MAC 0460 / 5832
Introduction to Machine Learning

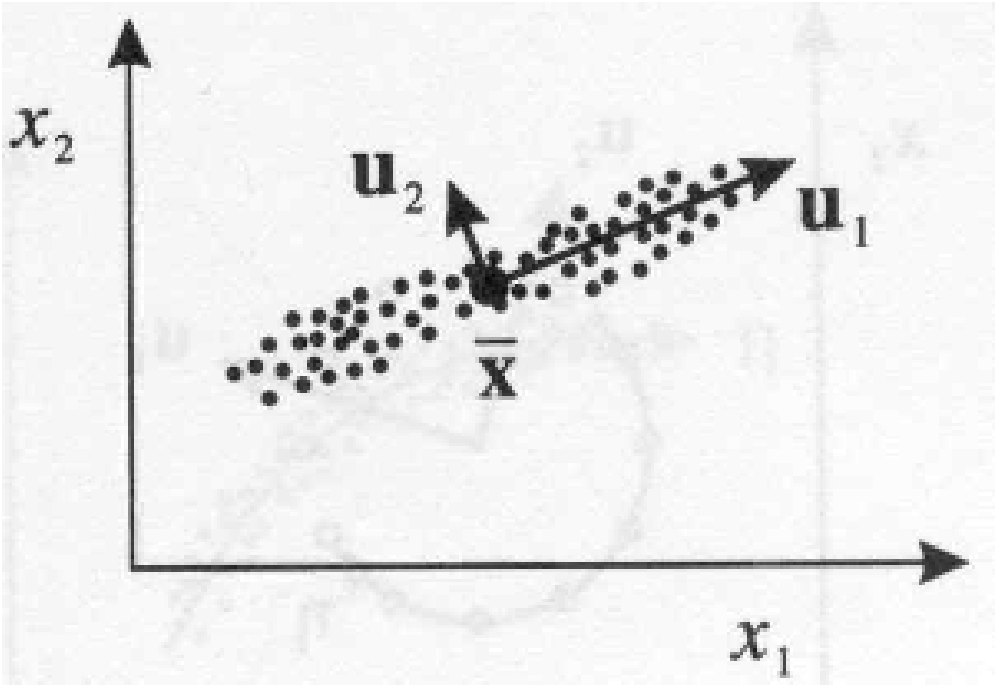
20 – From ML to DL

IME/USP (28/06/2021)

Data simplification / reduction

PCA – Principal component analysis

PCA – Intuition



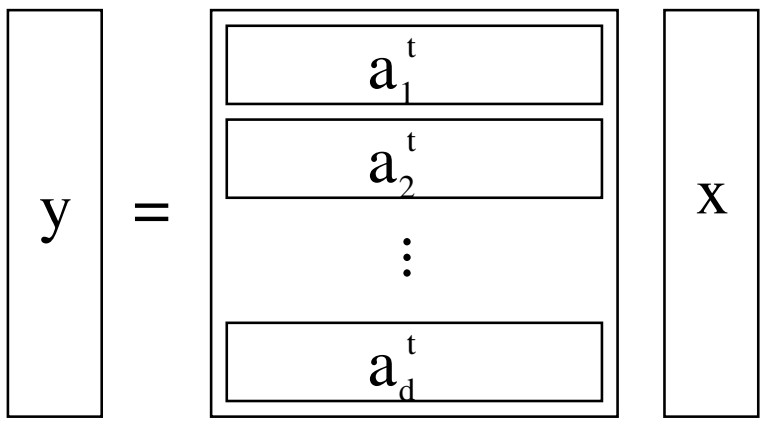
Change the basis, to align the axis to the directions with largest dispersion of the points.

PCA

The goal is to find a linear transformation $\mathbf{y} = \mathbf{A}\mathbf{x}$ that

(1) maximizes the variance of the transformed variables $y_i = \mathbf{a}_i^T \mathbf{x}$
(y_i can be seen as a projection of $\mathbf{x}$ on $\mathbf{a}_i$)

(2) any two vectors $\mathbf{a}_i$ and $\mathbf{a}_j$ are orthogonal each other



The desired linear transformation can be obtained from the eingenvectors of the covariance matrix of D

Covariance matrix: Example for $d = 3$

$$S = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{12} & s_{22} & s_{23} \\ s_{13} & s_{23} & s_{33} \end{bmatrix}$$

Diagonals: variance of x_1, x_2, and x_3

Remaining elements: covariance

$$s_{ij} = s_{ji} = \frac{1}{N-1} \sum_{n=1}^{N} (x_i^{(n)} - \overline{x}_i)(x_j^{(n)} - \overline{x}_j)$$

S : covariance matrix of dataset D

The covariance matrix S is symmetric and positive semi-definite ($\mathbf{x}^t\, S\, \mathbf{x} > 0, \forall \mathbf{x} \neq \mathbf{0}$)

Let $\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0$ be the eingenvalues of S (they exist and are all real, non-negative and distinct since S is a symmetric positive definite matrix)

Let $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_d}$ be the unit eigenvectors of S corresponding to the eingenvalues $\lambda_1 > \lambda_2 > \ldots > \lambda_d$

Let M be the matrix whose columns are the eingenvectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_d}$ of S

