Introduction to Deep Learning

Introduction (1)

Prof. Songhwai Oh

ECE, SNU

INTELLIGENT SYSTEMS

What is Intelligence?

Merriam-Webster

(1) **:** the ability to **learn** or **understand** or to deal with new or trying situations **: reason**; *also* **:** the skilled use of reason *(2)* **:** the ability to apply **knowledge** to **manipulate** one's environment or to think **abstractly** as measured by objective criteria (as tests)

Wikipedia

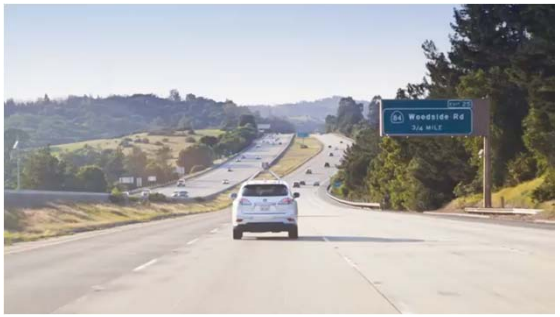
A very general mental capability that, among other things, involves the ability to **reason**, **plan**, **solve problems**, **think abstractly**, **comprehend complex ideas**, **learn quickly** and **learn from experience**.

It is not merely book learning, a narrow academic skill, or test-taking smarts. Rather, it reflects a broader and deeper capability for comprehending our surroundings—"catching on," "making sense" of things, or "figuring out" what to do. [Gottfredson, Linda S. (1997). "Mainstream Science on Intelligence (editorial)". *Intelligence* **24**: 13–23.]

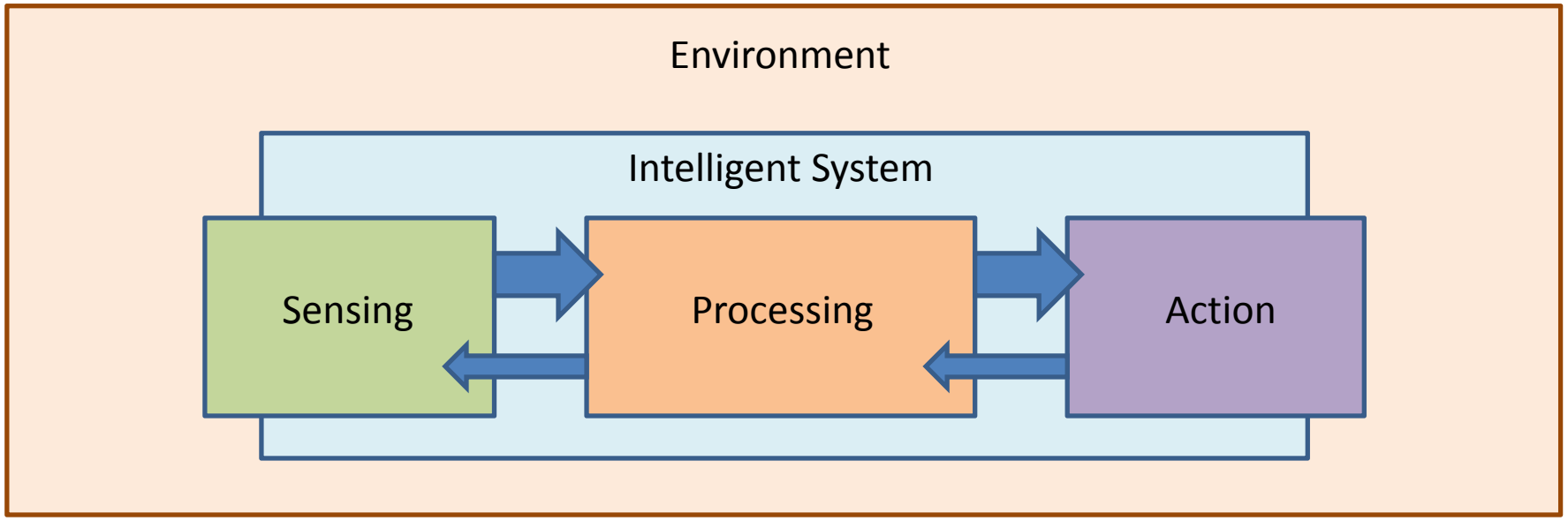
Intelligent Systems

BigDog (2005)

PR2 (UC Berkeley, 2010)

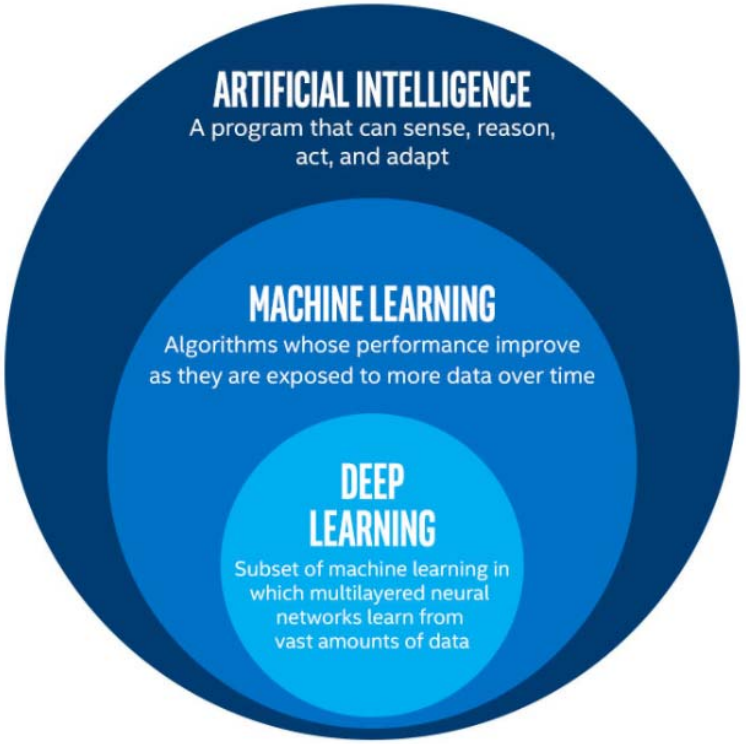


Google Car (2014)



Course Outline

	Lab
(Day 1) Introduction I – AI, machine learning, linear regression	TensorFlow, linear regression
(Day 2) Introduction II – Linear classification, neural networks – Deep learning overview	MNIST classification using multilayer perceptron (MLP)
(Day 3) Convolutional neural networks – Convolution, ReLU, pooling – Gradient descent, dropout, batch normalizations – AlexNet, VGGNet, ResNet	MNIST classification using CNNs
(Day 4) Recurrent neural networks – RNN, LSTM, GRU, Seq2Seq	CIFAR-10 image classification
(Day 5) Advanced topics – Generative adversarial networks, NestedNet – Deep reinforcement learning	Name generation using RNNs



ARTIFICIAL INTELLIGENCE (AI)

Artificial Intelligence (AI)

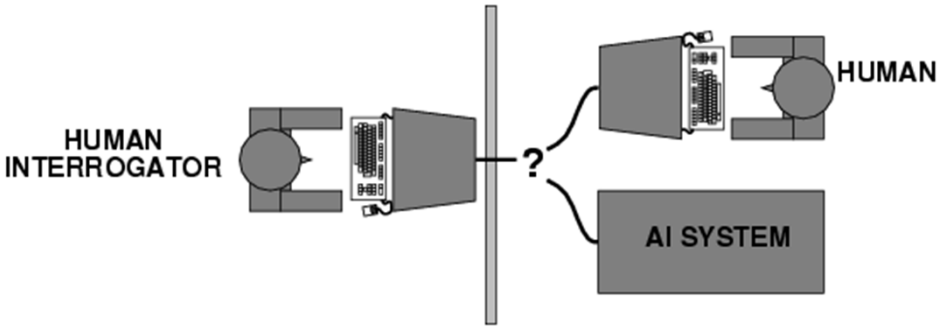
- Closely related field
- Heavily focused on (e.g., traditional AI)
  - Processing
  - Discrete world
  - Logic
- Four popular views about AI

