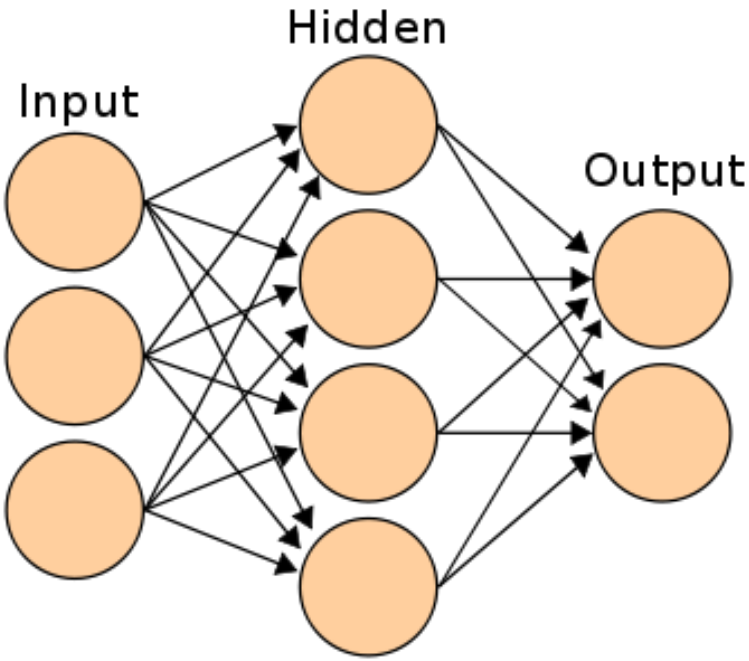
Neural networks

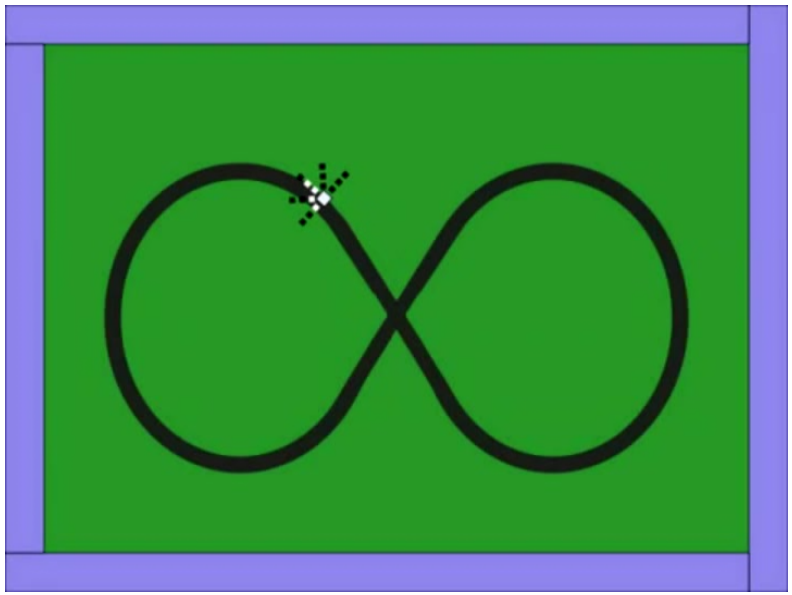
János Török

May 14, 2020

Neural networks



- Machine learning
- Pattern recognition
- Handwriting
- Speech recognition



- Input pattern
- Output pattern
- Adaptive weights
- Approximating non-linear functions

Neural networks

- Input vector I
- Output vector $O(I)$
- Transition matrix $W_{ij} \in [-1, 1]$
- Learning using a cost function
- Test goodness

Neural networks: Learning

- Supervised learning
  - Data training
  - Fitness function, energy:

$$E = T(I) - O(I),$$

    where $T(I)$ is the target vector for input I
  - Minimize E for available set of $\{I, I(O)\}$ pairs
  - Deep learning: many layers of neurons in the neural network
- Test goodness:
  - Use only part of $\{I, I(O)\}$ pairs for learning, the rest is for testing.
- Used for: pattern recognition, classification, etc.

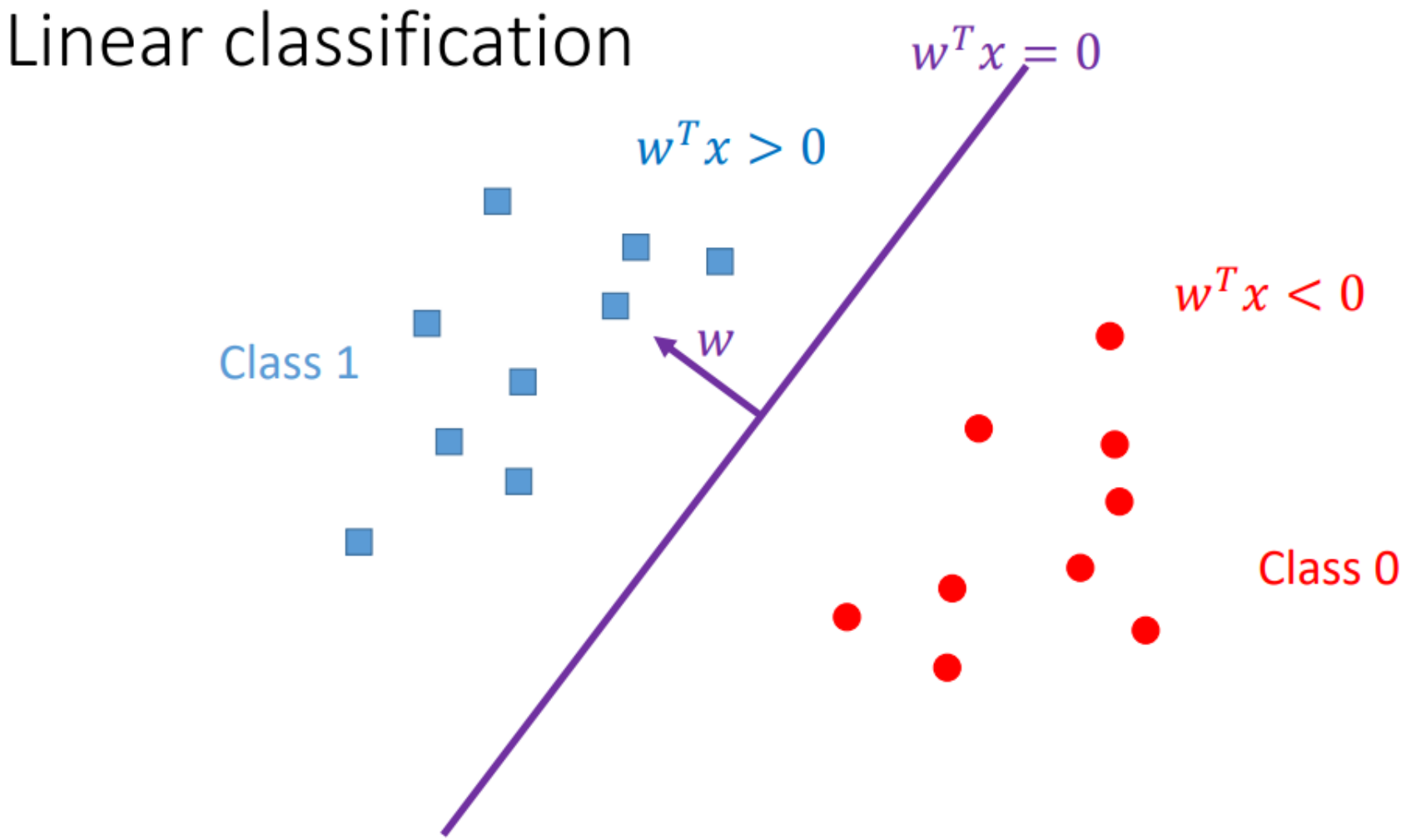
Neural networks: Learning

- Unsupervised learning: No fitness function
  - Reinforcement learning: e.g. Q-learning
    - Penalize wrong answers and reward good ones
    - Used for playing games
  - Random forest (pool of decision trees):
    - Generate a random synthetic data
    - Teach the decision tree to recognize real data (e.g. label them differently)
    - Data points closer in the decision tree are related
    - Cluster the data accordingly

Deep learning

- Literature: Introduction to deep learning: `https://www.cs.princeton.edu/courses/archive/spring16/cos495/`

