Neural Network Part 5: Unsupervised Models

Yingyu Liang

Computer Sciences 760

Fall 2017

http://pages.cs.wisc.edu/~yliang/cs760/

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Matt Gormley, Elad Hazan, Tom Dietterich, and Pedro Domingos.

Goals for the lecture

you should understand the following concepts

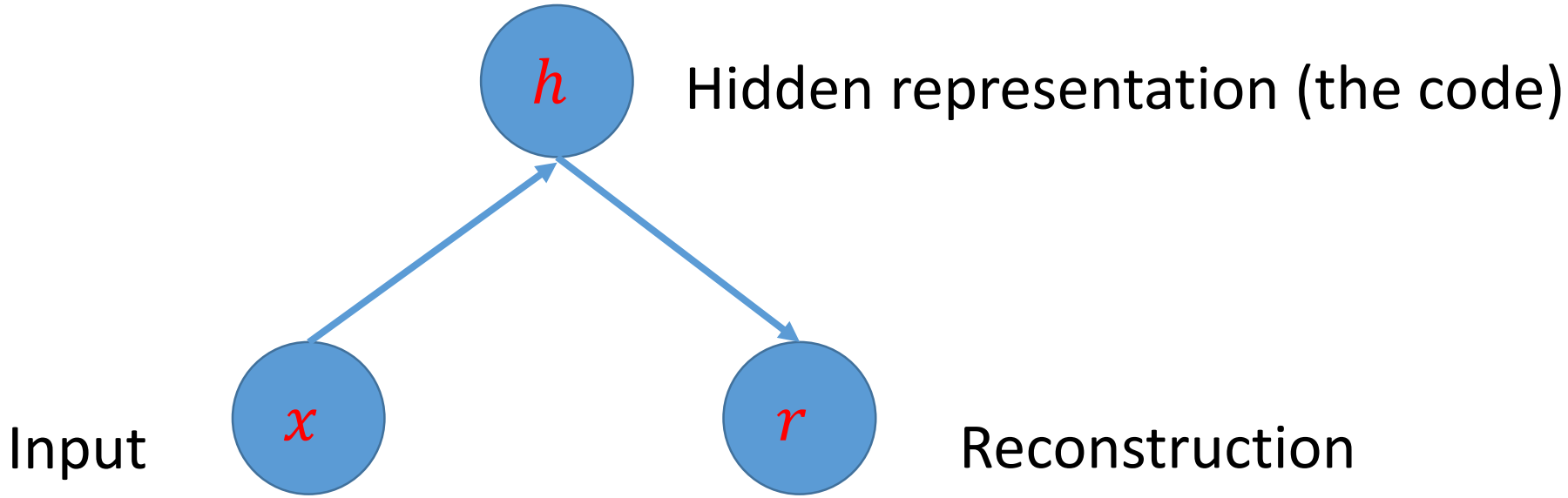
- autoencoder
- restricted Boltzmann machine (RBM)
- Nash equilibrium
- minimax game
- generative adversarial network (GAN)

Autoencoder

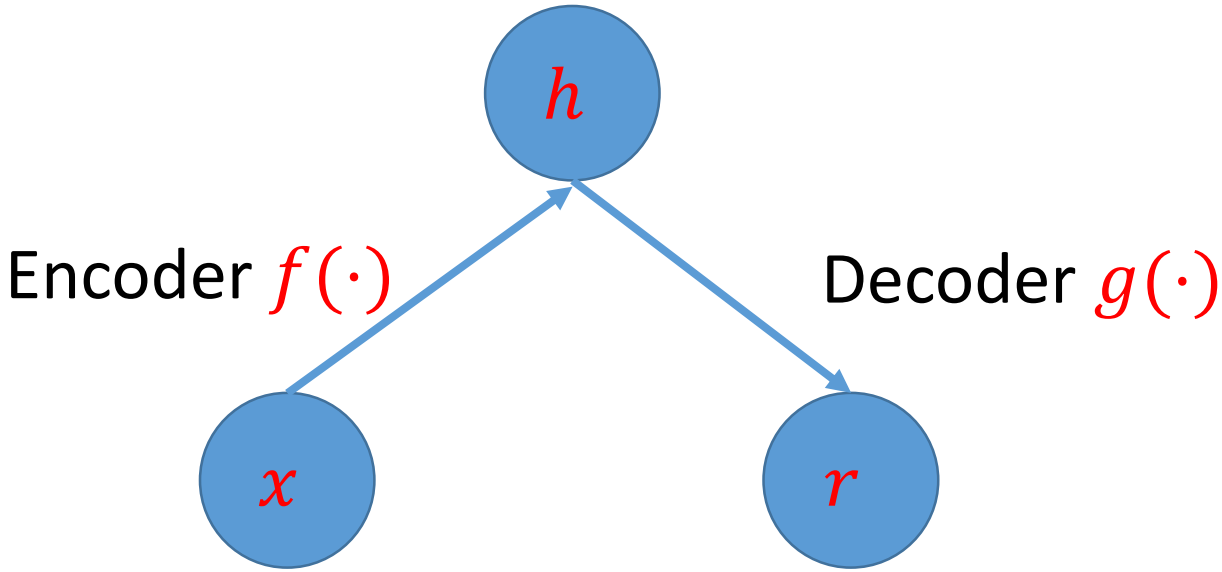
- Neural networks trained to attempt to copy its input to its output

