Particles and Cosmology

16th Baksan School on Astroparticle Physics

Machine Learning in Astroparticle Physics

Oleg Kalashev
Institute for Nuclear Research, RAS

Lecture 1

April 10-18, 2019

Overview

- Introduction to ML. Data Analysis and Tools

- Artificial neural networks (ANN) intro. Feed-forward ANN

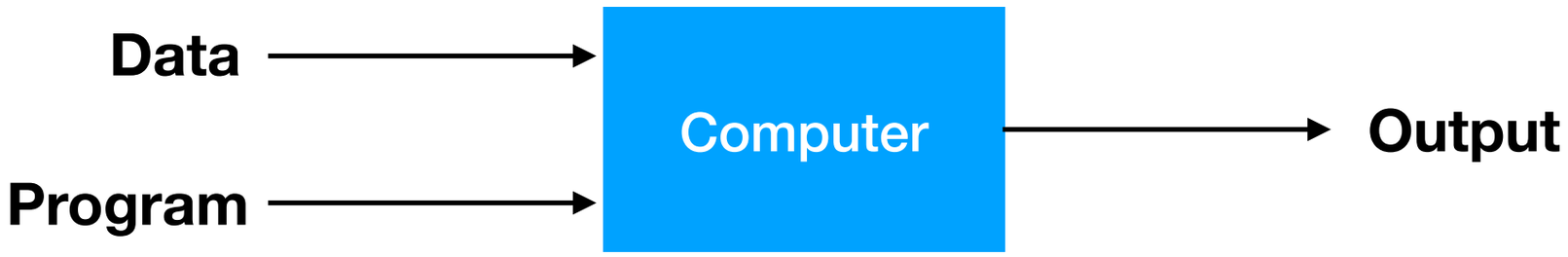
- Optimizing ANN.

  - Hackathon (solving real problem)

