|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Intelligence | | |  | | | | | perceive -> think/solve/learn -> act |
| AI | | field of study of Intelligent agents (hardware, software or both) | | | | | perceive -> analyze/compute/learn -> act | |
| Landmark achieve-ments | | | IBM Deep Blue vs Gary Kasparov | | | | | 1997 |
| DARPA Grand Challenge on Autonomous Vehicles (travel 240km) | | | | | 2005  2004 (11.78km) -> 2005 (212km) |
| IBM Watson and Jeopardy | | | | | 2011 |
| AlphaGo vs Lee Sedol (Deep learning)  close to 130,000 possible next moves | | | | | 2016 |
| RTS games: OpenAI 5 and DOTA2; AlphaStar and Starcraft 2 | | | | | 2019 |
| AlphaFold: recreate protein structure? (Deep learning) | | | | | 2020 |
| Moore's Law | | | num of transistor on microchips doubles every year | | -> exponential growth in computing resources  (computing power shld reach 1 human brain by 2025) | | | |
| AI | | ML | | | | Deep learning | | |
| Supervised | | |
| unsupervised | | |
| NLP  email spam filter | | | | content extraction | | |
| classification | | |
| machine translation | | |
| question answering | | |
| text generation | | |
| Expert systems  e.g. online medical symptoms checker | | | | human expert -> knowledge engineer -> knowledge base <-> interface engine <-> UI -> user | | |
| Vision | | | | image recognition (IMAGENET) | | |
| machine vision (detect faulty products) | | |
| Audio | | | | speech to text | | |
| text to speech | | |
| Planning | | | | finding shortest path | | |
| Robotics | | | | active cameras (pan tilt zoom) | | |
|  | Computers fast/good at: | | | arithmetic ops, sorting, searching/indexing, remembering instructions, some board games (chess, reversi) | | | | |
| Computers slow/poor at: | | | understanding images/videos, natural language, complex decision making, creative works, emotional intelligence | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opportuni-ties of AI in Arts, Science, Business | | Law of demand: all other factors being equal, as the price of a goods or service , the demand for it will decrease.  Law of supply: ceterus paribus, as price of a goods/service , the qty of the goods/service provided by suppliers will .  When prediction capabilities improve, the price of prediction gets cheaper.  Substitute good: can be used for same purpose -> consumers prefer cheaper good/service  Complementary good: adds value to another good -> will have demand and price increase tgt (data for AI)  Dramatic improvement in predictionaccuracy with approximation methods, particularly deep learning | | | |
| The ImageNet competition in 2012 was distinguished due to its large scale (1000 categories, 1millions pictures)  AlexNet produced results more than 10% higher than its nearest competition. By 2015, AI has exceeded human performance | | | |
| Go – search techniques led AlphaGo defeating human in 2016 using grid to represent board | | | |
| In DOTA 2 (AlphaStar) and Starcraft 2, AI is NOT able to utilise a single image to generate enough information to predict.  In OpenAI5 (Starcraft) and AlphaStar, deep learning with memory capability is used for prediction (reinforcement learning) | | | |
| AlphaFold achieves prediction ability (predicting structure of amino acids) that is competitive with experimental results.  Deep Learning was shown to be comparable to dermatologists in identifying skin cancer in 2017, comparable to pathologist in cancer grading in 2018, comparable to specialists in detecting diabetes in 2019 | | | |
| GPT3 language model is able to generate short passages in the news genre such that humans only have 52% accuracy to differentiate between the generated articles and real articles | | | |
| It is possible for AI to generate a novel image that has never been created before. | | | |
| AI in Organiza-tion | | |  |  |  |  | | --- | --- | --- | --- | | E.g. Banking | Front Office | Middle Office | Back Office | | More Mature | Chatbots | Anti-Fraud & Risk | Credit Underwriting | | KYC/AML | | Less Mature | Voice Assistants | Monitoring | Smart Contracts Infrastructure | | Authentication & Biometric | Complex Legal & Compliance Workflows | | | | |
| Deploying AI | | | Not an isolated decision. Digitalization -> Choose AI tech -> AI Deployment | | |
| Applications of AI in industry | | | | Nike+ Platform. Collect data from Nike shoes + smart devices + online platforms -> personalised service  Virtuous Circle of AI Deployment: More Users -> More Data -> Better App -> More Users | |
| Applica-tions of AI in business analytics | | BA: skills, technologies, applications and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning.  Descriptive Analytics: What happened? Usually data viz + unsupervised learning (clustering, PCA)  Predictive Analytics: What will happen? Usually statistical techniques from data mining, modelling, ML + supervised learning (classification, regression) Prescriptive Analytics: What to do? Using results of descriptive & predictive analytics + search/planning & decision making | | | |
