A brief sketch of Machine Learning

Francesco Cocina
University of Zurich

Rome, 26th March 2018

Outline

- Machine Learning (ML)
  - Introduction
  - Types of ML
    - Supervised
    - Unsupervised
    - Reinforcement Learning
  - Visualization: Dimensionality Reduction
- Deep Learning
- Packages and Research

Machine Learning

What does it mean?

Learning a representation that maps inputs to outputs without being explicitly programmed.

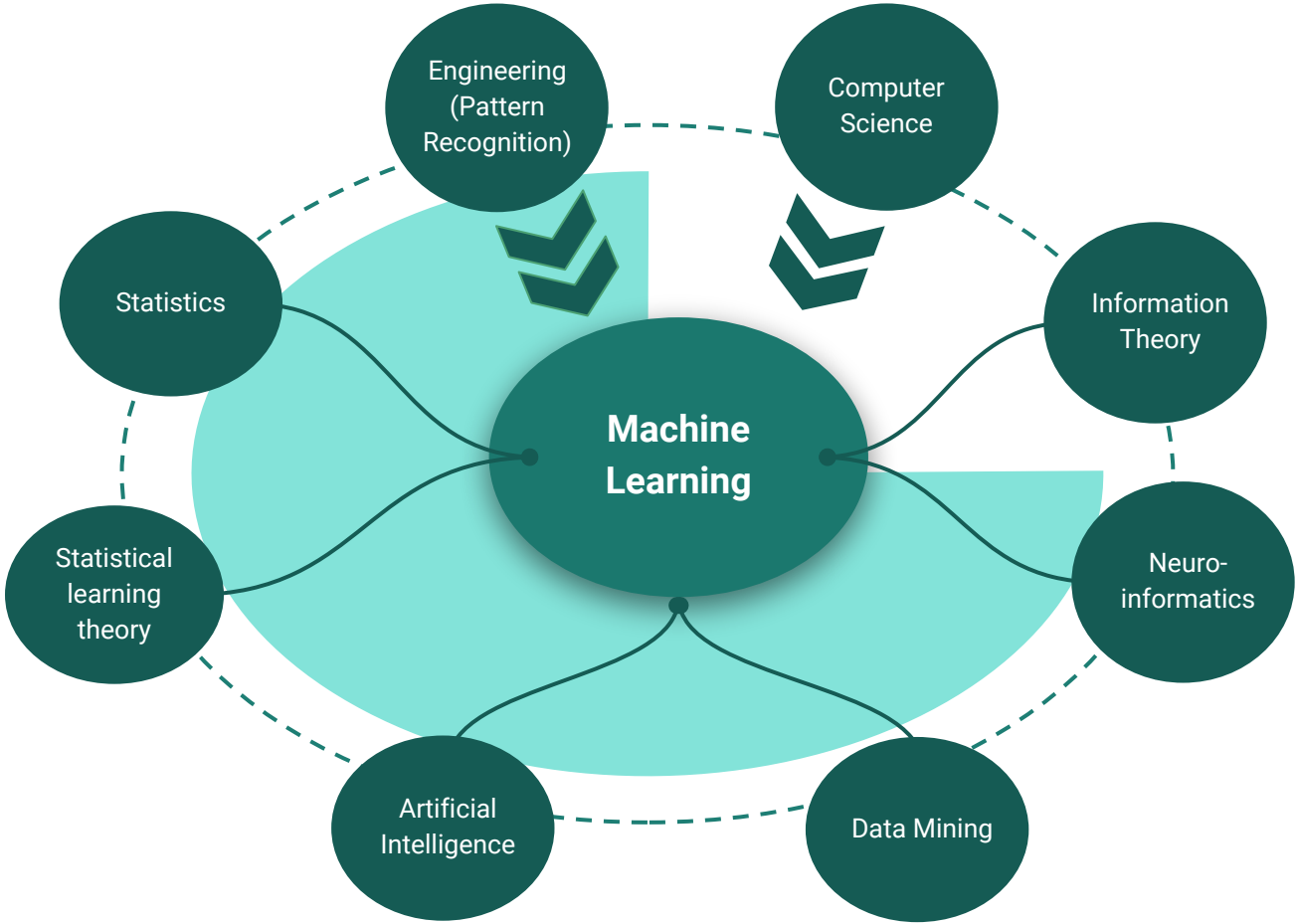
Estimation of Dependencies Based on Empirical Data

(V. Vapnik 1983)

Learning requires to infer a functional or statistical relationship between variables when we only observe noisy samples.

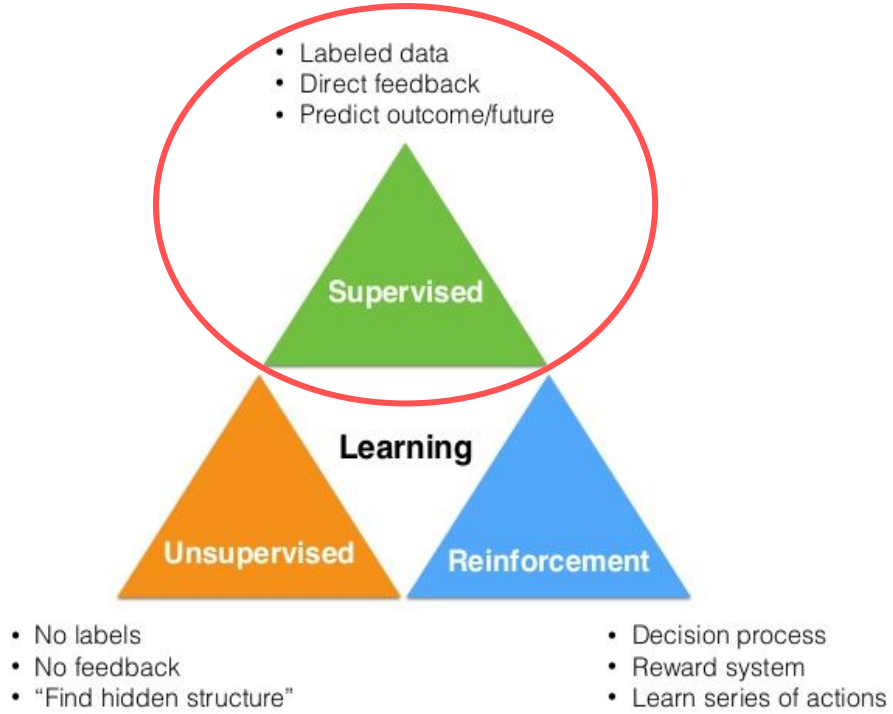
The problem without additional assumptions is ill-defined since many different functions might be compatible with noisy observations.
We, therefore, require that our inference has to “work” on future data (**generalization**).

Lots of fields involved



Types of Machine learning

- Data: N measurements of D **features** collected in the **design matrix** $\mathbf{X}$
- **Target** vectors $\mathbf{y}$:
  - Known **[supervised learning]**
  - Unknown **[unsupervised learning]**
  - Time-delayed, to be discovered by performing actions **[reinforcement learning]**
- Learning the representation $y = f(x)$ and use it to make predictions/decisions



Supervised Learning

- Data as pairs of feature and given response variables **[training set]**

$$\{(x_1, y_1), \ldots (x_n, y_n) \ : \ x_i \in \mathcal{X} \subset \mathbb{R}^d, y_i \in \mathbb{K}\}$$

  y_i an index/label **[classification]** or a continuous variable **[regression]**

- Define a function class with parameters w to be learned

$$\mathcal{C} = \{f(x, w) \ : \ w \in \mathcal{W}, x \in \mathbb{R}^d\}$$

- Once the model is trained, performance in predicting correct values are evaluated on a set of unlabeled data **[test set]**
- Three different approaches for modeling the function class:
  - Discriminant function
  - **Probabilistic models**: Generative or discriminative

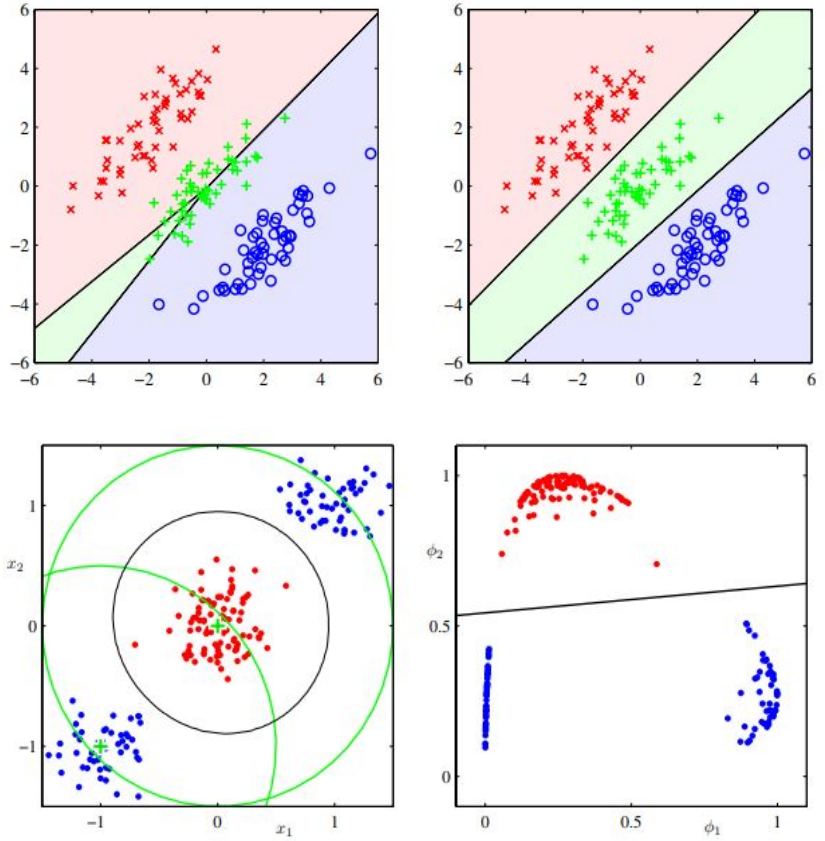
Supervised Learning. Classification

- Finding decision surfaces/boundaries.
- Non linear activation function f

  [Generalized Linear Models]

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0\right)$$

- $f(.)$ labels or posterior probabilities
- Decision surfaces are hyperplanes where y is constant
- Nonlinear features transformation using basis functions $\phi(\mathbf{x})$



Discriminative functions

- Direct assignment of $\mathbf{x}$ to a specific class
- **Perceptron** algorithm (Rosenblatt '59)

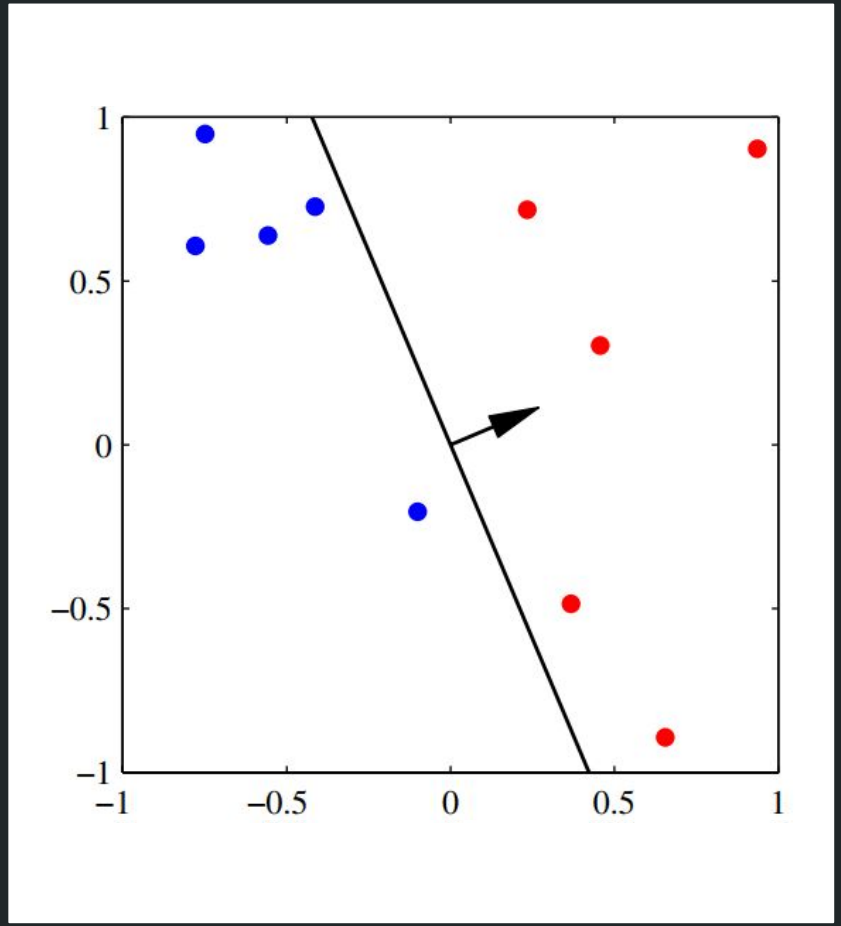
$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) \qquad f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases}$$

