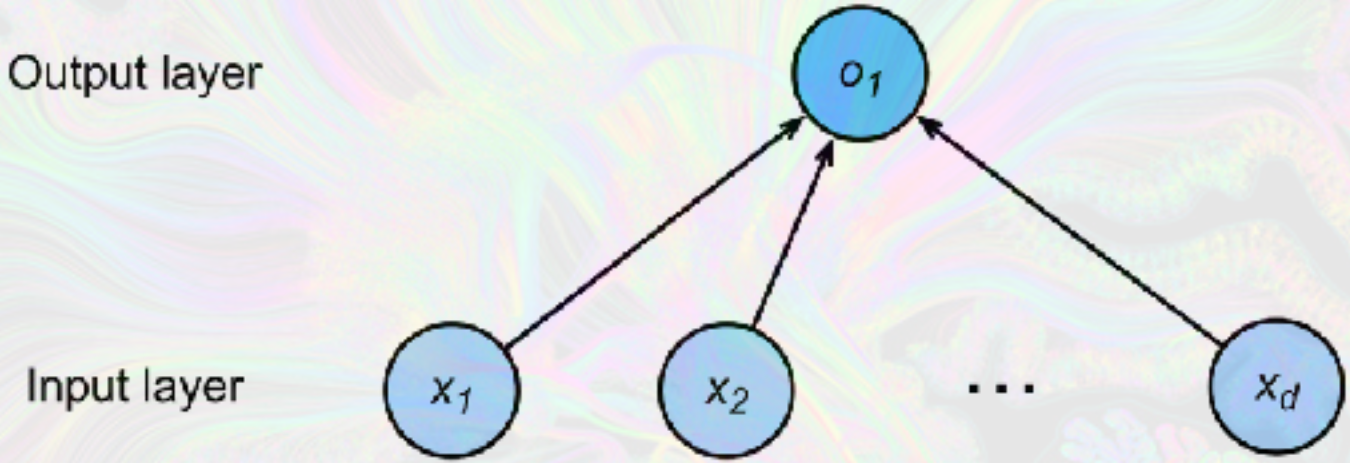
Softmax Regression Notebook


Multilayer
Perceptron

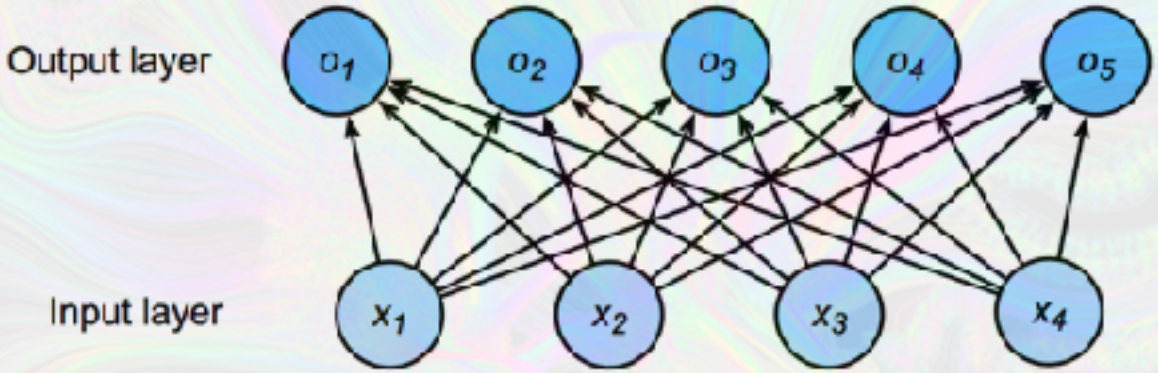
Neural Networks Derive from Neuroscience



