Adv. Computer Graphics

NYU - Spring 2024

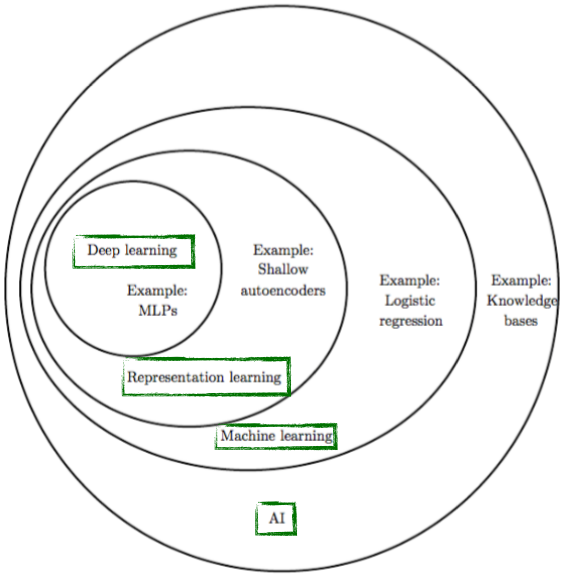
Machine Learning and Deep Learning

Luiz Velho
IMPA

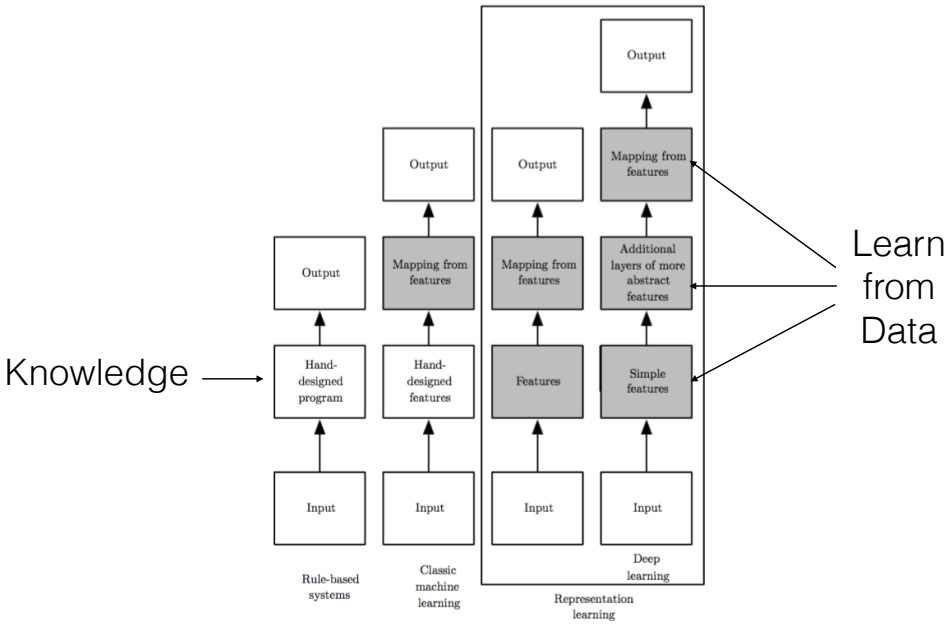
Machine Learning

Machine Learning Fundamentals

Contextualization

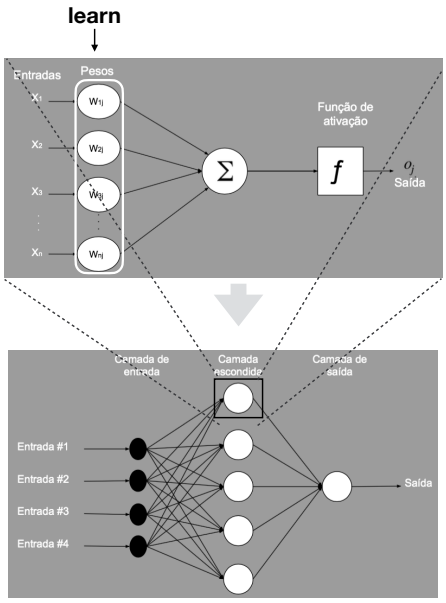


Historical Evolution



Neural Networks

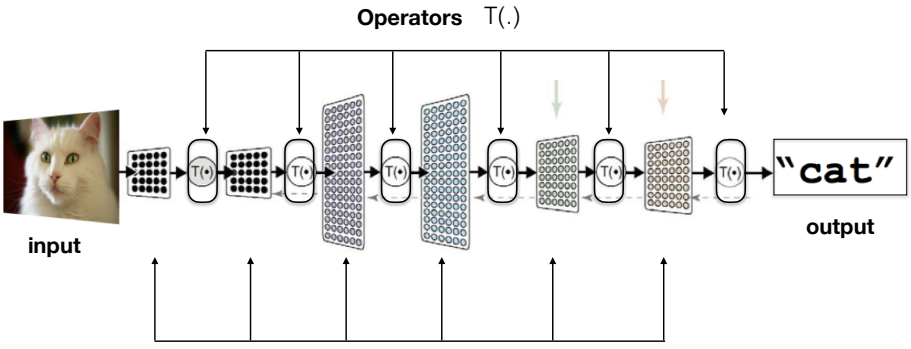
- Neuron (operator)
- Neural Nets (composition)



shallow

Deep Neural Networks

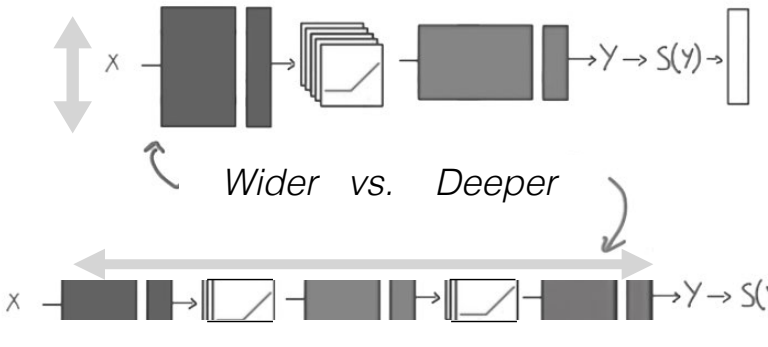
- Multi Layer Architecture



[input -> output]