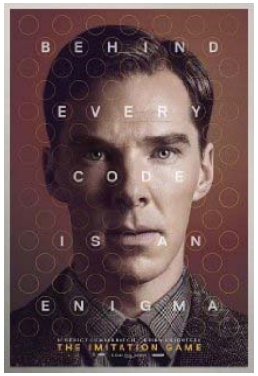
Thinking Humanly	Thinking Rationally
Acting Humanly	Acting Rationally

Acting Humanly: Turing Test

Turing (1950) "Computing machinery and intelligence":

- **"Can machines think?" → "Can machines behave intelligently?"**
- Operational test for intelligent behavior: the Imitation Game





The Imitation Game (2014)

- Predicted that by 2000, a machine might have a 30% chance of fooling a lay person for 5 minutes
  - Eugene Goostman (chatterbot) passed the Turing test on June 7, 2014 (10 out of 30 judges). http://www.princetonai.com/
- Anticipated all major arguments against AI in following 50 years
  - Chinese room argument (Searle)
- Suggested major components of AI: knowledge, reasoning, language understanding, learning

Problem: Turing test is not **reproducible**, **constructive**, or amenable to **mathematical analysis**.

Thinking Humanly: Cognitive Science

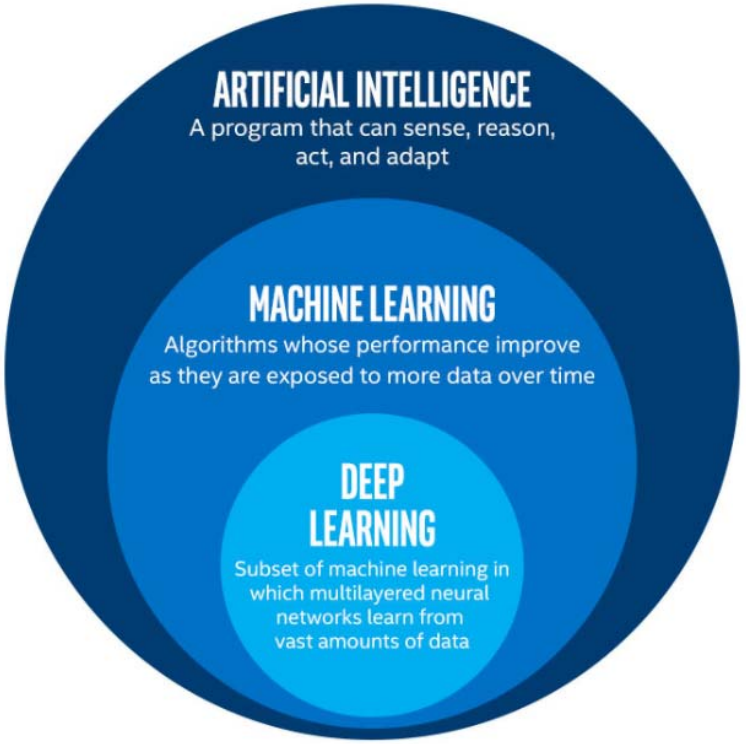
- 1960s "cognitive revolution": information-processing psychology
- Requires scientific theories of internal activities of the brain
  - What level of abstraction? Knowledge or circuits?
  - How to validate? Requires
    - Predicting and testing behavior of human subjects (top-down)
    - Direct identification from neurological data (bottom-up)
- Both approaches (roughly, Cognitive Science and Cognitive Neuroscience) are now distinct from AI

Thinking Rationally: Laws of Thought

- **Normative (or prescriptive) rather than descriptive**
- Aristotle: what are correct arguments/thought processes?
- Several Greek schools developed various forms of **logic: notation** and **rules of derivation** for thoughts; may or may not have proceeded to the idea of mechanization
- Direct line through mathematics and philosophy to modern AI
- Problems:
  - Informal or uncertain knowledge cannot be precisely stated formally
  - Big difference between solving a problem “in principle” and solving it in practice

Acting Rationally: Rational Agent

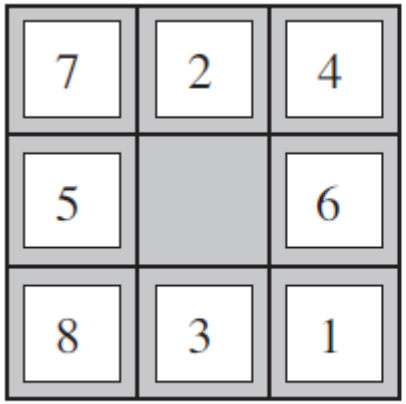
- **Rational behavior: doing the right thing**
- The right thing: that which is expected to maximize goal achievement, given the available information
- Doesn't necessarily involve thinking – e.g., blinking reflex – but thinking should be in the service of rational action
- Computational limitations make perfect rationality unachievable
  - Design the best program for given machine resources



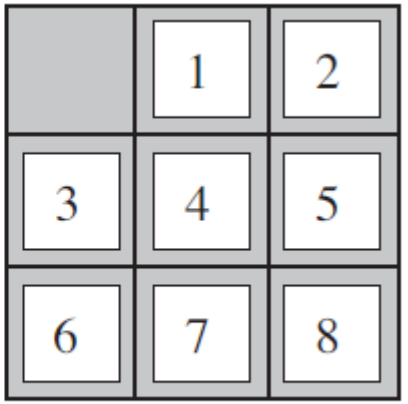
TRADITIONAL AI

Search

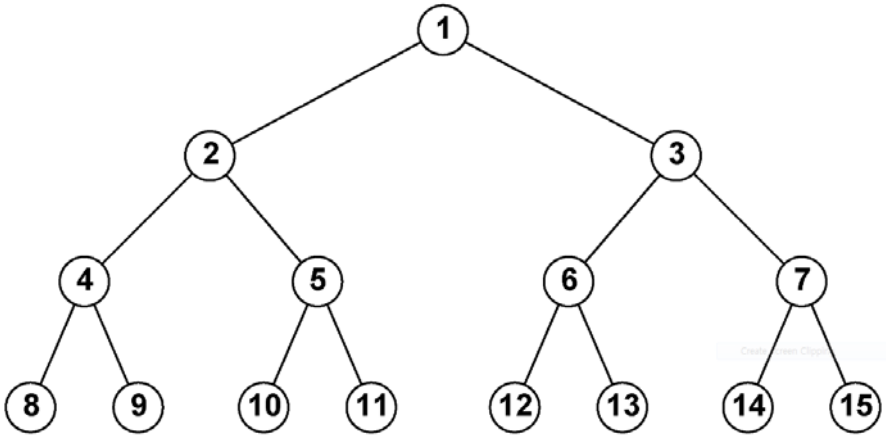
E.g., 8-Puzzle



Start State



Goal State



Search Algorithms:

- Breadth-first search
- Depth-first search
- A* search
- and many more.