- ML applications in astroparticle physics

What is Machine Learning

Traditional programming



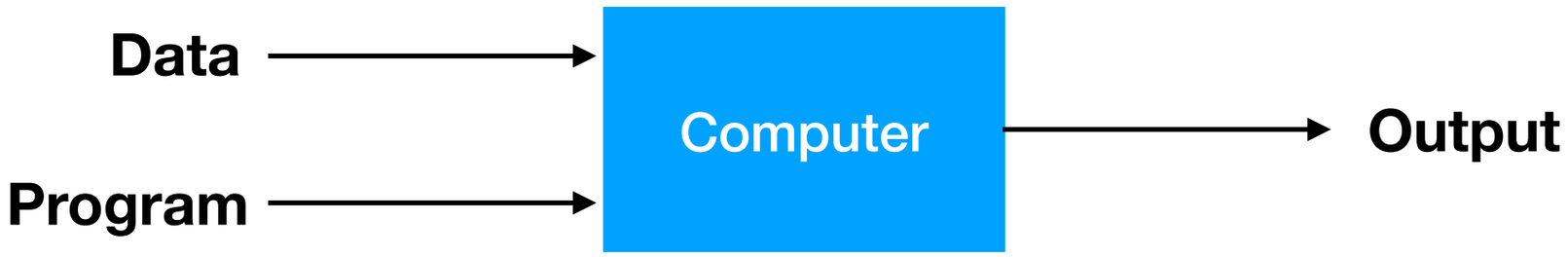
program = algorithm + data structure

Automating a task by writing rules for a computer to follow

2 + 3 ⟶ **5**

What is Machine Learning

Traditional programming



program = algorithm + data structure

Automating a task by writing rules for a computer to follow

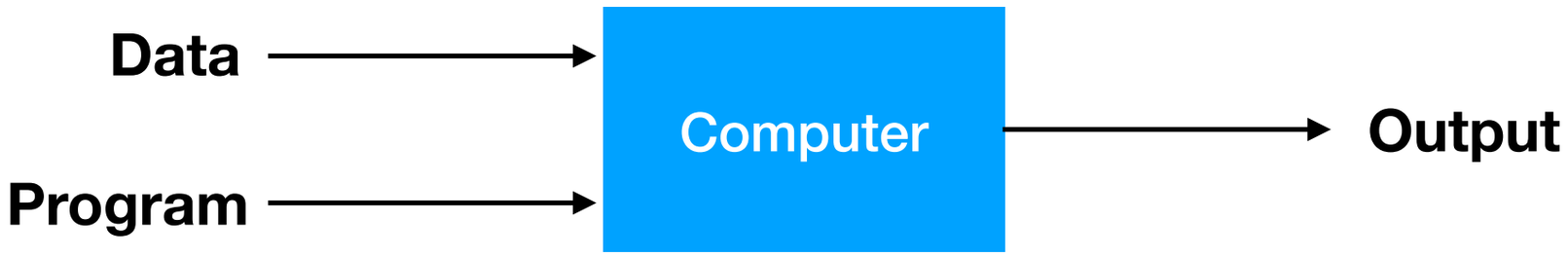
2 + 3 ⟶ **5**

$$Output = \sum Input$$

Good for relatively simple rules

What is Machine Learning

Traditional programming



program = algorithm + data structure

Automating a task by writing rules for a computer to follow

2 + 3 ⟶ **5**

$$Output = \sum Input$$

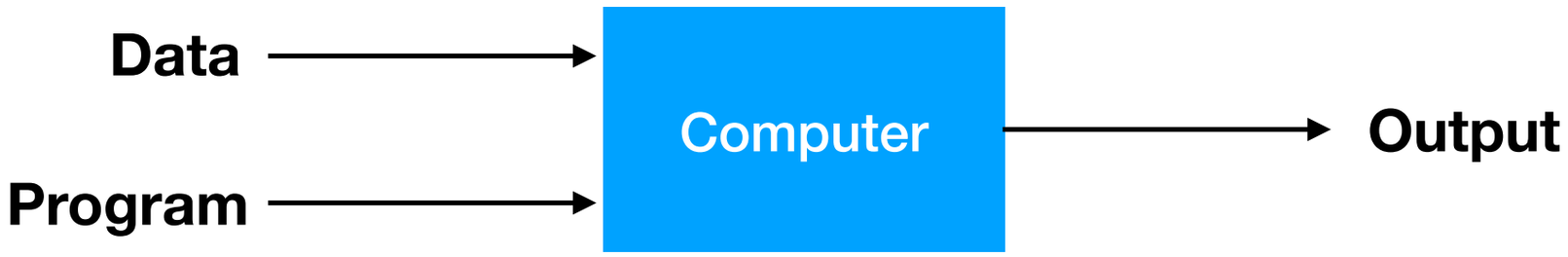
⟶ **504192**

Many examples

The more diverse is the data the more complicated and less effective is the algorithm

