CS 6890: Deep Learning

Introduction

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

How do we build predictive models?

- **Expert approach**: Manual engineering of complex models, **knowledge-driven**:
  - CV: SIFT features took more than 10 years to develop.
  - Diabetes: Physiological models using complex sets of state transition equations.

- Cognitively demanding, brittle …

- Solution: build models that find and exploit patterns in data.

How do we get value from data?

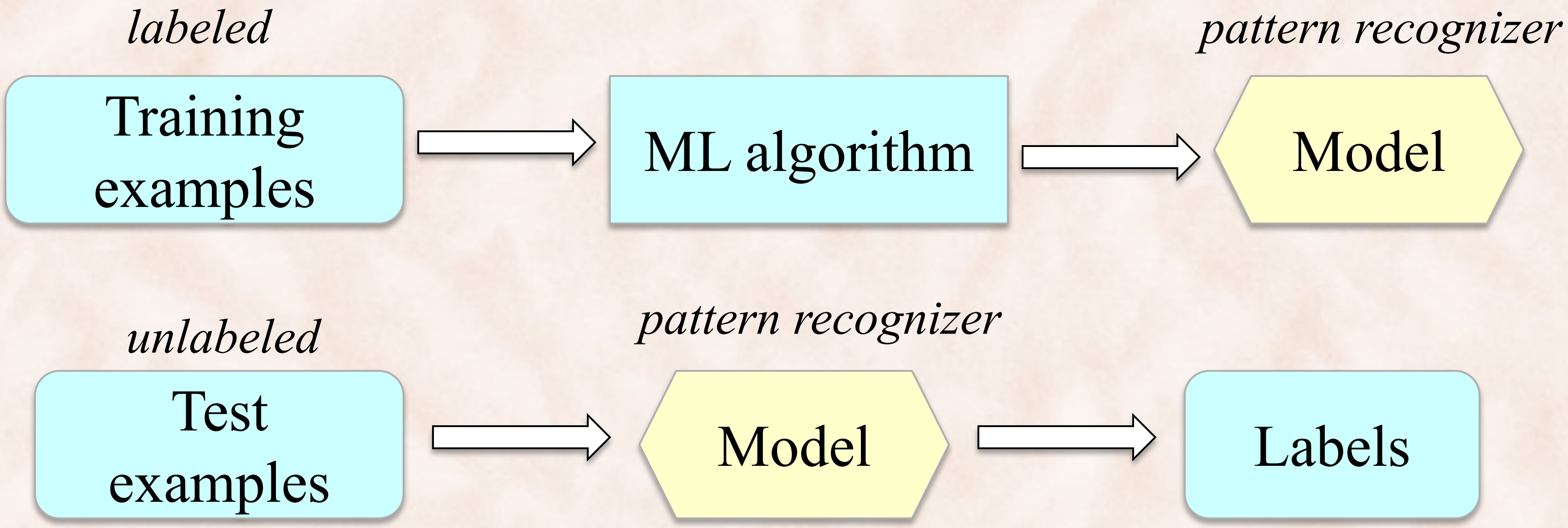
- Advances in Machine Learning (ML) enable a **data-driven** approach to building predictive models.

- **Machine learning approach**: Simple, generic architectures and algorithms; ***complexity comes from the data***.
  - CV: Edge detectors trained in minutes using auto-encoders.
  - Diabetes: Physiological models trained using RNNs.

- Large amounts of (labeled) data an important factor in the success of Deep Learning.

What is Machine Learning?

- **Machine Learning** = constructing computer programs that *learn* from *experience* to perform well on a given task.
  - Supervised Learning i.e. discover patterns from labeled examples that enable predictions on (previously unseen) unlabeled examples.

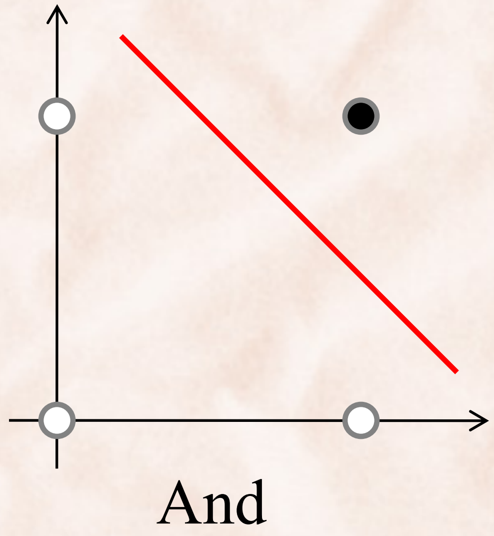


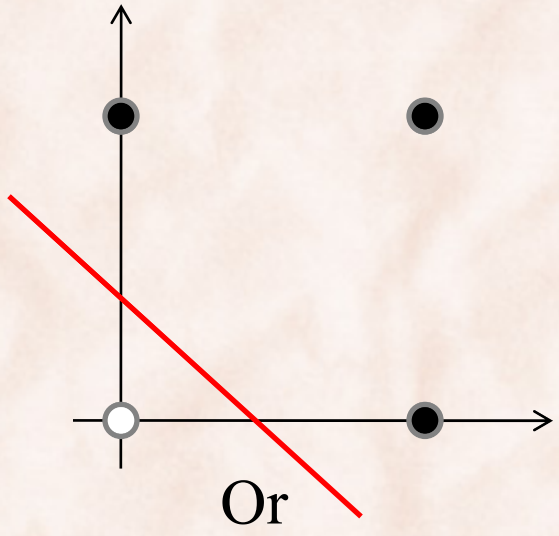
ML is Meta-Programming

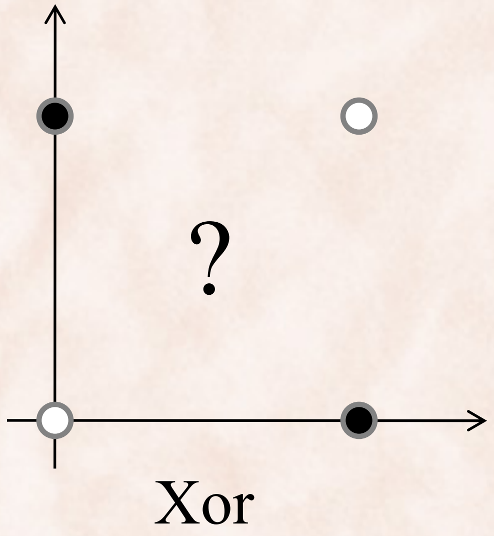
- An ML **model** (e.g. a *neural network*) is a computer **program**:
  - We do not want to explicitly program (model) the computer for each particular task.
  - Use a general ML algorithm and task-specific data to automatically create the Program, i.e. the Model, that solves the task.

$\Rightarrow$ An ML algorithm (e.g. *gradient descent*) is a **meta-program**.

Linear vs. Non-linear Decision Boundaries







$$\left.\begin{aligned}\varphi(\mathbf{x}) &= [1, x_1, x_2]^T \\ \mathbf{w} &= [w_0, w_1, w_2]^T\end{aligned}\right\} \Rightarrow \mathbf{w}^T\varphi(\mathbf{x}) = [w_1, w_2]^T[x_1, x_2] + w_0$$

Linear Discriminant Functions: Two classes ($K = 2$)

- Use a linear function of the input vector:

$$h(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + w_0$$

$\boxed{\textit{weight vector}}$ $\boxed{\textit{bias} = -\textit{threshold}}$

- Decision:

  $\mathbf{x} \in C_1$ if $h(\mathbf{x}) \geq 0$, otherwise $\mathbf{x} \in C_2$.

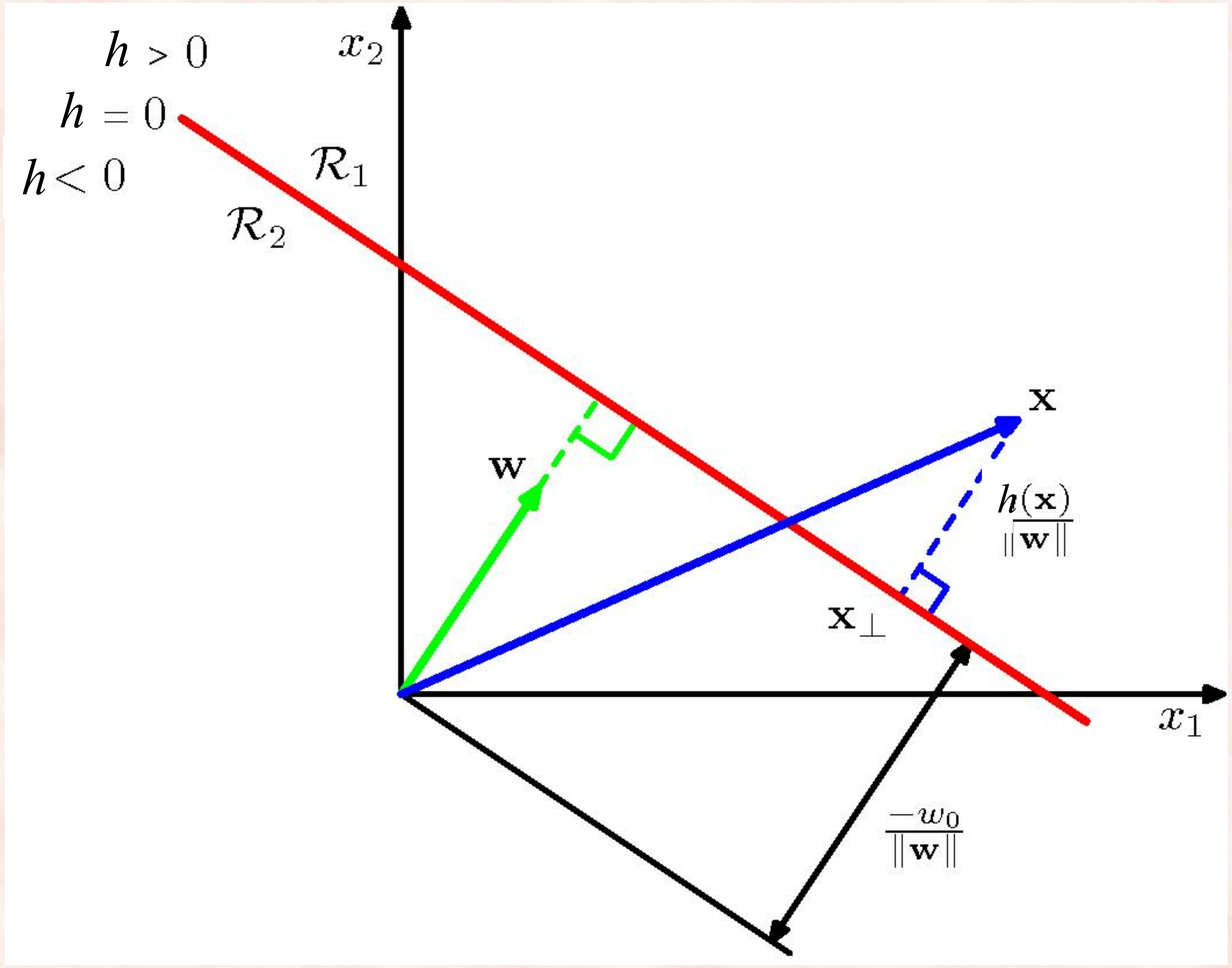
  $\Rightarrow$ decision boundary is hyperplane $h(\mathbf{x}) = 0$.

- Properties:
  - $\mathbf{w}$ is orthogonal to vectors lying within the decision surface.
  - w_0 controls the location of the decision hyperplane.

