Mathematics for Artificial Intelligence

Reading Course

Elena Agliari

Dipartimento di Matematica

Sapienza Università di Roma

(TENTATIVE) PLAN OF THE COURSE

Introduction

Chapter 1: Basics of statistical mechanics
The Curie-Weiss model

Chapter 2: Neural networks for associative memory and pattern recognition

Chapter 3: The Hopfield model
Hopfield model with low-load and solution via log-constrained entropy
Self-average, spurious states, phase diagram
Hopfield model with high-load and solution via stochastic stability

Chapter 4: Beyond the Hebbian paradigma

Chapter 5: A gentle introduction to machine learning
Maximum likelihood and Bayesian learning
Concepts and preliminaries in machine learning

Chapter 6: Neural networks for statistical learning and feature discovery.
Rosenblatt and Minsky&Papert perceptrons.
Supervised Boltzmann machines.

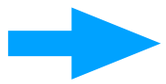
Chapter 7: A few remarks on unsupervised learning, “complex” patterns, deep learning
Unsupervised Boltzmann machines.
Non-Gaussian priors.
Multilayered Boltzmann machines and deep learning.

Seminars: Numerical tools for machine learning; Non-mean-field neural networks; (Bio-)Logic gates; Maximum entropy approach, Hamilton-Jacobi techniques for mean-field models, …

Chapter V
A gentle introduction to machine learning

A partial piece of a song is retrieved by Shazam, which names it in a “second”

A partial piece of a song is retrieved by Shazam, which names it in a “second”

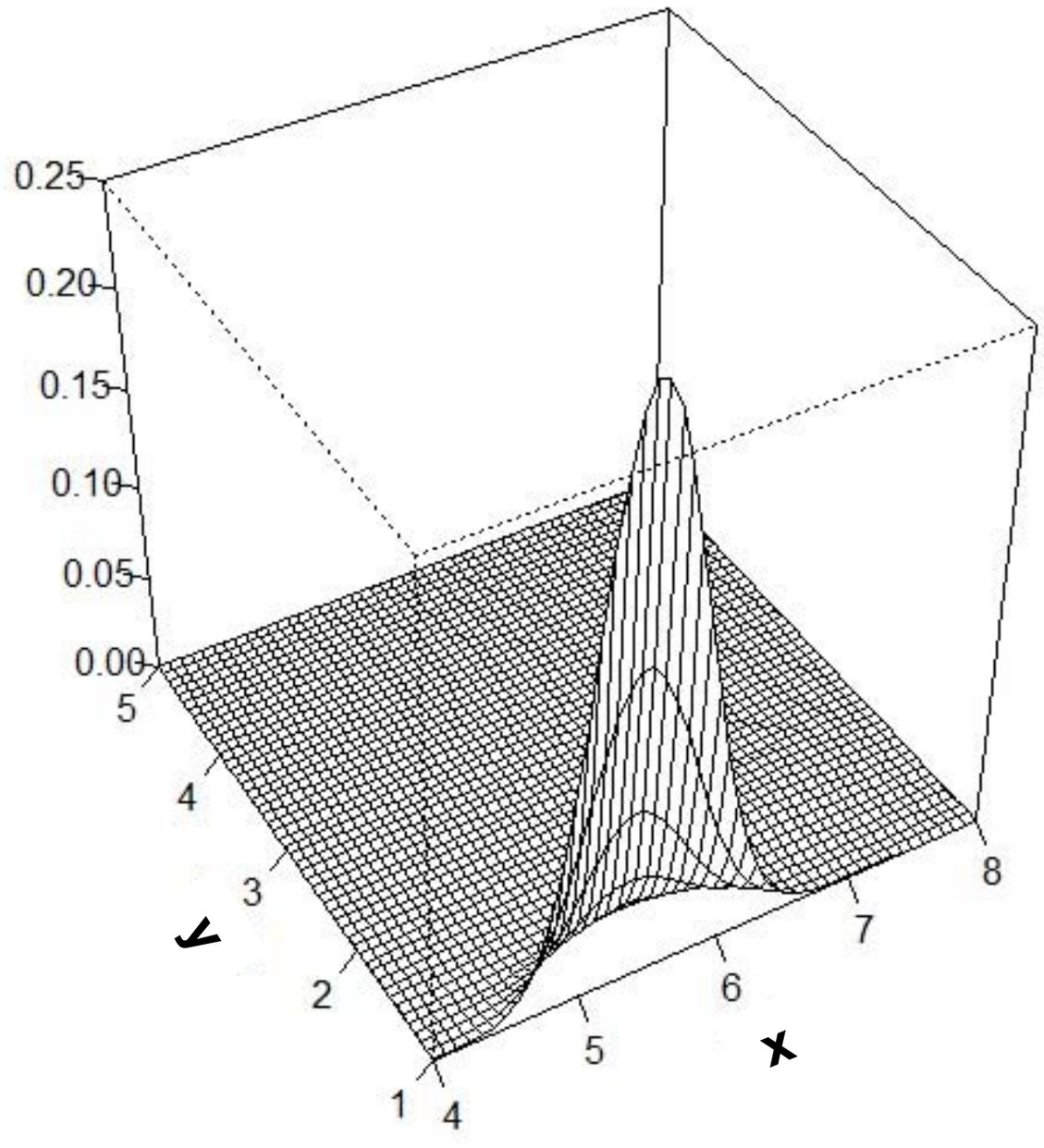


… even in the presence of noise

In order to accomplish this task the machine has to be previously trained, i.e., it has to undergo machine learning algorithms

The machine actually learns a probabilistic relation between the input ***x*** and the output ***y***
→ $P(\boldsymbol{x}|\boldsymbol{y})$
the output provided is the most likely one

As the machine is fed with more and more (noisy) experiences ***x***, its knowledge about the relation between ***y*** and ***x*** is more and more refined, that is $P(\boldsymbol{x}|\boldsymbol{y})$ gets closer and closer to the true one



Machine Learning
Concerned with the development of training algorithms for best estimating $P(\boldsymbol{x}|\boldsymbol{y})$
Statistical inference
Deduce properties of an underlying distribution by analysis of data.
Main questions:
- under what conditions is the information contained in the observations sufficient for satisfactory recovery of the variables?
- can the inference be done in an algorithmically efficient way? What are the optimal algorithms for this task?

“Statistical physics inference”

The methods and tools of statistical physics are designed to describe large assemblies of small elements, such as atoms and molecules, but also agents, neurons, bits, data points.

Today’s data deluge (big data) makes the “thermodynamic limit” perfectly suitable

As a matter of fact, several existing results and algorithms have been derived independently in statistical inference and in statistical physics.

Phase transition in thermodynamics $\longleftrightarrow$ threshold phenomena that arise in the ability to infer something reliable from data

developed the Bayesian interpretation of probability giving rise to the field of statistical inference

developed one of the first sophisticated derivations of the gas laws within the caloric theory

… there may be a link, overall! 😉

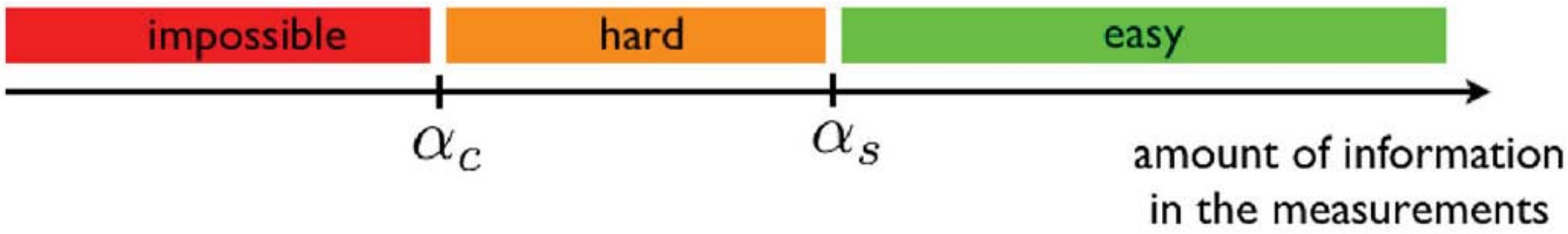
“Statistical physics inference”

The methods and tools of statistical physics are designed to describe large assemblies od small elements, such as atoms and molecules, but also agents, neurons, bits, data points.

Today’s data deluge (big data) makes the “thermodynamic limit” perfectly suitable

As a matter of fact, several existing results and algorithms have been derived independently in statistical inference and in statistical physics.

Phase transition in thermodynamics $\longleftrightarrow$ threshold phenomena that arise in the ability to infer something reliable from data



Different schools (or "paradigms") of statistical inference, not mutually exclusive (in fact, methods that work well under one paradigm often have attractive interpretations under other paradigms).

Connection between statistical physics and statistical inference is more most transparent in the Bayesian setting and in the likelihood-based statistics setting, which we shall follow.

The Bayesian setting (an anticipation)

One considers a set of variables $\boldsymbol{x}=\{x_i\}_{i=1,\dots,N}$ on which one is able is to perform some partial observations or measuremens $\boldsymbol{y}=\{y_\mu\}_{\mu=1,\dots,M}$.
The goal is to deduce, or infer, the values of the variables $\boldsymbol{x}$ based on the perhaps indirect and noisy information contained in the observed data $\boldsymbol{y}$.

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

$P(\boldsymbol{x}|\boldsymbol{y})$ posterior probability distribution for $\boldsymbol{x}$ given $\boldsymbol{y}$
$P(\boldsymbol{y}|\boldsymbol{x})$ likelihood
$P(\boldsymbol{y})$ evidence (also denoted as a normalization $Z(\boldsymbol{y})$)
$P(\boldsymbol{x})$ prior probability distribution for $\boldsymbol{x}$; it reflects our belief of what $\boldsymbol{x}$ should be before we receive any data

(see the Teacher-Student scenario or regression in the Bayesian framework in the following)

The Likelihood setting

Density estimation for a random variable X which is not completely specified, in the sense that the full set of probabilities $\{p(x);\ x\in A\}$ [or, in the case of a continuous variable, the probability density function] are unknown.

Maximum Entropy principle (1950s)

MEP estimation of a pdf when information is available only in the form of the values of averages $\langle f_\alpha(x)\rangle$ for some families $\{f_\alpha\}$ of functions of x. These could, but not need to, include the moments $\mu_n=\langle x^n\rangle$ of X.

Focus on discrete variables

Entropy associated with a random variable X

$$H(X) = -\sum_{x\in A} p(x)\log_2 p(x)$$

The best estimate of a set of probabilities $\{p(x);\ x\in A\}$, compatible with the available information is given by an assignment of probabilities wich maximizes the entropy, that is, our ignorance about X, subject to the contraints coming from the available information.

Maximum Entropy principle (1950s)

The best estimate of a set of probabilities $\{p(x); x \in A\}$, compatible with the available information is given by an assignment of probabilities wich maximizes the entropy, that is, our ignorance about X, subject to the contraints coming from the available information.

$$H[p] = -k \sum_{x \in A} p(x) \log p(x)$$

$$\langle f_\alpha(x) \rangle = \sum_{x \in A} p(x) f_\alpha(x) = \bar{f}_\alpha, \quad f_\alpha \in \mathcal{M}$$

$$\langle f_0(x) \rangle = \sum_{x \in A} p(x) = 1$$

Denote with p^* the best estimate compatible with the available information

$$p^* = \underset{\mathrm{p}}{\operatorname{argmax}} \{H[p]\} \quad \text{subject to } \langle f_\alpha(x) \rangle = \bar{f}_\alpha, \quad f_\alpha \in \mathcal{M}$$

Introduce Lagrange multipliers $\boldsymbol{\lambda}$

$$H_{\boldsymbol{\lambda}}[p] = H[p] + k\lambda_0 \left(\sum_{x \in A} p(x) - 1 \right) + k \sum_{\alpha > 0} \lambda_\alpha \left(\sum_{x \in A} p(x) f_\alpha(x) - \bar{f}_\alpha \right)$$

Solve

$$\frac{\partial H_{\boldsymbol{\lambda}}[p]}{\partial p(x)} = 0, \ \forall x \in A$$

$$\frac{\partial H_{\boldsymbol{\lambda}}[p]}{\partial \lambda_\alpha} = 0, \ \forall \alpha \geq 0$$

concavity of $H[p]$ & constraints are linear functionals of p → MaxEnt solution is unique (as long as the set of constraints is kept finite)

Maximum Entropy principle (1950s)

The best estimate of a set of probabilities $\{p(x); x \in A\}$, compatible with the available information is given by an assignment of probabilities wich maximizes the entropy, that is, our ignorance about X, subject to the contraints coming from the available information.

$$H[p] = -k \sum_{x \in A} p(x) \log p(x)$$

$$\langle f_\alpha(x) \rangle = \sum_{x \in A} p(x) f_\alpha(x) = \bar{f}_\alpha, \quad f_\alpha \in \mathcal{M}$$

$$\langle f_0(x) \rangle = \sum_{x \in A} p(x) = 1$$

Denote with p* the best estimate compatible with the available information

$$p^* = \underset{\mathrm{p}}{\operatorname{argmax}} \{H[p]\} \quad \text{subject to } \langle f_\alpha(x) \rangle = \bar{f}_\alpha, \quad f_\alpha \in \mathcal{M}$$

$$p^* = \exp\left(\lambda_0 - 1 + \sum_{\alpha \neq 0} \lambda_\alpha f_\alpha(x)\right) = \frac{1}{Z} \exp\left(\sum_{\alpha \neq 0} \lambda_\alpha f_\alpha(x)\right)$$

parameters chosen to solve

$$\lambda_0 : Z = \exp(1 - \lambda_0) = Z(\{\lambda_{\alpha \neq 0}\}) = \sum_{x \in A} \exp\left(\sum_{\alpha \neq 0} \lambda_\alpha f_\alpha(x)\right)$$

$$\lambda_{\alpha \neq 0} : \bar{f}_\alpha = \frac{\partial}{\partial \lambda_\alpha} \log Z(\{\lambda_{\alpha \neq 0}\}) = \sum_{x \in A} p^*(x) f_\alpha(x)$$

Maximum Entropy principle (1950s)

The best estimate of a set of probabilities {$p(x)$; $x \in A$}, compatible with the available information is given by an assignment of probabilities wich maximizes the entropy, that is, our ignorance about X, subject to the contraints coming from the available information.

$$H[p] = -k \sum_{x \in A} p(x) \log p(x)$$

$$\langle f_\alpha(x) \rangle = \sum_{x \in A} p(x) f_\alpha(x) = \bar{f}_\alpha, \quad f_\alpha \in \mathcal{M}$$

$$\langle f_0(x) \rangle = \sum_{x \in A} p(x) = 1$$

Denote with p* the best estimate compatible with the available information

$$p^* = \underset{\mathrm{p}}{\operatorname{argmax}} \{H[p]\} \quad \text{subject to } \langle f_\alpha(x) \rangle = \bar{f}_\alpha, \quad f_\alpha \in \mathcal{M}$$

MaxEnt approach gives rise to an exponential family of distribution
The Boltzmann-Gibbs distribution (eq. stat. mech.) can be recovered by the constraint of average energy

$$p^* = \exp\left(\lambda_0 - 1 + \sum_{\alpha \neq 0} \lambda_\alpha f_\alpha(x)\right) = \frac{1}{Z} \exp\left(\sum_{\alpha \neq 0} \lambda_\alpha f_\alpha(x)\right)$$

parameters chosen to solve

$$\lambda_0 : Z = \exp(1 - \lambda_0) = Z(\{\lambda_{\alpha \neq 0}\}) = \sum_{x \in A} \exp\left(\sum_{\alpha \neq 0} \lambda_\alpha f_\alpha(x)\right)$$

$$\lambda_{\alpha \neq 0} : \bar{f}_\alpha = \frac{\partial}{\partial \lambda_\alpha} \log Z(\{\lambda_{\alpha \neq 0}\}) = \sum_{x \in A} p^*(x) f_\alpha(x)$$

Maximum Likelihood (1920s)

ML estimation of a pdf on the basis of finitely many independent observations of a random variable *X*.

$D = \{x_i;\ i=1, \ldots, N\}$ set of available data iid

These data are supposedly sampled from *p*

ML attempts to estimate the unknown *p* by a parametrized family of pdf

$\{p_{\boldsymbol{\lambda}};\ \boldsymbol{\lambda} \in \Lambda\}$, $\Lambda \subseteq \mathbb{R}^n$

Likelihood of the data *D* for a given $\boldsymbol{\lambda}$

$$p(D|\boldsymbol{\lambda}) = \prod_{i=1}^{N} p_{\boldsymbol{\lambda}}(x_i) = \exp\left[\sum_{i=1}^{N} \log p_{\boldsymbol{\lambda}}(x_i)\right] \equiv \exp[\mathcal{L}(\boldsymbol{\lambda}|D)]$$

Best approximation whithin the family: $\tilde{p} = \max_{\boldsymbol{\lambda}} p(D|\boldsymbol{\lambda})$

Log-Likelihood of the data *D* for a given $\boldsymbol{\lambda}$

$$L(\boldsymbol{\lambda}|D) = \frac{1}{N}\mathcal{L}(\boldsymbol{\lambda}|D) = \frac{1}{N}\sum_{i=1}^{N} \log p_{\boldsymbol{\lambda}}(x_i)$$

Add a constant (i.e., the empirical entropy) independent of $\boldsymbol{\lambda}$

$$L(\boldsymbol{\lambda}|D) = -\frac{1}{N}\sum_{i=1}^{N} \log\left[\frac{p(x_i)}{p_{\boldsymbol{\lambda}}(x_i)}\right] - H_N[p], \quad H_N[p] = -\frac{1}{N}\sum_{i=1}^{N} \log p(x_i)$$

$$L(\boldsymbol{\lambda}|D) = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\frac{p(x_i)}{p_{\boldsymbol{\lambda}}(x_i)}\right] - H_N[p], \quad H_N[p] = -\frac{1}{N}\sum_{i=1}^{N}\log p(x_i)$$

$D_N(p||p_{\boldsymbol{\lambda}})$

$$\tilde{p} = \max_{\boldsymbol{\lambda}} p(D|\boldsymbol{\lambda}) \rightarrow \max_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}|D) \rightarrow \min_{\boldsymbol{\lambda}} D_N(p||p_{\boldsymbol{\lambda}})$$

Kullback-Leibler distance $D_{KL}(p||q)$ between two densities p and q

$$D_{KL}(p||q) = \sum_{x\in A} p(x)\log_2\left[\frac{p(x)}{q(x)}\right]$$ aka “relative entropy” or “cross entropy”

Remarks

Not a true distance as $D_{KL}(p||q) \neq D_{KL}(q||p)$

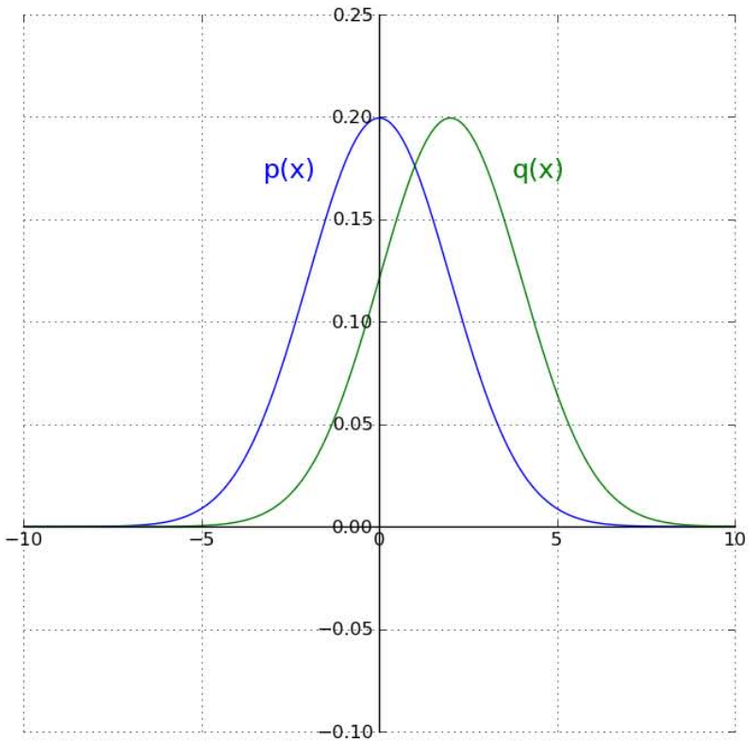
$D_{KL}(p||q) \geq 0$ for any pair of distributions p, q