Game

IBM Deep Blue vs. World Chess Champion Garry Kasparov, 1997

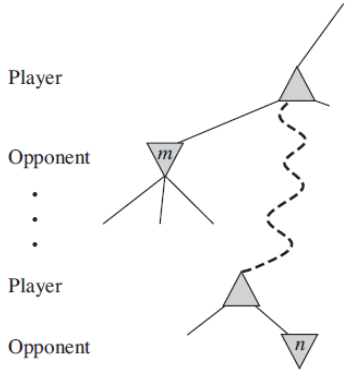
First match

- February 10, 1996: takes place in Philadelphia, Pennsylvania
- Result: **Kasparov**–Deep Blue (4–2)
- Record set: First computer program to defeat a world champion in a *game* under tournament regulations

Second match (rematch)

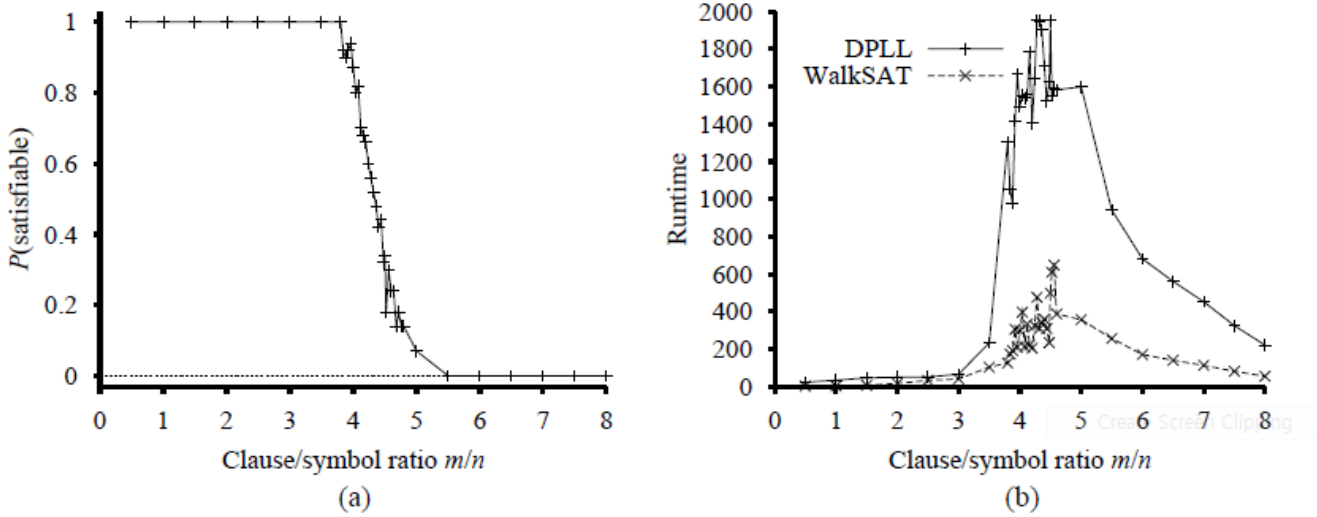
- May 11, 1997: held in New York City, New York
- Result: **Deep Blue**–Kasparov (3½–2½)
- Record set: First computer program to defeat a world champion in a *match* under tournament regulations

Alpha-Beta Pruning



Propositional Logic

- **Operators**: $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies), $\Leftrightarrow$ (if and only if)
- Propositional theorem proving
- **SAT problem**: The problem of determining the satisfiability of sentences in propositional logic. NP-complete.
- **Conjuntive normal form (CNF)**: A conjunction of clauses and each clause is a disjunction of literals. E.g., $(\neg B_1 \vee P_1 \vee P_2) \wedge (\neg P_1 \vee B_1) \wedge (\neg P_2 \vee B_1)$. 3-CNF is NP-complete.



First-Order Logic

- **Elements**: objects (constant symbols), relations (predicate symbols), functions (function symbols); E.g. $\neg Brother(LeftLeg(Richard), John)$
- **Quantifiers**: $\forall$ (for all, universial quantification), $\exists$ (there exits, existential quantification)

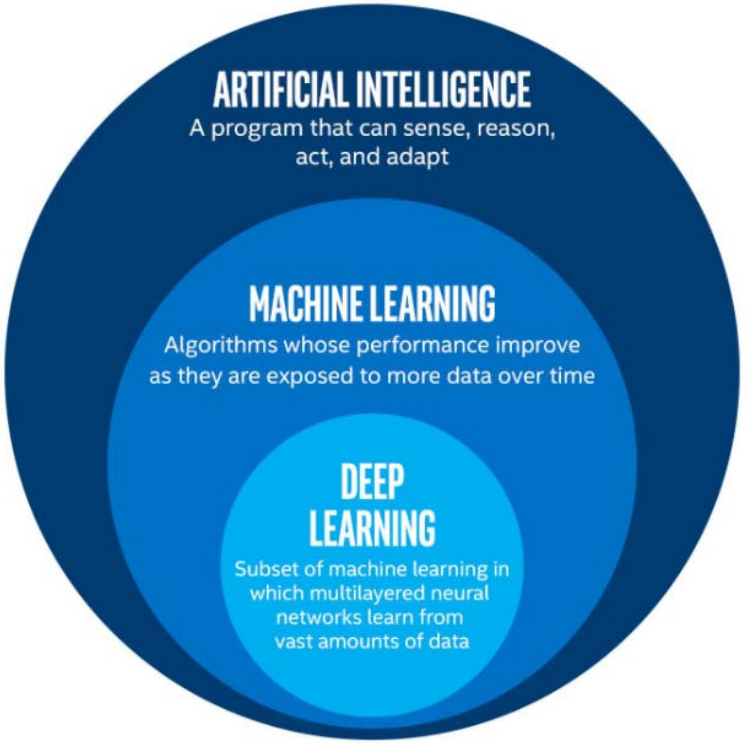
$$\begin{aligned}
&\forall x \quad King(x) \Rightarrow Person(x) \\
&\exists x \quad Crown(x) \wedge OnHead(x, John) \\
&\forall x \exists y \quad Loves(x, y) \\
&\exists y \forall x \quad Loves(x, y)
\end{aligned}$$

- **Inference algorithms**: forward chaining (deductive database, production system), backward chaining (logic programming, e.g., Prolog), resolution-based theorem proving
- **Gödel's incompleteness theorem**: There are true statements in mathematics that cannot be proven. E.g., Turing's undecidability theorem.

Planning

$$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A)$$
$$\wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$$
$$Goal(On(A, B) \wedge On(B, C))$$
$$Action(Move(b, x, y),$$
$$\text{Precond: } On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y)$$
$$\wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y),$$
$$\text{Effect: } On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$$
$$Action(MoveToTable(b, x),$$
$$\text{Precond: } On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$$
$$\text{Effect: } On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$$

Start State Goal State



MACHINE LEARNING

Learning

- **Learning**: to improve agent's performance on future tasks after making observations about the world.
- Components to be learned:
  1. A direct mapping from conditions on the current state to actions.
  2. A means to infer relevant properties of the world from the percept sequence.
  3. Information about the way the world evolves and about the results of possible actions the agent can take.
  4. Utility information indicating the desirability of world states.
  5. Action-value information indicating the desirability of actions.
  6. Goals that describe classes of states whose achievement maximizes the agents utility.
- Representation and prior knowledge: e.g., first-order logic, Bayesian networks; inductive learning vs. deductive learning.

Types of Learning

- Categorized by types of feedback available for learning.
- **Unsupervised learning**: learns patterns in the input even though no explicit feedback is supplied. E.g., clustering, density estimation, anomaly detection.
- **Reinforcement learning**: learns from a series of reinforcements – rewards or punishments.
- **Supervised learning**: observes some example input-output pairs and learns a function that maps from input to output.
- **Semi-supervised learning**: given a few labeled examples and must make what we can of a large collection of unlabeled examples.