- Contain two parts:
  - Encoder: map the input to a hidden representation
  - Decoder: map the hidden representation to the output

Autoencoder



Autoencoder



$$h = f(x), r = g(h) = g(f(x))$$

Why want to copy input to output

- Not really care about copying

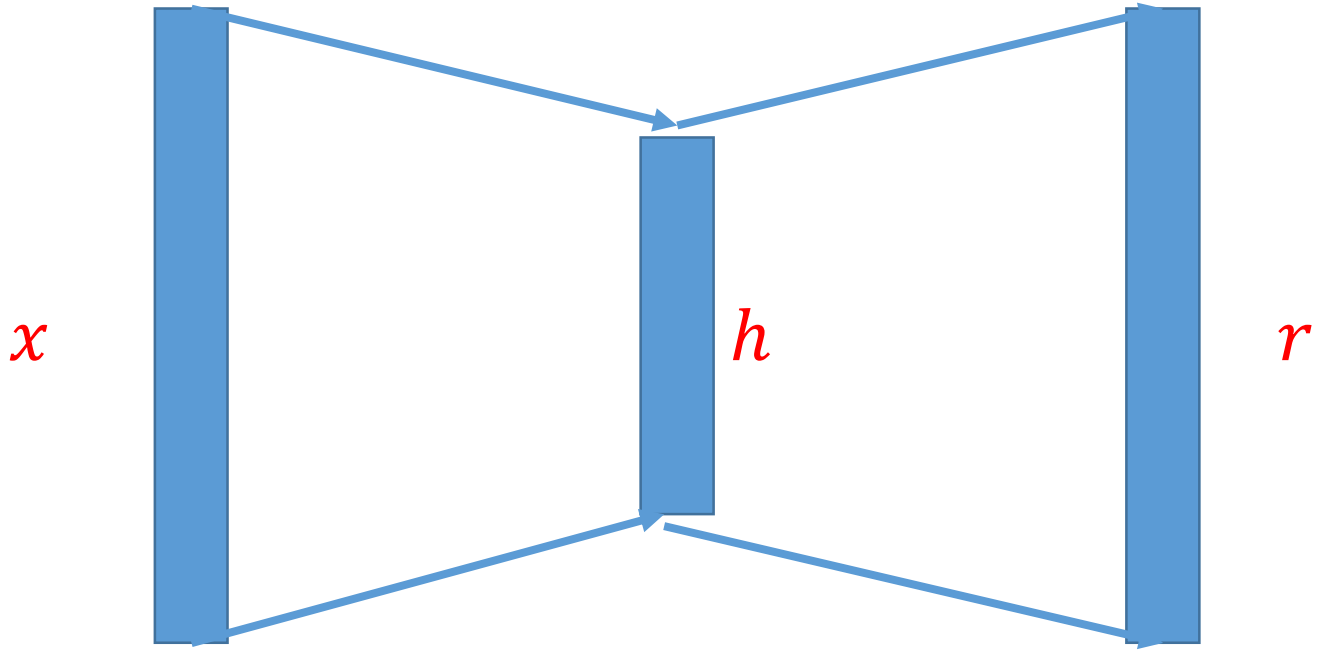
- Interesting case: NOT able to copy exactly but strive to do so
- Autoencoder forced to select which aspects to preserve and thus hopefully can learn useful properties of the data

- Historical note: goes back to (LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1994).

