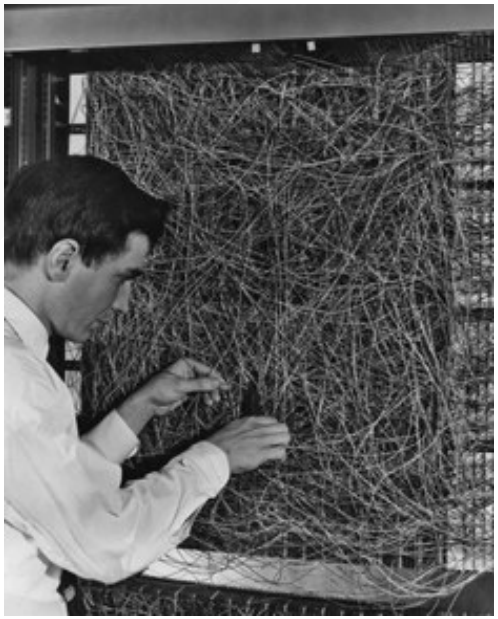
Introduction to Artificial Intelligence

Lecture 8: Learning

LIÈGE université

Today

- Learning
- Statistical learning
- Supervised learning
- Reinforcement learning
- Unsupervised learning

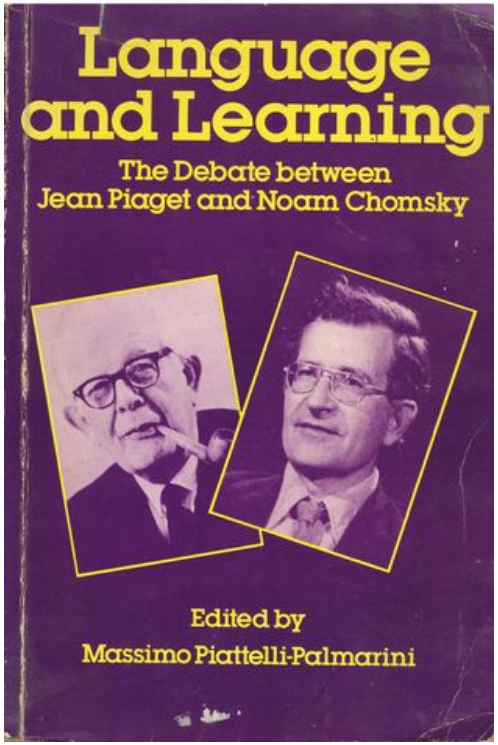


Intelligence?

- What we covered so far...
  - Search algorithms, using a state space specified by domain knowledge.
  - Adversarial search, for known and fully observable games.
  - Constraint satisfaction problems, by exploiting a known structure of the states.
  - Logical inference, using well-specified facts and inference rules.
  - Reasoning about uncertain knowledge, as represented using domain-motivated graphs.
- Enough to implement complex and rational behaviors, in some situations.
- But is that intelligence? Aren't we missing a critical component?

Chomsky vs. Piaget

- Noam Chomsky (innatism):
  - State that humans possess a genetically determined faculty for thought and language.
  - The structures of language and thought are set in motion through interaction with the environment.
- Jean Piaget (constructivism):
  - Deny the existence of innate cognitive structure specific for thought and language.
  - Postulate instead all cognitive acquisitions, including language, to be the outcome of a gradual process of construction, i.e., a learning procedure.



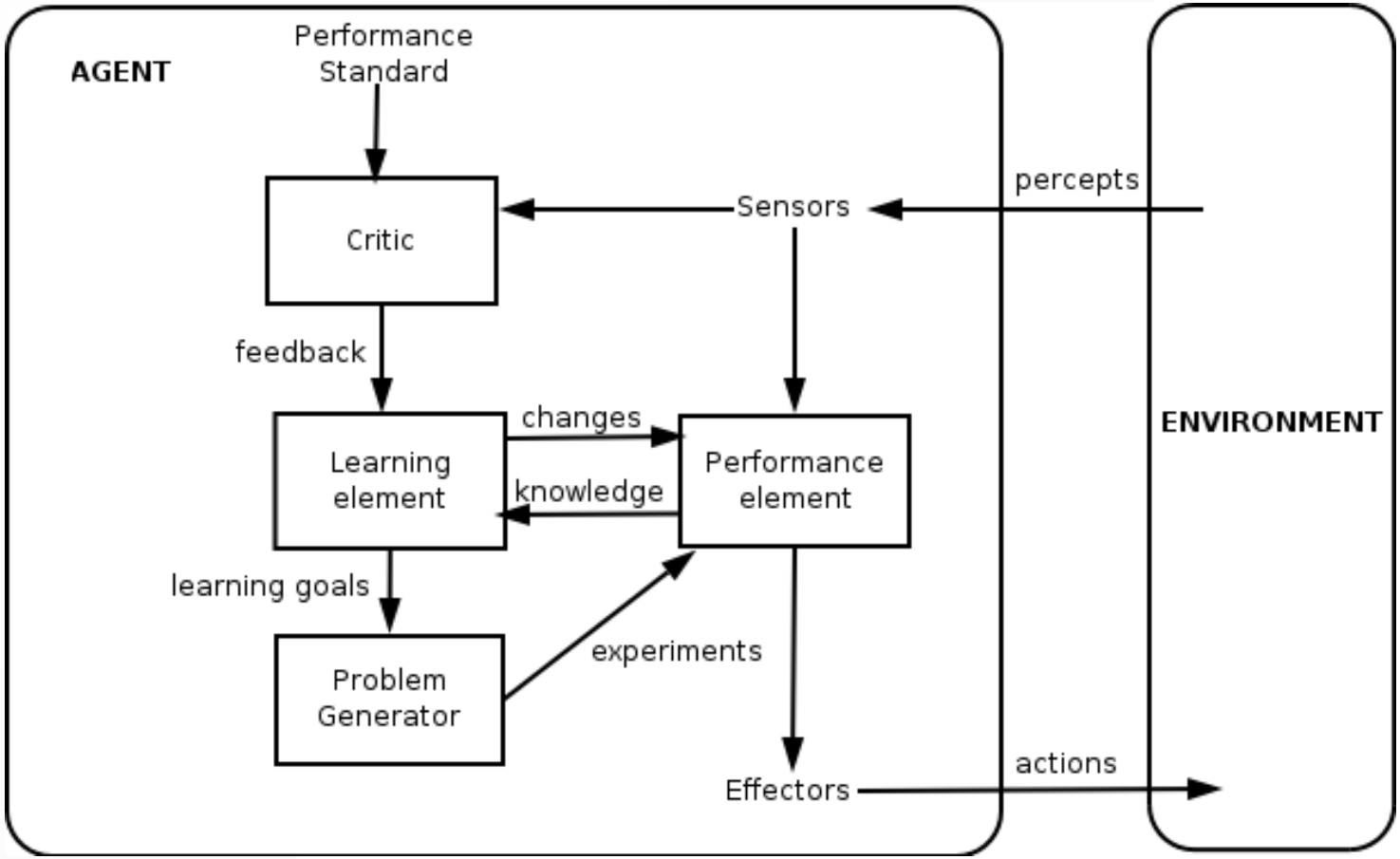
[Q] What about AI? Should it be a pre-wired efficient machine? Or a machine that can learn and improve? or maybe a bit of both?