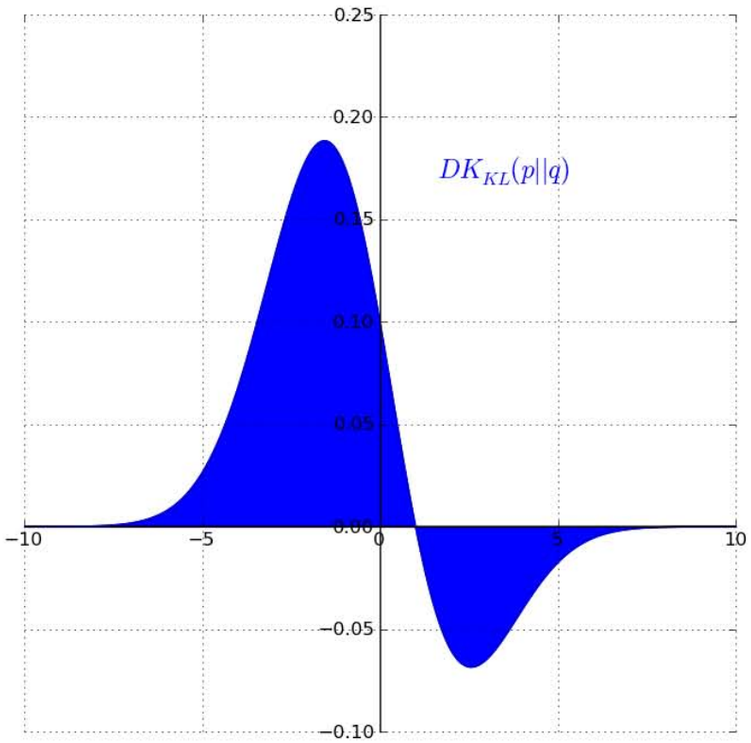
$D_{KL}(p||q) = 0$ iff $p(x)=q(x)$, $\forall x \in A$

$$D_N(p||p_{\boldsymbol{\lambda}}) = \frac{1}{N}\sum_{i=1}^{N} \log\left[\frac{p(x_i)}{p_{\boldsymbol{\lambda}}(x_i)}\right]$$

Up to a factor log2, $D_N(p||p_{\boldsymbol{\lambda}})$ is nothing but an empirical approximation of KL distance $D_{KL}(p||p_{\boldsymbol{\lambda}})$

$$D_N(p||p_{\boldsymbol{\lambda}}) \to \int \mathrm{d}x\ p(x) \log\left[\frac{p(x)}{p_{\boldsymbol{\lambda}}(x)}\right] = \log 2\ D_{KL}(p||p_{\boldsymbol{\lambda}}), \quad \text{as } N = |D| \to \infty$$





ML estimation attempts to approximate the unknown probability density function p as closely as possible by a member of the family $p_{\boldsymbol{\lambda}}$, using the empirical approximation $D_N(p||p_{\boldsymbol{\lambda}})$ to the KL distance as a distance measure

Example of algorithms: <u>iterative gradient descent</u>

$$D_N(p||p_{\boldsymbol{\lambda}}) \to \int \mathrm{d}x\ p(x) \log\left[\frac{p(x)}{p_{\boldsymbol{\lambda}}(x)}\right] = \log 2\ D_{KL}(p||p_{\boldsymbol{\lambda}}), \quad \text{as } N = |D| \to \infty \quad (*)$$

An intuitive algorithm: iterative grad. desc. improvement of parameters

$$\boldsymbol{\lambda}(t+\epsilon) = \boldsymbol{\lambda}(t) - \epsilon \nabla_{\boldsymbol{\lambda}} D_N(p||p_{\boldsymbol{\lambda}})$$

For sufficiently small learning rate ε this process is guaranteed to reduce $D_N(p||p_{\boldsymbol{\lambda}})$.
Indeed, taking the limit ε → 0 we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\lambda} = -\nabla_{\boldsymbol{\lambda}} D_N(p||p_{\boldsymbol{\lambda}})$$

$$\Rightarrow \frac{\mathrm{d}}{\mathrm{d}t} D_N(p||p_{\boldsymbol{\lambda}}) = \nabla_{\boldsymbol{\lambda}} D_N(p||p_{\boldsymbol{\lambda}}) \cdot \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\lambda} = -[\nabla_{\boldsymbol{\lambda}} D_N(p||p_{\boldsymbol{\lambda}})]^2 \leq 0$$

This process is of the batch learning type, as it uses the complete data set *D* for each iterative improvement. Asymptotically, the algorithm is expected to converge, as (*) states that $D_N(p||p_{\boldsymbol{\lambda}})$ converges to a multiple of the KL distance, which is bounded from below.

However…
… for finite *N*, due to the fact that the convergence (*) is defined in a probabilistic sense, boundedness of $D_N(p||p_{\boldsymbol{\lambda}})$ and hence also convergence of the algorithm is not guaranteed.

Application: inverse problem for the CW

recall the CW Hamiltonian $$H_N(\boldsymbol{\sigma}; J, h) = -\frac{1}{2N}\sum_{i,j=1}^{N} J\sigma_i\sigma_j - \sum_{i=1}^{N} h\sigma_i$$

M independent spin configuration $\{\boldsymbol{\sigma}^i\}_{i=1,...,M}$ for a system *J*, *h* ($\beta = 1$)

$$L(J,h) = P_N(\sigma^1, ..., \sigma^M | J, h) = \prod_{i=1}^{M} P_N(\sigma^i | J, h)$$

$$L(J,h) = \prod_{i=1}^{M} \frac{e^{-H_N(\sigma^i|J,h)}}{\sum_\sigma e^{-H_N(\sigma^i|J,h)}}$$

$$\log L(J,h) = \sum_{i=1}^{M} \left(-H_N(\sigma^i|J,h) - \log \sum_\sigma e^{-H_N(\sigma^i|J,h)} \right)$$

Extensive entropy over *M* replicas

$$\frac{\partial \log L}{\partial J} = \frac{N}{2}\sum_{i=1}^{M}(m^2(\sigma^i) - \langle m^2 \rangle)$$

$$\frac{\partial \log L}{\partial h} = N\sum_{i=1}^{M}(m(\sigma^i) - \langle m \rangle)$$

$$J_{\text{exp}} = \frac{1}{1 - m_{\text{exp}}^2} - \frac{1}{\chi_{\text{exp}}}$$

$$h_{\text{exp}} = \text{atanh}(m_{\text{exp}}) - J_{\text{exp}} m_{\text{exp}}$$

Ideas and concepts underlying machine learning

A machine learning algorithm is an algorithm that is able to **learn** from data

A computer program is said to learn from **experience** E with respect to some class of **tasks** T and **performance measure** P, if its performance at tasks T, as measured by P, improves with experience E

Ideas and concepts underlying machine learning

A machine learning algorithm is an algorithm that is able to **learn** from data

A computer program is said to learn from **experience** E with respect to some class of **tasks** T and **performance measure** P, if its performance at tasks T, as measured by P, improves with experience E

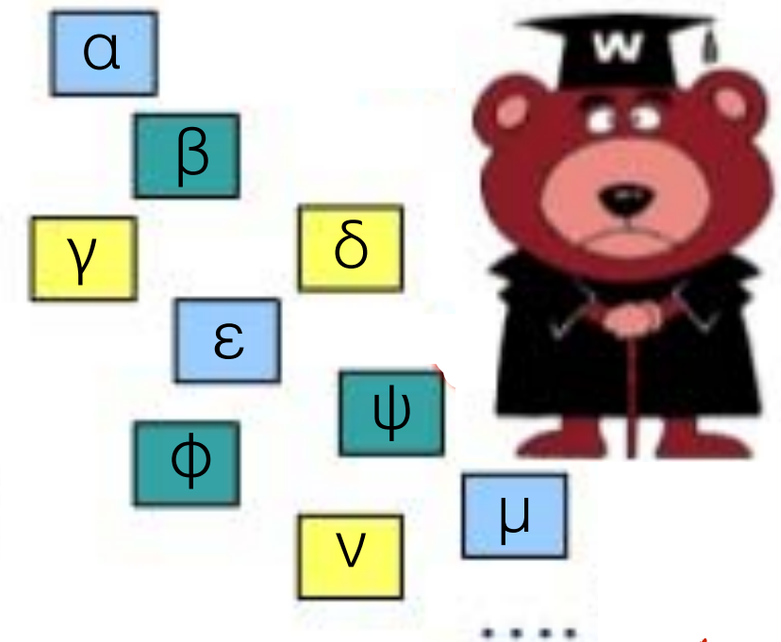
<u>The performance measure, P</u>

Design a quantitative measure of the performance to evaluate the abilities of a machine learning algorithm.
P is specific to the task being carried out by the system.

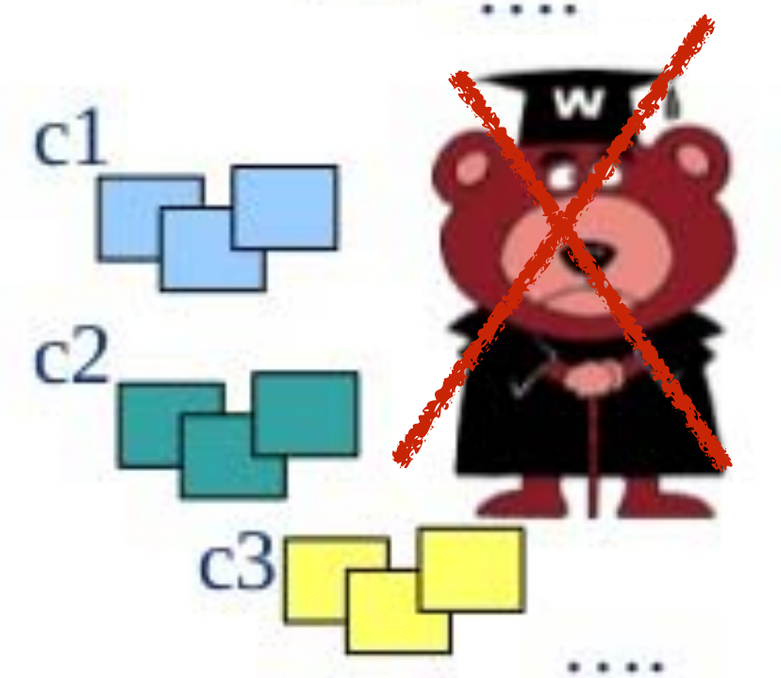
E.g. accuracy of the model, namely the proportion of examples for which the model produces the correct output.

The experience, E

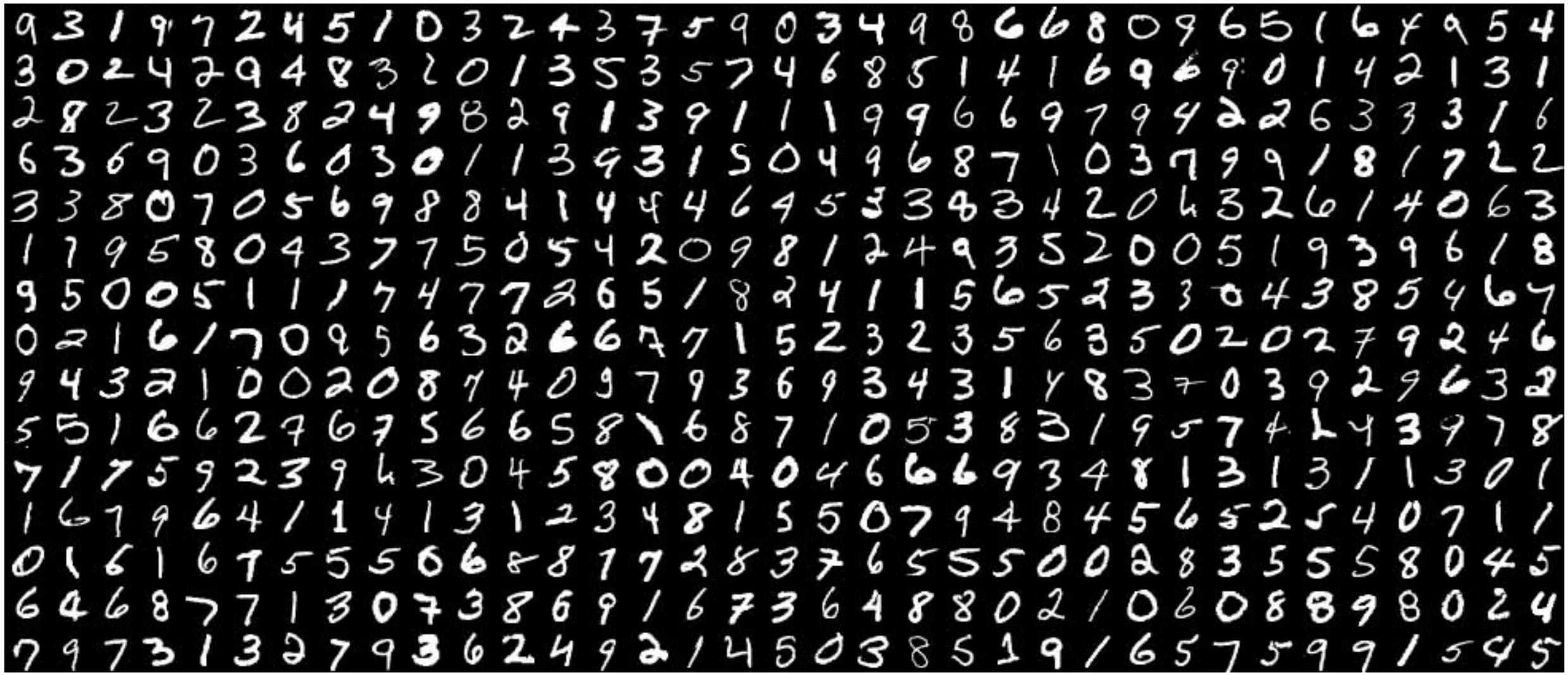
Supervised learning algorithms experience a dataset containing features, and each example is also associated with a label.



Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset. E.g., clustering (PCA): divide the dataset into clusters of similar examples



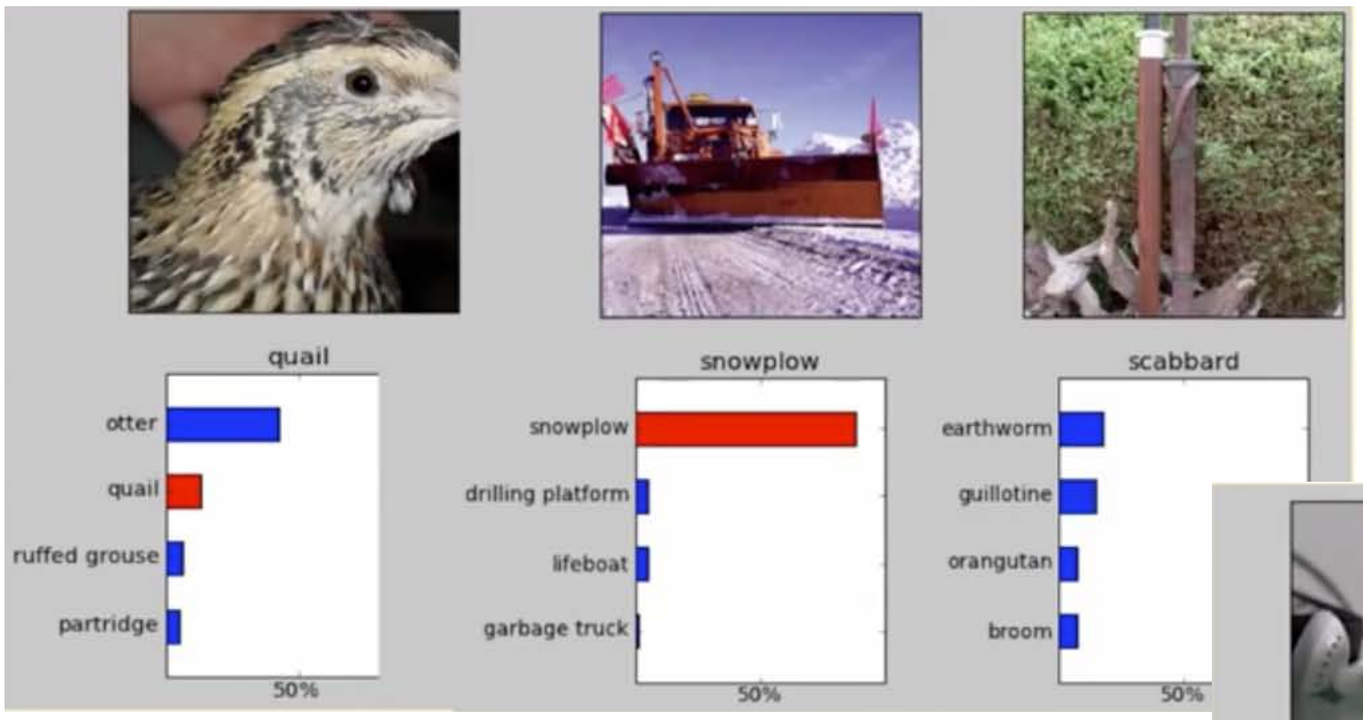
Supervised learning ➛ observe several examples of a random vector x and an associated value or vector y, and learn to predict y from x, usually by estimating p(ylx).
Unsupervised learning ➛ observe several examples of a random vector x, and learn the probability p(x), or some interesting properties of that distribution

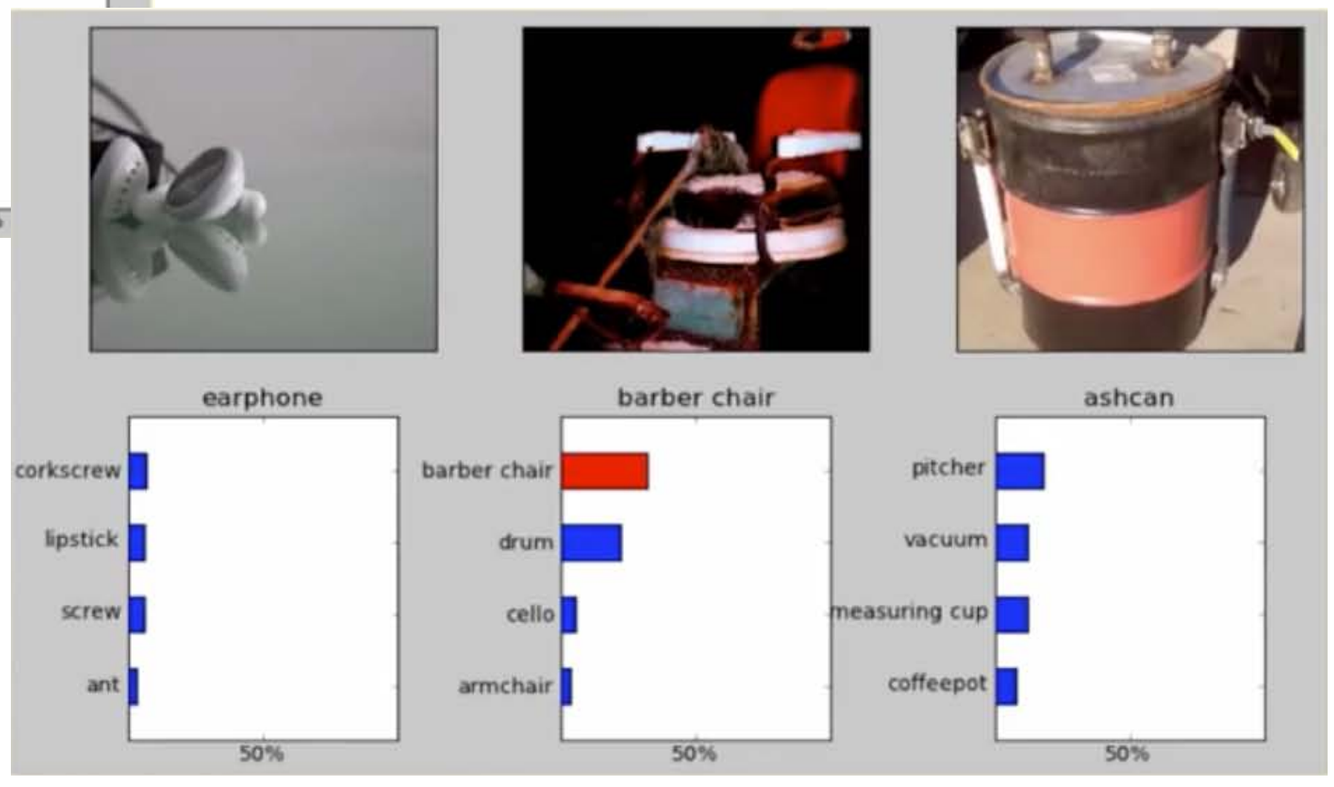


Example inputs: the MNIST database.
NIST = National Institute of Standards and Technology (the agency that originally collected this data)
M = modified (data has been reprocessed for easier use with machine learning algorithm)

The MINIST database consists of scans of handwritten digits and associated labels describing which digit 0-9 is contained in each image. This classification problem is one of the simplest and most widely used in machine learning.
It is the drosophile of machine learning → it allows machine learning researchers to study their algorithms in controlled laboratory conditions

The ImageNet task
1000 different object classes
Best system in 2010 competition got 47% error for its first choice and 25% error for its top 5 choices
In 2012 error rate of 15.4% in top 5





Ilya Sutskever (2011) trained a special type of recurrent neural net to predict the next character in a sequence
After training for a long time on a string of half a billion characters from English Wikipedia, he got to generate new text

It generates by predicting the probability distribution for the next character and then sampling a character from this distribution

> In 1974 Northern Denver had been overshadowed by CNL, and several Irish intelligence agencies in the Mediterranean region. However, on the Victoria, Kings Hebrew stated that Charles decided to escape during an alliance. The mansion house was completed in 1882, the second in its bridge are omitted, while closing is the proton reticulum composed below it aims, such that it is the blurring of appearing on any well-paid type of box printer.