- Perceptron Criterion: Error function

$$E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n t_n$$

- Stochastic Gradient Descent, learning step:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\boldsymbol{\phi}_n t_n$$



Discriminative functions

- Direct assignment of $\mathbf{x}$ to a specific class
- **Perceptron** algorithm (Rosenblatt '59)

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) \qquad f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases}$$

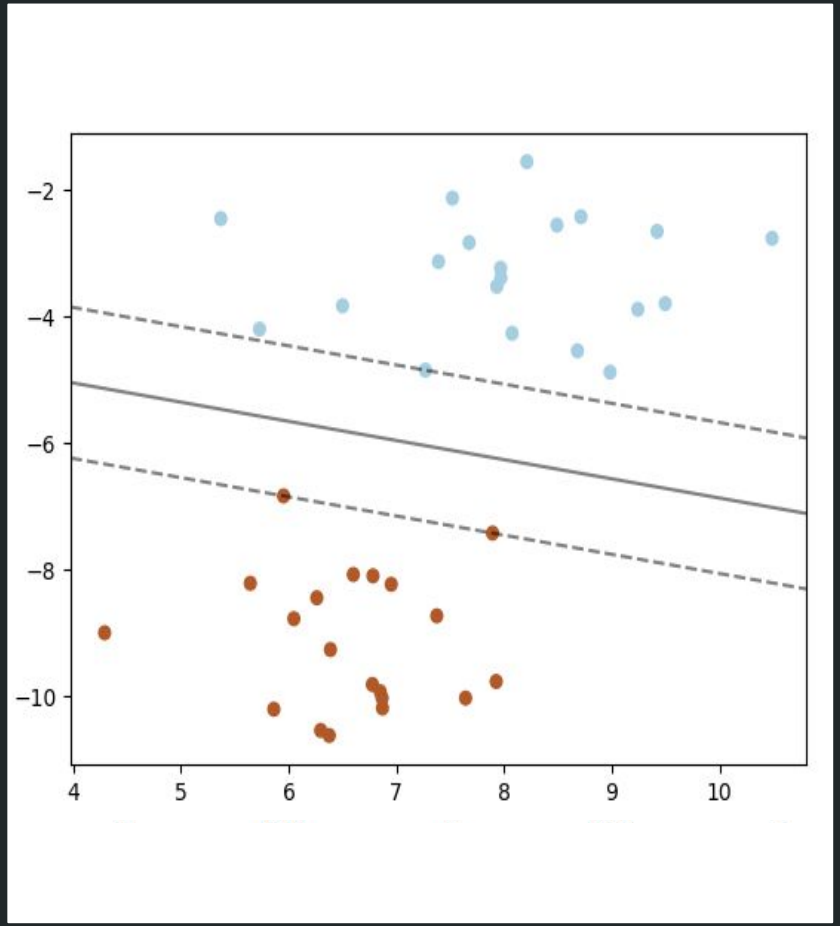
- Perceptron Criterion: Error function

$$E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n t_n$$

- Stochastic Gradient Descent, learning step:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\boldsymbol{\phi}_n t_n$$

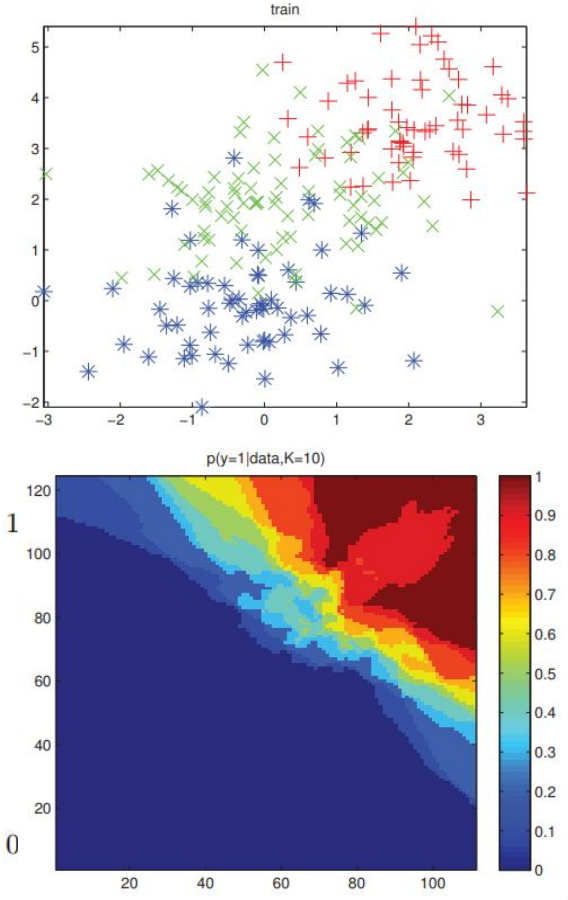
- Most popular approach involves the margin maximization [Support vector machines]



Probabilistic approach. Discriminative models

- **Inference** stage before decision: i.e determining $p(\mathcal{C}_k|\mathbf{x})$
- $p(\mathcal{C}_k|\mathbf{x})$ modeled directly. Models can be
  - Parametric with parameters optimization during training.
  - Non parametric
- Why do we want the posterior distribution? E.g.
  - Rejection criterion determination
  - Combining different models

$$
\begin{aligned}
p(\mathcal{C}_k|\mathbf{x}_\mathrm{I}, \mathbf{x}_\mathrm{B}) &\propto p(\mathbf{x}_\mathrm{I}, \mathbf{x}_\mathrm{B}|\mathcal{C}_k)p(\mathcal{C}_k) \\
&\propto p(\mathbf{x}_\mathrm{I}|\mathcal{C}_k)p(\mathbf{x}_\mathrm{B}|\mathcal{C}_k)p(\mathcal{C}_k) \\
&\propto \frac{p(\mathcal{C}_k|\mathbf{x}_\mathrm{I})p(\mathcal{C}_k|\mathbf{x}_\mathrm{B})}{p(\mathcal{C}_k)}
\end{aligned}
$$



Discriminative models. Logistic regression

- Posterior probability as logistic sigmoid function

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right) \qquad \sigma(a) = \frac{1}{1+\exp(-a)}$$

- Likelihood function of Bernoulli variables

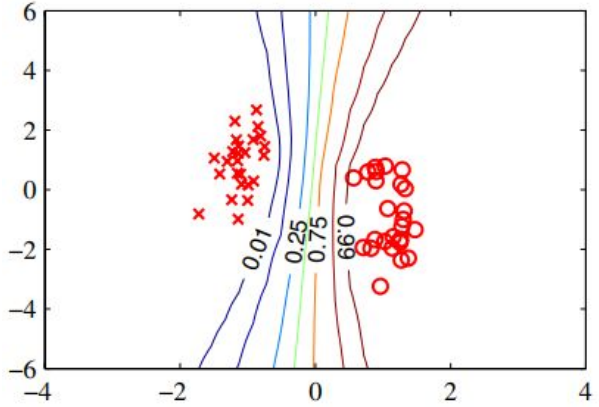
$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\left\{1-y_n\right\}^{1-t_n} \qquad \begin{aligned} y_n &= p(\mathcal{C}_1|\boldsymbol{\phi}_n) \\ t_n &\in \{0,1\} \end{aligned}$$

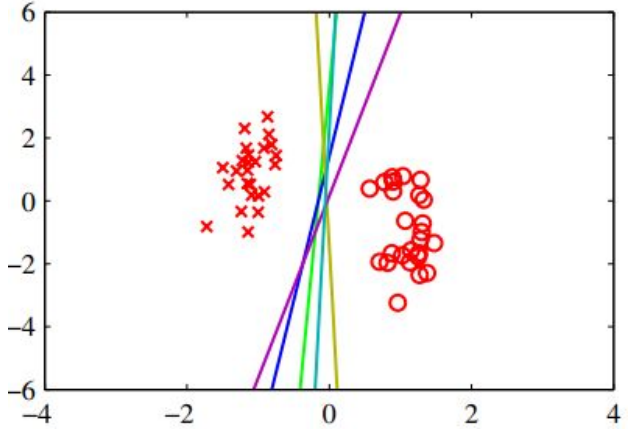
- **Cross Entropy** Error function minimization

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\right\}$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}(y_n - t_n)\boldsymbol{\phi}_n$$

- Prone to **overfitting** if training data are linearly separable.
  - Prior and MAP for $\mathbf{w}$ (**Bayesian** approach)
  - **Regularization** term for the error function





Probabilistic approach. Generative models

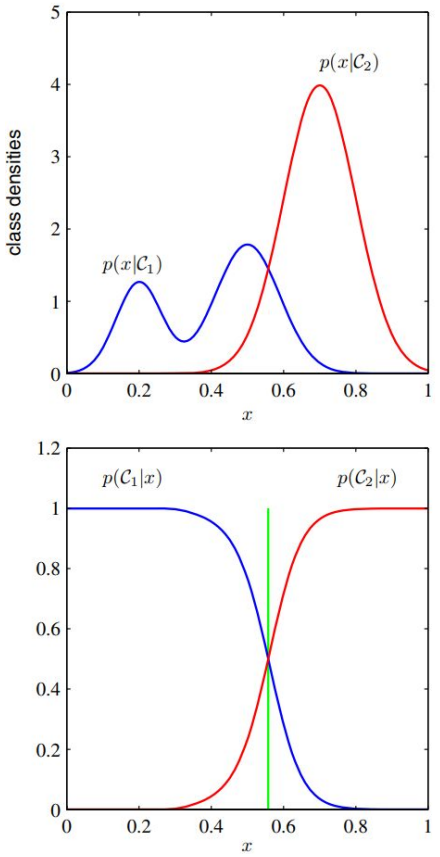
- Modelling class conditional densities and prior probabilities, i.e

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

- Generating synthetic data point by sampling
- Allow to estimate marginal densities of data (outliers detection)

❑ Most demanding, lots of training samples needed.