Linear Model as a Single-layer Neural Network

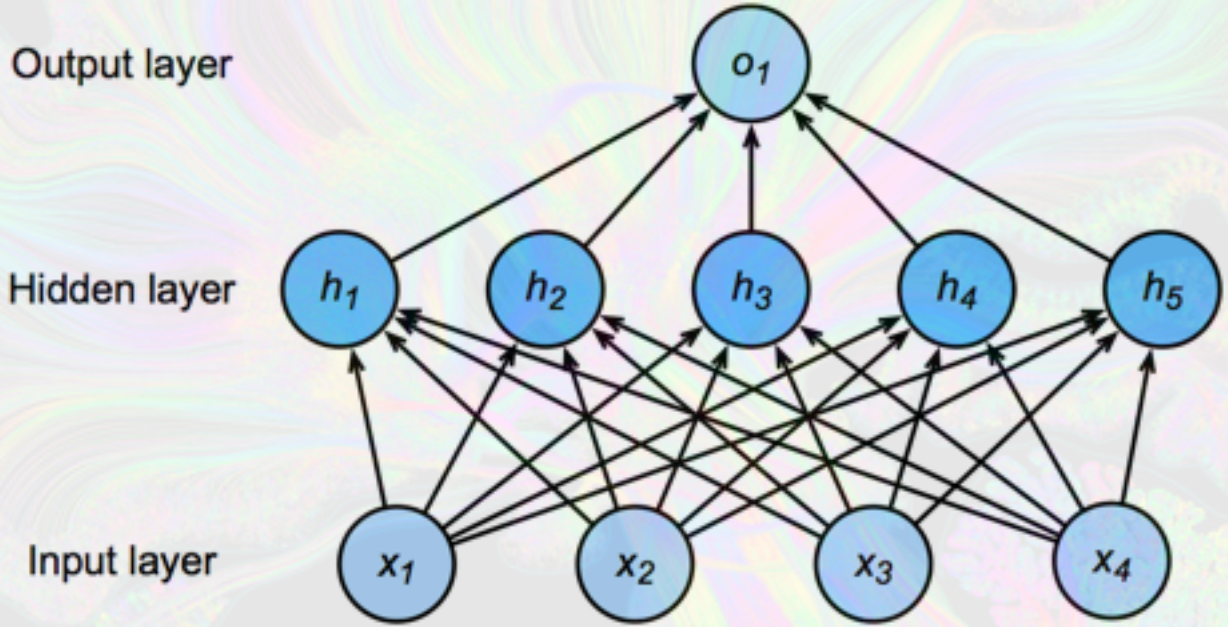


Single Hidden Layer



We can stack multiple neurons in one layer.

Single Hidden Layer



We can stack multiple layers to get deep neural networks.