Deep learning: Classification, linear



Deep learning: Classification, linear

Attempt

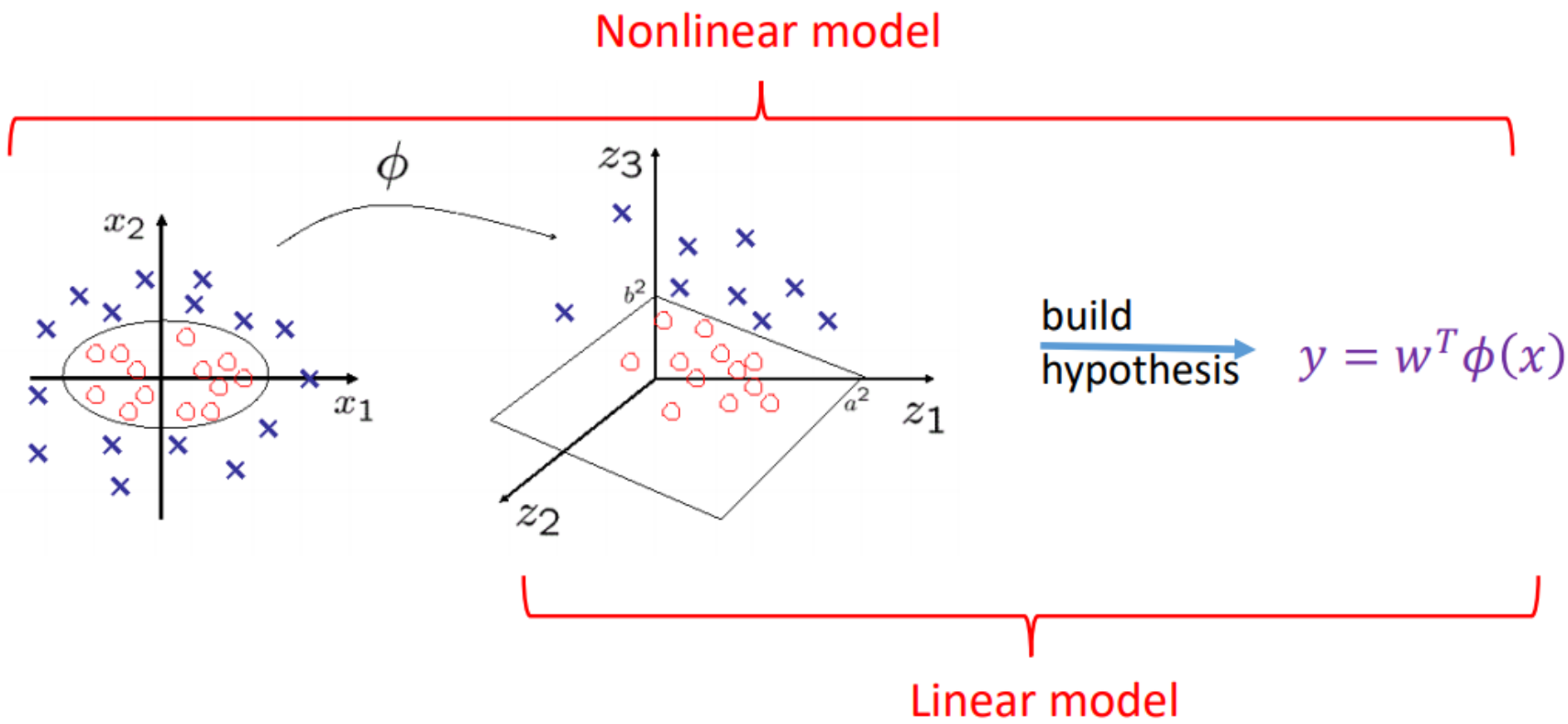
- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Hypothesis $y = \text{sign}(f_w(x)) = \text{sign}(w^T x)$
  - $y = +1$ if $w^T x > 0$
  - $y = -1$ if $w^T x < 0$

- Let's assume that we can optimize to find w

Deep learning: Kernel model

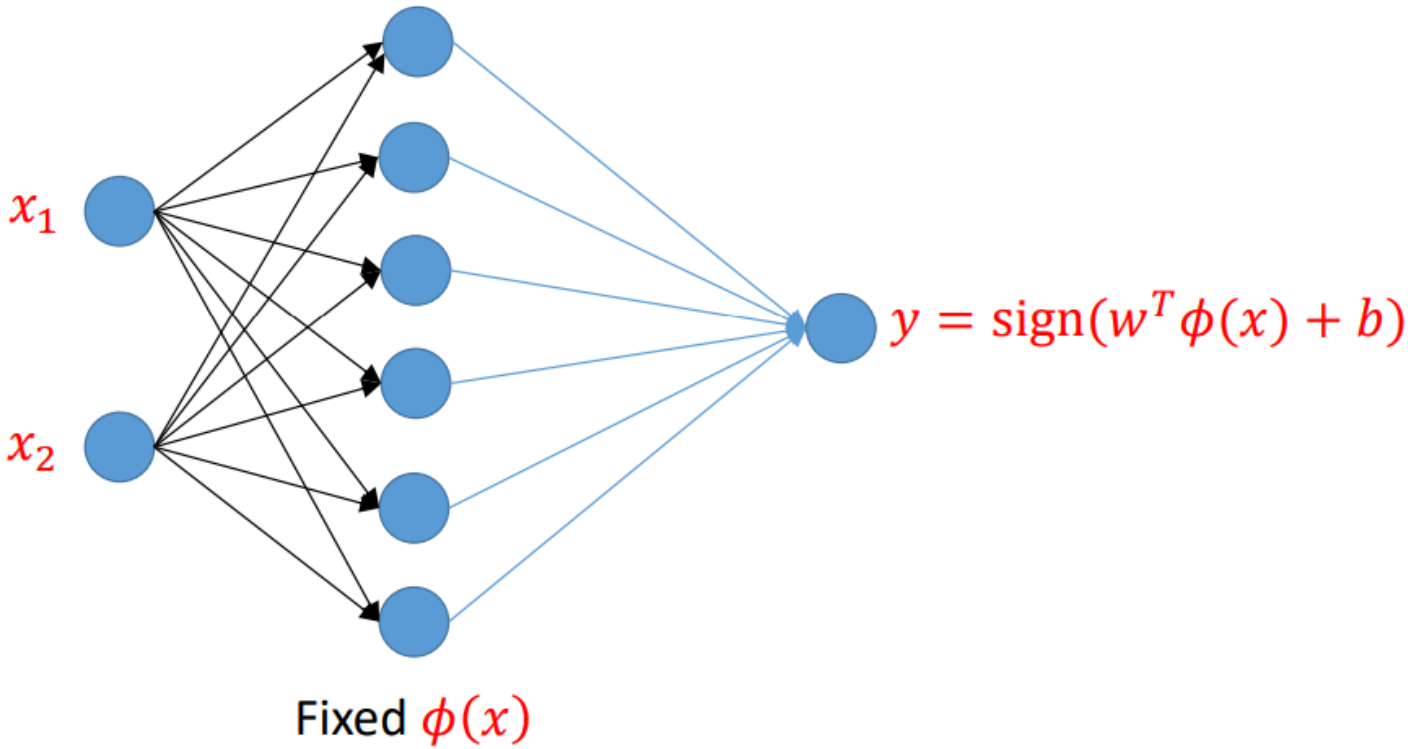
Make non-linear model linear

Features: part of the model



Deep learning: Feed forward

Example: Polynomial kernel SVM



Deep learning: Feed forward

Motivation: representation learning

- Why don't we also learn $\phi(x)$?