| Prescrip-tive Analytics | Inventory Management: What, when, how much to order?  Production Planning: What, when, how much to produce? | | | | Warehouse Management: What, how to store? What, how to retrieve?  Supply chain optimization: Minimize cost across the supply chain |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Problem solving by search | | | Search: process of looking for a seq of actions that reaches the goal  Input: problem formulations  - state space (representation of environment), action space (possible actions to take), successor fn (modify state space after action taken), start state, goal test, path cost  Output (solution) : seq of actions to goal state  - variants: least-cost path  Executions of the actions  In search tree, root node = start state, child nodes = successor states, edges = actions and cost  Each node in tree represent a state | | | | | Diagram  Description automatically generated |
| Search tree | | Explored set: uniques instances of states already explored  - needs to provide fast insertion and searching (hash table, red-black tree [type of self-balancing binary tree])  Frontier: determines order of nodes to be expanded (queue: FIFO for BFS, stack: LIFO for DFS, priority queue)  Branching factor: max num of child possible for ea node | | | | | algo:  loop do  if frontier empty, return failure  choose leaf node and remove from frontier  if node contains goal state, return corresponding soln  expand chosen node, adding resulting node to frontier | |
| Search algos | | Metrics: - completeness (guarantee of finding soln)  - optimality (guarantee of finding an optimal soln)  - time complexity, space complexity  Algos split into uninformed (blind) & informed search algos  Informed Search:  - can distinguish goal & non-goal state  - can tell which non-goal state is better  - use evaluation/heuristic fn that controls path of search  - Best-first search: Greedy best-first search, A\* search | | | | Uninformed search use only info available in problem definition  - no add info about states beyond that provided in problem defn  - only knows goal vs non-goal state  - distinguished by order in which each node is expanded  - blindly try all possible ways to reach target (inefficient for large num of states)  - BFS, DFS, Uniform-cost search (UCS) | | |
| BFS | | Expand shallowest unexpanded node  Frontier is FIFO queue | | | | Complete: Yes (if branching factor is finite)  Optimal: Yes (if step cost = 1)  Time and space: space is the bigger problem | | |
| UCS | | Weighted BFS: find an optimal soln w any positive step cost  Expands node from frontier w cheapest path cost  Frontier is priority queue | | | | Complete: Yes (if all state's actions have non-negative values)  Optimal: Yes  Time and space: both worse than BFS | | |
| DFS | Expands deepest unexpanded node  Frontier is LIFO stack | | | | Complete: Yes (if state space if finite w no repeated states and paths)  Optimal: No (won't return optimal path on right side of tree, if soln found on left)  Time and space: both better than BFS | | | |
| Backtracking | | | | Only 1 successor generated at a time rather than all successors  Ea partially expanded node remembers which successor to generate next  One a node has been expanded, and all its descendants fully explored, can be removed from memory  Achieved using recursion | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Informed Search | | Best-first Search: use an evaluation fn f(n) for each node  - estimate of "desirability"  - expands most desirable unexpanded node | | | - Order nodes in frontier in decreasing order of desirability  - Greedy best-first search, A\* search are e.g. of best-first search | | |
| Heuristic function | | h(n) is admissible if for every node n, h(n) ≤ h\*(n), where h\*(n) is the true cost to reach the goal state  - admissible heuristic never overestimates cost to reach goal | | | | h(n) is consistent if for every node n, every successor n' of n generated by any action a, h(n) ≤ c(n,a,n') + h(n'), where c(n,a,n') is estimated step cost from n to n' | |
| Greedy Best-First Search | | UCS expands node w cheapest path cost  Greedy best-first search uses heuristic fn h(n) to compute cheapest cost to goal state  - expands node that appears to be closest to goal  E.g. h(n): straight line dist from n to goal state | | | | | Complete: Yes (if state space is finite and no duplicate nodes are expanded)  Optimal: No  Time and Space: exponential if bad heuristic fn used |