Supervised Learning

- Categorized by types of feedback available for learning.
- **Supervised learning problem**: Given a **training set** of N example input-output pairs

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N),$$

where each y_j was generated by an unknown function $y = f(x)$, discover a function $h \in \mathcal{H}$ that approximates the true function f.

- Learning is a search through the space of possible hypotheses ($\mathcal{H}$, hypothesis space) for one that will perform well, even on new examples beyond the training set.
- A hypothesis **generalizes** well if it correctly predicts the value of y for new examples (**test set**).
- **Classification** (if $y \in \mathcal{Y}$ and $|\mathcal{Y}|$ is finite); **Regression** (if y is continuous).
- Supervised learning can be posed as a problem of finding h^* such that

$$h^* = \arg\max_{h \in \mathcal{H}} P(h|data) = \arg\max_{h \in \mathcal{H}} P(data|h)P(h)$$

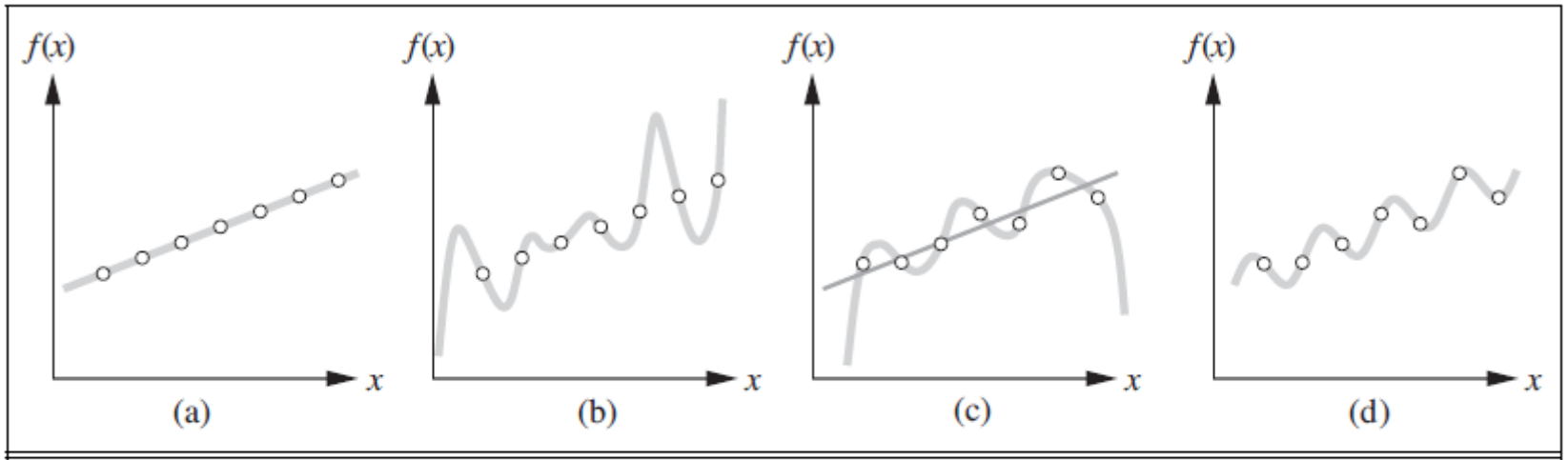
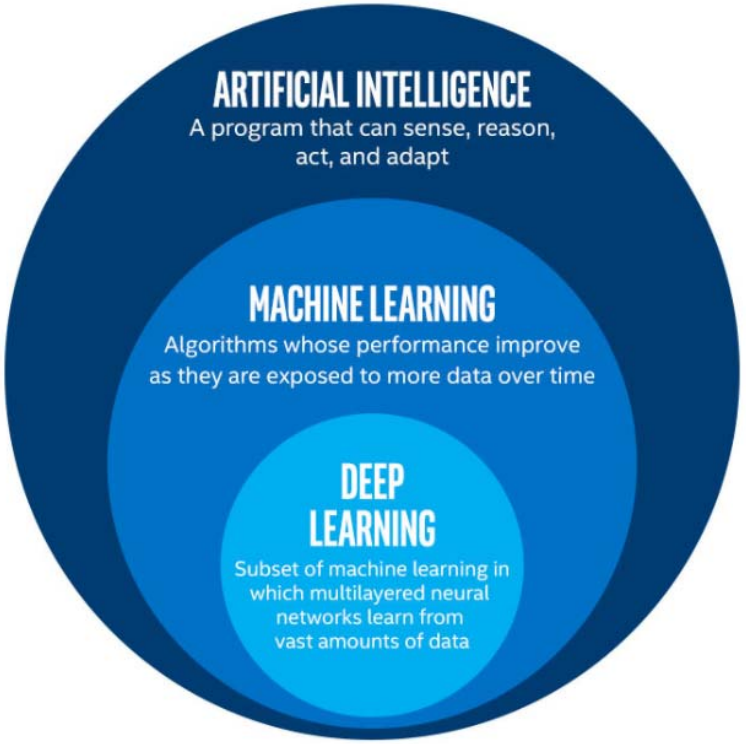


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data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

- **Ockhams razor**: prefer the simplest hypothesis consistent with the data over more complex hypotheses.
- **Expressiveness-complexity tradeoff**: tradeoff between the expressiveness of a hypothesis space and the complexity of finding a good hypothesis within that space.



GENERALIZATION ERROR

Overfitting and Other Considerations

- **Overfitting**: When a complex model is used for a small number of data, it may fit the training set well but have a large generalization error on future data.
- Decision tree pruning: A method to prevent overfitting in a decision tree. For each node having leaf nodes as descendants, perform a significance test. If a null hypothesis is accepted, prune away the node.
- Other considerations:
  - Missing data. Solution: Weighting all possible paths.
  - Multivalued attributes: When an attribute has many possible values, the information gain measure gives an inappropriate indication of the attributes usefulness. Solution: Gain ratio = (information gain) / (intrinsic value).
  - Continuous and integer-valued input attributes: Continuous or integer-valued attributes have an infinite set of possible values. Solution: Finding a split point with the highest information gain.
  - Continuous-valued output attributes. Solution: Regression tree.

Loss Function

- **Loss function**: $L(x, y, \hat{y})$ is defined as the amount of utility lost by predicting $h(x) = \hat{y}$ when the correct answer is $f(x) = y$.