S is diagonalizable : $\Lambda = M^{-1}SM$

If we set $\mathbf{y} = M^{-1}\mathbf{x}$, we have:

- $y_i = \mathbf{e}_i^t\mathbf{x}$ is the i-th **principal component**
- $Var(y_i) = \lambda_i$
- $Cov(y_i, y_j) = 0, \forall j < i$

PCA keeps the total variance

$$\begin{aligned}
\sum_{i=1}^{d} Var(x_i) &= s_{11} + s_{22} + \cdots + s_{dd} \overset{(1)}{=} tr(S) \\
&\overset{(2)}{=} tr(M\Lambda M^{-1}) \overset{(3)}{=} tr(M\Lambda M^{T}) \overset{(4)}{=} tr(\Lambda M^{T} M) \\
&\overset{(5)}{=} tr(\Lambda M^{-1} M) \overset{(6)}{=} \lambda_1 + \lambda_2 + \cdots + \lambda_d \overset{(7)}{=} \sum_{i=1}^{d} Var(y_i)
\end{aligned}$$

(1) s_{ii} is the variance of x_i; variances are in the diagonal of S

(2) $\Lambda = M^{-1}SM \implies S = M\Lambda M^{-1}$

(3,5) M is orthogonal; hence $M^{-1} = M^T$

(4) $tr(AB) = tr(BA)$

(6) the diagonal of Λ contains the eigenvalues of S

(7) property (previous page)

Interpretation

Total variance is equal to the sum of the eigenvalues

Thus, the **percentage of total variance explained by the k-th principal component** is

$$\frac{\lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_d}$$

Dimensionality reduction using PCA

How many components one should choose?

Choose the d' principal components such that

$$\frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^{d} \lambda_i} > T$$

In general, $T = 0.90$ or $T = 0.95$

In many cases, a few components is sufficient to explain most of the total variance.

What is the meaning of the selected components ?

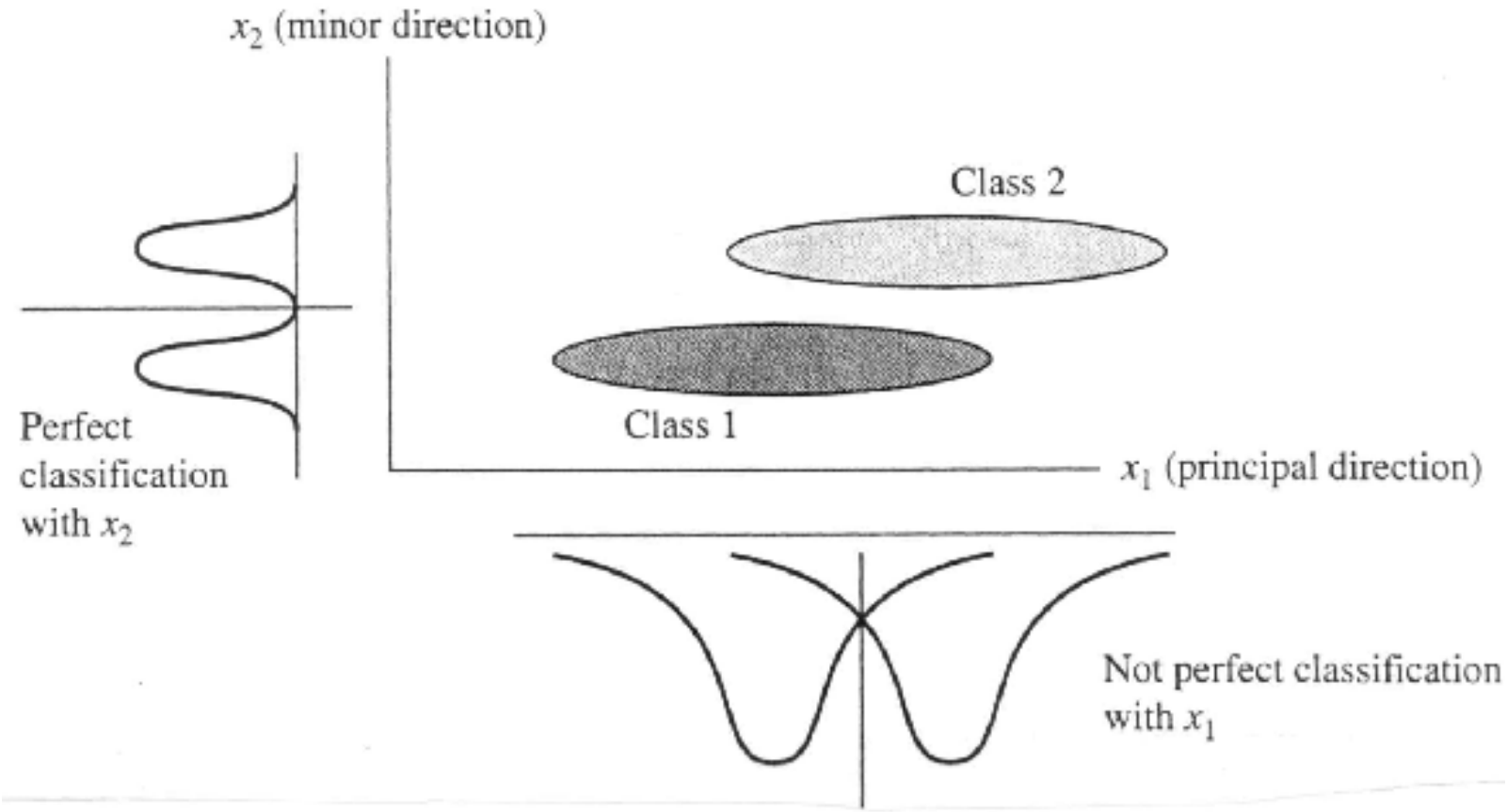
How do they relate to the original variables ?