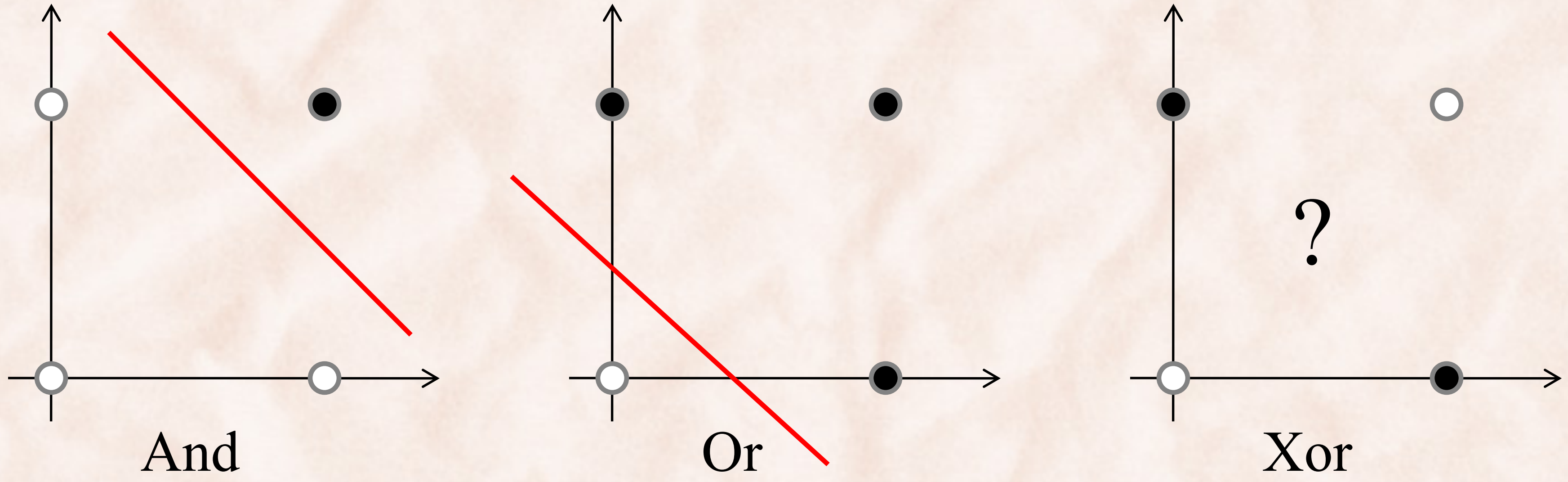
$$h(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + w_0$$

weight vector

bias = – threshold



Linear vs. Non-linear Decision Boundaries



$$\left.\begin{array}{r}\varphi(\mathbf{x}) = [1, x_1, x_2]^T \\ \mathbf{w} = [w_0, w_1, w_2]^T\end{array}\right\} \Rightarrow \mathbf{w}^T\varphi(\mathbf{x}) = [w_1, w_2]^T[x_1, x_2] + w_0$$

How to Find Non-linear Decision Boundaries

1) Logistic Regression with manually engineered features:
   - Quadratic features.

2) Kernel methods (e.g. SVMs) with non-linear kernels:
   - Quadratic kernels, Gaussian kernels.

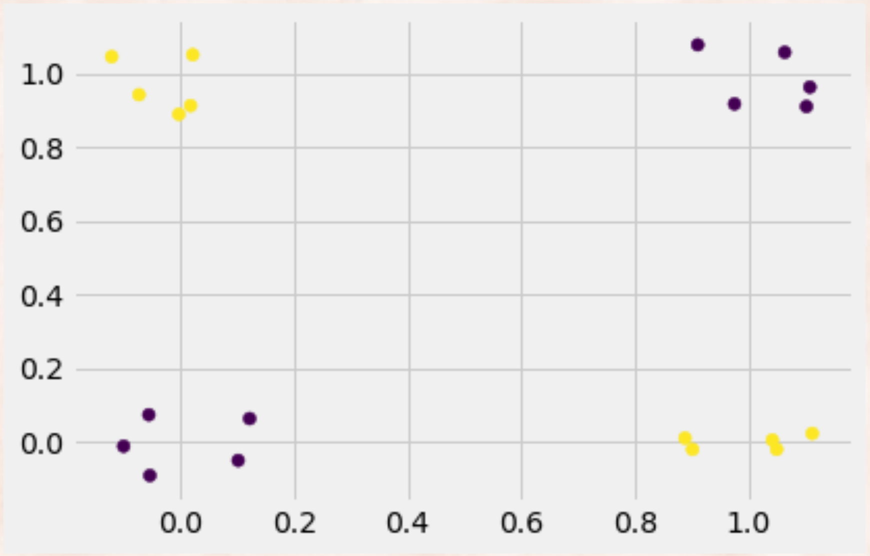
Deep Learning class

3) Unsupervised feature learning (e.g. auto-encoders):
   - Plug learned features in any linear classifier.

4) Neural Networks with one or more hidden layers:
   - Automatically learned features.

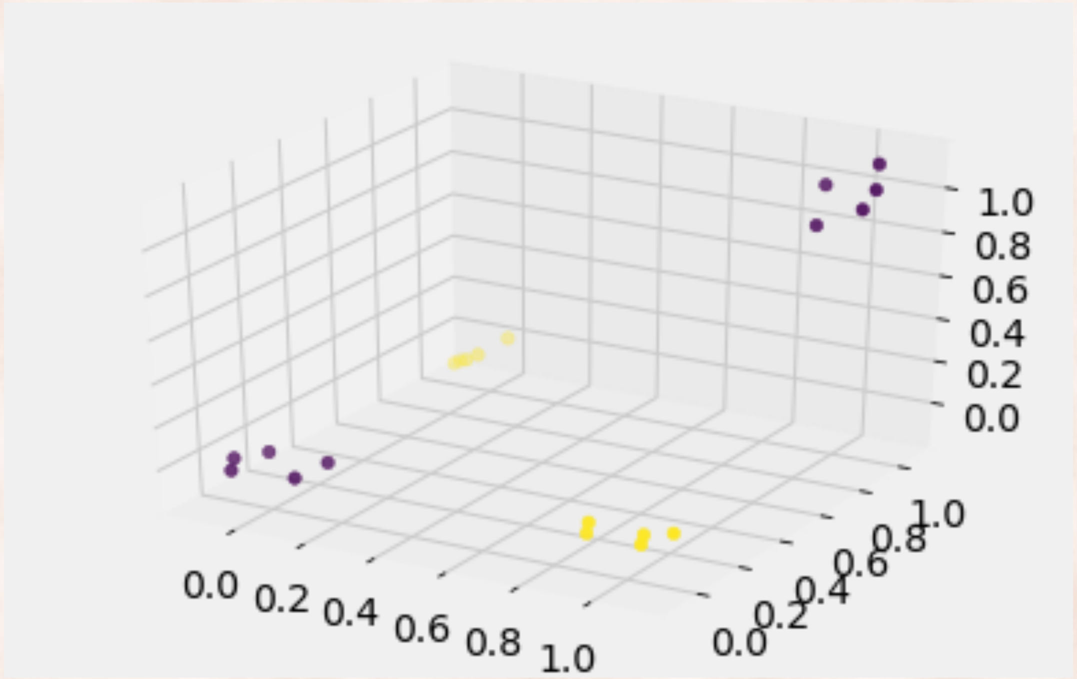
Non-Linear Classification: XOR Dataset

$$\mathbf{x} = [x_1, x_2]$$



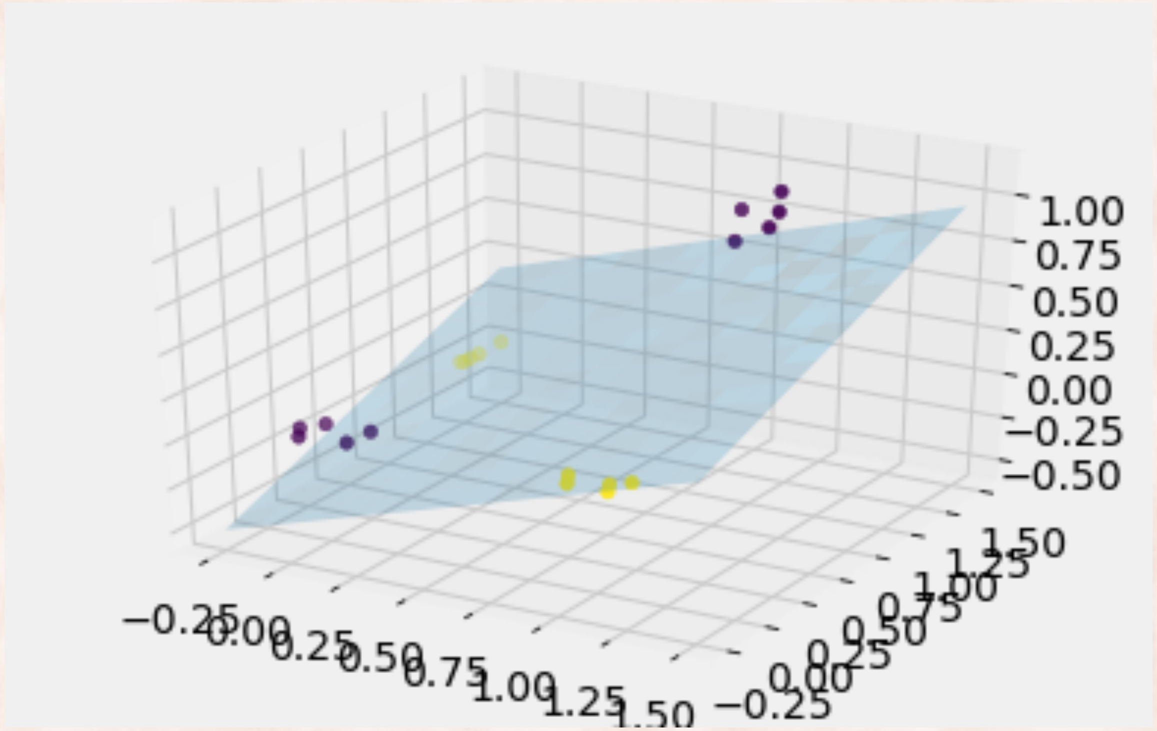
1) Manually Engineered Features: Add x_1x_2

$$\mathbf{x} = [x_1, x_2, x_1x_2]$$



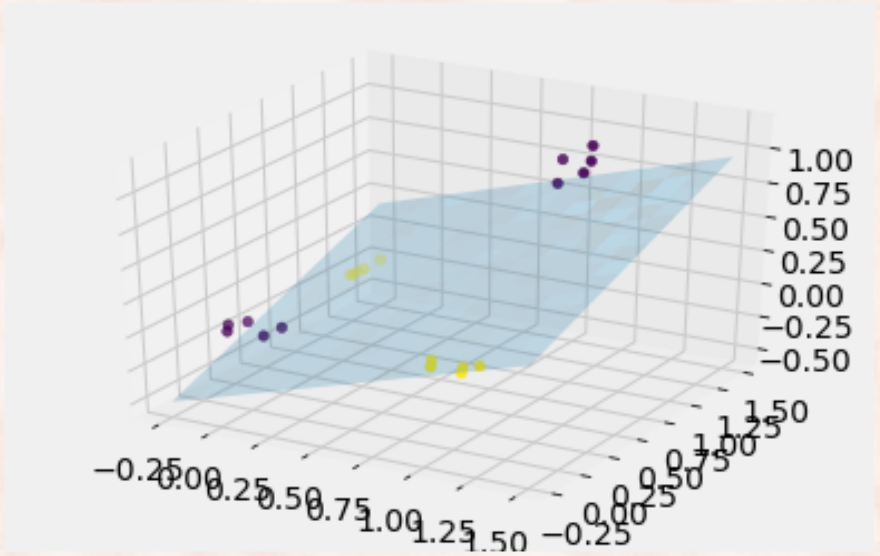
Logistic Regression with Manually Engineered Features

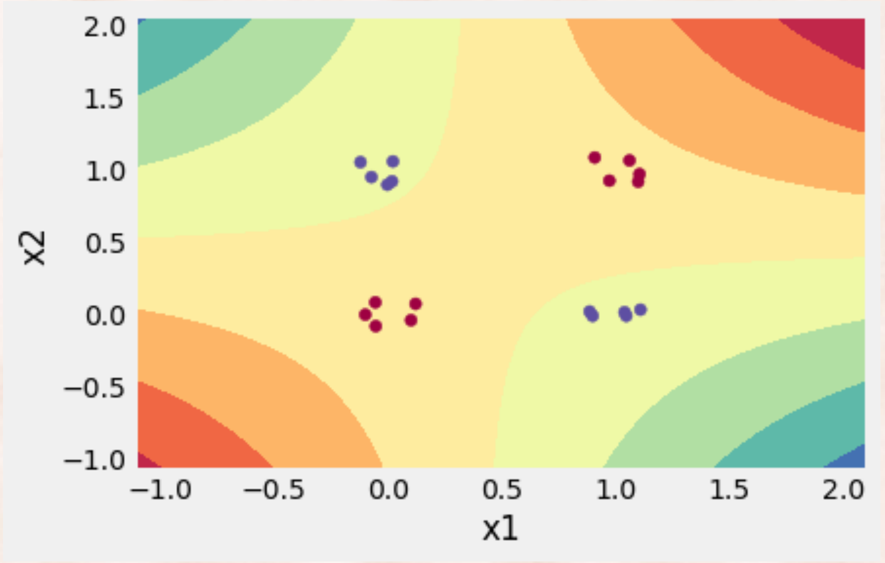
$$\mathbf{x} = [x_1, x_2, x_1 x_2]$$



Perceptron with Manually Engineered Features

Project $\mathbf{x} = [x_1, x_2, x_1x_2]$ and decision hyperplane back to $\mathbf{x} = [x_1, x_2]$