BLEU (bilingual evaluation understudy): algorithm for evaluating the quality of text which has been machine-translated from one natural language to another.

Word error rate (WER): common metric of the performance of a speech recognition or machine translation system

Incidentally…

Natural Language Process (NLP) is a field concerned with the interactions between computers and human (natural) languages.

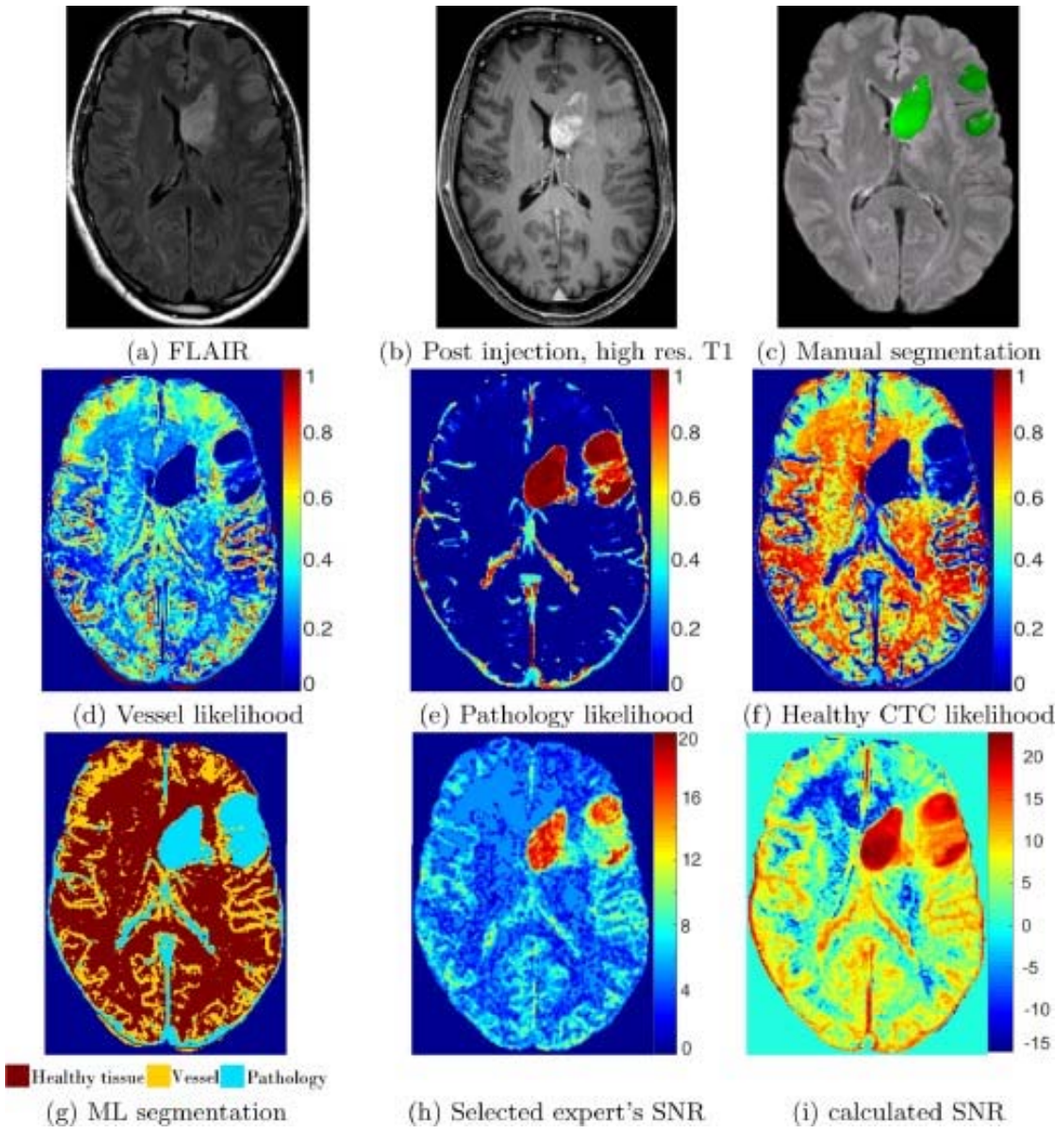
Huge investments (e.g., from Google) involve speech recognition, natural language understanding, natural language generation, translations etc.

Language is particularly challenging due to its intrinsic non-Gaussian nature
Zipf's law is universal

This states that given a large sample of words used, the frequency of any word is inversely proportional to its rank in the frequency table.
Ex. In one sample of words in the English language, the most frequently occurring word, "the", accounts for nearly 7% of all the words, the second-place word "of" accounts for 3.6%, followed by "and" which accounts for 2.7%. Only 135 vocabulary items are needed to account for half the sample of words.

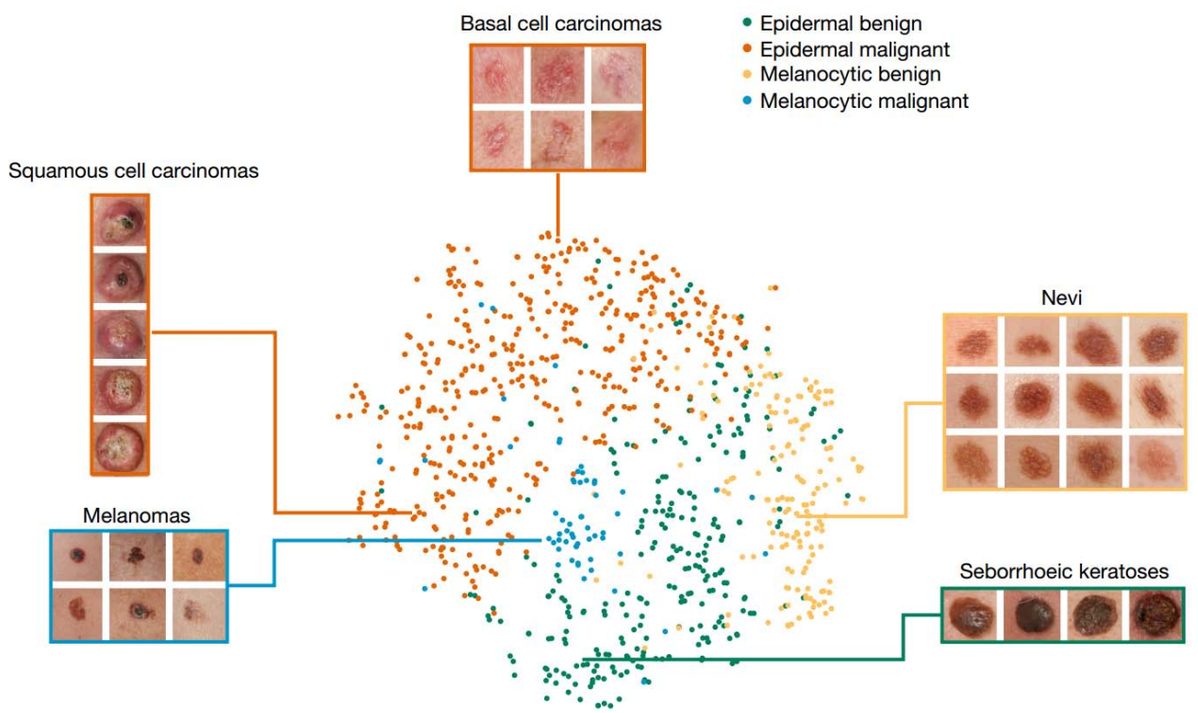
BabelNet

Among applications also denoising and diagnostic medicine

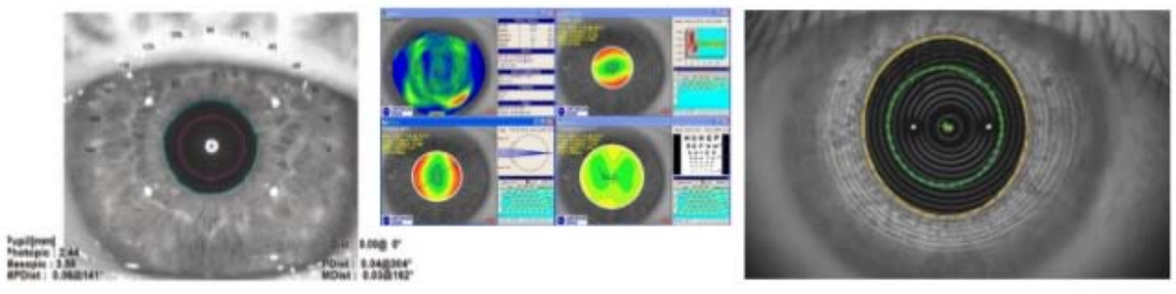


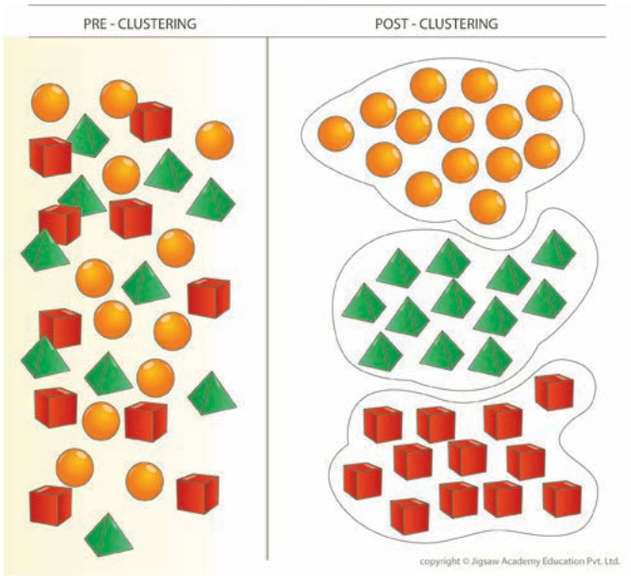
Using a dataset of brain images of people with full-blown Alzheimer's, those with mild cognitive impairment (MCI), and those with MCI that subsequently developed Alzheimer's, Hongyoon and Kyong were able to train a NN that could predict the disease with an accuracy of 84%. That's much better than what a trained diagnostician could manage to do.

Early diagnosis is key to reducing costs and improving outcomes for most diseases. For example, with skin cancer, the five-year survival rate is 97% if it's detected in its earlier stages, but drops to 14% if it's detected in its latest stages.



Pupillometry for clinical diagnosis: pre-processing with outlier detection, denoising and analysis using machine learning





Supervised

NO.	SIZE	COLOR	SHAPE	FRUIT NAME
1	Big	Red	Rounded shape with a depression at the top	Apple
2	Small	Red	Heart-shaped to nearly globular	Cherry
3	Big	Green	Long curving cylinder	Banana
4	Small	Green	Round to oval,Bunch shape Cylindrical	Grape

Learn to associate features to a label
Input: image of a fruit
Output: the fruit name

Unsupervised

- RED COLOR AND BIG SIZE (apple)
- RED COLOR AND SMALL SIZE (cherry fruits)
- GREEN COLOR AND BIG SIZE (bananas)
- GREEN COLOR AND SMALL SIZE (grapes)

Learn to group items according to features
Input: image of a fruit
Output: the related group

Chapter VI
Supervised learning

The perceptron

For a McCulloch-Pitts neuron $S(\boldsymbol{x}) = \theta(\boldsymbol{J}\boldsymbol{x}\text{-}U)$ learning means the adaptation of the set of connections $\{J_i\}$ and the threshold U, in order to improve the accuracy in the execution of a given task M: $\{0,1\}^N \rightarrow \{0,1\}$.

M not known explicitly

Available examples provided by some “teacher” of “questions" (input vectors $\boldsymbol{x}\in\{0,1\}^N$) with corresponding “answers” (the associated output values $M(\boldsymbol{x})$)

step 1: Draw at random a question $x \in \Omega$

step 2: Check whether teacher and student agree on the answer:

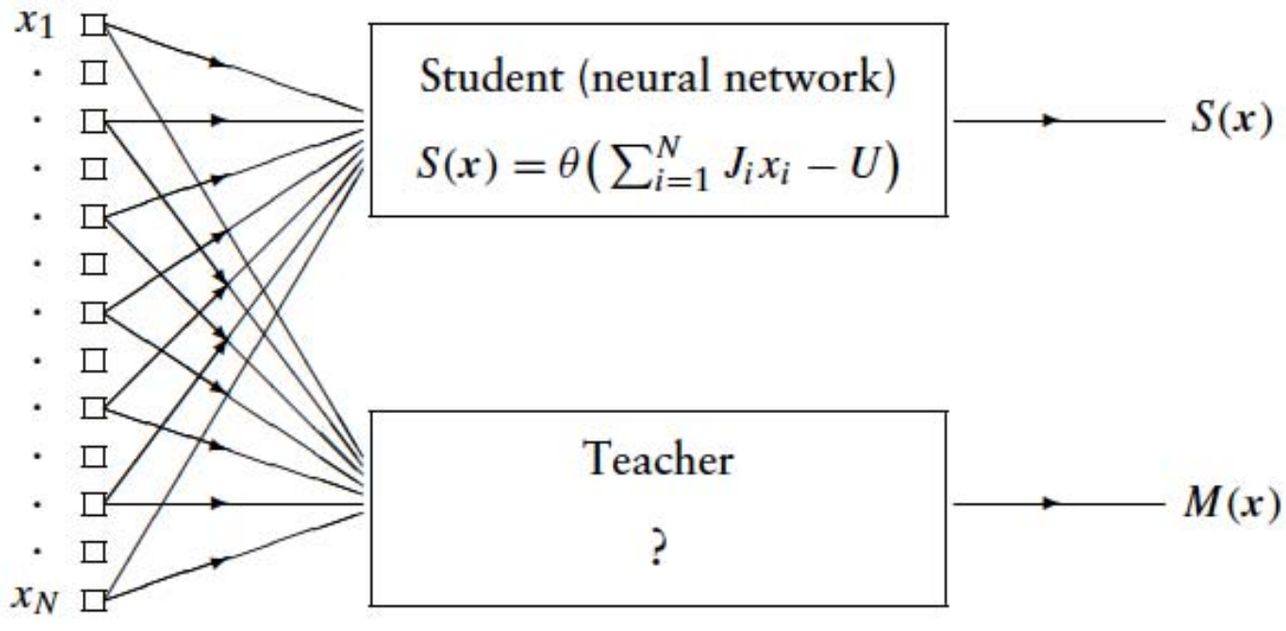
“Online” training session

$M(x) = S(x)$: do nothing, return to 1

$M(x) \neq S(x)$: modify parameters, then return to 1

modify parameters:

$$\begin{aligned} \boldsymbol{J} &\rightarrow \boldsymbol{J} + \Delta\boldsymbol{J}(\boldsymbol{J}, U; x, M(x)) \\ U &\rightarrow U + \Delta U(\boldsymbol{J}, U; x, M(x)) \end{aligned}$$



Goal: $M(\boldsymbol{x}) = S(\boldsymbol{x}) = \theta(\boldsymbol{J}\boldsymbol{x}\text{-}U)$

(possible only as M is itself linearly separable)

perceptron: MP neuron, learning in an online fashion, according to the following rule for updating its parameters

$$\text{perceptron learning rule:} \quad \begin{cases} M(x) = 0, \;\; S(x) = 1: & \Delta \boldsymbol{J} = -\boldsymbol{x}, \;\; \Delta U = 1 \\ M(x) = 1, \;\; S(x) = 0: & \Delta \boldsymbol{J} = \boldsymbol{x}, \;\; \Delta U = -1 \end{cases}$$

"I've been learning so much from my mistakes that I'm keeping on doing mistakes"

Goal: $M(\boldsymbol{x}) = S(\boldsymbol{x}) = \theta(\boldsymbol{J}\boldsymbol{x}\text{-}U)$

(possible only as M is itself linearly separable)

perceptron: MP neuron, learning in an online fashion, according to the following rule for updating its parameters

$$\text{perceptron learning rule:} \quad \begin{cases} M(x) = 0, \;\; S(x) = 1: & \Delta \boldsymbol{J} = -\boldsymbol{x}, \;\; \Delta U = 1 \\ M(x) = 1, \;\; S(x) = 0: & \Delta \boldsymbol{J} = \boldsymbol{x}, \;\; \Delta U = -1 \end{cases}$$

Works as a binary classifier

Perceptron convergence theorem (Rosenblatt, 1962)

Theorem. If the task M is *linearly separable*, then the above procedure will converge in a finite number of modification steps to a stationary configuration, where $\forall \boldsymbol{x} \in \{0,1\}^N$: $S(\boldsymbol{x}) = M(\boldsymbol{x})$

F. Rosenblatt (1959) *Two theorems on statistical separability in the perceptron*, Proc. of a Symposium on the Mechanization of Thought Processes. London: Her Majesty's Stationary Office

Linear separability

The three elementary logical operations ($\wedge$, $\vee$, $\neg$) can be realized with McCulloch-Pitts neurons

$$S_i(t+\Delta) = \theta\left(\sum_{k=1}^{N} J_{ik} S_k(t) - U_i^*\right)$$

provided we choose appropriate values of the parameters (J_{ik}, U_i^*).
By construction…

x	y	$x \wedge y$	$x+y-3/2$	$\theta[x+y-3/2]$
0	0	0	−3/2	0
0	1	0	−1/2	0
1	0	0	−1/2	0
1	1	1	1/2	1

N=2

x	y	$x \vee y$	$x+y-1/2$	$\theta[x+y-1/2]$
0	0	0	−1/2	0
0	1	1	1/2	1
1	0	1	1/2	1
1	1	1	3/2	1

N=2

x	$\neg x$	$-x+1/2$	$\theta[-x+1/2]$
0	1	1/2	1
1	0	−1/2	0

N=1

x	y	NAND(x,y)	$-x-y+3/2$	$\theta[-x-y+3/2]$
0	0	1	3/2	1
0	1	1	1/2	1
1	0	1	1/2	1
1	1	0	−1/2	0

N=2

Do all operations $\{0,1\}^N \rightarrow \{0,1\}$ can be performed with MP neuron(s)?

N=1 ✓ N>1 ✗

Counterexample for N=2: XOR

x	y	XOR(x, y)
0	0	0
0	1	1
1	0	1
1	1	0

Prop. No set of real numbers $\{J_x, J_y, U\}$ exists such that

$$\theta(J_x x + J_y y - U) = \text{XOR}(x,y) \quad \forall (x,y) \in \{0,1\}^2.$$

(Proof by contraddiction).