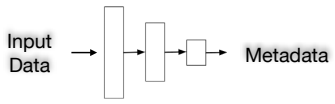
NN: Size and Dimensions

- Number of Layer Nodes / Number of Layers

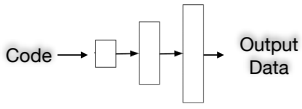


Neural Networks Types

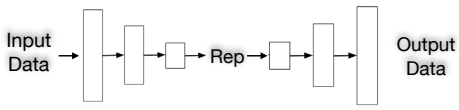
- Analysis



- Synthesis



- Full

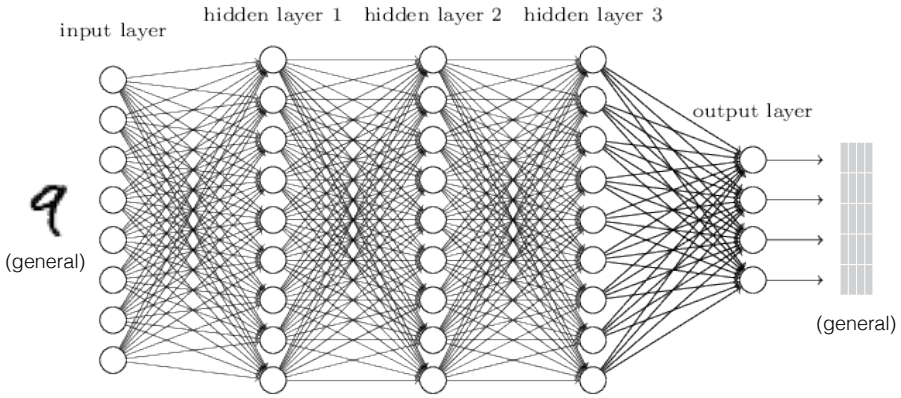


Main Network Architectures

- Fully Connected MLP
  (Multi Layer Perceptron)
- Convolutional NN
- Recurrent NN
- Auto-Encoder
- Generative / Adversarial NNs
- etc…

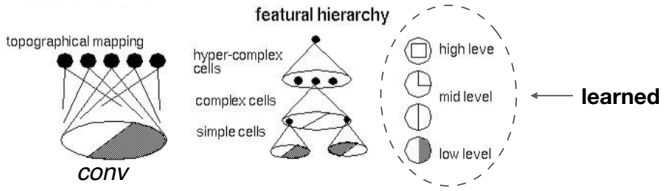
Fully Connected MLP

- Learn General Functions

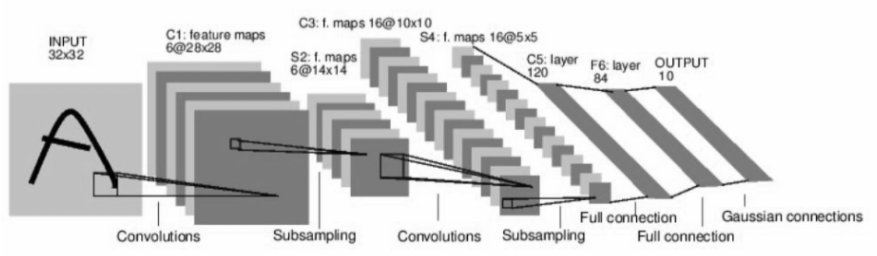


Convolutional Networks

- Convolution Operators. (*human vision*)

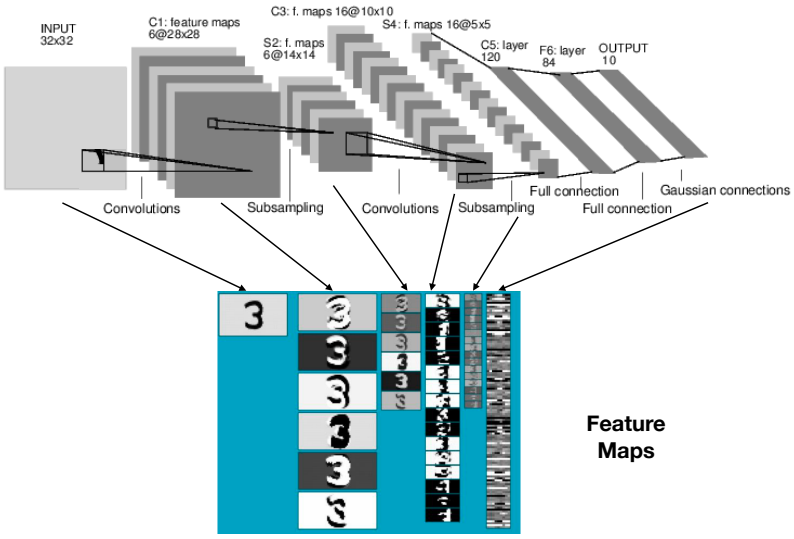


- Deep Convolutional Neural Networks



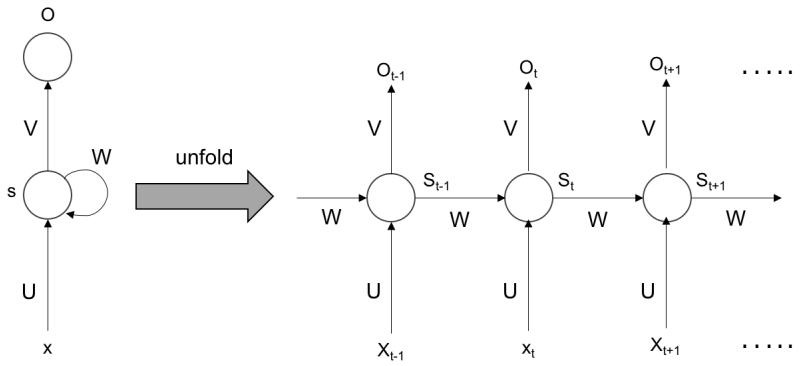
CNN in Action

- Learn <u>Image Features</u>



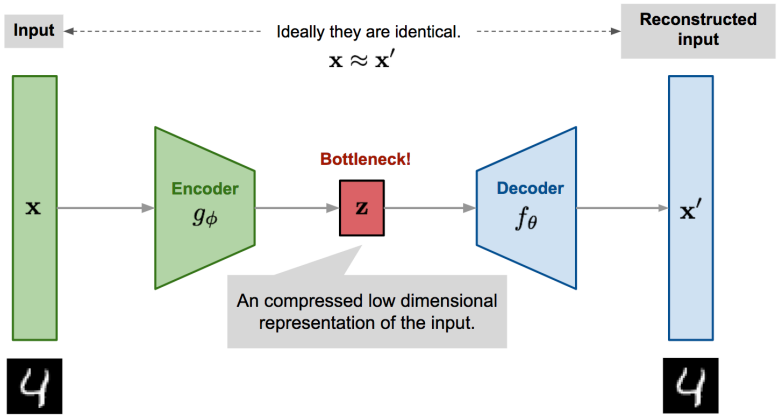
Recurrent Networks

- Learn a Time Dependent Process (sequence)