Absolute value loss:	$L_1(y, \hat{y}) = \lvert y - \hat{y} \rvert$
Squared error loss:	$L_2(y, \hat{y}) = (y - \hat{y})^2$
0/1 loss:	$L_{0/1}(y, \hat{y}) = 0$ if $y = \hat{y}$, else 1

- Suppose each example (x, y) is independent and identically distributed (i.i.d.) from $P(X, Y)$.
- **Generalization loss** for a hypothesis h with respect to loss function L is

$$GenLoss_L(h) = \sum_{(x,y)\in\mathcal{E}} L(y, h(x))P(x, y),$$

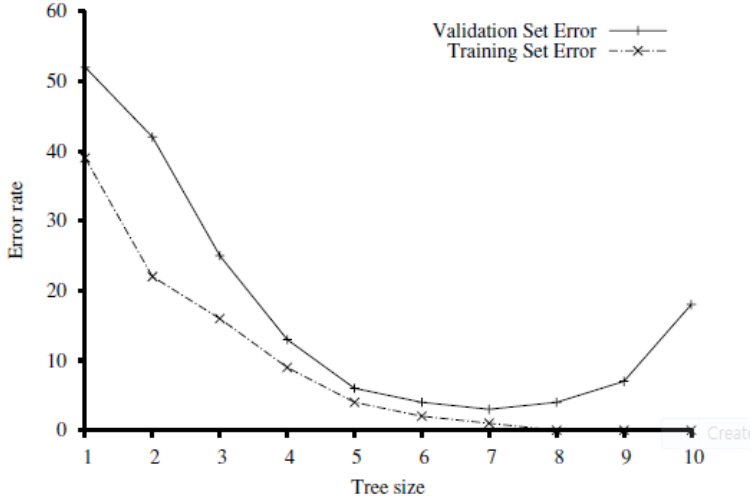
  where $\mathcal{E}$ is the set of all input-output pairs.
- Our goal is to find $h^* = \arg\min_{h\in\mathcal{H}} GenLoss_L(h)$.
- **Empirical loss**: $EmpLoss_{L,E}(h) = \frac{1}{N}\sum_{(x,y)\in E} L(y, h(x))$, where E is the training set, estimates generalization loss.
- The estimated best hypothesis may differ from the true function due to: unrealizability, variance, noise, and computational complexity.

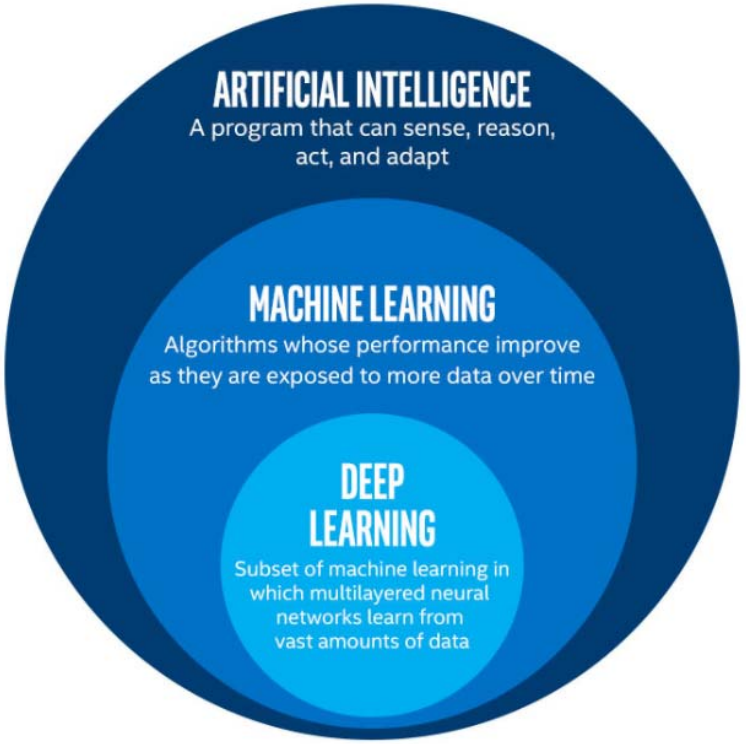
Cross Validation, Model Selection

- **k-fold cross validation**: to measure how well a hypothesis generalizes for the future data.

  1. Split the data into k equal subsets.
  2. Perform k rounds of learning; on each round $1/k$ of the data is held out as a test set and the remaining examples are used as training data.
  3. Compute the average test set score of the k rounds.

- **Leave-one-out cross-validation** when $k = n$.
- **Model selection**: the problem of finding an optimal model class (hypothesis space) in terms of the expressiveness-complexity tradeoff.





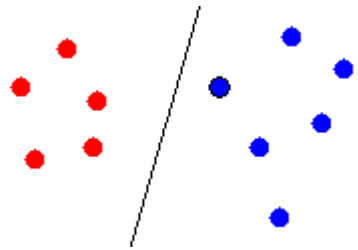
Classification

Regression

Density Estimation

Three Major Machine Learning Problems

Classification



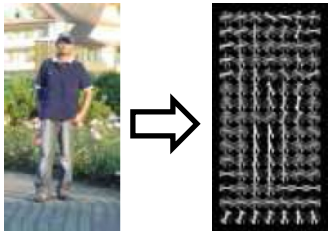
Data: $\{(x_i, y_i): i=1,\ldots,N\}$

x_i: input feature; y_i: class label

Given a new input feature, classify its label.

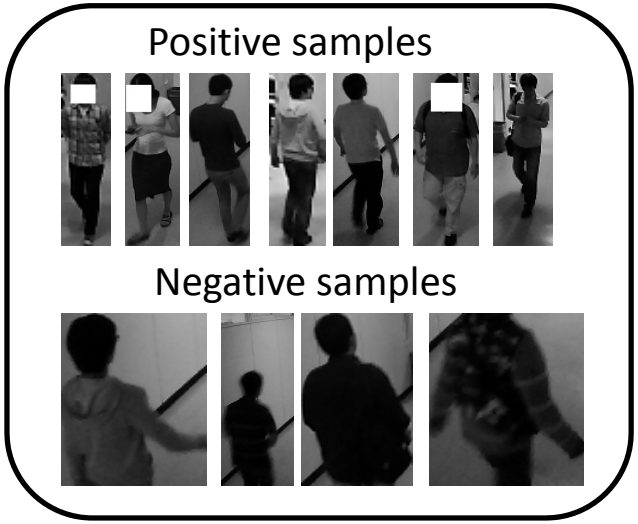
Example: Human detection

Feature Extraction



HOG

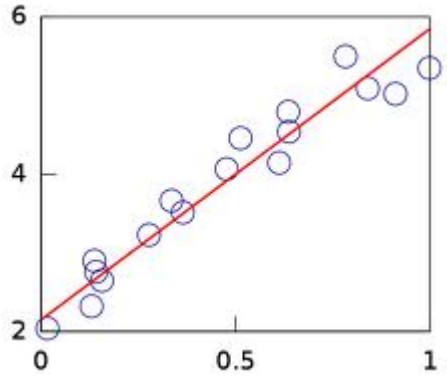
Histogram of Oriented Gradients



Training set

Support Vector Machine (SVM) Classifier

Regression

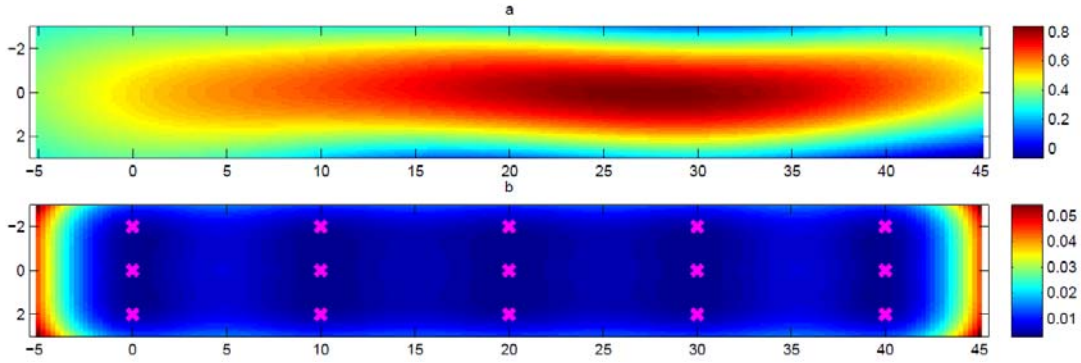


Data: $\{(x_i, y_i): i=1,\ldots,N\}$

x_i: input; y_i: output

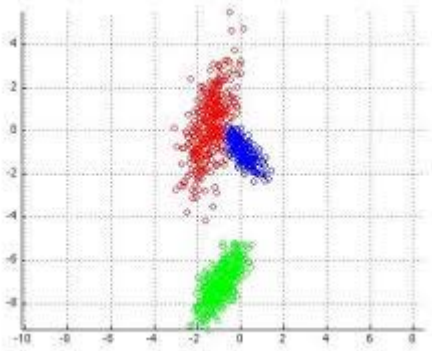
Given data, find a function f such that $f(x_i) \approx y_i$.