| A\* Search | UCS minimizes cost of path, g(n) and finds optimal sol but is inefficient when domain is large  Greedy best-first search minimize search cost to the goal, h(n) by reducing search domain but result is not optimal  A\* search takes advantage of both approaches, f(n) = h(n) + h\*(n), i.e. f(n) = estimated total cost, h\*(n) = actual cost, h(n) = estimated cost | | | | | | Complete: Yes (if there are finite num of nodes w cost ≤ to cost of optimal solution path)  Optimal: Yes, if h(n) is consistent  Time and Space: bad, w space worse than time  h(n): straight line dist; h\*(n): actual cost; g(n): dist from one node to another |
| Optimization | | | , find min value of f(x), where X is state space  x\* = , find optimal value x\* | | | | |
| Previous search mtds: solution based on following some path from initial to goal state (e.g. BFS, DFS, A\*...)  - resource demanding, - finite space search | Local search: path to goal irrelevant, Only need to find goal state  - checks and updates only current state along path  - require very little/constant memory  - find reasonable soln in large/infinite state spaces | | | |
| Local Search | | Start w feasible soln x  Define neighborhood of x  Identify an improved neighbor y  Replace x by y and repeat | | | | | E.g. - Hill-Climbing search (cannot detect local or global min/max),  - Map coloring (can use num of conflicting or non-conflicting regions for optimization fn) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AI that learns | | Humans – learn from experiences  Computers – follow instructions | | | | | | | | | | | | | | ML – make computer learn and make decisions from experiences | | |
| ML | | Traditional Programming: Input data & rules. Output answers | | | | | | | | | | | | | | ML: Input answers & data. Output rules | | |
| Types of ML | | Supervised Learning  (Labeled data) | | | | | | | | | Classification | | | | | | classify email as spam | |
| Regression | | | | | | predict housing price | |
| Unsupervised Learning  (Unlabeled data) | | | | | | | | | Clustering | | | | | | group similar customer profiles | |
| Dimension Reduction | | | | | | find key features | |
| Anomaly Detection | | | | | | credit card fraud detection | |
| Reinforcement Learning | | | | | | | | | Learns by reacting to environment | | | | | | self driving cars | |
| Supervised Learning | | | | Learn patterns in data and build general set of rules to map input to the class or event  Human labeled data is passed as input  Model building has 3 stages: Training, Testing/Validation, Prediction/Classification | | | | | | | | | | | | | | |
| Unsupervised Learning | | | | | | | Find similar patterns that can be grouped into specific classes or events  No human labeling of data required – expensive process | | | | | | | | | | | |
| Reinforcement Learning | | | | | | | | Maps situation to actions that yields maximum rewards  For every action, there is a reward defined by the user | | | | | | | Algo learns to find action that maximizes the rewards | | | |
| Major tasks in ML | | | | | | | | | Classification, Regression & Clustering | | | | | | | | | |
| Types of Data | | Numerical  (Quantitative) | | | | | | | | | | Discrete data | | | | | | num of students in class |
| Continuous data | | | | | | weight of student |
| Categorical  (Qualitative) | | | | | | | | | | Nominal data | | | | | | gender (Male, Female) |
| Ordinal data | | | | | | university grades (A,B,C...) |
| Numerical Data | | | | Expressed as a num and can be measured  Discrete data: involves only integers and cannot subdivide discrete values into parts  Continuous data: can be meaningfully divided into its finer levels. Can be measured on a scale or continuum and can have almost any numeric value | | | | | | | | | | | | | | |
| Categorical Data | | | | | | Can't be expressed as a num and can't be measured. Info is sorted by category  Nominal Data: Data used just for labeling variables, w/o having any quantitative values, i.e. no order  Ordinal Data: values are placed in some order | | | | | | | | | | | | |
| Multi-dimensional Data | | | | | | | | | | 1-D, 2-D, 3-D,... data | | | | | | Use eqn of hyperplane for > 3-D data | | |
| KNN algo | k-Nearest Neighbor: supervised, non-parametric and lazy learning algo  Finds k-most similar datapoints from dataset  Can be used for classification & regression | | | | | | | | | | | | | Supervised – needs labeled training data  Non-parametric – no assumption made on data dist  Lazy learning – no need for training data, hence no model generated | | | | |
| Works based on majority vote of the k-nearest neighbors class  New datapoint assigned to most common class of k-nearest neighbors | | | | | | | | | | | | | | | | | |
| 1) Determine k = num of nearest neighbors  2) Calculate dist btw new data and all training data  3) Sort distance and determine nearest neighbors based on the k min distances  4) Determine class value of these k-nearest neighbors  5) Use simple majority of class of nearest neighbors as predicted class of new data | | | | | | | | | | | | | | | | | |
