Review for Exam 2

1

Symbolic AI Techniques

2

Forward Chaining Inference

- Tell-Ask Systems

King(John)
∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]
+
Inference Engine
(Forward Chaining)

← Tell Greedy(John)

→ Evil(John)!

3

3

Backward Chaining Inference

- Tell-Ask Systems

King(John)
Greedy(John)
∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]
+
Inference Engine
(Backward Chaining)

← Ask Evil(John)?

→ Yes!

4

4

PROLOG Knowledgebase

```
isa(X, mammal) :-
    has(X, hair).

isa(X, bird) :-
    has(X, feathers),
    flies(X).

isa(X, carnivore) :-
    isa(X, mammal),
    has(X, pointed_teeth),
    has(X, claws),
    has(X, forward_pointing_eyes).

isa(X, ungulate) :-
    isa(X, mammal),
    chews_cud(X);
    has(X, hoofs).

isa(X, cheetah) :-
    isa(X, carnivore),
    color(X, tawny),
    has(X, dark_spots).

isa(X, giraffe) :-
    isa(X, ungulate),
    has(X, long_legs),
    has(X, long_neck),
    color(X, tawny),
    has(X, dark_spots).
```

```
% Facts:
has(stretch, hair).
chews_cud(stretch).
has(stretch, long_legs).
has(stretch, long_neck).
color(stretch, tawny).
has(stretch, dark_spots).

has(swifty, hair).
has(swifty, pointed_teeth).
has(swifty, claws).
has(swifty, forward_pointing_eyes).
color(swifty, tawny).
has(swifty, dark_spots).
```

5

Vocabulary

Knowledge Engineering
FOPC
Knowledge Base
Tell-Ask Systems
Forward Chaining Inference
Backward Chaining Inference
Definite Clauses
Logic Programming
PROLOG

6

6

Meaning Representation Systems

- Logic
- Semantic Networks
- Frames
- Conceptual Dependency
- many many others…
- Japan's Fifth Generation Project, and several others
- Successes of KR&R
- Limitations of KR&R

7

7

Frames – Organized in a Hierarchy/Network

Restaurant
Name: Gulliftys
Address: 1149 Lancaster Ave, Bryn Mawr, PA 19010
Phone: (610) 525-1851
Website: http://gulliftys.com
Cuisine Type: American
Seating Capacity: 120
…etc…

isa

Restaurant

has

Menu
Salads: …
Main Dishes:
Entrée: Chicken Cacciatore, $23.00
Entrée: Hamburger, $12.00
Entrée: etc.
Desserts:
Dessert: Rice Pudding, $8.00
Dessert: Crème Caramel, $8.00
Beverages:
Soda: Coke, $3.00
Hot Beverage: Tea, $3.50

John went to Gulliftys.
He sat down.
He read a menu.
The server came to take his order.
He ordered a hamburger.
He ate the hamburger.
He paid the check.
He left a tip.
He left the restaurant.

8

8

CD – Representations & Inferences

- *John went to New York.*

  ACTOR: John
  ACTION: PTRANS
  OBJECT: John
  DIRECTION TO: New York
  FROM: unknown

- John bought a book from Mary.

  ACTOR: John
  ACTION: ATRANS
  OBJECT: money
  DIRECTION TO: John
  FROM: Mary

- John read a book.

  ACTOR: John
  ACTION: ATTEND
  OBJECT: eyes
  DIRECTION TO: book
  FROM: unknown

- John drank a glass of milk.

  ACTOR: John
  ACTION: INGEST
  OBJECT: milk
  DIRECTION TO: mouth of John
  FROM: glass

Instrument:
ACTOR: John
ACTION: PTRANS
OBJECT: glass containing milk
DIRECTION TO: mouth of John
FROM: table

Instrument:
ACTOR: John
ACTION: MOVE
OBJECT: hand of John
DIRECTION TO: glass
FROM: unknown

Instrument:
ACTOR: John
ACTION: GRASP
OBJECT: glass of milk
DIRECTION TO: hand of John
FROM: unknown

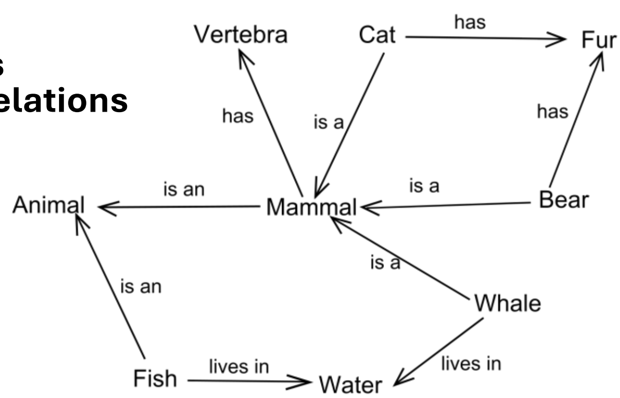
Semantic Networks

- Represents semantic relations between concepts in a network.