Example: Chemical concentration mapping



Chemical concentration prediction by Gaussian process regression

Density Estimation



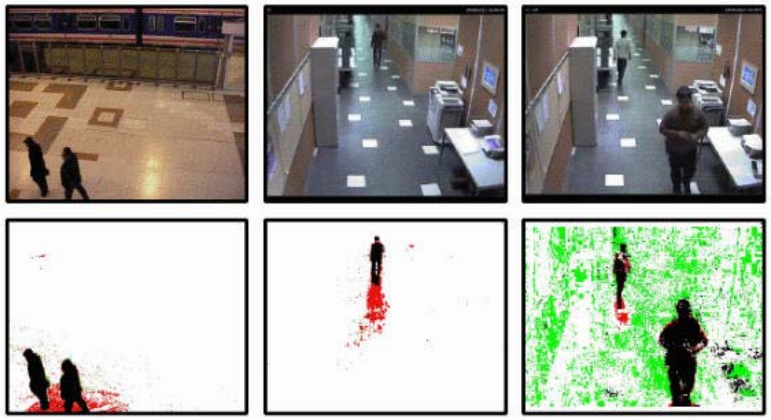
Data: $\{x_i: i=1,\ldots,N\}$

x_i: feature

Given data, find the distribution (or a simpler description) of x_i.

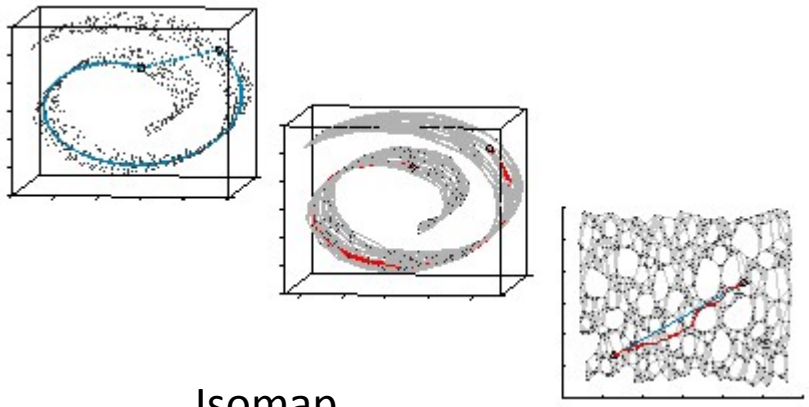
Unsupervised learning

Example: Background subtraction,

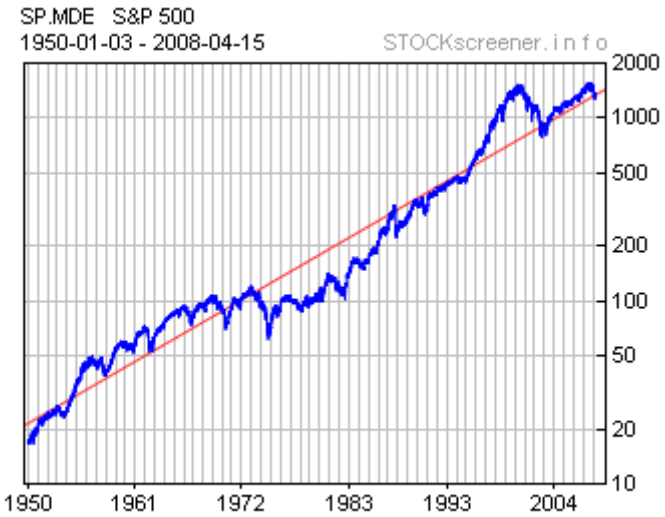


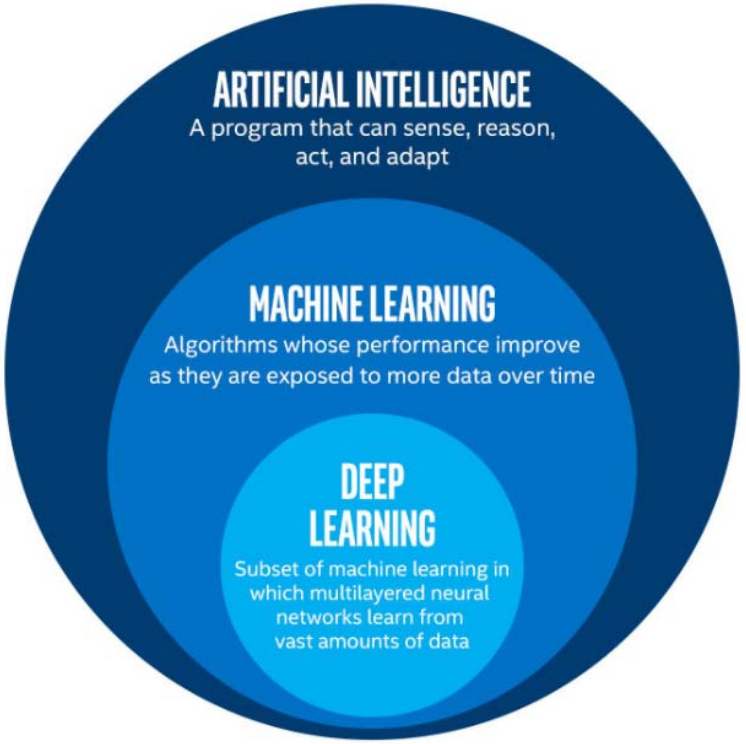
Background modeling using a mixture of Gaussians

Dimensionality reduction



Isomap,
Locally linear embedding (LLE)





LINEAR REGRESSION

Univariate Linear Regression

- Univariate linear function: $y = w_1 x + w_0$, where $w_0, w_1 \in \mathbb{R}$.
- Let $\mathbf{w} = [w_0, w_1]^T$ and define

$$h_{\mathbf{w}}(x) = w_1 x + w_0.$$

- **Linear regression**: Given a training set $\{(x_i, y_i) : 1 \leq i \leq N\}$, find $h_{\mathbf{w}}$ that best fits the training set.

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2.$$

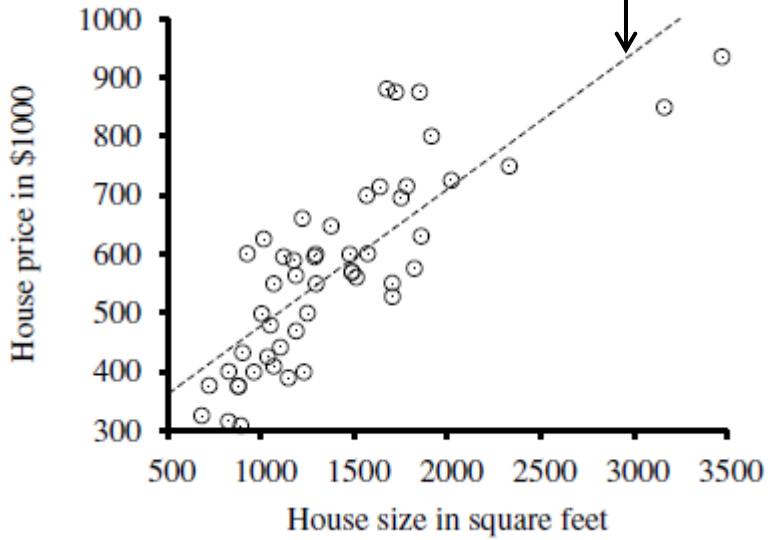
- Our goal is to find $\mathbf{w}^* = \arg\min_{\mathbf{w}} Loss(h_{\mathbf{w}})$.