❑ Efforts no needed if we want just predictions (little effect of conditional densities)



Generative models. Naive Bayes classifiers

- Conditional independence given the class, **Naive Bayes model**

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k) \quad \Rightarrow \quad p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

- MAP as decision rule **Naive Bayes classifier**

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

- Gaussian $p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$
- Multinomial $p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$

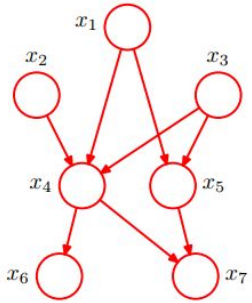
- **Spam filtering** application: x text mail [**bag-of-words**],

  p_{ki} word i probability in each class (spam/ham)

Feature	Appearances in Spam	Appearances in Ham	$P(S\|w_i)$
A	165	1235	0.2512473
Advised	12	42	0.4177898
As	2	579	0.0086009
Chance	45	35	0.7635468
Clarins	1	6	0.2950775
Exercise	6	39	0.2787054
For	378	1829	0.3417015
Free	253	137	0.8226372
Fun	59	9	0.9427419
Girlfriend	26	8	0.8908609
Have	291	2008	0.2668504
Her	38	118	0.4471509
I	9	1435	0.0155078
Just	207	253	0.6726596
Much	126	270	0.5396092
Now	221	337	0.6222218
Paying	26	10	0.8671995
Receive	171	98	0.8142107
Regularly	9	87	0.2062346
Take	142	287	0.5541010
Tell	76	89	0.6820062
The	185	930	0.3331618
Time	212	446	0.5441787
To	389	1948	0.3340176
Too	56	141	0.4993754
Trial	26	13	0.8339739
Vehicle	21	58	0.4762651
Viagra	39	19	0.8375393
You	391	786	0.5554363
Your	332	450	0.6494897

Probabilistic Graphical Models

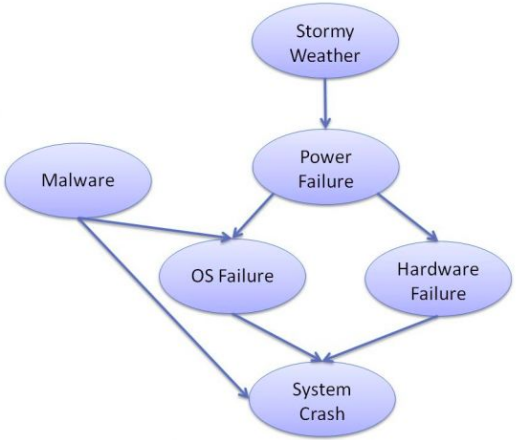
- Joint probabilities as diagrammatic models (visualization, insights, inference algorithms)



Bayesian Networks

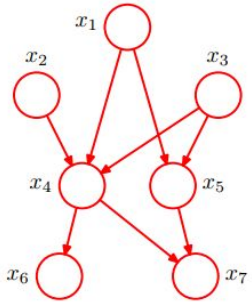
$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$



Probabilistic Graphical Models

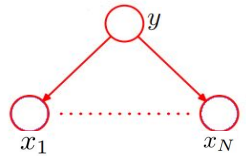
- Joint probabilities as diagrammatic models (visualization, insights, inference algorithms)



Bayesian Networks

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

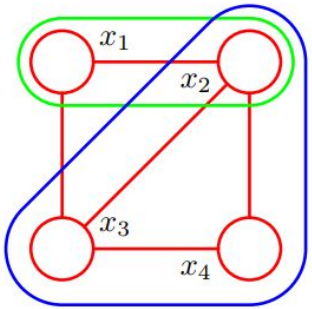
$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$



$$p(y,\mathbf{x}) = p(y)\prod_{i=1}^{n} p(x_i|y)$$

Bayes naive model

- No assumptions about causality: undirected graphs and distribution as product of maximal cliques



Markov Random Fields

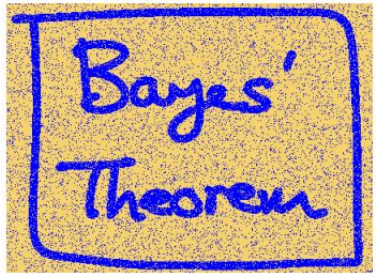
$$p(\mathbf{x}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

Potential functions

Denoising application

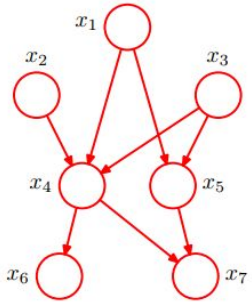
Original image



Corrupted (10%)

Probabilistic Graphical Models

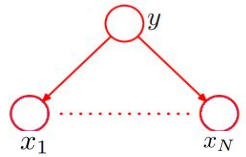
- Joint probabilities as diagrammatic models (visualization, insights, inference algorithms)



Bayesian Networks

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

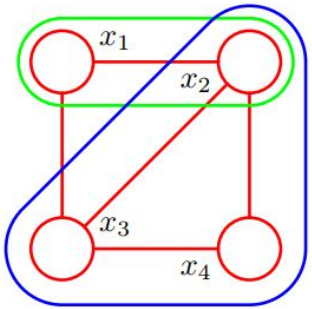
$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$



$$p(y,\mathbf{x}) = p(y)\prod_{i=1}^{n} p(x_i|y)$$

Bayes naive model

- No assumptions about causality: undirected graphs and distribution as product of maximal cliques



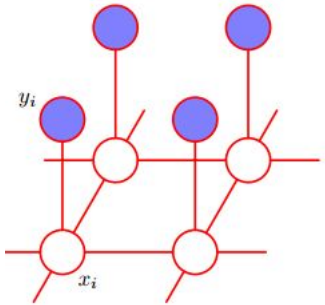
Markov Random Fields

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$

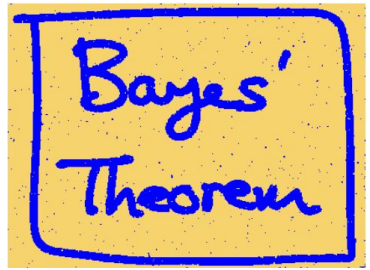
$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

Potential functions

Denoising application



$$E(\mathbf{x},\mathbf{y}) = h\sum_i x_i - \beta\sum_{\{i,j\}} x_i x_j - \eta\sum_i x_i y_i$$



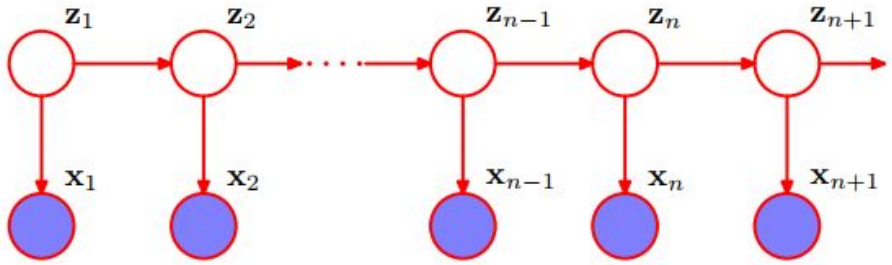
Restored (96%)

Probabilistic graphical models. Sequential models

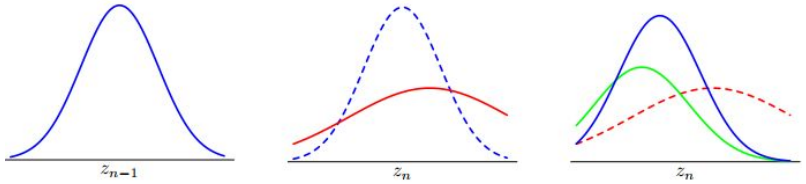
- Noisy observations x_i of a **latent** variable z

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_N,\mathbf{z}_1,\ldots,\mathbf{z}_N) = p(\mathbf{z}_1)\left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1})\right]\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n)$$

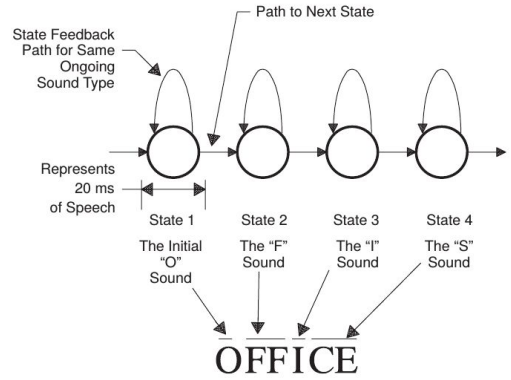
Transition Sensor

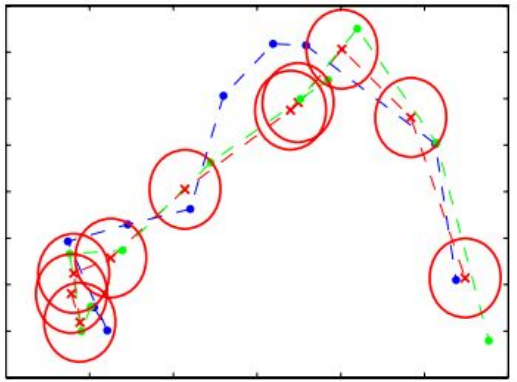


- Main inference tasks:
  - **Prediction** $p(\mathbf{z}_n|\mathbf{x}_1,\ldots,\mathbf{x}_{n-1})$
  - **Filtering** $p(\mathbf{z}_n|\mathbf{x}_1,\ldots,\mathbf{x}_n)$



- $\mathbf{z}$ a discrete value ➪ **Hidden Markov Models**
  - Application: Speech recognition
- Gaussian variables ➪ **Kalman Filters**
  - Application: motion tracking