  It is a **graph**
  vertices represent **concepts**
  edges represent **semantic relations**

- Also represented as a **triple**:

  <entity, relation, entity>
  <mammal, has, vertebra>
  <bear, is a, mammal>
  <fish, lives in, water>



From: https://en.wikipedia.org/wiki/Semantic_network#/media/File:Semantic_Net.svg

Large-Scale AI/KR&R Efforts

- **Japanese Fifth Generation Project**

  10-year project started in 1982.
  Creating supercomputers for future AI development using PROLOG.
  Develop Knowledge Information Processing Systems.
  Spent ~$300 million over twelve years.
  (IBM's research expenditure in 1982 was $1.5 billion!)

- **Limited Results**

  Developed foundations for concurrent logic programming.
  Developed Kappa (a parallel DBMS)
  Developed an automated theorem prover, MGTP
  Attempt at applications in bioinformatics.

- **Did not meet commercial success**

  The arrival of SUN Workstations and x86 based machines far surpassed the abilities.
  Failed on attempts to develop concurrent KR&R systems.

- **Perhaps the ideas were far ahead of its time.**

  For example, current multi-core processors being used in current AI work.

11

CYC – The Ultimate Expert System

- Methodology

  Developed a representation language, CycL.
  Developed a set of representations (ontological engineering)
  Developed a massive knowledgebase comprising human knowledge.
  Connected CYC's knowledge to Wikidata (Wikipedia) and other large knowledgebases.
  An inference engine.

12

The Seasons of AI

- **1950s – 1966 First AI Summer: Irrational Exuberance**

  Early successes in game playing, theorem proving, problem solving

- **1967 – 1977 First AI Winter**

  No useful deliverables led to loss of research funding and cancellation of AI programs. In UK *The Lighthill Report* (toy AI systems do not scale due to combinatorial explosion).

- **1978 – 1987 Second AI Summer/Spring**

  Rise of knowledge-based systems, success of Expert Systems. Boom times.

- **1988 – 1993 Second AI Winter**

  Failure of AI Hardware companies (Symbolics, LMI, Lisp Machines) and AI Companies (Teknowledge, Inference Corp. etc.) Commercial deployments of Expert Systems were discontinued.

- **1993 – 2011 Third AI Summer (Mostly academic advances)**

  Statistical approaches and extensions to logic (Bayesian Nets), Non-Monotonic Reasoning (in Logic), Fuzzy Logic, advances in Machine Learning (Decision Trees, Random Forests, Neural Nets), Cognitive Models, Logic Programming, Case-Based Reasoning, Genetic Algoritms, Agent-based approaches, etc.

- **2011 – Now Third AI Spring**

  Rise of Deep Learning, Neuro-symbolic AI, ChatGPT and other chatbots, generative AI.

13

13

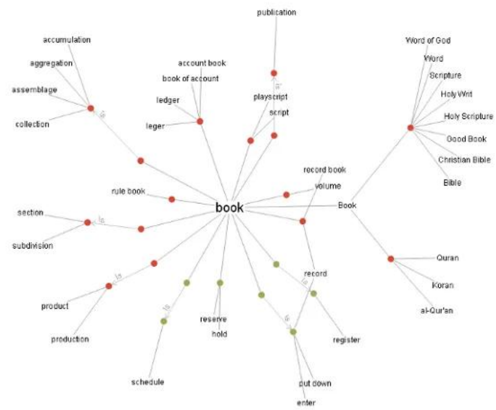
Towards Usable Representations (1985-now)

- **Wordnet**, 1985 (Princeton U.)

  A lexical database of semantic relations between words.

  Over 150,000 organized in 207,000 word-sense pairs (eat-out, car-pool).

  Now available for multiple languages.



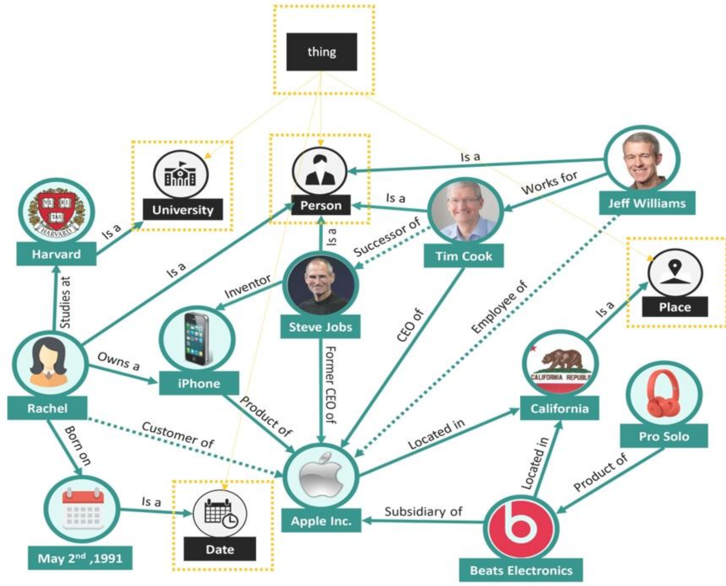
14

14

An Example KG

Concepts, relations etc. in a KG can be obtained from and linked to several disparate structured Sources of data:

ConceptNet
ATOMIC
Wikidata
WordNet
Roget
VerbNet
FrameNet
VisualGenome
ImageNet



From: https://atonce.com/blog/knowledge-graph

15

15

Commonsense Knowledge

On stage, a woman takes a seat at the piano. She

1. sits on a bench as her sister plays with a doll.
2. smiles with someone as the music plays.
3. is in the crowd, watching the dancers.
4. nervously sets her fingers on the keys.

