PRACTICE PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE

TensorFlow: Deep learning with Keras

Primož Godec[1] and Rok Hribar[2]

[1]University of Ljubljana, Faculty of Computer and Information Science

[2]Jožef Stefan Institute, Computer Systems department

September 2020

Regarding the title of this course

TensorFlow: Deep learning with Keras

Regarding the title of this course

TensorFlow: Deep learning with Keras

- Deep learning is a set of methods for using artificial neural networks

Regarding the title of this course

TensorFlow: Deep learning with Keras

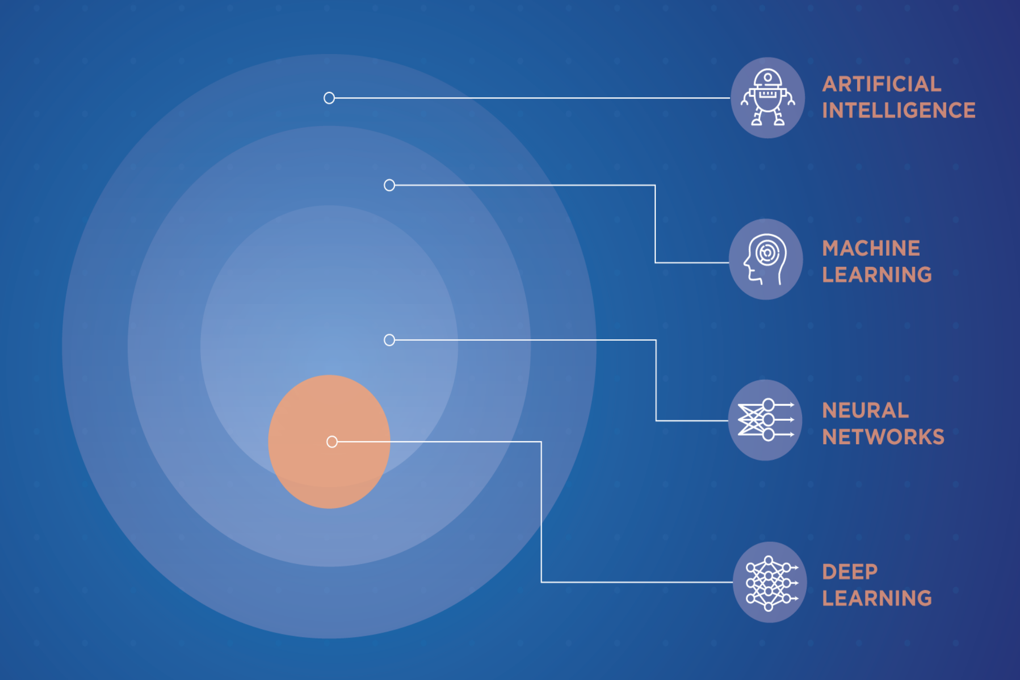
- Deep learning is a set of methods for using artificial neural networks
- Keras is probably the most popular library that implements Deep learning methods

Regarding the title of this course

TensorFlow: Deep learning with Keras

- Deep learning is a set of methods for using artificial neural networks
- Keras is probably the most popular library that implements Deep learning methods
- TensorFlow is a library that includes Keras as a submodule

Deep learning

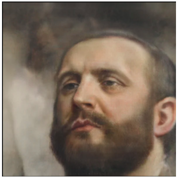


ARTIFICIAL INTELLIGENCE
MACHINE LEARNING
NEURAL NETWORKS
DEEP LEARNING

Artificial intelligence

- machines (or computers) that mimic cognitive functions that we associate with the human mind
  - translate text (like a book)
  - recognize object in image (face, handwriting)
  - recognize speech
  - creativity (poetry, music, paintings)
  - expert diagnosis (physician, mechanic)





What kind of murderer has moral fiber? – A cereal killer.

Artificial intelligence

- machines (or computers) that mimic cognitive functions that we associate with the human mind
  - translate text (like a book)
  - recognize object in image (face, handwriting)
  - recognize speech
  - creativity (poetry, music, paintings)
  - expert diagnosis (physician, mechanic)
- Tesler: AI is whatever hasn't been done yet
  - optical character recognition
  - playing chess

What kind of murderer has moral fiber? – A cereal killer.

Machine learning

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so.

Machine learning

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so.

Traditional algorithm:

- A human programmer designs an algorithm telling the machine how to execute all steps required to solve the problem at hand.

Machine learning

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so.

Traditional algorithm:

- A human programmer designs an algorithm telling the machine how to execute all steps required to solve the problem at hand.

For some tasks, it can be challenging for a human to manually create the needed algorithm.

Machine learning

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so.

Traditional algorithm:

- A human programmer designs an algorithm telling the machine how to execute all steps required to solve the problem at hand.

For some tasks, it can be challenging for a human to manually create the needed algorithm.

Machine learning algorithm:

- A human programmer designs an algorithm that helps the computer develop its own algorithm, rather than having human programmer specify every needed step.

Machine learning

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so.

Traditional algorithm:

- A human programmer designs an algorithm telling the machine how to execute all steps required to solve the problem at hand.

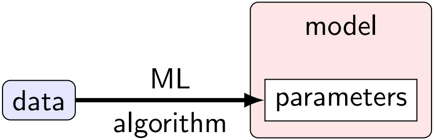
For some tasks, it can be challenging for a human to manually create the needed algorithm.

Machine learning algorithm:

- A human programmer designs an algorithm that helps the computer develop its own algorithm, rather than having human programmer specify every needed step.
- Do not let the word "learning" mislead you.

Machine learning example: Spam filtering

Text	Category
secret prize! claim secret prize now	spam
could you send me that image we talked about	ham
account compromised reset password	spam
free entry for 2 week tournament	spam
are you coming to a secret party for Mark	ham
you have a virus please download	spam
I'm in Ljubljana on Thursday, have time?	ham
$50 gift card for Amazon	spam



Basic machine learning algorithm:

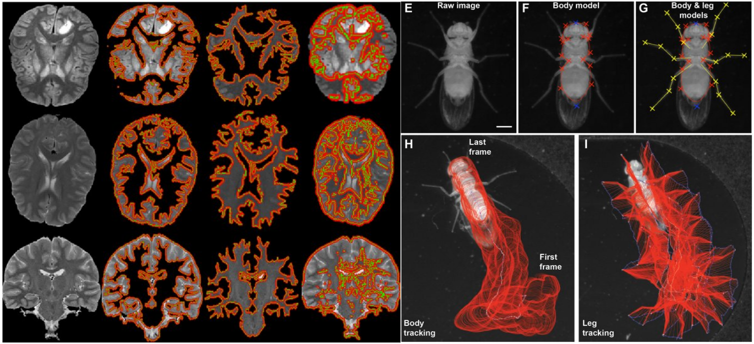
- Count the words that appear in spam/ham messages
- Calculate probabilities that a word is present in a message belonging to a given class