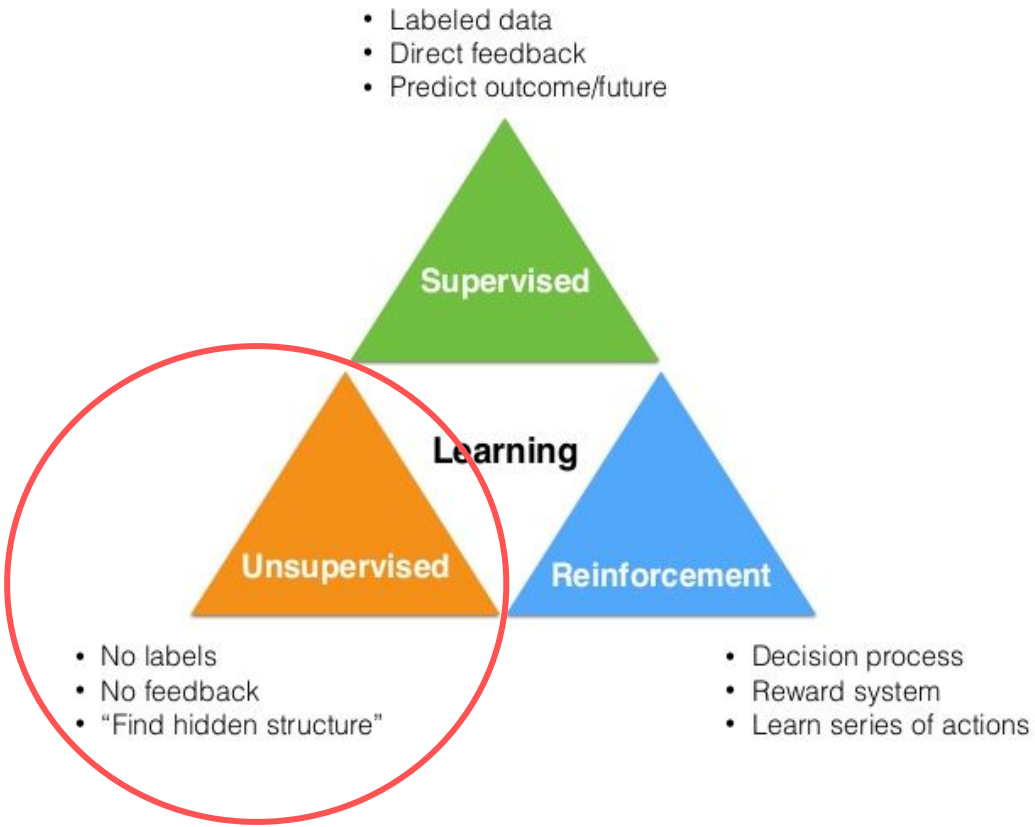




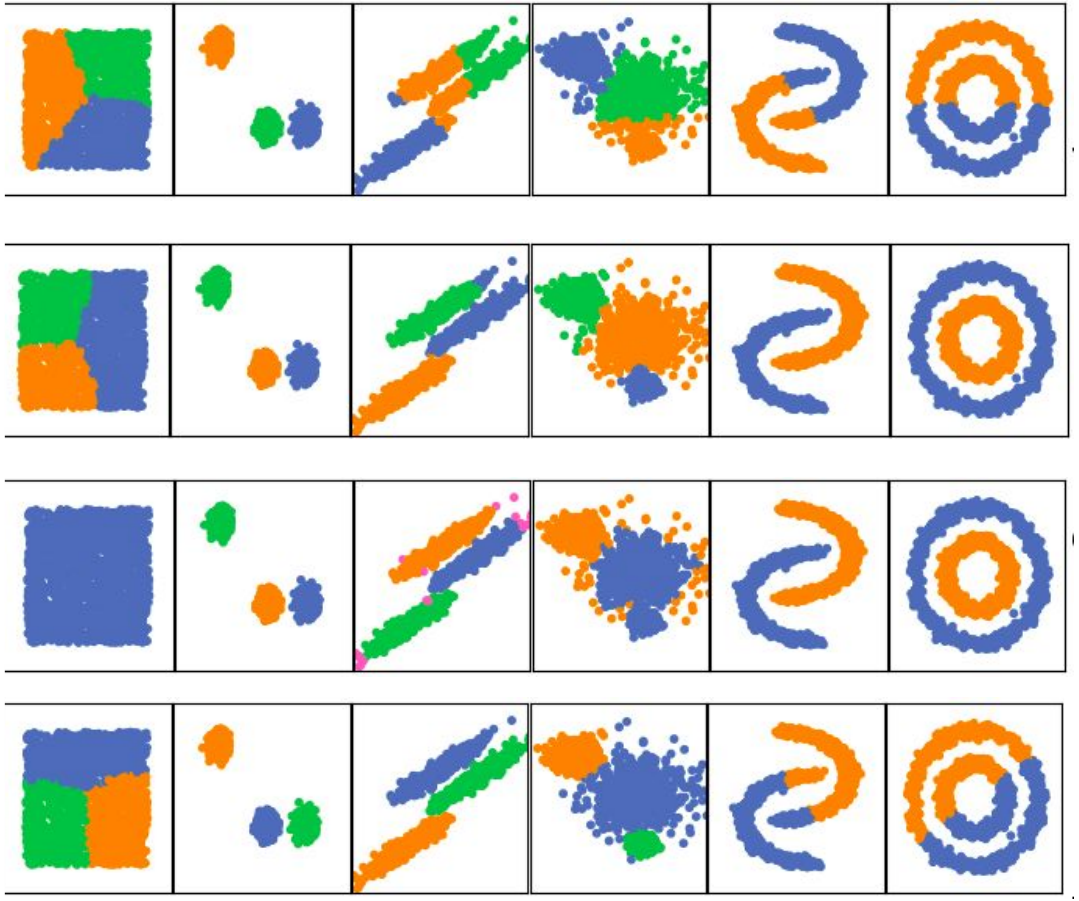
$$\mathbf{x}_k = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{G}a_k$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$


• Labeled data
• Direct feedback
• Predict outcome/future
Supervised
Learning
Unsupervised
Reinforcement
• No labels
• No feedback
• "Find hidden structure"
• Decision process
• Reward system
• Learn series of actions

Unsupervised learning. Clustering



K-Means

Minimizing $$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

Spectral clustering

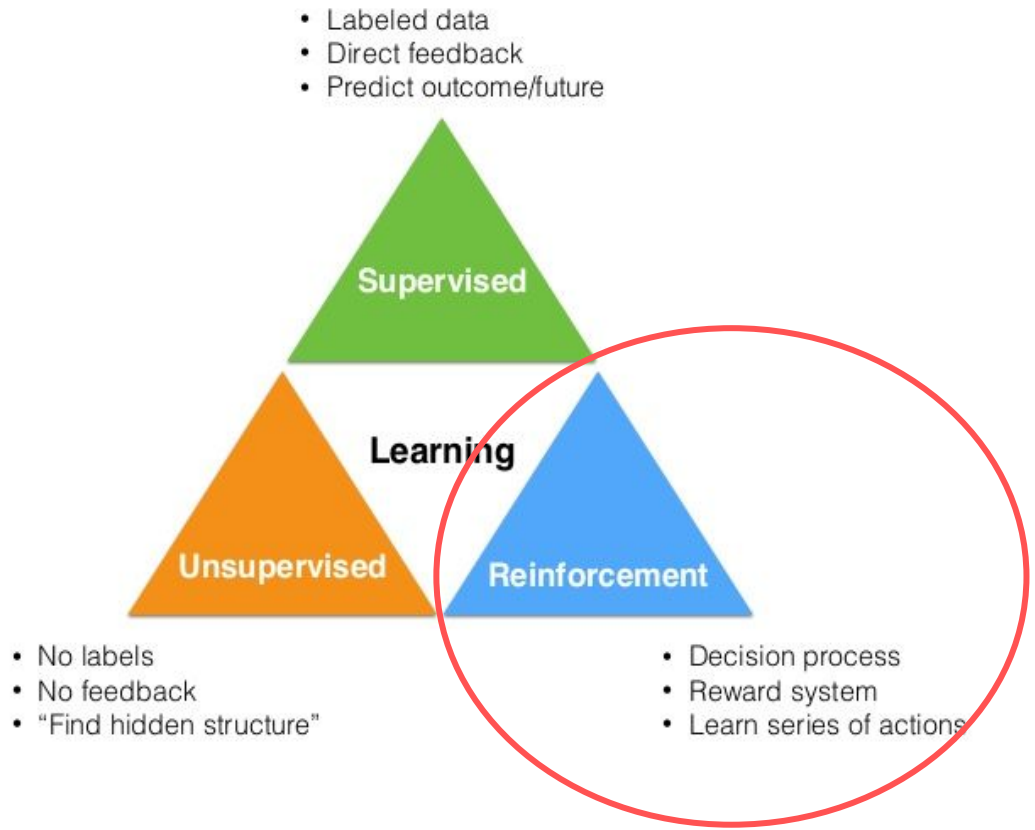
Graph distance based. Involves spectrum estimation of the Laplacian.

DBSCAN

Density-based. Based on the fixed radius Nearest Neighbours tree (frNN).

Gaussian Mixture models

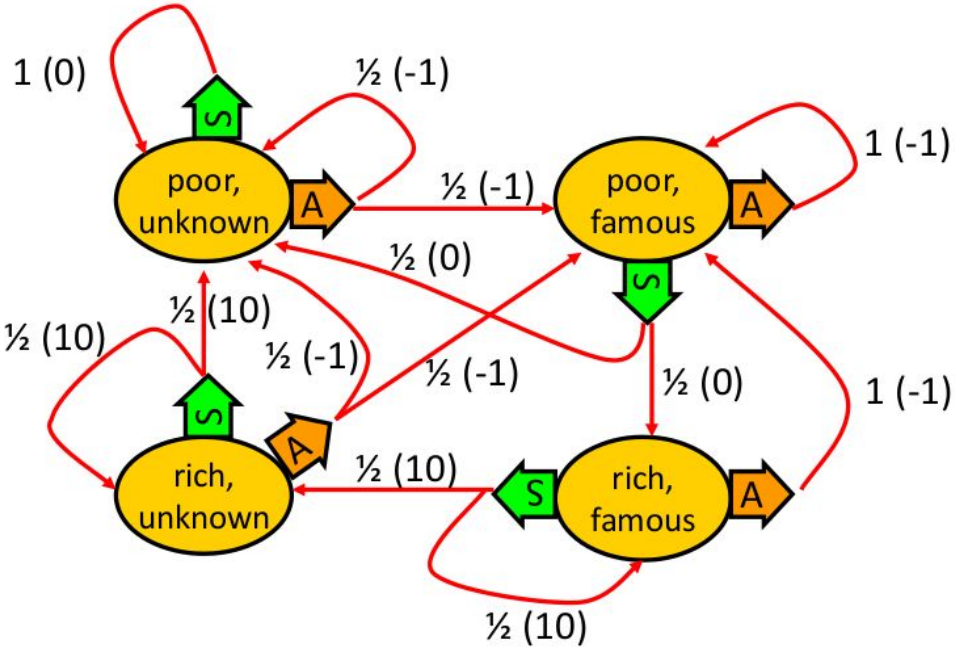
Fuzzy/soft/probabilistic. Data from a mixture of K Gaussians.
Estimation of the posterior distribution.


• Labeled data
• Direct feedback
• Predict outcome/future
Supervised
Learning
Unsupervised
Reinforcement
• No labels
• No feedback
• "Find hidden structure"
• Decision process
• Reward system
• Learn series of actions

Markov Decision Processes

- Modeling decision making:
  - A set of states X, A set of action A
  - Transition Probabilities $P(x'|x,a)$
  - A reward function $r(x,a)$
- Choosing optimal **policy** π to maximize the expected rewards **[value function]**

$$V^{\pi}(x) = \mathbb{E}[\sum_{t=0} \gamma^t r(x_t, \pi(x_t)) \,|X_0 = x]$$

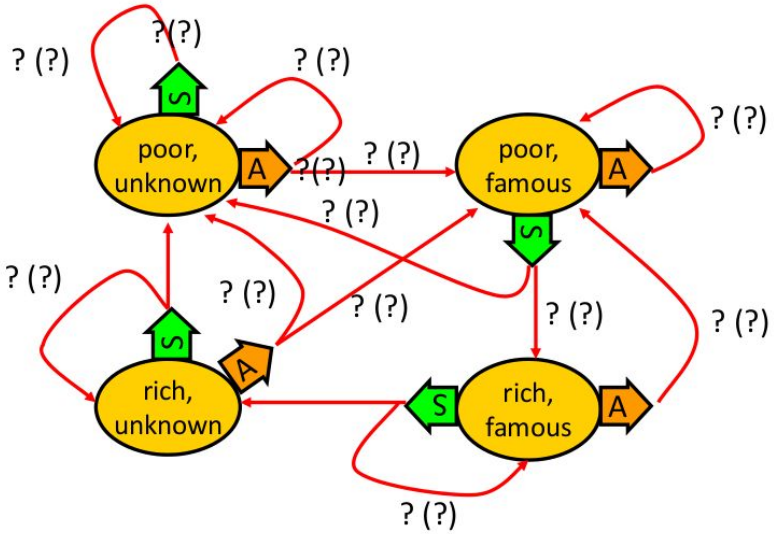


- **Bellmann Theorem**
  - **A policy is optimal ⇔ it's greedy with respect to its induced value function**

$$V^*(x) = \max_a r(x,a) + \gamma \sum_{x'} P(x' \mid x, a)\, V^*(x')$$

Reinforcement Learning

- Data we get depends on our actions!
- *Exploration-Exploitation dilemma.* Should we:
  - **Explore**: Gather more data to avoid missing out large rewards
  - **Exploit**: Stick with our current knowledge and built an optimal policy



- Two basic approaches:
  - Model-Based RL. Learn the MDP
  - Model-free RL

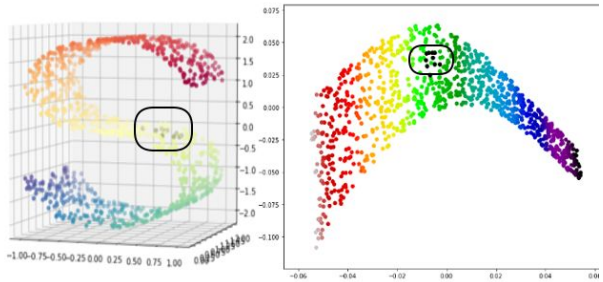
$$V^*(x) = \max_a Q^*(x,a)$$
$$Q^*(x,a) = r(x,a) + \gamma \sum_{x'} P(x' \mid x,a) V^*(x')$$

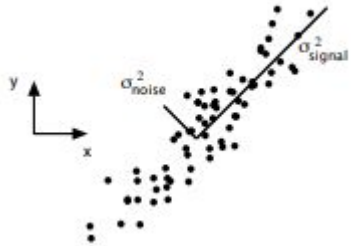
Q-learning $\longrightarrow$ $$Q(x,a) \leftarrow (1-\alpha_t)Q(x,a) + \alpha_t \left(r + \gamma \max_{a'} Q(x',a') \right)$$

Parametric Q-function $\longrightarrow$ $Q(x,a;\theta)$

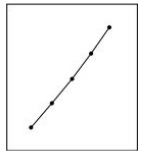
Visualization. Dimensionality reduction

- Projecting data in a low-D space
- Identify the intrisic dimensionality and preserving relevant structures [manifold learning]
- Remove redundancies and noisy features from the datasets
- Covariance based: e.g. PCA
- Distance preserving: e.g. Multidimensional scaling
- Nonlinear techniques: e.g. Sammon Mapping, Isomap