Undercomplete autoencoder

- Constrain the code to have smaller dimension than the input
- Training: minimize a loss function

$$L(x, r) = L(x, g(f(x)))$$



Undercomplete autoencoder

- Constrain the code to have smaller dimension than the input
- Training: minimize a loss function

$$L(x, r) = L(x, g(f(x)))$$

- Special case: f, g linear, L mean square error
- Reduces to Principal Component Analysis

Undercomplete autoencoder

- What about nonlinear encoder and decoder?

- Capacity should not be too large
- Suppose given data $x_1, x_2, \ldots, x_n$
  - Encoder maps x_i to i
  - Decoder maps i to x_i
- One dim h suffices for perfect reconstruction

Regularization

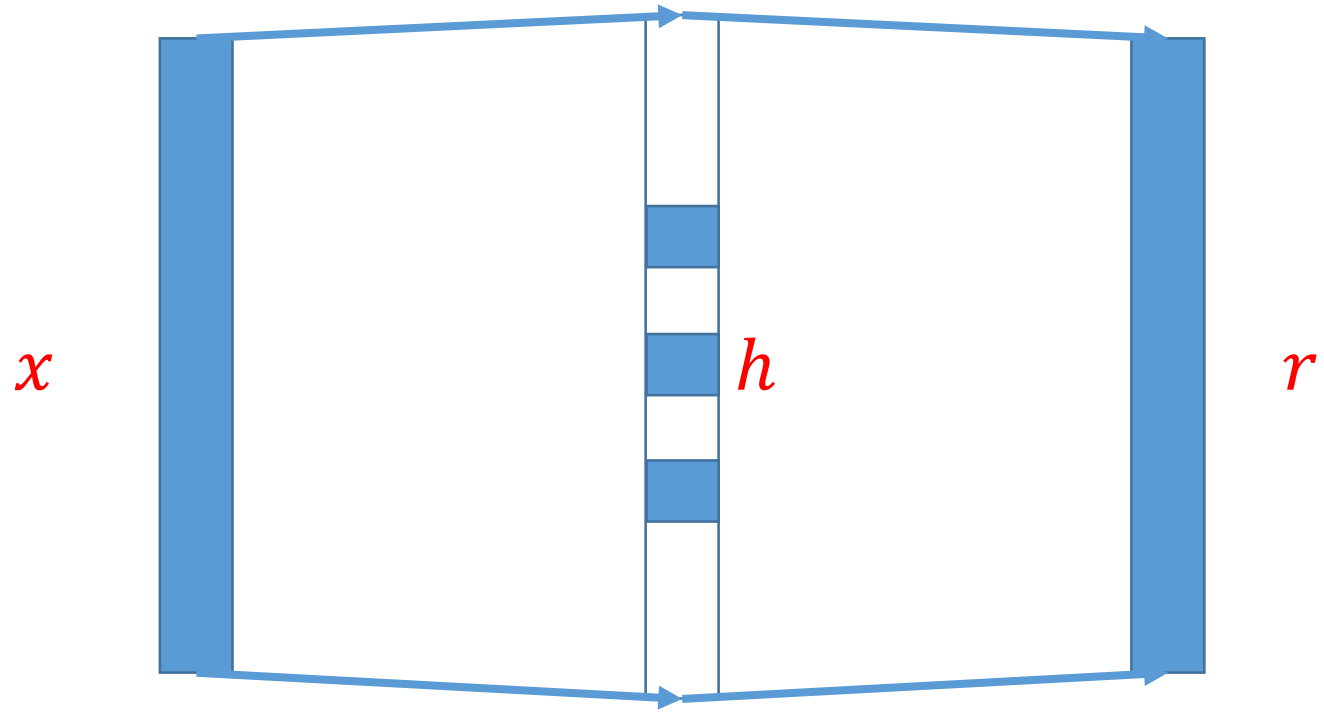
- Typically NOT
  - Keeping the encoder/decoder shallow or
  - Using small code size