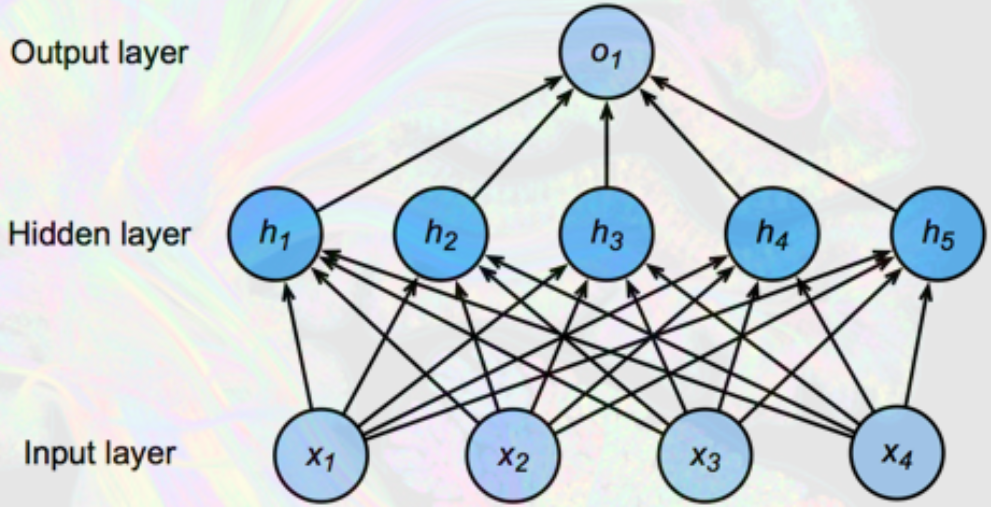
Single Hidden Layer

- Input $\mathbf{x} \in \mathbb{R}^n$
- Hidden $\mathbf{W}_1 \in \mathbb{R}^{m \times n}, \mathbf{b}_1 \in \mathbb{R}^m$
- Output $\mathbf{w}_2 \in \mathbb{R}^m, b_2 \in \mathbb{R}$

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T \mathbf{h} + b_2$$

σ is an element-wise activation function



DIVE INTO DEEP LEARNING

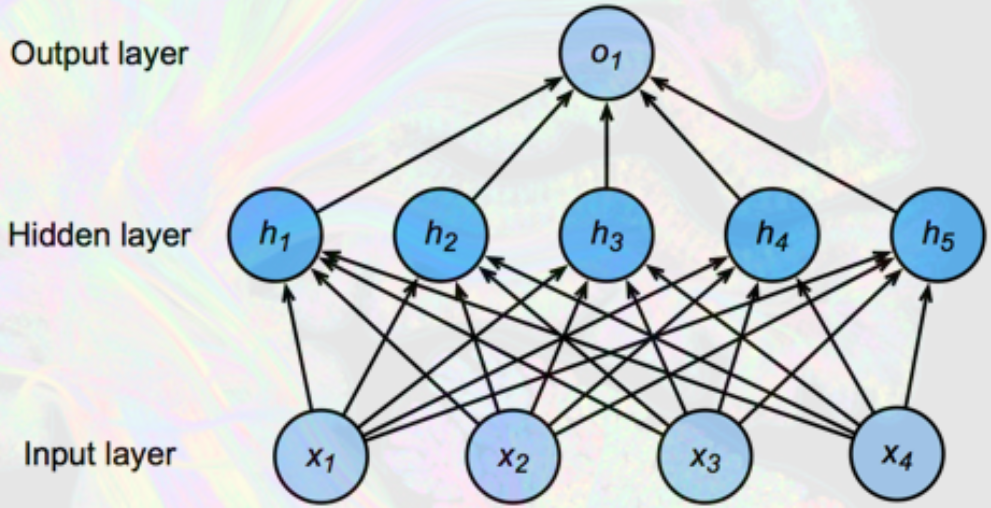
Single Hidden Layer

Why do we need an a nonlinear activation?

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + b_2$$

σ is an element-wise activation function



Single Hidden Layer

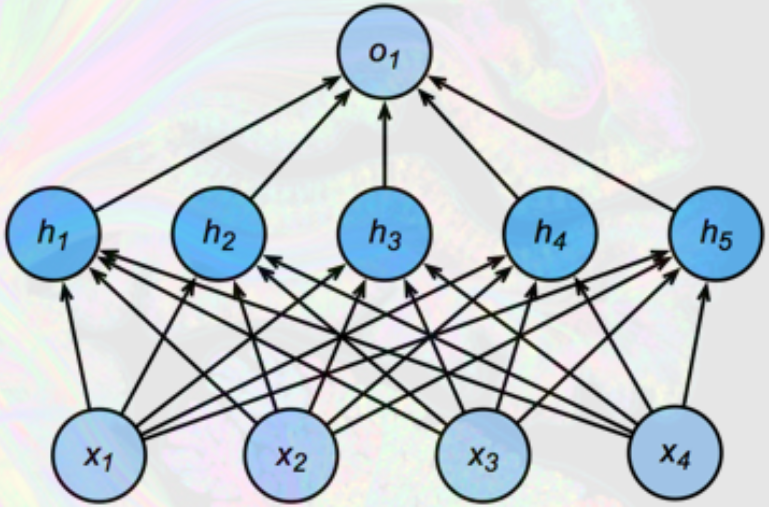
Why do we need an a nonlinear activation?

