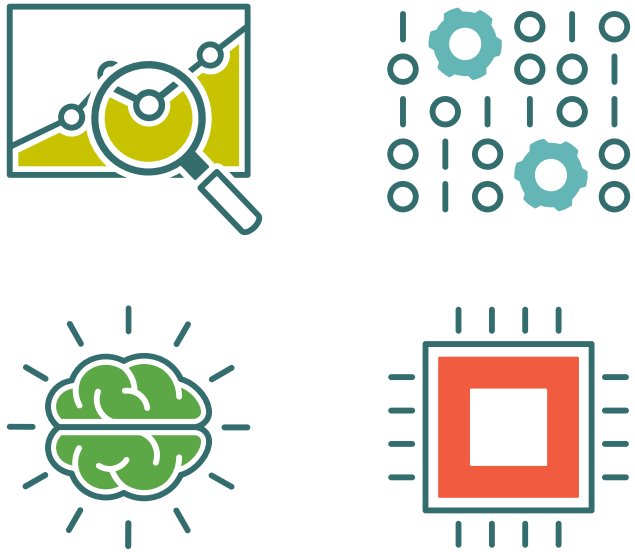
Making the World More Efficient Using AI & Machine Learning



Krzysztof Kowalczyk
Visiting Data Scientist at BCG GAMMA
President of Machine Learning Society at MIM UW

CAMP IT, MILÓWKA, 7/09/2019

Camp IT

Difference between machine learning and AI:
If it is written in Python, it's probably machine learning
If it is written in PowerPoint, it's probably AI

Mat Velloso (@matvelloso on Twitter, 17:25 - 22 Nov 2018)

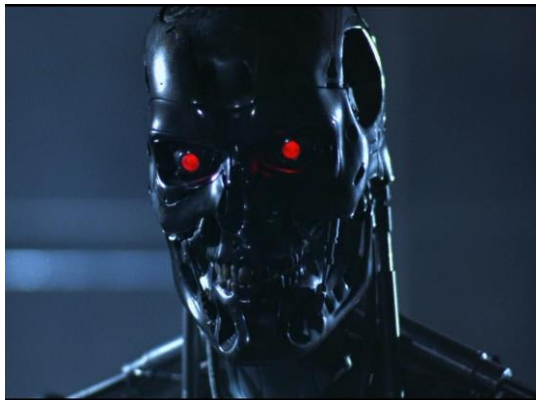
There is no clear definition of what AI means

It is still important to know some of the popular definitions, especially when it comes to an often misused term such as "AI".

Weak (narrow) AI

Can apply intelligence to one specific task, solving it at least as good as a human being

It's already used everywhere



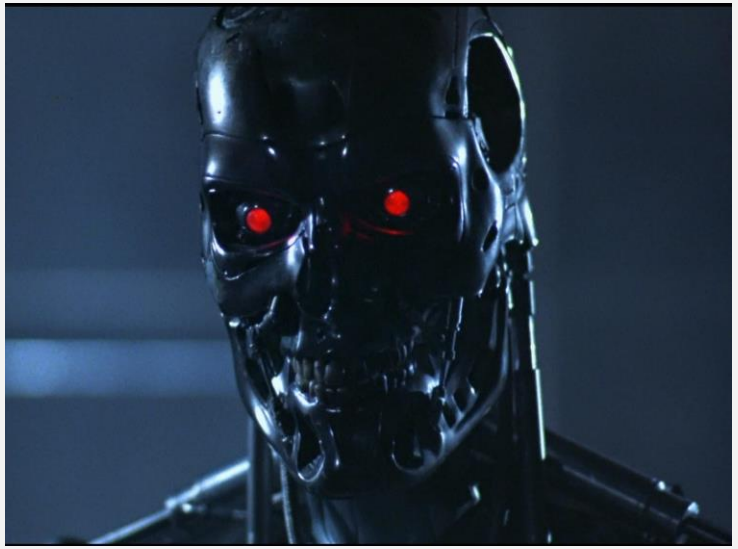
Strong (general) AI

Has a capacity to understand or learn any intellectual task that a human being can

It's debatable whether it can exist

Computers learn to make predictions from historical data

Data → Model → Prediction



Example: house price can be predicted using weights tuned to match historical data

$$\textit{house price} = w_1 \cdot \textit{area} + w_2 \cdot \textit{district} + w_3 \cdot \textit{price per } m^2 \textit{ in city} + \ldots$$

Data: area, district, price per m^2 in city, …

Model: w_1, w_2, w_3, …

Prediction: house price

Human work is still necessary to make machine learning models work

Data

What are the historical values of area, district, average price associated with house prices?

Comes from historical observations, usually needs to be cleaned by a human

Model

What model do we choose?
How to come up with an accurate model?
How to measure model quality?

Typically it's a human job to design or select an appropriate model and quality metrics

Parameters

What should values of w_1, w_2, w_3, … be equal to?

This part is usually done by computers, based on historical data and specified model

Prediction

What price should we expect for a house with a given area, district, average city price?

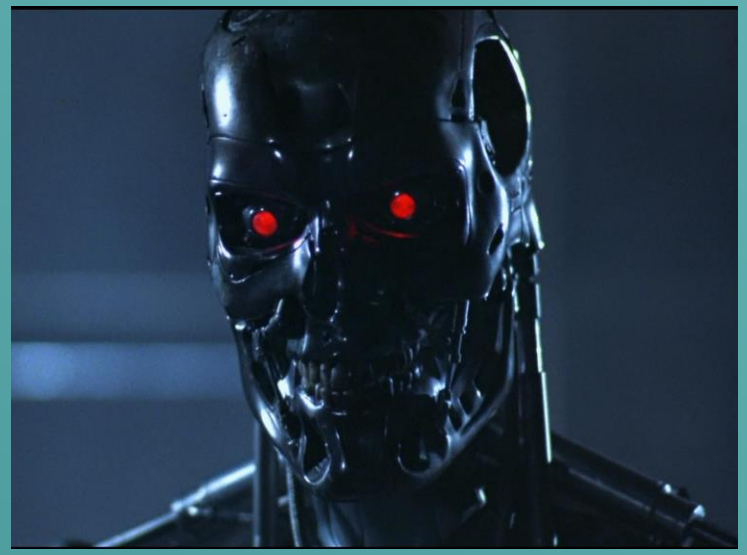
Correct model and parameters enable automatic prediction of price for new houses

$$\textit{house price} = w_1 \cdot \textit{area} + w_2 \cdot \textit{district} + w_3 \cdot \textit{price per } m^2 \textit{ in city} + \ldots$$

Data science vs machine learning

Data science is a human process of analyzing and cleaning historical data, in order to design a model that can be *trained* to predict future values of some variable based on new data

Machine learning is an automated process of finding optimal model parameters *(training)*, based on historical data and an evaluation method specified by a human Data Scientist

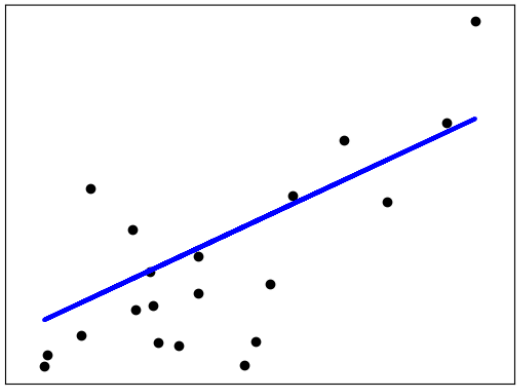


Machine learning is so simple!

Machine learning tasks are easy to understand

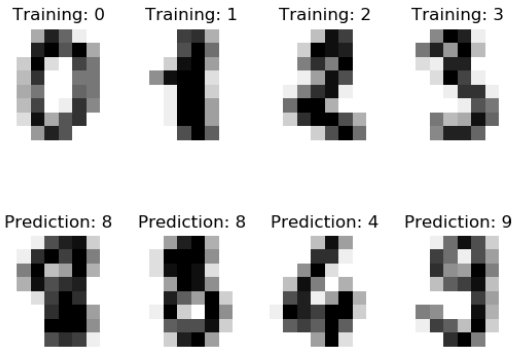
Regression

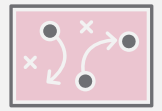
Predicting value of a continuous variable (house price, air temperature, airplane delay)



Classification

Predicting value of a categorical variable (plant species, illness, handwritten characters)





Clustering

Separating data into groups (types of customers, similar parts of image, spam e-mails)

Linear regression – the simplest machine learning model

Assume target variable is a weighted mean of features:

$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n$$

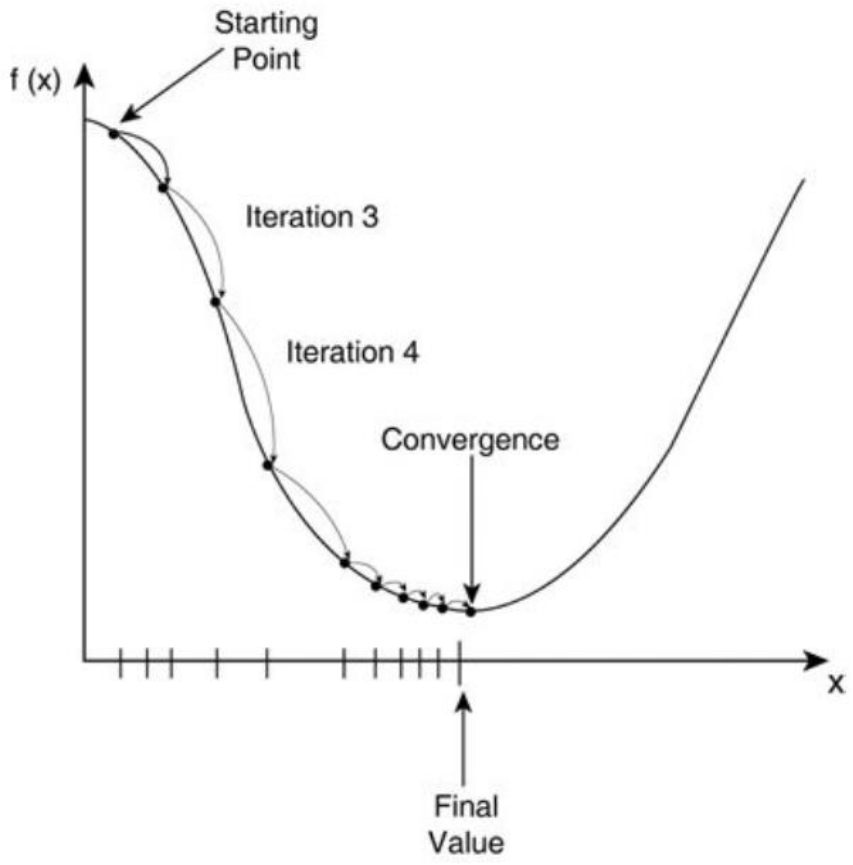
Given historical values of *xs* and *ys*,
find weights (*ws*) that minimize mean squared error:

$$MSE = mean_{over\ all\ historical\ samples}([y_{true} - y_{predicted}]^2)$$

where for each historical sample:

$$y_{predicted} = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n$$

Learning algorithm 1: gradient descent



Idea

- change the weights in direction that will decrease the error
- repeat until local minimum is reached

How to know whether to increase or decrease a weight?