For N>2 we can construct a similar operation M by just applying the XOR operation to the first two of the N input variables:

$$M: \{0,1\}^N \rightarrow \{0,1\} \quad M(x_1, \ldots, x_N) = \text{XOR}(x_1, x_2).$$

A geometric picture

The set of all possible operations M: $\{0,1\}^N \rightarrow \{0,1\}$ consists of all possible ways to fill the right column with ones and zeros

x_1	x_2	$\cdots$	$\cdots$	x_{N-1}	x_N	$M(x)$
0	0	$\cdots$	$\cdots$	0	0	$*$
1	0	$\cdots$	$\cdots$	0	0	$*$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	$\cdots$	$\cdots$	1	0	$*$
1	1	$\cdots$	$\cdots$	1	1	$*$

Number of operations M: $\{0,1\}^N \rightarrow \{0,1\} = 2^{2^N}$

The set $\{0,1\}^N$ of possible binary input vectors consists of the corners of the unit hypercube in $\mathbb{R}^N$. A MP neuron, performing the operation

$$S : \{0,1\}^N \rightarrow \{0,1\} \quad S(\mathbf{x}) = \theta\left(\sum_{k=1}^{N} J_k x_k - U\right)$$

can be seen as dividing $\mathbb{R}^N$ into two subsets which are separated by the hyperplane $\sum_{k=1}^{N} J_k x_k = U$.

Operation outcome: which of the two subsets the corner ***x*** of the hypercube is located in.

$N = 1$: $x \in \{0,1\}$

●: $x \in \Omega^+, \ M(x) = 1$

○: $x \in \Omega^-, \ M(x) = 0$

●: outcome 0

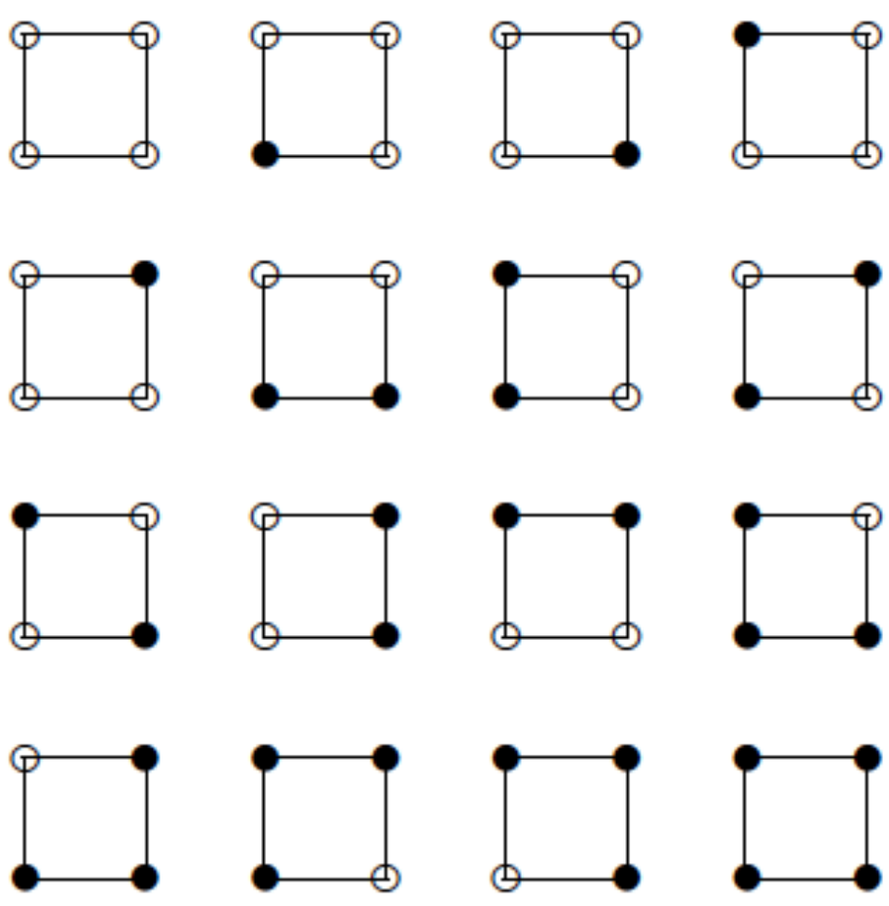
○: outcome 1

left: input 0
right: input 1

$N = 2$: $x \in \{0,1\}^2$

●: $x \in \Omega^+, \ M(x) = 1$

○: $x \in \Omega^-, \ M(x) = 0$

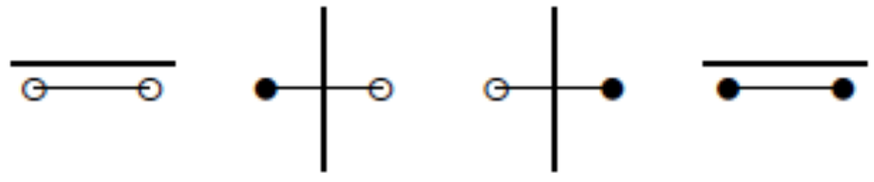


●: outcome 0

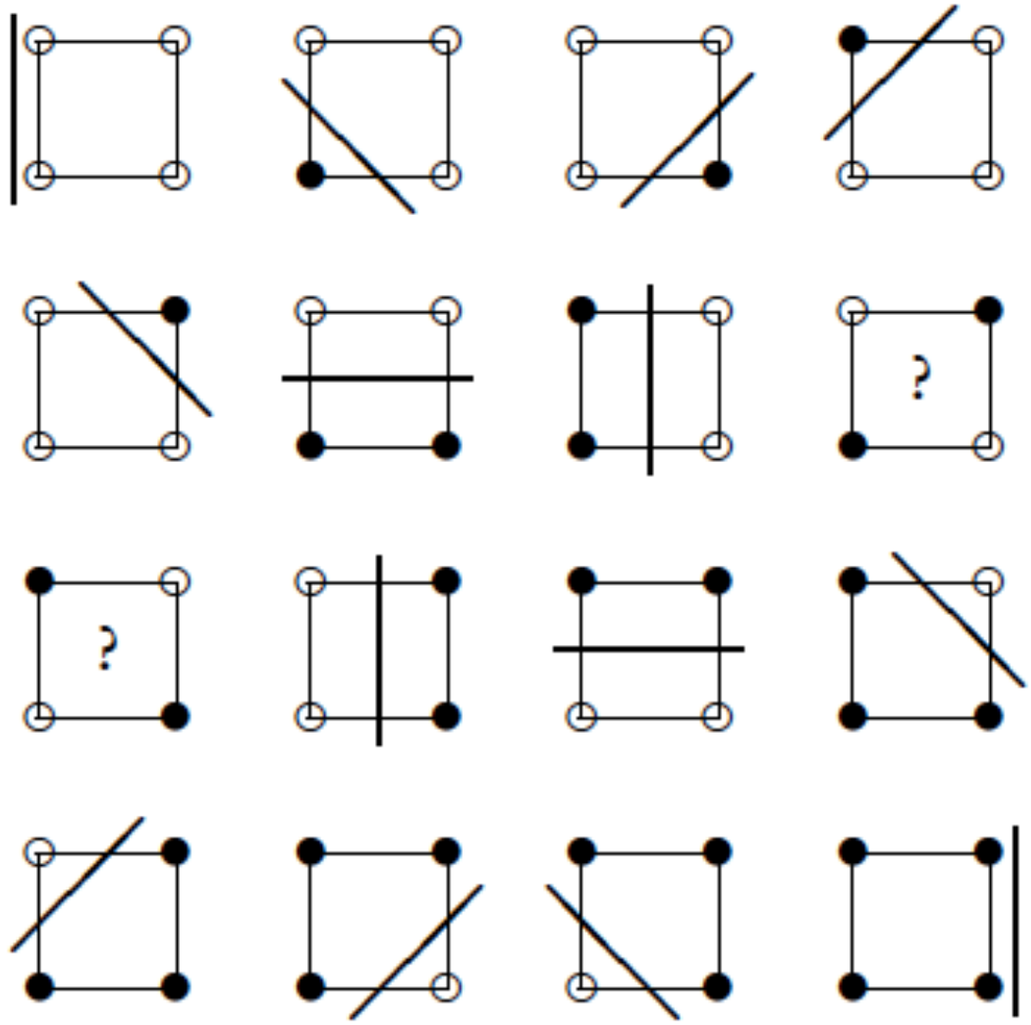
○: outcome 1

up left: input 00
up right: input 01
down left: input 10
down right: input 11

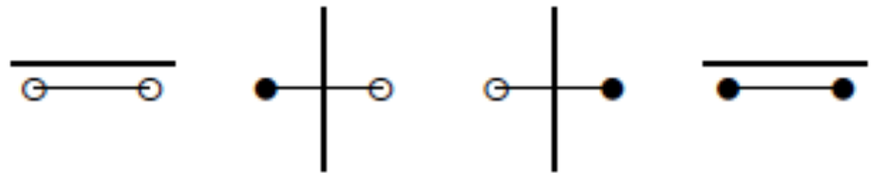
$N = 1$:



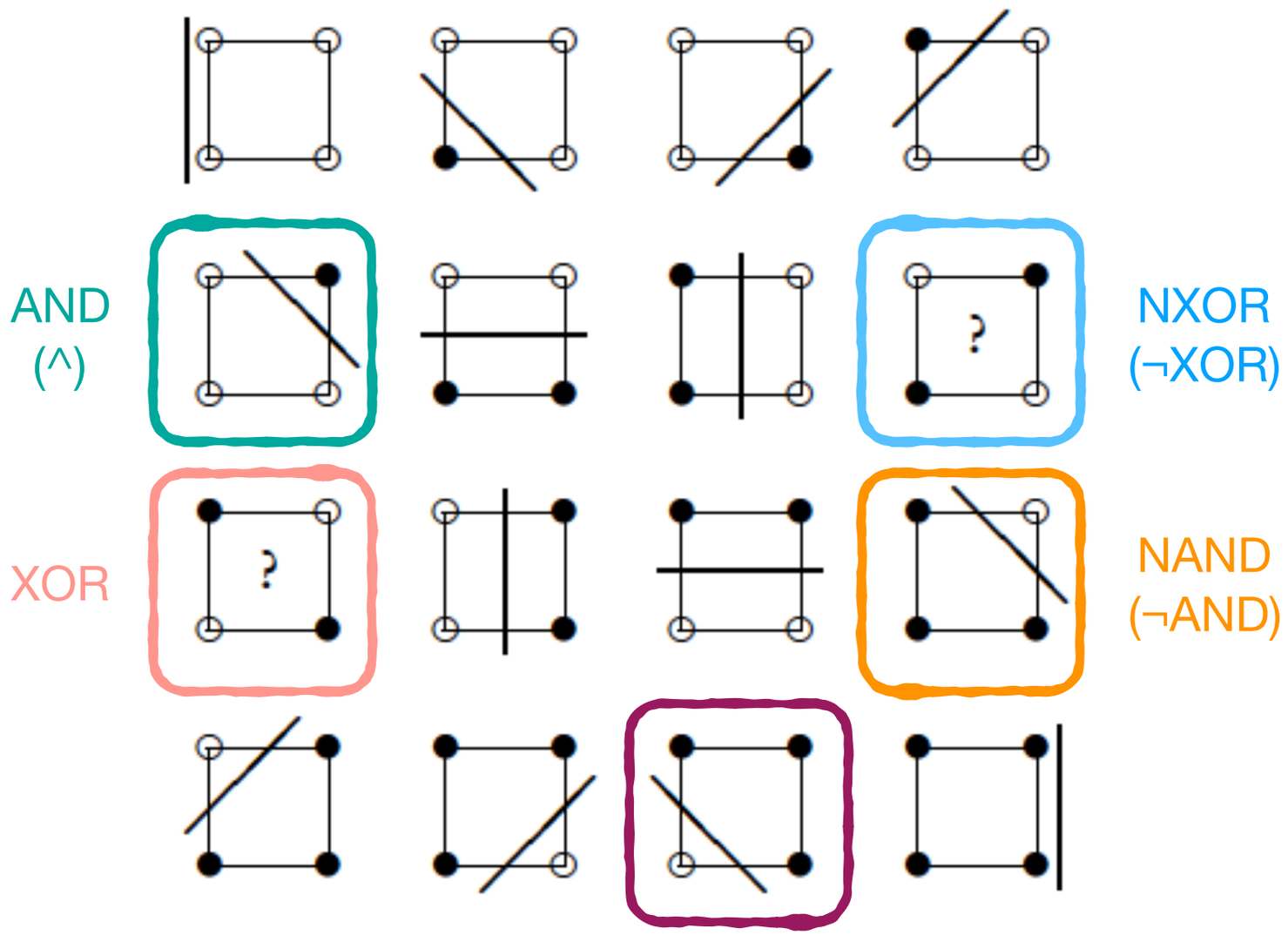
$N = 2$:



$N = 1$:

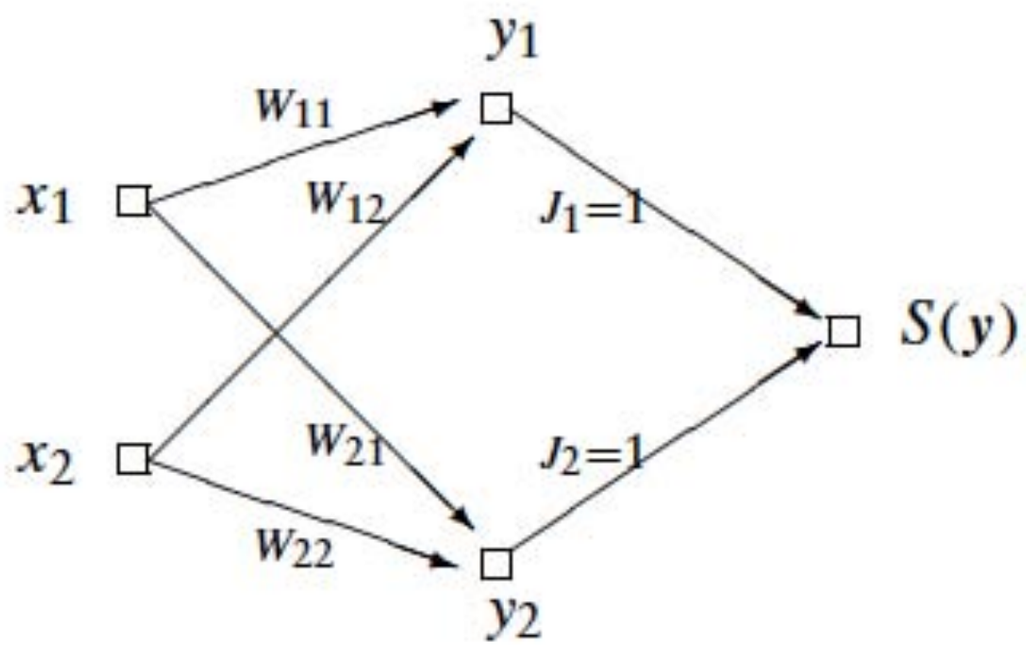


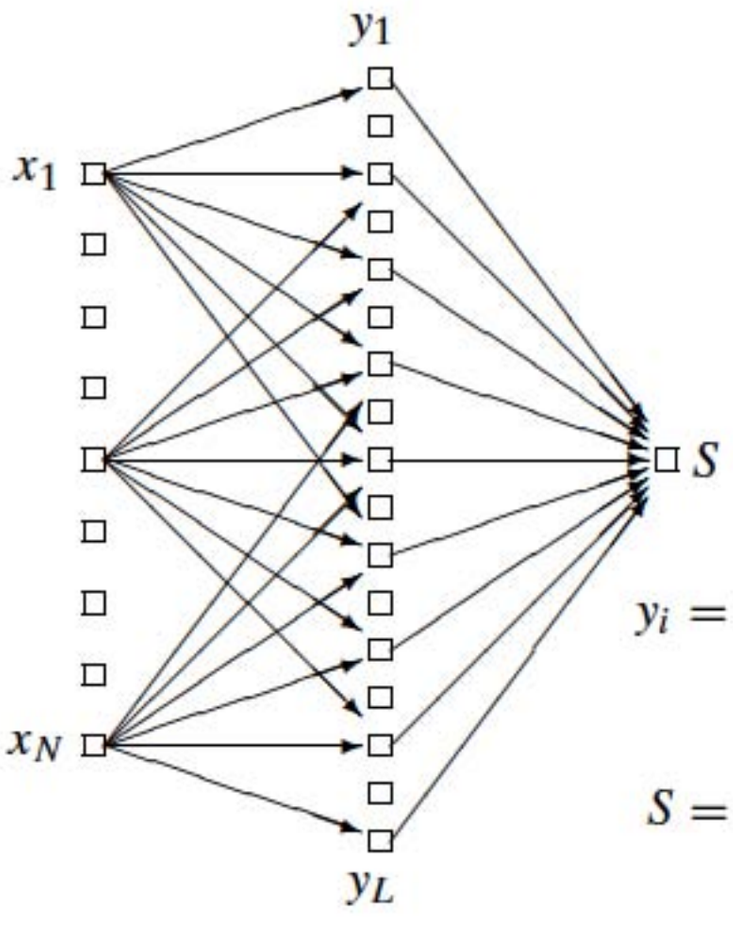
$N = 2$:



Multilayer network

Every mapping M: $\{0,1\}^N \rightarrow \{0,1\}$ can at least be performed by a two-layer feed-forward network of MP neurons, with only one neuron in the second layer.

