Introduction to Machine Learning
Lecture I

Oleg Filatov [1] Stepan Zakharov [2]

[1]DESY [2]Novosibirsk State University

Outline

- Briefly about data
- Supervised Learning
  - General picture: objects, features, models
  - Types of problems
  - Model training/testing
  - Examples
- Semi-supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Briefly about data

Introduction

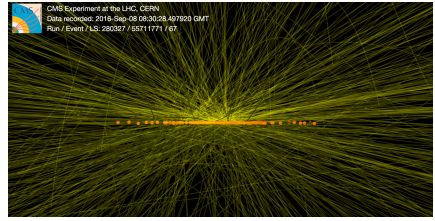
In the beginning was the Data:

- Information about goods and sales
- Data from subsystems of detector
- Subscribers database

Shall one question the Data:

- predict demand for this product?
- estimate energy of a particle?
- like or dislike the post?

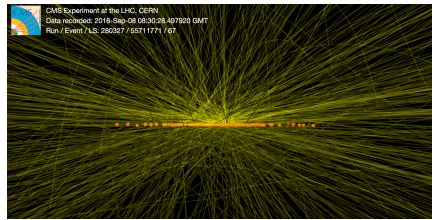
How can we answer these questions?



Introduction

In the beginning was the Data:

- Information about goods and sales
- Data from subsystems of detector
- Subscribers database



Shall one question the Data:

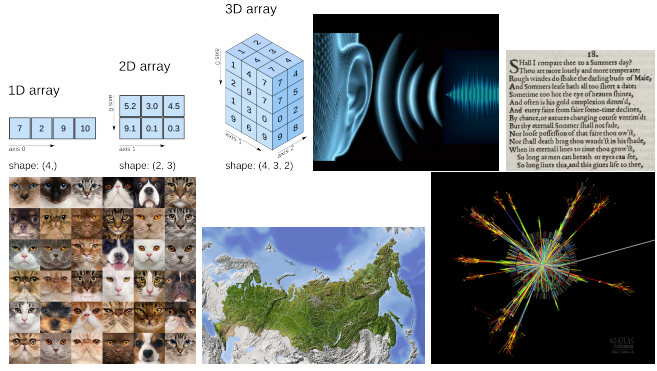
- predict demand for this product?
- estimate energy of a particle?
- like or dislike the post?

How can we answer these questions? → this lecture

Let's talk about data

Anything can be data:

1. Numbers
2. Text
3. Images
4. Sound
5. Geomap
6. Particle collisions
7. Knowledge
8. You name it



Can we use data out of the box?

- Most likely, no: garbage in, garbage out!
- Main drawbacks of raw data:
  - Not in ML comprehensive format
  - Can have missing values or might be noisy
  - Useless/lack of useful features

! Analysing data improves its understanding

Comprehensive format

more in the seminar!

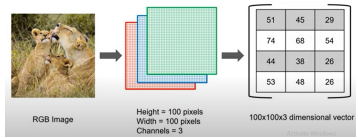
ML algorithms operate only with numerical sets, therefore:

- categorical data require one hot encoding (or any different method)
- text should also be vectorized
- pictures can be treated similarly

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

	good	movie	not	a	did	like
good movie	1	1	0	0	0	0
not a good movie	1	1	1	1	0	0
did not like	0	0	1	0	1	1

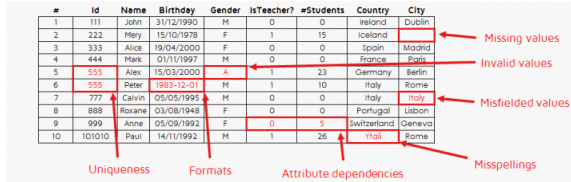


Data preprocessing

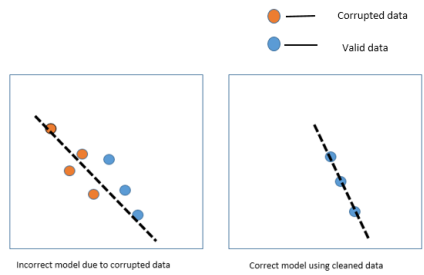
more in the seminar!