Which one?

Answering the question requires knowledge that humans possess and apply, but machines cannot distill from the communication.

Also, remember Winograd schemas?

16

16

Review: Meaning Representation Languages

Knowledge Engineering
FOPC
Knowledge Base
Tell-Ask Systems
Forward Chaining Inference
Backward Chaining Inference
Definite Clauses
Logic Programming
PROLOG

Frames
Conceptual Dependency
Semantic Networks

5th Gen. Project, SGI, Alvey, ESPRIT
CYC

Wordnet
Knowledge Graphs

Commonsense Knowledge

17

17

Behavioral AI

- **Subsumption Architecture**

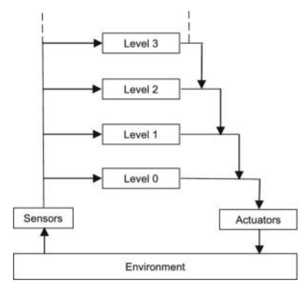
  Situatedness
  Embodiment
  Intelligence bottom up
  Emergence

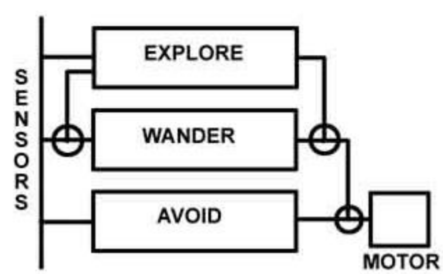
- Layers of control

  All layers may have an action to suggest.
  Only one will be carried out at any time.
  The action from the "lowest" (highest) module.
  E.g. AVOID subsumes WANDER.
  WANDER subsumes EXPLORE.

- Built several robots: Allen, Herbert, Tom & Jerry, Seymour, **Genghis**, Squirt.

Genghis Link: https://www.youtube.com/watch?v=1j6CliOwRng

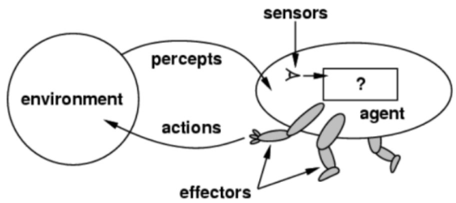




18

18

Agent Types



- **Table-Driven Agents**
  Use a **percept-action table** in to find next action.
- **Simple Reflex Agents**
  Based on **condition-action rules**
- **Agents with Memory**
  Have **internal states** that are used to keep track of past states of the world.
- **Agents with Goals**
  Have **state and goal information** to take future states into consideration.
- **Utility-Based Agents**
  Use **utility theory** to act rationally.
- **LLM-Based Agents**
  Carry out actions recommended by an LLM.

Other Successful Approaches

- Rational Agents (Utility based agents)
- Bayesian Inference
- 1997: IBM's Deep Blue beat Garry Kasparov
- 2012: IBM's Watson wins Jeopardy!

Robot Architectures: Vocabulary

Intelligence w/o
Representation
Behavioral AI
Subsumption Architectures
Genghis, Cog
NASA Pathfinder
NASA Spirit & Opportunity
NASA Curiosity, Ingenuity
Boston Dynamics Big Dog
Boston Dynamics Spot, Atlas
Agent-Based AI
Types of Agents
Agentic AI
IBM Watson

21

21

Subsymbolic AI

22

Symbolic *versus* Subsymbolic AI

- **Symbolic AI**

  Everything is represented using symbols.

  A is a block `Block(A)`

  Representation of a state 

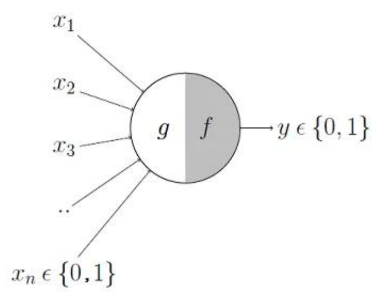
  Expert Systems, Frames, Scripts, Semantic Nets, Knowledge Graphs etc.

- **Subsymbolic AI**

  There are NO SYMBOLS.

  Approaches that employ Neural Networks and other statistical mechanisms

McCulloch-Pitts Neuron, 1943



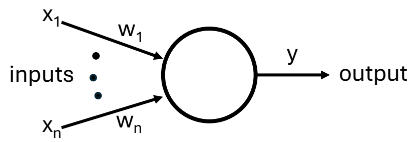
- Binary Threshold Units
- Captures the inhibitory and excitatory connections between biological neurons.
- Limited in what such a model can actually do.
- Missing the learning capability: how to model changes in inhibitory and excitatory connections. This was later included in Hebb's model (1949). Repeated firings can modify the nature of the connections.
- Frank Rosenblatt, 1958 combined these ideas into the model of a **Perceptron**.

$$g(x_1, x_2, x_3, \ldots, x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

From: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

The Perceptron – A Gross Approximation (Rosenblatt, 1958)



- A single "neuron" (**unit**)
  aka Threshold Logic Unit (TLU)

- **Transfer Function**
  T is the Threshold value (assume $T = 0$)

$$I = \sum_{i=1}^{i=n} w_i x_i$$

$$y = \begin{cases} +1, \text{if } I \geq T \\ -1, \text{if } I < T \end{cases}$$