We can examine the coefficients of the original variables in the projection $y_i = \mathbf{e}_i^t \mathbf{x}$

The magnitude of the Coefficients indicate the relevance or how each original variable contributes to the selected components.

Dimensionality reduction using PCA

In classification problems, not necessarily good



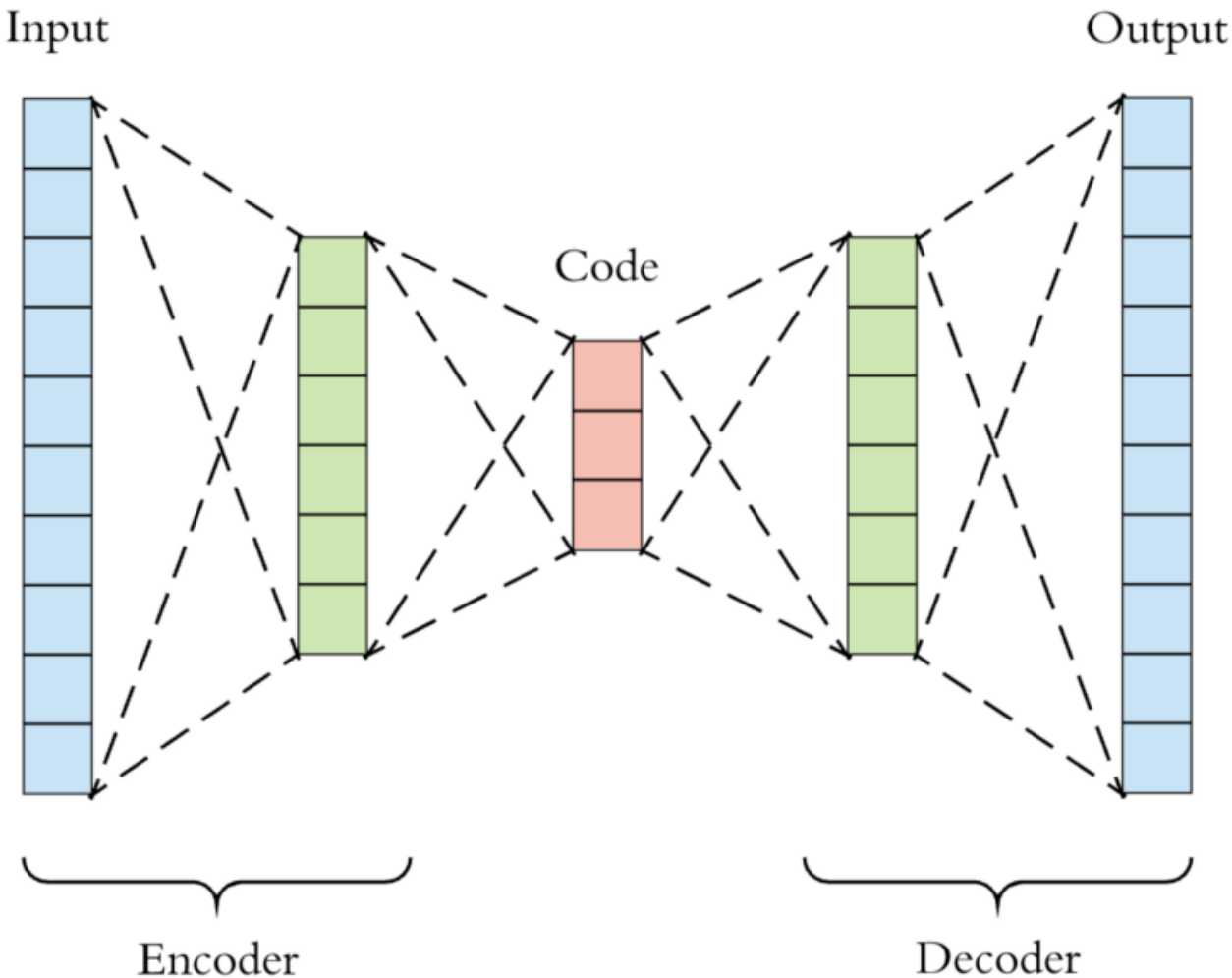
Some references for PCA

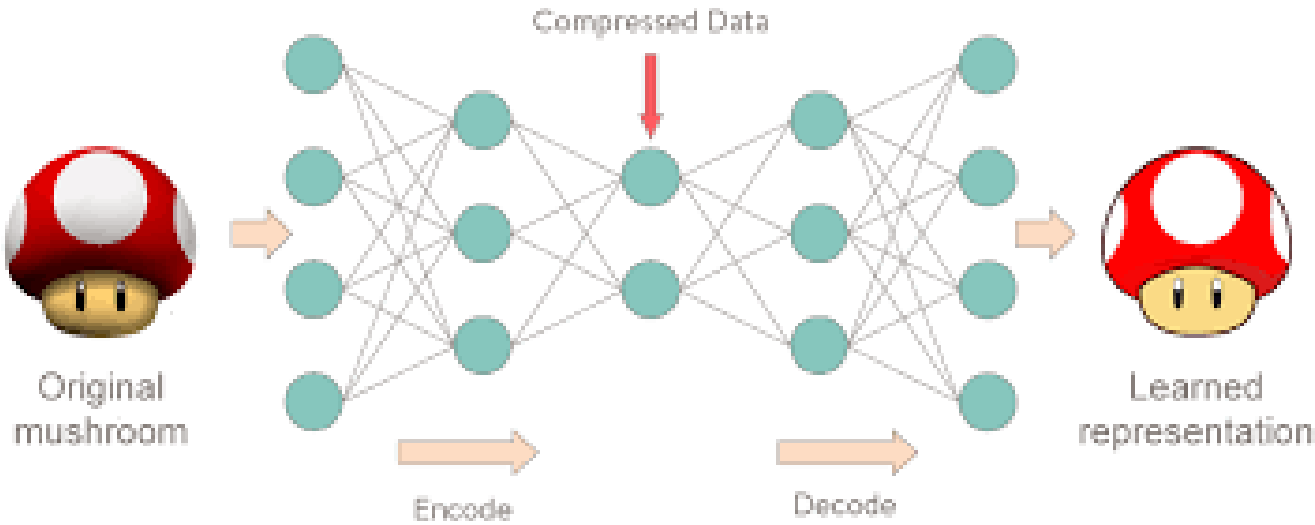
- Richard A. Johnson and Dean W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall
- Jon Shlens, *Tutorial on Principal Component Analysis*, December 2005.
- Lindsay I. Smith, *A tutorial on Principal Components Analysis*, February 2002 (some application examples in its final part)

Data simplification / reduction

Auto-encoders

Auto-encoders




Compressed Data
Original mushroom
Encode
Decode
Learned representation

From ML to DL

- **Supervised ML**
  - Classification $\times$ Regression
  - Perceptron
  - Linear regression
  - Logistic regression
  - ML diagram (X, Y, unknown target f, dataset D, Hypothesis space, learning algorithm, final hypothesis h)
  - E_{in}, E_{out}, performance metrics
  - VC theory (Hoeffding inequality, dichotomies, growth function, VC dimension)
  - Bias-variance model
  - Overfitting
  - Regularization
  - Validation / Model Selection
  - Neural networks
  - SVM

Unsupervised ML

- Clustering (hierarchical, K-means), SOM
- Dimensionality reduction (PCA, auto-encoders)

Other topics (not discussed yet)

- Feature selection / Ensembles
- Decision trees, Random Forest, GBM (XGBoost)
- Active learning
- Reinforcement learning