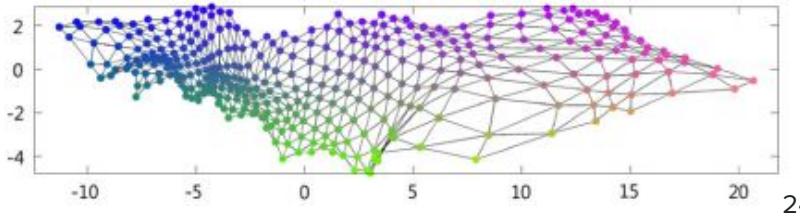




$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$



$$\min_Y \sum_{i=1}^{t} \sum_{i=1}^{t} (d_{ij}^{(X)} - d_{ij}^{(Y)})^2$$



Dimensionality reduction. t-SNE

- Convert distances into conditional/joint probabilities (similarities)
  - Gaussian in the high dimensional space
  - t-Student in the low dimensional (heavy tails)
- Variance in the high dimensional accounting for the local density
- Minimizing the Kullback-Leibler distance

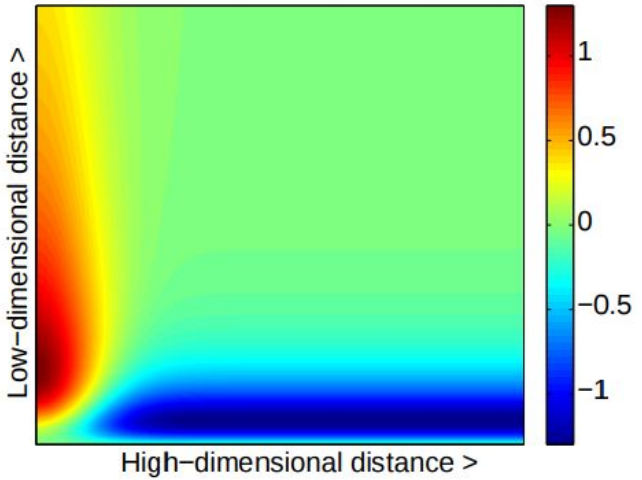
$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

$$Perp(P_i) = 2^{H(P_i)}$$

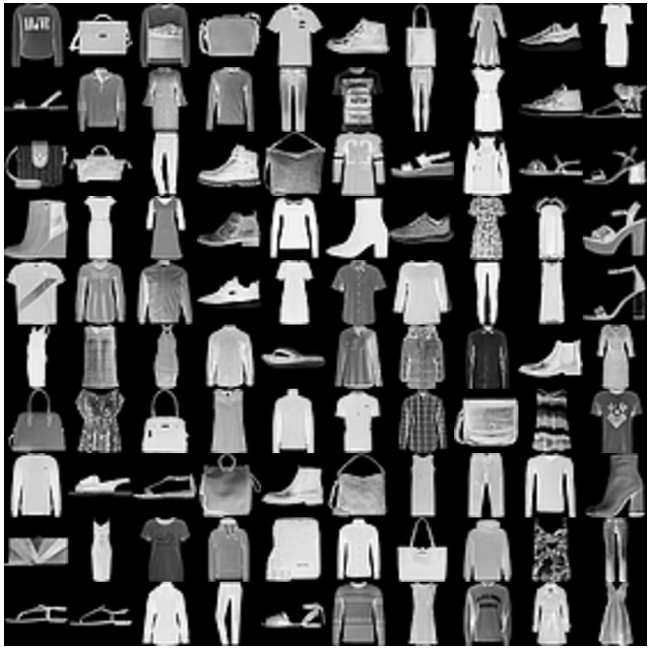
$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1}$$

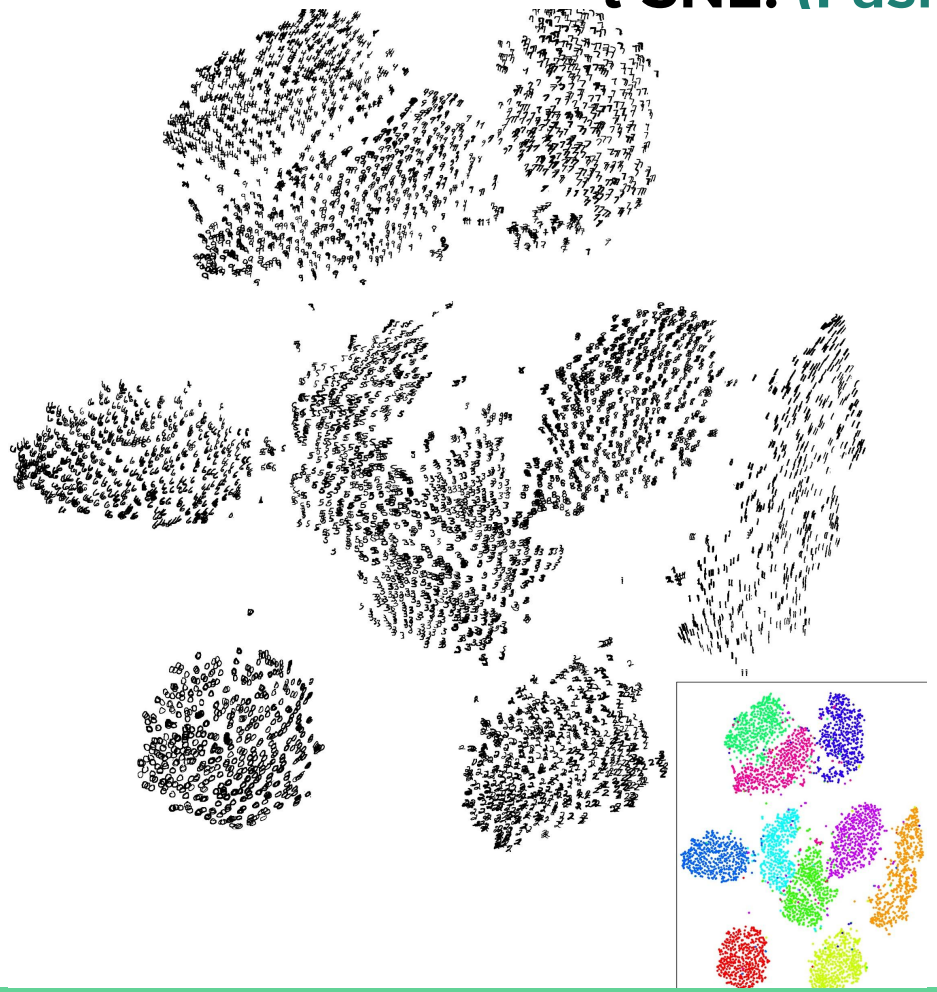


Van der Maaten L., Hinton G. Journal of Machine Learning. 2008.

t-SNE. (Fashion) MNIST.



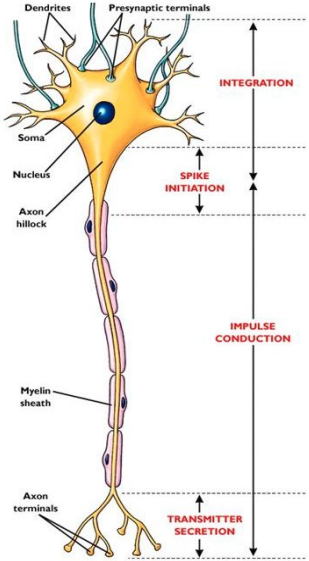
t-SNE. (Fashion) MNIST.

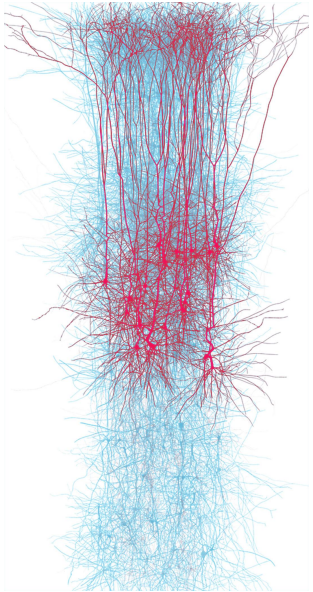


Neural networks and Deep Learning

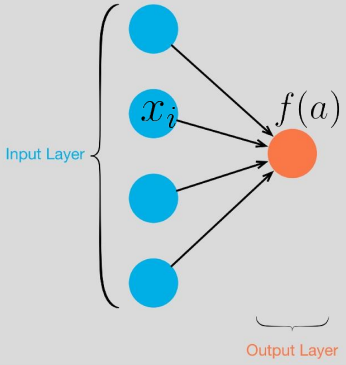
Neural Networks

- Inspired by biological networks of neurons
- Rapidly becoming tools of pattern recognition
- Starting point is Multilayer Perceptron/Feed forward networks





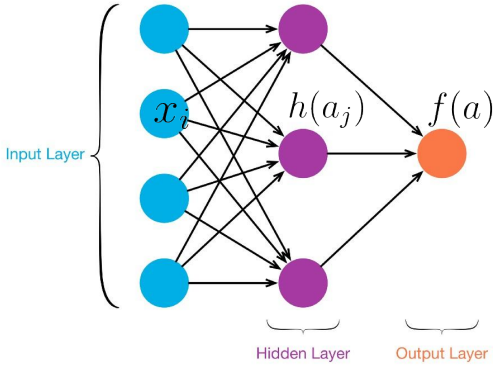
Perceptron



$$a = \sum_{i=1}^{D} w_i x_i + w_0$$

$$f(a) = \begin{cases} +1 & a > 0 \\ -1 & a > 0 \end{cases}$$

Multilayer Perceptron



$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

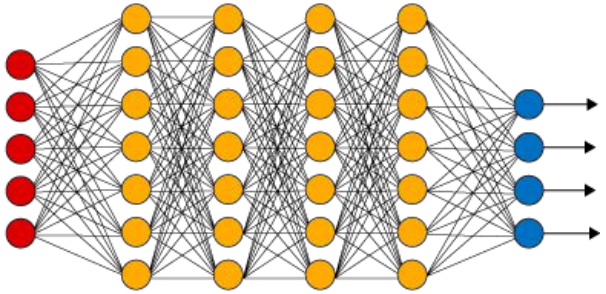
$$z_j = h(a_j)$$

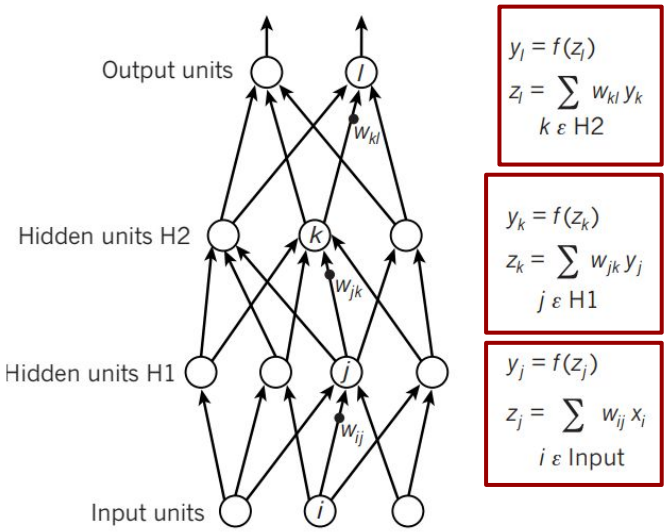
$$a = \sum_{j=1}^{K} w_i^{(2)} h(z_j) + w_0^{(2)}$$

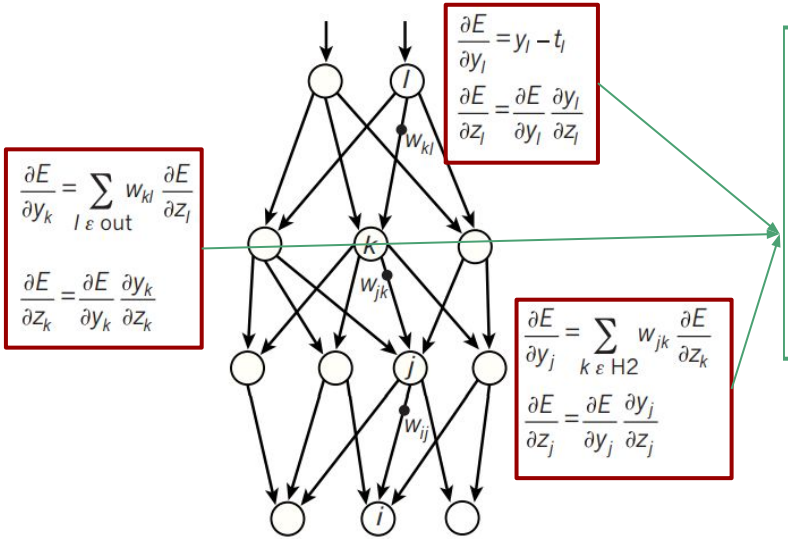
- $\boldsymbol{h}$ is the activation function
  - ReLU
  - Tanh
  - Logistic