Perceptron Learning Rule

- Changes the weights

$\overline{\boldsymbol{w}} = [w_1, w_2]$	weight vector
$\overline{\boldsymbol{x}} = [x_1, x_2]$	input vector
$\overline{\boldsymbol{w_{new}}} = \overline{\boldsymbol{w_{old}}} - y^{\star}\, \overline{\boldsymbol{x}}$	Training Rule

Perceptron: Vocabulary

- **Labelled Training Dataset**
  N samples/patterns/input vector with desired outputs (targets/labels)
- **Output Error (Loss)**
  Error = Desired Output – Actual Output
- **Learning Rule**
  Specifies change in the weights using the Error
- **Prediction/Forward Pass**
  Application of a pattern to produce output
- **Epoch**
  1 pass through the training dataset

27

Perceptron Training Algorithm

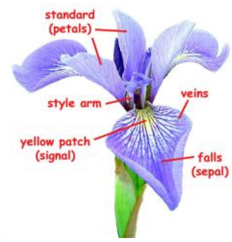
Initialize all weights to random values
    #In what range? Typically [-1.0..1.0]
Set #Epochs to some N
    // How to decide what N should be?
Do N times or until all outputs are correct
    Do for each pattern in the training set
        apply the pattern to the perceptron
        change the weight vector as defined

28

Example – The Iris Dataset

- 150 Samples, 50 of each variety

Labelled: 0 (Setosa), 1(Versicolor), 2(Virginica)



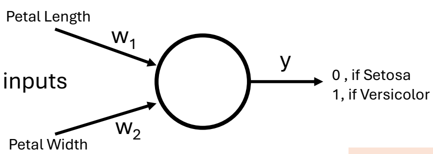
From: https://www.fs.usda.gov/wildflowers/beauty/iris/flower.shtml

Sepal Length & Width

```
[5.1, 3.5, 1.4, 0.2],
[4.9, 3. , 1.4, 0.2],
[4.7, 3.2, 1.3, 0.2],
[4.6, 3.1, 1.5, 0.2],
...
```

Petal Length & Width

https://peaceadegbite1.medium.com/iris-flower-classification-60790e9718a1



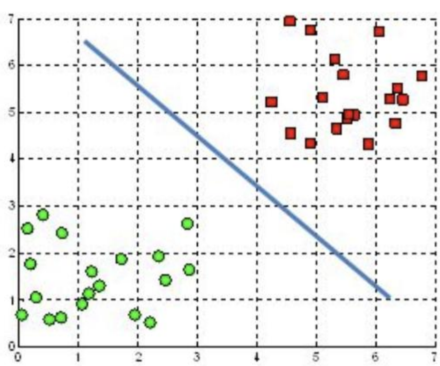
Parameters = 2
Hyperparameters: # of epochs

29

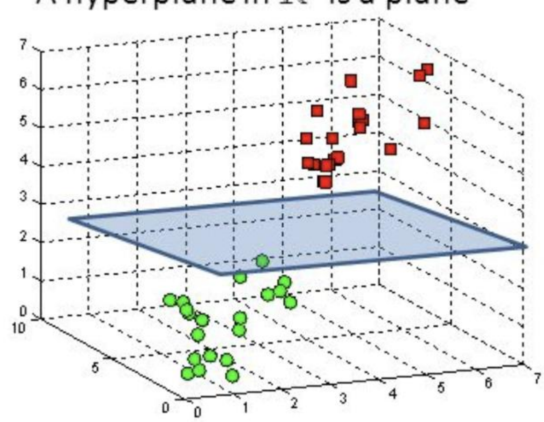
29

Perceptron Learning

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



30

30

Another Example – MNIST Dataset

- 70,000 images of handwritten digits
- Each image is 28x28 pixels
- Each pixel is in the range [0..255]
  0 = white, 255 = black. Greys in between.
- Training set: 60,000 images
- Testing set: 10,000 images
- Task: Given an image, classify it as [0,1,...,9]

Limitations of Perceptrons

- Limited to binary classification tasks only
- *Perceptrons*, by Minsky & Papert, 1969

  Types of problems Perceptrons could solve were limited to linearly separable problems. Real world problems are not linearly separable.

  Perceptron Learning Algorithm would not scale up to tasks requiring large number of weights and thresholds.

  For networks with three or more layers there is no obvious way of knowing what the desired output of hidden layers should be.

  There is no training procedure possible for networks of three or more layers.

  Led to drying up of funding in neural network research in the 1970s.