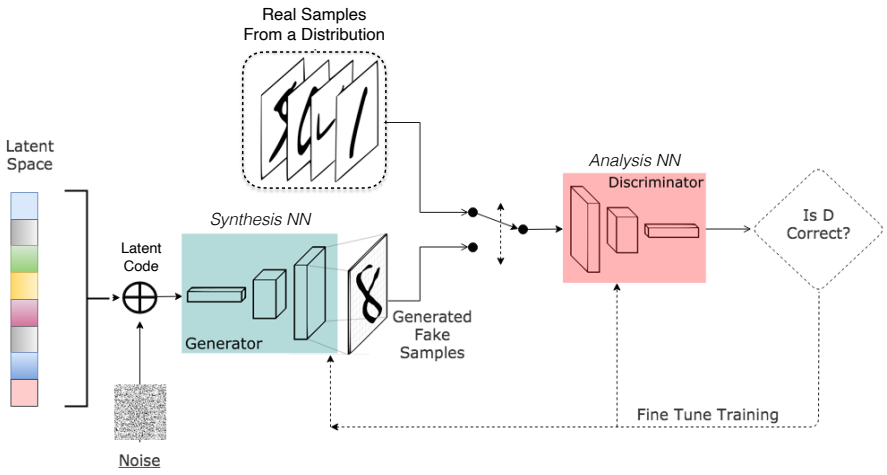
Variational Auto-Encoders

- Learn Representations



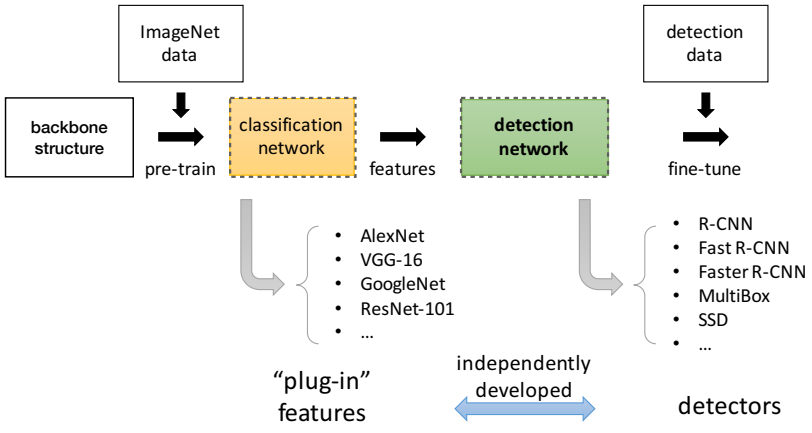
Generative Networks

- Learn to Reproduce from a Distribution



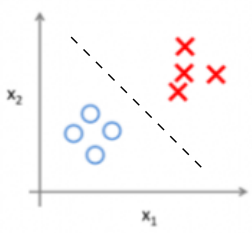
Combining Networks

- Example: Classification / Detection



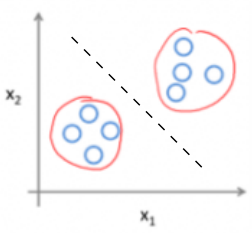
Learning Methods

Supervised



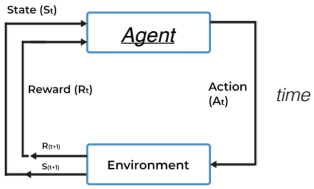
Labeled Data

Unsupervised



Unlabeled Data

Reinforcement



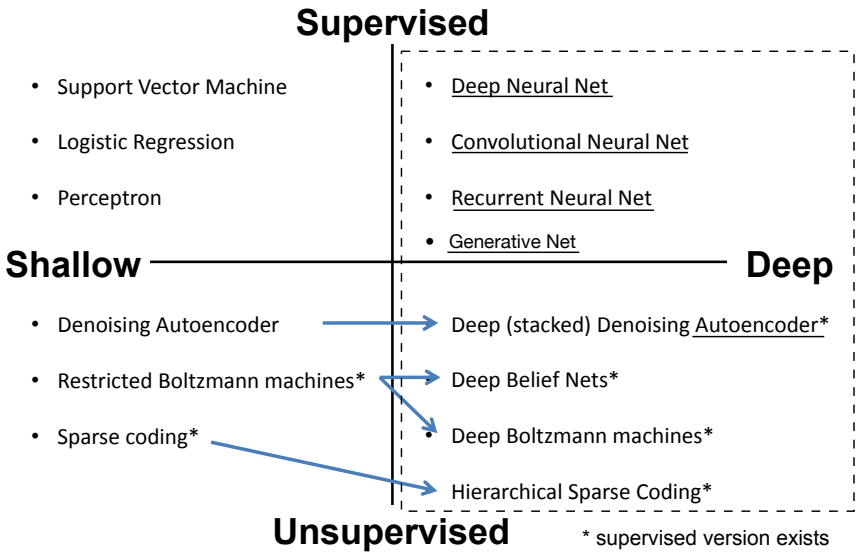
Simulation

OBS: Semi-Supervised

Comparison

- Supervised Learning
  - End-to-end learning of deep architectures (e.g., CNNs)
  - Works well when the amounts of labels is large
  - Structure of the model is important (e.g. convolutional structure)
- Unsupervised Learning
  - Learn statistical structure or dependencies of the data from unlabeled data
  - Layer-wise training
  - Useful when the amount of labels is not large

Taxonomy



Basic Principles of Learning

Learning Algorithms

Machine Learns from Data

- Task
  - class of T
- Performance
  - measured by P
- Experience
  - learns from E

Task

ability to perform task T

- From Examples: $\{x_i\}, x \in \mathbb{R}^n$ (i.e., Collection of Features)
- Types of Tasks:
  - Classification
  - Regression
  - Transcription / Translation
  - Synthesis and Sampling
  - Density Estimation

Performance

quantitative measure of success

Depends on the Task

- Classification / Transcription
  - Accuracy of the Model
  - Error Rate (0-1 Loss)
- Density Estimation
  - Continuous Score (log-probability)

Use *Test Set*

Experience (learning)

from a dataset

- Supervised
  - Data: (feature ***x,*** label ***y***)
  - Predict ***y*** from ***x*** - i.e., $p(\mathbf{y}|\mathbf{x})$
- Unsupervised
  - Data: examples ***x*** (random vector)
  - Estimate Probability Distribution $p(\mathbf{x})$
- *Reinforcement*

Example: Linear Regression

Model

- Linear Function $\hat{y} = w^T x$
- Parameters $w \in \mathbb{R}^n$

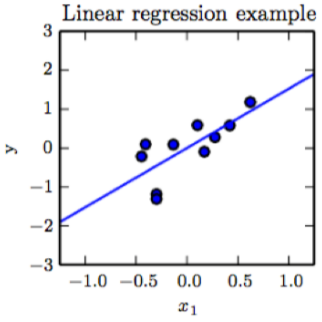
- Task
  - predict *y* from *x* $f(x) = y$
- Experience
  - Design Matrix $\omega(X)$
- Performance
  - Mean Squared Error $\text{MSE}_{\text{test}} = \frac{1}{m} \sum_i (\hat{\boldsymbol{y}}^{(\text{test})} - \boldsymbol{y}^{(\text{test})})_i^2$.