What is Machine Learning

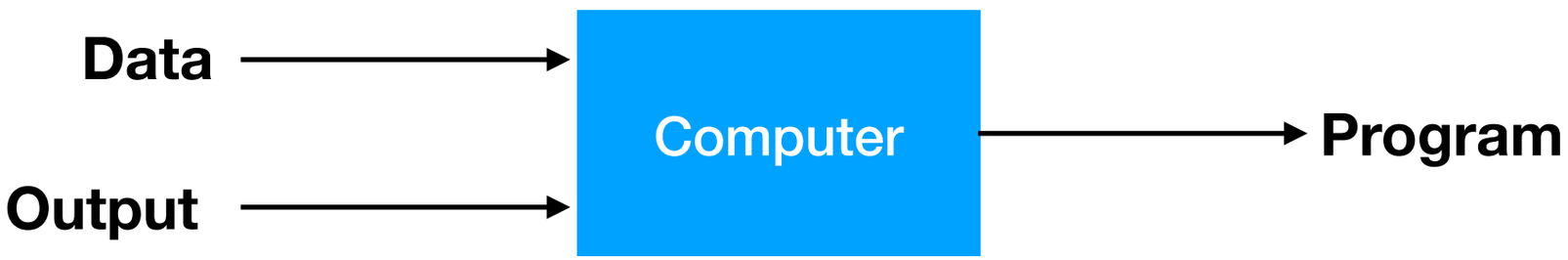
Traditional programming



program = algorithm + data structure

Automating a task by writing rules for a computer to follow

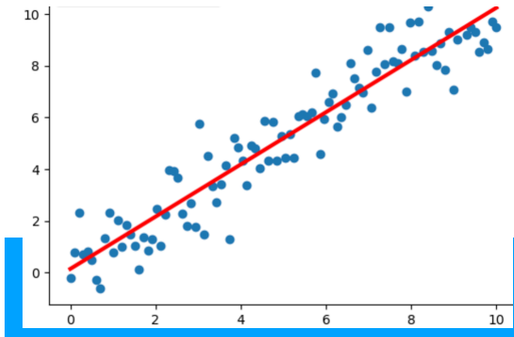
Machine learning



A step further: automate the task of writing the rules based on data

Machine Learning Tasks

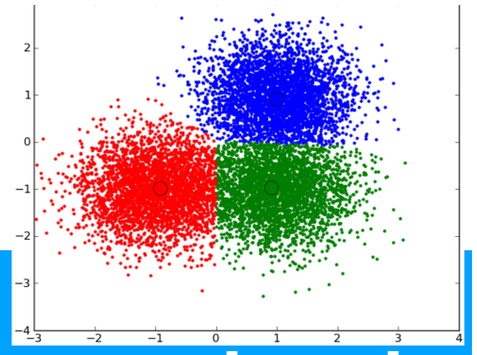
Machine Learning



Supervised Learning

Regression,
Classification

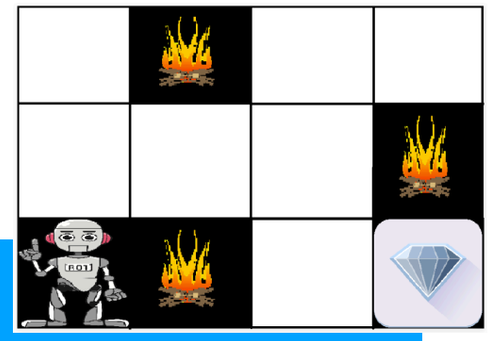
data is labeled



Unsupervised Learning

clustering,
data generation,
noise reduction,
anomaly detection

data is not labeled



Reinforcement Learning

optimize behavior in a specific environment

e.g. chess game

Supervised Machine Learning Tasks

$x \in \mathbb{X}$ **- sample** $x = (x^1, \ldots, x^d)$ **- sample features known**

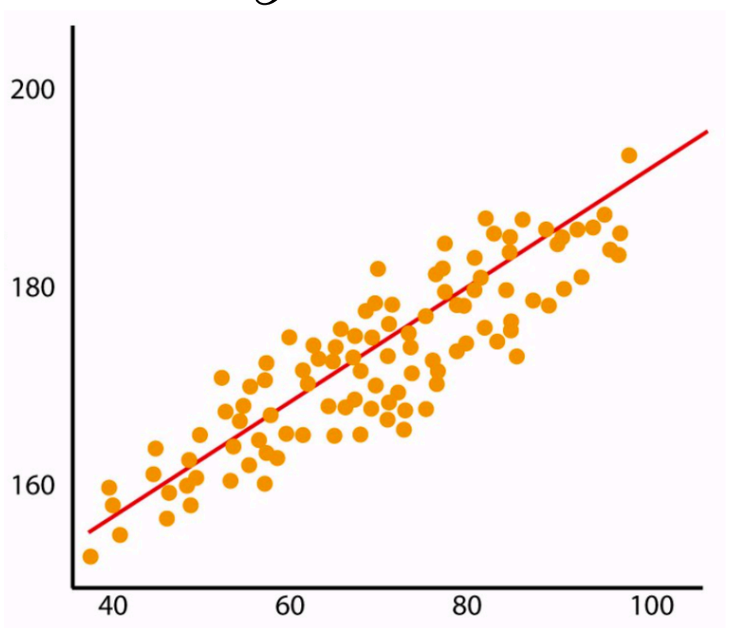
$y \in \mathbb{Y}$ **- answer (sample property which we want to predict)**