x

Learn $\phi(x)$

$\phi(x)$

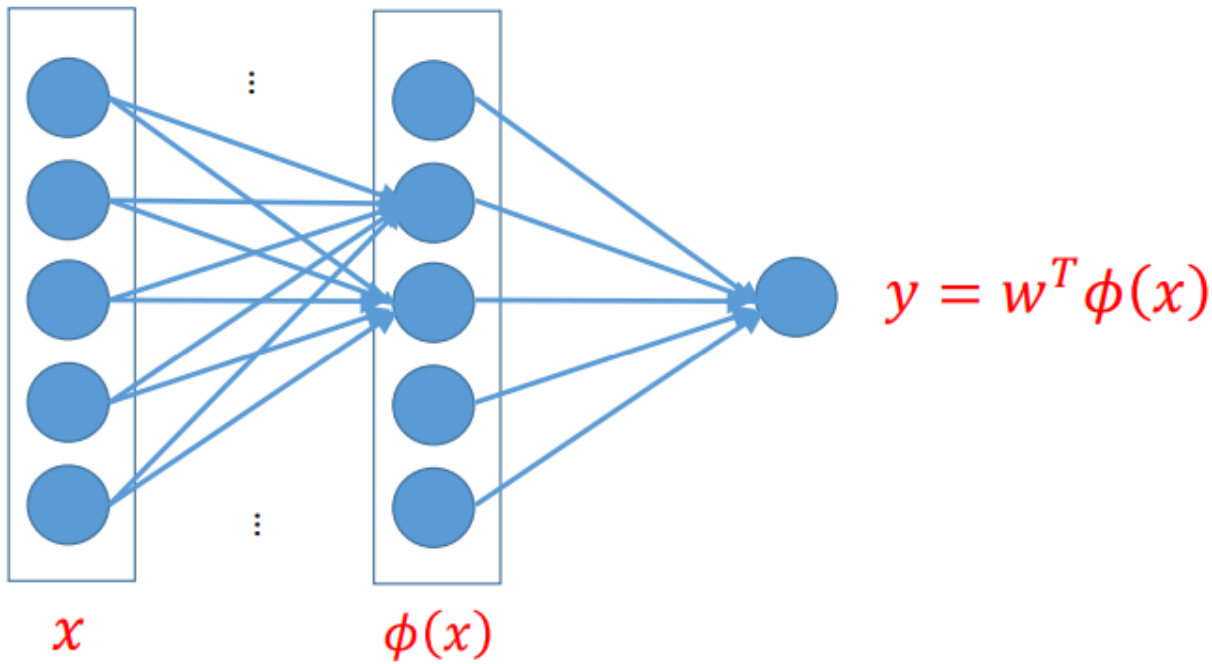
Learn w

$y = w^T \phi(x)$

Deep learning: Feed forward

Feedforward networks

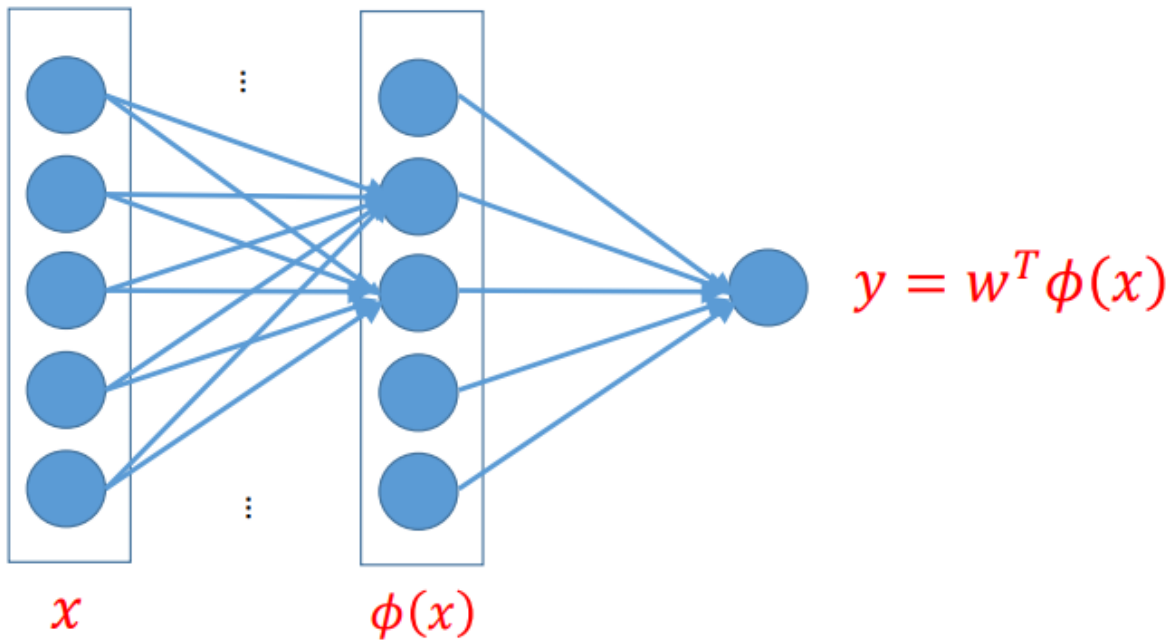
- View each dimension of $\phi(x)$ as something to be learned



Deep learning: Feed forward

Feedforward networks

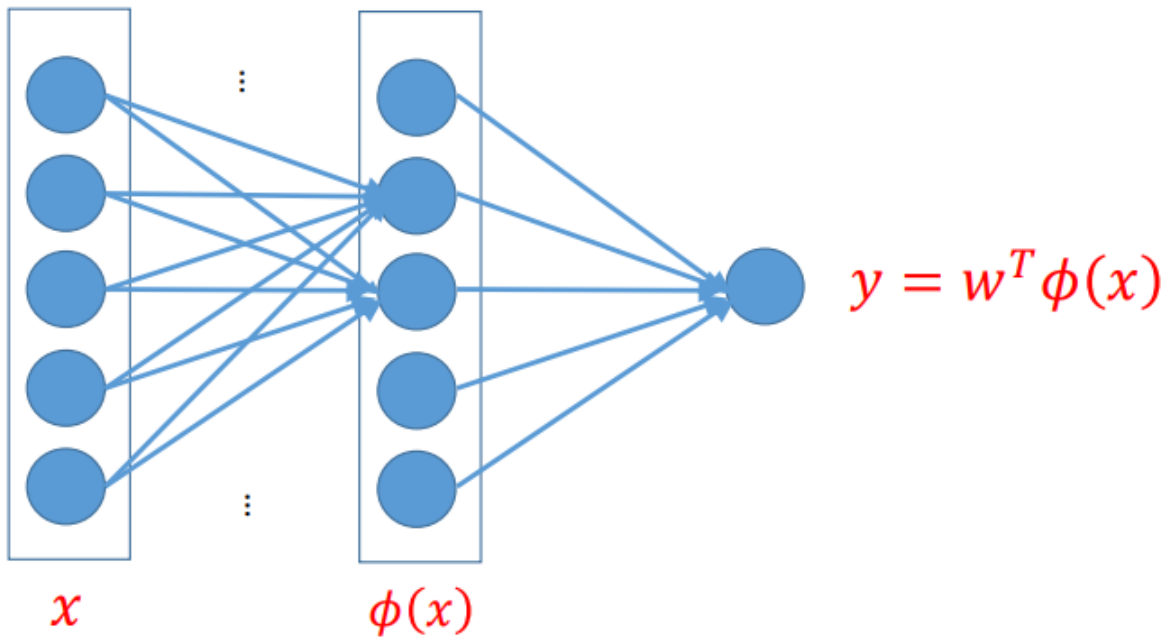
- Linear functions $\phi_i(x) = \theta_i^T x$ don't work: need some nonlinearity



Deep learning: Feed forward

Feedforward networks

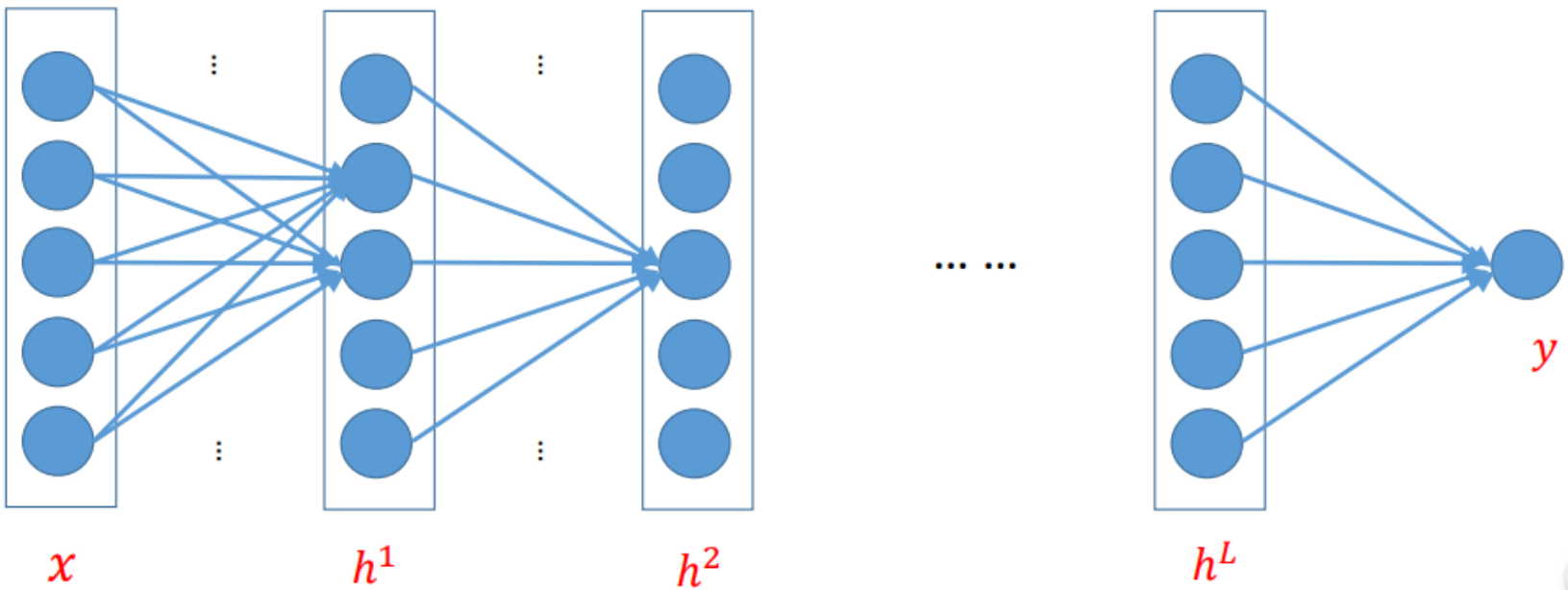
- Typically, set $\phi_i(x) = r(\theta_i^T x)$ where $r(\cdot)$ is some nonlinear function



Deep learning: Feed forward

Feedforward deep networks

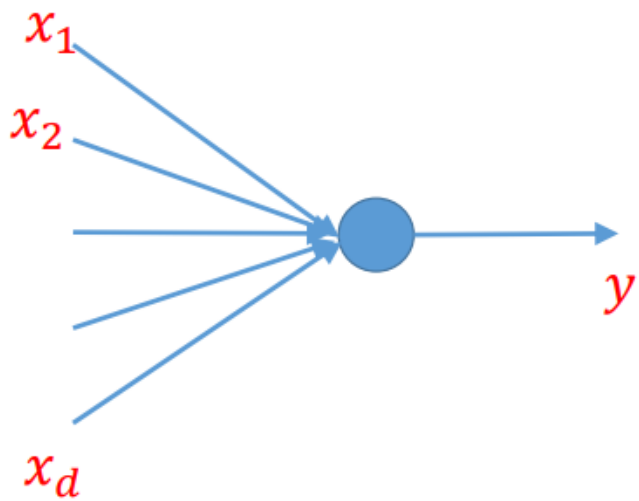
- What if we go deeper?