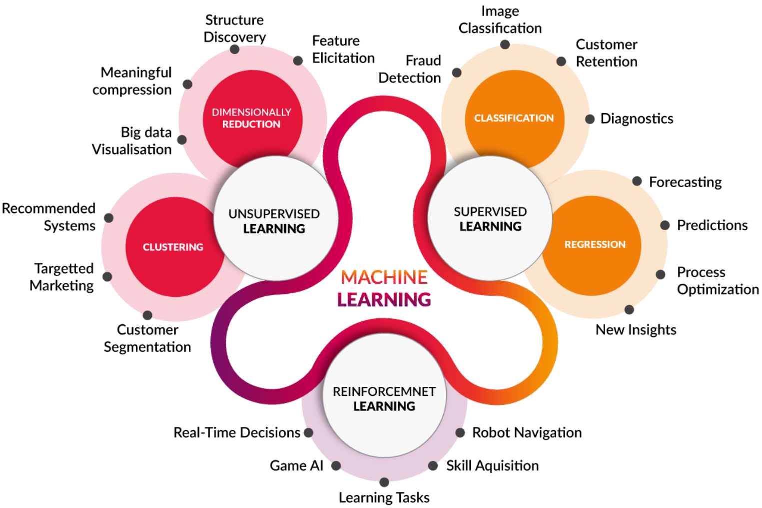
Result is a model that can calculate probability that a message is spam

Machine learning

Artificial intelligence that is not machine learning:

- rule-based systems (natural language processing, theorem proving)
- early computer vision




MACHINE
LEARNING
UNSUPERVISED
LEARNING
SUPERVISED
LEARNING
REINFORCEMNET
LEARNING
DIMENSIONALLY
REDUCTION
CLUSTERING
CLASSIFICATION
REGRESSION
Structure
Discovery
Feature
Elicitation
Meaningful
compression
Big data
Visualisation
Recommended
Systems
Targetted
Marketing
Customer
Segmentation
Image
Classification
Customer
Retention
Fraud
Detection
Diagnostics
Forecasting
Predictions
Process
Optimization
New Insights
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

Artificial neural network

Despite its name it doesn't have much to do with biological brain.

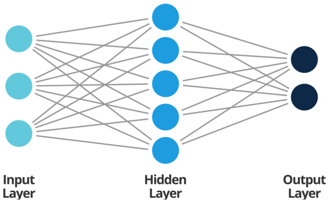
Artificial neural network

Despite its name it doesn't have much to do with biological brain.

It is a simple mathematical model that:

- is fast – can be easily parallelized

Traditional neural network



- can capture wide range of functions

$$a = W_1 x + b_1$$

$$h = \sigma(a)$$

$$y = W_2 h + b_2$$

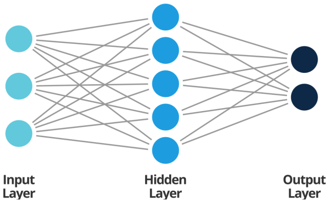
Artificial neural network

Despite its name it doesn't have much to do with biological brain.

It is a simple mathematical model that:

- is fast – can be easily parallelized
  - matrix multiplication is highly parallelizable and optimized
- can capture wide range of functions

Traditional neural network



$$a = W_1 x + b_1$$
$$h = \sigma(a)$$
$$y = W_2 h + b_2$$

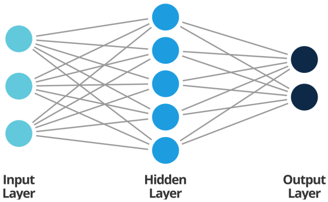
Artificial neural network

Despite its name it doesn't have much to do with biological brain.

It is a simple mathematical model that:

- is fast – can be easily parallelized
  - matrix multiplication is highly parallelizable and optimized
  - composition of linear functions is linear – we need nonlinearity
- can capture wide range of functions

Traditional neural network



$$a = W_1 x + b_1$$
$$h = \sigma(a)$$
$$y = W_2 h + b_2$$

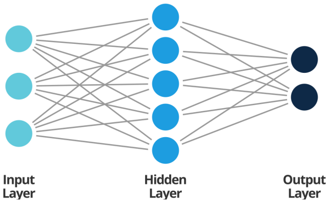
Artificial neural network

Despite its name it doesn't have much to do with biological brain.

It is a simple mathematical model that:

- is fast – can be easily parallelized
  - matrix multiplication is highly parallelizable and optimized
  - composition of linear functions is linear – we need nonlinearity
  - the fastest nonlinear functions are those of a single variable
- can capture wide range of functions

Traditional neural network



$$a = W_1 x + b_1$$

$$h = \sigma(a)$$

$$y = W_2 h + b_2$$

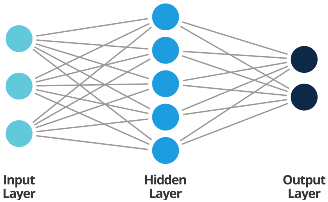
Artificial neural network

Despite its name it doesn't have much to do with biological brain.

It is a simple mathematical model that:

- is fast – can be easily parallelized
  - matrix multiplication is highly parallelizable and optimized
  - composition of linear functions is linear – we need nonlinearity
  - the fastest nonlinear functions are those of a single variable
- can capture wide range of functions
  - L-NL and NL-L are not universal approximators

Traditional neural network



$$a = W_1 x + b_1$$

$$h = \sigma(a)$$

$$y = W_2 h + b_2$$

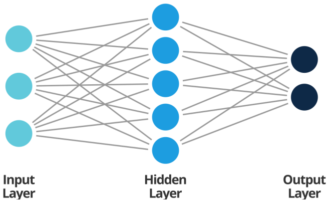
Artificial neural network

Despite its name it doesn't have much to do with biological brain.