Calculate the derivative of function:

$$J(w) = MSE(prediction(w), y)$$

where:

$$prediction(w) = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n$$

note that $X=(x_1, x_2, \ldots, x_n)$ and y are just constants from historical data:

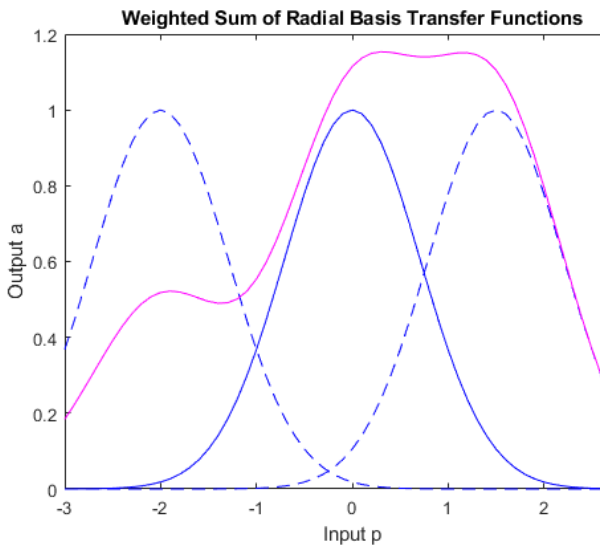
- matrix consisting of vectors of features' values
- vector of values for target variables

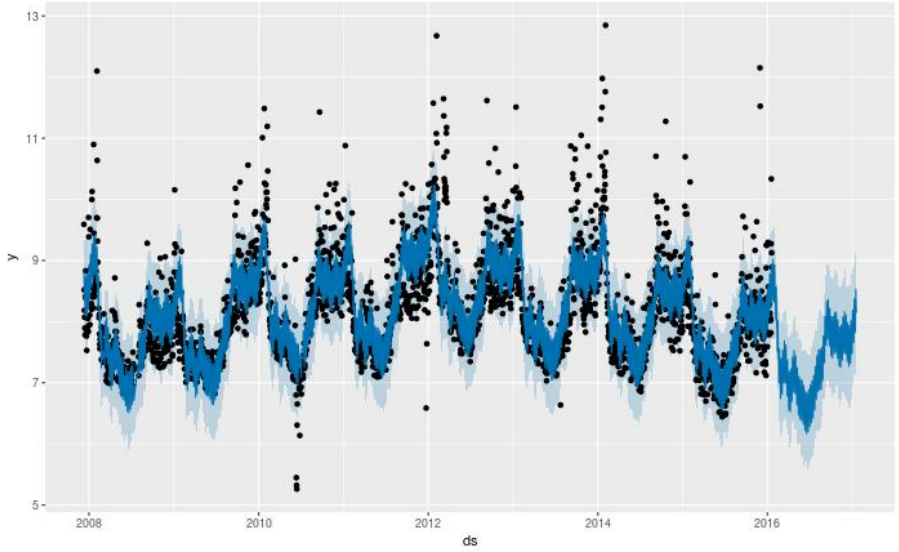
$$J(w) = \frac{1}{n}\sum_{k=1}^{n}(y_i - w^T x_i))^2 \qquad w_j^{(t+1)} := w_j^{(t)} - \alpha\frac{\partial J(w)}{\partial w_j}$$

Learning algorithm 2: just calculate the exact optimal solution

$$w = (X^T X)^{-1} X^T y$$

Regression: Predicting demand for store items



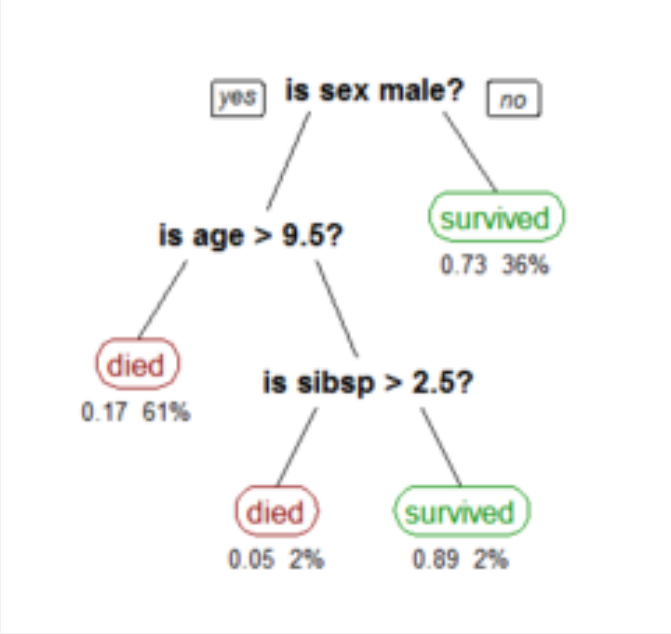


Good features are more important that the sophisticated model – example:
By using radial basis functions (left picture) we can easily model seasonality (right picture).

Some other use cases for regression (including the linear regression):
life expectancy analysis, stock market prediction, airplane delay forecasts

An excellent talk on feature engineering for simple models: https://www.youtube.com/watch?v=68ABAU_V8qI
Visualizations: https://facebook.github.io/prophet/docs/quick_start.html

Decision tree – a simple classification model



Build a binary tree where each node makes a binary decision and leaves contain classes.

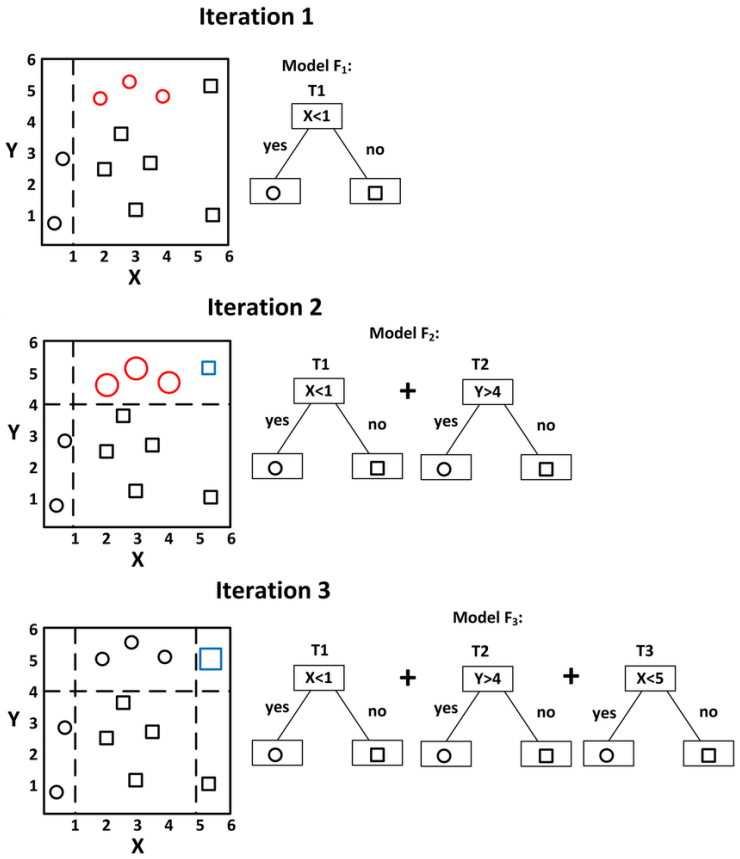
Algorithm:

- Select the best feature and threshold value to make a decision
- Repeat recursively until all values matching previous decisions have same class

How to select best feature for a tree node?

- Gini impurity
- Information gain

By applying gradient boosting we can make weak models very powerful



Gradient boosting is a model ensembling algorithm – it creates a model by combining other models. It's often applied to trees because of their simplicity and relatively fast generation.

The algorithm:

- Fit a weak learner (any simple model – usually a decision tree) on a random small subset of historical data and add it to a pool of already trained weak learners
- Increase weights of other samples that were wrongly predicted by the newly created tree
- Create a new tree and training it on the subset (sampled with weights, that reflect which samples are poorly predicted by the trees we already have)

Gradient boosted trees are in practice the most often used classifier. They are much cheaper than even the most sophisticated neural networks to train and maintain, and keep the explainability properties of ordinary decision trees (you can always see their gain distributions)

More about gradient boosting:
https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d
More about ensembling models in general: https://mlwave.com/kaggle-ensembling-guide/

Clustering: an unsupervised machine learning task

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad X = \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}$$

Unsupervised machine learning is a family of algorithms that work without labeled data.

They are often used for grouping similar data points, to find similar ones or to identify anomalies.

Breakthroughs in unsupervised learning may be necessary for general-purpose AI.

Clustering is the process of separating data points into groups of similar ones.

Note the difference: on the image above, the algorithm identified similar pixels – not classified them as human, car or sign!