The debate continues...

Learning

- What if the environment is unknown?
- Learning can be used as a system construction method.
  - i.e., expose the agent to reality rather trying to hardcode reality into the agent's program.
- Learning can be used as an automated way to modify the agent's decision mechanisms to improve performance.

Learning agents



Learning element

- The design of the learning element is dictated by:
  - What type of performance element is used.
  - Which functional component is to be learned.
  - How that functional component is represented.
  - What kind of feedback is available.
- Examples:

Performance element	Component	Representation	Feedback
Alpha–beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor–state axioms	Outcome
Utility–based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept–action fn	Neural net	Correct action

- The nature and frequency of the feedback often determines a learning strategy:
  - Supervised learning: correct answer for each instance.
  - Reinforcement learning: occasional rewards.
  - Unsupervised learning: no feedback!

Statistical learning

Learning probabilistic models

- Agents can handle uncertainty by using a specified probabilistic model of the world.
- This model may then be combined with decision theory to take actions.
- In lack of a good probabilistic model, what should an agent do?
- Learn it from experience with the world!

Bayesian learning

- View learning as Bayesian updating of probability distribution over the hypothesis space.
  - H is the hypothesis variable, values are $h_1, h_2, ...$ and the prior is $P(H)$.
  - $\mathbf{d}$ is the observed data.
- Given the data so far, each hypothesis has a posterior probability

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

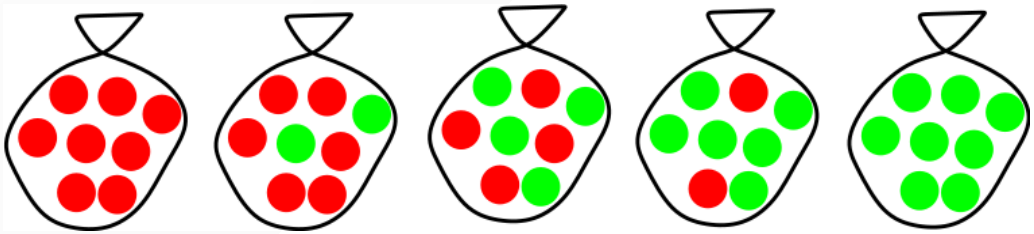
  where $P(\mathbf{d}|h_i)$ is called the likelihood.
- Predictions use a likelihood-weighted average over the hypotheses:

$$P(X|\mathbf{d}) = \sum_i P(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = P(X|h_i)P(h_i|\mathbf{d})$$

- No need to pick one best-guess hypothesis!

Example

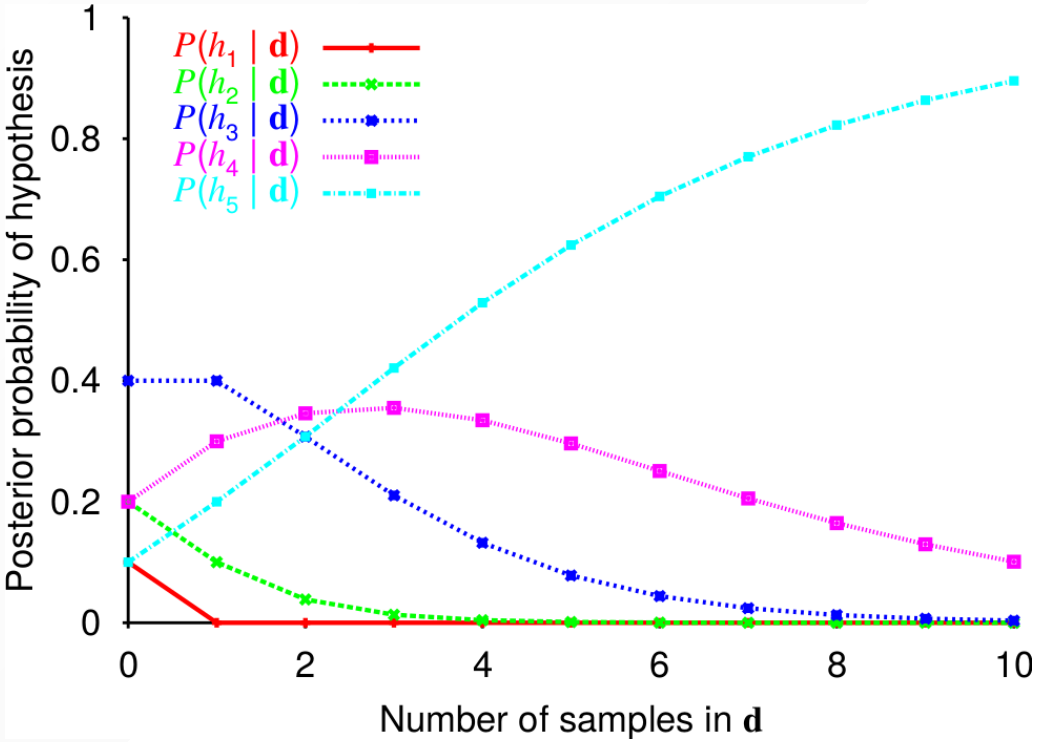
- Suppose there are five kinds of bags of candies. Assume a prior $P(H)$:
  - $P(h_1) = 0.1$, with h_1: 100% cherry candies
  - $P(h_2) = 0.2$, with h_2: 75% cherry candies + 25% lime candies
  - $P(h_3) = 0.4$, with h_3: 50% cherry candies + 50% lime candies
  - $P(h_4) = 0.2$, with h_4: 25% cherry candies + 75% lime candies
  - $P(h_5) = 0.1$, with h_5: 100% lime candies



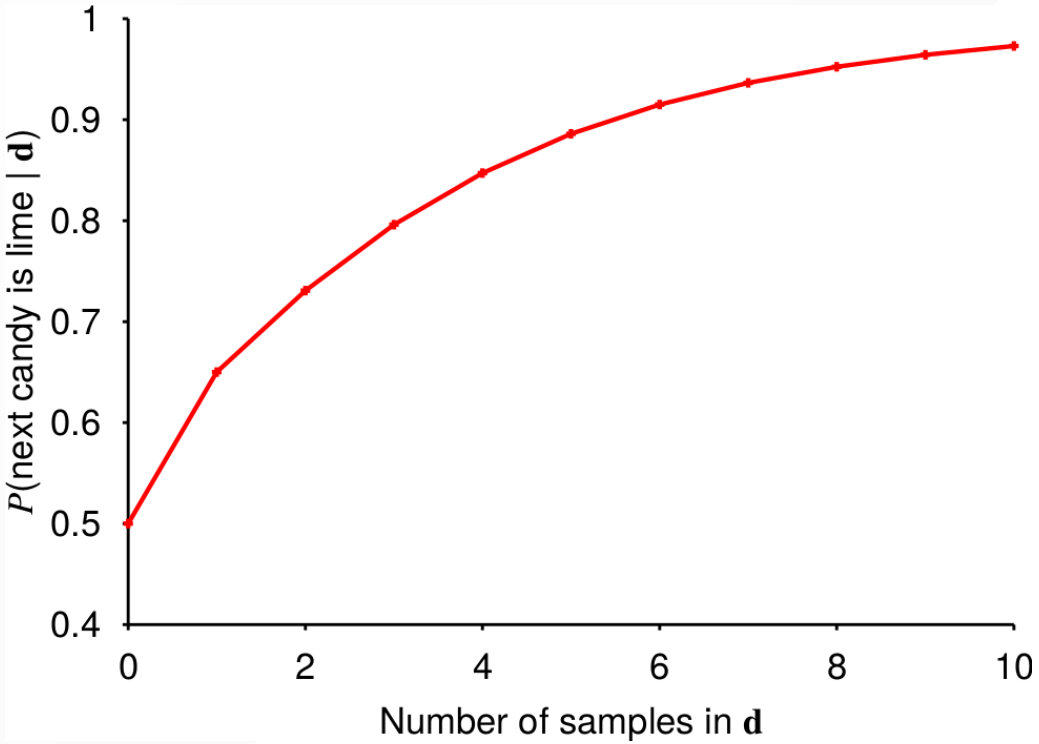
- Then we observe candies drawn from som bag:

- What kind of bag is it? What flavour will the next candy be?

Posterior probability of hypotheses



Prediction probability



- This example illustrates the fact that the Bayesian prediction eventually agrees with the true hypothesis.
- The posterior probability of any false hypothesis eventually vanishes.

MAP approximation

- Summing over the hypothesis space is often intractable.
  - e.g., there are 2^{2^n} n-ary boolean functions of boolean inputs.
- Maximum a posteriori (MAP) learning:

$$h_{MAP} = \arg\max_{h_i} P(h_i|\mathbf{d})$$

- That is, maximize $P(\mathbf{d}|h_i)P(h_i)$ or $\log P(\mathbf{d}|h_i) + \log P(h_i)$.
  - Log terms can be be viewed as (negative of) bits to encode data given hypothesis + bits to encode hypothesis.
  - This is the basic idea of minimum description length learning, i.e., Occam's razor.
- Finding the MAP hypothesis is often much easier than Bayesian learning, since it requires solving an optimization problem instead of a large summation problem.
- For deterministic hypotheses, $P(\mathbf{d}|h_i) = 1$ if h_i is consistent, and 0 otherwise.
  - Therefore, MAP yields the simplest consistent hypothesis.

Maximum likelihood

- For large data sets, the prior $P(H)$ becomes irrelevant.
- Maximum likelihood estimation (MLE):

$$h_{MLE} = \arg\max_{h_i} P(\mathbf{d}|h_i)$$

- That is, simply get the best fit to the data.
  - Identical to MAP for uniform prior.
- MLE is the standard (non-Bayesian) statistical learning method.
  - Procedure:
    - Choose a parameterized family of models to describe the data.
      - requires substantial insight and sometimes new models.
    - Write down the likelihood of the data as a function of the parameters.
      - may require summing over hidden variables, i.e., inference.
    - Write down the derivative of the log likelihood w.r.t. each parameter.
    - Find the parameter values such that the derivatives are zero.
      - may be hard/impossible; modern optimization techniques help.

Parameter learning BNs

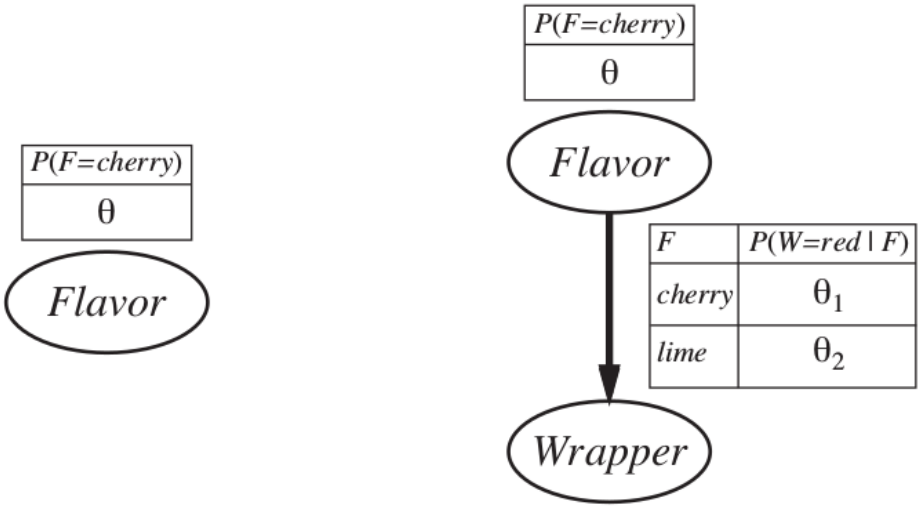


Figure 20.2 (a) Bayesian network model for the case of candies with an unknown proportion of cherries and limes. (b) Model for the case where the wrapper color depends (probabilistically) on the candy flavor.

MLE, case (a)

- Bag from a new manufacturer; fraction θ of cherry candies?
  - Any θ is possible: continuum of hypotheses h_θ.
  - θ is a parameter for this simple binomial family of models.
- Suppose we unwrap N candies, and get c cherries and $l = N - c$ limes.
- These are i.i.d. observations, so:

$$P(\mathbf{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^c(1-\theta)^l$$

- Maximize this w.r.t. θ, which is easier for the log-likelihood:
  - $L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = c\log\theta + l\log(1-\theta)$
  - $\frac{dL(\mathbf{d}|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0$, therefore $\theta = \frac{c}{N}$.
  - Seems sensible, but causes problems with 0 counts!

MLE, case (b)

- Red/green wrapper depends probabilistically on flavor.
- Likelihood for e.g. a cherry candy in green wrapper:

$$
\begin{aligned}
&P(F = cherry, W = green | h_{\theta,\theta_1,\theta_2}) \\
&= P(F = cherry | h_{\theta,\theta_1,\theta_2}) P(W = green | F = cherry, h_{\theta,\theta_1,\theta_2}) \\
&= \theta(1-\theta_1)
\end{aligned}
$$

- The likelihood for the data, given N candies, r_c red-wrapped cherries, g_c green-wrapped cherries, etc. is:

$$P(\mathbf{d} | h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^l \theta_1^{r_c} (1-\theta_1)^{g_c} \theta_2^{r_l} (1-\theta_2)^{g_l}$$

$$
\begin{aligned}
L = &\, c \log\theta + l \log(1-\theta) \\
&+ r_c \log\theta_1 + g_c \log(1-\theta_1) \\
&+ r_l \log\theta_2 + g_l \log(1-\theta_2)
\end{aligned}
$$

MLE, case (b), cont.

- Derivatives of L contain only the relevant parameter:
  - $\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c+l}$
  - $\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \quad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c+g_c}$
  - $\frac{\partial L}{\partial \theta_2} = \frac{r_l}{\theta_2} - \frac{g_l}{1-\theta_2} = 0 \quad \Rightarrow \quad \theta_2 = \frac{r_l}{r_l+g_l}$
- Again, results coincide with intuition.
- This can be extended to any Bayesian network with parameterized CPTs.
- Importantly, with complete data, maximum likelihood parameter learning for a Bayesian network decomposes into separate learning problems, one for each parameter.