Main techniques:

- Imputation of missing data
- Data cleaning
- Handling outliers
- Shuffling
- Balancing classes
- Partitioning



#	Id	Name	Birthday	Gender	IsTeacher?	#Students	Country	City
1	111	John	31/12/1990	M	0	0	Ireland	Dublin
2	222	Mery	15/10/1978	F	1	15	Iceland	
3	333	Alice	19/04/2000	F	0	0	Spain	Madrid
4	444	Mark	01/11/1997	M	0	0	France	Paris
5	555	Alex	15/03/2000	A	1	23	Germany	Berlin
6	555	Peter	1983-12-01	M	1	10	Italy	Rome
7	777	Calvin	05/05/1995	M	0	0	Italy	Italy
8	888	Roxane	03/08/1948	F	0	0	Portugal	Lisbon
9	999	Anne	05/09/1992	F	0	5	Switzerland	Geneva
10	101010	Paul	14/11/1992	M	1	26	Ytali	Rome

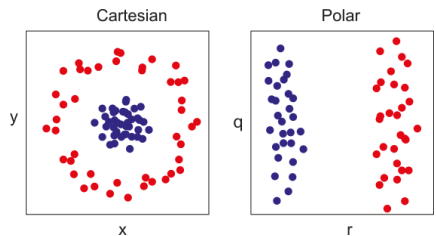


Feature engineering

more in the seminar!

- **Construction:** vectorization techniques + your domain knowledge
- **Transformation:** functions, scaling, binning, croping, resizing, blurring
- **Selection:** PCA, statistical tests, feature importance, hand-picking

Example: Using using Polar coordinates it's much easier to separate red dots from blue than using Cartesian coordinates:



Supervised Learning

Example

- Dataset of houses with corresponding areas and prices
- We know that price somehow depends on area of the house
- We want to know this relation to predict the price for a given house's area

In a formal way

- Let's introduce **object** $x \in X$, where X is a set of objects
- Objects are used to make predictions
- Output of the prediction is **answer** $y \in Y$, where Y is a set of answers
- There is an unknown **function** g: $X \to Y$ we are looking for

Supervised Learning is called "supervised" because there is a set of objects with **known answers** and we use this prior knowledge while approximating g: $X \to Y$ relation. Effectively, the **model is supervised by our prior knowledge.**

Problem statement

Given:

- Set of object that forms training sample $\{\boldsymbol{x_1} \dots \boldsymbol{x_N}\} \subset X$
- Set of answers $y_i = g(\boldsymbol{x_i}),\ \{y_1 \dots y_N\} \subset Y$

Find:

- a: $X \to Y$ – algorithm that approximates g on the whole set X

Questions:

- Why do we need a? why not directly find g?
- What does ″approximates″ mean?
- How to find this algorithm?

What is algorithm?

- Let $a(\boldsymbol{x}, \boldsymbol{\theta})$ be a function that depends on object $\boldsymbol{x}$ and vector of parameters $\boldsymbol{\theta}$.
- This function belongs to algorithm set $A = \{a(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Theta\}$, where Θ is a parameter space
- By definition of g: $\begin{bmatrix} g(\boldsymbol{x_1}) \\ \cdots \\ g(\boldsymbol{x_N}) \end{bmatrix} = \begin{bmatrix} y_1 \\ \cdots \\ y_N \end{bmatrix} = \boldsymbol{y}$, the exact expression for g is unknown
- Let's try to approximate g using the model a by estimating "the best" $\boldsymbol{\theta} \equiv \hat{\boldsymbol{\theta}}$
- Then $\begin{bmatrix} a(\boldsymbol{x_1}, \hat{\boldsymbol{\theta}}) \\ \cdots \\ a(\boldsymbol{x_N}, \hat{\boldsymbol{\theta}}) \end{bmatrix} = \begin{bmatrix} y'_1 \\ \cdots \\ y'_N \end{bmatrix} = \boldsymbol{y'}$
- Vector $\boldsymbol{y'}$ approximates $\boldsymbol{y}$ if the procedure of $\boldsymbol{\theta}$ estimation is correct

What does it mean y' approximates y?

