University of Zurich[UZH]

Institute of Computational Linguistics

Machine Translation

5 Math Fundamentals

Mathias Müller

Watch talks from the 2019 Tensor

TensorFlow > API r1.13 > Python

tf.linalg.matmul

matrix multiplication

Aliases:

- `tf.linalg.matmul`
- `tf.matmul`

```
tf.linalg.matmul(
    a,
    b,
    transpose_a=False,
    transpose_b=False,
```

Last time

PB-SMT

Translation: rank hypotheses by s

Input : "Fallout 76 is a crap

- for new sentences:

"Fallout 76 ist ein tolles S

- we now know the TM score and LM score

TM score : 0.0071 LM score : 0.00001

- and can combine them:

$$\text{score} = \text{TM score}^{\lambda_{TM}} * \text{LM score}^{\lambda_{LM}}$$

$\lambda_{TM} = 0.7$ $\lambda_{LM} = 0.3$

Statistical poetry!

Moses

Wie Moses sich ganz leis und schnell,
von reinem Text ernährt,
am besten viel und parallel,
wird hier im Gedicht erklärt.

Nimm den Text und gib ihn schlicht,
in einen Satz-Aligner,
der sagt was sich entspricht,
und schon ist die Struktur viel feiner.

fast_align

Jetzt ist klar, was Sätze sind,
doch Wörter sind noch ganz verloren,
aber nur bis Giza ganz geschwind,
hat Alignment-Punkte auserkoren.

IBM Model 1, 2, 3
draus Phrasen extrahiert,
ist keine Hexerei,
mit grow-diag-final navigiert.

So kriegt man auf die schnelle,
eine schöne Phrasentabelle!

Ein Sprachmodell dazu, trainiert,
auf Zielsprachtext, ne ganze Menge,
das bewertet Sätze ungeniert,
treibt die Übersetzung in die Enge.

A language model, trained,
To target language, a whole lot,
Which evaluates sentences
uninhibited,
Drives the translation into the
narrowness.

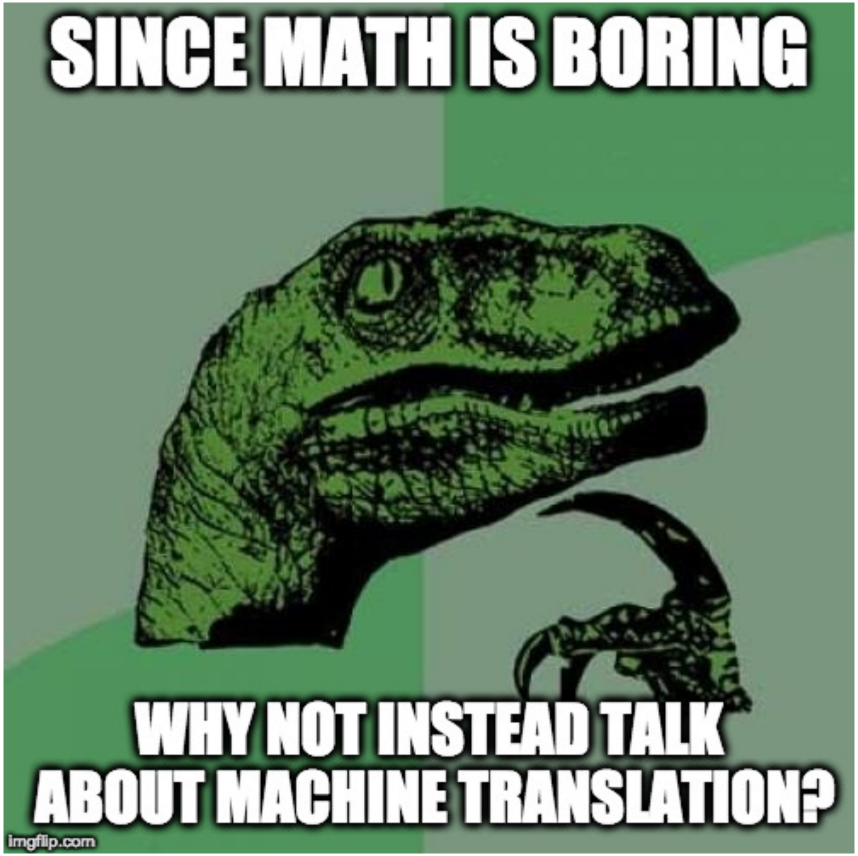
Neue Sätze schliesslich gibt man,
dem Decoder, der aus Kandidaten,
den besten finden kann,
mit log-linearem Raten.