$$\mathbf{h} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + b_2$$

hence $o = \mathbf{w}_2^\top\mathbf{W}_1\mathbf{x} + b'$

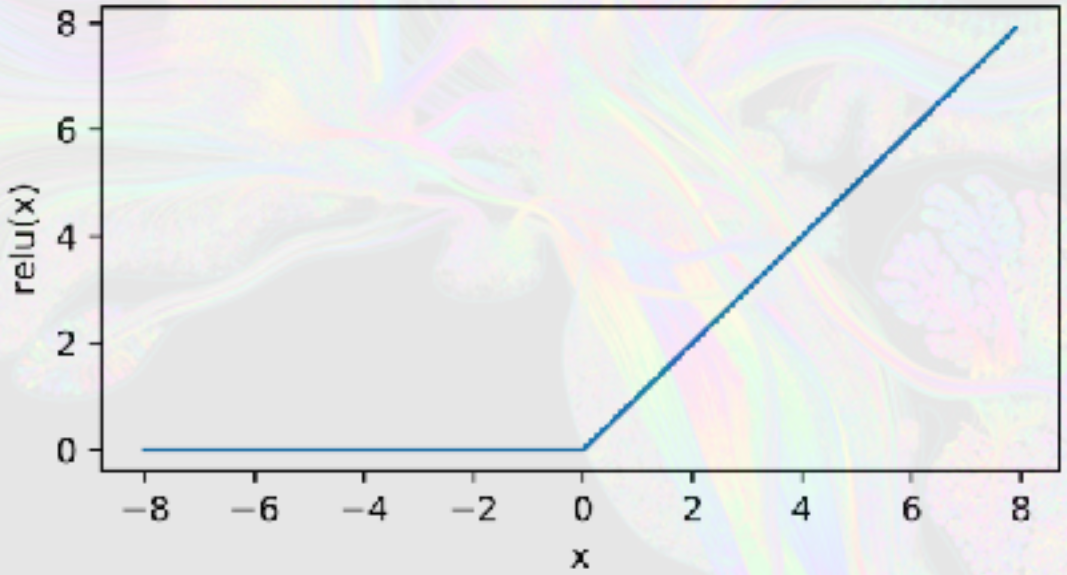
Linear ...



ReLU Activation

ReLU: rectified linear unit

$$\mathrm{ReLU}(x) = \max(x, 0)$$

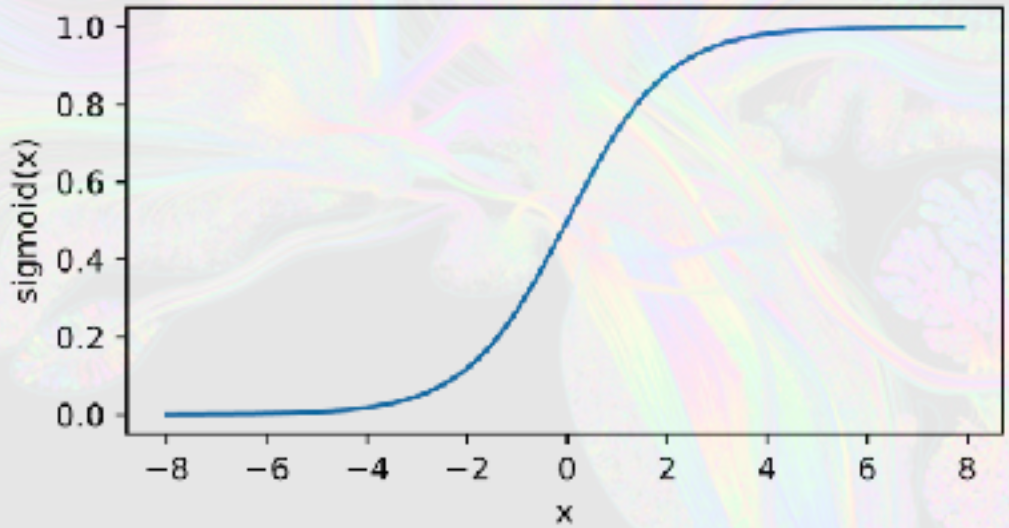


DIVE INTO DEEP LEARNING

Sigmoid Activation

Map input into (0, 1), a soft version of $\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