Linear Regression

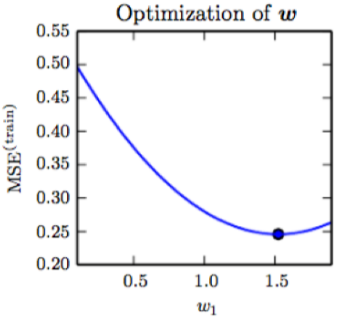
- Learning ***w***
  - Minimize **MSE** train

$$\nabla_{\boldsymbol{w}} \text{MSE}_{\text{train}} = 0$$
$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\text{train})} - \boldsymbol{y}^{(\text{train})}||_2^2 = 0$$
$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} ||\boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})}||_2^2 = 0$$

data



training



Generalization

perform well on new, unseen inputs

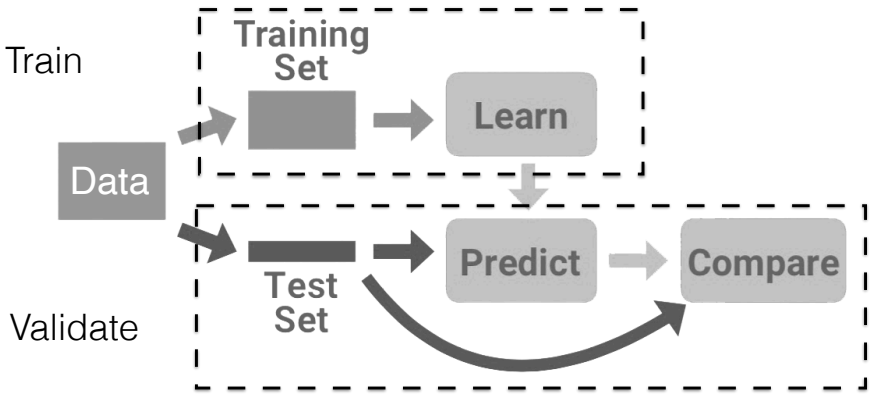
- Data Generating Process (assume I.I.D.)
  - Training Set
  - Test Set
  - (Validation Set)
- Generalization Error
  - Minimize *training error*
  - Evaluate *test error*

$$\frac{1}{m^{(\text{train})}}||\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})}||_2^2,$$

$$\frac{1}{m^{(\text{test})}}||\boldsymbol{X}^{(\text{test})}\boldsymbol{w} - \boldsymbol{y}^{(\text{test})}||_2^2.$$

Metodology

- Stages



Performance Goals

- Challenges

  1.Make the Training Error Small

  2. Reduce the Gap btw Training and Test Error

- Hypothesis Space
  - Set of Functions for the Model

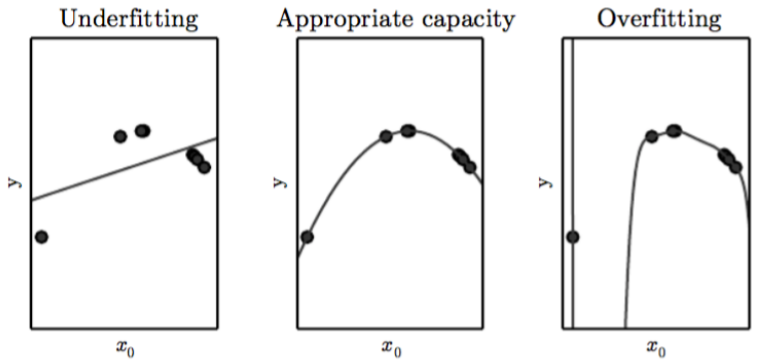
Reaching a Balance

- Things to Avoid
  - Underfitting
    - Large Error on Training Set
  - Overfitting
    - Large Gap btw Training and Test Error

- Desired
  - Ability to Fit a Wide Variety of Functions
    - Correct Capacity

The Right Model

- Effects of Hypothesis Space



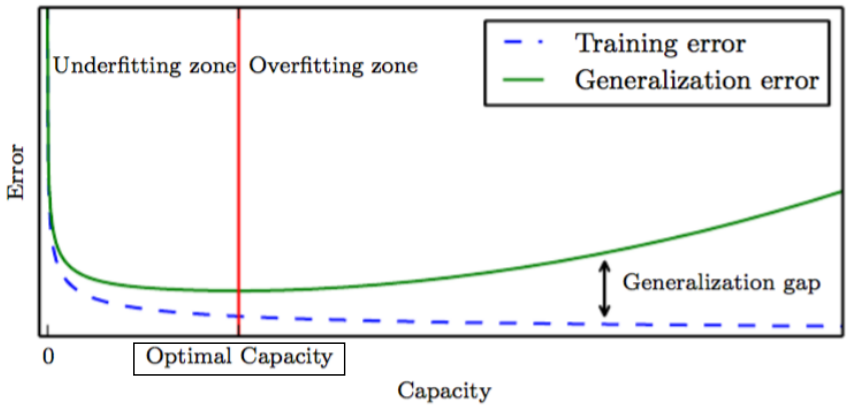
Linear Quadratic Degree 9

Models and Capacity

- Representational Capacity
  - Model
- Effective Capacity
  - Algorithm
- Types of Models
  - Parametric
  - Non-Parametric

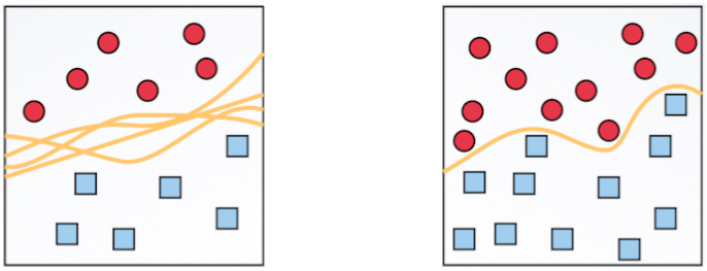
Capacity and Error

- Regimes



Data Size

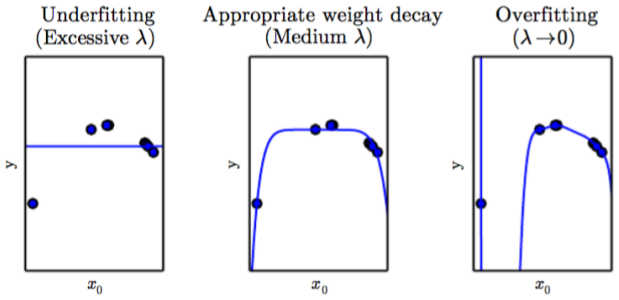
- More Data = Better Generalization



Regularization

- Give preference for one *specific solution* in the Hypothesis Space
- Example:
  Weight Decay (i.e., smaller squared L2 norm)

$$J(\boldsymbol{w}) = \text{MSE}_{\text{train}} + \boxed{\lambda \boldsymbol{w}^\top \boldsymbol{w},}$$



degree 9 polynomial

Hyperparameters

- Hyperparameter
  - Control the Behaviour of the Learning Algorithm
- Example:
  - Degree of Polynomial (capacity hyperparameter)
  - Weight Decay Value (regularization hyperparameter)

Validation Sets

- Evaluates Hyperparameter Effectiveness
  - Data not used in training / testing
- Typical Data Proportions
  - 80% Training + Test
  - 20% Validation

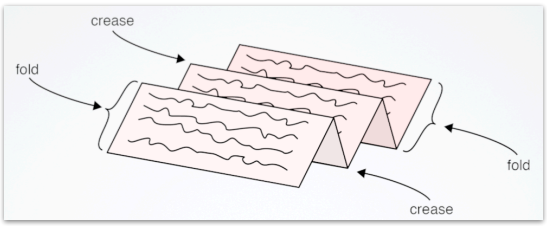
Cross-Validation

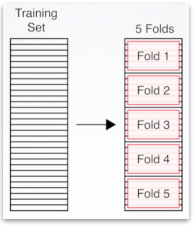
- Reduces Statistical Uncertainty for Small Datasets
- Repeat Train / Test on *Random Subsets* of Data