MLE for linear Gaussian models

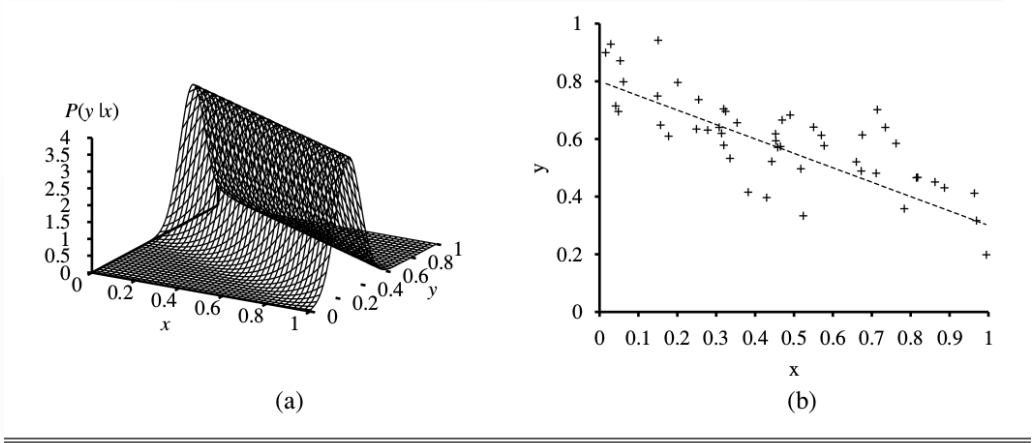


Figure 20.4 (a) A linear Gaussian model described as $y = \theta_1 x + \theta_2$ plus Gaussian noise with fixed variance. (b) A set of 50 data points generated from this model.

- Assume a parameterized linear Gaussian model with one continuous parent X and one continuous child Y.
- To learn the conditional distribution $P(Y|X)$, we maximize

$$P(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y - (\theta_1 x + \theta_2))^2}{2\sigma^2})$$

w.r.t. θ_1 and θ_2 over the data $\mathbf{d}$.

MLE for linear Gaussian models

- Constraint the derivatives of the log-likelihood to 0 and simplify. We arrive to the problem of minimizing

$$\sum_{j=1}^{N}(y_j - (\theta_1 x_j + \theta_2))^2$$

- That is, minimizing the sum of squared errors corresponds to MLE solution for a linear fit, assuming Gaussian noise of fixed variance.
- This is also known as linear regression.

[Q] Can you derive the equivalence?

Recap

- Full Bayesian learning gives best possible predictions but is intractable.
- MAP learning balances complexity with accuracy on training data.
- Maximum likelihood is equivalent to assuming a uniform prior.

Going further

- When some variables are hidden, local maximum likelihood solutions can be found using the EM algorithm or approximation methods such as variational inference.
- Learning the structure of Bayesian networks is also possible. This is an example of model selection and usually involves a discrete search in the space of structures.
- Non-parametric models represent a distribution using the collection of data points. Thus, the number of parameters grows with the training set.

Supervised learning

Supervised learning

- Assume a training set $\mathbf{d}_{\text{train}} \sim P(X, Y)$ of N example input-output pairs

$$\mathbf{d}_{\text{train}} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\},$$

  where

  - $\mathbf{x}_i$ are the input data;
  - y_i was generated by an unknown function $y_i = f(\mathbf{x}_i)$.

- From this data, we wish to learn a function h that approximates the true function f.
- Sometimes, f is stochastic, i.e., y is not strictly a function x, and we have to learn the conditional $P(Y|x)$.

Generalization

- The function h is a hypothesis.
- To measure the accuracy of a hypothesis, we evaluate its predictions on a test set $\mathbf{d}_{\text{test}} \sim P(X, Y)$ that is independent of the training set.
- A hypothesis h generalizes well if it correctly predicts the value of y for novel examples (resp., its conditional density).

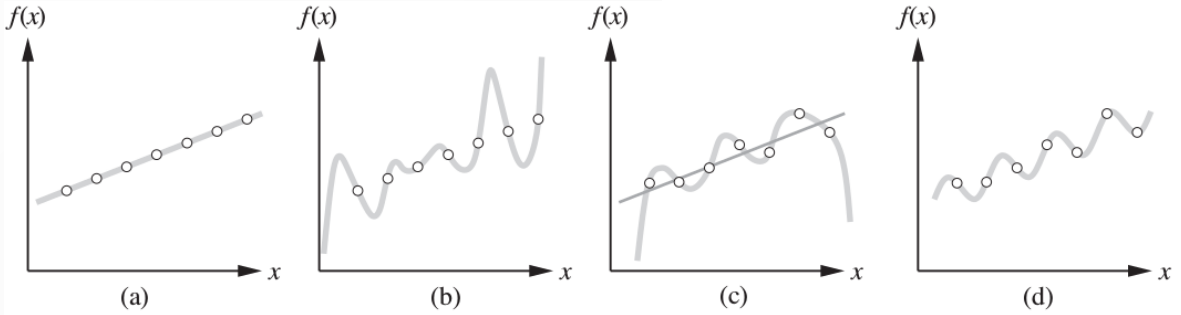


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

[Q] Which of those is best?

Best model?

- Supervised learning can be done by choosing the hypothesis h^* that is most probable on new data $\mathbf{d}_{\text{test}} \sim P(X, Y)$:

$$h^* = \arg\max_{h \in \mathcal{H}} P(h|\mathbf{d}_{\text{test}})$$

- By Bayes's rule, this is equivalent to

$$h^* = \arg\max_{h \in \mathcal{H}} P(\mathbf{d}_{\text{test}}|h)P(h)$$

  where $P(\mathbf{d}_{\text{test}}|h) = \prod_{(\mathbf{x},y) \in \mathbf{d}_{\text{test}}} P(y|x, h)$

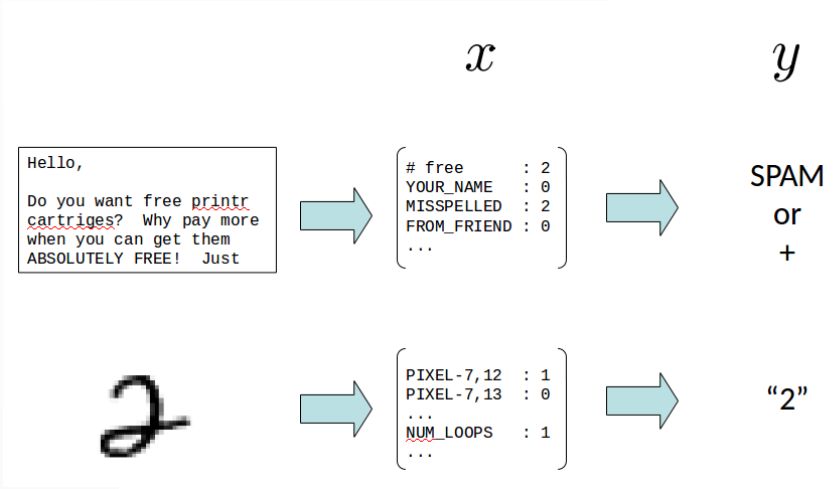
- This is the same as:
  - Maximum a posteriori estimation.
  - Maximum likelihood estimation if $P(h)$ is uniform.
- Issue: in practice, we often only have a finite test set!