| Similarity Measures | | | Dist measure w dimensions representing features of the objs  If dist is small, features are having a high deg of similarity | | | | | | | | | | | | | Similarity measured in range [0,1] (aka similarity score)  Similarity score of X and Y = 1: X = Y  Similarity score of X and Y = 0: X ≠ Y | | |
| Types of Similarity Measures | | | For 2 n-dimensional data points p and q, | | | | | | | | | | | | | | | |
| Euclidean Dist: | | | | | | | | | | Minkowski Dist: , where m is the order of the norm | | | | | |
| Manhattan Dist: , where pi and qi are real values | | | | | | | | | | Hamming Dist: , where pi and qi are binary values | | | | | |
| Feature Scaling | | Most dataset will contain features highly varying in magnitudes, units and range  ML algos operates on numeric values of data – dist measures & optimization  Since algo takes only magnitude of features (neglecting units), there is a chance that higher weightage is given to features w higher magnitude | | | | | | | | | | | | | | | | |
| Feature Scaling Mtds | | | | | x' is scaled value of x | | | | | | | | | | | | | |
| Standardization: x' = , [-1,1] | | | | | | | | | | | Normalization: x' = , [0,1] | | |
| k-value | | No structued method to find k-value  Smaller k-value – sensitive to noisy data  Larger k-value – more computationally expensive | | | | | | | | | | | | | | Rule of thumb: k = sqrt(N), N = number of training data  Try to keep k-value odd to avoid tie btw 2 classes | | |
| Pros and Cons | | Pros: - algo simple and easy to implement  - no training required (new data can be added at any time) | | | | | | | | | | | | Cons: - complexity (memory and computation time is higher)  - does not work well w large datasets  - does not work well w high dimension | | | | |
|  | | from sklearn.neighbors import KNeighborsClassifier  knn = KNeighborsClassifier(n\_neighbors = 3)  knn.fit(x,y) | | | | | | | | | | | | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Linear Regression | | Linear Regression is a supervised ML algo: finds best-fit linear line btw indep (X) and dependent vars (y) (i.e. find the linear r/s btw vars)  Fitted value for 1 indep var: = w0 + w1x | | | | |
| Error function | | Best-fit line for given data is one that minimizes the error btw the given data and the predicted value for all the data pts  E(w0, w1) = | | | | |
| Gradient Descent Algo | Find optimal params =      Linear Regression Model (LRM):  Feature scaling required for GD | | | 1. Start w random value for (w0, w1). Let L be learning rate that controls the rate of change of values of (w0, w1) at ea step. Set L = 0.00001  2. Calculate partial derivative of the error fn w.r.t w0 and w1. Use current values of x,y,w0 and w1 to calculate this derivative value  3. Update current values of (w0, w1)  w0 := w0 – L, w1 := w1 – L  4. Repeat step 2 & 3 until convergence (i.e. error fn is minimized to small value or 0). Resulting values are the optimum values of param | | |
| Convergence & Learning Rate | | Convergence: Run algo for fixed num of iterations or when convergence condition is achieved: | | | | Learning Rate: if L is too large, algo can overshoot minimum, causing algo to fail to converge. If L is too small, algo takes more time to converge |
| Prediction | | Use to predict for new data | | | | |
| Python Code | | from numpy import genfromtxt  from sklearn.linear\_model import LinearRegression  data = genfromtxt('.csv', delimiter=".", skip\_header=1)  X = data[:, :1]  y = data[:, 1:2] | | | | model = LinearRegression()  model.fit(X, y)  newx = np.array([1,2,3,4]).reshape((-1,1))  y\_pred = model.predict(newx)  print(model.intercept\_, model.coef\_) |
| Multivariate Linear Regression | | | Multiple indep vars and 1 dependent var  = w0 + w1x1 + w2x2 + ... + wdxd = = **Wx**T | | | Note x0 = 1 (aka bias)  **W**, **x**: row vectors |
| Error function | | E(**W**) =  Can be found using Gradient Descent algo | | | **W**\* = = = | |
| Gradient Descent Algo | | **W** := random values or zeros [0 0 ... 0]  repeat until convergence:  for each wj in **W** do:  wj := wj + L | | | | is x value of ith observation and jth covariate  **W**\* = , optimal params  OR **W** = **X**T(**y** –  **XW**), X is mxd, W is dx1 here |
| Prediction | | = **W**\***x**T | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Logistic Regression | Supervised ML algo  Classification algo – Assigns data to discrete set of classes  Email spam, Fraud, Credit approval, | Transforms its output using the logistic fn to return a prob value  Output of logistic regression is 0 or 1 | |
| Hypothesis function | Hypothesis/sigmoid fn: = g(**Wx**T), where g(z) =  Hypothesis fn give prob value | Conventional: | |