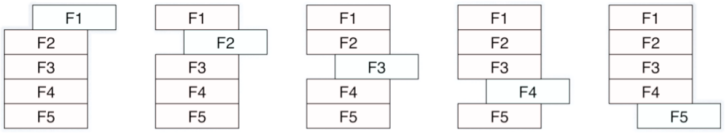
Define `KFoldXV`$(\mathbb{D}, A, L, k)$:
Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$
Require: A, the learning algorithm, seen as a function that takes a dataset as input and outputs a learned function
Require: L, the loss function, seen as a function from a learned function f and an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$
Require: k, the number of folds
  Split $\mathbb{D}$ into k mutually exclusive subsets $\mathbb{D}_i$, whose union is $\mathbb{D}$.
  for i from 1 to k **do**
    $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$
    for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ **do**
      $e_j = L(f_i, \boldsymbol{z}^{(j)})$
    end for
  end for
  Return $\boldsymbol{e}$

C.V. Scheme

- Folding and Testing







No Free Lunch

- Theorem:
  "*No machine learning algorithm is better than any other, when averaged over all possible data distributions*"

- Philosophy
  - Not an *Universal Learning Algorithm*
  - But an Algorithm that *performs well on target data*

Machine Leaning in Practice

Building a Machine Learning Algorithm

- Dataset Specification
- Cost Function
- Optimization Procedure
- Model

ex: Linear Regression

- Data

$$\boldsymbol{X} \text{ and } \boldsymbol{y}$$

- Cost

$$J(\boldsymbol{w}, b) = -\mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(y \mid \boldsymbol{x})$$

- Optimization

$$\nabla_{\boldsymbol{w}} \text{MSE}_{\text{train}} = 0$$

- Model

$$p_{\text{model}}(y \mid \boldsymbol{x}) = \mathcal{N}(y; \boldsymbol{x}^\top \boldsymbol{w} + b, 1)$$

Q&A

Deep Learning

The Paper


REVIEW

doi:10.1038/nature14539

Nature, May 2015

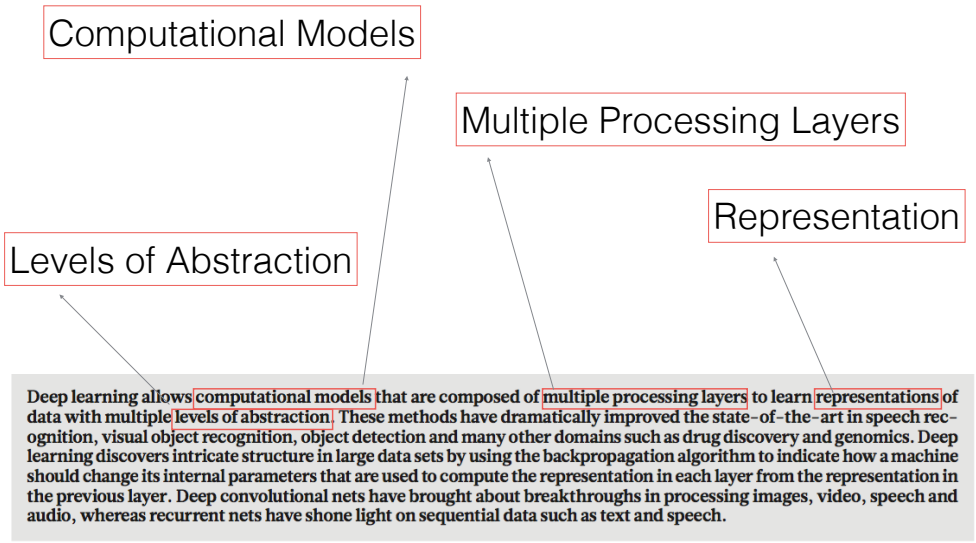
Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

- Nature, VOL. 521 - 28 MAY 2015
- Authors:
  - Yann LeCun (NYU / Facebook)
  - Yoshua Bengio (U. Montreal / MILA)
  - Geoffrey Hinton (U. Toronto / Google)

ACM Turing Award 2018

Foreword



Representation Learning

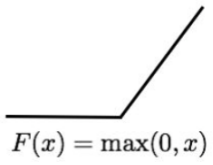
- Set of Methods
  - Allows a Machine
  - to Automatically Discover Representation
  - for Analysis, Synthesis or Description

Deep Learning Methods

- Representation Learning
- Multiple Levels of Representation
- Composition
  - Simple, but Non-Linear Modules

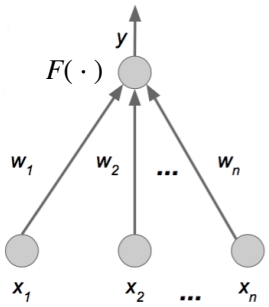
The Neuron

- Non-Linear Classification



ReLU

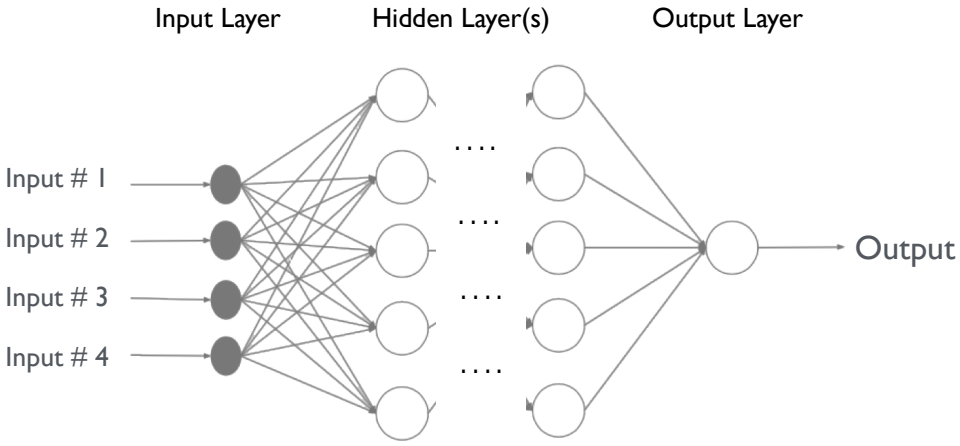
- Anatomy
  - Linear Weights
  - Non-Linear Function (differentiable)



$$y = F\left(\sum_i w_i x_i\right)$$

Neural Network

- Multiple Interconnected Layers

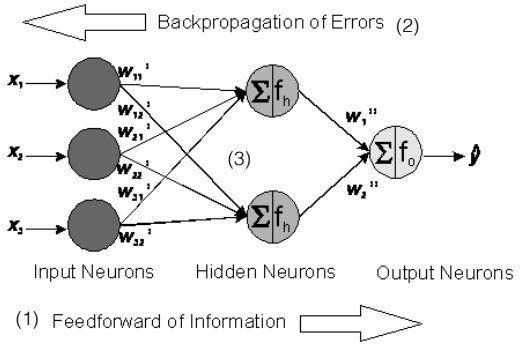


OBS: Computational Graph

Supervised Learning Algorithm

- Learning Algorithm

While not done:
(1) Pick a random training example "(input, output)"
(2) Run neural network on "input" (feedforward)
(3) Adjust weights on edges to make output closer to "output" (backpropagation)