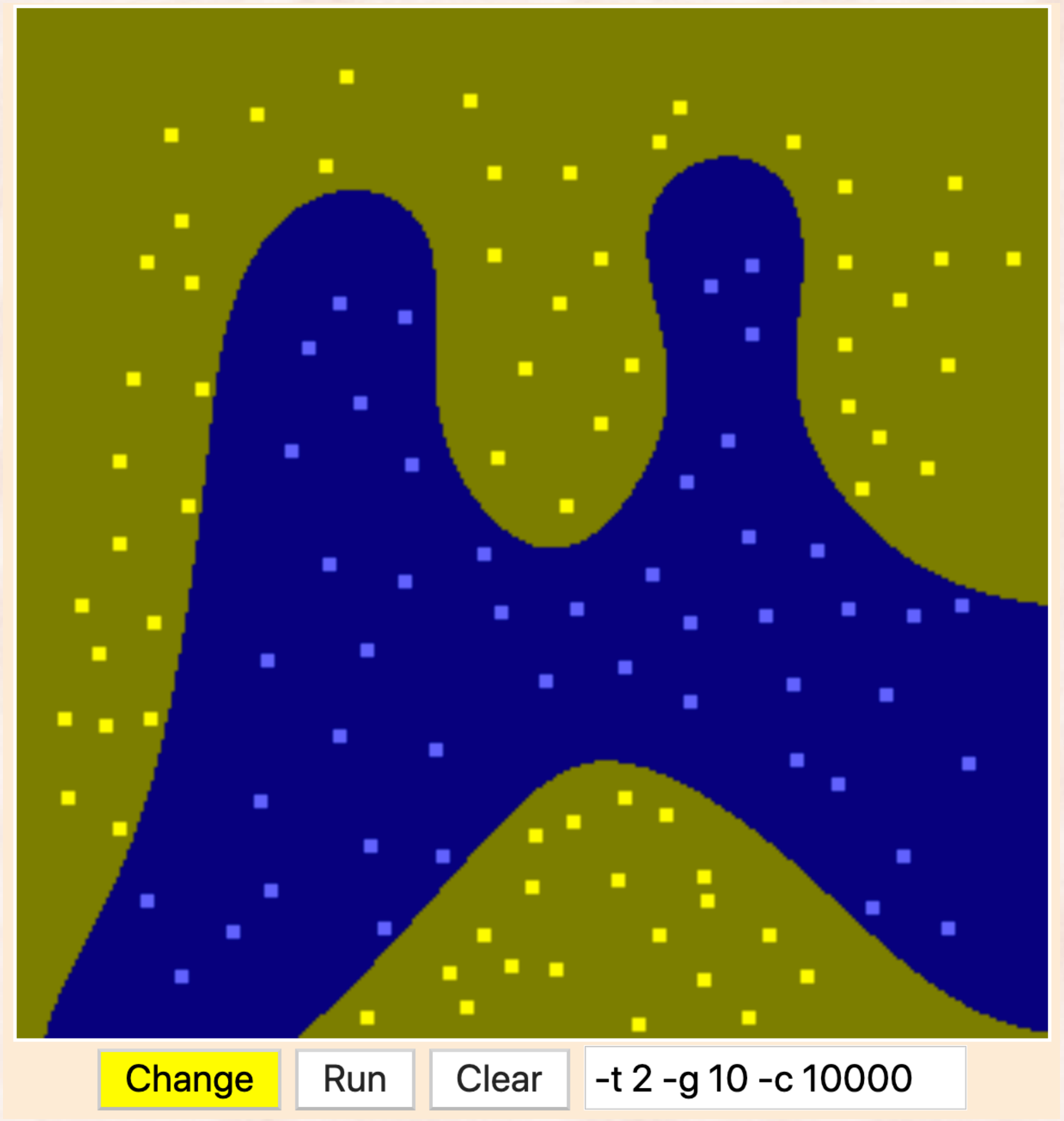




2) Kernel Methods with Non-Linear Kernels

- Perceptrons, SVMs can be '*kernelized*':
  1. Re-write the algorithm such that during training and testing feature vectors $\mathbf{x}$, $\mathbf{y}$ appear only in dot-products $\mathbf{x}^T\mathbf{y}$.
  2. Replace dot-products $\mathbf{x}^T\mathbf{y}$ with *non-linear kernels* $K(\mathbf{x}, \mathbf{y})$:
     - K is a kernel if and only if $\exists\varphi$ such that $K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y})$
       - φ can be in a much higher dimensional space.
         - » e.g. combinations of up to *k* original features
       - $\varphi(\mathbf{x})^T \varphi(\mathbf{y})$ can be computed efficiently without enumerating $\varphi(\mathbf{x})$ or $\varphi(\mathbf{y})$.

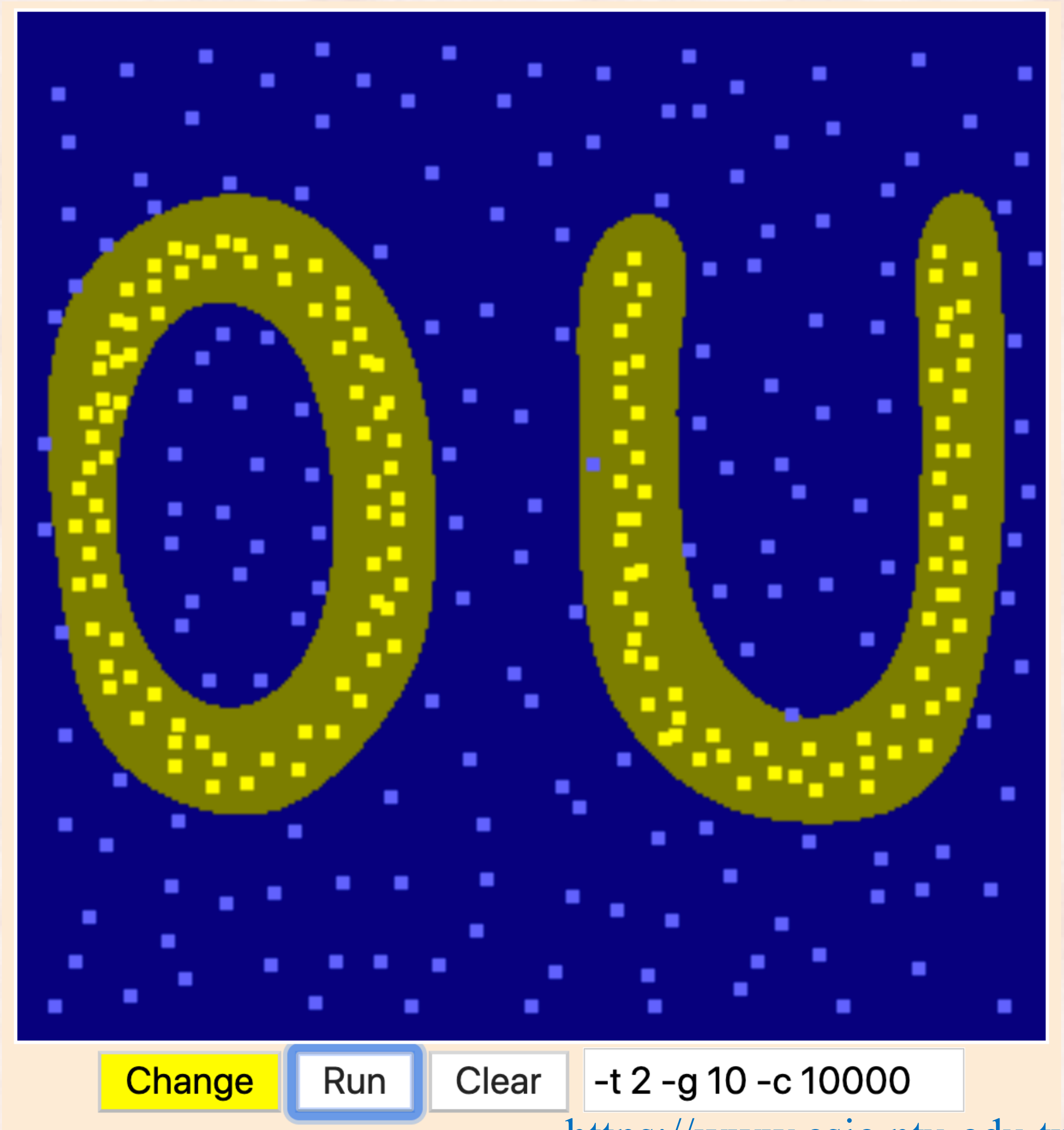
Most Data is Not Linearly Separable



-t 2 -g 10 -c 10000

https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Most Data is Not Linearly Separable

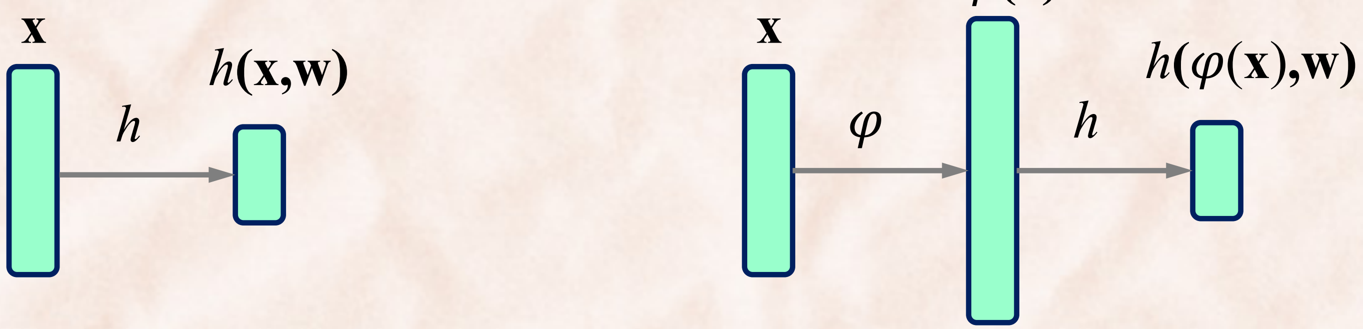


```
-t 2 -g 10 -c 10000
```

https://www.csie.ntu.edu.tw/~cjlin/libsvm/

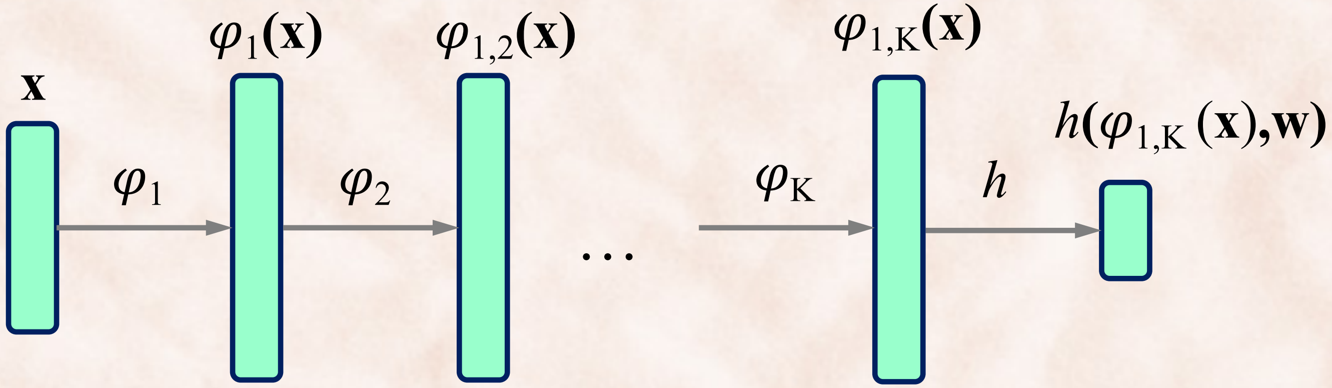
Machine Learning vs. Deep Learning

Machine Learning



$\mathbf{x}$
h
$h(\mathbf{x},\mathbf{w})$
$\mathbf{x}$
φ
$\varphi(\mathbf{x})$
h
$h(\varphi(\mathbf{x}),\mathbf{w})$

Deep Learning



$\mathbf{x}$
φ_1
$\varphi_1(\mathbf{x})$
φ_2
$\varphi_{1,2}(\mathbf{x})$
…
φ_K
$\varphi_{1,K}(\mathbf{x})$
h
$h(\varphi_{1,K}(\mathbf{x}),\mathbf{w})$