- Semi-supervised learning
- Self-learning
- Probabilistic learning (generative approaches)
- Deep learning / Adversarial training (GAN)

Revisiting

Generative $\times$ Discriminative

Probabilistic formulation

Bayes' Theorem

$$P(y \,|\, \mathbf{x}) = \frac{P(y)\, P(\mathbf{x} \,|\, y)}{P(\mathbf{x})}$$

If you know the distributions, you have the winning rule:

$$y^* = \arg\max_y \{P(y \,|\, \mathbf{x})\}$$

Statistics $\times$ Machine Learning ?

Bayes' Theorem

$$P(y \,|\, \mathbf{x}) = \frac{P(y)\, P(\mathbf{x} \,|\, y)}{P(\mathbf{x})}$$

1) Focus on the distribution of $\mathbf{x}$

$\Longrightarrow$ **Generative approaches**

2) Focus on the expected output $P(y \,|\, \mathbf{x})$:

$\Longrightarrow$ **Discriminative approaches**

$\Longrightarrow$ Distribution-free learning

Bayes classifier requires computation of

$$P(y_j|\mathbf{x}) \sim P(y_j)P(\mathbf{x}\,|\,y_j), \quad \mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$P(\mathbf{x}\,|\,y_j)$ is a multivariate distribution !!

Naïve Bayes Classifier: Assumes variables x_i are independent

multivariate

$$\begin{aligned} P(\mathbf{x}\,|\,y_j) &= P(x_1, x_2, \ldots, x_d\,|\,y_j) \\ &= P(x_1\,|\,y_j)P(x_2\,|\,y_j)\ldots P(x_d\,|\,y_j) \\ &= \prod_{i=1}^{d} P(x_i\,|\,y_j) \end{aligned}$$

univariate

Example of naïve Bayes classifier application: spam detection

Two-class problem: $y = +1$ (spam) and $y = -1$ (non-spam)

Vocabulary: $V = \{v_1, v_2, \ldots, v_K\}$

Given training email texts, estimate

$P(y = +1)$ and $P(y = -1)$

$P(v_i \,|\, y = +1)$ and $P(v_i \,|\, y = -1)$, $\forall i$ (*)