It is a simple mathematical model that:

- is fast – can be easily parallelized
  - matrix multiplication is highly parallelizable and optimized
  - composition of linear functions is linear – we need nonlinearity
  - the fastest nonlinear functions are those of a single variable
- can capture wide range of functions
  - L-NL and NL-L are not universal approximators
  - NL-L-NL and L-NL-L are and out of those L-NL-L is faster

Traditional neural network



$$a = W_1 x + b_1$$

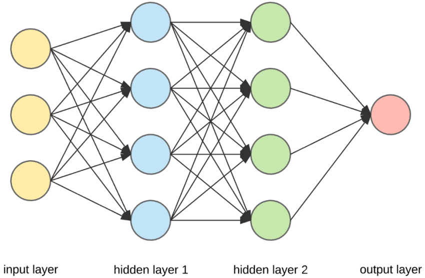
$$h = \sigma(a)$$

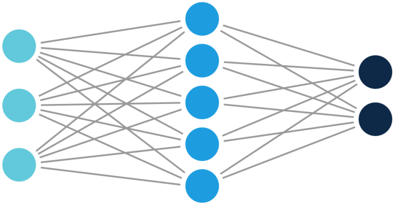
$$y = W_2 h + b_2$$

Deep neural network

The number of all possible models for a network with a single hidden layer is

$$\frac{a^{\#\text{parameters}}}{\#\text{hidden units!}}$$





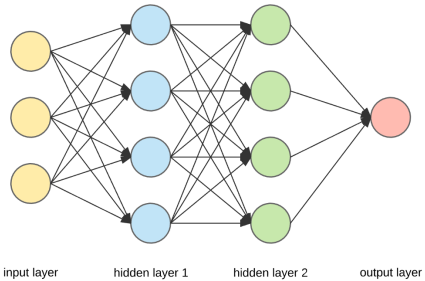
Deep neural network

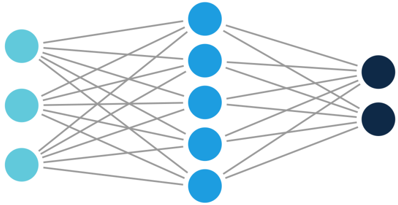
The number of all possible models for a network with a single hidden layer is

$$\frac{a^{\#\text{parameters}}}{\#\text{hidden units!}}$$

More formal result for capacity of a deep network (per parameter)

$$\frac{w^{f-2}}{d}(w/f)^{(d-1)f}$$



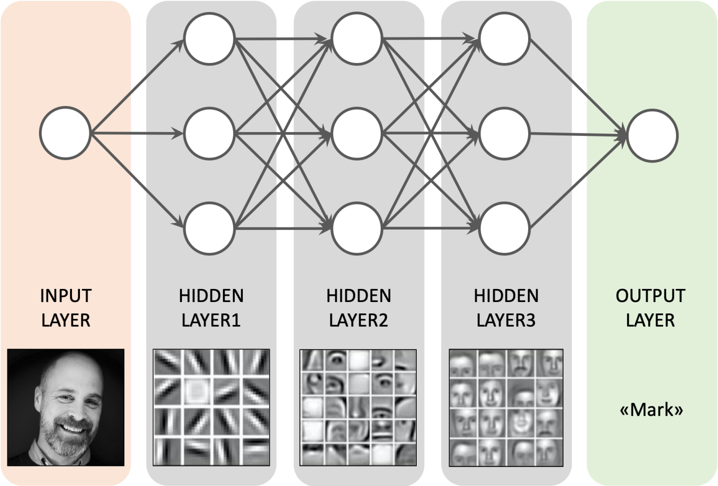


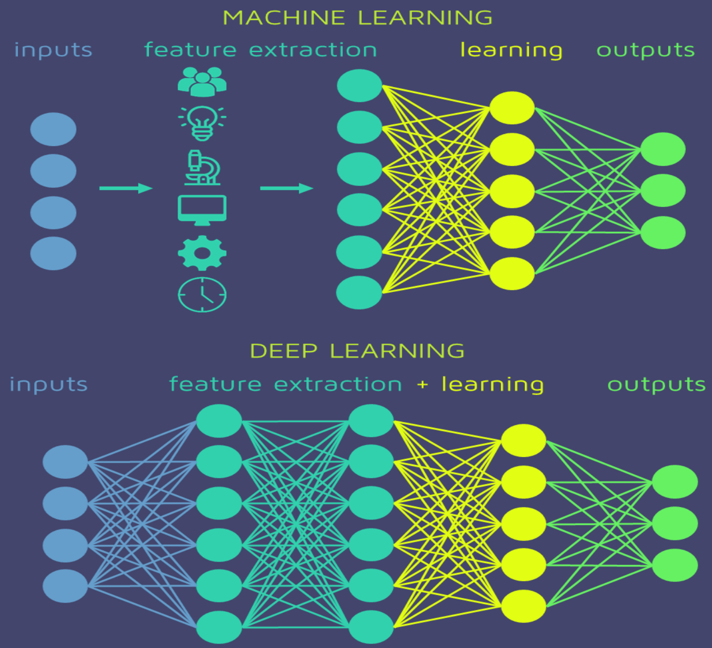
Deep neural network

More philosophical reasons for why depth is good:

- belief that the function we want to learn is a computer program consisting of multiple steps, where each step makes use of the previous step's output
- belief that the nature of knowledge is hierarchical, where more abstract concepts build on simpler ones
- belief that the learning problem consists of discovering a set of underlying factors of variation that can in turn be described in terms of other, simpler underlying factors of variation

Deep neural network




MACHINE LEARNING
inputs
feature extraction
learning
outputs
DEEP LEARNING
inputs
feature extraction + learning
outputs

Training a neural network

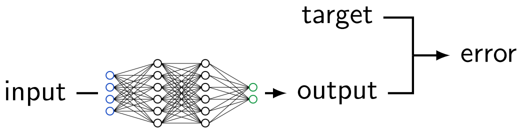
There have been many procedures to train neural networks through history.

- learning rules (Hebian, correlation)

Training a neural network

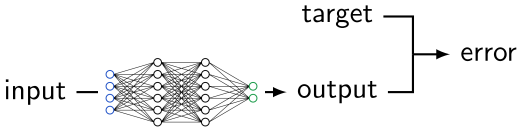
There have been many procedures to train neural networks through history.

- learning rules (Hebian, correlation)
- perceptron learning (linear least squares)
- neuroevolution
- gradient based methods

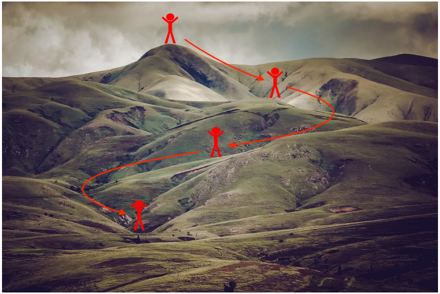


Training a neural network

- gradient based methods



- Derivative of error with respect to all parameters of the network are calculated using backpropagation algorithm.
- Parameters of the network are changed in direction that minimizes the error.

