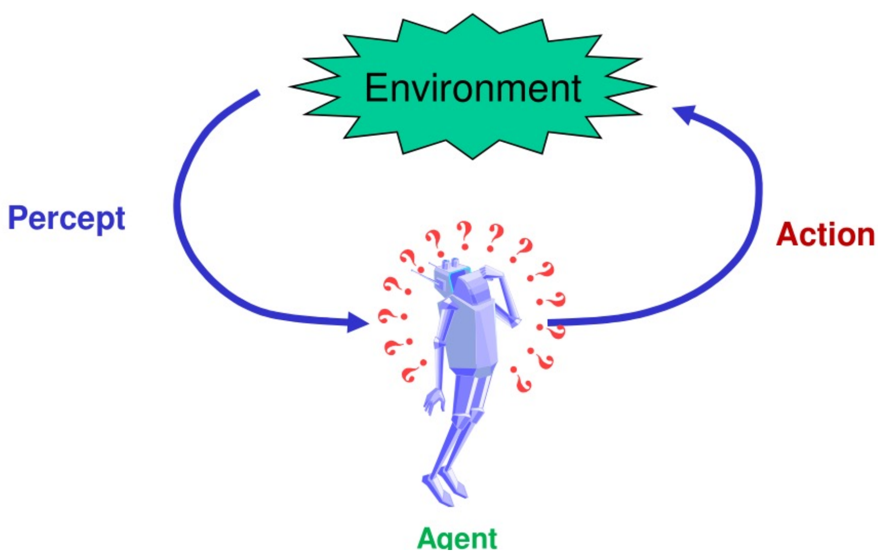
Introduction to machine learning

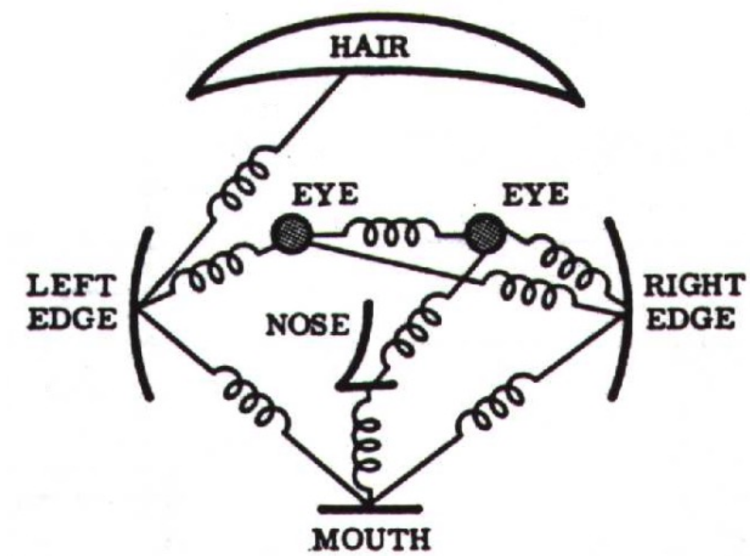
Artificial intelligence

- Emulate human intelligence: how?