Automatisch evaluieren immer,
mit BLEU und METEOR und TER,
nicht schwieriger oder schlimmer,
als Kochen mit Jamie Oliver.

Das ist dir zu banal?
Dann werd neuronal.

Topics of Today

- **linear algebra** concepts, such as vectors, or dot products
- Python library **numpy**, most important functions
- **differential calculus** concepts, such as slope, rate of change, derivative


SINCE MATH IS BORING
WHY NOT INSTEAD TALK ABOUT MACHINE TRANSLATION?
imgflip.com

Why math topics

only NMT

- **linear algebra** because most computation in NMT sytems is tensor manipulation

- **differential calculus** because learning in neural networks is guided by the derivatives of functions

Linear Algebra

Linear Algebra

Concepts we will cover:

- **objects:** scalars, vectors, matrices, tensors

- **operations** defined on objects: element-wise, dot product, sum, (norm, …)

addition

Objects

	Notation	Example
scalar	0 a	17
vector	1 $\mathbf{a}$ $\vec{a}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$ 3×1 $\begin{bmatrix}4 & 5 & 6\end{bmatrix}$ 1×3
matrix	2 A $\mathbb{R}^3$ $\mathbb{R}^2$ 3×2	$\begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix}$
|| tensor	n T	$\begin{bmatrix}\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}\\\begin{bmatrix}1 & 0 & -1\\7 & 8 & 9\end{bmatrix}\end{bmatrix}$

Row vectors vs. column vectors

row

$$\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

column

$$\vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Objects

- all objects are a kind of tensor

- all operations operate on tensors
  - defined only for vectors

Tensor Operations

- important operations are
  - **element-wise** operations

$$10 * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \longrightarrow \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \end{bmatrix}$$

  - **aggregate** operations

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \longrightarrow 21$$

Element-wise addition and multiplication

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$s = 2$$

$$s * M$$

$$= \begin{bmatrix} 1*2 & 2*2 & 3*2 \\ 0*2 & -1*2 & 1*2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 4 & 6 \\ 0 & -2 & 2 \end{bmatrix}$$

Sum of tensor elements

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix} \qquad \vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\operatorname{sum}(M)$$

$$= 1 + 2 + 3 + 0 + (-1) + 1$$

$$= \underline{\underline{6}}$$

Vector-vector multiplication: dot product

$$\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \vec{b} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\vec{a} \cdot \vec{b} = \underbrace{1 \times 4}_{4} + \underbrace{2 \times 5}_{10} + \underbrace{3 \times 6}_{18}$$

$$= \underline{\underline{32}}$$

Matrix-vector multiplication (sorry!)

right multiplication

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$M\vec{a}$$

left multiplication

$$N = \begin{bmatrix} 1 & 0 \\ 2 & -1 \\ 3 & 1 \end{bmatrix}$$

$$\vec{b} = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

$$\vec{b}N$$

Matrix-vector multiplication: right

$m_1 = [1 \;\; 2 \;\; 3]$

$$M = \begin{array}{c} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix} \end{array} \begin{array}{l} m_1 \\ m_2 \end{array} \qquad \vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$2 \times 3 \qquad 3$

$$M\vec{a} = \begin{bmatrix} (m_1)^T \cdot \vec{a} \\ (m_2)^T \cdot \vec{a} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \\ \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \underbrace{1 \times 1}_{1} + \underbrace{2 \times 2}_{4} + \underbrace{3 \times 3}_{9} \\ \underbrace{0 \times 1}_{0} + \underbrace{(-1)2}_{-2} + \underbrace{1 \times 3}_{3} \end{bmatrix}$$

$$\left\{ \begin{array}{l} \cancel{14} \\ 1 \end{array} \right. \quad \begin{bmatrix} 14 \\ 1 \end{bmatrix}$$