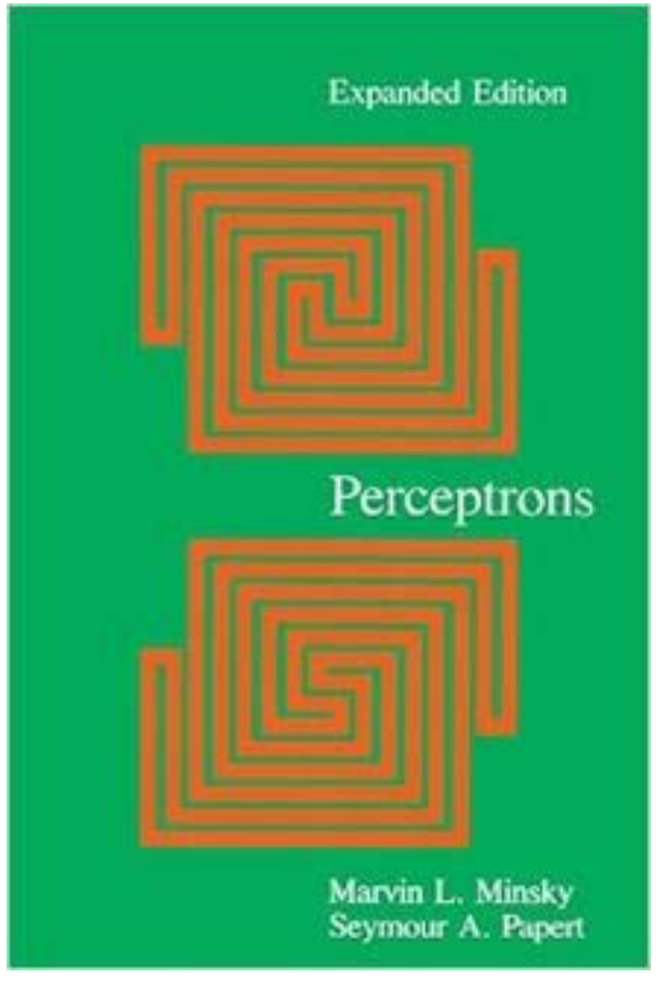




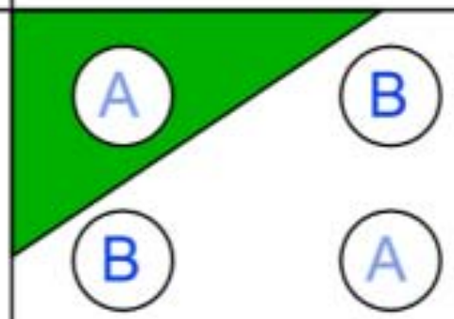
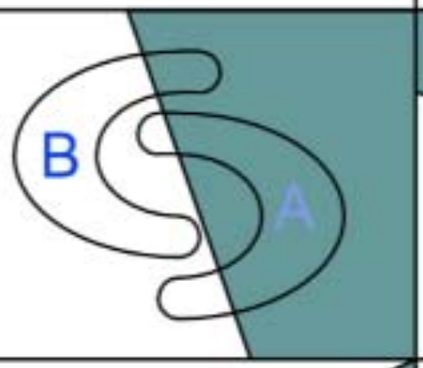
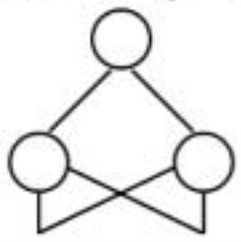
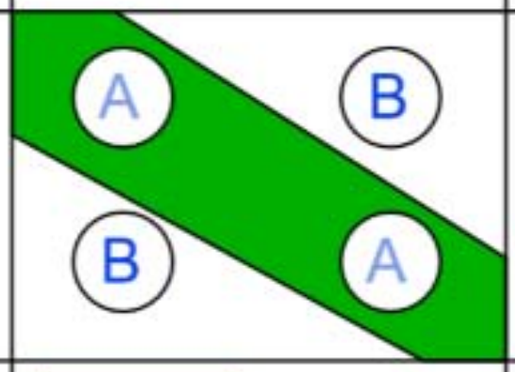
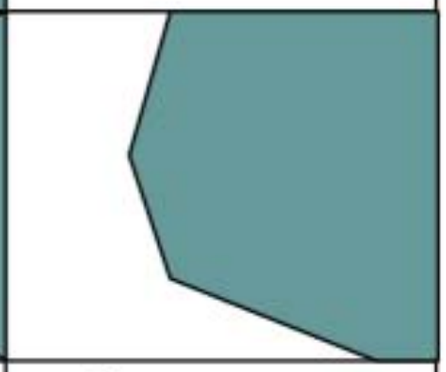
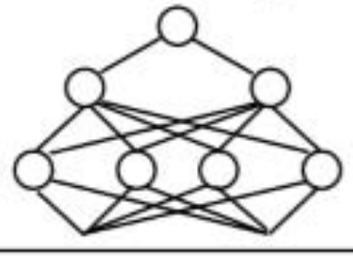
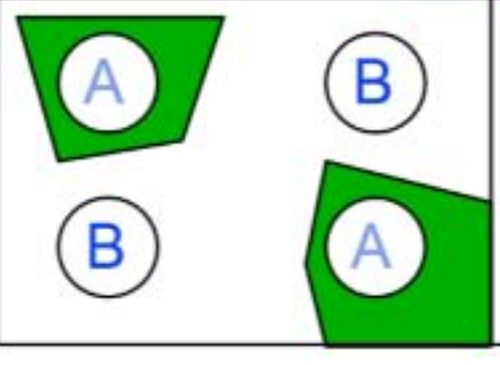
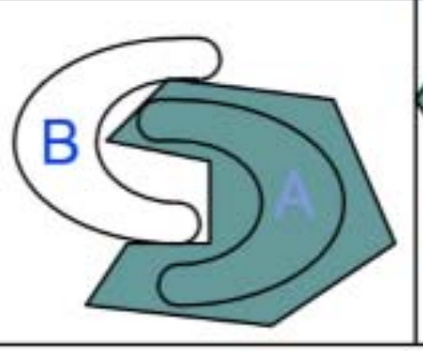
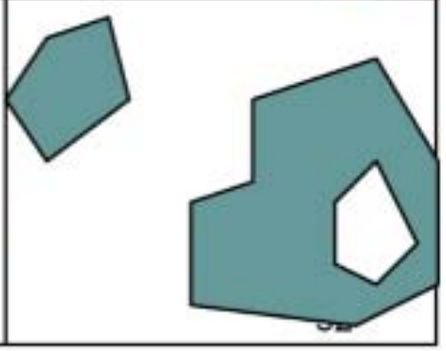
General construction for the arbitrary input dimensions N and for arbitrary operations M: $\{0, 1\}^N \rightarrow \{0, 1\}$.

$$y_i = \theta\left(\sum_{j=1}^{N} W_{ij} x_j - V_i\right)$$

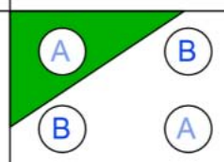
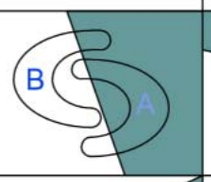
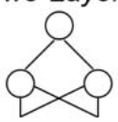
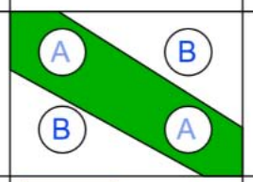
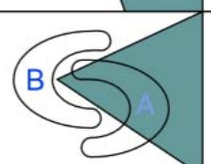
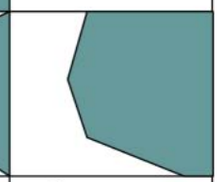
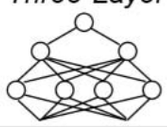
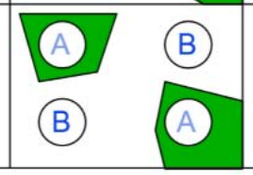
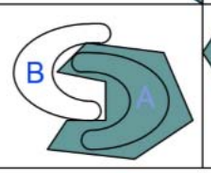
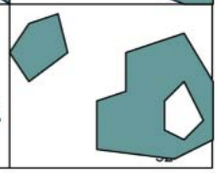
$$S = \theta\left(\sum_{i=1}^{L} J_i y_i - U\right)$$



1969

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

If data are linearly separable Rosenblatt's theorem holds
Multilayer perceptrons are meant to iteratively perform transformation over the variables (and over their transform, over transform of transform…) in such a way that in the last layer one finally has a rearrangement of data for which Rosenblatt's theorem holds.
Namely, hidden layers make data linearly separable.
Multilayer perceptron is at the basis of deep learning.

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

An intuitive sketch:
1. $y= ax+b \rightarrow$ simple perceptron
2. $y= ax^2 + b \rightarrow$ perceptron with one hidden layer; the latter applies the transform $x^2=z$ in such a way that we get $y=az+b$ which can be handled by the simple perceptron and then get back to x
…

Feedforward networks
The perceptron theorem is the only clue to get mathematical control.

For recurrent networks (see in the following Boltzmann machines) we can rely on detailed balance and this ensures that we can reach the Boltzmann-Gibbs distribution which, in turns, allows to build up effective algorithms such as the contrastive divergence.

Binary classification and regression

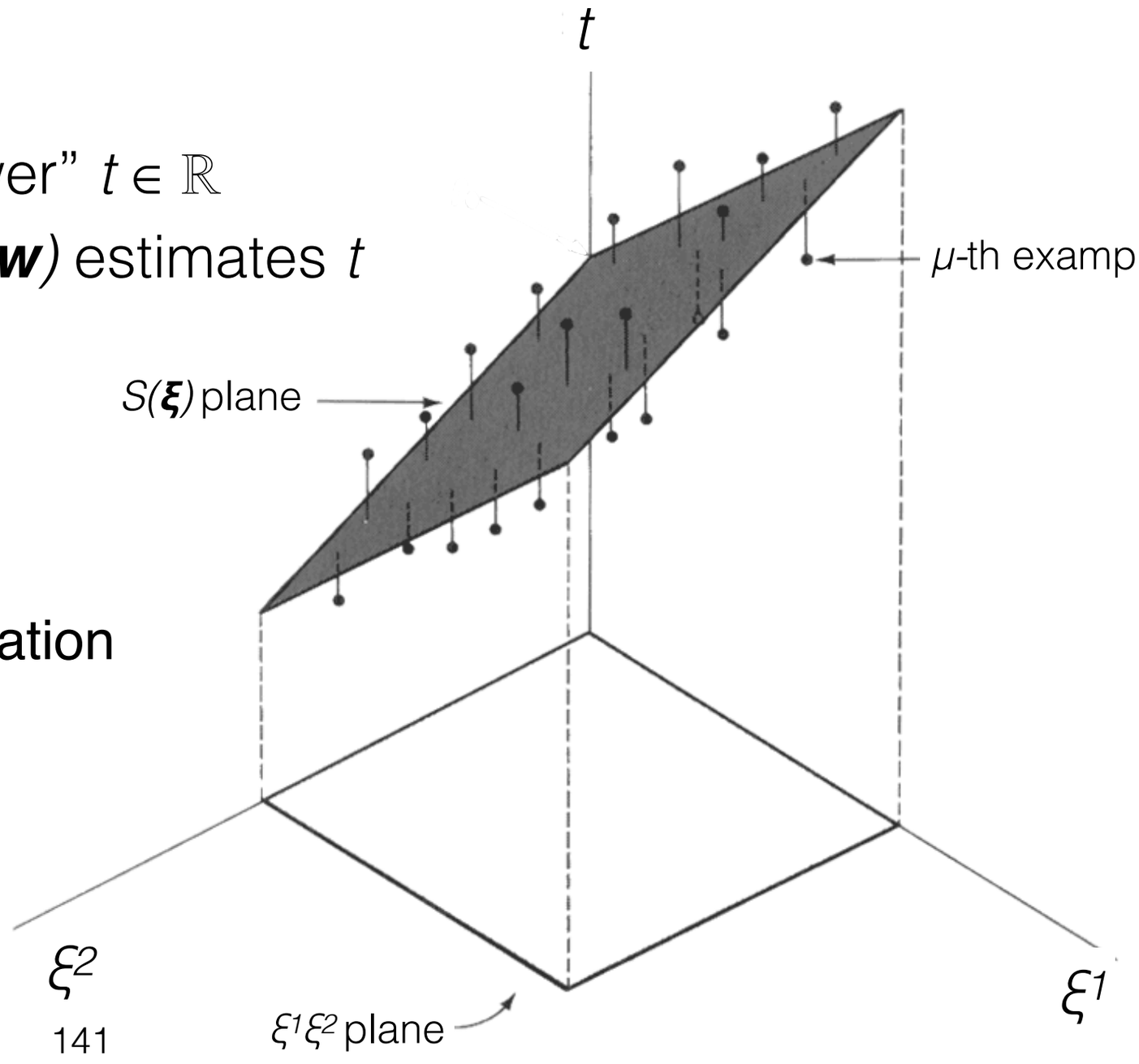
Experience p input-output pairs $(\boldsymbol{\xi}^{\mu}, t_{\mu})_{\mu=1,\ldots,p}$ ➛ supervised

$$\{\boldsymbol{\xi}^1, ..., \boldsymbol{\xi}^p\}, \quad \boldsymbol{\xi}^{\mu} \in \Omega \subseteq \mathbb{R}^N \quad \text{drawn from some } p(\boldsymbol{\xi})$$

$$\{t_1, ..., t_p\}, \quad t_{\mu} \in \mathbb{R}$$

Task "question" $\boldsymbol{\xi} \in \mathbb{R}^N \rightarrow$ "answer" $t \in \mathbb{R}$

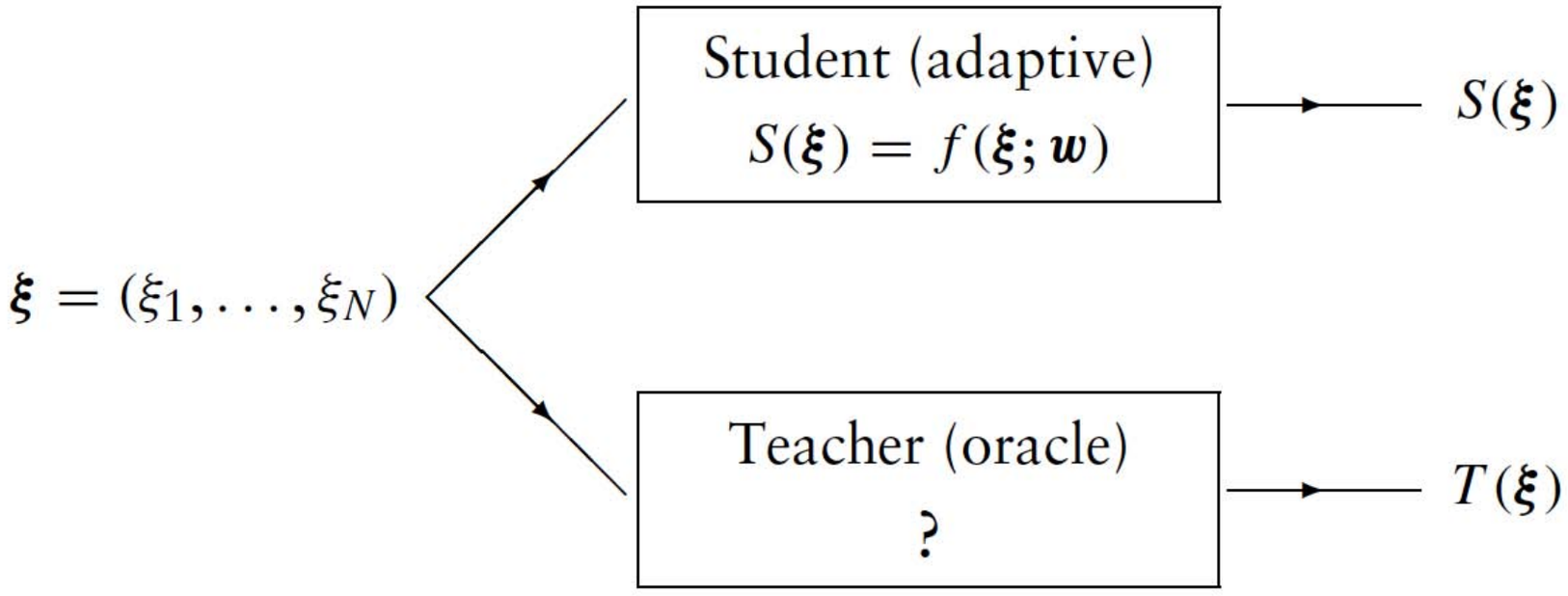
S: $\mathbb{R}^N \rightarrow \mathbb{R}$, where $S(\boldsymbol{\xi}) = f(\boldsymbol{\xi};\boldsymbol{w})$ estimates t



Answer t_{μ} generated by a teacher

Student executes parametrized operation
S: $\mathbb{R}^N \rightarrow \mathbb{R}$, where $S(\boldsymbol{\xi}) = f(\boldsymbol{\xi};\boldsymbol{w})$

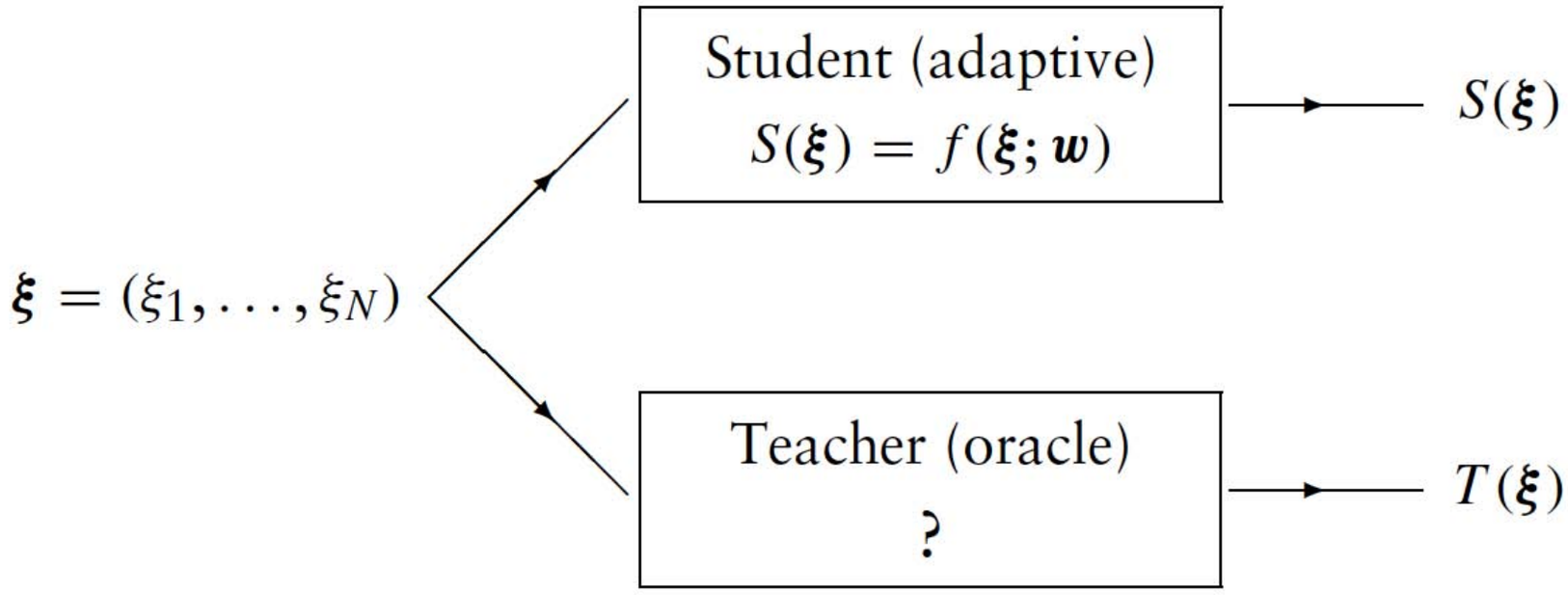
Parameters $\boldsymbol{w}$ determine operation
Learning means adapting $\boldsymbol{w}$



Answers generated by function T: $\mathbb{R}^N \rightarrow \mathbb{R}$

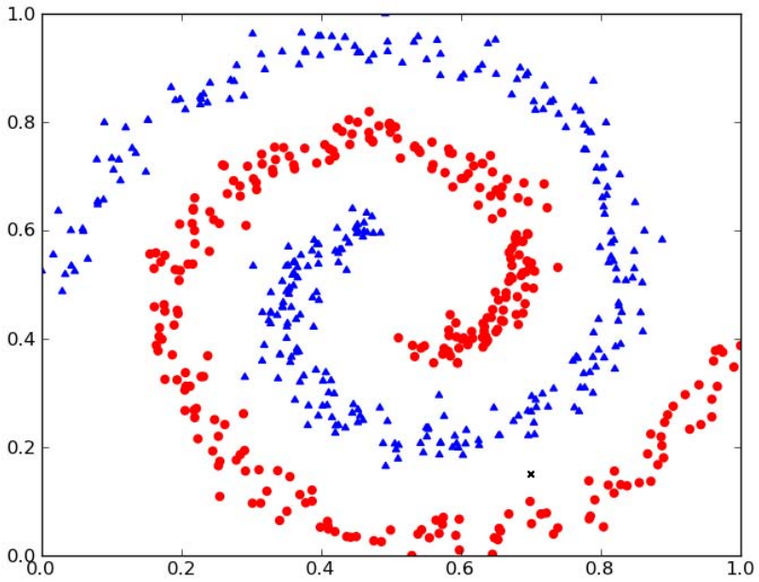
Student has to infer T from the p input-output pairs ($\boldsymbol{\xi}$, t_μ)

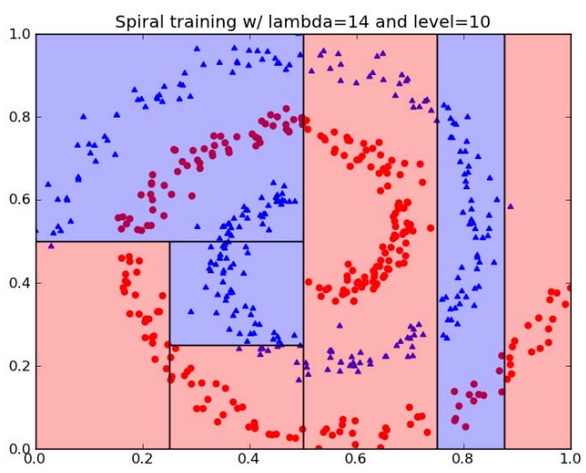
Data are not perfect
Hp: inaccuracy (or noise) is independent for the different output
(the teacher is sloppy rather than using a consistently wrong rule)
$t_\mu = T(\xi^\mu) + z_\mu, \quad z_\mu \in \mathbb{R}$ drawn randomly from $P(z)$
Without loss of generality $\langle z \rangle = 0$ (a non zero average would be equivalent to a structural modification of the rule T, and could be absorbed into its definition)