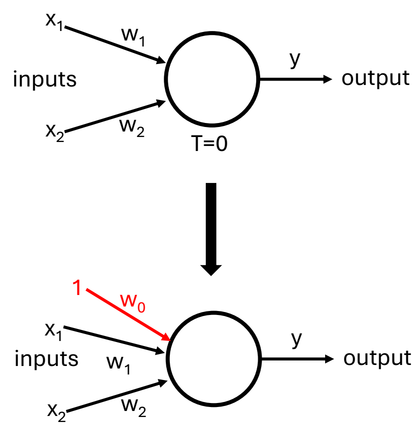
Introducing Bias

- Instead of using an arbitrary Threshold value, we can turn it into an input (=1)
- The weight on the bias, w_0 can then be learned using the same algorithm.

$$\overline{\boldsymbol{w}} = [w_0, w_1, w_2]$$
$$\overline{\boldsymbol{x}} = [1, x_1, x_2]$$

- More often, the net input is determined using the following (and no bias is used for output layer):

$$I = \sum_{i=1}^{i=n} w_i x_i + \overline{\boldsymbol{b}}$$



33

33

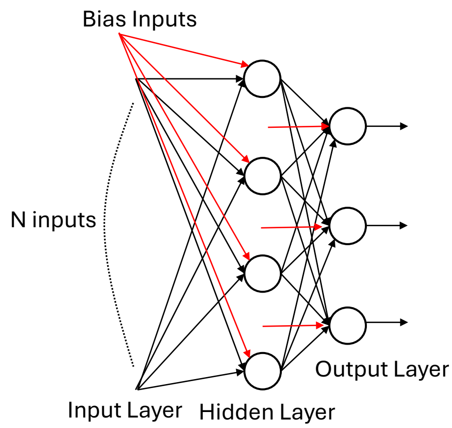
Multi-Layer Perceptron Network

- **Example:** This could be a network that can recognize all three categories of irises from the Iris dataset.

  4 inputs, 3 outputs (Hyperparameters)
  4x4 (input to hidden) + 4x3 (hidden to output) weights + 7 bias inputs

  #Parameters = 16+12+7 = 35

- Since all units are linear TLUs this network can only learn linear functions.
- We need to make each unit non-linear.



34

34

Backpropagation Network (Classic Version)

- **Net Input**

$$I = \sum_{i=1}^{i=n} w_i x_i + \bar{\boldsymbol{b}}$$

- **Activation Function** (Sigmoid)

$$\mathrm{f(I)} = \frac{1}{1 + e^{I}}$$

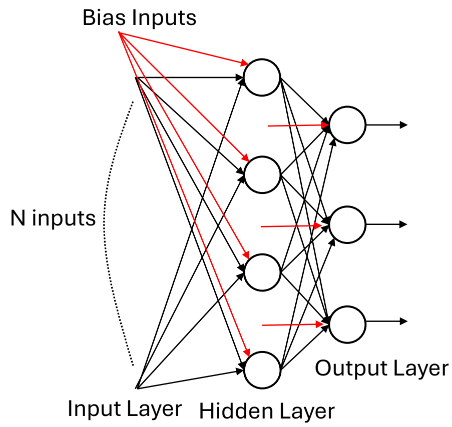
- **Learning Rule**

$$\Delta w_{ij} = \beta * \mathrm{E} * \mathrm{f}(\mathrm{I}_j)$$

- **Error/Loss**

$$\mathrm{E}_{\mathrm{j}}^{\mathrm{output}} = y_j^{\mathrm{desired}} - y_j^{\mathrm{actual}}$$

$$\mathrm{E}_{\mathrm{i}}^{\mathrm{hidden}} = \frac{\mathrm{df}(\mathrm{I}_{\mathrm{i}}^{\mathrm{hidden}})}{\mathrm{dI}} \sum_{j=1}^{n} (w_{ij} \mathrm{E}_{\mathrm{j}}^{\mathrm{output}})$$



35

35

Backpropagation (Classic) Training Algorithm

set minimum acceptable error and #epochs to train, set β
set #epochs = 0

repeat

total error = 0

for each pattern in training set **do**
do a forward pass
for each unit in the hidden layer do
compute net input I, and activation, f(I)
save f(I) for backpropagation
for each unit in the output layer do
compute net input I, and activation f(I)
output y = f(I)

do backward pass
for each unit in the **output layer** do
compute error = desired – actual output ($\mathrm{E}_{\mathrm{j}}^{\mathrm{output}} = y_j^{\mathrm{desired}} - y_j^{\mathrm{actual}}$)
total error = total error + error

for each unit in the **middle layer do**
compute incoming error = weighted sum of output later errors ($\sum_{j=1}^{n}(w_{ij}\mathrm{E}_{\mathrm{j}}^{\mathrm{output}})$)
compute final error = incoming error * f(I) * (1 – I) [derivative)

for each unit in the **output layer** do
for each weight from a hidden layer to unit do
compute weight change $\beta *$ error $* f(\mathrm{I})$ and update weight

for each unit in the **middle** layer do
for each weight from an input layer unit do
compute weight change $\beta *$ final error $* \mathrm{f(I)}$ and update weight

#epochs = #epochs + 1

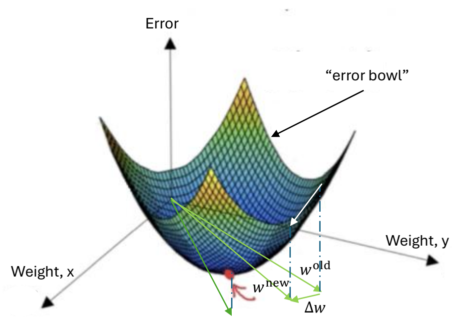
until total error < maximum acceptable error or #epochs reaches limit

36

36

Backpropagation: Gradient Descent

- Learning in a neural network using Backpropagation is essentially a Gradient Descent process.
- Each change in the weights is an attempt to reduce error and descend into the lowest possible position in the "error bowl" (as shown in a 2-D weight vector case)
- In higher dimensional weight vectors (typical ML situations), the error surface can be quite complex.



