CPSC 340: Machine Learning and Data Mining

Deep Learning

Summer 2021

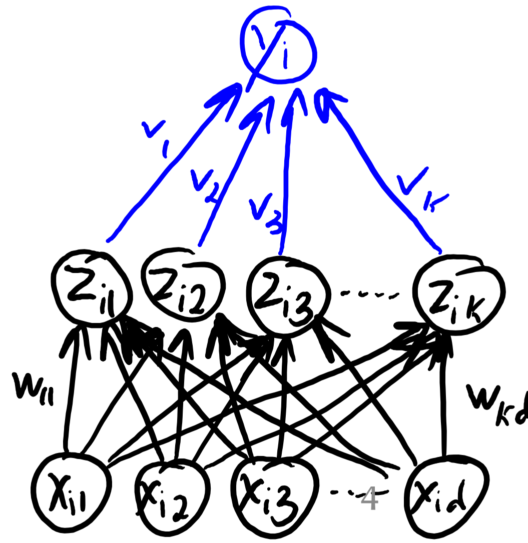
In This Lecture

1. Encoder-Predictor Learning
   - aka deep learning
2. Artificial Neural Networks
3. “Biological” Motivations for Deep Learning
4. History of Deep Learning

Part 5: Deep Learning

Supervised Learning Roadmap

- Part 1: "Direct" Supervised Learning.
  - We learned parameters 'w' based on the original features x_i and target y_i.
- Part 3: Change of Basis.
  - We learned parameters 'v' based on a change of basis z_i and target y_i.
- Part 4: Latent-Factor Models.
  - We learned parameters 'W' for basis z_i based on only on features x_i.
  - You can then learn 'v' based on change of basis z_i and target y_i.
- Part 5: Neural Networks.
  - Jointly learn 'W' and 'v' based on x_i and y_i.
  - **Learn features z_i that is good for supervised learning.**

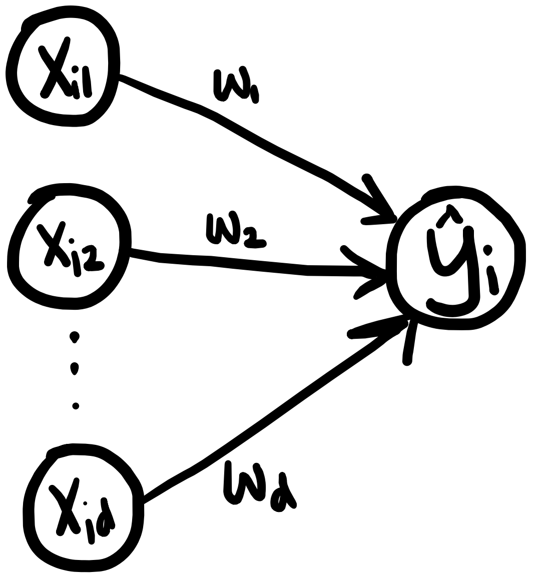


Coming Up Next

ENCODER-PREDICTOR LEARNING

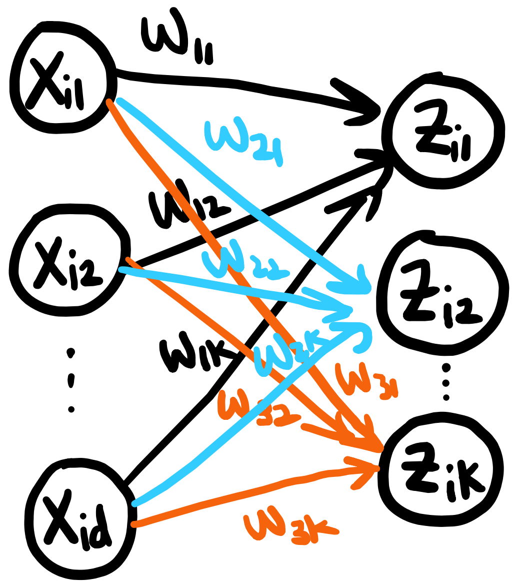
“Graph” View of Matrix Multiplication

$$\underbrace{\hat{y}_i}_{\text{"target" node}} = \underbrace{w^T}_{\text{edge weights}} \underbrace{x_i}_{\text{"input" nodes}}$$



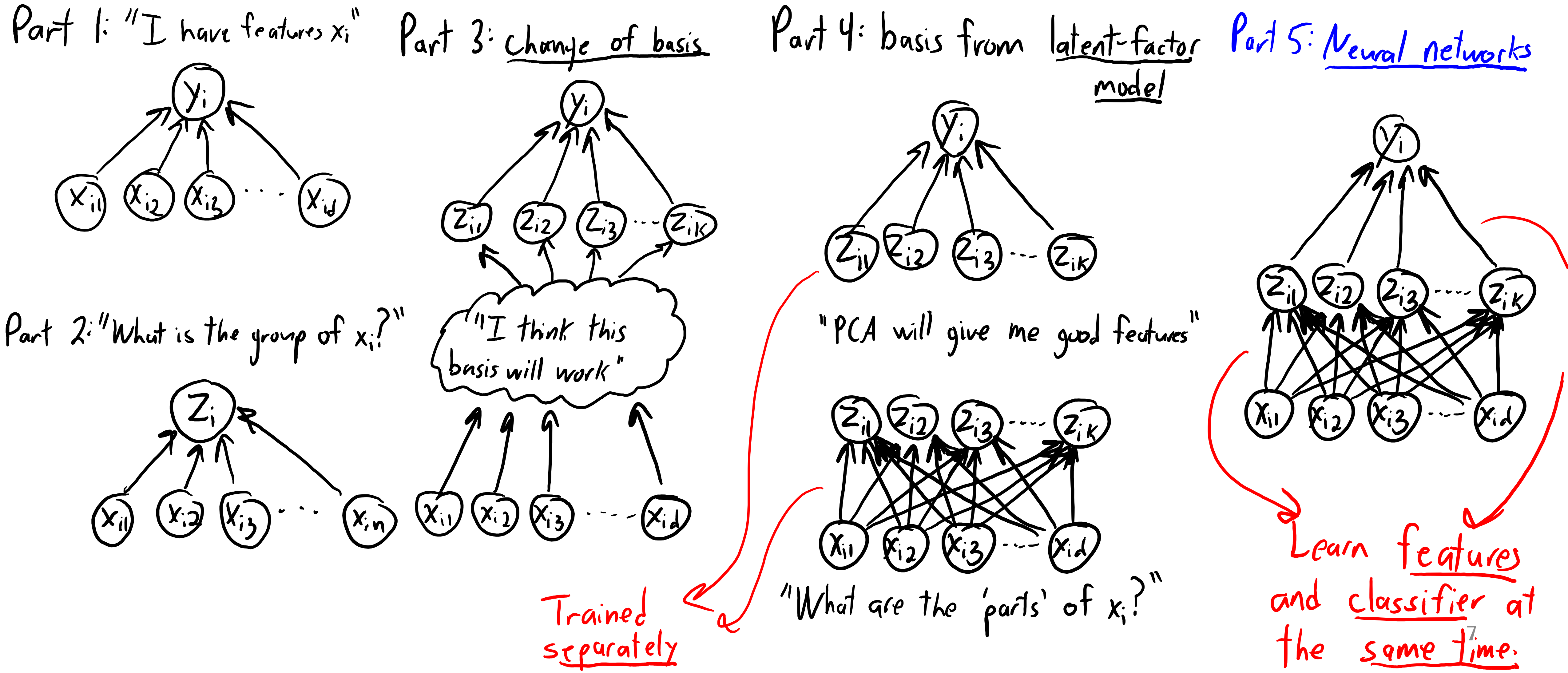
$$\hat{y}_i = [w_1\ w_2 \cdots w_d] \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}$$

$$\underbrace{z_i}_{\text{"output" nodes}} = \underbrace{W}_{\text{edge weights}} \underbrace{x_i}_{\text{"input" nodes}}$$

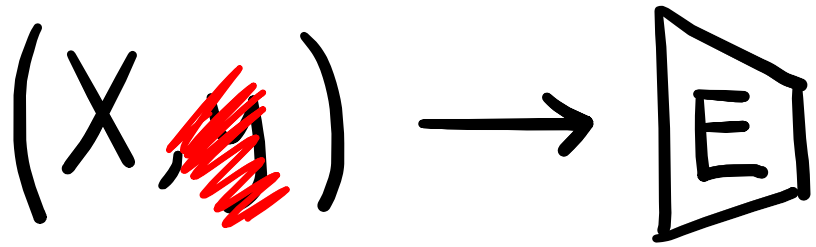


$$\begin{bmatrix} z_{i1} \\ z_{i2} \\ \vdots \\ z_{ik} \end{bmatrix} = {}_{k}\underbrace{\begin{bmatrix} -w_1- \\ -w_2- \\ \vdots \\ -w_k- \end{bmatrix}}_{d} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}$$

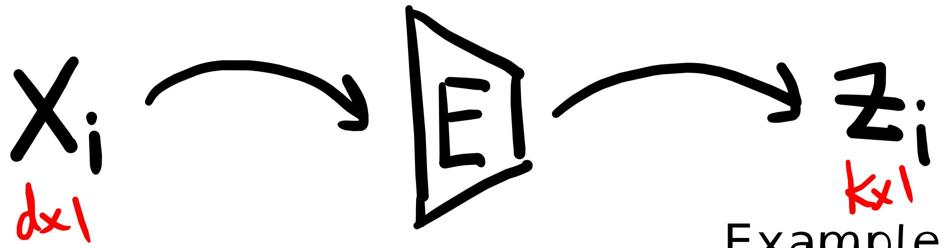
A Graphical Summary of CPSC 340 Parts 1-5


Part 1: "I have features x_i"
y_i
x_{i1} x_{i2} x_{i3} ... x_{id}
Part 2: "What is the group of x_i?"
z_i
x_{i1} x_{i2} x_{i3} ... x_{in}
Part 3: Change of basis
y_i
z_{i1} z_{i2} z_{i3} ... z_{ik}
"I think this basis will work"
x_{i1} x_{i2} x_{i3} ... x_{id}
Part 4: basis from latent-factor model
y_i
z_{i1} z_{i2} z_{i3} ... z_{ik}
"PCA will give me good features"
z_{i1} z_{i2} z_{i3} ... z_{ik}
x_{i1} x_{i2} x_{i3} ... x_{id}
"What are the 'parts' of x_i?"
Trained separately
Part 5: Neural networks
y_i
z_{i1} z_{i2} z_{i3} ... z_{ik}
x_{i1} x_{i2} x_{i3} ... x_{id}
Learn features and classifier at the same time.

Recall: Encoder Learning



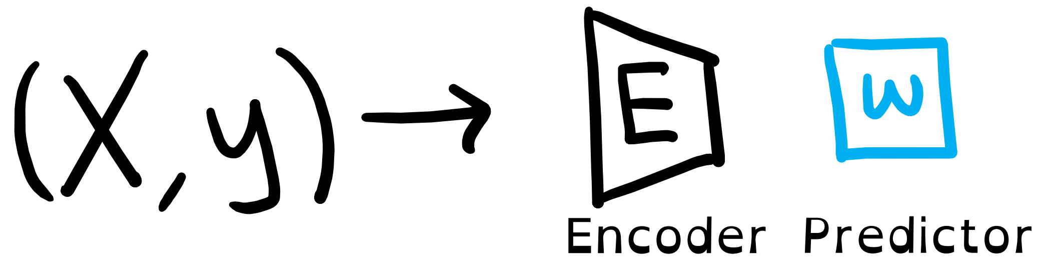
Learned encoder



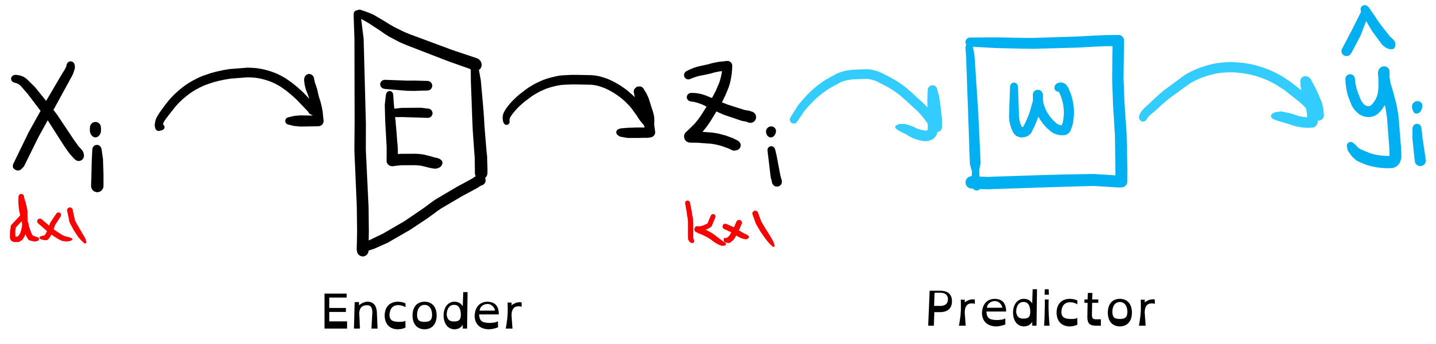
Example i in
learned feature space

"Encoder-Predictor Learning"

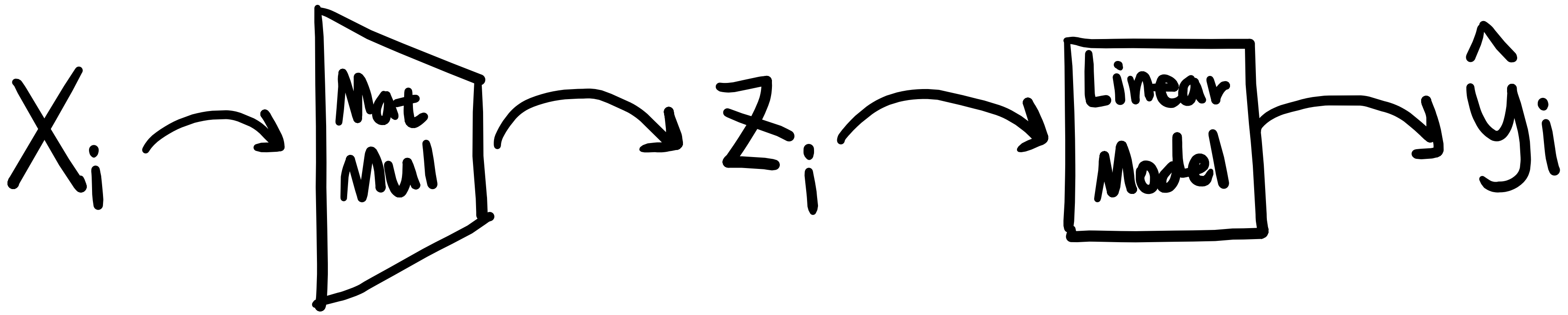
- Encoder-Predictor learning problem:
  - Input: Labeled examples
  - Output: Encoder E and predictor w