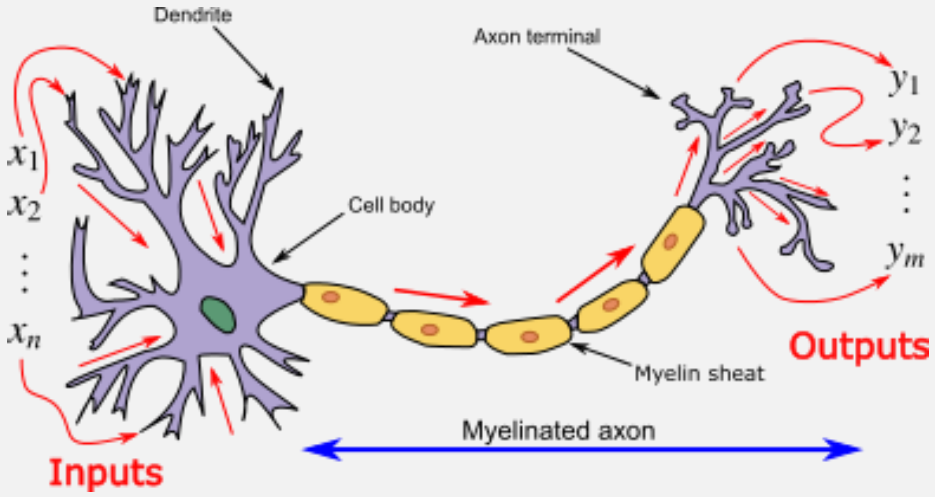
Popular algorithms: K-Means, Agglomerative Hierarchical Clustering, DBSCAN

Read more about clustering algorithms: https://scikit-learn.org/stable/modules/clustering.html

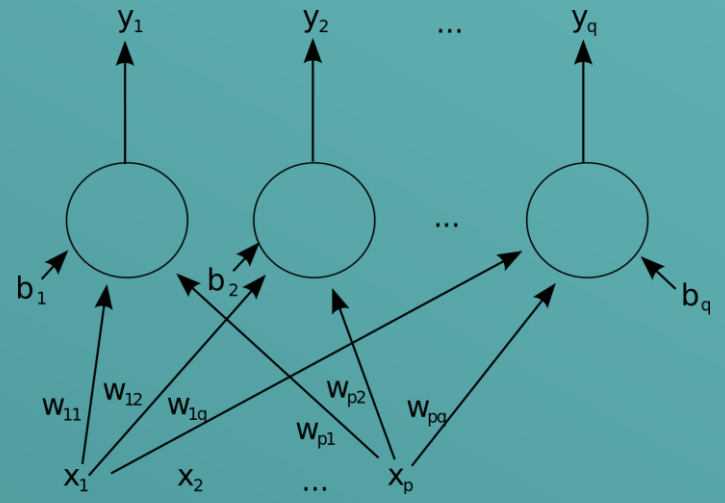
We need to go deeper!

Computer science can be inspired by biology

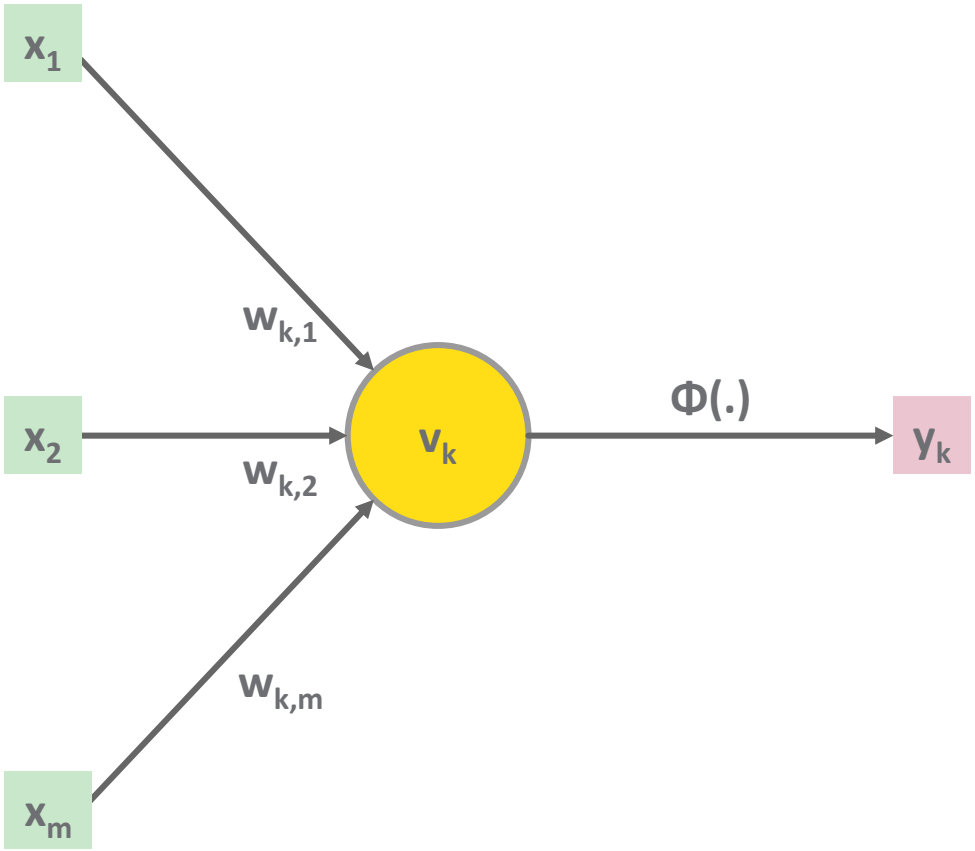
Brain is a network of cells called neurons, which can receive, process and send signals to other cells



Similarly, artificial neural network is made of layers of nodes, each node transforms data and passes it on to the next layer



Neuron – the building block of artificial neural networks



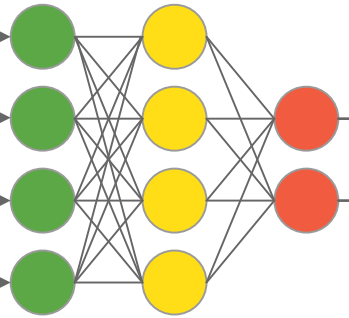
$$y_k = \Phi(v_k) = \Phi(w_{k,1}x_1 + w_{k,2}x_2 + \ldots + w_{k,m}x_m)$$

x_i – inputs
$w_{k,i}$ – neuron weights
v_k – weighted inputs
Φ – transfer function
y_k – output of the neuron

Neural networks are solving a simple optimization problem

By using gradient descent & backpropagation we can converge to a local minimum of any differentiable function

DATA



INPUT LAYER

HIDDEN LAYER

OUTPUT LAYER

92%
HOT DOG

8%
PIZZA

PREDICTION

TRUE VALUE

HOT DOG

0.08

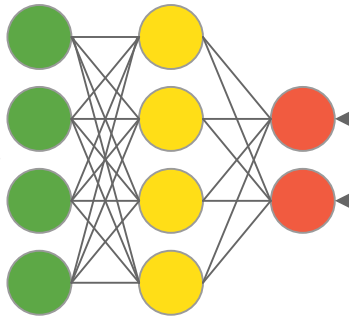
LOSS FUNCTION VALUE

This can be any differentiable function that we wish to minimize

UPDATE WEIGHTS (BACKPROPAGATION):

1. Calculate the gradient (derivative of loss function wrt weights of the network) for the output layer
2. Update weights of the output layer in correct directions
3. Recursively calculate gradient & update all other layers (always using outputs from the following layer to calculate gradient)

REPEAT UNTIL LOSS FUNCTION MINIMUM IS REACHED



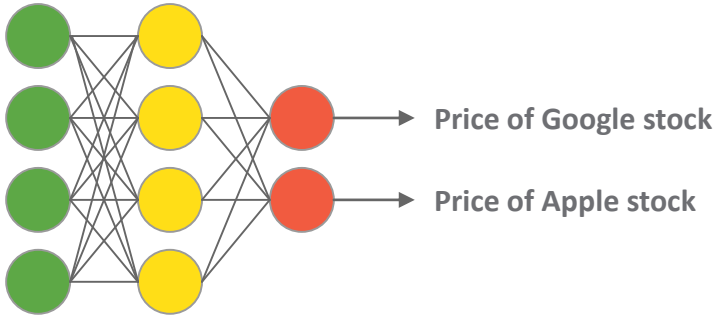
FEED-FORWARD NEURAL NETWORK (PERCEPTRON)

This is a simplified example – in real life we always use deep neural networks, which have much more hidden layers. There are also other types of layers than feed-forward ones.

Neural network: architecture + loss function = selection of machine learning task

Regression

Architecture:
Output one value for every variable you want to predict

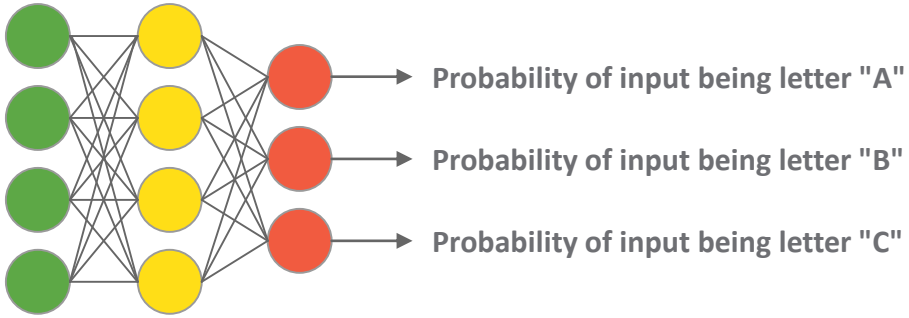


Loss function:
Mean square error (measure of distance between predicted and actual values)

$MSE = mean([y_{true} - y_{pred}]^2)$

Classification

Architecture:
Output one value for every possible class



Loss function:
Cross-entropy (measure of error between probability distributions)

$LogLoss = -sum(y_{true} \cdot log(y_{pred}))$

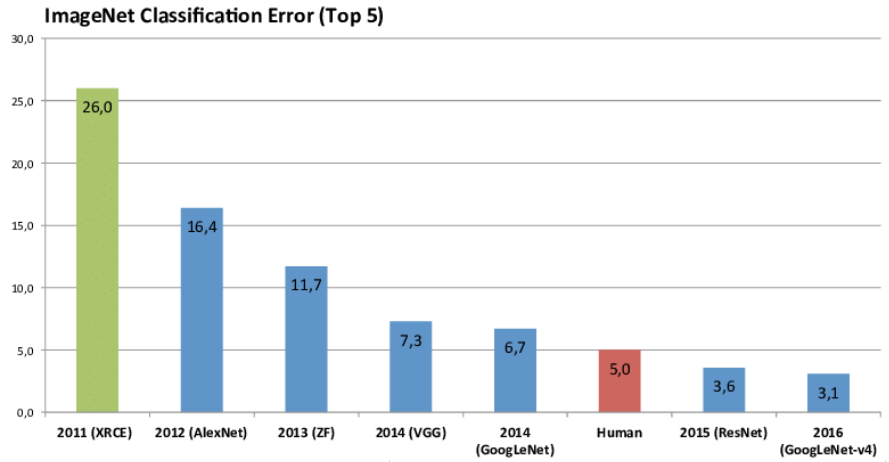
How to select an appropriate loss function for your problem? Use a maximum likelihood estimator[1]
How can we make sure our output probabilities are in [0,1]? Use a sigmoid function[2]

1. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/
2. https://en.wikipedia.org/wiki/Sigmoid_function

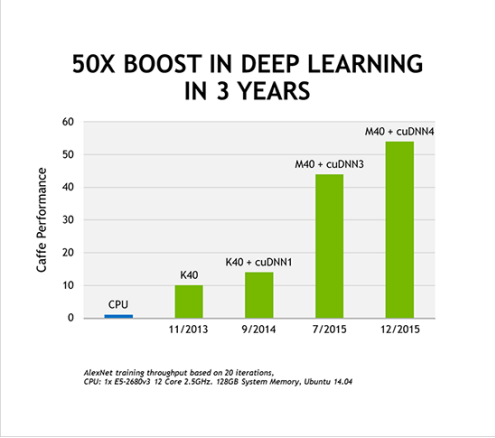
An old theory becomes possible thanks to Graphics Processing Units (GPUs)

A brief history of artificial neural networks

- **1943:** First mathematical model of neurons, implemented with electrical circuits (Warren McCulloch, Walter Pitts)
- **1957:** Definition & implementation of perceptron: a single-layer neural network, exactly same as we use today (Frank Rosenblatt)
- **1969:** Proof that perceptron is limited: cannot solve the XOR problem (Marvin Minsky, Seymour Papert)
- **1986:** Neural networks with multiple layers can in theory learn non-linear functions using backpropagation (Geoffrey E. Hinton). Despite rising interest in neural networks and discovery of new layers (convolutional, recurrent) they are simply too inefficient to be applied to any real problem
- **2012:** Creation of AlexNet, an improved convolutional neural network, trained using GPUs, that won a ILSVRC-2012 ImageNet classification competition, beating classical machine learning methods (Alex Krizhevsky, Ilya Sutskever Geoffrey E. Hinton)
- **2016:** AlphaGo neural network model beats Lee Sedol in Go – a game which has about 10^170 possible move combinations (comparing to about 10^50 for chess) and cannot be solved by even the most sophisticated search algorithm (Google DeepMind)

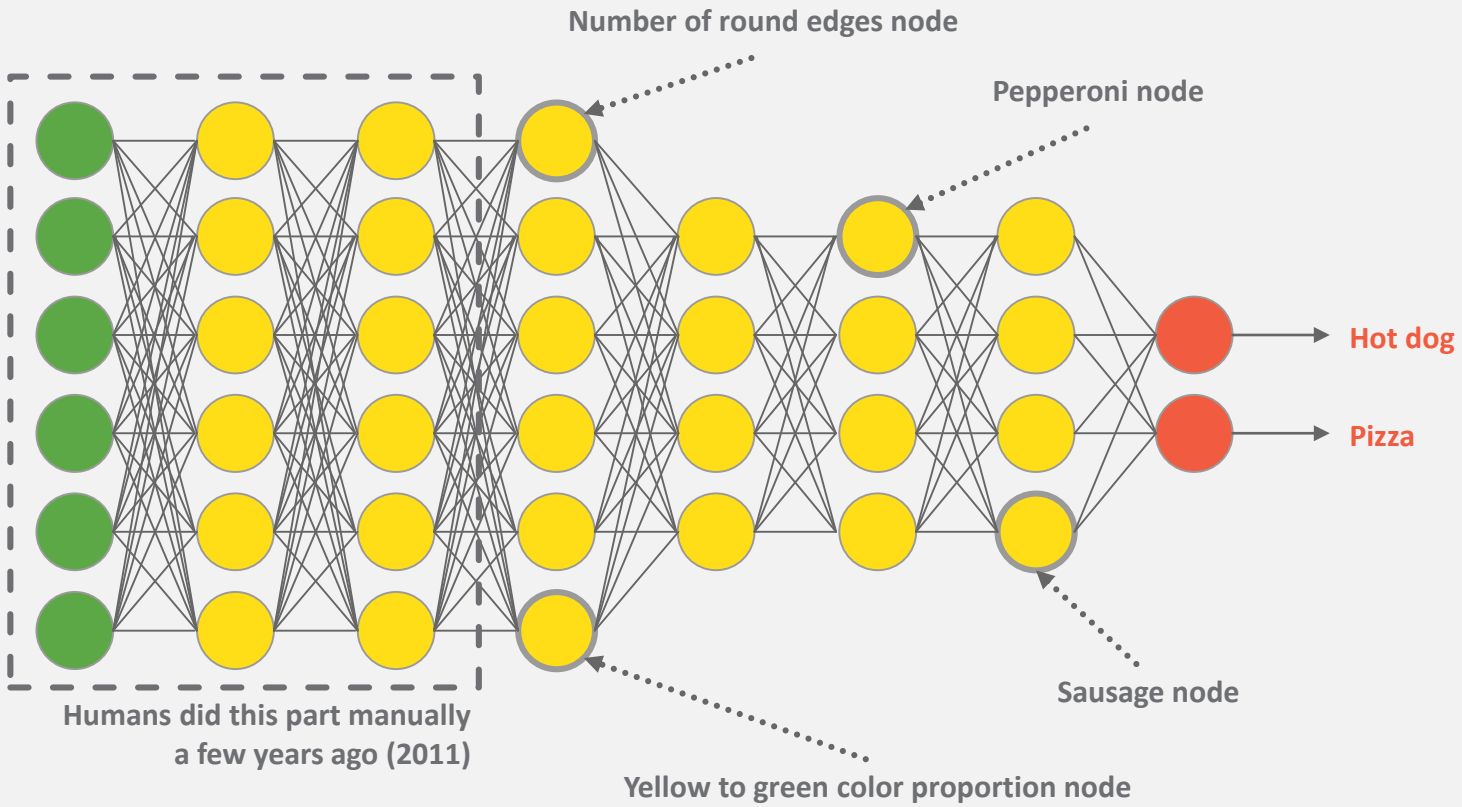


Neural networks proved suitable for solving real problems only recently. This change is mainly a result of improvement in GPU efficiency, as both forward and backward propagation in a network are just a simple vector and matrix operations.