- We can define a function on the Y space that returns the state of closeness for two vectors
- This function is usually called a **loss function**, since it also describes how much we "lose" when mispredict y with poor predictions of $a(x)$
- Loss function $\mathcal{L}(a(x), y)$ gives error estimation for the particular object

What are typical loss functions?

- $\mathcal{L}\big(a(x), y\big) = \big[a(x) \neq y\big]$ – error indicator for the classification
- $\mathcal{L}\big(a(x), y\big) = \big|a(x) - y\big|$ – absolute error for the regression
- $\mathcal{L}\big(a(x), y\big) = \big(a(x) - y\big)^2$ – quadratic error also for the regression

Optimisation problem

Note: Loss function returns the error only for a single object.

- Define **empirical risk** for a whole set of objects: $Q(a, X, Y) = \sum_{i=1}^{N} \mathcal{L}\big(a(\boldsymbol{x_i}), y_i\big)$
- $Q(a, X, Y)$ is usually called "loss function" too because there is no need to use directly $\mathcal{L}\big(a(\boldsymbol{x}), y\big)$ for single object
- Optimal algorithm $\hat{a}$ minimizes $Q(a, X, Y)$ on the training set $[X, Y]$

$$\hat{a} = \arg\min_{a \in A} Q(a, X, Y)$$

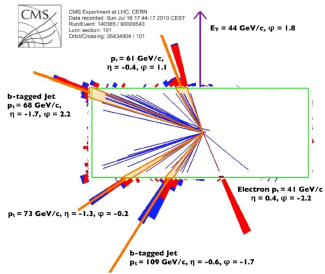
This procedure is called **training**

Objects and features

- Object can be described by a vector of variables, aka **features**
- Let's define f_j as a function $f_j(\boldsymbol{x})$ that takes object $\boldsymbol{x}$ for the input and returns j–th feature
- $f_j(\boldsymbol{x}) \in D_j$ — set of destinations of j–th feature
- This defines its **type**:
  - $D_j = \{0, 1\} \Rightarrow$ binary feature
  - $D_j \subset \mathbb{Z} \Rightarrow$ categorical feature
  - $D_j \subset \mathbb{Z}$ + total order $\Rightarrow$ ordinal feature
  - $D_j \subset \mathbb{R} \Rightarrow$ numerical feature

Objects and features — Example

- Object $\leftrightarrow$ Event at the detector
- Feature vector $\leftrightarrow$
  $[\text{HasJets}, \text{Type of lepton}, [p_T, \eta, \varphi, m](\text{lepton}), \text{NTracks}]$
- HasJets $= \{0, 1\}$ $\leftrightarrow$ binary feature
- Type of lepton $= \{\text{muon}, \text{electron}, \text{tau.}\}$ $\leftrightarrow$ categorical feature
- NTracks $= \{1, 2, \ldots\}$ $\leftrightarrow$ ordinal feature
- p_T, η, φ, m $\leftrightarrow$ numerical features

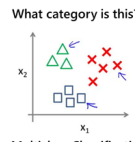


Supervised Learning types

Answer set Y can be different. Its domain mainly defines the problem to work with:

- $Y = \{0, 1\} \Rightarrow$ **binary classification**
  Ex: $Y = \{\text{signal, background} \ldots\}$
- $Y \subset \mathbb{Z} \Rightarrow$ **multiclass classification**
  Ex: $Y = \{\text{electron, muon, tau,} \ldots\}$
- $Y \subset \mathbb{Z}$ + total order $\Rightarrow$ **ranking**
  Ex: $Y = \{0, 1, \ldots, N\}$
- $Y = \mathbb{R} \Rightarrow$ **regression**
  Ex: $Y = \text{p}_\text{T}$ of jets

Binary Classification



Multiclass Classification

Regression

Example — linear models

more in the next lecture!