- Using learned encoder and predictor:

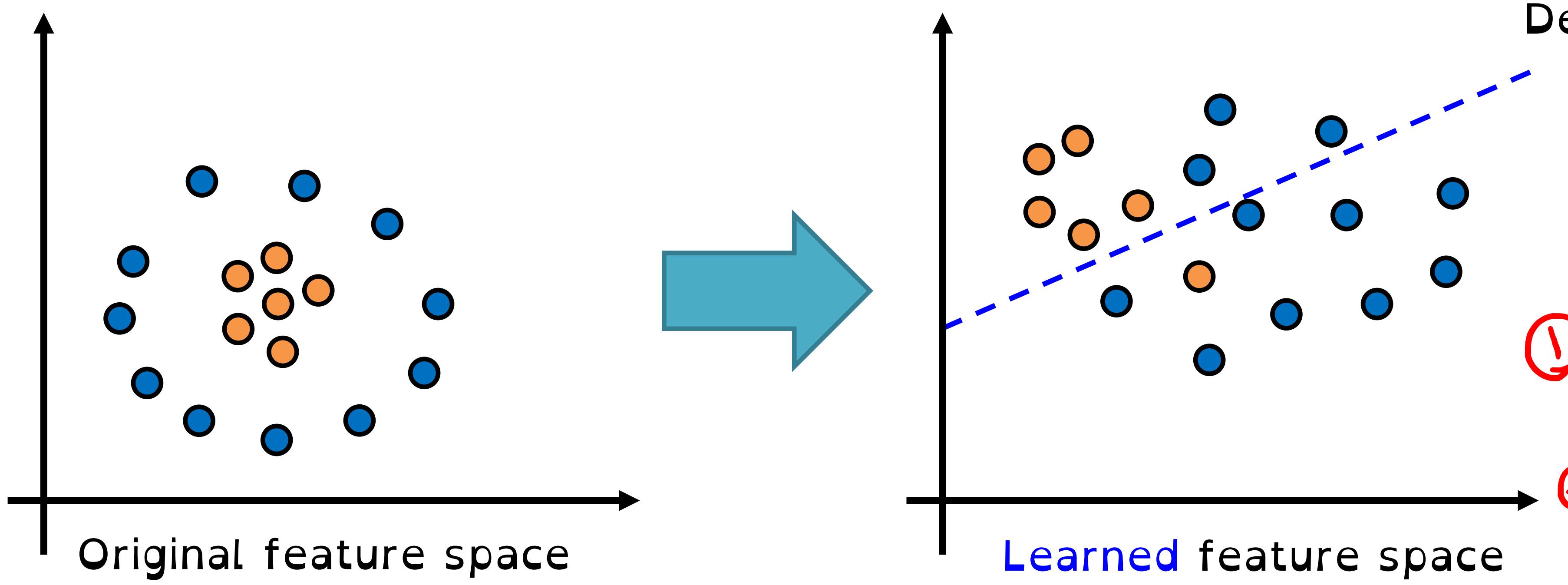


"Artificial Neural Networks"



- "Artificial neural network" := encoder-predictor model using matrix multiplication for encoder
  - Must use non-linear activations (soon)
  - Usually use linear model as predictor
- "Deep neural network" := artificial neural network that uses more than one matrix multiplication

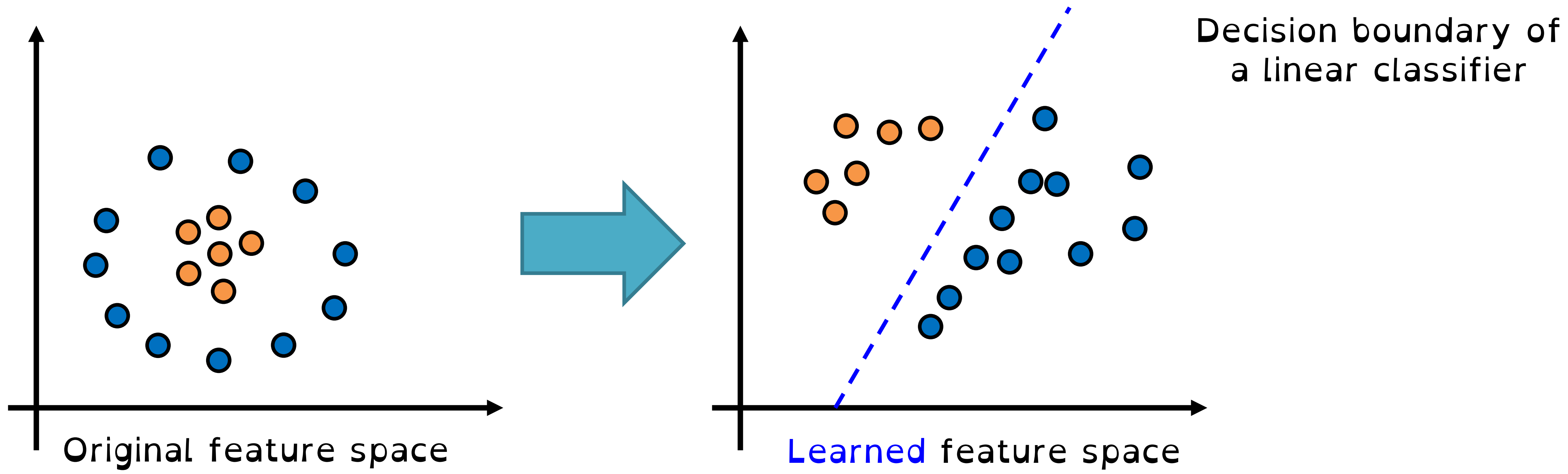
Visualizing Encoder-Predictor



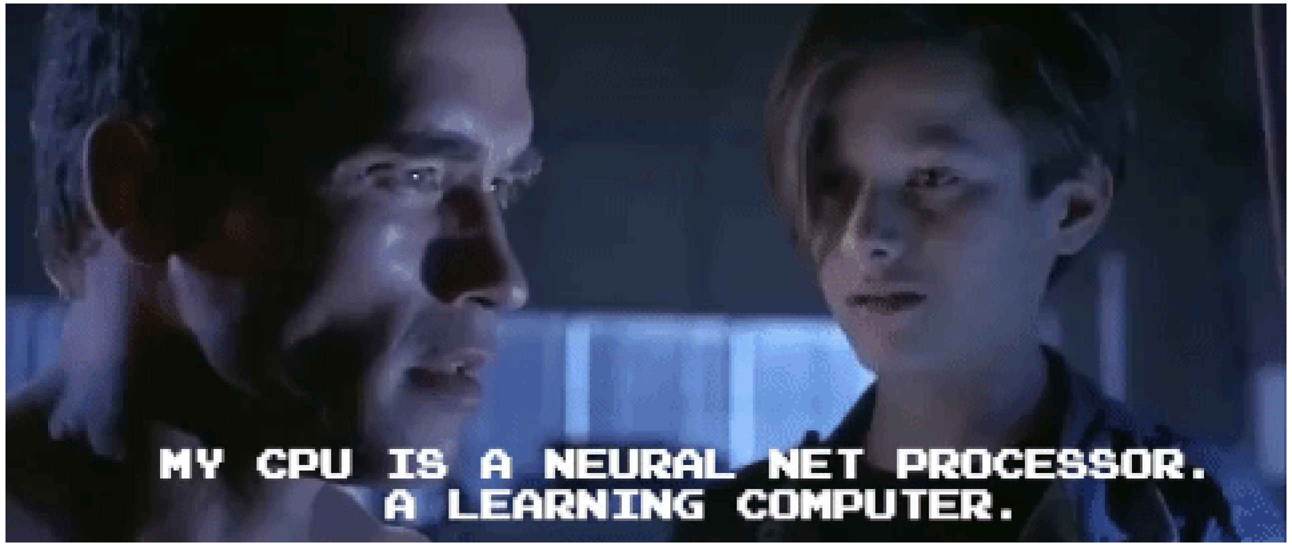
① learn better feature space
② learn better classifier

Q: How can we make our model better than this?

Visualizing Encoder-Predictor



- Intuition: supervised latent factor model
  - Loss function based on labels
  - Encourage encoder to produce more linearly separable results



Coming Up Next

MORE FORMAL DETAILS ON NEURAL NETWORKS

Notation for Neural Networks (MEMORIZE)

We have our usual supervised learning notation:

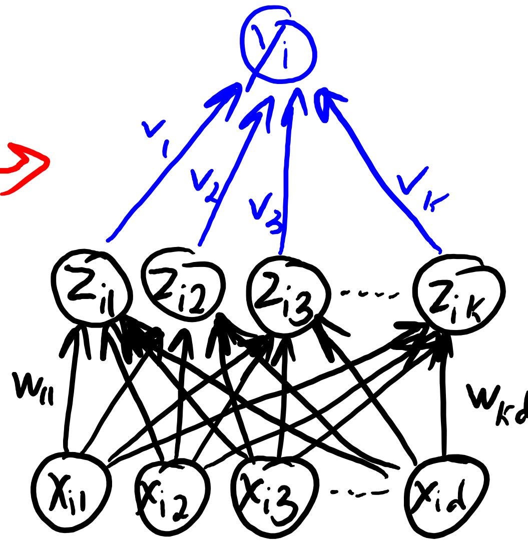
$$X = \underbrace{\begin{bmatrix} — x_1^T — \\ — x_2^T — \\ \vdots \\ — x_n^T — \end{bmatrix}}_{n \times d} \qquad y = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{n \times 1}$$

We have our latent features:

$$Z = \underbrace{\begin{bmatrix} — z_1^T — \\ — z_2^T — \\ \vdots \\ — z_n^T — \end{bmatrix}}_{n \times k}$$

We have two sets of parameters:

$$v = \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}}_{k \times 1} \qquad W = \underbrace{\begin{bmatrix} — w_1 — \\ — w_2 — \\ \vdots \\ — w_k — \end{bmatrix}}_{k \times d}$$



Linear-Linear Neural Net

- Obvious choice: linear latent-factor encoder with linear regression predictor

Use features from latent-factor model: $z_i = Wx_i$

Make predictions using a linear model: $y_i = v^T z_i$

- We want to train 'W' and 'v' jointly, so we could minimize:

$$f(W,v) = \frac{1}{2}\sum_{i=1}^{n} \underbrace{(v^T z_i - y_i)^2}_{\text{linear regression with } z_i \text{ as features}} = \frac{1}{2}\sum_{i=1}^{n}(v^T \underbrace{(Wx_i)}_{z_i \text{ come from latent-factor model}} - y_i)^2$$

Q: What can go wrong with this?

Linear-Linear Neural Net

- Obvious choice: linear latent-factor encoder with linear regression predictor

Use features from latent-factor model: $z_i = Wx_i$

Make predictions using a linear model: $y_i = v^T z_i$

- We want to train 'W' and 'v' jointly, so we could minimize:

$$f(W,v) = \frac{1}{2}\sum_{i=1}^{n} \underbrace{(v^T z_i - y_i)^2}_{\text{linear regression with } z_i \text{ as features}} = \frac{1}{2}\sum_{i=1}^{n}(v^T \underbrace{(Wx_i)}_{z_i \text{ come from latent-factor model}} - y_i)^2$$

- This is just a linear model:

$$\hat{y}_i = v^T z_i = v^T (Wx_i) = \underbrace{\overbrace{(v^T W)}^{1 \times d}}_{\text{some vector 'w'}} x_i = w^T x_i$$

Introducing Non-Linearity

- To increase flexibility, something needs to be non-linear.
- Typical choice: transform z_i by non-linear function 'h'.

$$z_i = Wx_i \qquad y_i = v^T h(z_i)$$

  - Here the function 'h' transforms 'k' inputs to 'k' outputs.
- Common choice for 'h': applying sigmoid function element-wise:

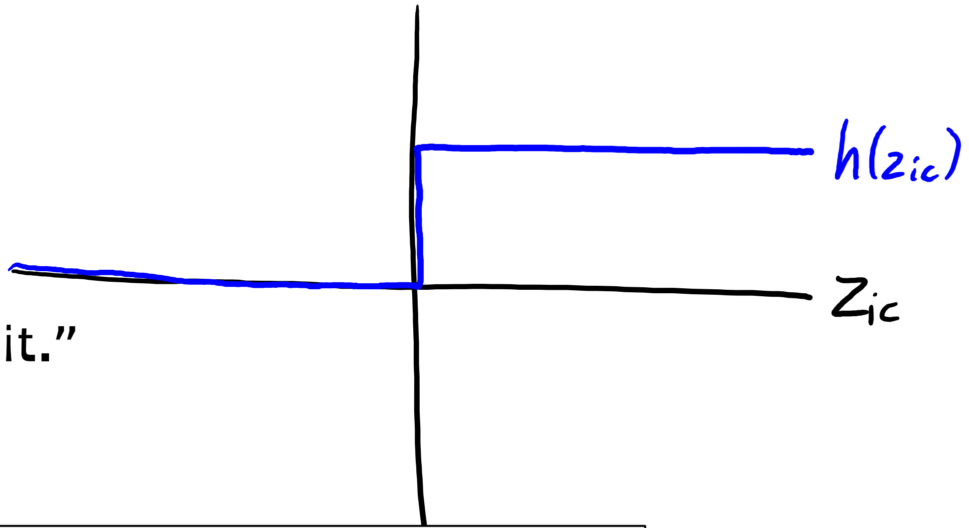
$$h(z_{ic}) = \frac{1}{1 + \exp(-z_{ic})}$$

- So this takes the z_{ic} in $(-\infty,\infty)$ and maps it to $\underline{(0,1)}$.

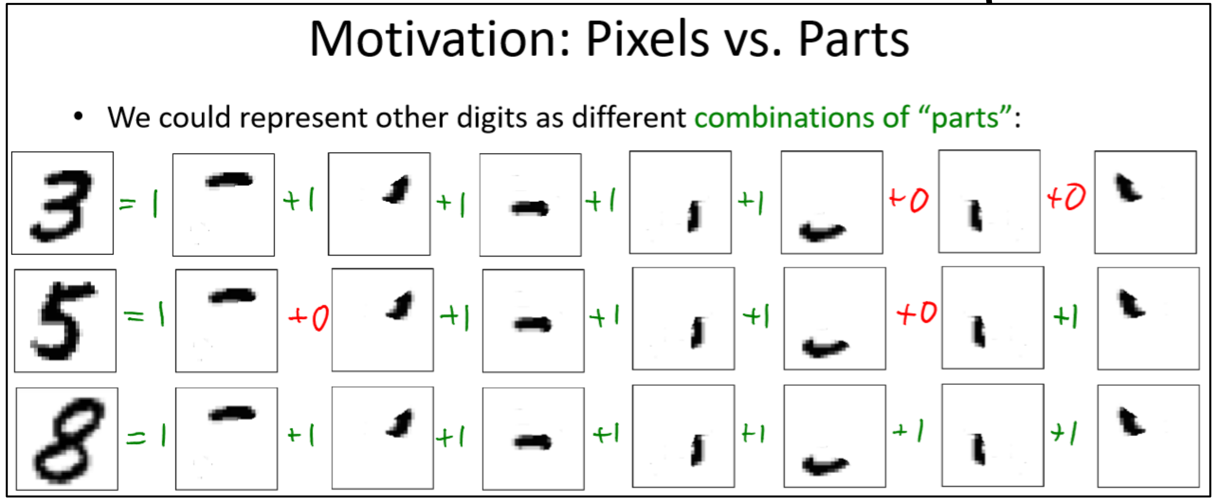
Why Sigmoid?