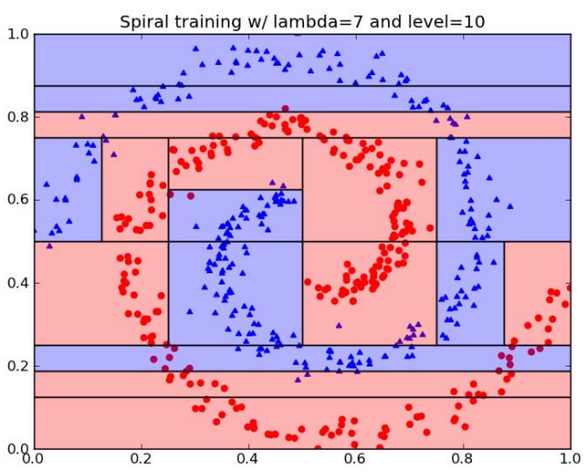


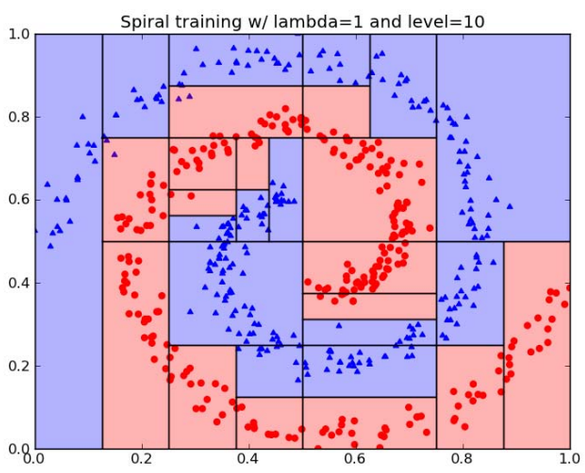
If the outputs are binary, e.g., $t_\mu \in \{0,1\}$ or $t_\mu \in \{-1, 1\}$ → task is binary classification

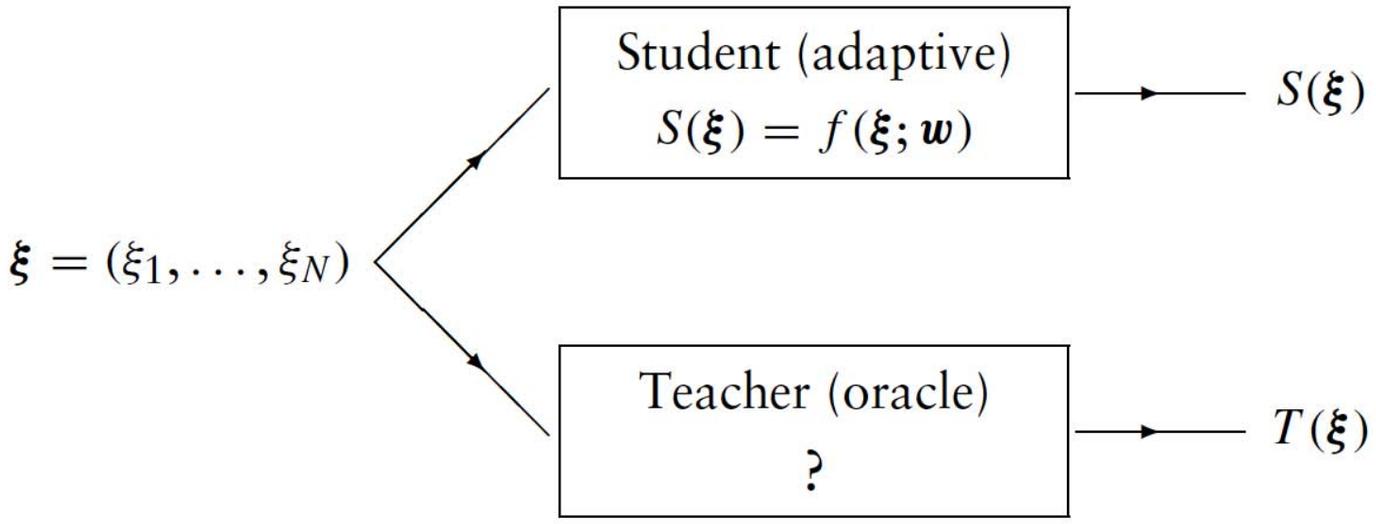
In this case noise is introduced as
Prob[$t_\mu = T(\boldsymbol{\xi}^\mu)$] = 1 - ϵ, $0 \leq \epsilon \leq 1$











Output t_μ binary ➛ binary classification
Output t_μ real ➛ regression

<u>Task</u> "question" $\boldsymbol{\xi} \in \mathbb{R}^N \rightarrow$ "answers" $t \in \mathbb{R}^N$

S: $\mathbb{R}^N \rightarrow \mathbb{R}$, where $S(\boldsymbol{\xi}) = f(\boldsymbol{\xi}; \boldsymbol{w})$ estimates t

<u>Experience</u> p input-output pairs $(\boldsymbol{\xi}, t_\mu)$ ➛ training set

<u>Performance measure</u> $\mathcal{E}(x, y) = |x - y|^\gamma, \gamma > 0$

$$E_t = \frac{1}{p} \sum_{\mu=1}^{p} \mathcal{E}(t_\mu, S(\boldsymbol{\xi}^\mu))$$

training error

$$E_g = \int_\Omega \mathrm{d}\boldsymbol{\xi}\, p(\boldsymbol{\xi}) \mathcal{E}(T(\boldsymbol{\xi}), S(\boldsymbol{\xi}))$$

generalization error

training error

$$E_t = \frac{1}{p}\sum_{\mu=1}^{p} \mathcal{E}(t_\mu, S(\boldsymbol{\xi}^\mu))$$

$$\downarrow \qquad S(\boldsymbol{\xi}) = f(\boldsymbol{\xi}; \boldsymbol{w})$$

$$E_t(\boldsymbol{w}) = \frac{1}{p}\sum_{\mu=1}^{p} \mathcal{E}(t_\mu, f(\boldsymbol{\xi}^\mu; \boldsymbol{w})$$

Average only over the p *available* questions $\{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^p\}$

Check performance relative to the noisy answers $\{t_\mu\}$ given

Accessible error in learning is $E_t(\boldsymbol{w})$.

generalization error

$$E_g = \int_\Omega \mathrm{d}\boldsymbol{\xi}\, p(\boldsymbol{\xi}) \mathcal{E}(T(\boldsymbol{\xi}), S(\boldsymbol{\xi}))$$

$$\downarrow$$

$$E_g(\boldsymbol{w}) = \int_\Omega \mathrm{d}\boldsymbol{\xi} p(\boldsymbol{\xi}) \mathcal{E}(T(\boldsymbol{\xi}), f(\boldsymbol{\xi}^\mu; \boldsymbol{w})$$

Involves *all* questions in Ω

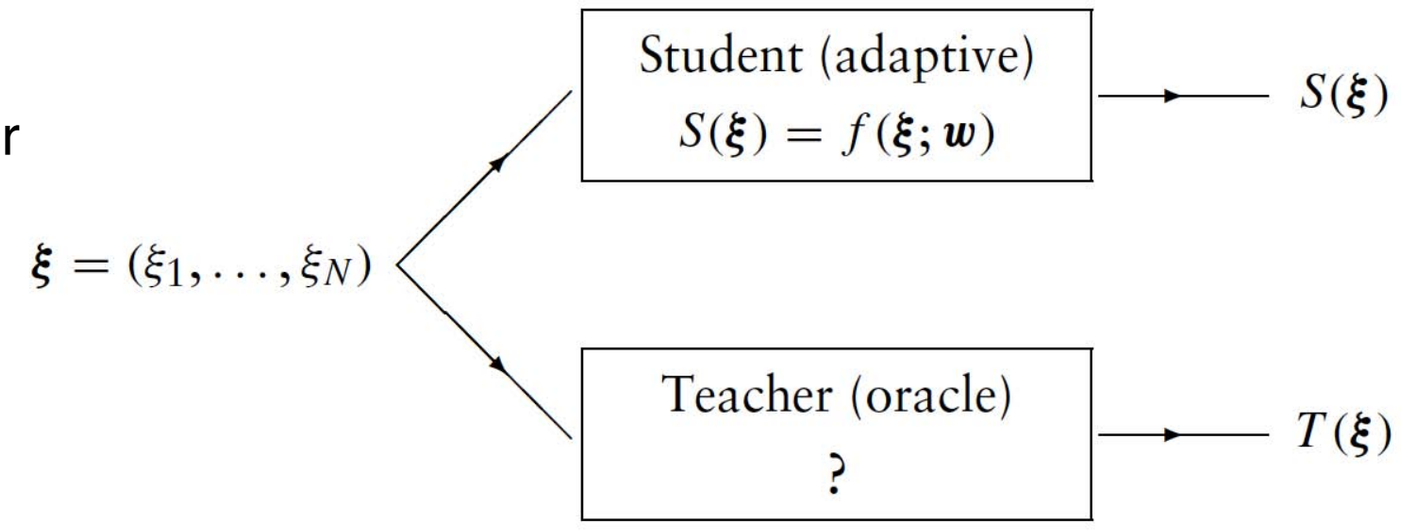
Check performance relative to the correct underlying rule T.

Real object of learning is to minimize $E_g(\boldsymbol{w})$.

Strategy in the early stages of neural computation…

$$E_t = \frac{1}{p}\sum_{\mu=1}^{p} \mathcal{E}(t_\mu, S(\boldsymbol{\xi}^\mu)) \qquad \text{training error}$$

$$\mathcal{E}(x, y) = \frac{1}{2}(x-y)^2$$



Find the best $\boldsymbol{w}$ by performing gradient descent on the surface $E_t(\boldsymbol{w})$ with learning rate η

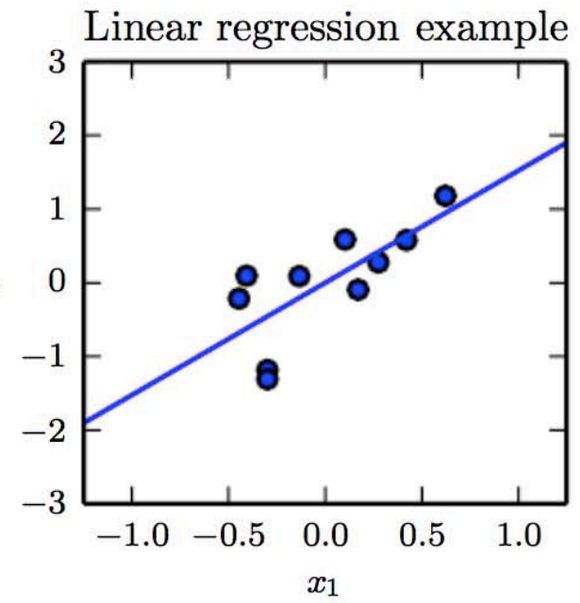
$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta \nabla_{\boldsymbol{w}} E_t(\boldsymbol{w}) \qquad E_t(\boldsymbol{w}) = \frac{1}{2p}\sum_{\mu=1}^{p}[t_\mu - f(\boldsymbol{\xi}^\mu; \boldsymbol{w})]^2$$

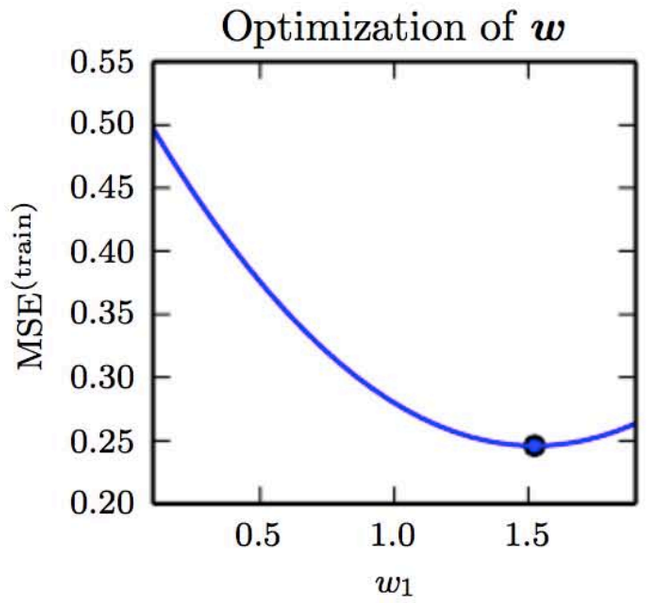
Look for $\boldsymbol{w}^*$ | $\min_{\mathbf{w}} E_t(\boldsymbol{w}) = E_t(\boldsymbol{w}^*)$

Above process leads to (at least) local minimum

$$\frac{\mathrm{d}}{\mathrm{d}t}E_t(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} E_t(\boldsymbol{w}) \cdot \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta\,(\nabla_{\boldsymbol{w}} E_t(\boldsymbol{w}))^2 \leq 0$$

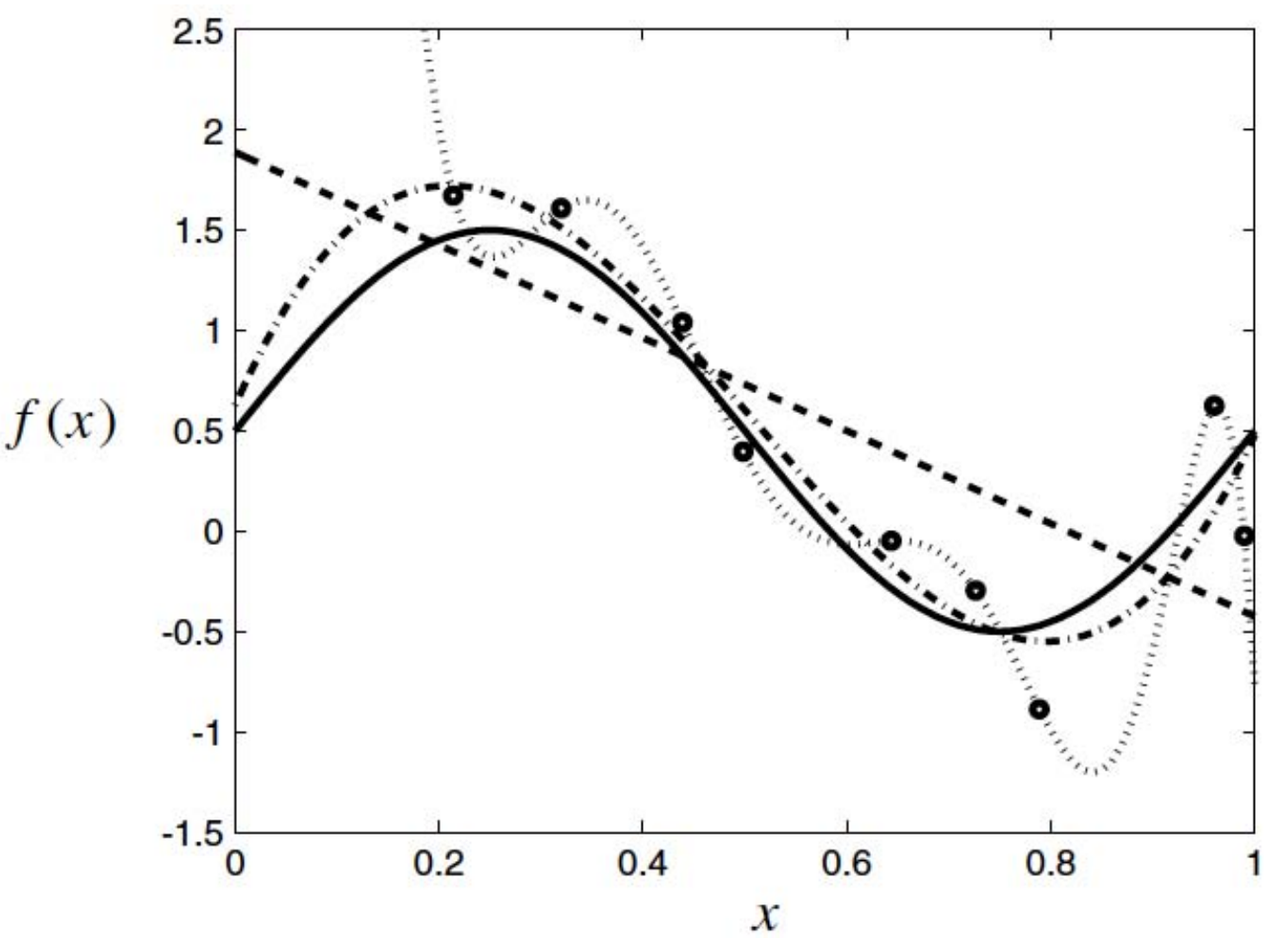
$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = 0 \iff \nabla_{\boldsymbol{w}} E_t(\boldsymbol{w}) = 0$$



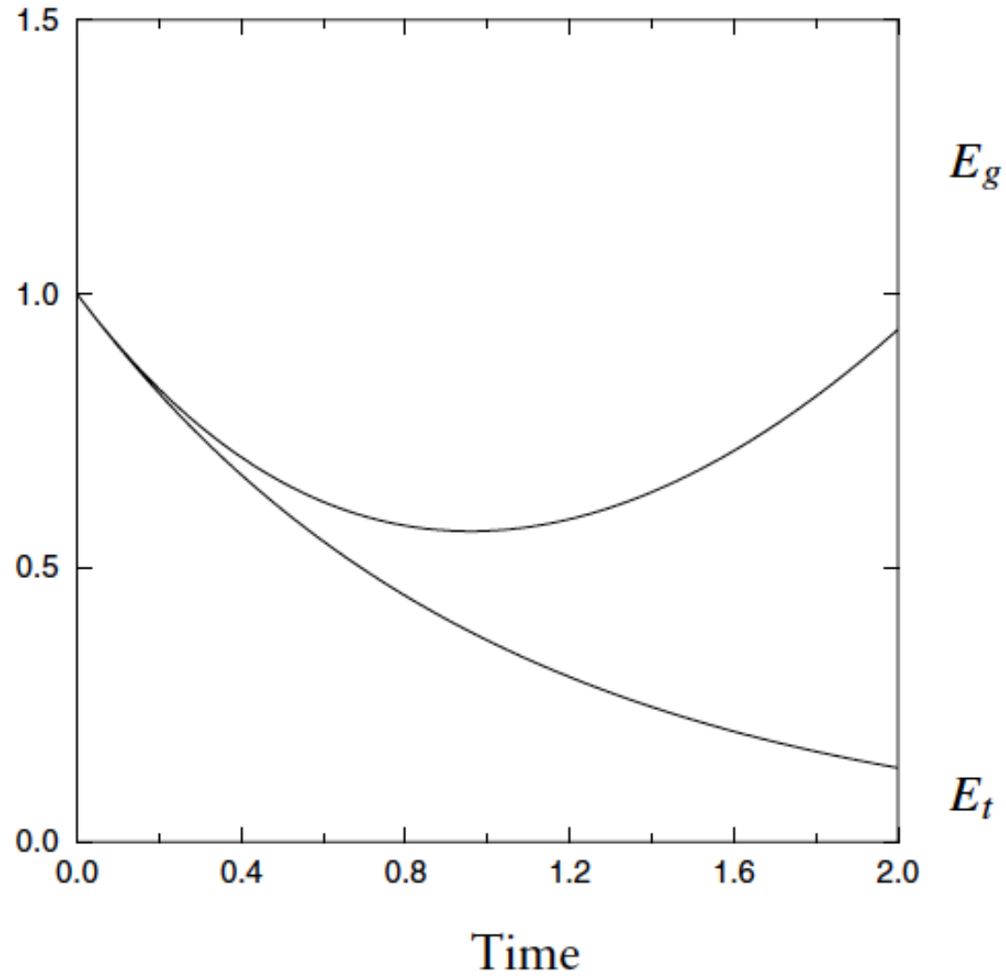


Possible problems

- Local, rather than global, minimum
- Gradient descent not the optimal process to find minima on an error surface
- Overfitting



Learning process where a system learns via gradient descent to approximate a function g(x) = 1/2 + sin(2πx) (solid line) by an M-th order polynomial $f(x;\mathbf{w}) = w0 + w_1x + \ldots + w_Mx^M$, on the basis of nine noisy sample points (circles). The resulting polynomials are shown for M = 1 (dashed), M = 3 (dot-dashed), and M = 9 (dotted).



Evolution of training and generalization errors during a learning process defined as gradient descent on an error surface, with noisy data, in the regime where the student starts overfitting, that is, reproducing also the noise in the data.

We trained the model by minimizing the training error, but we actually care about the test error.

How affect performance on test set when we only observe training set?