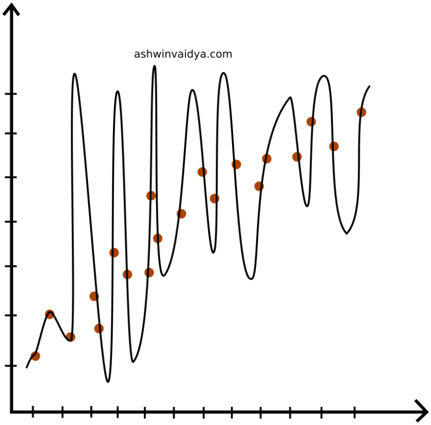


Overfitting and underfitting

Overfitting is a modeling error that occurs when a model has learned too much.

- model capacity is so high that noise is being modeled
- model doesn't generalize well from our training data to unseen data
- this can usually be avoided by

$$\#\text{data instances} \gg \#\text{parameters}$$



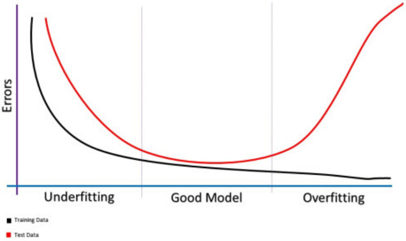
Overfitting and underfitting

However, overfitting is a complicated phenomenon.

- model capacity
- data set distribution
- complexity of an underlying problem

The most bulletproof way to know if overfitting happened is to measure error on unseen data

- Test error



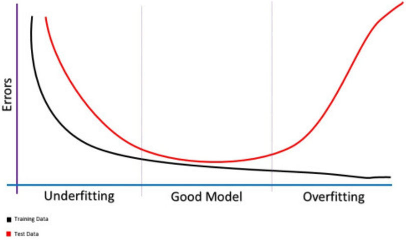
Overfitting and underfitting

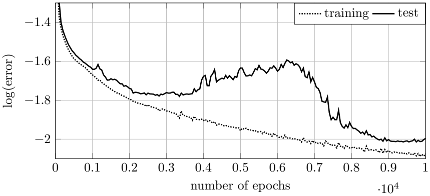
However, overfitting is a complicated phenomenon.

- model capacity
- data set distribution
- complexity of an underlying problem

The most bulletproof way to know if overfitting happened is to measure error on unseen data

- Test error





Regularization

AI problems normally require high capacity models.

- depth due to problem complexity
- width to ensure information flow

Regularization

AI problems normally require high capacity models.

- depth due to problem complexity
- width to ensure information flow

To reduce overfitting we handicap the network without reducing its size.

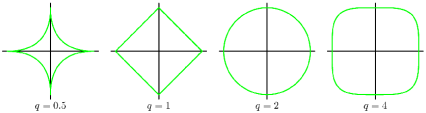
- constraints on the structure of the network
- disruptions in the training phase

Techniques:

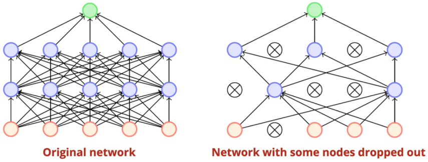
- weight decay
- parameter sharing
- semi-supervised learning
- dropout
- early stopping
- sparse representations
- data augmentation
- batch/layer normalization

Weight decay

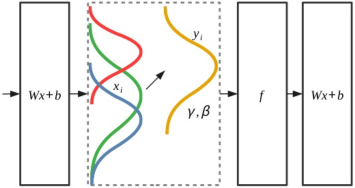
$$\text{error} + \lambda \|\text{parameters}\|_q$$



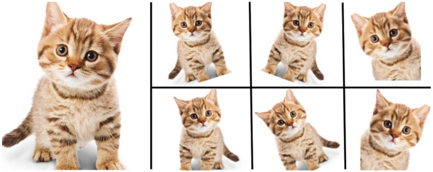
Dropout

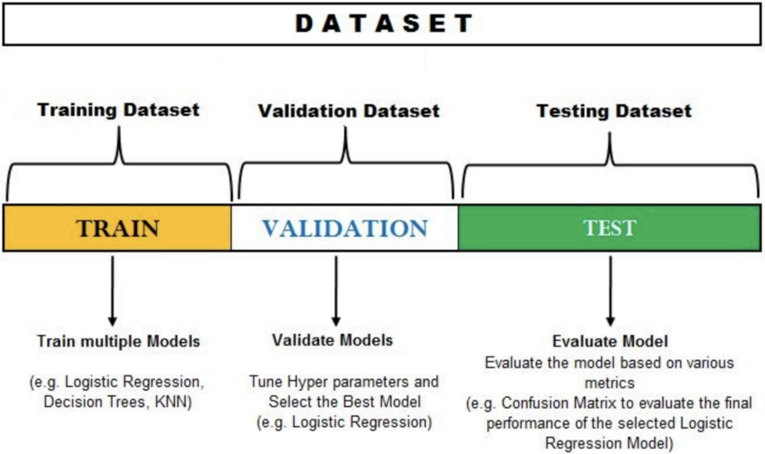


Batch normalization

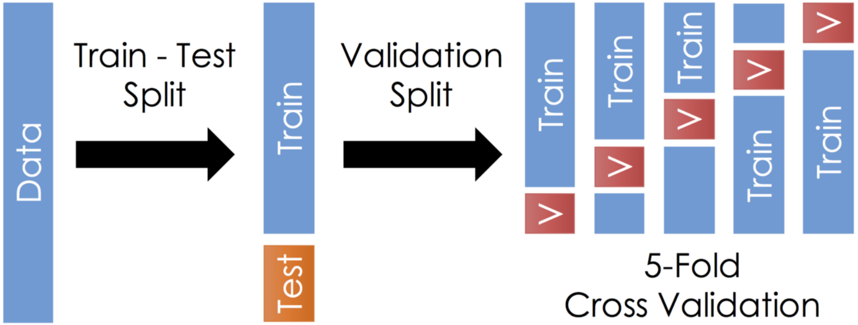


Data augmentation




DATASET
Training Dataset
Validation Dataset
Testing Dataset
TRAIN
VALIDATION
TEST
Train multiple Models
(e.g. Logistic Regression, Decision Trees, KNN)
Validate Models
Tune Hyper parameters and Select the Best Model (e.g. Logistic Regression)
Evaluate Model
Evaluate the model based on various metrics
(e.g. Confusion Matrix to evaluate the final performance of the selected Logistic Regression Model)

Cross-validation method



Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

- gradient based training (on GPU)

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

- gradient based training (on GPU)
- availability of large quantity of data

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

- gradient based training (on GPU)
- availability of large quantity of data
- appropriate cost functions

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

- gradient based training (on GPU)
- availability of large quantity of data
- appropriate cost functions
- new regularizations

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

- gradient based training (on GPU)
- availability of large quantity of data
- appropriate cost functions
- new regularizations
- new representation mappings (eg. embeddings)

Deep learning

Official definition: Deep learning is the study of machine learning models composed of multiple layers of functions that progressively extract higher level features from the raw input.