What is $\mathcal{H}$?

How to find $h^* \in \mathcal{H}$?

Feature vectors

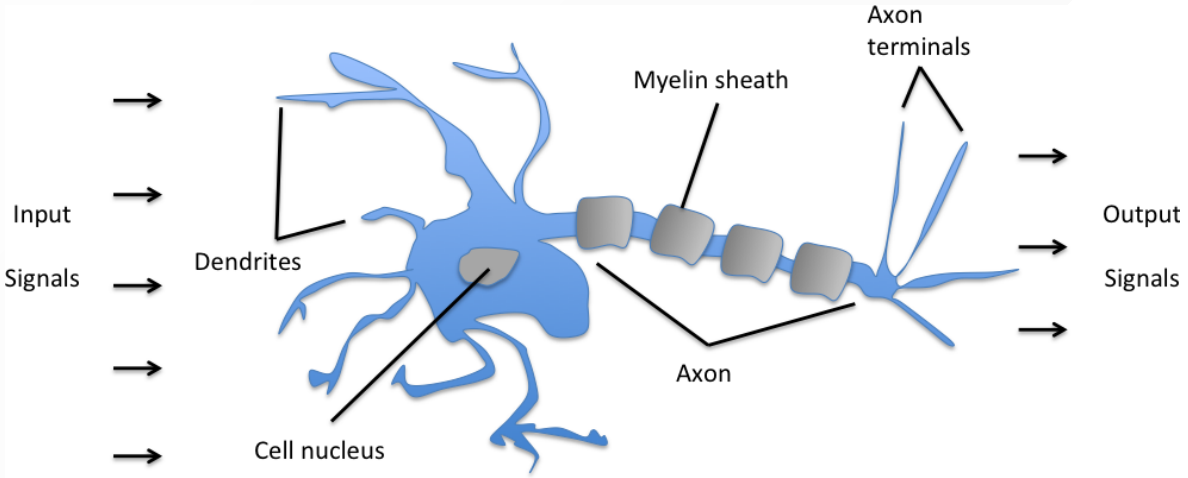
- Assume the input samples $\mathbf{x}_i \in \mathbb{R}^p$ are described as real-valued vectors of p attribute or feature values.
- If the data is not originally expressed as real-valued vectors, then it needs to be prepared and transformed to this format.



[Q] Given this data representation, what family $\mathcal{H}$ of hypothesis shall we consider?

Credits: UC Berkeley, CS188

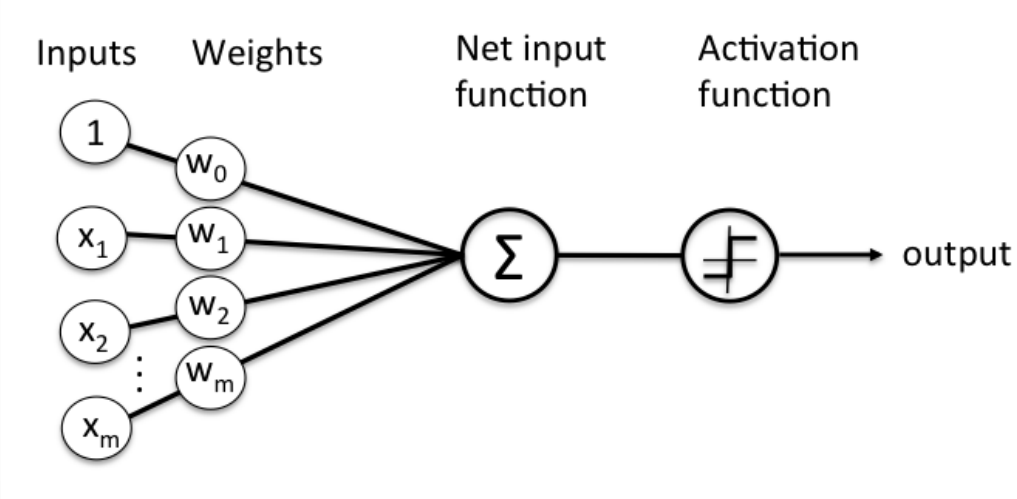
Brains (simplified)



- 10^{11} neurons of > 20 types, 10^{14} synapses.
- Information are (presumably) stored in synapses.
- Signals are noisy spike trains of electrical potential.

Credits: Sebastian Raschka, Single-Layer Neural Networks and Gradient Descent

Linear classifiers (1)



- Taking loose inspiration from neuroscience, we may consider an overly simplified model of neuron (McCulloch and Pitts, 1943).
- The linear classifier model is a squashed linear function of its inputs.

Credits: Sebastian Raschka, Single-Layer Neural Networks and Gradient Descent

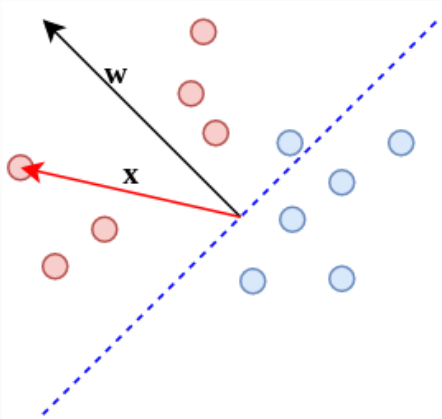
Linear classifiers (2)

- Assume Y takes binary values in $\{-1, 1\}$.
- The linear classifier model is a squashed linear function of its inputs:

$$h(\mathbf{x}; \mathbf{w}) = sign(w_0 + \sum_{j=1}^{p} w_j x_j)$$

  - Inputs are feature values x_j;
  - Each feature has a weight w_j;
  - w_0 is the intercept;
  - Feature and weight values are linearly combined as $w_0 + \sum w_j x_j$
    - Assuming a constant dummy input feature $x_0 = 1$, we write $\mathbf{w}^T \mathbf{x}$;
  - This sum is squashed through an activation function. E.g.,
    - if positive, output $+1$,
    - if negative, output -1.