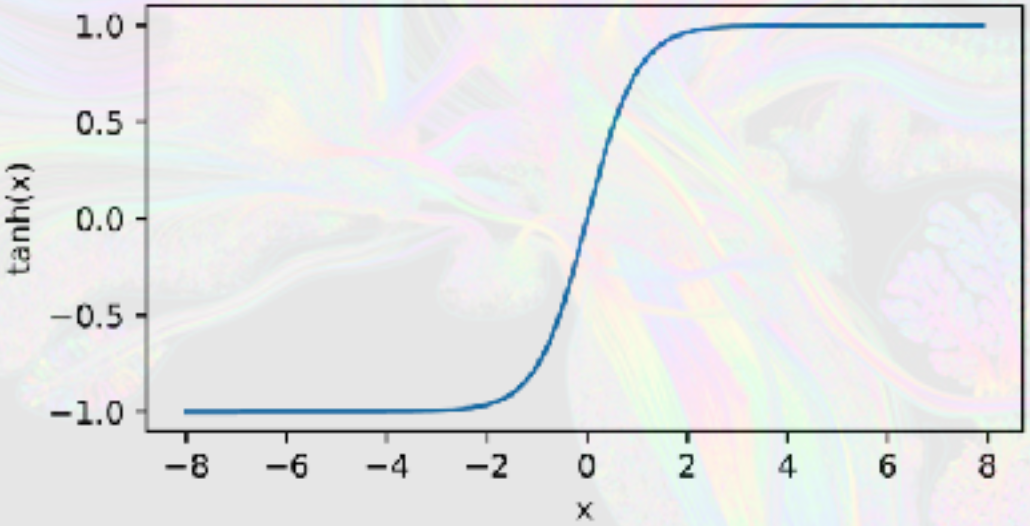
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$



Tanh Activation

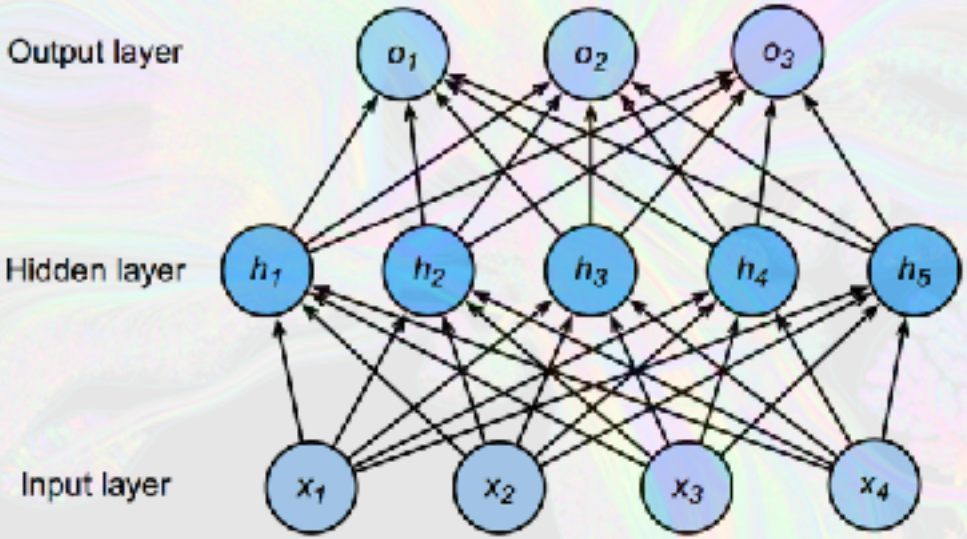
Map inputs into (-1, 1)