Given a document $\mathbf{D}$, we can compute:

$$P(y = +1 \,|\, \mathbf{D}) \approx P(y = +1) \prod_{v_i \in \mathbf{D}} P(v_i \,|\, y = +1)$$
$$P(y = -1 \,|\, \mathbf{D}) \approx P(y = -1) \prod_{v_i \in \mathbf{D}} P(v_i \,|\, y = -1)$$

* To avoid $P(v_i \,|\, y) = 0$ we use $P(v_i \,|\, y) = \frac{\#v_i + \alpha}{\#v + \alpha K}$, where $\#v_i$ and $\#v$ are, respectively, the number of occurrences of word v_i and words $v \in V$ in documents of class y

Reference for generative approaches:

Machine Learning — A probabilistic perspective
Kevin P. Murphy

Ghahramani, Z. Probabilistic machine learning and artificial intelligence. Nature 521, 452–459 (2015). https://doi.org/10.1038/nature14541

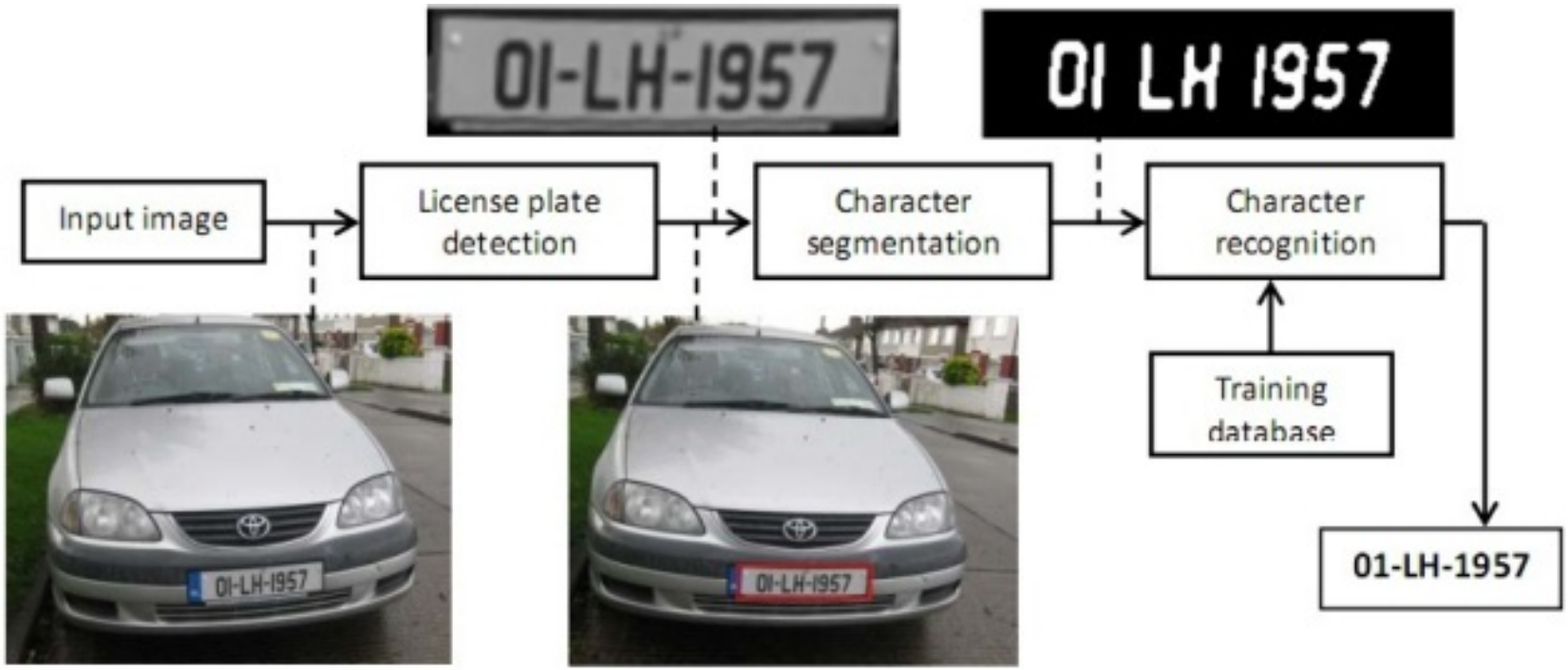
Traditional $\times$ Deep learning

Traditional machine learning: Feature computation

Features that describe the object

- concentration
- asymmetry
- smoothness
- entropy
- spirality, etc

$(x_1, x_2, \ldots, x_n) \Rightarrow$ generic model $y = f_w(x_1, x_2, \ldots, x_n) \Rightarrow y = \text{spiral}$


01-LH-1957
01 LH 1957
Input image
License plate detection
Character segmentation
Character recognition
Training database
01-LH-1957

When DL started: (This information is not precise)

> [From Wikipedia] In 2006, publications by Geoff Hinton, Ruslan Salakhutdinov, Osindero and Teh[59] [60][61] showed how a many-layered feedforward neural network could be effectively pre-trained one layer at a time, treating each layer in turn as an unsupervised restricted Boltzmann machine, then fine-tuning it using supervised backpropagation.[59] The papers referred to learning for deep belief nets.

[59] Hinton, Geoffrey E. (2007). "Learning multiple layers of representation".