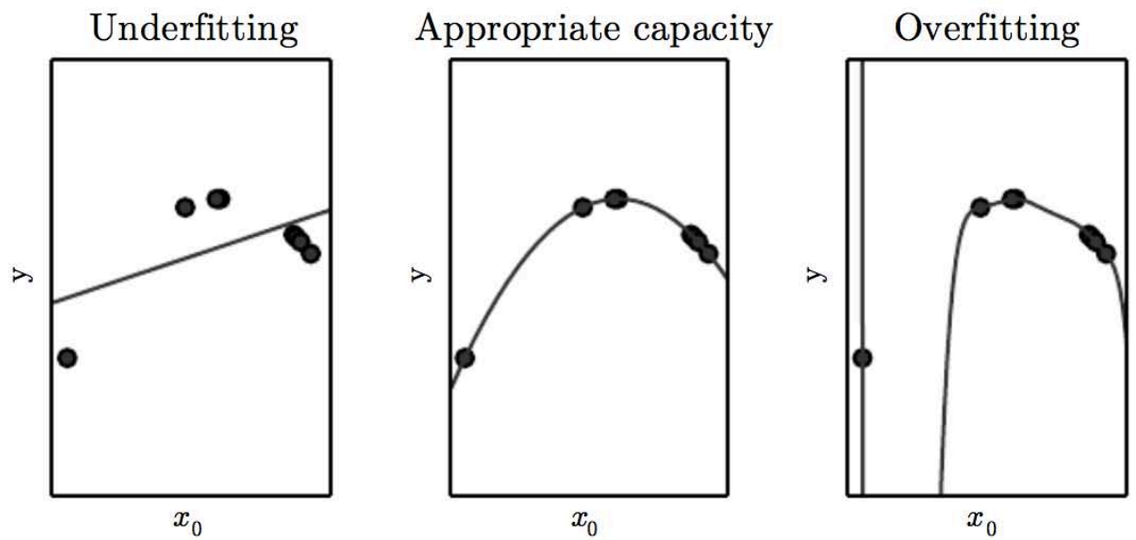
For a given $\boldsymbol{w}$, the expected E_t equals the expected E_g

Under machine learning algorithm, fix $\boldsymbol{w}$ to minimize E_t
➛ $E_t \leq E_g$

1. Make E_t small
2. Make the gap between E_t and E_g small

Generalization

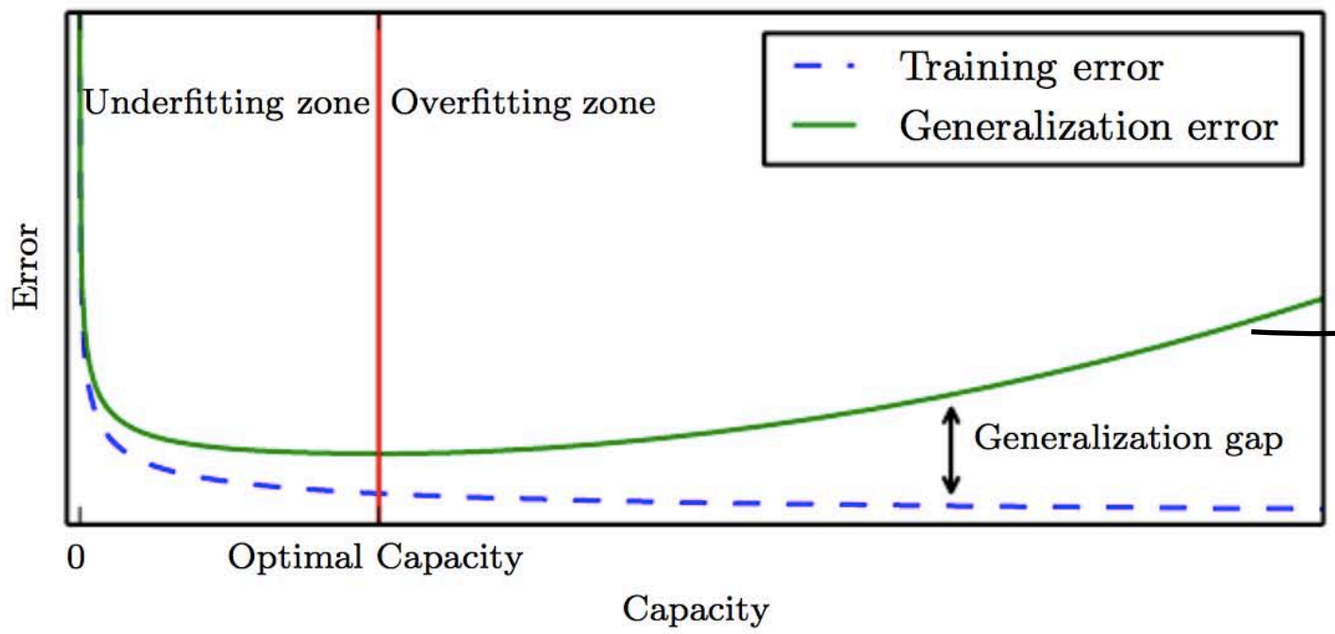
Tune the model capacity
$S = f(\boldsymbol{\xi}; \boldsymbol{w})$
$f(x;\boldsymbol{w}) = \mathbf{w} \cdot \boldsymbol{\Phi}_{\mathbf{M}}(x)$, $\boldsymbol{\Phi}_{\mathbf{M},\ell}(x)=x^{\ell}$
$p \geq M+1$
Learning means adopting $\boldsymbol{w} = (\boldsymbol{w_0}, \boldsymbol{w_1}, \ldots, \boldsymbol{w_M})$



Cross-validation

$\{(\boldsymbol{\xi}^\mu, t_\mu)\}_{\mu=1,\ldots,p} = \{(\boldsymbol{\xi}^\mu, t_\mu)\}_{\mu=1,\ldots,k} \cup \{(\boldsymbol{\xi}^\mu, t_\mu)\}_{\mu=k+1,\ldots,p}$

$$E_t(\boldsymbol{w}) = \frac{1}{k}\sum_{\mu=1}^{k} \mathcal{E}(t_\mu, f(\boldsymbol{\xi}^\mu; \boldsymbol{w})) \qquad E_g(\boldsymbol{w}) \approx \frac{1}{p-k}\sum_{\mu=k+1}^{p} \mathcal{E}(t_\mu, f(\boldsymbol{\xi}^\mu; \boldsymbol{w}))$$



Here I am fitting noise as well

Gap $\leq \bar{G}$ ↑Capacity ↓ p

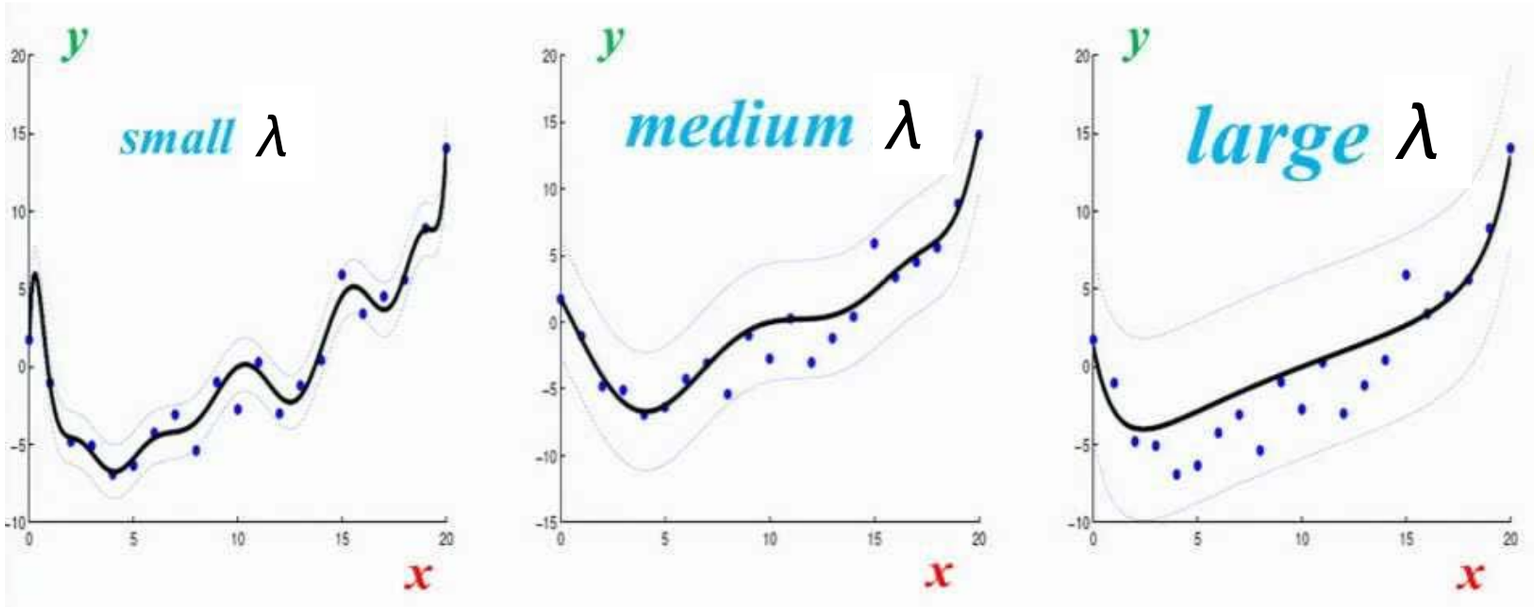
Discrepancy between training error and generalization error is bounded from above by a quantity that grows as the model capacity (informally, a model's capacity is its ability to fit a wide variety of functions) grows but shrinks as the number of training examples increases. These bounds provide intellectual justification that machine learning algorithms can work, but they are rarely used in practice in part because the bounds are quite loose and in part because it can be quite difficult to determine the capacity of (deep) learning algorithms.

V. Vapnik (1995) *The Nature of Statistical Learning Theory,* Springer-Verlag

Regularization

Both *T* (to be learned) and *S* (the trial function) are in principle smooth
Any observed irregularity and discontinuity of data must have been due to noise
The only way for the student to reproduce irregular or discontinuous functions is by having large and possibly diverging components of the parameter vector ***w***
→ prevent ***w*** from becoming too large by adding a suitable penalty term to the function to be minimized

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta \nabla_{\boldsymbol{w}} \left[E_t(\boldsymbol{w}) + \frac{1}{2}\lambda \boldsymbol{w}^2 \right]$$



Generalization

$f(x;\boldsymbol{w}) = \mathbf{w}\cdot\boldsymbol{\Phi}_{\mathbf{M}}(x)$, $\boldsymbol{\Phi}_{\mathbf{M},\ell}(x)=x^{\ell}$

Cross-validation

$\{(\boldsymbol{\xi}^{\mu}, t_{\mu})\}_{\mu=1,\ldots,p} = \{(\boldsymbol{\xi}^{\mu}, t_{\mu})\}_{\mu=1,\ldots,k} \cup \{(\boldsymbol{\xi}^{\mu}, t_{\mu})\}_{\mu=k+1,\ldots,p}$

Regularization

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta\nabla_{\boldsymbol{w}}\left[E_t(\boldsymbol{w}) + \frac{1}{2}\lambda\boldsymbol{w}^2\right]$$

But neither of them ensures the global minimum!

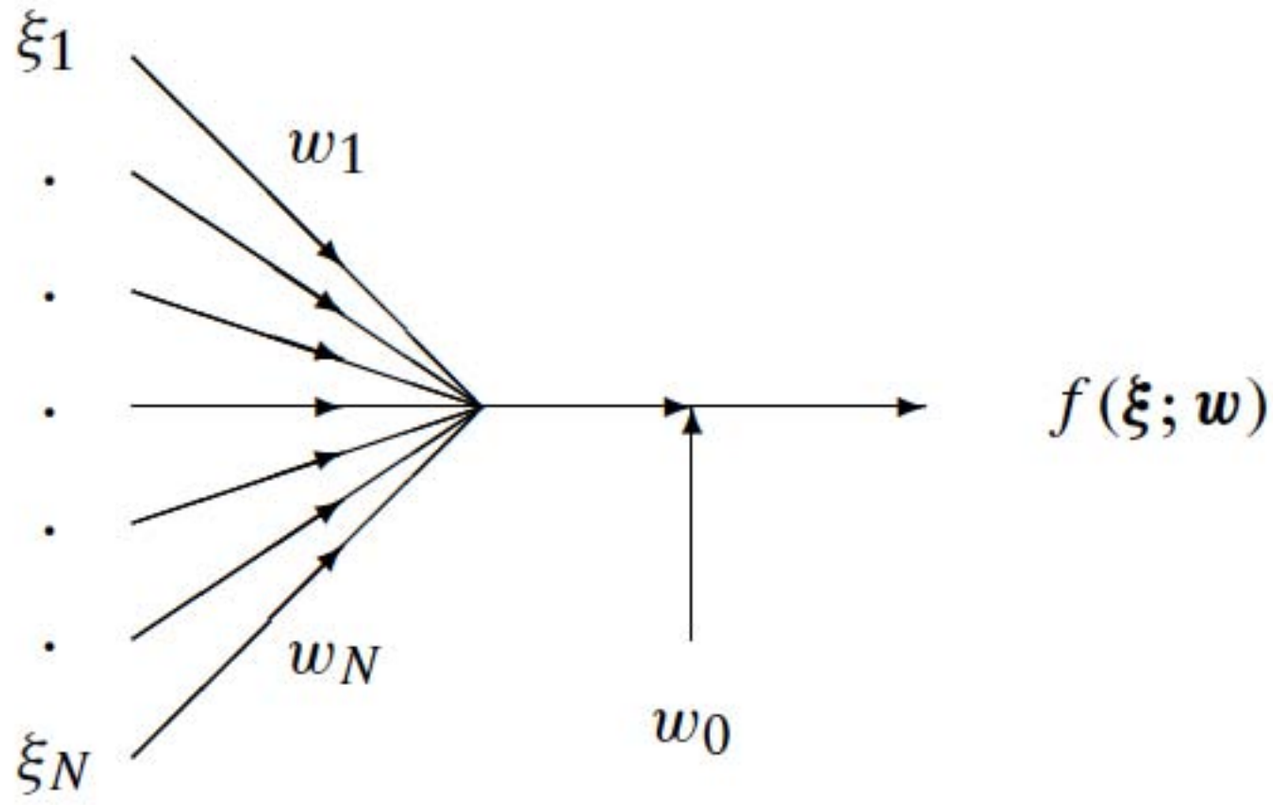
Generalization can yield to overfitting
Cross-validation wastes data and CPU time
Regularization guesses the form of the penalty term and its coefficient

… we still try to minimize E_t which differs from the real object E_g, which, in turn, is not accessible…

- Single layer NN (perceptron)

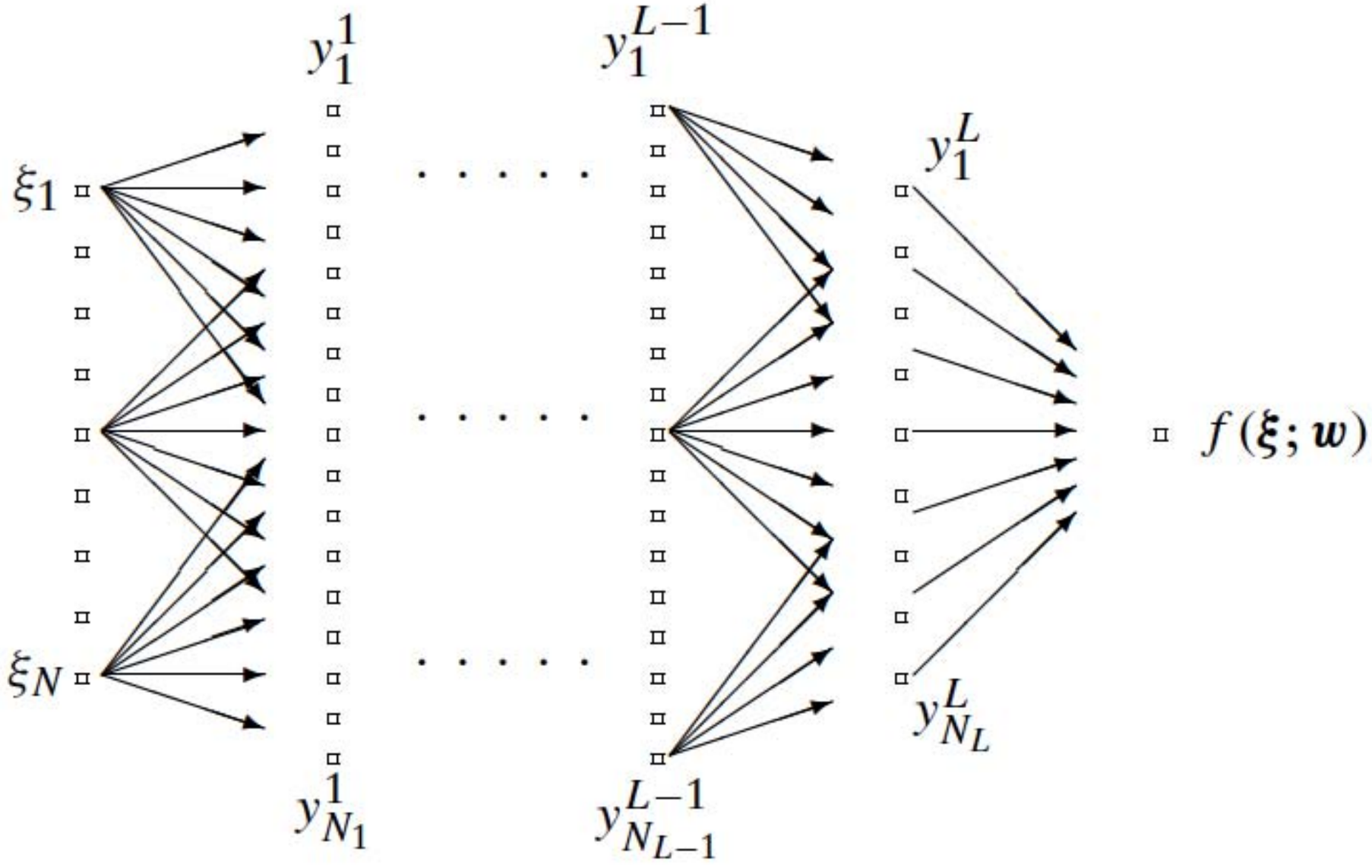
output

$$f(\boldsymbol{\xi};\boldsymbol{w}) = \boldsymbol{w}\boldsymbol{\xi} + w_0$$



This works for linear regression (the function is linear in ***w***), namely for linearly separable inputs

- Multi layer NN (or multi layer perceptron)



output

$$f(\boldsymbol{\xi};\boldsymbol{w}) = \sum_{j=1}^{N_L} w_j^L y_j^L(\boldsymbol{\xi}) + w_0^L$$

This system is proven to work as <u>universal approximators</u> (as long as *L* can be made arbitrarily large)

Bayesian learning

Set of noisy data $D = \{(\boldsymbol{\xi}^1, t_1), \ldots, (\boldsymbol{\xi}^p, t_p)\}$

generated by a system $S(\boldsymbol{\xi}) = f(\boldsymbol{\xi}; \boldsymbol{w})$ + noise

Ensemble of parameter vectors $p(\boldsymbol{w})$ ➛ prior distribution

Use Bayesian relation P(AIB)P(B) = P(BIA)P(A)
to express the desired object $p(\boldsymbol{w}|D)$ in terms of $p(D|\boldsymbol{w})$

Learning as a process during which the arrival of data reduces our uncertainty about the "right" parameter vector $\boldsymbol{w}$ from a prior distribution $p(\boldsymbol{w})$ (reflects prior knowledge or prejudice) to a posterior distribution $p(\boldsymbol{w}|D)$ (reflects both prior assumptions and the evidence from data)

$$p(D|\boldsymbol{w})p(\boldsymbol{w}) = p(\boldsymbol{w}|D)p(D)$$

$$p(D) = \int \mathrm{d}\boldsymbol{w} p(\boldsymbol{w}, D)$$

$$\Rightarrow p(\boldsymbol{w}|D) = \frac{p(D|\boldsymbol{w})p(\boldsymbol{w})}{\int \mathrm{d}\boldsymbol{w'} p(\boldsymbol{w'})p(D|\boldsymbol{w})}$$

Stage 1: definitions. Define the parametrized model, assumed responsbile for the data, the prior distribution $p(\boldsymbol{w})$ of its parameters, and the data $D=\{(\boldsymbol{\xi}^1, t_1), \ldots, (\boldsymbol{\xi}^p, t_p)\}$.