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$



Multiclass Classification

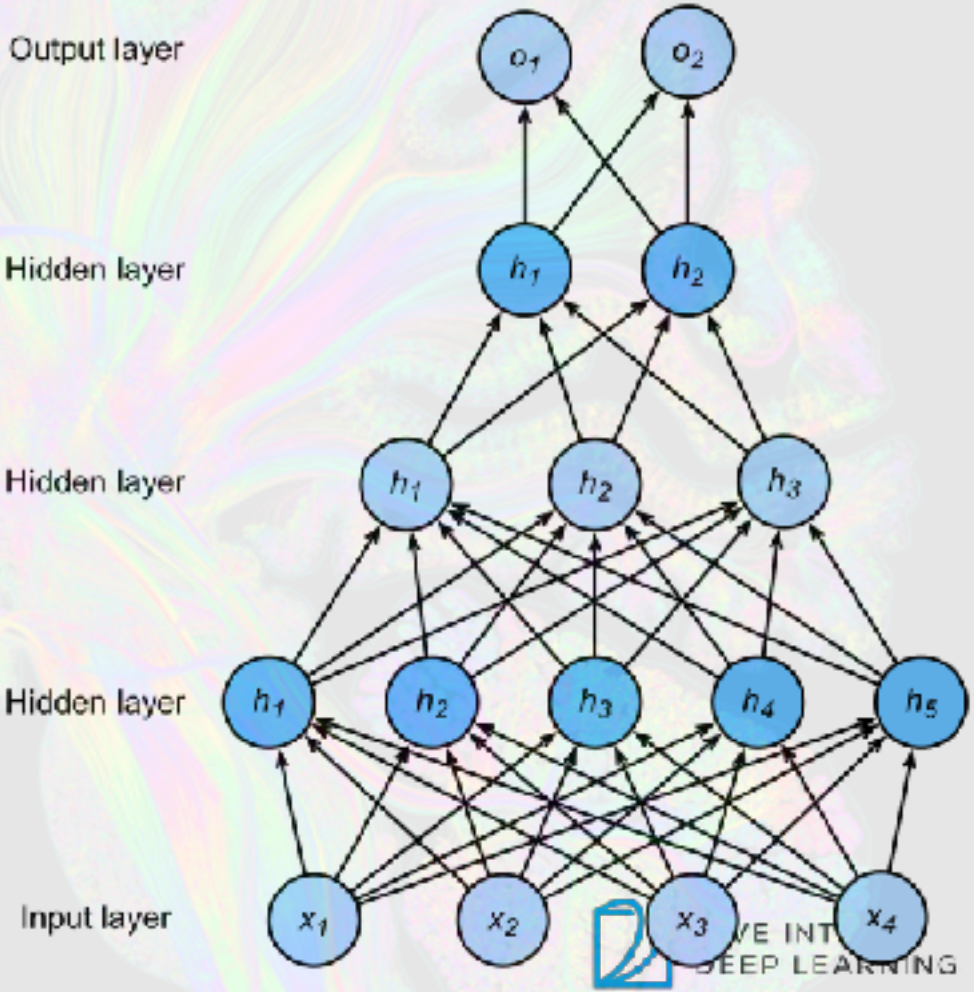
$$y_1, y_2, \ldots, y_k = \text{softmax}(o_1, o_2, \ldots, o_k)$$



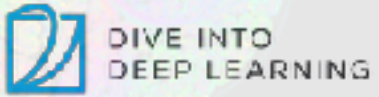
Multiple Hidden Layers

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{o} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$

Hyper-parameters
- # of hidden layers
- Hidden size for each layer



MLP Notebook



Summary

- Installations
- Deep Learning Motivations
- DeepNumpy & Calculus
- Regression
- Optimization
- Softmax Regression
- Multilayer Perceptron (train MNIST)


Questions?