Trends in Cognitive Sciences. 11(10):428–434

[60] Hinton, G. E.; Osindero, S.; Teh, Y. W. (2006). "A Fast Learning Algorithm for Deep Belief Nets". Neural

Computation. 18(7):1527–1554

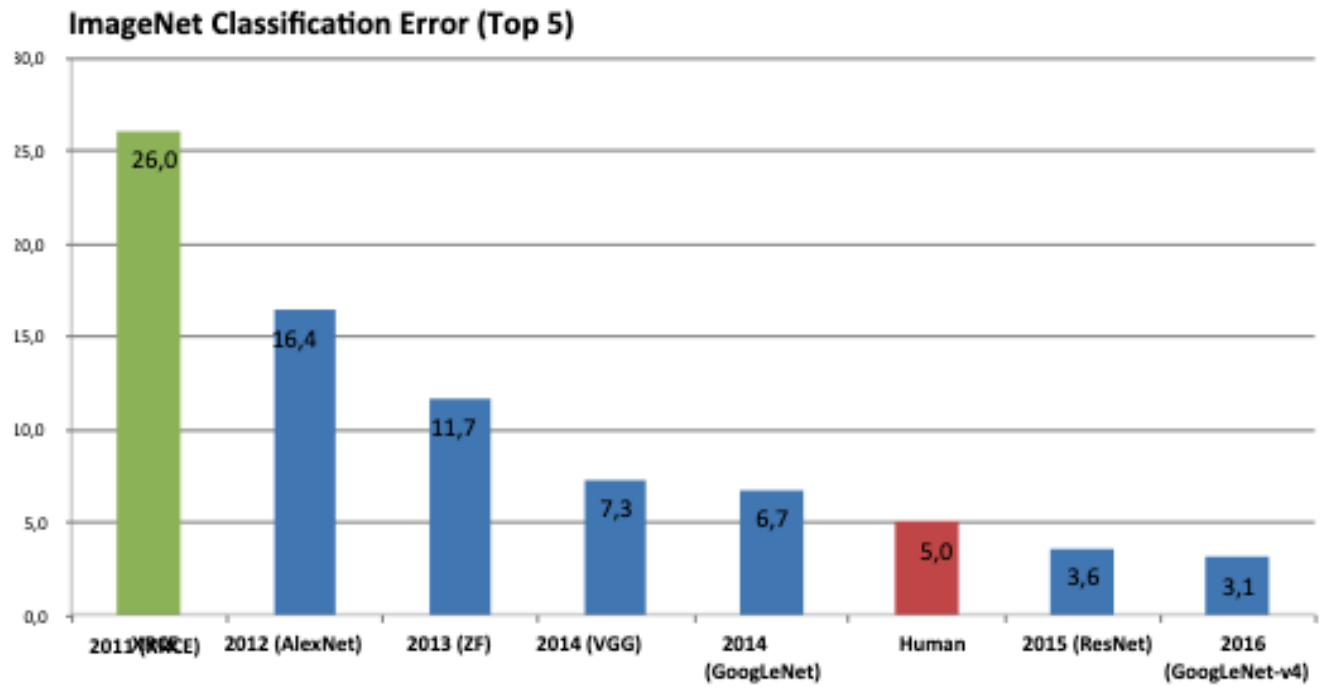
[61] Bengio, Yoshua (2012). "Practical recommendations for gradient-based training of deep architectures"

Deep learning in Computer Vision

> [From Wikipedia] In October 2012, ... a system by Krizhevsky et al.[5] won the large-scale ImageNet competition by a significant margin over shallow machine learning methods.

[5]. Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffry (2012). "ImageNet Classification with Deep Convolutional Neural Networks" (PDF). NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada.

Large Scale Visual Recognition Challenge (ILSVRC) - ImageNet



IMAGENET

www.image-net.org

22K categories and 14M images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activities

Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

IMAGENET Large Scale Visual Recognition Challenge

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

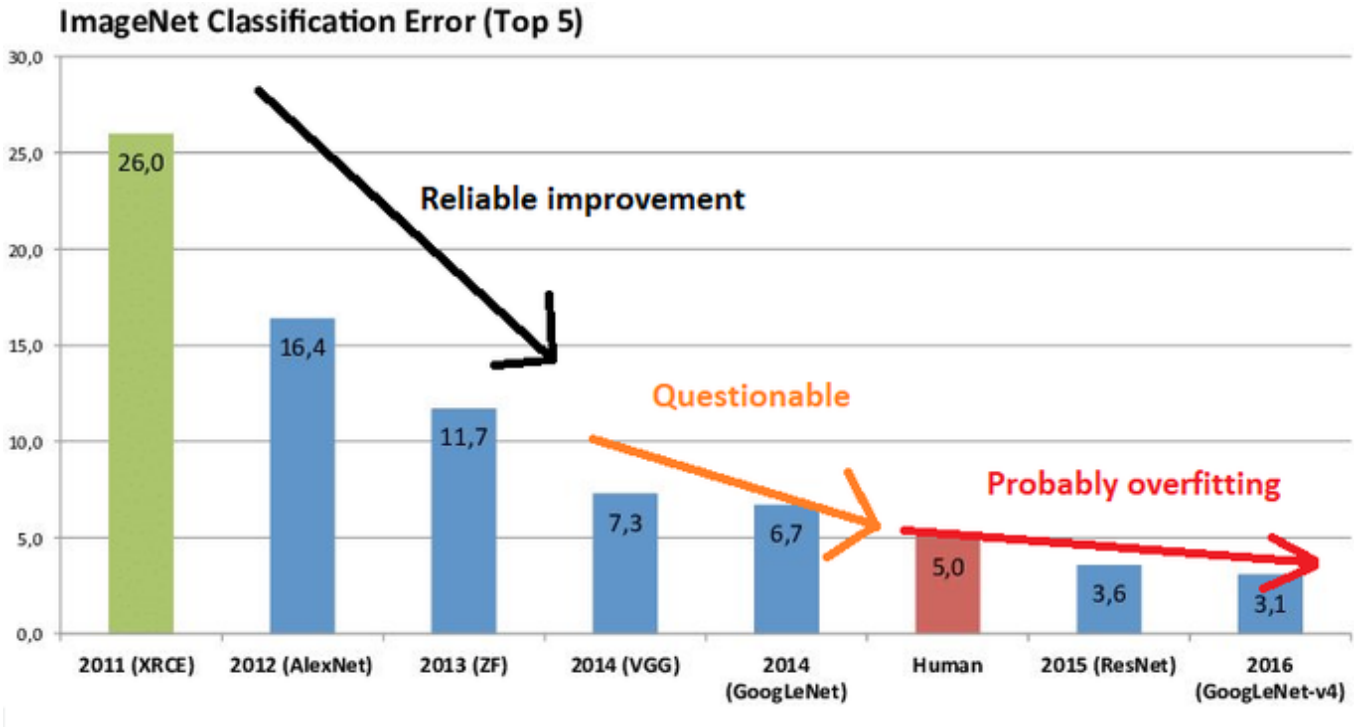
✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

Russakovsky et al. arXiv, 2014

Large Scale Visual Recognition Challenge (ILSVRC) - ImageNet



Deep learning

Large neural networks

Training them became feasible thanks to enhanced hardware, large amount of data, improvements in training strategies and network components

Traditional ML

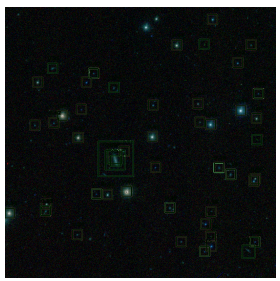
$\Rightarrow$ feature extraction/ selection $\Rightarrow (x_1, x_2, \ldots, x_n) \Rightarrow$ Traditional ML algoritm $\Rightarrow y$

Traditional ML

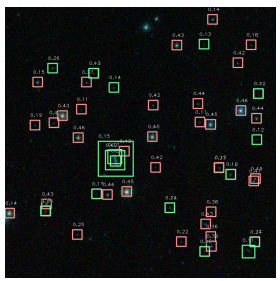
⇒ feature extraction/ selection ⇒ $(x_1, x_2, \ldots, x_n)$ ⇒ Traditional ML algoritm ⇒ y

Deep learning: end-to-end processing

⇒ Deep learning algorithm ⇒ y



⇒ Deep learning algorithm ⇒



Machine learning: evolution through time

no data

rules

```
if age > 40:
    if is_home_owner:
        print("give a credit")
    else:
        if income > 5000:
            print("give a credit")
        else:
            print("to refuse")
else:
    if education == "university":
        print("...")
    else:
        print("...")
```