Classification

- $Y = \{0, 1\}$
- N objects with K numeric features: $D = \mathbb{R}^K$
- $a(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{K} f_j(\boldsymbol{x}) \cdot \boldsymbol{\theta_j} = \langle \boldsymbol{x}, \boldsymbol{\theta} \rangle$
- $\mathcal{L}\big(a(\boldsymbol{x_i}), y_i\big) = \big[a(\boldsymbol{x}) \neq y_i\big]$
- $Q(a, X, Y) = \sum_{j=1}^{N} \mathcal{L}\big(a(\boldsymbol{x_i}), y_i\big)$

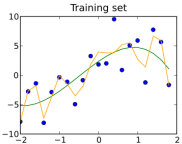
Regression

- $Y = \mathbb{R}$
- N objects with K numeric features: $D = \mathbb{R}^K$
- $a(\boldsymbol{x}, \boldsymbol{\theta}) = \langle \boldsymbol{x}, \boldsymbol{\theta} \rangle$
- $\mathcal{L}\big(a(\boldsymbol{x_i}), y_i\big) = \big(a(\boldsymbol{x}) - y_i\big)^2$
- $Q(a, X, Y) = \sum_{j=1}^{N} \mathcal{L}\big(a(\boldsymbol{x_i}), y_i\big)$

At the final step both problems transfer into the procedure of empirical risk minimization.

Measuring model performance

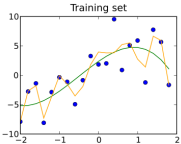
- After training we want to understand the quality of the model
- Therefore we need something like:



$\Rightarrow$ Number

Measuring model performance

- After training we want to understand the quality of the model
- Therefore we need something like:

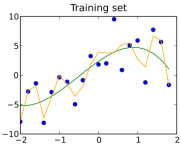


$\Rightarrow$ Number

- Since $\mathcal{L}(a(\boldsymbol{x}), y)$ is an error estimation

Measuring model performance

- After training we want to understand the quality of the model
- Therefore we need something like:

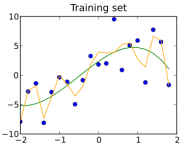


$\overset{\text{metric}}{\Rightarrow}$ Number

- Since $\mathcal{L}(a(\boldsymbol{x}), y)$ is an error estimation
- Then it can be used as **metric**

Measuring model performance

- After training we want to understand the quality of the model
- Therefore we need something like:



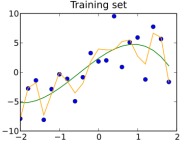
$\overset{\text{metric}}{\Rightarrow}$ Number

- Since $\mathcal{L}(a(\boldsymbol{x}), y)$ is an error estimation
- Then it can be used as **metric**

- Which data to use for measurements?

Measuring model performance

- After training we want to understand the quality of the model
- Therefore we need something like:

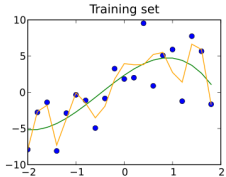


$\overset{\text{metric}}{\Rightarrow}$ Number

- Since $\mathcal{L}(a(\boldsymbol{x}), y)$ is an error estimation
- Then it can be used as **metric**

- Which data to use for measurements?

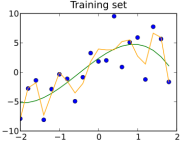
- Well, we can use the training dataset:



- Then it might happen that $\mathcal{L}_{\text{train}} \to 0$

Measuring model performance

- After training we want to understand the quality of the model
- Therefore we need something like:

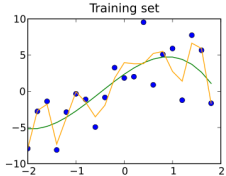


$\overset{\text{metric}}{\Rightarrow}$ Number

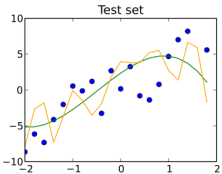
- Since $\mathcal{L}(a(\boldsymbol{x}), y)$ is an error estimation
- Then it can be used as **metric**

- Which data to use for measurements?