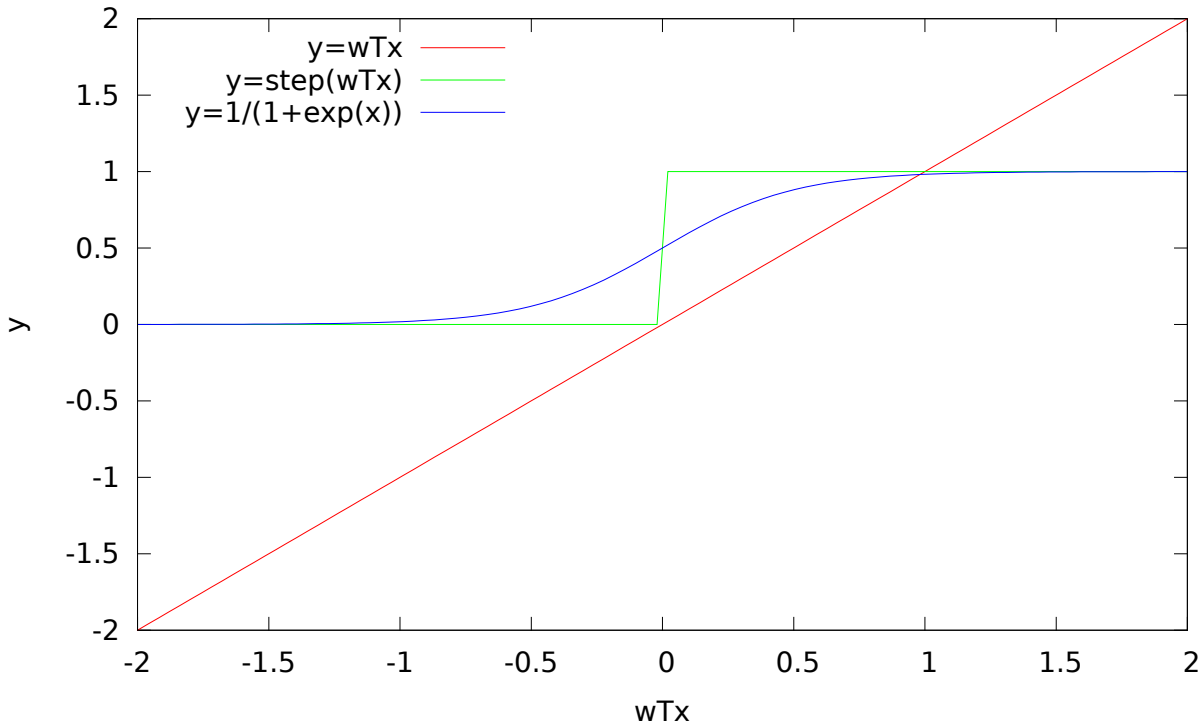
Deep learning: Feed forward

Motivation: abstract neuron model

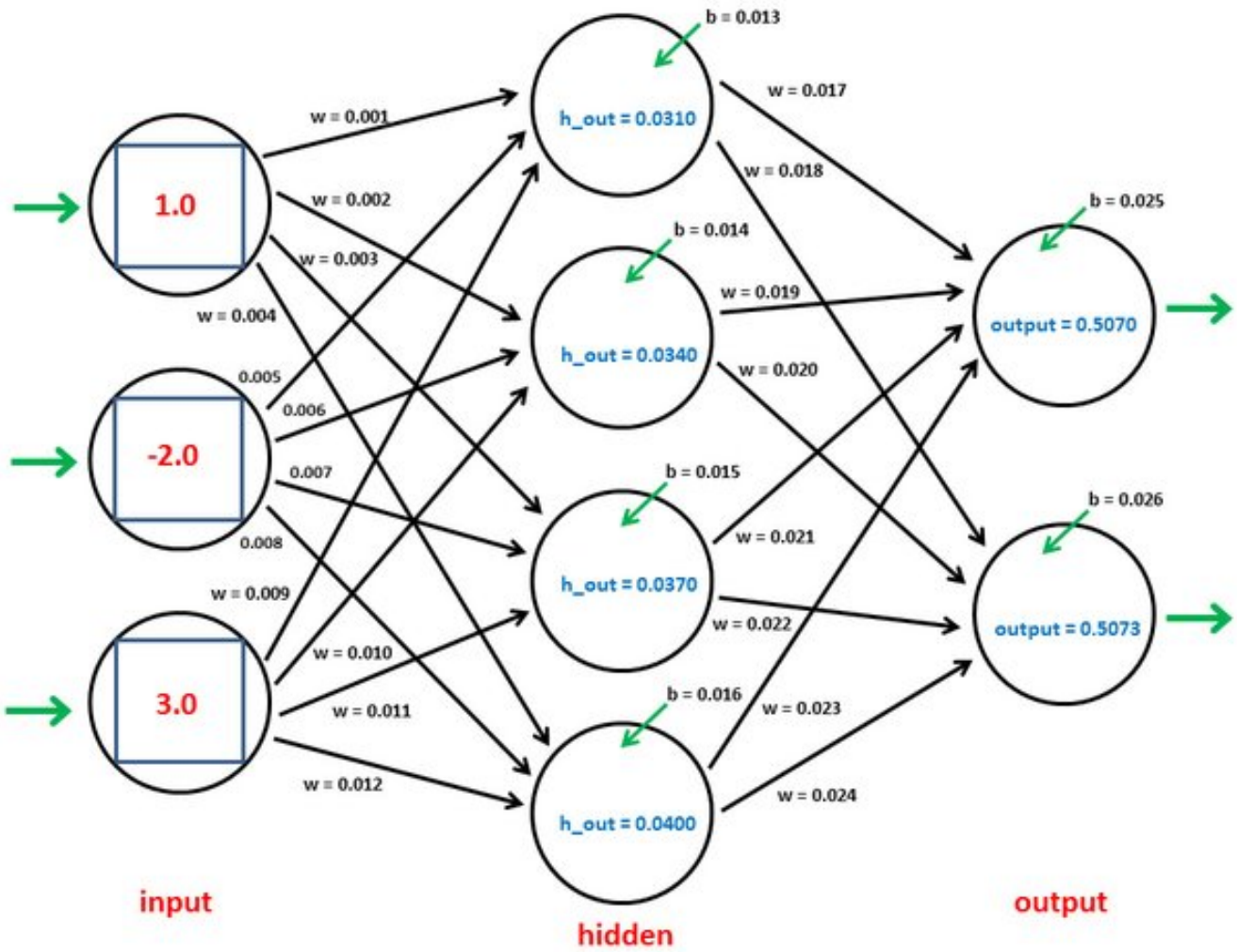
- Neuron activated when the correlation between the input and a pattern θ exceeds some threshold b
- $y = \text{threshold}(\theta^T x - b)$ or $y = r(\theta^T x - b)$
- $r(\cdot)$ called activation function