Artificial intelligence

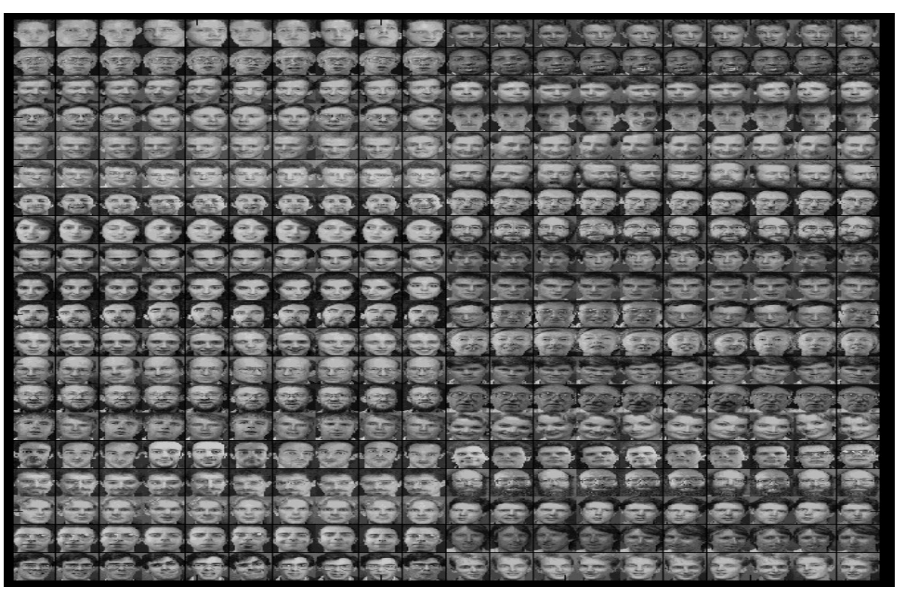
- Encode human knowledge



(doesn't work)

Artificial intelligence

- Use large amounts of examples



2

(it works!)

Machine learning

- Extract information (*patterns*) from data
- Applications:
  - Predictions (e.g. energy demand, sales)
  - Recommendations (e.g. Netflix)
  - Anomaly detection (e.g. intrusions, virus mutations)
  - Classification (e.g. image recognition, medical diagnoses)
  - Ranking (e.g. Google search)
  - Decision making (e.g. AI, robotics, games, trading)

5

Machine learning

- When it is useful
  - Lack of human support (e.g. robot on Mars)
  - Difficulty in encoding human experience (e.g. speech recognition, vision, language)
  - Too dimensionally-large problems (web page ranking, personalized advertising)

6

Success of machine learning

- Datasets
  - Image datasets in the '90s: 100 images
  - Image datasets in 2012: 1,000,000 images
  - Key factors: storage, data collection (image search), validation (crowdsourcing)
- General-purpose GPU
  - 100x speed-up w.r.t. CPUs
  - Training times of complex models: 1-2 weeks (on GPUs!)

7

Dataset

- Collection of data
  - Observation/samples/features: $\{x_i\}$
    - Potentially noisy, correlated, redundant
  - Label/target/output: $\{y_i\}$
- Examples:
  - Images + type of depicted content
  - Medical exams + positive/negative diagnosis
  - Past stock market trend + future stock market trend
  - Points $(x_1, x_2, \dots, x_n)$ of a function and values $y = f(x_1, x_2, \dots, x_n)$

8

Model

- Objective: find the law that associates x to y
- **Model**: family of functions, defined by a set of **parameters**
- Algorithm design:
  – Choose the model according to our knowledge/expectation of the dataset
  – Choose the model's parameters: **training**

9

Non-parametric models

- E.g. nearest-neighbor
  – Dataset: $\{x_i, y_i\}$
  – Unknown input z
  – Associate to z the value y_j such that:

$$|z - x_j| = \min_{x_i} |z - x_i|$$

10

Learning modalities

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Learning modalities

- Supervised learning:
  - Model is trained from $\{x_i, y_i\}$ pairs
  - Pros:
    - Most common, most studied, simplest
    - Training quality monitoring
  - Cons:
    - «Overfit» risk: the model learns the dataset **too well**
    - Dataset *annotation* (collecting $\{y_i\}$)

Learning modalities

- Unsupervised learning:
  - Model is trained from $\{x_i\}$
  - Pros:
    - Uncovers hidden dynamics (unknown even to experts)
    - Annotation is not necessary
  - Cons:
    - We may not know the set of possible values for y_i
    - Current techniques not satisfactory
  - E.g. *clustering*, *anomaly detection*

Learning modalities

- Reinforcement learning:
  - State: x_t, action: a_t
  - *Reward*: r_t, new state: x_{t+1}
  - No dataset, but an *environment simulator*
  - Pros:
    - Similar to human learning
    - Annotation is not necessary
  - Cons:
    - Hard to establish link between past actions and rewards
    - Hard to define reward
  - E.g. games

Supervised learning: classification

- $y_i \in \{C_1, C_2, \dots, C_k\}$
- *Categorical/discrete* labels
- **Binary** classification: $k = 2$
  - E.g. positivity/negativity of medical diagnosis
  - E.g. identify static/moving pixels in video
- **Multiclass** classification: $k > 2$
  - E.g. identify an object in an image