| Error function | Same error fn in linear regression would give many local minima for logistic regression  So need to use cost fn: cost(, ) = OR cost(, ) =  Error fn for logistic regression: E(**W**) =  Then use gradient fn to minimize error fn:  **W**\* = = = | | |
| Gradient Descent | **W** := random values or zeros [0 0 ... 0]  repeat until convergence:  for each wj in **W** do:  wj := wj + L |  | |
| Prediction | = g(**W\*x**T) = , then convert this value to 0 or 1 |  | |
| Python code | from sklearn.model\_selection import train\_test\_split  xtrain, xtest, ytrain, ytest = train\_test\_split(xvals, yvals, test\_size=0.25)  from sklearn.preprocessing import StandardScaler  std\_x = StandardScaler()  xtrain = std\_x.fit\_transform(xtrain)  xtest = std\_x.transform(xtest)  from sklearn.linear\_model import LogisticRegression  classifier = LogisticRegression()  classifier.fit(xtrain, ytrain)  y\_pred = classifier.predict(xtest) | | data = genfromtxt('file', delimiter="", skip\_header=1)  xvals = data[:, 0:5]  yvals = data[:, 5]  accuracy =  precision =  recall/TPR =   |  |  |  |  | | --- | --- | --- | --- | |  | | Predicted class | | | 1 | 0 | | Actual class | 1 | TP | FN | | 0 | FP | TN | |
| from sklearn.metrics import confusion\_matrix, accuracy\_score  print(confusion\_matrix(ytest, y\_pred)  print(accuracy\_score(ytest, y\_pred)) | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Unsupervised learning | Supervised: labeled output; Unsupervised: no labeled data  Aim: find similar patterns to group into specific classes/events  OR identify and extract similarities btw input data so that similar data can be categorized tgt | | Types: Clustering (group similar customer profiles)  Dimensionality Reduction (find key features)  Anomaly Detection (fraud detection)  Association Rule Mining (buy new house -> buy new furniture) | | | |
| Clustering | Each datapoint is given a cluster-id  Group data w same patterns using similarity measure (feature scaling required)  K-means algo: centroid based clustering algo | | | | | |
| K-Means Clustering algo | Input: K, Training data  Output: training data points labeled w cluster-id  Random initialization: randomly pick K training samples to initialise as centroids  K < N | 1. Randomly initialize K-cluster centroids (M1, M2, ..., Mk)  2. Repeat: Cluster assignment step: Assign data points to closest cluster center  for i = 1 to N: Ci = {index (from 1 to K) of cluster centroid closest to xi}  Move centroid step: Change cluster center to the average of its assigned points  for k = 1 to K: Mk = {average of points assigned to cluster k}  3. Stop when no change in points' assignment | | | | |
| Optimization Objective | E(C1,...,CN, M1,...,Mk) = (squared Euclidean Distance) | | | | | Cluster assignment step:  Moving Centroid: |
| How to choose K? | 1) Elbow method  2) Choose K manually (when we know how many clusters there should be) | | | | error = []  for i in range(1,11):  km = KMeans(n\_clusters = i, init = 'random', max\_iter = 300)  km.fit(X)  error.append(km.inertia\_)  plt.plot(range(1,11), error) | |
| Python | from sklearn.datasets import make\_blobs  X, y = make\_blobs(n\_samples=200, n\_features=2,centers=4, cluster\_std=0.5, shuffle = True)  plt.scatter(X[:,0], X[:,1]) | | | from sklearn.cluster import KMeans  km = KMeans(n\_clusters = 3, init='random', max\_iter = 300)  cluster\_ids = km.fit\_predict(X) | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Biological  Neural Network | | Neuron contain dendrites, cell body and axon  Dendrites receive input from other neurons  Cell body aggregate the inputs, if inputs exceed some threshold, then will fire another signal through axon | | ANN is inspired by biological learning system: inter-connection of neurons in brain  Human brain contains billion of neurons, around 1011 neuron. Each neuron is connected to thousands of other neurons | | |
| ALVINN | | Autonomous Land Vehicle In a Neural Network, 1989. Input from camera of 30x32 pixels, goes through 4 hidden units, then get 30 outputs for steering. Able to travel at 70mph | | | | |
| Neuron in Neural Network | | Input [x0 x1 x2 x3]. Data may only contain x1, x2 and x3. x0 is bias and usually input as 1  Weights [w0 w1 w2 w2]  For each neuron in hidden and output layer, Calculate , then place value in activation fn, = output of 1 neuron  Aggregate output layer to get prediction,  Error = – y  Each layer has same neuron functions/same functionality/same activation fn | | | | |
| Activation fns | | Helps to solve complex non-linear model  Common activation fns: step fn, sigmoid fn, ReLu (Rectified Linear Units), Leaky ReLu (Leaky Rectified Linear Units) | Graphical user interface, diagram, application  Description automatically generated | | | |
| Single-layer NN (Perceptron) | | Perceptron takes a vector of real-valued inputs, calculates a LC of these inputs, put it through activation fn, then output 1 if result > a threshold, 0 otherwise  Weights are adjusted based on the predicted output and actual output y using e.g. gradient descent algo | | | | |