Deep learning: Activation function



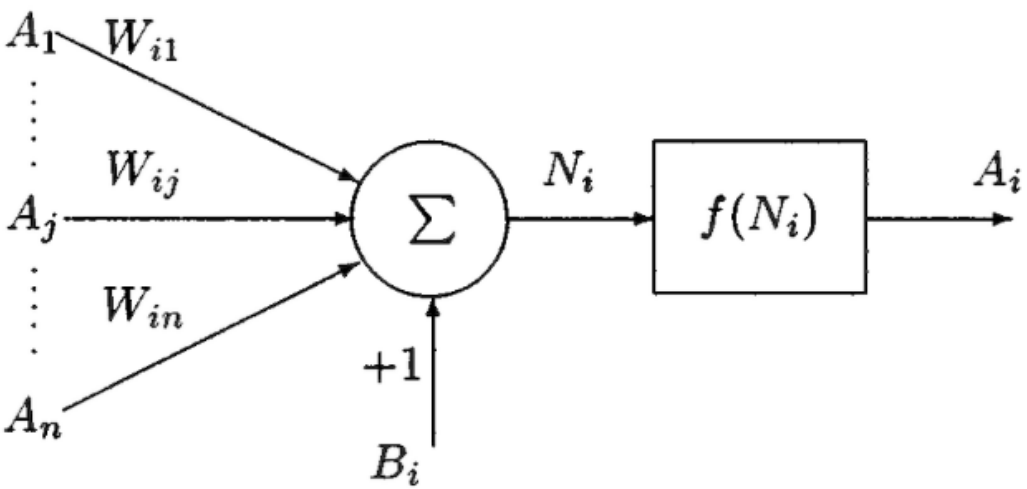
Fully connected neural networks



Ideas from Piotr Skalski, Pataki Bálint Ármin

Fully connected neural networks

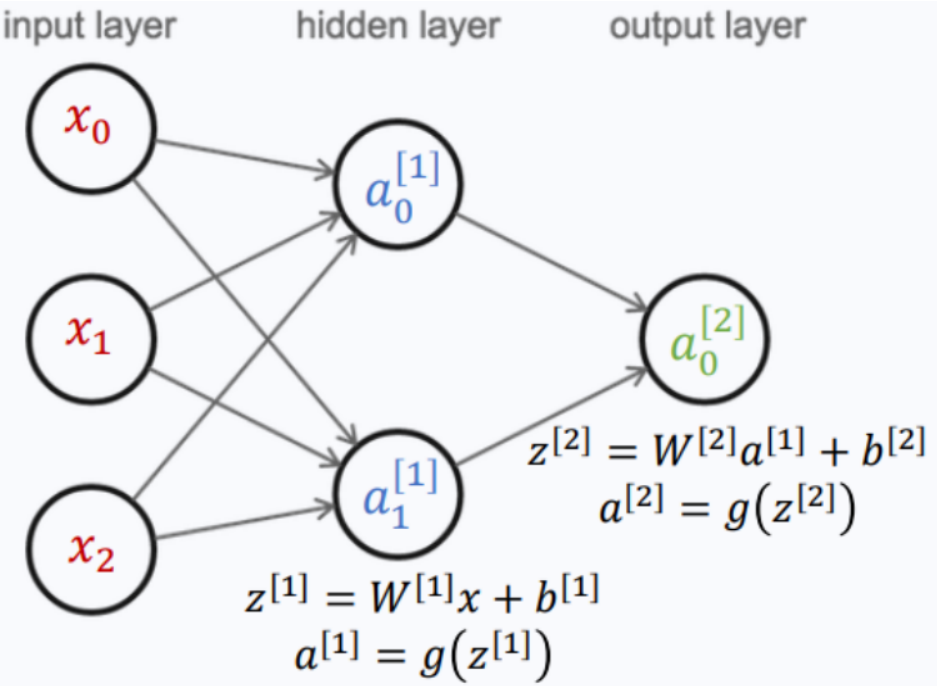
- Model:
  - Inputs (x_j) or for hidden layer l: A_j^{l-1}
  - Weight w_{ij}^l
  - Bias b_i^l
  - Weighted sum of input and bias: $z_i^l = \sum_j A_j^{l-1} w_{ij}^l + b_i^l$
  - Activation function (nonlinear) g: $A_i^l = g(z_i^l)$



Yang et el, 2000.

Feed forward

- Example



$$z_0^{[1]} = w_{0,0}^{[1]}x_0 + w_{0,1}^{[1]}x_1 + w_{0,2}^{[1]}x_2 + b_0^{[1]}$$
$$a_0^{[1]} = g\left(z_0^{[1]}\right)$$

$$z_1^{[1]} = w_{1,0}^{[1]}x_0 + w_{1,1}^{[1]}x_1 + w_{1,2}^{[1]}x_2 + b_0^{[1]}$$
$$a_1^{[1]} = g\left(z_1^{[1]}\right)$$

$$z_0^{[2]} = w_{0,0}^{[2]}a_0 + w_{0,1}^{[2]}a_1 + b_0^{[2]}$$
$$a_0^{[2]} = g\left(z_0^{[2]}\right)$$

- We have an output, how to change weights and biases to achieve the desired output?
- Error L

Backpropagation

- $$\Delta W = -\alpha \frac{\partial L}{\partial W}$$
- W is a large three dimensional matrix
- Chain rule!

Geoffrey Hinton

FOLLOW

Emeritus Prof. Comp Sci, U.Toronto & Engineering Fellow, Google
Verified email at cs.toronto.edu - Homepage

machine learning neural networks artificial intelligence cognitive science computer science

TITLE	CITED BY	YEAR
Learning internal representations by error-propagation DE Rumelhart, GE Hinton, RJ Williams Parallel Distributed Processing: Explorations in the Microstructure of …	63004 *	1986

Backpropagation

- Chain rule

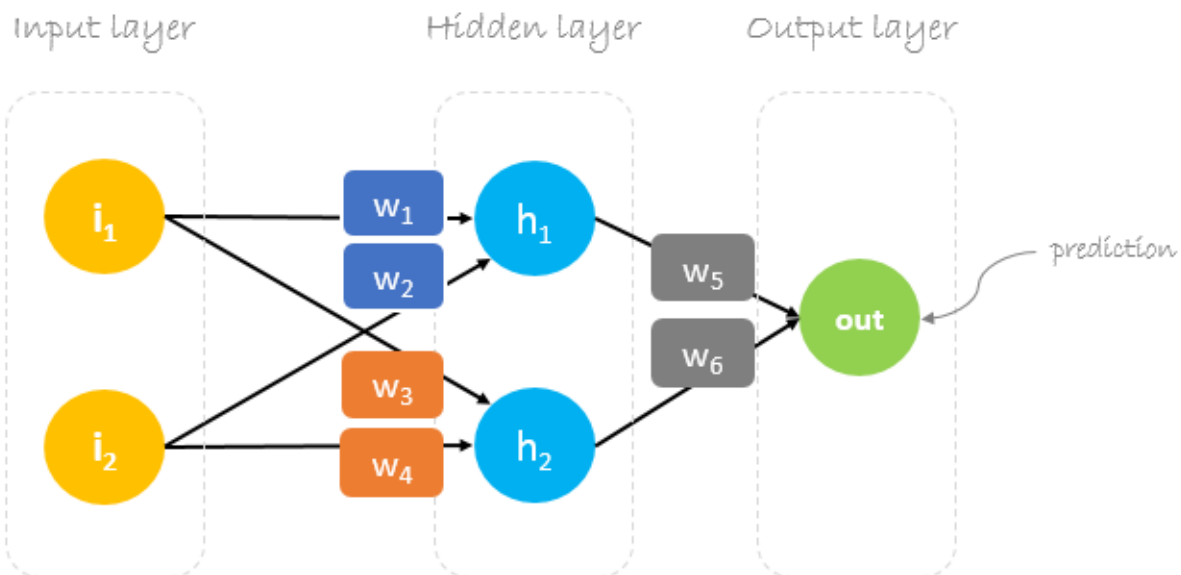
$$a_i = g(z_i) = g(w_{ij}a_j + b_i)$$

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial w_{ij}} = \frac{\partial L}{\partial a_i}g'(z_i)a_j$$

$$\frac{\partial L}{\partial a_i} = \sum_{l \in L}\frac{\partial L}{\partial a_l}\frac{\partial a_l}{\partial z_l}\frac{\partial z_l}{\partial a_i} = \sum_{l \in L}\frac{\partial L}{\partial a_l}g'(z_l)w_{li}$$

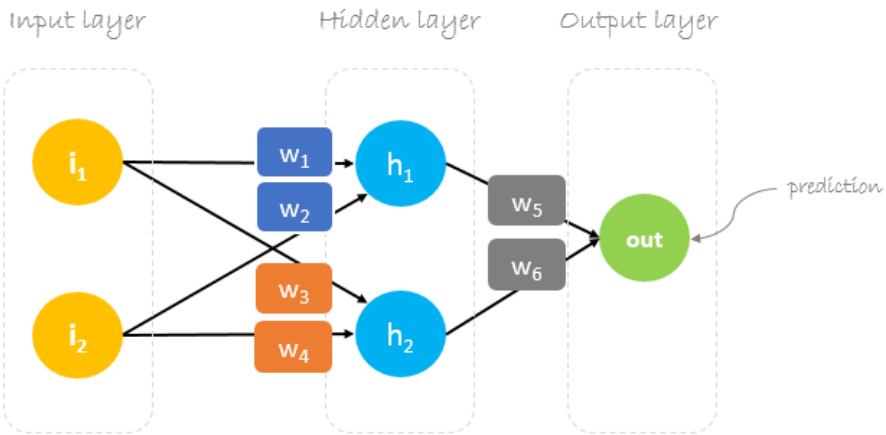
Backpropagation: Example

- From HMKCode
- Note that there is no activation function (it would just add one more step in the chain rule)