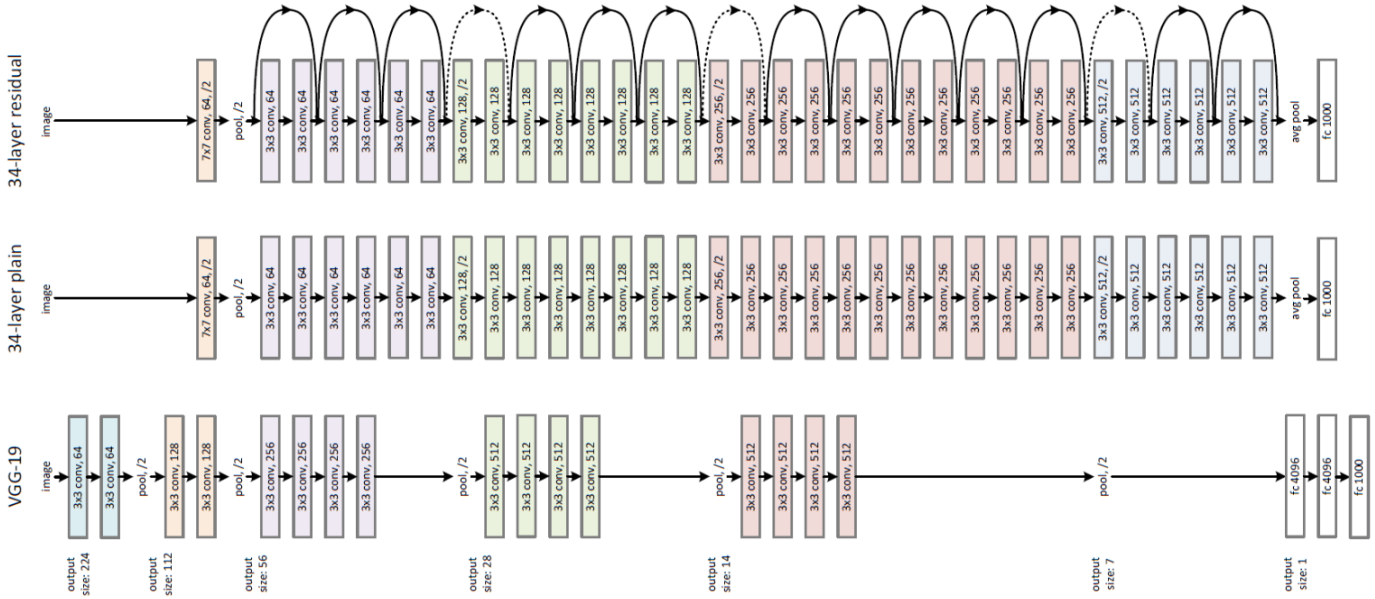


Source: https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/

Deep learning eliminates the need for feature engineering



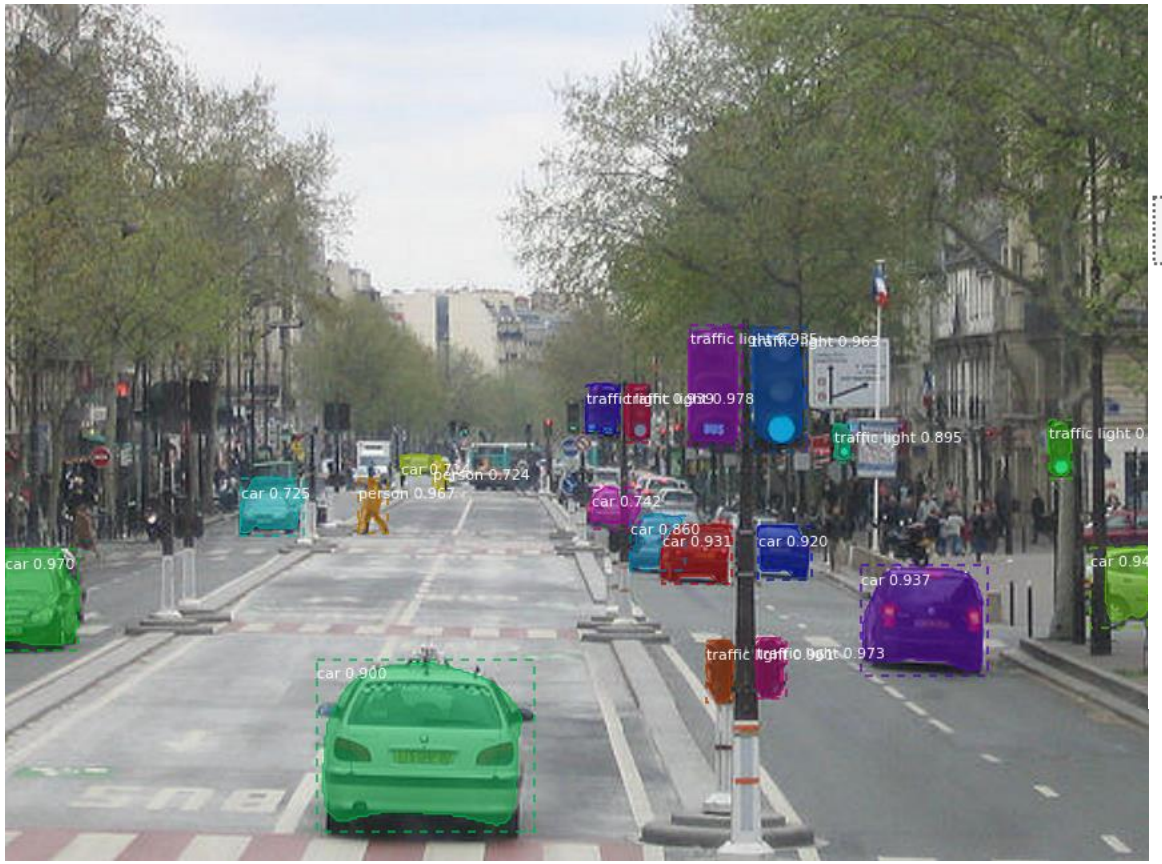
Deep learning: Image classification

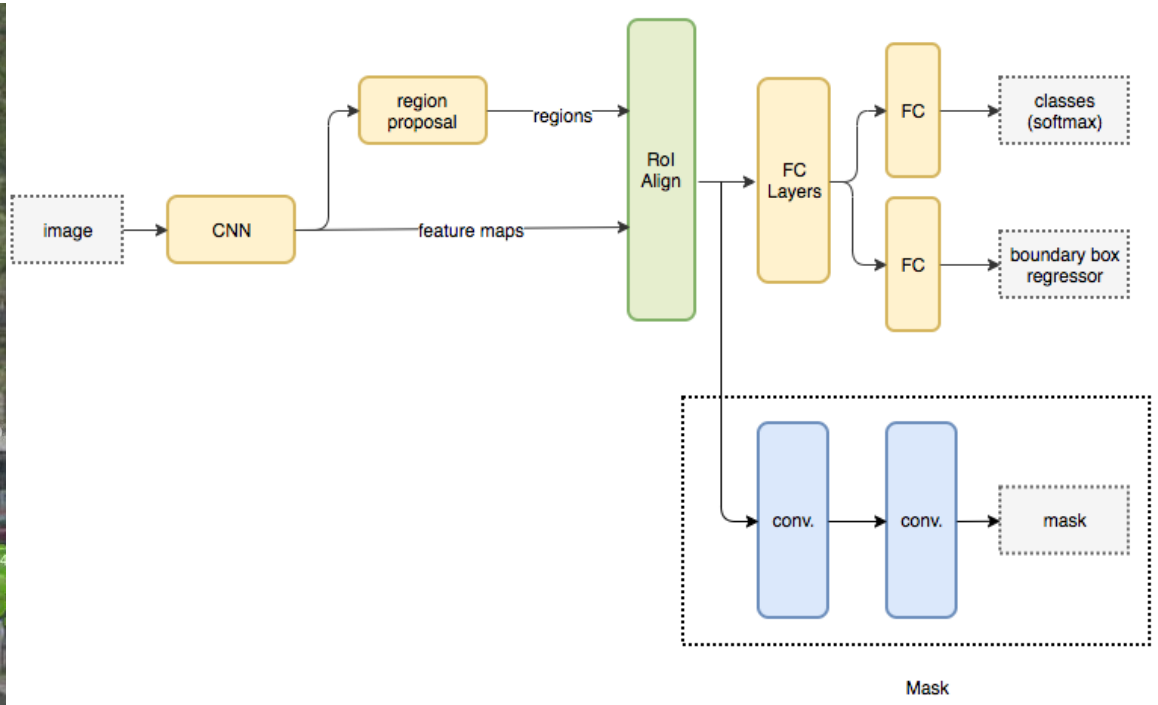


Three popular neural network architectures used for image classification: ResNet with Skip (top), plain convolutional network (middle), VGG-19 (bottom). Each box denotes a network layer, "NxM conv K" denotes a convolutional layer with kernel dimensions (N,M) and K input variables. Convolutional layers are extremely powerful for extracting information out of images, to read more about them visit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
Image source: https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

Deep learning: object detection & image segmentation

To create models that are actually useful in the real world, we need even more sophisticated network architecture

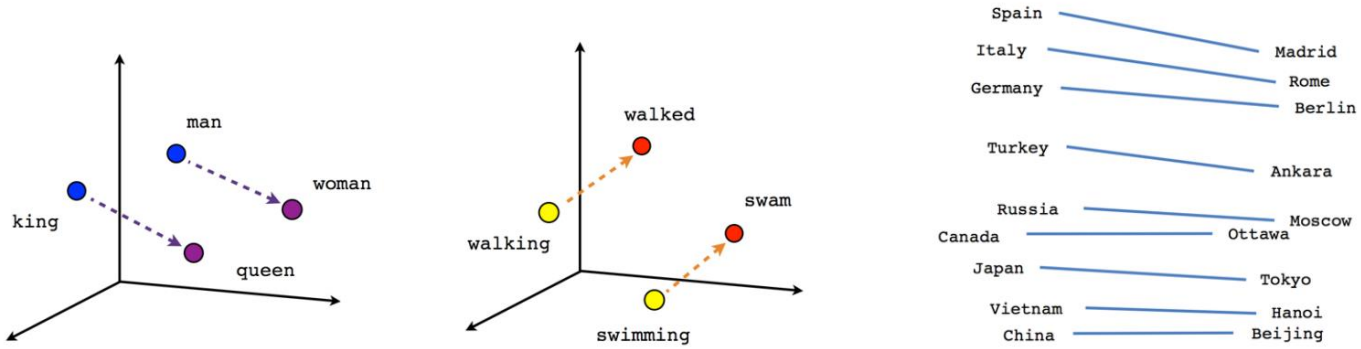




Example: to perform image segmentation (left picture), the network needs multiple outputs (right picture) and a more complex loss function taking them all into account. To read more on image segmentation see: https://arxiv.org/pdf/1703.06870.pdf and the implementation: https://github.com/matterport/Mask_RCNN

By representing data as vectors, we can apply deep learning to any problem

king – man + woman = queen



Male-Female | Verb tense | Country-Capital

Images were a natural first choice for deep learning applications: the image is essentially a matrix of color values

Other types of data needs to be embedded into a fixed-size vectors in order to train neural networks on it. For example, natural language processing (document classification, named entity detection) usually requires mapping words into a vector space in a way that preserves semantic relationships (as in the picture above).

For this purpose, unsupervised models such as word2vec and GloVe were invented.

Neural networks expand the scope of machine learning

Generative models

Some models allow to create new data from the same distribution as historical data.

Neural networks can have this property too, giving computers ability to generate images, films, music or stories.

Reinforcement learning

Instead of training models on historical data, we can have an environment which reacts to network outputs. This approach is perfect for train game bots, or even real robots!

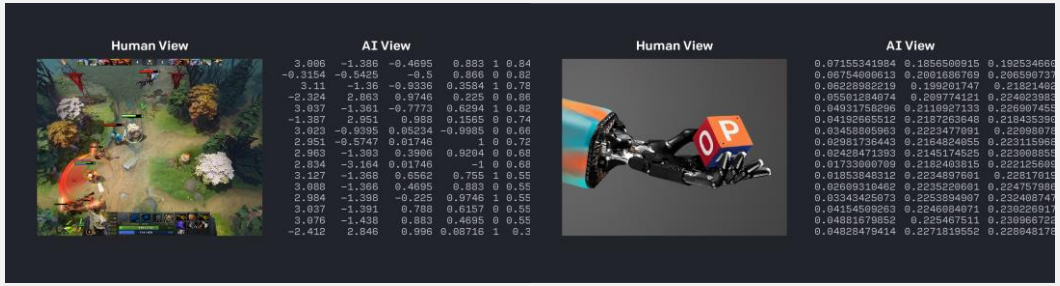
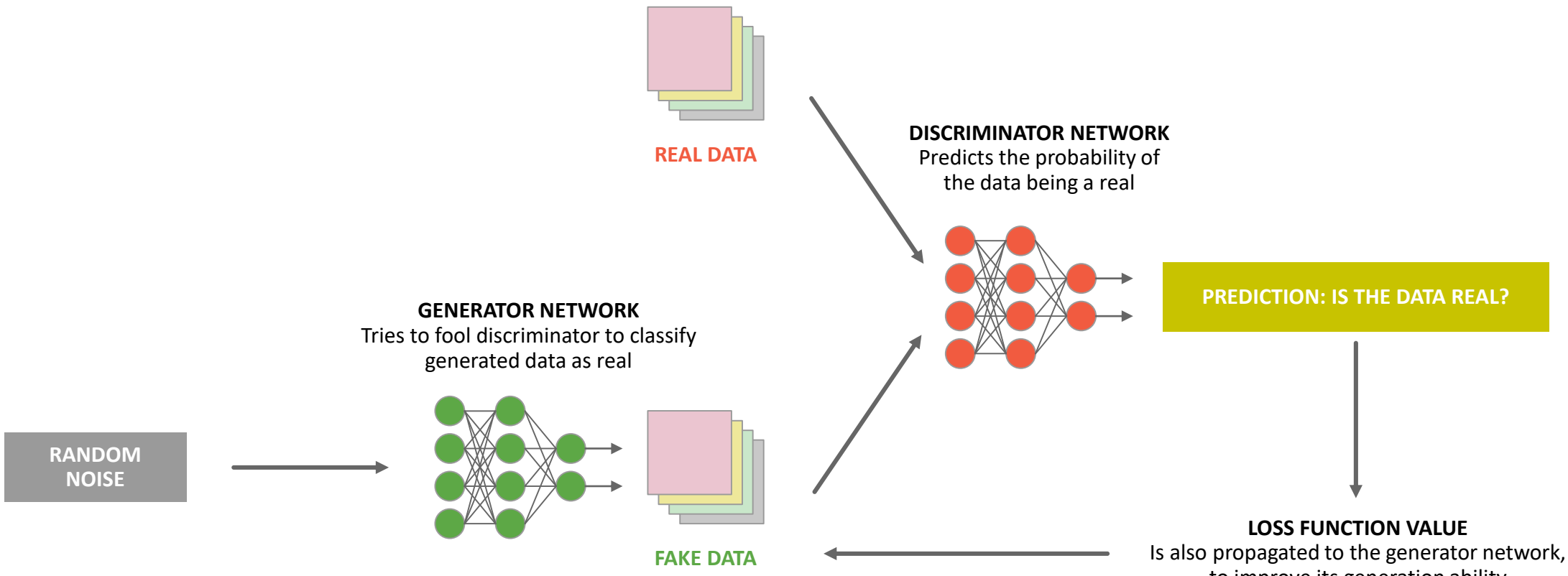