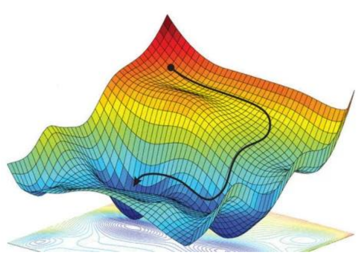
From: https://builtin.com/data-science/gradient-descent

37

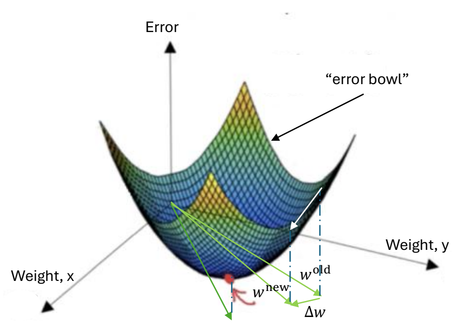
37

Backpropagation: Gradient Descent

- In higher dimensional weight vectors (typical ML situations), the error surface can be quite complex.



From: https://poissonisfish.com/2023/04/11/gradient-descent/



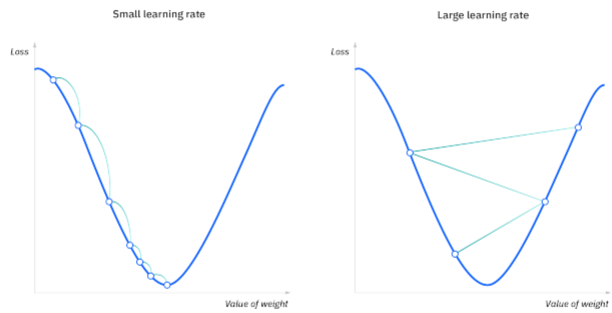
From: https://builtin.com/data-science/gradient-descent

38

38

Backpropagation: Learning rate

- Learning Rate, β (0 < 1)
- The value of β determines how fast or slow the gradient descent takes place.
- Typically, one starts with a higher value (say 0.5 or 0.6) and then decrease it as the learning/epochs progresses. This is called a **Learning Rate Schedule**.

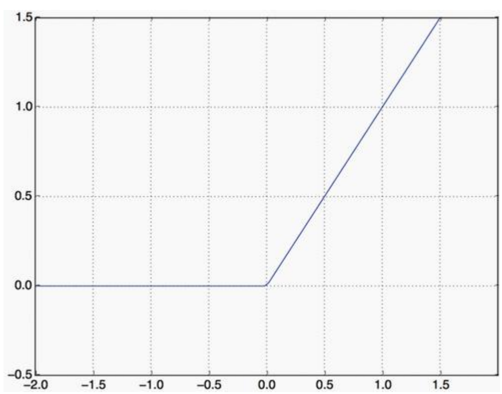


From: https://www.ibm.com/topics/gradient-descent

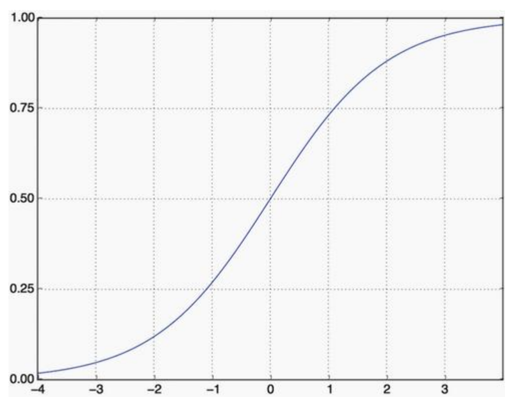
39

39

Popular Activation Functions



Relu – Rectified Linear Unit: $f(I) = \max(0, I)$



Sigmoid: $f(I) = \frac{1}{1+e^{I}}$

40

40

Softmax Activation Function

- Transforms a vector of arbitrary numbers into a probability distribution.

  Each value is in [0.0..1.0]
  Summ of all values is 1.0

  Useful for multi-class classification problems. E.g., Recognizing handwritten digits (ten possible outcomes [0, ..9])

Output layer | Softmax activation function | Probabilities

$$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \\ 0.7 \\ 1.1 \end{bmatrix} \longrightarrow \boxed{\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}} \longrightarrow \begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$$

41

Backpropagation: Vocabulary

- In what order do we present the patterns? As they are in the training set? Or, randomly?
  If patterns are chosen at random (without replacement), we call it **Stochastic Gradient Descent (SGD)**

  Choices:

  Do a backpropagation pass after every input. **True SGD**

  Do the backward pass after all the inputs have been seen, and errors recorded. **Full batch SGD**