- Regularized autoencoders: add regularization term that encourages the model to have other properties
  - Sparsity of the representation (sparse autoencoder)
  - Robustness to noise or to missing inputs (denoising autoencoder)

Sparse autoencoder

- Constrain the code to have sparsity
- Training: minimize a loss function

$$L_R = L(x, g(f(x))) + R(h)$$



Probabilistic view of regularizing h

- Suppose we have a probabilistic model $p(h, x)$
- MLE on x

$$\log p(x) = \log \sum_{h'} p(h', x)$$

- ☹ Hard to sum over h'

Probabilistic view of regularizing h

- Suppose we have a probabilistic model $p(h, x)$
- MLE on x

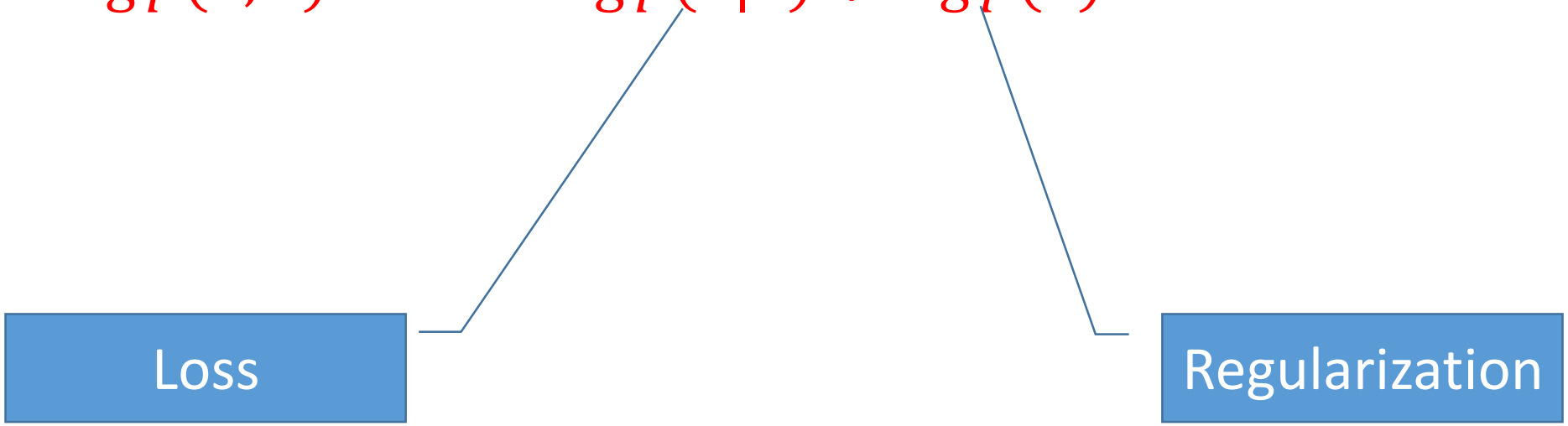
$$\max \log p(x) = \max \log \sum_{h'} p(h', x)$$

- Approximation: suppose $h = f(x)$ gives the most likely hidden representation, and $\sum_{h'} p(h', x)$ can be approximated by $p(h, x)$

Probabilistic view of regularizing h

- Suppose we have a probabilistic model $p(h, x)$
- Approximate MLE on $x, h = f(x)$

$$\max \log p(h, x) = \max \log p(x|h) + \log p(h)$$



Sparse autoencoder

- Constrain the code to have sparsity
- Laplacian prior: $p(h) = \frac{\lambda}{2}\exp(-\frac{\lambda}{2}|h|_1)$

- Training: minimize a loss function

$$L_R = L(x, g(f(x))) + \lambda|h|_1$$

Denoising autoencoder

- Traditional autoencoder: encourage to learn $g\big(f(\cdot)\big)$ to be identity

- Denoising : minimize a loss function

$$L(x, r) = L(x, g\big(f(\tilde{x})\big))$$

where $\tilde{x}$ is $x + noise$

Boltzmann machine

- Introduced by Ackley *et al.* (1985)

- General “connectionist” approach to learning arbitrary probability distributions over binary vectors
- Special case of energy model: $p(x) = \frac{\exp(-E(x))}{Z}$