Image sources: https://prisma-ai.com/ https://openai.com/blog/how-to-train-your-openai-five/

Generative adversarial networks – a simple trick that makes models generative



Generative Adversarial Networks:
Making AI creative

Read more: https://medium.com/element-ai-research-lab/stabilizing-neural-style-transfer-for-video-62675e203e42

Reinforcement learning: a new task in the realm of AI

The one thing game bots, industrial robots and self-driving cars have in common

ENVIRONMENT

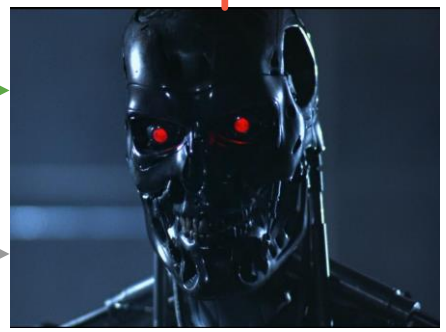
STATE

ACTION

INTERPRETER

REWARD

STATE REPRESENTATION



AGENT

Agents often need to combine multiple machine learning tasks:

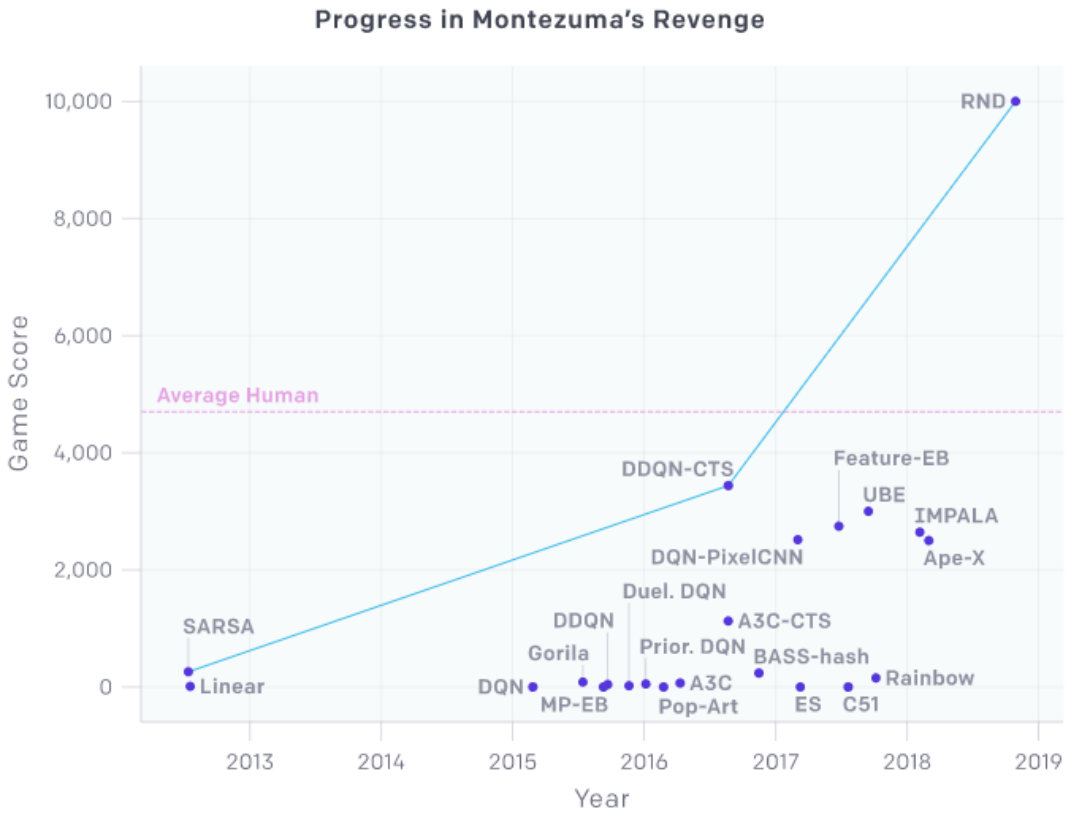
- Classification to select an action: move, shoot, etc.
- Regression to select action values: how much power to use, which direction to shoot

Neural network:
input=state,
result=action

Backpropagation:
calculates loss based on rewards received from the environment

Reinforcement learning: can we make AI curious?

Right now, even the most sophisticated models can look stupid in some situations



By adding a penalty to the loss function when the agent can accurately predict what's going to happen in the environment, we can trick it to explore more, showing behavior similar to human curiosity. This allowed OpenAI to create a state-of-the-art model for Montezuma's Revenge – a game that requires a lot of exploration to earn points.

However, such creative agents are still very stupid – placing such AI in a Quake environment with a TV on a wall made AI stare at it forever (left animation). While TV is off the agent continues to explore the world (right animation).

https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/

How to get started with AI?

Common foundation for doing machine learning

KNOWLEDGE

Probability theory
Statistics
Machine learning models
Evaluation metrics

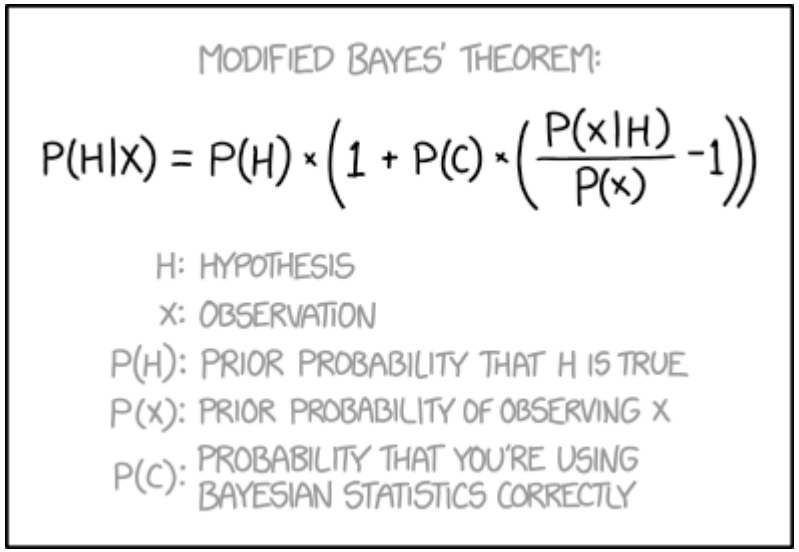
TOOLS

Python
SQL
Numpy & Pandas
Scikit-Learn

SKILLS

Data exploration
Model selection
Model evaluation
Data visualization

Theoretical background - is it really necessary?



Using machine learning is simple thanks to many good libraries and extremely helpful community

You can easily achieve good results without understanding what you're doing in detail

Solutions to real-world problems often need to be theoretically sound and explainable

Universities: Top AI & Machine Learning research hubs worldwide

Top AI & ML Universities Worldwide[1]

- Carnegie Mellon University
- Cornell University
- Tsinghua University
- Stanford University
- Technion
- Peking University
- Georgia Institute of Technology
- Massachuesetts Institute of Technology
- National University of Singapore
- HKUST
- Chinese Academy of Sciences
- Columbia University
- University of California – Berkley
- University of Toronto
- University of Illinoi at Urbana-Champaign
- University of Maryland – College Park

Practical tips on choosing where to apply

Machine learning & artificial intelligence are extremely international subjects – most researchers travel around the world to work or present their results.

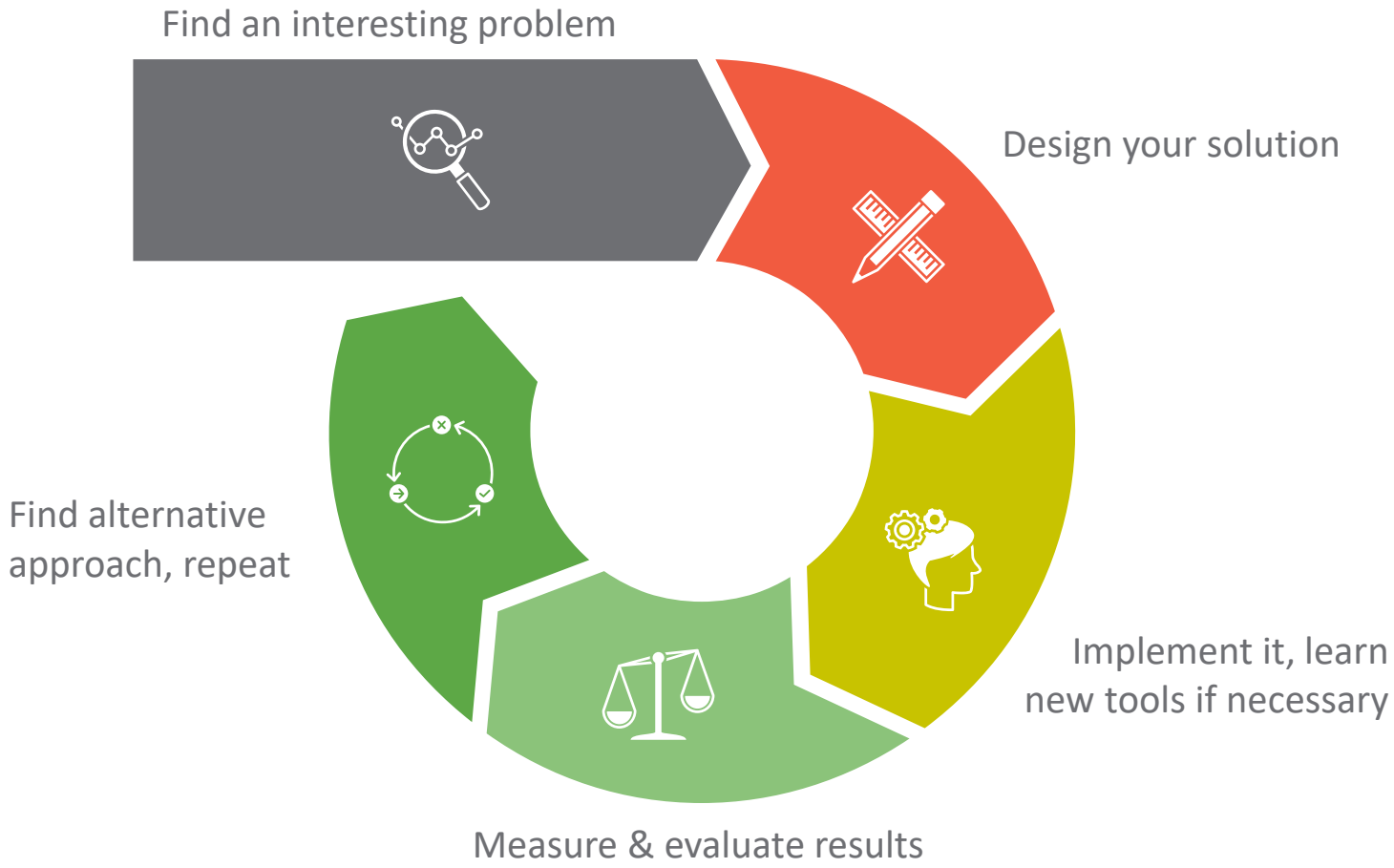
It's not that important to be in a top university worldwide – what matters, is that you have access to international conferences & local researchers that are up-to-date with latest findings in machine learning.