$X = (x_i, y_i)_{i=1}^{l}$ **training set is used to find a model** $y = a(x), a \in \mathbb{A}$

by minimising loss (cost) function $Q(a, X) : a(x) = argmin_{a \in \mathbb{A}} Q(a, X)$

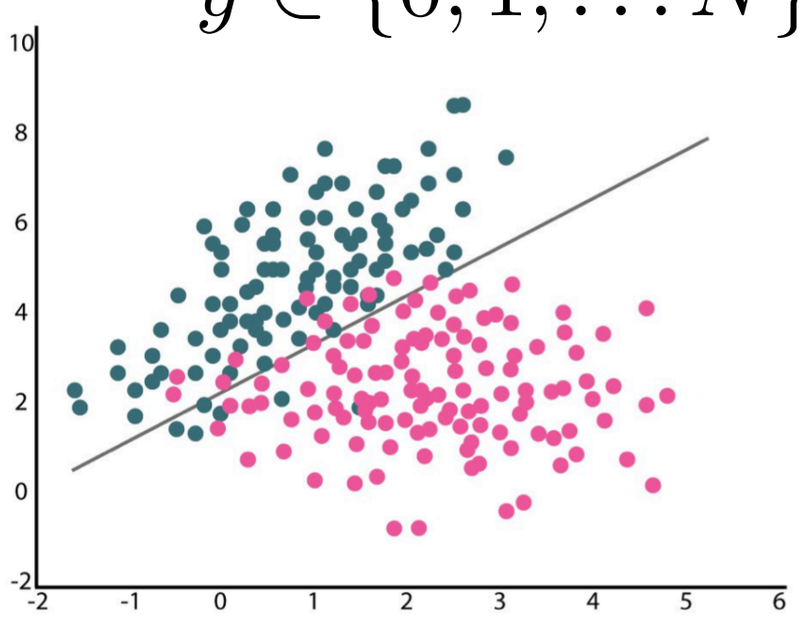
Regression: predict real value

$y \in \mathbb{R}$



Classification: predict class

$y \in \{0, 1, \ldots N\}$



Linear Regression

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x^j \quad \textbf{or} \quad a(x) = \sum_{j=0}^{d} w_i x^j = \langle w, x \rangle \quad \textbf{where} \quad x_0 \equiv 1$$

Choice for Q: $|a(x) - y|$ **- not smooth,** $(a(x) - y)^2$ **is ok**

$$Q(a, x) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2$$

$$Q(w, X) = \frac{1}{\ell} \|Xw - y\|^2 \to \min_w, \qquad X = \begin{pmatrix} x_{11} & \dots & x_{1d} \\ \dots & \dots & \dots \\ x_{\ell 1} & \dots & x_{\ell d} \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}$$

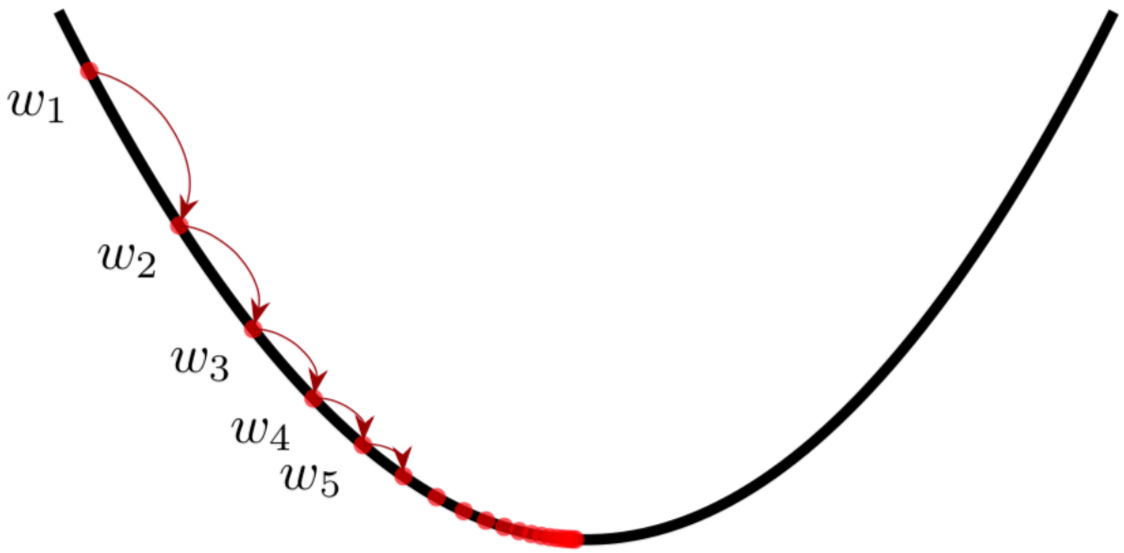
Analytic solution $w_* = (X^T X)^{-1} X^T y.$ **requires** $O(d^3)$ **operations**