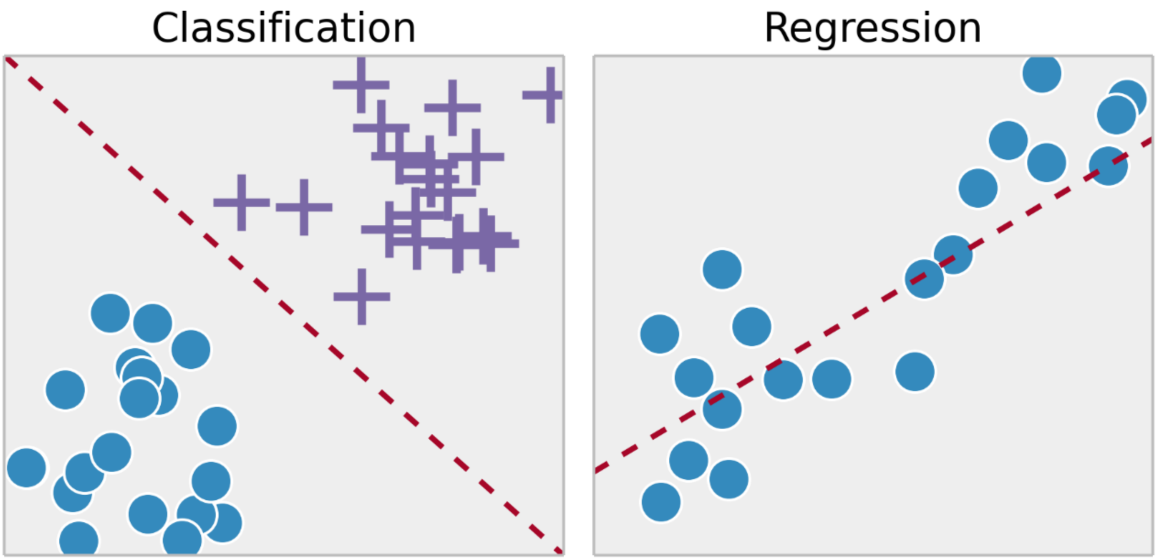
15

Supervised learning: regression

- $y_i \in \mathbb{R}$
- *Continuous* labels
- Examples:
  - Predict stock market prices
  - Predict user age on YouTube
  - Predict home energy consumption from temperature, sensors, weather, etc.

16

Classification vs regression



- Classification **separates**
- Regression **approximates**

Training

- How to choose model parameters?
- **Cost/loss/error/energy** function
  - Evaluates how good a set of parameter values is
  - Lower value → better parameters
- Analytical approach:
  - Compute θ such that $\frac{\partial L}{\partial \theta} = 0$
  - Usually intractable
- Iterative approach:
  - Initialize parameters θ_0
  - Repeat until convergence:
    - Evaluate loss $L(\theta_i)$
    - Re-compute $\theta_{i+1} \leftarrow \mathrm{update}(\theta_i)$
- A «swipe» on the whole dataset is called **epoch**

Dataset splits

- **Training set**: fraction of the dataset used for training
  - The training algorithm uses these data only
  - Model/parameter choices must be conditioned on these data only
- **Test set**: fraction of the dataset used for verifying the model
  - The training algorithm and the designer **MUST NOT** see these data!
  - Evaluates how good the trained model in reality: **generalizability**

Dataset splits

- Why can't we trust the training set?
- **Overfit**: over-adaptation of the model on the training set, causing loss of generalizability
  - Reasons: model too complex, dataset too small
- **Unbalance**
  - Example: your dataset has 90 dog images and 10 cat images
  - Your model, for any image, predicts a dog
  - Accuracy of your model: 90% (?)

How to detect overfit

- The training algorithm can only use the training set
- The test set can be used at the end of training
- The model may have overfitted, but we only find out after evaluating on the test set
  - And we can do nothing about it, because we can't use the test set to improve our model
  - **Problem**: how to detect overfit?

How to detect overfit

- **Solution**: divide the original training set into a smaller training set and a **validation set**
- The new training set is used to update parameters
- The validation set is used during training to detect overfit
  - It's fine, it was part of the original training set
- The test set is only used at the end of training

Where we'll go from here

- Linear regression and classification
- Neural networks
  - Mostly for classification
- Convolutional neural networks
  - Mainly, 2D data processing (e.g. images)
- Recurrent neural networks
  - Sequential data processing (e.g. text, audio)

What you need to know

- Elements of linear algebra: scalar product, matrix multiplications
- Elements of calculus: derivatives and gradients
- Python