Boltzmann machine

- Energy model:

$$p(x) = \frac{\exp(-E(x))}{Z}$$

- Boltzmann machine: special case of energy model with

$$E(x) = -x^T U x - b^T x$$

where U is the weight matrix and b is the bias parameter

Boltzmann machine with latent variables

- Some variables are not observed

$$x = (x_v, x_h), \qquad x_v \text{ visible}, \; x_h \text{ hidden}$$

$$E(x) = -x_v^T R x_v - x_v^T W x_h - x_h^T S x_h - b^T x_v - c^T x_h$$

- Universal approximator of probability mass functions

Maximum likelihood

- Suppose we are given data $X = (x_v^1, x_v^2, \dots, x_v^n)$
- Maximum likelihood is to maximize

$$\log p(X) = \sum_i \log p(x_v^i)$$

where

$$p(x_v) = \sum_{x_h} p(x_v, x_h) = \sum_{x_h} \frac{1}{Z} \exp(-E(x_v, x_h))$$

- $Z = \sum \exp(-E(x_v, x_h))$: partition function, difficult to compute

Restricted Boltzmann machine

- Invented under the name *harmonium* (Smolensky, 1986)
- Popularized by Hinton and collaborators to *Restricted Boltzmann machine*

Restricted Boltzmann machine

- Special case of Boltzmann machine with latent variables:

$$p(v,h) = \frac{\exp(-E(v,h))}{Z}$$

where the energy function is

$$E(v,h) = -v^T W h - b^T v - c^T h$$

with the weight matrix W and the bias b, c

- Partition function

$$Z = \sum_v \sum_h \exp(-E(v,h))$$

Restricted Boltzmann machine

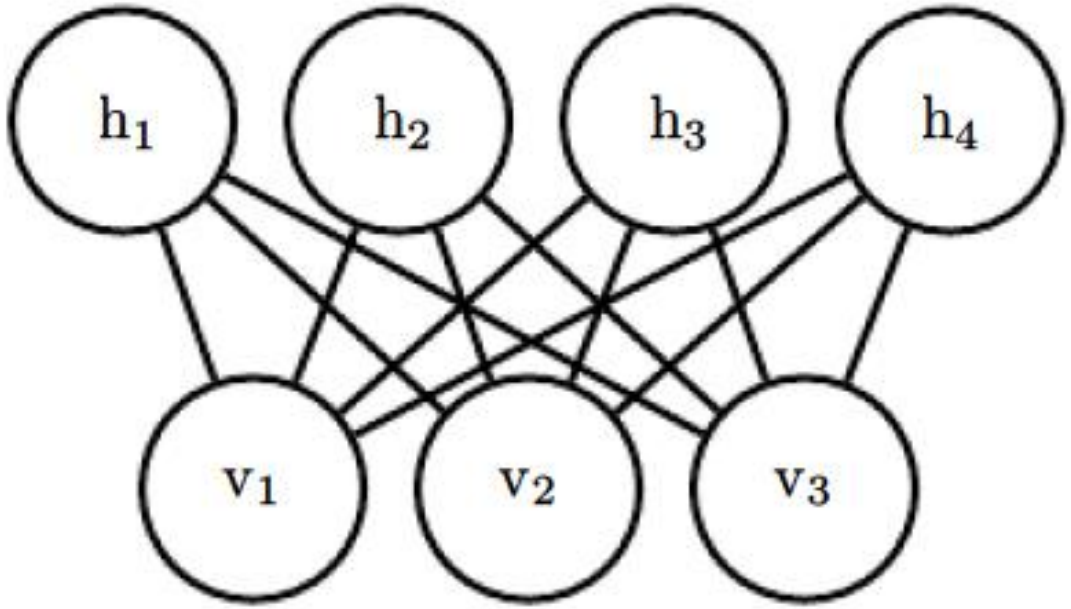


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Restricted Boltzmann machine

- Conditional distribution is factorial

$$p(h|v) = \frac{p(v,h)}{p(v)} = \prod_j p(h_j|v)$$

and

$$p(h_j = 1|v) = \sigma(c_j + v^T W_{:,j})$$

is logistic function

Restricted Boltzmann machine

- Similarly,

$$p(v|h) = \frac{p(v,h)}{p(h)} = \prod_i p(v_i|h)$$

and

$$p(v_i = 1|h) = \sigma\left(b_i + W_{i,:}h\right)$$

is logistic function

Prisoners' Dilemma

Two suspects in a major crime are held in separate cells. There is enough evidence to convict each of them of a minor offense, but not enough evidence to convict either of them of the major crime unless one of them acts as an informer against the other (defects). If they both stay quiet, each will be convicted of the minor offense and spend one year in prison. If one and only one of them defects, she will be freed and used as a witness against the other, who will spend four years in prison. If they both defect, each will spend three years in prison.