Deep Neural Networks

- “Deep” if more than 2 hidden layers
- Each hidden layer “learns“ a representation
- Training the network through **backpropagation** (aka chain rule for derivatives)







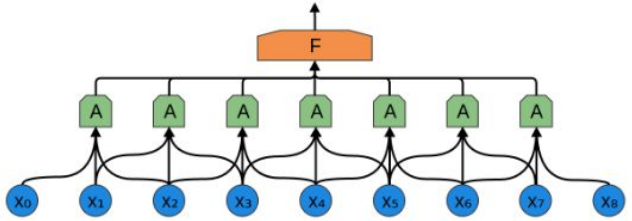
$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial w_j k}$$

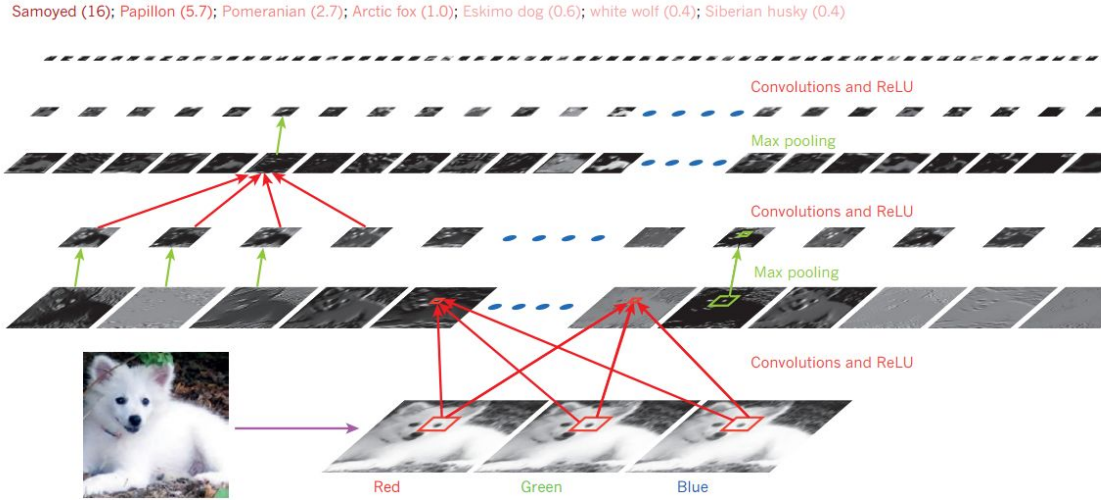
$$\frac{\partial E}{\partial w_{jk}} = y_j \frac{\partial E}{\partial z_k}$$

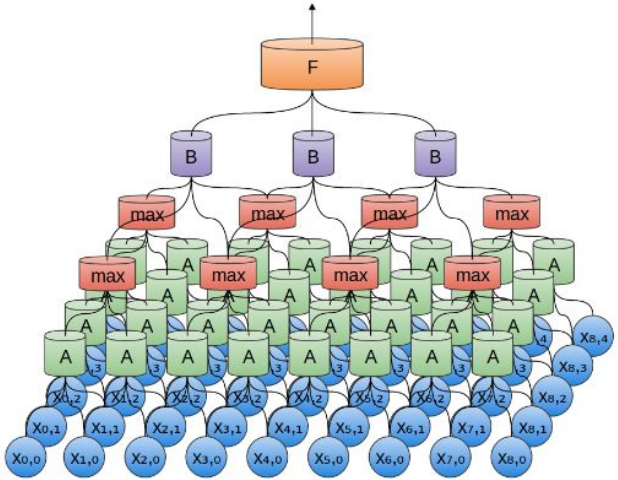
$$\Delta w_{jk} = -\alpha \frac{\partial E}{\partial w_{jk}}$$

Main Architectures. Convolutional NN

- Inspired by Hubel and Wiesel's work on early visual cortex
- Looking into the local properties of the data
- Max pooling layers for invariance to input translations

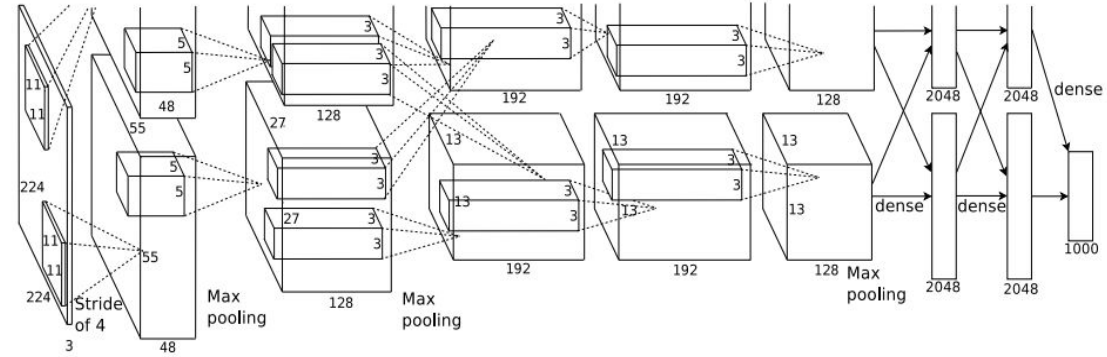


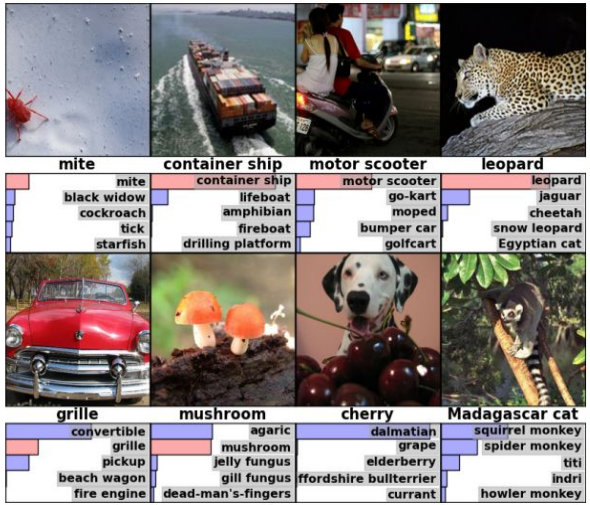


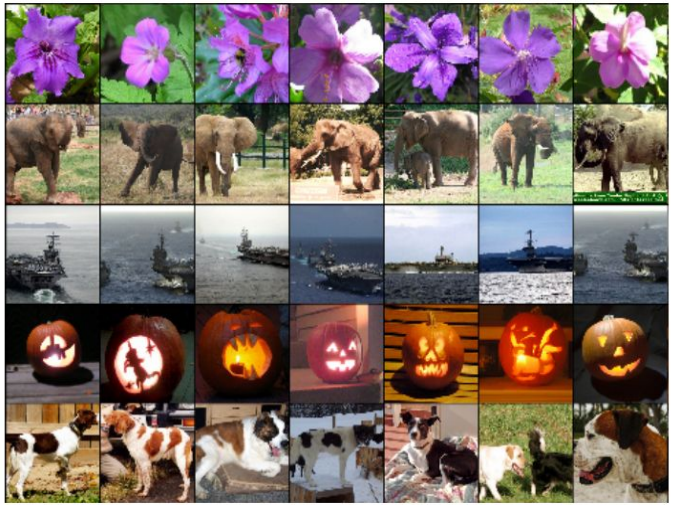


AlexNet (Krizhevsky et al. 2011)

- ImageNet competition
- 1.2 mln training images (1000 labels)
- 5 conv + 3 full connected layers
- Depth is important (drop 2% x layer)
- 60 mln parameters
- **Data augmentation**
- **Dropout:** setting to zero the output of each hidden neuron with p=0.5 (essential but expensive)
- 96 Conv kernels learned
- Top 48 (GPU1) color agnostic
- Down (GPU2) color specific

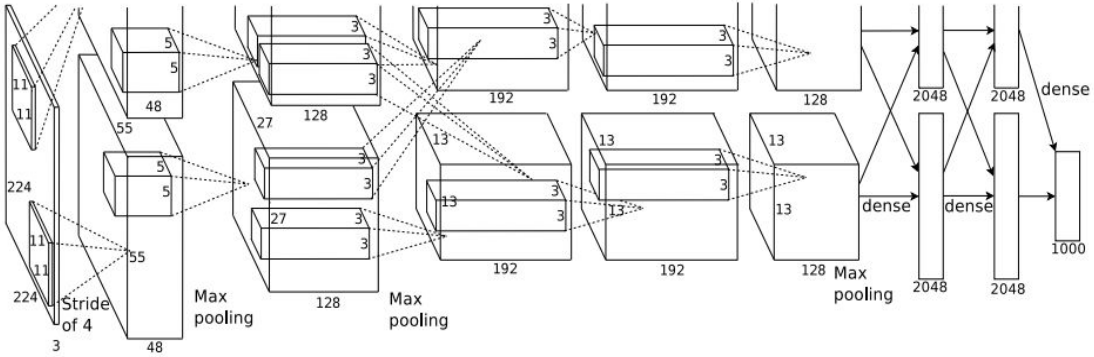


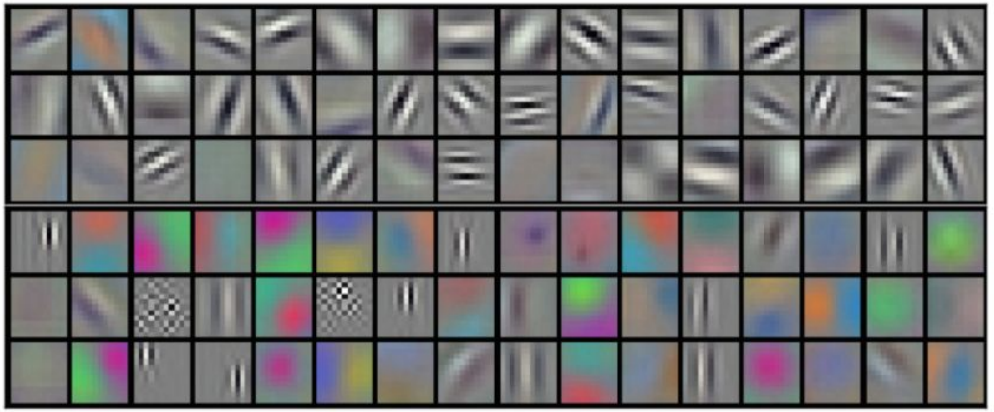




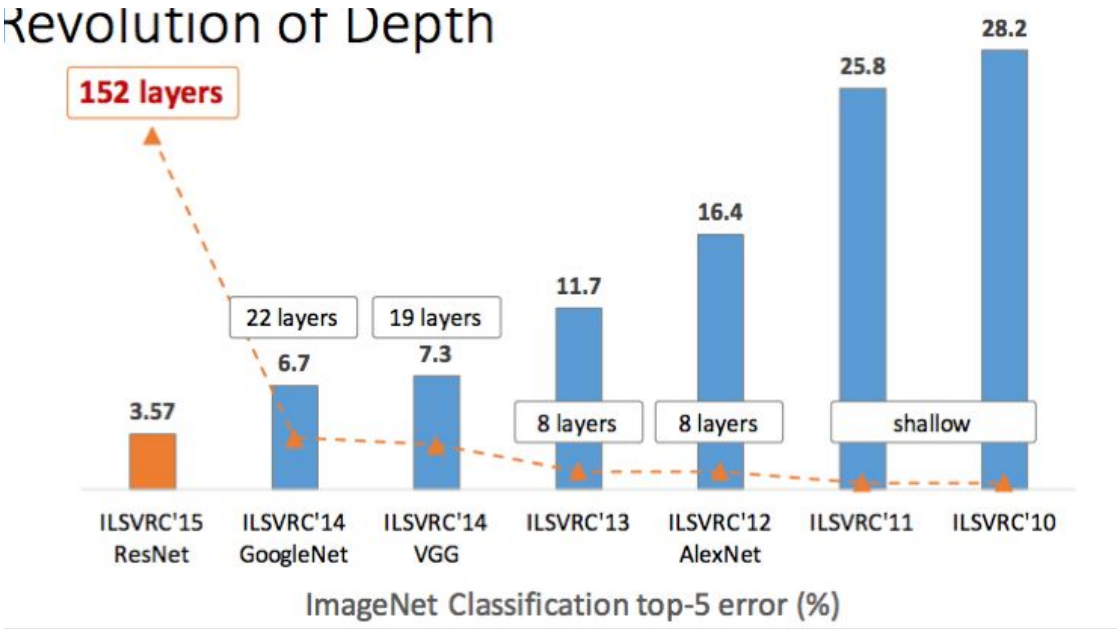
AlexNet (Krizhevsky et al. 2011)