Backpropagation: Example

- Δ: prediction $-$ actual

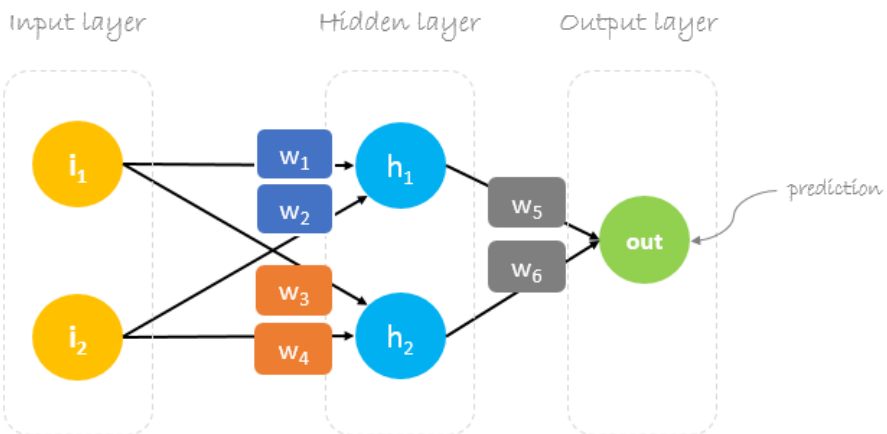


Updated weights

$$
\begin{aligned}
{}^{*}w_6 &= w_6 - a\,(h_2 \cdot \Delta) \\
{}^{*}w_5 &= w_5 - a\,(h_1 \cdot \Delta) \\
{}^{*}w_4 &= w_4 - a\,(i_2 \cdot \Delta w_6) \\
{}^{*}w_3 &= w_3 - a\,(i_1 \cdot \Delta w_6) \\
{}^{*}w_2 &= w_2 - a\,(i_2 \cdot \Delta w_5) \\
{}^{*}w_1 &= w_1 - a\,(i_1 \cdot \Delta w_5)
\end{aligned}
$$

Backpropagation: Example

- Summarized in matrix form
- No wonder why graphic cards are so useful for this!



$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - a\,\Delta \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - \begin{bmatrix} a h_1 \Delta \\ a h_2 \Delta \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - a\,\Delta \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} . \begin{bmatrix} w_5 & w_6 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - \begin{bmatrix} a\, i_1 \Delta w_5 & a\, i_1 \Delta w_6 \\ a\, i_2 \Delta w_5 & a\, i_2 \Delta w_6 \end{bmatrix}$$

Backpropagation: Formula

- Last error is in $d\mathbf{A}^l$. Originally it is

$$\mathbf{L} = \mathbf{Y} - \mathbf{A}^l$$

- Algorithm:

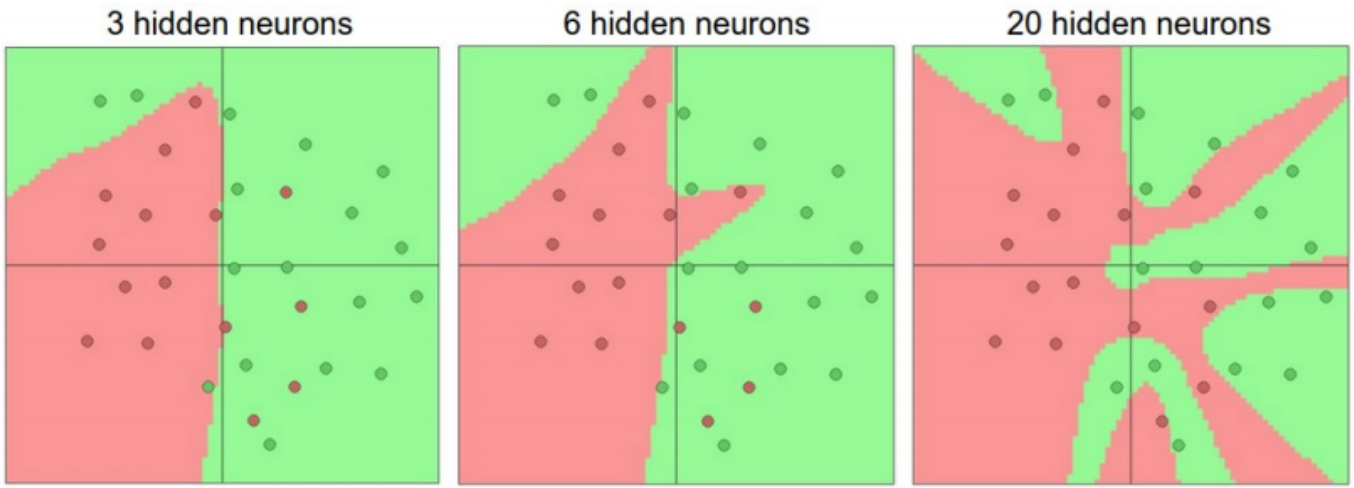
$$\begin{aligned} d\mathbf{Z}^l &= d\mathbf{A}^l * g'(\mathbf{Z}^l) \\ d\mathbf{A}^{l-1} &= \frac{\partial \mathbf{L}}{\partial \mathbf{A}^{l-1}} = (\mathbf{W}^l)^T d\mathbf{Z}^l \\ d\mathbf{W}^l &= \frac{\partial \mathbf{L}}{\partial \mathbf{W}^l} = \frac{1}{m} d\mathbf{Z}^l (\mathbf{A}^{l-1})^T, \end{aligned} \tag{1}$$

  where m is the number of components of the layer l

- $\mathbf{Z}^l$ is the result of the sum at layer l
- $\mathbf{A}^l$ is the result after applying the nonlinear activation function
- $*$ is an elementwise product of two vectors

How many layers?

- Neural network with at least one hidden layer is a universal approximator (can represent any function).



Do Deep Nets Really Need to be Deep? Jimmy Ba, Rich Caruana,

Deep learning: Overfitting: Need a lot of data

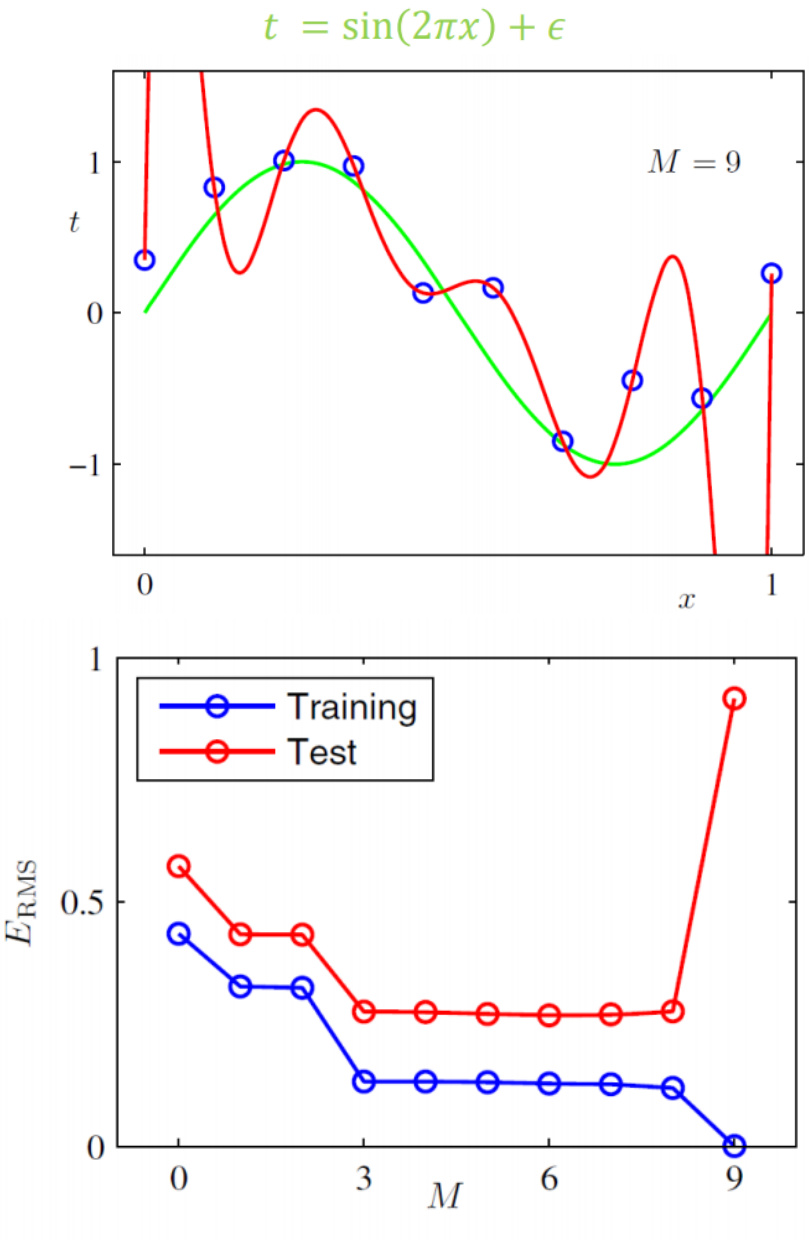
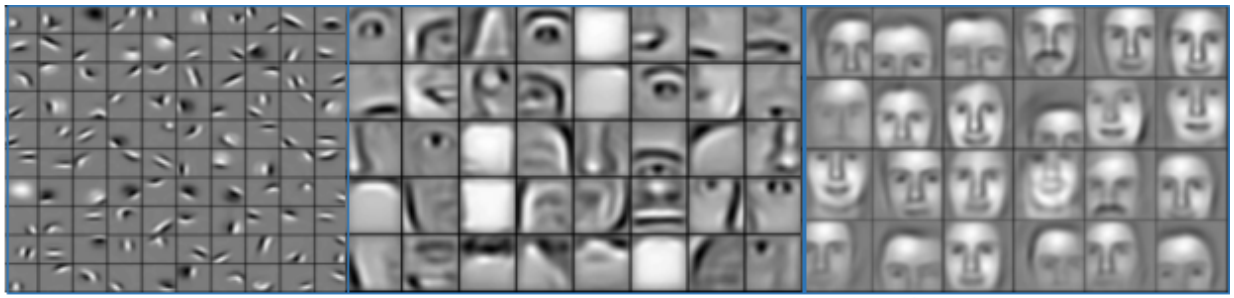
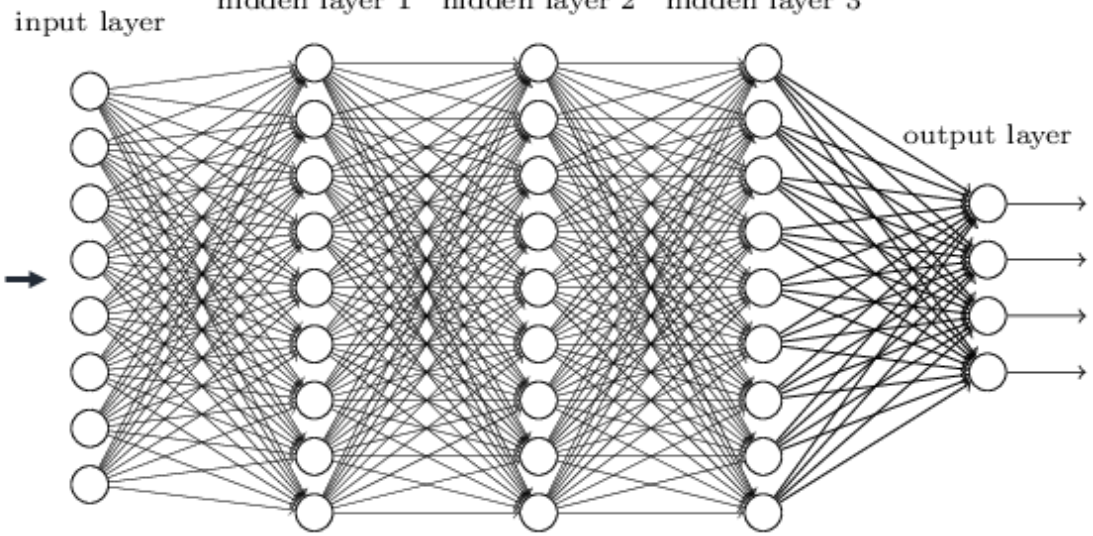