Binary decision rules

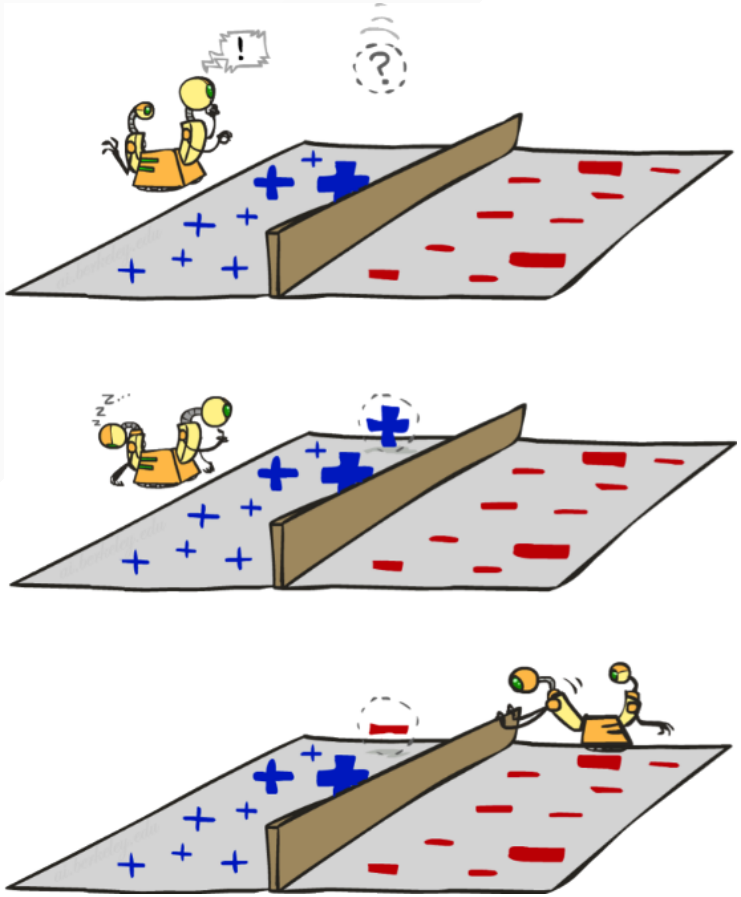


- Intuitively, predictions are computed by comparing the feature vector $\mathbf{x}$ to the weight vector $\mathbf{w}$.
- Learning boils down to figuring out a good weight vector from the training set.
- The family $\mathcal{H}$ of hypothesis is induced from the set of possible parameters values $\mathbf{w}$, that is $\mathbb{R}^{p+1}$.

[Q] $\mathcal{H}$ is huge! How do we find a good hypothesis?

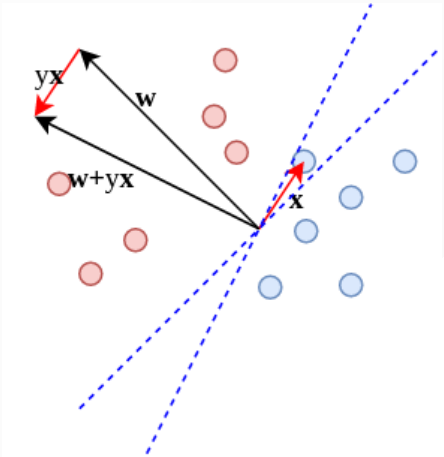
Learning: Binary perceptron

- Start with $\mathbf{w} = 0$.
- For each training example:
  - Classify with current $\mathbf{w}$.
  - If the prediction is correct, then do nothing.
  - If the prediction is incorrect, then update parameters.



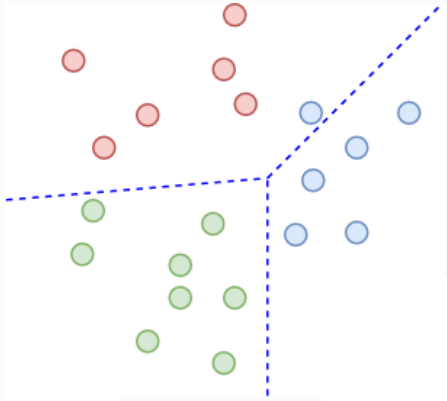
Credits: UC Berkeley, CS188

Learning: Binary perceptron



- Start with $\mathbf{w} = 0$.
- For each training example $(\mathbf{x}, y)$:
  - Classify with current weights: $\hat{y} = sign(\mathbf{w}^T \mathbf{x})$
  - If $y = \hat{y}$, do nothing.
  - Otherwise, update parameters: $\mathbf{w} = \mathbf{w} + y\mathbf{x}$

Multiclass perceptron

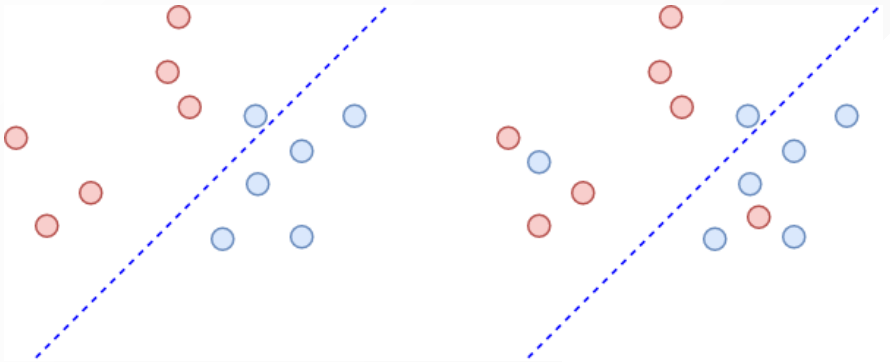


- If we have more than $C > 2$ classes, then
  - Define a weight vector $\mathbf{x}_c$ for each class c.
  - The activation for class c is $\mathbf{w}_c^T \mathbf{x}$.
- Learning:
  - Start with $\mathbf{w}_c = 0$ for all c. For each training example $(\mathbf{x}, y)$:
    - Classify with current weights: $\hat{y} = \arg\max_c \ \mathbf{w}_c^T \mathbf{x}$
    - If $y = \hat{y}$, do nothing.
    - Otherwise, update parameters:
      - $\mathbf{w}_y = \mathbf{w}_y + \mathbf{x}$ (raise score of right answer)
      - $\mathbf{w}_{\hat{y}} = \mathbf{w}_{\hat{y}} + \mathbf{x}$ (lower score of wrong answer).

Multiclass perceptron

Credits: UC Berkeley, CS188

Properties of the perceptron

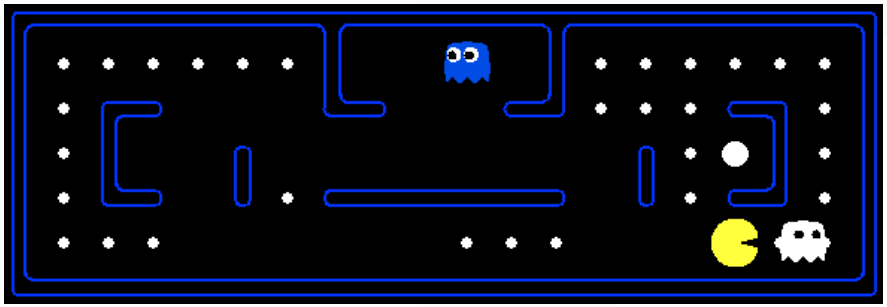


Separable vs. Non-separable

- (Novikoff, 1962): If the training set is linearly separable, the perceptron will eventually converge (binary case).
- Assume the training set is linearly separable. Let S be a sequence of labeled examples consistent with a linear threshold function $\mathbf{w}^{*T}\mathbf{x}$, where $\mathbf{w}^*$ is a unit-length vector. The number of mistakes M on S made by the Perceptron is at most $\frac{1}{\gamma^2}$, where γ is the L_2 margin of $\mathbf{w}^*$ on S:

$$\gamma = \min_{\mathbf{x} \in S} \frac{|\mathbf{w}^{*T}\mathbf{x}|}{||\mathbf{x}||}.$$

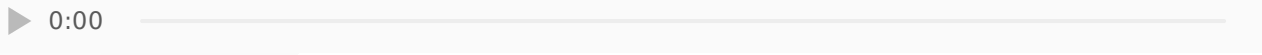
Pacman apprenticeship



- Examples are state-action pairs (s, a) that we collect by observing an expert playing.
- Features are defined over states, e.g. $g(s)$.
- We want to learn the actions that the expert would take in a given situation.
  - i.e., learn the mapping $a = f(g(s))$.
- This is a multiclass classification problem.