- Consider setting ‘h’ to define binary features z_i using:

$$h(z_{ic}) = \begin{cases} 1 & \text{if } z_{ic} \geq 0 \\ 0 & \text{if } z_{ic} < 0 \end{cases}$$



  - Each $h(z_i)$ can be viewed as binary feature.
    - “You either have this ‘part’ or you don’t have it.”
  - We can make 2^k objects by all the possible “part combinations”.



Motivation: Pixels vs. Parts

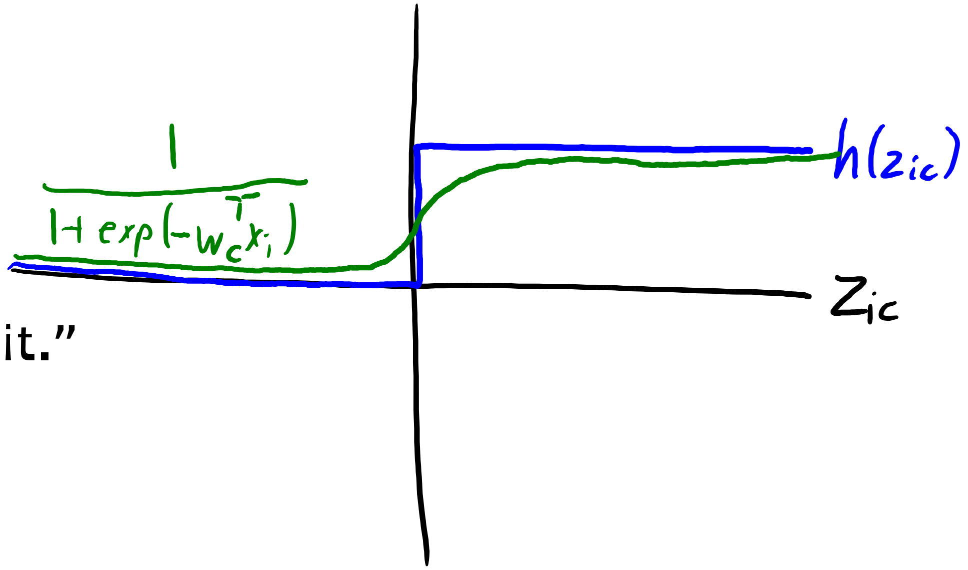
- We could represent other digits as different combinations of “parts”:

3 = 1 +1 +1 +1 +1 +0 +0
5 = 1 +0 +1 +1 +1 +0 +1
8 = 1 +1 +1 +1 +1 +1 +1

Why Sigmoid?

- Consider setting 'h' to define binary features z_i using:

$$h(z_{ic}) = \begin{cases} 1 & \text{if } z_{ic} \geq 0 \\ 0 & \text{if } z_{ic} < 0 \end{cases}$$



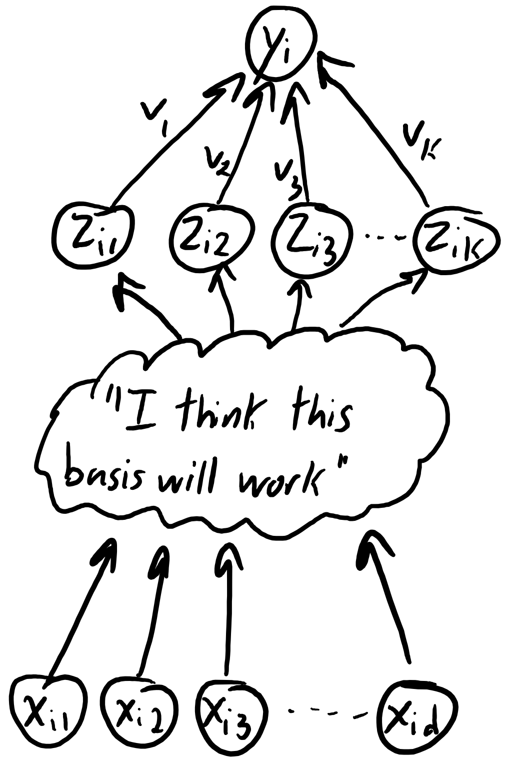
  - Each $h(z_i)$ can be viewed as binary feature.
    - "You either have this 'part' or you don't have it."

$\underset{n \times k}{Z}$

- But this is hard to optimize (non-differentiable, discontinuous).
- Sigmoid is a smooth approximation to these binary features.
  - Non-parametric version is a universal approximator:
    - If 'k' grows appropriately with 'n', can model any continuous function.

Supervised Learning Roadmap

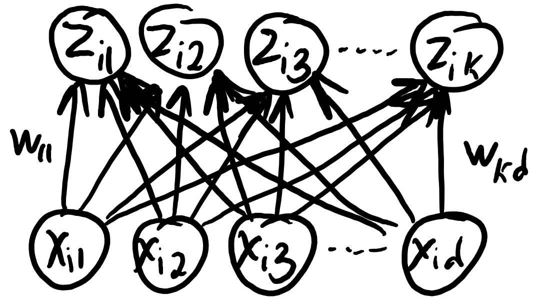
Hand-engineered features:



y_i
v_1
v_2
v_3
v_k
z_{i1} z_{i2} z_{i3} .... z_{ik}
"I think this basis will work"
x_{i1} x_{i2} x_{i3} .... x_{id}

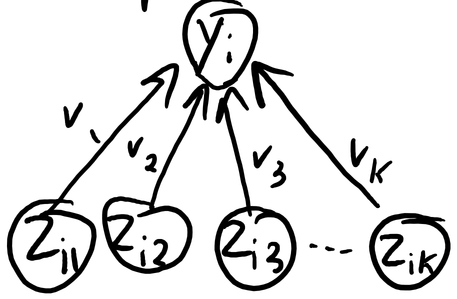
Requires domain knowledge and can be time-consuming

Learn a latent-factor model:



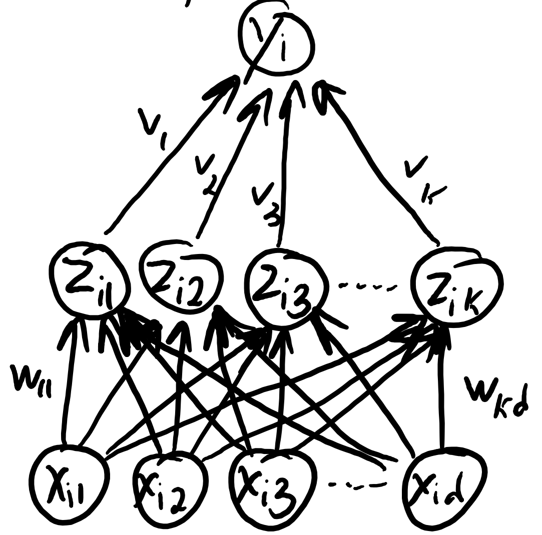
z_{i1} z_{i2} z_{i3} .... z_{ik}
w_{11}
w_{kd}
x_{i1} x_{i2} x_{i3} .... x_{id}

Use latent features in supervized model:



Good representation of x_i might be bad for predicting y_i

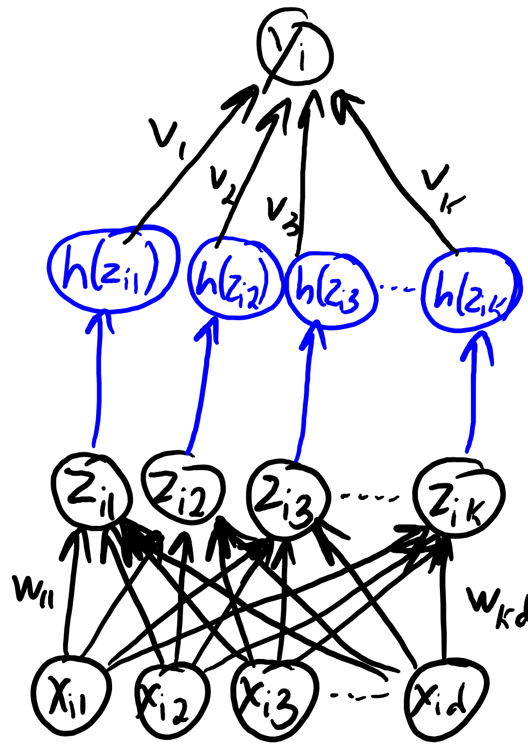
Learn 'w' and 'W' together:



y_i
v_1
v_2
v_3
v_k
z_{i1} z_{i2} z_{i3} .... z_{ik}
w_{11}
w_{kd}
x_{i1} x_{i2} x_{i3} .... x_{id}

But still gives a <u>linear</u> model.

Neural network:



y_i
v_1
v_2
v_3
v_k
$h(z_{i1})$ $h(z_{i2})$ $h(z_{i3})$ ... $h(z_{ik})$
z_{i1} z_{i2} z_{i3} .... z_{ik}
w_{11}
w_{kd}
x_{i1} x_{i2} x_{i3} .... x_{id}

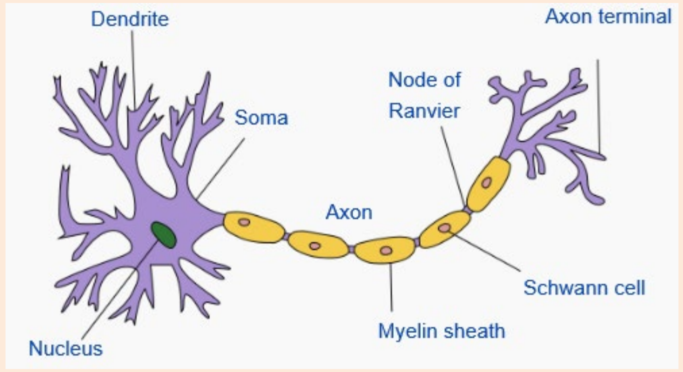
Extra <u>non-linear</u> transformation 'h'!

Coming Up Next

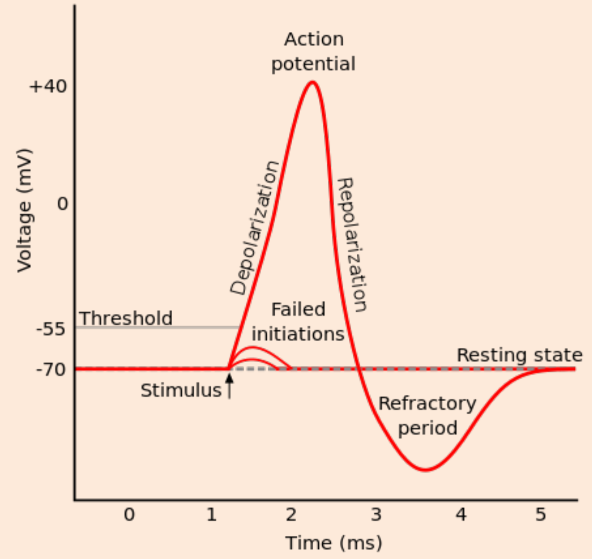
(SUPPOSEDLY) BIOLOGICAL MOTIVATION FOR ARTIFICIAL NEURAL NETWORKS

Why “Neural Network”?

- Cartoon of “typical” neuron:

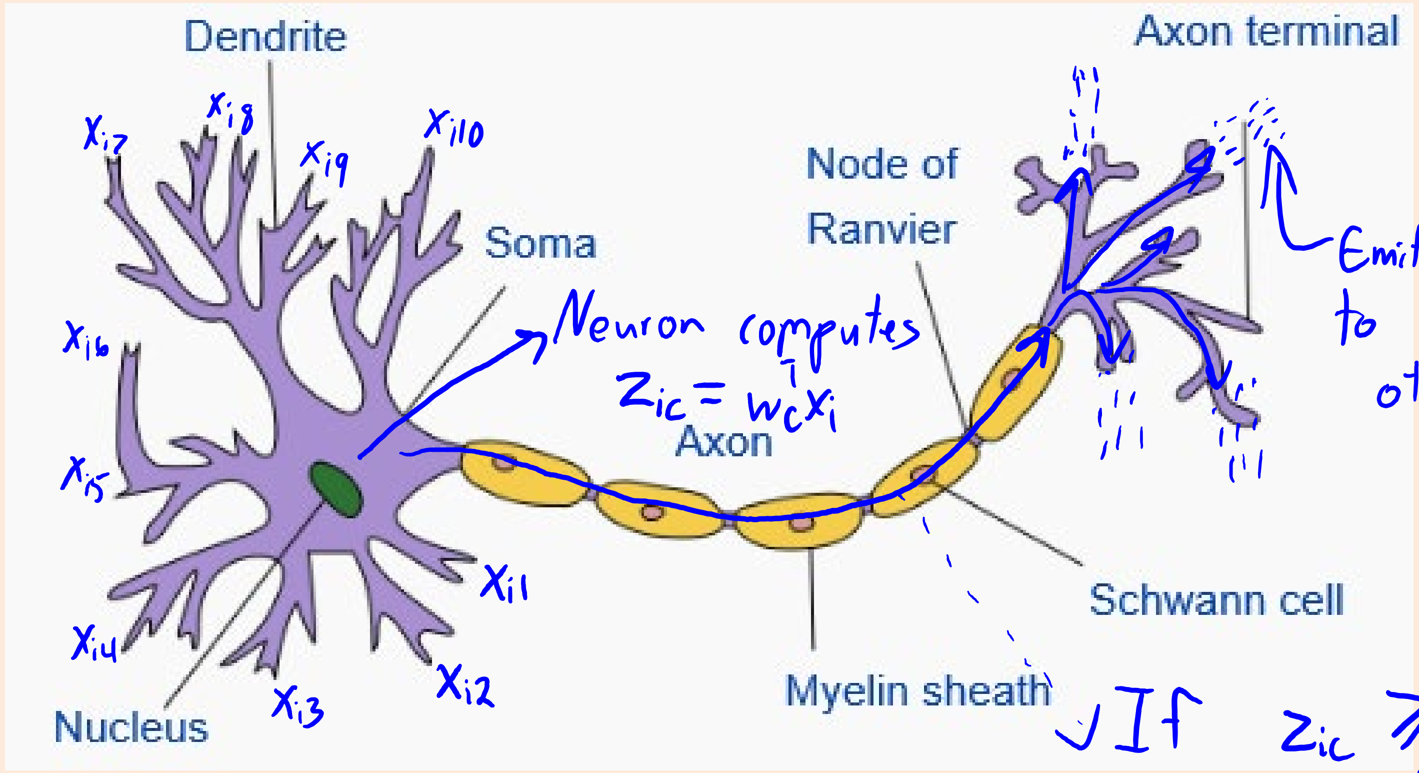


- Neuron has many “dendrites”, which take an input signal.
- Neuron has a single “axon”, which sends an output signal.
- With the right input to dendrites:
  - “Action potential” along axon (like a binary signal):



https://en.wikipedia.org/wiki/Neuron
https://en.wikipedia.org/wiki/Action_potential

Why "Neural Network"?