| Multi-layer NN | | Input layer (no computations) -> Hidden layer (1 or more) -> Output layer  Note there are bias added to hidden layer and output layer. And all densely related (i.e. all inputs goes to next layer)  Feedforward: get 1 labeled training data -> input layer -> comput values for hidden layer using input layer & weights -> compute values for output layer using hidden layer as input & weights -> output layer is output of NN  Backpropagation: Cycle through training samples. If output of NN correct, no changes made to weights, else if error -> weights adjusted to reduce error | | | | |
|  | | Diagram  Description automatically generated | | | Algo: initialize random values for weights in all layers  repeat until convergence:  for each training datapoint:  propagate inputs forward to comput outputs  propagate deltas backwards from output layer to input layer  update every weight in network using deltas | |
| Python | model = keras.models.Sequential()  model.add(keras.layers.Flatten(input\_shape=[28,28]))  model.add(keras.layers.Dense(300,activation="relu"))  model.add(keras.layers.Dense(100,activation="relu"))  model.add(keras.layers.Dense(10,activation="softmax"))  model.compile(loss="sparse\_categorical\_crossentropy", optimizer="sgd", metrics=["accuracy"])  history = model.fit(X\_train, y\_train, epochs = 30, validation\_data = (X\_valid, y\_valid)) | | | | | import tensorflow as tf  from tensorflow import keras  model.summary()  #softmax for multiclass probability similar to logistic |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Deep Learning | | | | DL has More hidden layers & More types of layers than NN.  DL is 1 of the branches of ML – operations applied one after the other – mathematical framework to learn representations from data  These operations are structured into modules called layers. DL has successive layers of representation  Layers are parameterized by weights (parameters learned during training). Process of learning consists of finding "good values" for these weights, i.e. values that minimize loss fn  DL is a distillation process, to extract the most relevant info | | | |
| Limitations of ML | | | | Feature extraction is challenging for complex problems like object recognition  Face Detection in ML: Have to manually define facial features like eye, ears, nose, lips,... and ML program will identify which more ppl  Face Detection in DL: Will automatically find out features which are impt for face detection, through large amt of data  Note feature scaling still required as GD used to find weights | | | |
| Why DL now? | | | | Now have better algos and understanding. More powerful computing infrastructures – GPU (graphical processing unit), TPU (tensor processing unit), Cloud,...  Huge amt of data – labelled data from smartphones & sensors  More open source tools (keras, pytorch, tensorflow) and models (pretrained models; open for use) | | | |
| Computer Vision | | | | Applications: Object recognition, image segmentation, style transfer, automatic coloring, image inpainting (repairing), image super-resolution, image synthesis  CV is earliest and biggest success of DL. They power image search services, self-driving cars, auto video/image classification sys,... Convolutional NN (CNNs) is DL framework for computer vision tasks – convnets  ANN learn from global features while CNN learn from local features | | | |
| CNN | | | | 1. Pattern they learn are translation-invariant. E.g. after learning a certain pattern in lower-right corner of picture, CNN can recognize same pattern at another place, e.g. upper-left corner | | | 2. CNN can learn spatial hierachies of patters. E.g. 1st layer learn small local patterns like edges, 2nd layer learn larger patterns from 1st layers like line, 3rd layer: shape,... |
| CNN architecture | | | | AlexNet (2012). Input 64 x 64 x 3 (RGB) -> hidden layers (convolution/filtering) -> Dense -> outputDiagram  Description automatically generated | | VGGNet (2014)  Diagram  Description automatically generated with low confidence | |
| Convolution | | | | Convolution Layer – series of filters known as conolution kernels.  e.g. keras.layers.Conv2D(filters=32, kernel\_size=3, strides=1, padding="same", activation="relu")  Filter/Kernel – matrix of ints that are used on a subset of input pixel values, same size as kernel. Each pixel is multiplied by the corresponding values in the kernel, then result is summed up for a single value for simplicity representing a grid cell, in the output channel/feature map. The ints are trainable variables which NN can learn which filters work best  Input Images – 3 channel RGB or 1 channel Greyscale image  Convolution Ops – kernel strides over the input matrix of nums moving horizontally col by col, sliding/scanning over 1st rows in input matrix pixel values. Then kernel strides down vertically to subsequent rows  Padding – handle edge pixels {valid: no padding, same: pad with 0s}  Pooling Layer – to reduce computational load, memory usage and num of params in network. No trainable weights in this layer, empty sliding window. Uses max or mean aggregate functions | | | |