Credits: UC Berkeley, CS188

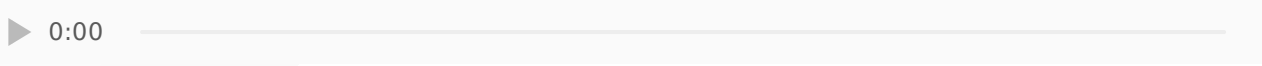
Training (1)



The Perceptron agent observes a very good Minimax-based agent for two games and updates its weight vectors as data are collected.

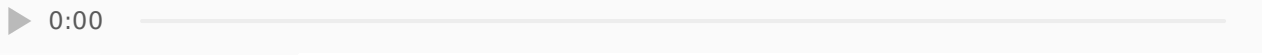
Credits: UC Berkeley, CS188

Training (2)



Credits: UC Berkeley, CS188

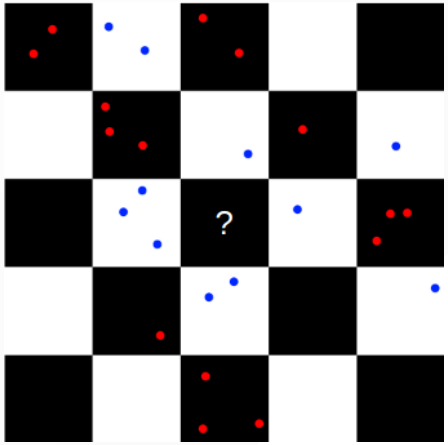
Apprentice



After two training episodes, the Perceptron agents plays. No more Minimax!

Credits: UC Berkeley, CS188

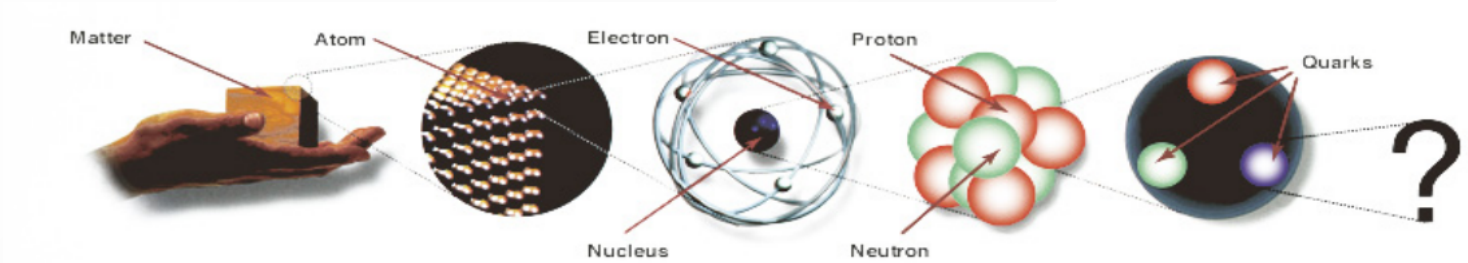
Checkerboard problem



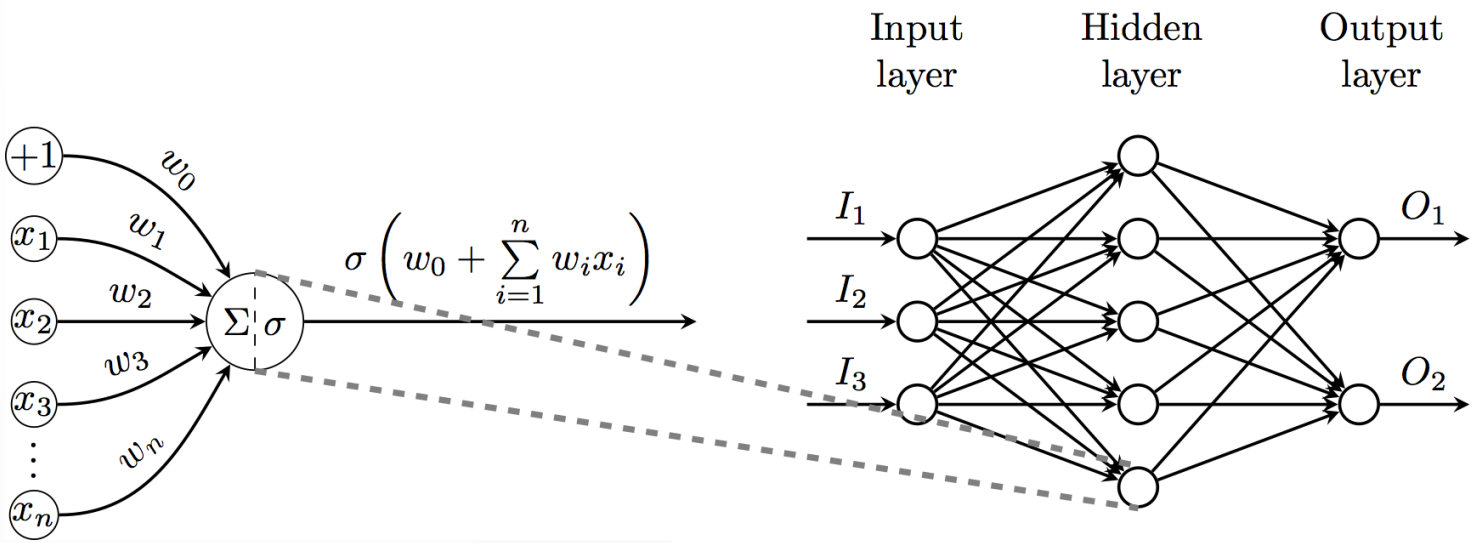
- We want to learn a binary classifier to determine the color (blue or red) of a point given its position.
- Based only on local generalization, we can correctly guess the color of a new point if it lies within the same square as a training example.
- No guarantee that the learned hypothesis correctly extends the checkerboard pattern in squares that do not contain training examples.
- In general, to distinguish $O(k)$ regions, traditional supervised learning methods require $O(k)$ examples.

Compositional assumption

Hypothesis: The data was generated by a composition of factors, potentially at multiple levels in a hierarchy.