Stage 2: model translation. Convert the model definition into a standard probabilistic form, that is, specify the likelihood of finding output t upon presentation of input $\boldsymbol{\xi}$, given the parameters $\boldsymbol{w}$:

$$p(t \mid \boldsymbol{\xi}, \boldsymbol{w})$$

Stage 3: the posterior distribution. Calculate the data likelihood $p(D|\boldsymbol{w}) = \prod_\mu p(t_\mu| \boldsymbol{\xi}^\mu,\boldsymbol{w})$, as a function of the prameters $\boldsymbol{w}$.

$$p(\boldsymbol{w}|D) = \frac{p(D|\boldsymbol{w})p(\boldsymbol{w})}{\int \mathrm{d}\boldsymbol{w}' p(\boldsymbol{w}')p(D|\boldsymbol{w})} \Rightarrow p(\boldsymbol{w}|D) = \frac{p(\boldsymbol{w}) \prod_{\mu=1}^{p} p(t_\mu|\boldsymbol{\xi}^\mu, \boldsymbol{w})}{\int \mathrm{d}\boldsymbol{w}' p(\boldsymbol{w}') \prod_{\mu=1}^{p} p(t_\mu|\boldsymbol{\xi}^\mu, \boldsymbol{w}')}$$

Stage 4: prediction. The residual uncertainty in the parameters $\boldsymbol{w}$ generates uncertainity in the subsequent data prediction. Prediction of the output t corresponding to a new input $\boldsymbol{\xi}$, given our observation of the data D and our choice of the model, thus takes the probabilistic form:

$$p(t|\boldsymbol{\xi}, D) = \int \mathrm{d}\boldsymbol{w}\, p(t|\boldsymbol{\xi}, \boldsymbol{w})\, p(\boldsymbol{w}|D)$$

★ data are iid for a given $\boldsymbol{w}$.
care with joint distribution.

Advantages

- there is no need for cross-validation → all data can be used for learning
- allows for the selection of the optimal complexity and for the optimal model
- provides not only predictions, but also specific estimates of our confidence in these predictions (i.e., error bars)
- traditional learning via gradient descent training error minimization, with regularization, is recovered as a particular approximation within the Bayesian framework

Examples

1. Deterministic model, one real-valued input and one real-valued output. Definitions:

$$t(\xi) = \tanh(w\xi)$$

$$p(w) = (2\pi)^{-1/2} e^{-w^2/2}$$

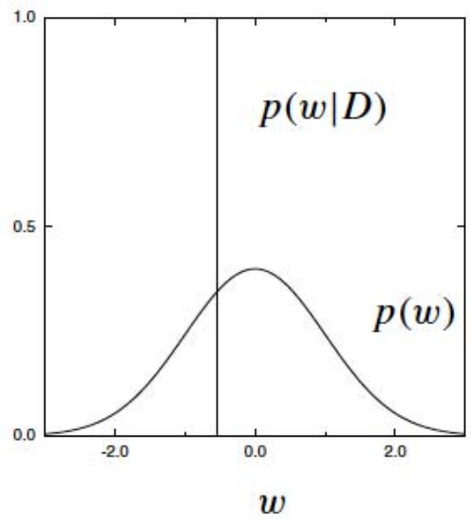
$$D = (1, -1/2)$$

2. Convert model into probabilistic form $p(t|\xi, w) = \delta(t - \tanh(w\xi))$

3. Calculate the posterior $p(w|D) = \frac{p(w)p(t_1|\xi^1, w)}{\int dw' p(w')p(t_1|\xi^1, w')} = \frac{\delta(-(1/2) - \tanh(w))e^{-w^2/2}}{\int dw' \delta(-(1/2) - \tanh(w'))e^{-w'^2/2}} =$

$$= \delta(w - \operatorname{atanh}(-1/2)) = \delta(w + \log\sqrt{3})$$

Before data, uncertainty about the parameter *w* described by the Gaussian prior *p*(w); after data, collapse of the prior to the δ-distributed posterior



1. Deterministic model, one real-valued input and one real-valued output, two-parameters. Definitions:

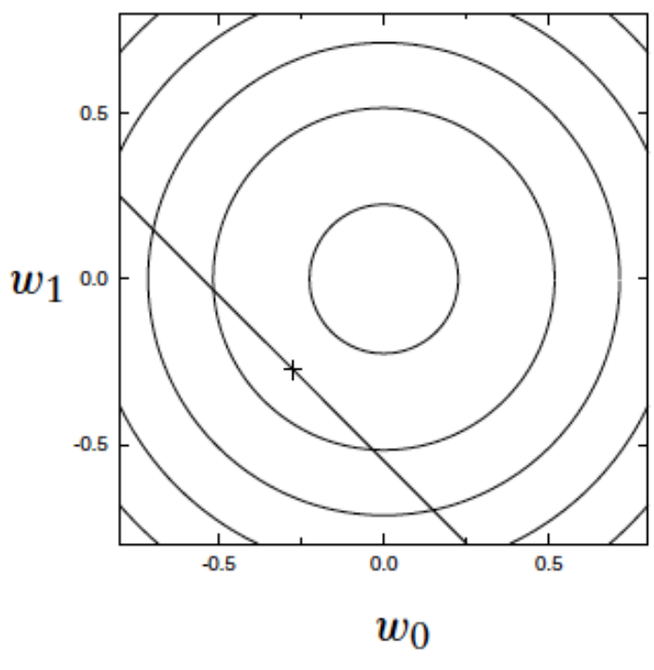
$$t(\xi) = \tanh(w_0 + w_1\xi)$$
$$p(\boldsymbol{w}) = (2\pi)^{-1} e^{-\boldsymbol{w}^2/2}$$
$$D = (1, -1/2)$$

2. Convert model into probabilistic form $p(t|\xi, \boldsymbol{w}) = \delta(t - \tanh(w_0 + w_1\xi))$

3. Calculate the posterior $p(\boldsymbol{w}|D) = \dfrac{p(\boldsymbol{w})p(t_1|\xi^1, \boldsymbol{w})}{\int d\boldsymbol{w}' p(\boldsymbol{w}')p(t_1|\xi^1, \boldsymbol{w}')} = \dfrac{\delta(-(1/2) - \tanh(w_0 + w_1))e^{-\boldsymbol{w}^2/2}}{\int d\boldsymbol{w}' \delta(-(1/2) - \tanh(w_0' + w_1'))e^{-\boldsymbol{w}'^2/2}} =$

$$= \frac{\delta(w_0 + w_1 + \operatorname{atanh}(1/2))e^{-\boldsymbol{w}^2/2}}{\int d\boldsymbol{w}' \delta(w_0' + w_1' + \operatorname{atanh}(1/2))e^{-\boldsymbol{w}'^2/2}} =$$

$$= \frac{\delta(w_0 + w_1 + \log\sqrt{3})e^{-\boldsymbol{w}^2/2}}{\int d\boldsymbol{w}' \delta(w_0' + w_1' + \log\sqrt{3})e^{-\boldsymbol{w}'^2/2}}$$



The posterior is nonzero only along the line $w_0 + w_1 = -\log\sqrt{3}$ in the parameter plane
Along that line it is maximal for the choice $w_0 = w_1 = -\log\sqrt{3/2}$

Before data, uncertainty about the parameter $\boldsymbol{w}$ decribed by the Gaussian prior $p(\boldsymbol{w})$; after data, collapse of the prior to a slice of the two-dimentional Gaussian prior
Most likely parameter vector $\boldsymbol{w}_{MP} = (-\log\sqrt{3/2}, -\log\sqrt{3/2})$ and we have also quantitative information about uncertainty on $\boldsymbol{w}_{MP}$

The link between Bayesian learning and "traditional" learning

Traditional gradient descent learning on an error surface, possibly with regularization

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta\nabla_{\boldsymbol{w}}[E_t(\boldsymbol{w}) + \lambda E_w(\boldsymbol{w})]$$

$$E_t(\boldsymbol{w}) = \frac{1}{p}\sum_{\mu=1}^{p}\mathcal{E}(t_\mu - f(\boldsymbol{\xi}^\mu;\boldsymbol{w}))$$

$$p(\boldsymbol{w}|D) = \frac{p(\boldsymbol{w})\prod_{\mu=1}^{p}p(t_\mu|\boldsymbol{\xi^\mu},\boldsymbol{w})}{\int \mathrm{d}\boldsymbol{w'}p(\boldsymbol{w'})\prod_{\mu=1}^{p}p(t_\mu|\boldsymbol{\xi^\mu},\boldsymbol{w'})}$$

$$\rightarrow \log p^{-1}(\boldsymbol{w}|D) = -\log p(\boldsymbol{w}) - \sum_{\mu=1}^{p}\log p(t_\mu|\boldsymbol{\xi^\mu},\boldsymbol{w}) + \log\int \mathrm{d}\boldsymbol{w'}p(\boldsymbol{w'})\prod_{\mu=1}^{p}p(t_\mu|\boldsymbol{\xi^\mu},\boldsymbol{w'})$$

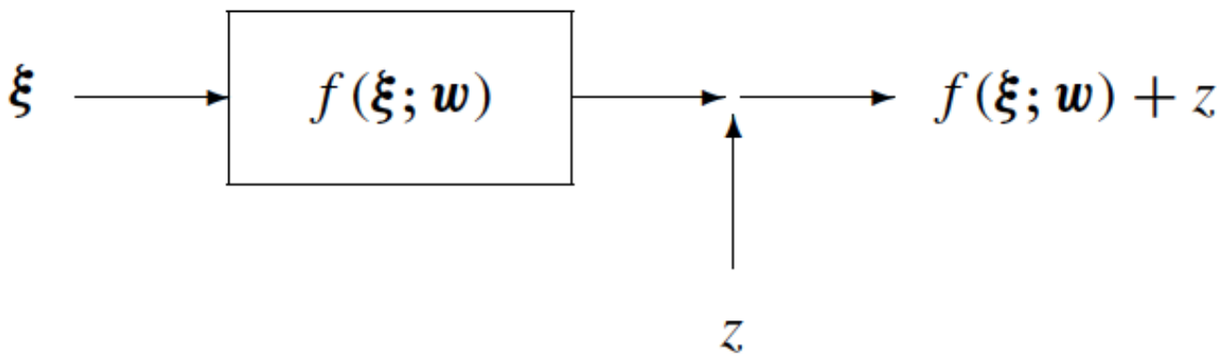
The most probable parameter vector $\boldsymbol{w}_{\mathrm{MP}}$, given the data D, is found by minimizing $\log p^{-1}(\boldsymbol{w}|D)$

$$\rightarrow \min_{\boldsymbol{w}} S(\boldsymbol{w}, D),$$

$$S(\boldsymbol{w}, D) = -\log p(\boldsymbol{w}) - \sum_{\mu=1}^{p}\log p(t_\mu|\boldsymbol{\xi^\mu},\boldsymbol{w})$$

$$\rightarrow \min_{\boldsymbol{w}} S(\boldsymbol{w}, D),$$

$$S(\boldsymbol{w}, D) = -\log p(\boldsymbol{w}) - \sum_{\mu=1}^{p} \log p(t_\mu | \boldsymbol{\xi}^{\boldsymbol{\mu}}, \boldsymbol{w})$$



$$t = f(\boldsymbol{\xi}; \boldsymbol{w}) + z$$

z zero-average random number drawn from *P*(*z*)

$$p(t|\boldsymbol{\xi}; \boldsymbol{w}) = P(t - f(\boldsymbol{\xi}; \boldsymbol{w}))$$

$$\frac{1}{p} S(\boldsymbol{w}, D) = -\frac{1}{p} \sum_{\mu=1}^{p} \log P(t_\mu - f(\boldsymbol{\xi}^\mu; \boldsymbol{w})) - \frac{1}{p} \log p(\boldsymbol{w})$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{w} = -\eta \nabla_{\boldsymbol{w}} [E_t(\boldsymbol{w}) + \lambda E_w(\boldsymbol{w})]$$

$$E_t(\boldsymbol{w}) = \frac{1}{p} \sum_{\mu=1}^{p} \mathcal{E}(t_\mu - f(\boldsymbol{\xi}^\mu; \boldsymbol{w}))$$

min on a training error surface, with regularizer $\longleftrightarrow$ most probable parameter vector $\mathbf{w}_{MP}$

$\mathcal{E}(u) \longleftrightarrow P(z) \sim e^{-\mathcal{E}(z)}$

specific regularizer $\lambda E_w(\boldsymbol{w}) \longleftrightarrow$ prior $p(\boldsymbol{w}) \sim e^{-p\lambda Ew(\boldsymbol{w})}$

$\lambda \sim p^{-1}$

Example

$$\mathcal{E}(u) = \frac{1}{2}u^2$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta\nabla_{\boldsymbol{w}}\left\{\frac{1}{2p}[t_\mu - f(\boldsymbol{\xi}^{\boldsymbol{\mu}};\boldsymbol{w})]^2 + \frac{1}{2}\lambda\boldsymbol{w}^2\right\}$$

equivalent to

$$p(t|\boldsymbol{\xi};\boldsymbol{w}) = \left(\frac{\beta}{2\pi}\right)^{1/2} e^{-\beta[t-f(\boldsymbol{\xi};\boldsymbol{w})]^2/2}$$

Gaussian model with variance β

$$p(\boldsymbol{w}) = \left(\frac{\alpha}{2\pi}\right)^{M/2} e^{-\alpha\boldsymbol{w}^2/2}$$

Gaussian prior with variance α ($\boldsymbol{w}\in\mathbb{R}^M$)

leads to

$$S(\boldsymbol{w};D) = \frac{\beta}{2}\sum_{\mu=1}^{p}[t_\mu - f(\boldsymbol{\xi}^\mu;\boldsymbol{w})]^2 + \frac{\alpha}{2}\boldsymbol{w}^2 = \beta p\left\{\frac{1}{2p}\sum_{\mu=1}^{p}[t_\mu - f(\boldsymbol{\xi}^\mu;\boldsymbol{w})]^2 + \frac{1}{2}\frac{\alpha}{\beta p}\boldsymbol{w}^2\right\}$$

Example

$$\mathcal{E}(u) = \frac{1}{2}u^2$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\eta\nabla_{\boldsymbol{w}}\left\{\frac{1}{2p}[t_\mu - f(\boldsymbol{\xi^\mu};\boldsymbol{w})]^2 + \frac{1}{2}\lambda\boldsymbol{w}^2\right\}$$

equivalent to

$$p(t|\boldsymbol{\xi};\boldsymbol{w}) = \left(\frac{\beta}{2\pi}\right)^{1/2} e^{-\beta[t-f(\boldsymbol{\xi};\boldsymbol{w})]^2/2}$$

Gaussian model with variance β

$$p(\boldsymbol{w}) = \left(\frac{\alpha}{2\pi}\right)^{M/2} e^{-\alpha\boldsymbol{w}^2/2}$$

Gaussian prior with variance α ($\boldsymbol{w}\in\mathbb{R}^M$)

leads to

$$S(\boldsymbol{w};D) = \frac{\beta}{2}\sum_{\mu=1}^{p}[t_\mu - f(\boldsymbol{\xi}^\mu;\boldsymbol{w})]^2 + \frac{\alpha}{2}\boldsymbol{w}^2 = \beta p\left\{\frac{1}{2p}\sum_{\mu=1}^{p}[t_\mu - f(\boldsymbol{\xi}^\mu;\boldsymbol{w})]^2 + \frac{1}{2}\frac{\alpha}{\beta p}\boldsymbol{w}^2\right\}$$

no need to guess the value of λ: can be related to our assumptions about the prior and the noise level

$$\lambda = \frac{\alpha}{\beta p} \quad \alpha = \frac{1}{\langle w_i^2\rangle_{\text{prior}}} \quad \beta = \frac{1}{\langle t^2\rangle_{\text{noise}} - \langle t\rangle^2_{\text{noise}}}$$

hyperparameters (not adjustable)

Approximation of the posterior parameter distribution

$$S(\boldsymbol{w}, D) = -\log p(\boldsymbol{w}) - \sum_{\mu=1}^{p} \log p(t_\mu | \boldsymbol{\xi^\mu}, \boldsymbol{w})$$

$$\Rightarrow p(\boldsymbol{w}|D) = \frac{e^{-S(\boldsymbol{w},D)}}{\int \mathrm{d}\boldsymbol{w'}\ e^{-S(\boldsymbol{w'},D)}}$$

Hp: $S(\boldsymbol{w},D)$ has only a single relevant minimum $\boldsymbol{w}_{\mathrm{MP}}$

$$S(\boldsymbol{w}, D) = S(\boldsymbol{w}_{\mathrm{MP}}, D) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}) \cdot \mathbf{A}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}) + \mathcal{O}(|\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}|^3)$$

$$A_{ij} = \left.\frac{\partial^2 S}{\partial w_i \partial w_j}\right|_{\boldsymbol{w}_{\mathrm{MP}}}$$ Hessian of S at $\boldsymbol{w}_{\mathrm{MP}}$

Truncation after the quadratic term → Gaussian approximation of the posterior

$$S(\boldsymbol{w}, D) \to \tilde{S}(\boldsymbol{w}, D) = S(\boldsymbol{w}_{\mathrm{MP}}, D) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}) \cdot \mathbf{A}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}})$$

$$p(\boldsymbol{w}|D) \to \tilde{p}(\boldsymbol{w}|D) = \left[\frac{\det \mathbf{A}}{(2\pi)^M}\right]^{1/2} e^{-(\boldsymbol{w}-\boldsymbol{w}_{\mathrm{MP}})\cdot\mathbf{A}(\boldsymbol{w}-\boldsymbol{w}_{\mathrm{MP}})/2}$$

$\langle \boldsymbol{w} \rangle = \boldsymbol{w}_{\mathrm{MP}}, \quad \langle w_i w_j \rangle - \langle w_i \rangle \langle w_j \rangle = (\mathbf{A}^{-1})_{ij}$ connected correlation

where $\langle f(\boldsymbol{w}) \rangle = \int \mathrm{d}\boldsymbol{w} f(\boldsymbol{w}) \tilde{p}(\boldsymbol{w}|D)$

Approximation of the posterior parameter distribution

$$S(\boldsymbol{w}|D) = -\log p(\boldsymbol{w}) - \sum_{\mu=1}^{p} \log p(t_\mu|\boldsymbol{\xi}^{\boldsymbol{\mu}}, \boldsymbol{w})$$

$$\Rightarrow p(\boldsymbol{w}|D) = \frac{e^{-S(\boldsymbol{w},D)}}{\int \mathrm{d}\boldsymbol{w'}\ e^{-S(\boldsymbol{w'},D)}}$$

Hp: $S(\boldsymbol{w},D)$ has only a single relevant minimum $\boldsymbol{w}_{\mathrm{MP}}$

$$S(\boldsymbol{w}, D) = S(\boldsymbol{w}_{\mathrm{MP}}, D) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}) \cdot \mathbf{A}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}) + \mathcal{O}(|\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}|^3)$$

$$A_{ij} = \left.\frac{\partial^2 S}{\partial w_i \partial w_j}\right|_{\boldsymbol{w}_{\mathrm{MP}}}$$ Hessian of S at $\boldsymbol{w}_{\mathrm{MP}}$

Gaussian approximation under MEA equals exponentials of a quadratic form for the Hamiltonian, that is Classical Physics (energy are quadratic form and force are linears) is recovered

Truncation after the quadratic term → Gaussian approximation of the posterior

$$S(\boldsymbol{w}, D) \to \tilde{S}(\boldsymbol{w}, D) = S(\boldsymbol{w}_{\mathrm{MP}}, D) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}}) \cdot \mathbf{A}(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MP}})$$

$$p(\boldsymbol{w}|D) \to \tilde{p}(\boldsymbol{w}|D) = \left[\frac{\det \mathbf{A}}{(2\pi)^M}\right]^{1/2} e^{-(\boldsymbol{w}-\boldsymbol{w}_{\mathrm{MP}})\cdot\mathbf{A}(\boldsymbol{w}-\boldsymbol{w}_{\mathrm{MP}})/2}$$

$$\langle \boldsymbol{w} \rangle = \boldsymbol{w}_{\mathrm{MP}}, \quad \langle w_i w_j \rangle - \langle w_i \rangle \langle w_j \rangle = (\mathbf{A}^{-1})_{ij}$$