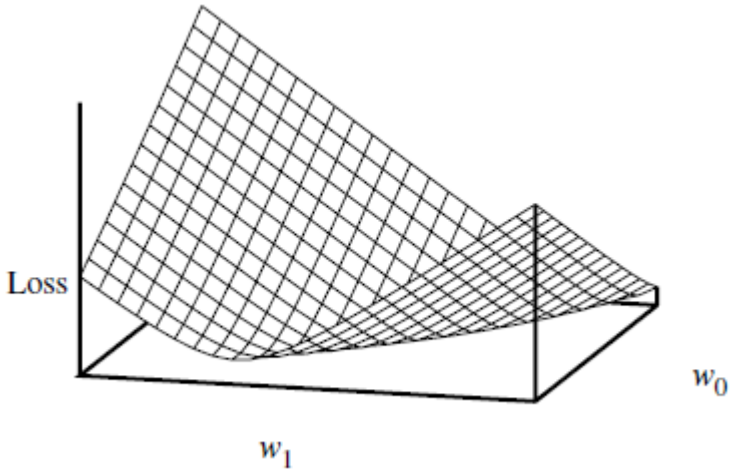
$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N \ .$$

Example

$$y = 0.232x + 246.$$





$$\mathbf{w}^* = [w_0^*, w_1^*]^T = [246, 0.232]$$

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2.$$

Gradient Descent

$\mathbf{w} \leftarrow$ any point in the parameter space

loop until convergence **do**

  for each w_i **in** w **do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

α: step size or learning rate.

$$\begin{aligned}
\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i} (y - h_{\mathbf{w}}(x))^2 \\
&= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i} (y - h_{\mathbf{w}}(x)) \\
&= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i} (y - (w_1 x + w_0))
\end{aligned}$$

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x))$$

$$\frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x$$

$$w_0 \leftarrow w_0 + \alpha\,(y - h_{\mathbf{w}}(x))\,; \quad w_1 \leftarrow w_1 + \alpha\,(y - h_{\mathbf{w}}(x)) \times x$$

Batch gradient descent: (Steepest descent)

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))\,; \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

Stochastic gradient descent: processes one data point at a time

Multivariate Linear Regression

- When $\mathbf{x}_j \in \mathbb{R}^n$, the hypothesis space for linear regression is spanned by functions of the following form.

$$h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + \sum_{i=1}^{n} w_i x_{j,i}.$$

- Now, redefine $\mathbf{x}_j = [1, x_{j,1}, x_{j,2}, \ldots, x_{j,n}]^T \in \mathbb{R}^{n+1}$. Then

$$h_{\mathbf{w}}(\mathbf{x}_j) = \sum_{i=0}^{n} w_i x_{j,i} = \mathbf{w}^T \mathbf{x}_j.$$

- Solution to linear regression:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j L_2(y_j, \mathbf{w}^T \mathbf{x}_j).$$

- Gradient descent update equation: $w_i = w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))$.
- Closed form solution: $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

Closed-Form Solution to Linear Regression

- Let $\mathbf{y} = [y_1, \ldots, y_N]^T$ and $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$ be the data matrix (or design matrix).

  Then

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} (y_j - \mathbf{w}^T \mathbf{x}_j)^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}).$$

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} Loss(h_{\mathbf{w}}) &= \frac{\partial}{\partial \mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \frac{\partial}{\partial \mathbf{w}} (\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}) \\ &= -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w} \end{aligned}$$

- Setting it to zero, we have $\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$, which is known as a **normal equation**. If $\mathbf{X}^T\mathbf{X}$ is invertible, we have

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

General Linear Model

$$y = \sum_{i=1}^{M} w_i h_i(x)$$

$h_i[n]$: nonlinear function of n

$$h_i(x) = x^{i-1}$$

$$h_i(x) = \exp\left(-\frac{1}{2s^2}(x - \mu_i)^2\right)$$

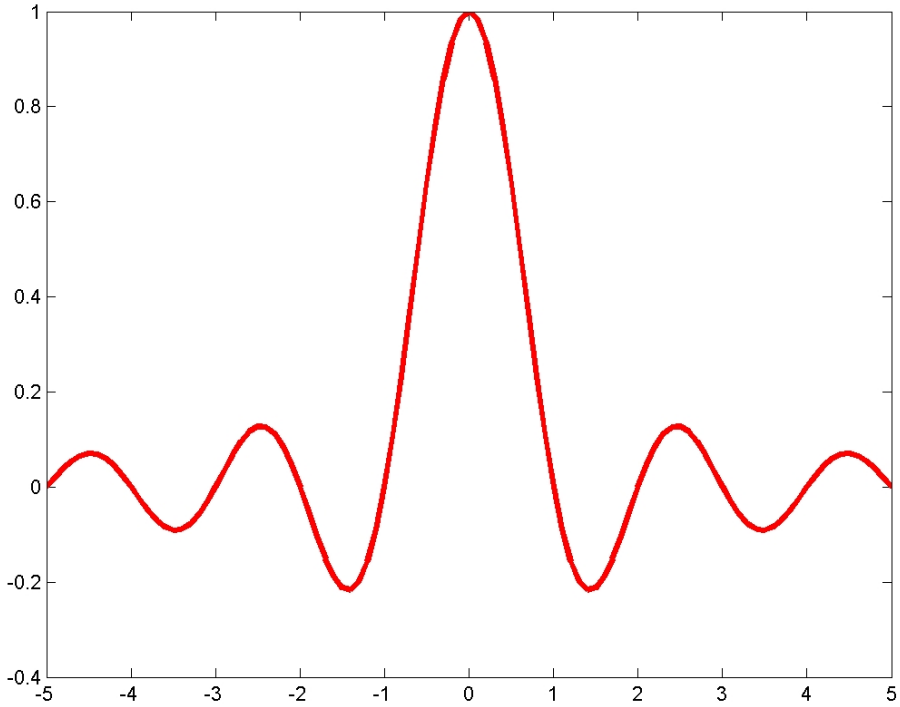
s and μ_i are fixed parameters.