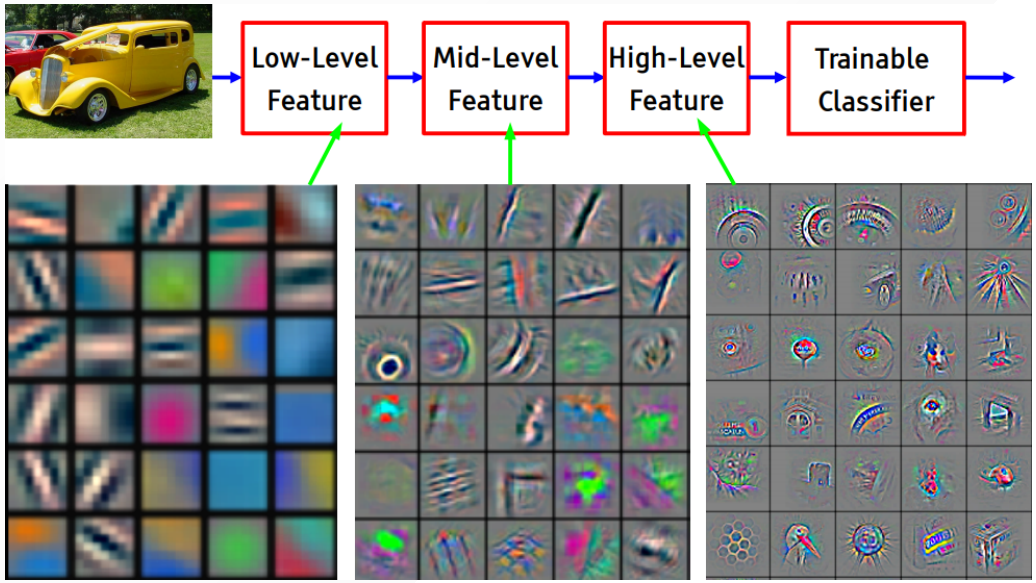


Multi-layer perceptron



- The multi-layer perceptron (MLP) is a hierarchical composition of individual perceptron-like models, called neurons. The MLP is an example of neural network.
- The network parameters are the union of the individual neuron parameters.
- Parameters are learned efficiently using stochastic gradient descent.

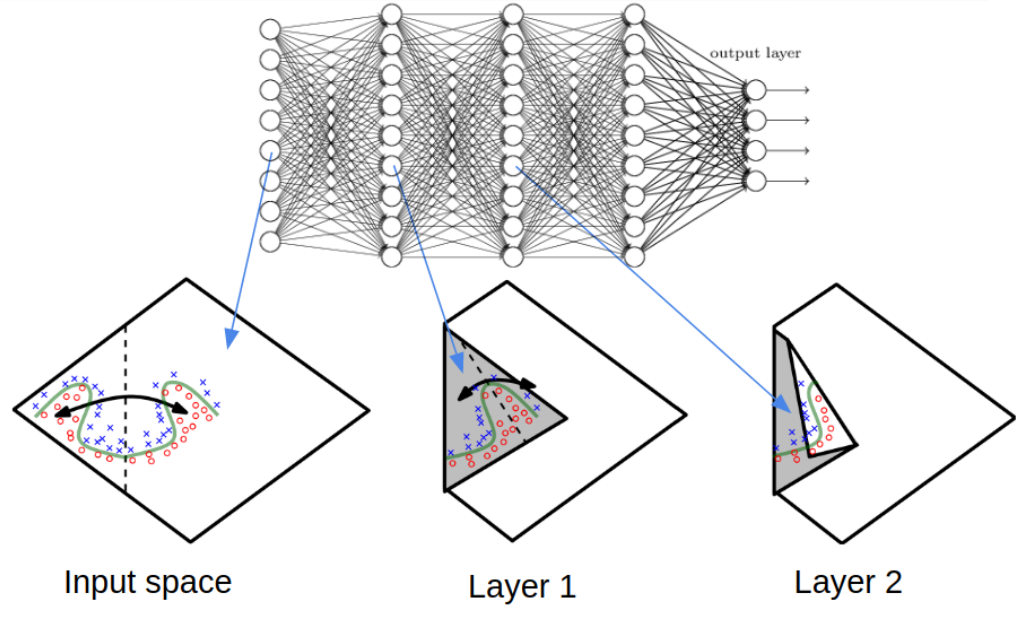
Deep Learning



- Effectively, training a deep neural network amounts to learn the parameters of a hierarchical representation of the data tailored for the target task.
- The high-level features to extract are not-preprogrammed! Concepts are learned by the network itself.

Credits: Yann Lecun (NYU), Deep Learning, 2017

Expressiveness



- The compositional assumption used in deep learning allows exponential gains in the number of required training examples.
- $O(2^k)$ regions can be defined from $O(k)$ examples, as along as one introduces dependencies between disconnected regions.

Applications

Deep learning is now at the core of many state-of-the-art systems, including:

- Image recognition
- Speech recognition
- Natural language processing
- Scientific studies
- Autonomous agents

Reinforcement learning

Reinforcement learning

- Supervised learning can be used to learn functions or probabilistic models, provided the (large) availability of training data.
- Reinforcement learning (RL) is learning what to do in the absence of labeled examples of what to do.
- Instead of labeled examples, an agent may perceive sparse feedback, called rewards, indicating whether or not the consequences of its actions were good.
- In addition, the agent does not know:
  - how the world works;
  - how rewards are computed.
- RL might be considered to encompass all of AI: an agent is placed in an unknown environment and must learn to behave successfully therein.

Unsupervised learning

Unsupervised learning

- Most of the learning performed by animals and humans is unsupervised.
  - Without labeled examples nor rewards.
- We learn how the world works by observing it:
  - We learn that the world is 3-dimensional.
  - We learn that objects can move independently of each other.
  - We learn object permanence.
  - We learn to predict what the world will look one second or one hour from now.
- We build a model of the world through predictive unsupervised learning.
  - This predictive model gives us common sense.
  - Unsupervised learning discovers regularities in the world.

Credits: Yann Lecun (NYU), Deep Learning, 2017

Common sense

- Learning a predictive model of the world gives us common sense.
- If I say: "Bernard picks up his bag and leaves the room".
- You can infer:
  - Bernard stood up, extended his arm to pick the bag, walked towards the door, opened the door, walked out.
  - He and his bag are not in the room anymore.
  - He probably did not dematerialized or flied out.

Credits: Yann Lecun (NYU), Deep Learning, 2017

How do we do that?

We have no clue!

Summary

- Learning is a key element of intelligence.
- Statistical learning aims at learning probabilistic models (their parameters or structures) automatically from data.
- Supervised learning is used to learn functions from a set of training examples.
  - Linear models are simple predictive models, effective on some tasks but usually insufficiently expressive.
  - Neural networks are composition of squashed linear models.
  - Deep learning = learning hierarchical representations.
- Reinforcement learning = learning to behave in an unknown environment from sparse rewards.
- Unsupervised learning = learning a model of the world by observing it.

Going further?

We barely scratched the surface today...

- ELEN0062: Introduction to Machine Learning
- INFO8004: Advanced Machine Learning (Spring 2018)
- INFO8003: Optimal decision making for complex problems (Spring 2018)
- INFOXXXX: Deep Learning (Spring 2019)