However, this idea existed also 1950–2010 when success of deep learning was very limited.

- gradient based training (on GPU)
- availability of large quantity of data
- appropriate cost functions
- new regularizations
- new representation mappings (eg. embeddings)
- new network architectures

Keras

Neural networks with Keras

- Introduction to neural networks through classification
- Neural network for regression
- Image classification

Convolutional neural networks

- Image classification with convolutional neural networks
- Exercise: Classification of images from CIFAR10 dataset

Lake Cerknica is the largest lake in Slovenia ($\sim$ 28km^2).

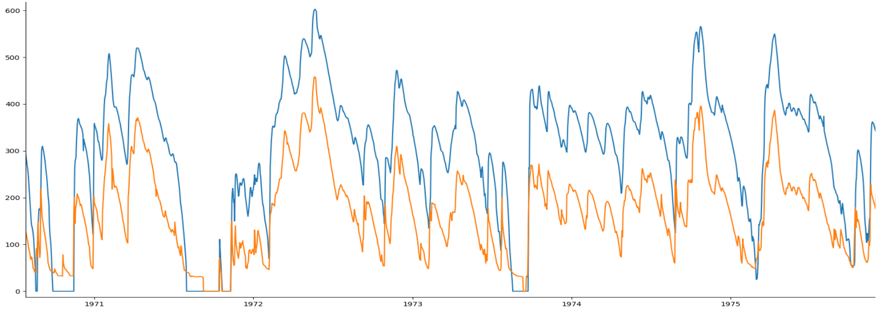
Lake Cerknica is the largest lake in Slovenia ($\sim 28\text{km}^2$).

When it is full – intermittent lake.

Lake Cerknica is the largest lake in Slovenia ($\sim 28\text{km}^2$).

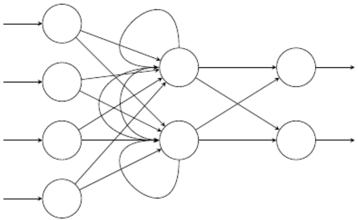
When it is full – intermittent lake.

Karst country involves an underground drainage system with sinkholes and caves.



Recurrent neural network

- part of output from a layer is fed as additional input along with the next instance
- short term memory



Recurrent neural networks

Internal state doesn't depend only on current data instance but also on all previous ones.

Advantages

- no need to choose time window
- weight sharing
- partially observable modeling

Disadvantages

- less parallelizable
- difficult to train
- vanishing and exploding gradients

Applications:

- Time series prediction
- Robot control
- Text generation
- Music composition
- Video processing
- Machine translation
- Handwriting recognition
- Genetics and protein related ML
- Speech recognition

TensorFlow

What can TensorFlow do?

1. It can perform numerical operations on data (in a parallel way – multi-core, GPU).

```
import tensorflow as tf

A = tf.Variable( [[1.0, 2.0], [3.0, 4.0]] )
B = tf.Variable( [[5.0, 6.0], [7.0, 8.0]] )

C = tf.matmul(A, B)         # matrix multiplication
D = A - B*C                 # elementwise operations
cos_D = tf.cos(D)           # elementwise math functions
sum_D = tf.reduce_sum(D)    # sum of all D components
max_D = tf.reduce_max(D)    # max component of D
svd_D = tf.linalg.svd(D)    # singular value decomposition
```

```
C = tf.matmul(A, B)
# <tf.Tensor: id=17, shape=(2, 2), dtype=float32,
#  numpy=array([[19., 22.], [43., 50.]], dtype=float32)>

cos_D = tf.cos(D)
# <tf.Tensor: id=30, shape=(2, 2), dtype=float32,
#  numpy=array([[ 0.96945935, -0.36729133],
#              [-0.89988   ,  0.9873345 ]],
#             dtype=float32)>

max_D = tf.reduce_max(D)
# <tf.Tensor: id=26, shape=(), dtype=float32,
#  numpy=-94.0>

C.numpy()
# array([[19., 22.], [43., 50.]], dtype=float32)
```

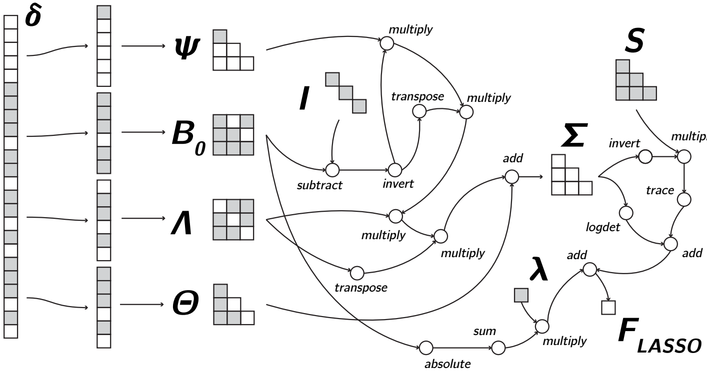
```
svd_D = tf.linalg.svd(D)
# (<tf.Tensor: id=27, shape=(2,), dtype=float32,
#   numpy=array([520.9103, 2.9102921], dtype=float32)>,
#
#  <tf.Tensor: id=28, shape=(2, 2), dtype=float32,
#   numpy=array([[-0.30792360,  0.95141107],
#                [-0.95141107, -0.30792360]],
#               dtype=float32)>,
#
#  <tf.Tensor: id=29, shape=(2, 2), dtype=float32,
#   numpy=array([[0.59984480,  0.80011636],
#                [0.80011636, -0.59984480]],
#               dtype=float32)>)
```

What can TensorFlow do?

1. It can perform numerical operations on data (in a parallel way – multi-core, GPU).

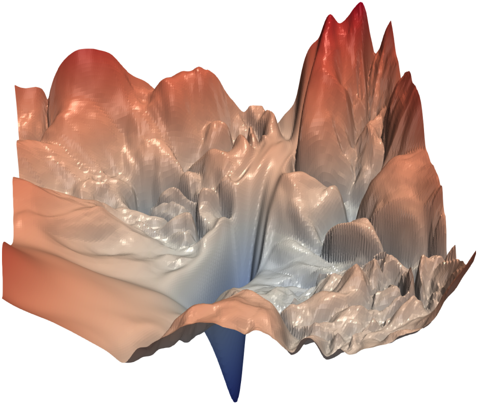
What can TensorFlow do?

1. It can perform numerical operations on data (in a parallel way – multi-core, GPU).
2. It can calculate derivatives using automatic differentiation.