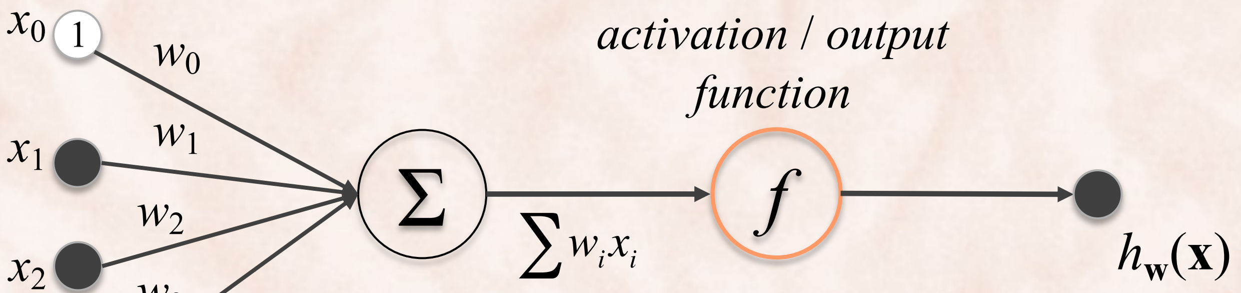
From Linear Discriminant Functions to Logistic Regression

- Use a linear function of the input vector:

$$h(\mathbf{x}) = \underbrace{\mathbf{w}^T}_{\text{weight vector}} \varphi(\mathbf{x}) + \underbrace{w_0}_{bias\,=\,-\,threshold}$$