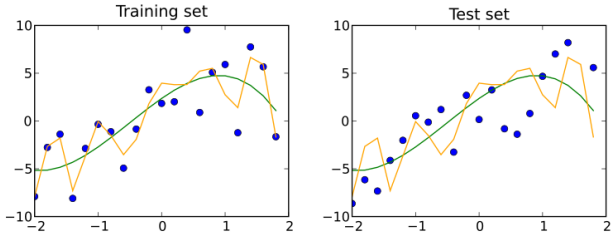
- Well, we can use the training dataset:



- Then it might happen that $\mathcal{L}_{\text{train}} \to 0$
- But on unseen data $\mathcal{L}_{\text{test}}$ will be obscenely large:



Observations

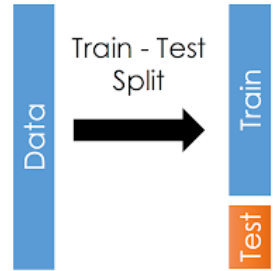


- If on unseen data the model performs bad $\Rightarrow$ it has **low generalization capability**
- This is called **overtraining (overfitting)** → next lecture
- To identify this, we applied the model to **unseen data**

Where to get new data?

train/test split

- Divide initial dataset into two parts: train and test
- Train on a train set, then measure performance on a test set
- ! Test dataset should not be used in training
- ! Split between train and test should be uniform



Problem: results depend on the choice of splitting

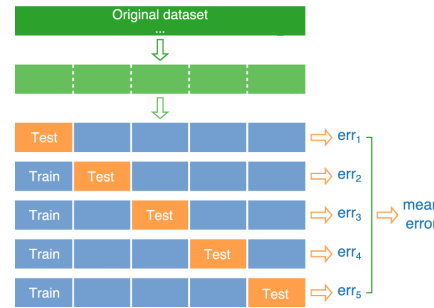
→ how to check model performance on the whole dataset w/o split dependance?

Where to get new data?

cross-validation

- Split dataset into K folds
- Train on K-1 folds, test on remaining one
- Repeat for different folds
- Aggregate metrics to get mean and variance

✓ Provides interval error estimation
✓ More robust to train–test split
× Requires computational resources
× Use only a part of dataset for training



Inference

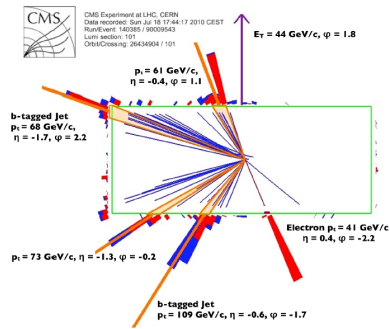
Once the model has been trained and tested, we can start applying it in real-life settings

SL examples

- **Area:** medical diagnostics
- **Objects:** patients
- **Features:**
  - Binary: headache, cough, fever, results of tests
  - Categorical: sex, condition severity, date of last check-up
  - Numerical: age, heart rate, blood pressure, hemoglobin level
- **Problems:**
  - Classification: establishing diagnosis, choosing treatment scheme
  - Regression: calculation of drug dose, prediction of metabolism time
  - Ranking: forming list of high risk patients

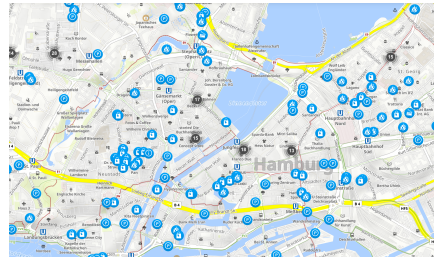
SL examples

- **Area:** HEP
- **Objects:** events in the detector
- **Features:**
  - Binary: jet from b-quark, signal presence in subsystems
  - Categorical: number of jets, number of tracks, lepton type
  - Numerical: pT, η, φ, m, E, $\Delta R(\text{jet}, \gamma)$ of particles
- **Problems:**
  - Classification: event is signal or background? particle type?
  - Regression: jet energy, invariant mass of lepton pair
  - Ranking: list of candidates for primary vertices



SL examples

- **Area:** marketing
- **Objects:** (product, shop)
- **Features:**
  - Binary: parking, ATM in shop
  - Categorical: type of product, opening hours
  - Numerical: shop location, amount in stock, price
- **Problems:**
  - Categorization: need a discount?
  - Regression: set the product price
  - Ranking: top 5 relevant products