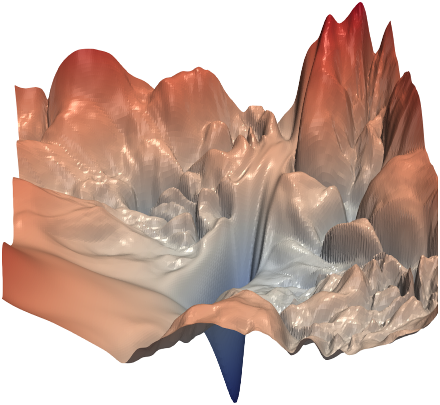
Why do we need derivatives?

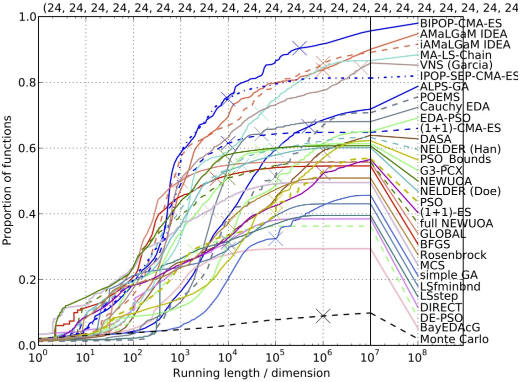
Knowing in which direction “down” is, can help us when solving optimization problems.



Why do we need derivatives?

Knowing in which direction "down" is, can help us when solving optimization problems.





- So, various competitions show that evolutionary algorithms outperform gradient based optimization algorithms.
- **However**, all those competitions use functions of “low” dimension (≤ 100) and gradient based optimization excels in high dimensions.

- So, various competitions show that evolutionary algorithms outperform gradient based optimization algorithms.
- **However**, all those competitions use functions of “low” dimension (≤ 100) and gradient based optimization excels in high dimensions.

How many local minima are there with respect to dimension?

- So, various competitions show that evolutionary algorithms outperform gradient based optimization algorithms.
- **However**, all those competitions use functions of "low" dimension (≤ 100) and gradient based optimization excels in high dimensions.

How many local minima are there with respect to dimension?

$$
\begin{bmatrix}
\dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2.2ex]
\dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2.2ex]
\vdots & \vdots & \ddots & \vdots \\[2.2ex]
\dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2}
\end{bmatrix}
$$

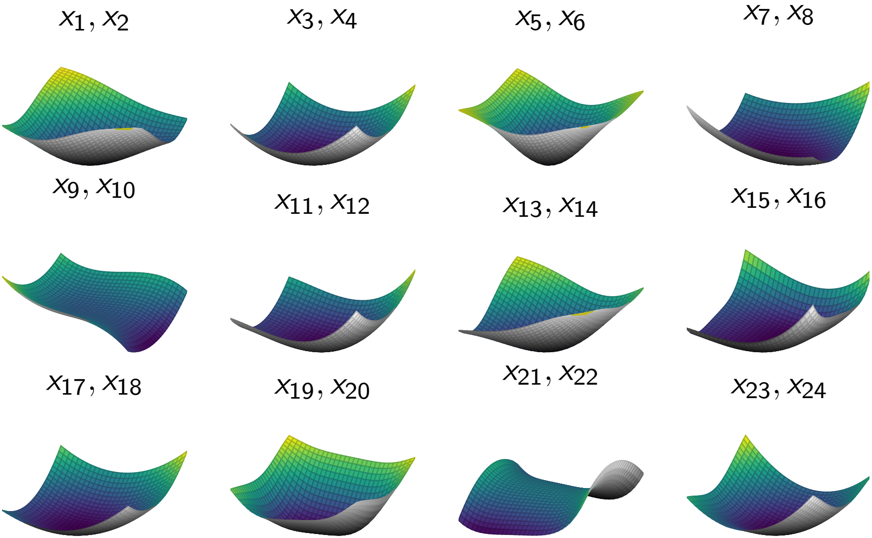
- So, various competitions show that evolutionary algorithms outperform gradient based optimization algorithms.
- **However**, all those competitions use functions of "low" dimension (≤ 100) and gradient based optimization excels in high dimensions.

How many local minima are there with respect to dimension?

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \, \partial x_1} & \frac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

If eigenvalues of Hessian matrix are randomly distributed, then probability that a stationary point is a local minimum is $\mathbf{2^{-n}}$.

- Saddle points are exponentially more common than local minima.


x_1, x_2
x_3, x_4
x_5, x_6
x_7, x_8
x_9, x_{10}
x_{11}, x_{12}
x_{13}, x_{14}
x_{15}, x_{16}
x_{17}, x_{18}
x_{19}, x_{20}
x_{21}, x_{22}
x_{23}, x_{24}

Ways to calculate derivatives of a program

1. Numerical differentiation

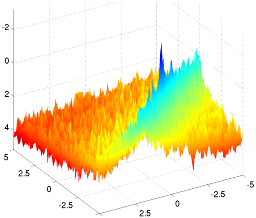
$$\frac{\partial}{\partial x_1} f(x_1, x_2, \dots) \approx \frac{f(x_1 + h, x_2, \dots) - f(x_1 - h, x_2, \dots)}{2h}$$

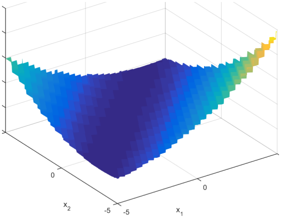
Very efficient for

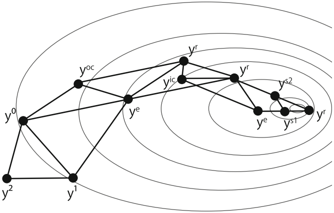
- Noisy functions
- Locally flat functions

Algorithms that use it

- Nelder-Mead algorithm
- OpenAI evolution strategy





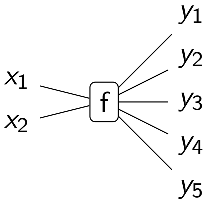


Ways to calculate derivatives of a program

2. Symbolic differentiation

$$\frac{\partial}{\partial x} \log\left(1 + \exp\left(ax + b\right)\right) = \frac{a \exp\left(ax + b\right)}{1 + \exp\left(ax + b\right)}$$

Very efficient in case function has large number of outputs

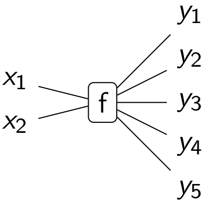


Ways to calculate derivatives of a program

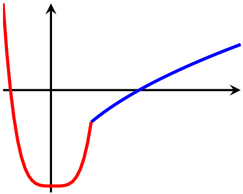
2. Symbolic differentiation

$$\frac{\partial}{\partial x} \log\left(1 + \exp\left(ax + b\right)\right) = \frac{a \exp\left(ax + b\right)}{1 + \exp\left(ax + b\right)}$$