In practice, being in the top university in your country is enough, unless you don't want to get into research (but I strongly encourage to try)

Most university rankings are created based on the number of publications in certain domains – this isn't always the best metric. Always try to find what people are giving lectures and what kind of research are they doing exactly.

1. Ranked by number of publications in AI, ComputerVision, ML & Data Mining, Natural Language Processing, The Web & Information Retrieval from 1990 to 2019. Source: http://csrankings.org

Learn the tools by using them in your projects

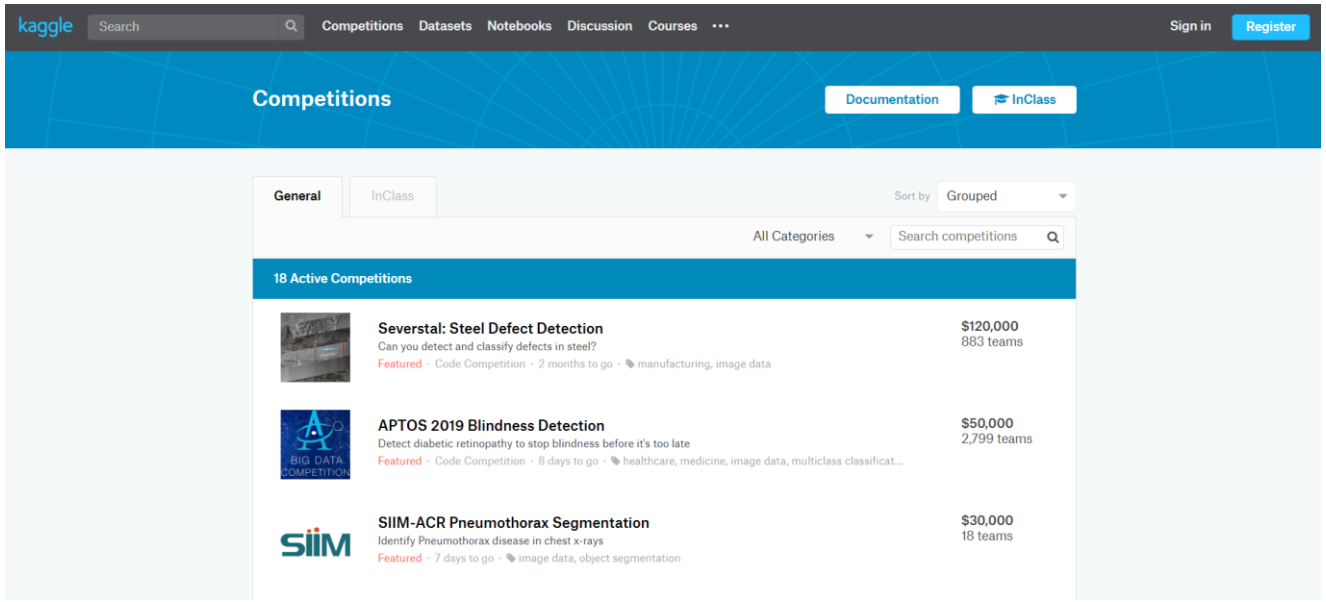


Popular source of interesting projects #1

Kaggle – the cornerstone of every data science career

Taking part in data science competitions is often the best way to learn & test skills

- Competitions – solve problems submitted by real companies, win prizes (or at least get experience)
- Datasets & notebooks
  - Explore solutions of other people shared as public notebooks
  - Use hosted Jupyter Notebook with easy access to datasets & a free GPU
  - Edit other people's notebooks to improve or play with their solutions
- Discussion – data science community is very helpful, even during competitions people often share tips
- Courses – new feature of Kaggle, for a start this is probably where you want to go



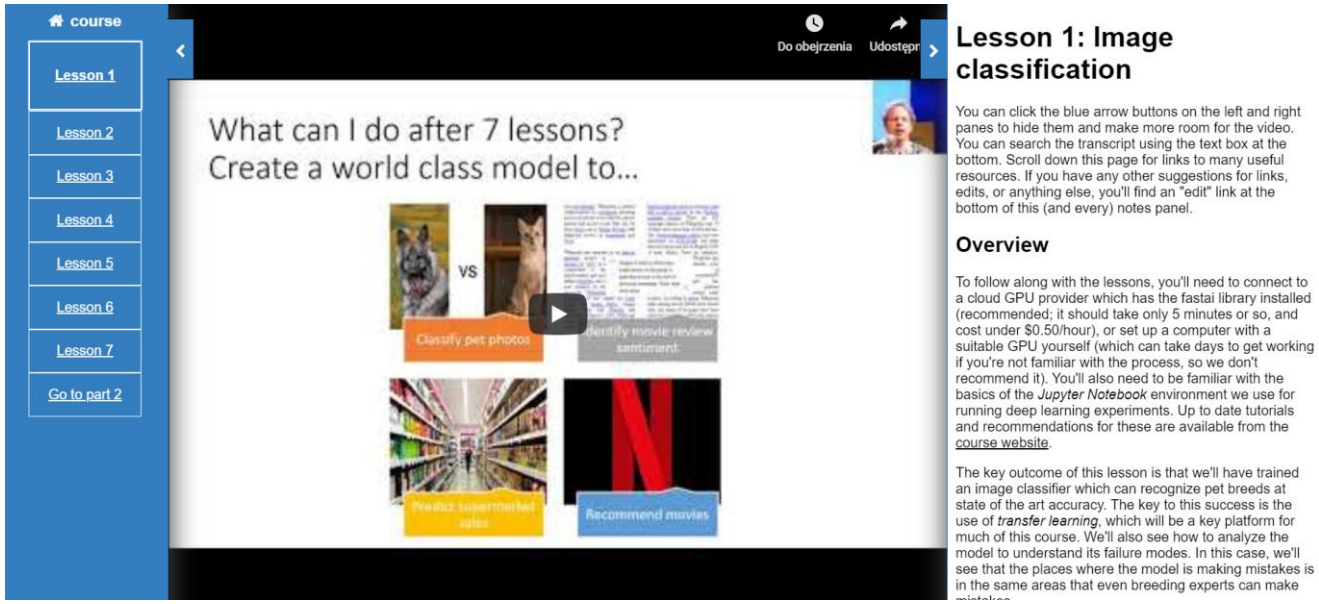
Website: https://www.kaggle.com/

Popular source of interesting projects #2

Fast.ai – hack your way into machine learning

If you are more of a developer than a mathematician, this courses are for you

- Assumes that you already know how to program (and know basics of Python as well)
- Focused on practical aspects of machine learning
- Most courses focus on deep learning (neural networks)
- You won't learn much theory there
- Plenty of practical tips:
  - Where to get free (or cheapest) computing power – especially GPUs
  - Best practices when working with machine learning models & deploying them
  - Links to tutorials & other relevant content
- Community is more goal-oriented (similar to coding bootcamp) rather than process-oriented (similar to university)



Website: https://www.fast.ai/

Popular source of interesting projects #3

University: student societies & research groups

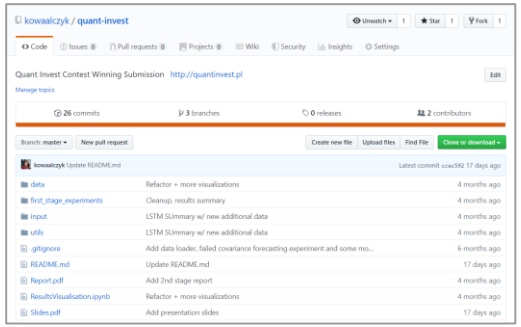
Enroll in a machine learning subject or reach out to fellow students & researchers

- Machine learning subjects
  - Quality depends heavily on your university
  - Can be a good way of getting into ML & AI
- Students societies
  - Usually ran by people interested in machine learning and willing to share their skills
  - Good places to look for competition teammates, project ideas and new knowledge sources
- Machine learning & AI research groups
  - Researchers often could use free help, and will teach you necessary skills to make it possible
  - This is excellent way of getting an interesting topic for your BSc, MSc or PhD
- If you're in one of the top universities worldwide, ML & AI subjects should be the best way to learn
- Otherwise, being in the top university in any country should still allow you to meet skilled researchers

Machine Learning Society at University of Warsaw: http://knum.mimuw.edu.pl

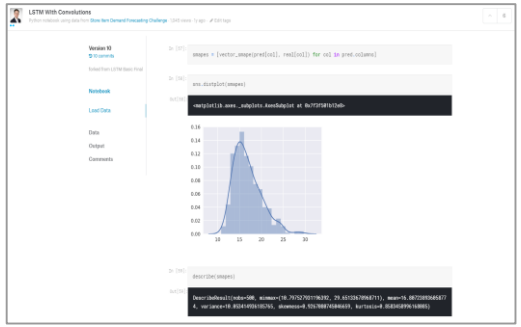
Share your work with others

Take time to clean and publish your work – make it easy for other people to discover and use