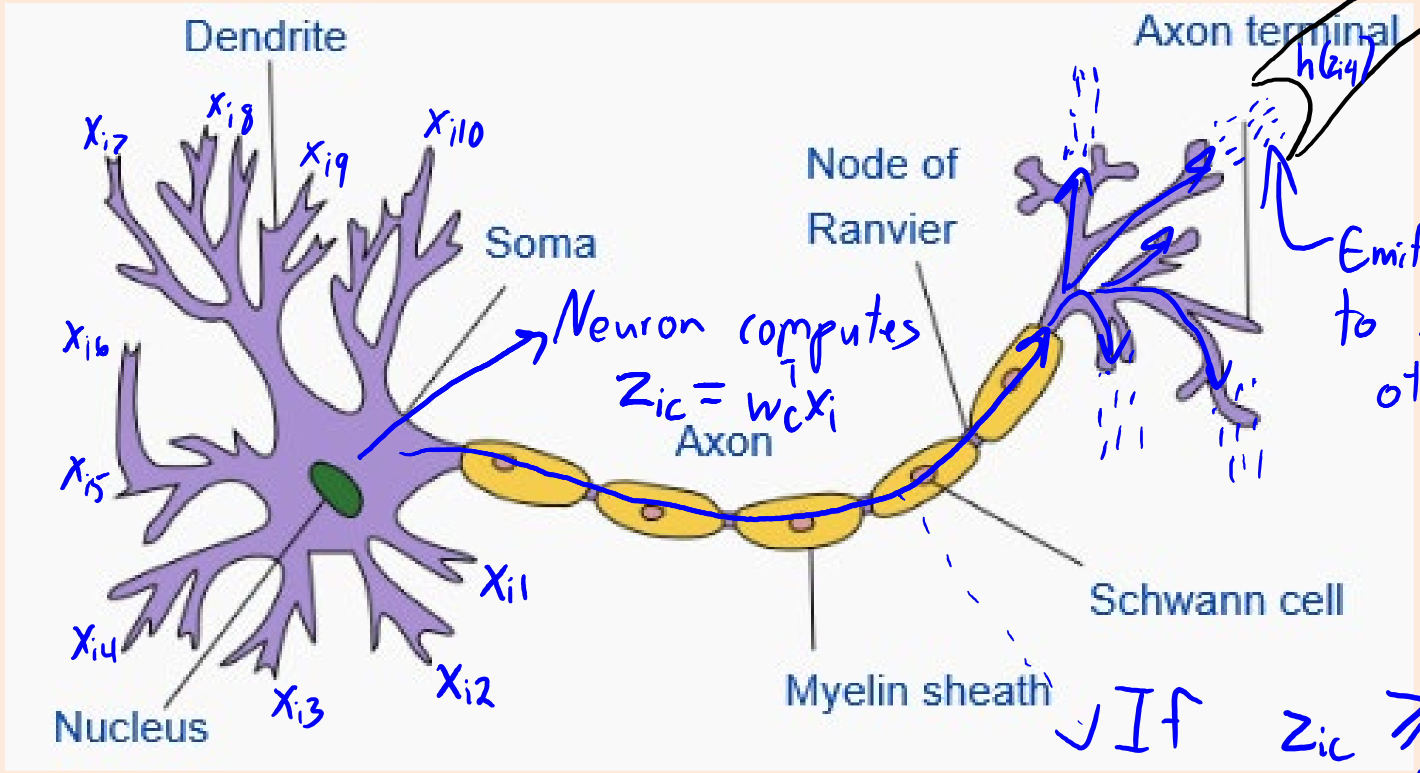


Emits "neurotransmitter" to send signal to other neurons.

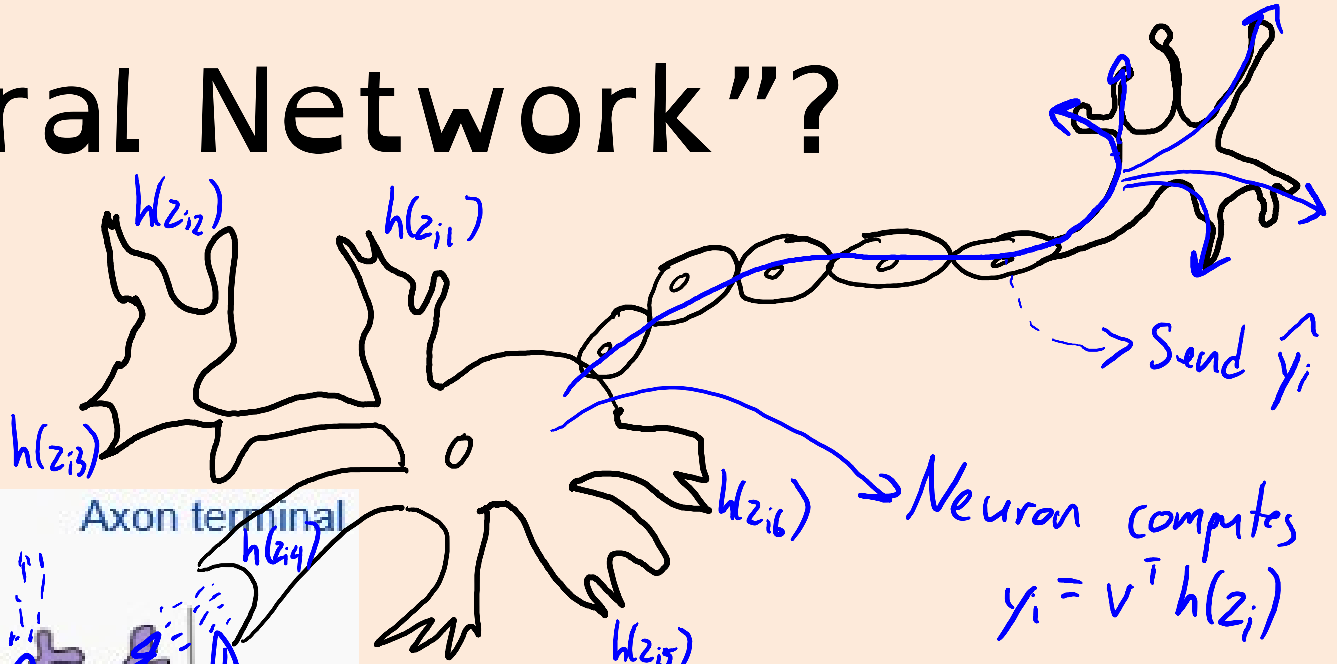
If $z_{ic} \geq 0.5$ neuron sends signal along axon. } We approximate binary signal with $\frac{1}{1+\exp(-z_{ic})}$

https://en.wikipedia.org/wiki/Neuron

Why "Neural Network"?



Neuron computes $z_{ic} = w_c^T x_i$



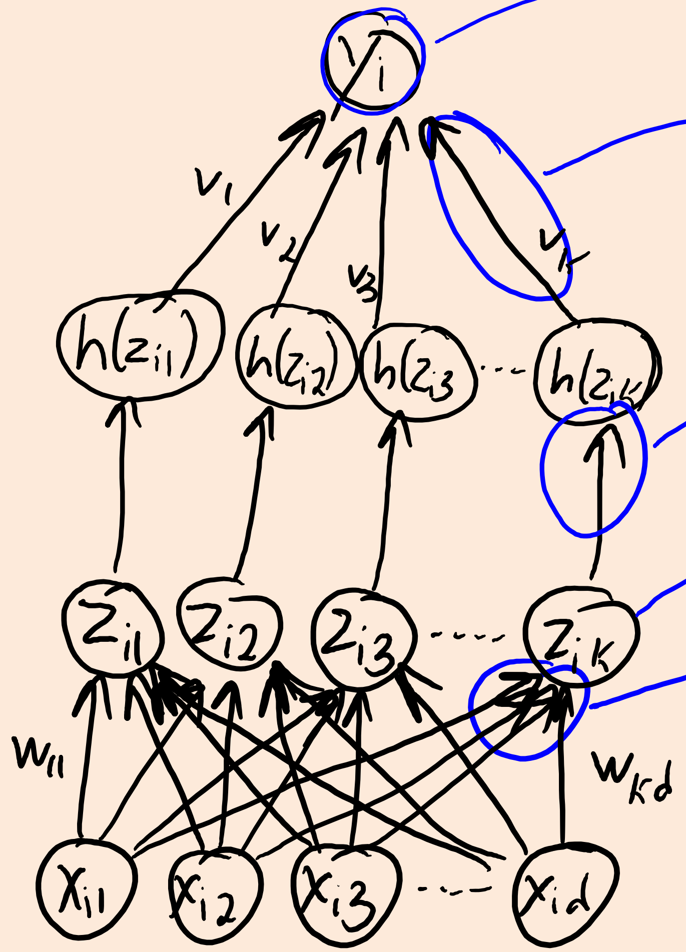
Neuron computes $y_i = v^T h(z_i)$

Emits "neurotransmitter" to send signal to other neurons.

If $z_{ic} \geq 0.5$ neuron sends signal along axon. $\Bigg\}$ We approximate binary signal with $\frac{1}{1+\exp(-z_{ic})}$

https://en.wikipedia.org/wiki/Neuron

Why "Neural Network"?



→ Predictions based on aggregation $v^T h(Wx_i)$ at y_i "neuron"

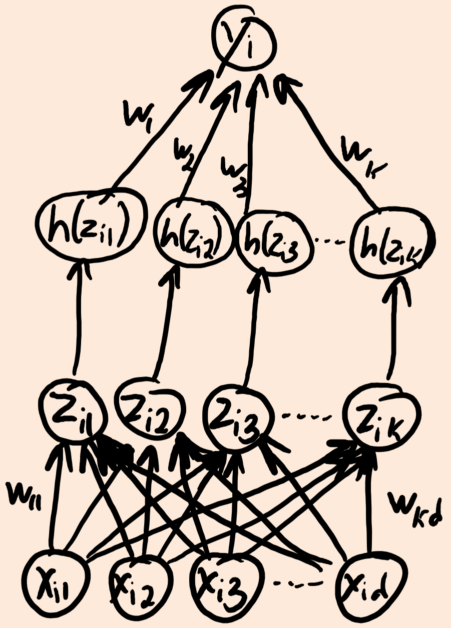
→ Synapse between z_{ik} and y_i "neuron"

→ binary signal $h(w_c^T x_i)$ sent along "axon"

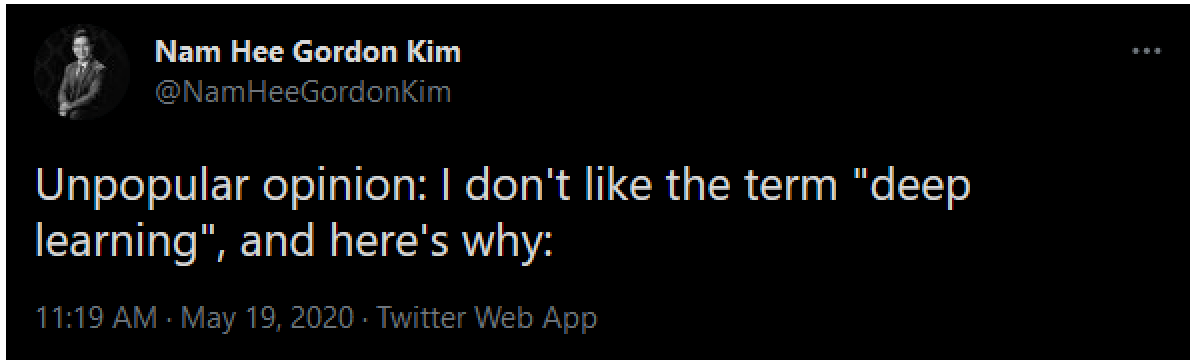
→ Neuron aggregates signals: $w_c^T x_i$

→ "dendrites" for z_{ik} "neuron" are reciving x_{ij} values

"Artificial" Neural Nets vs. "Real" Networks Nets



- Artificial neural network:
  - x_i is measurement of the world.
  - z_i is internal representation of world.
  - y_i is output of neuron for classification/regression.
- Real neural networks are more complicated:
  - Timing of action potentials seems to be important.
    - "Rate coding": frequency of action potentials simulates continuous output.
  - Sparsity of action potentials.
  - How much computation is done inside neuron?
  - Brain is highly organized (e.g., substructures and cortical columns).
  - Connection structure changes.
  - Different types of neurotransmitters.



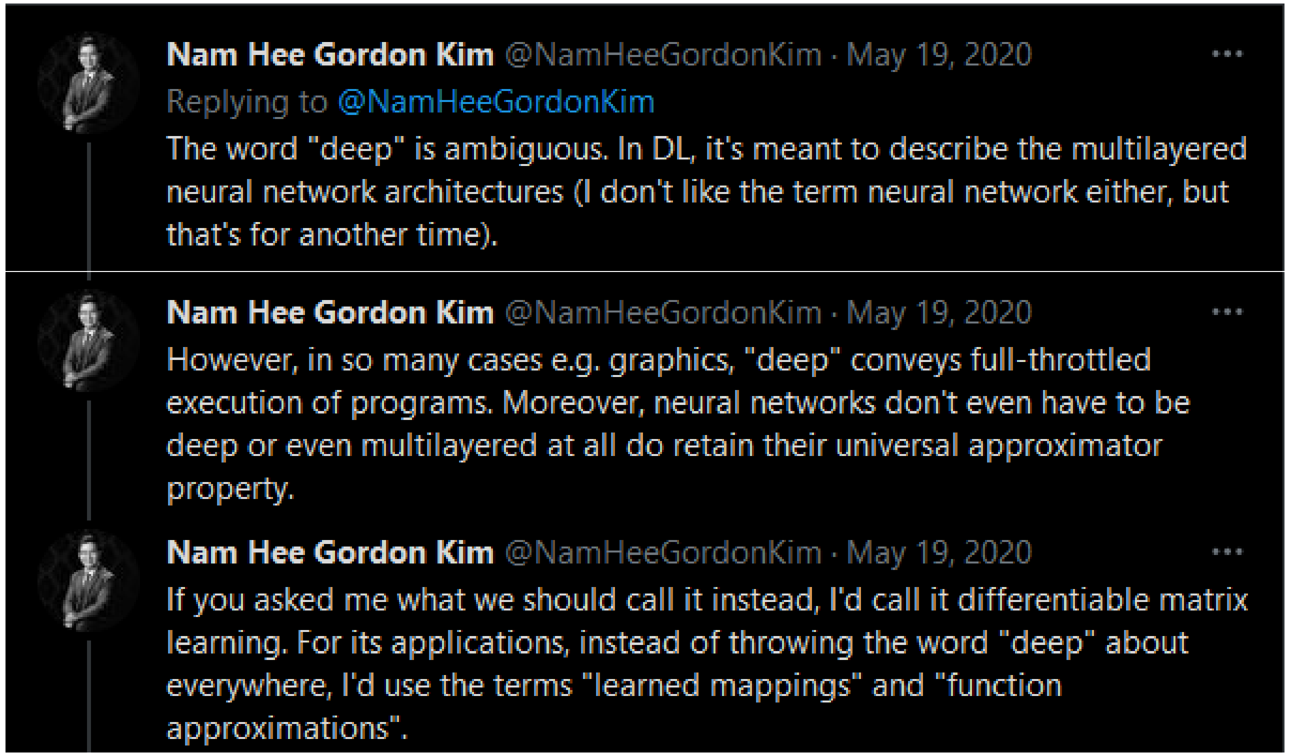
Nam Hee Gordon Kim
@NamHeeGordonKim

Unpopular opinion: I don't like the term "deep learning", and here's why:

11:19 AM · May 19, 2020 · Twitter Web App

Coming Up Next

WHAT IS DEEP LEARNING?