Alternative: numerical optimisation

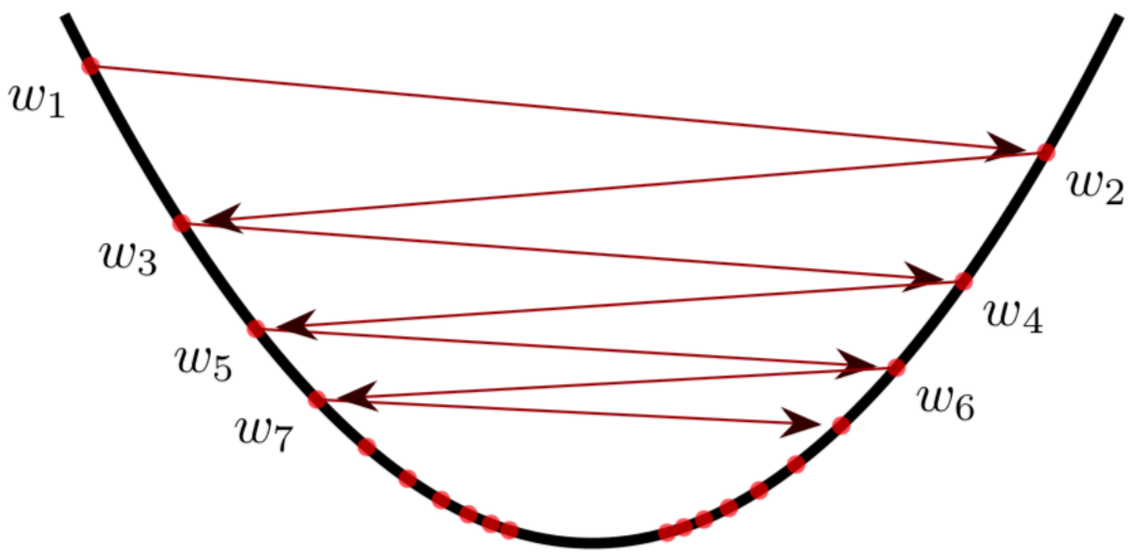
Linear Regression

optimisation with gradient decent method

- **start with some random or zero** w
- **at step** *t-1* **calculate loss function gradient** $\nabla_w Q(w, X) = \frac{2}{\ell} X^T (Xw - y)$
- **update weights** $w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, X)$ **and repeat**



small steps



big steps

Stochastic Gradient Decent

$$\nabla_w Q(w, X) = \frac{2}{\ell} X^T (Xw - y)$$

- expensive (requires calculation of loss term for each sample)

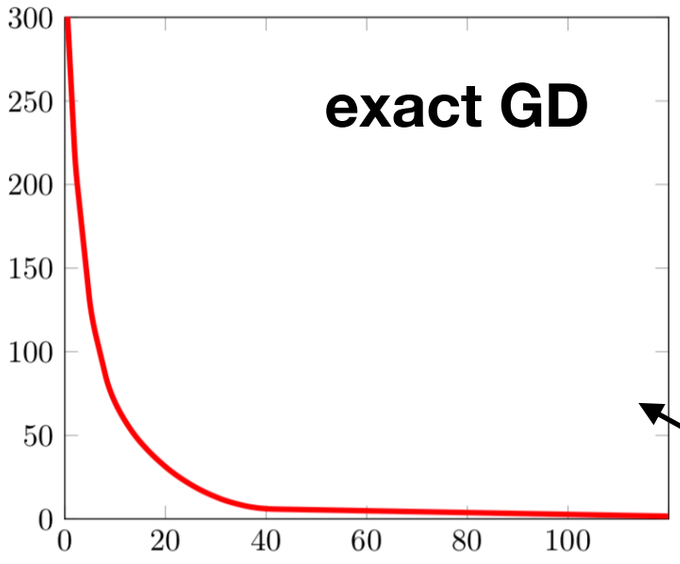
Alternative:

on each step select random sample $\{x_i\}$

adjust weights $w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, \{x_i\})$

repeat until e.g. $\|w^t - w^{t-1}\| < \varepsilon$

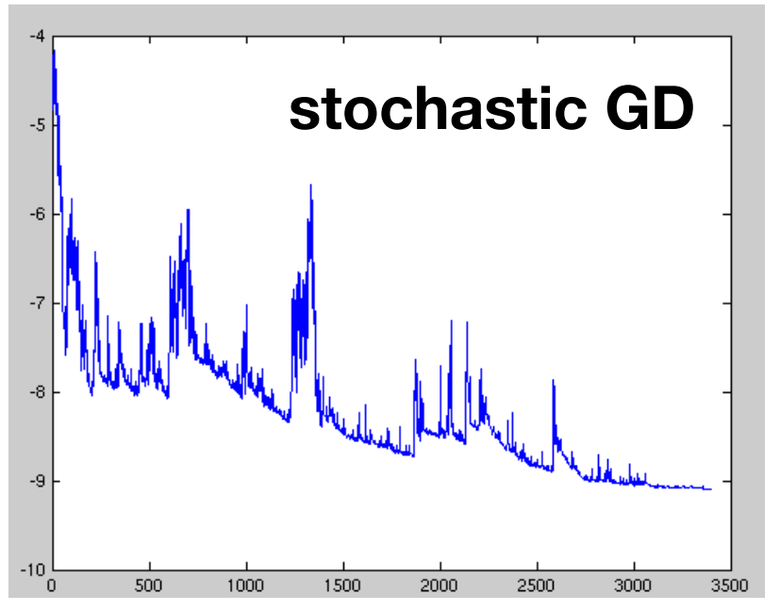
good for online training



Loss function dependence on iteration number

longer step

more steps



Mini-batch Gradient Decent

Exact gradient decent: $$w^t = w^{t-1} - \eta_t \nabla_w Q(w, \{x_1, \ldots x_l\})$$

Stochastic gradient decent: $$w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, \{x_i\})$$

Mini-batch gradient decent: **approximate GD using *m* samples**

$$w^t = w^{t-1} - \eta_t \nabla_w Q(w, \{x_{(t-1)m}, \ldots x_{tm}\})$$

Iterate through training entire training set in *l//m* steps (epoch) and then repeat

Linear Classification

$$\mathbb{Y} = \{-1, +1\}$$

replace regression model $a(x) = w_0 + \sum_{j=1}^{d} w_j x^j$

with $a(x) = \text{sign}\left(w_0 + \sum_{j=1}^{d} w_j x^j\right)$

Loss function:

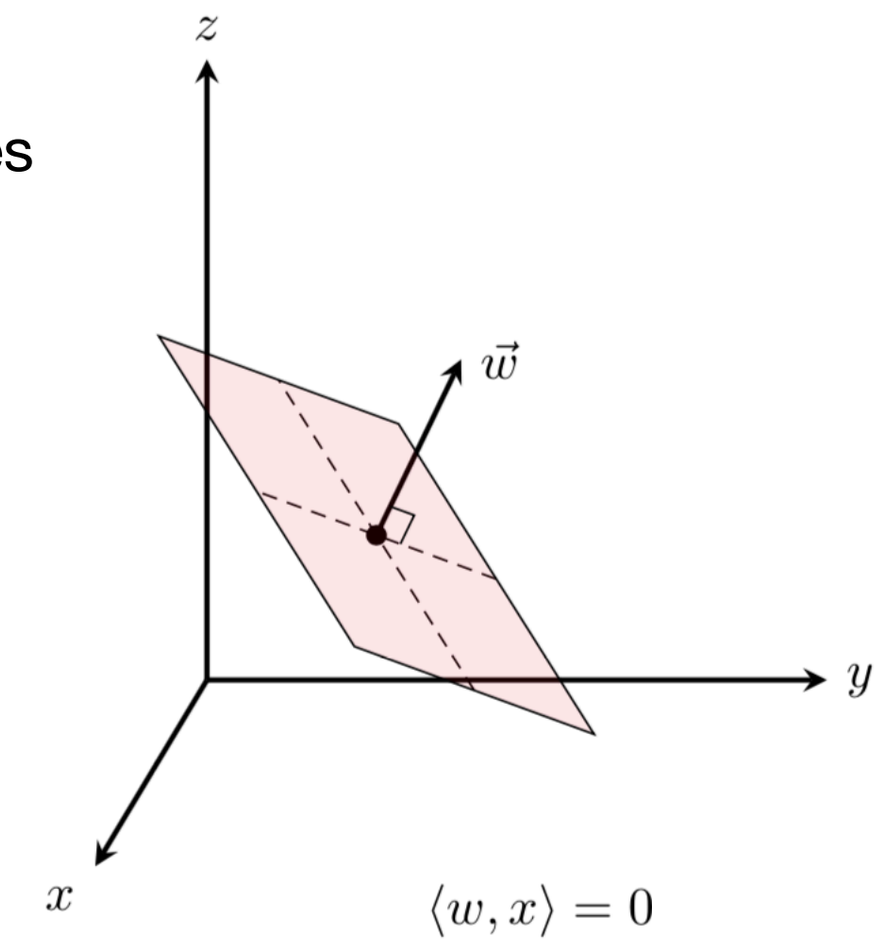
$$Q(a, x) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) \neq y_i]$$ **has step**

The hyperplane $\langle w, x \rangle = 0$ separates points of different classes

$y_i \langle w, x_i \rangle \propto$ **distance from plane**

$y_i \langle w, x_i \rangle < 0$ **for wrong answers**

$y_i \langle w, x_i \rangle > 0$ **for correct answers**



Possible smooth loss function:

$$-\frac{1}{l} \sum_{i=1}^{l} y_i \langle w, x_i \rangle$$

Practical part

ML and data analysis packages

- Python (numpy, pandas, matplotlib, sklearn, tensorflow, pytorch, keras, etc.)

- R (rpart, CARET, nnet, neuralnet, RSNNS, etc.)

- Matlab/Octave (LIBSVM, LIBLINEAR, NN, etc.)

- Java/scala (JDMP, DL4J, Spark MLlib, etc.)

- C++ (CERN ROOT, Boost.uBLAS, Dlib, stats++, tensorflow, torch, CNTK, Caffe, etc.)

Why Python

- Free
- Multiplatform
- Easy to learn
- Interpreted (can be executed interactively in shell)
- Lots of packages
- Among the most popular in data science

Running Python Samples

- **Preinstalled virtualbox Ububtu 16.04 environment (recommended)**
  - **download from http://tiny.cc/uz404y**
  - **virtualbox software needed and 64bit OS**
- **Manually install python environment (see next slide)**

To activate the python environment, run in terminal:
source init.sh

To start jupyter notebook:
jupyter notebook

The code samples will be uploaded on school web page

Python environment setup

List of libraries required:

tensorflow (ver. 1.12), matplotlib, pandas, keras, h5py, sklearn, jupyter, pillow, seaborn

Approximate instructions:

- - install miniconda for your platform from https://docs.conda.io/en/latest/miniconda.html (at this step you may choose either Python 2.7 or Python 3.7 based miniconda version)
- - open command shell window and create Python 3.5 virtual environment with the following shell commands
- conda create --name ML python=3.5
- conda activate ML
- pip install --upgrade pip
- pip install tensorflow==1.12 matplotlib pandas keras h5py sklearn jupyter pillow seaborn