Players: The two suspects.

Actions: Each player's set of actions is {Quiet, Defect}.

Preferences: Suspect 1's ordering of the action profiles, from best to worst, is (Defect, Quiet) (he defects and suspect 2 remains quiet, so he is freed), (Quiet, Quiet) (he gets one year in prison), (Defect, Defect) (he gets three years in prison), (Quiet, Defect) (he gets four years in prison). Suspect 2's ordering is (Quiet, Defect), (Quiet, Quiet), (Defect, Defect), (Defect, Quiet).

Suspect1/ Suspect 2	Quiet	Defect
Quiet	2,2	0,3
Defect	3,0	1,1

3 represents best outcome, 0 worst, etc.

Nash Equilibrium

Let (S, f) be a game with n players, where S_i is the strategy set for player i, $S = S_1 \times S_2 \times \cdots \times S_n$ is the set of strategy profiles and $f(x) = (f_1(x), \ldots, f_n(x))$ is its payoff function evaluated at $x \in S$. Let x_i be a strategy profile of player i and x_{-i} be a strategy profile of all players except for player i. When each player $i \in \{1, \ldots, n\}$ chooses strategy x_i resulting in strategy profile $x = (x_1, \ldots, x_n)$ then player i obtains payoff $f_i(x)$. Note that the payoff depends on the strategy profile chosen, i.e., on the strategy chosen by player i as well as the strategies chosen by all the other players. A strategy profile $x^* \in S$ is a Nash equilibrium (NE) if no unilateral deviation in strategy by any single player is profitable for that player, that is

$$\forall i, x_i \in S_i : f_i(x_i^*, x_{-i}^*) \geq f_i(x_i, x_{-i}^*).$$

Thanks, Wikipedia.

Another Example

		Player 2	
		I	*A*
Player 1	*I*	2,1	0,0
	A	0,0	1,2

Thanks, Prof. Osborne of U. Toronto, Economics

Minimax with Simultaneous Moves

- *maximin* value: largest value player can be assured of *without* knowing other player's actions

$$\underline{v_i} = \max_{a_i} \min_{a_{-i}} v_i(a_i, a_{-i})$$

Where:

- i is the index of the player of interest.
- $-i$ denotes all other players except player i

- *minimax* value: smallest value other players can force this player to receive *without* knowing this player's action