weight vector

bias = – threshold



Logistic sigmoid $f(z) = \dfrac{1}{1+\exp(-z)}$

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$$

Activation Functions

unit step $f(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$

Perceptron

logistic $f(z) = \dfrac{1}{1+e^{-z}}$

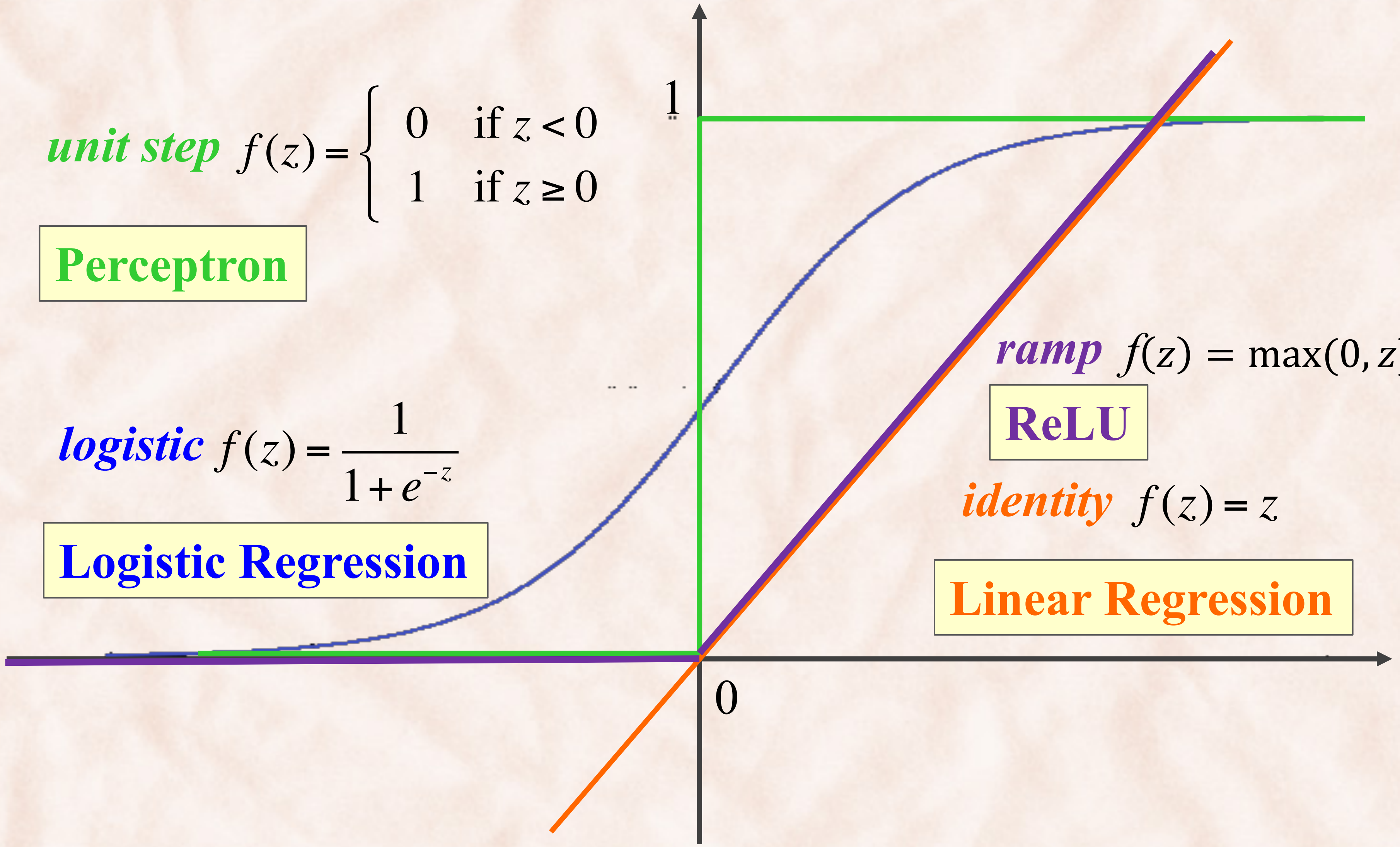
Logistic Regression

ramp $f(z) = \max(0, z)$

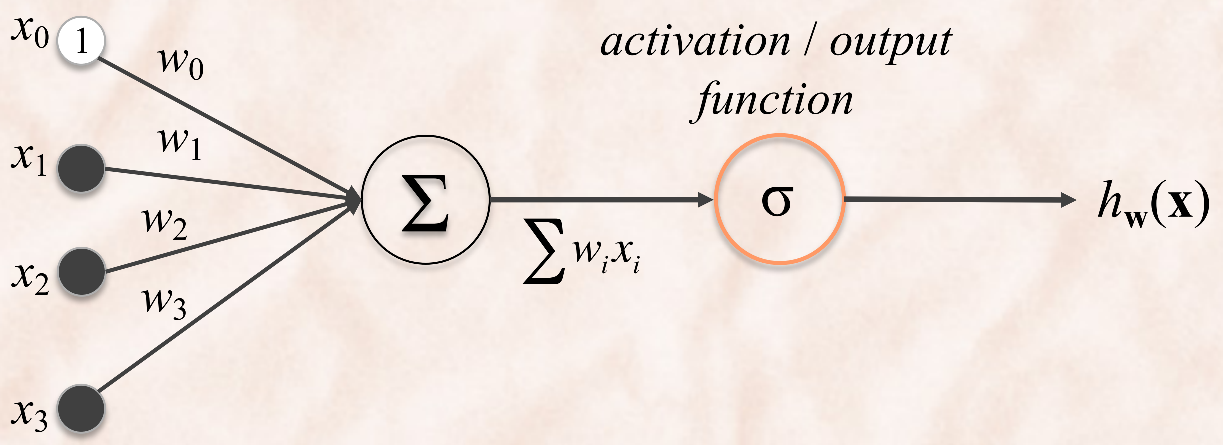
ReLU

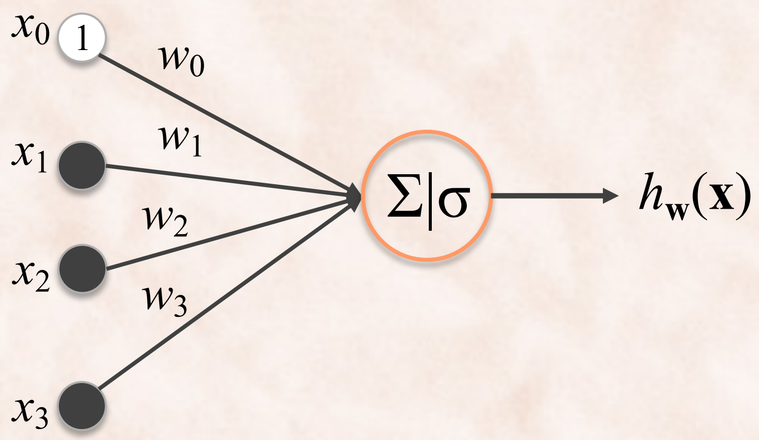
identity $f(z) = z$

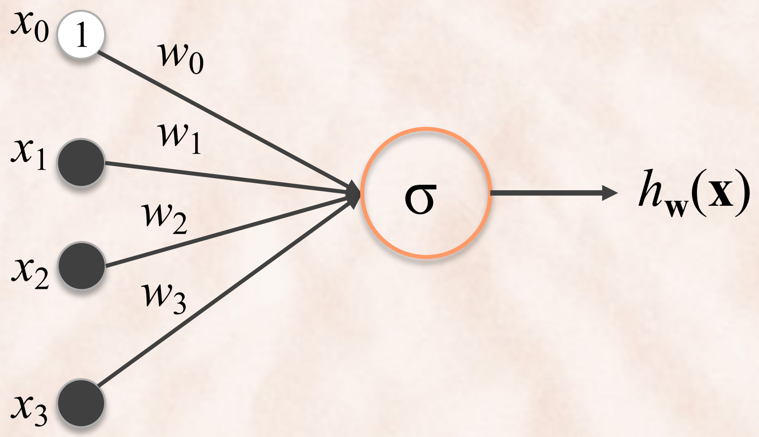
Linear Regression



Logistic Regression is a Logistic Neuron

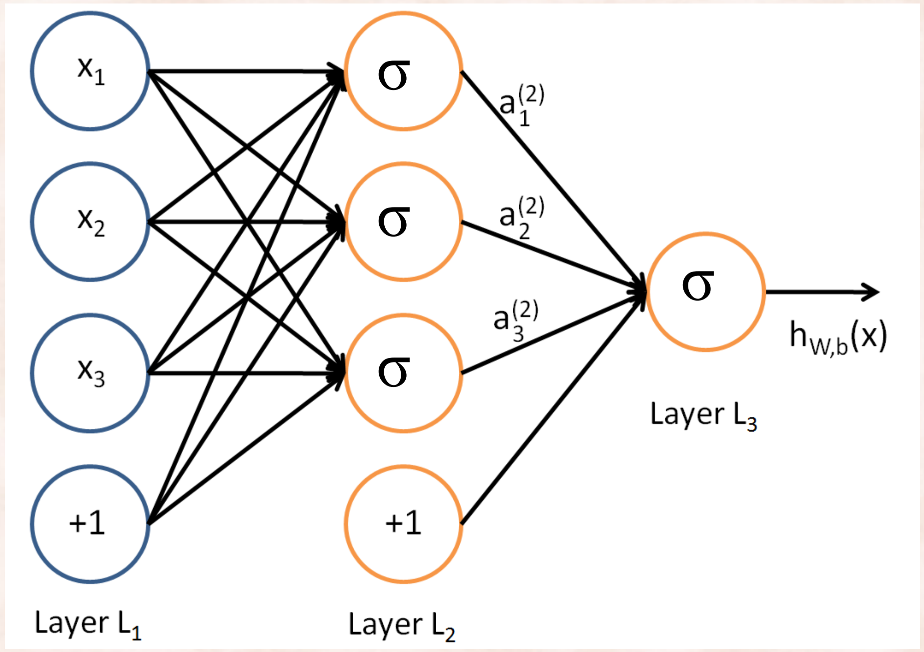






Feed-Forward Neural Network Model

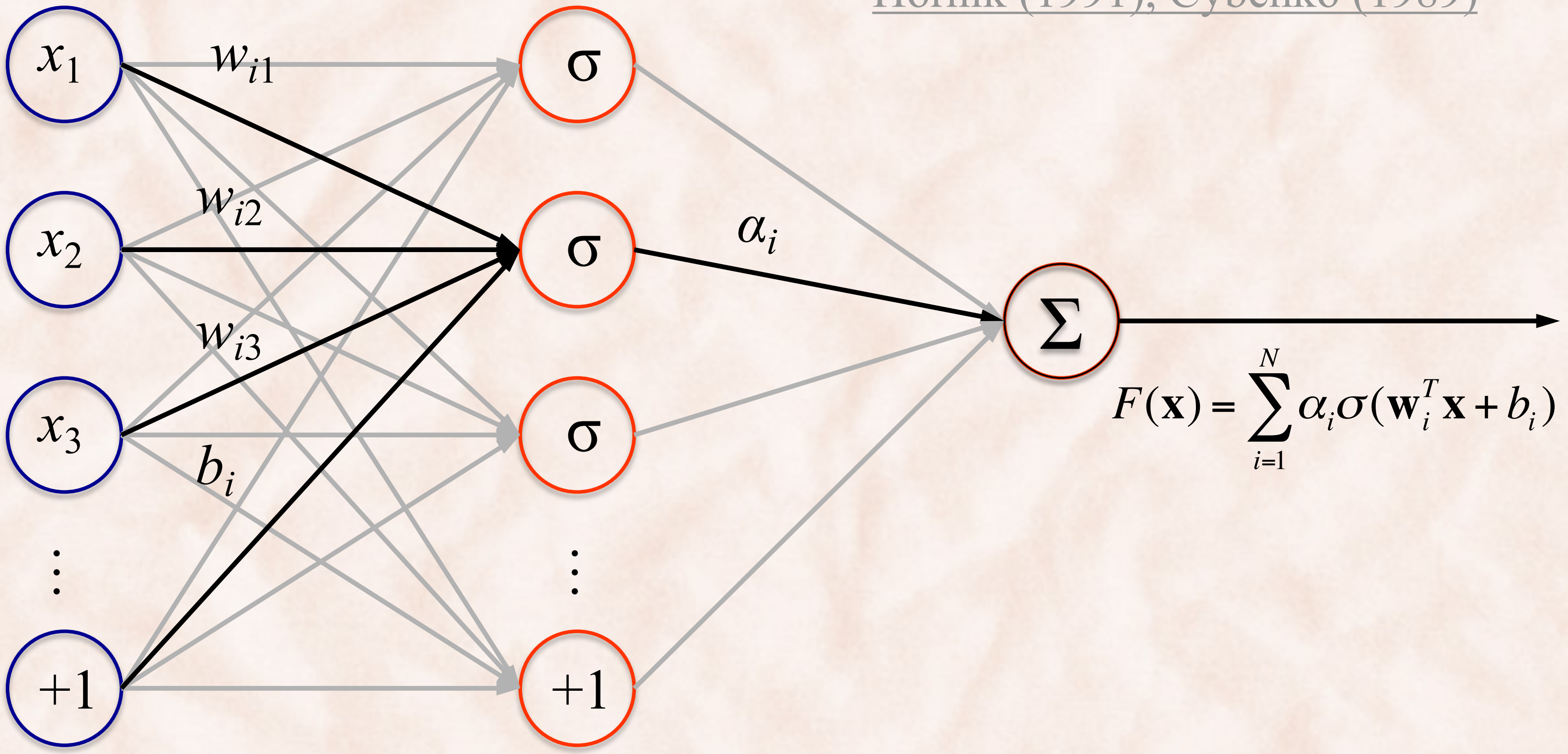
- Put together many neurons in layers, such that the output of a neuron can be the input of another:



input layer ***hidden layer*** ***output layer***

Universal Approximation Theorem

Hornik (1991), Cybenko (1989)



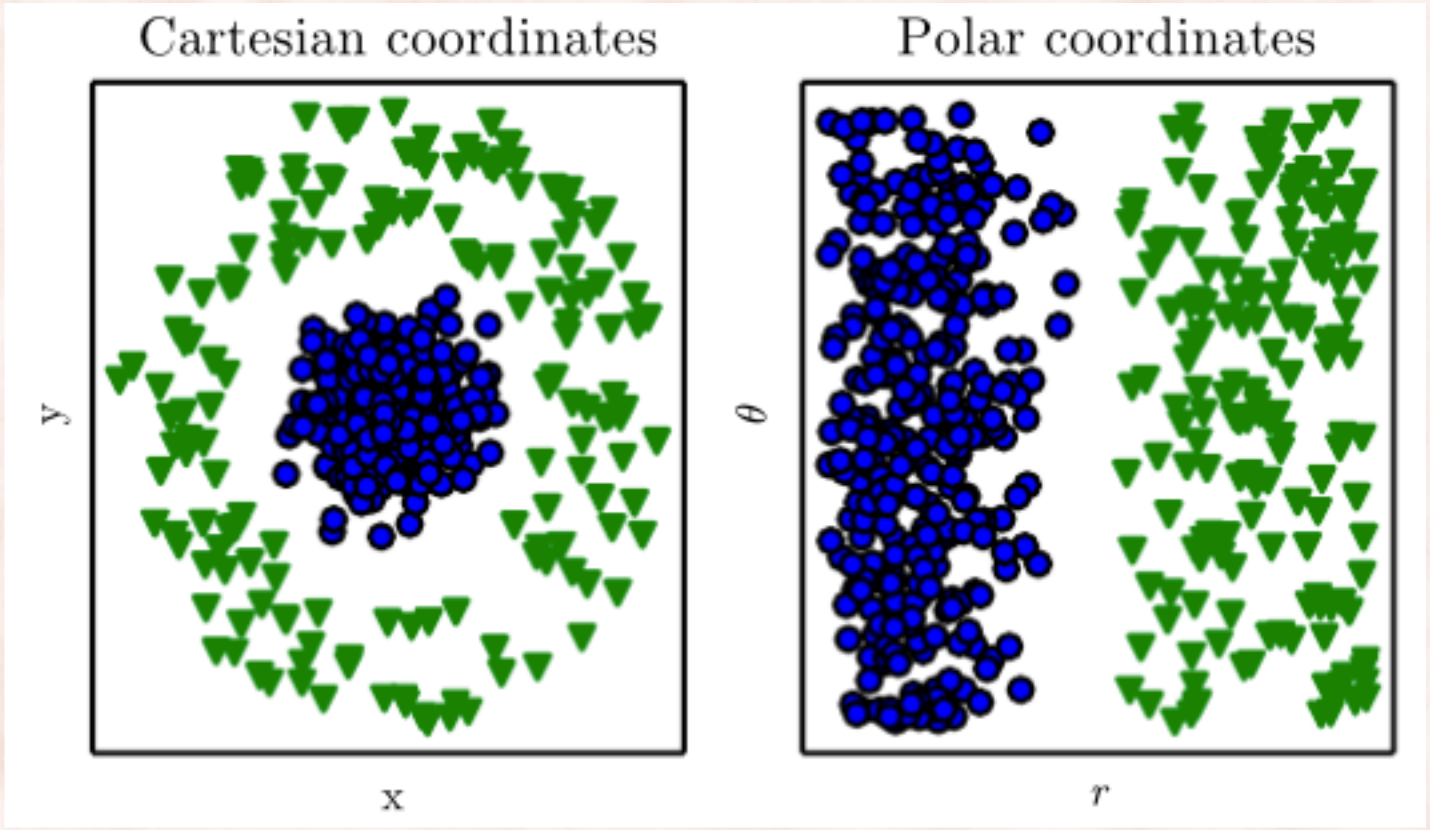
$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w}_i^T \mathbf{x} + b_i)$$

➢**Theorem**: Given any function $f \in C(I_m)$ and $\varepsilon > 0$, there exist an integer N and real constants α_i, $b_i \in R$, $\mathbf{w}_i \in R^m$, where $i = 1, \ldots, N$, such that:

$$\left|F(\mathbf{x}) - f(\mathbf{x})\right| < \varepsilon,\ \forall \mathbf{x} \in I_m$$

The Importance of Representation

http://www.deeplearningbook.org



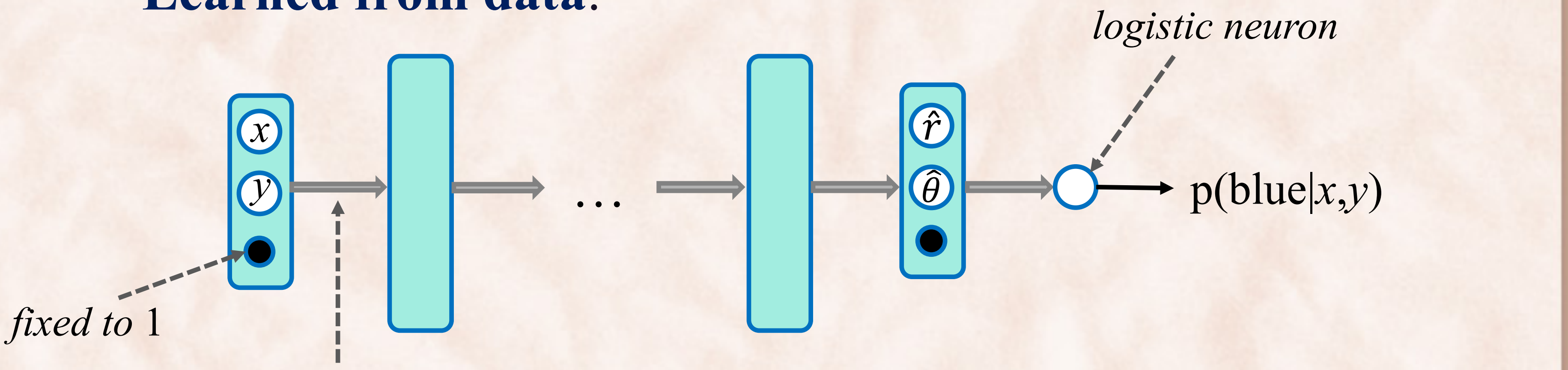
From Cartesian to Polar Coordinates

- **Manually engineered**:

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}\left|\frac{y}{x}\right| \text{ (first quadrant)}$$

- **Learned from data**:



Fully connected layers: linear transformation W + *element-wise nonlinearity f* => *f*(W**x**)

Object Recognition: Cats

Pixels as Features?

φ(x) = [25, 63, 125, 32, 84, 257, ..., 13,
27, 39, 8, 213, 107, 54, 73, ..., 91,
67, 59, 72, 33, 112, 54, 35, ..., 9,
[illegible], ..., 28,
93, 44, 69, 85, 68, 54, 87, ..., 11,
117, 59, 117, 210, 177, 54, 72, ...]^T

Poor recognition accuracy!

- Learning = finding parameters $\mathbf{w} = [w_1, w_2, w_3, \ldots w_k]^T$ such that:

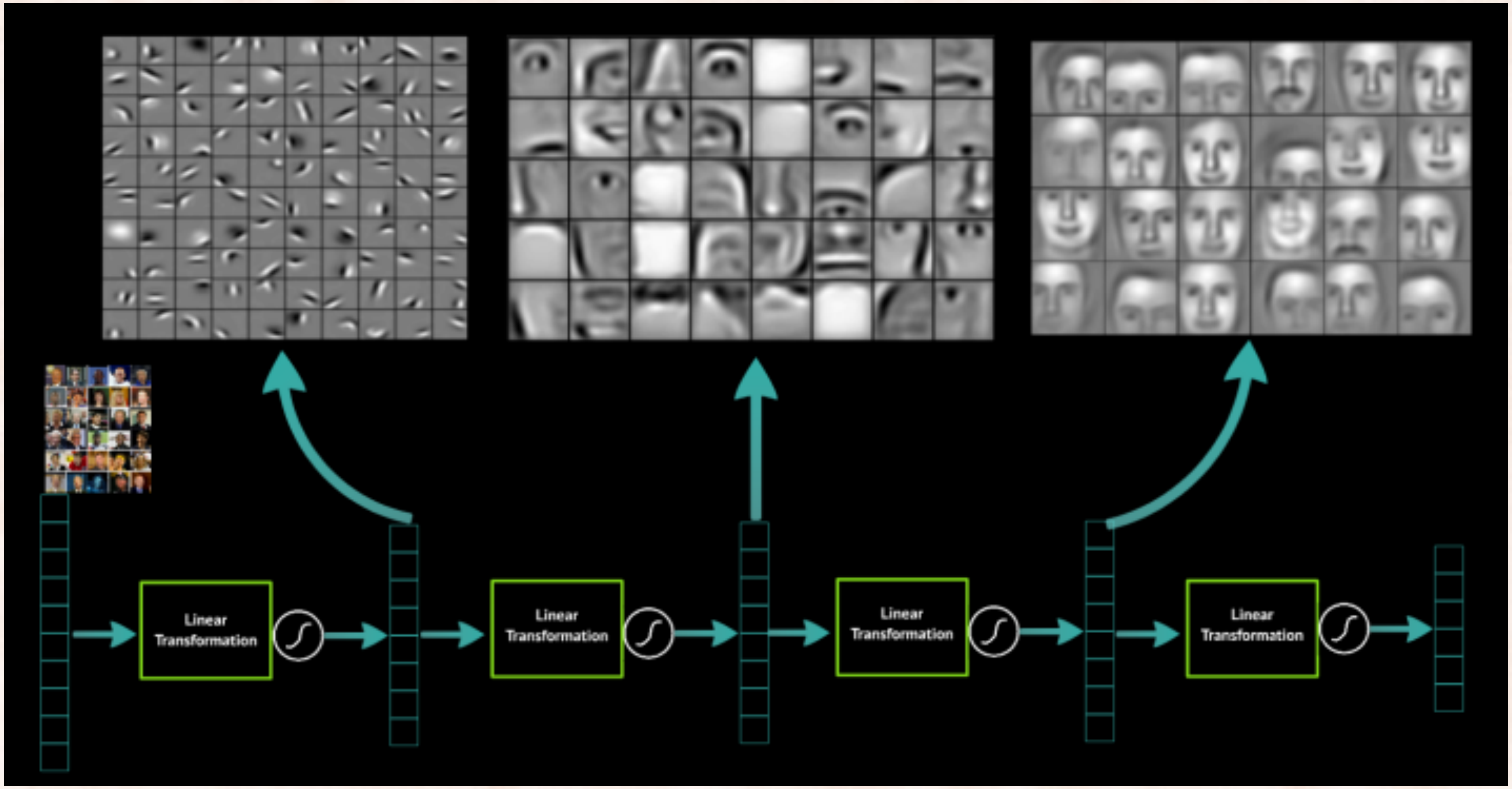
  $\mathbf{w}^T\varphi(x_i) \geq \tau$, if $y_i = +1$ (cat)