Figure from *Machine Learning and Pattern Recognition*, Bishop

Deep learning: Features example

Deep neural networks learn hierarchical feature representations

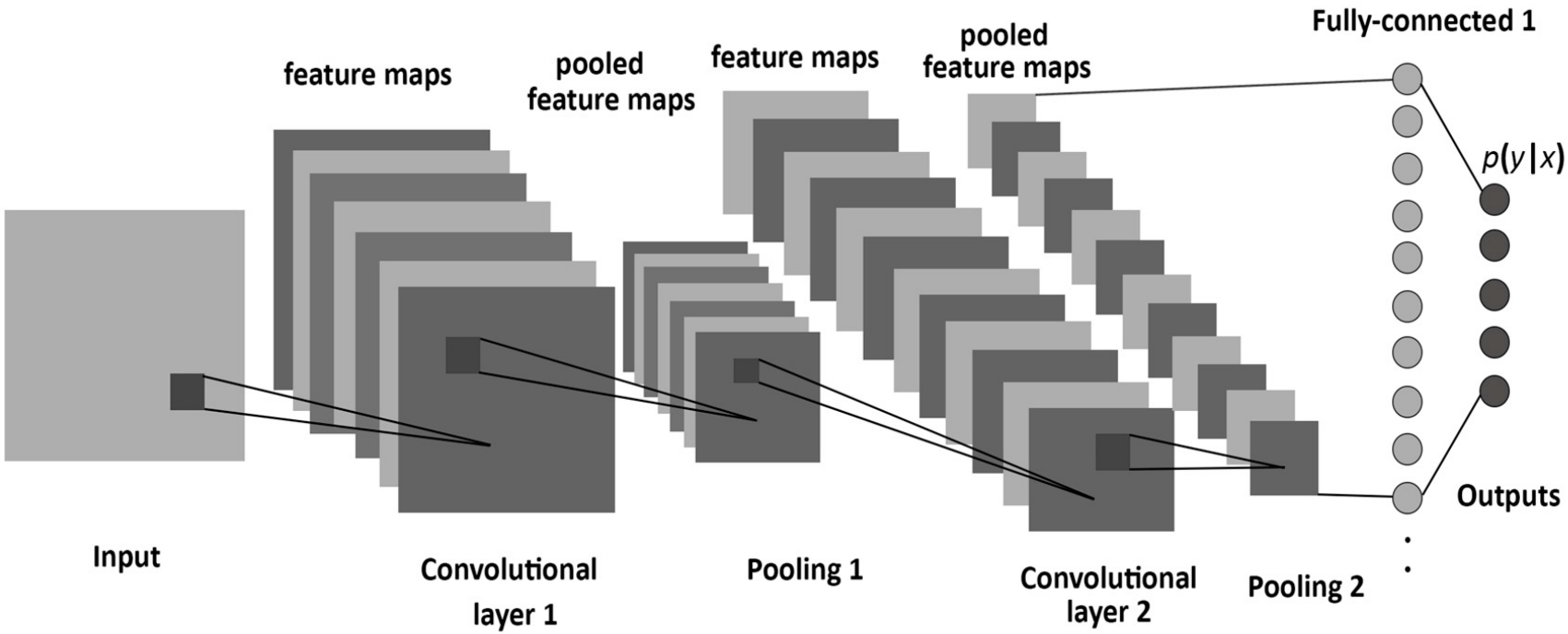




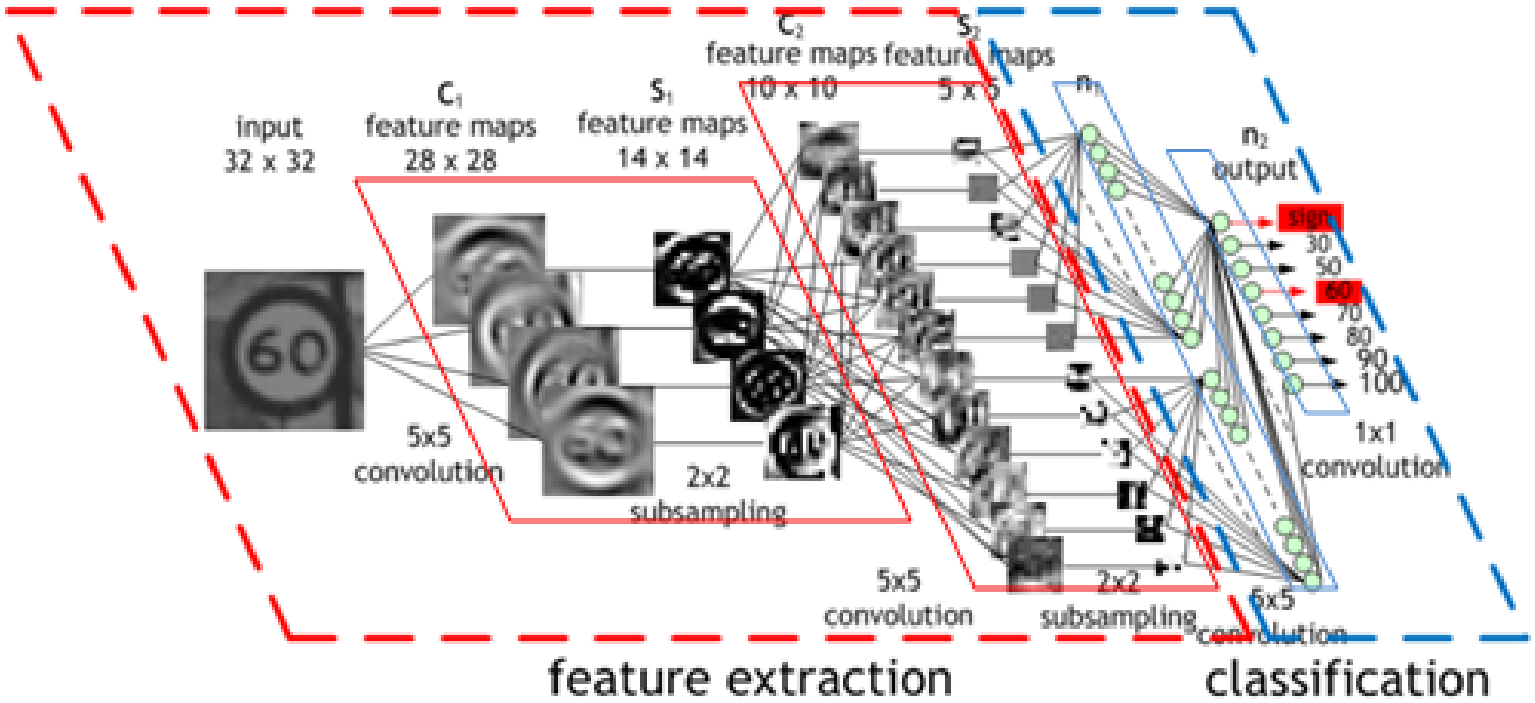
Convolution operator: examples

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Deep learning: Convolutional Neural Network