| Keras code (CNN) | | | model.add(tf.keras.layers.Conv2D(filters=32, kernal\_size=(3,3), strides=(1,1), padding=“valid”, activation = “relu”, input\_shape=(28,28,1)))  model.add(tf.keras.layers.MaxPooling2D(pool\_size=(2,2), strides=(2,2)))  model.add(tf.keras.layers.Flatten())  model.add(tf.keras.layers.Dense(units=128, activation=“relu”)) #units: output size  model.add(tf.keras.layers.Dense(units=10, activation=“softmax”)) | | | | |
| NLP | | | | Applications: autocomplete, translation, social media analytics, chatbot, document summarization, voice assistant, spam filtering, grammer correction. Google search engine, Amazon voice assistant (Alexa), Apple Siri, auto text generation  NLP deals w building computational algo to auto analyze and represent human language  Useful to teach machines ability to perform complex natural language related tasks such as machine translation and dialogue generation | | | |
| Recurrent NN (RNN) | | Commonly used in time series analysis, document classification, text/speech generation  Designed for sequence problems (e.g. predicting next word in sentence) Predictions made by RNN dependent on previous predictions, i.e. output of node is feedback as an input to the same node w the next input  Limitations: RNN okay if words have short-term dependencies (short sentences), but fails if word has long-term dependencies  Might also have vanishing/exploding gradient problems in RNN | | | | | |
| LSTM | Diagram  Description automatically generatedLong Short-term Memory networks have memory blocks that are connected into layers  LSTM consist of 3 gates (input, forget and output gates) and calculate hidden state through a combination of the 3  Forget Gate: decide which info to discard that are not impt from previous time step  Input Gate: decide which infrom from input to be used to update the memory state  Output Gate: decide what to output based on input and memory of the unit | | | | | | |
| Text Representation | | | | | A picture containing text  Description automatically generated1. 1-hot encoding  - sparse  - high dimensional  - hardcoded | | A picture containing diagram  Description automatically generated2. Word Embedding  - dense  - lower-dimensional  - learned from data |
| Keras code (LSTM) | | | | from keras.layers import LSTM; from keras.layers import Embedding; from keras.preprocessing import sequence  sequence.pad\_sequence(X, maxlen=)  model=Sequential()  model.add(Embedding(top\_words, embedding\_vector\_length, input\_length))  model.add(LSTM(100))  model.add(Dense(1, activation=“sigmoid”))  model.compile(loss=“binary\_crossentropy”, optimizer=“adam”, metrics=[“accuracy”]) | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data privacy | Ethics and Regulation: GDPR Rights of Individuals over their Personal Data  - Right of access, - Right to rectification, - Right to erasure, - Right to restrict processing, - Right to data portability, - Right to object, - Right not to be subject to a decision based solely on automated processing | | | | | | | |
| Privacy – many apps keep track of your location | | | | | | | |
| Anonymization – not fool proof. (Removal of names in medical data from Massachussetts hospitals, but combined w Voter roll data, could identify people) | | | | | | | |
| Differential Privacy: Add noise to dataset and compute, publish dataset.  More noise added -> less infomation leakage but less accurate | | | | With differential privacy, noise is added s.t. output of algo is approximately the same  Data has composibility + robustness to post processing | | | |
| Fairness | Bias in Data: more male applicants -> Amazon AI recruting showed bias against women | | | | | | | |
| Statistical Parity: Should be equal across different groups | | | | | | | |
| Trade-off btw fairness and accuracy: | | If fair subset includes optimum -> still optimal  If fair subset does not include optimum -> have to decide trade-off | | | | | Chart, scatter chart  Description automatically generated Pareto Optimality |
| Equal Opportunity: To maintain same percentage across groups, an eligible person may be denied to maintain statistical parity | | | | | | | |
| Interpre-table models | Linear/logistic Regression, Decision tree – intepretable models | | | | | | | |