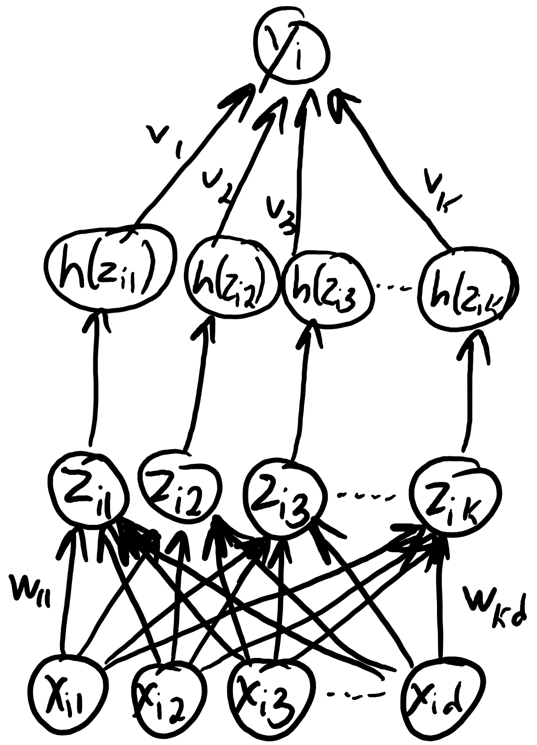

Nam Hee Gordon Kim @NamHeeGordonKim · May 19, 2020
Replying to @NamHeeGordonKim
The word "deep" is ambiguous. In DL, it's meant to describe the multilayered neural network architectures (I don't like the term neural network either, but that's for another time).

Nam Hee Gordon Kim @NamHeeGordonKim · May 19, 2020
However, in so many cases e.g. graphics, "deep" conveys full-throttled execution of programs. Moreover, neural networks don't even have to be deep or even multilayered at all do retain their universal approximator property.

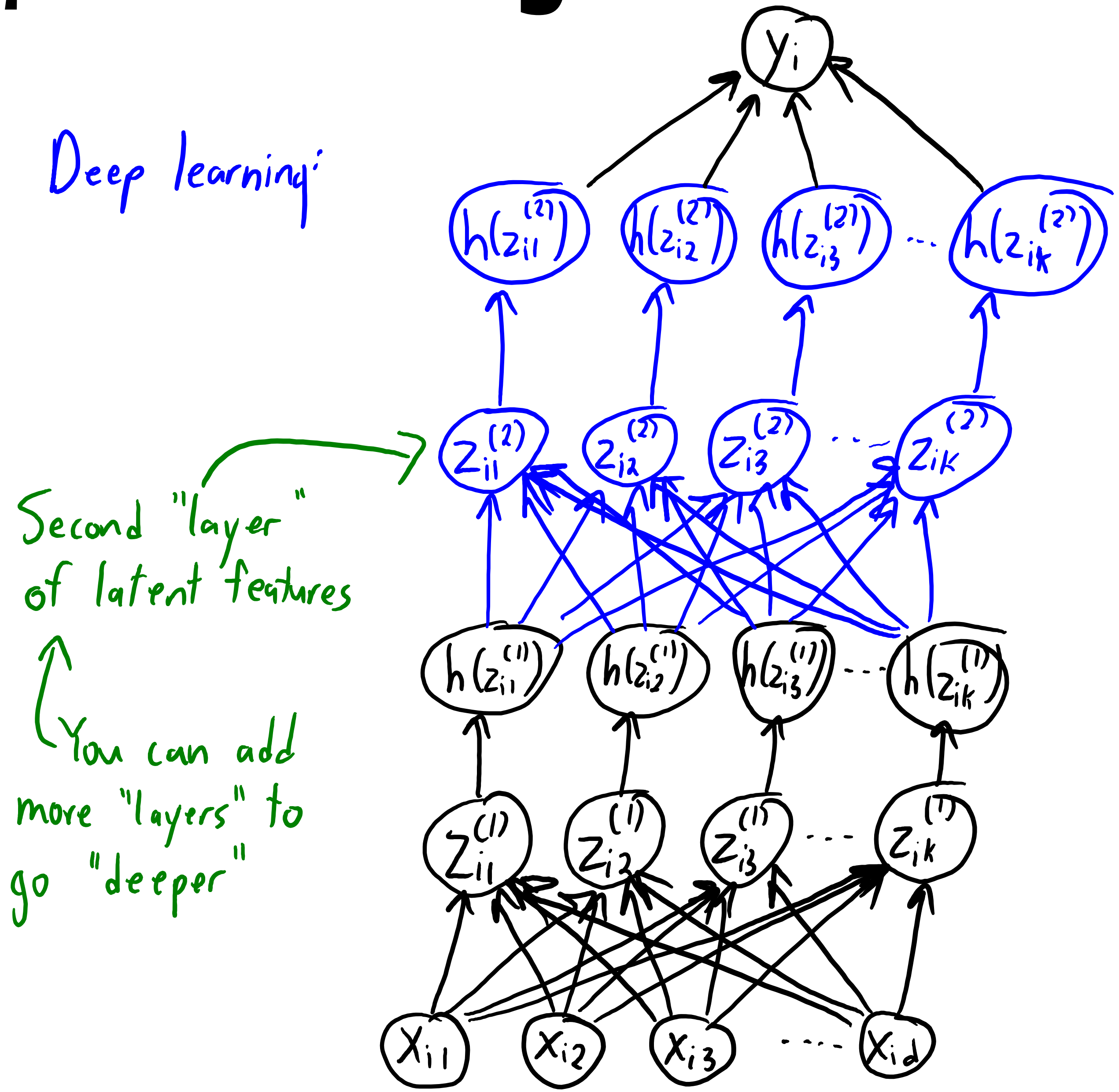
Nam Hee Gordon Kim @NamHeeGordonKim · May 19, 2020
If you asked me what we should call it instead, I'd call it differentiable matrix learning. For its applications, instead of throwing the word "deep" about everywhere, I'd use the terms "learned mappings" and "function approximations".

Deep Learning

Neural network:



Deep learning:



Encoder-Predictor View of Deep Learning

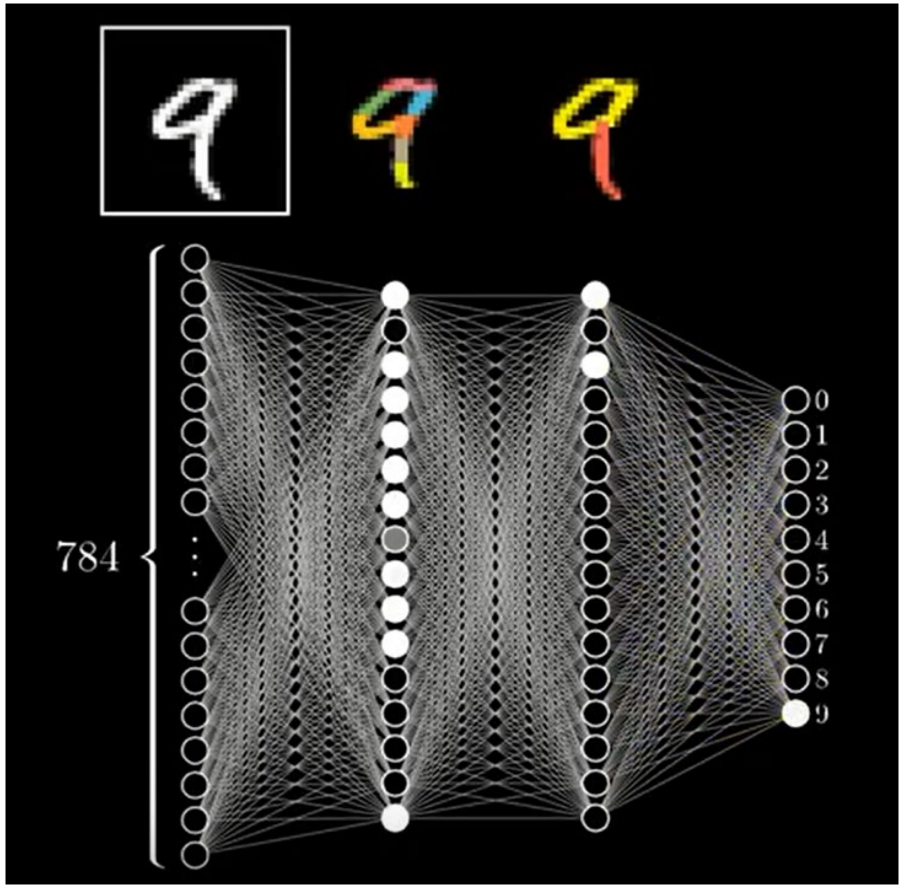
$$\begin{tikzcd}X_i \arrow[r, bend left] & \boxed{\text{Mat Mul}} \arrow[r, bend left] & z_i \arrow[r, bend left] & \boxed{\text{Linear Model}} \arrow[r, bend left] & \hat{y}_i\end{tikzcd}$$

$$\begin{tikzcd}X_i \arrow[r, "\text{matmul}", "\text{activate}"'] & h_i^{(1)} \arrow[r, "\text{matmul}", "\text{activate}"'] & h_i^{(2)} \arrow[r, "\text{matmul}", "\text{activate}"'] & \cdots \arrow[r, "\text{matmul}", "\text{activate}"'] & h_i \arrow[r, "\text{predict}"] & \hat{y}_i\end{tikzcd}$$

- Compose multiple non-linear encoders
- Overall idea is still the same:
  - Train encoder and predictor at the same time
    - (we have a “bigger” encoder now)

"Hierarchies of Parts" Motivation for Deep Learning

- Each "neuron" might recognize a "part" of digit.
  - "Deeper" neurons might recognize combinations of parts.
  - Represent complex objects as hierarchical combinations of re-useable parts (a simple "grammar").



- Watch the full video here:
  - https://www.youtube.com/watch?v=aircAruvnKk

- Theory:
  - 1 big-enough hidden layer already gives universal approximation.
  - But some functions require exponentially-fewer parameters to approximate with more layers (can fight curse of dimensionality).

Deep Learning

Linear model:

$$\hat{y}_i = w^T x_i$$

Neural network with 1 hidden layer:

$$\hat{y}_i = v^T h(\underbrace{W x_i}_{z_i})$$

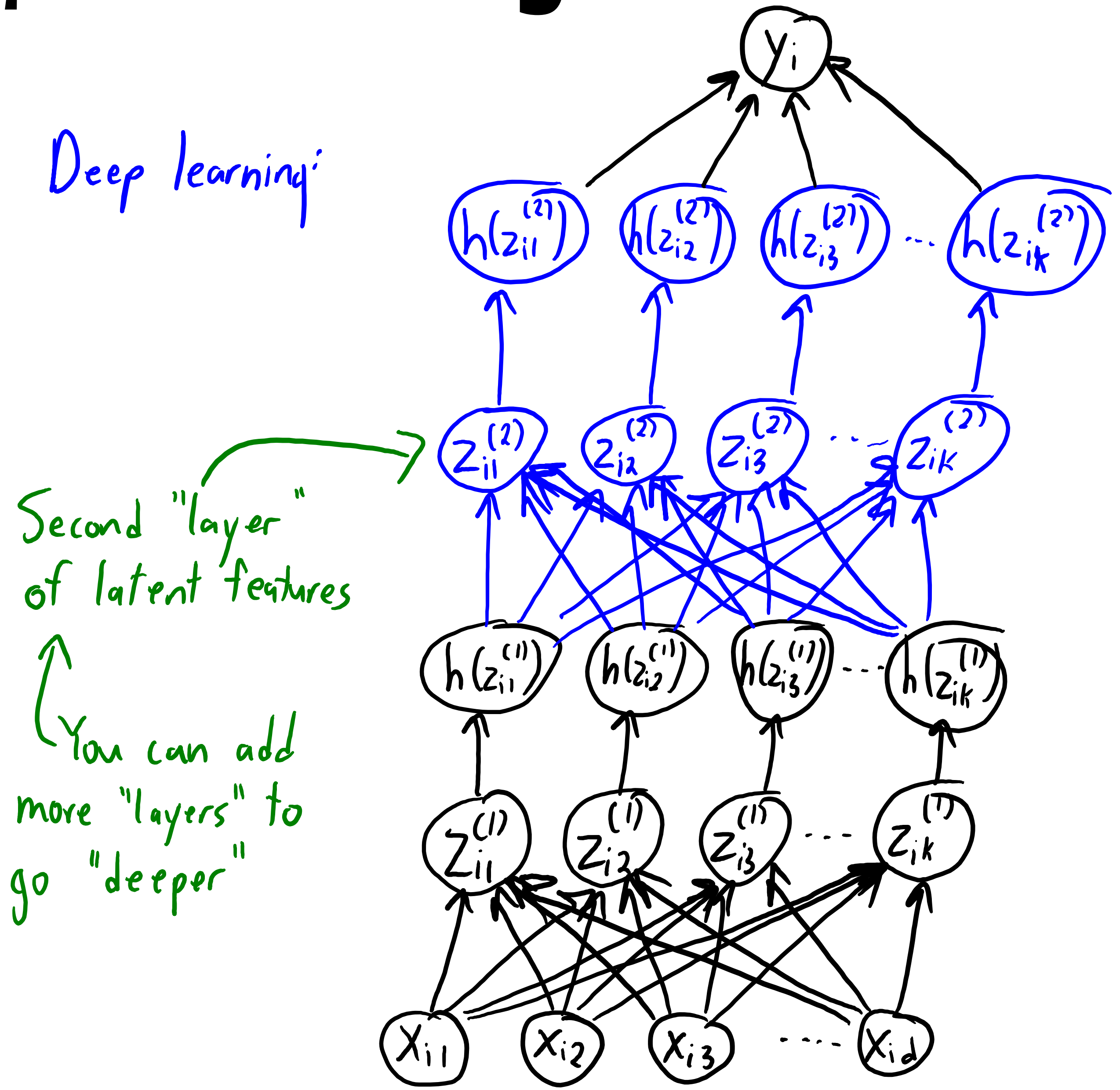
Neural network with 2 hidden layers:

$$\hat{y}_i = v^T h(\underbrace{W^{(2)} h(\underbrace{W^{(1)} x_i}_{z_i^{(1)}})}_{z_i^{(2)}})$$