  $\mathbf{w}^T\varphi(x_i) < \tau$, if $y_i = -1$ (other)

  where $\mathbf{w}^T\varphi(x) = w_1 \times \varphi_1(x) + w_2 \times \varphi_2(x) + w_3 \times \varphi_3(x) + \ldots w_k \times \varphi_k(x)$

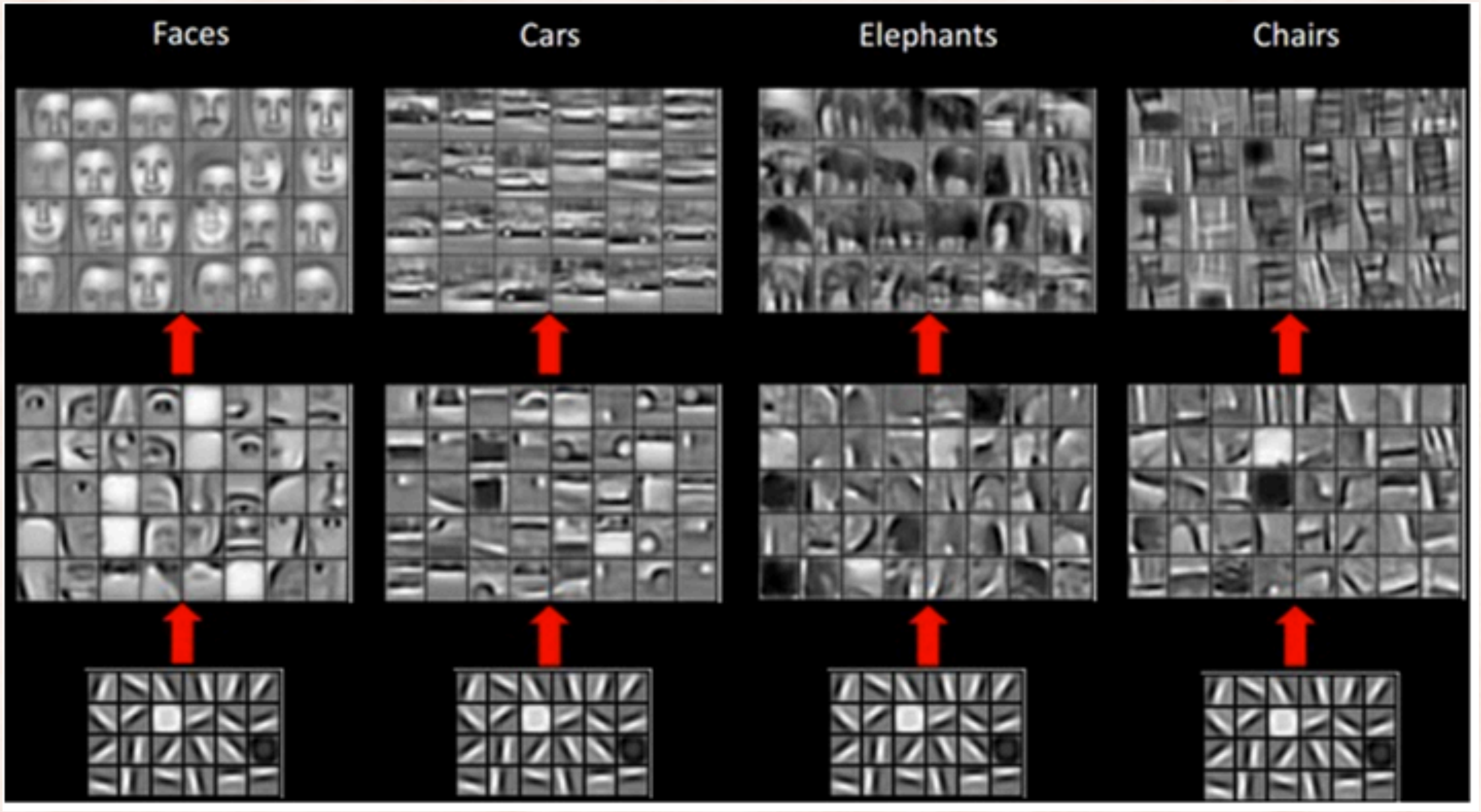
Representation Learning: Images

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



Representation Learning: Images

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



Transformer: Attention is all you need. NIPS 2017.

Representation Learning: Text

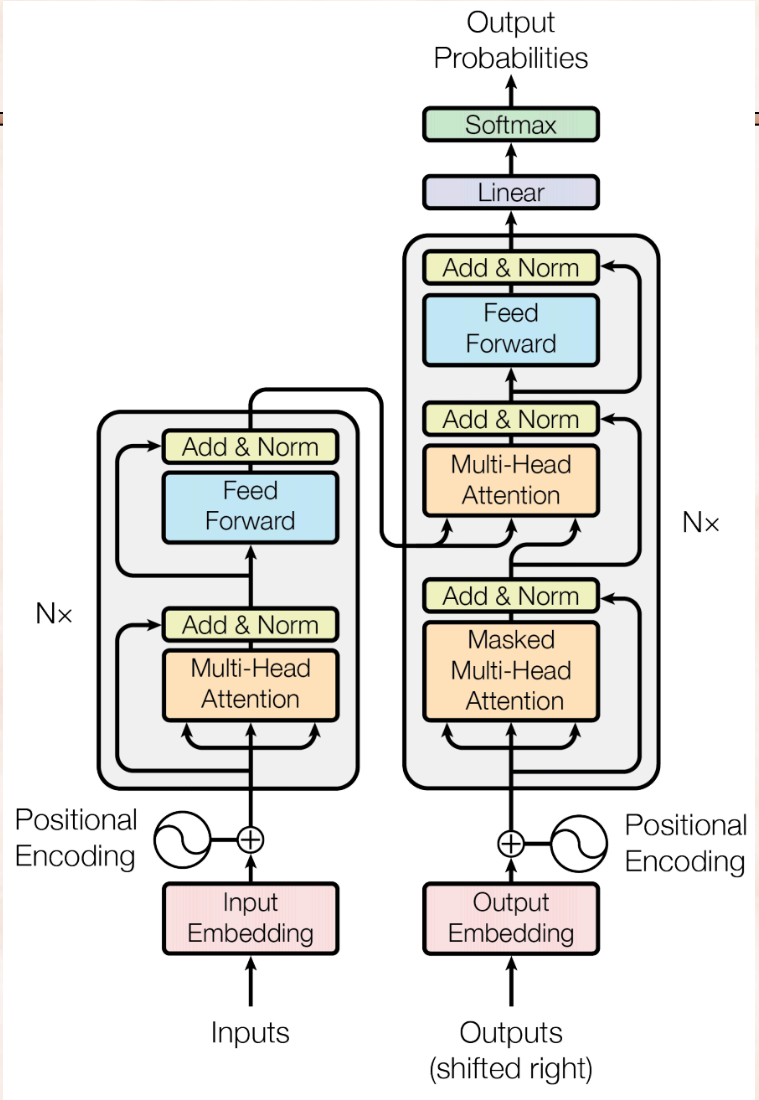


Figure 1: The Transformer - model architecture.

Representation Learning: Text

Transformers: Attention is all you need. NIPS 2017.

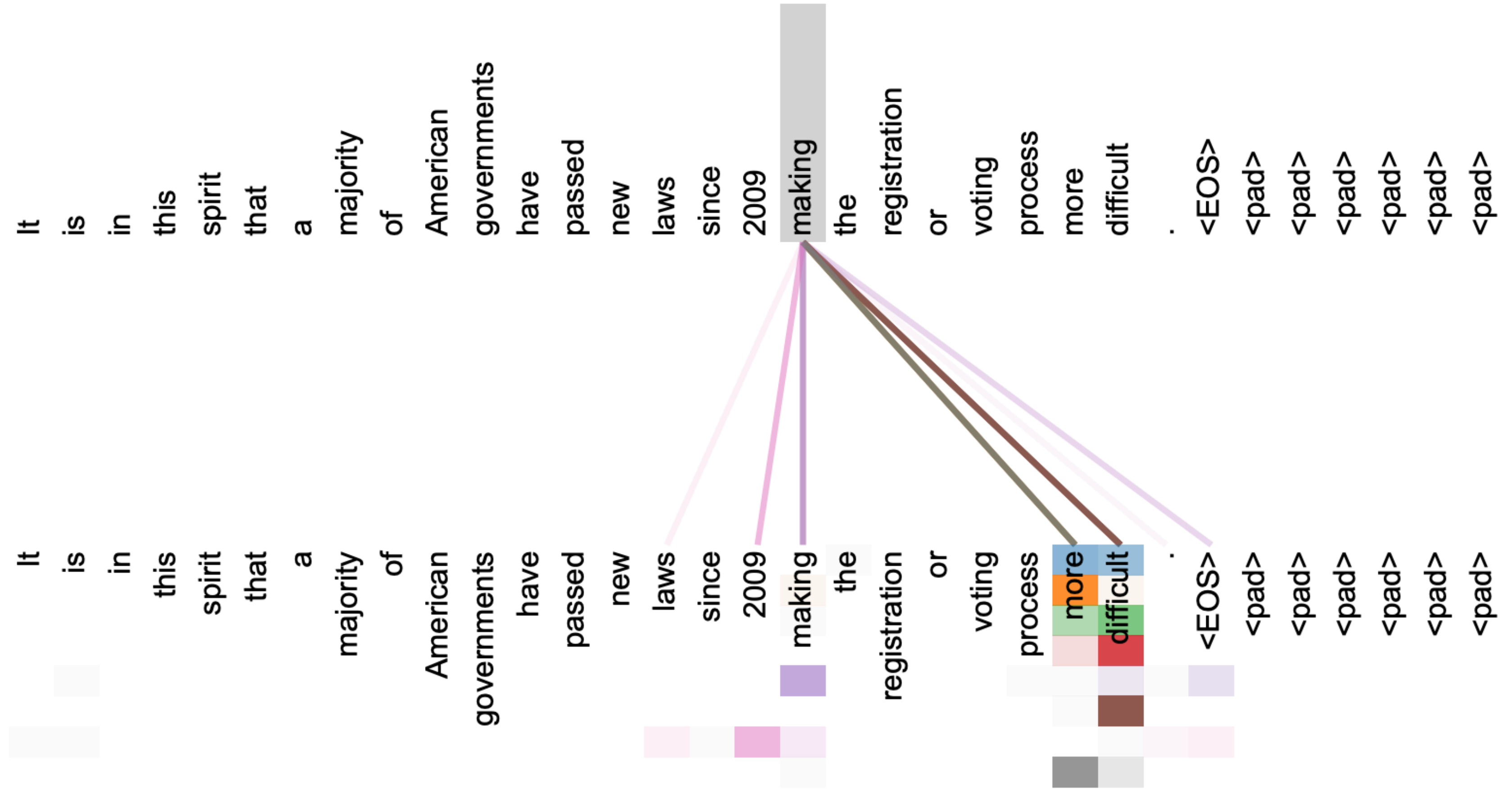


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

Representation Learning: Text

Transformers: Attention is all you need. NIPS 2017.