| Model Agnostic Explanations – can explain both blackbox and whitebox model | | | Examine effects of each feature in learned model through partial dependence plot (plot output of learned function for diff values of the specified feature, averaged over the training set) | | | | |
| 1) Global surrogate methods: approximate black box model w interpretable model (e.g. train linear morel on training data labelled w output of black box model). Interpret the surrogate model | | | | | 2) Local interpretable model-agnostic explanations (LIME): select instance of interest. Perturb the data around the instance to get new dataset. Weight data according to distance from instance of interest. Train interpretable model | | |
| Counterfactual explanation: describes smallest change in feature values that changes the prediction to a predefined value  - Find an input x' that minimizes a weighted sum of loss btw the output f(x') and a counterfactual target t and the dist btw x' and the actual input x | | | | | | Adversarial e.g.: find noise n that minimizes l(f(x+n),t) + a|n|, where l is a loss fn, t is adversarial target, and |n| is size of noise n | |
| Undesirable effects of AI | | Malicious use of AI: expansion of existing threats & introduction of new threats | | | | | | |
| Physical security, digital security, political security (personalized propaganda), | | | | | | |
| Vienna Manifesto on Digital Humanism | | | | | | |
| Unintended side effects: Waze directing drivers to fire due to "less traffic" | | | | | | |
| Singularity, Asimov (1. Don't harm humans, 2. Obey orders, 3. Protect yourself), Slow/fast takeoff | | | | | | |
|  | Jobs that can be substituted by AI will disappear  Demand will increase for skills that are complementary to those provided by AI  Human-AI teams will form if they are more cost-effective than using human workers alone  Inequality due to different skillsets – to tackle this: Universal Basic Income, Life-long education  Big tech companies and certain countries might have monopoly | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Weakness of Humans | Division of Labour -> greatly increase productivity | | | | | |
| Human weakness | Humans are Poor statisticians | Unable to assess probabilities of events accurately | | | |
| Even w accurate prob, ofter make poor decision (Bias coin, P(head) = 0.6.)  Predict 0.6 head, 0.4 tail -> 0.52 accuracy but if we just Predict 1 head -> 0.6 accuracy | | | |
| Hospital 1: 45 births/day. H 2: 15 births/day. Which hospital would have more days when ≥ 60% of births are boys. Ans: Hospital 2 as hospital 1 tends to 50% due to law of large num | | | |
| Framing Problems | Survival rate is 90% -> 84% choose surgery  10% mortality rate -> 50% choose surgery | | | |
| Inconsistency | Even w same info, can still predict inconsistently. Doctors contradict themselves 1 in 5 times on X-rays | | | |
| Weakness of AI | Current AI are narrow AI and not artifical general intelligence (AGI) | | | | | |
| Humans better at: | Learning from small amt of data | Current AI techniques require large amt of training data for good performance  E.g. Humans can cook using new recipe after small num of demonstrations, play new games after few trials  Humans can integrate all experience over lifetimes whereas ML often done in isolation on task of interest | | | |
| Causal Inference | Association does not imply causation. | | | |
| Gary Kasparow early chess program immediately sacrificed queen as it learned that doing so leads to winning grandmaster games | | | |
| Simpson's paradox: phenomenon where association btw 2 variables disappears or reverses itself when the pop is divided into subpopulation | | | |  |  |  | | --- | --- | --- | | Kidney Stone | Treatment A | B | | Large stones | **93%** | 87% | | Small stones | **73%** | 69% | | Both | 78% | **83%** | |
| Low prices associated w low sales in business  But Economist assumes higher prices would cause demand to fall (law of demand)  When time is taken into account, the direction of association btw demand and prices may change | | | |
| Estimating utility | Credit card company trying to predict whether transaction if fradulent  Expected utility of decline: pu1 + (1-p)u2 . Expected utility of accept: pu3 + (1-p)u4  Optimal action is action w higher expected utility | | | |
|  | Human judgement usually required in developing ML methods. E.g. classification problem w huge unbalance in data. Predicting negative all the time would have 97% accuracy but is useless  Often tradeoff btw sensitivity and specificity or tradeoff btw precision and recall | | | TPR = recall = sensitivity = TP/Positive  TNR = specificity = TN/Negative  Precision = TP/Prediced Positive | | |
| Human-AI team can complement each other's weakness -> better overall performance.  Machine do initial prediction -> human combine initial prediction w own assesment | | | | E.g. search engine rank search result -> human look through result and decides if relevant | |