$$\overline{v_i} = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i})$$

- *minimax* is an upper bound on *maximin*

Key Result

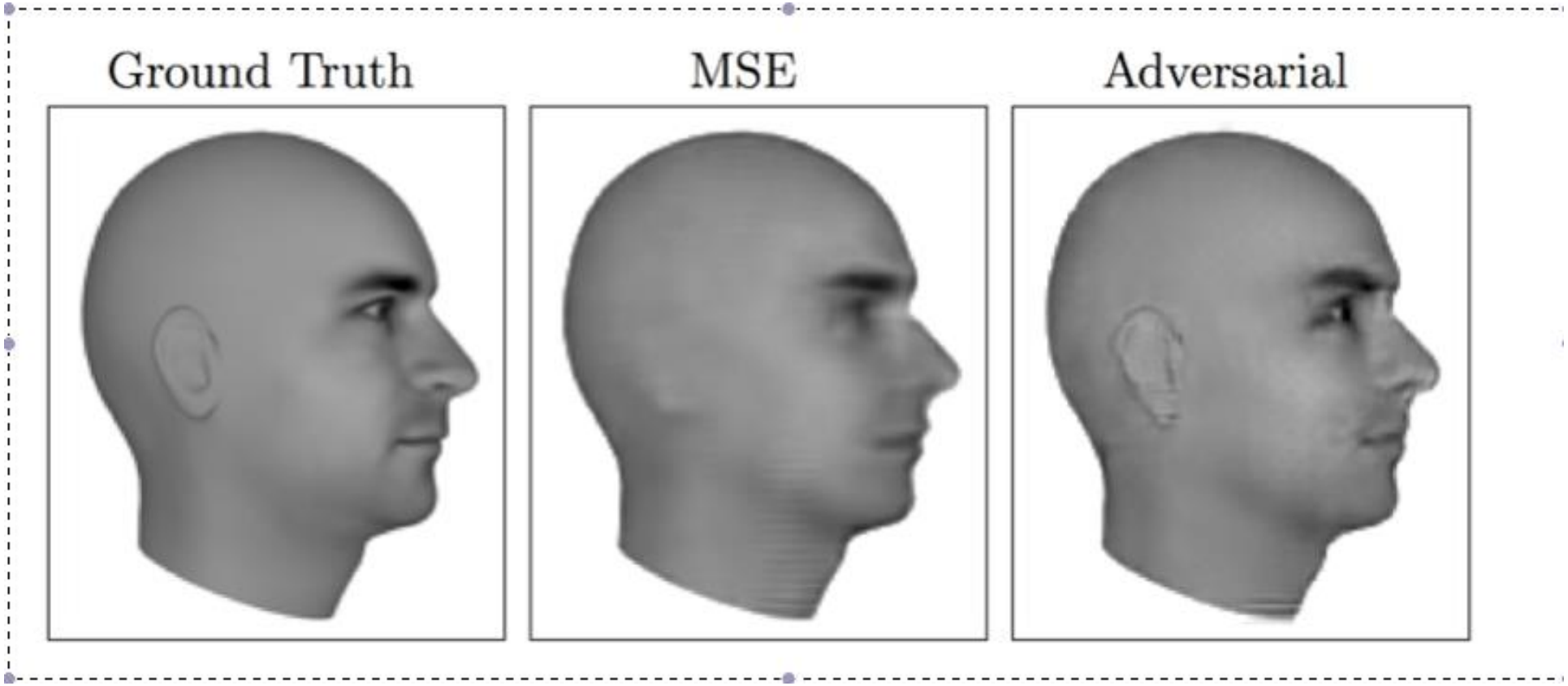
- **Utility**: numeric reward for actions
- **Game**: 2 or more players take turns or take simultaneous actions. Moves lead to states, states have utilities.
- Game is like an optimization problem, but each player tries to maximize own objective function (utility function)
- **Zero-sum game**: each player's gain or loss in utility is exactly balanced by others'
- **In zero-sum game, *Minimax* solution is same as *Nash Equilibrium***

Generative Adversarial Networks

- **Approach:** Set up zero-sum game between deep nets to
  - **Generator:** Generate data that looks like training set
  - **Discriminator:** Distinguish between real and synthetic data