Matrix-vector multiplication: left

$$\vec{b} = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix} \qquad 1 \times 3$$

$$\begin{array}{cc} \vec{n}_1 & \vec{n}_2 \end{array} \qquad \vec{n}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$N = \begin{bmatrix} 1 & 0 \\ 2 & -1 \\ 3 & 1 \end{bmatrix} \qquad 3 \times 2$$

$$\vec{b} N = \begin{bmatrix} \vec{b} \cdot \vec{n}_1 & \vec{b} \cdot \vec{n}_2 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} & \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \underbrace{3 \times 1}_{3} + \underbrace{2 \times 2}_{4} + \underbrace{1 \times 3}_{3} & 3 \times 0 + 2(-1) + 1 \times 1 \end{bmatrix}$$

$$\left[10 \quad -1\right]$$

Matrix-matrix multiplication

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix} \qquad B = \begin{array}{c} \begin{array}{cc} \vec{b}_1 & \vec{b}_2 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 2 & -1 \\ 3 & 1 \end{bmatrix} \end{array}$$

$$2 \times 3 \qquad\qquad 3 \times 2$$

$$AB = \begin{bmatrix} A\vec{b}_1 & A\vec{b}_2 \end{bmatrix}$$

$$\neq$$

$$BA = \begin{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} & \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} & \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \end{bmatrix}$$

$$2 \times 2$$

Summary tensor-tensor-multiplication

	$M\vec{a}$	$\vec{b}N$	AB
shape constraints	columns of M = rows of $\vec{a}$	rows of N = columns of $\vec{b}$	columns of A = rows of B
result type	column vector	row vector	matrix
result dimensions	# rows in M	# columns in N	(# rows in A, # columns in B)

- linalg ✓
- **Numpy** ←
- derivatives

numpy

Python

- library for scientific computing

pip install numpy

- knows tensors, but calls them **arrays**

- implements plenty of array operations

In numpy, tensors are arrays

```
>>> import numpy as np
```

array([[1,2],[3,4]])

- how to construct an array

```
>>> a = np.array([1,2,3])
```

- array has a shape

```
>>> a.shape
(3,)
```

(2,2)

- elements in array have a data type

```
>>> a.dtype
np.int32
```

Important functionality in numpy

Research the following topics: 2×3

a) how to generate an array with random numbers, with a specific shape ~~and dtype~~

b) how to add two arrays element-wise

c) how to compute a matrix-vector right multiplication

b) $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$

→ row / column vectors ! ! ☹

Important functionality in numpy

a) >>> r = np.random.sample((2, 3))

array([[0.199956, ...],
[...   ]])

6 → (2, 3)

b) >>> c = a + b

__add__

c) >>> np.dot(a, v)

1 x 3   3 x 1

Summary Numpy

- numpy can represent arbitrary tensors as arrays
- efficient implementations of very many tensor operations

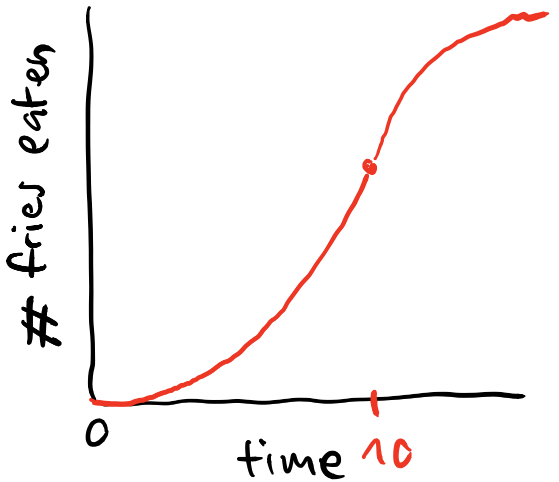
University of Zurich[UZH]
Institute of Computational Linguistics