$$\mathbf{y} = \mathbf{H}\mathbf{w}$$

$$\mathbf{H} = \begin{bmatrix} h_1(x_0) & h_2(x_0) & \cdots & h_M(x_0) \\ h_1(x_1) & h_2(x_1) & \cdots & h_M(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_{N-1}) & h_2(x_{N-1}) & \cdots & h_M(x_{N-1}) \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}$$

$$\mathbf{w}^* = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

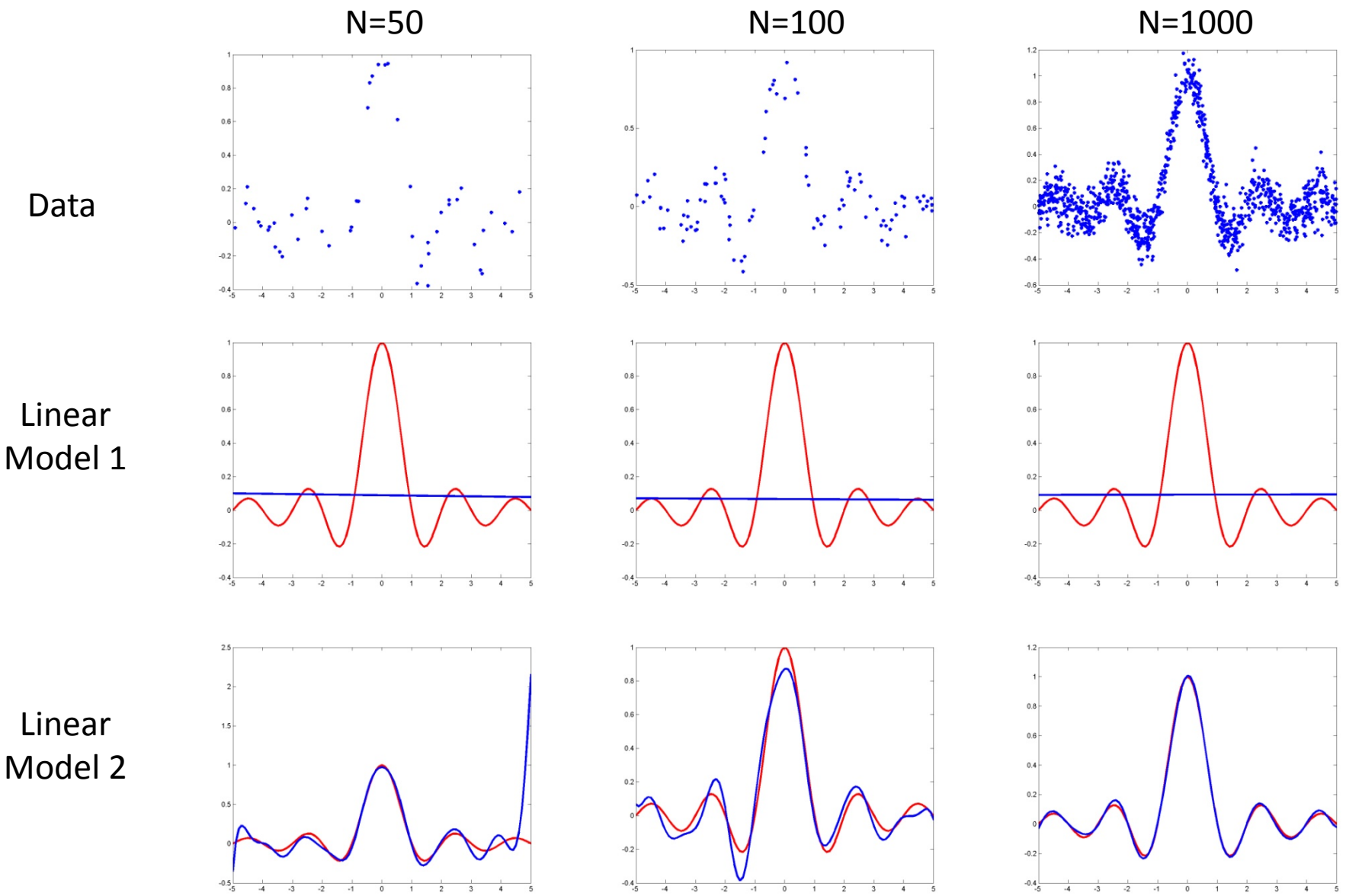
Example: Linear modeling of the SINC function



Model 1: $y = w_0 + w_1 x$

Model 2: $y = \sum_{i=1}^{M} w_i h_i(x)$ $\qquad h_i(x) = \exp\left(-\frac{1}{2s^2}(x - \mu_i)^2\right)$

Example: Linear modeling of the SINC function



Regularization

- To avoid overfitting, we may use regularization

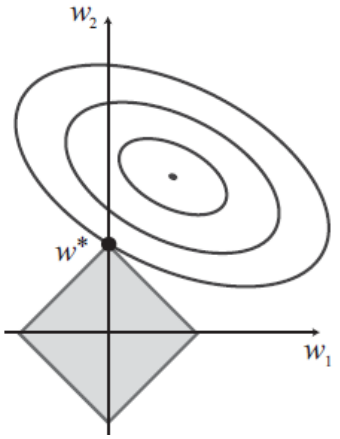
$$Cost(h) = EmpLoss(h) + \lambda Complexity(h),$$

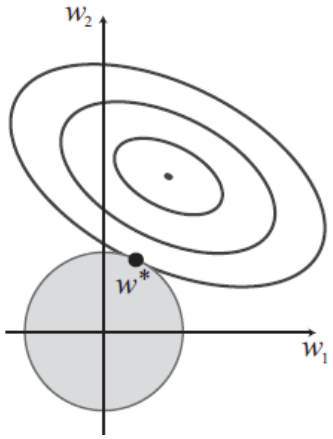
where we can consider

$$Complexity(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q.$$

- L_1 regularization if $q = 1$; L_2 regularization if $q = 2$.
- L_1 regularization tends to produce a sparse model, i.e., it sets many weights to zero. A sparse model is less likely to overfit.

L_1 Regularization





L_2 Regularization

AI

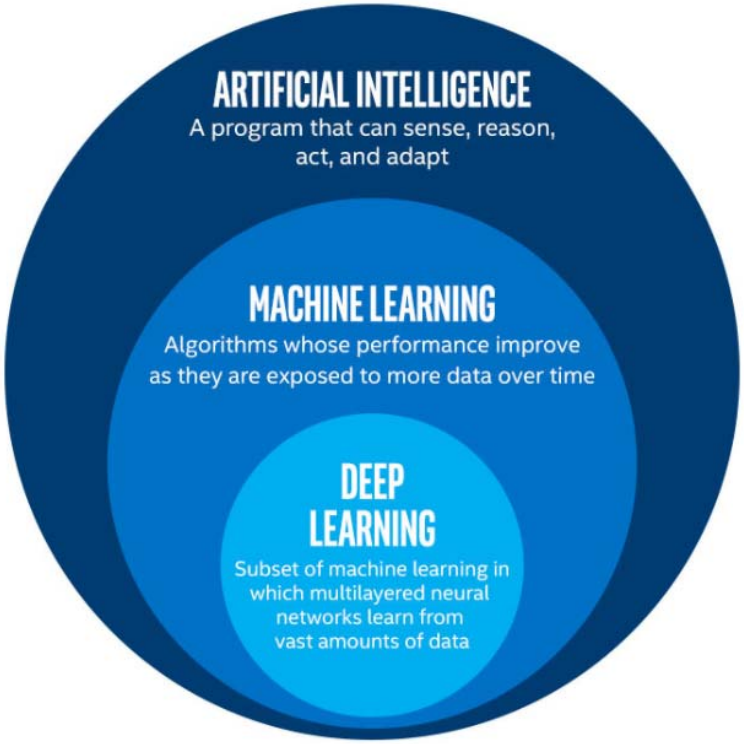
Machine Learning

Generalization Error

Classification

Regression

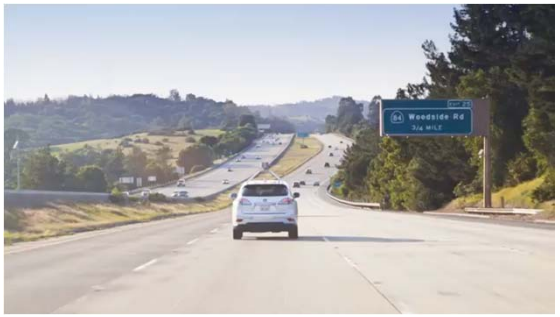
Wrap Up



Intelligent Systems

BigDog (2005)

PR2 (UC Berkeley, 2010)



Google Car (2014)