Neural network with 3 hidden layers

$$\hat{y}_i = v^T h(\underbrace{W^{(3)} h(\underbrace{W^{(2)} h(\underbrace{W^{(1)} x_i}_{z_i^{(1)}})}_{z_i^{(2)}})}_{z_i^{(3)}})$$

Deep learning:



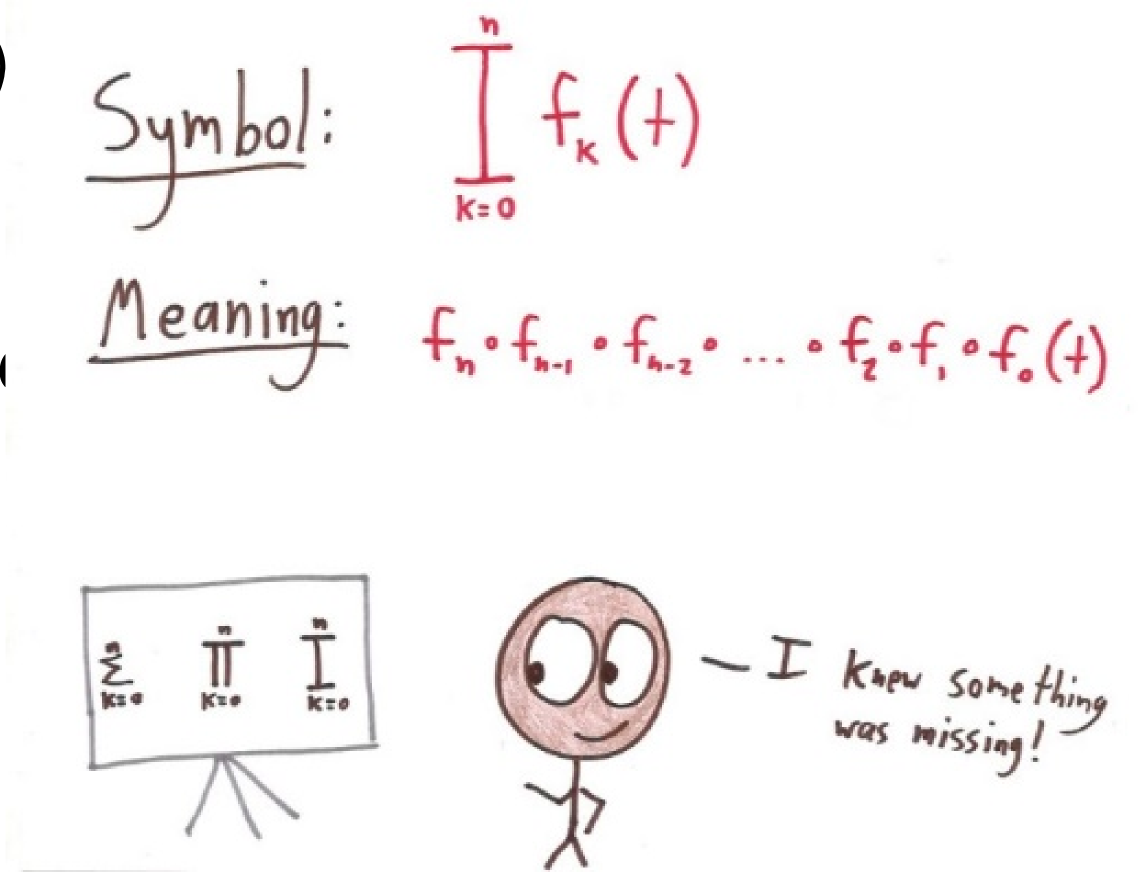
Deep Learning

- For 4 layers, we could write the prediction as:

$$\hat{y}_i = v^T h(W^{(4)} h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))))$$

- For 'm' layers, we could us

$$\hat{y}_i = v^T \left(\mathop{\text{I}}_{\ell=1}^{m} h(W^{(\ell)} x_i) \right)$$



Symbol: $\mathop{\text{I}}_{k=0}^{n} f_k(t)$
Meaning: $f_n \circ f_{n-1} \circ f_{n-2} \circ \ldots \circ f_2 \circ f_1 \circ f_0(t)$
$\sum_{k=0}^{n}$ $\prod_{k=0}^{n}$ $\mathop{\text{I}}_{k=0}^{n}$
— I knew something was missing!

https://mathwithbaddrawings.com/2016/04/27/symbols-that-math-urgently-needs-to-adopt

Coming Up Next

HISTORY OF DEEP LEARNING

‘Godfathers of AI’ honored with Turing Award, the Nobel Prize of computing

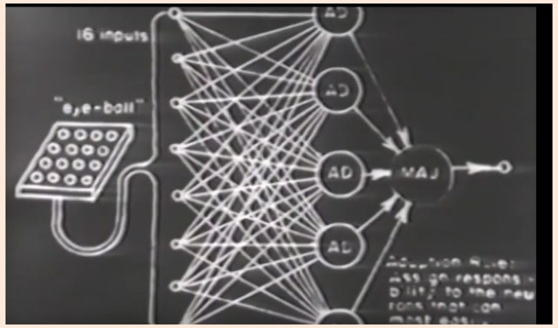
Yoshua Bengio, Geoffrey Hinton, and Yann LeCun laid the foundations for modern AI

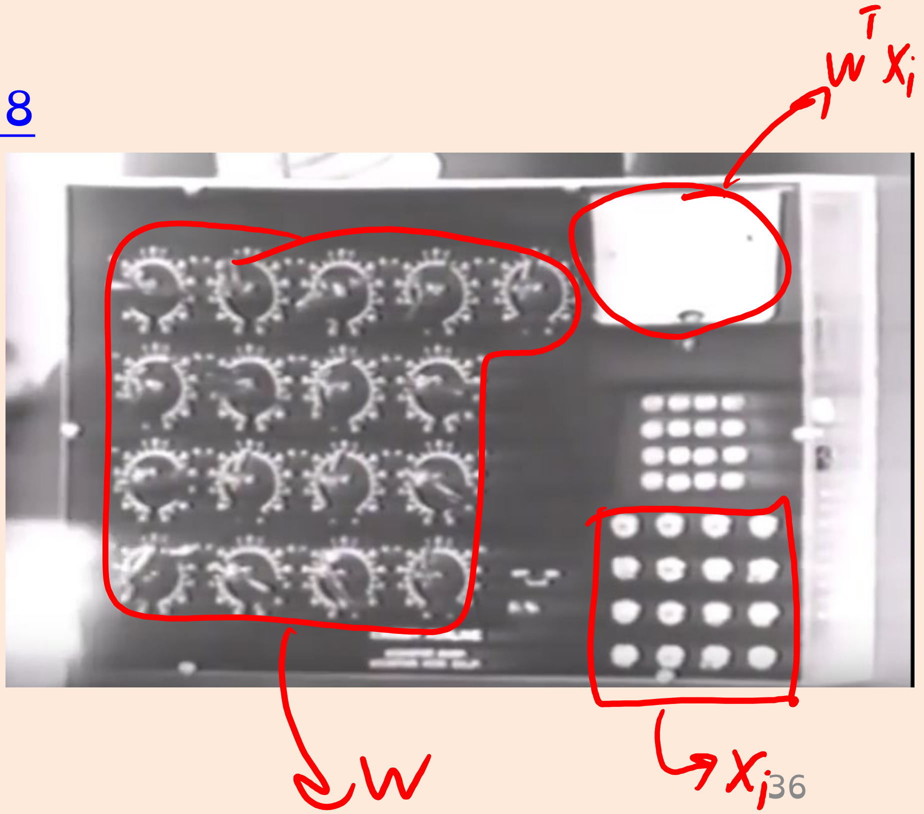
By James Vincent | Mar 27, 2019, 6:02am EDT

From left to right: Yann LeCun | Photo: Facebook; Geoffrey Hinton | Photo: Google; Yoshua Bengio | Photo: Botler AI

ML and Deep Learning History

- 1950 and 1960s: Initial excitement.
  - Perceptron: linear classifier and stochastic gradient (roughly).
  - "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." New York Times (1958).
    - https://www.youtube.com/watch?v=IEFRtz68m-8
  - Object recognition assigned to students as a summer project



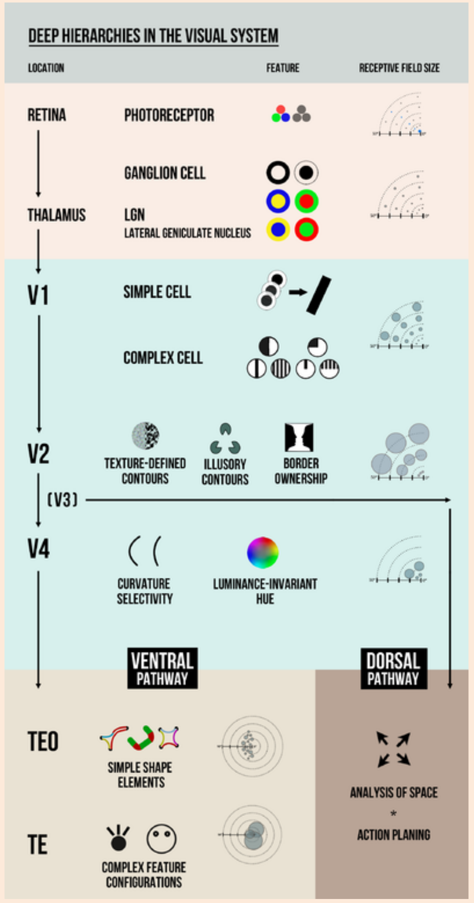


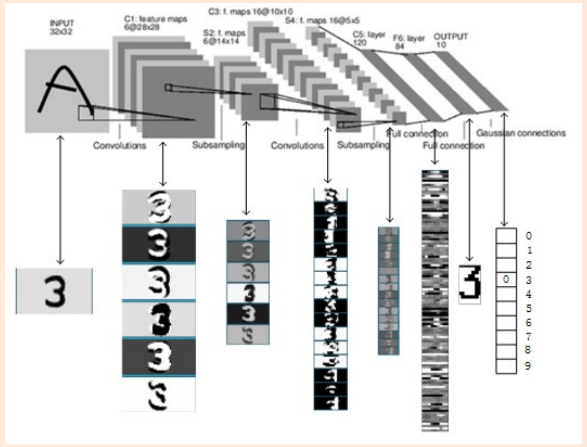
- Then drop in popularity:
  - Quickly realized limitations of linear models.

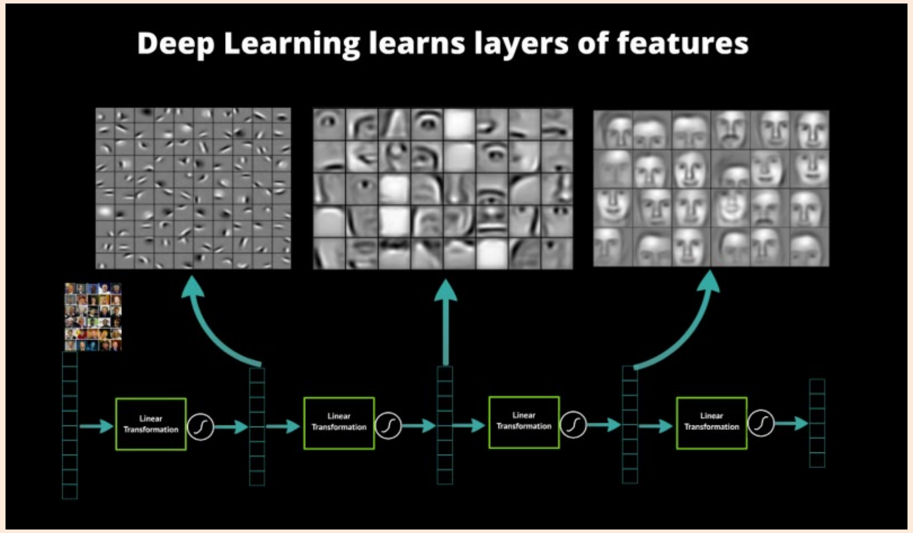
https://mitpress.mit.edu/books/perceptrons/

ML and Deep Learning History

- 1970 and 1980s: Connectionism (brain-inspired ML)
  - Want "connected networks of simple units".
    - Use parallel computation and distributed representations.
  - Adding hidden layers z_i increases expressive power.
    - With 1 layer and enough sigmoid units, a universal approximator.
  - Success in optical character recognition.







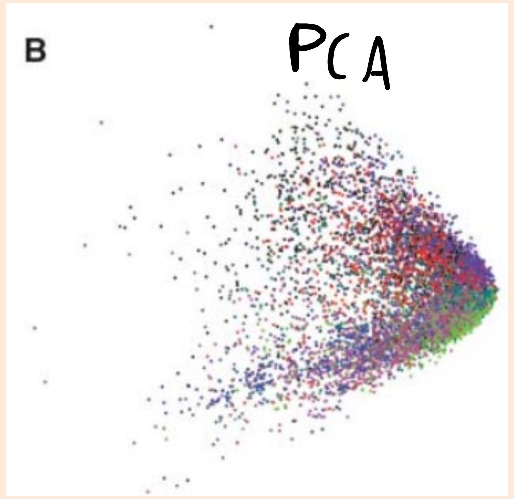
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing
http://www.datarobot.com/blog/a-primer-on-deep-learning/
http://blog.csdn.net/strint/article/details/44163869

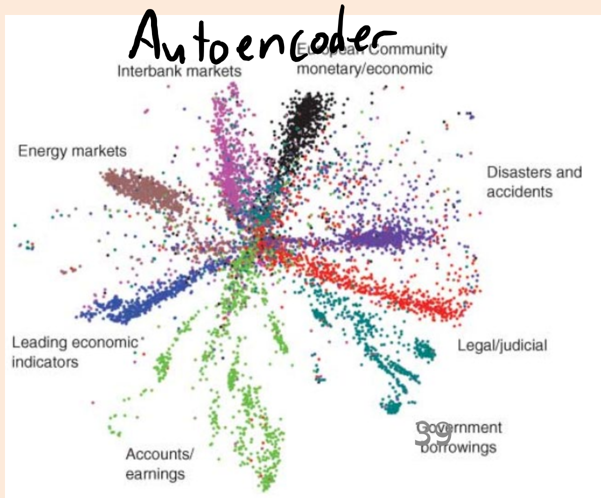
ML and Deep Learning History

- 1990s and early-2000s: drop in popularity.
  - It proved really difficult to get multi-layer models working robustly.
  - We obtained similar performance with simpler models:
    - Rise in popularity of logistic regression and SVMs with regularization and kernels.
  - Lots of internet successes (spam filtering, web search, recommendation).
  - ML moved closer to other fields like numerical optimization and statistics.