Machine learning: evolution through time

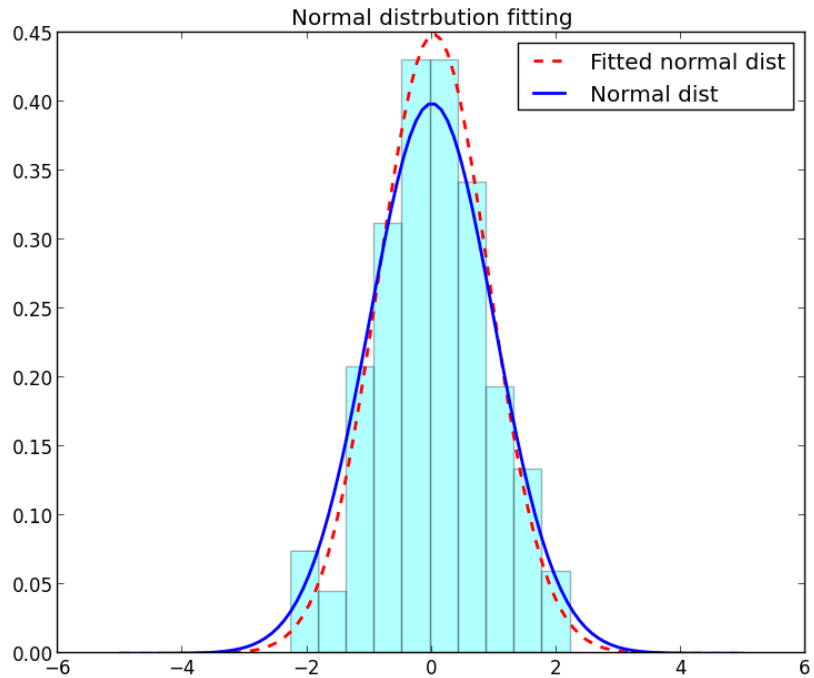
no data

rules

some data

parametric estimation

model-based



Machine learning: evolution through time

no data

rules

some data

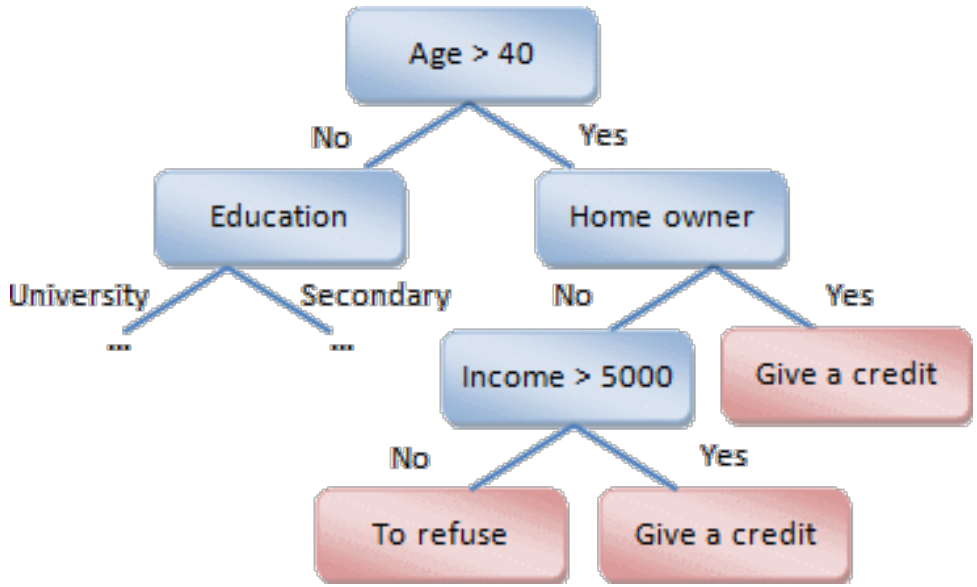
parametric estimation

model-based

more data

model induction

(ML algorithms)



Machine learning: evolution through time

no data

rules

some data

parametric estimation

model-based

more data

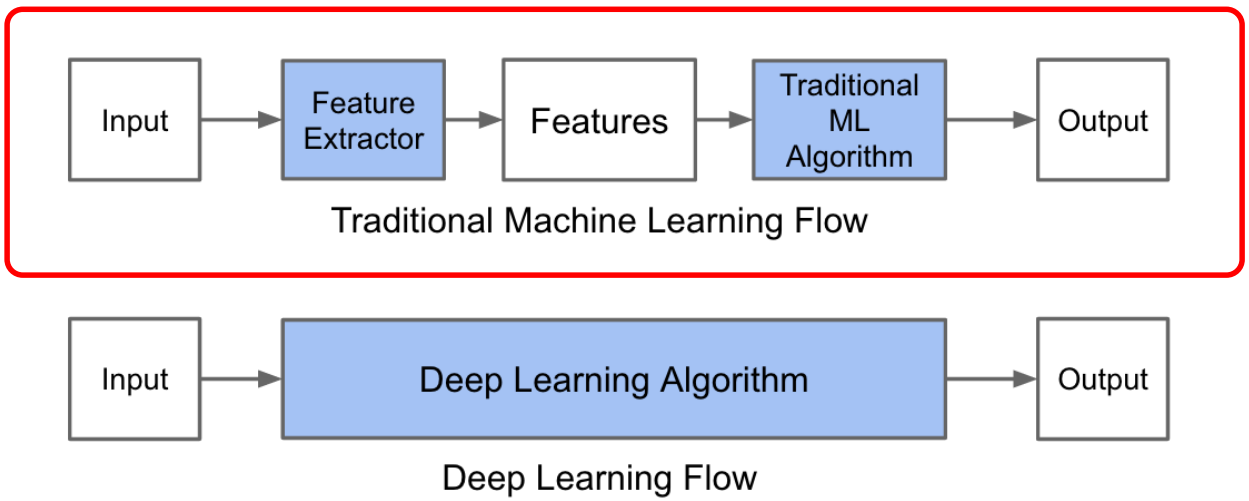
model induction

(ML algorithms)

much more data

feature engineering

ensembles



Machine learning: evolution through time

no data

rules

some data

parametric estimation

model-based

more data

model induction

(ML algorithms)

much more data

feature engineering

ensembles

Big Data

representation learning

(deep learning, end-to-end)

Traditional Machine Learning Flow

Deep Learning Flow

Machine learning: evolution through time

no data

rules

some data

parametric estimation

model-based

more data

model induction

(ML algorithms)

much more data

feature engineering

ensembles

Big Data

representation learning

(deep learning, end-to-end)

Data-driven

ML evolution (a very rough account)

1970: Bayes classifier

1990: SVM, boosting, feature selection

2000: feature engineering / ensemble learning

2005 $\sim$ **2010**: representation learning, auto-encoders

2010 on: deep learning

Where deep learning is making a difference:

- Computer Vision
- Speech/Audio recognition
- Natural language processing (NLP)
- etc