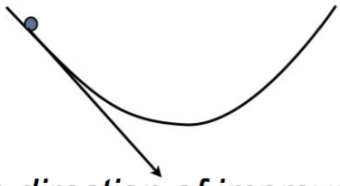


Learning w/ Back Propagation

- Adjusting the Weights

Use partial derivatives along the paths in the neural net

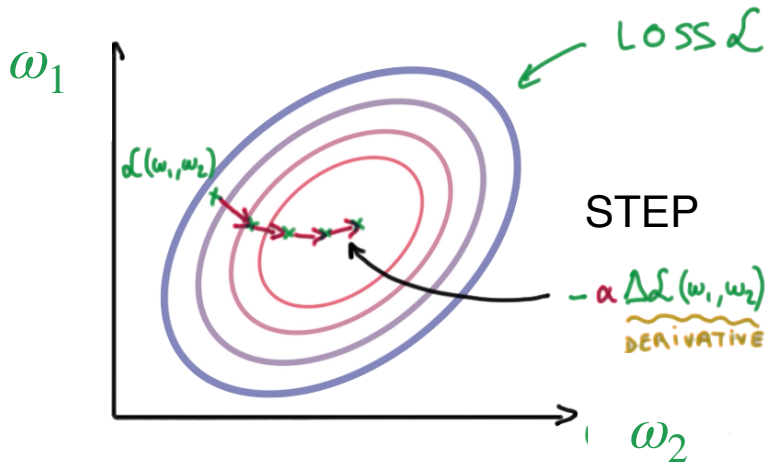
Follow the gradient of the error w.r.t. the connections



Gradient points in direction of improvement

Iterative Gradient Descent

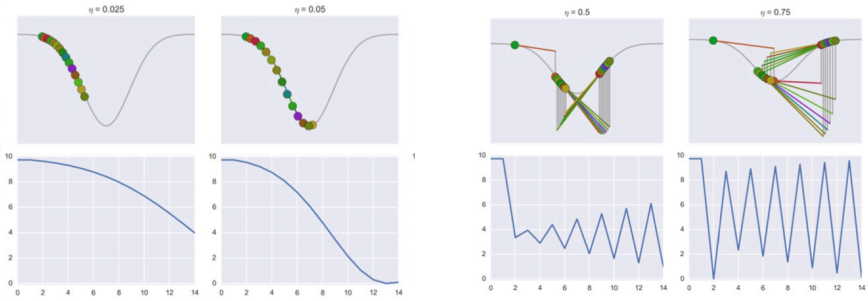
- Minimize Loss



Learning Rate

- Constant Step Size

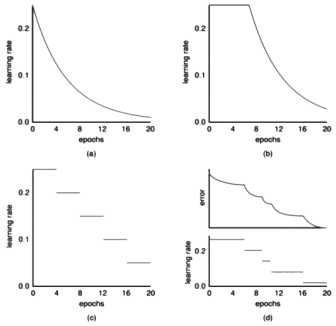
$$\omega := \omega - \eta \nabla L_i(\omega)$$

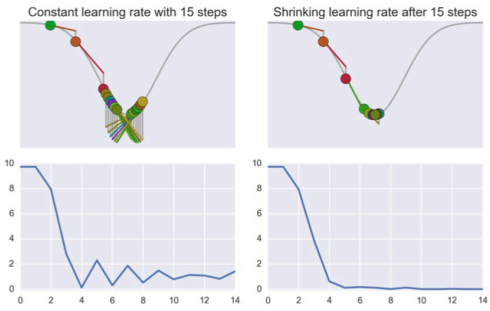


Small Large

Adaptive Learning Rate

- Decay





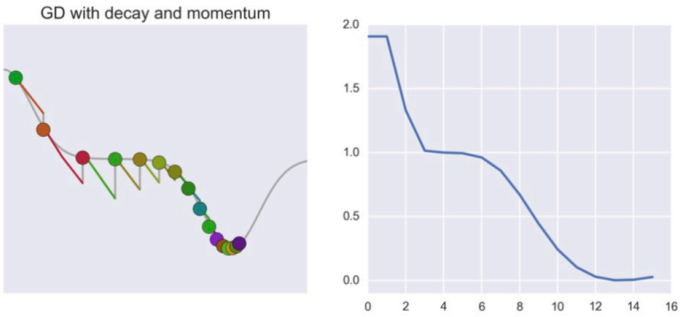
Decay Schedules Comparison

Learning w/ Momentum

- Inertia

$$\omega := \omega - \eta \nabla L_i(\omega) + \gamma \Delta\omega$$

previous step



Stochastic Gradient Descent

Computationally Efficient for Large Datasets

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

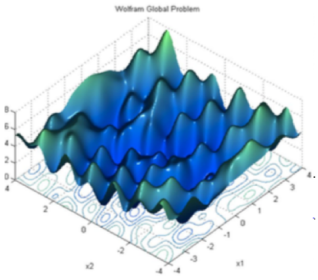
- Gradient Descent Cost = $O(m)$
- SGD Algorithm $m' \ll m$
  - Sample a minibatch $\mathbb{B} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m')}\}$
  - Estimate $\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$
  - Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$

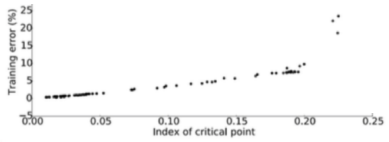
Optimization Solution

Non-convexity

-Low-D => local minima
-High-D => saddle points

-Most local minima are close to the global minima



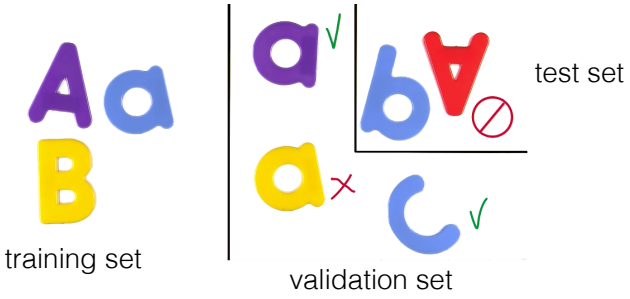


This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

Slide Credit: Yoshua Bengio

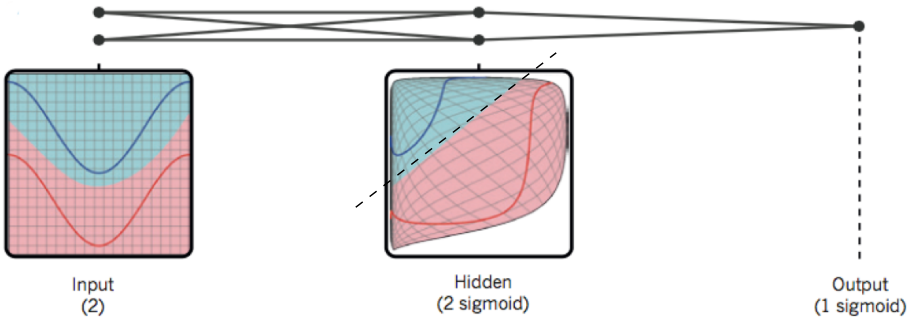
Learning Methodology

- Training (*Learning: i.e., adjust the weights*)
- Validation (*Tune Architecture: i.e., minimize overfitting*)
- Testing (*Access Performance: i.e., generalization*)