ML and Deep Learning History

- Late 2000s: push to revive connectionism as "deep learning".
  - Canadian Institute For Advanced Research (CIFAR) NCAP program:
    - "Neural Computation and Adaptive Perception".
    - Led by Geoff Hinton, Yann LeCun, and Yoshua Bengio ("Canadian mafia").
  - Unsupervised successes: "deep belief networks" and "autoencoders".
    - Could be used to initialize deep neural networks.
    - https://www.youtube.com/watch?v=KuPai0ogiHk

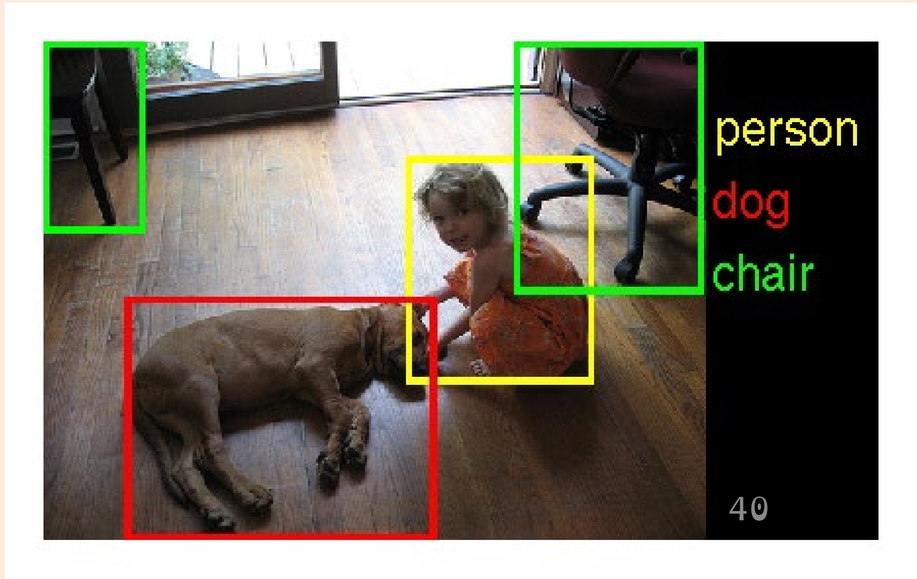




https://www.cs.toronto.edu/~hinton/science.pdf

2010s: DEEP LEARNING!!!

- Bigger datasets, bigger models, parallel computing (GPUs/clusters).
  - And some tweaks to the models from the 1980s.
- Huge improvements in automatic speech recognition (2009).
  - All phones now have deep learning.
- Huge improvements in computer vision (2012).
  - Changed computer vision field almost instantly.
  - This is now finding its way into products.



http://www.image-net.org/challenges/LSVRC/2014/

2010s: DEEP LEARNING!!!

- Media hype:
  - "How many computers to identify a cat? 16,000"
    New York Times (2012).
  - "Why Facebook is teaching its machines to think like humans"
    Wired (2013).
  - "What is 'deep learning' and why should businesses care?"
    Forbes (2013).
  - "Computer eyesight gets a lot more accurate"
    New York Times (2014).

- 2015: huge improvement in language understanding.

Cut-off for Final Exam

(Final exam will have materials from everything before this slide)

Summary

- Neural networks learn features z_i for supervised learning.
- Sigmoid function avoids degeneracy by introducing non-linearity.
  - Universal approximator with large-enough ‘k’.
- Biological motivation for (deep) neural networks.
- Deep learning considers neural networks with many hidden layers.
  - Can more-efficiently represent some functions.
- Unprecedented performance on difficult pattern recognition tasks.

- Next time:
  - Training deep networks.

Please Do Course Evaluation!

Review Questions

- Q1: What is the problem with using a linear encoder and a linear predictor for a neural network?
- Q2: What is the motivation for using multiple layers of encoders?
- Q3: Does it make sense to use neural networks for classifying linearly separable data?

Why $z_i = Wx_i$?

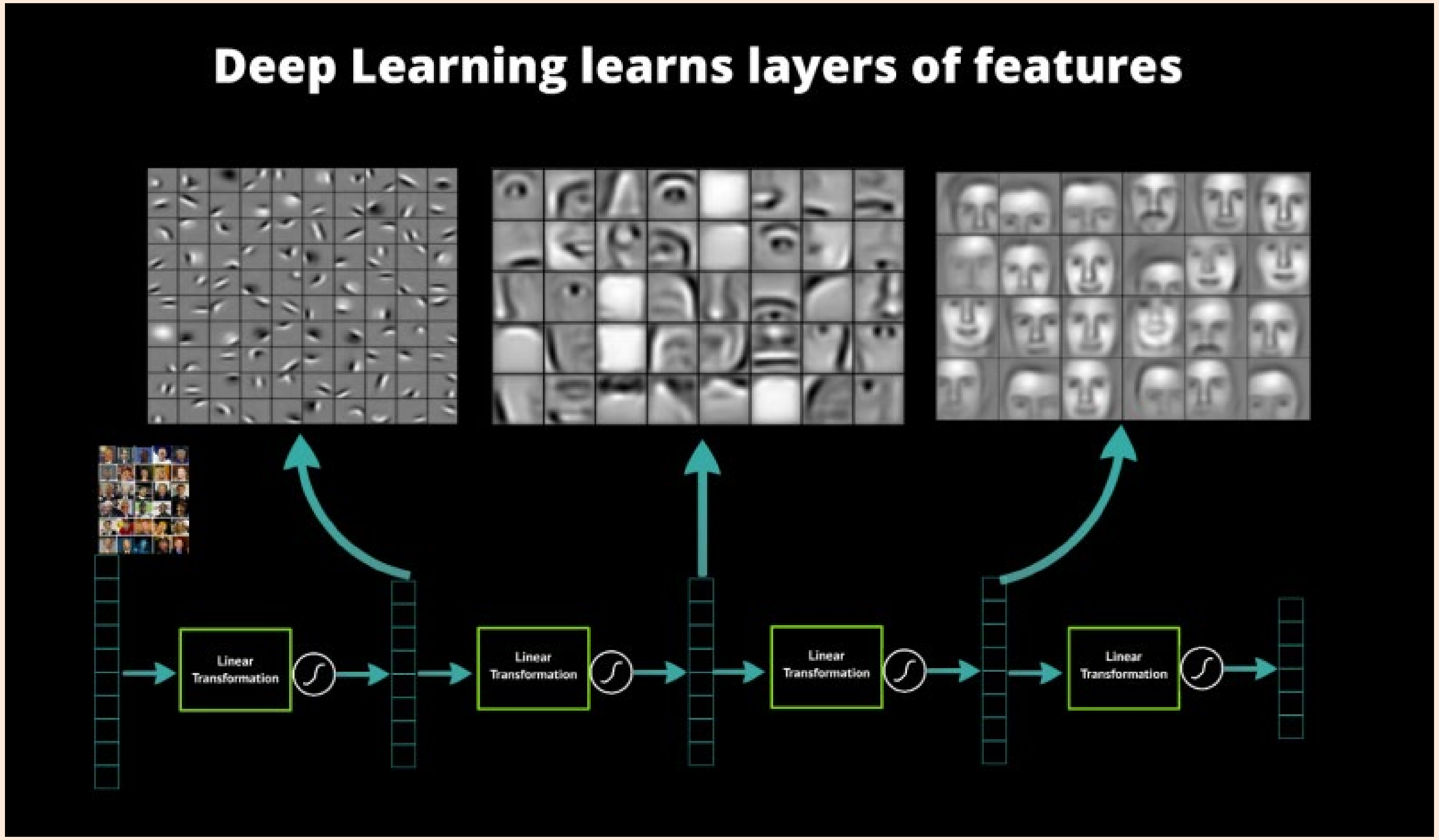
- In PCA we had that the optimal $Z = XW^T(WW^T)^{-1}$.
- If W had normalized+orthogonal rows, $Z = XW^T$ (since $WW^T = I$).
  - So $z_i = Wx_i$ in this normalized+orthogonal case.

- Why we would use $z_i = Wx_i$ in neural networks?
  - We didn't enforce normalization or orthogonality.

- Well, the value $W^T(WW^T)^{-1}$ is just "some matrix".
  - You can think of neural networks as just directly learning this matrix.

Cool Picture Motivation for Deep Learning

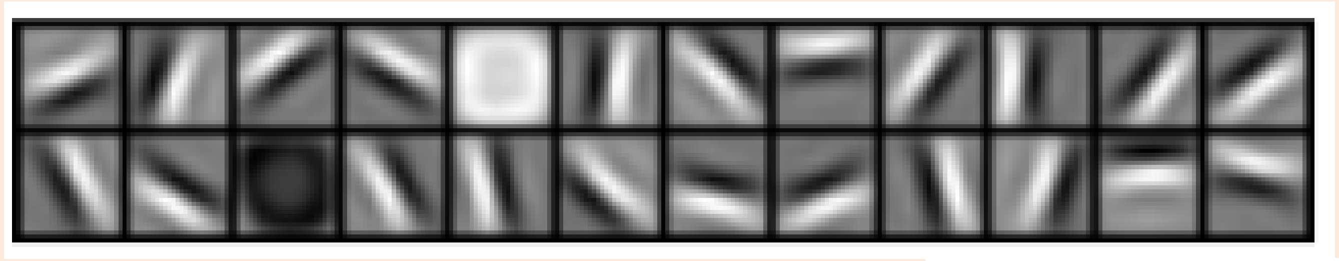
- Faces might be composed of different “parts”:



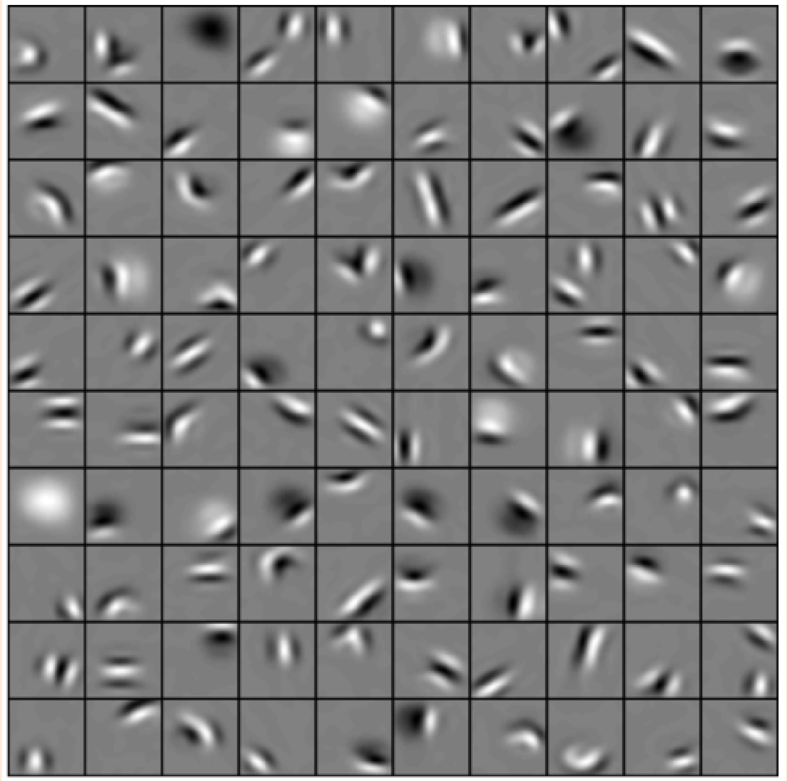
http://www.datarobot.com/blog/a-primer-on-deep-learning/

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



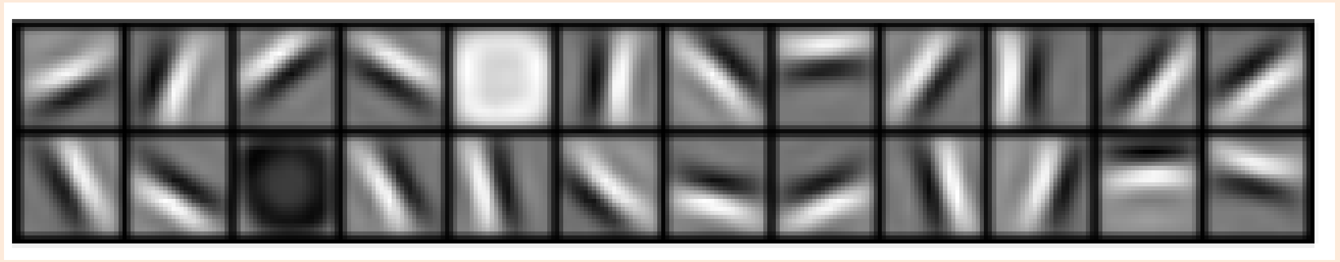
} "Gabor filters"



- Attempt to visualize second layer:
  - Corners, angles, surface boundaries?
- Models require many tricks to work.
  - We'll discuss these next time.