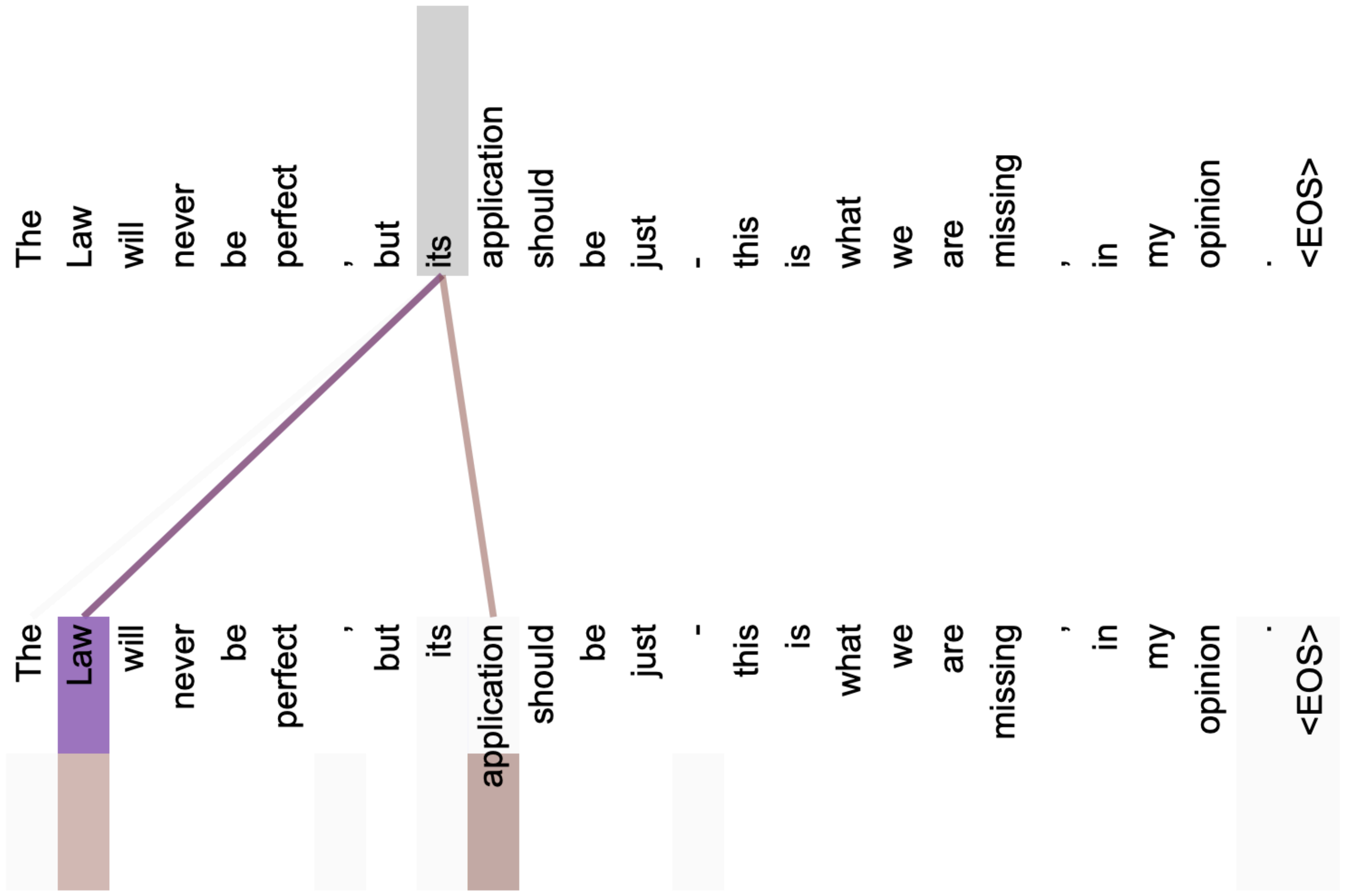
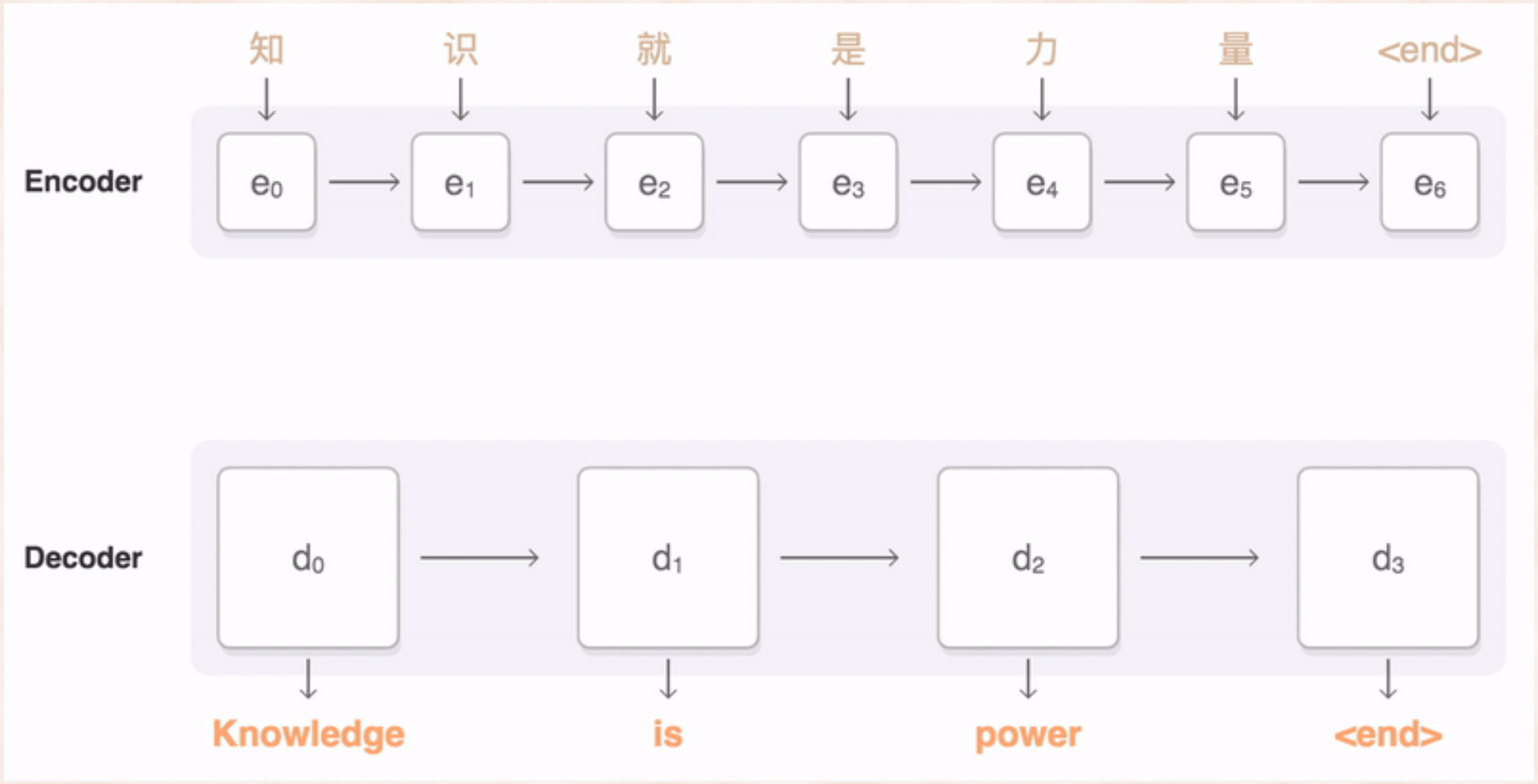


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

Machine Translation

https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

- From Phrase-Based Machine Translation (PBMT) to Neural Machine Translation (NMT):



Machine Translation

http://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html

- **Phrase-Based SMT**:
  - Kilimanjaro is 19,710 feet of the mountain covered with snow, and it is said that the highest mountain in Africa. Top of the west, “Ngaje Ngai” in the Maasai language, has been referred to as the house of God. The top close to the west, there is a dry, frozen carcass of a leopard. Whether the leopard had what the demand at that altitude, there is no that nobody explained.
- **Neural MT (after 2016)**:
  - Kilimanjaro is a mountain of 19,710 feet covered with snow and is said to be the highest mountain in Africa. The summit of the west is called “Ngaje Ngai” in Masai, the house of God. Near the top of the west there is a dry and frozen dead body of leopard. No one has ever explained what leopard wanted at that altitude.

Conditional Text Generation

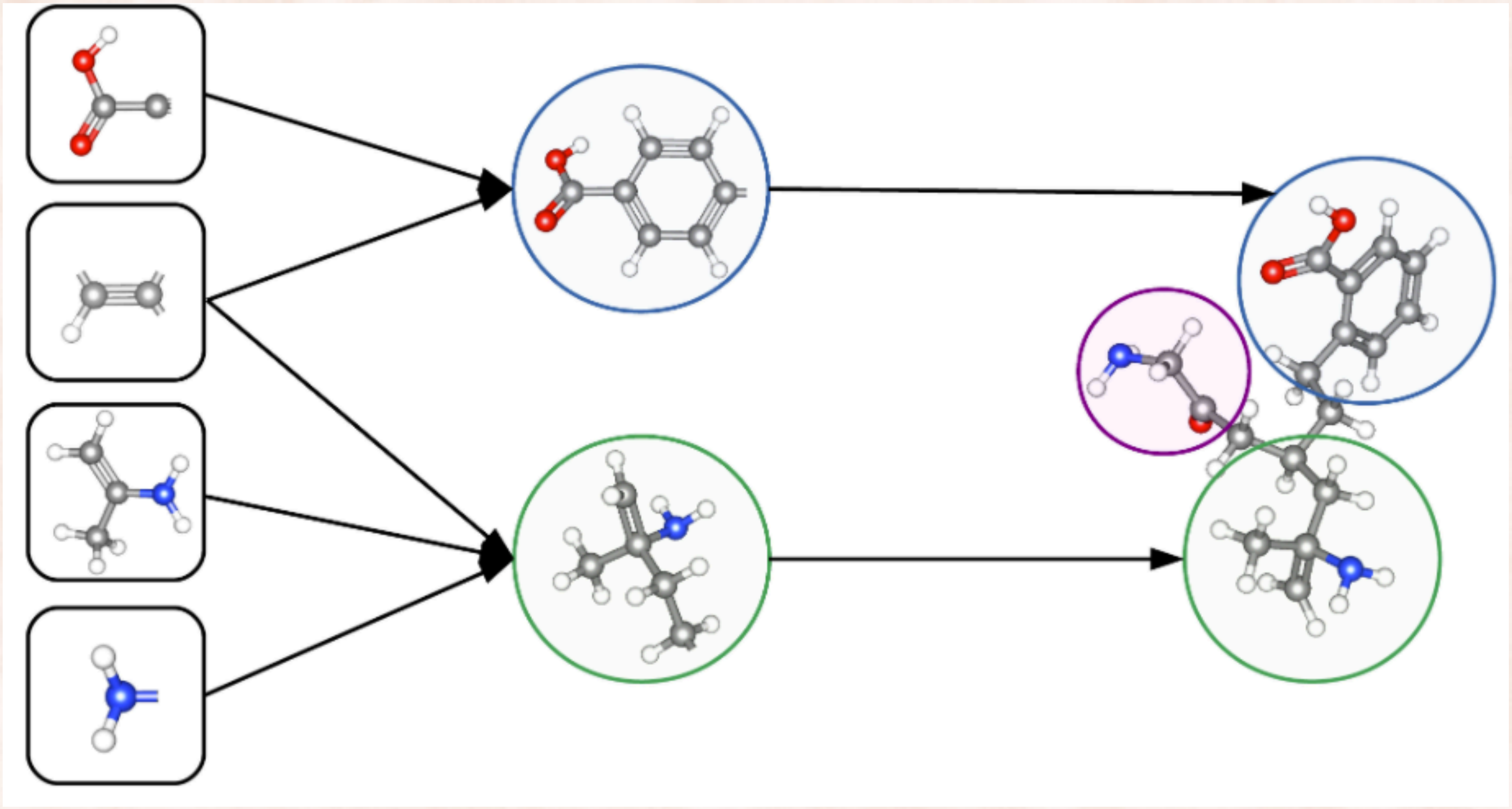
https://openai.com/blog/better-language-models

- **System prompt (human written):**
  - *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*

- **Model completion (machine written):**
  - The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez …

Representation Learning: Chemical Compunds

http://www.bioinf.jku.at/research/DeepTox



What can we do with more data?

We can use the data:

1) To make more accurate predictions.
    - To make better decisions.
        - To live healthier and happier.
2) **To understand things better**.

AI Feynman: a Physics-Inspired Method for Symbolic Regression

[Udrescu & Tegmark, 2019]

- **Symbolic regression**: discovering a symbolic expression that accurately matches a given data set:
  - In 1601, Johannes Kepler got access to the world's best data tables on planetary orbits, and after 4 years and about 40 failed attempts to fit the Mars data to various ovoid shapes, he launched a scientific revolution by discovering that Mars' orbit was an ellipse.
- Designed a recursive symbolic regression algorithm that combines **neural network fitting** with a suite of **physics-inspired techniques**.
  - Discovered all 100 equations from the Feynman Lectures on Physics.