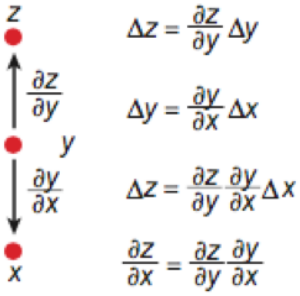
Example - Step I

- Multilayer Neural Network
  - 2 classes



Example - Step II

- Computing the Derivatives
  - Chain Rule



$$\Delta z = \frac{\partial z}{\partial y}\Delta y$$

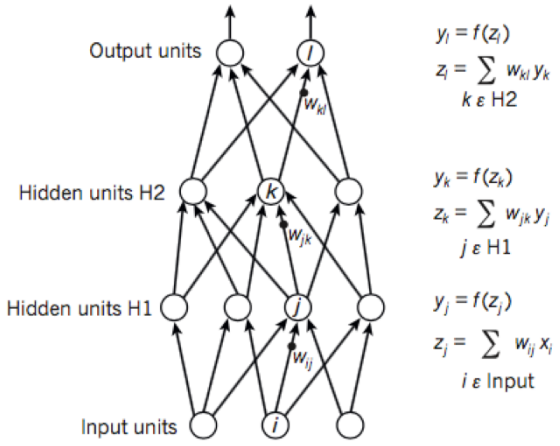
$$\Delta y = \frac{\partial y}{\partial x}\Delta x$$

$$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

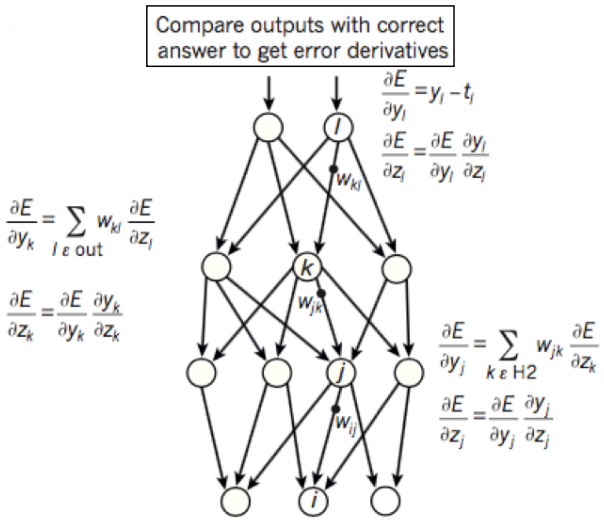
Example - Step III

- Forward Pass



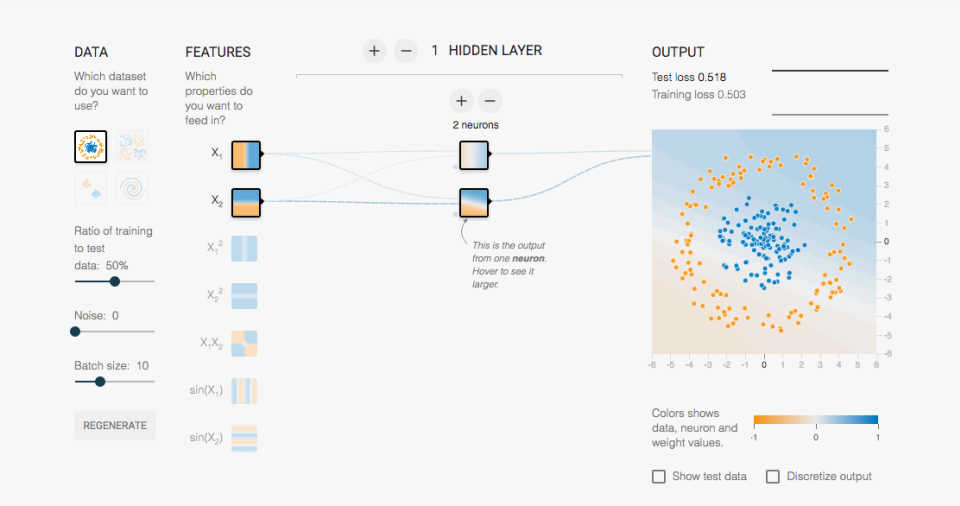
Example - Step IV

- Backward Pass



Hands-On

- Tensorflow Playground



Key Property of DNNs

Results get better with

more data +
bigger models +
more computation

- Huge Datasets
- Lot's of Research
- Fast GPUs

Major Advances

- Technology Companies
  - Google / Facebook / Microsoft / IBM / Twitter / Adobe
- Hardware Implementations
  - NVIDIA / Intel / Qualcomm / Samsung / Mobileye

Representations and Structure

Challenge

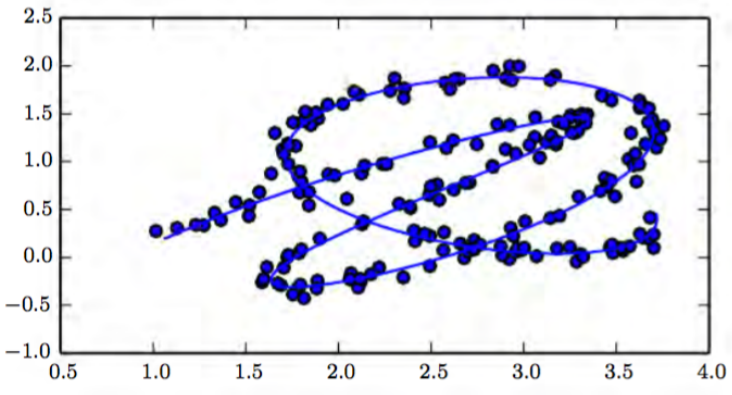
- Curse of Dimensionality

- *Assumption: Local Constancy / Smoothness*

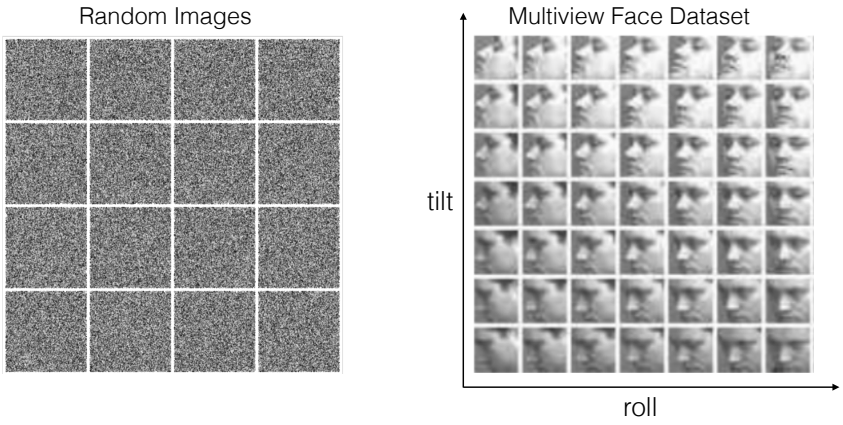
$$f^*(\boldsymbol{x}) \approx f^*(\boldsymbol{x} + \epsilon)$$

Manifold Learning

- Embedded Model Subspace ($1D \subset 2D$)