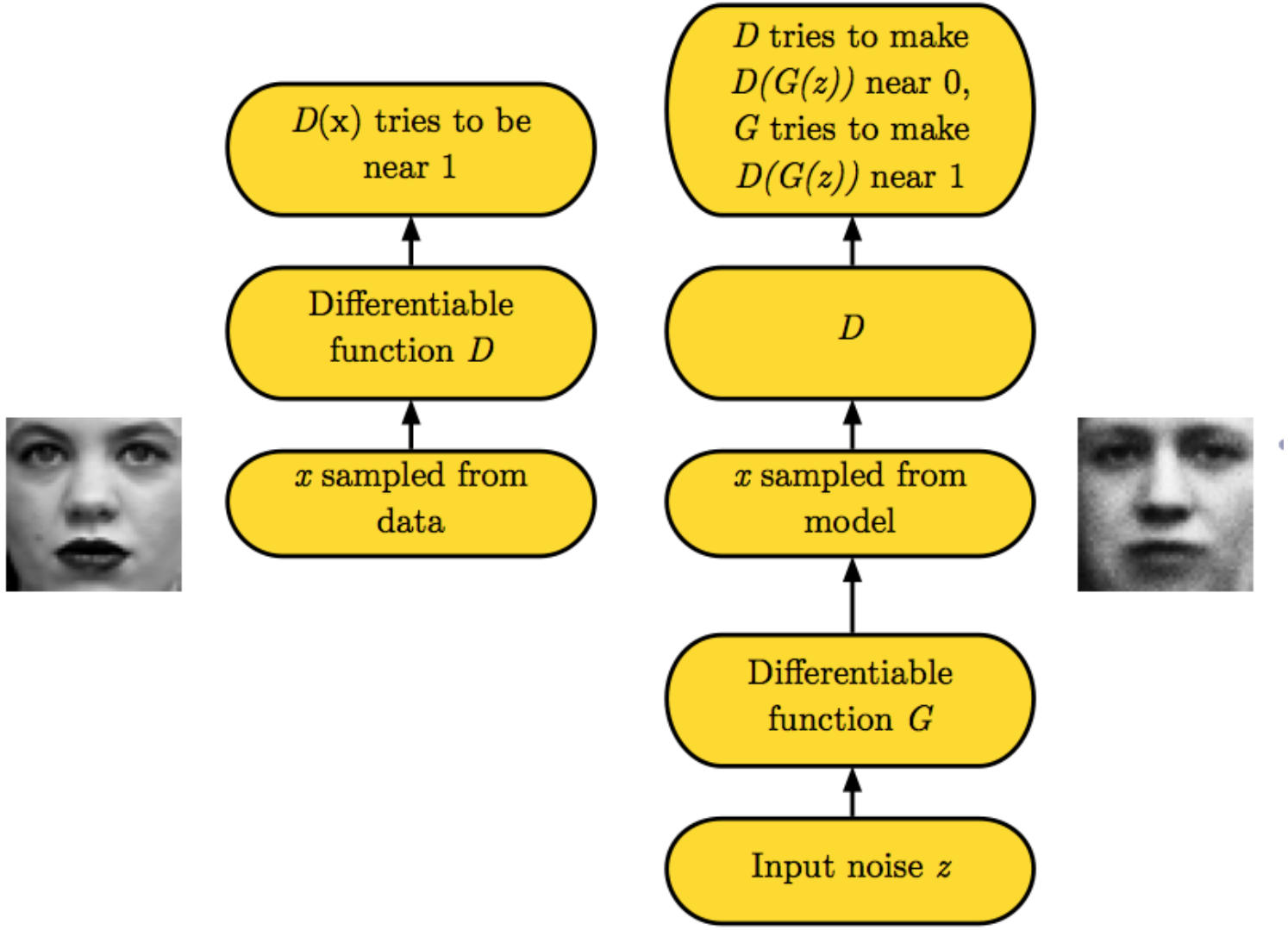
- **Motivation:**
  - Building accurate generative models is hard (e.g., *learning* ***and*** *sampling* from Markov net or Bayes net)
  - Want to use all our great progress on *supervised learners* to do this *unsupervised* learning task better
  - Deep nets may be our favorite supervised learner, especially for image data, if nets are convolutional (use tricks of sliding windows with parameter tying, cross-entropy transfer function, batch normalization)

Does It Work?



Thanks, Ian Goodfellow, NIPS 2016 Tutorial on GANS, for this and most of what follows…

A Bit More on GAN Algorithm



The Rest of the Details

- Use deep convolutional neural networks for Discriminator D and Generator G

- Let **x** denote trainset and **z** denote random, uniform input

- Set up zero-sum game by giving D the following objective, and G the negation of it:

$$-\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}}\log\left(1 - D\left(G(z)\right)\right)$$

- Let D and G compute their gradients simultaneously, each make one step in direction of the gradient, and repeat until neither can make progress… Minimax

Not So Fast

- While preceding version is theoretically elegant, in practice the gradient for G vanishes before we reach best practical solution

- While no longer true Minimax, use same objective for D but change objective for G to:

$$-\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

- Sometimes better if instead of using one minibatch at a time to compute gradient and do batch normalization, we also have a fixed subset of training set, and use combination of fixed subset and current minibatch

Comments on GANs

- Potentially can use our high-powered supervised learners to build better, faster data generators (can they replace MCMC, etc.?)
- While some nice theory based on Nash Equilibria, better results in practice if we move a bit away from the theory
- In general, many in ML community have strong concern that we don’t really understand why deep learning works, including GANs
- Still much research into figuring out why this works better than other generative approaches for some types of data, how we can improve performance further, how to take these from image data to other data types where CNNs might not be the most natural deep network structure