Semi-supervised Learning

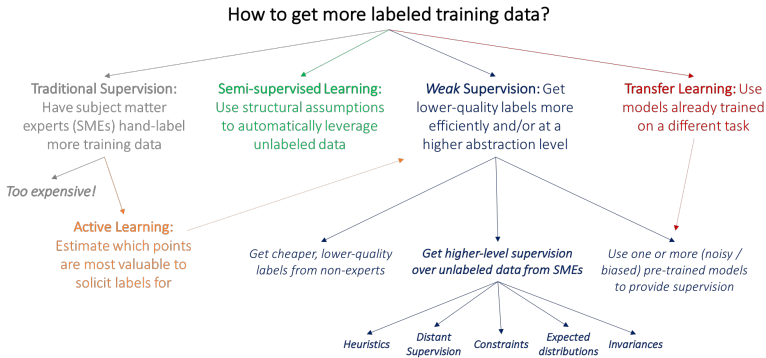
What if we don't have labels?

cs231n slides

- Labels might be imperfect or even don't exist
- We can't train a good model w/o properly labelled and large enough dataset
- It is hard to get more properly labelled data
  Ex: How much would it cost for a cardiologist to label thousands of MRIs?
- Can we still train models in these settings?

Indication: Chest pain. Findings: Mediastinal contours are within normal limits. Heart size is within normal limits. No focal consolidation, pneumothorax or pleural effusion. Impression: No acute cardiopulmonary abnormality.

What if we don't have labels?

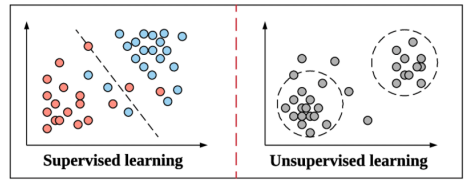


Unsupervised Learning

What if there is **no labels at all**?

We just have *some* data and we want to:

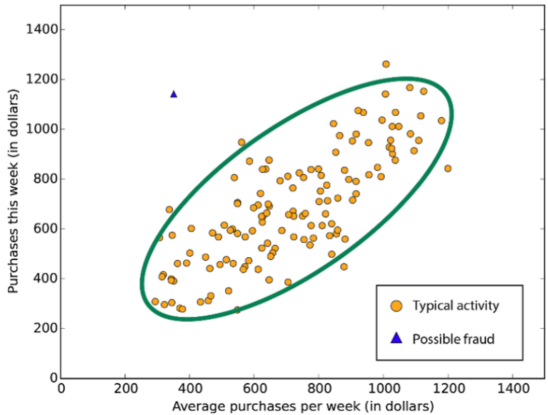
- Study its **structure**
  - identify novelties/outliers
  - cluster objects
  - estimate densities
  - reduce dimensionality
- Generate it



→ no prior knowledge ⇒ learning is **unsupervised**

Unsupervised Learning

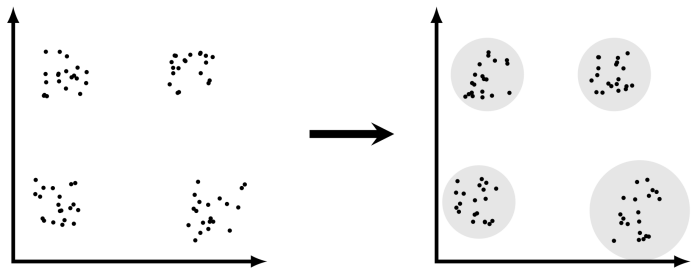
identify anomalies



link

Unsupervised Learning

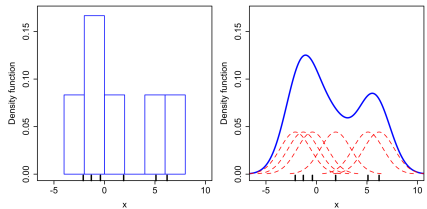
cluster objects



link

Unsupervised Learning — estimate densities

- Sometimes we have some data and don't know the underlying analytical **probability density function** (pdf)
- Which would be nice to know for:
  - visualizing data
  - understanding its shape
  - performing statistical inference
  - doing analytical calculations
  - sampling more data from it
- In HEP **histograms** have been used since the beginning of times
- But they induce information loss withing the bin and aren't nicely differentiable