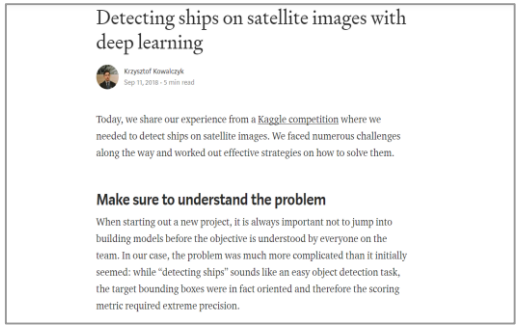
GitHub

- A perfect place to store & showcase your code
- You will use git to manage your source code version anyway
- Nice place to discover new software projects



Kaggle Notebooks

- If you're working on Kaggle, publishing a notebook is easy
- Very useful during competitions, to suggest ideas or present your results
- Attracts less people but more discussion than GitHub project



Medium blogs

- If you like to write, consider making a small blog
- Platforms like Medium & LinkedIn make it easy to publish new articles
- Depending on your taste, you may

LinkedIn Slideshare

- When you make a presentation, post it to Slideshare in order to reach a wider audience
- Using LinkedIn will also make your work easy to discover

Always use the most reliable source of knowledge

Likely unreliable

Video tutorials
- Almost always too simplified
- Only introduce confusion when you really need to understand what you're doing

Entry-level courses
- Aimed at people without programming or math skills
- Not approved by any scientific institution

Online communities
- Social media groups

Simplified, but still useful

Blog posts
- Published by top researchers & professionals

Kaggle notebooks
- Less popular ones are almost never correct
- Sometimes, notebook is not very popular, but is approved by many experienced users (by commenting / upvoting)
- Ones that are both popular and approved by many experienced users can be quite reliable – also, seeing references to related research papers is a good sign

Presentations
- From local meetups (student club meetings, smaller conferences, etc.)

Mostly correct

Research papers
- Popular (referenced often)
- Reviewed & publish in a journal

Library documentation
- Or alternatively, GitHub readme and issues

Professional forums
- Stackoverflow, stackexchange
- Only answers that are both valid & upvoted

University lectures
- Can be outdated, but always correct
- Top universities (MIT, Stanford) also publish their lectures & related materials online

Presentations
- From top scientific & technical conferences (PyCon, PyData, NIPS, ICML, etc.)

Cheat sheet: some useful & reliable sources of machine learning knowledge

https://www.fast.ai

https://paperswithcode.com/

https://www.kaggle.com

https://openai.com

https://www.coursera.org/learn/machine-learning

https://arxiv.org/

http://cs231n.stanford.edu/

https://docs.scipy.org/doc/

https://www.deeplearningbook.org/

https://pandas.pydata.org/pandas-docs/stable/

https://scikit-learn.org/stable/user_guide.html

https://towardsdatascience.com

https://machinelearningmastery.com/start-here/

https://github.com/owainlewis/awesome-artificial-intelligence

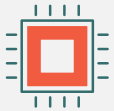
Do what you enjoy the most: 4 typical career paths

AI Researcher

Designs new algorithms or applies existing ones to new domains

Typically works at a university or research team in large company

Implements things from scratch, uses well-known data to compare results



ML Developer

Optimizes models that are already proven to solve a specific problem well

Necessary only in projects where model inference cost is very high

Works with algorithms & code, rather than data

Data Scientist

Chooses the right tools and methods to solve a specific, real-life problem

In smaller companies also performs development & data engineering work

Uses existing libraries & tools, spends more time interacting with people

Data Engineer

Manages the flow of data from its sources to final model predictions

Crucial in projects with high data throughput or real-time predictions

Uses database & pipeline management tools

AI researcher – exploring the unknown

Designs new algorithms or applies existing ones to new domains

Skills

- Understand optimization algorithms in detail
- Follow scientific process of hypotheses testing
- Create deep learning models from scratch

Responsibilities

- Stay up to date with current research
- Teach & share knowledge with others
- Adapt fresh research to new domains

Tools

- High-performance programming language
- Experiment & data management tools
- Low-level machine learning libraries

Products

- Research papers & other publications
- Tools for developers & data scientists
- Contributions to open-source projects

ML developer – working close to the metal

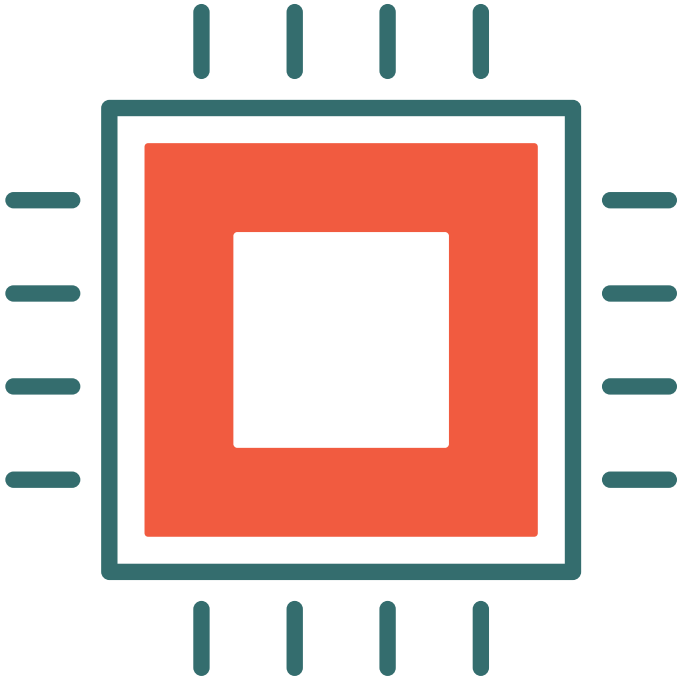
Optimizes models that are already proven to solve a specific problem well

Skills

- Understand optimization algorithms in detail
- Trade off accuracy for speed or memory
- Develop efficient & correct software

Responsibilities

- Stay up to date with current research
- Improve efficiency of existing software
- Help others improve code efficiency



Tools

- High-performance programming language
- High- and low-level ML libraries
- Testing, build & deployment tools

Products

- Optimized machine learning models
- Models ported to embedded devices
- Workshops & talks on code efficiency

Data scientist – the sexiest job of the XXI century?

Chooses the right tools and methods to solve a specific, real-life problem

Skills

- Analyze data & create visualizations
- Finding the right model for a given task
- Measure & compare model effectiveness

Responsibilities

- Solve real-world business problems
- Create models & prove their effectiveness
- Gather insights from large pools of data

Tools

- High-level machine learning libraries
- Data manipulation & visualization tools
- Optimization & simulation tools

Products

- Presentation of business problem solution
- Model optimally solving a given problem
- Dashboards clearly visualizing data

Data engineer – making the most of "big data"

Manages the flow of data from its sources to final model predictions

Skills

- Design software architecture & data model
- Choose right databases & compute platform
- Integrate & manage many data sources

Responsibilities

- Optimize data processing for a given model
- Improve speed of model inference
- Stay up to date with software & infrastructure

Tools

- High-level machine learning libraries
- Databases & pipeline management tools
- Cloud infrastructure monitoring tools

Products

- Efficient pipeline for data processing
- Models scaled to handle "big data"
- Dashboards clearly visualizing data

Specializations have flexible requirements and a common foundation

	AI Researcher	ML Developer	Data Scientist	Data Engineer
EXPERIENCE TO GET	PhD in computer science, mathematics or related field	BSc in computer science, related industry experience or projects	MSc in mathematics, statistics, computer science or related field	BSc in computer science or experience with large datasets
KNOWLEDGE TO LEARN	Algorithms & data structures Advanced machine learning Numerical methods	Algorithms & data structures Parallel programming Software & systems architecture	Advanced probability & statistics Advanced machine learning Probabilistic programming	Software systems architecture Parallel programming Distributed systems
TOOLS TO MASTER	Low-level ML library[1] High-performance language[2]	Low-level ML library[1] High-performance language[2]	Data visualization libraries[3] Optimization algorithms & tools	Databases (SQL and No-SQL) Pipeline management tools[4]

KNOWLEDGE TOOLS SKILLS

[1] Low-level ML libraries: Tensorflow, Pytorch, Theano
[2] High-performance languages: C++, Scala, Go, Rust
[3] Data visualization libraries: Matplotlib, Seaborn, Plotly, Bokeh
[4] Pipeline management tools: Spark, Airflow, Dagster, DVC

How to choose the right development path?

Learn the basics

No matter what you decide to do in your life, knowing the basics of machine learning won't hurt

Start by taking university classes or online courses (or both, one may not be enough to start quickly)

Take part in machine learning competitions, don’t hesitate to share your successes (or takeaways from failures)

Write a personal project or publish code you already have

Do everything to get your first internship or job, after that it gets much easier to try new things

Try different specializations

Learn enough to get an internship in one of the 4 roles

When working, learn as much new tools as possible, you will likely continue to use at least some of them

The same is true for people: get to know as many as you can – some of them may help you in the future, others may need help from you

Internships are a perfect way of exploring different career paths – you don't need to get the first one right

Note down your experiences and take them into account when applying for the next job - until you find one you know you can do longer than 3 months

Dig deeper when it's interesting

The more interesting the problem you are facing, the more time you should take to explore it

If people want you to do other things, learn to politely say no – more often than not it will be beneficial for everyone

Value your time – if you get this far, it's likely you will find most problems interesting

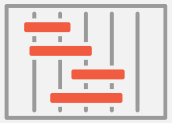
Find mentors – people a little more experience than you, don't fear reaching out to them

Final thought: AI is not limited to machine learning

KNOWLEDGE REPRESENTATION

RESOURCE ALLOCATION

CURRENT MACHINE LEARNING SYSTEMS
CANNOT SOLVE ALL OF OUR PROBLEMS



PLANNING & SCHEDULING

CONSTRAINT SATISFACTION PROBLEM

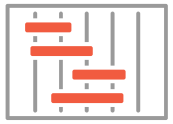
EXPERT SYSTEMS

Solving the unsolvable: NP-complete problems in artificial intelligence

How quantum computers and unsupervised learning can bring us closer to general AI

RESOURCE ALLOCATION

CONSTRAINT SATISFACTION PROBLEM



PLANNING & SCHEDULING

Currently solved using approximation algorithms, quantum computers will make optimal solutions achievable

KNOWLEDGE REPRESENTATION

EXPERT SYSTEMS

Currently built manually using well-established algorithms, breakthroughs in unsupervised learning have potential to make this process much more efficient

Q&A