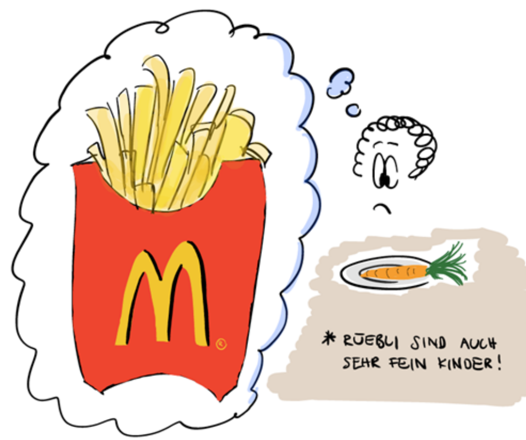
differential

Calculus:
Intuitions about Derivatives

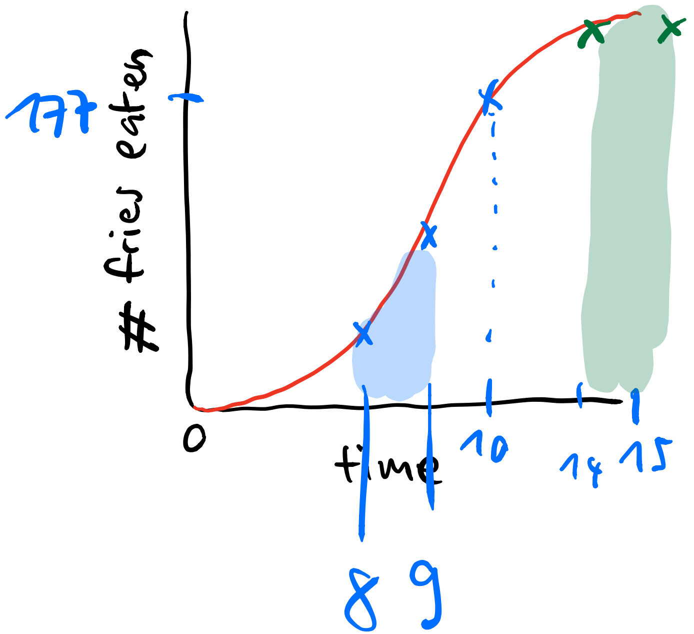
A single-input, single-output function

$$f(10) = 20$$





How functions change as the input changes

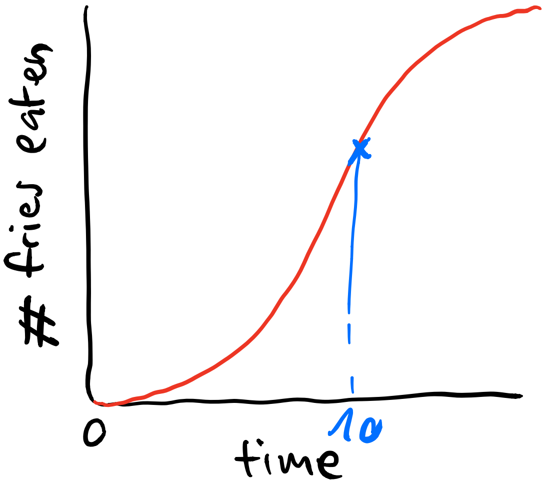


Derivative of a function