- ImageNet competition
- 1.2 mln training images (1000 labels)
- 5 conv + 3 full connected layers
- Depth is important (drop 2% x layer)
- 60 mln parameters
- **Data augmentation**
- **Dropout:** setting to zero the output of each hidden neuron with p=0.5 (essential but expensive)
- 96 Conv kernels learned
- Top 48 (GPU1) color agnostic
- Down (GPU2) color specific

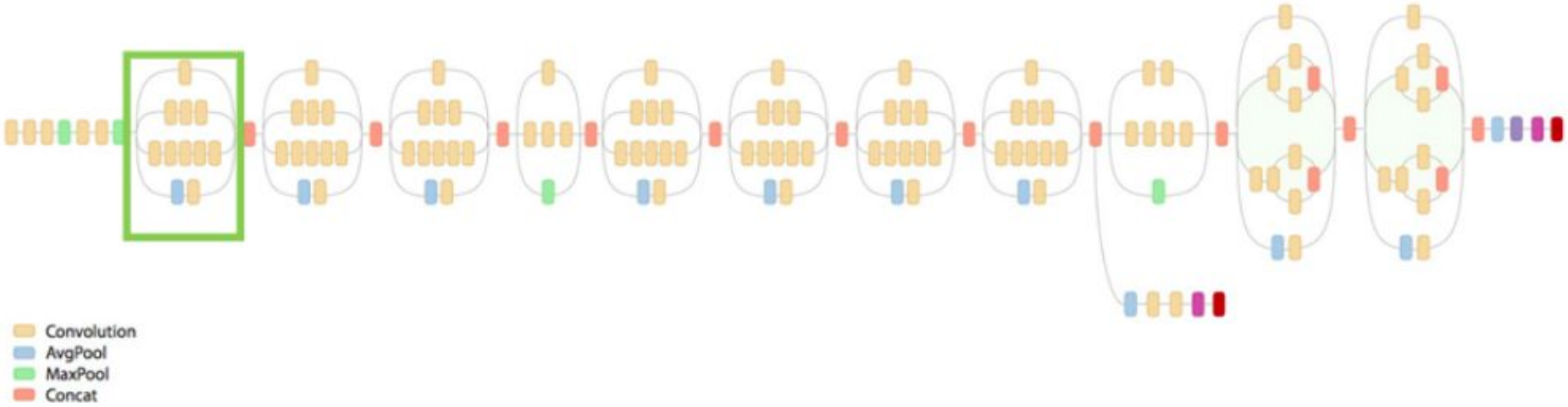


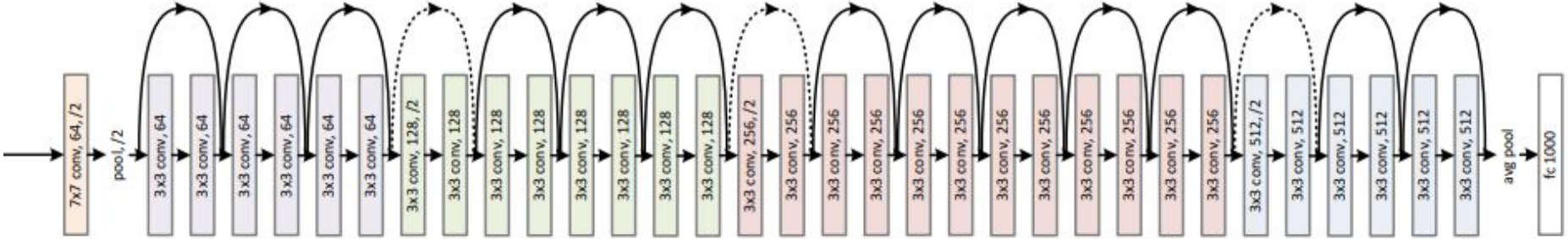


Going deeper



Going deeper

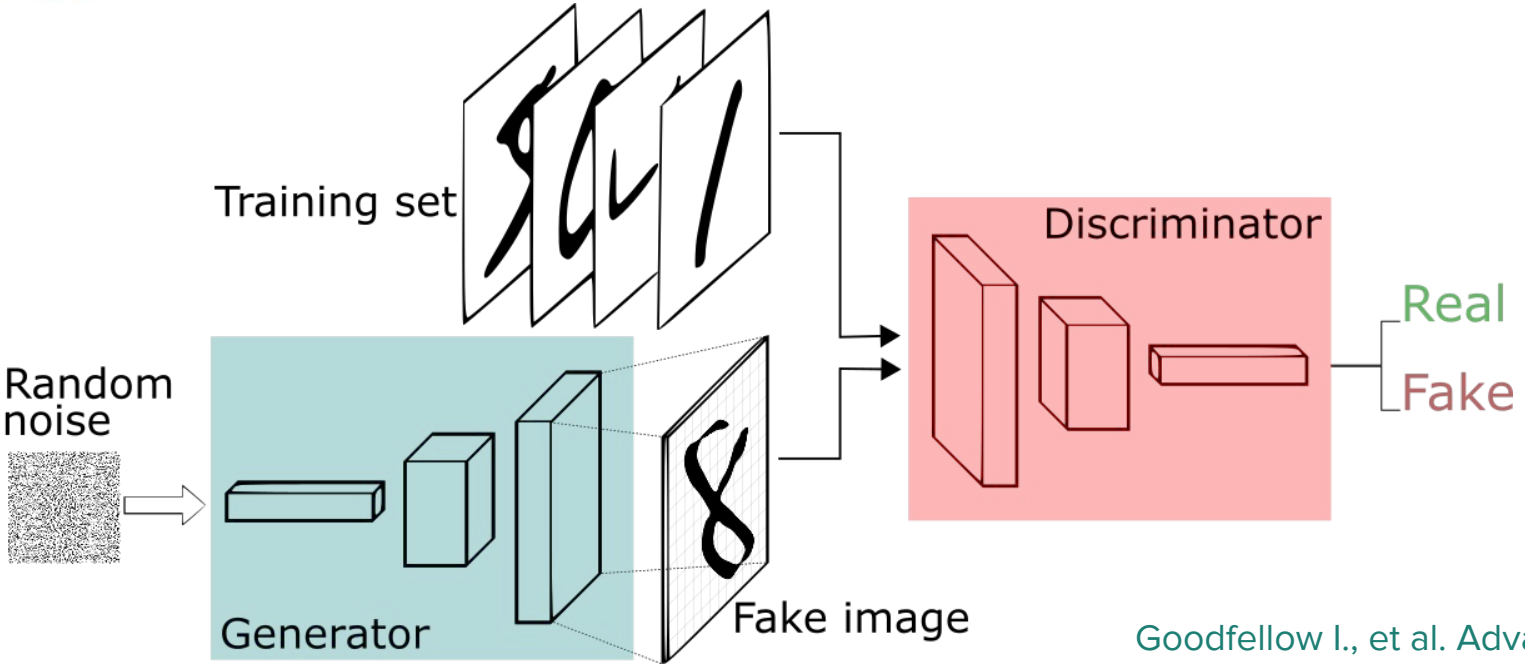




ImageNet Classification top-5 error (%)

Other Architectures. Adversarial Generative Network

The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.



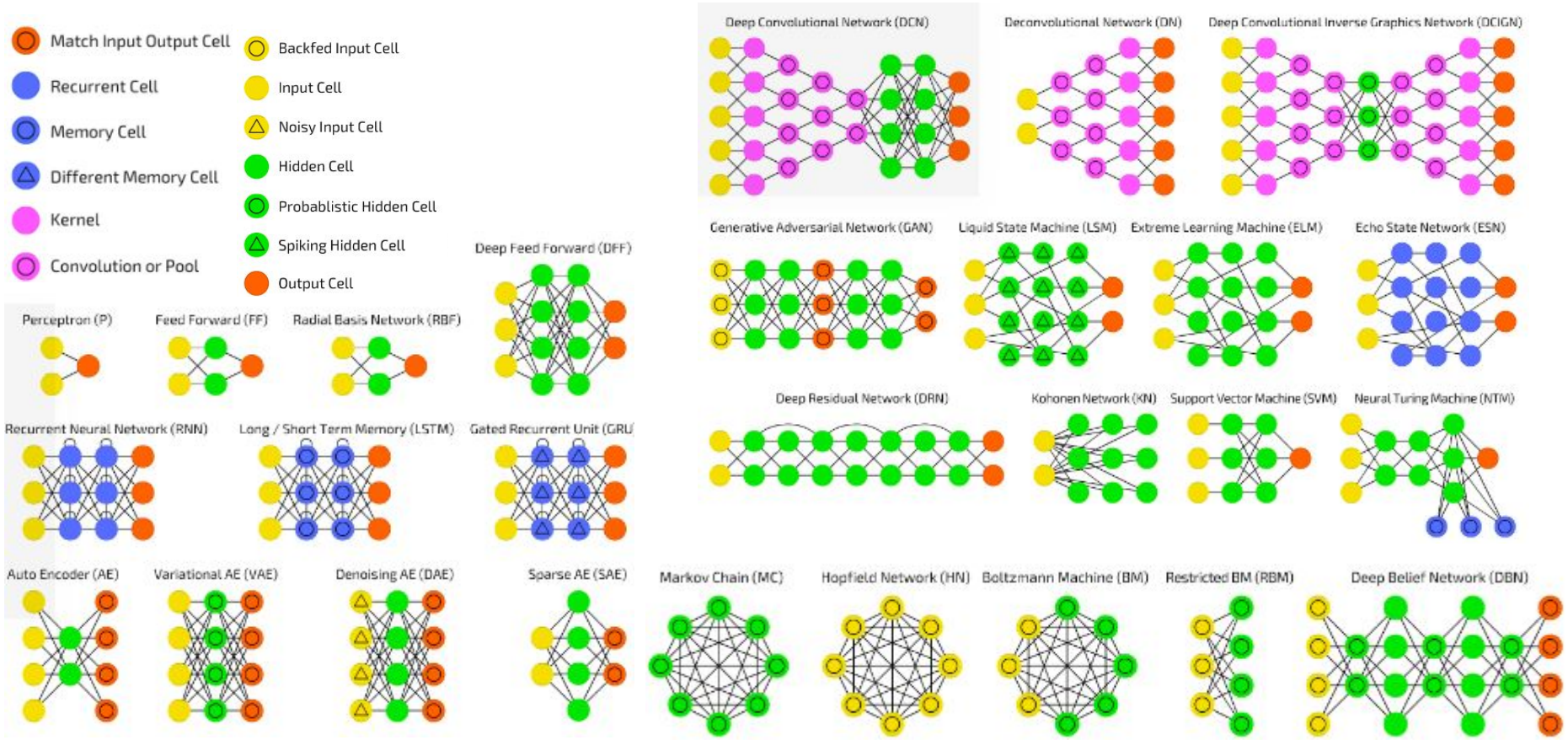
Goodfellow I., et al. Advances in neural information processing systems. 2014.