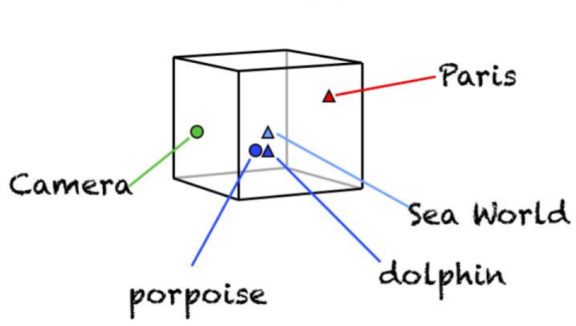
Embedding Structure (*parametrization*)



- Discover and Disentangle Manifold Coordinates

Embeddings of Vector Spaces

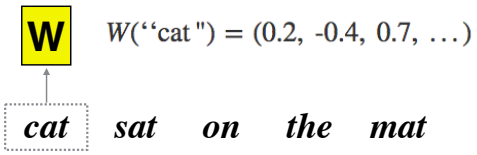
- Embedding Space (100D to 1000D, and more..)

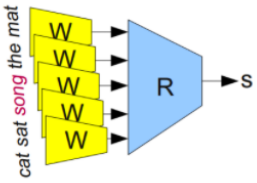


Embedding Function: A look-up-table that maps sparse features into dense floating point vectors.

Learning Text Embeddings

- Embedding Function $W : \text{words} \rightarrow \mathbb{R}^n$
- Prediction if a 5-gram is valid $R : \text{5-gram} \rightarrow \{0,1\}$





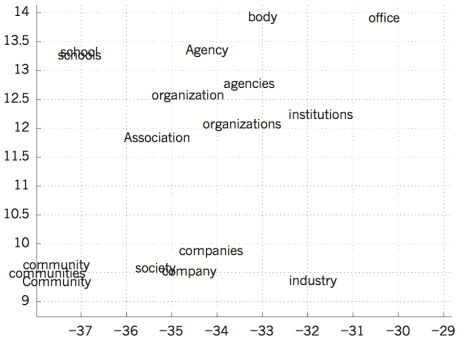
- Training: (*valid / invalid data*)

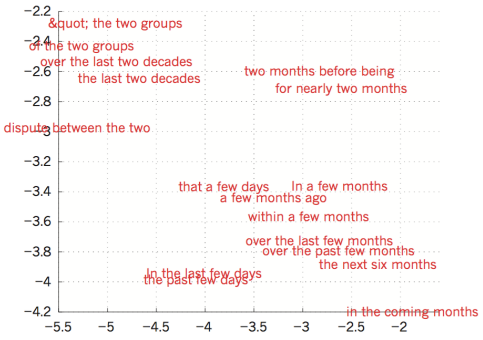
$$R(W(\text{“cat ”}), W(\text{“sat ”}), W(\text{“on ”}), W(\text{“the ”}), W(\text{“mat ”})) = 1$$

$$R(W(\text{“cat ”}), W(\text{“sat ”}), W(\text{“song ”}), W(\text{“the ”}), W(\text{“mat ”})) = 0$$

Word 2 Vect

- Semantically related words are close in vector space





Categories

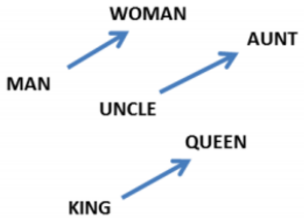
- Similar Meanings

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

What words have embeddings closest to a given word? From Collobert *et al.* (2011)

Analogies

- Differences btw words ~ Structure



$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“aunt”}) - W(\text{“uncle”})$$

$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“queen”}) - W(\text{“king”})$$

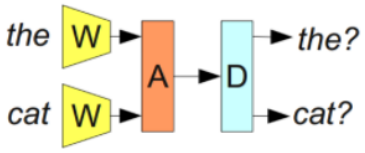
Learning Sequences

Distributed Representations

- Hidden Layers
  - Predict Target Output from Input
- Sequential Data
  - Markov Chains
- Example:
  - Predict next word in a sequence from local context

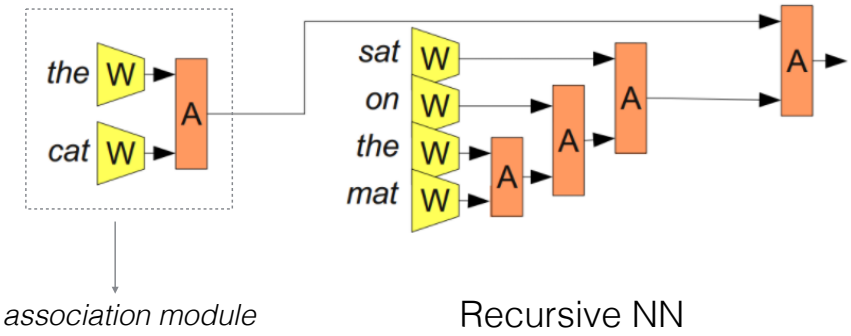
Encoder / Decoder

- Reversible Representations



Recursive Neural Networks

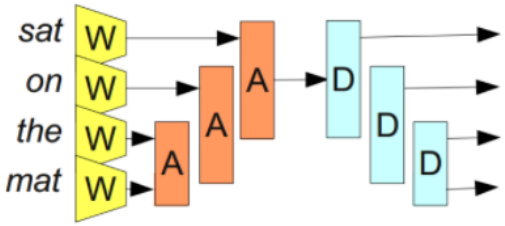
- Merging sequences



Recursive NN

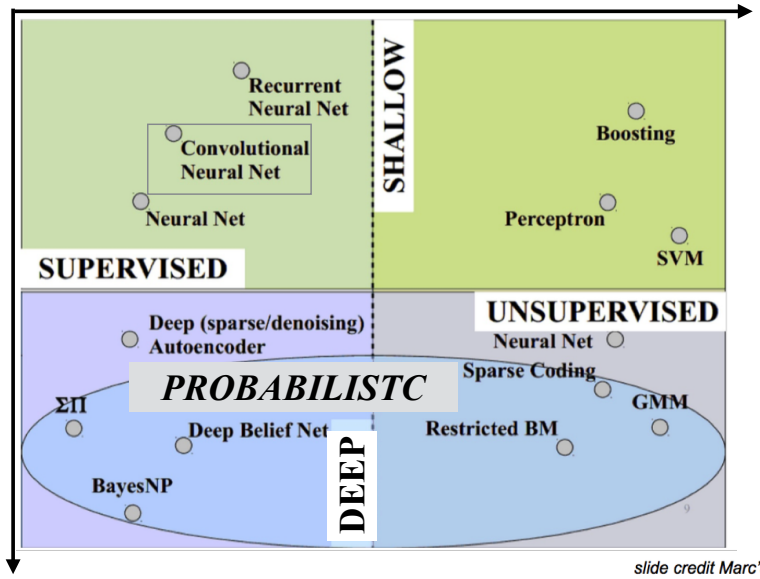
Learning Sequences

- Encoding / Decoding Sequences



Overview

- Global Picture



slide credit Marc'aurelio Ranzato, CVPR '14 tutorial.

Q&A