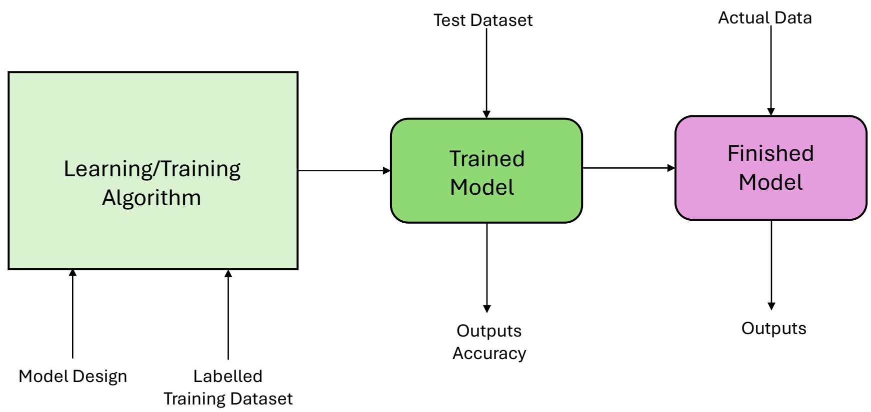
  Do a small batch of data and then do a backward pass. **Mini Batch SGD** (each batch is a power of 2)
- Is there a better way to assess error/loss? **Loss Functions**
- Are there any other weight update mechanisms? **Optimizers** (also manage Learning rate schedules)

  Most of the period from 1986 until now has been spent on studying these questions.

Adam
Backpropagation
Bias
Binary Cross Entropy
Categorical Cross Entropy
Epochs
Exponential
Forward Pass
Full Batch SGD
Gradient Descent
Hyperparameters
Labelled Dataset
Learning Rule
Loss Function
Mean Absolute Error
Mean Squared Error
Mini Batch SGD
Model
Optimizer
Parameters
Relu
RMSProp
SGD
Sigmoid
Softmax
Tanh
True SGD

42

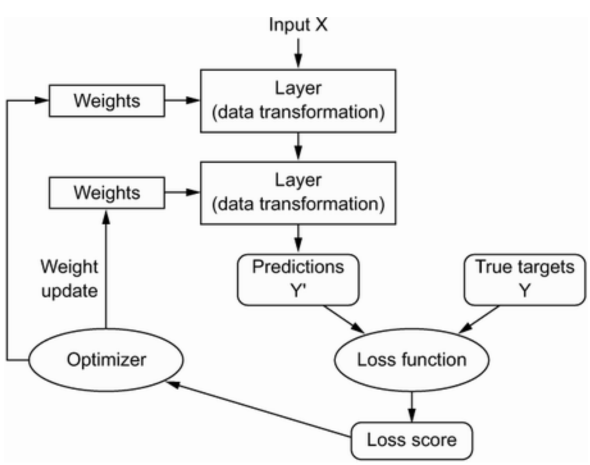
The Learning Paradigm



43

43

How NN Learning Works

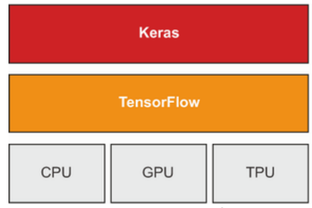


From: Chollet, 2021.

44

44

Introducing Keras



From: Chollet, 2021.

- Deep Learning API for Python (2016-17)
- Built on top of TensorFlow (2015)
- Can run on a typical CPU, or can be accelerated with specialized hardware, if available. GPUs (Graphics Processing Units), TPUs (Tensor Processing Units)
- Makes Neural Network design, implementation, and exploration akin to building with LEGOs!

45

45

Example: Recognizing Handwritten Digits

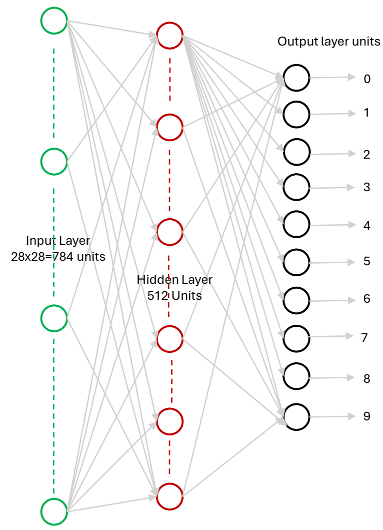
- MNIST Dataset
  70,000 images (28x28 pixels), grayscale values (in range 0 (white) to 255 (black).
- Training set: 60,000 images
- Testing set: 10,000 images
- Task: Given an image, classify it as [0,1,…,9]

46

46

MNIST Digit Recognition: The Design

- **A 3-layer network:** input, hidden, and output layers
- **Input** will be 28x28=784 units
- 10 **outputs**, one for each digit. 512 units in **hidden layer**.
- **Parameters**
  784x512 + 512 (bias) + 512x10 + 10 (Bias) = 407,050
- **Hyperparameters**
  3 Layers
  784 Units in input layer
  512 units in hidden layer (Sigmoid/Relu)
  10 Units in output layer (Softmax)
  Loss: Mean Squared Error/Categorical Cross-Entropy
  Optimizer: (SGD, RMSProp),
  Learning Rate: β (decided by the optimizer)
  Accuracy Metric: Accuracy (% Correct)
  # epochs (? Start with a max of 10)



47

47

Commonsense Baseline Accuracy

- Before we begin, we should always estimate a baseline accuracy.

  That is, given any data set, if we were to randomly assign an output, what would be the accuracy?

  For binary classification, the baseline is 50% accuracy.