Other Architectures. Adversarial Generative Network

The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

Goodfellow I., et al. Advances in neural information processing systems. 2014.

Neural Network Zoo

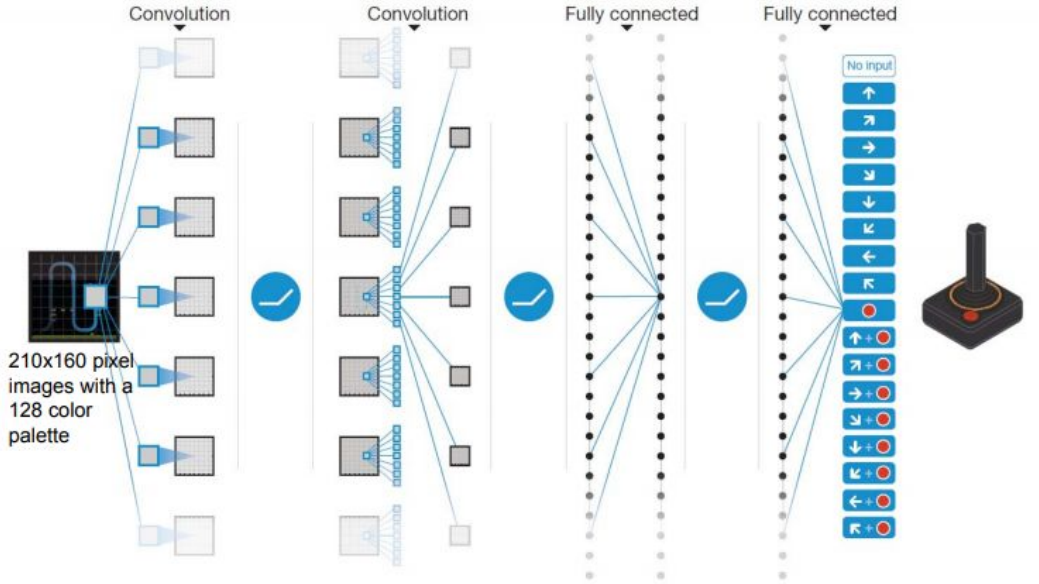


http://www.asimovinstitute.org/neural-network-zoo/

“Deep reinforcement learning”

- 49 Atari games. Pixels as unique input
- Learning the Q-function!
- Outperformed human players in >75%

Breakout | Space Invaders | Seaquest | Beam Rider



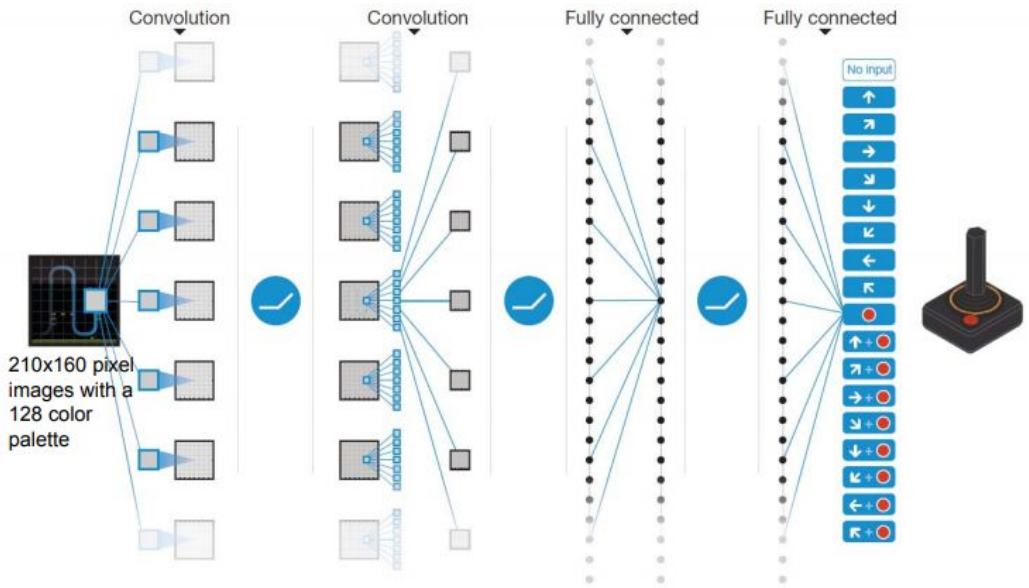
Mnih at al. (DeepMind), Nature 2015

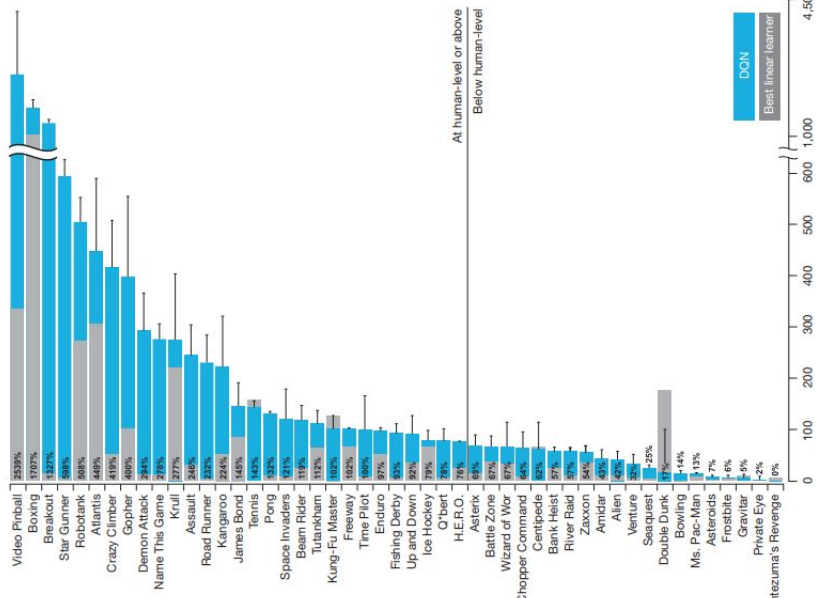
“Deep reinforcement learning”

- 49 Atari games. Pixels as unique input
- Learning the Q-function!
- Outperformed human players in >75%

$$V^*(x) = \max_a Q^*(x,a)$$

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s,\ a_t = a,\ \pi\right]$$

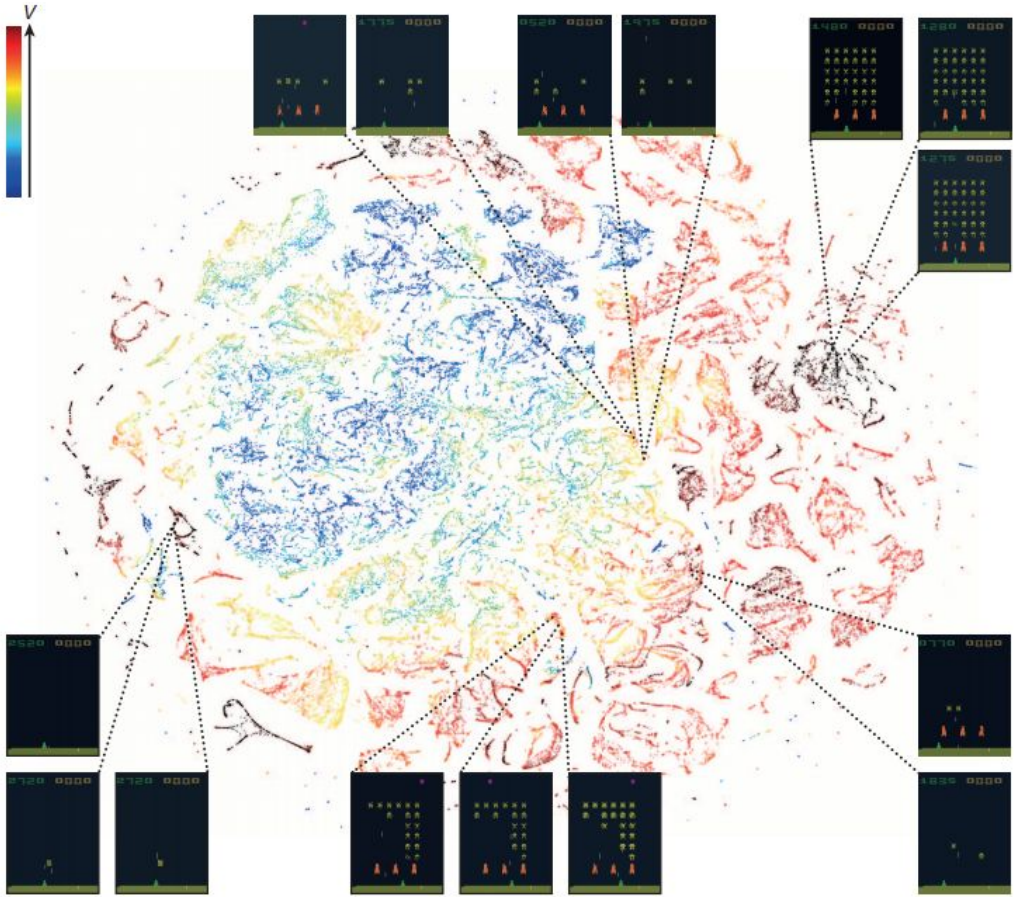




Mnih at al. (DeepMind), Nature 2015

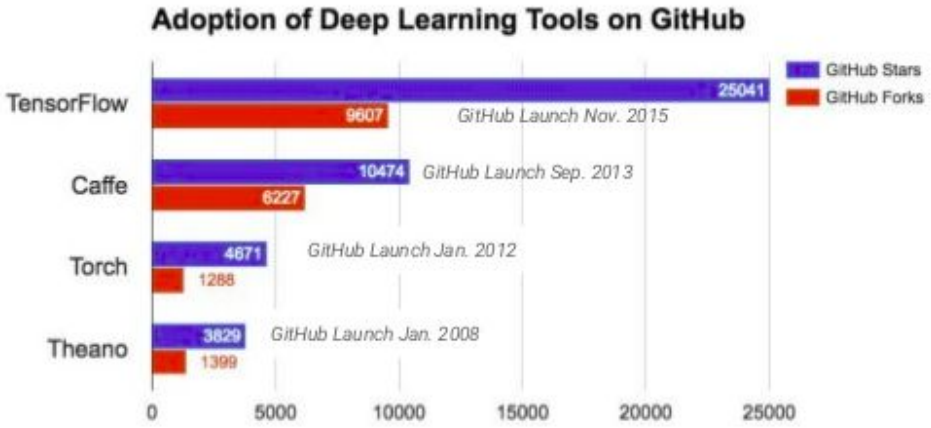
“Deep reinforcement learning”

- t-SNE of the last hidden layer after 2h of training in Space Invaders
- High value function when plenty of aliens or almost completely defeated.



Mnih at al. (DeepMind), Nature 2015

Packages



Caffe

TensorFlow PYTORCH theano

Packages

Blogs

kaggle

Distill

colah's blog

R2
D3

Caffe

TensorFlow

PYTORCH

theano

ML @ ETH/UZH

- Institute of Machine Learning (D-INFK): **4** groups
- Courses with tutorial and projects. Examples:

  - Reinforcement Learning for Spacecraft Maneuvering using Optic Flow and Time-To-Contact
  - Stochastic Calculus for Nonconvex Optimization.
  - Sleep pattern recognition from EEG/EMG.
  - Optimising Shared Mobility using Machine Learning

Machine Learning
Data Mining
Deep Learning
Probabilistic Artificial Intelligence
Learning and intelligent systems
Computational intelligence lab
Statistical learning theory
Computational statistics
A Mathematical Introduction to ML
Approximation Algorithms

Acknowledgements

- A. Polosa
- G. Cavoto
- A. Caflisch
- A. Vitalis
- Big-O & Caflisch group

Thank you

for your attention

Universität Zürich[UZH]

COMPUTATIONAL SCIENCE ZURICH

Amedeo Caflisch Research Group

Positions available for Deep Learning applications!

http://www.biochem-caflisch.uzh.ch/