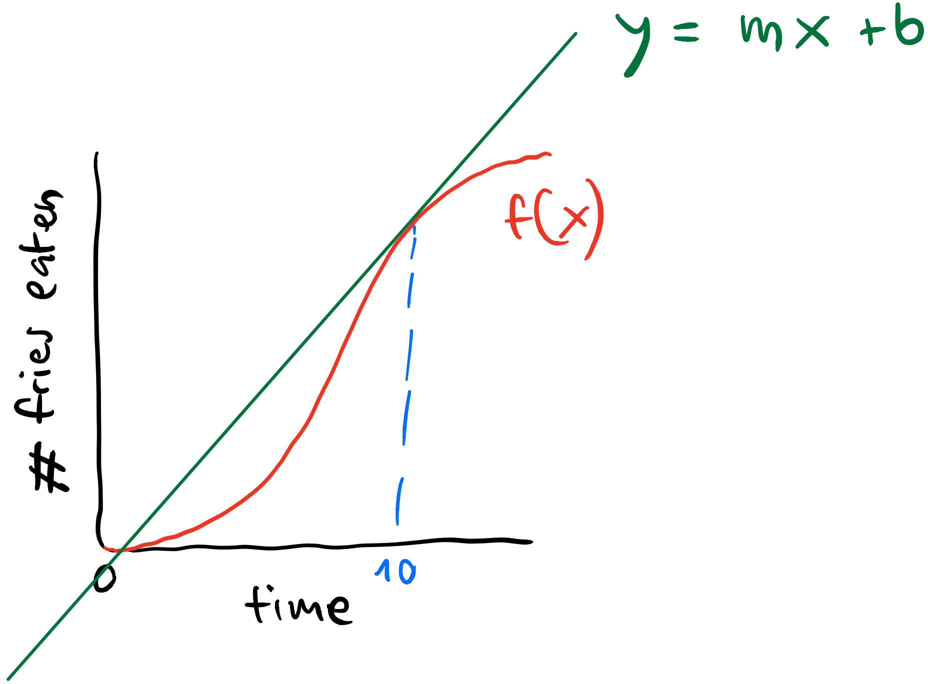
(= instantaneous)

- For a **very** small change in x, how does y change?

f(10) f(10.0001)



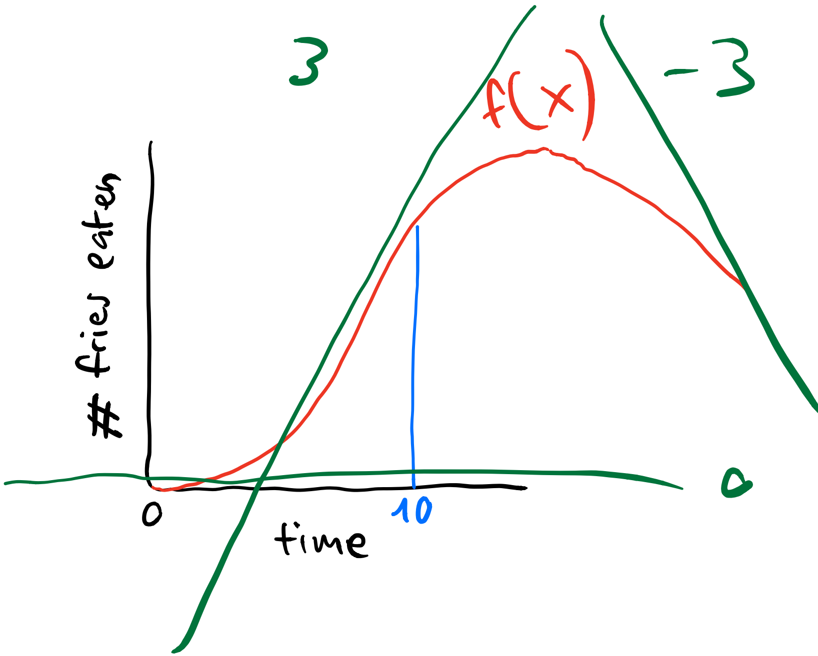
Derivative at a point as slope of the tangent line



1 2

Derivative at a point versus function that returns the derivative of another function