ML is Meta-Programming

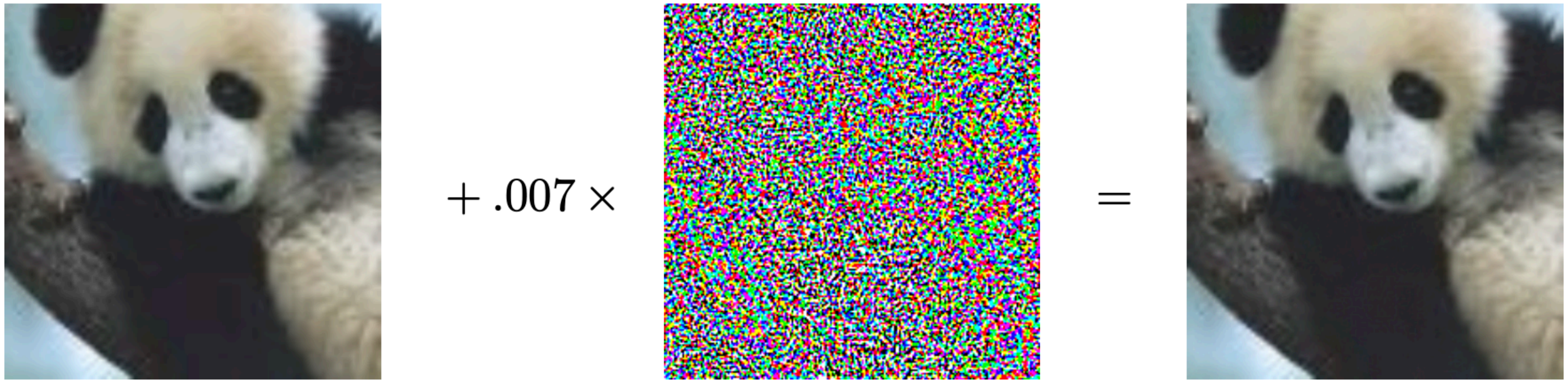
- An ML **model** (e.g. a *neural network*) is a computer **program**:
  - We do not want to explicitly program (model) the computer for each particular task.
  - Use a general ML algorithm and task-specific data to automatically create the Program, i.e. the Model, that solves the task.

$\Rightarrow$ An ML algorithm (e.g. *gradient descent*) is a **meta-program**.

ML is Meta-Programming

- An ML **model** is a computer **program**:
  - Is it **correct**?
    - IID: Can we guarantee a level of **generalization** performance?
    - Non-IID: Can we defend against **adversarial examples**?

Adversarial Examples



$\boldsymbol{x}$

“panda”
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

“nematode”
8.2% confidence

$\boldsymbol{x} +$
$\epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

“gibbon”
99.3 % confidence

: A demonstration of fast adversarial example generation applied to GoogLeNet (

[Goofellow et al., ICLR 2015. Explaining and Harnessing Adversarial Examples]

ML is Meta-Programming

- An ML **model** is a computer **program**:
  - What is its **time complexity**?
  - What is its **memory complexity**?

- An ML algorithm is a **meta-program**:
  - What is its **time complexity**?
  - What is its **memory complexity**?

"Intelligence per Kilowatthour" Max Welling, invited talk at ICML'18

"In the 19th century the world was revolutionized because we could transform energy into useful work. The 21st century is revolutionized due to our ability to **transform information (or data) into useful tools**. Driven by Moore's law and the exponential growth of data, artificial intelligence is permeating every aspect of our lives. But **intelligence is not for free, it costs energy**, and therefore money. Evolution has faced this problem for millions of years and made brains about a 100x more energy efficient than modern hardware (or, as in the case of the sea-squirt, decided that it should eat its brain once is was no longer necessary). I will argue that energy will soon be one of the determining factors in AI. Either companies will find it too expensive to run energy hungry ML tools (such as deep learning) to power their AI engines, or the heat dissipation in edge devices will be too high to be safe. **The next battleground in AI might well be a race for the most energy efficient combination of hardware and algorithms**."

Energy and Policy Considerations for Deep Learning in NLP Strubell, Ganesh, & McCallum, ACL 2019

- “**Exceptionally large computational resources** that necessitate similarly **substantial energy consumption**" needed to develop recent state-of-the-art ML models for NLP:
  - Tuning and training a state-of-the-art machine translation architecture led to CO2 emissions = **5 times the CO2 emissions of an average car over its lifetime**.
  - The R&D cost of a deep NLP architecture (best paper award at EMNLP 2018) led to the CO2 equivalent of **2 years of an average American life**.
    - Estimated to require between $103,000–$350,000 in cloud computing cost and $9870 in electricity cost.

Transfer Learning in Computer Vision and Natural Language Processing

- Trained state-of-the-art models are made publicly available:
  - [CV] AlexNet, ResNet, GoogLeNet, …
  - [NLP] GPT, BERT, …

- Remove softmax layer and use as **pre-trained models** in many other related tasks:
  1. [**Features**] Keep fixed, use last layer as features.
  2. [**Fine-tune**] Use last layer as input to another ML model, fine-tune (train together).

Why Deep Learning so Successful?

- **Large amounts of (labeled) data**:
  - Performance improves with depth.
  - Deep architectures need more data.
- **Faster computation**:
  - Originally, GPUs for parallel computation.
  - Google's specialized TPUs for TensorFlow.
  - Microsoft's generic FPGAs for CNTK.
    - https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/
- **Better algorithms and architectures**.

A Rapidly Evolving Field

- Used to think that training deep networks requires **greedy layer-wise pretraining**:
  - Unsupervised learning of representations with **auto-encoders** (2012).
- Better random **weight initialization** schemes now allow training deep networks from scratch.
- **Batch normalization** allows for training even deeper models (2014).
  - Replaced by the simpler **Layer Normalization** (2016).
- **Residual learning** allows training arbitrarily deep networks (2015).
- Attention-based **Transformers** replace RNNs and CNNs in NLP (2018):
  - **BERT**: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019).

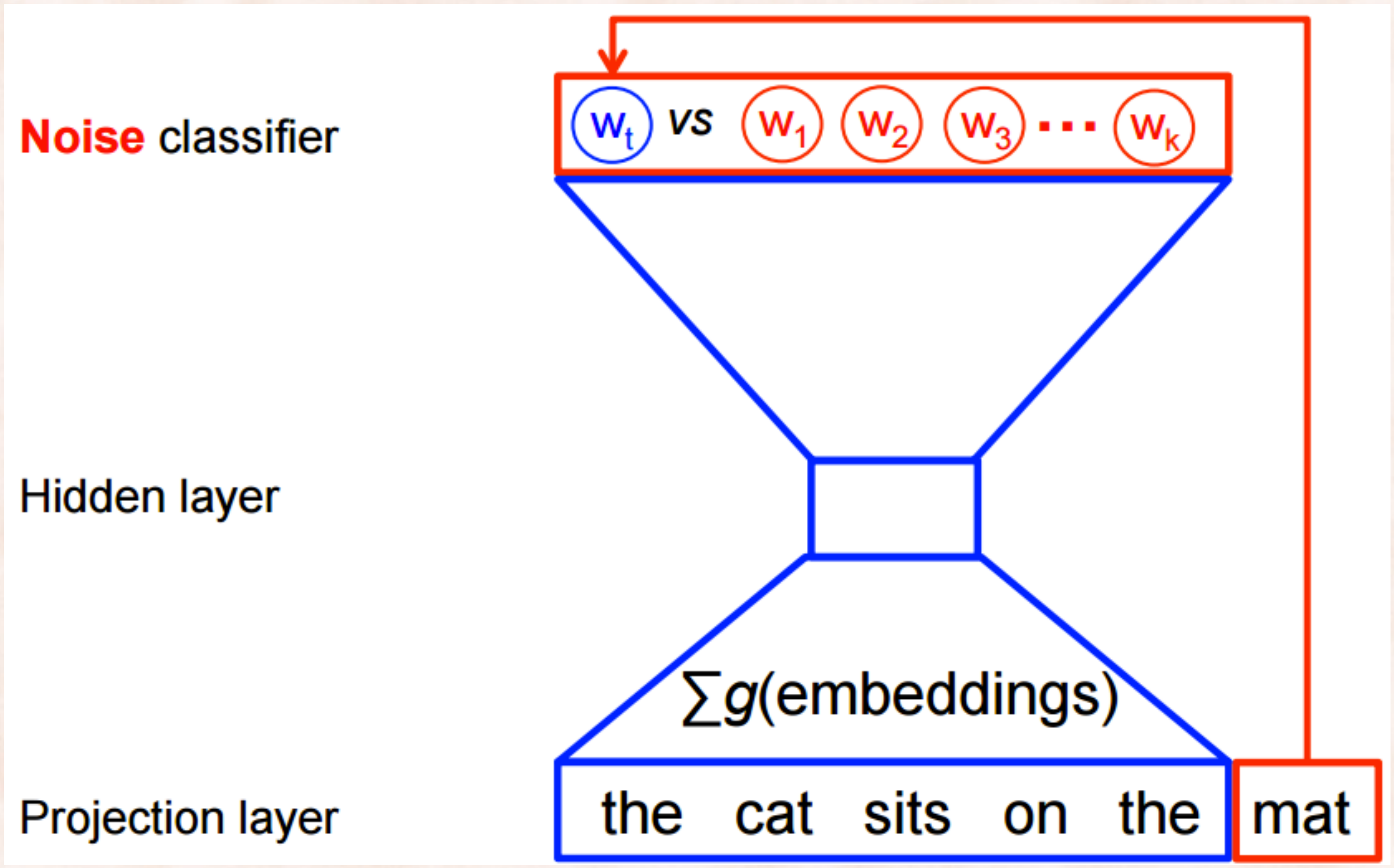
Reading Materials

1) Andrew Ng's introductory presentation, tutorial at UCLA 2012.
   - https://www.youtube.com/watch?v=n1ViNeWhC24

2) Introduction and Part I from the DL textbook, MIT Press 2016.

Representation Learning: Text

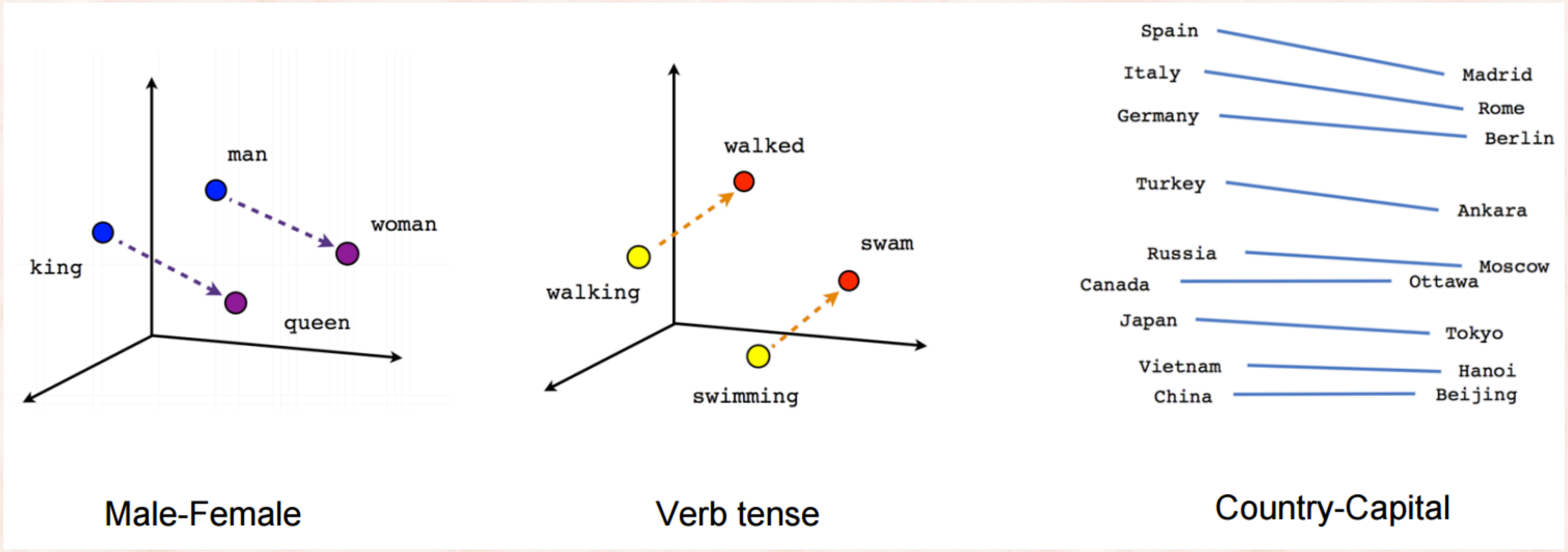
https://www.tensorflow.org/tutorials/word2vec



Representation Learning: Text

https://www.tensorflow.org/tutorials/word2vec

- Word embeddings, projected 2D through PCA:


man
woman
king
queen
Male-Female
walked
swam
walking
swimming
Verb tense
Spain
Madrid
Italy
Rome
Germany
Berlin
Turkey
Ankara
Russia
Moscow
Canada
Ottawa
Japan
Tokyo
Vietnam
Hanoi
China
Beijing
Country-Capital