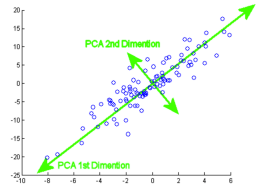


amazing visuals

→ Kernel Density Estimation (KDE)

Unsupervised Learning

reduce dimensionality



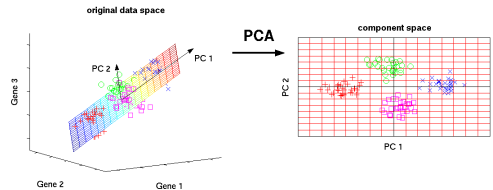
- might need to reduce number of features
- simply removing them might lead to information loss
- is there a clever way?

→ Principal Component Analysis (PCA)

→ Matrix Factorization

→ t-SNE

→ Autoencoder



awesome playground

Unsupervised Learning

generate data

1810.08771

playground

Unsupervised Learning

generate data in HEP

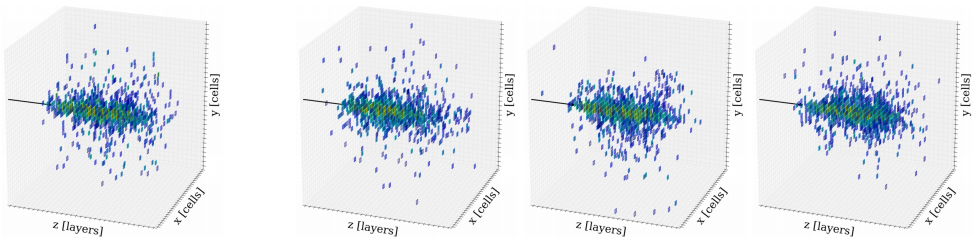


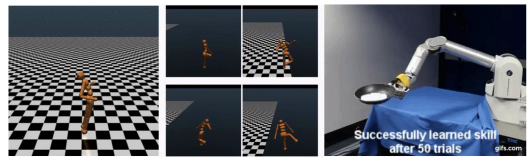
Fig. 5 Examples of individual 50 GeV photon showers generated by GEANT4 (left), the GAN (center left), WGAN (center right), and BIB-AE (right) architectures. Colors encode the deposited energy per cell.

Reinforcement Learning

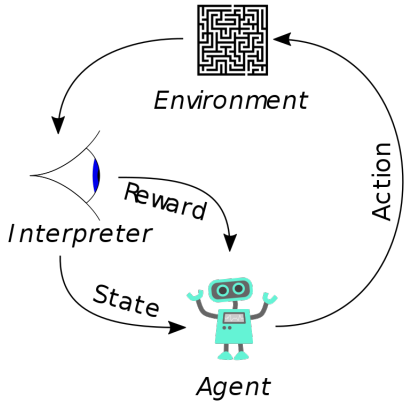
What if there's need for action?

- Sometimes we might want to model **behaviour**
  Ex: consumer modeling
- That is, to teach a machine to **act in a given world**
  Ex: robotics, trading, game playing

→ Reinforcement Learning makes this AI dream come true



Reinforcement Learning — in a nutshell



more in cs234 course

Summary

- All ML problems start from:
  - Data preprocessing
  - Feature engineering
- Type of model/approach is defined by our intentions:
  - **SL:** seek for a **predictive model**: for a given object based on its **features** returns the **answer**
    - classification
    - regression
    - ranking
  - **UL:** look for insights into the **structure of data**
  - **RL:** want to build an agent capable of **acting in the given environment**
- Model is constructed through the **training process** involving **loss function optimization**
- To evaluate **model performance** and check for **overfitting** we compute **metrics**
- For that we apply model to **unseen data** using **train/test and cross-validation** strategies
- Once we are sure that the model is reasonable we can **deploy it in production**

Summary

- General pipeline