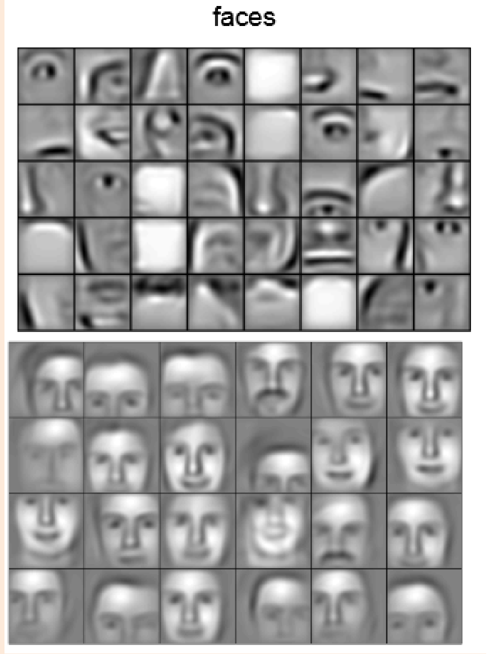
Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



$\left.\right\}$ "Gabor filters"

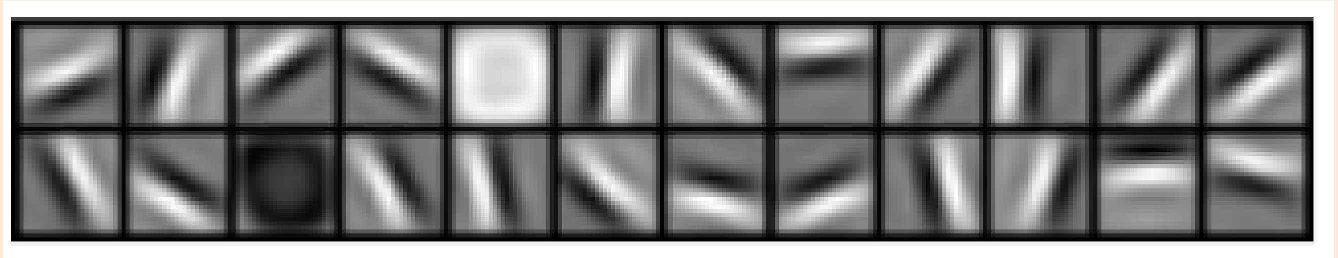
- Visualization of second and third layers trained on specific objects:



http://www.cs.toronto.edu/~rgrosse

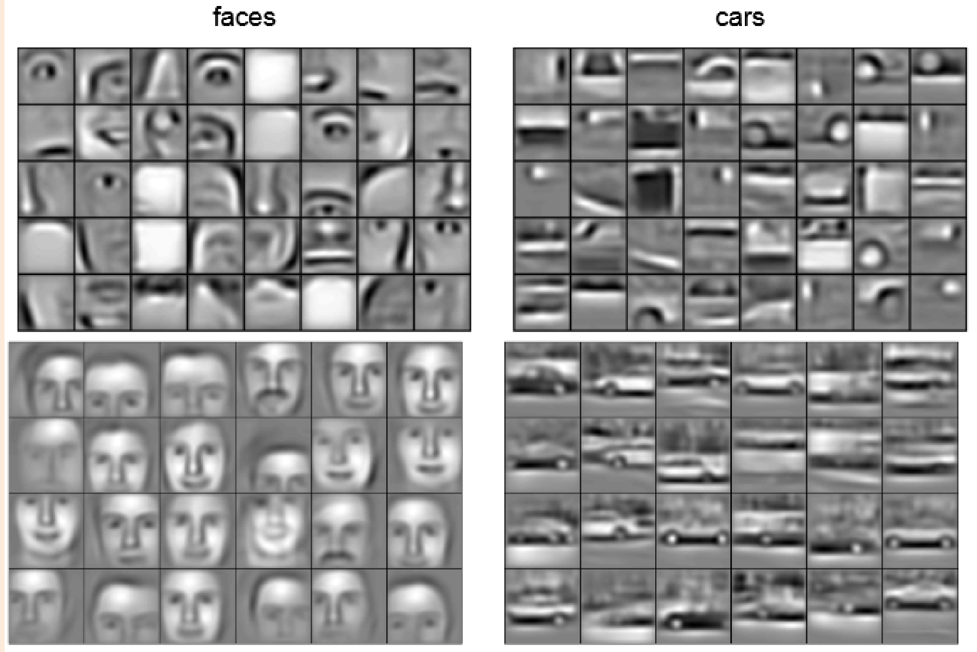
Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



$\Big\}$ "Gabor filters"

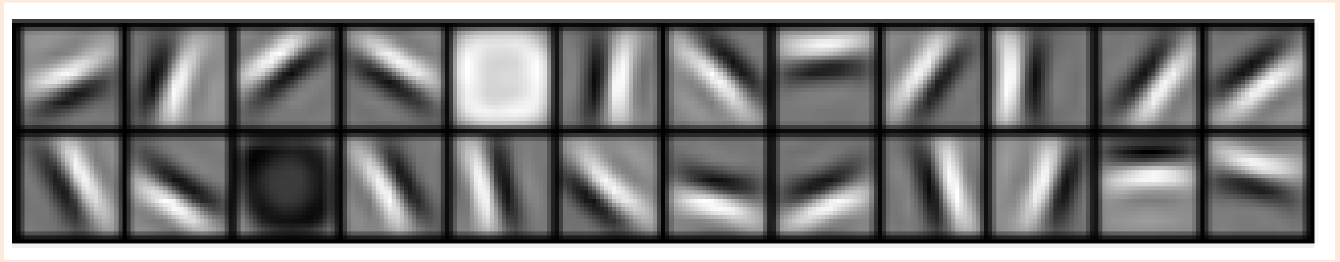
- Visualization of second and third layers trained on specific objects:



http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

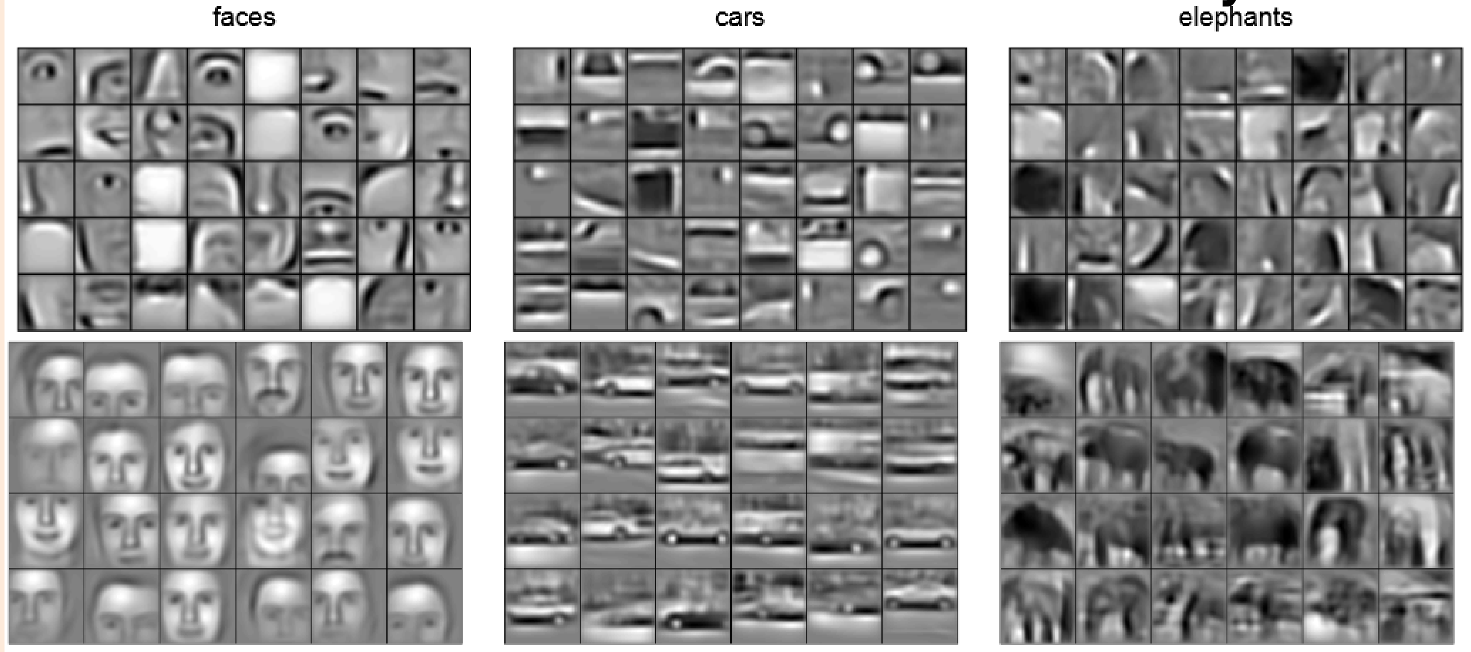
Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



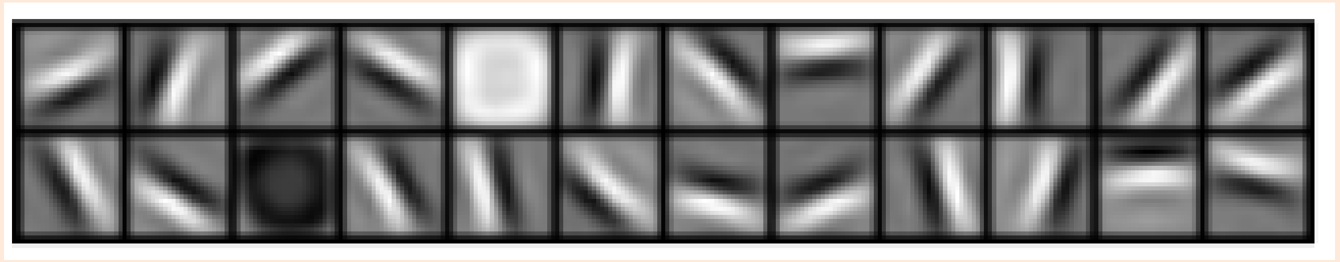
$\Big\}$ "Gabor filters"

- Visualization of second and third layers trained on specific objects:



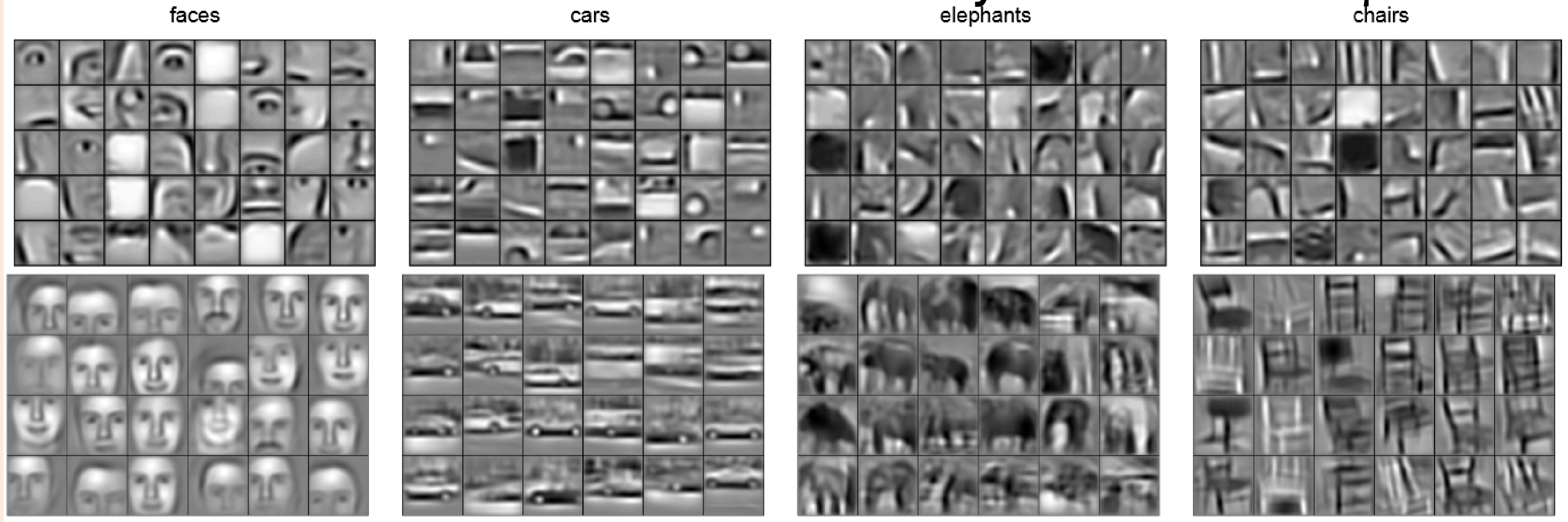
Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



$\}$ "Gabor filters"

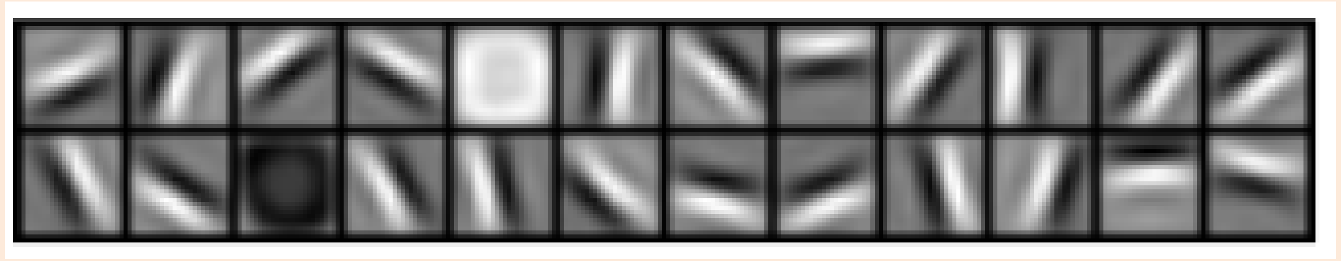
- Visualization of second and third layers trained on specific objects:



http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

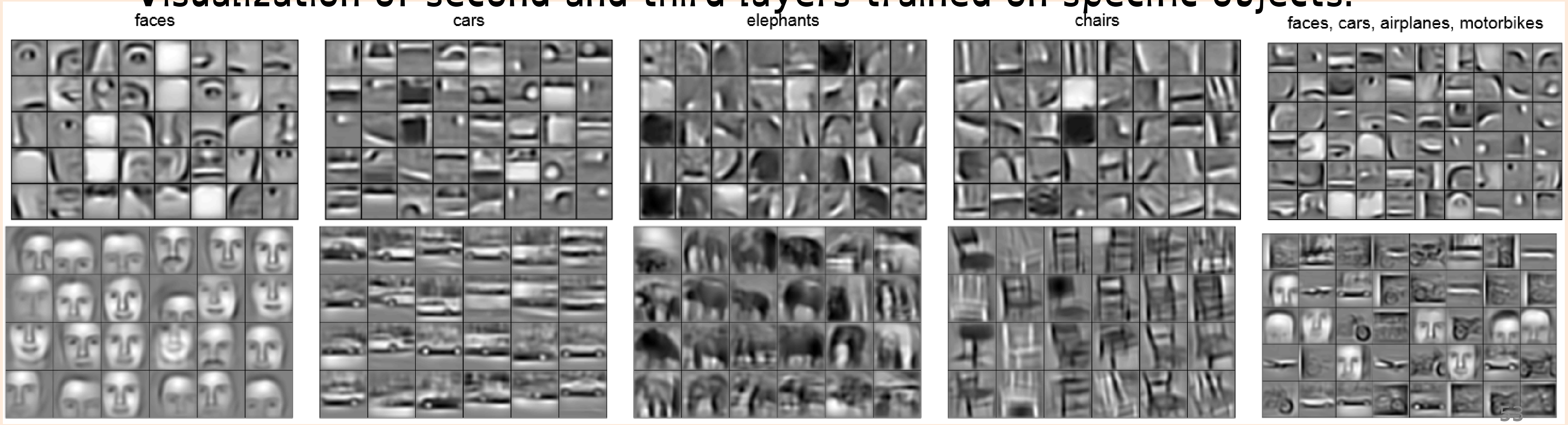
Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



$\left.\right\}$ "Gabor filters"

- Visualization of second and third layers trained on specific objects:


faces
cars
elephants
chairs
faces, cars, airplanes, motorbikes