Very efficient in case function has large number of outputs



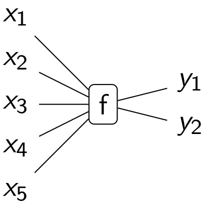

```
if f(x, data) > 0:
    g(x, data)
else:
    h(x, data)
```



Ways to calculate derivatives of a program

3. Automatic differentiation
   - Sort of a hybrid between simbolic and numerical differentiation
   - There exist forward and reverse automatic differentiation – `TensorFlow` uses reverse automatic differentiation

Very efficient in case function has large number of inputs

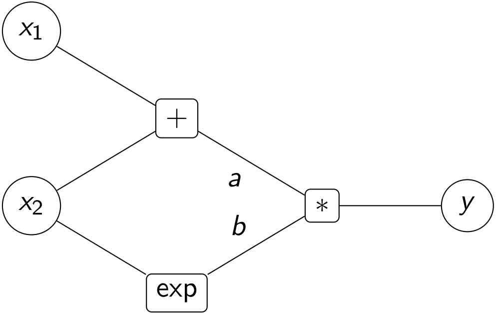


Example:

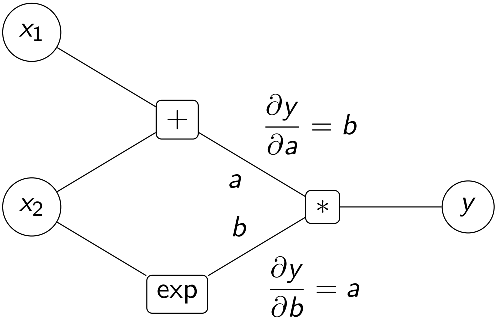
$$f(x_1, x_2, \ldots, x_n) = x_1 \cdot x_2 \cdot \ldots \cdot x_n$$

$$\nabla f = \begin{bmatrix} x_2 \cdot x_3 \cdot \ldots \cdot x_n \\ x_1 \cdot x_3 \cdot \ldots \cdot x_n \\ \vdots \\ x_1 \cdot x_2 \cdot \ldots \cdot x_{n-1} \end{bmatrix}$$

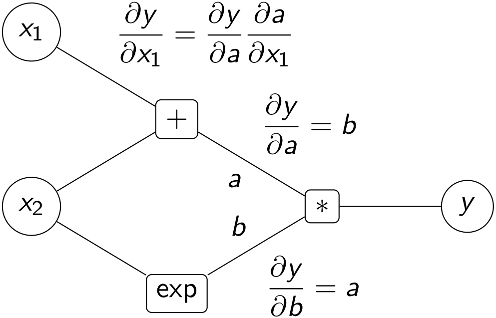
$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$



$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$



$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$

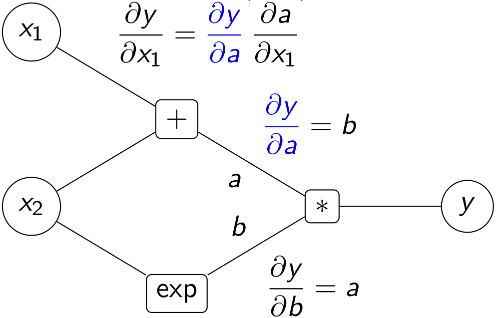


$$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial x_1}$$

$$\frac{\partial y}{\partial a} = b$$

$$\frac{\partial y}{\partial b} = a$$

$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$

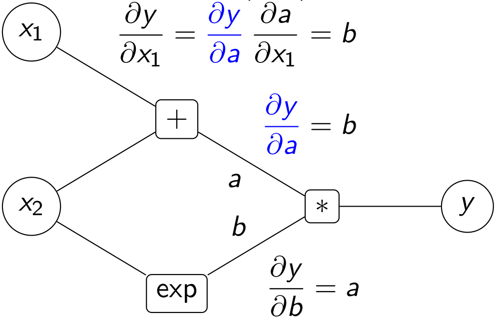


$$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial a}\overbrace{\frac{\partial a}{\partial x_1}}^{1}$$

$$\frac{\partial y}{\partial a} = b$$

$$\frac{\partial y}{\partial b} = a$$

$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$



$$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial a} \overbrace{\frac{\partial a}{\partial x_1}}^{1} = b$$

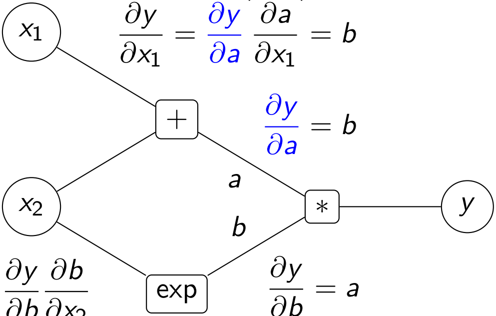
$$\frac{\partial y}{\partial a} = b$$

$$\frac{\partial y}{\partial b} = a$$

$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$

$$\frac{\partial y}{\partial x_1} = \color{blue}{\frac{\partial y}{\partial a}}\,\overbrace{\frac{\partial a}{\partial x_1}}^{1} = b$$

$$\color{blue}{\frac{\partial y}{\partial a}} = b$$

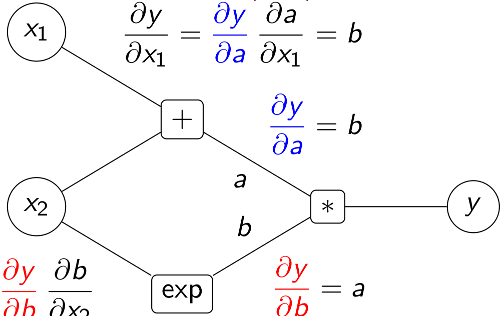


$$\frac{\partial y}{\partial b} = a$$

$$\frac{\partial y}{\partial x_2} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial x_2} + \frac{\partial y}{\partial b}\frac{\partial b}{\partial x_2}$$

$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$

$$\frac{\partial y}{\partial x_1} = {\color{blue}\frac{\partial y}{\partial a}} \overbrace{\frac{\partial a}{\partial x_1}}^{1} = b$$