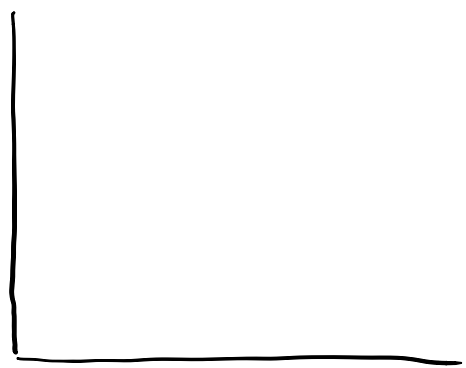
-2



1 2

2 $f'(x)$

$f'(10) = 2$



Intuition for how derivatives relate to machine learning

Training set

X   y

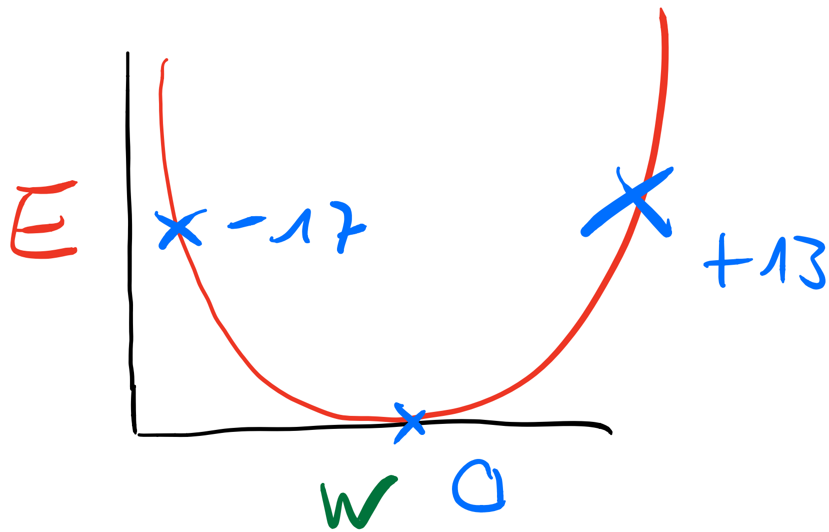
Estimator E

$E: X \longmapsto y$

$\hat{y} = E(x)$

e.g. $E(x) = w \cdot x$

Error Function



Intuition for how derivatives relate to machine learning

Overall Summary

- **linear algebra** defines important tensor objects and operations
- **numpy** implements all those objects and operations
- **derivatives** are about instantaneous rate of change and its direction

Recommendations for further reading / learning

- Khan Academy videos on linear algebra and single-variable differential calculus are **superb:** https://www.khanacademy.org/
- Matrix multiplication visualized by Eli Bendersky: https://eli.thegreenplace.net/2015/visualizing-matrix-multiplication-as-a-linear-combination/
- *Introduction to Linear Algebra*, Gilbert Strang.
- Numpy Tutorial by Justin Johnson for cs231n: http://cs231n.github.io/python-numpy-tutorial/#numpy

Next time

Termin	Thema
19.02.	Einführung; regelbasierte vs. datengetriebene Modelle
26.02.	Evaluation
05.03.	Trainingsdaten, Vor- und Nachverarbeitung
12.03.	N-Gramm-Sprachmodelle, statistische Maschinelle Übersetzung
19.03.	Grundlagen Lineare Algebra und Analysis, Numpy
26.03.	Lineare Modelle: lineare Regression, logistische Regression
02.04.	Neuronale Netzwerke: MLPs, Backpropagation, Gradient Descent
09.04.	Word Embeddings, Recurrent neural networks
16.04.	Tensorflow und Google Cloud Platform
30.04.	Encoder-Decoder-Modell
07.05.	Decoding-Strategien
14.05.	Attention-Mechanismus, bidirektionales Encoding, Byte Pair Encoding
21.05.	Maschinelle Übersetzung in der Praxis (Anwendungen)
28.05.	Zusammenfassung, Q&A Prüfung
Eventuell: Gastvortrag Prof. Artem Sokolov	
04.06., Raum TBA, 16:15 bis 18:00 Uhr	
Prüfung (schriftlich)	
18.06., AND-2-48, 16.15 bis 18:00 Uhr	

EVALUATION
TRAINING DATA
SMT

NMT

this is kinda important