Deep learning: Convolutional Neural Network

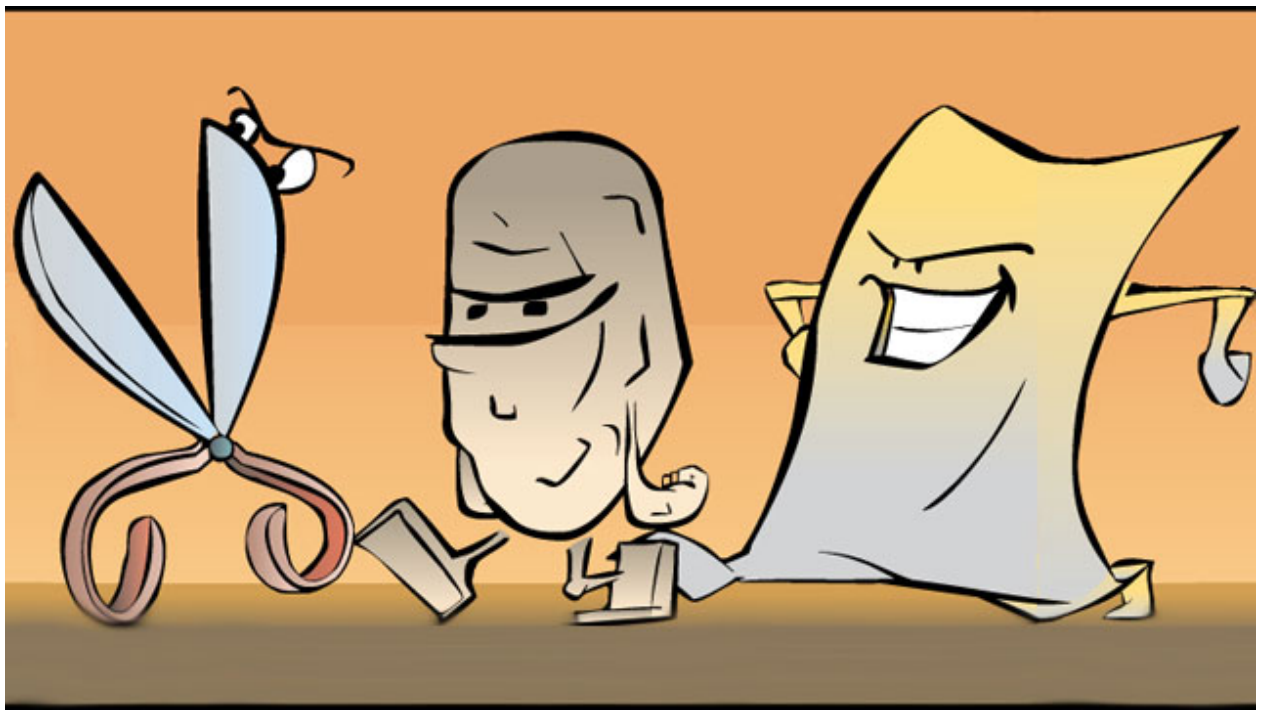


Game models:

- Rock-paper-scissors
- Prisoner's dilemma
- Chicken, hawk-dove game

Rock-paper-scissors

- No winning strategy on (truly) random opponent
- E.g bacteria and antibiotics in mice
- Grass-rabbit-fox
- Popular in games



Prisoner's Dilemma

- Two people playing the game
- Two options: Cooperate, Defect
- Cooperate: Confess the crime
- Defect: deny the crime
- Result: years in prison

	Cooperate	Defect
Cooperate	-1, -1	-3, 0
Defect	0, -3	-2, -2

Prisoner's Dilemma

- Payoff matrix
- Reward for actions based on other player's actions

	Cooperate	Defect
Cooperate	2, 2	0, 3
Defect	3, 0	1, 1

	Cooperate	Defect
Cooperate	1, 1	0, 2
Defect	0, 2	0, 0

Prisoner's Dilemma

- Each player with a preferred strategy that collectively results in an inferior outcome
- Dominating strategy regardless of the opponent's action
- Nash equilibrium, from which no individual player benefits from deviating

	Cooperate	Defect
Cooperate	2, 2	0, 3
Defect	3, 0	1, 1

	Cooperate	Defect
Cooperate	1, 1	0, 2
Defect	0, 2	0, 0

Prisoner's Dilemma

- One game $\rightarrow$ defect
- Fixed number of games $\rightarrow$ defect

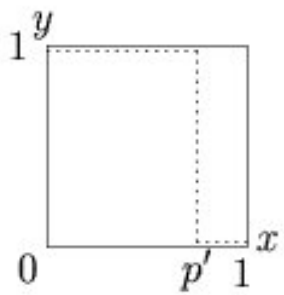
Chicken game, Hawk-Dove game

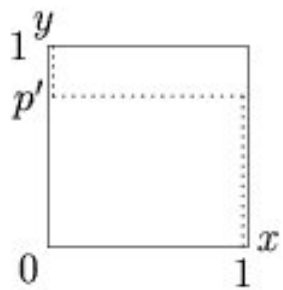
Chicken game, Hawk-Dove game

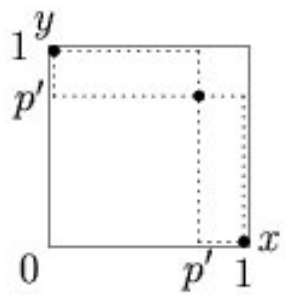
- No preferred strategy
- The best strategy is to anti-coordinate with your opponent

	Cooperate	Defect
Cooperate	0, 0	-1, 2
Defect	2, -1	-5, -5

- Example: Cold war
- Solution: anti-correlated pure strategy
- Probabilistic, or mixed strategy (play Hawk with p')







Chicken game, Hawk-Dove game difference to Prisoner's dilemma

	Cooperate	Defect
Cooperate	Reward	S, T
Defect	T, S	Punish

Hawk-Dove

	C	D
C	2, 2	1, 3
D	3, 1	0, 0

Prisoner's dilemma

	C	D
C	2, 2	0, 3
D	3, 0	1,1

- Prisoner's dilemma:
  Temptation(T)>Reward(R)>Punish(P)>Sucker(S)
- Chicken game:
  Temptation(T)>Reward(R)>Sucker(S)>Punish(P)

Stag game

Prisoner's dilemma

	C	D
C	2, 2	0, 3
D	3, 0	1,1

Hawk-Dove

	C	D
C	2, 2	1, 3
D	3, 1	0, 0

Stag game

	C	D
C	3, 3	0, 2
D	2, 0	1,1

- Prisoner's dilemma:
  Temptation(T)>Reward(R)>Punish(P)>Sucker(S)
- Chicken game:
  Temptation(T)>Reward(R)>Sucker(S)>Punish(P)
- Stag game:
  Reward(R)>Temptation(T)>Punish(P)>Sucker(S)

Prisoner's dilemma: multiple agents

- Against all others
- Against itself
- Against a fully random agent
- Number of agents: 14, 62

Prisoner's dilemma: multiple agents: Strategies

- Strategies for repeated games in Axelrod's tournament (1980):
- AllD: choosing D always (unconditional defector, the bad guy, . . .)
- AllC: choosing C always („the good guy" or sucker)
- Random: chooses D or C with probabilities q or (1-q)
- TFT (Tit-for tat): chooses C first, then she repeats/reciprocates the previous strategy of the co-player
- Generous TFT: TFT, but chooses C (instead of D) with a probability q
- WSLS (win-stay-lose-shift): first C or D, then she changes it if her payoff is smaller than an aspiration level (Ux<a)
- Stochastic reactive strategies: Chooses C or D with probabilities dependent on the previous decision of the co-player
- Stochastic reactive strategies with longer memory: Etc.
- Go-by-Majority cooperates on the first round, then takes majority strategy.
- . . . and many more

Multiple agents: Winning strategy

- The winner is: Tit-for-tat!
- Human law
- Note that Common good was not included
- Why not "always defect"(AD), which is the Nash equilibrium of the
- Prisoners' dilemma for any finite number of plays?
- Nash equilibrium means that AS is the best strategy against AD
- AS is not dominant strategy
- It is not the best strategy for all strategies

Multiple agents: Best strategy

- Large pool of players (movie):
- It can be shown that for a repeated PD game there is no best strategy for all possible strategies
- But for a good strategy it has to be
  - Nice (do not defect first)
  - Punish others for being nasty
  - Forgive fast
  - Be efficient against yourself