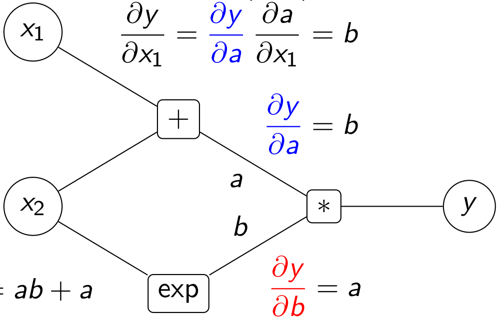
$${\color{blue}\frac{\partial y}{\partial a}} = b$$

$${\color{red}\frac{\partial y}{\partial b}} = a$$

$$\frac{\partial y}{\partial x_2} = {\color{blue}\frac{\partial y}{\partial a}} \underbrace{\frac{\partial a}{\partial x_2}}_{a} + {\color{red}\frac{\partial y}{\partial b}} \underbrace{\frac{\partial b}{\partial x_2}}_{1}$$

$$\boxed{y = (x_1 + x_2)\exp(x_2)}$$

$$\frac{\partial y}{\partial x_1} = {\color{blue}\frac{\partial y}{\partial a}} \overbrace{\frac{\partial a}{\partial x_1}}^{1} = b$$



$${\color{blue}\frac{\partial y}{\partial a}} = b$$

$${\color{red}\frac{\partial y}{\partial b}} = a$$

$$\frac{\partial y}{\partial x_2} = {\color{blue}\frac{\partial y}{\partial a}} \underbrace{\frac{\partial a}{\partial x_2}}_{a} + {\color{red}\frac{\partial y}{\partial b}} \underbrace{\frac{\partial b}{\partial x_2}}_{1} = ab + a$$

The same thing in TensorFlow

```
import tensorflow as tf

x1 = tf.Variable(3.1)
x2 = tf.Variable(-1.4)

with tf.GradientTape() as tape: # Save graph to tape
    tape.watch([x1, x2])        # Watch for x1 and x2
    f = (x1+x2)*tf.exp(x2)      # Calculate f(x1, x2)

df = tape.gradient(f, [x1, x2])
# [<tf.Tensor: id=22, shape=(), dtype=float32, numpy
    =0.24659698>,
#  <tf.Tensor: id=25, shape=(), dtype=float32, numpy
    =0.66581184>]
```

Optimizers

Vanilla update

```
x += - learning_rate * dx
```

Momentum update

```
v = mu * v - learning_rate * dx # integrate velocity
x += v # integrate position
```

Adam

```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

Optimizers

Optimizers are available in `tf.keras.optimizers` module

- Vanilla update (`tf.keras.optimizers.SGD`)
- Adagrad (`tf.keras.optimizers.Adagrad`)
- RMSprop (`tf.keras.optimizers.RMSprop`)
- Adam (`tf.keras.optimizers.Adam`)

Matrix factorization example

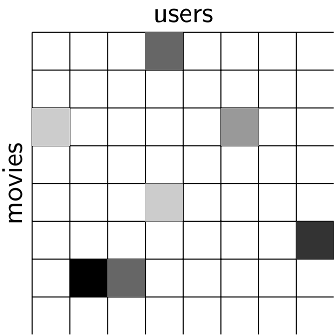
Suppose we have movie ratings from various people for set of movies they have watched.

user id	movie id	rating
4160	14501	5
182	14502	2
6649	14502	3
17240	14502	1
115	14503	4
⋮	⋮	⋮

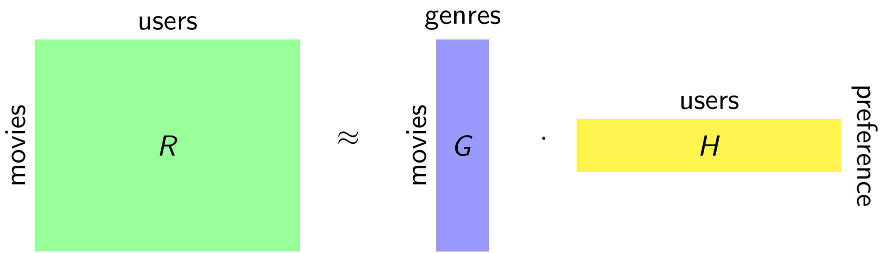
Matrix factorization example

Suppose we have movie ratings from various people for set of movies they have watched.

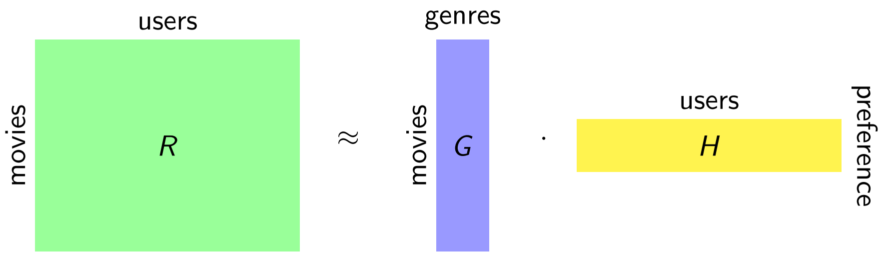
user id	movie id	rating
4160	14501	5
182	14502	2
6649	14502	3
17240	14502	1
115	14503	4
⋮	⋮	⋮



Matrix factorization example



Matrix factorization example



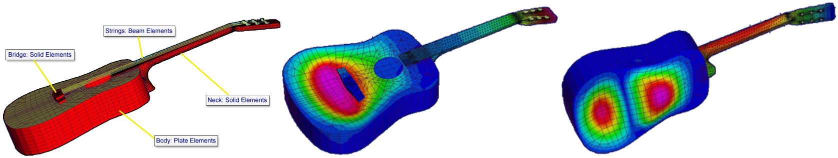
$$\text{error} = \|W \odot (R - G \cdot H)\| = \min. \qquad G, H \geq 0$$

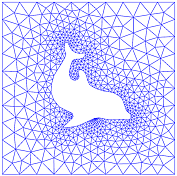
```python
# Initialize variables.
G = tf.Variable(...)
H = tf.Variable(...)

# Choose a gradient based optimizer.
optimizer = tf.keras.optimizers.Adam()

# Perform gradient descent.
for i in range(num_steps):
    with tf.GradientTape() as tape:
        tape.watch([G, H])
        absG = tf.abs(G)
        absH = tf.abs(H)
        dR = R - tf.matmul(absG, absH)
        loss = tf.reduce_sum(tf.square(dR))
    dG, dH = tape.gradient(loss, [G, H])
    optimizer.apply_gradients([[dG, G], [dH, H]])
```

Example: Finite element method





$\xrightarrow[\text{stuff}]{\text{geometric}}$ $\boxed{Ku = \lambda Mu}$ $\xrightarrow{\text{solve}}$ $\boxed{\text{resonance spectra}}$