  For MNIST Digits classification, the baseline is 10% accuracy.

- Any neural network model we train should be able to perform far better!

48

48

Commonsense Baseline Accuracy

- Before we begin, we should always estimate a baseline accuracy.

  That is, given any data set, if we were to randomly assign an output, what would be the accuracy?

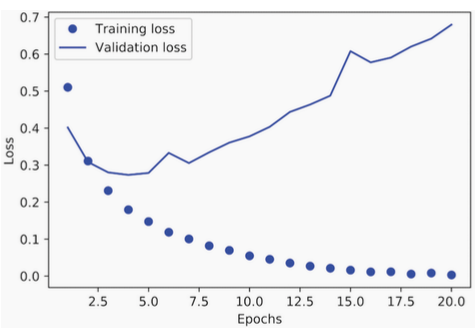
  For binary classification, the baseline is 50% accuracy.

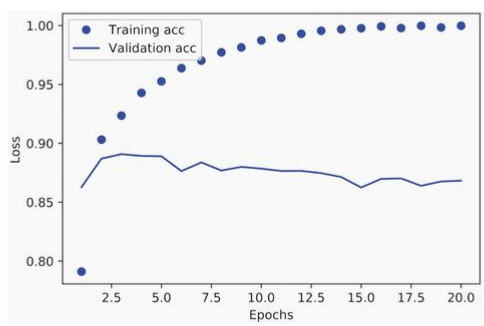
  For MNIST Digits classification, the baseline is 10% accuracy.

- Any neural network model we train should be able to perform far better!

49

Training & Validation Loss and Accuracy



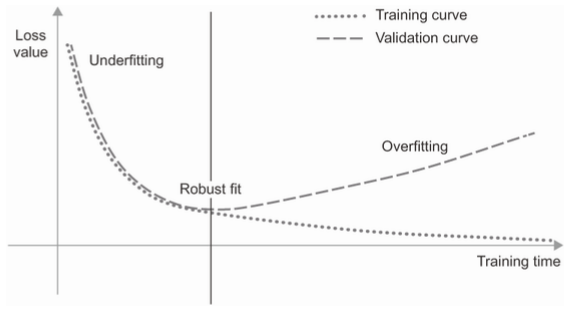


- Notice that the validation loss and validation accuracy both diverge after ~ 4^{th} epoch.
- The model performs better on training data doesn't necessarily do well on the testing data.
- This is called **overfitting**.

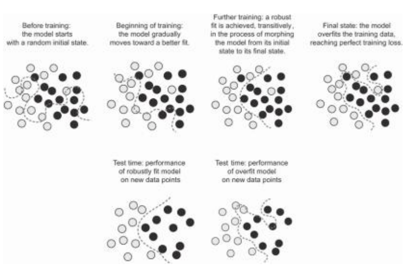
50

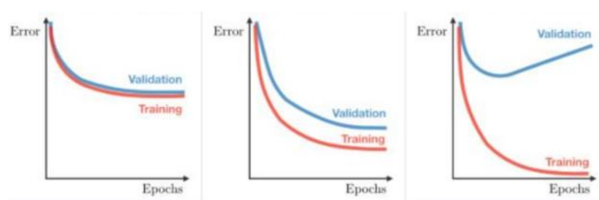
Underfitting and Overfitting

- Initially, as the training proceeds, the lower the loss on training data, the lower the loss on test data. This is **underfitting**. The network hasn't yet modeled the all the patterns in the training data.
- As training proceeds further, the testing stops improving and starts degrading: This is **overfitting**. The network is starting to learn patterns specific to the training data.
- Overfitting can occur when the training data is noisy, ambiguous, or involves uncertainty.



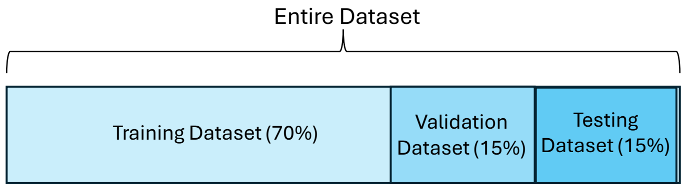
From a Random Model to Overfitting or Robust Fit





From: Chollet, 2021, and
https://medium.com/@datascienceeurope/do-you-know-overfitting-and-underfitting-f27f87ac2f37

Training Data, Validation Data, Testing Data



- **Training dataset** is for use during training
- **Validation dataset** is to estimate loss/accuracy of the model to tune the hyperparameters
- **Testing dataset** is for evaluating the model after training. (how well does it generalize?)

53

53

Backpropagation: Review

Adam
Backpropagation
Bias
Binary Cross Entropy
Categorical Cross Entropy
Commonsense Baseline Accuracy
Epochs
Exponential
Forward Pass
Full Batch SGD
Gradient Descent
Hyperparameters
Keras
Labelled Dataset
Learning Rule
Loss Function
Mean Absolute Error
Mean Squared Error
Mini Batch SGD
Model

Optimizer
Overfitting
Parameters
Relu
RMSProp
Scikit-Learn
SGD
Sigmoid
Softmax
Tanh
Testing Data
Training Accuracy
Training Data
Training Loss
True SGD
Underfitting
Validation Accuracy
Validation Data
Validation Loss

54

54